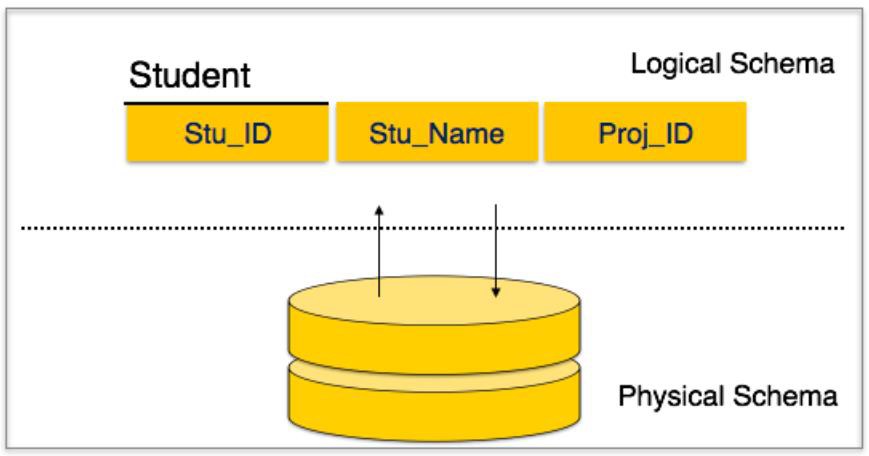
# Database Management System

Md. Raiyanul Islam Lecturer,

Dept. of CSE

Dhaka International University

* ***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.
* A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.
* It’s the database designers who design the schema to help programmers understand the database and make it useful.
* A database schema can be divided broadly into two categories −



**Physical Database Schema** − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.

**Logical Database Schema** − This schema defines all the logical constraints that need

to be applied on the data stored. It defines tables, views, and integrity constraints.

**Database Instance**

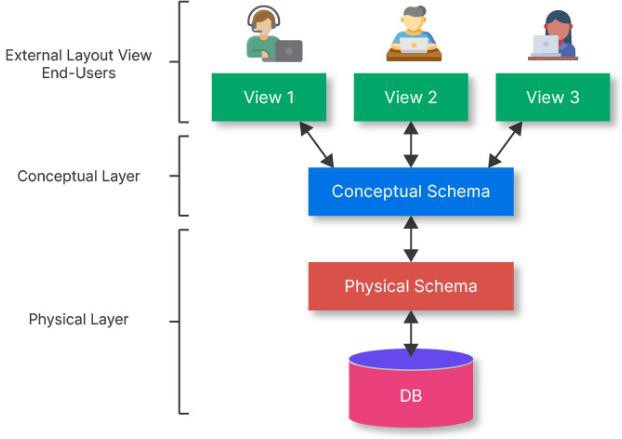
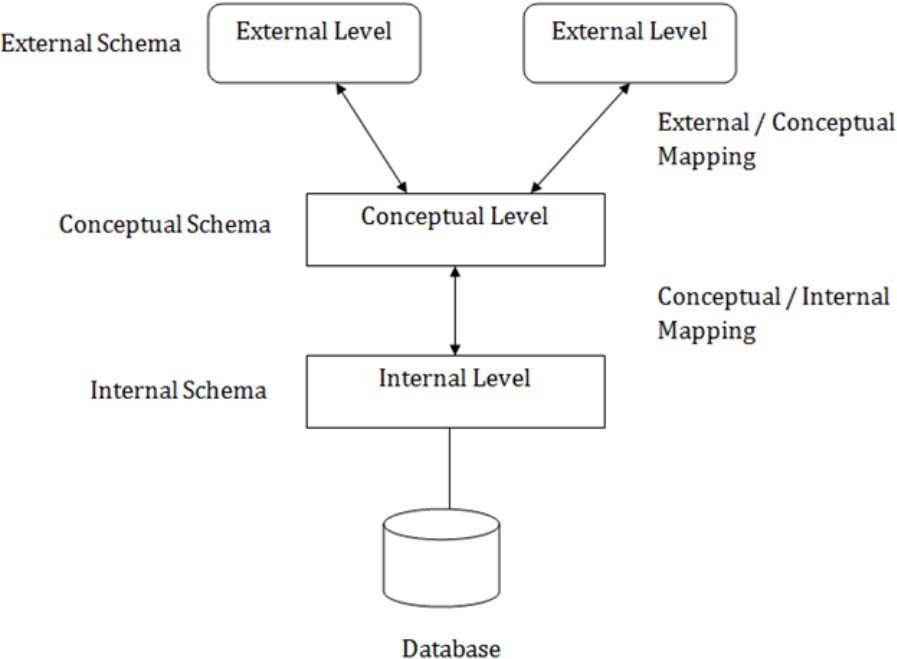
A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions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

