EPISODE 1426

[INTRODUCTION]

[00:00:00] JM: A data catalog provides an index into the datasets and schemas of a company. Data teams are growing in size and more companies than ever have a data team, so the market for data catalog is larger than ever. Amundsen is a data catalog that came out of Lyft. We've previously explored the basics of Amundsen. In today's episode, Mark Grover returns to the show to talk about the art and science and usage of data catalogs. He also talks more about the engineering of Amundsen and the hosted version of Amundsen, which he's building at Stemma.

If you're interested in reaching our audience, we reach more than 250,000 developers per month. You can contact us for sponsorship details at sponsor@softwareengineeringdaily.com.

[INTERVIEW]

[00:00:45] JM: Mark, welcome back to the show.

[00:00:47] MG: Thank you for having me. Super glad to be back.

[00:00:50] JM: We've done a few shows about Stemma or Amundsen more appropriately. But for people who aren't familiar with those episodes, or who just need a refresher, can you start us off with just a description of what a data catalog is and what problem it solves?

[00:01:07] MG: Yeah, so the problem here is that organizations over the last few years, let alone a decade have collected a ton of data within the company. And in fact, innovation in various different data technology has only worsened this problem, right? So, it's really easy to stand up a data warehouse, whether that be Snowflake, RedShift or BigQuery. It's really easy to get data into that warehouse to Fivetran or Airbyte. Very easy to process that data to Airflow, dbt, things of that sort, and then consume that data through insights analytics, like Tableau, Looker and Mode, right? So, that's one category. We've created a bunch of innovation, that gets us to a lot of data.

Second thing we've done is we've said all organizations are data driven. So not only is a data scientist job to use data, but a product manager’s job and operational manager’s job, marketing manager, so on and so forth. The list goes on. And this has led to a unique chaos in companies around data, and the chaos is a lot of people want to use data, there's a lot of data and no one's got any clue what's out there? What do I use? What's the source of truth? What's trustworthy? And how do I use it? And that problem is what I call lack of trust in data. And the solution to that problem is a data catalog, which you can think of it as a Google search engine for data within your organization. So, there's a PageRank style algorithm that uses metadata from within your data warehouse, your BI tools, and provides a search, understanding ability, who's using it, what data we have, what dashboards are built on top of it, what conversations in Slack are happening that are related to the data, so on and so forth.

[00:02:56] JM: Let's talk a little bit about the data structure or the backing store that this data catalog is actually holding everything in and the process of ingesting that data. So, it's easy to imagine this data in all these disparate areas, but the actual ingest process sounds kind of complicated, because there's so many disparate data sources. Describe the process of enumerating all these data sources and how they get normalized, and what storage format or system they get put into.

[00:03:33] MG: Totally Yeah. So, I'll start at an abstract level, and then give you concrete examples of two data catalogs that I have helped create. So, at the abstract level, your data essentially forms a graph. And in this graph are tables that have columns in them. Their columns get used in dashboards. Those dashboards are owned by people. Those peoples are teams and hierarchies themselves. And then you can add in usage information so that this particular individual often queries this particular data. You can add SLA information this particular table should get delivered by 9 AM. You can add delivery information. This often gets delivered at 9:02 AM. Things of that sort, right?

So, that's the centerpiece, it's a back-end store that essentially stores this graph. Now mind you, just because it's a graph doesn't mean you have to use a graph database, though, it may be an interesting choice, depending on the usage patterns that you're queuing it. So that's the core centerpiece, you can put API's to consume this data. But a pretty common way to consume this metadata is to have a user interface on top. There's a user interface, which you can click on essentially any entity and entity being a table or a dashboard, and being able to read what is in that table, various different profile information about the table, who often queries that table? When was that table last updated? Who are the owners of that table? Things of that sort.

Now, in addition to this central store, and we've talked about this user interface, you need this metadata to be ingested. And this is what the thing you were talking about Jeff, this gnarly thing, where I have to go read my data warehouses information, my BI tools information, my ETL tools information to get this metadata. And so, you're right. At the highest level, I've seen data catalogs integrate with five different kinds of systems to get interesting metadata. The first one is a data warehouse. So, what you get is information schema style information from the data warehouse. So, you get names of columns, the names of tables, the schemas, databases, they're in.

