3/12/2021

Declaration

I, bearing Reg. no. 106119142, agree and acknowledge that:

1. The assessment was answered by me as per instructions applicable and that I have not resorted to unfair means to deliberately improve my performance.

2. I have neither impersonated nor have been impersonated by anyone.

Name: VIGNESH.A.S

Roll. no: 106119142

Sub: CSPC52-DBMS

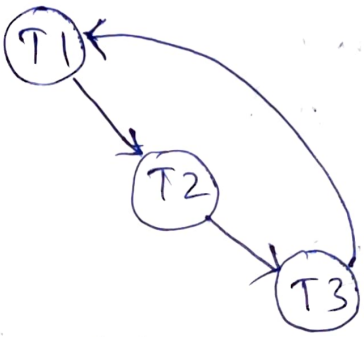
Ph no: 9952053860

Sign: A.Vignesh

1) (a) Schedule - 1 :

T1	T2	T3
R(A)		
	R(B)	
		R(C)
W(A)		
	W(B)	
		W(C)

The precedence graph for the schedule is



A cycle exists here, hence it is inferred that the schedule is not conflict seriatizable.

→ Initial read satisfies as only one read is happening.

→ Final write satisfies as only one write operation happens.

→ Due to the cycle in the serial schedule, there would be update conflicts. Hence update read fails to satisfy.

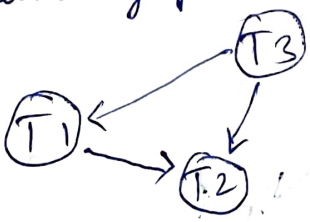
⇒ The schedule is not view serializable.

1)

(b)

T1	T2	T3
R(A)		
R(B)		
W(A)		
	R(B)	
	W(A)	
		R(C)
		R(B)
		R(D)
	W(B)	
W(C)		
		W(D)

The precedence graph is:



∴ There is ~~no conflict~~ no cycle in this graph.
Hence it is conflict serializable.

By Topological sort,

Serial schedule => T3, T1, T2

=> Since the schedule is conflict serializable, it is also view serializable.

3) Gn: $R(A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8)$

FD: $A_1, A_2 \rightarrow A_3$

$A_1, A_3 \rightarrow A_2$

$A_1, A_4 \rightarrow A_5$

$A_2 \rightarrow A_4$

$A_2, A_3 \rightarrow A_1$

$A_5 \rightarrow A_7$

To find: candidate Keys

Soln:

$\Rightarrow$ A_6, A_8 not in FDS

$\Rightarrow$ $A_1, A_2, A_3, A_4, A_5, A_7$

$A_1 A_2^+ \Rightarrow A_1 A_2 A_3 A_4 A_5 A_7$

$A_2 A_3^+ \Rightarrow A_2 A_3 A_1 A_4 A_5 A_7$

$A_1 A_3^+ \Rightarrow A_1 A_3 A_2 A_4 A_5 A_7$

$\rightarrow$ Adding A_6, A_8 to all.

candidate Keys.
- $A_1 A_2 A_6 A_8$
- $A_2 A_3 A_6 A_8$
- $A_1 A_3 A_6 A_8$

To decompose into BCNF:

for every non trivial FD (that's not superkey)

decompose R.

$\alpha \rightarrow \beta$ $\xRightarrow{\text{decompose}}$ i) $\alpha \cup \beta$

ii) $R - (\beta - \alpha)$

3) cont...

i) $A_1 A_2 \rightarrow A_3$

Decompose R as $R_0(A_1 A_2 A_3 A_4 A_5 A_7)$ &
$R_1(A_1 A_2 A_6 A_8)$

ii) $A_1 A_3 \rightarrow A_2$

$A_1 A_3$ is superkey in R_1 => no change.

iii) $A_1 A_4 \rightarrow A_5$

Decompose R_1 as $R_2(A_1 A_4 A_5 A_7)$ and
$R_3(A_1 A_2 A_3 A_4)$

iv) $A_2 \rightarrow A_4$

R_3 is split as $R_4(A_2, A_4)$ &
$R_5(A_1 A_2 A_3)$

v) $A_2 A_3 \rightarrow A_1$

Already a superkey in R_5

vi) $A_5 \rightarrow A_7$

violates R_2 as A_5 is not a superkey.

R_2 is split as $R_6(A_5, A_7)$ and $R_7(A_1, A_4, A_5)$

So we finally get,

$R_0(A_1 A_2 A_6 A_8)$
$R_4(A_2 A_4)$
$R_5(A_1 A_2 A_3)$
$R_6(A_5 A_7)$
$R_7(A_1 A_4 A_5)$

} BCNF is satisfied.

