Chapter 1
Introduction

Outline

- Introduction
- File system
- History and evolution of database systems
- Types of Databases and Database Applications
- Characteristics of Database
- Database Management System
- Comparison of File System with Database Approach
- Database Users
- When Not to Use Databases
- Concerns when using a Enterprise Database

Introduction

- Modern enterprises have data that need to store in ways which will be easy to retrieve later
- Common database: list of names and addresses of people who deal with enterprise
  - For smaller business, data can be maintained on paper, word processor or spreadsheet
  - For larger business, database technology is needed
- Database management is about managing information resources of an enterprise

File System

- Computer systems were used to process business records and produce information which were **faster and more accurate than equivalent manual systems**.
- Stored groups of records in separate files, and so they were called **file processing systems.**

1. Collection of data. Any management with the file system, user has to write the procedures
2. Gives the details of the data representation and Storage of data
3. Storing and retrieving of data cannot be done efficiently
4. Concurrent access to the data in the file system has many problems
5. Doesn’t provide crash recovery mechanism
6. Protecting a file under file system is very difficult

Example:

- Consider part of a savings-bank enterprise that stores information about all customers and savings accounts in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including:
- A program to debit or credit an account
- A program to add a new account
- A program to find the balance of an account
- A program to generate monthly statements

- **For example, suppose that the savings bank decides to offer checking accounts.**

Drawbacks of File System

- **Data Redundancy and Inconsistency**
- **Difficulty in Accessing Data**
- **Data Isolation**
- **Integrity Problems**
- **Atomicity Problem**
- **Concurrent Access Anomalies**
- **Security Problems (ex:**payroll personnel)

History of Database Systems

1950s and early 1960s:

- **Magnetic tapes** were developed for data storage
- Data processing tasks such as payroll were automated, with data stored on **tapes.**
  - Data could also be input from punched card decks, and output to printers.
- Late 1960s and 1970s: The use of **hard disks** in the late 1960s changed the scenario for data processing greatly, since hard disks allowed direct access to data.
- With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.
- In the 1970's the EF CODD defined the **Relational Model.**

History of Database Systems(contd..)

In the 1980's:

- Initial commercial relational database systems, such as **IBM DB2, Oracle, Ingress, and DEC Rdb**, played a major role
- In the early 1980s, relational databases had become competitive with network and hierarchical database systems even in the area of performance.
- The 1980s also saw much research on **parallel and distributed databases**, as well as initial work on **object-oriented databases.**

History of Database Systems(contd..)

Early 1990s:

- The **SQL language** was designed primarily for the transaction processing applications.
- **Decision support and querying** were seen as a major application area for databases.
- Database vendors also began to add object-relational support to their databases.

Late 1990s:

- The major event was the explosive growth of the World Wide Web.
- Database systems now had to support very high transaction processing rates, as well as very high reliability and 24 * 7 availability
- Database systems also had to support Web interfaces to data.

Evolution of Database Systems

- File Management System
- Hierarchical database System
- Network Database System
- Relational Database System

1. File Management System

- All data is stored on a single large file
- Disadvantage :
  - searching a record or data takes a long time
  - updating or modifications to the data cannot be handled easily
  - sorting the records took long time and so on
- All these drawbacks led to the introduction of the Hierarchical Database System.

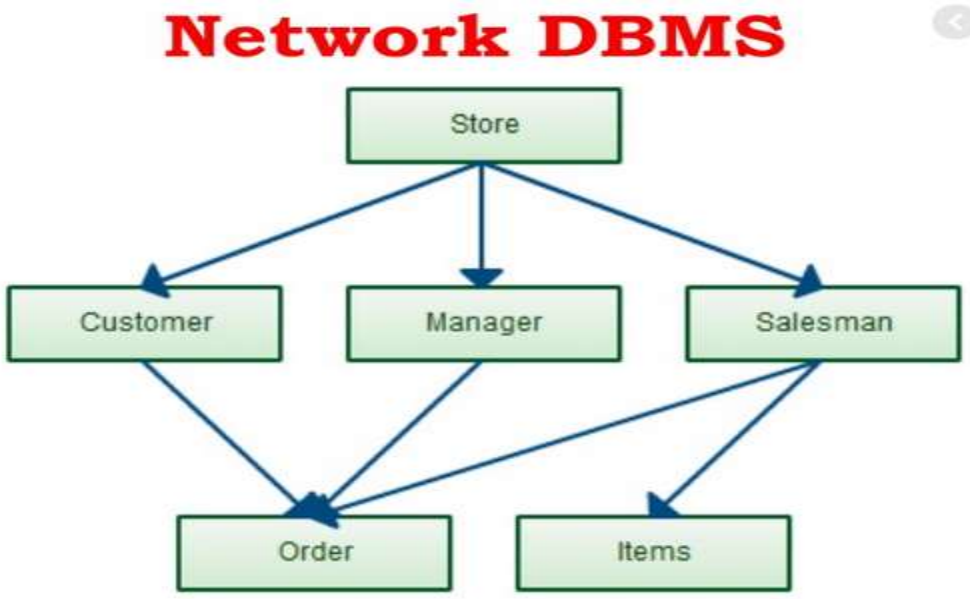
2. Hierarchical Database System

- Advantage: FMS drawback of accessing records and sorting records which took a long time was **removed by parent-child relationship between records in database**.
- Drawback: any modification or addition made to the structure then the whole structure needed alteration which made the task a tedious one.

