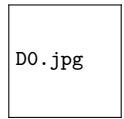
Lecture 03: Data Models, Database Architecture & DBMS Internals

Types of Databases, 3-Schema Architecture, and DBMS Components

Z2004: Database Management Systems



IIT Madras Zanzibar

Thursday, March 5, 2026

Lecture 02 Recap: The Three Anchors

File systems failed. DBMS was born. Today we move into Database Design.



Atomicity

The "All or Nothing" rule. If a transaction fails halfway, the system rolls back to prevent corruption.

Data Independence

Change physical storage (files) without rewriting application code.

DBMS vs. RDBMS

While a DBMS manages data, an RDBMS uses the Relational Model to enforce data relationships.

Master these anchors before we start the ER Diagramming lab.

Lecture Objectives

From "What kind of database?" to "How does the engine actually work?"

- **1. Types of Databases**
  Relational, NoSQL, NewSQL, Vector
- **2. Data Models**
  Relational, ER, Hierarchical, Network
- **3. Building Approaches**
  Top-down vs. Bottom-up design
- **4. 3-Schema Architecture**
  ANSI/SPARC: External, Conceptual, Internal
- **5. DBMS Components**
  The storage engine & query processor

In-Class Activity

"Design a DB for IIT Madras Zanzibar"

Next Step: Transitioning from Theory to Design

Types of Databases

Choosing the right tool depends on the **shape** and **speed** of your data.

Relational (RDBMS)

MySQL, PostgreSQL, Oracle

Best for: Structured data

Document Store

MongoDB, CouchDB

Best for: JSON/BSON blobs

Key-Value Store

Redis, DynamoDB

Best for: Fast sessions

Graph DB

Neo4j, Amazon Neptune

Best for: Social networks

Time-Series DB

InfluxDB, TimescaleDB

Best for: IoT/Sensors

Vector DB

pgvector, Pinecone

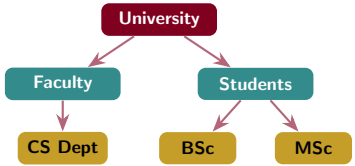
Best for: **LLMs & AI**

Crucial for L03: We focus on **Relational** first, as it remains the industry standard for data integrity.

Data Models: The Blueprint Language of Databases

A data model is the **formal language** *used to describe what data exists and how it relates. It is the bridge between the real world and the database.*

Model	Structure	Example
Relational	Tables + FK	MySQL, PostgreSQL
Entity-Rel.	Entities + Rel.	ER Diagrams (Design)
Hierarchical	Tree (Parent-Child)	XML, IBM IMS
Network	Graph (Any-to-Any)	CODASYL (1960s)
Object-Orient.	Objects + Methods	ObjectDB
Document	Nested JSON	MongoDB
Vector	High-Dim. Arrays	Pinecone, pgvector



Hierarchical Model (Tree Structure)

Key Insight for the Term

The **Relational Model** (Codd, 1970) is the foundation of this course. The **Vector Model** is where it connects to your AI work.

The Relational Model: Tables, Rows, Keys

The Vocabulary You Must Know:

- **Relation** = A Table (Set of tuples)
- **Tuple** = A Row (One record)
- **Attribute** = A Column (One field)
- **Domain** = Valid values for an attribute
- **Schema** = The structure (Columns + Types)
- **Instance** = The actual data at a point in time
- **Primary Key** = Uniquely identifies a row
- **Foreign Key** = Links two tables together

IIT Madras Zanzibar Example:

StudentID	Name	Program	CGPA
ZAN001	Amani Said	BSDSAI	8.7
ZAN002	Fatuma Ali	BSDSAI	9.1
ZAN003	Jabir Hassan	BSDSAI	7.8

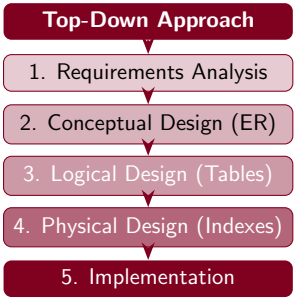
Formal Definition

Schema: Student(StudentID, Name, Program, CGPA)

Instance: 3 tuples shown above.

