Week 1 - Learn

Introduction

Welcome to CS340. This course is all about relational databases. There is a major focus on relational database design. We will talk about how we structure databases so that we can add things to them, along with ways we can keep track of how those things are related to each other.

For example, we might use a database to track roads in a city (one thing) and traffic control devices (another thing). A relationship between them might tell us what sort of traffic control device is at mile 3 of 2nd St.

In addition to this sort of design, we will also get into the more technical aspects of writing queries to get the database to do what we want. Finally, we will end by integrating a database with a front end. Usually we do this by connecting a MySQL database up to a website being served with Node.js.

So what will you be doing during your stay here? Mainly you will be working on designing databases. For a while you will spend some time writing queries to create tables (or data within tables), but it will all be in support of designing databases. In the end you will put it all together into a personal database project.

The day-to-day work will be somewhat varied. There will be some tasks which require you to create diagrams, others will require writing SQL queries or Node.js code, and yet others will require using mathematical notation to describe queries. What is challenging and time-consuming for one individual might be enjoyable and quick for another. So you have to use some judgement when you see estimated times for various portions of the class.

Key Questions

- Generally, what are the topics covered in this class?
- What are the major assessments in this course? (e.g. exams or projects)
- What are the major tools that are used in this course?
- What defines a database as separate from a collection of data on your hard drive?
- When is it appropriate to use a database?
- What are some general attributes that databases often have?

Introduction to Databases

You have probably heard the term "database" in the context of computing so many

times that you don’t ever even stop to consider what it means. Every month major news outlets report on some company’s database that got "compromised". But what does it mean that their database got compromised? Is that somehow different than their hard drive being compromised where the data was stored? Databases and hard drives both store data, so what is *actually* the difference between them?

It's all About Relationships

The common factor among databases is the existence of relationships. Databases store collections of related data. So employees might be related to departments as well as to job descriptions, students might be related to classes, orders could be related to customers, and so on. There is a relationship between things.

If you cannot set up relationships between things then you are not working with a database. For example, if you are using Google Sheets, an online spreadsheet tool, you can set up relationships between cells but that is about the extent of it. It isn’t really possible to define particular cells as being employees and others as being departments. You could get close but in the end the spreadsheet will not enforce those sorts of rules reliably.

A database will do that.

Structured Data

Not too long ago I would have made the argument that having structured data is a defining requirement of a database. And by some academic definitions this should still be the case. But there are more and more offerings which allow the user to put *whatever* data they want and to set up relationships between that data.

These offerings tend to offload a lot of the responsibility onto the application. They don’t enforce as many rules as a more ‘standard’ database would, but they will still enforce and allow for relationships between objects to be formally defined.

But the databases you will be working with in this class do allow you to set up a data structure, and that is by far the more common setup. You can define ahead of time what sorts of things can be in a particular table, much like defining a class in an object oriented programming language. Then the database will complain and throw an error if you try to put in data which does not fit the defined structure. It also allows you to do some operations like check for greater-than or less-than relationships between numbers, or perform regular expressions on string-like data types.

This also makes it easier to keep track of types when you pull data out of a database and manipulate it in a different programming language.

What's Not a Database?

Data Stores

A good, catch-all term for things which are *like* databases but not *really* databases is "data store". There are a lot of *very fast* data stores which are tasked with keeping track of visitors to a website or monitoring the status of complex systems. They might keep data for minutes or hours, but that data will later get filtered and stored in a database.

These systems can be so fast because they don’t enforce rules about relationships or do as many checks to make sure the data being collected is ‘good’ data.

Spreadsheets etc.

There are also things that really are not data stores or databases, but still get advertised as such. Things like spreadsheet software or personal backup software. These better represent storage. There are all kinds of interesting problems related to data security or making spreadsheets more useful for business, but they are not database problems because they don’t set up relationships between things or even set up a structure for the data to fit into.

Concluding Remarks

The line between data stores and databases continues to blur as technology improves and it simply isn’t worth splitting hairs over what is formally a database and what isn’t. But it is important to ask the questions about capabilities before committing to a using a particular tool or before planning a project that has to interact with one. Here are some good examples of questions to ask when choosing a database tool:

Are relationships supported? How well are they enforced? How much structure can I put in place to make sure my data meets certain requirements?

The more you can enforce rules about structure and relationships, the more you can do in terms of having the database ensure data correctness.

Video Lecture

This is an old but still relevant video by Justin Wolford from the oldest version of this course when the course number was different.

Introduction to Databases **(https://media.oregonstate.edu/media/t/0_woizuvqb)** (19:50)

OSU
CS 275 – Introduction to Databases
Week 1 - Overview of Databases
0:00 / 19:50
1x

View the Slides PDF (https://oregonstate.instructure.com/courses/1727186/files/73434298/download?wrap=1) (https://oregonstate.instructure.com/courses/1727186/files/73434298/download?wrap=1) (https://oregonstate.instructure.com/courses/1727186/files/73434298/download?wrap=1)

Week 1 Assignments

This week you will complete a task with relatively few requirements. Basically you will set up a very simple database and add a little bit of data to it. This will prove to yourself and to the instructor that your system is set up to not get in the way of completing the future assignments in this course. (https://oregonstate.instructure.com/courses/1727186/assignments/7471309)

You will also get started on your project this week by getting a website connected to a database and forming a Project Group with another classmate.

Review

This module is all about getting everything set up and getting you oriented. At this point you should have been able to connect to an OSU database, create a table there and put some data into it. In addition you should have confirmed you can get a 'Hello World' program working in your server side language.

In addition to those technical tasks you should also have a basic idea about the sorts of things that separate a database from other sorts of data storage options.

From here we will move on to talk about the key components of database design in the upcoming weeks.