Database Management System (MDS 505)

Jagdish Bhatta

Unit-1

Fundamental Concept of DBMS

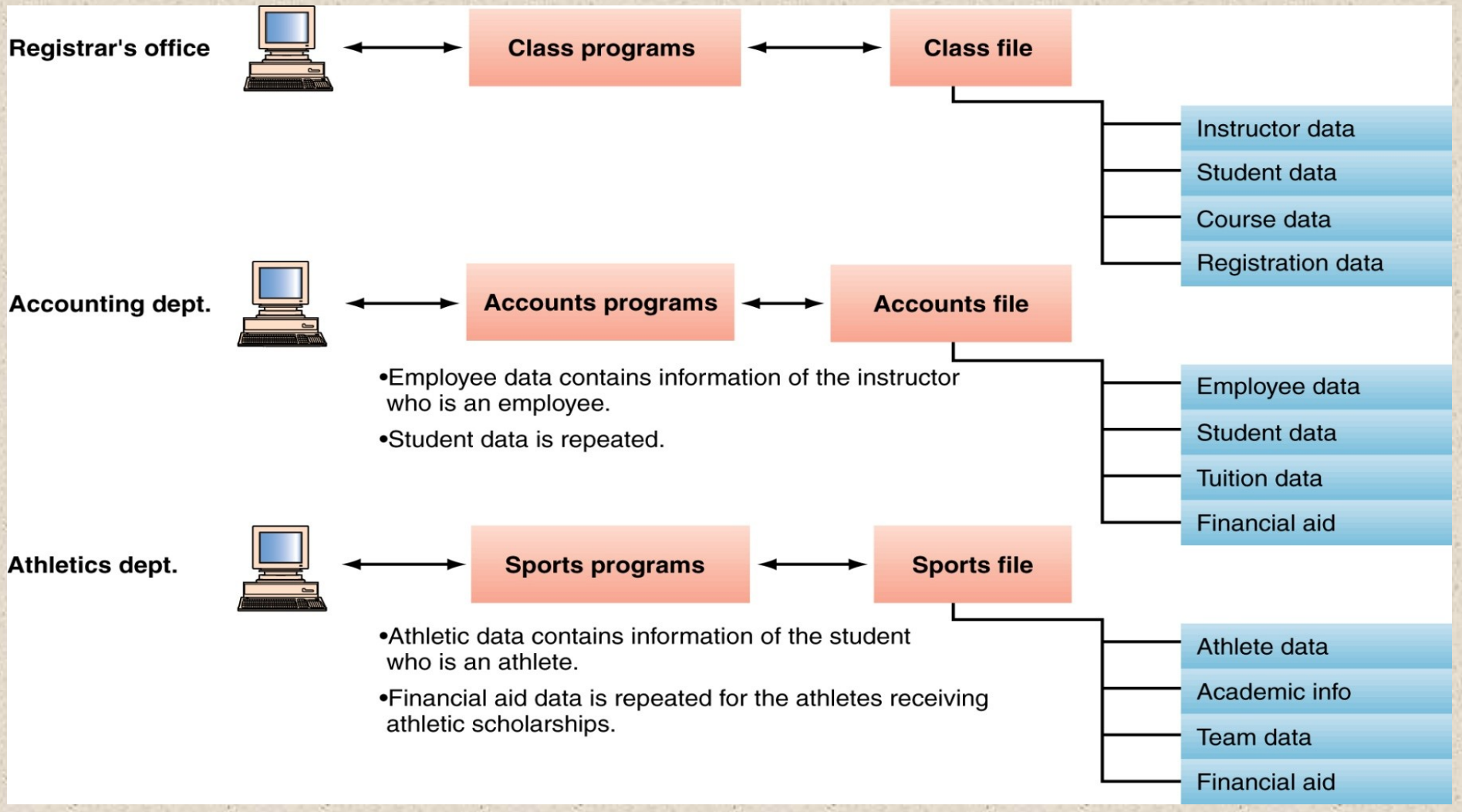
The Data Hierarchy

- **Byte -** 1…8 bits => 1 byte => 1 character
- **Field -** a logical grouping of characters into a word, or a small group of words is called record. It is analogous to column of a table.
- **Record** - a logical grouping of related fields is called record. It is analogous to row of a table.
- **File -** a logical grouping of related records is called file.
- **Database -** a logical grouping of related files is called database.

