Chapter 2

Database System Concepts and Architecture

Content

- Data Model, Schema and Instance
- Three schema architecture and data independence
- Database languages & Interfaces
- Database systems environment

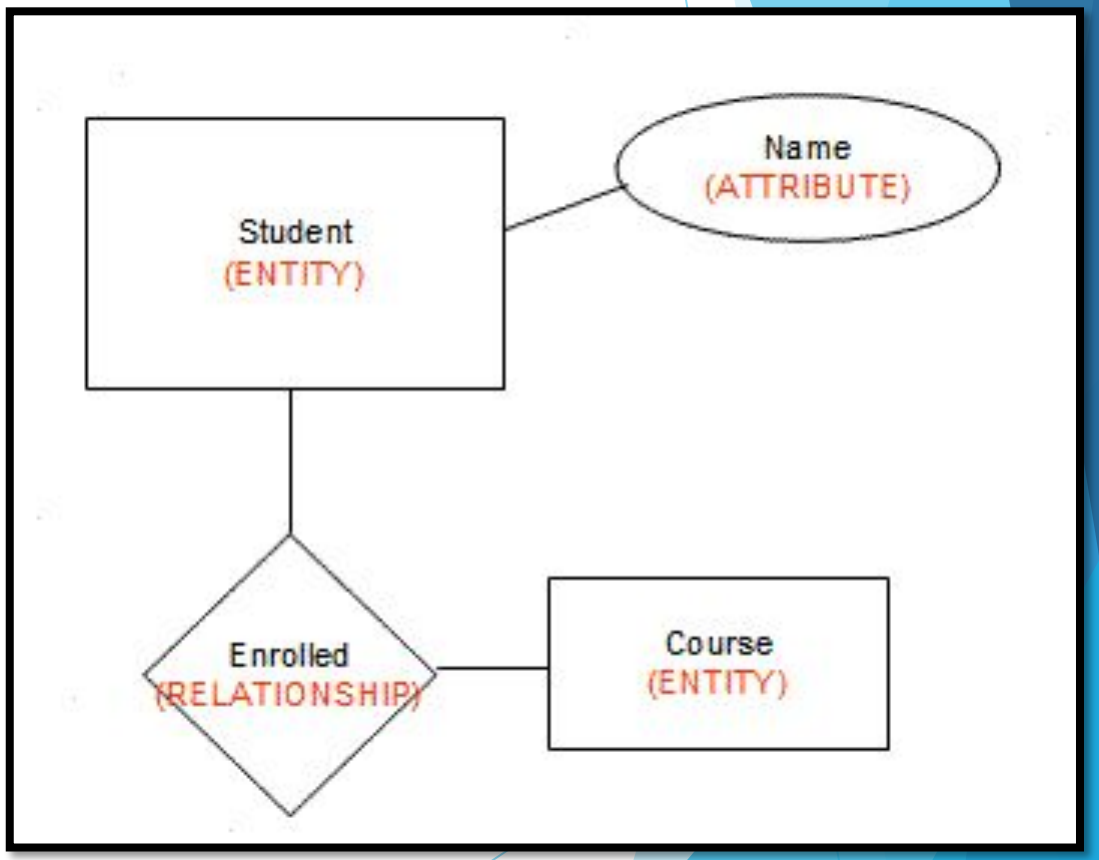
- Classification of DBMS

Data Models, Schemas, and Instances

- **Data abstraction :** hiding the storage and data organization details & highlight essential features only.
- **data model**—a concept that describes the structure of a database to achieve abstraction. Some data models also show manipulation operations.
- Data Models can be divided into 3 categories:
  - Conceptual Data Model: **defines WHAT the system contains**.
    - **entities, attributes & relationship**
    - Created by stakeholders & data architects
    - Purpose: Create business rules
  - Logical Data Model: **Defines HOW the system should be implemented regardless of the DBMS.**
    - Done by BA and data architects
  - Physical Data Model: **describes HOW the system will be implemented using a specific DBMS.**
    - Create schemas , mappings
    - Created by DBA, developers
    - Actual implementation of DB

