Database Management Systems

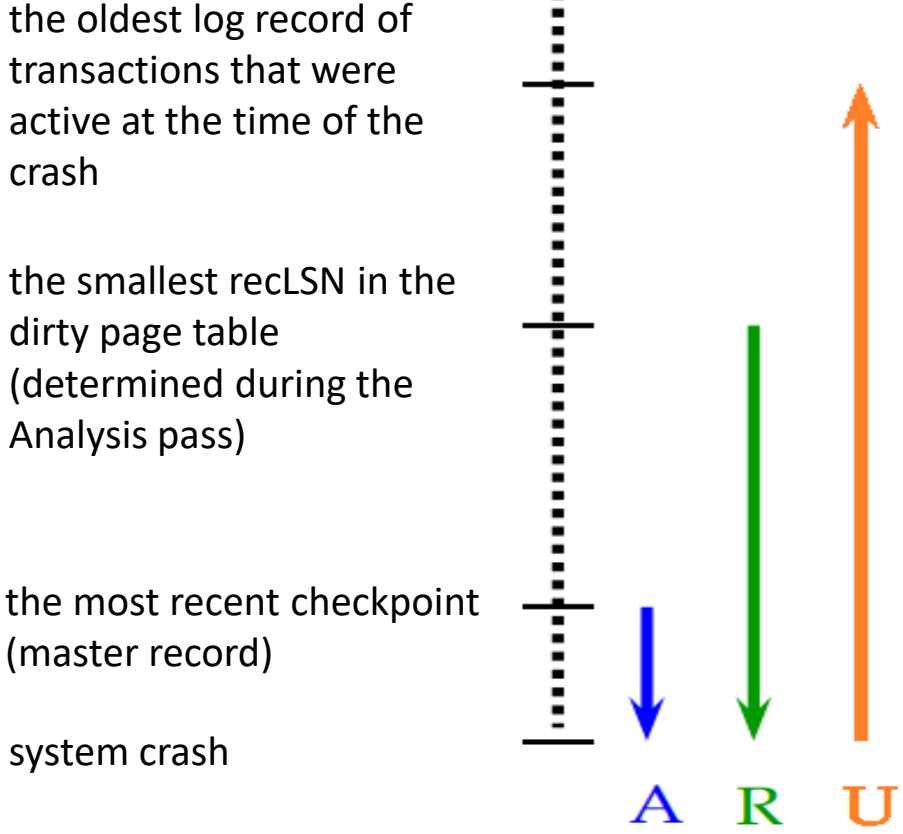
Lecture 5

Crash Recovery

Sabina S. CS

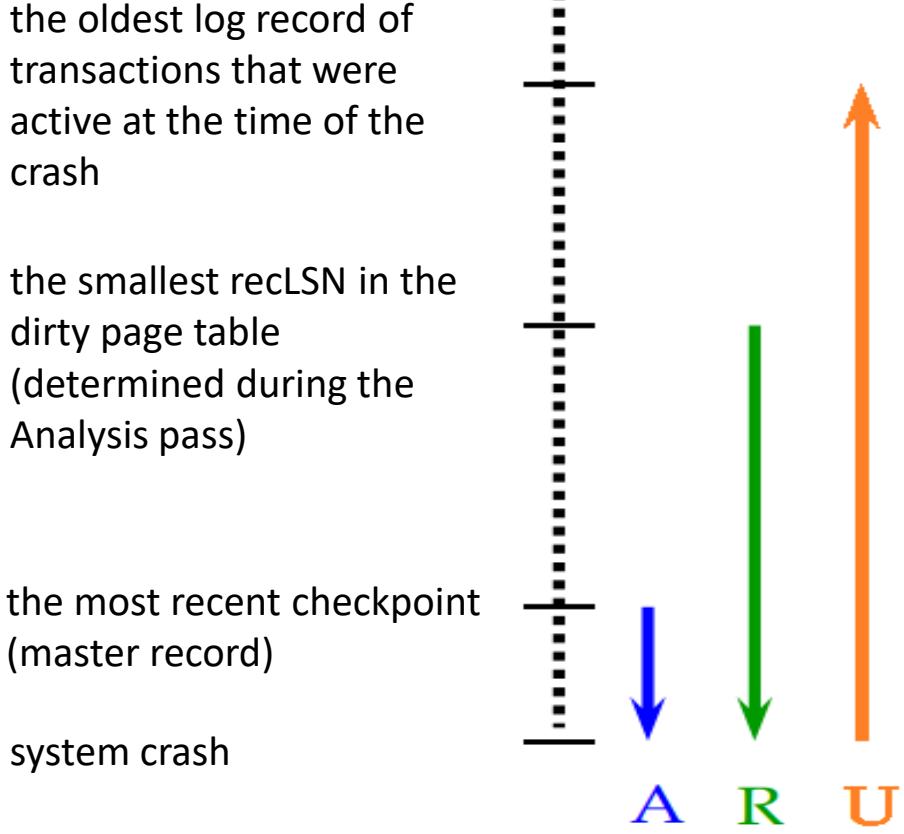
Recovery - overview

- system restart after a crash – 3 phases:
  - Analysis
    - reconstructs state at the most recent checkpoint
    - scans the log forward from the most recent checkpoint
    - identifies:
      - active transactions at the time of the crash (to be undone)
      - potentially dirty pages at the time of the crash
      - the starting point for the Redo pass



