

Databases



Spring-2025
Lec02





Content

Chapter Two: Database System Concepts and Architecture

- ◆ Data Models and Their Categories
- ◆ History of Data Models
- ◆ Schemas, Instances, and States
- ◆ Three-Schema Architecture
- ◆ Data Independence
- ◆ DBMS Languages and Interfaces
- ◆ Database System Utilities and Tools
- ◆ Centralized and Client-Server Architectures





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.

- ◆ **Data Model Structure and Constraints:**

- ✓ Constructs are used to define the **database structure**
 - ✓ Constructs typically include elements (and their **data types**) as well as groups of elements (e.g. **entity**, **record**, **table**), and relationships **among such groups**
 - ✓ Constraints specify some restrictions on **valid** data; these constraints must be enforced at all times





Data Models (continued)

- ◆ **Data Model Operations:**

- ✓ These operations are used for specifying database **retrievals** and **updates** by referring to the constructs of the data model.
- ✓ Operations on the data model may include **basic model operations** (e.g. generic insert, delete, update) and user-defined **operations** (e.g. compute-student-gpa, update-inventory)



Categories of Data Models

- ◆ **Conceptual (high-level, semantic) data models:**
 - ✓ Provide **concepts** that are **close to the way many users perceive data.**
 - ✓ (Also **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. These are usually specified in an **ad-hoc** manner through **DBMS design** and **administration manuals**



Categories of Data Models

- ◆ **Implementation (representational) data models:**
 - ✓ Provide concepts that fall **between** the above two, used by many commercial DBMS implementations (e.g. **relational data models** used in many commercial systems).
- ◆ **Self-Describing Data Models:**
 - ✓ Combine the **description** of data with the **data values**. Examples include XML, key-value stores and some **NoSQL** systems.



Schemas versus Instances

- ◆ Database Schema:
 - ✓ The **description** of a database.
 - ✓ Includes descriptions of the database **structure, data types, and the constraints** on the database.
- ◆ Schema Diagram:
 - ✓ An illustrative **display** of (most aspects of) a database schema.
- ◆ Schema Construct:
 - ✓ A component of the schema or an **object** within the schema, e.g., STUDENT, COURSE.



Schemas versus Instances

- ◆ **Database State:**
 - ✓ The actual **data** stored in a database at a **particular moment** in time. This includes the collection of all the data in the database.
 - ✓ Also called **database instance** (or **occurrence** or **snapshot**).
 - ❖ The term **instance** is also applied to individual database components, e.g. record instance, table instance, entity instance



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 state when it is initially loaded into the system.
- ◆ **Valid State:**
 - ✓ A state that satisfies the structure and constraints of the database.



Database Schema vs. Database State (continued)

- ◆ **Distinction**
 - ✓ The database schema **changes** very infrequently.
 - ✓ The database state changes every time the database is updated
- ◆ **Schema** is also called **intension**.
- ◆ **State** is also called **extension**.



Example of a Database Schema

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
----------------	--------------------	-------



Figure 2.1

Schema diagram for the database in Figure 1.2.

Example of a database state

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	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	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

Figure 1.2
A database that stores student and course information.





Three-Schema Architecture

- ◆ Proposed to support DBMS characteristics of:
 - ✓ Program-data independence.
 - ✓ Support of multiple views of the data.
- ◆ Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization





Three-Schema Architecture

- ◆ Defines DBMS schemas at three levels:
 - ✓ Internal schema at the internal level to describe physical storage structures and access paths (e.g indexes).
 - ❖ 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.





