Database System Concepts and Architecture

Data Models

- **Data Model**: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.
- **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model. Operations on the data model may include *basic operations* and *user-defined operations*.

Categories of data models

- **Conceptual** (**high-level**, **semantic**) data models: Provide concepts that are close to the way many users *perceive* data. (Also called **entity-based** or **object-based** data models.)
- **Physical** (**low-level**, **internal**) data models: Provide concepts that describe details of how data is stored in the computer.
- **Implementation** (**representational**) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

The Entity Relationship Model

- The ER model is the most commonly used conceptual model
- In this model, the real world consists of a collection of basic objects called entities and the relationships among these objects
- An *entity* is an object that is distinguishable from other objects by a specific set of attributes
- An *entity set* is the set of all entities of the same type
- A *relationship* is an association among entities
- The set of all relationships of the same type is a *relationship set*
- One nice thing about this model is that you can represent the logical structure of a DB graphically, using an ER diagram.

Example ER Diagram

The Object Oriented Model

- The OO model is a representational data model that is still at a fairly high level
- It's similar to the ER model in that it's based on a collection of objects, but the objects are designed differently
- The real world consists of a collection of objects called *objects*, which store both data values and code for operating on these values
- The values themselves may be objects, and so we can get nesting of objects
- We can also have two or more objects containing all the same values that are nevertheless distinct. Physical address identifiers are used to distinguish them. In the ER model, entities must be distinguished by some unique value.

More Representational Models

- Most representational models are record-based. In a record-based model, data is structured in fixed-format records.
- Each record has a fixed number of fields, and each field usually has a fixed length.
- Two older record-based models, the network model and hierarchical model, are no longer used to build new systems. They use pointers, or hard-coded links, to connect the records of a DB.
- The representational model supported by Oracle is the *relational model.*
- In the relational model, you view data as being arranged in tables, with rows and columns. Each column has a unique name. Each row is a record. The examples given for the University mini-world in Chapter 1 were shown in the relational model.

Definitions

- **Database Schema**: The *description* of a database. Includes descriptions of the database structure and the constraints that should hold on the database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

FIGURE 2.1

Schema diagram for the database in Figure 1.2.

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

PREREQUISITE

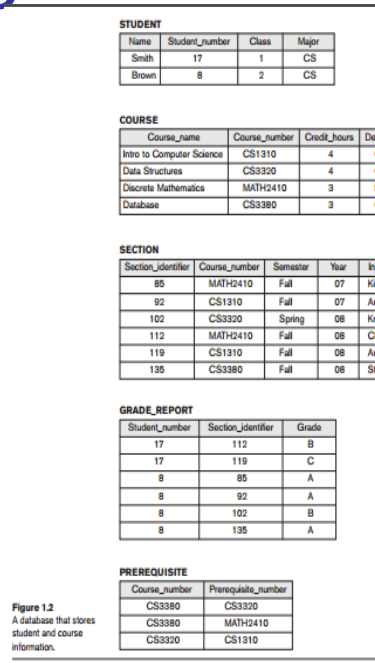
CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	De
Intro to Computer Science	CS1310	4	
Data Structures	CS3320	4	
Discrete Mathematics	MATH2410	3	
Database	CS3380	3	

SECTION

Section_identifier	Course_number	Semester	Year	In
85	MATH2410	Fall	07	Ki
92	CS1310	Fall	07	An
102	CS3320	Spring	08	Ki
112	MATH2410	Fall	08	Ch
119	CS1310	Fall	08	An
135	CS3380	Fall	08	Sm

