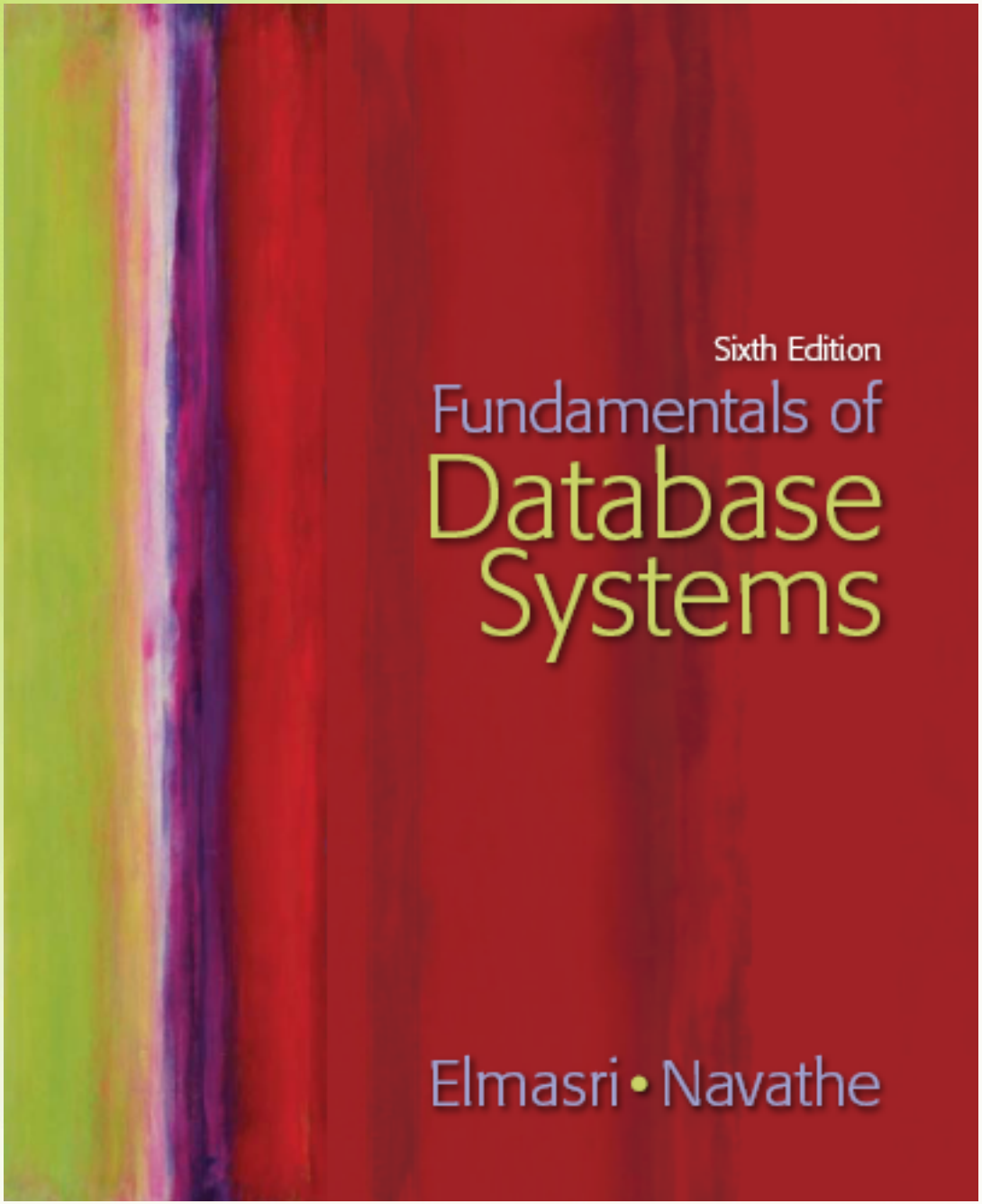
# Chapter 2

# Database System Concepts and Architecture




Addison-Wesley
is an imprint of
PEARSON

# Chapter 2 Outline

- Data Models, Schemas, and Instances
- Three-Schema Architecture and Data Independence
- Database Languages and Interfaces
- The Database System Environment
- Centralized and Client/Server Architectures for DBMSs
- Classification of Database Management Systems

