i.e, xtra Info is removed here & we can't recover it later.

3.) updation Anomaly :→

Simple query → [S-id-4] change name from Amrit to Amritpal.

```
update student
set Sname = 'Amrit pal'
  where S-Id = 4
```

→ Code.

No problem here.

Now,
If we want to change the salary of faculty F_1 from 30k to 40k.

[Change Salary of F_1 from 30k to 40k.]

Now, how many times the F_1 repeats in table, the same no. of times updation query runs & changes them all from 30 k to 40 k.

Note! → There is only 1 faculty F_1, then his salary must also changes 1 times. But, due to the column level duplicacy, it runs no. of times. Hence, It takes more time.

It is updation anomaly.

* Now, Normalizeⁿ removes Redundancy

How ?
→ a simple solⁿ may be, if we divide that table into multiple tables. Like,

P.K. | S-id | S-name |

P.K. | C-id | C-name |

P.K. | F-id | F-name | Salary. |

This can be 1 of the solⁿ.

→ Now, we don't get any anomaly in insertion, deletion & updaⁿ. There is no effect on others. Easy.

21. First Normal Form :→ (1 NF)

EF Codd → father of D.B.M.S

→ Table should not contain any multivalued attribute.

Student

Roll No.	Name	Course
1	Sai	C/C++
2	Anurag	Java
3	Onkar	C/DBMS

→ Not in 1st NF

[Null means not available]

Now, how to convert in 1st NF :-

1st way

Roll no.	Name	Course
1	Sai	C
1	Sai	C++
2	Anuray	Java
3	Onkar	C
3	Onkar	DBMS

primary key (P.K.) = Roll no. Course

(Combined, it is composite P.K.)

2nd way

Roll no.	Name	Course 1	Course 2
1	Sai	C	C++
2	Anurag	Java	Null
3	Onkar	C	DBMS.

[P.K. = Roll no.]

3rd way

Roll no	Name
1	Sai
2	Anurag
3	Onkar

Base Table

P.K. = Roll no.

F.K.

Roll no.	Course
1	C
1	C++
2	Java
3	C
3	DBMS.

Referencing Table

P.K. = Roll no Course

F.K. = Roll no.

(22) Closure Method :→

helps
↳ To find all the Candidate keys in the Table.

Ex:- Candidate key (C.K.)

R (ABCD).

FD { A → B , B → C , C → D }.

(Functional Dependency).

> Meaning of closure is that what 'A' can determine.

Here, A is determining B (from FD ①).

closure sign. ← $A^+ = B$

↳ $\boxed{A^+ = BCDA}$.

(A can determine itself also)
Ex:- Roll no. can determine itself.

A → B
B → C

Transitive.

Now, R (ABCD) has all 4 Attributes that are in $A^+ = BCDA$.

∴ A can determine all the attributes of Table.
This is the prop. of the Candidate key.

Now, $B^+ = BCD$

Here, B not determine A.

∴ B cannot be a C.K.

& $C^+ = CD$

$D^+ = D$.

Prime att. = {A}

Non prime att. = {B, C, D}.

∴ only A is C.K.

$\boxed{CK = \{A\}}$ Ans.

Note:- $(AB)^+ = ABCD$

(AB - itself
B → C
C → D.)

Here,
AB can be a C.K.
But,
It is not a C.K.
bcz,
C.K. is always minimal.
&
in AB only A is C.K.
& xtra is super key (S.K.)

$\underbrace{AB}_{\text{Super key}}$

(A ke saath adds any thing, becomes S.K.)

∴ AB is a Super key.

Ex:- R(ABCD).

FD = {A→B, B→C, C→D, D→A}.

$A^+ = ABCD$
$B^+ = BCDA$
$C^+ = CDAB$
$D^+ = DABC$

∴ C.K. = {A, B, C, D}
all 4.

Date :
Page: 43

prime Attribute :- are attribute which is used in making of the C.K.

So,

prime att. = {A, B, C, D} . { bcz, all are C.K. }

&

Non-prime att. = { ϕ } . → NULL .

Q :- R(ABCDE).

FD = {A → B, BC → D, E → C, D → A} .

⇒ Now, we have to check that which attributes are coming on the Right side. bcz, Attributes on the Right side, वो सब determine हो रहे हैं।

= BDCA . (E नहीं आया).

E = BDCAE

Note :- Each & Every candidate key must contain E. bcz, E if present on left side, then only it is written in Right side also.

जो Att. Right side में नहीं आ रहा। यानि वो left side में होना ही चाहिए। वो Candidate key बनाने में use होगा ही होगा।

Now,

$E^+ = EC$ → (E alone is not Candidate key but, it is used in making C.K.)

Now, Start :-

With A :-

$AE^+ = ABECD$ ∴ A is C.K.

$BE^+ = BECDA$

$CE^+ = CE$ ×

$DE^+ = DEABC$

$\boxed{C.K. : \{AE, BE, DE\}}$ Ans

★ Trick :- First, we got AE as C.K. So, check either (A or E) on the right side of FD.