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

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

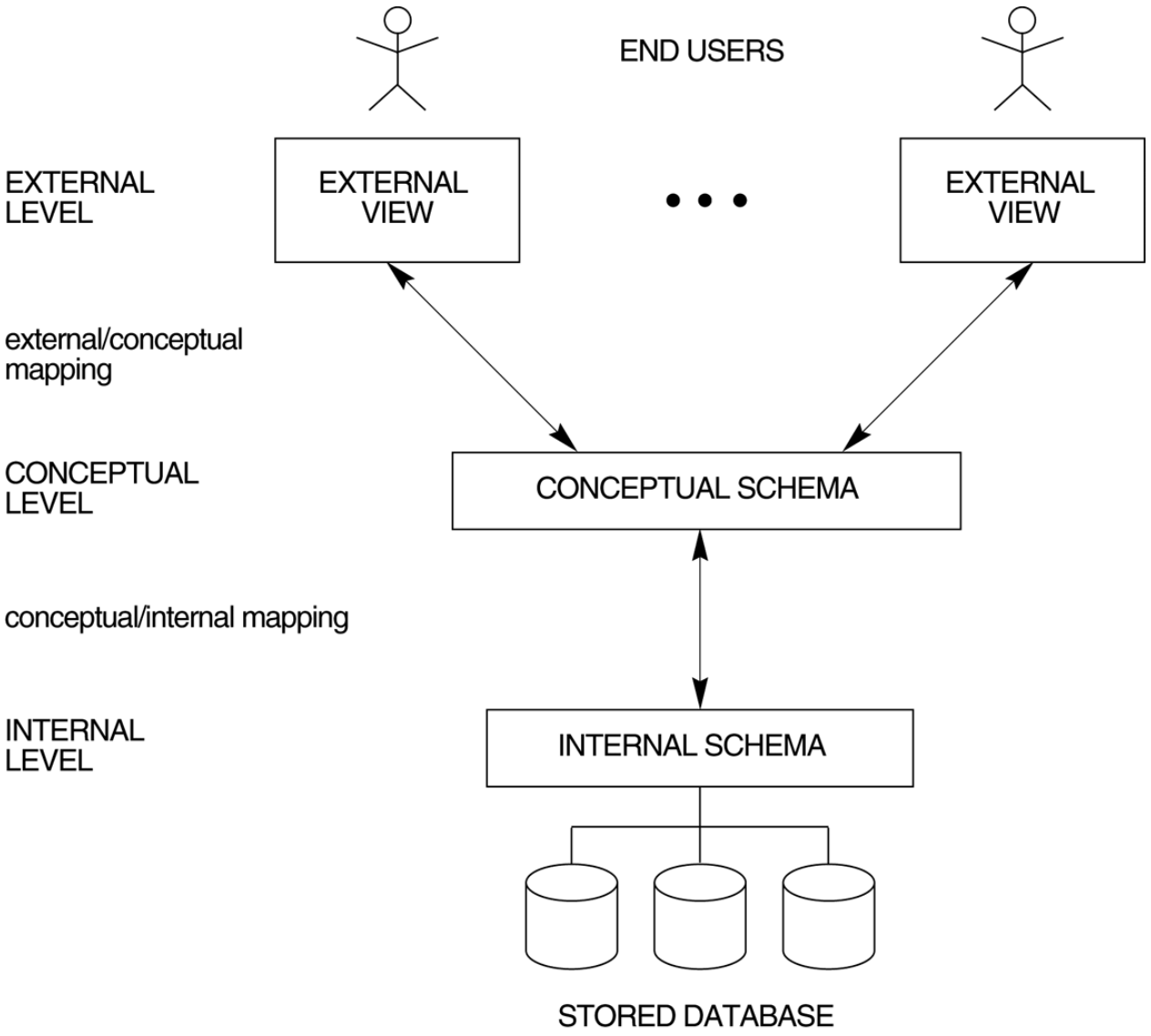
Database Schema Vs. Database State

- **Database State:** Refers to the content of a database at a moment in time.
- **Initial Database State:** Refers to the database when it is loaded
- **Valid State:** A state that satisfies the structure and constraints of the database.
- **Distinction**
  - The **database schema** changes *very infrequently*. The **database state** changes *every time the database is updated.*
  - **Schema** is also called **intension**, whereas **state** is called **extension**.

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence**.
  - Support of **multiple views** of the data.

FIGURE 2.2
The three-schema architecture.



Three-Schema Architecture

- Defines DBMS schemas at *three levels*:
  - **Internal schema** at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.
  - **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.
  - **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

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.

