Module 1
Database Systems Concepts and Architecture

Need for database systems – Characteristics of Database Approach – Advantages of using DBMS approach - Actors on the Database Management Scene: Database Administrator - Classification of database management systems- **Data Models – Schemas and Instances - Three-Schema Architecture - The Database System Environment -Centralized and Client/Server Architectures for DBMSs – Overall Architecture of Database Management Systems**

Reference :

- ***R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7th Edition, 2016***
- ***A. Silberschatz, H. F. Korth & S. Sudarshan, Database System Concepts, McGraw Hill,7th Edition 2019.***

Data Models

- **Definition :** A collection of concepts that can be used to describe the structure of a database.
- Data model provides the necessary means to achieve this **abstraction**
- **Structure of a database -** includes
  - data types
  - relationships
  - constraints that apply to the data
- Most data models also include a set of **basic operations** for specifying retrievals and updates on the database.

Data Models -Three Main Categories

- **Conceptual (High-level)**
  - Entity-Relationship (ER) Model

- **Representational (Implementation-level)**
  - Relational Model (most common)
  - Network Model (legacy)
  - Hierarchical Model (legacy)

- **Physical (Low-level)**
  - Linear
  - Hierarchical (Tree-based)
  - Indexes, File Storage

Schemas, Instances, and Database State

- **Schema:** Description/structure of the database
  - Defined during **database design**
  - Rarely changes; used by the **DBMS**
  - Describes:
    - Record types,Attributes,Constraints
  - Also called **Intension**
  - Stored in **DBMS catalog** (as metadata)
- **Instance:** Actual data stored in the database at a given moment
- **Database State:** The content of the database at a specific time (snapshot)

Schema diagram for the database

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

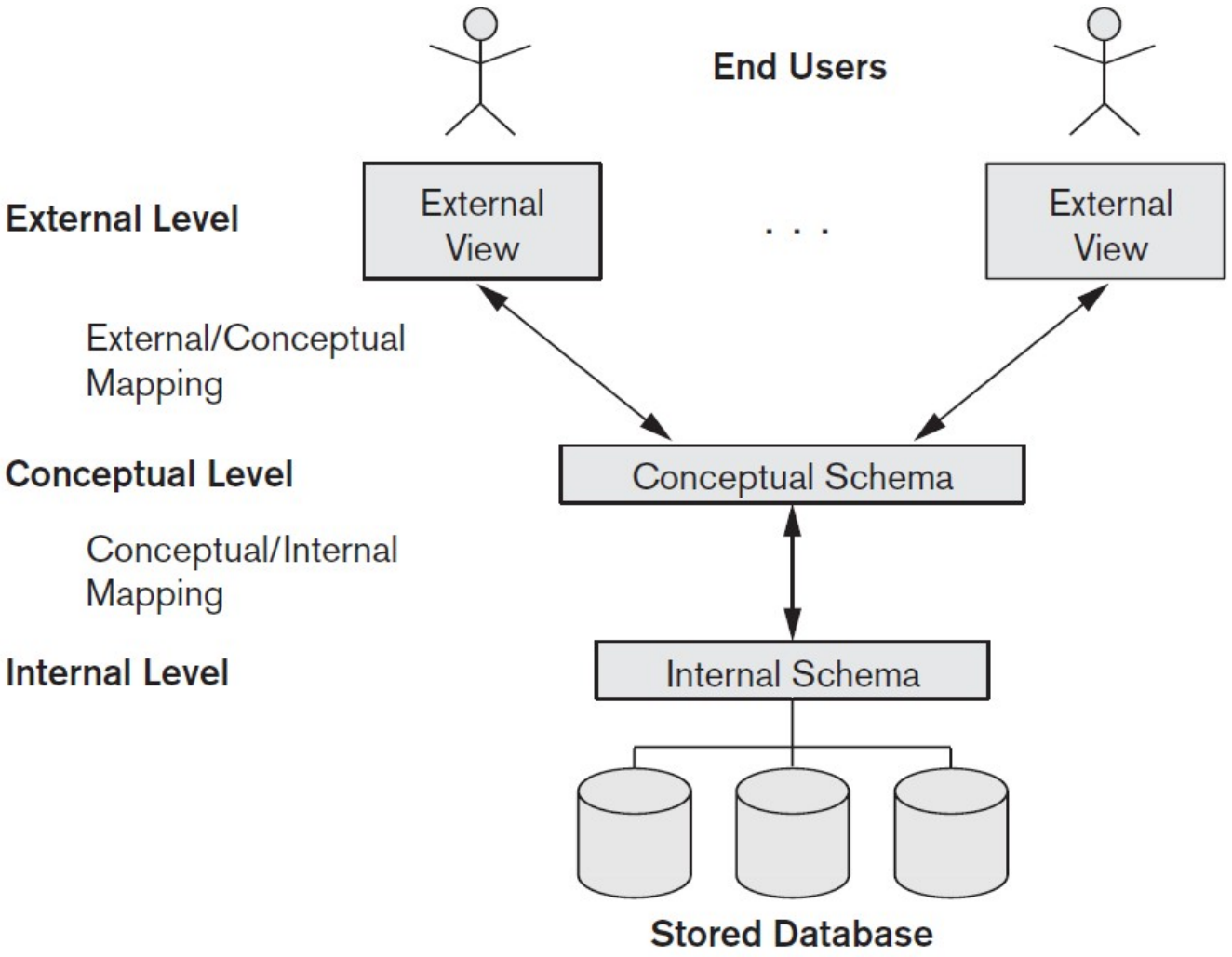
SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

