Database Management Systems

Lecture 5

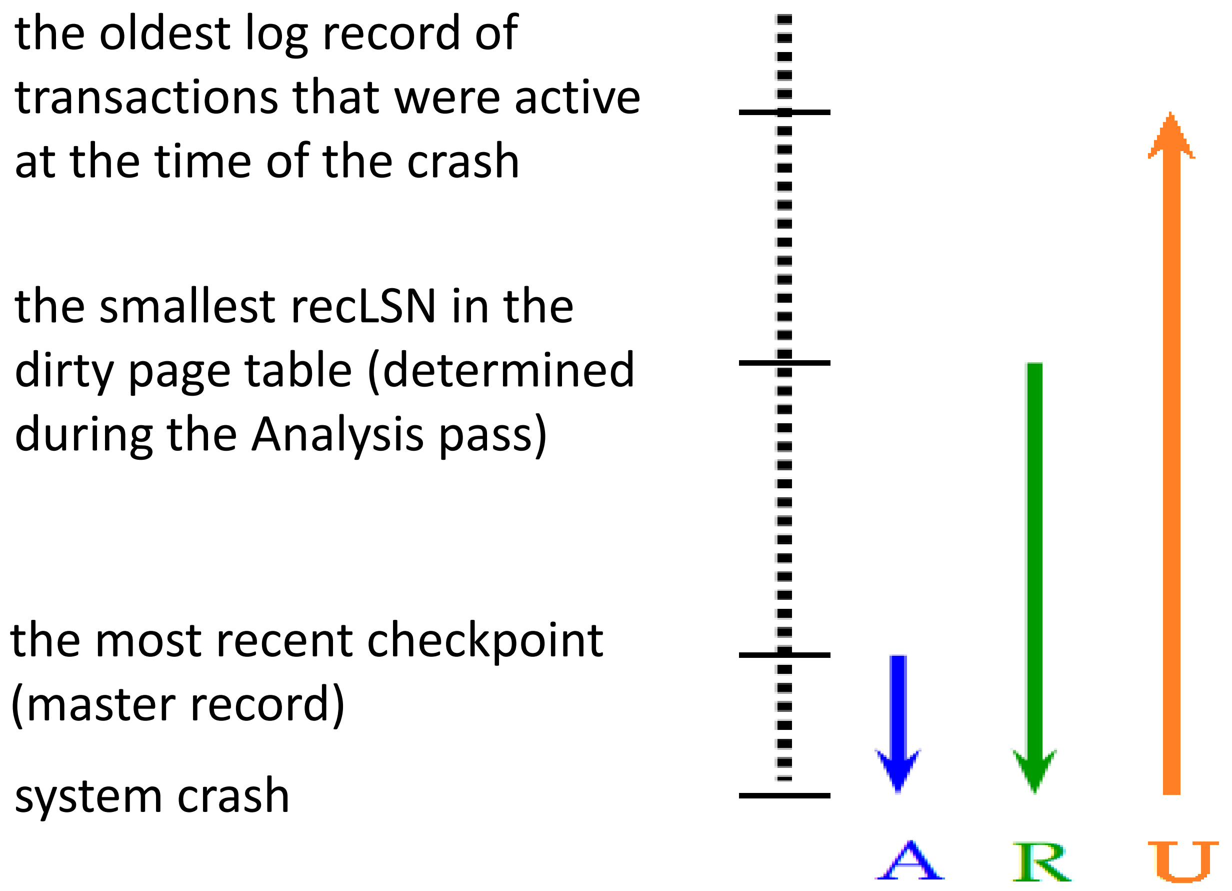
Crash Recovery (II)

Security

Crash Recovery

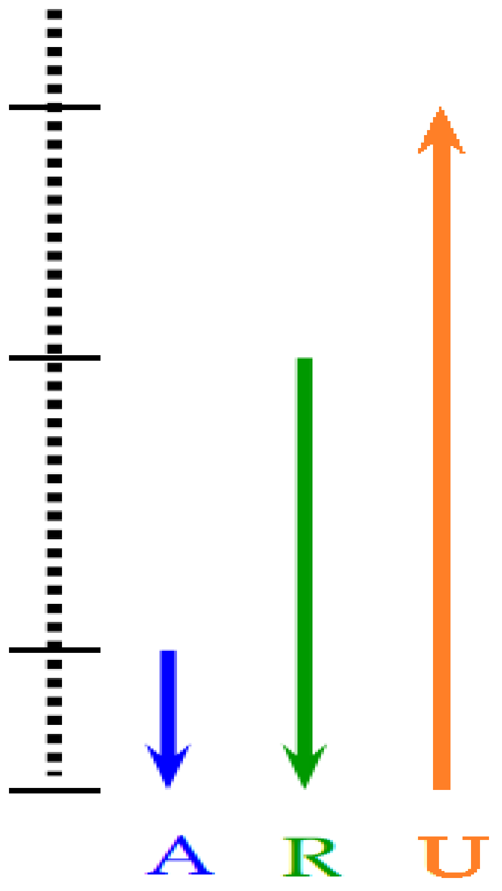
Recovery - overview

- system restart after a crash
- restart – 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)


the oldest log record of transactions that were active at the time of the crash
the smallest recLSN in the dirty page table (determined during the Analysis pass)
the most recent checkpoint (master record)
system crash
A
R
U

Recovery - overview

- system restart after a crash
- restart – 3 phases
  - Analysis
    - identifies:
      - potentially dirty pages at the time of the crash
      - the starting point for the Redo pass


the oldest log record of transactions that were active at the time of the crash
the smallest recLSN in the dirty page table (determined during the Analysis pass)
the most recent checkpoint (master record)
system crash
A
R
U

Recovery - overview

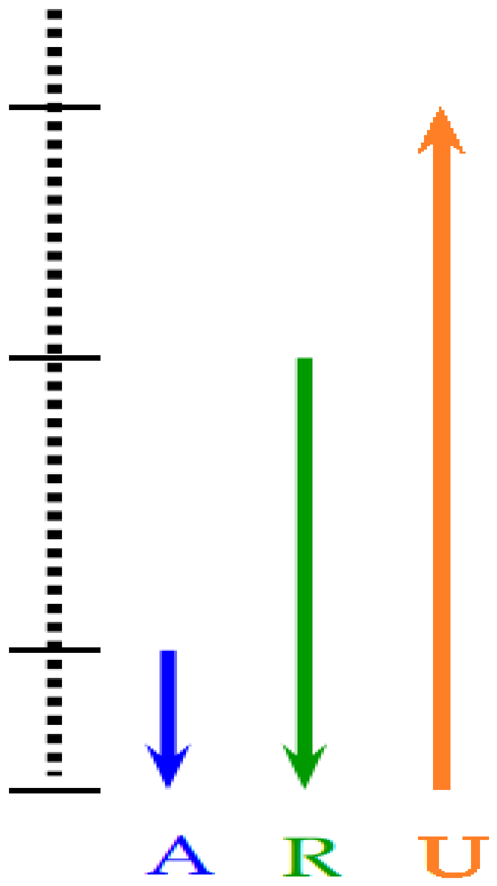
- system restart after a crash
- restart – 3 phases
  - Redo
    - repeats history, i.e., reapplies changes to dirty pages
    - all updates are reapplied (regardless of whether the corresponding transaction committed or not)

the oldest log record of transactions that were active at the time of the crash

the smallest recLSN in the dirty page table (determined during the Analysis pass)