Traditional File Environment

- A data file is a collection of logically related records. In the traditional file management environment, each application has a specific data file related to it, containing all the data records needed by the application. It stores data in plain text files and is also called flat file system. This type of database is ideal for simple databases that do not contain a lot of repeated information.
- Examples include **excel spreadsheet or word data** list file.

Traditional File Environment



Problems in Traditional File Environment

- Keeping organizational information in a file-processing system has a number of major disadvantages:
  - **Data redundancy**: The address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost.
  - **Data inconsistency**: The various copies of the same data may no longer agree. For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.

Problems in Traditional File Environment

- **Difficulty in accessing data:** Conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner.
- **Data isolation**: Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- **Integrity problems**: The problem of integrity is the problem of ensuring that the data in database is correct after and before the transaction. For example, the balance of a bank account may never fall below a prescribed amount (say, $25). When new constraints are added, we have to change the programs to enforce them.

Problems in Traditional File Environment

- **Atomicity problems:** Execution of transactions must be atomic. This means transactions must execute at its entirety or not at all. Consider a program to transfer \$50 from account *A* to account *B*. If a system failure occurs during the execution of the program, it is possible that the \$50 was removed from account *A* but was not credited to account *B*, resulting in an inconsistent database state. It is difficult to ensure atomicity in a conventional file-processing system.
- **Concurrent-access anomalies**: Concurrent updates may result in inconsistent data. Consider bank account *A*, containing \$500. If two customers withdraw funds (say \$50 and \$100 respectively) from account *A* at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state, if the programs executing on behalf of each withdrawal read the old balance

Problems in Traditional File Environment

- **Security problems**: Not every user of the database system should be able to access all the data. For example, in a banking system, payroll personnel need to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts. But, since application programs are added to the system in an ad hoc manner, enforcing such security constraints is difficult in flat file system.

Database

- **Database** is a collection of *persistent data,* that have some implicit meaning, and is used by the application systems of some given *enterprise*. A database has the following implicit properties:
  - A database represents some aspects of the real world called universe of discourse.
  - A database is a logically coherent collection of data with some inherent meaning.
  - A database is designed, built and populated with data for a specific purpose.
- DB may be generated & maintained manually or it may be computerized.

Database

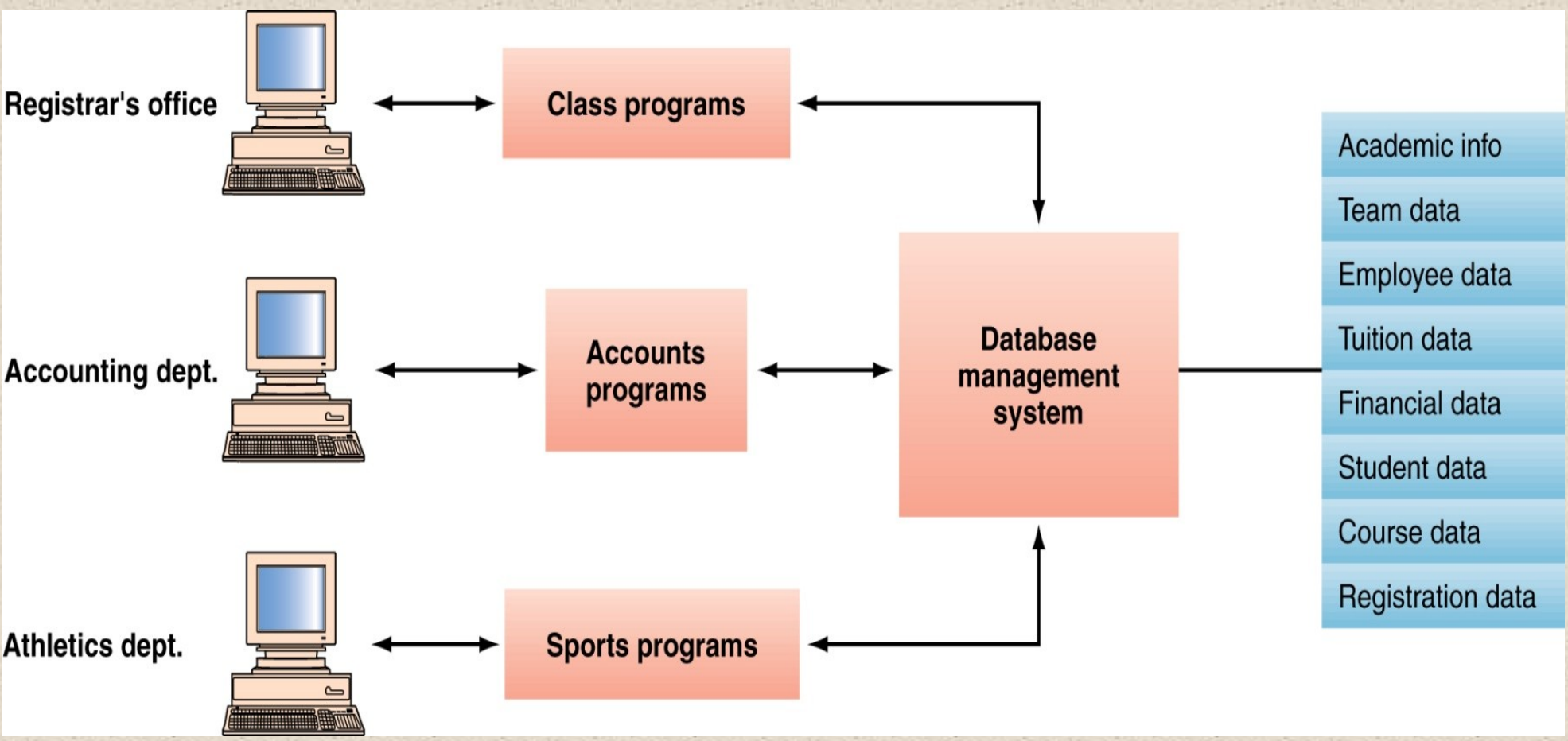
- A **database** is a collection of related data. By **data**, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. Nowadays, this data is typically stored in mobile phones, which have their 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.

