Recovery

Failure Handling (Recovery)

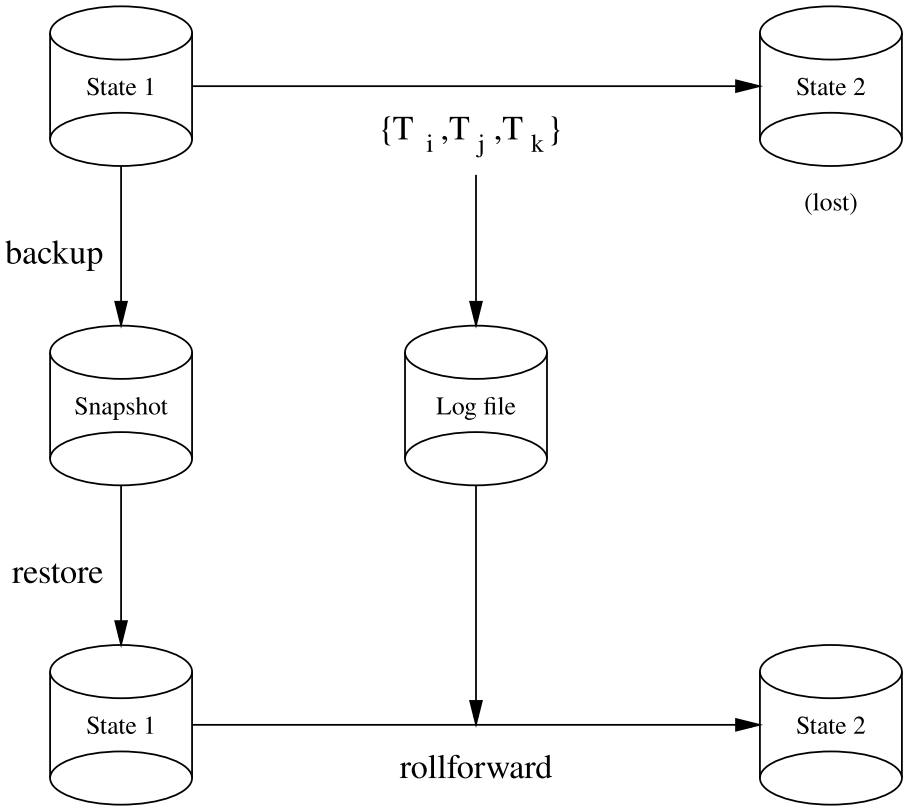
Recovery

- An important task of a DBMS is preventing data loss caused by system crashes
- The two most important mechanisms of recovery are:
  - Backups (snapshots)
  - Log files

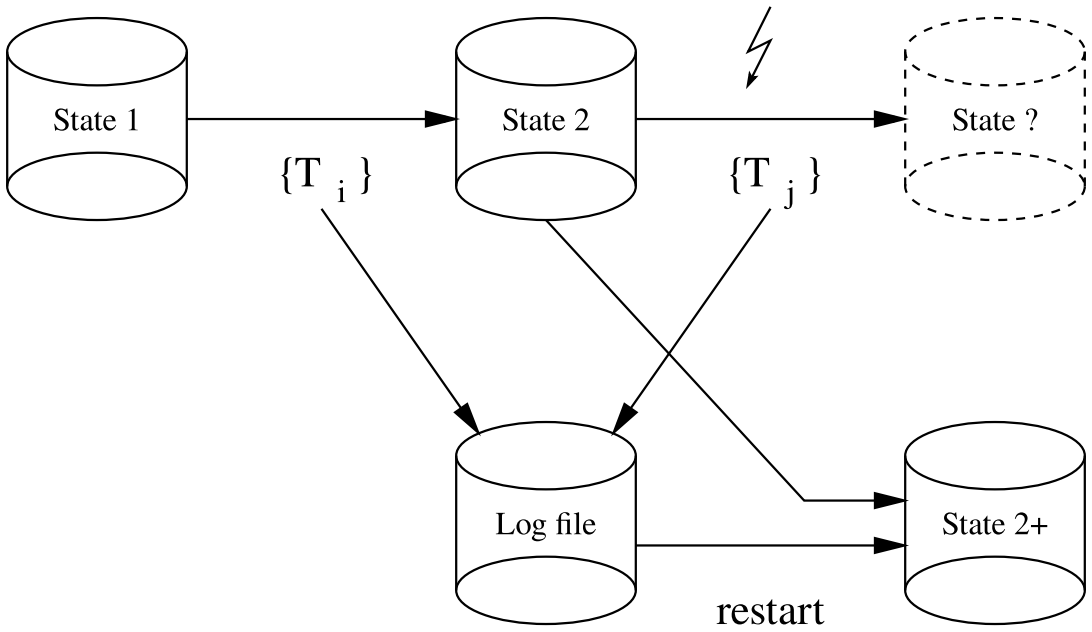
Recovery(2)

- A *backup* is a snapshot of the database contents at a certain point in time
- A *log file* records all changes to the database
- Obviously, backups and log files should not be stored on the same machine ...

Complete Loss



Loss of Main Memory



- Problem: some transactions in $\{T_j\}$ were still active, others had already committed
- Restart restores state 2 + the changes of the committed transactions in $\{T_j\}$

