EPISODE 920

[INTRODUCTION]

[00:00:00] JM: Every successful company generates a large amount of data. Large companies have multiple databases, multiple data formats and multiple applications that need to use the data. Every data engineer needs to move data between these different components of a system. Moving data between different parts of the system is often called ETL, an acronym for extract, transform, load.

Data engineers spend much of their time writing code for ETL, and this ETL code often moves data from a source such as a distributed file system into a data warehouse such as Amazon Redshift. ETL code is also used to take data out of platforms such as Google Analytics and put that data into a system where it can be joined with internal datasets.

George Fraser is the CEO of Fivetran, a company that builds data connectors to improve the data engineering process. George joins the show to discuss modern data warehousing and the business of building Fivetran. Data connectors may sound like a simple or trivial product, but it turns out to be a gigantic opportunity.

We are hiring a head of growth for Software Engineering Daily. If you like Software Engineering Daily and you consider yourself competent in sales, marketing, and strategy, send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:30] JM: Today's show is sponsored by Datadog, a modern full stack monitoring platform for cloud infrastructure, applications, logs and metrics all in one place. Use Datadog's rich, customizable dashboards to monitor, correlate, visualize and alert on data from disparate devices and cloud backends to have full visibility into performance.

Datadog breaks down the silos within an organization's teams and removes blind spots that could cause potential downtime. With more than 250 integrations, Datadog makes it easy to collaborate together and monitor every layer of their stack within a single platform.

Try monitoring with Datadog today using a free 14-day trial at softwareengineeringdaily.com/datadog and they'll send you a free t-shirt. That's softwareengineeringdaily.com/datadog. You sign up and you get a free t-shirt. Check it out at softwareengineering.com/datadog.

[INTERVIEW]

[00:02:43] JM: George Fraser, welcome to Software Engineering Daily.

[00:02:46] GF: Thank you. Glad to be with you.

[00:02:48] JM: We're going to be talking about data platform, data warehousing. Let's start with that term. What does that term data warehouse mean?

[00:02:56] GF: Data warehouse has a long history. Sometimes it means different things to different people. I think a data warehouse is just a database that you use to store all of the data about all of the things that have ever happened in your company. Some people feel strongly that a data warehouse has to have a star schema, that a data warehouse has to have a particular kind of technology under it. My opinion is it's just a database that has that role within your company.

[00:03:29] JM: Tell me about data warehousing pre and post-cloud.

[00:03:33] GF: So before the cloud, data warehouses were on-prem, like everything before the cloud. They were very expensive and they were mostly used by the largest companies because they were very expensive. A lot of data warehouses were using the same technology that you used for your production database. So you'd use the same kind of database to run your website or to run your retail inventory as you would use to operate your data warehouse. They might both be Oracle databases.

Right and around the same time as the cloud revolution happened, another revolution happened in data warehouses, which was the widespread adaption of column stores. Column stores are a fundamentally different way of building a relational database that is way, way faster for the kinds of queries that you tend to write against the data warehouse.

So, really, two things happened at the same time. Data warehouses moved to the cloud and column store databases, which already existed got widespread adoption, and they became far cheaper. So a lot more companies started using data warehouses that previously wouldn't have.

[00:04:42] JM: Why do people put data in a data warehouse today?

[00:04:45] GF: Modern businesses and many non-modern businesses use many different tools to operator their business. They will typically have some kind of finance tool, like QuickBooks, or NetSuite to keep track of their books. They'll often have payment systems like Stripe, or real-world payment systems, like Square. They will have tools that their developers use if they develop software at this company, like GitHub, like Jira. They'll have tools that their marketing team uses. They'll sometimes have dozens or hundreds of marketing tools.

If you want to know what is happening inside your business, the first step is usually to replicate all of that data into a single database. That you can write a query that references any piece of data about anything that happens in your business. That single database is your data warehouse.

[00:05:42] JM: Is the data in the data warehouse, is it sitting in-memory? Is it on disk? Do we know what it's doing?

[00:05:51] GF: So the data in a data warehouse will be sitting on disk until you query it and then it will be pipelined in the memory to run your query. A lot of data warehouses today, not only do they store the data on disk, but they actually store the data in a cloud-based blob store, like S3. So Snowflake works like this. BigQuery works like this. The data, when it's cold, actually resides in blob storage. Then when you start querying it, they load it into local disk. Then right at the moment when you run the query, they load it into memory.

