



Logging und Recovery

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

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




© N. Ritter



Anforderungen / Begriffe


- Voraussetzungen für die Wiederherstellung der Daten
 - quasi-stabiler Speicher
 - fehlerfreier DBVS-Code
 - fehlerfreie Log-Daten
 - Unabhängigkeit der Fehler
- Recovery-Arten
 1. **Transaktions-Recovery**
 - 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




© N. Ritter

N. Ritter, HMS, 13.09.2007

2



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




© N. Ritter

Anforderungen / Begriffe


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

N. Ritter, HMS, 13.09.2007

3



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



© N. Ritter

Anforderungen / Begriffe

- Recovery-Arten (Forts.)
 - 4. **Katastrophen-Recovery**
 - 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)

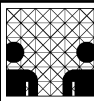
N. Ritter, HMS, 13.09.2007

4

Anforderungen / Begriffe

■ 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



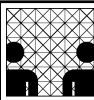
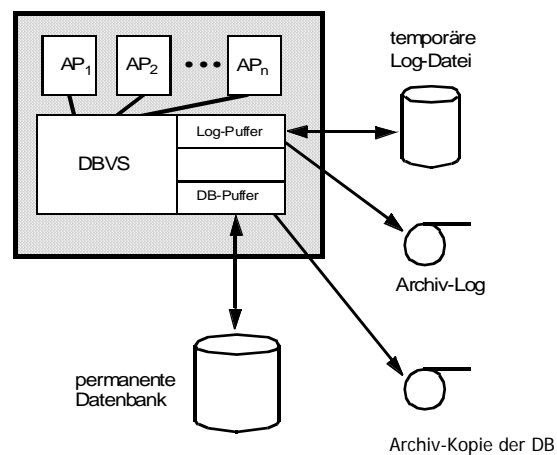
© N. Ritter

N. Ritter, HMS, 13.09.2007

5

Anforderungen / Begriffe

■ DB-Recovery – Systemkomponenten



Begriffe

Logging

Komponenten

Sicherungs-
punkte

Recovery



© N. Ritter

N. Ritter, HMS, 13.09.2007

6

Anforderungen / Begriffe

- 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



© N. Ritter

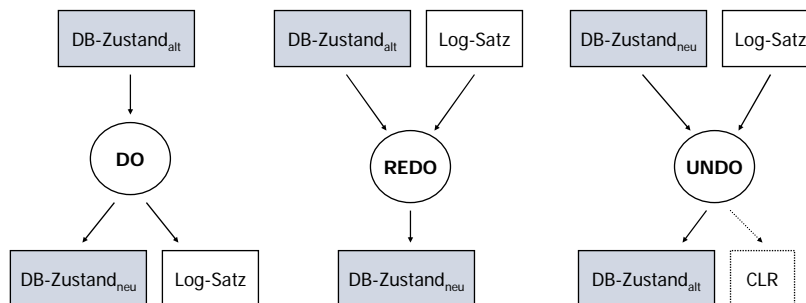
N. Ritter, HMS, 13.09.2007

7

Logging

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

■ Do-Redo-Undo-Prinzip



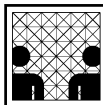
CLR: Compensation Log Record (für Crash während Recovery)



© N. Ritter

N. Ritter, HMS, 13.09.2007

8



Begriffe

Logging

Komponenten

Sicherungs-
punkte

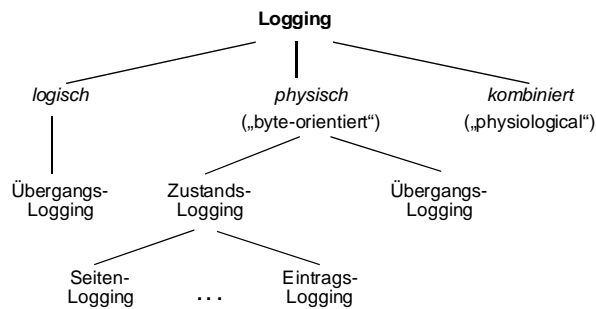
Recovery



© N. Ritter

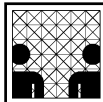
Logging

- 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, HMS, 13.09.2007

9



Begriffe

Logging

Komponenten

Sicherungs-
punkte

Recovery




© N. Ritter

Logging


- 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

N. Ritter, HMS, 13.09.2007

10



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




© N. Ritter

Logging


- 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 Löscho- und Einfügeoperationen