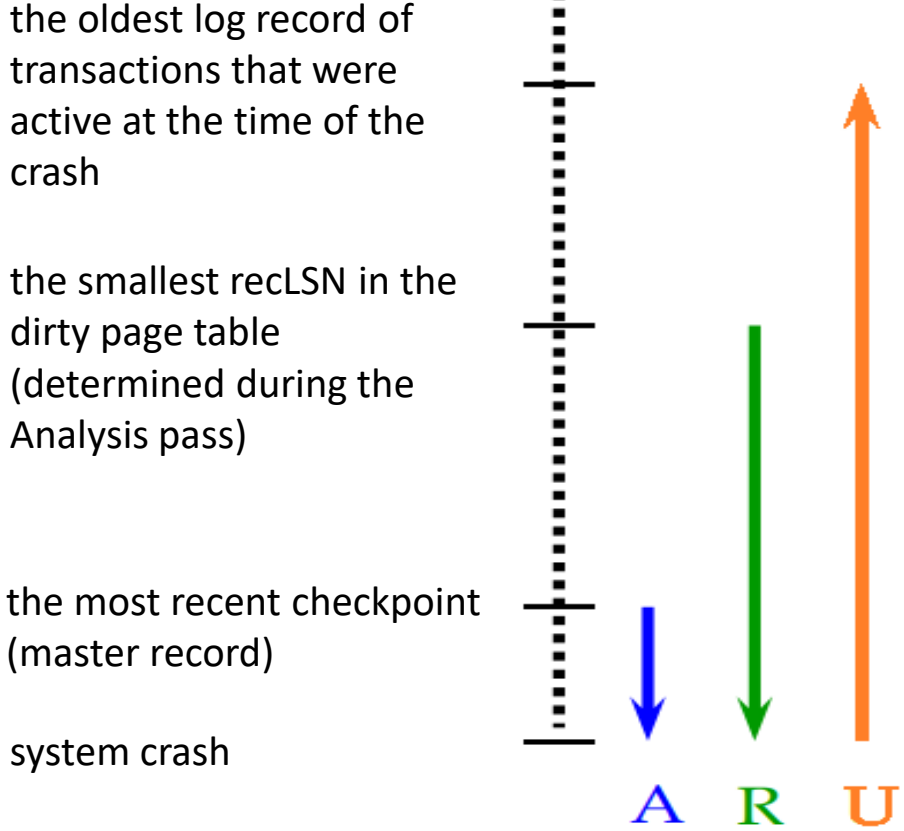
Recovery - overview

- system restart after a crash – 3 phases:
  - Redo
    - repeats history, i.e., reapplies changes to dirty pages
    - all updates are reapplied (regardless of whether the corresponding transaction committed or not)
    - starting point is determined in the Analysis pass
    - scans the log forward until the last record



Recovery - overview

- system restart after a crash – 3 phases
  - Undo
    - the effects of transactions that were active at the time of the crash are undone
    - such changes are undone in the opposite order (i.e., Undo scans the log backward from the last record)



* Analysis

- investigate the most recent *begin_checkpoint* log record
  - get the next *end_checkpoint* log record EC
- set Dirty Page Table to the copy of the Dirty Page Table in EC
- set Transaction Table to the copy of the Transaction Table in EC

->

* Analysis - scan the log forward from the most recent checkpoint:

- transactions:
  - encounter end log record for transaction T:
    - remove T from Transaction Table
  - encounter other log records (LR) for transaction T:
    - add T to Transaction Table if not already there
    - set T.lastLSN to LR.LSN
    - if LR is a commit type log record:
      - set T's status to *C*
    - otherwise, set status to *U* (i.e., to be undone)

* Analysis - scan the log forward from the most recent checkpoint:

- pages:
  - encounter redoable log record (LR) for page P:
    - if P is not in the Dirty Page Table:
      - add P to Dirty Page Table
      - set P.recLSN to LR.LSN

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					
	T10	update	P11	3	20	GFX	YTR

log

- first 5 log records are written to stable storage
- system crashes before the 6^{th} log record is written to stable storage

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Analysis

- most recent checkpoint – beginning of execution (empty Transaction Table, empty Dirty Page Table)
- 1st log record
  - add T10 to the Transaction Table
  - add P100 to the Dirty Page Table (recLSN = LSN(1st log record))

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Analysis

- 2nd log record
  - add T15 to the Transaction Table
  - add P2 to the Dirty Page Table (recLSN = LSN(2nd log record))

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Analysis

- 4th log record
  - add P10 to the Dirty Page Table (recLSN = LSN(4th log record))
- active transactions at the time of the crash:
  - transactions with status *U*, i.e., T10 (T15 is a committed transaction)

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Analysis

- Dirty Page Table:
  - can include pages that were written to disk prior to the crash
  - assume P2's update is the only change written to disk before the crash, i.e., P2 is not dirty, but it's in the Dirty Page Table
  - the pageLSN on page P2 is equal to the LSN of the 2nd log record

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Analysis

- log record

T10	update	P11	3	20	GFX	YTR

is not seen during Analysis (it was not written to disk before the crash)
- Write-Ahead Logging protocol => the corresponding change to page P11 cannot have been written to disk

* Redo

- *repeat history*: reconstruct state at the time of the crash
  - reapply *all* updates (even those of aborted transactions!), reapply CLRs
- scan the log forward from the log record with the smallest recLSN in the Dirty Page Table
- for each redoable log record LR affecting page P, redo the described action unless one of the conditions below is satisfied:
  - page P is not in the Dirty Page Table
  - page P is in the Dirty Page Table, but P.recLSN > LR.LSN
  - P.pageLSN (in DB) $\geq$ LR.LSN
- to redo an action:
  - reapply the logged action
  - set P.pageLSN to LR.LSN
  - no additional logging!