# Database System Concepts and Architecture

- Basic client/server DBMS architecture
  - **Client module**
  - **Server module**

# Data Models, Schemas, and Instances

- **Data abstraction**
  - Suppression of details of data organization and storage
  - Highlighting of the essential features for an improved understanding of data

# Data Models, Schemas, and Instances (cont'd.)

- **Data model**
  - Collection of concepts that describe the structure of a database
  - Provides means to achieve data abstraction
  - **Basic operations**
    - Specify retrievals and updates on the database
  - **Dynamic aspect** or **behavior** of a database application
    - Allows the database designer to specify a set of valid operations allowed on database objects

# Categories of Data Models

- **High-level** or **conceptual data models**
  - Close to the way many users perceive data
- **Low-level** or **physical data models**
  - Describe the details of how data is stored on computer storage media
- **Representational data models**
  - Easily understood by end users
  - Also similar to how data organized in computer storage

# Categories of Data Models (cont'd.)

- **Entity**
  - Represents a real-world object or concept
- **Attribute**
  - Represents some property of interest
  - Further describes an entity
- **Relationship** among two or more entities
  - Represents an association among the entities
  - **Entity-Relationship model**

# Categories of Data Models (cont'd.)

- **Relational data model**
  - Used most frequently in traditional commercial DBMSs
- **Object data model**
  - New family of higher-level implementation data models
  - Closer to conceptual data models

# Categories of Data Models (cont'd.)

- **Physical data models**
  - Describe how data is stored as files in the computer
  - **Access path**
    - Structure that makes the search for particular database records efficient
  - **Index**
    - Example of an access path
    - Allows direct access to data using an index term or a keyword

# Schemas, Instances, and Database State

- **Database schema**
  - Description of a database
- **Schema diagram**
  - Displays selected aspects of schema
- **Schema construct**
  - Each object in the schema
- **Database state** or **snapshot**
  - Data in database at a particular moment in time

# Schemas, Instances, and Database State (cont'd.)

**Figure 2.1**
Schema diagram for the database in Figure 1.2.

**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 |
|---|---|---|

[6]Schema changes are usually needed as the requirements of the database applications change. Newer database systems include operations for allowing schema changes, although the schema change process is more involved than simple database updates.

[7]It is customary in database parlance to use *schemas* as the plural for *schema*, even though *schemata* is the proper plural form. The word *scheme* is also sometimes used to refer to a schema.

# Schemas, Instances, and Database State (cont'd.)

- **Define** a new database
  - Specify database schema to the DBMS
- **Initial state**
  - **Populated** or **loaded** with the initial data
- **Valid state**
  - Satisfies the structure and constraints specified in the schema

