Kapitel 14:
Recovery

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

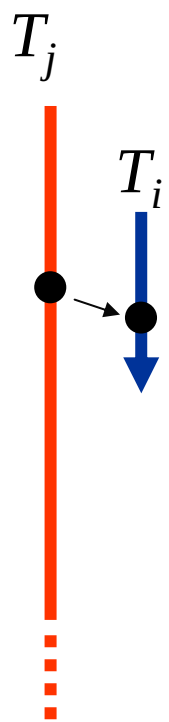
Transaktion T_j schreibt etwas,
was von T_i gelesen wird.

Sekunden später:

T_i : commit

T_j: abort

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

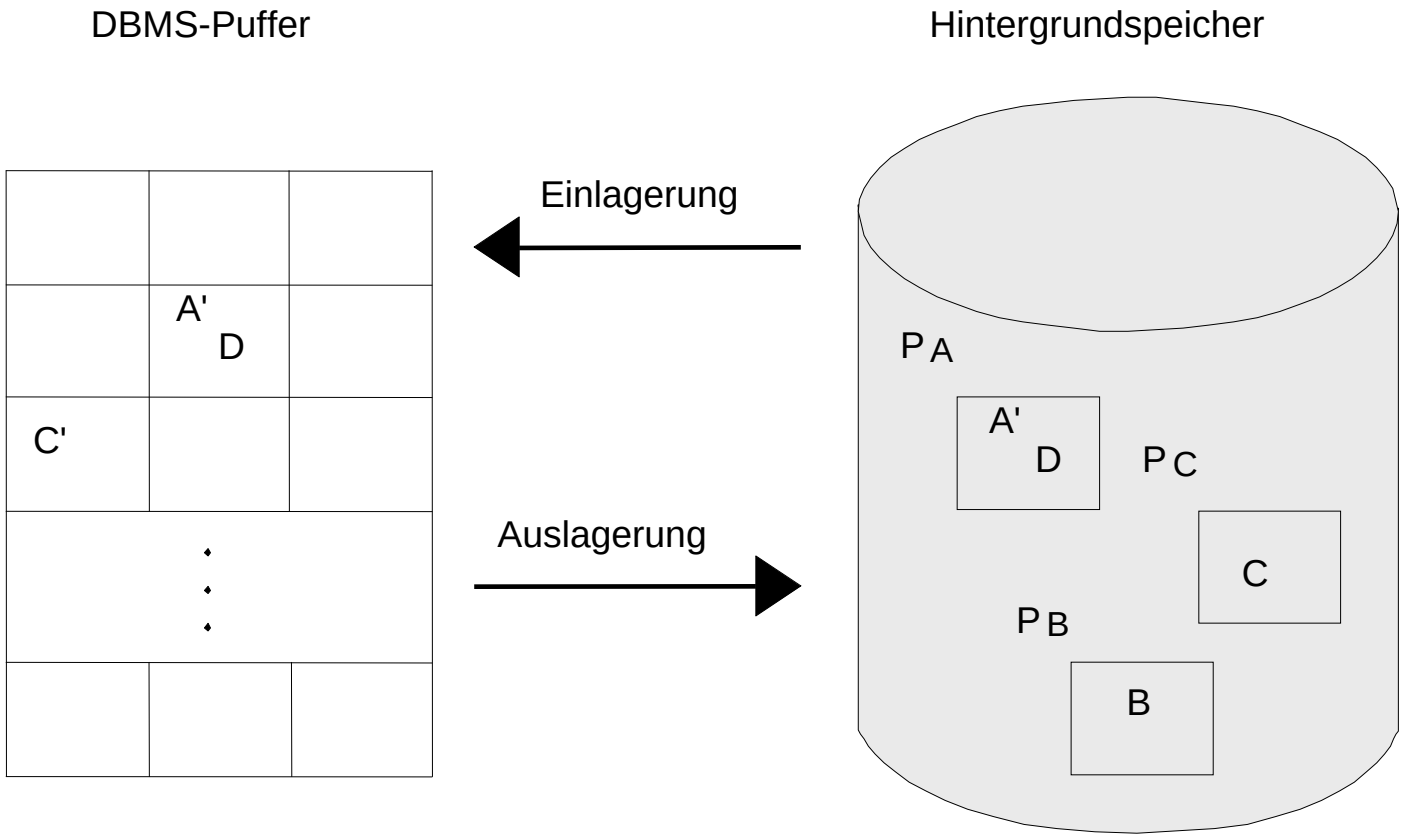


Rücksetzbare Historien

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

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

Speicherhierarchie



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:

¬*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	¬force
¬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, a_1)		
	BOT	**[#2, T_2, BOT, 0]**
	r(C, c_2)	
a_1 := a_1 - 50		
w(A, a_1)		**[#3, T_1, P_A, A-=50, A+=50, #1]**
	c_2 := c_2 + 100	
	w(C, c_2)	**[#4, T_2, P_C, C+=100, C-=100, #2]**
r(B, b_1)		
b_1 := b_1 + 50		
w(B, b_1)		**[#5, T_1, P_B, B+=50, B-=50, #3]**
commit		**[#6, T_1, commit, #5]**
	r(A, a_2)	
	a_2 := a_2 - 100	
	w(A, a_2)	**[#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*

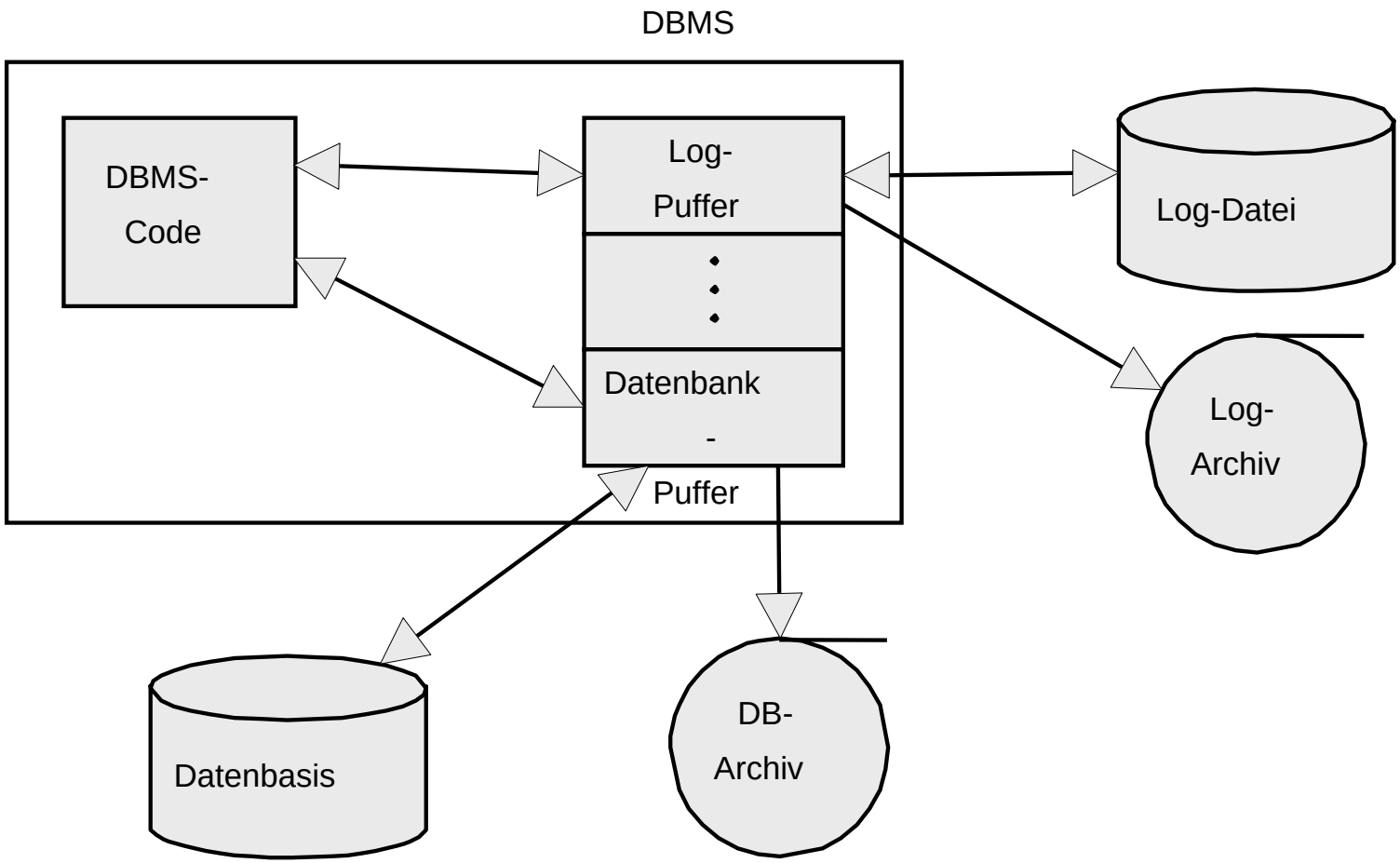
Before-Image versus After-Image

- Beim Anlegen eines Log-Eintrages wird die neu generierte LSN in einen reservierten Bereich der Seite geschrieben.

