- **Functional Dependency:** A formal tool for analysis of relational schemas. In a relation R, if 'X' $\subseteq$ R and Y $\subseteq$ R, then attribute or a set of attribute 'X' functionally-derives an attribute or set of attribute 'Y'.

For all pairs of tupples $t_1$ and $t_2$ in R such that,

R

| | X | Y |
|---|---|---|
| $T_1$ | a | 4 |
| $T_2$ | a | 4 |
| | b | 2 |

if $T_1[X] = T_1[Y]$

then, $T_1[Y] = T_2[Y]$

X is determinant which determines Y value,
Y is dependent (Dependent on X).

**Note:** A functional dependency is a property of the relation schema R, not of a particular legal relation state/instance r of R.

F.D $\longrightarrow$ data ✓

data $\longrightarrow$ F.D ✗

It simply means we can judge data from F.D. but F.D cannot be judged by data.

- **Trivial Functional dependency:** If Y is a subset of X, then the functional dependency $X \rightarrow Y$ will always holds.

e.g: R(A, B, C, D) $\boxed{Y \subseteq X}$

$\underset{X}{ABC} \longrightarrow \underset{Y}{AB}$ $AB \subseteq ABC$

Q. Consider the following relation instance, which of the following dependencies are satisfied by the above relational instance?

| A | B | C |
|---|---|---|
| 1 | 2 | 4 |
| 3 | 5 | 4 |
| 3 | 7 | 2 |
| 1 | 4 | 2 |

(a) $A \to B$, $BC \to A$

(b) $C \to B$, $CA \to B$

(c) $B \to C$, $AB \to C$

(d) $A \to C$, $BC \to A$

→ (a) $A \to B$

| A | B |
|---|---|
| 1 | 2 |
| 3 | 5 |
| 3 | 7 |
| 1 | 4 |

$\alpha \to \beta$

for same $\alpha$ getting different values of $\beta$.

∴ $A \to B$ F.D does not holds. ✗

$BC \to A$

| BC | A |
|---|---|
| 24 | 1 |
| 54 | 3 |
| 72 | 3 |
| 42 | 1 |

All values of $\alpha$ different

∴ $BC \to A$ F.D holds. ✓

(b) $C \to B$

| C | B |
|---|---|
| 4 | 2 |
| 4 | 5 |
| 2 | 7 |
| 2 | 4 |

for same $\alpha$ different $\beta$

✗

| CA | → B |
|---|---|
| 41 | 2 |
| 43 | 5 |
| 23 | 7 |
| 21 | 4 |

$\alpha$ is unique

✓

(c) $B \rightarrow C$

| B | C |
|---|---|
| 2 | 4 |
| 5 | 4 |
| 7 | 2 |
| 4 | 2 |

✓

$AB \rightarrow C$

| AB | C |
|---|---|
| 12 | 4 |
| 35 | 4 |
| 37 | 2 |
| 14 | 2 |

✓

(d) $A \rightarrow C$

| A | C |
|---|---|
| 1 | 4 |
| 3 | 4 |
| 3 | 2 |
| 1 | 2 |

✗

$BC \rightarrow A$

| BC | A |
|---|---|
| 24 | 1 |
| 54 | 3 |
| 72 | 3 |
| 42 | 1 |

✓

$\therefore$ (c) is correct ✓

Note: Funtional Dependency will always holds if :

1. $\alpha \rightarrow \beta$, where $\alpha$ is unique
2. $\alpha \rightarrow \beta$, where $\beta$ is having same value.

## Practice Question

Q1. Check F.D holds or not?

(a) $A \rightarrow B$ (b) $BC \rightarrow A$ (c) $B \rightarrow C$

(d) $AC \rightarrow B$

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 2 | 3 |
| 5 | 3 | 3 |

→ (a), (c), (d) holds

Q2. From the following instance R (A,B,C) we can conclude :

| A | B | C |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 2 | 3 | 2 |
| 2 | 3 | 2 |

(a) $A \rightarrow B$ , $B \rightarrow C$

(b) $A \rightarrow B$ , $B \not\rightarrow C$

(c) $B \not\rightarrow C$

(d) $A \not\rightarrow B$, $B \not\rightarrow C$

→ (b), (c) correct.

