

# LECTURE 27

## SQL AND SQLITE

MCS 275 Spring 2023

David Dumas

# LECTURE 27: SQL AND SQLITE

Reminders and announcements:

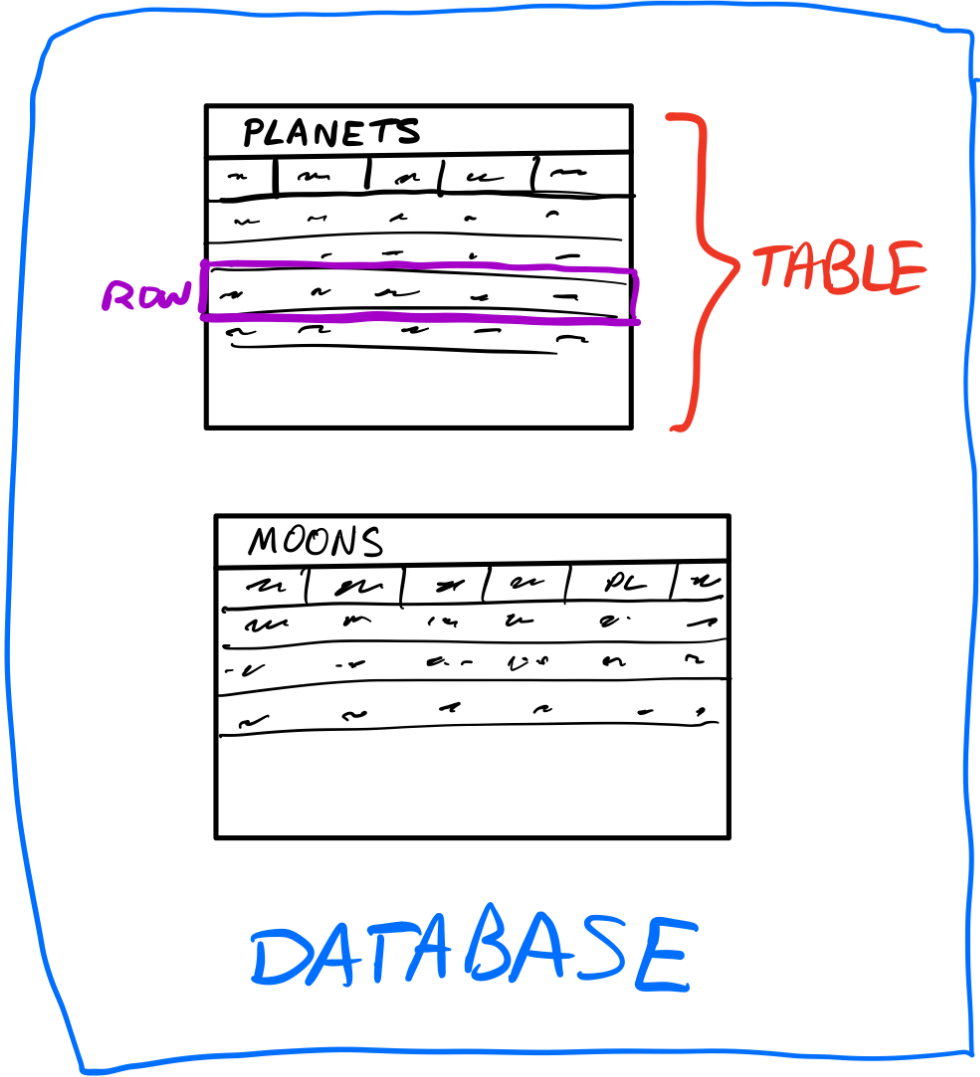
- Project 3 grading underway
- Project 4 coming soon
- Homework 10 due tomorrow

# DATABASE

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

Usually this term is used when the system offers a rich command language.

We'll only cover **relational databases**, which are based on collections of **tables**.



