Database Systems
(M3020008)

Lecture 1

School of Computer Science and Engineering (SoCSE)

Digital University Kerala ,Thiruvananthapuram

Databases and Database Users

- 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.

- **For example**, If we go to the bank to deposit or withdraw funds, if we make a hotel or airline reservation, if we access a computerized library catalog to search for a bibliographic item, or if we purchase something online—such as a book, toy, or computer—chances are that our activities will involve some one or some computer program accessing a database.

- Even purchasing items at a supermarket often automatically updates the database that holds the inventory of grocery items.

Traditional Database/Big Data/NoSQL/Cloud Storage Systems

- These interactions are examples of what we may call **traditional database applications**, in which most of the information that is stored and accessed is either **Textual** or **Numeric**.

- In the past few years, advances in technology have led to exciting new applications of database systems.

- The proliferation of social media Web sites, such as Facebook, Twitter, and Flickr, among many others, has required the creation of huge databases that store non traditional data, such as posts, tweets, images, and video clips.

- New types of database systems, often referred to as **Big data** storage systems, or **NOSQL systems**, have been created to manage data for social media applications.

- These types of systems are also used by companies such as Google, Amazon, and Yahoo, to manage the data required in their Web search engines, as well as to provide **cloud storage**, whereby users are provided with storage capabilities on the Web for managing all types of data including documents , programs, images, videos and emails.

Other Databases

- We now mention some other applications of databases.

- The wide availability of photo and video technology on cellphones and other devices has made it possible to store images, audio clips, and video streams digitally.

- These types of files are becoming an important component of **Multimedia databases**.

- **Geographic information systems (GISs)** can store and analyze maps, weather data, and satellite images.

- **Data warehouses** and **On-Line Analytical Processing (OLAP)** systems are used in many companies to extract and analyze useful business information from very large databases to support decision making.

- **Real-time** and **active database technology** is used to control industrial and manufacturing processes.

- And database **Search techniques** are being applied to the **World Wide Web** to improve the search for information that is needed by users browsing the Internet.

Introduction

- Databases and database technology have had a major impact on the growing use of computers.

- It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, social media, engineering , medicine, genetics, law, education, and library science.

- The word *database* is so commonly used that we must begin by defining what a database is.

Database

- A **database** is a collection of related data.
- By **data**, we mean known facts that can be recorded and that have implicit meaning.
- For example, consider the names, telephone numbers, and addresses of the people you know.
- Nowadays, this data is typically stored in mobile phones, which have their own simple database software.
- This data can also be recorded in an indexed address book or stored on a hard drive, using a personal computer and software such as Microsoft Access or Excel.
- This collection of related data with an implicit meaning is a database.
- The preceding definition of database is quite general;
- For example, we may consider the collection of words that make up this page of text to be related data and hence to constitute a database.
- However, the common use of the term *database* is usually more restricted.

Database – Implicit Properties

- A database represents some aspect of the real world, sometimes called the **mini world** or the **universe of discourse (UoD)**. Changes to the mini world are reflected in the database.

- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.

- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

- In other words, a database has some source from which data is derived, some degree of interaction with events in the real world, and an audience that is actively interested in its contents.

- The end users of a database may perform business transactions (for example, a customer buys a camera) or events may happen (for example, an employee has a baby) that cause the information in the database to change.

- 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.

Size and Complexity of Database

- 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.
- A database of even greater size and complexity would be maintained by a social media company such as Facebook, which has more than a billion users.
- The database has to maintain information on which users are related to one another as *friends,* the postings of each user, which users are allowed to see each posting, and a vast amount of other types of information needed for the correct operation of their Web site.
- For such Web sites, a large number of databases are needed to keep track of the constantly changing information required by the social media Web site.

Amazon.com

- An example of a large commercial database is **Amazon.com.**

- It contains data for over 60 million active users, and millions of books, CDs, videos, DVDs, games, electronics, apparel, and other items.

- The database occupies over 42 terabytes (a terabyte is 1012 bytes worth of storage) and is stored on hundreds of computers (called servers).

- Millions of 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.

Database Generation/Maintenance

- A database may be generated and maintained manually or it may be computerized.

- For example, a library card catalog is a database that may be created and maintained manually.

- A computerized database may be created and maintained either by a group of application programs written specifically for that task or by a database management system.

- Of course, we are only concerned with computerized databases in this course.

DBMS (Data Base Management System)

- A **database management system (DBMS)** is a computerized system that enables users to **create** and **maintain** a database.

- The DBMS is a *general-purpose software system* that facilitates the processes of *defining, constructing, manipulating,* and *sharing* databases among various users and applications.

- **Defining** a database involves specifying the data types, structures, and constraints of the data to be stored in the database.

- The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called **meta-data**.

- **Constructing** the database is the process of storing the data on some storage medium that is controlled by the DBMS.

- **Manipulating** a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.

- **Sharing** a database allows multiple users and programs to access the database simultaneously.

Contd…

- An **application program** accesses the database by sending queries or requests for data to the DBMS.

- A **query** typically causes some data to be retrieved; (The term *query*, originally meaning a question or an inquiry, is sometimes loosely used for all types of interactions with databases, including modifying the data).

