UNIT-1

- What is Database Management System
- Advantage of DBMS over File Processing System
- Introduction and applications of DBMS
- Purpose of database system
- Views of data
- Database system Architecture
- Data Independence
- The evolution of Data Models
- Degrees of Data Abstraction
- Database Users and DBA
- Database Languages

WHAT IS DATABASE MANAGEMENT SYSTEM[DBMS]

DBMS is a **collection of interrelated data** and a **set of programs** to **access those data**. The **collection of data** (referred as **database)** contains information relevant to an enterprise.

Primary **goal of DBMS** is to provide a **way to store and retrieve database information** that is both **convenient and efficient**.

DATABASE-SYSTEM APPLICATIONS:

1) Enterprise Information:

(i) Sales: For **customer**, **product**, and **purchase information**.

(ii)Accounting: For **payments, receipts, account balances, assets and other accounting information**.

(iii) Human resources: For **information** about employees**, salaries,payroll taxes, and benefits, and for generation of paychecks.**

(iv) Manufacturing: For management of supply chain and for **tracking production of items** in factories, **inventories** of **items** In **warehouses and stores** and **orders for items**.

(v) Online retailers: For sales data plus **online order tracking**, **generation of recommendation lists**, and maintenance of **online product evaluations.**

2)Banking and Finance:

(i) Banking: For customer information, accounts, loans, and banking transactions.

(ii)Credit card transactions: For purchases on **credit cards** and generation of **monthly statements**.

(iii)Finance: For storing information about **stocks and bonds**; for storing **real-time market data** and **automated trading**.

3) Universities: For **student information, course registrations**, and grades.

4) Airlines: For **reservations** and **schedule information**. Airlines were among the **first** to **use databases** in a **geographically distributed manner.**

5) Telecommunication: For keeping **records of calls made**, generating **monthly bills**, **maintaining balances** on prepaid **calling cards**, storing **information** about **communication networks.**

PURPOSE OF DATABASE SYSTEMS

Before DBMS were introduced, organizations usually stored information in **File-processing** systems.

Keeping organizational information in file-processing system has a number of disadvantages:

ADVANTAGE OF DBMS OVER FILE PROCESSING SYSTEM

1)Data redundancy and inconsistency:

Same information may be **duplicated in files**.

For example, if a student has a double major (music and mathematics) the address and telephone number may appear in student records in the Music department and in student records of students in Mathematics department. This redundancy leads to **higher storage and access cost**.

It may lead to **data inconsistency**.

For example, a **changed** student address may be **reflected** in **Music department records** but not in the system.

2)Difficulty in accessing data:

For example university clerks needs to find out names of all students who live within a particular postal-code area.

The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are unsatisfactory.

File-processing environments **do not allow needed data** to be **retrieved in a convenient and efficient manner**.

3)Data isolation:

Because data are **scattered** in **various files,** and **files** may be in **different formats**, writing **new application programs** to **retrieve appropriate data** is **difficult.**

4)Integrity problems:

The data values stored in database must **satisfy certain types of consistency constraints**. Suppose university **maintains** an **account for each department** and records the **balance amount** in each account. Suppose also that the university requires that the **account balance of a department may never fall below zero**. Developers **enforce these constraints** in the system by **adding appropriate code in the various application programs.** However, when new constraints are added, it is difficult to change the programs to enforce them. The **problem is compounded** when **constraints involve several data items from different files.**

5)Atomicity problems:

A computer system is subject to failure. If a failure occurs, data be restored to consistent state that existed prior to failure.

Consider a program to transfer $500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state.

Funds transfer must be atomic—**it must happen in its entirety or**

not at all. It is difficult to ensure atomicity in a conventional file-processing system.

6) Concurrent-access anomalies:

Interaction of concurrent updates is possible and may result in incon-

sistent data.

Suppose two students register concurrently, with the count at (say) 39. The two program executions may both read the value 39, and both would then write back 40, leading to an incorrect increase of only 1.

7)Security problems:

Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But in file-processing system enforcing such security constraints is difficult.

View of Data:

Provide users with an abstract view of data. That is, system hides certain details of how data are stored and maintained.

Data Abstraction:

- use **complex data structures** to represent **data** in the **database**.
- Developers **hide complexity** from users through **several levels of abstraction**, to **simplify users' interactions** with system:

(i)Physical level:

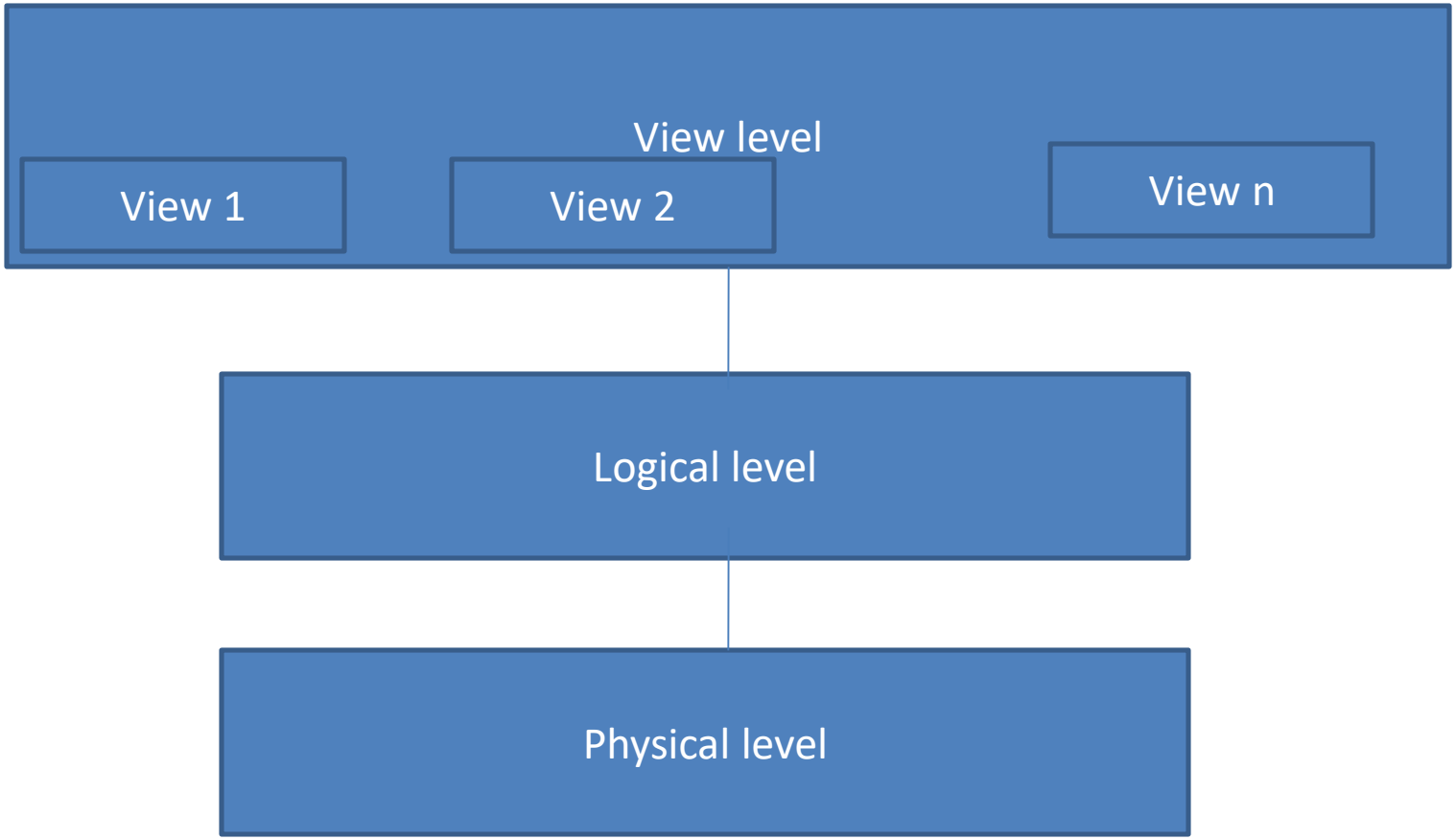
- How data are actually stored
- describes complex low-level data structures in detail

(ii) Logical level:

- **describes what data** are **stored in database** and **what relationships** exist among those data.
- **describes entire database** in terms of a **small no. of relatively simple structures.**
- user of logical level does not need to be aware of complexity. (**physical data independence)**
- Database administrators, use logical level of abstraction.

