

Visit: [Whatisdbms.com](http://www.whatisdbms.com) for more Articles

Database Normalization: Explain 1NF, 2NF, 3NF, BCNF With Examples

The purpose of normalization is to make the life of users easier and also to save space on computers while storing huge amounts of data. The added advantage of getting an organized package of data that helps in a performance boost is also a very notable use of normalization. This discussion is all about Database Normalization: Explain 1NF, 2NF, 3NF, BCNF With Examples. It can be mainly classified into 4 types:

- 1) 1st Normal Form.
- 2) 2nd Normal Form.
- 3) 3rd Normal Form.
- 4) 4th Normal Form.
- 5) 5th Normal Form, and
- 6) Boyce- Codd Normal Form.

The discussion here includes the 1st, 2nd, 3rd and 4th Normal Forms.

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. 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.

Example:

Consider a table containing the details of a company. The fields to be included are employee name, employee ID no, employee location and employee contact no. For better understanding, this will be displayed in a table form.

EMPLOYEE NAME	EMPLOYEE ID	EMPLOYEE LOCATION	EMPLOYEE CONTACT NO
AJAY	12455	DELHI	987565,356212
AMIT	89752	AGRA	455145,988965
AKSHAR	23654	HARYANA	987454
ALOKE	98765	KOCHI	124787

Table 1.1

In the above table, we can see the employee details of a certain company. All rows have values as shown but we can see that the values in the first two rows under the column 'EMPLOYEE CONTACT NO' have multiple values. Employees AJAY and AMIT have multiple contact numbers which cannot be accepted in the first normal form. It brings ambiguity to the database and can generate anomalies. Hence the need arises to maintain the uniqueness of the field. In order to bring it to the first normal form, one of the values from the field of employee contact no should be removed (from both Ajay and Amit's data). So the correct first normal form will be obtained upon editing in such a manner. The correct table will be:

EMPLOYEE NAME	EMPLOYEE ID	EMPLOYEE LOCATION	EMPLOYEE CONTACT NO
AJAY	12455	DELHI	987565
AMIT	89752	AGRA	455145
AKSHAR	23654	HARYANA	987454
ALOKE	98765	KOCHI	124787

Table 1.2

The correct table complies with the first normal form criteria i.e., "each attribute of a table must have atomic values". The extra contact numbers were removed to obtain the required form design.

WHAT IS ATOMICITY IN DATABASE?

A definition of first normal form makes reference to the concept of 'atomicity'. It states that the domain should have values in the relation which are impossible to be broken down into smaller contents of data, with respect to DBMS. Codd defines an atomic value as one that "cannot be decomposed into smaller pieces.

2. SECOND NORMAL FORM (2NF)

An entity is said to be in the second normal form when it is already in 1NF and all the attributes contained within it are dependent solely on the unique identifier of the entity. In other words, it maintains two important criteria to be met in order to provide a normalized data with the second normal form tag.

- 1) It is in the first normal form
- 2) All non-key attributes are fully functional and dependent on the primary key.

To give more clarity to the statements said above, consider a table and two attributes within the table, A and B. Suppose attribute B is functionally dependent on A, but is not on a proper subset of A. Then B can be considered to be fully functional and dependent on A. Therefore in a 2NF table, all of the non-key attributes cannot be dependent on the primary key's subset. A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

Example:

Consider a toy shop that has three branches in three different locations. A table is prepared indicating the customer IDs, store IDs and store location.

CUSTOMER ID	STORE ID	STORE LOCATION
1	112	BANGALORE
2	234	MUMBAI
3	323	DELHI

Table 2.1

The above table is a composite one and has a composite primary key (CUSTOMER ID, STORE ID). The non-key attribute in this arrangement is STORE LOCATION. In the above case, the STORE LOCATION only depends on the STORE ID, which is the sole part of the primary key. Hence the table does not satisfy the second normal form.

CUSTOMER ID	STORE ID
1	112
2	234
3	323

TABLE 2.2

STORE ID	STORE LOCATION
112	BANGALORE
234	MUMBAI
323	DELHI

Table 2.3

To resolve this issue and to convert the entity into the 2NF, the table is split into two separate tables. By splitting the table, the partial functional dependency is removed and atomicity is achieved for both the tables (thus realizing 1NF in the process). Now, the column STORE LOCATION is completely dependent on the primary key, the STORE ID thereby achieving 2NF for the table under consideration.

3. THIRD NORMAL FORM

An entity is said to be in the third normal form when,

- 1) It satisfies the criteria to be in the second normal form.
- 2) There exists no transitive functional dependency. (Transitive functional dependency can be best explained with the relationship link between three tables. If table A is functionally dependent on B, and B is functionally dependent on C then C is transitively dependent on A and B). It can also be said that the transitive functional dependency of non-prime attribute on any super key is removed. (A super key is a combined column system that is used to uniquely identify a row within any RDBMS. A super key is reduced to a minimum no of columns required to uniquely identify each row.)

3NF states that every column reference in referenced data which are not dependent on the primary key should be removed or that only foreign key columns should be used to reference another table, and no other columns from the parent table should exist in the referenced table.

