EPISODE 1002

[INTRODUCTION]

[00:00:00] JM: Large companies generate large volumes of data. This data gets pumped into a data lake for long-term storage, then pulled into memory for processing and analysis. Once it is in memory, it's often read into a dashboard, which presents a human with a visualization of the data. The end user who is consuming this data is often a data scientist who is looking at the data to find trends and design new machine learning models.

Another kind of user is the operational analyst. An operational analyst is creating complex queries across this data to find latencies in the infrastructure or perhaps slicing and dicing clickstream data that's coming from online advertisements in order to figure out how to tweak those advertising algorithms and spend money more effectively.

For an operational analyst a key use case for a data warehouse is fast interactive querying. The operational analyst needs to be able to query the data quickly to create a dashboard, make judgments based on that dashboard and then change the query slightly to look at a slightly different dashboard.

Druid is a high-performance database that is used for these kinds of queries. Druid is used for ad hoc queries and operational analytics. Imply Data is a company that builds visualization, monitoring and security around Druid and the applications that people build on top of Druid.

Jed Naous is a vice president of R&D for Imply and he joins the show to talk about the use case for Druid, the architecture, and the business model of Imply. Imply is hiring. So if you are curious about the technology, you can check out the jobs that are available. If you enjoy the show, you can find all of our past episodes about data infrastructure by going to softwaredaily.com and searching or the technologies with the companies that we discuss in this episode. If there's a subject that you want to hear covered, feel free to leave a comment on the episode or you can send us a tweet @Software_Daily.

[SPONSOR MESSAGE]

[00:02:07] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[INTERVIEW]

[00:03:44] JM: Jad Naous, welcome to Software Engineering Daily.

[00:03:46] JN: Thank you.

[00:03:47] JM: You lead R&D at Imply, which is an analytics system built on the Druid database. The end user of Imply is often an analyst, a business analyst of some kind. Take me inside the job of a typical analyst.

[00:04:02] JN: Well, actually do the typical user of Imply is not an analyst. Generally, analysts usually interact with BI tools or traditional BI tools, like Tableau, or like Looker, PowerBI. Imply is not really focused on that set of people yet. Mainly what we are focused on delivering is analytics capabilities for operational level people. Let's say you're an engineer and you need to look at the performance of your cluster and identify which machines seem to be exhibiting a behavior that’s problematic. What versions of the software, like what combination of software versions and configurations are causing a particular problem? Things like that. If you're a marketing, say, you're like a growth hacker or you’re doing advertising you're doing AB testing and you want to understand what segments of the population or of the audience are actually interacting with a particular ad versus others and trying to optimize the performance of those ads.

In these kinds of situations, you have very complex datasets. Each kind of event or each click may have hundreds of fields potentially and what you're trying to do is looking at some aggregate metrics for certain number of events over a period of time and then trying to find what are the common dimensions or the common fields within those events that seem to exhibit similar behaviors and which ones aren't. We have a whole bunch of customer. One of the one of our biggest customers is Twitter and their main use case is around ads.

[00:05:50] JM: Tell me some of the typical queries that might be issued to this database.

[00:05:55] JN: For the most part, these types of queries are less about like dashboard inquiries and more like slice and dice analytics. The example would be – I’ll take an engineering one, because those are always the easiest for me. Give me the request latency, the 99th percentile request latency over my entire cluster over the past, let's say, two months, and then drill down through the past two days. Compare the past two days against like the two days a month ago. Then zoom in to a particular spike, let's say, that you're seeing in the past two days. Break it down by, let's say, AWS regions and then break it down by Java version and then compare those versions to each other and figure out which of the Java version seem to be having problems versus not.

Those are kinds of the queries, like a very simple workflow for the things that you're trying to do. But imagine you're trying to do this over hundreds of these dimensions and trying to figure out what group of dimensions or kind of key value pairs are the ones that are problematic.

[00:06:59] JM: There's been tooling for issuing these kinds of queries for a long time and you're in a business where you’re competing with some of these products, like data warehousing products. What are the traditional responsibilities of a data warehouse? How does that compare to the responsibilities of Druid?

[00:07:18] JN: Great question. If you look at what time series databases do, they're really focused on providing these fast type real-time, like super real-time analytics where you put an event and it's supposed to – The actual metric should be impacted almost immediately. You should be able to get alerts on those metrics pretty quickly and you should be able to store tons of these metrics, like billions of metrics.

