Database I

(60012301-1)

Lecture 2: Database System Concepts and Architecture

Dr. Obead Alhadreti

Outline

- Data Models and Their Categories
- Schemas and States
- Three-Schema Architecture & Data Independence
- DBMS Languages and Interfaces
- The Database System Environment
- DBMS Architectures
- Classification of DBMSs

Data Models

- **Data Model:**
  - A set of concepts to describe the *structure* of a database, the *operations* for manipulating these structures, and certain *constraints* that the database should obey.

- **Database Structure:**
  - By data structure, we mean the data types, relationships, and constraints that should hold on the data.

Data Models (continued)

- **Data Model Operations:**
  - Operations are used for specifying retrievals and updates on the database by referring to the concepts of the data model.
  - Operations on the data model may include:
    - basic model operations(insert, delete, update)
    - user-defined operations(compute_student_gpa, update_inventory)

Categories of Data Models

1. **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.

2. **Physical (low-level, internal) data models:**
   - Provide concepts that describe details of how data is stored in the computer.
   - Meant for computer specialists, not for typical end users.

3. **Implementation (representational) data models:**
   - Provide concepts that may be understood by end users but that are not too far from the way data is organized within the computer.

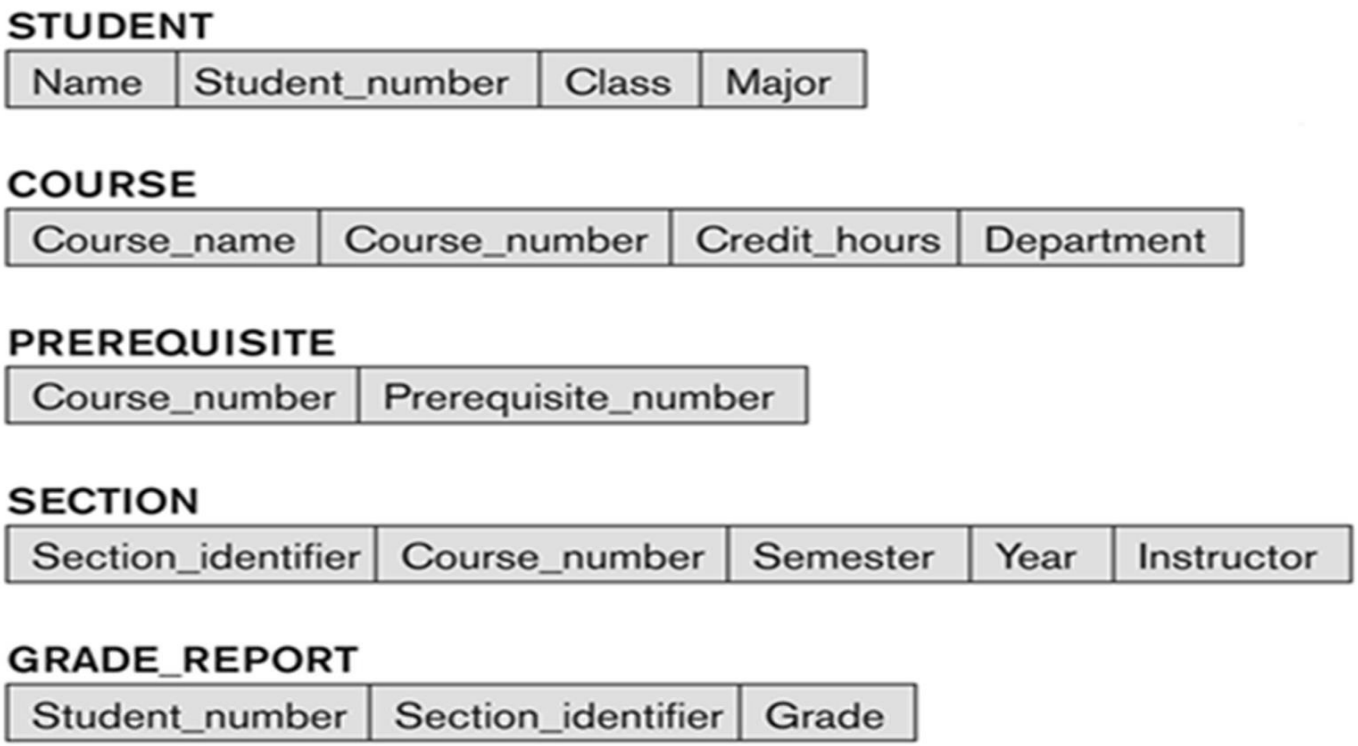
Database Schema & State

- **Database Schema**:
  - The *description* of a database.
  - Not expected to change frequently

- **Schema Diagram**:
  - A diagrammatic display of (some aspects of) a database schema.
  - Changes applied to schema as application requirements change.

- **Database State**:
- The actual data stored in a database at a particular moment in time. Also called database *instance* or *occurrence*.
- Every update operation changes the database from one state to another

Example of a Database Schema



Example of a Database State

STUDENT

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

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

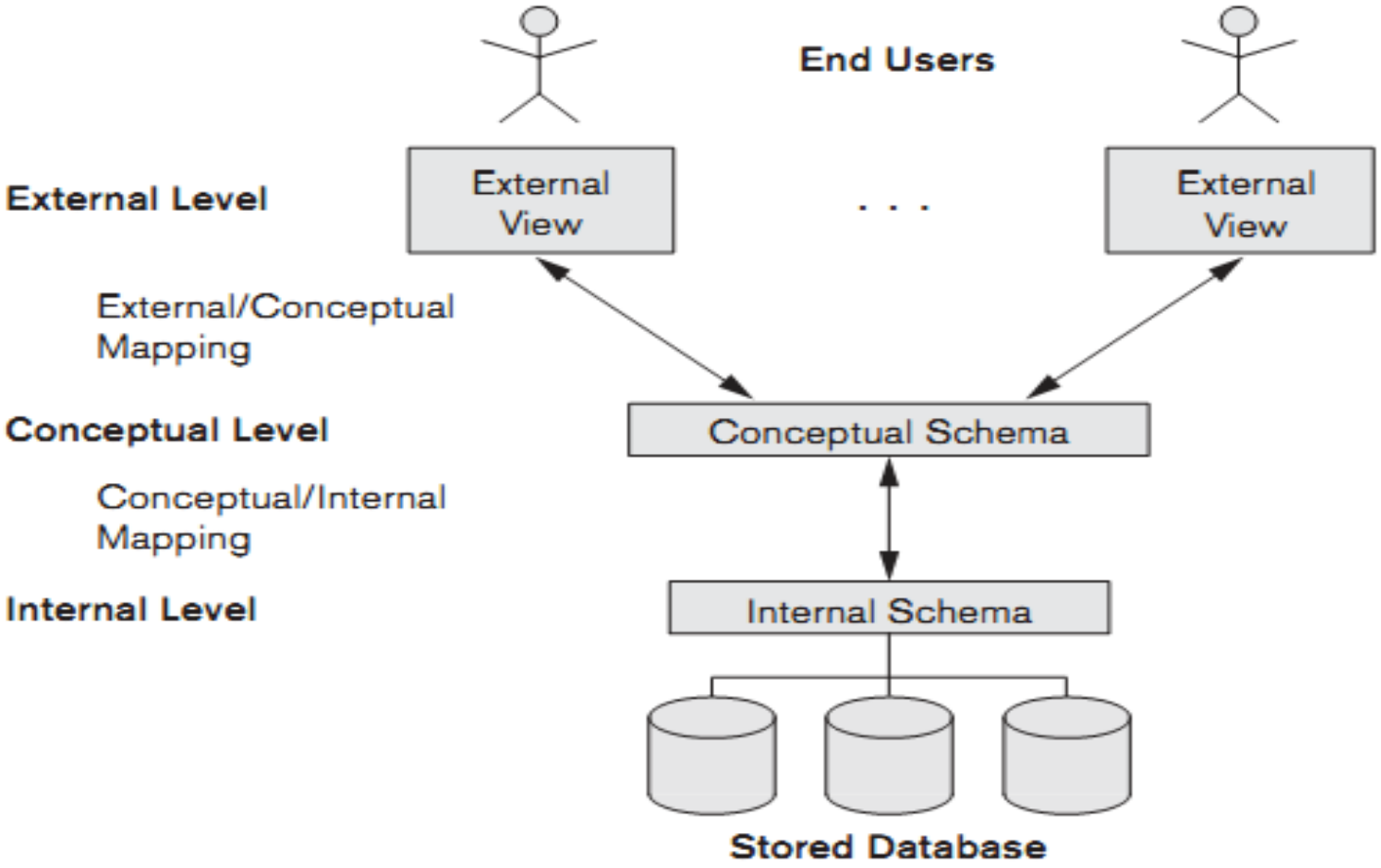
PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

