An Evolution of ANSI Isolation Levels

Ethan Zhu

Department of Computer Science and Technology
Nanjing University

Nov, 7, 2020

Table of Contents

Table of Contents

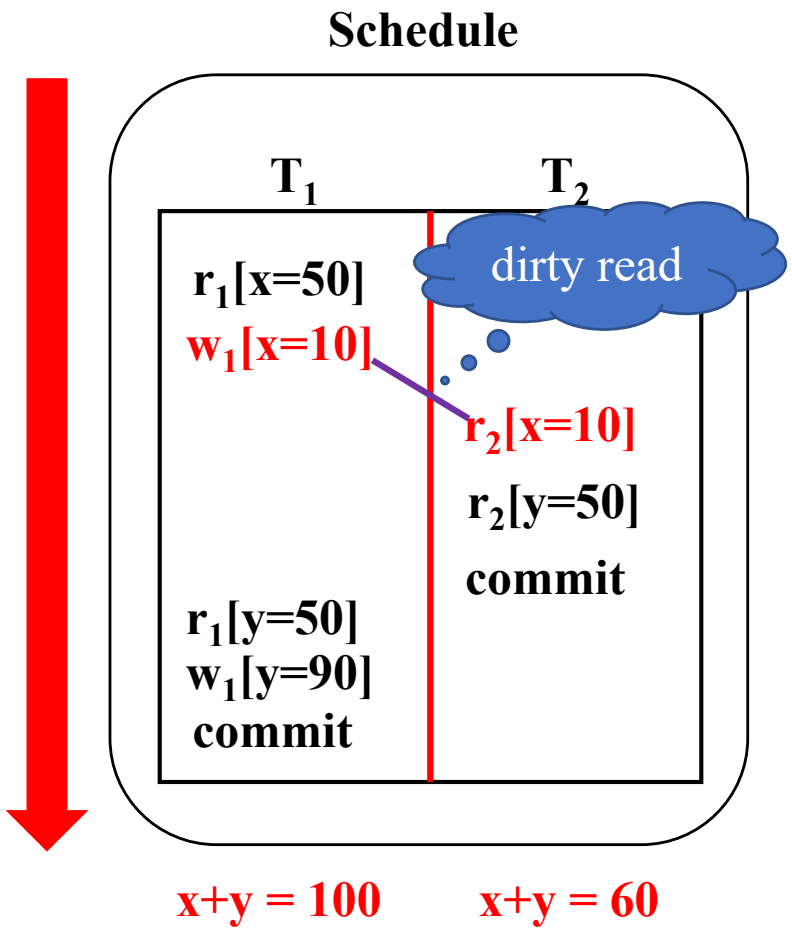
ANSI SQL-92 Isolation Levels

Table 1. ANSI SQL Isolation Levels Defined in terms of the Three Original Phenomena			
Isolation Level	**P1 (or A1) Dirty Read**	**P2 (or A2) Fuzzy Read**	**P3 (or A3) Phantom**
ANSI READ UNCOMMITTED	Possible	Possible	Possible
ANSI READ COMMITTED	Not Possible	Possible	Possible
ANSI REPEATABLE READ	Not Possible	Not Possible	Possible
ANOMALY SERIALIZABLE	Not Possible	Not Possible	Not Possible

- Dirty Read
  - $P0 : W_1(x)...R_2(x)...((c_1 \text{ or } a_1) \text{ and } (c_2 \text{ or } a_2) \text{ in any order})$
  - $A0 : W_1(x)...R_2(x)...(a_1 \text{ and } c_2 \text{ in any order})$
- Fuzzy Read
  - $P1 : R_1(x)...W_2(x)... ((c_1 \text{ or } a_1) \text{ and } (c_2 \text{ or } a_2) \text{ in any order})$
  - $A1 : R_1(x)...W_2(x)... \ c_2...R_1(x) \ ...c_1$
- Phantom
  - $P2 : R_1(P)...W_2(y \text{ in } P)... ((c_1 \text{ or } a_1) \text{ and } (c_2 \text{ or } a_2) \text{ in any order})$
  - $A2 : R_1(P)...W_2(y \text{ in } P)... \ c_2...R_1(P) \ ...c_1$

Analyzing ANSI SQL Isolation Levels: Dirty Read

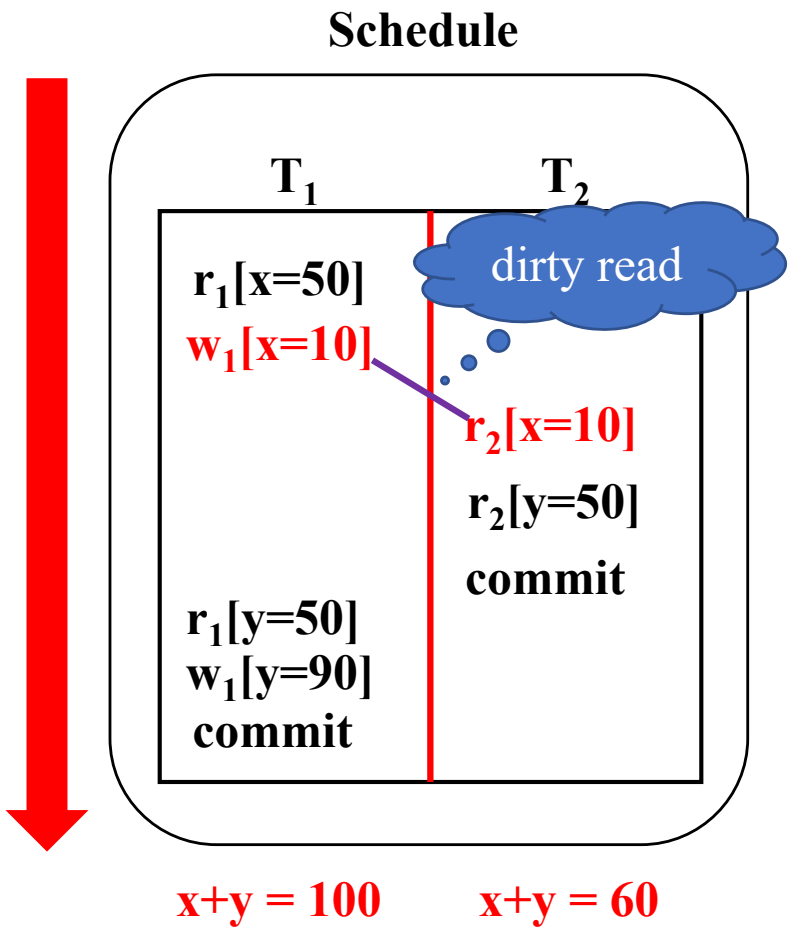
- There is a technical distinction between anomalies and phenomena.



- The direction from $w_1[x = 10]$ to $r_2[x = 10]$ and from $r_2[y = 50]$ to $w_1[y = 90]$.

Analyzing ANSI SQL Isolation Levels: Dirty Read

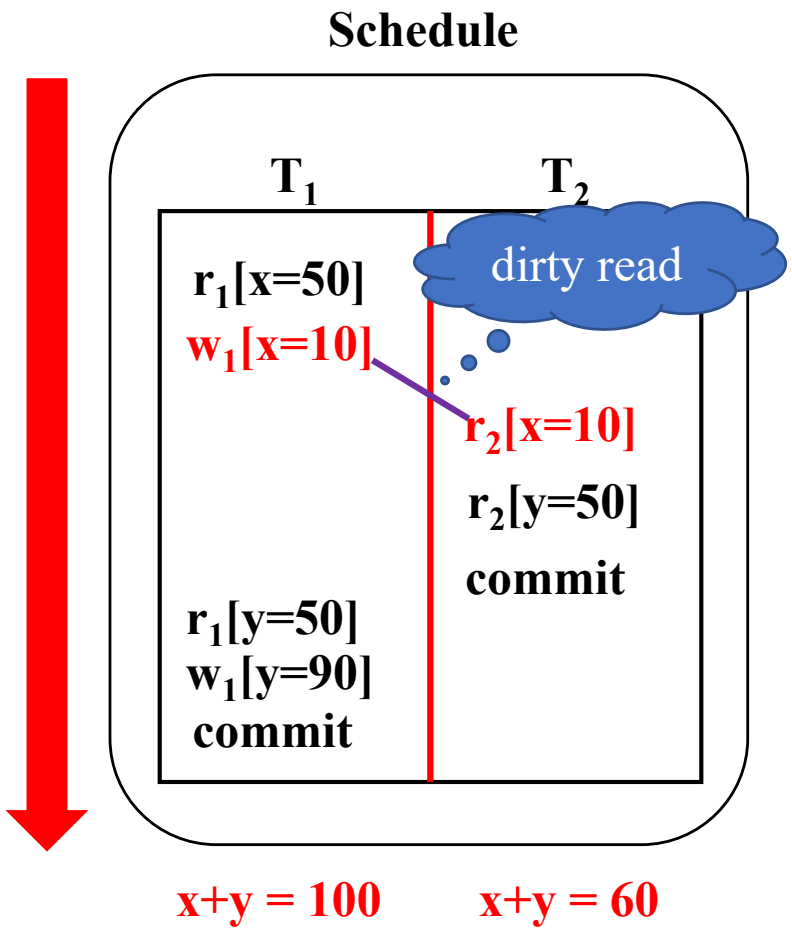
- There is a technical distinction between anomalies and phenomena.



- The direction from $w_1[x = 10]$ to $r_2[x = 10]$ and from $r_2[y = 50]$ to $w_1[y = 90]$.
- The schedule is non-serializability due to dirty read.