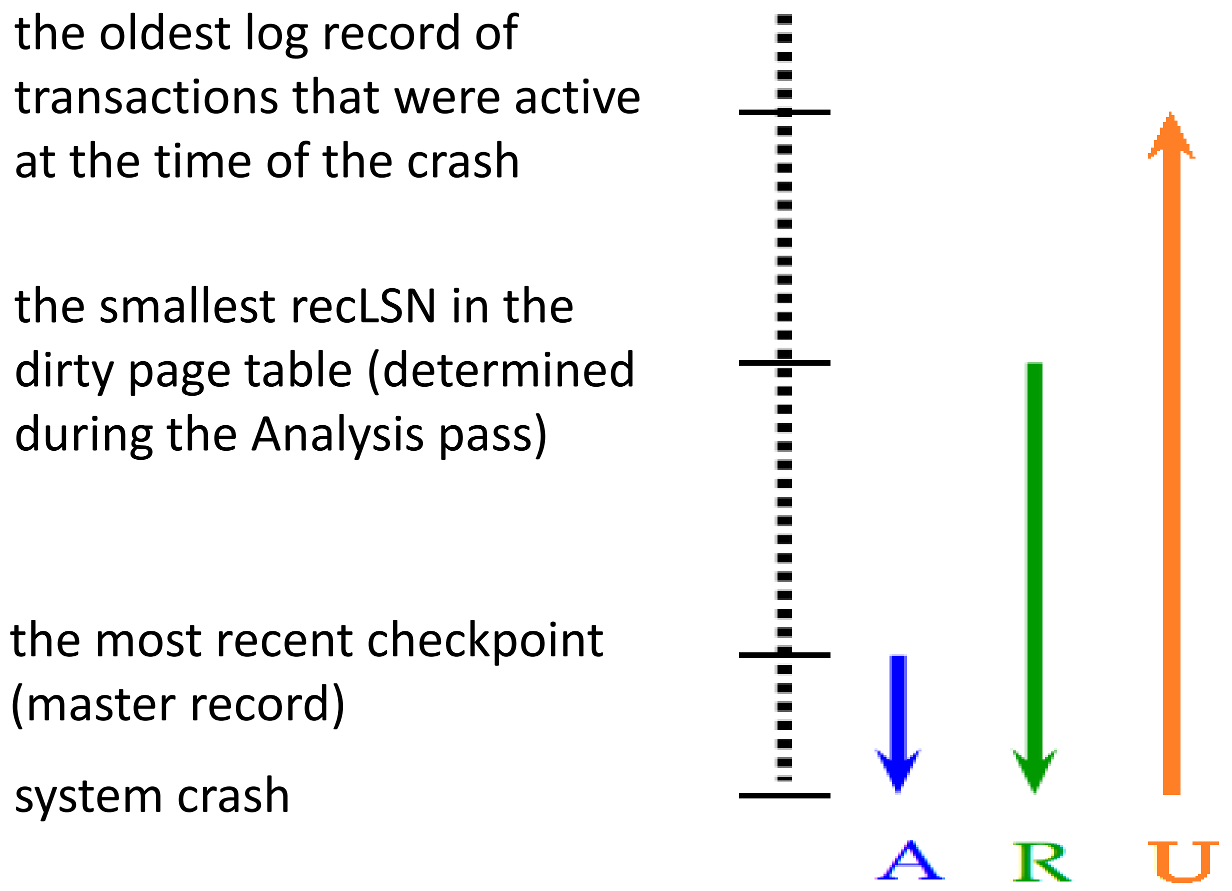
the most recent checkpoint (master record)

system crash



Recovery - overview

- system restart after a crash
- restart – 3 phases
    - Redo
        - starting point is determined in the Analysis pass
        - scans the log forward until the last record



the oldest log record of transactions that were active at the time of the crash

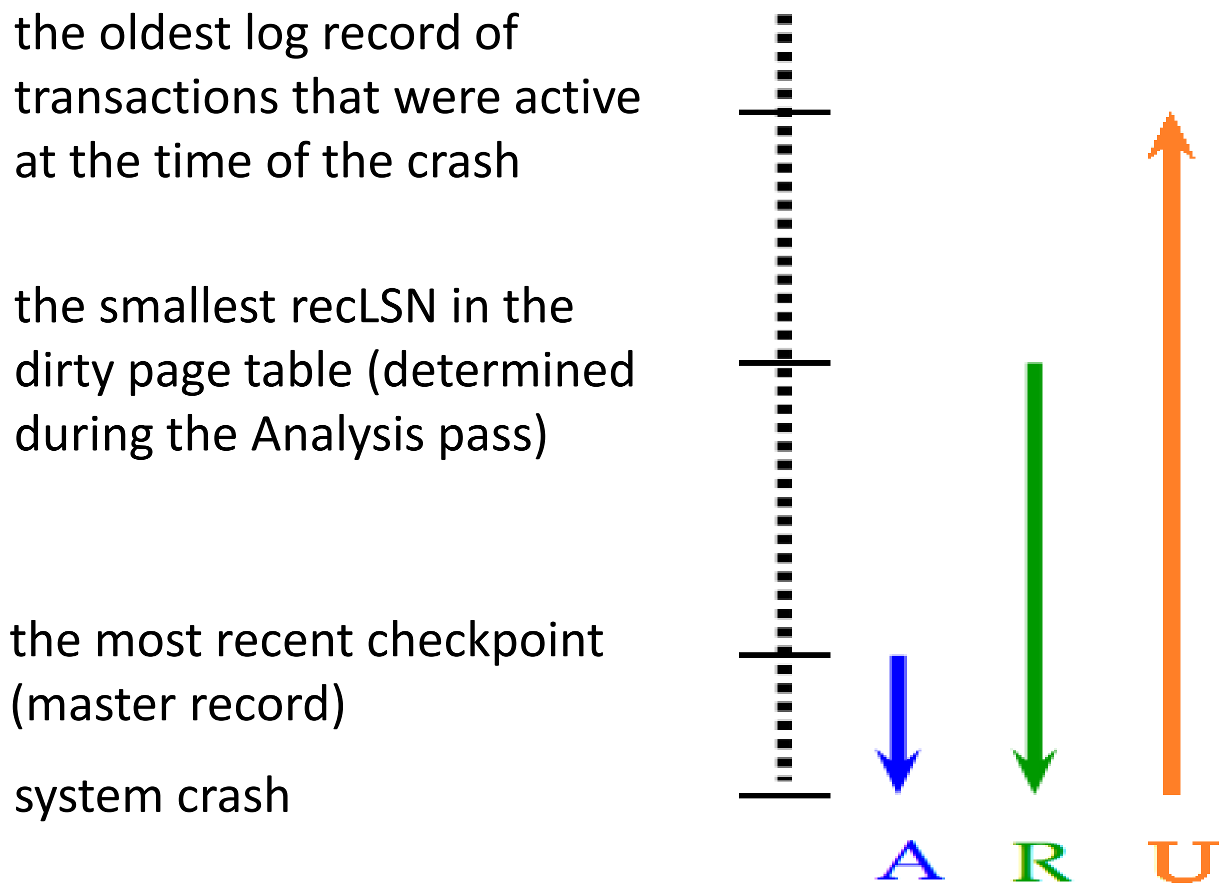
the smallest recLSN in the dirty page table (determined during the Analysis pass)

