Introduction

Database System Applications
and
Purpose Of Database Systems

Introduction

- **DBMS** - Data Base Management System
- Consists of 2 parts:
  1. Database
  2. Management system

Contd..

- **Data** – raw facts that provides no standalone meaning.

  Eg: 1, Jim , 3000

- **Information** – processed or organized or interrelated data

  Eg:

S.NO	NAME	SALARY
1	Jim	3000

- **Database** – collection of interrelated data i.e., it contains information of an organization

  Eg: student register, address book, dictionary,

- **Management** – set of rules and procedures which helps to create, organize and manipulate the database.

What is a DBMS?

- Collection of interrelated data and a set of programs to access those data

Goal of DBMS:

- To provide an environment that is both *convenient* and *efficient* to use for retrieving and storing database information.

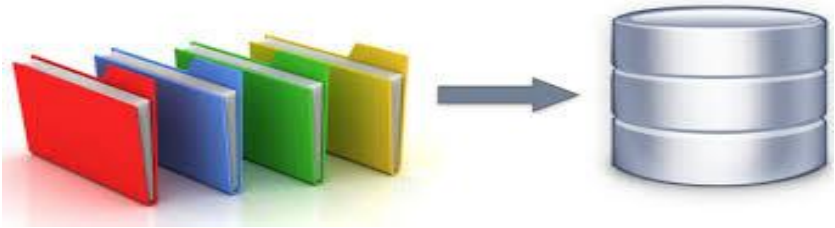
Some Examples

Database Applications

- **Banking:** all transactions
- **Airlines:** reservations, schedules
- **Universities**: registration, grades
- **Sales:** customers, products, purchases
- **Online retailers:** order tracking, customized recommendations
- **Manufacturing:** production, inventory, orders, supply chain
- **Human resources:** employee records, salaries, tax deductions
- **Databases touch all aspects of our lives**

Purpose of Database Systems

- File management system (FMS):
  - Used in earlier days
  - First method to store data in a computerized manner
  - The system stores permanent records in various files
  - It needs different application programs to extract records from and add records to the appropriate files.
  - Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems.



Contd..

- Disadvantages of FMS:
  - Data redundancy and inconsistency
  - Difficulty in accessing data
  - Data isolation
  - Integrity problems
  - Atomicity of updates
  - Concurrent access by multiple users
  - Security problems

- Database systems offer solutions to all the above problems

Advantages of DBMS

- Data independency
- Efficient data access
- Data integrity and security
- Data Administration
- Concurrent access and Crash recovery
- Reduced application development time

