UNIT-3(Data Base Design & Normalization)

Functional dependencies:

A functional dependency X->Y in a relation holds if two tuples having same value of attribute A also have same value for attribute B.

Where x ->determinant

Y->dependent

X	Y
a	1
b	2

X->Y

X	Y
a	1
a	2

here ,
y is not functionaly dependent on x

---

a->1

b->2

a->1

a->2

i.e Attribute Y is functionally dependent upon attribute X if a value of X determines a single value of attribute Y at any one time.

If t1(x)=t2(x) then t1(y)=t2(y)

	X	Y
t1	a	1
	b	2
t2	a	1
	b	2

Here in the above relation: t1(x)=a , t2(x)=a and t1(y)=1,t2(y)=1

so ,here X→Y

Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

Employee number→Employee Name

Employee number->Salary

Employee number→City

Example2:

eid	ename	department	Designation	Age
1	Amit	Production	Manager	40
2	Sumit	Marketing	Asst.Manager	35
3	Rahul	Production	Team Leader	30
1	Amit	Markting	Asst.manager	40
5	Ronit	Finance	Team Leader	32
6	Mohit	R & D	Manager	45

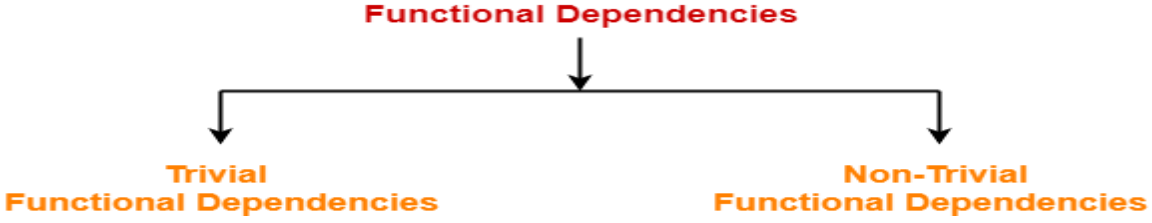
In the above table:
Functional Dependencies are:
1.eid→ename
2.eid→age
3.ename,age→eid
4.emane→age

Invalid functional Dependencies are

1.eid->department
2.eid→Designation
3.ename→department
4.ename→designation

Types Of Functional Dependencies-

There are two types of functional dependencies-



1. Trivial Functional Dependencies
2. Non-trivial Functional Dependencies

1. Trivial Functional Dependencies-

=>A functional dependency X → Y is said to be trivial if and only if Y ⊆ X.

=>Thus, if RHS of a functional dependency is a subset of LHS, then it is called as a trivial functional dependency.

Examples-

The examples of trivial functional dependencies are-

AB → A

AB → B

AB → AB

2. Non-Trivial Functional Dependencies-

- A functional dependency X → Y is said to be non-trivial if and only if Y ⊄ X.
- Thus, if there exists at least one attribute in the RHS of a functional dependency that is not a part of LHS, then it is called as a non-trivial functional dependency.

Examples-

The examples of non-trivial functional dependencies are-

- AB → BC
- AB → CD

Armstrong Axioms or Inference Rules-

1. Reflexivity-

If B is a subset of A, then A → B always holds.

2.Transitivity-

If A → B and B → C, then A → C always holds.

3.Augmentation-

If A → B, then AC → BC always holds.

4.Decomposition-

If A → BC, then A → B and A → C always holds.

5.Composition -

If A → B and C → D, then AC → BD always holds.

6.Additive or Union

If A → B and A → C, then A → BC always holds.

Rules for Functional Dependency-

Rule-01:

A functional dependency X → Y will always hold if all the values of X are unique (different) irrespective of the values of Y.

Example-

Consider the following table-

A	B	C	D	E
5	4	3	2	2
8	5	3	2	1
1	9	3	3	5
4	7	3	3	8

The following functional dependencies will always hold since all the values of attribute 'A' are unique-

- A → B
- A → BC
- A → CD
- A → BCD
- A → DE
- A → BCDE

In general, we can say following functional dependency will always hold-

A → Any combination of attributes A, B, C, D, E

Similar will be the case for attributes B and E.

Rule-02:

A functional dependency X → Y will always hold if all the values of Y are same irrespective of the values of X.

Example-

Consider the following table-

A	B	C	D	E
5	4	3	2	2
8	5	3	2	1
1	9	3	3	5
4	7	3	3	8

The following functional dependencies will always hold since all the values of attribute 'C' are same-

- A → C
- AB → C
- ABDE → C
- DE → C
- AE → C

In general, we can say following functional dependency will always hold true-

Any combination of attributes A, B, C, D, E → C

Closure of an Attribute Set-

- The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.
- Closure of attribute set {X} is denoted as $\{X\}^+$.

Example-

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

$$A \rightarrow BC$$

$$BC \rightarrow DE$$

$$D \rightarrow F$$

$$CF \rightarrow G$$

Now, let us find the closure of some attributes and attribute sets-

Closure of attribute A-

$A^+ = \{ A \}$

$= \{ A , B , C \}$ (Using $A \rightarrow BC$)

$= \{ A , B , C , D , E \}$ (Using $BC \rightarrow DE$)

$= \{ A , B , C , D , E , F \}$ (Using $D \rightarrow F$)

$= \{ A , B , C , D , E , F , G \}$ (Using $CF \rightarrow G$)

Thus,

$$A^+ = \{ A , B , C , D , E , F , G \}$$

Closure of attribute B-

$B^+ = \{B\}$

Closure of attribute D-

$D^+ = \{ D \}$

$= \{ D , F \}$ (Using $D \rightarrow F$)

We can not determine any other attribute using attributes D and F contained in the result set.

