Principles of Database Design

Santhosh Kumar K P
Asst. Professor,
Dept. of Computer Science
SH College Thevara

Introduction to Database Management System

Data: structured, semi-structured and unstructured data

- **Data**: Known facts that can recorded and that has implicit meaning.
- **Structured data**: Information stored in databases is known as structured data because it is represented in a **strict** format.
- The DBMS then checks to ensure that all data follows the structures and constraints specified in the schema.

Data: structured, semi-structured and **unstructured data**

- **Unstructured** data, because there is very limited indication of the type of data.
- Example: File (text, excel, pdf etc)

Data: structured, **semi-structured** and unstructured data

- This data may have a certain structure, but **not** all the information collected will have **identical structure**. This type of data is known as **semi-structured data**.
- In semi-structured data, the **schema information is mixed in with the data values**, since each data object can have **different attributes** that are not known in advance.
- Example: Email (email id, subject, body, file(unstructured))

Data base

- **Database**: is a collection of related data.
- Traditional database: text, numbers
- Multimedia database: videos, mp3, movies
- GIS(geographical information system): satellite images
- Real Time db: related to time, eg: used in supermarket
- Data Warehouse: huge and historical

Database Management System (DBMS)

- DBMS is a general purpose software system that facilitates the process 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
- **Constructing** the database is the process of *storing the data on some storage medium* that is controlled by the DBMS.
- **Manipulating is** querying the database *to retrieve* specific data, *updating* the database, generating *reports* from the data
- **Sharing** a database allows multiple users and programs to *access the database simultaneously*.

Database Applications:

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Drawbacks of using file systems to store data:

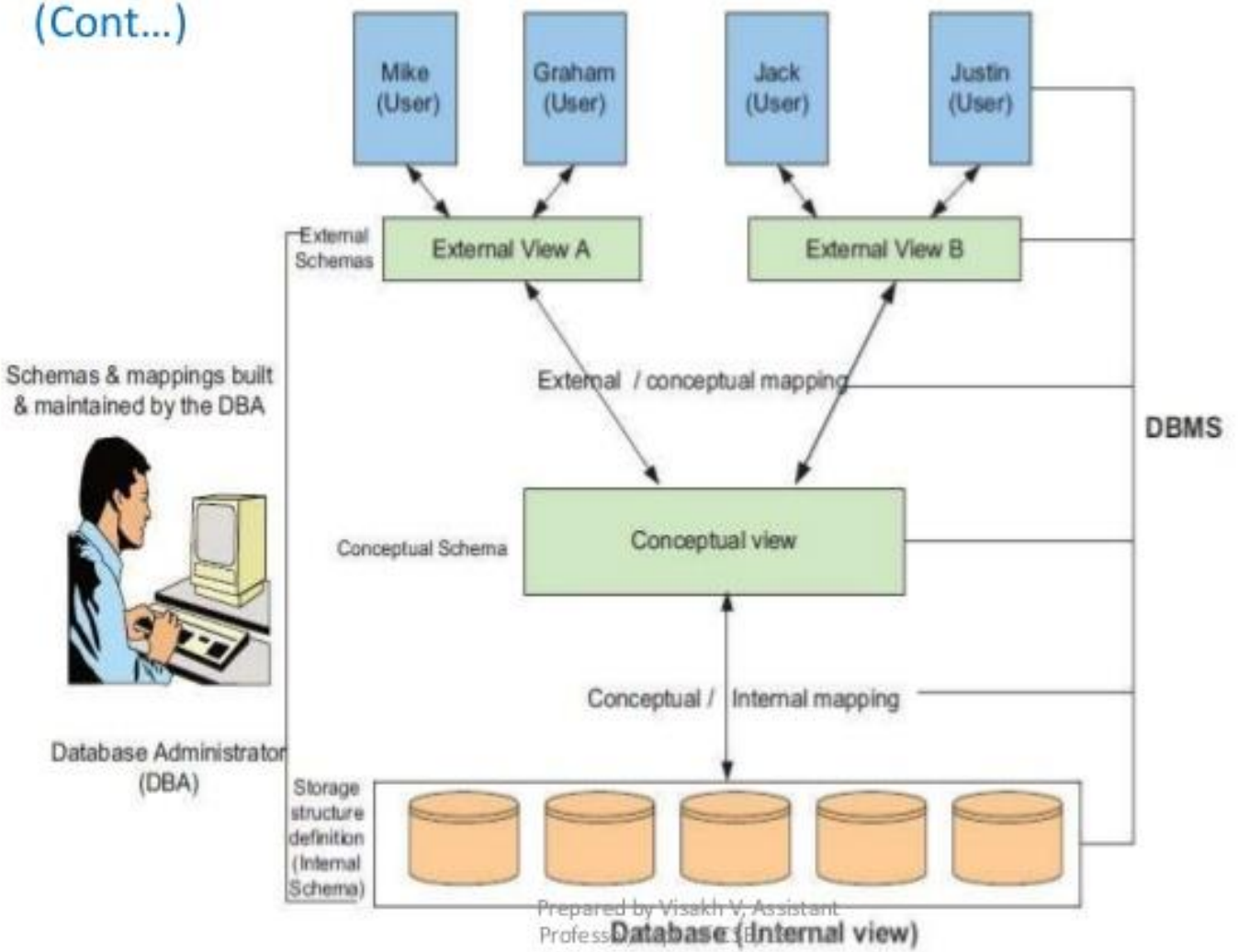
- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
  - Integrity constraints (e.g. account balance > 0) become part of program code
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data:

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - E.g. transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent accessed needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - E.g. two people reading a balance and updating it at the same time
- Security problems

Detailed Three Schema Architecture

(Cont…)