FD = {A → B, BC → D, E → C, D → A}.

So, directly DE is also becomes the C.K.

now check at D, it is depend on 2. So, check ~~them~~ both

$\boxed{\text{prime Att.} = \{A, B, D, E\}}$ (used in making C.K.).

$\boxed{\text{non-prime att.} = \{C\}}$ Ans.

---

23. Functional Dependency ! → (F.D.) is the method which describes the rela'ship b/w the attributes.

X → Y

Determinant → X ; Y → Dependent Att.

X determines Y (or) Y is determined by X.

Ex!-

S-id		Sname
1	→	Ranjit
2	→	Ranjit

→ valid. } ∵ These 2 are diff.

Ex!-

1	→	Ranjit
1	→	Ranjit

} Same Student

Valid case

Ex:- 1 → Ranjit
2 → Varun } Valid.

Ex:- 1 → Ranjit
1 → Varun } Not Valid.

(A) F.D. are of 2 types:→

1.) Trivial F.D.
2.) Non-Trivial F.D.

1.) Trivial F.D.:→

If $x \to y$

then,

y is subset of x.

These Trivial F.D. are valid. (Always True).

Ex:- $\underset{x}{\text{Sid}} \to \underset{y}{\text{Sid}}$ ✓

NOTE:→ $x \to y$

L.HS ∩ RHS ≠ ϕ (Never Null).

Ex:- Sid Sname → Sid.

∩ Sid. ✓ Valid.

2.) Non-Trivial F.D.:→

If $x \to y$

Then,

y is not a subset of x.

i.e. $\boxed{X \cap Y = \phi}$ (NULL)

ex! -

Sid → S.name

Sid → phone no.

Sid → Loca"

Now, In this we have to check cases & find which is valid or not.

Properties of F.D! →

1.) **Reflexivity**: If Y be subset of X, then (Trivial)

$X \rightarrow Y$. (Sid → Sid).

2.) **Augmentation**: If $X \rightarrow Y$, then

$XZ \rightarrow YZ$

Sid → Sname

Sid phone → Sname phone.

3.) **Transitive**: If

$X \rightarrow Y$ & $Y \rightarrow Z$

then,

$X \rightarrow Z$.

Sid → Sname & Sname → City

Sid → City.

4.) **Union**:- If $X \rightarrow Y$ & $X \rightarrow Z$

then $X \rightarrow YZ$

5.) **Decomposition:** If

$X \rightarrow YZ$ then

$X \rightarrow Y$ and $X \rightarrow Z$.

But,

$XY \rightarrow Z$ ~~$X \rightarrow Z$ & $Y \rightarrow Z$~~ ✗

6.) **Pseudo Transitive:**

If

$X \rightarrow Y$ & $WY \rightarrow Z$ then,

$WX \rightarrow Z$

7.) **Composition:**

If $X \rightarrow Y$ & $Z \rightarrow W$

then

$XZ \rightarrow YW$.

(24.) 2nd Normal Form: → (2nd NF).

2 Rules

→ Table or Relaⁿ must be in 1st NF.

→ All the non-prime attributes should be fully functional dependent on Candidate key (C.K. or CP.K.) (or)

[There should be no partial dependency in the Relaⁿ].

Ex:

part of CK ← Non prime

AB
C.K.

A → C
(a part) (non-prime)

If it's partial depend
⇒ not 2nd NF.

Ex! →

Customer

Customer-Id	Store-Id	Locaⁿ
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

C.K. : Customer-Id Store Id

Prime attributes: C-Id
Store-Id.

Non prime: locaⁿ

Here, locaⁿ is only depend on Store-Id.
i.e. partial dependency.

bcz,
(to be in 2nd NF it should depend on the both C-id & S-id (bcz both are C.K.)).

Now →
Convert it into 2nd NF: →

Here, we make 2 Table.

C-Id	Store-Id
1	1
1	3
2	1
3	2
4.	3

Store-Id	Locaⁿ
1	Delhi
2	Bangalore
3	Mumbai

(here, it is fully dependent bcz only 1 C.K. is here).

2nd NF

Q:- R (ABCDEF)

FD: $\{C \to F,\ E \to A,\ EC \to D,\ A \to B\}$

Sol^n:- [C.K: ?]

First, check Right hand side

Step 1):- F A D B

means
(Now, these att are determined by some of the values.)

So,

[On LHS, there must be CE].

(C.K. जो भी बनेगी, उसमे CE तो होगा ही)

CE = FADB

Now.

$EC^+ = \underline{ECFADB}$ (all 6 are present)

∴, $\underline{EC \text{ is C.K.}}$

Now, use Trick
Either E or C must be present at the RHS of any F.D
but
not, neither E nor C, noone is present.

