EPISODE 749

[INTRODUCTION]

[0:00:00.3] JM: If a business has been operating successfully for a few years, that business has accumulated a high volume of data. That data exists in spreadsheets, CSV files, log files and balance sheets. Data might be spread across local files on a user's laptop, databases in the cloud, or storage systems in an on-premise data center. Older businesses have more data, in more places, in more formats.

Legacy systems and old batch processing jobs that have been running for years are taking data from one place and porting it to another. Every mature company needs to access and analyze the data in all of these different places, whether they're a publication with millions of readers like The Economist, or a fast-growing infrastructure provider like Twilio.

Business intelligence is a term that is often used to describe tools for analyzing data in the form of charts and graphs and reports. Business intelligence applications are crucial to the success of a business, because they are used by everyone in an organization. Whether you are a business analyst forecasting sales for the next quarter, an engineer who is determining how many servers to provision, or a CEO trying to decide what the best area of your business to focus on is.

There have been several generations of business intelligence tools. Each generation of business intelligence is built for the trends and the infrastructure of that generation. Looker is a more recent business intelligence tool that was built in light of several trends in software, the growth in volume of data, the growth in the number of systems that users need, the changing types of users that need to access that data. There's a lot more different types of users in an organization that need to access this data at scale, and also the need to share business intelligence across social workplace tools, like Slack and Asana and e-mail.

Daniel Mintz joins the show to describe his experience using business intelligence tooling and his work at Looker, as well as the landscape of business intelligence, ETL and data engineering.

We're gathering feedback for the survey process, the listener survey process in our 2019 listener survey. You can go to softwareengineeringdaily.com/survey and tell us how we can improve. We read every piece of feedback there and we'd like to know how to adjust the direction of the show to better serve you as a listener. Also, if you have ideas for the show for 2019, please send me an e-mail, jeff@softwareengineeringdaily.com. I'm scoping out the landscape of ideas that we should be exploring in 2019 and crafting a show calendar. I'd really like there to be, perhaps a more of a consistent narrative across the different shows.

I'm trying to architect this from the beginning, and a little bit more of a careful fashion, rather than the slipshod variety of topics that I know people do like, but perhaps it could have more of a consistent narrative from day-to-day.

Also, we're looking for sponsors for 2019. If you're interested in reaching the 50,000 developers that listen to Software Engineering Daily, you can check out softwareengineeringdaily.com/sponsor to learn more. You can also help us out by sending an e-mail to your marketing director, or your CMO, or your CEO. Many people are surprised by how effective podcast advertising can be to get their message out to technologists. Believe it or not, it's often difficult to convince advertisers to buy podcast ads, despite the fact that they are very successful for hiring people, or getting your product out there into the hands of new developers who have not heard about it.

As a developer working in an organization, or a senior executive, or product manager, whoever you are, your recommendation can go a long way to actually having us get some support from sponsors. We do heavily invest back into Software Engineering Daily. We've got some very exciting projects and ideas that we're working on. You can rest assured that the sponsorship dollars that go towards Software Engineering Daily go into our product offerings and delivering you improved content.

As always, you can send me an e-mail with any feedback, criticism or suggestions, jeff@softwareengineeringdaily.com. Let's get on with today's show.

[SPONSOR MESSAGE]

[0:04:43.0] JM: For years when I started building a new app, I would use MongoDB. Now I use MongoDB Atlas. MongoDb Atlas is the easiest way to use MongoDB in the cloud. It's never been easier to hit the ground running.
MongoDB Atlas is the only database as a service, from the engineers who built MongoDB. The dashboard is simple and intuitive, but it provides all the functionality that you need. The customer service is staffed by people who can respond to your technical questions about Mongo.

With continuous backup, VPC peering, monitoring and security features, MongoDB Atlas gives you everything you need from MongoDB in an easy to use service. You could forget about needing to patch your Mongo instances and keep it up to date, because Atlas automatically updates its version. Check out mongodb.com/sedaily to get started with MongoDB Atlas and get $10 credit for free.

Even if you're already running MongoDB in the cloud, Atlas makes migrating your deployment from another cloud service provider trivial with its live import feature.

Get started with a free three-node replica set. No credit card is required. As an exclusive offer for Software Engineering Daily listeners, use code SEDAILY for $10 credit when you're ready to scale up. Go to mongodb.com/sedaily to check it out.

Thanks to MongoDB for being a repeat sponsor of Software Engineering Daily. It means a whole lot to us.

[INTERVIEW]

[0:06:40.9] JM: Daniel Mintz, you are the Chief Data evangelist at Looker. Welcome to Software Engineering Daily.

[0:06:45.6] DM: Thanks for having me, Jeff.

[0:06:47.2] DM: Looker got started in 2011. Companies had data problems back then. Companies still have data problems today. Describe the data problems that exist within a typical company.

[0:07:00.0] DM: I thought you were going to ask me why haven't we solved all the problems yet. Yeah, sure. The data problems are that more and more applications are proliferating inside every company. I forgot the number, but the number of average SaaS applications that Salesforce found inside your average organization was just staggering. Eery one of those applications puts off data exhaust, which means that companies are dealing with more data than they ever have before. People are spending time in applications, basically adding data and that data is going into databases and data warehouses and data lakes. They're not doing a great job of getting meaning back out, right? Of getting useful information and knowledge back out of that data. That's fundamentally what the data problem is.

[0:07:49.1] JM: There are also problems around different teams having to coordinate. Actually though, I guess the problem that you just described could be arguably something around teams not coordinating well. If you've got this huge data ingest pipeline, and then maybe the ingest pipeline isn't putting enough structure around the data, or it's not going to the right place, and so that people can't access the data and extract the right analysis from it, or maybe they don't know how to query that data source that it's being written to.

