EPISODE 1005

[INTRODUCTION]

[00:00:00] JM: A data warehouse provides low-latency access to large volumes of data. A data warehouse is a crucial piece of infrastructure for a large company because it can be used to answer complex questions involving a large number of data points. But a data warehouse usually cannot hold all of a company's data at any given time. Users need to move a subset of the data into the data warehouse by reading large files from a data lake on disk and putting that data into the data warehouse.

The process of moving data from one place into another is broken down into three sequential steps, often called ETL, or extract, transform, load. It also might be called ELT, extract, load transform. In ETL, the data is extracted from a source such as a data lake transformed into a schema that is customized for the data warehouse application, and then load it into the data warehouse. In ELT, the last two steps are reversed, because modern systems can often leave the necessary schema transformation or other kinds of transformations until after the data has been loaded into the data warehouse. In today's show, we talk about the ETL versus ELT acronyms.

Matthew Scullion is the CEO of Matillion, a company that specializes in building tools for data transformations. Matthew joins the show to talk about the problem of data transformation and how that problem has evolved over the 9 years since he started Matillion.

If you enjoy the show, you can find all of our past episodes about data infrastructure by going to softwaredaily.com. You can also download our mobile apps in the iOS or Android app stores. You can search for technologies or topics and find all of our podcast episodes. You can listen to them on our website or in our apps. If there's a subject that you want to hear covered, you can leave a comment on an episode or you can send us a tweet @Software_Daily and we'll take a look, and this is how we find some of our new subjects to cover.

[SPONSOR MESSAGE]

[00:02:14] JM: Today's show is brought to you by Heroku, which has been my most frequently used cloud provider since I started as a software engineer. Heroku allows me to build and deploy my apps quickly without friction. Heroku's focus has always been on the developer experience, and working with data on the platform brings that same great experience. Heroku knows that you need fast access to data and insights so you can bring the most compelling and relevant apps to market.

Heroku's fully managed Postgres, Redis and Kafka data services help you get started faster and be more productive. Whether you're working with Postgres, or Apache Kafka, or Redis, and that means you can focus on building data-driven apps, not data infrastructure.

Visit softwareengineeringdaily.com/herokudata to learn about Heroku's managed data services. We build our own site, softwaredaily.com on Heroku, and as we scale, we will eventually need access to data services. I'm looking forward to taking advantage of Heroku's managed data services because I'm confident that they will be as easy to use as Heroku's core deployment and application management systems.

Visit softwareengineeringdaily.com/herokudata to find out more, and thanks to Heroku for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:03:46] JM: Matthew Scullion, welcome to Software Engineering Daily.

[00:03:49] MS: Thank you. Pleased to be here.

[00:03:50] JM: You started Matillion in 2011, and Matillion is based around, at least today, helping people with their data platform. For a large company, what was a typical data platform back in 2011?

[00:04:06] MS: Yeah, great question. I guess it was probably just starting to transition. Certainly, prior to that sort of time, for a large company, a data platform would often times be a data warehouse, appliance market leader, probably Teradata, you've also got Netezza, and Vertica

and other similar data warehouse appliances. These were columnar analytical databases optimized for large volumes of data and resolving queries really quickly. People could crunch their data and get insight. They turned up on a back of a semi-truck, they cost several millions dollar a piece. Just like an F1 team, they have a group of highly qualified engineers looking after them.

[00:04:52] JM: What did the term data warehouse mean back in 2011?

[00:04:56] MS: Yeah. It's a good question and something I find myself talking about a fair bit. I think data warehouse means two different things actually. What people mean both back then and today when they describe data warehouse is the engine on which you put the data. In 2011 and certainly prior to that, it would have meant Teradata or Netezza appliance. The physical bit of kit back then, the engine on it, stored and organized data optimized and resolve queries over that data.

The other meaning of data warehouse both back then and this time today as well is what you do with that engine. To me, as a data professional and a founder/CEO of a company that makes tools for creating these warehouses, we're not creating tools that create a data warehouse appliance or engine or a subservice. We're creating a tool to allow to customers to put data on to one of those and then crucially get it organized ready for analytics to join data together, embellish it with business metrics, sort our quality and granularity to turn siloed source data that doesn't tell the story into de-normalized organized data that adds value to the business and unlocks insight.

Yeah, two different definitions of data warehouse; the engine, the physical bit of kit that you do that on, but also the finished database with data on it and organized that delivers business value.

[00:06:38] JM: What kinds of questions was the data warehouse being used for back in 2011? Who is the end user and what kind of questions were they asking?