So,
there is only 1 C.K. in this table.

i.e, $\boxed{C.K. = \{EC\}}$

Step 1 is comp. here,
After finding $\underline{C.K. = \{EC\}}$

proof:- check

$A^+ = AB$
$B^+ = B$
$C^+ = CF$

∴, proved.

proper subset is always less than a set.

$X \subset XY$ → proper subset

$X \subseteq XY$ → subset

Step 2 :- prime Attributes : {E, C}

non-prime attributes : {A, B, D, F}.

Step 3 :- C.K. = {EC}

What is proper subset of EC

↳ Either 'E' or 'C'.

Now check :-

F.D. = {C → F , E → A , EC → D , A → B}.

~~(Either proper subset of C.K.)~~

for partial dependency i.e., not in 2nd NF

Order

→ Check on LHS → Either 'E' or 'C').

AND

check on RHS → whether it is non-prime att.

We get,

C → F

↳ partial dependency.

So,

Table is not in 2nd NF Ans.

C → F	{PD}	partial
E → A	{PD}	
EC → D	{FD}	fully.
A → B.	{FD}.	

→ 1 भी P.D. मिलता है, table is not in 2nd NF.

25. 3rd NF :→

→ Table or Relⁿ must be in 2nd NF.

&

→ There should be no transitive dependency in table.

Non prime as Non-unique
prime as unique.

Date :
Page:

→ not sufficient cond'n.

(NPA)

* Mean, Non-prime att. को कोई Non-prime att. determine नहीं कर सकता।

(C.K. & Prime att. (P.A.) तो कर सकते हैं determine NPA को।).

Ex:-

Rollno.	State	City
1	punjab	Mohali
2	Haryana	Ambala
3	punjab	Mohali
4	haryana	Ambala
5	Bihar	patna

C.K. = {Roll no.}
F.D → {Roll no → state,
state → city}

⇒ PA = {Roll no.}
⇒ NPA = {State, City}.

(NPA → NPA).

here, Roll → State and State → City.

It is Transitive dependency & we don't want that.

So,
It is not in 3rd NF.

Ex:-
R (ABCD)
FD : {AB → C, C → D}

⇒ C.K. = {AB}
PA = {A, B}
NPA = {C, D}

Transitive
AB^+ = ABCD.

C → D
NPA NPA

It is not in 3rd NF.

$\boxed{C.K. + \text{any thing} = S.K.}$

Super key

Date: ....
Page: ....

Ex → R (ABCD).

FD: (AB → CD, D → A).

Soln:

C.K. : {AB, DB}.
PA : {A, B, D}.
NPA : {C}.

(B not on RHS)

$B^+ = B$

$\boxed{AB^+ = ABCD}$

Now, A on RHS.
so,
$\boxed{DB^+ = DBAC}$
is also C.K.

Imp: Now, for each F.D

Then, Table is in 3rd NF.

LHS → C.K. or S.K.
OR
RHS → is a P.A.

→ check only 1 bcz OR.

FD: (AB → CD, D → A).

$\therefore$ $\boxed{\text{Table is in 3rd NF.}}$

bcz,

(NPA → NPA) is not present here.

---

26. BCNF. (Boyce Codd Normal Form) :→

⇒ also called as special case of 3rd NF.

~~LHS → we can also ... S.K. ... only ... C.K.~~

Student

Roll-no	Name	Voter-Id	Age
1	Ravi	K0123	20
2	Varun	M034	21
3	Ravi	K786	23
4	Rahul	D286	21

Table is in 3rd NF already.

C.K. = {Roll no, voter-Id}.

F.D. : = { Roll no → name
Roll no → voterid
voterid → age
voterid → Roll no. }

Note! → LHS of each FD should be C.K. or S.K.

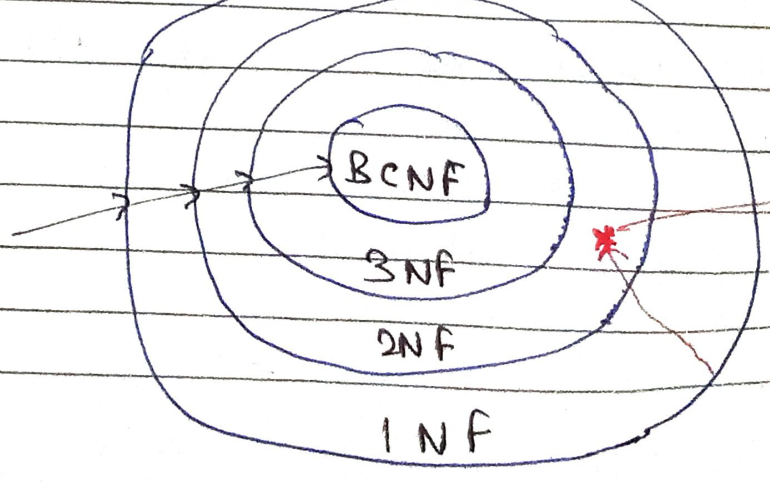
Here, 3NF की (OR) वाली condiⁿ थी हटा दी, जिसमें RHS के P.A. होने से भी काम चल जाता था).