- Attribute Closure / Closure on Attribute set / Closure set :

→ Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from F denoted by $F^+$.

→ Set of all attributes functionally determined by X either directly from FD's or logically derived.

e.g: R (A,B,C)

$A \rightarrow B$

$B \rightarrow C$

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

$= A\underline{B}$

$= ABC$

$(B)^+ = \underline{B}$

$= BC$

$(C)^+ = \underline{C}$

Q. R (ABCDE)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

$(B)^+ = ?$

→ $(B)^+ = \underline{B}$

$= \underline{BD}$

Q. R (ABCDEF)

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

$(AB)^+ = ?$

→ $(AB)^+ = \underline{AB}$

$= A\underline{BC}$

$= ABC\underline{D}$

$= ABCDE$

Q. R (ABCDEFGH)

$A \rightarrow BC$

$CD \rightarrow E$

$E \rightarrow C$

$D \rightarrow AEH$

$ABH \rightarrow BD$

$DH \rightarrow BC$

$\boxed{BCD \rightarrow H}$ ?

$\rightarrow$ $(BCD)^+ = B\underline{C}\underline{D}$

$= BC\underline{D}E$

$= ABCDEH$

$\therefore$ $BCD \rightarrow H$, True ✓

• Key: Which can find all tuples uniquely.

$\alpha \rightarrow \beta$

key

| $\alpha$ | $\beta$ |
|---|---|
| 1 | a |
| 2 | b |
| 3 | c |

$\rightarrow$ A key is an attribute or set of attribute using which we can find all other attribute that is the whole relation.

key $\rightarrow$ R
(relation)

• Super key: set of attributes using which we can identify each tuple uniquely is called Super key. that is the set of attribute used to differentiate each tuple of a relation.

→ A relation of 'n' attributes with every attributes being a super key, then there are total $2^n - 1$ super keys.

e.g: R (A, B, C)

$A \rightarrow BC$

$B \rightarrow AC$

$C \rightarrow AB$

$(A)^+ = ABC$, $(B)^+ = ABC$, $(C)^+ = ABC$

$= R$ $\quad = R$ $\quad = R$

As, A, B, C are able to find the whole relation ~~each~~ using functional dependency.

∴ A, B, C are Super key.

Total attributes in the relation = n and all the attributes are S.k.

Combining S.k with any attribute of R will also be a S.k.

Total Combination (Total S.k)

A
B
C
AB
BC
AC
ABC

$$2^n - 1 = 2^3 - 1 = 7$$

→ Biggest Super key possible in a relation, is a set comprising all attributes of a relation.

e.g: R(ABCD), No functional dependency then super key?

→ If no dependency given then we need to assume a super key comprising of all attribute in a relation.

$$(ABCD)^+ = ABCD$$

↑ S.K

→ Let X be a set of attributes in a relation R, if $X^+$ (closure of X) determines all attributes of R then X is said to be super key of R.

Q. R(ABCD) Find Super key?

$ABC \rightarrow D$

$AB \rightarrow CD$

$A \rightarrow BCD$

→ $(A)^+ = ABCD$ ✓

$(AB)^+ = ABCD$ ✓

$(ABC)^+ = ABCD$ ✓

$(ABCD)^+ = ABCD$ ✓

∴ A, AB, ABC, ABCD are S.K

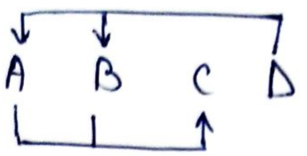
Q. R (ABCD)

$AB \rightarrow CD$

$D \rightarrow ABC$

Find S.k?

