1. Consider this schedule of two transactions:

| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

- conflict serializable?

1. Consider this schedule of two transactions:

| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

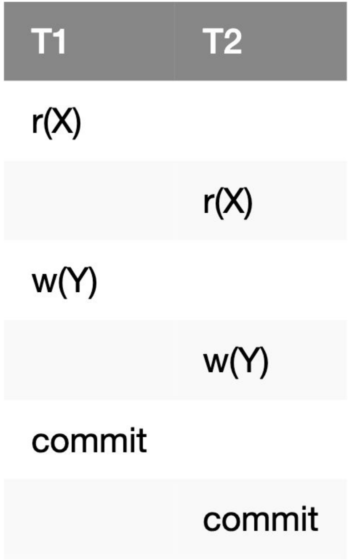
- conflict serializable?

**Identify the Conflicting Actions!**

- Actions in different transactions conflict if:
  1) they involve the same data item

  *and* 2) at least one of them is a write

1. Consider this schedule of two transactions:



| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

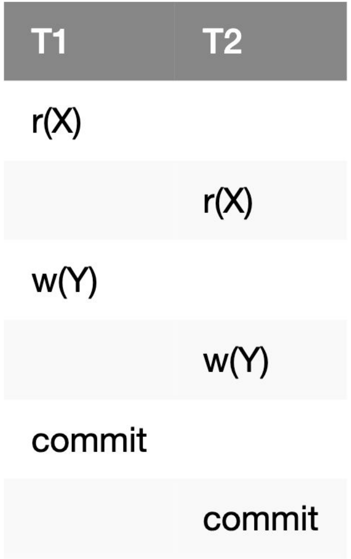
- conflict serializable?

## Identify the Conflicting Actions!

- Actions in different transactions conflict if:
  1) they involve the same data item
  *and* 2) at least one of them is a write

w1(Y), w2(Y) - T1 must come before T2

1. Consider this schedule of two transactions:



| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

- conflict serializable?

w1(Y), w2(Y) - T1 must come before T2

-> Yes, it is Conflict Serializable

1. Consider this schedule of two transactions:

| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

- conflict serializable?
- serializable?

yes. if it’s conflict serializable, it’s serializable

1. Consider this schedule of two transactions:

| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

- conflict serializable?
- serializable?
- recoverable?

Are there dirty reads?

1. Consider this schedule of two transactions:

| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

- conflict serializable?
- serializable?
- recoverable?

Are there dirty reads?
**no dirty reads means recoverable**

1. Consider this schedule of two transactions:

| T1 | T2 |
|---|---|
| r(X) | |
| | r(X) |
| w(Y) | |
| | w(Y) |
| commit | |
| | commit |

Is this schedule:

- conflict serializable?
- serializable?
- recoverable?
- cascadeless?

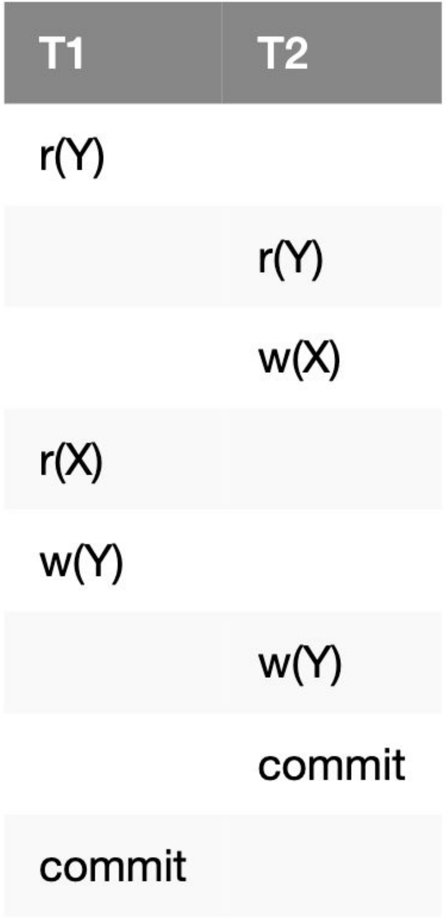
Are there dirty reads?

