Module-3 Database Design

M1 Requirements (Text)
↓
M1 logical design (ER Model)
↓
M2 physical design (Relational model)
↓
M3 Refinement (normalisation)

Stud

roll	name	mob	sub	faculty	Dept
1	Aadi	1234	DBMS	CRP	IT
			OS	MRN	IT
2	Aaditya	2345	DSS	MRS	MATH
1	Aadi	1234	DBMS	CRP	IT
26	Harshit	3456	ECO	AM	Humanity
177	Harshit	4567			

Anomalies → unwanted problems occurs due to data manupulation

→ updation Anomaly
→ Deletion "
→ Insertion "

→ updating a value at single instance causes data inconsistency.

→ updation anamoly occurs once we update the value in certain instances but not in all the instances of same copy.

→ Deletion anamoly occurs when we are trying to delete certain things, some other things are implicitly deleted.

→ Insertion anamoly occurs when we are entering few things instead of entering all.

⊛ Functional Dependency (FD):-

It is the relationship b/w two sets of attributes

$X \rightarrow Y$ → dependent

↓

determinant

$x \rightarrow y$ is a FD

if $\left.\begin{array}{l} x_1 \rightarrow y_1 \\ x_2 \rightarrow y_2 \end{array}\right\}$ if $x_1 = x_2 \Rightarrow y_1 = y_2$

For the same x same y should be there roll → name

mob → name

mob → roll

roll → mob

Faculty → dept

sub → faculty

Faculty → sub

{roll, name} → mob

roll → {name, mob}

⋮

Trivial FD → The FD is said to be trivial if $y \subseteq x$.

Non-trivial FD → The FD is said to be non-trivial if $y \not\subseteq x$.

* Armstrong's Axioms :-

Fundamental Axioms :-

i) Reflexivity ; if $y \subset x$ then $x \to y$

ii) Transitivity ; if $x \to y$ and $y \to z$ then $x \to z$

iii) Augmentation ; if $x \to y$, then $xz \to yz$

Secondary Axioms :-

i) union → If $x \to y$ and $x \to Z$ then $x \to yZ$

ii) Decomposition → If $x \to yZ$, then $x \to y$ and $x \to Z$

iii) composition → If $x \to y$ and $Z \to W$, then $xZ \to yW$

iv) pseudo transitivity → If $x \to y$ and $yZ \to W$ then $xZ \to W$.

Logical implication :-

Let, F ~~without~~ is a set of FD without including $x \to Z$

If F can generate $x \to Z$, then $F \models x \to Z$

- closure of Functional Dependency set

Let, F is a set of FD, the closure of F is represented as F^+ contains all the functional dependencies which can be generated.

$$F^+ = \{ x \to y \mid F \models x \to y \}$$

↳ logically implies

Ex :- R(A, B, C) it includes { A → B & B → C }

F⁺ = { A → B, B → C, A → C, AB → C, AB → A, A → A, B → B, BC → C, AC → A, AC → C, C → C, AB → B, BC → B, ABC → C, ABC → AB, AC → B, ABC → A, ABC → B, ABC → ABC, A → BC, ABC → BC, ABC → AC, BC → BC, A → ABC, AB → AC, AC → AC, AB → AB, AB → ABC .... }

Ex → R(A, B, C, D) it includes F = { A → BC, B → C, B → D }

F⁺ = { A → BC, B → C, B → D, A → B, A → C, B → CD, BC → D; BD → CD, A → D, A → DC, ABC → D, AB → BCD, AC → D..... }

Application of closure of FD set :-

1) It is used to check whether two functional dependency sets are equivalent or not

ii) Two functional dependency sets f and g are equivalent if and only if $f^+ = g^+$

$R(A,B,C,D)$ $F = \{A \to BC, A \to D, C \to D\}$

$G = \{A \to BC, C \to D\}$

$F^+ = \{A \to BC, A \to D, C \to D\}$

$G^+ = \{A \to BC, C \to D, \underbrace{A \to D}_{\text{decomposition + transitivity}}\}$

$\therefore F^+ = G^+$

$\therefore$ F & G are equivalent

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

$F_1 = \{A \to B, B \to C, D \to E\}$

$F_2 = \{A \to BC, D \to E\}$

$F_1^+ = \{A \to B, B \to C, D \to E, \underbrace{A \to BC}_{\text{union}}\}$

$F_2^+ = \{A \to BC, D \to E, A \to B, A \to C\}$

$F_1^+ \neq F_2^+$

$\therefore F_1$ & F_2 are not equivalent

Closure of Attribute set :-

F → set of FDs

X → set of attributes

X^+ → contains all the attributes that can be generated from X

Ex→ R(A, B, C, D)   F = {A→B, B→C, A→CD}

A^+ = {B, C, D, A}   {A, B}$^+$ or AB^+ = {A, B, C, D}

B^+ = {B, C} AC^+ = {A, B, C, D}

C^+ = {C} AD^+ = {A, B, C, D}

D^+ = {D} BC^+ = {B, C}

BD^+ = {B, C, D}

CD^+ = {C, D}

ABC^+ = {A, B, C, D}

ABD^+ = {A, B, C, D}

ACD^+ = {A, B, C, D}

BCD^+ = {B, C, D}

$ABCD^+$ = {A, B, C, D}

A is the Candidate key & are super keys

Applications of closure of Attribute set :-

1) To find the keys.

- If the closure of a set of attribute contains all the attributes of the reln. then that set of attribute is called as a key.

ii) If X^+ contains the Y then X determines Y $(X \to Y)$.

Ex→ $R(A,B,C,D,E)$; $F = \{A \to BC, B \to C, D \to E, C \to A\}$

Find all Candidate keys.

→ $A^+ = \{A, B, C\}$ $\quad$ $AD^+ = \{A,B,C,D,E\}$

$B^+ = \{B, C, A\}$ $\quad$ $BD^+ = \{A,B,C,D,E\}$

$C^+ = \{C, A, B\}$ $\quad$ $CD^+ = \{A,B,C,D,E\}$

$D^+ = \{D, E\}$ $\quad$ $AB^+ = \{A,B,C\}$

$AC^+ = \{A,B,C\}$

$E^+ = \{E\}$ $\quad$ $BC^+ = \{A,B,C\}$

Redundancy in FD :-

$R(A, B, C, D)$; $F = \{ A \to B, B \to C, \cancel{A \to C}, C \to D \}$

$A \to C$ is reduntant

Redundancy in RHS :-

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

$F = \{ A \to BC, B \to C, C \to D \} = \{ A \to B, B \to C, C \to D \}$

C is reduntant because

$A \to B, B \to C \Rightarrow A \to C$

Redundancy in LHS :-

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

$F = \{ A \to B, AB \to C, C \to D \}$

reduntant

$\left.\begin{array}{l} A \to A \\ A \to B \end{array}\right\} A \to AB$

$AB \to C$

$\therefore$ B is reduntant

Canonical Cover / Minimal Cover :-

Let, F is a set of FD

Canonical cover is represented as F_c.

The minimal cover contains the FDs of F by removing the redundancies.

F and F_c are equivalent

i.e. $F^+ \equiv F_c^+$

Let $R(A,B,C,D)$, $F = \{A \to BC, B \to C, A \to C, C \to D, B \to D\}$

step1 → Remove the redundant FD if it exists

step2 → Simplify or decompose the RHS part of the FDs.

If any redundant FD exists, remove it

step3 → If the LHS of a FD contains the redundancy remove it.

