SQL Database Data Types

Introduction

In this lesson, you'll cover the importance of specifying data types, and the different types of data you can store in a SQLite database.

Objectives

You will be able to:

- Describe how data types operate within databases and explain why they are necessary
- Compare the 4 different types of data you can store in a SQLite database

Why Do Data Types Matter?

You've seen that when you create a table, you need to include a name for it as well as define at least one column. You define columns in a `CREATE` statement by including a name and a data type to let SQLite know the kind of data you will be storing there. The practice of explicitly declaring a type is known as "typing."

Why is it important that you use typing in our database? Simply put, typing allows us to exercise some level of control over our data. Typing not only informs our database of the kind of data you plan to store in a column but it also restricts it. For instance, look at the age column below in our cats table. What do you mean by age? What if you had this:

name	breed	age
Maru	Scottish Fold	3
Hannah	Tabby	two
Lil' Bub	American Shorthair	5.5

Did you intend age to be represented as a whole-number, a word, or a decimal? If you were asked to add up the ages of all the cats you could simply convert the `'two'` to `2` in your head, but your database can't do that. It doesn't have that ability because the logic involved in converting a word into a number would be dense and inefficient. What about different languages? What about different spellings? Capitalization, typos, or different hyphenation conventions? These are just some reasons this might start to get overwhelming. In other words, because databases are designed to store large amounts of data, they are very concerned with storing, accessing, and acting upon that data as efficiently as possible, which requires typing of columns.

Typing gives us the ability to perform all kinds of operations with predictable results. For instance, the ability to perform math operations like `SUM` (i.e. summing integers). If you tried, for example, to `SUM` all of the cats in the above table, SQLite would actually attempt to convert, or cast, their type to something it can `SUM`. It would try to convert anything it can to an `INTEGER` and ignore alpha characters. This can lead to real problems. Without typing, our data might get complicated and messy, and it would be difficult to ask the database questions about large sets of data.

Simply put, it's important to adhere strictly to only storing data that fits with the data type you have given to a particular column. Once you have assigned a data type, most databases will actually throw errors rather than allowing the invalid data to be inserted.

Data Types

Different database systems also have different data types available, which are important and useful to know whenever you are dealing with those systems. SQLite is a good starting point to learn about data types because it only has four basic categories of data types; they are:

```
TEXT
INTEGER
REAL
BLOB
```

Let's explore each category in more detail.

TEXT

Any alphanumeric characters which you want to represent as plain text. The body of this paragraph is text. Your name is text. Your email address is a piece of text. Your height, weight, and age, however, are probably not.

This SQL data type roughly corresponds to the `str` data type in Python.

INTEGER

Anything you want to represent as a whole number. If it's a number and contains no letter or special characters or decimal points then you should store it as an integer. If you use it to perform math or create a comparison between two different rows in our database, then you definitely want to store it as an integer.

If it's a number that doesn't represent a quantity (e.g. an ID number or zip code), it depends on the context whether you should store it as an integer. You might never add two house address numbers together, but you might want to sort them numerically. On the other hand, if you have meaningful leading zeroes (e.g. `02134` , which you don't want to treat as equivalent to `2134`), then it's better to store it as text.

This SQL data type roughly corresponds to the `int` data type in Python.

REAL

Anything that's a plain old decimal like 1.3 or 2.25. SQLite will store decimals up to 15 characters long. You can store 1.2345678912345 or 1234.5678912345, but 1.23456789123456789 would only store 1.2345678912345. In other database systems this is called 'double precision.'

This SQL data type roughly corresponds to the `float` data type in Python.

BLOB

You may encounter the `BLOB` data type while you're Googling or doing any further reading on SQLite. For now, you will not use `BLOB` . It stands for "binary large object" and is generally used for holding binary data such as images or audio files. When we use those kinds of files in this course, we will generally just store them using the computer's file storage system (hard drive), rather than using databases.

SQLite and Other SQL Implementations

To increase its compatibility with other database engines (e.g. mySQL or PostgreSQL), SQLite allows the programmer to use other common data types outside of the four mentioned above. This is why we are referring to `TEXT` , `INTEGER` , `REAL` , and `BLOB` as data type "categories". All other common data types are lumped into one of the four existing data types recognized by SQLite.

For example, `INT` is a common data type used outside of SQLite. SQLite won't complain if you define a column as an `INT` data type. It will simply lump it into the `INTEGER` category and store it as such. Similarly, if something is declared as `VARCHAR` , SQLite will treat it as `TEXT` .

To accommodate this, SQLite has a pretty complicated system of categorizing data types that involves *Storage Classes*, *Type Affinities*, and *Datatypes*. There is also a fifth data type category called `NUMERIC` that can actually contain any of `TEXT` , `INTEGER` , `REAL` , and `BLOB` types. This one mainly comes up for Python developers when you are working with `BOOLEAN` data (like `True` or `False` in Python), which becomes a `NUMERIC` column containing 0s and 1s rather than some separate boolean data type.

For a deeper dive, check out the **SQLite3 Documentation on Datatypes** (http://www.sqlite.org/datatype3.html)

Summary

Great! Now that you've finished this lesson you know why it is important to specify data types, and you know about the different types of data you can store in a SQLite database.

Have specific feedback?

Tell us here! (https://flatironschoolforms.formstack.com/forms/canvas_feedback?CourseID=965&LessonID=sql-database-data-types&LessonType=pages&CanvasUserID=5434&Course=None)