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. It describes part of the database that a particular user group is interested in.
    - Usually uses the same data model as the *conceptual* level.

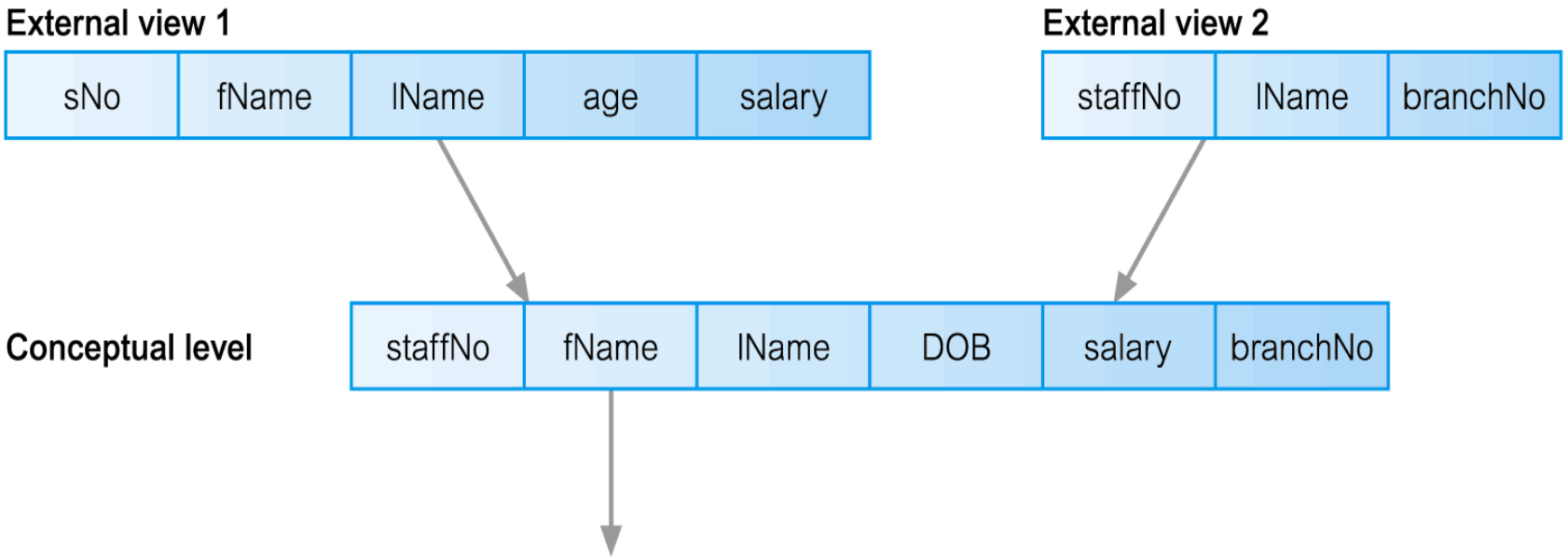
Three-Schema Architecture (continued)



Three-Schema Architecture (continued)

- Proposed to support DBMS characteristics of:
  - **Program-data independence**.
  - Support of **multiple views** of the data.
- **Objectives of Three-Schema Architecture:**
  1. All users should be able to access same data.
  2. A user's view is immune to changes made in other views.
  3. Users should not need to know physical database storage details.
  4. Database administrator (DBA) should be able to change database storage structures without affecting the users' views.
  5. Internal structure of database should be unaffected by changes to physical aspects of storage.
  6. DBA should be able to change conceptual structure of database without affecting all users.