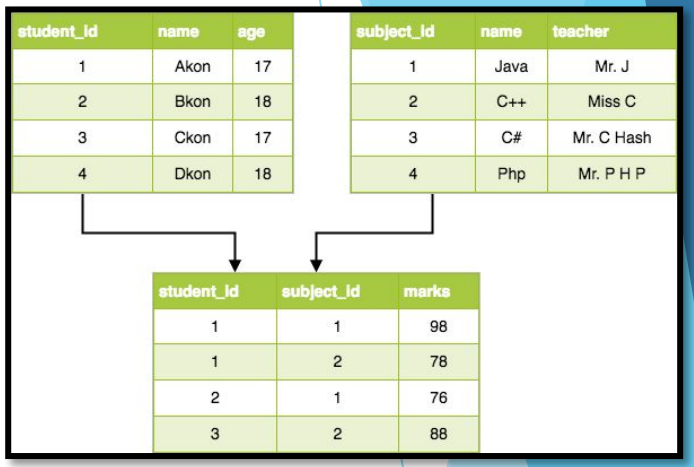
Data Models, Schemas, and Instances

- **Categories of Data Models:**
- **High-level or conceptual data models such as entity-relationship model** provide concepts that are easy for users to understand.
- Conceptual data models use concepts such as **entities, attributes, and relationships**.
- An **entity** represents a real-world object or concept, such as a student or a course.
- An **attribute** further describes an entity, such as the student's name or age.
- A **relationship** among two or more entities represents an association among the entities, for example, a student is enrolled in a course.
- 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**.



Data Models, Schemas, and Instances

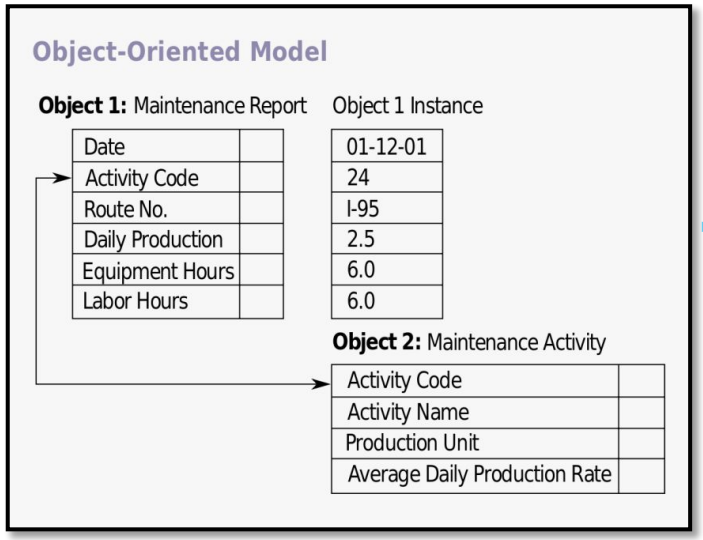
- **Representational or implementation data models** are used most frequently in traditional commercial DBMSs. These include the widely used **relational data model**.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

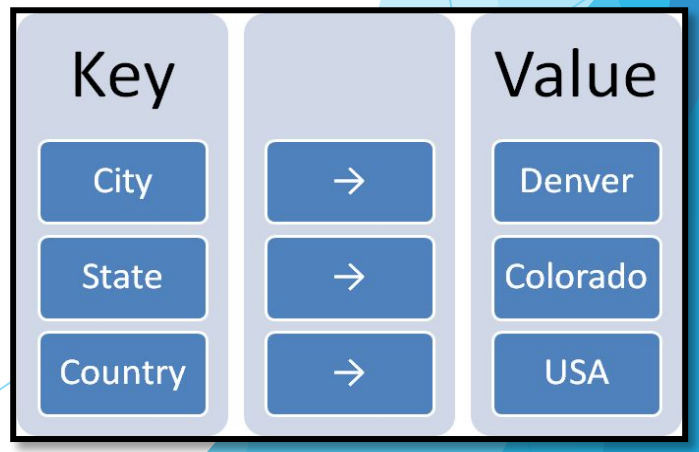
subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88



- Object data model is an example of a higher-level implementation data model that is closer to conceptual data models.

