23CSE202
Database Management Systems

Syllabus

Unit 1

Introduction: Overview of DBMS fundamentals – Overview of Relational Databases and Keys. Relational Data Model: Structure of relational databases – Database schema. Database Design: Overview of the design process - The E-R Models – Constraints - Removing Redundant Attributes in Entity Sets - E-R Diagrams - Reduction to Relational Schemas - Entity Relationship Design Issues - Extended E-R Features – Alternative E-R Notations.

Unit 2

Relational Database Design: Features of Good Relational Designs - Atomic Domains and 1NF - Decomposition using Functional Dependencies: 2NF, 3NF, BCNF and Higher Normal Forms. Functional Dependency Theory - Algorithm for Decomposition – Decomposition using multi-valued dependency: 4NF and 4NF decomposition. Database design process and its issues. SQL: review of SQL – Intermediate SQL – Advanced SQL.

Unit 3

File Organization – Indexing and Hashing - Storage Structure - Transactions: Transaction concepts- ACID Properties – Serializability – Recoverable schedules, Cascade less schedules. Need for Concurrency -Locking Protocols- Deadlock and Recovery. Overview and applications of NoSQL databases – MongoDB, Neo4j/GraphDB.

Textbook(s)

Silberschatz A, Korth HF, Sudharshan S. "Database System Concepts". Seventh Edition, TMH publishing company limited; 2019.

Chapter 1
Introduction to DBMS

Introduction to DBMS

- DBMS stands for Database Management System.
- DBMS = Database + Management System.
- Database is a collection of data and Management System is a set of programs to store and retrieve those data.
- Based on this we can define DBMS like this: DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.

What is the need of DBMS?

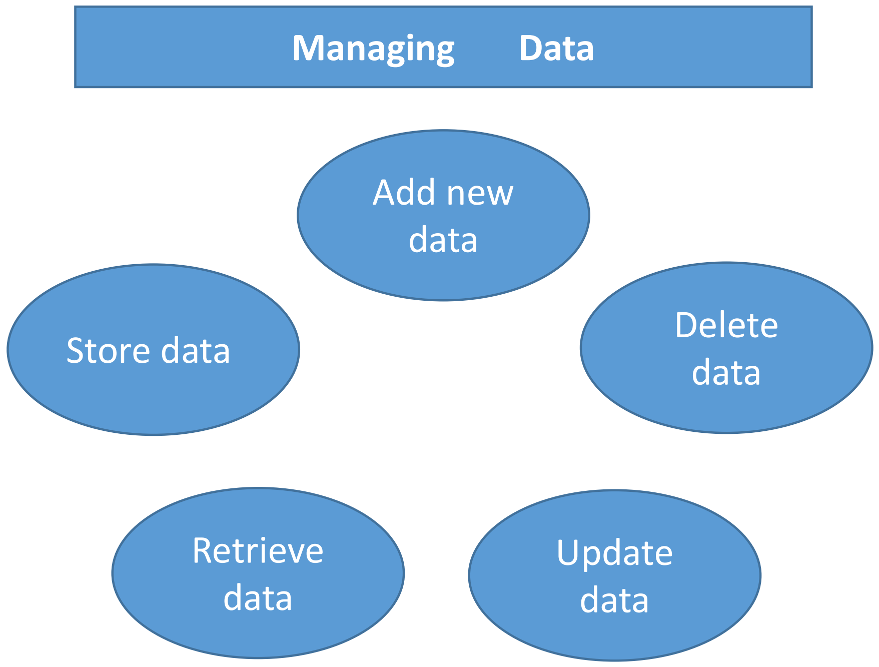
- **Storage**: Acquires less space as the redundant data (duplicate data) has been removed before storage.
  Example: A person having 2 accounts and the details

- **Fast Retrieval of data**: Faster retrieval of data through queries

Purpose of Database Systems

- To manage the data.
- Consider a University that keeps the data of students, teachers, courses, books etc.

- For all the above , we need a **Database Management System.**



Few Applications of DBMS

- **Airlines**: reservations, schedules, etc
- **Telecom**: calls made, customer details, network usage, etc
- **Universities**: registration, results, grades, etc
- **Sales**: products, purchases, customers, etc
- **Banking**: all transactions etc
- **Education sector**: staff ,student & course details, registration, etc
- **Online shopping**: product details, billing, shipping, etc

DBMS vs File System

Advantages of File Processing System :

- **Cost friendly –**
  There is a very minimal to no set up and usage fee for File Processing System. (In most cases, free tools are inbuilt in computers.)

- **Easy to use –**
  File systems require very basic learning and understanding, hence, can be easily used.

- **High scalability –**
  One can very easily switch from smaller to larger files as per his needs.

Drawbacks of File system

- **Data redundancy:** Refers to the duplication of data.

Ex: Students registers for 2 courses, the details of that student has to be saved in two files, this will take more storage than needed. Data redundancy often leads to higher storage costs and poor access time.

- **Data inconsistency:** Data redundancy leads to data inconsistency.

Ex: Considering the student in the above example, if the student requests to change his address. If the address is changed at one place and not on all the records then this can lead to data inconsistency.

- **Data Isolation:** Data are scattered in various files and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

- **Difficulty in accessing data**
    - Need to write a new program to carry out each new task

- **Dependency on application programs:** Changing files would lead to change in application programs.

Drawbacks of File system

- **Integrity problems**. The data values stored in the database must satisfy certain types of **consistency constraints**.

Suppose the 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

Drawbacks of File system

- **Atomicity issues:** All the operations in a transaction executes or none.

  Ex: Fund transfer across accounts involves debit and credit operations, incase after debit, system fails. It is difficult to achieve atomicity in file processing systems.

- **Concurrent access by multiple users**
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- **Data Security:** Unauthorized access should be avoided.

Ex: A student in a college should not be able to see the payroll details of the teachers, such kind of security constraints are difficult to apply in file processing systems.

- Hard to provide user access to some, but not all, data

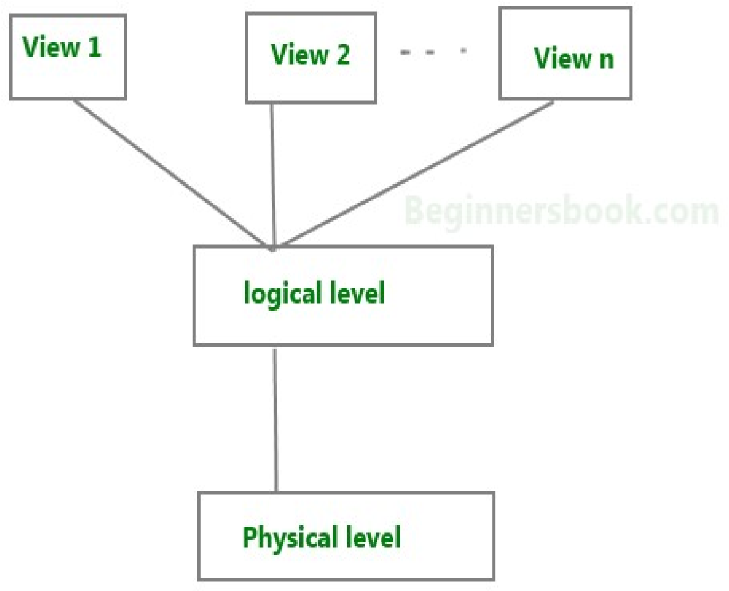
Database systems offer solutions to all the above problems

Advantages of DBMS over file system

- **No redundant data**: Redundancy removed by data normalization. No data duplication saves storage and improves access time.
- **Data Consistency and Integrity**: Since data normalization takes care of the data redundancy, data inconsistency is checked
- **Data Security**: It is easier to apply access constraints in database systems so that only authorized user is able to access the data.
- **Privacy**: Limited access means privacy of data.
- **Easy access to data** – Database systems manages data in such a way so that the data is easily accessible with fast response times.
- **Easy recovery**: Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
- **Flexible**: Database systems are more flexible than file processing systems.