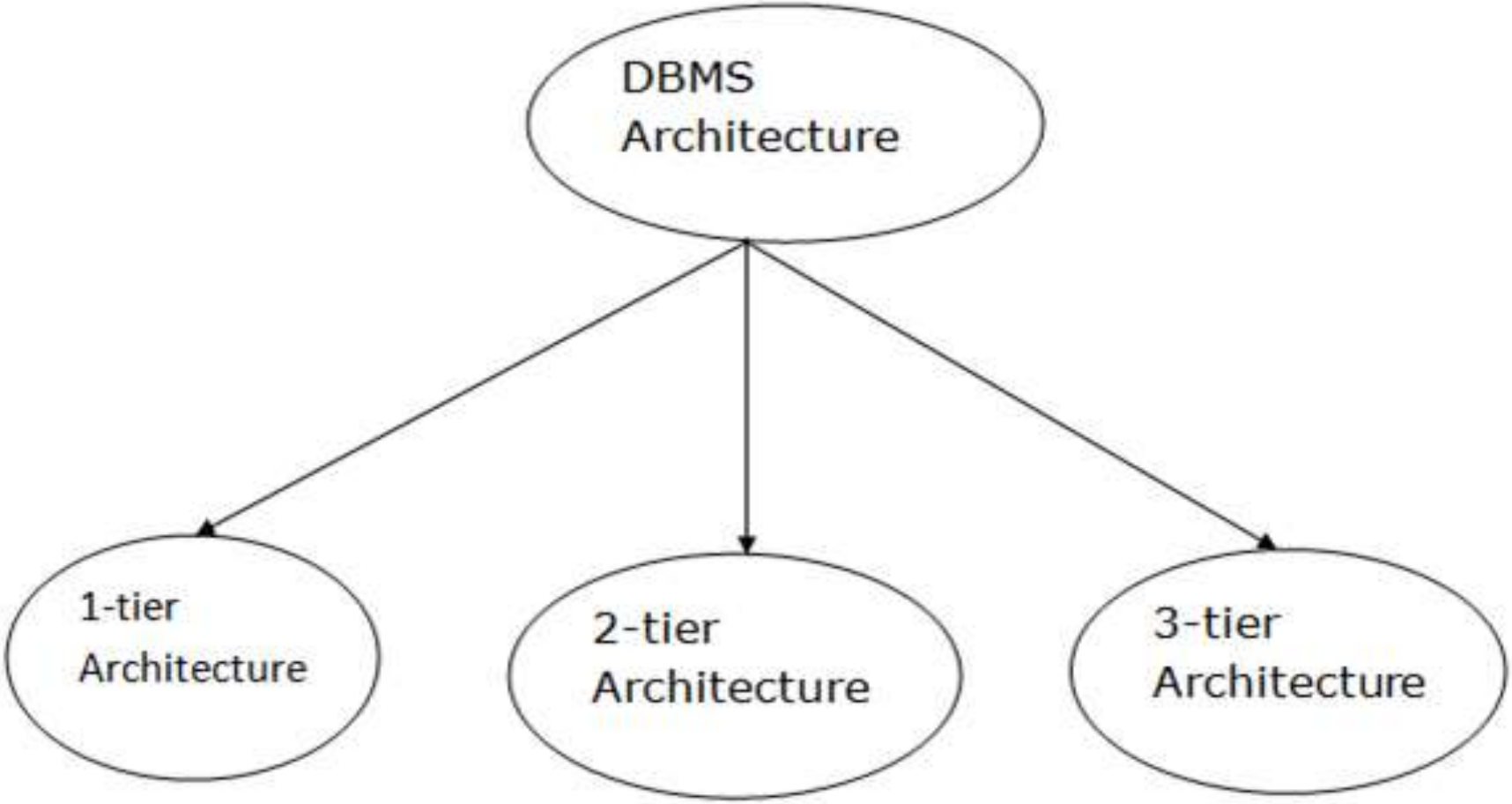
(iii)View level:

- describes only part of entire database.
- simplify user interaction with the system.
- system may provide many views for same database.



DBMS Architecture

Types of DBMS Architecture



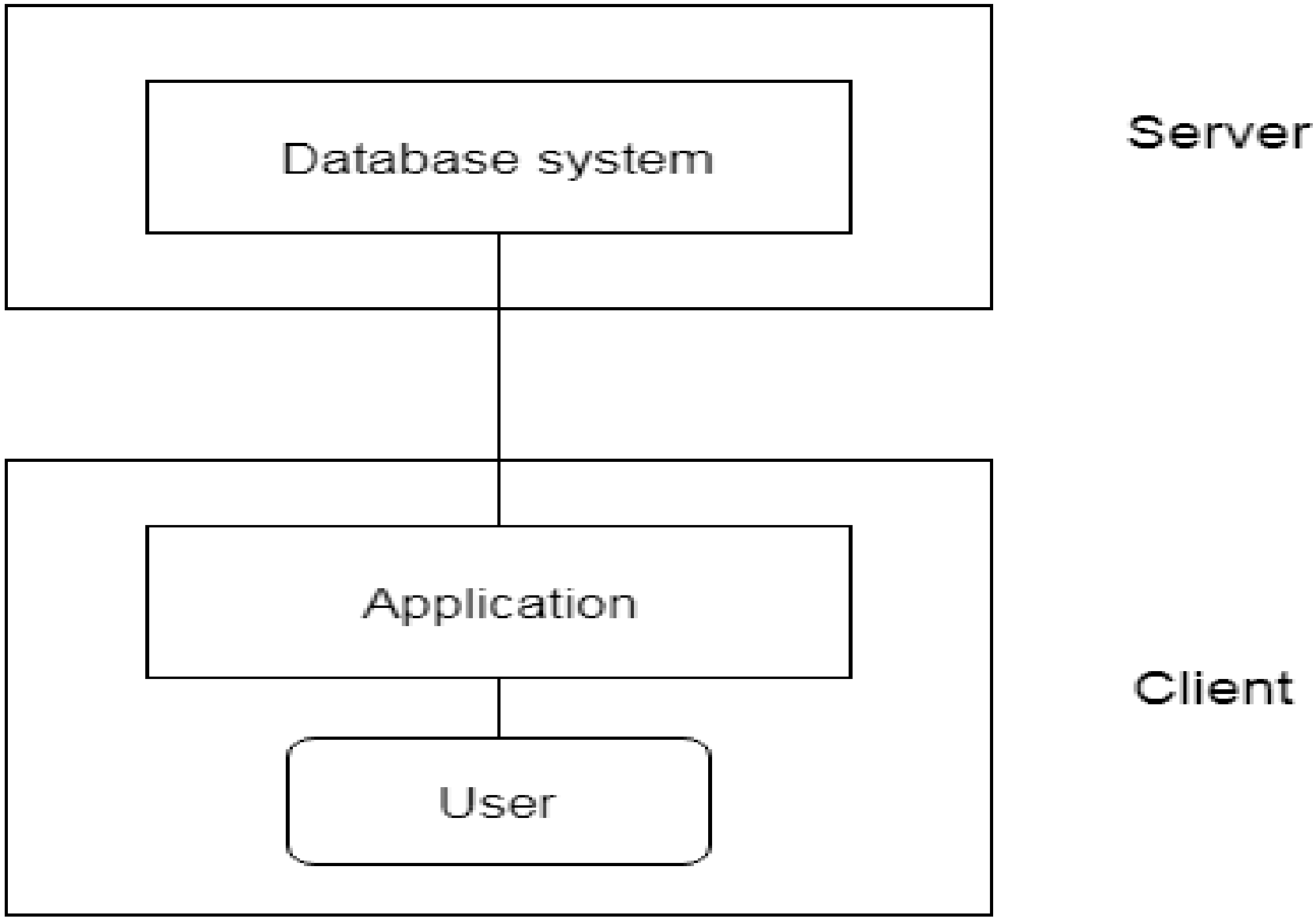
1-Tier Architecture:

