

Normalization in DBMS

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

FIRST NORMAL FORM (1NF)

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

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

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

No Non-prime attribute such as COURSE_FEE is dependent on a proper subset of the candidate key i.e. {STUD_NO, COURSE_NO} so no partial dependency and so these relations are in 2NF.

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.

THIRD NORMAL FORM

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency $X \rightarrow Y$

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

Transitive dependency – If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency.

Example:

Consider relation $R(A, B, C, D, E)$

$A \rightarrow BC$,

$CD \rightarrow E$,

$B \rightarrow D$,

$E \rightarrow A$

All possible candidate keys in above relation are $\{A, E, CD, BC\}$ All attributes are on right sides of all functional dependencies are prime.

BOYCE CODD NORMAL FORM

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency $X \rightarrow Y$, X is a super key.

Example:

Relation Student_detail { Stud_id, Stud_Name, City, Zip }

We find that in the above Student_detail relation, Stud_ID is the key and only prime key attribute. We find that City can be identified by Stud_ID as well as Zip

itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, $\text{Stud_ID} \rightarrow \text{Zip} \rightarrow \text{City}$, so there exists transitive dependency.

To bring this relation into third normal form, we break the relation into two relations as follows –

Student_detail { Stud_id, Stud_Name, Zip }

ZipCodes { Zip, City }

In the above relationships Stud_id is the super key in the relation Student_detail and Zip is the Super key in the relation ZipCodes.

So

$\text{Stud_id} \twoheadrightarrow \text{Stud_Name, Zip}$

And

$\text{Zip} \rightarrow \text{City}$

Which confirms that both the relations are in BCNF.

FOURTH NORMAL FORM

A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.

For a dependency $A \twoheadrightarrow B$, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example:

Student { Stud_id, Course, Hobby }

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

Student_course { Stud_id, Course }

Student_Hobby { Stud_id, Hobby }

FIFTH NORMAL FORM

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example:

Relation P={Subject, Lecturer, Semester}

Subject	Lecturer	Semester
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 1
Chemistry	Praveen	Semester 1

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject

as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1= { Semester, Subject }

P2={ Subject, Lecturer }

P3={Semester, Lecturer }

All three relations are now in 5NF.
