5303 Advanced Database

General Course Information

- **Days:** Monday / Wednesday
- **Time:** 4:00 p.m. – 5:20 p.m.
- **Location:** Bolin 248
- **Semester:** Monday August 23rd – Friday December 4th
- **Labor Day:** Monday September 7th
- **Thanksgiving Holiday:** Wednesday November 25th – Sunday November 29th
- **Last Day for "W":** Monday November 23rd
- **Last Day of Class:** Friday December 4th
- **Final Exam:** Monday December 7th, 3:30 p.m. – 5:30 p.m.

Course Zoom

https://msutexas-edu.zoom.us/j/9403974439

Course Description

Modern applications rarely have the luxury of asking simply, *"Which SQL database should we use?"* Instead, developers must choose among relational databases, document stores, key-value systems, graph databases, search engines, distributed databases, and increasingly systems that combine features from several of these categories.

This course is a survey of modern database systems with an emphasis on:

> **Understanding why different database models exist, how they differ, and how to choose an appropriate system for a particular problem**.

Rather than attempting shallow coverage of a large number of products, we will study representative systems from several major database families in enough depth to build applications, conduct experiments, and compare their behavior.

Our primary systems will include:

- **PostgreSQL** as our primary relational database management system;
- **SQLite** as an embedded relational database;
- **MongoDB** as our primary document-oriented database; and
- **Redis** as our primary key-value / in-memory data store.

Additional databases may be selected for comparison, student presentations, or individual projects.

The goal is not to determine which database is universally "best." There isn't one.

Instead, we will ask:

> **Which database model and implementation best fits a particular workload, application, and set of requirements?**

Course Philosophy

Traditional database courses often devote substantial time to relational algebra, relational calculus, normalization theory, and formal relational database design.

These topics are important foundations of database systems, and this course will cover the relational concepts necessary to understand modern relational databases. Students should understand relations, keys, functional dependencies, normalization, joins, constraints, and the principles underlying relational query processing.

However, this course is **not primarily a course in formal relational database theory**.

Our emphasis will instead be on the design, implementation, use, and evaluation of modern database systems.

This distinction allows us to study questions that become increasingly important when moving beyond a single database model:

- What does "schema" mean in a relational database versus a document database?
- When is normalization desirable, and when might deliberate denormalization make sense?
- How do joins compare with embedded documents or application-side relationships?
- How do indexes affect read and write performance?
- What guarantees does a database provide when several operations occur concurrently?
- What does ACID actually guarantee?
- Which guarantees are weakened, modified, or implemented differently by distributed systems?
- When does eventual consistency make sense?
- When is an in-memory key-value store a better solution than a relational table?
- How should databases be benchmarked fairly?
- What are the operational costs of running one database architecture instead of another?

The intent is to connect database theory to the engineering decisions developers actually make.

Learning Objectives

By the end of the course, students should be able to:

1. Explain the major characteristics of relational, document-oriented, and key-value database models.
2. Design and implement databases using multiple data models.
3. Write effective queries using SQL and non-SQL query interfaces.
4. Explain relational concepts including keys, relationships, normalization, joins, and referential integrity.
5. Explain transactions and the ACID properties of database systems.
6. Compare consistency, isolation, durability, and concurrency guarantees across database systems.
7. Design and evaluate indexes for different workloads.
8. Explain common approaches to replication, partitioning, caching, and distributed storage.
9. Measure database performance using reproducible experiments rather than anecdotal claims.
10. Identify tradeoffs among consistency, performance, scalability, complexity, and developer convenience.
11. Select an appropriate database technology for a given application and justify that decision.

12. Communicate technical findings through presentations, written documentation, and reproducible software projects.

Core Database Systems

The course will concentrate on a small set of representative systems.

1. Relational Databases

PostgreSQL — Primary RDBMS

PostgreSQL will serve as the primary relational database for the course.

Topics may include:

- database and schema design;
- SQL;
- tables and relationships;
- primary and foreign keys;
- joins;
- constraints;
- transactions;
- indexes;
- query planning and optimization;
- views and materialized views;
- stored functions and procedures;
- concurrency and isolation;
- JSON/JSONB;
- full-text search;
- extensions; and
- selected advanced PostgreSQL capabilities.

SQLite — Embedded Relational Database

SQLite provides an important contrast to PostgreSQL because it demonstrates that a relational database does not necessarily require a traditional database server.

We will examine:

- embedded databases;
- zero-configuration deployment;
- file-based storage;
- transaction behavior;
- concurrency limitations and strengths;
- appropriate SQLite workloads; and
- situations in which SQLite may be preferable to a client/server RDBMS.