In any case, you have a difference between perhaps the data engineer that is writing the data, or that is coordinating the data lake formation. Then you have the business analyst that needs to study that data. They may not know where the data is, or how to get the data. Describe the different teams that are involved in the data analytics pipeline and the frictions that develop between them.

[0:08:44.4] DM: Sure. I mean, I think what you described is exactly right, that there are lots of people putting data in. I tend to think of data lakes in most cases as the place that data engineers, or engineers in general ask people to put data when they don't want to deal with it, just to like, “Oh, I'll throw it in the data lake. That way if we need it later, it'll be there,” but no one actually does anything with it, because they can't find it, or they can't make sense of it.

I think the issue is that nobody actually wants data. They want knowledge. They want that knowledge to be consistent across the org and they want it to be accessible across the org. You have product owners who own all these SaaS applications, which could be really in any department at this point. You have data engineers who are responsible for getting the data out of its source system, cleaning it up a little bit, moving it into whatever the long-term storage medium is.

You have core BI analysts, business intelligence analysts who are trying to give that data some structure that used to be a really huge job to structure it really nicely, so that it could go into the data warehouse, because those data warehouses couldn't deal with anything that wasn't a perfect star schema. It's a lot easier now, because the data warehouses are much more capable of dealing with messy data, real-world data.

You've got those BI analysts, then you've got business analysts who live across the organization in every department. Then I think, you can't forget about all the people in business roles who want information from that data, right? They're fundamentally the consumers. They're the ones who are making the demands, in terms of trying to get value from that data. They even more than everybody else, don't want to have to worry about where that data lives, whether that data was collected correctly, if it's clean, if it's structured properly, if it's put into terms that the business understands, that makes sense to the business, right? They don't want that data raw. They want it as business metrics.

That data pipeline involves a lot of people. Then it can go even beyond that, because maybe you're turning that data around and giving it to your customers, and so you have developers who are then building out portals and embedded analytics for your customers. There's just a lot of touch points and keeping everybody in sync is a big challenge.

[0:10:56.8] JM: Before we get into Looker, which is where you currently work, I'd to give people who are listening some context on data analytics as it's practiced today through a case study. Before you joined Looker, you worked at Upworthy, which is a media site and you've written about your time there in terms of the data pipeline and the data challenges that you had there. I think this is a good case study, because media sites they have to make data-driven decisions around the advertising and the content.

It can be a very data-intensive job, because you have all these users that go on the site and you have all these events that happen and you have to correlate events and content and what advertising is making you money. Take me inside the life of a modern data analyst as you experienced it at Upworthy.

[0:11:53.6] DM: Yeah. I do think that publishers, media is an interesting case study, because they have an enormous amount of data, much more than an e-commerce site, for example, would have. A lot of that data is not very well correlated. You don't really necessarily know who the user that read the article was, and so trying to figure out those patterns at a very high level is tricky.

My life as the data analyst at Upworthy a lot of the time was spent architecting the data system to be able to do data capture at the scale that we needed it to and to make that available, that turn that data around to make it available to all of the folks as you said in the customer-facing positions, the advertiser-facing positions and also the editorial staff.

I think data analysts and data engineers, it's easy to focus on the technical requirements, because those are intellectually demanding and challenging and fun in a lot of ways, and the technology that's available these days is pretty amazing, so getting to play with that stuff is great. I often found that some of the biggest challenges were less on the technical side, less how do we structure this table so that it joins efficiently? More, how do we present this data to a writer, or an editor, or a revenue operations manager in a way that will make sense to them that will actually be useful in their job? That can be a much bigger challenge trying to put yourself in their shoes and think about what they need to do their jobs.

Often as an analyst you'll say, “Well, I'm sure this dashboard will work,” and you put it in front of them and they go, “Yeah, that's nice.” They're never going to look at it again. I'm a big advocate of data product managers, because I think in the engineering world, engineers have figured out that there really is an art to understanding users and user-centric design and thinking about it as a process where you're constantly refining your understanding of how users are interacting with your product. I think data folks need to very much think in the same way.

[0:13:54.7] JM: Like many startups, Upworthy was built on MongoDB as the core business database. Mongo is a non-relational database, is a document store. Why is it non-relational database, like Mongo? Why does that make it challenging for data analysis?

[0:14:14.2] DM: Sure. I mean, I think the reason – so Mongo was chosen before I got to Upworthy by the engineering team and their reasons for choosing it were pretty reasonable, which were – they didn't know exactly what Upworthy was going to be and they didn't know what data structures it was going to have, they didn't know what scaling requirements it was going to have. Mongo seemed a reasonable choice.

From an analytics perspective, you actually don't want all the flexibility that unstructured or semi-structured data gives you. There are times when you do. If you're collecting API results or something and you don't quite know what the format is going to be; a semi structured data set can be useful. When you're actually doing analysis, you want the data to have a predictable structure, because you need that in order to do good analysis. You need to know what is expected and what deviates from those expectations, and structure and schemas give you those expectations, their way to concretize your expectations and then at scale, look at where the reality differs from the expectations.

[0:15:20.9] JM: Right. That presents challenges for data analysis, because it's a non-relational database. Why is that? If you have this unstructured data and then you have data analysis that you need to do in a relational database, I guess help explain people – help explain to people why, if you have a non-relational database where for example, you have these JSON objects that are in Mongo, some of them have a field like, let's say timestamp and then other ones don't have that field. You have this unstructured data. How does that make it challenging for doing large-scale data analysis?

[0:16:03.5] DM: Yeah. I mean, I think I can give you some more concrete examples there. I think, to use the data example, if what we're doing is a is a time-based analysis and I want to see page views per day, I need that timestamp. The data points that lack that timestamp are actually really problematic for me. To not be able to tell which ones lack that timestamp until I dive down into each and every document in the document store is pretty inefficient and just makes that analysis really hard.

