ADVANCED
DATABASE
SYSTEMS

Course Overview & Logistics

00

Andy Pavlo
CMU 15-721
Spring 2024

Carnegie Mellon University

WHY YOU SHOULD TAKE THIS COURSE

DBMS developers are in demand and there are many challenging unsolved problems in data management and processing.

If you are good enough to write code for a DBMS, then you can write code on almost anything else.

And people will pay you lots of money to do it…

[redacted] salesforce
[redacted] salesforce
[redacted] amazon
[redacted] Materialize
[redacted] [tile]DB

[redacted] snowflake
[redacted] HyPer
[redacted] databricks
[redacted] Google
[redacted] databricks

[redacted] amazon
[redacted] salesforce
[redacted] [ROCKSET]
[redacted] yugabyteDB
[redacted] snowflake

[redacted] RisingWave
[redacted] OTTERTUNE
[redacted] yugabyteDB
[redacted] databricks

[redacted] salesforce
[redacted] impira
[redacted] ORACLE
[redacted] databricks

[redacted] snowflake
[redacted] facebook
[redacted] OTTERTUNE
[redacted] NEON

[redacted] OTTERTUNE
[redacted] Google
[redacted] databricks
[redacted] yugabyteDB

COURSE OBJECTIVES

Learn about modern practices in database internals and systems programming for analytical workloads.

Students will become proficient in:
→ Writing correct + performant code
→ Proper documentation + testing
→ Code reviews
→ Working on a large code base

We will cover state-of-the-art topics.
This is **<u>not</u>** a course on classical DBMSs.

COURSE TOPICS

Data Formats, Encoding, Compression
Vectorized Execution + Compilation
Query Scheduling + Coordination
Parallel Join Algorithms
Networking Protocols
Query Optimization (!!!)
Modern System Analysis

BACKGROUND

I assume that you have already taken an intro course on databases (e.g., 15-445/645).

We will discuss modern variations of classical algorithms that are designed for today's hardware.

Things that we will **not** cover:
SQL, Relational Algebra, Storage Models, Memory Management, Basic Algorithms + Data Structures.

COURSE LOGISTICS

Course Policies + Schedule:

→ Refer to course web page.

Academic Honesty:

→ Refer to CMU policy page.
→ If you're not sure, ask me.
→ I'm serious. Don't plagiarize or I will wreck you.

OFFICE HOURS

After class in my office:
→ Mon/Wed: 3:30 – 4:30
→ Gates-Hillman Center 9019

Things that we can talk about:
→ Issues on implementing projects
→ Paper clarifications/discussion
→ How to get a database dev job.
→ How to handle the police

TEACHING ASSISTANT

Head TA: William Zhang

→ 3rd Year PhD Student (CSD)
→ He took this class in his sophomore year and crushed it so hard we had to keep him around.
→ Canadian (the good kind)
→ #1 Ranked Database Ph.D. Student at Carnegie Mellon University.

COURSE MAILING LIST

On-line Discussion through Piazza:
https://piazza.com/cmu/spring2024/15721

If you have a technical question about the projects, please use Piazza.
→ Don't email me or TAs directly.

All non-project questions should be sent to me.

GRADE BREAKDOWN

Reading Reviews (15%)
Lecture Notes (10%)
Final Exam (15%)
Semester-long Project (60%)

READING ASSIGNMENTS

One mandatory reading per class (👑). You can skip **three** readings during the semester.

You must submit a synopsis **before** class:
→ Overview of the main idea presented (3 sentences).
→ Key findings/takeaways (2-3 sentences).
→ Description of the system discussed and how it was modified/extended (1 sentence).
→ Workloads / benchmarks used in evaluation (1 sentence).

Submission Form:
https://cmudb.io/15721-s24-submit

☠ PLAGIARISM WARNING ☠

Each review must be your own writing.

→ You may **not** copy text from the papers or other sources that you find on the web.

→ You may **not** use AI tools to generate the summary.

Plagiarism will **not** be tolerated.

See CMU's Policy on Academic Integrity for additional information.

LECTURE NOTES

Each student will be assigned one class during the semester to write notes about the lecture's contents.

→ Summarize the key topics and material from the lecture.

→ You do **not** need to include ancillary discussions from student questions or when "Andy goes off the chain".

Notes will be available on the course website + CMU-DB's Github repository.

→ Latex templates and samples are available.

→ Must include paper citations (Bibtex).

→ You are allowed to use images from course slides.

LECTURE NOTES

Each student must submit their notes as a PR within **one week** (seven days) of the lecture.

See course administration spreadsheet for your assigned lecture date.
→ You are allowed to swap without notifying instructors.