**no dirty reads also means cascadeless**

| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- conflict serializable?

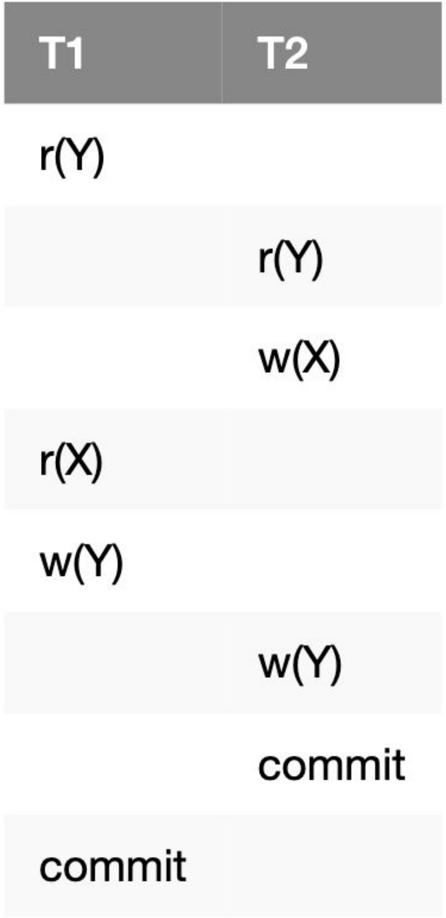
Conflicting Pairs:



| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- conflict serializable?

Conflicting Pairs:
w2(x), r1(x), T2 -> T1



| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

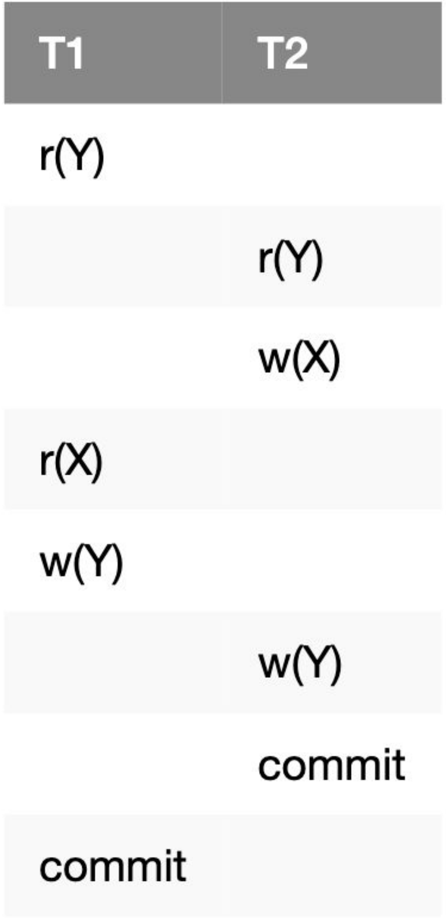
- conflict serializable?

Conflicting Pairs:
w2(x), r1(x), T2 -> T1
w1(Y), w2(Y), T1 -> T2

Constraints contradict -> Not conflict serializable

| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- serializable?



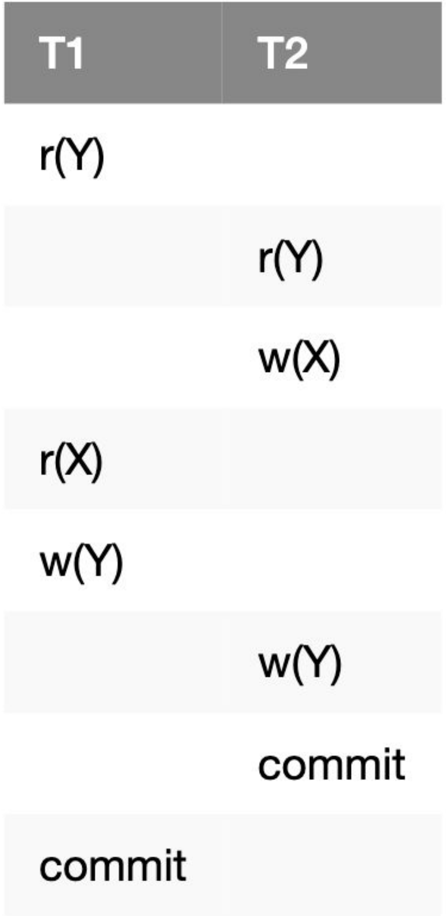
| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- serializable?

