📘 Lecture 3: Database Models

Topic: How Data is Structured, Connected, and Physically Stored in Databases

---

🧩 1. Introduction to Data Models

A **data model** is a method for organizing and representing data in a structured way within a database. It defines:

- What data will be stored (entities, attributes)
- How data elements relate to each other (relationships)
- Any rules that govern the data (constraints)

Purpose: To guide database design by visually or logically representing how the data system should behave.

Think of a data model as a **blueprint** for building a house. Just as an architect needs a clear design before construction, a database designer needs a data model before implementation.

🧱 2. Levels of Data Modeling

There are three levels of data modeling:

Level	Description	Example Tool or Form
Conceptual	High-level overview using entities and relationships	ER Diagrams
Logical	Detailed version with attributes, keys, constraints	Relational Schema
Physical	Actual implementation on storage media	SQL Tables, Indexes

---

🔄 3. Conceptual vs. Logical Data Models (with Example)

Aspect	Conceptual Data Model	Logical Data Model
Focus	What data is needed, and how it's related	How the data will be structured for implementation
Audience	Business analysts, end users, DB designers	Database architects, developers
Details	Entities, relationships, high-level attributes	Tables, primary/foreign keys, data types
Independence	Independent of DBMS	Independent of physical storage, but DBMS-aware

🎓 **Example – Student Registration System**

Conceptual Model:

- **Entities**: Student, Course, Department
- **Relationships**:
    - Student *registers* for Course
    - Course *belongs to* Department

Logical Model:

- **Tables**:
    - Students (student_id, name, email)
    - Courses (course_code, title, dept_id)
    - Departments (dept_id, name)
    - Registrations (student_id, course_code, semester)
- **Keys**:
    - Primary Key: student_id in Students
    - Foreign Key: dept_id in Courses → Departments

🧠 4. Difference Between a Relation and a Table

- A **relation** is a *mathematical concept* used in the relational model. It is a set of tuples (rows) that all share the same attributes.
- A **table** is the *practical implementation* of a relation in an RDBMS.

📌 **Summary:**

Term	Definition	Practical Use
Relation	A set of tuples defined over a schema	Theory
Table	A structure in a database containing rows	Implementation

All tables in a relational database are based on the **relational model**, but not all tables are perfect relations (due to constraints, NULLs, etc.).

🌳 5. Hierarchical Data Model (Expanded)

This model arranges data in a **tree structure**:

- One parent → many children
- Each child has **only one parent**

Structure:

University

|

├── Faculty

| ├── Department

| └── Lecturer

Features:

- Fast access to parent-child data
- Uses pointers or links between records

Limitations:

- Not good for **many-to-many** relationships
- If structure changes, the whole model may break

Example in Kenya: NHIF clinic visits—where one patient can be traced under one regional center (parent), but not multiple ones.

🔗 6. Network Data Model (Expanded)

- Allows **many-to-many** relationships
- Uses **record types** and **set types** (i.e., link structures)
- More flexible than hierarchical model

Example:

In a telecom system:

- A **Customer** may subscribe to multiple **Services**
- A **Service** may belong to multiple **Customers**

Structure resembles a graph. While powerful, it is complex to maintain and query.

Historically used in systems like **Airline Reservations** or **Telco Billing Systems** in Kenya.

🧮 7. Relational Data Model (Core Model Today)

- Organizes data into **tables (relations)**
- Data is represented as rows and columns
- Tables can be linked using **keys**

Example Tables:

- Student(student_id, name)
- Course(course_code, title)
- Registration(student_id, course_code)

Advantages:

- Simple and standardized (based on relational algebra)
- Uses SQL
- Excellent support in systems like MySQL, PostgreSQL, Oracle

---

💾 8. Physical Data Model Concepts

The **physical model** focuses on how data is stored on disk. It affects performance and efficiency.

Key Concepts:

8.1 File Formats:

- **CSV**, **JSON**, **Parquet**, **Binary** formats
- Relational DBs use binary file formats with headers and indexes

8.2 Data Blocks:

- Disk storage is divided into **blocks**
- Each block holds multiple records
- Example: 4KB block contains 20 student records

8.3 Record Placement:

- **Heap file** (random placement)
- **Sequential file** (sorted by a key)
- **Hashed file** (using hash functions for lookup)

8.4 Performance Tuning:

- Use of **indexes**
- **Partitioning** of large tables
- Use of **cache and memory buffers**

Example: Locating a name in a school register that's sorted alphabetically (sequential) vs. one where names are randomly written (heap file).

🛠️ 9. Designing Data Models: Step-by-Step Process

Step 1: Requirements Collection

- Interview stakeholders (e.g., registrar, lecturer)
- Identify what data needs to be tracked

Step 2: Conceptual Design

- Create an **ER Diagram**
- Identify **entities**, **attributes**, **relationships**

Step 3: Logical Design

- Convert ERD into **tables**
- Define **keys**, **data types**, and **constraints**

Step 4: Physical Design

- Decide on file storage, indexing, and access paths
- Optimize for speed and reliability

Step 5: Implementation and Testing

- Create tables in DBMS
- Populate with sample data and test queries

📐 10. Sample Notation and Diagrams (Conceptual Overview)

- **Entity:** Rectangle
- **Attribute:** Ellipse
- **Relationship:** Diamond
- **Primary Key:** Underlined attribute
- **One-to-Many Relationship:** Line with arrow
- **Many-to-Many:** Double arrow or crow's foot

Example ERD Snippet:

[Student] —< Registers >— [Course]

- Student (student_id, name)
- Course (course_code, title)

✅ **Summary**

By the end of this session, you should be able to:

- Describe different types of data models and their purposes
- Distinguish between conceptual, logical, and physical models
- Explain the difference between a relation and a table
- Understand how hierarchical, network, and relational models work
- Know the process of designing a complete database model
- Recognize physical storage concerns like record placement and tuning

---