CHAPTER 2
Database System Concepts (Data Models)

Note: Slides, content, web links and end chapter questions are prepared from Pearson textbook (Elmasri & Navathe), and other Internet resources.

Topic of Discussion

A. Data Models and Their Categories, History of Data Models
B. Schemas, Instances, and States
C. Three-Schema Architecture
D. DBMS Languages and Interfaces
E. Database System Utilities and Tools
F. Centralized and Client-Server Architectures
G. Classification of DBMSs

A. 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

- **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 in STUDENT database, update_inventory in PURCHASE database)

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. Conceptual data models use concepts such as entities, attributes, and relationships.
- **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.
- **Implementation (representational) data models:**
  - Provide concepts that fall between the above two, used by many commercial DBMS implementations (Ex: 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.

B. Database Schema

- **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.

Example of a Database Schema

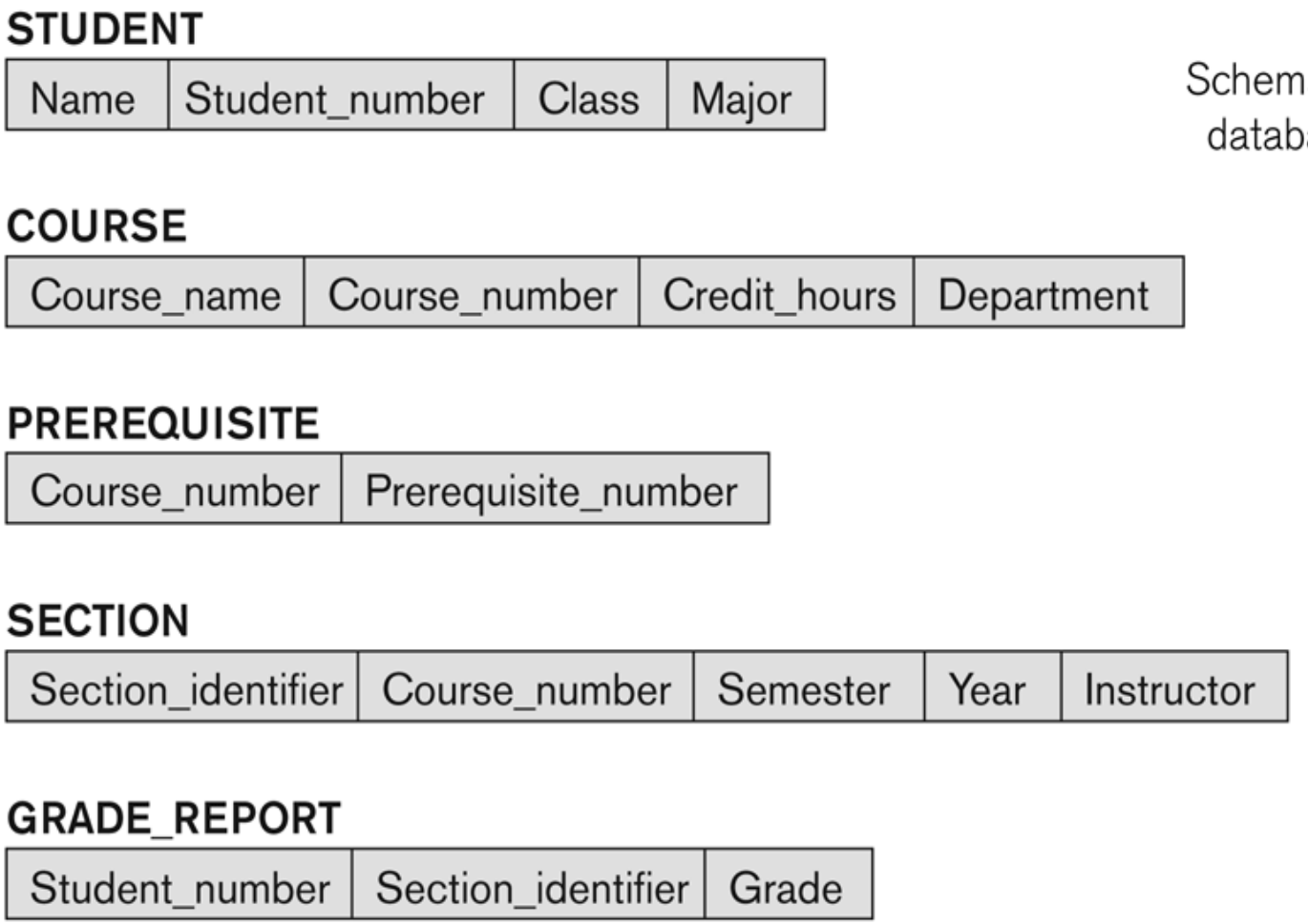


Figure 2.1
Schema diagram for the database in Figure 1.2.

Database Schema v/s Database State

- **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 very much in use and it also applied to individual database components, e.g. *record instance, table instance, entity instance*
  - **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.

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.

Database Schema v/s Database State

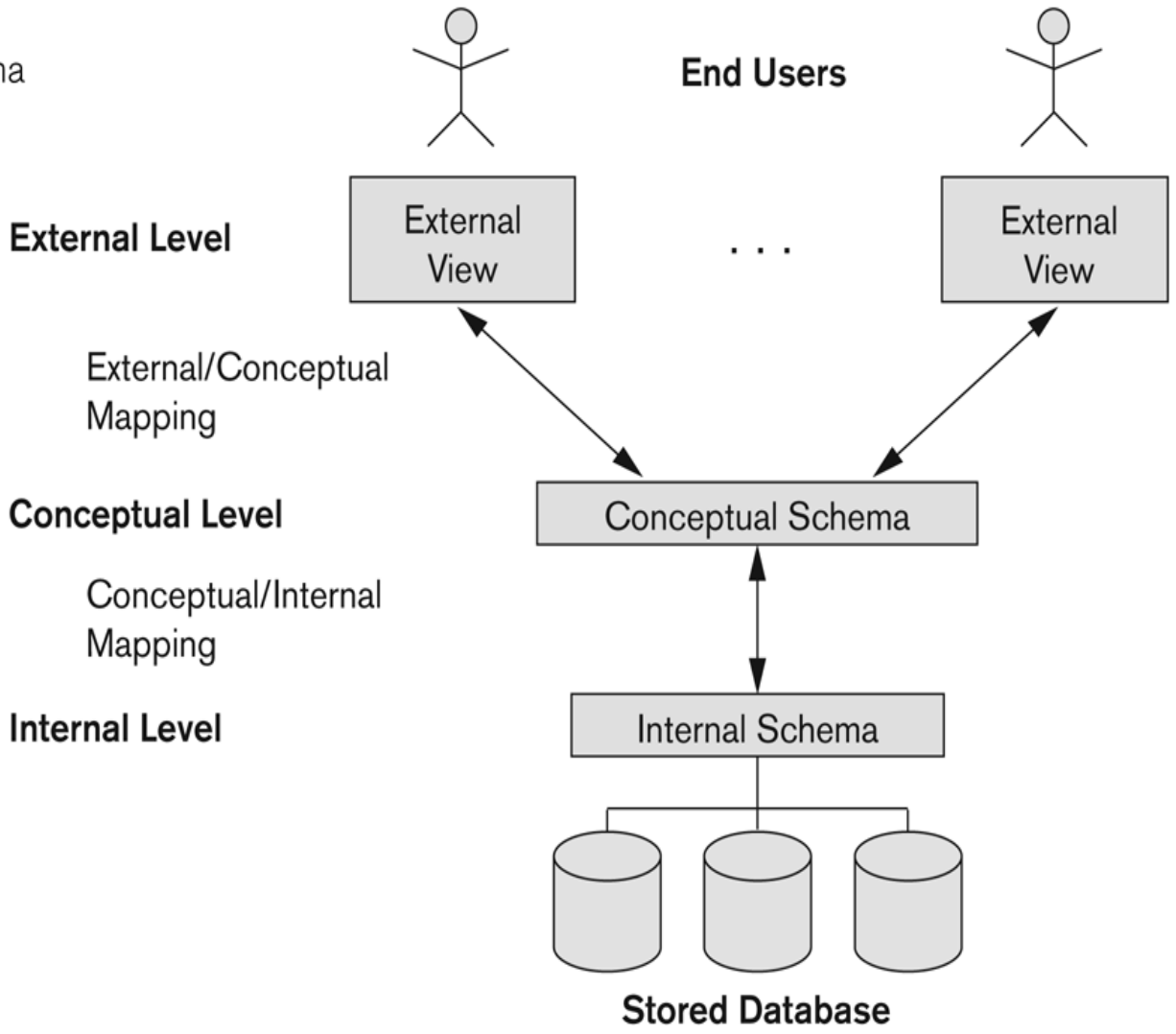
- **Distinction**
  - The ***database schema*** changes very infrequently. Once created than change only any business rule change.
  - The ***database state*** changes every time the database is updated.
- **Schema** is also called **intension**.
- **State** is also called **extension**.

