Module - 4

Anomaly

- Db anomaly is normally the flaw in db which occurs because of poor planning and storing everything in a flat db.
- Generally this is removed by the process of normalization which performed by splitting Joining of table
- Type of Anomalies
  1) Insert
  2) Delete
  3) Update.

Insert Anomaly

(cannot be added one attribute because absent of other attributes)

- An Insert Anomaly occurs when certain attributes cannot be inserted into the db without the presence of other attributes

- If a tuple is inserted in referencing relation ca referencing attribute value is not present in referenced attribute, then not allow inserting in referencing relation

StudentID	Enroll no	Roll no	name	addres	city	dep: ID	
1	AX001	1-BSCS-218	John	House 1	Lahore	3	
2	AX002	2	11	Faiz	house 2	Kasur	4
3.	AX003	3	11	Nouman	house 3	Faisalabad	5
4	AX004	4	11	Jerry	house 29	Jhang	6.

Dept ID	Name	Phone Extension
3	CS	398974
4	Botany	988784
5	English	898418.

Jerry is a

There is no

Hence, the

should be

only than

Delete anom

- It happen

information

be deleted a

- So to say

causes deletion

anomaly.

StudentID
2
3
4.

Jerry is a new student with department 6

There is no department with this dept ID 6.

Hence, the anomaly. The usual behaviour should be new department id with 6 and only then student could have it.

Delete Anomaly

- It happen when the deletion of unwanted information causes desired information to be deleted as well.

- So to say 'deletion of some attribute which causes deletion of other attributes is deletion anomaly.

Student ID	Enroll no	Roll no	Name	Dep ID
2	AX00 2	2 -	faiz	4
3	" 3	3	Neuman	5
4.	" 4	4	Jerry	6.

dep id	Name	phone
4	Botany	988-184
8	English	898418.

- Now if some one decides to delete cs department, he may end up deleting all students data who had the department of cs.

Update Anomaly

- This occurs in case of data redundancy and partial update.
- When duplicated data is updated at one instance and not across all instances where it was duplicated. That is an update Anomaly
- See below English department has now dept 108, but ~~unfortunate~~

Unfortunately it was not updated in student table.

Functional dependency

consider the schema: Following schema stores current student data in college hostels. (college has more than 1 hostel buildings)

Student_Hostel					
admno	studname	sex	dept	hostelname	roomno

- Since admno is a key

admno ⟶ {stud name, sex, dept, hostel name, room no}.

adm ⟶ stud name

admno ⟶ sex

admno ⟶ dept

adm no ⟶ hostel name

admno ⟶ room no.

Student – Hostel

admno	Stdname	Sex	dept	hostelname	roomno
A2003	ARav	m	CSE	Nila	101
A2033	Remya	F	EC	Yamuna	101
A2013	Soumya	F	CS	Ganga	102
A2025	Aneesh	m	CE	Nila	102
A2044	Rakesh	m	CE	Yamuna	102

→ hostel no , room no ⟶ adm no.

- FDs are additional constraints that can be specified by designers.

→ Functional dependency is trivial

$X \rightarrow Y$

if $Y \subseteq X$

eg. $A \rightarrow A$

SSN , PNO ⟶ P number.

hostel name , room no ⟶ room no

- The set of all dependencies that can be referred from the set of given Functional dependencies, say F is called Closure of F (F^+).

- These FDs can be deduced using some inference rules.

Rule 1: Re Flexivity

"A, B, C are set of 1 or more attributes"

If B is subset of A,

A determines B.

$A \longrightarrow B$

eg. Last name $\subseteq$ First name, Last name

First name, Last name $\longrightarrow$ Last name

Rule 2: Augmentation.

If $A \longrightarrow B$ and C is a set of attributes,

then AC determines BC:

$\{AC \longrightarrow BC\}$.

Eg:

Regno ⟶ Firstname, Lastname.

Regno, address ⟶ Firstname, Lastname, address.

Rule 3 : Transitivity

If A ⟶ B & B ⟶ C then A ⟶ C

is also true.

Eg.

Roll no ⟶ address and.

address ⟶ pincode

Roll no ⟶ pincode.

