Database Design in Boyce-Codd Normal Form (BCNF)

To design a database in Boyce-Codd Normal Form (BCNF), you need to ensure that it meets certain criteria, typically achieved through a series of normalization steps. Here's a step-by-step guide to achieve BCNF:

Step 1: Understand the Requirements

Gather all the necessary information about the database's intended use, including the data to be stored, the relationships between different types of data, and the constraints.

Step 2: Identify All Attributes

List all attributes that will be included in the database. For example, let's consider a simple student-course registration system with the following attributes:

- StudentID
- StudentName
- CourseID
- CourseName
- InstructorName
- EnrollmentDate

Step 3: Identify Functional Dependencies

Identify all functional dependencies between attributes. Functional dependencies describe the relationship between attributes, where one attribute uniquely determines another. For example:

- StudentID -> StudentName
- CourseID -> CourseName, InstructorName
- {StudentID, CourseID} -> EnrollmentDate

Step 4: Create Initial Relations (1NF)

Create initial tables ensuring that each table has a primary key and that all attributes are atomic (1NF - First Normal Form). For our example, the initial tables might look like this:

1. Students(StudentID, StudentName)

2. Courses(CourseID, CourseName, InstructorName)

3. Enrollments(StudentID, CourseID, EnrollmentDate)

Step 5: Ensure 2NF (Second Normal Form)

To achieve 2NF, ensure that all non-key attributes are fully functionally dependent on the primary key. In our case, the tables are already in 2NF because there are no partial dependencies.

Step 6: Ensure 3NF (Third Normal Form)

To achieve 3NF, ensure that all attributes are only dependent on the primary key, meaning there are no transitive dependencies. Our tables are already in 3NF.

Step 7: Ensure BCNF (Boyce-Codd Normal Form)

To achieve BCNF, ensure that for every functional dependency X -> Y, X is a superkey. If a table violates BCNF, decompose it.

Let's consider another example to illustrate BCNF normalization. Suppose we have the following table with attributes and functional dependencies:

- Table: Students (StudentID, CourseID, InstructorName)

- Functional Dependencies:

  - StudentID -> InstructorName

  - CourseID -> InstructorName

This table is not in BCNF because InstructorName depends on both StudentID and CourseID, and neither StudentID nor CourseID is a superkey.

Decompose the table:

1. Original Table: Students(StudentID, CourseID, InstructorName)

2. Decomposition:

   - Table 1: Students(StudentID, CourseID)

   - Table 2: Courses(CourseID, InstructorName)

Final Tables in BCNF

After ensuring all tables are in BCNF, you should have:

1. Students(StudentID, StudentName)

2. Courses(CourseID, CourseName, InstructorName)

3. Enrollments(StudentID, CourseID, EnrollmentDate)

These steps ensure your database design is in BCNF, eliminating redundancy and maintaining data integrity.