**Q1. (20 pts.)** Consider the relation R(A, B, C, D, E, F, G) with functional dependencies

F={AB → C, B → F, E → G, A → DE }.

(a) Find all keys of this relation. Do not report a superkey that is not a (minimal) key. **(3 points)**

$AB^+ = ABCDEFG$

AB is the key

(b) Is R in 3NF? Justify your answer. **(2 points)**

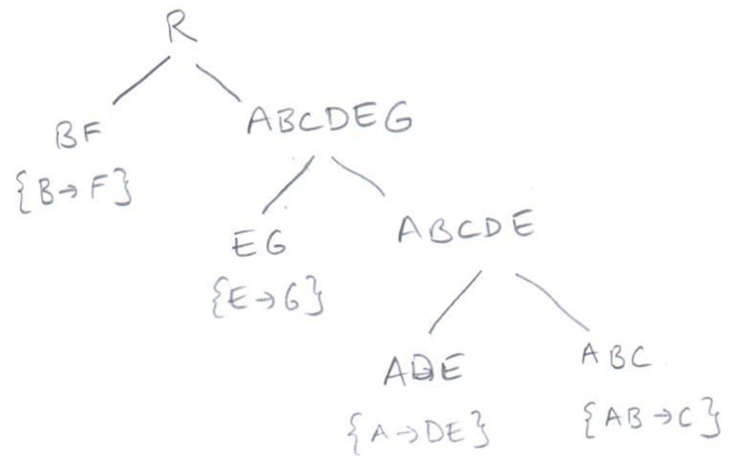
– not part of a key

No. B→F E→G A→DE all violate 3NF

not a superkey

(c) Is this table in BCNF? If the answer is yes explain why it is. If your answer is no, decompose R into a collection of BCNF relations and state whether the decomposition is lossless-join and dependency preserving. **(15 points)**

No.



It is lossless-join & dependency preserving decomposition

**Q2. (20 pts.)** Consider a relation R with attributes {A, B, C, D, E, F, G, H} and the set of functional dependencies F = {G → FD, E → D, GD → CE, BD → A }

a) Find the minimal cover of the given set of functional dependencies F. (**10 points**)
b) Decompose the relation into a collection of 3NF relations using the minimal cover found in part a). (**5 points**)
c) Is your decomposition also BCNF, lossless-join and dependency preserving? Explain. (**3 points**)
d) In designing a relational database schema why might we choose a non BCNF but 3NF design? (**2 points**)

a) Minimal Cover

G → F
E → D
G → C
G → E
BD → A

b) G → CEF
E → D
BD → A

⇒ Decomposition
R1 (G, C, E, F)
R2 (E, D)
R3 (B, D, A)
R0 (G, B, H)

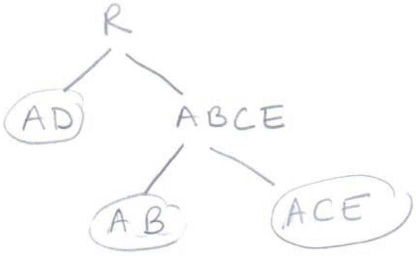
key = GBH

**Q3. (8 pts.)** Consider the relation R(A,B,C,D,E) with multi-valued dependencies MD = { A→→ B, AB→→ C} and functional dependencies FD = {A →D, AB→ E}.

a) Find key(s) for the relation? (**3 points**)
b) Decompose the relation into a collection of relation schemas in 4NF. (**5 points**)

a) ABC

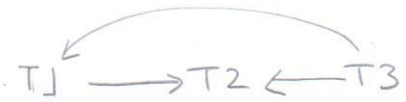
b)



**Q4. (12 pts.)** For each of the following schedules determine whether it is conflict-serializable and/or view-serializable. Justify your answers.

a) $W_3(C)\ W_2(B)R_2(B)R_3(C)W_1(C)R_2(C)$

Is it conflict-serializable? Why or why not? If it is give the equivalent serial schedule.



No cycles

It is conflict serializable (CS)

T3, T1, T2

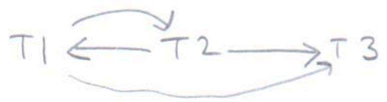
(VS)

Is it view-serializable? Why or why not? If it is give the equivalent serial schedule.

Because it is CS it is also VS.

b) $R_2(C)\ W_1(C)W_2(C)R_1(B)W_3(C)$

Is it conflict-serializable? Why or why not? If it is, give the equivalent serial schedule.



cycle. It is not C.S.

Is it view-serializable? Why or why not? If it is, give the equivalent serial schedule.

It is V.S equivalent to:

$T_1, T_2, T_3$

or

$T_2, T_1, T_3$

| B | C |
|---|---|
| $r_1(B)$ | $r_2(C)$ |
| | $w_1(C)$ |
| | $w_2(C)$ |
| | $w_3(C)$ |

**Q5. (20 pts.)** Consider the following schedule involving three transactions T1, T2 and T3:

| Time | T1 | T2 | T3 |
|---|---|---|---|
| 1 | R(C) | | |
| 2 | | | R(B) |
| 3 | R(A) | | |
| 4 | | R(C) | |
| 5 | | | W(B) |
| 6 | W(A) | | |
| 7 | | R(B) | |
| 8 | | | R(A) |
| 9 | | W(C) | |
| 10 | | W(B) | |
| 11 | Commit | | |
| 12 | | Commit | |
| 13 | | | Commit |

a) Is this schedule a recoverable schedule? Justify your answer. (**5 points**)

$R_2(B)$ is a dirty read. So it is not a recoverable schedule.

b) Can this schedule be produced by the **strict two-phase locking protocol?** Justify your answer. (**10 points**)

No. Because T2 & T3 are suspended at time 7 & 8 respectively due to lock conflicts.

