**Normalization**

If a database design is not perfect, it may contain anomalies, which are bad for any database administrator. Managing a database with anomalies is next to impossible.

Database Normalization is used for following Purpose:

1. To Eliminate the redundant or useless data
2. To Reduce the complexity of the data
3. To Ensure the relationship between tables as well as data in the tables
4. To Ensure data dependencies and data is logically stored.

**First normal form (1NF)**

According to first normal form, an attribute of a table cannot hold multiple values. It should hold only atomic values.

|  |  |  |
| --- | --- | --- |
| teacher\_id | subject | teacher\_age |
| 111 | Maths  Biology | 38 |
| 222 | Biology | 50 |
| 333 | Physics | 40 |

Here teacher with id 111 is teaching two subjects so this table is not in first normal form.

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

|  |  |  |
| --- | --- | --- |
| teacher\_id | Subject | teacher\_age |
| 111 | Maths | 38 |
| 222 | Biology | 50 |
| 333 | Physics | 40 |
| 111 | Biology | 38 |

**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

**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 |
| 111 | Maths | 38 |
| 111 | Physics | 38 |
| 222 | Biology | 38 |
| 333 | Physics | 40 |
| 333 | Chemistry | 40 |

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

**teacher\_details table:**

|  |  |
| --- | --- |
| teacher\_id | teacher\_age |
| 111 | 38 |
| 222 | 38 |
| 333 | 40 |

**teacher\_subject table:**

|  |  |
| --- | --- |
| teacher\_id | Subject |
| 111 | Maths |
| 111 | Physics |
| 222 | Biology |
| 333 | Physics |
| 333 | Chemistry |

**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** of non-prime attribute on any super key should be removed i.e. for each functional dependency X-> Y at least one of the following conditions hold:
* X is a super keyof table
* Y is a prime attribute of table

**Example**: Suppose teacher details that looks like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| teacher\_id | subject | teacher\_zip | state | city | district |
| 111 | Maths | 282005 | UP | Agra | Dayal Bagh |
| 111 | Physics | 222008 | MP | Chennai | M-City |
| 222 | Biology | 282007 | Bihar | Chennai | Urrapakkam |
| 333 | Physics | 292008 | UK | Pauri | Bhagwan |
| 333 | Chemistry | 222999 | TN | Gwalior | Ratan |

**Super keys**: {teacher\_id}, {teacher\_id,subject}, {teacher\_id, subject, teacher\_zip}

**Candidate Keys**: {teacher\_id}

* Here, state, city and district are dependent on teacher\_zip. And, teacher\_zip is dependent on teacher\_id that makes non-prime attributes (state, city ,district) transitively dependent on super key (teacher\_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:

**Teacher subject table:**

|  |  |  |
| --- | --- | --- |
| teacher\_id | subject | teacher\_zip |
| 111 | Maths | 282005 |
| 111 | Physics | 222008 |
| 222 | Biology | 282007 |
| 333 | Physics | 292008 |
| 333 | Chemistry | 222999 |

**Teacher details table:**

|  |  |  |  |
| --- | --- | --- | --- |
| teacher\_id | subject | teacher\_zip | state |
| 111 | Maths | 282005 | UP |
| 111 | Physics | 222008 | MP |
| 222 | Biology | 282007 | Bihar |
| 333 | Physics | 292008 | UK |
| 333 | Chemistry | 222999 | TN |

**Boyce Codd normal form (BCNF)**

* It is an advance version of 3NF that’s why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a teacher that teaches more than one subject. They store the data like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| teacher\_id | nationality | subject | subject\_code | teachers\_teaching\_subject |
| 111 | Indian | Maths | M001 | 20 |
| 111 | Indian | Biology | B001 | 25 |
| 115 | American | Physics | P134 | 10 |
| 167 | American | IT | I122 | 60 |

* **Functional dependencies in the table above**:  
  teacher\_id -> nationality  
  subject -> {subject\_code, teachers\_teaching\_subject}
* **Candidate key**: {teacher\_id, subject}
* The table is not in BCNF as neither teacher\_id nor subject alone are keys.
* To make the table comply with BCNF we can break the table in three tables like this:

**Teacher\_nationality table:**

|  |  |
| --- | --- |
| teacher\_id | nationality |
| 111 | Indian |
| 115 | American |
| 167 | American |

**Teacher\_subject table:**

|  |  |  |
| --- | --- | --- |
| subject | subject\_code | teachers\_teaching\_subject |
| Maths | M001 | 20 |
| Biology | B001 | 25 |
| Physics | P134 | 10 |
| IT | I122 | 60 |

**Teacher\_subject\_mapping table:**

|  |  |
| --- | --- |
| teacher\_id | subject |
| 111 | Maths |
| 111 | Biology |
| 115 | Physics |
| 167 | IT |

* **Functional dependencies**:  
  teacher\_id -> nationality  
  subject -> {subject\_code, teachers\_teaching\_subject}

**Candidate keys**:  
For first table: teacher\_id  
For second table: subject  
For third table: {teacher\_id, subject}