Tobias Ternström
You will start in the public cloud because the public cloud is very cost effective for smaller workloads, even though they generally have a higher price per compute, if you will, per compute second, it still makes a lot of sense to start in the public cloud.

Tobias Ternström
Those most critical databases would be the last that you automate and probably also the databases where you're more selective into what kind of automation you use.

Tobias Ternström
You have a team of experts. You want those DBAs to focus on literally the top handful of databases. So let's say you have five to 10 out of your 10,000 databases. Those are super important. That's where your DBA team should spend the absolute majority of their time.

Jason Lopez
That's Tobias Ternström. He runs engineering for database workloads at Nutanix. This is the Tech Barometer podcast, I'm Jason Lopez. What you're about to hear are his thoughts on databases in the 2020s. So what makes his views important? He's worked at Microsoft on Azure, at Google on cloud databases, and at AWS on Amazon Aurora. In this podcast he covers topics like the explosion in databases, databases on hybrid multi clouds, automation and careers in databases. Tobias emphasizes that a company's databases are critical properties and thus company leaders and decision makers, who may not have specific expertise in the technology, should absorb what they can about database engines. That's one of the aims of this podcast, to provide insight. We started off the conversation on the kinds of databases out in the marketplace.

Tobias Ternström
So generally there are two databases in the world, as far as I see it kind of simplistically, but I think it makes a lot of sense. So you optimize for one of two things generally. So either when you build a database and you optimize for features, so the database can do lots of things. A good example here would be know relational database, like post SQL or, or, you know, document database like MongoDB, both I would argue or optimized for features you can do, you know, joins. You can do lots of different types of queries. They have query optimizers. You don't have to think about exactly which index to use the database kind of figures out for you. But the problem is because of the fact that a query, when you submit it to a database, it's a little program. So you never know exactly how much time the query will take to run. And how many machines, if you use multiple machines for the data will need to be involved. So it's hard to get predictable, especially predictable, right performance, but predictable performance after this. So you have to be very careful with how you write your app. The other type of database is basically a database that's focused on predictable, write performance. At scale examples, there would be Cassandra or DynamoDB and AWS or big table and GCP. So you have to basically pick between these two. I would hope that over time, it becomes easier when you write an app to use a single API independent of if you want to use the feature rich side of things or the write scale outside of things, obviously it needs to be clear to the app because otherwise you kind of

break the premise or the feature rich versus versus predictable, right? skip performance at scale. But right now I think it's fairly Ify as a developer because you're building an app and you use multiple database engines. You use my SQL or, or Postgre SQL and use, you know, read this for a cash and maybe use elastic for search, maybe use Mongo for documents. There's a bunch of APIs to use. And your data gets kind of siloed across all of these APIs. And you also have to become an expert or learn all of these APIs. And that obviously slows down development. So it would be nice if we went towards a path where even if you have optimized database engines, you can have a simple API experience, independent of the backing technology.

Jason Lopez
Let’s add a dimension to this. What’s your sense of how the movement to hybrid and multi-cloud systems is affecting how people manage databases?

Tobias Ternström
Yeah, sure. I think it's important when we talk about hybrid multi-cloud or you talk about cloud, generally, you talk about this movement of data and applications from on premises to the public cloud, but I think that's really a misnomer that's mainly because cloud is obviously newer than on-premises, but really what I think we'll see more and more, and we already see is you will start in the public cloud because the public cloud is very cost effective for smaller workloads. Even though they generally have a higher price per compute, if you will, per compute second, it still makes a lot of sense to start in the public cloud. Now that could be because you're building a new app or because you have something existing and you want to now launch in a new region. And obviously in that region, you'll start in the public cloud. But as you gain more and more momentum makes sense. If you have a very burst the application on average for long periods of time, it's not using a lot of resources. Public cloud is, will be very, very post effective, but workloads that are more steady state over time, as they grow, you'll find that it's more cost effective to run them in something like a colo. And then as it keeps growing, starts making more and more sense to run them on premises. Now provided you can build on something that gives you the same cloud operating model, if you will. So I can get the same database automation in the colo and in the public cloud and on premises, then I can scale and I can move my app as I evolve, if you will. That helps me basically run as cost effective as possible. So you want to make sure that both your application and your database is moveable between these environments, either because you're moving between regions, right, and expanding a footprint between regions or because you're growing your workload. And over time, you may find that it's more cost effective to run it on your own servers or Nicole or on premises. You know, we often, when you talk about cloud, you talk about and say, well, you know, no one runs their own power plant at home. You get the electricity through the power grid, but now with solar, more and more folks are saving quite a lot, not using the utility, but instead using a combination of solar and batteries, which is very much akin to going from the public cloud towards on premises. So it's the same analogy as also renting versus buying your own home over time, buying your own home is going to be more cost effective than renting.