# Schemas, Instances, and Database State (cont'd.)

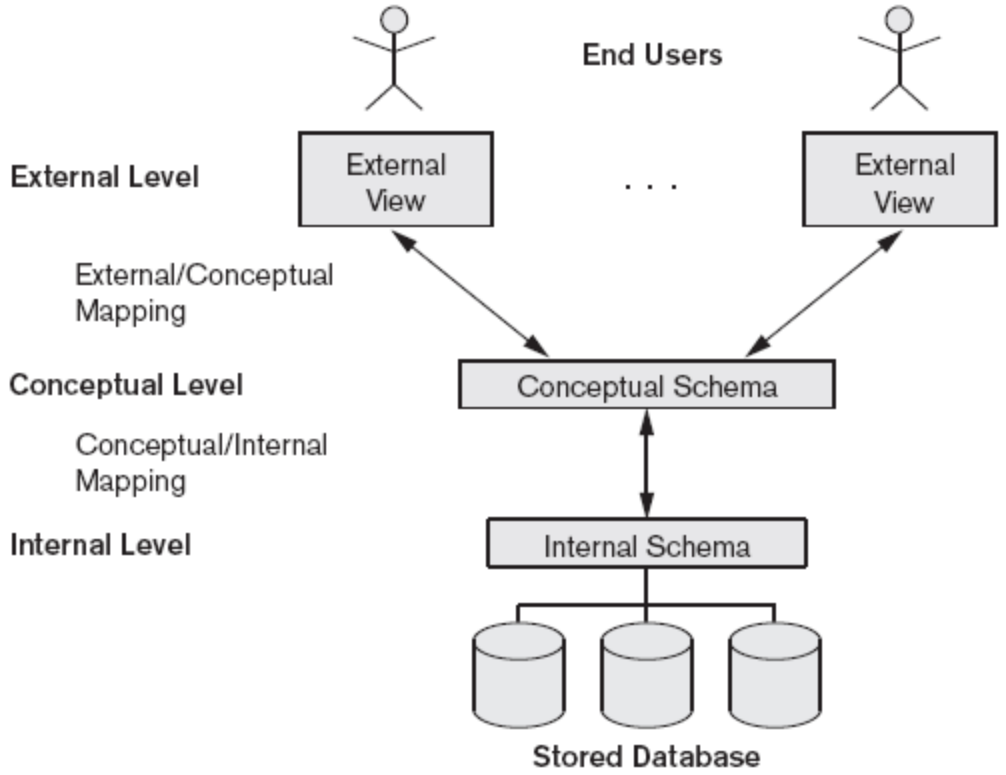
- **Schema evolution**
  - Changes applied to schema as application requirements change

# Three-Schema Architecture and Data Independence

- **Internal level**
  - Describes physical storage structure of the database
- **Conceptual level**
  - Describes structure of the whole database for a community of users
- **External** or **view level**
  - Describes part of the database that a particular user group is interested in

# Three-Schema Architecture and Data Independence (cont'd.)

**Figure 2.2**
The three-schema architecture.



# Data Independence

- Capacity to change the schema at one level of a database system
  - Without having to change the schema at the next higher level
- Types:
  - **Logical**
  - **Physical**

# DBMS Languages

- **Data definition language (DDL)**
  - Defines both schemas
- **Storage definition language (SDL)**
  - Specifies the internal schema
- **View definition language (VDL)**
  - Specifies user views/mappings to conceptual schema
- **Data manipulation language (DML)**
  - Allows retrieval, insertion, deletion, modification

# DBMS Languages (cont'd.)

- **High-level** or **nonprocedural** DML
  - Can be used on its own to specify complex database operations concisely
  - **Set-at-a-time** or **set-oriented**
- **Low-level** or **procedural** DML
  - Must be embedded in a general-purpose programming language
  - **Record-at-a-time**

# DBMS Interfaces

- Menu-based interfaces for Web clients or browsing
- Forms-based interfaces
- Graphical user interfaces
- Natural language interfaces
- Speech input and output
- Interfaces for parametric users
- Interfaces for the DBA

# The Database System Environment

- DBMS component modules
  - **Buffer management**
  - **Stored data manager**
  - **DDL compiler**
  - **Interactive query** interface
    - **Query compiler**
    - **Query optimizer**
  - **Precompiler**

# The Database System Environment (cont'd.)

- DBMS component modules
  - **Runtime database processor**
  - **System catalog**
  - **Concurrency control system**
  - **Backup and recovery system**