Data Independence

- **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.

Data Independence

When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are *unchanged*. Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

- **Data Definition Language** (**DDL**): Used by the DBA and database designers to specify the *conceptual schema* of a database. In many DBMSs, the DDL is also used to define internal and external schemas (views). In some DBMSs, separate **storage definition language** (**SDL**) and **view definition language** (**VDL**) are used to define internal and external schemas.

DBMS Languages

- **Data Manipulation Language (DML)**: Used to specify database retrievals and updates.
  - DML commands (**data sublanguage**) can be *embedded* in a general-purpose programming language (**host language**), such as COBOL, C or an Assembly Language.
  - Alternatively, *stand-alone* DML commands can be applied directly (**query language**).

DBMS Languages

- **High Level** or **Non-procedural Languages:** e.g., SQL, are *set-oriented* and specify what data to retrieve than how to retrieve. Also called *declarative* languages.
- **Low Level** or **Procedural Languages:** record-at-a-time; they specify *how* to retrieve data and include constructs such as looping.

DBMS Interfaces

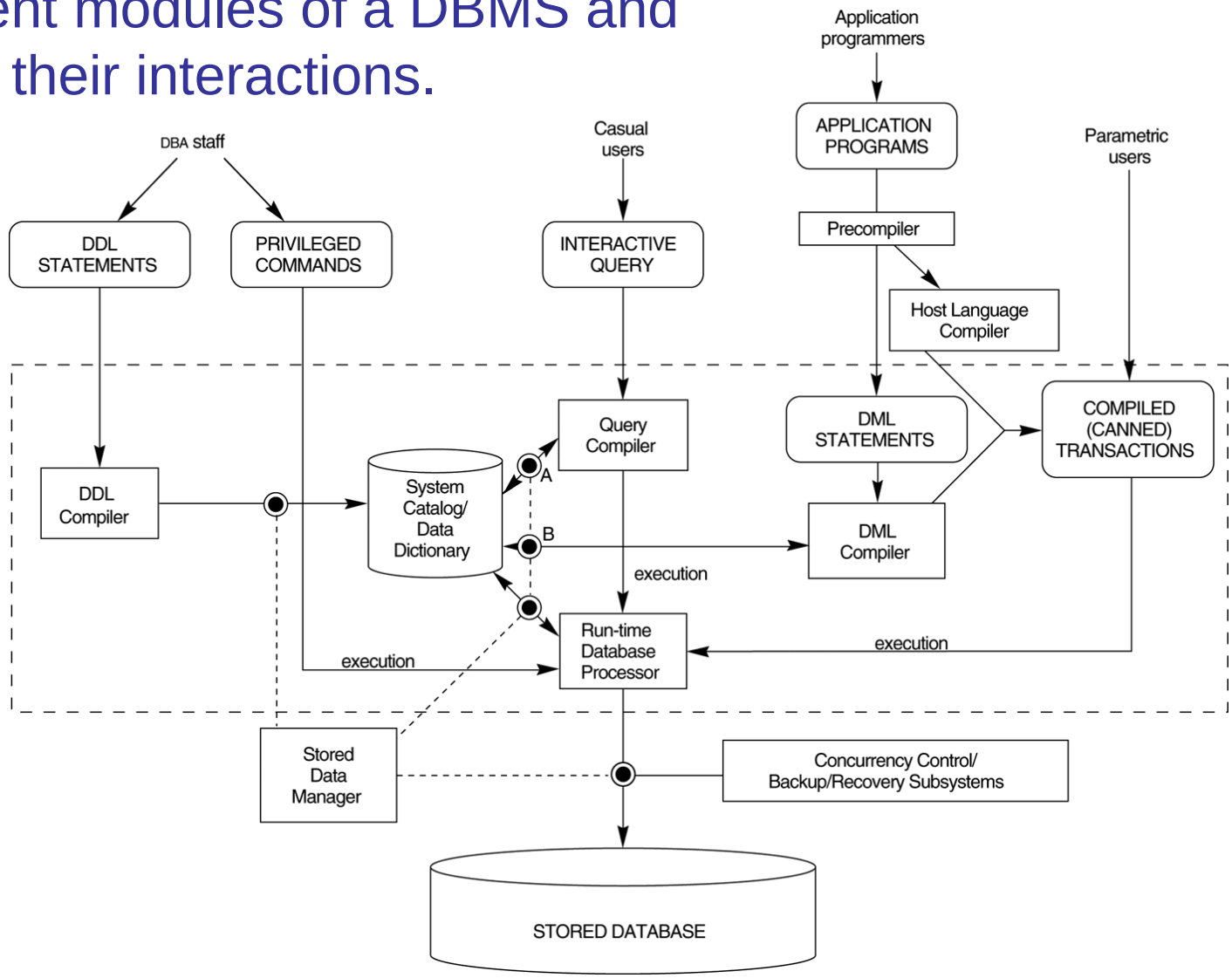
- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naïve users
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Natural language: requests in written English
  - Combinations of the above

