EPISODE 794

[INTRODUCTION]

[00:00:00] JM: Bol.com is the biggest ecommerce company in the Netherlands and Belgium. For 20 years, Bol has been developing its software architecture, which includes a variety of services and databases and a mix of physical and cloud infrastructure. For an ecommerce company, the search engine is critical for allowing customers to find the products that they are looking for. Search also has many applications for internal systems. A search engine is a database with a query engine, and internal application developers want to build on top of that database.

Volkan Yazici is an engineer at bol.com specializing in search, and he's the author of the blog post *Using ElasticSearch as the Primary Data Store.* In his post, Voklan describes the process of scaling Elasticsearch to fit the use cases of both internal and external users at a large ecommerce company. This article was tremendously insightful and quite long. It's long, but detailed, and it's also indexed quite well. So it reads kind of like a small book. It's a story of how Elasticsearch has been scaled at a large ecommerce company, and I think it is quite useful for anybody who is deep in the weeds of an Elasticsearch deployment, or who's just getting acquainted with software architecture.

Volkan joins the show to discuss how search infrastructure at scale can require a carefully architected data pipeline in order to propagate changes to a large dataset that needs to be indexed within search. This is a complex problem, and Volkan is able to explain it in elegant and simple terms.

We're having a few events in the near future. On April 3rd, we're having a meet up at Cloudflare. You can go to softwareengineeringdaily.com/meetup to check that out and sign up before we ran out of seats. Haseeb Qureshi is joining me for a conversation about the world of investing, cryptocurrency, engineering. He's a cryptocurrency investor and has been on the show several times. In the past, he was an engineer at Airbnb and a professional poker player before that. So it's going to be a great meet up, and I hope to see you there.

Also, we're having an in-person hackathon on April 6th at App Academy. It's a hackathon for the company I'm working on called FindCollabs, and if you have a project you're working on, whether an open source project, or an art project, a music project, FindCollabs is a place to find collaborators for your projects, and most people would rather be working with other people rather than just in isolation. FindCollabs is a great place to find people to work on your projects with or to join other people in their projects. So you can enter into that hackathon by going to softwareengineeringdaily.com/hackathon and signing up for the in-person event, but it's also a virtual hackathon. So anybody across the world who wants to enter this hackathon can compete for the $5,000 price purse of the coolest projects being made. If you're interested, go to softwareengineeringdaily.com/hackathon, and I hope to see you there.

Let's get on with the show.

[SPONSOR MESSAGE]

[00:03:34] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and Mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals. Go to softwareengineeringdaily.com/g2i to learn more about that G2i has to offer.

We've also done several shows with the people who run G2i, Gabe Greenberg and the rest of his team. These are engineers who know about the React ecosystem, about the Mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization. In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily, both as listeners and also as people who have contributed code that have helped me out in my projects. So if you

want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW]

[00:05:25] JM: Volkan Yazici, you are an engineer a bol.com. Welcome to Software Engineering Daily.

[00:05:30] VY: Hi! Thanks having me for here.

[00:05:32] JM: I'm excited to talk to you about a blog post you wrote about using Elasticsearch in a creative way. But first let's set some context for where you work and the scale of the applications that you work on. So you work at bol.com, which is the biggest ecommerce company in the Netherlands and Belgium. What do you work on at Bol?

[00:05:55] VY: I like to call myself as Java shepherd in the fleet of search. So I've been working close to five years at bol.com, and during that time I've been involved in many aspects of the search, from building the search engine itself and also helping with the development of the ETL pipeline. So, basically, that I would say.

[00:06:17] JM: Bol has been around for almost 20 years. Describe the backend infrastructure.

[00:06:24] VY: As you can imagine, it involved from a monolithic architecture that used to be just a single Java application running on a single Oracle database, but over the time due to the humungous growth that company experienced, they also migrated to a microservice architecture. I think this had happened in the past decade.

So we have something like our own internal AWS. It is partly on our premises, so in our data centers, and it's partly running on Google Cloud. But what we are basically – Like we can spin up easily new services in a matter of minutes and start deploying our applications that are facing with the customers. Many people actually tend to ask about, "Hey, what kind of languages are you using, or what kind of databases are you using?"

But like this is a really west ecosystem. I think we have close to 500 developers and I suppose like almost 100 of them are what you can call devops. So they provide us certain facilities for the platform. So any language that you can imagine, but mostly we are a Java shop and we are also using plenty of storage solutions. So depending on a need, but we are trying to stick to the most cutting edge technology as much as possible, and I personally also really enjoy that.

