

Chapter Two

Database Management Systems (DBMS)

Database Management System (DBMS)

- ***DBMS* is a software system that enables users to define, create, maintain, and control access to the database.**
- **It allows user to define the structure the database using Data Definition Language (DDL) and manipulate the data in the database using Data Manipulation Language (DML)**

History of Database Systems

- **First-generation**
 - **Hierarchical and Network**
- **Second generation**
 - **Relational**
- **Third generation**
 - **Object-Relational**
 - **Object-Oriented**

Database Languages

- **Data Definition Language (DDL)**
 - **Allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints.**
 - **The result of the compilation of the DDL statements is a set of tables stored in special files collectively called the system catalog.**

Database Languages

Cont'd..

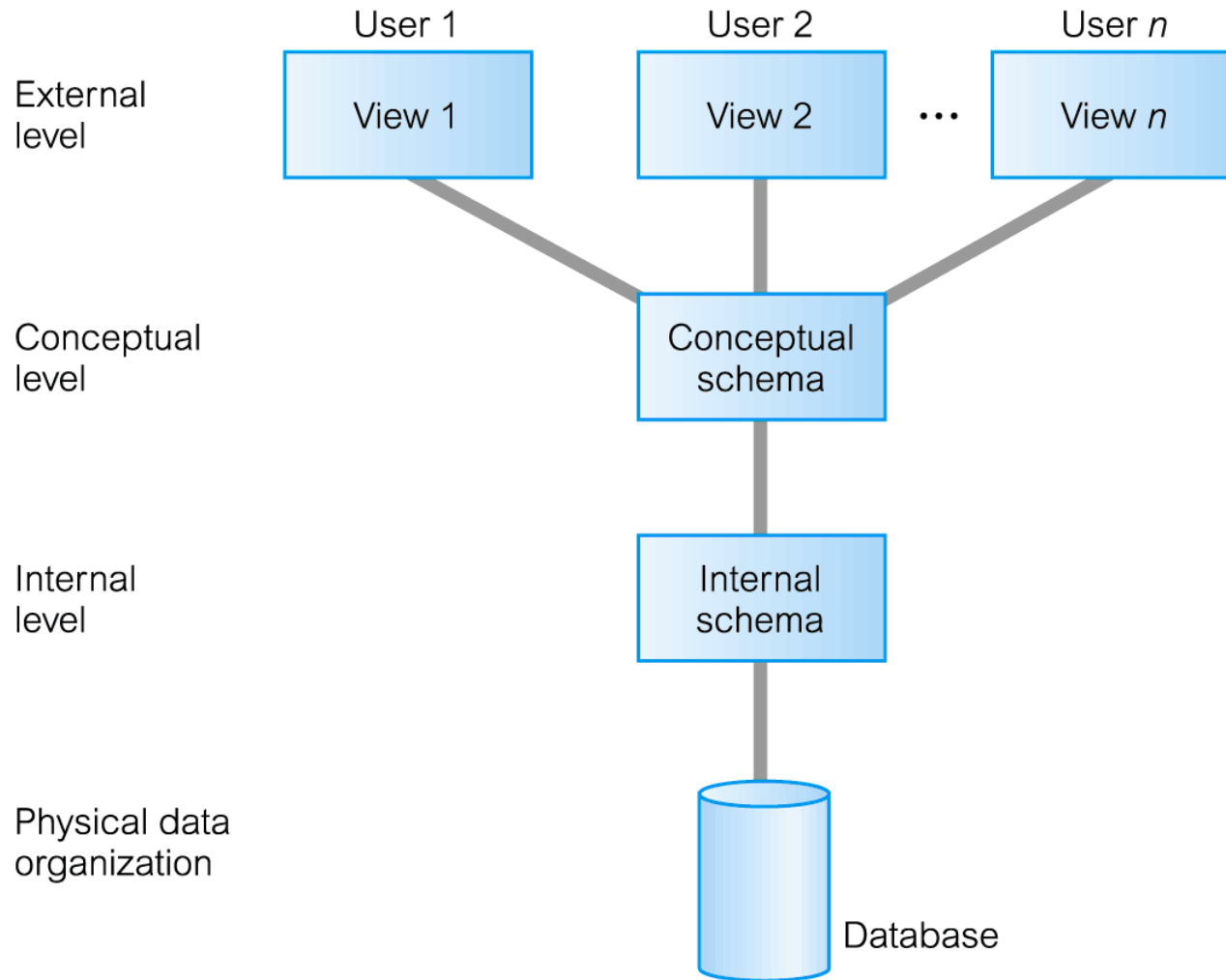
- **A *System Catalogue* is a repository of information (metadata) describing the data in the database.**
- **Typically stores:**
 - **names, types, and sizes of data items;**
 - **constraints on the data;**
 - **names of authorized users;**
 - **data items accessible by a user and the type of access;**
 - **usage statistics.**

Database Languages

Cont'd..

- **Data Manipulation Language (DML)**
 - **Provides basic data manipulation operations on data held in the database.**
- **Procedural DML**
 - **allows user to tell system exactly how to manipulate data.**
- **Non-Procedural DML**
 - **allows user to state what data is needed rather than how it is to be retrieved.**
- **Fourth Generation Languages (4GLs)**
 - **Automated CASE tools**

ANSI-SPARC Three-Level Architecture



Objectives of Three-Level ANSI-SPARC Architecture

- **All users should be able to access same data.**
- **A user's view is immune to changes made in other user's views.**
- **Users should not need to know physical database storage details.**

Objectives of Three-Level ANSI-SPARC Architecture

- **DBA should be able to change database storage structures without affecting the users' views.**
- **Internal structure of database should be unaffected by changes to physical aspects of storage.**
- **DBA should be able to change conceptual structure of database without affecting all users.**

ANSI-SPARC Three-Level Architecture

- **External Level**
 - **Users' view of the database.**
 - **Describes the part of a database which is relevant to a particular user.**
- **Conceptual Level**
 - **Community view of the database.**
 - **Describes what data is stored in the database and the relationships among the data.**