[00:06:46] MS: Yeah. I think the key difference between 2011 and today to that question would be not just who they were, but how many of them there were. What I mean by that is data

warehouse is used to be expensive and a heavy lift. If you're on-RAM cost is many millions of dollars. To get hold of a data warehouse, you need to run through a traditional solution sales cycle with a vendor over many months. Then to physically get hold of this thing, it needs to be delivered to your on-prem data center, configured, set up and then a project to execute it to get it working. You just don't do that super frequently, right? It has to be a really important use case that's well-defined and delivered over a long project.

I think where data warehouse is, and therefore I guess to some extent, data analytics we used in large business prior to the timeframe you're talking about, through the 2000s or through the 90s, were on the most important analytical questions in a given business. How much money they're making and where and who from? Broken down by products and customers and sexes, that sort of thing. It's going to depend on the business, but it's going to be those most important questions that justify the large sum costs in terms of dollars and time in getting to insights. That's the key difference from back then to today. Because of large sum costs in terms of dollars and time, you really have to prioritize what you wanted to know about.

What probably hasn't changed quite so much is the areas of the business that that could possibly apply to. Today we tend to talk about data innovation usually sitting in one of maybe three main areas. It's around understanding customer behavior better, whether that's how you market to them, how your company is monetizing customers or how you deal with them. It could be around products. Understanding how products are working or improving products. Finally, it could be streamlining business processes. But each of those cases, back in that timeframe you're asking about, it would have just been the most important, most obvious places that you need an analytical insight.

[00:09:07] JM: If we go back in time even further, I think 2005, 2006 was around when the MapReduce paper came out and I think two, three years after that, companies started being built to productize Hadoop, the open source MapReduce project, Cloudera, and MapR, and Hortonworks. You had companies adapting Hadoop. You had Hive. How did the open source MapReduce project, Hadoop, how did that change data warehousing together with Hive?

[00:09:43] MS: You're stretching me a little bit, because I wasn't actually active in this industry at that time.

[00:09:49] JM: Well, I guess in 2011, there were people using Hadoop and Hive for ETL, right?

[00:09:54] MS: Yeah. That's exactly right. Those products certainly came along mid-2000s and by 2011 when I started to get into this, I guess, really we saw at that time two camps or two thesis around how to deal with data or volume. Camp one, as you rightly say, was that MapReduce camp, and Hadoop, and Pig and on those vendors that you mentioned that productize those products.

I think the thing that was appealing around those technologies to end users was scale, ability to deal with large data volumes, particularly web scale data volumes. Those products professed and in many cases did successfully deal with that amount of data, which previously would have been, at the very minimum, very expensive or possibly prevents to be difficult.

The other thing which I always thought was a misnomer, and I think maybe that comes to be proven out, the other thing was people liked the perceived unit economics, right? The technology, the open source, the hardware as commodity, that sounds cheap.

The other camp was data warehouses and then later cloud data warehouses, which were also designed and are capable of delivering analytical insight over large volumes of data. But on the one hand, pay for that. Certainly back in 2011, you'd been buying a client space data warehouse, which would have been expensive as we've already talked about. There might be a third factor in there as well, and I risk some social media backlash by saying this maybe. I hope not too much. But of course, with new technology, with software people and engineers, have always just interested in it. It's interesting and tempting to get into because it's a new area of technology.

I think the way that played out is those MapReduce technologies did they do what they set on the ten in so far as they could deal with huge volumes of data, web scale volumes of data and the unit economics on the software cost and the hardware cost were good. They were offset however by the human cost of doing that, because they were complex. So you needed highly skilled people to look after this stuff and to build it. That taught to total cost of ownership and it

also taught to ability to innovate at pace, at the requisite pace to keep up with what the rest of the industry was doing with simpler technology I suppose.

I think if you fast-forward to today, you can see how that played out. Where those MapReduce technology use cases resonated was in the really largest scale, web scale use cases were the investment in people was worth it. If you're a Netflix perhaps or some of the web scale company, first of all, you can attract to retain and afford the talents, and B, you need that genuinely top hand capability of dealing with huge volumes of data. For everybody else, the slow time to value and highs and cost in terms of people I think made out an economic.

[00:12:59] JM: Great answer. Throughout this time, what has the role of Oracle been in data warehousing historically?

[00:13:07] MS: It's probably easier for me to answer that in today's context than it was for me to look back to around 2011. I'll tell you the honest reason for that. I wasn't thinking a whole lot about Oracle back in 2011. I have reflected on it quite a bit since and some of the market dynamics that Oracle have left us with.

Today, and I think we're probably going to get on and talk about this in a few minutes perhaps, and so I don't want to get too far ahead of your questions. But today people think a lot about consuming cloud services in general, give them competitive advantage. In the data space, they think about consuming data lake and particularly cloud data warehouse technology in order to allow them to innovate with data at pace, and as I mentioned before, gain competitive advantage of doing that.

