

## Database Normalization

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The inventor of the relational model Edgar Codd proposed the theory of normalization with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of Boyce–Codd Normal Form.

Here is a list of Normal Forms

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce–Codd Normal Form)

Database **Normalization Example** can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. Without any normalization, all information is stored in one table as shown below.

Table 1

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

Here you see **Movies Rented column has multiple values**. Now let's move into 1st Normal Forms:

### ***1NF (First Normal Form) Rules***

- Each table cell should contain a single value.
- Each record needs to be unique.

The table 1 into **1NF Example**

Table 2

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

### **A. What is a KEY?**

A KEY is a value used to identify a record in a table uniquely. A KEY could be a single column or combination of multiple columns

Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

### What is a Primary Key?

A primary is a single column value used to identify a database record uniquely.

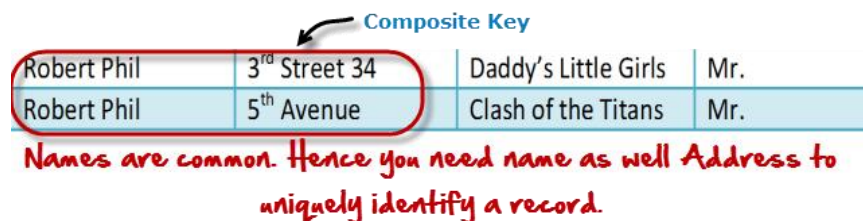
It has following attributes

- A primary key cannot be NULL
- A primary key value must be unique
- The primary key values should rarely be changed
- The primary key must be given a value when a new record is inserted.

### B. What is Composite Key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely

In our database, we have two people with the same name Robert Phil, but they live in different places.



Composite Key			
Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

*Names are common. Hence you need name as well Address to uniquely identify a record.*

Figure 1

Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.

### C. 2NF (Second Normal Form) Rules

- Rule 1– Be in 1NF
- Rule 2– Single Column Primary Key

It is clear that we can't move forward to make our simple database in 2<sup>nd</sup> Normalization form unless we partition the table above.

Table 3

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Table 4

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

We have divided our 1NF table into two tables. Table 3 and Table 4. Table 3 contains member information. Table 4 contains information on movies rented.

We have introduced a new column called Membership\_id which is the primary key for table 3. Records can be uniquely identified in Table 3 using membership id

### D. Database – Foreign Key

In Table 4, Membership\_ID is the Foreign Key

Foreign Key references the primary key of another Table! It helps connect your Tables

- A foreign key can have a different name from its primary key
- It ensures rows in one table have corresponding rows in another
- Unlike the Primary key, they do not have to be unique. Most often they aren't
- Foreign keys can be null even though primary keys can not

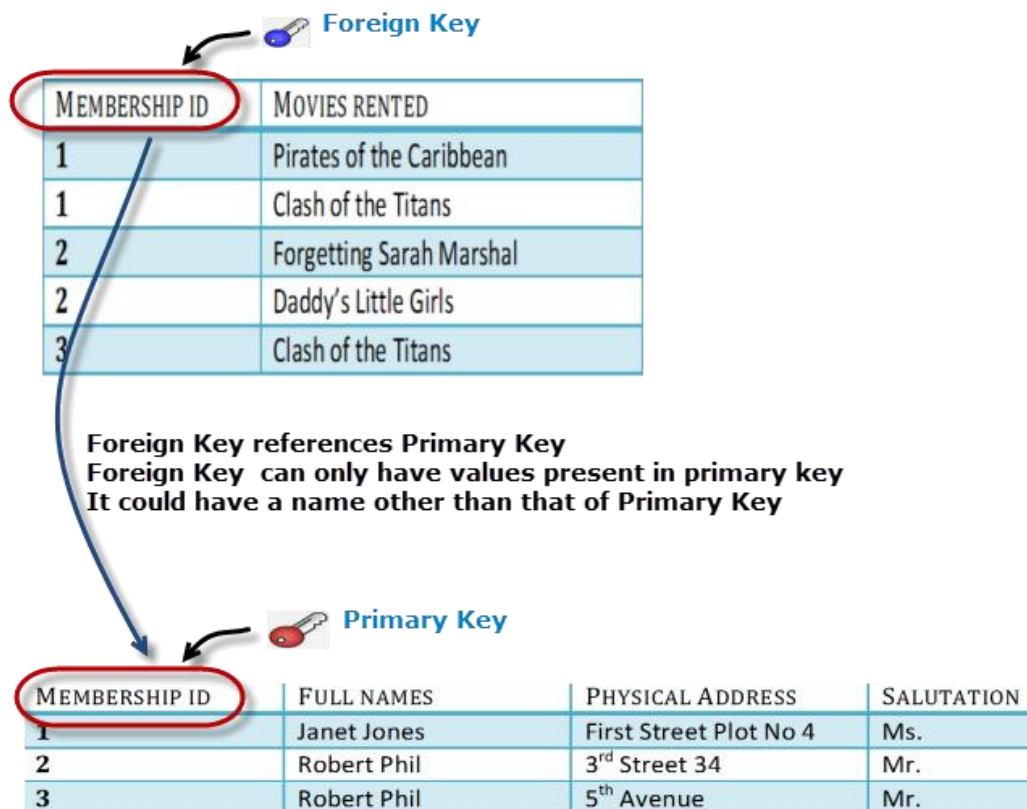


Figure 2

Why do you need a foreign key?

Suppose, a novice inserts a record in Table B such as

Insert a record in Table 2 where Member ID =101

MEMBERSHIP ID	MOVIES RENTED
101	Mission Impossible

But Membership ID 101 is not present in Table 1

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Database will throw an **ERROR**. This helps in referential integrity

Figure 3

You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.

The above problem can be overcome by declaring membership id from Table 4 as foreign key of membership id from Table 3

Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

### E. What are transitive functional dependencies?

A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change

Consider the table 1. Changing the non-key column Full Name may change Salutation.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Change in Name → May Change Salutation

Figure 4

## F. 3NF (Third Normal Form) Rules

- Rule 1– Be in 2NF
- Rule 2– Has no transitive functional dependencies

To move our 2NF table into 3NF, we again need to again divide our table.

Table 5

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3 <sup>rd</sup> Street 34	1
3	Robert Phil	5 <sup>th</sup> Avenue	1

Table 6

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 7

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

We have again divided our tables and created a new table which stores Salutations.

There are no transitive functional dependencies, and hence our table is in 3NF

In Table 7 Salutation ID is primary key, and in Table 5 Salutation ID is foreign to primary key in Table 7

Now our little example is at a level that cannot further be decomposed to attain higher normal forms of normalization. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of normalizations in brief in the following.

### **G. BCNF (Boyce–Codd Normal Form)**

Even when a database is in 3<sup>rd</sup> Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.

Sometimes BCNF is also referred as **3.5 Normal Form**.

#### **A. 4NF (Fourth Normal Form) Rules**

If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4<sup>th</sup> Normal Form.

#### **B. 5NF (Fifth Normal Form) Rules**

A table is in 5<sup>th</sup> Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

#### **C. 6NF (Sixth Normal Form) Proposed**

6<sup>th</sup> Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6<sup>th</sup> Normal Form in the near future.