Read Chapter 2

Database System Concepts and Architecture

Dr. Hela Chamkhia

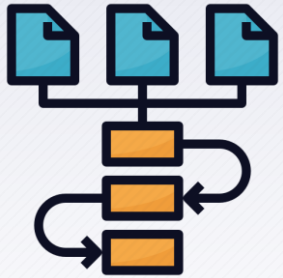
Outlines

1. **Data Model, Schemas, and instances**
2. **Three-Schema Architecture and Data Independence**
3. **Centralized and Client/Server Architectures for DBMSs**

1

Data Model, Schemas, and instances

Data Models



- **Data Model:**
  - A set of concepts to **describe the *structure* (*data types, relationships, and constraints that apply to the data)*** 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 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(**record formats, record orderings, and access paths**.). These are usually specified in an ad-hoc manner through DBMS design and administration manuals

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 (is specified during database design and is not expected to change frequently)
  - 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.

Example of a Database Schema

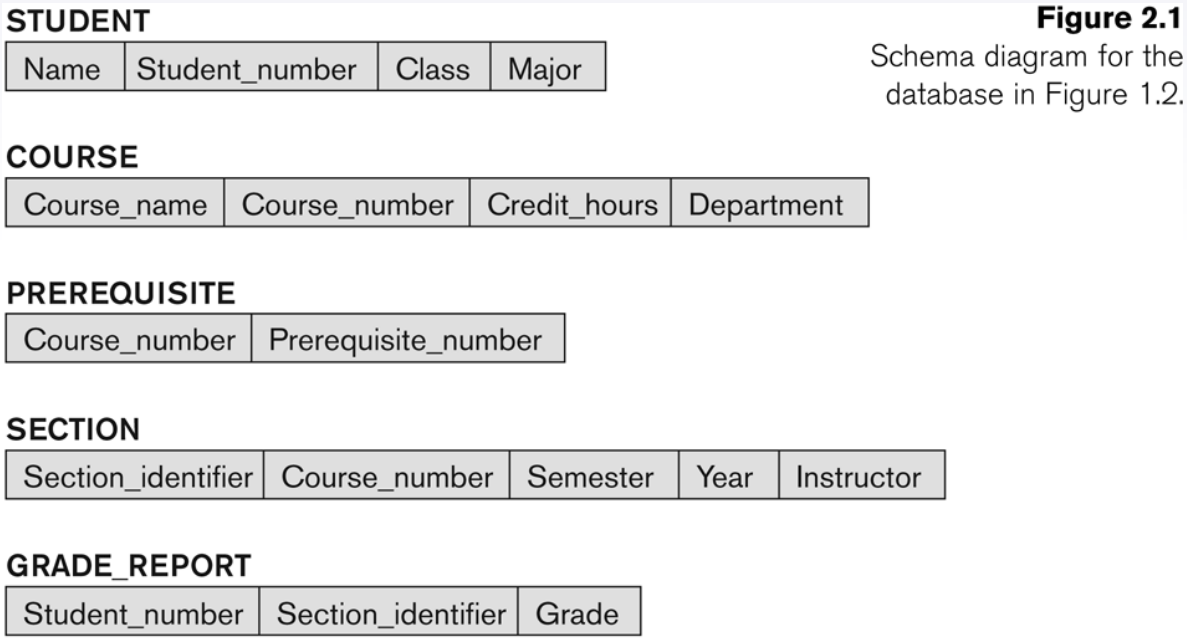
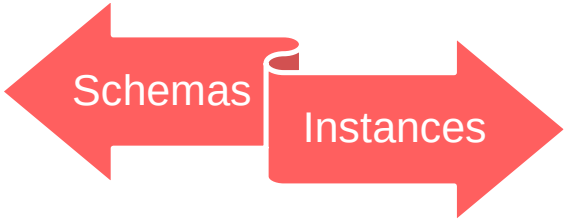


Figure 2.1
Schema diagram for the database in Figure 1.2.

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 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.

