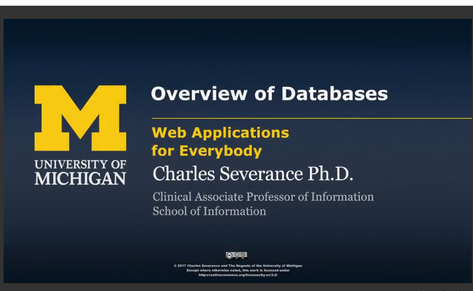
Lecture Materials

Bonus Materials

Assignment



Overview of Databases

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

Interactive Transcript

Search Transcript

English

0:08
So welcome to our lecture on relational databases. SQL is a beautiful programming language. I sure hope that before you start this lecture you already know how to program, because once you learn SQL, you might not want to learn any other programming language. And unfortunately you can't just do everything in SQL. I think that's kind of unfortunate. But I want to take you back a little bit in time. The notion of storing data. The first computers computed, right? They computed trajectories and weather, and computations are sort of like inside the computer. But when you start thinking things like banking and things like that, the data doesn't fit in the computer. You have to have some way of storing larger amounts of data than the computer can hold within itself. And so there was a time, you may not believe this, before we had disk drives. And so we had tape drives, now some of you were so young you don't even know what a tape is. But we used to have these things called cassette tapes and you would play your music. And if you wanted to go to another song, you'd have to fast forward and have to spin for a while. You couldn't just hit the next button and immediately go. That's because music players have random access memory, but tapes did not. Meaning that you had to physically move them. You could store a lot of data on them, but you had to physically find the right place. And so imagine the problem that you're facing is that you're bank. And you got a tape drive, which is sequential media, that you gotta read one thing at a time, just a loop, and you have all of the accounts and all of the current balances on those accounts at the end of yesterday. And then today people put money in and took money out, and then we need to come up with the new account balance at the end of the day. And literally, if you go back to the 60s and the 70s, this was what happened. Your bank balance might only be updated once per day, because we couldn't store all this stuff, and we didn't have spinning disk drives that do this. So, here's a really cool algorithm that we used back then called Sequential Master Update. And the way it worked was you would have all of your records sorted by accounts. So the lowest account number is first, count one, two, three, four, five, but they're in sequence, you can't just bounce around in here. And then you would sort, 1, 2, 3, 4, 5 would be the accounts and then account 3 and account 5 like account 3 is -100 and 5 is +100, right? So how is it that you can read this data and then produce a new data, because you can't rewrite this in place, because as soon as you rewrite this, you start to destroy its data. So this was the trick, we call this sequential master update, you can go read up on this on Wikipedia. And what you would do is you write a program that would read the first record, record 1, and it would read the first transaction and that would be 3. And it would say, well we haven't got to 3 yet, so 1 just gets copied, so we put 1 on here. Then we would read 2, and we compare it, we still have 3, and then we copy 2 to the output. And then we would read 3, and now 3 we would be able to combine the subtraction and put the new balance for 3 up.

3:11
And then we would read to the next file, 5, we get number 5, account 5, 4 would get copied, 5 would be brought in and modified and then a new 5 would be changed. And you can kind of see how this thing you could have thousands and thousands of transactions and thousands and thousands of records all in order. And at the end of the run you would have all the new balances on a new tape.

3:34
And then what you do as the next day is you would take this tape and you would make it the old tape. Actually what you often did is had like five tapes, so you rotated last week's tape, last month's tape, we used those as backups. But this notion that we had to take time to work through data sequentially. And if you've written Python programs in any of my Python classes, you could say go, and like one second later, unless you ran some of the later stuff, like the scraping or the email stuff that might take days to run. And this took a long time, because you physically had to move a tape across. And so this might take two hours at night or three hours at night and this whole concept of sequential master update. But the problem that we were solving was we needed to have permanent storage that lasted longer and was much larger than the storage inside of computers.