Thus,

$$D^+ = \{ D , F \}$$

Closure of attribute set {B, C}-

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

$= \{ B , C , D , E \}$ (Using $BC \rightarrow DE$)

$= \{ B , C , D , E , F \}$ (Using $D \rightarrow F$)

$= \{ B , C , D , E , F , G \}$ (Using $CF \rightarrow G$)

Thus,

$$\{ B , C \}^+ = \{ B , C , D , E , F , G \}$$

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

$$AB \rightarrow CD$$

$$AF \rightarrow D$$

$$DE \rightarrow F$$

$$C \rightarrow G$$

$$F \rightarrow E$$

$$G \rightarrow A$$

Find the closure of the following

(a) $\{ CF \}^+$

(b) $\{ BG \}^+$

(c) $\{ AF \}^+$

(d) $\{ AB \}^+$

Solution-

(a):

$\{ CF \}^+ = \{ C , F \}$

$= \{ C , F , G \}$ (Using $C \rightarrow G$)

$= \{ C , E , F , G \}$ (Using $F \rightarrow E$)

$= \{ A , C , E , E , F \}$ (Using $G \rightarrow A$)

$= \{ A , C , D , E , F , G \}$ (Using $AF \rightarrow D$)

(b):

$\{ BG \}^+ = \{ B , G \}$

$= \{ A , B , G \}$ (Using $G \rightarrow A$)

$= \{ A , B , C , D , G \}$ (Using $AB \rightarrow CD$)

(c):

$\{ AF \}^+ = \{ A , F \}$

$= \{ A , D , F \}$ (Using $AF \rightarrow D$)

$= \{ A , D , E , F \}$ (Using $F \rightarrow E$)

(d):

$\{ AB \}^+ = \{ A , B \}$

$= \{ A , B , C , D \}$ (Using $AB \rightarrow CD$)

$= \{ A , B , C , D , G \}$ (Using $C \rightarrow G$)

Equivalence of Two Sets of Functional Dependencies-

In DBMS,

- Two different sets of functional dependencies for a given relation may or may not be equivalent.
- If F and G are the two sets of functional dependencies, then following 3 cases are possible-

Case-01: F covers G ($F \supseteq G$)

Case-02: G covers F ($G \supseteq F$)

Case-03: Both F and G cover each other ($F = G$)

Case-01: Determining Whether F Covers G-

Following steps are followed to determine whether F covers G or not-

Step-01:

=>Take the functional dependencies of set G into consideration.

=>For each functional dependency $X \rightarrow Y$, find the closure of X using the functional dependencies of set G.

Step-02:

=>Take the functional dependencies of set G into consideration.

=>For each functional dependency $X \rightarrow Y$, find the closure of X using the functional dependencies of set F.

Step-03:

=>Compare the results of Step-01 and Step-02.

=>If the functional dependencies of set F has determined all those attributes that were determined by the functional dependencies of set G, then it means F covers G.

- Thus, we conclude F covers G ($F \supseteq G$) otherwise not.

Case-02: Determining Whether G Covers F-

Following steps are followed to determine whether G covers F or not-

Step-01:

Take the functional dependencies of set F into consideration.

- For each functional dependency $X \rightarrow Y$, find the closure of X using the functional dependencies of set F.

Step-02:

Take the functional dependencies of set F into consideration.

- For each functional dependency $X \rightarrow Y$, find the closure of X using the functional dependencies of set G.

Step-03:

Compare the results of Step-01 and Step-02.

- If the functional dependencies of set G has determined all those attributes that were determined by the functional dependencies of set F, then it means G covers F.
- Thus, we conclude G covers F ($G \supseteq F$) otherwise not.

Case-03: Determining Whether Both F and G Cover Each Other-

- If F covers G and G covers F, then both F and G cover each other.
- Thus, if both the above cases hold true, we conclude both F and G cover each other

  (F = G).

Example:

A relation R (A , C , D , E , H) is having two functional dependencies sets F and G as shown-

Set F-

$$A \rightarrow C$$

$$AC \rightarrow D$$

$$E \rightarrow AD$$

$E \rightarrow H$

Set G-

$A \rightarrow CD$

$E \rightarrow AH$

Which of the following holds true?

(A) $G \supseteq F$

(B) $F \supseteq G$

(C) $F = G$

(D) All of the above

OR

Ques: Consider the following two sets of FDs, F={A→C, AC→D, E→AD,E→H} and G={A→CD,E→AH} check whether they are equivalent. (AKTU MCA-2009-10)

Solution-

Determining whether F covers G-

Step-01:

$(A)^+ = \{A\}$

$=\{A,C,D\}$(using A→CD)

$(E)^+ = \{E\}$

$=\{E,A,H\}$

$=\{E,A,H,C,D\}$

$=\{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AH$ using set G

Step-02:

- $(A)^+ = \{ A\}$

$=\{A,C,D\}$// closure of left side of $A \rightarrow CD$ using set F

$(E)^+ = \{E, A, D, C, H\}$

$= \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AH$ using set F

Step-03:

Comparing the results of Step-01 and Step-02, we find-

- Functional dependencies of set F can determine all the attributes which have been determined by the functional dependencies of set G.
- Thus, we conclude F covers G i.e. F ⊇ G.

