

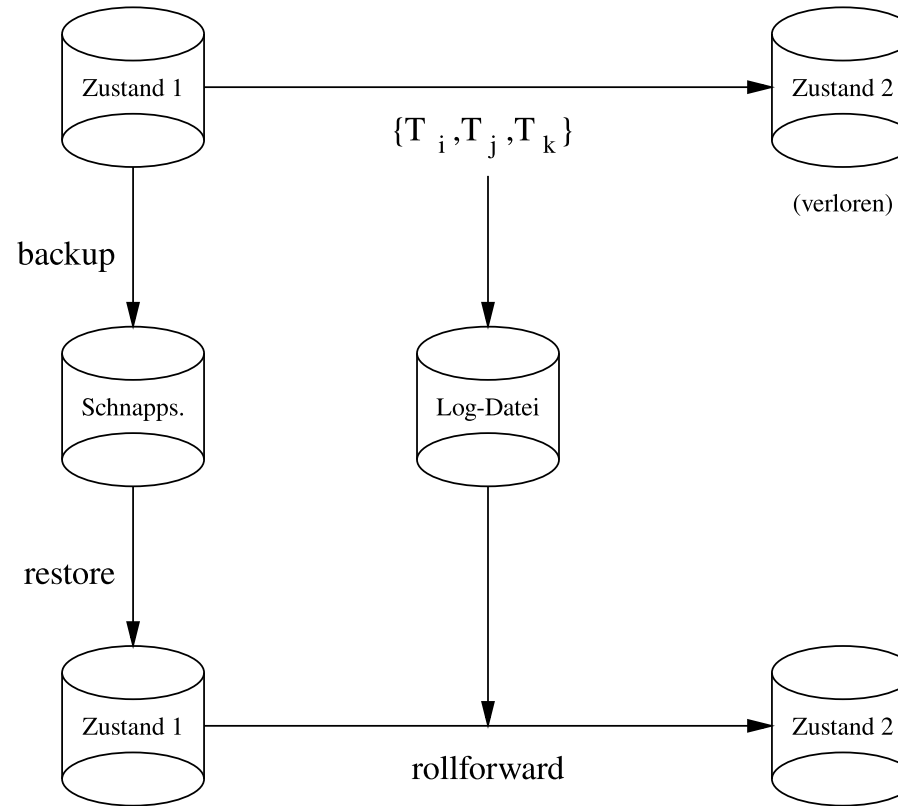
Recovery

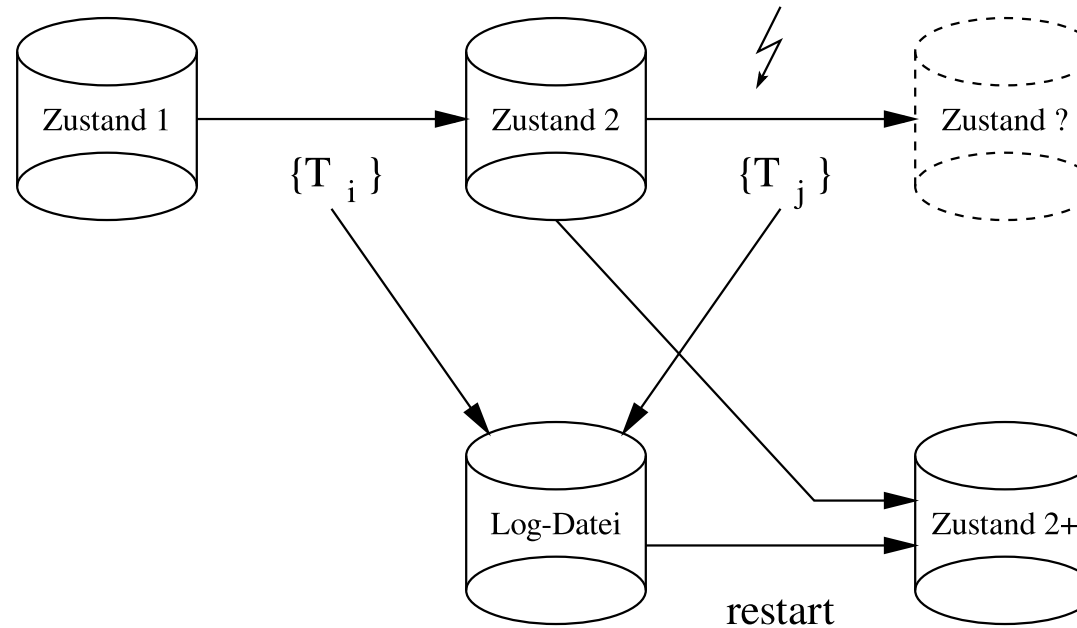


Fehlerbehandlung (Recovery)

- Wichtige Aufgabe eines DBMS ist das Verhindern von Datenverlust durch Systemabstürze
- Die zwei wichtigsten Mechanismen des Recovery sind:
 - Sicherungspunkte (Backups)
 - Log-Dateien

- Ein *Sicherungspunkt* ist ein Schnappschuß des Datenbankinhalts zu einem bestimmten Zeitpunkt
- In einer *Log-Datei* werden alle Änderungen an der Datenbasis mitprotokolliert
- Offensichtlich sollten Sicherungspunkte und Log-Dateien nicht auf der gleichen Maschine gespeichert werden ...