[00:06:26] JM: If it’s on disk, why are we putting it in the data warehouse? What advantage is the data warehouse giving us? Why don’t we just leave it in our database until we’re ready to query it?

[00:06:35] GF: So the format that the data is stored in in the data warehouse is a very optimized format for the kind of queries that you run inside of a data warehouse. It’s a columnar format. That’s where the term column store that I referred to earlier comes from. The data is literally laid out in the transposed way compared to how you would lay it out in a normal database.

So instead of writing one row, like if I was writing a table of people. Instead of writing the person's identifier, and then their name, and then their email address, and then writing the next person's ID and their name and their email address. In a column store, I would actually write all of the IDs, and then all the names, and then all of the email addresses. For very complex and fascinating reasons, this layout lends itself to writing a very fast query processor for the kinds of queries that you run against the data warehouse.

[00:07:40] JM: Right. I think one example of that is if you layout your data in a columnar fashion, if you want to do an analysis of an entire column, like if you wanted to aggregate a total of an entire column, you would be able to iterate through every cell in that column without skipping over the other information in the rows. If you had a row-wise data format, you would have to be iterating over the information in these other columns, and that's going to be a much slower query.

[00:08:17] GF: That is exactly right. That is the first and most obvious advantage of the columnar format, that if you aren’t interested in a data in a column, you can simply not read it and you don't even have to skip over it as you read the data. It turns out that in order to fully take advantage of the column store concept, you actually need not just a different file format, but you need a different query planner. You need a black oriented query processor that actually if you look inside, deep inside the code of a column store data warehouse, the inner function that runs the query actually operates over vectors instead of operating over tuples.

So column stores are actually go way beyond just the file format. Every level of the database is implemented in different, and in many cases, opposite ways. But that's really where it all began with that insight that you just gave.

[00:09:14] JM: Why are all the popular data warehouses proprietary tools? Why don’t we have open source data warehouses that are popular?

[00:09:23] GF: There is one open source data warehouse that is fairly popular, which is Presto. It was created at Facebook and it is heavily used by some large technology companies. It’s still a bit of a beast to operate. You need to have a team to deploy and manage a Presto installation. But it is a legitimately fast column store data warehouse that is widely used among a certain type of company.

But you are correct, that amongst the sort of great masses of companies that aren't Netflix, that aren't Facebook, virtually all data warehouse column stores are proprietary. I think that just reflects the stage of evolution of the ecosystem that we’re at. Conventional row store OLTP databases that you would use to run a bank, to run a retailer, to run a website, those were all proprietary too in the 80s and in most of the 90s. Then MySQL and then PostgreS came along and eventually gained huge market share.

Today, I think someone doing a new bill wouldn’t consider a proprietary database. If you are like running a technology company, like a venture back technology company, you're going to use PostgreS or you’re going to use MySQL or something like that. Other data warehouse technology will undergo a similar evolution. I don't know. It may simply follow that same path where eventually the open source data columns or databases become popular.

It may follow a different path, because at the end of the day, the primary users of data warehouses aren’t engineers. They’re mostly analysts, and analysts don't care as much about whether something is open source. So it may not follow that same evolution. Time will tell.

[00:11:16] JM: Listeners who have not scaled a technology company may be confused, because when a product or project just got started, there's just a database. It’s got accounts. It's got some user transactions. But then when we talk about these companies at scale, we’re talking about data platforms. There’s a data warehouse. There’s a data lake. There’s a distributed queue. There're multiple databases.

People who have not seen the progression from just a database with users and accounts into this distributed platform of multiple data sources, multiple data processing units. They may not understand how that progression occurs. Why do companies end up having this massive diverse data platform?

[00:12:08] GF: Well, not all companies do have a massive diverse data platform. In fact, I would say most companies don't need a diverse data platform. A lot of companies, especially Bay Area technology companies, are guilty of overthinking this sector. Finding a way to use Kafka and use Spark and use a data lake and use everything you've read a blog post about, everything that has come on and maybe done an interview with you here, and many of those companies really just needed a fast column store data warehouse for their analysts and BI tools to go nuts with.

So it's not the case that every company should undergo that evolution. So I would say that we tend to see two things with fast-growing companies that are going through this journey as supposed to older, more established companies that went through it years ago and they have a different set of problems.

One pattern we see is sometimes people get a little carried away and they build may be more data infrastructure than they really needed. They may be used more different tools than they really needed to. That's definitely a thing that happens.