[00:07:39] JM: Are there any internal rules around when you can use cloud services, when you have to use open source, when you have to use the on-prem infrastructure, or can people do what they want when they want?

[00:07:51] VY: Exactly. That is the case, like we have a SCRAM setting within the company, at least in the case of IT. So each SCRAM team has the responsibility of their own services. So if there is enough – Let's say you're just about to try Rust in a new microservice and if the platform is supported by that, what do I mean is like you need to integrate with the metrics, with the logging, with the health checks and so on, also the alerts that wakes up SREs in the middle of the night. If everything is totally fine, then it's up to the team to decide on the technology that they want to use.

[00:08:27] JM: What re some of the pain points of the legacy infrastructure that have cropped up in the last 20 years?

[00:08:34] VY: I don't know it is exactly a pain point, but like I would say the most dangerous thing with legacy, it works, and it makes money. So you can't just get rid of it in a blink of an eye. So let's you're about to build a new search engine, all right? Building a new search engine and migrating your legacy search engine to a new search engine is totally two different beasts. Like the transition period, like you need to take that in stages. Also, you need to employ A-B tests along the path you need to check and evaluate the success of the new engine, like the conversion rate, latency, throughput. Is the indexing keeping up with the data, and since this is something new, you are also experimenting with the technologies as well and most of the time you don't know whether they would stand against the scale that you had in your mind.

I think Elasticsearch is a really good example of that. Maybe not anymore, but like a couple of years ago when you would speak with people and tell them, "Hey, are you using elastic?" They

would definitely tell you, “Yes,” but if you would ask them, “What is the size of your index and how often are you updating it?” You get like various answers, but most of the time not in the scale that you had in your mind if you’re working at bol.com.

[00:09:48] JM: Let’s talk about search as a general application type. So in an ecommerce application like Bol, there are a gigantic number of entities that you want to search over. You could have customers. You could have products. You could have maybe categories that you would want to search over. At Bol, search applications are useful both to external customers who are searching for products, but also to internal developers.

Describe some of the use cases for search, broadly speaking.

[00:10:24] VY: First, let me get it straight. What you said is right. There are many entities that you can integrate into a search engine and serve them to the rest of the ecosystem so they can facilitate that. But at bol.com, the main search engine, its main focus is just product search, but it doesn’t necessarily mean we are being used just by the customers.

I think half of the ecosystem also depends on the product search. One of the basic examples that I can give is like the data science team. They want to correlate the traffic of the customers and the conversion of the products, conversion rate, which I generally use for like if a customer – Let’s say a customer is browsing to a certain product page, but are they buying it or not, or a customer is getting interacted with certain facets. Maybe they are using brand, maybe they are using color, but at the end of the day, are they buying that product or is it helping them find some other product category that they weren’t initially aware of? So data science teams also needs to get integrated with the search engine to correlate traffic metrics with these certain entities. That’s one part of it.

In addition to that, we have different services for serving to desktop customers and mobile customers. Let me think a little bit more about it. Also like live chat system. This is one of the systems that we also support. There’s also a smart, intelligent engine going on there and trying to semantically analyze the text that is being typed by the customer and trying to correlate that with a certain product or with a certain order. These are one of many things that we try to facilitate in the search fleet.

[00:12:08] JM: You use Elasticsearch. We've done a show or two on Elasticsearch. But I'd like you to describe what Elasticsearch is in your own words.

[00:12:18] VY: I think in 2019, if you want to build a search engine no matter which expert that you would go to, they would recommend you Elasticsearch. First of all, yes. It is like a document store So you could have [inaudible 00:12:30] that why not using MongoDB or just having a JSON fuel and a PostgreS grail? Yes, but like I think the main strength that Elasticsearch possess is its full text searching capabilities, so fuzzy searching, because it is hard. Elasticsearch is more or less like a distributed Lucene, right?

I think like Lucene is one of the most battle-tested full text searching solutions on the entire planet and Elasticsearch is racing on its shoulders. So like more or less it is your only viable option, like it or hate it, Elasticsearch is dominating this market.

But another aspect that I also really like in Elasticsearch is it is – I generally use a couple of dimensions to describe storage solutions. Is it distributed? Is it running on multiple machines? Does it support sharding? So can you split your data into multiple machines such that you can execute your queries in parallel? Does it support implicit indices? What do I mean by this is do you need to know the fields up ahead that you're about to query? This is one of the best features that I like about Elasticsearch. You don't really care with any of these three dimensions. It is distributed out of the box. It is sharded out of the box and everything is indexed by default, like unless you define your own configuration, but everything is indexed by default. That's really handy from a development point of view.

