

Database basics and SQLite

15-113 – Monday 3/9

Quick poll on Project 2

- Fill this out: <https://bit.ly/4ujwcRk>
- Then talk with your neighbors about what you made, what you learned, what worked well or didn't work well.
- We'll revisit Project 2 a little more once we've graded them and seen your presentations
- Note for posterity: I "beautified" this



Quick poll on Project 2

- Fill this out: <https://bit.ly/4ujwcRk>
- Then talk with your neighbors about what you made, what you learned, what worked well or didn't work well.
- We'll revisit Project 2 a little more once we've graded them and seen your presentations
- Note for posterity: I "beautified" this



Updates!

- Project 2 presentations are this week! Watch Ed and your inboxes carefully for details!
- The TAs will *not* have office hours this week due to Project 2 presentations. Mike *will* have office hours, but might change Thursday's time to be a little later if it's more accessible to you
- Software developer and CMU alumnus Kian Nassre will be guest-lecturing **Wednesday**. Please be on time!

Onward to databases

Organizing data efficiently

First, consider storing some info about books in a list, like this:
(Assume the list entries are always in the form of title, author, rating)

```
[ ('Dune', 'Herbert', 5),  
  ('Neuromancer', 'Gibson', 4),  
  ('Foundation', 'Asimov', 5),  
  ('Snow Crash', 'Stephenson', 3) ,  
  ('I Robot', 'Asimov', 5), ]
```

What's the Big O for finding all books by a certain author?

Can we use dictionaries to make this faster?

Some problems with dictionaries

Sure, we could make a dictionary where the keys are the author...

- ...but we'll have to be careful if there are multiple books by the same author...
- ...so maybe we make a nested dictionary for each author keyed on books...
- ...but what if we also want to be able to find info on a specific quickly? Do we make a second dictionary with the title as the key?

Imagine we have several dictionaries, and we have information we want to update. How eager are you to make sure your code updates all the dictionaries properly? And remember to keep your keys immutable!

How do databases fix this?

- Databases also use keys for each entry, but they organize those entries in trees. Imagine a binary tree for simplicity. Lookup is $O(\log N)$, which is slower than $O(1)$, but most other operations are also $O(\log N)$ or better if we manage them properly.

How do databases fix this?

- Databases organize entries using using **trees** (e.g., binary trees).
- Lookup is **$O(\log N)$** .
- Slower than $O(1)$ dictionaries, but...
- Most other operations are also **$O(\log N)$** or better.



Database benefits

- Scalable and **persistent** data storage and organization
 - Writes survive crashes and are all-or-nothing
- Relatively efficient even at large scale (and in some cases with many concurrent users)
- Can be **queried** with a language called SQL, which (somewhat) mimics human language for asking questions about the data
 - Query planners figure out the optimal execution path so you don't have to

An important distinction

- **SQL is a query language** that allows us to format requests to read and write data
- **SQLite is a lightweight local database management system** built into Python.
 - It is **serverless** which means it's very easy to run, though because it's local, it can't be updated by many users simultaneously, and updates have to happen one at a time
 - This won't really be a problem for most (or any) of you, unless you're dealing with *lots* of users and *lots* of data

CRUD

- **CRUD** stands for:
 - **C**reate
 - **R**ead
 - **U**ppdate
 - **D**eleate
- These are the fundamental operations that may be necessary for a database. Your assignment this week should do each of these. Let's go look at that now

Let's see some example code

We'll share this on the website after class