Class Note: Database Normalization with Student Course Registration Example

Introduction to Normalization

Normalization organizes a database to eliminate redundancy, ensure data integrity, and remove anomalies (issues with inserting, updating, or deleting data). We'll use a "Student Course Registration" example to demonstrate normalization through **First Normal Form (1NF)**, **Second Normal Form (2NF)**, and **Third Normal Form (3NF)**.

We start with an unnormalized table full of anomalies, then normalize it step-by-step. To aid understanding, we include both SQL schemas and visual tables showing the data structure and sample data at each stage.

Unnormalized Table: StudentCourseRegistration

The initial table, `StudentCourseRegistration`, combines student, course, and instructor data in a single table, leading to anomalies:

- **Non-atomic values**: Columns like `Courses` store multiple values (e.g., "CS101,CS102").
- **Redundancy**: Course and instructor details repeat across rows.
- **Dependencies**: Partial and transitive dependencies cause inconsistencies.

Unnormalized Table Schema (SQL)

```sql
CREATE TABLE StudentCourseRegistration (
    StudentID VARCHAR(10),
    StudentName VARCHAR(50),
    StudentContact VARCHAR(100), -- Non-atomic: phone,email
    Courses VARCHAR(200), -- Non-atomic: multiple course IDs
    CourseTitles VARCHAR(200), -- Non-atomic: multiple titles
    CourseCredits VARCHAR(50), -- Non-atomic: multiple credits
    Instructors VARCHAR(100), -- Non-atomic: multiple instructors
    InstructorOffice VARCHAR(100), -- Non-atomic: multiple offices
    Department VARCHAR(50), -- Tied to courses
    RegistrationDate DATE
);
```

Sample Data (SQL)

```sql
INSERT INTO StudentCourseRegistration VALUES
('S001', 'John Doe', '555-1234,john@email.com', 'CS101,CS102', 'Database,Algorithms', '3,4', 'Smith,Jones', 'Bldg1-101,Bldg2-202', 'Computer Science', '2025-01-15'),
('S002', 'Jane Smith', '555-5678,jane@email.com', 'CS101,CS103', 'Database,Networking', '3,3', 'Smith,Wilson', 'Bldg1-101,Bldg3-303', 'Computer Science', '2025-01-16'),
('S003', 'Bob Johnson', '555-9012,bob@email.com', 'CS102', 'Algorithms', '4', 'Jones', 'Bldg2-202', 'Computer Science', '2025-01-17');
```

Visual Table: StudentCourseRegistration

StudentID	StudentName	StudentContact	Courses	CourseTitles	CourseCredits	Instructors	InstructorOffice	Department	I
S001	John Doe	555-1234,john@email.com	CS101,CS102	Database,Algorithms	3,4	Smith,Jones	Bldg1-101,Bldg2-202	Computer Science	2
S002	Jane Smith	555-5678,jane@email.com	CS101,CS103	Database,Networking	3,3	Smith,Wilson	Bldg1-101,Bldg3-303	Computer Science	2
S003	Bob Johnson	555-9012,bob@email.com	CS102	Algorithms	4	Jones	Bldg2-202	Computer Science	2

Anomalies

1. **Non-atomic Values**: Columns like `Courses` ("CS101,CS102") and `StudentContact` ("555-1234,john@email.com") contain multiple values, violating 1NF.
2. **Insertion Anomaly**: Cannot add a course without a student (e.g., no way to store a new course without registration).
3. **Update Anomaly**: Changing Smith's office requires updating multiple rows, risking errors.
4. **Deletion Anomaly**: Deleting S003's registration removes CS102's data.
5. **Dependencies**:
   - **Partial Dependency**: `CourseTitles`, `CourseCredits`, `Department` depend only on `Courses`.
   - **Transitive Dependency**: `InstructorOffice` depends on `Instructors`, which depends on `Courses`.

First Normal Form (1NF)

1NF Criteria

- **Atomic Values**: Each column must contain single, indivisible values.
- **Key Attributes**: Define primary keys for each table.
- **Related Data Groups**: Split data into separate tables for distinct entities (e.g., Students, Courses).

