

NORMALIZATION

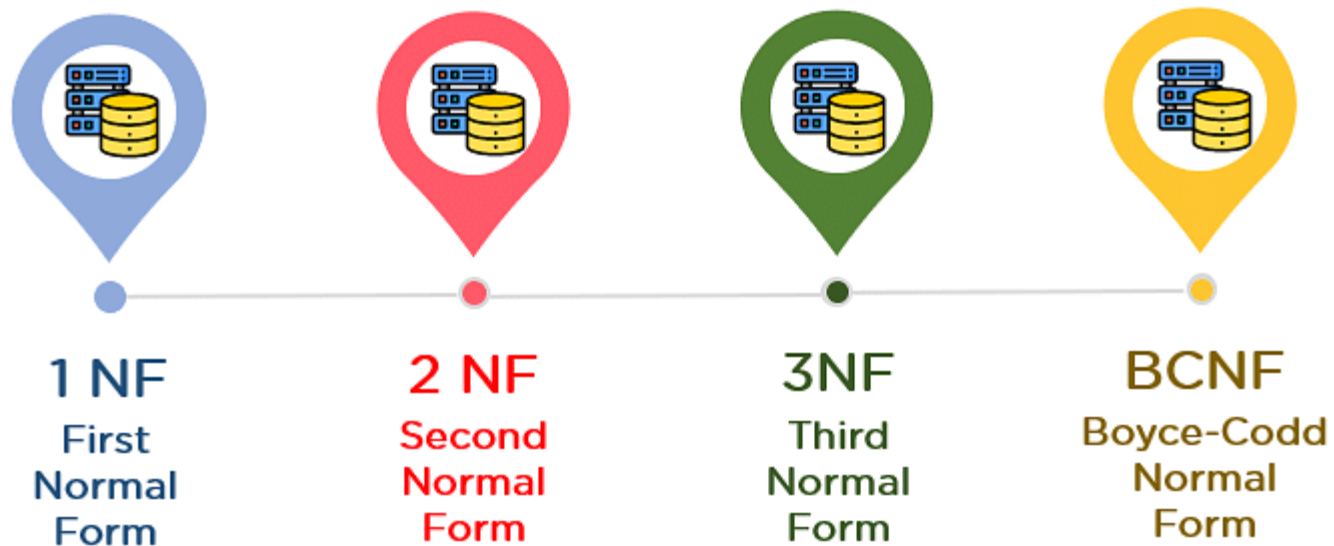
Normalization is the process to eliminate data redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

Normalization organizes the columns and tables of a database to ensure that database integrity constraints properly execute their dependencies. It is a systematic technique of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update, and Deletion anomalies.

Types of normalization

In 1970 Edgar F. Codd defined the First Normal Form.

Now let's understand the types of Normal forms with the help of examples.



1st Normal Form (1NF)

- A table is referred to as being in its First Normal Form if atomicity of the table is 1.
- Here, atomicity states that a single cell cannot hold multiple values. It must hold only a single-valued attribute.
- The First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Now you will understand the First Normal Form with the help of an example.

Below is a students' record table that has information about student roll number, student name, student course, and age of the student.

	rollno	name	course	age
▶	1	Rahul	c/c++	22
	2	Harsh	java	18
	3	Sahil	c/c++	23
	4	Adam	c/c++	22
	5	Lisa	java	24
	6	James	c/c++	19
*	NULL	NULL	NULL	NULL

In the student's record table, you can see that the course column has two values. Thus it does not follow the First Normal Form. Now, if you use the First Normal Form to the above table, you get the below table as a result.

	rollno	name	course	age
▶	1	Rahul	c	22
	1	Rahul	c++	22
	2	Harsh	java	18
	3	Sahil	c	23
	3	Sahil	c++	23
	4	Adam	c	22
	4	Adam	c++	22
	5	Lisa	java	24
	6	James	c	19
	6	James	c++	19