C. 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.
  - **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.



D. 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. (will see this in detail in Ch-10)
  - A library of functions can also be provided to access the DBMS from a programming language.
  - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

DBMS Languages

- **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, SSMS in SQL Server)
- 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

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach**: e.g. JDBC for Java, ODBC for other programming languages as API's (application programming interfaces)
  - **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language. MS SQL Server has Transect-SQL (T-SQL) language incorporates SQL and its data types as integral components
  - **Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

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
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

User-Friendly DBMS Interfaces

- Natural language: free text as a query
- 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

Typical DBMS Component Modules

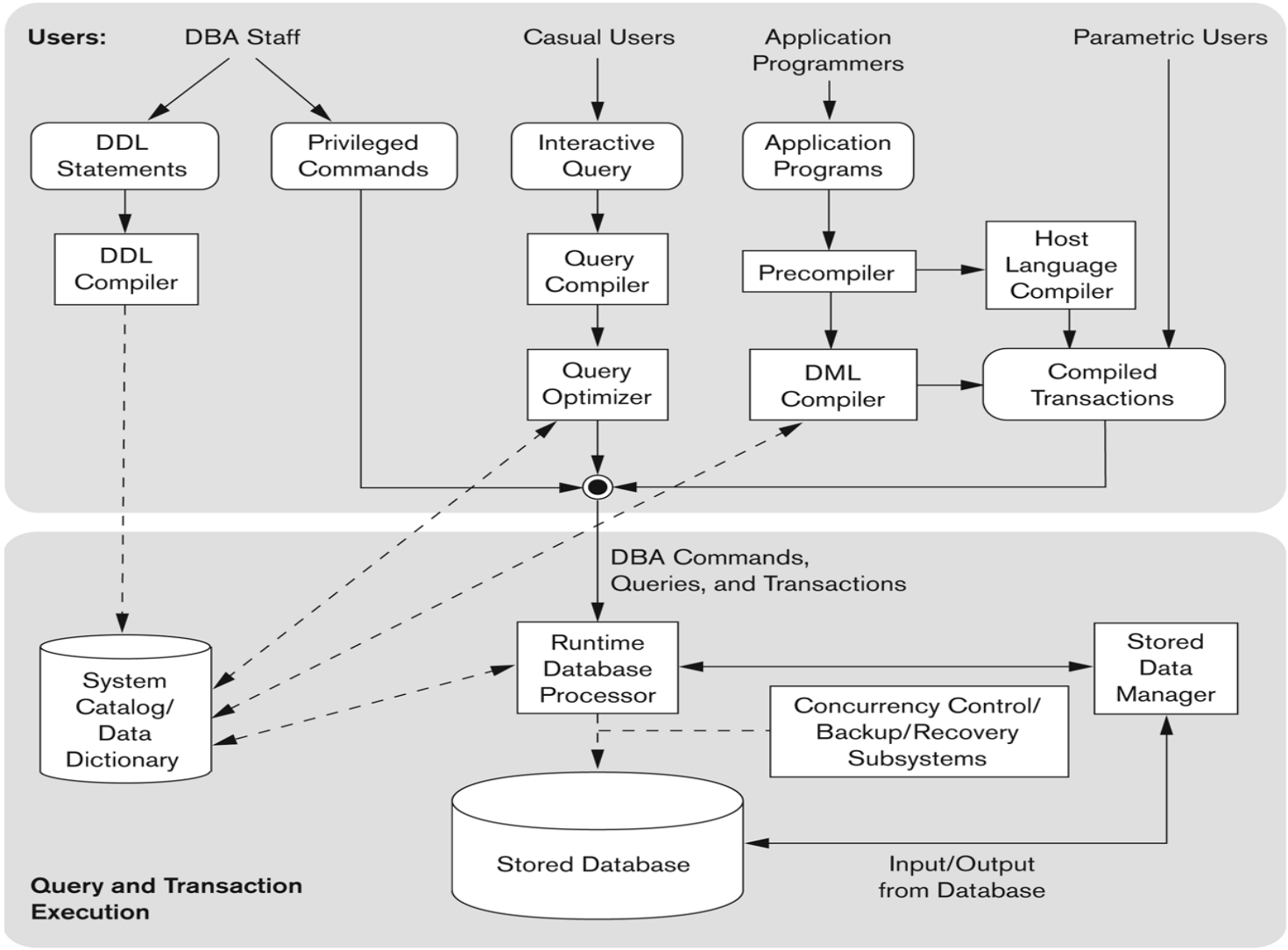


Figure 2.3
Component modules of a DBMS and their interactions.

E. 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.
  - Performance monitoring utilities.
  - Report generation 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 design decisions, 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.
- Application Development Environments and CASE (computer-aided software engineering) tools:
- Examples:
  - PowerBuilder (Sybase), JBuilder (Borland)
  - JDeveloper 10G (Oracle)