* Redo
- at the end of Redo:
  - for every transaction T with status *C*:
    - add an end log record
    - remove T from the Transaction Table

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Redo

- previously stated assumption: P2's update is the only change written to disk before the crash, i.e., P2 is not dirty, but it's in the Dirty Page Table
- Dirty Page Table -> smallest recLSN is the LSN of the 1st log record

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Redo

- 1st log record
  - fetch page P100 (its pageLSN is less than the LSN of the current log record) => reapply update, set P100.pageLSN to the LSN of the 1st log record

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Redo

- 2nd log record
  - fetch page P2
  - P2.pageLSN = LSN of the current log record => update is not reapplied
- 3rd, 4th log records – processed similarly

* Undo

- *loser transaction* – transaction that was active at the time of the crash
- ToUndo = { l | l - lastLSN of a *loser* transaction}
- repeat:
  - choose the largest LSN in ToUndo and process the corresponding log record LR; let T be the corresponding transaction
  - if LR is a CLR:
    - if undoNextLSN == NULL
      - write an end log record for T
    - else {undoNextLSN != NULL}
      - add undoNextLSN to ToUndo
  - else {LR is an update log record}
    - write a CLR
    - undo the update
    - add LR.prevLSN to ToUndo

until ToUndo is empty

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

log

Undo

- active transaction at the time of the crash: T10
- lastLSN of T10: LSN of the 4th log record
- 4th log record
  - undo update, write CLR
  - add LSN of 1st log record to ToUndo

Example 1

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

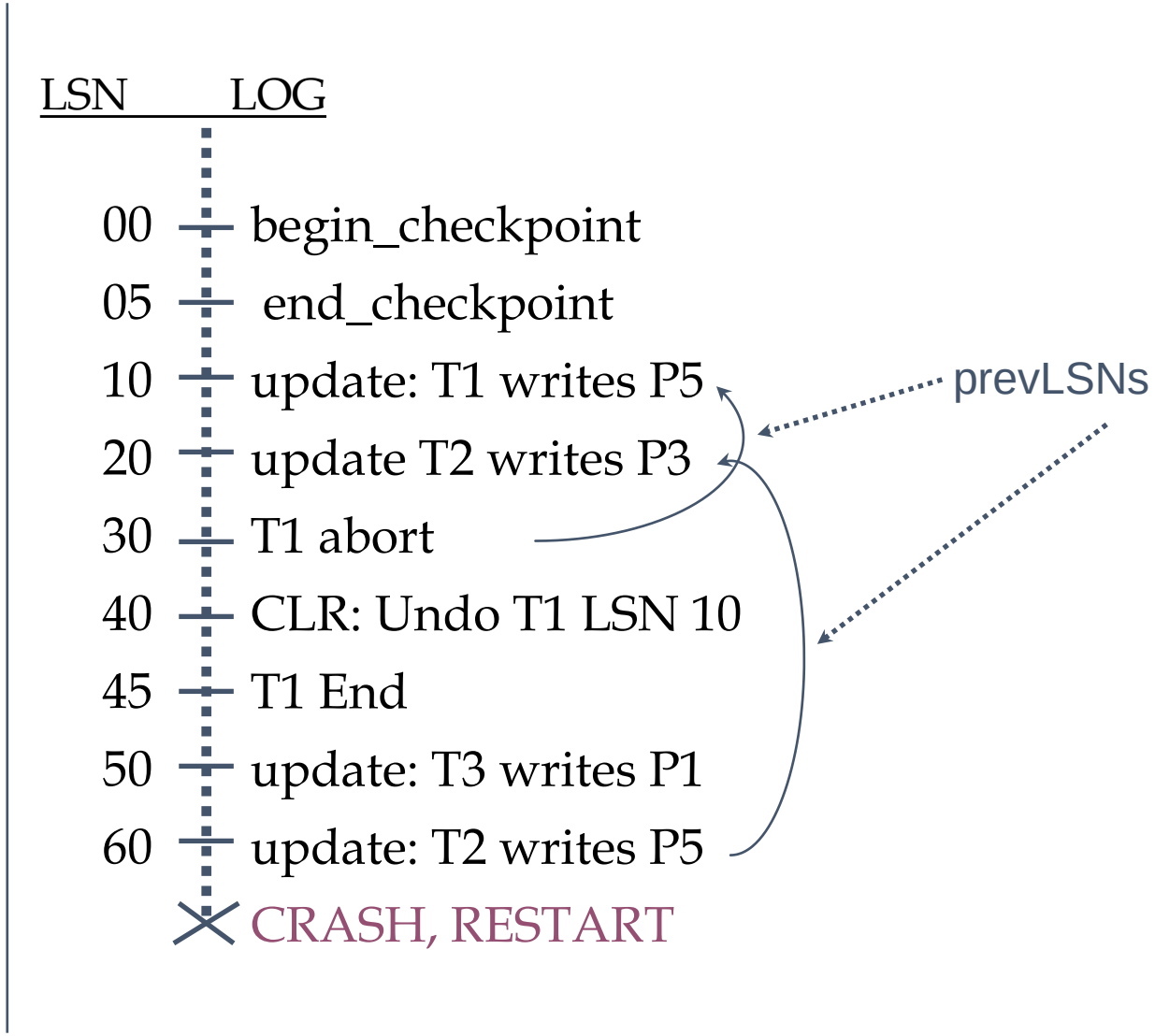
log

Undo

- 1st log record
  - undo update (!T15's change to P100 is lost!)
  - write CLR, write end log record for T10
- obs. if Strict 2PL is used, T15 cannot write P100 while T10 is active (T10 has also modified P100)