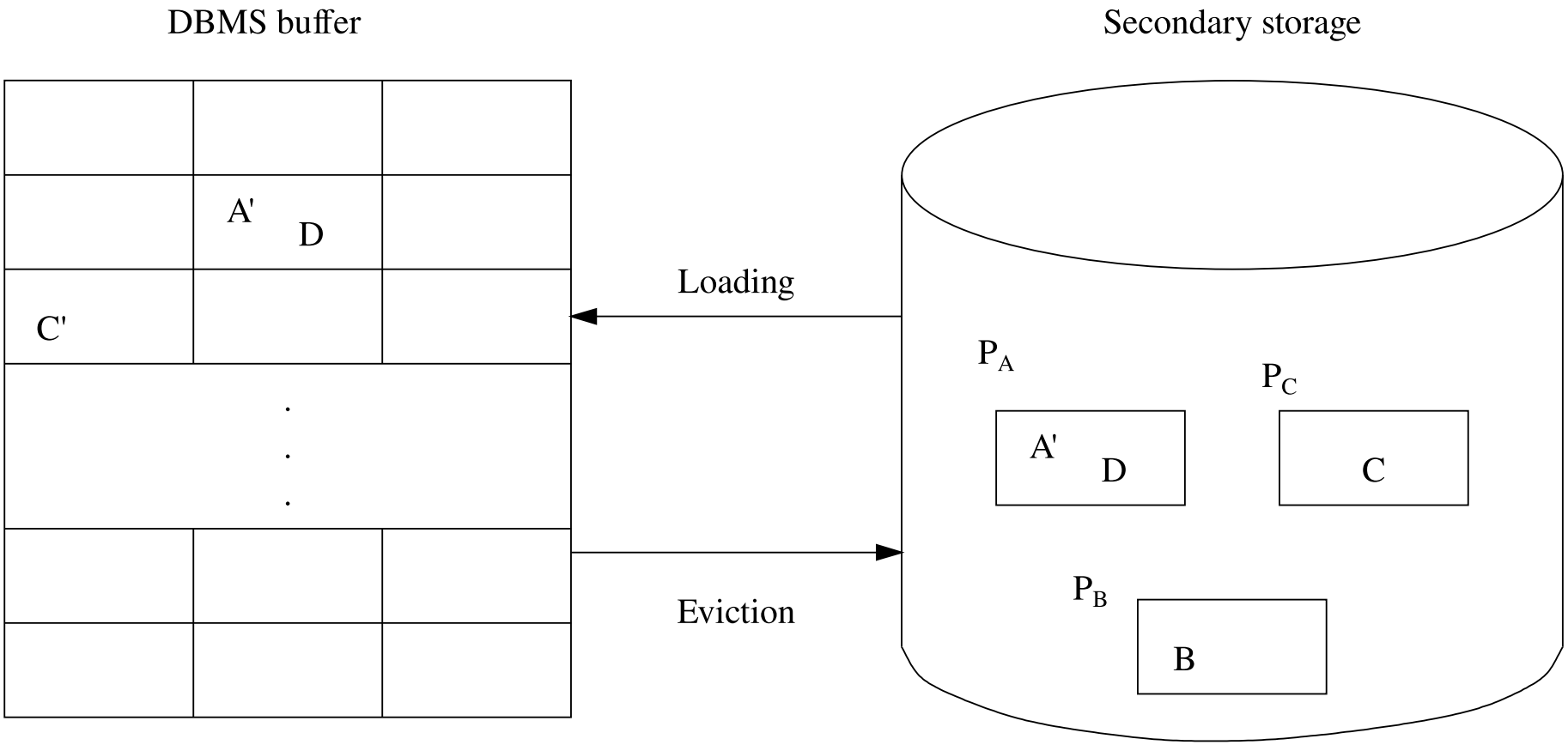
Abort of a Transaction

- Log files can also be used to undo the changes of an aborted transaction
- After this rough overview, we now take a look behind the scenes

Failure Classification

- Local failure in a not yet committed transaction
  - Its effects must be undone
  - $R1$-Recovery
- Failure with loss of main memory
  - completed transactions must be preserved (R2-Recovery)
  - not yet completed transactions must be undone (R3-Recovery)
- Failure with loss of secondary storage
  - R4-Recovery

Storage Hierarchy



Storage Hierarchy(2)

- Replacement strategies for buffer pages
  - ¬*steal*: replacement of pages modified by a still active transaction is not allowed
  - *steal*: every non-fixed page is in principle a candidate for replacement if new pages have to be loaded

Storage Hierarchy(3)

- Propagation of changes of completed transactions
  - *force* strategy: changes are written to secondary storage at the end of the transaction
  - ¬*force* strategy: modified pages can remain in the buffer and be written back later

Impact on Recovery

	force	¬force
¬steal	• no Redo • no Undo	• Redo • no Undo
steal	• no Redo • Undo	• Redo • Undo

Propagation Strategies

- Update in Place
  - every page has exactly one “home” on secondary storage
  - the old state of the page is overwritten
- Twin-block scheme

P_A^0	P_A^1	P_B^0	P_B^1	P_C^0	P_C^1	
						…

- Shadow paging
  - only modified pages are duplicated
  - less redundancy than with the twin-block scheme

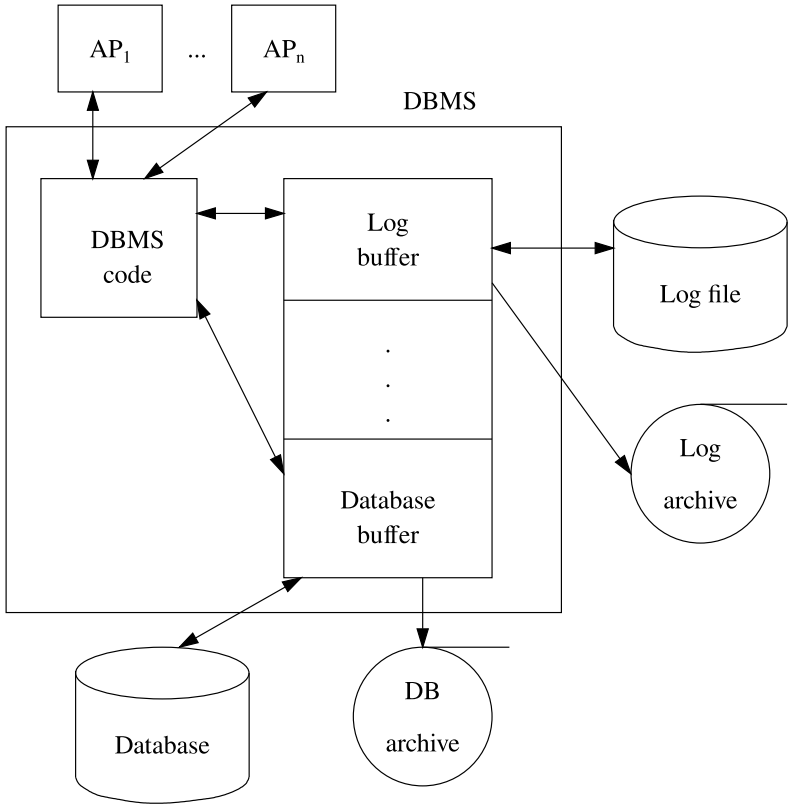
System Configuration

- steal
- ¬force
- update-in-place
- Small lock granularity

ARIES Protocol

- The ARIES protocol is a widely used protocol for recovery in DBMSs
- The log file contains:
  - Redo information: specifies how changes can be redone
  - Undo information: describes how changes can be undone

