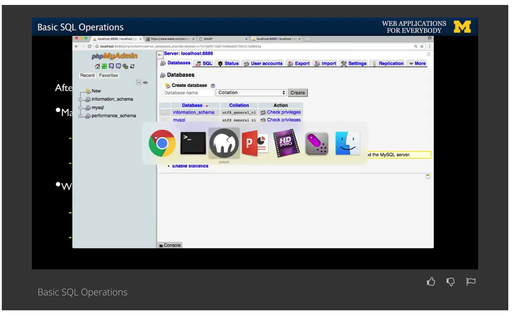
Lecture Materials

Overview of Databases 15 min
Basic SQL Operations 21 min
Data Types in SQL 8 min
Database Keys and Indexes 13 min

Bonus Materials

Assignment



Basic SQL Operations

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

Interactive Transcript

Search Transcript English

0:08
So after all that long-winded introduction, now it's time to actually start using SQL, but before we do, I want to give you a picture of where we're at. So this is the request-response cycle. We have a browser over here on the side. You are way over here, that's you on this side. You click on a link, the browser goes and makes a network request using HTTP to a web server, and then it maybe pulls a file in and maybe that file is PHP code. The PHP code starts to run, but then has to talk to the database server so it sends SQL across, that's what we're talking about now. And then this SQL server, MySQL software, is really smart about where to find all this stuff, and it sends the data back to PHP and the PHP loops through the data it got from MySQL server and write some more HTML, which goes back as the response. This is the request-response cycle here, and it parses this response, fills up this thing called the document object model. And here we are, sitting here and we see the new screen. And so that's the request-response cycle, but for today, for this lecture, we're really here. As a matter of fact, we're going to learn the language that's flowing back and forth across the network. Now, often these are on three separate servers, one, two, three, but on your laptop and on my laptop, they'll always just be one server. But it doesn't matter, we're still sending commands over. So we are learning in this next couple of lectures, the syntax of the SQL language that goes into the database server, gives it instructions and is used then to return the data from the database server. So we'll zoom in now. We're going to do two basic things. We have this piece of software, the database server and I keep saying how wonderful this software is and it's millions of lines of a highly-tuned computer sciencey code that also has itself some files that it works with. We don't ever look straight at these files, we would never do that. We send SQL commands in, the magic software does its thing and retrieves the data and then gives it back to us. And it turns out, because this is kind of a network, we can send these commands simultaneously even from multiple sources. And so this is like a server. Meaning that in this case, the PHP could be a client, MySQL could be a client. And the role that we're going to be in is we're going to be what's called a database administrator because usually, when you're writing software or running production software, we developers never get to touch the end user database, but for a while, so we can learn SQL, we're going to be the all powerful database administrator. So by now, I hope that you installed something like XAMPP or LAMP because you're going to need it for the PHP parts of this class, because that has a database server all built in, a PHP server built in, all kinds of things and it just saves you a long time. It has a big bunch of files. On Macintosh, we have a thing called MAMP and we have XAMPP on Windows. I'm not going to spend a lot of time on command line, but command line is really very important, especially if you're going to run stuff on servers, maybe with a Linux server. And so, I'll just show you a little bit of command line at the very beginning with the understanding that maybe you're running Linux and you're gonna use the command line the whole time, and I think that's really cool if you do. But most of us use the command line rarely and we tend not to do too much on our servers so we just use the command line to go in on the server and figure things out if something's not working. But when we're doing software development, we tend to use this software called phpMyAdmin. So let me show you some of that. So, here's MAMP. So MAMP has this control panel that can start and stop the MySQL server and Apache server. These are two programs waiting for network requests, and I can open this web start page. Like XAMPP, they all have these little control panels. You go to the starting page. And so this is running on port 8888. If you remember URLs, we're on localhost which is a loopback, so it's a web server that's running on the server as this browser is. The outside world can't talk to this. Normally, port 80 is for web but this is going to run on port 8888 in case you're doing something because it's kind of a developer environment. So one of the tools is called phpMyAdmin. And so this is a console that database administrators, and if you run stuff in production, lots of people use this exact software. So we're not really playing in a toy environment. There are more sophisticated ways to do this but this is a very common tool that a lot of people use to do real production management of database servers. Okay? And so, we'll come to this in a second. We can type commands here but now, I'm going to show you how you can talk to the database server from the command line. I've got this handout. Let me grab the command line text from the handout and go into or it wants to test it. So, this is what you do in MAMP. It's really just a long path to a file called mysql. And I'm logging in with a user name of root and a password of root. This is not that big a deal because you can't really come in from outside into my database because a localhost is how this works. And so now I'm at the prompt, on a mysql prompt. And so, this mysql is connected to the same MySQL server that's running on port 8888 that here I am on this port. Actually it's on port 8889. The server's on port 8889. And so, I am also on port 8889. It already knew there's port 8889. So, there's commands that you can type here. Like I can type show databases;. So that's really the same information here as it is here. And, this software and the command line are simultaneously talking to the same SQL server. Neither of these are the SQL server, they are just clients that are sending commands back and forth. So watch what can happen here. I want to get rid of this database. Click on databases. Click check. Get rid of that database. Now, by the way, don't get rid of any of these databases that are here. If you didn't put it in, don't take it out. Those are internal databases that MySQL uses for its own purposes. Now I'm going to go back to the command line. I'm trying to show you how both pieces of software are talking to the same database server at the same time. So, last time I typed show databases; you saw People. If I type show databases; again, it's gone. I deleted it from one client and I see the effect of the deletion on the other client. So there can literally be tons of clients hitting the same database server. There is still one database server. And if I came over here to my MAMP console and I stop the servers, this is shutting down the database server. I've got to get it back up after I shut it down. So, MySQL server is down and the Apache server is down. Now I'm going to say show databases; This gets out. That server has gone away. Let's see if I can bring it back. Start servers. Come back. Come back. SQL came back. You see little green dot. In XAMPP, it looks a little different. You might have trouble when you first get this set up. Get some help from somebody. If it comes up, great. You need a little bit of work, whatever. Now, I think I should be able to just hit up arrow and ask again show databases; because there we go. So, that is just really showing you that there are two ways of simultaneously talking. We're talking from phpMyAdmin or MySQL but both are doing SQL. Even if I do things in here, it's actually sending SQL commands back to the database server. Now, we're going to make a database. So I'm going to use the statement CREATE DATABASE and I've got this little file that's got all these commands here. Now if you're in the command line, you actually have to switch databases. I would do this not in the command line because I did the last time in the command line. I'm going to do this in phpMyAdmin. So I'm going to be here with SQL. Let me hit refresh here. People's gone. And that one I hadn't refreshed. Wait, I've got to create a database. Hang on. This is going to fail because I'm not in a database. CREATE DATABASE People; Is