Normalization:-

Student_info

S_id	Name	Age	Br_code	B_name	HoD_n
1	A	18	101	CS	XYZ
2	B	19	101	CS	XYZ
3	C	18	101	CS	XYZ
4	D	21	102	EC	PQR
5	E	20	102	EC	PQR
6	F	19	103	ME	KLM

⇒ Decompose

S_id	Name	Age	Br_code
1	A	18	101
2	B	19	101
3	C	18	101
4	D	21	102
5	E	20	102
6	F	19	103

Br_code	B_name	HoD_n
101	CS	XYZ
102	EC	PQR
103	ME	KLM

⇒ One table should contain a singe idea (logic.)

⇒ As one paragraph contains a single idea, similarly one table must contain direct & main data about an entity.

⇒ Normalization (Decomposition of tables) of table is done ~~of~~ the basic of functional dependencies

⇒ Normalization is a process which we use to remove redundancy, and to do normalization we will use functional dependency

1) 1NF (First Normal form):-

Roll_no	Name	Course
101	Modi	CN OS
102	Sonia	DBMS CO

X

⇒

Roll_no	Name	Course
101	Modi	CN
101	Modi	OS
102	Sonia	DBMS
102	Sonia	CO

✓

⇒ A table is said to be in 1NF if every cell contains atomic value.

⇒ We can't take a multivalued column in a table

⇒ We must have a single value in a single cell.

Idea :- In the table student-Info we have tried to store entire data about student

Result :- Entire branch data of a branch must be repeated for many students of the branch.

Student_Info

S-id	Name	age	Branch Code	B-name	HOD Name
1	A	18	101	CS	XYZ
2	B	19	101	CS	XYZ
3	C	18	101	CS	XYZ
4	D	21	102	EC	PQR
5	E	20	102	EC	PQR
6	F	19	103	ME	KLM

Redundancy :- When same data is stored multiple times unnecessarily in a database.

Disadvantages :- (i) Insertion, deletion and modification anomalies

(ii) Inconsistency (data)

(iii) Increase in database size and increase in time (slow)

→ Insertion Anomalies :- When certain data (attribute) cannot be inserted into data base without the presence of other data.

→ Deletion Anomalies :- If we delete some (unwanted) data, it cause some other data (wanted).

→ Updation / Modification Anomalies :- When we want to update a single piece of data, but it must be done at all of its copies

2NF :-

R(ABCD)

Functional Dependencies :-

AB → D

B → C

$(AB)^+ = ABCD$ ∴ AB is candidate key

⇒ A, B ∈ Prime attributes

Those attribute which are part of candidate key are prime attributes

⇒ C, D ∈ Non Prime attributes

Those attribute which are not a part of candidate key is called non-prime attribute

⇒ AB → D

It means D is dependent on both A and B which are the parts of Candidate key

⇒ B → C

Here C is dependent on only B, it is not dependent on both part of candidate key, which are A and B. This type of dependency is called partial dependency.

In other words when a non-prime attribute instead of depending on entire candidate key, it is depending on a part of candidate key, this is called Partial Dependency.

⇒ Primary key value can't be null, but when two attributes make a primary key then one attribute can be null

A	B	
1	-	✓
-	2	✓
-	-	✗
3	4	✓

⇒ In 2NF we have to remove partial dependency, because if non prime attribute is dependent on a part of candidate key, if that attribute value holds null then we can't find the non-prime attribute

eg B → C

Here C is dependent on B which is the part of a candidate key, if B holds null value we can't find C's value.

⇒ It means all the non-prime attributes should have dependent on the whole candidate key

⇒ **How to translate a table into 2nd Normal Form :-**

R(A B C D) $(AB)^+ = ABCD$

AB ← C.K

R_1 (A B D)

R_2 (B C)

⇒ Make different Tables from existing table.

⇒ In first table take candidate key.

⇒ With candidate key only those attributes are entitled to enter, those who entirely dependent on the candidate key.

B leader

R_1 (A B D)

⇒ So we can't enter C in first Table.

⇒ Now here in first table there is no partial dependency because D is dependent on both A & B.

⇒ Make another table, where part of that candidate key itself is a candidate key.