---

2. Document Databases

MongoDB — Primary Document Store

MongoDB will serve as our primary example of a document-oriented database.

Topics may include:

- BSON documents;
- collections;
- flexible schemas;
- embedded documents;
- references;
- indexing;
- aggregation pipelines;
- replication;
- transactions;
- data modeling; and
- relational versus document-oriented design.

A recurring comparison will be:

> **MongoDB documents vs. PostgreSQL JSON/JSONB**

Rather than assuming that either approach is superior, we will examine workloads in which each model provides advantages.

Comparative Document Database

A second document-oriented system may be selected for comparison.

Possible systems include:

- Cloud Firestore;
- Couchbase; or
- another system selected because of current industry relevance.

---

3. Key-Value and In-Memory Databases

Redis — Primary Key-Value System

Redis will serve as our primary key-value and in-memory data system.

Topics may include:

- keys and values;
- expiration and TTL;
- caching;
- strings;
- lists;
- sets;
- sorted sets;
- hashes;
- persistence;
- transactions;
- pub/sub;

- streams;
- distributed caching; and
- performance characteristics of memory-oriented systems.

Redis also provides an opportunity to study the relationship between open-source software, commercial database products, licensing, and community forks.

Comparative Key-Value Database

A second system may be selected for comparison.

Candidates include:

- Valkey;
- DynamoDB;
- Dragonfly; or
- another current system appropriate to the course.

Additional Database Models

The primary goal of the course is depth with PostgreSQL/SQLite, MongoDB, and Redis rather than superficial exposure to every database product available.

However, several additional database models are important enough to examine conceptually or through student presentations.

Wide-Column Databases

Possible systems:

- Cassandra
- ScyllaDB
- HBase

Important concepts include partitioning, distributed writes, replication, high availability, and workloads involving very large datasets.

Graph Databases

Possible systems:

- Neo4j

Graph databases provide an opportunity to examine problems in which relationships themselves are first-class data.

Examples include:

- social networks;
- recommendation systems;
- dependency graphs;
- transportation networks; and

- knowledge graphs.

Search and Indexing Systems

Possible systems:

- Elasticsearch
- OpenSearch

These systems allow us to examine:

- inverted indexes;
- full-text search;
- ranking;
- distributed indexing; and
- the distinction between a primary database and a specialized search system.

Vector Search

Vector search has become increasingly important in machine learning and AI applications.

Possible systems or technologies include:

- PostgreSQL with `pgvector`;
- Redis vector search;
- Chroma;
- Weaviate;
- Pinecone; or
- other current vector-search technologies.

Vector databases also raise an important question for this course:

> When does a specialized database justify adding another database system to an application, and when is an extension to an existing database sufficient?

Major Course Topics

The semester will be organized around **concepts and problems**, rather than simply moving from one database product to another.

Database Models

- Relational databases
- Document databases
- Key-value databases
- Wide-column databases
- Graph databases
- Search databases
- Vector search

Data Modeling

- Relations and tables
- Documents and nested data
- Keys and relationships
- Normalization
- Denormalization
- Referential integrity
- Schema design
- Schema flexibility
- Data duplication and consistency

Querying

- SQL
- Joins
- Aggregation
- Document queries
- MongoDB aggregation pipelines
- Key-based access
- Search queries
- Query APIs

Indexing and Query Performance

- B-tree indexes
- Hash-based access
- Compound indexes
- Query planners
- Query execution plans
- Index selectivity
- Read/write tradeoffs
- Memory versus disk access

Transactions and Consistency

- Transactions
- Atomicity
- Consistency
- Isolation
- Durability
- Concurrency
- Isolation levels
- Locking
- Multi-version concurrency control
- Distributed consistency
- Eventual consistency

ACID will be treated as a **set of properties and guarantees to investigate**, rather than as a binary label dividing databases into "good" and "bad" systems.

Modern relational and non-relational databases provide different transaction capabilities, scopes, performance costs, and consistency guarantees. Understanding those tradeoffs is more useful than simply asking whether a

product claims to support ACID transactions.

Performance and Benchmarking

Students will investigate database performance experimentally.

Potential measurements include:

- insert throughput;
- query latency;
- update performance;
- indexed versus non-indexed queries;
- bulk operations;
- concurrent operations;
- memory consumption;
- storage requirements;
- cache performance; and
- scaling behavior.

A benchmark without a clearly defined workload is mostly just a number wearing a lab coat.

Therefore, students will be expected to document:

- hardware and software environment;
- dataset;
- database configuration;
- workload;
- indexes;
- number of trials;
- measurements;
- methodology; and
- limitations of their conclusions.