- A **transaction** may cause some data to be read and some data to be written into the database.

- Other important functions provided by the DBMS include *protecting* the database and *maintaining* it over a long period of time.

- **Protection** includes *system protection* against hardware or software malfunction (or crashes) and *security protection* against unauthorized or malicious access.

- A typical large database may have a life cycle of many years, so the DBMS must be able to **maintain** the database system by allowing the system to evolve as requirements change over time.

Database System

- It is not absolutely necessary to use general-purpose DBMS software to implement a computerized database.

- It is possible to write a customized set of programs to create and maintain the database, in effect creating a *special-purpose* DBMS software for a specific application, such as airlines reservations.

- In either case—whether we use a general-purpose DBMS or not—a considerable amount of complex software is deployed.

- In fact, most DBMSs are very complex software systems.

- To complete our initial definitions, we will call the **database** and **DBMS software** together a **database system**.
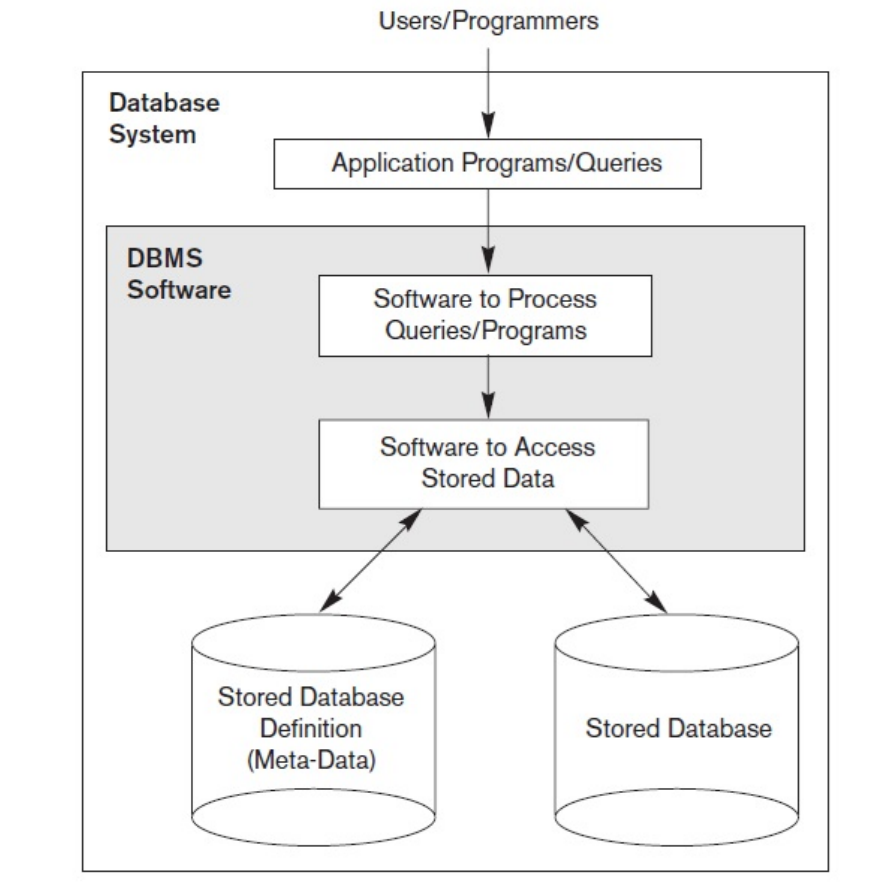
A Simplified Database System Environment

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

Figure 1

An Example : University Database

- Let us consider a simple example : a **UNIVERSITY** database for maintaining information concerning students, courses, and grades in a university environment.

- Figure shows the database structure and a few sample data records.

- The database is organized as five files, each of which stores **data records** of the same type.

- **STUDENT** file stores data on each student,

- **COURSE** file stores data on each course,

- **SECTION** file stores data on each section of a course,

- **GRADE_REPORT** file stores the grades that students receive in the various sections they have completed, and

- **PREREQUISITE** file stores the prerequisites of each course.

  - We use the term *file* informally here. At a conceptual level, a *file* is a *collection* of records that may or may not be ordered.

Records Structure

- To *define* this database, we must specify the structure of the records of each file by specifying the different types of **Data elements** to be stored in each record.

- In Figure, each STUDENT record includes data to represent the student's Name, Student_number , Class (such as freshman or '1', sophomore or '2', and so forth), and Major (such as mathematics or 'MATH' and computer science or 'CS');

- Each COURSE record includes data to represent the Course_name, Course_number , Credit_hours , and Department (the department that offers the course), and so on.

- We must also specify a **Data type** for each data element within a record.

- For example, we can specify that Name of STUDENT is a string of alphabetic characters, Student_number of STUDENT is an integer, and Grade of GRADE_REPORT is a single character from the set {'A', 'B', 'C', 'D', 'F', 'I'}.

- We may also use a coding scheme to represent the values of a data item.

