ASSIGNMENT 2

# 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 + .**

To check if 𝐴 → 𝐼 ∈ F+ we need to find A+

So,

A+ = ADECHGI

Therefore 𝐴 → 𝐼 ∈ 𝐹+

1. **Find a candidate key for 𝑅.**

B+ = BGI ∉ 𝐹

A+ = ADECHGI ∉ 𝐹

J+= J ∉ 𝐹

(ABJ)+ = ABJDEGICH ∈ 𝐹

(BEJ)+ = BEJGICDAH ∈ 𝐹

(BC)+ = BCGI ∉ 𝐹

Hence we have two candidate keys in the given relation R as follows

1. ABJ
2. BEJ
3. **Determine the highest normal form of 𝑅 with respect to 𝐹. Justify your answer.**

Candidate keys are ABJ, BEJ

Hence prime attributes are A, B, E, J and non-prime attributes are C, D, G, H, I

Checking for 2NF

𝐹 = { 𝐴 → 𝐷𝐸, 𝐵 → 𝐺𝐼, 𝐸 → 𝐶𝐷 } shows that non-prime attributes are not fully functionally dependent on the candidate keys.

Hence the given relation is not in 2NF and thus cannot be in any higher normal form than Second Normal Form (2NF).

Therefore the highest normal form of the relation R is 1NF.

1. **Find a minimal cover 𝐹𝑚 for 𝐹.**

𝐹 = {𝐴 → 𝐷𝐸, 𝐵 → 𝐺𝐼, 𝐸 → 𝐶𝐷, 𝐶𝐸 → 𝐴𝐷𝐻, 𝐻 → 𝐺, 𝐴𝐻 → 𝐼}.

Step 1: Split the FDs such that R.H.S contains a single attribute

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

AH → I}

Step 2: Find the redundant FDs and delete them from the set

A → D and E → D are redundant because we have A+ = AECDHGI and E+ = ECADHGI

Hence after deleting them we get

𝐹𝑚 = { A → E, B → G, B → I, E → C, CE → A, CE → D, CE → H, H → G, AH → I }

Step 3: Find the redundant attributes on L.H.S and delete them from the set

In CE → A, CE → D, CE → H. C is redundant as we have E+ = EC

In AH → I. H is redundant as we have A+ = ADECHGI

Hence after deleting them we get

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

Now we can combine them into more simpler FDs

Therefore

𝐹𝑚 = { A → EI, B → GI, E → ACDH, H → G }

1. **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.**

Using the minimal cover we have 𝐹𝑚 = { A → EI, B → GI, E → ACDH, H → G }

Decomposing 𝐹𝑚 into relations we get

R1 (A, E, I)

R2 (B, G, I)

R3 (A, E, C, D, H)

R4 (H, G)

Also we can easily combine R1 and R3 as A and E are common prime attributes,

we get our new relations as :-

R1 (B, G, I)

R2 (A, E, C, D, H, I)

R3 (H, G)

Dependency-preserving check

Here, the given relation 𝑅(𝐴, 𝐵, 𝐶, 𝐷, 𝐸, 𝐺, 𝐻, 𝐼, 𝐽) with FD set 𝐹 = {𝐴 → 𝐷𝐸, 𝐵 → 𝐺𝐼, 𝐸 → 𝐶𝐷, 𝐶𝐸 → 𝐴𝐷𝐻, 𝐻 → 𝐺, 𝐴𝐻 → 𝐼} is decomposed into R1 (B,G,I) and R2 (A,E,C,D,H,I) and R3 (H,G) which is dependency preserving because FD

𝐹 = {𝐴 → 𝐷𝐸 } is a part of R2 (A, E, C, D, H, I).

𝐹 = { 𝐵 → 𝐺𝐼 } is a part of R1 (B, G, I).

𝐹 = { 𝐸 → 𝐶𝐷 } is a part of R2 (A, E, C, D, H, I).

𝐹 = { 𝐶𝐸 → 𝐴𝐷𝐻 } is a part of R2 (A, E, C, D, H, I).

𝐹 = { 𝐻 → 𝐺 } is a part of R3 (H, G)

𝐹 = { 𝐴𝐻 → 𝐼 } is a part of R2 (A, E, C, D, H, I)

Hence the decomposition is dependency-preserving.

Lossless-join check

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

By the above table we get that the decomposition in lossy and hence to make it lossless we need to add another relation R4 (A, B, J) because ABJ is our candidate key.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | G | H | I | J |
| R1 | b | a | b | b | b | a | b | a | b |
| R2 | a | b | a | a | a | ~~b~~ a | a | a | b |
| R3 | b | b | b | b | b | a | a | b | b |
| R4 | a | a | ~~b~~ a | ~~b~~ a | ~~b~~ a | ~~b~~ a | ~~b~~ a | ~~b~~ a | ~~b~~ a |

As we can see the last row is complete ‘a’ hence now the decomposition is lossless.

Further lossless-join checks include :-

1. Att(R1) ∪ Att(R2) ∪ Att(R3) ∪ Att(R4) = (A, B, C, D, E, G, H, I, J) = Att(R)
2. Att(R1) ∩ Att(R2) ∩ Att(R3) ∩ Att(R4) ≠ ∅

Therefore a dependency-preserving and lossless-join 3NF decomposition is

R1 (B, G, I)

R2 (A, E, C, D, H, I)

R3 (H, G)

R4 (A, B, J)

# QUESTION 2

**Consider the schedule below. Here, R(\*) and W(\*) stand for ‘Read’ and ‘Write’, respectively. T1, T2 , T3 and T4 represent four transactions and ti represents a time slot.**

**A close up of a screen

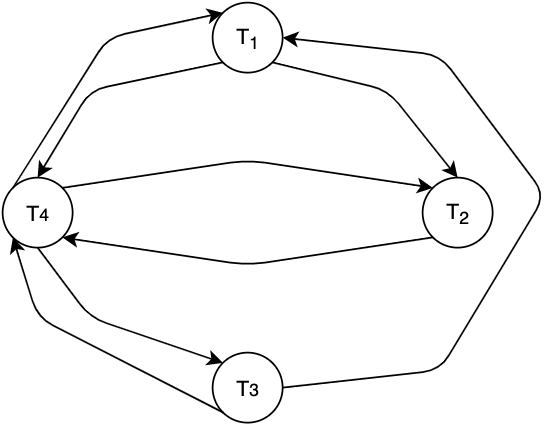
Description automatically generated**

**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 serializable? Give the precedence graph to justify your answer.**

Precedence graph



The precedence graph is cyclic in nature therefore the transaction is NOT conflict serializable.

1. **Give a serial schedule of these four transactions.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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) |

1. **Lock the transactions T1 and T2 according to the simple locking scheme. You only need to consider the order of the operations, not the timestamps.**

|  |  |
| --- | --- |
| T1 | T2 |
| write\_lock(B) |  |
| R(B) |  |
| write\_lock(A) |  |
| R(A) |  |
| W(B) |  |
| unlock(B) |  |
|  | write\_lock(B) |
|  | R(B) |
| W(A) |  |
| unlock(A) |  |
|  | W(B) |
|  | unlock(B) |