**Question 1 :**

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

1. **Check if 𝐴 → 𝐼 ∈ 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+

1. **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 𝐹𝑚 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 → I }

**Step2 :**

For CE → A :

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

So E → A can be inferred by F’;

Same reason for E → D and E → H can be inferred by F’;

for AH → I :

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

Thus, A → I can be inferred by F’;

F’’ = { A → D, A → E, A → I, B → G, B → I, E → C, E → D, E → A, E → H, H → G }

**Step3 :**

D **∈** A+|F’’ – {A → D}; thus A→D can be inferred by F’’ - {A→B} and we remove A→B;

A+|F’’’ – {A → E} = { A }, so A → 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 Fmin:

Fmin = F’’’ = { A → E, A → I, B → G, B → I, E → C, E → D, E → A, E → H, H → 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 );

Fmin = { A → E, A → I B → G, B → I, E → C, E → D, E → A, E → H, H → G }

Candidate key is : (B, E, J)

So we have a decomposition of R such that:

R1 = (A, E, I) R2 = (B, G, I) R3 = (E, A, C, D, H) R4 = (H, G)

R5 = (B, E, J)

Clearly, R has a set of FD of Fmin , 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 |
| R1 = (A, E, I) | a | b | b | b | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | b | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | b | b | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | b | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | b | b | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | b | b | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | b | b | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | a | b | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | a | a | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (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 |
| R1 = (A, E, I) | a | b | a | a | a | b | b | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (B, E, J) | a | a | a | a | a | a | b | a | a |

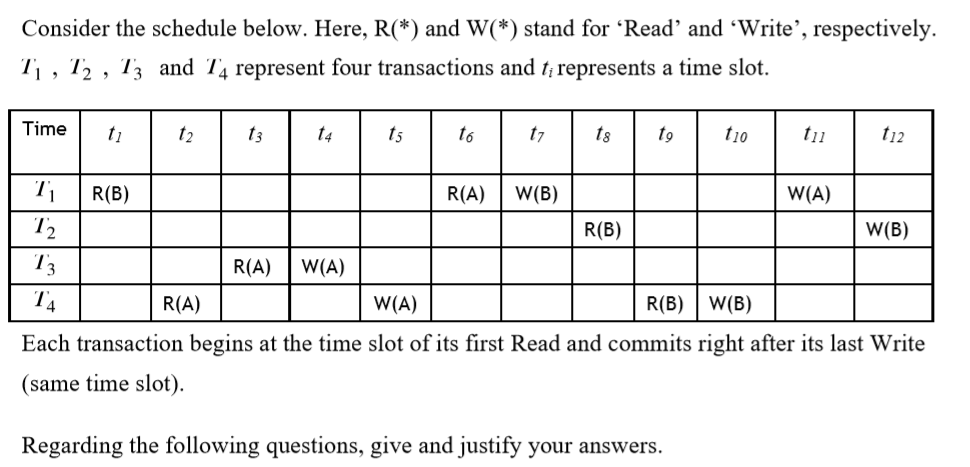
Test E 🡪 H:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| decomposition | A | B | C | D | E | G | H | I | J |
| R1 = (A, E, I) | a | b | a | a | a | b | a | a | b |
| R2 = (B, G, I) | b | a | b | b | b | a | b | a | b |
| R3 = (E, A, C, D, H) | a | b | a | a | a | b | a | a | b |
| R4 = (H, G) | b | b | b | b | b | a | a | b | b |
| R5 = (B, E, J) | a | a | a | a | a | a | a | a | a |

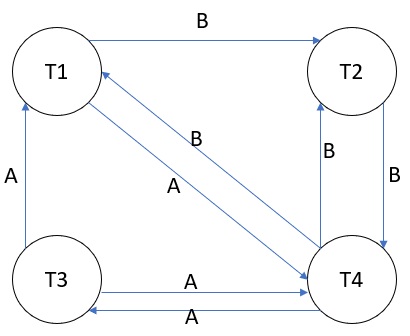
By testing E🡪H, we can get a row is made up entirely of “a” symbols.

So this decomposition is lossless and dependency-preserving.

**Question 2:**



**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 | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
| T1 | R(B) | R(A) | W(B) | W(A) |  |  |  |  |  |  |  |  |
| T2 |  |  |  |  | R(B) | W(B) |  |  |  |  |  |  |
| T3 |  |  |  |  |  |  | R(A) | W(A) |  |  |  |  |
| T4 |  |  |  |  |  |  |  |  | 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 T1 and T2 according to the simple locking scheme :

|  |  |
| --- | --- |
| T1 | T2 |
| 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) |