Architecture of DBMS

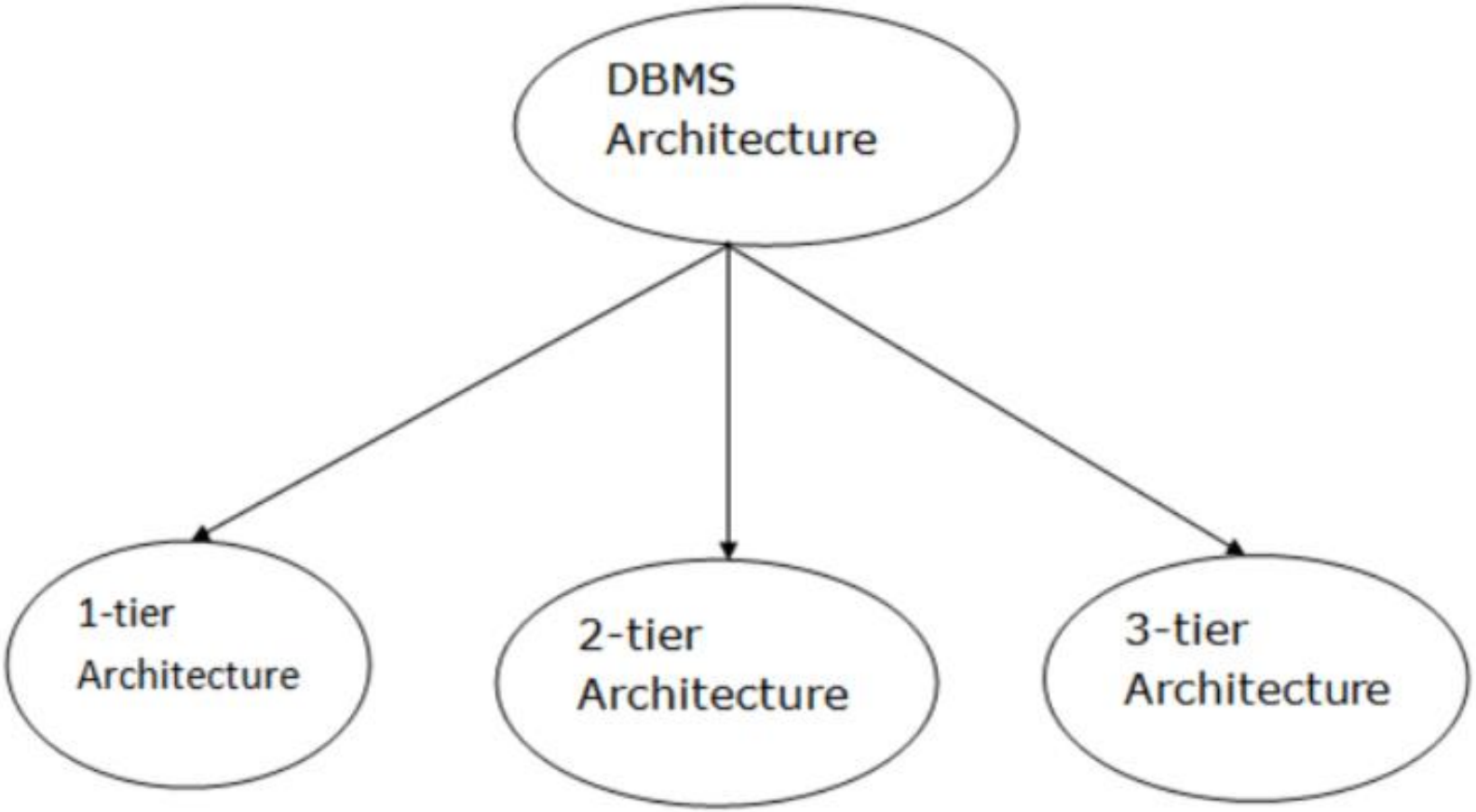
Outline

- DBMS architecture
- Data Independence
- Components of DBMS

What is Database Architecture ?

- It is a representation of DBMS design.
- It helps to design, develop, implement, and maintain the database management system.
- A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered.

Types of DBMS Architecture



1-Tier Architecture

- *Simplest* architecture of Database.
- Client, Server, and Database all *reside on the same machine.*
- Example: Anytime when you install a Database in your system and access it to practice SQL queries.
- Hardly used in production.

