Learning Material: ER Diagrams and Normalization

Introduction to Relational Database Design

Relational database design ensures that data is organized efficiently, consistently, and logically. Two essential concepts for designing strong databases are:

- **Entity-Relationship (ER) Diagrams** – for modeling data and relationships.
- **Normalization** – for organizing data to avoid redundancy and anomalies.

This learning material provides a concise understanding of both.

ER Diagrams (Entity-Relationship Diagrams)

What Are ER Diagrams?

An ER Diagram is a visual representation of the data structure for a system. It identifies:

- **Entities** (objects or concepts)
- **Attributes** (properties of entities)
- **Relationships** (connections between entities)

ER Diagrams help designers understand what data is needed and how different data relates.

Core Components of ER Diagrams

1. Entities

Represented as rectangles.

- Examples: *Student, Course, Teacher*

2. Attributes

Represented as ovals and connected to entities.

- Types:
  - **Key Attribute** – uniquely identifies an entity
  - **Composite Attribute** – can be broken into smaller parts
  - **Multivalued Attribute** – can have multiple values

3. Relationships

Represented as diamonds.

- Relationship types:
  - **One-to-One (1:1)**
  - **One-to-Many (1:M)**
  - **Many-to-Many (M:N)**

4. Cardinality

Shows how many instances of one entity relate to another.
Examples:

- One student *enrolls in* many courses
- Each course *is taught by* one teacher

Example Scenario

For a school database:

- **Entities:** Student, Course, Enrollment
- **Relationships:**
  - Student *enrolls in* Course (M:N)
  - Enrollment becomes a separate entity with attributes like *date enrolled* and *grade*

This structure later translates to relational tables.

Normalization

Normalization is the process of organizing data in a database to reduce redundancy and avoid update, insertion, and deletion anomalies.

Why Normalize?

- Prevents duplicated data
- Ensures consistent data
- Simplifies updates
- Improves storage efficiency

Normal Forms

Normalization proceeds through several stages called **normal forms (NF)**.

First Normal Form (1NF)

Rules:

- No repeating groups or multivalued attributes
- All values must be atomic (indivisible)

Example:

Not 1NF:

Student	Courses
Ana	Math, Sci

1NF version:

Student	Course
Ana	Math
Ana	Sci

Second Normal Form (2NF)

Applies only to tables with **composite primary keys**.

Rules:

- Already in 1NF

- No partial dependency (an attribute depends on only one part of a composite key)

Example:

Enrollment Table (StudentID + CourseID as key):

- *Grade* depends on both → OK
- *StudentName* depends only on StudentID → violates 2NF → move to Student table

---

Third Normal Form (3NF)

Rules:

- Already in 2NF
- No transitive dependency (non-key attribute depends on another non-key attribute)

Example:

Table with attributes:
(StudentID → DepartmentID → DepartmentName)

- DepartmentName depends on DepartmentID, not StudentID → violates 3NF

Solution: separate into **Department** table.

Combining ER Diagrams and Normalization

After creating the ER Diagram:

1. Convert entities into tables
2. Convert relationships into foreign keys or separate tables (for M:N)
3. Apply normalization rules to refine tables

Example Flow:

- ERD shows Student–Course M:N → create **Enrollment** table
- Normalize tables → make sure no redundant attributes remain

This ensures a logically sound and efficient relational database.