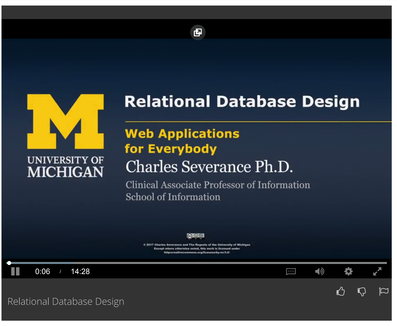
Lecture Materials

Assignment

Bonus Materials



Relational Database Design

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:08
So welcome to our second major section of databases. Previous lecture we learned about how to get work done in a single table. And now, we're going to start connecting and that's where the word relational starts to matter. And so these are the complex data models and relationships. And they're just techniques, how we represent these things in a way that allows databases to do that sort of magical high performance efficient data gathering that we need to do. So this is where we're going to, you know, rubber meets the road and really learn how to build. So this is where we switch from how to talk SQL to how to design databases. And, as I've said before, I learn databases kind of late in life. And I took databases in grad school and that was in the 1980s and databases, relational databases, weren't all that good. And so unlike databases, they're dumb, right? And I would just write loops that read stuff but now, I mean, it was about 2000 when I had to work on a web application as my professional job for the first time in my life. And then, they're like, we're going to do some database design. I'm like, I don't know anything about that. So I went in a room with a bunch of folks and I picked it up really quick. I think it's really a beautiful thing. I mean, you draw this picture, you draw these lines and you're sort of like creating a network of data for your application. And so the basics of database design are, I think, easy to understand. And I think you'll have a good sense of them after the course of these next few lectures. You can always learn more, right? And what people are like, why don't you teach me advanced databases? And it's like, the basics are easy to understand and the only time you really learn the advanced is if you face a problem and you ask someone or you look on stack overflow. So, it's a beautiful art form. You're going to actually be in great shape to participate in database designs. And you'll be able design moderately complex databases and not make mistakes. But then, there's always additional tricks that you can do. So this is a picture of a data model that I've built. It's kind of similar to something we are going to talk about in this lecture. It's a learning management system and it's a learning tool that I actually use for the auto graders for this class. This is the data model of the software that supports the auto graders of this class. And ultimately, each of these little boxes is a table. Each of the little lines is, so you know, that's a table, that's a table, that's a table, table, table. And then we got some keys and we got little data bits in it. And then, what's cool is we got these lines and eventually we will understand what these little characters mean. This means many, this means one, etc, etc. This is a Many to Many table. This is the last part, this is like two hours from now you'll see others. But ultimately, what we're doing is we're creating a picture and it's a network. And then the relational bit. And the thing that makes it super scorchingly fast is all the thinking that you do about these things, about these lines. And they would seem like they're just nerdy things but this is the magic of relational databases. And actually, in a couple of hours you will understand virtually everything on this table. And that means sort of when you go into a job place and you see a picture like this on the wall, because they can get kind of complex. This is another open source project that I work on called Sakai. It's an open source learning management system. And this is just like about one quarter of its data model. But what happens is when you're an application developer, you have to know the data model. You have to create the data model. And understanding the data model is the key to writing performant code, because anybody can write crappy code. But in a web application, if it's going to be successful at all, it's got to be relatively performant. So basically, the idea is as you take a user interface of some application, you don't build the application from the data model, you build the data model from the application. And then you kind of look at the application and say, "What kind of chunks of data do we have here? " And we cannot put it all in one table because that will be slow. So we are going to put it in a few tables. So what parts best go into what table etc. etc. etc. And so, the idea of building a data model from a mockup. And so let's pretend that we just started a company, right? And our idea for this company, and this is a brilliant idea, I mean, we really should actually start a company on this. Our idea for this company is that we think that in the future people will buy music based on tracks, not based on albums, right? An album, for those of you don't never seen an album, is a set of musical tracks that you buy all at once for like nine dollars. Or we could sell a track for a dollar, wouldn't people like that? So that's our innovation that we're going to do. So we've hired a graphic artist and the graphic artist has come back with this mockup, says: "You know, we're going to do really well if we build an application that can do this for people." So this is the mockup. Now, the first thing we got to do, as developers, is not necessarily argue with the mockup. We can't say: "Do you realize that this data model is not normalized properly?" And so you got to change the application because if you change it. Assume this is what we want it to look like this. This is like, to a data modeling person, this is like: "Oh wow, look at all the nulls. It's so scary. Look at the vertical duplication of strings. That can't happen." Our job is to build a data model that meets the users needs, not tell the user that just because they have the word paranoid more than one time in a column on this user interface, that they have violated the rules of normalization and they should come up with a different user interface. So the question is then, how to go from this user interface? We've got tracks, length, artist, album, genre rating and count. So we sort of have our columns. Now, you could imagine that one way to do this would be to make one table called "the music table" and put all these columns in. But what you would find is that, and you might have done this in your own life where you tried to put a spreadsheet of all your music tracks and you'd find you're typing this in over and over and your brain is going like something's wrong here. And then it turns out you made a typo, you're cutting and pasting and you made a typo. Then you go fix it a bunch of places. So the idea is, in relational, is this shouldn't be in one big table. This album information, it needs to be its own table and we are only going to put the word paranoid in one, cut to the chase, and then what we're going to do is we're just going to take these tracks and put a mark in. So paranoid will be like album number seven and then P and then we put like seven, seven, seven, seven, seven, seven, seven, seven. This would make a lot more sense later, okay? But that's the idea that we look at this as data model, people. And we don't tell you to change. We don't tell the designer to change their look and feel, we are going to compensate