You are allowed to use AI to assist with this.
→ Please include information about what tools you used to help future students.
→ We can provide video transcripts if needed.
→ **You are responsible for the contents of the notes.**

FINAL EXAM

Written long-form take-home examination on the readings and topics discussed in class.

We will provide the exam in class on the last day of the semester.

SEMESTER-LONG PROJECT

Students divide into groups and build a component of a cloud-native, OLAP DBMS for the entire semester.

→ Overarching design theme is adaptivity.
→ Core implementation must be in Rust.

The goal is to then "stich" these components together at next semester and have a working DBMS prototype.

→ Name to be determined. See the Pavlo Naming Method

SEMESTER-LONG PROJECT

Students will divide into groups of three.

Each group will be assigned to one of five topics.
→ They will have to research the component, develop an implementation/testing plan, and then build it.

Two groups will be assigned the same topic.
→ They will have to **collaborate** and **compete** against their rival group (*except the optimizer teams!*).
→ The class will decide at the end of the semester which component is the winner per group.

PROJECT TOPICS

Scheduler:
→ Responsible for sending query plan fragments to execute nodes and updating internal task state.

Execution Engine:
→ Executes physical plan operators to retrieve data, process tuples, and generate output. Single-node only.

Catalog Service
→ Internal database of data files and table schemas.

I/O Service & Distributed Cache:
→ Provides an access layer to retrieve and cache blocks from data files.

PROJECT TOPICS

Query Optimizer

→ Generate an efficient physical plan for a SQL statement using rule-based optimizations and cost-based search.
→ We have an existing optimizer prototype that we will use as the starting point for this component.
→ The two groups will work together on separate problems in the optimizer instead than competing!

All the other projects must be written from scratch.

→ You can take inspiration from existing systems (e.g., Apache Arrow DataFusion).

PROJECT MILESTONES

Project Proposal (January 31st)
Status Update #1 (February 28th)
Status Update #2 (April 1st)
Final Presentation + Code Drop (TBA)

PROJECT PROPOSAL

Each group must write a plan on how they will implement their component and give a f**five-minute** presentation to the class that discusses the high-level topic.

Each proposal must discuss:

→ Architecture and implementation overview of the project.
→ How you will test whether your implementation is correct.
→ What workloads you will use for your project.

PROJECT PROPOSAL

The proposal must also include a specification about the API that the component will provide.
→ Supported commands / operations
→ Input / output encodings
→ Status codes / error handling

The two groups building the same component **must** implement the same API.
→ Each group must designate a team member to act as the ***liaison*** to the other group.
→ The liaisons will agree on what API to use and give a five minute presentation to the class.
→ Liaisons may also need to coordinate with other groups.

PROJECT PROPOSAL

We will want to reuse open-source APIs where appropriate instead of reinventing everything.
→ Example: Apache Iceberg API
→ We will ignore authentication and security checks.

Using the same API per component ensures that implementations are (potentially) interchangeable.

It also enables groups to reuse high-level testing infrastructure via APIs.

STATUS UPDATES

Each group will give a **five-minute** presentation twice during the semester to update the class about the current status of their project.

Each presentation should include:
→ Current development status.
→ Whether your plan has changed and why.
→ Anything that surprised you during coding.
→ Testing code coverage
→ Demos are always hot!

STATUS UPDATES

As part of the status update, each group must provide a design document that describes the current state of your project's implementation:

- → Architectural Design
- → Design Rationale
- → Testing Plan
- → Trade-offs and Potential Problems
- → Future Work

Liaisons must also work together to develop a benchmark plan to compare implementations.

FINAL PRESENTATION

10-minute presentation on the final status of your project during the scheduled final exam.

Each group is required to provide performance benchmark numbers that compare their implementation against the other group.

→ We will provide CMU-DB dev machines for this evaluation.
→ Liaisons will be responsible for scheduling and running these benchmarks.

FINAL CODE DROP

A project is **not** considered complete until:
→ All comments from code reviews are addressed.
→ The project includes test cases that correctly verify that implementation is correct.
→ Source code contains clear documentation / comments.

☠ PLAGIARISM WARNING ☠

These projects must be all of your own code and writing. You may **not** copy source code from other groups or the web unless given permission from the instructors.

Plagiarism will **not** be tolerated.
See CMU's Policy on Academic Integrity for additional information.

NEXT CLASS

First Lecture: Modern Analytical Database Systems

→ Make sure you submit the first reading review!
https://cmudb.io/15721-s24-submit

→ We will also discuss more about project logistics.

Administrative Stuff:

→ Sign-up for a group (or mark yourself as a free agent)

→ Add your Github handle to spreadsheet.

→ Check your assigned lecture notes date.