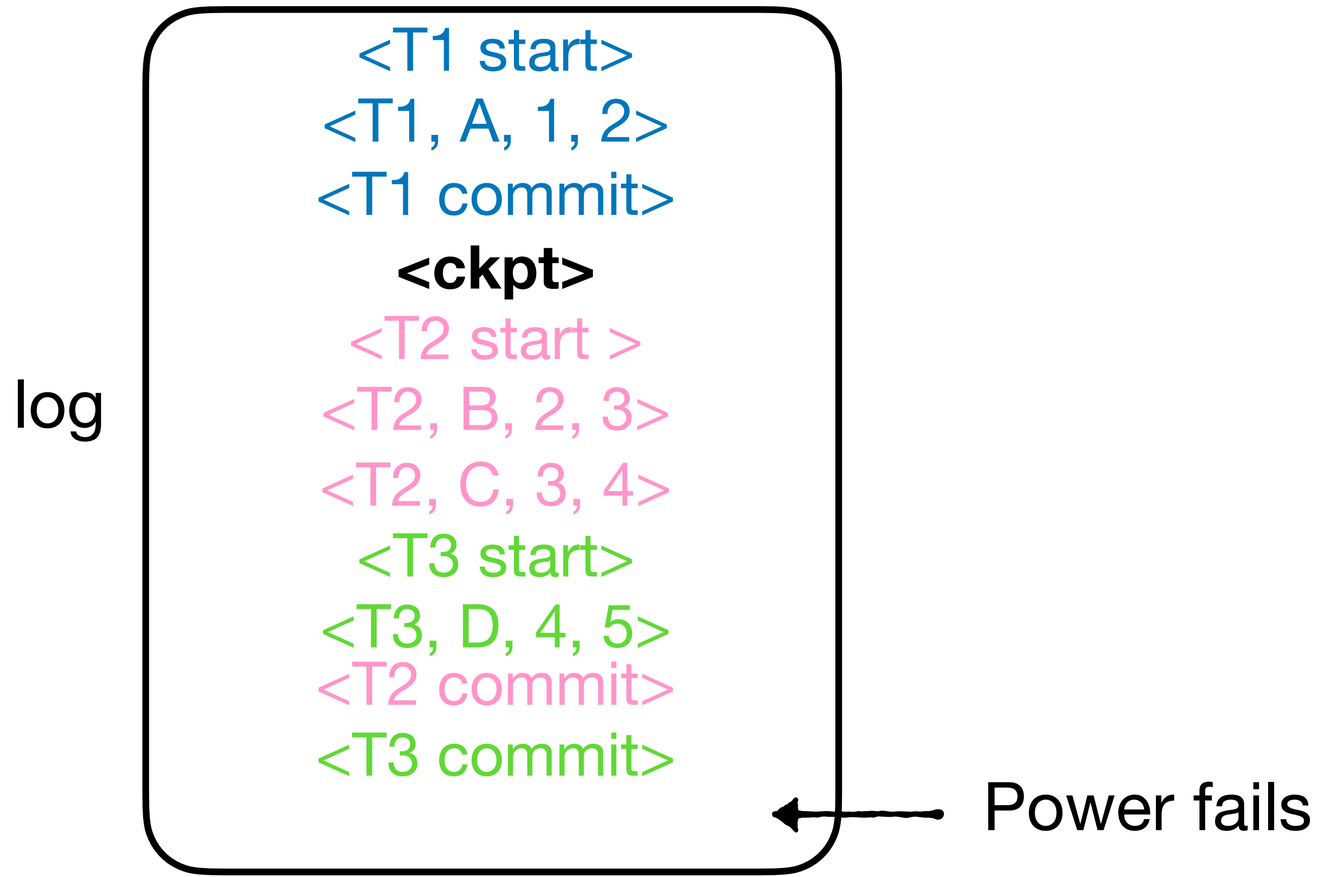
Recovery Tutorial

CSC443H1 Database System Technology

Niv Dayan

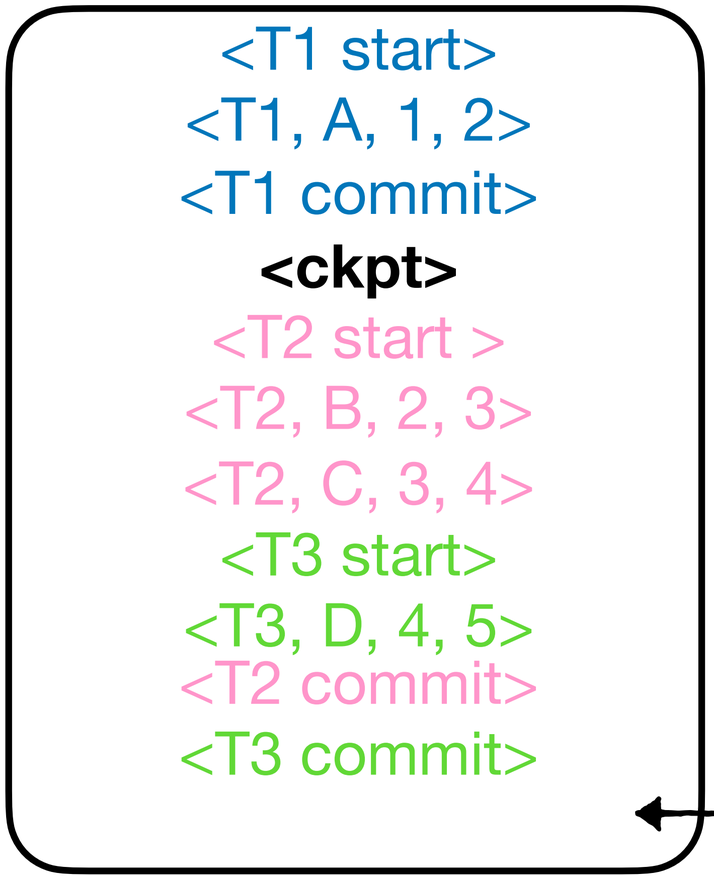
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