4) Gn: $F: \{A_1 \to A_2,\ A_2 \to A_3,\ A_1 \to A_3,\ A_1A_2 \to A_1\}$

To find: minimal cover.

Soln:

So we check FDs and get,

Step-1: FD with only one attribute on RHS

$A_1 \to A_2$
$A_2 \to A_3$
$A_1 \to A_3$
$A_1A_2 \to A_1$

Step-2: Remove trivial FD's:

$A_1 \to A_2$
$A_2 \to A_3$
$A_1 \to A_3$

Step-3: Minimizing LHS, we obtain same as above

$A_1 \to A_2$
$A_2 \to A_3$
$\Downarrow$
$A_1 \to A_3$ [Transitive]

Step-4: Removing Redundant FDs

∴ Minimal cover $\Rightarrow$ $\{A_1 \to A_2,\ A_2 \to A_3\}$

5)

Gn: $F\{A \to B, BC \to D\}$
on $R(A,B,C,D)$

Soln:

We do,

$A \to AB$
$B \to B$
$C \to C$
$D \to D$

$AB \to AB$
$[AC \to ABCD] \to$ candidate / super key
$AD \to ABD$
$BC \to BCD$
$BD \to BD$
$CD \to CD$

$[ABC \to ABCD] \to$ Super key

$ABD \to ABD$
$BCD \to BCD$
$[ACD \to ABCD] \to$ super key
$[ABCD \to ABCD] \to$ Super key.

$A \to AB$, $BC \to BCD$

$\Rightarrow \boxed{F_+ = \{A, B, C, D\}}$

10)

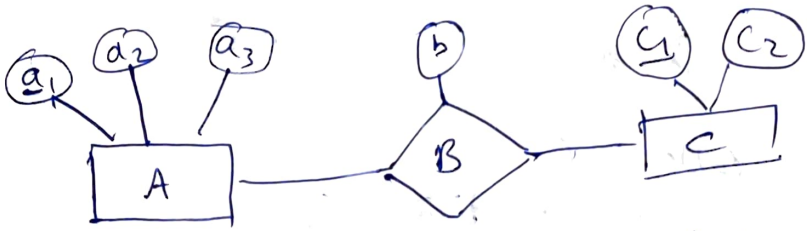
T_5 — already committed before checkpoint, no need to do anything.

T_6 — Not committed ∴ UNDO

T_7 — Committed before crash. ∴ REDO

T_8 — committed before checkpoint, no need to do anything.

8)



Many to one mapping from A to C via B.

→ Max Tables = 3

```
create table A (
    a1  <datatype> Primary key,
    a2  <datatype>
    a3  <datatype>);

Create table B (
    a1  <datatype>
    k1  <dt>
    b   <dt>
    foreign key  a1  references A(a1);
    foreign key  c1  references C(c1));
```

```
Create table c (
    c1 <dt> Primary key,
    c2 <dt> );
```

→ Min table => 2

```
Create table A (
    a1 <dt> Primary key,
    a2 <dt>,
    a3 <dt>,
    b  <dt>,
    c1 <dt>,
    foreign key c1 references c(c1);

Create table c (
    c1 int Primary key,
    c2 int);
```

9)

(a) ABCDE

No tuples as C & D together don't match

(b)

A	B	C	D	P'	E
3	4	2	6	1	3
2	2	5	1	4	5
4	2	5	3	4	5
3	4	2	6	5	7

(c)

B	C	D	E
2	5	4	7
2	5	4	5
4	2	1	3

=>

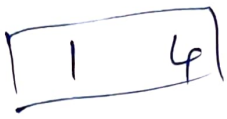
B	E
2	5
2	5
4	3

(d) $\Pi_{A,B} R - \vartheta_{S(A,B)} (\Pi_{B,C} S)$

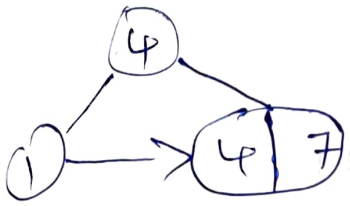
A	B
2	2
4	2

6)