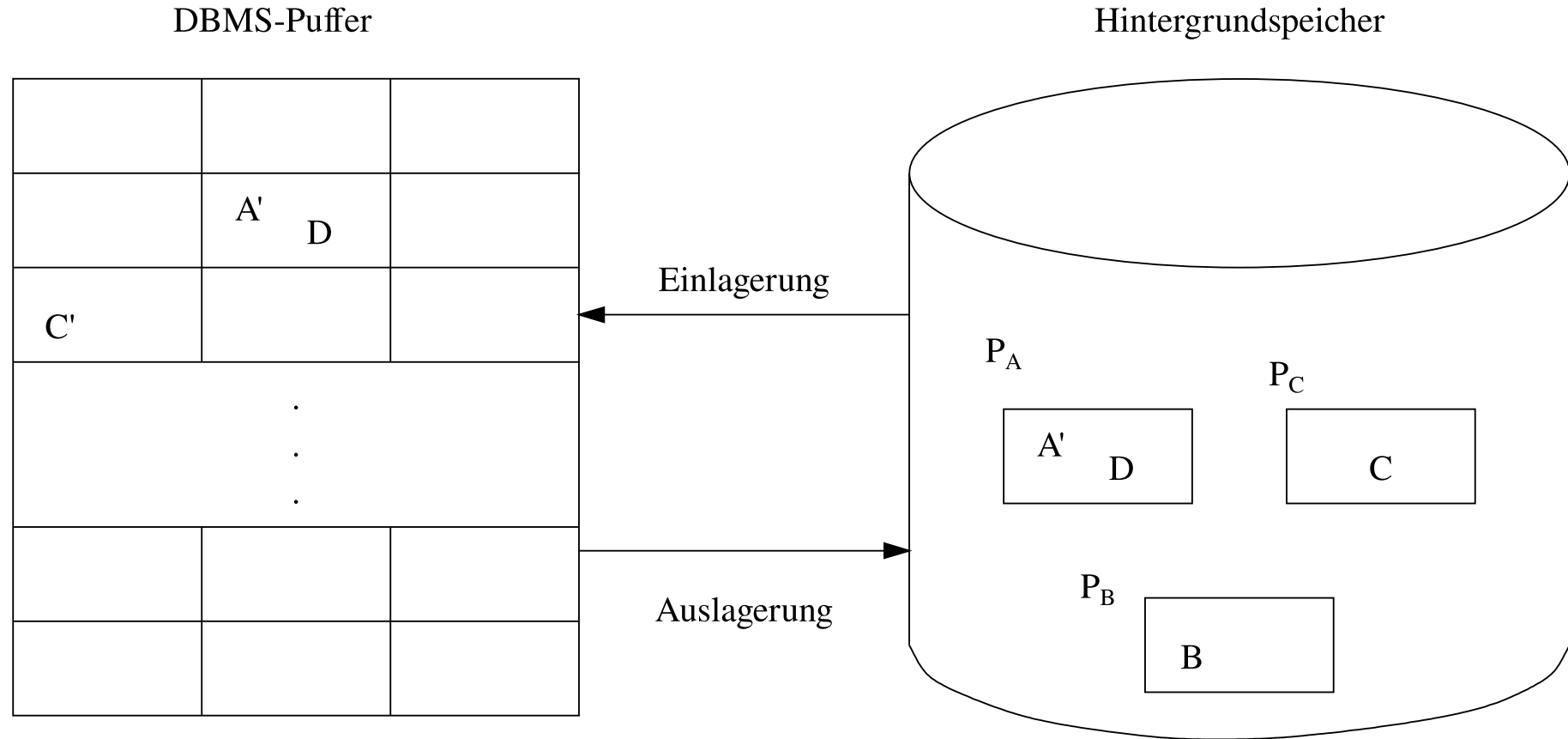




- Problem: einige TAs in $\{T_j\}$ waren noch aktiv, andere haben schon committet
- Restart stellt Zustand 2 + die Änderungen der committeten TAs in $\{T_j\}$ wieder her

- Log-Dateien können auch dazu verwendet werden die Änderungen einer abgebrochenen TA zurückzusetzen
- Nach diesem groben Überblick werfen wir jetzt einen Blick hinter die Kulissen

- Lokaler Fehler in einer noch nicht festgeschriebenen (committed) Transaktion
 - Wirkung muss zurückgesetzt werden
 - *R1-Recovery*
- Fehler mit Hauptspeicherverlust
 - abgeschlossene TAs müssen erhalten bleiben (*R2-Recovery*)
 - noch nicht abgeschlossene TAs müssen zurückgesetzt werden (*R3-Recovery*)
- Fehler mit Hintergrundspeicherverlust
 - *R4-Recovery*



- Ersetzungsstrategien von Puffer-Seiten
 - $\neg steal$: Ersetzung von Seiten, die von einer noch aktiven Transaktion modifiziert wurden, ausgeschlossen
 - *steal*: Jede nicht fixierte Seite ist prinzipiell ein Kandidat für die Ersetzung, falls neue Seiten eingelagert werden müssen

- Einbringen von Änderungen abgeschlossener TAs
 - *force*-Strategie: Änderungen werden zum Transaktionsende auf den Hintergrundspeicher geschrieben
 - $\neg$ *force*-Strategie: geänderte Seiten können im Puffer verbleiben und später zurückgeschrieben werden

	force	$\neg$ force
$\neg$ steal	• kein Redo • kein Undo	• Redo • kein Undo
steal	• kein Redo • Undo	• Redo • Undo

