CompSci 316 Fall 2022
Discussion 12

Presenter: Haibo Xiu

Check in code xxxx (11:15 AM)
Check in code xxxx (14:15 PM)
Concurrency Control
Recap

Serial schedule: Schedule that does not interleave the actions of different transactions.

Serializable schedule: A schedule that is equivalent to some serial execution of the committed transactions.

Recoverable: Each transaction commits after all transactions from which it has read has committed.

Avoids Cascading Rollback: Each transaction reads only data written by committed transactions.
Recap

Precedence graph:

• A node for each transaction

• A directed edge from A to B if an operation of A precedes and conflicts with an operation of B in the schedule

Conflict-serializable schedule: iff its precedence graph has no cycles
Conflict Serializable Schedule - 1

Is this schedule with three transactions T1, T2, T3 conflict serializable? What is an equivalent serial schedule? (Tip: Show the precedence graph.)

Schedule 1:

\[ R2(B), W2(B), R3(C), W3(C), R3(A), W3(A), R2(C), W2(C), R1(A), R1(B), W1(A), W1(B) \]
Conflict Serializable Schedule - 1

Is this conflict serializable? What is an equivalent serial schedule?

Schedule 1:

\[
\text{R2(}B\text{), W2(}B\text{), R3(}C\text{), W3(}C\text{), R3(}A\text{), W3(}A\text{), R2(}C\text{), W2(}C\text{), R1(}A\text{), R1(}B\text{), W1(}A\text{), W1(}B\text{)}
\]

Ans for Schedule 1: Yes.
T1 reads/writes B AFTER T2 reads/writes B.
T2 reads/writes C AFTER T3 reads/writes C.
T1 reads/writes A AFTER T3 reads/writes A.

No cycle in precedence graph.
Equivalent serial schedule: T3 -> T2 -> T1 (Only one)
is this schedule with three transactions T1, T2, T3 conflict serializable? What is an equivalent serial schedule?

Schedule 2:

\[R2(B), W2(B), R3(C), W3(C), R1(A), R1(B), W1(A), W1(B), R2(C), W2(C), R3(A), W3(A)\]
Schedule 2:

\[ R_2(B), W_2(B), R_3(C), W_3(C), R_1(A), W_1(B), W_1(A), R_2(C), W_2(C), R_3(A), W_3(A) \]

Ans for Schedule 2: No.
T1 reads/writes B AFTER T2 reads/writes B.
T2 reads/writes C AFTER T3 reads/writes C.
T3 reads/writes A AFTER T1 reads/writes A.

Precedence graph forms a cycle.
No equivalent serial schedule.
Recoverable & Avoids Cascading Rollback

\[ r_1(X), w_1(Y), r_2(Y), w_2(X), c_2, w_1(X), c_1 \]

Serial?
Conflict Serializable?
Recoverable?
ACR (avoids cascading rollback)?

Reminder
Serial schedule: Schedule that does not interleave the actions of different transactions
Serializable schedule: A schedule that is equivalent to some serial execution of the committed transactions
Recoverable: Each transaction commits after all transactions from which it has read has committed.
Avoids Cascading Rollback: Each transaction reads only data written by committed transactions.
Recoverable & Avoids Cascading Rollback

\[ r_1(X), w_1(Y), r_2(Y), w_2(X), c_2, w_1(X), c_1 \]

Serial? No. Interleave actions between T1 and T2.

Conflict Serializable? No. precedence graph has a cycle. T1<->T2

Recoverable? No. \( r_2(Y) \) after \( w_1(Y) \). But T2 commits before T1 commits.

ACR (avoids cascading rollback)? No. \( r_2(Y) \) after \( w_1(Y) \). T1 does not commit before T2 reads \( r_2(Y) \).
Recap: Two-Phase Locking

Two-phase locking (2PL)

- All lock requests precede all unlock requests
  - Phase 1: obtain locks, phase 2: release locks

For each TX:

Strict 2PL

- Only release locks at commit/abort time
  - A writer will block all other readers until the writer commits or aborts
Two-Phase Locking - 1

Determine if the following schedule can be produced by 2PL

S1: R1(A), R3(C), W3(A), R2(B), W2(D), W1(B)

For a “No” answer, you have to give arguments if asked for explanations. For a “Yes” answer, show a valid lock/unlock sequence following 2PL
Two-Phase Locking - 1

Determine if the following schedule can be produced by 2PL

S1: $R1(A), R3(C), W3(A), R2(B), W2(D), W1(B)$

Ans:
S1 cannot be produced by 2PL. T1 must unlock A before W3(A) and then it cannot acquire the lock on B (all lock requests must precede all unlocks in 2PL). Therefore, 2PL cannot produce S1.
Two-Phase Locking - 2a

Determine if the following schedule can be produced by 2PL

S2: R3(C), R1(A), W1(B), R2(B), W2(D), W3(A), C1, C2, C3

For a “No” answer, you have to give arguments if asked for explanations. For a “Yes” answer, show a valid lock/unlock sequence following 2PL
Two-Phase Locking - 2a

Determine if the following schedule can be produced by 2PL

S2: L3(C), R3(C), L1(A), L1(B), R1(A), W1(B), U1(A), U1(B), L2(B), L2(D), R2(B), W2(D), U2(B), U2(D), L3(A), W3(A), U3(A), U3(C), C1, C2, C3

We use L1(A) for lock (exclusive lock) and U1(A) for unlock of item A by T1, etc.

Ans:
S2 can be produced by 2PL. See the above schedule with lock/unlocks (many other options are provided). For each transaction, all locks precede all unlocks.
Lock/unlock pattern For T1: L1(A), L1(B), U1(A), U1(B)
Lock/unlock pattern For T2: L2(B), L2(D), U2(B), U2(D)
Lock/unlock pattern For T3: L3(C), L3(A), U3(A), U3(C)
For a “Yes” answer, you can give a schedule allowed by 2PL.
Strict Two-Phase Locking - 2b

Determine if the following schedule can be produced by Strict 2PL:

S2: R3(C), R1(A), W1(B), R2(B), W2(D), W3(A), C1, C2, C3

For a “No” answer, you have to give arguments if asked for explanations. For a “Yes” answer, show a valid lock/unlock sequence following 2PL.
Strict Two-Phase Locking - 2b

Determine if the following schedule can be produced by Strict 2PL

S2: \text{L3}(C), \text{R3}(C), \text{L1}(A), \text{L1}(B), \text{R1}(A), \text{W1}(B), \text{U1}(A), \text{U1}(B), \text{L2}(B), \text{L2}(D), \text{R2}(B), \text{W2}(D), \text{U2}(B), \text{U2}(D), \text{L3}(A), \text{W3}(A), \text{U3}(A), \text{U3}(C), \text{C1}, \text{C2}, \text{C3}

We use L1(A) for lock (exclusive lock) and U1(A) for unlock of item A by T1, etc.

Ans: \text{NO}

The above schedule with lock/unlock no longer works (locks are being released before commits).

In fact, no other lock/unlock following Strict 2PL policy works either -- T1 cannot release locks on A and B until the commit C1 happens, so T2 and T3 cannot proceed.

CODE: 4617
Open Office Hour

- HW6 part 1 (Haibo)
- HW6 part 2 (Justin)

CODE: 4617