# Lecture 2

# Data Models & DBMS Architecture

**Dr. Vandana Kushwaha**

Department of Computer Science

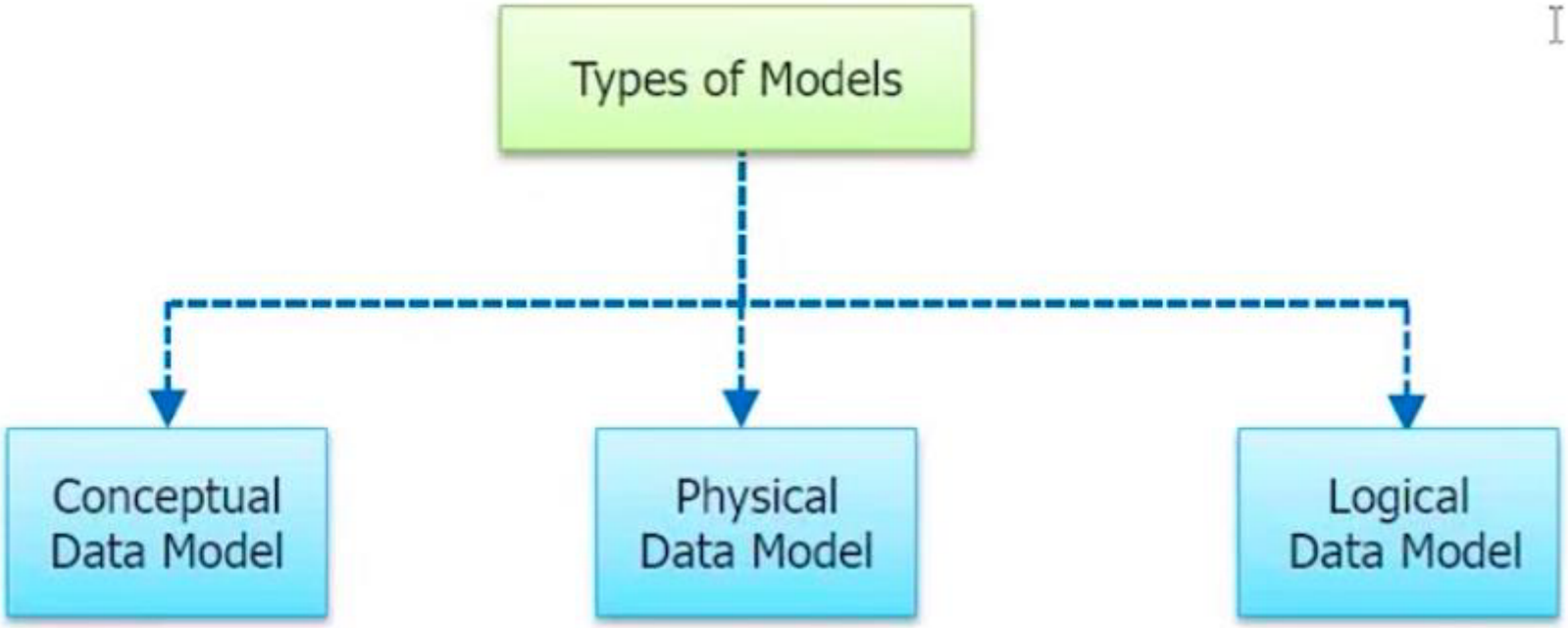
Institute of Science, BHU, Varanasi

# Introduction

- **Data abstraction** generally refers to the **suppression of details** of **data organization and storage**, and the **highlighting** of the **essential features** for an improved understanding of data.
- One of the **main characteristics** of the **Database approach** is to support **Data Abstraction** so that different users can perceive data at their preferred level of detail.
- A **Data model** is a **collection of concepts** that can be used to **describe the structure of a database**—provides the necessary means to **achieve** this **abstraction.**
- By ***structure of a database*** we mean the **data types, relationships**, and **constraints** that apply to the **data.**
- Most **data models** also include a set of **basic operations** for specifying **retrievals** and **updates** on the **database.**