Another thing that we also see is people who do the opposite who are very pragmatic, who only solve the problems they have today. What they will typically do is they will initially use their production database as their production database and as their data warehouse. If they want to run a query against the data in there, they simply run against that database. If they want to know something about what their sales team is up to in Salesforce or what their marketing team is up to in AdWords, they just use the built-in dashboards of those tools. That is absolutely the right thing to do when you are really small.

Then usually the next stage is they create a read replica of their production database. So they decide, “Oh, I don’t want the analyst running queries against the production database.” It's usually pretty easy. You can just go into the AWS console or whatever cloud provider you’re using and make a read replica. So that’s sort of stage two. You still have data in multiple systems, but at least you're no longer running analytical queries against your production database.

Then either they find that the queries against that read replica gets slower and slower and slower and they simply can't get the answers fast enough, or they decide they really want to have all the data together. They want to be able to write a query that references their Salesforce data, or their production data, or their ad data, or you name it. At that point they decide to adapt a proper data warehouse and centralize all of their data. Hopefully at that point, they make a good choice as to what to use for their data warehouse.

Unfortunately, some engineers, because they’ve spent their whole career working on production systems rather than analytic systems, are not aware of the distinction between a row store and a column store. So they end up using a row store, like PostgreS or MySQL as their data warehouse simply because they're familiar with it, and that is a very bad choice. When you are setting up a true data warehouse that you're going to sync historical data from many systems into, you want a column store.

[SPONSOR MESSAGE]

[00:15:24] JM: Find a better job on Hired at hired.com/sedaily. Join Hired and let more than 10,000 companies ranging from high-growth startups to Fortune 500 companies apply to have you on their team.

Here's how it works. You just tell Hired what you like to do, what languages you prefer, the industry you want to work in, the salary that you want to make, and Hired will do their best to match you with a better job for your career. Plus you get salary and equity offers upfront before the interview.

Put yourself in the driver seat and join Hired at hired.com/sedaily.

Thank you to Hired for being a sponsor of Software Engineering Daily. Hired was the first ever sponsor of Software Engineering Daily. So we really appreciate the loyalty. If you're looking for a job, check out hired.com/sedaily.

[INTERVIEW CONTINUED]

[00:16:29] JM: One aspect of this modern data platform that you've alluded to a couple times is this fact that we're not just interfacing with "databases". There're these APIs, like Segment, or Salesforce and Stripe, and Google Analytics. Of course, as we are using these APIs, we're accumulating data. That data happens to be behind an API stored in a database at these companies. But the way that we view it is much like a database. Explain how these API-based systems have changed the world of the data platform.

[00:17:12] GF: Yeah. So the fundamental thing that has happened is that people simply use more tools to operate their business, and these tools all have APIs. That means that the data is a little bit scattered in many different places. All of the APIs, they work differently. They make different choices. So in many ways it makes the problem a lot harder and it creates a lot more incidental complexity around the problem.

At the end of the day, the best way that anyone has ever figured out to create a centralized system to access all of your data from all your APIs and all of your data stores is to make a copy to replicate all the data from all the places it lives into one data warehouse. The fundamental reason is speed. APIs are just too slow to run arbitrary queries while there's a person waiting for them if you run a query directly against the Salesforce API. If you're lucky and it's a simple query, it may return fast. A lot of the time, you're not going to be lucky, and it could take hours for that query to complete. That just doesn't work when you're trying to answer questions about what's happening in your business. The same is going to be true, for the GitHub API, for the NetSuite API, you name it.

So that means you end up in a situation where you need to copy all that data into a faster database in order to get access to it at acceptable speeds. That's the problem Fivetran is designed to solve in the simplest possible way. We have prebuilt connectors to a little over 150 data sources, including all the major databases, and we replicate the data one-to-one from all the places it lives into one database so that you can do with it whatever you want. We like to think we solve the first and hardest step of integrating all of your data, which is getting it all in one place.

[00:19:09] JM: Explain what your company Fivetran does in more detail.

[00:19:13] GF: Well, our engineering team over the last five years has built automated connectors to all of the data sources that our customers use. So these are things like Salesforce, like NetSuite, like MySQL or PostgreS, databases payment systems like Stripe. You name it. The way these connectors work is unlike conventional ETL tools where the user specifies all of the details of what API endpoints to call, how to normalize the data, how to load into the destination. Fivetran connectors are automated.