Here,
we only want C.K. or S.K. in LHS, & RHS से कोई लेना देना नहीं।

Solⁿ! so, check all the F.D. one-by-one.
& in all the 4 F.D, the LHS is C.K.
so,

It is in BCNF form.

Compare :- 1NF, 2NF, 3NF, BCNF :→



It is outside 3NF & in 2NF & 1NF both.

19

1NF.

2NF = 1NF + condi^n
3NF = 2NF + condi^n
BCNF = 3NF + condi^n

---

27 Lossless & Lossy Join Decomposi^n :→

We normalise table → or we decompose table into 1NF, ... forms.

R

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

→ R_1 (AB)
→ R_2 (BC).

We divide this table into R_1 & R_2
& B is common in both the Table.

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

Q) find the value of C if the value of A='1'?

→ Now, for this we have to join R_1 & R_2 tables.

So,

Select $R_2.C$ from R_2 Natural Join R_1 where $R_1.A = '1'$.

(1st row को Multiply all rows से होगा of Table 2).

Cross product :- If R_1 has x rows & R_2 has y rows then their join has $x.y$ rows.

&

Condⁿ :- Common ~~element~~ colᵐ of both Tables (R_1 & R_2), here (B) has the same value in join Table.

&

Natural Join = Cross product + condⁿ.

Now,

R_1 A	R_1 B	R_2 B	R_2 C
1	2	2	1
1	2	2	2
~~1~~	~~2~~	~~3~~	~~2~~
2	2	2	1
2	2	2	2
~~2~~	~~2~~	~~3~~	~~2~~
~~3~~	~~3~~	~~2~~	~~1~~
~~3~~	~~3~~	~~2~~	~~2~~
3	3	3	2

Now. R'

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

→ table after Joining.

(take).

Spurious tuples.

Date :
Page:

→ In original Table (R), we have only 3 tuples (rows).
but,
after Joining, in R', we have 5 tuples.

(contd)
It is a flaw. It is called the Lossy Decomposition.

- Why Lossy :→
Here, we get 2 xtra rows, then, why lossy.
Here,
We don't talk about rows. We call it lossy because of inconsistency. There is a problem in Database.

⇒ In original, for A = 1, C = 1.
but
In join table, for A = 1 < C = 1, C = 2 }.

Why we get Error :→ (2 tuples more)

⇒ bcz, we take B as common in both Table.
but,

$\boxed{\text{Criteria for Common:→}}$ Common Attribute should be C.K. or S.K. of either R_1 or R_2 or both.
So, we have to C.K. or S.K. of original Table.

5) B has duplicacy in Table, We have to choose attri. 'A' for Right Ans.

bcz, A is unique. <1, 2, 3>.

R_1 ($\underline{A}B$)
R_2 ($\underline{A}C$).

→ We get 3 tuples also in joining table.

Cond.ⁿ for lossless Joining Decompos.ⁿ! →

1.) $R_1 \cup R_2 = R$.

$AB \cup AC = ABC$.

2.) $R_1 \cap R_2 \neq \phi$

$AB \cap AC$

$A \neq \phi$

3.) R_1 C.K. (or) R_2 C.K. (or) Both (C.K. of original Table)

To take common attribute ↗

(28.) All normal forms with Real life Examples →

6 forms here

1st NF	2nd NF	3rd NF.
→ No multivalued attribute. → only single valued.	⇒ In 1st NF + → No partial Dependency. * only full dependency. $\boxed{(AB) \rightarrow C}$ If A & B are 2 head of a Co., then both will use the Emplo. 'C'.	→ In 2nd NF. + → No Transitive dependency → No, NP A. should determine N.P.A.

BCNF	4th N.F.	5th N.F.
→ In 3rd NF + → LHS must be C.K. or S.K. $X \rightarrow Y$	→ In BCNF + → No multivalued Dependency. $X \rightarrow\rightarrow Y$	→ In 4th NF + → lossless Decomposi'n.

Varun → 3 Phone no.
→ 3 Mail Id.

(Varun depends on multiple att. i.e. phone & mail.)

table

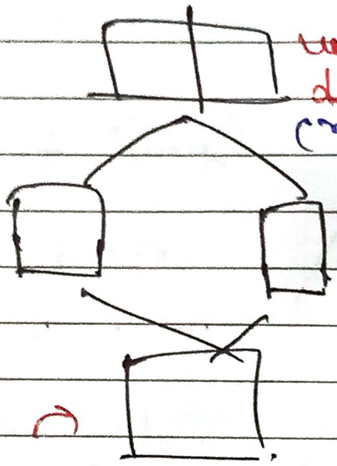
Varun	M_1	E_1
Varun	M_1	E_2
|	M_1	E_3
|	M_2	E_1
|	M_2	E_2
|	|	

→ Very long.
→ We also don't able to form a key.

So, Make 2 table.