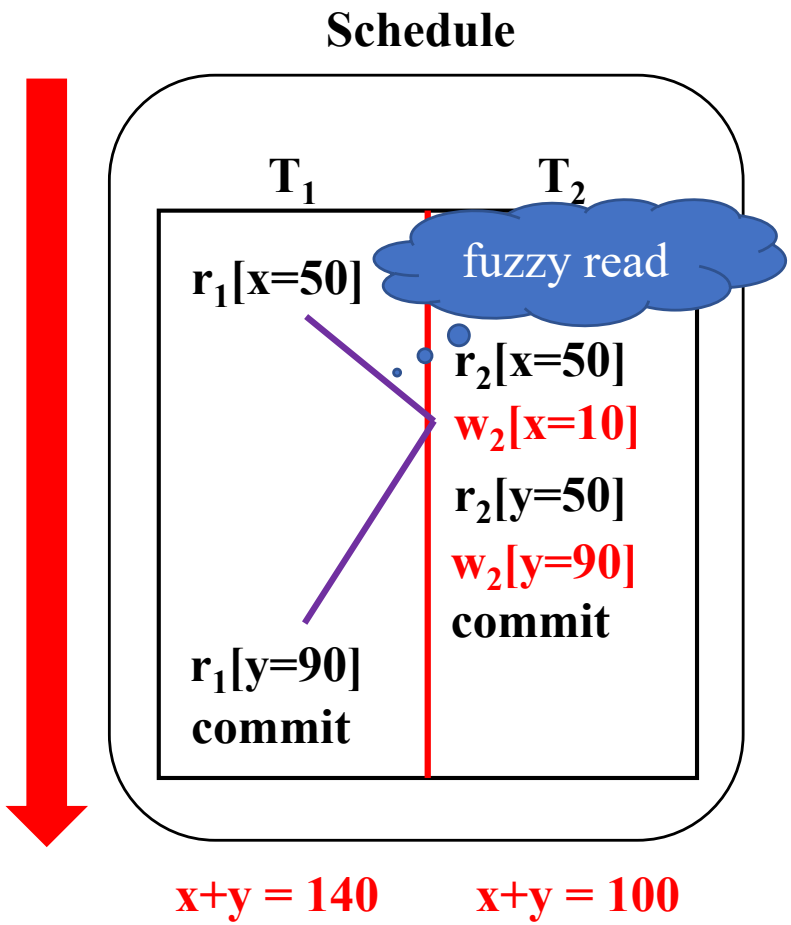
Analyzing ANSI SQL Isolation Levels: Dirty Read

- There is a technical distinction between anomalies and phenomena.



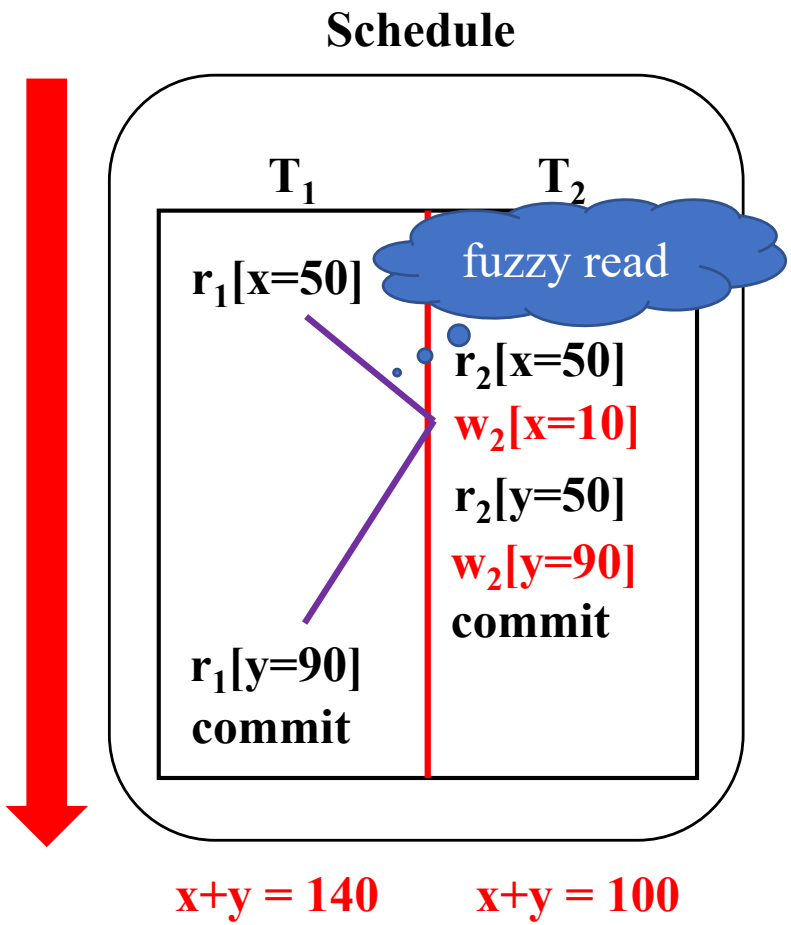
- The direction from $w_1[x = 10]$ to $r_2[x = 10]$ and from $r_2[y = 50]$ to $w_1[y = 90]$.
- The schedule is non-serializability due to dirty read.
- So, The dirty read means P0 instead of A0.

Analyzing ANSI SQL Isolation Levels: Fuzzy Read



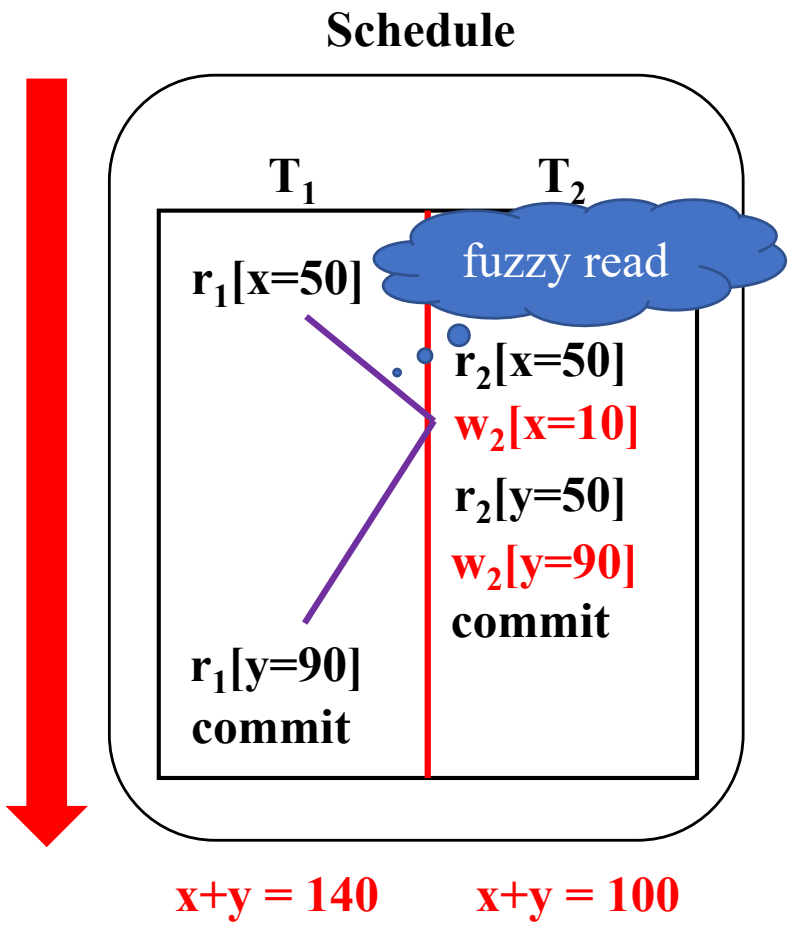
- The direction from $w_1[x = 10]$ to $r_2[x = 10]$ and from $w_2[y = 90]$ to $r_1[y = 90]$.

Analyzing ANSI SQL Isolation Levels: Fuzzy Read



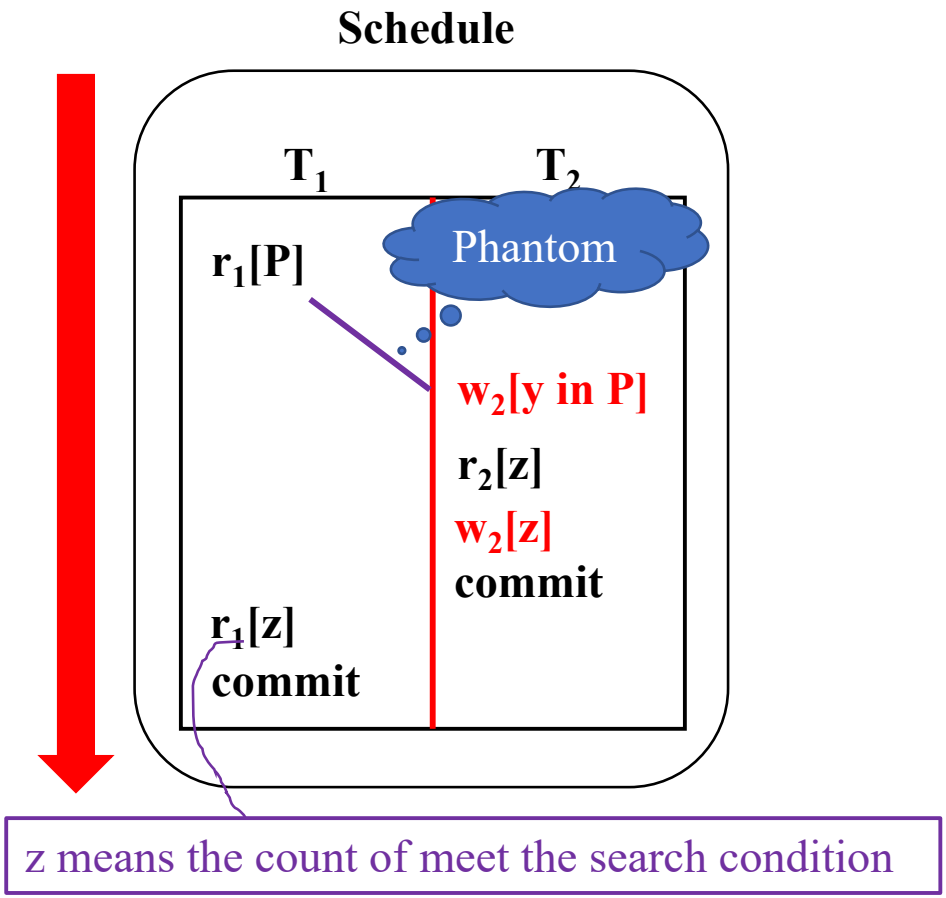
- The direction from $w_1[x = 10]$ to $r_2[x = 10]$ and from $w_2[y = 90]$ to $r_1[y = 90]$.
- The schedule is non-serializability due to fuzzy read.

