



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

# Transaktionale Informationssysteme

## 5. Logging und Recovery

Norbert Ritter  
Datenbanken und Informationssysteme  
vsis-www.informatik.uni-hamburg.de




◀ ▶

⏪ ⏩

Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

## Anforderungen / Begriffe (1)

- Aufgabe des DBVS
  - Automatische Behandlung aller erwarteten Fehler
- Erwartete Fehler
  - DB-Operation wird zurückgewiesen
  - Commit wird nicht akzeptiert
  - Stromausfall
  - Geräte funktionieren nicht (Spur, Zylinder, Platte defekt)
  - ...
- Besonderheiten der DBS-Fehlerbehandlung
  - Begrenzung und Behebung der zur Laufzeit möglichen Fehler (wie auch bei anderen fehlertoleranten Systemen)
  - „Reparatur“ der statischen Struktur der DB

TAIS – WS0506 – Kapitel 5: Logging und Recovery

2

## Anforderungen / Begriffe (2)

- Allgemeine Probleme
  - Fehlererkennung
  - Fehlereingrenzung
  - Abschätzung des Schadens
  - Durchführung der Recovery
- Fehlermodell von zentralisierten DBVS
  - Transaktionsfehler
  - Systemfehler
  - Gerätefehler
  - Katastrophen
- Voraussetzung
  - Sammeln redundanter Informationen während des Normalbetriebs

© N. Ritter      TAIS – WS0506 – Kapitel 5: Logging und Recovery      3

## Anforderungen / Begriffe (3)

- Transaktionsparadigma verlangt
  - Alles-oder-Nichts-Eigenschaft von Transaktionen
  - Dauerhaftigkeit erfolgreicher Änderungen
- Zielzustand nach erfolgreicher Recovery
  - jüngster transaktionskonsistenter DB-Zustand
    - durch die Recovery-Aktionen ist der jüngste Zustand vor Erkennen des Fehlers wiederherzustellen, der allen semantischen Integritätsbedingungen entspricht, der also ein möglichst aktuelles, exaktes Bild der Miniwelt darstellt
- Zustand der Systemumgebung?
  - Betriebssystem, Anwendungssystem, andere Komponenten

© N. Ritter      TAIS – WS0506      4



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Anforderungen / Begriffe (4)

- Wie soll Recovery durchgeführt werden?
  - Forward-Recovery
    - Non-Stop-Paradigma (Prozesspaare usw.)
    - i. allg. nicht anwendbar
    - Fehlerursache häufig falsche Programme, Eingabefehler u. ä.
    - durch Fehler unterbrochene TA sind zurückzusetzen
  - Backward-Recovery
    - setzt voraus, dass auf allen Abstraktionsebenen genau definiert ist, auf welchen Zustand die DB im Fehlerfall zurückzusetzen ist
    - Zurücksetzen auf konsistenten Zustand und Wiederholung

TAIS – WS0506 – Kapitel 5: Logging und Recovery

5



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Anforderungen / Begriffe (5)

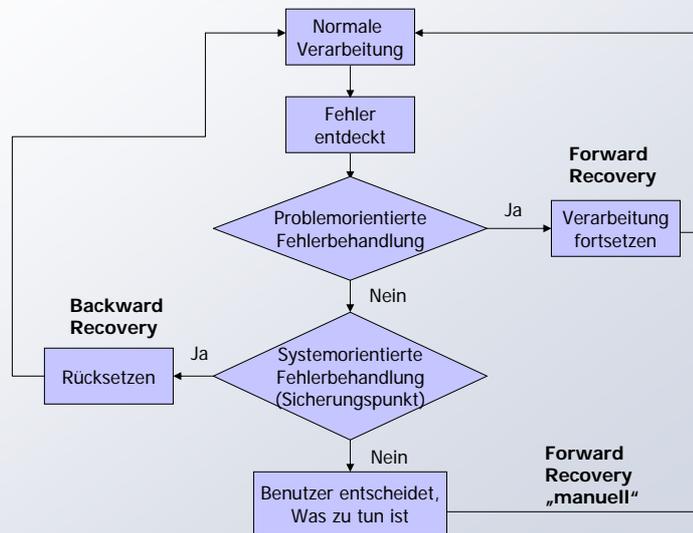
- Aufwand
  - “A recoverable action is 30% harder and requires 20% more code than a non-recoverable action” (J. Gray)
  - Anweisungs- und TA-Atomarität gefordert
    - zwei Prinzipien der Anweisungs-Atomarität möglich
      - „Do things twice“ (vorbereitende Durchführung der Operation; wenn alles OK, erneuter Zugriff und Änderung)
      - „Do things once“ (sofortiges Durchführen der Änderung; wenn Fehler auftritt, internes Zurücksetzen)
    - zweites Prinzip wird häufiger genutzt (ist optimistischer und effizienter)

TAIS – WS0506 – Kapitel 5: Logging und Recovery

6

## Anforderungen / Begriffe (6)

### ■ Grundsätzliche Vorgehensweise



## Anforderungen / Begriffe (7)

### ■ Fehlerarten

Auswirkung eines Fehlers auf	Fehlertyp	Fehlerklassifikation
eine Transaktion	Verletzung von Systemrestriktionen Verstoß gegen Sicherheitsbestimmungen übermäßige Betriebsmittelanforderungen anwendungsbedingte Fehler z. B. falsche Operationen und Werte	Transaktionsfehler
mehrere Transaktionen	geplante Systemschließung Schwierigkeiten bei der Betriebsmittelvergabe Überlast des Systems Verklebung mehrerer Transaktionen	
alle Transaktionen (das gesamte Systemverhalten)	Systemzusammenbruch mit Verlust der Hauptspeichereinhalte Hardware-Fehler Zerstörung von Sekundärspeichern Zerstörung des Rechenzentrums	Systemfehler Gerätefehler Katastrophe



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Anforderungen / Begriffe (8)

- Voraussetzungen für die Wiederherstellung der Daten
  - quasi-stabiler Speicher
  - fehlerfreier DBVS-Code
  - fehlerfreie Log-Daten
  - Unabhängigkeit der Fehler
- Recovery-Arten
  1. **Transaktions-Recovery (R1)**
    - Zurücksetzen einzelner (noch nicht abgeschlossener) Transaktionen im laufenden Betrieb (Transaktionsfehler, Deadlock)
    - Arten
      - Vollständiges Zurücksetzen auf Transaktionsbeginn (TA-UNDO)
      - Partielles Zurücksetzen auf Rücksetzpunkt (Savepoint) innerhalb der Transaktion

TAIS – WS0506 – Kapitel 5: Logging und Recovery

9



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Anforderungen / Begriffe (9)

- Recovery-Arten (Forts.)
  2. **Crash-Recovery (R2)** nach Systemfehler
    - Wiederherstellen des jüngsten transaktionskonsistenten DB-Zustands
    - Notwendige Aktionen
      - (partielles) REDO für erfolgreiche Transaktionen (Wiederholung verlorengegangener Änderungen)
      - UNDO aller durch Ausfall unterbrochenen Transaktionen (Entfernen der Änderungen aus der permanenten DB)
  3. **Medien-Recovery (R3)** nach Gerätefehler
    - Spiegelplatten bzw.
    - Vollständiges Wiederholen (REDO) aller Änderungen (erfolgreich abgeschlossener Transaktionen) auf einer Archivkopie

TAIS – WS0506 – Kapitel 5: Logging und Recovery

10

## Anforderungen / Begriffe (10)

- Recovery-Arten (Forts.)

- 4. **Katastrophen-Recovery** (R4)

- Nutzung einer aktuellen DB-Kopie in einem ‚entfernten‘ System oder
    - Stark verzögerte Fortsetzung der DB-Verarbeitung mit repariertem/neuem System auf der Basis gesicherter Archivkopien (Datenverlust)



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

## Anforderungen / Begriffe (11)

- Nicht systematisiert

- R5-Recovery

- Log-Daten sind fehlerhaft oder DB-Strukturen (ohne Log-Daten) sind unbrauchbar
    - kein TA-konsistenter, bestenfalls aktions- oder gerätekonsistenter Zustand erreichbar

- R6-Recovery

- Zusammenfassung aller Maßnahmen außerhalb des Systems
      - Kompensations-TA und
      - manuelle Behandlung der Auswirkungen



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

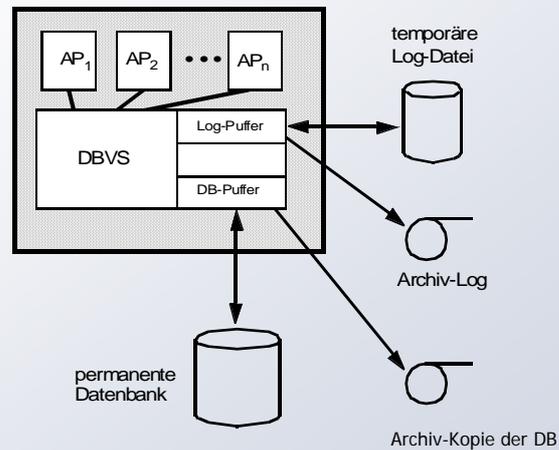
Recovery



© N. Ritter

## Anforderungen / Begriffe (12)

### ■ DB-Recovery – Systemkomponenten



## Anforderungen / Begriffe (13)

### ■ DB-Recovery – Systemkomponenten (Forts.)

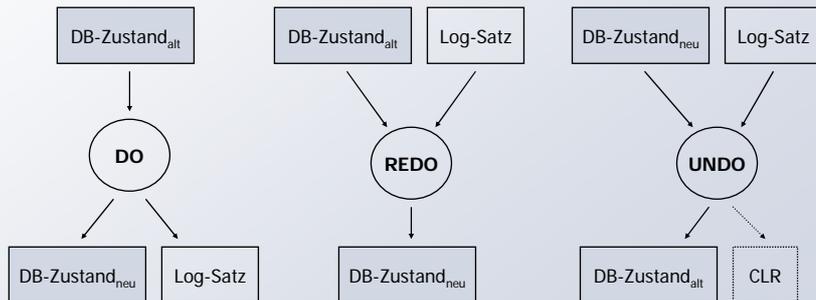
- Pufferung von Log-Daten im Hauptspeicher (Log-Puffer)
  - Ausschreiben spätestens bei Commit
- Einsatz der Log-Daten
  - Temporäre Log-Datei zur Behandlung von Transaktions- und Systemfehlern
    - DB + temp. Log ⇒ DB
  - Behandlung von Gerätefehlern
    - Archiv-Kopie + Archiv-Log ⇒ DB

## Logging (1)

### Aufgaben

- Sammlung redundanter Daten bei Änderungen im Normalbetrieb (DO) als Voraussetzung für Recovery
- Einsatz im Fehlerfall (Undo-, Redo-Recovery)

### Do-Redo-Undo-Prinzip



CLR: Compensation Log Record (für Crash während Recovery)



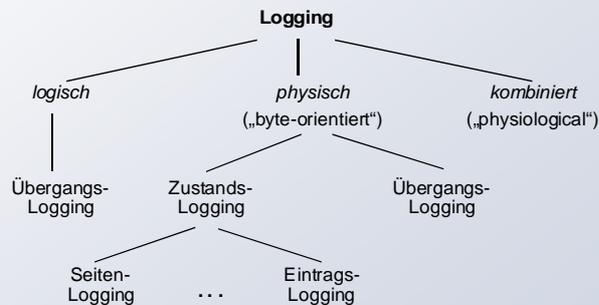
© N. Ritter

## Logging (2)

### Logging kann auf jeder (DBVS-)Systemebene erfolgen

- Sammeln von ebenenspezifischer Log-Information setzt voraus, dass bei der Recovery die Konsistenzbedingungen der darunter liegenden Abbildungsschicht im DB-Zustand erfüllt sind

