

UNIT=3

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 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

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1

Table 2

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C3	1000
C4	2000
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.

Decomposition in DBMS

Decomposition of a Relation-

Definition: The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

Properties of Decomposition-

The following two properties must be followed when decomposing a given relation-

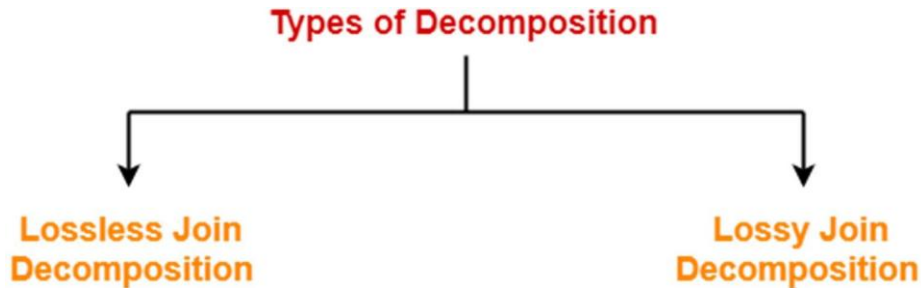
1. Lossless decomposition-

Lossless decomposition ensures-

- No information is lost from the original relation during decomposition.
- When the sub relations are joined back, the same relation is obtained that was decomposed. Every decomposition must always be lossless.

Types of Decomposition-

Decomposition of a relation can be completed in the following two ways-



1. Lossless Join Decomposition-

- Consider there is a relation R which is decomposed into sub relations R₁ , R₂ , , 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-

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

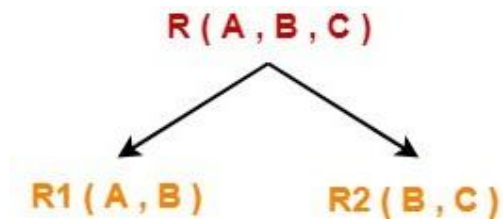
where \bowtie is a natural join operator

Example-

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

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

Consider this relation is decomposed into two sub relations R₁(A , B) and R₂(B , C)-



The two sub relations are-

A	B
1	2
2	5
3	3

B	C
2	1
5	3
3	3

R1(A , B)

R2(B , C)

Now, let us check whether this decomposition is lossless or not. For lossless decomposition, we must have- **$R1 \bowtie R2 = R$**

Now, if we perform the natural join (\bowtie) of the sub relations R1 and R2 , 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 R1 , R2 , , Rn.
- 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-

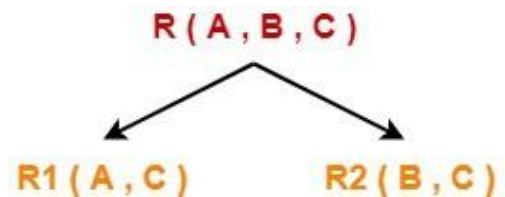
$$R_1 \bowtie R_2 \bowtie R_3 \dots\dots \bowtie R_n \supset R$$

where \bowtie is a natural join operator

Example-

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

Consider this relation is decomposed into two sub relations as R1(A , C) and R2(B , C)-



The two sub relations are-

A	C
1	1
2	3
3	3

R1(A , B)

B	C
2	1
5	3
3	3

R2(B , C)

Now, let us check whether this decomposition is lossy or not. For lossy decomposition, we must have-

$$R1 \bowtie R2 \supset R$$

Now, if we perform the natural join (\bowtie) of the sub relations R1 and R2 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, $R1 \bowtie R2 \supset R$. Thus, we conclude that the above decomposition is lossy join decomposition.

NOTE-

- Lossy join decomposition is also known as **careless decomposition**.
- This is because extraneous tuples get introduced in the natural join of the sub-relations.
- Extraneous tuples make the identification of the original tuples difficult.