→




$(AB)^+ = ABCD$

$(ABC)^+ = ABCD$

$(ABCD)^+ = ABCD$

$(D)^+ = ABCD$

$(AD)^+ = ABCD$

$(ABD)^+ = ABCD$

~~AB~~

$(BD)^+ = ABCD$

$(BCD)^+ = ABCD$

$(CD)^+ = ABCD$

∴ Possible S.k's are: AB, ABC, ABCD, D, AD, ABD, BD, BCD, CD

Q. The maximum no. of super keys for the relation schema R (E, F, G, H) with E as the key?

→ E is key, hence superset of E will be S.K.

∴ Total possible S.K are = 8

E
EF
EH
EFG
EFH
EFGH
EGH
EG

Q. A super key for an entity consists of:
(a) One Attribute only (b) atleast two attribute
(c) at most two attribute (d) One or more attribute

→ (d) ✓

- Candidate key:

→ Minimum set of attributes that differentiates the tuple of a relation. No proper subset as ~~super~~ Candidate key also called a Minimal Super key

$\boxed{\text{Candidate key} = \text{Minimal S.k}}$

**Note:** All candidate key is super key. But all SK are not C.K.

→ There should be atleast one candidate key with not null constraint.

→ <u>Prime attribute</u>: Attributes that are member of candidate keys are called Prime attributes.

→ <u>Non-Prime attribute</u>: Attributes that are not a member of C.K are called Non-Prime attribute.

→ In a relation there are n attribute, then maximum candidate key will be n and minimum C.K will be 1.

Q. R(A, B, C, D) maximum candidate key?

→ No dependency given.

$$(ABCD)^+ = ABCD = R$$

∴ ABCD is super key and it is minimal.

∴ ABCD is C.K for the given relation.

Hence, max C.K = 1.

Q. R(ABCD) maximum C.K?

$A \rightarrow BCD$

$B \rightarrow ACD$

$C \rightarrow ABD$

$D \rightarrow ABC$

→ $(A)^+ = ABCD = R$

$(B)^+ = ABCD = R$

$(C)^+ = ABCD = R$

$(D)^+ = ABCD = R$

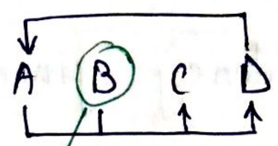
S.K = A, B, C, D

as these are minimal, C.K = A, B, C, D = 4

Q. R(ABCD) Find C.K.

$AB \rightarrow CD$

$D \rightarrow A$



Essential Attribute (no incoming edge)

$(B)^+ = B$ ✗

→ $(AB)^+ = ABCD$ ✓

$(BC)^+ = BC$ ✗

$(BD)^+ = ABCD$ ✓

∴ AB and BD are C.K as these are minimal S.K

Q. R = {ABCDE}

F = {A → BC, CD → E, B → D, E → A}

Find C.K.

→ A, E are C.K.

Q. R (ABCDEF)

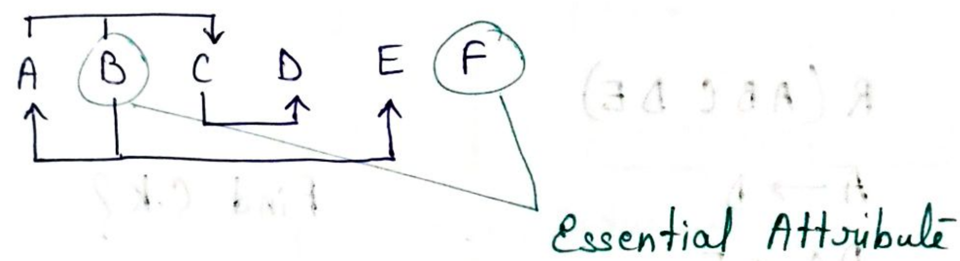
AB → C

C → D

B → AE

Find C.k ?

→




$(BF)^+ = BF$

$= ABEF$

$= ABCEF$

$= ABCDEF$ ✓

$(ABF)^+ = ABCDEF$ ✓

~~(CF)+ = CBDX~~

BF is minimal S.K

∴ BF is candidate key.