Analyzing ANSI SQL Isolation Levels: Fuzzy Read



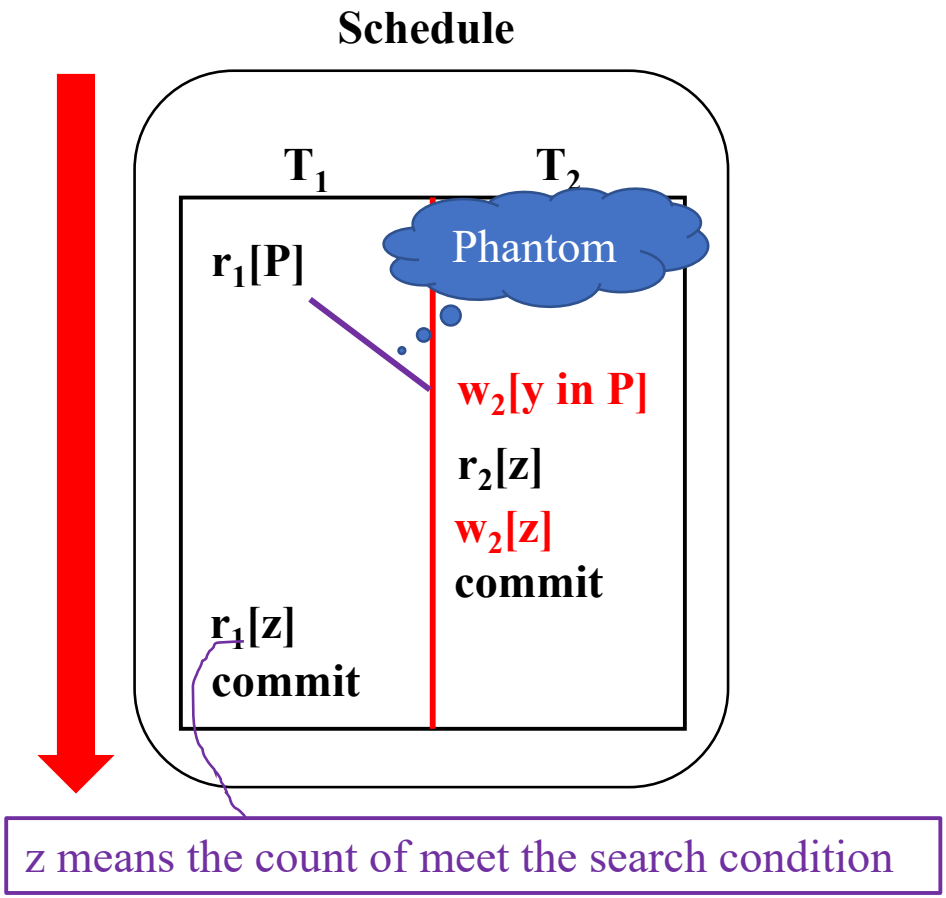
- The direction from $w_1[x = 10]$ to $r_2[x = 10]$ and from $w_2[y = 90]$ to $r_1[y = 90]$.
- The schedule is non-serializability due to fuzzy read.
- So, The fuzzy read means P1 instead of A1.

Analyzing ANSI SQL Isolation Levels: Phantom



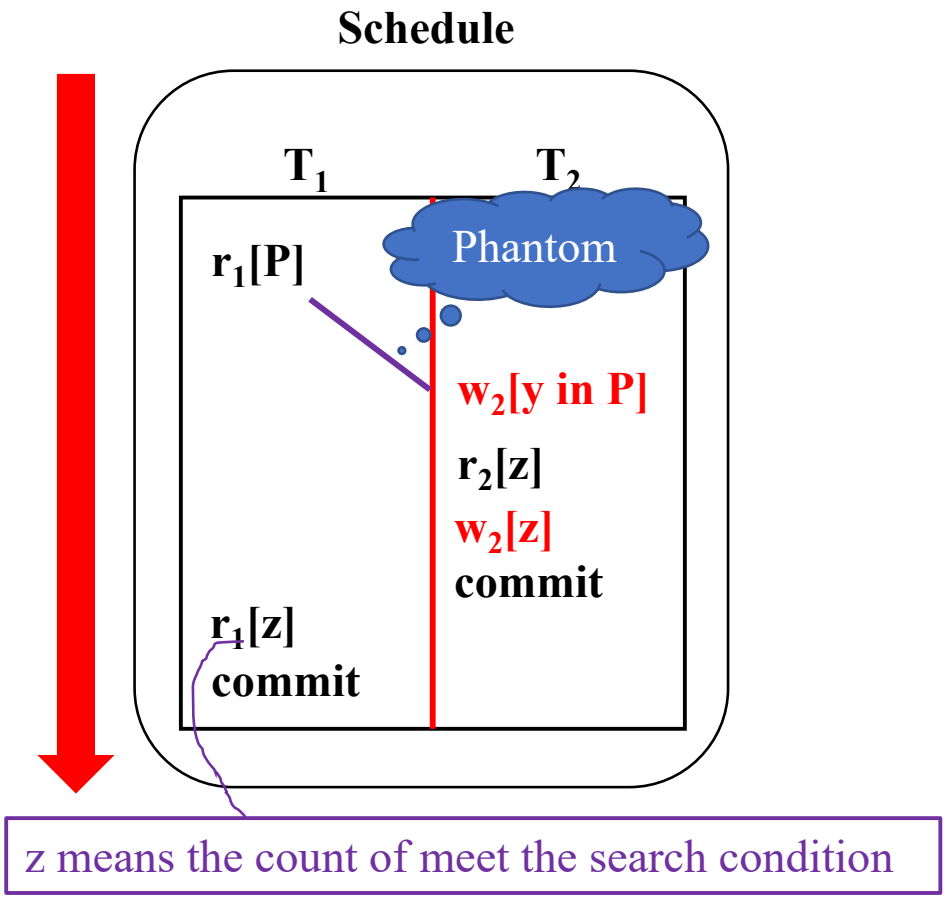
- The direction from $r_1[P]$ to $w_2[y\ in\ P]$ and from $w_2[z]$ to $r_1[z]$.

Analyzing ANSI SQL Isolation Levels: Phantom



- The direction from $r_1[P]$ to $w_2[y\ in\ P]$ and from $w_2[z]$ to $r_1[z]$.
- The schedule is non-serializability due to phantom.

Analyzing ANSI SQL Isolation Levels: Phantom



- The direction from $r_1[P]$ to $w_2[y\ in\ P]$ and from $w_2[z]$ to $r_1[z]$.
- The schedule is non-serializability due to phantom.
- So, The phantom means P2 instead of A2.

Conclusions of ANSI Isolation Levels

- ANSI isolation levels firbid phenomena instead of anomaly.

Conclusions of ANSI Isolation Levels

- ANSI isolation levels firbid phenomena instead of anomaly.

Question: Is the current phenomenon adequate?

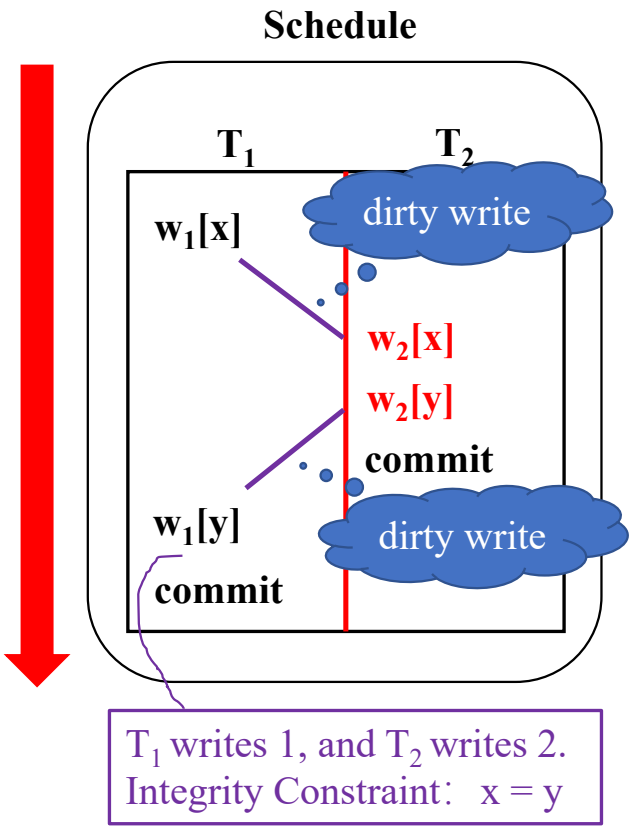
Conclusions of ANSI Isolation Levels

- ANSI isolation levels firbid phenomena instead of anomaly.

Question: Is the current phenomenon adequate?

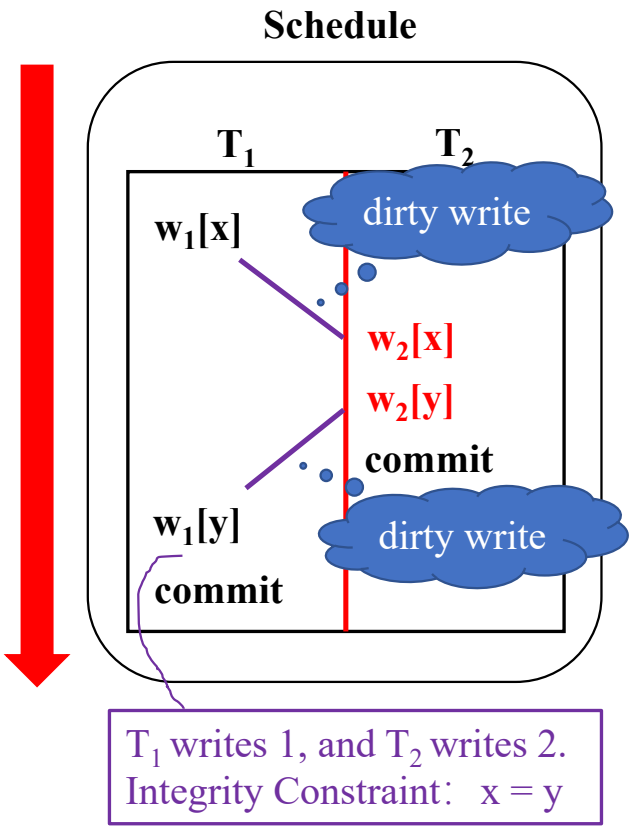
Answer: ANSI SQL isolation phenomena are incomplete. There are a number of anomalies that still can arise.

