

# 22CS510 - Database Management Systems

**Dr. Manimala.S**

Associate Professor

Dept. of CS & E, SJCE, Mysuru

# COURSE OBJECTIVES:

Sl.No.	Course Objectives
1	Introduce the basic concepts of database systems.
2	Learn to design entity relationship and relational models for given user requirements.
3	Formulate the relational algebraic equations and SQL statements.
4	Apply database design theory and normalization techniques.
5	Learn transaction processing techniques.

# Course Outcomes

After completing this course, students should be able to :

CO #	Course Outcomes	Highest Level of Cognitive Domain
CO1	Describe the fundamental concepts of database system environment.	L2
CO2	Design a database schema for a given problem specifications and user-requirements.	L3
CO3	Use SQL language to create, populate, maintain, and query a database.	L3
CO4	Design a simple database system applying normal forms.	L3
CO5	Design and build a simple database system in line with transaction properties.	L3

# Unit I

**Introduction to Database Concepts and Architecture** . Introduction; An example; Characteristics of Database approach; Database users, Advantages of using DBMS approach, Data models, schema and instances, Three schema architecture and data independence; Database languages and interfaces; The database system environment; Centralized and client-server architectures; Classification of DBMS.

# Unit II

**Data Modeling Using the Entity-Relationship (ER) Model :** Using High-Level Conceptual Data Models for Database Design, A Sample Database Application, Entity Types, Entity Sets, Attributes, and Keys, Relationship Types, Relationship Sets, Roles, and Structural Constraints, Weak Entity Types, Refining the ER Design for the COMPANY Database, ER Diagrams, Naming Conventions, and Design Issues, Relationship Types of Degree Higher than Two, Relational Database Design Using ER to-Relational Mapping

**The Relational Data Model and Relational Database Constraints :** Relational Model Concepts, Relational Model Constraints and Relational Database Schemas, Update Operations, Transactions, and Dealing with Constraint Violations

# Unit III

**The Relational Algebra :** Unary Relational Operations: SELECT and PROJECT, Relational Algebra Operations from Set Theory, Binary Relational Operations: JOIN and DIVISION, Additional Relational Operations : Generalized Projection, Aggregate Functions and Grouping, OUTER JOIN Operations, Examples of Queries in Relational Algebra

**Basic SQL:** SQL Data Definition and Data Types, Specifying Constraints in SQL, Basic Retrieval Queries in SQL, INSERT, DELETE, and UPDATE Statements in SQL, More Complex SQL retrieval Queries, Specifying Constraints as Assertions and Actions as Triggers, Views (Virtual Tables) in SQL

# Unit IV

**Database Design Theory and Normalization:**  
Informal Design Guidelines for Relation Schemas; Functional Dependencies; Normal Forms Based on Primary Keys; General Definitions of Second and Third Normal Forms; Boyce-Codd Normal Form, Further Topics in Functional Dependencies: Inference Rules, Equivalence, and Minimal Cover, Properties of Relational Decompositions,

# Unit V

## Transaction Processing, Concurrency Control, and Recovery :

Introduction to Transaction Processing, Transactions, Database Items, Read and Write Operations and DBMS Buffers, Why Concurrency Control is Needed, Why Recovery Is Needed, Desirable Properties of Transactions, Two-Phase Locking Techniques for Concurrency Control,



# References

## Text Books:

1. Elmasri and Navathe: Fundamentals of Database Systems, 7<sup>th</sup> Edition, Pearson Education, 2016.
2. Raghu Ramakrishnan and Johannes Gehrke: Database Management Systems, 3rd Edition, McGraw-Hill, 2015.

## Reference Books:

1. Silberschatz, Korth and Sudharshan: Data base System Concepts, 6th Edition, Mc-GrawHill, 2016.
2. C.J. Date, A. Kannan, S. Swamynatham: An Introduction to Database Systems, 8th Edition, Pearson Education, 2016.

## Web Resources:

1. <http://nptel.ac.in/courses/106106093/>
2. <https://nptel.ac.in/courses/106/104/106104135/>

# ***Database Management Systems Laboratory (22CS57L)***

<b>CO#</b>	<b>Course Outcomes</b>	<b>CD</b>
CO1	Implement a database schema for a given problem specifications and user-requirements.	L4
CO2	Use SQL language to create, populate, maintain, and query a database.	L4
CO3	Apply normalization theory to validate and revise the logical database design.	L3

# Course Content

Consider a structure named Student with attributes as SID, NAME, BRANCH, SEMESTER, ADDRESS.

