DBMS

(DATABASE MANAGEMENT SYSTEM)

LESSON 5

Normalization:

Database normalization is the process of organizing the attributes of the database to reduce or eliminate **data redundancy (having the same data but at different places)** .

Problems because of data redundancy

Data redundancy unnecessarily increases the size of the database as the same data is repeated in many places. Inconsistency problems also arise during insert, delete and update operations.

Functional Dependency

Functional Dependency is a constraint between two sets of attributes in relation to a database. A functional dependency is denoted by an arrow (→). If an attribute A functionally determines B, then it is written as A → B.

For example, employee_id → name means employee_id functionally determines the name of the employee. As another example in a timetable database, {student_id, time} → {lecture_room}, student ID and time determine the lecture room where the student should be.

What does functionally dependent mean?

A function dependency A → B means for all instances of a particular value of A, there is the same value of B.

For example in the below table A → B is true, but B → A is not true as there are different values of A for B = 3.

A	B
1	3
2	3
4	0

1 3

4 0

Trivial Functional Dependency

X → Y is trivial only when Y is subset of X.

Examples

ABC → AB

ABC → A

ABC → ABC

Non Trivial Functional Dependencies

X → Y is a non trivial functional dependency when Y is not a subset of X.

X → Y is called completely non-trivial when X intersect Y is NULL.

Example:

Id → Name,

Name → DOB

Semi Non Trivial Functional Dependencies

X → Y is called semi non-trivial when X intersect Y is not NULL.

Examples:

AB → BC,

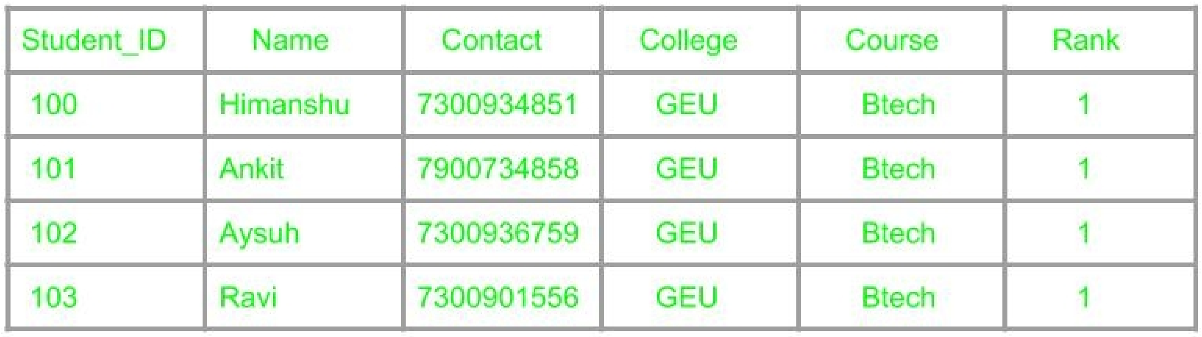
AD → DC

NORMAL FORMS:

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

Problem of redundancy in Database:

Redundancy means having multiple copies of same data in the database. This problem arises when a database is not normalized. Suppose a table of student details attributes are: student Id, student name, college name, college rank, course opted.

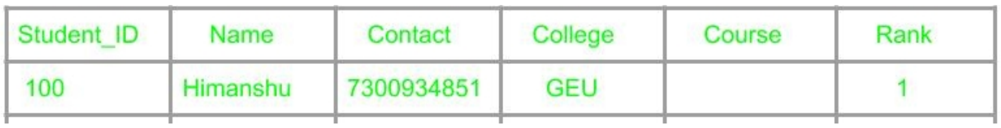


Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Aysuh	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1

As it can be observed that values of attribute college name, college rank, course is being repeated which can lead to problems. Problems caused due to redundancy are: Insertion anomaly, Deletion anomaly, and Updation anomaly.

