Data Management Assignment 6

Cuihuan Zhang

May 20, 2025

---

1. (a)

Database items	Operation
T1	Not affected by the failure
T2	Not affected by the failure
T3	Undo
T4	Redo
T5	Undo
T6	Undo

   i. T4 needs to be Redo, since T4 is committed after the checkpoint.
   ii. Obviously, T3, T5 and T6 are uncommitted items, so we need to Undo them.
   iii. T1 and T2 are not affected by the failure since T1 and T2 are committed before the checkpoints. The modifications of T1 and T2 have been persisted to the disk through checkpoints.

   (b) i.

Database items	Possiable values
a	1 or 2
b	1 or 2 or 3
c	1

   ii.

Database items	Possiable values
a	2
b	2
c	0

2. (a) i. Recoverable: In a scheduling, if transaction T' reads the modified data of another transaction T, then the commit of transaction T' must occur follow the commit of transaction T.
   ii. Cascadeless (no cascading aborts): If transaction T' reads only values produced by transaction T that have already committed.

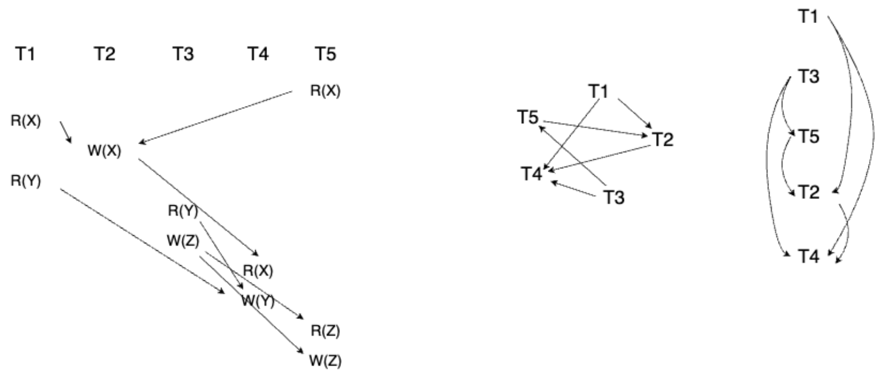
iii. Strict: if 1. satisfies the condition for avoiding cascading aborts and 2. for every transaction, if it writes an item, all the transactions that previously wrote that item have already committed or aborted.

(b) i. Recoverable: This schedule is recoverable. T5 reads the value z after the commit of T3. Although T2 haven't committed x, but T4 also haven't committed yet.
Cascadeless (no cascading aborts): T4 violated cascadeless. T4 reads the value x that have not already committed by T2.
Strict: T4 violates strictness. Since it is not satisfies the condition for avoiding cascading aborts.

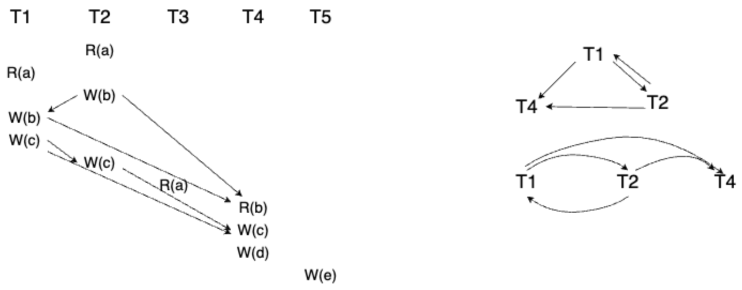
ii. Recoverable: This schedule is not recoverable. T4-read(b) have been committed before T1-write(b), so T4 violated recoverable.
Cascadeless (no cascading aborts): T4 violated cascadeless. T4-read(b) reads b that have not committed by T1-write(b).
Strict: T1, T2 and T4 violates strictness. T1 writes b before T2 committed write(b) (T1 write b overwrites the uncommitted b in T2) and same problem for c. And it is obvious that T4 does not satisfied the condition for avoiding cascading aborts.

3. (a) The schedule is serializable (no cycle), the serial equivalent can be [T1, T3, T5, T2, T4] or [T3, T5, T1, T2, T4] or [T3, T1, T5, T2, T4].

(b) The schedule is not serializable, in any order of [T1, T2, T4, T3, T5], [T2, T1, T4, T3, T5] etc. Always a conflict between [T1, T2] since it is a cycle.
Note: conflict graphs and examples are on the next page.



Serializable schedule since there is no cycle.
Serial equivalent can be represented as [T1, T3, T5, T2, T4].



The schedule is not serializable since there is a cycle between [T1, T2]. In whatever the order we have between [T1, T2], there is a conflict.

Figure 1: Serializability of concurrent transactions