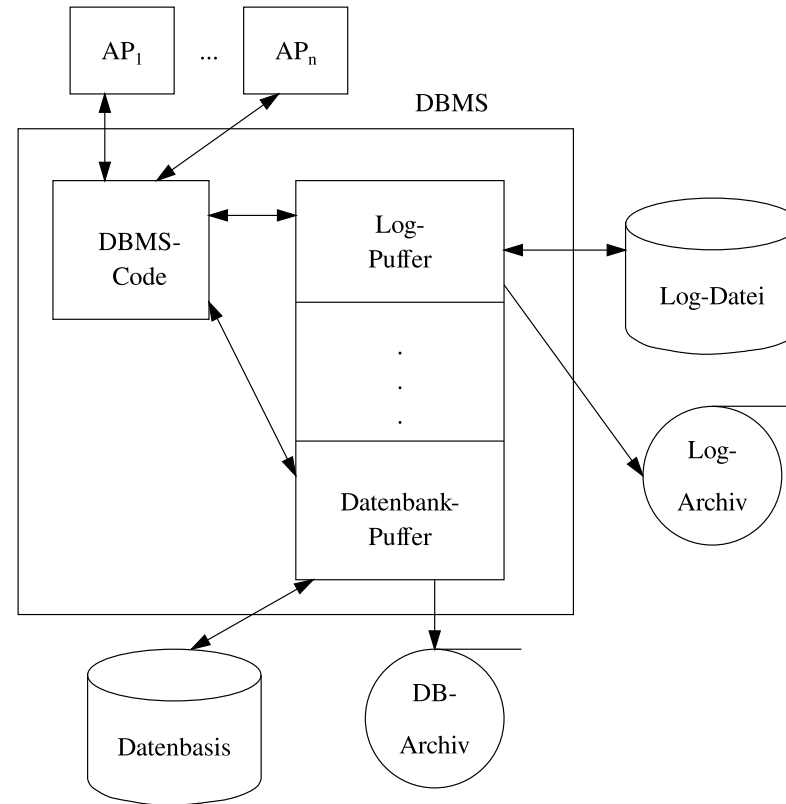
- Update in Place
 - jede Seite hat genau eine “Heimat” auf dem Hintergrundspeicher
 - der alte Zustand der Seite wird überschrieben
- Twin-Block-Verfahren

P_A^0	P_A^1	P_B^0	P_B^1	P_C^0	P_C^1	
						...

- Schattenspeicherkonzept
 - nur geänderte Seiten werden dupliziert
 - weniger Redundanz als beim Twin-Block-Verfahren

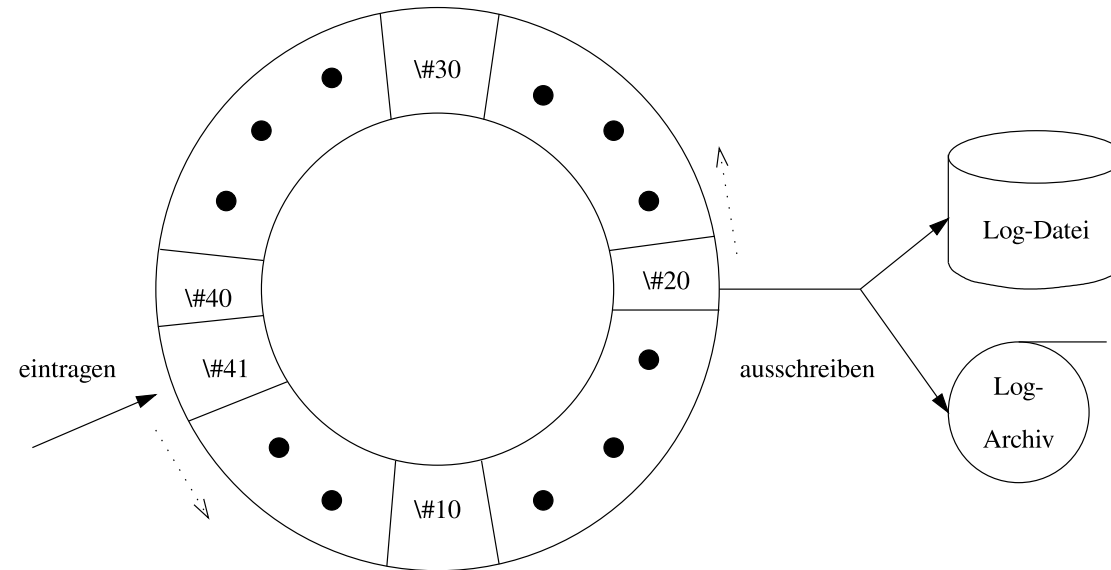
- steal
- $\neg$ force
- update-in-place
- Kleine Sperrgranulate

- ARIES-Protokoll ist weit verbreitetes Protokoll zur Fehlerbehandlung in DBMSen
- Log-Datei enthält:
 - Redo-Information: gibt an, wie Änderungen nachvollzogen werden können
 - Undo-Information: beschreibt, wie Änderungen rückgängig gemacht werden können

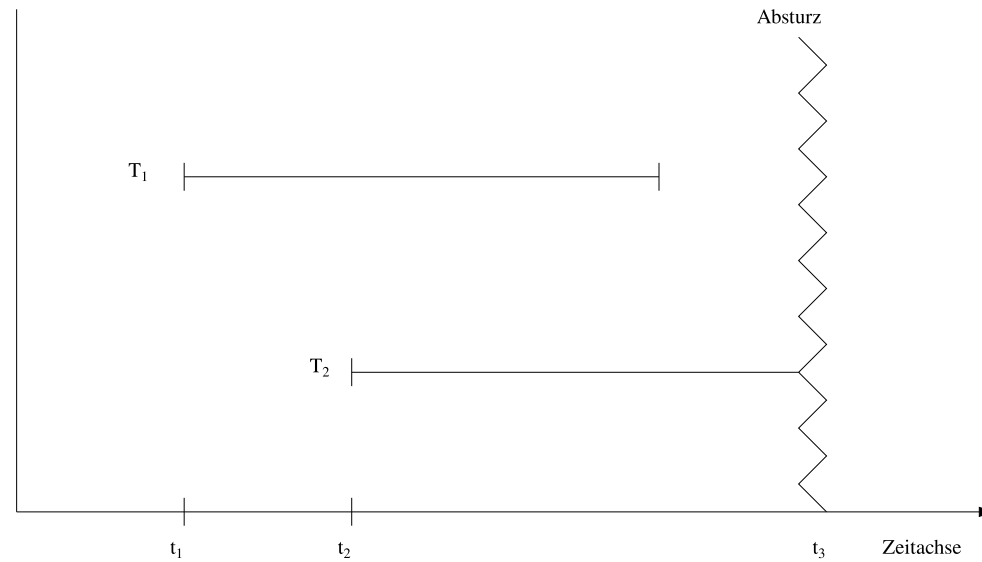


- Die Log-Information wird zweimal geschrieben
 - Log-Datei für schnellen Zugriff: R1, R2 und R3-Recovery
 - Log-Archiv: R4-Recovery

