DATABASES

GS WEB APPLICATION DEVELOPMENT

Teacher: Vanesa Maldonado Guerrero
Curs 2025/26

Introduction

In relational databases, especially large ones, you need to arrange entries so that other maintainers and administrators can read them and work on them. This is why database normalization is important.

In simple words, database normalization entails organizing a database into several tables in order to reduce redundancy. You can design the database to follow any of the types of normalization such as 1NF, 2NF, and 3NF.

What is Database Normalization?

- Database normalization is a database design principle for organizing data in an organized and consistent way.
- It helps you avoid redundancy and maintain the integrity of the database. It also helps you eliminate undesirable characteristics associated with insertion, deletion, and updating.

What is the Purpose of Normalization?

The main purpose of database normalization is to avoid complexities, eliminate duplicates, and organize data in a consistent way. In normalization, the data is divided into several tables linked together with relationships.

Database administrators are able to achieve these relationships by using primary keys, foreign keys, and composite keys.

To get it done, a primary key in one table, for example, employee_wages is related to the value from another table, for instance, employee_data.

- **N.B.: A primary key** is a column that uniquely identifies the rows of data in that table. It's a unique identifier such as an employee ID, student ID, voter's identification number (VIN), and so on.
- A **foreign key** is a field that relates to the primary key in another table.
- A **composite key** is just like a primary key, but instead of having a column, it has multiple columns.

What is 1NF 2NF and 3NF?

- **1NF, 2NF, and 3NF** are the first three types of database normalization. They stand for first normal form, second normal form, and third normal form, respectively.
- There are also **4NF (fourth normal form) and 5NF (fifth normal form)**. There's even 6NF (sixth normal form), but the commonest normal form you'll see out there is 3NF (third normal form).
- All the types of database normalization are cumulative – meaning each one builds on top of those beneath it. So all the concepts in 1NF also carry over to 2NF, and so on.

Normal Form – 0NF

We begin with a table that has not been normalized and, as a result, does not adhere to the normalization rules of relational databases:

Employee_ID	Name	Department	Languages
1	Ana Gómez	HR	English, French
2	Juan Pérez	IT	English
3	Laura Díaz	Marketing	German, English, Portuguese

Problems:

- The languages are lists.
- What happens if a new employee comes in and is also fromMarketing or HR systems? I'm going to have duplicate data.

The First Normal Form – 1NF

The First Normal Form (1NF) is a fundamental concept in relational database normalization.

For a table to comply with 1NF, it must meet all of the following conditions:

1) **Atomicity of attributes (indivisible values):** Each column must contain atomic values, meaning they cannot be divided further. Lists, sets, arrays, or fields with multiple values are not allowed.
2) **No repeating groups of columns:** There should be no groups of columns that represent the same thing (e.g., Phone1, Phone2). If multiple related values exist, they should be moved to a separate table.
3) **Each row must be unique:** Every record should be uniquely identifiable using a primary key.
4) **Consistent data types in each column:** Each column must store data of the same type and domain (for example, a date column should contain only dates).

Employee_ID	First_Name	Last_Name	Department	Language
1	Ana	Gómez	HR	English
1	Ana	Gómez	HR	French
2	Juan	Pérez	IT	English
3	Laura	Díaz	Marketing	German
3	Laura	Díaz	Marketing	English
3	Laura	Díaz	Marketing	Portuguese

The First Normal Form – 1NF

Here, what we did was separate it, which **is the first normal form**, but we still have data redundancy.

What are the problems? We have redundancy in both the person's name and the department they belong to—they are duplicated and even triplicated in this case. This limits scalability, meaning our table cannot keep growing efficiently.