$F = \{A \to BC, B \to C, A \to C, C \to D, B \to D\}$

$= \{A \to B, A \to C, B \to C, \cancel{A \to C}, C \to D, B \to D\}$

$= \{A \to B, \cancel{A \to C}, B \to C, C \to D, B \to D\}$ ∵ $A \to B$ & $B \to C$ is given

$= \{A \to B, B \to C, C \to D, \cancel{B \to D}\}$ ∵ $B \to C$ & $C \to D$ is given

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

$$\boxed{F_c = \{A \to B, B \to C, C \to D\}}$$

* $R(A,B,C,D,E)$; $F = \{A \to BC,\ AB \to C,\ C \to DE,\ D \to E\}$

$F = \{A \to B,\ A \to C,\ AB \to C,\ C \to D,\ \cancel{C \to E},\ D \to E\}$

$= \{A \to B,\ A \to C,\ \cancel{AB \to C},\ C \to D,\ D \to E\}$

$A \to B$ & $A \to A$
$A \to AB$ $A \to C$
$AB \to C$

$F_c = \{A \to B,\ A \to C,\ C \to D,\ D \to E\}$

or $F_c = \{A \to BC,\ C \to D,\ D \to E\}$

---

* $R(A,B,C)$; $F = \{A \to B,\ B \to C,\ C \to A,\ B \to A,\ C \to B,\ \cancel{A \to C}\}$

$F = \{A \to B,\ B \to C,\ C \to A,\ \cancel{B \to A},\ C \to B\}$

$= \{A \to B,\ B \to C,\ C \to A,\ \cancel{C \to B}\}$

$F_c = \{A \to B,\ B \to C,\ C \to A\}$

or $F = \{\cancel{A \to B},\ B \to C,\ C \to A,\ B \to A,\ C \to B,\ A \to C\}$

$= \{B \to C,\ \cancel{C \to A},\ B \to A,\ C \to B,\ A \to C\}$

$= \{\cancel{B \to C},\ B \to A,\ C \to B,\ A \to C\}$

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

* For a same FD set there may be multiple canonical covers.

NORMALISATION

$R \to R_1, R_2 \ldots R_n$

Normalization is the process of decomposing or breaking a relational schema R into fragments (i.e. smaller schemas) $R_1, R_2 \ldots R_n$ such that following conditions hold -

- Lossless decomposition
- dependency preservation
- Good form (anomaly free)

Property → • Decomposition should be lossless (data should not be lost) [closure of attributes]

• Decomposition should be dependency preserving [closure of FD]

• Anomaly free

* Lossless Decomposition :-

The decomposition of $R \to R_1, R_2 \ldots R_n$ is lossless iff

$$\boxed{R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n = R}$$

- The decomposition of $R \to R_1, R_2 \cdots R_n$ is lossless iff the consecutive fragmented tables are related with primary key - foreign key relationship.

* $R(A, B, C, D)$, $F = \{A \to BC, B \to C, C \to D\}$

⇓

$R_1(A, B, C)$, $F_1 = \{A \to BC, B \to C\}$

$R_2(C, D)$, $F_2 = \{C \to D\}$

In R_1, $C^+ = \{C\}$

∴ C^+ is not containing all the attributes, so it can't be primary key.

In R_2, $C^+ = \{C, D\}$

It is primary key

∴ R_1 & R_2 can be Joined on C, and it is lossless

* $R(A, B, C, D, E)$, $F = \{A \to BC, B \to CD, A \to E\}$

$\Downarrow$

$R_1(A, B, C)$, $F_1 = \{A \to BC, B \to C\}$

$R_2(B, D)$, $F_2 = \{B \to D\}$

$R_3(A, E)$, $F_3 = \{A \to E\}$

In R_1, $B^+ = \{B, C\}$

In R_2, $B^+ = \{B, D\}$ w PK

~~In R_3, $C^+ = \{A, E\}$~~

R_1 & R_2 can be Joined on B

In R_1, $A^+ = \{A, B, C\}$

In R_3 $A^+ = \{A, E\}$ w PK

$\therefore$ R_1 & R_3 can be Joined on A

$\therefore$ Decomposition is losless.

(*) $R(A, B, C, D, E)$, $F = \{A \to BC, B \to C, D \to B, A \to E\}$

⇓

$R_1(A, B, C)$, $F_1 = \{A \to BC, B \to C\}$

$R_2(B, D)$, $F_2 = \{D \to B\}$

$R_3(A, E)$, $F_3 = \{A \to E\}$

$$\begin{array}{c} \\ B \\ \\ \end{array}\left[\begin{array}{l} R_1 \ (A, B, C) \\ R_2 \ (B, D) \\ R_3 \ (A, E) \end{array}\right] \begin{array}{c} \\ A \\ \\ \end{array}$$

In R_1, $B^+ = \{B, C\}$ ~~PK~~

In R_2, $B^+ = \{B\}$ ~~PK~~

$\therefore$ R_1 & R_2 can't be Joined

$\therefore$ Decomposition is lossy.

→ Eg

⊛ Lossless Join Algorithm :-

step 1 → create a table by considering the rows for each fragments and columns for all the attributes of the original rel^n

step 2 → Rename the columns as A_i

step 3 → Fill the cells :-

* If the attribute is present in the fragment, put a_i
* otherwise, put b_{ij}

step 4 → For each FDs, do the changes :-

* If one of the RHS is a_i, then make b_{ij} as a_i.
* otherwise, make the b_{ij} as same b_{ij}

step 4 → If any row contains all a_is, decomposition is lossless

(*) R(A, B, C, D) , F = { A → BC, B → C, C → D }

⇓

R_1 (A, B, C) , F_1 = { A → BC , B → C }

R_2 (C, D) , F_2 = { C → D }

Sol^n. →

	(A_1) A	(A_2) B	(A_3) C	(A_4) D
R_1	a_1	a_2	a_3	~~b_{14}~~ a_4
R_2	b_{21}	b_{22}	a_3	a_4

for A → B → No change

for A → C → No change

for B → C → No change

for C → D → b_{14} will be modified as a_4

∴ R_1 contains all a_is

∴ It is lossless

* $R(A, B, C, D, E)$; $F = \{A \to BC,\ B \to C,\ D \to B,\ A \to E\}$

⇓

$R_1 (A, B, C)$

$R_2 (B, D)$

$R_3 (A, E)$

	A (A_1)	B (A_2)	C (A_3)	D (A_4)	E (A_5)
R_1	a_1	a_2	a_3	b_{14}	$\cancel{b_{15}}\ a_5$
R_2	b_{21}	a_2	$\cancel{b_{23}}\ a_3$	a_4	b_{25}
R_3	a_1	$\cancel{b_{32}}\ a_2$	$\cancel{b_{33}}\ a_3$	b_{34}	a_5

for $A \to B$, change b_{32} to a_2

for $A \to C$, change b_{33} to a_3

for $B \to C$, change b_{23} to a_3

for $D \to B$, No change

for $A \to E$, change b_{15} to a_5

∴ It is lossy

Dependency Preservation :-

Decomposition of $R \to R_1, R_2 \dots R_n$ is dependency preserving iff

$$\cup F_i^+ = F^+$$

union of all the FDs and the original FD should be equivalent.

- $R(A,B,C)$; $F = \{A \to B,\ B \to C,\ A \to C\}$

$\Downarrow$

$R_1(A,B)$, $F_1 = \{A \to B\}$

$R_2(B,C)$, $F_2 = \{B \to C\}$

B

In R_1, $B^+ = \{B\}$ PK ✗

In R_2, $B^+ = \{B, C\}$ PK ✓

$\therefore$ R_1 & R_2 can be Joined on B

So, lossless

$(F_1 \cup F_2)^+ = \{A \to B,\ B \to C,\ A \to C\}$

$F^+ = \{A \to B,\ B \to C,\ A \to C\}$

$\therefore$ The decomposition is also dependency preserving.