- Anordnung des Log-Ringpuffers:



- **W**rite **A**head **L**og-Prinzip (WAL-Prinzip)
 - Bevor eine Transaktion festgeschrieben (**committed**) wird, müssen alle “zu ihr gehörenden” Log-Einträge ausgeschrieben werden
 - Bevor eine modifizierte Seite ausgelagert werden darf, müssen alle Log-Einträge, die zu dieser Seite gehören, in das temporäre und das Log-Archiv ausgeschrieben werden



- TAs der Art T_1 sind *Winner*: müssen vollständig nachvollzogen werden
- TAs der Art T_2 sind *Loser*: müssen rückgängig gemacht werden

- *Analyse:*
 - Ermittlung der *Winner*-Menge von Transaktionen des Typs T_1
 - Ermittlung der *Loser*-Menge von Transaktionen der Art T_2 .
- *Wiederholung der Historie:*
 - *Alle* protokollierten Änderungen werden in der Reihenfolge ihrer Ausführung in die Datenbasis eingebracht
- *Undo der Loser:*
 - Die Änderungsoperationen der *Loser*-Transaktionen werden in umgekehrter Reihenfolge ihrer ursprünglichen Ausführung rückgängig gemacht



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

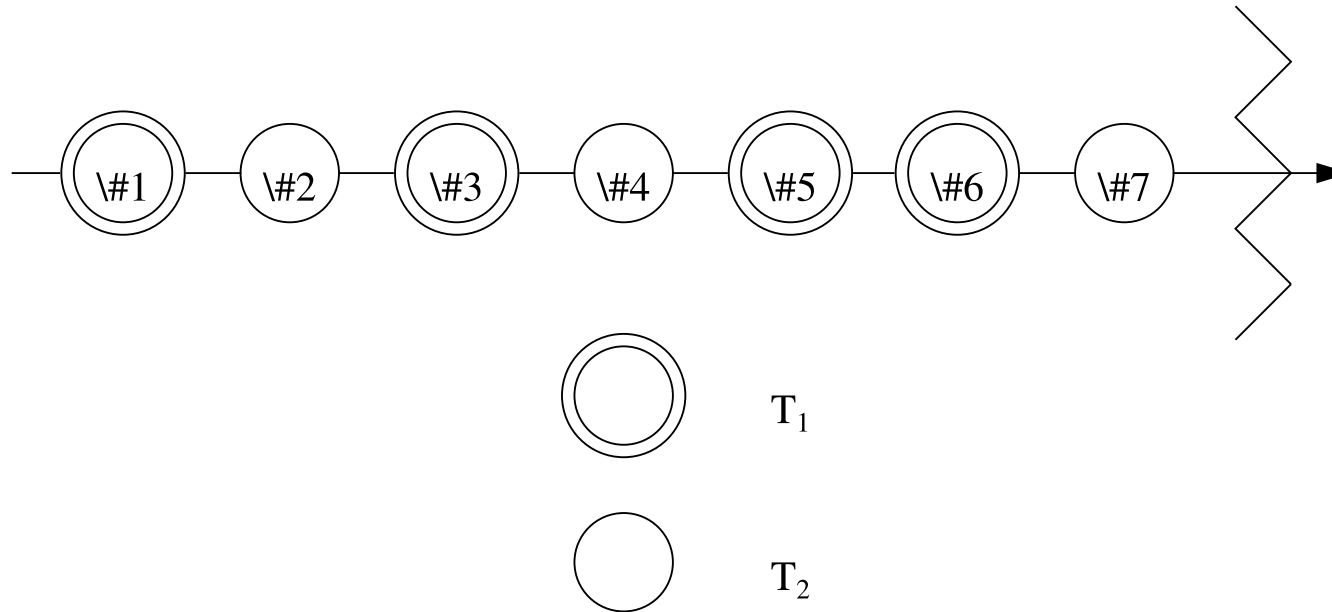
- Redo:
 - Physische Protokollierung: After-Image
 - Logische Protokollierung: Code mit dem aus dem Before-Image das After-Image erzeugt werden kann
- Undo:
 - Physische Protokollierung: Before-Image
 - Logische Protokollierung: Code mit dem aus dem After-Image das Before-Image erzeugt werden kann