- For example, in Figure , we represent the Class of a STUDENT as 1 for freshman, 2 for sophomore, 3 for junior, 4 for senior, and 5 for graduate student.

A Database – Student and Course Information

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 2

UNIVERSITY Database

- To *construct* the **UNIVERSITY** database, we store data to represent each student , course, section, grade report, and prerequisite as a record in the appropriate file.

- Notice that records in the various files may be related.

- For example, the record for Smith in the STUDENT file is related to two records in the GRADE_REPORT file that specify Smith's grades in two sections.

- Similarly, each record in the PREREQUISITE file relates two course records: one representing the course and the other representing the prerequisite.

- Most medium-size and large databases include many types of records and have *many relationships* among the records.

Database Manipulation

- Database *manipulation* involves querying and updating.

Examples of <u>queries</u> are as follows:

- Retrieve the transcript—a list of all courses and grades—of 'Smith'
- List the names of students who took the section of the 'Database' course offered in fall 2008 and their grades in that section
- List the prerequisites of the 'Database' course

Examples of <u>updates</u> include the following:

- Change the class of 'Smith' to sophomore
- Create a new section for the 'Database' course for this semester
- Enter a grade of 'A' for 'Smith' in the 'Database' section of last semester

These informal queries and updates must be specified precisely in the query language of the DBMS before they can be processed.

Database Description

- At this stage, it is useful to describe the database as part of a larger undertaking known as an information system within an organization.

- The Information Technology(IT) department within an organization designs and maintains an information system consisting of various computers, storage systems, application software , and databases.

- Design of a new application for an existing database or design of a brand new database starts off with a phase called **Requirements Specification and Analysis**.

- These requirements are documented in detail and transformed into a **Conceptual Design** that can be represented and manipulated using some computerized tools so that it can be easily maintained, modified, and transformed into a database implementation. (We will introduce a model called the **Entity-Relationship model** for this purpose.)

- The design is then translated to a **logical design** that can be expressed in a data model implemented in a commercial DBMS. (Various types of DBMSs will be discussed, with an emphasis on relational DBMSs mainly.)

- The final stage is **physical design**, during which further specifications are provided for storing and accessing the database.

- The database design is implemented, populated with actual data, and continuously maintained to reflect the state of the miniworld.

Database Approach - Characteristics

- A number of characteristics distinguish the database approach from the much older approach of writing customized programs to access data stored in files.

- 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.

Contd…

- In the database approach, a single repository maintains data that is defined once and then accessed by various users repeatedly through queries, transactions, and application programs.

- The main characteristics of the database approach versus the file-processing approach are the following:

✔ Self-describing nature of a database system

✔ Insulation between programs and data, and data abstraction

✔ Support of multiple views of the data

✔ Sharing of data and multiuser transaction processing

Self-describing nature of a database system

- A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a **complete definition or description of the database structure and constraints**.

- This definition is stored in the **DBMS catalog**, which contains information such as the **structure of each file, the type and storage format of each data item, and various constraints on the data.**

- The information stored in the catalog is called **meta-data**, and it describes the structure of the primary database (Figure 1).

- It is important to note that some newer types of database systems, known as **NOSQL** systems, do not require meta-data.

- Rather the data is stored as **self-describing data** that includes the data item names and data values together in one structure.

Contd…

- The catalog is used by the DBMS software and also by database users who need information about the database structure.

- A general-purpose DBMS software package is not written for a specific database application.

- Therefore, it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access.

- The DBMS software must work equally well with *any number of database applications*—

- For example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog.

Traditional File Processing Systems

- In traditional file processing, data definition is typically part of the application programs themselves.

- Hence, these programs are constrained to work with only *one specific database,* whose structure is declared in the application programs.

- For example, an application program written in C++ may have struct or class declarations.

- Whereas file-processing software can access only specific databases, DBMS software can access diverse databases by extracting the database definitions from the catalog and using these definitions.

An example of a database catalog for the database in Figure 2

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 enumerated type with all known majors.
XXXXNNNN is used to define a type with four alphabetic characters followed by four numeric digits.

Figure 3

Database Catalog

- Figure 3 shows some entries in a database catalog.

- Whenever a request is made to access, say, the Name of a STUDENT record, the DBMS software refers to the catalog to determine the structure of the STUDENT file and the position and size of the Name data item within a STUDENT record.

- By contrast, in a typical file-processing application, the file structure and, in the extreme case, the exact location of Name within a STUDENT record are already coded within each program that accesses this data item.

Insulation between Programs and Data and Data Abstraction

- 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.

- By contrast, DBMS access programs do not require such changes in most cases.

- The structure of data files is stored in the DBMS catalog separately from the access programs.

- We call this property **program-data independence**.

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 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.

Internal Storage Format – Student Record

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.

Figure 4

Program Operation Independence

- In some types of database systems, such as object-oriented and object-relational systems, users can define operations on data as part of the database definitions.

- An **operation** (also called a *function* or *method*) is specified in two parts.

- The *interface* (or *signature*) of an operation includes the operation name and the data types of its arguments (or parameters).

- The *implementation* (or *method*) of the operation is specified separately and can be changed without affecting the interface.

- User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented.

