Introduction to Database Systems

Notes by Mannan Ul Haq (BDS-4A)

Types of Databases and Database Applications:

- **Databases:**
  - Organized collections of data.
  - Types: Relational, NoSQL, Object-oriented, etc.
- **Database Applications:**
  - Software programs interacting with databases.
  - Examples: Inventory systems, Flex software, etc.

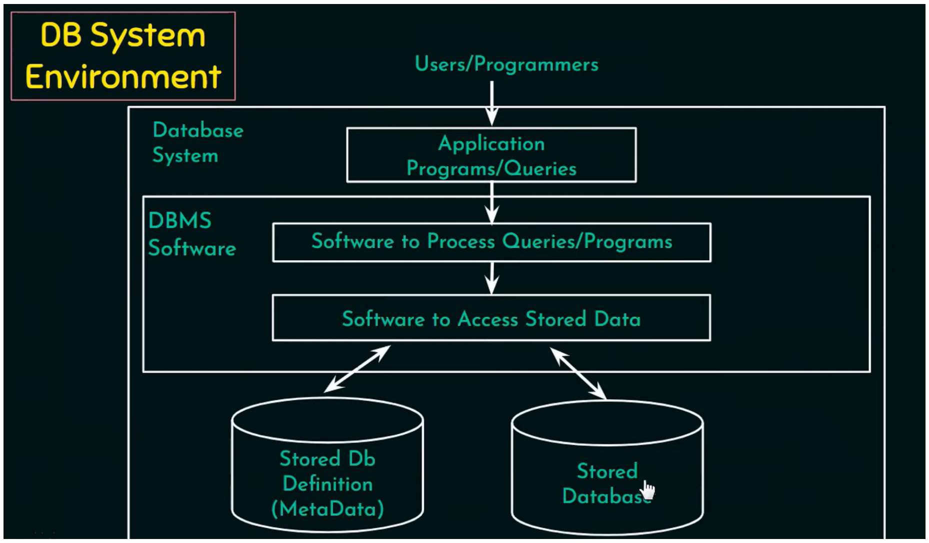
Basic Definitions:

- **Data:**
  - Raw facts and figures.
- **Information:**
  - Processed and meaningful data.
- **Database:**
  - Structured collection of related data.
- **Meta-data:**
  - The database definition. Metadata is data about data. In databases, it describes the structure, content, and properties of stored data, aiding in organization, understanding, and governance.

DBMS Functionalities:

- **Define:**
  - Specifying the data type, structures and constraints for the data to be stored.
- **Construct:**
  - Process of storing data on some storage medium.
- **Manipulate:**
  - Querying the database to retrieve specific data, updating database and generating reports.
- **Share:**
  - Allows multiple users and programs to access the database concurrently.

DB System Environment:



DB System Environment

Users/Programmers

Database System

Application Programs/Queries

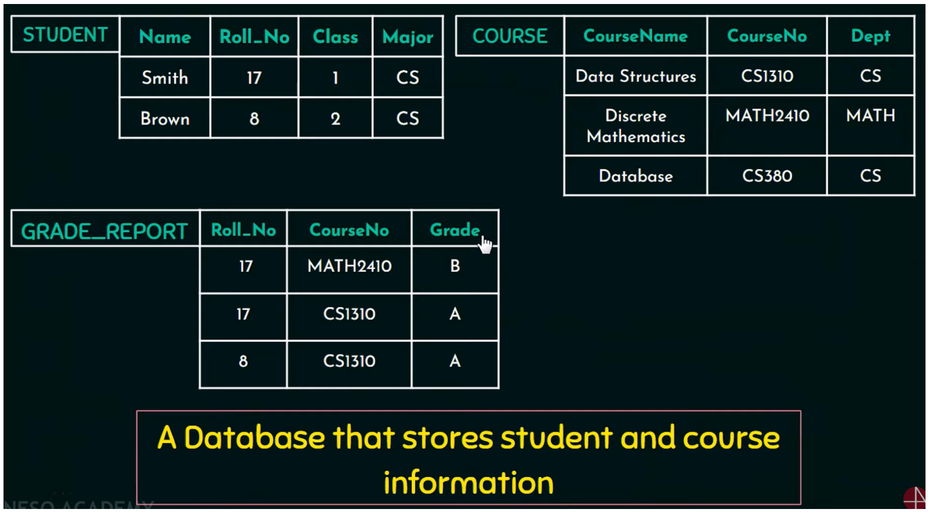
DBMS Software

Software to Process Queries/Programs

Software to Access Stored Data

Stored Db Definition (MetaData)