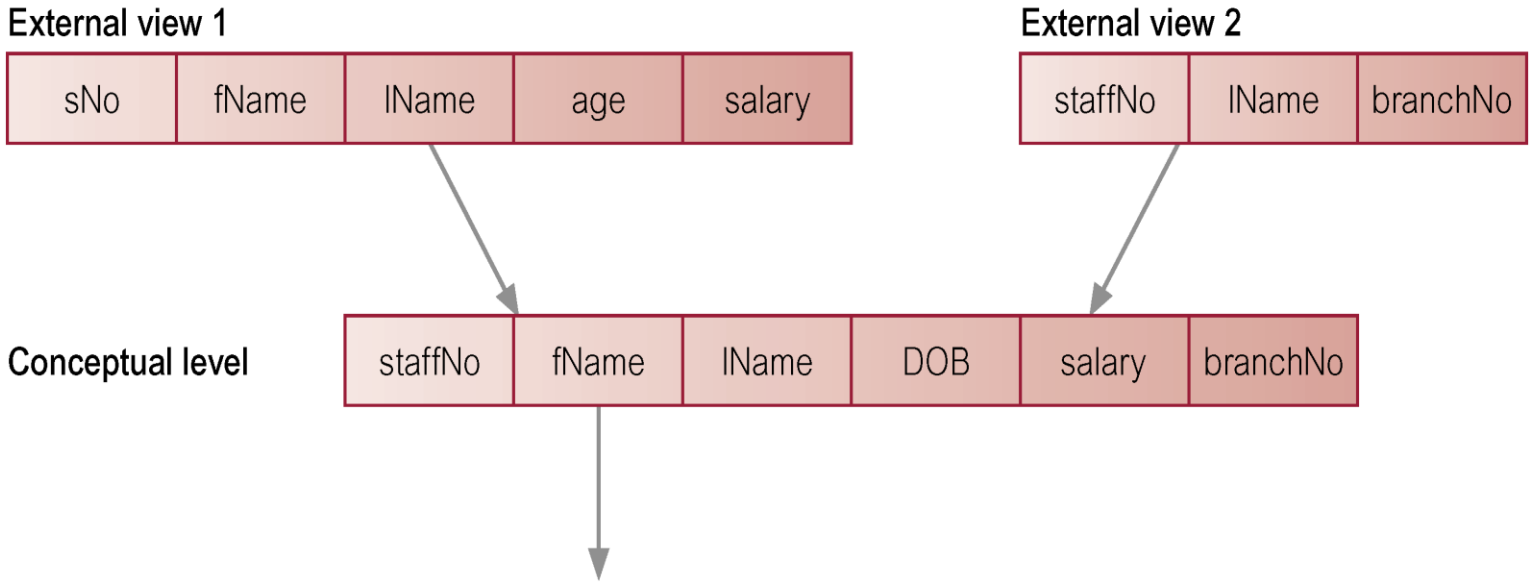


Three-Schema Architecture

- **External schemas/High level** at the external level to describe the various **user views.**
- **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
- **Internal schema/low level** at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.

Three-Schema Architecture

- **Mappings** among schema levels are needed to transform requests and data.
- Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.



Internal level

```
struct STAFF {
      int staffNo;
      int branchNo;
      char fName [15];
      char lName [15];
      struct date dateOf Birth;
      float salary;
      struct STAFF *next;                  /* pointer to next Staff record */
};
index staffNo; index branchNo;             /* define indexes for staff */
```

Three-Schema Architecture

- **Data Independence: T**he capacity to change the schema at one level of the database system without changing the schema at next higher level.

Two types

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.

Actors on the Scene

- **Database Administrators:** Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.
- Duties:
  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Monitoring performance and responding to changes in requirements

Actors on the Scene

- **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.
- **End Users:** are the people whose jobs require access to the database for querying, updating, and generating reports: Database is already developed for their use.

Actors on the Scene: Software Engineers

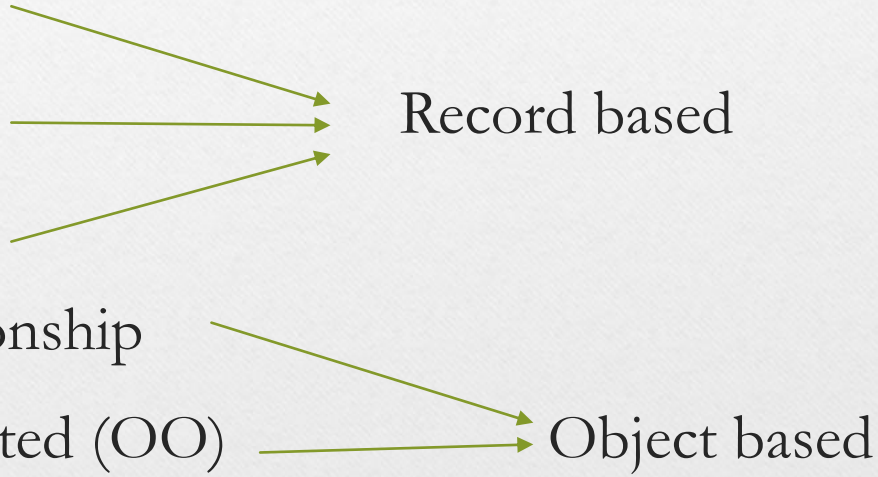
- **System analysts** determine the requirements of end users, especially naive and parametric end users, and develop **specifications**
- **Application programmers** implement these specifications as **programs;** then they test, debug, document, and maintain these canned transactions.