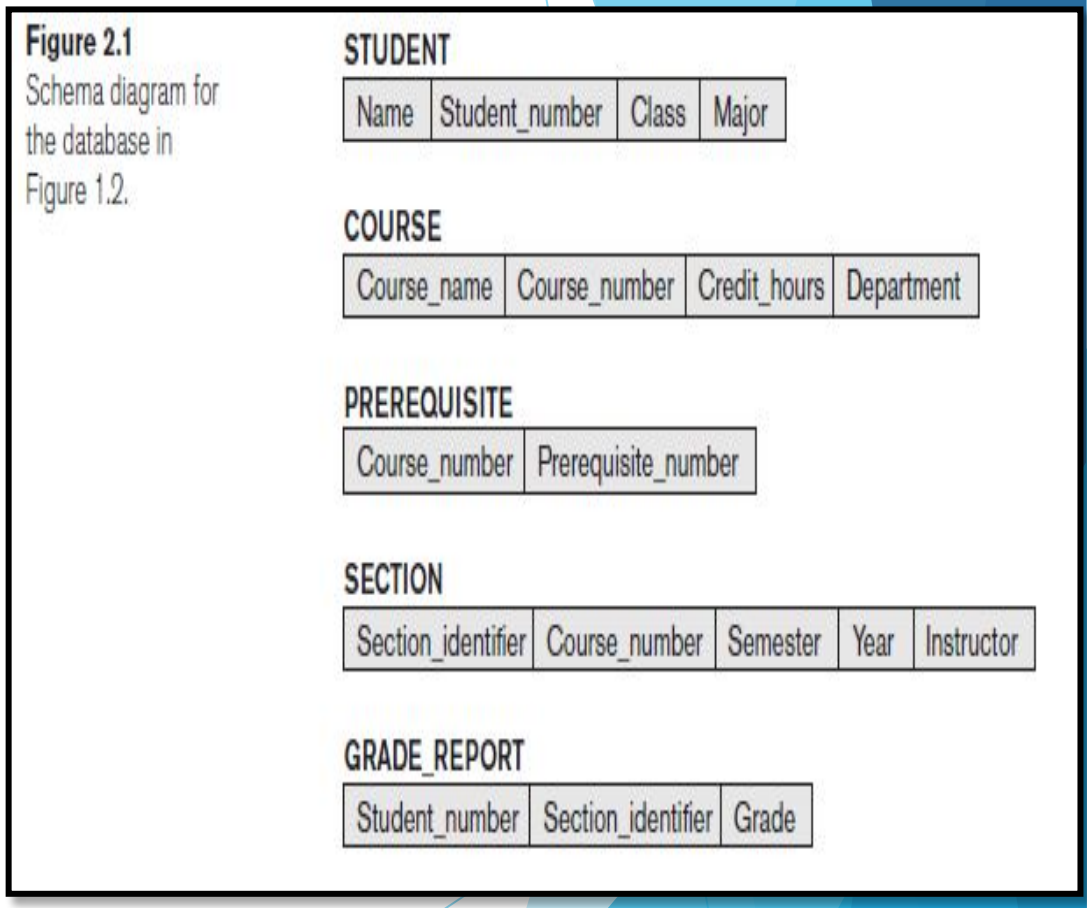
- **Self-describing data models** - The data storage in these models combines the description of the data with the data values. These models include XML, key-value stores, and NOSQL systems



Data Models, Schemas, and Instances

Schemas, Instances, and Database State

- The **description of a database is called the database schema**, which is specified during database design.
- A displayed schema is called a **schema diagram** shown in figure. The diagram displays the structure of each record type but not the actual instances of records.
- Each object in the schema—such as STUDENT or COURSE is **a schema construct**.
- A schema diagram displays only some aspects of a schema, such as the **names of record types and data items**. Other aspects such as the **data type** of each data item nor the **relationships** among the various files are not specified.