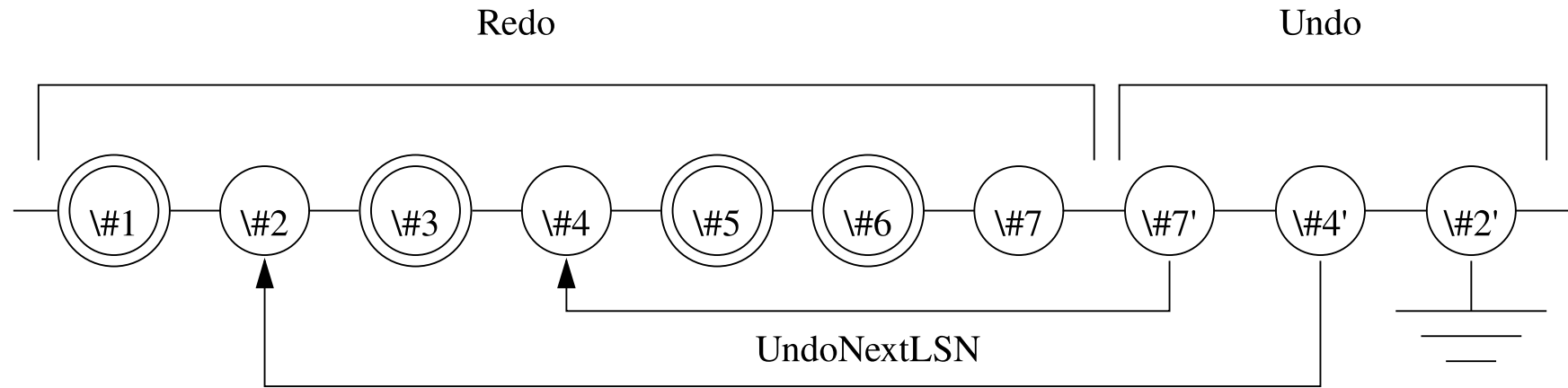
- *LSN (Log Sequence Number)*,
 - eine eindeutige Kennung des Log-Eintrags
 - *LSNs* müssen monoton aufsteigend vergeben werden,
 - die chronologische Reihenfolge der Protokolleinträge kann dadurch ermittelt werden
- *TA*
 - Transaktionskennung der Transaktion, die die Änderung durchgeführt hat

- *PageID*
 - die Kennung der Seite, auf der die Änderungsoperation vollzogen wurde
 - Wenn eine Änderung mehr als eine Seite betrifft, müssen entsprechend viele Log-Einträge generiert werden
- *PrevLSN*,
 - Zeiger auf den vorhergehenden Log-Eintrag der jeweiligen Transaktion
 - Diesen Eintrag benötigt man aus Effizienzgründen

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

$undo(undo(\dots(undo(a))\dots))$ $undo(a)$
 $redo(redo(\dots(redo(a))\dots))$ $redo(a)$



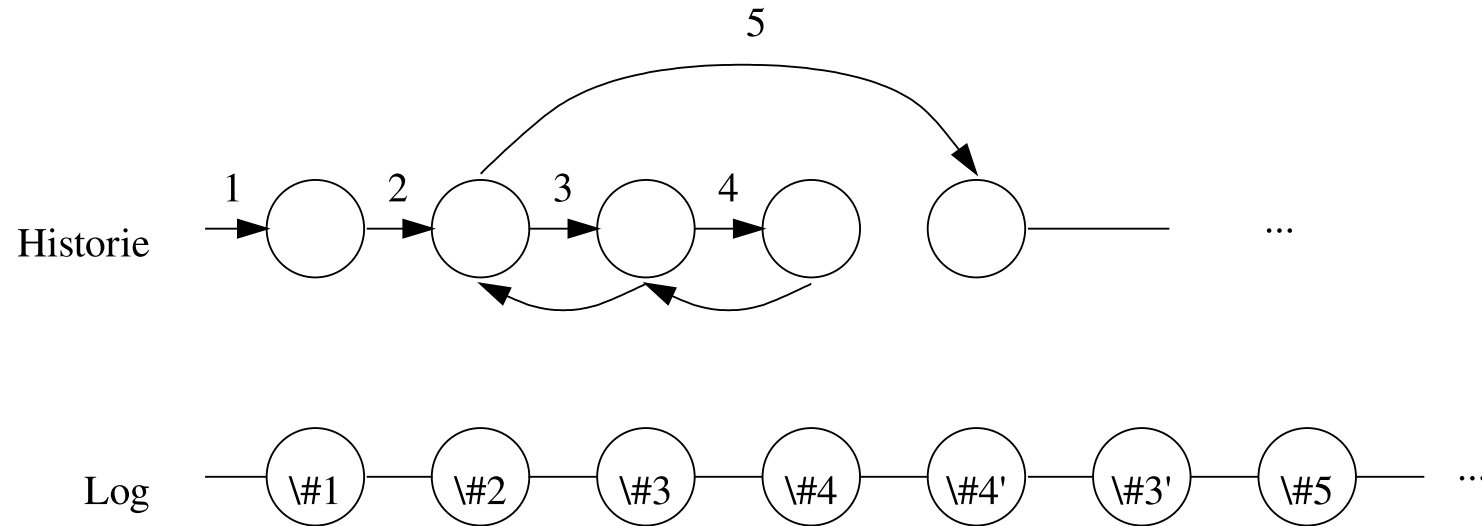


- Kompensationseinträge (CLR: compensating log record) für rückgängig gemachte Änderungen.
- #7 ist CLR für #7
- #4 ist CLR für #4