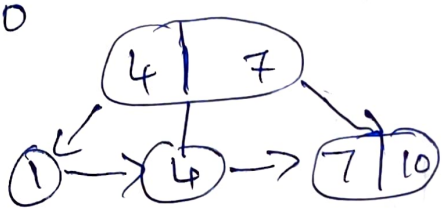
→ 4



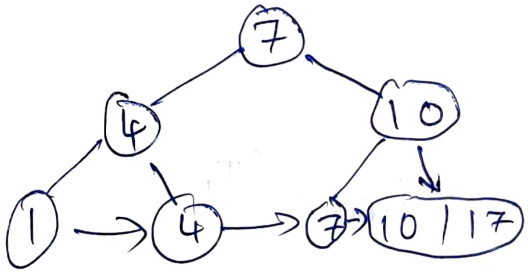
→ 7



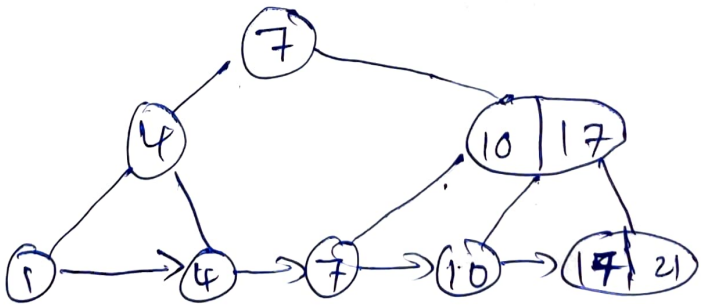
→ 10



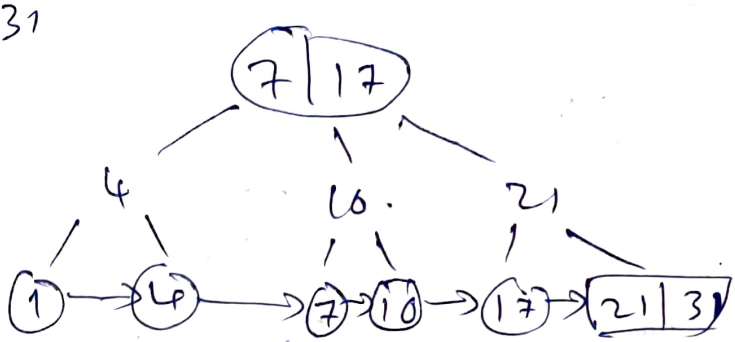
→ 17



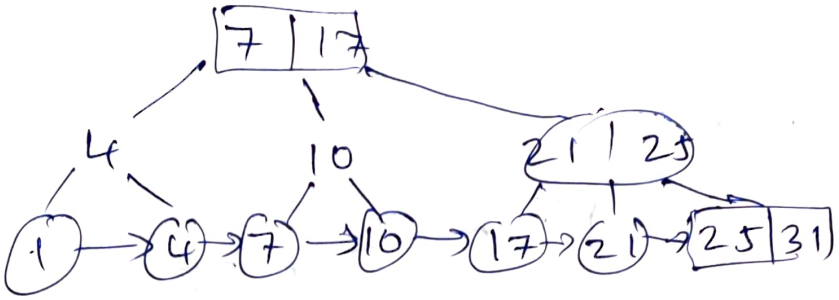
→ 21



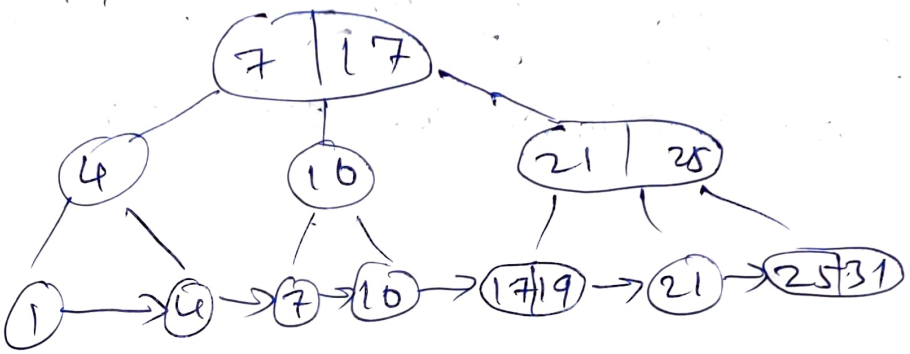
→ 31



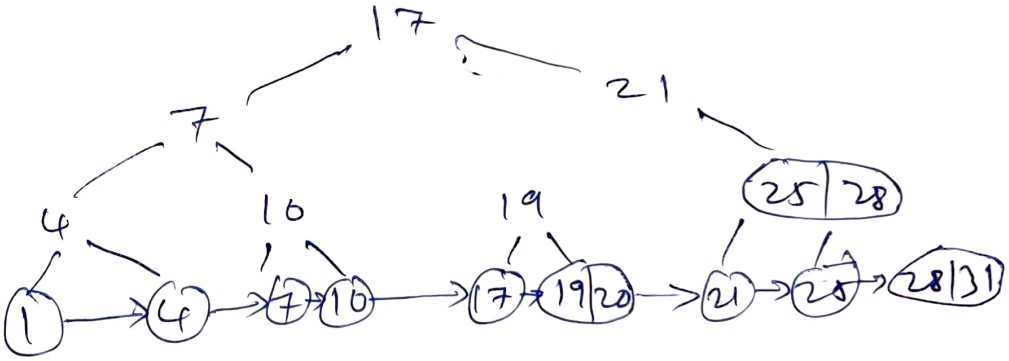
→ 25



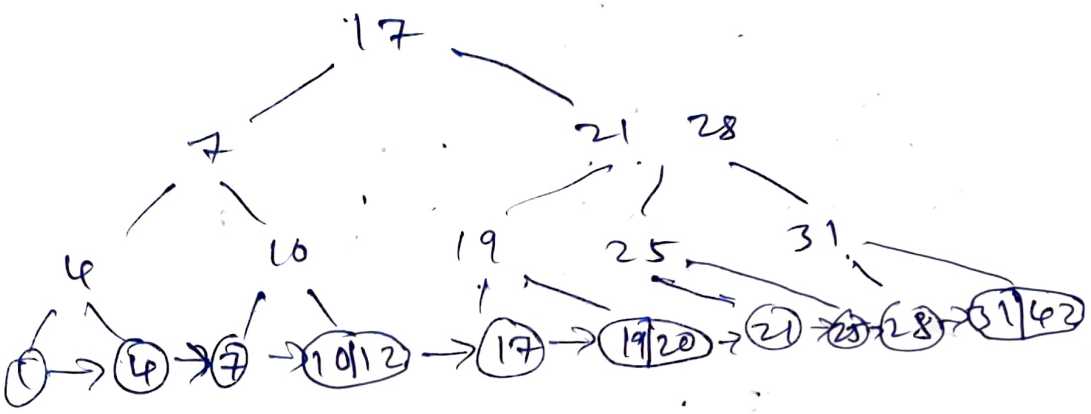
→ 19, ~~30~~



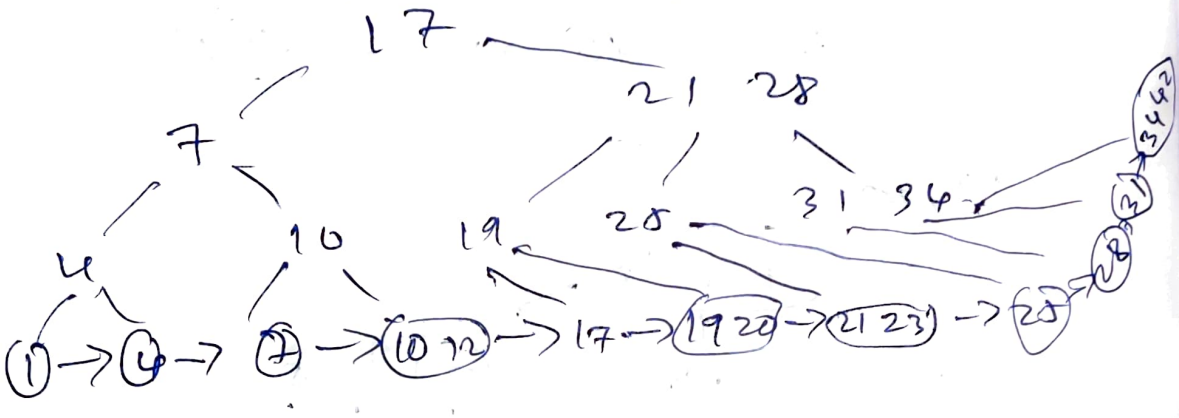
→ 20, 28



→ 42, 12


17
7
21
28
4
10
19
25
31
1
4
7
10 12
17
19 20
21
28
31 42

→ 34 & 23


17
7
21
28
4
10
19
25
31
34
1
4
7
10 12
17
19 20
21 23
25
28
31
34 42

7) 6-bit hash → 2^6 → 64

→ Directory Bucket structure

g = 2

00	01	10	11
i=2	i=2	i=2	i=2
8	148	288	120
204		1064	700
641		44	1586
258			

i) insert 68 => 68 mod 64 = 4

g = 3

000	001	010 011	100	101	110	111
i=3	i=3	i=2	i=2		i=2	
641	8	148	288		120	
258	204		1064		700	
68			44		1586	

ii) Insert 168, 57 → Bucket split, no direct split

g = 3

000	001	010 011*	100 101	110	111
i=3	i=3	i=2	i=2	i=3	i=3
641	8	148	28	1586	120
258	204		1064	48	700
68			4		57

iii) Insert 16, 18, 22, 23

multiple splits

16 mod 64 = 16

18 → 18

22 % 64 → 22

23 → 23

Setting i=3 gives table.