Steps to Achieve 1NF

1. **Split Non-Atomic Columns**: Break `StudentContact` into `StudentPhone` and `StudentEmail`; split `Courses`, `CourseTitles`, etc., into separate rows.
2. **Create Tables**: Define `Students`, `Courses`, `Instructors`, and `Registrations` tables.
3. **Define Keys**: Use `StudentID` (Students), `CourseID` (Courses), `InstructorName` (Instructors), and composite key (`StudentID`, `CourseID`) (Registrations).
4. **Link Tables**: Use foreign keys to maintain relationships.

1NF Tables (SQL)

```sql
-- Students table: Atomic student details
CREATE TABLE Students (
    StudentID VARCHAR(10) PRIMARY KEY,
    StudentName VARCHAR(50),
    StudentPhone VARCHAR(15),
    StudentEmail VARCHAR(50)
);

-- Courses table: Atomic course details
CREATE TABLE Courses (
    CourseID VARCHAR(10) PRIMARY KEY,
    CourseTitle VARCHAR(50),
    CourseCredits INT,
    Department VARCHAR(50)
);

-- Instructors table: Atomic instructor details
CREATE TABLE Instructors (
    InstructorName VARCHAR(50) PRIMARY KEY,
    InstructorOffice VARCHAR(50)
);

-- Registrations table: Links students to courses and instructors
CREATE TABLE Registrations (
    StudentID VARCHAR(10),
    CourseID VARCHAR(10),
    InstructorName VARCHAR(50),
    RegistrationDate DATE,
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
    FOREIGN KEY (InstructorName) REFERENCES Instructors(InstructorName)
);

-- Sample data
INSERT INTO Students VALUES
('S001', 'John Doe', '555-1234', 'john@email.com'),
('S002', 'Jane Smith', '555-5678', 'jane@email.com'),
('S003', 'Bob Johnson', '555-9012', 'bob@email.com');

INSERT INTO Courses VALUES
('CS101', 'Database', 3, 'Computer Science'),
('CS102', 'Algorithms', 4, 'Computer Science'),
('CS103', 'Networking', 3, 'Computer Science');

INSERT INTO Instructors VALUES
('Smith', 'Bldg1-101'),
('Jones', 'Bldg2-202'),
('Wilson', 'Bldg3-303');

INSERT INTO Registrations VALUES
('S001', 'CS101', 'Smith', '2025-01-15'),
('S001', 'CS102', 'Jones', '2025-01-15'),
('S002', 'CS101', 'Smith', '2025-01-16'),
('S002', 'CS103', 'Wilson', '2025-01-16'),
('S003', 'CS102', 'Jones', '2025-01-17');
```

Visual Tables (1NF)

Students

StudentID	StudentName	StudentPhone	StudentEmail
S001	John Doe	555-1234	john@email.com

StudentID	StudentName	StudentPhone	StudentEmail
S002	Jane Smith	555-5678	jane@email.com
S003	Bob Johnson	555-9012	bob@email.com

Courses

CourseID	CourseTitle	CourseCredits	Department
CS101	Database	3	Computer Science
CS102	Algorithms	4	Computer Science
CS103	Networking	3	Computer Science

Instructors

InstructorName	InstructorOffice
Smith	Bldg1-101
Jones	Bldg2-202
Wilson	Bldg3-303

Registrations

StudentID	CourseID	InstructorName	RegistrationDate
S001	CS101	Smith	2025-01-15
S001	CS102	Jones	2025-01-15
S002	CS101	Smith	2025-01-16
S002	CS103	Wilson	2025-01-16
S003	CS102	Jones	2025-01-17

1NF Achievements

- **Atomic Values**: `StudentContact` split into `StudentPhone` and `StudentEmail`; `Courses`, etc., are in separate rows.
- **Primary Keys**: Defined for each table (`StudentID`, `CourseID`, `InstructorName`, and composite `StudentID`, `CourseID`).
- **Separate Tables**: Data grouped into logical entities.
- **Foreign Keys**: Relationships enforced.
- **Remaining Issues**:
  - Partial dependencies in `Courses` (e.g., `CourseTitle`, `Department` depend on `CourseID`).
  - Transitive dependency in `Courses` (e.g., `Department` may depend on `CourseTitle`).

Second Normal Form (2NF)

2NF Criteria

- **Meets 1NF**: Atomic values and defined keys.
- **No Partial Dependencies**: Non-key attributes must depend on the entire primary key.
- **Foreign Key Relationships**: Tables linked appropriately.

