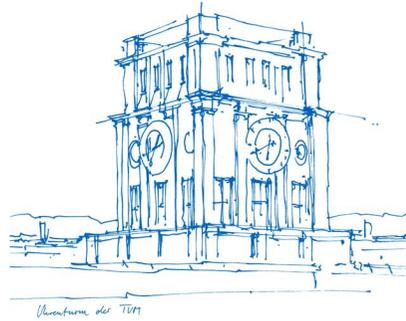


Fehlerbehandlung/Recovery: ACID



Atomarität und Dauerhaftigkeit



Fehlerbehandlung (Recovery)



Fehlerklassifikation

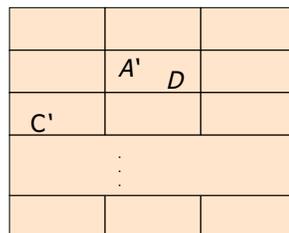
1. Lokaler Fehler in einer noch nicht festgeschriebenen (committed) Transaktion
 - Wirkung muss zurückgesetzt werden
 - **R1-Recovery**
2. Fehler mit Hauptspeicherverlust
 - Abgeschlossene TAs müssen erhalten bleiben
 - **R2-Recovery** → redo
 - Noch nicht abgeschlossene TAs müssen zurückgesetzt werden
 - **R3-Recovery** → undo
3. Fehler mit Hintergrundspeicherverlust
 - **R4-Recovery**

2

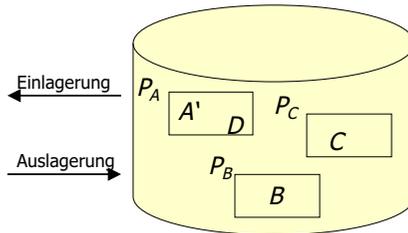
Zweistufige Speicherhierarchie



DBMS-Puffer



Hintergrundspeicher



3

Die Speicherhierarchie



Ersetzung von Puffer-Seiten

- **¬steal**: Bei dieser Strategie wird die 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.
- **¬force**-Strategie: geänderte Seiten können im Puffer verbleiben.

4



Hier zugrunde gelegte Sytemkonfiguration

steal

- „dreckige Seiten“ können in die (persistente) Datenbank (auf Platte) geschrieben werden

~force

- geänderte Seiten sind möglicherweise noch nicht auf die Platte geschrieben

update-in-place

- Es gibt von jeder Seite nur eine Kopie auf der Platte

Kleine Sperrgranulate

- auf Satzebene
- also kann eine Seite gleichzeitig „dreckige“ Daten (einer noch nicht abgeschlossenen TA) und „committed updates“ enthalten
- das gilt sowohl für Puffer – als auch Datenbankseiten

9



Protokollierung von Änderungsoperationen im Log

Struktur der Log-Einträge

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

LSN (Log Sequence Number)

- eine eindeutige Kennung des Log-Eintrags.
- LSNs müssen monoton aufsteigend vergeben werden,
- die chronologische (zeitliche) Reihenfolge der Protokolleinträge kann dadurch ermittelt werden.

Transaktionskennung TA der Transaktion, die die Änderung durchgeführt hat.

PageID

- die Kennung der Seite, auf der die Änderungsoperationen vollzogen wurde.
- Wenn eine Änderung mehr als eine Seite betrifft, müssen entsprechend viele Log-Einträge generiert werden.

10



Protokollierung von Änderungsoperationen II

Struktur der Log-Einträge II

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

Die *Redo* -Information gibt an, wie die Änderung nachvollzogen werden kann.

Die *Undo* -Information beschreibt, wie die Änderung rückgängig gemacht werden kann.

PrevLSN, einen Zeiger auf den vorhergehenden Log-Eintrag der jeweiligen Transaktion. Diesen Eintrag benötigt man aus Effizienzgründen.