Data Models, Schemas, and Instances

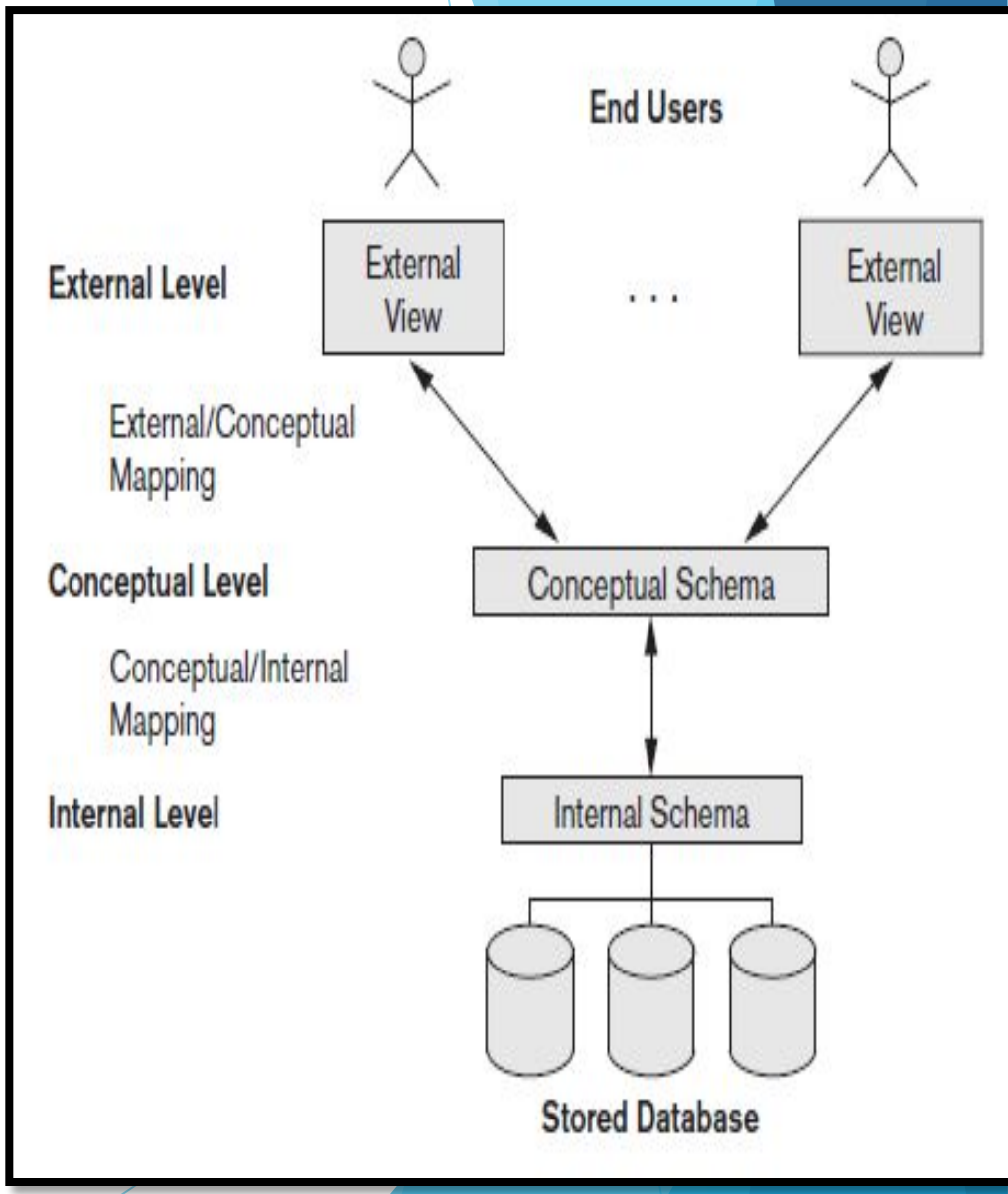
- **Schemas, Instances, and Database State**
- The actual data in a database may change quite frequently. For example, the database changes every time we add a new student or enter a new grade.
- The data in the database at a particular moment in time is called a database state or snapshot.

- Difference between database schema and database state.
- When we define a new database, we specify its **database schema** only to the DBMS. At this point, the database state is the **empty state**.

- **Initial state** - when the database is first populated with the initial data.
- **Current state** – the state right now
- **Valid state** — a state that satisfies the structure and constraints specified in the schema.
- The schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.
- **Instance** - an individual record in a database
- **Schema evolution** – very rare, but needed because of change in business requirements.

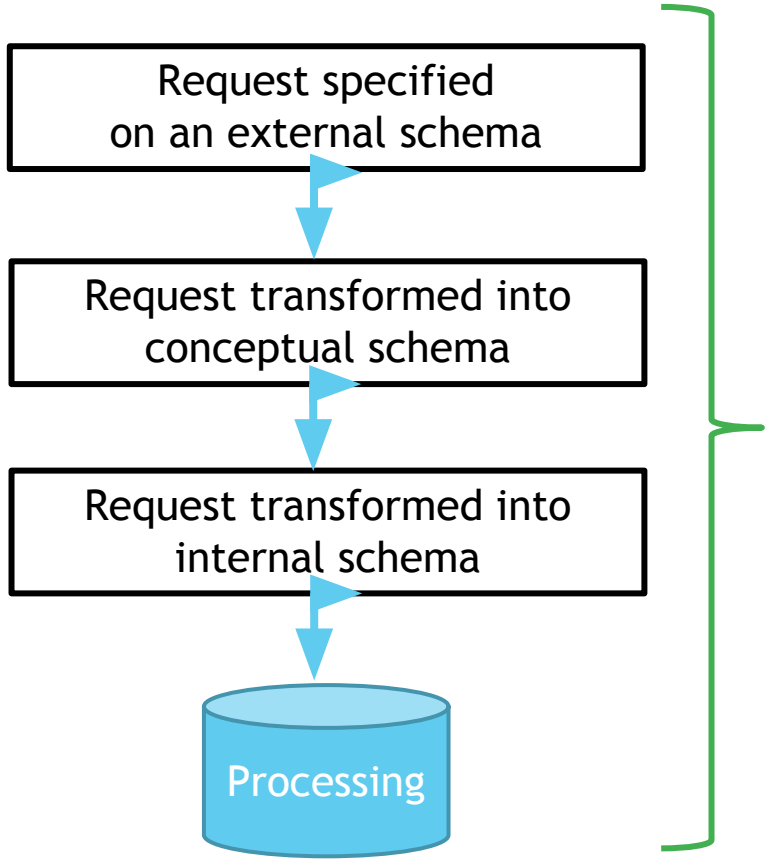
Three schema architecture and data independence