Analysis

- **Students**, **Instructors**: Single-column primary keys, so no partial dependencies.
- **Courses**: `CourseID` is the primary key; all attributes depend on it.
- **Registrations**: Composite key (`StudentID`, `CourseID`). Attributes (`InstructorName`, `RegistrationDate`) depend on the full key.

To illustrate a partial dependency, imagine adding `CourseTitle` to `Registrations`. It would depend only on `CourseID`, violating 2NF. Since our 1NF `Registrations` table is already correct, we confirm it has no partial dependencies.

2NF Tables (SQL)

The 1NF tables are already in 2NF. For clarity, we restate the schema:

```
-- Students table (unchanged)
CREATE TABLE Students (
    StudentID VARCHAR(10) PRIMARY KEY,
    StudentName VARCHAR(50),
    StudentPhone VARCHAR(15),
    StudentEmail VARCHAR(50)
```

```
);

-- Courses table (unchanged)
CREATE TABLE Courses (
    CourseID VARCHAR(10) PRIMARY KEY,
    CourseTitle VARCHAR(50),
    CourseCredits INT,
    Department VARCHAR(50)
);

-- Instructors table (unchanged)
CREATE TABLE Instructors (
    InstructorName VARCHAR(50) PRIMARY KEY,
    InstructorOffice VARCHAR(50)
);

-- Registrations table: No partial dependencies
CREATE TABLE Registrations (
    StudentID VARCHAR(10),
    CourseID VARCHAR(10),
    InstructorName VARCHAR(50),
    RegistrationDate DATE,
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
    FOREIGN KEY (InstructorName) REFERENCES Instructors(InstructorName)
);

-- Sample data (same as 1NF)
INSERT INTO Students VALUES
('S001', 'John Doe', '555-1234', 'john@email.com'),
('S002', 'Jane Smith', '555-5678', 'jane@email.com'),
('S003', 'Bob Johnson', '555-9012', 'bob@email.com');

INSERT INTO Courses VALUES
('CS101', 'Database', 3, 'Computer Science'),
('CS102', 'Algorithms', 4, 'Computer Science'),
('CS103', 'Networking', 3, 'Computer Science');

INSERT INTO Instructors VALUES
('Smith', 'Bldg1-101'),
('Jones', 'Bldg2-202'),
('Wilson', 'Bldg3-303');

INSERT INTO Registrations VALUES
('S001', 'CS101', 'Smith', '2025-01-15'),
('S001', 'CS102', 'Jones', '2025-01-15'),
('S002', 'CS101', 'Smith', '2025-01-16'),
('S002', 'CS103', 'Wilson', '2025-01-16'),
('S003', 'CS102', 'Jones', '2025-01-17');
```

Visual Tables (2NF)

Identical to 1NF, as no changes were needed:

Students

StudentID	StudentName	StudentPhone	StudentEmail
S001	John Doe	555-1234	john@email.com
S002	Jane Smith	555-5678	jane@email.com
S003	Bob Johnson	555-9012	bob@email.com

Courses

CourseID	CourseTitle	CourseCredits	Department
CS101	Database	3	Computer Science
CS102	Algorithms	4	Computer Science
CS103	Networking	3	Computer Science

Instructors

InstructorName	InstructorOffice

InstructorName	InstructorOffice
Smith	Bldg1-101
Jones	Bldg2-202
Wilson	Bldg3-303

Registrations

StudentID	CourseID	InstructorName	RegistrationDate
S001	CS101	Smith	2025-01-15
S001	CS102	Jones	2025-01-15
S002	CS101	Smith	2025-01-16
S002	CS103	Wilson	2025-01-16
S003	CS102	Jones	2025-01-17

2NF Achievements

- **1NF Compliance**: Atomic values and keys.
- **No Partial Dependencies**: All attributes in `Registrations` depend on the full key.
- **Foreign Keys**: Relationships maintained.
- **Remaining Issue**: Transitive dependency in `Courses` (`Department` depends on `CourseTitle`).

Third Normal Form (3NF)

3NF Criteria

- **Meets 2NF**: No partial dependencies.
- **No Transitive Dependencies**: Non-key attributes depend only on the primary key, not other non-key attributes.

Analysis

- **Students**, **Instructors**, **Registrations**: No transitive dependencies.
- **Courses**: `Department` depends on `CourseTitle` (e.g., "Database" implies "Computer Science"), creating a transitive dependency (`CourseID` → `CourseTitle` → `Department`).

