Need a discount on popular programming courses? Find them here. View offers

Aman Goel | 02 Jan, 2023

Normalization in DBMS: 1NF, 2NF, 3NF, and BCNF [Examples]

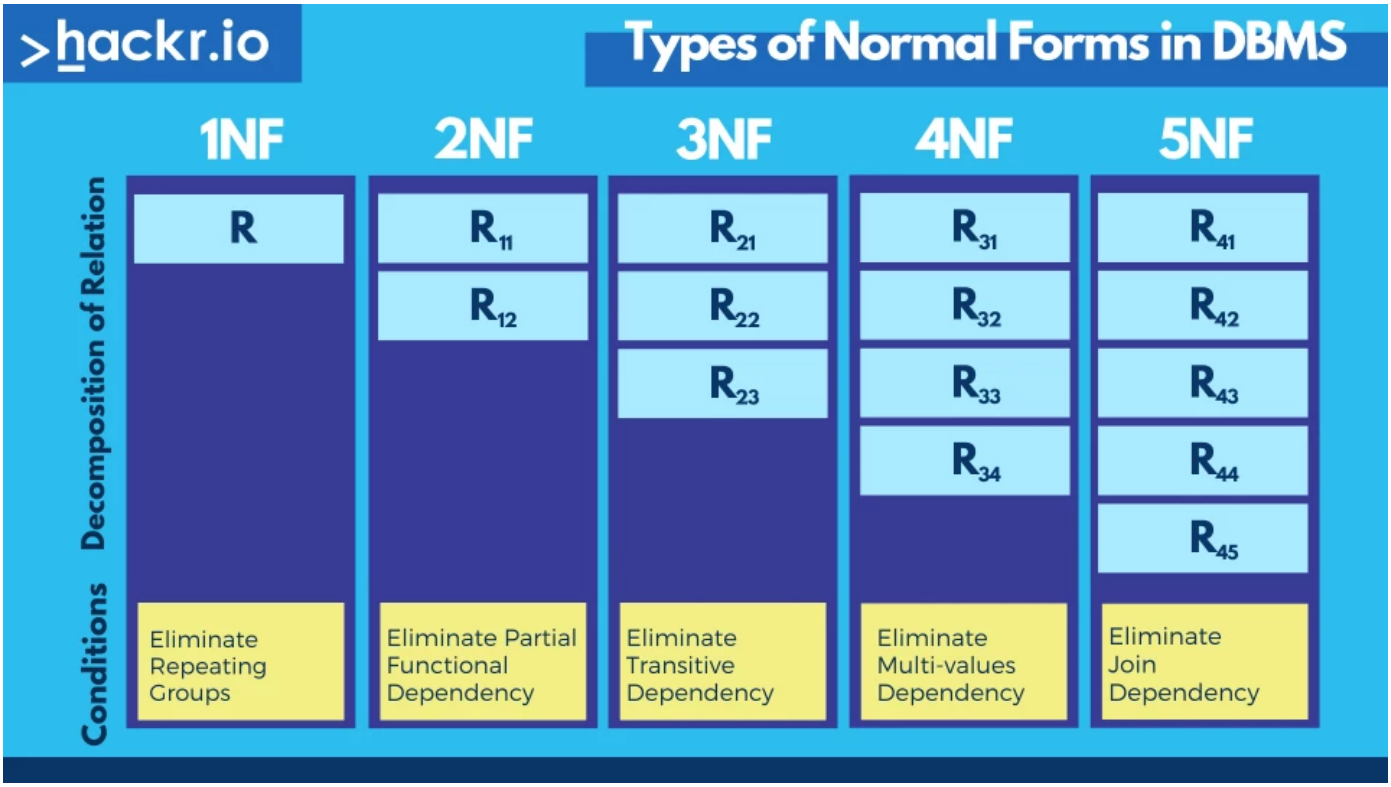
When developing the schema of a relational database, one of the most important aspects to be taken into account is to ensure that the duplication of data is minimized. We do this by carrying out database normalization, an important part of the database schema design process.

Here, we explain normalization in DBMS, explaining 1NF, 2NF, 3NF, and BCNF with explanations. First, let's take a look at what normalization is and why it is important.