What they also think about though is not getting locked in to a platform. If you think about data warehousing pre-2011 and through the 90s and 2000s, a really common pairing of technology would perhaps be something like Informatica PowerCenter is the data integration layer and Teradata as the data warehouse appliance. Two fantastic products, fantastic technologies from outstanding companies, because to get into the club where you're playing with that technology would have cost you millions of dollars.

Your likelihood of wanting to switch real quick off one of those was pretty low. It's not like you're going to spend a few million bucks on a Teradata appliance, a little more money also in Informatica. Negotiate all that with your CFO. Go through procurement. Get your project alive. Then a month later, Netezza released some new feature that just made it slightly better in one particular way than Teradata. You go back to your C Suite and say, "Hey, I know we've just spent all that money. Let's change because this one is now marginally faster and marginally shinier." You aren't going to do that because you have so many capital and efforts sunk in to whatever choice you made off the bat.

Today in the cloud world, that's much less the case. The switching cost between cloud service A and cloud service B is incredibly low. I think customers think a lot more carefully about making sure that they don't accidentally erase that low switching cost and tie themselves into a given technology, because that would stop them being able to take advantage of this ability to switch between the best technology at a given point in time.

I think that conditioning came from everybody's Oracle experience, because we all built Oracle databases, we built into them, so it was difficult to move off them. Then, respectfully, to what is a very successful company that many of us still rely on is called technology, of course. Many people came to live to regret that decision about baking themselves into Oracle so much later on in their life with them.

For me, whenever anyone asked me about Oracle, I think that's the biggest effect. It focuses people's mind on not being locked in to a particular technology, particularly now there aren't large sum costs because of the cloud.

[SPONSOR MESSAGE]

[00:16:24] JM: Today's show is sponsored by StrongDM. StrongDM is a system for managing and monitoring access to servers, databases and Kubernetes clusters. You already treat infrastructure as code. StrongDM lets you do the same with access. With StrongDM, easily extend your identity provider to manage infrastructure access.

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

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

Thank you to StrongDM for sponsoring Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:17:46] JM: One of the reasons I wanted to start with these historical questions is I wanted to bring us up to 2012 when Amazon Red Shift was released, and Amazon Red Shift is – Maybe you can tell me differently, but probably the most popular data warehouse today. It's a data warehouse is actually named after – I mean, the idea of Red Shift, this is something I learned in the research for this episode is –

[00:18:14] MS: Doppler effect.

[00:18:15] JM: Actually, what I read on Wikipedia, maybe this is wrong, is the idea that Oracle's log is red and we're shifting away from red.

[00:18:27] MS: Well, there you go. I mean –

[00:18:30] JM: I mean, the thing is it sounds like a very – I read that and I was like, "Is that true?" It might not be true. I might be quoting. This is might be the first example of actually reading on Wikipedia that is very wrong, because it sounds very non-Amazon to name a product after basically in opposition to some other contemporary product.

[00:18:51] MS: Well, later, while after launching Red Shift, of course they got pretty poky about pulling out Oracle. Hey, maybe it was just a brilliant looking to the future of what the corporate position of AWS against Oracle was going to be. I must confess, either this is very amusing because you just embarrassed the CEO of the perhaps world's number one data integration tool built for Red Shift who's been personally hanging out with the Red Shift team as well.

The original etymology of the name that I understood was it was simply from physics and the Doppler effect, and therefore a metaphor for speed. I think the color of light changes to red accelerating away from you.

The nice thing about all these histories of things is of course mostly you don't know and therefore you can happily be revisionist about it. Let's see if after this podcast the world picks up one or another of our explanations.

[00:19:43] JM: Well, getting away from etymology, how did Red Shift affect your business when Amazon released a cloud data warehouse in 2012?

[00:19:53] MS: Well, that's really at the heart of the Matillion kind of founder story as this sometimes gets called. Matillion is a business that's been around since 2011. We launched the product range for which we now know and which we have many customers across the world using every day to innovate with data, Matillion ETL. We launched that product range in October of 2015, at the AWS Reinvent Conference, which sometimes jives the question what were you doing between 2011 and 2015, right?

[00:20:25] JM: Yeah.

[00:20:25] MS: Matillion was incorporated to do something different than build ETL tools. I pride to founding the company. I've been working for some large SI's in Europe. One of the things that we did within the business I was working in before setting up Matillion was implement BI systems for people. Analytics as we'd call it today. I had a team of guys that went out implementing – It was actually IBM Cognos that we used to use back in those days, and we built a data warehouse, implement Cognos and delivery business value that way. Customers loved it. Got value out of it, but it was expensive.

Another part of the team I ran in my private company, I had a team of really great guys run by a gentleman called Ed Thompson who was employee number one at Matillion, my cofounder and our CTO. He ran an integration team that always got to play with the all the latest, greatest technology, including a relevant history, AWS.