I think, particularly when you're talking about big, analytic warehouses, which tend to be columnar stores, the whole insight that makes data warehouses work, that makes them so darn good at analysis and terrible at transactional data workflows is that they're columnar. All the timestamps are together, all of the first name fields are together, all of the zip codes are together, because they're all one column, right? There are many, many rows, but they're all one column and that column is stored in memory together.

What that means is if I want to see users by zip code, I only need to go get the zip code column. I don't care about the first name, the last name, the number of orders, the first order timestamp. I don't need any of that stuff. I can literally just go get the zip codes, because they're all stored together in one place in the database, and then on top of that, they can press really nicely, because they're all of the same type.

That is what makes analytic databases work is that all of the data that I'm likely to care about for a particular analysis lives together in one place in memory and it makes incredibly efficient for analysis. Both document stores and transactional databases take the opposite approach, which is to store the row, or the document altogether in one place. If I have to load all of that extraneous data every time that I want to do an analysis on zip codes, that's really inefficient and really slow.

[0:17:59.5] JM: That motivation of the columnar store, at Upworthy you had this MongoDB database that was the business database. This had originally had all the data in it and then you copied it over to PostgresSQL. You were copying it over constantly to PostgresSQL, which is a relational database. Then PostgresSQL couldn't handle the volume of ad hoc queries from data scientists. I think also, PostgresSQL is not columnar. For all I know, there could be some plugins that allow you to treat it as columnar, but I don't think it's columnar by nature.

[0:18:35.3] DM: That's right. Yeah.

[0:18:36.7] JM: You also stood up Amazon Redshift, which is a data warehouse. Explain what a data warehouse does for you in this situation. What were the pressures that you've had from a data analytics standpoint that made you need a data warehouse?

[0:18:50.9] DM: Yeah. You give a pretty good description of what our data architecture look like. We have that Mongo database, MongoDB, document store that was running our website. We would ship that data over to a PostgresSQL database, transactional database, in order to get it into a relational context where we could query it easily. That was actually a really small data. It was our document data. It was the data about what was on our website. It wasn't the data, the event log of who was visiting our website and how many hits each page was getting, because that data at its most granular level would have broken the PostgresSQL database, or the Mongo database for that matter, right?

I mean, the Mongo database I guess could have handled it, but it wouldn't have been very useful in an analytic context, because of the things I was talking about before. We had the PostgresSQL data, which was relatively small. It was a couple dozen tables. None of which had more than, I don't know, a few tens of thousands of rows. The data about how many people were visiting the website and what they were doing on the website was enormous. By contrast, it was many millions of rows per day.

We built an entirely separate pipeline to pull that data down, because it's much bigger to process it pretty lightly and then to get it into Redshift, because red shift really is built to handle tens, hundreds of millions, billions of rows pretty effortlessly. In order to make sense of that data, we still needed the data that had originated in the Mongo database, right? We wanted to not just know what was the hash, or the ID of the page that lots of people had visited, but what was the content of that page.

This is a pattern that you see a lot in data analysis, where the really large-scale event stream data comes from one place, and then the dimensional data about what the content was comes from a different place and you can join them using foreign keys. Redshift was a really great option for doing that. Redshift was the first player in the cloud data warehouse space, where in the past going back to the very beginnings of data warehouses, data warehouses were things that were incredibly expensive and relatively slow. You had to spend many hundreds of thousands or millions of dollars upfront to have one of them trucked into your data center.

The idea that AWS now made Redshift available and something that you could just pay for by the hour was really a game changer. There are a number of other big players now in that space, which is great. We love the competition and all of the cool stuff that they're all doing, but for us at that point, Redshift was the only real option.

[0:21:31.6] JM: How often did you need to copy the data from Mongo to PostgresSQL to Redshift; this ETL job, this copying of data?

[0:21:43.1] DM: Yeah. The Mongo to PostgresSQL copy was actually not copies, as much as this amazing little program that Stripe made, because they face the same problem, which was needing to move data from Mongo to PostgresSQL. They built this little tool called MoSQL and they open sourced it. I don't know that they're maintaining it anymore, but it basically would tail the oplog of MongoDB and turn those operations into inserts and updates and DML statements for PostgresSQL.

It did not seem like there was any way that it was going to work. It just seemed like this is a crazy thing. This should never work. Instead, it worked brilliantly and worked year-after-year. The latency between the Mongo database and the PostgresSQL database was sub-second. It was amazing. That was great.

Then in terms of ETL'ing the data from the PostgresSQL database to Redshift, that was a proper ETL, where we would extract the data from PostgresSQL, put it on S3 and then ingest it into Redshift. I think that was running probably hourly, though I don't specifically remember whereas the event stream by the end, we had flowing in. I think the latency was about 10 minutes. It took about 10 minutes for the data to get from a user's computer up into the cloud and then get pushed through the various parts of the ETL and get into Redshift.

[0:23:09.7] JM: The process of streaming the logs from MongoDB into PostgresSQL so that you have this PostgresSQL database that mirrors the Mongo database, why couldn't you just do the same thing for Redshift, so that Redshift would have none of this hour-long, or 10-minute long latency?

[0:23:32.4] DM: Yeah. It goes back to the core differences between analytic databases and transactional databases. PostgresSQL being transactional is really good at doing a ton of small jobs constantly, right? Do this one row insert, do this one row update, create this table; that's really, really fast on PostgresSQL. Redshift on the other hand pretty like much all databases, or data warehouses, love batching. It wants data to be big and to do bulk jobs, because it is MPP, because it's massively parallel. In order to take care, take advantage of that parallelization, it needs bulk, because it has lots of workers and one leader node.