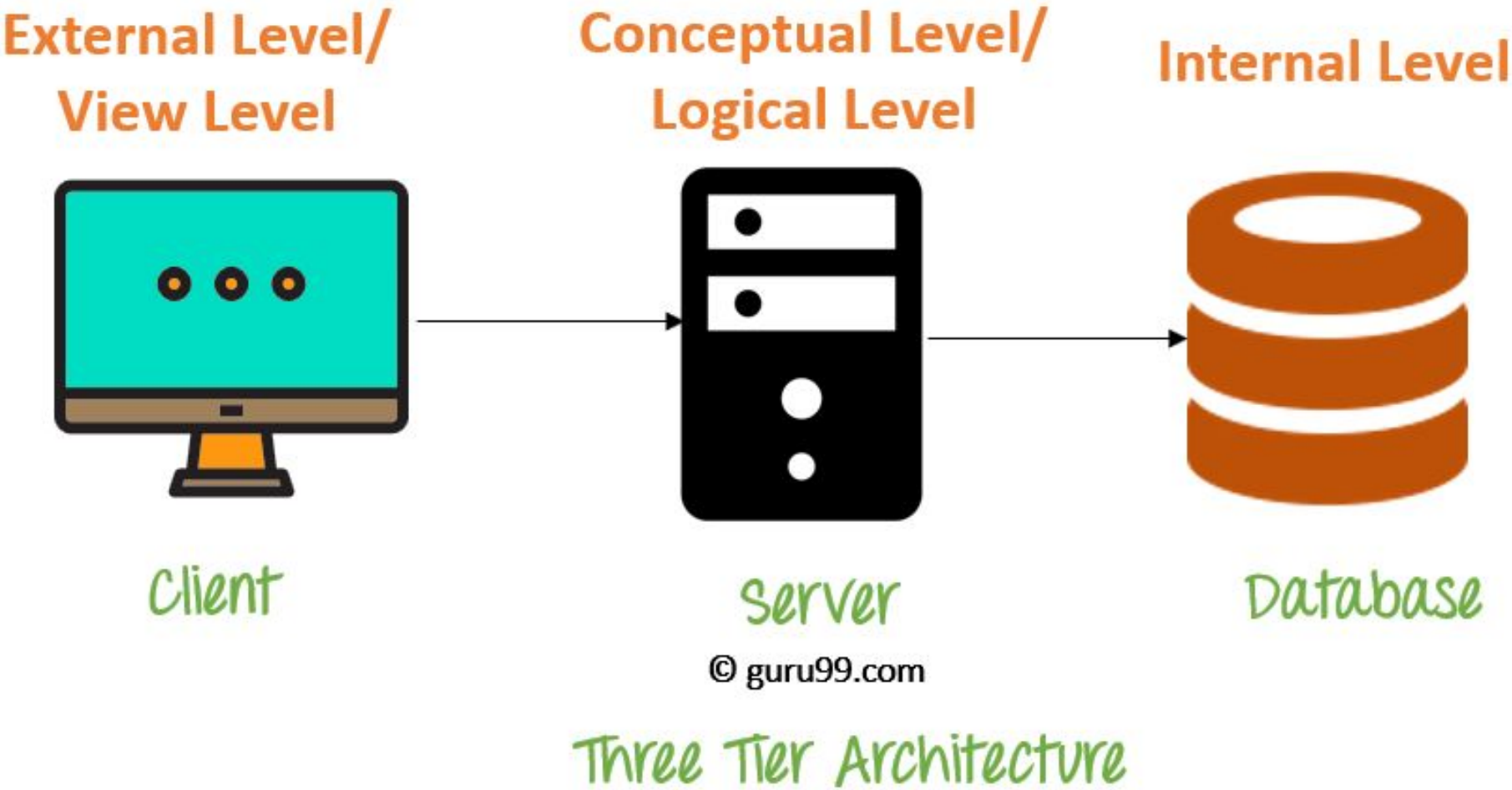
View of Data

- Database system provides – users with an abstract view of data
- System hides certain details of how the data are stored and maintained.

Data Abstraction

- Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.
- How to rate a system usable? – retrieve data efficiently
- To achieve efficiency – complex data structures are used to represent data
- Not all users of DB are experts in computer usage – developers of DB hide complexity

Three levels of data abstraction:



Levels of abstraction

- Developers hide complexity through several levels of abstraction.
  - Physical level
  - Logical level
  - View level

Physical Level

- lowest level of abstraction
- Show ***how*** data are actually stored
- Describes complex low level data structure
- The internal view tells us what data is stored in the database and how.
- It helps you to keeps information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records

Logical Level

- Next higher level of abstraction
- Shows ***what*** data are stored
- ***What*** relationship exists among the stored data
- Describes DB in simple structures
- DB admins use this level to decide what information to keep in database.