Determining whether G covers F-

Step-01: for FDs F={A→C, AC→D, E→AD,E→H}

$(A)^+$ = {A,C,D}

={ A , C , D } // closure of left side of A → C using set F

$(AC)^+$ = {A,C,D}

={ A , C , D } // closure of left side of AC → D using set F

$(E)^+$ = {E,A,D,H,C}

={ A , C , D , E , H } // closure of left side of E → AD and E → H using set F

Step-02: for FDs G={A→CD,E→AH}

- $(A)^+$ = {A,C,D}

  ={ A , C , D } // closure of left side of A → C using set G
- $(AC)^+$ ={A,C,D}

  ={ A , C , D } // closure of left side of AC → D using set G
- $(E)^+$ ={E,A,H,C,D}

  = { A , C , D , E , H } // closure of left side of E → AD and E → H using set G

Step-03:

Comparing the results of Step-01 and Step-02, we find-

- Functional dependencies of set G can determine all the attributes which have been determined by the functional dependencies of set F.
- Thus, we conclude G covers F i.e. G ⊇ F.

Determining whether both F and G cover each other-

From Step-01, we conclude F covers G.

From Step-02, we conclude G covers F.

Thus, we conclude both F and G cover each other i.e. F = G.

Thus, Option (D) is correct.

i.e Both functional dependencies are Equivalent.

Ques: Consider the following relation R(P,Q,R,S) with two sets of FDs, X={P→Q, Q→R, R→S} and

Y={P→QR, R→S} check whether they are equivalent.

Ques: Consider the following relation R(A,B,C) with two sets of FDs, F={A→B, B→C, C→A} and G={A→BC, B→A,C→A} check whether they are equivalent.

Canonical Cover or Canonical form or Irreducible set of Functional Dependencies:

- A canonical cover is a simplified and reduced version of the given set of functional dependencies.
- Since it is a reduced version, it is also called as **Irreducible set**.

Canonical cover is free from all the extraneous functional dependencies.

- Working with the set containing extraneous functional dependencies increases the computation time.
- Therefore, the given set is reduced by eliminating the useless functional dependencies.
- This reduces the computation time and working with the irreducible set becomes easier.

Extraneous attributes: An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies.

Example: The following functional dependencies hold true for the relational scheme

R (W , X , Y , Z) –

$$X \rightarrow W$$

$$WZ \rightarrow XY$$

$$Y \rightarrow WXZ$$

Write the canonical cover or irreducible equivalent for this set of functional dependencies.

Solution-

Step-01: Write all the functional dependencies such that each contains exactly one attribute on its right side using decomposition rule-

$$X \rightarrow W$$

$$WZ \rightarrow X$$

$$WZ \rightarrow Y$$

$$Y \rightarrow W$$

$$Y \rightarrow X$$

$$Y \rightarrow Z$$

Step-02: Check the essentiality of each functional dependency one by one.

For X → W: Considering $X \rightarrow W$, $(X)^+ = \{X, W\}$

Ignoring $X \rightarrow W$, $(X)^+ = \{ X \}$

Now, Clearly, the two results are different.

- Thus, we conclude that $X \rightarrow W$ is essential and can not be eliminated.

For $WZ \rightarrow X$: Considering $WZ \rightarrow X$, $(WZ)^+ = \{ W , Z, X, Y \}$

Ignoring $WZ \rightarrow X$, $(WZ)^+ = \{ W ,Z, Y, X\}$

Now, Clearly, the two results are same.

- Thus, we conclude that $WZ \rightarrow X$ is non-essential and can be eliminated.

Eliminating $WZ \rightarrow X$, our set of functional dependencies reduces to-

$$X \rightarrow W$$

$$WZ \rightarrow Y$$

$$Y \rightarrow W$$

$$Y \rightarrow X$$

$$Y \rightarrow Z$$

Now, we will consider this reduced set in further checks.

For $WZ \rightarrow Y$: Considering $WZ \rightarrow Y$, $(WZ)^+ = \{ W , Z, Y, X\}$

Ignoring $WZ \rightarrow Y$, $(WZ)^+ = \{ W, Z \}$

Now, Clearly, the two results are different.

- Thus, we conclude that $WZ \rightarrow Y$ is essential and can not be eliminated.

For $Y \rightarrow W$: Considering $Y \rightarrow W$, $(Y)^+ = \{ Y,W,X,Z \}=\{W,X,Y,Z\}$

Ignoring $Y \rightarrow W$, $(Y)^+ = \{ Y,X,W,Z \}=\{W,X,Y,Z\}$

Now, Clearly, the two results are same.

- Thus, we conclude that $Y \rightarrow W$ is non-essential and can be eliminated.

Eliminating $Y \rightarrow W$, our set of functional dependencies reduces to-

$$X \rightarrow W$$

$$WZ \rightarrow Y$$

$$Y \rightarrow X$$

$$Y \rightarrow Z$$

For $Y \rightarrow X$: Considering $Y \rightarrow X$, $(Y)^+ = \{ Y,X,W,Z \}$

Ignoring $Y \rightarrow X$, $(Y)^+ = \{ Y, Z\}$

Now, Clearly, the two results are different.

- Thus, we conclude that $Y \rightarrow X$ is essential and can not be eliminated.

For $Y \rightarrow Z$: Considering $Y \rightarrow Z$, $(Y)^+ = \{ Y,Z,X,W \}$

Ignoring $Y \rightarrow Z$, $(Y)^+ = \{ Y,X,W\}$

Now, Clearly, the two results are different.

- Thus, we conclude that $Y \rightarrow Z$ is essential and can not be eliminated.

From here, our essential functional dependencies are-

$$X \rightarrow W$$

$$WZ \rightarrow Y$$

$$Y \rightarrow X$$

$$Y \rightarrow Z$$

Step-03: Consider the functional dependencies having more than one attribute on their left side.

=>Check if their left side can be reduced.

In our set, Only $WZ \rightarrow Y$ contains more than one attribute on its left side.