F. 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

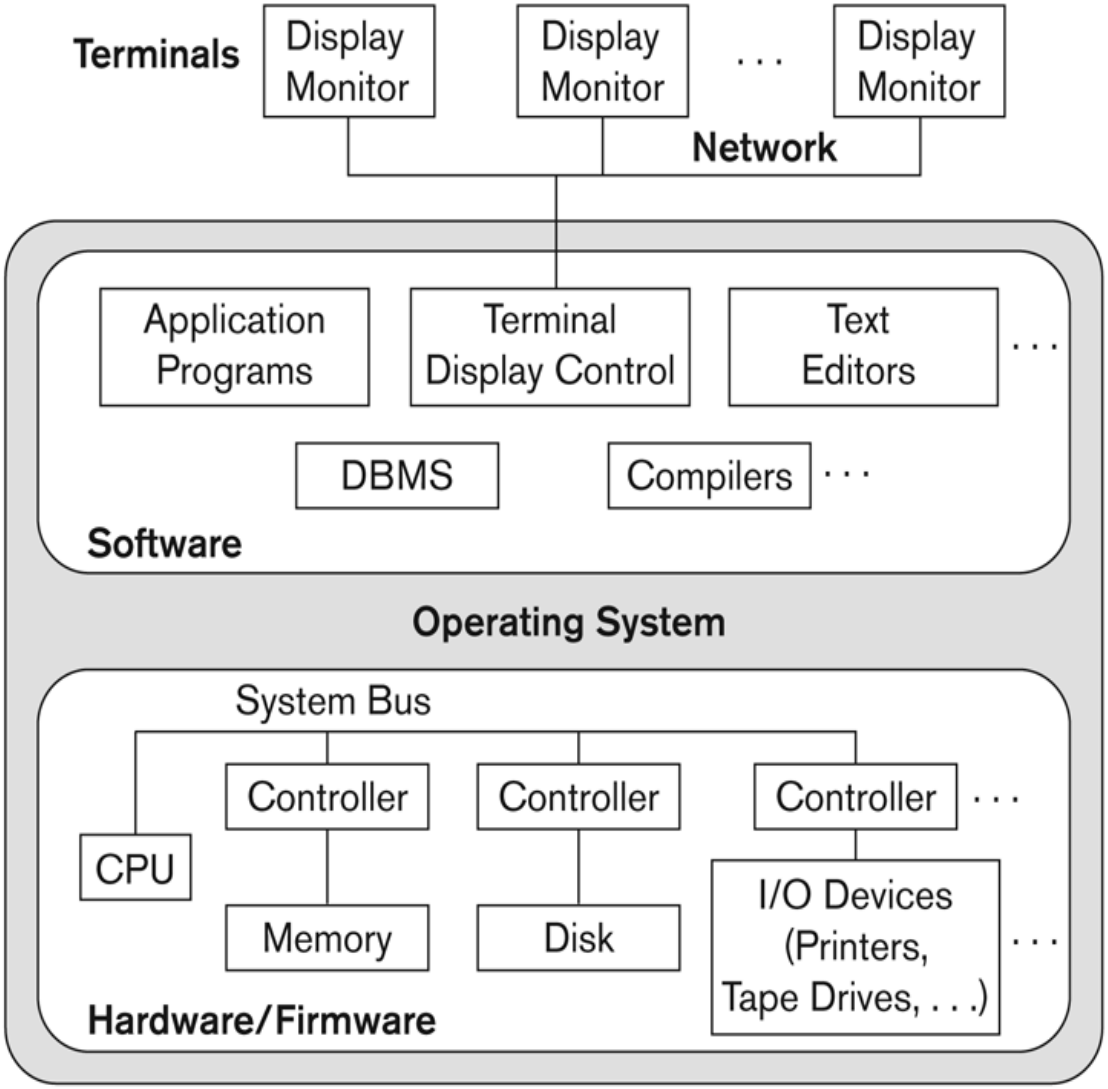


Figure 2.4
A physical centralized architecture.

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