Write a program in C/C++/ and perform the following operations using the concept of files.

- a. Insert a new student
- b. Modify the address of the student based on SID
- c. Delete a student
- d. List all the students
- e. List all the students of CSE branch.
- f. List all the students of CSE branch and reside in Kuvempunagar.

Create a table for the structure Student with attributes as SID, NAME, BRANCH, SEMESTER, ADDRESS, PHONE, EMAIL, Insert atleast 10 tuples and perform the following operations using SQL.

- a. Insert a new student
- b. Modify the address of the student based on SID
- c. Delete a student
- d. List all the students
- e. List all the students of CSE branch.
- f. List all the students of CSE branch and reside in Kuvempunagar

## **Data Definition Language (DDL) commands in RDBMS**

Consider the database schemas given below.  
Write ER diagram and schema diagram. The primary keys are underlined and the data types are specified.  
Create tables for the following schema listed below by properly specifying the primary keys and foreign keys.

Enter at least five tuples for each relation.

Altering tables,

Adding and Dropping different types of constraints.

Also adding and dropping fields in to the relational schemas of the listed problems.

Delete, Update operations

### A. Sailors database

**SAILORS** (sid, sname, rating, age)

**BOAT** (bid, bname, color)

**RSERVERS** (sid, bid, date)

### B. Insurance database

**PERSON** (driver id#: string, name: string, address: string)

**CAR** (regno: string, model: string, year: int)

**ACCIDENT** (report \_number: int, acc\_date: date, location: string)

**OWNS** (driver id#: string, regno: string)

**PARTICIPATED** (driver id#: string, regno: string, report \_number: int, damage\_amount: int)

### C. Order processing database

**Customer** (Cust#: int, cname: string, city: string)

**Order** (order#: int, odate: date, cust#: int, order-amt: int)

**Order-item** (order#: int, Item#: int, qty: int)

**Item** (item#: int, unitprice: int)

**Shipment** (order#: int, warehouse#: int, ship-date: date)

**Warehouse** (warehouse#: int, city: string)

## **Views and Triggers**

Views: creation and manipulating content.

Triggers: creation and execution of database triggers on every insert, delete and update operation.

## **Text Books:**

1. Elmasri and Navathe: Fundamentals of Database Systems, 7<sup>th</sup> Edition, Pearson Education, 2016.
2. Raghu Ramakrishnan and Johannes Gehrke: Database Management Systems, 3rd Edition, McGraw-Hill, 2015.

## **Reference Books:**

1. Silberschatz, Korth and Sudharshan: Data base System Concepts, 6th Edition, Mc-GrawHill, 2016.
2. C.J. Date, A. Kannan, S. Swamynatham: An Introduction to Database Systems, 8th Edition, Pearson Education, 2016.



# **CHAPTER 1**

## **Databases and Database Users**

# OUTLINE

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- When Not to Use Databases

# Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Biological and Genome Databases
  - Data Warehouses
  - Mobile databases
  - Real-time and Active Databases

# Recent Developments (1)

- Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:
  - Facebook
  - Twitter
  - Linked-In
- All of the above constitutes data
- Search Engines- Google, Bing, Yahoo : collect their own repository of web pages for searching

# Recent Developments (2)

- New Technologies are emerging from the so-called non-database software vendors to manage vast amounts of data generated on the web:
- Big Data storage systems involving large clusters of distributed computers (Chapter 25)
- NOSQL (Not Only SQL) systems (Chapter 24)
- A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