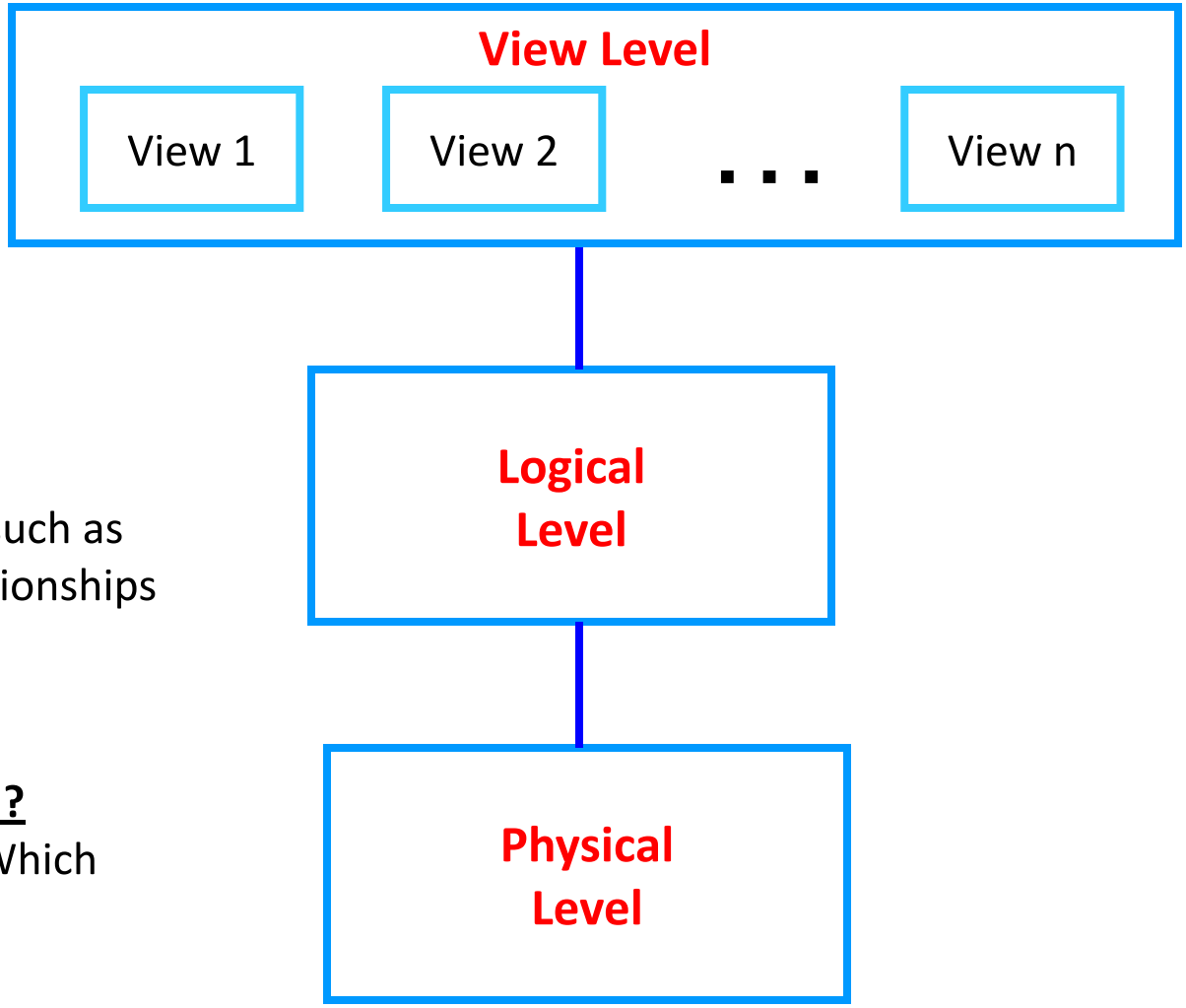
View Level

- Highest level of abstraction
- Many users of DB don’t need all such information
- The view level of abstraction simplify the interaction with the system.
- An external level is only related to the data which is viewed by specific end users.
- This level includes some external schemas.
- External schema level is nearest to the user

- This level of abstraction is reliant on software and hardware. This level is created last because designers will know the exact specifications of the database and can from there decide what exact hardware specs are needed.
- The goal of the physical level is to create a design where the physical model can be changed without affecting the internal model.

View of Data

What data users and application programs see ?



What data is stored ?

describe data properties such as data semantics, data relationships

How data is actually stored ?

e.g. are we using disks ? Which file system ?