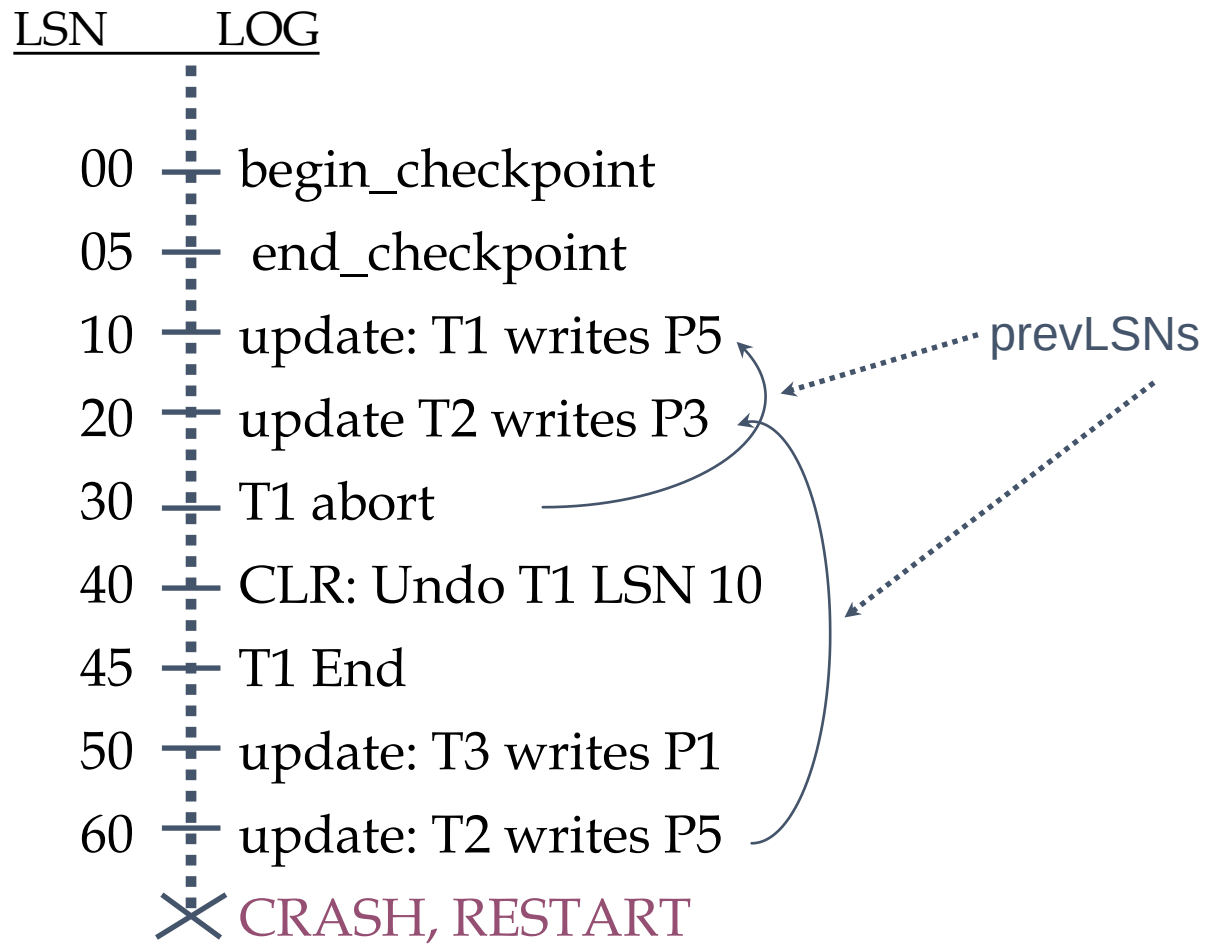
Example 2 – system crashes during Undo

- consider the execution history below:



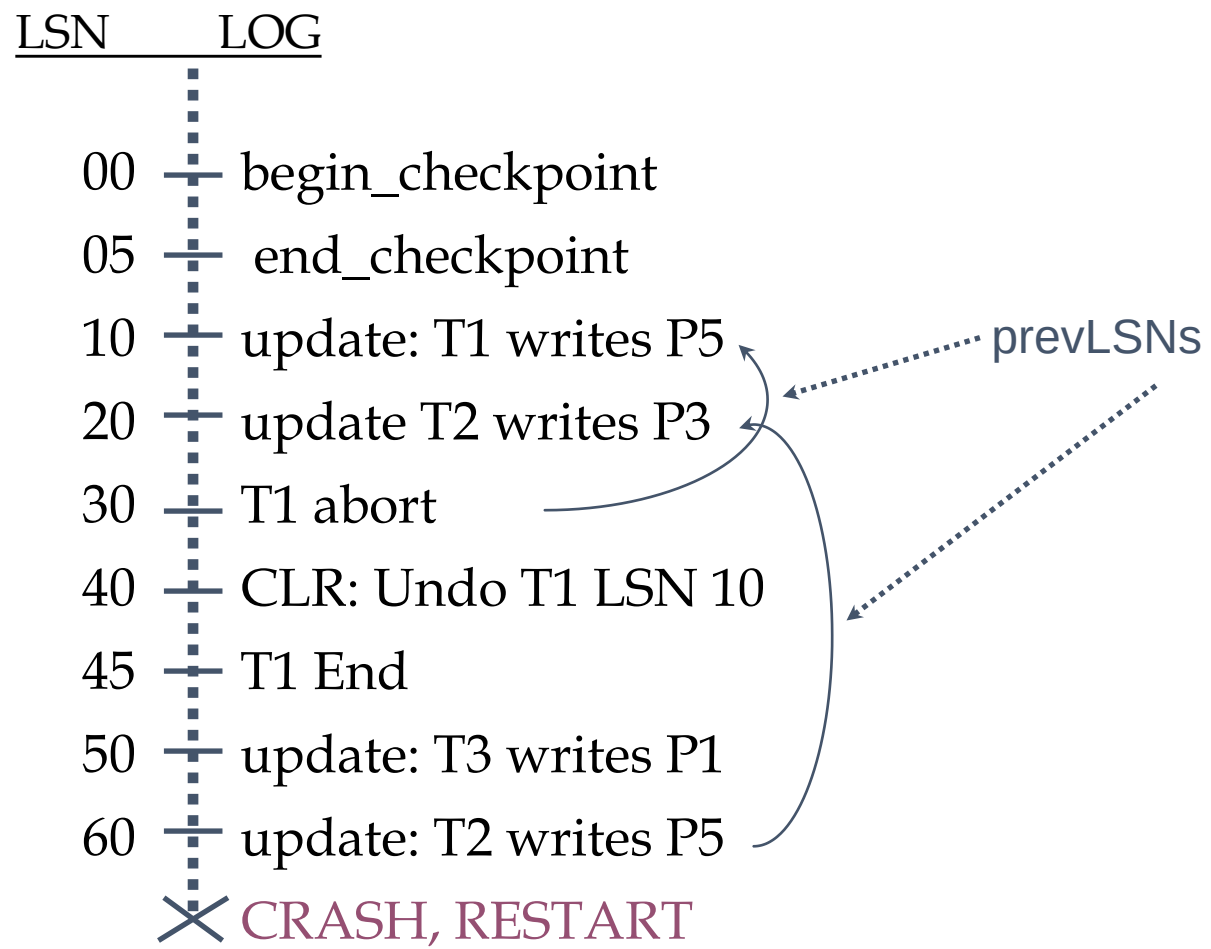
Example 2

- T1 aborts
  => its only update is undone
  (CLR with LSN 40)
  - T1 - terminated
- 1st crash:
  - Analysis:
    - dirty pages: P5 (recLSN 10), P3 (recLSN 20), P1 (recLSN 50)
    - active transactions at the time of the crash: T2 (lastLSN 60), T3 (lastLSN 50)



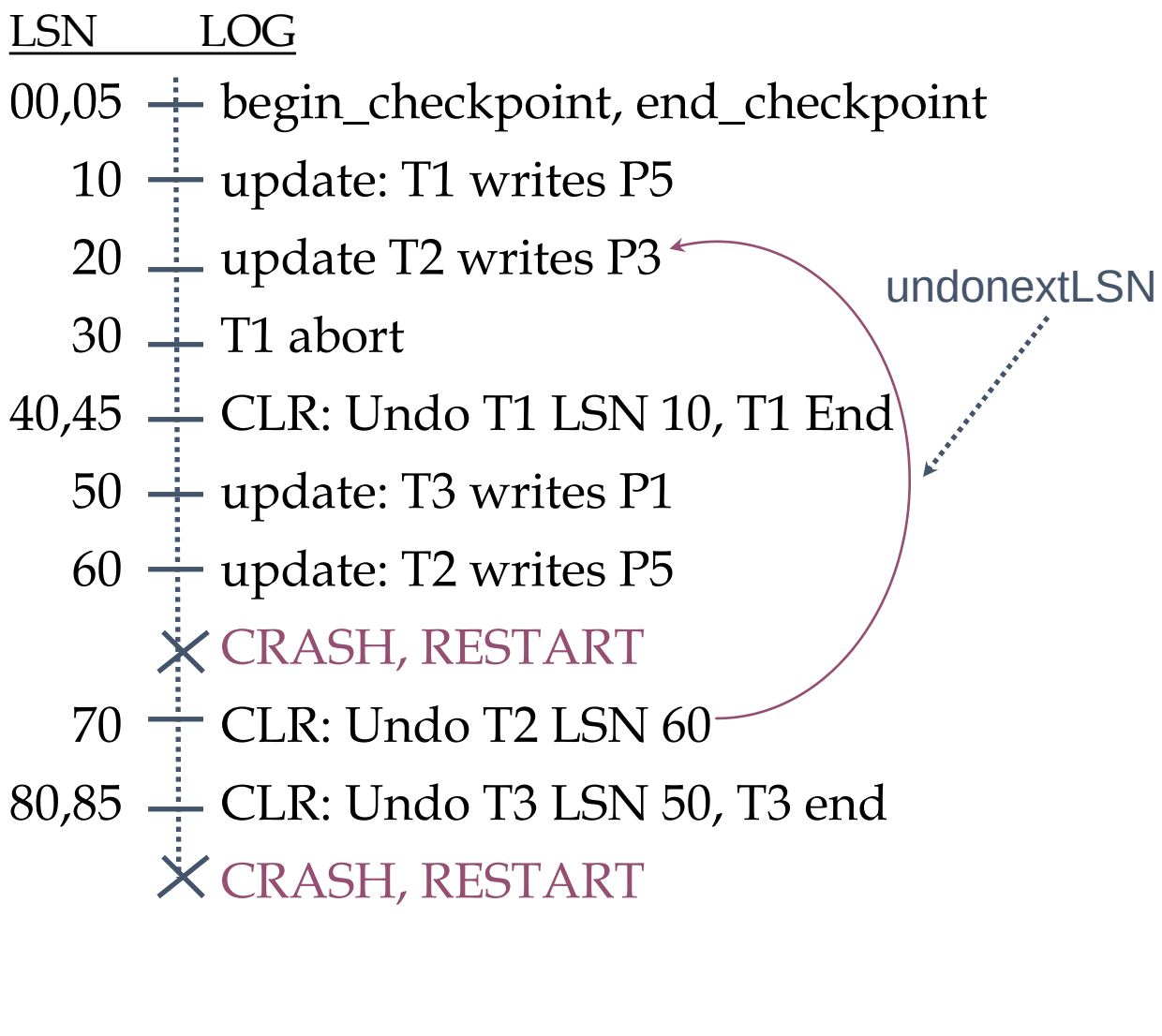
Example 2

- 1st crash:
  - Redo:
    - starting point
      - log record with LSN = 10 (smallest recLSN in the Dirty Page Table)
    - reapply required actions in update log records / compensation log records