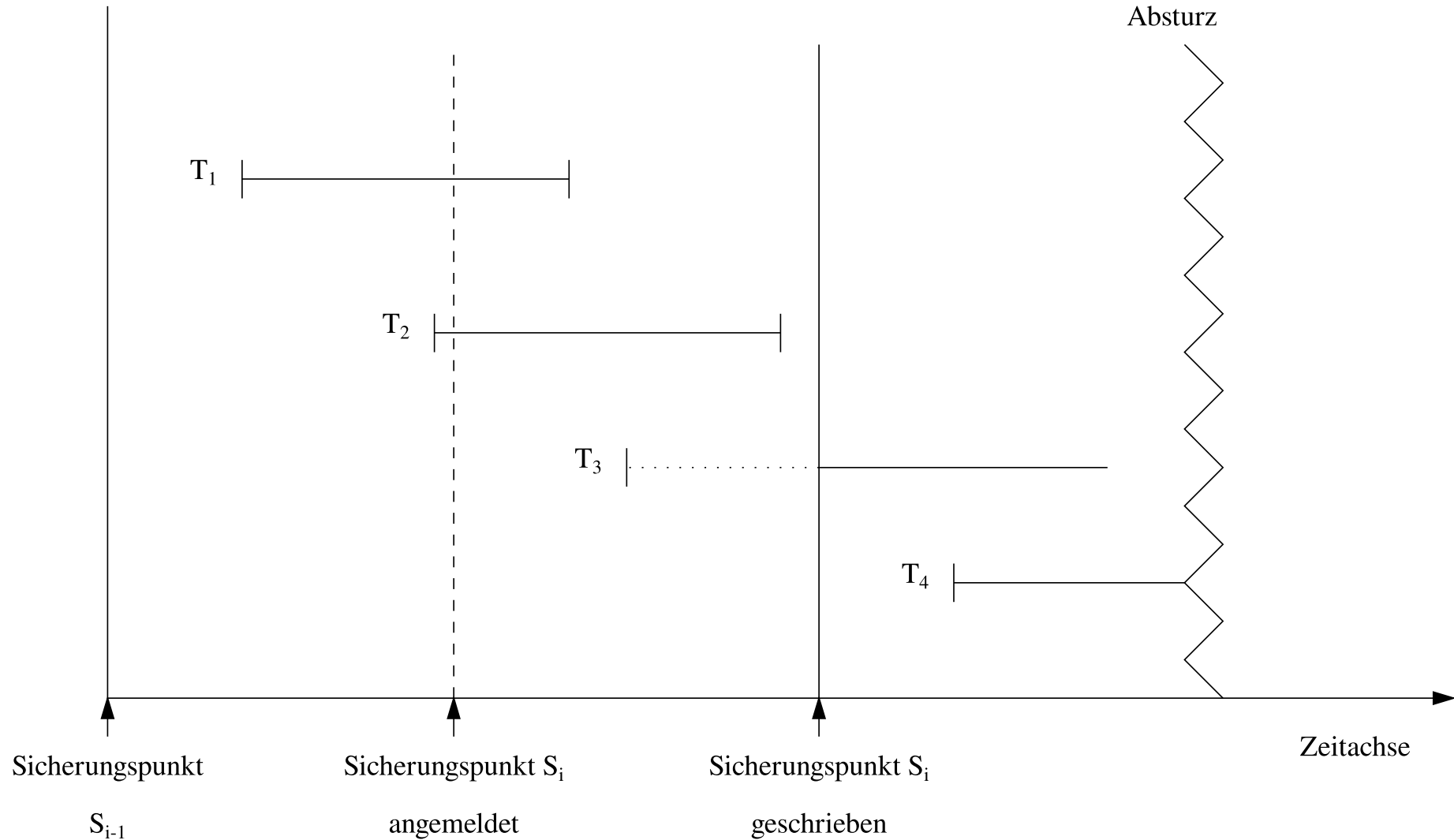
[#1, T_1 , **BOT**, 0]
[#2, T_2 , **BOT**, 0]
[#3, T_1 , P_A , $A--=50$, $A+=50$, #1]
[#4, T_2 , P_C , $C+=100$, $C--=100$, #2]
[#5, T_1 , P_B , $B+=50$, $B--=50$, #3]
[#6, T_1 , **commit**, #5]
[#7, T_2 , P_A , $A--=100$, $A+=100$, #4]
angle.l#7', T_2 , P_A , $A+=100$, #7, #4*angle.r*
angle.l#4', T_2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

- CLRs sind durch spitze Klammern *angle.l...angle.r* gekennzeichnet

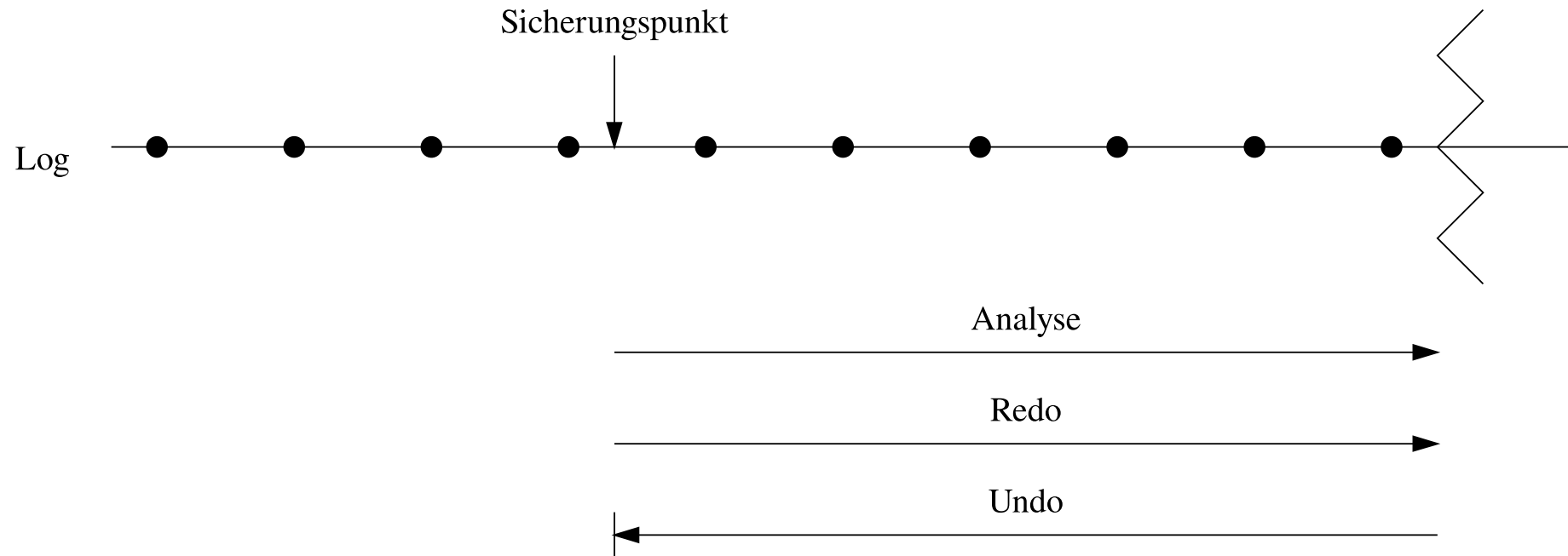
- der Aufbau eines CLR ist wie folgt
 - LSN
 - TA-Identifikator
 - betroffene Seite
 - Redo-Information
 - PrevLSN
 - UndoNxtLSN (Verweis auf die nächste rückgängig zu machende Änderung)