We thought, “Hey, there’s a gap in the market here for people that need more sophisticated analytics,” But they either don’t have the time or perhaps the IT expertise to deliver it themselves swiftly. What we’re going to do is we’re going to stand up a company, we’ll call it Matillion and we’re going to deliver turnkey, fully managed BI as a service. We’re going to sell that in a SaaS business model. Of course, we’re going to do it in the cloud. That’s what we did.

When we stood that company up in January of 2011, we started building end-to-end BI solutions for our customers. It have that data warehouse layer, an ETL layer, a BI frontend and we clothed it with some services and take a customer, threw in a small number of weeks to having great quality business insights driven by analytics.

In 2011, as you rightly say, Red Shift didn’t exist, but AWS did, and we’ve built this business on AWS. We still use a cloud-based data warehouse of sorts. We used an open source columnar analytical database that we kind of configured and hosted our self. It was a product called Infobright, but we did have a bit of a technical policy back then. I say bits of that because we’re a small company. This wasn’t a 300-page document. It was just an approach. But wherever we could, we avoid reinventing the wheel by consuming AWS services.

As such, when AWS launched Red Shift in late 2012, we immediately thought, “Well, gosh! We’ve been using an open source columnar analytical database. It’s a fully managed cloud service analytical database from AWS. Let’s use that.”

I think we often forget when telling our story is we then chose not to, because when Red Shift was launched, it was only launched on traditional disks and on some machine image sizes that were not super-duper performant actually and actually wasn’t fast enough for us. We didn’t use it straight away. But the following summer, summer of 2013, they launched some new instance types that were independent by SSD disk. It was much, much faster.

So at that point, we migrated all the customers that we built over the past few years on AWS, yes, but not on Red Shift, rather on this info broad database. We migrated all those guys across to Red Shift and every incremental new project that we built out for our managed BI customers at that point, we did that on Amazon Red Shift.

At that point, how did it change our business, to answer your original question, well it just made it a bit easier and a bit faster, I suppose, at that point. We've been managing this open source columnar database ourselves and now we have to do less of that. It was easier to scale up and down. Like any good consumer of cloud services, we were able to somewhat get out of the business of managing infrastructure.

Then as innovations came along in Red Shift, we were able to pass those on to our customers. Just one thing coming right from the debts of my memory, I remember being excited about launching back then was the ability to encrypt the data warehouse, which Red Shift could do really easily. That was valuable to these managed BI customers that we have. So we'd sell that on to them as an option.

The real change came from something perhaps less obvious, but really what built Matillion as it is today and has driven all our growth over the past few years. The real change came a couple of years later or a year or so later. We had this growing managed BI as a service business. I will say in retrospect if anyone's thinking you're setting one of these up, it was quite a tough gig that business. The product was great. Customers who bought it loved it, but the idea of a customer to buy a fully managed BI service is like a medium sized customers. Anyone bigger wants to do it themselves. Anyone smaller, those who need it can't afford it.

It turned out that they just itself persuading the CFO of a like 30-year-old privately held fasteners business to buy a cloud project, right? Anyway, we built our business around it. We have all these customers, and we were building and maintaining scores of cloud data warehouses for our customers. Actually, the byproducts or the fact we're finding it pretty tough to sell it, we wanted to get even more efficient building data warehouses. We've done pretty good if I'm honest with you. We kind of tried to cut all the waste out of the process. We template it all. We handle automations to make it slicker.

But we got to a point where there was no more fact to trim out to the process because almost everything we were doing was ETL. It was mapping the source data from the customer's production systems, which were typically different every time into the data warehouse, into the shape and size of fact and dimension tables required to support the customer's analysis downstream. That mapping process, that adding of revenue and cost to make margin, that filtering out of the data that was wrong goal as test data, all that kind of gritty stuff that you do as you're taking siloed source data from production systems and turning into beautiful organized analytic ready data, we just couldn't get faster at that.

One of the reasons was we were using a pre-cloud ETL tool. I probably prefer not to say which one it was, because this is a public forum and I wouldn't want to be disrespectful to that company, not least because this was a great product. I've actually used it in my private job. This is now a public company and this product had everything you needed to build data warehouses, join data together, clean it up, embellish it with metrics and participate in a sophisticated IT setting, but it was just all built for the pre-cloud world. So it was slow and it was clunky and it was reducing our ability to innovate with data in the cloud to put finer points on to it.

The biggest way Red Shift changed our world back then was that we were really delighted with Red Shift, but we became frustrated about our ability to build data warehouses on it quickly. What that led to is in early 2014, we do what every good techy does when he's got a problem, we got into Google and we tried to find a product that looks and smelt just like what we've been using already, but it was built for the cloud.

As it happened, no such product existed. Funny enough, no such product really exist today, I would argue, apart from Matillion ETL. That's maybe a little plug, so I'll back away from that. But no such products existed. Of course, we've been using Red Shift very aggressively since almost immediately after it's launched.

