EPISODE 312

[INTRODUCTION]

[0:00:00.3] JM: Expectations for mobile applications have gone steadily up since the iPhone was first released, but the choice for databases built from mobile apps has remained limited mostly to SQLite. RealmDB was created as a new option for mobile developers on iOS, Android, or other mobile platforms. Realm is not just a database; it is a database platform offering a variety of systems such authentication, sync, and an event framework.

In this interview with RealmDB VP of Engineering, Brian Munkholm, he explains why we need not only a new mobile dataset type, but an entire dataset platform. We talk about the business of mobile, IoT, cloud, and cross-platform. It's a wide range of conversation, but also deeply technical.

[SPONSOR MESSAGE]

[0:00:59.3] JM: As applications become more distributed and sophisticated, managing and maintaining them becomes more challenging. Running one of the world's largest PaaS platforms, Heroku understands the value of operations and the opportunity cost of developer time. Listen to the live podcast with Heroku Engineering on February 28th to learn about Heroku's metrics platform architecture and how it laid the foundation for auto scaling.

We'll talk about the importance of maintaining application health and best practices for monitoring application user experience. I love Heroku both as a user and because they are a sponsor of Software Engineering Daily, but I want to emphasize that I do use Heroku in my own projects. So, I really am a fan and I'm looking forward to doing some continued episodes about Heroku.

[INTERVIEW]

[0:02:00.3] JM: Brian Munkholm is the VP of engineering at Realm. Brian, welcome to Software Engineering Daily.

[0:02:05.5] BM: Well, thank you very much.

[0:02:07.4] JM: The expectations of mobile apps have been going up since mobile started, and at the same time, mobile apps often don't meet those expectations. How have those expectations about mobile apps changed and why is it difficult to meet them?

[0:02:23.9] BM: Yeah. It's amazing to think back on how quickly things have actually gone with the mobile apps, right? Actually, when we started in Realm and we were talking about doing a mobile database, a lot of people including the investors were sort of quite dubious whether that would be a good idea. Because even five years ago, mobile apps were not really a big thing yet, and so it's really been quite an amazing high speed race to getting some apps that work.

Now, the problem today is obviously that that development has only taken place over the last couple of years and really accelerated here the last few years, and it just seems like the technology is a bit immature and that's pretty obvious from the few years that has just taken to develop it. Therefore, there're a lot of complications in making apps. Some of the technologies are just not mature yet. There goes a lot of components into making that.

[0:03:19.8] JM: Yeah, it was been interesting to see the contrast between the pace of web development and even mobile web, versus the pace of native app development.

[0:03:29.8] BM: Yeah. Yeah, it's true that a lot of things has happened then and I think particularly for some of the smaller companies, or even IT developers, they are actually now competing with the biggest corporations in the world that has endless resources to put into app development, because now it's gone — Everybody uses the big apps from Google, and Facebook, and what not. Therefore, the expectations are just that things works, so they do pour in endless resources into app development and, obviously, make some good stuff, and that just becomes the baseline of how people expect apps to work. If they don't, then people get disappointed. It's just really, really how to do. Because it's so hard to do, it requires a lot of resources from people. Therefore, the best thing we think we can do is actually to try and make that a lot simpler. That's basically our mission.

[0:04:22.9] JM: Indeed. Indeed. Realm is trying to build the perfect back-end for mobile apps. What are the back-end challenges that are unique to a mobile application?

[0:04:34.3] BM: The unique thing is that you don't know if you're online. For many years, we've been dreaming about this situation where you're just always connected, and that's a promise we've heard for many years, but reality is just that that doesn't happen and it will never happen. You can go down to San Francisco, or you can go to a metro, or you can go into a supermarket and you won't have a connection sometimes. That's just also a lot of latency. The mobile networks weren't built for this originally.

The communication between a mobile app and a back-end is just very different from the expectations — And you have a desktop computer and a browser where you know that you were always connected. If you're not, nothing works. Now, the expectation is today with mobile apps, that things works whether you're connected or not.

[0:05:25.8] JM: Realm is this offline first database. There are many mobile apps like Facebook or Netflix that are online first. There's way too much data in these applications to keep everything on the phone. Explain what offline first database means and how that contrasts with these companies where you have to be online to use them.