If you tried to do those ongoing, small transactional jobs moving data right into Redshift, you'd very quickly run into problems, because you'd be using the analytic warehouse incorrectly, so to speak.

[SPONSOR MESSAGE]

[0:24:39.8] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU-optimized droplets perfect for highly active frontend servers, or CICD workloads.

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily. As a bonus to our listeners, you will get a $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great

new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily. Thanks to DigitalOcean for being a sponsor.

The co-founder of DigitalOcean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[0:26:47.3] JM: Now we've described the data engineering and the database stack at Upworthy. What about the data analysts? There are these data analysts that need to ask questions at Upworthy. I think you were one of them. What kinds of questions did the analysts need to ask and what tools were they using to ask those questions?

[0:27:09.9] DM: Yeah. I was one of them. I was the head of data and analytics, and so I was in charge of that team and which encompass both the data engineering, and we had some great data engineers who I got to work with and then the analysts. The analysts, I would say we were answering two kinds of questions; one were more open-ended. They were like, “Here's a pile of data. What can you make of this? Are there any really important interesting analytic findings in it?”

I think, those kinds of questions it can – analysts I would say, have a tendency to get stuck on interesting, but not useful to the business. Part of my job as somebody who managed analysts was trying to warn them off of those things, or get them out of the rabbit holes before they got too deep. Sometimes that leads to really valuable insight about overall patterns, about, “Oh, it seems these kinds of photos perform better on Facebook than on Twitter, or it seems this headline is really valuable.” Those tend to come from more open-ended analysis.

Whereas, the other question that you're always dealing with are more operational, right? They’re, “Oh, I need a dashboard that shows how close we are to fulfilling this advertiser’s order. Or I need a dashboard that shows the results of the tests that I'm running right now, comparing different write-ups of this event.” Those are very directed questions. In both cases,

an analyst toolkit is mostly when you're dealing with large-scale data, some way to manipulate that large-scale data, which I would say is almost always some version of SQL.

Then if you're doing more data sciency things, you're probably using something like R, or Python, or some other data science environment. That's been the standard toolkit for data analysts for a long time and I'd say Excel probably is also in that bucket. I think, particularly for the more operational needs, the things where you've got a directed question and it's not something that's a one-off, but something that you're going to want to put in front of business users on an ongoing basis, that's where would have historically been called business intelligence tools become very valuable, because they're not just looking at rows and columns, they give you the ability to visualize, they give you the ability to refresh that data in different ways, they give you the ability to present something to a business user that's going to make sense to them, which fundamentally is what the job is about.

[0:29:42.9] JM: When you were at Upworthy, it was a reasonably young company. This is a 2 – I think you were there in the first four or five years of the company. Actually maybe – No. Well, something like that.

[0:29:55.9] DM: Yeah. Yeah, I was there for about three and a half years. I joined about a year after they had started. They didn't have a data stack, or a data team to speak of and I stayed for about three and half years. Yeah.

[0:30:04.5] JM: Right. Then after that, you joined Looker. I think in your time at Looker, you've probably seen a great number of older companies that have many, many more databases and more departments and they really start to suffer from this interdepartmental data problem and a whole host of other problems, like big database, things that have been there for a really long time because the company's 40 or 50-years-old. What are the problems that you see at much later stage companies?

[0:30:38.9] DM: Yeah. I mean, I think you're describing it well. I would say, so the way that I generally tell the story is chronologically, because I think it is actually a chronological story. If you go back to the beginnings of business intelligence in the 80s, which is really the first time that businesses are doing what we now know as BI, or data analytics at a serious level, the core

constraint that drove most of their decision-making was that databases were extremely slow and extremely expensive.

When you bought one of the business intelligence tools, you bought it from a giant company, like Oracle, or Business Objects, or MicroStrategy and you probably were buying what was then called the appliance, which was the physical hardware and the software from the same vendor, because everything had to be monolithic for it to work at all.

If you got it up and running and got your data really nicely manicured and loaded into that data appliance, you could present a handful of dashboards to your users, which was great, because that was a lot more than they had to date. If somebody wanted to change to that dashboard, or a new dashboard, that was very expensive and probably meant buying more data appliances and took six months. It was slow, and so you had this bottleneck where requesting any new data was a really slow process.

The idea of doing ad hoc analysis was simply impossible, because letting people do that would have just crushed the data appliance. There's just no way that they were powerful enough to keep up with that. That was the state of the world in that first period where the bottleneck was the core problem.

Then the world starts to change and people start to get computers at least in their departments and maybe even on their desktops that are fast by historical standards. They are starting to be able to hold on to and slice and dice reasonable amounts of data. Certainly not as much data as could fit in the data warehouse, but enough to be meaningful to a business user. You start to see these more self-service tools come about where again, you can't hold all of the data from the data warehouse, but let's say I'm on the finance team, I can ask for an extract of just the finance data from a particular period from the data warehousing team and they give me that extract as a CSV or something and I load it my departmental server.

I can slice and dice it. I can ask new questions of that data in basically real-time. That's a real move forward. That's a lot better, because now you can answer novel questions. People are really happy, because the bottleneck is to some extent, at least broken. Now the old systems continue to exist, because when you're doing your taxes, or your SCC reports, you want the

absolutely correct, not somebody poking around on part of the data version of the truth. You want the absolutely correct version of the truth that was blessed and built by the data analysis team.

You still have those old systems, but people have these newer systems and everything seems great, until you from finance get in a room with somebody from sales and somebody from marketing and they also have gotten data extracts and sliced and diced them themselves. You go, “Guys, what's going on? Sales are really down this quarter.” Sales goes, “No, they're not. They're up.” Marketing goes, “No, they're not. They're flat.” Now you have data chaos, right? You have different extracts that were pulled from the data warehouse at different times. Maybe those came from different sources. Then people analyzed them differently.