Data Abstraction in DBMS

A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data is stored and maintained.



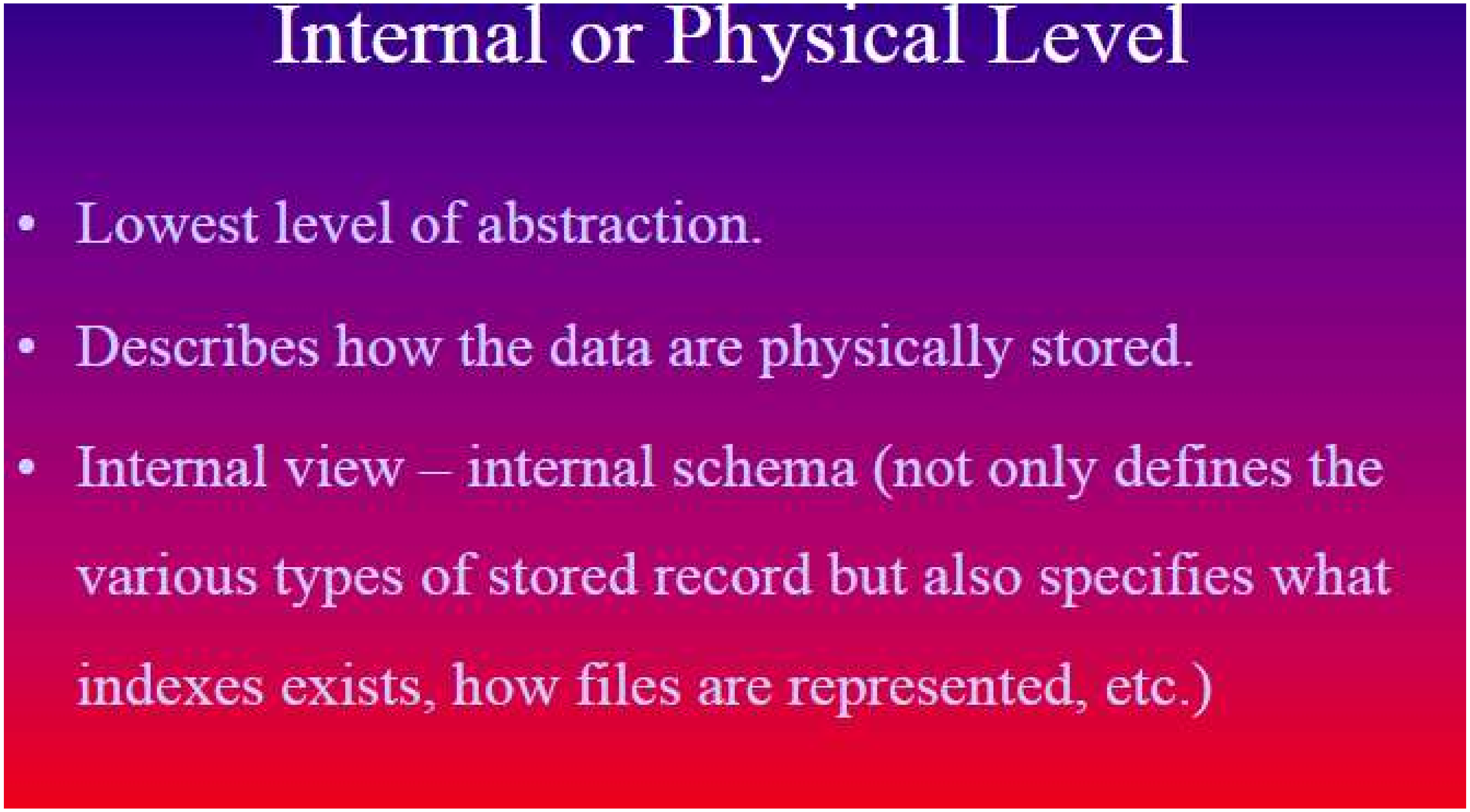
Three Levels of data abstraction

Internal/Physical level: describes how a record (e.g., customer) is stored.

Conceptual/Logical level: describes what data stored in database, and the relationships among the data.

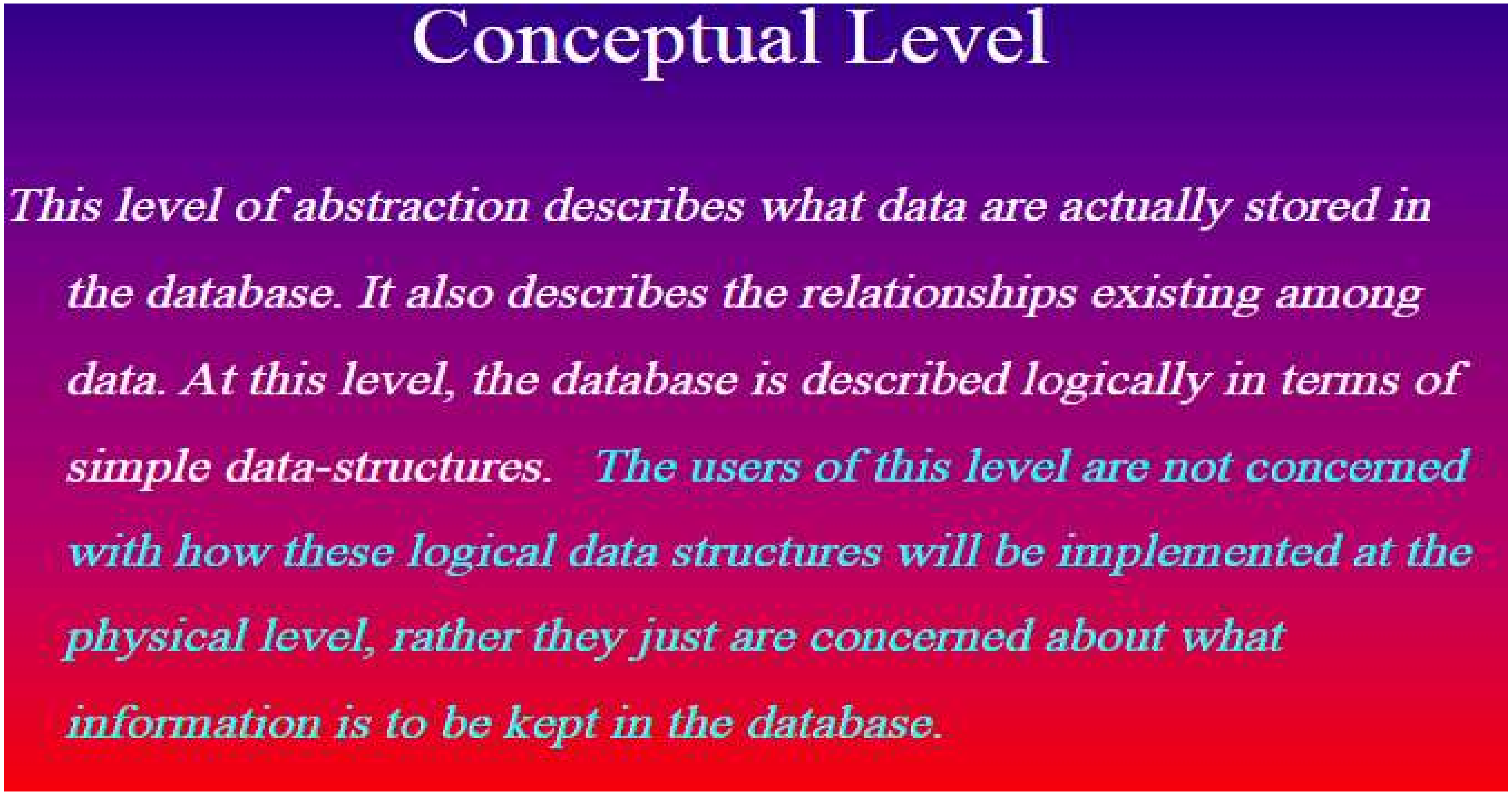
```
type instructor = record
        ID : string;
name : string;
dept_name : string;
salary : integer;
end;
```

External/View level: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.



