

Kicking off 114 with (vector) databases

15-113 – Monday 10/19

NOTE: WE ARE UPDATING HW6!

We just decided to make some changes to HW6 to make it more interesting, so hold off on starting just yet. We'll have a new writeup released before Wednesday.

Welcome back (to 15-114)

- I hope you all had a great break!
- This half of the semester will have a similar workload, but will be a little more technical / will require more thinking at the code level
- We need to reset a few class norms:
 - Please try to be on time
 - When we share our work, you need to actually talk to your classmates
 - Stay focused in lecture and participate / ask questions
- You'll learn more if you do these things (and you won't be able to get an A without doing these things)

Quick poll on Project 2

- Fill this out:
<https://tinyurl.com/y4u44fj7>
- Then talk with your neighbors about what you made, what you learned, what worked well or didn't work well.
- Really really, please talk with each other



...For those of you who were late

- We need to reset a few class norms:
 - Please try to be on time
 - When we share our work, you need to actually talk to your classmates
 - Stay focused in lecture and participate / ask questions
- You'll learn more if you do these things (and you won't be able to get an A without doing these things)
- Also... don't get lazy with reading the writeups. *Really* don't get lazy with reading the stuff you hand in to us.

Embarrassing stuff we saw in several Project 2 READMEs...

AI-generated documentation

Codex drafted this README, including the technical explanations, setup instructions, and development summary. The author has requested a draft to review and rewrite; it should not yet be treated as a student-written final submission. The video link will be added after publication.

⚠ **TEMPLATE:** rewrite every section in your own words before submitting. The assignment requires the README to be "written yourself, in your own words." The bullets are reminders of what each section must cover. Delete this box when done.

AI-generated

This README was drafted by Grok 4.7 in Cursor on October 7, 2026, from the repository and `prompt_log.md`. It was not written by hand. The course asks for the overview, the features you are most proud of, and the AI summary to be in your own words. Rewrite those parts before you treat this file as your submission. The prompt log stays in `prompt_log.md` and is not copied here.

What it does

TODO: your own words.

How to use it

TODO: your own words.

Features I'm most proud of

TODO: your own words.

It's fine. It's fine! We can still fix this.

- Just like... read your stuff, pals!
- All of those examples (and at least half a dozen others) are still up on Github and public / probably linked from your portfolio.
 - It might not be a great look to potential employers, so at least change the bits that indicate you didn't read your own docs.
- Also, write the stuff we ask you to write yourself. Don't just feed the whole assignment to an AI and tell it to sound human. Authentic but rough is better than inauthentic and polished.



But the projects themselves were mostly very cool!

- Great to see your creative visions here
- Like... look at this one:
<https://kewuan-lyk.github.io/-Jack-Li-MusicUniverse/>
- We'll make a compilation video and gallery for the first half of the semester. It's not done yet!

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 **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

<https://github.com/mtaylor-create/sqliteBasics/blob/main/exampleCode.py>

Pivoting to Vector Databases

15-113 – Monday 10/21

Eberly focus group

- Lock in! This is a chance for you all to provide some valuable feedback on 113 and 114, and it's also a good way to reflect on whether you're engaging with the course in a way that's helping you achieve your own goals
- No attendance code today. Instead... if you collectively participate actively and eagerly with the Eberly team, we'll count this as attendance and participation for everyone, so don't be shy and share your thoughts

Project demos?

- Anybody willing to get up here and show us your Project 2?
- Whatever you show us, we'll almost certainly love it and want to know how you did it.
- Let's watch the compilation video (if it's ready, and if not, we'll share it very soon)

Pivoting to vector databases

- We've taught you a little bit about classic databases, but rather than having you do a rather cut-and-dry assignment with CRUD, let's talk about vector databases
- These are highly relevant to AI (in some ways that we'll discuss more directly on Monday) and we suspect we'll get more unique projects this way
- READ THE NEW HW6 WRITEUP! It's a little similar, but fundamentally new in some ways

Vector databases in a nutshell

- Instead of storing entries using a key and searching using columns/fields, vector databases create an *embedding*
- An embedding is a list of numbers (a vector) that encodes the *semantic meaning* of a piece of text
- For a particular piece of text in the database, you can imagine that it occupies a point in some N-dimensional space

A toy example

- Embeddings are long lists with several hundred dimensions, and the dimensions don't necessarily have human-understandable meanings, but...
- Imagine we're creating just a two-dimensional embedding for food, where each number corresponds to:
[numberOfIngredients, crunchiness]
- Let's draw this up and populate it with some foods

This won't necessarily put similar items next to each other but...

- Are fresh veggies generally near each other?
- Are a bunch of desserts like tiramisu and pudding and cake close to each other?
- How about different types of sandwiches?
- Some dissimilar things might also be near each other. If we were to add another dimension, what might separate those from each other? (Maybe sweetness? Cooked or uncooked?)

The more dimensions we add, the more clustered different words become

- With hundreds of dimensions, we can even encode content similarity of larger pieces of text, like sentences, paragraphs from books, job adds, or product reviews
- Imagine a big vector database full of quotes. How might this be useful?

Find stuff that's similar to _____

- I want to find a Kobe Bryant quote that says something about “I’ve got self-doubt and insecurity and sometimes I don’t want to play but I don’t let it beat me and I acknowledge it instead.”
 - So... imagine we calculate its embedding
- Here’s the actual quote: “I have self-doubt. I have insecurity. I have fear of failure. I have nights when I show up at the arena and I’m like, my back hurts, my knees hurt. I don’t have the energy. I just don’t want to play. We all self-doubt. You don’t deny it, but you also don’t capitulate to it. You embrace it.”
 - We’d expect this would have a pretty similar embedding, right?

When we search...

- The vector database finds the **k nearest neighbors** to our embedded query within the embedding space. Then we get the k most similar entries, along with a similarity score.
- We'd probably also get Naomi Osaka: "I have a lot of self-doubt. I think I'm very insecure sometimes... I've learned to just acknowledge that feeling instead of fighting it or letting it consume me. It's okay to not be okay, but you still have to try and find a way forward."
- Maybe Phelps: "There were days where I didn't want to get out of bed, where I didn't want to go to the pool. I had so much self-doubt. But I realized that hiding it or letting it beat me didn't work. The breakthrough comes when you actually acknowledge it, embrace it, and say 'Okay, this is how I feel today, but I still have a job to do.'"

Semantic search is powerful

- Think about some cool projects you could do with this
- For HW6, we recommend using Python and a module called ChromaDB.
 - It handles embeddings without you having to use API keys or paid models
 - Everything runs locally
 - AI is generally able to create example code for this module (though sometimes it's out of date)
- One caution: there's a limit to the size of the text you can embed (both in terms of efficiency and in terms of embedding usefulness)

Chunk it up!

- If you're looking to store something like a collection of books, you probably want to store *pieces* of books instead
- Before you store stuff in the database, consider breaking the text into chapters or paragraphs and storing those instead
- Also... if you wanted to (for example) make something that would search the CMU student handbook:
 - It would be useless to create a database with just one entry that contains the whole handbook
 - What you really want is something that stores smaller pieces of the handbook so that you can fetch just the part that's relevant to the query

If time, let's roll an example