[00:13:56] JM: When we're talking about Elasticsearch from the point of view of a database, one way of thinking about the data that we are storing or that we're building in Elasticsearch is a search index. What is required to build a search index in Elasticsearch and what purpose does a search index serve?

[00:14:19] VY: Speaking for ecommerce – But I think basically it's like searching along the documents in addition to filters. For instance, in SQL you can type [inaudible 00:14:29] clouds that has Boolean filters. Is this field equal to that value? What Elasticsearch brings in addition to

that in a search engine, you can also type fuzzy queries. You can combine them with certain significance. In addition to that, you can create aggregates on a searching fields, which is also referred to as faceted searching. I think like faceted searching and fuzzy searching is the core of search engine I would say in the domain of ecommerce search.

[00:14:58] JM: These features, faceted searching and fuzzy searching, are there customizations that you would want to make to the fuzzy searching engine or the faceted searching properties or Elasticsearch. Are there things that you would want to specify that are domain-specific to Bol, for example, for the ecommerce applications?

[00:15:21] VY: I think the difficult part is getting the fuzzy searching right. Let me give a basic example for that. Let's say you just opened up your browser and went to bol.com and you type Harry Potter. What do you want? You really want Harry Potter book? Do you want a DVD? Do you want a VCD? Do you want a game? Do you want bag for your kid with a Harry Potter picture on it? Do you want a blanket, or do you want a t-shirt with Harry Potter in it?

So these circumstances, you really need to take the context into account, and context is something really dynamic along the flow of a customer's journey in your website. If I would have a prior knowledge that you're interested in movies, then maybe I can prioritize the DVDs, in your case, or if I would take your last couple of searches into account, maybe I can also conclude that, "Hey, this guy is really looking for a blanket. I should just show him something for the bedroom. He's not really interested in the DVDs or whatsoever."

So this context, analyzing that context and converting it into a query that is supposed to be executed by Elastic is quite difficult, and I think that the heart of business logic. At the end of the day, when a user types something, we are converting it just to Elasticsearch query. But the important bit is like really making the right conversion. I can tell you, pretty difficult. It is in the order of tens of thousands of lines of code just this business logic.

Coming back to your original question, yes, we need to customize Elasticsearch queries quite a lot, but is query language flexible enough to cover any of the cases that we can think of? We are way more satisfied with it than we initially told.

[00:17:10] JM: I see. So if I understand correctly, much like overtime we have learned to Google in different ways when we're searching for different things. Like on Google I might search Harry Potter movies if I'm looking for Harry Potter movies, because I know that if I just search Harry Potter, it's too general for Google to deliver me the Harry Potter movies.

Similarly, if you have a customer that comes to bol.com and has been coming to bol.com for 10 years and you know that they buy lots of movies on bol.com, if they go to the bol.com search box and enter Harry Potter, you actually have modules that sit between the bol.com search box and the Elasticsearch query engine that might translate a query from the website search box that the user enters Harry Potter into, and you translate that query into Harry Potter movies before it hits the Elasticsearch cluster. Am I understanding you correctly?

[00:18:13] VY: Exactly.

[00:18:14] JM: Okay. So there's a lot of middleware you could build in front of the Elasticsearch query engine itself in order to make it more usable for the user, and this also applies to internal users. We could talk about internal applications you could build. I guess before we talk about the usage that you've customized your Elasticsearch application infrastructure around – Let's talk a little bit more about Elasticsearch.

So if you have a gigantic ecommerce application like Bol, do you have to do any customization to scale an Elasticsearch cluster or is it easy to scale?

[00:18:52] VY: It is easy to scale once you get the configuration right, but that's, I would say, quite a voodoo thing. It requires quite a lot of experimentation. For instance, optimizing Elasticsearch for indexing performance is different than optimizing Elasticsearch for search latency. I can give a more concrete example for this. Time to time, we need to create our indices from scratch, and this is a process that can take up to hours.

So in such cases, we have a special set of Elasticsearch configurations. So we can create an index, load that configuration, which solely optimizes the indexing speed. It doesn't even care about the query performance at all. Then we started indexing, and then we also deploy post-

configuration to turn that index into one that can also cover query latency requirements as well, and this also another operation that is taking some time.

If I would speak more technically, there is the segmentation issue with Elasticsearch. So while bulk indexing and index from scratch, it creates quite some segments and you better wait at the end of the index operation to get merged by Elasticsearch.

Unfortunately, not all of these operations are under your control. So you can't really have an influence of all of these operations. If you would search for Elasticsearch segmentation problem or Elasticsearch increasing merge count, you would find like hundreds of blog posts mentioning about this problem and they're also giving away their own custom solutions.