Rule 4: Union

If {A ⟶ B} {A ⟶ C} hold

A ⟶ BC true.

Eg:

Roll

Re

Ro

Rule

IF

Roll n

adm

Rule 6

IF

Rule 7

Eg :

Roll no ⟶ name

Roll no ⟶ address

Roll no ⟶ name, address.

Rule 5 : composition

IF $\{A \longrightarrow B\}$ $\{AX \longrightarrow BY\}$ is

$\{X \longrightarrow Y\}$ true.

Roll no ⟶ name Roll no, adm no ⟶

Adm no ⟶ address name, address

Rule 6 : Decomposition

IF $A \longrightarrow BC$ then

$A \longrightarrow B$

$A \longrightarrow C$ is true.

Rule 7 : Pseudo transitivity

$A \longrightarrow B$

$BC \longrightarrow D$ $\{AC \longrightarrow D\}$

Eg:

Roll no ⟶ name.

name, mark ⟶ percentage

Rollno, mark ⟶ percentage.

Armstrong's Axioms

- ~~It is that~~

Armstrong's Axioms is a inference rules like

- Reflexivity
- Augmentation
- Transitivity

? Consider relation $\in = (P, Q, R, S, T, U)$

having set of functional dependencies (FD).

$\{P \to Q,\ P \to R,\ QR \to S,\ Q \to T$
$QR \to V,\ PR \to U\}$

Show Using Inference rules that the Following dependencies can be Inferred From the above set.

1) $P \to T$

2) $PR \to S$

3) $QR \to SU$

4) $PR \to SU$

- $P \to T$

Given $P \to Q$

$Q \to T$.

$P \to T$ (1, 2, transitivity rule)

Hence, proved

→ $PR \rightarrow S$ Given

$P \rightarrow Q$

$QR \rightarrow S$

$PR \rightarrow S$ (Pseudo Transitivity)

Hence proved

→ $PR \rightarrow SU$ Given

$P \rightarrow Q$

$QR \rightarrow S$

$PR \rightarrow S$ (Pseudo transitivity)

$PR \rightarrow U$

$PR \rightarrow SU$ (union rule)

→ $QR \rightarrow SU$ Given

$QR \rightarrow S$

$QR \rightarrow U$

$QR \rightarrow SU$ (union rule)

Normalization

The process of organizing the data in db to avoid the redundancy and anomalies.

Normal forms (rules)

1nf
2nf
3nf
BCnf

1NF

Relation should not contain any multivalued attribute

Product

ProductID	Color	price
1	red, green	20
2	yellow	25
3	Green	10
4	red, blue	25

INF =>

Product id	Color	price.
1	red	20
1	green	20
2	yellow	25
3	green	10
4	red	10
4	blue	25

⇓

Product ID	color1	color2	price.
1	red	green	20
2	yellow	null	25
3	green	null	10
4	red	~~null~~ blue	25

1NF Decompose

ProductID	color

ProductID	price

2NF

Relation R is in 2nd normal form

- R should be in 1NF
- R should not contain any partial functional dependency.

Eg: R(ABCD)

FD : (A → B , C → D).

L.H.S candidate key (BD)(AC)

AC+ = ABCD

L.H.S / Prime attribute = AC (candidate key)

non prime attribute = BD.

Functional dependency

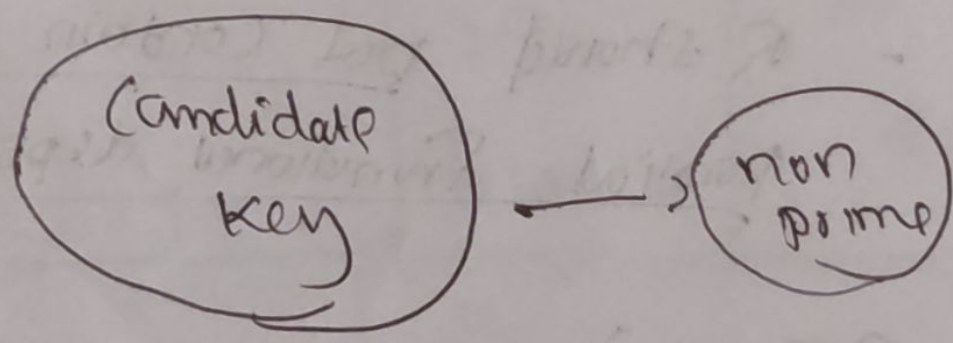
A → B C → D.