R_2 (B C)

⇒ Here C is dependent on prime attribute & there is not any partial dependency.

Example :-

R (A B C)

FD :- $B \rightarrow C$

$(AB)^+ = ABC$

A, B ← Prime

C ← Non Prime

$P \rightarrow NP$

$B \rightarrow C$ [Here B is a proper subset of Candidate key] So it is partial Dependency

R

A	B	C
a	1	x
b	2	y
a	3	z
c	3	z
d	3	z
e	3	z

R_1

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

R_2

B	C
1	x
2	y
3	z

R(A B C)

R_1(A B) R_2(B C)

Here redundancy is also removed

⇒ How to normalize or decompose a relation into 2NF

① R(A B C D E)

CD ⇒ $(ABD)^+ = R$

AB → C (PD)

D → E (PD)

— R_1(A B C)

— R_2(D E)

— R_3(A B D) (for candidate key)

② R(A B C D E)

$(AC)^+ = R$

A → B (PD)

B → E

C → D (PD)

— R_1(A B E)

— R_2(C D)

— R_3(A C)

③ R(A B C D E F G H I J)

AB → C – PD

AD → GH – PD

BD → EF – PD

A → I – PD

H → J

$(ABD)^+ = R$

— R_1(A B C)

— R_2(A I)

— R_3(A D G H J)

— R_4(B D E F)

— R_5(A B D)

✩ 3NF :-

Transitive Dependency :- A FD from $\alpha \to B$ is called transitive if $\alpha, B \in$ non-prime.

3NF :- A relation R is in 3NF if

(a) It is in 2NF

(b) No transitive Dependency

R(A B C)

A → B

B → C

— R_1(BC)

— R_2(AB)

A	B	C
a	1	x
b	1	x
c	1	x
d	2	y
e	2	y
f	3	z
g	3	z

R_2

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

R_1

B	C
1	x
2	y
3	z

⇒ Every dependency from $\alpha \to B$

(i) either α is superkey

(ii) or B is a prime attribute

PD: P → NP

TD: NP → NP

b [$\alpha \to B$] (P)

a [SK → ; $\alpha \to B$]

Scanned with CamScanner

→ How to decompose a table into 3NF :-

① $R(ABCDE)$

$A \rightarrow B$
$B \rightarrow E$
$C \rightarrow D$

→ $(AC)^+ = ABCDE = R$

∴ AC = Candidate key

- $R_1(ABE)$
  - $R_{1.1}(AB)$ ✓
  - $R_{1.2}(BE)$ ✓
- $R_2(CD)$ ✓
- $R_3(AC)$ ✓

② $R(ABCDEFGHIJ)$

$AB \rightarrow C$
$A \rightarrow DE$
$B \rightarrow F$
$F \rightarrow GH$
$D \rightarrow IJ$

$(AB)^+ = ABCDEFGHIJ = R$

AB is candidate key

- → $R_1(ADEIJ)$
  - $R_{1.1}(ADE)$ ✓
  - $R_{1.2}(DIJ)$ ✓
- $R_2(BFGH)$
  - $R_{2.1}(BF)$ ✓
  - $R_{2.2}(FGH)$ ✓
- $R_3(ABC)$ ✓

③ $R(ABCDE)$

$AB \rightarrow C$
$B \rightarrow D$
$D \rightarrow E$

$(AB)^+ = ABCDE = R$ → AB Candidate key

- $R_1(BDE)$
  - $R_{1.1}(BD)$
  - $R_{1.2}(DE)$
- $R_2(ABC)$

④ $R(ABCDEFGHIJ)$

$AB \rightarrow C$
$AD \rightarrow GH$
$BD \rightarrow EF$
$A \rightarrow I$
$H \rightarrow J$

$(ABD)^+ = ABCDEFGHIJ = R$

ABD is a candidate key

- $R_1(ABCI)$
  - $R_{1.1}(ABC)$ — 1
  - $R_{1.2}(AI)$ — 2
- $R_2(ADGHJ)$
  - $R_{2.1}(ADGH)$ — 3
  - $R_{2.2}(HJ)$ — 4
- $R_3(BDEF)$ — 5
- $R_4(ABD)$ — 6

★ BCNF :-

FD: $\alpha \rightarrow \beta$

P $\quad$ NP (Partial Dependency)

⇒ When α is a part of candidate key then it is called partial dependency. 2NF removes this logic; because it is not allowed in 2NF.