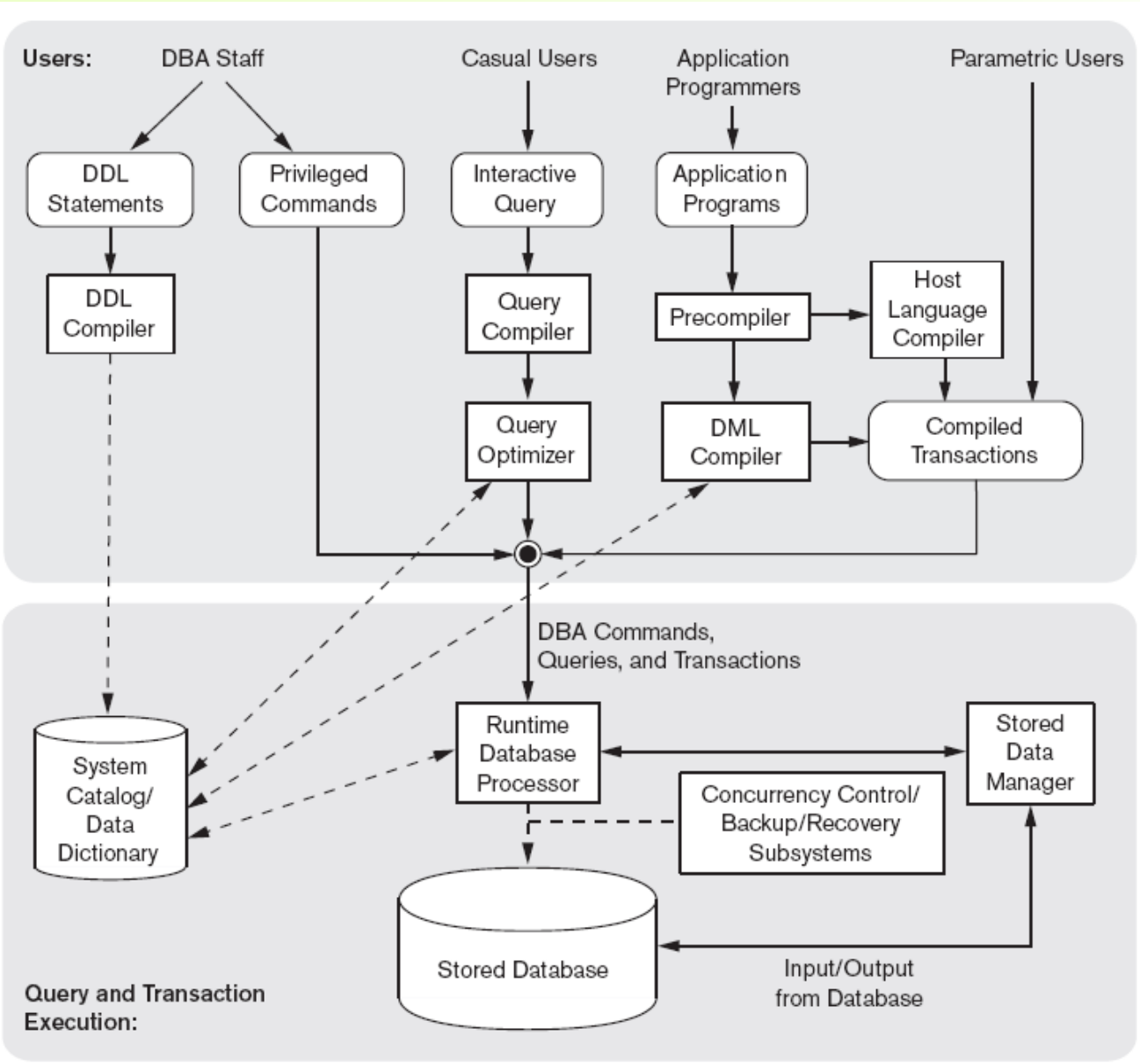



**Figure 2.3**
Component modules of a DBMS and their interactions.

# Database System Utilities

- **Loading**
  - Load existing data files
- **Backup**
  - Creates a backup copy of the database

# Database System Utilities (cont'd.)

- **Database storage reorganization**
  - Reorganize a set of database files into different file organizations
- **Performance monitoring**
  - Monitors database usage and provides statistics to the DBA

# Tools, Application Environments, and Communications Facilities

- CASE Tools
- **Data dictionary** (**data repository**) **system**
  - Stores design decisions, usage standards, application program descriptions, and user information
- **Application development environments**
- **Communications software**

# Centralized and Client/Server Architectures for DBMSs

- **Centralized DBMSs Architecture**
  - All DBMS functionality, application program execution, and user interface processing carried out on one machine