Data models

- Relatively simple representations, usually graphical, of complex real-world data structures
- Facilitate interaction among the designer, the applications programmer, and the end user
- End-users have different views and needs for data
- Data model organizes data for various users

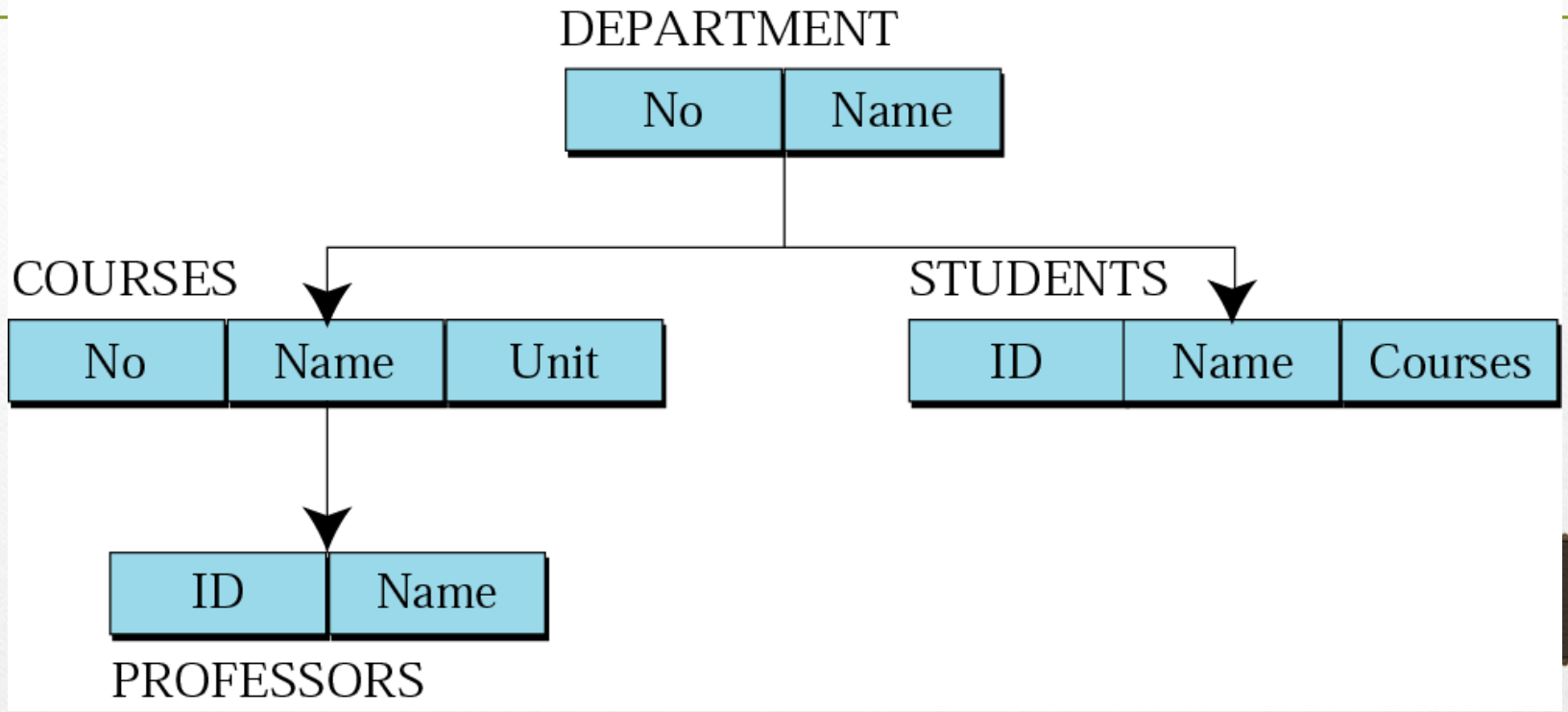
The Evolution of Data Models

- Hierarchical
- Network
- Relational
- Entity relationship
- Object oriented (OO)



Hierarchical Data Model

- The hierarchical structure contains levels, or segments.
- Basic logical structure is represented by an upside-down "tree"
- Depicts a set of one-to-many (1:M) relationships between a parent and its children segments
  - Each parent can have many children
  - each child has only one parent



- To retrieve data: Starts from Top and traverse different levels to bottom.

Hierarchical Data Model

Advantage

- Promote data sharing
- Simple: parent child development
- Efficient in 1:M relationships

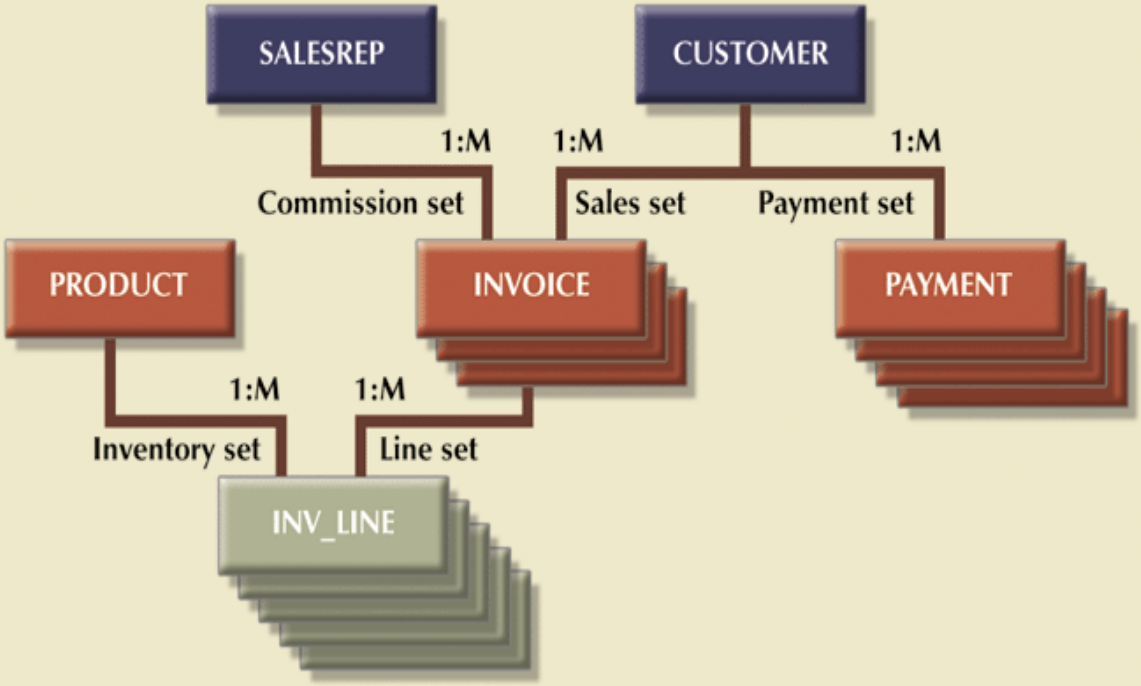
Disadvantage

- Requires knowledge about hierarchical paths
- No multi-parent relationship
- Lack of standards

