DECEMBER 21, 2022 / #DATABASE

Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples

Kolade Chris

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.

In this article, we'll look at what database normalization is in detail and its purpose. We'll also take a look at the types of normalization – 1NF, 2NF, 3NF – with examples.

What We'll Cover

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 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.

The First Normal Form – 1NF

For a table to be in the first normal form, it must meet the following criteria:

- a single cell must not hold more than one value (atomicity)
- there must be a primary key for identification

table

The Second Normal Form – 2NF

The 1NF only eliminates repeating groups, not redundancy. That's why there is 2NF.

A table is said to be in 2NF if it meets the following criteria:

- it's already in 1NF
- has no partial dependency. That is, all non-key attributes are fully dependent on a primary key.

The Third Normal Form – 3NF

When a table is in 2NF, it eliminates repeating groups and redundancy, but it does not eliminate transitive partial dependency.

This means a non-prime attribute (an attribute that is not part of the candidate's key) is dependent on another non-prime attribute. This is what the third normal form (3NF) eliminates.

So, for a table to be in 3NF, it must:

- be in 2NF
- have no transitive partial dependency.

Examples of 1NF, 2NF, and 3NF

Database normalization is quite technical, but we will illustrate application needs to store data about the company's employees and it starts out by creating the following table of employees:

employee_id	name	job_code	job	state_code	home_state
E001	Alice	J01	Chef	26	Michigan
E001	Alice	J02	Waiter	26	Michigan
E002	Bob	J02	Waiter	56	Wyoming
E002	Bob	J03	Bartender	56	Wyoming
E003	Alice	J01	Chef	56	Wyoming

All the entries are atomic and there is a composite primary key (employee_id, job_code) so the table is in the **first normal form (1NF)**.

But even if you only know someone's `employee_id`, then you can determine their `name`, `home_state`, and `state_code` (because they should be the same person). This means `name`, `home_state`, and `state_code` are dependent on `employee_id` (a part of primary composite key). So, the table is not in **2NF**. We should separate them to a different table to make it 2NF.

Example of Second Normal Form (2NF)

`employee_roles` Table

E001	J02
E002	J02
E002	J03
E003	J01

`employees` Table

employee_id	name	state_code	home_state
E001	Alice	26	Michigan
E002	Bob	56	Wyoming
E003	Alice	56	Wyoming

`jobs` table

job_code	job
J01	Chef
J02	Waiter
J03	Bartender

`home_state` is now dependent on `state_code`. So, if you know the `state_code`, then you can find the `home_state` value.

To take this a step further, we should separate them again to a different table to make it 3NF.

`employee_roles` Table

employee_id	job_code
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

`employees` Table

employee_id	name	state_code
E001	Alice	26
E002	Bob	56
E003	Alice	56

`jobs` Table

job_code	job
J01	Chef
J02	Waiter
J03	Bartender

`states` Table

56	Wyoming

Now our database is in 3NF.

Conclusion

This article took you through what database normalization is, its purpose, and its types. We also look at those types of normalization and the criteria a table must meet before it can be certified to be in any of them.

It is worth noting that most tables don't exceed the 3NF limit, but you can also take them to 4NF and 5NF, depending on requirements and the size of the data at hand.

If you find the article helpful, don't hesitate to share it with friends and family.

Kolade Chris
I'm a software developer and tech writer focusing on frontend technologies

If you read this far, thank the author to show them you care.

Say Thanks

Build an AI Chatbot

Python Code Examples

HTTP Networking in JS

Learn Algorithms in JS

Learn PHP

Learn Swift

Learn Node.js

Learn Solidity

Learn JS Modules

REST API Best Practices

Learn to Build REST APIs

What is Programming?

Open Source for Devs

Write React Unit Tests

How to Write Clean Code

Learn Java

Learn Golang

Learn CSS Grid

Learn Express.js

Learn Apache Kafka

Front-End JS Development

Intermediate TS and React