Fig: Hierarchical Database System

3. Network Database System

- In this the main concept of **many-many relationships got introduced**.
- But this also followed the same technology of pointers to define relationships with a difference in this made in the introduction if grouping of data items as sets.



4. Relational Database System

- In order to overcome all the drawbacks of the previous systems, the Relational Database System got introduced in which **data get organized as tables and each record forms a row with many fields or attributes in it**.
- Relationships between tables are also formed in this system.

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

What is Database?

- A database is a well organized collection of data that are related in a meaningful way which can be accessed in different logical orders but are stored only once

  Data in the database is therefore integrated, structured and shared

- Main features of data in a database are:
  - Well organized
  - Related
  - Accessible in different orders without great difficulty
  - Stored only once

Database-System Applications

- ***Enterprise Information****: (Sales, Accounting, Human resources, Manufacturing)*
- ***Banking and Finance****: (Banking, Credit card transactions, Finance*)
- ***Universities***: (student information, course registrations, grades, standard enterprise information)
- ***Airlines***: (reservations, schedule information)
- ***Telecommunication***:(call records, monthly bills, balances on prepaid calling cards, and communication networks information)

Different Types of Databases

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Distributed Database
  - Real-time Databases

Main Characteristics of the Database System Approach

1. 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.

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

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	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 enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Main Characteristics of the Database System Approach (contd..)

2. Insulation between programs and data:

- The structure of data files is stored in the DBMS catalog separately from the access programs, call this property as **program-data independence**
- Allows changing data structures and storage organization without having to change the DBMS access programs.

3. 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

Main Characteristics of the Database System Approach (contd..)

4. 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.

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(a)

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

(b)

Main Characteristics of the Database System Approach (contd..)

5. Sharing of data and multi-user transaction processing:

- *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
- Allowing a set of **concurrent users** to retrieve from and to update the database.
- *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 Management System

- A **database-management system**(DBMS) is a collection of interrelated data and a set of programs to access those data.
- General purpose software system

Goals of DBMS:

primary goal: **provide a way to store and retrieve database information that is both *convenient* and *efficient***

1. Manage large bodies of information
2. Secure information against system failure or tampering
3. Permit data to be shared among multiple users