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.)

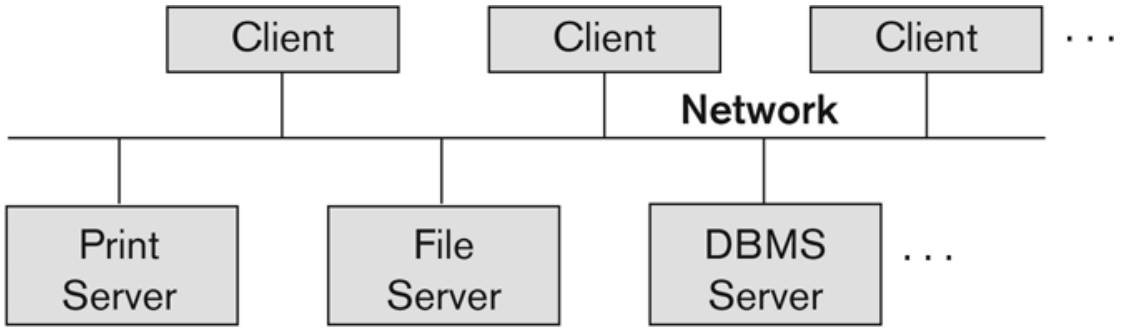


Figure 2.5
Logical two-tier client/server architecture.