Single Tier Architecture

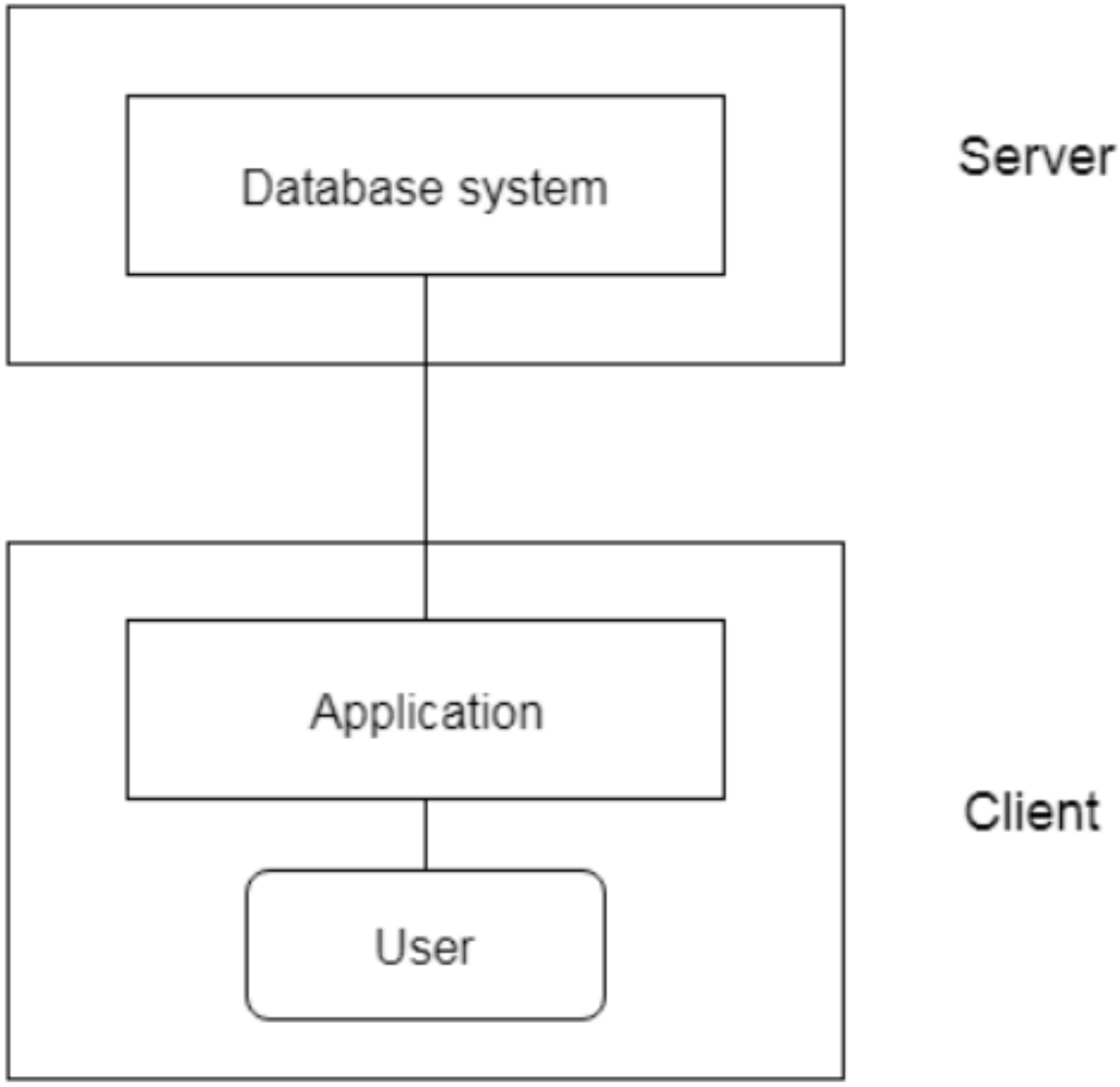
1 Tier Architecture Diagram

Early Two Level Architecture

- Early proposal for a standard terminology and general architecture for database systems was produced in 1971 by the DBTG (Data Base Task Group)
- Appointed by the Conference on Data Systems and Languages (CODASYL, 1971).
- Recommend two level approach with a system view called *SCHEMA* and user view called *SUB-SCHEMA*

2-Tier Architecture

- Same as basic client-server architecture .
- Here the applications on the *client-side* can directly communicate with the database at the *server-side*. For this interaction, API's like: **ODBC**, **JDBC** are used.
- User interfaces and application programs run on the *client-side*.
- *Server-side* provides the functionalities like: query processing and transaction management.
- For communication, client-side application establishes a connection with the server side.


Database system
Server
Application
Client
User

3 TIER ARCHITECTURE

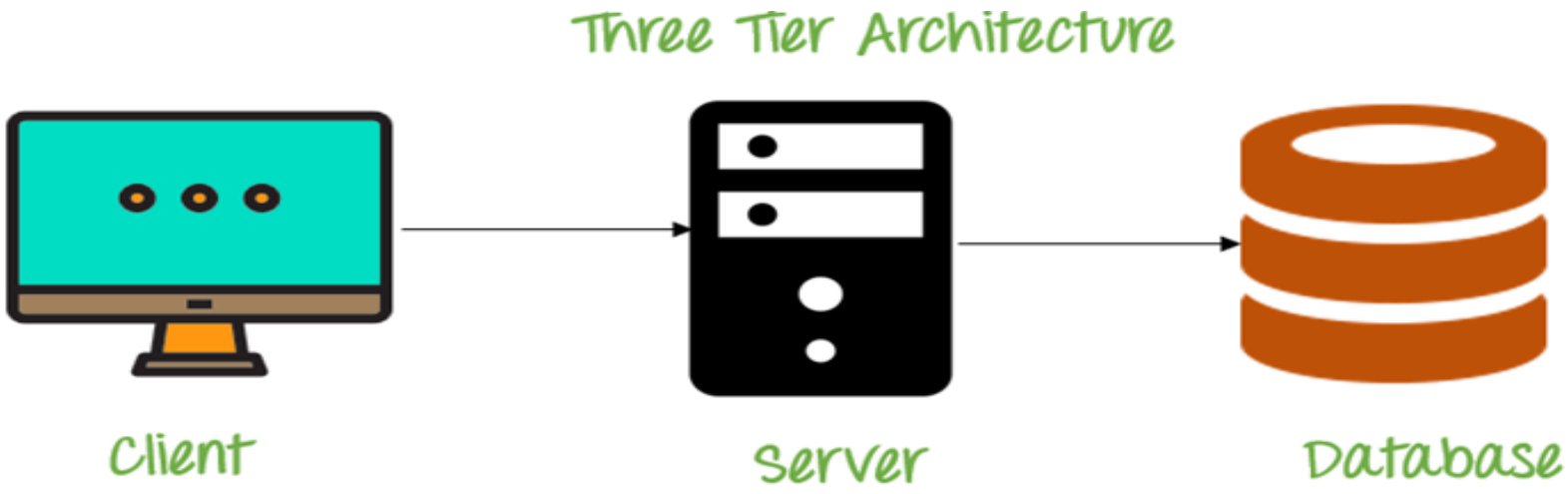
- Most popular client server architecture in DBMS.
- Development and maintenance of functional processes, logic, data access, data storage, and user interface is done independently as separate modules.
- Contains another layer between the client and server.
- Client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- Used in case of large web application.