the most recent checkpoint (master record)

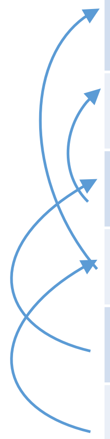
system crash

Recovery - overview

- system restart after a crash
- restart – 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)


the oldest log record of transactions that were active at the time of the crash
the smallest recLSN in the dirty page table (determined during the Analysis pass)
the most recent checkpoint (master record)
system crash
A
R
U

Running Example



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

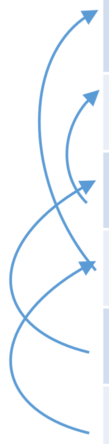
log

- 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:
      - end log record for transaction T:
        - remove T from transaction table
      - 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 status to C
        - otherwise, set it to U (i.e., to be undone)

- Analysis
    - scan the log forward from the most recent checkpoint
        - pages
            - redoable log record (LR) for page P:
            - if P is not in the Dirty Page Table
                - add P to the Dirty Page Table
                - set P.recLSN to LR.LSN

Running Example



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 6th log record is written to stable storage

Running Example

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 (i.e., 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))

Running Example

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))
  - 4th log record
    - add P10 to the Dirty Page Table (recLSN = LSN(4th log record))

Running Example

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
  - active transactions at the time of the crash
    - transactions with status U, i.e., T10 (T15 is a committed transaction)
  - 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
    - nevertheless, the pageLSN on page P2 is equal to the LSN of the 2nd log record

Running Example

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

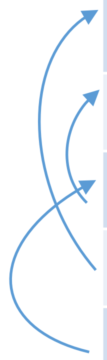
T10	update	P11	3	20	GFX	YTR

    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:
    - 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

Running Example

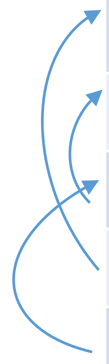


prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

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

Running Example



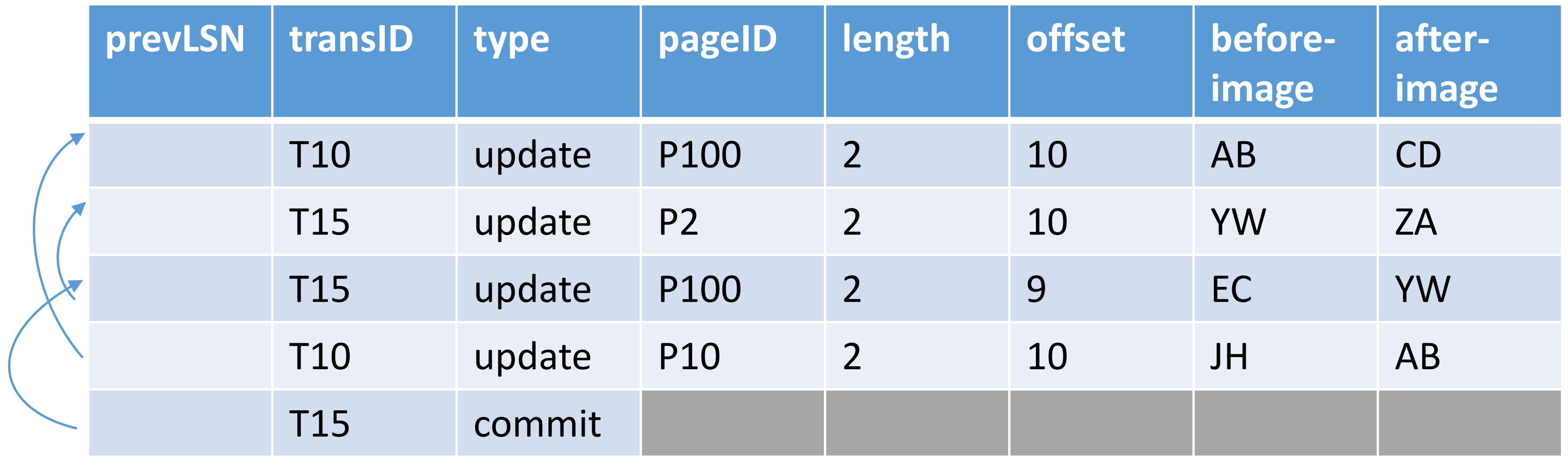
prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

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}
            - undo the update
            - write a CLR
            - add LR.prevLSN to ToUndo
    - until ToUndo is empty