ANSI-SPARC Three-Level Architecture

- **Internal Level**
 - **Physical representation of the database on the computer.**
 - **Describes how the data is stored in the database.**
 - **This is how the OS and DBMS view Data**

Differences between Three Levels of ANSI-SPARC Architecture

External view 1

sNo	fName	lName	age	salary
-----	-------	-------	-----	--------

External view 2

staffNo	lName	branchNo
---------	-------	----------

Conceptual level

staffNo	fName	lName	DOB	salary	branchNo
---------	-------	-------	-----	--------	----------

Internal level

```
struct STAFF {  
    int staffNo;  
    int branchNo;  
    char fName [15];  
    char lName [15];  
    struct date dateOf Birth;  
    float salary;  
    struct STAFF *next;  
};  
index staffNo; index branchNo;
```

/* pointer to next Staff record */
/* define indexes for staff */

Data Independence

- **Main Concept is Upper layers are immune to changes in the lower layers**
- **Logical Data Independence**
 - **Refers to immunity of external schemas to changes in conceptual schema.**
 - **Conceptual schema changes (e.g. addition/removal of entities) should not require changes to external schema or rewriting of application programs.**
 - **Obviously particular users will be affected but not all users**

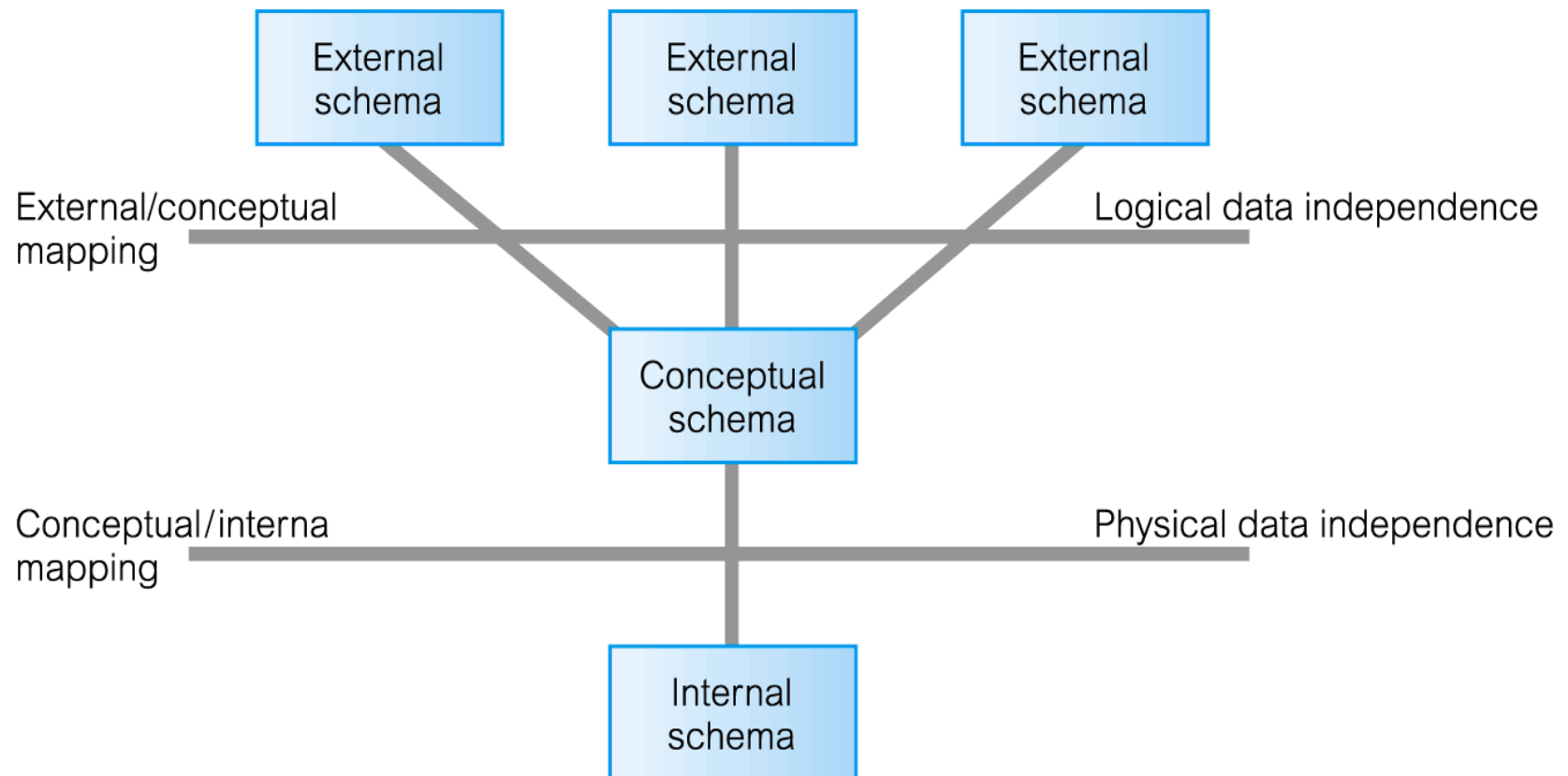
Data Independence

- **Physical Data Independence**
 - **Refers to immunity of conceptual schema to changes in the internal schema.**
 - **Internal schema changes (e.g. using different file organizations, storage structures/devices) should not require change to conceptual or external schemas.**

Schema Mapping- Provision for Data Independence

- ANSI-SPARC has three Layers of Describing Organizational Data
 - The DBMS is responsible for mapping between these three types of schema.
- **External/Conceptual Mapping**
 - This enables the DBMS to map names in the user's view on to the relevant part of the conceptual schema.
- **Conceptual/Internal Mapping**
 - This enables the DBMS to find the actual record or combination of records in physical storage that constitute a logical record in the conceptual schema, together with any constraints to be enforced on the operations for that logical record.