Stored Database



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

COURSE	CourseName	CourseNo	Dept
	Data Structures	CS1310	CS
	Discrete Mathematics	MATH2410	MATH
	Database	CS380	CS

GRADE_REPORT	Roll_No	CourseNo	Grade
	17	MATH2410	B
	17	CS1310	A
	8	CS1310	A

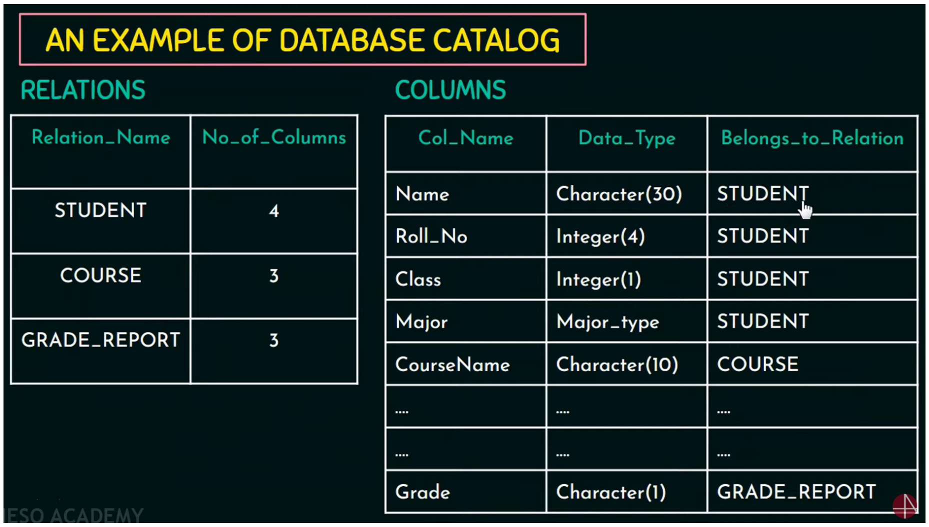
A Database that stores student and course information

Main Characteristics of the Database Approach:

- **Integration:**
  - Data centralized in one place.
- **Non-redundancy:**
  - Minimizing data duplication.
- **Data Integrity:**
  - Ensuring data accuracy and consistency.
- **Multiple Views:**
  - Support of multiple views of the data.
- **Multiuser Transaction:**
  - Sharing of data and multiuser transaction processing.

Database Catalog:

A catalog is a repository storing metadata about database objects like tables and indexes.



AN EXAMPLE OF DATABASE CATALOG

RELATIONS

Relation_Name	No_of_Columns
STUDENT	4
COURSE	3
GRADE_REPORT	3

COLUMNS

Col_Name	Data_Type	Belongs_to_Relation
Name	Character(30)	STUDENT
Roll_No	Integer(4)	STUDENT
Class	Integer(1)	STUDENT
Major	Major_type	STUDENT
CourseName	Character(10)	COURSE
....	....	....
....	....	....
Grade	Character(1)	GRADE_REPORT

Types of Database Users:

1. Actors on the Scene:

a.

Database Administrators (DBAs): Manage and maintain the database system, ensuring its efficiency, security, and reliability.

b. **Database Designers:** Plan and create the structure of the database, determining how data is organized and stored.

c. **End Users:**

-

Casual Users: Interact occasionally, needing simple access for basic information.

-

Naïve Users: Rely on pre-defined queries or forms without in-depth knowledge.

-

Sophisticated Users: Have a good understanding and utilize advanced features.