Database

E.g.:- ***University database*** for maintaining information about students, courses and grades in university.

- Data consists of information about:
  - Students
  - Instructors
  - Classes etc.
- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

Database


Registrar's office
Class programs
Accounting dept.
Accounts programs
Database management system
Athletics dept.
Sports programs
Academic info
Team data
Employee data
Tuition data
Financial data
Student data
Course data
Registration data

Simplified Database System Environment

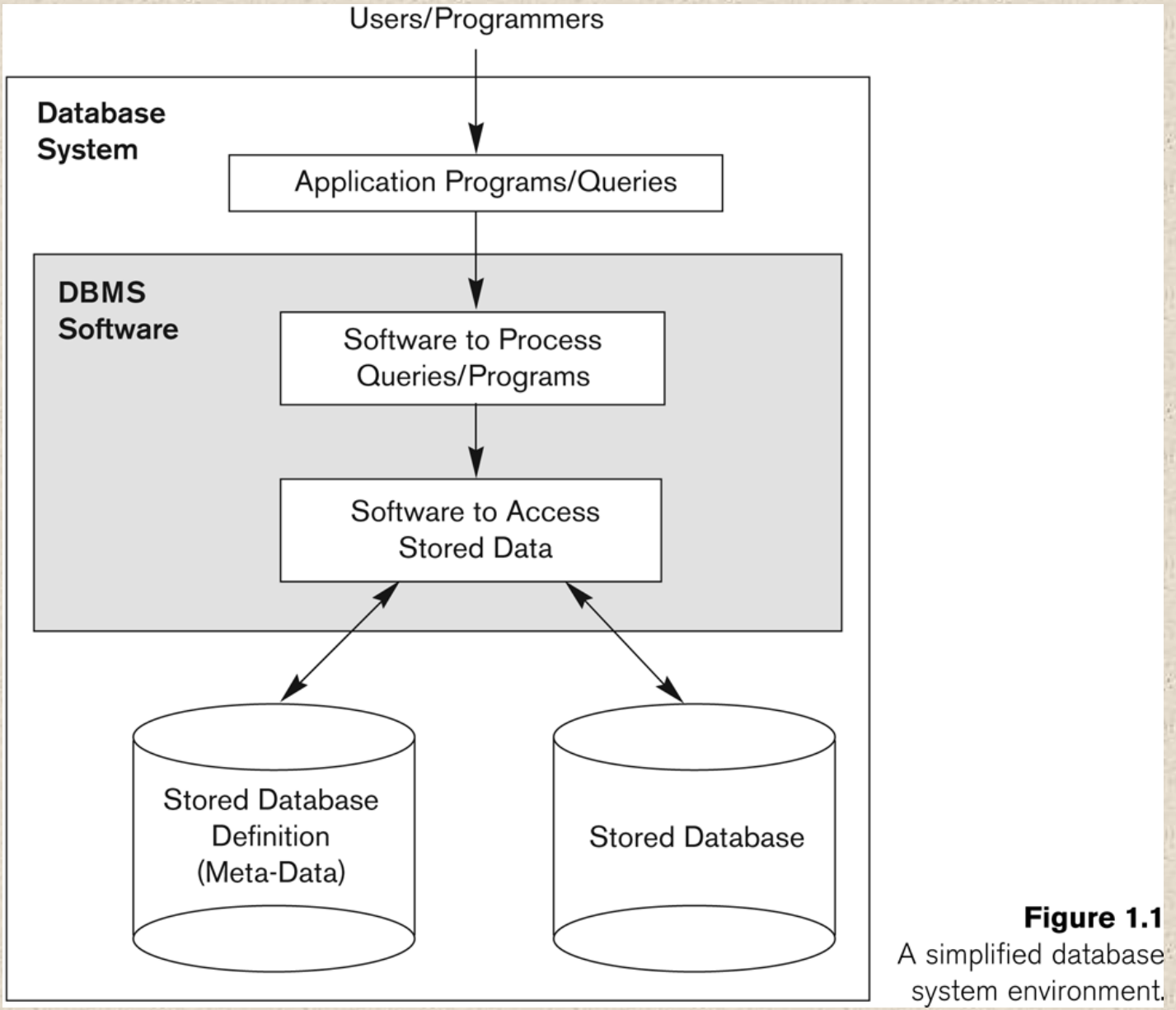


Figure 1.1
A simplified database system environment.

Database Management System [DBMS]

- The ***database management system*** (DBMS) is the software that handles all access to the database.
- It is defined as the collection of interrelated data and a set of programs to access those data. It enables users to create and maintain database. Simply, it is a database manager.
- **This general purpose software system includes process of *defining, constructing, manipulating and sharing* database among various users and applications.**

Database Management System [DBMS]

- **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.

Database Management System [DBMS]

- 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 Management System [DBMS]

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- 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.
- For example: MSAccess, Oracle, MySql, MSSql, Cassandra, MongoDB etc.

Example Of A Simple Database

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.

Main Characteristics of the Database Approach

- **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 data is stored as **self-describing data** that includes the data item names and data values together in one structure.
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.

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.

Main Characteristics of the Database Approach

- **Insulation between programs and data:**
  - 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**.
  - 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**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Main Characteristics of the Database Approach (continued)

- **Data Abstraction:**
  - The characteristic that allows program-data independence and program-operation independence is called **data abstraction**. 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.