That's kind of like one point in the design space for kind of data stores where you want super real-time, superfast, but the complexity of the queries that you can do and the complexity of the data that you can get is very limited. On the other hand, you have the lease very capable, very powerful analytical databases, data warehouse like BigQuery, or Snowflake, or Red Shift where you can execute these super complex joins on vast amounts of data, but they take a really long time to execute.

Druid is kind of the open-source OLAP real-time data – Well, I call it a real-time data warehouse that we are responsible for, because we are the main company behind it. Today it sits somewhere in between a time series database where the number of dimensions is small and a full data warehouse where you can do these super complex queries.

Druid is real-time and high-performance in the sense that you can get results at interactive speeds so you can actually execute these kinds of slice and dice analytics and queries pretty quickly. A human can sit in front of a UI and then start clicking on things, drag-and-drop, and execute these kind of drill downs and aggregations and so on in order to do these kind of root cause analysis explorations. But at the same time, it's powerful enough to get a good subset of the use cases that people use data warehouses for.

Mainly, all these use cases are like really operational use cases, like people or like performance related use cases, like people are trying to get kind of like the performance of a certain store, overtime, the performance of a certain item within a store, like these kinds of things. How credit card transactions are working? Who's executing them? What are people buying for these transactions and so on?

Those are kind of the use cases that we see. That's kind of the scope. You've got data warehouses on one side. You've got time series databases on the other. In terms of technology, Druid is kind of in between these two. What we're seeing happening is that more and more use cases that people actually care about are moving towards real-time and more and more of the people who want to actually use data to make decisions, like day-to-day, are these operational type people, like marketing people, like customer success people, or support, or engineers.

What they're finding is that data warehouses can't fulfill this really interactive need. So they start to move and look for something that is more capable of interactive type queries that are fast enough and they find Druid. Today, Druid is very much kind of focused on this hot data where you look and find the data that is most used by people. People want to query it all the time. They care about it being fresh. They care about it being available all the time. They move that data and they put it in Druid.

At the same time, people are starting to push us towards the data warehouse use cases, the traditional data warehouse use cases because they see how fast they can execute queries in Druid and then they say , "Well, why can't you do this one more query and this one more query and like this additional capability and like add this additional function?"

Bit by bit we are actually moving more and more into the traditional data warehouse space. The vision that we think we are building out at Imply is this idea of this operational real-time data warehouse, and that's not something that currently exist. Today, you have either time series databases, which are really operational. I mean, Druid has been here for a while, but like when people think about high-level concepts and the things that currently exist, they have like, "Well, okay. It's either a time series database or it's a data warehouse." But there's actually something in the middle which is this operational real-time data warehouse which is druid.

Overtime, we're adding – In our next release we're going to add [inaudible 00:11:52] star schema joins. That's kind of pushing us bit by bit towards the things that data warehouses traditionally have been doing.

[00:12:01] JM: Architecturally speaking, as I understand Snowflake and BigQuery, you can treat them as – And correct me if I'm wrong, like a unified data lake data warehouse. You can dump all your data into them and they take care of figuring out what should be on disk or what should be pulled into memory and a lot of these based on when you're querying it.

[00:12:22] JN: Yup.

[00:12:22] JM: Druid, you fit it into your data infrastructure in a way where you're doing routine ETL jobs into Druid and you also have data streaming into it from another angle, like streaming in from Kafka or something. Can you tell me how it fits in to your architecture? How are you dumping data into Druid and when are you dumping data into Druid?

[00:12:48] JN: There's two ways of putting data into Druid, either you do it through streaming ingestion like you had mentioned with Kafka or Kinesis. You just pipe the data straight from Kafka into Druid or you do it through batch ingestion. What that means is you have your data sitting somewhere. It's usually in like parquet files, or ORC files inside a data lake, or in S3 buckets, or HDFS, or whatever. It used to be that you would get Hadoop, drawn certain who do jobs that are controlled by Druid to read those files and suck them into the druid format.

Sucking those files actually either through a stream ingestion or batch ingestion, what we are doing is we are reading all the data and adding a whole bunch of indexes and optimizing it for the types of queries that we want to do and then we dump it into what we call deep storage. Deep storage is usually another HDFS or S3 bucket, but that's basically where the main store kind of highly available, reliable store for all this data is.

The way Druid works is you have a bunch of historicals. We call them historicals, and these are really the worker nodes. Each of them will be responsible for a subset of the data. This data is stored as segments. We call these each kind of packet of data that's stored as a segment. Each historical is responsible for a number of those segments. They load them and then they serve them to any queries that come in.