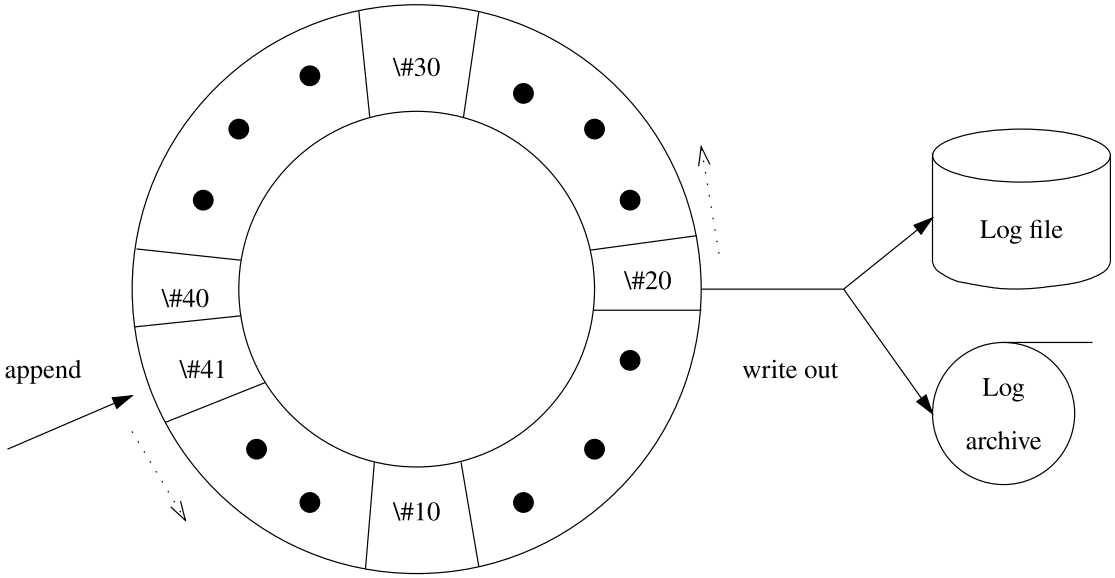
Writing the Log Data



- The log information is written twice
  - Log file for fast access: R1, R2, and R3 recovery
  - Log archive: R4 recovery

Writing the Log Data(2)

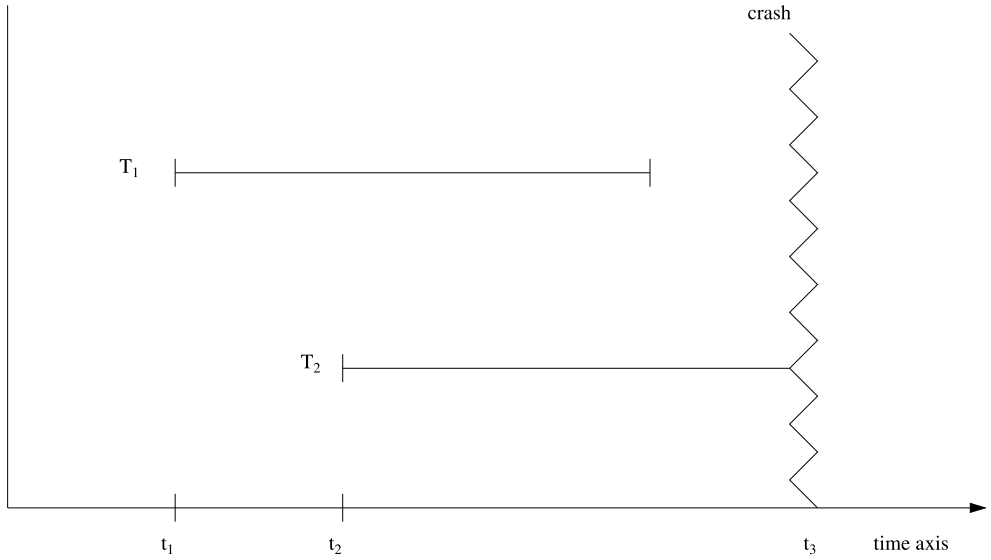
- Organization of the log ring buffer:



Writing the Log Data(3)

- **W**rite **A**head **L**og principle (WAL principle)
  - Before a transaction is **committed**, all log records "belonging to it" must be written out
  - Before a modified page may be evicted, all log records belonging to this page must be written to the temporary log and to the log archive

