**Q1)**

1. Checking if DGH -> J belongs to F+.
   1. Calculating (DGH)+
   2. (DGH)+ = {D,G,H}
   3. (DGH)+ = {D,G,H,A} because H -> A
   4. (DGH)+ = {D,G,H,A, E,I} because DG -> EI
   5. (DGH)+ = {D,G,H,A,E,I,J} because AI -> EHJ
   6. Thus DGH -> J belongs to F+ because every relation in (DGH)+ belongs to F+.
2. Finding all the candidate keys of R.
   1. Relation R(A,B,C,D,E,G,H,I,J)
   2. Finding closure for single attributes.
   3. (A)+ = {A}, (B)+ ={B}, (C)+ = {C}, (D)+ = {D}, (E)+ = {E}, (G)+ = {G}, (H)+ = {H,A}, (I)+ = {I}, (J)+ = {J}. There are no candidate keys with one attribute.
   4. (AB)+ ={A,B,C,D,E,G,H,I,J} because AB -> DGH, DG -> EI, AI -> EHJ, DEJ -> CI. AB is a candidate key.
   5. The candidate key must have B in the relation because B is always a determinant in F. It must as small as possible because of AB(candidate key is a minimal super key). The other key is HB because A is dependent on H.
   6. (HB)+ = {A,B,C,D,E,G,H,I,J} because H->A, AB -> DGH, DG -> EI, AI -> EHJ, DEJ -> CI.
   7. Thus K1= AB and K2= HB.
3. Highest normal form of R with respect to F.
   1. 1NF: It is in 1NF because attribute values are atomic and is part of the definition of the relational model. There are no multivalued attributes, composite attributes, and their combinations.
   2. 2NF: AB -> DGH is a relation in 2NF because there no partial dependencies, AB is a full proper candidate key. DG -> EI has no partial dependency because D and G are non-prime attributes. H -> A has no partial dependency because H is a non-prime attribute. DEJ -> CI has no partial dependency because D, E and J are non-prime attributes. AI -> EHJ has a partial dependency because AI -> EHJ can be written as AI -> E and AI -> J. A is a prime attribute and part of the candidate key and J and E are non-prime attributes. Thus, relation is not in 2NF and R’s highest normal form is 1NF. There is no need for checking 3NF and BCNF because 3NF relations needs to be 2NF relations as well and BCNF relations needs to be 3NF relations.
4. Is the decomposition R1 = {ADGH}, R2 = {AEIJ}, R3 = {BCDGJ} of 𝑅 satisfy the lossless join property?
   1. It is not.

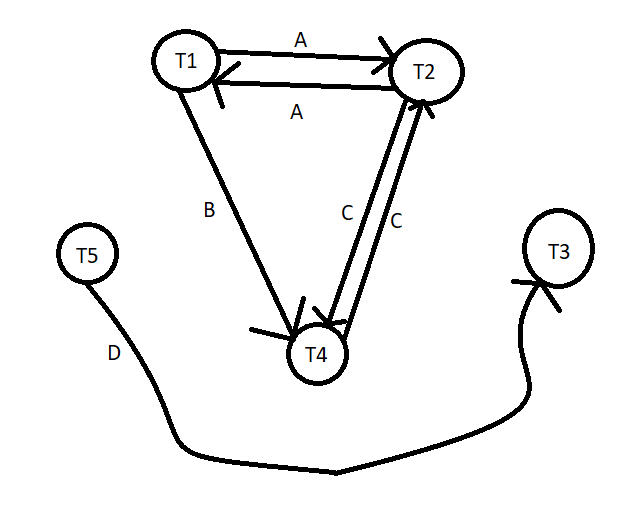
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decomposition | A | B | C | D | E | G | H | I | J |
| R1(ADGH) | **a** | b | b | a | b | a | a | b | b |
| R2(AEIJ) | **a** | b | b | b | a | b | b | a | a |
| Decomposition | A | B | C | D | E | G | H | I | J |
| R1(ADGH) | a | b | b | **a** | b->**a** | **a** | a | b->**a** | b |
| R2(BCDGJ) | b | a | a | **a** | b->**a** | **a** | b | b->**a** | a |
| Decomposition | A | B | C | D | E | G | H | I | J |
| R1(AEIJ) | a | b | b | b | a | b | b | a | **a** |
| R2(BCDGJ) | b | a | a | a | b | a | b | b | **a** |

* 1. For R1 and R2, intersection is A. ‘A’ alone does not determine anything as (A)+ = {A} For R1 and R3, intersection is DG. DG determines EI, making (DGEI). (DGEI)+ = (D,G,E,I), which is not cover every attribute. For R2 and R3, intersection is J. ‘J’ alone does not determine anything as (J)+ = {J}. Thus it does not satisfy the lossless join property.