It used to be that you used to need Hadoop for doing a lot of this ingestion. What we've done in this release that's coming out tomorrow is really focus and double down on building batch ingestion within Druid such that you don't need this external Hadoop system to actually do this batch ingestion. Everybody knows Hadoop is just really hard to run.

[00:14:52] JM: Does that mean that a user that already has a Hadoop cluster, are they typically copying all their data from HDFS into Druid?

[00:15:04] JN: They would usually select the data that they care about and they would copy it into Druid. What we see is people often timing out the data that's in Druid. It kind of overtime kind of ages out, and so people would store – In Druid, you can actually have tiering of your historical nodes. You can have very expensive historicals that have a ton of memory such that the entire data that they load fits in memory so you can actually execute queries really fast and then you can have a lower performance node, which maybe serves data from SSD's and that could be for things that are maybe like a month or three months old or something like that and then something that has magnetic disks and that would be for things that are maybe a year old.

Generally, people don't – In these operational use cases, people rarely care about data for multiple years or things like that. They usually wouldn't load that amount of data into Druid, especially given like – Because usually the queries that people want to do on Druid, they want them to execute superfast and so they have to balance the resources and figure out what data has to be in memory versus not in memory, things like that.

[00:16:23] JM: Just to clarify the terminology, the historicals are the data that you're actually going to be querying in the operational use case, whereas you have might have the raw data after it's ingested, written to deep storage. The deep storage is not typically queried, right?

[00:16:42] JN: Let me re-describe this in a way that's a little bit more organized. I kind of was trying to selectively describe the architecture. The way Druid works is you usually have some raw data coming in from somewhere, either through Kafka that you want to ingest just straight into Druid, or you have a bunch of data that's sitting somewhere in parquet files or ORC files somewhere in HDFS or in S3 or like your data lake. Those files that are sitting in your data lake actually could be every once in a while updated.

Some of our customers, for example, like update these files like once a day and so they want to re-ingest those files into Druid such that Druid has kind of the daily fresh version for execs to look at dashboards and things like that.

The way Druid works is in order to actually create the data within it, it has to suck that data through something that we call ingestion. If you’re sucking it in from Kafka, then it’s stream ingest. If you're taking it from files or something where like you have to actually launch a task every once in a while to do it, so it's not a real-time task, it would be batch ingestion.

The way druid operates is that it has this thing that it calls a segment. A segment is a container of the data. It's kind of like a file or something like that. It’s an index data structure that contains the data as well as indices to make it much faster to query that particular segment. A segment usually is assigned to a certain time interval. Data could be partitioned into different segments overtime, things like that.

As this data gets ingested either from Kafka or batch ingest, segments are created. The segments go into something we call deep storage. That's where those segments are stored reliably and highly available and they’re, to a certain extent, permanent. Overtime, as people can set retention strategies for data, and so over time like segments can timeout.

Now, in order to serve those segments, there are components or processes, servers, that we call historicals. You can have a large number of historicals. The design of Druid is such that the historicals can be stateless. A lot of people, we see them run these historicals within Docker, or in Kubernetes, or something like that so that they can quickly scale up and down as needed.

The way historicals work is they come up somebody. I'm not going to describe the whole control system, but somebody tells them, “This is the time range that you’re responsible for and the data that you're responsible for,” and they go and find the right segments from deep storage and then they load them to serve queries.

As queries come in, they go through a node that we call the broker, a type of node that's called a broker. That broker parses the query. Figures out what historicals are responsible for what datasets and then goes and queries all those historicals and collects the data back and merges that data and returns it to the user.

The historicals, in order to execute the query, they load the data that they acquired from the deep storage into memory and then they execute those queries potentially segment by segmenting and then return the result for the segments that they actually care about. That's kind of how the whole system fits together, if that makes sense.

[SPONSOR MESSAGE]

[00:20:13] JM: Looking for a job is painful, and if you are in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace to connect highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies access is exclusive and you can apply to find a job through Vetter by going to vetter.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you.

No more of those recruiters sending you blind messages that say they are looking for a Java rockstar with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery.

Vettery is changing the way people get hired and the way that people hire. So check outvettery.com/sedaily and get a $300 at bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily.

Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:22:02] JM: Could you frame it comparatively to Snowflake and BigQuery or perhaps Red Shift again? Why is it able to outperform those data warehouses?

[00:22:16] JN: That's a question I'm not going to be fully able to answer. That's a question I've wanted to look into, but I'll give you a hint of why. I'll give you a little bit of a flavor for how I understand it.