4:22
But of course that was only for a while. Ultimately what we ended up with was disc drives. And a key thing to a disc drive, if you tear one apart, and eventually we're going to have SSDs, which are even more random. But the idea of a disc drive is it's always spinning. And if you have a disc drive on your computer, you can kind of hear this whirring noise, and you had this head that moved in and out. And so instead of having to read all of the data sequentially, what you could do is if you needed to read this piece of data right here, you would move the head to the right location. And then you would wait until it came underneath the head and then you'd read the data. So if you think about the disc drives, they're often like 7,200 revolutions per minute.

5:05
And there is two millisecond access time and that's how fast this head could move back and forth. This goes around 7,200 times per second. It doesn't take much to calculate on average to get from one piece of data read here, to a different piece of data read here. It is the time it takes to move the head and the time it takes to wait for it. And with the speed, the revolutions, and the speed of the head, you could kind of calculate that.

5:34
And so databases came out of the notion that, wow, we didn't have to wait until 10 o'clock at night to update the data. We could store an account record here, an account record here, an account record here, and 1,000 a second we could go read and write an account record.

5:50
And the question of course wasn't just how to do it, because there are simple ways, but you'd still have to read all the data. And then read all the data, around and around and around and around and around, and it would still take like an hour to read all the data. So you think to yourself, is there a way to sort of skip ahead and know every record? We have a little table of contents that's right here. And it's a little thing, so it says record one is here, record two is right there, record three is here, record four is here, record five is there, right? And so I can say, I don't have to read all of them, I can just say I want number five, I look up in the table of contents and I jump over there. That's called an index. It's like breadcrumbs, a way for you to go back where you were. And so instead when you write the records, you'd write a little tiny thing about where they're at as a shortcut to get to them. And you had to read the whole index, but the index was generally small compared to the amount of data that was being indexed. And we'll talk about that as we build these things, what gets indexed, what doesn't get indexed, what rows and columns? But this was all about how to make the best use of this amazing new technology to make it so that you could check an updated bank balance within a second rather than within a day.

7:01
So, relational databases are a technique. So when we're figuring out how to best use this random access storage, we're computer scientists, we would argue, we would say, I have an idea, sort of like the picture I just drew for you. Another one says, I have a different idea. And so there's in the 60s and the 70s a debate as to the best way to make maximum use. And there was things called index sequential and network databases, etc., etc., etc. And out of that debate came a standard, a standard way of thinking about it. And it's really quite ironic that the relational databases is a rather theoretical, not practical thing. And I'm kind of a computer scientist and so it always appeals to me to do basic simple things. And I know when I first saw my first relational database I'm like this is way too fancy for me, this will never work. Now give me something crude and practical, and I can just sort of code and do the thing I want to do. And those first relational databases were pretty terrible. But then as they got better, I'm like, wow, this is so magical, and I don't even want to begin to know how this magical stuff works. And so that's what we get, there's a powerful underlying mathematical theory that makes relational databases work. Now, our view onto relational databases is a language. And so instead of you knowing how that data is stored on the hard drive, you're going to talk to a very complex and powerful piece of software called a database server. So MySQL is this database server, and it knows where all that stuff is at and it's got indexes, hundreds of millions of hours of computer science when it's writing MySQL or Oracle or some other database system. But what we did is we end up with this really simple, but beautiful, elegant way to say, hey this is the data I want. You are a genius, you go figure out how to get it to me and get it back to me as fast as possible. That is SQL, structured query language. SQL is a interesting bit of computer technology and I've got a video interview that I did with Elizabeth Fong of the National Institute of Standards and Technology. I encourage you go watch the video, but the short summary of the video is that while the US Government, in the form of NIST, can take some credit for SQL, what Liz will tell you is that the vendors were sort of fighting amongst themselves, and they were fighting with the government. And with they were trying to do is, each of them had their favorite idea, and they're telling the government you should adopt our favorite idea, because those other company's favorite ideas are really bad. Another company's saying, no, no, no, their favorite idea is bad, you should adopt our idea. And the government, who was buying lots of technology basically said, look we are not going to pick vendor A over vendor B. What we're going to do is you're all going to get into a room and you're going to agree on a way for us to talk to all three pieces of software. We're not going to tell you what that's going to be, but until you come out of that room with one agreement, we're going to not buy any of your products. And so that's how SQL happened. The industry came together, as Liz will say in the video, she will say that it was the perfect time in that it wasn't too early, meaning that all the vendors had good ideas about what they thought was good database. And it wasn't too late, meaning none of the vendors had sort of by hook or crook become dominant. And so SQL is this magical thing. And again, I told you before, I just love this language. And it came out of this really interesting [LAUGH] consensus process that was convened by the US National Institute of Standards and Technology, but not created by the National Institute of Technology and Standards, NIST. So SQL is this abstraction. It wraps complex software, lots of storage in a very elegant thing. And some of you maybe have used Microsoft Access or something, and you've even typed a select statement or an update statement. And you didn't even realize that you'd learned a programming language and like I said it's one of the more elegant ones. And what it does is it basically says, the things we do in databases are we put stuff in, we delete stuff, we update stuff, and we read it. CRUD, create, read, update, delete, it's in order so it sounds better, create, read, update, delete. We don't call it read, we call it select, but CRUD is the idea of the four basic operations to databases. And SQL said, look, let's make these four operations that are so common and done all the time, super simple.