Jason Lopez
Has this changed how you look at or design databases?

Tobias Ternström
Yeah, I think it changes how lots of people look at it. Now, the nice thing is if you start in the public cloud, which also that's the natural way, if you will, if you start in the public cloud, you generally want to start using this little bit higher level of services, like a database as a service. And if you build on that, that also makes your app generally more portable provided the database service can be found elsewhere. So if you build on a database service, that's completely proprietary to a certain cloud, you obviously will have higher switching costs. It's going to be trickier to move. I think most people that have been in idea have experienced high switching costs, especially related to databases. You obviously want to pick a technology where you can find the same API if you will, with a different vendor. So if you, by build on something like Postgres SQL for examples, open source database, even if I use a certain service for that, as long as I know that that's say API with similar performance characteristics is available somewhere else, I'm fairly portable. And generally, sometimes you hear people talk about failing over between, I don't know, Azure and GCP or, or Azure and AWS. Obviously there are apps where you want that type of availability, but those are extremely far and few in between where you talk about, I need to move in literally seconds here. It's more, can I move in a matter of days or weeks without a lot of risk. And that's really how you want to think about it.

Jason Lopez
Tobias Ternström grew up in Sweden. As a kid he would accompany his dad to work at offices in Stockholm where in the basement were some older large computers and mainframes. These were the kinds of toys that would hook him on information technology.

Tobias Ternström
I started writing code in the eighties and nineties. I started writing code, but that was on, on PC.

Jason Lopez
Well, what is it that led you to databases?

Tobias Ternström
Databases was completely by accident. One of the things was databases and using SQL server. So I used to basically code in the evening and teach SQL server in during daytime. And it was mostly because we had a startup and we were self-funded and in order to basically make revenue while we were building applications, we taught, there was a lot of demand for databases. So they asked me to go and teach, basically development and administration of, of SQL server. So I started on that.

Jason Lopez
Well, to start simply, can you tell us, what is a database?

Tobias Ternström
Well, databases have been around for literally thousands of years. So it just used to be written down, look at the index of a book, for example, that's a typo database, right? You look up in the

back and you find out which page contains whatever it is you're looking for. So the first big thing that happened was obviously going, digital. So I think it was bank of America in the, in the sixties, if I don't misremember that first digitized banking. And then from that database has gotten into absolutely everything. And there is obviously lots and lots of revolutions that's happened along the way. One thing that's remained fairly constant since let's say the seventies, when it developed the relational database model is still, you know, going strong today. So relational databases is mainly what the world runs on as well as, there was a big thing that happened end of the two thousands and beginning 2000 tens where no SQL databases really became more and more use, which basically straight away from the relational model due to mainly reasons of scale.

Jason Lopez
Well, let's talk about managing the explosion of databases. How did we get where we are now with thousands -- tens of thousands -- more databases than we used to have?