[00:22:28] JM: Sure.

[00:22:28] JN: The larger cloud data warehouses need to have two restrictions that we don't. The first one is they have to store data pretty much indefinitely. They have to be designed in such a way that –

[00:22:46] JM: This data lake data warehouse combination thing.

[00:22:48] JN: Maybe I should talk a little bit what's a data warehouse and what's a data lake. Would that be a good description?

[00:22:52] JM: Sure.

[00:22:54] JN: A data lake is usually a place where you just dump files. A lot of people have – For a long time, people would just bring up something like HDFS and then they'd start dumping files into it. They don't know whether they're going to use those files or not, and then –

[00:23:11] JM: It's cheap. It's on disk. Throw it all there.

[]00:23:13JN: Exactly. Then they hope that like something like at some point in time somebody's going to create some system that makes it easy for them to get the answers that they want out of those files. There's been things like Hadoop that doesn't seem to work that well. Then people started using Hive, which turns out to be pretty slow. Then people started using things like Presto, which is doing a much better job, but still kind of slow. Then they started building things – Using things like Spark and so on. Anyways, those are kind of like the ways to extract data from. It's not nice.

[00:23:46] JM: Yeah. Because for those ad hoc queries, you're ultimately reading files from disk and it's going to be expensive. In contrast to data warehouse, you actually can pull it in memory and then do ad hoc queries and it's much faster.

[00:23:59] JN: Exactly. For data warehouse, you're actually dumping the data and the data warehouse, basically you give it data through their ingestion process and the data warehouse looks at the data and it actually indexes as you decide how it should index this data and then dumps it into an optimized storage format for the way it executes queries, whereas that doesn't happen in data lakes.

Those are kind of like the differences between a data lake and a data warehouse. To me, a data lake is pretty much like a file store somewhere. It just expands. It's basically an unlimited pool of – An unlimited storage space for files. What was your question?

[00:24:41] JM: I was just wondering about the performance, how Druid is able to outperform the databases thanks to architecture, but we can save that for another podcast. But we just did a show about Presto. I'm glad you have kind of put this in a nice contrast. It sounds like Presto, performance-wise, sits somewhere in between the performance of Hive and the performance of a data warehouse, or the performance of Druid. Have you looked into Presto significantly?

[00:25:11] JN: Yes. Presto's performance sits somewhere between Hive and traditional data warehouse. Operational data warehouse like Druid sits actually even beyond the regular data warehouse. Presto is designed to be mainly like a federated SQL engine and it's limited by the performance of the storage locations and like the storage formats and things like that. Data warehouses don't have the same problem. Specifically, Druid doesn't.

[00:25:46] JM: Got it. Talking at higher level, when you're talking to customers, for example, I was looking at Airbnb's use case, how did they adapt Druid? Is there like a particular use case they find that their data warehouse is not fast enough for? What pushes them to adapt Druid?

[00:26:08] JN: Airbnb specifically, the use case that the use it for is performance monitoring. If you remember what I said about time series databases, they're nice and that they're fast, but it's actually not that great to try to debug or kind of do analytical queries on a time series database. It's really hard to figure out things like clustering and so on when you have very highly dimensional data. What you need is like more OLAP type queries and data cubes. Time series databases don't have that.

They were thinking about, "Well, what actually allows us to do pretty complex aggregation and slice and dice queries on highly dimensional data but still be real-time?" That's how they found Druid. The reason they like it is because you can connect it to Kafka and as soon as a Kafka is – An event is available through Kafka, it can be ingested and available for querying in less than a second.

That is the type of performance they're looking for, because when there's a problem in their clusters, they want to be able to immediately identify that issue and alert on it and then very quickly go and troubleshoot and slice and dice the data to figure out what, let's say, machines are causing that problem.

[00:27:28] JM: So the events get pulled off of Kafka. They get indexed in Druid really rapidly, and then if there's an issue – There's like alerting systems that are hooked up to Druid. The alerting systems can do anomaly detection or whatever.

[00:27:44] JN: Yes. It depends on how – I don't exactly know how they do alerting and anomaly detection. I can't speak to that.

[00:27:52] JM: Okay.

[00:27:52] JN: I think they might be actually doing it. I'm not going to speak to that.

[00:27:55] JM: Yeah. Fair enough. Now, from a product standpoint, Imply a built its products – Imply, which was founded by the guy who created Druid, Fangjin. The product was created by integrating Druid with full boarding and visualization. Why take that product route? Why not just sell the database and let other people build BI tools and dashboards on top of it?