Instances and Schemas

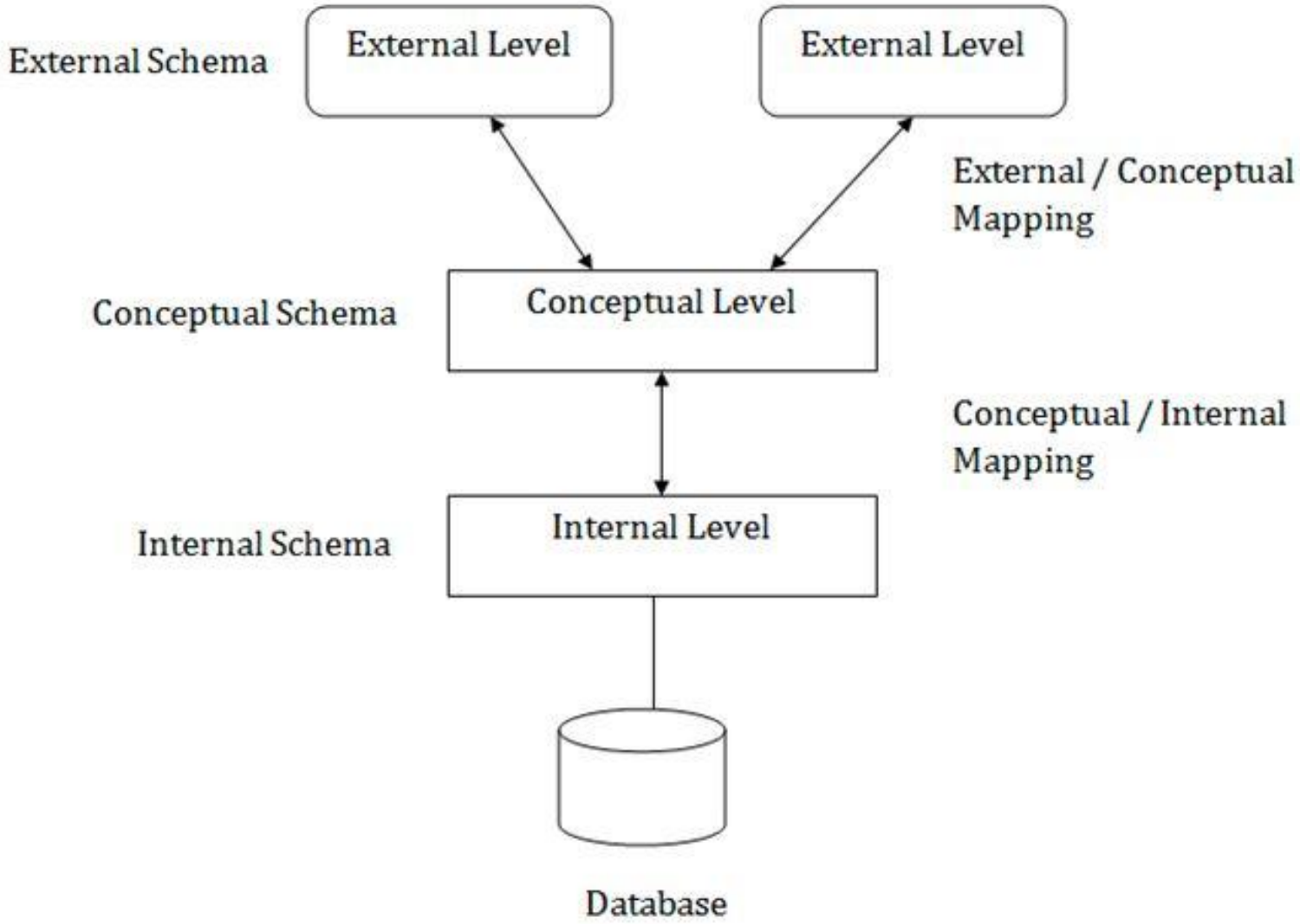
- Instance – the actual content of the database at a particular point in time
- Schema – overall design of DB, schema changes are infrequent
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
  - **Sub-schemas:** several schemas at view level to describe different views of DB

Database Schema

- A database schema is the skeleton structure that represents the logical view of the entire database.
- It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

Three schema Architecture

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.