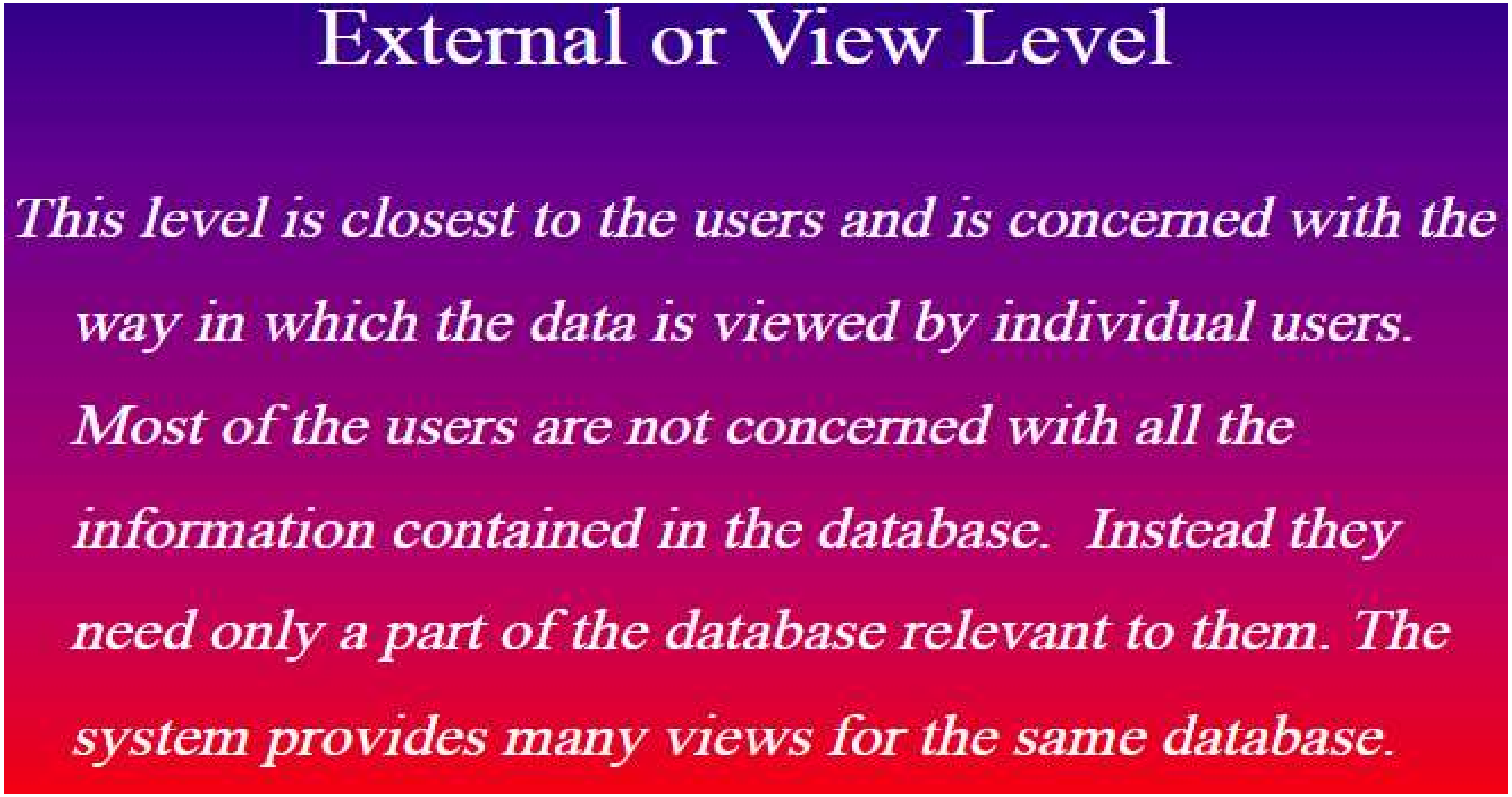
Internal or Physical Level

- Lowest level of abstraction.
- Describes how the data are physically stored.
- Internal view – internal schema (not only defines the various types of stored record but also specifies what indexes exists, how files are represented, etc.)