[00:28:21] JN: Great question. If you actually think about a lot of the operational use cases and the types of interactions people want with data, you'll find that most BI tools are pretty limited. Specifically, the type of interaction that you want is this kind of data cubes slice and dice interaction where you can like drag-and-drop and be able to break out dimensions from aggregates and so on. There is no BI tool that does this out-of-the-box generically across kind of data in general. You'll find that there are kind of vertical specific tools that do these kinds of things.

For example, I was at AppDynamics, we built something to a certain extent similar to that that's now called business IQ. That was based on Elasticsearch. You see it in other systems, like Amplitude Analytics has a similar thing, but they have their own internal database. But there is no kind of system that just does this for everybody.

That's why the team ended up getting Pivot out of meta-markets where Druid was initially created in addition to Druid itself. The reason that they really took out Pivot and want it to make it available for other people as well is because they wanted to showcase the type of things that you could do with Druid. Today, we don't yet have this – The BI tools haven't yet woken up to the needs of these operational people to kind of do this slice and dice analytics. In order to actually provide people with a UI that can do that on top of druid and kind of showcase its capabilities and platelet play to its strengths, we have to build a UI. That's Pivot.

[00:30:10] JM: What about this whole Tableau, Looker, Periscope Data, whatever, Space. I thought those companies did the BA next-generation stuff?

[00:30:23] JN: They mostly focused on creating more dashboards as supposed to interactive exploration of data. You can have somebody going like, "Okay. Well, I need to understand like how a particular thing happens," and then they go into the dashboard building tool or they'd go into the widget building tool and they execute a few queries and then they dashboard that. They kind of create a chart from that data as supposed to doing this, I guess, slice and dice capability, which is like create an aggregate, like maybe you start off with a query and maybe you just start off with show me the aggregate, like total latency and then break that down by regions, break that down by Java versions, drill down into this piece. Compare that to other pieces. This kind of very iterative, very interactive workflow is not one that these BI tools are really focused on at the moment. They're really focused on is like dashboarding, like showing you something overtime. Something that you put on a screen. Something that you would look at to see if things are working normally. Not this kind of interactive mode of working with data.

[00:31:33] JM: When you're setting up like a series of dashboards or series of analyses, Imply or –

[00:31:42] JN: Pivot.

[00:31:42] JM: Is it Pivot open source?

[00:31:43] JN: No. Pivot is the UI. Pivot is just the UI piece of Imply.

[00:31:49] JM: Got it. Okay. The UI product, this is on top of Druid. Does that necessitate Druid serving multiple queries in parallel?

[00:31:56] JN: That's right.

[00:31:57] JM: Is that hard to engineer?

[00:31:58] JN: Yes. For the most part, what we see people using Druid for are these highly concurrent, but potentially low-volume use cases. Something where you have potentially hundreds of people looking at the same – Like trying to slice and dice and understand for the same dataset what's actually happening? What's not? What are our users doing here versus there and so on? That's really what Druid is best at, because it can farm out a lot of the queries and be very efficient in terms of who's executing what query.

Data warehouses don't operate in the same kind of hierarchical way, like strictly hierarchical way that Druid operates in. They suffer from a lot of kind of shuffling of data around potentially in order to execute queries or different people trying to access certain buffer pools within certain nodes that might end up conflicting. That's mainly like a lot of the reasons why the performance of Druid ends up being better for this use cases.

[00:33:02] JM: How much flexibility do you give over to the database operator or the engineer or the operational analytics person, whoever is operating their Imply set of dashboards? How much control do you give them over the underlying infrastructure? Do you want to give them control over spinning up new nodes, or moving stuff to disk, or archiving things, or do you want to take care of all the infrastructure under the hood?

[00:33:30] JN: With Druid being an open source project, we found that people always want to have more and more controls. We have huge clusters that are running Druid, and when you have something like thousands of nodes that are running Druid, small optimizations and small changes can lead to significant cost savings or performance improvements.

For the longest time, we've really focused on making Druid super powerful and easy to operate at scale, but the downside of this has always been that it's a lot harder to start with. It's a lot harder to deploy it in small situations. It's a lot harder to kind of fully understand all the knobs and all the things that you can tweak and how to get the best performance for the things that you're trying to do depending on your scale.

