Q1. Consider schedules H1 and H2 given below:
H1: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y); c1; c2; c3;
H2: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y); c3; c2;

Determine whether each schedule is conflict serializable, cascadeless, recoverable or not. [5]

Answer:

In schedule H1,

As C3 comes after C2, the schedule will be not cascadeless and not recoverable. However it is conflict serializable..

H2:Non Conflict-serializable, No dirty read hence recoverable and cascadeless shecdule

(T1→ T2 due to Z, T3 → T1 due to X, T3 → T2 due to Y) in both schedules.
Recoverable Schedule: The transaction which does uncommitted read operation should not commit before the commit/rollback of the transaction which updated that data item.

Cascadeless Schedule: No uncommitted read is allowed.

(2.5 marks for each schedule) (only 1 mark if without any explanation)

Q2. For the lock requests in Tables below, determine which lock will be granted or blocked by the lock manager.
Does there exist a deadlock in the lock requests in Tables, explain why or why not.
To prevent deadlock, Determine which lock request will be granted, blocked or aborted. IF we use a lock manager that adopts the Wait-Die policy and wound- Wait policy. We assume that in terms of priority: T1> T2> T3> T4. [5]

Time	t1	t2	t2	t4	t5	t6	t7	t8
T1	S(A)		S(B)					
T2		X(B)						X(D)
T3				S(C)	X(D)		X(A)	
T4						X(C)		

Lock Requests: [1 mark]

Deadlock detection: [1 mark]

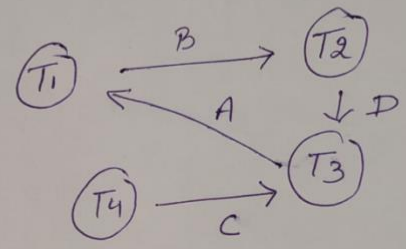
Wait- die: [1.5 mark]

Wound-Wait: [1.5 mark]

Q2.

S(A) at t_1 – granted
X(B) at t_2 – ~~blocked~~ granted
S(B) at t_3 – blocked
S(C) at t_4 – granted
X(D) at t_5 – granted
X(C) at t_6 – blocked
X(A) at t_7 – blocked
X(D) at t_8 – blocked

Deadlock exists because there is a cycle ($T_1 \to T_2 \to T_3 \to T_1$)



Wait-Die Policy :-

S(A) at t_1 : granted
X(B) at t_2 : granted
S(B) at t_3 : blocked (T_1 wait for T_2)
S(C) at t_4 : granted
X(D) at t_5 : granted
X(C) at t_6 : Abort. (T_4 dies)
X(A) at t_7 : abort (T_3 dies)
X(D) at t_8 : granted

Wound-wait Policy.

S(A) at t_1 – granted
X(B) at t_2 – granted
S(B) at t_3 – granted (T_1 wound T_2)
S(C) at t_4 – granted
X(D) at t_5 – granted
X(C) at t_6 – blocked
X(A) at t_7 – blocked
X(D) at t_8 – (T_2 is dead).

Q3. Write a trigger total_salary to maintain a derived column totsal that stores total salary of all members in a department(after each operation of insert, update or delete) [4]

Answer:

```
Create or replace Trigger total_salary
After delete or insert or update of deptno, sal on emp
For each row
Begin
If deleting or updating and :old.deptno=: new.deptno then
Update dept
Set totsal= totsal-:old.sal
Where deptno=: new.deptno
End if;
If inserting or updating and :old.deptno=: new.deptno then
Update dept
Set totsal= totsal+:new.sal
Where deptno=: new.deptno
End if;
If  updating and :old.deptno=: new.deptno
And :old.sal!=:new.sal then
Update dept
Set totsal= totsal-:old.sal+:new.sal
Where deptno=: new.deptno
End if;
End;
```

Marking: (insert, update or delete- 1 mark each + Trigger body- 1 mark]

Q4. Consider the relational schema given below, where EId of the relation dependent is a foreign key referring to empId of the relation employee. Assume that every employee has at least one associated dependent in the dependent relation.

employee (empId, empName, empAge)
dependent(depId, eId, depName, depAge)