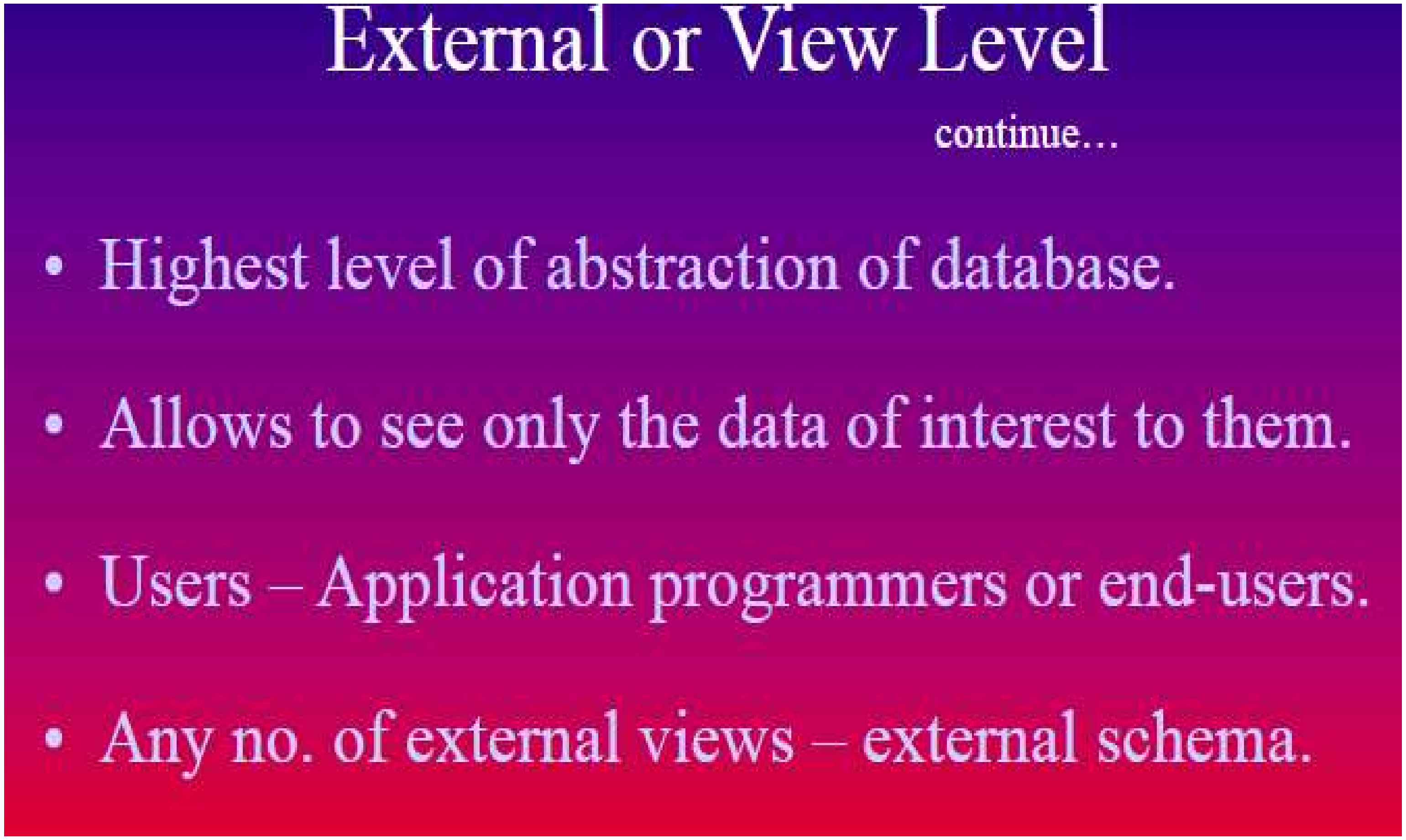
Conceptual Level

This level of abstraction describes what data are actually stored in the database. It also describes the relationships existing among data. At this level, the database is described logically in terms of simple data-structures. The users of this level are not concerned with how these logical data structures will be implemented at the physical level, rather they just are concerned about what information is to be kept in the database.



External or View Level

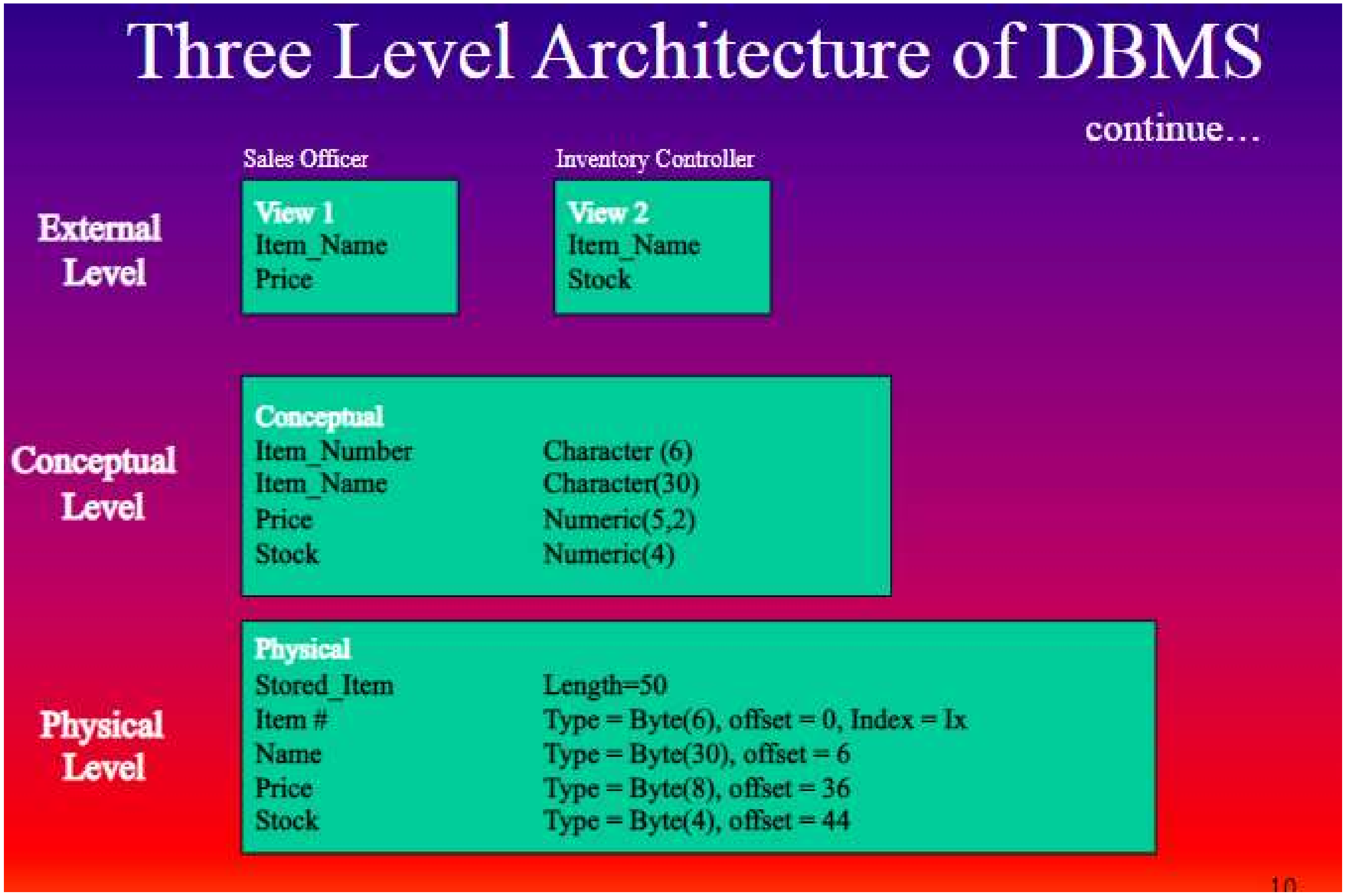
This level is closest to the users and is concerned with the way in which the data is viewed by individual users. Most of the users are not concerned with all the information contained in the database. Instead they need only a part of the database relevant to them. The system provides many views for the same database.



External or View Level

continue…

- Highest level of abstraction of database.
- Allows to see only the data of interest to them.
- Users – Application programmers or end-users.
- Any no. of external views – external schema.