Running Example



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

Running Example

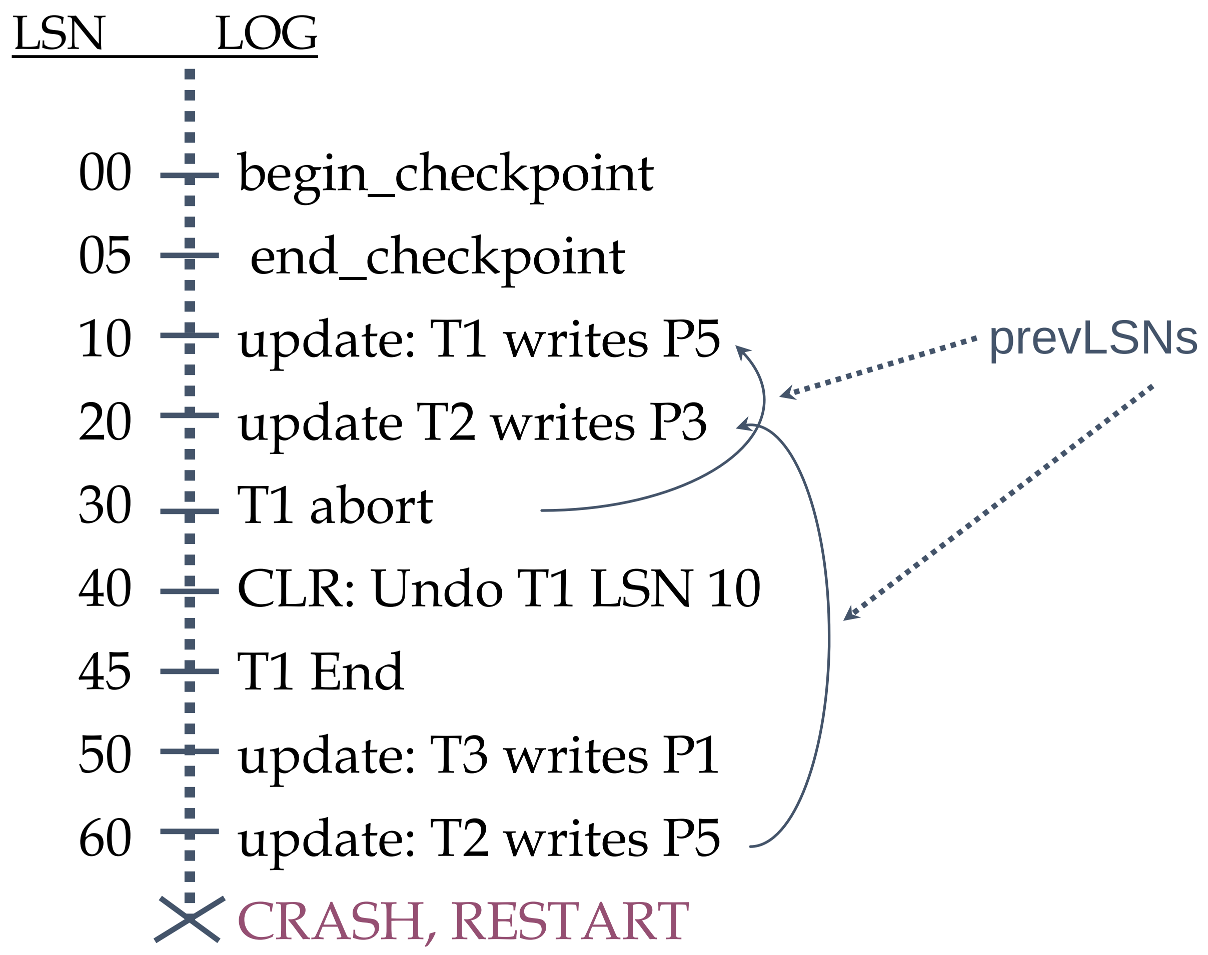
prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T10	update	P100	2	10	AB	CD
	T15	update	P2	2	10	YW	ZA
	T15	update	P100	2	9	EC	YW
	T10	update	P10	2	10	JH	AB
	T15	commit					

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)

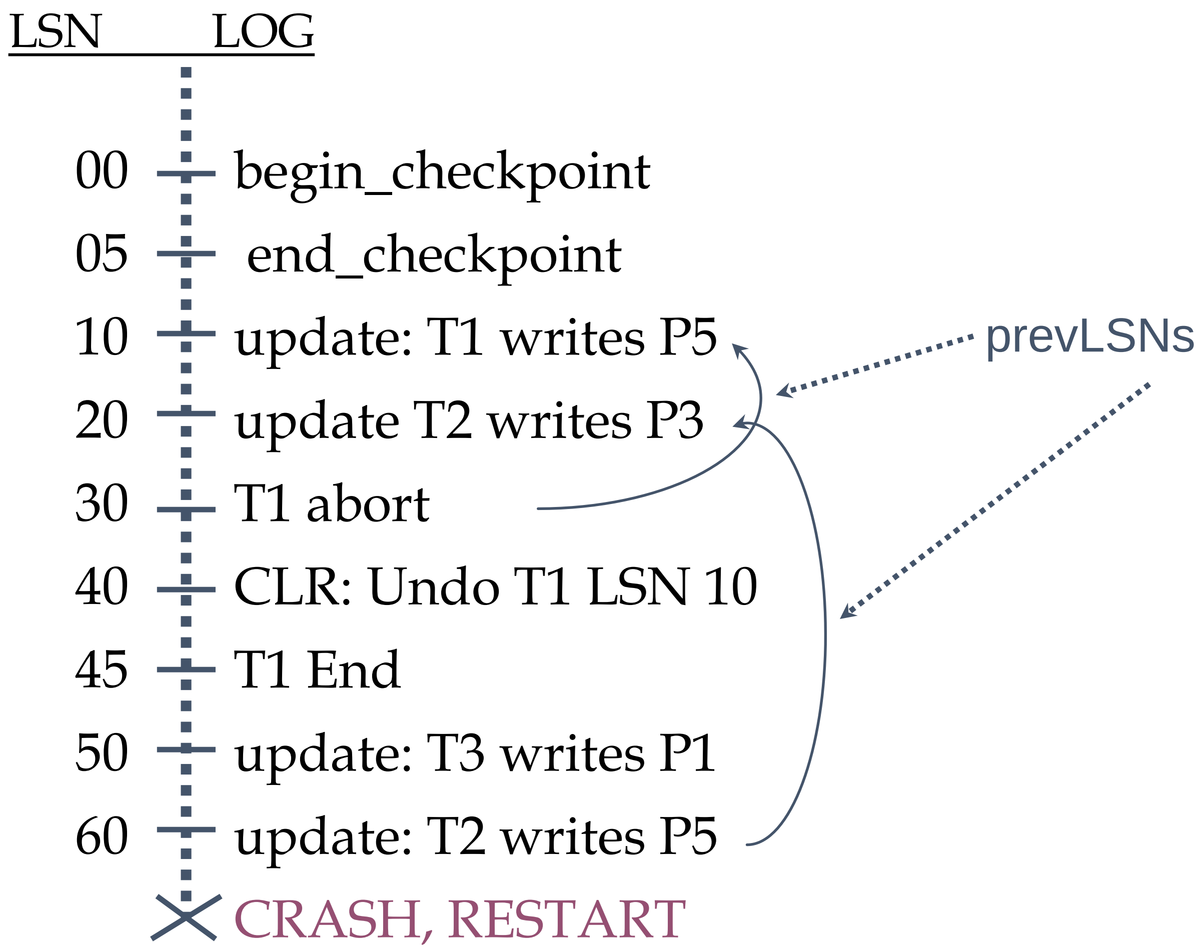
Example 2 – system crashes during Undo

- consider the execution history below:



