Week 10 - Relational Databases: SQL and SQLite

Last week, we learned about technologies that power the web, like HTTP and HTML. For example, you learned that when you make a request to a web server from your browser, that server can respond to that request with an HTML document. However, in a lot of cases, when you go to a URL, you are not getting the same content every time you make the request. When you go to facebook.com, for instance, the content you see will be different every time. In this case, the HTML you are receiving is being *dynamically generated*, which means that the HTML is being constructed on the fly.

How does that work? Where is the data coming from that populates this HTML response? This week we will move on to another big topic in the Web stack, and that is relational databases. They are responsible for storing data that can then be *queried* for that data. Relational databases are not the only kind of databases out there, but are crucial to most web sites, as they are the source of data that is uses to generate HTML responses.

This week we will go over how relational databases work, how to setup SQLite and how to use SQLite from Python.

Relational Databases

So, what exactly are relational databases? Relational databases organize data into one or more tables of rows and columns. Think of a table like a Excel spreadsheet. Each spreadsheet is made up of rows and columns. Typically, one of the columns in a relational table is what they a unique or primary key. This key can pinpoint a particular row in a table.

Generally, each table represents some kind of thing or entity. For example, we can have a table called Employees, which is meant to store data about a company's employees. Each row in the table represents an instance of that entity (in this case, Employees) and the columns represent certain attributes per each instance. A column in the Employees table might be "first_name". That means, every row will have some value in the first_name column, which represents that employee's first name.

For this section, go over the Relational Databases Tutorial, which should cover the basics when using and designing a relational database.

SQL and SQLite

Why is it that we store data in a relational, or any kind of, database? Relational databases not only store data, but they let you *query* them to get back that data when you need it. The way to query this data is called *SQL*, or Structured Query Language. SQL is a formal language for how we can ask these databases to get data for us.

There are many different relational databases out there. One of the most popular, free, open-source relational databases out there is called MySQL. MySQL is very popular in the web development world, but we will not use it here since it can be a bit of a pain to setup. A better choice for us is SQLite, which is a smaller and easier to use relational database. You may already have some experience using SQLite from previous courses. Even if you have, read the Back to Basics: Writing SQL Queries tutorial, which goes over how to use SQL and SQLite together.

Also, make sure to check the Optional Reading and Useful Tools section of this week's material, which will have some more links about SQL.

SQLite and Python

Now that you have a good understanding of SQLite, we need to know how we can utilize it from within a Python program. We can do this by utilizing the *sqlite3* Python module. Luckily, the module should be installed already in your Python installation. To read up on how to to use this module, read this SQLite Python Tutorial.

NoSQL

Recently, there has been a 'movement' around what is called NoSQL. Essentially, NoSQL databases are databases that store data that is not in the relational form of tables linking to one another. For example, Redis is a popular NoSQL database, but it works as a key-value store. That means Redis is more like a dictionary than it is a table. In some future courses, you will run into other NoSQL databases, like MongoDB (a document store) or Neo4j (a graph database). For the purposes of this course, we only need to know about SQL databases; however, they all have the same purpose: store data in such a way that we can query it to create dynamic responses to web requests.

Flask and Relational Databases

So, how does all this fit in with what we are doing in the second half of the course? We will be using SQLite to store data persistently, which can then be queried by Flask to generate dynamic responses.