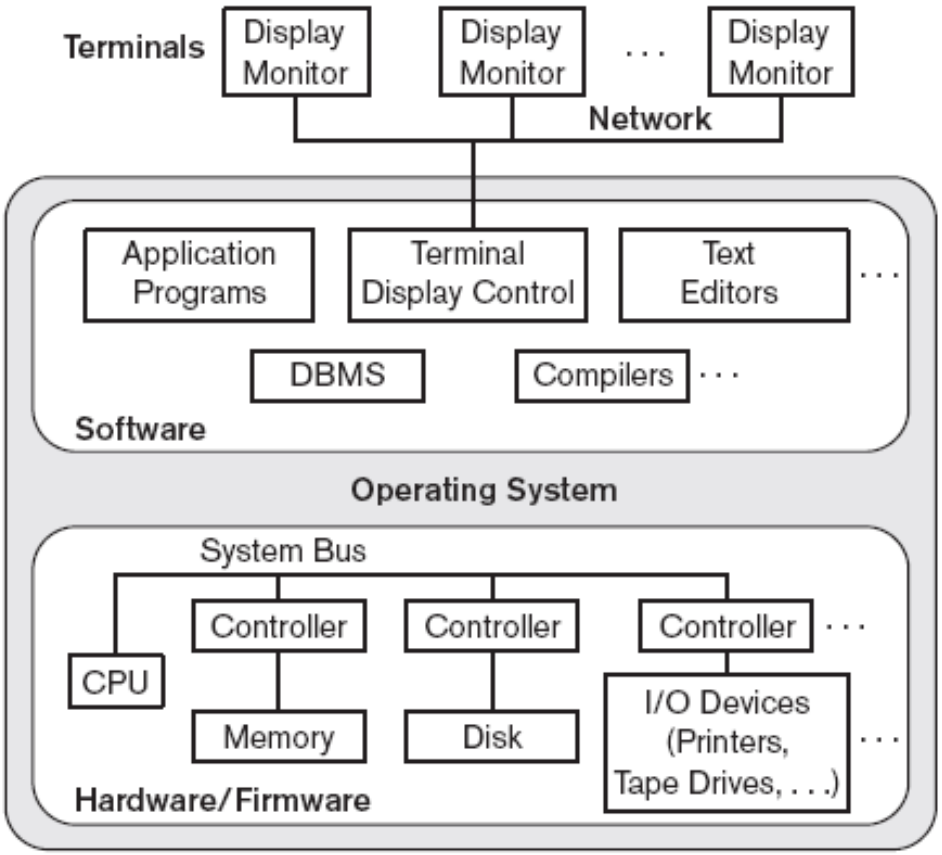



**Figure 2.4**
A physical centralized architecture.

# Basic Client/Server Architectures

- **Servers** with specific functionalities
  - **File server**
    - Maintains the files of the client machines.
  - **Printer server**
    - Connected to various printers; all print requests by the clients are forwarded to this machine
  - **Web servers** or **e-mail servers**

# Basic Client/Server Architectures (cont'd.)

- **Client machines**
  - Provide user with:
    - Appropriate interfaces to utilize these servers
    - Local processing power to run local applications