- database is **directly available to user.**
- user can **directly sit on DBMS** and **uses it.**
- local application

2-Tier Architecture:

- client-server.
- applications on client end can **directly communicate with database** at server side. API's like: **ODBC**, **JDBC** are used.
- **user interfaces and application programs** are **run on client-side**.
- **server side** is responsible to provide functionalities like: **query processing and transaction management.**
- To **communicate** with the DBMS, client-side application **establishes a connection** with the server side.

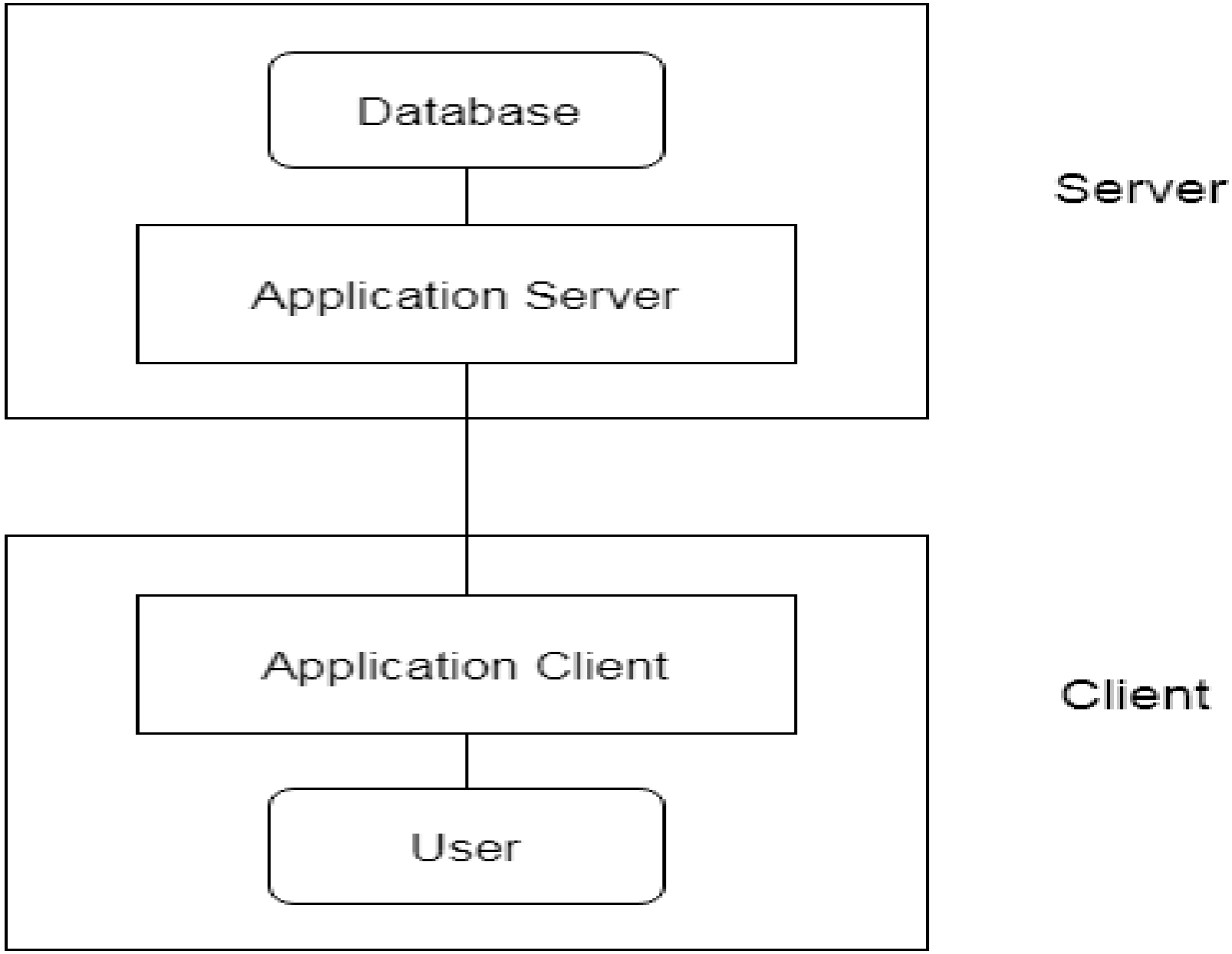
2-Tier Architecture:

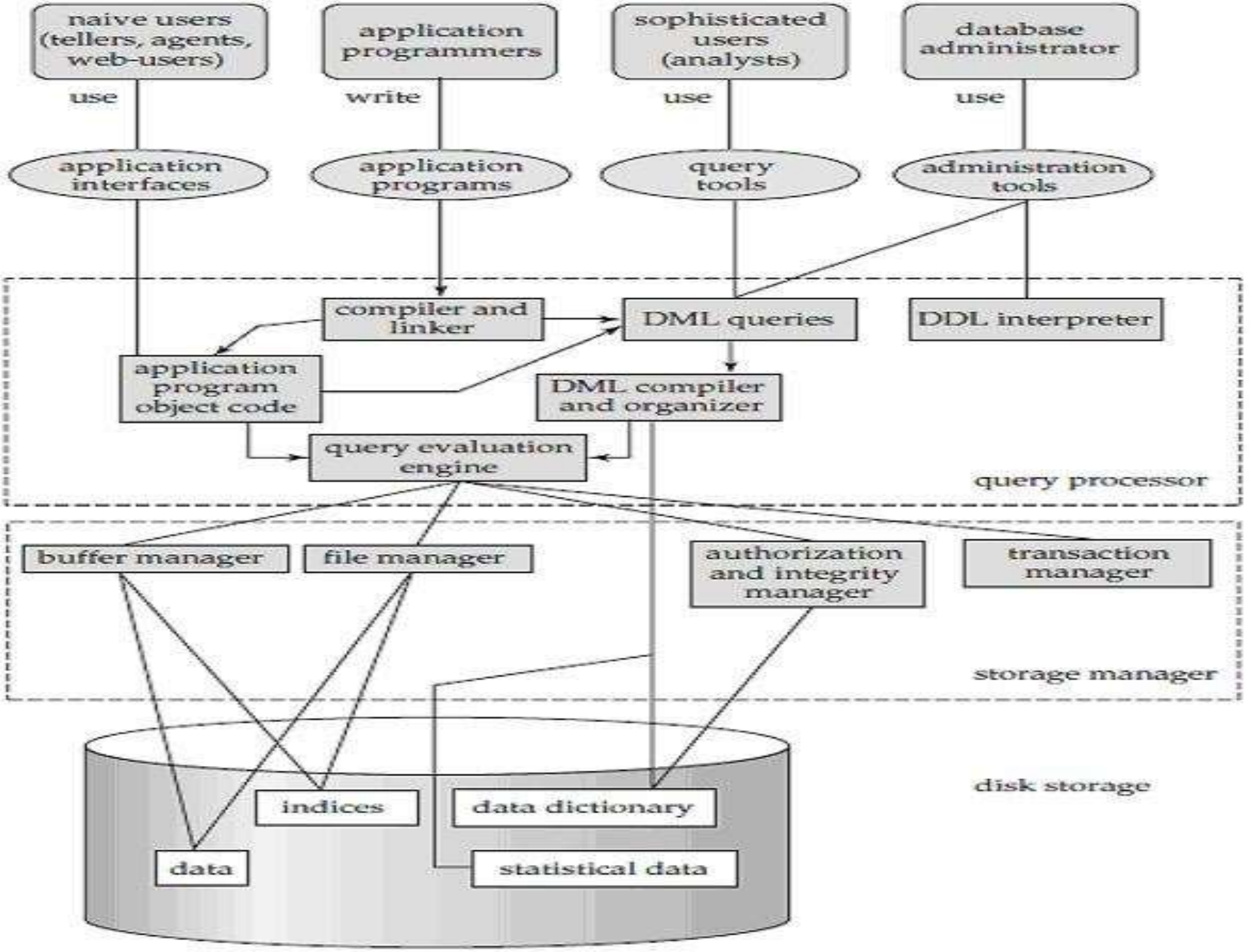


3-Tier Architecture:

- contains **another layer** between **client and server**
- client **can't directly communicate** with server.
- application on client-end interacts with **an application server** which further communicates with **the database system.**
- End user has **no idea about existence of database** beyond application server. Database also has **no idea about any other user** beyond application.
- used in case of **large web application.**

3-Tier Architecture:




naive users (tellers, agents, web-users)
application programmers
sophisticated users (analysts)
database administrator
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
indices
data dictionary
data
statistical data
disk storage

Database Architecture:

Database Users and Administrators

- DBMS Goal-> to **store & retrieve info** in database
- who **work with a database.**

Database Users and User Interfaces:

4 types of database-system users:

Type->interact with system

(1)Naıve Users:

- unsophisticated users
- interact with system by invoking application programs
- For example, a **clerk** in university needs to **add new instructor** to department A **invokes a program** called **new _hire**
- **user interface** for naıve users is **forms interface**

(2)Application programmers:

- **computer professionals** who **write** application programs.
- Rapid application development (RAD) tools

(3)Sophisticated Users:

- interact without writing programs.
- database query language
- tools ->data analysis software.
- Analysts

(4)Specialized users:

- write specialized database applications.
- computer-aided design systems
- knowledgebase and expert systems,
- systems that store graphics and audio data
- environment-modeling systems.

Database Administrator(DBA):

central control over the system

Functions:

- **Schema definition**
  - executing a set of data definition statements in **DDL.**
- **Storage structure and access-method definition**
- **Schema and physical-organization modification**
- **Granting of authorization for data access**
- **Routine maintenance**

Query Processor:

Components

- **DDL interpreter:**

  interprets DDL statements and records definitions in data dictionary.

- **DML compiler:**

  - translates DML statements ->Query evaluation engine understands.

  performs query optimization->lowest cost evaluation plan from among the alternatives.

- **Query evaluation engine:**

executes low-level instructions generated by DML compiler.

Storage Manager

- interface between the low-level data stored in database and the application programs and queries submitted to system.
- interaction with the file manager
- Translates DML statements into low-level file-system commands.
- storing, retrieving, and updating data in database.

Components :

(I)Authorization and integrity manager:

- integrity constraints and authority of users to access data.

(II) Transaction manager:

consistent (correct) state, and concurrent transaction executions without conflicting.

(III) File manager:

allocation of space on disk storage and the data structures

(IV) Buffer manager:

fetching data from disk storage into main memory and deciding what data to cache in main memory

Storage manager implements data structures :

- **Data files:**

store the database itself.