• $R(A, B, C)$, $F = \{A \to B,\ B \to C,\ A \to C\}$

$R_1(A, B)$, $F_1 = \{A \to B\}$

A [$R_2(A, C)$, $F_2 = \{A \to C\}$

In R_1, $A^+ = \{A, B\}$ PK ✓

In R_2, $A^+ = \{A, C\}$ PK ✓ FK ✓

∴ It is lossless

$(F_1 \cup F_2)^+ = \{A \to B,\ A \to C\}$

$F^+ = \{A \to B,\ B \to C,\ A \to C\}$

∴ The decomposition is lossless but not dependency preserved.

⊛ Guidelines for Normalisation :-

1) Do not mix the details of multiple entity sets together. Make seperate table.

2) Design the database in such a way that all anomaly should be resolved.

3) Design the database in such a way that the null values should be properly treated.

4) Design the database in such a way that no spurious or extra record should be generated.

Advantages of Normalisation :-

i) It improves database design.

ii) It ensures minimum redundancy of data.

iii) It removes anomalies for database activities. It removes/solves Insertion, updation and deletion anomaly.

iv) This makes it easier to maintain in the database in a consistent state.

Normal Form :-

First Normal form (1NF)

The relⁿ R is in 1NF iff the values are single-valued and atomic

* Seperate rows are created for 1NF
* In 1NF none of the anomalies are removed

Golden Rule :-

If the ~~table contains~~ relational schema is provided without any data it is present in 1NF.

• Second Normal form (2NF) :-

The relⁿ R is present in 2NF iff

i) It is in 1NF

ii) All non-key attributes are depending on the key totally

T

Module	Dept	lect	Text
M_1	D_1	L_1	T_1
M_1	D_1	L_1	T_2
M_2	D_1	L_1	T_3
M_3	D_1	L_2	T_1
M_3	D_1	L_2	T_4
M_4	D_2	L_3	T_5
M_5	D_2	L_4	T_3
M_5	D_2	L_4	T_6

1NF ✓

2NF ✗

F = { Module → Dept,
Module → lect,
lect → dept }

× Module$^+$ = { module, Dept, lect }

× Dept$^+$ = { dept }

× lect$^+$ = { lect, dept }

× text$^+$ = { Text }

× { Module, dept }$^+$ = { module, dept, lect }

× { module, lect }$^+$ = { module, dept, lect }

✓ { module, Text }$^+$ = { module, dept, lect, text }

× { lect, dept }$^+$ = { lect, dept }

× { lect, text }$^+$ = { lect, dept, text }

× { dept, text }$^+$ = { dept, text }

Prime / Key attribute → Module, text

Non-Prime / Non-key " → Dept, lect

* Partial and Total FD

$x \rightarrow y$ is a partial FD if

$Z \subset X$ and $Z \rightarrow y$

whole
roll → city

{roll, name} → city
partial

key : {Module, Text}

key attr: module, text

Non-key " : dept, lect

lect & dept is depended partially

∴ It is not in 2NF

* How to convert non-2NF rel^n. to 2NF rel^n. :-

Rules:-

i) create a new table for the offending or problematic FD.

The determinant should be the key.

ii) The RHS of the problematic F.D. are removed from the original table

iii) If the same determinant appears in more than one problematic F.D.s, create a single table; do not create multiple tables

T_1

Module	Dept	lect
M_1	D_1	L_1
M_2	D_1	L_1
M_3	D_1	L_2
M_4	D_2	L_3
M_5	D_2	L_4

2NF ✓

T_2

Module	Text
M_1	T_1
M_1	T_2
M_2	T_3
M_2	T_1
M_3	T_4
M_3	T_5
M_4	T_3
M_5	T_6

2NF ✓

updation anomaly ✓
deletion " ✓
insertion " ✓

NOTE → If the key contains a single attribute, the 1NF relations are also present in 2NF.

* All anomalies are available but it is reduced in 2NF

* Decompose $R(A, B, C, D, E)$ with $F = \{A \to BC, B \to C, D \to E\}$ into 2NF.

Sol.: Since rel.ⁿ schema is provided, R is in 1NF.

Key : $\{A, D\}$

Key attributes : A, D

Non-key " : B, C, E

R

<u>A</u>	B	C	<u>D</u>	E

2NF ✗

R_1

<u>A</u>	B	C

$F_1 = \{A \to BC, B \to C\}$

2NF ✓

R_2

<u>A</u>	<u>D</u>	E

$F_2 = \{D \to E, AD \to AD\}$

R_3

<u>D</u>	E

$F_3 = \{D \to E\}$

2NF ✓

R_4

<u>A</u>	<u>D</u>

$F_4 = \{AD \to AD\}$

2NF ✓

After 2NF decomposition,

$$\begin{array}{l} R_1(\underline{A}, B, C) \\ R_3(\underline{D}, E) \\ R_4(\underline{A}, \underline{D}) \end{array}$$

A

D

∴ It is lossless

R(A, B, C, D, E) ; F = {A → BC, C → DE, D → E}

→ Since, relⁿ. schema is provided it is in 1NF

Key: A

Key-attr: A

Non-key ": B, C, D, E

∴ It is in 2NF, as Key contains a single attr.

Third Normal Form (3NF) :-

The relⁿ R is in 3NF iff

- It is in 2NF
- No non-key attribute is depending on the key transitively.

Module	dept	lect
M_1	D_1	L_1
M_2	D_1	L_1
M_3	D_1	L_2
M_4	D_2	L_3
M_5	D_2	L_4

module → lect
lect → dept

module → dept

transitively

$F_1 = \{$ module → dept, module → lect, lect → dept $\}$

key → non-key attr → non-key attr

2NF ✓
3NF ✗

* If a non-key – nonkey FD is present it is not in 3NF, as it is transitive.

After decomposing,

T_{11}

lect	dept
L_1	D_1
L_2	D_1
L_3	D_2
L_4	D_2

$F_{11} = \{ \text{lect} \rightarrow \text{dept} \}$

T_{12}

Module	lect
M_1	L_1
M_2	L_1
M_3	L_2
M_4	L_3
M_5	L_4

$F_{12} = \{ \text{module} \rightarrow \text{lect} \}$

All anomalies are resolved in 3NF.

i) Decomposition upto 3NF are always lossless and dependency preserving.

ii) As per the industry standard, the table should be present minimum in 3NF.

(*) $R(A, B, C, D, E)$, $F = \{ A \rightarrow BC, B \rightarrow C, D \rightarrow E \}$

Key: $\{A, D\}$

Key attr: A, D

Non-key attr: B, C, E

R: | <u>A</u> | B | C | <u>D</u> | E |

R_1: | <u>A</u> | B | C |

$F_1 = \{ A \rightarrow BC, B \rightarrow C \}$

2NF ✓

3NF ✗

R_5: | <u>B</u> | C |

$F_5 = \{ B \rightarrow C \}$

R_6: | <u>A</u> | B |

$F_6 = \{ A \rightarrow B \}$

R_2: | <u>A</u> | <u>D</u> | E |

R_3: | <u>D</u> | E |

$F_3 = \{ D \rightarrow E \}$

R_4: | <u>A</u> | <u>D</u> |

$F_4 = \{ AD \rightarrow AD \}$

$D \left[\begin{array}{ll} R_3(\underline{D}, E), & F_3 = \{D \to E\} \\ R_4(\underline{A}, \underline{D}), & F_4 = \{AD \to AD\} \end{array} \right.$

$B \left[\begin{array}{ll} R_5(\underline{B}, C), & F_5 = \{B \to C\} \\ R_6(\underline{A}, B), & F_6 = \{A \to B\} \end{array} \right.$