11:24
Nothing about of how it's going to be stored, nothing about where it's stored, just what do I want done? Delete that row, get rid of it, I don't know how, just get rid of it. And so, if you read database documentation, because of this background that was sort of a mathematical theory that underpinned all of SQL, you'll read documentation that it's like there's two ways of describing it. One is like what we nerds describe it as, like it's the thing with rows and columns and tables. And then the math people would say, well, it's not exactly rows and columns and tables, it's relations and tuples and attributes and connections between relations and tuples and attributes. And while I'm making fun of the mathematicians, the mathematicians are the ones that make it go really fast, right? We programmers it's like, that's slow, now it's fast, thanks for the math, right, that's how it kind of worked out. I mean I love the math, even though, whatever. So what I'm saying is, if you're reading documentation, you'll see me talk about rows and columns and tables, because I'm a nerd, that's the easy, every person's view of this, rows, columns, and tables. But if you read stuff and you see relations and tuples and attributes,

12:35
don't feel bad, don't feel like somehow you don't know whats going on.

12:40
And so you can read this sort of more highfalutin version of the language of database and understand it really is fancy ways of talking about rows and columns. And I like to think of a database and [LAUGH] some of you may have already used something like Access that really gives a very spreadsheet-like view on a database, right? Rows and columns, tables are kind of like tabs, right, you got these tabs at the bottom of the spreadsheet, we got rows and columns. Now the one thing that's interesting is one of the things that we've kind of come as convention, and spreadsheet software even kind of knows about this, is we use the first row as actually not data. This is data and this first row is metadata.

13:22
It's not much metadata, but it's not real data, it's metadata. It's data about the data. First column's Title, second column is Rating. In database we're going to do this a lot, we're going to call is schema.

13:35
And we're going to be very, very precise about what each column is, how big it can be, what kind of data it can put in, what's the character set of these things. And so we actually write very complex statements about what's in a column. In a database we'd say, this is an integer that can be no more than 2 billion. So less than, it's an integer with a sign that's less than or equal to 2 billion, and that's what we'll say Rating is. And so we put a lot of detail in, but again it's still kind of the same thing as the first row of a spreadsheet being the metadata about what's in each of the other rows.

14:09
The database system that we're going to use in this class is free, MySQL. And there's also a version, kind of a fork of MySQL called MyIDB, that's also roughly equivalent for this class. But it's not the only one, and there are other really good database servers. Oracle is probably the largest commercial enterprise scalable database that lots of companies and corporations use. MySQL is kind of like the database used for lots of the web. Then there is SqlServer, which is Microsoft, if you're a Microsoft shop. Everything we're going to talk to, just about everything will be just fine if you're using Microsoft. And Access is kind of a lite tool that helps you write SQL often. I mean there's lots of other databases like Postgres is another very popular open source, SQLite, HSQL, if you took my Python classes we played with SQLite in those classes. But now because we're working on building web applications, we've switched to using the MySQL database server.

15:03
So, that gives you sort of a bit of a history and a context about SQL. And then coming up we're going to start talking about typing some SQL and just doing some work.

Downloads

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

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