Three Level Architecture of DBMS

continue…

View 1 (Sales Officer)
Item_Name
Price

View 2 (Inventory Controller)
Item_Name
Stock

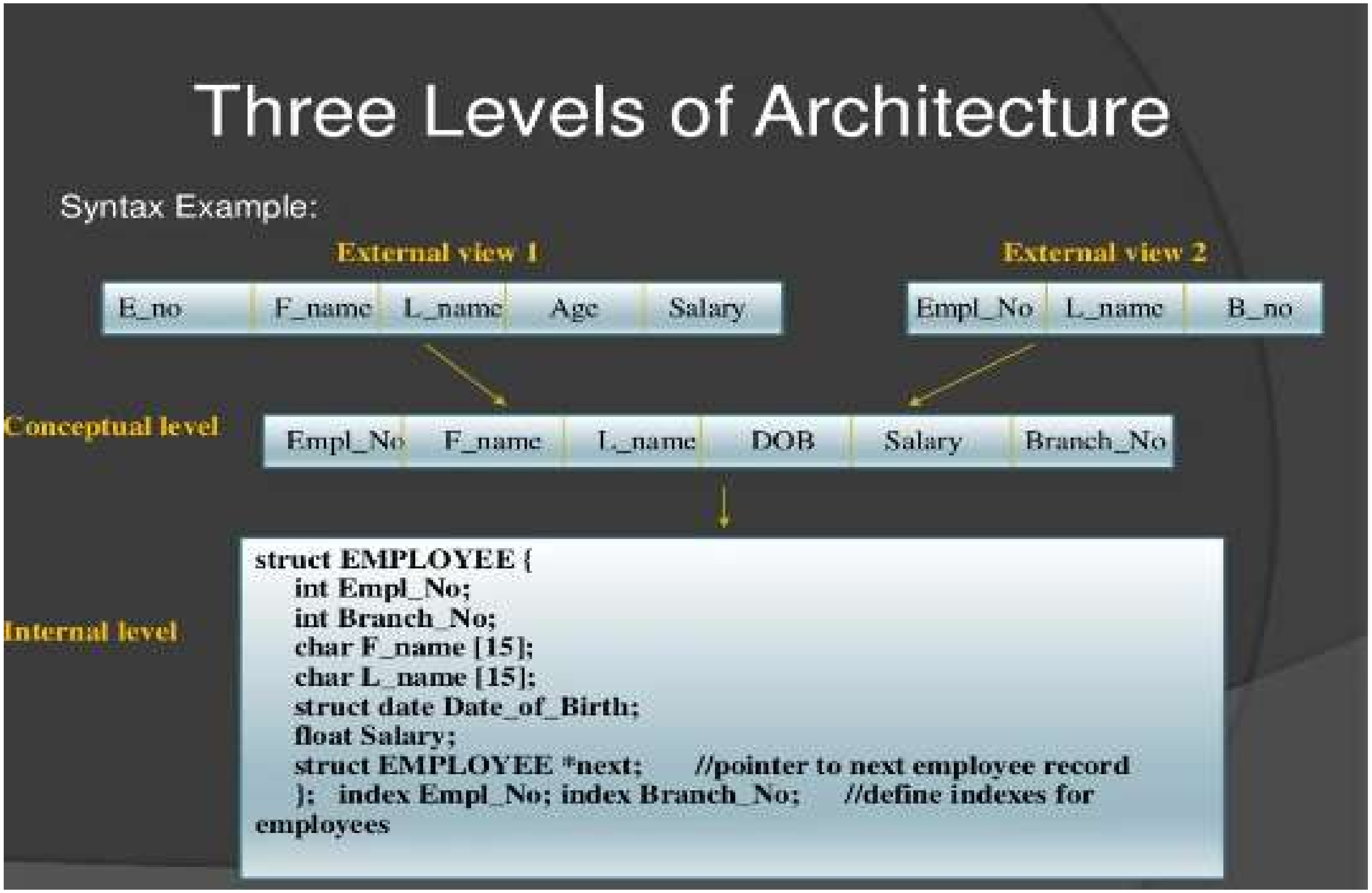
Conceptual

Item_Number	Character (6)
Item_Name	Character(30)
Price	Numeric(5,2)
Stock	Numeric(4)

Physical

Stored_Item	Length=50
Item #	Type = Byte(6), offset = 0, Index = Ix
Name	Type = Byte(30), offset = 6
Price	Type = Byte(8), offset = 36
Stock	Type = Byte(4), offset = 44

10



Three Levels of Architecture

Syntax Example:

```
struct EMPLOYEE {
    int Empl_No;
    int Branch_No;
    char F_name [15];
    char L_name [15];
    struct date Date_of_Birth;
    float Salary;
    struct EMPLOYEE *next;      //pointer to next employee record
    };   index Empl_No; index Branch_No;     //define indexes for
employees
```

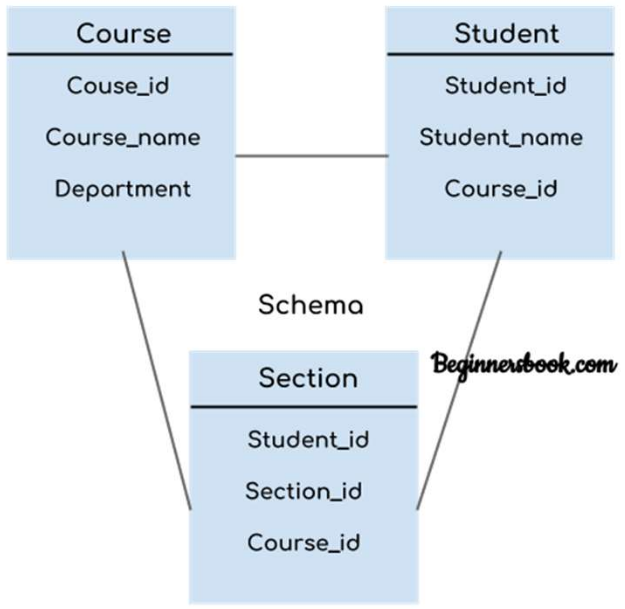
Schema

Schema – the logical structure of the database.
 Example: The database consists of information about a set of customers and accounts and the relationship between them

Analogous to type information of a variable in a program

Schema is of three types:

- Physical schema
- Logical schema
- View schema

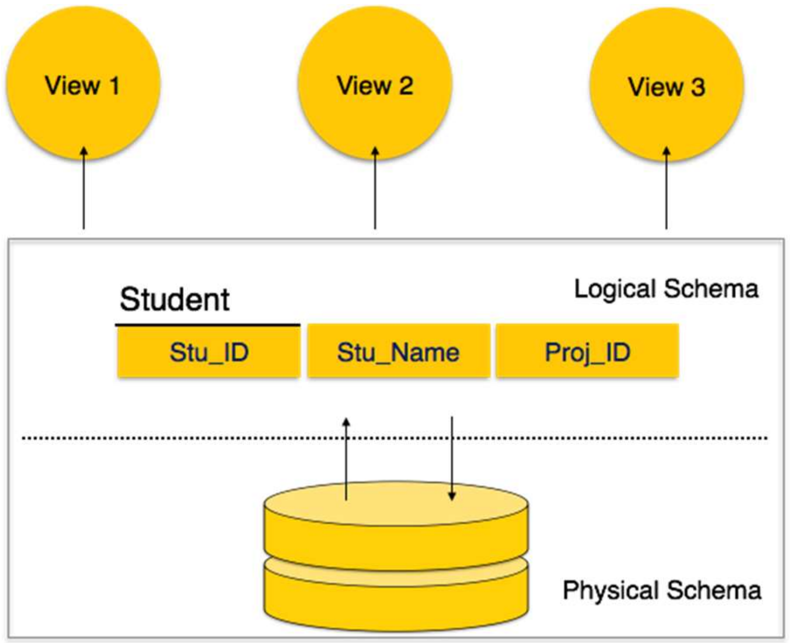


For example: In the diagram, we have a schema showing the relationship between three tables: Course, Student and Section. It does not show the data present in those tables. Schema is only a structural view(design) of a DB

Types of Schema

View schema: **Design of database at view level**

Describes end user interaction with database systems



Logical schema: **Design of database at logical level**

Programmers and database administrators work at this level, which describes the table structure , how attributes/columns are interrelated, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

Physical schema : The design of a database at physical level

How the data stored in blocks of storage is described at this level.

Instance of a DB

Definition

The data stored in database at a particular moment of time.