$(F_3 \cup F_4 \cup F_5 \cup F_6)^+ = \{D \to E,\ AD \to AD,\ B \to C,\ A \to B,\ A \to BC\}$

$F^+ = \{A \to BC,\ B \to C,\ D \to E,\ AD \to AD,\ A \to B\}$

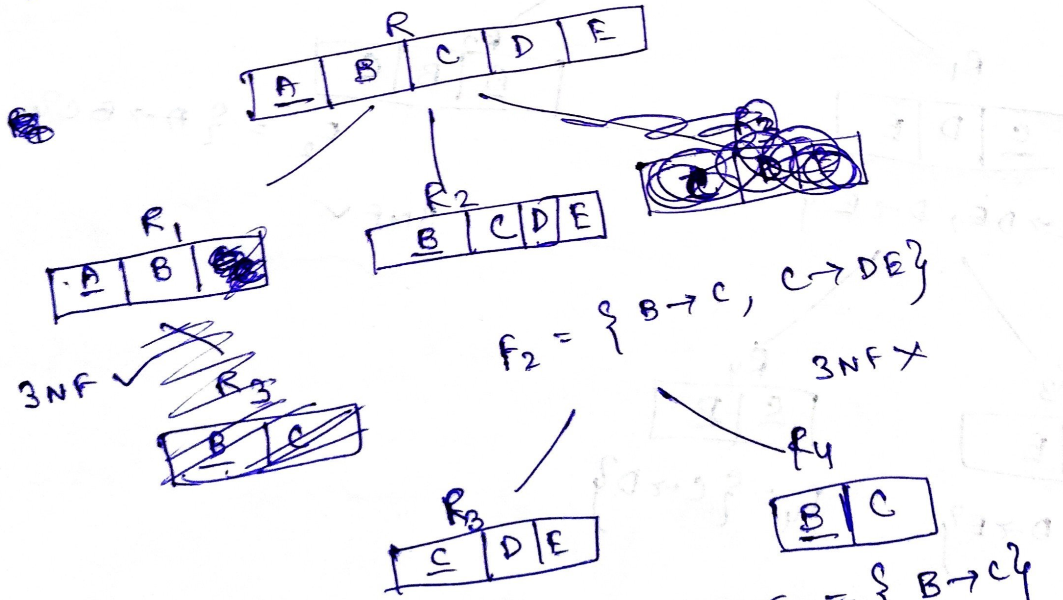
(*) $R(A, B, C, D, E)$, $F = \{A \to BC,\ B \to C,\ C \to DE\}$

Key: A

Key attr: A

Non-Key: B, C, D, E

$\therefore$ It is in 2NF



3NF ✓

$F_2 = \{B \to C,\ C \to DE\}$

3NF ✗

$F_3 = \{C \to DE\}$

3NF ✓

$F_4 = \{B \to C\}$

3NF ✓

* $R(A, B, C, D, E)$, $F = \{A \to BC, C \to DE, D \to E\}$

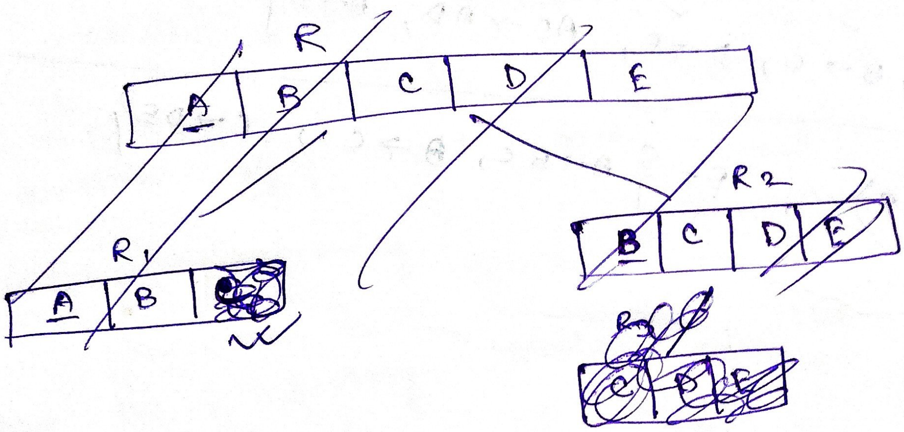
Decompose R into 3NF.

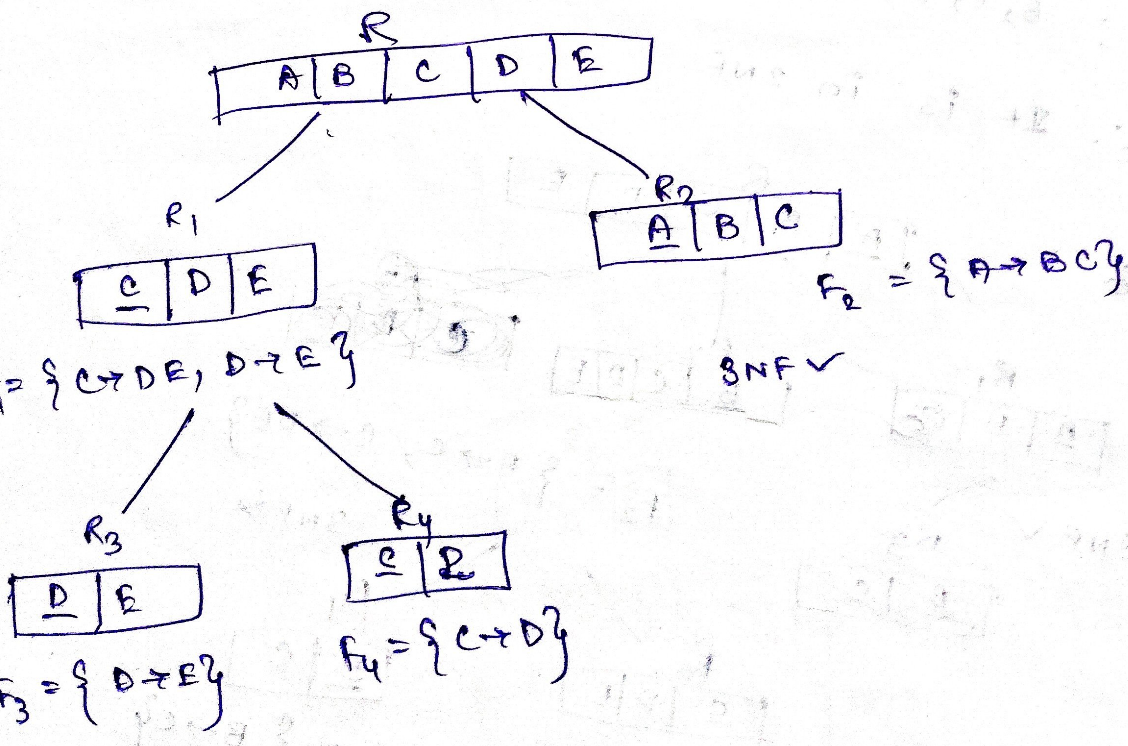
→ key : $\{A\}$

Key attributes : A

Non-key : B, C, D, E

It is in 2NF





After 3NF,

R_2 (A, B, C) , $F_2 = \{A \to BC\}$

R_3 (D, E) , $F_3 = \{D \to E\}$

R_4 (C, D) , $F_4 = \{C \to D\}$

∴ It is lossless

$(F_2 \cup F_3 \cup F_4)^+ = \{A \to BC, D \to E, C \to D, C \to DE\}$

$F^+ = \{A \to BC, C \to DE, D \to E, C \to D\}$

(*) T

Stud	Sub	Time
Prathaam	DBMS	8:00
Bhatt	DBMS	8:00
Navdeep	DBMS	10:00
Mishra	OS	11:00
Navdeep	OS	11:00

Key : {Stud, Sub} , {Stud , Time}

Key attributes : Stud, Sub, Time

non-key " : ϕ

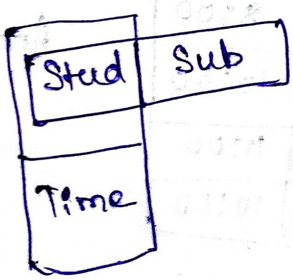
∴ It is in 2NF

There is no non-key - non-key relationship

∴ It is in 3NF.