Second one within this category is query logs. This, by the way, is a hidden source of really great information. You can take these query logs and parse them out. What are the common data sets that get joined? How often does a data set actually get updated every day? Who are the people who are querying this dataset? Where are the queries coming from? Is it a service account? Is it an individual? This list goes on and on. Rich set of information comes from the query logs. So, that's just the data warehouse.

The second is BI tool, right? What dashboards exists. And then you can take those queries and you can parse them and connect them to the data warehouse. So, what dashboards are coming from what tables, and the other way around. What tables lead to what dashboard? So on and so forth. Third system is ETL tool or transformation tools. I use this term pretty broadly, it can include a traditional ETL tool, like Airflow, Prefect, Daxter. It can also include things like dbt. What do you get from this information? Its ETL information. What data sets are being produced from what? Who are the people who get notified when this job fails? What was the status of the last time this job ran?

The fourth category is collaboration and communication tools. And so, an example of this is Slack. Often in organizations, you will find that there is a lot of chatter around data on Slack. You have to understand like, what are these conversations about and link these conversations to their data catalog, so they don't get lost. Similar things, you may have Jira tickets that are being filed around data, but connecting them to a different dashboard, or a table that they're linked to is a pretty interesting piece of information.

And number five is HR system. A pretty important part of the data catalog that often gets forgotten is who are the people and how are they organized, right? The fact that my teammate and the same team uses this particular dashboard every day is a pretty interesting piece of information. And yet, we continue to just focus on the tables and think of all employees in the company as equals. Oftentimes, the fact that a data analyst using it versus an operations manager using it can create a very different need or search for the dataset based on who is searching for it. So, that's where we can ingest some team information and influence it that way as well.

Now, I can go to the various different implementations for Amundsen, the open source data catalog I created or Stemma, but I'll just pause here to see whichever direction you want to take it in.

[00:08:13] JM: I'd love to know a little bit more about that ingestion process and how much work went into being able to consume all those different sources and normalize it or are there off the shelf tools for consuming and normalizing those different sources?

[00:08:32] MG: Yeah. So, a little bit of background here is that I started Stemma in 2020. Part of that I was at Lyft, where I created this Amundsen project with my team, which was an open source data catalog. So, I'll start by sharing the experience with Amundsen. Amundsen was started at Lyft as an automated data catalog and its primary goal was to serve a data analyst and data scientist audience, and it was pretty successful at Lyft. 750 users every week, 80% of their data scientists, data analysts and data engineers used it every week to search, discover, and understand data.

In terms of the integration and how much work that was to bring in that data and normalize, there's an ETL framework in Amundsen, that's written in Python called Data Builder. So essentially, in order to get information from your dashboarding system, or your data warehouse, you have to write a new data filter job for a new kind of system. So, you bring in Snowflake. There needs to be data for a job for that. You can configure it and then ingest your metadata, and so on and so forth. Writing a Data Builder job is writing code. It's in Python, but it's also not as simple as punching in your credentials and just having the job run.

So, one of the things that we've done in Stemma, for example, is we've taken all that complexity behind the scenes. So, in Stemma, you go to an admin console, you put in your Snowflake public key, and Stemma is able to pull the appropriate metadata from Snowflake through a service account that you created. But someone has to do the dirty work, right? In one case, that's you, in the other case, that's your provider of the data catalog. The same story kind of exists for other systems, too. There's a Data Builder job that pulls in team information. There's a Data Builder job that pulls in information from your BI tool’s API's to get information about what dashboards exist to commonly views it, so on and so forth.

[00:10:34] JM: Awesome. So, we've given a basic overview of how data gets into Amundsen. You touched on a little bit the backing database of using Neo4j, can you just clarify why a graph database make sense as the backing storage system for a data catalog?

[00:10:56] MG: Yeah. So, that is correct. For that core centerpiece in the open source project that we created at Lyft, Amundsen, it uses Neo4j in the back. And it made sense because we thought off all the metadata we were storing as a graph. And we thought that the traversal of the graph would be one of the key ways in which that metadata gets accessed. That was the primary reason. I'm being honest with you, what we have learned over time is that it is pretty clear that the data being represented as a graph, but often the traversal patterns off that graph are not most graph like, right? So, compare compare that to a RDBMS. We could store that there and we could have a graph stored in RDBMS, but there are certain styles of traversal patterns, which will not perform very well in the relational database. And we found that the patterns, the cost we were paying for storing in this Neo4j graph, we were not taking advantage of it in the patterns.