that in my little sheet? Oh, they're right here. CREATE DATABASE People; This default character set is probably a good idea. Default character set means that you can have non-Latin characters so Asian characters, Greek characters, Russian characters, et cetera. So that's a good idea to do in general so I'm going to go back and come over here and create my People database. Create database People with the right character set. And so now I have a database People. And now I'm going write some SQL. I'm writing this SQL in the database People because the table that I'm about to create is going to be in that database. Come back. CREATE TABLE. So, this CREATE TABLE statement, what this is is, remember I talked about a spreadsheet and the columns of the spreadsheet? This is making the columns of the table. But I told you that when we do this for a database, we are far more precise. We're going to make two columns, one is going to be called name, the other's called email. They're both variable length character fields capable of storing characters not just in the Latin character set, and their maximum amount of 128 characters. If you try to put 129 characters in, it blows up. And you say, "Well, that's pretty mean." They say like, "Whoa, no. We told it. We only wanted 128 characters." So we're establishing a contract at this point. And part of the reason for that contract is for the efficiency that's required so it can store precisely in a certain way because it knows it's never going to be asked to store more than 128 characters. If it was going to store a million characters, it would take a different approach to storing it, but we don't have to worry about that. We just say, "Our contract with you, Mr. Database Server, is 128 characters." So I'm going to type this CREATE TABLE Users and I'm creating it in the People database. People database. There we go. So now, I see that if I go into the People database. And I find out, I've got one table called Users and I can go look at it and I can look at the columns of the Users table. And there we go, we have two columns. Every time you see UTF8, what you are knowing is that we can put non-Latin characters in here, which is a good idea in general. So there's our first database with lots of color commentary. We already did the CREATE TABLE , The DESCRIBE is something that you can type in the command line, I'll type that in the command line. So now you see we have the People database to switch to, so all your commands are aimed at the database. You say use People; database. Multiple tables can be in one database and now I can say describe, Now the keywords like describe and select et cetera are upper lower case. I tend in documentation to make all the keywords uppercase and so that is the same as what we saw in phpMyadmin, where it said oh we've got two columns there. varchar variable length character up to 128 characters. Now I didn't create the table here, but that's because the database server has the table created. Okay, so we're done with that. You can actually create all these tables using WYSIWYG GUI like the editor inside of phpMyAdmin. That's possible. Okay So now we have a table and we're going to do some INSERT statements. And so this is our first bit of SQL. And so the INSERT statement actually is INSERT INTO. Now if really super programmers were doing this would be like, oh INTO is extra. There is a goal inside SQL for you to kind of cognitively look at it and go uh, that's what's going on. So INSERT INTO is kind of just pretty there for us to read it. It's just there's a programming language that adds things to make it more readable. We are going to send this query. So there's a file somewhere and the database has this file. And so let me go to here and grab the INSERT statement. And I go into SQL, the database People and I'm going to say INSERT INTO Users, name, email, values, chuck and away I go. Okay. So now if I take a look I see that I now have one row with a name column and email column, easy enough. Now let's do it again. And it turns out that the way SQL commands work, is you can make more than one of them as long as you put a semicolon at the end. And so this is the next four INSERT statements. And I can type them all in at the same time and hit go. And it ran into the INSERTs because it has semicolon at the end of each one. And so now I have five lines sitting in there. So that's your first SQL command. Pretty straightforward. The next SQL command is the DELETE statement. And like the INSERT we think of DELETE FROM is like the keyword. Users is the name of the table. And we're going to see a lot of the WHERE clause. The way a lot of these SQL commands work is they operate against the whole table unless you limit them. So WHERE is saying don't do everything. Only do it WHERE email equals ted@umich.edu. And so this is like an IF statement in a programming language except that, it's like this whole thing is a loop. The DELETE is like go through that whole table and DELETE FROM Users but only if that happens. So you could think of that as like a loop that's going around but if the there's an if statement in the loop. And then this is only this delete is only done to the records where this turns out to be true. So it's like writing a loop and an if statement all in one statement. And this is one of the reasons that people like this. So I go grab the DELETE statement. Now if I go look at the data, one row was affected. So I take a look and Ted's gone. So there's your second SQL statement. You're doing very well learning SQL. You're actually halfway through. So the U of CRUD, Create Read Update and Delete is Update. So here we have the UPDATE keyword, table to update, The SET keyword then a column name and a value with equal signs name equals Charles. And then again if this is an implied loop the UPDATE is updated everywhere I don't think of the word everywhere sitting here. Update all the rows in Users and set the name equals Charles except for those rows where name, email equals csev@umich.edu. So let's take a look at that, copy that and paste it in. Hopefully you're keeping up. [click] SQL. Go and take a look at my data and son of a gun ever right there, the name has been updated. And I hope right now you're thinking to yourself wow that's pretty easy because we're three quarters of the way of learning SQL. I'm sure there's a few other things beyond this but we're most of the way. Select. So it's Create Read Update and Delete. But the way we read stuff is by select. So we still call it CRUD for fun. So SELECT is the statement that says read a table and the first parameter is select. Then the columns and star means all the columns. This way if you don't know what the columns are, you kind of forget what the columns are, you could just say select * From is the keyword. Users of the table and you can ask. This is a good example of how you don't put a where clause in is going to get them all, because the select implicitly is a loop. Select all from users, and then if you want just one or two you can apply a WHERE clause to that so you just tack a where clause on to the end. So let's take a look at those two commands. Select * from users; go. So now we get all four tables. I can go and then I can say, select * from users where a name or email equals csev@umich.edu. And then I can type go. Oh I type too fast I type from backwards but that shows you what an error looks like. From. So that works. But you saw what an error is. Here's another thing let me just show you this. So let's take a look at our data really quick. We got four rows, here are other things. I must say this, select * from users where name equals Alex. Now you might think this is an error. And when we run it you're going to see that it's not at all an error. And what it says is you got no rows and it's true. What did you ask for. You said go through all the rows. Show me every single one, where there's a name of Alex and it did, because there was no name Alex. That's not a flaw. You asked for something and you got it. You could check to see how many rows you got back and sometimes we do a select, say did I get a row back, if you didn't get row back you might consider as a programmer that was an error but other than that not, it doesn't have to be in there. Okay we're making good progress. So the select has a couple of features like you can change the order, so select * from users order by email. That will order the results by email.