[0:05:52.3] BM: Yeah. The offline first thing where, obviously — I was taking a plane ride, actually, from Copenhagen to San Francisco here not too long ago, and I was sitting in the plane and it came to my mind that, "Oh, I have some few tasks here. I need to remember when I land." I took up my phone, I started up my task app and I don't want to see what name it was, but I was just so disappointed that I couldn't add a single task in my task list. That was just stupid. Obviously, a task list is one of the first things where you really expect it to be available offline and then connect and then sync up, right? That just doesn't work.

When I go to some city, another country where I don't have a mobile data plan, because, first of all, that's expensive and say, "I won't have it immediately when I land." Then, I try to go into to my Google Mail to look up the address of the hotel where I'm driving. The first thing that happens is, obviously, I'm offline and even though I've been searching for that hotel before I left Copenhagen, I can't even get the address of my hotel. That just sucks.

Offline databases is also about having the things available that you used to have either when you're totally offline, or even if you are online, to give a superior user experience, because the moment that you have data available on your phone, even if you're online and you don't have to make that request to the server, you just get a much faster user experience.

The thing is that when we've been sitting with our desktop computers and accessing the web, we're sort of gotten used to delays. We've gotten used to things takes a while, but the expectation from users changes completely when you have things in your hand that you touch, because you expect it to be live. If you move your finger, you expect things to move. You can't wait two seconds before the thing moves around. The physical device just changes the expectations and the requirements, basically, for all apps to be real time.

Let's take an example, if you have a map, you zoom in or out on that map and you see all your hotels, or whatnot, or gas stations, and the minute you just zoom out, you don't want to wait two seconds for it to populate the things that are now available on your map. You want that available immediately.

[0:08:10.1] JM: Yeah. Let's talk some about the engineering solutions pre-Realm. What are the back-end solutions that people have been using on mobile apps for a long time? I know there's SQLite, there are some other solution. How well do these satisfy the demands of the modern mobile application?

[0:08:32.5] BM: Yeah, we found out that it didn't at all. That's obviously why we thought we use something better, because you say the SQLite and others. I can really only — Only SQLite comes to my mind as something that has been built for an embedded system anyways.

When we started out, it was — At that time we've seen a NoSQL database popping up every month. So much happened to server side. So much innovation on database has happened server side. Meantime, on mobile, nothing has really happened since SQLite 20 years ago.

The problem is not so much the technology and such, it's more also the APIs that you have available as a programmer. SQLite is great and it's a great technology, no question about it, but

it just doesn't provide — It wasn't built for modern software development and especially on mobile. We thought, "Okay. What will it look like if you really start it from fresh clean slate and you could really build what you want."

Because what happened was that, with SQLite, I don't know how many hundred ORMs has been built on top of that to sort of band aid what we'll need. A modern optic-oriented API that actually works like it was built for your language instead of a big impedance mismatch between your program, your language, your objects, and then just a lot of copies of data. Nothing has really been built from scratch to solve how to deal with data in programmer language. Not for mobile, at least.

Some of the differences in mobile is obviously that you have some limited space. You have some limited capacity as CPUs are a bit slower. Although, you can say these days, it's really, really powerful actually. There's still low power Android devices around in the world. Those also need to be served right.

Something that is built for small data, for being superefficient, and also for something that actually are quite efficient and fast. That was something we started out with. What we realized talking to users was that their biggest problems was the ease of use.

[0:10:40.4] JM: Right, speed is certainly something that's great to have, but there are lots of other problems to solve, and we're going to get to those. That's why Realm is a database framework. It is solving a lot of problems. The first thing is the speed though. What contributes to that speed increase? Why is the Realm benchmark able to be so much faster than the alternatives?

[0:11:06.1] BM: Yeah. Basically, because we started from scratch, understanding how modern processors works, how important it is that you're very, very efficient with your memory and the memory layout. If you look back, there are some cool graphs which obviously on this podcast is pretty difficult to show.

When you see the speed difference between — The performance differs between memory access and CPU. That could, at one point, maybe be one-to-one. These days, there's a huge,

huge speed difference between memory and CPUs. CPUs have gone up very much in performance, whereas memory hasn't. That big, big gap just means that the only thing you need to concentrate on is your memory layout. Forget anything about CPU performance and how many instructions you need to crunch through. You should only concentrate on memory layout and then you can gain a tremendous speed improvement.