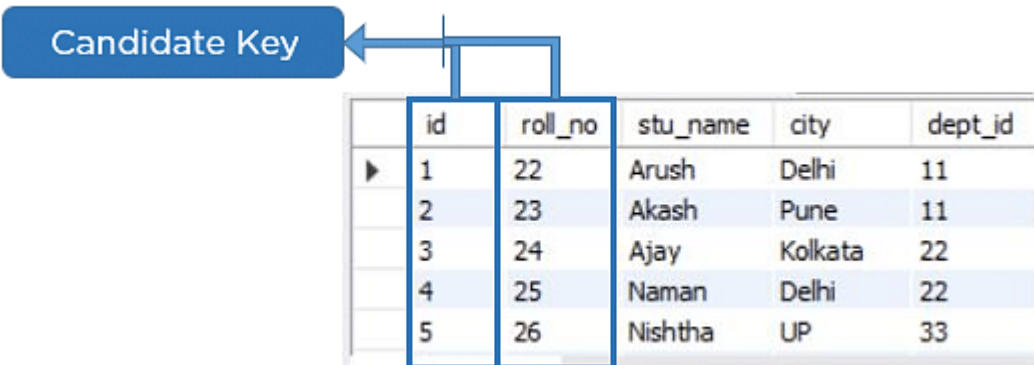
By applying the First Normal Form, you achieve atomicity, and also every column has unique values.

Before proceeding with the Second Normal Form, get familiar with Candidate Key and Super Key.

Candidate Key

A candidate key is a set of one or more columns that can identify a record uniquely in a table, and YOU can use each candidate key as a [Primary Key](#).

Now, let's use an example to understand this better.



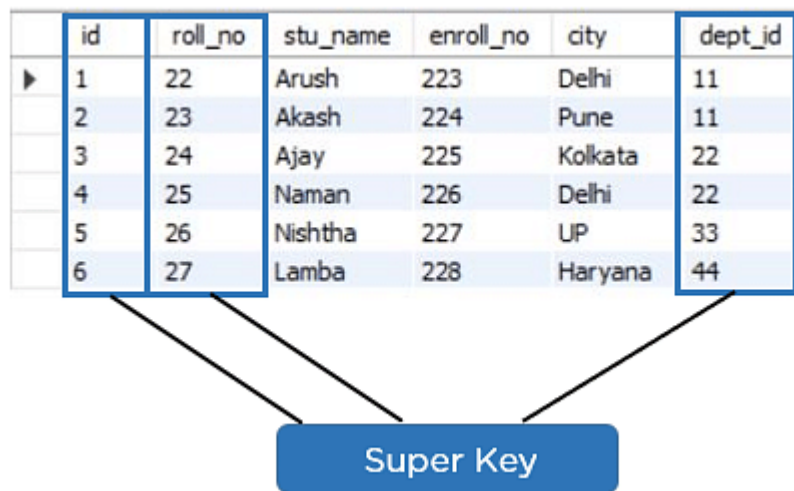
A diagram illustrating a candidate key. A blue box labeled "Candidate Key" has an arrow pointing to the first two columns of a table, "id" and "roll_no". These two columns are highlighted with a blue border. The table contains five rows of data.

	id	roll_no	stu_name	city	dept_id
▶	1	22	Arush	Delhi	11
	2	23	Akash	Pune	11
	3	24	Ajay	Kolkata	22
	4	25	Naman	Delhi	22
	5	26	Nishtha	UP	33

Super Key

Super key is a set of over one key that can identify a record uniquely in a table, and the Primary Key is a subset of Super Key.

Let's understand this with the help of an example.



Second Normal Form (2NF)

The first condition for the table to be in Second Normal Form is that the table has to be in First Normal Form. The table should not possess partial dependency. The partial dependency here means the proper subset of the candidate key should give a non-prime attribute.

Now understand the Second Normal Form with the help of an example.

Consider the table Location:

	cust_id	storeid	store_location
▶	1	D1	Toronto
	2	D3	Miami
	3	T1	California
	4	F2	Florida
	5	H3	Texas

The Location table possesses a composite primary key cust_id, store id. The non-key attribute is store location. In this case, store location only depends on storeid, which is a part of the primary key. Hence, this table does not fulfill the second normal form.

To bring the table to Second Normal Form, you need to split the table into two parts. This will give you the below tables:

	cust_id	storeid
►	1	D1
	2	D3
	3	T1
	4	F2
	5	H3

	storeid	store_location
►	D1	Toronto
	D3	Miami
	T1	California
	F2	Florida
	H3	Texas

As you have removed the partial functional dependency from the location table, the column store location entirely depends on the primary key of that table, storeid.

Now that you understood the 1st and 2nd Normal forms, you will look at the next part of this Normalization in SQL tutorial.