-

Stand-alone End Users: Work independently with personal databases.

d. **System Analysts & Application Programmers (Software Engineers):**

-

System Analysts: Analyze information systems, ensuring they meet organizational needs.

-

Application Programmers: Develop software applications that interact with databases.

2. Workers behind the Scene:

a.

System Designers & Implementers: Develop the overall structure and functionality of the information system, including the database.

b. **Tool Developers:** Create tools and utilities for database design, development, and maintenance.

c. **Operators & Maintenance Personnel:** Handle the day-to-day operation of the database system, ensuring it runs smoothly and efficiently. Conduct routine maintenance and troubleshooting.

Advantages of Using the Database Approach:

- **Data Consistency:**
  - Uniform and accurate data.
- **Data Security:**
  - Controlled access.
- **Data Independence:**
  - Applications separate from data storage.

Extending Database Capabilities:

- **Data Warehousing:**
  - Centralized data for analysis.
- **Data Mining:**
  - Discovering patterns in data.

When Not to Use Databases:

- **Small-scale Projects:**
  - Simple projects may not need the complexity of a database.
- **Budget Constraints:**
  - Building and maintaining databases can be expensive.
- **Technical Constraints:**
  - If the technical requirements don't align with database benefits.

Database System Concepts and Architecture

Data Model:

- A framework describing a database's structure, operations for manipulating data, and applicable constraints.

Data Model Structure and Constraints:

- Constructs define the database's structure, including elements (with data types) and groups (e.g., entities, tables), and their relationships.
- Constraints impose rules on valid data, enforced consistently.

Data Model Operations:

- Operations facilitate database retrieval and updates, referring to the data model's constructs.
- Includes basic operations (e.g., insert, delete, update) and user-defined operations (e.g., compute_student_gpa, update_inventory).

Categories of Data Models:

1. Conceptual (High-level) Data Models:

- **What They Are:** These are high-level representations of how users perceive data.
- **In Simple Terms:** Think of it like drawing a map of what you want your data to look like.
- **Example:** If you're designing a library system, you'd have entities like "Books," "Authors," and "Borrowers," with relationships like "Author writes Book" and "Borrower borrows Book."

2. Physical (Low-level) Data Models:

- **What They Are:** These are about how the data is actually stored inside the computer.
- **In Simple Terms:** Imagine it's like figuring out how to organize your clothes in your closet.
- **Example:** In a library system, it's like deciding whether to keep books on shelves sorted by genre or by author's last name.

3. Implementation (Representational) Data Models:

- **What They Are:** These are sort of in-between, used by the database system itself.
- **In Simple Terms:** It's like the blueprint used by construction workers when building a house.
- **Example:** For the library system, it's the detailed plan that says exactly how each book's information is stored in the computer memory.

4. Self-Describing Data Models:

- **What They Are:** These models combine the description of data with the actual data itself.
- **In Simple Terms:** It's like having labels on jars telling you what's inside, instead of needing to peek inside to find out.
- **Example:** In a library system, instead of just having a list of books, you'd have information about the books stored alongside the actual book data.

Database Schema:

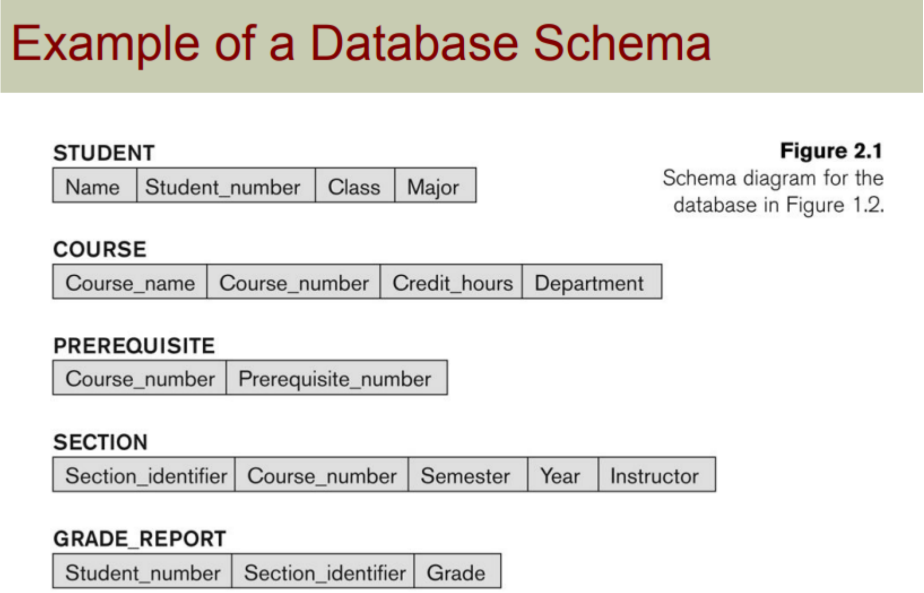
- It's like the description of a database.
- It tells you how the database is structured, what kinds of data it holds, and any rules it follows.

Schema Diagram:

- It's a picture or visual representation of the database schema.
- It's like drawing a map that shows all the different parts of the database and how they're connected.

Schema Construct:

- A component of the schema or an object within the schema, e.g., STUDENT, COURSE.



Example of a Database Schema

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Figure 2.1
Schema diagram for the database in Figure 1.2.

Database State:

- It's the actual data stored in the database at a specific moment.
- It's like taking a photo of your closet - it shows exactly how everything is arranged right now. It is also known as database instance.
- **Example:** If the library database has a record of all the books currently checked out and who has them, that's the database state.

Initial Database State:

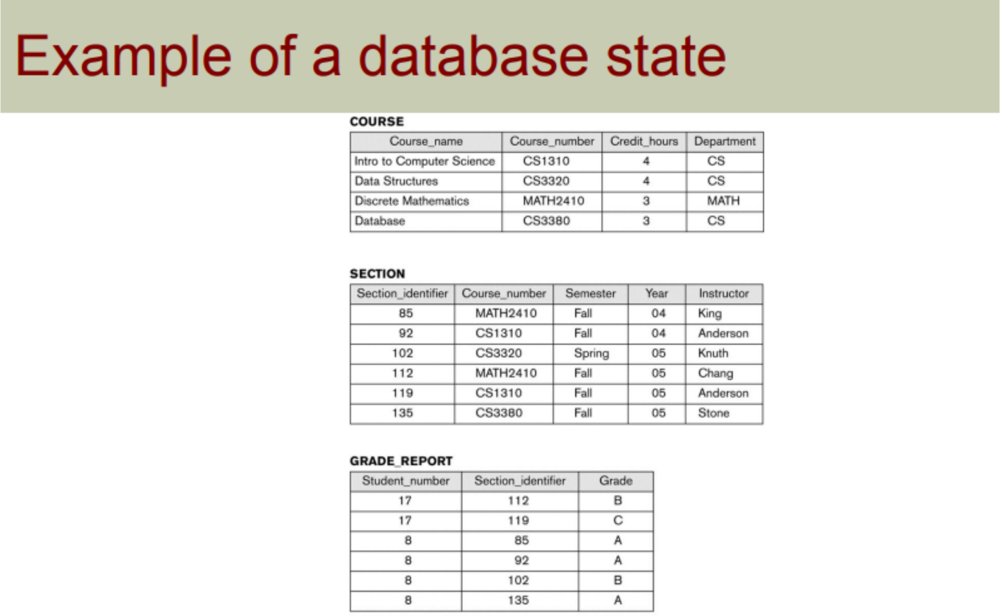
- It's the state of the database when it's first set up or loaded into the system.

Valid State:

- It's a database state that follows all the rules and structure laid out in the schema.

Distinction:

- The schema changes rarely, if at all, while the database state can change every time new data is added or existing data is updated.



