Database design

: relies heavily on functional dependencies and normalization to minimize redundancy and enhance data integrity. These principles systematically guide the organization of data into tables, ensuring dependencies are logically structured and enforced. Each normal form (1NF, 2NF, 3NF, BCNF, 4NF, 5NF) imposes increasingly strict rules to eliminate specific types of data anomalies. Functional dependencies

A functional dependency (FD) exists when the value of a set of attributes uniquely determines the value of another set of attributes .

* **Determinant:** The attribute set on the left-hand side of the arrow is called the determinant.
* **Dependent:** The attribute set on the right-hand side is the dependent.

**Example of functional dependency**

Consider a table with a StudentID, StudentName, and StudentAge.

* A functional dependency StudentID StudentName exists because for every StudentID, there is only one associated StudentName.
* The reverse, StudentName StudentID, is not a valid FD because multiple students could have the same name.

**Normalization forms**

**First Normal Form (1NF)**

A table is in 1NF if it meets the following conditions:

* + **Atomic values:** Each column must contain atomic (single, indivisible) values. It should not contain repeating groups of values or arrays within a single row.
  + **Unique rows:** Each row must be unique, and a primary key should uniquely identify each record.
  + **Example (Before 1NF):** A table storing StudentID, StudentName, and StudentSubjects might list "Physics, Chemistry" in one cell for a student.
  + **Example (After 1NF):** The table is decomposed so each subject has its own row, creating a composite primary key of StudentID and Subject.

**Second Normal Form (2NF)**

A table is in 2NF if it is already in 1NF and there are no partial dependencies.

* + **Partial dependency:** This occurs when a non-prime (non-key) attribute is functionally dependent on only part of the primary key.
  + **Example (Before 2NF):** Consider a table (StudentID, CourseID, Instructor, CourseName). The composite primary key is (StudentID, CourseID).
    - Instructor depends on CourseID.
    - CourseName depends on CourseID.
    - This is a partial dependency because CourseName and Instructor do not depend on the entire primary key (StudentID, CourseID).
  + **Example (After 2NF):** Split the table into (StudentID, CourseID) and (CourseID, CourseName, Instructor). This removes the partial dependency, ensuring that all non-key attributes depend on the *full* primary key of their respective tables.

**Third Normal Form (3NF)**

A table is in 3NF if it is in 2NF and there are no transitive dependencies.

* + **Transitive dependency:** This exists when a non-key attribute is dependent on another non-key attribute.
  + **Example (Before 3NF):** A table (BookID, BookGenreID, BookGenreName) has the primary key BookID.
  + BookID BookGenreID.
  + BookGenreID BookGenreName.
  + This creates a transitive dependency BookID BookGenreName via BookGenreID.

**Example (After 3NF):** Decompose the table into (BookID, BookGenreID) and (BookGenreID, BookGenreName). Now, there is no transitive dependency

**Boyce-Codd Normal Form (BCNF)**

BCNF is a stricter version of 3NF that handles anomalies in tables with multiple overlapping candidate keys.

* **Condition:** For every non-trivial functional dependency , must be a superkey.
* **Edge Case for 3NF:** A 3NF table is not in BCNF if a non-key attribute determines a part of a candidate key.
* **Example (3NF but not BCNF):** A table (Student, Course, Instructor).
  + Candidate keys: (Student, Course) and (Student, Instructor).
  + Dependency: Instructor Course.
  + This violates BCNF because Instructor is not a superkey, but it is a determinant.
* **Example (After BCNF):** Decompose into (Student, Instructor) and (Instructor, Course).

Fourth Normal Form (4NF)

A table is in 4NF if it is in BCNF and has no non-trivial multivalued dependencies (MVDs).

* **Multivalued dependency ():** A dependency where an attribute determines multiple, independent values of attribute . For an MVD to exist, there must be at least three attributes, and the dependent attributes must be independent of each other.
* **Example (Before 4NF):** A (Student, Course, Hobby) table shows a student taking multiple courses and having multiple hobbies.
  + MVDs: Student Course and Student Hobby.
  + Course and Hobby are independent of each other, causing redundancy.
* **Example (After 4NF):** Decompose into (Student, Course) and (Student, Hobby). This removes the MVD by separating the independent, one-to-many relationships.

Fifth Normal Form (5NF)

A table is in 5NF if it is in 4NF and has no join dependencies.