So, like one thing that the community has done is its created different connectors and made this particular back end swappable. So, the first back end for the project was Neo4j, and then now it supports a relational backend, and also supports Apache Atlas back end. And that way it's compatible with Apache Atlas, which is an older data governance project from the Hadoop world started by Hortonworks, now Cloudera.

[00:12:23] JM: Are there in particular – can you give what kinds of queries are more efficient, or like for people who choose to use a relational back end versus using a graph back end, I mean, for those who might be confused, like the database back end doesn't actually matter, from the user's perspective, but at least from the UI, but you might have different results in terms of, I guess, maybe mainly latency. And in any case, isn't the users just mostly issuing queries through Elastic Search, right? But then Elastic Search hits the backing database? Is that how it works?

[00:13:01] MG: No, this is a part we didn't touch. So, a common pattern for getting to this data catalog is you start with search. You can search for something, and that search takes you to a table detail page or a dashboard detail page, which then you can use to traverse the rest of the graph. So, a fourth piece, which we haven't discussed thus far, is a search engine. And the search engine is decoupled – from from an access perspective, is decoupled from the graph back end. And there is obviously exchange of information and consistency that's happening between these two systems. But they store sort of similar information, but in different forms. The search back end is an inverted index, while the main back end stores graph-based information.

So, during any particular use case, we usually end up accessing both those because it starts with a search, and then it traverses the graph to get some information.

[00:13:57] JM: Okay. So, are there particular query patterns that where it actually matters if the data is in a relational format or a graph? It seems to me like all these queries would be really not that results intensive. But I don't know. Does it matter that much? Are these in like data intensive queries?

[00:14:18] MG: I think you're in the right track. I don't think it matters that much. And it's actually one of the things we're changing in Stemma is to have a relational backend. And the benefit of a relational back end is that you can do versioning off the data, the various fields, and those are harder to do in the graph database, because it's not a schemafide. And so, I'm with you, I think it was a choice that was made but in retrospect, perhaps not the choice that will – the query patterns aren't demanding of that choice as we anticipated it wrong.

[00:14:57] JM: So, I guess one thing we should point out is, as far as I understand, what's being indexed in that backing database is the metadata of your datasets. It's not actually, you're not indexing all of the data in your entire data lake. So, it's really more like this entire system is like a metadata index over all your data across your company, not necessarily like all of the data itself.

[00:15:31] MG: Yup, 100% right.

[00:15:34] JM: So, in terms of enriching that data, the metadata, so not all of these data sources are going to be highly readable or highly consumable by an analyst or a data scientist. So, who's responsible for getting that data tagged or enriched? Such that it's more accessible and more readable? Or is that often necessary?

[00:15:58] MG: Yeah, it is often necessary to augment this information. Let me share how that has evolved over the years. So, if we were talking five, seven years ago, this was a problem that only exists in very large companies, and they had this team called a data governance team. And this data governance team is responsible for ensuring that the augmentation of the metadata is happening, right? What I mean by augmentation is like, we may scrape all this metadata from all these five different places. But then, what you would find is like, humans still need to put information like, what does this particular field mean? Or what's the grain of this table? Or what is the description of this column, so on and so forth.

So, what the data governance team would do is that then they would appoint these data stewards, which are then embedded in the organization. Now, these people are often full-time jobs, who are embedded in various parts of the company, to then document this data. And essentially, it looks like a Wikipedia style exercise, like where you're just trying to fill in the organization's data Wikipedia internally. We'll talk more about this a little bit later. But these efforts failed. And the reason is, this data steward would often have no context about the data. So, what they're doing is they're essentially getting other people in the organization to put in information that they have in their head, right?

So, Mark, the data steward is asking Jeff, the data engineer in the sales data domain, to go put information about sales. And what's changed now is that this information is handled to automation to get as much of it as possible, right? And so, that means that you can get information from dbt Docs. You can get information from like existing descriptions, from Snowflake comments, from Looker dashboards, so on and so forth. But still, there's tribal knowledge in people's head. And the idea is not to centralize that information from one certain team, but it is to democratize and get those humans that have that tribal knowledge to enter that. And that happens through a combination of a good product, as well as embedding in their workflows.