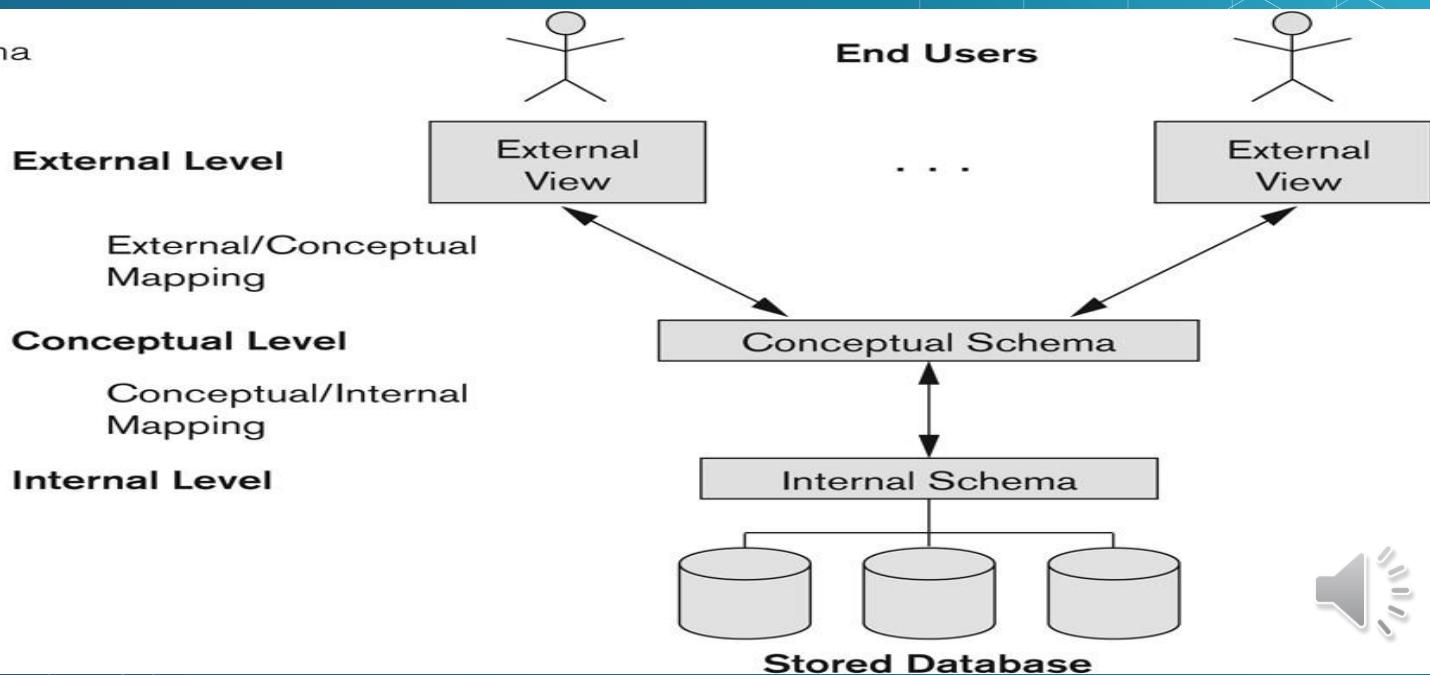
Three-Schema Architecture

- External schemas at the external level to describe the various user views.
 - ✓ Usually uses the same data model as the conceptual schema.



The three-schema architecture

Figure 2.2
The three-schema architecture.





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 **extracted** from the **internal DBMS level** is **reformatted** to match the **user's external view** (e.g. formatting the results of an SQL query for display in a Web page)





Data Independence

- ◆ **Logical Data Independence:**
 - ✓ The capacity to change the **conceptual schema** **without** having to change the **external schemas** and their associated application programs.
- ◆ **Physical Data Independence:**
 - ✓ The capacity to change the **internal schema** without having to change the conceptual schema.
For example, the internal schema may be changed **when** certain file structures are reorganized or **new indexes** are created to improve database performance



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.
 - ❖ SDL is typically realized via DBMS commands provided to the DBA and database designers





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, C++, or Java.
 - ✓ Alternatively, stand-alone DML commands can be applied directly (called a query language).