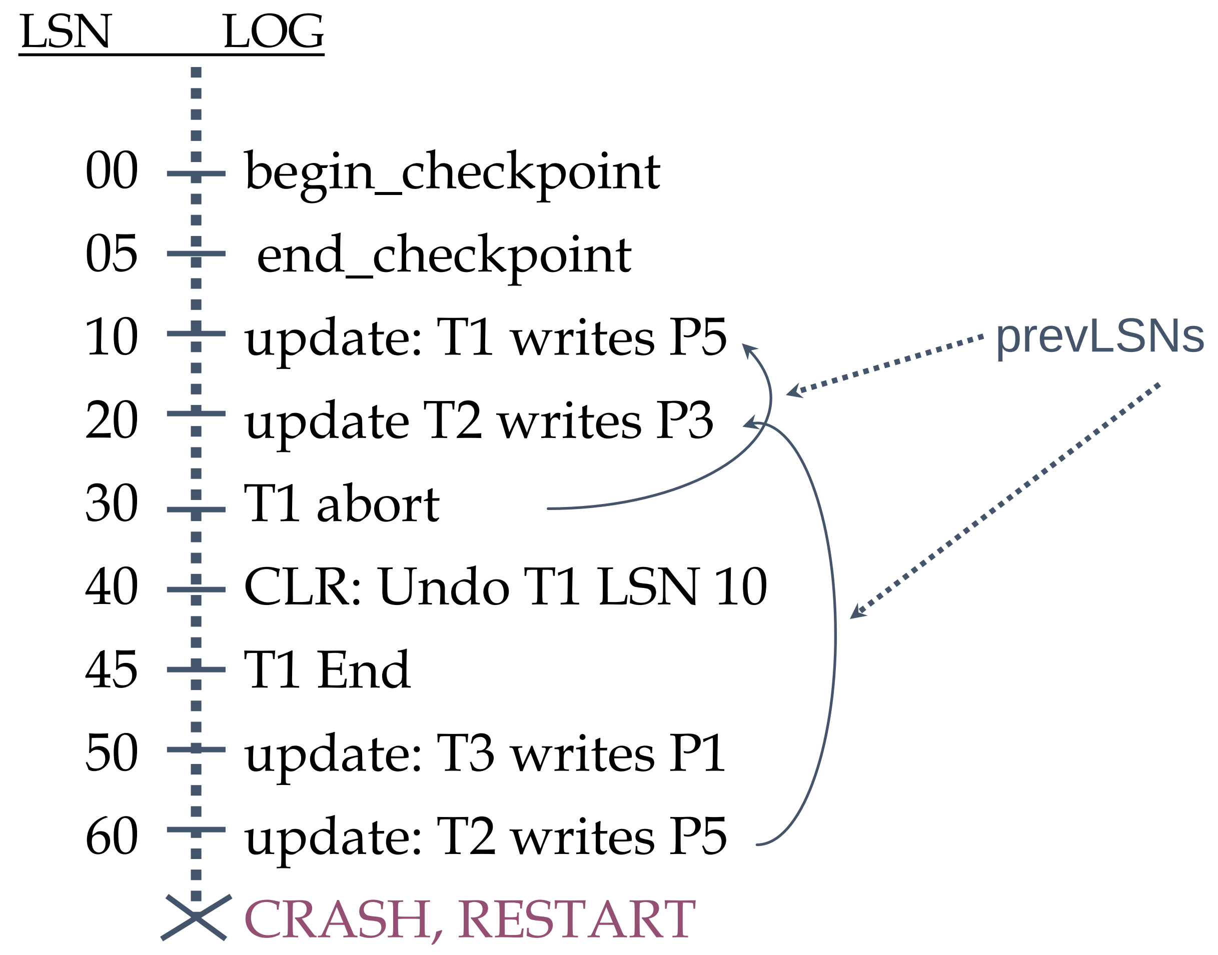
Example 2

- 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)



