Tutorial 10: Concurrency Control

Exercise 1:

Of the following 2 schedules, indicate if they are valid according to 2PL, the type of 2PL (basic, conservative, strict, rigorous), and if they successfully complete or may result in a deadlock state. Write the actual executed schedule.

i) sl1(A);r1(A);ul1(A);xl2(B);r2(B);w2(B);xl1(B);c2;ul2(B);w1(B);c1;ul1(B);
ii) xl1(B);w1(B);sl1(A);r1(A);xl2(A);w2(A);xl2(B);c1;ul1(A);ul1(B);c2;ul2(A);ul2(B);

Answer:

i) The schedule is not valid as T1 unlocks A before it acquires the lock on B.
ii) The schedule is valid and implements rigorous 2PL as locks held until after commit. It will successfully complete with the equivalent serial order of T1, T2, as T2 waits for locks from T1.

Exercise 2:

For the following 2 schedules, insert the appropriate locks (shared and exclusive) into the schedule to follow the strict 2PL protocol. Also explain what happens as the scheduler executes each schedule. Write the actual executed schedule.

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

Answer:

i) xl1(C);r1(C);xl2(B);r2(B);xl3(A);r3(A);sl1(B);sl2(A);sl3(C);a3;ul3(A);r2(A);w2(B);c2;ul2(A);ul2(B);
r1(B);w1(C);c1;ul1(C);ul1(B);xl3(A);r3(A);sl3(C);r3(C);w3(A);c3;ul3(A);ul3(C);

The scheduler grants exclusive locks on C,B,A to T1,T2,T3 respectively. When T1 requests sl1(B) it must wait as B its exclusively locked by T2. Similarly, T2 must wait for sl2(A) as A is exclusively locked by T3, and T3 must wait for sl3(C) as C is exclusively locked by T1. The wait-for graph thus has a cycle and one of the transactions must be aborted. We arbitrarily pick T3 to abort. This allows r2(A) to occur as T2 can acquire the shared lock on C and continue. T2 can then commit and unlock A and B. T1 can then acquire its shared lock on B, continue, commit, and release its locks. Finally, we show T3 re-executing serially after being re-submitted to the system.
The scheduler grants all locks until xl1(B) is requested which blocks as a shared lock on B is held by T2. Then, xl2(A) blocks as a shared lock is held on A by T3. The xl3(D) is granted, and T3 is able to commit and release its locks. Then, T2 acquires its exclusive lock on A, commits, and releases its locks. Finally, T1 acquires its exclusive lock on B, commits, and releases its locks. The equivalent serial schedule is T3, T2, T1. Note that by strict 2PL T3 can release sl3(A) before commit and immediately after it is granted xl3(D). The same holds for sl1(C) and sl2(B).

**Exercise 3:**

Consider the following two transactions:

T1:

- Read (A)
- Read (B)
- If $A = 0$ then $B := B + 1$
- Write (B)

T2:

- Read (B)
- Read (A)
- If $B = 0$ then $A := A + 1$
- Write (A)

Add lock and unlock instructions to transactions T1 and T2, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock?

**Answer:**

a. Lock and unlock instructions:

**T1:**

- lock-S(A)
- read(A)
- lock-X(B)
- read(B)
- if $A = 0$
- then $B := B + 1$
- write(B)
- unlock(A)
- unlock(B)
\textbf{T}_2: \quad \text{lock-}S(B) \\
\text{read}(B) \\
\text{lock-}X(A) \\
\text{read}(A) \\
\text{if } B = 0 \text{ then } A := A + 1 \\
\text{write}(A) \\
\text{unlock}(B) \\
\text{unlock}(A)

b. Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

\begin{tabular}{c|c}
\text{T}_1 & \text{T}_2 \\
\hline
\text{lock-}S(A) & \text{lock-}S(B) \\
\text{read}(A) & \text{read}(B) \\
\text{lock-}X(B) & \text{lock-}X(A) \\
\end{tabular}

The transactions are now deadlocked.