### Logging-Verfahren: Klassifikation



© N. Ritter



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Logging (3)

---

- Logisches Logging
  - Protokollierung der ändernden DML-Befehle mit ihren Parametern
  - Generelles Problem
    - mengenorientierte Aktualisierungsoperation (z. B. DELETE <relation>)
  - UNDO-Probleme v.a. bei nicht-relationalen Systemen
    - z.B. Löschen einer Hierarchie von Set-Ausprägungen – ERASE ALL
  - Voraussetzung
    - nach einem Systemausfall müssen auf der permanenten Datenbank DML-Operationen ausführbar sein, d.h., sie muss wenigstens speicherkonsistent sein (Aktionskonsistenz)
  - verzögerte (indirekte) Einbringstrategie erforderlich

TAIS – WS0506 – Kapitel 5: Logging und Recovery

17



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Logging (4)

---

- Physisches Logging
  - Log-Granulat: Seite vs. Eintrag/Satz
  - Zustands-Logging
    - alte Zustände (Before-Images) und neue Zustände (After-Images) geänderter Objekte werden in die Log-Datei geschrieben
  - Übergangs-Logging
    - Protokollierung der Differenz zwischen Before- und After-Image
  - physisches Logging ist bei direkten und verzögerten Einbringstrategien anwendbar
- Probleme logischer und physischer Logging-Verfahren
  - Logisches Logging: für Update-in-Place nicht anwendbar
  - Physisches, „byte-orientiertes“ Logging: aufwendig und unnötig starr v.a. bezüglich Löschen- und Einfügeoperationen

TAIS – WS0506 – Kapitel 5: Logging und Recovery

18

## Logging (5)

### ■ Physiologisches Logging

- Kombination physische/logische Protokollierung
  - physical-to-a-page, logical-within-a-page
  - Protokollierung von elementaren Operationen innerhalb einer Seite
  - jeder Log-Satz bezieht sich auf eine Seite
  - Technik ist mit Update-in-Place verträglich



Navigation icons: back, home, forward.

Sidebar menu:

- Begriffe
- Logging
- Komponenten
- Sicherungs-punkte
- Recovery

Logo: vsis

© N. Ritter

## Logging (6)

### ■ Anwendungsbeispiel

- Änderungen bezüglich einer Seite A
  1. ein Objekt  $a$  wird in Seite A eingefügt ( $A_1 \rightarrow A_2$ )
  2. in A wird ein bestehendes Objekt  $b_{alt}$  nach  $b_{neu}$  geändert ( $A_2 \rightarrow A_3$ )

	<i>logisch</i>	<i>physisch</i>
<i>Zustände</i>		Protokollierung der Before- und After-Images 1. $A_1$ und $A_2$ 2. $A_2$ und $A_3$
<i>Übergänge</i>	Protokollierung der Operationen mit Parameter 1. Insert ( $a$ ) 2. Update ( $b_{alt}, b_{neu}$ )	Differenzen-Logging 1. $A_1 \oplus A_2$ 2. $A_2 \oplus A_3$



Navigation icons: back, home, forward.

Sidebar menu:

- Begriffe
- Logging
- Komponenten
- Sicherungs-punkte
- Recovery

Logo: vsis

© N. Ritter

## Logging (7)

- Anwendungsbeispiel (Forts.)

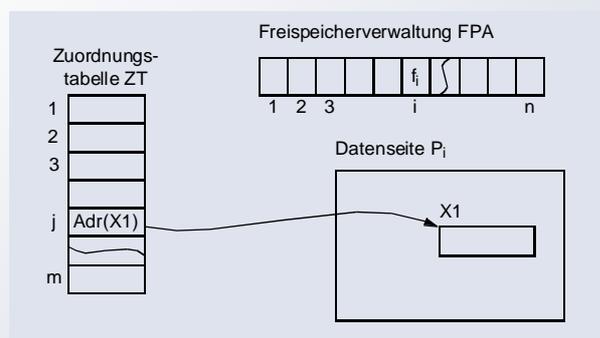
- Rekonstruktion von Seiten beim Differenzen-Logging
  - A1 als Anfangs- oder A3 als Endzustand seien verfügbar
  - REDO-Recovery
    - $A_1 \oplus (A_1 \oplus A_2) = A_2$
    - $A_2 \oplus (A_2 \oplus A_3) = A_3$
  - UNDO-Recovery
    - $A_3 \oplus (A_2 \oplus A_3) = A_2$
    - $A_2 \oplus (A_1 \oplus A_2) = A_1$



## Logging (8)

- Aufwand bei physischem Zustandslogging

- einfachste Form der Implementierung: Seiten-Logging



## Logging (9)

- Aufwand bei physischem Zustandslogging (Forts.)
  - Operation: STORE X-RECORD (X1)

<i>Aufwand</i>	<i>Datenseite</i>	<i>ZT</i>	<i>FPA</i>	<i>n Zugriffspfadseiten</i>
<i>normaler Betrieb (DO)</i>	neues $P_i$	Adr(X1)	$f_i$	$n DS_{neu}$
<i>UNDO-Log</i>	altes $P_i$	alter Inhalt	alter Inhalt	$n DS_{salt}$
<i>REDO-Log</i>	neues $P_i$	Adr(X1)	$f_i$	$n DS_{neu}$



Begriffe

Logging

Komponenten

Sicherungspunkte

Recovery



© N. Ritter

## Logging (10)

- Bewertung der Logging-Verfahren

	Logging-Aufwand im Normalbetrieb	Restart-Aufwand im Fehlerfall (Crash)
Seitenzustands-Logging	--	+
Seitenübergangs-Logging (Differenzen)	-	+
Eintrags-Logging / physiologisches Logging	+	+
Logisches Logging	++	--



Begriffe

Logging

Komponenten

Sicherungspunkte

Recovery



© N. Ritter



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Logging (11)

---

- Eintrags-Logging vs. Seiten-Logging
  - Vorteile von Eintrags-Logging
    - geringerer Platzbedarf
    - weniger Log-E/As
    - erlaubt bessere Pufferung von Log-Daten (Gruppen-Commit)
    - unterstützt feine Synchronisationsgranulate (Seiten-Logging → Synchronisation auf Seitenebene)
  - Nachteil von Eintrags-Logging
    - Recovery ist komplexer als mit Seiten-Logging
      - z.B. müssen Seiten vor der Anwendung der Log-Sätze wieder in den Speicher eingelesen werden

TAIS – WS0506 – Kapitel 5: Logging und Recovery

25



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Logging (12)

---

- Aufbau der (temporären) Log-Datei
  - Verschiedene Satzarten erforderlich
    - BOT-, Commit-, Abort-Satz
    - Änderungssatz (UNDO-Informationen, z. B. ‚Before-Images‘, und REDO-Informationen, z. B. ‚After-Images‘)
    - Sicherungspunkt-Sätze
  - Protokollierung von Änderungsoperationen
    - Struktur der Log-Einträge:  
[LSN, TAID, PageID, Redo, Undo, PrevLSN]
    - LSN: Log Sequence Number
      - eindeutige Kennung des Log-Eintrags
      - LSNs müssen monoton aufsteigend vergeben werden
      - chronologische Reihenfolge der Protokolleinträge kann dadurch ermittelt werden

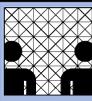
TAIS – WS0506 – Kapitel 5: Logging und Recovery

26

## Logging (13)

### ■ Aufbau der (temporären) Log-Datei (Forts.)

- Transaktionskennung TAID der TA, die die Änderung durchgeführt hat
- PageID
  - Kennung der Seite, auf der die Änderungsoperation vollzogen wurde
  - wenn die Änderung mehr als eine Seite betrifft, müssen entsprechend viele Log-Einträge generiert werden
- Redo
  - Redo-Information gibt an, wie die Änderung nachvollzogen werden kann
- Undo
  - Undo-Information beschreibt, wie die Änderung rückgängig gemacht werden kann
- PrevLSN
  - ist ein Zeiger auf den vorhergehenden Log-Eintrag der jeweiligen TA
  - diesen Eintrag benötigt man aus Effizienzgründen



- Begriffe
- Logging
- Komponenten
- Sicherungspunkte
- Recovery

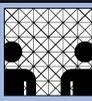


## Logging (14)

### ■ Aufbau der (temporären) Log-Datei (Forts.)

- Beispiel

Schritt	T <sub>1</sub>	T <sub>2</sub>	Log
			[LSN, TAID, PageID, Redo, Undo, PrevLSN]
1.	<b>BOT</b>		[#1, T <sub>1</sub> , <b>BOT</b> , 0]
2.	r(A, a <sub>1</sub> )		
3.		<b>BOT</b>	[#2, T <sub>2</sub> , <b>BOT</b> , 0]
4.		r(C, c <sub>2</sub> )	
5.	a <sub>1</sub> := a <sub>1</sub> - 50		
6.	w(A, a <sub>1</sub> )		[#3, T <sub>1</sub> , P <sub>A</sub> , A-=50, A+=50, #1]
7.		c <sub>2</sub> := c <sub>2</sub> + 100	
8.		w(C, c <sub>2</sub> )	[#4, T <sub>2</sub> , P <sub>C</sub> , C+=100, C-=100, #2]
9.	r(B, b <sub>1</sub> )		
10.	b <sub>1</sub> := b <sub>1</sub> + 50		
11.	w(B, b <sub>1</sub> )		[#5, T <sub>1</sub> , P <sub>B</sub> , B+=50, B-=50, #3]
12.	<b>Commit</b>		[#6, T <sub>1</sub> , <b>Commit</b> , #5]
13.		r(A, a <sub>2</sub> )	
14.		a <sub>2</sub> := a <sub>2</sub> - 100	
15.		w(A, a <sub>2</sub> )	[#7, T <sub>2</sub> , P <sub>A</sub> , A-=100, A+=100, #4]
16.		<b>Commit</b>	[#8, T <sub>2</sub> , <b>Commit</b> , #7]



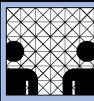
- Begriffe
- Logging
- Komponenten
- Sicherungspunkte
- Recovery



## Logging (15)

### ■ Aufbau der (temporären) Log-Datei (Forts.)

- Log ist eine sequentielle Datei
  - Schreiben neuer Protokolldaten an das aktuelle Dateiende
- Log-Daten sind für Crash-Recovery nur begrenzte Zeit relevant
  - Undo-Sätze für erfolgreich beendete TA werden nicht mehr benötigt
  - nach Einbringen der Seite in die DB wird Redo-Information nicht mehr benötigt
  - Redo-Information für Medien-Recovery ist im Archiv-Log zu sammeln!