So when we talk to a data source like, say, Salesforce, the first thing we do is we call up the metadata API and we ask Salesforce what are all of the objects and fields that are available in the Salesforce account. Then we write the requests against the Salesforce API dynamically so that we're getting all of the data out of this particular customer's Salesforce source. Then we ingest that data, again, through an automated process into the destination data warehouse. This may seem obvious for software engineers. But believe it or not, this is a very radical approach in the land of ETL.

The idea of just a zero touch, set it and forget it ETL tool was very different than what was out there when we set out to do this for a variety of historical reasons. But we have found that it is a highly successful approach for our customers. They get a perfect replica of all of their data in their data warehouse. Then they can transform and model that data nondestructively by writing SQL queries. SQL turns out to be a great language for managing data, cleaning it up and reorganizing it into a format that's ready for analysts to query.

[00:21:04] JM: Let's go a little bit deeper on this term ETL. Define the term ETL.

[00:21:10] GF: Yes. That stands for extract, transform, load, which just refers to the nitty-gritty of the process of getting data from a source like, to continue the example, Salesforce into a database like, let's say, Snowflake. So when you call the Salesforce API, it returns a bunch of JSON responses. That's the extract phase. Then you have to convert that JSON into a format that the data warehouse can ingest. So in this example of Snowflake, that would be a CSV file. Then you load the data into Snowflake by executing a copy command and then usually a delete and insert step. So that is one example of the extract transform load process.

[00:21:55] JM: So these connectors that you build, essentially what they do is they alleviate the work of the engineer that would be having to write some kind of script that would query and API like Salesforce a bunch of times for the data. Put it into the data warehouse in the proper format. So these connectors give you a better interface for making that extraction.

[00:22:25] GF: That's exactly right. There's no magic to it. At the end of the day, we're just studying the Salesforce API and writing a bunch of code just as you would if you were building a connector yourself. The big difference is that when we write connectors, we write general purpose connectors that anyone can use. So, as I explained earlier, if we're talking to Salesforce, we use the metadata API to discover the available objects and then we ingest all of the data that's in your Salesforce account. That means that if you change your Salesforce account, if you add a new custom field, or a new custom table, we will discover that and add that your data warehouse automatically. Whereas, when people write scripts to do ETL internally, they typically just write a script that handles the particular configuration that existed at the time they wrote the script.

So in this example, they would write a script that queries all the fields and all the objects that existed in Salesforce when they wrote the script. That's a lot easier. It's a lot faster to write things that way. Fivetran kind of does it the hard way. But the advantage for us is because we're a product company, there are hundreds of customers who use this connector. So we just have to write one perfect Salesforce connector that anyone can use, and then we can sell it over and over.

So there's a kind of economy of scale of the way Fivetran works and the work our engineers do. We have to understand every little quirk of every API endpoint of all the data sources that we support, but we only have to figure out all those quirks once, and then every customer can benefit from them.

[00:24:01] JM: How do you build these different connectors at scale? Because I can imagine, this is ultimately an integrations problem. First of all, you have to figure out what are all the APIs and systems that we need to integrate with. Then you have to figure out what is going to be our repeatable process for assigning an engineer to watching this API and making sure that this API, we have it covered. What's the scalable procedure for getting all of the connectors for all of these different systems built?

[00:24:35] GF: Yeah. Repeatable processes is the key phrase there, and that is something we have discovered little by little over years. We initially built the connectors in a very simple way, which I think is always the right place to start with every new software engineering enterprise. Overtime, we built up a layer of core code that sits in between the connectors to the sources, and the data warehouse destinations. That core code does things like deduplicate data. It does things like type inference.

So we were able to find a lot of commonality between the connectors and put that into a core pipeline that they all share. Then we were able to do a lot to design an internal API that all of our developers write their connectors against that makes it easy to do the right thing, that makes it easy to write predictable connectors. There is no simple explanation that encompasses all these little lessons we've accumulated. None of them would've been obvious in advance. I think when people build connectors internally, they often build a lot of architecture that turns out to be not as useful as they think.

For example, a lot of people build their data pipelines around a streaming system, like Kafka. We do not, because it turns out that streaming systems do not solve any important problems in the kind of data pipelines that we build, because all the data sources are batch. All of the data destinations are batch. All of the data sources are durable. So if there is a failure, you can just go back to the source and get it again. It turns out, you really don't need a streaming system in the center of that, and files are a great way to buck for data.

The difficulty comes in other places. It comes in managing all of the incidental complexity. It comes in managing schema changes. That is a huge category of commonality between different connectors, is managing schema changes. But none of this stuff would've been obviously when we set out on this journey. All these lessons have been accumulated over years and incorporated it into this core code that all of our connectors share.

