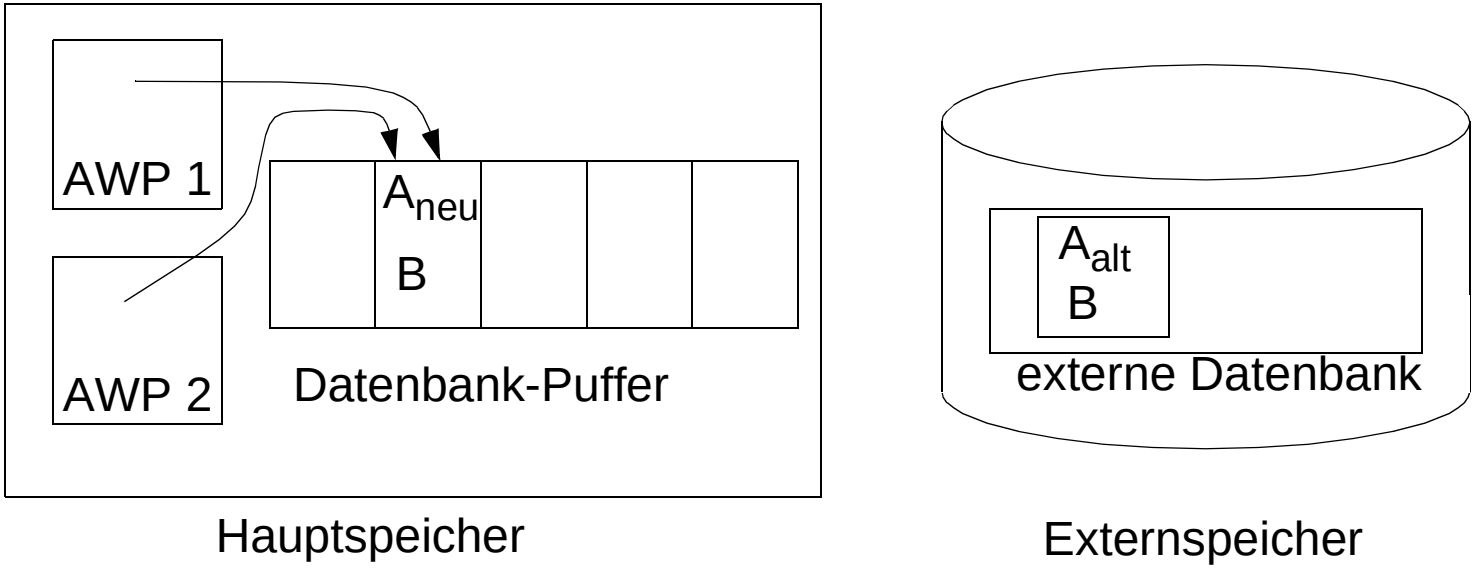
9.3 Fehlerbehandlung

- Schutz vor Beeinträchtigungen durch Fehler des Systems oder eines Benutzers
- nach Systemzusammensturz innerhalb einer TA
  - inkonsistenter Zustand der DB
  - physische und logische Inkonsistenz
- Recovery-Komponente eines DBS:
  - dient zum Wiederherstellen eines korrekten DB-Zustandes
  - basiert auf dem TA-Konzept des DBS

Fehlerklassen:

- Transaktionsfehler (z. B. Deadlock, Konsistenzverletzung, Division durch 0)
  - Rücksetzen einer oder mehrerer TAs im laufenden Betrieb
- Systemfehler (DBS ist funktionsunfähig, Verlust des Inhalts im Hauptspeicher)
  - Rückgängigmachen aller laufenden TAs (Neustart des Systems)
- Speicherfehler (Verlust des Plattenspeichers durch “head crash”, selten)
  - Rekonstruieren der Datenbank