- This may be termed **Program-operation independence**.

Data Abstraction

- The characteristic that allows **Program-data independence** and **Program-operation independence** is called **data abstraction**.

Data Model

- A DBMS provides users with a **conceptual representation** of data that does not include many of the details of how the data is stored or how the operations are implemented.

- Informally, a **data model** is a type of data abstraction that is used to provide this conceptual representation.

- The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts.

- Hence, the data model *hides* storage and implementation details that are not of interest to most database users.

Contd…

- Looking at the example in Figures 2 and 3, the internal implementation of the STUDENT file may be defined by its record length—the number of characters (bytes) in each record—and each data item may be specified by its starting byte within a record and its length in bytes.

- The STUDENT record would thus be represented as shown in Figure 4.

- But a typical database user is not concerned with the location of each data item within a record or its length;

- Rather, the user is concerned that when a reference is made to Name of STUDENT, the correct value is returned.

- A conceptual representation of the STUDENT records is shown in Figure 2.

- Many other details of file storage organization—such as the access paths specified on a file—can be hidden from database users by the DBMS.

Contd…

- In the database approach, the detailed structure and organization of each file are stored in the catalog.

- Database users and application programs refer to the conceptual representation of the files, and the DBMS extracts the details of file storage from the catalog when these are needed by the DBMS file access modules.

- Many data models can be used to provide this data abstraction to database users.

- A major part of our course is devoted to presenting various data models and the concepts they use to abstract the representation of data.

Contd…

- In object-oriented and object-relational databases, the abstraction process includes not only the data structure but also the operations on the data.

- These operations provide an abstraction of miniworld activities commonly understood by the users.

- For example, an operation CALCULATE_GPA can be applied to a STUDENT object to calculate the grade point average.

- Such operations can be invoked by the user queries or application programs without having to know the details of how the operations are implemented.

Support of Multiple Views of the Data

- A database typically has many types of users, each of whom may require a different perspective or **view** of the database.

- 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.

- Some users may not need to be aware of whether the data they refer to is stored or derived.

- A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

- For example, one user of the database of Figure 2 may be interested only in accessing and printing the transcript of each student; the view for this user is shown in Figure 5(a).

- A second user, who is interested only in checking that students have taken all the prerequisites of each course for which the student registers, may require the view shown in Figure 5(b).

Two views derived from the database in Figure 2 : Figure 5 (a) The TRANSCRIPT view. 5(b) The COURSE_PREREQUISITES view.

TRANSCRIPT

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

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

Sharing of Data and Multiuser Transaction Processing

- A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time.

- This is essential if data for multiple applications is to be integrated and maintained in a single database.

- The DBMS must include **concurrency control** software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.

- 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.

- These types of applications are generally called **online transaction processing (OLTP)** applications.

- A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.

Transaction/Isolation/Atomicity

- The concept of a **transaction** has become central to many database applications.

- A transaction is an *executing program* or *process* that includes one or more database accesses, such as reading or updating of database records.

- Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions.

- The DBMS must enforce several transaction properties.

- 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.

Actors on the Scene

- For a small personal database, such as the list of addresses discussed previously ,one person typically defines, constructs, and manipulates the database, and there is no sharing.

- However, in large organizations, many people are involved in the design, use, and maintenance of a large database with hundreds or thousands of users.

- In this section we identify the people whose jobs involve the day-to-day use of a large database; we call them the *actors on the scene*.

- In next section ,we consider people who may be called *workers behind the scene*—those who work to maintain the database system environment but who are not actively interested in the database contents as part of their daily job.

Database Administrators

- In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources.

- In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software.

- Administering these resources is the responsibility of the **Data Base Administrator (DBA)**.

- The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.

- The DBA is accountable for problems such as security breaches and poor system response time.

- In large organizations, the DBA is assisted by a staff that carries out these functions.

Database Designers

- **Database designers** are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.

- These tasks are mostly undertaken before the database is actually implemented and populated with data.

- It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

- In many cases, the designers are on the staff of the DBA and may be assigned other staff responsibilities after the database design is completed.

- Database designers typically interact with each potential group of users and develop **views** of the database that meet the data and processing requirements of these groups.

- Each view is then analyzed and *integrated* with the views of other user groups.

- The final database design must be capable of supporting the requirements of all user groups.

End Users

- **End users** are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use.

- There are several categories of end users:

Casual end users occasionally access the database, but they may need different information each time.

- They use a sophisticated database query interface to specify their requests and are typically middle- or high-level managers or other occasional browsers.

Naive or parametric end users

- **Naive** or **parametric end users** make up a sizable portion of database end users.
- Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates— called **canned transactions**—that have been carefully programmed and tested.
- Many of these tasks are now available as **mobile apps** for use with mobile devices.
- The tasks that such users perform are varied.
- **<u>A few examples are:</u>**
- Bank customers and tellers check account balances and post withdrawals and deposits.
- Reservation agents or customers for airlines, hotels, and car rental companies check availability for a given request and make reservations.
- Employees at receiving stations for shipping companies enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages.
- Social media users post and read items on social media Web sites.