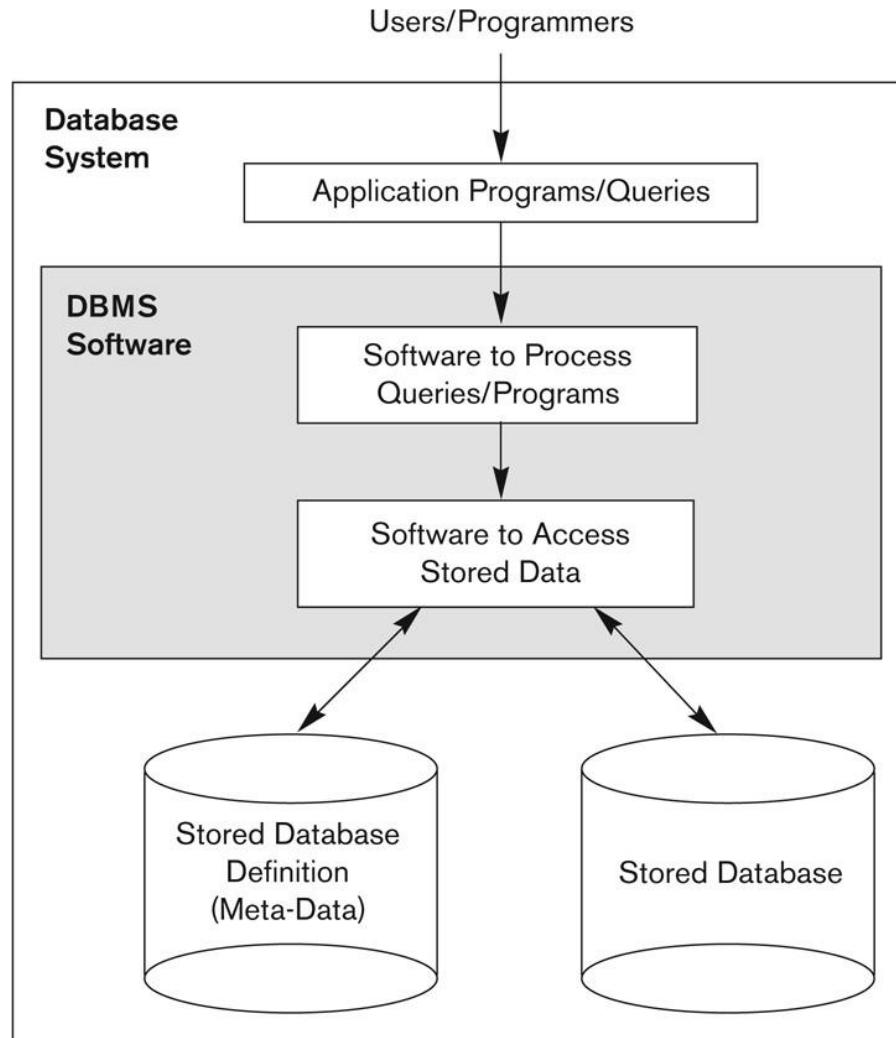
# Basic Definitions

- **Database:**
  - A collection of related data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Mini-world:**
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS):**
  - A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

# Impact of Databases and Database Technology

- **Businesses:** Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing
- **Service Industries:** Financial, Real-estate, Legal, Electronic Commerce, Small businesses
- **Education :** Resources for content and Delivery
- **More recently:** Social Networks, Environmental and Scientific Applications, Medicine and Genetics
- **Personalized Applications:** based on smart mobile devices

# Simplified database system environment



**Figure 1.1**  
A simplified database  
system environment.



# Typical DBMS Functionality

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or Load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing and Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

# Application Activities Against a Database

- Applications interact with a database by generating
  - Queries: that access different parts of data and formulate the result of a request
  - Transactions: that may read some data and “update” certain values or generate new data and store that in the database
- Applications must not allow unauthorized users to access data
- Applications must keep up with changing user requirements against the database

# Additional DBMS Functionality

- DBMS may additionally provide:
  - Protection or Security measures to prevent unauthorized access
  - “Active” processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintenance of the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

# Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example:**
  - Part of a UNIVERSITY environment.
- **Some mini-world *entities*:**
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

# Example of a Database (with a Conceptual Data Model)

- **Some mini-world *relationships*:**
  - SECTIONs *are of specific* COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have prerequisite* COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs
- **Note:** The above entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model (see Chapters 3, 4)

# Example of a simple database

## 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.

# Main Characteristics of the Database Approach

## ■ Self-describing nature of a database system:

- A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
- The description is called **meta-data**\*
- This allows the DBMS software to work with different database applications.

## ■ Insulation between programs and data:

- Called **program-data independence**.
- Allows changing data structures and storage organization without having to change the DBMS access programs.

-----

\* Some newer systems such as a few NOSQL systems need no meta-data: they store the data definition within its structure making it self describing

# Example of a simplified database catalog

## RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

**Figure 1.3**

An example of a database catalog for the database in Figure 1.2.

## COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	....
....	....	....
....	....	....
Prerequisite_number	XXXXNNNN	PREREQUISITE

*Note:* Major\_type is defined as an enumerated type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits



# Main Characteristics of the Database Approach (continued)

- **Data Abstraction:**

- A **data model** is used to hide storage details and present the users with a conceptual view of the database.
- Programs refer to the data model constructs rather than data storage details

- **Support of multiple views of the data:**

- Each user may see a different view of the database, which describes **only** the data of interest to that user.

# Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

# Database Users

- Users may be divided into
  - Those who actually use and control the database content, and those who design, develop and maintain database applications (called “Actors on the Scene”), and
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (called “Workers Behind the Scene”).

# Database Users – Actors on the Scene

- Actors on the scene
  - **Database administrators:**
    - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
  - **Database Designers:**
    - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

# Database End Users

- Actors on the scene (continued)
  - **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
    - **Casual:** access database occasionally when needed
    - **Naïve** or Parametric: they make up a large section of the end-user population.
      - They use previously well-defined functions in the form of “canned transactions” against the database.
      - Users of Mobile Apps mostly fall in this category
      - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
      - Social Media Users post and read information from websites

# Database End Users (continued)

- **Sophisticated:**

- These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
- Many use tools in the form of software packages that work closely with the stored database.

- **Stand-alone:**

- Mostly maintain personal databases using ready-to-use packaged applications.
- An example is the user of a tax program that creates its own internal database.
- Another example is a user that maintains a database of personal photos and videos.

# Database Users – Actors on the Scene (continued)

- **System Analysts and Application Developers**

This category currently accounts for a very large proportion of the IT work force.

- **System Analysts:** They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
- **Application Programmers:** Implement the specifications developed by analysts and test and debug them before deployment.
- **Business Analysts:** There is an increasing need for such people who can analyze vast amounts of business data and real-time data (“Big Data”) for better decision making related to planning, advertising, marketing etc.

# Database Users – Actors behind the Scene

- **System Designers and Implementors:** Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- **Tool Developers:** Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- **Operators and Maintenance Personnel:** They manage the actual running and maintenance of the database system hardware and software environment.



# Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
  - Sharing of data among multiple users.
- Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
- Providing persistent storage for program Objects
  - E.g., Object-oriented DBMSs make program objects persistent— see Chapter 12.
- Providing Storage Structures (e.g. indexes) for efficient Query Processing – see Chapter 17.

# Advantages of Using the Database Approach (continued)

- Providing optimization of queries for efficient processing.
- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Drawing inferences and actions from the stored data using deductive and active rules and triggers.

# Additional Implications of Using the Database Approach

- Potential for enforcing standards:
  - This is very crucial for the success of database applications in large organizations. **Standards** refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.

# Additional Implications of Using the Database Approach (continued)

- Flexibility to change data structures:
  - Database structure may evolve as new requirements are defined.
- Availability of current information:
  - Extremely important for on-line transaction systems such as shopping, airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

# When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change.
  - If access to data by multiple users is not required.
- When a DBMS may be infeasible:
  - In embedded systems where a general purpose DBMS may not fit in available storage

# When not to use a DBMS

- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)
  - If the database system is not able to handle the complexity of data because of modeling limitations (e.g., in complex genome and protein databases)
  - If the database users need special operations not supported by the DBMS (e.g., GIS and location based services).

# Chapter Summary

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- When Not to Use Databases

# **CHAPTER 2**

## **Database System Concepts and Architecture**



# Outline

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

## ■ 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. 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 (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

