DBMS CS 502 Assignment: UNIT-1

Name: Aryan Singh Chandel
Roll No.: 0902CS231028

Submission Deadline: 29/09/2025

1. Data Independence in DBMS

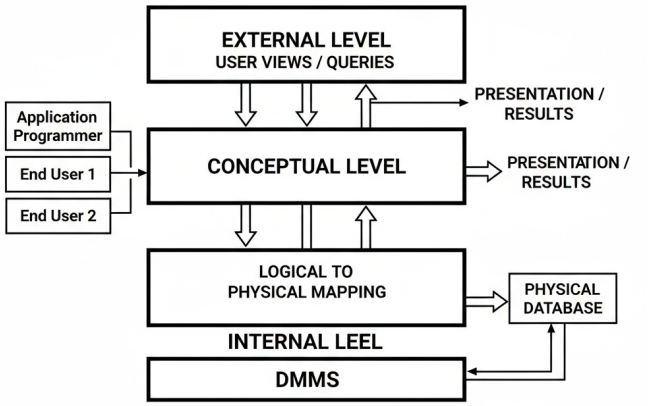
Data Independence is the capacity to modify the schema at one level in a database system without requiring changes at the next higher level. This property makes a database flexible and resistant to change, facilitating robust growth and low maintenance.

Importance:

- Reduces impact of changes across the stack.
- Allows performance optimizations and schema evolution without code rewriting.
- Supports organizational growth and collaboration.

DBMS Three-Level Architecture:

1. **Internal Level:** Physical storage and organization.
2. **Conceptual Level:** Logical structure (tables, links).
3. **External/View Level:** User/application view of data.



Types of Data Independence:

a) **Logical Data Independence:** Changing conceptual schema without altering external schemas or applications.
Example: Adding `email` to Employees table; combining Department and Projects into a new relationship.

b) **Physical Data Independence:** Changing internal schema without affecting logical schema.
Example: Moving files to different drives, changing index structure, compressing data storage.

Aspect	**Physical Data Independence**	**Logical Data Independence**
Concerned Level	Internal Storage	Conceptual Schema
Example Change	File move, Index	Add column, Merge tables
Effect on User	None	None
Main Use	Storage, performance	Business adaptivity

Summary:
Physical independence enables invisible storage upgrades. Logical independence allows for flexible business logic and user views that survive changes to structure.

Real World Example:
If a bank adds a `Branch_Code` field to their internal branch table, users continue banking as usual. When the DBA upgrades the server with faster SSDs, users notice faster service but no disruption.

2. Define Schema and Subschema. Illustrate with an example how subschema provides different user views in DBMS.

Definition of Schema:
A **schema** in DBMS is the overall logical design or blueprint of the entire database. It describes the organization of all data elements, tables, fields, relationships, constraints, and structure. The schema is defined by the database administrator and remains relatively stable.

Definition of Subschema:
A **subschema** is a tailored or partial view of the overall schema, created for particular users or applications. Subschemas show only the specific part of the database relevant for a particular group, restricting access to sensitive or irrelevant data.

Key Differences (Table)

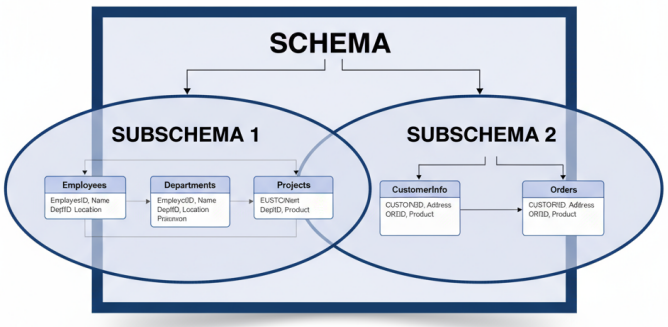
Aspect	Schema	Subschema
Definition	Complete logical structure of the database	Tailored view for users/applications
Scope	Entire database (all tables, fields, relationships)	Subset of database (only relevant tables/fields)
Visibility	Seen by DBA and developers	Seen by end users or specific applications
Change Frequency	Rare	Can be created/modified frequently
Example	College DB with Students, Courses, Exams, Fees	Academic office sees only Students, Courses; Accounts sees Students, Fees