No.

T1 read T2’s write of X
But T2 wrote the final value to Y

It is neither equivalent to T1;T2 nor T2;T1



| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

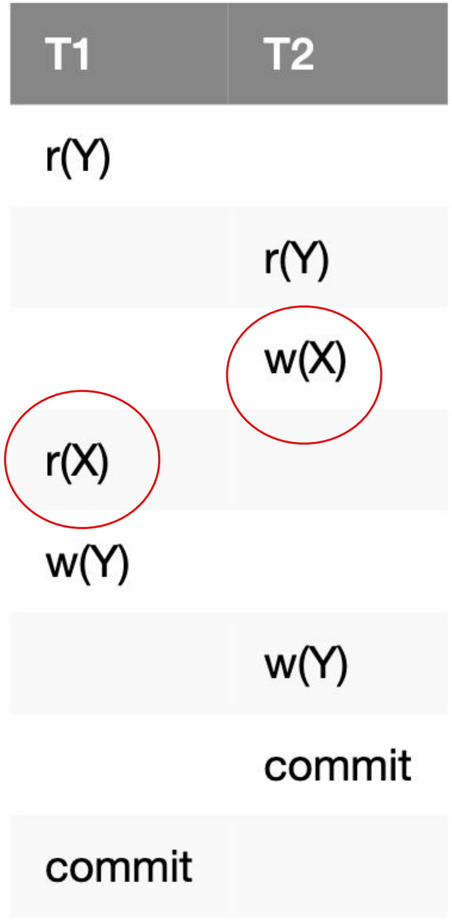
- recoverable?

Is there a dirty read?

| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- recoverable?

Is there a dirty read?



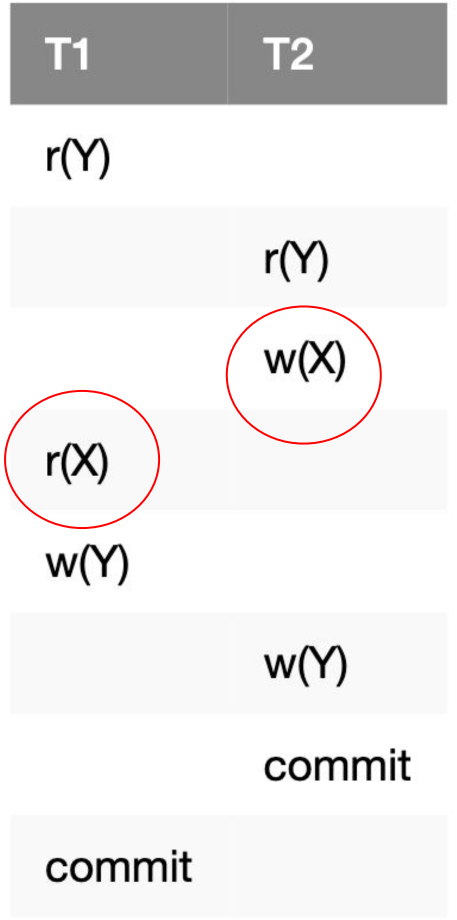
| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- recoverable?

T2 must commit before T1 in order for it to be recoverable

Yes

If we swapped the order of commit, T2 will commit after T1 — no longer recoverable



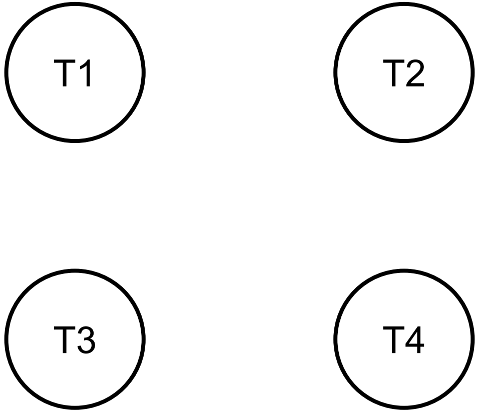
| T1 | T2 |
|---|---|
| r(Y) | |
| | r(Y) |
| | w(X) |
| r(X) | |
| w(Y) | |
| | w(Y) |
| | commit |
| commit | |

- cascadeless?