Types of DML

- ◆ High Level or Non-procedural Language:
 - ✓ For example, the SQL relational language
 - ✓ Are “**set**”-oriented and **specify** what data to **retrieve** rather than how **to retrieve it**.
 - ✓ Also called **declarative** languages.
- ◆ Low Level or Procedural Language:
 - ✓ Retrieve data **one record-at-a-time**;
 - ✓ Constructs such as **looping** are needed to **retrieve multiple records**, along with positioning pointers.



DBMS Interfaces

- ◆ Stand-alone query language interfaces
 - ✓ Example: Entering SQL queries at the DBMS **interactive** SQL interface (e.g. SQL*Plus in ORACLE)
- ◆ Programmer interfaces for embedding **DML** in programming languages
- ◆ User-friendly interfaces
 - ✓ **Menu-based, forms-based, graphics-based, etc.**
- ◆ Mobile Interfaces: Interfaces allowing users to perform **transactions** using mobile apps





User-Friendly DBMS Interfaces

- ◆ Menu-based (Web-based), **popular** for browsing on the web
- ◆ Forms-based, designed for naïve users used to **filling** in entries on a form
- ◆ Graphics-based
 - ✓ Point and Click, Drag and Drop, etc.
 - ✓ Specifying a query on a schema diagram
- ◆ Natural language: requests in written English





Other DBMS Interfaces

- ◆ Speech : Input query and Output response
- ◆ Web Browser with keyword search
- ◆ Parametric interfaces, e.g., bank tellers using function keys.
- ◆ Interfaces for the DBA:
 - ✓ Creating user accounts, granting authorizations
 - ✓ Setting system parameters
 - ✓ Changing schemas or access paths



Database System Utilities

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





Other Tools

- ◆ Data dictionary / repository:
 - ✓ Used to **store schema descriptions** and other information such as application program descriptions, user information, usage standards, etc.
 - ✓ Active data dictionary is accessed by **DBMS software** and **users/DBA**.
 - ✓ Passive data dictionary is accessed by **users/DBA** only.



Centralized and Client-Server DBMS Architectures

- ◆ **Centralized DBMS:**
 - ✓ Combines everything into **single system** including- DBMS software, hardware, application programs, and user interface processing software.
 - ✓ User can still connect through a **remote terminal** – however, **all processing is done at centralized site.**