Third Normal Form (3NF)

- The first condition for the table to be in Third Normal Form is that the table should be in the Second Normal Form.
- The second condition is that there should be no transitive dependency for non-prime attributes, which indicates that non-prime attributes (which are not a part of the candidate key) should not depend on other non-prime attributes in a table. Therefore, a transitive dependency is a functional dependency in which $A \rightarrow C$ (A determines C) indirectly, because of $A \rightarrow B$ and $B \rightarrow C$ (where it is not the case that $B \rightarrow A$).
- The third Normal Form ensures the reduction of data duplication. It is also used to achieve data integrity.

Below is a student table that has student id, student name, subject id, subject name, and address of the student as its columns.

	stu_id	name	subid	sub	address
►	1	Arun	11	SQL	Delhi
	2	Varun	12	Java	Bangalore
	3	Harsh	13	C++	Delhi
	4	Keshav	12	Java	Kochi

In the above student table, stu_id determines subid, and subid determines sub. Therefore, stu_id determines sub via subid. This implies that the table possesses a transitive functional dependency, and it does not fulfill the third normal form criteria.

Now to change the table to the third normal form, you need to divide the table as shown below:

	stu_id	name	subid	address
►	1	Arun	11	Delhi
	2	Varun	12	Bangalore
	3	Harsh	13	Delhi
	4	Keshav	12	Kochi

	subid	subject
►	11	SQL
	12	java
	13	C++
	12	Java

As you can see in both the tables, all the non-key attributes are now fully functional, dependent only on the primary key. In the first table, columns name, subid, and addresses only depend on stu_id. In the second table, the sub only depends on subid.

Fourth Normal Form (4NF)

The Fourth Normal Form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF, and 3NF) and the [Boyce-Codd Normal Form \(BCNF\)](#). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

Properties

A relation R is in 4NF if and only if the following conditions are satisfied:

1. It should be in the Boyce-Codd Normal Form (BCNF).

2. The table should not have any Multi-valued Dependency.

A table with a multivalued dependency violates the normalization standard of the Fourth Normal Form (4NF) because it creates unnecessary redundancies and can contribute to inconsistent data. To bring this up to 4NF, it is necessary to break this information into two tables.

Example: Consider the database table of a class that has two relations R1 contains student ID (SID) and student name (SNAME) and R2 contains course id (CID) and course name (CNAME).

Table R1

SID	SNAME
S1	A
S2	B

Table R2

CID	CNAME
C1	C
C2	D

When their cross-product is done it resulted in multivalued dependencies.

Table R1 X R2

SID	SNAME	CID	CNAME
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

Multivalued dependencies (MVD) are:

SID \twoheadrightarrow CID; SID \twoheadrightarrow CNAME; SNAME \twoheadrightarrow CNAME

Join Dependency

Join decomposition is a further generalization of multivalued dependencies. If the join of R1 and R2 over C is equal to relation R then we can say that a join dependency (JD) exists, where R1 and R2 are the decomposition R1 (A, B, C) and R2(C, D) of a given relations R (A, B, C, D). Alternatively, R1 and R2 are a lossless decomposition of R. A JD $\bowtie \{R1, R2, Rn\}$ is said to hold over a relation R if R1, R2, Rn is a lossless-join decomposition. The $\bowtie(A, B, C, D), (C, D)$ will be a JD of R if the join of joins attribute is equal to the relation R. Here, $\bowtie(R1, R2, R3)$ is used to indicate that relation R1, R2, R3 and so on are a JD

of R. Let R is a relation schema R1, R2, R3.....Rn
 be the decomposition of R. r(R) is said to satisfy join
 dependency if and only if

$$\bowtie_{i=1}^n \Pi_{R_i}(r) = r.$$

Joint Dependency

Example:
Table R1

Company	Product
C1	Pen drive
C1	mic
C2	speaker
C2	speaker

Company->->Product

Table R2

Agent	Company
Aman	C1
Aman	C2

Agent	Company
Mohan	C1

Agent->->Company

Table R3

Agent	Product
Aman	Pen drive
Aman	Mic
Aman	speaker
Mohan	speaker

Agent->->Product

Table R1⋈R2⋈R3

Company	Product	Agent
C1	Pen drive	Aman
C1	mic	Aman
C2	speaker	speaker

Company	Product	Agent
C1	speaker	Aman