Module III

Relational DB design

=> Anomalies in DBMS =

Anomalies → faults.

1) Ins anomaly
2) Dlttion anomaly
3) Updation anomaly.

st id	st name	st age	Dept ID	Dpt name.
1	A	18	3	CS
2	B	20	4	Cs
3	C	30.	5	maths.

← seg-①

① ins { If we want to ins economics. dept. bt no. Students, that error

② dlt { If we want to dlt C stdnt. it will dlt entirely

③ update { update entirely, that errors.

ഇതിനെ മാറ്റാൻ solution → Normalization.
(errors ഒന്നും വരാതിരിക്കാൻ).

~~2) Anomaly~~

1) Ins anomaly =
It may not be possible to store certain info unless some other unrelated info is stored as null.

2) deletion anomaly =
It may not be possible to dlt certain info without loosing some other unrelated info as well.

(5) ABC → Y
eg full[illegible] ()
dependency work
[illegible] → fully FD.

(6) Set → ABC → Y
eg Subset → A B C
[illegible]
work [illegible] → partial FD

A → Y
B → Y
C → Y
any one

3) update anomaly =
If 1 copy of a related data is updated & inconsistency is created unless all copies are similarly updated.

eg - ①.

⇒ Functional Dependency = (FD)

* relationship b/w 1 attribute with another in a table.
* Denoted as →.
* eg →

	X	Y
t_1 →	10	5
	20	6
t_2 →	10	5

X → Y
10 → 5 ✓
20 → 6 ✓
10 → 5 ✗ (repeated) should be unique
x = 10

t → tuples.

$t_1 \cdot x = t_2 \cdot x$
$10 = 10$.
then,
$t_1 \cdot y = t_2 \cdot y$
$5 = 5$.
∴ () dependent

eg
$t_1 \cdot x = t_2 \cdot x$.
then
$t_1 \cdot y = t_2 \cdot y$
should satisfies. → () dependent.

10 → 5 ✓

* X & Y are attributes.
* X (Determinant)
* Y (dependent)
* X → Y.

→ Types of FD =

1) Trivial FD =

(2) Non-trivial FD =

* A dependent is always a subset of determinant
* eg → Rollno, name → name.

A dependent is strictly not a subset of determinant
eg → Rollno → name, age
([illegible])

(3) Multivalued FD =
* Entities of dependent set not dependent on each other.
* eg → Rollno → name, age

(4) Transitive FD =
A dependent is indirectly dependent on determinant
eg → X → Y, Y → Z
= X → Z.

(5) Fully FD =
* Y is said to be fully FD determined if it cannot be fully determinent by any of subset of X

(6) partial FD =
It can be any subset of Y.

⇒ Normalization =

errors ഉള്ള data-നെ normalise ചെയ്യുന്നു

Un normalized data
↓ → remove repeating grp.
1NF
↓ → remove partial dependencies
2 NF
↓ → remove transitive dependencies
3 NF
↓ → determinates are candidate key

NF → normalized form

BCNF → remove multivalued dependencies
↓
4 NF → remove join dependencies
↓
5 NF
↓
Normalized data ✓

def * Normalization is a database design technique that reduce data redendency & dependency & eliminates undesirable characteristics like ins, update & dlt anomalies normalisation rules divides larger tables into smaller table. & links them using relationship the purpose of normalisation in SQL into eliminate redundnt & ensure data is stored logically

1 Normal form. (1NF) =

Input – unnormalized data.
output – Relation with 1NF.

conditions =

1) Attributes should not hve multivalues, composite attributes.
2) Attributes values should be atomic (single)

eg → multivalued

Sid	Sname	course
101	ak	C, C++, Java
102	bk	Python, C++
103	ck	Java.

output

Sid	Sname	course
101	akshaya	C
101	akshaya	C++
101	akshaya	Java
102	bkshaya	Python
102	bkshaya	C++
103	ckshaya.	Java.

1NF

2 Normal form =

Input → Relation with 1NF
output → Relation with 2NF.

conditions =

* Relation should be in 1NF
* Relation should not hve partial FD (PFD)
* If in pfd then decompose the table. → (split the table)

eg →

sid	cid	c_fee.
101	Java	10,000
102	Py	20,00
101	C++	30,000
103	C	40,000
103	Java	50,000
102	JS	60,000

Check FD

Sid → cid ✗ $t_1.x = t_2.x$ then $t_1.y = t_2.y$.