Other End Users

Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.

Standalone users maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a financial software package that stores a variety of personal financial data.

Contd…

- 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 mobile apps or 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.

System Analysts and Application Programmers (Software Engineers)

- **System analysts** determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements.

- **Application programmers** implement these specifications as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers—commonly referred to as **software developers** or **software engineers**—should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

Workers behind the Scene

- In addition to those who design, use, and administer a database, others are associated with the design, development, and operation of the **DBMS** *software and system environment.*

- These persons are typically not interested in the database content itself.

- We call them the *workers behind the scene*, and they include the following categories:

DBMS system designers and implementers design and implement the DBMS modules and interfaces as a software package.

- A DBMS is a very complex software system that consists of many **components**, or **modules**,

- including modules for implementing the catalog, query language processing , interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security.

- The DBMS must interface with other system software, such as the operating system and compilers for various programming languages.

Contd…

- **Tool developers** design and implement **tools**—the software packages that facilitate database modeling and design, database system design, and improved performance.

- Tools are optional packages that are often purchased separately.

- They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation.

- In many cases, independent software vendors develop and market these tools.

- **Operators and maintenance personnel** (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system.

- Although these categories of workers behind the scene are instrumental in making the database system available to end users, they typically do not use the database contents for their own purposes.

DBMS Approach - Advantages

- In this section we discuss some additional advantages of using a DBMS and the capabilities that a good DBMS should possess.

- These capabilities are in addition to the four main characteristics discussed previously.

- The DBA must utilize these capabilities to accomplish a variety of objectives related to the design, administration, and use of a large multiuser database.

DBMS Approach - Advantages

- Controlling Redundancy

- Restricting Unauthorized Access

- Providing Persistent Storage for Program Objects

- Providing Storage Structures and Search Techniques for Efficient Query Processing

- Providing Backup and Recovery

- Providing Multiple User Interfaces

- Representing Complex Relationships among Data

- Enforcing Integrity Constraints

- Permitting Inferencing and Actions Using Rules and Triggers

- Additional Implications of Using the Database Approach

Controlling Redundancy

- In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications.
- For example, consider the UNIVERSITY database example; here, two groups of users might be the course registration personnel and the accounting office.
- In the traditional approach, each group independently keeps files on students.
- The accounting office keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades.
- Other groups may further duplicate some or all of the same data in their own files.
- This **redundancy** in storing the same data multiple times leads to several problems.

Contd…

- First, there is the need to perform a single logical update—such as entering data on a new student—multiple times: once for each file where student data is recorded.

- This leads to **duplication of effort**.

- Second, **storage space** *is wasted* when the same data is stored repeatedly, and this problem may be serious for large databases.

- Third, files that represent the same data may become **inconsistent**.

- This may happen because an update is applied to some of the files but not to others.

- Even if an update—such as adding a new student—is applied to all the appropriate files, the data concerning the student may still be *inconsistent* because the updates are applied independently by each user group.

- For example, one user group may enter a student's birth date erroneously as 'JAN-19-1988', whereas the other user groups may enter the correct value of 'JAN-29-1988'.

Contd…

- In the database approach, the views of different user groups are integrated during database design.

- Ideally, we should have a database design that stores each logical data item—such as a student's name or birth date—in *only one place* in the database.

- This is known as **data normalization**, and it ensures consistency and saves storage space.

- However, in practice, it is sometimes necessary to use **Controlled redundancy** to improve the performance of queries.

- For example, we may store Student_name and Course_number redundantly in a GRADE_REPORT file (Figure 6(a)) because whenever we retrieve a GRADE_REPORT record, we want to retrieve the student name and course number along with the grade, student number, and section identifier.

- By placing all the data together, we do not have to search multiple files to collect this data. This is known as **denormalization**.

Redundant storage of Student_name and Course_name in GRADE_REPORT : (a) Consistent data (b) Inconsistent record

GRADE_REPORT

Student_number	Student_name	Section_identifier	Course_number	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

(a)

GRADE_REPORT

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Brown	112	MATH2410	B

(b)

Contd…

- In such cases, the DBMS should have the capability to *control* this redundancy in order to prohibit inconsistencies among the files.

- This may be done by automatically checking that the Student_name –Student_number values in any GRADE_REPORT record in Figure 6(a) match one of the Name–Student_number values of a STUDENT record (Figure 2).

- Similarly, the Section_identifier – Course_number values in GRADE_REPORT can be checked against SECTION records.

- Such checks can be specified to the DBMS during database design and automatically enforced by the DBMS whenever the GRADE_REPORT file is updated.

- Figure 6(b) shows a GRADE_REPORT record that is inconsistent with the STUDENT file in Figure 2; this kind of error may be entered if the redundancy is *not controlled*.

- Can you tell which part is inconsistent?

Restricting Unauthorized Access

- When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database.

- For example, financial data such as salaries and bonuses is often considered confidential, and only authorized persons are allowed to access such data.

- In addition, some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update.

- Hence, the type of access operation—retrieval or update—must also be controlled.

- Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database.

- A DBMS should provide a **Security and Authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions.