Data Independence and the ANSI-SPARC Three-Level Architecture



Data Model

- **Data Model:** Integrated collection of concepts for describing data, relationships between data, and constraints on the data in an organization.
- **Purpose**
 - To represent data in an understandable way.
- **Categories of data models include:**
 - Object-based
 - Record-based
 - Physical.

Data Models

- **Object-Based Data Models-**
 - **based on the concept of Entity (distinct object)**
 - Entity-Relationship- Considers only the data aspect
 - Object-Oriented- considers both data and behaviour.
- **Record-Based Data Models (Hierarchical, Network and Relational)**
 - **based on fixed format records**
 - Each record has fixed number of fields
 - each field is of a fixed length
- **Physical Data Models**
 - **Models for describing physical storage characteristics**

Object-Based Data Models

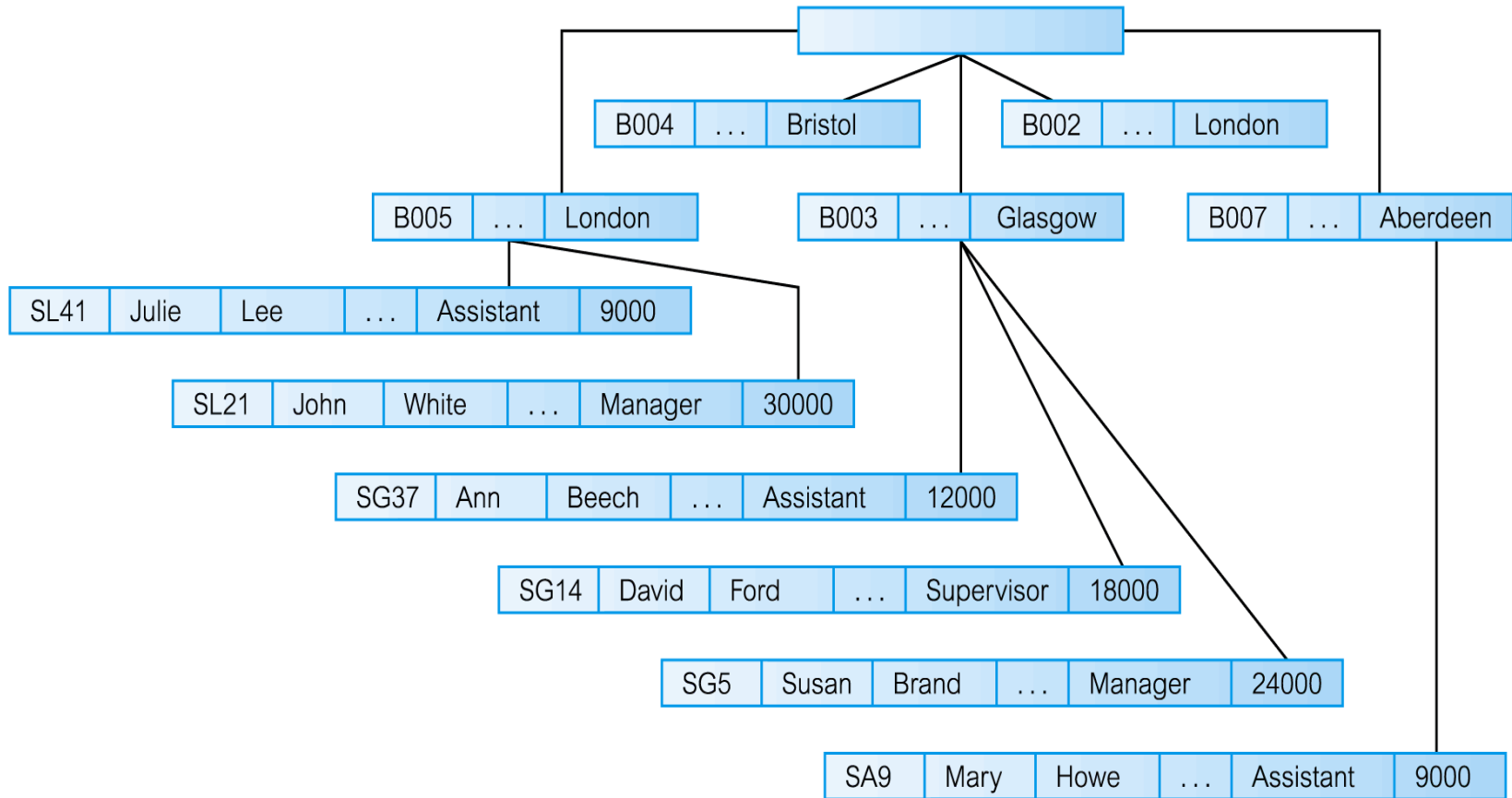
- Object-based data models use concepts such as entities, attributes, and relationships
- An **entity** is a distinct object (a person, place, thing, concept, event) in the organization that is to be represented in the database.
- An **attribute** is a property that describes some aspect of the object that we wish to record
- A **relationship** is an association between entities.

Record-based Data Models

● Hierarchical Data Model

- Record type is referred to as node or segment
- The top node is the root node
- Nodes are arranged in a hierarchical structure as sort of upside down tree
- A parent node can have more than one child node
- A child node can only have one parent node
- The relationship between parent and child is one-to-many
- Relationship is established by creating physical link between stored records (implemented as pointer)
- To add new record type or relationship, the database must be redefined and then stored in a new form.