So, when a table is being created, you embed in that workflow and intercept documentation when it's being created, instead of making it an afterthought be like, “Hey, engineer, you just created a segment event, like two days later.” You notify them and be like, “Hey, you never put any documentation”, except you should be forcing when they're creating the segment event to put the documentation because that event is going to land in the warehouse as a table with no documentation.

[00:18:37] JM: Gotcha. So, you've talked a little bit about antiquated data team formats. And I think one of the main reasons that data teams have been updated is the invention of these higher leverage tools that just save everybody a lot of time and give people more of a meaningful structure for how to position their teams. So, in this world of post Fivetran or reverse ETL tools and better analytics and dbt, I think I can understand where the data lake fits in. But I'd love to understand the usage patterns and what you're seeing in terms of how data teams are structuring and how the division of labor looks, and particularly, what they're doing with the data lake.

[00:19:30] MG: Yeah, so let's zoom out here a little bit. We got people in an organization who need to make decisions using data. These are what are often called business users, though I'm not a big fan of that term, but that's the most commonly used terms. It includes execs, it includes like operations people who are just looking at a dashboard and operating the company.

Now, the things that they're looking at, are being produced by someone savvy, someone thoughtful about data, about understanding the business and connecting the dots between those two. That person is usually a data analyst. So, let's go down this analytics chain. And if we have time, we'll go down the ML chain as well. So, someone is looking at the inside doing operations of the company, then there's an analyst producing these insights, they understand the business really well, they understand the data and the gotchas really well.

Now, these analysts are operating off of certain base data models that are being produced data models, data tables, same thing, that are being produced by a person who is more technical and is responsible for a pipeline that's generating them. That's a data engineer. And this is the kind of most standard sort of organizational structure. So, there's a team of data engineers usually centralized, called the data platform team or data engineering team, followed by a team of analysts. They often have a head, that's one person, head of analytics, but they're embedded in various different business groups. So, your analysts would be working in embedding and marketing or finance and things of that sort. And then in each of these domains within the company, let's say finance, they're working very closely with the business users and finance and that could be your CFO, for example.

[00:21:24] JM: Do you have any particular patterns that you've seen for, as I mentioned, these more modern tools? Any particular patterns for integrations you're seeing between Amundsen and some of these more modern tools?

[00:21:42] MG: Yeah, and example of these modern tools are reverse ETL tools like new ETL tools centralizer.

[00:21:49] JM: Yeah, yeah.

[00:21:51] MG: Yeah, definitely, Stemma and Amundsen have to integrate with these tools. So, it's pretty important for a data catalog to be able to ingest information from the source of truth where it is, right? So, take an example of a newer modern data stack tool, dbt. In dbt, the data engineer or analytics engineer is a writing job in order to derive one raw piece of data to another piece of data. And as a part of that, there are defining certain tags. It's often in the dbt file. They're also defining the descriptions for the columns that are being put in the dbt file, which is version controlled. There's absolutely no reason why this data engineer needs to put all that documentation in the GitHub file for dbt, as well as again in the catalog, right? So, the way that a catalog that's modern would integrate is it would slurp up all the descriptions from the dbt Docs into it, and make sure that they're available in the catalog and don't need to be rewritten.

That's just one example. And there's like countless examples, similar with Airflow. When a DAG fails, that a user, like an analyst should be able to see that the pipeline that produced this data every day at 9 AM, has failed last night, and therefore this data is no longer available in its most up to date form and it's delayed by a day, right? All of that information doesn't need to be like manually added by the data engineer and sent a blanket Slack message, but should be automatically important. These are the kinds of things that we integrate with.

[00:23:31] JM: We talked a little bit about how data gets into the data lake initially, how does the data lake get managed over time? How do database updates and new datasets get propagated to the data lake?

[00:23:49] MG: Yeah, in the data lake, I often see there's three kinds of data. The most common kind is replication from third-party systems. So, if you're a modern company, you've probably got a bunch of SaaS apps that you use to power your company, and that could be laddus or your HR stuff. It could be Salesforce for CRM, the list goes on, right? The very first need for analytics or decision making in the company is to take data from these apps, and bring it into a centralized place so you can connect. You can find the lifetime value of a customer or the amount of money you're spending on ads and how it connects to the lifetime value of the customer and things of that sort. So, that's one kind of data that gets brought in, replication from third-party systems.