# Categories of Data Models

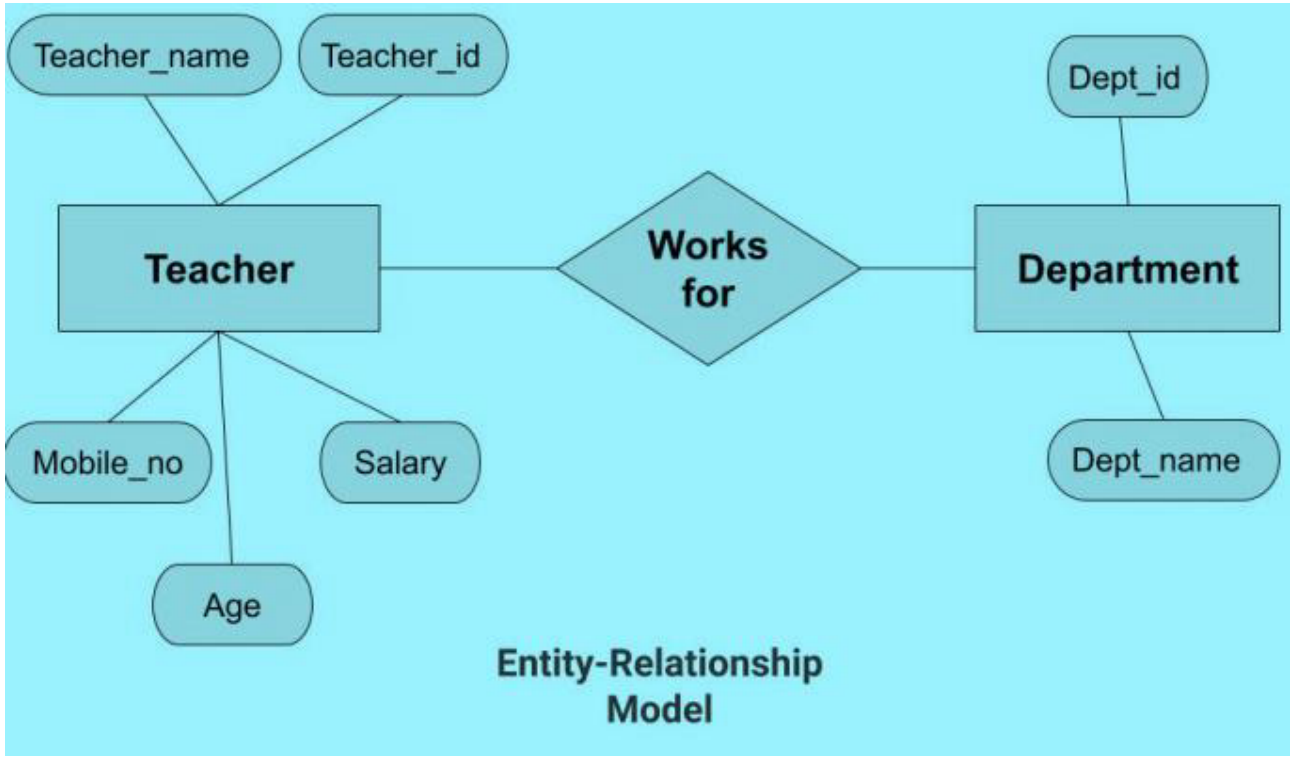
1. **High-level** or **Conceptual Data models**
2. **Representational** or **Implementation** or **Logical Data models**
3. **Low-level** or **Physical Data models**



# 1. High-level or Conceptual Data models

- It provide concepts that are close to the way many **users perceive data.**
- **Conceptual data models** use concepts such as **entities, attributes,** and **relationships.**
- An **entity** represents a **real-world object or concept**, such as an **employee** or a **project** that is described in the **database.**
- An **attribute** represents **some property** of interest that further describes an entity, such as the **employee's name** or **salary.**
- A **relationship** among **two or more entities** represents an **association among the entities**, for **example**, a **works-on relationship** between an **employee** and a **project.**
- **Example**: **Entity–Relationship Model(ER-Model)**—a popular **high-level conceptual data model.**