Network Data Model

- Resembles hierarchical model
- Data is represented as Collection of records in 1:M relationships
- Set
  - Relationship
  - Composed of at least two record types
    - Owner
      - Equivalent to the hierarchical model's parent
    - Member
      - Equivalent to the hierarchical model's child

FIGURE 2.2 A network data model



Network Data Model

Advantage

- Support multi-parent
- Flexible than hierarchical system

Disadvantage

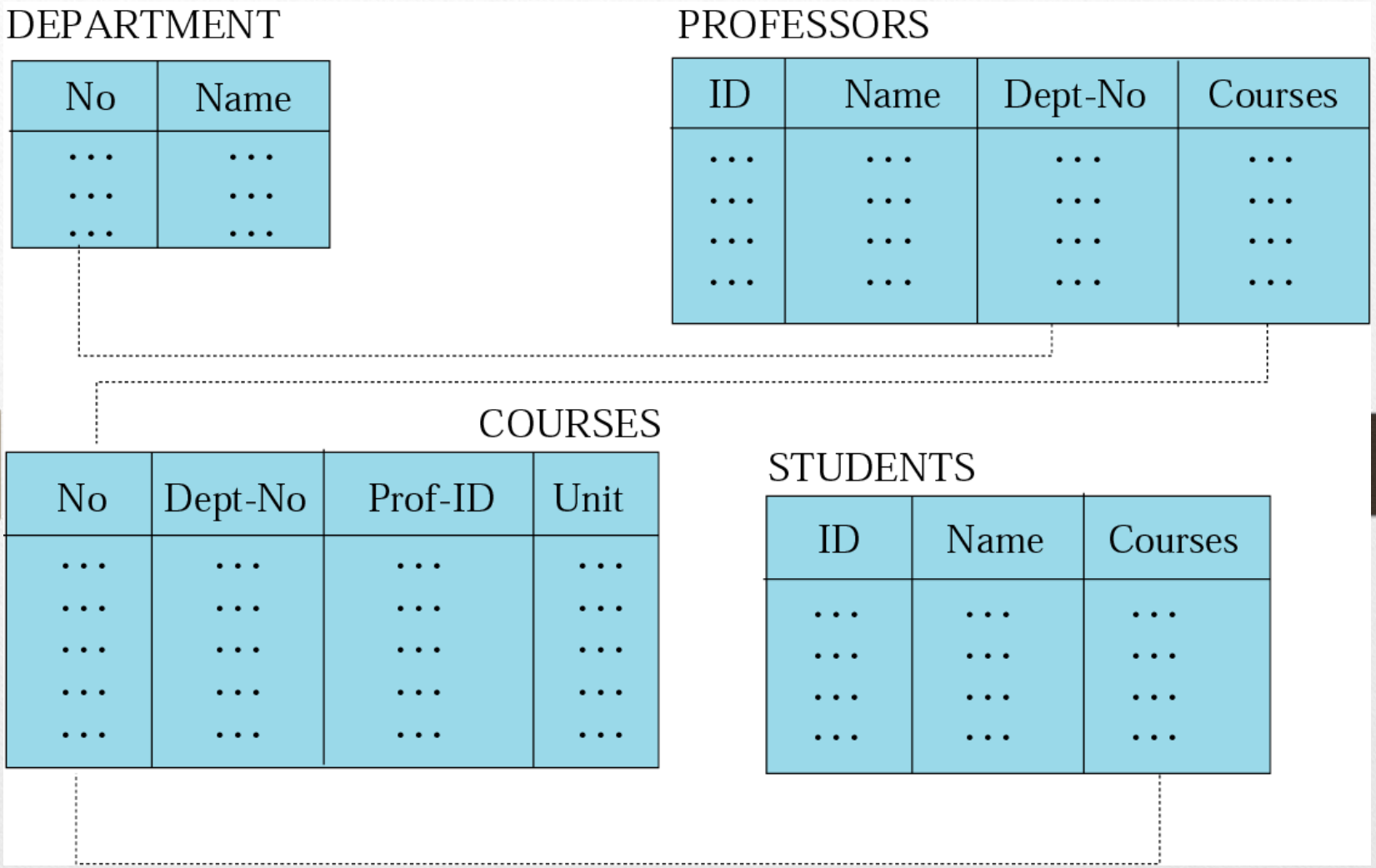
- Complex in implementation development, management
- Complex limits efficiency

Relational Data Model

- Table (relations)
  - Matrix consisting of a series of row/column intersections
  - Related to each other through sharing a common entity characteristic
- Relational diagram
  - Representation of relational database's entities, attributes within those entities, and relationships between those entities

Relational Data Model

- Relational Table
  - Stores a collection of related entities
- Relational table is purely logical structure
  - How data are physically stored in the database is of no concern to the user or the designer
  - This property became the source of a real database revolution


DEPARTMENT
No
Name
PROFESSORS
ID
Name
Dept-No
Courses
COURSES
No
Dept-No
Prof-ID
Unit
STUDENTS
ID
Name
Courses

Entity Relationship Model

- Entity relationship diagram (ERD)
  - Uses graphic representations to model database components
  - Entity is mapped to a relational table
- Entity instance (or occurrence) is row in table
- Entity set is collection of like entities
- Connectivity labels types of relationships
  - Diamond connected to related entities through a relationship line

FIGURE 2.5 The basic Chen ERD

A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.