Write the relational algebra and its equivalent SQL query

1. To produces *empIds* of those employees who have at least one dependent with age greater than or equal the employee's age. [2]
2. To produces *empIds* of employees whose age is greater than that of all of his/her dependents.[2]

3. To insert new tuple into dependent relation. Is it allowed to insert new dependent information without having matching empId in employee relation. [2]

a)

RA->

$$\Pi_{empId}(employee \bowtie_{(empId = eID) \wedge (empAge \leq depAge)} dependent)$$

(1.5 Mark)

SQL->

Select empid from Employee, dependent

Where empid=eid AND emp_age<=dept_Age

(1.5 Mark)

b)

RA->

$$\Pi_{empId}(employee) - \Pi_{empId}(employee \bowtie_{(empId = eID) \wedge (empAge \leq depAge)} dependent)$$

(1.5 Mark)

SQL

Select empid from employee

Where empid NOT IN (Select empid from Employee, dependent

Where empid=eid AND emp_age<=dept_Age)

(1.5 Mark)

c)

RA-

dependent <- dependent U {“123”, “E1”, “XYZ”, “33” }

SQL – Insert into dependent values (“123”, “E1”, “XYZ”, 33);

(1.5 Mark)

No , it is not allowed to insert new dependent information without having matching empid in employee relation because of *Integrity constraints.*

(1.5 Mark)

Q5. Consider the relation R(A,B,C,D,E) with the following set of FDs:

AB->B,

A->C,

CD->E,

B->D,

E->A.

Answer the following questions:

1. Find the minimal cover of R.
2. What is the highest normal form the relation R satisfies? Convert it upto BCNF Normal form. [5]

Answer:

Q5. $AB \to B$, $A \to C$, $CD \to E$, $B \to D$, $E \to A$.

(1) Minimal Cover

Step 1 – Reduce R.H.S side. – Already reduced

Step 2 – Remove extraneous attributes

$AB \to B$

A is extraneous as $B^+ = B$.

Thus, $B \to B$.

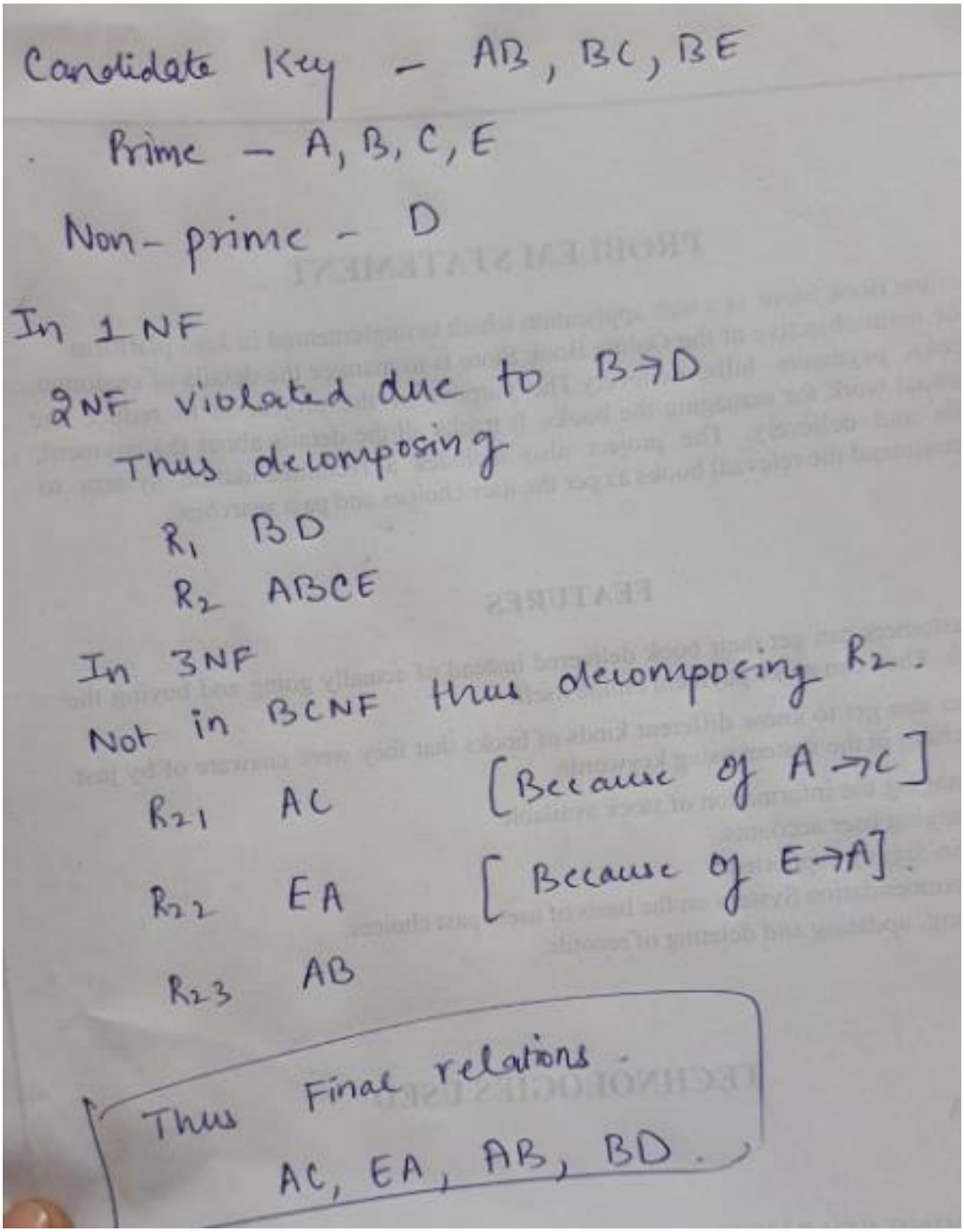
Trivial FDs are not included.