## Practice Questions

**Q1.** R (A B C D E F G H)

A → BC

ABE → CDGH

C → GD

D → G

E → F

Find C.K?

→ AE is C.K

**Q2.** R (A B C D E)

A → D

BC → E

DE → A

Find C.K?

→ ACD, BCD, ECD are C.K

**Q3.** R (A B C D E F G H)

CH → G

A → BC

B → CFH

E → A

F → EG

→ DE, DF, AD, BD are C.K

- Normalization of Data (Decomposition of Relation):

→ As one paragraph contains a single idea, similarly one table/Relation must contain an information about single idea, otherwise some values needs to be repeated for other values.

e.g: Student_Info

| USN | Name | Course_ID | Course_Name | Teacher |
|---|---|---|---|---|
| 1 | Ananya | C1 | DBMS | Shyama |
| 2 | Pruthvi | C1 | DBMS | Shyama |
| 3 | Sreemitha | C1 | DBMS | Shyama |
| | | | | |

Duplicacy / Redundancy

The above table/Relation comprises of two ~~relation~~ information, one is of student details and other is of course-details. which is a result of applying the Natural Join operation on two tables. The information of student appears only once which is completely fine. In contrast information of Course table ~~(Course-ID)~~, (Course-ID, Course_Name, Teacher) are repeated for every information of student table which is nothing but redundancy. And Redundancy leads to another serious problems such as Updation Anomalies, Inconsistancy etc.

→ Update Anomalies can be classified into insertion anomalies, deletion anomalies and modification anomalies.

① Insertion Anomalies: Insertion anomalies can be differentiated into two types, illustrated by the following example scenario:

Scenario 1:

To insert a new student we must know the course he/she enrolled. if student has not enrolled in any course we need to enter null ~~for~~ in all the attributes of course information. if student has enrolled in a course say C2 then we need to enter the course information C2 correctly otherwise it may lead to data inconsistency.

Scenario 2:

If we want to insert a new course then it will be impossible to insert if no student has enrolled in that course. As USN is the primary key in the table and we know primary key cannot be null.

② ***Deletion Anomalies:***

If we ~~want~~ delete all the students information then course details will also be deleted.

③ ***Modification Anomalies:***

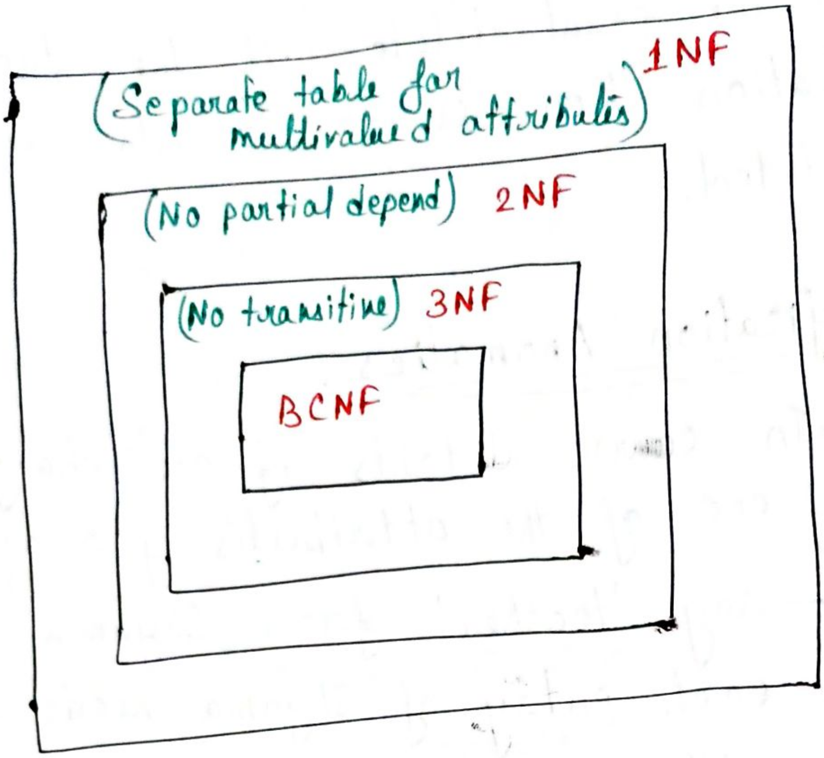
In course details if we change the value of one of the attributes of a particular course - say 'teacher' from Shyama to XYZ then in each entry of Shyama needs to be update with XYZ otherwise it will create inconsistency.