External Schema
External Level
External Level
External / Conceptual Mapping
Conceptual Schema
Conceptual Level
Conceptual / Internal Mapping
Internal Schema
Internal Level
Database

- It shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

The External Model:

- The external model is the end users' view of the data environment.
- The term end users refers to people who use the application programs to manipulate the data and generate information. End users usually operate in an environment in which an application has a specific business unit focus. Companies are generally divided into several business units, such as sales, finance, and marketing.
- Each business unit is subject to specific constraints and requirements, and each one uses a data subset of the overall data in the organization.
- It helps to ensure security constraints in the database design. Damaging an entire database is more difficult when each business unit works with only a subset of data.
- It makes application program development much simpler.

The Conceptual Model:

- Having identified the external views, a conceptual model is used, graphically represented by an ERD to integrate all external views into a single view.
- The conceptual model represents a global view of the entire database as viewed by the entire organization. That is, the conceptual model integrates all external views (entities, relationships, constraints, and processes) into a single global view of the data in the enterprise.

- The conceptual model yields some very important advantages.
- First, it provides a relatively easily understood bird's-eye (macro level) view of the data environment.
- Second, the conceptual model is independent of both software and hardware. Software independence means that the model does not depend on the DBMS software used to implement the model.
- Hardware independence means that the model does not depend on the hardware used in the implementation of the model.
- Therefore, changes in either the hardware or the DBMS software will have no effect on the database design at the conceptual level. Generally, the term logical design is used to refer to the task of creating a conceptual data model that could be implemented in any DBMS.

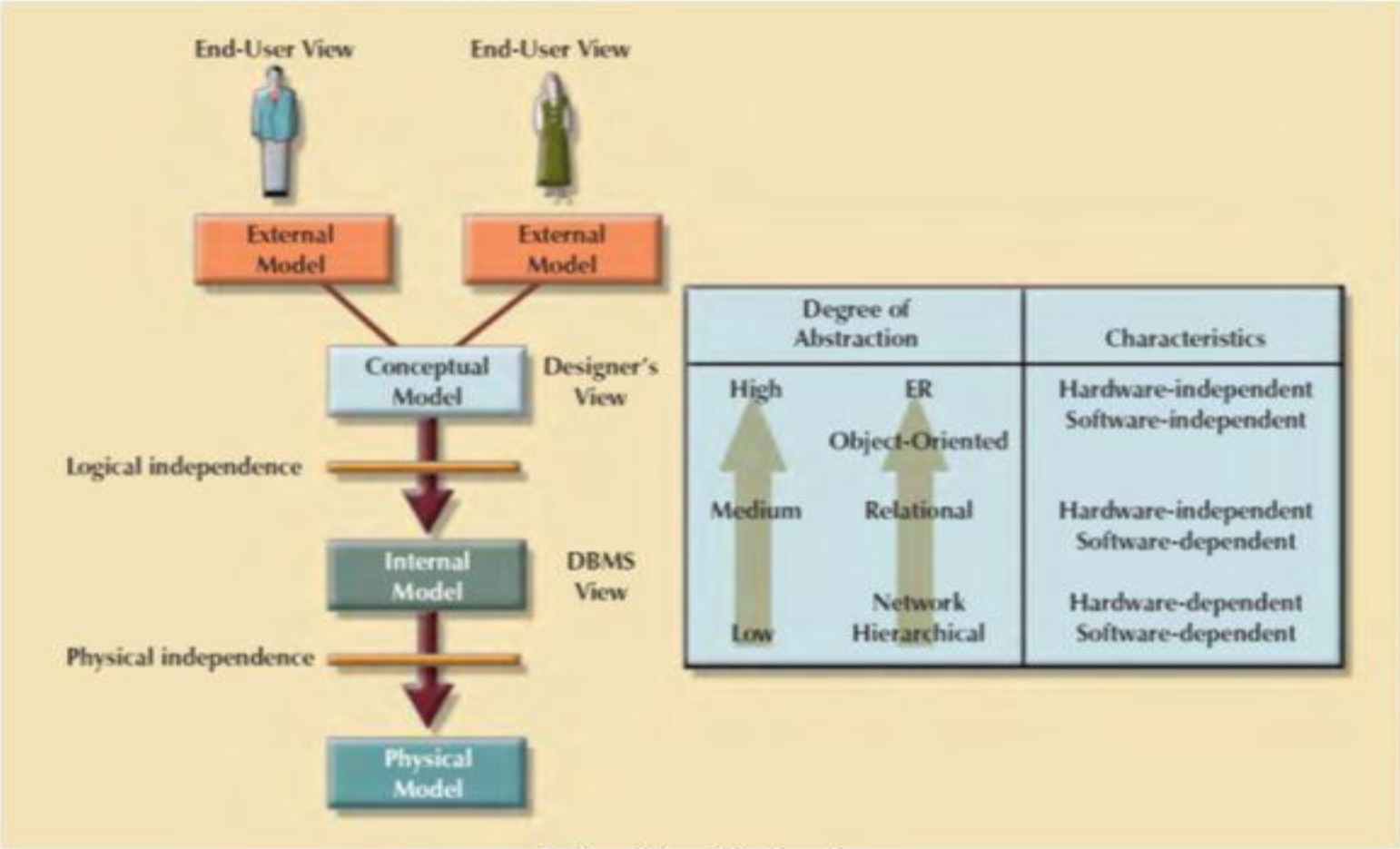
The Internal Model:

- The internal model requires the designer to match the conceptual model's characteristics and constraints to those of the selected implementation model.
- An internal schema depicts a specific representation of an internal model, using the database constructs supported by the chosen database.
- Because the internal model depends on specific database software, it is said to be software-dependent. Therefore, a change in the DBMS software requires that the internal model be changed to fit the characteristics and requirements of the implementation database model.

The Physical Model:

- The physical model operates at the lowest level of abstraction, describing the way data are saved on storage media such as disks or tapes.
- The storage structures used are dependent on the software (the DBMS and the operating system) and on the type of storage devices that the computer can handle.
- The precision required in the physical model's definition demands that database designers who work at this level have a detailed knowledge of the hardware and software used to implement the database design.
- As noted earlier, the physical model is dependent on the DBMS, methods of accessing files, and types of hardware storage devices supported by the operating system.

Degree of data Abstraction



Data abstraction levels

Database Instances

- A database instance is a state of operational database with data at any given time. It contains a snapshot of the database.
- Database instances tend to change with time.
- A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

Data Independence

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: strin g, age: integer)`
	`Courses(id: int, cname.string, credits:intege r)`
	`Enrolled(id: int, grade:string)`

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

Data Independence

- 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.

Types of Data Independence

Physical Data Independence

- Physical data independence helps you to separate conceptual levels from the internal/physical levels.
- Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels.

Examples of changes under Physical Data Independence

- Using a new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method.
- Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D Drive

Logical Data Independence

- Logical Data Independence is the ability to change the conceptual scheme without changing External views
- External API or programs
- Any change made will be absorbed by the mapping between external and conceptual levels.
- When compared to Physical Data independence, it is challenging to achieve logical data independence.

Examples of changes under Logical Data Independence

- Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
- Merging two records into one
- Breaking an existing record into two or more records

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
- It allows you to improve state which is undamaged or undivided
- Easily make modifications in the physical level is needed to improve the performance of the system.

Relational databases

- *Relational database* is a type of database that stores and provides access to data points that are related to one another.
- Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables.
- In a relational database, each row in the table is a record with a unique ID called the *key*.
- The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.