Simplified Database System Environment

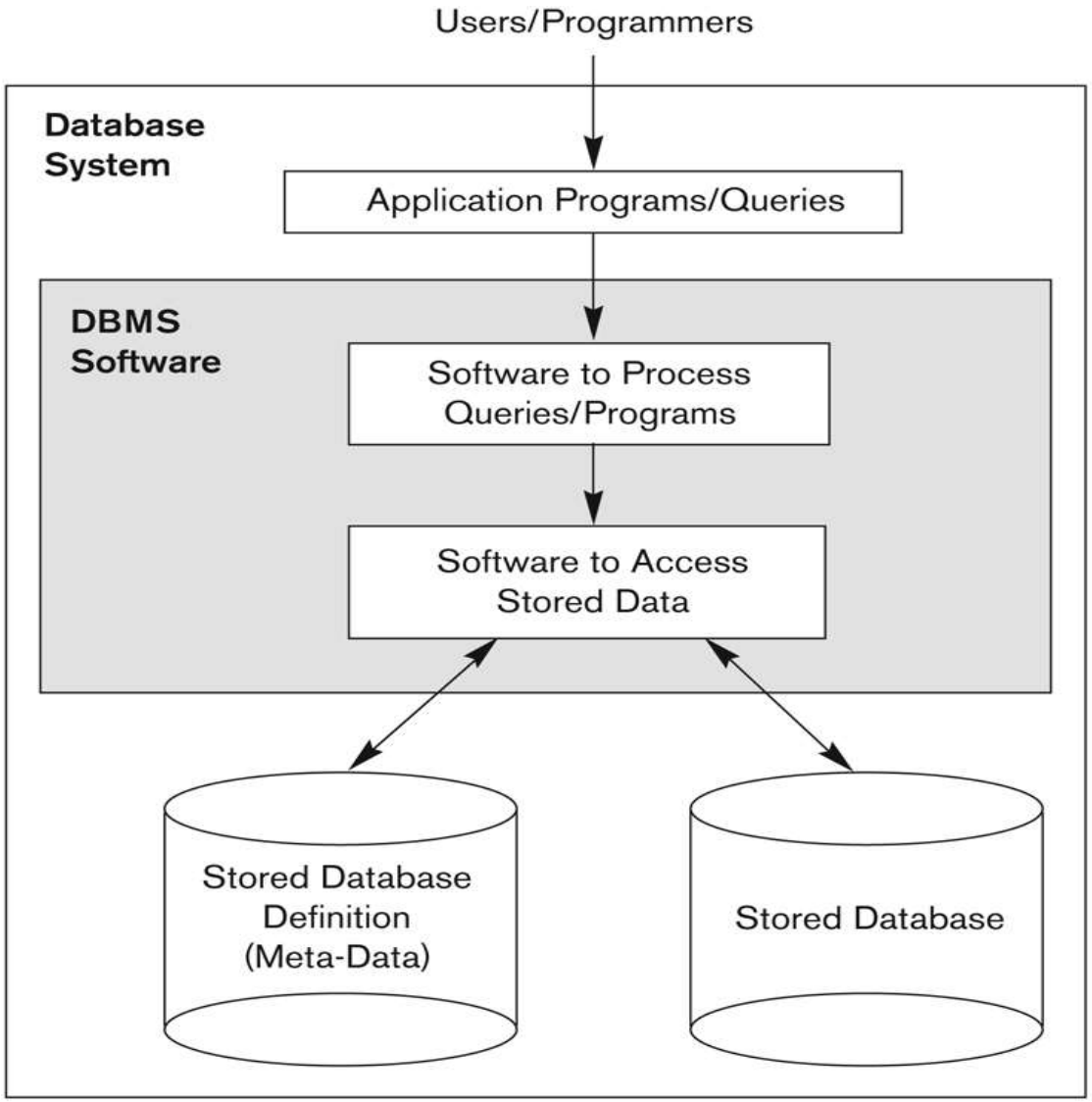


Figure 1.1
A simplified database system environment.

Advantages of DBMS over File System

- Controlling redundancy in data storage and in development and maintenance efforts
  - Sharing of data among multiple users
- Restricting unauthorized access to data.
- Providing persistent storage for program Objects
- Providing Storage Structures (e.g. indexes) for efficient Query Processing

Advantages of DBMS over File System

- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.

Comparison of File System with Database Approach

File Processing System	DBMS
File system is a collection of data. Any management with the file system, user has to write the procedures	DBMS is a collection of data and user is not required to write the procedures for managing the database.
File system gives the details of the data representation and Storage of data.	DBMS provides an abstract view of data that hides the details.
In File system storing and retrieving of data cannot be done efficiently.	DBMS is efficient to use since there are wide varieties of sophisticated techniques to store and retrieve the data.
Concurrent access to the data in the file system has many problems like : Reading the file while deleting some information, updating some information	DBMS takes care of Concurrent access using some form of locking.

Comparison of File System with Database Approach(contd..)

File Processing System	DBMS
File system doesn't provide crash recovery. Eg. While we are entering some data into the file if System crashes then content of the file is lost	DBMS has crash recovery mechanism, DBMS protects user from the effects of system failures.
Protecting a file under file system is very difficult.	DBMS has a good protection mechanism.

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 (contd..)

- **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 Users(contd..)

- **Logical designers**: identify data and relationships between them for application without taking into account how software will be used
- **Physical designers**: look at how database model can be best mapped to physical storage(hard disk)

- **System Analysts and Application Programmers:**
  - Determine the requirements of end users
  - ***System analysts*** develop specifications for application that meet user requirements
  - ***Application Programmers*** then implement these specifications as application programs is appropriately tested, documented and maintained

Database Users(contd..)

- **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.
      - Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.
    - **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.

Database Users(contd..)

- **Workers behind the scene**
  - Do not use database as part of their job
  - **Tool developers** that develop tools for database design, performance monitoring, graphical interfaces and test data generation
  - **Operators and maintenance personnel as well as system administrator** who are responsible for actual running and maintenance of hardware and software environment

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:**
  - Simple, well-defined database applications that are not expected to change
  - Stringent real-time requirements that may not be met because of DBMS overhead.
  - Embedded systems with limited storage capacity, where general purpose DBMS would not fit
  - No multiple users access to data

Enterprise Database

- When an enterprise uses a DBMS, all enterprise data may be integrated into one system, thereby providing following advantages:
  - Redundancies and inconsistencies can be reduced
  - Better service can be provided to users
  - Cost of developing and maintaining system is low
  - Standards can be enforced
  - Security can be improved
  - Integrity can be improved
  - Data model must be developed

Concerns with using an enterprise database

- Centralized enterprise DBMS provides online access to database for many users concurrently
  - As large number of users will be accessing enterprise database, enterprise may incur additional risks as follows:

1. Enterprise vulnerability maybe higher
2. Confidentiality, privacy and security maybe compromised
3. Data quality and integrity may be lower
4. Cost of building and using DBMS can be high
5. Difficult to change database when necessary

