

Datenbanksysteme 2011

Kapitel 14: Recovery

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Fehlerklassen

- lokaler Fehler in einer noch nicht festgeschriebenen Transaktion
- Fehler mit Hauptspeicherverlust
- Fehler mit Hintergrundspeicherverlust

Lokaler Fehler in einer Transaktion

Grund:	Fehler im Anwendungsprogramm Abort systemgesteuerter Abbruch einer Transaktion
Behebung:	Änderungen der abgebrochenen Transaktion rückgängig machen (lokales undo)
Frequenz:	minütlich
Aufwand:	Millisekunden

Fehler mit Hauptspeicherverlust

Grund: Stromausfall, Hardwareausfall,
Betriebssystemproblem

Behebung: alle durch nicht abgeschlossene
Transaktionen schon in die materialisierte
Datenbasis eingebrachten Änderungen
müssen rückgängig gemacht werden
(*globales undo*)

alle noch nicht in die materialisierte
Datenbasis eingebrachten Änderungen
durch abgeschlossene Transaktionen
müssen nachvollzogen werden
(*globales redo*).

Frequenz: täglich

Aufwand: Minuten

Fehler mit Hintergrundspeicherverlust

Grund:	head crash Feuer/Erdbeben Betriebssystemproblem
Behebung:	Archivkopie + Logarchiv
Frequenz:	jährlich
Aufwand:	Stunden

Lokales Zurücksetzen einer Transaktion

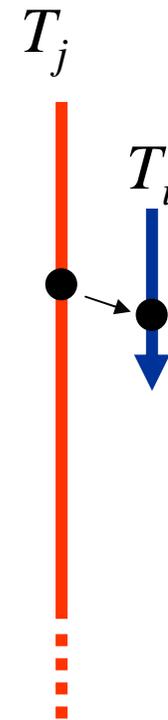
Transaktion T_j schreibt etwas,
was von T_i gelesen wird.

Sekunden später:

T_i : commit

T_j : abort

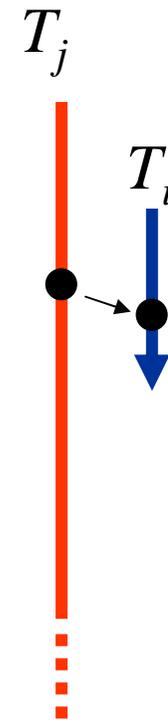
Problem: T_i hat Wert verarbeitet,
der "offiziell" nie existiert hat.



Rücksetzbare Historien

Eine Historie heißt *rücksetzbar*, falls immer die schreibende Transaktion T_j vor der lesenden Transaktion T_i ihr **commit** ausführt.