11



Beispiel einer Log-Datei

Schritt	T_1	T_2	Log
			[LSN, TA, PageID, Redo, Undo, PrevLSN]
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]

12

Logische oder physische Protokollierung



Physische Protokollierung

Es werden Inhalte / Zustände protokolliert:

1. **before-image** enthält den Zustand vor Ausführung der Operation
2. **after-image** enthält den Zustand nach Ausführung der Operation

Logische Protokollierung

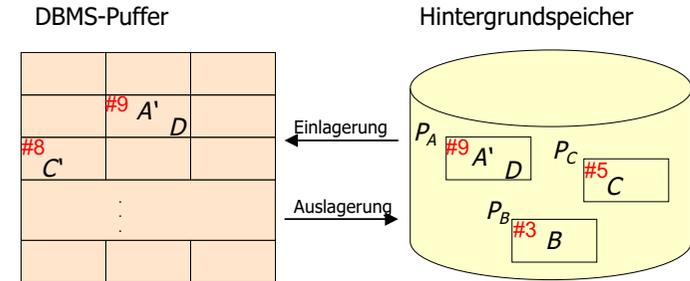
- das *Before-Image* wird durch Ausführung des **Undo-Codes** aus dem *After-Image* generiert und
- das *After-Image* wird durch Ausführung des **Redo-Codes** aus dem *Before-Image* berechnet.

Speicherung der Seiten-LSN

- Die „Herausforderung“ besteht darin, beim Wiederanlauf zu entscheiden, ob man das Before- oder das After-Image auf dem Hintergrundspeicher vorgefunden hat.
- Dazu wird auf jeder Seite die LSN des jüngsten diese Seite betreffenden Log-Eintrags gespeichert.

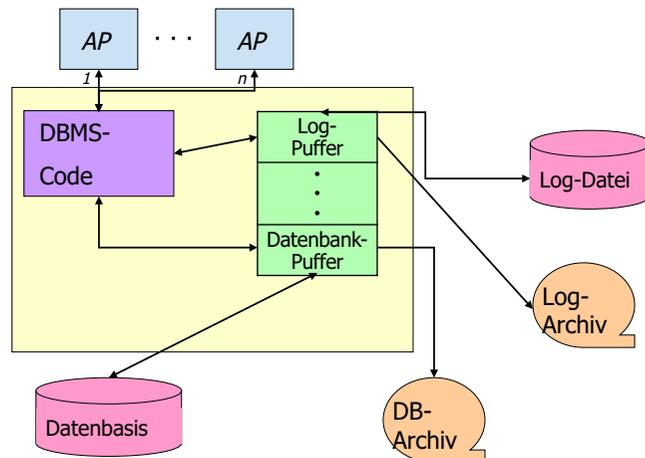
13

Illustration: Seiten-LSN



14

Schreiben der Log-Information



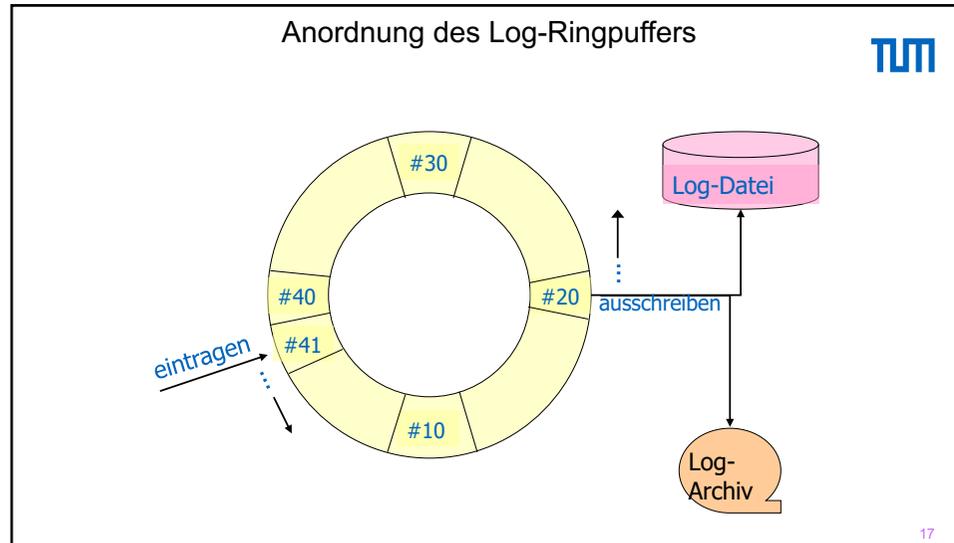
15

Schreiben der Log-Information



- Die Log-Information wird (mindestens) zweimal geschrieben
 1. Log-Datei für schnellen Zugriff
 - R1, R2 und R3-Recovery
 2. Log-Archiv
 - R4-Recovery

