Normalization in DBMS-

- Reducing the redundancies
- Ensuring the integrity of data through lossless decomposition

Normalization is done through normal forms.

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (BCNF)

First Normal Form-

A given relation is called in First Normal Form (1NF) if each cell of the table contains only an atomic value.

OR

A given relation is called in First Normal Form (1NF) if the attribute of every tuple is either single valued or a null value.

Example-

Student_id	Name	Subjects
100	Akshay	Computer Networks, Designing
101	Aman	Database Management System
102	Anjali	Automata, Compiler Design

Relation is not in 1NF

- This relation can be brought into 1NF.
- This can be done by rewriting the relation such that each cell of the table contains only one value.

Student_id	Name	Subjects
100	Akshay	Computer Networks
100	Akshay	Designing
101	Aman	Database Management System
102	Anjali	Automata
102	Anjali	Compiler Design

This relation is in First Normal Form (1NF).

NOTE-

- By default, every relation is in 1NF.
- This is because formal definition of a relation states that value of all the attributes must be atomic.

Second Normal Form-

A given relation is called in Second Normal Form (2NF) if and only if-

1. Relation already exists in 1NF.
2. No partial dependency exists in the relation.

Partial Dependency

A partial dependency is a dependency where few attributes of the candidate key determines non-prime attribute(s).

OR

A partial dependency is a dependency where a portion of the candidate key or incomplete candidate key determines non-prime attribute(s).

In other words,

$A \rightarrow B$ is called a partial dependency if and only if-

1. A is a proper subset of some candidate key
2. B is a non-prime attribute.

If any one condition fails, then it will not be a partial dependency.

Example-

Consider a relation- R (V , W , X , Y , Z) with functional dependencies-

$$VW \rightarrow XY$$

$$Y \rightarrow V$$

$$WX \rightarrow YZ$$

The possible candidate keys for this relation are-

VW , WX , WY

From here,

- Prime attributes = { V , W , X , Y }
- Non-prime attributes = { Z }

Now, if we observe the given dependencies-

- There is no partial dependency.
- This is because there exists no dependency where incomplete candidate key determines any non-prime attribute.

Thus, we conclude that the given relation is in 2NF.

Third Normal Form-

A given relation is called in Third Normal Form (3NF) if and only if-

1. Relation already exists in 2NF.
2. No transitive dependency exists for non-prime attributes.

Transitive Dependency

$A \rightarrow B$ is called a transitive dependency if and only if-

1. A is not a super key.
2. B is a non-prime attribute.

If any one condition fails, then it is not a transitive dependency.

A relation is called in Third Normal Form (3NF) if and only if-

Any one condition holds for each non-trivial functional dependency $A \rightarrow B$

1. A is a super key
2. B is a prime attribute

Example-

Consider a relation- R (A , B , C , D , E) with functional dependencies-

$$A \rightarrow BC$$

$$CD \rightarrow E$$

$$B \rightarrow D$$

$$E \rightarrow A$$

The possible candidate keys for this relation are-

A , E , CD , BC

From here,

- Prime attributes = { A , B , C , D , E }
- There are no non-prime attributes

Now,

- It is clear that there are no non-prime attributes in the relation.
- In other words, all the attributes of relation are prime attributes.
- Thus, all the attributes on RHS of each functional dependency are prime attributes.

Thus, we conclude that the given relation is in 3NF.

Boyce-Codd Normal Form-

A given relation is called in BCNF if and only if-

1. Relation already exists in 3NF.
2. For each non-trivial functional dependency $A \rightarrow B$, A is a super key of the relation.

