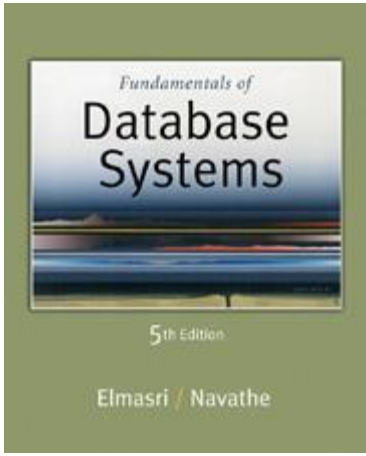
Database Management Systems

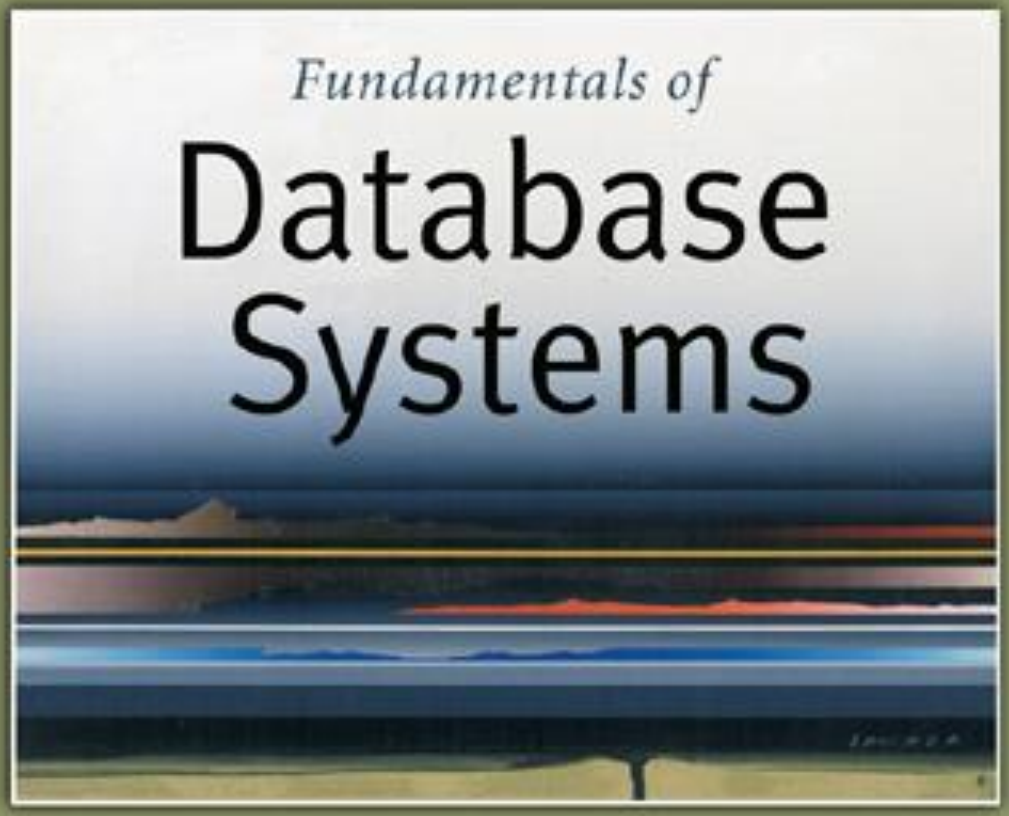
Dr. Huda Amin

Email: huda_amin@cis.asu.edu.eg



- Marks Distribution
  - Final 50
  - Mid Term 15
  - Year Work 10
  - Quiz 5
  - Practical 20
- Total 100

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe



5th Edition

Elmasri / Navathe

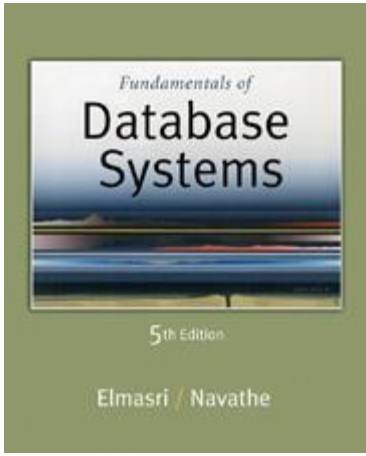
Course Contents

- Introduction to Database and Database Users
- DBMS data models, DBMS as a software architecture
- Entity-Relationship (ER) Model
- Enhanced ERD
- Relational Data Model
- Mapping of an ER schema into a relational schema
- Database Normalization
- Relational Algebra
- Indexing Structures for Files
- Algorithms for Query Processing and Optimization