Example:

Consider a table that shows the database of a bookstore. The table consists of details on Book ID, Genre ID, Book Genre and Price. The database is maintained to keep a record of all the books that are available or will be available in the bookstore. The table of data is given below.

BOOK ID	GENRE ID	BOOK GENRE	COST
121	2	FICTION	150
233	3	TRAVEL	100
432	4	SPORTS	120
123	2	FICTION	185
424	3	TRAVEL	140

Table 3.1

The data in the table provides us with an idea of the books offered in the store. Also, we can deduce that BOOK ID determines the GENRE ID and the GENRE ID determines the BOOK GENRE. Hence we can see that a transitive functional dependency has developed which makes certain that the table does not satisfy the third normal form.

TABLE_BOOK

BOOK ID	GENRE ID	PRICE
121	2	150
233	3	100
432	4	120
123	2	185
424	3	140

TABLE_GENRE

GENRE ID	BOOK GENRE
2	FICTION
3	TRAVEL
4	SPORTS

After splitting the tables and regrouping the redundant content, we obtain two tables where all non-key attributes are fully functional dependent only on the primary key.

To further explain the advanced step of the normalization process, we are required to understand the Boyce-Codd Normal Form and its comparison with the third normal form.

THE BOYCE-CODD NORMAL FORM AND RELATION WITH 3NF

The Boyce-Codd Normal Form or BCNF or 3.5 NF is a normal form which is slightly stronger than the 3NF. It was developed in 1974 to address certain types of anomalies that were not dealt by 3NF. A relational scheme, once prepared in BCNF, will remove all sorts of functional dependency (though some other forms of redundancy can prevail). Coming to the relation between BCNF AND 3NF, in certain rare cases, the 3NF table fails to meet the requirements of the BCNF. A 3NF table sans multiple overlapping candidate keys is guaranteed to be in BCNF and depending on the functional dependencies of the entity, a 3NF table that possesses two or more overlapping candidate keys may/may not be capable of being in BCNF.

Here is an example of a 3NF table that does not meet the demands of the BCNF:

The table shows the details of the booking schedule for a swimming pool complex with the fields of POOL NO, START TIME, END TIME and TYPE OF MEMBERSHIP. The details are filled in the rows and columns of the table below:

POOL NO	START TIME	END TIME	TYPE OF MEMBERSHIP
1	09:30	10:30	REGULAR
1	11:15	12:00	REGULAR
1	13:00	14:30	PREMIUM
2	10:00	11:30	EXECUTIVE B
2	11:30	12:30	EXECUTIVE B
2	15:00	16:00	EXECUTIVE A

In the above table, no non-prime attributes exist which means that all attributes belong to some candidate key. This justifies the table being of 2NF and 3NF. However, the table does not follow BCNF because of the dependency of the type of membership in which the determining attribute, type of membership on which pool no: depends is neither a candidate key nor a superset of a candidate key. The design needs to be modified in order to conform to the BCNF. The significance of explaining the BCNF comes when the step of normalization is to be explained. The 'Fourth Normal Form' or 4NF

4. FOURTH NORMAL FORM

A normal form that is used in database normalization. The 4NF came at a significant time period as the next level of normalization. It was introduced by Ronald Fagin in 1977, after the Boyce-Codd Normal Form. The 4NF is basically concerned with a more general type of dependency known as a multivalued dependency and is different from 2NF, 3NF and BCNF and their functional dependencies. A table can be considered to be in 4NF only if for every one of the non-trivial multivalued dependencies, $X \twoheadrightarrow Y$, when X is a super key. (This means that X is either a candidate key or a superset).

Example:

Consider the table given below:

COURSE ID	INSTRUCTOR	TEXTBOOK
BTEYO1	JAMES	R R MARK
BTEYO1	JAMES	L JAMES
BTEYO1	ASHLEY	R R MARK
BTEYO1	ASHLEY	L JAMES

Here COURSE ID and INSTRUCTOR are multivalued dependent attributes. They can be converted to 4NF by separating the single table into two tables which are as given below

COURSE-INST

COURSE ID	INSTRUCTOR
BTEYO1	JAMES
BTEYO1	ASHLEY

COURSE-TEXT

COURSE ID	TEXTBOOK
BTEYO1	R R MARK
BTEYO1	L JAMES

USE OF 4NF

At the higher levels of normalization, the teaching and use of database normalization slows down substantially mostly because most of the tables are in direct violation of the 4NF. A table that satisfies 4NF is hard to come by most of the business applications. A study proves that, as of now, over 20% of business processes contains tables that violates 4NF but successfully meets all lower forms of normalization.

BEYOND 4NF

At 4NF, the performance reduces considerably and a further 5NF procedure may not be feasible as it causes great chances of error and very few tables practically satisfy the criteria to be of 5NF. The 5NF is also called the project-join normal form and is the highest level of normalization designed to reduce redundancy in relational databases which is done by recording multi-valued facts by isolating semantically related multiple relationships.

So it was all about Database Normalization: Explain 1NF, 2NF, 3NF, BCNF With Examples. If you have any doubt then please comment below.

www.whatisdbms.com