Hierarchical Data Model

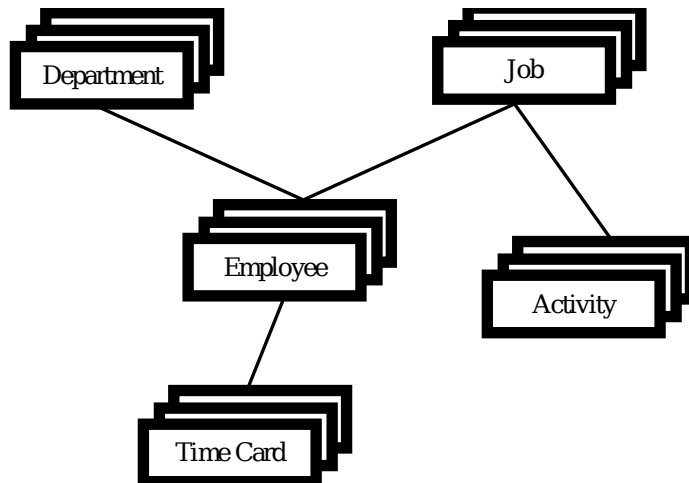
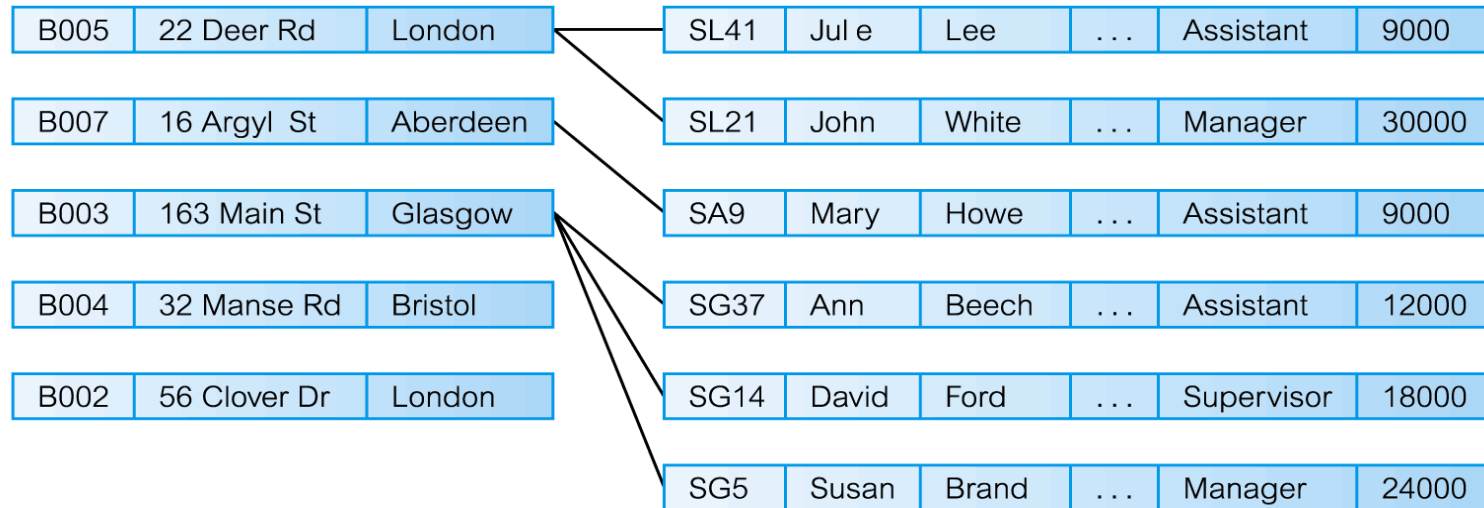


Record-based Data Models cont'd..

- **Network Data model**

- **Allows record types to have more than one parent unlike hierarchical model**
- **A network data models sees records as set members**
- **Each set has an owner and one or more members**
- **Allows many to many relationship between entities**
- **Like hierarchical model network model is a collection of physically linked records.**
- **Allows member records to have more than one owner**

Network Data Model



Record-based Data Models cont'd..

- **Hierarchical and Network**
 - **Navigational and procedural approach to data processing**
 - **Need to know the physical Database to access the data**
 - **Treat records as individual objects linked with pointers**

Record-based Data Models cont'd..

- **Relational Data Model**

- is based on the mathematical concept called **Relation**
- **Relation**: Two dimensional table that stores information or data in the form of tables □ **rows and columns**
- A **row** of the table is called tuple □ equivalent to **record**
- A **column** of a table is called attribute □ equivalent to **fields**
- Records are related by the data stored jointly in the fields of records in two tables. The related tables contain information that creates the relationship
- Uses **Declarative (as Opposed to Procedural)** approach to Database Processing

Relational Data Model

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Physical Data Models

- **Is the lowest level of data model related to database implementation.**
- **Prime importance of this model are record format, record ordering, access path and etc.**
- **Eg. Frame Memory, Unifying Model**

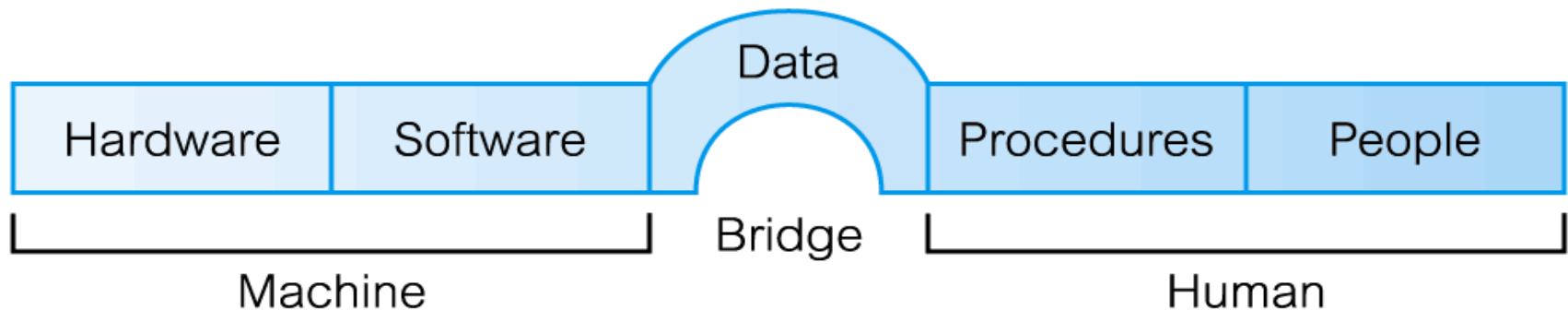
Functions of a DBMS

- **Data Storage, Retrieval, and Update.**
- **A User-Accessible Catalog.**
- **Transaction Support.**
- **Concurrency Control Services.**
- **Recovery Services.**