Thus removing $AB \to B$.

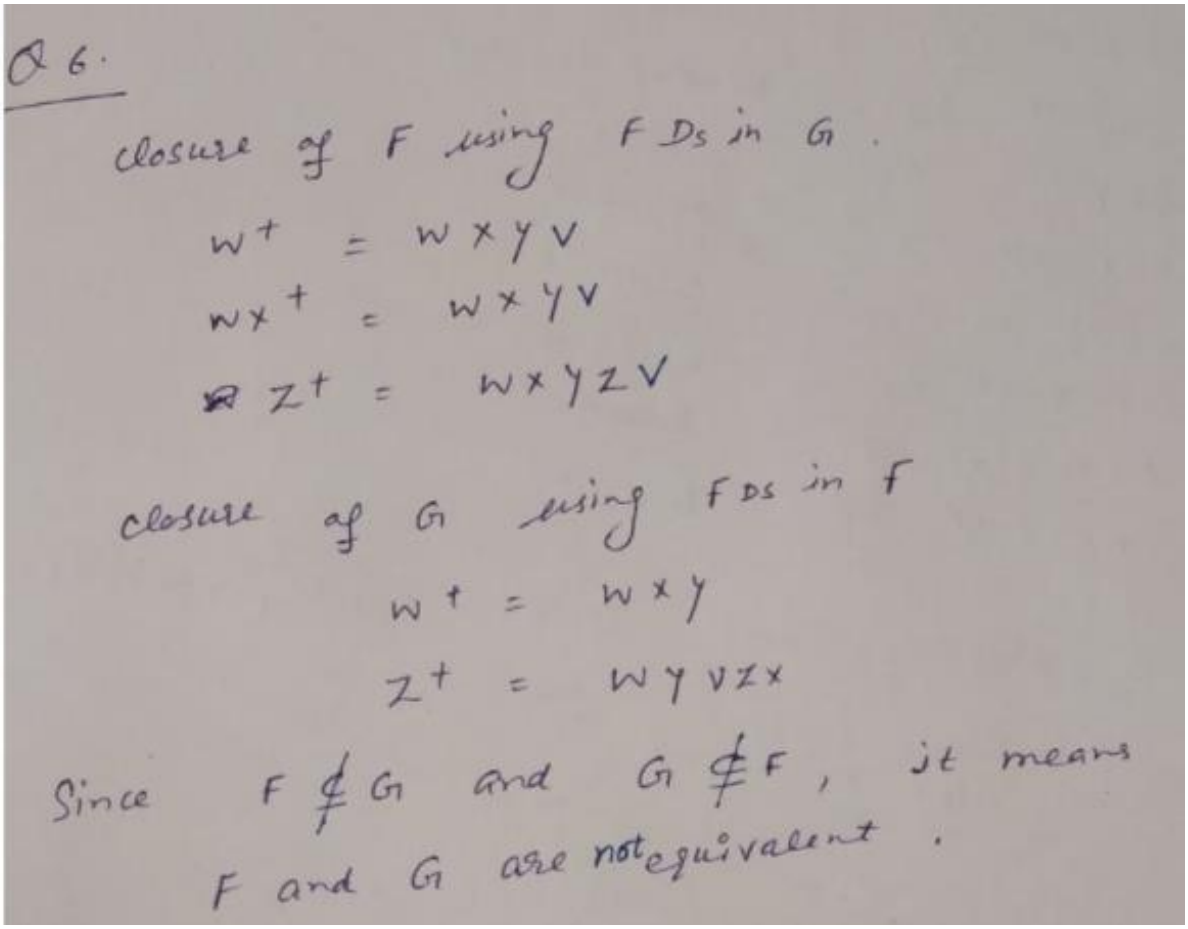
Step 3 – Removing redundant FDs.