Another trouble is Elasticsearch is evolving so fast. What we say problem today might have been solved in the next release that is coming in the next month, but it doesn't mean that your troubles are over. Maybe it brought something new that you weren't aware of. It is constantly changing and you need to adapt to them. So you need to keep an eye on your regular performance so you have a certain expectation about how long an index should take, what is the upper bond for query latency, what is the throughput upper bond for the throughput and so on. So if you see fluctuations on these – By the way, these fluctuations doesn't necessarily need to surface with just Elasticsearch version upgrade. Maybe you change your infrastructure, like you increase the number of data nodes, or the client nodes, or you added one new physical node, or you increased the RAM, or you decreased the CPU count per node, but you increased the size of the cluster in terms of nodes.

So you need to have a couple of stickies on your monitor and you need to make sure almost every day that, like, "Today, we are doing fine." Of course, there are like alerts in place that starts turning on the lights if something goes well. But most of the time we try to keep an eye on them.

[SPONSOR MESSAGE]

[00:21:54] JM: Deploying to the cloud should be simple. You shouldn't feel locked in and your cloud provider should offer you customer support 24/7, because you might be up in the middle

of the night trying to figure out why your application is having errors, and your cloud provider's support team should be there to help you.

Linode is a simple, efficient cloud provider with excellent customer support. Linode has been offering hosting for 16 years, and the roots of the company are in its name. Linode gives you Linux nodes at an affordable price with security, high availability and customer service. At linode.com/sedaily, you can get started with two gigabytes of RAM and 50 gigabytes of SSD for only $10. There are also plans for cheaper and for more money.

Linode makes it easy to deploy and scale your application with high-uptime and simplicity. Features like backups and node balances give you additional tooling when you need it. Go to linode.com/sedaily to support Software Engineering Daily and get your application deployed to Linode. That's L-I-N-O-D-E.com/sedaily.

Thank you, Linode, for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:23:24] JM: You wrote an article called Elasticsearch as the primary data store. Explain that title and what you are trying to accomplish with this article. What do you mean by primary data store?

[00:23:36] VY: Let me start with why that article, at least for me, was exciting. For instance, we have been talking about Elasticsearch for minutes, but we solely focused on its full text searching capabilities, like how would you purpose Elasticsearch to build an ecommerce search engine and so on? No one that much really attempt to use Elasticsearch as a database.

I think the main reason for this is Elasticsearch was never eager on claiming itself as a persistent storage engine, like they have a really market focus on fuzzy search, I would say. But in our experience, we also figured out Elasticsearch is really handy when it comes to implicit indices, like every field is indexed by default. This is like heaven from an application developer's point of view.

Long story short, it is not a recommended practice to use Elasticsearch as a primary data store. What do I mean by that is like the main source of your data, you don't have a backup or whatsoever. You don't have an RDBMS backing up Elasticsearch, just Elasticsearch itself. So it is not really a recommended practice.

But in the design of the new ETL layer, we figured that we can still purpose Elasticsearch and it can ease the implementation a lot, and given our experience with Elasticsearch in the last half a decade, we told it is like the way to go.

[00:25:03] JM: So in this idea that you're presenting of using Elasticsearch as a primary data store, because this is not something that is, I guess, widely discussed, you had to chart your own course. You had to figure out how to do this, and that's why I wanted to have you on the show. It's kind of a bold idea, and I would agree with you that this seems like Elasticsearch has the properties that we want out of a database, generally speaking. If you think from the point of view of a platform engineering team that's trying to provide a data platform for flexible use cases for different application developers within the company to use, maybe Elasticsearch is not perfect for every type of query, but it's pretty cool as a data platform to offer to your developers. It's obviously useful to the external application users. But in think through how to use Elasticsearch as the "primary data store", what were the challenges that confronted you?

[00:26:14] VY: First, everyone warned us against the update problem, like Elasticsearch doesn't have a good reputation for standing against high-frequency update rate. So like what do I basically mean by that is most of the time people are using a read-only index to execute their queries on and they just refresh that index a couple of times every day.

In our case, in the ETL layer, we would be hammering Elasticsearch with updates quite a lot, and time to time it is in the order of millions per second. Of course, Elasticsearch doesn't give you consistency and persistence guarantees of an RDBMS. That's totally a different ballpark figure. But Elasticsearch, I think, if you configure it good enough and if you are aware of its shortcomings, it has a couple of smalls what I would call maybe like small Swizz knife, really small – Let me give a more concrete example. For instance, I'm really in love with the [inaudible 00:27:15] web operator that is provided by Elastic.