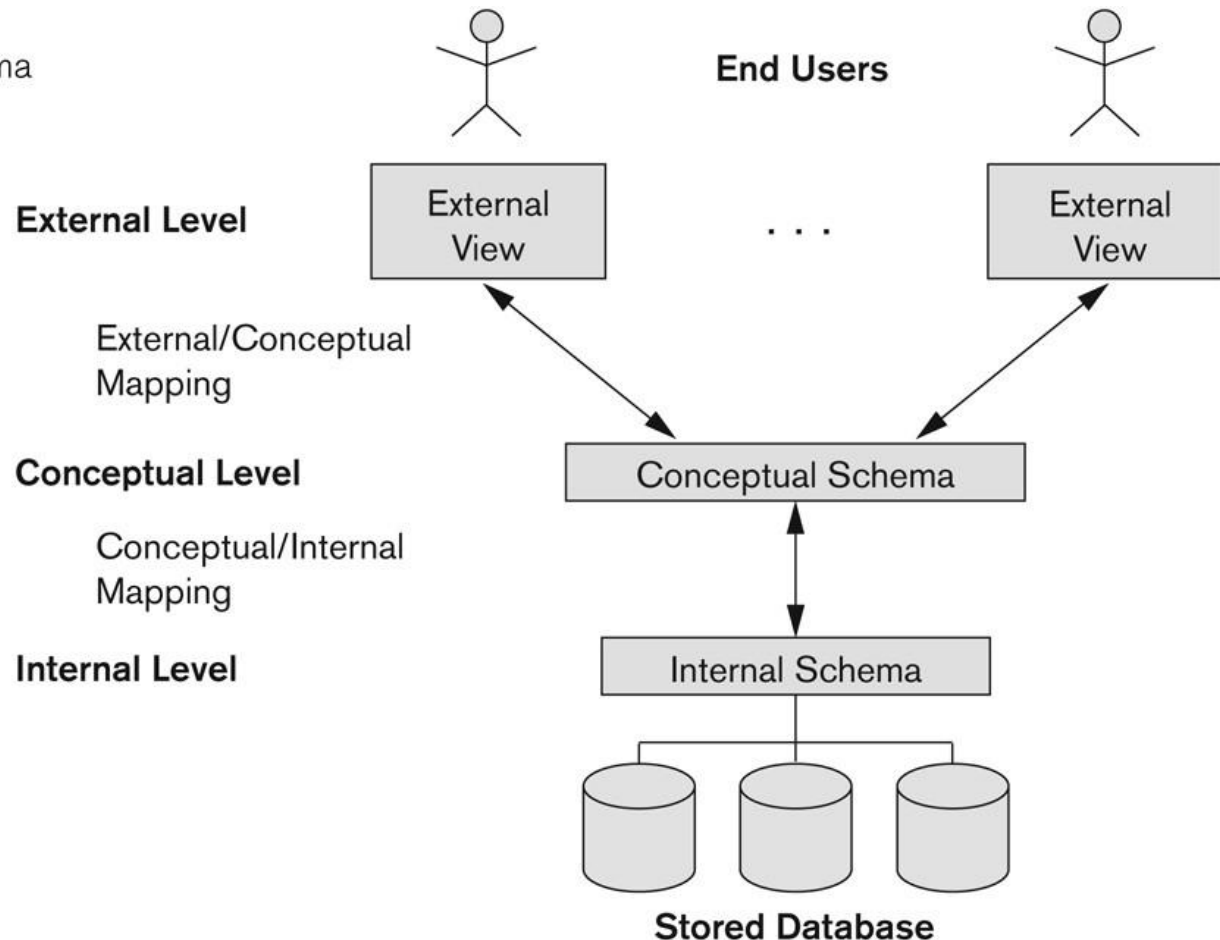
# Three-Schema Architecture

- 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.
  - **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

# 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.



# DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

# 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.
    - 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*).

# 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

# 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 (Open Database Connectivity) for other programming languages as API's (application programming interfaces)
  - **Database Programming Language Approach:** e.g. ORACLE has PL/SQL, a programming language based on 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