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

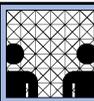
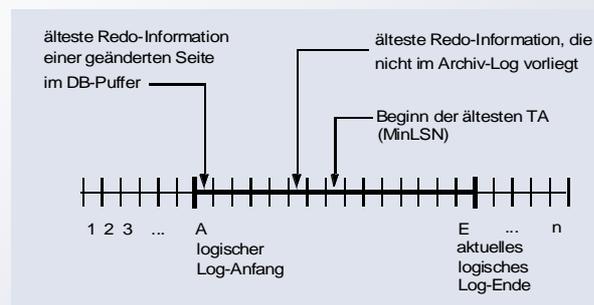
TAIS – WS0506 – Kapitel 5: Logging und Recovery

29

## Logging (16)

### ■ Aufbau der (temporären) Log-Datei (Forts.)

- Ringpuffer-Organisation der Log-Datei



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

30









**Abhängigkeiten der Systemkomponenten (1)**

- Überblick über die wichtigsten Abhängigkeiten
  - Einbringstrategie
    - direktes Einbringen von Änderungen (*non-atomic*)
    - verzögertes Einbringen von Änderungen (*atomic*)
  - Ersetzungsstrategie
    - Verdrängen 'schmutziger' Seiten (*steal*)
    - nur Verdrängung von Seiten abgeschlossener TAs (*nosteal*)
  - Ausschreibstrategie
    - Ausschreiben notwendigerweise bei Commit (*force*)
    - Ausschreiben möglicherweise nach Commit (*noforce*)
  - Wahl des Sperrgranulats
  - Commit-Behandlung

© N. Ritter TAIS – WS0506 – Kapitel 5: Logging und Recovery 31









**Abhängigkeiten der Systemkomponenten (2)**

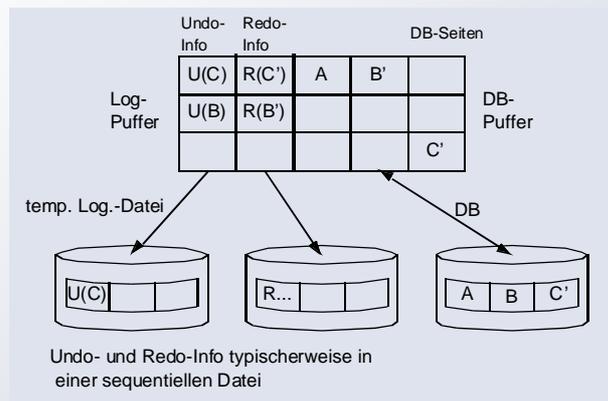
- Einbringstrategie
  - *non-atomic / direkt / update-in-place*
    - geänderte Seite wird immer in denselben Block auf Platte zurückgeschrieben
    - Schreiben ist gleichzeitig Einbringen in die permanente DB
    - ‚atomares‘ Einbringen mehrerer geänderter Seiten ist nicht möglich (non-atomic)
    - Forderungen
      - WAL-Prinzip: *Write Ahead Log* für Undo-Info; U(B) vor B' (vgl. nachfolgende Folie)
      - Ausschreiben der Redo-Info spätestens bei Commit; R(C') + R(B') vor Commit (vgl. nachfolgende Folie)

© N. Ritter TAIS – WS0506 – Kapitel 5: Logging und Recovery 32

## Abhängigkeiten der Systemkomponenten (3)

### ■ Einbringstrategie (Forts.)

- *non-atomic / direkt / update-in-place (Forts.)*



## Abhängigkeiten der Systemkomponenten (4)

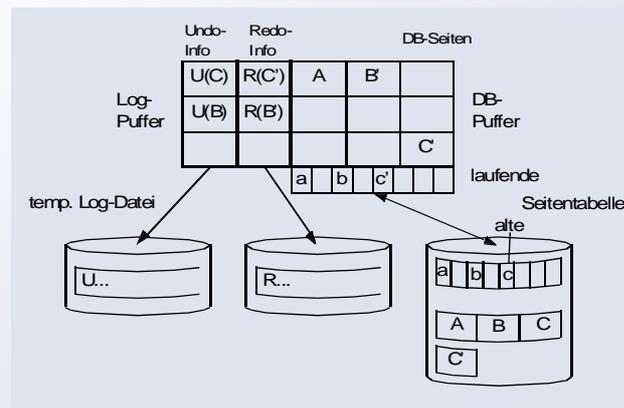
### ■ Einbringstrategie (Forts.)

- *atomic / verzögert*
  - z. B. in System R, SQL/DS
  - geänderte Seite wird in separaten Block auf Platte geschrieben, Einbringen in die DB erfolgt später
  - Seitentabelle gibt aktuelle Adresse einer Seite an
  - verzögertes, atomares Einbringen mehrerer Änderungen ist durch Umschalten von Seitentabellen möglich
  - aktions- oder transaktionskonsistente DB auf Platte
  - logisches Logging anwendbar
  - Forderungen
    - WAL-Prinzip bei verzögertem Einbringen:  
U(C) + U(B) vor Sicherungspunkt
    - R(C') + R(B') spätestens bei Commit



## Abhängigkeiten der Systemkomponenten (5)

- Einbringstrategie (Forts.)
  - *atomic / verzögert (Forts.)*



## Abhängigkeiten der Systemkomponenten (6)

- Ersetzungsstrategie
  - Problem: Ersetzung ‚schmutziger‘ Seiten
  - *steal*
    - geänderte Seiten können jederzeit, insbesondere vor EOT der ändernden TA, ersetzt und in die permanente DB eingebracht werden
    - große Flexibilität zur Seitenersetzung
    - Undo-Recovery vorzusehen (TA-Abbruch, Systemfehler)
    - *steal* erfordert Einhaltung des WAL-Prinzips, d.h., vor dem Einbringen einer schmutzigen Änderung müssen zugehörige Undo-Informationen (z. B. Before-Images) in die Log-Datei geschrieben werden
  - *nosteal*
    - Seiten mit schmutzigen Änderungen dürfen nicht ersetzt werden
    - es ist keine Undo-Recovery auf der permanenten DB vorzusehen
    - Probleme bei langen Änderungs-TA

## Abhängigkeiten der Systemkomponenten (7)

### ■ Ausschreibstrategie

- *force*
  - alle geänderten Seiten werden spätestens bei EOT (Commit) in die permanente DB eingebracht (Durchschreiben)
  - keine Redo-Recovery nach Rechnerausfall
  - hoher Schreibaufwand
  - große DB-Puffer werden schlecht genutzt
  - Antwortzeitverlängerung für Änderungs-TA
- *noforce*
  - kein Durchschreiben der Änderungen bei EOT
  - beim Commit werden lediglich Redo-Informationen in die Log-Datei geschrieben
  - Redo-Recovery nach Rechnerausfall
- Commit-Regel
  - bevor das Commit einer TA ausgeführt werden kann, sind für ihre Änderungen ausreichende Redo-Informationen (z. B. *After-Images*) zu sichern

## Abhängigkeiten der Systemkomponenten (8)

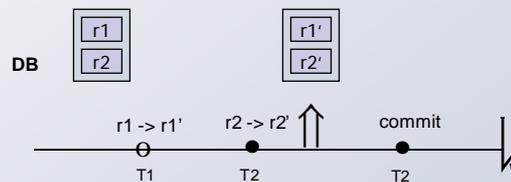
### ■ Auswirkungen von Ersetzungs- und Ausschreibstrategie auf Recovery-Maßnahmen

	steal	no steal
force	UNDO NO REDO	NO UNDO NO REDO
noforce	UNDO REDO	NO UNDO REDO

## Abhängigkeiten der Systemkomponenten (9)

### ■ Sperrverwaltung

- Log-Granulat muss kleiner oder gleich Sperrgranulat sein
  - Beispiel
    - Sperren auf Satzebene
    - *Before- bzw. After-Images* auf Seitenebene
    - Undo (Redo) einer Änderung kann parallel durchgeführte Änderungen derselben Seite überschreiben (*lost update*)



© N. Ritter

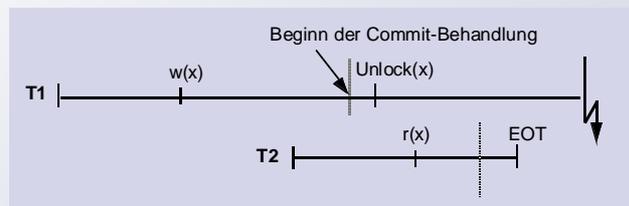
TAIS – WS0506 – Kapitel 5: Logging und Recovery

39

## Abhängigkeiten der Systemkomponenten (10)

### ■ Commit-Behandlung

- Forderungen
  - Änderungen einer TA sind mit Commit zuzusichern
  - andere TA dürfen Änderungen erst sehen, wenn ‚Durchkommen‘ der ändernden TA gewährleistet ist (Problem des rekursiven Zurücksetzens)



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

40



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Abhängigkeiten der Systemkomponenten (11)

- Commit-Behandlung (Forts.)
  - zweiphasige Commit-Bearbeitung
    - Phase 1: Wiederholbarkeit der TA sichern
      - ggf. noch Änderungen sichern
      - Commit-Satz auf Log schreiben
    - Phase 2: Änderungen sichtbar machen (Freigabe der Sperren)
    - Benutzer kann nach Phase 1 vom erfolgreichen Ende der TA informiert werden (Ausgabenachricht)
    - Beispiel: Commit-Behandlung bei *force*, *steal*:
      1. Before-Images auf Log schreiben
      2. Force der geänderten DB-Seiten
      3. After-Images (für Archiv-Log) und Commit-Satz schreiben
bei NoForce lediglich 3. für erste Commit-Phase notwendig

TAIS – WS0506 – Kapitel 5: Logging und Recovery

41



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Abhängigkeiten der Systemkomponenten (12)

- Commit-Behandlung (Forts.)
  - Gruppen-Commit
    - Log-Datei ist potentieller Leistungsengpass
      - pro Änderungstransaktion wenigstens 1 Log-E/A
      - max. ca. 250 sequentielle Schreibvorgänge pro Sekunde (1 Platte)
    - Gruppen-Commit bedeutet gemeinsames Schreiben der Log-Daten von mehreren TA
      - Pufferung der Log-Daten in Log-Puffer (1 oder mehrere Seiten)
      - Voraussetzung: Eintrags-Logging
      - Ausschreiben des Log-Puffers erfolgt, wenn er voll ist bzw. Timer abläuft
      - nur geringe Commit-Verzögerung

TAIS – WS0506 – Kapitel 5: Logging und Recovery

42

## Abhängigkeiten der Systemkomponenten (13)

### ■ Commit-Behandlung (Forts.)

#### • Gruppen-Commit (Forts.)

- Gruppen-Commit erlaubt Reduktion auf 0.1 - 0.2 Log-E/As pro TA
  - Einsparung an CPU-Overhead für E/A reduziert CPU-Wartezeiten
  - dynamische Festsetzung des Timer-Wertes durch DBVS wünschenswert
- Gruppen-Commit ermöglicht demnach Durchsatzverbesserung v.a. bei Log-Engpass oder hoher CPU-Auslastung



Begriffe

Logging

**Komponenten**

Sicherungs-  
punkte

Recovery



© N. Ritter

## Abhängigkeiten der Systemkomponenten (14)

### ■ Commit-Behandlung (Forts.)

#### • Vergleich verschiedener Commit-Verfahren

- Standard-2PC



- Gruppen-Commit



Begriffe

Logging

**Komponenten**

Sicherungs-  
punkte

Recovery



© N. Ritter

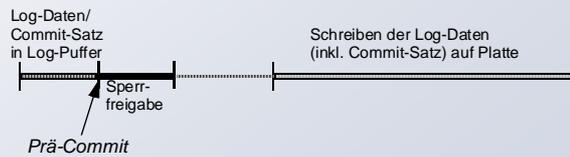
## Abhängigkeiten der Systemkomponenten (15)

### ■ Commit-Behandlung (Forts.)

#### • Vergleich verschiedener Commit-Verfahren (Forts.)

##### - Weitere Optimierungsmöglichkeit: *Prä-Commit*

- Sperren bereits freigeben, wenn Commit-Satz im Log-Puffer steht (vor Schreiben auf Log-Platte)
- TA kann nur noch durch Systemfehler scheitern
- In diesem Fall scheitern auch alle ‚abhängigen‘ TA, die ungesicherte Änderungen aufgrund der vorzeitigen Sperrfreigabe gesehen haben



- In allen drei Verfahren wird der Benutzer erst nach Schreiben des Commit-Satzes auf Platte vom TA-Ende informiert