Example-

Consider a relation- R (A , B , C) with the functional dependencies-

$$A \rightarrow B$$

$$B \rightarrow C$$

$$C \rightarrow A$$

The possible candidate keys for this relation are-

A , B , C

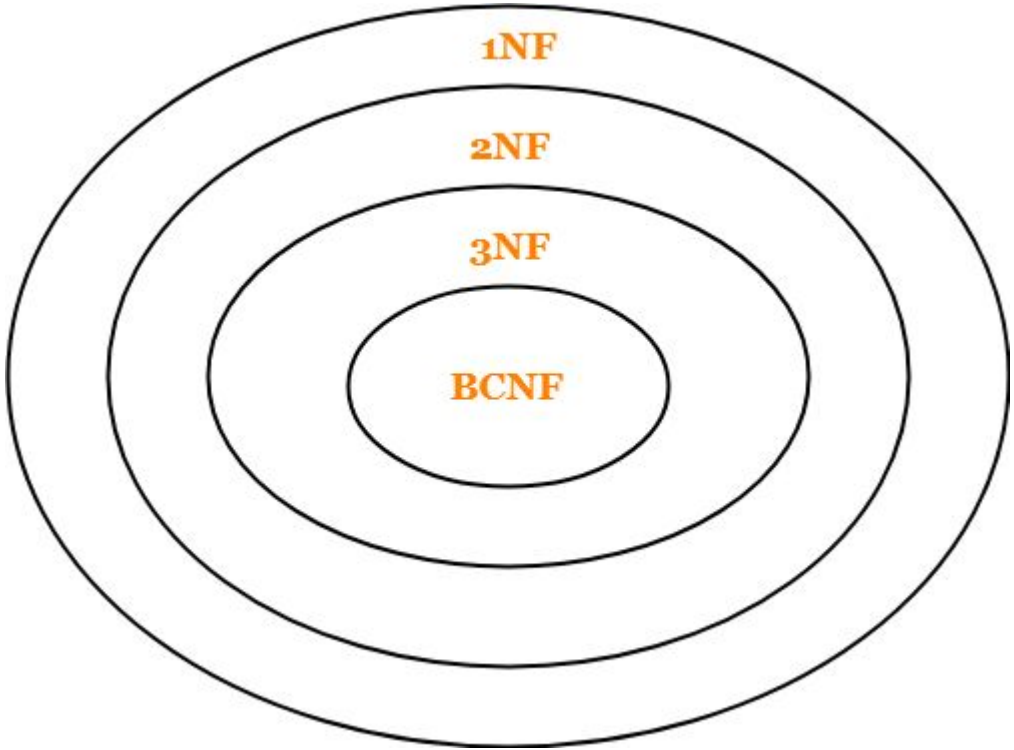
Now, we can observe that RHS of each given functional dependency is a candidate key.

Thus, we conclude that the given relation is in BCNF.

Normal Forms in DBMS-

Remember the following diagram which implies-

- A relation in BCNF will surely be in all other normal forms.
- A relation in 3NF will surely be in 2NF and 1NF.
- A relation in 2NF will surely be in 1NF.

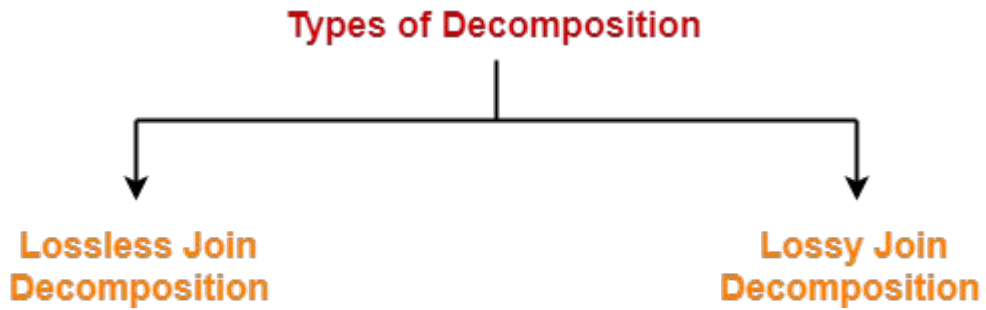

1NF
2NF
3NF
BCNF

Properties of Decomposition-

1. Lossless decomposition- No information is lost from the original relation during decomposition.

Dependency Preservation- None of the functional dependencies that holds on the original relation are lost.

Types of Decomposition-



1. Lossless Join Decomposition-

Consider there is a relation R which is decomposed into sub relations R_1 , R_2 , …. , $R_{n.}$

This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.