Restart after a Failure

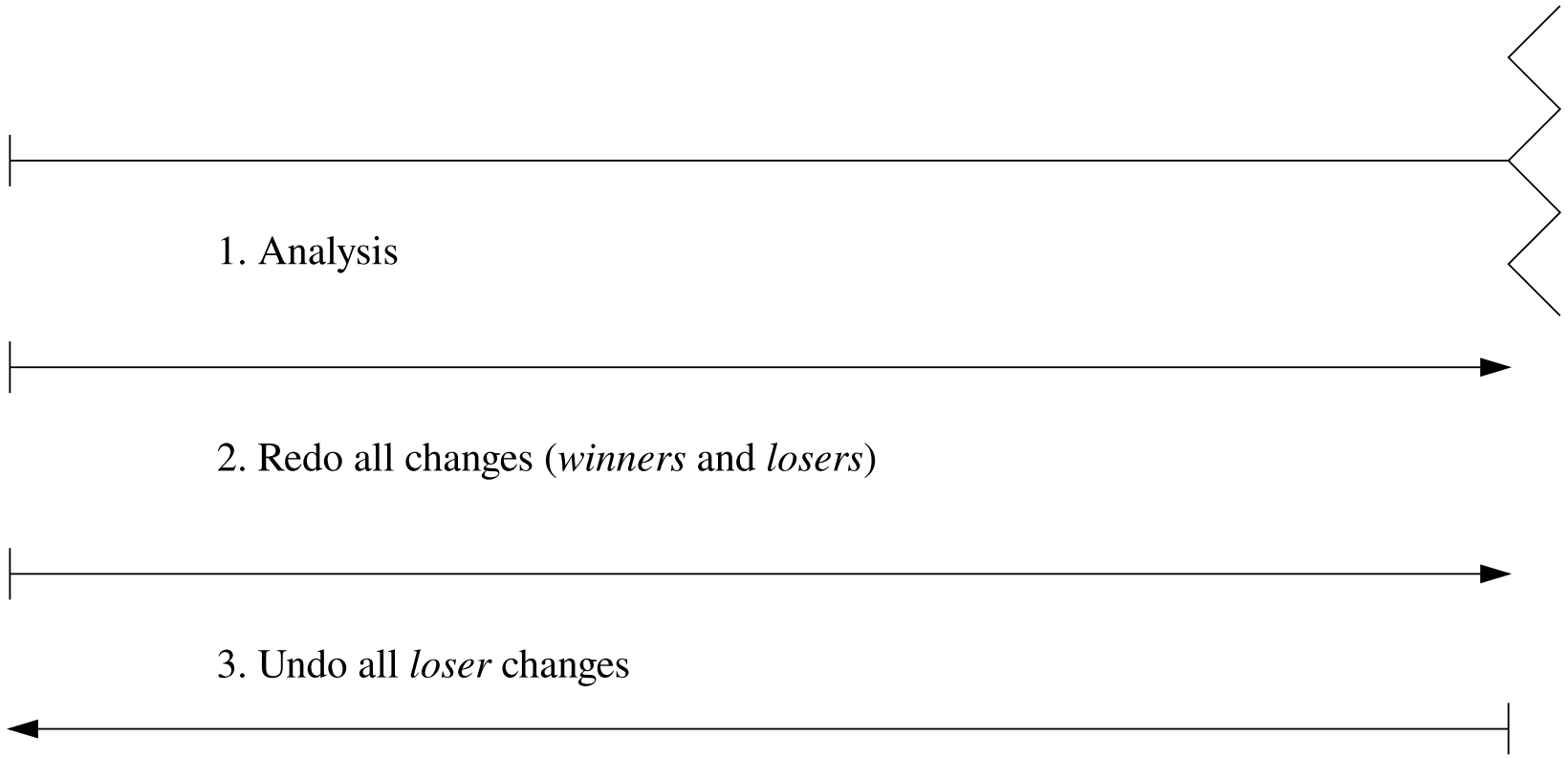


- transactions of type T_1 are *winners*: must be completely redone
- transactions of type T_2 are *losers*: must be undone

Phases of Restart

- *Analysis*:
  - Determine the *winner* set of transactions of type T_1
  - Determine the *loser* set of transactions of type T_2.
- *Repeating history*:
  - *All* logged changes are applied to the database in the order of their execution
- *Undo of the losers*:
  - The update operations of the *loser* transactions are undone in reverse order of their original execution

Phases of Restart(2)



Structure of the Log Records

[LSN,TA,PageID,Redo,Undo,PrevLSN]

- Redo:
  - Physical logging: after-image
  - Logical logging: code that produces the after-image from the before-image
- Undo:
  - Physical logging: before-image
  - Logical logging: code that produces the before-image from the after-image

Structure of the Log Records(2)

- *LSN* (*Log Sequence Number*),
  - a unique identifier of the log record
  - *LSNs* must be assigned in monotonically increasing order,
  - this allows determining the chronological order of the log records
- *TA*
  - transaction identifier of the transaction that performed the change

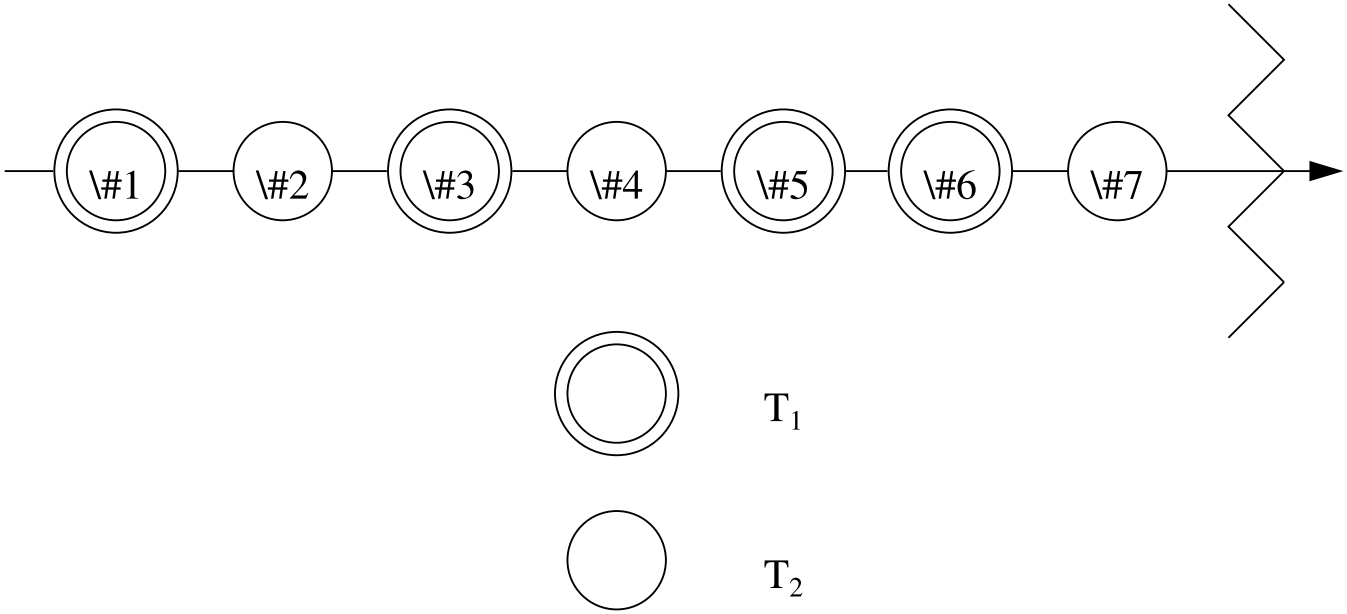
Structure of the Log Records(3)

