Database System Concepts and Architecture

Part 1

Data Models, Schemas, and Instances

- **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**—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

- **High-level** or **conceptual data models** provide concepts that are close to the way many users perceive data, whereas **low-level** or **physical data models** provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks.
- Between these two extremes is a class of **representational** (or **implementation**) **data models**, which provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage.

- Conceptual data models use concepts such as entities, attributes, and relationships.
- A popular high-level conceptual data model is the **Entity-Relationship model**.

- 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.
- Representational data models represent data by using record structures and hence are sometimes called **record-based data models**.

- We can regard the **object data model** as an example of a new family of higher-level implementation data models that are closer to conceptual data models.
- Object data models are also frequently utilized as high-level conceptual models, particularly in the software engineering domain.

- 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 structure that makes the search for particular database records efficient.
- An **index** is an example of an access path that allows direct access to data using an index term or a keyword.

Schema, Instances, and Database State

- In any 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.
- Most data models have certain conventions for displaying schemas as diagrams.
- A displayed schema is called a **schema diagram**.

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

- The actual data in a database may change quite frequently.
- The data in the database at a particular moment in time is called a **database state** or **snapshot**.
- It is also called the *current* set of **occurrences** or **instances** in the database.

- The distinction between database schema and database state is very important.
- When we **define** a new database, we specify its database schema only to the DBMS.
- At this point, the corresponding database state is the *empty state* with no data.
- We get the *initial state* of the database when the database is first **populated** or **loaded** with the initial data.
- From then on, every time an update operation is applied to the database, we get another database state.

- At any point in time, the database has a *current state*.
- The DBMS is partly responsible for ensuring that every state of the database is a **valid state**—that is, a state that satisfies the structure and constraints specified in the schema.
- Hence, specifying a correct schema to the DBMS is extremely important and the schema must be designed with utmost care.
- The DBMS stores the descriptions of the schema constructs and constraints—also called the **meta-data**—in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to.
- The schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.

- Although, as mentioned earlier, the schema is not supposed to change frequently, it is not uncommon that changes occasionally need to be applied to the schema as the application requirements change.
- This is known as **schema evolution**.