The second kind of data that gets brought in is replication from existing internal databases. So, you may have your own website or your own phone app that's backed by Mongo or Dynamo or a sharded MySQL, if you're like Facebook. And that data needs to be brought into to do product analytics. And lastly, you may be using product analytics tools that are sitting on your website and providing data about how your users are interacting on the website or the app. And these tools are things like Segment, Heap, Amplitude, things of that sort. And so, these often also end up logging data to warehouse.

These are the three kinds of data systems that I've seen most commonly being integrated into the warehouse, or the lake. And the most common tools I see for this are Fivetran as a proprietary tool for replicating that information, and then Airbyte as an open source equivalent of that for bringing all that information into the lake/warehouse.

[00:25:36] JM: So, I want to talk about a use case for a data lake, and just as an example of something that is made easier with a data lake, and you've written some about data migrations and steps to making a data migration more successful. Can you give an example of a complex data migration that you've taken part in? Maybe something that you did at Lyft that can illustrate what exactly a data migration is?

[00:26:09] MG: Yeah, totally. I'll share an example from Lyft. Lyft, as you know, I worked at Lyft from 2017 until 2020. And during that time, Lyft was doubling every year in its employee base, and growing exponentially in terms of its data footprint. Much of this data comes from two different sources. One is obviously the app, and so, when the app was open, did they go through the funnel for requesting a ride, and where did it drop off, things of that sort. And the second thing is coming from internal services, the service that shows you the ET or the price. What price was shown, sometimes you have an M in model that's running in parallel, that's logging, what would be the new models price, and then you can compare and contrast the performance of the new model, versus the existing model that was being shown.

Like many fast-growing organizations, when Lyft started first thinking about having a central place to store this data to power analytics, it was a pretty small team. Both on the data front. as well as in the analytics front. And they got what was the state of the art, I believe this was maybe 2015, 2016. But this happened before my time. So, I don't quite fully know. My belief is 2015, when they got Redshift as their data warehouse of choice. And in addition to Redshift being their warehouse choice, they also most commonly had this paradigm of rebuilt tables. Now, there are two kinds of tables in the warehouse, usually, one is rebuilt, which means every night you are rebuilding the whole table. And then the other ones are incremental, in which every night are simply adding on new rows to the table, maybe updating some old rows.

And rebuild tables are like more consistent, so it's easier to build them. But as you can realize, if you're dropping the same amount of data, let's imagine that rewrites table at Lyft, right? If you're rebuilding that table, every night, you're taking all the rights that are ever happened, then sort of reading them and rewriting them every night, even though we may – and actually, in some cases, you can go like for example, you can take a Lyft ride and tip a driver, three days after your ride. Then you have to update the ride entity with the tip amount. So, there are cases in which you can go a certain amount of time back in order to rewrite, and that makes sense for rebuild tables. But in general, it doesn't work. And it breaks.

So, in Lyft, it was a combination of these rebuild tables and Redshift performance that was like, with the growth it was just like going downhill, right? And so, the question is, okay, we need to change the paradigm. And potentially, we need to change the warehouse, and those two things were coupled together. I'll focus more on the warehouse, because there were a bunch of issues on Redshift. So, what Lyft embarked upon was an effort to go out of Redshift to a new data warehouse. In Lyft's case, that was a self-hosted hive cluster, and a self-hosted presto cluster. But that was a migration I'm talking about. So, taking all the data that exists in Redshift, and moving it all to this new data lake. The problem, however, is not only you have to move all the base data, so all the replication that's happening that needs to move, you have to move all the derived data that's being built. So, all the ETL jobs need to be moved. And not just that, you need to move the dashboards that are being built on that.

And so, it's a long, laborious process and I think migrations are one thing that are constantly underestimated in their complexity, and I have learned the hard way, a few key things that you can do to reduce the complexity and to manage migrations really well. And I published a blog post on this topic too, that we can share a link at the bottom of the post, but that's one of the things that was one of the key things I learned from Lyft.

[00:29:57] JM: So, the different components of making a successful data migration, you have three steps, determine the complexity of the order of migration, migrate and test level one tables, and provide tools for downstream owners, and kick off the ripple migration. So, I think of this as a process that is respectful of both the complexity of where the data assets themselves are sitting, and the importance of those data assets, and then also is respectful of all the downstream consumers, and takes a pretty delicate approach to making that migration successful. Because you have so many ramifications of – second order and third order effects of a migration. What's the process for actually being able to assess all of the second and third order effects that a migration is going to cause?