Other DBMS Interfaces

- Speech as Input (?) and Output
- Web Browser as an interface
- Parametric interfaces (e.g., bank tellers) using function keys.
- Interfaces for the DBA:
  - Creating accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access path

FIGURE 2.3

Component modules of a DBMS and their interactions.



Database System Utilities

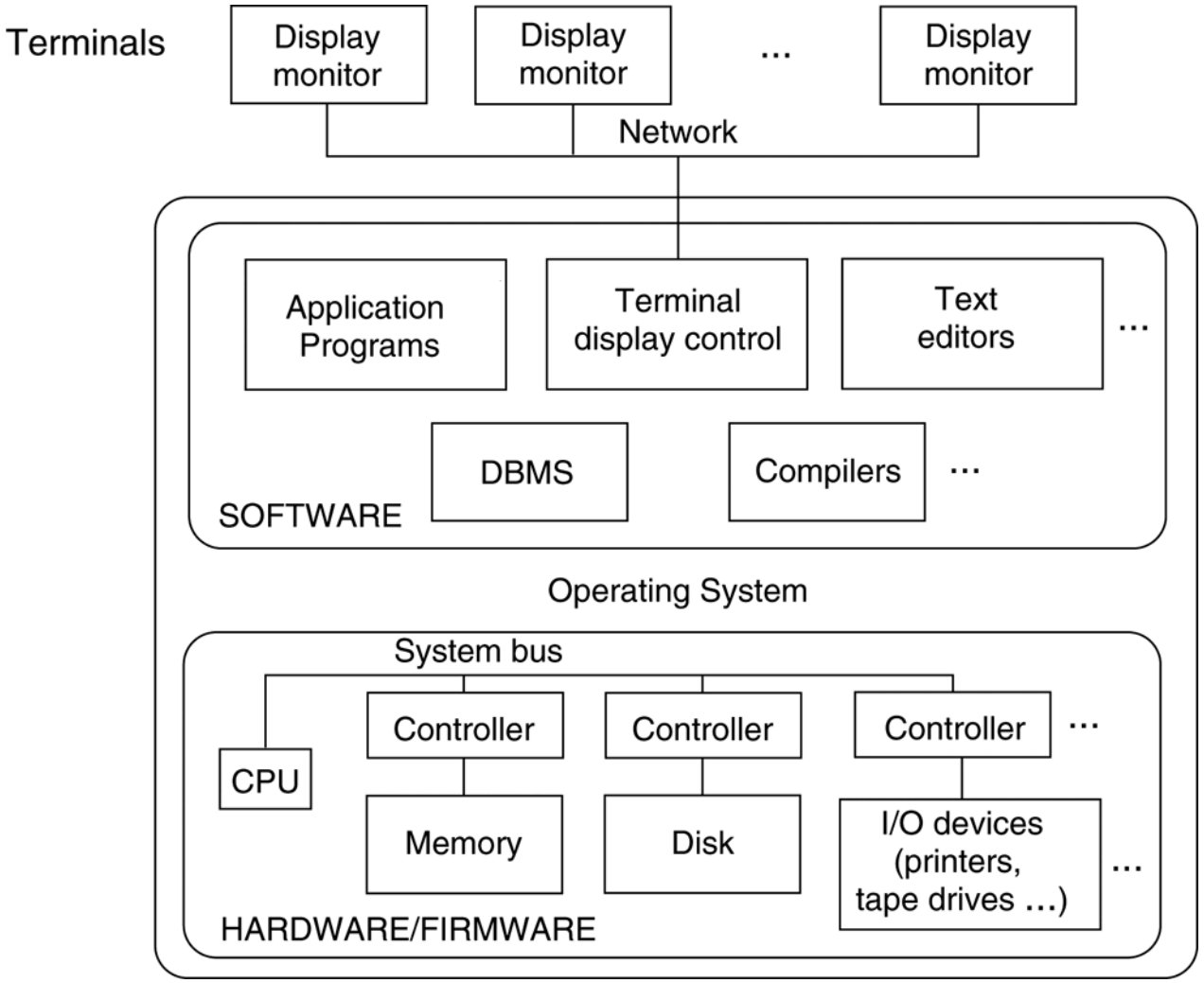
- To perform certain functions such as:
  - *Loading* data stored in files into a database. Includes data conversion tools.
  - *Backing up* the database periodically on tape.
  - *Reorganizing* database file structures.
  - *Report generation* utilities.
  - *Performance monitoring* utilities.
  - Other functions, such as *sorting*, *user monitoring*, *data compression*, etc.