Three-Schema Architecture (continued)



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 */
```

Data Independence

- Data Independence refers to the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

- Logical Data Independence: change the conceptual schema without having to change the external schemas and their associated application programs.

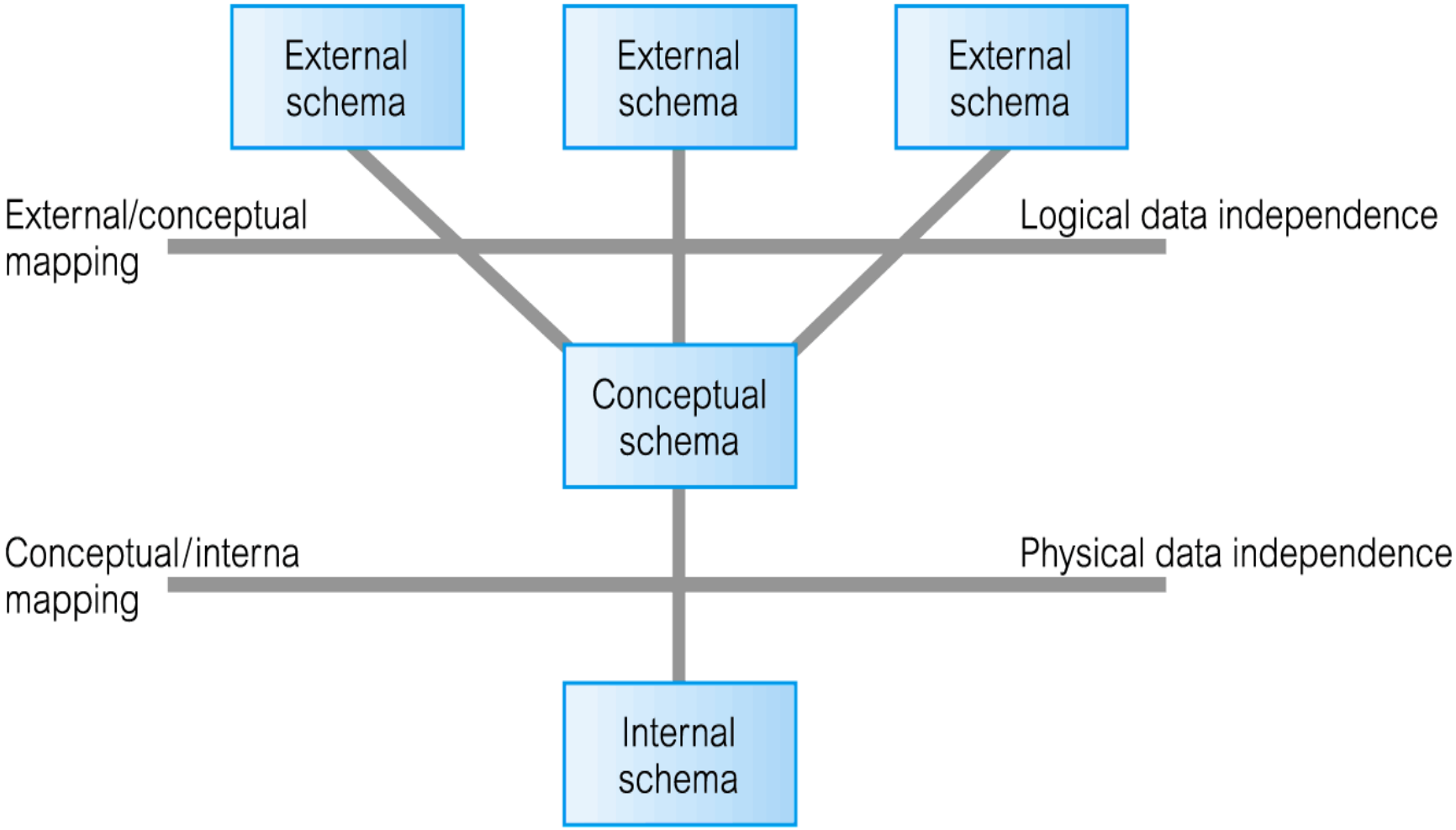
- Physical Data Independence: change the internal schema without having to change the conceptual schema.

Data Independence (continued)

- 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.

Data Independence (continued)



DBMS Languages

- **Data Definition Language (DDL):**
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - Examples: Create, alter, and drop schema.
  - 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):**
  - A language that is used to manipulate (retrieve, insert, delete, and modify) data.