Lesen und Schreiben von Datensätzen



- Datensätzen werden in sogenannten Seiten (Blöcken) gespeichert
- eine Seite ist die kleinste Transfereinheit zwischen Extern- und Hauptspeicher
- ein DBS besitzt einen Puffer, in dem Datensätze (und Datenseiten) für die AWPs bereit gestellt werden.
- zu jeder Seite im Puffer gibt es genau eine Seite in der externen DB
  - Seite in der externen DB ist aber ggf. veraltet.
  - Schreiben einer Seite in die DB zerstört den alten Zustand.

interner Ablauf einer Lese/Schreiboperation

1. Lese die Seite vom Externspeicher in den Puffer (wenn nicht bereits vorhanden)
2. Fixiere die Seite im Puffer, d.h. die Seite bleibt fest im Hauptspeicher.
3. Setze eine Sperre auf den gewünschten Datensatz.

   AWP liest/schreibt den Datensatz und führt weitere Operationen aus.

4. Hebe die Sperre auf.
5. Kennzeichne, daß das AWP die Seite nicht mehr benötigt (Unfix).

im Fall von Schreiboperationen:

- Datenbank gerät kurzzeitig in einen inkonsistenten Zustand (innerhalb einer TA)
- modifizierte Seiten im Puffer werden nicht sofort auf den Externspeicher übertragen
  - externe DB ist veraltet
  - externe DB hat nach dem Ende der TA einen inkonsistenten Zustand.
- Verlust des Hauptspeichers ==> inkonsistente DB

Ziel:

- DB soll auch bei Verlust des HSP in einem konsistenten Zustand sein.

Varianten beim Lesen/Schreiben

Freigabe von Seiten

- Pufferverwalter kann diese Seite aus dem Puffer entfernen und (wenn die Seite geändert wurde) auf den Externspeicher schreiben
- Varianten
  - steal: vor dem commit einer TA
  - no-steal: keine Freigabe vor dem commit der TA.

Schreiben der modifizierten Seiten auf den Externspeicher

- Varianten
  - force: Schreiben der Seiten beim commit
  - no-force: Schreiben der Seiten zu einem späteren Zeitpunkt

in heutigen DBS:

- no-force und steal

Protokollierung von Änderungsoperationen

- mögliche Fälle nach einem commit einer TA (im Fall von steal und no-force):
  - externe DB ist in einem inkonsistenten Zustand
  - Änderungen sind in der externen DB noch nicht wirksam geworden.
- Protokoll
  - REDO-Information:
    wenn Änderungen nachvollzogen werden sollen.
  - UNDO-Information:
    wenn Änderungen rückgängig gemacht werden sollen.
- Eintrag in der Protokolldatei (Log) besteht aus:
  - LSN: Log Sequence Number
    eindeutige Kennung (monoton wachsend)
  - TA_ID: Transaktionskennung
  - SID: Seitennummer
  - REDO-Information
  - UNDO-Information
  - P_LSN: Zeiger auf den vorherigen Log-Eintrag der Transaktion TA_ID.

