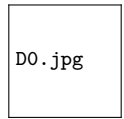
Lecture 04: Database Infrastructure, Data Processing & E-R Modeling

Languages, Architecture, Users & the Bridge to Conceptual Design

Z2004: Database Management Systems



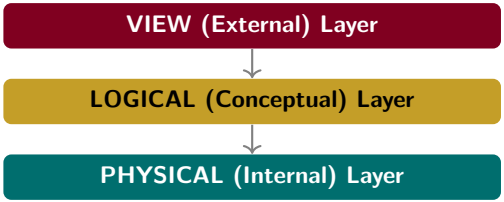
IIT Madras Zanzibar

Friday, March 6, 2026

Recap: Lecture 03 — Levels of Abstraction

WHERE WE LEFT OFF...

Three-Level Architecture



Core Principle

Data Independence:

Change one layer without breaking the applications in the layer above.

Key Takeaways

- **DBMS solves files:** Fixes redundancy, inconsistency, and poor access.
- **Physical Indep.:** Swap HDDs for SSDs; user queries run exactly the same.
- **Logical Indep.:** Add a new column; existing user views still work perfectly.
- **Data Models:** Relational, ER, Semi-structured, Object-Oriented.
- **Real-World:** M-Pesa relies on a relational core for transaction safety.

Today's Agenda & Learning Outcomes

By the end of this lecture, you will understand:

1. **Database Languages:** The critical difference between defining structures (DDL) and manipulating data (DML).
2. **The Data Dictionary:** What it is and why it serves as the "brain" or metadata catalog of the DBMS.
3. **Database Architectures:** How the system is built, comparing 2-tier (Client-Server) vs. 3-tier (Web) setups.
4. **Database Users:** The different roles interacting with the DB, highlighting the Database Administrator (DBA).
5. **E-R Modeling (The Pivot):** Transitioning from abstract theory into drawing actual Entity-Relationship blueprints.

Database Languages: DDL and DML

DDL

Data Definition Language

"Describe the structure of the data"

- Defines **schema**: tables, columns, types
- Specifies **integrity constraints** (PK, FK)
- Populates the **Data Dictionary**

```
CREATE TABLE Student (
 sid INT PRIMARY KEY,
 name VARCHAR(60) NOT NULL
);
```

DML

Data Manipulation Language

"Work with the data itself"

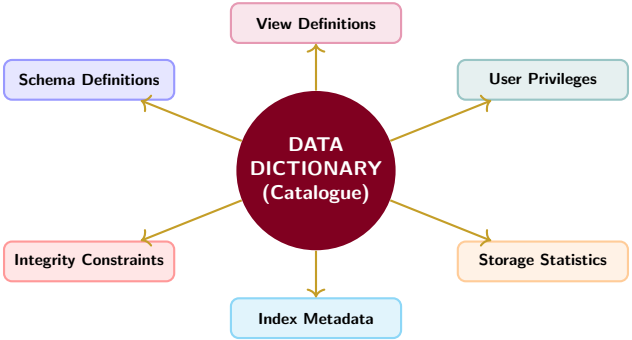
- **Retrieve, Insert, Update, Delete**
- **Procedural:** Specify *What + How*
- **Declarative (SQL):** Specify *What only*
  (You say WHAT, DBMS figures out HOW)

```
-- Declarative:  just specify WHAT
SELECT name
FROM Student
WHERE gpa > 3.5;
```

The Data Dictionary (System Catalogue)

The database *about* the database. Stores table names, constraints, and user privileges. The DBMS consults it before executing any query!

The Data Dictionary: Metadata is Everything



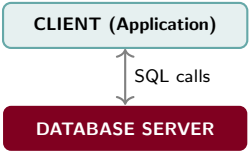
Think of the Data Dictionary as the DBMS's own **internal database**.

The Golden Rule:

It is queried by the Query Processor *before* executing **any** SQL statement to validate that the tables exist, the types match, and the user has permission.

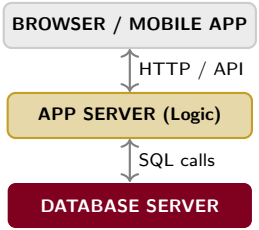
Database Architecture: 2-Tier vs. 3-Tier

Two-Tier Architecture



- App logic lives directly on the client.
- Simple, but tightly coupled.
- **Example:** SSMS connecting directly to SQL Server.

Three-Tier Architecture



- Business logic is isolated in the middle.
- Highly scalable and secure.
- Used by most modern web systems today.

Real World: WhatsApp's message history is a 3-tier system.
Your phone (client) talks to Meta's app servers, which in turn query a distributed database cluster, you never touch the DB directly.

Actors on the Scene: Database Users

Different users interact with the DBMS in completely different ways based on their technical expertise:

1. **Parametric (Naive) End Users**
   The vast majority of users. They interact strictly through graphical interfaces or mobile apps e.g., bank tellers, online shoppers. They don't know SQL exists.

