Computer Science 6360: Database Design

- Instructor: Weili Wu
- Email: weiliwu@utdallas.edu
- (972) 883-2194
- Office Hours: Wednesday 3:45pm – 5:00pm ECSS 3.229
- Class web page: http://www.utdallas.edu/~weiliwu/CS6360_S2023/CS6360_S2023.htm (available on eLearning)
- Assignments due on Mondays
- T.A. – TBD
- Prerequisite: CS5343 Algorithm Analysis and Data Structure

Textbooks

Required:

- Elmasri and Navathe, Fundamentals of Database Systems, 6th Edition, Addison Wesley, ISBN 10: 0-136-08620-9; ISBN 13: 978-0-136-08620-8.

Grading

- Assignments:
  - 1 @ 3% , 2@5%, 3@7% = **15%** of grade.
- Project: 15%
- Midterm:
  - **30%** of grade.
- Final:
  - **40%** of grade.
- Conditions for passing the class:
  - Submit all HWs and Project.
  - Scoring >= 50% on final exam

Assignments Submission

- **Submission Policy:**
  - HWs must be submitted via eLearning on specified due date (Mondays of designated weeks).
  - The project must be submitted via eLearning on due date.
  - **Late hws** also should be submitted via eLearning (postscript, pdf, text or MS Word doc files).
  - Late HWs **penalty**:
    - 1 day -- 30% will be deducted
    - 2 days – 70% will be deducted (count weekend days)
    - >=3 days – no credit

Comment

- This class is very interesting and useful.
- work regularly
- Good luck

- Schedule
  - View the candidate course schedule on course syllabus and eLearning
  - Check update via eLaening

Overview of Databases and Basic Concepts

Chapter 1, 2

What is a Database?

- Collection of data central to some enterprise
  - **Data**: Known facts that can be recorded and have an implicit meaning.
- Essential to operation of enterprise
  - Contains the only record of enterprise activity
- An asset in its own right
  - Historical data can guide enterprise strategy
  - Of interest to other enterprises
- Database is persistent
- **Mini-world**:
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.

What is a Database Management System?

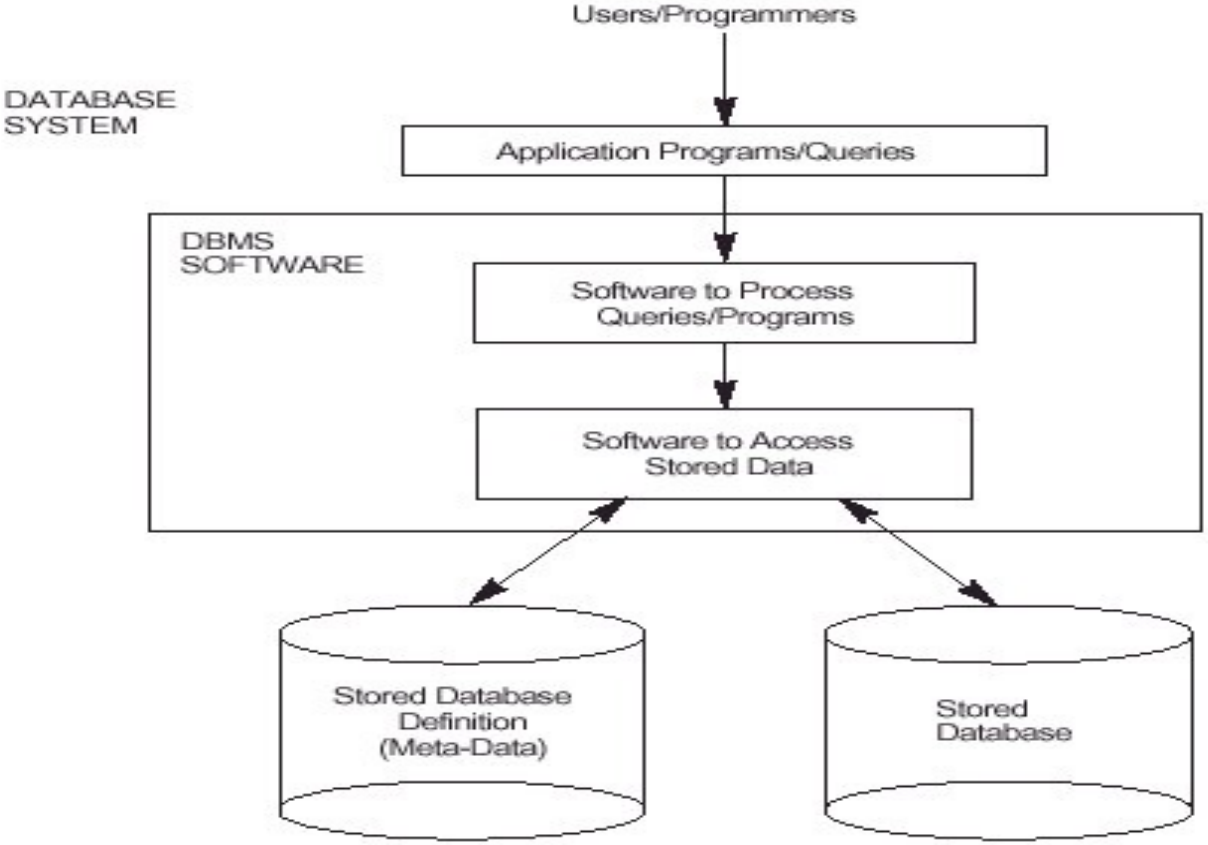
- A Database Management System (DBMS) is a program/software package that manages a database:
  - Supports a high-level access language (e.g. SQL).
  - Application describes database accesses using that language.
  - DBMS interprets statements of language to perform requested database access.