[00:31:03] MG: Yeah, the process usually includes understanding what is being built on the data that needs to be migrated. And then, being able to see if that artifact, those set of artifacts are going to have to either change so they can absorb the change, or they are not being used at all, and can be deprecated. And so, one of the key things that I think we don't do enough in migration, is understanding – let's take a given table like, the base Salesforce replica table, right? Understanding what are the leaves off this particular table. It may be 10 levels before we get to the leaves off this graph. But that leaf may be used by an exec dashboard, or an ML model that's powering some recommendation engine on your website. And most companies want to – so, that is a proxy for risk.

Another one is that a leaf could be a table that's 10 levels deep, but it has a pipeline, that's the most expensive pipeline that runs to generate that table. So, that is a proxy for cost, right. And so, I think one strategic decision that you can make when you are doing the migration is realize it's not an overnight thing. And then find the axes at which you want to prioritize your migration. And that's either a cost, like I want to move the most costly things first, because I'm burning cash, or its usage impact that I don't want my analyst to suffer as I'm doing migration, that's the thing I'm optimizing for. Or the third one is risk, like I want to migrate the most risky assets last, because I will get some practice migrating the earlier ones first.

My recommendation, unless you're burning cash is to always do risk-based migration. So, you start with all the exact dashboards get migrated last, right? And you start with the ones that don't have the leaves that are the most important. And often, when you start, you can actually ask your users to deprecate stuff. Like you would find in a migration, about half, I don't know, depends, obviously, in the organization. But in Lyft's case, anywhere from 25 to 50 version things were not used, and migration was a good way to kind of clear out that clutter, too.

[00:33:33] JM: And how does a data lake make this overall data migration easier?

[00:33:40] MG: Yeah, so what a data catalog does is it's tracking lineage across various different data sets, dashboards, and it is able to give a tool in the hands of these data engineers who are doing migration to see the impact of the proposed migration, right? That's one thing. So, it's tools in the hands of data engineers to see the impact. The second thing is, your data analysts or data scientists have to do some work in order to migrate their models, their insights, their analytics, from the old stuff to the new stuff. And what you need to do is to also give them tools to understand what are the things that you own? What are the things that you need to migrate? Hand them some example clauses in order to see how others have done this. So, it also serves on the other side of this marketplace where the consumers of this data are sitting, and have to migrate. So, a data catalog gives them information about, okay, I'm migrating this particular part of the datasets data first, and therefore, these dashboards that I own are impacted, but these other dashboards that I own are not impacted right now. So, it helps you sequence a migration and everybody will be in the same page as to what subset of datasets are available in the new system, and what subsets of data are still to be used in the old system.

[00:35:08] JM: Yeah, apologies. I refer to data catalog as data lake a couple of times, I realize now. When you're making a data migration, is it useful to have like – when you're issuing those test tables, do you have the downstream consumers actually read from all of the tests tables that you make? Or what does the testing process look like in practice? And to what extent do you propagate the tests to downstream consumers?

[00:35:44] MG: So, the testing process is relevant here, because say you're migrating from Redshift to Snowflake, and you've created it a – you're going to do it in three phases based on risk. So, you've taken phase zero, which is the least risky, and you've created replicas of these tables in Snowflake. But before you ask your analysts to start pouring over their analytics on it, you want to do some validation on it. So, that's what I'm referring to in the blog post as testing. And this is, in my opinion, the responsibility of the data engineering team before they actually farm this off to the analytics. There are various tools that exist that can help you here. At the core, it involves profiling your data in both places, and being able to diff these profiles on an ongoing basis, right? And then you can set thresholds, like, I'm okay with 0.2% of records being off, so on and so forth.

So, that's one very common way, but obviously, that's just the most simplest thing is to do record count. You can take various columns. I would suggest that you start with columns that are most important to you, and the right example, I'll keep going here. The type of ride, like whether it's a line ride, or an Excel ride, or regular ride, or a scheduled ride is a pretty important column. So, making sure that the distribution of that column remains the same, is also pretty important. You can also do distribution checks in certain columns.

[00:37:14] JM: What are the tools that you use to validate the tests during a data migration?

[00:37:22] MG: Yeah, at Lyft, there was a homegrown tool, which is also pretty commonly done in many organizations. Great expectations as a profiler that you can use great expectation being the open source project, leading us to compare profile. And then there is a proprietary tool called Data Diff, which is often used in this case as well.