No FD is redundant.

Final Minimal Cover – $A \to C$, $CD \to E$, $B \to D$, $E \to A$



Q.6. Suppose, a relational schema R (v w x y z) and set of functional dependencies F and G are as follow: F : { w →x, wx →y, z →wy, z →v } G :{ w →xy, z →wx , X → v}. Check the equivalency of functional dependencies F and G. [3]



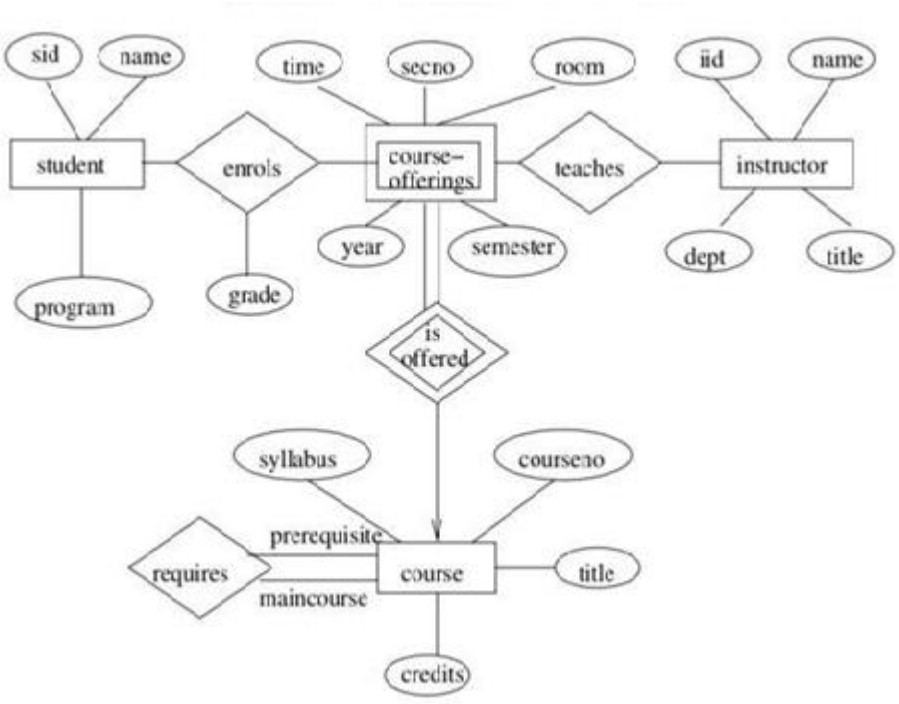
Q8 . A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's office.

Make the following assumptions:

i) A class meets only at one particular place and time. This E-R diagram cannot model a class meeting at different places at different times.

ii) There is no guarantee that the database does not have two classes meeting at the same place and time.

Also, identify the weak and strong entity sets.



Marking: [Each Entity and its attributes- 1 mark each, Weak Entity identification- 1 mark]