Comparative Database Study

Throughout the semester, we will return to a common set of questions.

Question	PostgreSQL	SQLite	MongoDB	Redis
What is the primary data model?	Relational	Relational	Document	Key-value / data structures
How is data organized?	Tables	Tables	Collections/documents	Keys and values
How is data queried?	SQL	SQL	MongoDB query language	Commands/API

Question	PostgreSQL	SQLite	MongoDB	Redis
How are relationships represented?	Keys/joins	Keys/joins	Embedding/references	Application-defined
What indexing mechanisms exist?	Multiple	Primarily B-tree	Multiple	Structure-dependent
What transaction guarantees exist?	Strong	Strong	Supported	Operation/transaction dependent
How is concurrency handled?	MVCC	File/database mechanisms	Database mechanisms	Primarily command execution model
How does persistence work?	Disk/WAL	Database file/WAL	Disk/journal	Memory + persistence options
What workloads fit naturally?	General-purpose	Embedded/local	Document-oriented	Cache/realtime/fast lookup
What are the major tradeoffs?	Complexity/operations	Concurrency/scale	Duplication/modeling	Memory/model limitations

The purpose of comparisons such as this is **not to produce a universal ranking**.

Database selection depends on workload.

Tentative Course Progression

The exact pace may change based on class progress, projects, and current database technologies.

Part I — Database Foundations

- Database models
- Relational model
- Keys and relationships
- SQL review
- Normalization and denormalization
- Indexes
- Transactions
- ACID
- Concurrency
- Query planning

Part II — Relational Systems

Primary systems:

- PostgreSQL
- SQLite

Students will build relational databases and investigate performance, transactions, indexing, and query execution.

Part III — Document Systems

Primary system:

- MongoDB

Comparisons may include:

- MongoDB vs. PostgreSQL;
- documents vs. normalized relations;
- MongoDB vs. PostgreSQL JSONB; and
- schema flexibility vs. schema enforcement.

Part IV — Key-Value and In-Memory Systems

Primary system:

- Redis

Possible comparisons:

- Redis vs. PostgreSQL;
- Redis vs. Valkey;
- Redis as cache vs. Redis as primary data store; and
- persistent storage vs. memory-oriented storage.

Part V — Distributed and Specialized Databases

Selected topics may include:

- replication;
- partitioning;
- distributed consistency;
- wide-column databases;
- graph databases;
- search systems;
- cloud databases;
- serverless databases; and
- vector search.

Part VI — Database Selection and Evaluation

Students will use the systems and concepts studied throughout the semester to answer the central question of the course:

> **Given an application and workload, what database architecture should we use, and why?**

Presentations

Each student will give **two technical presentations** during the semester.

Presentation topics will be assigned or approved by the professor and will generally involve a database technology, architecture, feature, or current development related to database systems.

Possible presentation topics include:

- PostgreSQL extensions;
- PostgreSQL vs. MySQL;
- PostgreSQL JSONB vs. MongoDB;
- SQLite and embedded databases;
- distributed SQLite;
- MongoDB;
- Firestore;
- Redis;
- Redis vs. Valkey;
- DynamoDB;
- Cassandra;
- ScyllaDB;
- Neo4j;
- Elasticsearch/OpenSearch;
- `pgvector`;
- vector databases;
- database licensing and open-source forks;
- cloud-managed databases;
- serverless databases; and
- emerging database technologies.

Presentations should do more than describe a product's feature list.

Students should attempt to answer questions such as:

- What problem does this technology solve?
- How does its data model differ from alternatives?
- What workloads favor it?
- What workloads do not?
- What guarantees does it provide?
- What are its performance characteristics?
- What are its operational costs?
- What competing technology could solve the same problem?
- Why would a developer choose one over the other?

Programming Projects

Programming projects will require students to build, query, measure, or compare database systems.

Projects may involve:

- constructing datasets;
- implementing equivalent schemas in different database models;
- generating workloads;
- benchmarking;
- indexing experiments;
- transaction experiments;
- concurrency experiments;
- caching;
- API development; and
- comparative analysis.

Projects must include sufficient documentation for another person to reproduce the work.

At minimum, submissions should contain:

- a README;
- setup instructions;
- dependency information;
- file descriptions;
- meaningful comments where appropriate;
- instructions for running the project;
- instructions for reproducing experiments; and
- a discussion of results.

A project that cannot be run cannot provide much evidence that it works.

GitHub Portfolio

Students will maintain a GitHub repository containing their work from the course.

The portfolio should demonstrate not only completed code but also professional technical documentation.