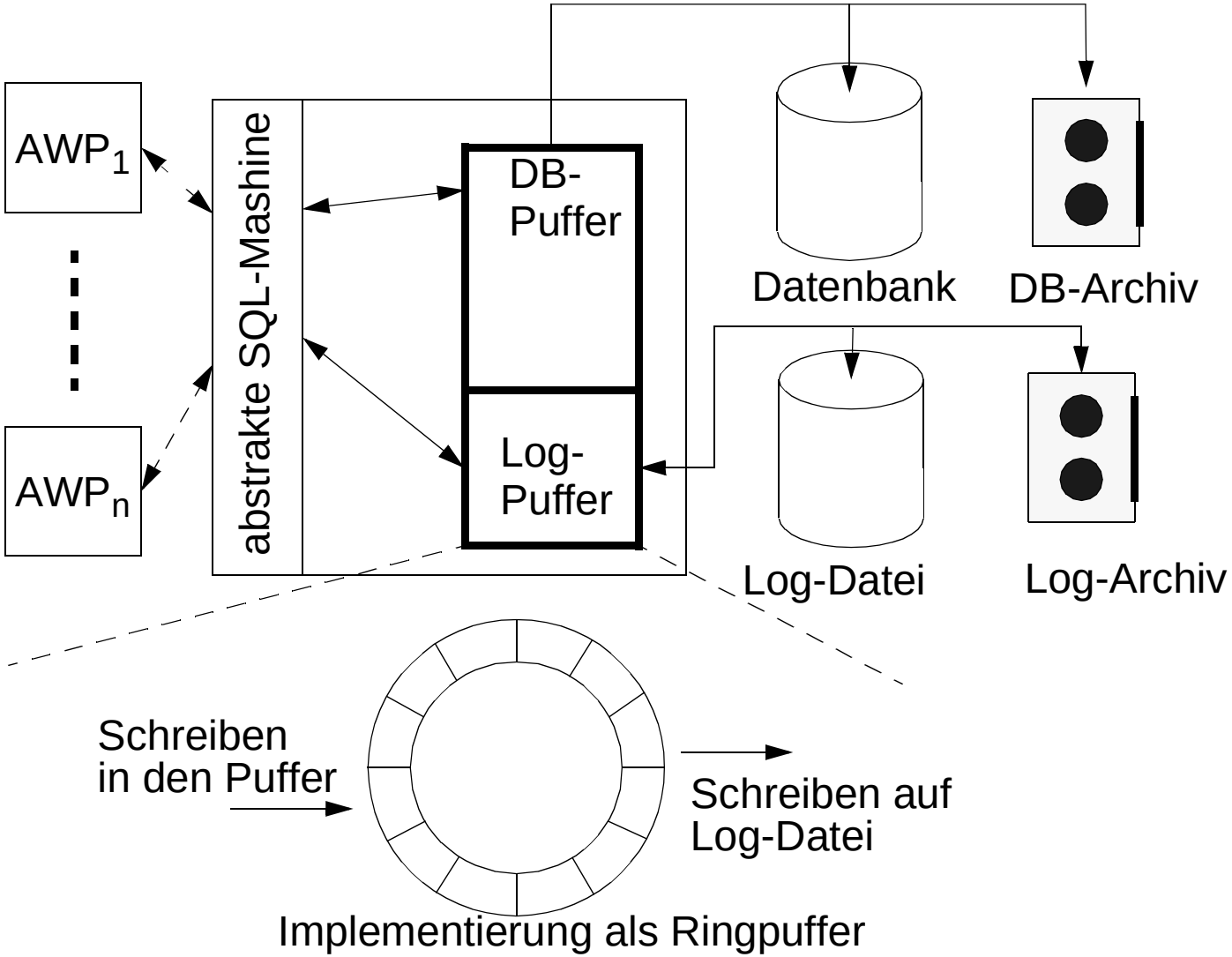
Beispiel:

Transaktion T1	Transaktion T2	logische Protokolldatei	physische Protokolldatei
begin-TA		<#1, T1, begin_TA, 0>	<#1, T1, begin_TA, 0>
r(A,$a1_{old}$)			
	begin_TA	<#2, T2, begin_TA, 0>	<#2, T2, begin_TA, 0>
	r(C,c_{old})		
$a1_{new} = a1_{old} - 10$			
w(A,$a1_{new}$)		<#3, T1, P_A, A = A-10, A = A+10, #1>	<#3, T1, P_A, RID_A, $a1_{new}$, $a1_{old}$, #1>
	$c_{new} = c_{old} + 20$		
	w(C,c_{new})	<#4, T2, P_C, C = C+20, C-20, #2>	<#4, T2, P_C, RID_C,c_{new}, c_{old}, #2>
r(B, b_{old})			
$b_{new} = b_{old} + 10$			
w(B, b_{new})		<#5, T1, P_B, B=B+10, B=B-10, #3>	<#5, T1, P_B, RID_B, b_{new}, b_{old}, #3>
commit		<#6, T1, commit, #5>	<#6, T1, commit, #5>
	r(A,$a2_{old}$)		
	$a2_{new} = a2_{old} - 20$		
	w(A,$a2_{new}$)	<#7, T2, P_A, A = A-20, A = A+20, #4>	<#7, T2, P_A, RID_A, $a2_{new}$, $a2_{old}$, #4>
	commit	<#8, T2, commit, #7>	<#8, T2, commit, #7>