Functions of a DBMS

- **Authorization Services.**
- **Support for Data Communication.**
- **Integrity Services.**
- **Services to Promote Data Independence.**
- **Utility Services(Import, Export, Statistical Analysis, etc.)**

Components of the DBMS Environment



In A DBMS Environment there are *Five* basic Components

- **Hardware**
- **Software**
- **Data**
- **Procedure**
- **People**

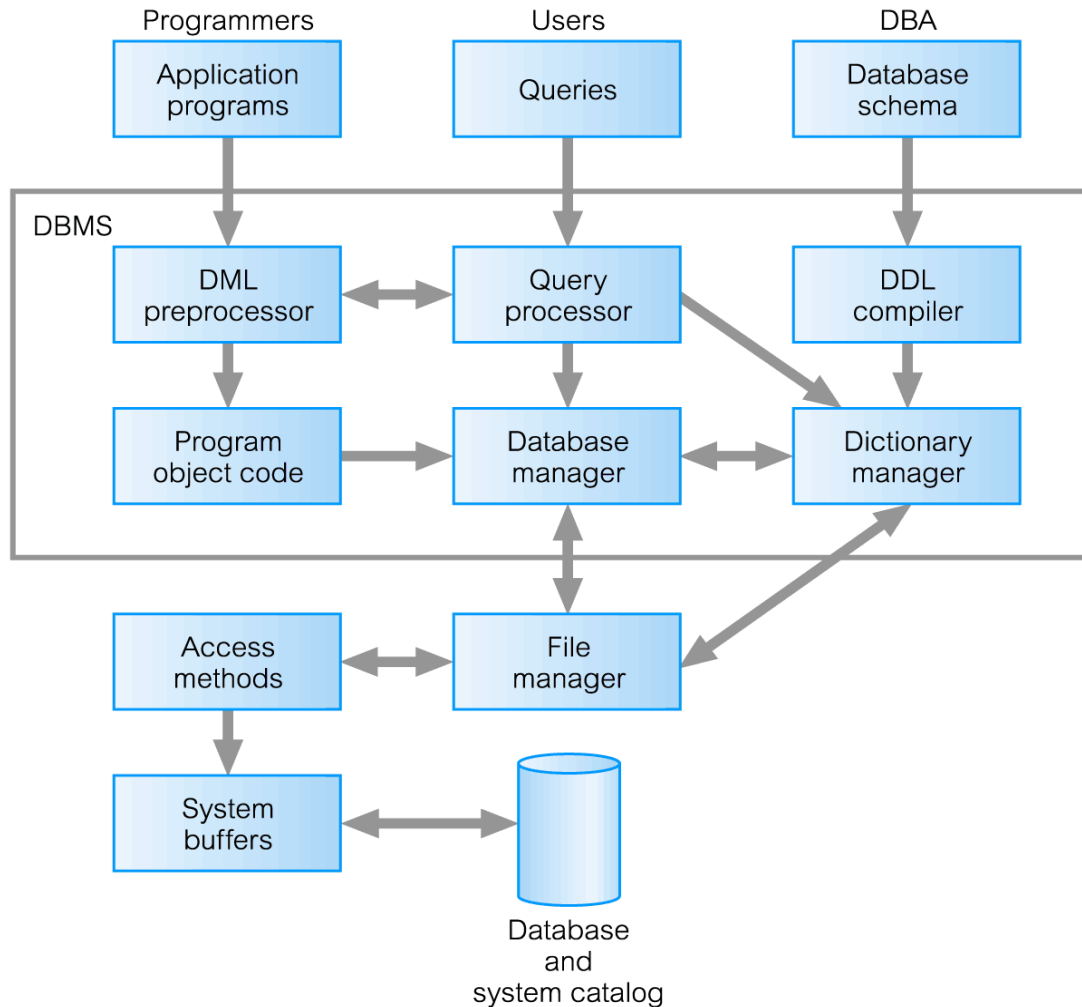
Components of DBMS Environment

- **Hardware**
 - **Can range from a PC to a network of computers.**
 - **Includes all the necessary input, output , storage and backup devices**
- **Software**
 - **DBMS, operating system, network software (if necessary) and also the application programs.**
- **Data**
 - **Data used by the organization and a description of this data called the schema.**