What we are doing at Imply overtime is reducing that difficulty. Some of it is by building additional operationalization features. One thing that we have at Imply is implies something called Clarity. That's our performance monitoring tool for Druid. If you have Clarity, you can actually understand how queries are executing. Why they're fast? Why they're not fast? How to optimize segment sizes, things like that, in order to get the best performance out of your cluster. We also have a cloud version that is already optimized for whatever you need to do so you don't end up having to deal with the difficulties of actually using Druid by yourself.

[00:35:05] JM: What is the experience you give people with Clarity? If I have a query that's doing poorly, do you tell me how to rewrite the syntax of the query differently or just tell me, it's like using too much memory? What exactly are you doing there?

[00:35:20] JN: We can tell you what parts are taking how much time where. We haven't yet gone to the point where we can actually suggest changes. I’m not actually sure how much of a gap we have before we can actually do that. But with the most recent version of Imply, so the one that we are about to release, we have actually moved Clarity to be a part of Pivot itself, and so now you have all the same capabilities that you have with Pivot on top of Clarity. That includes things like alerting in real-time, things like scheduled reporting, much more capable slice and dice abilities than what we had before.

[00:36:00] JM: I’d like to know more about what your role entails, because I understand this is an extremely technical product, but it's also one where you really need to understand this particular user type. I think this is a kind of a new class of user.

My understanding is you sit somewhere in between understanding what the user needs and interpreting that to technical program managers, program managers, and engineers, as well as perhaps filtering the things between the engineers and the customers and thinking up new features or new product ideas. Can you just take me inside how you operate that role?

[00:36:49] JN: Sure. My title is VP of R&D, but for most startups, that's kind of an invented role or invented title. I'm actually head of product and engineering at Imply. I juggle two things. First, I am working on building a scalable engineering operation with processes that lead to consistent quality across the products that we build as well as make sure that the features that we build are delivered on time. That's one half of my job.

The other half is head of product, which means I am responsible for making sure that we are building the right things, in summary. What that means is talking to a lot of customers. Understanding their needs, looking at the market, understanding how potential competition is looking. What does the data space look like? Where do we fit in that space strategically? How do we message the things that we’re building? How do we explain it to people? How do we break the high-level strategy that we’re doing into smaller items that we can then deliver, give engineering to go and build? That’s the product management piece of what I do.

At this point, our company is a little bit small. We don't yet have program managers. The way program managers fit here is when you have a much larger engineering organization, you end up with these kinds of cross-functional initiatives where you have like, let's say, “Oh! Let's redo how we build testing, or how we do CI, or how we execute on this particular feature or something like that, or let's change this architecture. Let's start supporting Java 11, things like that.” They take these initiatives and cannot follow them through and implement the plan to execute them across engineering, product, customer success and so on. We’re not yet there. Maybe some day.

[SPONSOR MESSAGE]

[00:38:59] JM: You probably do not enjoy searching for a job. Engineers don’t like sacrificing their time to do phone screens, and we don’t like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what’s the alternative? Triplebyte is the alternative.

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

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

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

Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[00:41:16] JM: When you're trying to find where exactly Druid fits in the market and where it does not fit, what are the places where you have trouble winning over the customer base? Like maybe you go and talk to them and you're like, "Look, you could totally use Druid. You could totally use Imply," and they're just saying, "Actually, our data warehouses are good enough. We don't really need the additional low-latency. It's not really going to help us. No. Thank you. Come back next year," and you know they could use it. You know they would get value. What are the places where you feel like the messaging or the product – And one of the reasons I'm asking this is because I know there's a lot of people who – Actually, well there's a number of people who listen to this show and they work at like Confluent or Databrix or they're building a data infrastructure company and there's so much change in the data infrastructure world right now that people building products in the data infrastructure space, they really have to hone their messaging, because otherwise you just hit a wall because you're selling to people that might have just fatigue.

[00:42:29] JN: Let describe a little bit of the journey that we're on at this point from the product side. Druid was initially created to serve a particular problem, to solve a particular problem in ads. The problem it was solving is this need for slice and dice analytics. It's necessary for doing things like understanding what segment of the population really likes pink shoes versus which one likes red shoes, and which one cares about the brand versus not, which one responds to an image with two people versus five people in an ad? Things like that. It was created with that in mind to begin with.

Overtime what has happened is we've discovered more and more of these use cases, these kind of operational use cases in different functions in the company, like engineering, the example of Airbnb is one, like marketing, like customer support, things like that. Because there wasn't this operational BI thing, this UI that you could just plug into any kind of database and it'll give you this interactive experience mainly because the data warehouses couldn't do this interactive experience. They were too slow for doing it, but it's kind of like a fish and egg problem. Sorry. Fish and egg.