I guess at that stage, it was a particular point in time, most customers were either using Red Shift once or twice because it was for themselves. They were an end user and they were building their own data warehouse, or maybe they're an SI or consulting company that's building

data warehouses for their customers, but they're doing that out on site with their customers. So they're doing it scattered around the country or the world.

We were building lots and lots of data warehouses, I'm managing them, and we were doing it all from one, I'll admit now, quite unglamorous room in a little village called Knutsford about 10 miles south of Manchester, UK, and we were building up this concentrated knowledge of everything you'd really have in an ideal world. If you have a tool purpose-built for Red Shift or purpose-built for cloud native warehouse.

In 2014 when we got into Google and found there wasn't something like what we've been using but built for the cloud, we elected to build it ourselves. We had a very small engineering team. We're a really small company, but we spent a year building what we hoped would be all the good stuff from the product gen, but re-architected for the way you do it in the post-cloud world.

Around comes January 2015, we've got a very early version of this product ready and we try out on a live customer project, and it worked really well. It executed jobs. The job of sucking in data from a source system, transforming it into the shape and size that you need to support analytics and getting it into the data warehouse about a hundred times faster than the technology we've been using previously. That's really important at runtime, but also really important of development time, because as you build out ETL jobs, what a typical data professional will do is they'll say, "Pull in data from here. Join it to here. Filter here. Okay. I'm just going to check it works," and they'll press run. If that job takes 20 minutes to run, they've got to wait 20 minutes to see if their first three steps of development worked. If it does, they'll move on to the next one and a couple more components, rinse and repeat.

There's a 20-minute delay or an hour delay every time you try out what you have written that really slows you down. These 20-minute jobs were taking 12, 13 seconds in this new tool that we created. That's flushed out to much more efficiency overall in the job of building a data warehouse. It was taking less than half the time to build a data warehouse. We do really optimize the process, and remember the numbers actually, it used to take us about 35 days to stand up a customer. Of the 35, 5 days were kind of project-y stuff and training courses. 30 days was ETL. The first time we used this very early, very wet paint version of Matillion ETL, it took us about 14 days. Less than half the time.

Then finally we have this very authentic group of data professionals that all use the Informaticas and IBM DataStages and SSIS's and talons. They'd all use those products in previous job. It was these guys driving this new tool. They were a very cynical crowd, as many of us engineers are, and yet they really liked it.

With that in mind, I'm driven by our own pain, we've actually created a product purpose-built for Amazon Red Shift, and because we architected it to work well in the cloud, but because we also learned from what you need in those prior gen tools, we ended up with something very compelling.

Just one final thing, and I know I've been talking a long time in this question, but also AWS did us a massive favor about that time. They happened to announce, and they're not a company that shares its statistics very often, but they happen to announce that, A, they were this extremely fast-growing once in a generation tectonic shift in the way that B2B IT was being done, and they were making a lot of noise about that time.

They also happen to announce that of all the different services they had, EC2, and S3, and EBS, and all these other services, Red Shift was the fastest growing one. We just happened to have built a tool that was 100 times faster, twice as productive. Our guys really like using it and it happens to be for the fastest growing service, one of the fastest growing technology companies ever. That's how we changed our business.

[SPONSOR MESSAGE]

[00:32:59] 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]

[00:35:15] JM: It's such a great story, the fact that you started in 2011. Searched around for a product to build, found this full stack data warehouse that you build in the cloud based on open source technology, but realized that the market was not there to build a substantial business. Then AWS releases their own cloud data warehouse. You looked for a way to draft off of that and you end up building this tool that slots in perfectly into the changing trends of enterprise technology. I think engineering aside, it's quite a good story of entrepreneurship and finding a product that works for you, that allows you to build a big business. But since we've spent a lot of time talking about the basic story, I want to get into ETL today.

The ETL process, as I understand, basically you've got data in one place. You want to get it into another place and you want to make it queriable in that other place that you're moving into. If we talk about the ETL process today and particularly what it means in terms of your business at

Matillion, explain what that means. Explain what the average customer is doing when we're talking about ETL.

[00:36:43] MS: ETL is one of those rare IT acronyms that is pretty useful in describing what it does, but it's definitely worth dwelling on, because I think often times, particularly today with a lot more people innovating with data and many of them doing it for the first time fueled by the cloud. The emphasis is sometimes lost. You described it, ETL just then is I get my data out the source system. I load it into wherever I want to do the analysis, such as Amazon Red Shift, Snowflake, BigQuery, etc., and then I get the data organized and ready to support my queries. That's basically true.

What's interesting is where the value is and what uses the most time of the data professional, I think. First of all, let's start on the kind of raw bits and bytes. Let's say you're trying to do some analysis that helps you understand what's going on with your customers. Let's say you've got customer data in SAP or NetSuite. Well, we just started with one of those systems.