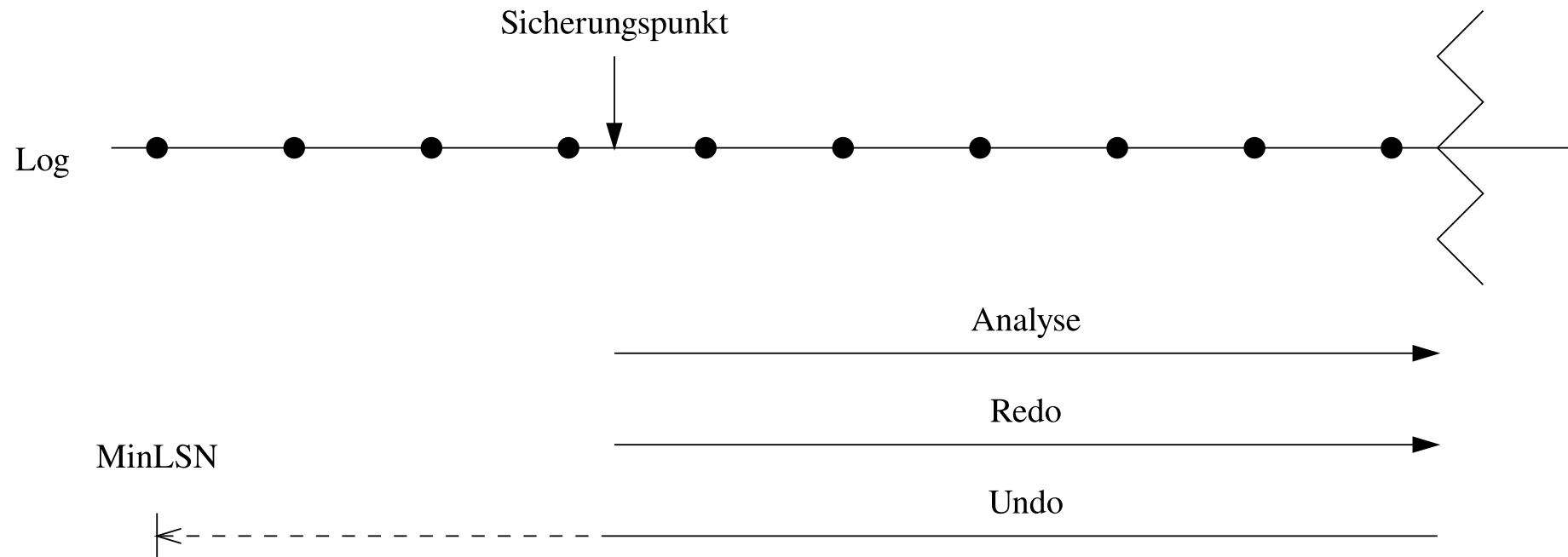
- Schritte 3 und 4 werden zurückgenommen
- notwendig für die Realisierung von Sicherungspunkten innerhalb einer TA