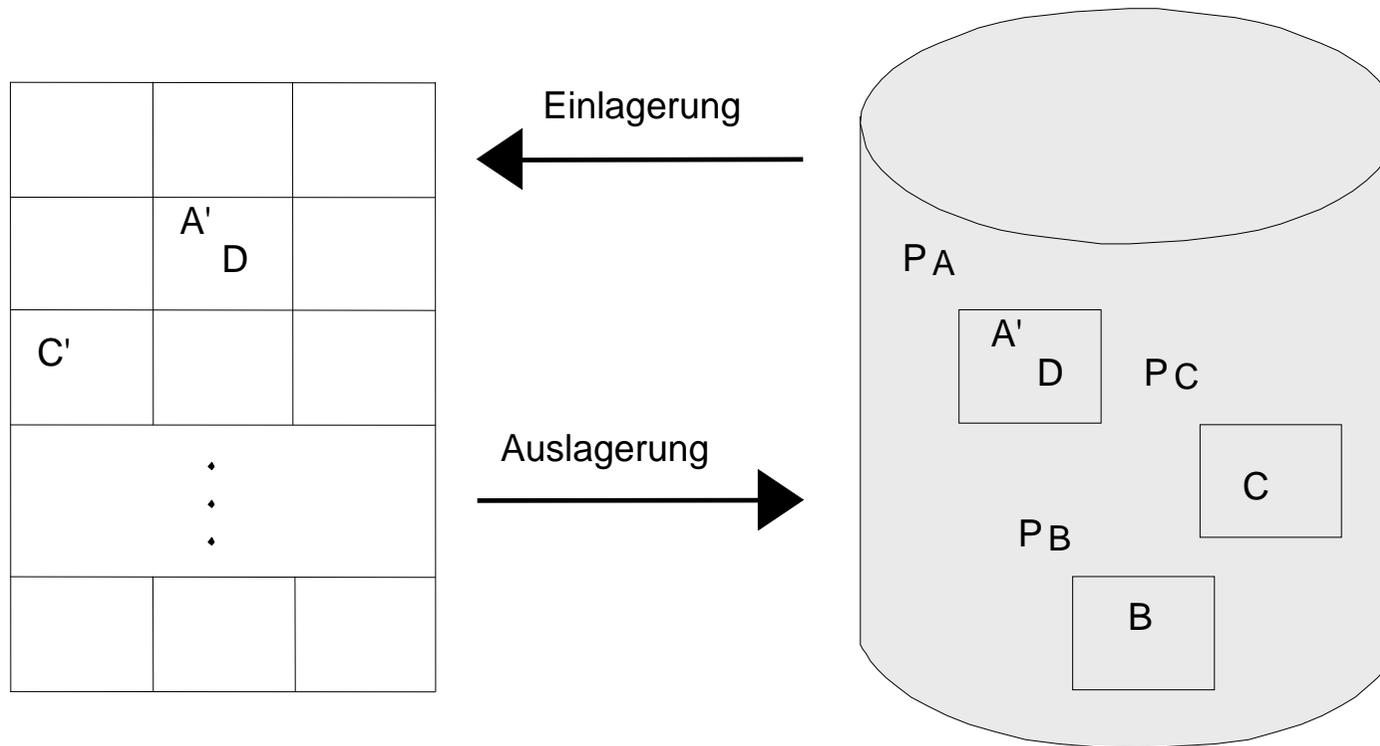
Eine Transaktion darf erst dann ihr **commit** durchführen, wenn alle Transaktionen, von denen sie gelesen hat, beendet sind.



Speicherhierarchie

DBMS-Puffer

Hintergrundspeicher



Für die Dauer eines Zugriffs wird die Seite im Puffer *fixiert*.

Fixierte Seiten werden nicht verdrängt.

Werden Daten geändert, so wird die Seite als *dirty* markiert.

Ersetzen von Pufferseiten

Bei Platzbedarf (wegen Einlagern):

¬steal:

Die Ersetzung von Seiten, die von einer noch aktiven Transaktion modifiziert wurden, ist ausgeschlossen.

steal:

Jede nicht fixierte Seite darf ausgelagert werden.

Zurückschreiben von Pufferseiten

Bei Transaktionsende:

force:

Beim **commit** einer Transaktion werden alle von ihr modifizierten Seiten in die materialisierte Datenbasis zurückkopiert.

¬force:

Modifizierte Seiten werden nicht unmittelbar nach einem **commit**, sondern ggf. auch später, in die materialisierte Datenbasis zurückkopiert.

(z.B. weil Seite heftig genutzt wird).

Kombinationsmöglichkeiten

	force	\neg force
\neg steal	Kein Redo kein Undo	Redo kein Undo
steal	Kein Redo Undo	Redo Undo

Einbringstrategie

- update-in-place
Jeder eingelagerten Seite P entspricht eine Seite P im Hintergrundspeicher
- Twin-Block-Verfahren
Jeder eingelagerten Seite P werden zwei Seiten P^0 und P^1 im Hintergrundspeicher zugeordnet mit dem letzten und vorletzten Zustand.
Zurückschreiben erfolgt jeweils auf den vorletzten Zustand.