010	011

i=3

i=3

148

16

18

22

23

iv) Multiple splits

setting i=4 & g=4

g=4

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011...

i=3 i=3 i=4 i=4 i=3 i=2

641 8 16 148 — 268

288 204 18 22 2064

68 23 44

1100 1101 1110 1111

i=3 i=3

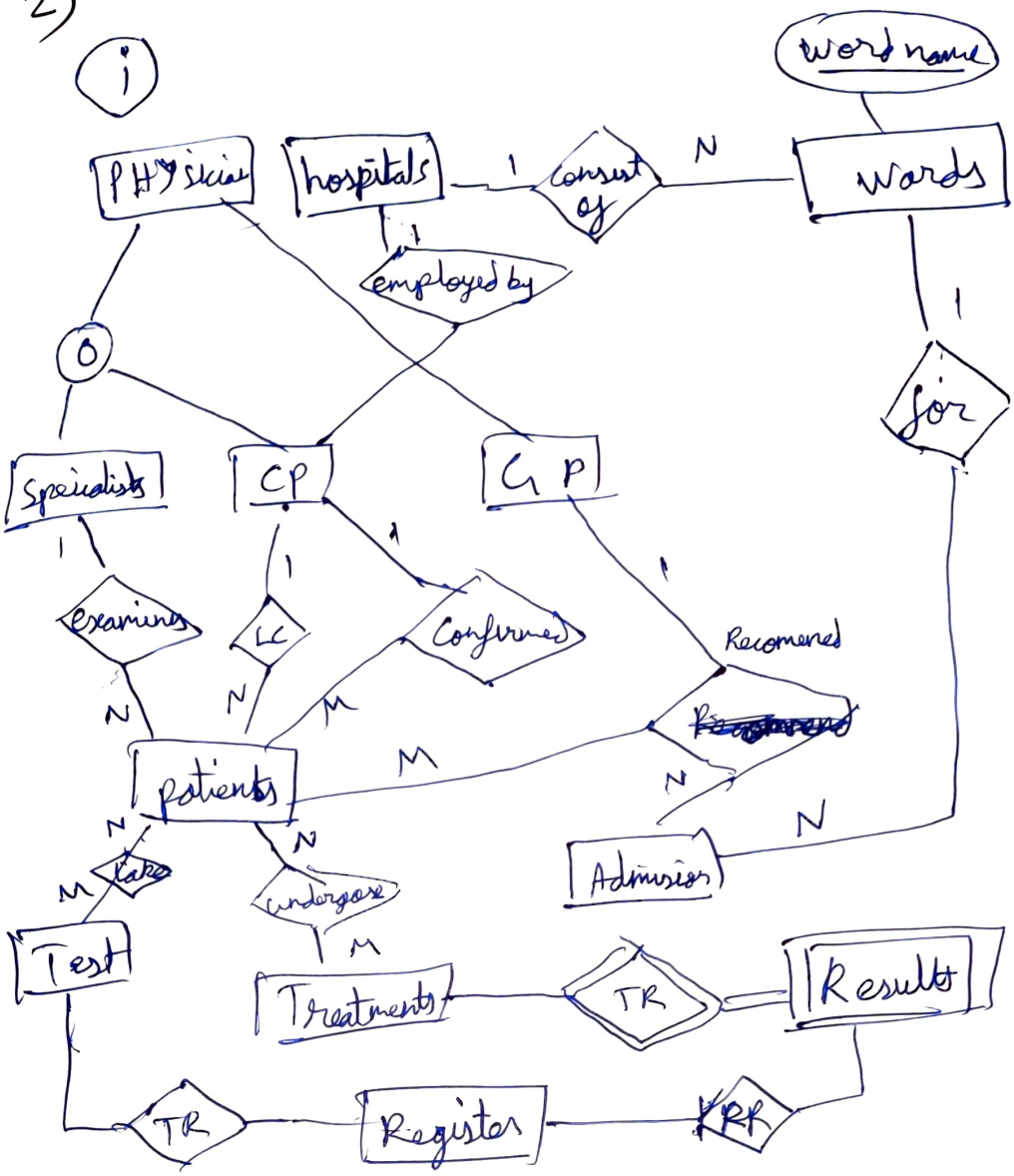
1586 120

48 700

575

2)

(i)



(iii) Select * from Tests; // List of all tests conducted.

select *count(*) from Patients; // no of patients

select Name from Specialists; // names of specialists

Select *, from: admissions; // Admissions tests.