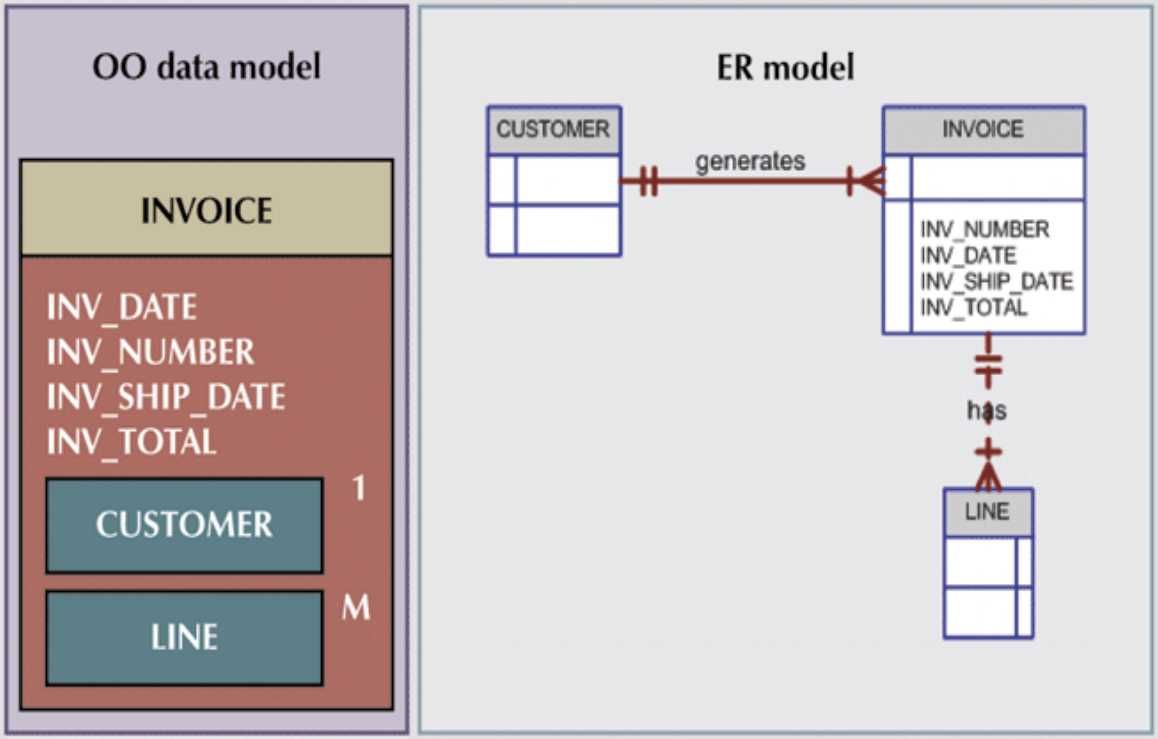
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.

A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.

Object Oriented Model

- Object is an abstraction of a real-world entity
- Attributes describe the properties of an object
- Includes information about relationships between facts within object, and relationships with other objects
- Subsequent OODM development allowed an object to also contain all operations
- Object becomes basic building block for autonomous structures

FIGURE 2.7 A comparison of the OO model and the ER model



Database Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)

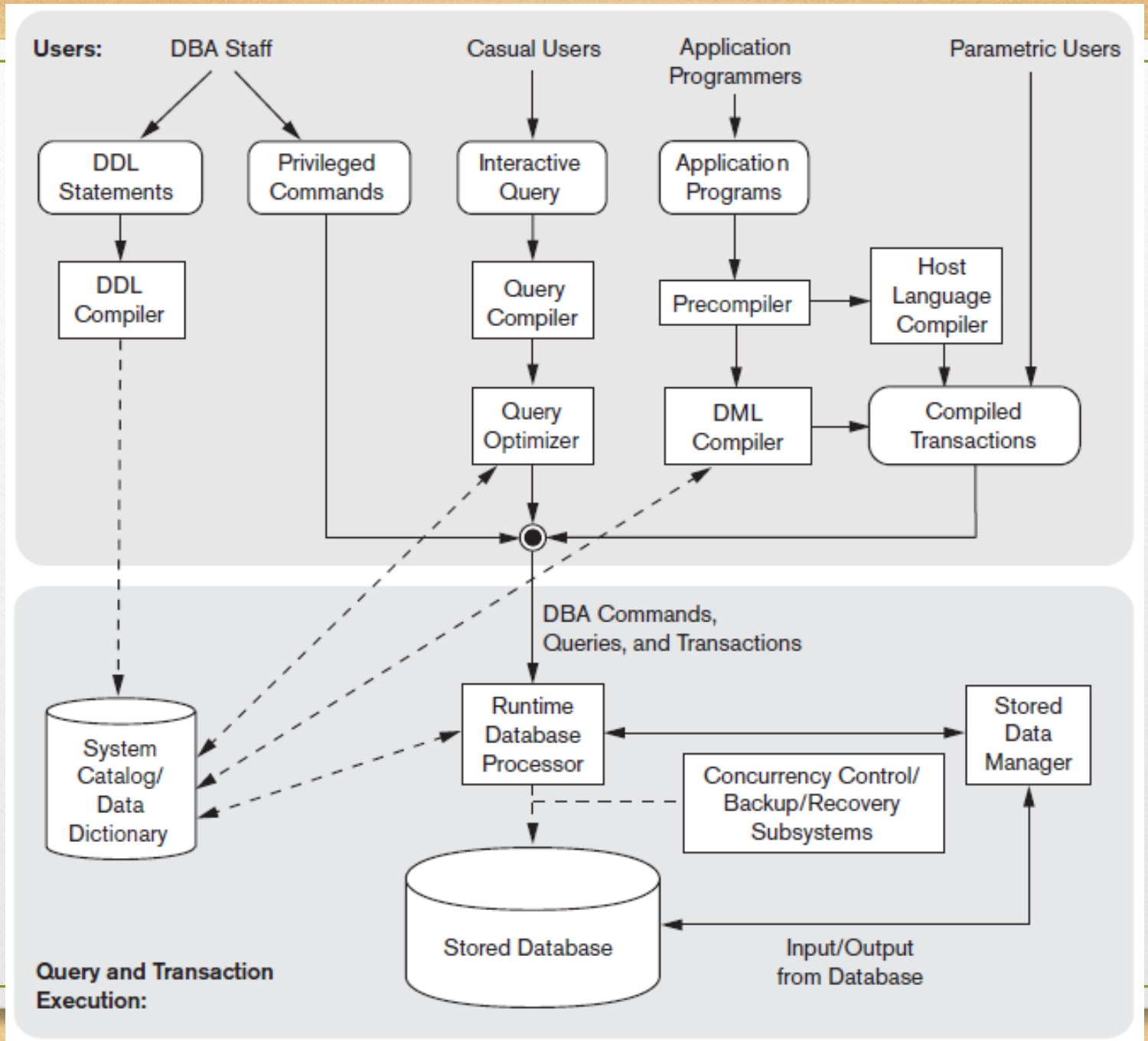
Data Definition Language (DDL)