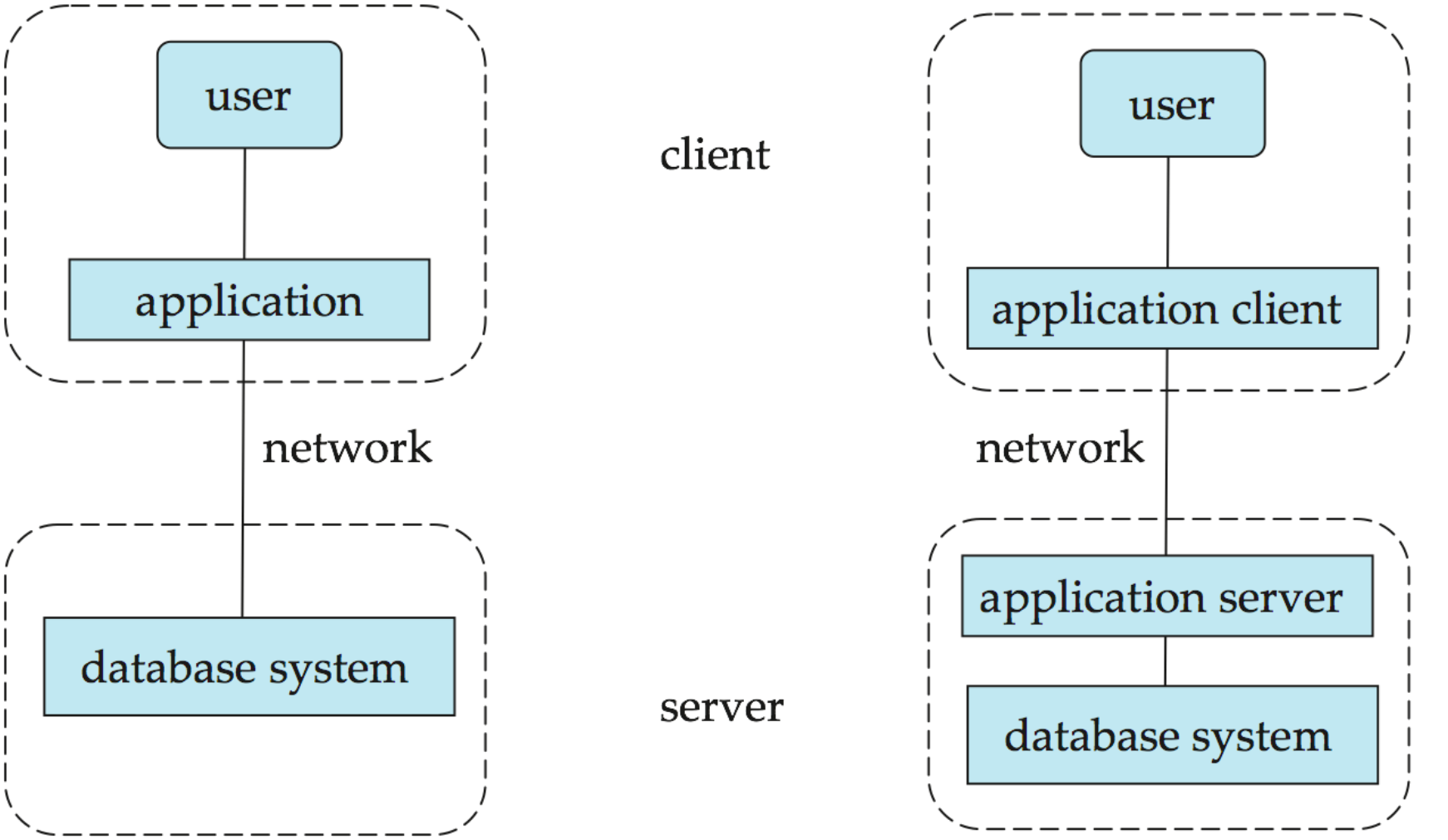
- 3-Tier database Architecture design is an extension of the 2-tier client-server architecture.
- A 3-tier architecture has the following layers:

i. Presentation layer (your PC, Tablet, Mobile, etc.)

ii. Application layer (server)

iii. Database Server





(a) Two-tier architecture

(b) Three-tier architecture

ANSI Three Level Architecture

ANSI Three Level Architecture include :

- **External Level: USERS>, CLIENTS……..**
- **Conceptual Level**
- **Internal Level**

- Main objective of three level architecture: to separate each user's view of the database from the way the database is physically represented.