- **The Three-Schema Architecture**
- The goal of the three-schema architecture is data abstraction from user, i.e., user is independent of how & where data is stored.
- Internal Level - The internal level has an internal schema, which describes the physical storage structure of the database.
  - The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
- Conceptual level - The conceptual level has a conceptual schema, which describes the structure of the whole database.
  - The conceptual schema hides the details of physical storage structures and **concentrates on describing entities, data types, relationships, user operations, and constraints**. e.g., ER model.
- External level - The external or view level includes a number of external schemas or user views depending on user roles.
  - 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.
- The three schemas are only descriptions of data; the actual data is stored at the physical level only.

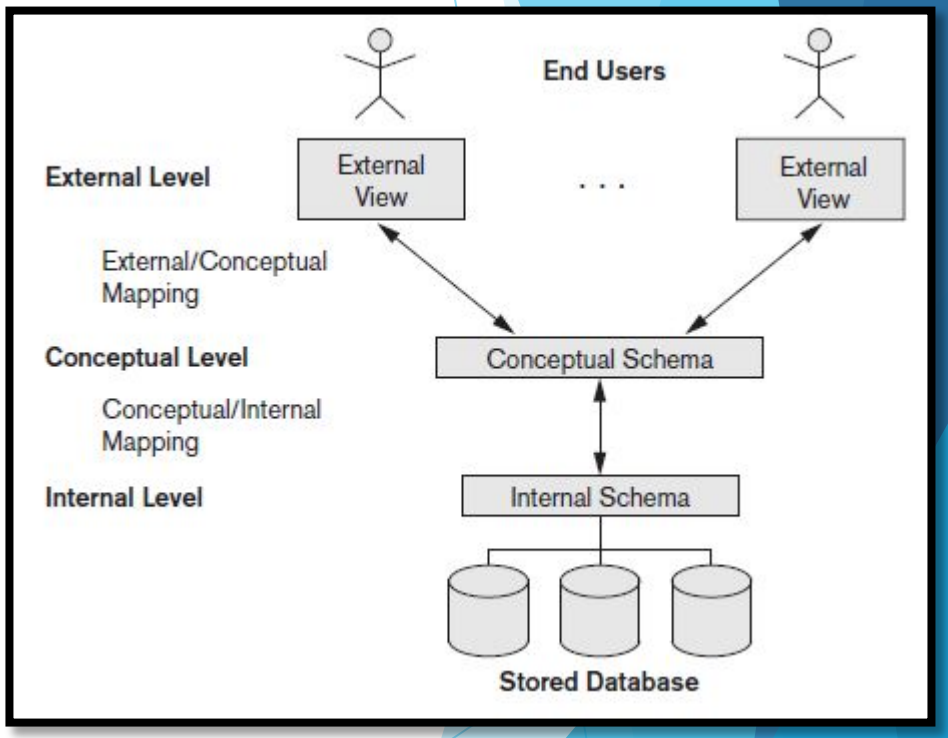


Three schema architecture and data independence

- The Three-Schema Architecture



The processes of transforming requests and results between levels are called mappings.



Three schema architecture and data independence

- **Data Independence**
- Data independence – the **capacity to change the schema at one level of a database system without having to change the schema at the next higher level**.
- **Logical data independence** is the capacity to *change the conceptual schema without having to change external schemas* .
  - We may change the conceptual schema (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).
  - After the conceptual schema changes, application programs that reference the external schema constructs must work as before.
  - **Changes to constraints** can be applied to the conceptual schema without affecting the external schemas or application programs.

Three schema architecture and data independence

- **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.
- Like changing file location/access paths, compressing and saving data records.

Which data independence is harder to achieve?
Physical or logical? Why?
logical independence is difficult to achieve