To overcome these above listed issues we use Normalization.

Normalization of Data (Decomposition of Data) can be considered as a process of analyzing the given relation schemas based on their functional dependencies and primary keys to acheive the desirable properties of minimizing redundancy.

→ The tool we use for normalization is fd and CK. Using fd we can only normalize up to BCNF.

→ 1NF >>> 2NF >>> 3NF >>> BCNF




→ A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalized to any desire degree.

**
Note: Normalization increases no. of tables. Because of which storage space also increases. Retrieval of data becomes slow. But still normalization is encouraged because we can not afford inconsistancy or loss of data.

## • First Normal Form:

A relational table is said to be in first normal form iff each attribute in each cell have single value (atomic). Means a relation should not contain any multivalued or composite attributes.

Other implications of first normal form.

① Every row should be unique, that is no two rows have the same values of all the attributes.

② There must be a primary key.

③ Every column should have a unique name.

④ Order of row and column is irrelevant.

e.g:

Student

| USN | Name | Ph.Number |
|---|---|---|
| 1 | ABC | 55861, 19222 |
| 2 | PQR | 18292, 45862 |
| 3 | XYZ | 5858, 78916 |

This table is not in first normal form, as column Ph_number contain multiple value in a single cell.

An apparent solution is to introduce more columns:

Student

| USN | Name | Ph_no 1 | Ph_no 2 |
|---|---|---|---|
| 1 | ABC | 55861 | 19222 |
| 2 | PQR | 18292 | 45862 |
| 3 | XYZ | 5858 | 78916 |

→ An arbitrary and hence meaningless ordering has been introduced: why 55861 put into the Ph_no1 column rather than the ~~telephone~~ Ph_no2 column?

→ There's no reason why ~~customers~~ students could not have more than two phone numbers. So how many ~~phonum~~ ph_no (N) columns should be there?

→ It is not possible to search for a Phone number without searching an arbitrary number of columns.

→ Adding an extra ph_no. may require the table to be recognized.

→ An alternative design is to make two tables. By taking USN as foreign key of new table named Phone_details refering to Student table.

Student

P.K

| USN | Name |
|---|---|
| 1 | ABC |
| 2 | PQR |
| 3 | X.Y Z |

F.K Phone_details

| USN | Ph_no. |
|---|---|
| 1 | 55861 |
| 1 | 19222 |
| 2 | 18292 |
| 2 | 45862 |
| 3 | 5858 |
| 3 | 78916 |

- **Partial Dependency:**

When a non-prime attribute is dependent only on a part (proper subset) of candidate key then it is called partial dependency.

$$\boxed{\text{PRIME} \longrightarrow \text{NON-Prime}}$$

- **Total Dependency:**

When a non-prime attribute is dependent on the entire candidate key then it is called total dependency.

- **Second Normal Form:**

Relation R is in 2NF if

① R should be in 1NF

② R should not contain partial dependency. (that is every non-key column should be fully dependent upon candidate key).

Q. R (ABCD) Is the relation is in 2NF?

$AB \rightarrow D$

$A \rightarrow C$

→ A B C D

$(AB)^+ = ABCD = R = C.K$

$(BC)^+ = BC \neq C.K$

$(BD)^+ = BD \neq C.K$

$\underset{C.K}{AB} \rightarrow \underset{NP}{D}$ (Total Dependency) ✓

$\underset{P}{A} \rightarrow \underset{NP}{C}$ (Partial Dependency)

As one dependency is partial

∴ It is not in 2NF but in 1NF.