For instance, once you insert the document, it automatically gets a version. So you can use that version for the next update. If you would think of Java, actually, the entire Java.util collection Atomic package is built on that [inaudible 00:27:33] operator. So to be able to implement lock-free data structures, they leverage that [inaudible 00:27:40] web operators, and that is exactly what we have done in Elasticsearch.

When we need to make an update, we fetch the document, we mutate it in the memory. While writing it back, we tell Elasticsearch, “Please write it back if the version still matches. Otherwise, just trigger me a failure and I will retry that.” So that aspect of Elasticsearch I think really paid off.

On the other hand, we also hit a wall of segmentation problem. We are still having issues with the Elasticsearch segmentation problem. Also, if you use nested fields a lot, that is known to be quite problematic with Elasticsearch again.

What do I mean by nested fields is basically list of objects. I’m not really an expert on the scene, but to the best of my knowledge, querying over a list of objects in Lucene is, I’m being told, like more or less like a hack. So it is not really one of the main strengths of Lucene, and I think this deficiency of Lucene is exposing itself in Elasticsearch as well.

When you start using nested queries, you need to start to get some quite performance impact. If you are trying to update nested fields, then it is doubling this performance impact. Fortunately, in the case of ETL, query latency is not much of a problem, because we are not really querying the search engine like a search engine of an ecommerce website.

For instance, we are not even performing fuzzy search at all. We’re just interested in if the field A equals to B, or like really basic filtering options. If the field C is greater than 1, those kind of simple predicates. So far, Elasticsearch is doing fine.

[00:29:20] JM: The first challenge that you articulated, if I understand correctly, is updating the search index. So within a company like bol.com, where you have thousands of employees, I believe, you might have operations people, and those operations people are working with internal applications to add new items or to modify items.

So let's say there's a supplier that has a new kind of t-shirt and they want to add that t-shirt to the bol.com product index. So the person working – The internal application user within bol.com is going to add that t-shirt and the different categories and the different colors and sizes that are associated with that t-shirt that make up the metadata, and you want to be able to index that product within your Elasticsearch index.

So if there're different people throughout the company that are adding these kinds of items, or removing items, or adding categories. Like let's say they want to update an existing t-shirt, like they want to add the word cotton to it so it gets a new category, you need to be able to update your Elasticsearch index with these new items. You need to be able to update the categories on existing items, and this I believe brings us to the problem of ETL. Can you explain what ETL is and how that applies to updating the Elasticsearch index?

[00:30:59] VY: ETL as a common paradigm is used in these kind of data pipelines and it stands for extract transform layer, but I think you get it pretty accurate. Let's first start with a basic example. If a customer changed one of the attributes of a certain product, it is not a big deal. It is just a single product update. Like there are some really bizarre changes that can propagate so big that it can cause the collapse of the entire system. I mean, collapse in terms of not being out of function, but really causing load on the system.

For instance, recently, one of the problems we had was like they decided to change currency rates, exchange rates pretty often in a single day – And in the search index by the way, the prices doesn't need to be 100% accurate. This is always the case, because like the prices that you actually see on the webpage is being populated in the last second. So let's say you go to search engine, it returns you back 10 products and then you go to another microsystem and tell, "Can you give me like the most recent prices for those 10 products?" and then it returns you back. So that's what you display to the user.

But it doesn't mean that price that is stored on the search engine is really that much accurate, but you still need to filter on the price. So price still needs to be in this search engine somewhere, and let's also assume the very same customer who changed the t-shirt of the color also have all of her t-shirts in the currency of USD, but on our website, we are displaying them

in Euros. So we need to convert them to their Euro rates, and if there's like too often changes to those rates in a day, it can cascade into millions of product changes every single hour.

Then this also brings up the question, like you also need to prioritize certain updates and maybe time-to-time discard certain updates on their heavy loads and postpone them for – I don't know, for a night, or for tomorrow, or when there's room to do them. So this like update volatility is quite painful. As we also discussed in our previous question, Elasticsearch is not really good with those high update rates, but yeah, they happen.

Also, people tend to think at bol.com, it is just bol.com who is changing the content, but that is definitely a big no. We have gigantic integrations with thousands of sellers and they have their own systems, and those systems are also streaming updates to us. So it is like a giant funnel coming to a stream into our ETL pipeline and you need to prioritize them, you need to transform them and you need to merge them into their final state, and then you need to deliver that final ETL state into the search indices. I would say these subjects are really like the bottom part of the iceberg, what many people tend to miss.

