# **A novice’s guide to normalization**

**What is normalization?**

Normalization is the process of organising data in a database. It is a systematic approach which includes decomposition of tables and establishing relationship between tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

**The aim of normalization is to overcome the anomalies (insertion, deletion, updation) and to reduce the redundancy.**

**Why is normalization so important in a database?**

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, then the data must be changed in exactly the same way in all locations. This is a tedious yet computationally heavy task, hence it is better to reduce redundancy.

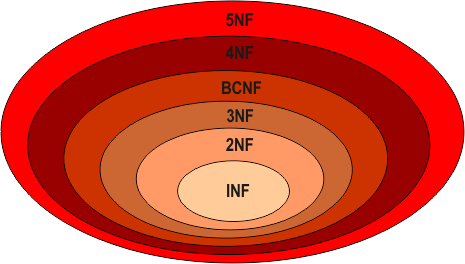
Inconsistent dependency refers to the difficulty of getting relevant data due to a missing or broken path to the data. It leads users to search the data in the wrong table, resulting in an error as output. This occurs due to redundant terms which leads to maintenance problems and affects the ACID properties as well.

There are a few rules for database normalization. Each rule is called a ‘Normal Form’. If the first rule is observed, the database is said to be in ‘First Normal Form’.

If the first three rules are observed, the database is considered in ‘Third Normal Form’. Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

**Types of Normal Forms:**

* 1NF
* 2NF
* 3NF
* BCNF



**First Normal Form(1NF)**

The main criteria for the first normal form are ‘Do not use multiple fields in a single table to store similar data’. This is achieved by

* Eliminating repeating groups in individual tables.
* Create a separate table for each set of related data.
* Identify each set of related data with a primary key.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_ID | Student\_Name | Course | Course\_code | Contact |
| 21BIF | Sam | Sociology, Ethics | SOC, ETH | 97697644221  Multi-valued attributes |
| 22BIF | Ira | Nutrition | NUTRI | 98762213444, 92217556181 |
| 23BIF | Tina | Drama, German | DR, GER | 98222345455 |
| 24BIF | Rahul | Political Science | PS | 87643212134, 76431212134 |

Single-valued attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_ID | Student\_Name | Course\_code | Course | Contact |
| 21BIF | Sam | SOC | Sociology | 97697644221 |
| 21BIF | Sam | ETH | Ethics | 97697644221 |
| 22BIF | Ira | NUTRI | Nutrition | 98762213444 |
| 22BIF | Ira | NUTRI | Nutrition | 92217556181 |
| 23BIF | Tina | DR | Drama | 98222345455 |
| 23BIF | Tina | GER | German | 98222345455 |
| 24BIF | Rahul | PS | Political Science | 87643212134 |
| 24BIF | Rahul | PS | Political Science | 76431212134 |

**Second Normal Form(2NF)**

To be in second normal form, a table must be in first normal form and the table must not contain any **partial dependency**. It envisages how each non-key attribute must depend on the entire primary key.

In the table below which is in first normal form, each column has a single-valued attribute but there are many duplicate terms. This is not favourable in a database and this issue can be solved applying second normal form.

Non-Key Attributes

Key Attributes

|  |  |  |
| --- | --- | --- |
| Student\_ID | Honours\_subject | Subject\_code |
| 21BIF | Chemistry | CHE |
| 22BIF | Mathematics | MATH |
| 23BIF | Zoology | ZOO |
| 24BIF | Computer Science | CS |
| 25BIF | Botany | BOT |

In the table above, non-key attribute Subject\_code is dependent on Honours\_subject which is a proper subset of candidate key. This is an example of partial dependency.

To convert the table into 2NF, we decompose it into two tables.

|  |  |
| --- | --- |
| Student\_ID | Honours\_subject |
| 21BIF | Chemistry |
| 22BIF | Mathematics |
| 23BIF | Zoology |
| 24BIF | Computer Science |
| 25BIF | Botany |

|  |  |
| --- | --- |
| Honours\_subject | Subject\_code |
| Chemistry | **CHE** |
| Mathematics | **MATH** |
| Zoology | **ZOO** |
| Computer Science | **CS** |
| Botany | **BOT** |

**Third Normal Form (3NF)**

In the third normal form, every non-key attribute in a table should depend on the key, the whole key and nothing but the key. It tries to eliminate the dependency of a non-key attribute on another non-key attribute, i.e., transitive dependency.

The database must be in First and Second Normal Form, in order for it to become worthy of third normal form.

|  |  |  |  |
| --- | --- | --- | --- |
| Tournament Name | Winner | Man of the match | MOTM\_country |
| IPL2022 | Gujarat Titans | Umran Malik | India |
| IPL2021 | Chennai Super Kings | Faf du Plessis | South Africa |
| IPL2020 | Mumbai Indians | Trent Boult | New Zealand |
| IPL2019 | Mumbai Indians | Jasprit Bumrah | India |
| IPL2018 | Chennai Super Kings | Shane Watson | Australia |
| IPL 2017 | Rising Pune Supergiants | Krunal Pandya | India |

|  |  |
| --- | --- |
| Man of the match | MOTM\_country |
| Umran Malik | India |
| Faf du Plesis | South Africa |
| Trent Boult | New Zealand |
| Jasprit Bumrah | India |
| Shane Watson | Australia |
| Krunal Pandya | India |

|  |  |  |
| --- | --- | --- |
| Tournament Name | Winner | Man of the match |
| IPL2022 | Gujarat Titans | Umran Malik |
| IPL2021 | Chennai Super Kings | Faf du Plessis |
| IPL2020 | Mumbai Indians | Trent Boult |
| IPL2019 | Mumbai Indians | Jasprit Bumrah |
| IPL2018 | Chennai Super Kings | Shane Watson |
| IPL 2017 | Rising Pune Supergiants | Krunal Pandya |

Here, the main table is split into two sub-tables because the ‘Man of the match’ is dependent on the winning team and the tournament name. However, MOTM\_country is not dependent on the tournament, it only depends on the man of the match.

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

It is a slightly stronger version of third normal form, there are only minute and subtle differences between the third normal form and Boyce-Codd Normal Form

It says that with the exception of trivial functional dependencies, every functional dependency in a table must be dependent on a super key.

BCNF is free from redundancy, a relation with two attributes is always in BCNF.

|  |  |  |  |
| --- | --- | --- | --- |
| Release\_year | Popularity\_Ranking | Movie\_Name | Release\_year\_and\_month |
| 2008 | 1 | The Dark Knight | 2008-01 |
| 2008 | 2 | Indiana Jones | 2008-05 |
| 2008 | 3 | Kung Fu Panda | 2008-06 |
| 2009 | 1 | Harry Potter | 2009-12 |
| 2009 | 2 | Avatar | 2009-06 |
| 2009 | 3 | Ice Age | 2009-07 |

|  |  |  |  |
| --- | --- | --- | --- |
| Release\_year | Popularity\_Ranking | Movie\_Name | Release\_month |
| 2008 | 1 | The Dark Knight | January |
| 2008 | 2 | Indiana Jones | May |
| 2008 | 3 | Kung Fu Panda | June |
| 2009 | 1 | Harry Potter | December |
| 2009 | 2 | Avatar | June |
| 2009 | 3 | Ice Age | July |

NOTE: There are many more normal forms that exist after BCNF, like 4NF, 5NF and many more, but in real world database systems it’s generally not required to go beyond BCNF.