[00:37:42] JM: Do you have any other examples of how data migrations have – what you've seen amongst like customers, in terms of data migrations, since you have a pretty wide variety of customers, and I think a lot of them probably have some esoteric tools that maybe make a migration more complicated.

[00:38:04] MG: Yeah, I'll share an example of a customer that we have that's a public financial company, and they're undergoing a migration from one of their existing. They actually have a collection of old school data warehouses, that they're migrating to one new school data warehouse. I think, Snowflake/BigQuery, right? And for them, the hardest thing has been twofold. One is there's all these analytics that exists on top off your existing data warehouses. But it's not clear who's using that analytics, who's the owner, and is it still being actively used? So, there's actually a lot of organizational energy spent on understanding all the analysis that's been done on top of various different data warehouses, and then finding owners for it. That's one place where I think they they're spending a lot of their energy in.

The second place is that you still need to connect this analysis that's been done to the data sets that exist in the warehouse, because the people migrating the warehouse thinks in terms of data sets or tables, those two terms can be used interchangeably. And so, if you are deprecating a particular data set in the old warehouse, and you want people to start using that the new warehouse, you need to also notify all the various dashboard owners that exist on top of the data set that exists in the old warehouse. And that's the place where Stemma has been very helpful for them because it shows them the connections between various different tables and the dashboards.

So, once they have identified that these are the key dashboards that need to be deprecated, they can connect them to see if there are tables, what tables they come from, and see if there are any other dashboards that are actively being used off of those tables that need to be moved over to the new warehouse before they can be deprecated. So, that part has been very helpful with Stemma, but the first part around ownership, like that's a very organizational thing. I think that's pretty common for any company that's been around a while for them to invest some energy into figuring out who the owners are and be able to document it.

[00:40:14] JM: I want to shift the conversation to something that you wrote a post about fairly recently, which is data citizenship. And since we've been talking about the internals of how to manage data at a company, I think we can just close off by talking more philosophically about about data management. Can you define what it means to be a data citizen?

[00:40:40] MG: Yeah, it's not a whole lot different than how we think about ourselves as a citizen of a country, right? So, the concept we've been taught all along in schools, is that we should be good citizens, we vote, we participate in local politics, and we elect leaders. And then we have started to bring some of the same culture and responsibilities around leadership to the data domains. And so, in the data team today, we say it is your responsibility to document the data that you are creating. We lean a lot on this good citizenship behavior of an individual to contribute to the data health of the company. And we use citizenship as a primary mechanism for documenting what data is a source of truth and what's not.

This actually has not been working out for us as a community. The problem is that 10 years ago, data teams were very centralized, right? We were a full-service data team, which produced all the data that was needed for the organization. But now we are more self-service data teams. We are allowing people to create their own analytics. And in many cases now, we're allowing people and enabling our users to create their own tables, their own data models, right? So, they're getting closer and closer to doing the work that was previously done by a centralized team and being democratized across the organization. That's led to another problem where we have just too much data in the companies and ownership is scattered around. With ownership being scattered around, and this notion of citizenship, which doesn't come with clear responsibilities, essentially lead to a failure of that work to document the data and to understand what's the source of truth. That's really the key problem that we've been trying to solve to citizenship and it's not working out.

[00:42:50] JM: So, is the article that you wrote or co-wrote, is it just a stance that this simply isn't working? Or is it a call to action to change something? Is this solvable problem?

[00:43:02] MG: I hope that it's a solvable problem. I don't think we've cracked the nut on it. And the article actually is proposing that we learn from existing alternative domains that have had a similar set of problems. So, our problem here is you got too much, and it's hard for you to find the needle in the haystack. If you look at what are some of the adjacent domains that have this problem? So, you think about Wikipedia. Wikipedia is a place where you can successfully document an extraordinary amount of information. But what we can learn from it as it actually only works on an enormous scale, about 0.5% of Wikipedia's readers edit an article a given month. We don't have that kind of proportion within an organization.

Another example here is Yelp. And one thing we can see from Yelp is rather than sourcing reviews from professional critics, it relies on restaurant goers like you and me to write them. I think what we've also done as a part of the citizenship is, we focus too much on these owners. We rely on owners too much to put in this information. But a savvy consumer who knows the domain well can also help a lot in upleveling the amount of information we know about the data. And their activity, my peer's activity around what data they use, what dashboard they use, is very relevant for me, and showcasing that to me is a very useful thing, right? Instead of just putting responsibility in the owner of the document.

