COMP9311 - Database System Assignment 2

Question 1:

R (A, B, C, D, E, G, H, I, J)

$F = \{ \quad AD \rightarrow B,$

$BD \rightarrow G,$

$BE \rightarrow I,$

$AE \rightarrow DI,$

$AI \rightarrow E,$

$AEI \rightarrow C\}.$

(1) To Prove→ $AB \rightarrow G$

AB+ = AB

Since G is not present in the closure of AB, it is not derivable, or it cannot be proved.

(2) Find all the candidate keys for R.

Let X = { A, B, C, D, E, G, H, I, J }

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

We can remove B → $\{ A, C, D, E, G, H, I, J\}^+ = \{ A, B, C, D, E, G, H, I, J \}$

We can remove C → $\{ A, D, E, G, H, I, J\}^+ = \{ A, B, C, D, E, G, H, I, J \}$

We can remove D → $\{ A, E, G, H, I, J\}^+ = \{ A, B, C, D, E, G, H, I, J \}$

We can remove G → $\{ A, E, H, I, J\}^+ = \{ A, B, C, D, E, G, H, I, J \}$

We can remove G → $\{ A, E, H, I, J\}^+ = \{ A, B, C, D, E, G, H, I, J \}$

Either E, or I can be removed to form a candidate key.

{A, H, J} will be present in every candidate key as none of them are dependent on another attribute (i.e. they are independent attribute)

{A, E, H, J}$^+$ = {A, B, C, D, E, G, H, I, J} It is a candidate key.

{A, I, H, J}$^+$ = {A, B, C, D, E, G, H, I, J} It is a candidate key.

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

{A, E, I, H, J} is not a candidate key but a super key since it is the union of both the candidate keys. Also, candidate keys are supposed to be minimal basically, if E or I is taken out, it will result in becoming a candidate key.

(3) Determine the highest normal form of R with respect to F.

Using CK = {{A, E, H, J}, {A, I, H, J}}

Prime attributes = {A, E, I, H, J}

Non-prime attributes = {B, C, D, G}

The highest form for relation R is First Normal Form (1NF) since all the attributes are atomic.

The relation is not in 2NF because {A, E} being a proper subset of candidate key {A, E, H, J} is determines a non-prime attribute D in functional dependency $AE \rightarrow DI$.

Meaning D is partially dependent on candidate key {A, E, H, J}. This violates the rule of 2NF where a relation schema R is in 2NF if every nonprime attribute A in R is not partially dependent on any key of R.

(4) Find a minimal cover F_m for F

Step 1: Reduce Right

F' = {AD → B, BD → G, BE → I, AE → D, AE → I, AI → E, AEI → C}

Step 2: Reduce Left

For AD → B: D$^+$ = {D} meaning D → B is not inferred by F' thus it can't replace AD → B

A$^+$ = {A} meaning A → B is not inferred by F' thus it can't replace AD → B

For BD → G: D$^+$ = {D} meaning D → G is not inferred by F' thus it can't replace BD → G

B$^+$ = {G} meaning B → G is not inferred by F' thus it can't replace BD → G

For BE → I: E$^+$ = {E} meaning E → I is not inferred by F' thus it can't replace BE → I

B$^+$ = {B} meaning B → I is not inferred by F' thus it can't replace BE → I

For AE $\rightarrow$ D: $A^+ = \{A\}$ meaning A $\rightarrow$ G is not inferred by F' thus it can't replace AE $\rightarrow$ D

$E^+ = \{E\}$ meaning E $\rightarrow$ G is not inferred by F' thus it can't replace AE $\rightarrow$ D

For AE $\rightarrow$ I: $E^+ = \{E\}$ meaning E $\rightarrow$ I is not inferred by F' thus it can't replace AE $\rightarrow$ I

$A^+ = \{A\}$ meaning A $\rightarrow$ I is not inferred by F' thus it can't replace AE $\rightarrow$ I

For AI $\rightarrow$ E: $I^+ = \{I\}$ meaning I $\rightarrow$ E is not inferred by F' thus it can't replace AI $\rightarrow$ E

$A^+ = \{A\}$ meaning A $\rightarrow$ E is not inferred by F' thus it can't replace AI $\rightarrow$ E

For AEI $\rightarrow$ C: $EI^+ = \{E, I\}$ meaning EI $\rightarrow$ C is not inferred by F' thus it can't replace AEI $\rightarrow$ C

$AE^+ = \{A, E, I, D, B, G\}$, AE $\rightarrow$ C is not inferred by F', thus it can't replace AEI $\rightarrow$ C

F" = {AD $\rightarrow$ B, BD $\rightarrow$ G, BE $\rightarrow$ I, AE $\rightarrow$ D, AE $\rightarrow$ I, AI $\rightarrow$ E, AEI $\rightarrow$ C}