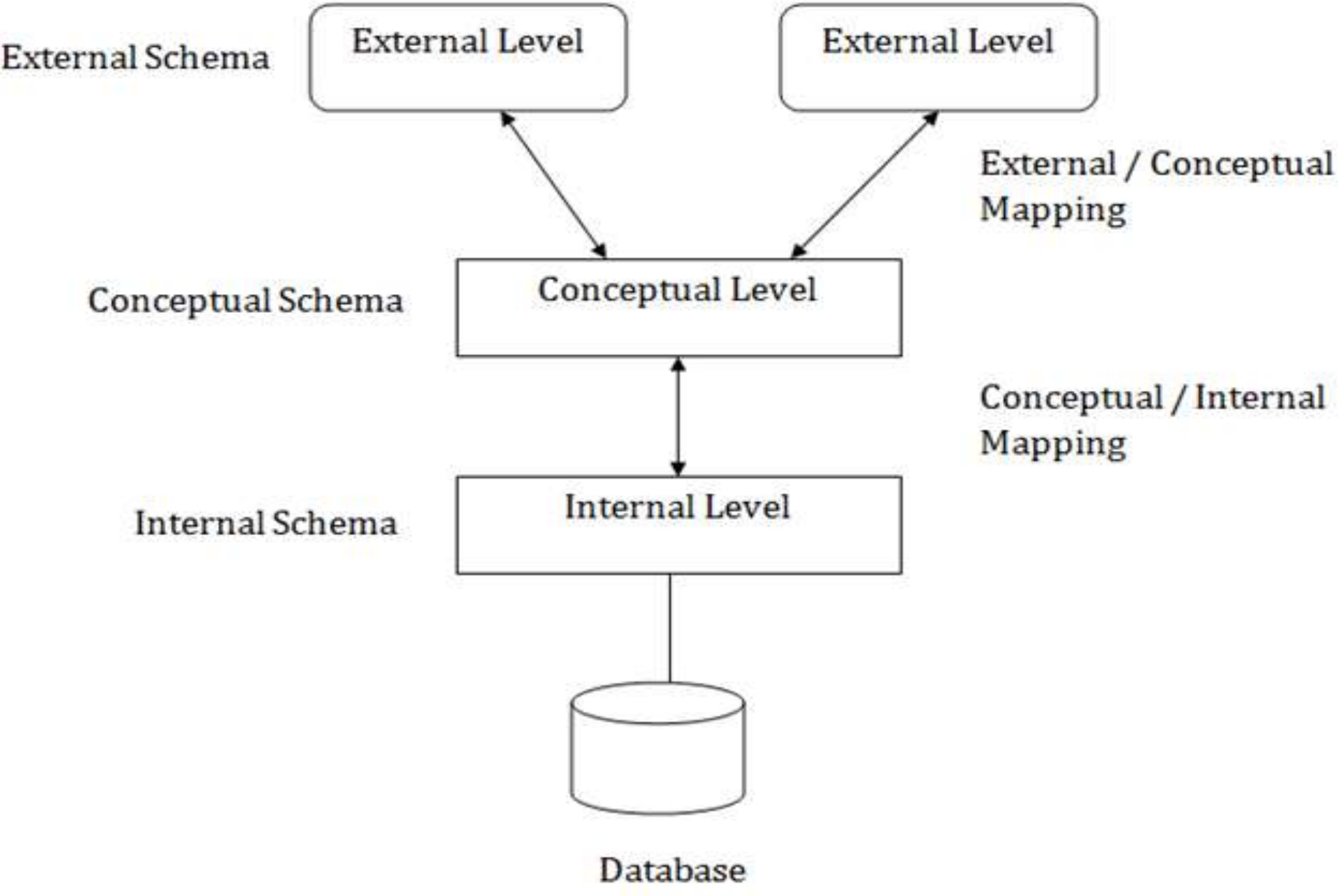
- **Data dictionary:**

stores metadata about structure of database,

- **Indices:**

fast access to data items. Like the index in textbook.

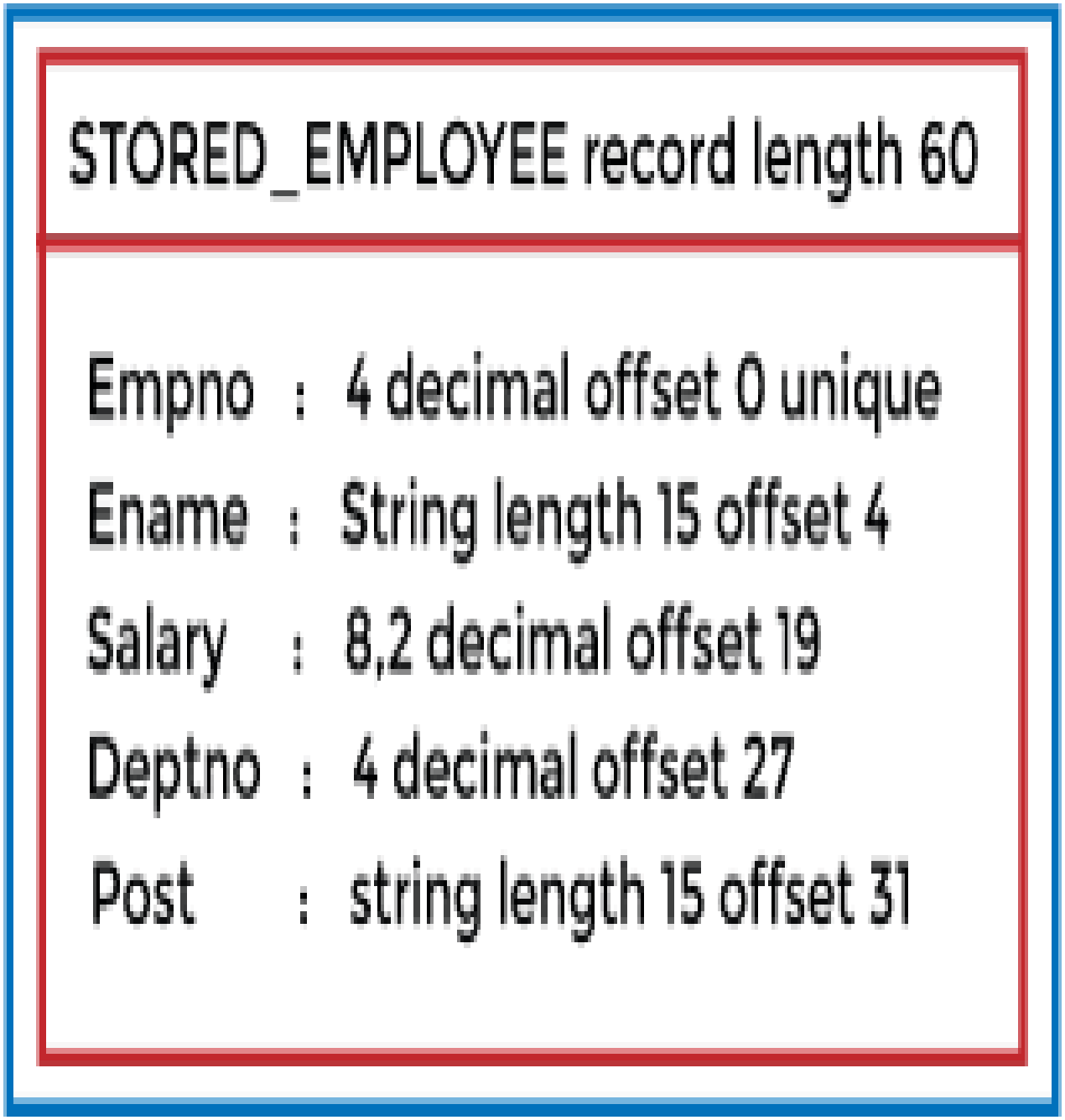
Three schema Architecture



1.Interenal Level:

- has an internal schema which describes **physical storage structure of database**.
- internal schema is also known as **physical schema**.
- uses physical data model.
- define **how data will be stored** in block.
- physical level is used to **describe complex low-level data structures**

