Introduction to Databases

loosely based on the following, and other sources as noted

Web Development with Node and Express by Ethan Brown Published by O'Reilly Media, Inc., 2014

Persistence

- At some point, we're going to want to keep data around for a little longer than just a single session. We need to store data in a persistent way.
  - What are some examples of data persistence in websites or web applications that you use?
- Databases allow us to separate that which varies from that which stays the same. (Connolly and Hoar)
  - What do we mean by this? What is varying and what is staying the same?
- Most websites that are at least moderately complex will require a database.

Web Development with Node and Express by Ethan Brown Published by O'Reilly Media, Inc., 2014
(**Fundamentals of Web Development (2nd Edition) Paperback – Feb 8 2017** by Randy Connolly (Author), Ricardo Hoar (Author)

Databases: Relational

- Traditionally, **relational** database management systems have been used, based on the **SQL** query language.
- Relational databases are very powerful and allow for organized and fast manipulation of data.
  - Relational SQL-based database management systems allow you to store chunks of data as **rows** within collections called **tables**. The **columns** of the tables represent different types of data within each row. The **tables** are collections of related data.

Databases: Relational

- Example: in a database called **online-store**, we may have a **table** called **customers.** Each **row** in this table represents a **customer**, and may have **columns** such as **first name, last name, address,** etc.
- We could use the SQL language to quickly get customers from this database according to certain filters and constraints (for example, maybe we only want to get information about customers who purchased a particular product) and quickly add additional customers to the database when they register on our site.

Databases: NoSQL

- However, so-called "NoSQL" databases have become more popular recently. These databases break the **table/row/column** paradigm that has been established. They have some advantages, although they are not appropriate for every scenario. They happen to be good for quickly prototyping a web application.
- **Document-oriented databases**: this is one type of NoSQL database that stores **objects (called "documents") rather than rows.**
  - Why could this be a good way of structuring databases?
- MongoDB is a popular document-oriented database system.

Database Management Systems

In this course, we'll study both **Relational** and **NoSQL** database systems, starting with **Relational**. But even within this paradigm of databases, there are multiple **database management systems** (or DBMS for short) to choose from. They each have advantages and are suited to different cases, and the one you pick for a given application will depend on your needs. You will probably even use more than one DBMS for a large application.

Database Management Systems

Here are a few common relational DBMSs:

- MySQL: the most-used DBMS; it's older and well established
- MariaDB: a drop-in replacement for MySQL that is mostly the same
- PostgreSQL: a newer, more flexible DBMS with rapidly-growing popularity
- SQLite: a simple and easy-to-install SQL DBMS that doesn't require a special server; it simply uses the file system and memory. It's appropriate for small-to-medium web applications and growing in popularity.

Database Management Systems

Here are a few common relational DBMSs:

- MySQL: the most-used DBMS; it's older and well established
- MariaDB: a drop-in replacement for MySQL that is mostly the same
- PostgreSQL: a newer, more flexible DBMS with rapidly-growing popularity
- SQLite: a simple and easy-to-install SQL DBMS that doesn't require a special server; it simply uses the file system and memory. It's appropriate for small-to-medium web applications and growing in popularity. ***Due to its simplicity and ease of installation, we'll use this one!***