Systemkonfiguration

- *steal*
- *-force*
- *update-in-place*
- *Kleine Sperrgranulate*
Seiten können Änderungen von abgeschlossenen und nicht abgeschlossenen Transaktionen enthalten

Struktur der Log-Einträge

- *LSN (Log Sequence Number)*
- *Transaktionskennung TA*
- *PageID*
- *Redo-Information*
- *Undo-Information*
- *PrevLSN*

Beispiel einer Log-Datei

T_1	T_2	[LSN, TA, PageID, Redo, Undo, PrevLSN]
BOT		[#1, T_1, BOT, 0]
r(A, a₁)		
	BOT	[#2, T_2, BOT, 0]
	r(C, c₂)	
a₁ := a₁ - 50		
w(A, a₁)		[#3, T_1, P_A, A-=50, A+=50, #1]
	c₂ := c₂ + 100	
	w(C, c₂)	[#4, T_2, P_C, C+=100, C-=100, #2]
r(B, b₁)		
b₁ := b₁ + 50		
w(B, b₁)		[#5, T_1, P_B, B+=50, B-=50, #3]
commit		[#6, T_1, commit, #5]
	r(A, a₂)	
	a₂ := a₂ - 100	
	w(A, a₂)	[#7, T_2, P_A, A-=100, A+=100, #4]
	commit	[#8, T_2, commit, #7]

Logische versus physische Protokollierung

Logische Protokollierung:

Notiere *Undo*-Operation, um vorherigen Zustand zu erzeugen

Notiere *Redo*-Operation, um Nachfolgezustand zu erzeugen

Physische Protokollierung:

Notiere *Before-Image* des Datenobjekts statt Undo-Operation

Notiere *After-Image* des Datenobjekts statt Redo-Operation

Before-Image = *Undo*-Code angewendet auf *After-Image*

After-Image = *Redo*-Code angewendet auf *Before-Image*

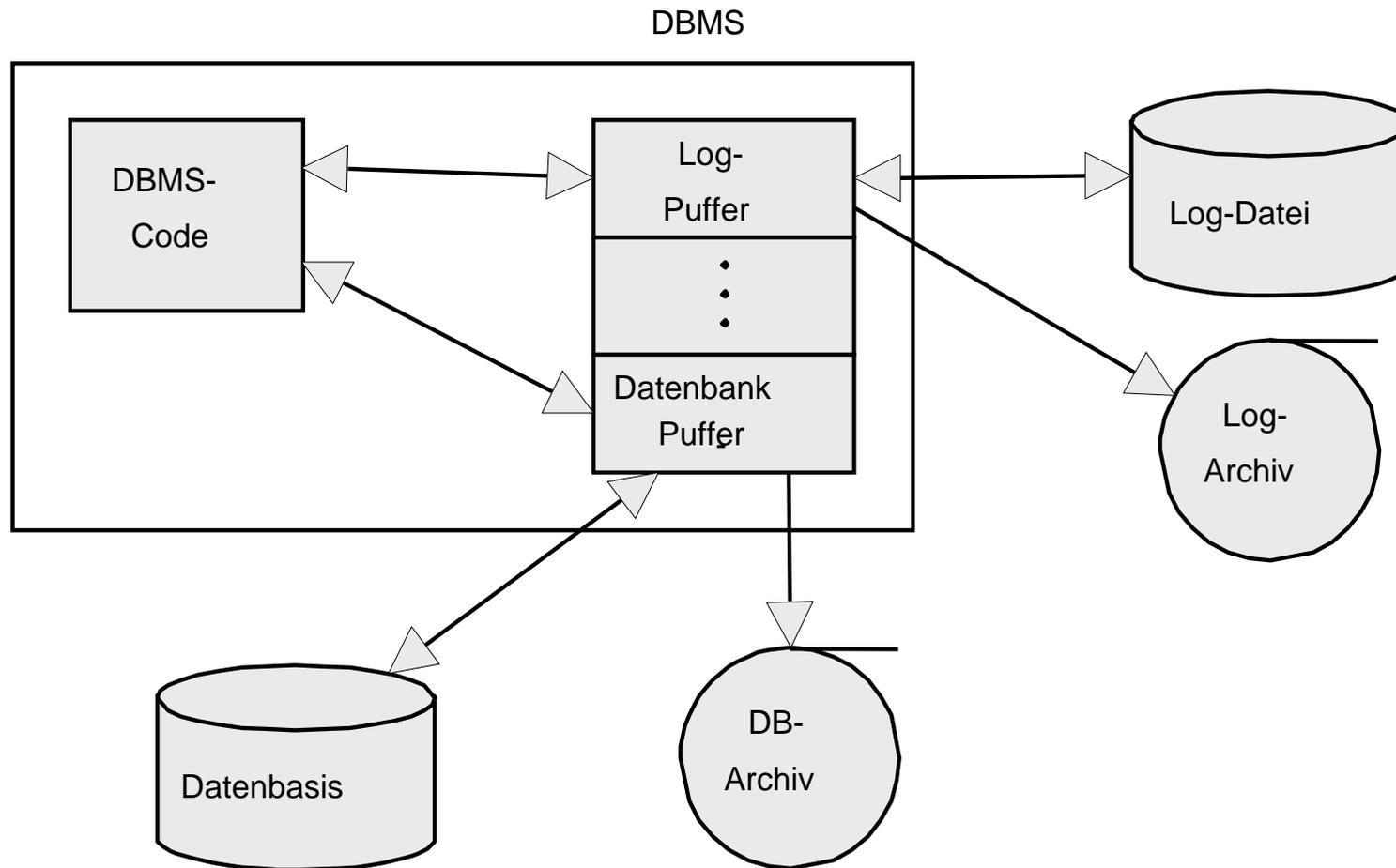
Before or After, that is the question

- Problem: In der Recovery-Phase ist zunächst unklar, bis zu welcher Einzelaktion eine Transaktion bereits ihre Effekte in die Datenbank geschrieben hat.
- Beim Anlegen eines Log-Eintrages wird die neu generierte LSN in einen reservierten Bereich der Seite geschrieben.
- Beim Auslagern wird LSN mitkopiert.
- Wenn die LSN der Seite im Hintergrundspeicher einen kleineren Wert als die LSN des Log-Eintrags enthält, handelt es sich um das *Before-Image*.
- Ist die LSN der Seite im Hintergrundspeicher größer oder gleich der LSN des Log-Eintrags, dann wurde bereits das *After-Image* auf den Hintergrundspeicher propagiert.

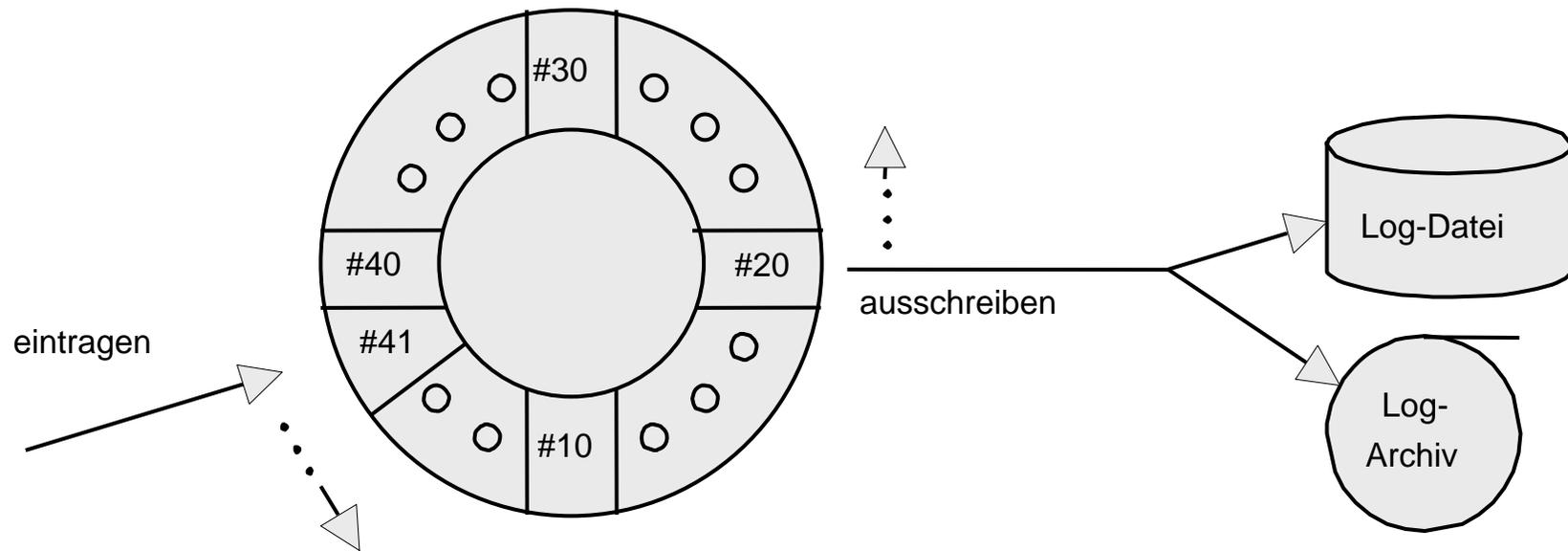
WAL-Prinzip (Write Ahead)