Main Characteristics of the Database Approach (continued)

- **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 1.2 (in slide number 19) 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) in next slide. 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 1.5(b) in next slide.

Main Characteristics of the Database Approach (continued)

- **Support of multiple views of the data:**

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.

Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user 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.

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.
- **Providing persistent storage for program Objects**
  - In Object-oriented DBMSs
- **Providing Storage Structures** (e.g. indexes) for efficient Query Processing

Advantages of Using the Database Approach

- Providing **backup and recovery** services.
- Providing **multiple user interfaces** to different classes of users.
- Representing **complex relationships** among data.
- Enforcing **integrity constraints** on the database.
- Drawing **inferences and actions from the stored data using rules and triggers**

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.

Additional Implications of Using the Database Approach (continued)

- 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.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

Applications Areas of Database System

- **Databases form an essential part of almost all enterprises. Some applications include:**
  - ***Banking:*** For customer information, accounts, and loans, and banking transactions.
  - ***Airlines:*** For reservation and schedule information.
  - ***Universities:*** For student information, course registrations, and grades.
  - ***Credit card transactions:*** For purchase on credit cards and generation of monthly statements.
  - ***Telecommunication:*** For keeping records of call made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

- ***Finance:*** For storing information about holdings, sales, and purchase of financial instruments such as stocks and bonds.
- ***Sales:*** For customer, product, and purchase information
- ***Manufacturing***: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.
- ***Human resources:*** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.
- **And many more …**