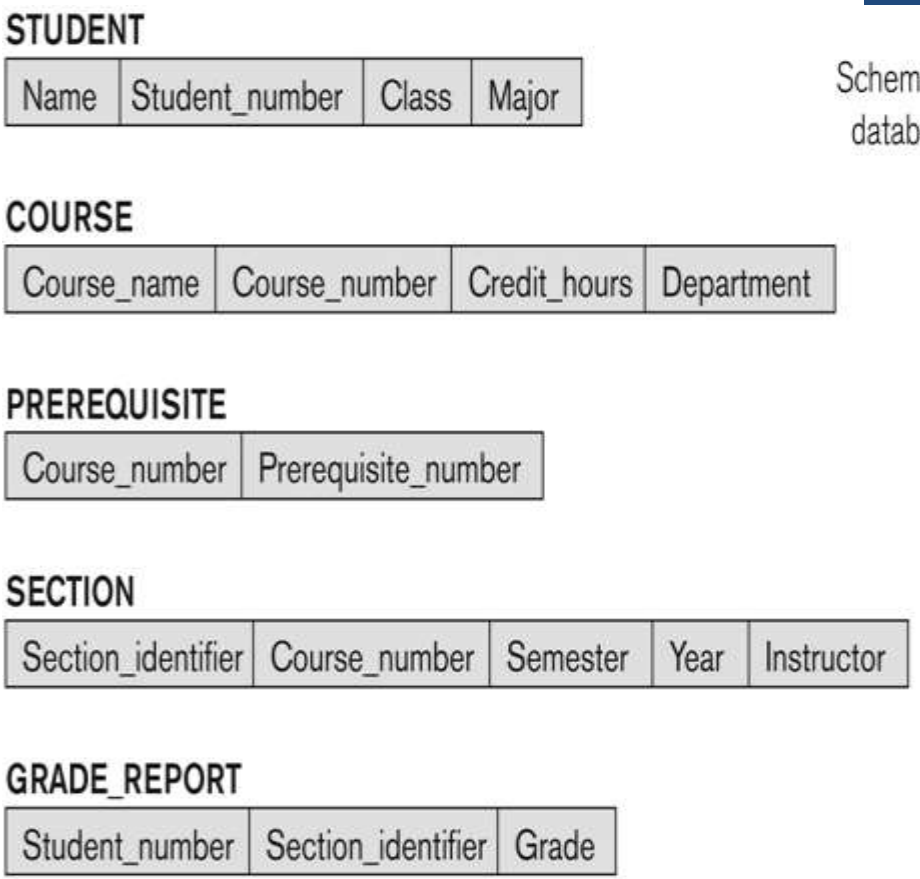
Module 1.2

Outline

- **Schemas and States**
- **Three-Schema Architecture**
- **Data Independence**
- **Database administrator**
- **DBMS Languages and Interfaces**
- **Database System Utilities and Tools**
- **Centralized and Client-Server Architectures**

Schemas

- **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.



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.

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.

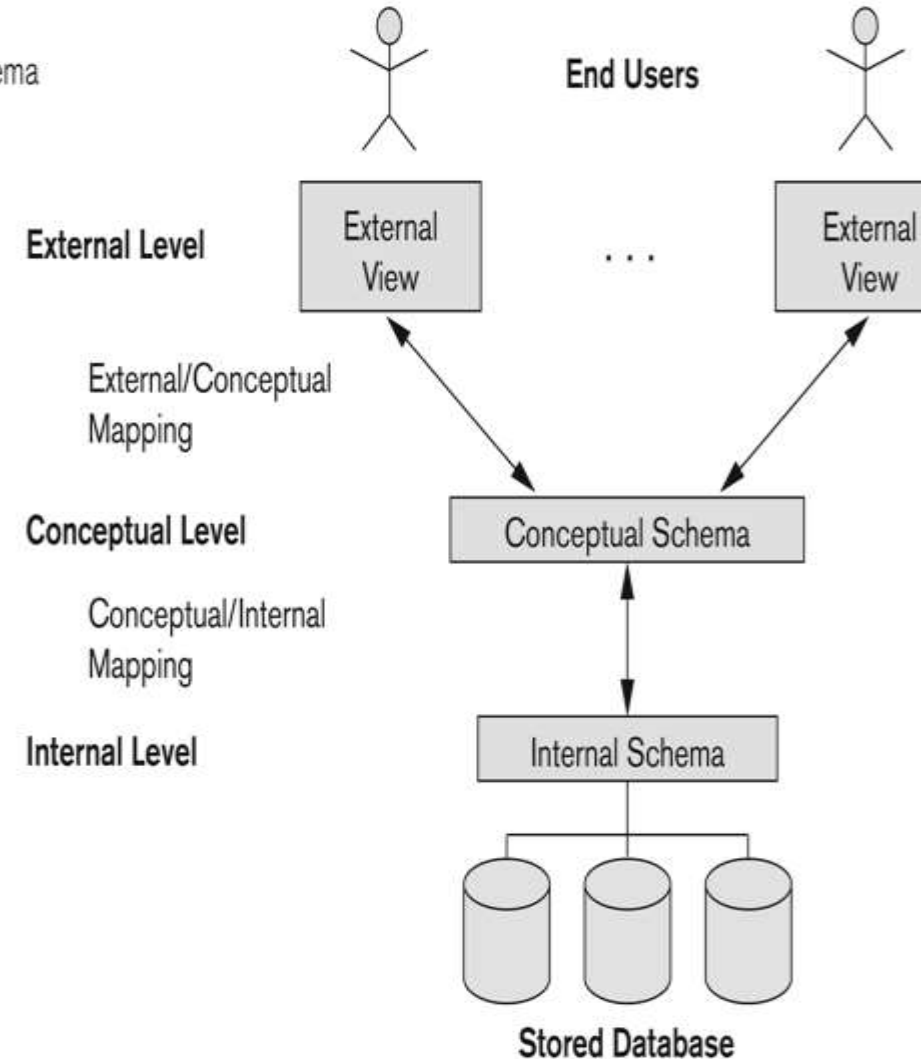
Database Schema vs. Database State

- **Distinction**
  - The ***database schema*** changes very infrequently.
  - The ***database state*** changes every time the database is updated.