N. Ritter, HMS, 13.09.2007

11



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




© N. Ritter

Logging


- 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

N. Ritter, HMS, 13.09.2007

12



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




© N. Ritter

Logging


- 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

© N. Ritter, HMS, 13.09.2007

13



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



© N. Ritter

Logging

- 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

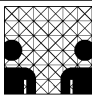
© N. Ritter, HMS, 13.09.2007

14

Logging

■ 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

Sicherungs-
punkte

Recovery

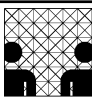


Logging

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

- Beispiel

Schritt	T ₁	T ₂	Log
			[LSN, TAID, 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]



Begriffe


Logging

Komponenten


Sicherungs-
punkte

Recovery





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




© N. Ritter

Logging


- Aufbau der (temporären) Log-Datei (Forts.)
 - Log ist eine sequentielle Datei
 - Schreiben neuer Protokoll Daten 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!

N. Ritter, HMS, 13.09.2007

17



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



© N. Ritter

Abhängigkeiten der Systemkomponenten

- 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, HMS, 13.09.2007

18

Abhängigkeiten der Systemkomponenten

■ Einbringstrategie

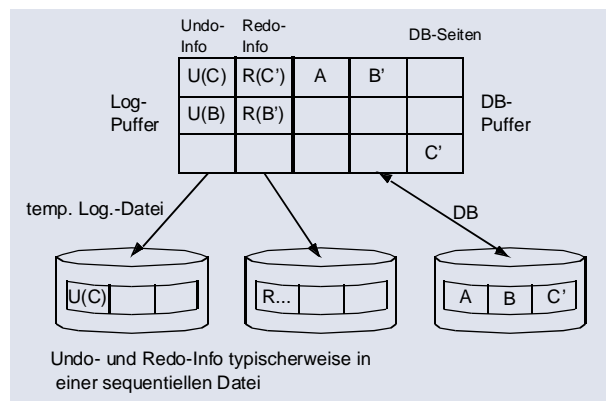
- *non-atomic / direkt / update-in-place*
 - geänderte Seite wird immer in denselben Block auf Platte zurückgeschrieben
 - Schreiben ist gleichzeitig Einbringen in die permanente DB
 - ‚atomares‘ Einbringen mehrerer geänderter Seiten ist nicht möglich (non-atomic)
 - Forderungen
 - WAL-Prinzip: *Write 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)



Abhängigkeiten der Systemkomponenten

■ Einbringstrategie (Forts.)

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



Abhängigkeiten der Systemkomponenten

■ Einbringstrategie (Forts.)

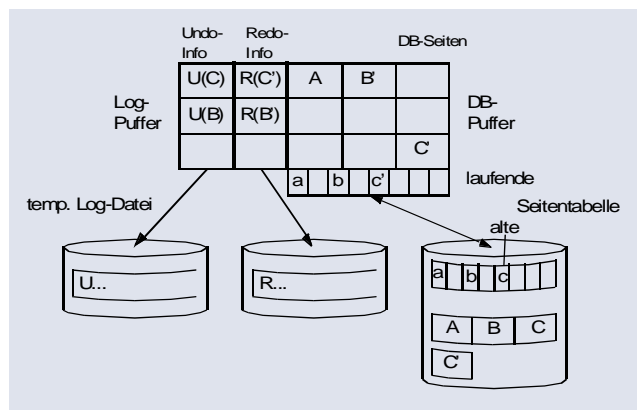
- *atomic / verzögert*
 - z. B. in System R, SQL/DS
 - geänderte Seite wird in separaten Block auf Platte geschrieben, Einbringen in die DB erfolgt später
 - Seitentabelle gibt aktuelle Adresse einer Seite an
 - verzögertes, atomares Einbringen mehrerer Änderungen ist durch Umschalten von Seitentabellen 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


■ Einbringstrategie (Forts.)