[Varun — phone] [Varun — Mail]

(Now, no multivalued dependency).



vertically divide (rows same)

May be xtra tuples come. So, make C.K. as common attribute in both tables.

29. Minimal Cover :- (Irreducible) :-

Q:- For the following functional dependencies, find the correct Minimal Cover →

{A → B, C → B, D → ABC, AC → D}.

a) A → B , C → B , D → A , AC → D .
b) A → B , C → B , D → C , AC → D .
c) A → BC , D → CA , AC → D .
d) A → B , C → B , D → AC , AC → D   Ans

Sol^n: Our RHS in F.D. must be single.

Step 1: {A → B , C → B , D → ABC , AC → D}.

By decompos^n prop, separate them.

A → B , C → B , D → A , D → B , D → C , AC → D.

Step 2: Remove the Redundant F.D. :-

A → B , C → B , D → A , D → B , D → C , AC → D.

→ let, A → B को हटा दिया, Now check the closure of A,

$\boxed{A^+ = A}$ (not all +).

So, A → B is not Redundant. We can't remove it.

→ Same check this for every F.D. :-

D → B is redundant,

bcz let, ~~D → B~~ then, D^+ = DABC (all +).

So,

$\boxed{\text{remove, } D \to B}$

अब जब आगे check करेंगे, तो इसको नहीं लेंगे, इसको तो यहाँ ही remove कर दिया।

Date: ....
Page: ....

Now, for $AC \to D$ ~~×~~

$$AC^+ = ACB$$

So, also include $AC \to D$ ✓

Now, we get

$\{A \to B,\ C \to B,\ D \to A,\ D \to C,\ AC \to D\}$.

Step 3) Now, we only want 1 Att in LHS.

Here, $AC \to D$

Now, check by remove A & then check closure of C

$$C^+ = CB$$

- अगर C+ में हमें 'A' भी मिल जाता, तो हम A को हटा सकते थे। लेकिन नहीं मिला. तो,

$AC \to D$ ही रहेगा।

&

Same check + A, by removing C

&

$$A^+ = AB.$$

So, Can't remove C.

So, $AC \to D$ can't be reduced.

So,

$\{A \to B,\ C \to B,\ \boxed{D \to A,\ D \to C},\ AC \to D\}$.

compose

So,

$\boxed{\{A \to B,\ C \to B,\ D \to AC,\ AC \to D\}}$. Ans:

(30) Question on Normalization :→

Q! R (ABCDEF), check the highest normal form?

F.D. ! {AB → C, C → DE, E → F, F → A}.

Soln!- Step 1: Find all CKs in Relaⁿ !→

By closure Method !→

(B is not on RHS.) So, B compulsⁿ.

$B^+ = B$, $A^+ = A$

$AB^+ = ABCDEF$ (call +)

∴ $\boxed{\text{AB is C.K.}}$

Now, check ∀ A on RHS.

we get, F → A

∴ $\boxed{\text{FB is also CK.}}$

Now, check F on RHS !→

E → F.

So, $\boxed{\text{EB is also C.K.}}$

Now, check E on RHS.

C → DE.

So, $\boxed{\text{CB is also C.K.}}$

Now check ∀ C on RHS.

$AB \rightarrow C$.

&

AB is already our C.K.

So, we get all C.K.

$C.K = \{AB, FB, EB, CB\}$ ∴ 4 CK.

Step 2 :- Write all prime att. & NPA :→

$P.A = \{A, B, C, E, F\}$.

$NPA = \{D\}$.

Step 3 :→ Now, check FD :→

$\{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A\}$.

Now, check 1-by-1.

Highest NF :- 1NF, 2NF, 3NF, BCNF,

Redundancy decreases

* mean, when table is in BCNF, then redundancy is lowest. & in 1NF redundancy is highest.

* So, to check highest NF, we start from BCNF :→

→ In BCNF, we know, all LHS of all FD's should be CK or SK.

{ AB → C, | C → DE, | E → F, | F → A }

✓ | × | × | ×

It is not in BCNF form.

→ Now, 3NF :→

Check :→ transitive dependency (NPA → NPA) ×

$\boxed{\begin{array}{c} \text{LHS} \to \text{C.K. or S.K.} \\ \text{OR} \\ \text{RHS} \to \text{is a P.A.} \end{array}}$ → then it is 3NF.

Now, 1st Condi^n is already checked in BCNF.
So, here check only 2nd condi^n that whether RHS is P.A. or not.

	{ AB → C	C → DE	E → F	F → A }
		×	✓	✓
BCNF	✓	×	×	×
3NF	✓	×	✓	✓

∴ not in 3NF :→

→ Now, 2NF :→

Same thing → if there is already a tick in 3NF, then we don't have to check that for 2NF. वो already 2NF भी होगा ही। Check only for '×' tick.

Check! →

$\boxed{\begin{array}{l}\text{LHS is proper subset of C.K.} \\ \text{RHS is non-prime att.}\end{array}}$

↓

for partial dependency i.e,
if true then not in 2nd NF.