1. Step-by-step lossless decomposition of R into BCNF normal form.
   1. For 𝐹 = {AB -> DGH, DG->EI, H->A, DEJ->CI, AI->EHJ}
   2. For DEJ -> C, DEJ is not a superkey. Split R into **R1(D,E,J,C)** and R2(A,B,D,E,G,H,I,J)
   3. For AI -> EHJ, AI is not a superkey. Split R2 into **R3(A,I,E,H,J)** and R4(A,B,D,G,I)
   4. For H->A, H is not a superkey. Split R3 into **R31(H,A)** and R32(I,E,H,J)
   5. For DG -> EI, DG is not a superkey. Split R4 into **R5(D,G,E,I)** and **R6(A,B,D,G)**.
   6. One of the possible lossless join decompositions to BCNF is: R1, R3, R31, R5, R6.

**Q2)**

1. Assume a checkpoint is made between t10 and t11, what should be done to the five transactions when the crash happens between t15 and t16.
   1. T1 -> Undo because T1 is active during the crash.
   2. T2 -> Redo because T2 is active during the checkpoint before the crash.
   3. T3 -> No action needed because T3 commits before the checkpoint.
   4. T4 -> Undo because T4 is active during the crash
   5. T5 -> Undo because T5 is active during the crash.
2. Is the transaction schedule conflict serializable? Give the full precedence graph to justify your answer.
   1. The transaction schedule is not conflict serializable. Because of cycles between T1 and T2, T2 and T4, and T1, T4, T2 and T1.



1. The schedule below will result in a deadlock. There will always be a schedule that can cause a deadlock as long as it has more than one transaction. There will be a deadlock on t6, t7, t8, t9, t10 which has Write\_lock(B), Write\_lock(C), Write\_Lock(D), Write\_Lock(E), Write\_Lock(A).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | T1 | T2 | T3 | T4 | T5 |
| t1 | Read\_lock(A) |  |  |  |  |
| t2 |  | Read\_lock(B) |  |  |  |
| t3 |  |  | Read\_lock(C) |  |  |
| t4 |  |  |  | Read\_Lock(D) |  |
| t5 |  |  |  |  | Read\_Lock(E) |
| t6 | Write\_lock(B) |  |  |  |  |
| t7 |  | Write\_Lock(C) |  |  |  |
| t8 |  |  | Write\_Lock(D) |  |  |
| t9 |  |  |  | Write\_Lock(E) |  |
| t10 |  |  |  |  | Write\_Lock(A) |

**Q3)** P1, P2, P3, P4, P5, P2, P6, P4, P7, P3, P8, P5

1. Least Recently Used (LRU) policy.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query-> | P1 | P2 | P3 | P4 | P5 | P2 | P6 | P4 | P7 | P3 | P8 | P5 |
| Buffer1 | P1 | P1 | P1 | P1 | P5 | P5 | P5 | P5 | P7 | P7 | P7 | P7 |
| Buffer2 | N | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P3 | P3 | P3 |
| Buffer3 | N | N | P3 | P3 | P3 | P3 | P6 | P6 | P6 | P6 | P8 | P8 |
| Buffer4 | N | N | N | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P5 |

1. Most Recently Used (MRU) policy.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query-> | P1 | P2 | P3 | P4 | P5 | P2 | P6 | P4 | P7 | P3 | P8 | P5 |
| Buffer1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 |
| Buffer 2 | N | P2 | P2 | P2 | P2 | P2 | P6 | P4 | P7 | P7 | P7 | P7 |
| Buffer 3 | N | N | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P8 | P8 |
| Buffer 4 | N | N | N | P4 | P5 | P5 | P5 | P5 | P5 | P5 | P5 | P5 |

1. MRU is better than LRU for this query because MRU has lesser page faults than LRU. MRU has 9 page faults and LRU has 10.