COMP 7/8116: Advanced Database Systems

Xiaofei Zhang

Fall, 2025

E-mail: xiaofei.zhang@memphis.edu
Office Hours: Upon Request
TA: Cong Guo (cguo@memphis.edu)

Web: Canvas
Class Hours: 11:20–12:45 TR
Class Room: IC 115

Course Description & Objectives

This project-based seminar explores advanced topics in modern database systems through a mix of lectures and student-led paper presentations. In addition to core concepts in distributed databases, the course integrates emerging AI-driven techniques—such as learned indexes, ML-assisted query optimization, and vector databases—into hands-on projects. The goal is to equip graduate students with the principles, methods, and mindset needed for impactful careers in data science and data management research.

Required Materials

- Course notes and the reading list will be available on Canvas.

Prerequisites

Students are expected to understand the fundamentals of databases (e.g., COMP3115), algorithms (e.g., COMP4030), and operating systems (e.g., COMP4270) each at least at the level of an introductory course.

Workload & Mark Breakdown

Project (40%)

Students will work in groups (2–3 students) on a theme-based course project aligned with one of the research topics outlined below, each supported by curated readings and open-source systems for hands-on exploration:

1. **Learning-Based Index Structures** Example projects: implement and benchmark a learned index (e.g., RMI, PGM-index, ALEX) against traditional indexes; extend an existing learned index with support for updates; integrate a learned index into an open-source DBMS.
2. **ML-Based Query Optimization & Cardinality Estimation** Example projects: develop a learned cardinality estimator for multi-join queries; integrate an ML-based cost model into PostgreSQL; compare deep learning vs. traditional statistical models for query plan selection.
3. **Cloud-Native Database Systems & Auto-Tuning** Example projects: deploy and evaluate TiDB or CockroachDB under varying workloads; build an RL-based workload tuner; extend an open-source auto-tuning framework such as OtterTune for cloud DB optimization.
4. **Vector Databases for ML Applications** Example projects: implement a hybrid search pipeline (vector + keyword) using Milvus, FAISS, or Weaviate; optimize ANN index structures for specific embedding models; evaluate vector DB performance under large-scale workloads.
5. **LLM Serving Infrastructure** Example projects: design a retrieval-augmented generation (RAG) pipeline with vector DB backends; evaluate serving architectures for low-latency LLM queries; extend LangChain or LlamaIndex for domain-specific reasoning.
6. **Reasoning Systems** Example projects: implement chain-of-thought or tree-of-thought reasoning in LLM applications; integrate structured reasoning graphs into a question-answering system; benchmark reasoning strategies for complex query answering.

Projects are intended to be mini-research efforts that explore, implement, or extend key ideas in one of the themes. Example formats include (but are not limited to):

- Building a new application on top of an existing system.
- Extending or enhancing a feature in an open-source system.
- Implementing a new algorithm or module that replaces a core component (e.g., indexing, query optimization, scheduling).

Groups will meet with the instructor periodically to discuss progress. Theme selection and initial project ideas should be finalized after the first round of presentation.

Project Deliverables: project proposal & preliminary study (10%), project presentation (10%), project report (10%), and source code (10%).

Group Work Policy: To foster collaboration, the course uses a group project as the primary assessment. Each member must submit a statement detailing their own role and contributions, as well as those of their teammates. These statements will be used to evaluate individual effort, and the instructor may adjust grades based on demonstrated contributions.

Paper Presentation (20%)

Each group will select one theme topic from the list provided (available on Canvas) and prepare a research field presentation based on multiple papers within that theme. The goal is to demonstrate a comprehensive understanding of the field, not just a single work. While each group should read broadly within the theme, they may choose one or two papers to examine in greater

depth, explaining their techniques and contributions in detail. Please consult the instructor if you wish to include papers outside the provided list.