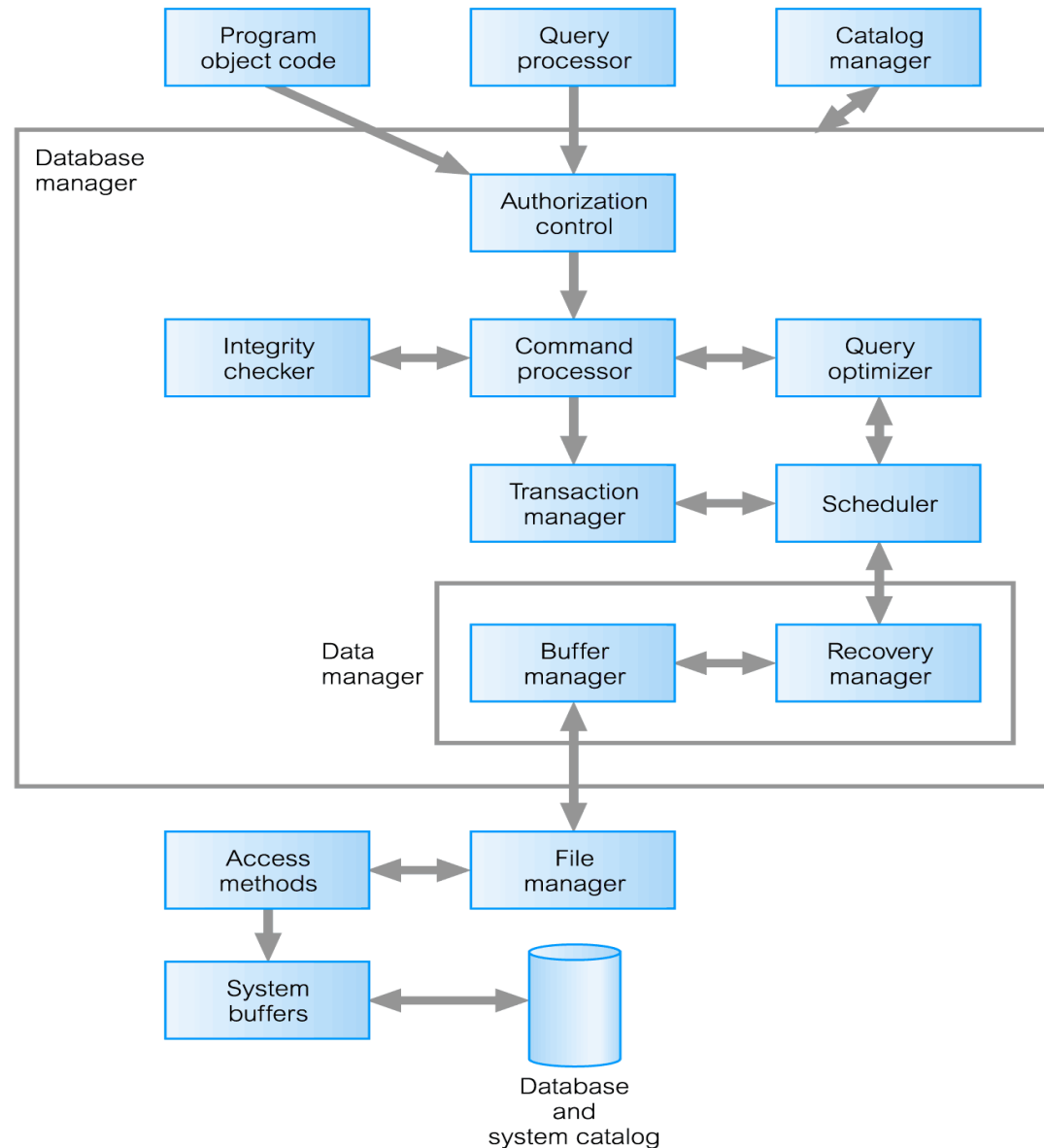
Components of DBMS Environment

- **Procedures**
 - **Instructions and rules that should be applied to the design and use of the database and DBMS.**
- **People**
 - **Different Roles taken by people while designing and using a Database systems**

Components of a DBMS



Components of Database Manager

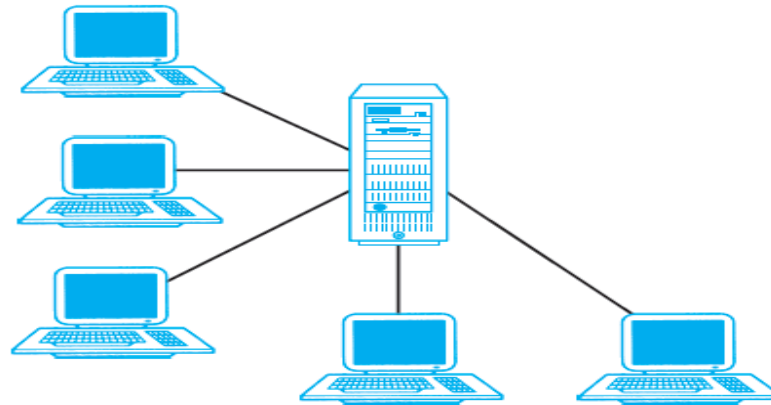


Multi-User DBMS Architectures

- **There are Common architectures that are used to implement multi-user database management systems,**
 - **Teleprocessing,**
 - **File-server, and**
 - **Client-server.**

Tele-Processing

- The traditional architecture for multi-user systems was teleprocessing, where there is one computer with a single central processing unit (CPU) and a number of terminals
- Users, just as in a mainframe system, use terminals to communicate with the central computer.