New Phenomena: Dirty Write



- The direction from $w_1[x]$ to $w_2[x]$ and from $w_2[y]$ to $w_1[y]$.

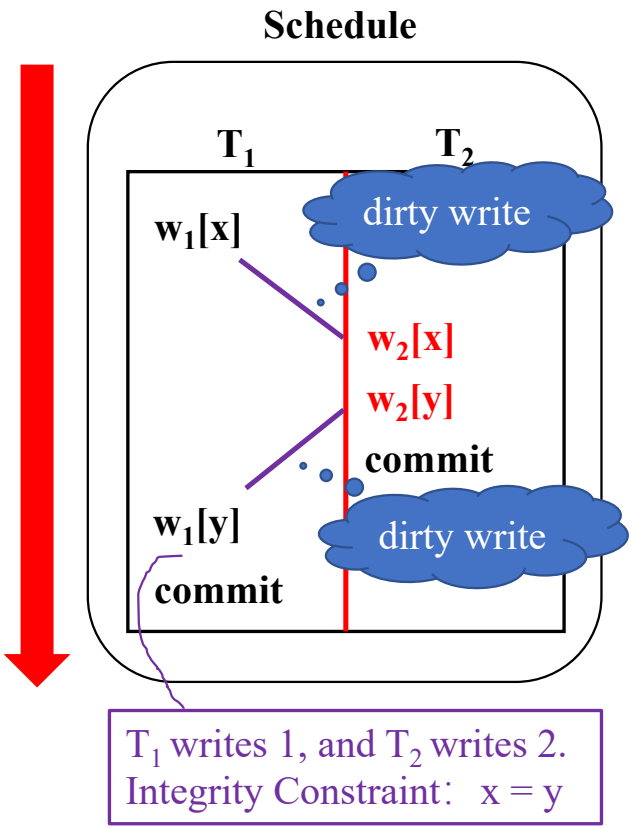
New Phenomena: Dirty Write



- The direction from $w_1[x]$ to $w_2[x]$ and from $w_2[y]$ to $w_1[y]$.
- The schedule is non-serializability due to dirty write.

$$P0 : W_1[x] \ldots W_2[x] \ldots ((c_1 \text{ or } a_1) \text{ and } (c_2 \text{ or } a_2) \text{ in any order})$$

New Phenomena: Dirty Write



- The direction from $w_1[x]$ to $w_2[x]$ and from $w_2[y]$ to $w_1[y]$.
- The schedule is non-serializability due to dirty write.
- So, The dirty write means P0.

$$P0 : W_1[x] \ldots W_2[x] \ldots ((c_1 \text{ or } a_1) \text{ and } (c_2 \text{ or } a_2) \text{ in any order})$$

Modification of ANSI Isolation Levels

Table 3. ANSI SQL Isolation Levels Defined in terms of the four phenomena

Isolation Level	P0 Dirty Write	P1 Dirty Read	P2 Fuzzy Read	P3 Phantom
READ UNCOMMITTED	Not Possible	Possible	Possible	Possible
READ COMMITTED	Not Possible	Not Possible	Possible	Possible
REPEATABLE READ	Not Possible	Not Possible	Not Possible	Possible
SERIALIZABLE	Not Possible	Not Possible	Not Possible	Not Possible

- ANSI SQL isolation should be modified to require P0 for all isolation levels.
- The interpretations of A1, A2, and A3 have unintended weaknesses. The correct interpretations are the P1, P2, and P3. We assume in what follows that ANSI meant to define P1, P2, and P3.
- ANSI SQL isolation phenomena are incomplete.
  - ✓ There are a number of anomalies that still can arise.

Definitions of Phenomenon

P0: w1[x]...w2[x]...(c1 or a1) (Dirty Write)

P1: w1[x]...r2[x]...(c1 or a1) (Dirty Read)

P2: r1[x]...w2[x]...(c1 or a1) (Fuzzy or Non-Repeatable Read)

P3: r1[P]...w2[y in P]...(c1 or a1) (Phantom)

Definitions of Phenomenon

P0: w1[x]...w2[x]...(c1 or a1) (Dirty Write)
P1: w1[x]...r2[x]...(c1 or a1) (Dirty Read)
P2: r1[x]...w2[x]...(c1 or a1) (Fuzzy or Non-Repeatable Read)
P3: r1[P]...w2[y in P]...(c1 or a1) (Phantom)

Question 1: How to do we characterisrtic the differences between some isolation levels (e.g., READ committed) and serializability isolation level?

Definitions of Phenomenon

P0: w1[x]...w2[x]...(c1 or a1) (Dirty Write)
P1: w1[x]...r2[x]...(c1 or a1) (Dirty Read)
P2: r1[x]...w2[x]...(c1 or a1) (Fuzzy or Non-Repeatable Read)
P3: r1[P]...w2[y in P]...(c1 or a1) (Phantom)

Question 1: How to do we characterisrtic the differences between some isolation levels (e.g., READ committed) and serializability isolation level?

Answer 1: Phenomenon or Anomalies!

Table of Contents

Locking Scopes, Modes, Durations

- Well-formed write and read
- Locking scopes
  - ✓ Data items
  - ✓ Predicates
- Locking modes
  - ✓ Read
  - ✓ Write
- Locking duration
  - ✓ Short duration
  - ✓ Long duration

Locking Scopes, Modes, Durations

- Well-formed write and read
- Locking scopes
  - ✓ Data items
  - ✓ Predicates
- Locking modes
  - ✓ Read
  - ✓ Write
- Locking duration
  - ✓ Short duration
  - ✓ Long duration

The fundamental serialization theorem is that well-formed long duration two-phase locking guarantees serializability!

Locking-based Isolation Levels

Table 2. Degrees of Consistency and Locking Isolation Levels defined in terms of locks.

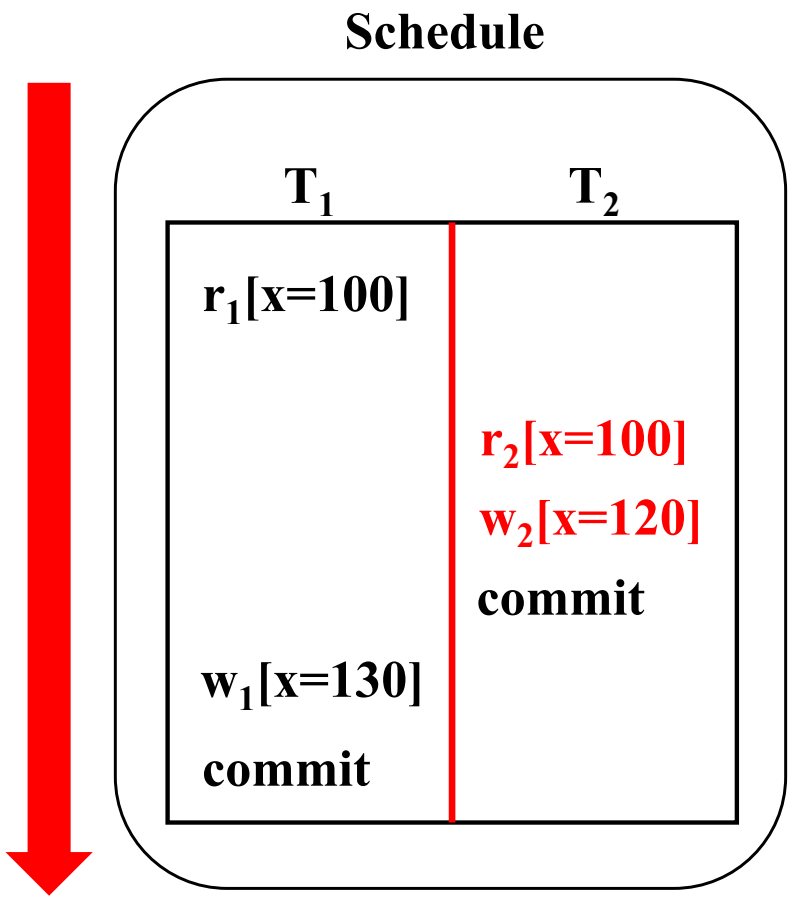
Consistency Level = Locking Isolation Level	**Read Locks on Data Items and Predicates (the same unless noted)**	**Write Locks on Data Items and Predicates (always the same)**
Degree 0	none required	Well-formed Writes
Degree 1 = Locking READ UNCOMMITTED	none required	Well-formed Writes Long duration Write locks
Degree 2 = Locking READ COMMITTED	Well-formed Reads Short duration Read locks (both)	Well-formed Writes, Long duration Write locks
Cursor Stability (see Section 4.1)	Well-formed Reads Read locks held on current of cursor Short duration Read Predicate locks	Well-formed Writes, Long duration Write locks
Locking REPEATABLE READ	Well-formed Reads Long duration data-item Read locks Short duration Read Predicate locks	Well-formed Writes, Long duration Write locks
Degree 3 = Locking SERIALIZABLE	Well-formed Reads Long duration Read locks (both)	Well-formed Writes, Long duration Write locks

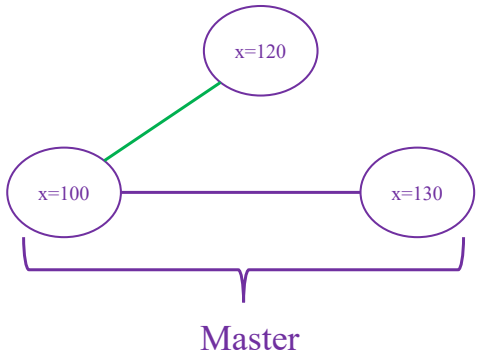
Conclusions

- The locking isolation levels of Table 2 and the phenomenological definitions of Table 3 are equivalent.
- Put another way, P0, P1, P2, and P3 are disguised redefinition's of locking behavior.
- In what follows, we will refer to the isolation levels listed in Table 3 by the names in Table 3, equivalent to the Locking versions of these isolation levels of Table 2.
- When we refer to ANSI READ UNCOMMITTED, ANSI READ COMMITTED, ANSI REPEATABLE READ, and ANOMALY SERIALIZABLE, we are referring to the ANSI definition of Table 1 (inadequate, since it did not include P0).