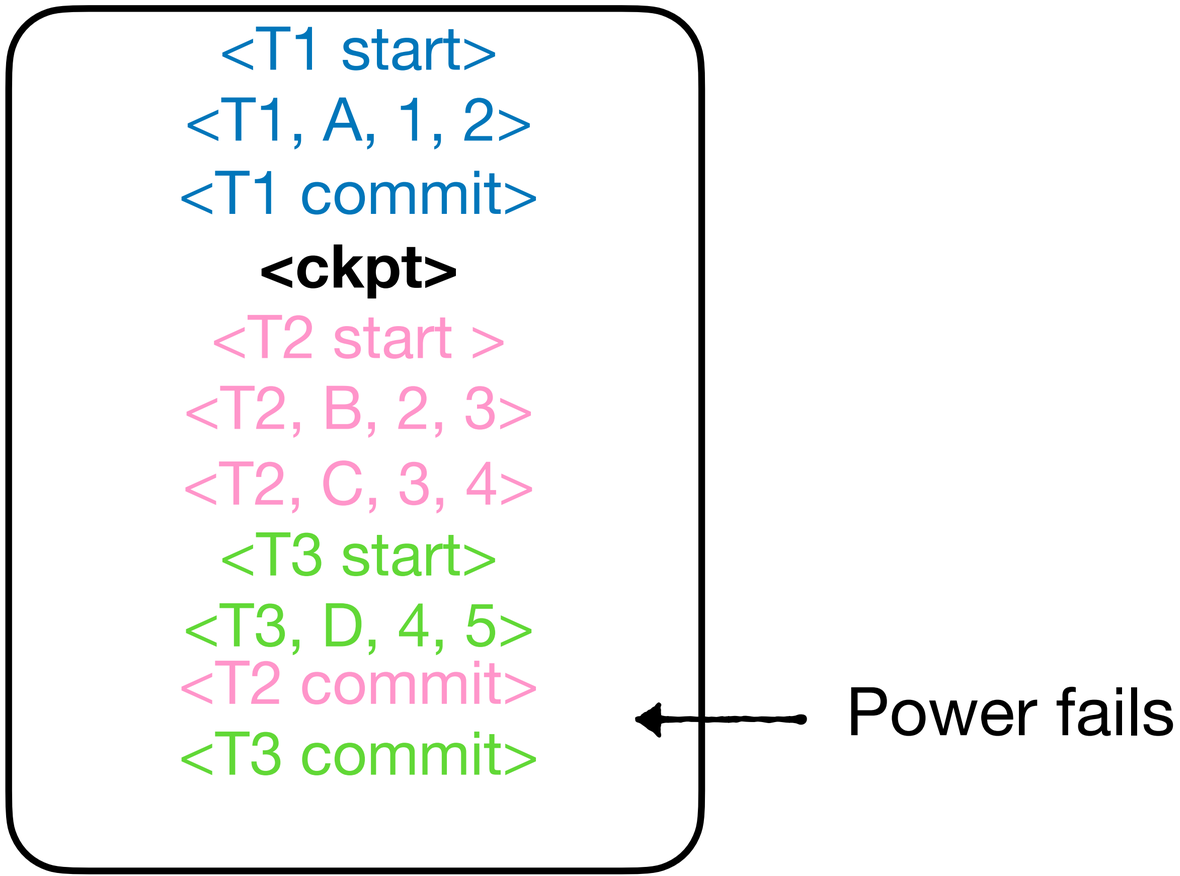
Nothing to undo

Redo T2 and T3 since checkpoint

Power fails

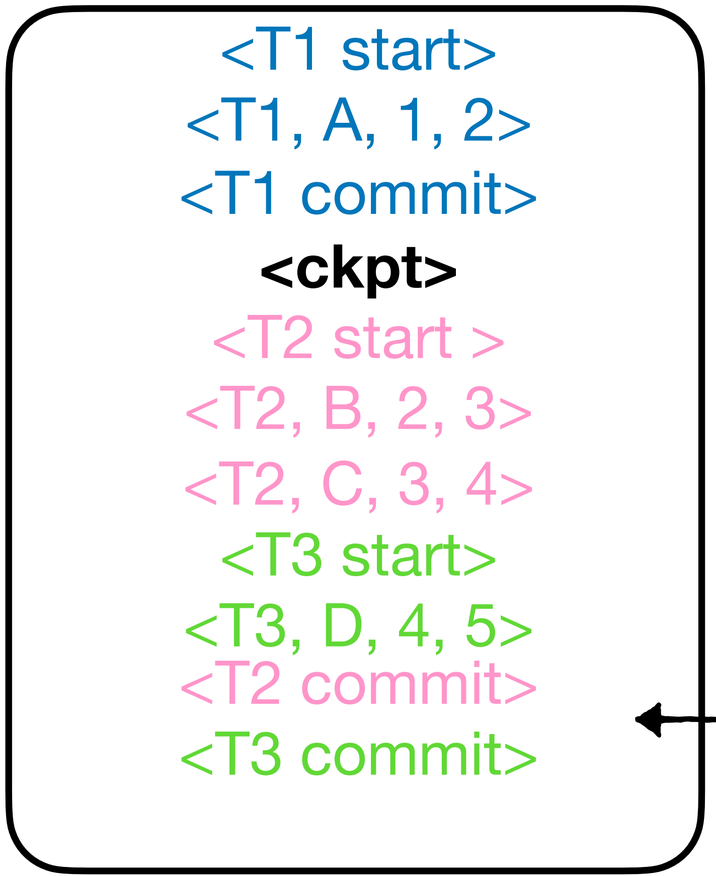
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