- Bevor eine Transaktion festgeschrieben (**committed**) wird, müssen alle zu ihr gehörenden Log-Einträge geschrieben werden.
(wegen Redo)
- Bevor eine modifizierte Seite ausgelagert werden darf, müssen alle Log-Einträge, die zu dieser Seite gehören, in die Log-Datei geschrieben werden.
(wegen Undo)

Speicherhierarchie zur Datensicherung



Log-Ringpuffer



Wiederaanlauf nach einem Fehler



Transaktion T_1 ist ein *Winner* und verlangt ein *Redo*.

Transaktion T_2 ist ein *Loser* und verlangt ein *Undo*.

1. Analyse: *Winner* und *Loser* ermitteln (commit !)
2. Log-Datei vorwärts: *Redo, falls erforderlich (LSN !)*
3. Log-Datei rückwärts: *Undo*

Redo für Winner & Loser

- Logdatei vorwärts durchlaufen
- Referierte Seite aus Hintergrundspeicher holen
- Update durchführen, falls erforderlich (LSN)
- LSN updaten
- Seite zurückschreiben.

Undo für Loser

- Logdatei rückwärts durchlaufen
- Referierte Seite aus Hintergrundspeicher holen
- Update durchführen
- [Compensation Log Records schreiben]
- Seite zurückschreiben.

Winner & Loser

LSN	TA	PageID	Redo	Undo	PrevLSN
#1	T1	BOT	0		
#2	T2	BOT	0		
#3	T1	PA	A-=50	A+=50	#1
#4	T2	PC	C+=100	C-=100	#2
#5	T1	PB	B+=50	B-=50	#3
#6	T1	commit			#5
#7	T2	PA	A-=100	A+=100	#4

Fehlertoleranz des Wiederanlaufs

Absturz bei Recovery:

Forderung: Redo-Phase und Undo-Phase
müssen *idempotent* sein:

$$\text{undo}(\text{undo}(\dots\text{undo}(a)\dots)) = \text{undo}(a)$$

$$\text{redo}(\text{redo}(\dots\text{redo}(a)\dots)) = \text{redo}(a)$$

Idempotenz der Redo-Phase durch LSN:
Doppeltes redo wird verhindert.

Idempotenz der Undo-Phase:
Für jede Undo-Operation wird CLR (Compensation Log Record)
angelegt.

