LECTURE 30

DATABASES

MCS 275 Spring 2021

Emily Dumas

LECTURE 30: DATABASES

Course bulletins:

- Project 3 grades expected by end of day on Wednesday.
- Worksheet 11 to be posted today.

DATABASE WEEK

This week we'll learn about databases, a language for writing database queries (SQL), and the use of the SQLite database in Python.

We'll only scratch the surface; you can spend a semester on this topic. E.g. CS 480 covers database design in detail, and MCS 565 covers theoretical aspects at the graduate level.

DATABASES

A **database** is a storage and retrieval system for structured data, usually in a persistent storage medium.

This is such a general definition that it sounds similar to other concepts like a data structure. What's the difference?

A database is defined by the way it can be used (storage and retrieval). In contrast, a data *structure* (like list, dictionary, BST) refers to the way data is arranged.

E.G. CSV?

A CSV file could be seen as a database in this broad sense. But to retrieve, modify, and save data in such a file, you need to do all the work yourself.

The term database is most often used when the storage and retrieval is handled by a separate piece of software—a **database management system** (DBMS).

When you use a DBMS, you let it handle the storage details. You interact with the data using a set of commands the DBMS supports.

E.G. CSV MODULE?

Python's CSV module could almost be seen as a DBMS, but one that makes changing data very cumbersome. How do you update one row of data in a CSV?

Read file, change data, delete file, write the new CSV? (Fancier file IO tricks with `.seek()`?)

Most DBMS would have a single command to update a record.

TYPICAL DBMS FEATURES

- Find and return data based on criteria
- Insert new data or make new collections
- Update (change) existing data
- Distribute storage across multiple servers
- Receive requests and return data over a network

DBMS command usually express intent (e.g. find X, change Y) rather than implementation details (e.g. the loops, data structures, algorithms, IO operations).

RELATIONAL DATABASES

A **relational database** organizes data into rectangular tables, called **relations**. It is the most common type.

Each column of a table has a name (e.g. "year") and a type (e.g. 16-bit unsigned integer).

A row in a table, called a **tuple** in DB terminology, is the basic unit of information you work with. The values in a single row refer to different attributes of some item entity (e.g. a book, an event, a user, a song, ...).

TABLE EXAMPLE

A table in a relational database might be used to store tasks in a to-do list application.

task_id	**description**	**t_added**	**t_due**	**done**
integer	*string*	*float*	*float*	*boolean*
964	Enjoy spring break	22	47	True
971	Write Lecture 30	22	47	True
973	Write Lecture 31	46	53	False
978	Write Project 4	46	66	False
408	Finish novel	3	null	False

RDBMS

Some widely-used relational database management systems (RDBMS):

- Open source
  - PostgreSQL
  - MySQL
  - SQLite
- Proprietary
  - Microsoft SQL Server
  - Oracle
  - IBM DB2

UIC EXAMPLES

UIC stores your course registration and final grades in an Oracle database.

Blackboard stores all course-related data (e.g. grade center) in a Microsoft SQL Server Database.

UIC's web site (uic.edu) uses Wordpress, which stores site content in a MySQL database.

I store some research data in a PostgreSQL database.

ASIDE: (NOT R)DBMS

MongoDB, Cassandra, Redis, Couchbase are popular examples of database management systems that use a non-relational structure.

For example, Redis is a key-value store, like a persistent version of Python's dict (with lots of additional features). It is often used for highly concurrent, latency-sensitive applications.

SQL

Structured Query Lanuage or **SQL** is a language for making requests (queries) to a RDBMS.

(Pronounced "ess kyoo ell" or "sequel".)

Think of SQL as a data request programming language. Most RDBMS use SQL exclusively, so if you want to ask a RDBMS to do something, you need to write it in SQL.

Learning SQL is learning a new programming language. We'll only cover the basics in MCS 275.

SUMMARY SO FAR

"Database" has a rather general meaning, but outside of theoretical CS it usually refers to a relational database management system (RDBMS) where you access and store tabular data using a special language called SQL.

KEY SQL COMMANDS

- **SELECT** finds rows in a table that match given conditions and returns them
- **INSERT** adds a new row to a table
- **UPDATE** modifies rows in a table (e.g. change the data in a single existing row)
- **CREATE TABLE** adds a new table
- **DROP TABLE** deletes a table

SQLITE

SQLite is an open source RDBMS that stores an entire database in a single file.

It consists of a standalone program where you can run SQL commands in a REPL environment, as well as libraries for most popular programming languages.

It was created by D. Richard Hipp in 2000 and is in the public domain. It is incredibly widely used (e.g. every Android or iOS mobile device, every copy of Windows 10, Chrome, Safari, Firefox). Many car stereos use it!

LITE?

The "lite" part of SQLite refers to its limited scope. It doesn't handle network requests or distributing data across multiple files, disks, or computers. But for many purposes it is a good choice.

If you want a database server that many programs will interact with concurrently, SQLite is not a good choice.

And any time you consider inventing your own file format to store a program's data, you should think carefully about whether CSV, JSON, or a SQLite database might be better.

SQLITE IN MCS 275

SQLite is the RDBMS we cover in MCS 275 because the `sqlite3` module is part of the Python standard library, and it can be used with a minimal amount of setup. A single file contains all the data.

In comparison, MySQL or PostgreSQL require installing and configuring both server and client software, creating user accounts, distributing credentials, having active network connections, etc..

NEXT TIME

We'll start working with SQLite databases in Python (and in the SQLite shell), focusing on learning basic SQL commands.

REFERENCES

- SQLite home page
- sqlitetutorial.net has a nice tutorial where you can run SQL command directly in your browser. Their SQLite install instructions are detailed and easy to follow, too.
- *Intro to Python for Computer Science and Data Science* by Deitel and Deitel, Section 17.2. (This is an O'Reilly book, free for anyone with a UIC email; see course page for login details.)
- *Computer Science: An Overview* by Brookshear and Brylow, Chapter 9.

REVISION HISTORY

- 2021-03-29 Initial publication
- 2021-04-07 Add info about database courses at UIC