# Other 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



# 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.

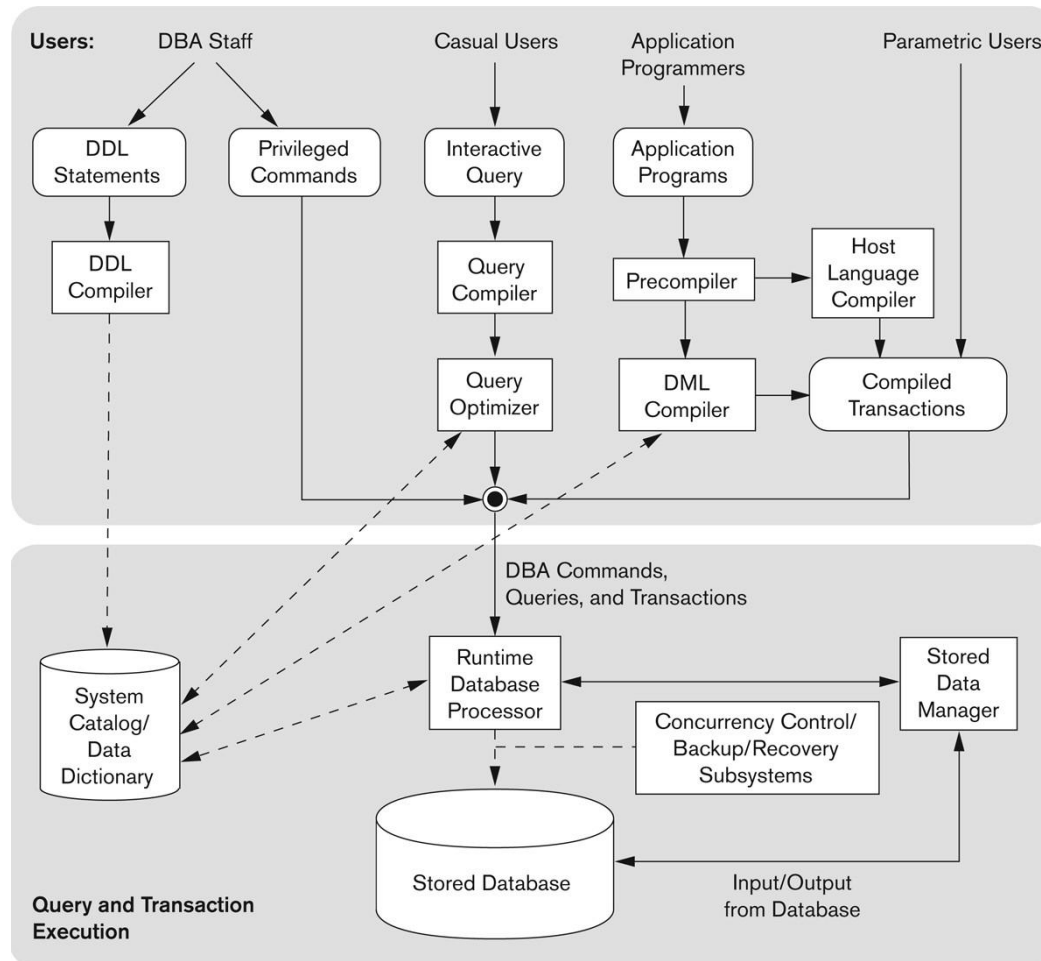
# Other Tools

- 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.

# Other Tools

- Application Development Environments and CASE (computer-aided software engineering) tools:
- Examples:
  - PowerBuilder (Sybase)
  - JBuilder (Borland)
  - JDeveloper 10G (Oracle)

# Typical DBMS Component Modules



**Figure 2.3**

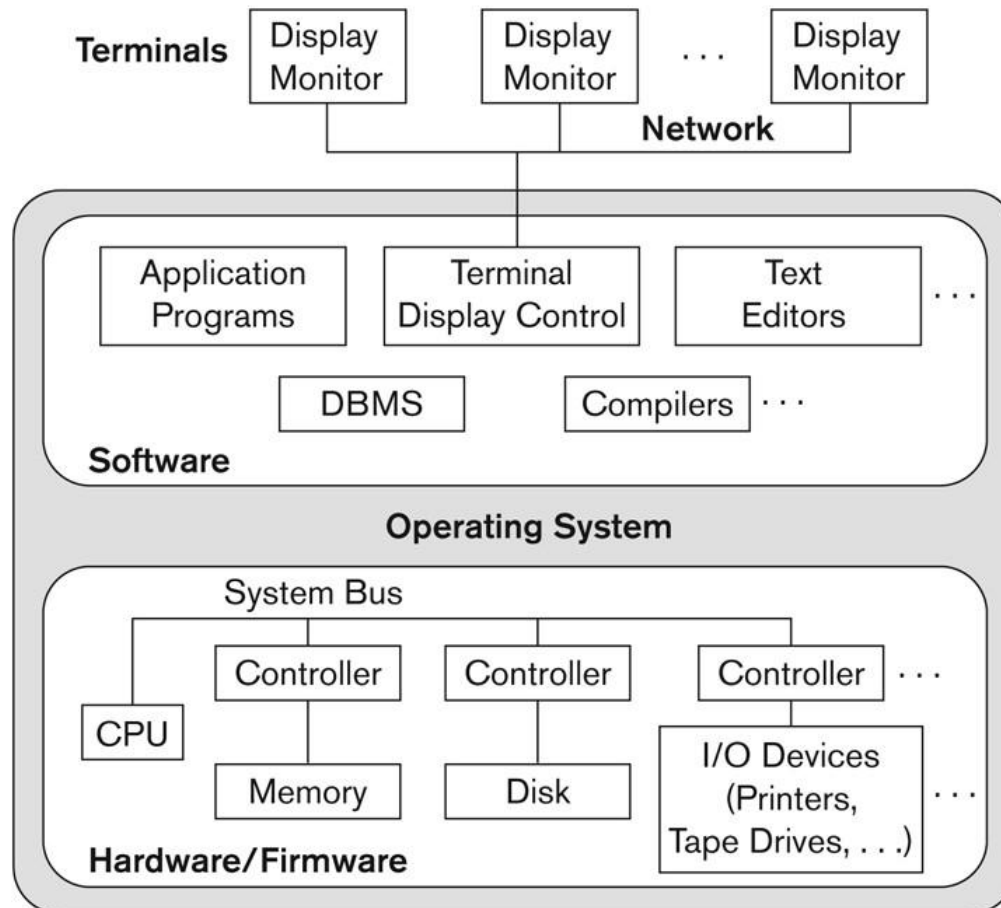
Component modules of a DBMS and their interactions.