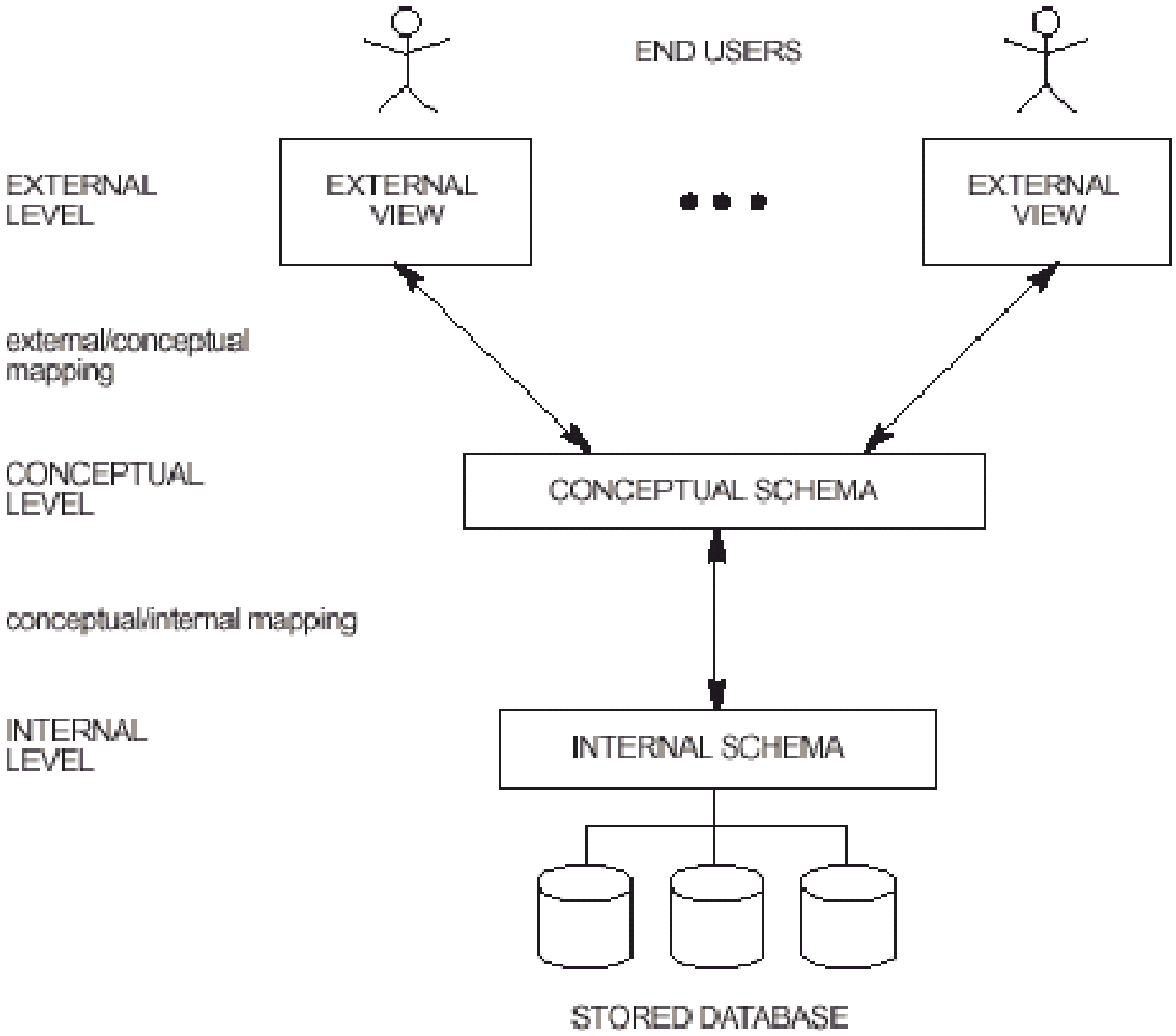
But why separate??????

There are reasons for why this separation is desirable:

a) Each user **should be able to access the same data** but have **different customized view** of the data.

b) Users **should not have to deal directly with physical database storage details**. In other words, a user's interaction with the database should be independent of storage consideration.

c) DBA should be able to **change the database storage** structures **without affecting the user's view**.


END USERS
EXTERNAL LEVEL
EXTERNAL VIEW
EXTERNAL VIEW
external/conceptual mapping
CONCEPTUAL LEVEL
CONCEPTUAL SCHEMA
conceptual/internal mapping
INTERNAL LEVEL
INTERNAL SCHEMA
STORED DATABASE

EXTERNAL LEVEL or VIEW LEVEL

- It is the users' view of the database.
- Describes that part of the database that is relevant to each user.
- It is the one level which is closest to end user's. This level deals with the way in which individual users view data.
- Individual users are given different views according to the user's requirement.

- A view involves only those portions of the database which are concerned to a user.
- Therefore, same database can have different views for different users.
- It is also known as ***VIEW LEVEL*** , it may have different representation of the same data.
- For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).

CONCEPTUAL LEVEL or LOGICAL LEVEL

- It is the community view of the database.
- Describes what data is stored in the database and the relationships among the data.
- Contains the logical structure of the entire database as by the DBA.
- It is a complete view of the data requirements of the organization that is independent of any storage considerations.

It represents:

I. All entities, their attributes, and their relationships;

II. The constraints on the data;

III. Security and integrity information.

- An ***ENTITY*** is an object whose information is stored in database.
- An ***ATTRIBUTE*** is a characteristic of interest about an entity.
- Eg: In student database student is entity , regno, name, class, address, etc are attributes.

- The conceptual level supports each external view ,in that any data available to a user must be contained in or derivable from , the conceptual level.
- However, this level ***must not*** contain any storage dependent details.
- For instance , the description of an entity should contain only data type of attributes and their length(char name[20]) but not any storage considerations such as number of bytes occupied.
- This level is also known as the ***LOGIC LEVEL.***