$F = \{Time \to Sub,$
$\{Stud, Sub\} \to Time,$
$\{Stud, Time\} \to Sub\}$

* overlapping Key →



Boyce - Codd Normal Form (BCNF) :-

also called as 3.5NF

R is in BCNF iff

i) It is in 3NF

ii) for every non-trivial FD, determinant should be the key.

case - 1 :-

PK → {stud, sub}

T

Stud	Sub	Time

T_1

Time	sub

T_2

Stud	Time

F_1 = {Time → Sub}

key upgraded

case - 2 :-

PK → {stud, Time}

T

Stud	Sub	Time

BCNF ✗

F = {Time → Sub,
{Stud, Time} → Sub,
{Stud, Sub} → Time}

T_1

Time	Sub

T_2

Stud	time

F_1 = {time → Sub}

F_2 = {{Stud, Time} → Stud, time}

T_1

Time	Sub
8:00	DBMS
10:00	DBMS
11:00	OS

T_2

Stud	Time
Pratham	8:00
Bhatt	8:00
Navdeep	10:00
Mishra	11:00
Navdeep	10:00

Anomalies are resolved

$F_1 \cup F_2^+ = \{ Time \to sub, \{stud, Time\} \to \{stud, Time\}, \{stud, Time\} \to sub \}$

$F^+ = \{ Time \to Sub, \{stud, time\} \to sub, \{stud, sub\} \to time \}$

$\therefore$ $\{stud, sub\} \to time$ is lost

* Decomposition upto BCNF is always lossless but may not preserve the dependencies

Imp * If the table has only one key, the 3NF tables are also present in BCNF

(*) $R(A, B, C, D, E, G)$; $F = \{ AB \to CD, D \to EG, E \to G, AC \to BD \}$

Decompose R in BCNF.

$\Rightarrow$ Since schema is given, it is in 1NF

key : $\{AB, AC\}$

Key-attr : A, B, C

Non-key " : D, E, G

~~... is in 2NF~~

Let, us consider $PK \to \{A, B\}$

R

A	B	C	D	E	G

3NF ✗

$F = \{ AB \to CD,\ D \to EG,\ E \to G,\ AC \to BD \}$

2NF ✓

non-key rela^n is not broken

Mult

R1

D	E	G

$F_1 = \{ D \to EG,\ E \to G \}$

R2

A	B	C	D

~~3NF ✗~~

$F_2 = \{ AB \to CD,\ AC \to BD \}$

3NF ✓

BCNF ✓

R3

E	G

$F_3 = \{ E \to G \}$

3NF ✓

BCNF ✓

R4

D	E

$F_4 = \{ D \to E \}$

3NF ✓

BCNF ✓

Stud

name	hobby	food
UJJwal	volley cricket	Puchka chat
Aadi	Tennis cricket	Burger chat

1NF ✗

Stud

name	hobby	food
UJJwal	volley	Puchka
"	"	chat
"	cricket	Puchka
"	"	chat

1NF ✓

2NF ✓

3NF ✓

4NF? ✗

Stud1

name	hobby
UJJwal	volley
"	cricket
aadi	tennis
"	cricket

$\{ name \to\to hobby \}$

4NF ✓

5NF ✓

Stud 2

name	food
UJJwal	Puchka
"	chat
aadi	Burger
"	chat

$\{ name \to\to food \}$

4NF ✓

5NF ✓

Multivalued Dependency (MVD) :-

Trivial FD
$X \to Y$ is trivial
iff $Y \subset X$

$x \to\to y$ or $X \to\to Y$

Trivial MVD :-

$x \to\to y$ is trivial iff
$y \subset x$ or $x \cup y = R$

* Fourth Normal Form (4NF) :-

The R is in 4NF iff

* It is in 3NF
* Every MVD should be trivial type

* Join Dependency (JD) :-

If the rel^n. $R \to R_1, R_2 \ldots R_n$ is Join dependent iff

$\boxed{R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n = R}$

Fifth Normal Form (5NF) :-

The relⁿ R is in 5NF iff

* It is in 4NF and
* All fragments should be Join dependent on the key.

* 5NF is used to check if 4NF decomposition is correct or not.

stud1 ⋈ stud2 = stud

∴ It is in 5NF

T

P	Q	R
Jasmin	Coffee	BBS
Jasmin	coffee	MUM
Jasmin	Pao	MUM
aadi	Coffee	BBS

1NF ✓
2NF ✓
3NF ✓
BCNF ✗

P →→ Q
P →→ R

T₁

P	Q
Jasmin	Coffee
"	Pao
aadi	coffee

P →→ Q
4NF ✓
5NF ✗

T₂

P	R
Jasmin	BBS
"	MUM
aadi	BBS

P →→ R
4NF ✓
5NF ✗

$T_1 \bowtie T_2$

P	Q	R
Jasmin	Coffee	BBS
"	"	MUM
"	Pao	BBS
"	Pao	MUM
Aadi	Coffee	BBS

$T_1 \bowtie T_2 \neq T$

(*) **Denormalization :-** It is the process of increasing redundancy in the database either for convenience or to improve performance.

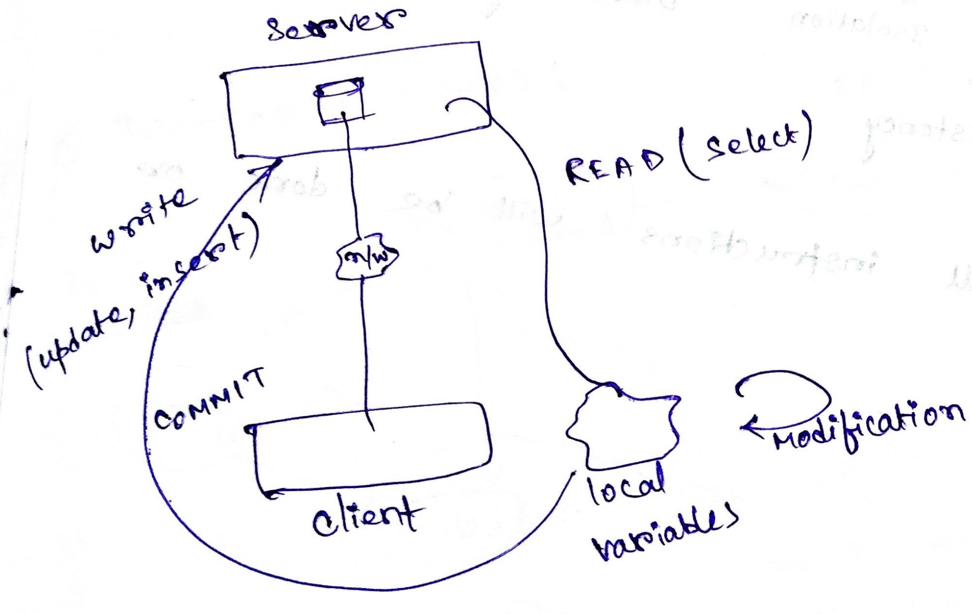
Disadvantages of Normalization :-

i) It is costlier / expensive

ii) It is time-consuming because it leads to more tables in the database.