2. **Application Programmers**
   Software engineers who write the applications (in Java, Python, C#) that the naive users interact with. They connect their code to the DB using APIs.

3. **Sophisticated Users**
   Users who do not write application code, but are fluent in SQL. They write complex queries to generate business reports, analytics, and insights. E.g Data Analysts, BI engineers, data scientists

But who actually manages the database engine itself?

The Database Administrator (DBA)

The "Superuser"

The DBA is the chief coordinator of the database system. They have absolute control over the data and the programs that access it.

They are the mechanic, the security guard, and the architect all rolled into one.

Core Responsibilities

- **Schema Definition:** Writing the original DDL to construct the tables and physical storage.
- **Security & Authorization:** Deciding who gets to see what. Granting access privileges to users.
- **Performance Tuning:** Monitoring the system and adding indexes to speed up slow queries.
- **Backup & Recovery:** Ensuring the data survives hardware crashes, power failures, or ransomware.
- **Routine Maintenance:** Upgrading software and monitoring disk space.

Class Activity: Think - Pair - Share

Scenario: The Hospital Upgrade

A local hospital in Zanzibar is moving from old paper records to a secure digital database. Doctors need to pull up patient history on tablets, and patients will eventually book appointments via a web portal.

- **1. THINK (1 min): Choose the Architecture:**
  Based on the requirements (mobile tablets, web portal), should the hospital build a **2-Tier** or **3-Tier** system? Jot down your reason.

Class Activity: Think - Pair - Share

Scenario: The Hospital Upgrade

A local hospital in Zanzibar is moving from old paper records to a secure digital database. Doctors need to pull up patient history on tablets, and patients will eventually book appointments via a web portal.

- **1. THINK (1 min): Choose the Architecture:**
  Based on the requirements (mobile tablets, web portal), should the hospital build a **2-Tier** or **3-Tier** system? Jot down your reason.
- **2. PAIR (2 mins): Assign the Roles:**
  Turn to your neighbor. Identify who the *Parametric Users* are in this scenario. Then, list two critical setup tasks the **DBA** must do before the system goes live.

Class Activity: Think - Pair - Share

Scenario: The Hospital Upgrade

A local hospital in Zanzibar is moving from old paper records to a secure digital database. Doctors need to pull up patient history on tablets, and patients will eventually book appointments via a web portal.

- **1. THINK (1 min): Choose the Architecture:**
  Based on the requirements (mobile tablets, web portal), should the hospital build a **2-Tier** or **3-Tier** system? Jot down your reason.
- **2. PAIR (2 mins): Assign the Roles:**
  Turn to your neighbor. Identify who the *Parametric Users* are in this scenario. Then, list two critical setup tasks the **DBA** must do before the system goes live.
- **3. SHARE: Class Discussion:**
  We will call on a few groups to present their architecture decision and their DBA task lists to the class.

Class Activity: Think - Pair - Share

Scenario: The Hospital Upgrade

A local hospital in Zanzibar is moving from old paper records to a secure digital database. Doctors need to pull up patient history on tablets, and patients will eventually book appointments via a web portal.

- **1. THINK (1 min): Choose the Architecture:**
  Based on the requirements (mobile tablets, web portal), should the hospital build a **2-Tier** or **3-Tier** system? Jot down your reason.
- **2. PAIR (2 mins): Assign the Roles:**
  Turn to your neighbor. Identify who the *Parametric Users* are in this scenario. Then, list two critical setup tasks the **DBA** must do before the system goes live.
- **3. SHARE: Class Discussion:**
  We will call on a few groups to present their architecture decision and their DBA task lists to the class.

Hint: Think about security and who writes the DDL commands!

Activity Debrief: The Hospital Upgrade

1. The Architecture: 3-Tier

- **Why?** Web portals and mobile apps need an **App Server** in the middle to handle web traffic, enforce business rules, and protect the database from direct public access.

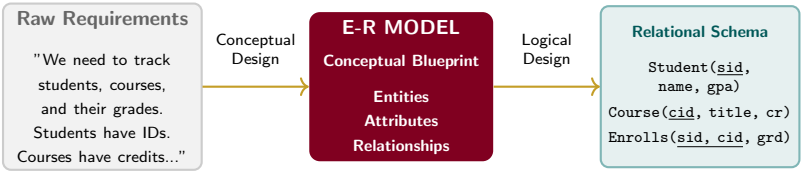
2. The Parametric Users

- **Who?** Doctors and Patients.
- **Why?** They interact purely via GUIs (tapping buttons, filling forms) and do not write or see any SQL.

3. Critical DBA Setup Tasks

- **Schema Creation (DDL):** Building the actual tables for Patients, Records, etc.
- **Security & Authorization:** Ensuring a doctor can only view their *assigned* patients' records.
- **Backup Strategy:** Setting up disaster recovery before any real data goes live.

The Blueprint Before the Build



The E-R Model is Implementation-Independent.

It captures *what the data means* and how it relates — not how it is stored.