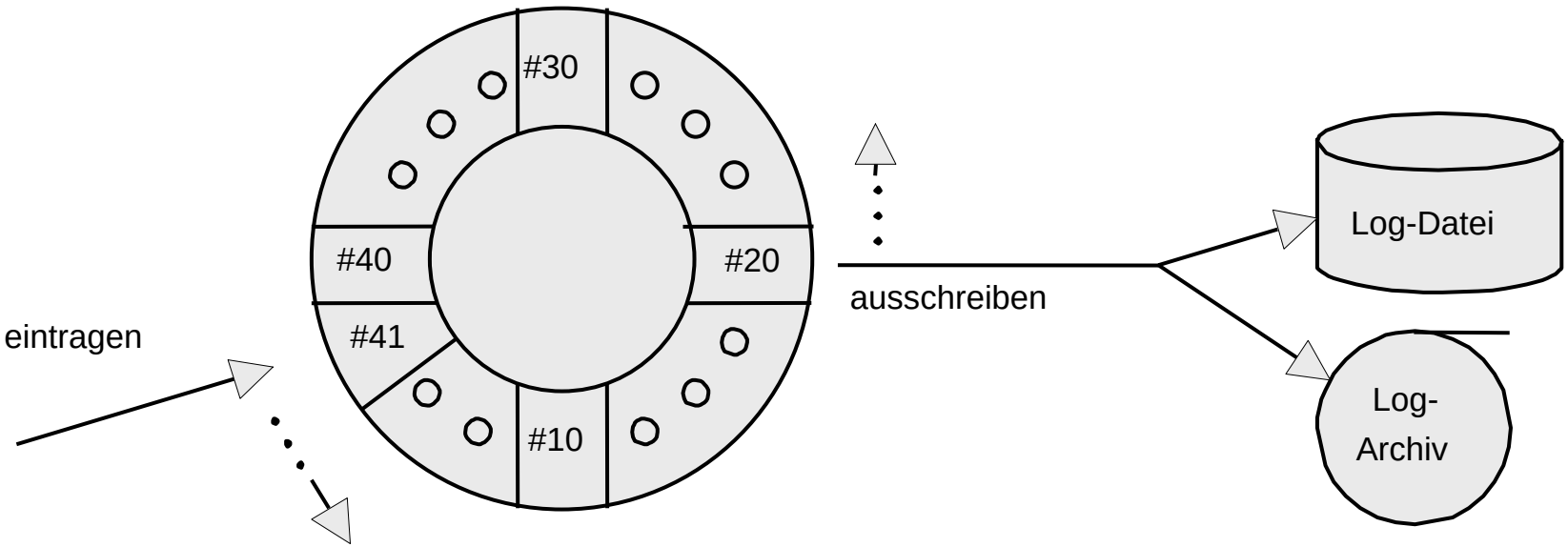
- Wenn die LSN der Seite einen kleineren Wert als die LSN des Log-Eintrags enthält, handelt es sich um das *Before-Image*.

- Ist die LSN der Seite größer oder gleich der LSN des Log-Eintrags, dann wurde bereits das *After-Image* auf den Hintergrundspeicher propagiert.