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Chapter 1

Introduction: Databases and Database Users



PEARSON
Addison
Wesley

- Databases and database systems are an essential component of life in modern society:
  - Most of us encounter several activities every day that involve some interaction with a database.
- Databases play a critical role in almost all areas where computers are used
  - including business, electronic commerce, engineering, medicine, genetics, law and education.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Outline

- Types of Databases and Database Applications
- Basic Definitions
- Example of a Database (UNIVERSITY)
- Typical DBMS Functionality
- Main Characteristics of the Database Approach
- Database Users
- Advantages of Using the Database Approach
- When not to use a DBMS

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Types of Databases and Database Applications

- **Traditional database applications** stores textual or numeric information.
- **Multimedia databases** stores images, audio clips, and video streams digitally.
- **Geographic information systems (GIS)** can store and analyze maps, weather data, and satellite images.
- **Data warehouses and online analytical processing (OLAP) systems** are used to Extract and analyze useful business information from very large databases to support decision making.
- **Real-time** database is a database system which uses real-time processing to handle workloads whose state is constantly changing. For example, a stock market changes very rapidly and is dynamic.

- **What else?** At least 3 more examples

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Basic Definitions

- **Database:**
  - A collection of **related** data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
    - Example!
- **Mini-world:**
  - represents some aspect of the real world. For example, student grades and transcripts at a university.
  - In order for a database to be accurate and reliable at all times, it must be a **true reflection of the mini- world** that it represents; therefore, changes must be reflected in the database as soon as possible.
- **Transaction**
  - It is an executing program or process that includes one or more database accesses, such as reading or updating of database records.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

A database can be of any size and complexity.

For example:

- The list of names and addresses referred to earlier may consist of only a few hundred records, each with a simple structure.

- On the other hand, the computerized catalog of a large library may contain half a million entries organized under different categories—by primary author's last name, by subject, by book title—with each category organized alphabetically.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- A database of even greater size and complexity is maintained by the Internal Revenue Service (IRS) to monitor tax forms filed by U.S. taxpayers. If we assume that there are 100 million taxpayers and each taxpayer files an average of five forms with approximately 400 characters of information per form, we would have a database of 100 × 106 × 400 × 5 characters (bytes) of information. If the IRS keeps the past three returns of each taxpayer in addition to the current return, we would have a database of 8 × 1011 bytes (800 gigabytes). This huge amount of information must be organized and managed so that users can search for, retrieve, and update the data as needed.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- An example of a large commercial database is Amazon.com. It contains data for over 20 million books, CDs, videos, DVDs, games, electronics, apparel, and other items. The database occupies over 2 terabytes (a terabyte is 1012 bytes worth of storage) and is stored on 200 different computers (called servers). About 15 million visitors access Amazon.com each day and use the database to make purchases. The database is continually updated as new books and other items are added to the inventory and stock quantities are updated as purchases are transacted. About 100 people are responsible for keeping the Amazon database up-to-date. [2010]
  - **59 million active customers**
  - **More than 42 terabytes of data**

  - **More Examples!**

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Basic Definitions

- **Database Management System (DBMS):**
  - A software package/ system to facilitate the creation and maintenance of a computerized database.

- **Database System:**
  - The DBMS software together with the database itself. Sometimes, the applications are also included.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Simplified Database System Environment