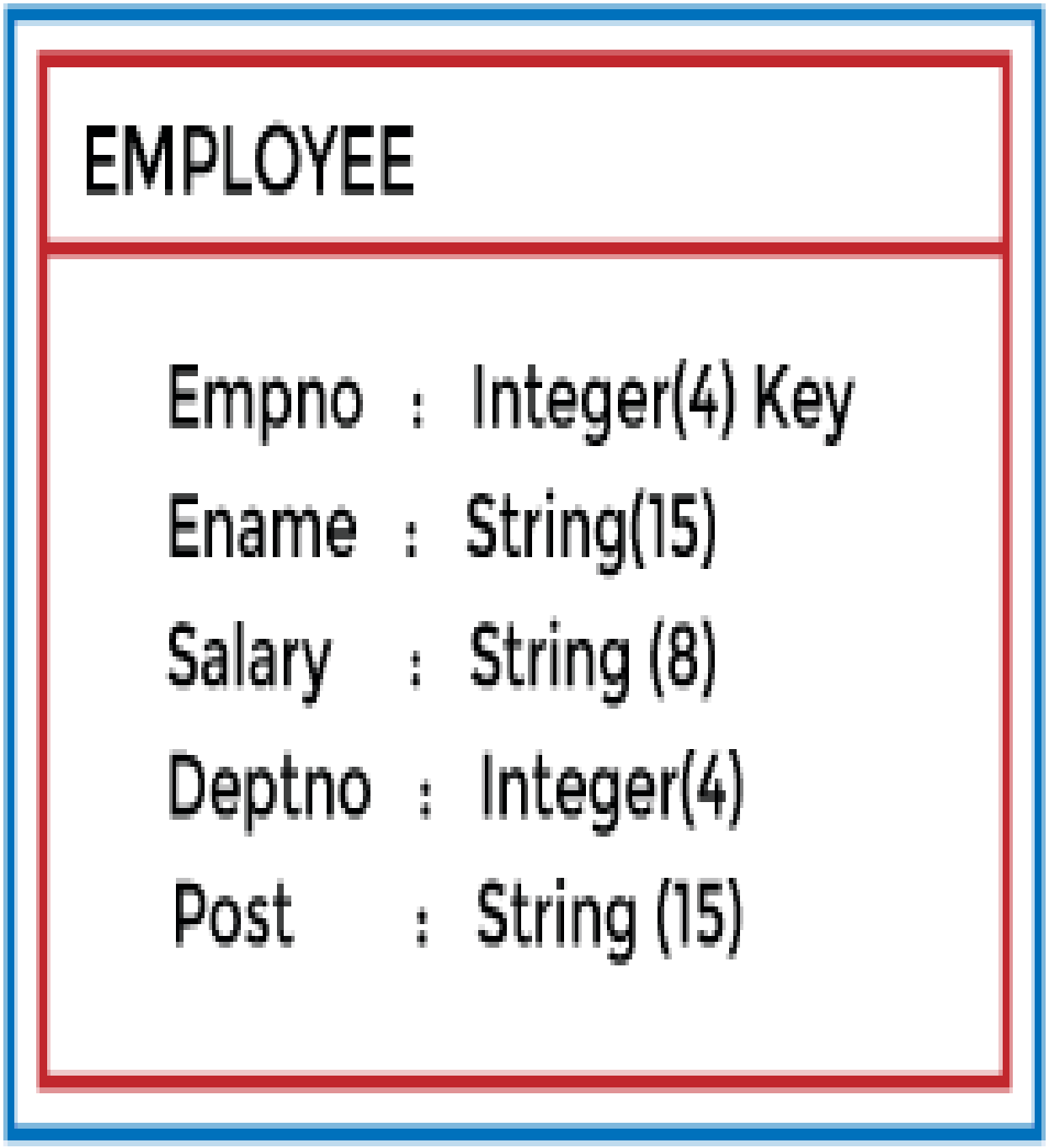
Internal view



2. Conceptual Level:

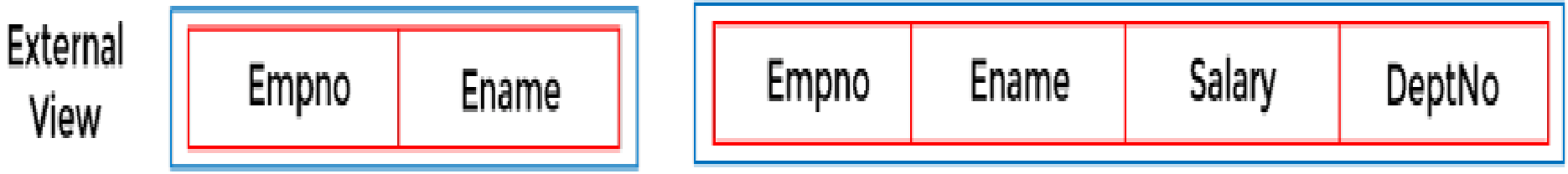
- Describes design of a database at conceptual level.
- Conceptual level is also known as **logical level.**
- Conceptual schema describes structure of whole database.
- Describes **what data are to be stored** in database and **what relationship exists** among those data.
- internal details such as implementation of data structure are **hidden.**
- Programmers and database administrators work at this level.

Global view


EMPLOYEE
Empno : Integer(4) Key
Ename : String(15)
Salary : String (8)
Deptno : Integer(4)
Post : String (15)

3. External Level

- At external level, a database contains several schemas called as subschema.
- subschema is used to describe different view of database.
- external schema is also known as view schema.
- Each view schema describes **database part** and **hides the remaining database**.
- view schema **describes the end user interaction** with database systems.



Mapping between Views

DBMS is responsible for correspondence between three types of schema. This correspondence is called Mapping.

Conceptual/ Internal Mapping:

- lies between conceptual level and internal level.
- Define correspondence b/n records & fields of conceptual level & files & data structures of internal level

External/ Conceptual Mapping:

- external/Conceptual Mapping lies between external level and Conceptual level.
- Define **correspondence between** particular **external and conceptual view.**

Data Independence

- **Modify schema** at **one level** of database system **without altering schema** at **next higher level**.

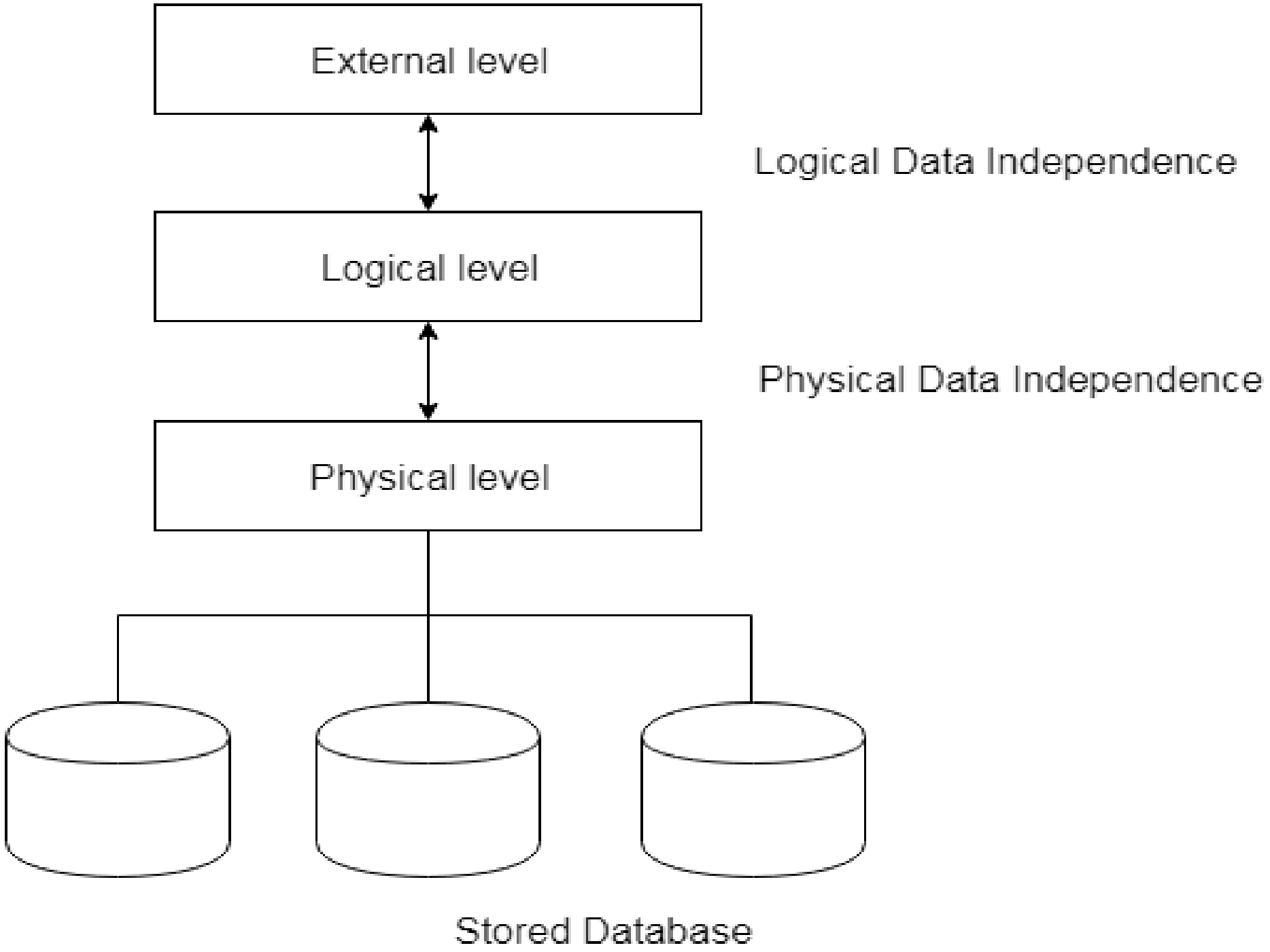
Two types of data independence:

