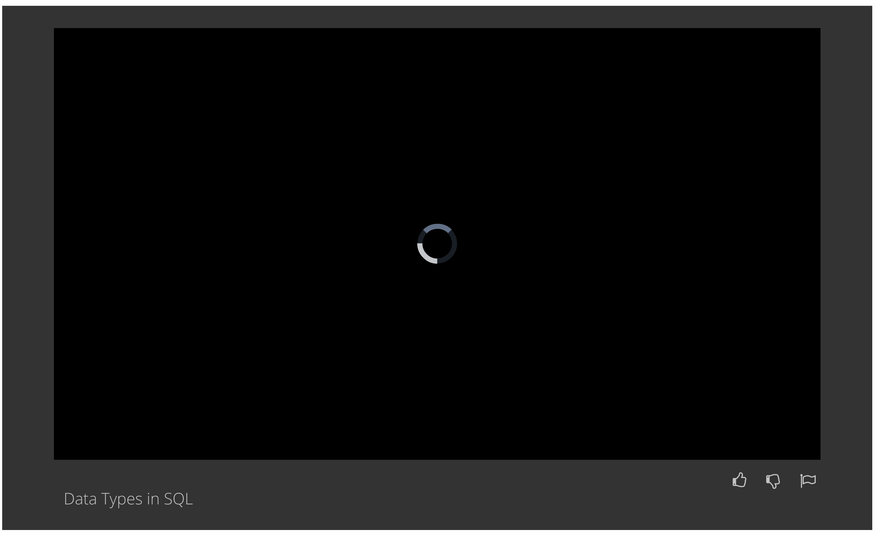

Data Types in SQL

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

Interactive Transcript

Search Transcript English

0:09
So now, we're going to talk about different kinds of ways that we can describe the columns that make up our tables. So we're in a table and we're working on columns. Different kinds are text fields, binary data fields, numeric fields, AUTO_INCREMENT fields and other kinds of fields.

0:25
So the first two columns I made of name and email were of type VARCHAR, which is probably one of the most common fields or character fields. CHAR and VARCHAR are character fields. They are aware of character sets, which meaning they can be Latin character sets or Asian character sets or Russian or Persian or whatever. A lot of what you're doing to the database server is you're giving it a clue that says, look, this might be 512 characters long, but it might also be 4. So come up with a way to store things that will be efficient both for 4-character strings and 512-character strings. If you don't say VAR, if you say CHAR, what you're saying is, this is 20 characters, and pretty much all the time it's mostly going to be 20 characters. Which means if it's only 5, it might not be efficiently stored, but it'd be very efficiently stored for things that are 20 characters. So, if it's somewhere between 5 and 500, you want to use a VARCHAR, and if it's pretty much always 20. Now, the lengths that you give to these things are absolute maximums. It's not like it compromises, like in the previous example. We did 128 and that's it, that's a contract. You're basically saying, please enforce upon me the rule that these are no more than 128 characters.

1:39
And so, there are some text fields that, the key to the text fields is that they're not indexable in the same way. Sorting, like the ORDER BY, doesn't work so well with them. But if you're putting in a blog post or a comment in Facebook, you'd probably put one of these things in. They all have a maximum length. The TEXT field is up to 65,000 characters, because it might be a small blog post, or MEDIUMTEXT, or LONGTEXT, you can have 4 gigabytes of text, of characters. Now, because this is character-set-aware field, they're real characters. And so, 65,000 Latin characters is the same as 65,000 Asian characters. And it's capable of handling all that. So the TEXT field and the VARCHAR field understand character sets. Now, the types that are very rarely used are byte types. And so, in the Latin character set, using ASCII, a character is 8 bits, that's just how big it is. And so, if all you're doing is Latin, one character is 8 bits. Whereas, if you're doing other Unicode things, characters can be up to 32 bits, whereas a byte is exactly 8. And so, you can tell, if it's binary data and you know, really, that it's only up through values from 0 through 255 characters, you can say, hey, I'd like a binary stuff, like a BYTE field or a variable length binary field, up to that many bytes. It's not really commonly used if you don't want to index, if you don't want to sort it. It's sort of not aware, it's very unaware of its content. But sometimes you're putting stuff in, you're pulling something off, perhaps a sensor or something, and you're just putting it in, and it's just a bunch of 0s and 1s. That's what this is used for. You can also store things like images or PDFs or videos in databases. It turns out that databases are pretty good at this, and these are called BLOBs, binary large objects. And they also have various lengths, depending on whether they're tiny, regular, medium or long. So it turns out the databases are really good at storing this stuff, but the problem you run into is that it starts to slow your database backup down quite a bit after a while. And so, what you find is that if you're doing medium-size things, like maybe a picture here and there, or a profile photo or something, we tend to put that in the database because then it's kind of with all the rest of the data about you. But if we're doing things like videos or large documents, we tend to find some other way to store them inside the computer, usually just as a file somewhere, and then we serve them up. But it's not bad to store binary objects like this in a database, except when it comes to your backup getting too large.