What is Normalization in D

Database normalization is a technique that helps desig
optimal way. The core idea of database normalization i
subtables and store pointers to data rather than replic


Smarter business tools
intu qb


>hackr.io
Types of Normal Forms in DBMS
1NF
2NF
3NF
4NF
5NF
Decomposition of Relation
R
R11
R12
R21
R22
R23
R31
R32
R33
R34
R41
R42
R43
R44
R45
Conditions
Eliminate Repeating Groups
Eliminate Partial Functional Dependency
Eliminate Transitive Dependency
Eliminate Multi-values Dependency
Eliminate Join Dependency

There are two primary reasons why database normalization is used. First, it helps reduce the amount of storage needed to store the data. Second, it prevents data conflicts that may creep in because of the existence of multiple copies of the same data.

If a database isn't normalized, then it can result in less efficient and generally slower systems, and potentially even inaccurate data. It may also lead to excessive disk I/O usage and bad performance.

What is a Key?

You should also be aware of what a key is. A key is an attribute that helps identify a row in a table. There are seven different types, which you'll see used in the explanation of the various normalizations:

- Candidate Key
- Primary Key
- Foreign Key
- Super Key
- Alternate Key
- Composite Key
- Unique Key

To understand (DBMS)normalization with example tables, let's assume that we are storing the details of courses and instructors in a university. Here is what a sample database could look like:

Course code	Course venue	Instructor Name	Instructor's phone number
CS101	Lecture Hall 20	Prof. George	+1 6514821924
CS152	Lecture Hall 21	Prof. Atkins	+1 6519272918
CS154	CS Auditorium	Prof. George	+1 6514821924

Here, the data basically stores the course code, course venue, instructor name, and instructor's phone number. At first, this design seems to be good. However, issues start to develop once we need to modify information. For instance, suppose, if Prof. George changed his mobile number. In such a situation, we will have to make edits in 2 places.

What if someone just edited the mobile number against CS101, but forgot to edit it for CS154? This will lead to stale/wrong information in the database. This problem can be easily tackled by dividing our table into 2 simpler tables:

Table 1 (Instructor):

- Instructor ID
- Instructor Name
- Instructor mobile number

Table 2 (Course):

- Course code
- Course venue
- Instructor ID

Now, our data will look like the following:

1	Prof. George	+1 6514821924
2	Prof. Atkins	+1 6519272918

Table 2 (Course):

Course code	Course venue	Instructor ID
CS101	Lecture Hall 20	1
CS152	Lecture Hall 21	2
CS154	CS Auditorium	1

Basically, we store the instructors separately and in the course table, we do not store the entire data of the instructor. Rather, we store the ID of the instructor. Now, if someone wants to know the mobile number of the instructor, the table. Also, if we were to change the mobile number of exactly one place. This avoids the stale/wrong data pr

Further, if you observe, the mobile number now need r stored it in just 1 place. This also saves storage. This m simple example. However, think about the case when t instructors and for each instructor, we have to store nc

Database Management System (DBMS) & SQL : Complete Pack 2023

Types of DBMS Normalization

There are various normal forms in DBMS. Each normal form has an importance that helps optimize the database to save storage and reduce redundancies. We explain normalization in DBMS with examples below.

First Normal Form (1NF)

The first normal form simply says that each cell of a table should contain exactly one value. Assume we are storing the courses that a particular instructor takes, we can store it like this:

Instructor's name	**Course code**
Prof. George	(CS101, CS154)
Prof. Atkins	(CS152)

Here, the issue is that in the first row, we are storing 2 courses against Prof. George. This isn't the optimal way since that's now how SQL databases are designed to be used. A better method would be to store the courses separately. For instance:

Instructor's name	**Course**
Prof. George	CS101
Prof. George	CS154

This way, if we want to edit some information related to CS101, we do not have to touch the data corresponding to CS154. Also, observe that each row stores unique information. There is no repetition. This is the First Normal Form.

Data redundancy is higher in 1NF because there are multiple columns with the same in multiple rows. 1NF is not so focused on eliminating redundancy as much as it is focused on eliminating repeating groups.

Second Normal Form (2NF)

For a table to be in second normal form, the following 2 conditions must be met:

- The table should be in the first normal form.
- The primary key of the table should have exactly 1 column.

