UNI CS 3140 (Fall 2025)
Database Systems, Section 01

Course Syllabus

Course Information

Course Name and Number

CS 3140 Database Systems

Meeting Times

Lecture: MWF 1:00pm–1:50pm in CEE 115

Contact Information

(Lead Professor): Dr. Sarah Diesburg – sarah.diesburg@uni.edu
Office: CEE 010A
Office hours: MWF 8:30-10:00am and W 2:00-4:00pm in CEE 010A. My office hours are drop-in and do not require appoints, but those with appointments have first priority. You can make a 15-minute appointment at https://tinyurl.com/diesburg
Class website: Available through UNI eLearning: https://elearning.uni.edu

Credit Hours

3 Credit Hours

This course meets the Course Credit Hour Expectation outlined in the Course Catalog. **Students should expect to work approximately 2 hours per week outside of class for every course credit hour.**

Course Materials (Required)

Follow these directions to obtain access to your mandatory online textbook:

- Sign in or create an account at http://learn.zybooks.com
- Enter zyBook code: UNICS3140_5140DiesburgFall2025
- Subscribe

Course Description

Storage of, and access to, physical databases; data models, query languages, transaction processing, and recovery techniques; object-oriented and distributed database systems; and database design. Prerequisite(s): CS 1520; CS 1800; junior standing. Prerequisite(s) for Data Science minors: CS 2150; junior standing. (Fall)

Course Learning Outcomes

- Understand the role of a database management system in an organization.
- Understand basic database concepts, including the structure and operation of the relational data model.
- Understand and apply basic database design principles, including ER diagrams.
- Construct database queries using Structured Query Language (SQL).
- Compare and contrast data storage models.
- Design, implement, and program the interface to a database containing real-world data.

Performance Evaluation

Grade Determination

The final grade you earn in this course will be based on the points accumulated over five activities as described below.

Note: The final is optional and, if taken, will replace the lowest exam grade. Note that the final will replace the lowest exam grade, regardless if it is higher or lower than that grade.

Activity	Percent
Lab Work	15%
Zybook Participation Activities	10%
Zybook Challenge Activities	10%
Final Hands-on Project	20%
Regular Exams (3 regular exams @ 15% each)	45%
Final Cumulative Exam (replaces lowest regular exam)	optional
Total	***100%***

For this class to count towards the computer science and network and system administration majors, you must earn at least a C-.

Grading Scale

100 – 92	A	77.9 – 72	C
91.9 – 90	A-	71.9 – 70	C-
89.9 – 88	B+	69.9 – 68	D+
87.9 – 82	B	67.9 – 62	D
81.9 – 80	B-	61.9 – 60	D-
79.9 – 78	C+	59.9 – 0	F

Class Attendance and Participation

If you miss a class due to illness, quarantine, or any other reason, it is your responsibility to find out what was covered by watching the recording of the lecture (Panopto link found on elearning). If you must miss a class with an in-class lab, you must finish the lab on your own.

Lab Work

At different points in the class, we will pause from regular lectures to implement a learned concept during a lab (either in-class or on your own). These labs are meant to give you experience outside of your textbook environment with a real database. In-class labs are typically meant to be finished within the regular class period, but if they are not completed in class, they must be completed by the due date.

Lab work is to be completed individually unless made into a partner/group activity in class.

Zybook Participation and Challenge Activities

Zybook participation activities can be found within the textbook chapter section. These questions are meant to gauge if you are reading and understanding the material and should be completed before the class in which they are discussed.

Zybook challenge activities are more challenging than patriation activities and can also be found with the textbook chapter section. These activities are a bit more like traditional homework. It is expected that you attempt these activities as you read the section and bring your questions to class.

Both weekly zybook participation and challenge activities for the weekly assigned readings are due each Friday at 11:59pm. Again, it is a great idea for you to read these assigned sections before the class in which they are discussed so that you can engage more fully with the lecture and have an

opportunity to ask questions on any challenge activity for which you are stuck.

Participation and challenge activities are to be completed individually.

Exams

There is a total of three in-class regular exams this semester.

- Each regular exam must be taken during the scheduled time and will cover content from the knowledge unit.
- The final exam is cumulative and is optional. If you elect to take the final, it will replace the lowest regular exam grade, regardless if the final grade is higher or lower than the lowest regular exam grade.

By default these exams are closed-book/closed-notes exams. The dates of these exams are listed on the class schedule. You are expected to be present for these exams unless you have made prior arrangements. Make-up exams will be offered under very limited circumstances. If you are aware of conflicts prior to the exam, please bring these to my attention as early as possible.

