1) Join Ordering:

1) EMP → Site 2 [cost = 300]

Site 2 computes ASG' = EMP ⋈ ASG

ASG' → site 3 [cost = 500]

Site 3 computes ASG' ⋈ PROJ

[Total cost = 800]

2) ASG → site 1 [cost = 400]

Site 1 computes ASG' = EMP ⋈ ASG

ASG' → Site 3 [cost = 500]

Site 3 computes ASG' ⋈ PROJ

[Total cost = 900]

3) ASG → site 3 [cost = 400]

Site 3 computes ASG' = ASG ⋈ PROJ

ASG' → site 1 [cost = 400]

Site 1 computes ASG' ⋈ EMP

[Total cost = 800]

4) PROJ → site 2 [cost = 500]

Site 2 computes ASG' = PROJ ⋈ ASG

ASG' → Site 1 [cost = 400]

Site 1 computes ASG' ⋈ EMP

[Total cost = 900]

5) EMP → site 2 [cost = 300]

PROJ → site 2 [cost = 500]

Site 2 computes EMP ⋈ PROJ ⋈ ASG

[Total cost = 800]

6) EMP → site 3 [cost = 300]

ASG → Site 3 [cost = 400]

Site 3 computes EMP ⋈ PROJ ⋈ ASG

[Total cost = 700]

Since method 6 has the smallest communication time (700), it becomes the optimal join program with minimum response time and the result is stored in site 3.

2) <u>Part 1</u>: Consider Site-1 and Site-2

As we know $EMP \bowtie_{ENO} ASG = \underbrace{(EMP \ltimes_{ENO} ASG)}_{Emp'} \bowtie_{ENO} ASG$

1) $ASG' = \pi_{ENO}(ASG)$

2) $ASG' \longrightarrow$ Site 1

3) Site 1 compute $EMP' = EMP \ltimes_{ENO} ASG'$

4) $EMP' \rightarrow$ site 2

5) Site 2 computes $ASG'' = EMP' \bowtie_{ENO} ASG$

<u>Part 2</u>: Consider Site-2 and Site-3

As we know $PROJ \bowtie_{PNO} ASG'' = (PROJ \ltimes_{PNO} ASG'') \bowtie_{PNO} ASG''$

1) $ASG''' = \pi_{PNO}(PROJ)$

2) $ASG''' \longrightarrow$ site 3

3) Site 3 computes $PROJ' = PROJ \ltimes_{PNO} ASG'''$

4) $PROJ' \longrightarrow$ site 2

5) Site 2 computes $PROJ' \bowtie_{PNO} ASG''$

3) 1) First we perform ASG ⋈ PROJ because we get a smaller intermediate result. The join result size of ASG ⋈ PROJ is 2000 which is smaller than the size of EMP ⋈ ASG which is 3000.

2) Since the complete PROJ table is in site 3 we send it to site 2.

size of PROJ

1) PROJ (site 3) ⟶ site 2 (cost = 2000)

2) site 2 computes ASG' = ASG ⋈ PROJ

3) The intermediate result which is ASG ⋈ PROJ is stored at site 2 whose size is 2000.

Because the EMP is fragmented at different sites we need to bring it to site 2 for the second join.

EMP Fragment at site 1

1) EMP (site 1) ⟶ site 2 (cost = 2000)

EMP Fragment at site 3

2) EMP (site 3) ⟶ site 2 (cost = 2000)

3) EMP is unioned at site 2

EMP = EMP(site 1) ∪ EMP (site 2) ∪ EMP(site 3)

4) site 2 computes EMP ⋈ ASG' = EMP ⋈ ASG ⋈ PROJ

Hence, the result is stored at site 2.

The below are all steps together.

Step 1: PROJ(site 3) $\rightarrow$ site 2 (cost = 2000)

Step 2: site 2 computes $ASG' = ASG \bowtie PROJ$

Step 3: EMP(site 1) $\rightarrow$ site 2 (cost = 2000)

Step 4: EMP(site 3) $\rightarrow$ site 2 (cost = 2000)

Step 5: $EMP = EMP(site 1) \cup EMP(site 2) \cup EMP(site 3)$

Step 6: site 2 computes $EMP \bowtie ASG' \Rightarrow EMP \bowtie ASG \bowtie PROJ$

(Total cost = 6000)

4) 1) Conflicting Operations Pairs:

H1: W2(x) W1(x)
W2(x) R3(x)
W2(x) R1(x)
W1(x) R3(x)
W1(x) R2(x)
W2(y) R3(y)

H2: R3(y) W2(y)
W1(x) R3(x)
W1(x) W2(x)
R3(x) W2(x)
W2(x) R1(x)

H3: W2(x) R1(x)
W2(x) R3(x)
W2(x) W1(x)
W2(y) R3(y)
R3(x) W1(x)

H4: W2(x) W1(x)
W2(x) R1(x)
W2(x) R3(x)
W2(y) R3(y)
W1(x) R3(x)

2)

conflict

H1: W2(x) W1(x) R3(x) R1(x) W2(y) R3(y) R2(x) R3(z)