A Physical Centralized Architecture

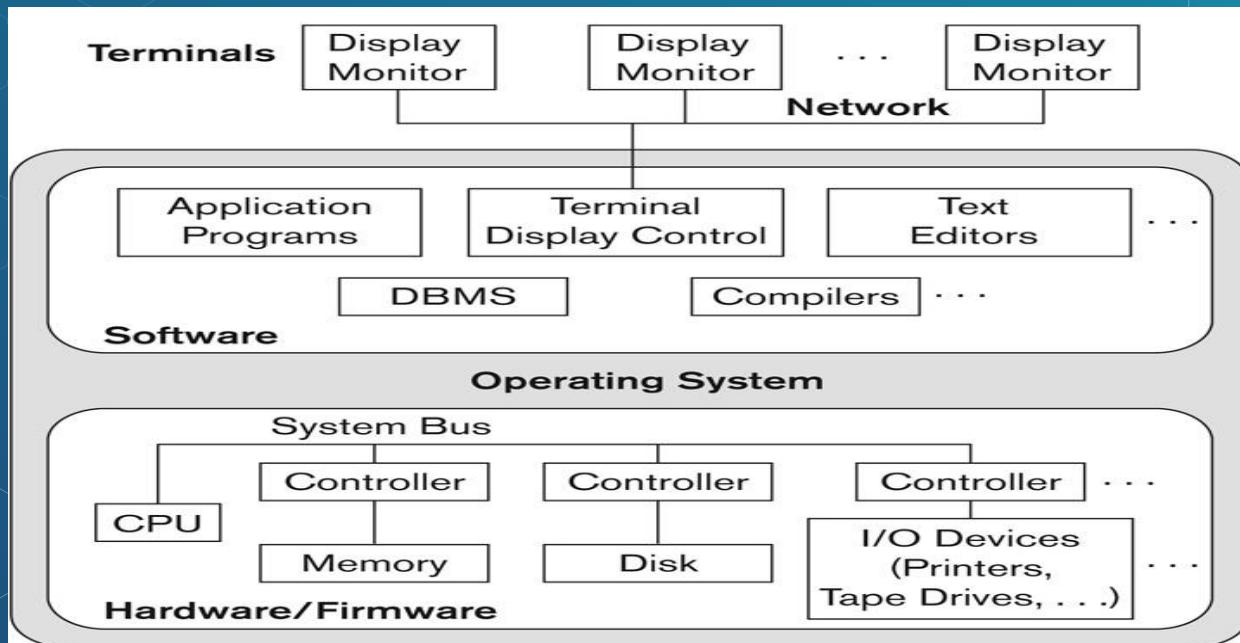


Figure 2.4

A physical centralized architecture.





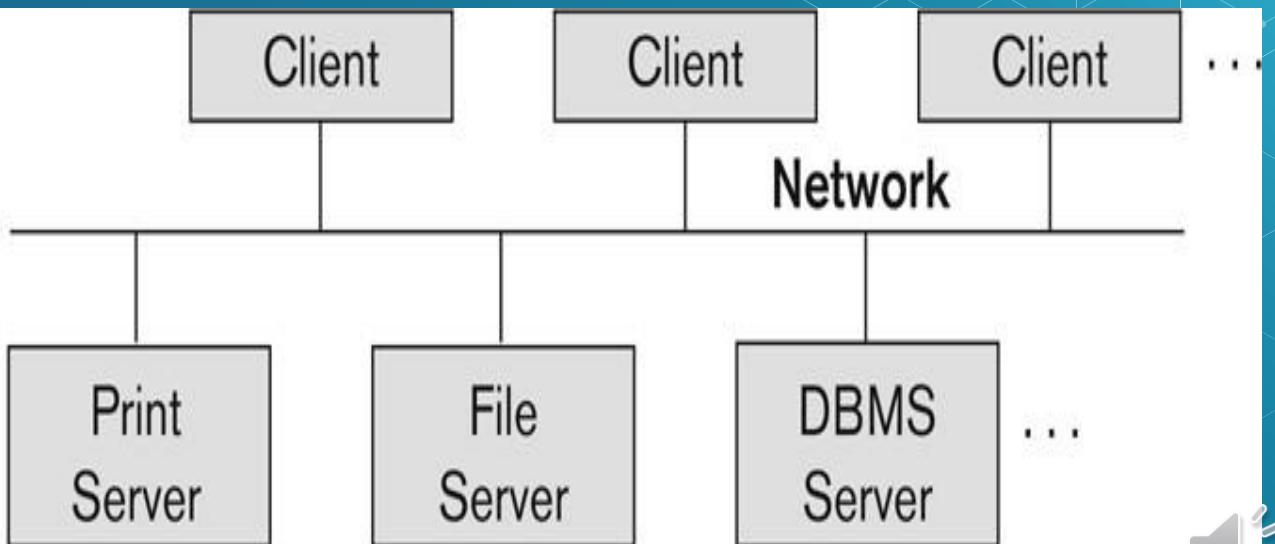
Basic 2-tier Client-Server Architectures

- ◆ Specialized Servers with Specialized functions
 - ✓ Print server
 - ✓ File server
 - ✓ DBMS server
 - ✓ Web server
 - ✓ Email server
- ◆ Clients can access the specialized servers as needed



Logical two-tier client server architecture

Figure 2.5
Logical two-tier
client/server
architecture.