The Three-Schema Architecture



Goal for the Three-Schema Architecture

- Self-describing database using **catalog**
- **Program-data independence**
- **Support for multiple user views**
- Visualize and manage **data abstraction**

Separates user applications from physical storage

Composed of:

1) Internal Level - Describes **physical storage structure**

- Uses **physical data model and**
- Specifies:
  - File structures
  - Indexing/access paths
  - Record layouts
- Known as the internal schema

Three-Schema Architecture

Conceptual Level

Describes **entire logical structure** of the database

Uses **representational data model**

Independent of physical storage

Captures:

- Entities, relationships
- Data types, operations
- Integrity constraints

Known as the **conceptual schema**

External Level (User Views)

Defines **user-specific views** of data

Multiple external schemas possible

Hides details irrelevant to the user

Each view tailored for a **specific user group**

Known as **external schema**

Three-Schema Architecture

Data Independence : nature whereby one can change the structure of the database without having to change its implementation or data.

Physical Data Independence:

Change in internal schema does **not** affect conceptual schema

Examples of changes under Physical Data Independence

- It is by the use of new storage devices like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method.
- Modifying indexes.
- To change the compression techniques or hashing algorithms.
- To change the Location of the Database from say C Drive to D Drive.

Three-Schema Architecture- Data Independence

Logical Data Independence:

Change in conceptual schema does **not** affect external schemas

Examples of changes under Logical Data Independence

·To Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs.

·Merging two records into one.

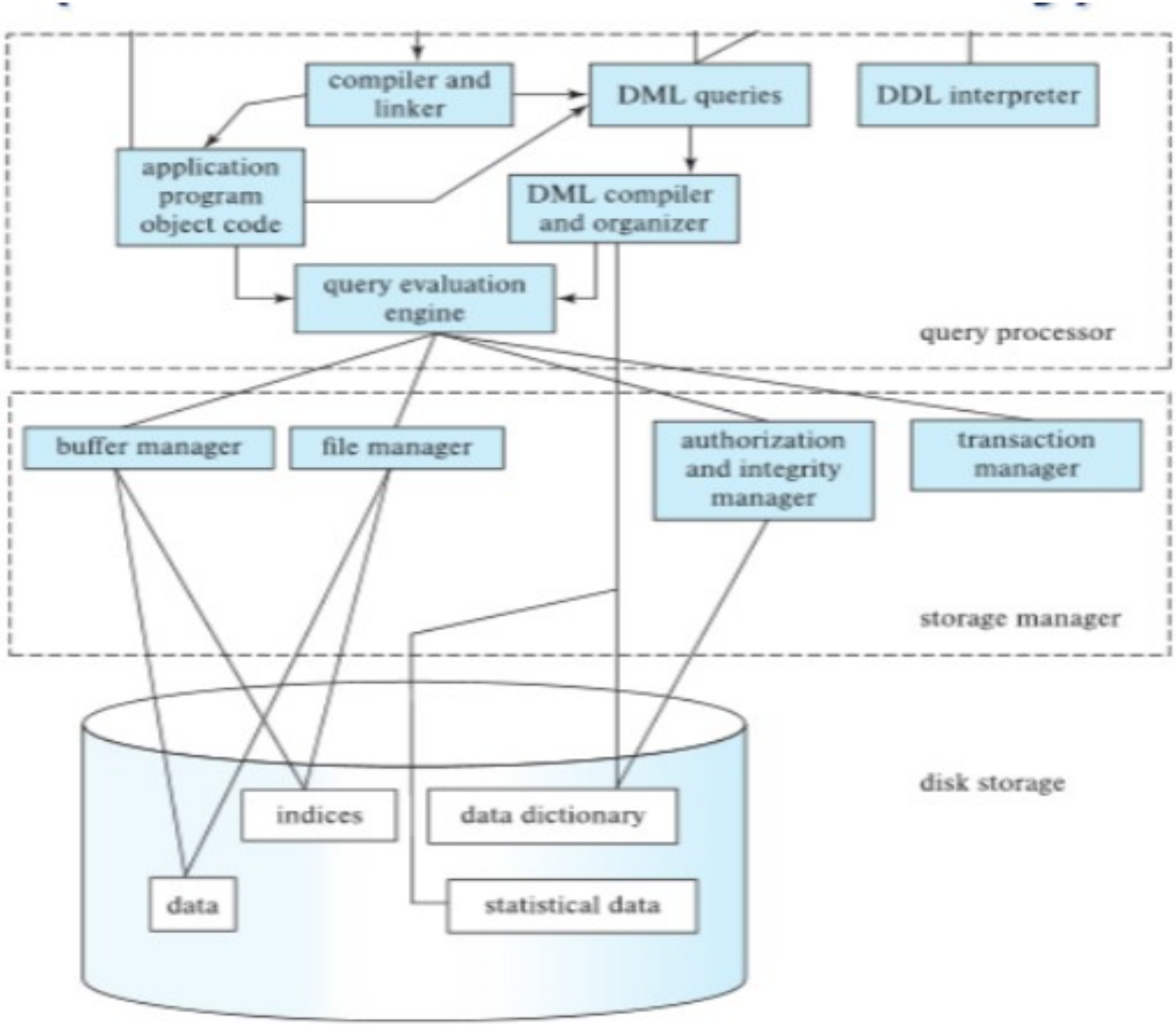
·To break an existing record i.e to divide the record into two or more records.

