

(a) Consider these three transactions:

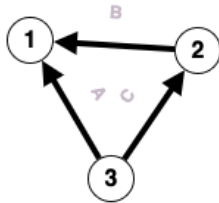
- $T_1 : R_1(A), R_1(B), W_1(A), W_1(B), Co_1$
- $T_2 : R_2(B), W_2(B), R_2(C), W_2(C), Co_2$
- $T_3 : R_3(C), W_3(C), R_3(A), W_3(A), Co_3$

i. Schedule 1:

$R_2(B), W_2(B), R_3(C), W_3(C), R_3(A), W_3(A), Co_3, R_2(C), W_2(C), Co_2, R_1(A), R_1(B), W_1(A), W_1(B), Co_1$

Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: yes: 3,2,1. No cycles in the graph.



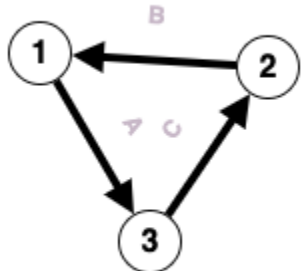
1 <--B-- 2 <--C-- 3
 ^ |
 +-----A-----+

ii. Schedule 2:

$R_2(B), W_2(B), R_3(C), W_3(C), R_1(A), R_1(B), W_1(A), W_1(B), Co_1, R_2(C), W_2(C), Co_2, R_3(A), W_3(A), Co_3$

Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: no, Graph has cycles



1 <--B-- 2 <--C-- 3
 | ^
 +-----A-----+

(b) Consider the following three transactions:

- $T_1 : R_1(A), W_1(B), Co_1$
- $T_2 : R_2(B), W_2(C), Co_2$
- $T_3 : R_3(C), W_3(D), Co_3$

Give an example of a conflict-serializable schedule that has the following properties: transaction T_1 commits before transaction T_3 starts, and the equivalent serial order is T_3, T_2, T_1 .

Solution:

- **$R_1(A), R_2(B), W_1(B), Co_1, R_3(C), W_2(C), Co_2, W_3(D), Co_3$**

Variations include:

- **Swap the first two reads (of A and B)**
- **Swap last two writes (of C and D, together with the commit order)**