Q. Why partial dependency avoided / Conversion of 1NF to 2NF.

→ R (A, B, C)

$B \to C$

essential attribute

A B C

$(AB)^+ = ABC$

∴ AB is candidate key.

e.g.:

R

| A | B | C |
|---|---|---|
| a | 1 | X |
| b | 2 | Y |
| a | 3 | Z |
| c | 3 | Z |
| d | 3 | Z |
| e | 3 | Z |

Redundancy

$B \to C$

P NP (Partial)

As partial dependency present, redundancy is there.

Solution : As B being a prime attribute ~~find~~ determining C. we need to make B as foreign ~~primary~~ key of one table $R_1$ (A, B) and primary ~~foreign~~ key of other table $R_2$ (B, C).
Means decomposition of R into $R_1$ and $R_2$

$R_1$ → $R_2$

| A | B (F.K) |
|---|---|
| a | 1 |
| b | 2 |
| a | 3 |
| c | 3 |
| d | 3 |
| e | 3 |

| B (P.K) | C |
|---|---|
| 1 | x |
| 2 | y |
| 3 | z |

- **Third Normal form :**

Let R be the relational schema, it is said to be in 3NF iff :

① R should be in 2NF

③ It must not contain transitive dependency.

- **Transitive Dependency :**

A functional dependency from non-prime attribute to non-prime attribute is called Transitive Dependency.

$\boxed{NP \rightarrow NP}$

Q. R (A, B, C, D) Determine 3NF or not?

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

→ A B C D

Essential Attribute

$(A)^+ = ABCD = C.K$

$\underset{C.k}{A} \rightarrow \underset{NP}{B}$ (Transitive ✗)

$\underset{NP}{B} \rightarrow \underset{NP}{C}$ (Transitive ✓)

$\underset{NP}{C} \rightarrow \underset{NP}{D}$ (Transitive ✓)

The given relation is not in 3NF as transitive dependency present.

- **BCNF (Boyce CODD Normal form):**

A ~~relat~~ relational schema R is said to be BCNF if every function ~~F~~ from $\alpha \rightarrow \beta$, $\alpha$ must be a super key

$F: \alpha \rightarrow \beta$

super key                P/NP

Q. Determine the normal form:

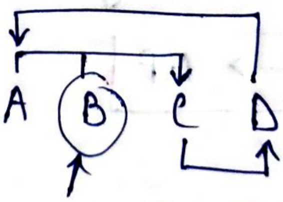
$R(A, B, C, D)$

$AB \rightarrow C$

$C \rightarrow D$

$D \rightarrow A$

$\rightarrow$




essential attribute

$(B)^+ = B$

$(AB)^+ = \underline{AB}\,\underline{C}\,D = C.K$

$(BC)^+ = ABCD = C.K$

$(BD)^+ = ABCD = C.K$

$AB \rightarrow C$ ✓ (Partial X, TD X)
S.K

$C \rightarrow D$ ✓ (Partial X, TD X)
P P

$D \rightarrow A$ ✓ (PD X, TD X)
P P

No partial dependency, transitive dependency present. Hence the given relation is in BCNF.

- Third Normal Form Direct Definition :

A relational schema R is said to be 3NF, if every functional dependency in R from $\alpha \rightarrow \beta$, either $\alpha$ is super key or $\beta$ is the prime attribute.

$$\alpha \rightarrow \beta$$

① if $\beta$ is prime attribute then it is in 3NF

$$\alpha \rightarrow \underset{(P)}{\beta}$$

② if $\alpha$ is a super key then it is in 3NF

$$\underset{(S.K)}{\alpha} \rightarrow \beta$$

Q. In RDBMS, if relation R is in BCNF, then which of the following is true about relation R?

1. R is in 4NF
2. R is not in 1NF
3. R is in 2NF and not in 3NF
4. R is in 2NF and 3NF

→ 4 ✓

Practice Question on Normal forms:

Q1. R (ABC) Assume AB, BC are keys.