- A language that allows DBA or user to describe and name the entities, attributes and relationships required for an application, together with any associated integrity and security constraints

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language

Database System Environment/
Component Modules


Users:
DBA Staff
Casual Users
Application Programmers
Parametric Users
DDL Statements
Privileged Commands
Interactive Query
Application Programs
DDL Compiler
Query Compiler
Precompiler
Host Language Compiler
Query Optimizer
DML Compiler
Compiled Transactions
DBA Commands, Queries, and Transactions
System Catalog/ Data Dictionary
Runtime Database Processor
Stored Data Manager
Concurrency Control/ Backup/Recovery Subsystems
Stored Database
Input/Output from Database
Query and Transaction Execution:

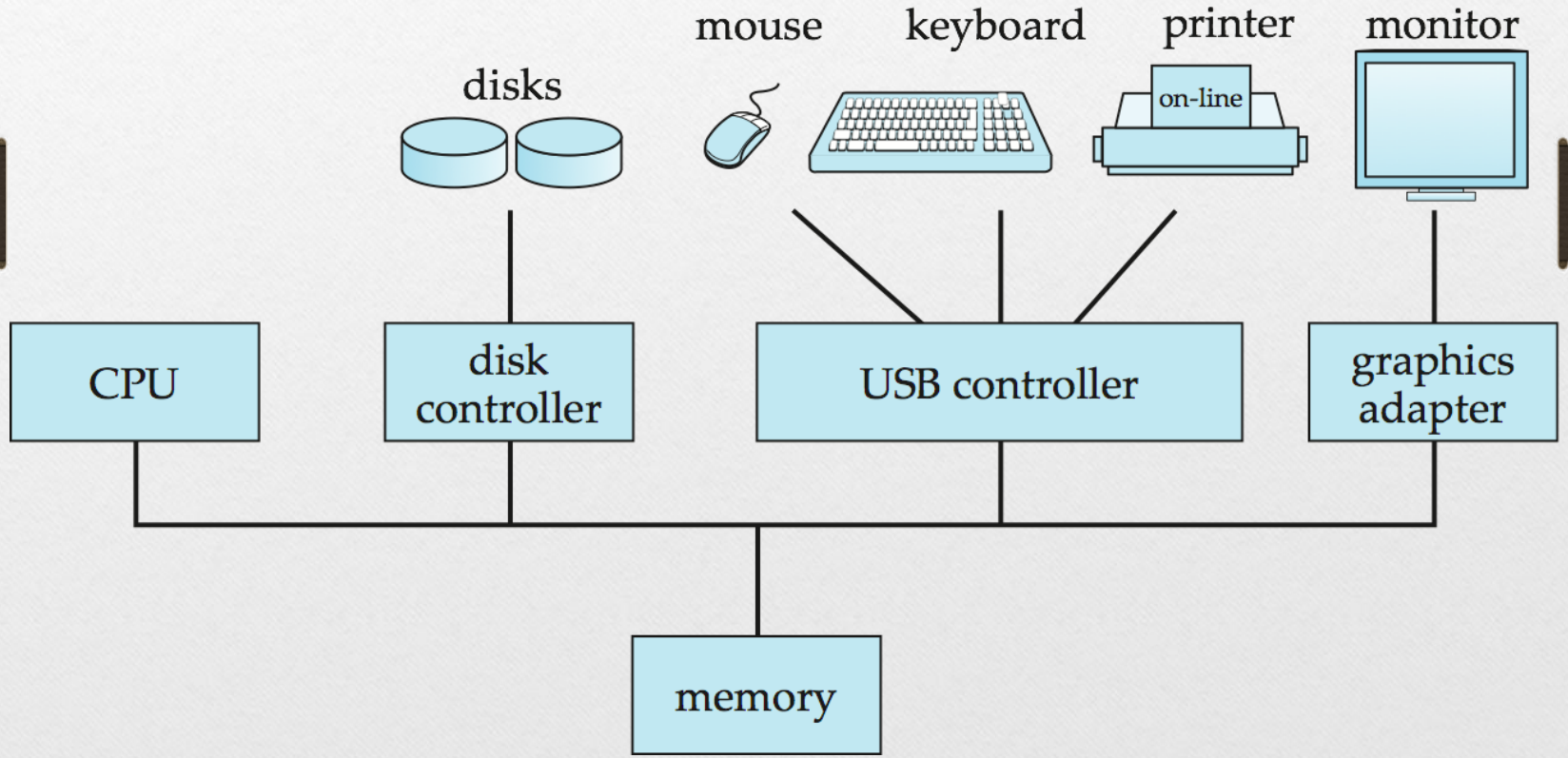
Database System Architectures

- **Centralized and Client-Server Systems**
- **Server System Architectures**
- Parallel Systems
- Distributed Systems
- Network Types

