BYU-Idaho Online Learning
Video Transcript

Data Types

[One Speaker.]

[There is an animated picture of a computer on a desk. The computer screen says "Characters, Numbers, Dates and Times."]

Narrator: In this lesson, we'll be looking at how computers and devices store your data and what that means for storage requirements, and using your resources efficiently, while you're storing your data. We will also look at the types of data, or data types, that can be used for all the data that we'll be storing in our database.

[On the screen, it says, "How data is stored." There are "icebergs" with the labels Bits and Bytes, Unicode, and Basic Data Types. A man on an origami boat looks at them through a telescope.]

When it comes to the computer storing your data, it's a good idea to understand how that works. At the most basic level inside of your computer...

[The screen zooms in to show "ASCII," "Byte," and "Bit" by the Bits and Bytes iceberg]

...the CPU, or central processing unit, makes all of the important calculations while your computer is running.

The only basic unit of communication that the CPU understands is ones or zeros.

[The video zooms in on "bit." There is a picture of a CPU, and "1 or 0" is labeled on the screen.]

There are billions of transistors or little gates in your CPU that are either open, which is the one or closed which is a zero. This is how all the data going through the CPU is going to be recognized. This system of ones and zeros is called binary. Using binary, there are just the two digits: either the one or the zero. And each one or zero is called a bit.

[The slide changes and zooms in on "byte." Beneath, it is says "8 bits."]

Eight bits together form a byte. Every character and symbol of data is made up of eight bits or one byte. Here are the bytes or combinations of ones and zeros, that make up the letters in the name Anderson.

[A:1000001, N:1101110, D: 1100100, E: 1100101, R:1110010, S: 1110011, O:1101111, N: 1101110]

So this is how your computer understands what's typed on the keyboard, as it goes into your computer. It will be transferred into these ones and zeros and that's all that your CPU or any computer system's processing unit will understand.

[The slide changes and zooms into the "ASCII Table."]

[There is a table with the columns Decimal, Hex, and Char.]

ASCII, is the character encoding based on the English language. It has 255 characters that correspond to the different numbering systems. So a lot of these characters, make anything that's on your keyboard would fit in this ASCII. So each of these characters also correspond to a binary byte. So I'm going to zip through all 255 of them here in a minute...

[The screen shows a table with the ASCII, Decimal, Hexadecimal, Octal, and Binary, and many tables appear on the screen one by one of similar charts of information]

...and you can see that not only are they letters but they are printable characters, special symbols, carriage returns, lined spaces, tabs all sorts of things. Then when you start inserting into symbols, and you'll see the uppercase letters and if you look over to the right very right column you'll see the actual bits that make up a byte that represent that character. So here's the rest of the uppercase, lowercase letters. At the end we've got some mathematical symbols, language symbols, trade symbol things like that.

[The screen changes to go to the Character sets iceberg, and next to it are two labels: Unicode and latin1.]

So those are the 255 different characters that are represented with those different bytes. You don't need to really remember that but it's kind of interesting to see how that works. You don't need to remember all the particulars of that. But because these 255 ASCII characters sets were not enough to cover all the languages in the world, internationally, other character sets were developed.

[The screen zooms in on "Unicode." The screen says, "Enough to cover all European, Middle Eastern and Asian languages, 1-3 bytes per character." There is a diagram at the bottom of some characters and their code.]

So Unicode, is also known as UTF-8 adds on to the ASCII so that all international languages can be included including, Asian language characters which takes a little more space. So now some characters needed code have to be made up of more than one byte per character. The first 255 are the same as ascii and only take one byte per character, but the additional characters in Unicode in the Unicode set can take up to three bytes each. So Unicode is now the standard that you should build your databases in because it can transfer to any language easily.

[The screen changes to show, "latin1." The screen says, "Only supports Western European Language (english). 1 byte per character."]

You will see maybe an option to save it as Latin one, and database but that was the old standard that just includes the English based languages. But it does just have one byte per character.

[The screen changes to show the Basic Data Types iceberg. The words, "Character, ABC Corporation Smith, 208 321-4321" appear next to the iceberg.]

Okay. So when we enter into our database. We have to decide what type of data it is and make sure it fits in the categories of data that we're going to cover here. There are three main types of data. Okay the first is characters, they are also referred to as strings or text. This could be names, titles, it could even be things like social security numbers, phone numbers or zip codes even though those are numbers we know we're really never going to be doing math on them. So any text that you won't really be doing, need Tab math than on them would fit in this character type.