No. There’s a dirty read

|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

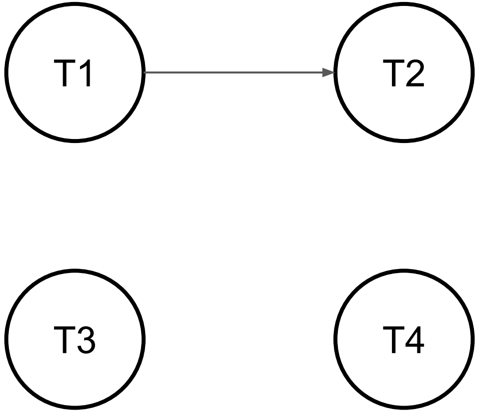
## Conflicting Pairs



|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

## Conflicting Pairs

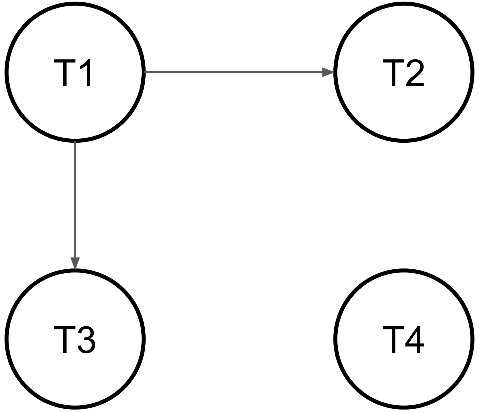
- r1(X), w2(X), T1 -> T2



|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

## Conflicting Pairs

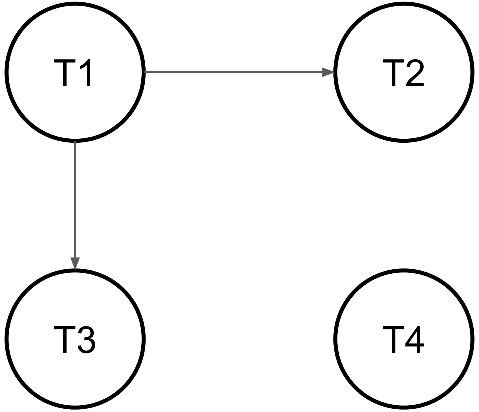
- r1(X), w2(X), T1 -> T2
- w1(Y), r3(Y), T1 -> T3



|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

## Conflicting Pairs

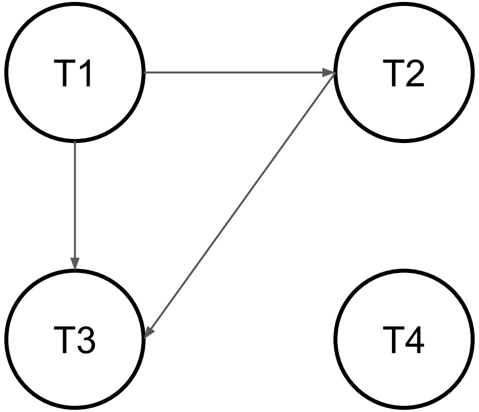
- r1(X), w2(X), T1 -> T2
- w1(Y), r3(Y), T1 -> T3
- w1(Y), r2(Y), T1 -> T2



|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

## Conflicting Pairs

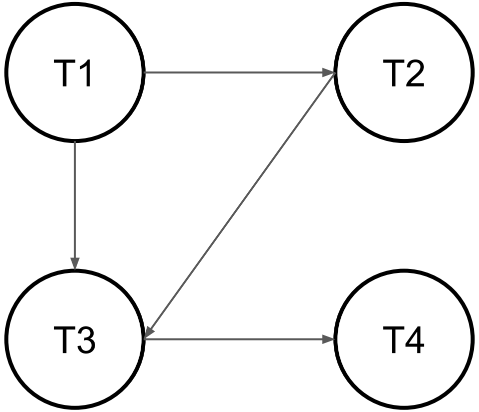
- r1(X), w2(X), T1 -> T2
- w1(Y), r3(Y), T1 -> T3
- w1(Y), r2(Y), T1 -> T2
- r2(Y), w3(Y), T2 -> T3