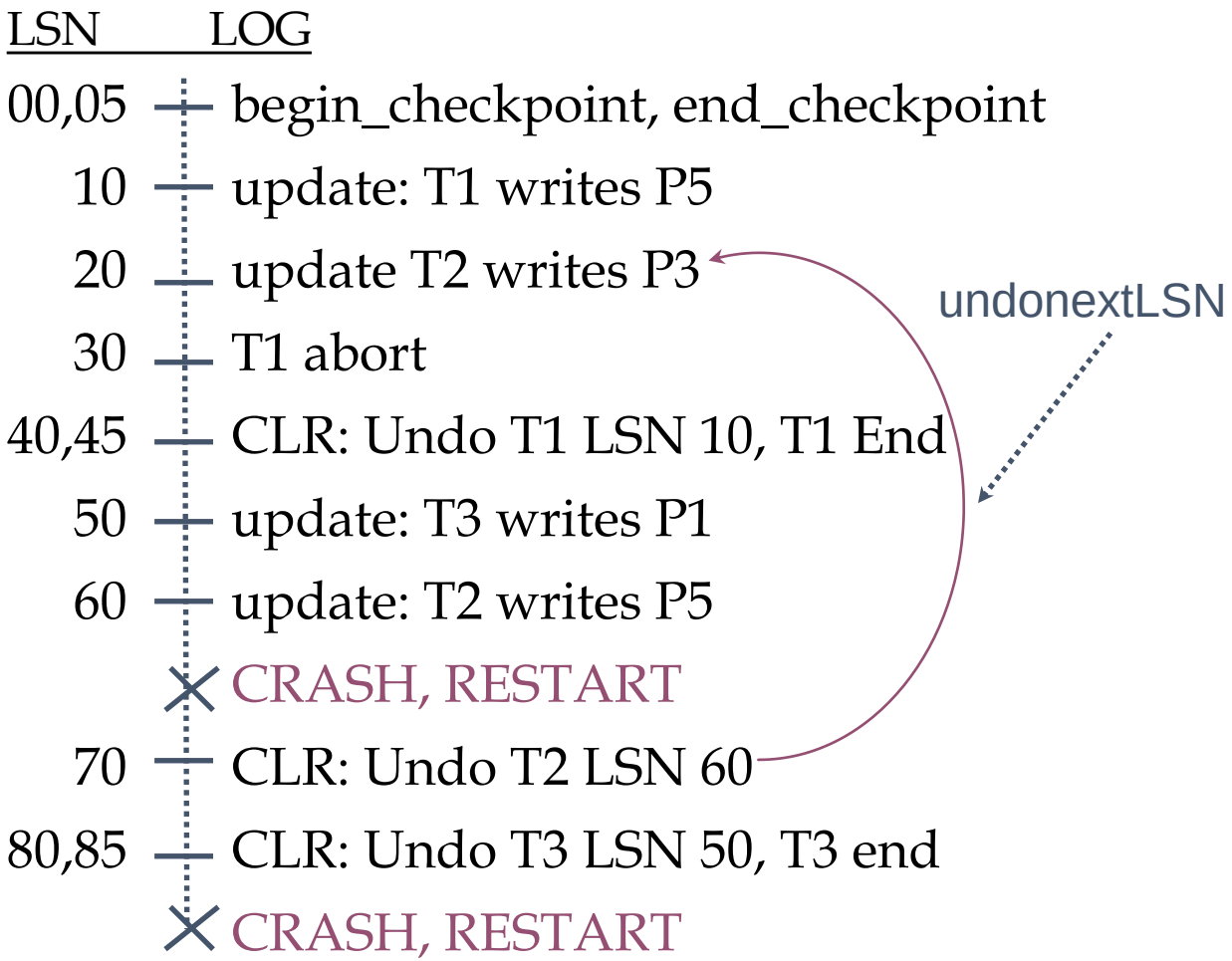
Example 2

- 1st crash:
  - Undo:
    - T2, T3 – loser transactions => ToUndo = {60, 50}
    - process log record with LSN 60:
      - undo update
      - write CLR (LSN 70) with undoNextLSN 20 (i.e., the next log record that should be processed for T2)