2

Three-Schema Architecture and Data Independence

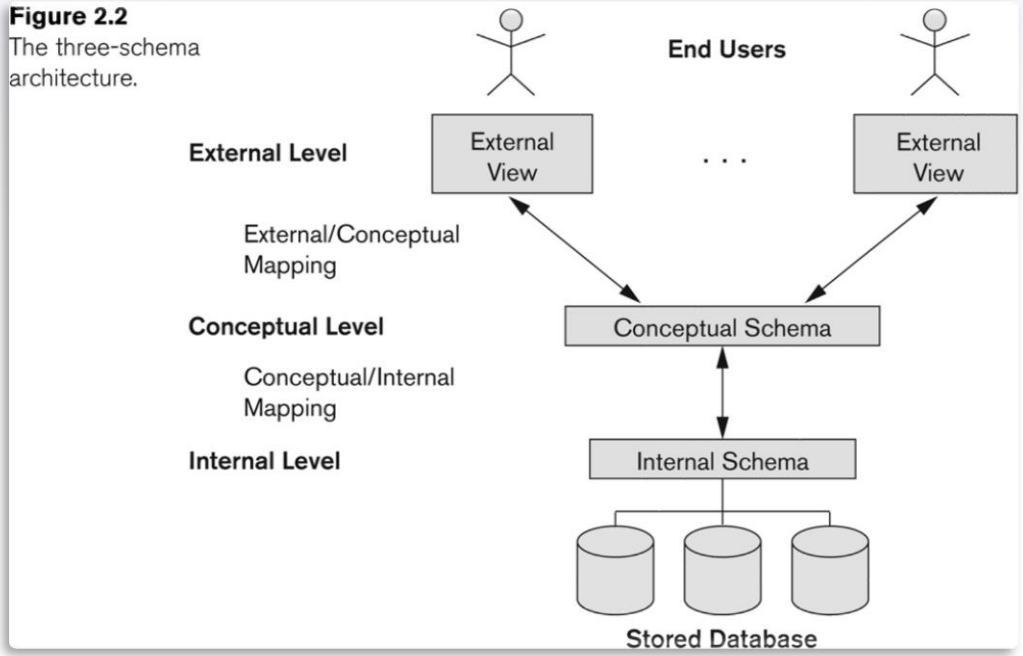
Three-Schema Architecture

- 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 (continued)

- 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** data model.
  - **External schemas** at the external level to describe the various user views.

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

- **Data independence** can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
- We can define two types of data independence:
  - **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. Only the view definition and the mappings need to be changed in a DBMS (ex.Changes to constraints)
  - **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.(For example, providing an access path to improve retrieval speed of SECTION records by semester and year should not require a query such as list all sections offered in fall 2008 to be changed,)

3

Centralized and Client/Server Architectures for DBMSs

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

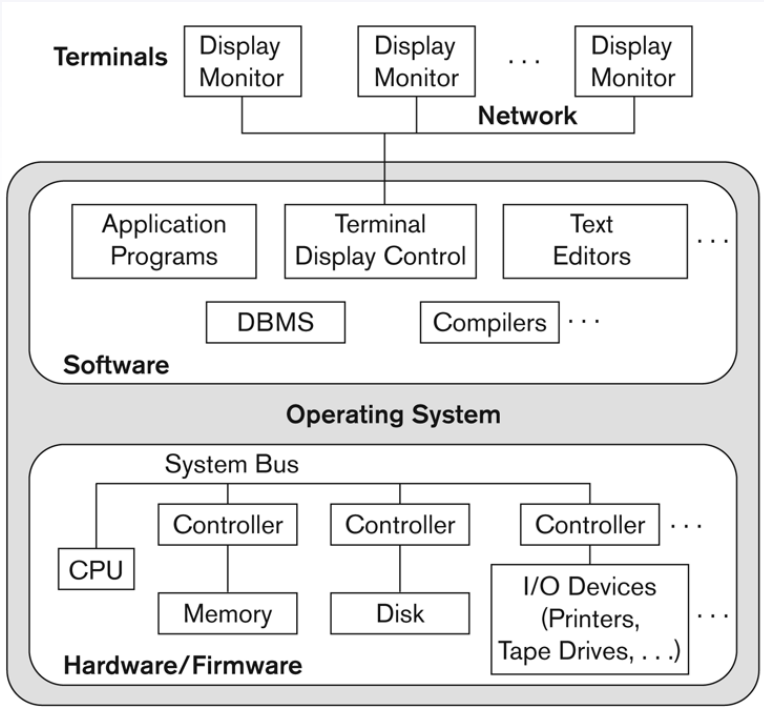


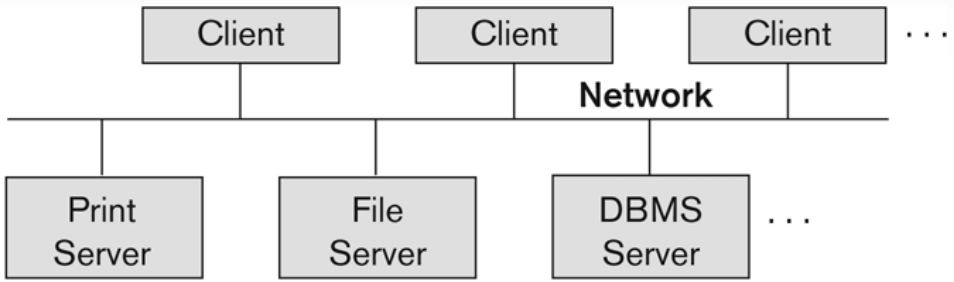
Figure 2.4
A physical centralized architecture.

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.)

Client

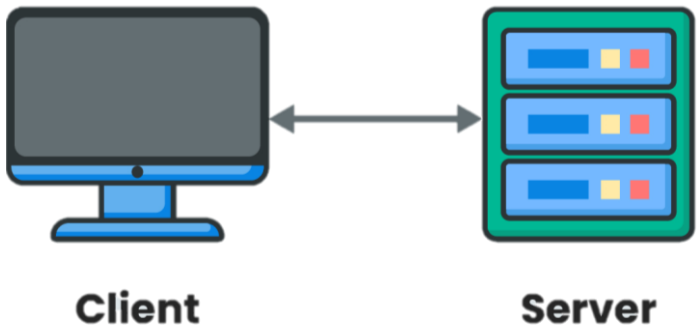
DBMS Server

- 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

Server

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.