|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

## Conflicting Pairs

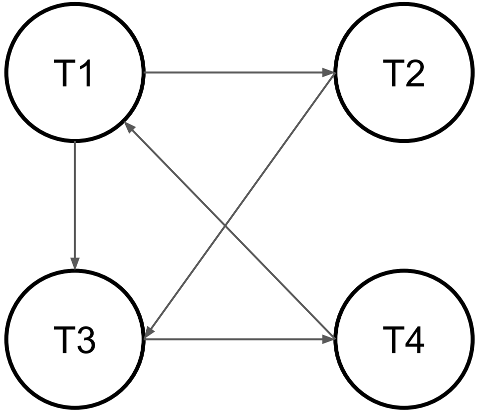
- r1(X), w2(X), T1 -> T2
- w1(Y), r3(Y), T1 -> T3
- w1(Y), r2(Y), T1 -> T2
- r2(Y), w3(Y), T2 -> T3
- r3(W), w4(W), T3 -> T4



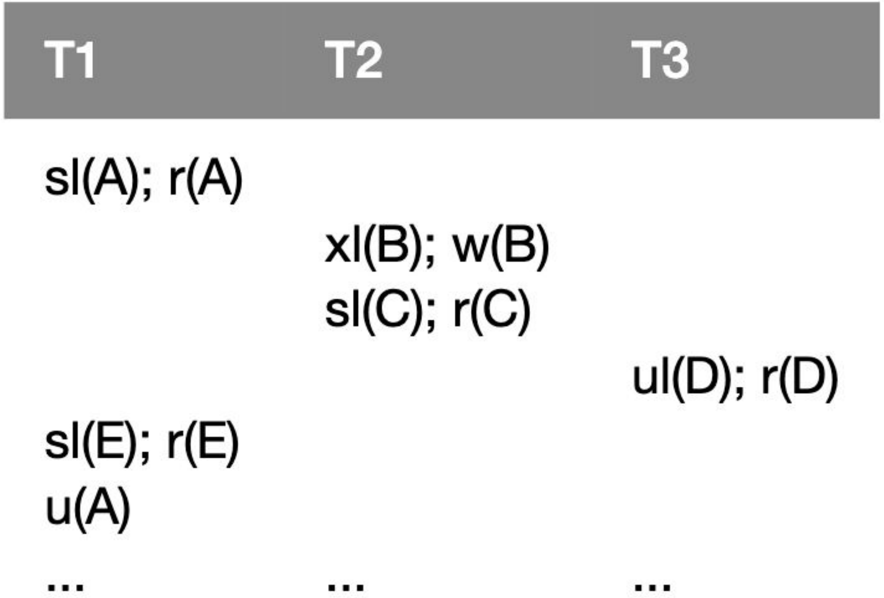
|  | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | r(X) |  |  |  |
| 2 |  | r(X) |  |  |
| 3 | w(Y) |  |  |  |
| 4 |  |  | r(Y) |  |
| 5 |  | r(Y) |  |  |
| 6 |  | w(X) |  |  |
| 7 |  |  | r(W) |  |
| 8 |  |  | w(Y) |  |
| 9 |  |  |  | r(W) |
| 10 |  |  |  | r(Z) |
| 11 |  |  |  | w(W) |
| 12 | r(Z) |  |  |  |
| 13 | w(Z) |  |  |  |

## Conflicting Pairs

- r1(X), w2(X), T1 -> T2
- w1(Y), r3(Y), T1 -> T3
- w1(Y), r2(Y), T1 -> T2
- r2(Y), w3(Y), T2 -> T3
- r3(W), w4(W), T3 -> T4
- r4(Z), w1(Z), T4 -> T1



