Step-by-Step Database Normalization Guide

What is Normalization?

Database normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves decomposing tables into smaller, related tables and defining relationships between them.

Goals of Normalization

- Eliminate redundant data
- Reduce storage space
- Prevent data anomalies (insertion, update, deletion)
- Ensure data consistency
- Improve data integrity

Starting Example: Unnormalized Student Registration System

Let's work with this initial table that contains all information in one place:

Student_Course_Registration (Unnormalized)

StudentID	StudentName	StudentEmail	StudentPhone	CourseID	CourseName	Credits	InstructorID	InstructorName	InstructorEmail	Department	RoomNumber	Grade	Semester
S001	Alice Brown	alice@email.com	555-0101	CS101	Intro to CS	3	I001	Dr. Smith	smith@university.edu	Computer	R101	A	Fall2024
S001	Alice Brown	alice@email.com	555-0101	MA201	Calculus I	4	I002	Dr. Johnson	johnson@university.edu	Math	R205	B+	Fall2024
S002	Bob Green	bob@email.com	555-0102	CS101	Intro to CS	3	I001	Dr. Smith	smith@university.edu	Computer	R101	B	Fall2024
S002	Bob Green	bob@email.com	555-0102	EN101	English Comp	3	I003	Dr. Wilson	wilson@university.edu	English	R150	A-	Fall2024
S003	Carol White	carol@email.com	555-0103	MA201	Calculus I	4	I002	Dr. Johnson	johnson@university.edu	Math	R205	A	Fall2024

Problems with this table:

- Data redundancy (student info repeated for each course)

- Storage waste
- Update anomalies (changing Alice's phone requires multiple updates)
- Insertion anomalies (can't add a course without a student)
- Deletion anomalies (removing last student from a course loses course info)

First Normal Form (1NF)

Rules for 1NF:

1. Each column contains atomic (indivisible) values
2. Each column contains values of the same data type
3. Each row is unique
4. No repeating groups or arrays

Step 1: Check for 1NF Violations

Our table already satisfies 1NF because:

- ✅ All values are atomic (no comma-separated lists)
- ✅ Each column has consistent data types
- ✅ Each row is unique (combination of StudentID + CourseID)
- ✅ No repeating groups

Result: Our table is already in 1NF.

Second Normal Form (2NF)

Rules for 2NF:

1. Must be in 1NF
2. No partial dependencies (non-key attributes must depend on the entire primary key)

Step 2: Identify the Primary Key

The primary key is the combination: **(StudentID, CourseID)**

Step 3: Identify Partial Dependencies

Let's analyze each non-key attribute:

Attributes that depend only on StudentID (partial dependencies):

- StudentID → StudentName
- StudentID → StudentEmail
- StudentID → StudentPhone

Attributes that depend only on CourseID (partial dependencies):

- CourseID → CourseName
- CourseID → Credits
- CourseID → InstructorID
- CourseID → InstructorName
- CourseID → InstructorEmail
- CourseID → Department
- CourseID → RoomNumber

Attributes that depend on the full key (StudentID, CourseID):

- (StudentID, CourseID) → Grade
- (StudentID, CourseID) → Semester

Step 4: Decompose to Eliminate Partial Dependencies

Students Table:

StudentID	StudentName	StudentEmail	StudentPhone
S001	Alice Brown	alice@email.com	555-0101
S002	Bob Green	bob@email.com	555-0102
S003	Carol White	carol@email.com	555-0103

Courses Table:

CourseID	CourseName	Credits	InstructorID	InstructorName	InstructorEmail	Department	RoomNumber
CS101	Intro to CS	3	I001	Dr. Smith	smith@university.edu	Computer	R101
MA201	Calculus I	4	I002	Dr. Johnson	johnson@university.edu	Math	R205
EN101	English Comp	3	I003	Dr. Wilson	wilson@university.edu	English	R150

Enrollments Table:

StudentID	CourseID	Grade	Semester
S001	CS101	A	Fall2024
S001	MA201	B+	Fall2024
S002	CS101	B	Fall2024
S002	EN101	A-	Fall2024
S003	MA201	A	Fall2024

Result: Now in 2NF - all partial dependencies eliminated.

Third Normal Form (3NF)

Rules for 3NF:

1. Must be in 2NF
2. No transitive dependencies (non-key attributes cannot depend on other non-key attributes)

Step 5: Identify Transitive Dependencies

Looking at our Courses table:

Transitive Dependencies:

- CourseID → InstructorID → InstructorName
- CourseID → InstructorID → InstructorEmail
- CourseID → InstructorID → Department (assuming instructor determines department)

Step 6: Decompose to Eliminate Transitive Dependencies

Students Table: (No changes - already in 3NF)

StudentID	StudentName	StudentEmail	StudentPhone
S001	Alice Brown	alice@email.com	555-0101
S002	Bob Green	bob@email.com	555-0102
S003	Carol White	carol@email.com	555-0103

Instructors Table: (New table)

InstructorID	InstructorName	InstructorEmail	Department
I001	Dr. Smith	smith@university.edu	Computer
I002	Dr. Johnson	johnson@university.edu	Math
I003	Dr. Wilson	wilson@university.edu	English

Courses Table: (Modified - removed transitive dependencies)

CourseID	CourseName	Credits	InstructorID	RoomNumber
CS101	Intro to CS	3	I001	R101
MA201	Calculus I	4	I002	R205
EN101	English Comp	3	I003	R150

Enrollments Table: (No changes)

StudentID	CourseID	Grade	Semester
S001	CS101	A	Fall2024
S001	MA201	B+	Fall2024
S002	CS101	B	Fall2024
S002	EN101	A-	Fall2024
S003	MA201	A	Fall2024

Result: Now in 3NF - all transitive dependencies eliminated.

