Chapter 1

Introduction To Database Systems

Data:

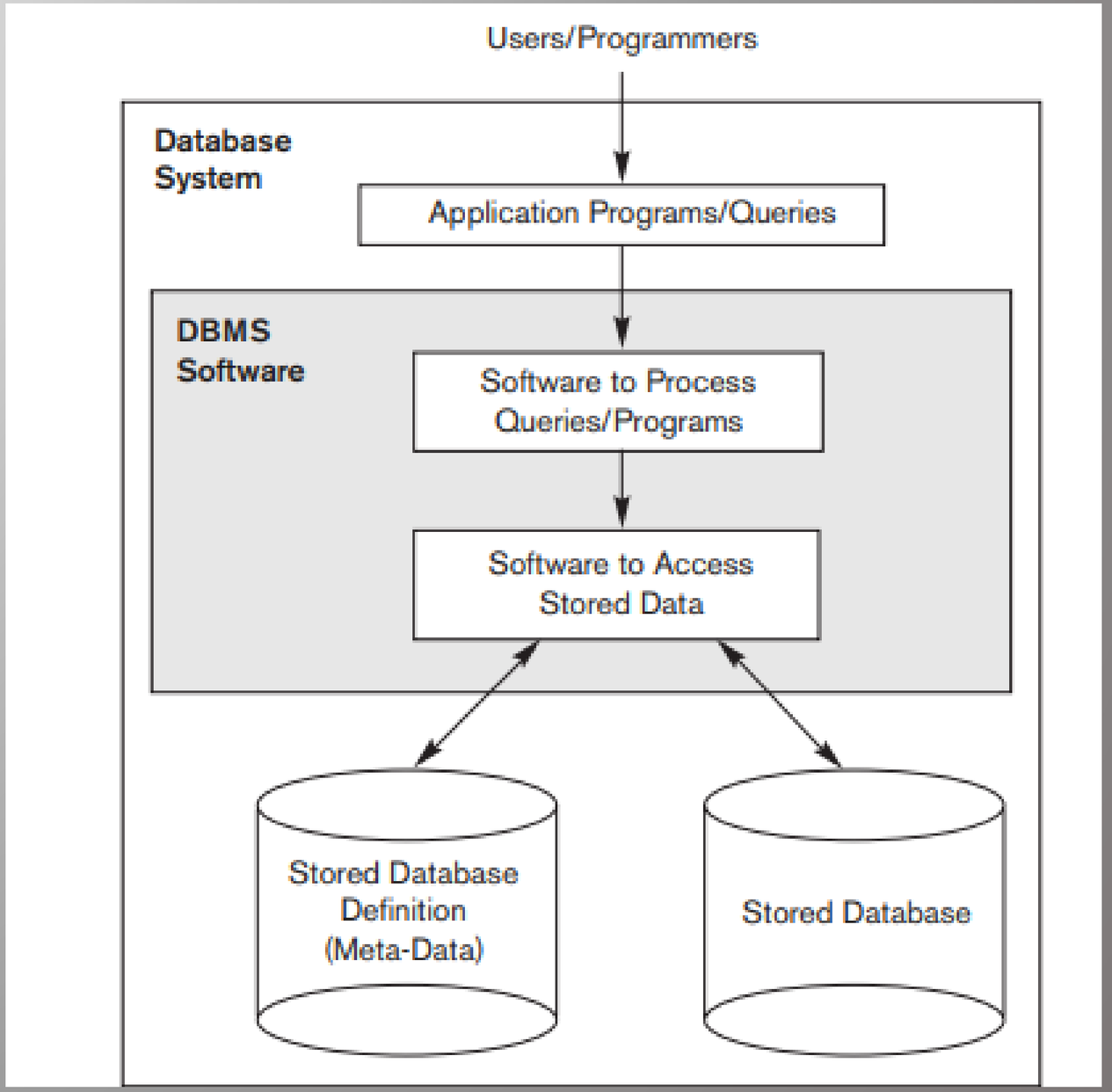
- Known facts that can be recorded which have implicit meanings
- Example: Name , Phone number

Database:

- Database is the collection of related data .
- Example : ?

Database Management System:

- Software system helps the process of defining, constructing, manipulate and sharing database among various users and applications


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

Characteristics of the database approach

The main characteristics of the database approach are :

- 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

1. **Self-Describing Nature of a Database System**

Meta data:

- 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**

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

2. Insulation between programs and data, and data abstraction :

- The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property **program-data independence.**
- User application programs can operate on the data by invoking operations through their names and arguments, regardless of how the operations are implemented. This may be termed **program-operation independence**
- The characteristic that allows program-data independence and program-operation independence is called data abstraction.

Data model

- A data model is a type of data abstraction that is used to provide the conceptual representation.
- The data model uses logical concepts, such as objects, their properties, and their interrelationships.
- The data model hides storage and implementation details .

3.Support of Multiple Views of the Data

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

4.Sharing of Data and Multiuser Transaction Processing

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

Database Administrators

- T he 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.

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

End Users

- Casual end users

They 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 user

Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called **scanned transactions**—that have been carefully programmed and tested.

Few examples are:

- 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

- **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.

Advantages of Using the DBMS Approach

- 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

Controlling Redundancy

- Redundancy in storing the same data multiple times leads to several problems
- 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.
- Inconsistent of data when there is mismatch.

Data Normalization:

The views of different user groups are integrated during database design. The 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

For Example :Data of birth, Age

Denormalization:

Controlled redundancy is used to improve the performance of queries. For example, we may store Student_name and Course_number redundantly in a GRADE_REPORT 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**

- **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.

- **Providing Persistent Storage for Program Objects**

Databases can be used to provide persistent storage for program objects and data structures. 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.

- **Providing Storage Structures and Search Techniques for Efficient Query Processing**

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.

- **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).

- **Representing Complex Relationships among Data**

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.

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, Figure can specify that every section record must be related to a course record. This is known as a **referential integrity constraint**. 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.

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

- **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.

Additional Implications of Using the Database Approach

- Potential for Enforcing Standards
- Reduced Application Development Time.
- Flexibility
- Availability of Up-to-Date Information
- Economies of Scale

When Not to Use a DBMS

- 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