The post talks a lot about similar experience with Cora, Google, new sites like Vox, and what we can learn from them. So, I won't necessarily go into all of them right now, but those are some of the key things that I think we can learn from existing domains that are not in data.

[00:45:00] JM: Now, is the thrust of data citizenship or how you're outlining it here, essentially, that there's this gradient between private and public data? And that gradient is not really being managed properly. Is privacy the issue?

[00:45:21] MG: No, privacy is not the issue in my opinion. I'm more talking about ownership and responsibility.

[00:45:28] JM: So, meaning, like the origins of the data, like where data begins and how it evolves?

[00:45:35] MG: Yeah, that and documenting what you should use under what circumstances. What are the gotchas around the data? Where should you be careful? And whose responsibility is to document all that?

[00:45:47] JM: Okay, I see. So, do you have any words of wisdom for companies to improve their data citizenship?

[00:45:58] MG: Yeah. I would say there are four things that I think that help a lot with this. So, the problem is that there's just very little trust in data. And the first thing that I think actually helps is, if there is more review of metadata or descriptions, then documenting them. And we, as humans, we love correcting a missing piece of information or correcting something that's wrong, instead of giving us a blank slate and writing from scratch, right? So, one thing I have, as review more, document less.

The second thing is that let there be mess. We find that because the sheer amount of volume of data in organizations, you can't solve for trusting or documenting all the data that's present in the organization. You need to know what are the hotspots. What are the biggest bang for the buck based on usage, based on what dashboards are being viewed by whom, so on and so forth, and focus on documenting those, right? I don't think 100% documentation for all your data is a goal that's possible for any company.

The third thing I'll say, and this one comes from our learnings from Yelp is not just the owner. There's a lot of talk with this concept of data mesh that's been going around their community around decentralized ownership. And many of us are talking about it in a way that if we were able to delegate ownership to the sister team, or the team where the data is coming from, all of these problems will be solved. And the key point I'm making here is that it's not just the owner that a savvy consumer, therefore a savvy analyst, and their usage patterns can tell you a lot about what data is trustworthy and what's not, just by looking at their patterns.

And lastly, you have to get it while it's hot. And what I'm referring to this concept of RAM, right? When an engineer is creating a new dataset, they have a bunch of contexts in their head. And if I don't get that context, right when the dataset is being created, and I show up even 30 minutes or an hour later, they have offloaded that context on from the RAM, and there is a barrier to entry to getting that context again. So, instead of me coming to the engineer a day later and saying like, “Hey, you created this dataset, it’s not documented or it doesn't have a particular classification field that is expected from you to put, we have to tap in their workflow.” And when they are submitting that PR to add a new dataset, we need to show a warning, break the build if it doesn't contain certain level of attributes, documentation that it needs to have. So, those are my four recommendations going forward.

[00:48:43] JM: Just to wrap up, given that you've been so involved with data teams at both Lyft and just in your work with Stemma, do you have any high-level words of wisdom for managing data organizations, just maybe things that you've been thinking about organizational principles or management principles?

[00:49:08] MG: Yeah, three things come to my mind. One is around scale. And what I'm trying to say is that data teams rarely grow at the same scale that the company is growing or the number of users who are going to use the data that are growing. So, be mindful that you necessarily can't grow the team linearly to your user growth. And that means, that brings me to the second point, enable. So, your primary goal in a data team, is to enable others to do data driven decision making or data driven modeling. And that means that you either build a core dataset in the company so others can use it or you provide tools so others can build their own dataset, their own dashboards.

The third thing as you are doing all this and enabling, you have to keep guard on the best practices. You have to educate and govern. So, that would be the third principle, I would say, is that when there is chaos, you need to provide some guidelines around chaos. What does trustworthy mean? And enable your users to be able to share what's trustworthy is in their domain with their other peers and other users. So, those are the three sort of areas I would focus on. Think about the scale, enable your data users, and govern to prevent chaos.

[00:50:28] JM: Cool. Well, I think that's a good place to wrap up. Mark, thanks for coming back on the show. It's been a real pleasure.

[00:50:33] MG: Thank you for having me. Always a pleasure.

[END]