Step 3: Reduce Redundancy

$AD^+ |_{F'' - \{AD \rightarrow B\}} = \{A, D\}$; AD $\rightarrow$ B is not redundant since it's not inferred by F" – {AD $\rightarrow$ B}

$BD^+ |_{F'' - \{BD \rightarrow G\}} = \{B, D\}$; BD $\rightarrow$ G is not redundant since it's not inferred by F" – {BD $\rightarrow$ G}

$BE^+ |_{F'' - \{BE \rightarrow I\}} = \{B, E\}$; BE $\rightarrow$ I is not redundant since it's not inferred by F" – {BE $\rightarrow$ I}

$AE^+ |_{F'' - \{AE \rightarrow D\}} = \{A, E, I\}$; AE $\rightarrow$ D is not redundant since it's not inferred by F" – {AE $\rightarrow$ D}

$AE^+ |_{F'' - \{AE \rightarrow I\}} = \{A, E, D, B, G, I, C\}$; AE $\rightarrow$ I is redundant, so it can be removed from F" to get F'''

$AI^+ |_{F'' - \{AI \rightarrow E\}} = \{A, I\}$; AI $\rightarrow$ E is not redundant since it's not inferred by F''' – {AI $\rightarrow$ E}

$AEI^+ |_{F'' - \{AEI \rightarrow C\}} = \{A, E, I, D, B, G\}$; AEI $\rightarrow$ C is not redundant since it's not inferred by F''' – {AEI $\rightarrow$ C}

F_m = F''' = {AD $\rightarrow$ B, BD $\rightarrow$ G, BE $\rightarrow$ I, AE $\rightarrow$ D, AI $\rightarrow$ E, AEI $\rightarrow$ C}

(5) Regarding F, does the decomposition R1 = {ABCDJ}, R2 = {BDGI}, R3 = {BCEH} of R satisfy the lossless join property?

R1 = {ABCDJ},

R2 = {BDGI},

R3 = {BCEH}

For BD → G

	A	***B***	***C***	***D***	***E***	***G***	***H***	***I***	***J***
R1	a	a	a	a	b	~~b~~ a	b	b	a
R2	b	a	b	a	b	a	b	a	b
R3	b	a	a	b	a	b	a	b	b

R1, R2, R3 is not a lossless join since neither row is completely filled with a's

(6) Provide a step-by-step lossless decomposition of R into BCNF normal form

F = {AD → B, BD → G, BE → I, AE → DI, AI → E, AEI → C}

R (A, B, C, D, E, G, H, I, J) ➔ split using AD → B since AD is not a super key it violates the BCNF

R1 (A, B, D)

R2 (A, C, D, E, G, H, I, J) ➔ split using AE → DI since AE is not a super key it violates the BCNF

R1 (A, B, D)

R21 (A, D, E, I) ➔ split using AI → E since AI is not a super key it violates the BCNF

R22 (A, C, E, G, H, J)

R1 (A, B, D)

R211 (A, E, I)

R212 (A, D, I)

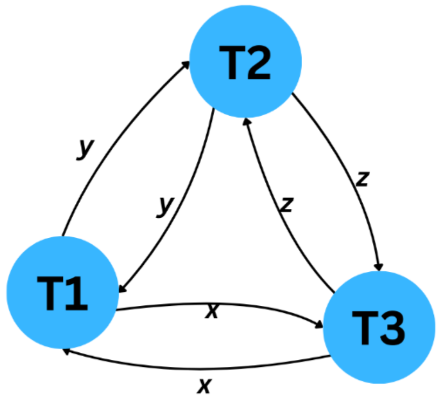
R22 (A, C, E, G, H, J)

This is the final lossless decomposition of R into BCNF.

Question 2:

(1)

No, since the precedence graph has a cycle, the schedule is not serializable. This means that it cannot be reordered into an equivalent serial schedule.



(2)

Time Slot	T1	T2	T3
t1	Lock-X(x)		
t2	R(x)		
t3	Lock-X(y)		
t4			Lock-X(z)
t5			R(z)
t6			Lock-X(x)
t7		Lock-X(y)	
t8		R(y)	
t9		Lock-X(z)	
t10	R(y)		
t11		R(z)	
t12			R(x)
t13			W(z)
t14	W(x)		
t15		W(y)	
t16			W(x)
t17			Unlock(z)
t18			Unlock(x)
t19	W(y)		
t20	Unlock(x)		
t21	Unlock(y)		
t22		W(z)	
t23		Unlock(y)	
t24		Unlock(z)	

(3)

The following steps are ignoring any computation,
T1 will get exclusive lock for x and y

T3 will get exclusive lock for z but not for x since T1 has it.

T1 releases lock on x and y

T3 will get exclusive lock for x

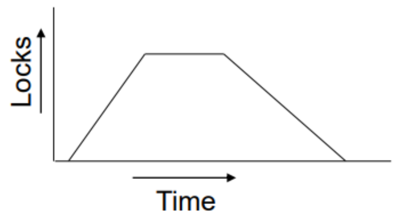
T2 will get exclusive lock on y but not for z since T3 has it.