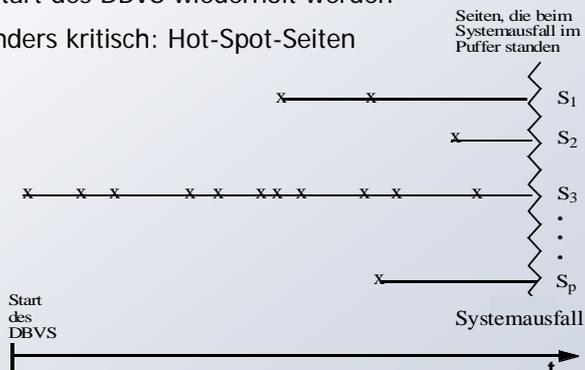


© N. Ritter

## Sicherungspunkte (1)

### ■ Motivation: Sicherungspunkte / Checkpoints

- Maßnahme zur Begrenzung des Redo-Aufwandes nach Systemfehlern (noforce)
- ohne Sicherungspunkte müssten potentiell alle Änderungen seit Start des DBVS wiederholt werden
- Besonders kritisch: Hot-Spot-Seiten



© N. Ritter



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Sicherungspunkte (2)

- Verwaltungsinformation
  - Log-Datei
    - BEGIN\_CHKPT-Satz
    - Sicherungspunkt-Informationen, u. a. Liste der aktiven TA
    - END\_CHKPT-Satz
  - Log-Adresse des letzten Sicherungspunkt-Satzes wird in spezieller Restart-Datei geführt
- Sicherungspunkte und Einbringverfahren
  - *atomic*
    - Zustand der permanenten DB beim Crash entspricht dem zum Zeitpunkt des letzten erfolgreichen Sicherungspunktes
  - *non-atomic*
    - Zustand der permanenten DB enthält alle ausgeschriebenen (eingebrachten) Änderungen bis zum Crash

TAIS – WS0506 – Kapitel 5: Logging und Recovery

47



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Sicherungspunkte (3)

- Arten von Sicherungspunkten
  - Direkte Sicherungspunkte
    - alle geänderten Seiten im DB-Puffer werden in die permanente DB eingebracht
    - Redo-Recovery beginnt bei letztem Sicherungspunkt
    - Nachteil: lange ‚Totzeit‘ des Systems, da während des Sicherungspunktes keine Änderungen durchgeführt werden können
    - Problem wird durch große Hauptspeicher verstärkt
    - *Transaktionskonsistente* oder *aktionskonsistente* Sicherungspunkte

TAIS – WS0506 – Kapitel 5: Logging und Recovery

48

## Sicherungspunkte (4)

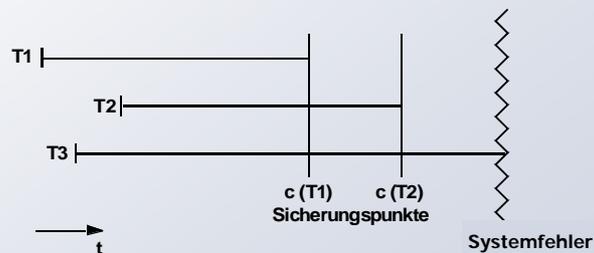
- Arten von Sicherungspunkten (Forts.)
  - Indirekte/Unschärfe Sicherungspunkte (Fuzzy Checkpoints)
    - kein Hinauszwingen geänderter Seiten
    - nur Statusinformationen (Pufferbelegung, Menge aktiver TA, offene Dateien etc.) werden in die Log-Datei geschrieben
    - sehr geringer Sicherungspunkt-Aufwand
    - Redo-Informationen vor letztem Sicherungspunkt sind i. allg. noch zu berücksichtigen
    - Sonderbehandlung von Hot-Spot-Seiten

Begriffe  
Logging  
Komponenten  
Sicherungspunkte  
Recovery

© N. Ritter

## Sicherungspunkte (5)

- Transaktionsorientierte Sicherungspunkte
  - Force kann als spezieller Sicherungspunkt-Typ aufgefasst werden: nur Seiten einer TA werden ausgeschrieben
  - Sicherungspunkt bezieht sich immer auf genau eine TA (TOC = transaction-oriented checkpoint  $\equiv$  *force*)



Begriffe  
Logging  
Komponenten  
Sicherungspunkte  
Recovery

© N. Ritter

## Sicherungspunkte (6)

### ■ Transaktionsorientierte Sicherungspunkte (Forts.)

- Eigenschaften
  - EOT-Behandlung erzwingt das Ausschreiben aller geänderten Seiten der TA aus dem DB-Puffer
    - Übernahme aller Änderungen in die DB
    - Vermerk in Log-Datei
  - nur *atomic* ermöglicht atomares Einbringen mehrerer Seiten
  - zumindest bei direktem Einbringen der Seiten ist demnach Undo-Recovery vorzusehen (*steal*)
- Abhängigkeit: *non-atomic, force* => *steal*

Navigation icons: back, home, forward.

Sidebar menu:

- Begriffe
- Logging
- Komponenten
- Sicherungspunkte**
- Recovery

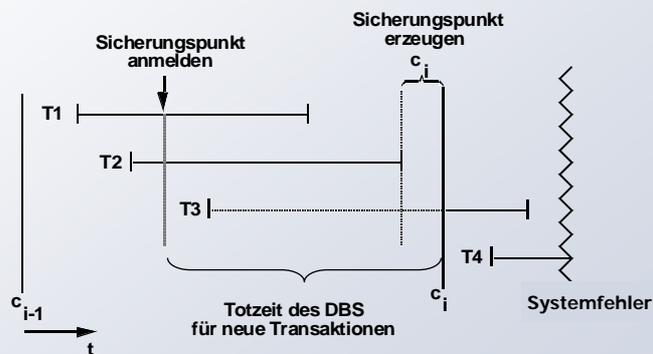
Logo: vsis

© N. Ritter

## Sicherungspunkte (7)

### ■ Transaktionskonsistente Sicherungspunkte

- Sicherungspunkt bezieht sich immer auf alle TA  
(TCC = transaction-consistent checkpoint: logisch konsistent)



Navigation icons: back, home, forward.

Sidebar menu:

- Begriffe
- Logging
- Komponenten
- Sicherungspunkte**
- Recovery

Logo: vsis

© N. Ritter

## Sicherungspunkte (8)

### ■ Transaktionskonsistente Sicherungspunkte (Forts.)

- Eigenschaften
  - Ausschreiben ist bis zum Ende aller aktiven Änderungs-TA zu verzögern
  - neue Änderungs-TA müssen warten, bis Erzeugung des Sicherungspunkts beendet ist
  - Crash-Recovery startet bei letztem Sicherungspunkt

Navigation und Menü:

- ◀ ▶
- ⏪ ⏩
- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte**
- Recovery

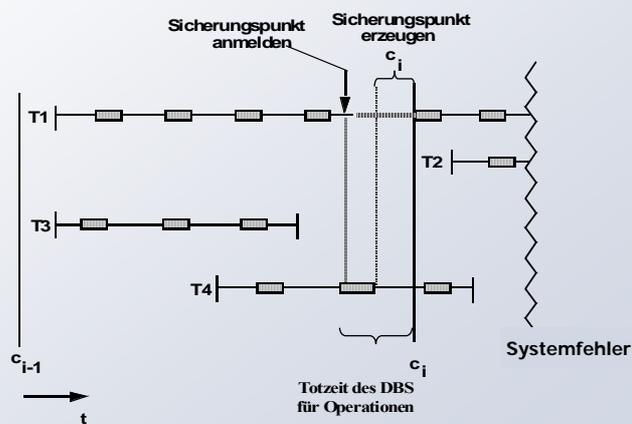
Logo: vsis

© N. Ritter

## Sicherungspunkte (9)

### ■ Aktionskonsistente Sicherungspunkte

- Sicherungspunkt bezieht sich immer auf alle TA  
(ACC = action-consistent checkpoint: speicherkonsistent)



Navigation und Menü:

- ◀ ▶
- ⏪ ⏩
- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte**
- Recovery

Logo: vsis

© N. Ritter



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Sicherungspunkte (10)

- Aktionskonsistente Sicherungspunkte (Forts.)
  - Eigenschaften
    - keine Änderungsanweisungen während des Sicherungspunktes
    - geringere Totzeiten als bei TCC, dafür Verminderung der Qualität der Sicherungspunkte
    - Crash-Recovery wird nicht durch letzten Sicherungspunkt begrenzt
  - Abhängigkeit: ACC => *steal*

TAIS – WS0506 – Kapitel 5: Logging und Recovery

55



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Sicherungspunkte (11)

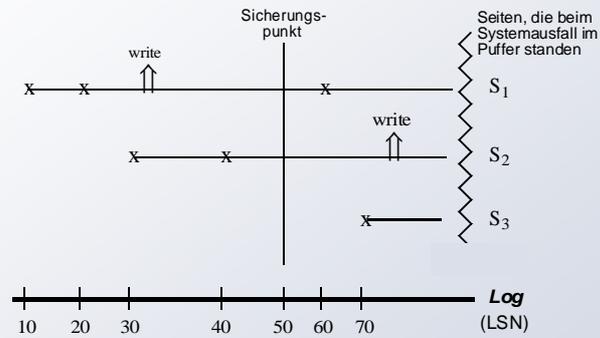
- Unscharfe Sicherungspunkte (Fuzzy Checkpoints)
  - DB auf Platte bleibt ‚fuzzy‘, nicht aktionskonsistent
    - nur bei Update-in-Place (*non-atomic*) relevant
  - Problem
    - Bestimmung der Log-Position, an der Redo-Recovery beginnen muss
    - Pufferverwalter vermerkt sich zu jeder geänderten Seite StartLSN, d. h. Log-Satz-Adresse der ersten Änderung seit Einlesen von Platte
    - Redo-Recovery nach Crash beginnt bei MinDirtyPageLSN (= MIN(StartLSN))
  - Sicherungspunkt-Information
    - MinDirtyPageLSN, Liste der aktiven TA und ihrer StartLSNs, ...