Instance – the actual content of the database at a particular point in time

Analogous to the value of a variable

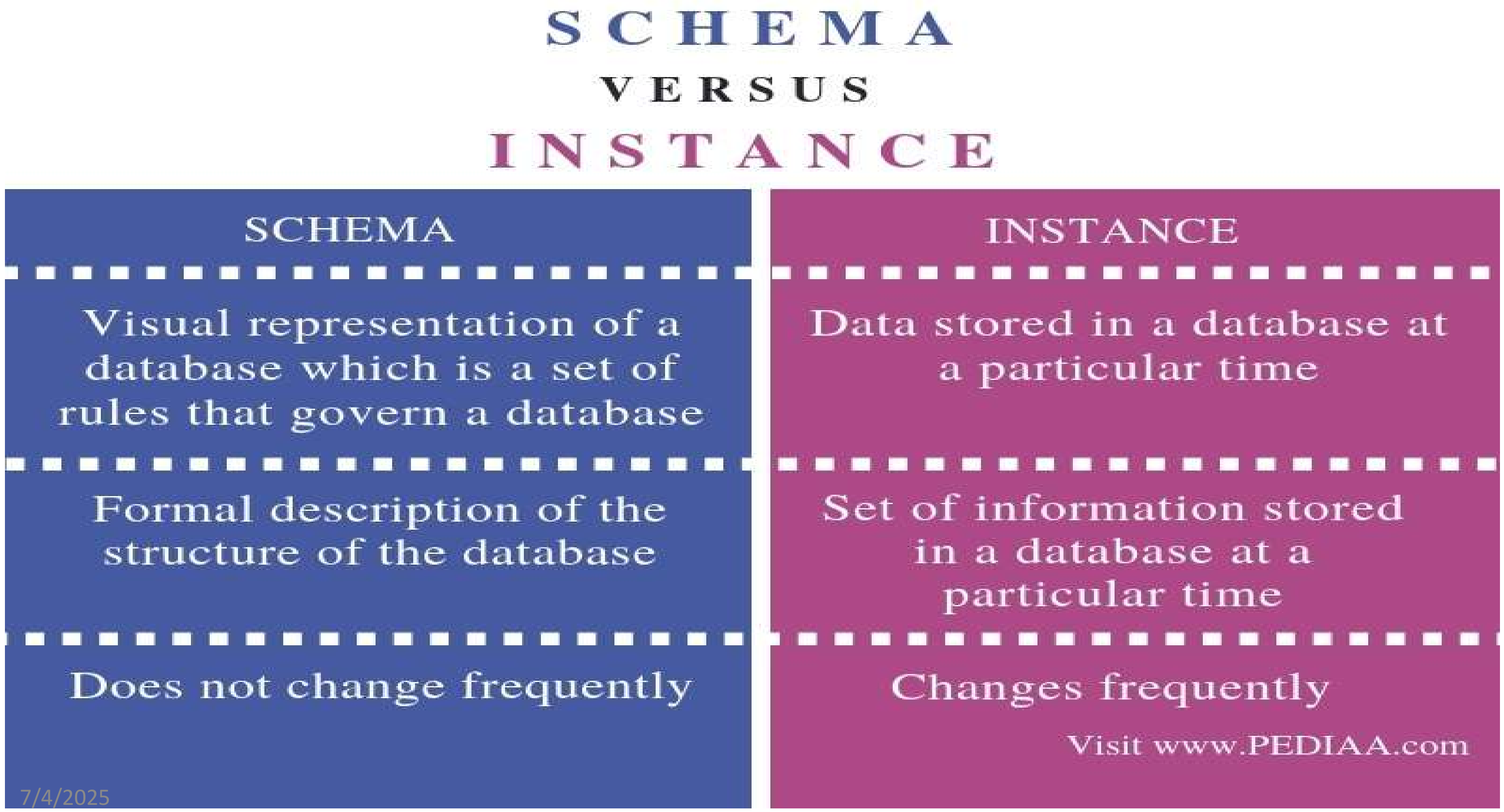
For example, let's say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Let's say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table.

In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

Question?????

Is it schema that changes frequently or is it instance?????

Difference between schema and instance

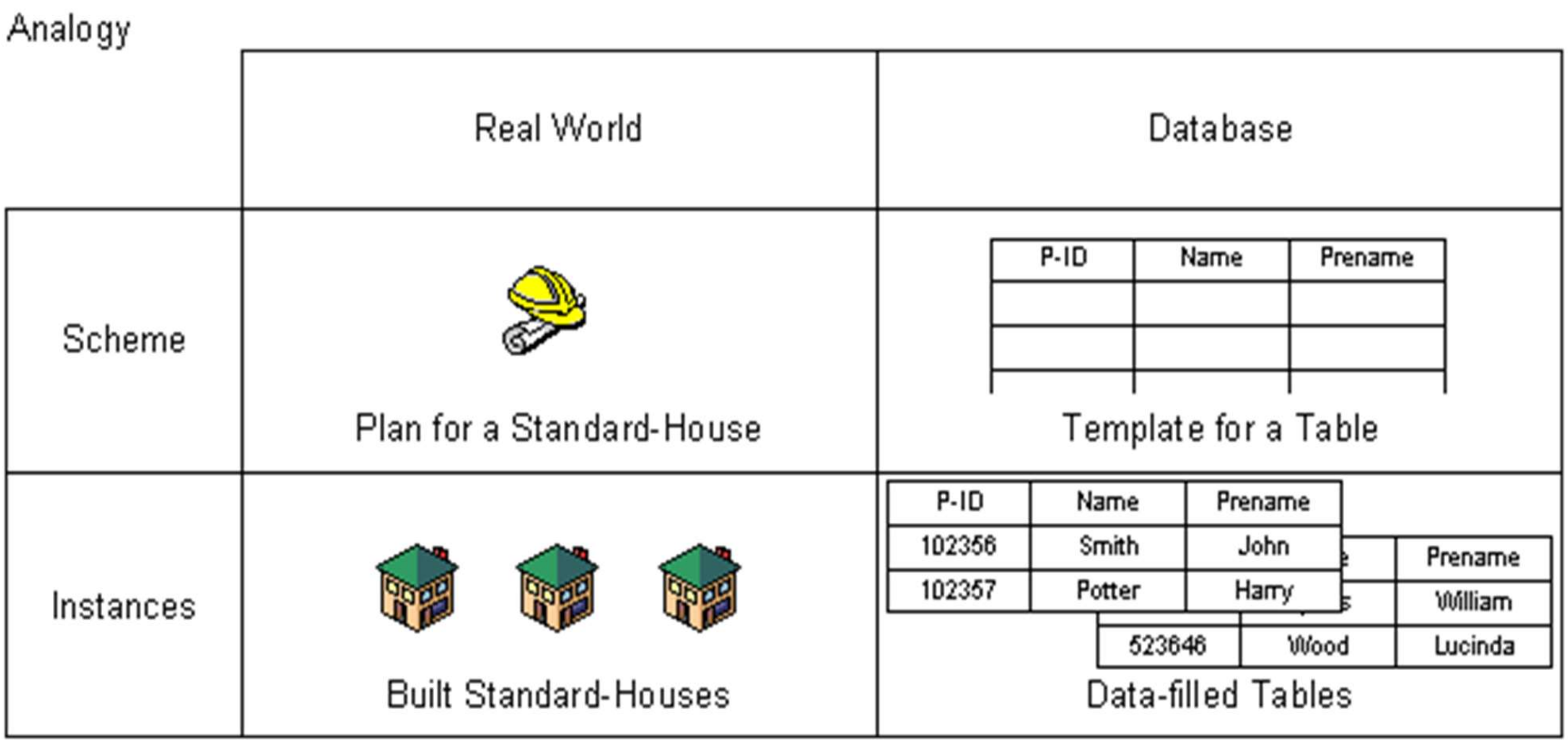


SCHEMA VERSUS INSTANCE

SCHEMA	INSTANCE
Visual representation of a database which is a set of rules that govern a database	Data stored in a database at a particular time
Formal description of the structure of the database	Set of information stored in a database at a particular time
Does not change frequently	Changes frequently

Visit www.PEDIAA.com

Difference between schema and instance


Analogy

	Real World	Database
Scheme	Plan for a Standard-House	Template for a Table
Instances	Built Standard-Houses	Data-filled Tables

P-ID	Name	Prename

P-ID	Name	Prename
102356	Smith	John
102357	Potter	Harry

		Prename
		William
523646	Wood	Lucinda

7/4/2025

- Data Abstraction (Levels of Abstraction):
- **Physical/Internal Level:**
- Describes how data is physically stored in the database (e.g., file structures, storage devices).
- **Logical/Conceptual Level:**
- Defines the structure of the entire database, including data types, relationships, and constraints.
- **View/External Level:**
- Presents specific parts of the database to different users, hiding unnecessary details and complexities.

