**Normal Forms in DBMS**

**Normalization** is the process of minimizing **redundancy** from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. **Normal forms** are used to eliminate or reduce redundancy in database tables.

**Normalization of DBMS**

In database management systems (DBMS), normal forms are a series of guidelines that help to ensure that the design of a database is efficient, organized, and free from data anomalies. There are several levels of normalization, each with its own set of guidelines, known as normal forms.

**Important Points Regarding Normal Forms in DBMS**

* **First Normal Form (1NF):** This is the most basic level of normalization. In 1NF, each table cell should contain only a single value, and each column should have a unique name. The first normal form helps to eliminate duplicate data and simplify queries.
* **Second Normal Form (2NF):** 2NF eliminates redundant data by requiring that each non-key attribute be dependent on the primary key. This means that each column should be directly related to the primary key, and not to other columns.
* **Third Normal Form (3NF):** 3NF builds on 2NF by requiring that all non-key attributes are independent of each other. This means that each column should be directly related to the primary key, and not to any other columns in the same table.
* **Boyce-Codd Normal Form (BCNF):** BCNF is a stricter form of 3NF that ensures that each determinant in a table is a candidate key. In other words, BCNF ensures that each non-key attribute is dependent only on the candidate key.
* **Fourth Normal Form (4NF):** 4NF is a further refinement of BCNF that ensures that a table does not contain any multi-valued dependencies.
* **Fifth Normal Form (5NF):** 5NF is the highest level of normalization and involves decomposing a table into smaller tables to remove data redundancy and improve data integrity.

Normal forms help to reduce data redundancy, increase data consistency, and improve database performance. However, higher levels of normalization can lead to more complex database designs and queries. It is important to strike a balance between normalization and practicality when designing a database.

**Advantages of Normal Form**

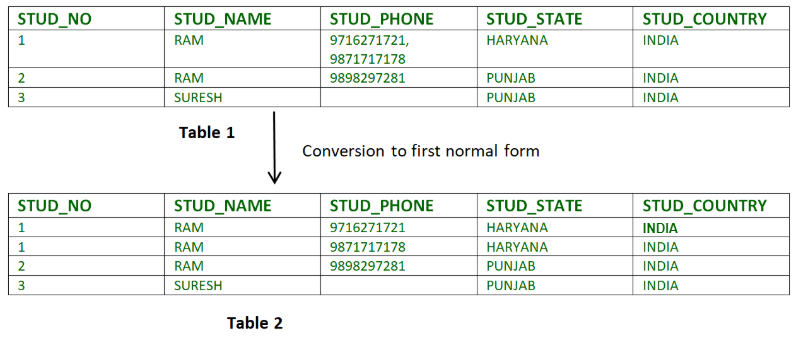
* **Reduced data redundancy:** Normalization helps to eliminate duplicate data in tables, reducing the amount of storage space needed and improving database efficiency.
* **Improved data consistency:** Normalization ensures that data is stored in a consistent and organized manner, reducing the risk of data inconsistencies and errors.
* **Simplified database design:** Normalization provides guidelines for organizing tables and data relationships, making it easier to design and maintain a database.
* **Improved query performance:**Normalized tables are typically easier to search and retrieve data from, resulting in faster query performance.
* **Easier database maintenance:** Normalization reduces the complexity of a database by breaking it down into smaller, more manageable tables, making it easier to add, modify, and delete data.

Overall, using normal forms in DBMS helps to improve data quality, increase database efficiency, and simplify database design and maintenance.

**First Normal Form**

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

* **Example 1 –** Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD\_PHONE. Its decomposition into 1NF has been shown in table 2.



*Example*

* **Example 2 –**

ID Name Courses  
------------------  
1 A c1, c2  
2 E c3  
3 M C2, c3

* In the above table Course is a multi-valued attribute so it is not in 1NF. Below Table is in 1NF as there is no multi-valued attribute

ID Name Course  
------------------  
1 A c1  
1 A c2  
2 E c3  
3 M c2  
3 M c3

**Second Normal Form**

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has **No Partial Dependency,**i.e.**,**no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table. **Partial Dependency –** If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

* **Example 1 –** Consider table-3 as following below.