Domain of CGPA: $\{0.0 \ldots 10.0\}$

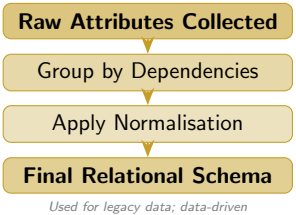
Approaches to Building a Database



Used when requirements are clear upfront

Bottom-Up Approach:

Start from raw data. Group. Normalise. Build schema.



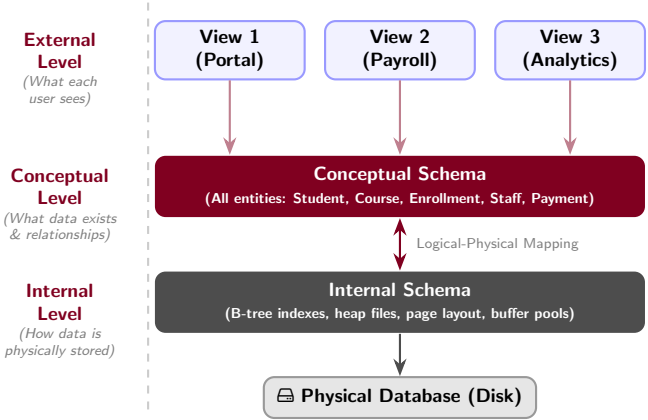
Used for legacy data; data-driven

Course Philosophy

We use Top-Down. Blueprint before building. ER Model first, SQL second.

The 3-Schema Architecture: The Core of DBMS

Proposed by ANSI/SPARC (1975). Separates **how users see data** *from* **what data exists** *from* **how data is stored.**



Data Independence: Why 3-Schema Architecture Matters

Logical Data Independence:

- Change the **conceptual schema** without touching external views.
- *Example:* Add an `Email` column to the `Student` table. The student portal app keeps working—it never asked for `Email`.

Physical Data Independence:

- Change the **internal schema** without touching the conceptual layer.
- *Example:* Switch from a heap file to a B-tree index. Queries return the same rows. Apps never know.

Physical Independence

M-Pesa upgraded its backend from Oracle to PostgreSQL in 2021.

Did every M-Pesa user have to re-learn the app or update their UI? **No.**

The internal storage changed, but the conceptual/external views did not.

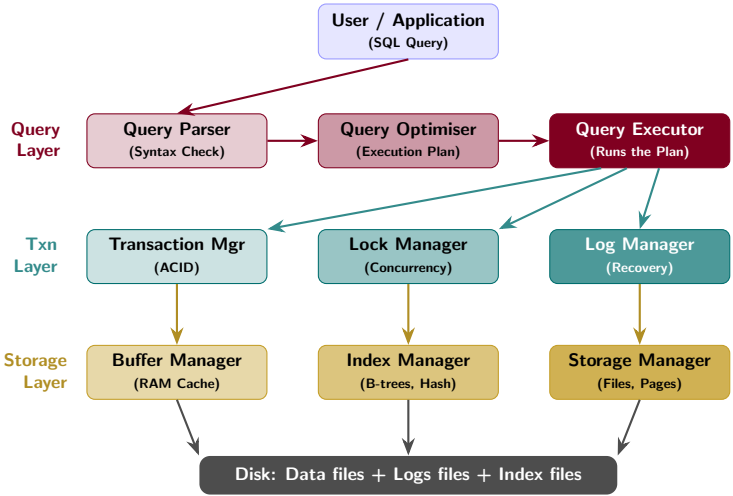
Logical Independence

WhatsApp adds a "Channels" feature (a brand new entity in the database).

Your standard Chat view and Call view remain completely unchanged.

The conceptual schema expanded, but existing external views were unaffected.

Inside the DBMS Engine: Component Map



Challenges in Building a DBMS Engine

1. Concurrency Control

1,000 users reading and writing simultaneously.

How do you prevent a dirty read or a lost update?

2. Crash Recovery

Power cut mid-transaction.

Half of an M-Pesa payment is written. Now what?