Lock Table

| | Locks | wait-for | |
|---|---|---|---|
| A | ~~$S_1$~~ $X_1$ | $T3 \to T1$ | at time 8 because of $R_3(A)$ |
| B | ~~$S_3$~~ $X_3$ | $T2 \to T3$ | at time 7 because of $R_2(B)$ |
| C | $S_1, S_2$ | | |

T3 can resume at time 11 after T1 commits.

T2 can resume after T3 commits

c) Assuming T1 is the oldest and T3 is the youngest transaction, can this schedule be produced by the **strict two-phase locking protocol with wait-die deadlock prevention scheme?** Justify your answer. (**5 points**)

No

$T1 < T2 < T3$.

Lock Table

| | Locks | |
|---|---|---|
| A | ~~$S_1$~~ $X_1$ | $R_3(A)$ @8 T3 wants to read A at time 8, but it is younger so it aborts. |
| B | ~~$S_3$~~ ~~$X_3$~~ | $R_2(B)$ @7 T2 wants to read B. at time 7 T2 is older so it waits $T2 \to T3$. |
| C | $S_1, S_2$ | |

T2 resumes after T3 aborts by acquiring shared lock on B

But when it wants to write C at time 9 it aborts too.

**Q6 (20 pts.)** Consider the following relational table:

**Enroll**(student_id, course_no, status)

a) You want to ensure that the number of courses that a student is enrolled should not be more than 7. Define an assertion to achieve this.

6 pts

```
Create assertion Q6a check (not exists
   (   select student_id
       from Enroll
       group by student_id
       having count (*) > 7
  ));
```

b) Define a trigger to achieve the same constraint in part a).

8 pts

```
Create trigger Q6b
After insert on Enroll
referencing new row as new
for each row
when (
      (select count(*) from Enroll
       where student_id = New.student_id) > 7
)
Rollback;
```

c) Suppose there is a functional dependency such that student_id → status. Write an assertion to enforce this functional dependency.

6 pts

```
Create assertion Q6c check (not exists
   (select student_id
    from Enroll
    group by student_id
    having count (distinct status) > 1
  ))
```

**Q3. (25 pts.)** Consider a database with data items {A, B, C, D}. The system has the following log records. Note that an entry <T, X, old, new> means transaction T changes the value of X from old to new. In the following log, DPT represents the Dirty Page Table and TT represents the Transaction Table.

1. <T2, C, 10, 11>
2. <T2 commit>
3. <T1, B, 20, 21>
4. <begin checkpoint>
   DPT = (B, 3)
   TT = (T1, running, 3)
   <end checkpoint>
5. <T3, D, 30, 31>
6. <T4, C, 11, 41>
7. <T4 commit>
8. <T3, A, 50, 51>
9. <T1, D, 31, 32>
10. <T5, A, 51, 52>
11. <T3 commit>
12. <T6, B, 21, 33>
13. <T5 commit>
    CRASH

a) What is the smallest Log sequence number (LSN) accessed during the Analysis phase? (2 pts.)
Answer: 4

b) Fill in the contents of the **Dirty Page Table** and the **Transaction Table** at the end of the Analysis phase. (8 pts.)

**DPT**

| PageID | RecLSN |
|---|---|
| B | 3 |
| D | 5 |
| C | 6 |
| A | 8 |

**TT**

| XID | Status | LastLSN |
|---|---|---|
| T1 | running | ~~3~~ 9 |
| T3 | ~~running~~ commit | ~~5~~ ~~8~~ 11 |
| T4 | ~~running~~ commit | ~~6~~ 7 |
| T5 | ~~running~~ commit | ~~10~~ 13 |
| T6 | running | 12 |

c) What is the first Log sequence number (LSN) to be redone during the Redo phase? (2 pts.)
Answer: 3

d) List all possible values of A, B, C and D on disk at the time of crash (after action 13)? Assume 3 buffer pages are available. (4 points)

Disk ( 50 ~~20~~ ~~11~~ 41 30 )
A B=21 C D

Memory
B ~~21~~ D ~~31~~ C ~~41~~
A ~~51~~ 52 32 33 B

Disk:
A = 50
B = 21
C = 41
D = 30

e) What log entries are written to the log during Undo phase? (5 points)

ignoring 'End' records, toUndoList: {12, 9}
abort T1, abort T6

~~#4~~. CLR: T6 12 -1
End T6
CLR: T1 9 3
CLR T1 3 -1
End T1

f) What are the values of A, B, C, D on disk at the end of crash recovery? (4 points)

A ~~52~~ D ~~32~~ 31 B ~~33~~ 21
B 20

Disk:
A = 52
B = 20
C = 41