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

2 Tier Client-Server Architecture

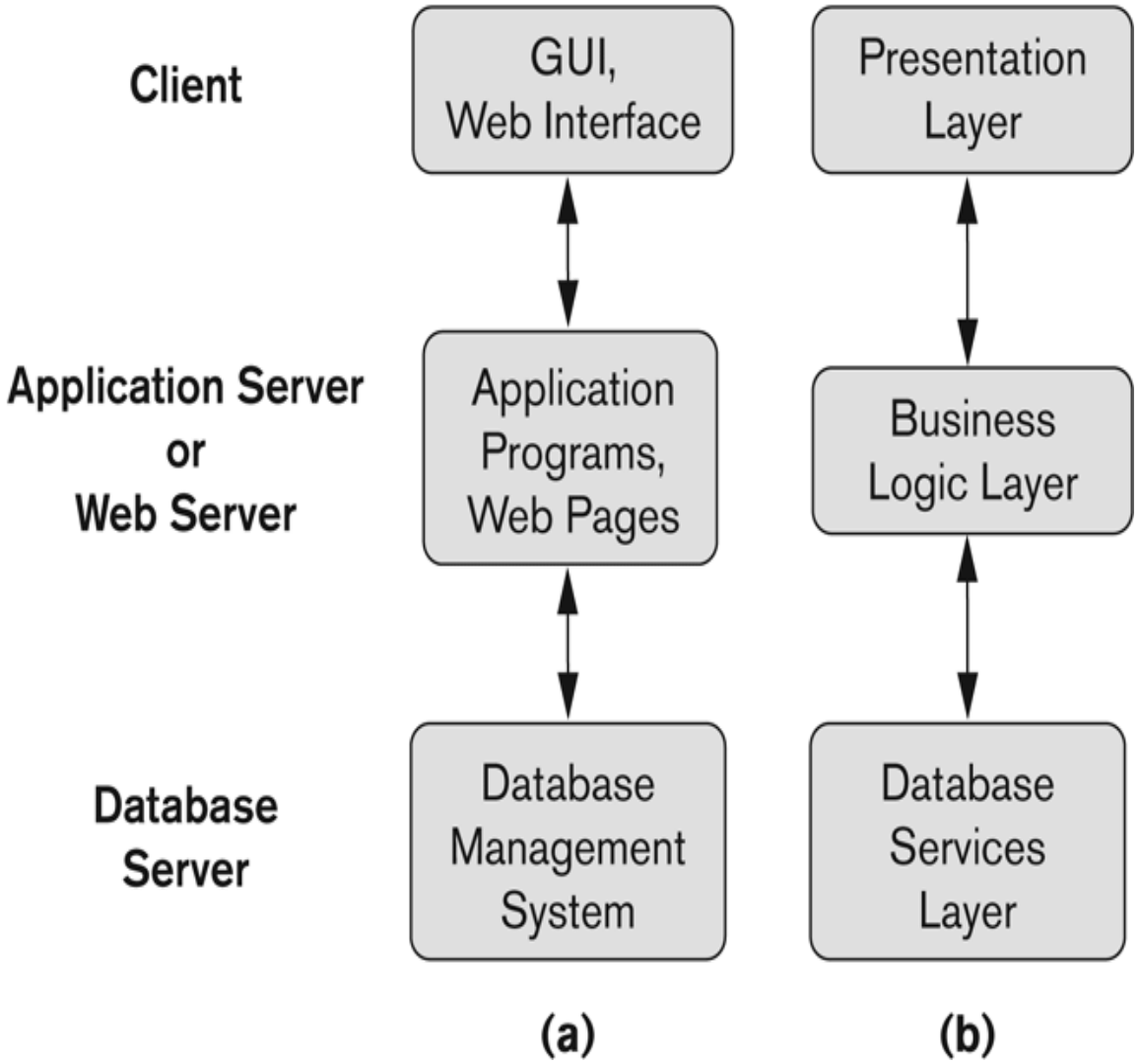
- Client and server must install appropriate client module and server module software for ODBC or JDBC
- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data. (We will see chapter 10 for details on Database Programming)

3 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 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

3 tier client-server architecture

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



G. Classification of DBMSs

- 1) 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 DBMSs.
- 2) User base classifications
  - Single-user (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

- 3) Variations of Distributed DBMSs (DDBMs)
  - Homogeneous DDBMS:
    - use of same DBMS software at all sites
  - Heterogeneous DDBMS :
    - use of different DBMS software at all/some sites
  - Federated or Multi database Systems
    - Participating Databases are loosely coupled with high degree of autonomy.
  - Distributed Database Systems have now come to be known as client-server based database systems because:
    - They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