	AB → C	C → DE	E → F	F → A.
BCNF	✓	✗	✗	✗
3NF	✓	✗	✓	✓
2NF	✓	✗ ↳ (bcz both cond'n true). ↳ not in 2NF.	✓	✓
1st NF	✓	✓	✓	✓

↳ $\boxed{\text{Ans is 1st NF.}}$ Ans

bcz
→ for 1st NF'd that we don't want any multivalued attribute in the table. All att must be single (atomic).
& in

Table → R (ABCDEF)

all are general att',

we can't tell them as atomic or multivalued by seeing them.

↳

Table already 1st NF में ही दिखती है।
↳ (assume already).

↳

$\boxed{\text{1st NF.}}$ Ans

proper Subset is always less than a set.

31:- Find Out Normal Form of a Relo^n :→
(from 1NF - BCNF).

Q:- R(ABCDEF),
FD's: {AB→C, C→D, C→E, E→F, F→A}

C.K. = {AB, FB, EB, CB}.

P.A. = {A, B, C, E, F}
NPA = {D}

Sol^n:- Now, let we assume that R(ABCDEF) is already in 1st NF.

→ Check → 2nd NF :→ (partial depen) P.D.

Cond^n ⇒ { LHS must be proper Subset of any C.K. [and] RHS must be a Non-P.A. }

F.D's {AB→C, C→D, C→E, E→F, F→A}

AB→C	C→D	C→E	E→F	F→A
F & F	T & T	T & F	T & F	T & F
FD	PD	FD	FD	FD

(Full dependency).

∴ not in 2nd NF.

→ Make it in 2nd NF :→
We do it by decompose the Table
(Divide into 2 parts).

Date: ....
Page: ....

Cond^n! → Common attribute must be C.K.

1.) Lossless Decomposi^n
2.) Dependency should be preserved.

Now, यहाँ problem है यहाँ. $C \rightarrow D$ उठाकर Alag करे हैं, Alag Table बना हैं।

Now.

R (ABCDEF)

R_1 (ABEF) $\quad$ R_2 (CD)

Now, we have to make a common att. b/w these 2 table! –

Criteria ∀ common! – can be C.K. of any R_1 & R_2.

So, in R_2 (CD)

$C \rightarrow D$ $\quad$ $C^+ = CD$ (all 2)

∴ C is C.K.

So, make 'C' common in both R_1 & R_2.

Now, again

R (ABCDEF).

R_1 (ABCEF) $\quad$ R_2 (CD)

R_1 $\{AB \rightarrow C, C \rightarrow E, E \rightarrow F, F \rightarrow A\}$

↓

2nd NF

R_2 $\{C \rightarrow D\}$

& F.D. T | C.K. is C & C itself is not its proper subset

↓

2nd NF.

Now:
→ Check 4 3NF: →

$$\left\{ \begin{array}{l} \text{LHS must be C.K.} \\ \quad\quad \text{[or]}. \\ \text{RHS [redacted] be P.A.} \end{array} \right\} \rightarrow \text{3NF.}$$

So,

R_1 (ABCEF)

$\{AB \to C,\ C \to E,\ E \to F,\ F \to A\}$ ✓ ✓ ✓ ✓

C.K. = {AB, FB, EB, CB}
PA = {A, B, C, E, F}.

∴ → It is 3rd NF.

R_2 (CD)

$\{C \to D\}$ ✓

C.K. = {C}.
PA = {C}.

It is also 3rd NF.

Now:
→ Check 4 BCNF: →

$\boxed{\text{Condi}^n \rightarrow \text{L.H.S. must be a C.K.}}$

R_1 $\{AB \to C,\ C \to E,\ E \to F,\ F \to A\}$ ✓ × × ×

not in BCNF form.

→ There is Redundancy.

R_2 $\{C \to D\}$ ✓

It is BCNF form.

↓

Redundancy – 0%.

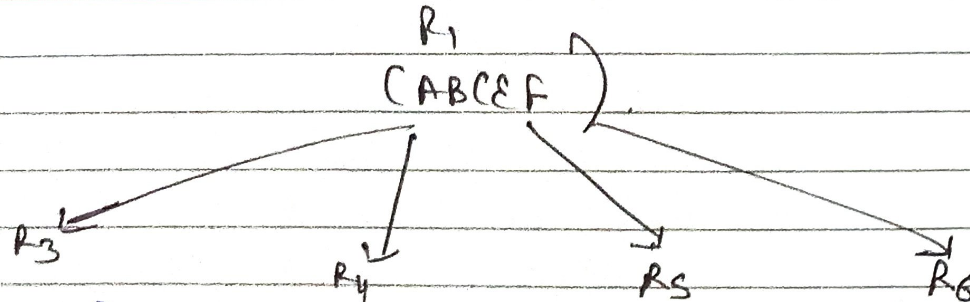
Now,
Again Decompose R_1 (ABCEF).