# Schema

* Student
  + roll no
  + reg no
  + email
  + name
* Course
  + Course code or id
  + Course title
  + Credit
* Teacher
  + Name
  + Email
  + Designation
* Logical representation of the DB
* It can have a single table or a collection of tables
* 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.



Objectives of 3-schema Architecture

## The main objective of three level architecture is to enable multiple users to access the same data.

* Different users need different views of the same data.
* The approach in which a particular user needs to see the data may change over time.
* The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
* All users should be able to access the same data according to their requirements.
* DBA should be able to change the conceptual structure of the database without

affecting the user’s activity.

* Internal structure of the database should be unaffected by changes to physical aspects of the storage.

The three-level architecture of a database, also known as the three-schema architecture, is a conceptual framework that describes the organization and structure of a database system. This architecture helps in separating the database into three distinct levels, each with its own purpose and abstraction. These levels are:

1. **External Level** (User View):
   1. The external level is the topmost layer of the three-level architecture and is also known as the user view or user interface level.
   2. This level is concerned with the way users interact with the database. It defines various user views or user interfaces that cater to the specific needs and requirements of different types of users, such as end-users, application programmers, and database administrators.
   3. Each user view presents a subset of the data from the overall database, showing only the relevant information to the users.
   4. Users at this level are typically unaware of the internal structure of the database and interact

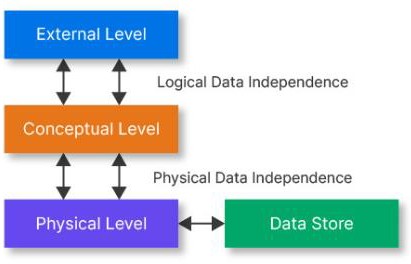
with it using high-level query languages and applications.

1. **Conceptual Level** (Logical Schema):
   1. The conceptual schema is the middle level of the three-level architecture.
   2. It represents the overall logical interface level and organization of the entire database system, independent of any specific user's view or application.
   3. At this middle layer, the data model is defined, which includes the schema (simple structure) of the entire database, relationships between data elements, integrity constraints, and security rules.
   4. The conceptual schema provides a global and integrated view of the data, ensuring data consistency and integrity across different user views.
   5. Changes to the conceptual schema affect all user views, but users at the external level are shielded from these changes.
2. **Internal Level** (Physical Schema):
   1. The internal level is the lowest layer of the three-level architecture, also known as the physical schema.
   2. It deals with the physical storage and internal implementation of data on the underlying storage devices

(such as hard drives or solid-state drives).

* 1. This level involves decisions related to data storage structures, indexing methods, data compression, and access paths for optimizing data retrieval and storage efficiency.
  2. The internal schema may be different from the conceptual schema, as it is optimized for performance and storage considerations rather than representing the logical structure of the data.
  3. Changes at this level, such as storage optimizations or database reorganization, do not impact the

Data independence in DBMS refers to the capacity to change the schema (structure) of the database without affecting the application programs or user views that access the data. It is a fundamental concept that simplifies database maintenance and enhances flexibility.



**Why**

Data independence simplifies database maintenance and management by reducing the impact of changes. It allows for greater flexibility in adapting to evolving requirements, reduces the risk of errors during schema modifications, and makes it easier to manage large and complex databases.

**Physical Data Independence** can be defined as the ability to change the physical level without affecting the logical or Conceptual level. Physical data independence gives us the freedom to modify the - Storage device, File structure, location of the database, etc. without changing the definition of conceptual or view level.