That's why Realm was build up a column store, basically. Making sure that things are in columns so that your individual properties, say, you're searching for a name and you have different name, age, address, and so on. If you lay that out in memory like an old programmer language would do, you would continuously jump over memory if you are actually searching just for the name to be something.

That's where column databases — They put the names off each other, which means it's very simple to search through the name. it's very efficient, because the processor will ensure that when you pick up memory, the cache will be very efficient.

Some of those things, thinking about that from the ground up, making sure that speed was the number one thing when we started and continuously optimizing that, utilizing the new instructions there are in modern processors. That just gave us a very, very big head start.

The original raw core database in C++ just turned out to be a thousand times faster than SQLite, which was tremendous. The thing is that as engineers, that's just so fun to do. It's just so fun tweaking all that stuff. Then you realize that that's not everything.

[0:13:12.1] JM: The column orientation is pretty cool. It's fascinating, because my sense of the historical nature of the column database is this is something that started getting popular to deal with "big data", things like Cassandra. It's interesting to see that propagate to the consumer side of things. You actually want this in your phone's database too.

[0:13:36.8] BM: Yeah. The other thing when you then think it through and say, "Okay. Data really has — The size of data, the amount of memory you chunk through, that's the most important part." Then you think about, "Okay — " In other databases, you have to specify what

size is your integers, or your numbers. It could be a bit, it could be a 64-bit, but you have to specify it upfront. We just thought that is really, really not interesting.

First of all, you don't know in advance. Sometimes it changes, and then you have to migrate your data and it's awful. So why not just make an integer that can actually be up to 64-bit but could also — if the numbers are really small and they usually are, like age as some kind of limit, usually. Then, we made this compressible integers so that a bit takes up a bit if there's only a bit in it and just use the amount memory necessary, which just means that it's really tight and very fast into searching.

[0:14:38.3] JM: Let's go into the technical depths of the Realm mobile database, and then we will zoom out and talk about the framework and the higher level things that you're solving. It's core, the Realm mobile database is in C++. Every mobile platform runs that C++ core and connects to it using Objective-C, or Swift, or Java, or any other mobile platform that you're compatible with. Describe what the responsibilities of that C++ core are and how it interfaces with the other languages.

[0:15:10.4] BM: Yeah, obviously, we chose a C language to make sure that we could actually have one core that would work efficiently and then interface to other languages, because we knew from the beginning we are going to interface to a lot of languages, and C and C++ is basically the most efficient way to do that.

At its essence, the core database basically stores properties in compressible form. It's searchable without doing any kind of sort of a deep decompression. Then, it provides a number of other features around that. Obviously, the first one that we wanted to be a completely ACID compatible database and so it has transactions, read and write transactions, and then it also uses multi-version concurrency control to ensure that you can actually perform a lot of reads at the same time as you're doing write without having a huge memory overhead. That's superefficient and, yeah, works great.

That's basically it, right? Then, you have, obviously, a lot of features like encryption and stuff like that that we apply on top of that. That's the essence of it, yeah.

[0:16:16.4] JM: Right. Okay. Is there anything worth noting to — Any interesting innovations you need to make on covering how transactions work just in terms of the mobile database? Any interesting properties of read, write, delete, anything else?

[0:16:32.2] BM: One of the things that we really — We really want to try and make things easy, and easy to comprehend and work with. One of the things we did was that we said, "Okay. We only have one write transaction at a time." It will basically block another write transaction." It's very easy for you to reason about what happens. Nothing happens in parallel. We're escaping sync for this moment, but only one write at a time. That is simple. We have to have as many readers as possible. It's a multi-threaded design, obviously, and that just has to work well. Any mobile app is multi-threaded.

One of the biggest things about doing any kind of ACID completely trustful databases that we need to be able to trust when actual data is written to disc, or SSD, or whatever. I think, there, we were called by some surprises to figure out that some of the operating systems are not necessary behaving all that nice. It can be quite costly to actually commit things to disc.

When you do a sync to disc and ensure that data is actually written before you actually return from your write transaction, you have to do that in a way that where it is impossible for any — If your phone dies at any instruction, right? It just has to be completely crash safe. You can even argue —

[0:17:44.8] JM: There's a TEEN fault-tolerant.

[0:17:45.8] BM: Yeah. That means you just have to do two things just for one transaction, and that is really costly. That's something we chose to do to make it absolutely reliable. At the time between one sync and the other sync, you basically just move one bit to indicate that you have moved your — In multi-version concurrencies control you basically have two pointers, one for the new data, one for the old data. You basically move that finder with just one instruction that has to be completely atomic and either it execute or it don't.