- Use slides for the presentation. The content should be your own, though you may incorporate materials from other sources (e.g., figures from the papers) with proper credit.
- Your presentation should address:
  - What is the problem domain for this theme?
  - Why is it important?
  - What makes it challenging, and why do naive/previous approaches fall short?
  - What solutions or techniques are proposed in the papers you read?
  - What are the common trends, differences, or open challenges across the papers?
  - How does this body of work relate to the broader database systems landscape?
  - In your view, what are the most significant contributions and potential impacts?
- If you are discussing a system, consider showing a brief demo of how to configure and run it.

Attendance (10%)

Attendance is mandatory for this course. Students are encouraged to actively participate in the Q&A session in class.

Assignment (30%)

There will be 3 assignments (10% each) with a mixture of written questions and coding tasks. Students can work with any programming language that they feel comfortable with. Note that it is the student's responsibility to make sure the submission is valid and readable.

Course Schedule (Tentative)

Week	Topic
1	Overview & Introduction
2, 3	Distributed query & transaction processing
4, 5	Research paper presentation
6, 7	P2P & Multidatabases
8	*Fall break*
9	Proposal & Preliminary study presentation
10, 11	HDFS & Spark & Streaming
12, 13	Graph processing and NoSQL systems
14, 15	Project presentation

Assessments & Grading

7000-level v.s. 8000-level

Students enrolled in 7000 and 8000 sessions will have the same amount of workload, but 8000-level students are expected to work on more challenging problems for the course project.

Grading Scale

We compute final letter grades in two ways and each student receives the higher of the two.

Fixed scale (absolute cutoffs).

A	94–100%	**A**−	90–<94%	**B**+	87–<90%	**B**	84–<87%	**B**−	80–<84%	**C**+	77–<80%
C	74–<77%	**C**−	70–<74%	**D**+	67–<70%	**D**	64–<67%	**D**−	61–<64%	**F**	<61%

Curved scale (top-percentage, approximately normal). We map *class percentiles* (based on the total score distribution) to letter grades using bins that approximate a normal curve:

A	top 7%	**A**−	next 9%	**B**+	next 14%	**B**	next 20%	**B**−	next 14%	**C**+	next 12%
C	next 10%	**C**−	next 7%	**D**+	next 4%	**D**	next 2%	**D**−	next 1%	**F**	remaining

Notes.

- Ties on percentile cutoffs are resolved in favor of the higher grade.
- For small classes, minor boundary adjustments may be made for fairness and consistency.
- The instructor may assign an **A**+ for truly exceptional performance or an **F** for lack of effort, independent of the curve, when clearly warranted in writing.

Course Policies

Plagiarism or cheating behavior in any form is unethical and detrimental to proper education and will not be tolerated. All work submitted by a student (projects, programming assignments, lab assignments, quizzes, tests, etc.) is expected to be a student's own work. Plagiarism is incurred when any part of anybody else's work is passed as your own (no proper credit is listed to the sources in your own work) so the reader is led to believe it is therefore your own effort. Students are allowed and encouraged to discuss with each other and look up resources in the literature, but appropriate references must be included for the materials consulted, and appropriate citations made when the material is taken verbatim.

If plagiarism or cheating occurs, the student will receive a failing grade on the assignment and (at the instructor's discretion) a failing grade in the course. The course instructor may also decide to forward the incident to the Office of Student Accountability for further disciplinary action. For further information on the UofM code of student conduct and academic discipline procedures, please refer to http://www.memphis.edu/studentconduct/misconduct.htm

Within this class, you are welcome to use foundation models (ChatGPT, GPT, DALL-E, Stable Diffusion, Midjourney, GitHub Copilot, and anything after) in a unrestricted fashion. However, you should note that all large language models still have a tendency to make up incorrect facts and fake citations, code generation models have a tendency to produce inaccurate outputs, and image generation models can occasionally come up with highly offensive products. You will be responsible for any inaccurate, biased, offensive, or otherwise unethical content you submit regardless of whether it originally comes from you or a foundation model. If you use a foundation model, its contribution must be acknowledged in the hand-in; you will be penalized for using a foundation model without acknowledgment. Having said all these disclaimers, the use of foundation models is encouraged, as it may make it possible for you to submit assignments with higher quality, in less time.