Question 1 :

Consider a relation $R(A,B,C,D,E,G,H,I,J)$ and its FD set $F = \{A \rightarrow DE, B \rightarrow GI, E \rightarrow CD, CE \rightarrow ADH, H \rightarrow G, AH \rightarrow I\}$.

1) Check if $A \rightarrow I \in$ F⁺. (3 marks)

During the process of scan in F:

$A^+ = \{ A \}$

$A^+ = \{ A, D, E \}$

$A^+ = \{ A, D, E, C, H \}$

$A^+ = \{ A, D, E, C, H, G, I \}$

→ A → I → A → I ∈ F^+

2) Find a candidate key for R. (3 marks)

Firstly, we let X = { A, B, C, E, H, J } is a super key because these attributes(A, B, C, E, H) appear in the left hand side of F and attribute J does not appear in both sides of F.

Try to remove : A

$\{ B, C, E, H, J \}^+ = \{ A, B, C, D, E, G, H, I, J \} = R$;

Try to remove : B

$\{ C, E, H, J \}^+ = \{ A, C, D, E, G, H, I, J \}$ =/= R;

∴ we cannot remove B.

Try to remove : C

$\{ B, E, H, J \}^+ = \{ A, B, C, D, E, G, H, I, J \} = R$;

Try to remove : E

$\{ B, H, J \}^+ = \{ B, G, H, I, J \}$ =/= R;

∴ we cannot remove E.

Try to remove : H

$\{ B, E, J \}^+ = \{ A, B, C, D, E, G, H, I, J \} = R$;

Try to remove : J

J is an attribute that does not appear in any side of F, so J cannot be removed.

Finally, we find a candidate key : { B, E, J }

3) Determine the highest normal form of R with respect to F. Justify your answer.

(3 marks)

The highest normal form of R is the 1NF.

Justify process :

∵ E → CD ∴ E → C and E → D;

∵ attribute D is not part of a candidate key, so D is non-prime.

This is not 2NF since E → D, attribute C is not prime, and { B, E, J } is a key, making attribute D partially dependent on a key.

4) Find a minimal cover F_m for F. (3 marks)

Step1 :

F' = { A → D, A → E, B → G, B → I, E → C, E → D, CE → A, CE → D, CE → H, H → G,

$AH \rightarrow I$ }

Step2 :

For $CE \rightarrow A$:

We know $C^+ = \{ C \}$, and $E^+ = \{ C, D, E, A, H, G, I \}$

So $E \rightarrow A$ can be inferred by F';

Same reason for $E \rightarrow D$ and $E \rightarrow H$ can be inferred by F';

for $AH \rightarrow I$:

We know $A^+ = \{ A, D, E, C, H, G, I \}$and $H^+ = \{ H, G \}$;

Thus, $A \rightarrow I$ can be inferred by F';

$F'' = \{ A \rightarrow D, A \rightarrow E, A \rightarrow I, B \rightarrow G, B \rightarrow I, E \rightarrow C, E \rightarrow D, E \rightarrow A, E \rightarrow H, H \rightarrow G \}$

Step3 :

$D \in A^+|_{F'' - \{A \rightarrow D\}}$; thus $A \rightarrow D$ can be inferred by F'' - {A→B} and we remove A→B;

$A^+|_{F''' - \{A \rightarrow E\}} = \{ A \}$, so $A \rightarrow E$ is not inferred by F''' - {A→E} and it is not redundant;

Same reason for the remaining fuction dependencies, all of them are not redundant;

Thus, we finally get F_{min}:

$F_{min} = F''' = \{ A \rightarrow E, A \rightarrow I, B \rightarrow G, B \rightarrow I, E \rightarrow C, E \rightarrow D, E \rightarrow A, E \rightarrow H, H \rightarrow G \}$

5) Decompose into a set of 3NF relations if it is not in 3NF step by step. Make sure your decomposition is dependency-preserving and lossless-join. (3 marks)

We know that R (A, B, C, D, E, G, H, I, J);

$F_{min} = \{ A \rightarrow E, A \rightarrow I\ B \rightarrow G, B \rightarrow I, E \rightarrow C, E \rightarrow D, E \rightarrow A, E \rightarrow H, H \rightarrow G \}$

Candidate key is : (B, E, J)

So we have a decomposition of R such that:

R_1 = (A, E, I) R_2 = (B, G, I) R_3 = (E, A, C, D, H) R_4 = (H, G)

R_5 = (B, E, J)

Clearly, R has a set of FD of F_{min} , so this decomposition is dependency-preserving.