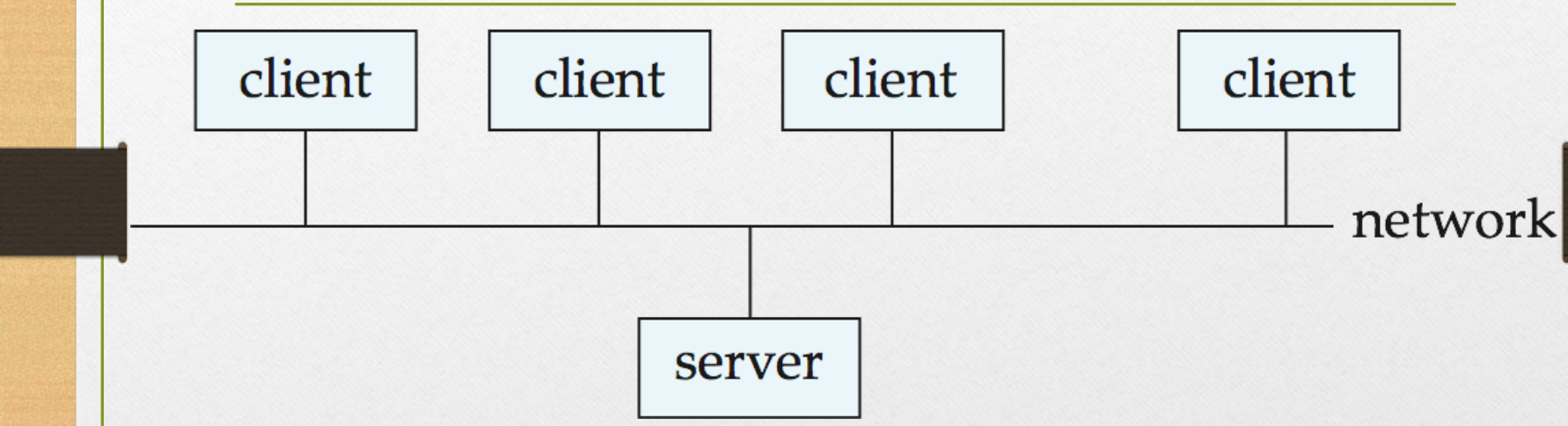
Centralized Systems

- Run on a single computer system and do not interact with other computer systems.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provides access to shared memory.
- **Single-user system** (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.
- **Multi-user system**: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system vie terminals. Often called *server* systems.

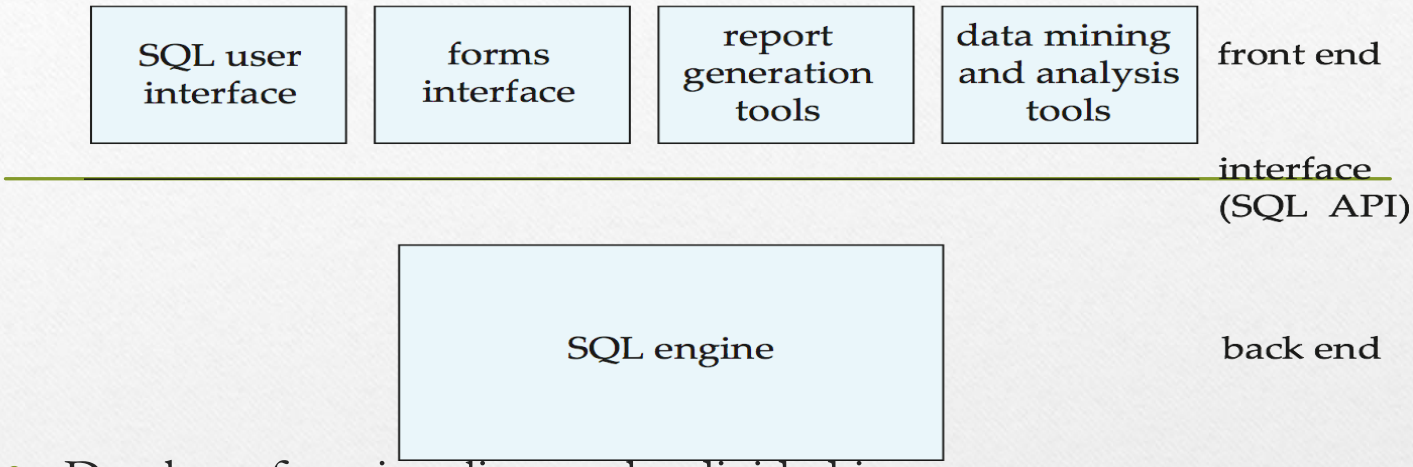
Centralized Systems



Client-Server Systems



- Server systems satisfy requests generated at *m* client systems, whose general structure is shown