[00:26:46] JM: You've made a few illusions in this conversation to – Or it sounds like a belief that many systems are over-engineered. Perhaps we're using more complex tools than we need to be using. Can you go a little bit deeper on that? Do you think there's some kind of epidemic of mass deployment of distributed systems tools that we shouldn't be using? We should just be having simpler systems?

[00:27:16] GF: Well, it depends what you're doing. The fundamental thing that tends to happen in this area is that engineers like to work on problems on technologies that are interesting, which is not always the same as technologies that are valuable.

When you're building data pipelines between data sources, like Salesforce, data sources like NetSuite, the data is not that big if you can absolutely handle it on one node. There aren't the kinds of database technology problems that you get with like durability and recovery from failure. Because the source is durable, you can always just go back and get the same data again. Instead, all the difficulties are around this massive amount of incidental complexity that comes from all the data sources.

So you really need to optimize for that. You need to spend a lot of time thinking about how you structure your code, how you structure your tests. That kind of thing doesn't always make for a great blog post. So I think people tend not to focus on that. So I do think there is a tendency for people to overthink data engineering in particular, and it is something we've definitely steered away from over the lifetime of the company. I think we've a lot of success because of that.

[00:28:29] JM: Let's come back to engineering in a moment. But I'd like to focus for a moment on the business, which is very well defined, which is really great in the software world. If you can have an extremely well-defined product and you can get some customers for that extremely well-defined product. You're in a really good spot, because there're lots of adjacencies that you can expand into. This vertical of connectors, basically connectors between a data source and a data sync. I think that's a deceptively big business. It probably looks – For a naïve person, it sounds like, “How can that be a big enough business to build a large technology company around?”

But I can imagine, A, it's deeper than people expect. B, there are many adjacencies to expand to. Tell me about the longer-term vision for what Fivetran turns into.

[00:29:32] GF: It is an astonishingly wide problem. If you talk to 10 engineers, three of them will have at some point in their career built an internal data pipeline for their company. It's kind of astonishing when you think about how much engineering effort goes into writing connectors to NetSuite over and over and over, and they're all really the same.

So the connector business by itself is an astonishingly large sector. So we’re very optimistic about our future prospects for growth. We've grown a lot over the last couple of years. Each year, actually, of the company's history has been the fastest growing year ever despite the base getting higher and higher. We grew over three times last year.

In terms of where is this going in the future, we’re continuing to work on expanding the scope of our connectors. So building more connectors and on the quality of our connectors. There are a staggering number of corner cases, especially in databases. Just different ways that you can configure MySQL, and we are intent on tracking down every last one of them so that you can plug in any of our supported data sources and it just works for as long as you keep paying the bill. That is our mission in our software engineering.

So the most important dimension that we are expanding on is the quality of the connectors and the scope of the connectors going forward. There are a bunch of adjacent opportunities where we think we can add a lot of value. We think we can add a lot of value around managing metadata in each of these data sources because we build all the connectors internally and because we control them very tightly. In principle, we should be able to tell you things like this row in your data warehouse came from an API call that looks like this that ran seven months ago against your Salesforce data source.

That kind of information is really valuable especially in larger companies with big analytics teams where you're not totally sure who set up this connection in the first place. So we think there are some things we can do in metadata management that are really valuable, and we think there are some things we can do in helping people to write transformations of the data that we deliver that run after we deliver it. Because our connectors are non-configurable and standardized, the schemas we deliver for API data sources, like NetSuite, or GetHub, or Stripe, are always the same.

That means that if you look at Fivetran's customers, there’s a lot of people writing the same queries against the NetSuite schema over and over. We think we can do a lot to write templates and recipes for people to say, “Hey! Here's a bunch of SQL that transforms your normalized NetSuite schema into a dimensionalized schema that's ready for you to throw Tableau at it. These SQL queries, they’re not going to be perfect. They’re not going to be exactly what you want, but they’ll probably be 80% of the way there and they can save your analytics team a lot of time.” So we think there's a lot we can do to help there as well.

[SPONSOR MESSAGE]

[00:33:03] JM: As a programmer, you think in objects. With MongoDB, so does your database. MongoDB is the most popular document-based database built for modern application developers and the cloud area. Millions of developers use MongoDB to power the world's most innovative products and services, from cryptocurrency, to online gaming, IoT and more.