4:21
Integers, there's different sizes of integers, and you say, why? And the answer is well, we're going to have millions of these things. And so, if your integer is only really a number somewhere between 1 and 15, we don't need to store the same amount of space as if it's a 2 billion number, right? And so, we have tiny integers, small integers, normal integers, that's 0 through 2 billion. Like the INT is a 32-bit integer, if you're sort of nerdy like that. And then there's even larger big integers that you can store that take up more space. Integers are nice, especially the INT integer, the one 0 through 2 billion. They sort really fast. They take up not that much storage. They're easy to compare. They're easy to sort. They're used for indexes. And so, we tend to use integers in a lot of things. And we tend to prefer them over strings, as we'll soon see representing information. Floating point numbers. Floating point numbers work like all programming languages. If you've been using things like 98.6 or blah-blah-blah.14 or something, 6.02 x 10 to the whatever, you've been using floating points inside computers all along. The key thing about floating point numbers like 1.7, they're not perfectly represented because they're not really real numbers. They're floating point numbers, but they're approximations. So the way to think about floating point numbers, FLOAT is the small one, it's a 32-bit floating point number. It has a range of up to 10 to the 38th power. But the key is, is that no matter how big or small it is, there's only seven digits of accuracy. And so for things like temperatures or speed or things like that, we tend to find floating point numbers are perfectly good, because can you really measure speed to more than seven digits of accuracy? If you can do better than that, then you go up to 14 digits of accuracy. But after a while you're like, okay, 14 digits of accuracy is probably enough. You just have to understand that you don't want to store money in a floating point, because $10.25 is not perfectly represented, and so you get inaccuracy. So you actually use scaled integers to store money, it turns out. But for scientific calculations, and you put plenty of scientific data into databases, floats and doubles are great.

6:41
There's a number of different time and date formats. There's a thing called the TIMESTAMP. And the TIMESTAMP is the number of seconds since 1970, and it's stored in a 32-bit number. And it's really efficient and easy to sort, because it's really an integer. But it has a couple of problems. It can only handle seconds, it can't go below a second. And it has an absolute sort of length. Being that the TIMESTAMP idea was invented in 1970-something, they said, it's the number of seconds since 1970, January of something 1970. Which means that we have kind of a Y2K problem in Unix systems and in database systems in 2037, because that's when the 2 billionth second after January of 1970 happens. But until then, they're fine. And then we'll make it a 64-bit number, and then we'll have until the sun dies or something like that, so we don't have to worry about that too much. So, if a data item that is only per second, like when a record was updated or created, we tend to use TIMESTAMP for all that. DATETIME is more general, takes up a little bit more space, and it can really represent any year, like the year 1300, you can still represent them. because you can't put 1300 in a TIMESTAMP, because it's before 1970. So DATETIME takes up a little more space, but literally any number that you can represent in this format, and it's beyond 2037 and it'll be fine. And the same is true for a DATE which is just any set of year-month-days that fit, and a TIME, that is hours, minutes and seconds. And there is a built-in function in MySQL that tells you the current date. So it would be, INSERT created_at value NOW(). So, NOW is a way to say what time is it now, as understood by the database server. So up next, I'm going to talk to you a little bit about how to give even more detail not just about what kind of data is in each column, but something about how you intend to use that data in each column.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

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