

Proof of work

Now I am testing the code on some exercises from the exam, checking whether the solutions proposed by my program are equal to the ones provided in the exam.

1) $r_1(x)w_3(x) r_2(x) r_1(y)w_1(x) r_2(y)w_2(y)w_3(y)$

This exercise is from 12/07/2019, and the solution says:

Since $w_3(y)$ is the final write on y in S , and $w_2(y)$ is in S , we observe that T_3 must appear after T_2 in every serial schedule S' on T_1, T_2, T_3 that is view-equivalent to S , because otherwise S' would have a different “final write set” with respect to S . On the other hand, the sequence $w_3(x) r_2(x)$ in S implies that in any serial schedule S' on T_1, T_2, T_3 that is view-equivalent to S transaction T_2 appears immediately after transaction T_3 , because otherwise S' would have a different “read from” relation with respect to S . We conclude that no serial schedule on T_1, T_2, T_3 exists that is view-equivalent to S , i.e., S is not view-serializable.

VS Playground solution:

Check Schedule

$r_1(x)w_3(x) r_2(x) r_1(y)w_1(x) r_2(y)w_2(y)w_3(y)$

[Guidelines](#)[Test](#)

Result

The schedule is not **View Serializable**

Did you want to see the full solution?
[click here](#)

Schedule: $r_1(x) w_3(x) r_2(x) r_1(y) w_1(x) r_2(y) w_2(y) w_3(y)$
Final write: $w_1(x) - w_3(y)$
Read from: $T_2 \rightarrow T_3$
View Eq Transaction: No view equivalent transaction
View Eq Schedule:
No serial schedule

2) $w_4(x) r_3(z) r_2(x)w_3(y) r_2(z)w_2(y) r_4(z)w_1(y)$

01/02/2016, solution:

- The serial schedule T_4, T_2, T_3, T_1 has the same read-from relation and the same final-write-set as S , whereas it has the conflicting action pair $(w_2(y), w_3(y))$ that appears in different order with respect to S . Therefore, such serial schedule is view equivalent but not conflict equivalent to S .

The exercise requires the user to identify a view-equivalent schedule that is not conflict-serializable. I need to verify whether the serial schedule proposed in the solution matches the one I provided.

VS Playground solution:

Check Schedule

w4(x) r3(z) r2(x)w3(y) r2(z)w2(y) r4(z)w1(y)

[Guidelines](#)

Test

Result

The schedule is **View Serializable**

Did you want to see the full solution?
[click here](#)

Schedule: w4(x) r3(z) r2(x) w3(y) r2(z) w2(y) r4(z) w1(y)
Final write: w4(x) - w1(y)
Read from: T2 → T4
View Eq Transaction: T4 T3 T2 T1 - T3 T4 T2 T1 - T4 T2 T3 T1
View Eq Schedule:
w4(x) r4(z) r3(z) w3(y) r2(x) r2(z) w2(y) w1(y)
r3(z) w3(y) w4(x) r4(z) r2(x) r2(z) w2(y) w1(y)
w4(x) r4(z) r2(x) r2(z) w2(y) r3(z) w3(y) w1(y)

3) r4(x)w3(y) r2(x)w1(z)w4(x)w4(z) r2(y)w1(y)

21/2/2014, solution:

S is view-serializable, since the serial schedule T_3, T_2, T_1, T_4 is clearly view-equivalent to S , because it has the same read-from relation and the same final-set as S . There is a single action that can be added to S in such a way that the resulting schedule is no longer view serializable: for example, if we add $r_1(x)$ at the end of the schedule, then the resulting schedule S' is no longer view serializable, because now T_1 reads from T_4 (implying that in every serial schedule view equivalent to S' , T_4 precedes T_1), and in S the final write on z is $w_4(z)$ (implying that in every serial schedule view equivalent to S' , T_1 precedes T_4).

VS Playground solution:

Check Schedule

r4(x)w3(y) r2(x)w1(z)w4(x)w4(z) r2(y)w1(y)

[Guidelines](#)

Test

Result

The schedule is **View Serializable**

Did you want to see the full solution?
[click here](#)

Schedule: r4(x) w3(y) r2(x) w1(z) w4(x) w4(z) r2(y) w1(y)
Final write: w4(x) - w1(y) - w4(z)
Read from: T2 → T3
View Eq Transaction: T3 T2 T1 T4
View Eq Schedule:
w3(y) r2(x) r2(y) w1(z) w1(y) r4(x) w4(x) w4(z)

4) r1(A) r3(C) w3(B) r2(A) w1(B) w1(A) w2(A) r1(C) w3(C) r3(A) r2(D)

18/02/2010, solution:

S is not view-serializable. Indeed, every serial schedule S' on the set of transactions constituting S will have either transaction t_1 before t_2 , or t_2 before t_1 . In the former case (t_1 before t_2), the pair $\langle w_1(A), r_2(A) \rangle$ is in the READS-FROM relation associated to S' , while it is not in the READS-FROM relation associated to S . In the latter case (t_2 before t_1), the pair $\langle w_2(A), r_1(A) \rangle$ is in the READS-FROM relation associated to S' , while it is not in the READS-FROM relation associated to S . Actually, it is easy to see that S is not even conflict-serializable, because the precedence graph associated to S is cyclic.

VS Playground solution:

Check Schedule

r1(A) r3(C) w3(B) r2(A) w1(B) w1(A) w2(A) r1(C) w3(C) r3(A) r2(D)

[Guidelines](#)

Test

Result

The schedule is not **View Serializable**

Did you want to see the full solution?
[click here](#)

Schedule: r1(a) r3(c) w3(b) r2(a) w1(b) w1(a) w2(a) r1(c) w3(c) r3(a) r2(d)
Final write: w2(a) - w3(c) - w1(b)
Read from: T3 → T2
View Eq Transaction: No view equivalent transaction
View Eq Schedule:
No serial schedule