Centralized and Client-Server Architectures

- **Centralized DBMS:** combines everything into single system including- DBMS software, hardware, application programs and user interface processing software.

FIGURE 2.4

A physical centralized architecture.



Specialized Servers with Specialized functions:

- File Servers
- Printer Servers
- Web Servers
- E-mail Servers

Clients:

- Provide appropriate interfaces and a client-version of the system to access and utilize the server resources.
- Clients maybe diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.
  (LAN: local area network, wireless network, etc.)

Two Tier Client-Server Architecture

- **User Interface Programs and Application Programs** run on the client side
- Interface called **ODBC (Open Database Connectivity** – see Ch 9**)** provides an Application program interface (API) allow client side programs to call the DBMS. Most DBMS vendors provide ODBC drivers.

Two Tier Client-Server Architecture

- A client program may connect to several DBMSs.
- Other variations of clients are possible: e.g., in some DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc. In such situations the server may be called the **Data Server**.

FIGURE 2.5

Logical two-tier client/server architecture.

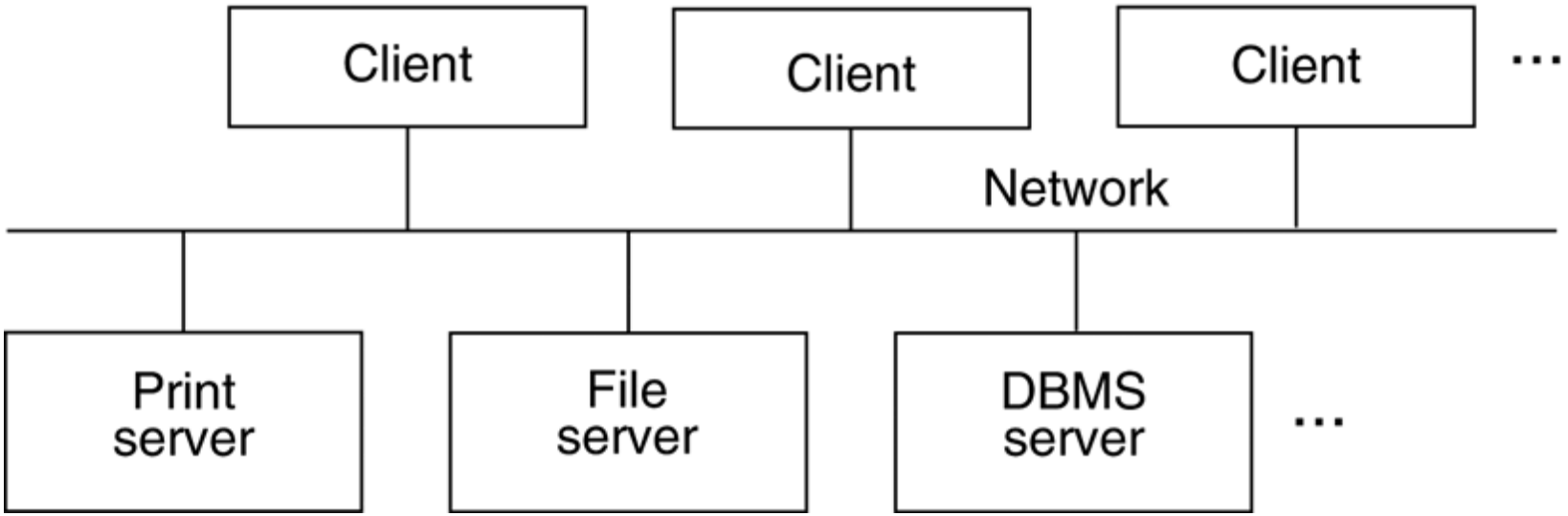
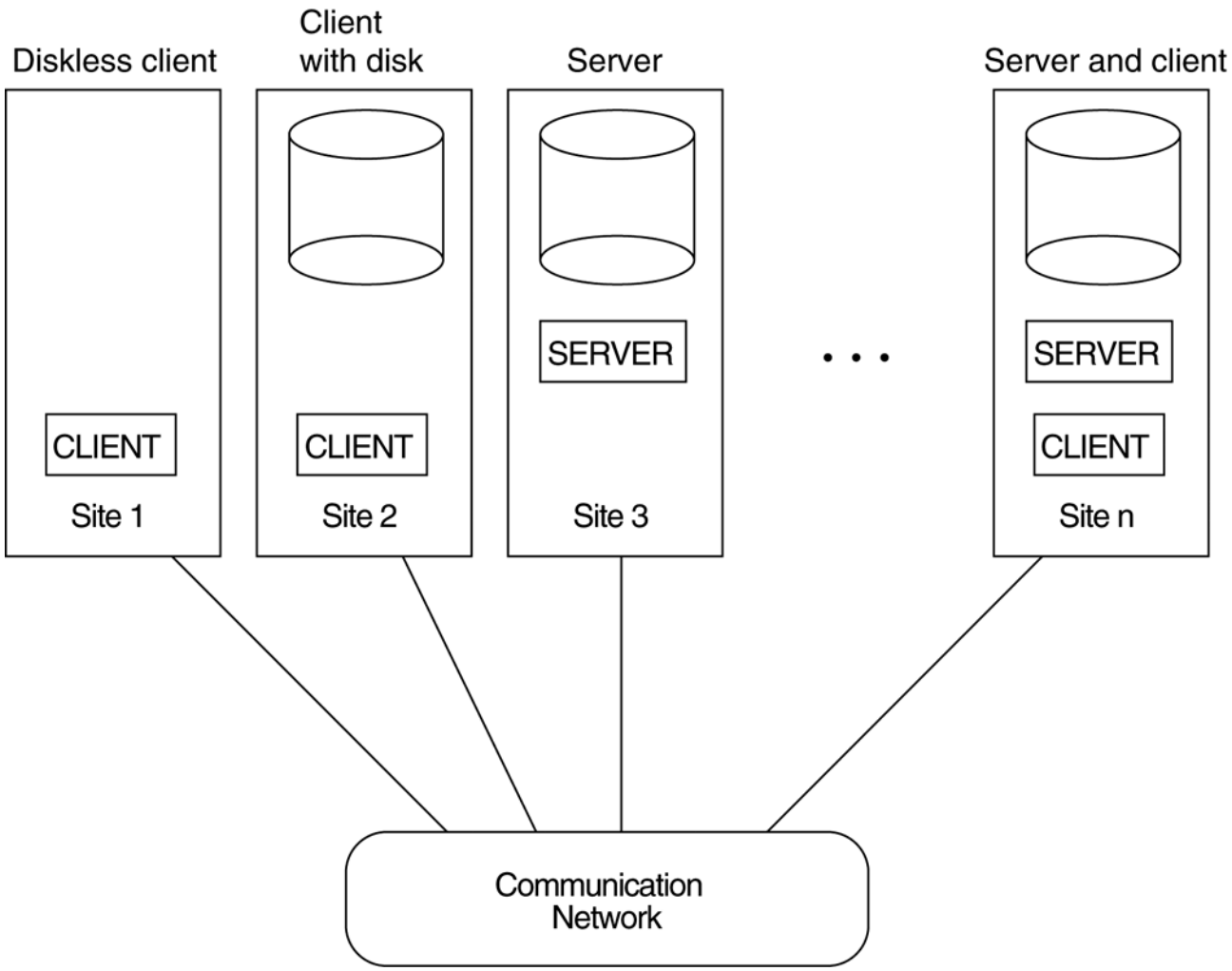