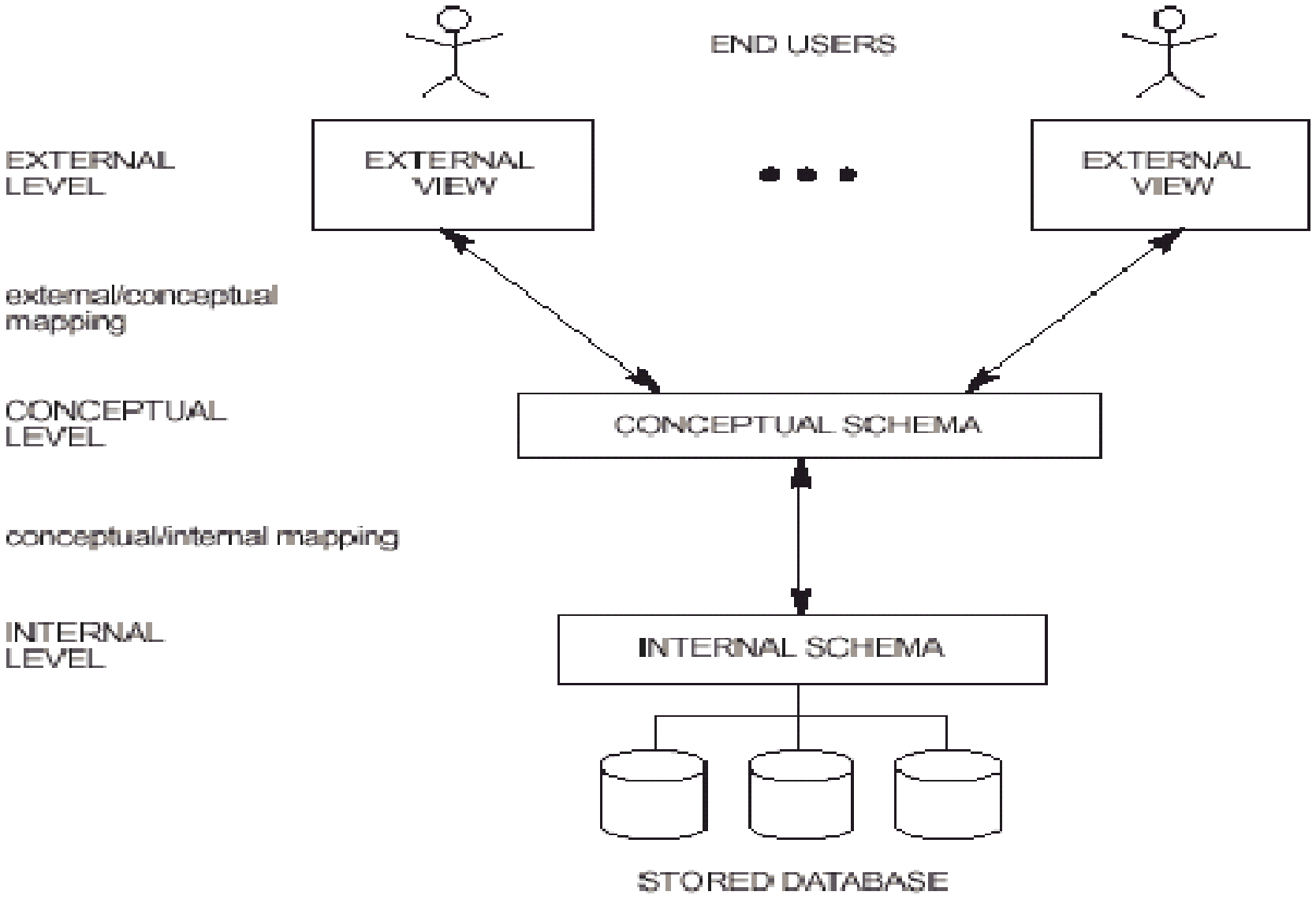
INTERNAL LEVEL or STORAGE LEVEL

- It is the physical representation of the database on the computer.
- Describes how the data is stored in the database.
- This level covers the physical implementation of the database to achieve optimal runtime performance and storage space utilization.
- It covers the data structure and file organization used to store data on storage devices.
- It interfaces with the operating system access methods to place data on the data on storage device.

- The internal level is concerned with:

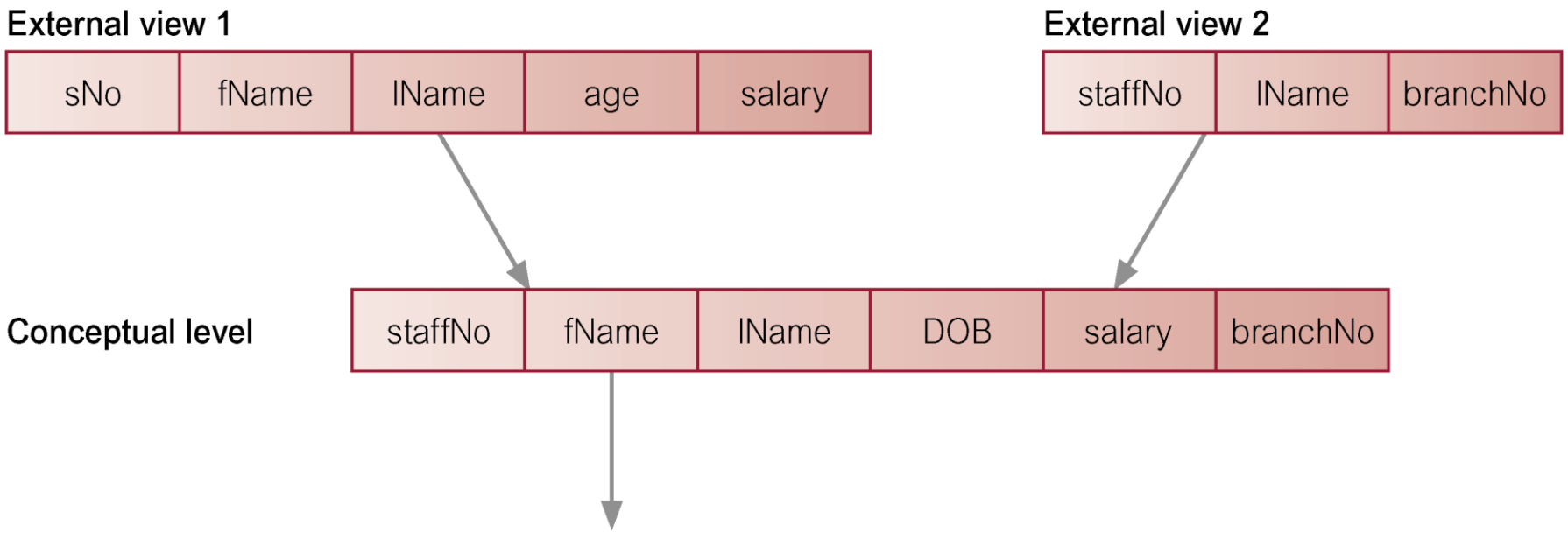
1. Storage space allocation for data and indexes;
2. Record descriptions for storage;
3. Record placement;
4. Data compression and data encryption.