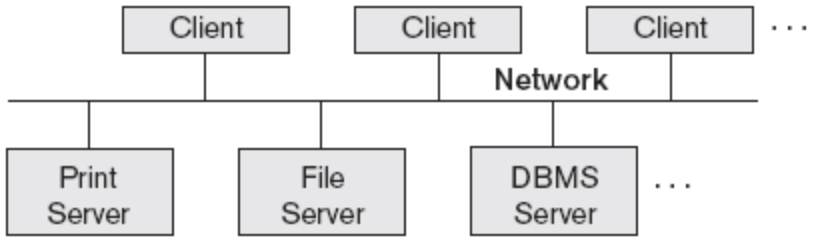



**Figure 2.5**
Logical two-tier client/server architecture.

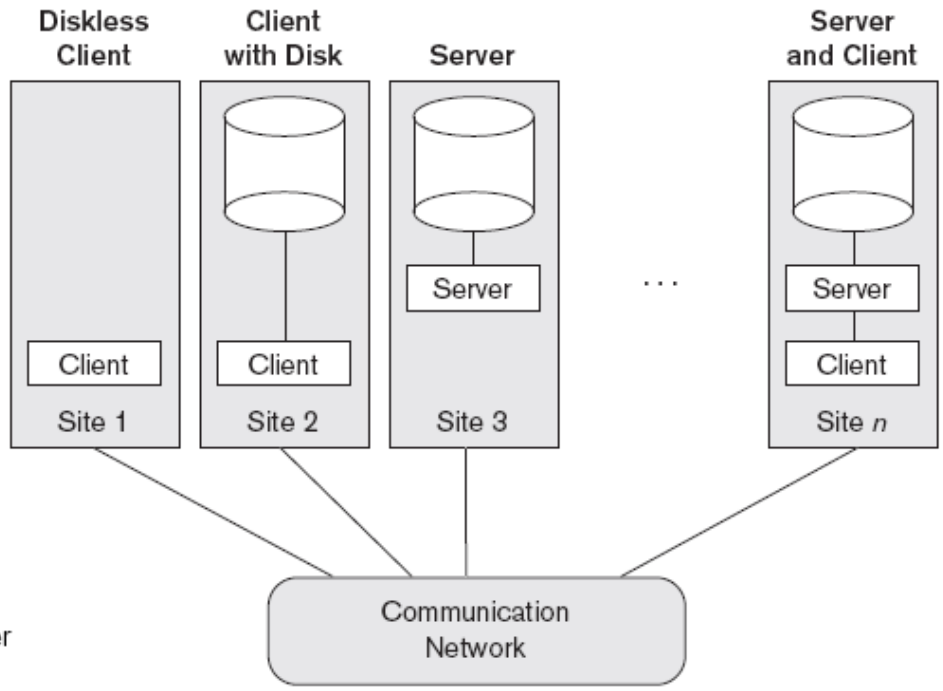



**Figure 2.6**
Physical two-tier client/server architecture.

# Basic Client/Server Architectures (cont'd.)

- **Client**
  - User machine that provides user interface capabilities and local processing
- **Server**
  - System containing both hardware and software
  - Provides services to the client machines
    - Such as file access, printing, archiving, or database access

# Two-Tier Client/Server Architectures for DBMSs

- Server handles
  - Query and transaction functionality related to SQL processing
- Client handles
  - User interface programs and application programs

# Two-Tier Client/Server Architectures (cont'd.)

- Open Database Connectivity (ODBC)
  - Provides application programming interface (API)
  - Allows client-side programs to call the DBMS
    - Both client and server machines must have the necessary software installed
- JDBC
  - Allows Java client programs to access one or more DBMSs through a standard interface

# Three-Tier and n-Tier Architectures for Web Applications

- **Application server** or **Web server**
  - Adds intermediate layer between client and the database server
  - Runs application programs and stores business rules
- N-tier
  - Divide the layers between the user and the stored data further into finer components

