**Normalization**

Here are the most commonly used normal forms:

* First normal form(1NF)
* Second normal form(2NF)
* Third normal form(3NF)
* Boyce & Codd normal form (BCNF)

**Boyce & Codd normal form (BCNF)**

Boyce Codd normal form (BCNF) BCNF is the advance version of 3NF. It is stricter than 3NF. A table is in BCNF if every functional dependency X → Y, X is the super key of the table.

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.

**Rules for BCNF**

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

1. It should be in the Third Normal Form
2. And for any dependency A → B, A should be a super key. In simple words, it means, that for a dependency A → B, A cannot be a non-prime attribute, if B is a prime attribute.

**Candidate Keys and BCNF**

* A candidate key is a column or combination of columns in a table that forms a unique key in the database.
* The combination of attributes can be used to identify a database record without referring to any other data.
* Each table can contain multiple candidate keys, any one of which can qualify as the primary key. A table contains only one primary key.
* Candidate keys must be unique.
* A relation is in BCNF if every determinant is a candidate key.

Consider a database table that stores employee information and has the attributes *<Employee\_Id*>, *<First\_name>*, *<Last\_name>*, and *<Title>*.

* In this table, the field *<Employee\_Id>* determines *first\_name* and *last\_name*. Similarly, the tuple (*<First\_name>*, *<Last\_name>*) determines *<Employee\_id>*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee Id** | **First name** | **Last name** | **Title** |
| 13133 | Emily | Smith | Manager |
| 13134 | Jim | Smith | Associate |
| 13135 | Emily | Jones | Associate |

**BCNF Example**

Suppose there is a company wherein employees work in more than one department. They store the data like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| emp\_id | emp\_nationality | emp\_dept | dept\_type | dept\_no\_of\_emp |
| 1001 | Austrian | Production and planning | D001 | 200 |
| 1001 | Austrian | stores | D001 | 250 |
| 1002 | American | design and technical support | D134 | 100 |
| 1002 | American | Purchasing department | D134 | 600 |

Functional dependencies in the table above:

emp\_id -> emp\_nationality

emp\_dept -> {dept\_type, dept\_no\_of\_emp}

Candidate key: {emp\_id, emp\_dept}

The table is not in BCNF as neither emp\_id nor emp\_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

**emp\_nationality table:**

|  |  |
| --- | --- |
| **emp\_id** | **emp\_nationality** |
| 1001 | Austrian |
| 1002 | American |

**emp\_dept table:**

|  |  |  |
| --- | --- | --- |
| **emp\_dept** | **dept\_type** | **dept\_no\_of\_emp** |
| Production and planning | D001 | 200 |
| stores | D001 | 250 |
| design and technical support | D134 | 100 |
| Purchasing department | D134 | 600 |

**emp\_dept\_mapping table:**

|  |  |
| --- | --- |
| **emp\_id** | **emp\_dept** |
| 1001 | Production and planning |
| 1001 | stores |
| 1002 | design and technical support |
| 1002 | Purchasing department |

Functional dependencies:

emp\_id -> emp\_nationality

emp\_dept -> {dept\_type, dept\_no\_of\_emp}

Candidate keys:

For first table: emp\_id

For second table: emp\_dept

For third table: {emp\_id, emp\_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.