A → B

Candidate key part

non prime

partial functional dependance ⇒ NOT 2NF



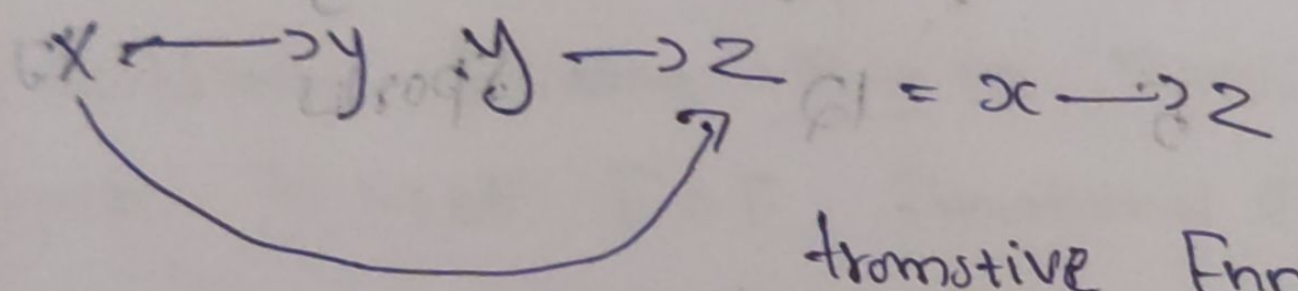
⇒ partial Functional dependency

3NF

3NF

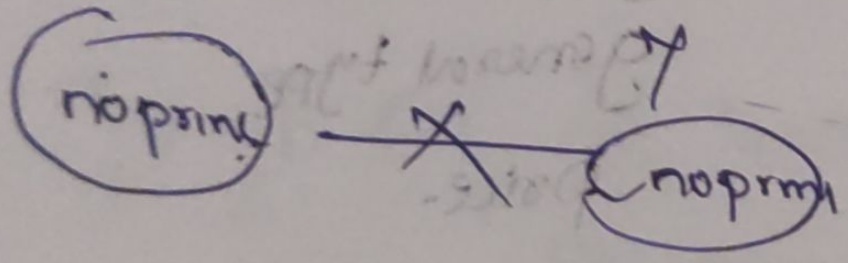
A table is in 3NF if it satisfy the following condetions

- It is in second normal form.
- There is no transtive Functional dependency.



transtive Functional dependency

- X



=> not correct

x ⟶ y

candidate key
Super key

or

prime

Eg : Books

BookID	GeneralID	Generaltype	price
1	11	physics	25
2	12	Sports	14
3	11	physics	10
4	13	Botany	12
5	12	Sports	15

Prime - Book id

non prime - generalID
general type
price.