Illustrative Example

Consider a University Management System:

- **Schema** includes all entities: Students, Courses, Fees, Exams, Staff, Results, Library.
- **Subschema for Academic Section**: Shows only Students, Courses, Exams.
- **Subschema for Accounts Department**: Shows only Students, Fees.
- **Subschema for Library**: Shows only Students, Library records.

This approach protects sensitive data by hiding unrelated records and streamlines data handling for each department.



Key Points

- Subschema ensures security, privacy, and relevance for different users.
- Multiple subschemas can be defined for the same database schema.
- Application developers can code with subschemas, lowering complexity.

3. Hospital E-R Diagram and Relational Tables

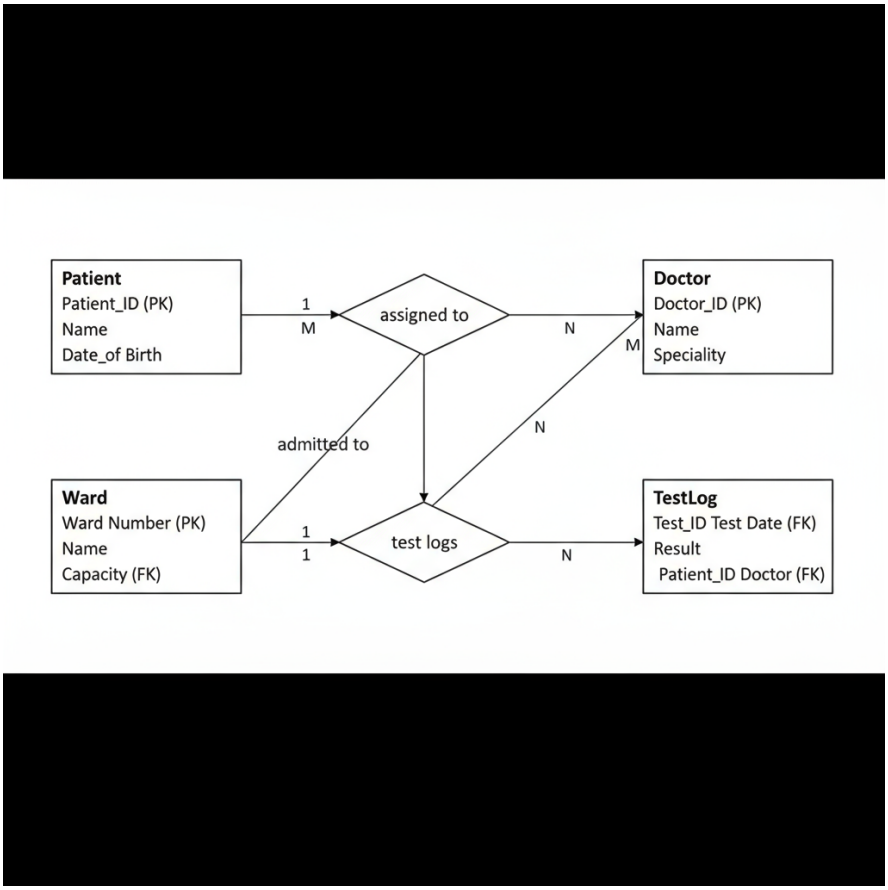
ER (Entity-Relationship) Diagram for a Hospital Database:

The hospital database consists of several entities and their interrelationships. Key entities include:

- **Patient:** Patient_ID (PK), Name, DOB, Address, Gender, Phone.
- **Doctor:** Doctor_ID (PK), Name, Specialization, Phone.
- **Ward:** Ward_ID (PK), Type, Capacity.
- **TestLog:** TestLog_ID (PK), Patient_ID (FK), Test_Date, Test_Type, Result.