19:26
Case is now sorting. Sorting is another thing databases do really really well. Hopefully you're just following along. Unlike clues there's another form of the WHERE clause it's like a wildcard where this is the letter E. Some characters before it letter E and then some characters after it, So I can type that one one. Okay, so I'm going to type, select * from users where name like percent sign that matches any number of characters %e%. That really is saying show me an e anywhere in it. In this case I could put a semicolon here but since it's only one command it doesn't really need the semicolon. When you really need the semicolon if I'm doing more than one command here. And so this shows me all the people that have each in their name. So it's kind of a wildcard like selection. You can also order these things and put limits on them. And this is a common thing where maybe you're paging and I'll come back to this later when we will find it more useful. But you can read up on that. You can ask for the number of rows with a count. Count as a built in function, so what we're saying is don't give me the actual rows but instead give me the number of rows that would have been returned had I ran the select statement. So I can say, select count * from users. And that's going to tell me basically, how many records are in the Users table. And so it told me four. And so, at this point, we have actually hit all the CRUD. We've learned how to insert, we've learnt how to delete. We've learnt how to update, we've learnt how to select. We learned a couple of different things that we can do in select. At this point, you've kind of learned half of SQL, because that's really what it does. It looks like a great big spreadsheet with a billion things in it and super fast. But the next thing we're going to learn, well when we start moving things between more than one tables and explore the relationships between those tables, is when it actually becomes interesting. So coming up next we're going to talk about some more detail on how we set up these tables and different kinds of fields that we can put into databases.

Downloads

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

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