TAIS – WS0506 – Kapitel 5: Logging und Recovery

56

## Sicherungspunkte (12)

### ■ Unscharfe Sicherungspunkte (Forts)



- geänderte Seiten werden asynchron ausgeschrieben
  - ggf. Kopie der Seite anlegen (für Hot-Spot-Seiten)
  - Seite ausschreiben
  - StartLSN anpassen / zurücksetzen



© N. Ritter

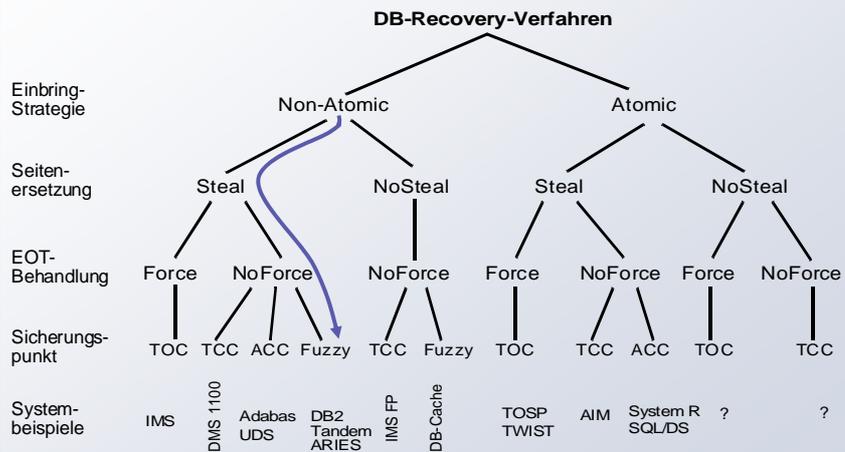
TAIS – WS0506 – Kapitel 5: Logging und Recovery

57

## Sicherungspunkte (13)

NonAtomic, NoSteal, Force, TOC

### ■ Kombinierbarkeit der Verfahren



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

58



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (1)

- Test zur Fehlerbehandlung

	Datenseite bereits in DB eingebracht	Log-Satz bereits in Log-Datei geschrieben	TA nicht beendet, ggf. Zurücksetzung	TA abgeschlossen, ggf. Wiederholung
1	nein	nein	a	e
2	nein	ja	a	c
3	ja	nein	d	e
4	ja	ja	b	a

- Mögliche Antworten:
  - a. tu' nichts
  - b. benutze die UNDO-Information und setze zurück
  - c. benutze die REDO-Information und wiederhole
  - d. WAL-Prinzip verhindert diese Situation
  - e. 2PC (Commit-Regel) verhindert diese Situation

TAIS – WS0506 – Kapitel 5: Logging und Recovery

59



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (2)

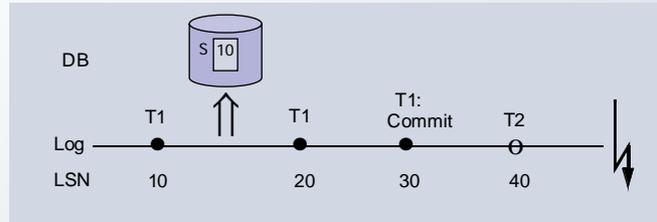
- Nutzung von LSNs
  - Herausforderung
    - beim Restart zu entscheiden, ob für die Seite Recovery-Maßnahmen anzuwenden sind oder nicht (ob der alte oder bereits der geänderte Zustand auf dem Externspeicher vorliegt)
    - dazu wird auf jeder Seite B (im Seitenkopf) die LSN des jüngsten diese Seite betreffenden Log-Eintrags L gespeichert (PageLSN (B) := LSN (L))
  - Entscheidungsprozedur
    - Restart hat eine Redo- und eine Undo-Phase
    - Redo ist nur erforderlich, wenn  
Seiten-LSN < LSN des Redo-Log-Satzes
    - Undo ist nur erforderlich, wenn  
Seiten-LSN ≥ LSN des Undo-Log-Satzes

TAIS – WS0506 – Kapitel 5: Logging und Recovery

60

## Recovery (3)

- Nutzung von LSNs (Forts.)
  - Vereinfachte Anwendung: Seitensperren

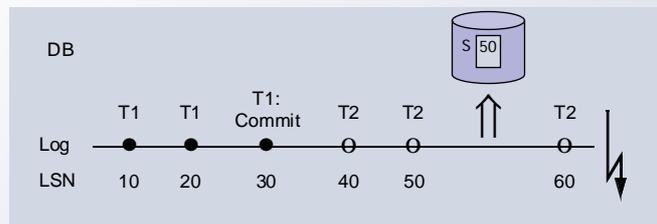


- Redo von T1:  $S(10) = T1(10)$ : -  
 $S(10) < T1(20)$ : Redo, S(20)
- Undo von T2:  $S(20) < T2(40)$ : -
- Seiten-LSN wird demnach bei Redo aktualisiert (wächst monoton)



## Recovery (4)

- Nutzung von LSNs (Forts.)
  - Vereinfachte Anwendung: Seitensperren (Forts.)



- Redo von T1:  $S(50) > T1(10)$ : -  
 $S(50) > T1(20)$ : -
- Undo von T2:  $S(50) < T2(60)$ : -  
 $S(50) \geq T2(50)$ : Undo  
 $S(50) \geq T2(40)$ : Undo



## Recovery (4)

- Nutzung von LSNs (Forts.)
  - Undo erfolgt in LIFO-Reihenfolge
    - ‚Undos‘ müssen speziell behandelt werden, so dass wiederholte Ausführung zum gleichen Ergebnis führt (Idempotenz)
    - Zustandslogging und LIFO-Reihenfolge gewährleisten Idempotenz

© N. Ritter TAIS – WS0506 – Kapitel 5: Logging und Recovery 63

## Recovery (5)

- Crash-Recovery
  - Ziel
    - Herstellung des jüngsten transaktionskonsistenten DB-Zustandes aus permanenter DB und temporärer Log-Datei
  - bei Update-in-Place (*non-atomic*)
    - Zustand der permanenten DB nach Crash unvorhersehbar („chaotisch“)
    - daher nur physische Logging-Verfahren anwendbar
    - ein Block der permanenten DB ist entweder
      - aktuell
      - oder veraltet (*no-force*) → Redo
      - oder ‚schmutzig‘ (*steal*) → Undo

© N. Ritter TAIS – WS0506 – Kapitel 5: Logging und Recovery 64



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (6)

- Crash-Recovery (Forts.)
  - bei *atomic*
    - permanente DB entspricht Zustand des letzten erfolgreichen Einbringens (Sicherungspunkt)
    - zumindest aktionskonsistent → DML-Befehle ausführbar (logisches Logging)
    - *force*: kein Redo
    - *noforce*:
      - transaktionskonsistentes Einbringen → Redo, jedoch kein Undo
      - aktionskonsistentes Einbringen → Undo + Redo

TAIS – WS0506 – Kapitel 5: Logging und Recovery

65



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (7)

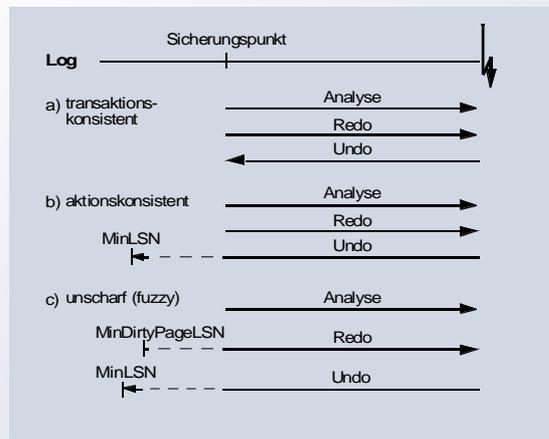
- Allgemeine Restart-Prozedur
  - temporäre Log-Datei wird 3-mal gelesen
    1. Analyse-Phase
      - vom letzten Sicherungspunkt bis zum Log-Ende
      - Bestimmung von Gewinner- und Verlierer-TA sowie der Seiten, die von ihnen geändert wurden
    2. Redo-Phase
      - Vorwärtslesen des Log: Startpunkt abhängig vom Sicherungspunkt-Typ
      - selektives Redo bei Seitensperren (*redo winners*) oder vollständiges Redo (*repeating history*) möglich
    3. Undo-Phase
      - Rücksetzen der Verlierer-TA durch Rückwärtslesen des Logs bis zum BOT-Satz der ältesten Verlierer-TA

TAIS – WS0506 – Kapitel 5: Logging und Recovery

66

## Recovery (8)

### ■ Allgemeine Restart-Prozedur (Forts.)



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

67

## Recovery (9)

### ■ Restart-Prozedur bei Update-in-Place

- Attribute: non-atomic, steal, noforce, fuzzy checkpoints
- Ablauf
  1. Analyse-Phase
    - vom letzten Sicherungspunkt bis zum Log-Ende
  2. Redo-Phase
    - Startpunkt abhängig vom Sicherungspunkt-Typ: hier **MinDirtyPageLSN**
    - selektives Redo: nur Wiederholung der Änderungen der Gewinner-TA
  3. Undo-Phase
    - Rücksetzen der Verlierer-TA bis **MinLSN**



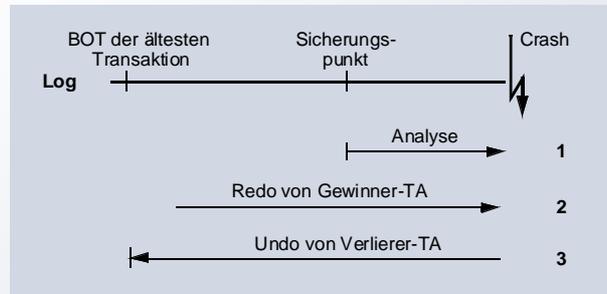
© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

68

## Recovery (10)

### Restart-Prozedur bei Update-in-Place (Forts.)



#### Aufwandsaspekte

- für Schritt 2 und 3 sind betroffene DB-Seiten einzulesen
- LSN der Seiten zeigen, ob Log-Informationen anzuwenden sind
- am Ende sind alle geänderten Seiten wieder auszuschreiben, bzw. es wird ein Sicherungspunkt erzeugt



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

69

## Recovery (11)

### Redo-Recovery

#### Notwendigkeit

- bei physischem und physiologischem Logging
  - Redo-Aktion für Log-Satz L wird über PageLSN der betroffenen Seite B angezeigt  
if (B nicht gepuffert) then (lies B in den Hauptspeicher ein);  
if LSN (L) > PageLSN (B) then do;  
    Redo (Änderung aus L);  
    PageLSN (B) := LSN (L);  
end;
- wiederholte Anwendung des Log-Satzes (z.B. nach mehrfachen Fehlern) erhält Korrektheit (Redo-Idempotenz)
- Recovery bei Crashes während des Restart?



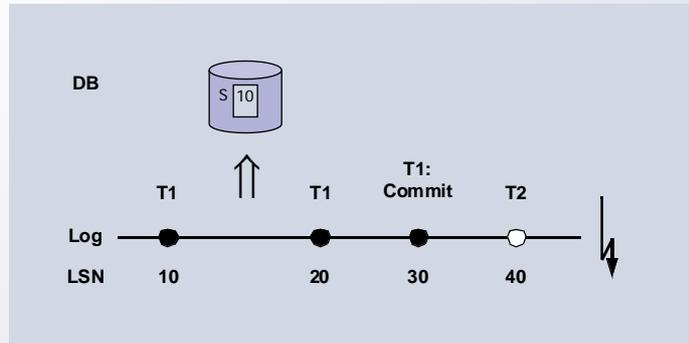
© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

70

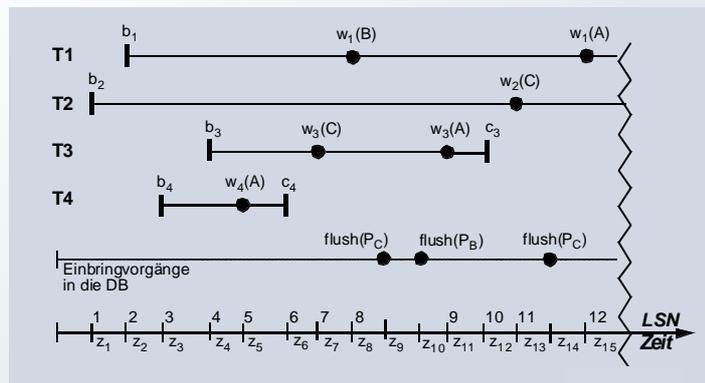
## Recovery (12)