on our end. So we look at all the columns because we do have to represent all the columns. We know they're going to be more than one table, but we want to group them in a sane way, right? And so we have to sort of figure out, how, what tables, what are the kind of core objects that are being represented here? It's not just one object for each column, that's not good either and we have to kind of find a balance between the things that belong together and the things that belong separate and need to be linked back and forth. And once we found that balance is where we get that sort of maximum efficiency. So you can pause. You're going to get cup coffee because we're going to sit down in this conference room on a whiteboard and we're going to figure out what the data model of this thing is. And we're going to debate. We're going to figure it out. So these are the columns we've got to make. So the first conversation that we have to have usually is, what's the first table? Because we're going to put this in multiple tables, there's going to be more than one, we know that. So in building a data model you often ask, what is the first table? And usually you think about it is like, what is the core purpose of this application? You might decide that if we're making Twitter, tweets might be the core thing. If you're making a learning management system, courses might be the core thing. If you're making a email system, users might be the core thing. So now we have to debate on our little thing. What is the core thing? Because users is not our core thing. Because we're building a little tool that everyone has separately. And then after we build this we're going to invent a phone that has like, you can touch the screen and stuff. It's going to be big. We're going to do really fine here. But it's one user. So the users, each person has its own phone and we're writing an application. So users is not our core thing. We don't have a column named users, so doing a column named email. So we can argue, but if you look at this I'm just going to say that fundamental thing, what does what is the aspect of every row? And that is, I think it's kind of a track. The track as the core thing. So let's say our first table is track, okay? And now what we're going to do is, we're going to look at all of these things and we're going to say, which of these things are just like an aspect of every track? Different for every track and which of these things are the same across multiple tracks? So that's the idea. And it really comes down to vertical duplication, right? So that's, I mean that's the clue. If you see something vertically duplicated that's a string, it's a problem. Vertical duplication with numbers turns out to be okay. So numbers are easy. So here we have the length. Let's just go with this, we go with the length. So we got the track. We've got to take the title because that's different. So we'll take the title. Title goes over here. So we got that figured out. The length, the length is just a number. The fact that two things are four minutes. Numbers are cheap and easy to store. So we don't worry about the fact that something is 300 seconds and another thing is also 300 seconds. So we're going to put that over here. The length this fine. This, don't do that. So length is taken cared of, but this has vertical duplication. So we're not going to do that. This has vertical duplication we're going to put that somewhere else. This has vertical duplication, we're going to put that somewhere else. But this rating, this is really like a 0-5 number, so that's good. So we're going to put rating over here, right? So we're done that. And this is the number of times we played and this is the count. So we've kind of made our first table. Okay. You've got this taken cared of, this taken cared of, this taken cared of and this taken cared of. So we've got track, we've got the title, we've got the length, we've got the rating and account. That's our first table. And we got this checked off, we got that checked off, we got that checked off and that checked off. So then, we're now looking at these ones that have vertical duplication and that's where you basically are going to make another table. So let's kind of figure out what's going on here. Let's let's make a table for albums. So every album, and we're going to have a title for each album. And then every track is connected to an album. So we're going to put it where it belongs to here. So this is saying that every track belongs to an album. So we know we've got that taken cared of. Now actually, that's not a album, sorry. This is the album. This is the artist. So now, what we're going to say is, well, okay so every track belongs to an album and then every album belongs to an artist. So we're going to make up an artist table here. Like AC/DC is the artist. And we're going to have the artist name AC/DC and then we're to say every album belongs to an artist. So albums belong to artists, tracks belong to albums, album belong to artists. We can argue. You're probably going like, it's not quite like that and you're probably right. We're going to have to simplify this but, so to do the data mining. So we've got this taken cared of, we got that taken cared of and the only thing we got left to take care of is the genre. Now the question is, where do we want connect the genre? Right? Do we want to connect the genre? Does a artist belong to a genre? Does the album belong to a genre or does a track belong to a genre? And this is one of the things you do in data modeling and of course this one is pretty simple, but this is the idea. But it turns out that wherever we connect genre, is going to make a difference in our application, okay? It's going to make a difference in our application. Because if we connect genre to an artist, we do it here, then that means that every AC/DC thing has got to be rock. Does AC/DC ever do a Christmas album? Has AC/DC ever do folk? It's like well, that might be a little limiting if we do that. So we're not going to put that there. We are not going to do that. Well, how about an album, right? So it's, Who Made Who. Do we want to do it here? Is this where you want to put it? And so that would mean that every track on the Who Made Who album would have to be rock. And we're like that closer, but what if AC/DC did a greatest hits and some albums were folk and some albums were rock? And I was like, yeah, that's okay, that's a bad idea because we want flexibility. We want to make it then. So the answer, is we're going to put genre right here. And we're going make every album has a genre. Now we got it, okay? So there's our picture. Four tables. One, two, three relationships. Well, what a mess. What a mess. Oh, look how beautiful that is. So you have instant beautification, right? And so this is what we achieve. Now, we don't worry too much about nerdy terminology, we're just trying to break the columns into things that belong together and establish sort of some kind of a meaningful human understandable relationship with them. And we're going to quickly turn this into sort of code, but now it's not code. This is sort of a philosophical concept and away we go. So up next, we're going to talk about how to map this picture that we just drew into a database. Conventions on columns and database features that let that all fit together.

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?