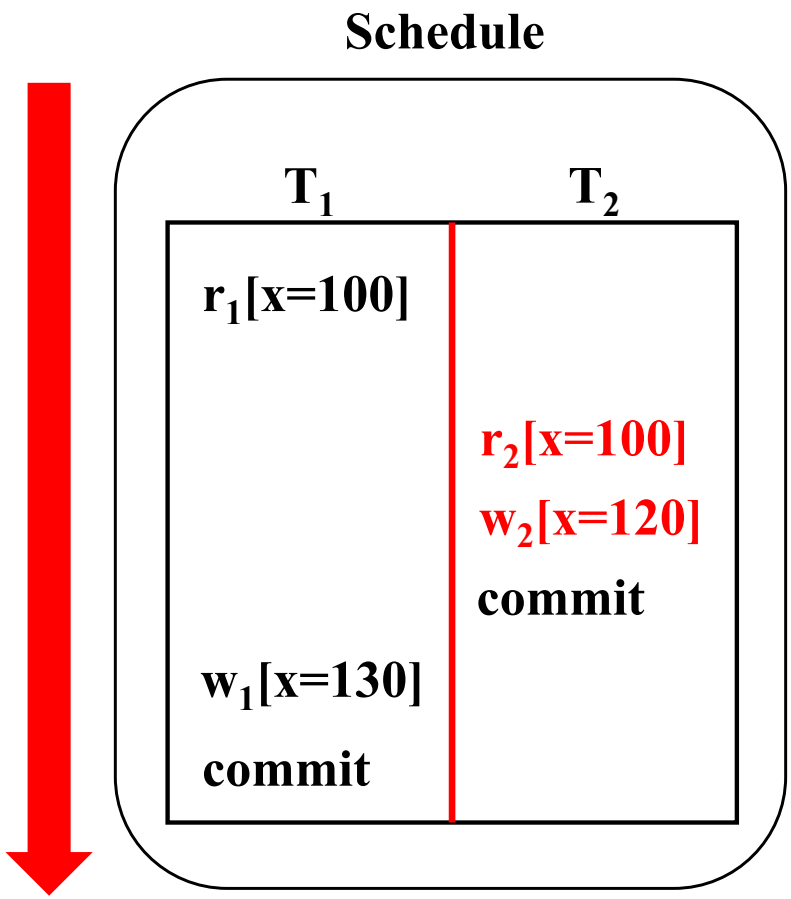
Table of Contents

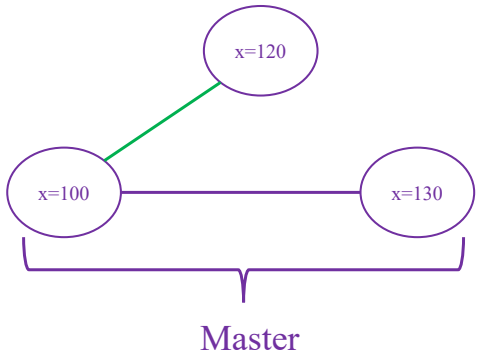
New Phenomena: Lost Update





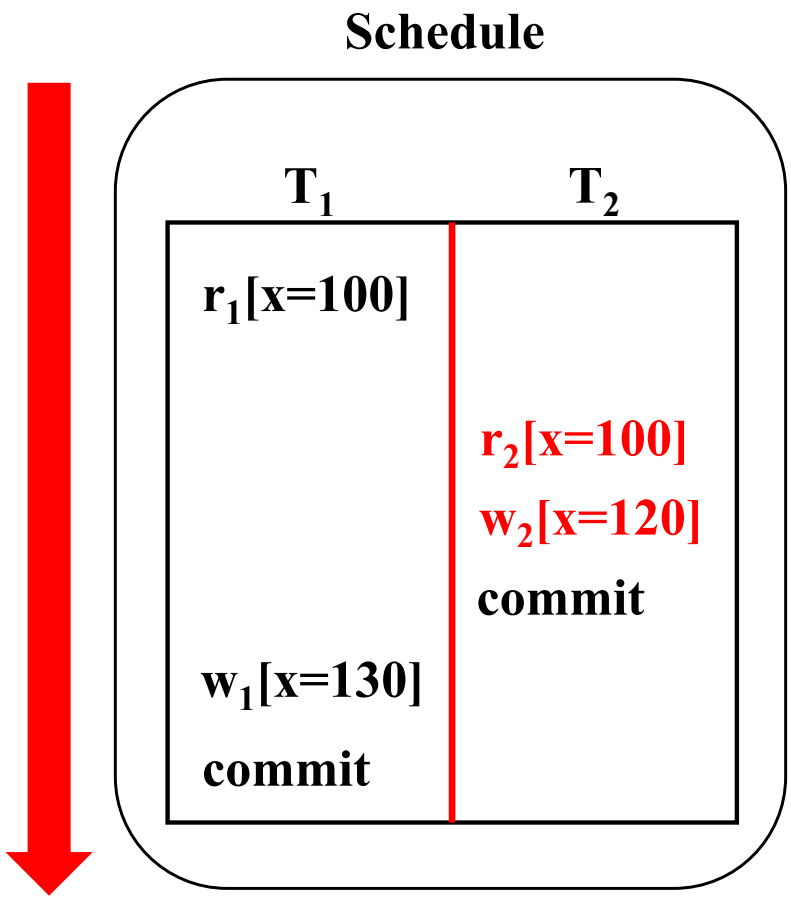
New Phenomena: Lost Update

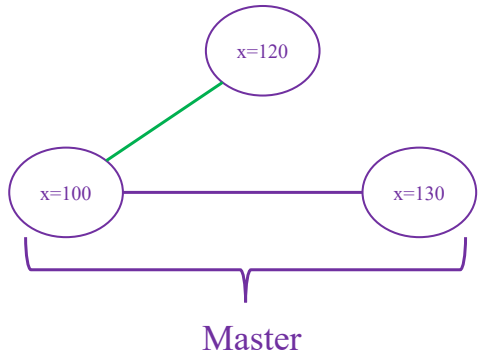




- The final value of x contains only the increment of 30 added by T1.
- It is non-serializability schedule.

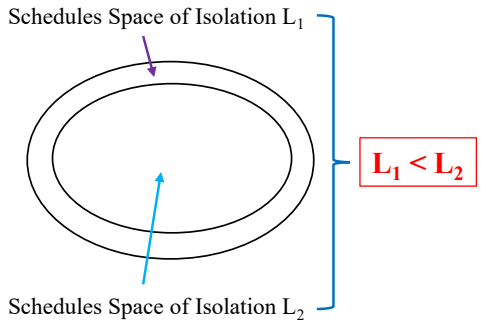
New Phenomena: Lost Update





- The final value of x contains only the increment of 30 added by T1.
- It is non-serializability schedule.
- It forbids P0, or P1, or P4 taht precludes P4.

Cursor Stability Isolation Level

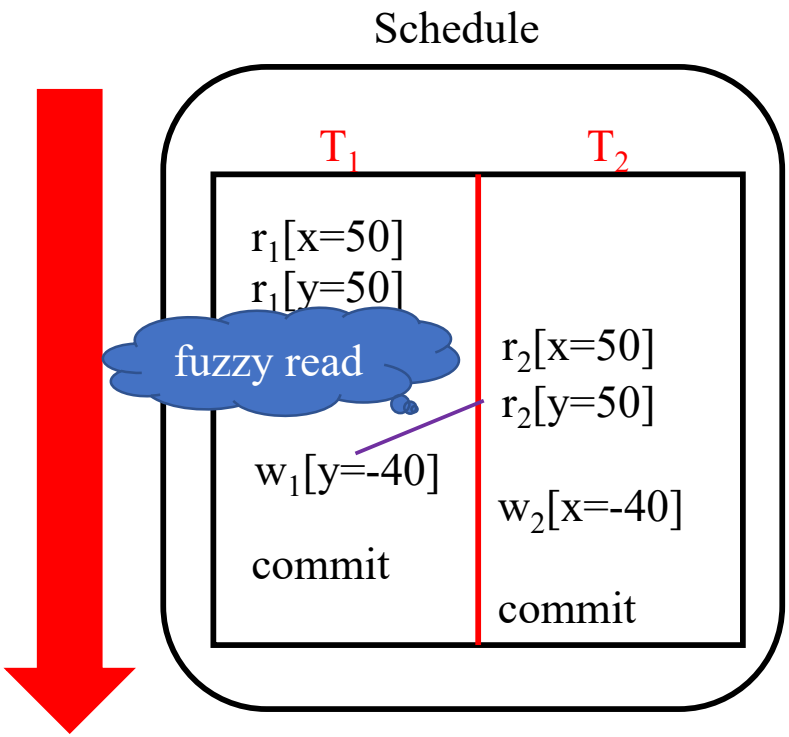


- Read Committed $<$ Cursor Stability $<$ Repeatable Read
- The Cursor Stability isolation level extends READ COMMITTED locking behavior for SQL cursors by adding a new read action for FETCH from a cursor and requiring that a lock be held on the current item of the cursor.
- The lock is held until the cursor moves or is closed, possibly by a commit.

$$P4 : R_1(x) \ldots W_2(x) \ldots W_1(x) \ldots c_1$$
$$P4C : RC_1(x) \ldots W_2(x) \ldots WC_1(x) \ldots c_1$$