(ये problem नहीं है, उसको अलग check।)

X self its subset

Normalisation aim → To make redundancy 0%.

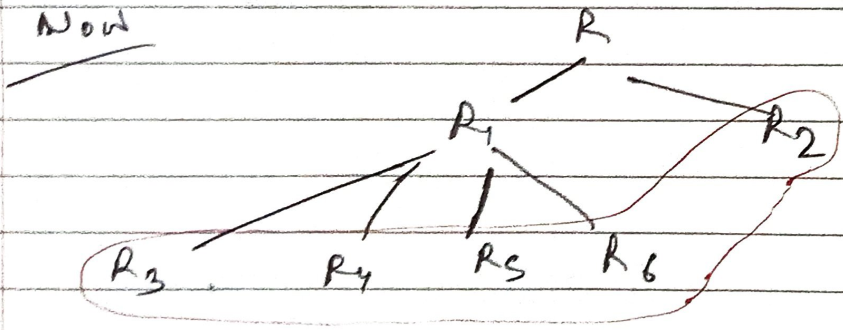
Date: decompose
Page: normalise



(AB→C)	(C→E)	(E→F)	(F→A)
(ABC)	(CE)	(EF)	(AF)
C.K.={AB}	C.K.={C}	C.K.={E}	C.K.={F}

Now, these all 4 tables have C.K.
so,
these all 4 are in BCNF now
& (कोई-ना-कोई common भी है हैं सब में)

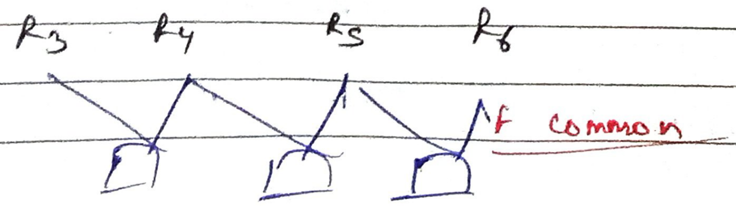
Now



Now, In all these 5 tables,
Redundancy is 0%.

- But, there are now multiple tables & If we join them again, it is Complex.

Cond. have to Join R3 & R4 (both have common 'C')



→ But, Join R_3 & R_5.
(ABC) (EF). ↳ Bcz nothing is common

We can't join them directly. So, take help from R_4.

→ So, Combine R_3 & R_4, let, R_3R_4 (ABCE)

Now,
We can combine it with R_5 bcz both have now 'E' as common att.

$R_3R_4R_5$ (ABCEF),
↳ E is C.K. now

(32) Normalisation Questions :→

→ A relaⁿ is in BCNF, then it is in 2NF also

Q: A Relaⁿ R has 8 att. (ABCDEFGH)

F = { CH → G, A → BC, B → CFH, E → A, F → EG }

how many candidate keys in R.
a) 3 ~~b)~~ 4 c) 5 d) 6.

Solⁿ: First, check on RHS, which att is absent.
D → so, now it comes with every.

$D^+ = D$ →

Date:
Page:

✓ $AD^+ = ADBCFHGE$ (all 8).

Now, check A on RHS,

so,

✓ ED^+ also a C.K.

Now, check E on RHS.

so,

✓ FD^+ also a C.K.

Now, check F on RHS.

so,

✓ BD^+ also a C.K.

Now, check B on RHS,

Now, completed,

so,

[AD, ED, FD, BD] 4 C.K.

(33.) Quesⁿ Explained on Normalisation :→

Schema is a structure of a Table.

→ so, [Scheme (or) Table] → same.

→ Non trivial F.D. means in which

[LHS ∩ RHS = ϕ] → we have to check that it is valid or not

→ Trivial F.D. are always valid.

Schema 1 : Registration (Roll. No, Courses)

Non trivial F.D of {roll no → Courses} :

Ans → We have to check that this table is in which form.

→ So, as always, start from higher form.

→ Check to BCNF! →
Condi'n : LHS of every FD must be C.K. or S.K. or P.K.

and,
Roll. no is already given as a C.K.
So,
LHS is a C.K. of F.D.
So,
Table is in BCNF form.
∴ also in 1st NF, 2nd NF & 3NF.

Schema 2 : Registra'n (roll no, Course id, email)

Non trivial FD of {roll no, Course id → email
email → roll no.}

Ans → C.K. = {roll no Course id}.
PA = {roll no, Co. Id}.
NPA = {email}.

→ Check BCNF! → 1st F.D. LHS is C.K.
but not in 2nd F.D.
So,
not in BCNF form.

Date:
Page:

→ Check for 3NF :-

Cond^n :-

LHS must be a CK or SK or PK
$\boxed{\text{OR}}$
RHS must be a P.A.

Now, 1st FD is already valid,
&
2nd FD: email → roll no.

P.A.
F T.
∴ T

It is in 3NF. ans.

Scheme 3 :- Registra^n (roll no, Co-Id, marks, grade).

Non trivial FD: {roll no, Co-id → marks, grade
marks → grade}.