That conversion from raw data into business metrics could go really sideways, because people had completely different ways of interpreting that data. I think the easiest way for most people to think about this is Excel, where in your workbook, in your spreadsheet is both the data itself and then all of the logical transformations that you've made of that data. If I am working on an Excel workbook and have done a bunch of stuff to it and then send it over to you Jeff and you start working on and then I go, “Oh, no. I sent you the wrong version. Hold on, let me send you the new one,” and then trying to walk back those change. You can see and probably have experienced how quickly that can get messy.

That's really the world that people are living in is where they have to either choose to lock everything down, or they have to choose to let people have a free-for-all, where they can interpret the data themselves. Neither of those is great. In both cases, the core reason, the core constraint that drove those decisions was the same, which is that databases and data warehouses are really expensive. They're really hard to maintain and they're really slow.

All of your decisions are about how do I protect that data warehouse from my users? While I would say that the big data revolution has not necessarily fully delivered on its promise, I think one place where it absolutely has delivered is on the speed and power of the underlying technology. I would say starting with the Hadoop ecosystem in the 2000s and then certainly continuing with these cloud data warehouses, like Redshift from Amazon and Google BigQuery and Snowflake and a bunch of others, they are incredibly powerful, they're incredibly fast, they

costs pennies compared to what the old data warehouses cost. You can spin them up in seconds. That opens up all new opportunities, because all of a sudden, that core constraint that you were bound by has gone away.

[0:36:11.9] JM: This changing nature of data engineering and the cost changes and also the problem statement that you gave of ending up with different departments, having different perspectives on what the quarterly revenue is for example, that has impacted the evolution of the BI layer. The set of problems, the set of constraints is in part the inspiration for Looker.

You've discussed in large part the changing nature of data engineering. Tell me a bit about the evolution of the BI layer, the visualization, the business intelligence layer from Excel to modern things like Looker.

[0:36:59.1] DM: Sure. I mean, I do think that it follows those waves. In that first wave where you had the data appliance, you had a very locked down BI tool that could make those couple of dashboards. You bought it from the same vendor that you bought the data appliance from, because it all had to work seamlessly together. Otherwise, it wasn't going to work at all.

In that second wave as people start to get desktop computers that are reasonably fast and departmental servers, you see things like Qlik and Tableau start to come out, which are really powerful tools for doing self-service data analysis on data extracts, right? They come with their own data engines that expect you to pull part of the data out of the data warehouse loaded into their data engine, which is optimized for slicing and dicing small to medium sized data and do the analysis right there.

That's what leads to the data chaos is the fact that the linkage between the core data source and the analysis gets broken, but wasn't a dumb decision. It was just the decision that was necessary given the technology of the time. Now I would say we're in this third wave where all of a sudden, you have the ability to leave the data where it lives right in the data warehouse and still access it directly, ask questions of it, make sense of it. That really is the perspective that Looker comes at this problem from.

When I was at Upworthy, we were building our data stack from scratch, from nothing. We got ready to figure out what should be on top as the BI layer. Just serendipitously, an investor of ours said, “Oh, I know these folks in Santa Cruz, California. They're just getting started. You should talk to them. Check out what they've got.” I got on the phone with them and they connected to our database and said, “Ask me any question.” I said, “Oh, this is going to be hilarious.” They're just going to fall on their faces.

I asked them a question and sure enough, they answered it pretty quickly and I said, “Well, all right, but that was an easy one. What about this one?” Then they answered that question. Then Lloyd, who's Looker’s founder said, “Well, but have you thought about slicing the data this other way and looking at it this way?” All of a sudden, he was showing me something brand new that I had never looked at. I was like, “Oh, my goodness. This is it. This is the thing.”

We ended up being Looker’s customer number 22. The core insight of Looker is that you can leave the data where it lies, you can query it directly in that database or data warehouse, you can leverage all of that power. Unlike that second wave, you can put a data model, right? A translation from raw data to business metrics in between people and the data. Generally, that middle layer is analysts themselves. They're the ones who have to keep track of, “Okay, how are we – what's the formula for a lifetime customer value now?” “Okay, it used to be this, but now it's this,” right? They keep those formulas, those recipes in their brain as SQL.

Looker has this thing called LookML, which is a markup language built on top of SQL; just an abstraction of SQL that lets those analysts get that knowledge about what the data means to the business out of their head and into software, where everybody can access it. That's the core idea of Looker, right?

When I'm talking to engineers, I often talk about it as an ORM for analytics, right? Engineers are quite familiar with the idea of an object relational mapping, where they're breaking the tight bond between the application they're building and the data structure and the way that it's in the data storage layer, so that they can change things without having to worry about going and tracking down every time that this field was accessed from the application.

Well, Looker does the same thing, but it does it for analytic data, right? It says, “Look, the definition of lifetime customer value should be defined in one place in code, which then let's Looker write the appropriate SQL. If we have to change that, we'll change it in one place and that will take effect everywhere.” Once you've got that code-based middle layer, you can do a lot more. You can add git version control, which has been there with Looker since the very beginning, which is a new concept in BI. It's something that obviously – version control goes way back in engineering, but in data it's a new concept, because data analysts are used to writing SQL and SQL, while I adore SQL, I think it's an amazing thing that has lasted for 40 years and will probably last for 40 more.

It does have a lot of drawbacks. It's not really real code. It's not meant to be executed and it's very, very difficult to read code, a SQL that even you've written. More than a couple of weeks ago, you go look at it and you go, “What was I thinking? Why did I write it that way? You know what? Forget it. I'll just start from scratch again.” Certainly, reading somebody else's SQL is near impossible.

Analysts got used to having these, just endless text files on their desktop with, “Oh, that's untitled43.SQL and that's untitled44.SQL.” That's just not a scalable way to collaborate and keep track of what's been happening.

