1. **FIRST NORMAL FORM**

It is a property of a relation in a relational database wherein only when the domain of each attribute has only atomic values (values that cannot be divided or simplified further) and the value of each attribute has only one value from the selected domain. Edgar Codd, an English Computer Scientist, stated that a relation is said to be in the first normal form when none of its domains have any sets as elements. This form is a very important relation in [RDBMS](http://whatisdbms.com/explain-relational-database-management-system-rdbms/). It enforces several criteria including: 1) Removing repeating groups in individual tables, 2) Creating separate tables for every set of related data and 3) Identifying related data using the primary key of a given set.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | ADDRESS | MOBILE |
| 101 | ABC | JAIPUR | 9832178564  9908764569 |
| 102 | XYZ | KANPUR | 9876543235 |
| 103 | LMN | JAIPUR | 8897645323  7789654467 |
| 104 | QRS | DELHI | 9876543223 |
| 105 | TUV | DELHI | 9876543436 |

Two employees (ABC & LMN) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF**as the rule says “each attribute of a table must have atomic (single) values”, the mobile values for employees ABC & LMN violates that rule.

To make the table complies with 1NF we should have the data like this:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | ADDRESS | MOBILE |
| 101 | ABC | JAIPUR | 9832178564 |
| 101 | ABC | JAIPUR | 9908764569 |
| 102 | XYZ | KANPUR | 9876543235 |
| 103 | LMN | JAIPUR | 8897645323 |
| 103 | LMN | JAIPUR | 7789654467 |
| 104 | QRS | DELHI | 9876543223 |
| 105 | TUV | DELHI | 9876543436 |

## Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

* Table is in 1NF (First normal form)
* No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

**Example**: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

|  |  |  |
| --- | --- | --- |
| teacher\_id | subject | teacher\_age |
| 101 | MATHS | 34 |
| 101 | SCI | 34 |
| 103 | SST | 33 |
| 104 | HINDI | 35 |

**Candidate Keys**: {teacher\_id, subject}  
**Non prime attribute**: teacher\_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher\_age is dependent on teacher\_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:  
**teacher\_details table:**

|  |  |
| --- | --- |
| teacher\_id | teacher\_age |
| 101 | 34 |
| 103 | 33 |
| 104 | 35 |

**teacher\_subject table:**

|  |  |
| --- | --- |
| teacher\_id | subject |
| 101 | MATHS |
| 101 | SCI |
| 103 | SST |
| 104 | HINDI |

## Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

* Table must be in 2NF
* [Transitive functional dependency](https://beginnersbook.com/2015/04/transitive-dependency-in-dbms/) of non-prime attribute on any super key should be removed.

An attribute that is not part of any [candidate key](https://beginnersbook.com/2015/04/candidate-key-in-dbms/) is known as non-prime attribute.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_zip | emp\_state | emp\_city | emp\_district |
| 1001 | John | 282005 | UP | Agra | Dayal Bagh |
| 1002 | Ajeet | 222008 | TN | Chennai | M-City |
| 1006 | Lora | 282007 | TN | Chennai | Urrapakkam |
| 1101 | Lilly | 292008 | UK | Pauri | Bhagwan |
| 1201 | Steve | 222999 | MP | Gwalior | Ratan |

**Super keys**: {emp\_id}, {emp\_id, emp\_name}, {emp\_id, emp\_name, emp\_zip}…so on  
**Candidate Keys**: {emp\_id}  
**Non-prime attributes**: all attributes except emp\_id are non-prime as they are not part of any candidate keys.

Here, emp\_state, emp\_city & emp\_district dependent on emp\_zip. And, emp\_zip is dependent on emp\_id that makes non-prime attributes (emp\_state, emp\_city & emp\_district) transitively dependent on super key (emp\_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

**employee table:**

|  |  |  |
| --- | --- | --- |
| emp\_id | emp\_name | emp\_zip |
| 1001 | John | 282005 |
| 1002 | Ajeet | 222008 |
| 1006 | Lora | 282007 |
| 1101 | Lilly | 292008 |
| 1201 | Steve | 222999 |

**employee\_zip table:**

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_zip | emp\_state | emp\_city | emp\_district |
| 282005 | UP | Agra | Dayal Bagh |
| 222008 | TN | Chennai | M-City |
| 282007 | TN | Chennai | Urrapakkam |
| 292008 | UK | Pauri | Bhagwan |
| 222999 | MP | Gwalior | Ratan |