# Regular 2PL, uses update locks

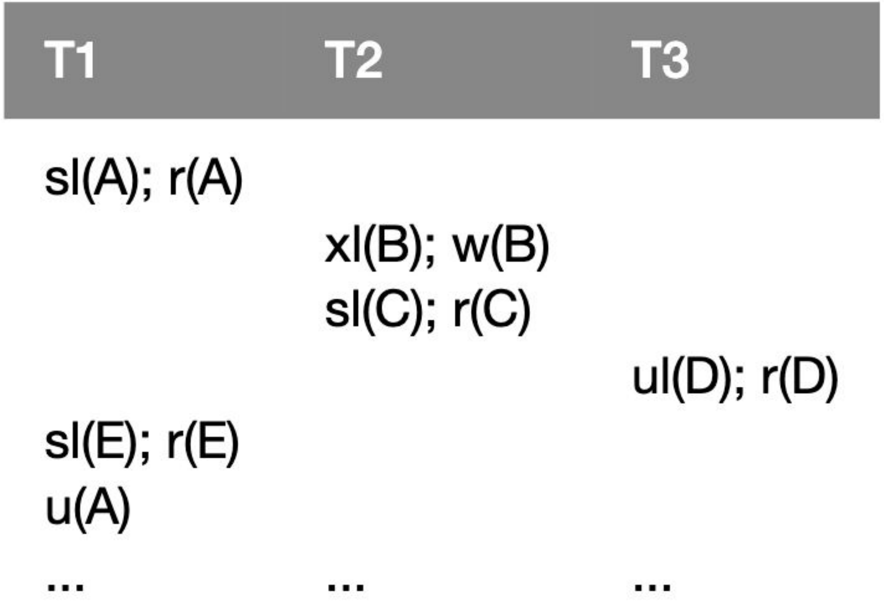


| T1 | T2 | T3 |
|---|---|---|
| sl(A); r(A) | | |
| | xl(B); w(B) | |
| | sl(C); r(C) | |
| | | ul(D); r(D) |
| sl(E); r(E) | | |
| u(A) | | |
| ... | ... | ... |

Given the locks that the transactions would need to acquire, which of the following operations could happen next in the schedule?

1. r1(F)
2. w2(A)
3. r2(E)
4. w3(D)
5. w2(C)

# Regular 2PL, uses update locks

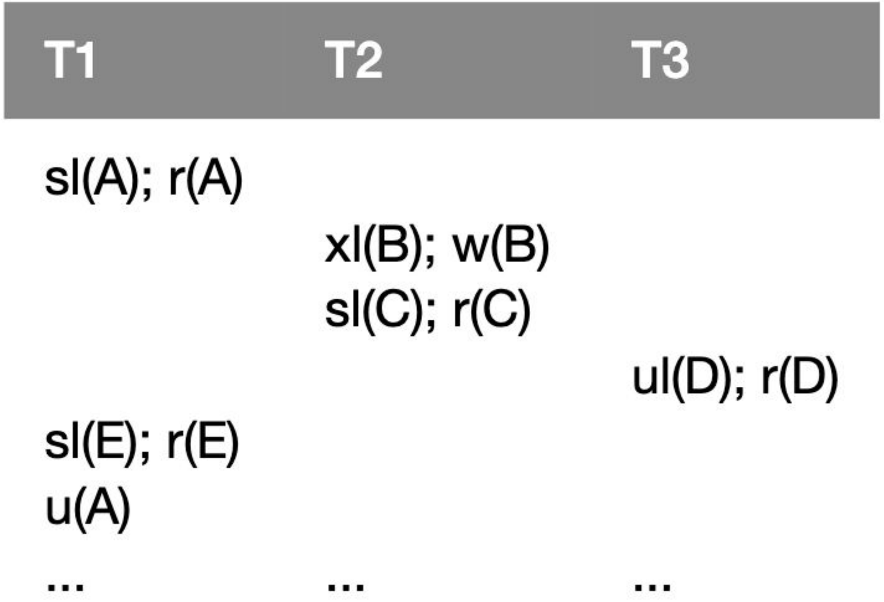


| T1 | T2 | T3 |
|---|---|---|
| sl(A); r(A) | | |
| | xl(B); w(B) | |
| | sl(C); r(C) | |
| | | ul(D); r(D) |
| sl(E); r(E) | | |
| u(A) | | |
| ... | ... | ... |

Given the locks that the transactions would need to acquire, which of the following operations could happen next in the schedule?