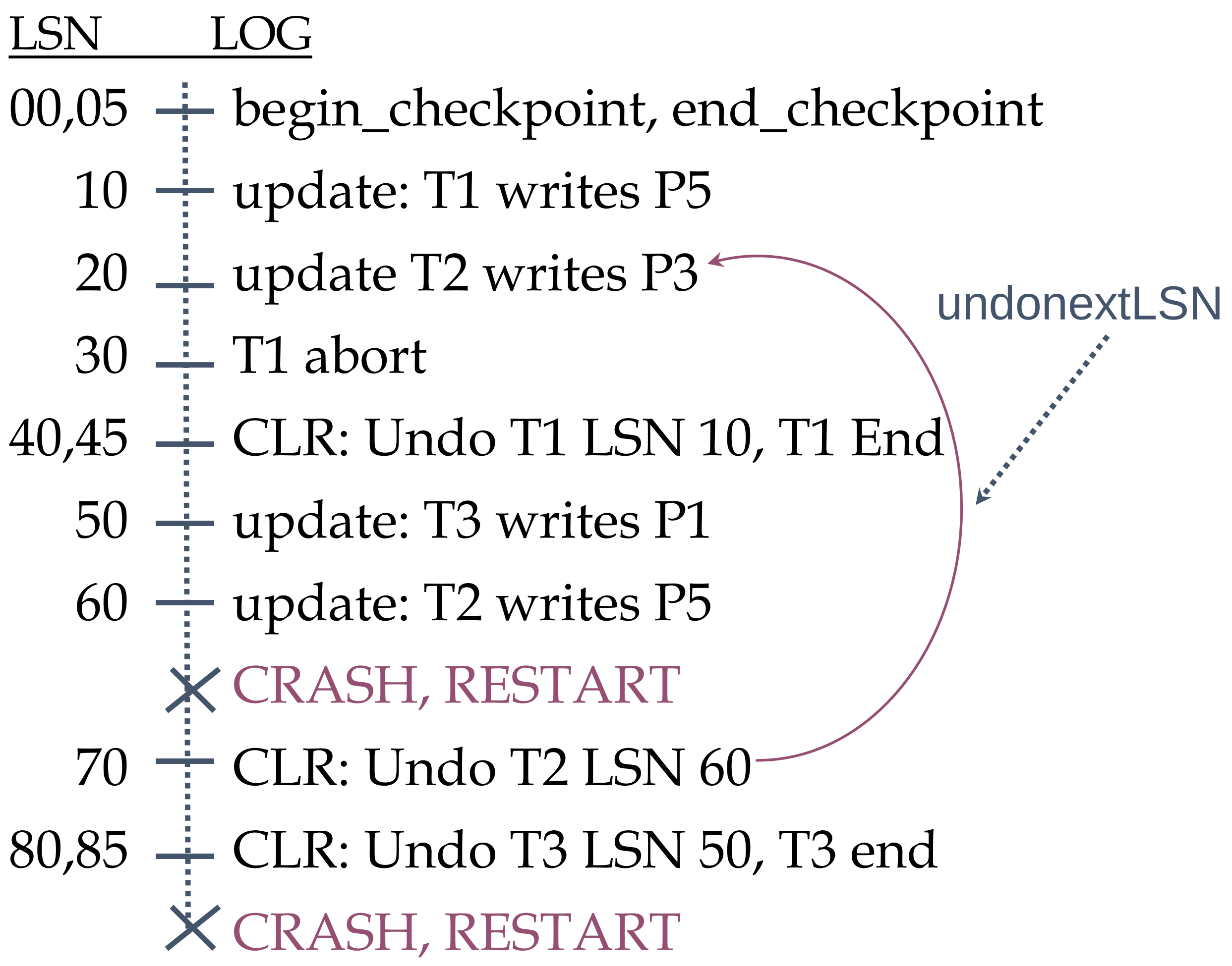
Example 2

- 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

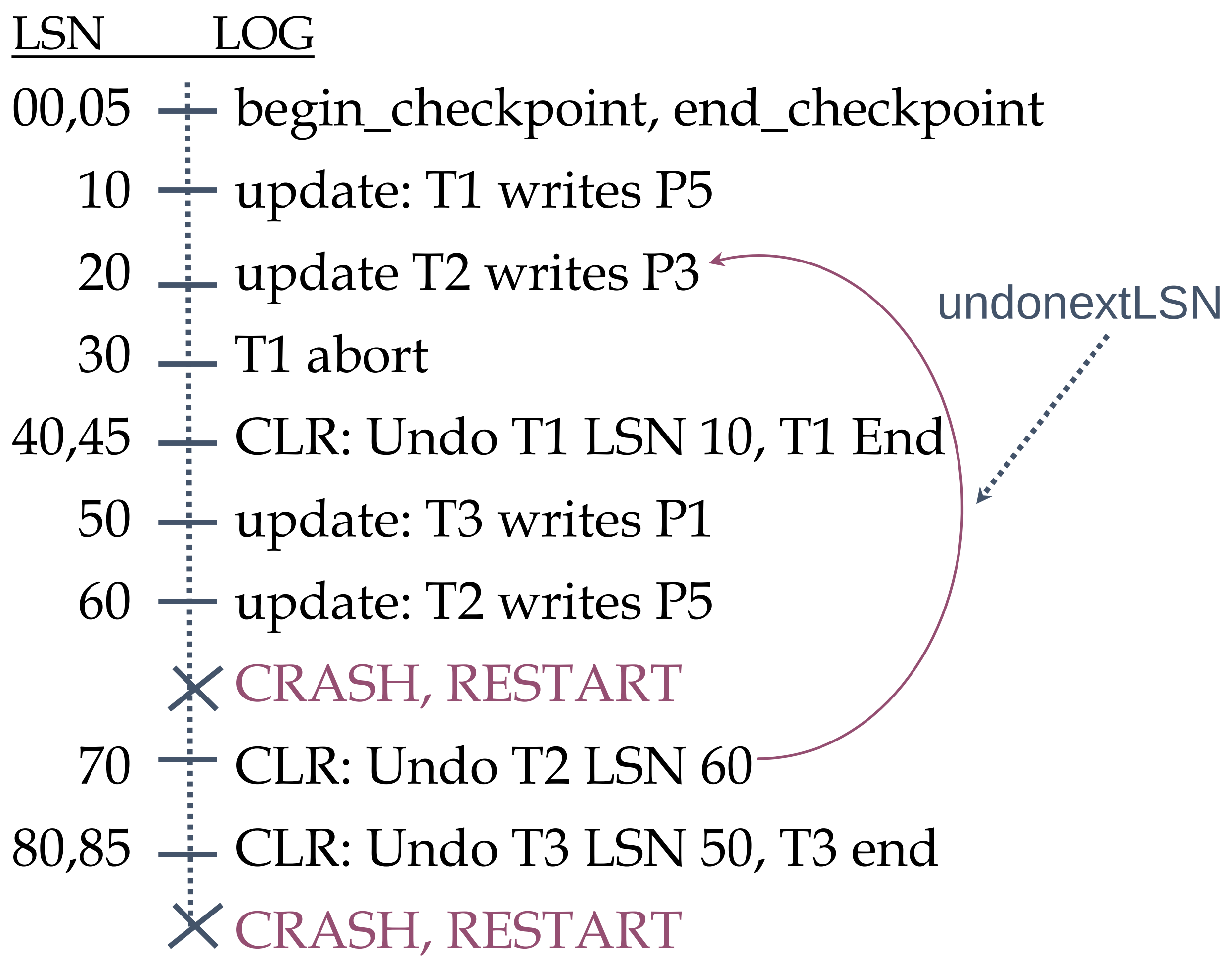
- 1st crash:
  - Undo:
    - T2, T3 – loser transactions => ToUndo = {60, 50}
    - process log record with LSN 60:
      - undo update
      - write CLR (LSN 70) with undoNextLSN 20 (i.e., the next log record that should be processed for T2)



LSN	LOG
00,05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update T2 writes P3
30	T1 abort
40,45	CLR: Undo T1 LSN 10, T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80,85	CLR: Undo T3 LSN 50, T3 end
	CRASH, RESTART