Speicherhierarchie zur Datensicherung



Log-Ringpuffer

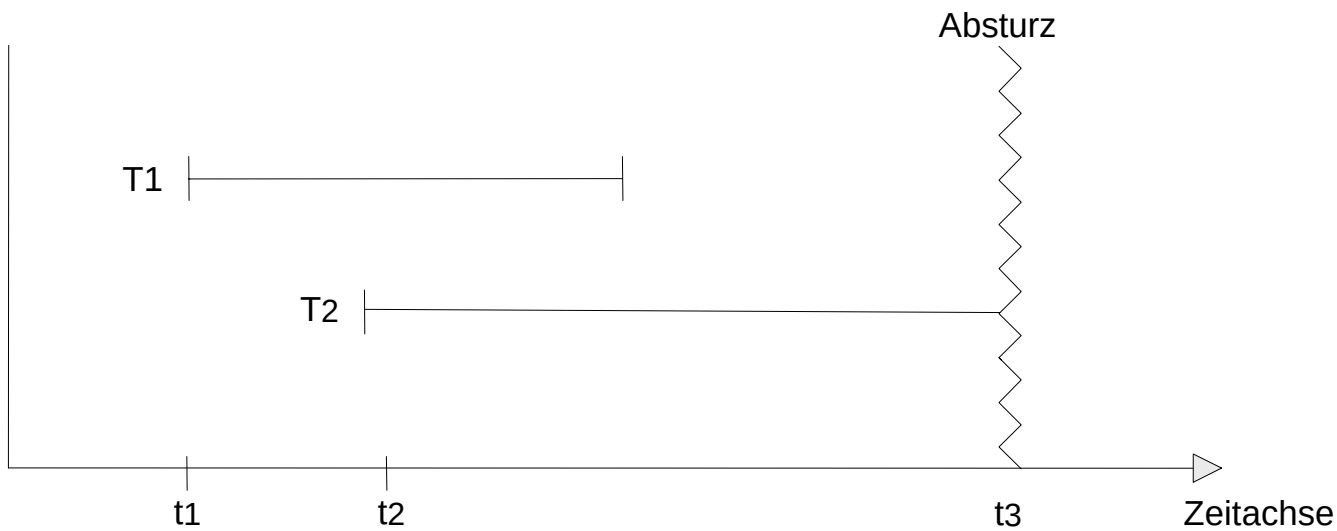


WAL-Prinzip (Write Ahead)

- 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)

Wiederanlauf 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*

Fehlertoleranz des Wiederanlaufs

Absturz bei Recovery:

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

$$undo(undo(\ldots undo(a))\ldots)) = undo(a)$$
$$redo(redo(\ldots redo(a))\ldots)) = redo(a)$$

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

Idempotenz der Undo-Phase:
Für jede Undo-Operation wird CLR angelegt.

Struktur der CLR-Einträge

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

Beispiel für CLR

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

PrevLSN UndoNxtLSN]

```
[#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]
<#7', T2, PA, A+=100, #7,     #4>
<#4', T2, PC, C-=100, #7',    #2>
```