It is the architectural blueprint drawn before a single line of SQL is written.

E-R Modeling — The Three Core Concepts

1. Entity

A **distinguishable real-world object**.



- Has a distinct identity
- Can be physical or conceptual
- **Ex:** Student, Course

Drawn as a **rectangle**

2. Attribute

A **property or descriptor** of an entity.



- **Simple**: age, GPA
- **Composite**: name (first+last)
- **Multi-valued**: {phone}
- **Derived**: age from DOB

Drawn as an **ellipse**

3. Relationship

An **association** between entities.



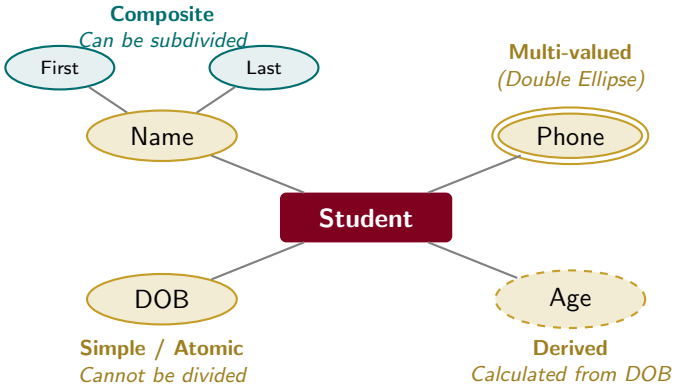
- Links two or more entities
- **Degree**: binary, ternary
- Can have its own attributes

Drawn as a **diamond**

Shape Code: **Rectangle** = Entity — **Ellipse** = Attribute — **Diamond** = Relationship — **Line** = Link

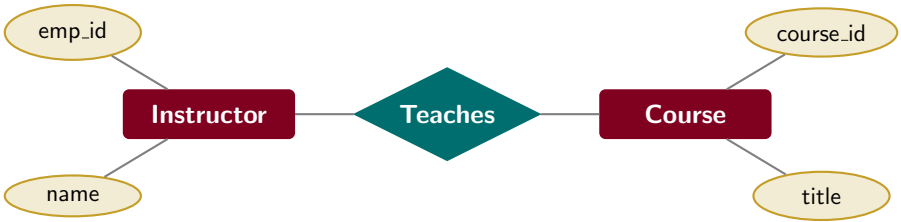
Types of Attributes

Not all attributes are created equal. E-R modeling uses specific shapes to tell the database designer exactly how to handle the data later.



Putting it Together: A Basic E-R Diagram

Let's look at a complete (but simple) blueprint connecting two entities before we dive into the rules of relationships.

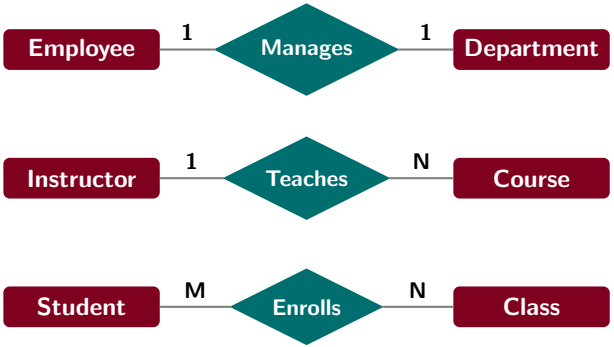


The Missing Piece:

This diagram shows *that* an instructor teaches a course... but it doesn't tell us **how many** courses they can teach, or if a course can have **multiple** instructors. We need a way to define those rules!

Relationships: Mapping the Rules (Cardinality)

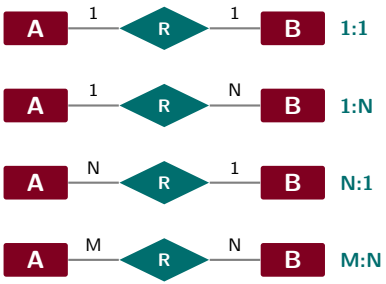
Cardinality Ratio specifies the maximum number of relationship instances that an entity can participate in. Here are the three main types:



Always read the relationship in **both directions** *to determine the correct cardinality!*

Mapping Cardinalities: Real-World Examples

The Four Types



Real-World Translations

1:1 **Person** owns one **Passport**

1:N **Dept** has many **Employees**

N:1 Many **Employees** report to one **Manager**

M:N **Students** enroll in many **Courses**

Why does Cardinality matter?

It dictates our SQL tables! For example, **M:N** relationships *always* force us to create a brand new, separate join table.

Today's Summary & Questions

The ER Modeling Blueprint



- **Meaning over Storage:** We focus on *what* the data represents before we worry about *how* SQL stores it.
- **Business Rules:** Cardinality (**1:1, 1:N, M:N**) defines how entities interact in the real world.
- **Next Step:** This diagram is the bridge between raw requirements and your final database schema.

Coming up on Monday:

We'll cover **Primary Keys** (underlining), **Weak Entities**,