Steps to Achieve 3NF

1. **Create Departments Table**: Move `Department` to a new table with `DepartmentID` as the primary key.
2. **Update Courses Table**: Replace `Department` with `DepartmentID` as a foreign key.

3NF Tables (SQL)

```sql
-- Departments table: Removes transitive dependency
CREATE TABLE Departments (
    DepartmentID VARCHAR(10) PRIMARY KEY,
    DepartmentName VARCHAR(50)
);

-- Courses table: References DepartmentID
CREATE TABLE Courses (
    CourseID VARCHAR(10) PRIMARY KEY,
    CourseTitle VARCHAR(50),
    CourseCredits INT,
    DepartmentID VARCHAR(10),
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);

-- Students table (unchanged)
CREATE TABLE Students (
    StudentID VARCHAR(10) PRIMARY KEY,
    StudentName VARCHAR(50),
    StudentPhone VARCHAR(15),
    StudentEmail VARCHAR(50)
);

-- Instructors table (unchanged)
CREATE TABLE Instructors (
    InstructorName VARCHAR(50) PRIMARY KEY,
    InstructorOffice VARCHAR(50)
);
```

```sql
-- Registrations table (unchanged)
CREATE TABLE Registrations (
    StudentID VARCHAR(10),
    CourseID VARCHAR(10),
    InstructorName VARCHAR(50),
    RegistrationDate DATE,
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
    FOREIGN KEY (InstructorName) REFERENCES Instructors(InstructorName)
);

-- Sample data
INSERT INTO Departments VALUES
('D001', 'Computer Science');

INSERT INTO Courses VALUES
('CS101', 'Database', 3, 'D001'),
('CS102', 'Algorithms', 4, 'D001'),
('CS103', 'Networking', 3, 'D001');

INSERT INTO Students VALUES
('S001', 'John Doe', '555-1234', 'john@email.com'),
('S002', 'Jane Smith', '555-5678', 'jane@email.com'),
('S003', 'Bob Johnson', '555-9012', 'bob@email.com');

INSERT INTO Instructors VALUES
('Smith', 'Bldg1-101'),
('Jones', 'Bldg2-202'),
('Wilson', 'Bldg3-303');

INSERT INTO Registrations VALUES
('S001', 'CS101', 'Smith', '2025-01-15'),
('S001', 'CS102', 'Jones', '2025-01-15'),
('S002', 'CS101', 'Smith', '2025-01-16'),
('S002', 'CS103', 'Wilson', '2025-01-16'),
('S003', 'CS102', 'Jones', '2025-01-17');
```

Visual Tables (3NF)

Departments

DepartmentID	DepartmentName
D001	Computer Science

Courses

CourseID	CourseTitle	CourseCredits	DepartmentID
CS101	Database	3	D001
CS102	Algorithms	4	D001
CS103	Networking	3	D001

Students

StudentID	StudentName	StudentPhone	StudentEmail
S001	John Doe	555-1234	john@email.com
S002	Jane Smith	555-5678	jane@email.com
S003	Bob Johnson	555-9012	bob@email.com

Instructors

InstructorName	InstructorOffice
Smith	Bldg1-101
Jones	Bldg2-202
Wilson	Bldg3-303

Registrations

StudentID	CourseID	InstructorName	RegistrationDate
S001	CS101	Smith	2025-01-15
S001	CS102	Jones	2025-01-15
S002	CS101	Smith	2025-01-16
S002	CS103	Wilson	2025-01-16
S003	CS102	Jones	2025-01-17

3NF Achievements

- **2NF Compliance**: No partial dependencies.
- **No Transitive Dependencies**: `Department` moved to `Departments`, linked via `DepartmentID`.
- **Benefits**:
  - **Reduced Redundancy**: Department names stored once.
  - **Data Integrity**: Foreign keys ensure consistency.
  - **No Anomalies**: Can add/update/delete data without issues.

Summary

The normalization process transformed the unnormalized `StudentCourseRegistration` table into five 3NF tables: `Departments`, `Courses`, `Students`, `Instructors`, and `Registrations`. The visual tables illustrate how data is reorganized at each step:

- **1NF**: Eliminated non-atomic values, defined keys, and split data into tables.
- **2NF**: Ensured no partial dependencies.
- **3NF**: Removed transitive dependencies.

This structure ensures data integrity, minimizes redundancy, and eliminates anomalies, making the database efficient and reliable.