- *PageID*
  - the identifier of the page on which the update operation was performed
  - If a change affects more than one page, a corresponding number of log records must be generated
- *PrevLSN*,
  - pointer to the previous log record of the respective transaction
  - This entry is needed for efficiency reasons

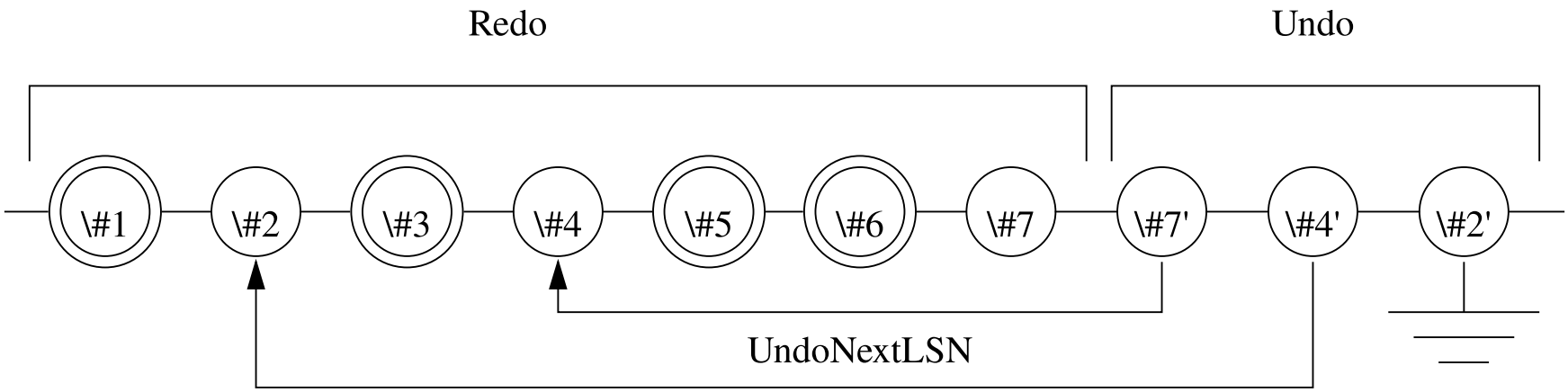
Example of a Log File

	T_1	T_2	Log
1.	**BOT**		$[\#1, T_1, \textbf{BOT}, 0]$
2.	$r(A, a_1)$		
3.		**BOT**	$[\#2, T_2, \textbf{BOT}, 0]$
4.		$r(C, c_2)$	
5.	$a_1 := a_1 - 50$		
6.	$w(A, a_1)$		$[\#3, T_1, P_A, A \text{ -= } 50, A \text{ += } 50, \#1]$
7.		$c_2 := c_2 + 100$	
8.		$w(C, c_2)$	$[\#4, T_2, P_C, C \text{ += } 100, C \text{ -= } 100, \#2]$
9.	$r(B, b_1)$		
10.	$b_1 := b_1 + 50$		
11.	$w(B, b_1)$		$[\#5, T_1, P_B, B \text{ += } 50, B \text{ -= } 50, \#3]$
12.	**commit**		$[\#6, T_1, \textbf{commit}, \#5]$
13.		$r(A, a_2)$	
14.		$a_2 := a_2 - 100$	
15.		$w(A, a_2)$	$[\#7, T_2, P_A, A \text{ -= } 100, A \text{ += } 100, \#4]$
16.		**commit**	$[\#8, T_2, \textbf{commit}, \#7]$

$$undo(undo(\cdots(undo(a))\cdots)) \quad undo(a)$$
$$redo(redo(\cdots(redo(a))\cdots)) \quad redo(a)$$



Fault-Tolerant Restart(2)



- Compensation log records (CLR) for undone changes.
- #7′ is the CLR for #7
- #4′ is the CLR for #4