Logical Data Independence Harder to Achieve Compared to Physical Data Independence?

·The application programs are heavily dependent on logical format of the data they access, hence a change in conceptual level might require the change of the entire program application .

·But ,change in the location of database or modifying file organization or use of new storage device etc. Will not require the change at the higher logical levels.

Overall Architecture of Database Management Systems



Overall Architecture of Database Management Systems

Database Architecture refers to the internal components of the DBMS, including the Query Processor, Storage Manager, and Disk Storage. It also defines the interaction of these components.

Query Processor:

- Interprets and executes user queries from application programs.
- Translates high-level queries into low-level instructions for execution.

Components of Query Processor:

- **DML Compiler**:
  Converts Data Manipulation Language (DML) statements into low-level machine instructions.
- **DDL Interpreter**:
  Processes Data Definition Language (DDL) statements and generates metadata stored in system catalog tables.
- **Embedded DML Pre-compiler**:
  Translates DML statements embedded in application programs into standard procedural calls.
- **Query Optimizer**:
  Improves query efficiency by selecting the optimal query execution plan. Considers factors like indexes, join order, and system resources to minimize execution time.

Overall Architecture of Database Management Systems

Storage Manager (Database Control System)

- Acts as an interface between the physical database and user queries.
- Manages **data storage, retrieval, updating, and deletion**.
- Ensures **data integrity, consistency, and security** by enforcing constraints and executing DCL (Data Control Language) statements.

Components of Storage Manager:

- **Authorization Manager**:
  Ensures **role-based access control**; verifies user permissions before allowing any operation.
- **Integrity Manager**:
  Validates **integrity constraints** during database modifications to maintain data correctness.
- **Transaction Manager**:
  Controls **concurrent transactions** to ensure that the database stays in a **consistent state** before and after execution.
- **File Manager**:
  Handles **file organization and storage**, including managing the structure and space of physical database files.
- **Buffer Manager**:
  Manages **cache memory**, coordinating the movement of data between **main memory** and **secondary storage** for efficient access.

Overall Architecture of Database Management Systems

Role of Database Administrator (DBA)

- responsible for the **overall management and maintenance** of the database system.
- Ensures **data availability, security, integrity, and performance**.
- **Database Design & Architecture**:
  - Plans and structures the **logical and physical design** of the database.
  - Ensures scalability, normalization, and proper data modeling.
- **Security Management**:
  - Implements **Role-Based Access Control (RBAC)**.
  - Applies **encryption** to protect data.
  - Enforces **strong authentication mechanisms**, including **Multi-Factor Authentication (MFA)**.
- **Backup & Recovery**:
  - Schedules and manages **regular backups**.
  - Develops and tests **disaster recovery plans** to ensure data restoration in case of failure or loss.
- **Performance Tuning**:
  - Monitors and optimizes **query execution**, **indexing**, and **resource allocation**.
  - Ensures the DBMS performs **efficiently under varying workloads**.