[0:42:11.2] JM: I want to understand the idea of the data model in even more detail, because as you said, this was something that Looker did that was novel, compared to the previous business intelligence tools that were in wide deployment at the time when Looker was first created. This data model, this idea of an ORM for analytics, if people are thinking about what makes this, because people have heard of a million different BI tools, but if people are trying to figure out what did Looker do that was different and differentiated. Explain a little bit more what that data model is and how business analysts are going to create a data model.

[0:42:53.8] DM: Sure. To be clear, the idea of a data model is not new, right? The idea of a data model is that you don't want raw data. You want data shaped the way that the business needs it. Whether that's like I said, lifetime customer value, which probably is a combination of a bunch of different fields from a bunch of different data sources, you need a way to define that.

The difference that Looker brings to the table and that is only possible because the data warehouses and databases have gotten so fast is that definition can be divorced from the data itself. The way that a data model was constructed in the old days was by tying it up with the data. The way that you'd get lifetime customer value is you'd actually transform all of those different fields and physically write the lifetime customer value as a column into the database. Because that way, it was already calculated, it was computed and you could access it in something approaching real-time.

The problem with that is by moving all of that data around and transforming all that data, that's very onerous, it's slow and it is very inflexible, right? It locks you in to that particular definition of lifetime customer value. If and when your definition of lifetime customer value has to be tweaked, you have to go and reprocess all that data. Not only do you have to reprocess all the data to recompute lifetime customer value, you have to go figure out what are all the downstream fields that rely on lifetime customer value, because we have to recalculate them too, right?

What Looker does is it says, “Well, we can make the data model virtual, right? We can have it sitting apart from the data, so that you can do those transformations at runtime. That wasn't possible before. Not because of Looker, but because the database wasn't fast enough. What Looker gives you is a tool, LookML, which is a markup language to write your transformations and write your data model in SQL, but to do it in a way that makes it sustainable version control that makes it collaborative, that makes it possible to scale and looker because it's founded by a couple of engineers, takes the position that code is a really good way to do that, right? That we don't want a visual medium. We want code, because code has a lot of advantages in that respect.

The way that a data analyst, or a data analyst team, a new Looker customer goes about doing this is Looker or looks at the way that the data is structured in the database and builds a really basic data model for you. It says, “Oh, here's user ID over here in the orders table.” There's a column in the user’s table called ID; those are probably a primary key foreign key relationship, I'll write that join for you.

Then the data analysis team comes in and starts elaborating on that model and they say, “Okay, but I want a field called lifetime customer orders, so I'm going to make a little fact table and I'm going to I'm going to write what the definition for that is.” I have a first name field and a last name field, but I actually want a full name field. I don't want to have to write a new field into the database. I just want to concatenate first and last name. They write all of that stuff in LookML, which then is effectively teaching Looker how to write the SQL for them.

Analysts don't really like being SQL monkeys. It's not actually the most fun part of our job when someone comes to us and says, “Hey, can I get sales by region for the last six?” You give it to them and then they come back and they go, “Actually, I need sales by country for the last nine months,” and so you have to go back and rewrite the SQL query. Then they come back and they make another slightly different ask. That's not a fun part of our jobs as analysts.

The idea that you could outsource that to software is wonderful from an analyst perspective, because I really don't want to be doing that. Frankly, business users if I give them an interface that's point-and-click and says, “Here's the region button, here's the country button and here's the filter where you can define how many months you need, and then you just hit run and Looker will write that SQL for you and you won't have to bother me, that's wonderful.”

[SPONSOR MESSAGE]

[0:47:07.1] JM: We are running an experiment to find out if Software Engineering Daily listeners are above average engineers. At triplebyte.com/sedaily, you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast.

I will admit that, though I’ve gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself. If you want to take that quiz yourself, you can help us gather data and take that quiz at triplybyte.com/sedaily.

We have been running this experiment for a few weeks and I’m happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that

everyone who has taken the test on average is three times more likely to be in their top bracket of quiz scores.

If you're looking for a job, Triplebyte is a great place to start your search, it fast-tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself. I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8-bytes.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW CONTINUED]

[0:49:05.8] JM: Dive a little bit deeper into that example. The example of somebody from the sales team coming over to the business analyst and saying, "Hey business analyst, I need sales by region for the last six months." Then the business analyst spends a lot of time writing that query. Then the salesperson comes back and says, "Actually, I need sales by country for the last nine months. Your query was not sufficient for me." Then the business analyst has to rewrite the query in SQL. You're saying that Looker has some features that help save time in that interaction. Give a little bit more detail there. How is that time being saved?

[0:49:44.1] DM: Sure. It's being saved, because the analyst was involved once in defining the data model and in saying, let's say, "Here's where the country field lives in the database, because that's a physical field in the database," but then I will assign countries to regions which is not something that lives in the database. Let me say if it's this country or this country, it's in this region. If it's this other country, or this other country then it's in this region, maybe I'm writing a case statement, and I write that once in LookML, into Looker.

Now anytime that anyone clicks the region button on the Looker interface that's presented to business users, not the back end but the front end, Looker automatically knows how to write that SQL that puts the right country in the right region. If I need to update that region field, because we reclassify one country into a different region, I can do that in one place in code and then every dashboard that uses that concept of region every time that somebody writes a query,

every time somebody asks a question that has that concept of region, Looker is going to update that and write the appropriate SQL that's been the most updated version.

It's a very dry model, right? It's a very don't repeat yourself model, which again, is a great engineering concept that for whatever reason just hasn't made its way into data analytics. Looker is bringing that concept to data analytics and saying, “That's really brilliant.” We don't want to have to define something six different times, or a hundred different times in a hundred different places, because then changing it is really painful. That is exactly what leads to data chaos, right? It's everybody having their own little workbook where they're doing their own data analysis and trying to remember, “Oh, what was the latest definition for region? What was the latest definition for lifetime customer value?”