The first point is obviously straightforward since we just studied 1NF. Let us understand the second point: a 1-column primary key. A primary key is a set of columns that uniquely identifies a row. Here, no 2 rows have the same primary keys.

Course code	Course venue	Instructor Name	Instructor's phone number
CS101	Lecture Hall 20	Prof. George	+1 6514821924
CS152	Lecture Hall 21	Prof. Atkins	+1 6519272918
CS154	CS Auditorium	Prof. George	+1 6514821924

In this table, the course code is unique so that becomes our primary key. Let us take another example of storing student enrollment in various courses. Each student may enroll in multiple courses. Similarly, each course may have multiple enrollments. A sample table may look like this (student name and course code):

Student name	**Course code**
Rahul	CS152
Rajat	CS101
Rahul	CS154
Raman	CS101

Here, the first column is the student name and the second column is the course taken by the student.

Clearly, the student name column isn't unique as we can see that there are 2 entries corresponding to the name 'Rahul' in row 1 and row 3. Similarly, the course code column is not unique as we can see that there are 2 entries corresponding to course code CS101 in row 2 and row 4.

However, the tuple (student name, course code) is unique since a student cannot enroll in the same course more than once. So, these 2 columns when combined form the primary key for the database.

As per the second normal form definition, our enrollment table above isn't in the second

›hackr.io

Student name	**Enrolm**
Rahul	1

Here the second column is unique and it indicates the enrollment number for the student. Clearly, the enrollment number is unique. Now, we can attach each of these enrollment numbers with course codes.

Courses:

Course code	Enrolment number
CS101	2
CS101	3
CS152	1
CS154	1

These 2 tables together provide us with the exact same information as our original table.

Third Normal Form (3NF)

Before we delve into the details of third normal form, let us understand the concept of a functional dependency on a table.

Column A is said to be functionally dependent on column B if changing the value of A may require a change in the value of B. As an example, consider the following table:

Course code	Course venue	Instructor's name	Department
MA214	Lecture Hall 18	Prof. Ge	
ME112	Auditorium building	Prof. Jo	

Here, the department column is dependent on the professor name column. This is because if in a particular row, we change the name of the professor, we will also have to change the department value. As an example, suppose MA214 is now taken by Prof. Ronald who happens to be from the mathematics department, the table will look like this:

Course code	Course venue	Instructor's name	Department
MA214	Lecture Hall 18	Prof. Ronald	Mathematics Department
ME112	Auditorium building	Prof. John	Electronics Department

Here, when we changed the name of the professor, we also had to change the department column. This is not desirable since someone who is updating the database may remember to change the name of the professor, but may forget updating the department value. This can cause inconsistency in the database.

Third normal form avoids this by breaking this into separate tables:

Course code	Course venue	Instructor's ID
MA214	Lecture Hall 18	
ME112	Auditorium building,	

Here, the third column is the ID of the professor who's

Instructor's ID	Instructor's Name	Department

2	Prof. John	Electronics Department

Here, in the above table, we store the details of the professor against his/her ID. This way, whenever we want to reference the professor somewhere, we don't have to put the other details of the professor in that table again. We can simply use the ID.

Therefore, in the third normal form, the following conditions are required:

- The table should be in the second normal form.
- There should not be any functional dependency.

Boyce-Codd Normal Form (BCNF)

The Boyce-Codd Normal form is a stronger generalization of the third normal form. A table is in Boyce-Codd Normal form if and only if at least one of the following conditions are met for each functional dependency A → B:

- A is a superkey
- It is a trivial functional dependency.

Let us first understand what a superkey means. To understand BCNF in DBMS, consider the following BCNF example table:

Course code	**Course venue**	**Instructor Name**	**Instructor's phone number**
CS101	Lecture Hall 20	Prof. George	+1 6514821924
CS152	Lecture Hall 21	Prof. At	
CS154	CS Auditorium	Prof. Ge	

Here, the first column (course code) is unique across v
Consider the combination of columns (course code, pr
across various rows. So, it is also a superkey. A superk

- Course code, professor name
- Course code, professor mobile number

A superkey whose size (number of columns) is the smallest is called a candidate key. For instance, the first superkey above has just 1 column. The second one and the last one have 2 columns. So, the first superkey (Course code) is a candidate key.