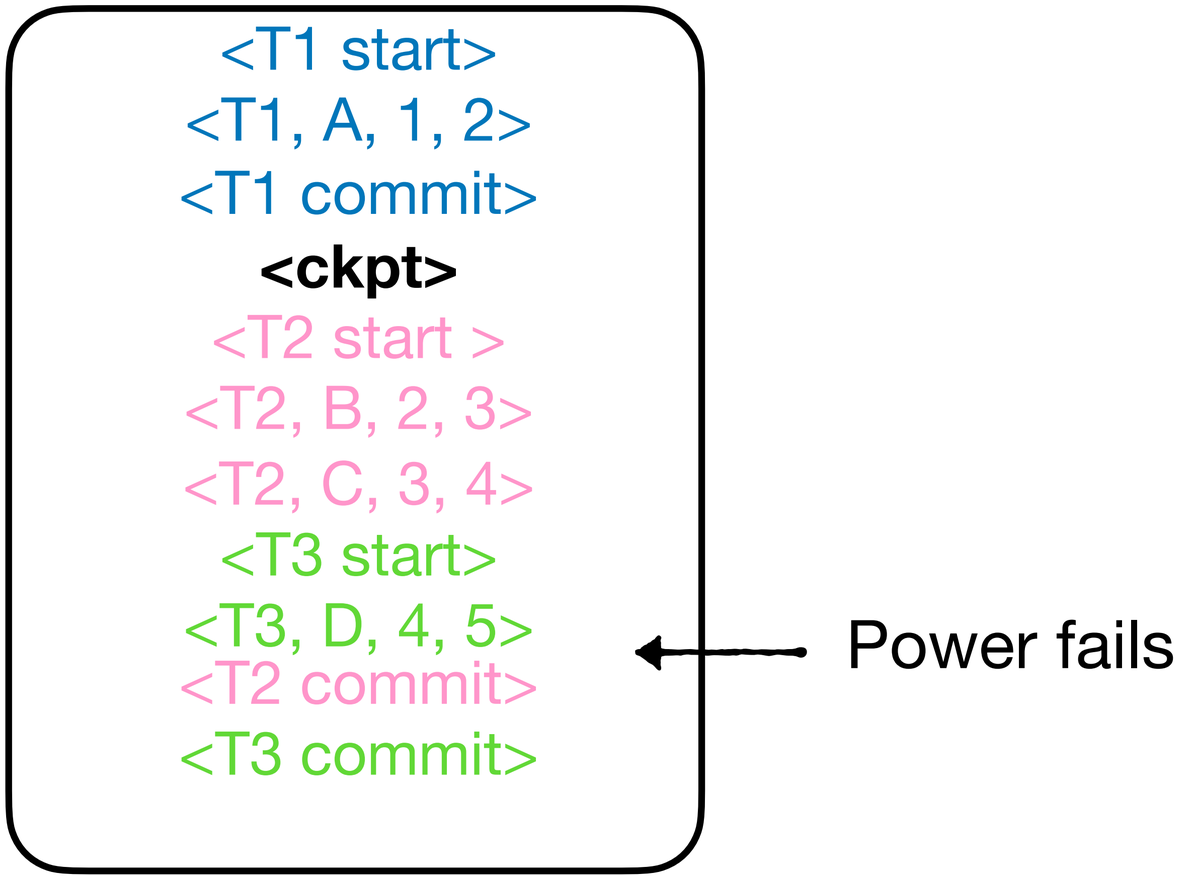
Undo T3

Redo T2

Power fails

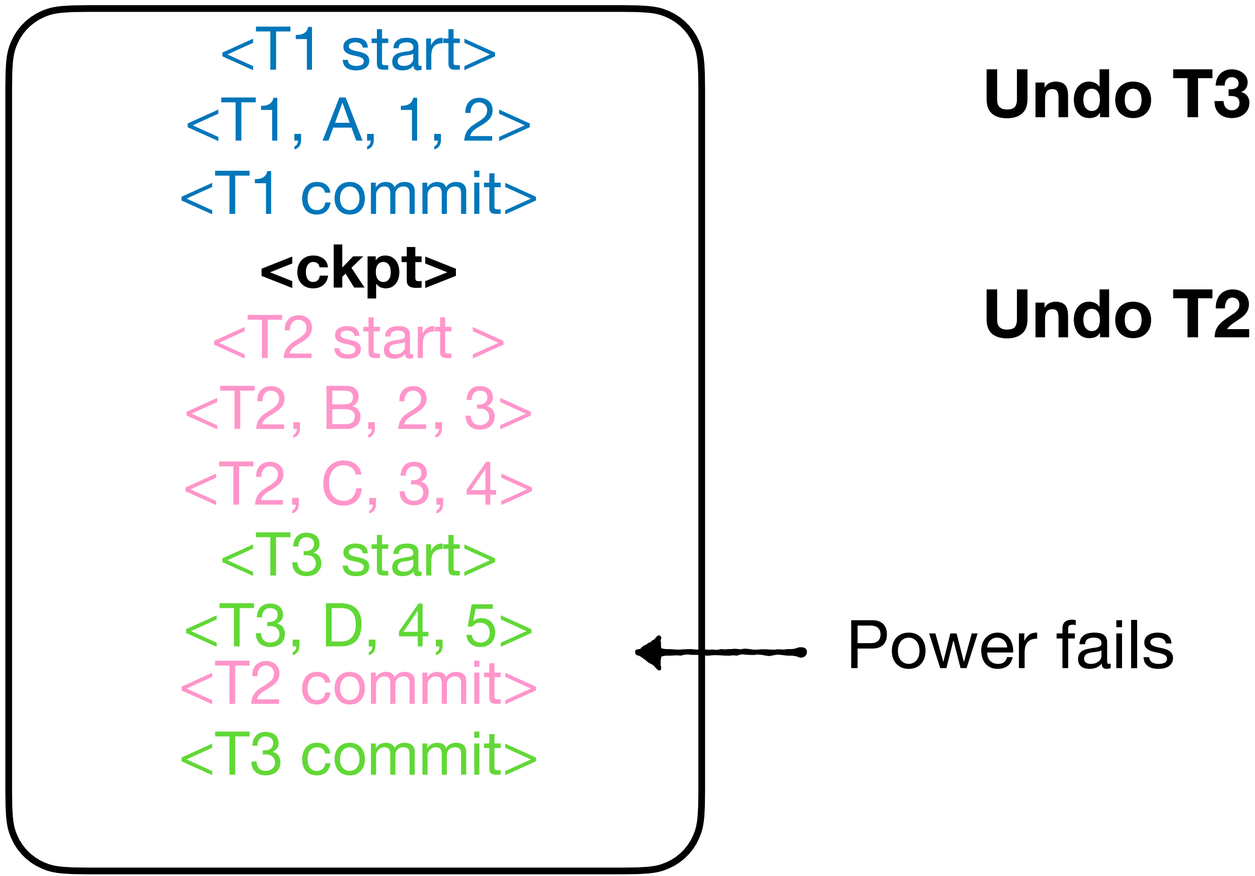
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



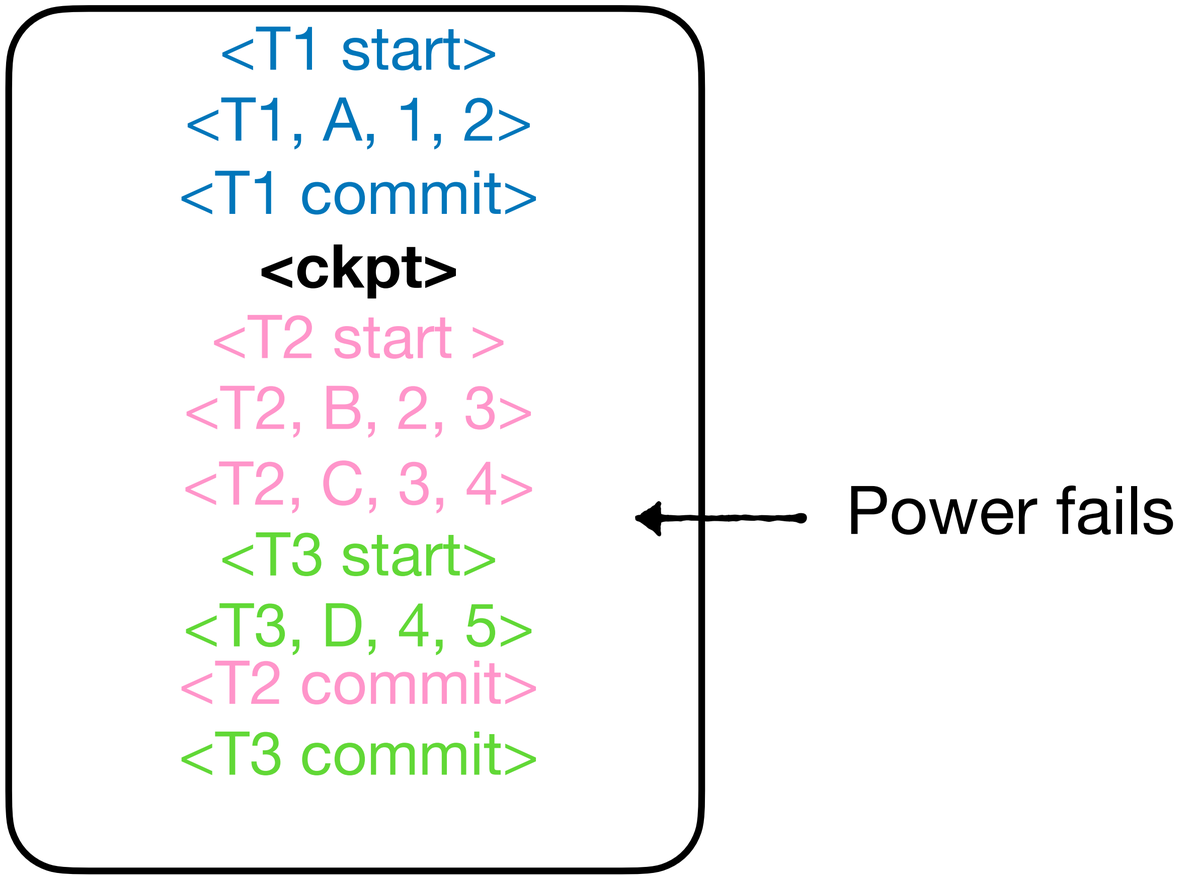
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



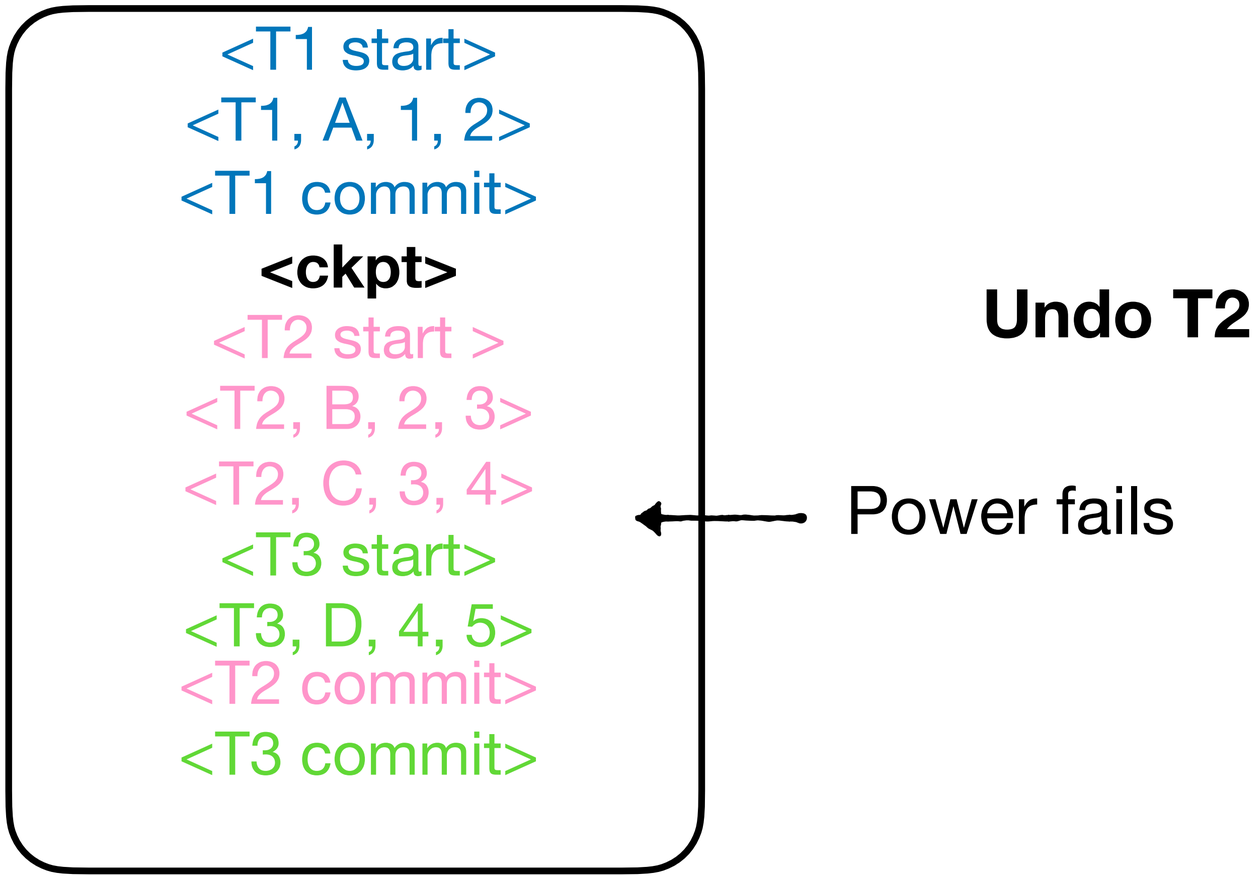
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