[The words on the screen change to say, "Numeric, 45.99, 20, 456742.92001."]

The next basic type is numbers, this could be decimals, or integers, or whole numbers, very precise numbers. Anything that could potentially need to be used in math some way would be stored as a number.

[The screen adds on the following: "Date and Time, 2025-12-30, 12:33:20."]

The third main section is date and time. Dates are always entered with year, year, dash, month, month, dash, day, day, and times as hours, colon, minutes, colon, seconds. Okay. Let's take a closer look at each of these.

[The screen zooms out to the first image with the computer on a desk. The screen zooms into the computer, and it says, "CHAR, fixed length strings."]

First we'll look at characters. One way to save your text or string data is using the CHAR, or character data type. So it's actually C-H-A-R. When you use CHAR, you're going to use that when you know the length of characters in your data. For example, we know the length of zip codes, it never changes. We know the length of social security numbers and phone numbers, that length of actual characters never changes. So we could use char with any of those. If we had a column that was for two letter states.

[The screen says, "3 bytes for each character (zip code, SSN, phone num.)"]

So let's just look at this for example, so with the fixed length characters per char it's going to take three bytes per character that you use.

[The screen says, "2 character state code: CHAR(2), 3 bytes for each character, so 6 bytes in total."]

You would do something like this, for example with a state, two digit state code. You would put CHAR and then inside parentheses you would put two representing that you just need two characters. But remember, for each of those bytes, for each of those two you would have three bytes of information. So six total for something like that.

[The screen changes to show, "VARCHAR, variable length strings."]

Another data type for characters is varchar or variable characters. You could use this data type when you don't know how long a string of characters might be ahead of time.

[The screen says, "1 byte per character plus 1 byter for length, Used when the length of the data entry will vary (names, titles, etc.)"]

We won't, maybe we won't know the length of each name that might be entered, or a title or a product name, or an address, things like that, would vary in size. If we had a first name column that we are defining we could say VARCHAR 20.

[The screen says, "first underscore name, VARCHAR(20), 1 byte for each character, plus 1 byte for the length, so 5 bytes in total."]

And in parentheses we'd have the 20 to say that we aren't sure how long the first name might be but we will set aside 20 characters for it just in case. Like that would be the biggest first name we might have.

The variable character data type has one byte of storage for each character. Plus just one byte per the hoping to keep track of the length. So if our name was Jane, J-A-N-E, we would use only four of the 20 possible bytes plus one bite. So to keep track of the length with that one byte. So Jane would use 5 bytes total. Even though we set aside 20 bytes possible. So it does a pretty good job of being efficient with our resources for storage. I don't use CHAR, C-H-A-R, very often, even though I might know the length of my data. If I know that varchar will use less storage, even if it's a set length, I will still use VARCHAR. So that's how the CHAR and VARCHAR works.

[At the bottom of the slide, it says, "for CHAR and VARCHAR use single quotes when entering the data in a statement."]

Remember when you use char or varchar when you're actually entering the data into your system, you do need to put single quotes around the text that you're entering when you enter data of this type.

[The screen changes to show, "Numeric, Integer and Real Numbers."]

Okay, let's look at the next type of data. We've got numbers or numeric data, we have integers to start off with.

[The screen says, "Integer (INT) is a whole number without decimals. Use TINYINT for Boolean(0 is considered false), 4 bytes of storage, (primary keys, inventory, etc.)"]

Integers are whole numbers or numbers without decimals. For example, primary keys are commonly integer data type, there are also types of integers that go by the name BIGINT, MEDIUMINT, and SMALLINT, and TINYINT. And as their name suggests they are also store integers but the range of how big or how small and negative numbers can differ. Usually INT works well for most data, TINYINT does have a good use though for

Boolean type data, Boolean is where you just have two possible values true or false. So a TINYINT with a one in parenthesis would work like a Boolean. So a value of 0 would be false and a value of one would be true, so there's a good use for TINYINT. I-N-T, integer data type takes four bytes of storage and TINYINT takes one byte of storage.

[The screen says, "Real Number can have decimals. fixed -point type: DECIMAL(9,2), 9 digits total (precision), 2 digits after the decimal (scale) 5643215.22, 4 bytes for 9 digits, (money, precise math, etc.)"]