Try MongoDB today with Atlas, the global cloud database service that runs on AWS, Azure and Google Cloud. Configure, deploy, and connect to your database in just a few minutes. Check it out at mongodb.com/atlas. That's mongodb.com/atlas.

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

[INTERVIEW CONTINUED]

[00:33:59] JM: Well, this starts to look quite interesting, because if you can be the metadata source for the schemas of all these different systems, then perhaps it becomes – I imagine a world today with all these disconnected tools, and you have the notion of a customer in

Segment. You have the notion of a customer in Stripe. You have the notion of a customer in Salesforce. I don't know. I'm thinking about joining these.

You can join these in the data warehouse, and I guess that nice. But I suppose it would also be nice to just know at all times what is the metadata. If I were to do a join, what could I know about my customer? If I were to take all the rows from Segment and Salesforce and Stripe and merge them together, what can I actually know about a cost? Am I understanding the metadata vision correctly?

[00:34:59] GF: Yeah. There are sort of two things there. One of them is the joins. Just to get really concrete about this. The first thing is what can I join? If I want to join my Stripe users table with my Segment people table, what column can I join on there? That's a metadata management problem for sure, and that's something we hope to help you with in the next year.

The other problem is deduplication. Is this person in system A the same person as this other person in system B? That is more of a transformation problem. That's what a lot of people do using SQL with the data after we deliver it. That's where we think we can come in and give you a base of transformations that get you 80% of the way there.

[00:35:45] JM: Can you provide a little more detail on both of those things?

[00:35:48] GF: Well, some of these things are roadmap things. So I can only explain – Stay tuned. Watch the change log.

[00:35:54] JM: All right. I will watch it.

[00:35:55] GF: I don't want to undersell the connectors also. I know that metadata management and semantics of data are sometimes more exciting to talk about. But the reality is that the hardest problem in getting insight out of your data – Well, the hardest problem is getting people in your company to look at the data. But the second hardest problem is getting all the data in one place. The connectors by themselves are really, really valuable.

[00:36:25] JM: Okay. So if we're sticking to your knitting and talking about the connectors, let's talk about engineering at your company. Just tell me more about how you've structured the company at least from the engineering and engineering management point of view.

[00:36:41] GF: Yeah. So the engineering team spends probably about half of its time working on things that are shared across all the data sources, whether that's the UI, or that core pipeline code that I talked about before, or the code that talks to the destinations. So probably about half of the engineering effort goes towards that, and the other half of the engineering effort goes towards writing and maintaining connectors.

One of the cool things about engineering at Fivetran is that we're always writing new connectors. So there's always new stuff getting built. There's always new lessons getting learned and we're very rigorous about taking those lessons back to the existing connectors and re-factoring them to reflect our latest best practices.

The connectors definitely are on a spectrum of difficulties. So the most difficult connectors are databases, and the least difficult connectors are always APIs of relatively new startups. There's always very clean. They're beautiful. Unfortunately, they usually don't have very many users. But if you look at an API of a company that's a few years old that has some good engineers there. They're always great.

So, typically, what you'll see for an engineer making their career at Fivetran is they'll go through a progression or they start out working on the more straightforward connectors. Then there is a very a straightforward path to working on core code, working on destinations, working on more difficult connectors like databases. So it's really a great place to grow your career as an engineer, because we have this variety of work that we do.

[00:38:17] JM: There's one thing I find interesting about infrastructure companies like yourself. You can kind of see the future in some regards. So you can look at what APIs people are starting to use more and more and more and what kinds of joins or what kinds of transformations people want to run, I think. I mean, to what degree can you get strange insights into APIs that may be are flying under the radar? I mean, NetSuite, you already mentioned. Full disclosure, they're a sponsor of the show. This is not sponsored content, but it's interesting to know that so many people are doing transformations around NetSuite. That’s something I wouldn't have known, for example.

[00:39:03] GF: Well, NetSuite has really valuable data. They have all the data about your finances. So a lot of people want to build dashboards about that. So that's why you end up using NetSuite data. NetSuite has a very complex schema. It's complex for a reason, but it does mean that it takes a lot of SQL to get from the NetSuite normalized schema to a dashboard that everyone in your company can look at.

In terms of what insights do we get from what people are doing. We don't actually know very much about what kinds of queries people are writing, because that's all taking place in the data warehouse after we deliver the data. We can’t see that.

[00:39:38] JM: Right. So you only know the connectors that people want.