Sid	cid
101	Java
102	Py
101	C++
103	C
103	Java
102	JS

repeat: 101 = 101, Java ≠ C++. So no FD.

like wise
102 = 102   Py ≠ JS.
103 = 103   C ≠ JS

Sid → cfee ✗
cid → cfee ✓ FD. $t_1.x = t_2.x$ then $t_1.y = t_2.y$ ✓

PFD or not

* Sid, cid → cfee.
  - Sid → cfee
  - cid → cfee ✓ — we found it is FD.
* Sid, cfee → cid
* cfee, cid → Sid

(PFD exists, so decompose)

Decompose

x → y

Sid	cid
101	Java
102	Py
101	C++
103	C
103	Java
102	JS

x → y.

cid	cfee
Java	10,000
Py	15,000
C++	2,000
C	10,000
Java	10,000
JS	20,000

III / * 3 Normal form = (3NF)

I/p → Relation with 2NF
O/p → " with 3NF.

Keys: Sid | name | adhar

all keys → candidate keys.
Sid → primary key

Condition,

1. No transitive dependencies for non-prime attribute. (2) (non-prime: candidate key other than key attribute)
3. If X → Y
   * X should be a primary key
   * Y " " a part of candidate key

eg →

S_id	S_course	fee
101	Java	20,000
102	C	10,000
103	C++	15,000
104	Py	25,000
105	Java	20,000
106	Py	25,000

Primary key → Sid.
candidate key → Sid   (course & fee → unique)

() dependencies –

Sid → course, fee.   Sid → course ✓, course → fee ✓
S_id → course
(1) ✓ (S_id → fee) ———→ ∴ Sid → fee.
S_id, course → fee
S_id, fee → course.
(2) ✓ (course → fee)   non-prime attribute
(3) ✓ X → Y. (X ↓ PK, Y ↓ not CK)   ∴ So 3 condi– not satisfied (✓).

So decompose the table

decompose

S_id	course
101	Java
102	C
103	C++
104	Py
105	Java
106	Py

course	fee
Java	20,000
C	10,000
C++	15,000
Py	25,000
~~Java~~	~~20,000~~
~~Py~~	~~25,000.~~

=> BCNF (Boyce Codd Normal Form) =

* 3.5 normal form. (b/w 3NF & 4NF)

conditions,

* Relation should be in 3NF
* For all () dependencies
  X → Y
  * should be candidate key (CK)

eg →

Rollno	name	voterid	age
1	ak	0123	20
2	bk	034	21
3	ck	036	23

CK → Rollno, voterid

X → Y

Rollno → name
Rollno → voter id
voter id → age
voter id → Roll no.

} all this condition satisfied.
∴ this table BCNF ✓

4 Normal form =

* It should satisfy BCNF
* It should not have multi-valued dependencies.

Conditions,

A → B_1, B_2, B_2

1. * A → B, for a single value of A, more than 1 value of B exist
2. * Table should be at least 3 colmns.
3. * For this table A, B, C colmns B & C should be indepndent.

eg →

(A) S_id	(B) course	(C) hobby
1	Sci	fb
1	Maths	cc
2	C++	fb
2	Php	cc.

① 1 → sci, maths.
2 → C++, Php.

② 3 colns.

③ B & C → Indpnt (no () dependencies)
B & C ബന്ധമില്ല.

①, ②, ③ } ഈ 3 condition satisfy ആണെങ്കിൽ we can decompose.

decompose

S_id	course
1	sci
1	maths
2	C++
2	Php

Course	hobby
Sci	fb
Maths	cc
C++	fb
Php	cc

VI 5th Normal form.

* Also → project normal form (PJNF).
* Similar to BCNF & 4NF except that join dependencies are used.

conditions:

* $(R_1, R_2 \dots R_n)$ is a trivial multivalued dependency.
* Every R_i is a superkey for schema R.

eg →

S	product	qty
AK	ABC	DK
BK	DEF	EK
CK	ghI	HK.

decomposition → SPC → SP, SC, PC.

SP

S	product
AK	ABC
BK	DEF
CK	ghI

SC

S	qty
AK	DK
BK	EK
CK	HK

PC

product	qty
ABC	DK
DEF	EK
ghI	HK.

S1