- There will be only one conceptual view , consisting of abstract representation of the database in its entirely.
- Similarly, there will be only one internal or physical view , representing the total database as it is physically stored


END USERS
EXTERNAL LEVEL
EXTERNAL VIEW
EXTERNAL VIEW
external/conceptual mapping
CONCEPTUAL LEVEL
CONCEPTUAL SCHEMA
conceptual/internal mapping
INTERNAL LEVEL
INTERNAL SCHEMA
STORED DATABASE

SCHEMA

- The overall description of the database is called the *Database Schema.*
- A *SCHEMA* is defined as an outline or a plan that describes the records and relationships existing at the particular level.
- The schema is sometimes called the *INTENSION OF THE DATABASE,* while an instance is called an *EXTENSION* (or state) of the database.



Internal level

```
struct STAFF {
        int staffNo;
        int branchNo;
        char fName [15];
        char lName [15];
        struct date dateOf Birth;
        float salary;
        struct STAFF *next;                    /* pointer to next Staff record */
};
index staffNo; index branchNo;                 /* define indexes for staff */
```

TYPES OF SCHEMA

- There are 3 types of schema in the database corresponding to each data view of database.
- A schema is defined as an outline or a plan that describes the record and relationships existing at the particular level.

I. The external view is described by means of schema called **EXTERNAL SCHEMA** that corresponds to different views of the data.

II. The conceptual view is defined by **CONCEPTUAL SCHEMA** which describes all the entities , attributes and relationship together with integrity constraints.

III. Internal view is defined BY **INTERNAL SCHEMA** which is complete description of the internal model, containing definition of stored records , the methods of representation ,the data fields and the indexes used.

Mapping between Views

A. External/Conceptual Mapping

A mapping between the external and conceptual views gives the correspondence among the records and the relationships of the external and conceptual views.

B. Conceptual/Internal Mapping

Conceptual schema is related to the internal schema by the 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 conceptual schema.

DATA INDEPENDANCE

Introduction

- *DATA INDEPENDENCE* is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Data independence helps you to keep data separated from all programs that make use of it.
- There are 2 kinds of data Independence:

i) Logical data Independence

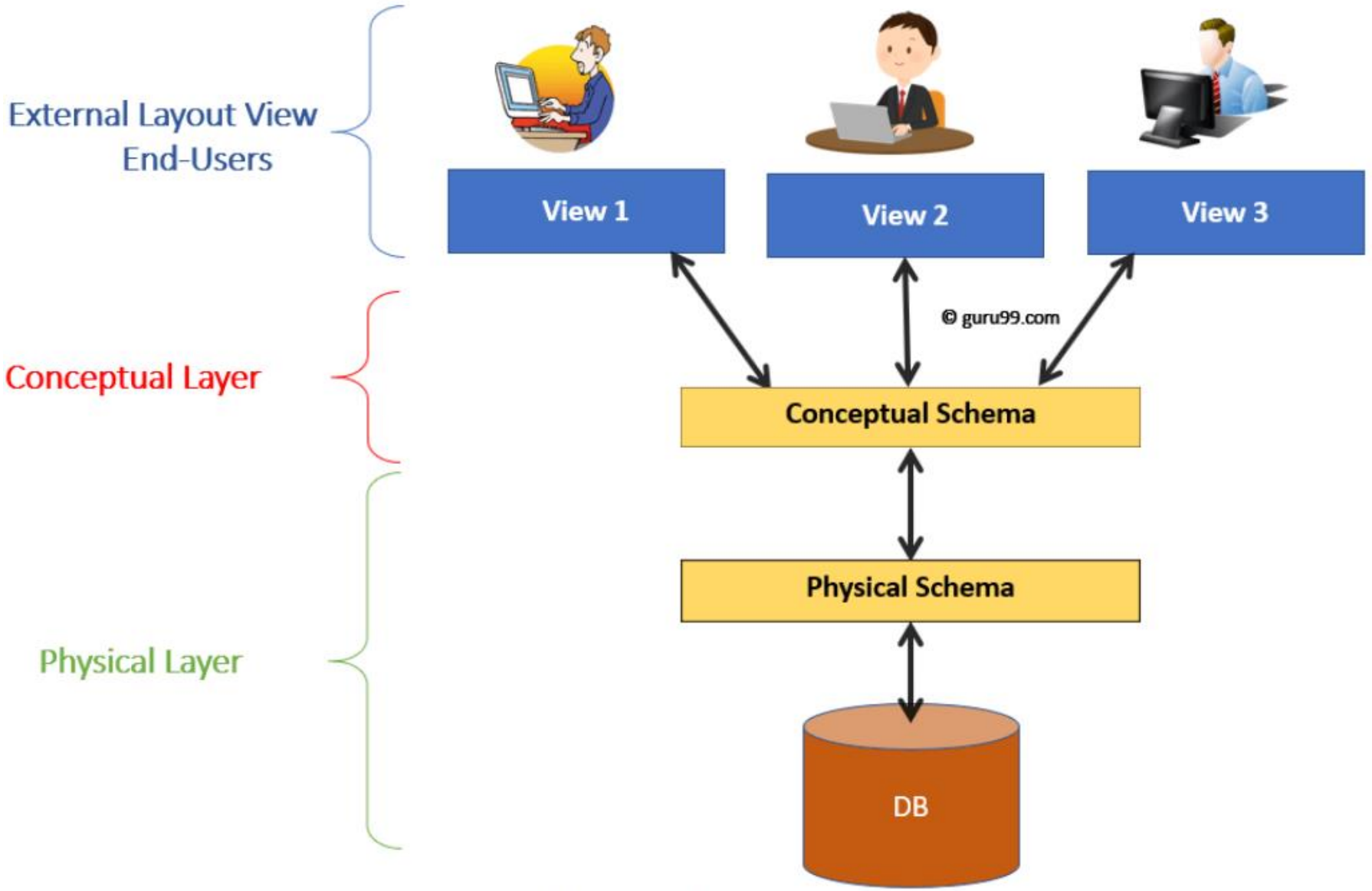
ii) Physical Data Independence.

