

# Database Systems

CS 361 - Spring 2020

**Instructor:** Hannah Hillberg  
**Email:** hillbergh@uwosh.edu  
**Office:** Halsey 217  
**Office Hours:** 2:00-3:30pm MW  
4:30-5:30pm Tu  
9:15-10:15am F

**Section:** 001C  
**Class:** TuTh 3:00-4:30pm Halsey 208

## Course Information

An introduction to database processing with emphasis on database techniques, design, and modeling. Programming projects may include implementation of selected database processing methods and the use of database software.

**Credits:** This is a 3 credit course.

**Prerequisites:** A grade of C or better in Computer Science 212 and Computer Science 271

## Recommended Textbooks (NOT required):

- *Database System Concepts* (7th Edition), Abraham Silberschatz, Henry Korth and S. Sudarshan, McGraw-Hill, 2010
- *Database Systems: The Complete Book* (2nd Edition), Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom, Prentice Hall, 2008

## Course Website: UWO Canvas

You should check Canvas on a regular basis - it will contain lecture notes, assignments, announcements, and grades. I'll do my best to let you know when something new and important comes up, but it is your responsibility to check the website frequently for information that you might not get otherwise.

## Course Topics

- Overview of database systems and introduction to database design
- The relational model, relational algebra and calculus
- Schema refinement and normal forms
- SQL: queries, constraints, triggers
- Database application development
- Overview of storage and indexing
- Overview of query evaluation and query optimization
- Overview of transaction management, concurrency control and crash recovery

### Course Grading Policy

Your final grade for this course will be based on four components, namely exams, programming and modeling assignments, homework, and general class participation. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Exams (3)	45%
Homework	20%
Programming and Modeling Assignments	30%
Class Participation	5%

Your letter grade for the course will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
$\geq 92$	A	72-78	C
90-92	A-	70-72	C-
88-90	B+	68-70	D+
82-88	B	62-68	D
80-82	B-	60-62	D-
78-80	C+	$<60$	F

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me as soon as possible. Do not wait until the end of the semester to bring up grading issues.

#### ***Tentative Exam Dates:***

- Exam 1 – Thursday, February 27
- Exam 2 – Thursday, April 9
- Exam 3 – Thursday, May 14

If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do both of the following, which are then subject to my approval:

1. Make arrangements prior to the scheduled exam. For last minute emergencies, email me immediately. No after-the-fact notifications will be accepted.
2. Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

### **Deadlines and Late Days**

Each homework will come with a deadline by which it must be submitted (usually, at the start of a lecture). Late homework submissions will NOT be accepted.

Each programming or modeling assignment will also come with a deadline (day and time) by which it must be submitted. You are allotted **three credit days** that you can use throughout the semester to submit an assignment after its deadline without penalty. A credit day is exactly 24 hours or less. Please keep in mind that three days is not much, so plan to hit the deadlines and save the late days for real emergencies. Any assignment submitted after the deadline, plus any credit days you have unused, will receive a zero.

### **Academic Dishonesty**

Academic dishonesty of any kind will not be tolerated. Unless otherwise stated, all homeworks, assignments, and exams are to be completed individually and must be entirely your own work. While discussion of ideas and problems at a high level with fellow students is encouraged, you must submit only your own work. In certain circumstances, code fragments from the instructor may be provided to eliminate tedious coding or to provide a common framework for all students. All other code must be original. Online resources may be used to help you understand the material, but you may not copy online code nor can you “borrow” code from other students, past or present.

Any suspected academic dishonesty will be dealt with on a case-by-case basis. Any clarification of what does or does not constitute academic dishonesty must take place *before* you turn in questionable work. For clarification on what constitutes academic dishonesty, contact me or consult the printed policy in the [UWO Student Discipline Code, Chapter UWS 14.](#)

### **Accessibility**

It is the University’s policy to provide, on a flexible and individual basis, reasonable accommodations to students who have documented disabilities that may affect their ability to participate in course activities or to meet course requirements. Students with disabilities must be registered with the Accessibility Center and receive an Accommodation Recommendation form to receive accommodations. The Accessibility Center is located in Dempsey Hall 125.

It is also the policy and practice of UW Oshkosh to create an inclusive learning environment. If there are aspects of the instruction or design of this course that result in barriers to your inclusion, please notify me as soon as possible. You are also welcome to contact the Accessibility Center at (920) 424-3100 or [accessibilitycenter@uwosh.edu](mailto:accessibilitycenter@uwosh.edu). For more

information, visit the Accessibility website at <http://www.uwosh.edu/deanofstudents/accessibilitycenter>.

## Learning Outcomes

Students should be able to do the following upon successful completion of the course:

- Fundamental concepts of database design
  - model customer requirements of a relational database with an Entity-Relational diagram (E-R diagram)
  - transform an E-R diagram to a database schema
  - write a query to a relational database in SQL
  - formulate a query to a relational database from the basic operators in relational algebra
  - design a database to provide the necessary information for an organization while minimizing redundancy and null entries.
- SQL
  - design and create a database using the Data Definition Language of a Database Management System
  - write queries to a relational database in an interactive mode
- Design of “good” relations, Schema Refinement, Concept of normalization and other theoretical issues
  - formulate the integrity constraints in the form of functional dependencies
  - eliminate extraneous attributes in a functional dependency
  - eliminate redundant functional dependencies
  - develop a cover from a set of functional dependencies
  - evaluate a proposed relational schema and determine whether it is in Third-Normal-Form (3NF) or Boyce-Codd-Normal-Form (BCNF)
  - implement a normalization program that checks whether a proposed relational schema is in 3NF or BCNF
  - decompose proposed relational schemas that are not in 3NF or BCNF into 3NF or BCNF
- Basic file organization and various file structure methods
  - determine the access time of records based on the file organization and file structure
  - specify the type of stable and non-stable storage in the design of a database management system.
  - analyze the requirements and select the design of an index (Hash, B+ tree)
  - organize data on disk to minimize disk accesses for various queries.
- Algorithms and implementation of large database systems
  - analyze the need of a database operator (scan, equality search, range search, insert, delete etc) and determine an appropriate and/or efficient algorithm (external sorting, hash, B+, clustered vs. unclustered, various join algorithms) in its implementation.

- implement a Relational Operation (Example: Join)
- Transaction processing, concurrency issues, and recovery
  - identify and prevent deadlocks in concurrent database accesses
  - be able to describe the recovery process of databases
  - design the data structure and program of a database recovery mechanism
  - provide accurate, consistent, and efficient transactions within the context of concurrency issues and the possibility of various kinds of failures, such as a system crash.