# 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.

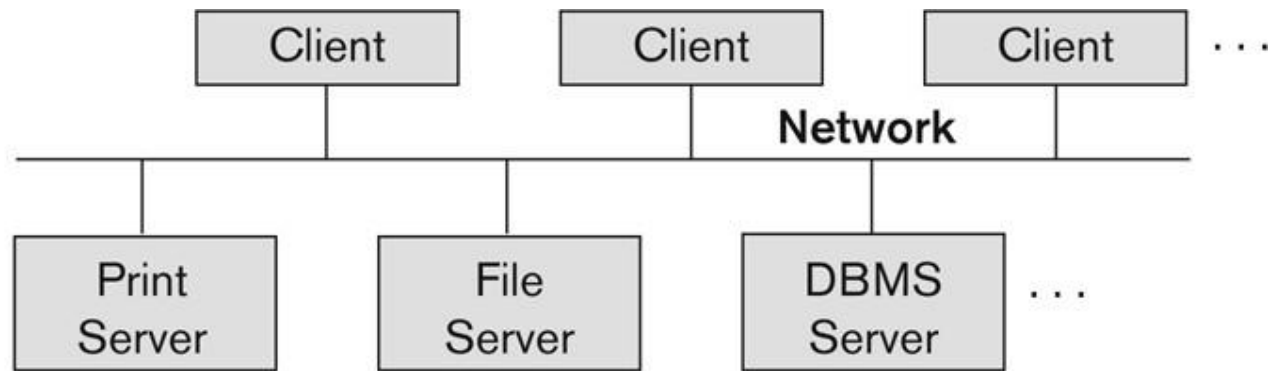
# Basic 2-tier Client-Server Architectures