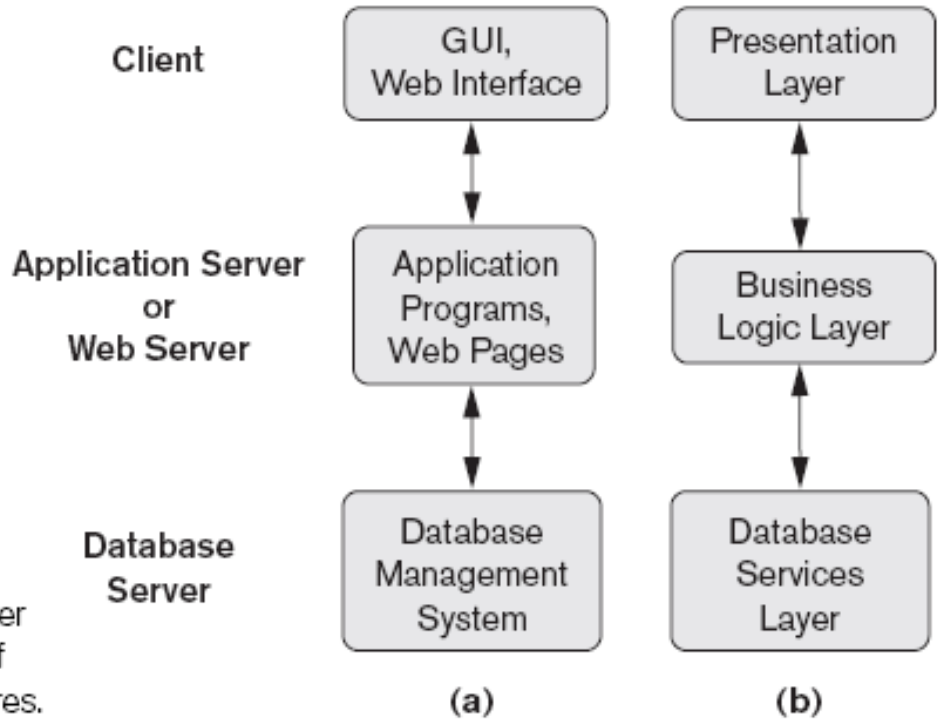



**Figure 2.7**
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

# Classification of Database Management Systems

- **Data model**
  - **Relational**
  - **Object**
  - **Hierarchical** and **network** (legacy)
  - **Native XML DBMS**
- **Number of users**
  - **Single-user**
  - **Multiuser**

# Classification of Database Management Systems (cont'd.)

- **Number of sites**
  - **Centralized**
  - **Distributed**
    - **Homogeneous**
    - **Heterogeneous**
- **Cost**
  - Open source
  - Different types of licensing

# Classification of Database Management Systems (cont'd.)

- **Types of access path options**
- **General** or **special-purpose**

# Classification of Database Management Systems (cont'd.)

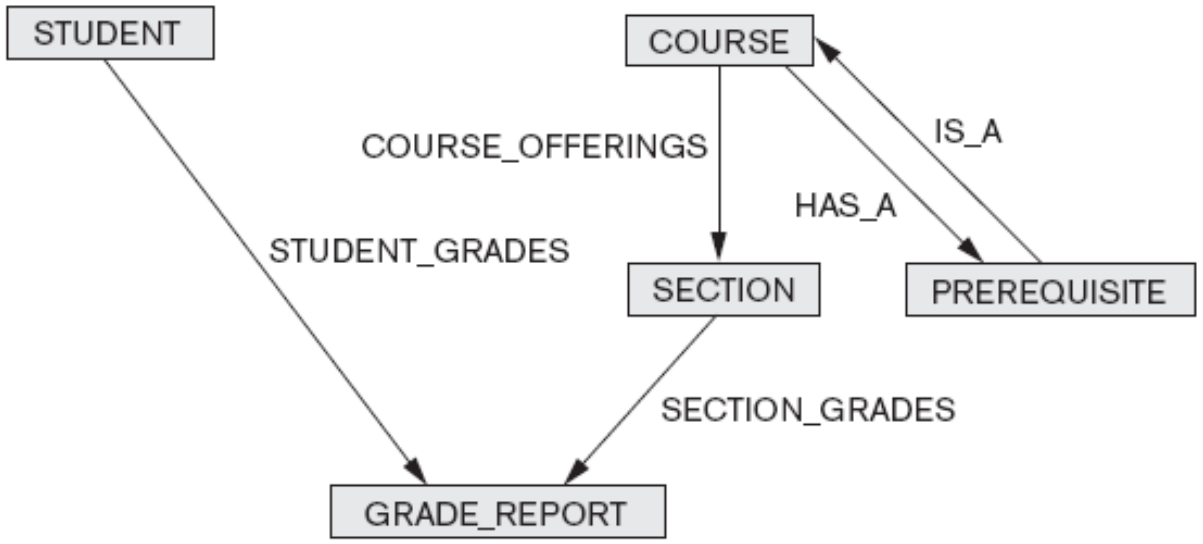



**Figure 2.8**
The schema of Figure 2.1 in network model notation.

[14]CODASYL DBTG stands for Conference on Data Systems Languages Database Task Group, which is the committee that specified the network model and its language.

# Summary

- Concepts used in database systems
- Main categories of data models
- Types of languages supported by DMBSs
- Interfaces provided by the DBMS
- DBMS classification criteria:
  - Data model, number of users, number of sties, access paths, cost