[00:43:52] JM: That works. That works. That works.

[00:43:55] JN: Which came first? The fish or the egg?

[00:43:56] JM: Who gives all the reproductive rights to the chickens? Fish and eggs gape is a fine analogy.

[00:44:02] JN: Yeah. Now you know that I'm not from this country. It's not even a translation from anything I know about. Anyway. So it's kind of a chicken and problem. You needed a data warehouse that can do these interactive type queries really fast in order for UI that can be interactive to even be possible.

Druid was created and then they created a UI on top of it, and that was pivot. Then they open sourced these things and more and more people have found that they wanted to provide either their internal customers or their external customers with this analytical experience for whatever things they do.

What that meant was they would take Druid and then they would store all the data that they want on it and then they would build their own UIs on top of it. That's been kind of our biggest segment of customers so far, ones who have built their own applications either to serve internal customers or external customers.

What we're seeing happen is that these customers are so happy with kind of the latency that they're seeing from requests going into Druid for their applications that they say things like, "Well, why don't you hook up to our BI systems as well?"

Now, we are kind of in this zone in the middle that hasn't existed before where you have these BI applications, things like Tableau and Looker and so on that traditionally used to connect to things like Hive, or Presto, or Snowflake, or Terradata, now want to hook into Druid. This wasn't the original intention, but this is kind of what is happening.

What that means is that more and more Druid is actually starting to be looked at as a data warehouse. It's not a full data warehouse. It can't do all the types of queries that a regular data warehouse does. It's an operational data warehouse because it stores kind of operational type data. You don't want to put all the data in Druid at this point. The types of queries that it does aren't as powerful as the types of queries that a full data warehouse would be able to do.

Because they are connecting it to BI tools, they're looking at it as a data warehouse. That's kind of what's happening and now people are starting to look at us as a data warehouse and that is now the thing that we need to build next. The vision for us going in the future is that we will be a data warehouse that you would connect to your BI tools and hopefully will happen overtime is that these BI vendors will realize that they need to provide these kinds of interactive analytical capabilities for users so that they could do slice and dice analytics the same way as Pivot does.

We don't really want to – Pivot is great. We want to always use it as a way to showcase the best things that Druid can do, but our dream is like for everybody in the world to be able to just have BI tools and UIs that can actually do this and for us to just sell Imply to them and have them connect their BI tools to us.

[00:47:18] JM: Yeah, because I always looked at Imply and I was like, "This sounds hard." You're already building a database. Now you're building a UI. That's like a lot of stuff to do.

[00:47:26] JN: Yeah. It's solving the chicken and egg problem.

[00:47:28] JM: Fair enough. A creative way –

[00:47:31] JN: Fish and egg problem.

[00:47:32] JM: Yeah. There just would not have been a way to go to market immediately as a database company or a data warehouse company.

[00:47:37] JN: Exactly, because we can't do all the queries. Yes, we're faster, but like in the beginning, we didn't have JSON. Sorry. We didn't even have SQL. It was a JSON kind of type query that you have to write in order for Druid to execute. We recently added SQL and now we're adding joins in the next release and things like that. We're moving more and more towards standard data warehouse stuff.

[00:48:04] JM: When Fangjin first built the database, was he just querying it and like reading the queries and then like sort of just like writing reports? There was just no UI layer to it originally? Because if you couldn't plug it in to any BI tool, that seems like a very cluster experience.

[00:48:22] JN: Well, they built an application called Pivot at Metamarkets as well. Pivot actually also was part –

[00:48:29] JM: Okay. They built them concurrently.

[00:48:30] JN: Yes. They built them both together and that's what they sold to their customers.

[00:48:34] JM: Got it. The journey from operational analytics database to data warehouse, what's hard about that?

[00:48:45] JN: The journey from operational database?

[00:48:48] JM: I'm saying right now you're trying to expand from this set of use cases that you have to a broader set of use cases. Basically you're trying to be a full-fledged data warehouse.

[00:48:56] JN: Well, full-fledged is too strong of a word. I don't think we'll ever be able to do all the things that BigQuery or Snowflake does, and I don't think we want to. I think we want to get to a point where we're doing a pretty good number and like people use us as kind of the hot tier, if you will, like the hot layer for doing the more important – Not more important. The more latency-sensitive queries or for doing the interactive type queries.