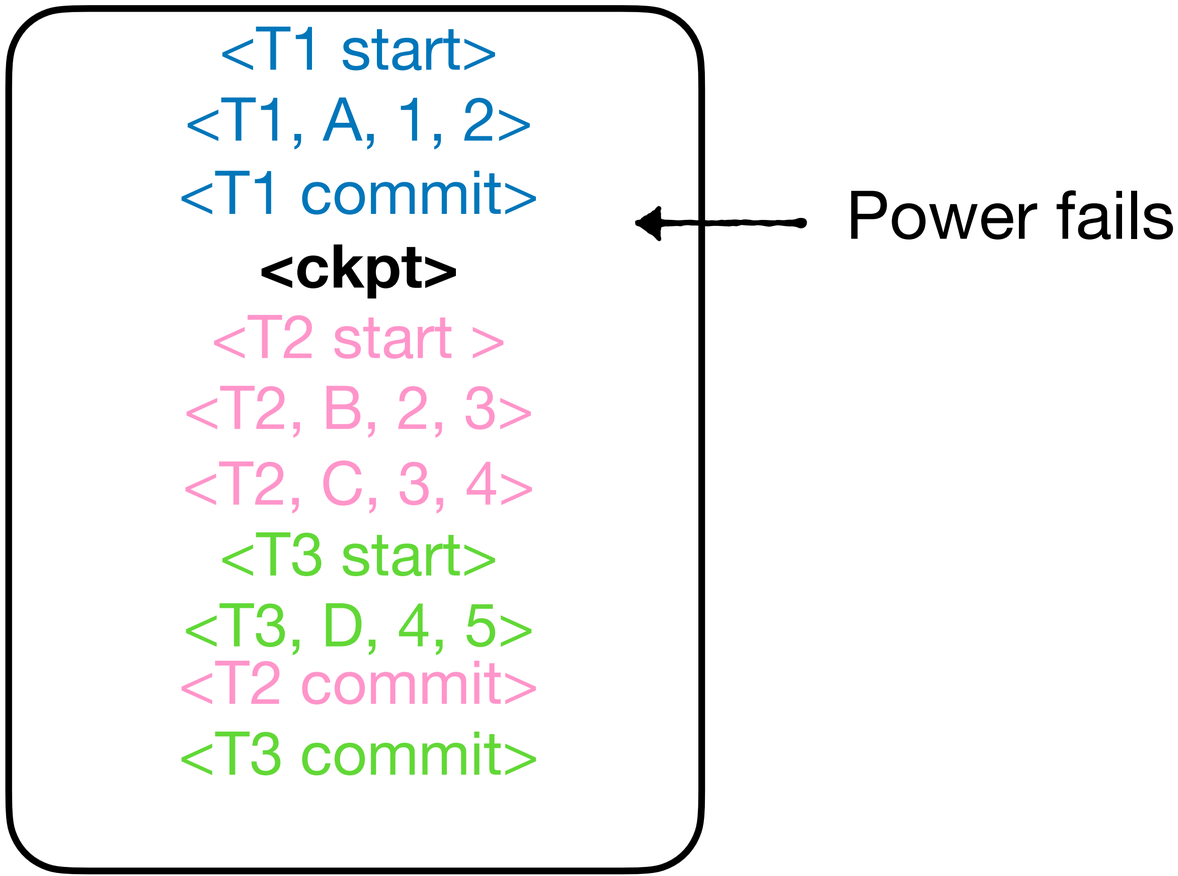
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



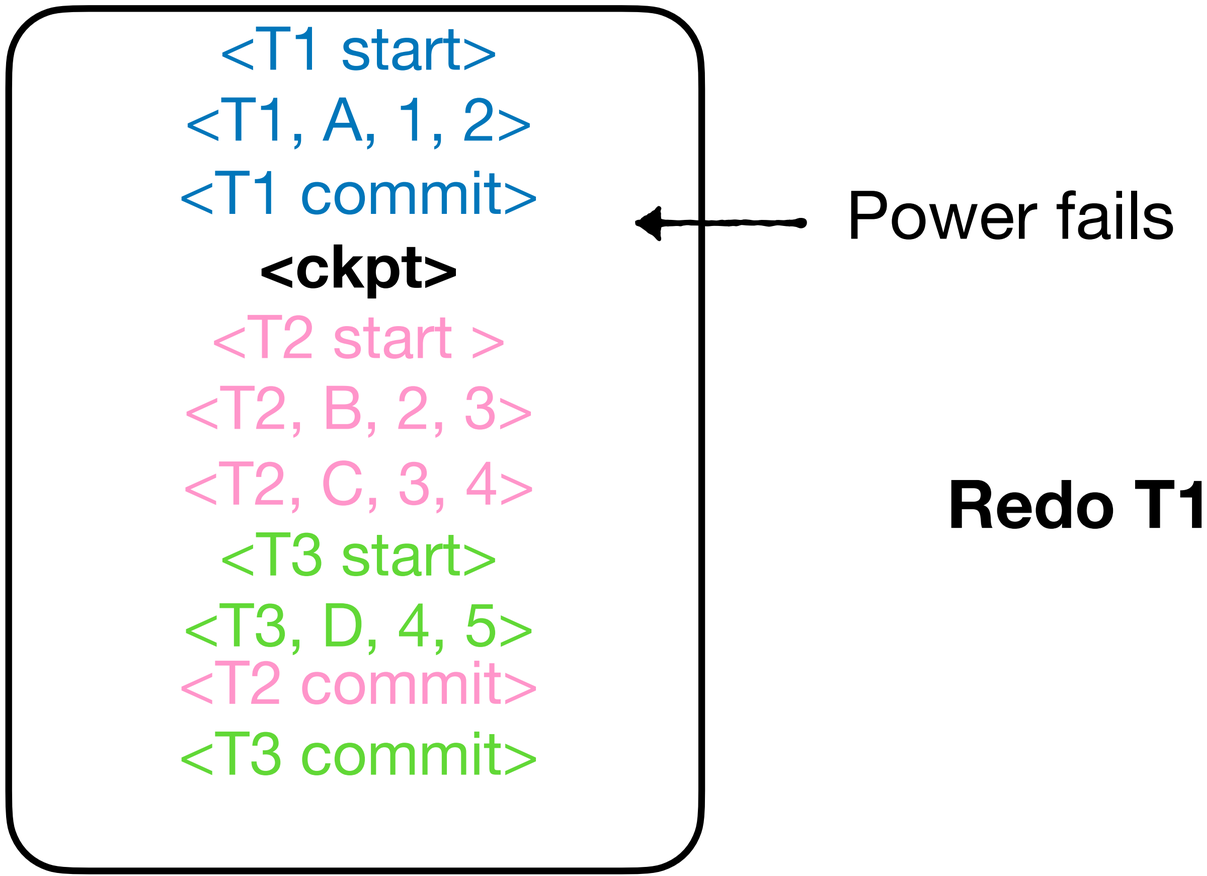
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