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

Figure 2.2
The three-schema architecture.



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.

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)

Example

- Determine three level view of database for staff payroll office in an university which has following functionalities to be performed as external view:

1. Each staff member's **employee number, full name ad address**
2. Each staff member's **tax information** like number of dependants and other relevant information
3. Each staff member's **bank information** like account number where salary is deposited
4. Each staff member's employment **status, salary level and leave information**

Data Independence

- Deals with independence between the way the data is structured and program that manipulate it
- Data independence is not possible in file processing systems, as file structure and application programs are tightly coupled together
- Data independence is possible in DBMS only because **application programs do not deal with data in database directly**
- Three level DBMS illustrates two types of data independence :
  - **Physical Data Independence**
  - **Logical Data independence**

Data Independence(contd..)

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

- **Physical Data Independence:**
  - 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

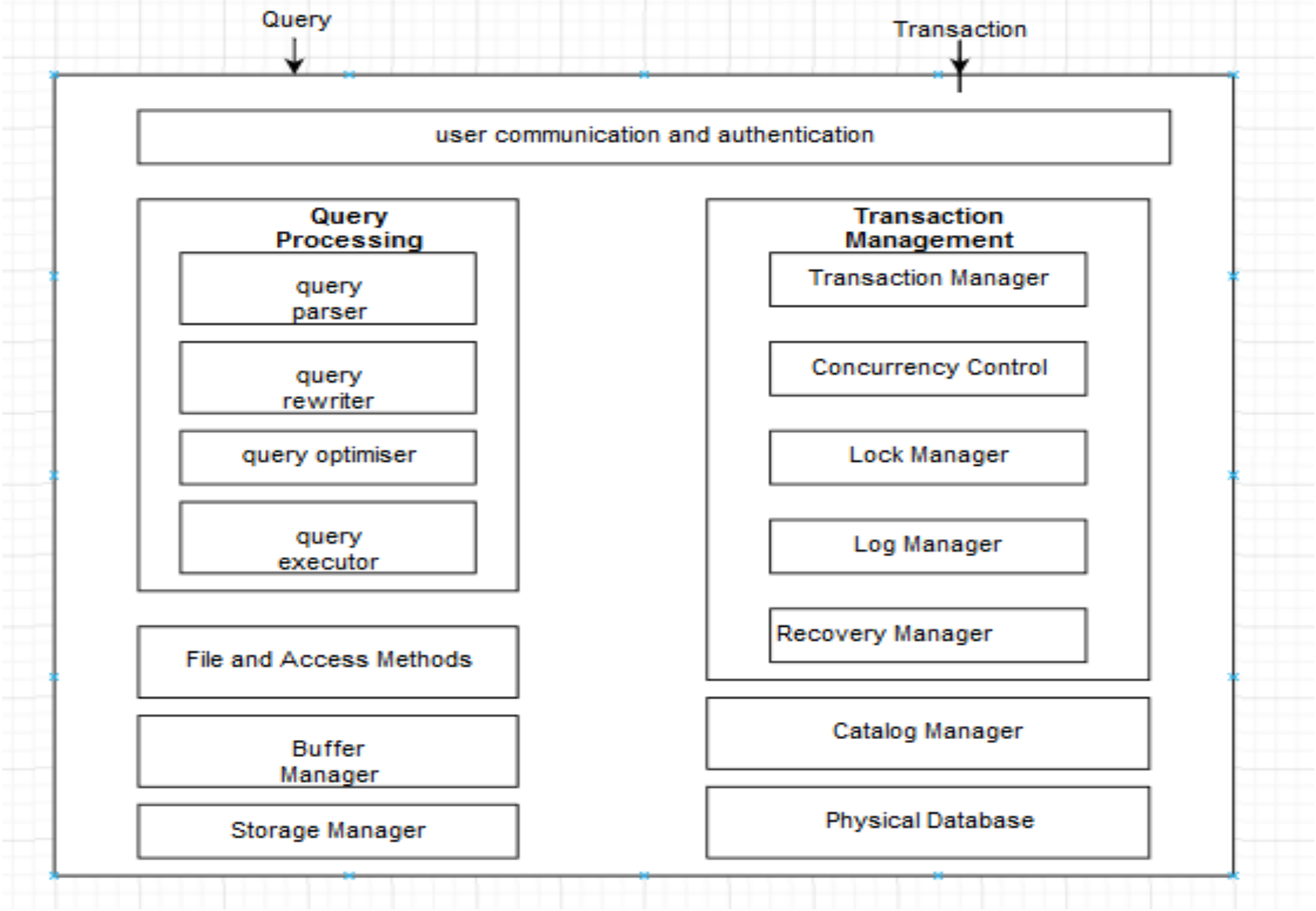
Data Independence (contd..)