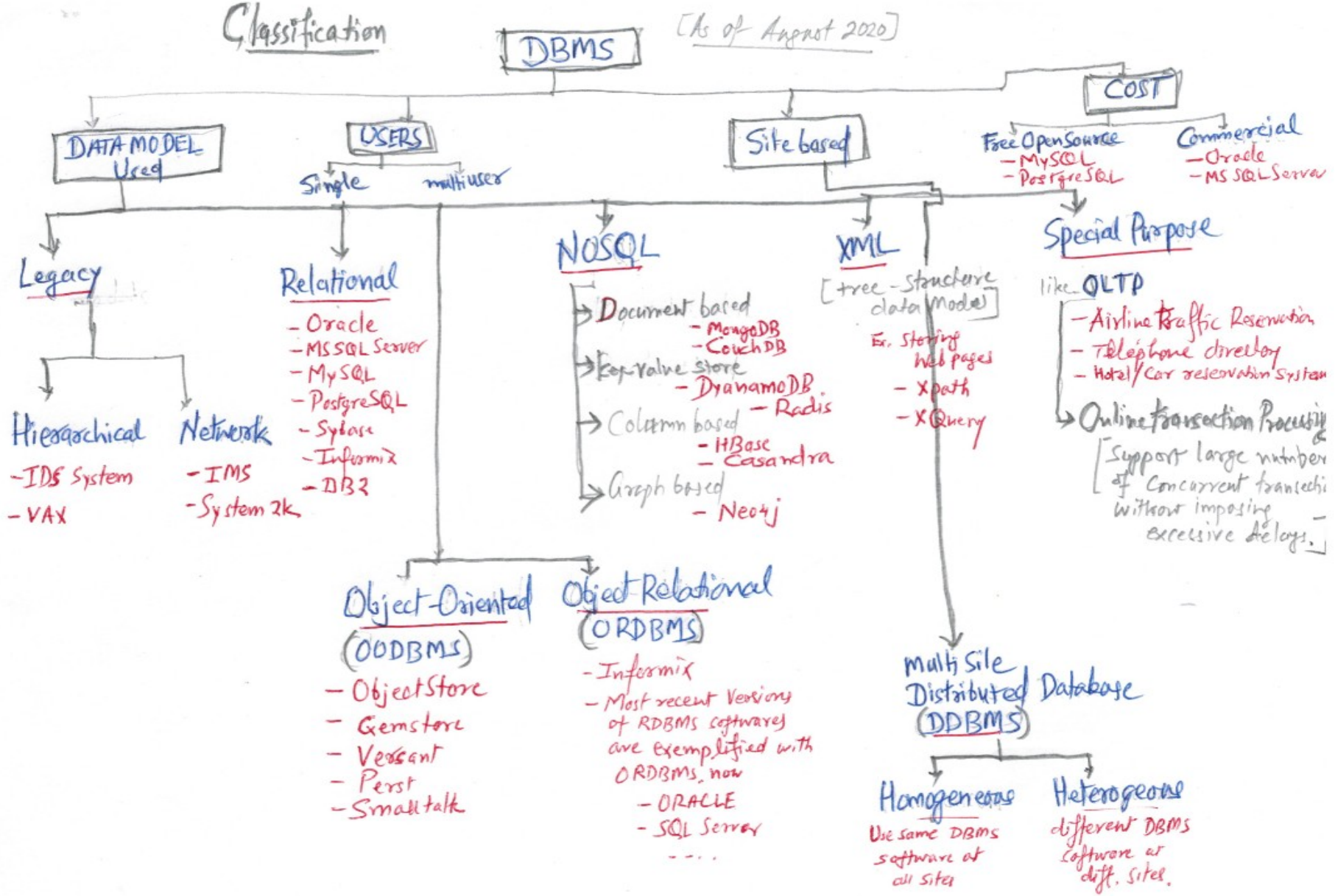
Classification of DBMSs

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

Classification of DBMSs

- 5) Other Considerations
- Type of access paths within the database system
  - E.g.- inverted indexing based (ADABAS is one such system). Fully indexed databases provide access by any keyword (used in search engines)
- General Purpose vs. Special Purpose
  - E.g.- Airline Reservation systems or many others- reservation systems for hotels/cars etc. Are special purpose OLTP (Online Transaction Processing Systems)

Classification of DBMSs



End Chapter Questions

- 2.1. Define the following terms: data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, DDL, DML, SDL, VDL, query language, host language, data sublanguage, database utility, catalog, client/server architecture, three-tier architecture, and n-tier architecture.
- 2.2. Discuss the main categories of data models. What are the basic differences among the relational model, the object model, and the XML model?
- 2.3. What is the difference between a database schema and a database state?

End Chapter Questions

- 2.6. What is the difference between procedural and nonprocedural DMLs?
- 2.7. Discuss the different types of user-friendly interfaces and the types of users who typically use each.
- 2.8. With what other computer system software does a DBMS interact?
- 2.12. Explain the classification of DBMSs. Draw classification chart.