[0:18:17.6] JM: Sure.

[0:18:19.6] BM: Those are some of the things, but working with that, you just discover that some operating systems don't necessarily perform actual sync when you actually request it.

[SPONSOR BREAK]

[0:18:37.4] JM: You are building a data-intensive application. Maybe it involves data visualization, a recommendation engine, or multiple data sources. These applications often require data warehousing, glue code, lots of iteration, and lots of frustration.

The Exaptive Studio is a rapid application development studio optimized for data projects. It minimizes the code required to build data-rich web applications and maximizes your time spent on your expertise. Go to exaptive.com/sedaily to get a free account today. That's exaptive.com/sedaily.

The Exaptive Studio provides a visual environment for using back end, algorithmic, and front-end component. Use the open source technologies you already use, but without having to modify the code, unless you want to, of course. Access a k-means clustering algorithm without knowing R, or use complex visualizations even if you don't know D3.

Spend your energy on the part that you know well and less time on the other stuff. Build faster and create better. Go to exaptive.com/sedaily for a free account. Thanks to Exaptive for being a new sponsor of Software Engineering Daily. It's a pleasure to have you onboard as a new sponsor.

[INTERVIEW CONTINUED]

[0:20:07.4] JM: Yeah, and so I want to get into the sync stuff, the higher level features, the stuff that is with the mobile device interacting with the cloud. Just as a side note, people who didn't understand some of the terms that Brian just mentioned, we did a couple of shows a while ago about Uber and their movement from — Well, movement between MySQL and PostgreS and back to MySQL. We went into depth of multi-version concurrency control, how that works, why it's mandatory for a database. Let's get into the high-level features of Realm.

Realm is a database framework as we mentioned. What does that mean? What were your goals with making a database framework?

[0:20:53.3] BM: Yeah. The main thing beside the role of storing data that matters — Let's get a bit back to the framework thing. Just before that, the important part of us was to make it seamless like a part of the language. Instead of having a normal database where you copy data in and copy it out again. What was really useful in our perception was just treat the programming language as an integral part — Or the database as an integral part of the programming language.

Stepping back a little bit here, that was some of the original ideas that you shouldn't be able to even see that it was a database. Obviously, then, the thing comes with how do you actually make transactions. It has to be a little transparent that you are actually storing data. But that's all of the thing that is very different with Realm is that the second you touch your properties, you are operating in the database itself. There's no copying a data out. You can ask us, "Is that a part of a framework, or just an integration into a language?" I don't know. That's probably a matter of definition.

It's, anyway, something that you can say that — The ORMs that have put on top of databases in the past has basically tried to solve that impedance mismatch between a database and objects. That is the first layer of your — I don't know if you want to call that a framework, but it's at least the first layer of integrating it into the language.

Now, the ORMs just have trouble doing things in a good and nice way, because they don't have a deep integration into the database. There is basically a band-aid on top of it. Because we can actually just make the database in any way we want, it's possible to make that integration really smooth. The thing about making apps today where you really want apps to be reactive, having events when changes happens to your data is very, very useful. That is something that is very simple to make when you can actually create those events when you want, because you own the database. On top of that obviously comes other features like syncing out and other components that people build on top of that, but that's another thing.

[0:23:00.1] JM: Let's talk about that sync. Realm has an automatic sync with the cloud. I know this is a complex distributed systems problem. What is the strategy for syncing with the cloud? I think along with this goes the — There's this term real time. It's a real time sync. Real time can mean a lot of different things, 'cause there's always going to be some amount of latency. It's like a kind of a question of how real time is it. Maybe you could just give it an outline for how you think about that automatic sync.

[0:23:33.3] BM: Yeah. Taking it back to the programming language, again, where the programmer is sitting and having this huge problem of how to deal with sync. One of the first things we heard when we language the database was it was probably one of the first issues that came up in GitHub, "What about sync? How do we deal with that?"

We were actually already working on that at that time, doing sync probably is a huge undertaking. It's a huge undertaking to make a mobile database. We've been — Five years working on that now, and doing sync is also a very, very huge part of it. It was built in and design from the beginning, so we knew where we were going with this. It was a bit difficult to not talk about it, but we wanted to give it the time that it takes to actually make it work properly.