When they don't remember that, or when they remember it wrong, or when they do it and it's right now, but then six months later it's wrong and there's no automatic way to update it, that's what leads to data chaos. Having all that live in one data model that is central that's governed, that's version controlled, that everybody has their own sandbox, their own dev version of it, so they can try new stuff out without affecting production. Having all that stuff in software is what allows everybody to stay on the same page and what gets you to that single source of truth, rather than a hundred different sources of truth that live in a hundred different workbooks.

[0:52:09.8] JM: There is sometimes the distinction between the role of a business analyst and a data scientist, maybe a data analyst. More and more companies have these kinds of roles. Maybe you could say data scientists write more code, business analysts spend more time in front of business intelligence tools. There's some work around machine learning algorithms that maybe data scientists are doing more often. Tell me about this range of data rolls as you've seen them and how do these different people use business intelligence tools. I guess, also do – is business intelligence useful for doing machine learning?

[0:52:51.8] DM: Yeah. I mean, I think on the title side I would say I don't want to step into them [inaudible 0:52:57.0], because it is an ongoing and hilarious at this point. It's like an ironic fight. It's such an old fight that it's become a joke. “Oh, what a data scientist is this and a statistician is this and a mathematician is this and a data analyst is this and a business analyst is this.”

I think the core idea is yeah, people bring different skill sets to the table, but the reality is and any data scientist worth their salt will tell you this; most of a data scientist time is spent cleaning data, right? Shaping it and extracting features and getting it ready for machine learning. Running the actual machine learning algorithm is not very difficult. Tuning it can be more interesting, but actually running it is not super hard and certainly not time-consuming.

The thing that's time-consuming is cleaning up the data and getting it ready. The only way to do that effectively is to have subject matter expertise in what the data is supposed to mean, right? That is something that any machine learning algorithm that you're bringing to the table needs. You can't just feed in a jumble of raw data and hope to get magic out the other side. You need data that makes sense that answer something for the business. Otherwise, what you're going to get back from the machine learning algorithm is, “Hey, seems like there's some correlation between month and temperature.” It’s like, “Yeah. Well, thanks. Awesome.” Not very helpful, right?

You need to give it something that makes sense. Data scientists, a lot of the time what they have to do is they have to go to the business owners and say, “Tell me what this means, or tell me how this data is supposed to be shaped, or what the features that I should feed in are,” because they can't possibly be subject matter experts in every possible part of the data that their business cares about.

What a data model gives you is it gives you data where somebody has already gone in; gives you metadata, where somebody has already gone in and given the data meaning and said, “This is what this means to our business. Don't feed in raw number of purchases. Deed in lifetime customer value instead.”

Whether you're a data scientist, or a data analyst, or a business analyst, or a statistician, or a business user, you're still going to care about that core concept of taking the data from its rawest form and turning it into something meaningful to the business. If you're a data scientist, that's really going to accelerate your process, because you don't have to start with the raw data and cleaning it and pulling out all the failed transactions, because they've already been pulled out by the business analyst.

I think a collaborative approach to getting that data from its raw form to something meaningful makes everybody's job better and easier, because those business analysts and those data analysts are often the ones who do have the subject matter expertise, right? The marketing analyst knows what the marketing data is supposed to mean. Data scientists might look at the marketing data and go, “Yeah, that looks fine.” The business analyst would look at it and say, “No, no, no, no, no. Something's really wrong with that data, because we never spent 10 trillion dollars that year on advertising.” The data scientists would say, “Well, it's just a number. I don't know if it's right or wrong. I'm just working with the data as it was given to me.”

You need that subject matter expertise, because those are the people who spot the problems and the things, both the important stuff and the stuff that's problematic before it gets into the machine learning algorithm.

[0:56:20.3] JM: Today as we've reviewed, the tools are getting a lot better, the engineering practices are getting better. There are organizations that of course still have trouble building predictive analytics, or data science throughout the company. Why is that? What are the challenges that they're running into and what are they missing from their tools?

[0:56:42.3] DM: Yeah. I mean, I am a bit of a fuddy-duddy about this. I think that most organizations are probably not doing predictive analytics particularly well, and that's okay. I would much rather them focus on the immediate term problem, which is probably not just that they're not doing predictive or prescriptive analytics well, but they're probably not even doing descriptive analytics well, right?

They probably have lots of data that would be meaningful and helpful and help people do their jobs better that those people can't access, or at least can't access in a reliable and timely way. Rather than chasing the shiny object that artificial intelligence or machine learning has become, I'd encourage people to just talk to the people across their company, talk to their colleagues and say, “What is it that you would need that would help you do your job better?” What is a question that you wish you had an answer to that maybe lives somewhere in the data?”

When you talk to – I've had a number of friends who are data scientists, very talented data scientists, and they get hired for data science jobs at companies that are ready to invest in

predictive analytics and data science. They get on-site and they say, “Great. It's my first week. I'm a data scientist. Give me the data that you want me to data science, right? Give me the data that you want me to build some models around.” The company says, “Well, our database is over there. Good luck.”

There's nothing there. There's nothing ready for them, because that company hasn't actually tackled the real problem, which is that nobody knows what lives in the data lake, nobody knows what lives in the database, nobody knows what it means. The data scientist spends six months doing very highly compensated work just doing basic data janitorial stuff. I think that's to some extent, a waste of the data scientist’s time, because they have this special skill set and it's really, really common.

I think if you have a really great data analytics program at your company and you feel people across the company are getting access to the data they need in a timely manner and that data is accurate and reliable, great. Go hire a data scientist and say, “Hey, are there places where you think your skill set would help us? We can build predictive models and go even beyond what we're already doing.”

