19UCC023 Mohit Akhouri DBMS

Q3) - - - - - - - - - - -

Soln:

Given set of FD are:

AB → C

C → B

A → D

* reducing to irreducible set:

For AB → C, calculating the closure

$(AB)^* = \cancel{\{A, B, D\}} \{A, B\}$

↓

closure of AB

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

$= \{A, B, C\} \cup \{B\}$ = already present

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

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

$(A)^* = \{A\}$

↓

closure of A

$= \{A\} \cup \{D\} = \{A, D\}$

19UCC023 Mohit Alchouri DBMS

$(B)^+ = \{B\}$

Closure of B $= \{B\}$

Since $(AB)^+ \neq (A)^+$

and

$(AB)^+ \neq (B)^+$

we cannot remove any term from $AB \to C$

also we cannot obtain $AB \to C$ from other remaining FD, so it is irreducible

So final set contains:

$$\left\{ \begin{array}{c} AB \to C \\ C \to B \\ A \to D \end{array} \right\} \Rightarrow \text{irreducible set}$$

* computing the candidate keys:

finding closure of ~~{A, B}~~ (AB)

$(AB)^+ = \{A, B\} \cup \{C\}$

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

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

19UCC023 Mohit Akhouri DBMS

calculating closure for ~~(A)*~~ (A)

$(A)^* = \{A, D\} = (\{A\} \cup \{D\})$

~~calculating closure for (B)~~

~~$(B)^* = \{B\}$~~

~~calculating closure for (D)~~

~~$(D)^* = \{D\}$~~

Calculating closure for (C)

$(C)^* = \{C\} \cup \{B\}$

$= \{C, B\}$

~~calculating~~

~~Since only $(AB)^*$ =~~

since closure of no element gives the complete relation (A, B, C, D, E),

"NO CANDIDATE KEY HERE"

~~Bad design since~~ duplicate records may be present

19UCC023   Mohit Alhouni   DBMS

* Since no candidate keys are there, first condition for 3NF i.e,

① Non-Primary keys irreducibly dependent on primary key = true

② but second condition that non-primary keys should be mutually independent not followed since, because of following

$C \rightarrow B$

$A \rightarrow D$

$AB \rightarrow C$

* "No decomposition" into 3NF since condition not ~~satisfied for~~ satisfied for 3NF

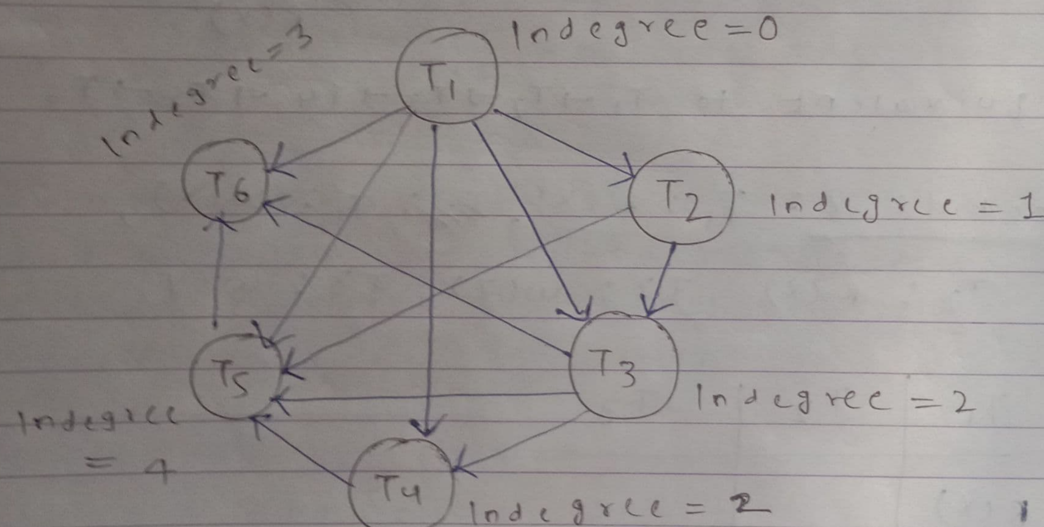
19UCC023 Mohit Akhouri DBMS

Q5) - - - - - - - - - - - - -

Soln:

Schedule 1: $W_1(X)$, $R_2(X)$, $W_3(X)$, $R_4(X)$, $W_5(X)$, $R_6(X)$

Precedence graph for schedule 1 is:



Precedence graph contains no cycle, Hence Schedule 1 is conflict serializable

If we go by indegree of precedence graph nodes, we get T_1 has indegree 0 so write T_1, then delete T_1, Now T_2 has indegree 0, - - - if we continue this process, we get the following ~~serial~~

Topological sequence : $T_1 \to T_2 \to T_3 \to T_4 \to T_5 \to T_6$

~~(one equivalent serial schedules)~~

If we draw in terms of diagram, we also get the following order
$T_1 \to T_2 \to T_3 \to T_4 \to T_5 \to T_6$

T_1	T_2	T_3	T_4	T_5	T_6
W(X)					
	R(X)				
		W(X)			
			R(X)		
				W(X)	
					R(X)

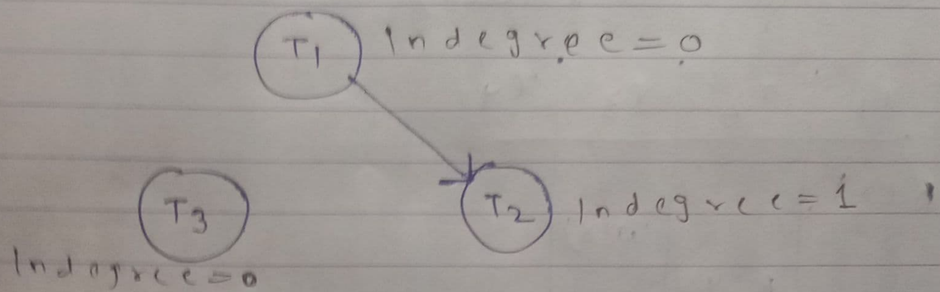
equivalent to $T_1 \to T_2 \to T_3 \to T_4 \to T_5 \to T_6$

<u>Schedule 2</u>: T_1 : R(X), ~~T_2~~: T_1 : W(X),
T_2 : R(X), T2 : W(Z), T3 : W(Y)

T_1	T_2	T_3
R(X)		
W(X)		
	R(X)	
	W(Z)	
		W(Y)

19ucc023 Mohit Akhouri DBMS

Precedence graph for schedule 2 is:



~~No edge from T_1 since no conflict from T_1~~

For T_1: write (X), there is ~~conflict~~ conflict for R(X) in T_2, so one edge from T_1 to T_2

No conflict from T_2 since R(X) is there and no later ~~s~~ transactions modify shared resource X, so no edge, no conflict for $W_2(Z)$ either

Since the precedence graph does not contain any cycle, schedule 2 is conflict serializable

Now, two nodes T_1 & T_3 have indegree 0, so if we start from T_1, two Topological sequence possible and from T_3, one sequence possible, possible Topological sequences are:

$$\boxed{\begin{array}{l} T_1 \rightarrow T_2 \rightarrow T_3 \\ T_1 \rightarrow T_3 \rightarrow T_2 \\ T_3 \rightarrow T_1 \rightarrow T_2 \end{array}}$$

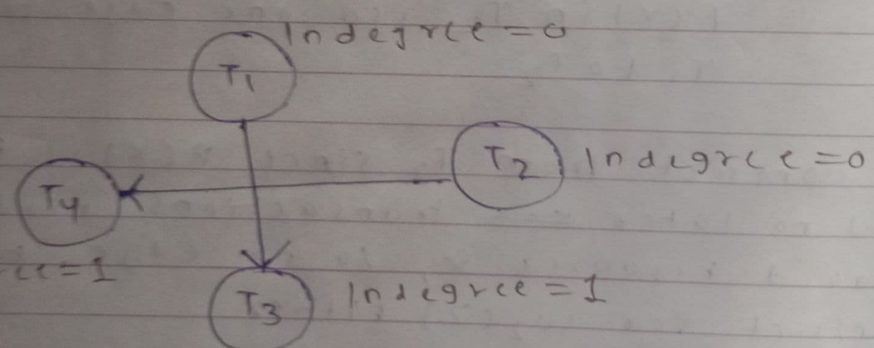
= Topological sequence (three equivalent serial schedules)

19UCC023 Mohit Akhouri DBMS

Schedule 3: $T_1 : R(X)$, $T_3 : W(X)$, $T_2 : R(Y)$, $T_4 : W(Y)$

T_1	T_2	T_3	T_4
R(X)			
		W(X)	
	R(Y)		
			W(Y)