Denormalization means we get lesser tables and with redundancies.

(*) **Transaction :-**



* It is the logical unit under a set of DML operations.

* Fund Transfer :-

$A \xrightarrow{1000} B$

1. Read (A)
2. A = A - 1000
3. write (A)
4. Read (B)
5. B = B + 1000
6. write (B)

Every transaction must have ACID property.

A → Atomicity

C → Consistency

I → Isolation

D → Durability

Atomicity → Either all instructions will be done or nothing.

(ROLLBACK)

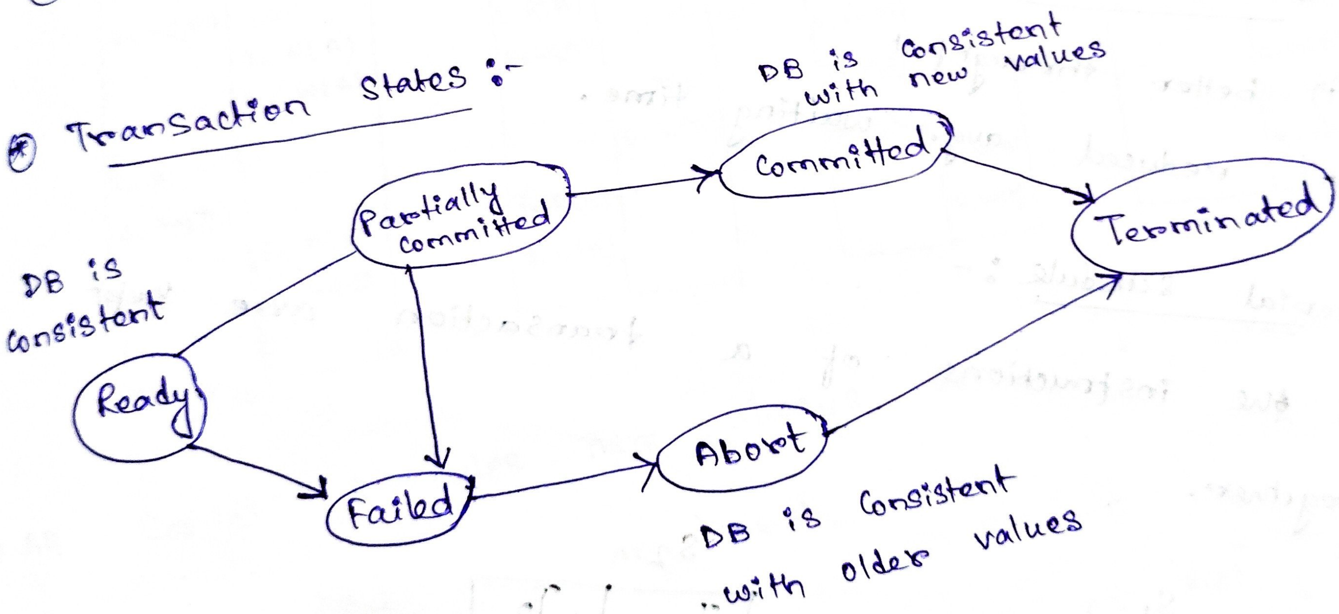
① Atomicity → Transaction Manager or Transaction Management component.

② Consistency → Application Programmer

③ Isolation → Concurrency Control Manager

④ Durability → Recovery Manger | Recovery Management component.

⊛ Transaction States :-



T_1

$A \xrightarrow{1000} B$

1. R(A)
2. A = A-1000
3. W(A)
4. R(B)
5. B = B+1000
6. W(B)

T_2

$A \xrightarrow{500} B$

1. R(A)
2. A = A-500
3. W(A)
4. R(B)
5. B = B+500
6. W(B)

Serial schedule

Concurrent "

for 2

3

Concurr

inst

int

Concurrent Exec^n :-

i) better throughput

ii) reduced avg. waiting time.

Serial schedule :-

All the instructions of a transaction are kept together.

S_1

T_1	T_2
1. R(A)	
2. W(A)	
3. R(B)	
4. W(B)	
	5. R(A)
	6. W(A)
	7. R(B)
	8. W(B)

S_2

T_1	T_2
	1. R(A)
	2. W(A)
	3. R(B)
	4. W(B)
5. R(A)	
6. W(A)	
7. R(B)	
8. W(B)	

• Al

To ch

A

for 2 transaction → 2 serial schedules

3 → 6

4 → 24

n → n!

<u>Concurrent Schedule</u> → It is the schedule where the instructions of different transactions are interleaved / mixed.

S_3

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
	4. W(A)
5. R(B)	
6. W(B)	
	7. R(B)
	8. W(B)

S_4

T_1	T_2
	1. R(A)
	2. W(A)
3. R(A)	
4. W(A)	
	5. R(B)
	6. W(B)
7. R(B)	
8. W(B)	

S_5

T_1	T_2
1. R(A)	
	2. R(A)
	3. W(A)
4. W(A)	
5. R(B)	
6. W(B)	
	7. R(B)
	8. W(B)

S_6

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
4. R(B)	
5. W(B)	
	6. W(A)
	7. R(B)
	8. W(B)

* • All Serial Schedules are always Consistent, whereas all Concurrent " are not.

To check the Concurrent Schedules are Consistent or not, there is a technique Serializability

Serializability → Conflict Serializability (in syllabus)

Serializability → View Serializability (not in syllabus)

Conflict Serializability :-

$T_1 \rightarrow Q \leftarrow T_2$

* $T_1 \rightarrow R(Q)$, $T_2 \rightarrow R(Q)$ — sequence doesn't matter
* $T_1 \rightarrow R(Q)$, $T_2 \rightarrow W(Q)$ — sequence matters
* $T_1 \rightarrow W(Q)$, $T_2 \rightarrow R(Q)$ — "
* $T_1 \rightarrow W(Q)$, $T_2 \rightarrow W(Q)$ — "

If both are only reading the value then sequence doesn't matter.

→ Conflict Serializability is used to check the concurrent schedule is consistent or not.

→ A concurrent schedule is conflict serializable if it is equivalent to a serial schedule.

• Target is to achieve S_1 from S_3

S_3

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
	4. W(A)
5. R(B)	
6. W(B)	
	7. R(B)
	8. W(B)

Swapping 4 & 5

T_1	T_2
R(A)	
W(A)	
R(A)	R(A)
W(B)	W(A)
	R(B)
	W(B)

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

S_5

T_1	T_2
1. R(A)	2. R(A)
	3. W(A)
4. W(A)	
5. R(B)	
6. W(B)	7. R(B)
	8. W(B)

Since both 3 & 4 are working on same value we cannot swap

∴ It is not conflict serializable

S_6

T_1	T_2
1. R(A)	
2. W(A)	3. R(A)
4. R(B)	
5. W(B)	6. W(A)
	7. R(B)
	8. W(B)

R(B) & R(A) swap

T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
W(B)	W(A)
	R(B)
	W(B)

, W(B) & R(A) swap

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

- **Precedence Graph** :- $G(V, E)$

Vertex → Transaction , E → Set of Edges.

