# Normalizing a Relation to BCNF in MySQL

## 1 Introduction

Normalization organizes a database to eliminate redundancy and ensure data integrity. A relation is in Boyce-Codd Normal Form (BCNF) if, for every functional dependency $X \to Y$, $X$ is a superkey. This document demonstrates normalizing a relation step-by-step to BCNF, using a MySQL example.

## 2 Example Relation

Consider the `StudentEnrollment` relation:

```
CREATE TABLE StudentEnrollment (
    StudentID INT,
    StudentName VARCHAR(50),
    CourseID INT,
    CourseName VARCHAR(50),
    Instructor VARCHAR(50),
    Department VARCHAR(50),
    PRIMARY KEY (StudentID, CourseID)
);
```

**Functional Dependencies**:

- `StudentID` $\to$ `StudentName`
- `CourseID` $\to$ `CourseName, Instructor`
- `Instructor` $\to$ `Department`
- `{StudentID, CourseID}` $\to$ `StudentName, CourseName, Instructor, Department`

**Candidate Key**: `{StudentID, CourseID}`

## 3 Normalization Steps

### 3.1 First Normal Form (1NF)

A relation is in 1NF if all attributes are atomic and there are no multivalued attributes or nested relations. The `StudentEnrollment` table is already in 1NF, as all attributes (`StudentID`, `StudentName`, etc.) are atomic.

## 3.2 Second Normal Form (2NF)

A relation is in 2NF if it is in 1NF and has no partial dependencies (non-key attributes depending on part of a candidate key). Here:

- `StudentName` depends on `StudentID`, part of the candidate key.
- `CourseName, Instructor` depend on `CourseID`, part of the candidate key.

To achieve 2NF, decompose the relation:

- `Student(StudentID, StudentName)`
- `Course(CourseID, CourseName, Instructor, Department)`
- `Enrollment(StudentID, CourseID)`

MySQL schema:

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    StudentName VARCHAR(50)
);

CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(50),
    Instructor VARCHAR(50),
    Department VARCHAR(50)
);

CREATE TABLE Enrollment (
    StudentID INT,
    CourseID INT,
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
```

## 3.3 Third Normal Form (3NF)

A relation is in 3NF if it is in 2NF and has no transitive dependencies (non-key attributes depending on other non-key attributes). In the `Course` table, `Instructor` $\rightarrow$ `Department` is a transitive dependency, as `Department` depends on `Instructor`, a non-key attribute.

Decompose `Course`:

- `Course(CourseID, CourseName, Instructor)`
- `InstructorInfo(Instructor, Department)`

Updated MySQL schema:

```sql
CREATE TABLE InstructorInfo (
    Instructor VARCHAR(50) PRIMARY KEY,
    Department VARCHAR(50)
);

CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(50),
    Instructor VARCHAR(50),
    FOREIGN KEY (Instructor) REFERENCES InstructorInfo(Instructor
        )
);
```

The `Student` and `Enrollment` tables remain unchanged.

### 3.4 Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if, for every functional dependency $X \to Y$, $X$ is a superkey. Check each table:

- `Student`: `StudentID` $\to$ `StudentName`, `StudentID` is the key. In BCNF.
- `InstructorInfo`: `Instructor` $\to$ `Department`, `Instructor` is the key. In BCNF.
- `Course`: `CourseID` $\to$ `CourseName, Instructor`, `CourseID` is the key. In BCNF.
- `Enrollment`: `{StudentID, CourseID}` $\to$ (no other attributes), the key determines nothing else. In BCNF.

All tables are now in BCNF.

## 4 Conclusion

Normalizing the `StudentEnrollment` relation to BCNF involved progressing through 1NF, 2NF, 3NF, and BCNF, eliminating partial and transitive dependencies. The final schema consists of `Student`, `Course`, `InstructorInfo`, and `Enrollment` tables, implemented in MySQL with appropriate primary and foreign keys to maintain data integrity.