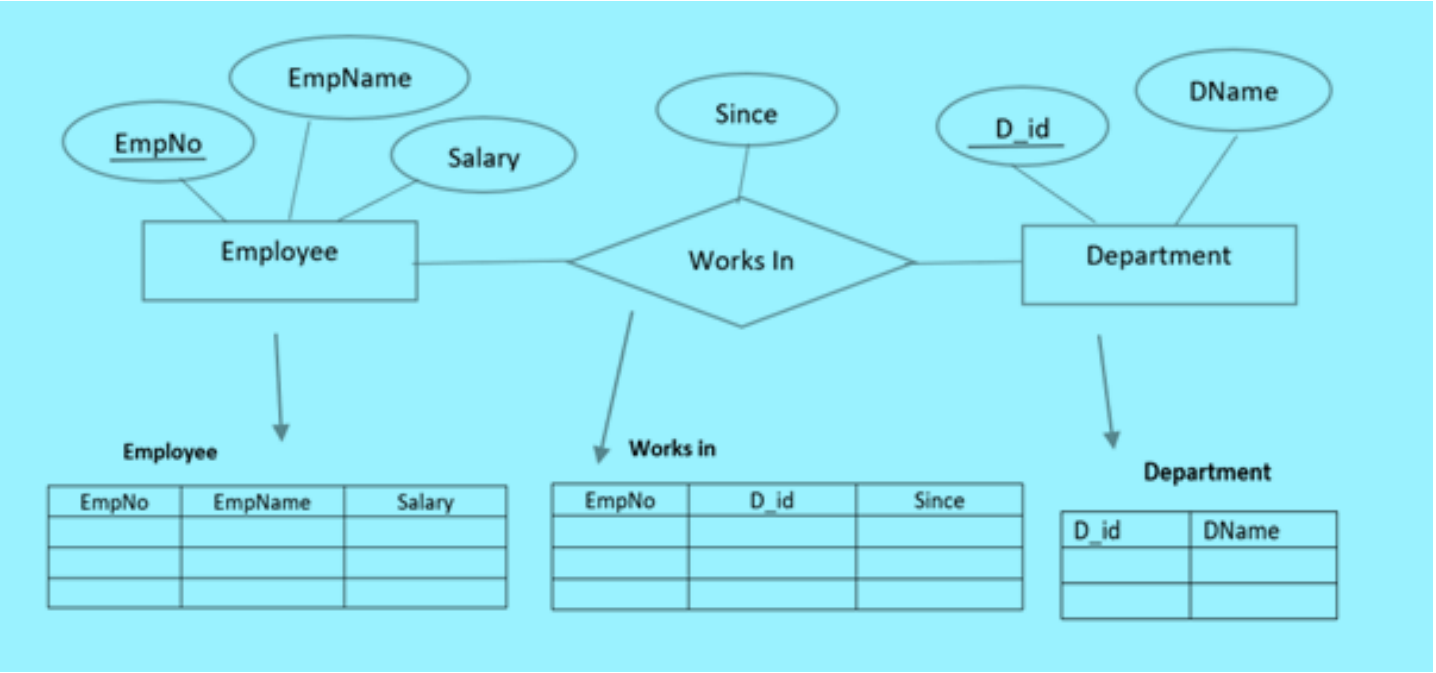
# 1. High-level or Conceptual Data models



# 2. Representational or Implementation or Logical Data models

- This type of **data model** is used to represent only the **logical part** of the **database** and **does not represent** the **physical structure** of the **databases**.
- It provide concepts that may be **easily understood by end users** .
- It **hide many details** of **data storage on disk** but **can be implemented** on a **computer system directly**.
- **Representational** or **implementation data models** are the models used most frequently in **traditional commercial DBMSs**.
- These include the widely used **Relational data model**, as well as the so-called legacy data models—the **Network** and **Hierarchical models**—that have been widely used in the **past.**