Tobias Ternström
You know, there is the famous quote that said, I see a market of, you know, couple of computers in the world. And after that, obviously, you know, computers are everywhere. It's the same with databases. I'm sure most companies, if you go back, couple of decades said, oh, you know, it would be great to just have one database. We have one database, we have all of our information in it. We can assign probations and let people query, but it turns out trying to centrally plan. This is very, very difficult and it slows down development. It started with, you should have just one database. Well then it comes, oh, I need to add a table into this database. You do, we have space for this table. How much compute power does the table need? And so on and so forth. And at some point you realize that that's going to be very, very complex to try to keep everything in a single database. And back then obviously a single database meant a single computer. And for the most part, most databases out there today, it tends to be single writer. So single computer that handles the rights for the database. So the next step was, well, we should use just one database engine, at least. So we standardize on something like Oracle or Informix or SQL server, but that also turned out to be tough because as you're building applications, different applications have different requirements. It may be around cost, maybe around scale. It may be around features. So you start saying, well, if I'm building this application, maybe I should use a different database today. It's definitely very different because you generally don't start from scratch. When you build a new application, you generally start from some application framework, some app, if you will, that have built obviously lots of ISV apps out there that folks build on or take an open source project. That's very popular WordPress. So if you're building on WordPress as your basis, you're going to use my SQL because that's what, what WordPress uses WordPress supports other databases also. But by and large, it's, very high percentage. It's my SQL. It's gone from, let's try to keep it one and then realize you can't, then let's try to stick to one database engine, then realize you can't. And then basically as applications evolve, or even now microservices evolve, you'll find that a single microservice may use one database where another microservice in the same app, if you will uses a different database

Jason Lopez
As you're now dealing with more than one database engine, what's the most sensible way to deal with the quantities and varieties of databases?

Tobias Ternström
Let's say you have 10,000 databases in your organization. Some of them are development. Some of them are test. Lots of them are production. First of all, it's all databases aren't equal. If you want all of these databases to have the maximum number of nines and performance and so on, you can obviously do that. But at cost, at the end of the day, you want to first decide which of these databases are absolutely critical for the organization. And those you invest in one way versus in other databases that maybe important, but not mission critical. You invest in a different way. And generally what you want to do is as you have a team of experts of DBAs, you want those DBAs to focus on literally the top handful of databases. So let's say you have five to 10 out of your 10,000 databases. Those are super important. That's where your DBA team should spend the absolute majority of their time. Optimizing doing database design, architectural of this and the rest of all of these databases. You want to automate to make sure that they are managed automatically, including patching again, configuring disaster recovery, high availability, and so on that's at the end of the day, how you scaled it.

Jason Lopez
That's interesting, the idea of focusing on the top databases. Is the profusion of databases making life difficult for companies? Or, even, database administrators?

Tobias Ternström
Well, the profusion of databases is really driven by applications and developers. So developers trying to solve a problem. They want to use the best tool for the job. So that applies to databases as well. So they pick the right database for the job, obviously because of this explosion of apps, microservices, and using the right database for the job organizations just run more and more databases and they run not just one or two, but multiple database engines. And at the end of the day, the tricky thing becomes, how do I make sure that all of these databases are configured, optimally are being patched or being backed up or configured correctly for high availability and disaster recovery. And having, for example, a DBA team, that's an expert in each of these database engines and can handle that at scale becomes very, very tricky.

Jason Lopez
The next part of our conversation turned to database automation. Automating allows people frees people up to focus on higher level tasks. Most database administrators would prefer to work on database design, database architecture, database performance tuning, rather than patching or configuring high availability or configuring disaster recovery.

Tobias Ternström
The thing that I think hits people, and I know lots of folks have built automation, scripts and solutions themselves is that it becomes difficult to scale it. First of all, the devil is in the details. So there is lots of things as you automate that you realize there are special cases all over the

place. What if this particular thing happened? Well, my automation script dealt with case a, but it didn't cater to case B, then I'm in trouble.

Jason Lopez
So are you saying that if you have one thing that's wrong in the database automation it can render the whole thing unusable?