- Considering $WZ \rightarrow Y$, $(WZ)^+ = \{ W , X , Y , Z \}$

Now, Consider all the possible subsets of WZ.

- Check if the closure result of any subset matches to the closure result of WZ.

$$(W)^+ = \{ W\}$$

$$(Z)^+ = \{ Z \}$$

Clearly,

- None of the subsets have the same closure result same as that of the entire left side.
- Thus, we conclude that we can not write $WZ \rightarrow Y$ as $W \rightarrow Y$ or $Z \rightarrow Y$.
- Thus, set of functional dependencies obtained in step-02 is the canonical cover.

Finally, the canonical cover is-

$$X \rightarrow W$$

$$WZ \rightarrow Y$$

$$Y \rightarrow X$$

$$Y \rightarrow Z$$

Ques1: A set of FD's for the relation R(ABCD) is

A→B

C→B

D→ABC

AC→D

Find a canonical cover for this set.

Ques2: A set of FD's for the relation R(VWXYZ) is

V→W

VW→X

Y→VXZ

Find a canonical cover for this set.

Ques:3

SECTION C

3. Attempt any *one* part of the following: 7 x 1 =7

(a). A set of FD's for the relation R-{A,B,C,D,E,F} is AB ->C, C ->A,BC -> D,ACD -> B, BE ->C, EC ->FA, CF -> BD,, f->E: Find a Canonical cover for this set.

(AKTU. MCA-2018-19)

Finding the Keys Using Closure-

Super Key-super key is a set of attribute with the help of which you can uniquely identify a row or tuple in a table or you can uniquely identify all the row in a table.

Finding super key using closure:

- If the closure result of an attribute set contains all the attributes of the relation, then that attribute set is called as a super key of that relation.

Example- **R(ABCD) with FD's**

ABC→D

AB→CD

A→BCD

Sol: $(ABC)^+$=(A,B,C,D)

ABC→Super key

A^+={A,B,C,D}

A→Super key

B+→{B}

B is not a super key.

C+={C}

C is not a super key.

AB+=(A,B,C,D)

AB is super key.

In the above example,

- The closure of attribute ABC,AB and A is the entire relation schema.
- Thus, attribute ABC, AB, A is a super key for that relation.

Candidate Key- A minimal super key is called candidate key.

Finding candidate key using closure:

If there exists no subset of an attribute set whose closure contains all the attributes of the relation, then that attribute set is called as a candidate key of that relation.

Example-

In the above example,

- No subset of attribute A contains all the attributes of the relation.
- Thus, attribute A is also a candidate key for that relation.

Finding Candidate Keys-

We can determine the candidate keys of a given relation using the following steps-

Step-01: Determine all essential attributes of the given relation.

- Essential attributes are those attributes which are not present on RHS of any functional dependency.
- Essential attributes are always a part of every candidate key.
- This is because they can not be determined by other attributes.

Example

Let R(A, B, C, D, E, F) be a relation scheme with the following functional dependencies-

$$A \rightarrow B$$

$$C \rightarrow D$$

$$D \rightarrow E$$

Here, the attributes which are not present on RHS of any functional dependency are A, C and F.

So, essential attributes are- **A, C and F**.

Step-02: The remaining attributes of the relation are non-essential attributes.

- This is because they can be determined by using essential attributes.

Now, following two cases are possible-

Case-01: If all essential attributes together can determine all remaining non-essential attributes, then-

- The combination of essential attributes is the candidate key.
- It is the only possible candidate key.

Case-02: If all essential attributes together can not determine all remaining non-essential attributes, then-

- The set of essential attributes and some non-essential attributes will be the candidate key(s).
- In this case, multiple candidate keys are possible.
- To find the candidate keys, we check different combinations of essential and non-essential attributes.

We will further understand how to find candidate keys with the help of following problems.

The following practice problems are based on Case-01.

Example:

Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies-

$$C \rightarrow F$$

$$E \rightarrow A$$

$$EC \rightarrow D$$

$$A \rightarrow B$$

determine the total number of candidate keys?.

Solution- We will find candidate keys of the given relation in the following steps-

Step-01: Determine all essential attributes of the given relation.

- Essential attributes of the relation are- C and E.
- So, attributes C and E will definitely be a part of every candidate key.

Step-02: Now,We will check if the essential attributes together can determine all remaining non-essential attributes.

- To check, we find the closure of CE.

So, we have-$\{ CE \}^+ = \{ C , E \}$

$= \{ C , E , F \}$ (Using $C \rightarrow F$)

$= \{ A , C , E , F \}$ (Using $E \rightarrow A$)

$= \{ A , C , D , E , F \}$ (Using $EC \rightarrow D$)

= { A , B , C , D , E , F } (Using A → B)

We conclude that CE can determine all the attributes of the given relation.

So, CE is the only possible candidate key of the relation.

Ques: Find total Number of candidate key in a relation R(A,B,C,D)

A→B

B→C

C→A

Sol: essential attribute: D

D+={D}(not a key)

AD+={A,D,B,C}={A,B,C,D}

AD-(candidate key)

BD+={B,D,C,A}={A,B,C,D}

BD-Candidate key

CD+={C,D,A,B}

CD-Candidate key

Now, if we check ACD,BCD,ABD and ABCD these can be super key but not candidate key because their subset AD,BD,CD are already candidate key.

Ques1: Find total Number of candidate key in a relation R(A,B,C,D) with following FD's

AB→C D

D→A

Ques2: Find total Number of candidate key in a relation R(A,B,C,D,E,F) with following FD's

AB→C

C→D

B→AE

Ques3: Find total Number of candidate key in a relation R(A,B,C,D) with following FD's

AB→CD

C→A

D→B

Sol: Here, all attribute of right side of FD's are in relation i.e there is no essential attribute.