For lossless join decomposition, we always have-

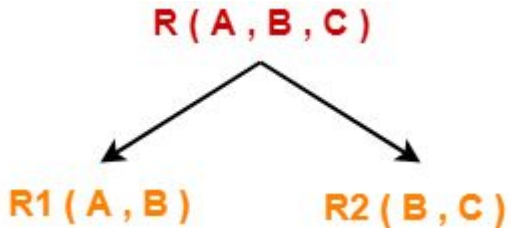
$$R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots \bowtie R_n = R$$

where $\bowtie$ is a natural join operator

Example-

A	B	C
1	2	1
2	5	3
3	3	3

Consider this relation is decomposed into two sub relations R_1(A , B) and R_2(B , C)-



The two sub relations are-

A	B
1	2
2	5
3	3

R_1(A , B)

B	C
2	1
5	3
3	3

R_2(B , C)

Now, let us check whether this decomposition is lossless or not.

For lossless decomposition, we must have-

$$R_1 \bowtie R_2 = R$$

Now, if we perform the natural join ($\bowtie$) of the sub relations R_1 and R_2 , we get-

A	B	C
1	2	1
2	5	3
3	3	3

This relation is same as the original relation R.

Thus, we conclude that the above decomposition is lossless join decomposition.

NOTE-

- Lossless join decomposition is also known as **non-additive join decomposition.**
- This is because the resultant relation after joining the sub relations is same as the decomposed relation.
- No extraneous tuples appear after joining of the sub-relations.

2. Lossy Join Decomposition-

- Consider there is a relation R which is decomposed into sub relations R_1 , R_2 , …. , R_n.
- This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed.
- The natural join of the sub relations is always found to have some extraneous tuples.
- For lossy join decomposition, we always have-

$$R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots \bowtie R_n \supset R$$

where $\bowtie$ is a natural join operator

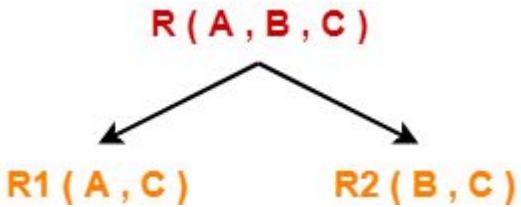
Example-

Consider the following relation R(A , B , C)-

A	B	C
1	2	1
2	5	3
3	3	3

R(A , B , C)

Consider this relation is decomposed into two sub relations as R_1(A , C) and R_2(B , C)-



The two sub relations are-

A	C
1	1
2	3
3	3

R_1(A , B)

B	C
2	1
5	3
3	3

R_2(B , C)

Now, let us check whether this decomposition is lossy or not.

For lossy decomposition, we must have-

$$R_1 \bowtie R_2 \supset R$$

Now, if we perform the natural join ($\bowtie$) of the sub relations R_1 and R_2 we get-

A	B	C
1	2	1
2	5	3
2	3	3
3	5	3
3	3	3

This relation is not same as the original relation R and contains some extraneous tuples.

Clearly, $R_1 \bowtie R_2 \supset R$.

Thus, we conclude that the above decomposition is lossy join decomposition.

Determining Whether Decomposition Is Lossless Or Lossy-

Consider a relation R is decomposed into two sub relations R_1 and R_2.

Then,

- If all the following conditions satisfy, then the decomposition is lossless.
- If any of these conditions fail, then the decomposition is lossy.

Condition-01:

Union of both the sub relations must contain all the attributes that are present in the original relation R.

Thus,

$$R_1 \cup R_2 = R$$

Condition-02:

- Intersection of both the sub relations must not be null.
- In other words, there must be some common attribute which is present in both the sub relations.

Thus,

$$R1 \cap R2 \neq \varnothing$$

Condition-03:

Intersection of both the sub relations must be a super key of either R_1 or R_2 or both.

Thus,

$$R_1 \cap R_2 = \text{Super key of } R_1 \text{ or } R_2$$