For example, if we add a new employee who speaks five languages, we would need five new records, which is very inefficient. **This is exactly where we need to move to the second normal form.**

Employee_ID	First_Name	Last_Name	Department	Language
1	Ana	Gómez	HR	English
1	Ana	Gómez	HR	French
2	Juan	Pérez	IT	English
3	Laura	Díaz	Marketing	German
3	Laura	Díaz	Marketing	English
3	Laura	Díaz	Marketing	Portuguese

The Second Normal Form – 2NF

The Second Normal Form (2NF) is the next step in normalization, aimed at eliminating partial redundancy that may remain after 1NF.

For a table to comply with 2NF, it must meet all of these conditions:

1) **Be in 1NF**: The table must already comply with First Normal Form (atomic values, unique rows, no repeating groups).
2) **No partial dependencies**: No non-key attribute should depend on only part of the primary key if the key is composite.
    - This means that if the primary key consists of more than one column, each non-key column must depend on the whole key, not just a part of it.
3) **Every non-key column fully depends on the primary key**: This eliminates partial redundancy and improves database scalability and maintainability.

The Second Normal Form – 2NF

Employee_ID	First_Name	Last_Name	Department	Language
1	Ana	Gómez	HR	English
1	Ana	Gómez	HR	French
2	Juan	Pérez	IT	English
3	Laura	Díaz	Marketing	German
3	Laura	Díaz	Marketing	English
3	Laura	Díaz	Marketing	Portuguese

Problems:

- Redundancy in employee name and department.
- Limited scalability if more attributes are added per employee.

Employees

Employee_ID	First_Name	Last_Name	Department
1	Ana	Gómez	HR
2	Juan	Pérez	IT
3	Laura	Díaz	Marketing

Employee_Languages

Employee_ID	Language
1	English
1	French
2	English
3	German
3	English
3	Portuguese

What is a Transitive Dependency?

A **transitive dependency** happens when a non-key attribute depends on another non-key attribute, instead of depending directly on the primary key.

If **A** → **B** and **B** → **C**, then **A** → **C** is a transitive dependency.

Where:

A = primary key

B and C = non-key attributes

This means that some data in the table can be derived indirectly through another attribute, which causes redundancy.

What is a Transitive Dependency?

Let’s say each **department** has additional information — for example, a department name and a location.

That gives us the following dependencies:

Employee_ID → Department

Department → Department_Location

Therefore, Department_Location depends transitively on Employee_ID (through Department).

Employee_ID	First_Name	Last_Name	Department
1	Ana	Gómez	HR
2	Juan	Pérez	IT
3	Laura	Díaz	Marketing

The Third Normal Form – 3NF

A table is in **Third Normal Form (3NF)** if it satisfies:

1) It is already in Second Normal Form (2NF).
2) There are no transitive dependencies, meaning:
   - Every non-key attribute depends only on the primary key,
   - and not on another non-key attribute.

Employees

Employee_ID	First_Name	Last_Name	Department_ID
1	Ana	Gómez	1
2	Juan	Pérez	2
3	Laura	Díaz	3
4	Martin	López	2
5	Sofia	Ramírez	1

Departments

Department_ID	Department_Name
1	HR
2	IT Systems
3	Marketing

Employees_Departments

Employee_ID	First_Name	Last_Name	Department_ID
1	Ana	Gómez	1
2	Juan	Pérez	2
3	Laura	Díaz	3
4	Martin	López	2 ← new (IT Systems)
5	Sofia	Ramírez	1 ← new (HR)

Example Library

This table not is in **First Normal Form (1NF):**

Book_ID	Title	Author	Publisher	Reader_Name	Return_Date
1001	Java	Author 1	Planeta	Frank Valencia Pinto	01/01/2024
1002	C#	Author 2	Anaya	Luis Gomez Fuentes	01/02/2024
1003	Python	Author 3	Parainfo	Ana Terán Perez	01/03/2024
1004	JavaScript	Author 4	Sello	Juan Roque Gomez	01/04/2024

Conditions for 1NF

To meet First Normal Form, the table must:

- Have atomic values (no multiple values in one field).
- Have a unique identifier (primary key).
- Have no repeating groups or multi-value attributes.
- Each column stores only one type of data.