- 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.
  - Application program need not know:
    - Ordering of data fields in a record
    - Ordering of records in a file
    - Size of each record
    - Size of each field
    - Format and type of each data item
    - Whether file is sorted or not

DBMS System Architecture

- Complex piece of software that consists of a number of modules
- **Agent** that allows communication between different types of users with physical database and operating system
- DBMS Provides following facilities:
  - User communication with the system
  - Authentication and authorization
  - Query processing (syntax checker and translator)
  - Transaction management (concurrency control, logging manager, locking manager, recovery manager)
  - Database Management
  - Catalog Manager

DBMS System Architecture (contd.)



DBMS System Architecture (contd.)

1. **User Communication, Authentication and Authorization**
   - User communicates with database which can be two tier or three tier system
   - System include mechanism for user to log in and be authenticated
2. **Query Processor**
   - Includes subcomponents like query parser, query rewriter, query optimizer and query executor
3. **Transaction Management and Concurrency Control**
   - Modern database systems are multi-user systems
   - Transaction is assumed to be logical unit of work
   - Concurrency control is needed to coordinate concurrent accesses to DBMS so that overall correctness is maintained

DBMS System Architecture (contd.)

4. **Files and Access Methods**
   - Interacts with physical storage of database
   - Responsible for storing database on disk and providing DBA a number of data structure opinions for storing database
   - Assist in maintaining metadata of database and indexes of various files
5. **Buffer Manager**
   - A part of database is loaded in main memory for retrieval and updates and is written back on disk once it is modified
   - Responsible for handling interfacing between main memory and disk
   - Maintains number of pages of same size and large buffer size is recommended

DBMS System Architecture (contd.)

7. **Storage Manager**
   - Support database management system for access paths and indexes
8. **Catalog Manager**
   - Metadata(describe data structure) is maintained in catalog and stored as relational tables
9. **Physical Database**
   - Physical storage of database normally on disk
   - Database tables are stored using different file structures and indexes
   - Performance of large database depends on how effectively data has been structured on disk

Database Administrator

- A person(group of persons) centrally located with an overall view of the database, is needed to keep the database running smoothly are called database administrator (DBA)

- DBA should normally understand the following:
  - What data is in database and where it come from?
  - What does data in database mean?
  - How reliable, accurate and authoritative is the data?
  - All enterprise applications that need data are able to use it?
  - Who governs definition, lifecycle and use of data in database?

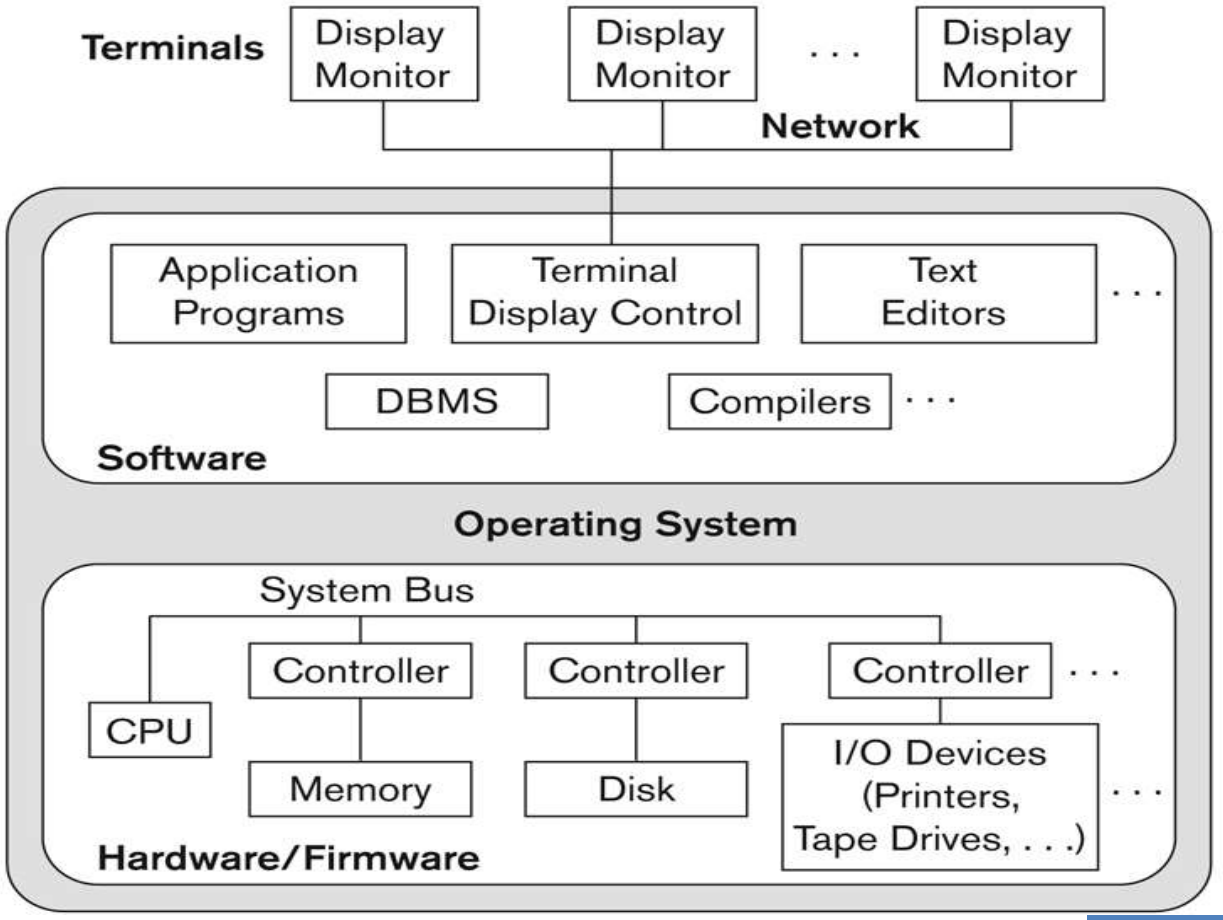
Database Administrator(contd..)