Functional dependency

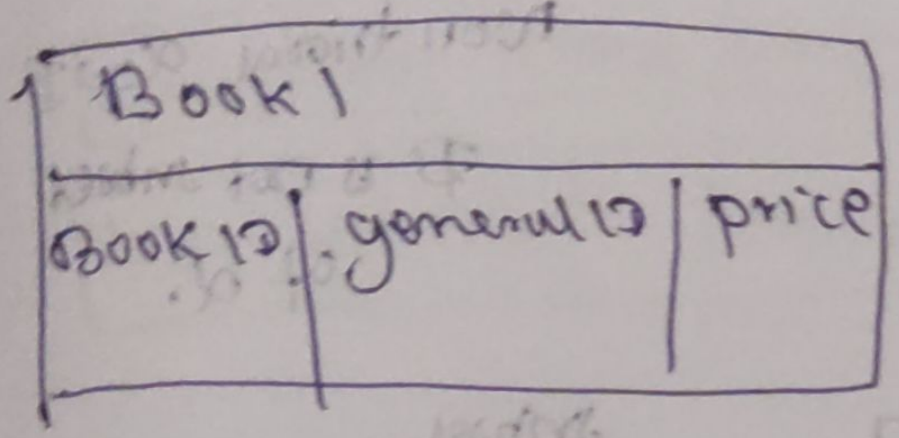
BookID → GeneralID

GeneralID → generaltype

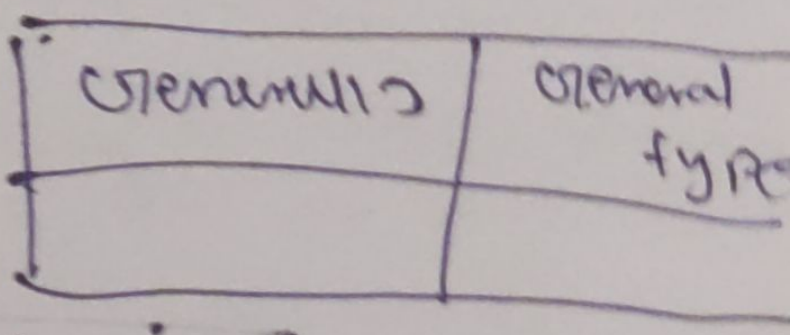
Prime
BookID → GeneralID (Satisfying)

GeneralID → General Type => no Satisfy

1) decomposi



Book 2.



BCNF:

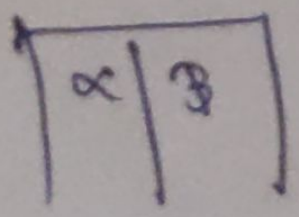
A relational Schema R is in BCNF with respect to a set F of functional dependency if for all functional dependencies the

are in the form of $\alpha \longrightarrow \beta$ where,

$\alpha, \beta \subseteq R$ at least one of the Following table.

- $\alpha \longrightarrow \beta$ is a trivial functional dependencies
- α is the superkey for all the schema R.

R



Fractional dependencies ← trivial $\beta \subseteq \alpha$

non trivial $\alpha \rightarrow \beta$

β is not subset of α.

Eg

A	B	C

subset ⇒ trivial

$AB \rightarrow A$

$\alpha \rightarrow \beta$

$AB \rightarrow B$

$AB \rightarrow AB$ } trivial

not subset

$A \rightarrow C$ ⟶ not trivial

→ A relation is in BCNF if every determinant is a candidate key

$A \rightarrow B$

determinat (candidate key) → dependent

2NF

(NF partial dependent (candidate key)

Eg: St

Student_no

Instructor.

INF

Student

Student

Instr

Comse

Com

? Eg: Student – Grade Report

Student_no	name	major	course no	course name

Instructor. no	Instructor name	Instructor Locat"	grade

1NF

Student no	Student name	major

Student no	course no	course name	Instructor no

Instructor name	Instructor Location	grade

2NF

(1NF
partially
dependent
(candidate key))

Student

Student no	Student name	major

course grade

Student no	course no	Grade

course Instructor

course no	course name	Instructor no	~~Instructor~~ name

Instructor location

3NF → transitive transitive

$x \to y$

$y \to z \Rightarrow x \to z$

Student

Student no	name	major

Course grade

Student no	Course no	Grade

course

Course no	Course name	Instructor no

Instructor

Instructor no	Instructor name	Instructor location

Dependency Preserving decomposition

- If we decompose a relation R into relations R_1 and R_2

- All dependencies of R either
  - must be a part of R_1 and R_2
  - must be derivable from combination of FD's of R_1 and R_2.

Eg:

Relation R (A B C D)

F_D $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$

decomposed into to $R_1(ABC)$ & $R_2(AD)$

- This is dependency preserving decomposition

① $A \rightarrow B$ and $A \rightarrow C$ are a part of $R_1(ABC)$ and

② $A \rightarrow D$ is part of $R_2(AD)$

Lossless join

If we decompose a relation R into Relations R_1 and R_2

- Decomposition is lossless

  If $R_1 \bowtie R_2 = R$.

  ↓

  (Natural join)

- Decomposition is lossy

  $R_1 \bowtie R_2 \supset R$.