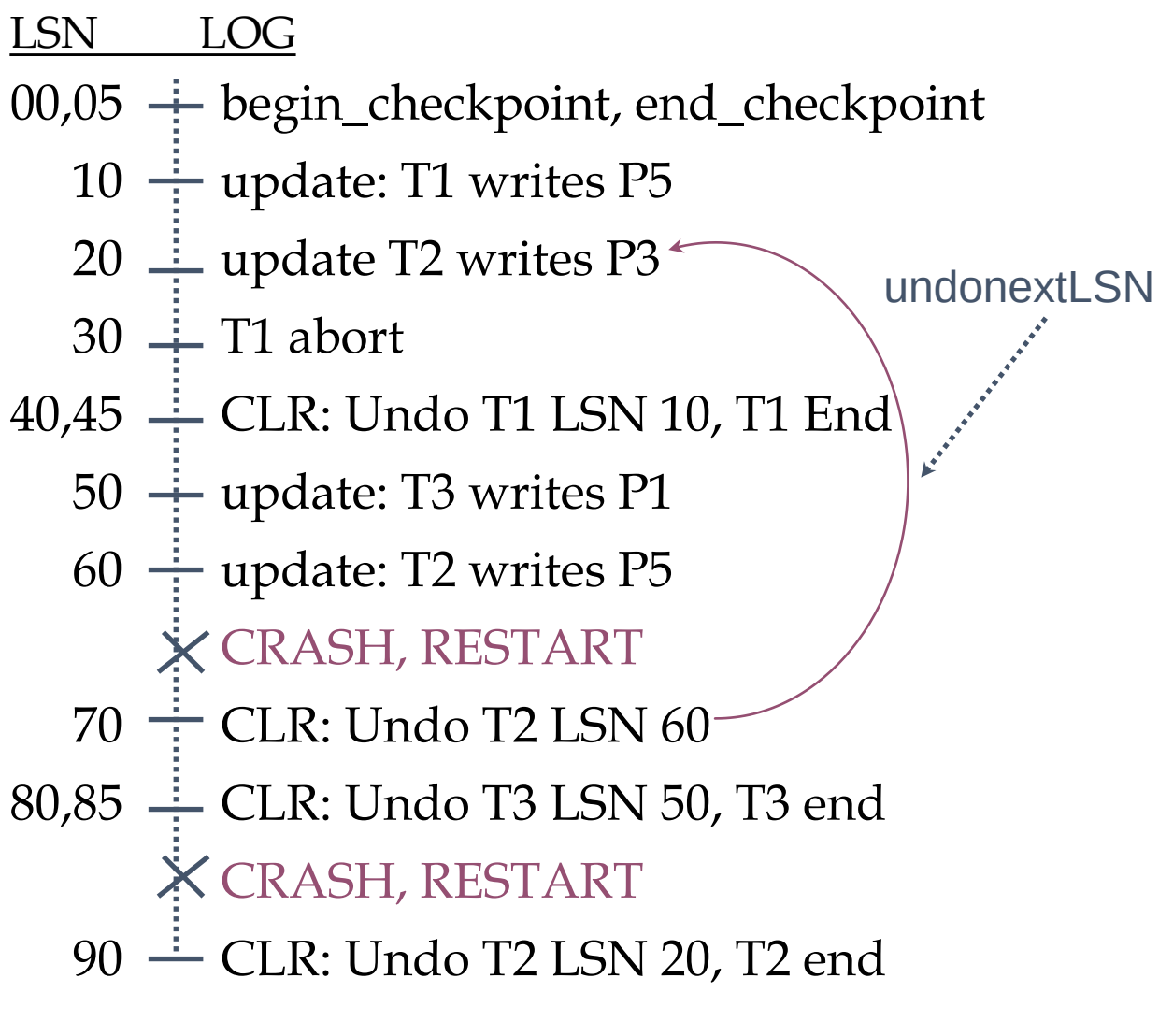
Example 2

- 1st crash:
  - Undo:
    - process log record with LSN 50:
      - undo update
      - write CLR (LSN 80) with undoNextLSN *null* (i.e., T3 completely undone, write end log record for T3)
    - log records with LSN 70, 80, 85 are written to stable storage
- 2nd crash (during undo)!



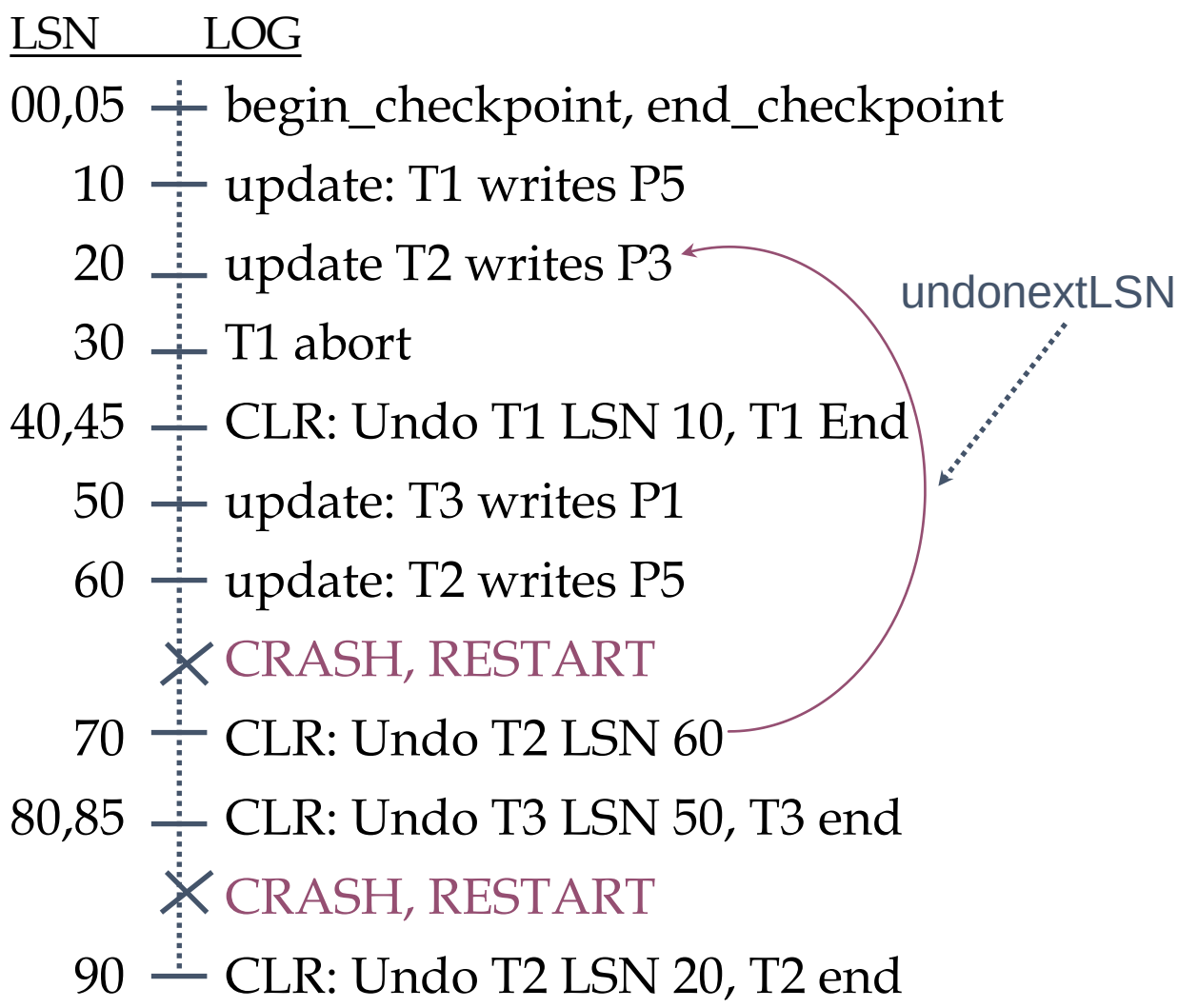
Example 2

- 2nd crash:
  - Analysis:
    - the only active transaction: T2
    - dirty pages: P5 (recLSN 10), P3 (recLSN 20), P1 (recLSN 50)
  - Redo:
    - process log records with LSN between 10 and 85