Three Tier Client-Server Architecture

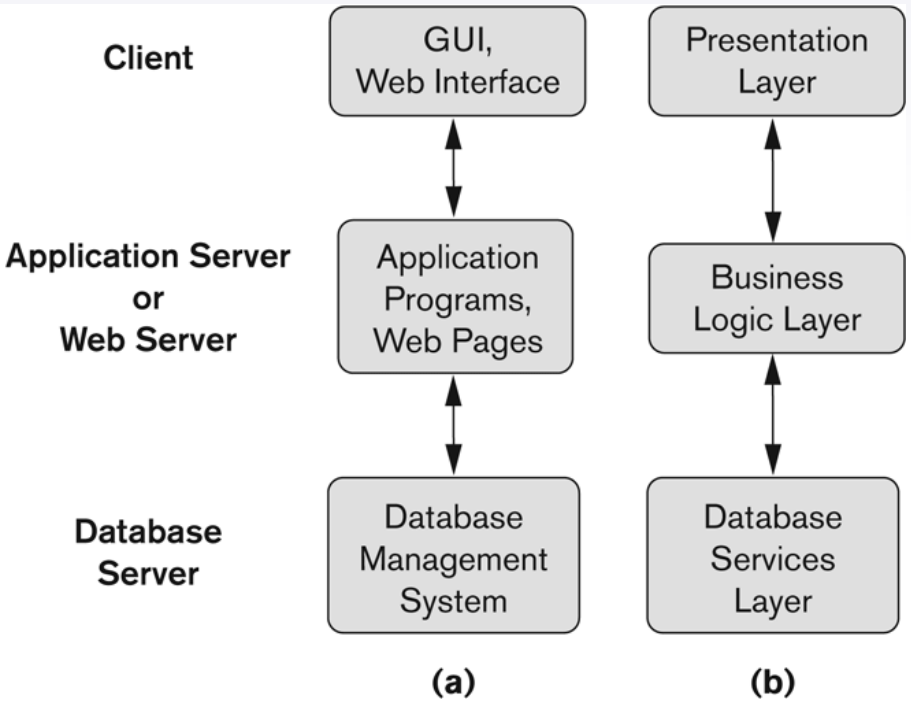
- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
  - Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
  - Acts like a conduit for sending partially processed data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



Classification of DBMSs

- **Based on the data model used**
  - Legacy: Network, Hierarchical.
  - Currently Used: **Relational**, Object-oriented, Object-relational
  - Recent Technologies: Key-value storage systems, NOSQL systems: document based, column-based, graph-based and key-value based.
  - **Native XML**(eXtended Markup Language) DBMSs, which is a **tree-structured data model.**
- **Other classifications**
  - **Single-user** (one user at a time, typically used with personal computers) vs. **multi-user** (most DBMSs).
  - **Centralized** (uses a single computer with one database) vs. **distributed** (multiple computers, multiple DBs)

Classification of DBMSs

Cost considerations for DBMSs

- Cost Range: from free open-source systems to configurations costing millions of dollars
- Examples of free relational DBMSs: MySQL, Postgre SQL, others
- Commercial DBMS offer **additional specialized modules**, e.g. time-series module, spatial data module, document module, XML module
  - These offer additional specialized functionality when purchased separately
  - Sometimes called cartridges (e.g., in Oracle) or blades
- Different licensing options: site license, maximum number of concurrent users (seat license), single user, etc.

Review questions

- Define the following terms: *data model*, *database schema*, *database state*, *client/server architecture*, *three-tier architecture*.
- What is the difference between a database schema and a database state?
- Describe the three-schema architecture. Why do we need mappings among schema levels?

Summary

- In this chapter we introduced the main concepts used in database systems. We defined a data model and we distinguished three main categories:
  - High-level or conceptual data models (based on entities and relationships)
  - Low-level or physical data models
  - Representational or implementation data models (record-based, object oriented)
- We distinguished the schema, or description of a database, from the database itself.
- The schema does not change very often, whereas the database state changes every time data is inserted, deleted, or modified. Then we described the three-schema DBMS architecture, which allows three schema levels:
  - An internal schema describes the physical storage structure of the database.
  - A conceptual schema is a high-level description of the whole database.
  - External schemas describe the views of different user groups..

Summary

- A DBMS that cleanly separates the three levels must have mappings among the schemas to transform requests and query results from one level to the next. Most DBMSs do not separate the three levels completely. We used the three-schema architecture to define the concepts of logical and physical data independence.
- We continued with an overview of the two-tier and three-tier architectures for database applications.
- Finally, we classified DBMSs according to several criteria: data model, number of users, number of sites, types of access paths, and cost. We discussed the availability of DBMSs and additional modules—from no cost in the form of open source software to configurations that annually cost millions to maintain. We also pointed out the variety of licensing arrangements for DBMS and related products.
- The main classification of DBMSs is based on the data model. We briefly discussed the main data models used in current commercial DBMSs.