- 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

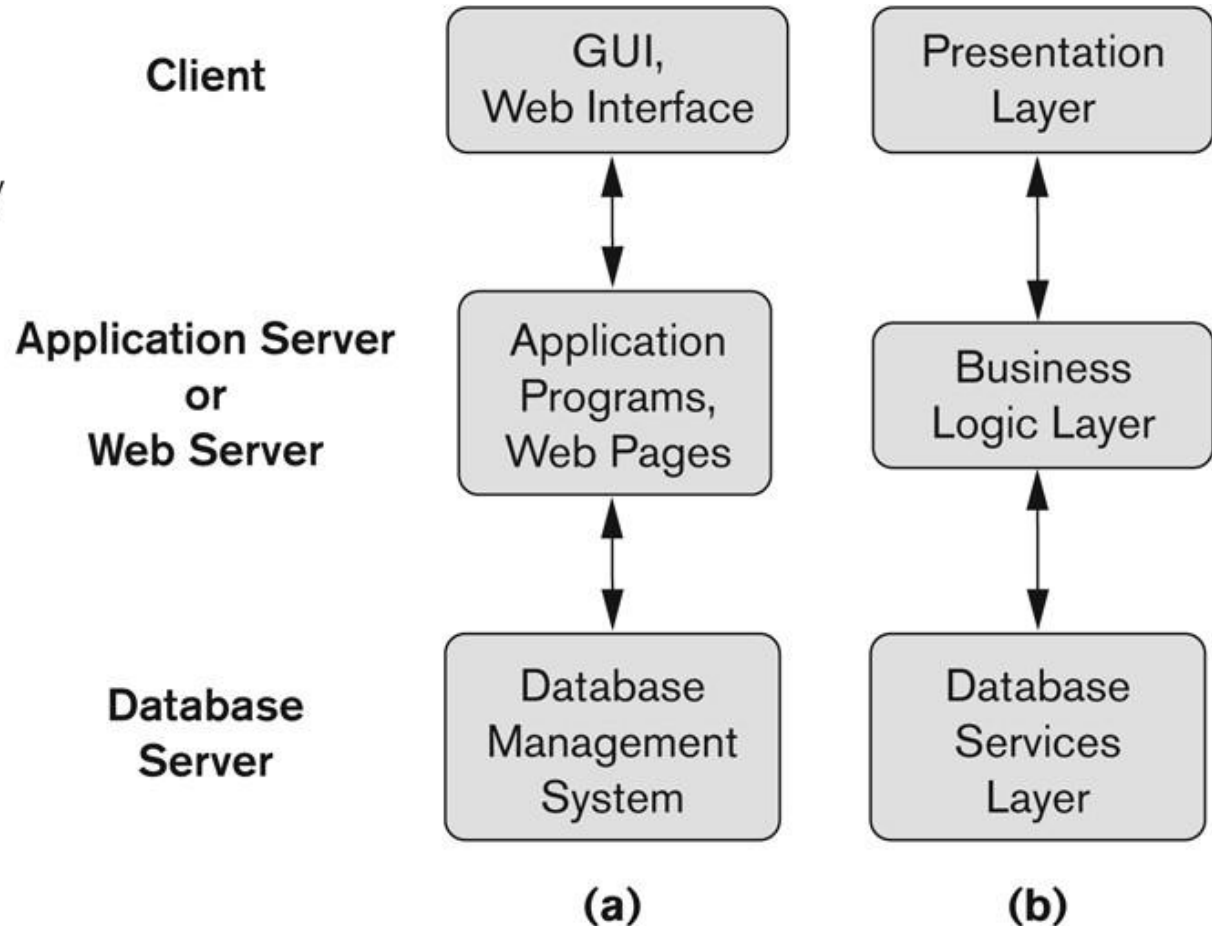
# Three Tier Client-Server Architecture

- 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 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 DBMSs.
- Other 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

# Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase 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.

# Cost considerations for DBMSs

- Cost Range: from free open-source systems to configurations costing millions of dollars
- Examples of free relational DBMSs: MySQL, PostgreSQL, 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.



# Other Considerations

- Type of access paths within 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 hotel/car etc. Are special purpose OLTP (Online Transaction Processing Systems)

# History of Data Models (Additional Material)

- Network Model
- Hierarchical Model
- Relational Model
- Object-oriented Data Models
- Object-Relational Models

# History of Data Models

## ■ Network Model:

- The first network DBMS was implemented by Honeywell in 1964-65 (IDS System).
- Adopted heavily due to the support by CODASYL (Conference on Data Systems Languages) (CODASYL - DBTG report of 1971).
- Later implemented in a large variety of systems - IDMS (Cullinet - now Computer Associates), DMS 1100 (Unisys), IMAGE (H.P. (Hewlett-Packard)), VAX -DBMS (Digital Equipment Corp., next COMPAQ, now H.P.).

# Network Model

- Advantages:
  - Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.
  - Can handle most situations for modeling using record types and relationship types.
  - Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET, etc.
    - Programmers can do optimal navigation through the database.

# Network Model

- Disadvantages:
  - Navigational and procedural nature of processing
  - Database contains a complex array of pointers that thread through a set of records.
    - Little scope for automated “query optimization”

# History of Data Models

## ■ Hierarchical Data Model:

- Initially implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems.
- IBM's IMS product had (and still has) a very large customer base worldwide
- Hierarchical model was formalized based on the IMS system
- Other systems based on this model: System 2k (SAS inc.)

# Hierarchical Model

- Advantages:
  - Simple to construct and operate
  - Corresponds to a number of natural hierarchically organized domains, e.g., organization (“org”) chart
  - Language is simple:
    - Uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT, etc.
- Disadvantages:
  - Navigational and procedural nature of processing
  - Database is visualized as a linear arrangement of records
  - Little scope for "query optimization"

# History of Data Models

## ■ Relational Model:

- Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
- Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
- Several free open source implementations, e.g. MySQL, PostgreSQL
- Currently most dominant for developing database applications.
- SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, ...
- Chapters 5 through 11 describe this model in detail



# History of Data Models

## ■ Object-oriented Data Models:

- Several models have been proposed for implementing in a database system.
- One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
- Additionally, systems like O2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
- Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.
- Chapter 12 describes this model.

# History of Data Models

## ■ Object-Relational Models:

- The trend to mix object models with relational was started with Informix Universal Server.
- Relational systems incorporated concepts from object databases leading to object-relational.
- Exemplified in the versions of Oracle, DB2, and SQL Server and other DBMSs.
- Current trend by Relational DBMS vendors is to extend relational DBMSs with capability to process XML, Text and other data types.
- The term “Object-relational” is receding in the marketplace.

# Chapter Summary

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