# 2. Representational or Implementation or Logical Data models



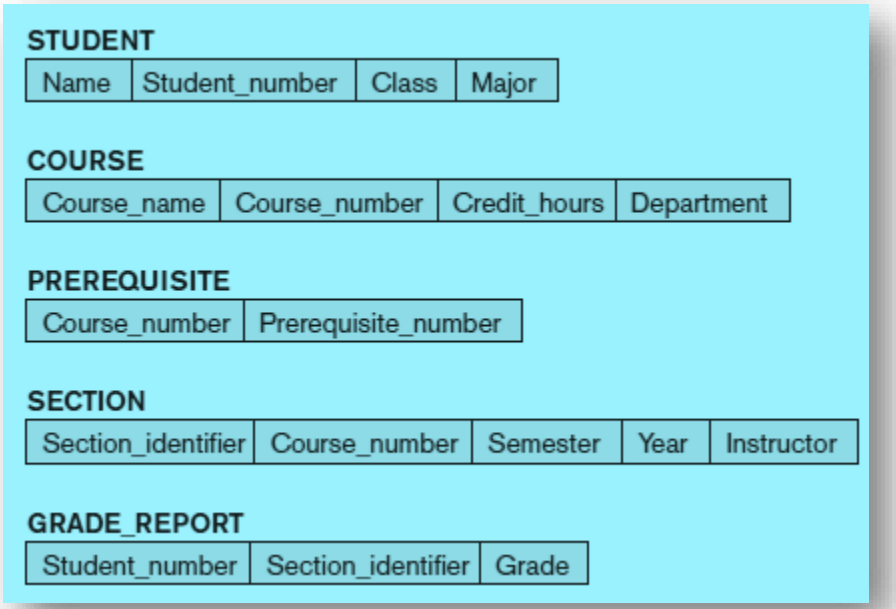

**Relational Model**

# 3. Low-level or Physical data models

- It **provide concepts** that describe the **details of how data is stored on the computer storage media**, typically **magnetic disks.**
- Concepts provided by **physical data models** are generally **meant** for **computer specialists, not for end users.**
- **Physical data models** describe **how data is stored as files in the computer** by representing information such as **record formats, record orderings,** and **access paths.**
- An **access path** is a **search structure** that makes the search for particular database records efficient, such as **indexing** or **hashing**.

# Schemas, Instances, and Database State

- In a **data model**, it is important to distinguish between the ***description* of the database** and the ***database itself.***
- The **description** of a **database** is called the **database schema**, which is specified during **database design** and is not expected to change frequently.