So, we need to find closure of All.

A+={A}

B+={B}

C+={C,A}

D+={D,B}

AB+={A,B,C,D}(super key)(candidate key)

AC+={A,C}

AD+={A,D,B,C}={A,B,C,D}(super key)(candidate)

BC+={B,C,A,D}(super key)(candidate key)

BD+={B,D}

CD+={C,D,A,B}(super key)(candidate key)

SO, Total candidate key will be 3. AD,BC,CD

Ques4: Find total Number of candidate key in a relation R(A,B,C,D,E) with following FD's

AB→CD

D→A

BC→DE

Ques5: Find total Number of candidate key in a relation R(WXYZ) with following FD's

Z→W

Y→XZ

XW→Y

Anomalies in DBMS :

Anomalies are caused when there is too much redundancy in the database's information. Anomalies can often be caused when the tables that make up the database suffer from poor construction. So, what does "poor construction" mean? Poor table design will become evident if, when the designer creates the database, he doesn't identify the entities that depend on each other for existence.

The normalization process was created largely in order to reduce the negative effects of creating tables that will introduce anomalies into the database.

Example:

e_id	e_name	e_address	e_dept
101	Rohit	Delhi	D001
101	Rohit	Delhi	D002
123	Manish	Agra	D890
166	Ganesh	Chennai	D900
166	Ganesh	Chennai	D004

The above table is not normalized. We will see the problems that we face when the table is not normalized.

Type of Anomalies in DBMS:

Update anomaly:

In the above table, we have two rows for employee Rohit as he belongs to two departments of the company. If we want to update the address of Rohit then we have to update the same in two rows or the data will become inconsistent.

If somehow, the correct address gets updated in one department but not in other then as per the database, Rohit would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly:

Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if the e_dept field doesn't allow nulls.

Delete anomaly:

Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having e_dept as D890 would also delete the information of employee Manish since he is assigned only to this department.

Normalization:

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.

The standard normal forms used are-

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

There exists several other normal forms even after BCNF but generally we normalize till BCNF only.

1.First Normal Form-

A given relation is called in First Normal Form (1NF) if each attribute 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-

The following relation is not in 1NF-

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

Relation is not in 1NF

However,

- This relation can be brought into 1NF.
- This can be done by rewriting the relation such that each attribute 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

Relation is in 1NF

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.

2. 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.

prime attribute :is an attribute that is part of any candidate key. Non prime attribute is an attribute that is not part of any candidate key. So, find out all possible candidate keys from

the given functional dependencies and mark the prime and non prime attributes.

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 → B is called a partial dependency if and only if-

1. A is a 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.

NOTE- To avoid partial dependency, incomplete candidate key must not determine any non-prime attribute.

- However, incomplete candidate key can determine prime attributes.

The normalization of 1NF relations to 2NF involves the **removal of partial dependencies**. If a partial dependency exists, we remove the partially dependent attribute(s) from the relation by placing them in a new relation along with a copy of their determinant.

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

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

Table 2

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C3	1000
C4	2000
C5	2000

EX: Check whether the following relation is in Second Normal form(2NF) or not if not conver it into 2NF

R(ABCD) with FDs

AB-->D
B-->C

AB are not in right side of dependency.
i.e AB are essential attribute
i.e AB will part of every candidate

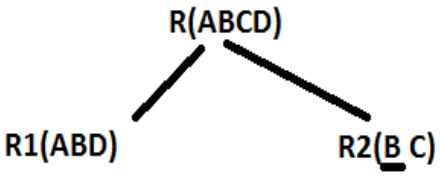
sol:

AB^+ ={A,B,D,C}={ABCD}
i.e AB is candidate key

prime attributes: AB
Non prime attribute:CD

AB-->D(full dependency)
B-->C(partial dependency)

so, relation is not in 2nd NF.



now relation R1 and R2
are in 2NF

ex: Let the relation R(ABC) with FDs

B-->C

Check wheterh above relation is in 2NF or not. if not convert it into 2NF

Sol: AB are essential attribute.

AB^+={A,B,C}

AB candidate key

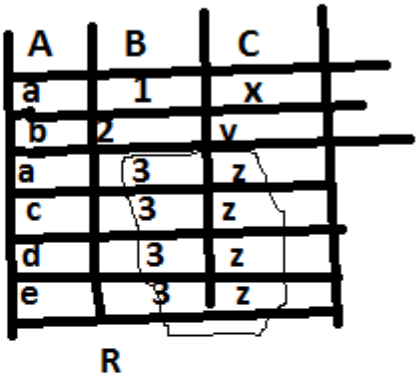
prime attribute: AB

non prime attribute:C

FD,

B-->C(partial dependency)

i.e table is not in 2NF



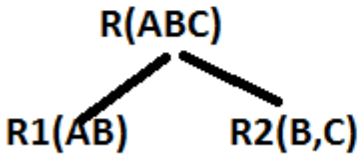
R

R1

A	B
a	1
b	2
a	3
c	3
d	3
e	3

R2

B	C
1	x
2	y
3	z



Q: Consider the relation R(a,b,c,d) with set F={a→c, b→d} Decompose this relation in 2NF.

