09 Database Design

- **Entity-Relationship (ER) Model**
- **Designing ER Diagrams**
- **Converting ER Diagrams to Tables**
- **Normalization Process**
- **Case Study: Designing a Small Database**

01 Entity-Relationship (ER) Model

- The Entity-Relationship (ER) Model is a conceptual framework for designing databases
- It represents the data and the relationships between data

Entities

- An entity is an object or concept that can have data stored about it
- In a university database, `Student` and `Course` could be entities

Attributes

- Attributes are the properties or details of an entity
- `Student` entity may have attributes like `StudentID`, `Name`, `DateOfBirth`
- `Course` entity may have attributes like `CourseID`, `CourseName`, `Credits`

Relationships

- Relationships describe how entities interact with each other
- A `Student` enrolls in a `Course`. This relationship could be labeled as `Enrolls`

02 Designing ER Diagrams

Components of an ER Diagram

- **Entities:** Represented as rectangles
- **Attributes:** Represented as ovals connected to their entity
- **Relationships:** Represented as diamonds connected to entities

Example

ER Diagram for University Database:

- **Entities:** `Student` and `Course`.
- **Attributes:**
  - `Student`: `StudentID`, `Name`, `DateOfBirth`
  - `Course`: `CourseID`, `CourseName`, `Credits`
- **Relationship:** `Enrolls` between `Student` and `Course`.

03 Converting ER Diagrams to Tables

Steps for Conversion

1. **Create a Table for Each Entity**
   - Each entity in the ER diagram becomes a table
2. **Define Primary Keys**
   - Each table must have a primary key that uniquely identifies each record
3. **Define Foreign Keys**
   - Relationships are implemented using foreign keys

Example

1. **Student Table:**

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(100),
    DateOfBirth DATE
);
```

2. **Course Table:**

```
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    Credits INT
);
```

3. **Enrolls Table (Relationship):**

```
CREATE TABLE Enrolls (
    StudentID INT,
    CourseID INT,
```

```
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
```

04 Normalization Process

- Normalization is the process of organizing data to minimize redundancy and improve data integrity

Normalization Forms

1. **First Normal Form (1NF)**
   - Ensure each column contains atomic values and each record is unique
   - **Example:** In a table with columns for multiple phone numbers, split it into separate rows
2. **Second Normal Form (2NF)**
   - Achieve 1NF and ensure all non-key attributes are fully functionally dependent on the primary key
   - **Example:** In a table with `StudentID` and `CourseName`, ensure that attributes related to `Course` are moved to a separate `Course` table
3. **Third Normal Form (3NF)**
   - Achieve 2NF and ensure no transitive dependencies (i.e., non-key attributes depend on other non-key attributes)
   - **Example:** If `Student` table contains `AdvisorName` and `AdvisorOffice`, and `AdvisorName` determines `AdvisorOffice`, then `Advisor` should be a separate table

Example of Normalization

StudentID	Name	CourseName	Instructor
1	John	Math	Dr. Smith
2	Jane	Science	Dr. Jones

- **1NF:** Ensure each column is atomic (already satisfied here).
- **2NF:** Separate tables for courses and students:
  - **Students Table:**

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
```

```
    Name VARCHAR(100)
);
```

- **Courses Table:**

```
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    Instructor VARCHAR(100)
);
```

- **3NF:** Ensure no transitive dependencies:
  - **Separate Table for Instructors:**

```
CREATE TABLE Instructor (
    InstructorID INT PRIMARY KEY,
    InstructorName VARCHAR(100)
);
```

  - **Updated Courses Table:**

```
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    InstructorID INT,
    FOREIGN KEY (InstructorID) REFERENCES
Instructor(InstructorID)
);
```

05 Case Study: Designing a Small Database

Scenario: A Library Database

1. **Identify Entities, Attributes & Design ER Diagram**
   - **Book:** BookID (Primary Key), Title, Author
   - **Member:** MemberID (Primary Key), Name, MembershipDate
   - **Loan:** BookID (Foreign Key), MemberID (Foreign Key), LoanDate
     - Relationship between `Book` and `Member`
2. **ER Diagram**

BookID	Title	Author

BookID	MemberID	LoanDate

MemberID	Name	MembershipDate

3. **Convert ER Diagram to Tables**

- **Book Table:**

```
CREATE TABLE Book (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100)
);
```

- **Member Table:**

```
CREATE TABLE Member (
    MemberID INT PRIMARY KEY,
    Name VARCHAR(100),
    MembershipDate DATE
);
```

- **Loan Table:**

```
CREATE TABLE Loan (
    BookID INT,
    MemberID INT,
    LoanDate DATE,
    PRIMARY KEY (BookID, MemberID),
    FOREIGN KEY (BookID) REFERENCES Book(BookID),
    FOREIGN KEY (MemberID) REFERENCES Member(MemberID)
);
```