Precedence graph is:



Since graph = Acyclic, $\boxed{\text{Schedule 3 conflict Serializable}}$

Since 2 nodes, have Indegree = 0, Total possible Topological sequence are 6, they are as follows:

If we delete T_1 first:
- $T_1 \rightarrow T_3 \rightarrow T_2 \rightarrow T_4$
- $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$
- $T_1 \rightarrow T_2 \rightarrow T_4 \rightarrow T_3$

If we delete T_2 first:
- $T_2 \rightarrow T_4 \rightarrow T_1 \rightarrow T_3$
- $T_2 \rightarrow T_1 \rightarrow T_4 \rightarrow T_3$
- $T_2 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4$

topological sequences (Total = 6) (No. of equivalent serial schedules)

Total no. of equivalent serial schedules ~~are~~ are

$\boxed{1+3+6 = 10}$

19UCC023 Mohit Akhouri DBMS

Q6) - - - - - - - - - - - - -

soln:

Schedule 1:

T_1	T_2	T_3	T_4
R(A)			
	R(C)		
R(C)			
		R(A)	
		R(B)	
W(A)			
COMMIT			
		W(B)	
		COMMIT	
	R(B)		
	W(C)		
	W(B)		
	COMMIT		

CONDITION FOR STRICT RECOVERABILITY

T_1	T_2
W(A)	
	R(A)/ W(A)

allowed only after commit of T_1

Checking strict recoverability conditions for all three shared resource (A, B, C)

For shared resource A: follows conditions

$W_1(A)$ occuring in T_1, but no W(A) OR R(A) occur in other transactions after $W_1(A)$, so conditions automatically satisfied for A.

19UCC023 Mohit Akhouri DBMS

For shared resource B: follows conditions

$W_3(B)$ occurs in T_3 and conflict for $R_2(B)$ and $W_2(B)$ also found in T_2.

but "commit has been done" after $W_3(B)$ for T_3, so R(B) and W(B) allowed after that.

Since $R_2(B)$ and $W_2(B)$ come after commit of $W_3(B)$, they are allowed and this follow the rule of strict recoverability

Hence B also satisfies conditions.

For shared resource C: follows conditions

$W_2(C)$ occurs in T_2 but W(C) or R(C) are found ~~after the~~ further in any transaction, and also commit of T_2 also done, so. C automatically follows the conditions of strict recoverability

Since all three shared data items (A, B, C) follows conditions of strict recoverability, schedule 1 is STRICT RECOVERABLE SCHEDULE

19UCC023 Mohit Akhouri DBMS

Schedule 2 :

T_1	T_2	T_3	T_4
R(A)			
	R(C)		
		R(C)	
R(C)			
	R(B)		
		R(B)	
W(A)			
COMMIT			
	W(C)		
		W(B)	
	W(B)		
	COMMIT		
		COMMIT	

checking strict recoverability conditions for all shared resource A, B, C :

checking for A : follows conditions

$W_1(A)$ found in T_1, but no where else W(A) / R(A) is found ~~and also commit~~ so A automatically satisfies conditions for strict recoverability.

19UCC023 Mohit Akhouri DBMS

Checking for B: does not follow condition.

$w_3(B)$ found in transaction T_3 and immediately after this $w_2(B)$ is found in T_2.

Since $w_2(B)$ is only allowed after "COMMIT" of T_3 after $w_3(B)$, ~~so B do~~ and here no COMMIT found after $w_3(B)$, so B does not follow the conditions for strict recoverability.

Checking for C: follows condition

$w_2(C)$ found but no other transaction is either doing $w(C)$ OR $R(C)$, so C automatically follows the conditions for strict recoverability.

By the analysis above, we found that

~~A 2 C~~ A and C follows conditions for strict recoverability but B does not follow, so the given schedule is not strict ~~recoverable~~ recoverable.

Schedule 2 is ~~not~~ "NOT STRICT RECOVERABLE"

Schedule 1 = strict recoverable

Schedule 2 = not strict recoverable

19UCC023 Mohit Akhouri DBMS

Q 7) — — — — — — — — — — — —

Soln:

Given schedule is:

$R_1(A), R_2(B), W_2(B), R_3(A), W_3(A),$
$R_1(B), R_3(B), R_2(A), W_2(A)$

T_1	T_2	T_3
R(A)		
	R(B)	
	W(B)	
		R(A)
		W(A)
R(B)		
		R(B)
	R(A)	
	W(A)	

Conditions for view serializable schedule

① Initial Read should be same

② all W-R sequences must be same

③ Final update should be same

To find whither this schedule is view-serializable or not, start with third condition first.