1. r1(F) No. 2PL - can't lock after unlock
2. w2(A)
3. r2(E)
4. w3(D)
5. w2(C)

# Regular 2PL, uses update locks

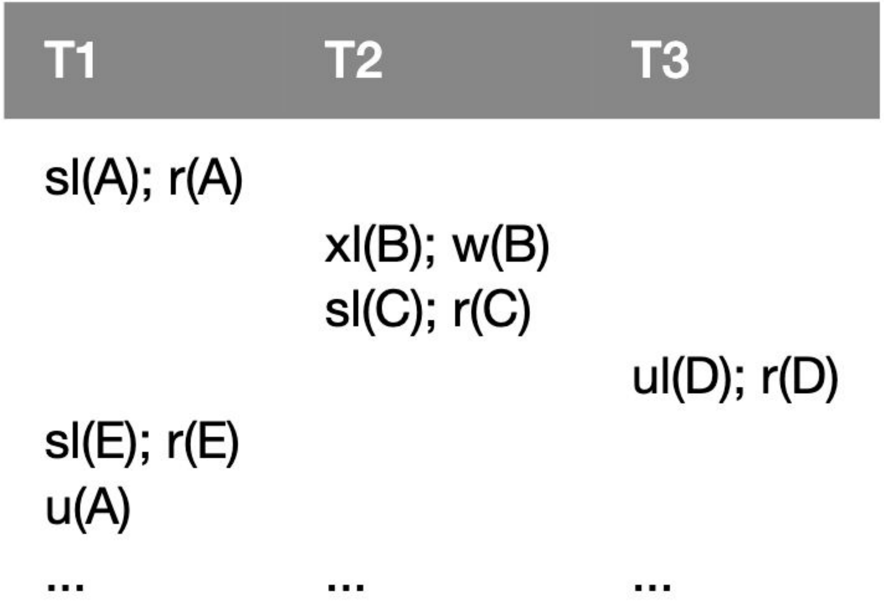


| T1 | T2 | T3 |
|---|---|---|
| sl(A); r(A) | | |
| | xl(B); w(B) | |
| | sl(C); r(C) | |
| | | ul(D); r(D) |
| sl(E); r(E) | | |
| u(A) | | |
| ... | ... | ... |

Given the locks that the transactions would need to acquire, which of the following operations could happen next in the schedule?

1. r1(F) No. 2PL - can’t lock after unlock
2. w2(A) Yes, no one else has lock A
3. r2(E)
4. w3(D)
5. w2(C)