STUD\_NO COURSE\_NO COURSE\_FEE  
1 C1 1000  
2 C2 1500  
1 C4 2000  
4 C3 1000  
4 C1 1000  
2 C5 2000

* {Note that, there are many courses having the same course fee} Here, COURSE\_FEE cannot alone decide the value of COURSE\_NO or STUD\_NO; COURSE\_FEE together with STUD\_NO cannot decide the value of COURSE\_NO; COURSE\_FEE together with COURSE\_NO cannot decide the value of STUD\_NO; Hence, COURSE\_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD\_NO, COURSE\_NO} ; But, COURSE\_NO -> COURSE\_FEE, i.e., COURSE\_FEE is dependent on COURSE\_NO, which is a proper subset of the candidate key. Non-prime attribute COURSE\_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF. To convert the above relation to 2NF, we need to split the table into two tables such as : Table 1: STUD\_NO, COURSE\_NO Table 2: COURSE\_NO, COURSE\_FEE

**Table 1** **Table 2**  
STUD\_NO COURSE\_NO COURSE\_NO COURSE\_FEE   
1 C1 C1 1000  
2 C2 C2 1500  
1 C4 C3 1000  
4 C3 C4 2000  
4 C1 C5 2000

* **NOTE:** 2NF tries to reduce the redundant data getting stored in memory. For instance, if there are 100 students taking C1 course, we don’t need to store its Fee as 1000 for all the 100 records, instead, once we can store it in the second table as the course fee for C1 is 1000.
* **Example 2 –** Consider following functional dependencies in relation  R (A,  B , C,  D )

AB -> C [A and B together determine C]  
BC -> D [B and C together determine D]

In the above relation, AB is the only candidate key and there is no partial dependency, i.e., any proper subset of AB doesn’t determine any non-prime attribute.

X is a super key.  
Y is a prime attribute (each element of Y is part of some candidate key).

**Example 1:**In relation STUDENT given in Table 4, FD set: {STUD\_NO -> STUD\_NAME, STUD\_NO -> STUD\_STATE, STUD\_STATE -> STUD\_COUNTRY, STUD\_NO -> STUD\_AGE}

Candidate Key: {STUD\_NO}

For this relation in table 4, STUD\_NO -> STUD\_STATE and STUD\_STATE -> STUD\_COUNTRY are true.

So STUD\_COUNTRY is transitively dependent on STUD\_NO. It violates the third normal form.

To convert it in third normal form, we will decompose the relation STUDENT (STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_COUNTRY\_STUD\_AGE) as: STUDENT (STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_AGE) STATE\_COUNTRY (STATE, COUNTRY)

Consider relation R(A, B, C, D, E) A -> BC, CD -> E, B -> D, E -> A All possible candidate keys in above relation are {A, E, CD, BC} All attributes are on right sides of all functional dependencies are prime.

**Example 2:** Find the highest normal form of a relation R(A,B,C,D,E) with FD set as {BC->D, AC->BE, B->E}

**Step 1:**As we can see, (AC)+ ={A,C,B,E,D} but none of its subset can determine all attribute of relation, So AC will be candidate key. A or C can’t be derived from any other attribute of the relation, so there will be only 1 candidate key {AC}.

**Step 2:** Prime attributes are those attributes that are part of candidate key {A, C} in this example and others will be non-prime {B, D, E} in this example.

**Step 3:** The relation R is in 1st normal form as a relational DBMS does not allow multi-valued or composite attribute. The relation is in 2nd normal form because BC->D is in 2nd normal form (BC is not a proper subset of candidate key AC) and AC->BE is in 2nd normal form (AC is candidate key) and B->E is in 2nd normal form (B is not a proper subset of candidate key AC).

The relation is not in 3rd normal form because in BC->D (neither BC is a super key nor D is a prime attribute) and in B->E (neither B is a super key nor E is a prime attribute) but to satisfy 3rd normal for, either LHS of an FD should be super key or RHS should be prime attribute. So the highest normal form of relation will be 2nd Normal form.

For example consider relation R(A, B, C) A -> BC, B -> A and B both are super keys so above relation is in BCNF.