16



Das WAL-Prinzip: Write Ahead Log



Write Ahead Log-Prinzip

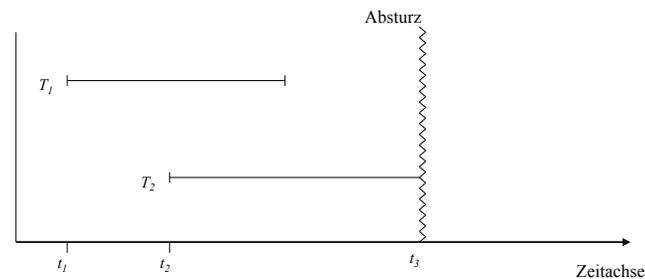
1. Bevor eine Transaktion festgeschrieben (**committed**) wird, müssen alle „zu ihr gehörenden“ Log-Einträge ausgeschrieben sein.
2. Bevor eine modifizierte Seite ausgelagert werden darf, müssen alle Log-Einträge, die zu dieser Seite gehören, in das temporäre und das Log-Archiv ausgeschrieben sein.

Die Einhaltung des WAL-Prinzips wird dadurch sichergestellt, dass der Log-Schreiber periodisch die LSN des zuletzt gesicherten Log-Records abspeichert, so dass die anderen Komponenten darauf warten können, dass die das WAL-Prinzip betreffenden Log-Records im „sicheren Hafen“ sind.

18

Wiederanlauf nach einem Fehler

Transaktionsbeginn und – ende relativ zu einem Systemabsturz



- Transaktionen der Art T_1 müssen hinsichtlich ihrer Wirkung vollständig nachvollzogen werden. Transaktionen dieser Art nennt man *Winner*.
- Transaktionen, die wie T_2 zum Zeitpunkt des Absturzes noch aktiv waren, müssen rückgängig gemacht werden. Diese Transaktionen bezeichnen wir als *Loser*.

Drei Phasen des Wiederanlaufs



1. *Analyse:*
 - Die temporäre Log-Datei wird von Anfang bis zum Ende analysiert,
 - Ermittlung der *Winner*-Menge von Transaktionen des Typs T_1
 - Ermittlung der *Loser*-Menge von Transaktionen der Art T_2 .
2. *Wiederholung der Historie:*
 - *alle* protokollierten Änderungen werden in der Reihenfolge ihrer Ausführung in die Datenbasis eingebracht.
 - Die LSN des Logeintrags wird mit der Seiten-LSN verglichen, um festzustellen, ob **redo** notwendig ist
 - Der LSN-Eintrag auf der Seite wird aktualisiert
3. *Undo der Loser:*
 - Die Änderungsoperationen der *Loser*-Transaktionen werden in umgekehrter Reihenfolge ihrer ursprünglichen Ausführung rückgängig gemacht.

20

Wiederanlauf in drei Phasen

Log _____

1. Analyse →

2. Redo aller Änderungen (*Winner* und *Loser*) →

3. Undo aller *Loser*-Änderungen ←

Fehlertoleranz (Idempotenz) des Wiederanlaufs

$undo(undo(\dots(undo(a))\dots)) = undo(a)$

$redo(redo(\dots(redo(a))\dots)) = redo(a)$

- auch während der Recoveryphase kann das System abstürzen



21

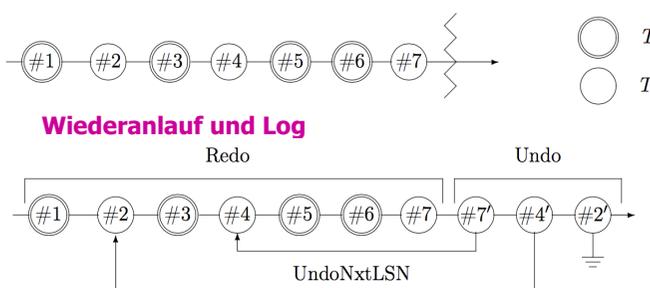
Beispiel einer Log-Datei

Schritt	T_1	T_2	Log
			[LSN, TA, PageID, Redo, Undo, PrevLSN]
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]



