EECS 495
Introduction to Database Systems
Fall 2018
Instructor: Mas-ud Hussain
Solution: Homework Assignment No. 3

Problem No. 1:

Schedule S1:
a) Yes; **b)** Yes; **c)** T1, T2

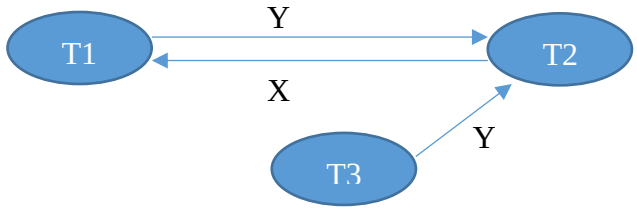
Schedule S2:
a) No; **b)** Yes; **c)** T1, T2

Schedule S1:
a) No; **b)** Yes; **c)** T1, T3, T2 or T3, T1, T2

Problem No. 2:

(a) Strict 2PL with deadlock detection.

Sequence S1:
1. T1 gets an exclusive-lock on X;
2. T2 gets a shared-lock on Y;
3. T2 blocks waiting for an exclusive-lock on X;
4. T1 blocks waiting for an exclusive-lock on Y;
5. T3 blocks waiting for an exclusive-lock on Y;
6. Deadlock; (draw the *waits-for* graph here)



Sequence S2:
1. T1 gets an exclusive-lock on X;
2. T2 blocks waiting for an exclusive-lock on X;
3. T1 already has an exclusive-lock on X;
4. T3 gets an exclusive-lock on Y;
5. T1 already has an exclusive-lock on X;
6. T1 commits and releases lock on X;
7. T2 wakes up to acquire an exclusive-lock on X;
8. T2 already has an exclusive-lock on X;
9. T2 commits and releases lock on X;
10. T3 commits and releases lock on Y;
11. No deadlock.

(b) Strict 2PL with timestamps used for deadlock prevention. (Show for wait-die only)

Sequence S1:
1. T1 gets an exclusive-lock on X;
2. T2 gets a shared-lock on Y;
3. T2 tries to get an exclusive-lock on X. Since T2 has a lower priority, it will be aborted;

4. T1 gets an exclusive-lock on Y;
5. T3 tries to get an exclusive-lock on Y. Since T3 has a lower priority, it will be aborted;
6. T1 commits and releases X and Y;
7. T2 restarts, acquires shared-lock on Y;
8. T2 gets an exclusive-lock on X;
9. T3 restarts, tries to get an exclusive-lock on Y, but is aborted;
10. T2 already has a shared-lock on Y;
11. T2 commits and releases X and Y;
12. T3 restarts, gets an exclusive-lock on Y;
13. T3 commits and releases Y;
14. No deadlock.

Sequence S2:

1. T1 gets an exclusive-lock on X;
2. T2 tries to get an exclusive-lock on X. Since T2 has a lower priority, it will be aborted;
3. T1 already has an exclusive-lock on X;
4. T3 gets an exclusive-lock on Y;
5. T1 already has an exclusive-lock on X;
6. T1 commits and releases lock on X;
7. T2 restarts and gets a shared-lock on X;
8. T2 commits and releases lock on X;
9. T3 commits and releases lock on Y;
10. No deadlock.

(c) Rigorous (i.e., with locks held until end-of-transaction) 2PL.

Sequence S1:

1. T1 gets locks on X and Y;
2. T2 blocks waiting for lock on X and Y;
3. T3 blocks waiting for lock on Y;
4. T1 commits and releases X and Y;
5. T2 gets locks on X and Y, commits, and releases both locks;
6. T3 gets lock on Y, commits, and releases Y;
7. No deadlock.

Sequence S1:

1. T1 gets lock on X;
2. T2 blocks waiting for lock on X;
3. T3 gets lock on Y;
4. T1 commits and releases X;
5. T2 gets lock on X;
6. T2 commits and releases X;
7. T3 commits and releases Y;
8. No deadlock.

Problem No. 3:

(a)

MRU:

Slot 1	Slot 2	Slot 3	M or H?
1			M
1	2		M
1	2	3	M
1	2	4	M
1	2	4	H
1	2	4	H
1	2	4	H
1	5	4	M
1	3	4	M
1	3	4	H

Hit rate: 40%

(b)

Clock Replacement:

Slot 1	Slot 2	Slot 3	M or H?
1			M
1	2		M
1	2	3	M
4	2	3	M
4	2	3	H
4	2	1	M
4	2	1	H
5	2	1	M
5	2	3	M
1	2	3	M

Hit rate: 20%

Which strategy do you recommend for this workload?

Answer: Most Recently Used (MRU).

Problem No. 4:

(a)

It is not in BCNF because there are non-trivial functional dependencies Service Time → Pay and Favorite Record → Favorite Artist where the left side attributes are not superkeys.

(b)

STEP 1
Start Schema:
(Employee ID, Favorite Record, Service Time, Pay, Favorite Artist)
FD: Service Time $\rightarrow$ Pay
End Schemas:
(Employee ID, Favorite Record, Service Time, Favorite Artist)
(Service Time, Pay)

STEP 2
Start Schemas:
(Employee ID, Favorite Record, Service Time, Favorite Artist)
(Service Time, Pay)
FD: Favorite Record $\rightarrow$ Favorite Artist
End Schemas:
(Employee ID, Favorite Record, Service Time)
(Favorite Record, Favorite Artist)
(Service Time, Pay)