What is a Database Management System? (Cont.)

- Supports concurrent access to very large amounts of data.
  - Example: bank and its ATM machines.
- Supports secure, atomic access to very large amounts of data.
  - Contrast two people editing the same UNIX file
  - with the problem if two people deduct money from the same account via ATM machines at the same time
  - new balance is wrong whichever writes last.

Database System

- The DBMS software together with the data itself. Sometimes, the applications are also included.


Users/Programmers
DATABASE SYSTEM
Application Programs/Queries
DBMS SOFTWARE
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

The DBMS Marketplace

- Relational DBMS companies – Oracle, Sybase – are among the largest software companies in the world.
- IBM offers its relational DB2 system. With IMS, a nonrelational system, IBM is by some accounts the largest DBMS vendor in the world.
- Microsoft offers SQL-Server, plus Microsoft Access for the cheap DBMS on the desktop, answered by “lite” systems from other competitors.
- Relational companies also challenged by “object-oriented DB” companies.
- But countered with “object-relational” systems, which retain the relational core while allowing type extension as in OO systems.

An Example of Database

- **Mini-world for the example**:
  - Part of a UNIVERSITY environment.
- **Some mini-world *entities***:
  - STUDENTs (ID, Name, SecId, CourseNum, …)
  - COURSEs
  - SECTIONs (of COURSEs)
  - DEPARTMENTs
  - INSTRUCTORs

- **Some mini-world *relationships***:
  - SECTIONs *are of* specific COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have* prerequisite COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

- The above could be expressed in the *ENTITY-RELATIONSHIP* (ER) data model

Figure 1.2 An example of a database that stores student records and their grades.

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

Benefits of Using DBMS

- Controlling redundancy in data storage and in development and maintenance efforts.
- Sharing of data among multiple users.
- Restricting unauthorized access to data.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Providing backup and recovery services.

Benefits of Using DBMS(cont.)

- Potential for enforcing standards
- Flexibility to change data structures.
- Reduced application development time.
- Availability of up-to-date information.
- Economies of scale.

Figure 1.5 The redundant storage of data items. (a) *Controlled redundancy:* Including StudentName and CourseNumber in the grade_report file. (b) *Uncontrolled redundancy:* A GRADE_REPORT record that is inconsistent with the STUDENT records in Figure 1.2, because the Name of student number 17 is Smith, not Brown.