- Then, the DBMS should enforce these restrictions automatically.

- Notice that we can apply similar controls to the DBMS software.

- For example, only the DBA's staff may be allowed to use certain **privileged software**, such as the software for creating new accounts.

- Similarly, parametric users may be allowed to access the database only through the predefined apps or canned transactions developed for their use.

Providing Persistent Storage for Program Objects

- Databases can be used to provide **Persistent storage** for program objects and data structures.

- This is one of the main reasons for **Object-oriented database systems**

- Programming languages typically have complex data structures, such as structs or class definitions in C++ or Java.

- The values of program variables or objects are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage.

- When the need arises to read this data once more, the programmer must convert from the file format to the program variable or object structure.

- Object-oriented database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.

- Hence, a complex object in C++ can be stored permanently in an object-oriented DBMS.

Contd…

- The persistent storage of program objects and data structures is an important function of database systems.

- Traditional database systems often suffered from the so called **impedance mismatch problem**,

- Since the data structures provided by the DBMS were incompatible with the programming language's data structures.

- Object-oriented database systems typically offer data structure **compatibility** with one or more object-oriented programming languages.

Providing Storage Structures and Search Techniques for Efficient Query Processing

- Database systems must provide capabilities for *efficiently executing queries and updates*.

- Because the database is typically stored on disk, the DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records.

- Auxiliary files called **indexes** are often used for this purpose.

- Indexes are typically based on tree data structures or hash data structures that are suitably modified for disk search.

- In order to process the database records needed by a particular query, those records must be copied from disk to main memory.

- Therefore, the DBMS often has a **buffering** or **caching** module that maintains parts of the database in main memory buffers.

- In general, the operating system is responsible for disk-to-memory buffering.

- However, because data buffering is crucial to the DBMS performance, most DBMSs do their own data buffering.

Contd…

- The **query processing and optimization** module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures.

- The choice of which indexes to create and maintain is part of *physical database design and tuning,* which is one of the responsibilities of the DBA staff.

Providing Backup and Recovery

- A DBMS must provide facilities for recovering from hardware or software failures.

- The **backup and recovery subsystem** of the DBMS is responsible for recovery.

- 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.

- **Disk backup** is also necessary in case of a catastrophic disk failure.

Providing Multiple User Interfaces

- Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.

- These include apps for mobile users, query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for standalone users.

- Both forms-style interfaces and menu-driven interfaces are commonly known as **graphical user interfaces (GUIs)**.

- Many specialized languages and environments exist for specifying GUIs.

- Capabilities for providing Web GUI interfaces to a database—or Web-enabling a database—are also quite common.

Representing Complex Relationships among Data

- A database may include numerous varieties of data that are interrelated in many ways.

- Consider the example shown in Figure 2.

- The record for 'Brown' in the STUDENT file is related to four records in the GRADE_REPORT file.

- Similarly, each section record is related to one course record and to a number of GRADE_REPORT records—one for each student who completed that section.

- A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

Enforcing Integrity Constraints

- Most database applications have certain **integrity constraints** that must hold for the data.

- A DBMS should provide capabilities for defining and enforcing these constraints.

- The simplest type of integrity constraint involves specifying a data type for each data item.

- For example, in Figure 3, we specified that the value of the Class data item with in each STUDENT record must be a one-digit integer and that the value of Name must be a string of no more than 30 alphabetic characters.

- To restrict the value of Class between 1 and 5 would be an additional constraint that is not shown in the current catalog.

- A more complex type of constraint that frequently occurs involves specifying that a record in one file must be related to records in other files.

- For example, in Figure 2, we can specify that *every section record must be related to a course record*.

- This is known as a **referential integrity** constraint.

Key / Semantics

- Another type of constraint specifies uniqueness on data item values, such as *every course record must have a unique value for Course_number*.
- This is known as a **key** or **uniqueness** constraint.
- These constraints are derived from the meaning or **semantics** of the data and of the miniworld it represents.
- It is the responsibility of the database designers to identify integrity constraints during database design.
- Some constraints can be specified to the DBMS and automatically enforced.
- Other constraints may have to be checked by update programs or at the time of data entry. For typical large applications, it is customary to call such constraints **business rules**.

Contd…

- A data item may be entered erroneously and still satisfy the specified integrity constraints.

- For example, if a student receives a grade of 'A' but a grade of 'C' is entered in the database, the DBMS *cannot* discover this error automatically because 'C' is a valid value for the Grade data type.

- Such data entry errors can only be discovered manually (when the student receives the grade and complains) and corrected later by updating the database.

- However, a grade of 'Z' would be rejected automatically by the DBMS because 'Z' is not a valid value for the Grade data type.

- When we discuss each data model in subsequent lectures , we will introduce rules that pertain to that model implicitly.

- For example, in the Entity-Relationship model, a relationship must involve at least two entities.

- Rules that pertain to a specific data model are called **inherent rules** of the data model.

Permitting Inferencing and Actions Using Rules and Triggers

- Some database systems provide capabilities for defining *deduction rules* for *inferencing* new information from the stored database facts.

- Such systems are called **deductive database systems**.