* **Join dependency (JD):** A table has a join dependency if it can be losslessly decomposed into smaller tables that, when joined, recreate the original table without spurious (incorrect) rows.
* **Condition:** 5NF is the decomposition of a table into as many smaller tables as possible without losing information. This applies to rare cases where a table cannot be fully normalized based on functional or multivalued dependencies alone.
* **Example (Before 5NF):** A table (Subject, Lecturer, Semester) tracking which lecturers teach which subjects in certain semesters could have a join dependency.
* **Example (After 5NF):** Decomposing the table into (Subject, Semester), (Lecturer, Subject), and (Lecturer, Semester) removes the join dependency, but only if all three tables must be joined to reconstruct the original data without creating new records.

To relate all normalization concepts to a database, we can use a sample company database and demonstrate how a single, unnormalized table is progressively broken down and reorganized to meet the requirements of each normal form.

**Unnormalized initial data**

Assume a company records employee, department, and project information in a single, unnormalized table. Notice the repeating groups of values in the Projects column and the redundant data in multiple columns.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmpID** | **EmpName** | **DeptID** | **DeptName** | **Projects** | **ProjectManager** |
| E101 | Alice | D01 | Marketing | P101, P102 | David |
| E102 | Bob | D02 | Sales | P102 | David |
| E103 | Carol | D01 | Marketing | P101 | David |
| E104 | Dave | D03 | HR | P103 | Eva |

First Normal Form (1NF)

To achieve 1NF, we eliminate the repeating groups in the Projects column by ensuring each row contains only atomic values. This requires adding rows for each project an employee works on.

**Table: Employee\_Projects\_1NF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmpID** | **EmpName** | **DeptID** | **DeptName** | **ProjectID** | **ProjectManager** |
| E101 | Alice | D01 | Marketing | P101 | David |
| E101 | Alice | D01 | Marketing | P102 | David |
| E102 | Bob | D02 | Sales | P102 | David |
| E103 | Carol | D01 | Marketing | P101 | David |
| E104 | Dave | D03 | HR | P103 | Eva |

* **Functional Dependencies:**
  + {EmpID, ProjectID} EmpName, DeptID, DeptName, ProjectManager
  + EmpID EmpName, DeptID, DeptName
  + ProjectID ProjectManager
* **Problem:** Partial dependencies exist. For example, EmpName and DeptID depend only on EmpID (part of the composite key). ProjectManager depends only on ProjectID.

Second Normal Form (2NF)

To reach 2NF, we remove partial dependencies by creating separate tables for logically distinct data.

**Table: Employees\_2NF**

|  |  |  |
| --- | --- | --- |
| **EmpID** | **EmpName** | **DeptID** |
| E101 | Alice | D01 |
| E102 | Bob | D02 |
| E103 | Carol | D01 |
| E104 | Dave | D03 |

**Table: Employee\_Projects\_2NF**

|  |  |
| --- | --- |
| **EmpID** | **ProjectID** |
| E101 | P101 |
| E101 | P102 |
| E102 | P102 |
| E103 | P101 |
| E104 | P103 |

**Table: Projects\_2NF**

|  |  |
| --- | --- |
| **ProjectID** | **ProjectManager** |
| P101 | David |
| P102 | David |
| P103 | Eva |

* **Functional Dependencies:**
  + DeptID DeptName (still a transitive dependency)
* **Problem:** The DeptID in the Employees\_2NF table determines the DeptName. This is a transitive dependency, as EmpID determines DeptID, which in turn determines DeptName.

Third Normal Form (3NF)

We achieve 3NF by moving the transitive dependency into its own table.

**Table: Employees\_3NF**

|  |  |  |
| --- | --- | --- |
| **EmpID** | **EmpName** | **DeptID** |
| E101 | Alice | D01 |
| E102 | Bob | D02 |
| E103 | Carol | D01 |
| E104 | Dave | D03 |

**Table: Departments\_3NF**

|  |  |
| --- | --- |
| **DeptID** | **DeptName** |
| D01 | Marketing |
| D02 | Sales |
| D03 | HR |

**Table: Employee\_Projects\_3NF**

|  |  |
| --- | --- |
| **EmpID** | **ProjectID** |
| E101 | P101 |
| E101 | P102 |
| E102 | P102 |
| E103 | P101 |
| E104 | P103 |

**Table: Projects\_3NF**

|  |  |
| --- | --- |
| **ProjectID** | **ProjectManager** |
| P101 | David |
| P102 | David |
| P103 | Eva |

Boyce-Codd Normal Form (BCNF)

Assume for this example that the ProjectManager is also a unique identifier for a project. Now we have two candidate keys in Projects\_3NF: ProjectID and ProjectManager.

* **Functional Dependency:** ProjectManager ProjectID.
* **Problem:** This violates BCNF because ProjectManager is a determinant but not a superkey of the original table structure that would include employee information.  
  To address this, we would likely combine the employee and project information more carefully, but in our simplified Projects\_3NF table, ProjectManager already acts as a candidate key, so the table is in BCNF. BCNF issues typically arise in more complex table designs involving overlapping keys.