Example 2

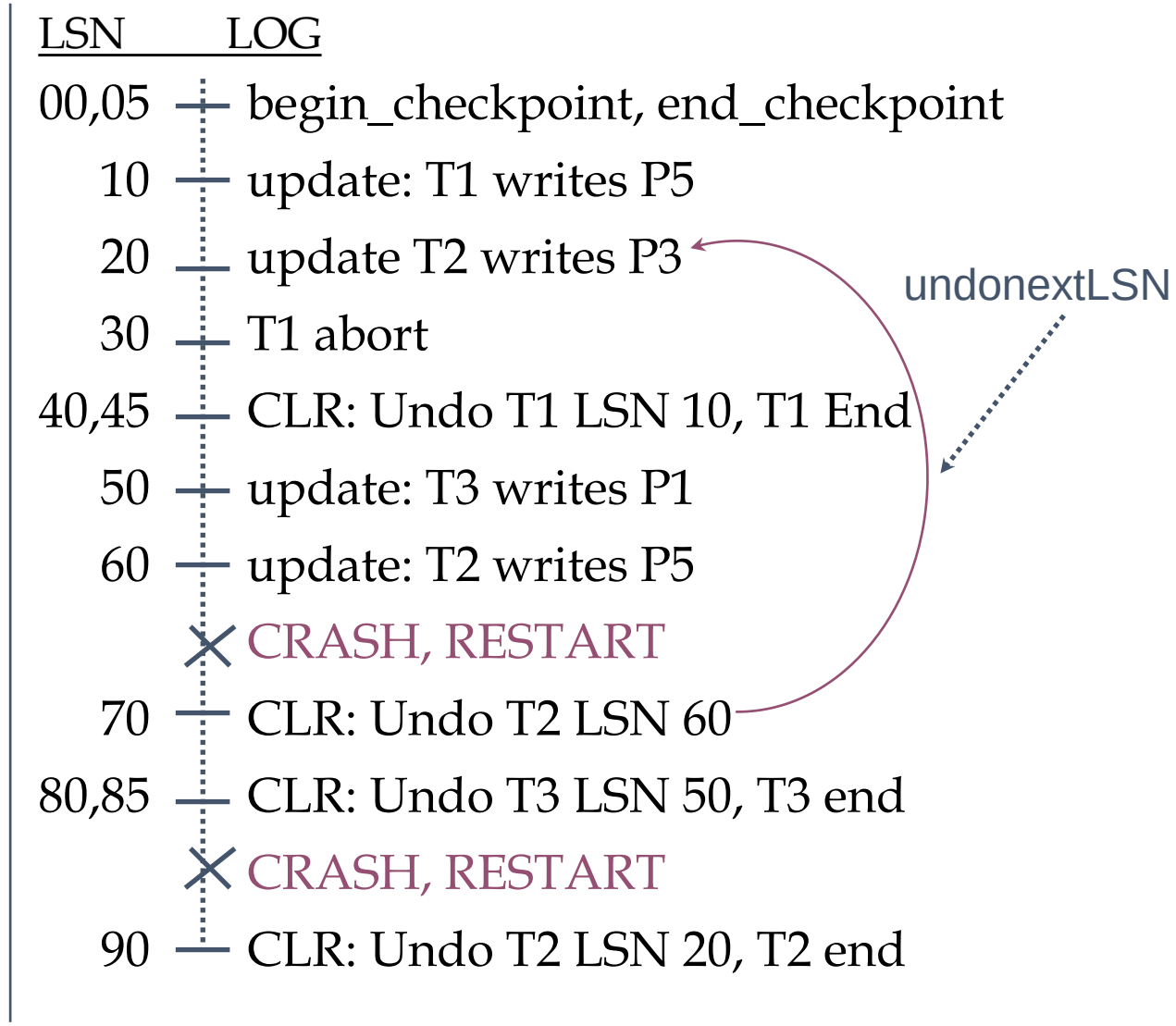
- 2nd crash:
  - Undo:
    - lastLSN of T2: 70
    - ToUndo = {70}
    - process log record with LSN 70:
      - add 20 (undoNextLSN) to ToUndo
    - process log record with LSN 20:
      - undo update
      - write CLR (LSN 90) with undoNextLSN *null* => write end log record for T2



Example 2

- 2nd crash:
  - Undo:
    - ToUndo empty

  => recovery complete!



- obs. aborting a transaction
  - special case of Undo in which the actions of a single transaction are undone
- obs. system crash during the Analysis pass
  - all the work is lost
  - when the system comes back up, the Analysis phase has the same information as before
- obs. system crash during the Redo pass
  - some of the changes from the Redo pass may have been written to disk prior to the crash
  - the pageLSN will indicate such a situation, so these changes will not be reapplied in the subsequent Redo pass

References

- [Ra02] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (3rd Edition), McGraw-Hill, 2002
- [Le99] LEVENE, M., LOIZOU, G., A Guided Tour of Relational Databases and Beyond, Springer, 1999
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Si19] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts (7th Edition), McGraw-Hill, 2019