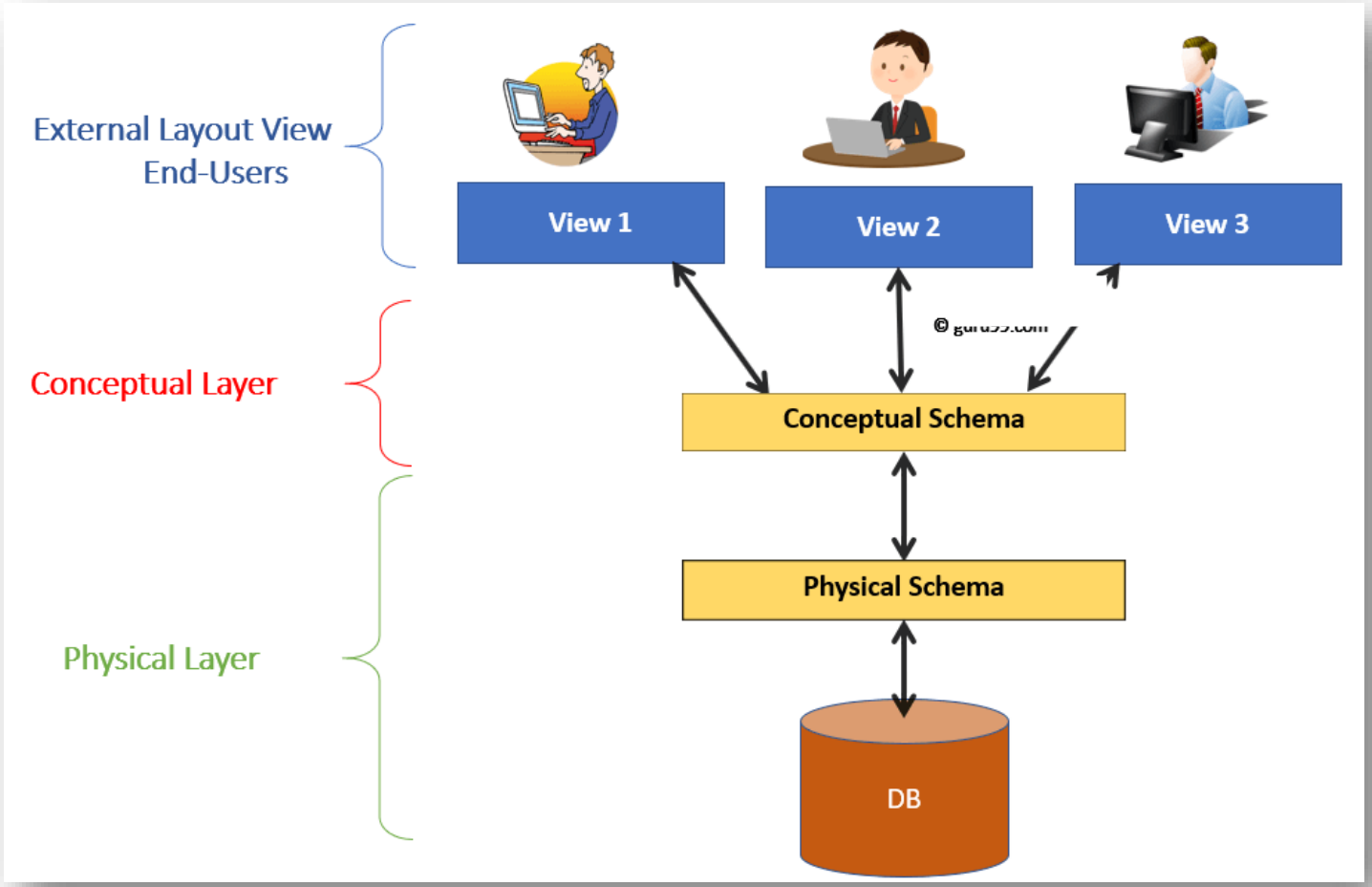


# The Three-Schema Architecture

- The goal of the **ANSI SPARC(System Planning and Requirement Committee) three-schema architecture**, is to separate the user applications from the physical database.
- In this architecture, **schemas** can be defined at the following **three levels:**

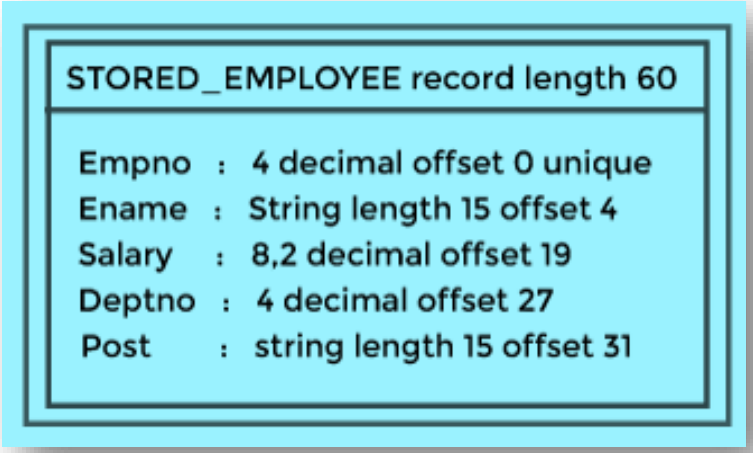
1. **Internal or Physical Level**
2. **Conceptual or Logical Level**
3. **External or View Level**

# The Three-Schema Architecture



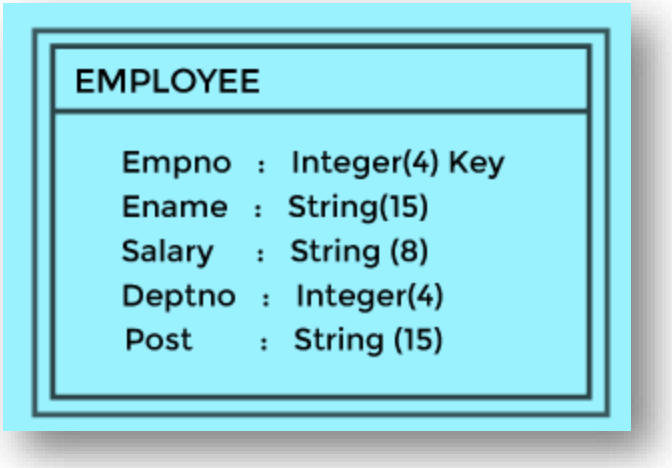
# Internal or Physical Level

- The **internal level** has an **internal schema**, which describes **HOW** the data in the database is to be physically stored?
- The **internal schema** uses a **physical data model** and describes the following:
  - **Data Structure and File Organization used to store data.**
  - **File access method to locate data, indexing techniques for the database.**
  - **Data Compression and Data Encryption technique.**