- **DBA Tasks include following:**

1. Developing Conceptual Schema
2. Deciding which DBMS to use
3. Defining the database and loading the database contents
4. Assisting and approving applications and access
5. Deciding data structures
6. Backup and recovery
7. Monitoring actual usage

Centralized and Client-Server DBMS Architecture

- **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.



Basic 2-tier Client-Server Architecture

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

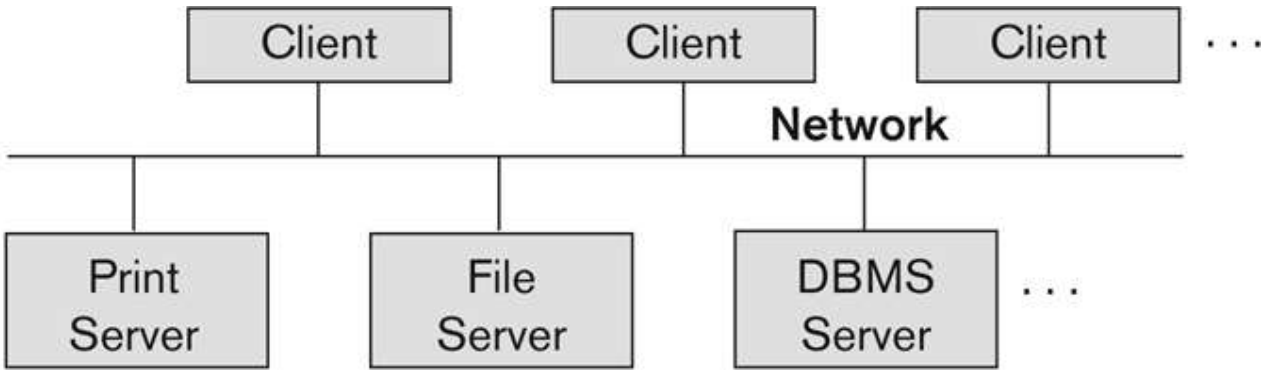


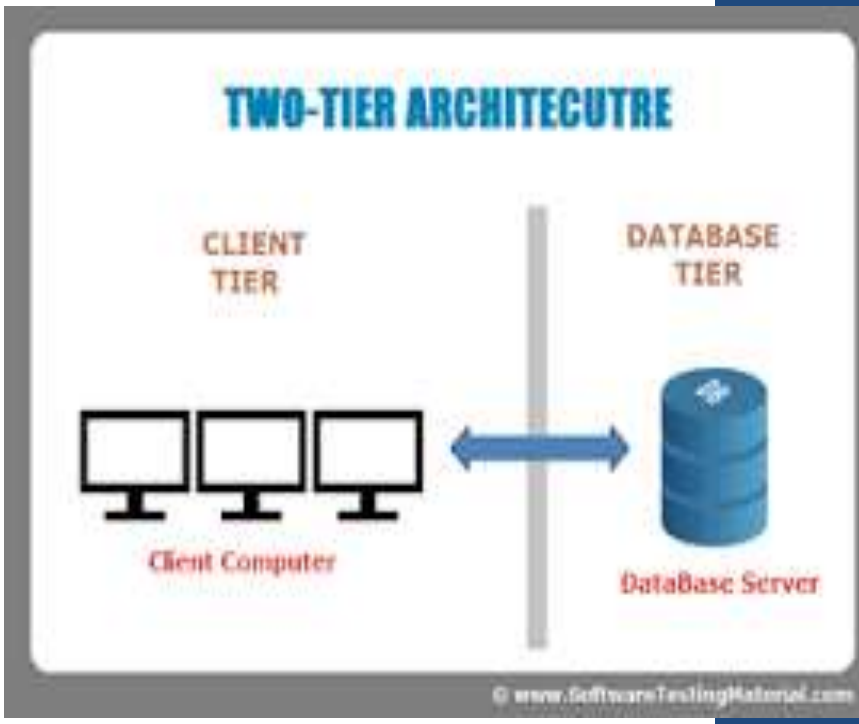
Figure 2.5
Logical two-tier client/server architecture.