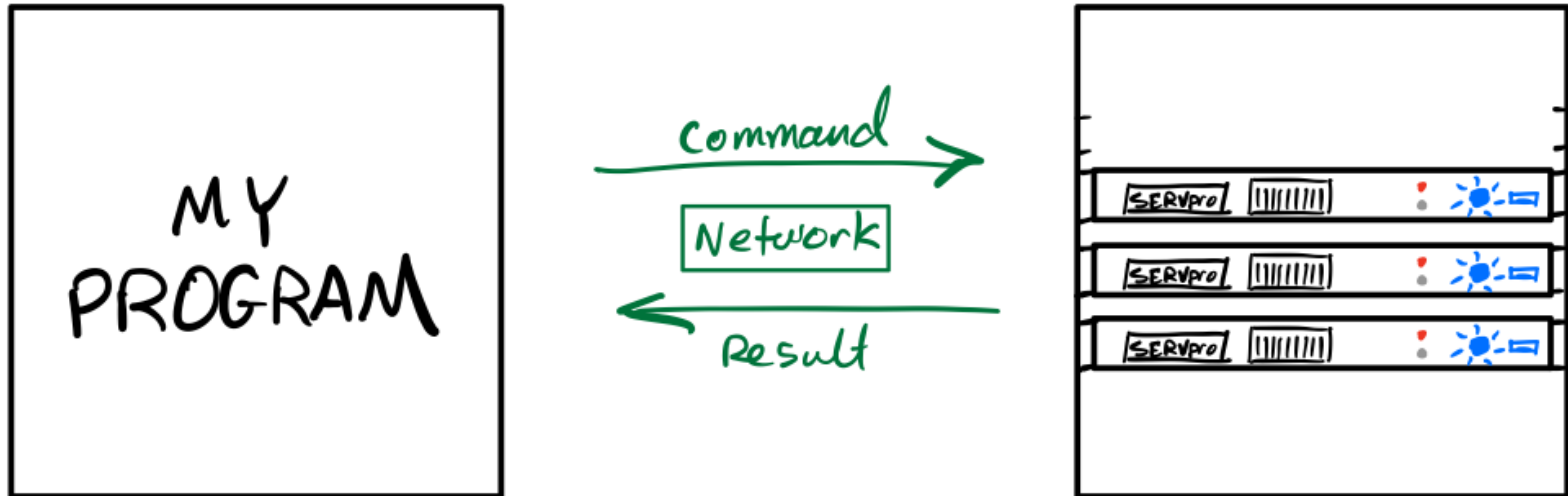
# TYPICAL RELATIONAL DB FEATURES

- Add, delete or modify tables
- Search a table for rows meeting given criteria
- Add or update rows

DB commands usually express intent (find and remove all rows with this property...), whereas file IO modules operate at a lower level (get the next line of text, ...) requiring you to build the required operations.

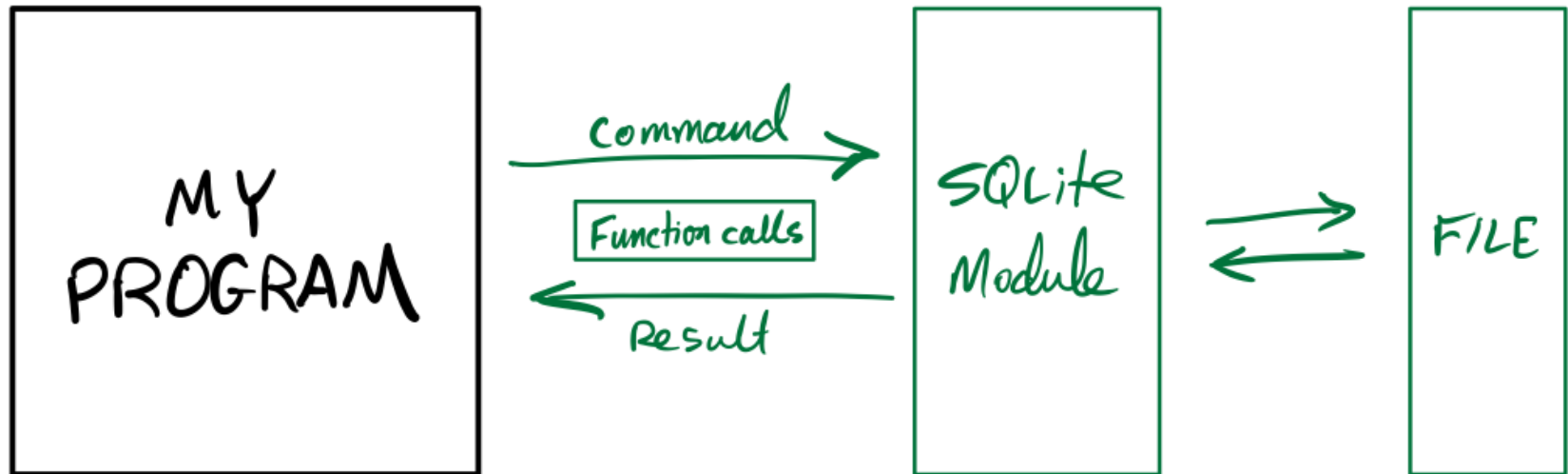
# APPLICATION STRUCTURE

TYPICAL



# APPLICATION STRUCTURE

WHEN USING SQLite



# SQLITE

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

It uses the same command language as many other popular databases: The **Structured Query Language**, or SQL. (Some people say "sequel".)

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



# DUAL GOALS

- Learn how to use the `sqlite3` module
- Learn enough SQL to make it useful

Learning a new language is challenging!

# SQLITE COMMAND LINE SHELL INSTALLATION

- **Windows** — Download the "sqlite-tools" zip file for windows from [the sqlite download page](#). Extract it to get "sqlite.exe". Put it somewhere you can find in the terminal and run it from powershell.
- **MacOS** — You already have it as `/usr/bin/sqlite3`. You can probably just type `sqlite3` in a terminal.
- **Linux** — You may already have it (try `sqlite3` in a terminal), otherwise use your package manager to install. In Ubuntu that package is called `sqlite3`.