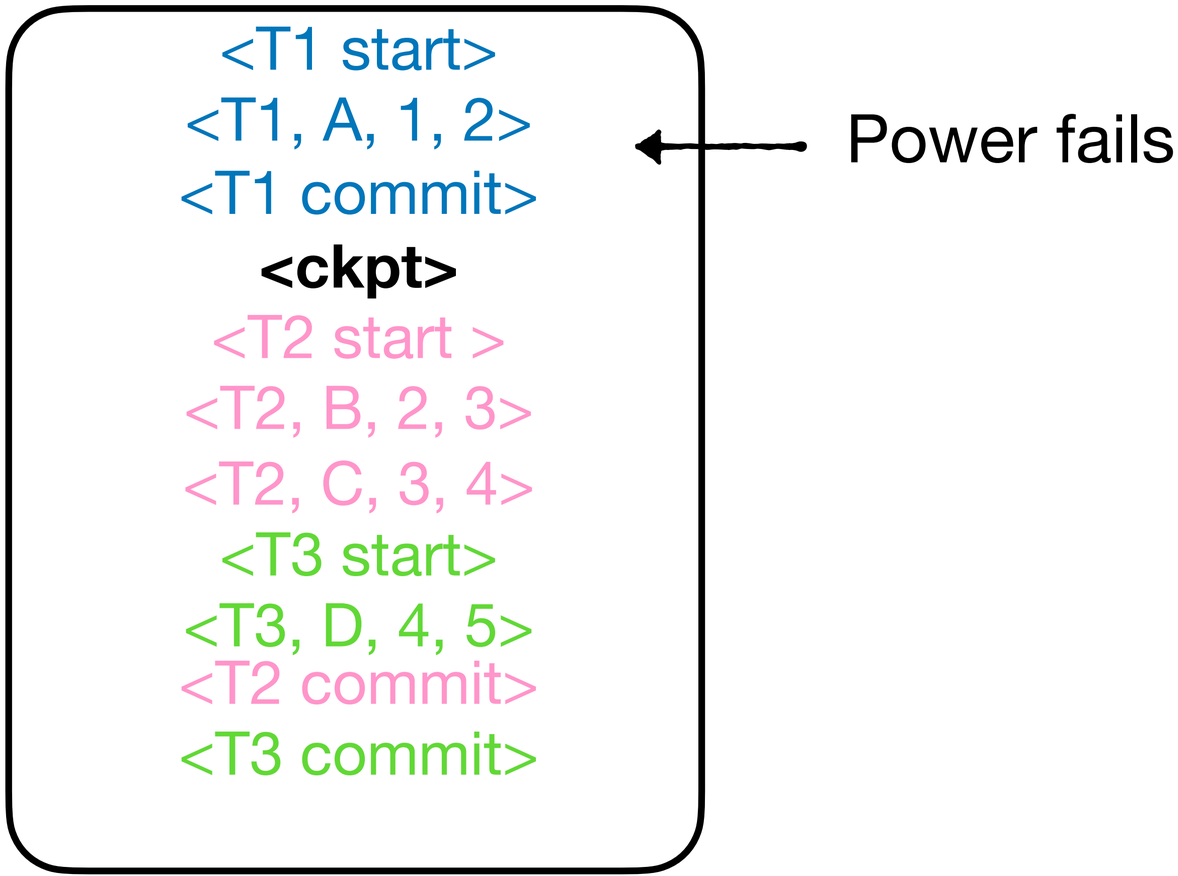
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



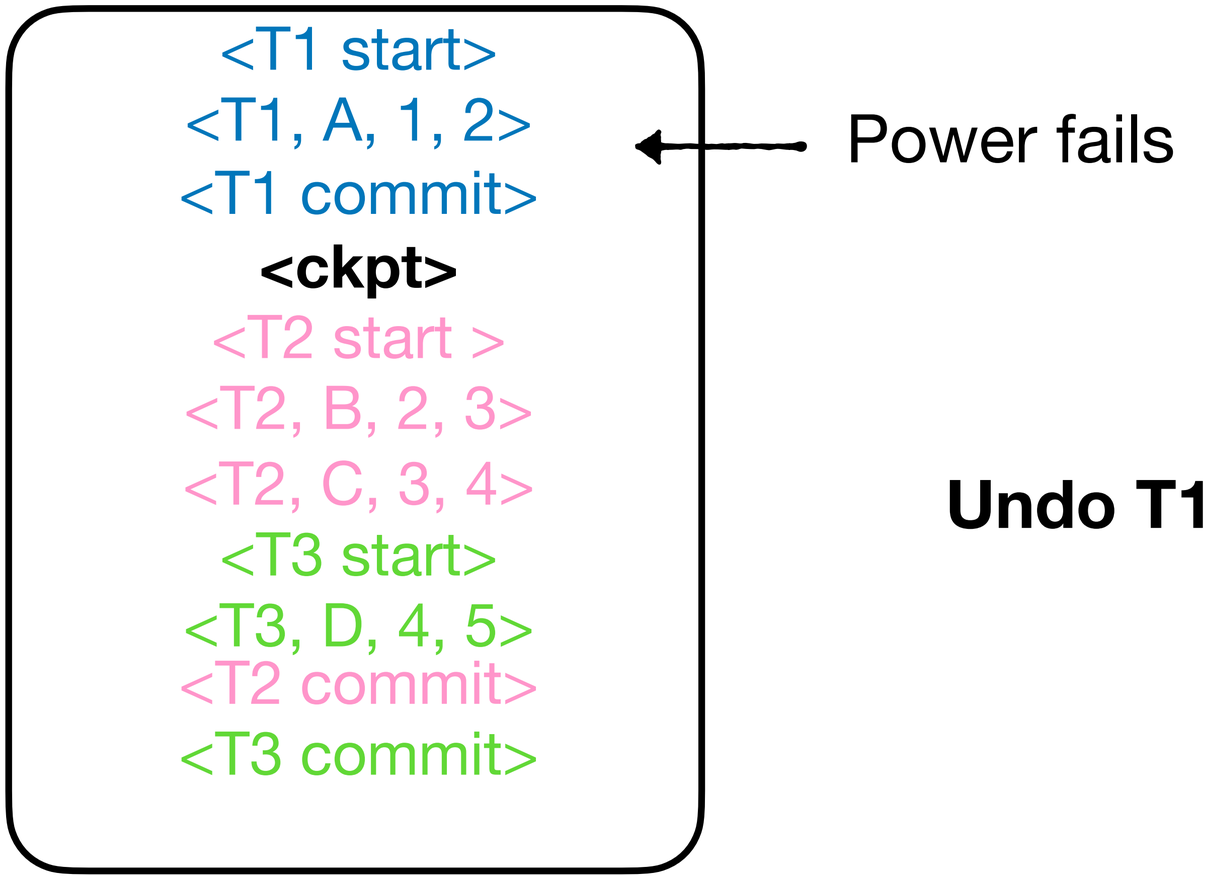
Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?



Question 1

Consider the following log, and note how records for different transactions are interleaved. Suppose power fails at different points. What would you need undo or redo?

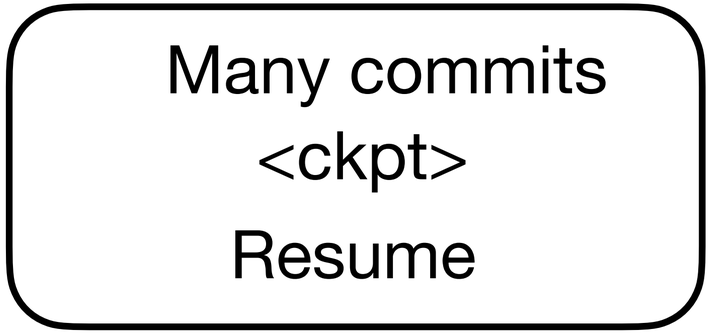


Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

Buffer pool

Database



log

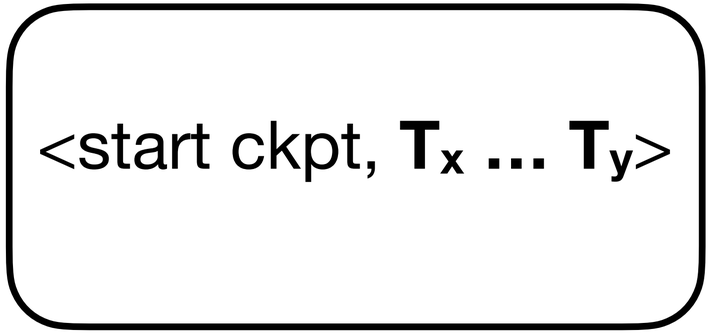
Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