Rule for edge :- $T_1 \to T_2$

- $T_1 \to R(A)$, $T_2 \to W(A)$
- $T_1 \to W(A)$, $T_2 \to R(A)$
- $T_1 \to W(A)$, $T_2 \to W(A)$
- $T_1 \to W(A)$,

2018

7) a)

In S_3

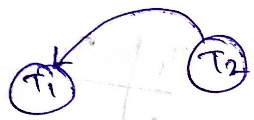
No. of transaction = 2

$\therefore$ Vertex = 2

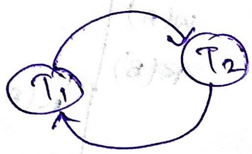
$T_1 \rightarrow R(A)$, $T_2 \rightarrow W(A)$

* If you get a cycle, it is not conflict serializable
* If you donot get a cycle, it is conflict "

In S_4 $\rightarrow$



In S_5 $\rightarrow$

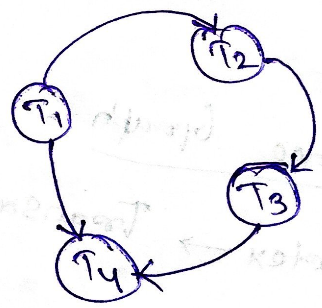


not conflict serializable

Q-2023

6) b) $S = R_2(X), W_3(X), W_1(Y), R_2(Y), W_2(Z), R_4(X), R_4(Y)$

T_1	T_2	T_3	T_4
	1. R(X)		
		2. W(X)	
3. W(Y)			
	4. R(Y)		
	5. W(Z)		
			6. R(X)
			7. R(Y)



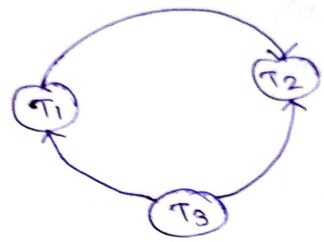
It is conflict serializable

$\therefore$ It is consistent

2018

7/a) S: r1(X), r2(Z), r1(Z), r3(X), r3(Y), w1(X), w3(Y), r2(Y), w2(Z), w2(Y)

T_1	T_2	T_3
1. R(X)		
	2. R(Z)	
3. R(Z)		
		4. R(X)
		5. R(Y)
6. W(X)		
		7. W(Y)
	8. R(Y)	
	9. W(Z)	
	10. W(Y)	



It is conflict Serializable

* Need of Controlling Concurrent execution :-

i) Lost update problem
ii) Dirty Read Problem (Temporary Read)
iii) Incorrect Summary problem

* **Lost update :-**

T_1	T_2
1. R(A)	
	2. R(A)
	3. W(A)
4. W(A)	
5. R(B)	
6. W(B)	

Let, A: 10,000
B: 2,000

T_1: $A \xrightarrow{1000} B$

T_2: withdraw 500

T_1	T_2
10,000	
	10,000
	9500
9000	
2000	
8000	

whatever T_2 has updated, it is lost

* **Dirty Read :-**

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
	4. W(A)
5. R(B)	
6. W(B)	

A: 10,000
B: 2000

T_1: $A \xrightarrow{1000} B$

T_2: withdraw 500

T_1	T_2
10,000	
9,000	9,000
	8,500
2000	

stopped
so it will
rollback

whatever T_1 reads, it is incorrect

*

* Incorrect Summary :-

	T1	T2	
10,000	1. R(A)		
9,000	2. W(A)		
		3. R(A)	9000
		4. R(B)	2000
		5. disp (A+B)	11,000
2000	6. R(B)		
8000	7. W(B)		

when one transaction is partially occurring and other one will display.

* For implementation of concurrent scheduling there are two mechanisms —

i) Locking mechanism
ii) Timestamp mechanism

DBMS 3/4/24 (wed)

Lock-Based Protocol :-

Transaction
↓
Read / write

i) T can perform R/W once lock is acquired

ii) T can perform only read once shared lock / Read lock acquired
↓
Lock-S (A)

iii) T can perform both Read and write once exclusive / write lock is acquired.
→ lock-X(A)

iv) T can release lock using Unlock(A)

All the rules are controlled by CCM (Concurrency Control Manager)

v) The read lock request will be granted on the data once it is under Read lock.

vi) The write lock request will be denied on the data once it is under Read lock.

vii) The read/write lock request will be denied on the data once it is under write lock.

S_3

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
	4. W(A)
5. R(B)	
6. W(B)	
	7. R(B)
	8. W(B)

T_1	T_2	CCM
1. lock-X(A)		2. Grant-X(A, T_1)
3. R(A)		
4. W(A)		
5. unlock(A)		
	6. lock-X(A)	7. Grant-X(A, T_2)
	8. R(A)	
	9. W(A)	
	10. unlock(A)	
11. lock-X(B)		12. Grant-X(B, T_1)
13. R(B)		
14. W(B)		
15. unlock(B)		
	16. lock-X(B)	17. Grant-X(B, T_2)
	18. R(B)	
	19. W(B)	
	20. unlock(B)	

All the instructions are successfully executed

S_5

T_1	T_2
R(A)	
	R(A)
	W(A)
W(A)	
R(B)	
W(B)	
	R(B)
	W(B)

T_1	T_2	CCM
1. lock-X(A)		2. Grant-X(A,T_1)
3. R(A)		
	4. lock-X(A)	

This schedule is not correct as write lock is already occupied

T_1	T_2
R(A)	
W(A)	R(A)
	W(A)
R(B)	R(B)
W(B)	W(B)

T_1	T_2	CCM
1. lock-X(A)		2. Grant-X(A,T_1)
3. R(A)		
4. W(A)		
5. Unlock(A)	6. lock-X(A)	7. Grant-X(A,T_2)
	8. R(A)	
	9. W(A)	
	10. Unlock(A)	
11. lock-X(B)		12. Grant-X(B,T_1)
13. R(B)	14. lock-X(B)	

not correct

(*) Incorrect summary problem :-

modification

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
	4. R(B)
	5. disp(A+B)
6. R(B)	
7. W(B)	

T_1	T_2	CCM
1. lock-X(A)		2. Grant-X(A, T_1)
3. R(A)		
4. W(A)		
5. unlock(A)		
	6. lock-S(A)	
		7. Grant-S(A, T_2)
	8. R(A)	
	9. unlock(A)	
	10. lock-S(B)	11. Grant-S(B, T_2)
	12. R(B)	
	13. unlock(B)	
	14. disp(A+B)	
15. lock-X(B)		16. Grant-X(B, T_1)
17. R(B)		
18. W(B)		
19. unlock(B)		

This should not be allowed, so the unlock should be delayed

Release the lock once the money is deducted & added (overall oper? is completed)

$\boxed{\text{Delay in unlocking}}$

Modification

T_1	T_2	CCM
1. lock-X(A)		
		2. Grant-X(A, T_1)
3. R(A)		
4. W(A)		
	5. lock-S(A) (rule no. 7)	X

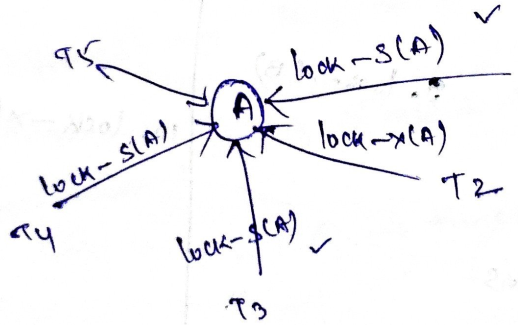
S_5

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
R(B)	
W(B)	
	R(A)
	W(A)

T_1	T_2	CCM
1. lock-X(A)		2. Grant-X(A, T_1)
3. R(A)		
4. W(A)		
	5. lock-X(B)	6. Grant-X(B, T_2)
	7. R(B)	
	8. W(B)	
9. lock-X(B)		
	10. lock-X(A)	

Deadlock

In lock based protocol we will face issues like deadlock and starvation.