[00:34:05] JM: How rapidly is the content catalogue within bol.com changing, and how aggressively do you need to propagate those changes to the search index?

[00:34:17] VY: Most of the time, the category tree changes are happening in small amounts, like they're adding small category or they are just splitting an existing category into two small sub-categories, because you also need to take the UX into account. Like maybe people are really accustomed to a certain category tree display. So you can't go with a big bang splitting a giant category into multiple categories or combining giant two categories into one category. So they need to be – Those kind of gigantic changes need to be aligned with the customer and also with the systems.

A recent example that I can also give, I think last year – Sorry, two years ago, we announced like a revamp of the toy shop that we had back then. So it was touching like hundreds of thousands of products. Then they told us like they're about to release commercials on the TV and they need to come up with a new category tree. Then we fixed their agendas with them. So they said, let's say, "At this state, before 7 in the morning, we must have that category tree."

In those circumstances, we can create exceptions for the business. We can maybe prioritize those messages or we can stop the rest of the messages and just let the category tree changes come in. But I would say 95% of the time, the changes are pretty small. They are moving gradually instead of going with big bangs.

[00:35:44] JM: Tell me more about the data infrastructure that we need to think about here. So let's say somebody within the company makes a change to a product in the core database that is your product database. First of all, that's one database, and then you need to have some way of propagating that there has been a change to the core data model through the system and queuing up the search index to be changed. Then you need to actually execute the change on that Elasticsearch engine.

Tell me about the different application components from the database that actually gets changed that represents the products to the middleware, kind of the queuing and processing pipeline and how that actually results in a change to the Elasticsearch index.

[00:36:36] VY: I think that would make quite a story, which I probably wouldn't be able to address exactly during this conversation, but I can give you a 2,000 feet overview.

[00:36:45] JM: That's perfect. That's what we're good at on the podcast.

[00:36:49] VY: All right. Cool. But like my point is that it is not just two or three services involved in that. I think it's close to more than a dozen services which needs to coordinate and align for the changes. Maybe some of them are communicating asynchronously with the queuing system. Some of them are communicating directly. Some needs a feedback loop. Some doesn't. Some need certain delivery guarantees, some doesn't. Some needs to be up all the time, like 24/7. Some doesn't necessarily need to be up all the time. There are quite some actors in this game.

But in a nutshell, as you pointed, out product catalogue changes, like they add any attribute or they just introduce a new product or they add any offering, selling offer, to an existing product.

That ends up as a message, our ETL layer first. In the ETL layer, we have what we call the configuration that is managed by our business people.

As a basic example, in the website, when you go into color facet on the left hand side, you generally see main colors; red, blue, green, these kind of things, or maybe a mixture of them; red-blue. For instance, let's say you have a color which is named red-blue, but in the original attribute, you probably don't have a red-blue color. So maybe the original color of the product was red, blue, green. So in such a case, then you need to take the configuration that's been done by the business into account.

So the business tells that for this attribute, when it has this value, I want to have a facet with this name and it needs to show this value. In the ETL layer, we basically apply these transformations. Most of the time, the stream that I just described is the easy one. If there's a product change, it reaches to us. We look at the configuration. We apply those configurations.

I think the tricky part in this entire flow is what if you have tens of thousands of configuration, which is the case at bol.com. So we also have sort of like our customized data structures to speed up those lookups. Let's say there are 10,000 of rules and just the color attribute is changed. So you shouldn't be executing 10,000 rules. So you should just find the ones which are interested in that attribute and execute them. So this is like I think one part of the coin.

The other part, which I find the daunting task, if someone makes a change to the configurations, that's like hell of a job. It's really difficult to get it right. I try to explain this briefly in the blog post as well, but if I would try to rephrase it, let's say in the past you said if the language attribute of a product is English, I want this product to end up in the category of English books. But at some point, a smart guy might decide to say, "Hey, this is not really a good idea. We shouldn't have categories like English books and Dutch books. We should just have one category, which is called books." So he changes that configuration.

So this creates a rippling effect. Then you need to find all the products that used to match with the old rule. You need to rollback all layer changes and then you need to find all the products, which are matching with the new rule, and then you need to execute the new rule on each of

them. These can really result in almost the recreation of the entire index. So, yeah, this is the most challenging task we have been dealing with.

[00:40:25] JM: For people who want more than the 2,000-foot view that you just provided, they can read your article, which is quite long and detailed. Can you say more about what you outlined in your article? What were you trying to present to the reader?