Another type of number is a real number or numbers that can have decimals, when you use the DECIMAL datatype, D-E-C-I-M-A-L, you can store fixed point numbers which are numbers that have a fixed number of digits to the right of the decimal point. If we had DECIMAL 9, 2 like we do here, we are saying that we have nine digits total, on the left and the right side of decimal points, so total on both sides, and two digits after the decimal place. So those are the two numbers, the nine and the two. The nine digit total is called "precision" and the two digits after the decimal is called "scale." So small numbers and ranging all the way up to a number like we have here, five billion six hundred and forty three thousand two hundred and fifteen point two two, would fit because there's actually just nine digits. So you couldn't go much higher if you've got to 60 billion then you would go up to ten digits and it wouldn't fit in that data type that you described there the nine comma two you'd have to go ten comma two to get up to there. So you've gotta be careful and make sure you've got their range that you need there.

The amount of storage for these numbers can vary according to how big the precision scale are that you set, precision will top out at 65, but that's still a really huge number and the scale part of it, the decimal part of it will top out at 30 decimal places.

[The screen says, "Other Real Number: DOUBLE (8 bytes) and FLOAT (4 bytes), floating-point numbers, considered approximate numbers)."]

Okay, another two data-types that fit within real numbers are DOUBLE and FLOAT numbers, they both store data that are floating point numbers, there's no fixed number of digits before or after the decimal point. They can handle really large or really small numbers. DOUBLE takes eight bytes and fixed takes four bytes. These types of numbers are usually needed for very scientific calculations that have to be super precise. We won't be using these data types in this course.

[The screen changes to say, "Date and Time: Date(no time): yyyy-mm-dd: 3 bytes

Time (no date): hh:mm:ss: 3 bytes

DATETIME (both): 1970-9999: yyyy-mm-dd hh:mm:ss: 8 bytes

TIMESTAMP (system time): 4 bytes

YEAR:1901-2155: 1 byte."]

Okay, the last data types that you can set with your database are date and time, and there's some different combinations here. Date, just D-A-T-E by itself, is just the year month and day and it takes three bytes. You have to enter it again with the four-digit year dash two-digit month dash and the two-digit day. Time is just the hour, minutes and seconds, it also takes three bytes and it's hours, two digit hour colon, two digit minute colon, two digits seconds. The date time has both the date and the time together, and it's good to remember that with date time you can only enter dates from the year 1970 to 9999, but they would have both the year with the month and day space and then hours minutes seconds. Okay, timestamp, we'll get a similar format to date time but it will get it from the user's computer or their system that they're currently using at that precise time, the year is a four digit year and it can range from 1901 to 2155 and it takes one byte.

[At the bottom, the screen says, "These date literals need to have single quotes around them when entering with a statement."]

As you enter these dates and times in statements again you need to have single quotes around them, like the characters because dates have dashes in them and the time has colon's, they're treated like strings and you have to have the quotes around them so you can answer them in when you're using the MySQL syntax.

[The screen changes to say, "ENUM and SET: ENUM ('yes', 'no', 'maybe') stores one value from a list

ENUM ('soup', 'salad')

ENUM ('cash', 'credit', 'debit')

SET ('pepperoni', 'mushrooms', 'olives', stores zero or more values from a list."]

Okay a few other data types just to be aware of ENUM and SET. These are good when you want to restrict the user to a set of string values. For example, ENUM stores values that are mutually exclusive, this means that you can only choose one from the set. For example, if you only wanted them to be able to enter the data yes, no, or maybe, then you could use those as the values when you set up the ENUM datatype. The user is restricted to only those choices. SET also restricts the number of options but the user can choose more than one value if they choose to. A good example here would be the possible toppings on a pizza, they can have one to many toppings but they're still restricted to what toppings are even offered.

[The screen shows a chart with two columns, "Checkboxes" and "Radio Buttons." Options 1 and 4 are checked in "Checkboxes," and Option 2 is selected in "Radio Buttons."]

I like to think of them as the radio buttons and the check boxes. So when you have an online form that you're filling out then the checkboxes are set and you can check more than one at a time. The ENUM are like the radio buttons where you can only choose one at a time they're mutually exclusive.

The screen changes to say, "BLOB: Stores string of binary data. Images, Audio and Video."]

All right, BLOB is the last data type I want to go over, this stores binary output. So nowadays we have a lot of images, audio files, and video files, and this can handle that as long as those audios, images, and videos are broken down into ones and zeros and binary, they can be stored into BLOB. This can even include PDFs or Word files, these can take a lot of storage. Sometimes it's easier to use them with other programming languages, so we won't be using this data type in this course but it's worth knowing what BLOB data is.

[The screen zooms out to the opening screen.]

So there we go. The ways data is stored and some of the more common data types that you will see as you put together your database so we can store our data properly.

[End of Video.]