Relationships:

- Patients are *admitted to* Wards.
- Doctors are *assigned to* Patients.
- TestLogs are *associated with* Patients.



(ER diagram showing entities Patient, Doctor, Ward, TestLog with connecting relationships.)

Conversion of ER Diagram to Relational Tables:

1. Each entity → a table (Primary key becomes underlined attribute).
2. Each relationship → a foreign key or linking/intermediate table (depending on cardinality).

Table Name	Primary Key	Foreign Keys
Patient	Patient_ID	–
Doctor	Doctor_ID	–
Ward	Ward_ID	–
TestLog	TestLog_ID	Patient_ID (references Patient)
PatientDoctor	Patient_ID + Doctor_ID	Patient_ID, Doctor_ID
PatientWard	Patient_ID + Ward_ID	Patient_ID, Ward_ID

Explanation:

- Each strong entity gets its own table with a suitable primary key.
- The TestLog table references a Patient using a foreign key.
- Many-to-many relationships (e.g., a patient has multiple doctors and vice versa) are captured via intersection tables such as `PatientDoctor`.
- One-to-many relationships (e.g., each patient is admitted to a ward) use a foreign key or linking table.

Example Rows:

- **Patient:** `(P001, John Smith, 1997-05-21, Mumbai, M, 999445566)`
- **Doctor:** `(D101, Dr. Arora, Cardiologist, 888995566)`
- **PatientDoctor:** `(P001, D101)`
- **TestLog:** `(T789, P001, 2025-09-10, Hemoglobin, 14.7)`

4. What is Data Model? Types

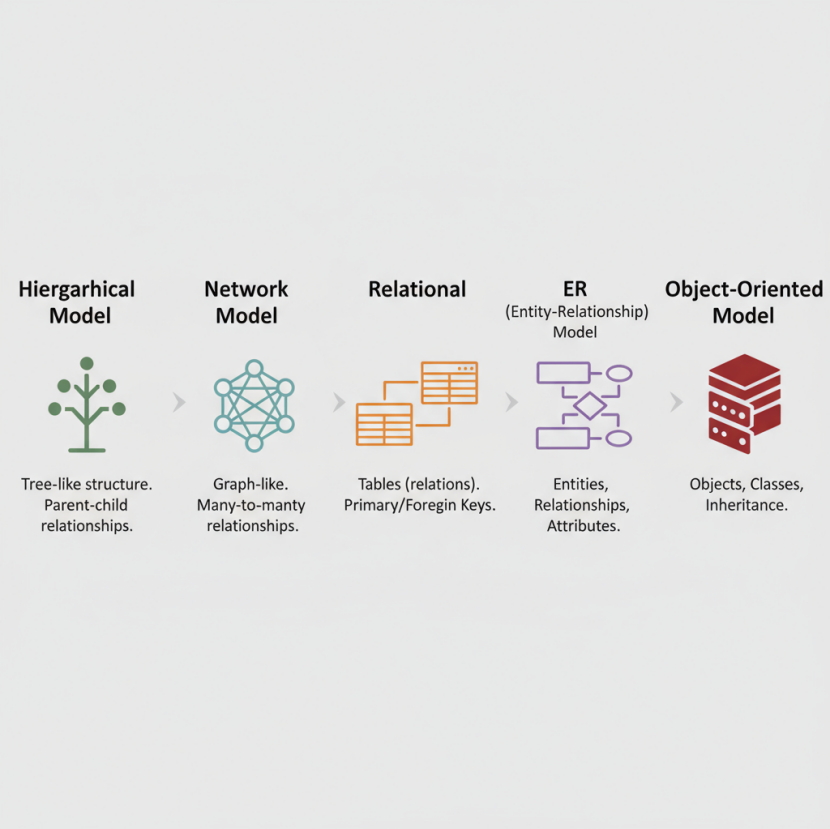
Definition:
A **data model** in DBMS is an abstract framework that defines how data is logically structured, stored, retrieved, and manipulated. It establishes rules for relationships between different data items and governs the constraints and operations that can be performed.