Buffer pool

Add all ongoing transaction names in checkpoint start record

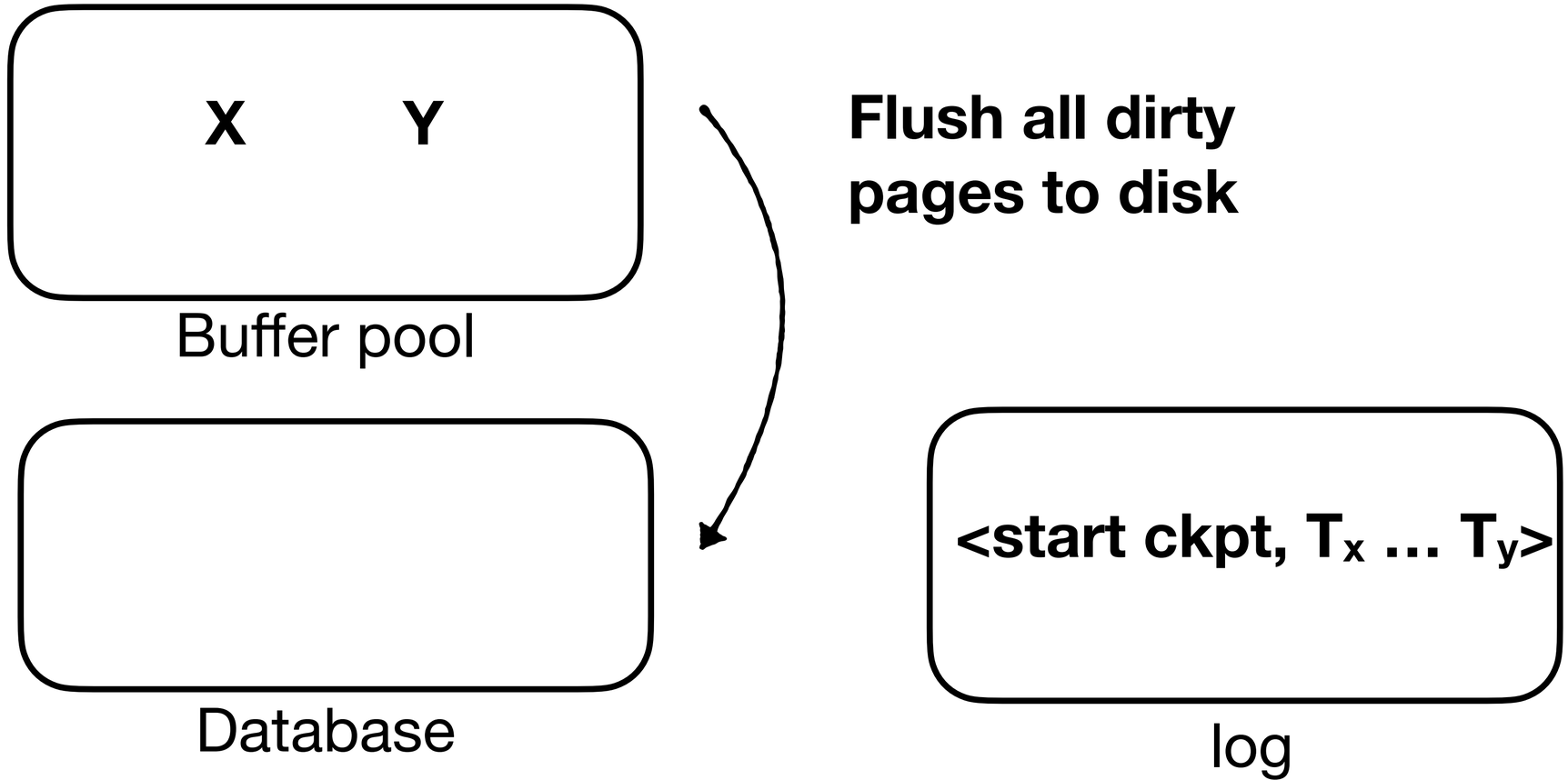
Database



log

Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

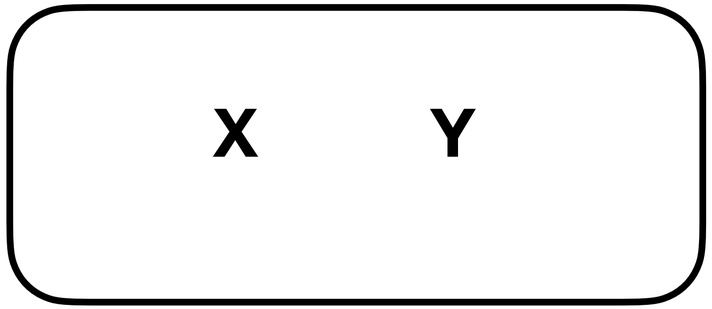


Question 2

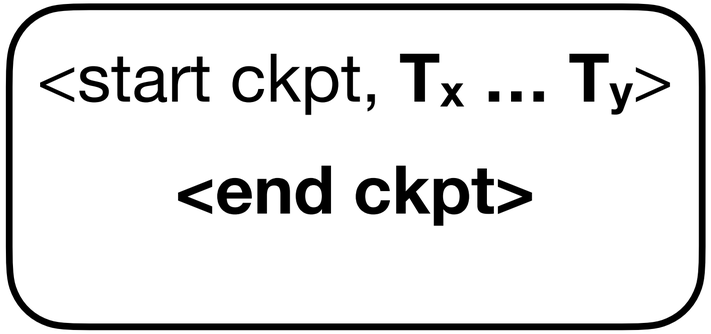
The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

Buffer pool

Flush end checkpoint record



Database



log

Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

how to recover?

log

<T1 start>
<T1, A, 1, 2>
<T2 start >
<T1 commit>
<T2, B, 2, 3>
<ckpt start T2>
<T2, C, 3, 4>
<T3 start>
<T3, D, 4, 5>
<ckpt end>
<T2 commit>
<T3 commit>

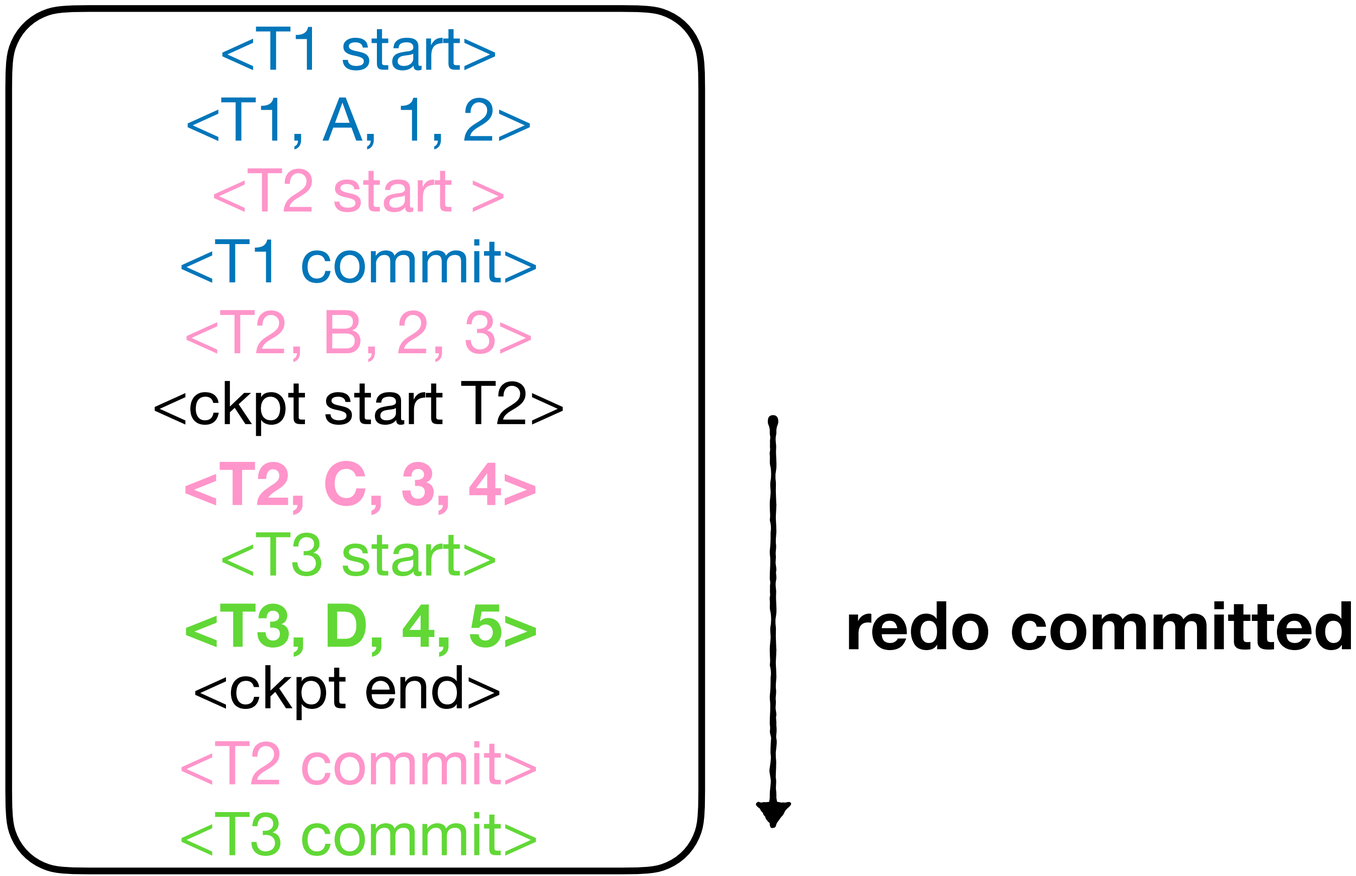
Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