Example Library

This table is in **First Normal Form (1NF):**

Book_ID	Title	Author	Publisher	Reader_FirstName	Reader_MiddleName	Reader_LastName	Return_Date
1001	Java	Author 1	Planeta	Frank	Valencia	Pinto	01/01/2024
1002	C#	Author 2	Anaya	Luis	Gomez	Fuentes	01/02/2024
1003	Python	Author 3	Parainfo	Ana	Terán	Perez	01/03/2024
1004	JavaScript	Author 4	Sello	Juan	Roque	Gomez	01/04/2024

Conditions for 2NF

To satisfy 2NF, the table must:

Already be in 1NF.

Remove partial dependencies, meaning all non-key attributes must depend on the whole primary key, not just part of it.

Example Library

This table is in **Second Normal Form (2NF):**

Book_ID	Title	Author	Publisher
1001	Java	Author 1	Planeta
1002	C#	Author 2	Anaya
1003	Python	Author 3	Parainfo
1004	JavaScript	Author 4	Sello

Reader_FirstName	Reader_MiddleName	Reader_LastName
Frank	Valencia	Pinto
Luis	Gomez	Fuentes
Ana	Terán	Perez
Juan	Roque	Gomez

Return_Date
01/01/2024
01/02/2024
01/03/2024
01/04/2024

Books

Book_ID	Title	Author	Publisher
1001	Java	Author 1	Planeta
1002	C#	Author 2	Anaya
1003	Python	Author 3	Parainfo
1004	JavaScript	Author 4	Sello

Readers

Reader_ID	First_Name	Last_Name	Middle_Name
501	Frank	Valencia	Pinto
502	Luis	Gomez	Fuentes
503	Ana	Terán	Perez
504	Juan	Roque	Gomez

Returns

Book_ID	Reader_ID	Return_Date
1001	501	01/01/2024
1002	502	01/02/2024
1003	503	01/03/2024
1004	504	01/04/2024

Example Library

This table is in **Second Normal Form (2NF):**

Books

Book_ID	Title	Author	Publisher
1001	Java	Author 1	Planeta
1002	C#	Author 2	Anaya
1003	Python	Author 3	Parainfo
1004	JavaScript	Author 4	Sello

Readers

Reader_ID	First_Name	Last_Name	Middle_Name
501	Frank	Valencia	Pinto
502	Luis	Gomez	Fuentes
503	Ana	Terán	Perez
504	Juan	Roque	Gomez

Returns

Book_ID	Reader_ID	Return_Date
1001	501	01/01/2024
1002	502	01/02/2024
1003	503	01/03/2024
1004	504	01/04/2024

Example Library

This table is in **Third Normal Form (3NF):**

Books

Book_ID	Title	Author	Publisher
1001	Java	Author 1	Planeta
1002	C#	Author 2	Anaya
1003	Python	Author 3	Parainfo
1004	JavaScript	Author 4	Sello

Books

Book_ID	Title
1001	Java
1002	C#
1003	Python
1004	JavaScript

Authors

Author_ID	Author_Name
601	Author 1
602	Author 2
603	Author 3
604	Author 4

Publishers

Publisher_ID	Publisher_Name
701	Planeta
702	Anaya
703	Parainfo
704	Sello

Book_Author

Book_ID	Author_ID
1001	601
1002	602
1003	603
1004	604

Book_Publisher

Book_ID	Publisher_ID
1001	701
1002	702
1003	703
1004	704

Readers

Reader_ID	First_Name	Last_Name	Middle_Name
501	Frank	Valencia	Pinto
502	Luis	Gomez	Fuentes
503	Ana	Terán	Perez
504	Juan	Roque	Gomez

Returns

Book_ID	Reader_ID	Return_Date
1001	501	01/01/2024
1002	502	01/02/2024
1003	503	01/03/2024
1004	504	01/04/2024