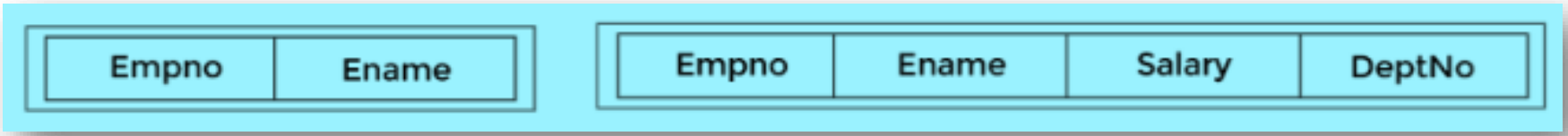
# Conceptual or Logical Level

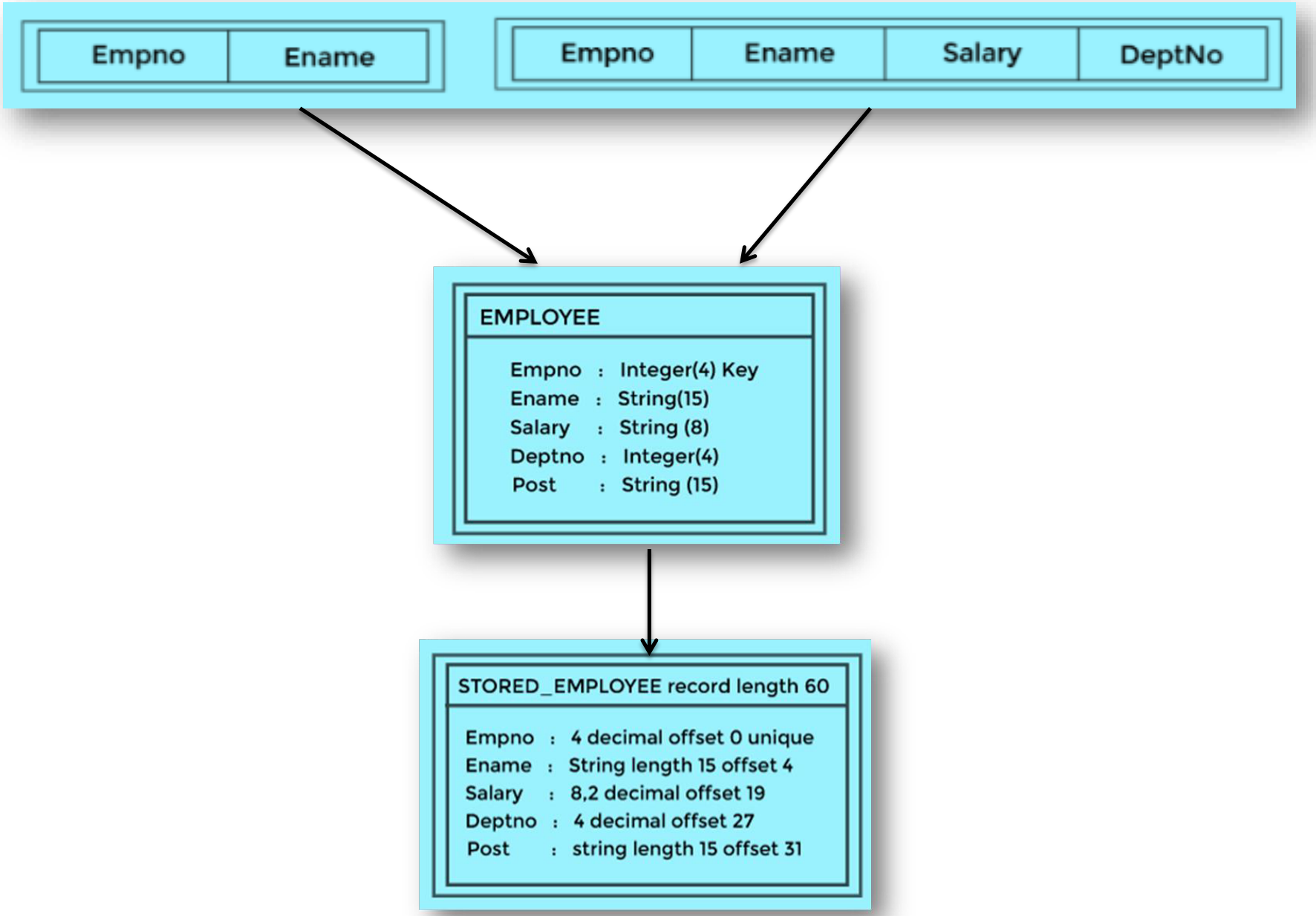
- The **conceptual level** describes **WHAT** data is stored within the whole database.
- The **conceptual schema** hides the details of **physical storage structures** and concentrates on **describing entities, data types, relationships, and constraints.**
- Usually, a **representational data model** is used to describe the **conceptual schema** when a database system is implemented.