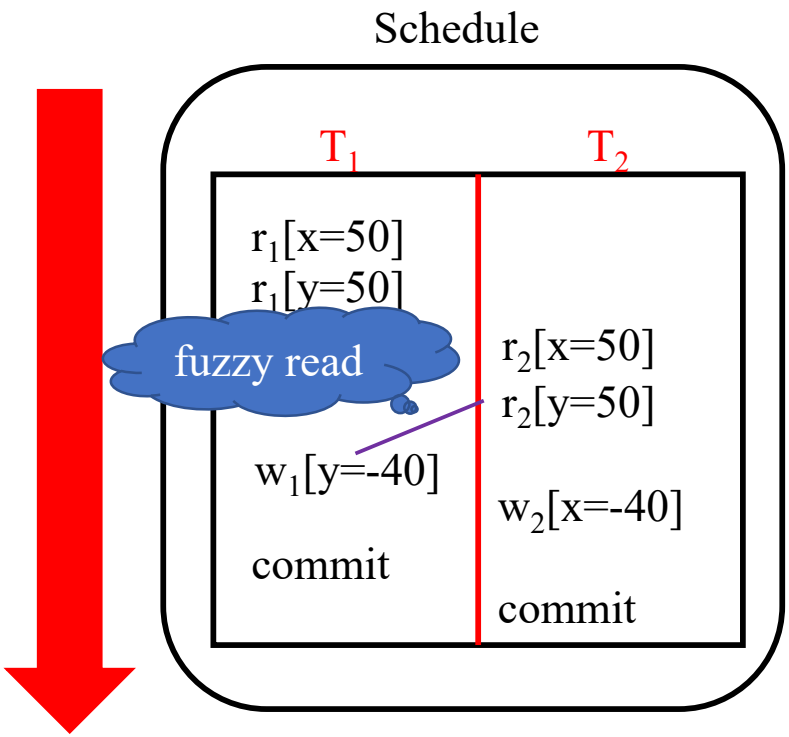
Table of Contents

Example of Schedule in Single Version



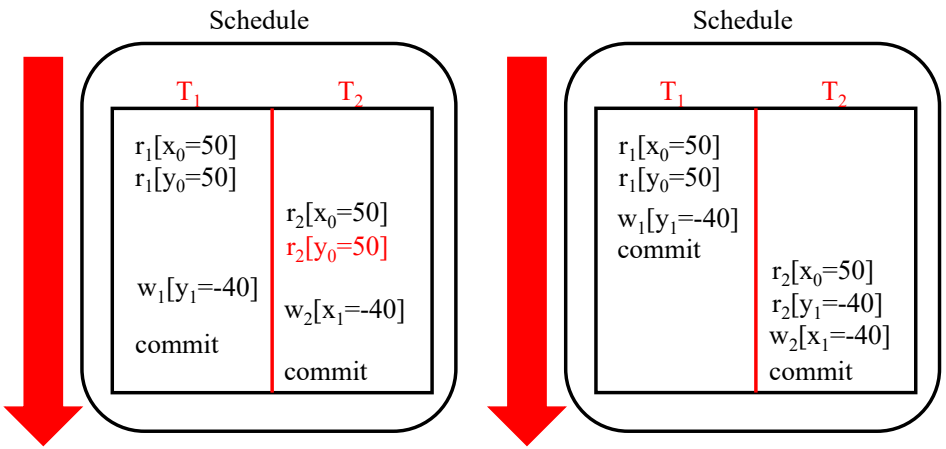
- It is non-serializability due to fuzzy read.

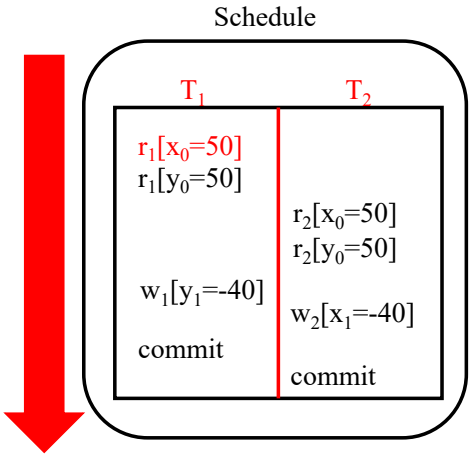
Example of Schedule in Single Version

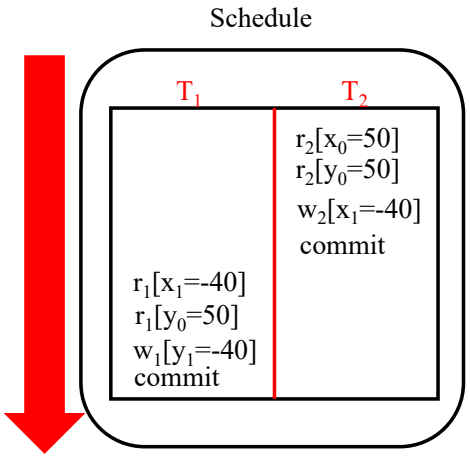


- It is non-serializability due to fuzzy read.
- It is not exist fuzzy read for multiple version systems.
  - ✓ Writer don't block reader.
  - ✓ Reader don't block writer.

Example of Schedule in Multiple Version

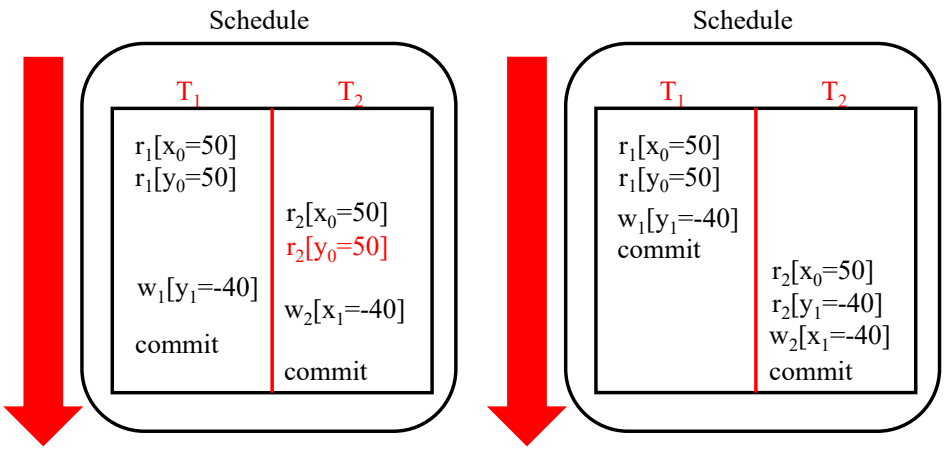


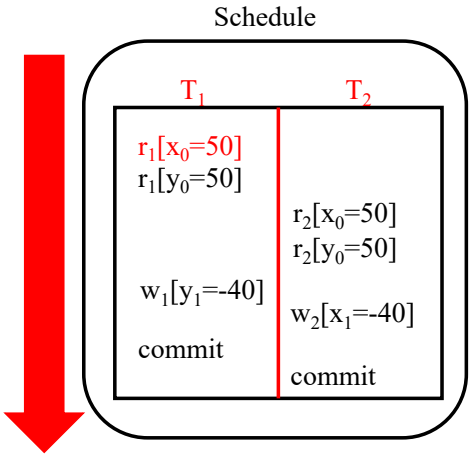


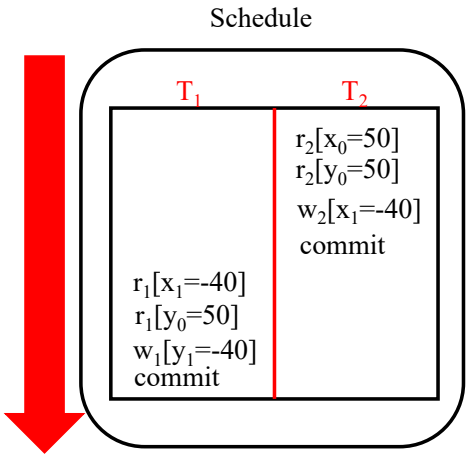


- The schedule is non-serializability even though hasn't fuzzy read in multiple version system.

Example of Schedule in Multiple Version

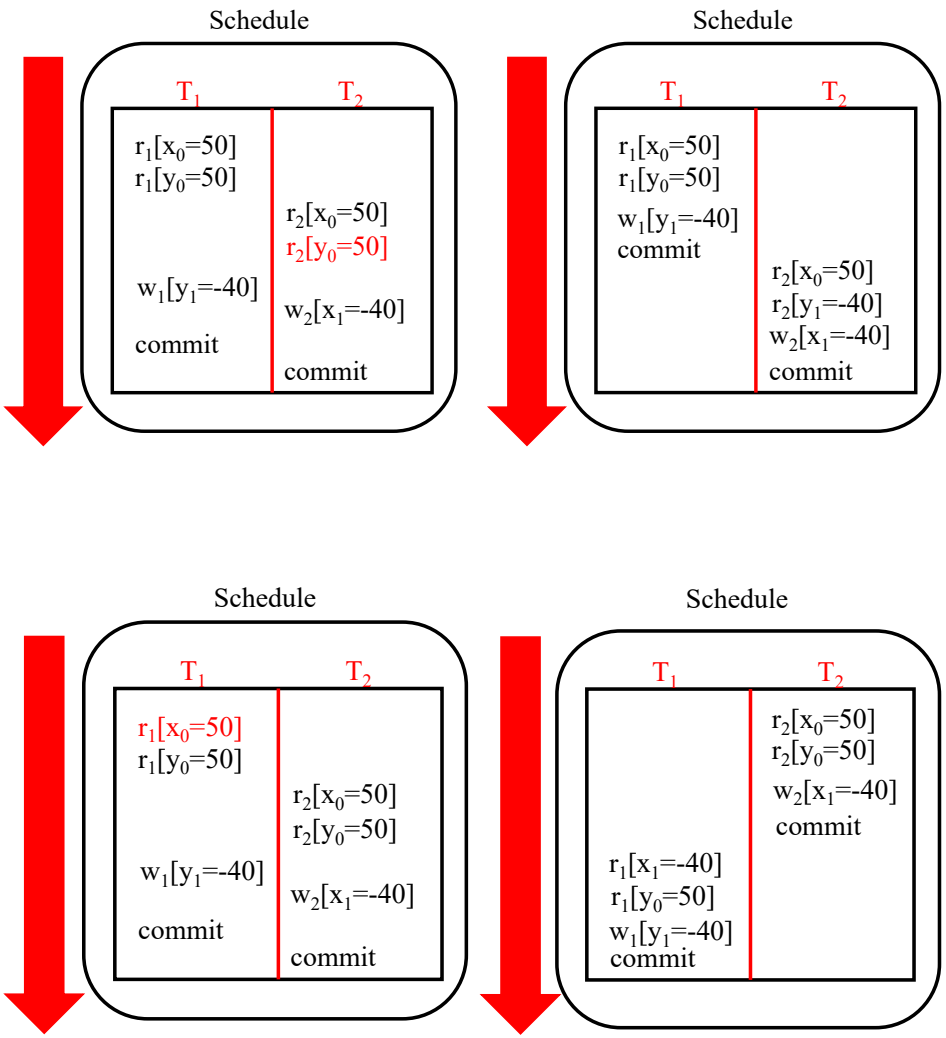






- The schedule is non-serializability even though hasn't fuzzy read in multiple version system.
- The schedule has the same inter-transactional dataflows as could occur under Snapshot Isolation (there is no choice of versions read by the transactions (e.g., $r_2[y_0 = 50] \rightarrow r_2[y_1 = -40]$ or $r_1[x_0 = 50] \rightarrow r_1[x_1 = -40]$)).