22

Kompensationseinträge im Log



Wiederanlauf und Log

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



23



24

Logeinträge nach abgeschlossenem Wiederanlauf

[#1, T₁, BOT, 0]
 [#2, T₂, BOT, 0]
 [#3, T₁, P_A, A-=50, A+=50, #1]
 [#4, T₂, P_C, C+=100, C-=100, #2]
 [#5, T₁, P_B, B+=50, B-=50, #3]
 [#6, T₁, commit, #5]
 [#7, T₂, P_A, A-=100, A+=100, #4]
 <#7', T₂, P_A, A+=100, #7, #4>
 <#4', T₂, P_C, C-=100, #7', #2>
 <#2', T₂, -, -, #4', 0>



25

Logeinträge nach abgeschlossenem Wiederanlauf II

CLRs sind durch spitze Klammern <...> 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)

CLRs enthalten keine Undo-Information

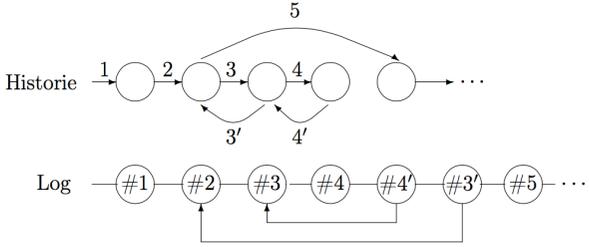
- warum nicht?



26

Lokales Zurücksetzen einer Transaktion

Partielles Zurücksetzen einer Transaktion



Schritte 3 und 4 werden zurückgenommen

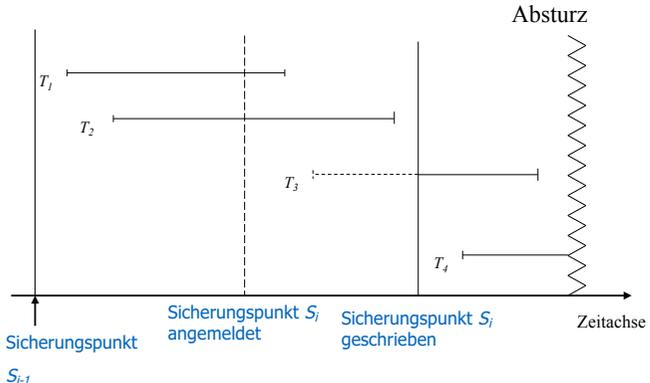
- 4' und 3' sind CLRs
- notwendig für die Realisierung von Sicherungspunkten innerhalb einer TA



27

Sicherungspunkte

Transaktionskonsistente Sicherungspunkte





28

Drei unterschiedliche Sicherungspunkt-Qualitäten

Log Sicherungspunkt

(a) transaktionskonsistent

Analyse →
Redo →
Undo ←

(b) aktionskonsistent

Analyse →
Redo →
Undo →

MinLSN ←

(c) unscharf (fuzzy)

Analyse →
Redo →
Undo →

MinDirtyPageLSN ←
MinLSN ←

29

Aktionenkonsistente Sicherungspunkte

Transaktionsausführung relativ zu einem aktionskonsistenten Sicherungspunkt und einem Systemabsturz

Systemabsturz

Zeitachse

Sicherungspunkt

30

Unschärfe (fuzzy) Sicherungspunkte

modifizierte Seiten werden nicht ausgeschrieben
 nur deren Kennungen (PageIDs) werden gesammelt ausgeschrieben

- *DirtyPages* = Menge der modifizierten Seiten

MinDirtyPageLSN: die minimale LSN, deren Änderungen noch nicht ausgeschrieben wurde

MinLSN: die kleinste LSN aller zum Sicherungszeitpunkt aktiven TAs

31

R4-Recovery / Media-Recovery

Recovery nach einem Verlust der materialisierten Datenbasis

Fehler

konsistente Datenbasis

32