SAP if I remember right, and I'm probably getting this massively wrong, but SAP under the hood has got I think 80,000 tables that support it. Even in that one system, the story that you want to analyze, "How are my customers behaving? How much money am I making from selling blue jeans in Texas to people 30 through 50-years-old?"

The information that answers that question might or be an SAP, but it's not in one place. It's spread across lots of different places. In ETL, the first thing we do is we get the raw data, the raw siloed data. We go and get the customer master file because that tells us who the customer is. We go and get the descriptions, because that maps a numeric code to a US state. So now we can start to say they're in Texas.

We go and get the product master, the order headers, the order lines, all these individual components that tell a little bit of the story and we bring all those over to our ETL tool, or in fact to our data warehouse. What all you've done there is moved it. You've just got the data from being buried in the production system in a place where you can start to play with it. In the old world, that place where you started to play with it was normally on the data integration system itself. It would be in fly as it went through Informatica or Talon.

The way we do it today at Matillion, and you can still do it the old way, we do it ELT actually. We extract and then load on to the data warehouse and then do the bit that I'm going to come on to, which is transformation.

Recap. We've taken the data from the siloed source system. We now got it to the place where we can do some stuff with it. To answer our question about how much money am I making from selling blue jeans in Texas to people in a particular age range, I now need to do a couple of things. I need to join those different bits of data together. Basic level, these are just SQL joins, right? I'm joining the order headers to the order lines, the order headers to the customer master, the customer master to the customer description, but typically in business analysis, those joins get pretty complicated pretty quick because you've got lots of different bits of data.

The second thing you're basically doing is embellishing on metrics. It wants the question how much money am I making? Well, that's actually a margin question, a profit question. To answer it, I need to get revenue from maybe these sales invoice and I need to get cost from somewhere else in this system. Pull all that together and then just do a tiny bit of math to say, "Take a cost away from the revenue, and that gives me the margin. That gives me how much money I've made."

I want to store that answer in the data warehouse so that when a hundred or a thousand or ten thousand users all go and look up the answer to how much money did we make selling blue jeans in Texas, they all get the same answer. If I give them the raw materials, they'll add that up themselves and they'll all do it different ways and get different answers. So I want to put the answer in the data warehouse and I'd do that at the ETL layer.

Now, inevitably, you also want to sort out quality. There'll be different bits of data from different parts of the company and that, for instance, SAP system. There'll be some old data that you know is a bit rubbish. There'll be some descriptions that have changed overtime that you need to sort out. There's quite a lot of cleaning up, joining together, embellishment and granularity changes of the data that you need to go from what's buried in SAP to what's going to support your beautiful Tableau or Looker dashboards downstream. That example was from one system.

In reality, we today, to get that picture of how our customers are behaving, load data from multiple systems. We're probably going to load it from the core production systems, like NetSuite or SAP, but also maybe from commerce platforms, from a marketing automation platform, Marketo, HubSpot, from our digital advertising platforms maybe, Google AdWords, Google analytics, maybe from ZenDesk to see how customers success and customer support interactions, change how many pairs of jeans people buy, all that sort of stuff.

What starts it off is a little bit complicated pulling data from several places of SAP, joining it together and embellishing it and sorting out the quality, that becomes exponentially more complicated when you're doing it across multiple systems.

The key takeaway here is that moving the data is actually pretty simple. Bear in mind, I make quite a lot of money from selling people software that allows them to move data. Get data from SAP to Snowflake. Get data from Marketo to Red Shift. My software does that.

I'm telling you that the data professional, that's the easy bit. Actually, trade secret, from the ISV's point of view, it's not that difficult either. Useful, essential, but not that difficult for either me or the end user. But where the data professionals spends their life is once they got it there, once they got into the data warehouse and in ELT architecture, once they've got into the data integration platform in a pre-cloud ETL architecture, where they spend their life is joining that data together, embellishing it with metrics, sorting out quality, obligation, granularity and doing all of that at scale.

A different way of saying that is that they're innovating with data, that adding business by value and getting insight, and they do it in the data warehouse so that the end products, the end insight that joins business value is trusted and then can be consumed by tens or dozens of systems, thousands or tens of thousands of users. Does that all make sense?

[00:43:39] JM: Yes, and I'd like to revisit the difference between ETL and ELT briefly. Can you just go a little bit deeper into why those acronyms have gone from ETL to ELT being the more popular order of operations?

[00:44:01] MS: Yeah, definitely. To the end user, the process is pretty similar, and I think if we both do our jobs right as either an ETL vendor or a vendor whose technologies is an ELT architecture like mine, the end user experience to the data professional shouldn't be that different, really. Ultimately, they don't want to care about the underlying architecture, they just want to get on innovating the data and creating business value.