(AKTU: B.TECH-2020-20210)

Q: Consider the relation R(A,B,C,D,E,F,G,H,I,J) and a set of following functional dependencies F={AB→C,A→DE,B→F,F→GH,D→IJ} .Determine the keys for R? Decompose R into 2NF.
(AKTU: B.TECH-2020-20210)

Third Normal Form (3NF):

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 → 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.

i.e Non-Prime→Non-Prime (transitive dependency)

OR

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

Any one condition holds for each non-trivial functional dependency A → B

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

i.e Non-prime → Non-prime, does not hold then relation is not in 3NF

OR

A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

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

The normalization of 2NF relations to 3NF involves the removal of transitive dependencies. If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.

Example-1:
In relation STUDENT given in Table 4,

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

FD set:
{STUD_NO -> STUD_NAME, STUD_NO -> STUD_STATE, STUD_STATE -> STUD_COUNTRY, STUD_NO -> STUD_AGE}

Candidate Key:
{STUD_NO}

Stud_no→stud_state and stud_state→stud_country

x→y and y→z then x→z

Stud_no→stud_country

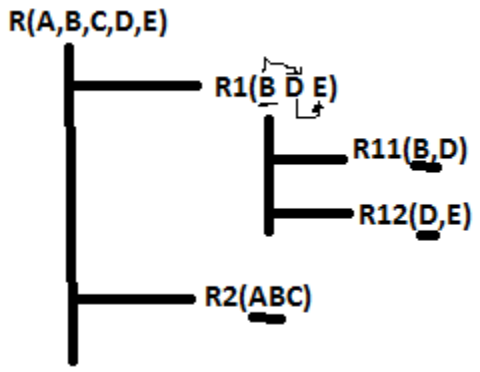
For this relation in table 4, STUD_NO -> STUD_STATE and STUD_STATE -> STUD_COUNTRY are true. So STUD_COUNTRY is transitively dependent on STUD_NO. It violates the third normal form. To convert it in third normal form, we will decompose the relation STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY_STUD_AGE) as:

STUDENT (STUD_NO, STUD_NAME, Stud_STATE, STUD_AGE)
STATE_COUNTRY (stud_STATE, stud_COUNTRY)

Q: Consider a relation R(A,B,C,D,E) with set F={AB→C,B→D,D→E} decompose the given relation in 3NF.
SOL:
Here, essential attributes are:{AB}
So AB will be part of every candidate key.

Finding AB^+={A,B,C,D,E} so, AB is candidate key.
Prime attribute are:AB
Non Prime attribute are: CDE
Now, from FD's
AB→C(it is in 3NF since AB is super key or candidate key)
B→D(Here,B is not super key and D is not prime so it is not in third normal form)
D→E(nonprime→nonprime) so this FD's is not in 3NF

Decompose it in 3NF:



Q: Consider a relation R(A,B,C,D,E) with set F={A→CD,C→B,B→AE}what are the prime attributes of this relation and decompose the given relation in 3NF. **(AKTU: B.Tech-2020-21)**

Q: Normalize the given relation upto 3NF R:{A,B,C,D} F={AB→D, AC→BD, B→C}

(AKTU: B.Tech-2020-21)

Boyce-Codd Normal Form (BCNF):

This is also known as 3.5 NF.

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 → B, A is a super key of the relation.

Consider the below table:

Student ID	Subject	Professor
1DT15ENG01	SQL	Prof. Mishra
1DT15ENG02	JAVA	Prof. Anand
1DT15ENG02	C++	Prof. Kanthi
1DT15ENG03	JAVA	Prof. Anand
1DT15ENG04	DBMS	Prof. Lokesh

- One student can enrol for multiple subjects.
- There can be multiple professors teaching one subject
- And, For each subject, a professor is assigned to the student

In this table, all the normal forms are satisfied except BCNF. Why?

As you can see **Student ID,** and **Subject** form the primary key, which means the **Subject** column is a **prime attribute**. But, there is one more dependency, **Professor → Subject**.

And while **Subject** is a prime attribute, **Professor** is a **non-prime attribute**, which is not allowed by BCNF.

Now in order to satisfy the BCNF, we will be dividing the table into two parts. One table will hold **Student ID** which already exists and newly created column **Professor ID**.

Student ID	Professor ID
1DT15ENG01	1DTPF01
1DT15ENG02	1DTPF02
1DT15ENG02	1DTPF03
:	:

And in the second table, we will have the columns Professor ID, Professor and Subject.

Professor ID	Professor	Subject
1DTPF01	Prof. Mishra	SQL
1DTPF02	Prof. Anand	JAVA
1DTPF03	Prof. Kanthi	C++
:	:	:

By doing this we are satisfied the Boyce Codd Normal Form.

Q: Consider the relation schema R=(ABC) and the following set of FD's:

F=(AB→C,C→B). Is R in BCNF ? if not decompose it .

Sol: here, A is essential attribute ,A will be part of every candidate key

A^+={A}

AB^+={A,B,C} so AB is candidate key

AC^+={A,C,B} so, AC is candidate key.

Prime attribute: ABC(part of candidate key)

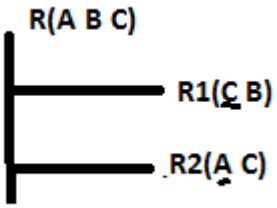
Non prime attribute:None

Now from Dependency,

AB→C(Since AB is candidate key i.e super key so this FD is in BCNF)

C→B(since C is not a candidate key so relation is not in BCNF)

So, decompose the Relation:



Q: Consider the relation schema R=(ABCDE) and the following set of FD's:

F=(A→B, BC→E, ED→A)

i. List all keys for R.
ii. Is R in 3NF? Justify your answer
iii. Is R in BCNF ? justify your answer

Sol: essential attributes: CD which will be part of every key.