Schema (Three-Schema Architecture):

- **Internal Schema (Physical Schema):** Maps the conceptual schema to the physical storage details.
- **Conceptual Schema (Logical Schema):** Describes the overall structure and content of the database.
- **External Schema (View Schema):** Provides different views of the database for various user groups.

Key Differences:

- **Purpose:** Data abstraction aims to simplify user interaction, while schema focuses on database design and structure.
- **Focus:** Abstraction hides complexity, while schema defines structure.
- **Perspective:** Abstraction is user-centric, and schema is system-centric.

- In simpler terms, data abstraction is like using a remote control to operate a TV without needing to know how the circuits work, while the schema is like the blueprint of the TV itself, showing how all the components are connected.

Objectives of using Three schema Architecture

- Every user should be able to access the same data but able to see a customized view of the data.
- The user need not to deal directly with physical database storage detail.
- The DBA should be able to change the database storage structure without disturbing the user's views
- The internal structure of the database should remain unaffected when changes made to the physical aspects of storage

Data Independence

What is Data Independence ???????

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

Why Data Independence ???????

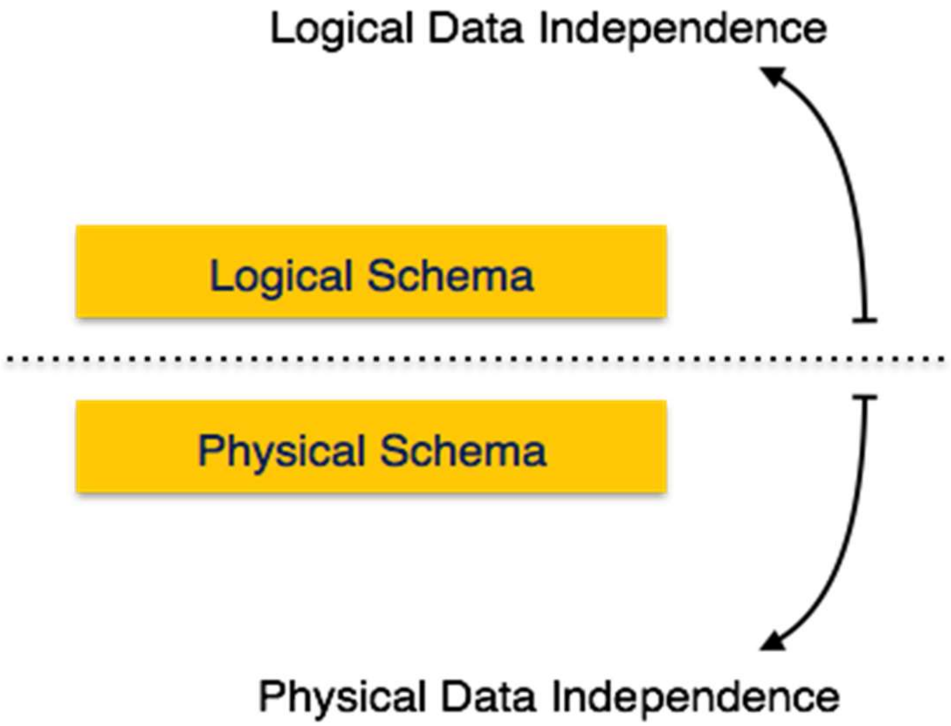
A database system normally contains a lot of data in addition to users' data.

Stores data about data, known as metadata, to locate and retrieve data easily.

It is rather difficult to modify or update a set of metadata once it is stored in the database.

But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

Data Independence



Physical Data Independence :

- Physical Data Independence is defined as the ability to make changes in the structure of the lowest level of the Database Management System (DBMS) without affecting the higher-level schemas.
- Hence, modification in the Physical level should not result in any changes in the Logical or View levels.
- Modifications at the physical level are occasionally necessary in order to improve performance of the system,

Example

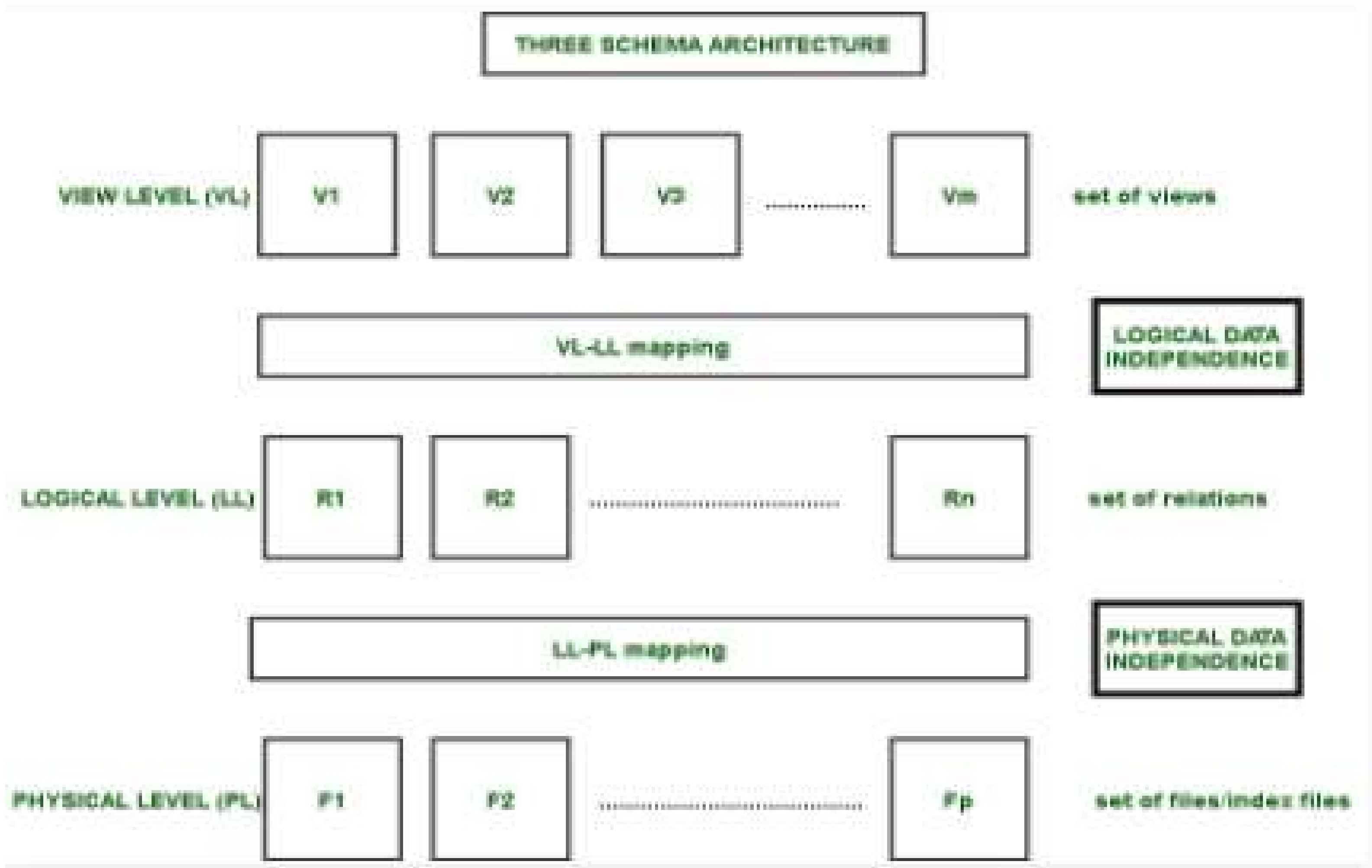
- Changes in the lowest level (physical level) are creating a new file, storing the new files in the system, creating a new index etc.
- Instances of why we may want to do any sort of Data modification in the physical level- We may want to alter or change the data in the physical level. This is because we may want to add or remove files and indexes to enhance the performance of the database system and make it faster.
- Hence, in this way, the Physical Data Independence enables us to do Performance Tuning. Ideally, when we change the physical level, we would not want to alter the logical and view level.