Find checkpoint start & identify committed & uncommitted transactions

Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?



Question 2

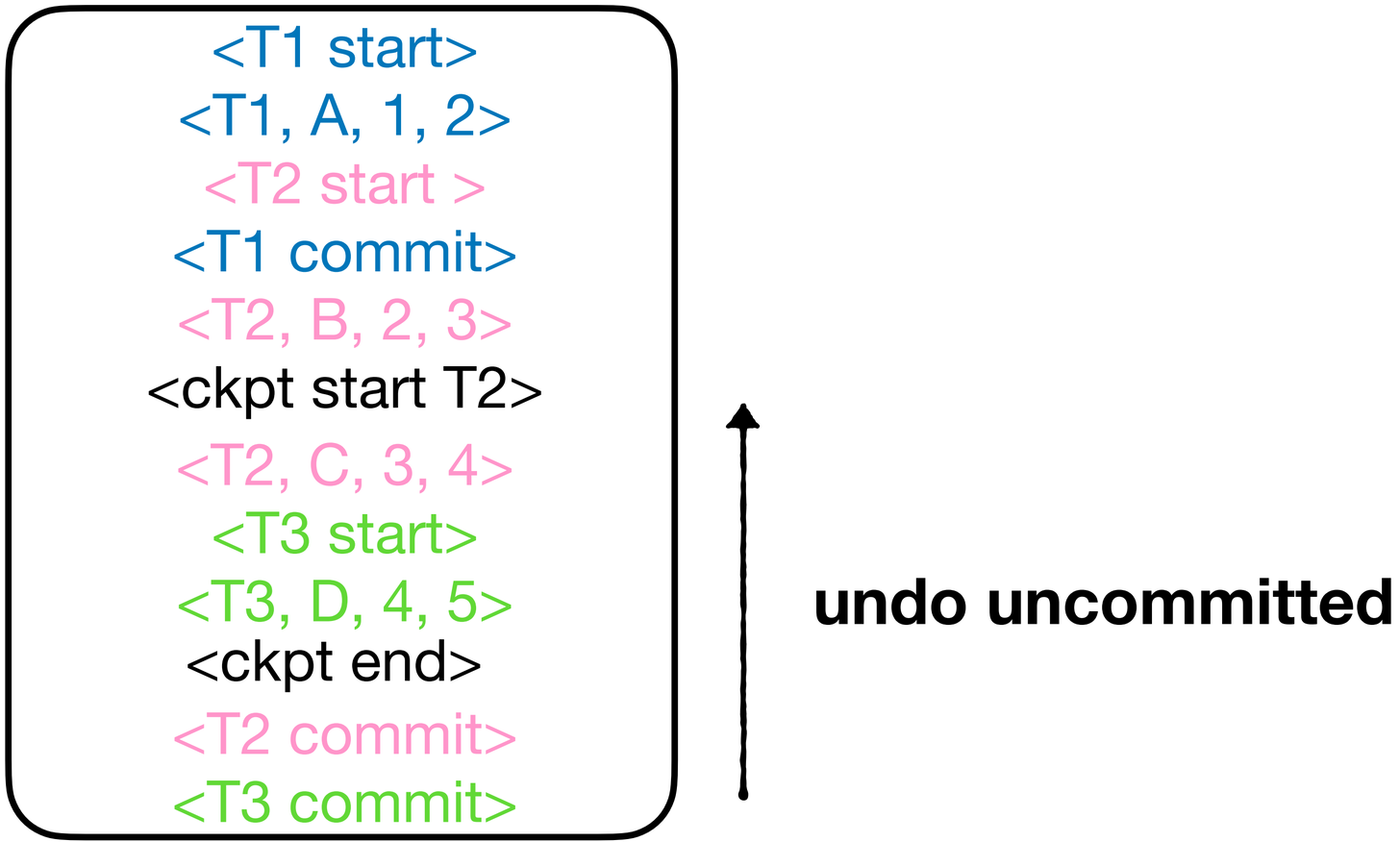
The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?

redo committed

Go from back to front so we keep latest copy of each value

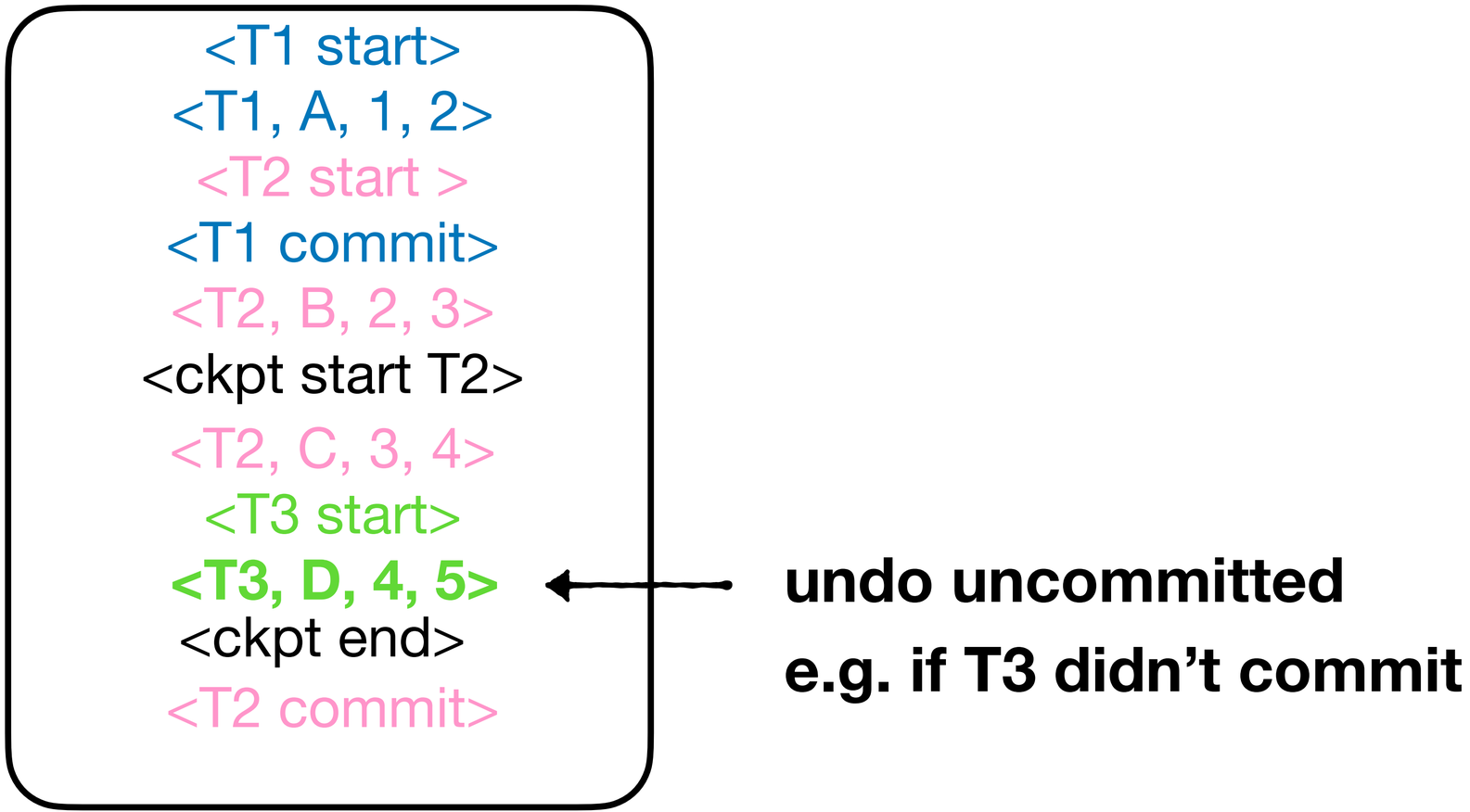
Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?