1. Insertion Anomaly –

If a student detail has to be inserted whose course is not being decided yet then insertion will not be possible till the time course is decided for student.

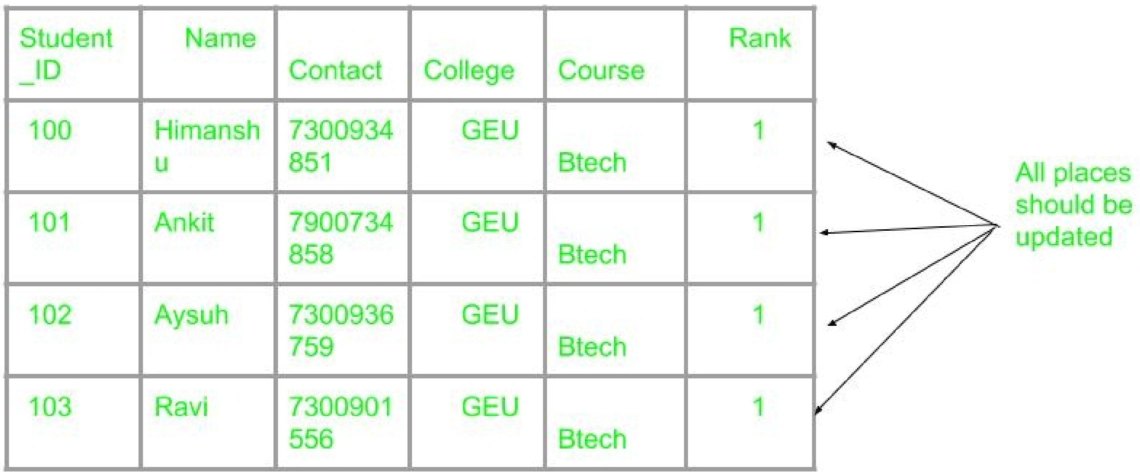


Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU		1

This problem happens when the insertion of a data record is not possible without adding some additional unrelated data to the record.

2. Deletion Anomaly –
If the details of students in this table are deleted then the details of college will also get deleted which should not occur by common sense.
This anomaly happens when deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.
It is not possible to delete some information without loosing some other information in the table as well.

3. Updation Anomaly –
Suppose if the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.



Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Aysuh	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1

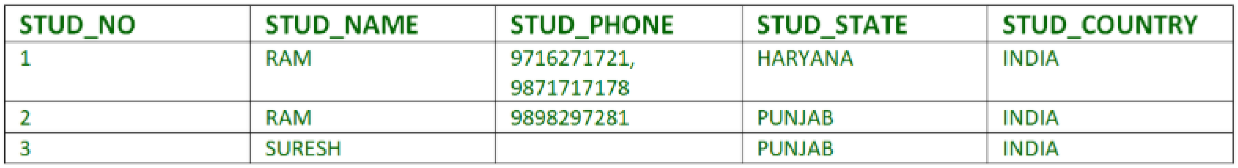
All places should be updated

Normal forms are used to eliminate the redundancy.

1. 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**.

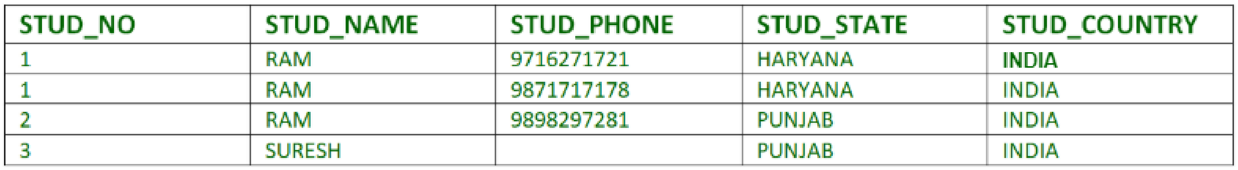
Example 1 – Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.



STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form



STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

2. 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 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 dont 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]

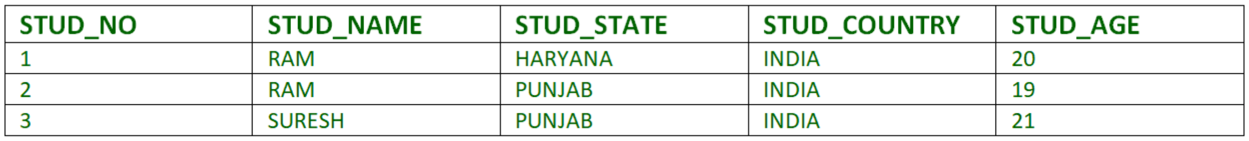
In the above relation, AB is the only candidate key and there is no partial dependency, i.e., any proper subset of AB doesn’t determine any non-prime attribute.

3. 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 -> Y

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



STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

Table 4

Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency.

- **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)

- **Example 2** – 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 attribute are on right sides of all functional dependencies are prime.

4. Boyce-Codd Normal Form (BCNF) –

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 –> Y, X is a super key.

- **Example 1** – 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 attribute which 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.

- **Example 2** –For example consider relation R(A, B, C)

  A -> BC,

B ->

A and B both are super keys so above relation is in BCNF.

Key Points –

1. BCNF is free from redundancy.
2. If a relation is in BCNF, then 3NF is also also satisfied.
3. If all attributes of relation are prime attribute, then the relation is always in 3NF.
4. A relation in a Relational Database is always and at least in 1NF form.
5. Every Binary Relation (a Relation with only 2 attributes) is always in BCNF.
6. 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).
7. 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.
8. 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.

GATE QUESTIONS:

1. Which normal form is considered adequate for normal relational database design?

(a) 2NF (b) 5NF (c) 4NF (d) 3NF

Ans: option (d)

Explanation:

A relational database table is often described as "normalized" if it is in the Third Normal Form because most of the 3NF tables are free of insertion, update, and deletion anomalies.

2. Consider a schema R(A, B, C, D) and functional dependencies A -> B and C -> D. Then the decomposition of R into R1 (A, B) and R2(C, D) is

(a) dependency preserving and lossless join

(b) lossless join but not dependency preserving

(c) dependency preserving but not lossless join

(d) not dependency preserving and not lossless join

Ans: option (c)

Explanation:

While decomposing a relational table we must verify the following properties:

i) Dependency Preserving Property: A decomposition is said to be dependency preserving if $F^+=(F1 \cup F2 \cup .. Fn)^+$, Where F^+=total functional dependencies(FDs) on universal relation R, F1 = set of FDs of R1, and F2 = set of FDs of R2.

For the above question R1 preserves A->B and R2 preserves C->D. Since the FDs of universal relation R is preserved by R1 and R2, the decomposition is dependency preserving.

ii) Lossless-Join Property:

The decomposition is a lossless-join decomposition of R if at least one of the following functional dependencies are in F+:-

a) R1 ∩ R2 -> R1

b) R1 ∩ R2 -> R2

It ensures that the attributes involved in the natural join ( ) are a candidate key for at least one of the two relations.In the above question schema R is decomposed into R1 (A, B) and R2(C, D), and R1 ∩ R2 is empty. So, the decomposition is not lossless.

3. Which one of the following statements about normal forms is FALSE?

(a) BCNF is stricter than 3NF

(b) Lossless, dependency-preserving decomposition into 3NF is always possible

(c) Lossless, dependency-preserving decomposition into BCNF is always possible

(d) Any relation with two attributes is in BCNF

Ans: option (c)

Explanation:

Achieving Lossless and dependency-preserving decomposition property into BCNF is difficult.

4. Which of the following is TRUE?

(a) Every relation in 2NF is also in BCNF

(b) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R

(c) Every relation in BCNF is also in 3NF

(d) No relation can be in both BCNF and 3NF

Ans: option c