  - Examples : DELETE, INSERT, SELECT, UPDATE
- **Data Control Language (DCL):**
  - Defines activities that are not in the categories of those for the DDL and DML, such as granting privileges to users, and defining when proposed changes to a databases should be irrevocably made.

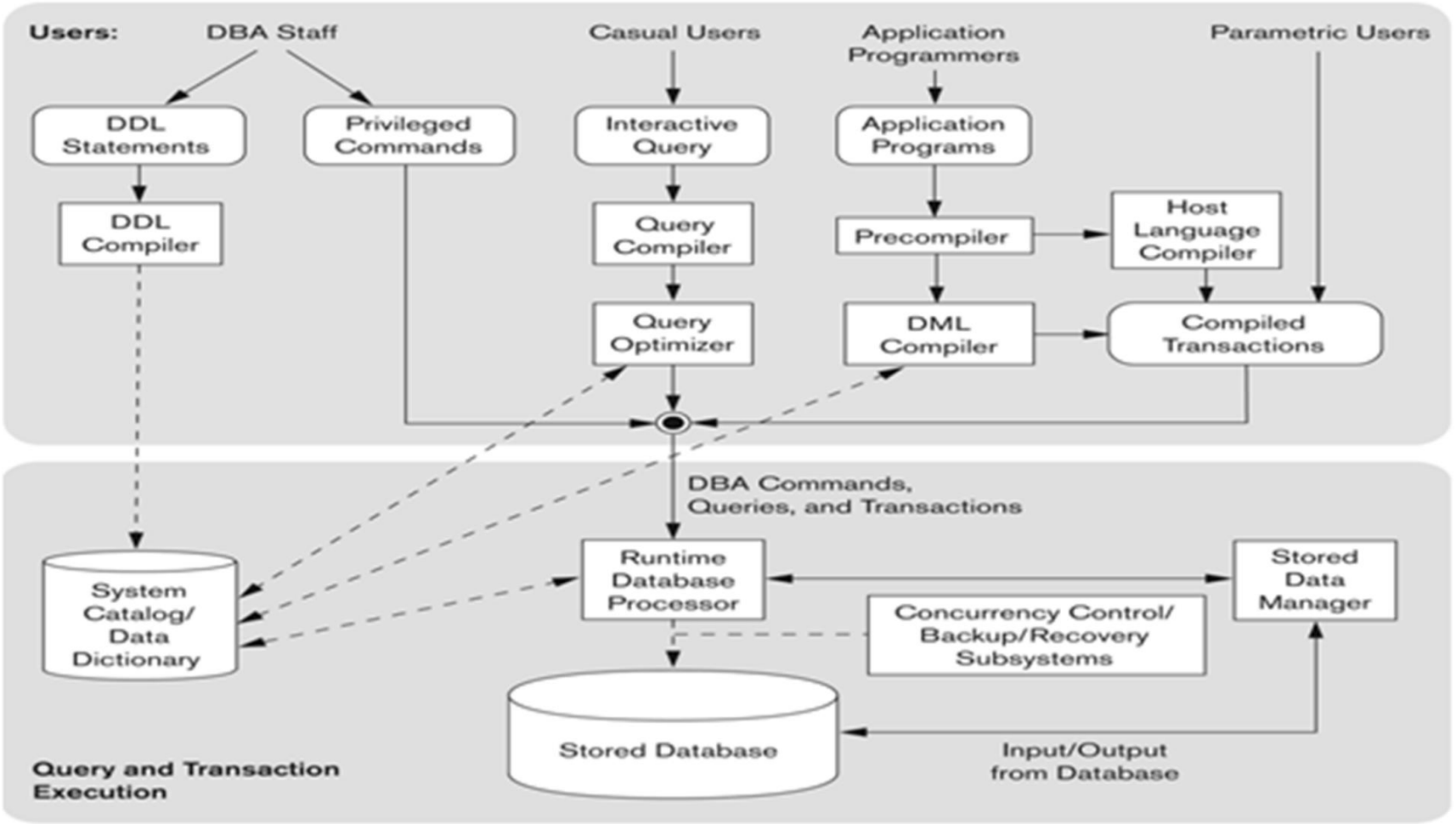
DBMS Interfaces

- 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

The Database System Environment

- **DBMS component modules:**
  - Buffer management
  - Stored data manager
  - DDL compiler
  - Interactive query interface
    - Query compiler
    - Query optimizer
  - Precompiler
  - Runtime database processor
  - System catalog
  - Concurrency control system
  - Backup and recovery system

Typical DBMS Component Module



Database System Utilities

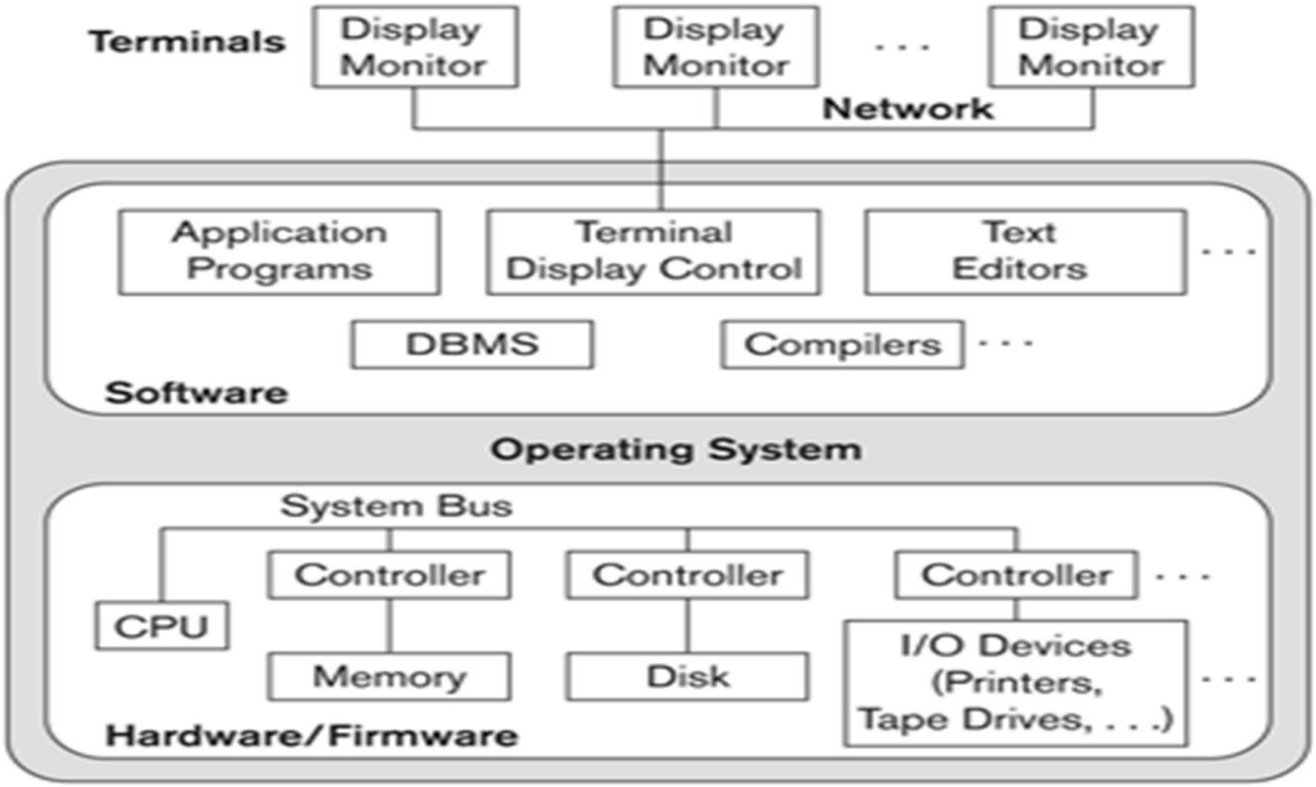
- Loading
  - Load existing data files
- Backup
  - Creates a backup copy of the database