Logical Data Independence :

- Logical Data Independence is defined as the ability to make changes in the structure of the middle level of the Database Management System (DBMS) without affecting the highest-level schema or application programs.
- Hence, modification in the logical level should not result in any changes in the view levels or application programs.

Example

- Changes in the lowest level are: adding new attributes to a relation, deleting existing attributes of the relation etc. Ideally, we would not want to change any application or programs that do not require to use the modified attribute.


THREE SCHEMA ARCHITECTURE
VIEW LEVEL (VL)
V1
V2
V3
Vm
set of views
VL-LL mapping
LOGICAL DATA INDEPENDENCE
LOGICAL LEVEL (LL)
R1
R2
Rn
set of relations
LL-PL mapping
PHYSICAL DATA INDEPENDENCE
PHYSICAL LEVEL (PL)
F1
F2
Fp
set of files/index files

Data Independence

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk.

The ability to change the physical data without impacting the schema at the logical level.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas. Or

may be we increase the memory capacity of the hard disk, and this is no way requires an explicit change in the logical level

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside.
The ability to change the logical data without impacting the schema at the view level

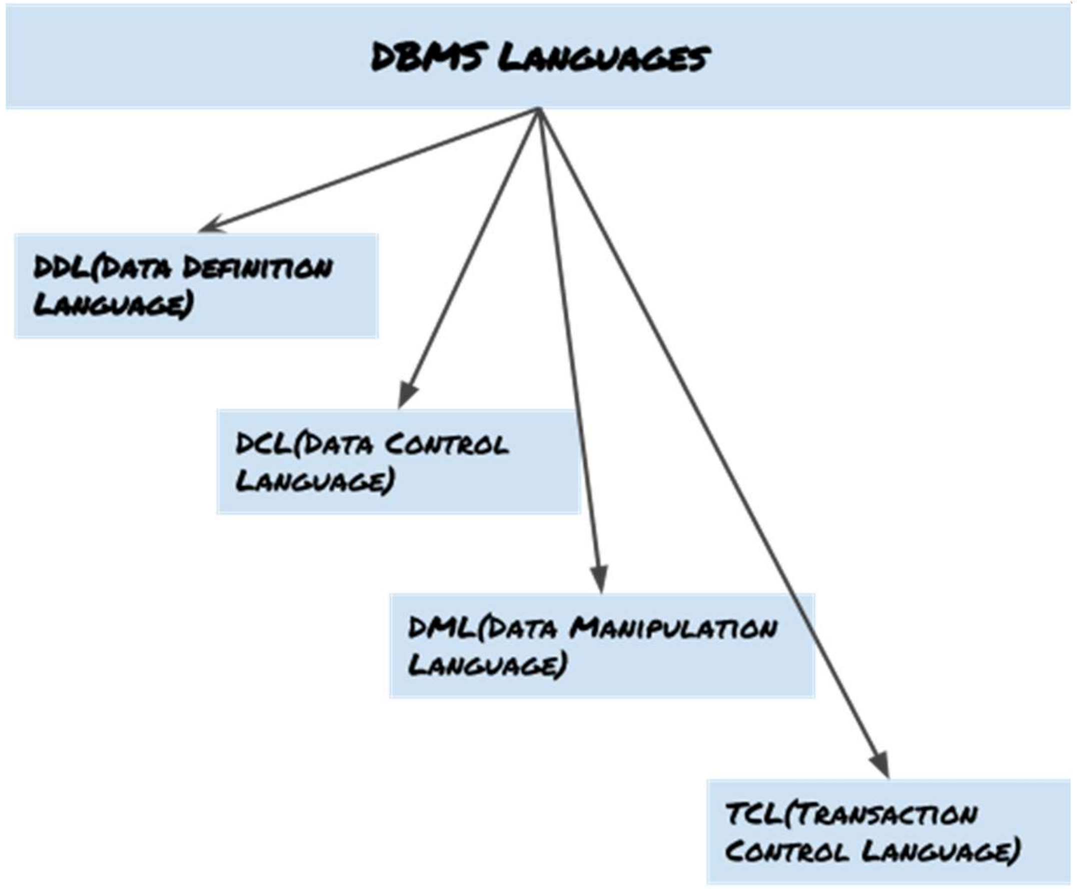
For example, a table (relation) stored in the database and all its constraints, applied on that relation. Or

We enforce a constraint in the logical level

Database Languages

- A database system provides a **data-definition language** to specify the database schema and a **data-manipulation language** to express database queries and updates.
- In practice, the data-definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

Data Base Languages-SQL


DBMS LANGUAGES
DDL(DATA DEFINITION LANGUAGE)
DCL(DATA CONTROL LANGUAGE)
DML(DATA MANIPULATION LANGUAGE)
TCL(TRANSACTION CONTROL LANGUAGE)

DBMS Languages

Data Definition Language (DDL)

DDL is used for specifying the database schema.

It is used for creating tables, schema, indexes, constraints etc. in database.

- To create the database instance – **CREATE**
- To alter the structure of database – **ALTER**
- To drop the structure of database – **DROP**
- To delete tables in a database instance – **TRUNCATE**

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ```
           ID         char(5),
           name       varchar(20),
           dept_name  varchar(20),
           salary     numeric(8,2))
  ```

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
    - Referential integrity (**references** constraint in SQL)
      - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation
  - Authorization

DBMS Languages

Data Manipulation Language (DML)

DML is used for accessing and manipulating data in a database.

The following operations on database comes under DML:

- To read records from table(s) – **SELECT**
- To insert record(s) into the table(s) – **INSERT**
- Update the data in table(s) – **UPDATE**
- Delete all the records from the table – **DELETE**

DBMS Languages

Data Control language (DCL)

DCL is used for granting and revoking user access on a database –

- To grant access to user – **GRANT**
- To revoke access from user – **REVOKE**

DBMS Languages

Transaction Control Language(TCL)

The changes in the database that we made using DML commands are either performed or rolled back using TCL.

- To persist the changes made by DML commands in database – **COMMIT**
- To rollback the changes made to the database – **ROLLBACK**