Boyce-Codd Normal Form says that if there is a functional dependency A → B, then either A is a superkey or it is a trivial functional dependency. A trivial functional dependency means that all columns of B are contained in the columns of A. For instance, (course code, professor name) → (course code) is a trivial functional dependency because when we know the value of course code and professor name, we do know the value of course code and so, the dependency becomes trivial.

Let us understand what's going on:

A is a superkey: this means that only and only on a superkey column should it be the case that there is a dependency of other columns. Basically, if a set of columns (B) can be determined knowing some other set of columns (A), th⊗ A should be a superkey. Superkey basically determines each row uniquely.

It is a trivial functional dependency: this means that t
dependency. For instance, we saw how the professor's
professor's name. This may create integrity issues sinc
name without changing the department. This may lead

Another example would be if a company had employee

Fourth normal form

A table is said to be in fourth normal form if there is no two or more, independent and multivalued data describing the relevant entity.

Fifth normal form

A table is in fifth normal form if:

- It is in its fourth normal form.
- It cannot be subdivided into any smaller tables without losing some form of information.

Normalization is Important for Database Systems

Normalization in DBMS is useful for designing the schema of a database such that there is no data replication which may possibly lead to inconsistencies. While designing the schema for applications, we should always think about how we can make use of these forms.

If you want to learn more about SQL, check out our post on the best SQL certifications. You can also read about SQL vs MySQL to learn about what the two are. To become a data engineer, you'll need to learn about normalization and a lot more, so get started today.

Frequently Asked Questions

1. Does database normalization reduce the

Yes, database normalization does reduce database siz
the database disk storage use becomes smaller.

2. Which normal form can remove all the a

5NF will remove all anomalies. However, generally, mos

Database normalization increases the number of tables. This is because we split tables into sub-tables in order to eliminate redundant data.

4. What is the Difference between BCNF and 3NF?

BCNF is an extension of 3NF. The primary difference is that it removes the transitive dependency from a relation.

People are also reading:

- SQL Courses
- SQL Certifications
- SQL Books
- Download SQL Cheat Sheet PDF
- Top DBMS Interview Questions & Answers
- Difference between MongoDB vs MySQL
- Create Database in MySQL
- What is Stored Procedure?
- Difference between OLTP vs OLAP
- What is MongoDB?
- Basic SQL Command

STAY IN LOOP TO BE AT

Subscribe to our

Enter email here

Subscribe Now

By Aman Goel

Entrepreneur, Coder, Speed-cuber, Blogger, fan of Air crash investigation! Aman Goel is a Computer Science Graduate from IIT Bombay. Fascinated by the world of technology he went on to build his own start-up - AllinCall Research and Solutions to build the next generation of Artificial Intelligence, Machine Learning and Natural Language Processing based solutions to power businesses.

View all post by the author

Disclosure: Hackr.io is supported by its audience. When you pu
earn an affiliate commission.

In this article

- Database Normalization Example

Learn More

What Does an SQL Developer Do? | Why Learn SQL in 2023

SQL

Download SQL Injection Cheat Sheet PDF for Quick References

sql

SQL vs MySQL: What's the Difference and Which One to Choose

MySql Sql

Find Your Place In The World

Search More Roles

Please login to leave c

Hackr Team

This video might be helpful to you: https://www.yo

2 years ago

Sagar Jaybhay

Very very nice explanation

3 years ago

Tiago Mendes

Thank you for your the tutorial, it was explained well and easy to folow!

1 year ago

Always be in the loop.

Get news once a week, and don't worry — no spam.

Manage here

Your email

Subscribe

Programming

DevOps

Data Science

Articles

Roadmaps

Programming Tips

Jobs

Help center

About us

We ❤ Feedback

Advertise / Partner

Write for us

Privacy Policy

Cookie Policy

Disclosure Policy

Terms and Conditions

Disclaimer

Refund Policy

Follow us

Disclosure: This page may contain affliate links, meaning when you click the links and make a purchase, we receive a commission.

A RAPTIVE PARTNER SITE