Redo-Phase

Undo-Phase

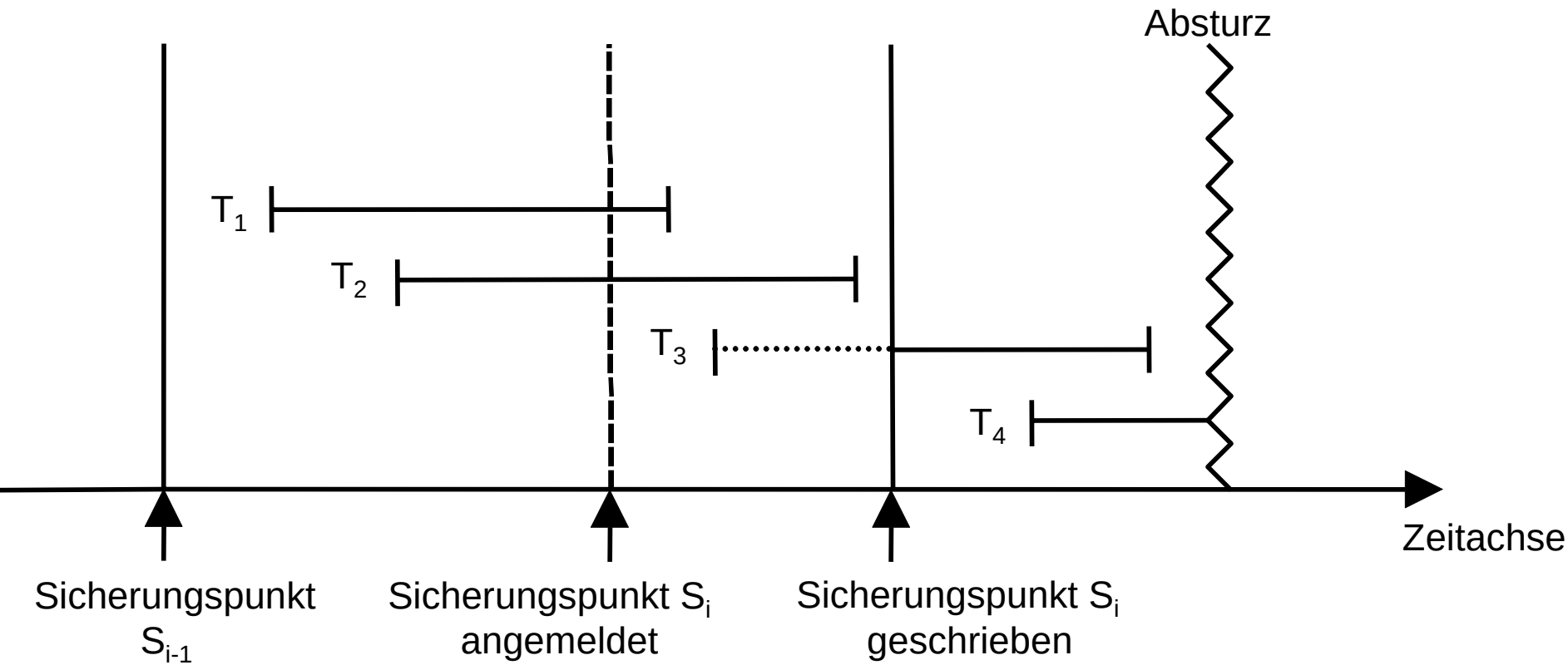
Absturz

erneuter Absturz

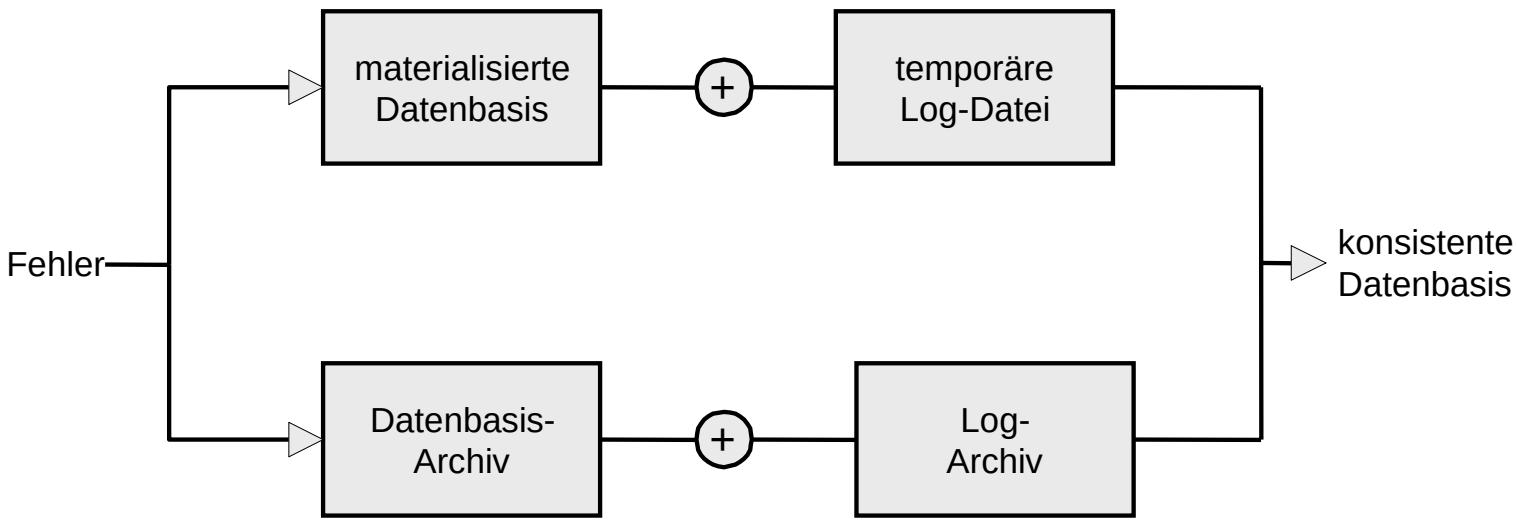
erneutes Redo

erneutes Undo

Sicherungspunkte



Recovery-Arten



Sicherung der Datenbank

```
EXEC sp_addumpdevice
            'disk',
            'unidump',
            'c:\dump\unidump.dat',

backup database uni to unidump

restore database uni from unidump
```

Sicherung der Log-Datei

```
EXEC sp_addumpdevice
      'disk',
      'unilog',
      'c:\dump\unilog.dat'

backup log uni to unilog from unilog

restore log uni from unilog
```