

|  |
| --- |
| **What is Normalization ? Why should we use it?**  **Normalization** is a database design technique which organizes tables in a manner that reduces **redundancy and dependency of data**.  It **divides larger tables to smaller** **tables** and **link them using relationships**.  The inventor of the relational model **Edgar Codd** proposed the theory of normalization with the introduction of First**Normal Form** and he continued to extend theory with **Second and** **Third Normal Form.** Later he joined with **Raymond F. Boyce** to develop the theory of **Boyce-Codd Normal Form**. |

Theory of Normalization is still being developed further. For example there are discussions even on 6th Normal Form. **But in most practical applications normalization achieves its best in 3rd Normal Form**. The evolution of Normalization theories is illustrated below-

alt

Let’s learn Normalization with practical example -

Assume a video library maintains a database of movies rented out. Without any normalization all information is stored in one table as shown below.

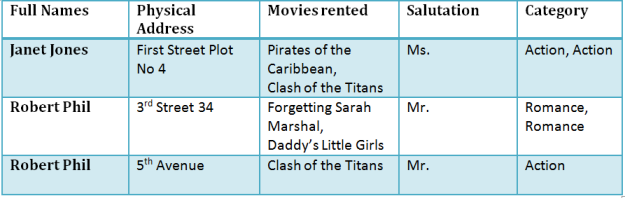


Table 1

Here you see **Movies  Rented column has multiple values**.

Now let’s move in to 1st Normal Form

**1NF Rules**

 Each table cell should contain single value.

* Each record needs to be unique.

The above table in 1NF-

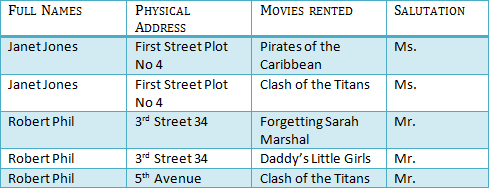


Table 1 : In 1NF Form

 Before we proceed lets understand a few things --

**What is a KEY ?**

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

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

**What is a primary Key?**

|  |  |
| --- | --- |
|  | A primary is a single column values used to uniquely identify a database record.  It has following attributes   * A primary key cannot be NULL * A primary key value must be unique * The primary key values can not be changed * The primary key must be given a value when a new record is inserted. |

**What is a 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 at different places.



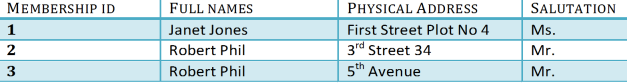
Hence we require both Full Name and Address to uniquely identify a record. This is a composite key.

Let’s move into 2NF

**2NF 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 2nd Normalization form unless we partition the table above.

Table 1

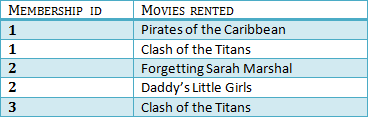


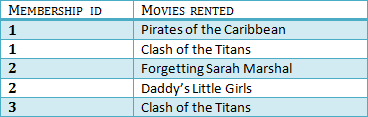
Table 2

 We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

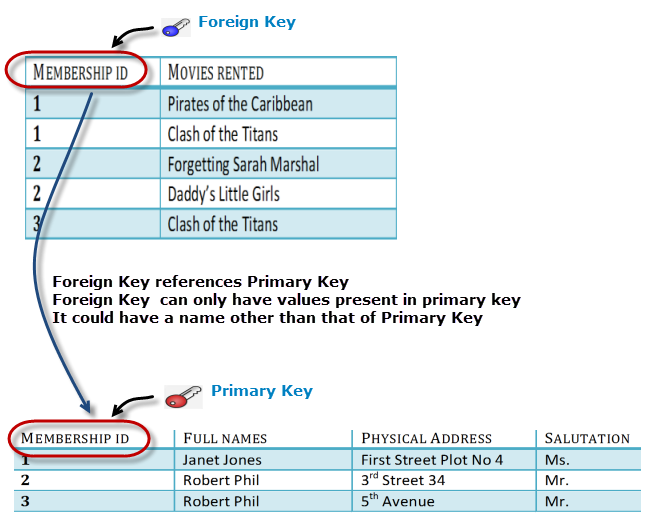
We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

**Introducing Foreign Key!**

 In Table 2, Membership\_ID is the foreign Key



|  |  |
| --- | --- |
|  | Foreign Key references 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 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 |



**Why do you need a foreign key ?**

 Suppose an idiot inserts a record in Table B such as

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.

Let’s move ito 3NF

**3NF Rules**

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| MEMBERSHIP ID | FULL NAME | PHISICAL ADDRESS | SULUTATION |
| 1 | Janet jones | First street plot No 4 | Ms. |
| 2 | Robert phil | 3rd street 34 | Mr. |
| 3 | Robert phil | 5th | Mr. |

TABLE 1

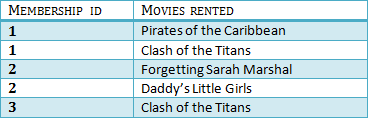


Table 2

 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 3 Salutation ID is primary key and in Table 1 Salutation ID is foreign to primary key in Table 3

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

**Boyce-Codd Normal Form (BCNF)**

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

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

**4th  Normal Form**

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

**5th  Normal Form**

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