# SQLITE COMMAND LINE SHELL

Then you use it as one of:

```
sqlite3 DBFILENAME  
sqlite3.exe DBFILENAME
```

If the file exists, it is opened in the SQLite REPL.

If it does not exist, it is created and opened in the SQLite REPL.

# SQLITE HELLO WORLD

Let's write a Python program to make SQLite database, add one table to it, add a couple of rows of data to the table, then read them back.

# CONNECTING TO A DATABASE

In `sqlite3`, opening a "connection" means opening or creating a database file.

```
import sqlite3
con = sqlite3.connect("solarsystem.sqlite") # .db also popula
con.execute( ...sql_statement_goes_here... )
con.commit() # Save any changes to disk
con.close() # Close the database file
```

# CREATE TABLE

```
CREATE TABLE planets (  
    name TEXT,  
    dist REAL,  
    year_discovered INTEGER  
);
```

Each item in parenthesized list has the form:  
column\_name COLUMN\_TYPE

# SAFER VARIANT

```
CREATE TABLE IF NOT EXISTS planets (  
    name TEXT,  
    dist REAL,  
    year_discovered INTEGER  
);
```

Now it's not an error if the table already exists.

# INSERT

The `INSERT` command adds a row to a table.

To pass values to a statement in `execute()`, use `?` characters as placeholders and then give a tuple of values in the second argument.

```
con.execute(
    "INSERT INTO planets VALUES (?, ?, ?);",
    ("Earth", 1.0, None)
)
```

Similar for `("Neptune", 30.1, 1846)`.



# PLANETS

Name	Distance from sun (AU)	Year discovered
Mercury	0.4	?
Venus	0.7	?
Earth	1	?
Mars	1.5	?
Jupiter	5.2	?
Saturn	9.5	?
Uranus	19.2	1781
Neptune	30.1	1846

# PLACEHOLDER GOTCHA

When calling `execute()` with placeholders in the SQL statement, the second argument **MUST** be an iterable of values.

So if you have only one value, you need to wrap it in a list or tuple.

```
con.execute("INSERT INTO tab VALUES (?);", 275) # FAILS
con.execute("INSERT INTO tab VALUES (?);", [275] ) # OK
con.execute("INSERT INTO tab VALUES (?);", (275,) ) # OK
```

These examples assume `tab` is a table with just one column.

# SELECT

Find and return rows. The most common query!

```
SELECT * FROM table_name; -- give me everything
SELECT * FROM table_name WHERE condition; -- some rows
SELECT col3, col1 FROM table_name; -- some columns
SELECT * FROM table_name LIMIT 10; -- at most 10 rows

SELECT * FROM table_name
ORDER BY col2; -- sort by col2, smallest first

SELECT * FROM table_name
ORDER BY col2 DESC; -- sort by col2, biggest first

SELECT DISTINCT ... ; -- no repeat answers
```

# SQL CONDITIONS

Examples of things that can appear after WHERE:

```
col = value    -- Also supports >, >=, <, <=, !=  
col IN (val1, val2, val3)  
col BETWEEN lowval AND highval  
col IS NULL  
col IS NOT NULL  
stringcol LIKE pattern    -- string pattern matching  
condition1 AND condition2  
condition1 OR condition2
```

# LIKE

```
coursetitle LIKE "Introduction to %"  
itemtype LIKE "electrical adapt_r"
```

In a pattern string:

- % matches any number of characters (including 0)
- \_ matches any single character

e.g. "%d\_g" matches "fossil dig" and "dog"  
but does not match "hypersonic drag", "dog  
toy", or "dg".

# WOBL

WHERE, ORDER BY, LIMIT can be used together, but must appear in that "WOBL" order. ([Details.](#))

# UPDATE

Change values in a row (or rows).

```
UPDATE table_name SET col1=val1, col5=val5 WHERE condition;
```

Warning: Every row meeting the condition is changed!

Also supports ORDER BY and LIMIT.

# DELETE

Remove rows matching a condition.

```
DELETE FROM table_name WHERE condition;
```

Also supports ORDER BY and LIMIT (e.g. to remove n rows with largest values in a given column).

Immediate, irreversible. Also, an empty table isn't the same thing as a deleted table.



# DROP TABLE

Deletes an entire table.

```
DROP TABLE table_name;           -- no such table = ERROR  
DROP TABLE IF EXISTS table_name; -- no such table = ok
```

Immediate, irreversible.

# INITIAL STATE

SQLite creates the database file if it doesn't exist, but with no tables or data in it.

Common bug:

- Program looks in the wrong place for a DB file
- It creates an empty DB but fails to work (no tables!)
- You try to debug but notice there's a DB file in correct place with correct name

# REFERENCES

- [SQLite home page](#)
- [sqllitetutorial.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.)
- [Getting Started with SQL](#) by Thomas Nield is a nice introduction to SQL that focuses on SQLite. It's another O'Reilly book you can access with your UIC email.
- Computer Science: An Overview by Brookshear and Brylow, Chapter 9.

# REVISION HISTORY

- 2022-04-15 Last year's lecture on this topic finalized
- 2023-03-26 Updates for 2023