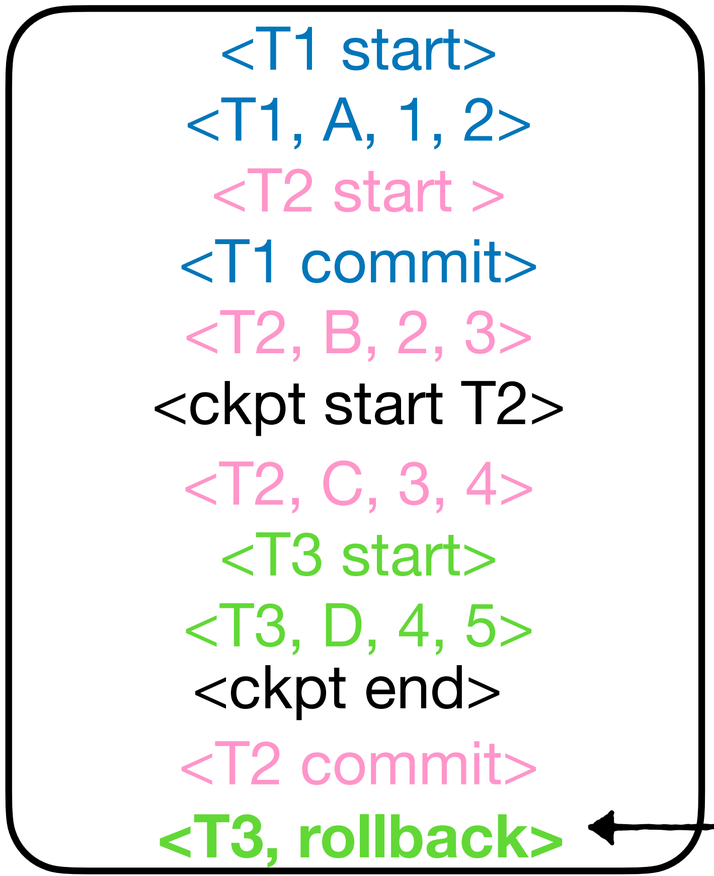
Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?



Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?



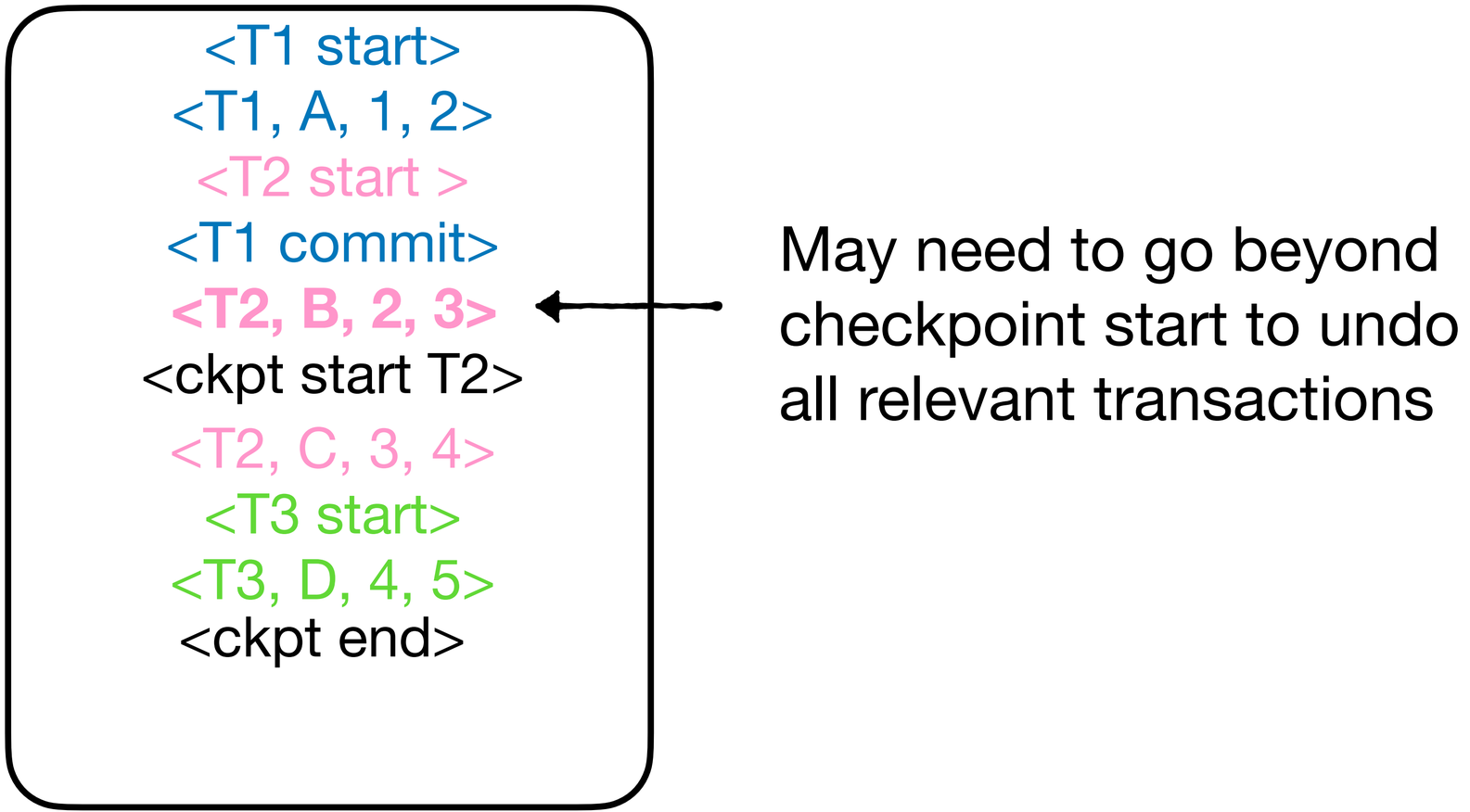
undo uncommitted

e.g. if T3 didn't commit

Add rollback

Question 2

The checkpointing mechanism we have seen requires all current transactions to commit, but this could lead to rejecting user transactions for a long while. How would you design a checkpoint mechanism that does not require all existing transactions to finish?



Question 3

Consider undo/redo logging. Flushing the log before can be a random I/O bottleneck. What can we do to address this?

Question 3

Consider undo/redo logging. Flushing the log before can be a random I/O bottleneck. What can we do to address this?

Put log on a separate disk

It only entails sequential access so this is ideal. Can also use SSD.

Question 3

Consider undo/redo logging. Flushing the log before can be a random I/O bottleneck. What can we do to address this?

Put log on a separate disk

It only entails sequential access so this is ideal. Can also use SSD.

Put DB on one or more SSDs or disks, maybe using RAID