The Value and Stages of Data Modelling

We established that data modelling is essential for ensuring **data integrity**, **reducing redundancy (storage)**, and **improving communication** between business and technical teams.

The process follows a mandatory three-stage sequence:

Model	Focus	Key Question	Output & Audience
Conceptual	**Business Scope**	What data is important?	High-level entities and relationships (Business Users).
Logical	**Structure & Rules**	How should data be organized (3NF)?	Detailed entities, all attributes, and primary/foreign keys (Data Architects).
Physical	**Implementation**	How do we build this in the chosen RDBMS (e.g., MySQL)?	Tables, specific data types, indexes, and SQL DDL (Developers/DBAs).

The **Entity-Relationship Diagram (ERD)** is the **visual tool** used throughout all three stages to represent entities, attributes, and relationships (like 1:M or M:N).

---

2. Normalization and Data Integrity (1NF to 3NF) 🛡️

We demonstrated the value of normalization by analysing a single, un-normalized table:

- **1NF (First Normal Form):** The starting point, requiring atomic (single) values and a Primary Key (PK). Data in this form is highly susceptible to dangers.
- **The Dangers of 1NF (Anomalies):** Data only in 1NF suffers from **Update Anomalies** (inconsistent data due to mass repetition), **Insertion Anomalies** (cannot record master data without a transactional record), and **Deletion Anomalies** (losing critical data when a single record is removed).
- **3NF (Third Normal Form):** Achieved by separating the single table into multiple, single-purpose tables to eliminate redundancy and all transitive dependencies (non-key columns depending on other non-key columns).

---

3. Key Relationships and Modelling Decisions

One-to-Many (1:M) Relationships

- This is the standard, normalized relationship (e.g., One Branch to Many Staff).
- It is enforced by placing the **Foreign Key (FK)** from the Parent table's PK into the Child table (e.g., branch_id FK in the staff table).

Many-to-Many (M:N) Relationships

- The relationship (e.g., Staff to Project) cannot be implemented directly.
- It must be resolved by introducing a **Junction Table** (Assignment), transforming the M:N into two separate 1:M relationships.

Physical Model Implementation (MySQL)

- The DDL (Data Definition Language) must be executed in the correct order: **Parent Tables** (with the PK) must be created **before** their **Child Tables** (with the FK).

- The final model used **snake_case** for all tables and attributes (e.g., staff_id, project_name) and MySQL-specific data types (INT, VARCHAR, AUTO_INCREMENT) for robustness.

---

4. Practical Design Trade-offs ⚖️

We discussed advanced concepts that balance theory with real-world needs:

- **Over-Normalization:** Normalization can go too far, creating excessive joins (e.g., separating names into their own tables), which degrades **performance** and increases **complexity** beyond the benefit of saved storage.
- **1:1 Relationships:** Can be used intentionally to **partition data for security** (e.g., separating confidential Staff_Details into a separate table) or for performance optimization, demonstrating that **security can override normalization rules**.
- **Denormalization:** The strategic *violation* of normalization rules (e.g., adding a branch_name column into the staff table) to improve read **performance** for high-traffic applications, trading integrity risk for speed.