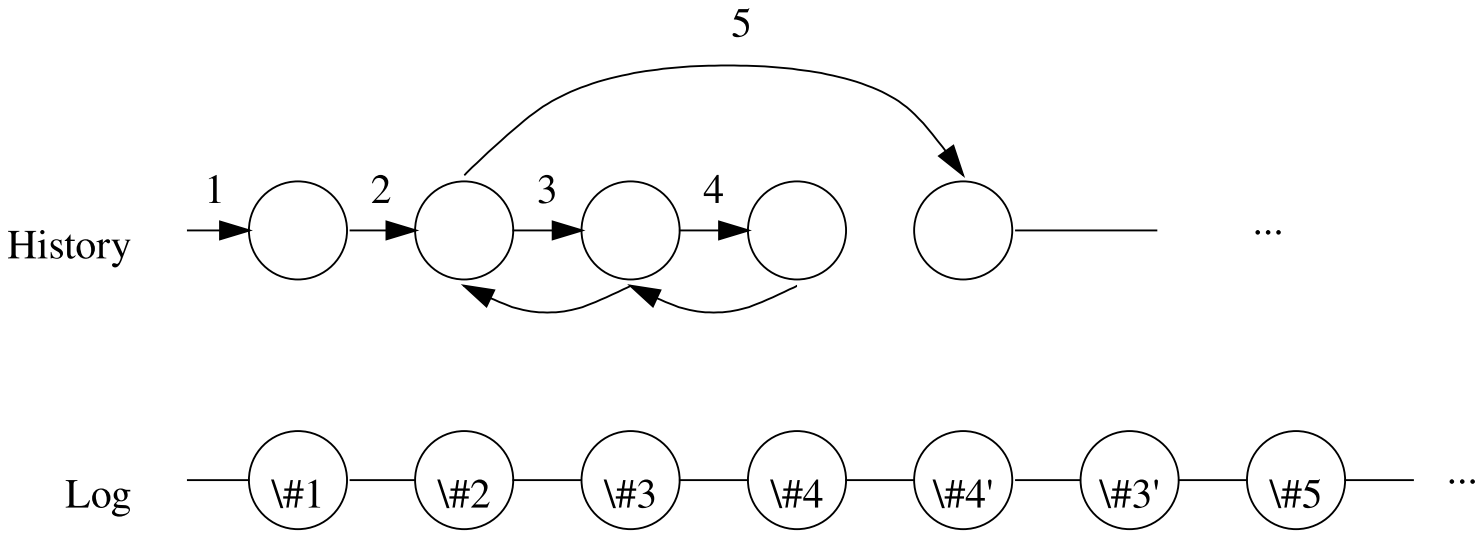
Log Records after Restart

$$[\#1, T_1, \textbf{\textit{BOT}}, 0]$$
$$[\#2, T_2, \textbf{\textit{BOT}}, 0]$$
$$[\#3, T_1, P_A, A\text{--}{=}50, A{+}{=}50, \#1]$$
$$[\#4, T_2, P_C, C{+}{=}100, C\text{--}{=}100, \#2]$$
$$[\#5, T_1, P_B, B{+}{=}50, B\text{--}{=}50, \#3]$$
$$[\#6, T_1, \textbf{\textit{commit}}, \#5]$$
$$[\#7, T_2, P_A, A\text{--}{=}100, A{+}{=}100, \#4]$$
$$angle.l\#7', T_2, P_A, A{+}{=}100, \#7, \#4 angle.r$$
$$angle.l\#4', T_2, P_C, C\text{--}{=}100, \#7', \#2 angle.r$$
$$angle.l\#2', T_2, -, -, \#4', 0 angle.r$$

- CLRs are marked by angle brackets $angle.l \ldots angle.r$

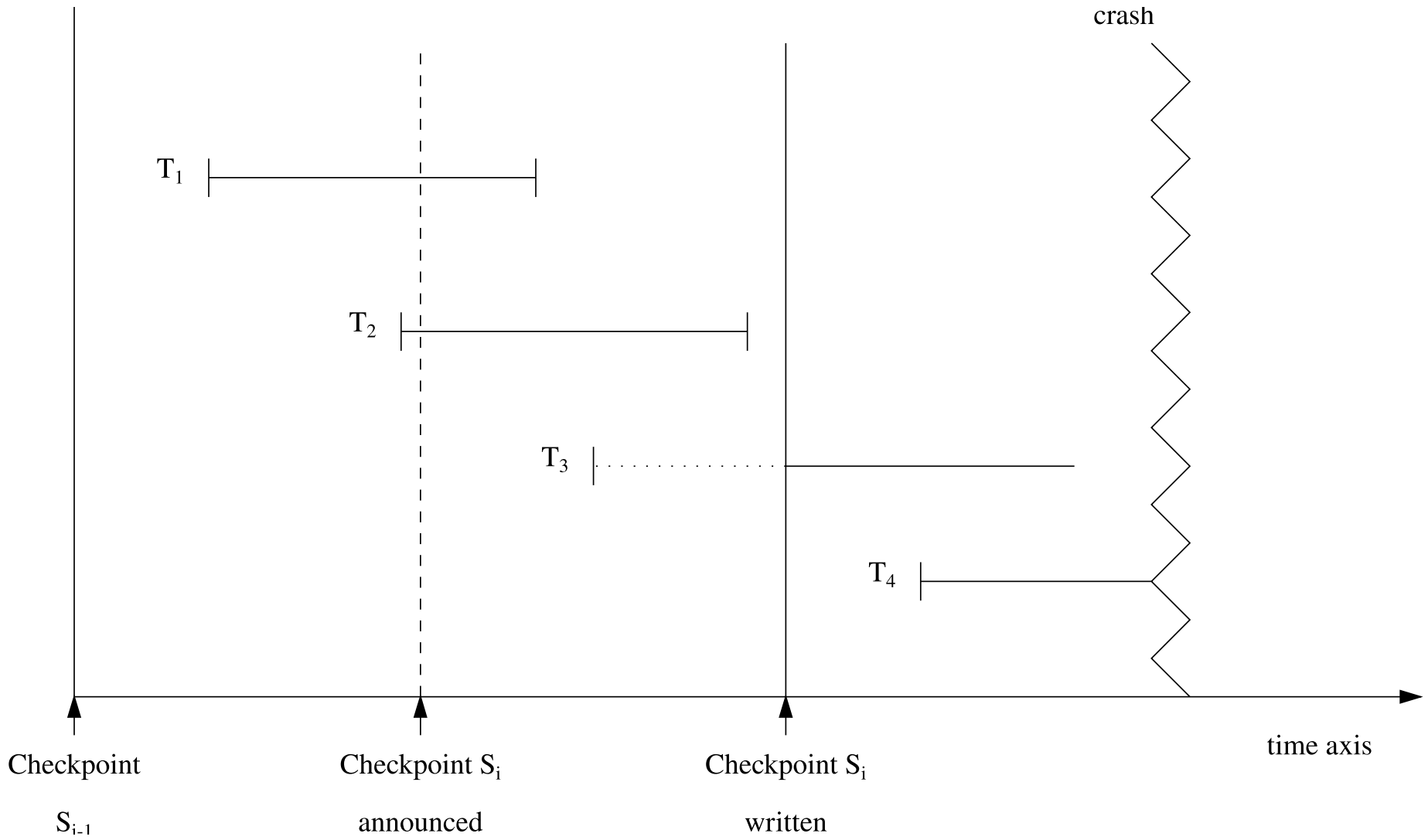
- the structure of a CLR is as follows
  - LSN
  - transaction identifier
  - affected page
  - Redo information
  - PrevLSN
  - UndoNxtLSN (reference to the next change to be undone)