- mit Hilfe der UNDO- und der REDO-Information ist es also möglich den vorherigen Zustand der Seite zu rekonstruieren.

- bei der **logischen Protokollierung** ist es aber noch wichtig, den Zustand der betroffenen Seite zu kennzeichnen:
  - LSN der zuletzt auf der Seite wirksamen Schreiboperation wird **in der Seite** zusätzlich abgespeichert.
- zwei Fälle können nun unterschieden werden:
  - LSN der Seite < LSN eines Protokolleintrags
  - LSN der Seite >= LSN des Protokolleintrags

Verwaltung der Protokolleinträge



Schreiben der Einträge in die Log-Datei

- folgende Regeln müssen beim Schreiben der Log-Einträge befolgt werden
  - bevor dem commit einer TA müssen alle zugehörigen Protokolleinträge in die Log-Datei geschrieben werden.
  - bevor dem Schreiben einer modifizierten Seite in die (externe) Datenbank müssen alle zugehörigen Protokolleinträge geschrieben werden.
- wenn ein Protokolleintrag mit LSN x in die Log-Datei geschrieben wird, so müssen vorher alle Einträge mit LSN y und $y < x$ geschrieben worden sein.
- Diese Vorgehensweise nennt man auch **Write Ahead Log** (WAL)

9.3.1 Wiederanlauf nach einem Systemfehler

- Ursache: Verlust des Hauptspeichers
- zwei Arten von TA
  - Winner:
    für TA, die bereits mit commit beendet wurden, müssen die durchgeführten Änderungen in der DB nachvollzogen werden (Warum ?)
  - Loser:
    für TA, die zum Zeitpunkt es Absturzes aktiv waren, aber noch nicht mit commit beendet wurden, müssen die Änderungen rückgängig gemacht werden.
- Wiederanlauf geschieht in drei Phasen:
  - Analyse: Bestimme Winner und Loser
  - Wiederholung der Historie (REDO)
  - Zurücksetzen der Loser (UNDO)

Analyse

- sequentielles Durchlaufen der Log-Datei vom Anfang bis zum Ende
  - TA mit einem Eintrag “begin_TA” und einem Eintrag “commit” sind Winner.
  - TA mit einem Eintrag “begin_TA” ohne einem Eintrag “commit” sind Loser.

REDO-Phase:

- sequentielles Durchlaufen der Einträge in der Log-Datei vom Anfang bis zum Ende
  - Lese die zugehörige Seite vom Externspeicher

    Falls LSN der Seite < LSN des Eintrags,
    (i) führe REDO-Operation aus
    (ii) Übertrage die LSN des Eintrags in die Seite.

UNDO-Phase:

- sequentielles Durchlaufen der Einträge in der Log-Datei vom Ende bis zum Anfang
  - führe für jede Loser-TA die UNDO-Operation aus.

  zusätzlich:
  - Schreiben von sogenannten Kompensationseinträgen in die Log-Datei

Fehlerbehandlung beim Wiederanlauf

- Schicksale im Leben einer Datenbank
  - Stromausfall: Verlust es Hauptspeichers
  - Wiederanlauf des Systems
  - erneuter Stromausfall (bevor der Wiederanlauf beendet wurde)
- Anforderung: Idempotenz der UNDO- und REDO-Operationen
  - Ergebnis einer beliebig oft ausgeführten UNDO/REDO-Operation entspricht dem Ergebnis einer einmalig ausgeführten UNDO/REDO-Operation
- REDO-Operationen sind idempotent