H2: R3(y) R3(z) W2(y) R2(z) W1(x) R3(x) W2(x) R1(x)

H3: W2(x) R3(z) W2(y) R1(x) R3(x) R2(z) R3(y) W1(x)

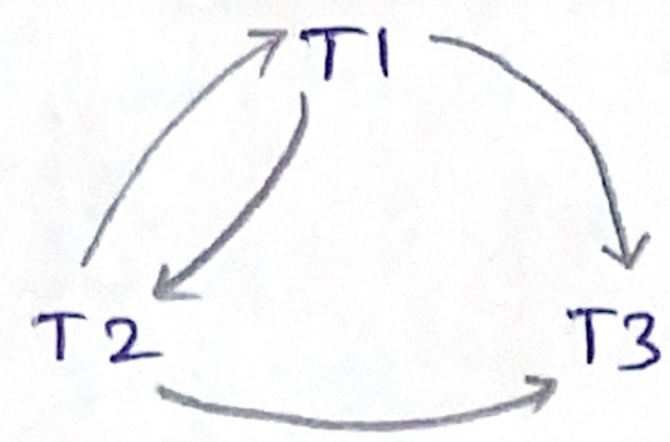
H4: R2(z) W2(x) W2(y) W1(x) R1(x) R3(x) R3(y) R3(z)

W2(x) and R1(x) is a conflict Equivalence because it has the same set of conflicting operations in same order.

W2(x), R1(x) appear in same order in all histories.

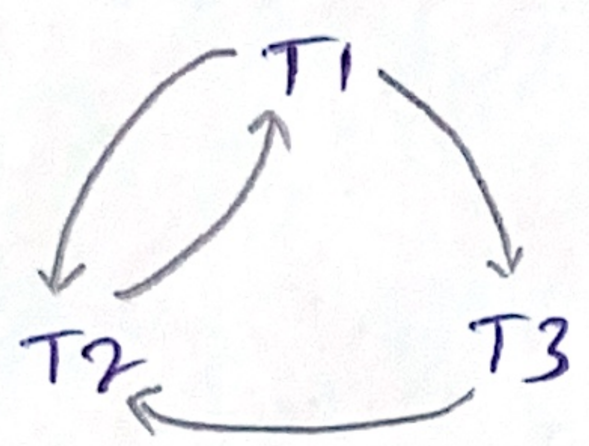
5) Serialization graphs:

✗
H1:



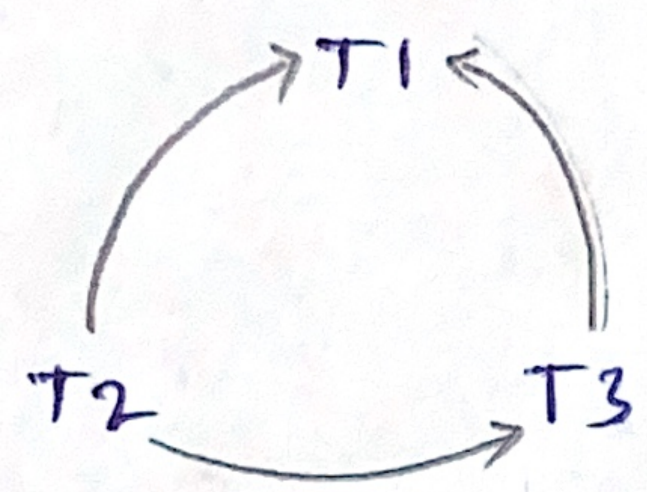
Because the serialization graph contains a cycle/Loop H1 is not serializable.

✗
H2:



Because the serialization graph contains cycle H2 is not serializable

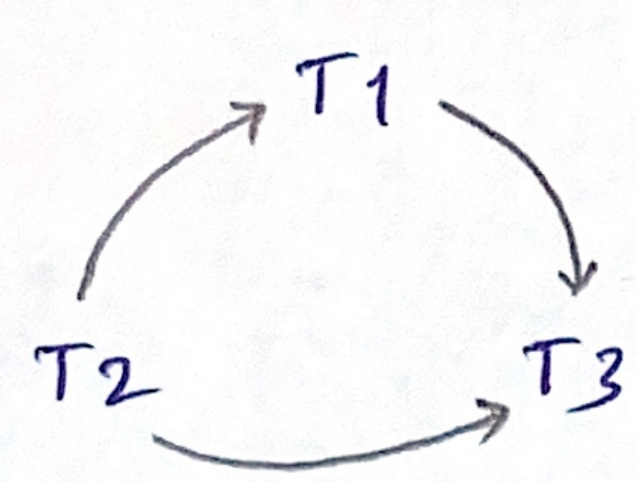
✓
H3:



Because the serialization graph do not contain any cycle, H3 is serializable. The order of execution is

T2 → T3 → T1 using in-degree's

✓
H4:



Because the serization graph do not contain any cycle, H4 is serializable. The order of execution is

T2 → T1 → T3.

H1 & H2 are not serializable because it contains cycle.

H3 & H4 are serializable because it do not contains cycle.