

Concurrency Control

Serializability and Precedence Graphs

Consider the following schedule:

$S = w_3(E) r_1(D) w_2(C) w_3(A) r_1(E) w_1(B) r_1(B) w_2(E) r_4(A) w_4(C)$

1. Draw the precedence graph.
2. Is this schedule conflict serializable? Why? Why not? If yes, provide the equivalent serial schedule.

Two Phase Locking

Consider the following two transactions:

$T1 = w_1(C) r_1(A) w_1(A) r_1(B) w_1(B)$

$T2 = r_2(B) w_2(B) r_2(A) w_2(A)$

Assume that the scheduler uses exclusive locks only. For each of the following instances involving T1 and T2, annotated with lock and unlock actions, complete the following table

Instance A

$T1 = L_1(C) w_1(C) L_1(A) r_1(A) w_1(A) U_1(A) L_1(B) r_1(B) w_1(B) U_1(C) U_1(B)$

$T2 = L_2(B) r_2(B) w_2(B) U_2(B) L_2(A) r_2(A) w_2(A) U_2(A)$

Instance B

$T1 = L_1(C) w_1(C) L_1(A) r_1(A) w_1(A) L_1(B) r_1(B) w_1(B) COMMIT U_1(A) U_1(C) U_1(B)$

$T2 = L_2(B) r_2(B) w_2(B) L_2(A) r_2(A) w_2(A) COMMIT U_2(A) U_2(B)$

Instance C

$T1 = L_1(C) L_1(A) w_1(C) r_1(A) w_1(A) L_1(B) r_1(B) w_1(B) COMMIT U_1(A) U_1(C) U_1(B)$

$T2 = L_2(B) r_2(B) w_2(B) L_2(A) r_2(A) w_2(A) COMMIT U_2(A) U_2(B)$

Instance D

$T1 = L_1(B) L_1(C) w_1(C) L_1(A) r_1(A) w_1(A) r_1(B) w_1(B) COMMIT U_1(A) U_1(C) U_1(B)$

$T2 = L_2(B) r_2(B) w_2(B) L_2(A) r_2(A) w_2(A) COMMIT U_2(A) U_2(B)$

Question	A	B	C	D
Is 2PL				
Is strict 2PL				
Will always result in conflict serializable schedule ¹				
Will always result in schedule that avoids cascading rollback				
Will result in a strict schedule				
Will result in a serial schedule				
May result in a deadlock				

¹ If no deadlock happens