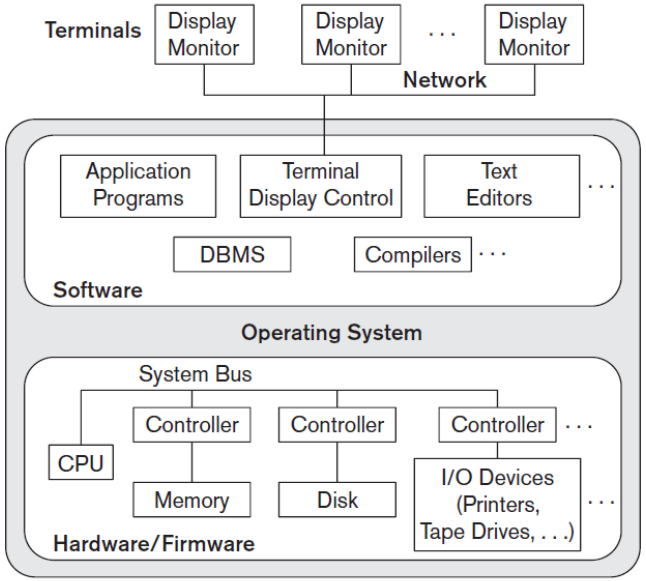
Centralized DBMS Architecture

All database processing and application logic occur on a **single central computer** (usually a mainframe or powerful server).

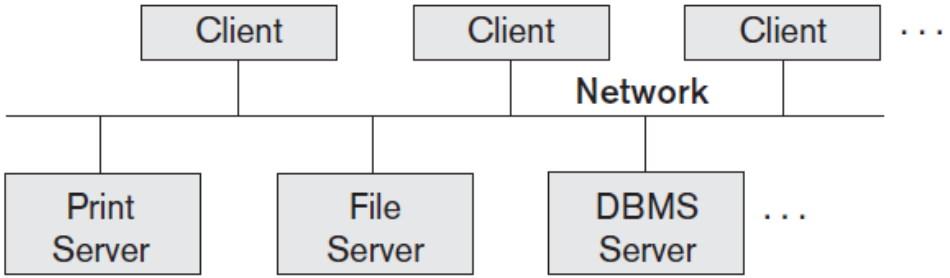
- **User Access**:

Users accessed the system via **dumb terminals** with no processing power
only capable of displaying output and sending input.

- **Characteristics**:
  - Central system handled:
    - ❖ User interface
    - ❖ Application program execution
    - ❖ All DBMS functionality
  - **No local processing** at the user's side
  - Simplified control but limited scalability.



Client/Server Architecture



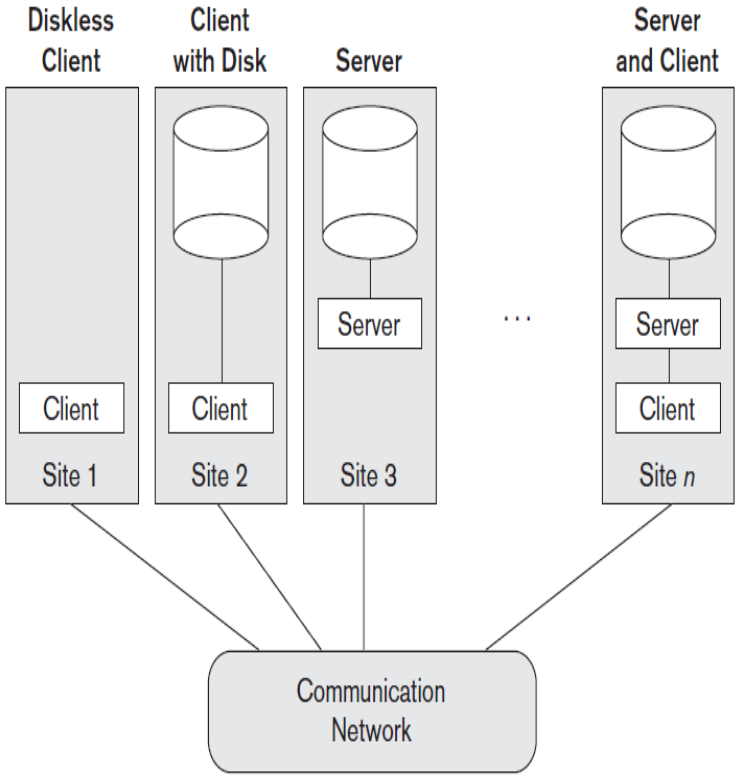
- A **distributed computing environment** where functionality is split between **clients (user machines)** and **servers (resource providers)**.
- **Client Role**:
  - Provides **user interface** and **local processing**.
  - Sends requests to the server for services it cannot perform locally (e.g., database access).
- **Server Role**:
  - Provides **specific services** like database access, file storage, printing, etc.
  - Can be a **dedicated machine** or a system with both server and client capabilities.
- **Network Dependency**:
  Relies on **LANs and other networks** to connect multiple client and server machines.