[00:40:43] VY: I think just like applying configurations to streaming data and retrospectively projecting back changes in the configuration. This is two orthogonal tasks, and I think we managed to get this right, but we got this right, because we have been trying to get this right for almost 20 years. So, anyway, I can't say I'm standing on the shoulders of giants. So I probably wouldn't be able to come up with the right solution.

So I think this is the gist of that blog post. If you would really look at this problem in detail, then you would also be able to see where Elasticsearch shines. What I call by configuration, these are rules, right? These are rules written by business people. They just say, "When this attribute is equal to that, do this."

The problem is, while building the ETL layer, you don't have a control over that attribute. The products change overtime as well. Maybe you have a subset of attributes today, but tomorrow some of them will be deprecated and new ones will be added. So having such a whitelist is really a bad idea, and we passed that route and we weren't really satisfied with it.

Then this brings up a good opportunity for Elasticsearch when you don't really need to care about which fields are queried. I think this is our main justification of using Elasticsearch for the ETL pipeline. You can query on any attribute that you want. In addition to that, you can also use it for storage for simplifying [inaudible 00:42:15]. These are the crucial needs for an ETL pipeline, I think, and this is more or less what I tried to cover in the blog post.

[SPONSOR MESSAGE]

[00:42:32] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and

leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[00:43:55] JM: Let's take a step back. So you worked on this re-architecture, this Elasticsearch and ETL pipeline. What was accomplished because of your work? What can either users or internal application developers at bol.com, what can they now do that they could not do before you went through this process?

[00:44:21] VY: In the earlier days, when we didn't have this layer, what we used to do is if you would want to create a new category, then you need to talk with the product owner of a SCRAM team. He needs to look at his agenda, talk with his developers how much time would it take to add this category. Would there be any impact in the rest of the ecosystem? Maybe there are some hard-coded dependencies on this category and so on.

But with this new layer, first of all, we had a configuration. So it is like the central unit that defines how should the search look like. So you don't need to have any hard-coded configuration anymore in your code. If you want to have the latest category tree, the service is there. You just need to look it up.

In addition to that, in the previous ETL pipeline, it was a giant PL/SQL mesh running in really beefy Oracle database machines. Let's first agree on something. Even if it would have been

written in COBOL, it doesn't mean it is in a state that it's not impossible to improve it. You could have still achieved everything that we have achieved with the old layer as well. You just need to be really good at PL/SQL and you just need to move the worlds to integrate it with the rest of the ecosystem, but it's doable. It's not something impossible.

Another caveat of that system is it really requires quite some technical knowledge into the system to make some change. So I would say the new ETL pipeline that we came up with the design is like way more lighter in terms of cold complexity, and we are trying to avoid any kind of hard-coded configuration as much as possible. So everything is totally configurable. Most important, it is real-time streaming, maybe real-time is a big word, but let's say eventually consistent.

When you make a change to the configuration today, depending on the radius of the impact, we are putting our best effort to reflect that as soon as possible into the search indices. In the old system, as I tried to exemplify, if you wanted to make a category, it was taking time to time weeks. But today, if you want to make a category change, it's matter of minutes. If you want to add a new facet, it's a matter of minutes. If you want to change an existing facet, it's, again, a matter of minutes. This is quite a win for the business and they have been longing for this feature for years.

[00:46:46] JM: Well, you can imagine the frustrations of – When I worked at Amazon, I saw a little bit of this, this kind of thing, where you have these different departments that if they need to coordinate on a minor change to product catalog, this can really slow things down. This can lead to hours or days of communication problems, and that's because it's a non-trivial problem. I mean, you have different geos and different types of users and different layers of legacy software. It's just very complicated to scale an ecommerce application.

Even at 20 years, 20-years-old, you're going to have these manual processes that are just a result of the challenges of scaling an ecommerce application. Then you have no choice but to sort of say, "Okay, we've got this problem where, literally, updating a category on an item and reflecting that in the search engine is really hard and it's going to take us a long time to refactor it." This sounds like a small problem to some people, but it's actually a gigantic problem operationally, and we need to invest the engineering resources to refactor it.

[00:48:06] VY: Yeah, definitely. After an amount of time, it is not just engineers who get frustrated. Business as well gets frustrated and they start telling, “Tell me how much time and money it would take? I’m ready to pay it. I really need it now. Actually, I need it yesterday. Let’s do it.” Fortunately, we weren’t at that point, but this can come, this can happen. This is not just my imagination.

[00:48:31] JM: Absolutely, yeah. If you have a problem like this and it really starts to slow down the business, it can spiral out of control and it can really damage a business if you have a problem like this.