### Redo-Recovery (Forts.)



## Recovery (13)

### Restart-Beispiel



## Recovery (14)

### Restart-Beispiel (Forts.)

Zeit	Aktion	Änderung im DB-Puffer (Seite, LSN)	Änderung in der DB (Seite, LSN)	Log-Puffer: (LSN, TAID, Log-Info, PrevLSN)	Log-Datei: zugefügte Einträge (LSNs)
z <sub>1</sub>	b <sub>2</sub>			1, T <sub>2</sub> , BOT, 0	
z <sub>2</sub>	b <sub>1</sub>			2, T <sub>1</sub> , BOT, 0	
z <sub>3</sub>	b <sub>4</sub>			3, T <sub>4</sub> , BOT, 0	
z <sub>4</sub>	b <sub>3</sub>			4, T <sub>3</sub> , BOT, 0	
z <sub>5</sub>	w <sub>4</sub> (A)	P <sub>A</sub> , 5		5, T <sub>4</sub> , U/R(A), 3	
z <sub>6</sub>	c <sub>4</sub>			6, T <sub>4</sub> , EOT, 5	1, 2, 3, 4, 5, 6
z <sub>7</sub>	w <sub>3</sub> (C)	P <sub>C</sub> , 7		7, T <sub>3</sub> , U/R(C), 4	
z <sub>8</sub>	w <sub>1</sub> (B)	P <sub>B</sub> , 8		8, T <sub>1</sub> , U/R(B), 2	
z <sub>9</sub>	flush(P <sub>C</sub> )		P <sub>C</sub> , 7		7, 8
z <sub>10</sub>	flush(P <sub>B</sub> )		P <sub>B</sub> , 8		
z <sub>11</sub>	w <sub>3</sub> (A)	P <sub>A</sub> , 9		9, T <sub>3</sub> , U/R(A), 7	
z <sub>12</sub>	c <sub>3</sub>			10, T <sub>3</sub> , EOT, 9	9, 10
z <sub>13</sub>	w <sub>2</sub> (C)	P <sub>C</sub> , 11		11, T <sub>2</sub> , U/R(C), 1	
z <sub>14</sub>	flush(P <sub>C</sub> )		P <sub>C</sub> , 11		11
z <sub>15</sub>	w <sub>1</sub> (A)	P <sub>A</sub> , 12		12, T <sub>1</sub> , U/R(A), 8	



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

73

## Recovery (15)

### Restart-Beispiel (Forts.)

- Annahme
  - zu Beginn seien alle Seiten-LSNs 0
- Analyse-Phase:
  - Gewinner-TA: T<sub>3</sub>, T<sub>4</sub>
  - Verlierer-TA: T<sub>1</sub>, T<sub>2</sub>
  - relevante Seiten: P<sub>A</sub>, P<sub>B</sub>, P<sub>C</sub>
- Anmerkung
  - im Beispiel ändert nie mehr als eine TA gleichzeitig in einer Seite, was einem Einsatz von Seitensperren entspricht; deshalb ist Selektives Redo, also nur das Redo der Gewinner-TA, ausreichend



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

74

## Recovery (16)

### Restart-Beispiel (Forts.)

- Redo-Phase:
  - Log-Sätze für  $T_3$  und  $T_4$  vorwärts prüfen

TA	Seite	Seiten-LSN	Log-Satz-LSN	Aktion
$T_4$	$P_A$	0 → 5	5	REDO
$T_3$	$P_C$	11	7	Kein REDO
$T_3$	$P_A$	5 → 9	9	REDO

- Redo nur, wenn Seiten-LSN < Log-Satz-LSN
- Seiten-LSNs wachsen monoton



## Recovery (17)

### Restart-Beispiel (Forts.)

- Undo-Phase:
  - Log-Sätze für  $T_1$  und  $T_2$  rückwärts prüfen

TA	Seite	Seiten-LSN	Log-Satz-LSN	Aktion
$T_1$	$P_A$	9	12	Kein Undo, ohnehin nicht in Log-Datei
$T_2$	$P_C$	11	11	Undo
$T_1$	$P_B$	8	8	Undo

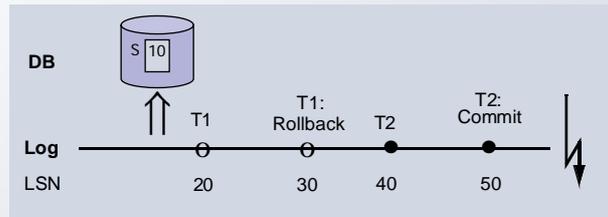
- Undo nur, wenn Seiten-LSN  $\geq$  Log-Satz-LSN
- wegen der Seitensperren gibt es auf einer Seite keine Interferenz zwischen Redo- und Undo-Aktionen; Zustands-Logging sichert Undo-Idempotenz!



## Recovery (18)

### ■ Probleme bei LSN-Verwendung für Undo

- Problem 1: Rücksetzungen von TA
  - bisherige LSN-Verwendung führt zu Problemen in der Undo-Phase bei vorherigem Rollback



- Redo von T2:  $S(10) < T2(40)$  : Redo,  $S(40)$
- Undo von T1:  $S(40) > T1(20)$  : Undo, Fehler



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

77

## Recovery (19)

### ■ Probleme bei LSN-Verwendung für Undo (Forts.)

- Problem 1: Rücksetzungen von TA (Forts.)
  - Bemerkung
    - es wird Änderung 20 zurückgesetzt, obwohl sie gar nicht in der Seite S vorliegt
    - Zuweisung von LSN = 20 zu S verletzt Monotonieforderung für Seiten-LSNs
    - was passiert bei Crash nach Zuweisung?



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

78

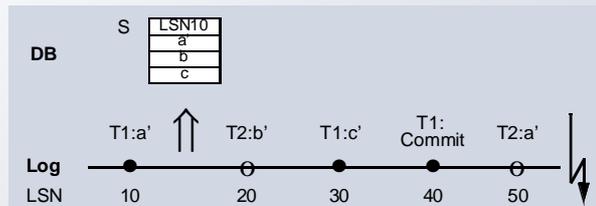
## Recovery (20)

### Probleme bei LSN-Verwendung für Undo (Forts.)

#### Problem 2: Satzsperrn

- Ausgangszustand der Seite S
- T1 und T2 ändern gleichzeitig in Seite

S	LSN 5
	a
	b
	c



- Redo von T1:
    - $S(10) \geq T1(10)$ : kein Redo
    - $S(10) < T1(30)$ : Redo, S(30)
  - Undo von T2 (LIFO):
    - $S(30) < T2(50)$ : kein Undo
    - $S(30) > T2(20)$ : Undo, Fehler!
- Allgemeinere Behandlung des Undo erforderlich!



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

79

## Recovery (21)

### Fehlertoleranz des Restart

- Forderung: Idempotenz des Restart

$$\text{Undo}(\text{Undo}(\dots(\text{Undo}(A))\dots)) = \text{Undo}(A)$$

$$\text{Redo}(\text{Redo}(\dots(\text{Redo}(A))\dots)) = \text{Redo}(A)$$

- Idempotenz der Redo-Phase wird dadurch erreicht, dass LSN des Log-Satzes, für den ein Redo tatsächlich ausgeführt wird, in die Seite eingetragen wird; Redo-Operationen erfordern keine zusätzliche Protokollierung
- Seiten-LSNs müssen monoton wachsen; deshalb kann in der Undo-Phase nicht entsprechend verfahren werden



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

80

## Recovery (22)

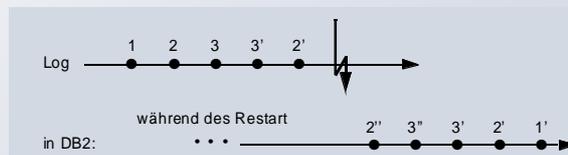
- Fehlertoleranz des Restart (Forts.)

- Forderung: Idempotenz des Restart (Forts.)

- Gewährleistung der Idempotenz der Undo-Phase erfordert ein neues Konzept: CLR = Compensation Log Record

- Änderungen der DB sind durch Log-Einträge abzusichern - und zwar im Normalbetrieb und beim Restart!

- was passiert im Fall eines Crash beim Undo?  
Aktionen 1-3 sollen zurückgesetzt werden:  
I' ist CLR für I und I'' ist CLR für I'



- Problem von kompensierenden Kompensationen!
      - Crash bei Restart!?



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

81

## Recovery (23)

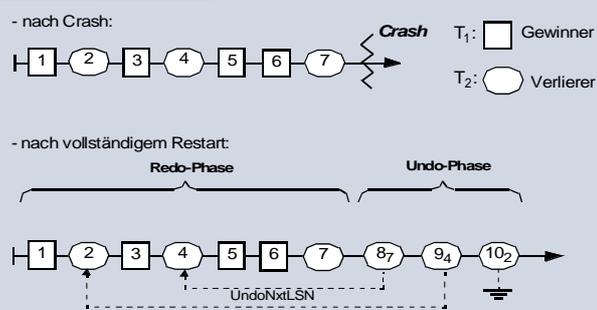
- Compensation Log Records (CLR)

- Optimierte Lösung

- Einsatz von CLRs bei allen Undo-Operationen: Rollback und Undo-Phase

- in der Redo-Phase: vollständiges Redo von Gewinnern und Verlierern („repeating history“)

- schematische Darstellung der Log-Datei



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

82

## Recovery (24)

- Compensation Log Records (Forts.)
  - Optimierte Lösung (Forts.)
    - schematische Darstellung der Log-Datei (zu Abbildung auf vorangegangener Folie)
      - die Redo-Information eines CLR entspricht der während der Undo-Phase ausgeführten Undo-Operation
      - CLR-Sätze werden bei erneutem Restart benötigt (nach Crash beim Restart); ihre Redo-Information wird während der Redo-Phase angewendet; dabei werden Seiten-LSNs geschrieben → die Redo-Phase ist idempotent!
      - CLR-Sätze benötigen keine Undo-Information, da sie während nachfolgender Undo-Phasen übersprungen werden (UndoNxtLSN)



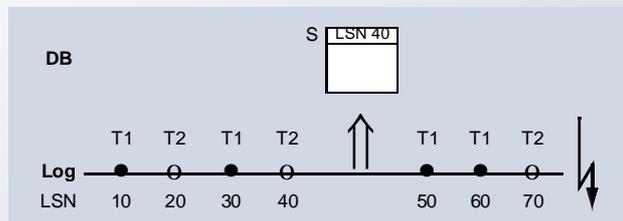
© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

83

## Recovery (25)

- Compensation Log Records (Forts.)
  - Detaillierung des Beispiels
    - alle Änderungen betreffen Seite S
    - Zustand nach Crash 1



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

84

## Recovery (26)

### ■ Compensation Log Records (Forts.)

- Fortsetzung des Beispiels (Forts.)

- Zustand nach Crash 1 (Forts.)

- repeating history:

S(40) > T1(10): -

...

S(40) ≥ T2(40): -

S(40) < T1(50): Redo, S(50)

S(50) < T1(60): Redo, S(60)

S(60) < T2(70): Redo, S(70)

- Undo von T2:

CLR(80): Kompensieren von T2(70), S(80)

Schreiben von S in die DB (Flush S)

CLR(90): Kompensieren von T2(40), S(90)

Crash



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

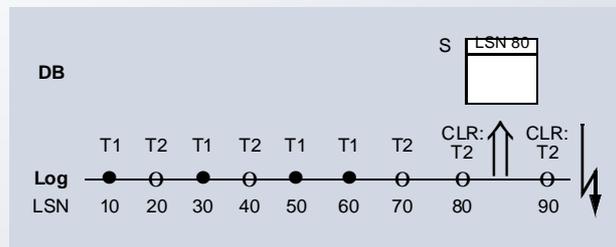
85

## Recovery (27)

### ■ Compensation Log Records (Forts.)

- Fortsetzung des Beispiels (Forts.)