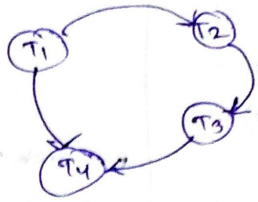
$T_2 \rightarrow$ starvation

2018

6) b)

(A)

T1	T2	T3	T4
	1. R(X)		
		2. W(X)	
3. W(Y)			
	4. R(Y)		
	5. W(Z)		
			6. R(X)
			7. R(Y)



origin is the point where no vertex is towards it

(B) $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$

$T_1 \rightarrow T_4 \rightarrow T_2 \rightarrow T_3$

Lock-Based Protocol :- unlocking will be delayed

T1

$A \xrightarrow{1000} B$

T2

$B \xrightarrow{1000} A$

T1	T2
1. R(A)	
2. W(A)	
	3. R(B)
	4. W(B)
5. R(B)	
6. W(B)	
	7. R(A)
	8. W(A)

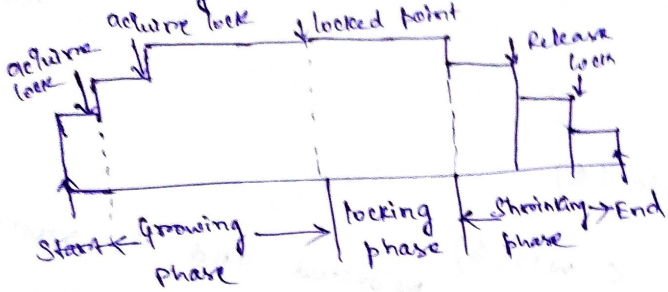
T1	T2	CCM
1. lock-X(A)		2. grant-X(A,T1)
3. R(A)		
4. W(A)		
	5. lock-X(B)	6. Grant-X(B,T2)
	7. R(B)	
	8. W(B)	
9. lock-X(B)		
	10. lock-X(A)	Deadlock

Locking Mechanism means deadlock will arise

⊛ 2PL (Two Phase Locking) Protocol :-

i) Growing phase :- locks will be acquired

ii) Shrinking phase :- locks will be released



theoritical
* Static 2PL → before starting sequence of R & W (i.e. no. of locks) should be known

* Strict 2PL → all exclusive locks should not be released until the transaction commits.

* Rigorous 2PL → The read as well as write locks should not be released until the transaction commits.

⊛ Time-Stamp Based Protocol :-

Date & Time or sequence/token no.

TS(T_1) TS(T_2)

If T_1 is starting earlier than T_2 then, $T_1 < T_2$ TS(T_1) < TS(T_2)

Every timestamp is unique

[Q] → RTS Q → time stamp at which the data is last read

[Q] → WTS Q → time stamp at which the data is last written

* T_1 issues Read(Q)
  - → If $TS(T_1) < WTS(Q)$ Rejected and T_1 rollback
  - → If $TS(T_1) \geq WTS(Q)$ Processed and $RTS(Q) = Max(RTS(Q), TS(T_1))$

* T_1 issue write(Q)
  - → If $TS(T_1) < WTS(Q)$ Rejected and T_1 roll back
  - → If $TS(T_1) < RTS(Q)$ Rejected " T_1 roll back
  - → otherwise it will be processed and $WTS(Q) = TS(T_1)$

S_3

T_1	T_2
1. R(A)	
2. W(A)	
	3. R(A)
	4. W(A)
5. R(B)	
6. W(B)	
	7. R(B)
	8. W(B)

$\boxed{A}$ RTS = $\cancel{\emptyset}$ $\cancel{10}$ 11, WTS = $\cancel{\emptyset}$ $\cancel{10}$ 11

$\boxed{B}$ RTS = $\cancel{\emptyset}$ $\cancel{10}$ 11, WTS = $\cancel{\emptyset}$ $\cancel{10}$ 11

$TS(T_1) = 10$

$TS(T_2) = 11$

1. T_1 issue R(A) ✓
2. T_1 " W(A) ✓
3. T_2 " R(A) ✓ 11 > 10
4. T_2 " W(A) ✓
5. T_1 issue R(B) ✓
6. T_1 " W(B) ✓
7. T_2 " R(B) ✓
8. T_2 " W(B) ✓

Ex-2

SS	
T_1	T_2
1. R(A)	
	2. R(A)
	3. W(A)
4. W(A)	
5. R(B)	
6. W(B)	
	7. R(B)
	8. W(B)

TS (T_1) = ~~10:30~~ 10:50

TS (T_2) = ~~10:40~~ 11:00

$\boxed{A}$ RTS = ~~0~~ ~~10:30~~ 10:40

WTS = ~~0~~ 10:40

$\boxed{B}$ RTS = 0

WTS = 0

1. T_1 issues R(A) ✓
2. T_2 " R(A) ✓
3. T_2 " W(A) ✓
4. T_1 " W(A) ✗ rollback

Database Recovery :-

Transaction failure :-

i) Power supply failure

ii) Logical Error

iii) Network failure

iv) Hardware error

For recovery -

1) Backup

Backup is stored as log files.

stored in secondary memory

DML opern's

regular activities

↑

* log

T_1		T_2
R(A)	A: 10,000	R(C)
A = A-1000	B: 3,000	C = C-500
W(A)	C: 5,000	W(C)
R(B)		
B = B+1000		
W(B)		

log-based database recovery

→ defferred database recovery

→ immediate database recovery

~~Content of log files :-~~

Deferred DB recovery :-

Here, the modified data is shown to the user after the transaction is being committed.

$T_1 \rightarrow T_2$

<T_1 start>

< T_1, data item, new value>

<T_1 commits>

~~log~~ Content of log files :-

log

<T_1 start>
<T_1, A, 9000>
<T_1, B, 4000>
<T_1 commit> — A = 9000, B = 4000
<T_2 start>
<T_2, C, 4500>
<T_2 commit> — C = 4500

Once the transaction is completed, user can get the updates.

Case-1 :-

log

<T_1 start>
<T_1, A, 9000>
<T_1, B, 4000>

No Commit

So erase the data

Case-2 :-

log

<T_1 start>
<T_1, A, 9000>
<T_1, B, 4000>
<T_1 Commit>

Server is not yet processed

Once the Commit is processed from client side it is saved forever

Therefore, Server will have to redo the operⁿ.

redo(T_1)

redo will be done when there will be <start> <Commit>

redo(T_1) is idempotent
↓ do once or multiple time same value will be

Case-3 :-

log

<T_1 start>
<T_1, A, 9000>
<T_1, B, 4000>
<T_1 commit>
<T_2 start>
<T_2, C, 4500>

T_1 redo
T_2 erase

Case-4 :-

<T_2 & commit>

redo (T_1)
redo (T_2)

Immediate DB recovery :-

Here, the updates are visible to the user just after the modification

<T_i start>
<T_i Commit>
<T_i, data item, old value, new value>

log

<T_1 Start>
<T_1, A, 10000, 9000> — A : 9000
<T_1, B, 3000, 4000> — B : 4000
<T_1 commit>
<T_2 start>
<T_2, C, 5000, 4500> — C : 4500
<... commit>

Case-1 :-

log

<T1 start>

<T1, A, 10000, 9000>

<T1, B, 9000, 4000>

System fails here

undo will be done

undo " " " once the Commit statement is missing

It is also idempotent

Case-2 :-

log

<T1 start>

<T1, A, 10K, 9K>

<T1, B, 3K, 4K>

<T1 commit>

redo (T1)

Slide no. → 22

Case-3 :-

log

<T1 start>

<T1, A, 10K, 9K>

<T1, B, 3K, 4K>

<T1, Commit>

<T2, C, 5000, 4500>

System fails here

T_1 redo

T_2 undo

Case-4 :-

log

<T1 Commit>

<T2 commit>

T_1 redo

T_2 redo