AB → C

C → A

→ $\underset{C.K}{AB} \rightarrow C$ ∴ 2NF ✓ [∵ NO PD]

3NF ✓ [∵ NO TD]

BCNF ✓ [∵ C.k is determining]

$\underset{P}{C} \rightarrow \underset{P}{A}$ ∴ 2NF ✓ [∵ NO PD]

3NF ✓ [∵ NO TD]

BCNF ✓ [∵ β is prime attribute]

∴ R is in BCNF

Ans

Q2. R(ABCD) AB is key

AB → C

B → D

→

AB → C
C.K

2NF ✓ [∵ NO PD]

3NF ✓ [∵ NO TD]

BCNF ✓ [∵ C.k determining]

B → D
P NP

2NF ✗ [∵ PD]

3NF ✗ [∵ ~~NO TD~~ NOT 2NF]

BCNF ✗ [∵ Not 3NF]

∴ R is in 1NF
Ans

Q. R(ABCDE) (ABD) is key.

BD → E

A → C

→

BD → E
P NP

2NF ✗ [∵ PD ✓]

A → C
P NP

∴ R is in 2NF

- **Dependency Preservation Property of a Decomposition:**

We want to preserve the dependencies because each dependency in Relation represents a constraint on the database.

It is not necessary that the exact dependencies specified in F appear themselves in individual relations of the decomposition D. It is sufficient that the union of the dependencies that hold on the individual relations in D be equivalent to F.

**Definition:** Given a set of dependencies F on R, the projection of F on $R_i$ ~~~~ denoted by $\pi R_i(F)$ where $R_i$ is a subset of R, is the set of dependencies $X \rightarrow Y$ in $F^+$ such that the attributes in $X \rightarrow Y$ are all contained in $R_i$,

Hence, the projection of F on each relation schema $R_i$ in the decomposition D is the set of functional dependencies in $F^+$, the closure of F, such that all their left and right-hand-side attributes are in R. We say that a decomposition $D = \{R_1, R_2, \ldots R_m\}$ of R is dependency-preserving with respect to F if the union of the projections of F on each $R_i$ in D is equivalent to F;

$$(\pi R_1(F)) \cup \cdots \cup (\pi_{Rm}(F)))^+ = F^+$$

## Simple Definition

- **Dependency Preserving:**

If a relation / table R having FD set F, is decomposed into two tables $R_1$ and $R_2$ having FD set $F_1$ and $F_2$ then

$$F_1 \subseteq F^+$$

$$F_2 \subseteq F^+$$

$$\underline{(F_1 \cup F_2)^+ = F^+}$$

Q. R (A B C)

F: $A \rightarrow B$

$B \rightarrow C$

$C \rightarrow A$

$(A)^+ = ABC$

$(B)^+ = ABC$

$(C)^+ = ABC$

the relation is decomposed into two tables $R_1(AB)$ and $R_2(BC)$

→

| $R_1$ (AB) | $R_2$ (BC) |
|---|---|
| $F_1$: $A \rightarrow B$<br>$B \rightarrow A$ | $F_2$: $B \rightarrow C$<br>$C \rightarrow B$ |
| $(F_1 \cup F_2)^+ = F^+$<br>$A \rightarrow B$ ✓<br>$B \rightarrow C$ ✓<br>$C \rightarrow A$ ✓ | $C^+ = ABC$ |

∴ Dependency preserved.

Q. R(MNPQ)

f: $M \rightarrow N$

$P \rightarrow Q$

$R_1(MN)$ , $R_2(PQ)$

→

| R(MN) | R(PQ) |
|---|---|
| $(M)^+ = MN$ | $(P)^+ = PQ$ |
| $(N)^+ = N$ | $(Q)^+ = Q$ |
| $f_1: M \rightarrow N$ | $f_2: P \rightarrow Q$ |

$f_1 \cup f_2 = f$

$$\left.\begin{array}{l} M \rightarrow N \\ P \rightarrow Q \end{array}\right\} f$$

∴ Dependency preserved. in the decomposition.