Repositories should be organized, readable, and reproducible.

Where appropriate, repositories should include:

- source code;
- database setup scripts;
- schema definitions;
- sample data or data-generation scripts;
- queries;
- experiment scripts;
- results;
- documentation; and
- presentation materials.

Credentials, passwords, private keys, connection strings containing secrets, and other sensitive information **must not be committed to GitHub**.

Participation

This course depends heavily on discussion, experimentation, troubleshooting, and comparison of results.

Participation therefore includes more than simply being physically present in class.

Examples of meaningful participation include:

- attending and participating in class;
- asking technical questions;
- contributing to class discussions;
- helping identify unclear instructions or unexpected behavior;
- discussing experimental results;
- helping classmates troubleshoot conceptual problems;
- participating in the class communication platform; and
- sharing useful discoveries with the class.

Questions are particularly valuable. A question asked by one student frequently identifies something that needs clarification for everyone.

Grading

Category	Weight
Presentations	30%
Programming Projects	30%
Final Exam	25%
GitHub Portfolio	10%
Participation	5%
Total	**100%**

Grade Scale

Grade	Percentage
A	90–100
B	80–89
C	70–79
D	60–69
F	Below 60

Final Exam

A comprehensive final examination will cover the major database concepts studied throughout the semester.

The examination will emphasize **understanding and comparison** rather than memorizing product-specific commands.

Students should be prepared to:

- explain database models;
- compare technologies;
- reason about transactions and consistency;
- evaluate indexing strategies;
- interpret performance results;
- discuss database design decisions; and
- recommend appropriate database architectures for particular workloads.

Course projects, presentations, discussions, and lectures will collectively provide the material necessary to prepare for the final examination.

The final examination will be administered at the officially scheduled university time.

Conflicts involving travel, airline reservations, weddings, vacations, or other personal scheduling should be resolved around the university examination schedule. Exceptions will be considered only when supported through the appropriate university process.

Late Work

Late work may be accepted on a case-by-case basis.

Unless otherwise specified:

- an initial late submission receives a **15-point penalty**;
- an additional **5-point penalty** is applied for each subsequent class period the assignment remains late;
- penalties may accumulate to a maximum reduction of **50 points**; and
- acceptance of extremely late work is at the instructor's discretion.

Students should communicate with the instructor as early as possible when circumstances may prevent timely completion of an assignment.

Technology Selection

Database technology changes quickly.

For that reason, some systems studied in this course may change from semester to semester.

Selection will favor technologies that are:

1. widely used or technically influential;

2. representative of an important database model;
3. accessible to students;
4. suitable for experimentation;
5. supported by useful documentation and development tools; and
6. interesting enough to teach us something about database design.

Popularity alone does not make a database good, and lack of popularity does not make one irrelevant.

The objective is to understand the **ideas represented by the systems**, not merely their current market position.

The Question Behind the Course

By the end of the semester, students should be able to look at a proposed application and resist immediately saying:

> "Let's use MongoDB."

Or:

> "Let's put everything in PostgreSQL."

Or, God help us:

> "I saw a YouTube video where they used Redis."

Instead, students should be able to ask:

- What does our data look like?
- How is it related?
- How will it be queried?
- How frequently will it change?
- What consistency guarantees do we require?
- What failures must we tolerate?
- How much data do we expect?
- What latency do we require?
- What operational complexity can we tolerate?
- What does this choice cost?
- What happens when the system grows?

Then—and only then—should we choose the database.

Official Course and Department Policies

Attendance Policy

Although student attendance is not calculated into the grade, attendance will be taken each day to track students. If a student is absent more than 2 classes without an excuse and is not performing well in class, a report can be submitted to the Dean of Students and the student may be dropped from the class. Classes will not be streamed for absent students, whether it is excused or not.

Behavior in the classroom

Students are to assist in maintaining a classroom environment that is conducive to learning. This means that the presence of electronic devices other than your calculator are not to be seen, heard, or implied, ever. Questions are encouraged and discussion is acceptable, provided it is pertinent and does not distract from the lesson.

Make Up Work/Exams/Quizzes:

- For planned excused absences: Exam may be taken early by prior arrangement.
- For unplanned documented absences: I may replace your exam grade with your final exam grade. I reserve the right to make that decision.
- For unplanned undocumented absences: Zero on the exam

Late Work

Late programs will be accepted with an initial **15 point** reduction and then a **5 point **deduction class day. ***No late programs for last programming assignment. No late homework will be allowed in the Connect System.***

Computer Requirements: Taking this class requires you to have access to a computer (with Internet access) to access online course material. ***Personal computer technical difficulties will not be considered a reason for extra time to submit assignments, tests, or online discussion postings.*** Computers are available on campus in various areas of the buildings, as well as the in the library. Contact your instructor immediately upon having computer trouble. If you have technical difficulties in the course, there is also a student helpdesk available to you. The university cannot work directly on student computers due to both liability and resource limitations, however they are able to help you get connected to our online services. For help, log into **D2L.**

Policy on Testing Process

The Department of Computer Science has adopted the following policy related to testing.