Ans :- C.K. = {roll no C-Id }.
PA = {roll no, C-Id }
NPA = { marks, grade },

→ check of BCNF :-

1st FD → valid.
2nd FD → not valid.
∴
not in BCNF.

→ check of 3rd NF :-
1st FD → already valid
2nd FD → marks → grade.
F F ∴ not in 3rd NF.
ans.

→ check ∀ 2nd NF :→

Cond^n :- LHS must be a proper subset of CK.
[AND]
RHS must NPA.

↳ for P.D. → ie, not in 2nd NF if both cond^n are True.

→ 1st FD → already valid.
2nd FD → marks → grade
F T
→ F.

↳ not in partial Dependency
∴ It is in 2nd NF. Ans.

Schema 4 :→ Registr^n (roll no, C-Id, Credit).

Non trivial FD { roll no, C-Id → Credit
C-Id → Credit }.

As.→ C.K. = { roll no C-Id }
PA = { roll no, C-Id }.
NPA = { credit }.

→ check ∀ BCNF :→
1st FD → ✓
2nd FD → ✗
} ∴ not in BCNF.

→ Check ∀ 3NF :-
1st FD → ✓
2nd FD → C-Id → Credit
F F → F.

So, not in 3rd NF.

Date : ....
Page: ....

→ Check ∀ 2NF :→

1st FD → ✓
2nd FD → C-Id → Credit
T T
(+ T)

→ It is Partial Dependency (P.D.)
so,
It is not in 2nd NF. ✓

∴
→ Table is in 1st NF. ✓
(we already assume it).

———×———×———

34. Cover & Equivalence of F.D. :→

→ If X covers Y $Y \subseteq X$
If Y covers X $X \subseteq Y$
& if both are True.

then, $\boxed{X \equiv Y}$ → Equivalent.

Q:- $X = \{A \to B, B \to C\}$ | $Y = \{A \to B, B \to C, A \to C\}$.

i) X covers Y :→ $(Y \subseteq X)$

(इसमें Y की FD. में LHS का closure लेंगे लेकिन X वाले में से।)

Sol.

45

$A^+ = ABC$ $\qquad$ $B^+ = BC$.

$Y = \{A \to B,\ B \to C,\ A \to C\}$.

⇒ all 3 covers in these closure forms.

⇒ X covers Y.

ii) Y covers X :→

$X = \{A \to B,\ B \to C\}$.

$A^+ = ABC$ $\qquad$ (X का closure Y से लेंगे।)

$B^+ = BC$

⇒ Y also covers X.

⇒ both symbols cancels each other & became Equivalent.

$X \not\supseteq Y \qquad Y \not\supseteq X$

$\boxed{X \equiv Y}$ Ans.

Q:- $X = \{AB \to CD,\ B \to C,\ C \to D\}$.

$Y = \{AB \to C,\ AB \to D,\ C \to D\}$.

i) X covers Y :→

$Y = \{AB \to C,\ AB \to D,\ C \to D\}$.

$AB^+ = ABCD$, $\qquad C^+ = CD$ $\qquad$ ⇒ True

iii) Y covers X! →

$X = \{AB \to CD,\ B \to C,\ C \to D\}$.

✓ × ✓

$AB^+ = ABCD$, $B^+ = B$, $C^+ = CD$ (from Y).

∴ Y not covers X.

It becomes false.

∴ $\boxed{X \neq Y}$ Ans.

&

$\boxed{X \supseteq Y}$ & $\boxed{Y \supseteq X}$

✓, True   ×, false ✓.

35. Dependency Preserving Decomposition! →

Defin! → Let, any table $\boxed{R(ABCD)}$

& also some F.D., $\boxed{FD\ \{A \to B,\ B \to C\}}$

Now, we find closure & then find C.K., & then we also find hidden dependencies! →

like

$FD^+\ \{A \to C\}$ by Transitive prop.

and,

→ In normalisation, we do decomposition, then, we divide

R

$R_1(\ )$ $R_2(\ )$

& we distribute attri. of R in R_1 & R_2
&
union of att. of R_1 & R_2, must be equal to the att. of R.

Now, F.D. also divides, like

R

R_1( ) R_2( )

FD_1 FD_2

Now,

Check! - Dependency should be preserved,
(कहीं कोई F.D. Lost ना हो यदि हो, सबको preserve करके रखना है।)

ie, $\boxed{FD_1 \cup FD_2 = FD^+}$ (original F.D के Equal.)

Q! Let R(ABCD) with F.D.

$\{A \to B,\ B \to C,\ C \to D,\ D \to B\}$.

R is decomposed into R_1(AB), R_2(BC), R_3(B,D).

R_1(AB)	R_2(BC)	R_3(BD)
~~A → A~~	B → C ✓	B → D ✓
~~B → B~~	C → B ✓	D → B ✓
A → B ✓		
B → A. ✗		

Bcz ↓
$B^+ = BCD$
(don't have A).