- Zustand nach Crash 2



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

86



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (28)

---

- Compensation Log Records (Forts.)
  - Fortsetzung des Beispiels (Forts.)
    - Zustand nach Crash 2 (Forts.)
      - Repeating History:
        - S(80) > T1(10): -
        - ...
        - S(80) > T2(70): -
        - CLR(80): -
        - CRL(90): Kompensieren von T2(40), S(90)
      - Undo von T2:
        - CLR(100): Kompensieren von T2(20), S(100)
        - Ende

TAIS – WS0506 – Kapitel 5: Logging und Recovery

87



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (29)

---

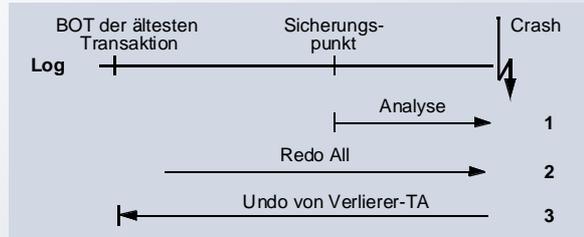
- Restart-Prozedur bei Update-in-Place
  - Attribute: non-atomic, steal, noforce, fuzzy checkpoints
    1. Analyse-Phase
      - vom letzten Sicherungspunkt bis zum Log-Ende
    2. Redo-Phase
      - Startpunkt abhängig vom Sicherungspunkt-Typ: hier MinDirtyPageLSN
      - Vollständiges Redo oder Repeating History: Wiederholung aller Änderungen (auch von Verlierer-TA), falls erforderlich
    3. Undo-Phase
      - Rücksetzen der Verlierer-TA bis MinLSN

TAIS – WS0506 – Kapitel 5: Logging und Recovery

88

## Recovery (30)

- Restart-Prozedur bei Update-in-Place (Forts.)
  - Attribute: non-atomic, steal, noforce, fuzzy checkpoints (Forts.)

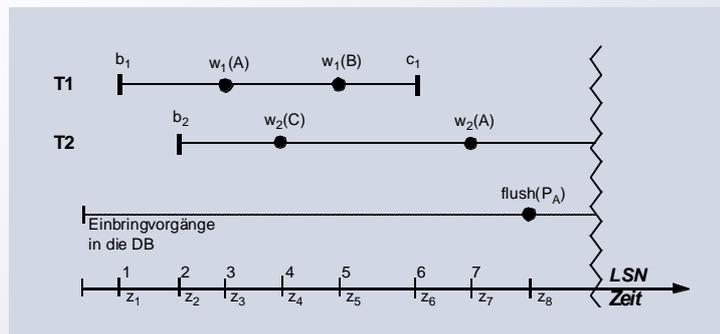


- Umsetzung durch ARIES
  - Algorithm for Recovery and Isolation Exploiting Semantics
  - entwickelt von C. Mohan et al. (IBM Almaden Research)
  - realisiert in einer Reihe von kommerziellen DBS

Mohan, C. et al.: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging, in ACM TODS 17:1, 1992, 94-162

## Recovery (31)

- Restart-Beispiel 2



## Recovery (32)

### Restart-Beispiel 2 (Forts.)

Zeit	Aktion	Änderung im DB-Puffer (Seite, LSN)	Änderung in der DB (Seite, LSN)	Log-Puffer: (LSN, TAID, Log-Info, PrevLSN)	Log-Datei: zugefügte Einträge (LSNs)
z <sub>1</sub>	b <sub>1</sub>			1, T <sub>1</sub> , BOT, 0	
z <sub>2</sub>	b <sub>2</sub>			2, T <sub>2</sub> , BOT, 0	
z <sub>3</sub>	w <sub>1</sub> (A)	P <sub>A</sub> , 3		3, T <sub>1</sub> , U/R(A), 1	
z <sub>4</sub>	w <sub>2</sub> (C)	P <sub>C</sub> , 4		4, T <sub>2</sub> , U/R(C), 2	
z <sub>5</sub>	w <sub>1</sub> (B)	P <sub>B</sub> , 5		5, T <sub>1</sub> , U/R(B), 3	
z <sub>6</sub>	c <sub>1</sub>			6, T <sub>1</sub> , EOT, 5	1, 2, 3, 4, 5, 6
z <sub>7</sub>	w <sub>2</sub> (A)	P <sub>A</sub> , 7		7, T <sub>2</sub> , U/R(A), 4	
z <sub>8</sub>	flush(P <sub>A</sub> )		P <sub>A</sub> , 7		7



© N. Ritter

## Recovery (33)

### Restart-Beispiel 2 (Forts.)

- Annahme
  - zu Beginn seien alle Seiten-LSNs 0
- Analyse-Phase
  - Gewinner-TA: T<sub>1</sub>
  - Verlierer-TA: T<sub>2</sub>
  - relevante Seiten: P<sub>A</sub>, P<sub>B</sub>, P<sub>C</sub>
- Anmerkung
  - im Restart-Beispiel 2 wird vollständiges Redo durchgeführt; zur Gewährleistung der Idempotenz der Undo-Operationen wird für jede ausgeführte Undo-Operation ein CLR mit folgender Struktur angelegt
    - [LSN, TAID, PageID, Redo, PrevLSN, UndoNextLSN]



© N. Ritter

## Recovery (34)

### Restart-Beispiel 2 (Forts.)

- Redo-Phase
  - Log-Sätze aller TA ( $T_1, T_2$ ) vorwärts prüfen

TA	Seite	Seiten-LSN	Log-Satz-LSN	Aktion
$T_1$	$P_A$	7	3	Kein REDO
$T_2$	$P_C$	$0 \rightarrow 4$	4	REDO
$T_1$	$P_B$	$0 \rightarrow 5$	5	REDO
$T_2$	$P_A$	7	7	Kein REDO

- Redo nur, wenn Seiten-LSN < Log-Satz-LSN



## Recovery (35)

### Restart-Beispiel 2 (Forts.)

- Undo-Phase
  - Log-Sätze der Verlierer-TA  $T_2$  rückwärts unabhängig von Seiten-LSN prüfen
    - für jeden Log-Satz wird die zugehörige Undo-Operation durchgeführt und mit einem CLR in der Log-Datei vermerkt

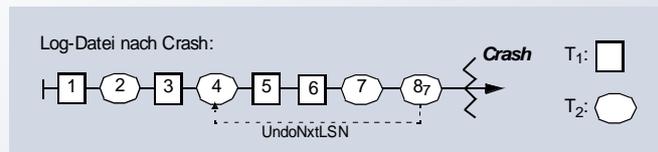
TA	Log-Satz-LSN	Aktion
$T_2$	7	UNDO und lege CLR[8, $T_2$ , $P_A$ , U(A), 7, 4] an
$T_2$	4	UNDO und lege CLR[9, $T_2$ , $P_C$ , U(C), 8, 2] an
$T_2$	2	UNDO und lege CLR[10, $T_2$ , _ , _ , 9, 0] an



## Recovery (36)

### Restart-Beispiel 2 (Forts.)

- Annahme
  - Crash während des Restart



- Analyse-Phase
  - dto
- Redo-Phase
  - Log-Sätze aller TA ( $T_1$ ,  $T_2$ ) inkl. CLR vorwärts prüfen
  - für jedes CLR wird jeweils Redo ausgeführt



## Recovery (37)

### Restart-Beispiel 2 (Forts.)

- Redo-Phase
  - Log-Sätze aller TA ( $T_1$ ,  $T_2$ ) inkl. CLR vorwärts prüfen
  - für jedes CLR wird jeweils Redo ausgeführt

TA	Seite	Seiten-LSN	Log-Satz-LSN	Aktion
$T_1$	$P_A$	7	3	Kein REDO
$T_2$	$P_C$	4	4	Kein REDO
$T_1$	$P_B$	5	5	Kein REDO
$T_2$	$P_A$	7	7	Kein REDO
$T_2$	$P_A$	7 → 8	8	REDO: U(A)



## Recovery (38)

- Restart-Beispiel 2 (Forts.)
  - Undo-Phase
    - Log-Sätze der Verlierer-TA  $T_2$  (inkl. CLR) rückwärts unabhängig von Seiten-LSN prüfen
    - für jeden Log-Satz wird die zugehörige Undo-Operation durchgeführt und mit einem CLR in der Log-Datei vermerkt

TA	Log-Satz-LSN	Aktion
$T_2$	8	UndoNxtLSN = 4, dann weiter mit 4. Log-Satz (7. Log-Satz wird übersprungen, da er bereits mit dem 8. kompensiert wurde)
$T_2$	4	UNDO und lege CLR[9, $T_2$ , $P_C$ , U(C), 8, 2] an
$T_2$	2	UNDO und lege CLR[10, $T_2$ , -, -, 9, 0] an

© N. Ritter
TAIS – WS0506 – Kapitel 5: Logging und Recovery
97

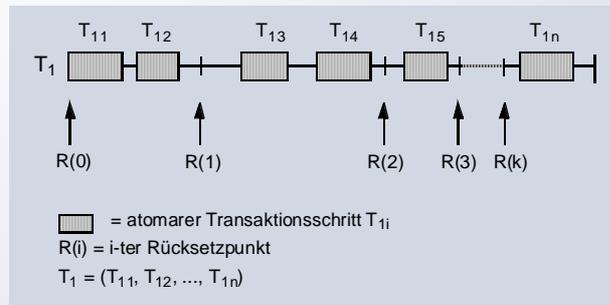
## Recovery (39)

- Zurücksetzen von Transaktionen
  - Transaktions-Recovery
    - Zurücksetzen einer TA im laufenden DB-Betrieb
    - Nutzung der PrevLSN-Kette im temporären Log
    - Schreiben von optimierten CLR, um mehrfaches Rücksetzen bei Restart zu vermeiden
  - Erweiterung zum partiellen Zurücksetzen
    - Voraussetzung
      - transaktionsinterne Rücksetzpunkte (*Savepoints*)
    - Zusätzliche Operationen: Save R(i), Restore R(j)
    - Protokollierung aller Änderungen, Sperren, Cursor-Positionen usw.
    - Undo-Operation bis R(j) in LIFO-Reihenfolge

© N. Ritter
TAIS – WS0506 – Kapitel 5: Logging und Recovery
98

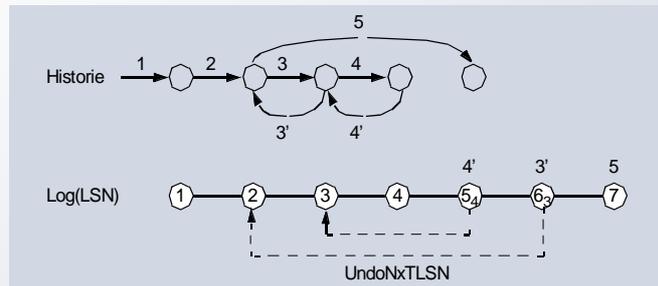
## Recovery (40)

- Zurücksetzen von Transaktionen (Forts.)
  - Erweiterung zum partiellen Zurücksetzen (Forts.)



