Lecture 2: Database Models & Architecture

1. Database Model (Data Model)

- **Definition:** A type of data model that determines the logical structure of a database. It dictates how data can be stored, organized, and manipulated.
- **Usage:** Provides a framework for database designers and developers. The most common model is the **relational model**.

2. Basic Terminologies

- **Entity:** A tangible or intangible object (e.g., person, place, event) about which data is stored.
- **Attribute:** A characteristic or property of an entity (e.g., `Name`, `ID` for a `STUDENT` entity).
- **Relationship:** An association between two or more entities. Types include **1:1** (one-to-one), **1:M** (one-to-many), and **M:N** (many-to-many).
- **Constraints:** Rules and restrictions placed on the data to ensure its integrity and accuracy.
- **Business Rules:** Precise, unambiguous descriptions of an organization's policies and procedures. They are crucial for identifying entities, attributes, relationships, and constraints.
    - **Rule:** Nouns become entities; verbs become relationships.

3. Types of Database Models (Comparison)

Model	Structure	Key Feature	Pros	Cons
Hierarchical	Tree (Root & Nodes)	One parent per child (1:M)	Conceptual simplicity, Integrity, Efficiency for large DBs	Rigid structure, Can't handle M:N, Complex to manage

Network	Graph (Nodes & Links)	Handles M:N relationships	High-speed retrieval, Handles complex relationships	No SQL support, Complex navigational access, Hard to modify
Relational	Tables (Rows & Columns)	Based on relational algebra & SQL	Ease of use (SQL), Data sharing, Controls redundancy, Powerful DBMS	High hardware/software cost, Can be complex, Large size
Object-Oriented (OODB)	Objects (Data + Methods)	Encapsulation & Inheritance	Semantic representation, Handles complex data (e.g., multimedia), Reusability	Lack of standards, Complex design, Slow transactions, High overhead

---

4. Relational Model Specifics (ER Model)

- **Table = Relation**, **Row = Tuple**, **Column = Attribute**.
- **Rules:**
    - Column values are atomic.
    - Each row must be unique.
    - Each column has a unique name and a domain (set of possible values).
    - The order of rows and columns is insignificant.
- **Keys:**
    - **Primary Key:** A unique identifier for a row.
    - **Foreign Key:** An attribute that creates a link between two tables.
- **Entity-Relationship Diagram (ERD):**
    - **Entity:** Rectangle (e.g., `STUDENT`).
    - **Attribute:** Oval, connected to its entity.
    - **Relationship:** Diamond, connecting entities.

- **Cardinality:** Defines the numerical relationship (1:1, 1:M, M:N) between entities.
- **Degree:** Number of entities participating in a relationship (Unary, Binary, Ternary, N-ary).

---

5. Database Schema vs. Database State

- **Database Schema:**
  - **Definition:** The **blueprint** or design of the database. It includes descriptions of structure, data types, and constraints.
  - **Analogy:** The structure of a variable (e.g., `int count;`).
  - **Changes:** Very infrequently (is **intension**).
- **Database State (Instance):**
  - **Definition:** The actual **data** stored in the database at a specific moment in time.
  - **Analogy:** The value of a variable (e.g., `count = 5;`).
  - **Changes:** Every time the database is updated (is **extension**).
- **Valid State:** A database state that satisfies all the structure and constraints defined in the schema.

---

6. Three-Level Architecture (ANSI/SPARC)

This architecture provides **data independence** by separating the user applications from the physical database.

Level	Description	Focus	Audience
1. External Level	**User Views.** Multiple customized views of the data relevant to specific users.	End-User Needs	Application programmers, end-users
2. Conceptual Level	**Community View.** A complete description of the entire database information structure	Global, logical view of all data	Database Administrators (DBA)

	(entities, attributes, relationships, constraints).		
3. Internal Level	**Physical View.** Describes how the data is physically stored (file structures, indexes, compression).	Storage efficiency, performance	DBMS developers/system programmers

- **Mappings:** The DBMS uses mappings to translate requests between levels.
    - **External/Conceptual Mapping:** Links external views to the conceptual schema.
    - **Conceptual/Internal Mapping:** Links the conceptual schema to the physical storage.

---

7. Data Independence

The immunity of user applications to changes made in the database structure. This is the primary goal of the three-level architecture.

- **Logical Data Independence:**
    - **Definition:** The capacity to change the **conceptual schema** without having to change **external schemas** or application programs.
    - **Example:** Adding a new entity or attribute to the overall database without affecting existing user views.
- **Physical Data Independence:**
    - **Definition:** The capacity to change the **internal schema** without having to change the **conceptual schema**.
    - **Example:** Changing the file organization (e.g., from a hash to a B-tree index) or adding data encryption without affecting the logical structure of the database.