**Third Normal Form**

A relation is said to be in third normal form, if we did not have any transitive dependency for non-prime attributes. The basic condition with the Third Normal Form is that, the relation must be in Second Normal Form.

Below mentioned is the basic condition that must be hold in the non-trivial functional dependency X -> Y:

* X is a Super Key.
* Y is a Prime Attribute ( this means that element of Y is some part of Candidate Key).

For more, refer to [Third Normal Form in DBMS.](https://www.geeksforgeeks.org/third-normal-form-3nf/)

**BCNF**

BCNF (Boyce-Codd Normal Form) is just a advanced version of Third Normal Form. Here we have some additional rules than Third Normal Form. The basic condition for any relation to be in BCNF is that it must be in Third Normal Form.

We have to focus on some basic rules that are for BCNF:

1. Table must be in Third Normal Form.  
2. In relation X->Y, X must be a superkey in a relation.

For more, refer to [BCNF in DBMS.](https://www.geeksforgeeks.org/boyce-codd-normal-form-bcnf/)

**Fourth Normal Form**

Fourth Normal Form contains no non-trivial multivaued dependency except candidate key. The basic condition with Fourth Normal Form is that the relation must be in BCNF.

The basic rules are mentioned below.

1. It must be in BCNF.  
2. It does not have any multi-valued dependency.

For more, refer to [Fourth Normal Form in DBMS.](https://www.geeksforgeeks.org/introduction-of-4th-and-5th-normal-form-in-dbms/)

**Fifth Normal Form**

Fifth Normal Form is also called as Projected Normal Form. The basic conditions of Fifth Normal Form is mentioned below.

Relation must be in Fourth Normal Form.  
The relation must not be further non loss decomposed.

For more, refer to [Fifth Normal Form in DBMS.](https://www.geeksforgeeks.org/introduction-of-4th-and-5th-normal-form-in-dbms/)

**Applications of Normal Forms in DBMS**

* **Data consistency:** Normal forms ensure that data is consistent and does not contain any redundant information. This helps to prevent inconsistencies and errors in the database.
* **Data redundancy:**Normal forms minimize data redundancy by organizing data into tables that contain only unique data. This reduces the amount of storage space required for the database and makes it easier to manage.
* **Response time:** Normal forms can improve query performance by reducing the number of joins required to retrieve data. This helps to speed up query processing and improve overall system performance.
* **Database maintenance:**Normal forms make it easier to maintain the database by reducing the amount of redundant data that needs to be updated, deleted, or modified. This helps to improve database management and reduce the risk of errors or inconsistencies.
* **Database design:** Normal forms provide guidelines for designing databases that are efficient, flexible, and scalable. This helps to ensure that the database can be easily modified, updated, or expanded as needed.

**Some Important Points about Normal Forms**

* BCNF is free from redundancy caused by Functional Dependencies.
* If a relation is in BCNF, then 3NF is also satisfied.
* If all attributes of relation are prime attribute, then the relation is always in 3NF.
* A relation in a Relational Database is always and at least in 1NF form.
* Every Binary Relation ( a Relation with only 2 attributes ) is always in BCNF.
* If a Relation has only singleton candidate keys( i.e. every candidate key consists of only 1 attribute), then the Relation is always in 2NF( because no Partial functional dependency possible).
* Sometimes going for BCNF form may not preserve functional dependency. In that case go for BCNF only if the lost FD(s) is not required, else normalize till 3NF only.
* There are many more Normal forms that exist after BCNF, like 4NF and more. But in real world database systems it’s generally not required to go beyond BCNF.

**Conclusion**

In Conclusion, relational databases can be arranged according to a set of rules called normal forms in [database](https://www.geeksforgeeks.org/what-is-database/) administration (1NF, 2NF, 3NF, BCNF, 4NF, and 5NF), which reduce data redundancy and preserve data integrity. By resolving various kinds of data anomalies and dependencies, each subsequent normal form expands upon the one that came before it. The particular requirements and properties of the data being stored determine which normal form should be used; higher normal forms offer stricter data integrity but may also result in more complicated database structures.