[00:39:41] GF: We do have this project to build prebuilt transformations and publicize them and build a community around them. I think as that starts to happen, we’ll just be talking to people more about what they're doing with the data, and maybe next year I’ll have more to say about that. But at the moment, we don't know a ton about what people are actually doing with the data.

In terms of data source popularity, it tends to be the obvious suspects who end up being the most popular. Things like Salesforce, like PostgreS are very widely used. It's very valuable data. So those end up being our most popular connectors.

[00:40:17] JM: Why did you first start working on this company?

[00:40:20] GF: Well, we originally started working on a data analysis tool. So the original concept was a vertically integrated data analysis tool, and through a long process of iteration, we discovered that data integration was actually the really unsolved part of this problem. That SQL works pretty well. That BI tools work pretty well.

But that data integration was just a nightmare. We didn't really understand why it was such a nightmare. If you look at the problem and you don't know the history of this space you go, “This seems obvious. We should just replicate everything into the data warehouse,” and it's not going to be easy. But if I centralize the effort, there’s an economy of scale there, and it's not impossible. I mean, you can figure out the NetSuite API. So that's how we attacked the problem and that first principles approach served us really well.

[00:41:12] JM: How did you get your first customers?

[00:41:14] GF: Our first customers were other Bay Area technology companies, larger Bay Area technology companies. Some of them we met through our angel investors. Some of them we met through partners. So we were doing partner deals very early.

If you think about it, Fivetran doesn't do anything by itself. It’s a data pipeline. It needs to pipe data somewhere. Even after the data arrives, you need some sort of BI tool to visualize it. So very early on, we partnered with Looker, a BI tool that recently was acquired by Google, and with Snowflake, a data warehouse company. Those two companies were like our big siblings in the early years. They not only referred us into many deals, but they really taught us how to be a company in many ways. It was like every time we had to figure out how to do something new in marketing and sales, you name it. We would call up our friends at Looker and Snowflake and be, “Hey, guys. How do you do this?”

[00:42:15] JM: So that’s an interesting couple partnerships the you mentioned there, because I think both of those as great point solutions, at least before Looker was acquired by Google Cloud. I guess now it's arguably a major cloud provider solution. But I think sometimes about this. The infrastructure companies that are not cloud providers versus the ones that are. The degree to which they're competing with one another, the degree to which they should be building alliances, the degree to which there’s just a competitive dynamic. Do you have any reflections on thriving in this age where there are lots of infrastructure companies that are trying to make it as independent players? Then there are these behemoths that kind of do everything, that there’s an air of fear around I think some of the infrastructure commentators, maybe not so much the actual companies, but some of the commentators. Some of the companies, I’m sure. But any reflections on a competitive dynamic?

[00:43:16] GF: For sure. I mean, it’s one of the questions that hangs over every infrastructure company. Is AWS eventually going to do this and take away all your market share? I think in our particular case, the really great advantage we have is the sheer amount of incidental complexity that comes from all these data sources. All of the corner cases that we have discovered and fixed over years and every one of these data sources, there is just no shortcut to discovering and fixing all of those problems no matter how much money and how many engineers you have. That makes Fivetran extremely hard to replicate. So that makes me sleep a little bit better at night, but I agree with you. That's a question that hangs over every infrastructure startup.

[00:44:03] JM: Isn't that true for almost every infrastructure company? It seems like every infrastructure company has so many little edges to sand, so many corner cases. I don't understand how – I guess the cloud providers have the edge of the sales channel on you. But in terms of actually building the best product, it just seems like the individual players have such an advantage.

[00:44:26] GF: Well, you’re right. Sometimes the majors try to replicate other company’s products and they fail for exactly that reason. I think that an example of something that the major cloud providers have done really well is hosting open source databases, so like Amazon RDS. Very widely used. Works great. We've used it for years. Love it, and they did a really great job because hosting MySQL or hosting PostgreS is kind of a self-contained technology problem.

You can have a team of engineers who work at it for years and eventually it's really solved. Whereas, other types of problems, like in our case building connectors to all these data sources involves all this incidental complexity that comes from all these sources. It's not self-contained. You're at the mercy of every API out there and whatever changes they’re going to make, and it never really ends. You have to manage it forever.

Our whole engineering process is optimized to deal with that kind of problem, and I think, I hope, that kind of problem is very hard to solve for one of these big, vertically integrated technology companies, because it's just not a self-contained problem.

[00:45:42] JM: You studied biology going as far as getting a PhD. How did you end up in data warehousing?