I would say the large majority of companies do not have that fundamental piece in place. I'd say stick to the basics. Just figure out what data you're collecting, figure out who needs it, figure out how it needs to be transformed to give meaning to those people and answer their questions that matter to them and doing their jobs, and I think you'll get a lot more value for a lot less time and money doing that and tackling that problem first.

That's not something that, “Oh well, only startups don't have their act together, or only big enterprises don't have their act together.” I've seen so many companies that just don't have this basic thing down from the biggest tech giants, where parts of their operations are very smooth and they have data analytics working great and other parts are a mess, to hip, new database startups to old guard companies that been around a hundred years.

If you're somebody who said, “Really seems like our data systems are not up to par. Everybody else seems like they're doing magic and things are going much better for them than they are

here," you're not alone. You're well within the majority. That's where I would encourage people to spend their time and energy, rather than chasing data science.

[1:00:25.1] JM: Is there something to be done there from on-high, like an imposition of well-organized, clean data inputs? Everybody must adhere to these clean data input rules, so that we never have any data cleaning tasks to be done ever again. What's the solution for getting that – avoiding getting into that state where you have a data lake and nobody understands what's in it?

[1:00:49.8] DM: Oh, we're only at that easy. That would be wonderful. It's not realistic, sadly. I do think that a commitment from leadership to being data-driven is really important, but I think the reality of data is that data tends to be messy. It is important to have checks all along the data pipeline to weed out the bad data, which can very quickly lead to a loss of trust in your data analytics program and to bad decisions based on bad data, and to just accept that yeah, data is dirty. The more humans involved, the dirtier the data is going to be. There's just no way to avoid it.

It's part of the job. It's in some ways, part of the – one of the cool parts of the job when you can – when your data cleansing is working nicely and you now that you're stripping away all the dirty data that you don't want in your analysis. I do think that a commitment from that leadership team saying, "We are going to be data-driven. We are going to invest in this." You do need to track what you're doing with data and then we will turn that around and give you reporting and give you access to that data that you need in order to do your job better, that's really important.

Then you also need the bottom-up resources to actually tackle those problems and make things better. If you're lacking either of those things, I think you're not going to have a ton of success, but I think more and more companies are realizing that the gains to be had here for this investment are really huge. Having reliable data is just – it's a big deal. It really changes the way people work. It's nice to see those investments coming to fruition, but I do think that the idea that you can just mandate that everyone only input clean data, good luck with that.

[1:02:39.1] JM: Okay Daniel. Well, you are the Chief Data evangelist at Looker and you spend a lot of time talking to different customers and you also spend time with the product

development teams at Looker. I'm sure you're a point of communication between the real world and the product developments at Looker. Could you close off by just giving us some of the future of what product development you're working on and what's in the future for Looker?

[1:03:06.4] DM: Sure. I think we at Looker have had the same vision, the same philosophy from the very beginning and that was the philosophy that was sold to me when I was a customer very, very early on, which is of doing data analytics right, of keeping everybody on the same page, but giving everybody access.

I think the thing that we have been realizing more and more over the last year or two and that we will be investing in over the next few years is that because the databases, data warehouses are so powerful and so fast, there are more and more touch points within any organization where people actually do want data, right? Where what they are using is fundamentally a data application, right? Whether that's something like Salesforce, or New Relic, or Datadog, or Gainsight, or Anaplan, or just like, there are all of these applications that are fundamentally about inputting data and then getting meaning back out from that data, right?

If you're if you're a DevOps engineer and you're running New Relic, you're looking at data about how your servers are doing, and exceptions and utilization and things like that. I think Looker's position is that having a common surface that can pull all that data together is incredibly valuable, because a lot of the biggest insights come from combining data from multiple sources, combining the payments data that tells you how much you're spending on your servers with the monitoring data that tells you how those servers are performing.

Combining the marketing data and adspend data that tells you where customers are coming from with the transactional data about how much they're spending at your store. Looker really, I think is uniquely positioned to be that common surface, to be that data platform and to power a lot of those different data experiences, which do need to be purpose-built and do need to be very easily grokkable for the people who are using them. One of the things that we're really investing in is that platform.

Building out custom-built, or purpose-built, not custom-built but purpose-built experiences that use data to present people with the analysis that they need, whether that be a marketing

analyst, or an ad campaign manager, or a website manager, or a DevOps person and thinking about how can we build something that feels very familiar and comfortable to them where it doesn't feel necessarily like a BI tool, because maybe they've never used a BI tool before. The idea of having to point-and-click their way around would be intimidating and a hard place to start. Giving them an interface that feels really familiar, but is built on that common platform is something that we're really excited about and that we've made a lot of progress with.

Already we've rolled out our first two beta applications. One is it's that marketing analytics one that pulls together your data from Facebook ads and Google ads and LinkedIn ads and Twitter ads. Another is more for website analytics that or really don't feel or look standard Looker, but instead feel like something that a marketer is going to feel immediately at home in. I think we'll continue to invest in those.

Then will also continue to invest in the platform that sits underneath all of those, because we know that there are way more data applications out there than Looker can ever build, and we want to make sure that we give companies the tools that they need, so that they're not just buying Looker the BI tool, but buying Looker the data platform, on which they can then build additional applications that meet their specific needs.

[1:06:49.0] JM: Daniel Mintz, thanks for coming on Software Engineering Daily.

[1:06:51.6] DM: My pleasure. Thanks for having me.

[END OF INTERVIEW]

[1:06:56.0] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CICD, built-in monitoring and health management and scalability.

With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure, as well as your own on-prem hardware. OpenShift from Red Hat gives you Kubernetes without the

complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

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

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

[END]