But the acronym switch around, if you like, is important in the context of the cloud and cloud data warehouses. Back in the day, as we discussed earlier in the podcast, on-prem data warehouse, very expensive, high-sum costs. If you buy something for several millions bucks and it takes ages to arrive, couple of things happen. One is you use it carefully to make sure you get value out of it. Secondly, you stand a small army of ops people around it saying, "Don't source this. It was expensive."

Therefore when you've got a team of data professionals kind of innovating the data and effectively doing development, they're doing development with data, it doesn't necessarily make sense to do that development directly on the data warehouse. When they want to join two tables together, they could do that in the data warehouse, but the data warehouse was expensive and is busy serving the needs of the business user consumer their reports and analytics.

For that reason, previously, ETL architecture made a lot of sense. You did the transformation piece. You joined the data together, embellished the metrics, sourced out the quality and they granularity, you did all that on the ETL system's infrastructure. You extracted from source. You translated the data into some kind of in-flight mediated state to kind of Rosetta stones states of data. You applied your transformation logic there and then you translated it again into the destination format of data for your destination data warehouse. That means that all your transformation compute is being done on your Informatica cluster, your time cluster, your data stage cluster. You need to do more transformation. You buy-in more Informatica or Talon, but you don't buy more of your very expensive data warehouse.

There is another benefit of ETL, and I don't do ETL. So this is me sort of trying to be authentic pointing out the benefits of the two architectures. The other benefit of ETL is because the data is transformed in this intermediate Rosetta stone state, you can then send it on to anywhere. If you're in any to any environment where on a Monday you're taking data from SQL server and

pushing it to Oracle, then on a Tuesday, you're taking data from Salesforce and you're pushing it to Red Shift. Then Thursday and Friday, you're taking some mainframe data and you're putting it into MySQL database. Then you need an any to any tool, and the only way of really doing that beautifully is ETL because the transformation happens in one place, which is on the ETL tool infrastructure.

Now the thing that changed really was cloud data warehouses. As we talked about earlier, one of the big events there was the launch of Red Shift in late 2012. Because that made it very easy to stand up data warehouse compute power and storage and also much cheaper. In ELT, we extract the data from source and we load into the warehouse and we then apply the transformations on to the warehouse. The engine doing the work of joining the data together is the data warehouse itself. You can do ELT without any tools [inaudible 00:47:43], right? Load two tables on to Red Shift or load two tables on to Snowflake. Fire some SQL at it. Create a view or create a new table based on a join. That's basically ELT. On Red Shift or Snowflake, that SQL will execute very fast because those platforms are very good at resolving that sort of query. If you combine really great performance at that sorts of thing with incredibly easy to stand up and pretty cheap, then what you get is, is a brilliant target for doing ELT.

[00:48:17] JM: I think I understand. What you're saying is pre-cloud, or more accurately, pre the kind of hardware and processing we have available today, what you needed to do was take your data, put it into this system that would create the materialized views that you need from your gigantic amount of data and then you would get essentially the tables that you need and the tables that you need would be what you would think of as the data warehouse, and that's what you would be accessing from a reading standpoint, and you couldn't really do complex queries against that data as much or you couldn't have as rich of a query range that you could query against.

Then something changed in the hardware or the underlying algorithms such that today, all we have to do is extract the raw data. We just have to figure out what data we want and throw it into the data warehouse. Then within the data warehouse, we can create the materialized views that we need and read from those.

[00:49:30] MS: Yeah, that's pretty much right. I mean, I think something we always used to say a lot at Matillion, and it's just a finessing on a playback you just gave, was that actually ELT was always a great idea. I mean, back in the day, you could run Informatica PowerCenter in an ELT mode over Teradata. It was great. It runs superfast and it leveraged the ability of the underlying Teradata warehouse to crunch all these data in a very optimized way is brilliant.

Just not many people did it, because Teradata and Informatica were really expensive. You'd run Informatica in ETL mode and have the work done on Informatica that was slower, or you'd run on a tool that wasn't Informatica, one of those other ETL tools. Again, it would run on its own infrastructure and would be slower and you just put the finished result into Teradata, or Netezza, or Vertica.

The thing that changed in that hardware, as you put it, is suddenly the whole world had access within a couple of clicks, a couple of minutes in the entry of [inaudible 00:50:31] details to a brilliant target to do ELT on, which was a data warehouse. You have like a Teradata-like capability, but just super easy to standup and super cheap. At that point it made sense to say, "Well, we're not going to do ETL anymore, because it's a bit of a kind of computer science clutch and it's a bit slow and it has some problems with it in terms of development that slow us down. What we're going to do is move to an ELT mode because that barrier of cost has been take out of the way and suddenly the world is awash with brilliant targets to do ELT over.