Final Hands-on Project

The hands-on group project will involve the design, implementation, and front-end programming of a small database containing real-world data. This project will be introduced shortly before Thanksgiving break. In the final two regular weeks of class after Thanksgiving break, you will be given class time to work on the project.

The hands-on project is to be completed individually.

Incompletes

Incompletes are awarded only in very rare instances when an unforeseeable event causes a student who has completed all the coursework to date to be unable to complete a small portion of the work in the last week or two of the semester (typically the final project or exam). Incompletes will not be awarded for foreseeable events including a heavy course load or a poorer-than- expected performance. Verifiable documentation must be provided for the incomplete to be granted.

Tentative Schedule (Subject to Change)

Date	Topics/Notes
8/25 – 8/29	Introduction to Databases
9/1 –9/19	Relational Database and Basic SQL
9/22	Exam #1
9/24 – 10/17	Normal Forms and Advanced SQL
10/20	Exam #2
10/22 –11/14	Database Design and Front-End Programming
11/17	Exam #3
11/19 – 12/12	Final Comprehensive Database Project
12/16 (Tuesday)	Optional Final (1-2:50pm)

AI Statement

Summary

Do not use AI tools to answer textbook questions, complete labs, or else generate programs and projects for this class unless given permission. You can use AI to help you understand something for learning purposes, but the materials submitted for points in the class need to be completely understood and generated by you. If not, then you'll probably fail the exams.

Full AI Statement

AI (LLM tools) are rapidly changing the face of software engineering. These tools are available in both external chat websites and smart editors that can generate full programs based on a prompt. These tools hold great promise to increase productivity in programmers who already know how to code, but they can also generate "AI slop", which is fragile code that incorporate bad programming practices and may break during stress. (See table below in section Extra: Is "vibe programming" really so bad?)

Employers know the difference between programmers that truly understand code versus programmers that overly rely on AI. Because of this, programming and technical knowledge tests are increasingly used to screen applicants to positions where AI use becomes obvious and detected. One purpose of this class is to give you the background fundamental knowledge you need to become an experienced programmer in the topic domain of this class.

Nearly half of the points in this class will be evaluated on your abilities to read and write code in exam environments without access to an AI. If you can't generate and read SQL code/schemas yourself without the use of an AI, you will fail exams and the class. You really must know your stuff.

Extra: Is "vibe programming" really so bad?

The term "vibe programming" refers to using an AI to create code without any or much programming knowledge. We already know that AI tools can hallucinate fake function calls, commands, and libraries that don't exist. In addition to hallucinations, the table below shows some bad programming practices AI coding tends to produce (reference), especially when the code becomes large. Inexperienced programmers can also produce these same errors! If you are not an experienced programmer or don't know the program domain, then it will be difficult to recognize when the AI causes these types of problems. ***In other words, it's important to take the time to understand the programming and your problem domain before you use AI.***

Common errors vibe programmers may not spot in AI-produced code:

Term	Description
spaghetti code	badly structured code that is difficult to detangle
god-object	a software component that does too many things and it must be simplified
shotgun surgery	when you make too many unrelated changes in too many files at the same time
glue code	code that connects other code - normally glue code is perceived as being lower quality than the components that it connects
throw-away code	a quick hack that will be inconsequential if removed
golden hammer	when you over-use a tool or a pattern even when it is not a good fit for the problem domain
cargo cult programming	using code or patterns without understanding them, just because they seem to work
fear-driven development	coding driven by fear of touching fragile systems
stovepipe system	system developed in isolated pieces with little to no integration
heisenbug	a bug that seems to disappear when you try to debug it

code smell	a surface indication that something may be wrong with the code
technical debt	poor design or quick fixes that need to be "repaid" with future refactoring
yoda conditions (yoda code)	when you write code in reverse or illogical order - for example **if (5 == x)** instead of **if (x == 5)**
magic numbers	using unexplained constants directly in code instead of named variables
hardcoding	embedding fixed values or logic inside the code, reducing flexibility
refuctoring	a process that makes the code worse and less understandable - the opposite of refactoring
dead code	code that is no longer used or it is not reachable
zebra pattern	different coding styles in the same program
factory factory	overuse of the factory pattern - a sign of over-engineering
over-engineering	too much planning and structure that leads to less flexibility
verbose code	code that could be expressed in simpler terms
AI slop (for code)	a new term that describes code that is written by vibe coder that exhibits many of the bad practices outlined above