(III) → Final update should be same

Final update table

Shared res.	Final update
A	T_2
B	T_2

— (I)
— (II)

Implications got from (I) and (II) are,

T_2 should update last, before that T_1, T_3 can update in any order

$\boxed{T_1\ T_3 \longrightarrow T_2}$ ⇒ Implication 1

~~(II) → W-R seque~~

~~(II) → Write-Read sequence:~~

~~Shared res.~~	~~Write-Read seq.~~
~~A~~	~~$T_3 \rightarrow T_1$~~ No W-R sequence
~~B~~	No W-R sequence

(ii) Write - Read Sequence :-

Shared resource	W-R sequence	
A	$T_3 \rightarrow T_2$	— (i)
B	$T_2 \rightarrow T_1 T_3$	— (ii)

from (i), we get: $T_3 \rightarrow T_2$

from (ii), we get: $T_2 \rightarrow T_1 T_3$
OR $T_2 \rightarrow T_1$, $T_2 \rightarrow T_3$

So total implications, we get are:

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

conflicting conditions
(so not view-serializable, but we will check condition 3 for surety)

(i) Initial Read :

Shared resource	initial read	write	
A	T_1 T_3	T_2 T_3	— (i)
B	T_2	T_2	— (ii)

19UCC023 Mohit Akhouri DBMS

From ① & (II), we get implications like:

$T_1 T_3 \rightarrow T_2 T_3$

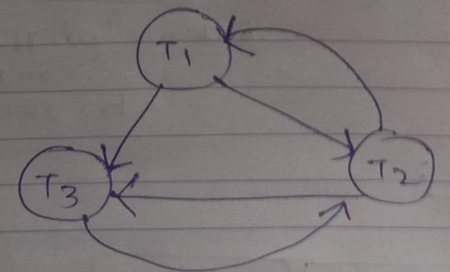
OR $T_1 \rightarrow T_2$

So after analysing all implications, we get final result that since there are conflicting implications, that is:

$T_3 \rightarrow T_2$
$T_2 \rightarrow T_3$

we say that schedule is "NOT VIEW SERIALIZABLE"

<u>Cross checking from Precedence graph:</u>



Since, we get 2 cycles, given schedule is not serializable.

Hence, we cross checked that "GIVEN SCHEDULE IS NOT VIEW SERIALIZABLE AND THEREFORE NOT SERIALIZABLE ALSO"

19UCC023 Mohit Akhouri DBMS

Q8) – – – – – – – – –

Soln:

Given log file is:

1) ~~(STA~~ (start, T_1)
2) (W, T_1, B, 3, 2)
3) (start, T_2)
4) (commit, T1)
5) (W, T2, C, 5, 7)
6) (checkpoint, {T2})
7) (start, T3)
8) (W, T3, A, 1, 9) Check till here
9) (start, T4)
10) (W, T4, C, 7, 2)
11) CRASH

Since CRASH occur at step 11, we will revisit the log file till the checkpoint at step 6

conditions found in traversing the log file are:

(i) T4 started but no commit of T4 found, Hence UNDO T4

19UCC023 Mohit Akhouri DBMS

(ii) T_3 started but no commit for T_3 found, Hence UNDO T_3

(iii) Activation list of checkpoint contains T2 which means T2 started earlier but no commit of T_2 found while traversing back
Hence UNDO T2.

(iv) Before the checkpoint, we don't need to check since ~~database~~ everything fine upto the checkpoint

So UNDO List contains T4, T3, T2 and no REDO list since no commit for any transaction found.

1) UNDO LIST: T_4, T_3, T_2
REDO LIST: empty

2) UNDO order: T4 then T_3 then T_2

3) Final values of A, B, C are:

A = 1
B = 2
C = 5

19UCC023

first C
due to
5 due to
chang
A ~~contains~~
UNDO of

B retains
Committed
UNDO / RE
the check

19UCC023 Mohit Alchouri DBMS

First C changes value from 2 to 7 due to UNDO of T4, then from 7 to 5 due to undo of T2.

A ~~contains~~ changes value from 9 to 1 due to UNDO of T3

B retains the value of 2 written by commmitted transaction T_1 since no UNDO/REDO of T_1 because it is before the checkpoint.