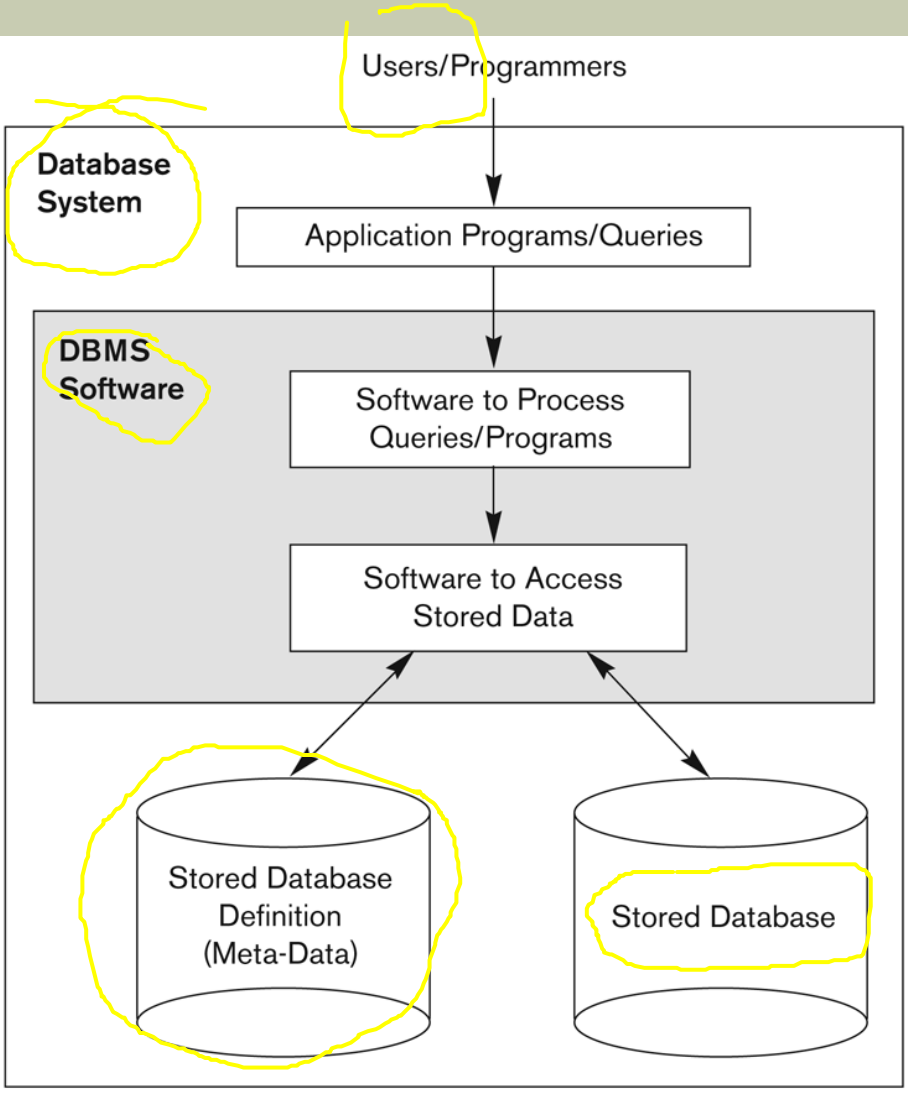


Figure 1.1
A simplified database system environment.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example:**
  - Part of a UNIVERSITY environment.
- **Some mini-world *entities*:**
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Example of a Database (with a Conceptual Data Model)

- **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

- Note: The above entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model (see Chapters 3, 4)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Example

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Typical DBMS Functionality

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or Load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Typical DBMS Functionality

- Other features:
  - Protection or Security measures to prevent unauthorized access
  - "Active" processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Main Characteristics of the Database Approach

- **Database approach vs. file processing approach!**
  - **Inconsistency**
  - **Redundancy**
  - **Waste storage**
  - **Much effort to maintain updates**

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- In traditional **file processing, each user** defines and implements the files needed for a specific software application as part of programming the application.

- For example, one user, the ***grade reporting office****, may* keep files on students and their grades. Programs to print a student's transcript and to enter new grades are implemented as part of the application. A second user, the ***accounting office****, may keep track of students' fees and their payments. Although* both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files.
- This redundancy in defining and storing data results in wasted storage space and in **redundant efforts to maintain common up-to-date data.**

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- In the database approach, a **single repository** maintains data that is **defined once** and then **accessed by various users**.

- In file systems, each application is free to name data elements independently. In contrast, in a database, the names or labels of data **are defined once,** and used repeatedly by queries, transactions, and applications.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Main Characteristics of the Database Approach (continued)

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Data Item Name	Starting Position in Record	Length in Characters (bytes)
Name	1	30
Student_number	31	4
Class	35	1
Major	36	4

Figure 1.4
Internal storage format for a STUDENT record, based on the database catalog in Figure 1.3.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Main Characteristics of the Database Approach (continued)

- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization 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.
    - Programs refer to the data model constructs rather than data storage details.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require ***changing all programs*** that access that file.