- Before we learn Data Independence, a refresher on Database Levels is important.
- The database has 3 levels as shown in the diagram in the next slide:

I. Physical/Internal

II. Conceptual

III. External



Levels of DBMS Architecture Diagram

- Consider an Example of a University Database. At the different levels this is how the implementation will look like:

Type of Schema	Implementation
External Schema	**View 1**: Course info(cid:int,cname:string) **View 2**: studeninfo(id:int. name:string)

Conceptual Shema	`Students(id: int, name: string, login: string, age: integer)`
	`Courses(id: int, cname:string, credits:integer)`
	`Enrolled(id: int, grade:string)`

Physical Schema	• Relations stored as unordered files. • Index on the first column of Students.

LOGICAL DATA INDEPENDENCE

- It indicates that the conceptual schema can be changed without affecting the existing external schemas.
- The change would be absorbed by the mapping between the external and conceptual levels.
- It also insulates application programs from operations such as combining two records into one
- This would require a change in the external/conceptual mapping so as to leave the external view unchanged

- **Examples of changes under Logical Data Independence**

Due to Logical independence, any of the below change will not affect the external layer:

i. Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs

ii. Merging two records into one

iii. Breaking an existing record into two or more records

PHYSICAL DATA INDEPENDENCE

- It indicates that the physical storage structures or devices could be changed without affecting conceptual schema.
- The change would be absorbed by mapping between the conceptual and internal levels.
- Physical data independence criterion requires that the conceptual level does not specify storage structures or access methods used to retrieve the data from physical storage medium.

- **Examples of changes under Physical Data Independence**

Due to Physical independence, any of the below change will not affect the conceptual layer:

i. Using a new storage device like Hard Drive or Magnetic Tapes.

ii. Modifying the file organization technique in the Database.

iii. Switching to different data structures.

iv. Changing the access method.

v. Modifying indexes.

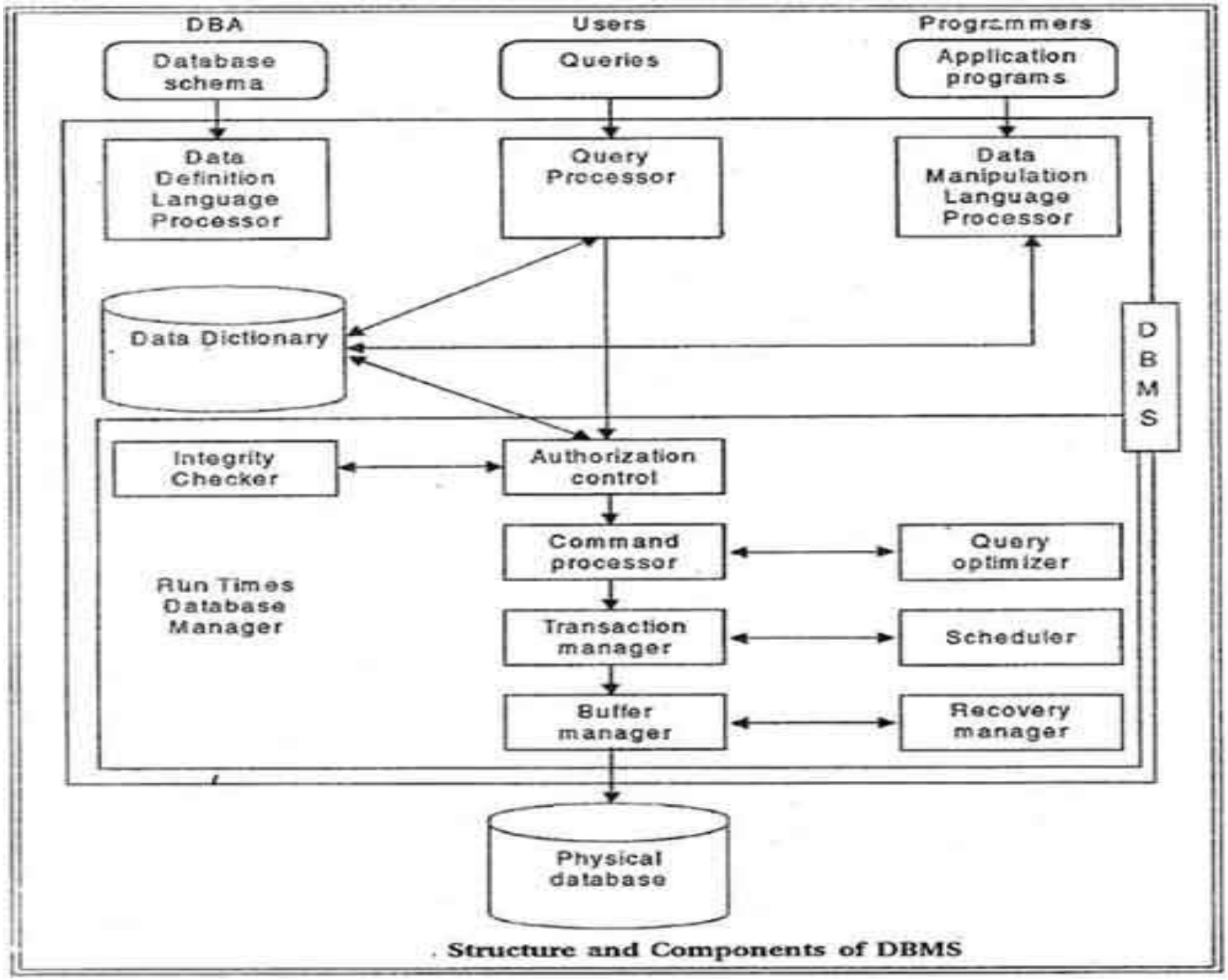
vi. Changes to compression techniques or hashing algorithms.

vii. Change of Location of Database from say C drive to D Drive

Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity(unsuitable or inappropriate) is vastly reduced.
- Easily make modifications in the physical level is needed to improve the performance of the system.