Example of Schedule in Multiple Version

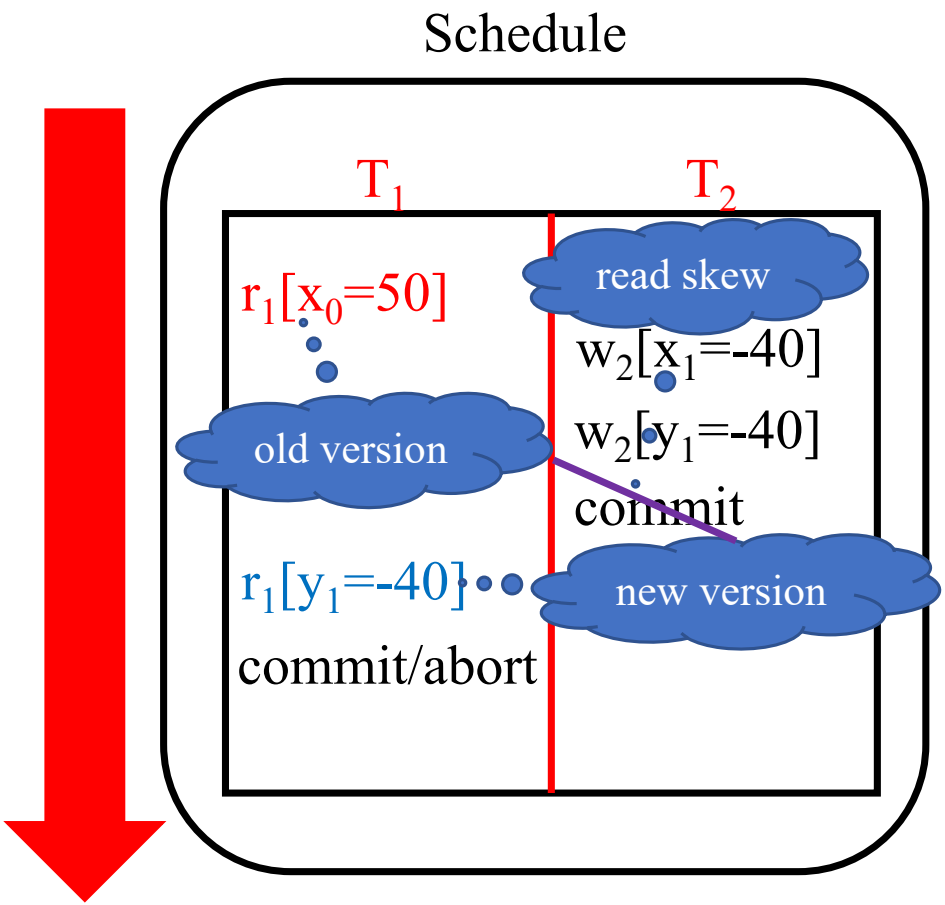


- The schedule is non-serializability even though hasn't fuzzy read in multiple version system.
- The schedule has the same inter-transactional dataflows as could occur under Snapshot Isolation (there is no choice of versions read by the transactions (e.g., $r_2[y_0 = 50] \rightarrow r_2[y_1 = -40]$ or $r_1[x_0 = 50] \rightarrow r_1[x_1 = -40]$)).
- Constraint violation (e.g., x = y in this example)

Constraint Violation

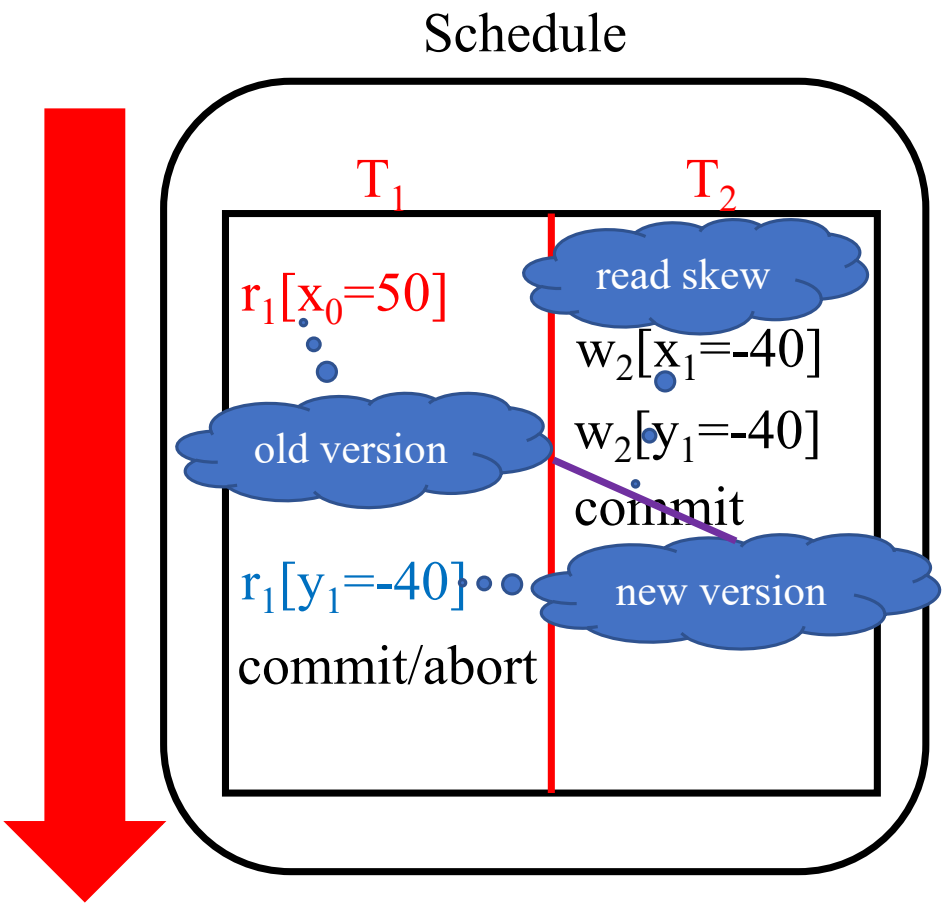
- Constraint violation is a generic and important type of concurrency anomaly.
- Individual databases satisfy constraints over multiple data items (e.g., uniqueness of keys, referential integrity, replication of rows in two tables, etc.).
- Together they form the database invariant constraint predicate, C(DB).
- Transactions must preserve the constraint predicate to maintain consistency
  - ✓ If the database is consistent when the transaction starts, the database will be consistent when the transaction commits.
  - ✓ If a transaction reads a database state that violates the constraint predicate, then the transaction suffers from a constraint violation concurrency anomaly.

New Phenomena: Read Skew (读交叉)



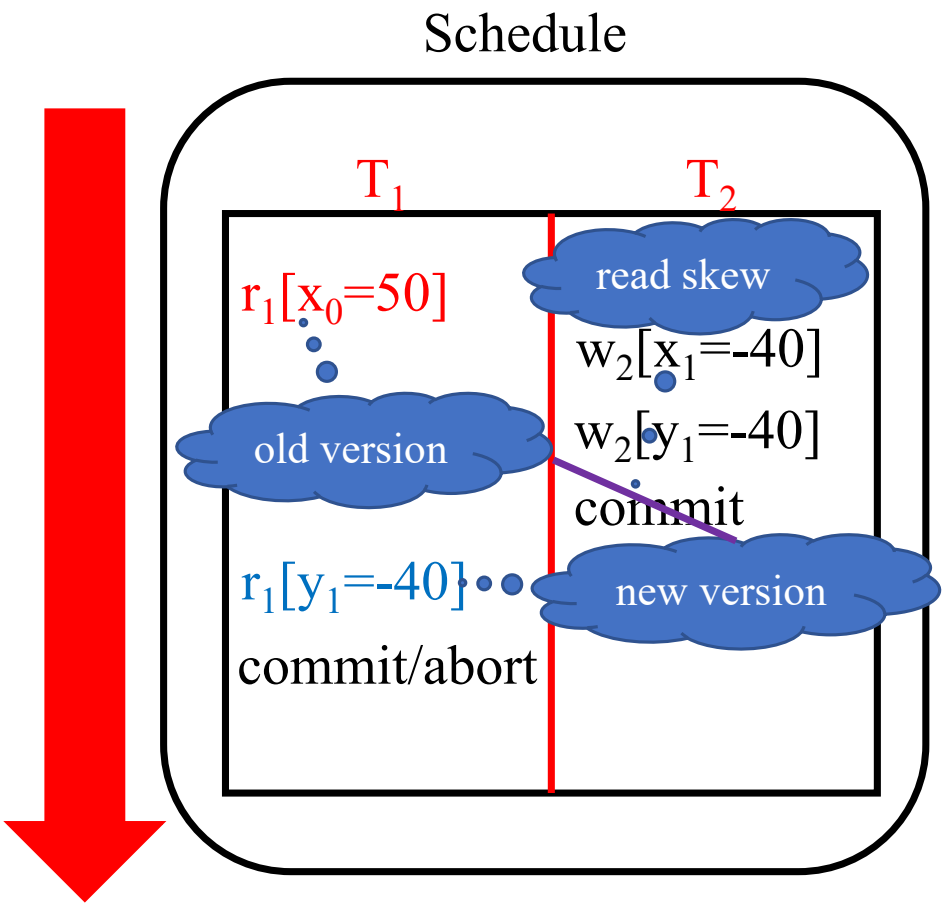
- Suppose transaction T1 reads x, and then a second transaction T2 updates x and y to new values and commits. If now T1 reads y, it may see an inconsistent state, and therefore produce an inconsistent state as output.