# External or View Level

- The **external** or **view level** includes a number of **external schemas** or **user views**.
- At the **external level,** a **database** contains several schemas that sometimes called as **subschema.**
- The **subschema** is used to describe **the different view** of the **database.**
- Each **external schema** describes the **part of the database** that a particular user group is interested in and **hides the rest of the database** from that user group.
- Each **external schema** is typically implemented using a **representational data model**, possibly **based on** an **external schema design** in a **high-level conceptual data model.**




Empno
Ename
Empno
Ename
Salary
DeptNo
EMPLOYEE
Empno : Integer(4) Key
Ename : String(15)
Salary : String (8)
Deptno : Integer(4)
Post : String (15)
STORED_EMPLOYEE record length 60
Empno : 4 decimal offset 0 unique
Ename : String length 15 offset 4
Salary : 8,2 decimal offset 19
Deptno : 4 decimal offset 27
Post : string length 15 offset 31

# Data Independence

- The **three-schema architecture** can be used to further explain the concept of **data independence**.
- **Data independence**. can be **defined** as the **capacity to change the schema at one level** of a **database** system **without having to change the schema at the next higher level.**
- We can define **two types** of **data independence:**

1. **Logical data independence**
2. **Physical data independence**

# Logical Data Independence

- **Logical data independence** is the **capacity to change the conceptual schema without** having to **change external schemas** or **application programs.**
- We may change the **conceptual schema** to **expand the database** (by adding a record type or data item), to **change constraints**, or to **reduce the database** (by **removing a record** type or data item).
- In the last case**, external schemas** that refer only to the **remaining data** should not be affected.
- Only the **view definition** and the mappings need to be changed in a DBMS that supports logical data independence.

# Logical Data Independence

- **Example:**
- If a relation **students(sid, sname, cgpa)** is replaced by:
  - **studentname(sid, sname)**
  - **studentcgpa(sid, cgpa)** for some reasons.
- Application programs that operate on the student relation can be shielded from this change by defining a view:
- **students(sid, sname, cspa)** by joining the above two relations.

# Physical Data Independence

- **Physical data independence** is the **capacity to change** the **internal schema** without having to change the **conceptual schema.**
- Hence, the **external schemas** need not be changed as well.
- Changes to the **internal schema** may be needed because some **physical files** were **reorganized—**for example, by creating **additional access structures**—to improve the performance of retrieval or update.
- If the same data as before remains in the database, we should not have to change the conceptual schema.

# Physical Data Independence

- **Example:**
- Providing an access path to improve retrieval of records by semester and year from relation **SECTION(sid, courseid, semester, year, instructure)** should not require a query such as “list all sections offered in Odd semester” to be changed,.
- Although the query can be executed more efficiently by the DBMS by utilizing the new access path.

# DBMS Languages

- The DBMS must provide appropriate languages and interfaces for each category of users.
- Once the design of a database is completed and a DBMS is chosen to implement the database.
- The first step is to specify conceptual and internal schemas for the database and any mappings between the two.
- There are three main categories of DBMS languages:

1. **Data Definition language(DDL)**
2. **Data Manipulation Language(DML)**
3. **Data Control Language(DCL)**

# DBMS Languages

- **Data Definition language(DDL)**
- **DDL** is used by database designer to specify the **conceptual schema** of a database.
- The DBMS has a **DDL compiler** whose function is to **process DDL statements**.
- In many DBMS's the DDL is also used to define internal & external schemas.
- In some DBMS separate **Storage Definition Language(SDL)** and **View Definition Language(VDL)** are used to define internal and external schemas.
- **Data Manipulation Language(DML)**
- Once the database schemas are compiled and the database is populated with data, users must have some means for **manipulating the database**.
- Typical manipulations include: **Retrieval, Insertion, Deletion** and **Modification** of Data.

# DBMS Languages

- **Data Control Language (DCL)**
- Used by **DBA(Database Administrator)** to **configure security access** to database.
- **DCL commands** are used to create **privileges** to allow users access to the database.

# DBMS Users

- There are **three types** of users:
- **Database Administrator(DBA):**
- **DBA** is the **person** most **familiar with the database** and **responsible** for **creating, modifying** and **maintaining databases, defines authorization checks, strategies** for **backup** and **recovery.**
- **Application Programmer** are **computer professionals** who **write application programs**. Application programmers can choose from **many tools** to **develop user interfaces.**
- **End Users:** are the people whose jobs require **access to the database** for **querying, updating,** and **generating reports**; the database primarily exists for their use.

# Database Design Steps