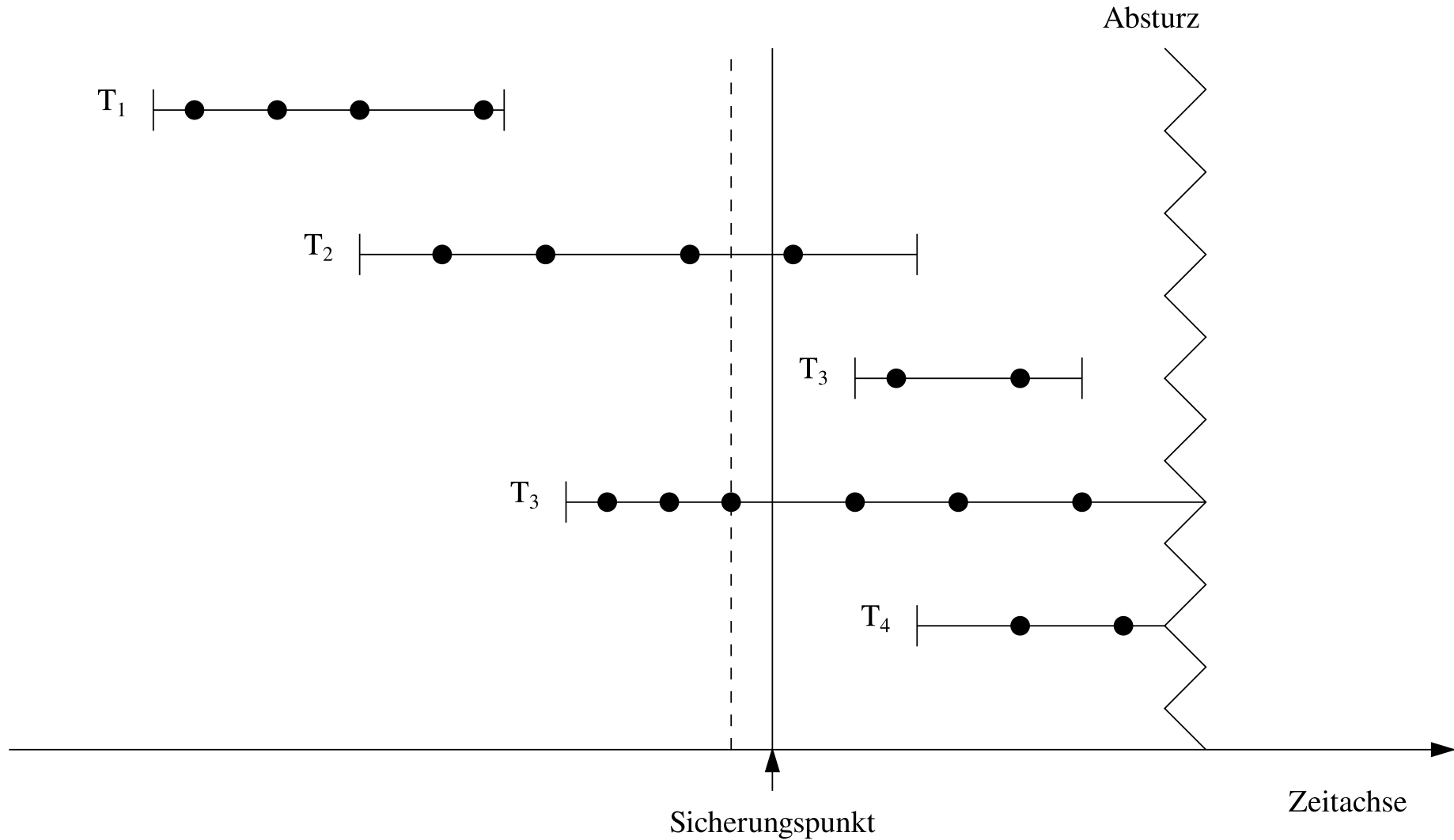


- Transaktionskonsistent:

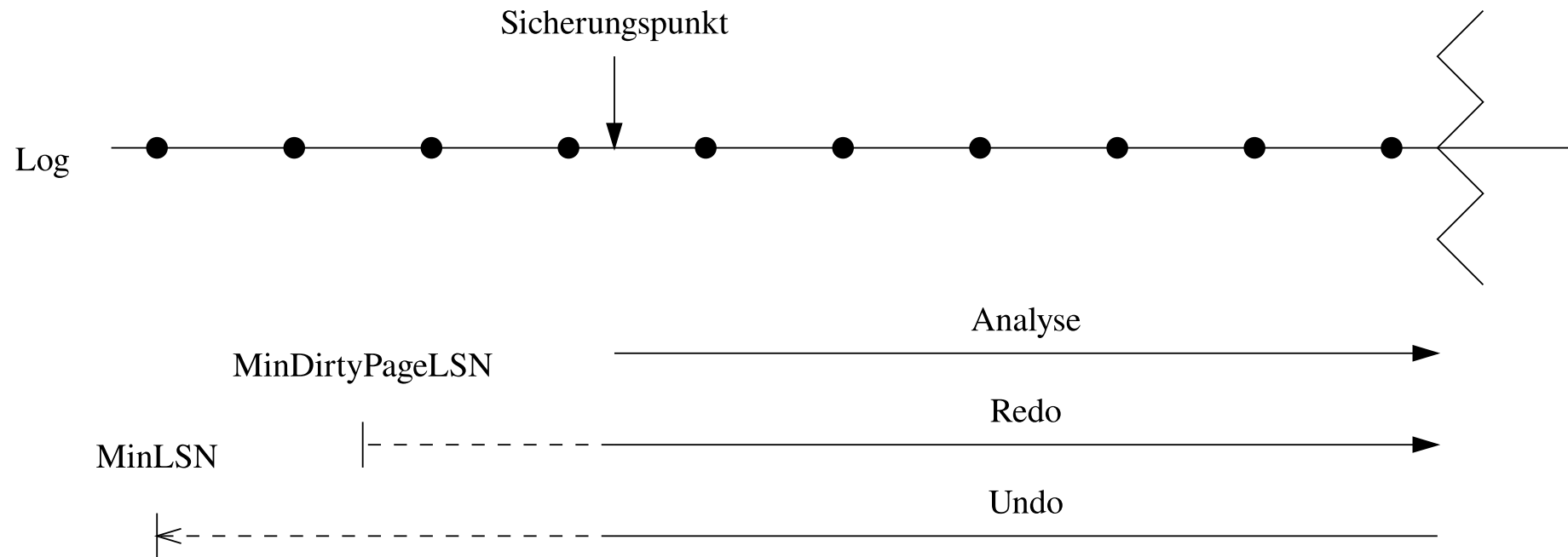


- Aktionskonsistent:





- Unscharf (fuzzy):



- modifizierte Seiten werden nicht ausgeschrieben
- nur deren Kennung wird ausgeschrieben
 - *Dirty Pages*=Menge der modifizierten Seiten
- *MinDirtyPageLSN*: die minimale LSN, deren Änderungen noch nicht ausgeschrieben wurde

- Fehlerbehandlung (Recovery) kann Zustand der Datenbasis zum Absturzzeitpunkt wiederherstellen
- Die zwei Hauptmechanismen dazu sind:
 - Log-Dateien
 - Sicherungspunkte