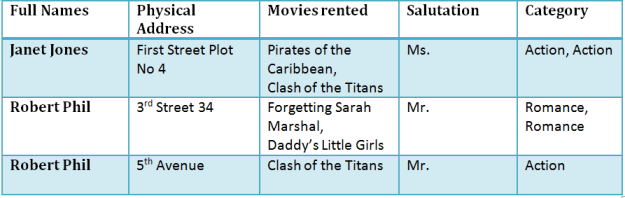
**Database Normalization Examples**

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

[[](https://www.guru99.com/images/NormalizationTable1.png)](https://www.guru99.com/images/NormalizationTable1.png)

[Table 1](https://www.guru99.com/images/NormalizationTable1.png)

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

**Database Normal Forms**

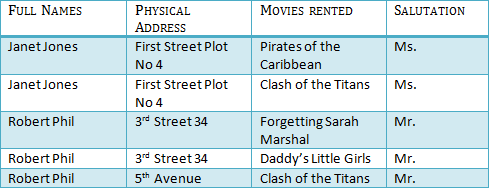
Now let's move into 1st Normal Forms

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

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

The above table in 1NF-

**1NF Example**

[[](https://www.guru99.com/images/1NF.png)](https://www.guru99.com/images/1NF.png)

[Table 1: In 1NF Form](https://www.guru99.com/images/1NF.png)

Before we proceed let's understand a few things --

**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.

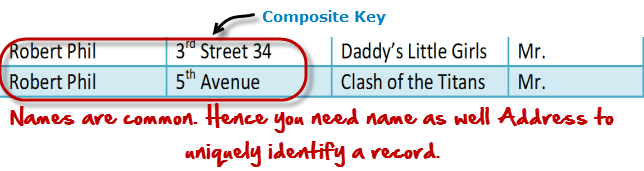
What is a Primary Key?

|  |  |
| --- | --- |
| [What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples](https://www.guru99.com/images/PrimaryKey.png) | 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 cannot be changed * The primary key must be given a value when a new record is inserted. |

**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.

[](https://www.guru99.com/images/CompositeKey.png)

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

Let's move into second normal form 2NF

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

[[](https://www.guru99.com/images/Table2.png)](https://www.guru99.com/images/Table2.png)

[Table 1](https://www.guru99.com/images/Table2.png)

[[](https://www.guru99.com/images/Table1.png)](https://www.guru99.com/images/Table1.png)

[Table 2](https://www.guru99.com/images/Table1.png)

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

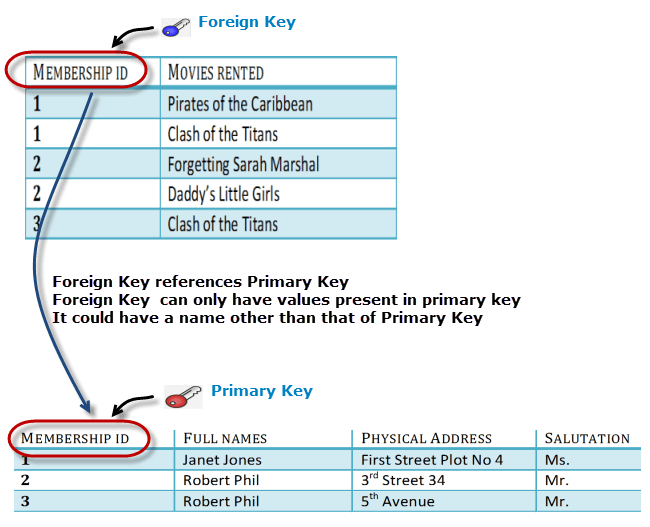
**Database - Foreign Key**

In Table 2, Membership\_ID is the Foreign Key

|  |  |
| --- | --- |
| [What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples](https://www.guru99.com/images/ForeignKey.png) |  |

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

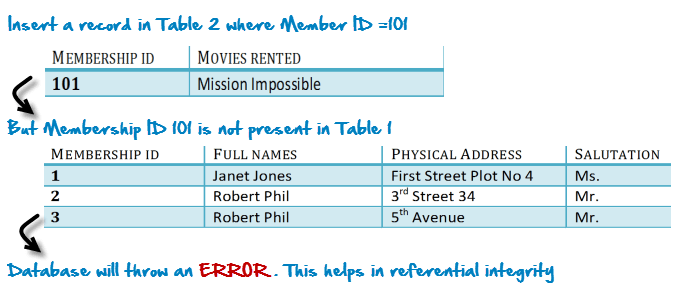
* A foreign key can have a different name from its primary key
* It ensures rows in one table have corresponding rows in another
* Unlike the Primary key, they do not have to be unique. Most often they aren't
* Foreign keys can be null even though primary keys can not

[](https://www.guru99.com/images/ForeignKeyRelationWithPrimary.png)

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.

[](https://www.guru99.com/images/WhyDataBaseIsImportant.png)

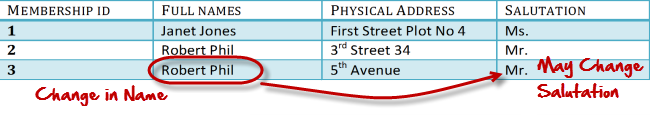
The above problem can be overcome by declaring membership id  from Table2  as foreign key of membership id from Table1

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!

**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.

[](https://www.guru99.com/images/transitive_functional_dependencies.png)

Let's move into 3NF

**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.

**3NF Example**

[[](https://www.guru99.com/images/2NFTable1.png)](https://www.guru99.com/images/2NFTable1.png)

[TABLE 1](https://www.guru99.com/images/2NFTable1.png)

[[](https://www.guru99.com/images/2NFTable2.png)](https://www.guru99.com/images/2NFTable2.png)

[Table 2](https://www.guru99.com/images/2NFTable2.png)

[[](https://www.guru99.com/images/2NFTable3.png)](https://www.guru99.com/images/2NFTable3.png)

[Table 3](https://www.guru99.com/images/2NFTable3.png)

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 at 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 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.

**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.**