CD^+={C,D} (not a candidate key)

CDA+={C,D,A,B,E}=(it is candidate key)

CDB^+={C,D,B,E,A}=(it is a candidate key)

CDE+={C,D,E,A,B}={it is a candidate key}

A+={A,B}=not super key

ABC={A,B,C,E}=not super key

ABD=

ABE=

BCE={B,C,E}not super key

ABCD={A,B,C,D,E}(it is a super key)

ABCE=

ABED=

ABCDE+={A,B,C,D,E}(Super key)

(ii). We have candidate keys are: CDA, CDB ,CDE

Prime attribute:{A,B,C,D,E}

Non Prime attribute:{none}

F=(A→B, BC→E, ED→A)

Now from FDs:

A→B(since B is prime attribute so it is in 3NF)

BC→E(since E is a prime attribute so it is in 3NF)

ED→A(since A is a prime attribute so it is in 3NF)

(iii). Checking for BCNF:

F=(A→B, BC→E, ED→A)

Now from FDs:

A→B(Since A is not super key so it is not in BCNF)

So, the relation is not in BCNF.

Q: Write the difference between 3NF and BCNF . find normal form of the relation

R(A,B,C,D,E) having FD's set F={A→B, BC→E, ED→A} (AKTU: BTECH-2018-19)

Example-1:
Q: What is highest normal form of the relation R(W,X,Y,Z) with the set F={WY→XZ, X→Y}

Q:Find the highest normal form of a relation R(A, B, C, D, E) with FD set as:

{ BC->D, AC->BE, B->E }

Sol: essential attribute: AC

- **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.

Checking for BCNF: from FDs

{ BC->D, AC->BE, B->E }

BC→D(not in BCNF since BC is not a super key)

So, relation is not in BCNF

Checking for 3NF: from FD's

BC→D(Not in 3NF because neither BC is super key nor D is prime attribute.

So,relation is not in 3NF

Checking for 2NF: form FDs

BC→D(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).

So the highest normal form of relation will be 2nd Normal form.

Decomposition of a Relation-

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.

Note: Every decomposition must always be lossless.

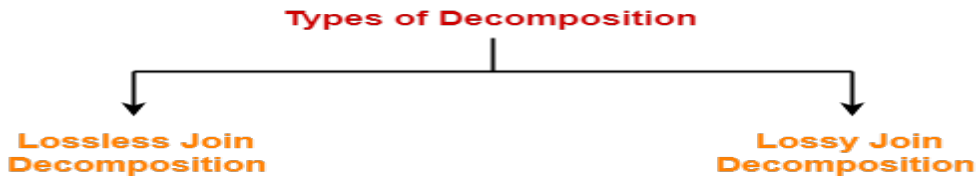
2. Dependency Preservation-

Dependency preservation ensures-

- None of the functional dependencies that holds on the original relation are lost.
- The sub relations still hold or satisfy the functional dependencies of the original relation.

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 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 ⋈ is a natural join operator

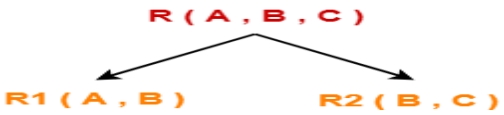
Example-

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 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, 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 (⋈) 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 ⋈ is a natural join operator

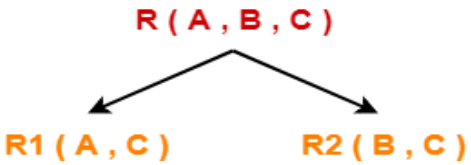
Example-

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.

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 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,

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

PROBLEMS BASED ON DETERMINING WHETHER DECOMPOSITION IS LOSSLESS OR LOSSY-

Problem-01:

Consider a relation schema R (A , B , C , D) with the functional dependencies A → B and C → D.

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

Solution-

To determine whether the decomposition is lossless or lossy,

- We will check all the conditions one by one.
- If any of the conditions fail, then the decomposition is lossy otherwise lossless.

Condition-01:

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

So, we have-

$$R_1 (A , B) \cup R_2 (C , D)$$

$$= R (A , B , C , D)$$

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-

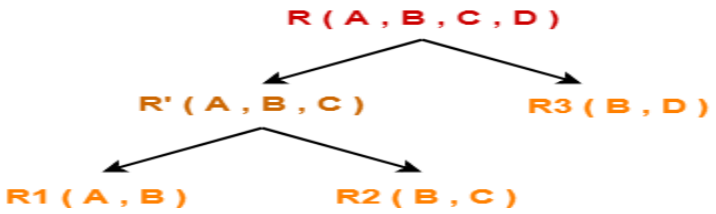
- Consider any one possible ways in which the relation might have been decomposed into those sub relations.
- First, divide the given relation into two sub relations.

- 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 R_3(B, D)-

To determine whether the decomposition is lossless or lossy,

Condition-01:

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

So, we have-

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

$$= R (A , B , C , D)$$

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-

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

$$= B$$

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-

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

$$= B$$

Given FD's $A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$$D \rightarrow B$$

Now, the closure of attribute B is-

$$B^+ = \{ B , C , D \}$$(not supper key for R' but it is super key for R3)

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 R_3.
- Thus, it is a super key of the sub relation R_3.

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.

Decomposition of R'(A, B, C) into R_1(A, B) and R_2(B, C)-

To determine whether the decomposition is lossless or lossy,

Condition-01:

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

So, we have-

$$R_1 (A , B) \cup R_2 (B , C)$$

$$= R' (A , B , C)$$

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-

$$R_1 (A , B) \cap R_2 (B , C)$$

$$= B$$

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-

$$R_1 (A , B) \cap R_2 (B , C)$$

$$= B$$

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_1.