$\alpha \rightarrow \beta$

NP $\quad$ NP (Transitive Dependency)

⇒ When Non-prime attribute findes NP. then its is called transitive dependency. This is not allowed in 3NF

⇒ 3NF rejects both Partial dependency & Transitive Dependency

⇒ If there is FD: $\alpha \rightarrow \beta$

P/NP $\quad$ P

⇒ If Prime or Nonprime attribute discovers prime attribute this case is not covered in 2NF and in 3NF. This case is solved ~~by~~ in BCNF. BCNF deals with these types of cases.

eg. for BCNF case

R (A B C)

AB → C

C → B

NP $\quad$ P

Candidate key :-

$(A)^+$ — ✗

$(AB)^+ = ABC$ ✓

$(AC)^+ = ABC$ ✓

So CK are AB and AC

⇒ Table is in 2NF because there is not any partial dependency

⇒ Table is also in 3NF because there is not any transitive dependency, which will be only when NP attribute discovers a non prime attribute

⇒ Here exists a problem because NP attributes finds prime attribute, So this table is not in BCNF.

$\alpha \longrightarrow \beta$

P/ Non-prime $\quad$ Prime

⇒ To make a table into BCNF, Left side of FD should be superkey, irrespective the value of right side

$\alpha \rightarrow \beta$

Super key

8

R(ABC)

AB → C

C → B

Here Candidate Keys are AB, AC

A	B	C
a	1	x
b	2	y
c	2	z
c	3	w
d	3	w
e	3	w

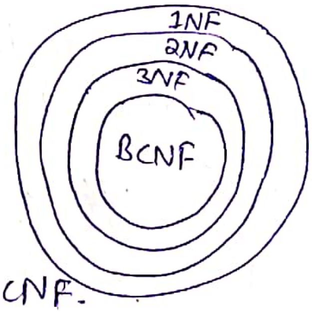
To be in BCNF in α → β, α should be super key

- R_1(C B) ✓
- R_2(A B C) ✗
- R_3(A C) ✓

A	C
a	x
b	y
c	z
c	w
d	w
e	w

C	B
x	1
y	2
z	2
w	3

BCNF is most restricted form of NF



⇒ Because only one CK can be Primary key

⇒ Some FD can be lost during decomposing in BCNF.

⇒ We should check for every FD that α is super key

⇒ Identify the Normal Form:-

① R(ABCDEFGH)

AB → C
A → DE
B → F
F → GH

BCNF ✗
3NF ✗
2NF ✗
1NF ✓

Candidate key = AB

② R(ABCDE)

CE → D
D → B
C → A

BCNF ✗
3NF ✗
2NF ✗
1NF ✓

CK = CE

③ R(ABCDEF)

AB → C
DC → AE
E → F

BCNF ✗
3NF ✗
2NF ✗
1NF ✓

CK = ABD, BCD

④ R(ABCDEFGHI)

AB → C
BD → EF
AD → GH
A → I

BCNF ✗
3NF ✗
2NF ✗
1NF ✓

CK = ABD

⑤ R(ABCDE)

AB → CD
D → A
BC → DE

BCNF ✗
3NF ✓
2NF

CK = AB, BD, BC

⑥ R(ABCDE)

BC → ADE
D → B

BCNF ✗
3NF ✓
2NF

CK = BC CD

⑦ R(ABCDEF)

ABC → D
ABD → E
CD → F
CDF → B
BF → D

BCNF ✗
3NF ✗
2NF ✗
1NF ✓

CK = ABC ACD

⑧ R(VWXYZ)

X → YV
Y → Z
Z → Y
VW → X

BCNF ✗
3NF ✗
2NF ✗
1NF ✓

CK = VW
XW

⑨ R(ABC)

A → B
B → C
C → A

BCNF ✓
3NF
2NF

CK = A, B, C