Tobias Ternström
Yeah, absolutely. Automation is at the end of day, day, you're automating something and that something is taking an action. It may issue a backup. It may create an index in the database. It does something, whatever that action is, it can fail. So then what should you do if that action fails? and do you need to roll back part of the action? Right? The action may have multiple steps. Those steps may not be possible to maintain in a database transaction, for example. So how do you walk back the steps if it fails halfway through, if you will, if it didn't have the intended consequences, for example, then how do you deal with that?

Jason Lopez
Hmm, interesting. It's kind of like Christmas lights.

Tobias Ternström
Yeah, that's exactly right. And so that's part of the problem. Then you get into the, well, I run multiple database engines and I run multiple versions of each database engine, and I may also run them on different operating systems and that quickly expands the testing matrix and the issues that you can run into.

Jason Lopez
I see. Well how do you map out the business case for automation, especially when high quality developers are so difficult to find and even keep on staff.

Tobias Ternström
Well, the business case, I think goes in two directions, one business case is obviously keep yourself safe. And that means having up to date patching back up, Dr ha and so on. It's the kind of, what if scenario, if you will. So that's part of it, of the business case. The other part of the business case is as you get more and more databases and more and more database engines, which the direction is pretty clear, that's what happening? Do you hire more and more individuals to manage those, or do you make sure that the folks that you have on staff can focus on higher level tasks and look at the whole, are we investing correctly? Are we, you know, focusing on the right database engines, are we performance tuning the right databases, so on and so forth? I think that's generally the business case that you want to make sure that the DBA team, the administrators and the developers that you have are effective and efficient. The other problem that you're running with developers is generally that developers need databases as part of day to day activities. So development testing, and so on part of a continuous integration, continuous deployment pipeline. And then is this all automated or does the developer have to work with, their DBA team in order to get those databases? And generally

DBAs don't to be bothered by developers. They want to focus on making sure all of the databases run smoothly and the developers definitely don't want to wait on getting their database.

Jason Lopez
One of our writer's Tom Mangan proposed this statement and we wonder how you'd riff on it. Let's just automate the management of our massive database portfolio, what does that mean to you?

Tobias Ternström
It's, it's no black and white it's gray. So first of all, you want to automate your least critical databases. First you have your 10,000, you want to start at the bottom and do the first hundred or thousand first the lowest. And then you work your way up. And generally those most critical databases would be the last that you automate. And probably also the databases where you're more selective into what kind of automation you use in a high availability setup. It means like if one machine, for example has an issue, I can fail over to another machine or maybe I fail over between buildings. If one building, for example, loses power, but I can only fail over automatically. If I know I had no data loss, if I can have data loss, it becomes very, very complex. If I don't replicate my data at all, that's clearly a problem. Then you will just lose your data. But if the connection between the two computers is fast enough, I can do what's called synchronous replication, which basically means I send information over to the other computer. And before I tell the user that I actually save their order or whatnot, I wait for an acknowledgement from that other machine. So I know that it's not just saved with me locally here. It's also saved on this other machine. And generally I wait for the other side to say, okay, the problem is that at some point, the speed of light gets in the way, if you will. And I can't wait for this acknowledgement because it slows down the application too much. So between two sites, for example, within the same city, let's say it's not a problem and I could do synchronous and I can automatically fail over. But let's say I have a data center that goes down in the region that takes us example of it goes down and I have, maybe New York has my fail oversight. And if I now do asynchronous replication, which means there may be a non zero recovery point, objective, meaning I may have lost some data. It might be seconds might be minutes, might be hours depending on how I configured things. Then generally you can't just make an automatic decision on, do I fail over? Because let's say we lost one minute of data is, was in Texas now contemplating moving over to New York. Is it good now? Should we wait another minute? And see if Texas comes up where we have no lost orders or whatever is going on or EKG records at the hospital or whatever it might be, right. Or do we fail over now and take the data loss? And that's generally a human being that needs to make this decision because obviously depends on, do we think there is a chance that the Texas data center or this machine will come up again while there is a chance, depending on you know, how critical the application is. You may opt to wait or you may opt to fail over. And obviously if you failed over to New York, now it continues on this new timeline. So if takes later comes up again, you have now this one minute of data that you need to manually generally go in and find, and then get that information somehow imported into the New York dataset.