- For example, there may be complex rules in the miniworld application for determining when a student is on probation.

- These can be specified *declaratively* as **rules**, which when compiled and maintained by the DBMS can determine all students on probation.

- In a traditional DBMS, an explicit *procedural program code* would have to be written to support such applications.

- But if the miniworld rules change, it is generally more convenient to change the declared deduction rules than to recode procedural programs.

- In today's relational database systems, it is possible to associate **triggers** with tables.

- A trigger is a form of a rule activated by updates to the table, which results in performing some additional operations to some other tables, sending messages, and so on.

- More involved procedures to enforce rules are popularly called **stored procedures**; they become a part of the overall database definition and are invoked appropriately when certain conditions are met.

- More powerful functionality is provided by **active database systems**, which

Additional Implications of Using the Database Approach

- This section discusses a few additional implications of using the database approach that can benefit most organizations.

Potential for Enforcing Standards.

- The database approach permits the DBA to define and enforce standards among database users in a large organization.

- This facilitates communication and cooperation among various departments, projects, and users within the organization.

- Standards can be defined for names and formats of data elements, display formats, report structures, terminology, and so on.

- The DBA can enforce standards in a centralized database environment more easily than in an environment where each user group has control of its own data files and software.

Contd..

Reduced Application Development Time

- A prime selling feature of the database approach is that developing a new application—such as the retrieval of certain data from the database for printing a new report—takes very little time.

- Designing and implementing a large multiuser database from scratch may take more time than writing a single specialized file application.

- However, once a database is up and running, substantially less time is generally required to create new applications using DBMS facilities.

- Development time using a DBMS is estimated to be one sixth to one-fourth of that for a file system.

Contd…

Flexibility

- It may be necessary to change the structure of a database as requirements change.

- For example, a new user group may emerge that needs information not currently in the database.

- In response, it may be necessary to add a file to the database or to extend the data elements in an existing file.

- Modern DBMSs allow certain types of evolutionary changes to the structure of the database without affecting the stored data and the existing application programs.

Contd…

Availability of Up-to-Date Information.

- A DBMS makes the database available to all users.

- As soon as one user's update is applied to the database, all other users can immediately see this update.

- This availability of up-to-date information is essential for many transaction-processing applications, such as reservation systems or banking databases, and it is made possible by the concurrency control and recovery subsystems of a DBMS.

Contd…

Economies of Scale

- The DBMS approach permits consolidation of data and applications, thus reducing the amount of wasteful overlap between activities of data-processing personnel in different projects or departments as well as redundancies among applications.

- This enables the whole organization to invest in more powerful processors, storage devices, or networking gear, rather than having each department purchase its own (lower performance) equipment.

- This reduces overall costs of operation and management.

A Brief History of Database Applications

- We now give a brief historical overview of the applications that use DBMSs and how these applications provided the impetus for new types of database systems.

Early Database Applications Using Hierarchical and Network Systems

- Many early database applications maintained records in large organizations such as corporations, universities, hospitals, and banks.

- In many of these applications, there were large numbers of records of similar structure.

- For example, in a university application, similar information would be kept for each student, each course, each grade record, and so on.

- There were also many types of records and many interrelationships among them.

Contd…

- One of the main problems with early database systems was the inter mixing of conceptual relationships with the physical storage and placement of records on disk.

- Hence, these systems did not provide sufficient **data abstraction** and **program-data independence** capabilities.

- For example, the grade records of a particular student could be physically stored next to the student record.

- Although this provided very efficient access for the original queries and transactions that the database was designed to handle, it did not provide enough flexibility to access records efficiently when new queries and transactions were identified.

- In particular, new queries that required a different storage organization for efficient processing were quite difficult to implement efficiently.

- It was also laborious to reorganize the database when changes were made to the application's requirements.

Contd..

- Another shortcoming of early systems was that they provided only programming language interfaces.

- This made it time-consuming and expensive to implement new queries and transactions, since new programs had to be written, tested, and debugged.

- Most of these database systems were implemented on large and expensive mainframe computers starting in the mid-1960s and continuing through the 1970s and 1980s.

- The main types of early systems were based on three main paradigms: hierarchical systems, network model–based systems, and inverted file systems.

Providing Data Abstraction and Application Flexibility with Relational Databases

- Relational databases were originally proposed to separate the physical storage of data from its conceptual representation and to provide a mathematical foundation for data representation and querying.

- The relational data model also introduced high-level query languages that provided an alternative to programming language interfaces, making it much faster to write new queries.

- Relational representation of data somewhat resembles the example we presented in Figure 2.

- Relational systems were initially targeted to the same applications as earlier systems, and provided flexibility to develop new queries quickly and to reorganize the database as requirements changed.

- Hence, *data abstraction* and *program-data independence* were much improved when compared to earlier systems.

Contd…

- Early experimental relational systems developed in the late 1970s and the commercial relational database management systems (RDBMS) introduced in the early 1980s were quite slow, since they did not use physical storage pointers or record placement to access related data records.

- With the development of new storage and indexing techniques and better query processing and optimization, their performance improved.