I can say the main goal with sync was that it should be totally transparent. As a developer, you shouldn't worry about it. We basically want to completely remove that big peak over here in terms of programming effort, thus in dealing with the networking. It's like that fundamental idea that you just have objects, and objects is your data. It is connected to your database somehow. You don't need to deal with it.

These objects, instead of just being easy to share between threads, it should be just as easy to share those objects between any device, or a server, or another mobile device. It should be completely transparent. That was a big idea with this, or aim, that you don't see it as being transferred anywhere. It's just immediately available anywhere that you want it to be available. That was the design philosophy of it.

That's obviously a little difficult. You can see now, APIs today —

[0:25:13.5] JM: Say the least.

[0:25:15.6] BM: Today, if you actually want to enable sync — If you've just been using Realm for anything, the only different between syncing it and not syncing it is just setting up just a configuration in the beginning of your app. Then you won't be able to see the difference whether an object is updated on one thread in your app, or it's updated on another thread in another device, or on a server. You can't see the difference. You just get an event when the data is updated, and that's it.

[0:25:44.4] JM: People who may be working why is this real time sync so useful? I think about an application like Dropbox where you've just got files and you want to have them synced with the cloud. I remember when Dropbox came out, you hear Steve Jobs say, "Oh! Dropbox is the feature," and it turns out, actually, this is a really hard company to build, and Google can't do it, Apple can't do it, Dropbox did it. Dropbox did it really well. They continued to do it, and it seems like they're continuing to build an entire business off of basically syncing.

There are some canonical application patterns that look a lot like syncing, and you're trying to solve those. You want to be able to enable people to have real time communication, you want to make it really easy to build messaging applications. You want people to be able to know if a user is using their app, is that person online right now. How do you track user state? In sessions? Sessions are a hard problem. I've done some shows about Google. Google's got this streaming thing; Google Dataflow, and Apache Beam, and they use user session tracking as an example. This is actually a really hard problem, because when does a session begin and end? That's kind of subjective. Realm in the real time sync problem set has many applications that it could potentially solve. How does Realm enable these sets of — If we think from a high-level, from the top-down perspective; messaging, tracking user sessions. How does Realm enable these kinds of applications? In what design decisions does that translate to at a lower level?

[0:27:30.4] BM: Yeah. The thing is that as soon as you have basically an open connection — Technically, we're just using WebSockets underneath for the transportation. You basically have an open connection. That obviously has a lot of advantages in terms of not having to comparing to REST API where you're basically transmitting a lot of more data and having to open connections all the time. That's one part of it.

The other is that when you want to build those kind of applications, it should obviously be very, very easy. Having this object API where you just update and object and that object is just transmitted to a server that would populate that out to whoever else want to know about these objects, just makes it super simple to make those kind of apps. It's basically trivial.

We're having a feature coming out very soon that basically is Google Docs property. You can have TextStream that will work like Google docs. Not only will you have enable a messaging like you would normally do, let's say. Now, I write something, you might be able to see that I'm typing, but you can't see what I'm typing, because it's not really cooperative, it's just indicating something is happening in the other end. Maybe sooner we'll have this out so that you can actually see real time I'm typing. You have Google Docs in a field.

[0:28:48.7] JM: It's incredible. I've done a couple of shows with developers from Meteor, and Meteor, this JavaScript framework say — Looked at — What is their real time story for a while? This real time stuff is — The potential for applications that we could build, I think is really an untapped world. We're seeing the early days — We obviously saw the earliest days in terms of Google Docs, and that's really cool that you're now democratizing these sort of things. it's certainly fascinating that when you think about Google Docs, those made — What? 8 or 10 years ago, and we're kind of just not — Still feels like real timeness has not been totally democratized, and they see Uber — And you're in San Francisco with great connective and you're looking at an Uber map and it's still — Like the car is like sliding around in these weird ways and it's like not really working as you would expect a real time, "real time" system to work.

I've done shows with Uber and I talked to them about it, it's like their real time sync problem is tremendously hard, because you've got a driver with a smartphone that's got constant GPS updates. You've got a rider who is looking on their phone at the constant GPS updates. It's a lot of data, a lot of synchronization, flappy networks. We've got a long a way to go.