Jason Lopez
If a company wants to embrace database automation, what should they look for in an automation provider?

Tobias Ternström
This is generally called database as a service. There is a couple of things. Do they run the databases that you need, which database engines are supported because of the fact that you don't just run Oracle or don't just run my SQL or Postgres, you probably run a few of them. Do they support the database engines that you need? I think that's first question. Second question is, do they support this, automation, if you will, where you need it to be run. So if you run in your own data center, maybe you run in a co-location facility and maybe you use public cloud. Can you use this across all of these or is it in just one of the environments? Another thing to keep an eye on is, you know, how flexible is the automation system is this database as a service. If you have requirements that you need to run a certain, maybe auditing plugin on your database server, or you need to maybe install your own extension that you built for the server, or you maybe you need to install certain hot fixes from a vendor, can you then install those? Do you have the control? And can you install those or is the Debas or the database automation system locked down such that you can't install these things. And the more of these caveats that you find, the more you need to run multiple systems, some databases will obviously be outside of automation, always when you run multiple systems, you have to make sure they now configure the same. Let's say I'm setting up high availability in one system does high availability in that system mean the same as high availability in another system. So it becomes, you know, harder to manage harder to make sure that you correctly live up to whatever business requirements you have.

Jason Lopez
Let’s close with your thoughts on careers in databases. How are all these changes and evolutions in DB technologies affecting people's career decisions? What would you tell people entering the field?

Tobias Ternström
It's a very good question. One answer might behave. If you haven't looked at open source, that's clearly a strong trend towards opensource databases. I would say, especially today, especially Postgre SQL, it's fashionable within databases also. So trends change at the same time. We all know. I think Cobal programmers make a lot of money today because they're far and few in between databases is such a large space. There's lots of activity. And the top five database engines. If you look dbengines.com, which I think is a fairly reasonable source is, you know, Oracle, my SQL SQL server Postgre SQL and MongoDB today. So if you're using one of those, you're probably in a pretty good spot. Now you may want to diversify probably a good idea to get a sense of if you're very much into one database engine. And especially if it's a proprietary one like Oracle or SQL server, I would definitely recommend venturing outside and learning a bit more about open source databases at the end of the day. I think as someone in the industry, it's still really knowing the details matter quite a bit, especially with databases.

Most of the time you would pick someone that knows a lot deeply about Oracle, for example, versus someone that knows a little about five different databases.

Jason Lopez
At one time, there was some talk about the DBA jobs going away. But more realistically it's probably changing. How do you think the DBA job will going to change in the next few years?

Tobias Ternström
Yeah, it's I don't, you know, DBA job isn't going away. First of all, it might say that definition of DBA is somewhat unclear or ambiguous, right? Because two DBAs next to each other may have very, very different jobs. So some DBA jobs are more leaning. It admin, you know, patching servers, configuring setting things up, and some DBA jobs lean very much towards developer. They're li writing, you know, SQL queries, they're optimizing queries, optimizing database, you know, physical structures and whatnot. And you know, some of them focus very much on, on, you know, database design, database architecture, things like this. So I would suspect that the ones leaning more, it admin will want to move higher up the stack and focus more on, you know, go closer and closer to the application. And how do we make sure that the application is, is highly available, highly performant. That would be the general expectation. I think.

Jason Lopez
Tobias Ternström is VP and GM of Databases at Nutanix.
This is the Tech Barometer Podcast, I'm Jason Lopez.
Tech Barometer is produced by the Forecast.
If you liked this interview with Tobias, check out his two part blog post, Modernize Your Database with Cloud Simplicity at nutanix.com/blog.
And for more stories on technology you'll find them at theforecastbynutanix.com.