New Phenomena: Read Skew (读交叉)



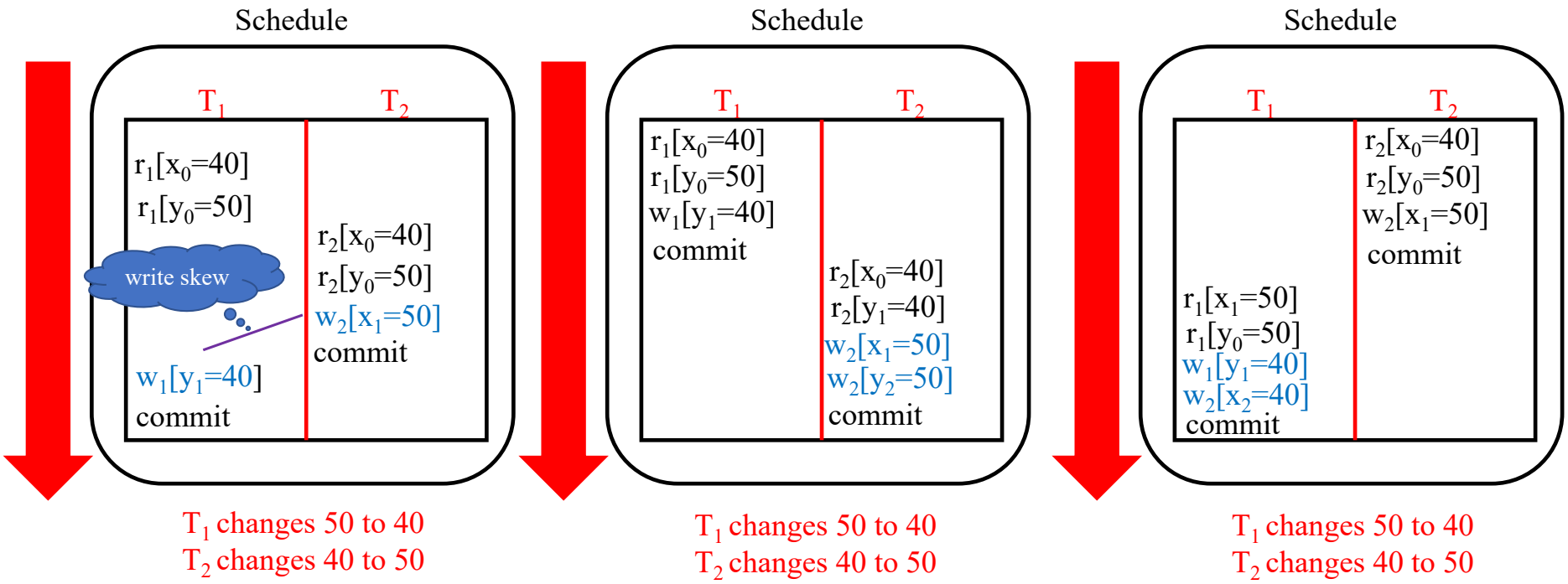
- Suppose transaction T1 reads x, and then a second transaction T2 updates x and y to new values and commits. If now T1 reads y, it may see an inconsistent state, and therefore produce an inconsistent state as output.
- The transaction read the old version for x and new version for y, which leads to constraint violation (e.g., x = y)).

New Phenomena: Read Skew (读交叉)



- Suppose transaction T1 reads x, and then a second transaction T2 updates x and y to new values and commits. If now T1 reads y, it may see an inconsistent state, and therefore produce an inconsistent state as output.
- The transaction read the old version for x and new version for y, which leads to constraint violation (e.g., x = y)).
- Fuzzy Reads (P2) is a degenerate form of Read Skew

New Phenomena: Write Skew (写交叉)



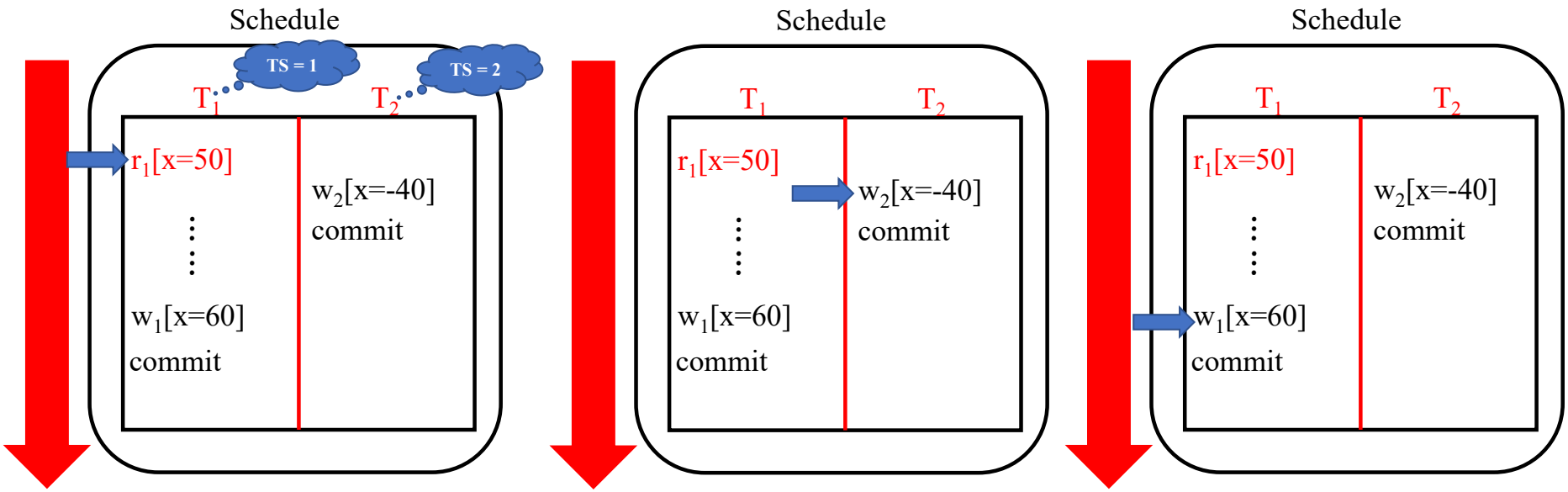
- Read skew leads to write skew.
- The schedule is non-serializability because it is not equivalent to some serializability schedule.

Formal Expression of Read/Write Skew Anomalies

Read Skew Constraint Violation :
$r_1(x_0)$ ... $w_2(x_1)$ ... $w_2(y_1)$ ... $c2$ ... $r_1(y_1)$ ... $c1$
Write Skew Constraint Violation :
$r_1(x_0)$ ... $r_1(y_0)$ ... $r_2(x_0)$ ... $r_2(y_0)$... $w_2(x_1)$ ... $c2$ ... $w_1(y_1)$ ... $c1$

- Read Skew is possible under READ COMMITTED, but not under the Snapshot Isolation timestamp mechanism.
- Write Skew obviously can occur in a Snapshot Isolation history.
- REPEATABLE READ $><$ Snapshot Isolation
- First-committer-wins requires the system to remember all updates (write locks) belonging to any transaction that commits after the Start-Timestamp of each active transaction. It aborts the transaction if its updates conflict with remembered updates by others.

Example of First-Committer-Wins



x_0	50	0	-

x_0	50	0	2
x_1	-40	2	-

x_0	50	0	2
x_1	-40	2	-
x_2	60	1	abort

Modifing x_2 conflicts T_2

Conclusions of Snapshot Isolation Levels

- T_2 commits firstly, and T_1 aborts.
- Snapshot isolation is not friendly to long-duration transactions.
- Certainly in cases where short update transactions conflict minimally and long-running transactions are likely to be read only, Snapshot Isolation should give good results.
- In regimes where there is high contention among transactions of comparable length, Snapshot Isolation offers a classical optimistic approach.

Table of Contents

Anomaly-based Isolation Levels

Table 4. Isolation Types Characterized by Possible Anomalies Allowed.

Isolation level	**P0** Dirty Write	**P1** Dirty Read	**P4C** Cursor Lost Update	**P4** Lost Update	**P2** Fuzzy Read	**P3** Phantom	**A5A** Read Skew	**A5B** Write Skew
READ UNCOMMITTED == Degree 1	Not Possible	Possible	Possible	Possible	Possible	Possible	Possible	Possible
READ COMMITTED == Degree 2	Not Possible	Not Possible	Possible	Possible	Possible	Possible	Possible	Possible
Cursor Stability	Not Possible	Not Possible	Not Possible	Sometimes Possible	Sometimes Possible	Possible	Possible	Sometimes Possible
REPEATABLE READ	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Possible	Not Possible	Not Possible
Snapshot	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Sometimes Possible	Not Possible	Possible
ANSI SQL SERIALIZABLE == Degree 3 == Repeatable Read Date, IBM, Tandem, ...	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible

- No P2, no A5A.
- No P0 for any isolation levels.
- Serializability has not any anomalies, etc.

Table of Contents

References I


Berenson, Hal, et al. "A critique of ANSI SQL isolation levels." ACM SIGMOD Record 24.2 (1995): 1-10.

Table of Contents

Thank you!
Welcome for any questions!

Ethan Zhu
Department of Computer Science and Technology
Nanjing University