3. Query Optimisation

10 ways to answer: `SELECT * FROM A JOIN B`.

One plan takes 2ms; another takes 4 hours.

4. Data Integrity

Enforcing strict rules at the engine level:

No negative salary, no orphan foreign keys, no duplicate student IDs.

5. Security & Access Control

A student should see only their grades.

An admin sees all. A lecturer sees only their course.

6. Scalability (The Modern Era)

WhatsApp processes 100 billion messages/day.

A single RDBMS cannot hold this. Welcome to distributed databases and NoSQL.

In-Class Activity: 7 Minutes

Think. Pair. Present.

Scenario: IIT Madras Zanzibar Registration System

The university wants to build a new student registration system. It must track: **students, courses, lecturers, enrollments, grades, and payments.**

Your task (pairs):

1. Identify **3 entities** from the list above.
2. For each entity, name **2 attributes**.
3. Which **type of database** best fits this system? Why?
4. Draw a quick **3-schema** sketch: what would each level contain?

One pair will present.

Activity Solutions: IITMZ Registration System

1 & 2. Entities & Attributes

- **Student:** `StudentID`, `FullName`
- **Course:** `CourseCode`, `Credits`
- **Lecturer:** `EmpID`, `Department`
- **Payment:** `ReceiptNo`, `Amount`

3. Ideal Database Type

Choice: Relational Database (RDBMS)

Justification:

- **Structure:** University rules are strict (fixed credits, ID formats).
- **ACID Compliance:** Essential so you don't lose payment or grade records during a crash.

4. The 3-Schema Architecture

External Level (User Views):

- *Student Portal:* Sees own grades, registered courses, and fee balance.
- *Lecturer Portal:* Sees class roster and grade entry forms.

Conceptual Level (Logic):

- The "Blueprint": all tables, foreign keys, and relationships.
- Business rules (e.g., Grade must be A-F).

Internal Level (Physical):

- B-Tree indexes for fast 'StudentID' lookups.
- Data files stored physically on SSDs.

Summasry

Concept	Definition	Example
Types of DB	Relational, NoSQL, Vector, Graph, Time-series	PostgreSQL, MongoDB, Pinecone
Data Model	Formal language to describe data structure	Relational model: tables + FKs
Top-Down Approach	Requirements → ER → Schema → Physical	Our course methodology
External Schema	What each user/app sees	Student portal view
Conceptual Schema	Full logical picture of all entities	Student, Course, Enrollment
Internal Schema	Physical storage details	B-tree on StudentID column
Logical Data Indep.	Change conceptual without breaking apps	Add Email column silently
Physical Data Indep.	Change storage without changing logic	Oracle → PostgreSQL migration
Query Optimiser	Chooses the cheapest execution plan	JOIN order selection

Exit Quiz: 3 Questions

Please answer these in your notebook before you leave.

1. **Designing a messaging app:** You need to store unstructured chat messages e.g text, images, video etc. Which database type is most appropriate, and why?

   Hint: Does a multimedia message fit neatly into rigid rows and columns?

2. **Data Independence:** A developer changes the physical storage of a database from a heap file to a B-tree index to improve speed. Which type of data independence does this demonstrate?

   [Logical] *or* **[Physical]**

3. **The 3-Schema Architecture:** What exactly does the **Conceptual Schema** describe? Write one short sentence in your own words.

Exit Quiz: Answers

1. **Designing a messaging app:**
   Answer: NoSQL Database (e.g., Document or Column-family).
   Why? Chat messages with mixed media are unstructured. They require a flexible, dynamic schema rather than the rigid tables and columns of an RDBMS.

2. **Data Independence (Heap to B-tree):**
   Answer: Physical Data Independence.
   Why? Changing how the data is stored on disk (Internal Schema) to improve speed does not require changing the logical tables or the user applications.

3. **The Conceptual Schema:**
   Answer: It is the complete logical "blueprint" of the database.
   Details: It defines all entities, attributes, relationships, and business rules for the entire organization, while completely hiding the physical storage details.

Any Questions?

We covered a lot of ground today.
What needs more clarification?

Coming Up Next:

Lecture 04: Entity-Relationship Model