1. Logical Data Independence:

- **change conceptual schema without** having to **change external schema**.
- used to **separate** external level from conceptual view.
- If we **do any changes** in **conceptual view** of data, then **user view of data would not be affected**.
- occurs at **user interface level.**

2. Physical Data Independence

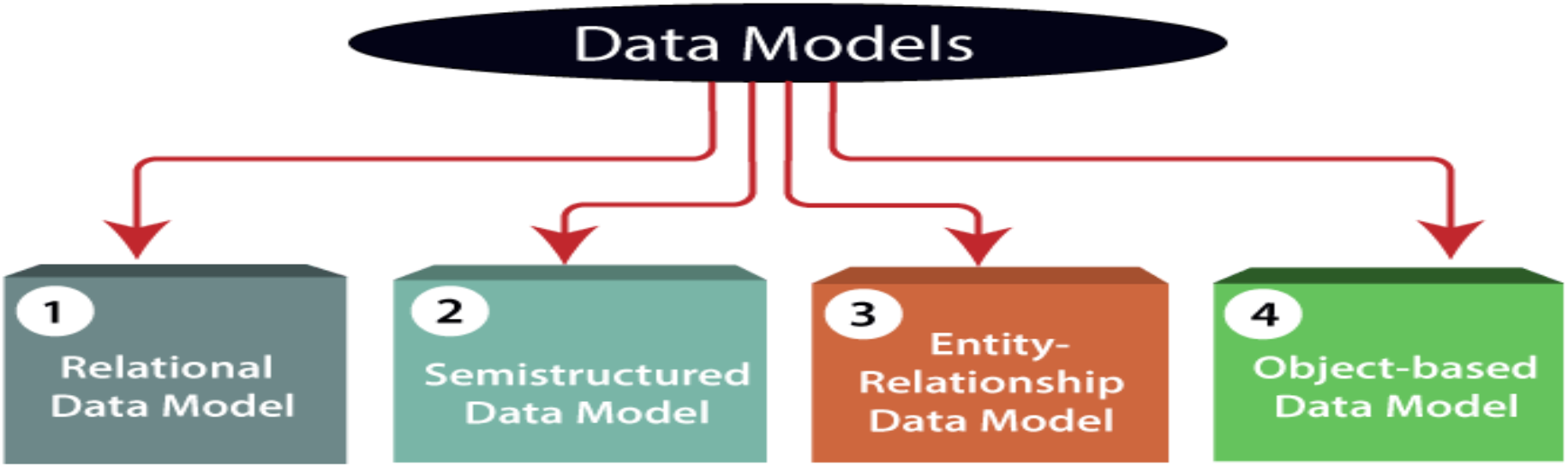
- **Change** internal schema **without** having to change **conceptual schema**.
- If we do any **changes in the storage size** of the database system server, then the **Conceptual structure of the database will not be affected**.
- used to **separate conceptual levels from the internal levels.**
- occurs at logical interface level.


External level
Logical Data Independence
Logical level
Physical Data Independence
Physical level
Stored Database

Evolution of Data Models

- **modeling** of **data** description, **data semantics**, **consistency constraints** of data.
- tools for describing **design of database** at each **level of data abstraction**.

4 data models[used for understanding structure of database]:



1) Relational Data Model:

- Designs data in the **form of rows and columns** within a **table.**
- **uses tables** for **representing data** and **in-between relationships**.
- **Tables** are also **called relations.**
- described by Edgar F. Codd, in 1969
- used by **commercial data processing applications.**

2) Entity-Relationship Data Model:

- logical **representation** of data **as objects and relationships** among them.
- objects are known as **entities. (Example-Employee)**
- relationship is known as an **association (Example-Works for)**
- designed by Peter Chen and published in 1976 papers
- used in **database designing.**
- A set of **attributes** describe **entities. (Example-Emp-ID,Emp-name)**
- set of **same type of entities** is known as an **'Entity set'**
- set of **same type of relationships** is known as **'relationship set'.**

3) Object-based Data Model:

- extension of ER model with notions of **functions, encapsulation, object identity**
- in 1980s, various database systems following the object-oriented approach were developed.
- objects are **data carrying its properties**.

4) Semistructured Data Model:

- Allows data specifications at places where individual **data items of the same type** may have **different attributes sets.**
- **XML,** is widely used for representing the **semistructured data.**
- XML was initially designed for including **markup information** to the text document
- application **in the exchange of data.**

Data model Schema and Instance:

- data which is stored in database **at a particular moment of time** is called an **instance of database**.
- overall **design of database** is called **schema.**
- database schema is **skeleton structure of database**. It represents **logical view of entire database**.
- schema contains **schema objects** like **table, foreign key, primary key, views, columns, data types, stored procedure**
- schema can be represented by **using visual diagram**. That diagram shows the **database objects** and **relationship** with each other.
- A database schema is **designed** by **database designers** to help programmers whose software will interact with database. The **process of database creation** is called **data modeling.**

For example, in given figure, database changes whenever we **add a new grade or add a student**. The **data** at a **particular moment of time** is called **instance of database.**

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

DEGREE OF ABSTRACTION:

Data Abstraction:

- use **complex data structures** to represent **data** in the **database**.
- Developers **hide complexity** from users through **several levels of abstraction**, to **simplify users' interactions** with system:

(i)Physical level:

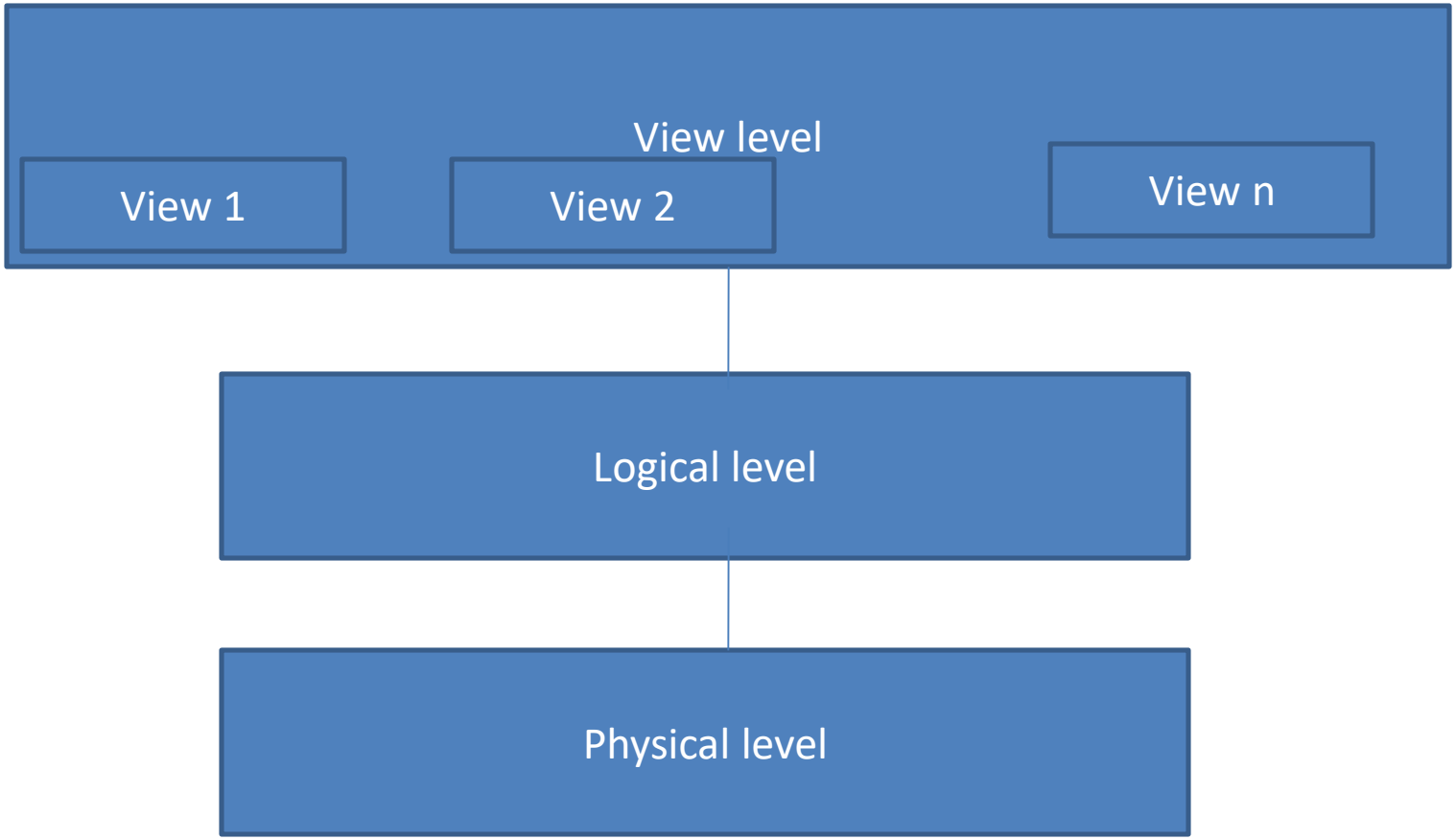
- How data are actually stored
- describes complex low-level data structures in detail

(ii) Logical level:

- **describes what data** are **stored in database** and **what relationships** exist among those data.
- **describes entire database** in terms of a **small no. of relatively simple structures.**
- user of logical level does not need to be aware of complexity. **(physical data independence)**
- Database administrators, use logical level of abstraction.

(iii)View level:

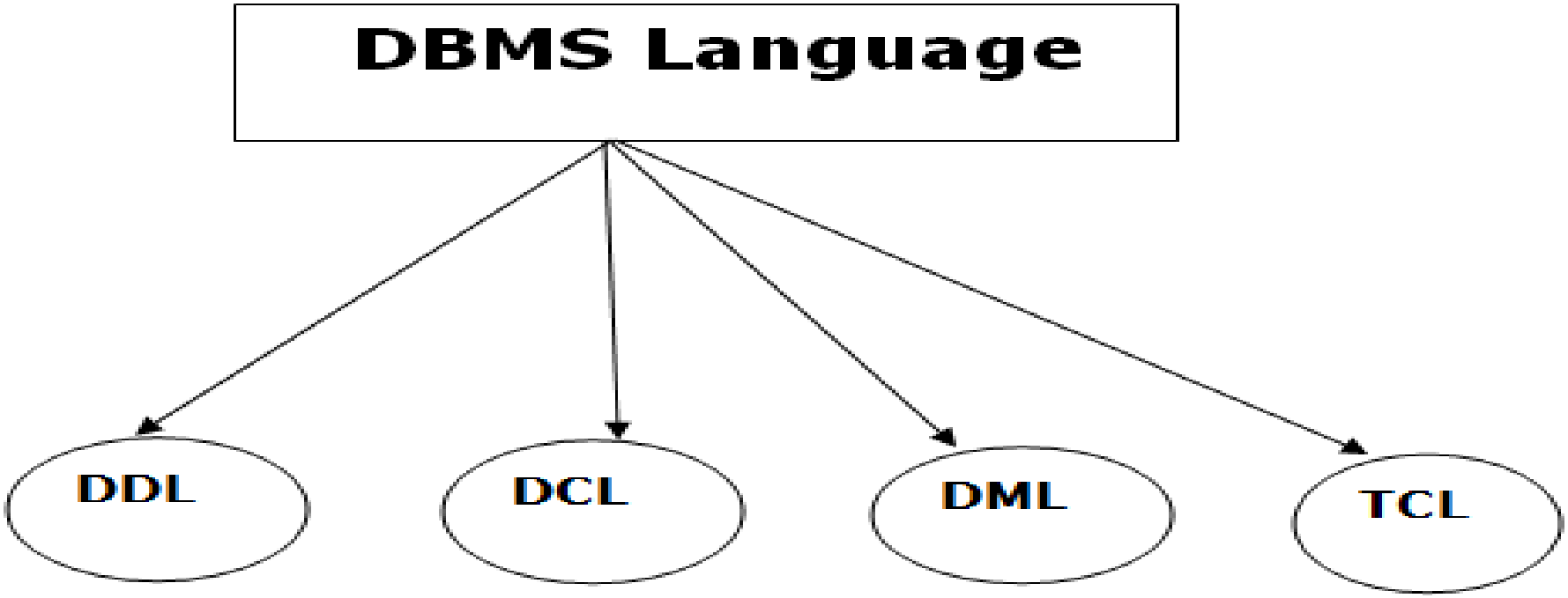
- describes only part of entire database.
- simplify user interaction with the system.
- system may provide many views for same database.



Database Language:

- A DBMS has appropriate languages and interfaces to express **database queries and updates**.
- Database languages can be used **to read, store and update the data** in **database.**

Types of Database Language:



1. Data Definition Language[DDL]:

- used to define **database structure or pattern.**
- to **create schema, tables, indexes, constraints**

Create skeleton of the database.

- used to store **information of metadata** like no. of tables and schemas, their names, indexes, columns in each table, constraints

Commands:

Create: used to **create objects** in database.

Alter: used to **alter structure** of database.

Drop: used to **delete objects** from database.

Truncate: used to **remove all records** from table.

Rename: used to **rename an object.**

Comment: used to comment on data dictionary.

2. Data Manipulation Language[DML]:

- used for **accessing and manipulating data** in database.
- Handles **user requests.**

Commands:

Select: used to **retrieve data** from a database.

Insert: used to **insert data** into a table.

Update: used to **update existing data** within a table.

Delete: used to **delete all records** from a table.

Merge: performs **UPSERT operation**, i.e., **insert or update** operations.

Call: used to **call** structured query language or Java subprogram.

Explain Plan: It has parameter of **explaining data.**

Lock Table: It controls **concurrency.**

3. Data Control Language[DCL]:

- used to retrieve stored or saved data.
- DCL execution is transactional. It also has rollback parameters.

Commands:

Grant: used to give user access privileges to database.

Revoke: used to take back permissions from user.

operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

4. Transaction Control Language[TCL]:

- used to run changes made by DML statement.
- TCL can be grouped into a logical transaction.

Commands:

Commit: used to save transaction on database.

Rollback: used to restore database to original since last Commit.