T3 releases lock on z and x

T2 will get exclusive lock for z

T2 releases lock on y and z

Only exclusive locks have been used because both read and write operations are taking place on an item. If shared lock was used first, we would need to release/unlock the shared lock then take the exclusive lock but this would break the rule for 2PL where a transaction cannot obtain another lock when after releasing one.



Serialization has been achieved in the schedule; we can find the serial order of the transactions using their lock point (where the last lock was acquired).

T1 – t3

T2 – t9

T3 – t6

The serial: -

T1 → T3 → T2

No deadlock was found since it is serializable.

Question 3:

P1, P2, P1, P4, P3, P7, P2, P1, P4, P5, P8, P6, P8, P2, P8

(1) <u>LRU</u>

Initial Empty Buffer Pool → [empty, empty, empty]

P1: Cache Miss → [**<u>P1</u>**, empty, empty]

P2: Cache Miss → [P1, **<u>P2</u>**, empty]

P1: Cache Hit → [**<u>P1</u>**, P2, empty]

P4: Cache Miss → [P1, P2, **<u>P4]</u>**

P3: Cache Miss → [P1, **<u>P3</u>**, P4]

P7: Cache Miss → [**<u>P7</u>**, P3, P4]

P2: Cache Miss → [P7, P3, **<u>P2]</u>**

P1: Cache Miss → [P7, **<u>P1</u>**, P2]

P4: Cache Miss → [**<u>P4,</u>** P1, P2]

P5: Cache Miss → [P4, P1, **<u>P5</u>**]

P8: Cache Miss → [P4, **<u>P8</u>**, P5]

P6: Cache Miss → [**<u>P6</u>**, P8, P5]

P8: Cache Hit → [P6, **<u>P8</u>**, P5]

P2: Cache Miss → [P6, P8, **<u>P2</u>**]

P8: Cache Hit → [P6, **<u>P8,</u>** P2]

cache hits = 3

cache misses = 12

Hit rate = #cache hits / (#cache hits + #cache misses)

Hit rate = 3/ (3 + 12) = **<u>0.2</u>** or **<u>20%</u>**

(2) <u>MRU</u>

P1, P2, P1, P4, P3, P7, P2, P1, P4, P5, P8, P6, P8, P2, P8

Initial Empty Buffer Pool → [empty, empty, empty]

P1: Cache Miss → [**<u>P1</u>**, empty, empty]

P2: Cache Miss → [P1, **P2**, empty]

P1: Cache Hit → [**P1**, P2, empty]

P4: Cache Miss → [P1, P2, **P4]**

P3: Cache Miss → [P1, P2, **P3**]

P7: Cache Miss → [P1, P2, **P7**]

P2: Cache Hit → [P1, **P2**, P7**]**

P1: Cache Hit → [**P1**, P2, P7]

P4: Cache Miss → [**P4,** P2, P7]

P5: Cache Miss → [**P5**, P2, P7]

P8: Cache Miss → [**P8**, P2, P7]

P6: Cache Miss → [**P6**, P2, P7]

P8: Cache Miss → [**P8**, P2, P7]

P2: Cache Hit → [P8, **P2**, P7]

P8: Cache Hit → [**P8**, P2, P7]

cache hits = 5

cache misses = 10

Hit rate = #cache hits / (#cache hits + #cache misses)

Hit rate = 5/ (5 + 9) = **0.333** or **33.333%**

(3) FIFO

P1, P2, P1, P4, P3, P7, P2, P1, P4, P5, P8, P6, P8, P2, P8

Initial Empty Buffer Pool → [empty, empty, empty]

P1: Cache Miss → [**P1**, empty, empty]

P2: Cache Miss → [P1, **P2**, empty]

P1: Cache Hit → [**P1**, P2, empty]

P4: Cache Miss → [P1, P2, **P4]**

P3: Cache Miss → [**P3**, P2, P4]

P7: Cache Miss → [P3, **P7**, P4]

P2: Cache Miss → [P3, P7, **P2]**

P1: Cache Miss → [**<u>P1</u>**, P7, P2]

P4: Cache Miss → [P1, **<u>P4,</u>** P2]

P5: Cache Miss → [P1, P4, **<u>P5</u>**]

P8: Cache Miss → [**<u>P8</u>**, P4, P5]

P6: Cache Miss → [P8, **<u>P6</u>**, P5]

P8: Cache Hit → [**<u>P8</u>**, P6, P5]

P2: Cache Miss → [P8, P6, **<u>P2</u>**]

P8: Cache Hit → [**<u>P8</u>**, P6, P2]

cache hits = 3

cache misses = 12

Hit rate = #cache hits / (#cache hits + #cache misses)

Hit rate = 3/ (3 + 12) = **<u>0.2</u>** or **<u>20%</u>**