Example of a database state

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	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	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

Three-Schema Architecture:

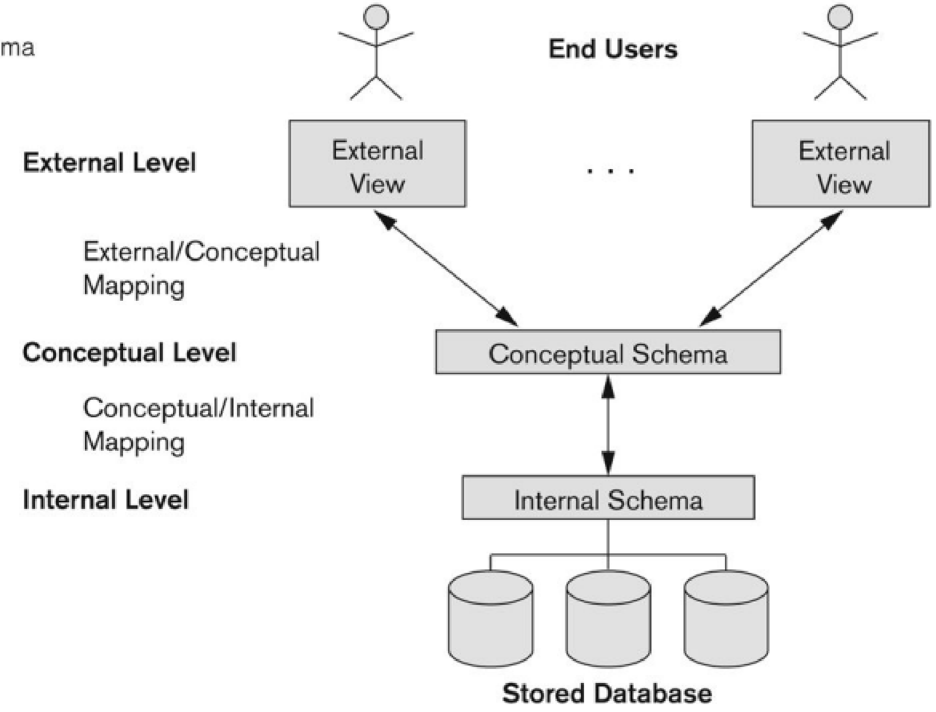
Purpose:

- Proposed to fulfill characteristics of DBMS:
  - Program-data independence.
  - Support for multiple views of data.

Description:

- Defines DBMS schemas at three levels:
  1. **Internal Schema:**
     - Describes physical storage structures and access paths (e.g., indexes).
     - Typically employs a physical data model.
  2. **Conceptual Schema:**
     - Describes the overall structure and constraints for the entire database for a community of users.
     - Utilizes a conceptual or implementation data model.
  3. **External Schemas:**
     - Describe various user views.
     - Generally utilizes the same data model as the conceptual schema.

Figure 2.2
The three-schema architecture.



Data Independence:

Logical Data Independence:

- **Definition:** The ability to modify the conceptual schema without needing to change the external schemas and their associated application programs.
- **Example:** You can change how data is structured or organized in the database without needing to update all the applications that use that data.

Physical Data Independence:

- **Definition:** The ability to alter the internal schema without needing to change the conceptual schema.
- **Example:** You can rearrange how data is stored on the disk or add new indexes to improve performance without affecting how users interact with the data.

Impact:

- When a lower-level schema changes, only the mappings between this schema and higher-level schemas need updating in a DBMS that fully supports data independence.
- Higher-level schemas remain unchanged, so application programs don't need modification since they reference external schemas.

DBMS Languages:

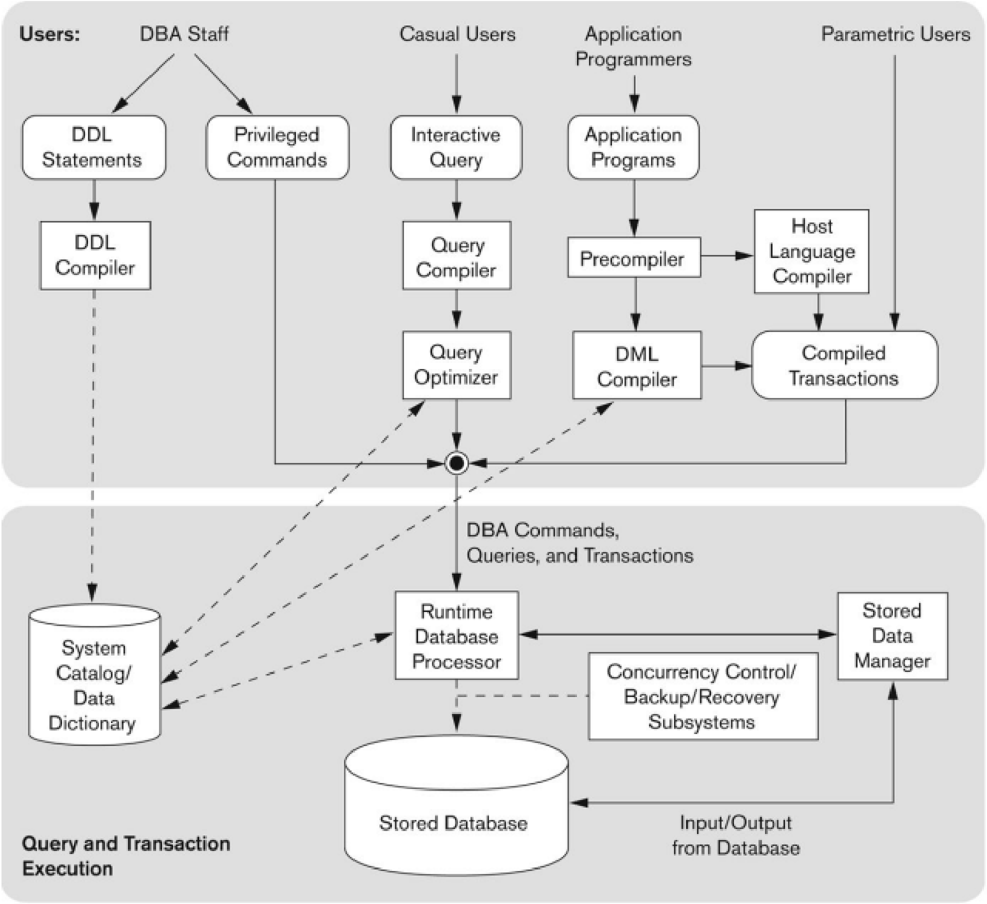
Data Definition Language (DDL):

- **Purpose:** Used by Database Administrators (DBA) and designers to define the conceptual schema of a database.
- **Functions:**
    - Specifies the structure of the database, including tables, columns, data types, and constraints.
    - In many DBMSs, also defines internal and external schemas (views).
- **Implementation:**
    - Some DBMSs use separate languages for storage definition (SDL) and view definition (VDL).
    - SDL is typically realized through commands provided to the DBA and designers.

Data Manipulation Language (DML):

- **Purpose:** Used for specifying database retrievals and updates.
- **Functions:**
    - Executes queries to retrieve specific data from the database.
    - Executes commands to update, insert, or delete data in the database.
- **Implementation:**
    - DML commands can be embedded within general-purpose programming languages like COBOL, C, C++, or Java.
    - Alternatively, standalone DML commands can be directly applied (referred to as a query language).
    - Some DBMSs provide a library of functions for accessing the DBMS from a programming language.

Typical DBMS Component Modules



Centralized DBMS Architecture:

- In a centralized architecture, all database processing is done on a single server or mainframe.
- Clients send requests to the central server, which handles data storage, retrieval, and manipulation.