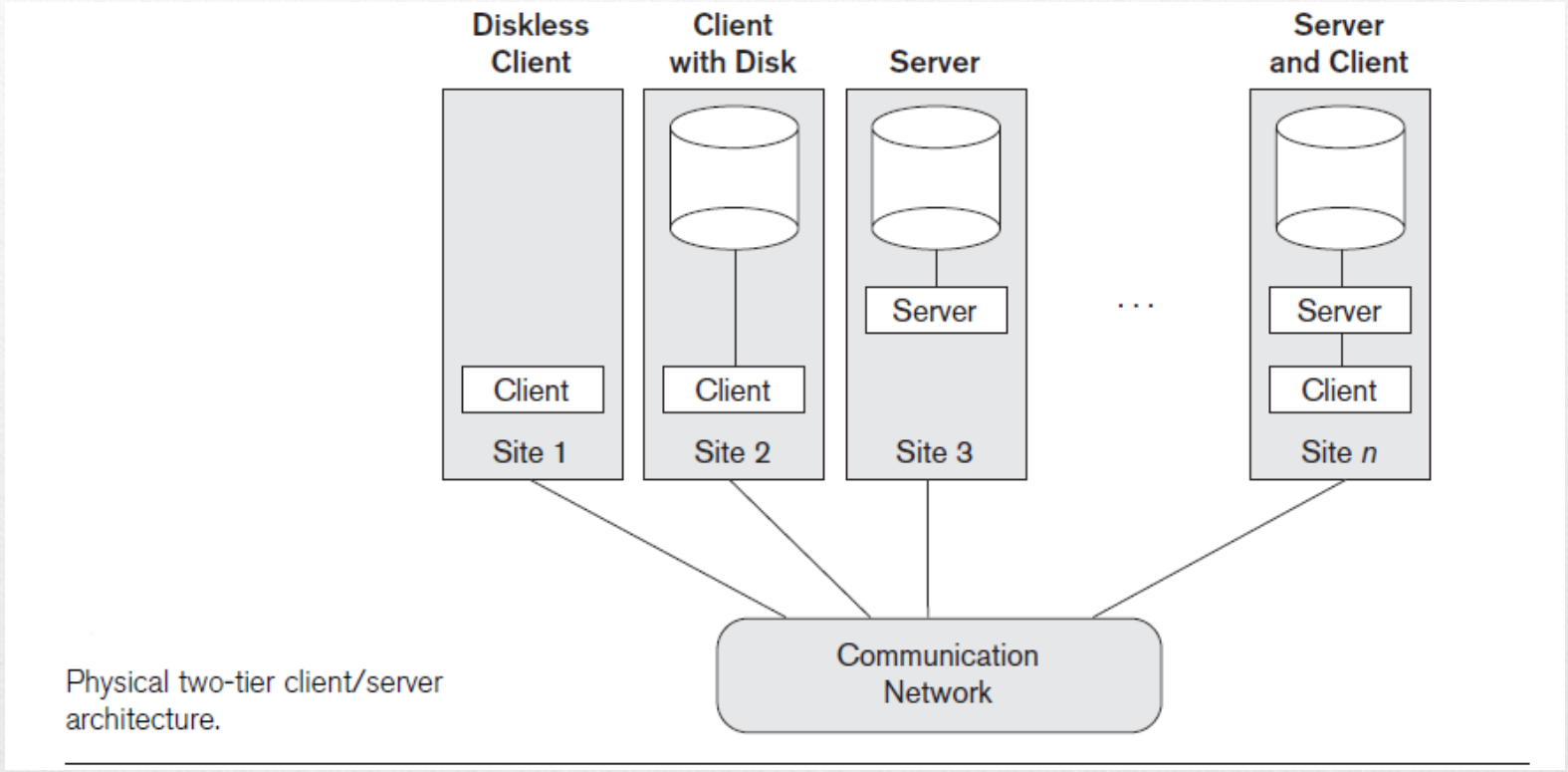


- Database functionality can be divided into:
  - **Back-end**: manages access structures, query evaluation and optimization, concurrency control and recovery.
  - **Front-end**: consists of tools such as *forms*, *report-writers*, and graphical user interface facilities.
- The interface between the front-end and the back-end is through SQL or through an application program interface.

Advantages

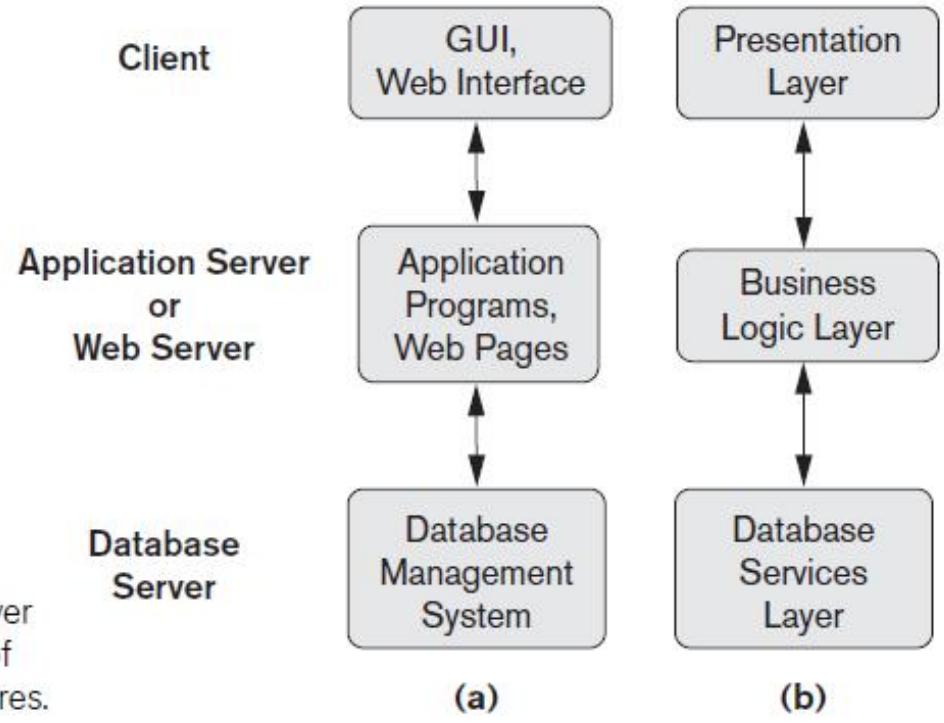
- Advantages of replacing mainframes with networks of workstations or personal computers connected to back-end server machines:
  - better functionality for the cost
  - flexibility in locating resources and expanding facilities
  - better user interfaces
  - easier maintenance

Two-tier client/server architecture



Physical two-tier client/server architecture.

Three-tier Client/server Architecture



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

Reference

- Fundamentals of database design: Ramiez Elsmari and Shamakant Navathe