To check for lossless join decomposition using FD set following conditions must hold

① Union of Attribute of R_1 and R_2 must be equal to attribute R.
Each attribute of R must be either in R_1 or in R_2

$$Att(R_1) \cup Att(R_2) = Att(R)$$

② Intersection of Attributes of R_1 and R_2 must not be NULL

$$Att\ R_1 \cap Att\ R_2 \neq \emptyset$$

③ common attribute must be a key for at least one relation (R_1 or R_2)

✓ $Att(R_1) \cap Att(R_2) \longrightarrow Att(R_1)$

✓ $Att(R_1) \cap Att(R_2) \longrightarrow Att(R_2)$

Eg:

A relation $R(A,B,C,D)$ with FD set

$\{A \to BC,\ A \to D\}$ is decomposed into

$R_1(ABC)$ and $R_2(AD)$

① First condition holds true as

$$Att(R_1) \cup Att(R_2) = \underline{(ABC) \cup (AD)}$$
$$= \underline{(ABCD) = Att(R)}$$

② second condition holds true.

$$Att(R_1) \cap Att(R_2) = (ABC) \cap (AD) = A \neq \emptyset$$

- Third condition

$Att(R_1) \cap Att(R_2) = D$ is key

$R_1(ABC)$ $A \rightarrow BC$ is given

AND key of $R_2(AD)$ $A \rightarrow D$.

Eg: 1

Let $R = ABCDE$

$R_1 = AD$ / $R_2 = AB$, $R_3 = BE$, $R_4 = CDE$

and $R_5 = AE$

Let the Functional dependencies

$A \longrightarrow C$

$B \longrightarrow C$

$C \longrightarrow D$

$DE \longrightarrow C$

$CE \longrightarrow A$

Apply standard algorithm to test IF the decomposition of R into $\{R_1, \ldots R_5\}$ is a lossless join

decomposition or NOT.

A) $R = ABCD$, $R_1 = AD$, $R_2 = AB$, $R_3 = BE$

$R_4 = CDE$ and $R_5 = AE$

$\{A \to C , B \to C , C \to D , DE \to C , CE \to A\}$

	A(1)	B(2)	C(3)	D(4)	E(5)
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	~~b_{23}~~ b_{13}	~~b_{24}~~ a_4	b_{25}
R_3	~~b_{31}~~ a_1	a_2	~~b_{33}~~ b_{13}	~~b_{34}~~ a_4	a_5
R_4	~~b_{41}~~ a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	~~b_{53}~~ b_{13}	b_{54}	a_5

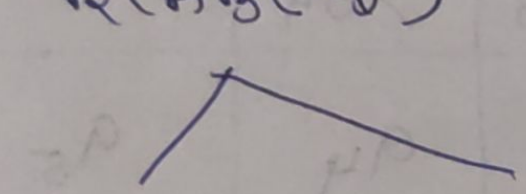
Unknown value = b

~~A → C~~ $R_3 = a_1 a_2 a_3 a_4$ → lossless_join

Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.

- There one mainly two type.
  - lossless Decomposition
  - lossy Decomposition. (not good)

R(ABCD)



lossless Decomposition

? Eg: given R(ABCD)

FD {AB → C , D → A} is decomposed to R_1(AC) & R_2(BCD) check whether it is lossless join or not

b) Given $R(A, B, C, D, E)$

Decomposed to $R_1(AB)$ $R_2(BCD)$, $R_3(CE)$

$f_D = \{AB \rightarrow CD,\ A \rightarrow E,\ C \rightarrow D\}$

① Initial value $S_{ij} = (b_{ij})$

	A	B	C	D	E
R_1	~~b_{11}~~ a_1	b_{12} a_2	b_{13} a_3	b_{14} a_4	b_{15}
R_2	b_{21}	~~b_{22}~~ a_2	~~b_{23}~~ a_3	~~b_{24}~~ a_4	b_{25}
R_3	b_{31}	b_{32}	~~b_{33}~~ a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5

① $f_D: \{AB \rightarrow CD\}$ => left side is not side.

$\{A \rightarrow E\}$ => not take any action.

$\{C \rightarrow D\}$ => Same value a_3 D value change a_4

Now look for any row with all a values.

Here there is no such row so we can conclude this is a lossy join.