COMP 1140: Database and SQL | Brendan Shea, PhD (Brendan.Shea@rctc.edu)

WHAT ARE DATABASES? WHY DO WE USE THEM?

In this lesson, we'll be answering the following questions:

1. What is the difference between **data** and **information** (or **knowledge)?**
2. What are databases, and how can they help us turn data into information?
3. What are the advantages of databases over **flat files,** such as paper records or spreadsheets?

CLASS OVERVIEW

In this class, we'll be studying the basics of **databases:** what they are, how they are used, and what you can do them. In particular, you'll be learning about the following:

1. General issues of database design and implementation, in particular of "relational" databases.
2. How information can be entered and retrieved from databases
3. How to use **Structured Query Language (SQL),** the most common way of *interacting* with relational databases.
4. How to query a database, using either SQL or other interfaces.
5. How to analyze and use the results of these queries.

A note: Databases should be fun! Databases often strike people (even some programmers) as being "boring." I hope this class will convince you this isn't true! Database design can often be the most creative part of a programming project, and the language used to access databases (SQL) offers a significantly different way of thinking about things than the **procedural** languages (like Python, C, or Java) that you may be used to. Designing and implementing a good database often requiring answering hard (and interesting!) questions about how different parts of the world "fit together" and how we can use computer systems to help us make sense of it.

DATA, INFORMATION, AND KNOWLEDGE

In some general sense, a "database" is simply a collection of data, whether this be a pile of notecards, a shopping list typed into a word processor, or a spreadsheet tracking your recent purchases. In the context of business and computer science, however, a **database** refers to a particular way of storing data. In particular, databases store **end-user data** (the data entered by users) together with **metadata** (or "data about the end-user data"). This is very different from other ways of storing data! So, for example:

1. An MS-word document (such as this one) might contain a string of characters like "$15,235". However, a well-designed database could use its metadata to determine that this is meant to be the amount of money in a particular person's bank account.
2. A spreadsheet might contain a series of number like 35, 54, 56, etc. A well-designed database would "know" that these represent the hours per week worked by different employees (and it would know how these numbers are *related* to things like the employees' total compensation).

A **database management system** is a program (or collection of programs) that allows people to *interact* with databases.

Why Databases Matter. Databases help us humans convert **data** (raw, unstructured facts about the world, many of which are perfectly useless to us!) into **information** (where we understand the *meaning* of specific data for our interests) and eventually **knowledge** (the "collected" information we have about a certain subject). This, in turn, helps us make better decisions about our lives. For example, suppose we are running a bookstore:

- *Data* consists of things such as ISBN's of various books, the price for each book, a shopper's credit card number.
- *Information* is the result of processing data in ways that make it meaningful to the user. For example, we might have information on "the total sales over the last month" or "number of books customer X purchased by author Y."
- *Knowledge* is a comprehensive collection of information relevant to our purposes. For example, as bookseller we might use our information to answer questions like "How do customer's reading tastes change from winter to summer?"

Well-structured databases can help us receive better medical care, make education work better, help us find products to purchase, and advance scientific research. In the right context, databases can make **querying** (asking questions) data much more efficient and accurate. The flip side of this is that NOT understanding how databases work can lead people to make bad, unproductive decisions. They might waste time searching through unorganized data (and potentially missing correct answers), make unnecessary and harmful mistakes, or even compromise the security of their data.

THE PROBLEM WITH "FLAT FILES" AND THE ADVANTAGES OF DATABASES

To fully understand the "why" of databases, it will be helpful to consider the main alternative to them: the so-called **flat file** or **"file system"** approach. In this approach, we collect **records** (information about a particular object or entity) into **files.** Which information gets stored in which record, and which record goes into which file, is determined entirely by what *use* the file's creator foresees (by contrast, databases lend themselves well to *multiple* uses). Examples of this approach would be.

1. Writing down (or typing) information on paper, and storing papers together (using things like file folders and fine cabinets). A small business, for example, might have a "file" for employees, a "file" for customers, a "file" for tax information, etc.
2. Doing the same thing electronically by storing information in **word-processing documents** or **spreadsheet files.**

Databases vs. File Systems. Many, many people (probably too many people!) use file systems to organize their information. However, database have significant advantages over file systems, at least in many contexts:

1. **Databases present users with a single "logical file."** While the data in databases might be spread over many locations (and different drives, computer files, etc.), the DBMS that users interact shows them the data as a single "unit." Many of the advantages below stem from this.
2. **Databases are "self-describing."** Databases allow us to easily encode **metadata** that is stored separately from the data itself. So, for example, a music database will "know" that "Taylor Swift" refers to an artist, and that artists are *related* in certain ways to songs and albums. In file systems, by contrast, there is no way to formally "encode" this sort of information.
3. **Databases minimize data redundancy and ensure consistency. They enforce transactions.** In a well-structured database, we should store a given piece of data (such as a person's current bank balance) exactly ONCE. This not only saves space, but minimizes the possibility for data inconsistency (for example, when we have multiple *different* values for a balance in different files.). This allows of easier support for **transactions** (for example, making sure that a user's bank balance *accurately* reflects all recent activity, including transfers, purchases, etc. even when these processes occur **concurrently**).
4. **Databases allow for (much) quicker answer to ad hoc queries.** A **query** is a question that "ask" about our data, while an **ad hoc query** is one that we ask "on the fly" (e.g., we'd like an answer quickly, and don't want to refer it to a librarian or other data expert). In a flat file, it can take a LONG time to find the data we are looking for, and even longer to "put it together" with other needed data. Databases make it much easier to ask questions (we can use SQL!) and much faster to get our answer.
5. **Data independence.** A database allows us to separate (1) the ways in which users (such as programmers, business analysts, researchers, medical staff, or even visitors to a website) *use* the data from (2) the way the data is stored. This is really important, because it means we can *change* the way the data is stored without causing problems for users. Flat files are *terrible* at this, and even small changes to things like spreadsheets can cause lots of problems for programmers and other users.
6. **Data can be shared between users, with different views.** In an organization such as hospital, many people need to access the same data. However, they need very different things from data, depending on their role (e.g., physicians' use of data is different from IT staff which is in turn different from that of patient). Databases allow not only for shared access to data, but for the design of multiple views (what the user "sees" and "interacts with" the data) for different users.
7. **Data security and backup/recovery.** Databases make it much easier to control *which* users can access/change *which* data. We can also *log* changes as they happen, which in turn makes it possible to recover from crashes or other problems.

The main *drawback* of databases concerns the cost (in terms of time, expertise, and money) to get them set up and running. So, they might not make sense for all projects, particularly very small ones. However (as I hope I'll convince you), it isn't all *that* difficult to set up a simple database and take advantage of these benefits.

WHY GOOD DESIGN MATTERS

In order to take advantage of the power of databases, you need to have good **database design.** Database design will be a major subject of this course, and it is somewhere between a "science" (you need to *know* things, both about databases and the subject matter) and an "art" (it requires creativity, and the ability to see "connections" between things that others may not have noticed).

To see how database design matters, let's consider an example from fictional "Hogwarts school" attended by aspiring witches and wizards in the fictional *Harry Potter* universe. Dumbledore, an excellent wizard, but subpar database designer, decides to encode data on students, instructors, and the courses they are taking or teaching as follows:

Name	Course 1	Course 2	Course 3	Notes
Ginny Weasley	Potions (3 cr)	History of Magic (4cr)	None	
Harry James Potter	Transmutation (4)	Potions (3 cr)	Defense Against Dark Arts	
Severus P Snape	Potions (three credits)			Teacher
Minerva McGonagall	Transfiguration (3)			Teacher
Hermione Granger	Hist of Magic (4cr)	Potions (3 cr)	Transfiguration (3)	TA for Potions

He keeps track of other student and faculty info (such as contact info for students' guardians or faculty salaries) in separate files. There are several things wrong with this table, including the following:

- **The data can't easily be searched or sorted in meaningful ways.** The fact that "name" contains both first and last name (and sometimes contains middle or last name) makes it difficult to easily sort by things like last name. It's also tough to search for things like "which classes are worth three credits?" or "how many classes is Professor Snape teaching right now?"
- **The database isn't easy to expand.** There are only three columns (or **fields)** for courses! What happens if someone wants to take a fourth class? Adding a fourth column for "class 4" is more painful than it might seem, since it will require that everyone who *interacts* with the database make adjustments to this.
- **There are data redundancies and (because of this) increased chance for inconsistencies and errors.** The fact that class names and credit information is *repeated* can cause problems. In particular, users (such as Dumbledore) may (unintentionally) enter the "same" data in different ways (for example, "History of Magic" vs "Hist of Magic"), which can lead to bad query results down the road. There's also an increased chance for error if some of the copies aren't updated correctly. For example, the Potions class is three credits every place but one.

A better solution might be to break down the data into (at least) three separate tables: **Person, Course** and **Enrolled.** I've shown parts of these tables below.

SELECT * FROM Person

Id	First	Last
1	Ginny	Weasley
2	Harry	Potter
3	Severus	Snape
4	Minerva	McGonagall
5	Hermione	Granger

SELECT * FROM Course

Id	Title	Credits
1	Potions	3
2	History of Magic	4
3	Transmutation	4
4	Defense Against the Dark Arts	3

SELECT * FROM Enrolled

person_id	course_id	role
1	1	3
1	2	3
3	1	1
5	1	2

Here, we might define "role" to be 1 = "teacher", 2 = "TA", and 3 = "student."

REVIEW QUESTIONS AND ACTIVITIES

1. Without looking back at the notes, see if you can define the following terms
   a. **Data, information,** and **knowledge**
   b. **Database** and **database system**
   c. **Metadata** and **end-user data**
2. Name THREE institutions (such as governments, companies, or medical providers) that likely have a database containing your information. What problems could occur if your data was not maintained or was incorrect?
3. Give an example of a time you have used a "flat file" or "file system" to store data. Might have a database worked better for this? Why or why not?
4. The above Hogwarts database is *very* simple. What other sorts of data might we want to include in a school database? What sorts of database "tables" might we use?