Importance of Data Models:

- Provide a clear blueprint for design and implementation.
- Help in communicating the data structure to all stakeholders.
- Ensure consistency, integrity, and efficiency.

Major Types of Data Models in DBMS

1. **Hierarchical Model**: Tree structure (one-to-many). *Example: File system*
2. **Network Model**: Graph structure (many-to-many). *Example: Airline routes*
3. **Relational Model**: Tables (relations); most common. *Example: Student records*
4. **Entity-Relationship (ER) Model**: Real-world entities and relationships using diagrams. *Example: Hospital database*
5. **Object-Oriented Model**: Data as objects. *Example: CAD databases*



Key Points

- Choice of data model impacts efficiency and flexibility.
- Relational and ER models are most widely adopted.

5. Specialization, Generalization, Categorization in Enhanced E-R (EER) Model

Introduction:
The **Enhanced Entity-Relationship (EER) model** expands on the basic

ER model by introducing advanced features such as **specialization**, **generalization**, and **categorization** (union types). These enable precise representation of complex real-world situations by using inheritance hierarchies.

Specialization

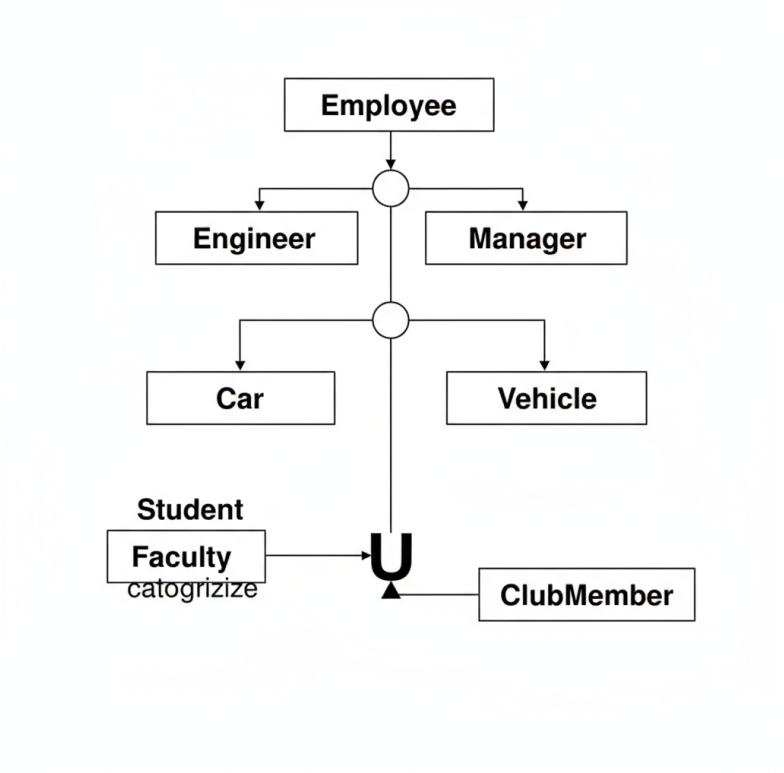
- Specialization is the process of defining subclasses from a superclass based on distinguishing attributes.
- Subclasses inherit attributes and relationships of the superclass.
- *Example: EMPLOYEE → ENGINEER, MANAGER, TECHNICIAN*

Generalization

- Generalization combines subclasses into a higher-level entity (superclass) by abstracting common features.
- *Example: CAR and TRUCK → VEHICLE*

Categorization (Union Types)

- Categorization models an entity as a subset of the union of entities from different types.
- *Example: CLUB_MEMBER may consist of both STUDENT and FACULTY*



Key Points

- Specialization divides a higher entity set into meaningful subclasses.
- Generalization merges low-level entity sets into higher-level abstractions.
- Categorization helps model union or overlapping roles using EER.