Fourth Normal Form (4NF)

Suppose employees can speak multiple languages and have multiple hobbies, with no relation between a language and a hobby.

**Table: Employee\_Languages\_Hobbies (violates 4NF)**

|  |  |  |
| --- | --- | --- |
| **EmpID** | **Language** | **Hobby** |
| E101 | English | Reading |
| E101 | English | Cooking |
| E101 | Spanish | Reading |
| E101 | Spanish | Cooking |

* **Multivalued Dependencies (MVDs):** EmpID Language and EmpID Hobby.
* **Problem:** The language and hobby lists are independent of each other, causing redundant rows.

**Solution for 4NF**

**Table: Employee\_Languages\_4NF**

|  |  |
| --- | --- |
| **EmpID** | **Language** |
| E101 | English |
| E101 | Spanish |

**Table: Employee\_Hobbies\_4NF**

|  |  |
| --- | --- |
| **EmpID** | **Hobby** |
| E101 | Reading |
| E101 | Cooking |

**Fifth Normal Form (5NF)**

5NF deals with join dependencies in rare cases where a table can be losslessly decomposed into more tables. This is often an academic exercise, as most real-world databases are already in 4NF. For our company example, assume there's a constraint that a supplier must supply a particular product to a project *only if* the supplier supplies that product and the project uses that supplier.

**Table: Project\_Supplier\_Product (violates 5NF)**

|  |  |  |
| --- | --- | --- |
| **Project** | **Supplier** | **Product** |
| P101 | A | X |
| P101 | B | Y |
| P102 | A | X |

This table contains a join dependency and can be broken down to eliminate redundant combinations that can be inferred from other pairings.

**Solution for 5NF**

**Table: Project\_Supplier\_5NF**

|  |  |
| --- | --- |
| **Project** | **Supplier** |
| P101 | A |
| P101 | B |
| P102 | A |

**Table: Supplier\_Product\_5NF**

|  |  |
| --- | --- |
| **Supplier** | **Product** |
| A | X |
| B | Y |

**Table: Project\_Product\_5NF**

|  |  |
| --- | --- |
| **Project** | **Product** |
| P101 | X |
| P101 | Y |
| P102 | X |

The original table can be losslessly reconstructed by joining these three smaller tables. This decomposition removes redundancy and prevents insertion anomalies related to complex, interwoven business rules.

**What is a multivalued dependency (MVD)?**

A multivalued dependency exists when there are at least three attributes in a table, and for each value of one attribute, there is a well-defined set of values for two or more other attributes that are independent of each other.

* **Representation**: A multivalued dependency is represented by a double-arrow symbol: . This is read as "A multidetermines B".
* **Condition for MVD**: For a multivalued dependency to exist, there must be at least three attributes in the table. The attributes and must be independent of each other, but both must depend on attribute .

Example of an MVD (before 4NF)

Consider a table for a car manufacturer that stores information about car models, the colors they come in, and the manufacturing years for each model.

**Table: Car\_Model\_Details**

|  |  |  |
| --- | --- | --- |
| **Car\_Model** | **Color** | **Manuf\_Year** |
| ModelX | Red | 2024 |
| ModelX | Red | 2025 |
| ModelX | Blue | 2024 |
| ModelX | Blue | 2025 |
| ModelY | Silver | 2025 |

In this table:

* The Color of a car model is independent of its Manuf\_Year.
* Both Color and Manuf\_Year are dependent on the Car\_Model.
* The data is redundant. For ModelX, we have to repeat the colors (Red, Blue) for each manufacturing year (2024, 2025), and vice versa. This creates update, insert, and delete anomalies.
* The MVDs are: Car\_Model Color and Car\_Model Manuf\_Year.

How to resolve MVDs and achieve 4NF

The solution is to decompose the table into smaller, separate tables that eliminate the redundancy caused by the independent multivalued attributes.

**Table: Car\_Colors**

|  |  |
| --- | --- |
| **Car\_Model** | **Color** |
| ModelX | Red |
| ModelX | Blue |
| ModelY | Silver |

**Table: Car\_Years**

|  |  |
| --- | --- |
| **Car\_Model** | **Manuf\_Year** |
| ModelX | 2024 |
| ModelX | 2025 |
| ModelY | 2025 |

After decomposition:

* The Car\_Colors and Car\_Years tables each represent a single multivalued dependency, removing the redundant combinations.
* The database is now in 4NF.
* This approach simplifies the data model, making it more efficient, easier to maintain, and less prone to errors.