- Database storage reorganization
  - Reorganize a set of database files into different file organizations
- Performance monitoring
  - Monitors database usage and provides statistics to the DBA
- Other Utilities
  - Sorting files
  - Handling data compression
  - Monitoring access by users

Tools, Application Environments, and Communications Facilities

- CASE tools
  - Used in the design phase
- Expanded Data Dictionary(information repository)
  - Stores catalog information about schemas and constraints
  - Stores other information such as design decisions, usage standards, application program descriptions, and user information
- Application Development Environments
  - Provide an environment for developing database applications
  - Include facilities for database design, GUI development, querying and updating, and application program development
- Communications software
- Allow users to access database from remote location

Centralized DBMS Architectures

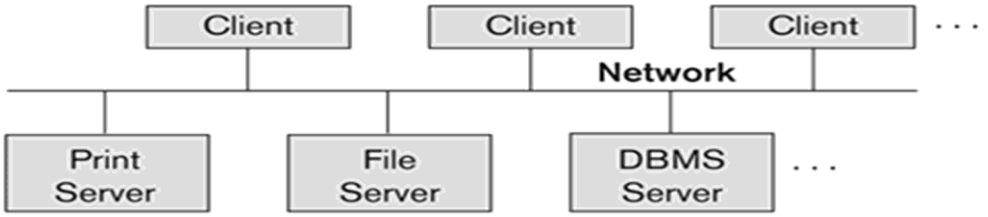
- All DBMS functionality, application program execution, and user interface processing carried out on one machine



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

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



Basic Client-Server Architectures (continued)

- Clients
  - Provide appropriate interfaces through a client software module to access and utilize the various server resources.
  - Clients may be 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.)
- DBMS Server
  - Provides database query and transaction services to the clients
  - Relational DBMS servers are often called SQL servers, query servers, or transaction servers

Two-Tire Client-Server Architectures

- **Client handles**
- User Interface Programs and Application Programs run on the client side.

- **Server handles**