Partial Rollback



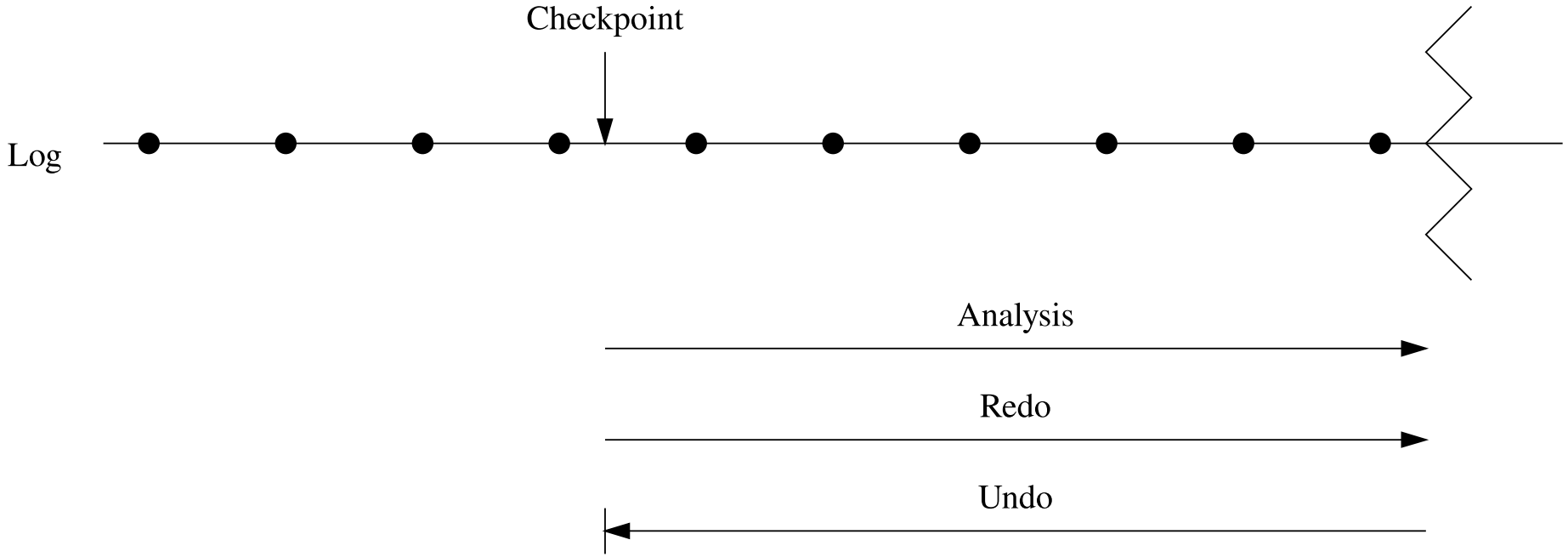
- Steps 3 and 4 are undone
- necessary for implementing savepoints within a transaction

Checkpoints



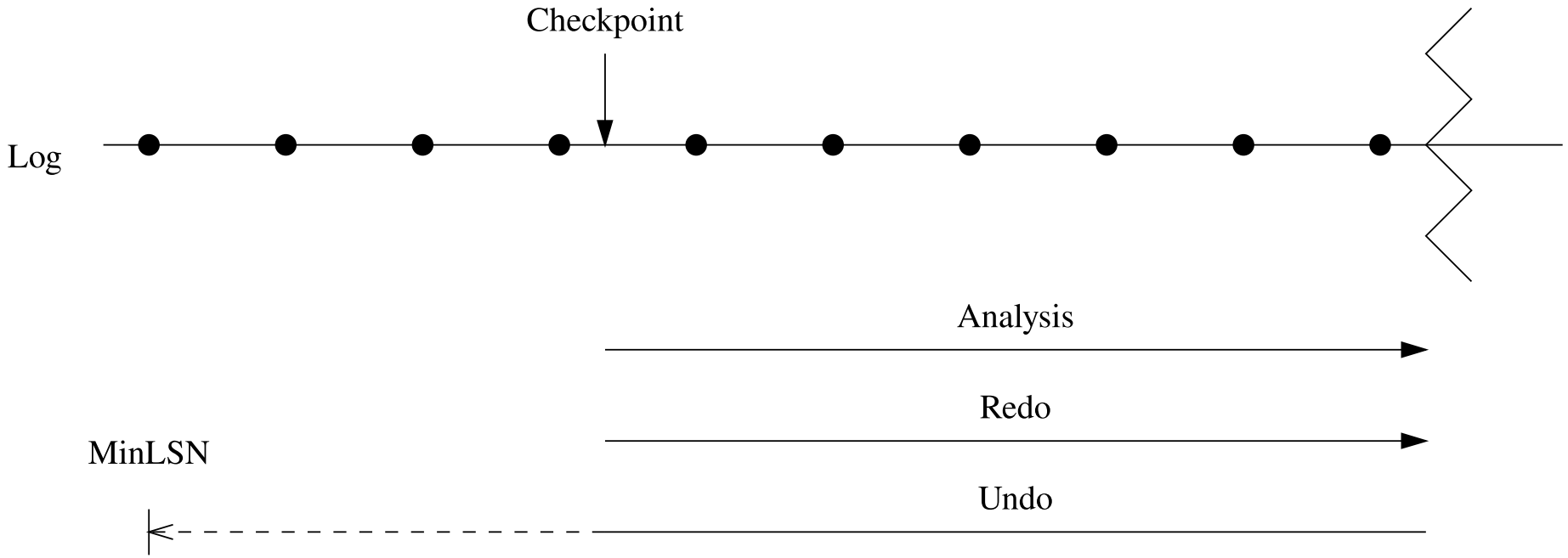
Kinds of Checkpoints

- Transaction-consistent:

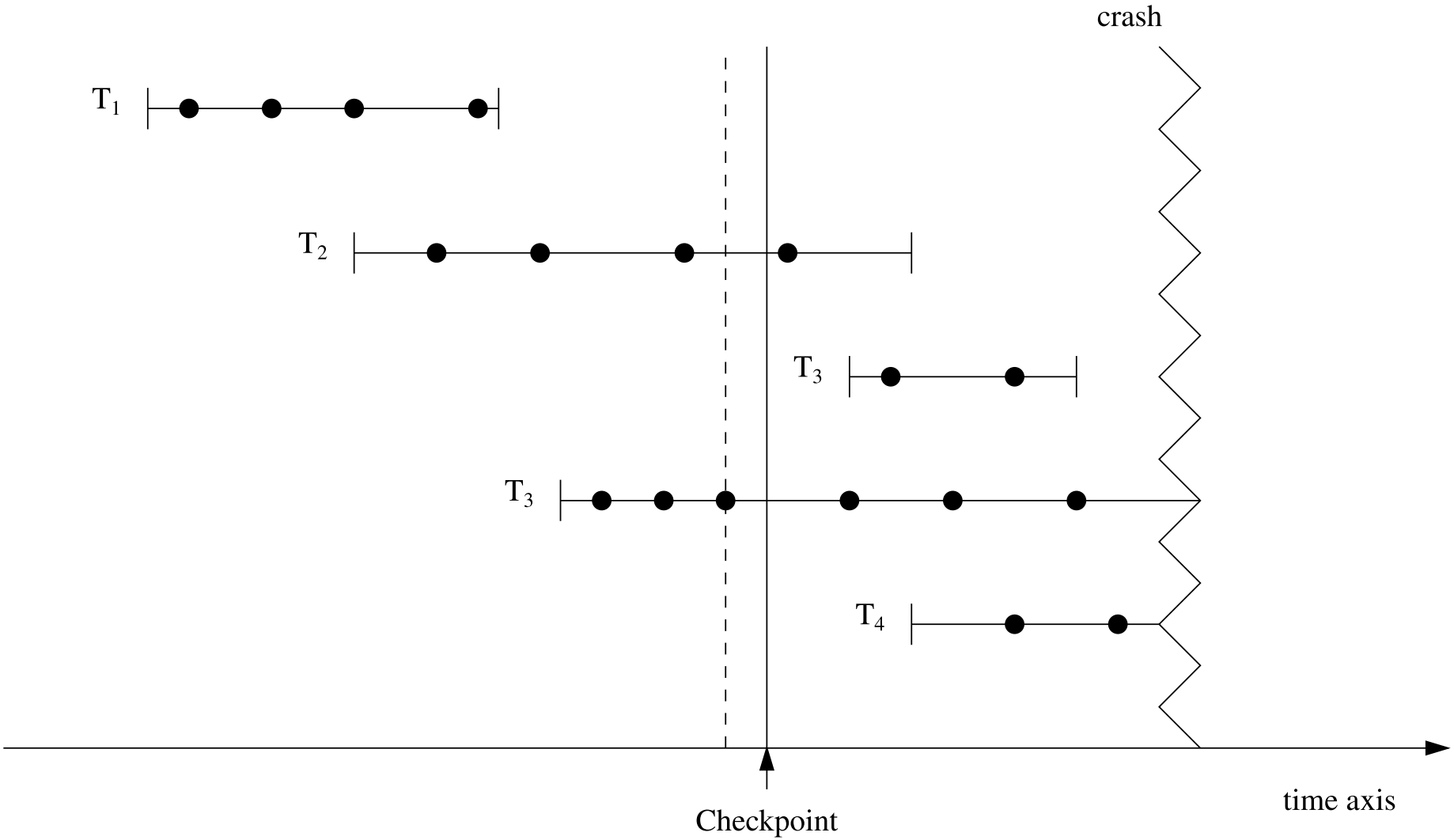


Kinds of Checkpoints(2)

- Action-consistent:

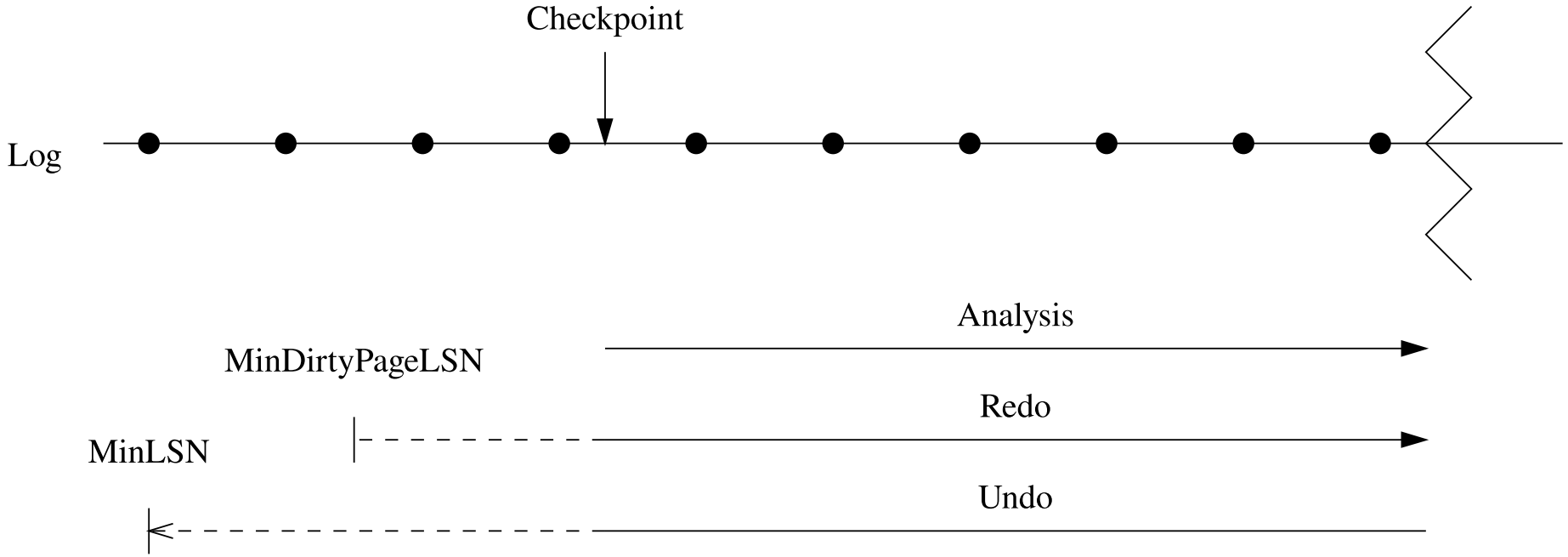


Action-Consistent Checkpoint



Kinds of Checkpoints(3)

- Fuzzy:



Fuzzy Checkpoint

- modified pages are not written out
- only their identifiers are written out
  - *Dirty Pages*=set of modified pages
- *MinDirtyPageLSN*: the minimal LSN whose changes have not yet been written out

Summary

- Failure handling (recovery) can restore the state of the database at the time of the crash
- The two main mechanisms for this are:
  - Log files
  - Backups and checkpoints