Two-Tier Client/Server Architecture

- **Tier 1 (Client)**:
  User interface and local application logic.
- **Tier 2 (Server)**:
  Centralized **DBMS server** for handling database queries and data storage.

Features:

- Client machines access **specialized servers** (e.g., file servers, printer servers, database servers).
- Improved resource sharing and **decentralized processing** compared to centralized architecture.
- Suitable for small to medium-scale applications.



Two-Tier Client/Server Architecture

Client Side:

Runs **user interface** and **application programs**.

Sends SQL queries to the server.

Server Side:

Called **SQL Server**, **Query Server**, or **Transaction Server**.

Handles SQL processing and database transactions.

Communication:

Uses **ODBC** (for various languages) or **JDBC** (for Java) to connect and interact with the DBMS.

Object-Oriented DBMS Variation:

DBMS functions are split between client and server more deeply.

Server: handles storage, local concurrency, recovery.

Client: handles complex object structuring, global control functions.

Advantages:

Simple and easy to implement.

Works well with **existing systems**.

Limitation:

Less scalable; led to **Three-Tier Architecture** with the rise of web applications.

Two-Tier Client/Server Architecture

Structure:

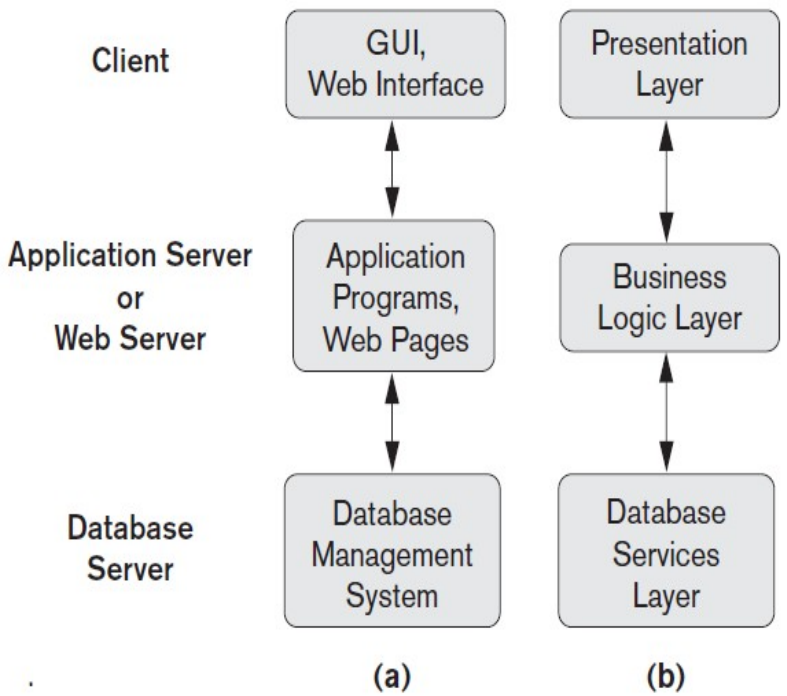
- **Client (Presentation Tier)** : User interfac (GUI)
- **Application Server (Middle Tier)** : Business logic and rules
- **Database Server (Data Tier)** : Data storage and query processing

Functionality:

- **Middle tier** handles client requests, applies business rules, and interacts with the database.
- Improves **security** by validating client credentials before accessing the database.
- Passes processed or raw data back to the client for presentation.

Advantages:

- **Modularity**: UI, business logic, and data access are separated.
- **Security**: Middle tier acts as a gatekeeper.
- **Scalability**: More users can be supported efficiently.



a) Web applications with an intermediate layer between the client and the database server
b)architecture used by database and other application package vendors.