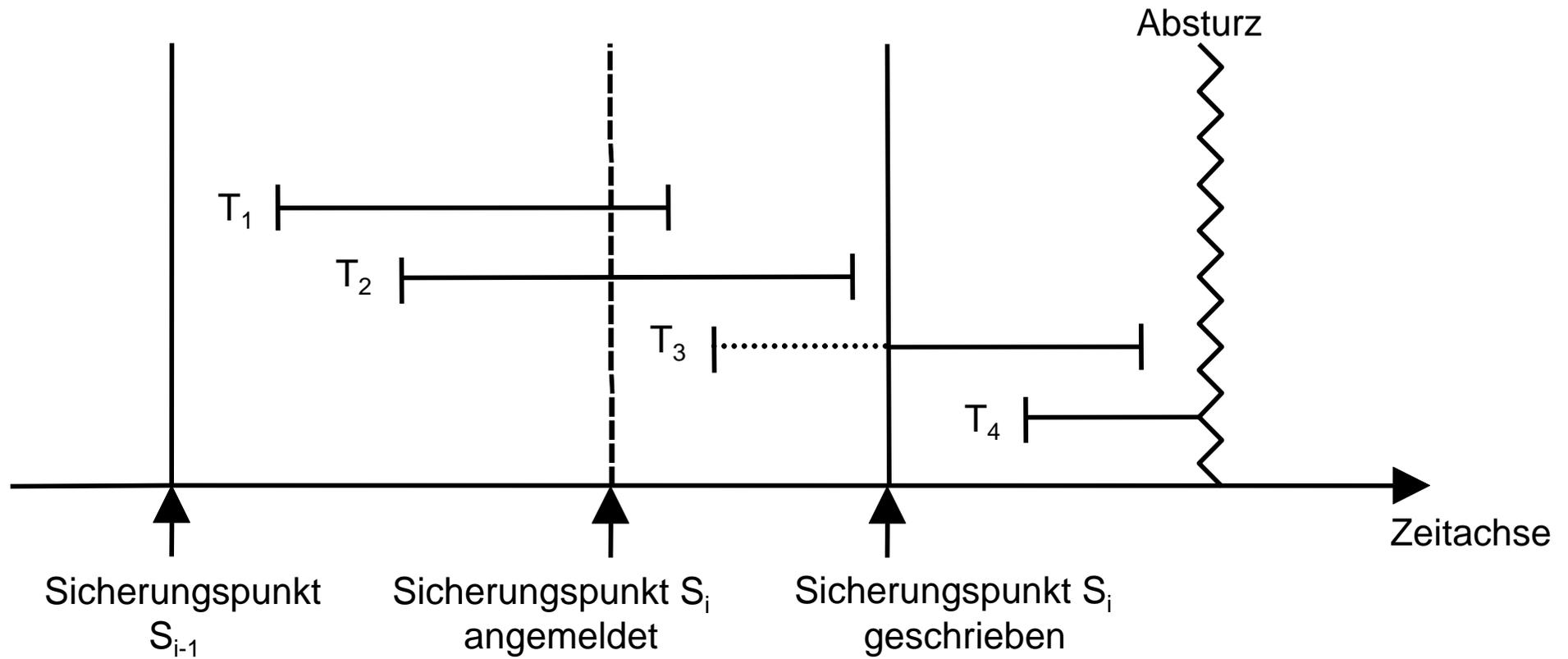
Struktur der CLR-Einträge

- *LSN (Log Sequence Number)*
- *Transaktionskennung TA*
- *PageID*
- *Redo-Information (= erfolgreiche Undo während Recovery)*
- *PrevLSN*
- *UndoNxtLSN*

Beispiel für CLR

[LSN,	TA,	PageID,	Redo,	Undo,	PrevLSN]		
				PrevLSN	UndoNxtLSN]		
●	[#1,	T1,	BOT,	0]		Redo-Phase	
●	[#2,	T2,	BOT,	0]			
●	[#3,	T1,	PA,	A-=50,	A+=50,	#1]	Undo-Phase
● ●	[#4,	T2,	PC,	C+=100,	C-=100,	#2]	
●	[#5,	T1,	PB,	B+=50,	B-=50,	#3]	Absturz
●	[#6,	T1,	commit,			#5]	
● ●	[#7,	T2,	PA,	A-=100,	A+=100,	#4]	erneuter Absturz
	<#7',	T2,	PA,	A+=100,	#7,	#4>	
	<#4',	T2,	PC,	C-=100,	#7',	#2>	
						erneutes Redo	erneutes Undo

Sicherungspunkte



Recovery-Arten