FIGURE 2.6

Physical two-tier client-server architecture.

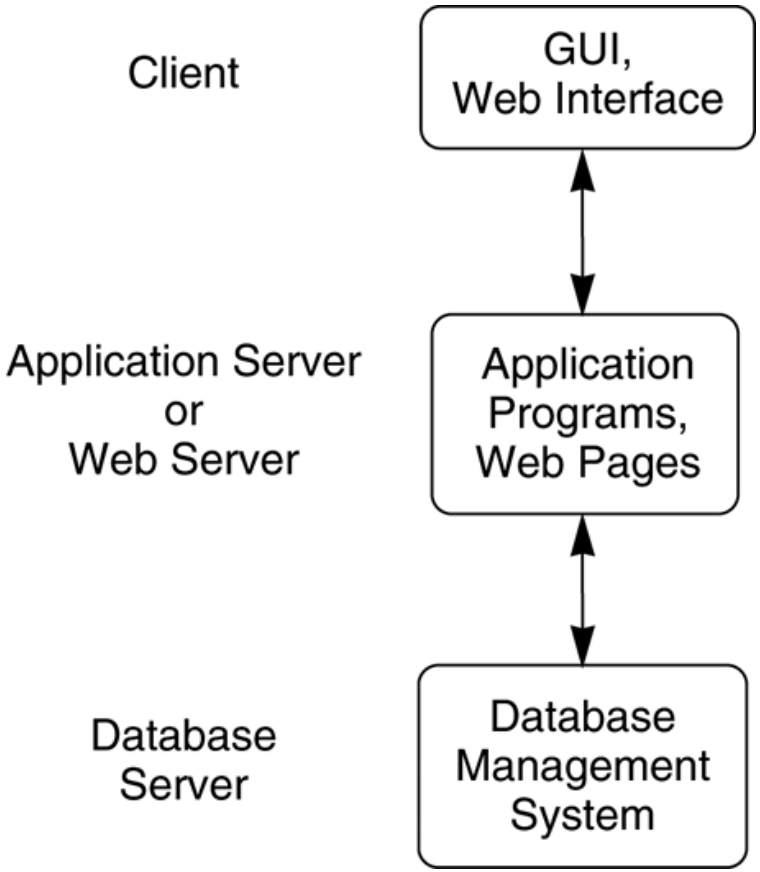


Three Tier Client-Server Architecture

- Common for **Web applications**
- Intermediate Layer called **Application Server** or **Web Server:**
  - stores the web connectivity software and **the rules and business logic (constraints)** part of the application used to access the right amount of data from the database server
  - acts like a conduit for sending partially processed data between the database server and the client.
- **Additional Features- Security:**
  - encrypt the data at the server before transmission
  - decrypt data at the client

FIGURE 2.7

Logical three-tier client/server architecture.



Classification of DBMSs

- **Based on the data model used**:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- **Other classifications:**
  - Single-user (typically used with micro-computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)