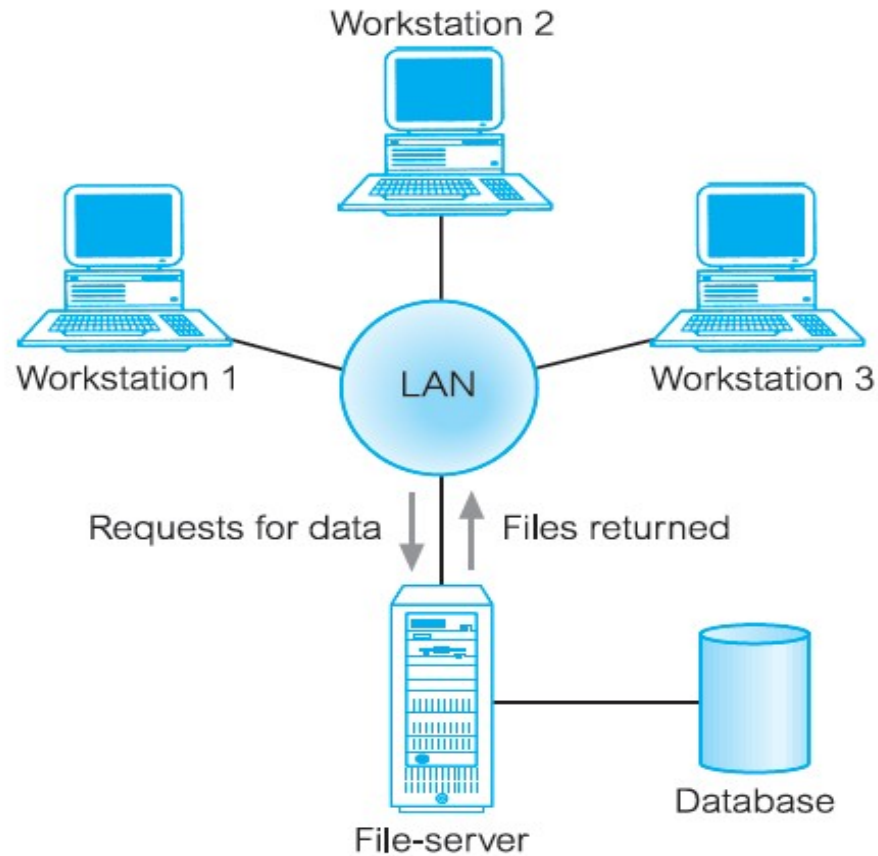
Disadvantage

- Unfortunately, this architecture placed a tremendous burden on the central computer, which not only had to run the application programs and the DBMS, but also had to carry out a significant amount of work on behalf of the terminals (such as formatting data for display on the screen).

File-Server Architecture

- In a file-server environment, the processing is distributed about the network, typically a local area network (LAN). The file-server holds the files required by the applications and the DBMS.
- However, the applications and the DBMS run on each workstation requesting files from the file-server when necessary.
- In this way, the file-server acts simply as a shared hard disk drive. The DBMS on each workstation sends requests to the file-server for all data that the DBMS requires that is stored on disk.

File-Server Cont'd...



The file-server architecture - three main disadvantages

- (1) There is a large amount of network traffic.
- (2) A full copy of the DBMS is required on each workstation.
- (3) Concurrency, recovery, and integrity control are more complex because there can be multiple DBMSs accessing the same files at the same time.

Client-Server Architecture

- To overcome the disadvantages of the first two approaches and accommodate an increasingly decentralized business environment, the client-server architecture was developed.
- Client-server refers to the way in which software components interact to form a system.
- As the name suggests, there is a **client** process, which requires some resource, and a **server**, which provides the resource.
There is no requirement that the client and server must reside on the same machine.

Possible combinations of the client-server topology.