- Thus, it is not a super key of the sub relation R_1.
- Attribute 'B' can determine all the attributes of sub relation R_2.
- Thus, it is a super key of the sub relation R_2.

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.

Conclusion-

Overall decomposition of relation R into sub relations R_1, R_2 and R_3 is lossless.

Q: suppose that we decompose the schema R(A,B,C,D,E) into R1 and R2 as R1:{A,B,C} R2:{C,D,E} .justify whether it is lossless decomposition or not. (AKTU MCA:2019)

(MVD)Multivalued Dependency:

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

1. BIKE_MODEL $\twoheadrightarrow$ MANUF_YEAR
2. BIKE_MODEL $\twoheadrightarrow$ COLOR

   This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

Fourth normal form (4NF):

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

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

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 4NF, 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

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Fifth normal form (5NF)

- 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).

(JD)Join Dependency:

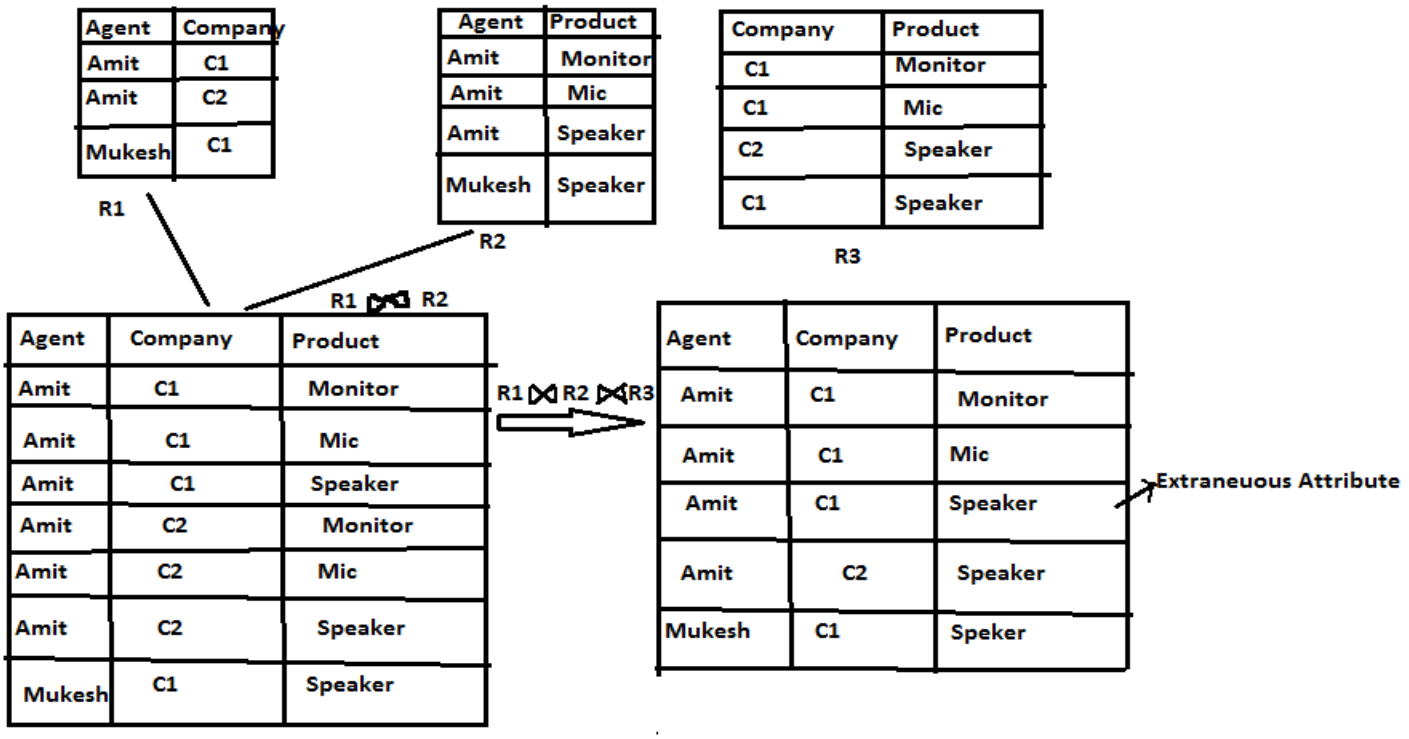
Let R be relation schema and R1,R2,.....Rn be decomposition of R, R is said to satisfy the join dependency *(R1,R2.....Rn) if and only if

$$R1 \bowtie R2 \bowtie R3 \ldots\ldots\ldots\ldots \bowtie Rn = R$$

Example of Join Dependency:

Agent	Company	Product
Amit	C1	Monitor
Amit	C1	Mic
Amit	C2	Speaker
Mukesh	C1	Speaker

We can decompose this table into table as R1(Agent,Company) R2(Agent,Product) ,R3(Company,Product)



Agent	Company
Amit	C1
Amit	C2
Mukesh	C1

Agent	Product
Amit	Monitor
Amit	Mic
Amit	Speaker
Mukesh	Speaker

Company	Product
C1	Monitor
C1	Mic
C2	Speaker
C1	Speaker

Agent	Company	Product
Amit	C1	Monitor
Amit	C1	Mic
Amit	C1	Speaker
Amit	C2	Monitor
Amit	C2	Mic
Amit	C2	Speaker
Mukesh	C1	Speaker

Agent	Company	Product
Amit	C1	Monitor
Amit	C1	Mic
Amit	C1	Speaker
Amit	C2	Speaker
Mukesh	C1	Speker

So relation is not in join dependency

Since , R1 ⋈ R2 ⋈R3 ≠ R

So, relation is in 5th Normal Form