(a)

GRADE_REPORT	StudentNumber	StudentName	SectionIdentifier	CourseNumber	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	Brown	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	StudentNumber	StudentName	SectionIdentifier	CourseNumber	Grade
	17	Brown	112	MATH2410	B

When not to use a DBMS

- **Main costs of using a DBMS**:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, recovery, integrity, and concurrency control .
- **When a DBMS may be unnecessary:**
  - simple, well defined, and not expected to change
  - If access to data by multiple users is not required
- **When no DBMS may suffice:**
  - DB system can not handle the complexity of data
  - Not support special operations

Three Aspects to Studying DBMS's

1. Modeling and design of databases.
   – Allows exploration of issues before committing to an implementation.
2. Programming: queries and DB operations like update.
   – SQL = “intergalactic data speak.”
3. DBMS implementation.
   - Query processing and optimization
   - Transaction
   - Concurrency control

Main Characteristics of Database Technology

- Self-contained nature of a db system: A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.
- Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.
- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Support of multiple views of the data: Each user may see a different view of the database, which describes *only* the data of interest to that user.
  - Virtual data (not actual stored)
- Sharing of data and multiuser transX processing: Concurrency control

Figure 1.4 Two views derived from the example database shown in Figure 1.2. (a) The student transcript view. (b) The course prerequisite view.

(a)

TRANSCRIPT	StudentName	Student Transcript				
		CourseNumber	Grade	Semester	Year	SectionId
	Smith	CS1310	C	Fall	99	119
		MATH2410	B	Fall	99	112
	Brown	MATH2410	A	Fall	98	85
		CS1310	A	Fall	98	92
		CS3320	B	Spring	99	102
		CS3380	A	Fall	99	135

(b)

PREREQUISITES	CourseName	CourseNumber	Prerequisites
	Database	CS3380	CS3320
			MATH2410
	Data Structures	CS3320	CS1310

Data Model

- **Data Model**:
  - A set of concepts to describe the *structure* of a database
  - certain *constraints, data types, relationships* that the database should obey
  - Provides abstraction
    - Hide low level storage details
- **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model.

Categories of data models

- **Conceptual** data models:
  - **high-level**, **semantic**
  - Provide concepts that are close to the way many users *perceive* data. (Also called **entity-based** or **object-based** data models.)
- **Physical** data models:
  - **low-level**, **internal**
  - Provide concepts that describe details of how data is stored in the computer.
- **Implementation** (**record-oriented**) data models:
  - Provide concepts that fall between the above two, balancing user views with some computer storage details.

High-level Data Models

- Record-based
  - Relational data model
  - Network
  - Hierarchical
- Object-based
  - Close to human perception
  - Farther from computer system

HISTORY OF DATA MODELS

- Relational Model: proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Now in several commercial products (ORACLE, SYBASE, INFORMIX, CA-INGRES).
- Network Model: the first one to be implemented by Honeywell in 1964-65 (IDS System). Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971). Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital).

- <u>Hierarchical Data Model</u> : implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems. The most popular model. Other system based on this model: System 2k (SAS inc.)
- <u>Object-oriented Data Model(s)</u> : several models have been proposed for implementing in a database system. One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE). Additionally, systems like O2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
- <u>Object-Relational Models</u> : Most Recent Trend. Exemplified in ILLUSTRA and UNiSQL systems

Schemas versus Instances

- **Database Schema**:
  - The *description* of a database
  - Includes descriptions of the database structure and the constraints that should hold on the database
  - Specified during DB design
  - Not expected to change frequently
- **Schema Diagram**:
  - A diagrammatic display of (some aspects of) a database schema (e.g. Fig 2.1)

Figure 2.1 Schema diagram for the database of Figure 1.2.

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

PREREQUISITE

CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade

Instance, State

- **Database Instance**:
  - The actual data stored in a database at a *particular moment in time* .
  - Also called **database state** (or **occurrence**).
- **Schema Vs Instance (State)**:
  - The **database schema** changes *very infrequently*
  - **Schema** is also called **intension**
  - The **database state** changes *every time the database is updated*
  - **state** is called **extension**.