Clients

- Provide appropriate interfaces through a client software module
- Clients may be diskless machines or PCs or Workstations with disks
- 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
- 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



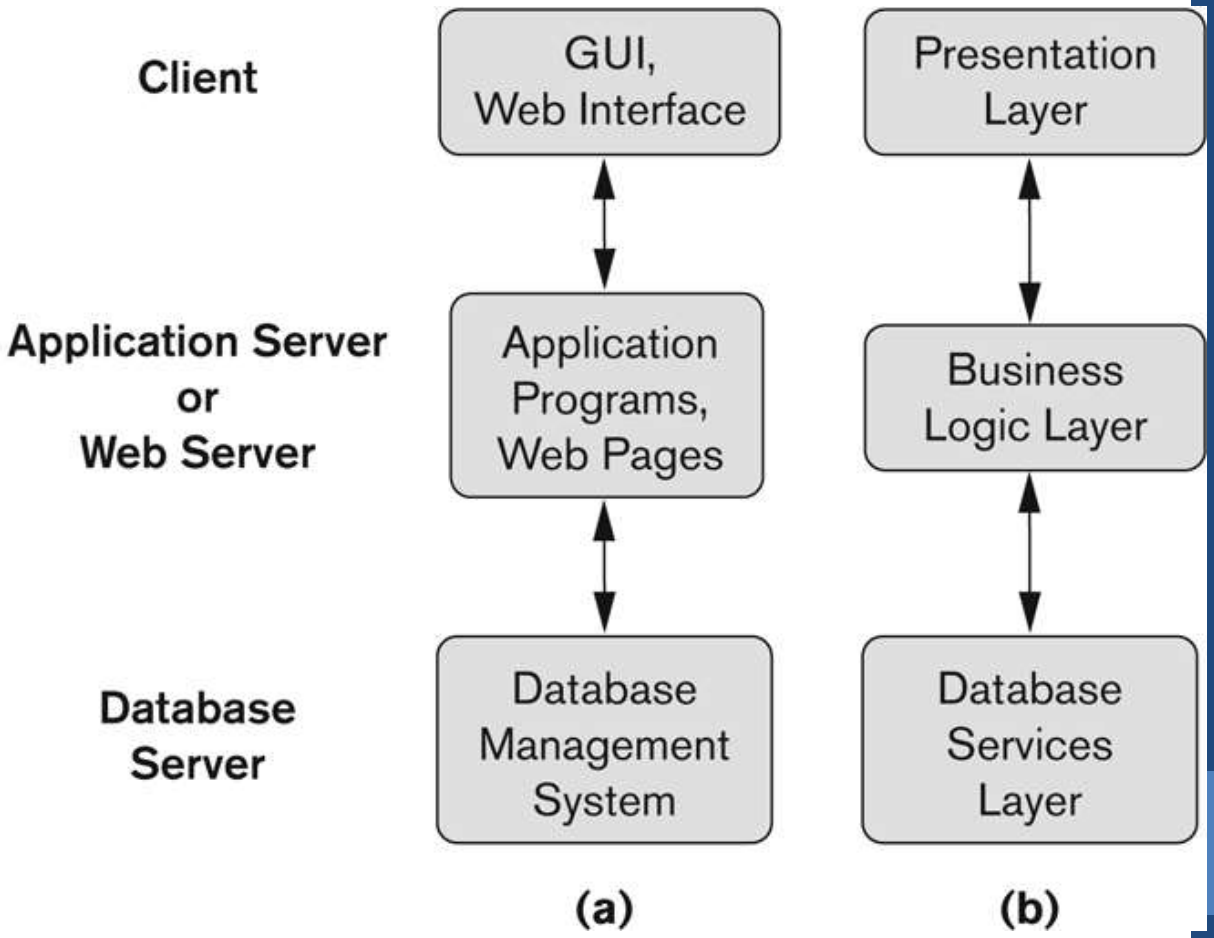
Client and server must install appropriate client module and server module software for ODBC or JDBC

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 agent 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

Three-tier client-server architecture(contd..)

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



DBMS Languages

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language(DCL)
4. Transaction Control Language(TCL)

DBMS Languages(contd..)

1. **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)
2. **Data Manipulation Language (DML):**
   - Used to specify database retrievals and updates
   - **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.
   - **Low Level or Procedural Language:**
     - Retrieve data one record-at-a-time
     - Loops are needed to retrieve multiple records

DBMS Languages(contd..)

3. Data Control Language

- Grant privilege to a user using the GRANT statement.
- **GRANT**: Give privilege to access the database.
- **REVOKE:** Take back the privilege to access the database

4. Transaction Control Language

- Manage transactions in the Database using the Transaction Control Language:
- **COMMIT**: Save the work.
- **SAVEPOINT**: Set a point in transaction to rollback later
- **ROLLBACK**: Restores since last commit

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.

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
  - **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

User-Friendly DBMS Interfaces

- Menu-based, popular for browsing on the web
- Forms-based, designed for naïve users
- Graphics-based
  - (Point and Click, Drag and Drop, etc.)
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces
- Natural language: requests in written English

Other DBMS Interfaces

- Speech as Input and Output
- Web Browser as an interface
- 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