## Recovery (41)

- Zurücksetzen von Transaktionen (Forts.)
  - Partielles Zurücksetzen einer TA



- Rücksetzpunkte müssen vom DBS sowie vom Laufzeitsystem der Programmiersprache unterstützt werden
  - derzeitige Implementierungen bieten keine Unterstützung von persistenten *Savepoints*!
  - nach Systemfehler wird TA vollständig zurückgesetzt



## Recovery (42)

- Platten-Recovery
  - Spiegelplatten
    - schnellste und einfachste Lösung
    - hohe Speicherkosten
    - Doppelfehler nicht auszuschließen
  - Alternative
    - Archivkopie + Archiv-Log
    - sind längerfristig verfügbar zu halten (auf Band)
    - Problem von Alterungsfehlern
      - Führen von Generationen der Archivkopie
      - Duplex-Logging für Archiv-Log

```

graph LR
    A[Archivkopie] --> B[Archiv-Log]
    B --> C[Temporärer Log]
    C --> D[(DB)]
  
```

© N. Ritter TAIS – WS0506 – Kapitel 5: Logging und Recovery 101

## Recovery (43)

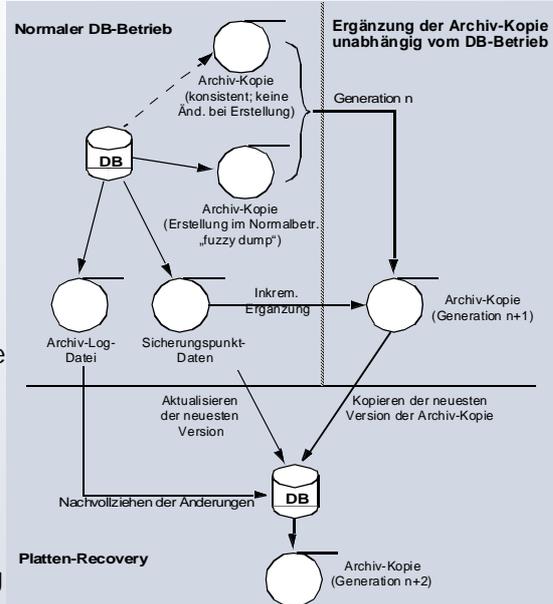
- Platten-Recovery (Forts.)
  - Ableitung von Archivdaten
    - Sammlung sehr großer Datenvolumina als nachgelagerter Prozess
    - Archiv-Log kann offline aus temporärer Log-Datei abgeleitet werden
    - Erstellung von Archivkopien und Archiv-Log erfolgt segmentorientiert

© N. Ritter TAIS – WS0506 – Kapitel 5: Logging und Recovery 102

## Recovery (44)

### ■ Platten-Recovery (Forts.)

- Szenario/  
Komponenten:
- Optimierung der Erstellung der Archivkopie durch inkrementelle Ergänzung mit Daten von Sicherungspunkten und ggf. Archiv-Log



## Recovery (45)

### ■ Erstellung der Archivkopie

- Anhalten des Änderungsbetriebs zur Erstellung einer DB-Kopie i. allg. nicht tolerierbar
- Alternativen:
  - Inkrementelles Dumping (a)
    - Ableiten neuer Generationen aus 'Urkopie'
    - nur Änderungen seit der letzten Archiv-Kopie protokollieren
    - Offline-Erstellung einer aktuelleren Kopie
  - Online-Erstellung einer Archivkopie (b)
    - parallel zum Änderungsbetrieb



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (46)

---

- Erstellung der Archivkopie (Forts.)
  - Unterschiedliche Konsistenzgrade:
    - fuzzy dump (b1)
      - Kopieren der DB im laufenden Betrieb, kurze Lesesperren
      - bei Plattenfehler Archiv-Log ab Beginn der Dump-Erstellung anzuwenden
    - aktionskonsistente Archivkopie (b2)
      - Voraussetzung bei logischem Operations-Logging
    - transaktionskonsistente Archivkopie (b3)
      - Voraussetzung bei logischem Transaktions-Logging
      - Black-/White-Verfahren
      - Copy-on-Update-Verfahren

TAIS – WS0506 – Kapitel 5: Logging und Recovery

105



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Recovery (47)

---

- Black-/White-Verfahren
  - Ziel
    - Erzeugung transaktionskonsistenter Archiv-Kopien
  - spezieller Dumpprozess zur Erstellung der Archiv-Kopie
  - Kennzeichnung der Seiten
    - Paint-Bit (pro Seite)
      - weiß: Seite wurde noch nicht überprüft
      - schwarz: Seite wurde bereits verarbeitet
    - Modified-Bit (pro Seite)
      - zeigt an, ob eine Änderung seit Erstellung der letzten Archiv-Kopie erfolgte

TAIS – WS0506 – Kapitel 5: Logging und Recovery

C. Pu: On-the-Fly, Incremental, Consistent Reading of Entire Databases, in: Algorithmica, 1986, 271 - 287

106

## Recovery (48)

### ■ Black-/White-Verfahren (Forts.)

#### • Kennzeichnung der Seiten (Forts.)

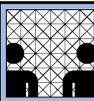
- Dumpprozess färbt alle weißen Seiten schwarz und schreibt geänderte Seiten in Archiv-Kopie:

```

WHILE there are white pages DO;
  lock any white page;
  IF page is modified THEN DO;
    write page to archive copy;
    clear modified bit;
  END;
  change page color;
  release page lock;
END;
    
```

#### • Rücksetzregel

- Transaktionen, die sowohl weiße als auch schwarze Objekte geändert haben ('graue Transaktionen'), werden zurückgesetzt
- 'Farbtest' am Transaktionsende



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

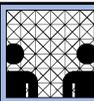
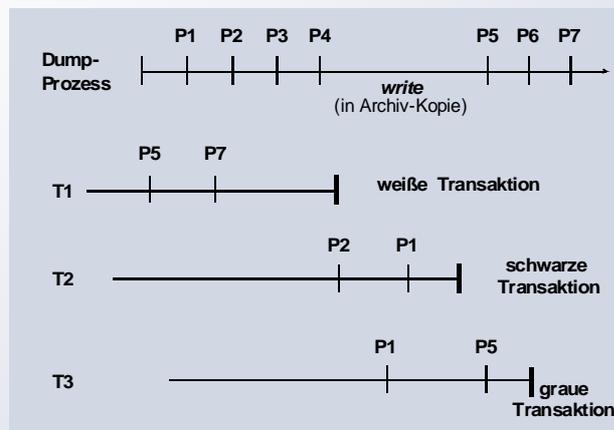
TAIS – WS0506 – Kapitel 5: Logging und Recovery

107

## Recovery (49)

### ■ Black-/White-Verfahren (Forts.)

#### • Beispiel



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

108



- Begriffe
- Logging
- Komponenten
- Sicherungspunkte
- Recovery



© N. Ritter

## Recovery (50)

- Black-/White-Verfahren (Forts.)
  - Erweiterungen zur Vermeidung von Rücksetzungen
    - Turn-White-Strategien (*Turn gray transactions white*)
      - für graue Transaktionen werden Änderungen 'schwarzer' Objekte nachträglich in Archiv-Kopie geschrieben
      - Problem: transitive Abhängigkeiten
      - Alternative: alle Änderungen schwarzer Objekte seit Dump-Beginn werden noch geschrieben (*repaint all*)
      - Problem: Archiv-Kopie-Erstellung kommt u.U. nie zu Ende
    - Turn-Black-Strategien
      - während der Erstellung einer Archiv-Kopie werden keine Zugriffe auf weiße Objekte vorgenommen
      - ggf. zu warten, bis Objekt gefärbt wird

TAIS – WS0506 – Kapitel 5: Logging und Recovery
109



- Begriffe
- Logging
- Komponenten
- Sicherungspunkte
- Recovery



© N. Ritter

## Recovery (51)

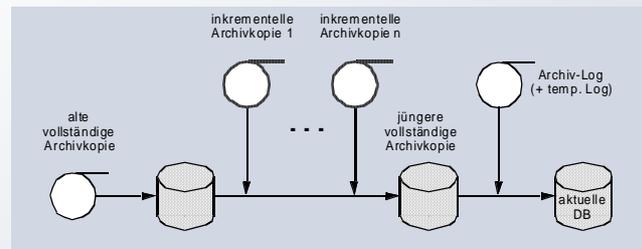
- Black-/White-Verfahren (Forts.)
  - Erweiterungen zur Vermeidung von Rücksetzungen (Forts.)
    - Alternative: *Copy-on-Update* ("save some")
      - während der Erstellung einer Archiv-Kopie wird bei Änderung eines weißen Objektes Kopie mit Before-Image der Seite angelegt
      - Dump-Prozess greift auf Before-Images zu
      - Archiv-Kopie entspricht DB-Schnappschuss bei Dump-Beginn
      - wird in einigen DBS eingesetzt (DEC RDB)

TAIS – WS0506 – Kapitel 5: Logging und Recovery
110

## Recovery (52)

### ■ Inkrementelles Dumping

- nur DB-Seiten, die seit der letzten Archivkopie-Erstellung geändert wurden, werden archiviert



- Erkennung geänderter Seiten
  - Archivierungs-Bit pro Seite → sehr hoher E/A-Aufwand
  - besser: Verwendung separater Datenstrukturen (Bitlisten)
- Setzen eines Änderungsbits falls
  - (PageLSN der ungeänderten Seite) < (LSN zu Beginn des letzten Dumps)

## Zusammenfassung (1)

### ■ Fehlerarten

- Transaktions-, System-, Gerätefehler und Katastrophen

### ■ Spektrum von Logging- und Recovery-Verfahren

- Logging kann auf verschiedenen Systemebenen angesiedelt werden
- erfordert ebenenspezifische Konsistenz im Fehlerfall
- Eintrags-Logging ist Seiten-Logging überlegen
  - in vielen DBS findet sich das physiologische Logging (flexiblere Recovery in einer DB-Seite, geringerer Platzbedarf, weniger E/As, Gruppen-Commit)



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Zusammenfassung (2)

- Zusammenspiel mit anderen Systemkomponenten
  - Synchronisationsgranulat muss größer oder gleich dem Log-Granulat sein
  - Atomic-Verfahren
    - erhalten den DB-Zustand des letzten Sicherungspunktes
    - gewährleisten demnach die gewählte Aktionskonsistenz auch bei der Recovery von einem Crash und
    - erlauben folglich logisches Logging
  - Update-in-Place-Verfahren
    - sind i. allg. Atomic-Strategien vorzuziehen, weil sie im Normalbetrieb wesentlich billiger sind und
    - nur eine geringe Crash-Wahrscheinlichkeit zu unterstellen ist
    - erfordern jedoch physisches Logging

TAIS – WS0506 – Kapitel 5: Logging und Recovery

113



- Begriffe
- Logging
- Komponenten
- Sicherungs-  
punkte
- Recovery



© N. Ritter

## Zusammenfassung (3)

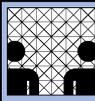
- Grundprinzipien bei Update-in-Place
  - WAL-Prinzip: Write Ahead Log für Undo-Info
  - Redo-Info ist spätestens bei Commit zu schreiben
- Grundprinzipien bei Atomic
  - WAL-Prinzip bei verzögertem Einbringen:
    - TA-bezogene Undo-Info ist vor Sicherungspunkt zu schreiben
  - Redo-Info ist spätestens bei Commit auf die Log-Datei zu schreiben
- NoForce-Strategien
  - sind Force-Verfahren vorzuziehen
  - erfordern den Einsatz von Sicherungspunkt-Maßnahmen zur Begrenzung des Redo-Aufwandes
    - Fuzzy Checkpoints erzeugen den geringsten Overhead im Normalbetrieb

TAIS – WS0506 – Kapitel 5: Logging und Recovery

114

## Zusammenfassung (4)

- Steal-Methoden
  - verlangen die Einhaltung des WAL-Prinzips
  - erfordern Undo-Aktionen nach einem Rechnerausfall
- Idempotenz des Restart
  - Operationen der Redo-Phase, falls erforderlich, erhöhen die Seiten-LSNs; Notwendigkeit der Wiederholung kann jederzeit erkannt werden
  - Idempotenz für Undo- und Rollback-Operationen durch Einführung von CLR; nach Crash in der Undo-Phase werden Undo-Operationen beim nachfolgenden Restart in der Redo-Phase kompensiert (Erhöhung der Seiten-LSNs, beliebig oft unterbrechbar)
- Erstellung von Archiv-Kopien
  - *fuzzy dump* oder *copy on update* am geeignetsten



Begriffe

Logging

Komponenten

Sicherungs-  
punkte

Recovery



© N. Ritter

TAIS – WS0506 – Kapitel 5: Logging und Recovery

115