[00:49:28] JM: I see. Okay. You really want to carve out a niche in that infrastructure monitoring example or the ad tracking example where you want to know these things on a very rapid basis, or you can imagine things like in financial services and you need stock trading applications where you really, really need low-latency as supposed to data warehousing where data warehousing, there's typically like a period of time where the data is not available in the data warehouse because you kind of basically have these micro-batches typically going into the data warehouse, right?

[00:50:00] JN: Yes. Not necessarily micro. I mean, like daily, it could be monthly, it could be weekly. Actually, here's an interesting use case. One of our customers has a big consulting shop and they are using Druid for insurance analytics. Insurance analytics, the data, accurate data will only exist for kind of like 6 months ago to a year ago. When they're looking at these kinds of data, they only want to look at data that's about a year ago. They only load that. They don't actually care about it being real-time in the sense that the data is fresh. What they do care about is the speed of the interactive queries that they execute.

That's a thing that you would say, "Well, these guys are so big. Why don't they use something like Snowflake or BigQuery or whatever?" Because those systems can't provide that same level of performance for those queries that they need.

[00:51:01] JM: I see. Just the way that these data warehouses are architected does not – There's a worse query performance.

[00:51:10] JN: Yes. It doesn't lend itself well to interactive kind of ad hoc analytics. The one other thing that I'll add is that a lot of these companies are also using a data warehouse. They always will – So far, they've had other queries that they can't execute in Druid, like things like for reporting that requires nested joins and things like that that Druid probably will never be able to do, but hopefully overtime Druid will be able to take SQL and kind of understand whether this is something Druid can execute or whether it's something that it can delegate to a data warehouse to kind of make this whole experience seamless for users, especially on the BI side.

[00:51:54] JM: Tell me something that has surprised you about developing a product in the modern data infrastructure environment.

[00:52:03] JN: Well, I've never really developed a product in the non-modern data infrastructure environment. It's really hard for me to kind of compare it. What has been surprising to me in this role in particular is just like how much stuff is still not yet thought about. It's really enlightening to me or really interesting or surprising to me that like this idea of an operational data warehouse still doesn't really exist pre se, or that the fact that engineers or marketing people or sales people or customer success people or whatever, the way that they interact with data through the UI is not really the BI method. It's not traditionally how BI tools work.

It's surprising to me that BI companies haven't yet caught up to that, but maybe they are. Maybe that is happening. I hope it is happening because that's really what I think will be a major driving factor in Druid adaption.

[00:53:04] JM: Last question. Any predictions for the next 5 years of data infrastructure?

[00:53:10] JN: Yes. One thing that I've been thinking about a lot is this idea of the reference kind of BI, open source BI architecture. I think at this point we kind of are starting to put together all the pieces and there are kind of open source driven companies that are doing these kinds of things now.

The idea would be that you would combine something like Presto for kind of data warehouse-y complex queries with something like Druid for the high-performance hot tiering of data and something like Superset or Pivot on top in order to actually give you kind of an end-to-end data warehousing, like open source data warehouse experience. That's a really interesting kind of framework to think about putting together for a company that's still too small to really want to go and buy large data warehouses or something like that and they want to just do things manually and put them together. Hopefully Imply overtime is going to make these kinds of architectures much easier to adapt.

[00:54:23] JM: Great answer. Thanks for coming on the show.

[00:54:25] JN: Thank you so much for having me.

[END OF INTERVIEW]

[00:54:36] JM: Apache Cassandra is an open source distributed database that was first created to meet the scalability and availability needs of Facebook, Amazon and Google. In previous episodes of Software Engineering Daily we have covered Cassandra’s architecture and its benefits, and we’re happy to have Datastax, the largest contributed to the Cassandra project since day one as a sponsor of Software Engineering Daily.

Datastax provides Datastax enterprise, a powerful distribution of Cassandra created by the team that has contributed the most to Cassandra. Datastax enterprise enables teams to develop faster, scale further, achieve operational simplicity, ensure enterprise security and run mixed workloads that work with the latest graph, search and analytics technology all running across hybrid and multi-cloud infrastructure.

More than 400 companies including Cisco, Capital One, and eBay run Datastax to modernize their database infrastructure, improve scalability and security, and deliver on projects such as customer analytics, IoT and e-commerce.

To learn more about Apache Cassandra and Datastax’s enterprise, go to datastax.com/sedaily. That's Datastax with an X, D-A-T-A-S-T-A-X, @datastax.com/sedaily.

Thank you to Datastax for being a sponsor of Software Engineering Daily. It’s a great honor to have Datastax as a sponsor, and you can go to datastax.com/sedaily to learn more.

[END]