Example: For example, if we take the database of the banking system and we want to scale up the database by changing the storage size and also want to change the file structure, we can do it without affecting any functionality of logical schema.

**Logical Data Independence** is a property of a database that can be used to change the logic behind the logical level without affecting the other layers of the database. Logical data independence is usually required for changing the conceptual schema without having to change the external schema or application programs. It allows us to make changes in a conceptual structure like adding, modifying, or deleting an attribute in the database.

Example: If there is a database of a banking system and we want to add the details of a new customer or we want to update or delete the data of a customer at the logical level data will be changed but it will not affect the Physical level or structure of the database.

* **Flexibility:** Data independence allows for changes to be made in the database schema (structure) without affecting the way data is accessed or presented to users. This flexibility makes it easier to adapt the database to evolving requirements and business needs.
* **Application Compatibility:** Changes to the logical schema do not impact the application programs or queries that rely on the database. This means that existing applications can continue to function correctly even when the database structure changes, reducing the risk of disruptions.
* **Easier Maintenance:** Database administrators can perform routine maintenance tasks, such as reorganizing data for performance optimization or implementing security updates, without disrupting user access or application functionality.
* **Data Continuity:** When migrating data to new storage technologies or platforms, data independence ensures that the logical schema remains consistent, preserving data continuity and application functionality.
* **Scalability:** As the database grows, data independence facilitates the addition of new data elements or tables without affecting existing queries or applications. This scalability is crucial for accommodating increasing data volumes.
* **Reduced Development Time:** Developers can focus on designing and building applications without needing to worry about changes in the underlying database structure. This separation of concerns can lead to faster development cycles.
* **Ease of Integration:** Data independence simplifies the integration of data from multiple sources into a single database storage system. External schemas can be defined to provide unified views of the data, regardless of its source or format.
* **Data Integrity:** Changes to the logical schema can be managed carefully to ensure data integrity and consistency. Referential integrity constraints and validation rules can be applied at the logical level to maintain data quality.
* **Adaptation to Technology Changes:** As technology evolves, the physical storage and organization of data may need to change to take advantage of new hardware or software capabilities. Data independence allows these changes to be made without affecting the logical schema.
* **Reduced Risk:** By minimizing the impact of schema changes, data independence reduces the risk of errors and data corruption

**Complexity:** Implementing data independence can add complexity to the database system, as it involves managing multiple levels of schemas (external, conceptual, and internal). This complexity can make the database design and maintenance more challenging, particularly in large and complex database systems.

**Performance Implications:** Achieving complete physical data independence, where changes to the physical schema have no impact on performance, can be challenging. Certain physical optimizations may be closely tied to the logical structure, making it difficult to implement changes without affecting database performance.

**Resource Overhead:** Maintaining data independence may require additional resources, such as disk space and processing power, especially when managing multiple layers of schemas. This overhead can affect database system server performance and scalability.

**Potential for Redundancy:** In some cases, achieving data independence may lead to redundancy in data storage. For example, if different external schemas require the same data to be stored in multiple formats or database locations, it can result in increased storage requirements and synchronization challenges.

**Migration Complexity:** While data independence facilitates schema changes, it may not eliminate all complexities associated with schema migrations. Migrating data between different versions of the database or across different DBMS platforms can still be a non-trivial task.

**Compatibility Issues:** Changes made to the logical or conceptual schema may not always be compatible with existing application programs or queries. In some cases, backward compatibility efforts may be required to ensure that legacy applications continue to work correctly.

**Potential for Data Integrity Issues:** Changes in the logical schema, if not carefully managed, can lead to data integrity problems. Ensuring that data remains consistent and that referential integrity constraints are maintained can be challenging when altering the logical schema.

**Development and Maintenance Effort:** Implementing data independence often requires careful planning, documentation, and adherence to best practices. It can involve additional development process and maintenance efforts to create and manage various schema layers and ensure that changes do not introduce errors.

**Training and Expertise:** Database administrators and developers may require specific training and expertise to effectively manage data independence in a DBMS. Understanding how changes at one level of the schema affect other levels is crucial for maintaining data integrity.

Thank You