[0:30:09.2] BM: Yeah. I think we're just at the beginning of this. One of the reasons is that it is so damn hard, honestly, that we've spent years and years, many years on this. It's hard to think about. It seems so simple when you start out on it, "Oh! We'll just do like this and this." Before we have it working, you've spend several — Many years. The reality, today, if you're not fastly doing your apps, you're dead. You can't spend years doing apps.

You got to do what you can with the resources you have, and very, very few companies are able — First of all; able to do it and have the resources to do it. There's something we just concentrate on to make write. This is what we try to be good at. Make that available to developers in a way that just makes it trivial for them is sort of our goal. Talking about Google Docs and syncing, it's some of the fundamental technology and theories this has been built on is operational transform. It has relations to the developments that has been done there.

[0:31:15.9] JM: Operational transforms?

[0:31:17.1] BM: Yeah. That is basically what —

[0:31:18.7] JM: What is that?

[0:31:19.6] BM: Yeah, that's just the algorithm that Google Wave originally was built on. It's old theoretical —

[0:31:24.1] JM: Google Wave. Wow!

[0:31:25.6] BM: Yeah. That is also what Google Docs uses today. It is basically tracking operations and transforming them. It will be another show if we have to go into the details of that.

[0:31:34.9] JM: Oh, okay. All right.

[0:31:35.5] BM: But you can look it up, operational transform. That's basically how most of the collaborative editing works today. The problem with that was that it is mostly described in the lecture of the documents and the research papers, dealing with, actually texts. We had to extend this to deal with complex structured data. We've spent a bit of time on that.

[0:31:57.7] JM: There is also an emphasis on being able to interact with third party APIs within the realm documentation. I'm looking at it. I think of things like Stripe, or Twilio, the "API" economy, it's really — It's getting big, and these things make is to much easier to build

applications. This is just tremendous leverage for developers. Why does that effect the architecture of what your database framework is? Why is the third party API economy affect that?

[0:32:31.1] BM: Why it affects it?

[0:32:32.4] JM: Because, I mean, I think of a database is a very different — I don't think of a database as being at all related to third party APIs, but C mentioned of it in the documentation. I don't know. Maybe —

[0:32:45.3] BM: Yeah. Maybe I'm missing out on the point here, because obviously third party APIs in terms of Stripe and those — We're really not interacting with them in any way. What you're sort of trying to place is usually the API should build yourself, because enterprises that has legacy databases — Obviously, everybody has that today, unless you're a startup.

You have something build already, and you need to be able to integrate somehow to that part, because otherwise it's hard to get in there —

[0:33:17.8] JM: Okay. This is my next question.

[0:33:19.1] BM: Yeah, throw out all your existing databases and people look at you, "Yeah, can I trust that?" That's just horrendous. On the syncing side, just really look at Realm at least in HLE if you don't have any, or if you have existing data that you basically build a front-end to your mobiles. That just means that all that horrible REST APIs that you have out in your mobile, you can scrap that. You don't have to build it, first of all, both for Android and for iOS.

Now you can get rid of all that, and then you can just concentrate on a REST API from your back-end to the realm of mobile platform, which makes it much more simpler and you can know all those REST APIs will go much faster, and you can skip all that air handling that you usually need to deal with, when connections drop and come back on and all that stuff.

[0:34:03.4] JM: Can you talk more about the legacy integration problem? If I'm some old company, I've got a ton of data, I've got various databases, what am I doing to integrate with Realm?

[0:34:13.0] BM: Yeah, right now at the back-end side of things, we have a JavaScript API. It works completely similar to other APIs. This is roughly the same, you can say. If you look at our native API — Actually, we recently launched just the JavaScript framework as a separate product, so you can even use it just for pure JavaScript and the back-end if you want to, in Node.

Then, that part of it works with objects just as you're used to within your other product from us. The simplicity of that is working with objects, getting events whenever something happens. Now, that works pretty well, because when you're sitting on the back-end there and you can basically just observe any kind of object on any device you have.

Let's say you have one Realm file that could be sharing state to, let's say, all your devices. Basically, you just want to share some common state. Then, you would be changing your object in your Node app and that object will just be transmitted to all your devices. The thing is that then you can — In Node, you could basically interact with all your back-end stuff where you want it. Let's say some of that data gets updated, you would just update your object in Realm.

To make that easier for your connectors to specific databases and also some generic API so that it will be simple for you to integrate with your other databases and sync up data whatever legacy database you have.