Finally, test for lossless join and initialize the matrix below:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	b	b	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	b	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	b	b	a	b	b	b	a

Test A → E:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	b	b	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	b	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	b	b	a	b	b	b	a

Test A → I:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	b	b	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	b	b	a	b	b	b	a

Test B → G:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	b	b	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	b	b	a	a	b	b	a

Test B → I:

decomposition	A	B	C	D	E	G	H	I	J

R_1 = (A, E, I)	a	b	b	b	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	b	b	a	a	b	a	a

Test E → C:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	a	b	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	a	b	a	a	b	a	a

Test E → D:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	a	a	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	b	a	a	a	a	a	b	a	a

Test E → A:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	a	a	a	b	b	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	a	a	a	a	a	a	b	a	a

Test E → H:

decomposition	A	B	C	D	E	G	H	I	J
R_1 = (A, E, I)	a	b	a	a	a	b	a	a	b
R_2 = (B, G, I)	b	a	b	b	b	a	b	a	b
R_3 = (E, A, C, D, H)	a	b	a	a	a	b	a	a	b
R_4 = (H, G)	b	b	b	b	b	a	a	b	b
R_5 = (B, E, J)	a	a	a	a	a	a	a	a	a

By testing E→H, we can get a row is made up entirely of "a" symbols.

So this decomposition is lossless and dependency-preserving.

Question 2:

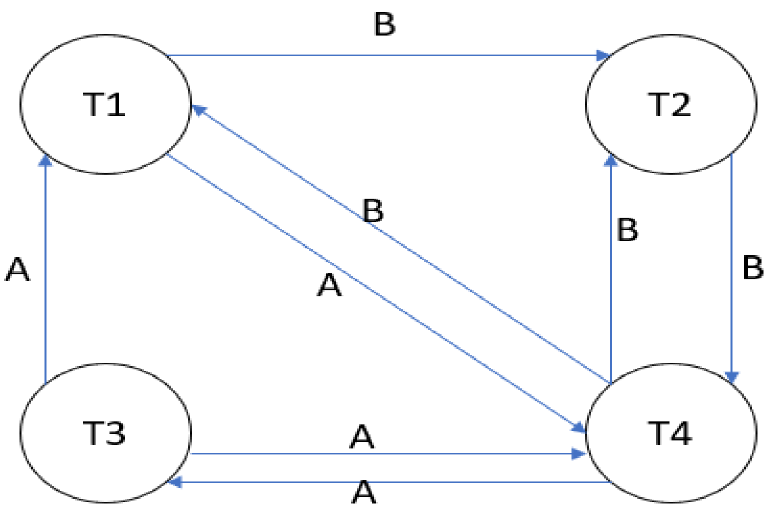
Consider the schedule below. Here, R(*) and W(*) stand for 'Read' and 'Write', respectively. T_1 , T_2 , T_3 and T_4 represent four transactions and t_i represents a time slot.

Time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
T_1	R(B)					R(A)	W(B)				W(A)	
T_2								R(B)				W(B)
T_3			R(A)	W(A)								
T_4		R(A)			W(A)				R(B)	W(B)		

Each transaction begins at the time slot of its first Read and commits right after its last Write (same time slot).

Regarding the following questions, give and justify your answers.

1) Is the transaction schedule conflict serialisable? Give the precedence graph to justify your answer. (4 marks)



This schedule is not conflict serialisable, because the corresponding precedence graph is cyclic.(there is a cycle such that T1→ T2→ T4→ T3→ T1).

So it is non-serializable.

2) Give a serial schedule of these four transactions. (3 marks)

Time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
T_1	R(B)	R(A)	W(B)	W(A)								
T_2					R(B)	W(B)						
T_3							R(A)	W(A)				
T_4									R(A)	W(A)	R(B)	W(B)

3) Lock the transactions and according to the simple locking scheme. You only need to consider the order of the operations, not the timestamps. (3 marks)

The table below is the process of locking transactions T_1 and T_2 according to the simple locking scheme :

T_1	T_2
Write_lock(B)	
Read(B)	
Write_lock(A)	
Read(A)	
Write(B)	
Unlock(B)	
	Write_lock(B)
	Read(B)
Write(A)	
Unlock(A)	
	Write(B)
	Unlock(B)