The benefit, by the way, of doing ELT I would say is twofold. One, it's about speed, and perhaps as a byproduct of that, management of infrastructure. Data warehouses are really faster resolving analytical queries of the sorts that you need to make when you're doing ETL or ELT. They just run real fast, right? Go figure. That's what they're designed to do. Your jobs run faster and anyone that remembers looking out for a data warehouse in the 90s will remember long batch jobs running overnight that occasionally fail at 4 o'clock in the morning and the next day all the reports went out wrong. If you make it faster, that just doesn't happen as often, and ELT is much faster.

The other thing is you can do something at the development level with ELT that you just fundamentally can't and can never do in ETL. In Matillion, which I'm just using here as an example of what you can achieve with ELT, as you drag components on to the screen. So let's

say you drag three tables of source data. One Salesforce, one SAP, one – I don't know, Google Analytics and you start joining them together. You do a few calculations. You filter a bit. You change the granularity a bit and then you send it out to a materialized table that you're going to point Tableau up.

Imagine a canvas with a video component for each of those steps. That's exactly what it looks like, because Matillion is a code optional graphical developments environment. Each one of those titles as you build it out, you could look inside and live in real-time, see what the data looks like because it's actually there on the data warehouse because you're in an ELT mode.

In ETL, you can't do that, and that slows down the development process. The only way you can see where you're up to at each stage is running the job, piping stuff out to the console or another file, all that nasty stuff we sometimes have to do in development, and that that just talks to speed of productivity and development. Ultimately, ability to innovate with data or pace. Yeah, that's why ELT is better. It was better prior to the public cloud. It was just preventatively expensive to do. Cloud comes along, cloud data warehouse comes along and suddenly the world is awash with beautiful cloud data warehouses that make ideal targets for ELT in a really easy and cheap to standup. That's what we took advantage of.

[00:53:33] JM: All right. Well, we've very quickly gotten near the end of our time and I had so many other questions I wanted to ask you, but I guess just to close off, we're in this time where the market of data warehouses is getting quite competitive. How are customer demands changing? Is the market shifting away from Red Shift and towards Snowflake and BigQuery? Give me your condensed perspective for how the data warehouse market is changing.

[00:53:59] MS: We've come real closely with Red Shift and Snowflake and probably just out of living one of Matillion's value, which is doing business with Integrity. Probably not to talk in too much detail about what we've seen going on in terms of the shift. What we'll say is Snowflake has done an amazing job at building a real consequential company, an outlier of a company based purely on this thesis of cloud data warehousing.

As a byproduct, and I'm forever grateful for the team at Snowflake who I'd get on with really well for this, is they kind of resigned posted the industry to cloud data warehousing. They branded

Snowflake as the cloud data warehouse. When they did that, it wasn't a term we've invoked. It now is again, and that's down to that. They're both very significant actors in this industry.

For BigQuery, I can't talk to quite as accurately. We have a small business line of BigQuery. My instinct says that, in general, BigQuery is generally used more in MarTech use cases and other areas where GCP is normally resonant. Most of the big grownup projects we see are either Red Shift or Snowflake. We see a lot of both, but certainly Snowflake has done an amazing job. Bear in mind, they launched their product I think maybe 4 years after Red Shift, something like that, and they're now one of the, if not the player in cloud native warehousing.

[00:55:31] JM: Matthew, thanks for coming on the show. It's been really great talking to you and if you ever want to come back on, discuss more details of data warehousing and ETL, I'd be happy to have you.

[00:55:40] MS: Hey, it would be my pleasure. Thanks very much indeed for your time, and I'll speak to you next time. Cheers!

[END OF INTERVIEW]

[00:55:54] JM: Being on-call is hard, but having the right tools for the job can make it easier. When you wake up in the middle of the night to troubleshoot the database, you should be able to have the database monitoring information right in front of you. When you're out to dinner and your phone buzzes because your entire application is down, you should be able to easily find out who pushed code most recently so that you can contact them and find out how to troubleshoot the issue.

VictorOps is a collaborative incident response tool. VictorOps brings your monitoring data and your collaboration tools into one place so that you can fix issues more quickly and reduce the pain of on-call. Go to victorops.com/sedaily and get a free t-shirt when you try out VictorOps. It's not just any t-shirt. It's an on-call shirt. When you're on-call, your tool should make the experience as good as possible, and these tools include a comfortable t-shirt. If you visit victorops.com/sedaily and try out VictorOps, you can get that comfortable t-shirt.

VictorOps integrates with all of your services; Slack, Splunk, CloudWatch, DataDog, New Relic, and overtime, VictorOps improves and delivers more value to you through machine learning. If you want to hear about VictorOps works, you can listen to our episode with Chris Riley. VictorOps is a collaborative incident response tool, and you could learn more about it as well as get a free t-shirt when you check it out at victorops.com/sedaily.

Thanks for listening and thanks to VictorOps for being a sponsor.

[END]