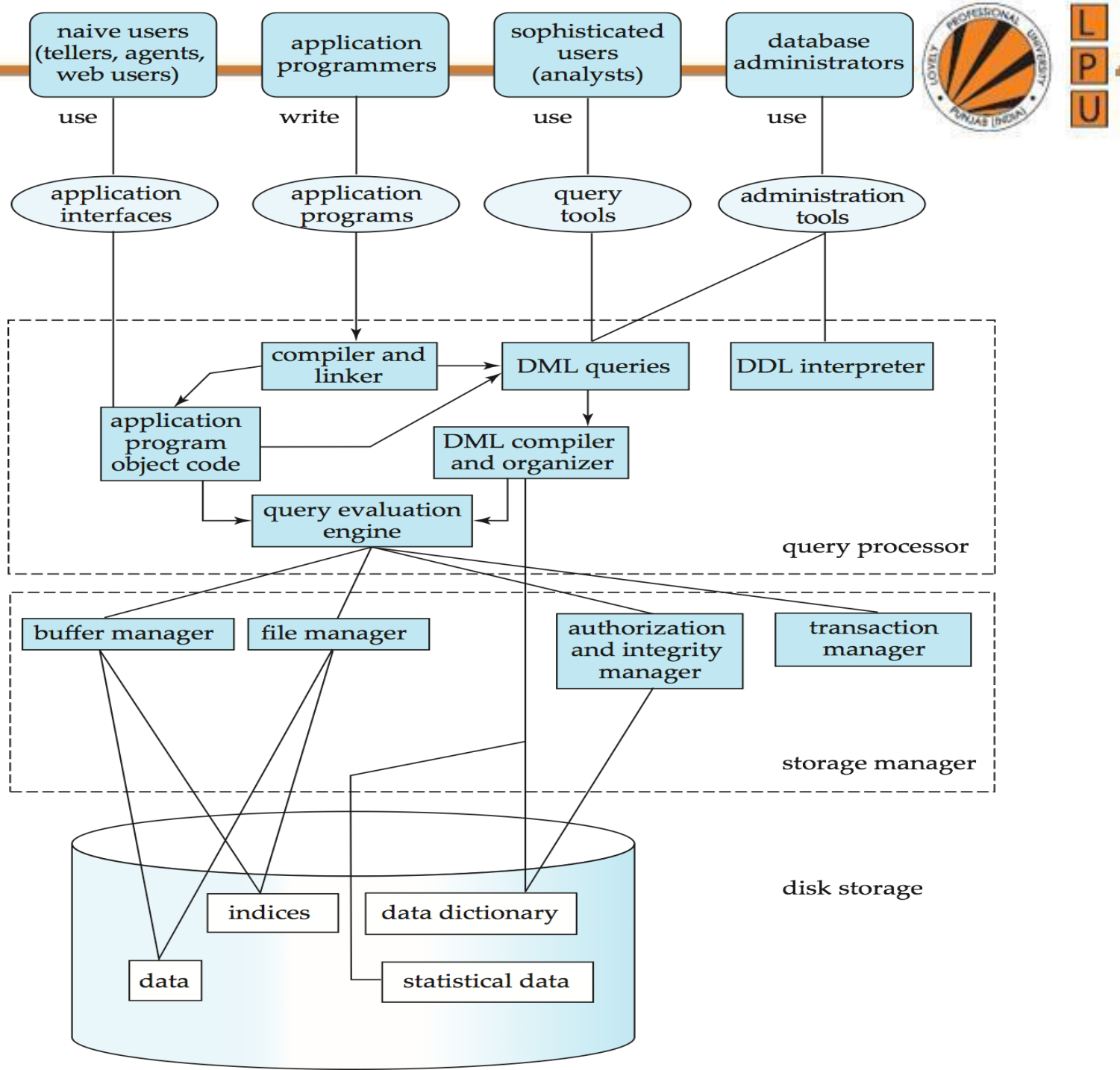
STRUCTURE AND COMPONENTS OF DBMS



Structure and Components of DBMS

Introduction

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into the ***STORAGE MANAGER*** and the ***QUERY PROCESSOR*** components.
- The ***STORAGE MANAGER*** is important because databases typically require a large amount of storage space.
- The ***QUERY PROCESSOR*** is important because it helps the database system simplify and facilitate access to data.


naive users (tellers, agents, web users)
application programmers
sophisticated users (analysts)
database administrators
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

Components Of DBMS

- The database system is divided into 3 components:

a. *Query Processor,*

b. *Storage Manager, and*

c. *Disk Storage.*

Query Processor

- It interprets the requests (queries) received from end user via an application program into instructions.
- It also executes the user request which is received from the DML compiler.
- Query Processor contains the following components :–

i. ***DML Compiler:*** Processes the DML statements into low level instruction (machine language), so that they can be executed.

ii. ***DDL Interpreter:*** Processes the DDL statements into a set of table containing meta data (data about data).

iii. A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs ***QUERY OPTIMIZATION***, that is, it picks the lowest cost evaluation plan from among the alternatives.

iv. ***QUERY EVALUATION ENGINE:*** Executes low-level instructions generated by the DML compiler.

Storage Manager

- It is a program that provides an interface between the data stored in the database and the queries received.
- Also known as *DATABASE CONTROL SYSTEM*.
- Maintains the consistency and integrity of the database by applying the constraints and executes the **DCL** statements.
- Responsible for updating, storing, deleting, and retrieving data in the database.

- It contains the following components :–

i. ***Authorization Manager –*** It ensures role-based access control, i.e,. checks whether the particular person is privileged to perform the requested operation or not.

ii. ***Integrity Manager –*** It checks the integrity constraints when the database is modified.

iii. ***Transaction Manager –*** It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.

iv. ***File Manager –*** It manages the file space and the data structure used to represent information in the database.

v. ***Buffer Manager –*** It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

Disk Storage

- It contains the following components –

i. ***Data Files –*** It stores the data.

ii. ***Data Dictionary –*** It contains the information about the structure of any database object. It is the repository of information that governs the metadata.

iii. ***Indices –*** It provides faster retrieval of data item.