[SPONSOR BREAK]

[0:35:44.6] JM: Indeed Prime flips the typical model of job search and makes it easy to apply to multiple jobs and get multiple offers. Indeed Prime simplifies your job search and helps you land that ideal software engineering position. Candidates get immediate exposure to the best tech companies with just one simple application to Indeed Prime.

Companies on Indeed Prime's exclusive platform will message candidates with salary and equity upfront. If you're an engineer, you just get messaged by these companies and the average software developer gets five employer contacts and an average salary offer of $125,000. If you're an average software developer on this platform, you will get five contacts and that average salary offer of $125,000.

Indeed Prime is a 100% free for candidates. There are no strings attached, and you get a signing bonus when you're hired. You get $2,000 to say thanks for using Indeed Prime. But if you are a Software Engineering Daily listener, you can sign up with indeed.com/sedaily, you can go to that URL, and you will get $5,000 instead. If you go to indeed.com/sedaily, it would support Software Engineering Daily and you would be able to be eligible for that $5,000 bonus instead of the normal $2,000 bonus on Indeed Prime.

Thanks to Indeed Prime for being a new sponsor of Software Engineering Daily and for representing a new way to get hired as an engineer and have a little more leverage, a little more optionality, and a little more ease of use.

[INTERVIEW CONTINUED]

[0:37:32.4] JM: I want to zoom out and talk about the business model and the developer ecosystem, 'cause RealmDB is an open source project mostly, I think. Can you describe the relationship between the open source community and the business model and how that all works?

[0:37:50.0] BM: Yeah, sure. Originally, when we thought about what to do in this space, first of all, we all like open source, so we actually just wanted to have that. Secondly, from a business perspective, when you come out with something new and you need some kind of adoption, one of the things that really takes developers is open source. Particular things like a database, you want to know that it works well. You want to be able to actually continue using it even if that company that created it suddenly should disappear. That's all the big advantages we have with open source.

Then, the other thing was that at the time, SQLite is free. Apple provide the core data for free. All the alternatives are free, so it's pretty difficult to come out with a paid product in that space. Basically, we really want to make something that developers love and we want to broaden that out to anyone. That's only one way, that is to give it away for free. Then you have to build your business around something else obviously.

For us, this synchronization framework that enables us to solve very, very, very large problem that a lot of companies are fighting with and that we get feedback on is something they spend years and years on without even solving it properly. That's obviously something that has a huge value to corporations.

On the other side, we're very, very involved in the open source community and in developers and the hundreds of thousands of developers that are using our products. We don't want to just not give anything to them. That's why we also actually provide the mobile platform with the syncing and all that for free. Just reserving the most enterprise features for payment from the corporations.

[0:39:30.9] JM: What are the challenges to building a successful company around developer tools? Because we saw this RethinkDB situation recently. RethinkDB, very cool product. I had Slava of RethinkDB on the show. We talked about RethinkDB. There are inherent challenges to building developer tools companies especially this precarious line where you're walking where — A lot of the times, the software — You think about Windows 95, what a great business, because it's not open source free operating system. It's proprietary software that they sell for $100 and print on a 12 cent CD. It's a lot different than the economics of an open source database solution. Tell me about the difficulties of building a developer tools business.

[0:40:24.5] BM: Yeah. I don't know if that many have been successful with it yet, right? That's something we are trying to learn and obviously learned from all the others that hasn't been successful with it. I think, from the start on, you need to have something that differentiates. You need to have some — Give some real value to people, and then you can argue that it is very, very difficult for people.

I'm a little surprised actually — As developers, we are spoiled with developer tools, honestly, these days. We get everything for free, and there are hundreds of many years of work behind all that stuff. It's amazing. That obviously also generates a lot of developers, because it's pretty easy to get started.

Yeah, it is a bit of problem to actually find a business model that works, because you got to give things away for free. It has to be open source. Unless it has the right open source licenses and very permissive, people don't want to use it. You really have to give a lot of stuff away for free. I don't know if there's a silver bullet to that question. It's something that we're exploring as well. Basically, we say, "Here's a cool stuff. Here's technology." We'll give it away for free too.

[0:41:35.8] JM: What is a compelling upsell I see in the modern developer tools environment is where you offer a cloud. If you have some kind of cloud offering where it's like, "Hey, we've got this great open source thing," and then there's a cloud solution that will save you 80% of set up and configuration and stuff, or you can just — You can create this conglomeration of stuff on top of the open source solution. I think Databricks is probably going to do something similar to this with Spark. You just make it easier to use.