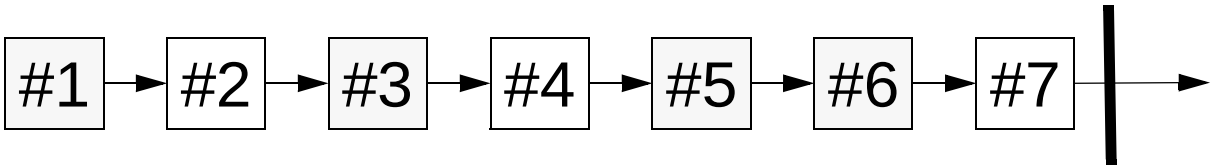
Idempotenz der UNDO-Operation

- Kompensationseinträge in der Protokolldatei sicher gestellt:
- für jede ausgeführte UNDO-Operation wird ein Eintrag erzeugt:
  - besitzt keine UNDO-Information
  - REDO-Information entspricht dabei der (ausgeführten) UNDO-Operation.
  - zusätzlich einen Verweis auf einen Eintrag in der Log-Datei (UNDO_P_LSN)

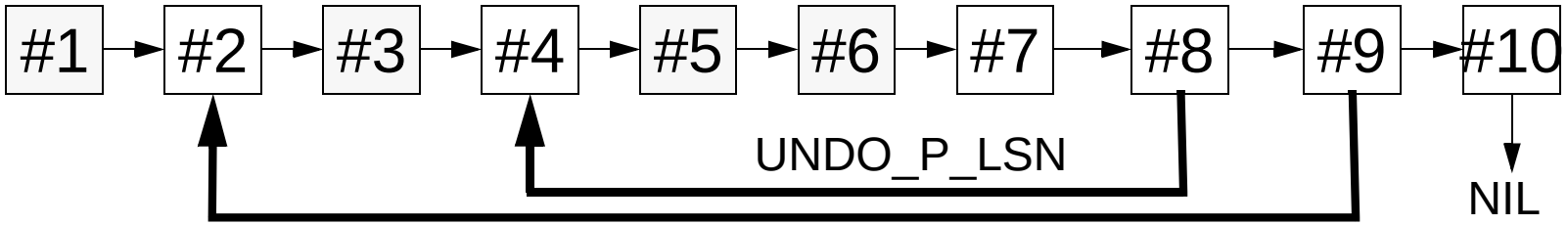
Beispiel:

- Transaktionen T_1 und T_2

Log-Datei (vor dem Wiederanlauf)



Log-Datei nach dem Wiederanlauf



- UNDO_P_LSN
  - Zeiger auf die nächste auszuführende UNDO-Operation der TA.

9.3.2 Rücksetzen einer TA

- System muß eine oder mehrere TAs zurücksetzen (Deadlock)
- Benutzer bricht seine TA ab

- Anforderung:
  - alle DB-Änderungen der TA müssen zurückgenommen werden
  - lokal für eine TA möglich, wenn noch keine Sperren freigegeben wurden

- sequentielles Durchlaufen der Protokolldatei vom Ende bis zum ersten Eintrag der TA, die zurückgesetzt werden soll:
  - Ausführen der UNDO-Operation
  - Eintrag eines Kompensationseintrags
  - Aufsuchen des nächsten Eintrags (mit P_LSN)
- Sperren der TA müssen zusätzlich beim Rücksetzen freigegeben werden.