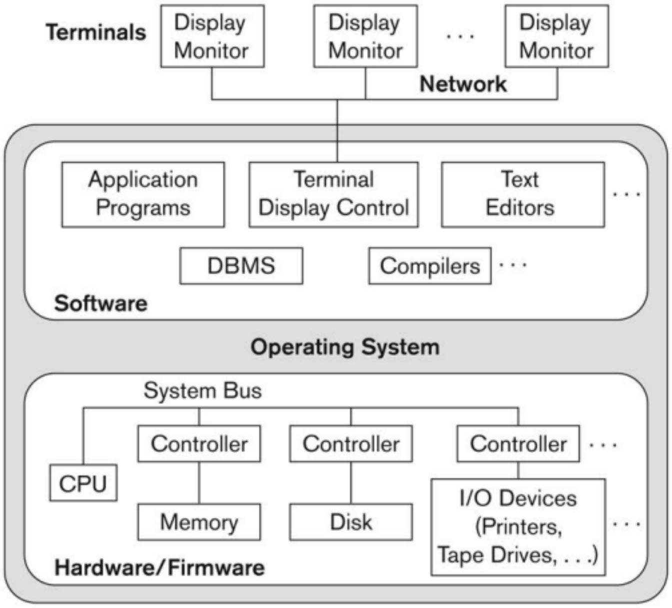


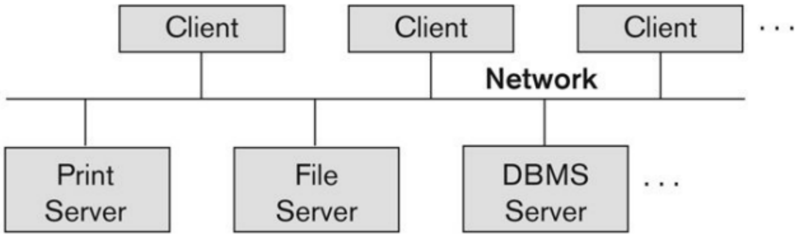
Figure 2.4
A physical centralized architecture.

Client-Server DBMS Architecture:

- In a client-server architecture, database processing is distributed between clients and servers.
- Clients, such as desktop computers or mobile devices, interact with the database through a client application.
- Servers manage data storage and processing, handling requests from clients and executing database operations.

Basic 2-tier Client-Server Architecture:

- In a basic 2-tier client-server architecture, there are two main components: the client and the server.
- The client is responsible for presenting the user interface and processing user requests.
- The server is responsible for managing data storage and processing requests from clients.



3-tier Client-Server Architecture:

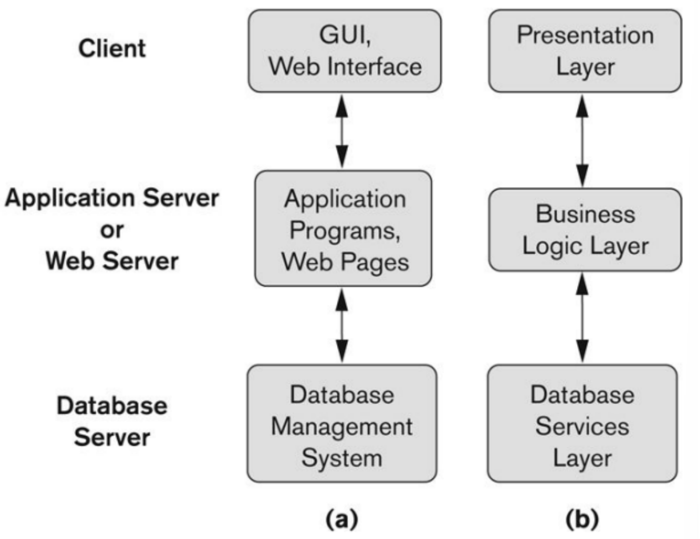
- In a basic 3-tier client-server architecture, the system is divided into three tiers: presentation tier, application tier, and data tier.

- Each tier has distinct responsibilities and communicates with the other tiers to fulfill user requests.

Characteristics:

- **Presentation Tier (Client Tier):** Responsible for presenting the user interface to the client.
- **Application Tier (Middle Tier):** Responsible for business logic and processing user requests.
- **Data Tier (Server Tier):** Responsible for managing data storage and performing database operations.

Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.


Client
GUI, Web Interface
Presentation Layer
Application Server or Web Server
Application Programs, Web Pages
Business Logic Layer
Database Server
Database Management System
Database Services Layer
(a)
(b)