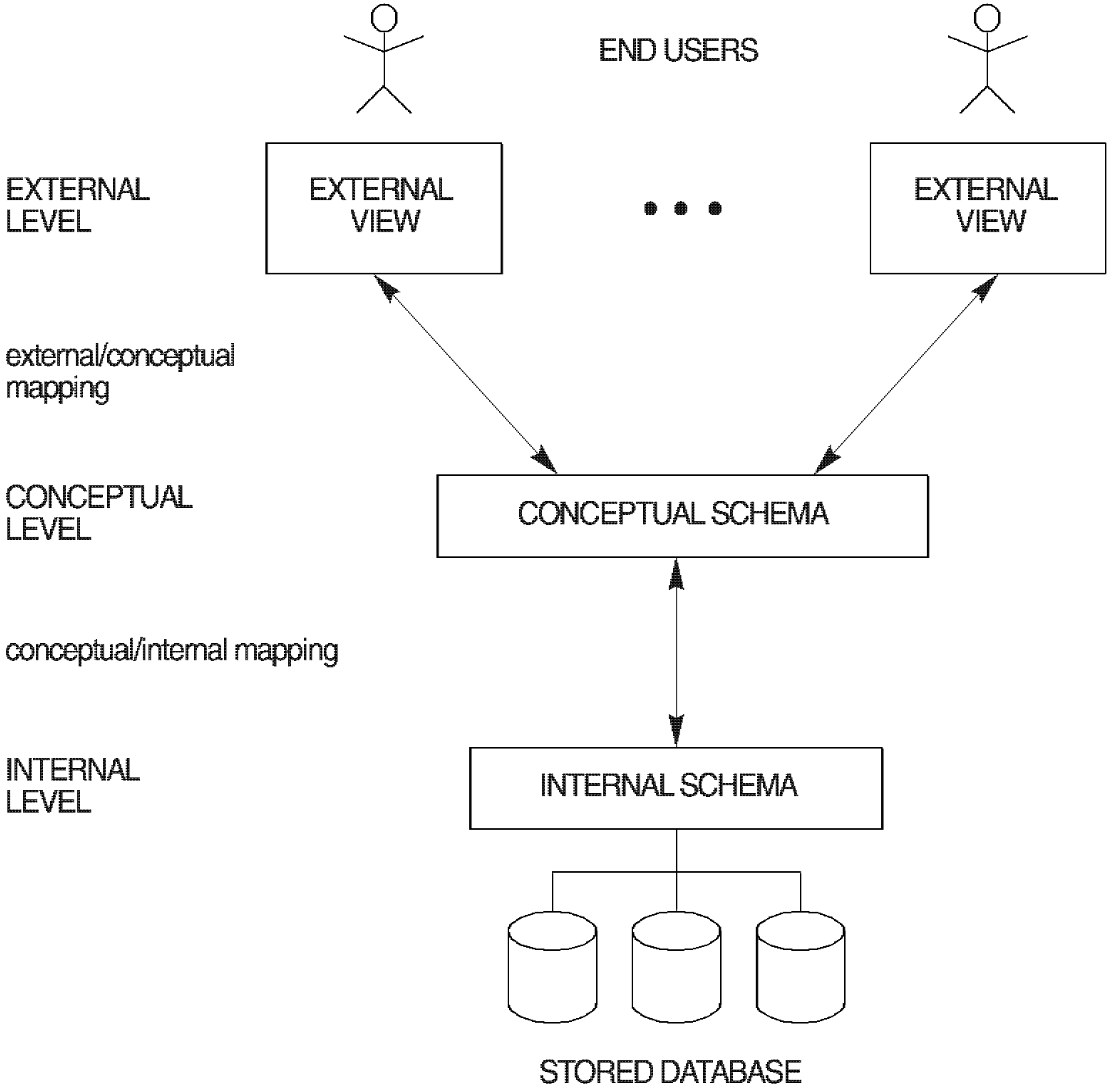
Three-Schema Architecture

- Proposed to support DBMS characteristics:
  - **Program-data independence**.
  - Support of **multiple views** of the data

- Defines DBMS schemas at *three levels*
  - **Internal schema**: describe data storage structures and access paths. Typically uses a *physical* data model.
  - **Conceptual schema:** at the conceptual level to describe the structure and constraints for the *whole* database. Uses a *conceptual* or an *implementation* data model.
  - **External schemas:** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

- **Mappings** among schema levels are also needed.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

Figure 2.2 Illustrating the three-schema architecture.


END USERS
EXTERNAL LEVEL
EXTERNAL VIEW
• • •
EXTERNAL VIEW
external/conceptual mapping
CONCEPTUAL LEVEL
CONCEPTUAL SCHEMA
conceptual/internal mapping
INTERNAL LEVEL
INTERNAL SCHEMA
STORED DATABASE