Boyce-Codd Normal Form (BCNF)

Rules for BCNF:

1. Must be in 3NF
2. For every functional dependency A → B, A must be a superkey

Step 7: Check for BCNF Violations

Looking at our tables, let's check if all determinants are superkeys:

Students Table:

- StudentID → StudentName, StudentEmail, StudentPhone
- StudentID is the primary key (superkey) ✅

Instructors Table:

- InstructorID → InstructorName, InstructorEmail, Department
- InstructorID is the primary key (superkey) ✅

Courses Table:

- CourseID → CourseName, Credits, InstructorID, RoomNumber
- CourseID is the primary key (superkey) ✅

Enrollments Table:

- (StudentID, CourseID) → Grade, Semester
- (StudentID, CourseID) is the primary key (superkey) ✅

Result: All tables are already in BCNF.

Fourth Normal Form (4NF)

Rules for 4NF:

1. Must be in BCNF
2. No multivalued dependencies

Step 8: Check for Multivalued Dependencies

Let's say we want to track student skills and student hobbies. A problematic design would be:

Student_Skills_Hobbies (Violates 4NF):

StudentID	Skill	Hobby
S001	Java	Reading
S001	Java	Swimming
S001	Python	Reading
S001	Python	Swimming
S002	JavaScript	Gaming
S002	React	Gaming

Multivalued Dependencies:

- StudentID →→ Skill (independent of Hobby)
- StudentID →→ Hobby (independent of Skill)

Step 9: Decompose to Eliminate Multivalued Dependencies

Student_Skills Table:

StudentID	Skill
S001	Java
S001	Python
S002	JavaScript
S002	React

Student_Hobbies Table:

StudentID	Hobby
S001	Reading
S001	Swimming
S002	Gaming

Result: Now in 4NF - multivalued dependencies eliminated.

Fifth Normal Form (5NF)

Rules for 5NF:

1. Must be in 4NF
2. No join dependencies that are not implied by candidate keys

Step 10: Check for Join Dependencies

Consider a scenario with Suppliers, Parts, and Projects:

Supplier_Part_Project (Potential 5NF violation):

Supplier	Part	Project
S1	P1	J1
S1	P2	J1
S2	P1	J2

If this can be losslessly decomposed into three binary relations and reconstructed, it might violate 5NF.

However, for most practical academic purposes, achieving 3NF or BCNF is sufficient.

Final Normalized Schema Summary

Our Final 3NF/BCNF Schema:

1. **Students**
   - Primary Key: StudentID
   - Attributes: StudentName, StudentEmail, StudentPhone
2. **Instructors**
   - Primary Key: InstructorID
   - Attributes: InstructorName, InstructorEmail, Department
3. **Courses**
   - Primary Key: CourseID
   - Foreign Key: InstructorID → Instructors(InstructorID)
   - Attributes: CourseName, Credits, RoomNumber
4. **Enrollments**
   - Primary Key: (StudentID, CourseID)
   - Foreign Keys:
     - StudentID → Students(StudentID)
     - CourseID → Courses(CourseID)
   - Attributes: Grade, Semester

Benefits Achieved

Before Normalization:

- 5 rows × 14 columns = 70 data points
- Massive redundancy
- Multiple anomalies

After Normalization:

- Students: 3 rows × 4 columns = 12 data points
- Instructors: 3 rows × 4 columns = 12 data points
- Courses: 3 rows × 5 columns = 15 data points

- Enrollments: 5 rows × 4 columns = 20 data points
- **Total: 59 data points (15% reduction)**

Anomalies Eliminated:

- ✅ **Update Anomaly:** Changing a student's phone number requires only one update
- ✅ **Insert Anomaly:** Can add new courses without requiring student enrollment
- ✅ **Delete Anomaly:** Removing student enrollment doesn't lose course information

---

Practice Exercise for Students

Given this unnormalized table, normalize it step by step:

Library_System (Unnormalized)

BookID	Title	AuthorName	AuthorEmail	PublisherName	PublisherAddress	MemberID	MemberName	MemberPhone	BorrowDate	ReturnDate
B001	Database 101	John Smith	js@email.com	TechBooks	123 Main St	M001	Alice	555-1111	2024-01-15	2024-02-15
B002	Java Guide	Jane Doe	jd@email.com	CodePress	456 Oak Ave	M001	Alice	555-1111	2024-01-20	2024-02-20

Challenge: Normalize this table through 3NF, identifying all dependencies and creating the appropriate tables with proper relationships.