Like you said, there's no silver bullet. There's probably a lot of interesting directions where you can go, but it's, I guess, a precarious world to play in.

[0:42:15.6] BM: Yeah. Yeah, it is. I also think we are exploring to — We have a pretty good idea on where we want to take this. Hopefully, it will work out. I think it will. I think the bottom line, when you're talking to companies, they have serious problems, they are in need of filling up. App development is not something they're fully under control.

There's a troublesome area, because — As we talked about beginning, the expectations are so, so high. Even for intern labs. People don't want to deal with that junk if it doesn't work as good as everything else. Expectations are really high, and if you're providing something that the corporation would normally really spend a lot of time on, they're willing to pay as well. They can see they will save a few man-years on that. I think that's worth it.

[0:43:01.6] JM: For sure. When you think about the problems you're solving as a mobile database, is this going to be the database for IoT, and drones, and — Does it have applications for cars? What are your thoughts on where this goes in the future when mobile doesn't just mean the piece of glass in your pocket, but it also means your wearable and all kinds of other things?

[0:43:29.1] BM: Yeah. I don't see an end to where this could go, honestly, because exactly all those new applications have the same requirements like — Maybe the space requirements and CPU requirements are not that low anymore. There's quite an abundance of CPU power for almost no cost, even in small IoT devices.

Sure, there is no app without data. We always need to work with data, and that's what we're trying to make very, very simple. I'd like to say virtual reality, or things like that. You have this thing about — You need quick update speed, otherwise you get all DC and things like that. You access to data just has to be fast.

Most importantly, all that development goes on precedent pace. That is mind-boggling. That just means that if you can save development time, that is really, really crucial for your business. That's what we are trying to do. Basically, make sure that people can save a lot of time and also make better apps. I just spoke to a customer the other day and then asked, "What is it really that you've sort of gotten out of this?" That was a customer who've been very reluctant to actually get started on Realm, because they were quite happy with what they had.

Then, when they did, they said, "Wow! We just saved ourselves tons of testing time, because now we don't have all these different objects that are out of sync with each other, and we have all these weird edge cases with our testing where it doesn't work when we go from one list view down to some sub-view and go back again and something is not updated, because now we just have one state. It's always — We know what it is. If it's changing, we get an event."

That just saved them so much time in the end. They were just very, very happy about that particular corner part of it. The bottom line I think is developer productivity is very, very important for most businesses.

[0:45:27.3] JM: Yeah, fascinating. What are the problems in mobile databases that you feel like you haven't been able to tackle yet?

[0:45:37.2] BM: I don't think we have any particular big problems. There are tons of things that we would like to do. There are tons of areas where — You can look at the GitHub issue list, right? That's all our development there. It's also public and in the open. We have a long backlog of things we want to do even better. In particular, probably in some of the search areas and geospatial, or indexing, and things like that where that's something that a lot of apps are using. Not too many, but most are getting that direction to some extent. Some of those areas where we can make it much easier as well.

[0:46:11.7] JM: These are like situations where people right now have to do polyglot sort of stuff where they have the, maybe, InfluxDB for the time series component and they use elastic search for their search component, and it's be cooler to have that stuff bundled into single database.

[0:46:30.1] BM: Yeah, it sure would. There's also the component side. We have explored that to some extent. Mostly, that has been from the community, basically building out maps component. You just plug in a map and it's super easy to — You have it already backed up by your data.

[0:46:45.9] JM: Oh, cool.

[0:46:48.5] BM: A search functionality. You just start typing and you have all your completions and all that stuff. That's just plugging in a component and then it works. I think that's an area where we could do much more, honestly. Although some good, cool parts have been done, but we have been very focused on getting the mobile platform ready here lately.

[0:47:07.7] JM: Okay. Brian, I want to thank you for coming to the show. This has been a really good show. Many people had requested RealmDB, so I'm really happy to have had you on. I know — I guess we mostly skated along the high-level, but I know there's a lot of interesting technical details and people can go and checkout documentation, or some good videos and stuff. Thanks for coming on the show.

[0:47:28.1] BM: Yeah, it was a great pleasure to be here. Thank you for taking me on the show.

[END OF INTERVIEW]

[0:47:35.9] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily.

Thanks again Symphono.

[END]