# Lossy / Loss less - Dependency Preserving Decomposition:

→ Because of a normalization a table is Decomposed into two or more tables, but during this decomposition we must ensure satisfaction of some properties out of which the most important is loss less join property / loss less join decomposition.

→ If we decompose a table R into two tables. $R_1$ and $R_2$ because of normalization then at some later stage if we want to join (combine using natural join, or any other operation) the decomposed tables $R_1$ and $R_2$, then we must get back the original table R.

→ Decomposition is lossy if $R_1 \bowtie R_2 \supset R$

Decomposition is lossless if $R_1 \bowtie R_2 \subseteq R$

→ Decomposition is lossless if $R_1 \bowtie R_2 = R$ "The decomposition of relation R into $R_1$ and $R_2$ is loss less when the join of both table yield the same relation as in R".

→ This property is extremely critical and must be achieved at any cost. Decomposed tables should be lossless.

- **How to check ~~for~~ lossless join decomposition using FD set**

**Step 1 :** Union of attributes of $R_1$ and $R_2$ must be equal to attribute of R.

$$\boxed{Att(R_1) \cup Att(R_2) = Att(R)}$$

**Step 2 :** Intersection of Attributes $R_1$ and $R_2$ must not be NULL.

$$\boxed{Att(R_1) \cap Att(R_2) \neq \phi}$$

Common attribute must be a key for atleast one relation ($R_1$ or $R_2$)

i.e.,

$$\boxed{Att(R_1) \cap Att(R_2) \rightarrow Att(R_1)}$$

OR

$$\boxed{Att(R_1) \cap Att(R_2) \rightarrow Att(R_2)}$$

Q. Determine whether lossy / lossless decomposition.

$R(A\,B\,C)$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow A$

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

→ $R(A, B, C)$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow A$

$(A)^+ = ABC$

$(B)^+ = ABC$

$(C)^+ = ABC$

| $R_1\,(AB)$ | $R_2\,(BC)$ |
|---|---|
| $(A)^+ = AB$ | $(B)^+ = BC$ |
| $(B)^+ = AB$ | $(C)^+ = BC$ |
| $f_1 : A \longrightarrow B$ | $f_1 : B \longrightarrow C$ |
| $f_2 : B \longrightarrow A$ | $f_2 : C \longrightarrow B$ |

Common attribute in $R_1$ and $R_2$ = B

Now check whether B is key in any table.

| | |
|---|---|
| $(B)^+ = AB$<br>$= R_1$ | $(B)^+ = BC$<br>$= R_2$ |
| B is key in $R_1$ | B is key in $R_2$ as well. |

∴ Decomposition is lossless

Q: Determine Lossy / Lossless

X (PQRS)

$QR \rightarrow S$

$R \rightarrow P$

$S \rightarrow Q$

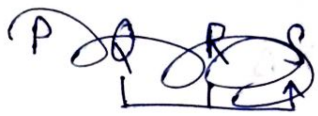
Decomposed into Y(PR) and Z (QRS).

→ X (PQRS)

$QR \rightarrow S$

$R \rightarrow P$

$S \rightarrow Q$




$(P)^+ = P$

$(R)^+ = PR$

$(S)^+ = SQ$

$(Q)^+ = Q$

$(QR)^+ = PQRS$

| Y (PR) | Z (QRS) |
|---|---|
| $(P)^+ = P$ | $(Q)^+ = Q$ |
| $(R)^+ = PR$ | $(R)^+ = R$ |
| | $(S)^+ = QS$ |
| $f_1: R \rightarrow P$ | $f_2: S \rightarrow Q$ |

Common attribute in Y and Z = R

Check R is key or not in any table.

| | |
|---|---|
| R is key Y as $(R)^+ = PR = Y$ | R is not key in Z as $(R)^+ = R \neq Z$ |

∵ Common attribute is key in one table

∴ lossless.