Database Lab 03: Normalization and ERD

DengFeng Qiao Yi He MingYi Wang

1 Basic Tasks

1. **Define Key Terms**
   - **Candidate Key**: A set of attributes that can uniquely identify a row in a table.
   - **Composite Key**: A key that consists of two or more attributes to uniquely identify a row.
   - **Foreign Key**: An attribute in one table that is a primary key in another, establishing a link between the tables.
   - **Functional Dependency**: A relationship between two attributes where the value of one attribute determines the value of the other.
2. **Integrity Constraints**
   - **Entity Integrity**: Ensures no primary key can have a NULL value.
   - **Referential Integrity**: Enforces valid foreign key values that reference existing primary keys.
   - **Domain Integrity**: Limits values for attributes to specific data types or formats.
3. **Relational Integrity Violations** For the given relational tables, identify any violations of primary key, foreign key, or domain constraints. This ensures all integrity constraints are upheld.

2 Medium Tasks

4. **Project-Employee Table Anomalies**
   - **INSERT Anomaly** - Insert a new project with incomplete data:

```
INSERT INTO ProjectEmployee
(ProjectCode, ProjectTitle, ProjectManag
VALUES ('PrC30', 'Skills Matrix', 'M. Uh
```

   - **DELETE Anomaly** - Deleting a project that may lead to loss of related data:

```
DELETE FROM ProjectEmployee WHERE Project
```

- **UPDATE Anomaly** - Updating a single field impacting multiple records:

```
UPDATE ProjectEmployee
SET DepartmentNo = 'L009'
WHERE EmployeeName = 'J␣Kirk';
```

5. **Normalization Steps for Project-Employee Table**

   - Derive **1NF**: Ensure that all columns contain atomic values, eliminating repeating groups.
   - Derive **2NF**: Remove partial dependencies by ensuring each non-key attribute depends on the whole primary key.
   - Derive **3NF**: Remove transitive dependencies by ensuring non-key attributes are not dependent on other non-key attributes.

6. **Bakery Orders Normalization**

   - **1NF Transformation** - Ensure atomic values by splitting repeated fields:

```
CREATE TABLE Orders (
 OrderID INT PRIMARY KEY,
 CustomerID INT,
 ProductID INT,
 Quantity INT,
 Price DECIMAL
 );
```

   - **2NF Transformation** - Remove partial dependencies by creating separate tables:

```
CREATE TABLE Products (
 ProductID INT PRIMARY KEY,
 ProductName VARCHAR(50)
 );

CREATE TABLE OrderDetails (
 OrderID INT,
 ProductID INT,
 Quantity INT,
 Price DECIMAL,
PRIMARY KEY (OrderID, ProductID)
 );
```

   - **3NF Transformation** - Remove transitive dependencies:

```sql
CREATE TABLE Customers (
CustomerID INT PRIMARY KEY,
CustomerName VARCHAR(50),
ContactInfo VARCHAR(100)
);
```

3 Advanced Tasks

7. **Functional Dependencies and Normalization for Given Tables**

   - Identify functional dependencies and derive the tables in 1NF, 2NF, and 3NF.
   - Example:

```sql
-- 1NF: Separate table with atomic value
CREATE TABLE EmployeeProject (
EmployeeID INT,
ProjectID INT,
HoursWorked INT,
PRIMARY KEY (EmployeeID, ProjectID)
);

-- 2NF: Eliminate partial dependencies
CREATE TABLE Projects (
ProjectID INT PRIMARY KEY,
ProjectName VARCHAR(50),
ProjectManagerID INT
);

-- 3NF: Remove transitive dependencies
CREATE TABLE Managers (
ManagerID INT PRIMARY KEY,
ManagerName VARCHAR(50),
DepartmentID INT
);
```

8. **Car Rental Report Normalization**

   - Identify functional dependencies and apply normalization to 1NF, 2NF, and 3NF.

```sql
-- 1NF: Ensure atomic values
CREATE TABLE Rentals (
RentalID INT PRIMARY KEY,
BranchID INT,
CarPlate VARCHAR(10),
RentalDate DATE,
BillAmount DECIMAL
);

-- 2NF: Remove partial dependencies
CREATE TABLE Branches (
BranchID INT PRIMARY KEY,
BranchName VARCHAR(50),
SupervisorID INT
);

-- 3NF: Remove transitive dependencies
CREATE TABLE Supervisors (
SupervisorID INT PRIMARY KEY,
SupervisorName VARCHAR(50)
);
```

9. **M70 Marine Service Company Database Design**

    - Logical and physical database model with test data.

```
-- Service Tasks Table
CREATE TABLE Services (
ServiceID INT PRIMARY KEY,
BoatID INT,
ServiceDate DATE,
Task VARCHAR(50)
);

-- Engineers and Man-Hours Table
CREATE TABLE EngineerHours (
ServiceID INT,
EngineerID INT,
HoursWorked INT,
PRIMARY KEY (ServiceID, EngineerID)
);

-- Boats Table
CREATE TABLE Boats (
BoatID INT PRIMARY KEY,
BoatType VARCHAR(50),
OwnerID INT
);
```

    - Populate tables with sample data:

```
INSERT INTO Services (ServiceID, BoatID,
VALUES (1, 101, '2024-01-15', 'Inspectio

INSERT INTO EngineerHours (ServiceID, En
VALUES (1, 201, 4);

INSERT INTO Boats (BoatID, BoatType, Owr
VALUES (101, 'Yacht', 301);
```