[00:48:41] VY: I think another aspect of running an ecommerce business, being in the software business in the domain of ecommerce is that like our product is not software. We are not selling our software. We are actually doing something else. We are doing commerce. So we are actually trying to sell products and we are using means of software to do this, and there’s a huge amount of room for improvement and innovation in the ecommerce, but this is a little bit orthogonal when you think about like the software development itself.

As a matter of fact, for instance, we designed an ETL layer with the idea that facets are always on a product level, like attributes are always on the product level. We always said to ourselves, “A color attribute can always be applicable to a certain product.” But nowadays, business is discussing can we have facets at the offer level? So did you get what I mean?

Actually, they want to have certain facts that are just enabled for certain product offers, not for every product in a certain category. I mean, I can implement search engine at bol.com till the end of my life. Even if I’m 80-years-old and if you’d knock my door and if you’d ask me, “Hey, Volkan. Are you done?” I would probably tell you, “I’m not even halfway there.” Because there’s such a huge room for business innovation. This is also driving the software development to its extremes. This has its ups and downs. It is really exciting to work in such challenging problems. On the other hand, it’s also really difficult to get the code quality right, like test-driven development. Things you care about software as a developer doesn’t necessarily need to be shared by the business people, and these are, time-to-time orthogonal things, and it might

become difficult to get them aligned and put them on the same page. Yeah, this is the way it is I suppose.

[00:50:42] JM: I want to ask you a question. I'm not sure you'll have an answer to, but I'll ask you nonetheless. So some of the people I talk to in the Software Engineering Daily community, they think about ways that they can invest in various software trends, and one of the differentiated companies that went public recently is Elastic. There's this, I guess, ongoing discussion over, "Oh! AWS has an Elasticsearch offering," and then Elastic, the company, is built around Elasticsearch. Some people will say, "Oh! Clearly, Amazon's service is going to be good enough," but the Elastic folks would probably say, "Actually, the problem of search is so hard, it's so complicated, and there's so much room for product innovation and maybe machine learning applications built on top of it, and better UX, that we can still build an enormous company around just this domain, and AWS won't be able to compete with us, because it's not existential for them."

Do you have any perspective on search as a platform as a service or infrastructure as a service offering for different software companies that have focused on offering it?

[00:52:03] VY: I think it might be really tempting for small and midsize companies, because most of the time they don't really need to care about the operational costs of their medium-sized Elasticsearch indices. But on the other hand, the impact of this on the open source, free and open source software ecosystem, is really ambiguous at this stage, and I really don't think I have the right expertise to comment on it. But I would gently pass this question. I think this is a really delicate question, and it really needs to be addressed correctly, and I don't think I'm the right guy to say a word on this.

[00:52:38] JM: All right. Fair enough. Fair enough. Last question; who else should be considering Elasticsearch as their "primary data source", or how should the software industry as a whole be rethinking using search engines as core database infrastructure?

[00:53:00] VY: I generally, for that purpose, use the metrics that I briefly described. It is just like put your storage solutions on a table, right? Is it distributed? Is it sharded and does it require explicit indexing, like this column needs to be indexed, that column needs to be indexed. Then

you can also add extra dimension to these metrics, like what is the average size that this storage solution starts to shine? Then it all depends on the problem that you have at hand.

I'm really optimistic with Elastic. I mean, I already told you about it. I have a love and hate relationship with Elastic. I like it so much, on the other hand, it has certain shortcomings that I hate. But I think it's a reliable product, it's a really solid product. People should start concentrating it as a primary data store as well. It was really unfortunate in the past like a couple of years ago, Elastic officially stated in their website like, "Please don't use Elasticsearch as the primary data storage." Yes, that happened.

At some point in time, there was such a phrase in the official Elasticsearch website, and it was really unfortunate situation. Right now, no matter how much effort they put on improving their database, no one is really taking them for seriously. But I would say forget about my blog post, or whatever other guys say. Just do your experiment. You have already your problem, evaluate it against multiple storage solutions. Just put up a benchmark, see how do they perform. Also, you need to take into account the operational costs as well. For instance, some databases are really difficult to get it right operationally.

Maybe I shouldn't be saying this, but like Cassandra is really difficult to operate if you want to have it in your own premises. But compared with that Elasticsearch, you can get it right without breaking a sweat. Yeah, just give it a try. I think it's really a feasible, solid option.

[00:54:55] JM: Volkan, thank you for coming on the show. It's been really fun talking to you.

[00:54:58] VY: Thanks so much for having me, Jeff. It was my pleasure as well.

[END OF INTERVIEW]

[00:55:05] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]