- For example, a file access program may be written in such a way that it can access only STUDENT records of the structure shown in Figure 1.4. If we want to add another piece of data to each STUDENT record, say the Birth_date, **such a program will no longer work and must be changed.** By contrast, in a DBMS environment, **we only need to change the description of STUDENT records in the catalog** (Figure 1.3) to reflect the inclusion of the new data item Birth_date; **no programs are changed**. The next time a DBMS program refers to the catalog, the new structure of STUDENT records will be accessed and used.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Main Characteristics of the Database Approach (continued)

- **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.
  - A view may be a subset of the database or it may contain **virtual data** that is derived from the database files but is not explicitly stored.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- For example, one user of the database of Figure 1.2 may be interested only in accessing and printing the transcript of each student; the view for this user is shown in Figure 1.5(a).
- A second user, who is interested only in checking that students have taken all the prerequisites of each course for which they register, may require the view shown in Figure 1.5(b).

TRANSCRIPT

(a)

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

COURSE_PREREQUISITES

(b)

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

Figure 1.5
Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS **guarantees** that each transaction is **correctly executed or aborted**
    - For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger.
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent **transactions** to execute per second.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- The **isolation property** ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently. The **atomicity property** ensures that either all the database operations in a transaction are executed or none are.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Database Users

- Users may be divided into
  - Those who actually use and control the database content, and those who design, develop and maintain database applications (called “Actors on the Scene”), and
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (called “Workers Behind the Scene”).

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Database Users

- Actors on the scene
  - **Database administrators:**
    - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
  - **Database Designers:**
    - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Categories of End-users

- Actors on the scene (continued)
  - **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
    - **Casual**: access database occasionally when needed
    - **Naïve** or Parametric: they make up a large section of the end-user population.
      - They use previously well-defined functions in the form of "canned transactions" against the database.
      - Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Categories of End-users (continued)

- **Sophisticated:**
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone:**
  - Mostly maintain personal databases using ready-to-use packaged applications.
  - An example is a tax program user that creates its own internal database.
  - Another example is a user that maintains an address book.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

- A typical DBMS provides multiple facilities to access a database.
  - **Naive end users** need to learn very little about the facilities provided by the DBMS; they simply have to understand the user interfaces of the standard transactions designed and implemented for their use.
  - **Casual users** learn only a few facilities that they may use repeatedly.
  - **Sophisticated users** try to learn most of the DBMS facilities in order to achieve their complex requirements.
  - **Standalone users** typically become very proficient in using a specific software package.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Advantages of Using the Database Approach

- Controlling **redundancy** in data storage and in development and maintenance efforts.
  - Sharing of data among multiple users.
- Restricting unauthorized access to data.
  - Unauthorized access: For example, financial data is often considered **confidential**, and only **authorized persons** are allowed to access such data. some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update.
- Providing Storage Structures (e.g. indexes) for efficient Query Processing.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Advantages of Using the Database Approach (continued)

- Providing backup and recovery services.
  - For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
  - The simplest type of integrity constraint involves specifying a data type for each data item. more complex type of constraint that frequently occurs involves specifying that a record in one file must be related to records in other files.
  - Another type of constraint specifies uniqueness on data item values constraints business rules

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Additional Implications of Using the Database Approach

- Potential for enforcing standards:
  - This is very crucial for the success of database applications in large organizations. **Standards** refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.
- Flexibility to change data structures:
  - Database structure may evolve as new requirements are defined.
- Availability of current information:
  - Extremely important for on-line transaction systems such as airline, hotel, car reservations.

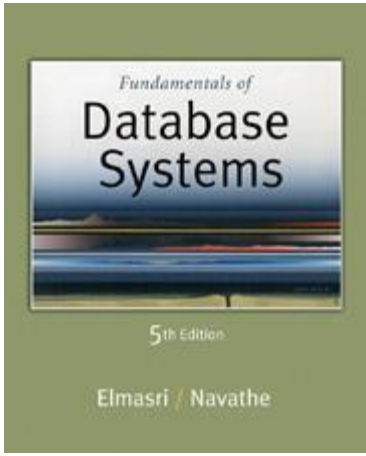
Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware.
  - **Overhead** for providing generality, security, concurrency control, recovery, and integrity functions.
- When a DBMS may be unnecessary:
  - If the database and applications are **simple**, well defined, and **not expected to change**.
  - If there are **embedded** systems with **limited storage** capacity, where a general-purpose DBMS would not fit
  - If access to data by **multiple users is not required**.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Thanks for Listening



PEARSON
Addison
Wesley