9.3.3 Sicherungspunkte

- Nachteil: Protokolldateien können sehr groß werden
- Einführung von Sicherungspunkte, so daß idealerweise (!)
  - Wiederanlauf startet am letzten Sicherungspunkt
  - ältere Protokolleinträge können gelöscht werden

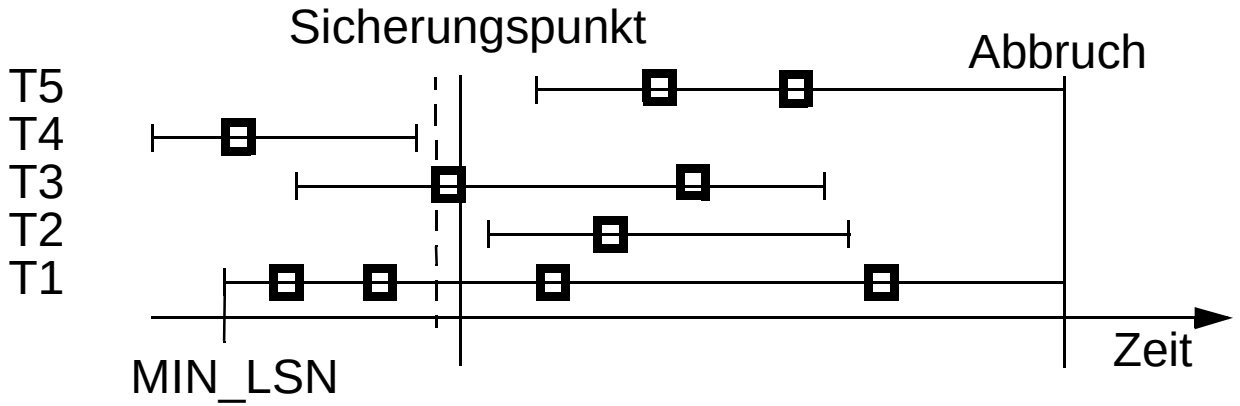
Arten von Sicherungspunkten (savepoints, checkpoints)

- Transaktionskonsistente Sicherungspunkte
  - System wird in einen Ruhezustand überführt (d.h. keine TA ist mehr aktiv)
  - Schreiben aller modifizierten Seiten
  - Neuinitialisierung der Protokolldatei
  - wartende TA können gestartet werden

  Nachteil:
  - Verzögerung der TA
  - großer Aufwand beim Schreiben des Pufferinhalts

Aktionsbasierte Sicherungspunkte

- keine Beruhigung des Systems erforderlich, stattdessen müssen nur die elementaren Änderungsoperationen abgeschlossen werden.



- folgende Aktionen werden ausgeführt
  - Schreiben des Log-Puffers und des DB-Puffers (WAL-Prinzip !)
  - Berechne die Liste S_{TA} aller zum Zeitpunkt des Sicherungpunktes aktiven TAs
  - Berechne MIN_LSN = min {LSN | LSN gehört zu einer TA aus S_{TA}}
- Analys- und REDO-Phase setzt beim Sicherungspunkt auf.
- UNDO-Phase muß aber bis MIN_LSN gehen

Unscharfe Sicherungspunke

- reduziert die Systembelastung durch schrittweises Weitersetzen von Sicherungspunkten

folgende Datenstrukturen werden nun benötigt

- “schmutzige” Pufferseiten sind miteinander verkettet bzgl. der LSN der in der Seite zuletzt ablaufenden Änderung.
  - Seite am Kopf der Liste besitzt die kleinste LSN (MIN_LSN_DIRTY).
  - MIN_LSN_DIRTY ist quasi die LSN des Sicherungspunkts
- Liste S_{TA} enhält alle zum Zeitpunkt MIN_LSN_Dirty aktiven TAs
- MIN_LSN ist die kleinste LSN die in TAs aus S_{TA} vorkommt.

9.3.4 Rekonstruktion

- Rekonstruktion eines korrekten DB-Zustand (mit möglichst wenig Verlust)

Vorgehensweise:

- DB-Kopie, ein sogenannter Dump, wird benötigt zu einem zurückliegenden Zeitpunkt (sehr teuer, deshalb selten)
- verwende Dump und wiederhole alle seitdem beendete TAs
- hierzu werden benötigt:
  - "after images" veränderter Objekte (benötigt bis zum letzten Dump)
  - entsprechende Protokolldatei wird von vorne nach hinten gelesen