Example 2

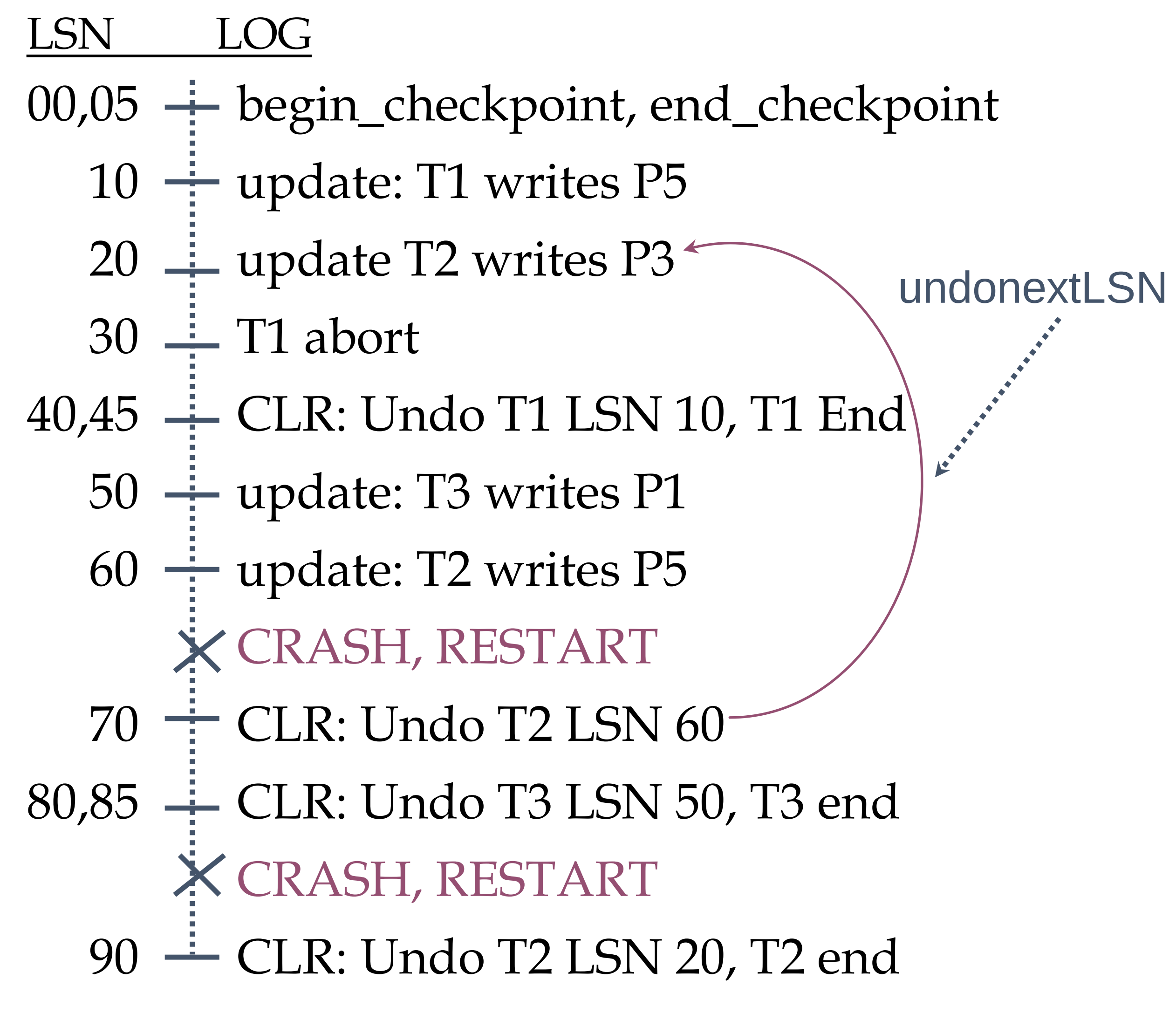
- 1st crash:
  - Undo:
    - process log record with LSN 50:
      - undo update
      - write CLR (LSN 80) with undoNextLSN *null* (i.e., T3 completely undone, write end log record for T3)
  - log records with LSN 70, 80, 85 are written to stable storage
- 2nd crash (during undo)!



LSN	LOG
00,05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update T2 writes P3
30	T1 abort
40,45	CLR: Undo T1 LSN 10, T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80,85	CLR: Undo T3 LSN 50, T3 end
	CRASH, RESTART

Example 2

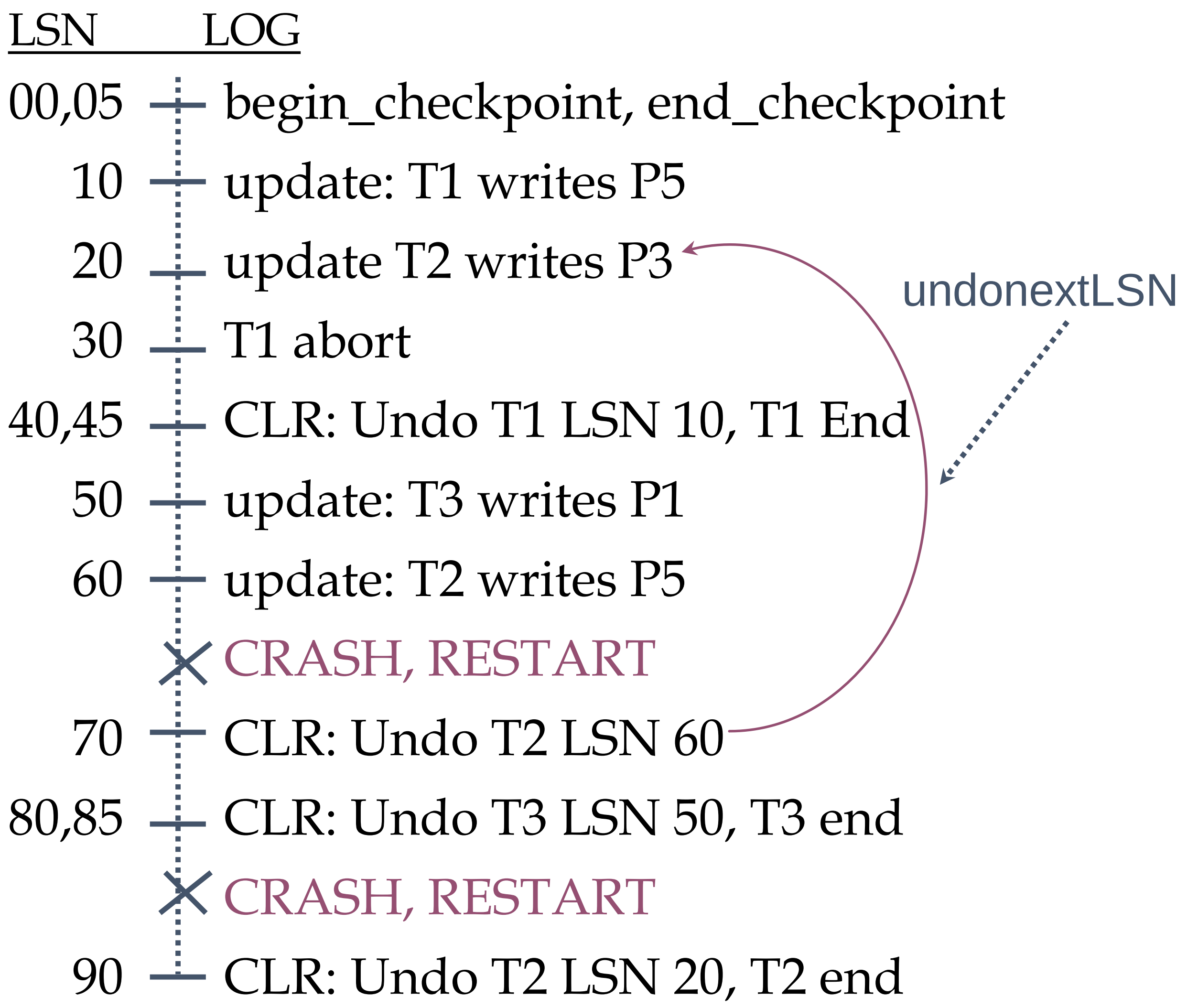
- 2nd crash:
  - Analysis:
    - the only active transaction: T2
    - dirty pages: P5 (recLSN 10), P3 (recLSN 20), P1 (recLSN 50)
  - Redo:
    - process log records with LSN between 10 and 85