- Eventually, relational databases became the dominant type of database system for traditional database applications.

- Relational databases now exist on almost all types of computers, from small personal computers to large servers.

Object-Oriented Applications and the Need for More Complex Databases

- The emergence of object-oriented programming languages in the 1980s and the need to store and share complex, structured objects led to the development of object-oriented databases (OODBs).

- Initially, OODBs were considered a competitor to relational databases, since they provided more general data structures.

- They also incorporated many of the useful object-oriented paradigms, such as abstract data types, encapsulation of operations, inheritance, and object identity.

- However, the complexity of the model and the lack of an early standard contributed to their limited use.

- They are now mainly used in specialized applications, such as engineering design, multimedia publishing, and manufacturing systems.

- Despite expectations that they will make a big impact, their overall penetration into the database products market remains low.

- In addition, many object-oriented concepts were incorporated into the newer versions of relational DBMSs, leading to object-relational database management systems, known as ORDBMSs.

Interchanging Data on the Web for E-Commerce Using XML

- The World Wide Web provides a large network of interconnected computers.

- Users can create static Web pages using a Web publishing language, such as Hyper-Text Markup Language (HTML), and store these documents on Web servers where other users (clients) can access them and view them through Web browsers.

- Documents can be linked through **hyperlinks**, which are pointers to other documents.

- Starting in the 1990s, electronic commerce (e-commerce) emerged as a major application on the Web.

- Much of the critical information on e-commerce Web pages is dynamically extracted data from DBMSs, such as flight information, product prices, and product availability.

- A variety of techniques were developed to allow the interchange of dynamically extracted data on the Web for display on Web pages.

- The eXtended Markup Language (XML) is one standard for interchanging data among various types of databases and Web pages.

- XML combines concepts from the models used in document systems with database modeling concepts.

Extending Database Capabilities for New Applications

- The success of database systems in traditional applications encouraged developers of other types of applications to attempt to use them.

- Such applications traditionally used their own specialized software and file and data structures.

- Database systems now offer extensions to better support the specialized requirements for some of these applications.

- The following are some examples of these applications:

Scientific applications that store large amounts of data resulting from scientific experiments in areas such as high-energy physics, the mapping of the human genome, and the discovery of protein structures

Storage and retrieval of images, including scanned news or personal photographs, satellite photographic images, and images from medical procedures such as x-rays and MRI (magnetic resonance imaging) tests

Contd…

Storage and retrieval of videos, such as movies, and **video clips** from news or personal digital cameras

Data mining applications that analyze large amounts of data to search for the occurrences of specific patterns or relationships, and for identifying

unusual patterns in areas such as credit card fraud detection

Spatial applications that store and analyze spatial locations of data, such as weather information, maps used in geographical information systems, and automobile navigational systems

Time series applications that store information such as economic data at regular points in time, such as daily sales and monthly gross national product figures

Contd…

- It was quickly apparent that basic relational systems were not very suitable for many of these applications, usually for one or more of the following reasons.
✔ More complex data structures were needed for modeling the application than the simple relational representation.
✔ New data types were needed in addition to the basic numeric and character string types.
✔ New operations and query language constructs were necessary to manipulate the new data types.
✔ New storage and indexing structures were needed for efficient searching on the new data types.

Contd…

- This led DBMS developers to add functionality to their systems.

- Some functionality was general purpose, such as incorporating concepts from object-oriented databases into relational systems.

- Other functionality was special purpose, in the form of optional modules that could be used for specific applications.

- For example, users could buy a time series module to use with their relational DBMS for their time series application.

When Not to Use a DBMS ?

- In spite of the advantages of using a DBMS, there are a few situations in which a DBMS may involve unnecessary overhead costs that would not be incurred in traditional file processing.

- The overhead costs of using a DBMS are due to the following:

■ High initial investment in hardware, software, and training

■ The generality that a DBMS provides for defining and processing data

■ Overhead for providing security, concurrency control, recovery, and integrity functions

Contd…

- Therefore, it may be more desirable to develop customized database applications under the following circumstances:

❑ Simple, well-defined database applications that are not expected to change at all

❑ Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead

❑ Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit

❑ No multiple-user access to data

Summary

In this Lecture,

- We defined a database as a collection of related data, where *data* means recorded facts.

- A typical database represents some aspect of the real world and is used for specific purposes by one or more groups of users.

- A DBMS is a generalized software package for implementing and maintaining a computerized database.

- The database and software together form a database system.

- We identified several characteristics that distinguish the database approach from traditional file-processing applications, and we discussed the main categories of database users, or the *actors on the scene*.

- We noted that in addition to database users, there are several categories of support personnel, or *workers behind the scene*, in a database environment.

Contd…

- We presented a list of capabilities that should be provided by the DBMS software to the DBA, database designers, and end users to help them design, administer, and use a database.

- Then we gave a brief historical perspective on the evolution of database applications.

- We pointed out the recent rapid growth of the amounts and types of data that must be stored in databases, and we discussed the emergence of new systems for handling "big data" applications.

- Finally, we discussed the overhead costs of using a DBMS and discussed some situations in which it may not be advantageous to use one.