Determining Whether Decomposition Is Lossless Or Lossy-

Consider a relation R is decomposed into two sub relations R₁ and R₂. 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,

$$R_1 \cap R_2 \neq \emptyset$$

Condition-03:

Intersection of both the sub relations must be a super key of either R₁ or R₂ or both. Thus,

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

Solved Examples to know whether a decomposition is lossy or lossless

Problem-01:

Consider a relation schema $R (A , B , C , D)$ with the functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Determine whether the decomposition of R into $R_1 (A , B)$ and $R_2 (C , D)$ is lossless or lossy.

Solution-

Condition-01

According to condition-01, union of both the sub relations must contain all the attributes of relation R . So, we have-

$$\begin{aligned} R_1 (A , B) \cup R_2 (C , D) \\ = R (A , B , C , D) \end{aligned}$$

Clearly, union of the sub relations contain all the attributes of relation R . Thus, condition-01 satisfies.

Condition-02:

According to condition-02, intersection of both the sub relations must not be null. So, we have-

$$R_1 (A , B) \cap R_2 (C , D) = \Phi$$

Clearly, intersection of the sub relations is null. So, condition-02 fails.

Thus, we conclude that the **decomposition is lossy**.

Problem-02:

Consider a relation schema $R (A , B , C , D)$ with the following functional dependencies-

$$A \rightarrow B$$

$$B \rightarrow C$$

$$C \rightarrow D$$

$$D \rightarrow B$$

Determine whether the decomposition of R into $R_1 (A , B)$, $R_2 (B , C)$ and $R_3 (B , D)$ is lossless or lossy.

Solution-

Strategy to Solve

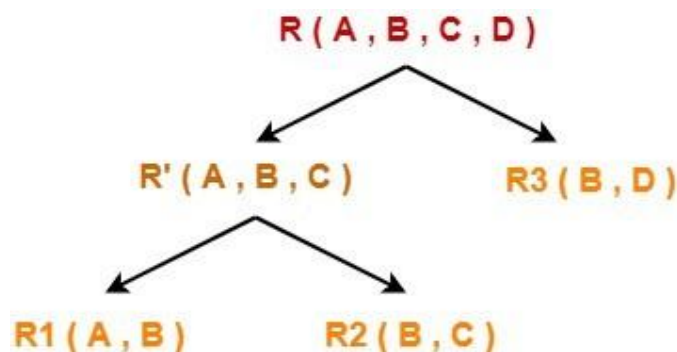
When a given relation is decomposed into more than two sub relations, then-

- Consider any one possible ways in which the relation might have been decomposed into those sub relations.
- First, divide the given relation into two sub relations.
- Then, divide the sub relations according to the sub relations given in the question.

As a thumb rule, remember-

Any relation can be decomposed only into two sub relations at a time.

Consider the original relation R was decomposed into the given sub relations as shown-



Decomposition of R(A, B, C, D) into R'(A, B, C) and R3(B, D)-

Condition-01

According to condition-01, union of both the sub relations must contain all the attributes of relation R.

So, we have-

$$\begin{aligned} R' (A , B , C) \cup R_3 (B , D) \\ = R (A , B , C , D) \end{aligned}$$

Clearly, union of the sub relations contain all the attributes of relation

Thus, condition-01 satisfies.

Condition-02:

According to condition-02, intersection of both the sub relations must not be null. So, we have-

$$R' (A , B , C) \cap R_3 (B , D) = B$$

Clearly, intersection of the sub relations is not null.

Thus, condition-02 satisfies.

Condition-03:

According to condition-03, intersection of both the sub relations must be the super key of one of the two sub relations or both.

So, we have-

$$R' (A , B , C) \cap R_3 (B , D) = B$$

Now, the closure of attribute B is-

$$B^+ = \{ B , C , D \}$$

Now, we see-

- Attribute 'B' can not determine attribute 'A' of sub relation R'.
- Thus, it is not a super key of the sub relation R'.
- Attribute 'B' can determine all the attributes of sub relation R3.
- Thus, it is a super key of the sub relation R3.

Clearly, intersection of the sub relations is a super key of one of the sub relations. So, condition-03 satisfies.

Thus, we conclude that the decomposition is lossless.