- All bags, purses, electronics (turned off), books, etc. will be placed in the front of the room during exams, or in an area designated by the instructor.
- Unless otherwise announced by the instructor, nothing is allowed on the desk but pen/pencil/eraser and test papers.
- A student who leaves the room during an exam must turn in the test and will not be allowed to return.

Policy on Programs

- Tests *will* have questions covering out-of-class assignments. Know the material!
- Students will be invited to orally answer questions regarding their assignments in my office and failure to answer those questions correctly will result in deductions from their grades. (Every student can expect to be invited 1-2 times during the semester to answer questions.)

Computer Science Tutoring

Tutoring is available in ***Bolin Room 119 & the Office of Tutoring and Academic Support Programs (TASP)*** in Moffett Library. A tutor may assist with programs and homework for computer science classes. The tutor will not do your work.

Academic Misconduct Policy & Procedures

Cheating, collusion, and plagiarism (the act of using source material of other persons, either published or unpublished, without following the accepted techniques of crediting, or the submission for credit of work not the individual's to whom credit is given). The Department of Computer Science has adopted the following policy related to cheating (academic misconduct). The policy will be applied to all instances of cheating on assignments and exams as determined by the instructor of the course. **(See below for link to MSU definitions.)**

- 1st instance of cheating in a course: The student will be assigned a non-replaceable grade of zero for the assignment, project or exam. *If the final grade in the course, does not result in a one letter grade reduction, the student will receive a one letter grade reduction in course.*
- 2nd instance of cheating in a course: The student will receive a grade of F in course & immediately be removed from course.
- All instances of cheating will be reported to the Department Chair and, in the case of graduate students, to the Department Graduate Coordinator.

Note: Letting a student look at your work is collusion and is academic misconduct!

See Also: MSU Student Handbook: Appendix E: Academic Misconduct Policy & Procedures *https://msutexas.edu/student-life/_assets/files/handbook.pdf*.

Recording of Class Lectures

Permission must be requested in writing and obtained from the instructor before recording of class lectures. If permission is granted, the recording may only be used by the student making the recording. Recordings (or any class materials) may NOT be posted on any internet source without written permission of the instructor. Failure to adhere to the policy may result in removal from the course with a grade of F or other appropriate punishment.

University Policies and Procedures

Student with Disabilities

Any student who, because of a disability, may require special arrangements in order to meet the course requirements should contact the instructor as soon as possible. Students should present appropriate verification from Disability Support Office during the instructor's office hours. Please note instructors are not allowed to provide classroom accommodations to a student until appropriate verification has been provided. For additional information, contact the Disability Support Office in Clark Student Center 168 - Phone: (940) 397-4140

Policy on Concealed Handguns on Campus

Senate Bill 11 passed by the 84th Texas Legislature allows licensed handgun holders to carry *concealed* handguns on campus, effective August 1, 2016. Please note, open carry of handguns, whether licensed or not, and the carrying of all other firearms, whether open or concealed, are prohibited on campus. Areas excluded from concealed carry are appropriately marked, in accordance with state law. For more information regarding campus carry, please refer to the University's webpage at MSU Campus Carry Policy *https://msutexas.edu/campus-carry/rules-policies*. If you have questions or concerns, please contact MSU Chief of Police Steven Callarman at Steven.callarman@msutexas.edu.

Midterm Progress Report

In order to help students keep track of their progress toward course objectives, the instructor for this class will provide a Midterm Progress Report for all students in the course through each student's MSU Portal account. Midterm grades will not be reported on the students' transcript; nor will they be calculated in the cumulative GPA. They simply give students an idea of where they stand at the midpoint of the semester. Students earning below a C at the midway point should a) schedule a meeting with the professor and b) Seek out tutoring.

Tobacco Policy

Midwestern State University seeks to provide a safe, healthy, pleasant environment for its faculty, staff, and students. To this end, the use of tobacco products, including smoke and smokeless tobacco, and the advertising, sale, free distribution, and discarding of tobacco products shall be prohibited in all indoor and outdoor facilities and in all university vehicles. The policy extends to faculty, staff, students, vendors, guests, and visitors.