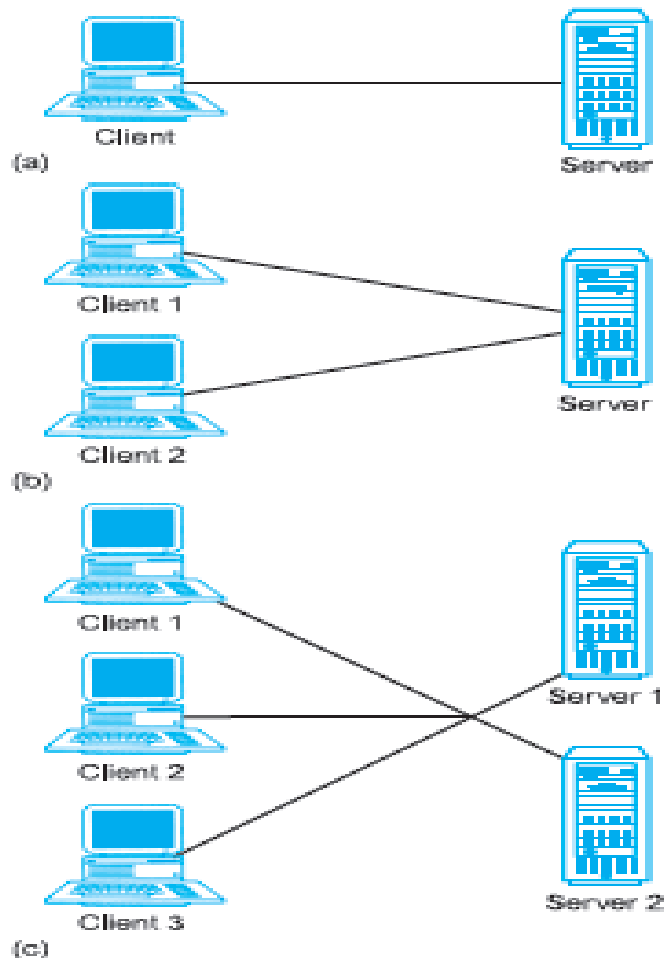


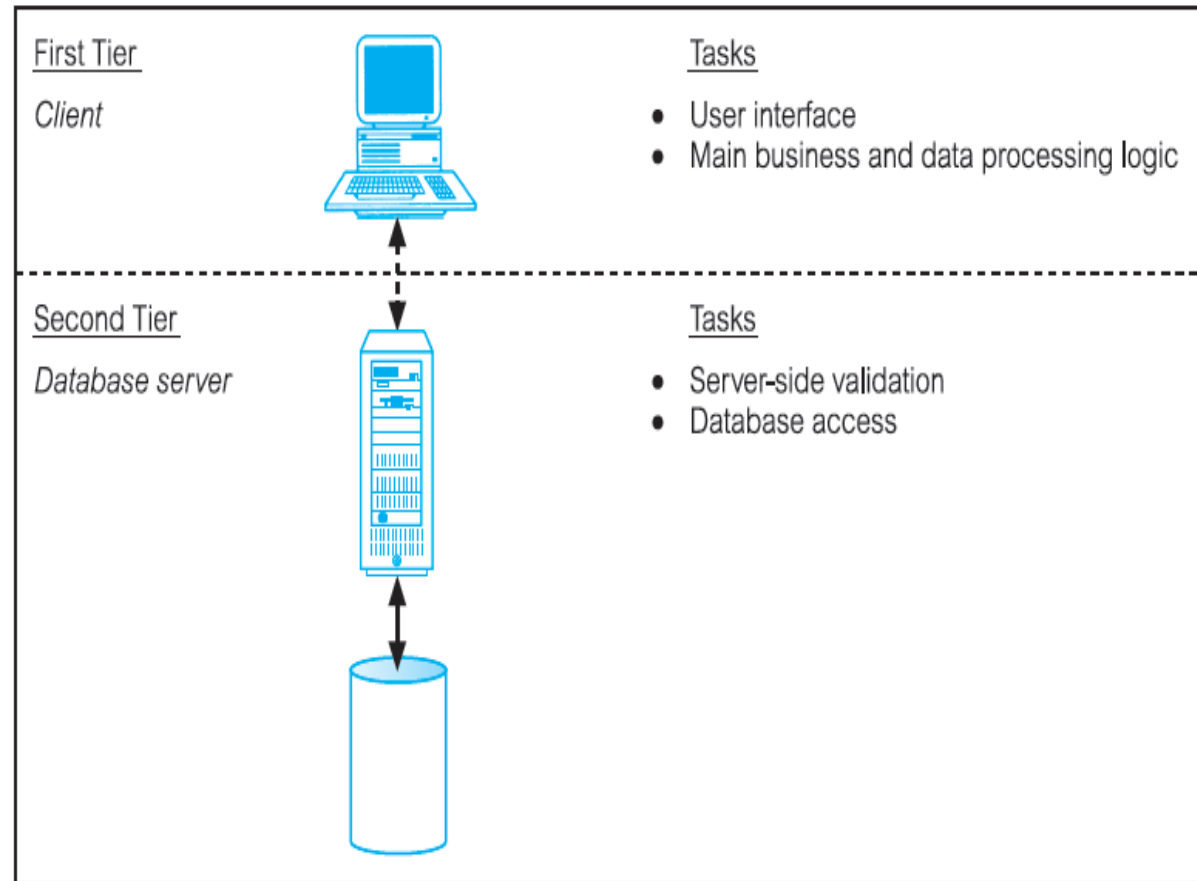
Figure 2.13

Alternative client-server topologies: (a) single client, single server; (b) multiple clients, single server; (c) multiple clients, multiple servers.

Traditional Two tier Applications

Figure 2.14

The traditional two-tier client-server architecture.



Two Tier- Problems

- The need for enterprise scalability challenged this traditional two-tier client-server model.
- A significant client-side administration overhead.

Solution is to go for Three Tier-Architecture

Three tier-Client Server Architecture

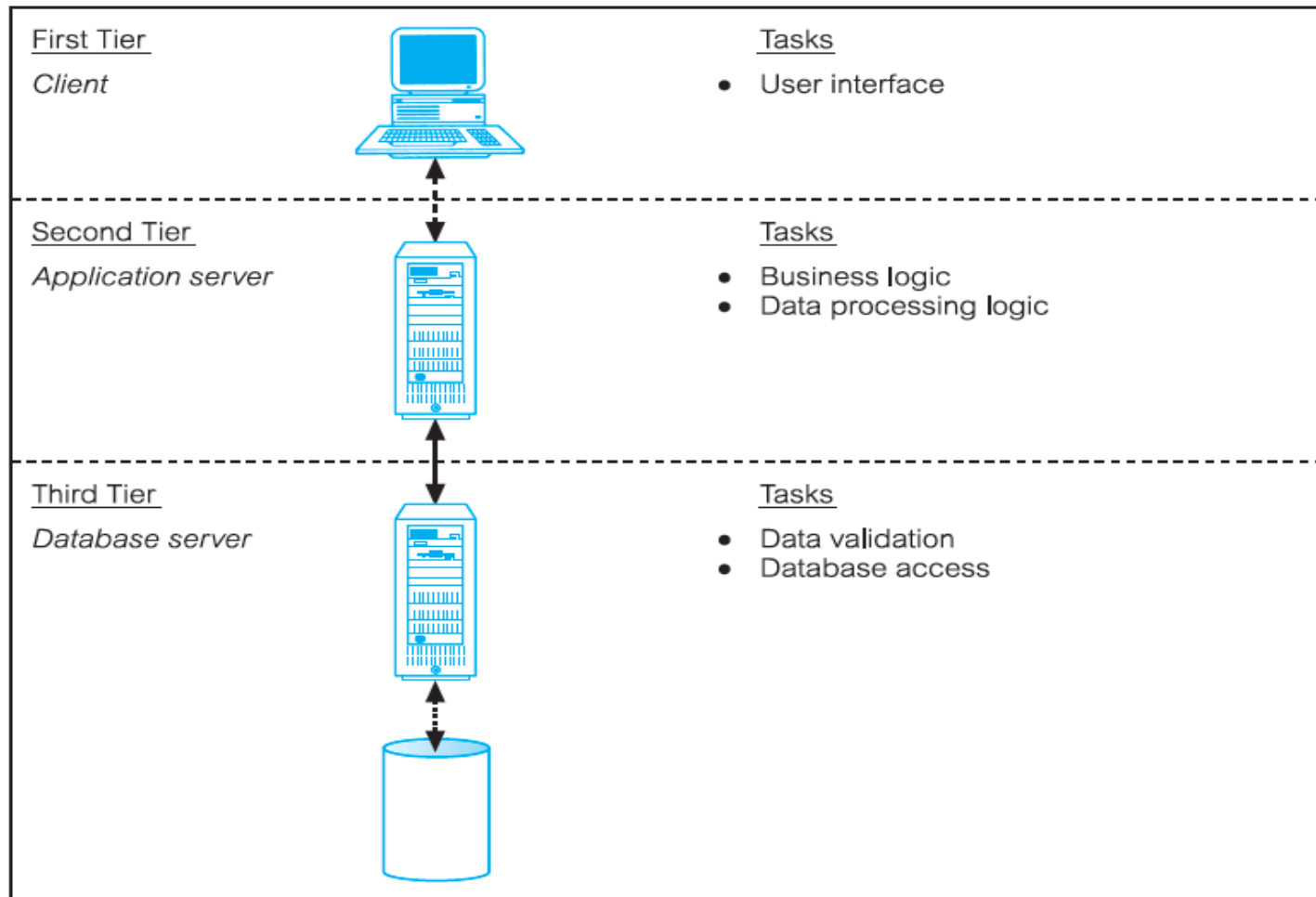


Figure 2.15

The three-tier architecture.