# Regular 2PL, uses update locks

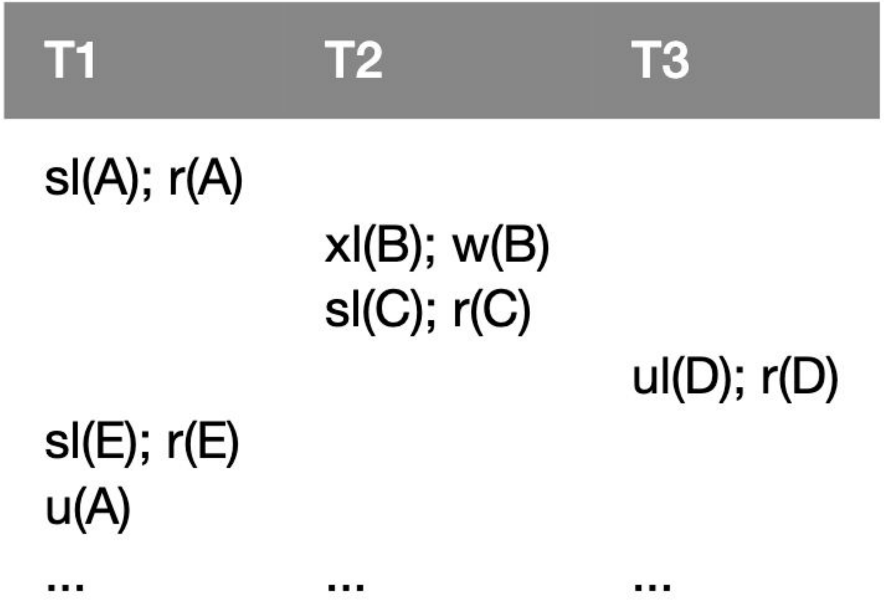


| T1 | T2 | T3 |
|---|---|---|
| sl(A); r(A) | | |
| | xl(B); w(B) | |
| | sl(C); r(C) | |
| | | ul(D); r(D) |
| sl(E); r(E) | | |
| u(A) | | |
| ... | ... | ... |

Given the locks that the transactions would need to acquire, which of the following operations could happen next in the schedule?

1. r1(F) No. 2PL - can't lock after unlock
2. w2(A) Yes, no one else has lock A
3. r2(E) Yes, get a shared lock for E
4. w3(D)
5. w2(C)

# Regular 2PL, uses update locks

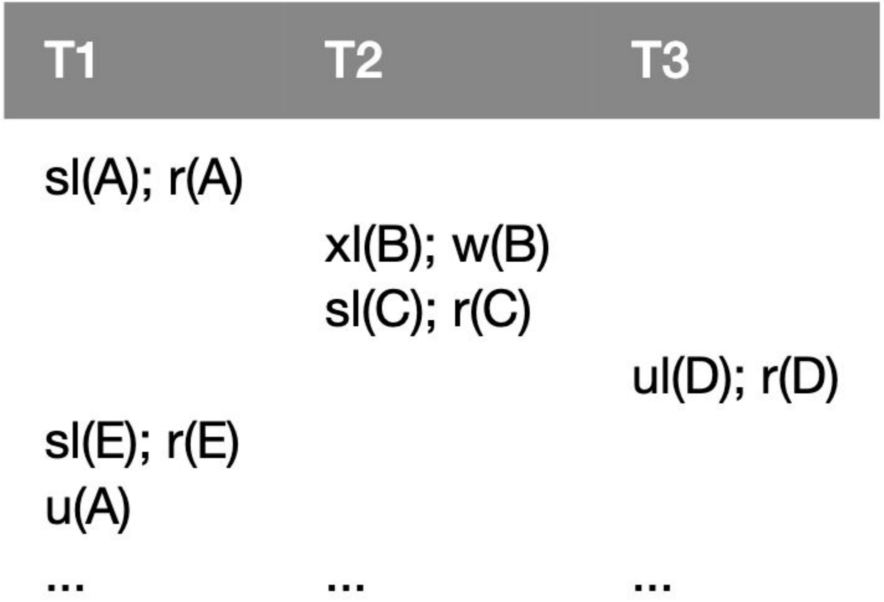


| T1 | T2 | T3 |
|---|---|---|
| sl(A); r(A) | | |
| | xl(B); w(B) | |
| | sl(C); r(C) | |
| | | ul(D); r(D) |
| sl(E); r(E) | | |
| u(A) | | |
| ... | ... | ... |

Given the locks that the transactions would need to acquire, which of the following operations could happen next in the schedule?

1. r1(F) No. 2PL - can’t lock after unlock
2. w2(A) Yes, no one else has lock A
3. r2(E) Yes, get a shared lock for E
4. w3(D) Yes, update the lock to exclusive
5. w2(C)

## Regular 2PL, uses update locks



| T1 | T2 | T3 |
|---|---|---|
| sl(A); r(A) | | |
| | xl(B); w(B) | |
| | sl(C); r(C) | |
| | | ul(D); r(D) |
| sl(E); r(E) | | |
| u(A) | | |
| ... | ... | ... |

Given the locks that the transactions would need to acquire, which of the following operations could happen next in the schedule?

1. r1(F) No. 2PL - can’t lock after unlock
2. w2(A) Yes, no one else has lock A
3. r2(E) Yes, get a shared lock for E
4. w3(D) Yes, update the lock to exclusive
5. w2(C) No. If uses update locks, cannot go from shared to exclusive