  Query and transaction functionality related to SQL processing.

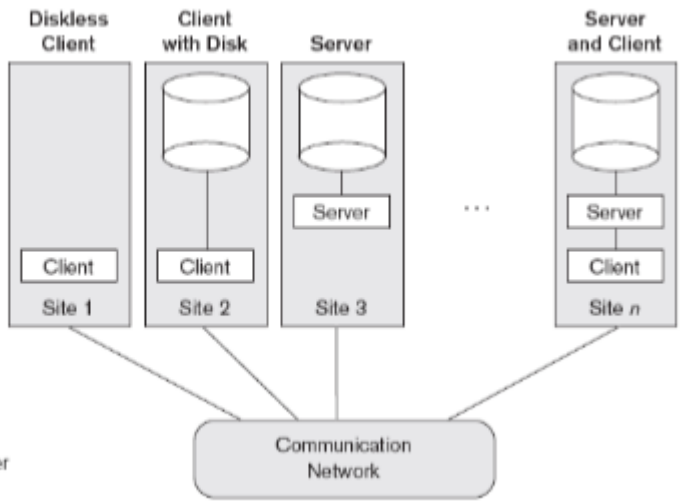


Figure 2.6
Physical two-tier client/server architecture.

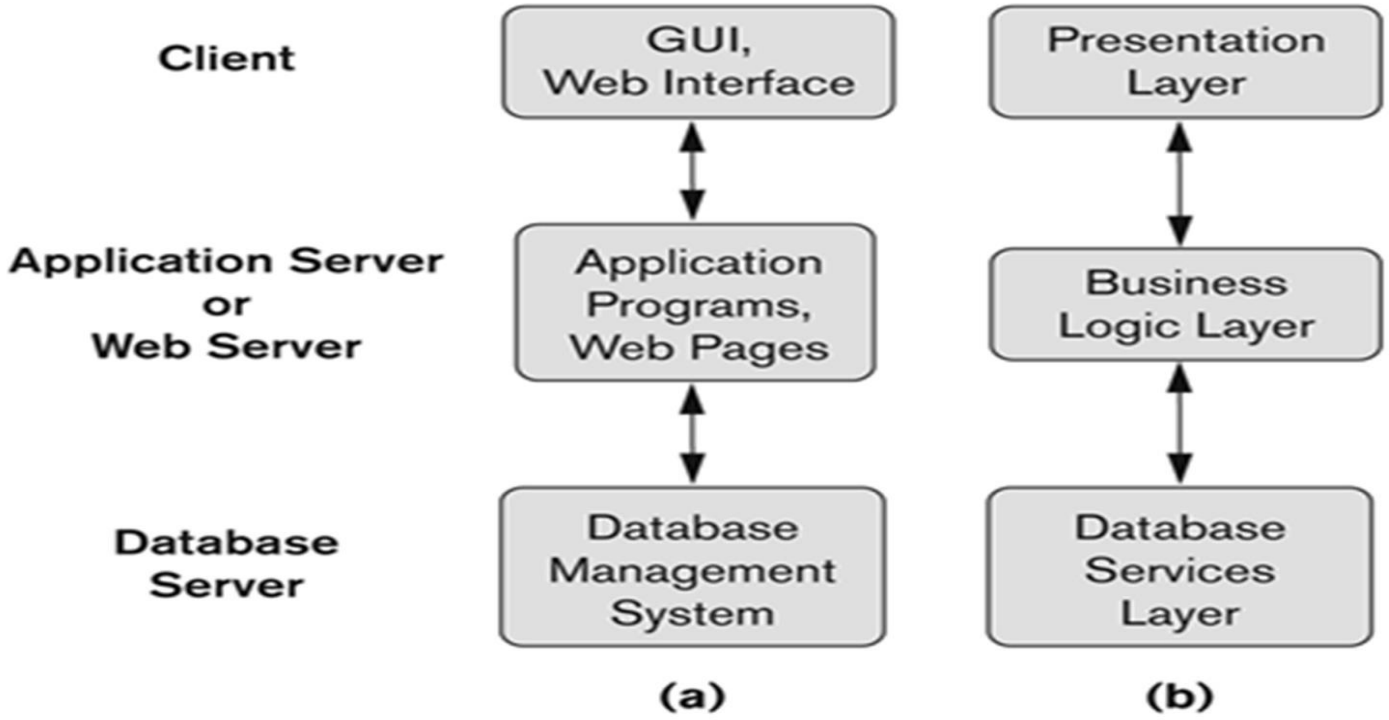
Three-Tier Client-Server Architectures

- Common for **Web applications.**

- Adds intermediate layer between client and the database server 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

Three-Tier Client-Server Architectures



Classification of DBMSs

- **Based on the data model used**:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.

- **Other classifications:**
  - Number of users: Single-user (typically used with micro-computers) *or* multi-user (most DBMSs).
  - Number of sites: Centralized (uses a single computer with one database) *or* distributed (uses multiple computers, multiple databases)
  - Cost: Open source *or* different types of licensing
  - Purpose: General *or* Special-purpose