LSN	LOG
00,05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update T2 writes P3
30	T1 abort
40,45	CLR: Undo T1 LSN 10, T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80,85	CLR: Undo T3 LSN 50, T3 end
	CRASH, RESTART
90	CLR: Undo T2 LSN 20, T2 end

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

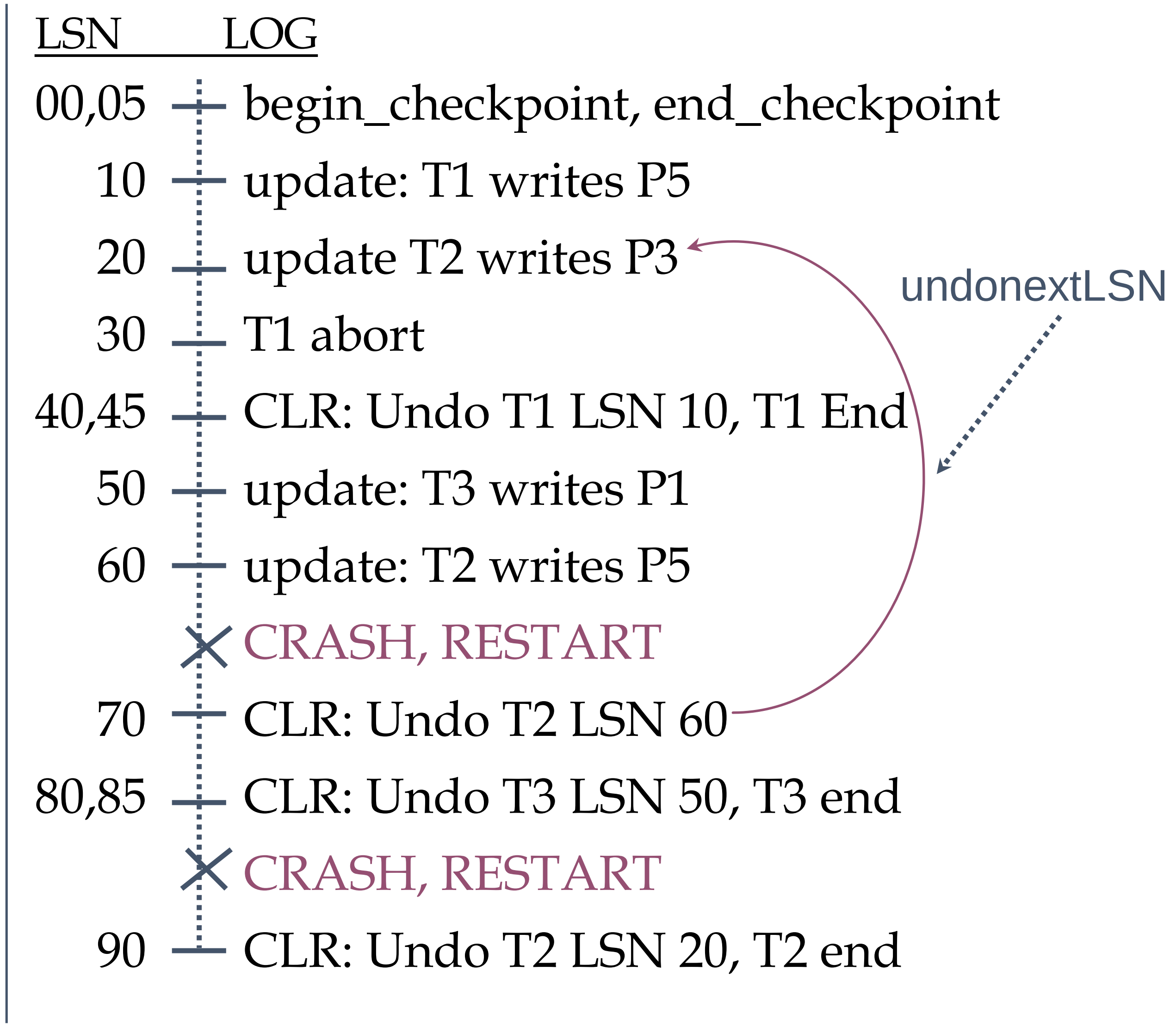


LSN	LOG
00,05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update T2 writes P3
30	T1 abort
40,45	CLR: Undo T1 LSN 10, T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80,85	CLR: Undo T3 LSN 50, T3 end
	CRASH, RESTART
90	CLR: Undo T2 LSN 20, T2 end

Example 2

- 2nd crash:
  - Undo:
    - ToUndo empty

=> recovery complete!



LSN	LOG
00,05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update T2 writes P3
30	T1 abort
40,45	CLR: Undo T1 LSN 10, T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80,85	CLR: Undo T3 LSN 50, T3 end
	CRASH, RESTART
90	CLR: Undo T2 LSN 20, T2 end

- 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

Security

- database protection
    - security
        - protecting the data against unauthorized users (who may want to read, modify, destroy the data)
        - users have the right to do what they are trying to do
    - integrity
        - protecting the data against authorized users
        - the operations that users are trying to execute are correct

- aspects related to security
    - legal, ethical aspects
        - a person searches for a password or accidentally finds such a password and uses it
    - physical controls
        - e.g., the computer room is locked (or guarded)
    - software controls
        - files protection
    - operational problems
        - if a password mechanism is used, how are the passwords kept secret and how often should a user change his / her password?

References

- [Ra00] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (2nd Edition), McGraw-Hill, 2000
- [Le99] LEVENE, M., LOIZOU, G., A Guided Tour of Relational Databases and Beyond, Springer, 1999
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book, Prentice Hall Press, 2008
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Si10] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, McGraw-Hill, 2010, http://codex.cs.yale.edu/avi/db-book/
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems, http://infolab.stanford.edu/~ullman/fcdb.html