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
- ◆ Applications running on clients **utilize** an Application Program Interface (**API**) to access server databases via standard interface such as:
 - ✓ ODBC: Open Database Connectivity standard
 - ✓ JDBC: for Java programming access





Two Tier Client-Server Architecture

- ◆ Client and server must **install** appropriate **client module** and **server module** **software** for ODBC or JDBC
- ◆ A client program may connect to several DBMSs; sometimes called the data sources.
- ◆ In general, **data sources** can be files or other non-DBMS software that manages data.
- ◆ See Chapter 10 for details on Database Programming





Three Tier Client-Server Architecture

- ◆ Common for **Web applications**
- ◆ **Intermediate Layer** called **Application Server or Web Server**:
 - ✓ Stores the web connectivity software and the business logic part of the application used to access the corresponding **data** from the **database server**
 - ✓ Acts like a **conduit** for sending partially processed data between **the database server and the client**.





Three Tier Client-Server Architecture

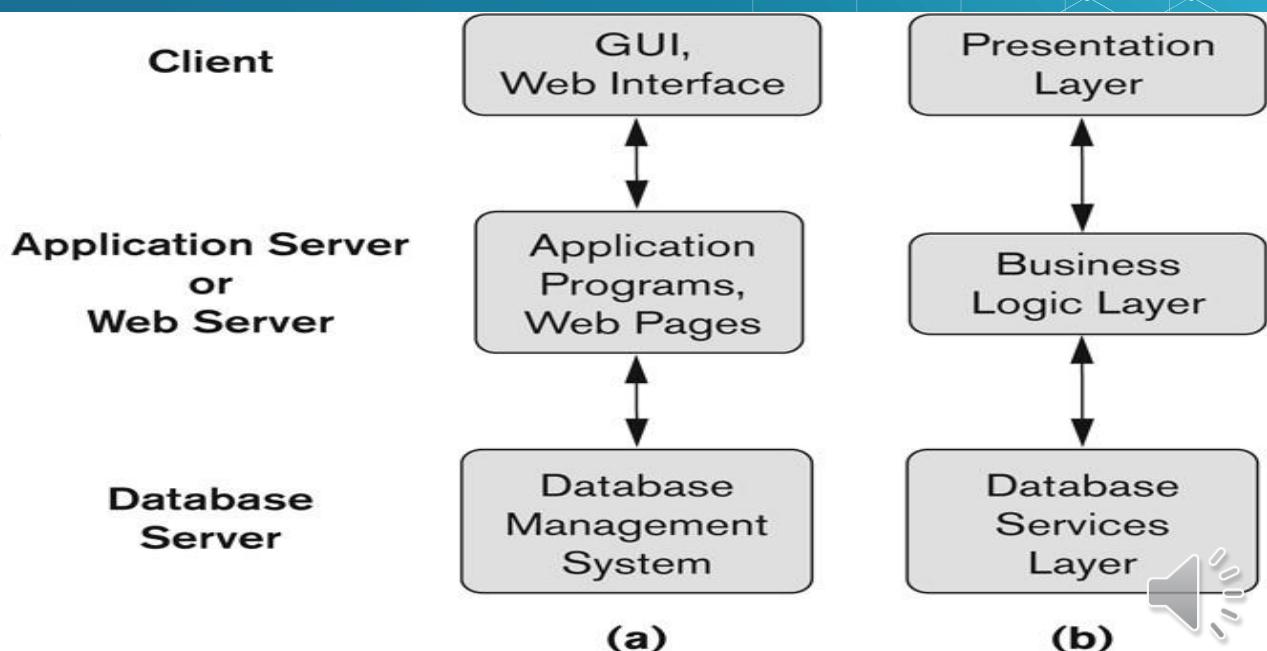
- ◆ Three-tier Architecture Can **Enhance Security**:
 - ✓ Database server only accessible via **middle tier**
 - ✓ Clients **cannot directly** access database server
 - ✓ Clients contain user **interfaces** and **Web browsers**
 - ✓ The client is typically a **PC** or a **mobile device connected to the Web**



Three Tier Client-Server Architecture

Figure 2.7

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



THANKS!

ANY QUESTIONS?

Mona_abbass12@hotmail.com