[00:45:50] GF: That is true. I did indeed do a PhD in neuroscience before getting into the technology industry. I did do some aspects of data analysis back in my days as a scientist. It's I funny, I just happened to have coffee with my PhD advisor this morning. He was in town, and it was great to see him and great to catch up with my past life as a neuroscientist. I had the ambition to start a company. I got the entrepreneur bug, and I wasn't brave enough to start a biotech company. So I started a software company with a longtime friend, Taylor Brown, is the other cofounder of Fivetran.

[00:46:27] JM: Where was the inflection points in your mindset where you went from doing PhD? I mean, PhD, you generally – I mean, I don't know what your trajectory was, but you're generally like going to either be a researcher full-time or you're going to be a professor. When did you catch the entrepreneurial bug?

[00:46:51] GF: Well, in the later years of graduate school, I started to realize that the funding system for science is really strange. It's, I am sorry to say, set up in a way that makes it really hard to do good work. I was really proud of the work that was going on in the lab that I was part of. That's why I had joined that lab. But looking out into my own future, the idea of becoming a professor and being at the mercy of this kind of crazy research funding process that in a weird way is set up to fund low-risk projects, which is like exactly the opposite of what you should be doing if you're trying to make a scientific breakthrough.

I just didn't know how I was going to succeed in that world, and that's what led me to go into the private sector. I didn't go straight to starting Fivetran from there. I went and worked at a biotech company called Emerald Therapeutics, now Emerald Lab, right in the early days of when it was getting started. Then it was after that that Taylor and I started Fivetran.

[00:47:56] JM: Do you believe in the idea of basic science research?

[00:48:00] GF: Absolutely. I with the funding agencies believed in basic science research. That's I would say. Yup.

[00:48:07] JM: How do you define basic science? Does the startup ecosystem successfully pursue things that resemble basic science?

[00:48:16] GF: No. You can't really do that kind of work in a commercial context. I don't believe. Because at a startup, you ultimately have to have a shot at getting to really big revenue numbers within a 10-year timescale, and that is unlikely even when we are trying to do it, and it's just really unlikely in the context of anything that could reasonably be called basic science. It's incredibly valuable. It's a classic public good as an economist would call it, but I don't think that it really gets done by startups.

[00:48:52] JM: All right, last question, how does the world of data engineering evolve in the next decade?

[00:48:57] GF: Personally, I think it's going to get a lot simpler. The underlying technology has gotten so much faster and so much cheaper that it is possible to do data engineering without this proliferation of tools that dominated in the past. So I think life is actually getting better and easier for data practitioners. They can use really fast, really cheap databases, like Snowflake, BigQuery and Redshift. They can use zero touch ETL tools, like Fivetran, and they can focus their efforts on the things that are really different for their business as supposed to building connectors, building query engines over and over and over. So I think we should be optimistic about the future of data engineering.

[00:49:44] JM: George Fraser, thank you coming on Software Engineering Daily. It's been a pleasure talking to you.

[00:49:47] GF: Thanks for having me.

[END OF INTERVIEW]

[00:49:58] JM: My favorite way to hire developers is Contract to Hire. In the Contract to Hire model, you pay a contractor to work with you on your company, and if you enjoy working with them, you hire them full-time. Compare this to the traditional hiring model of bringing an engineering candidate in for several rounds of interviews. Instead of spending lots of time crafting questions to be done in front of a whiteboard, Contract to Hire lets you actually work with the engineer on a real project and it compensates that engineer for their time.

It's always surprised me that Contract to Hire is not more widely used, but part of the reason for that is that there hasn't been a great platform that encourages Contract to Hire. Today, that has changed. Moonlight is a platform for hiring high quality software developers from all over the world to work on your project or your company. You can hire part-time or full-time developers, and if you enjoy working with them, you are free to bring them on to your team.

There are no lock-in effects. Moonlight does not charge you a finder's fee. They will work with you to make sure you and your developers that you hire are all happy. Go to moonlightwork.com/sedaily and get 50% off your company's first month of hiring access. That's moonlightwork.com/sedaily. If you're a developer who's looking for remote work and community, check out moonlight work.com to join for free.

I am a customer of Moonlight and I'm also a small investor. I love the platform and I'm actually amazed that something like it did not come out sooner. Check out moonlightwork.com/sedaily and get 50% off your first month of hiring access today. That's moonlightwork.com/sedaily.

Again full disclosure, I am a small investor in the company, but I'm also a customer, and I love what they're doing.

[END]