- *atomic / verzögert (Forts.)*





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




© N. Ritter

Abhängigkeiten der Systemkomponenten


- 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
 - 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

N. Ritter, HMS, 13.09.2007

23



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



© N. Ritter

Abhängigkeiten der Systemkomponenten

- 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

N. Ritter, HMS, 13.09.2007

24

Abhängigkeiten der Systemkomponenten

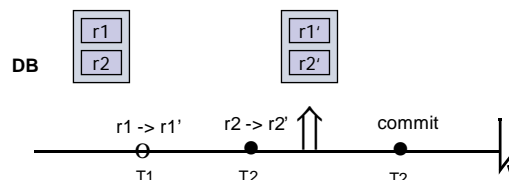
- Auswirkungen von Ersetzungs- und Ausschreibstrategie auf Recovery-Maßnahmen

	steal	nosteal
force	UNDO NO REDO	NO UNDO NO REDO
noforce	UNDO REDO	NO UNDO REDO



Abhängigkeiten der Systemkomponenten

- 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*)

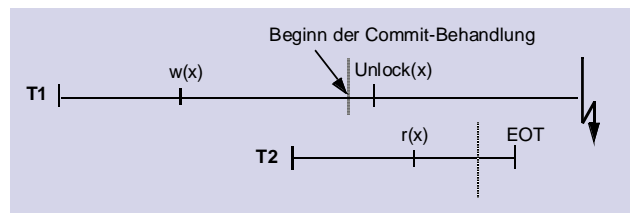


Abhängigkeiten der Systemkomponenten

■ 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

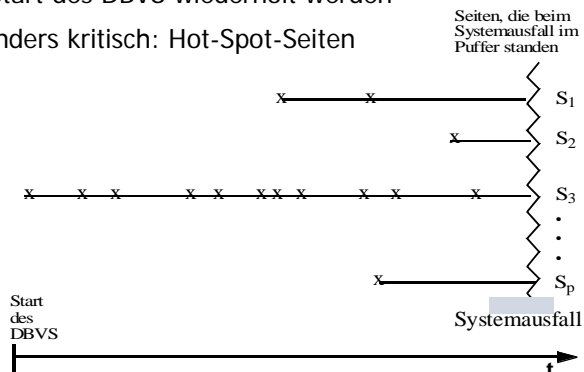
N. Ritter, HMS, 13.09.2007

27

Sicherungspunkte

■ 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

N. Ritter, HMS, 13.09.2007

28



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




© N. Ritter


Sicherungspunkte

- 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

© N. Ritter
N. Ritter, HMS, 13.09.2007
29



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



© N. Ritter

Sicherungspunkte

- 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

© N. Ritter
N. Ritter, HMS, 13.09.2007
30

Sicherungspunkte

■ Arten von Sicherungspunkten (Forts.)

- Indirekte/Unscharfe 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



© N. Ritter

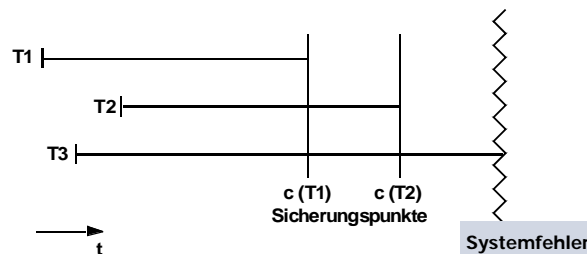
N. Ritter, HMS, 13.09.2007

31

Sicherungspunkte

■ 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*)



© N. Ritter

N. Ritter, HMS, 13.09.2007

32

Sicherungspunkte

■ 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*



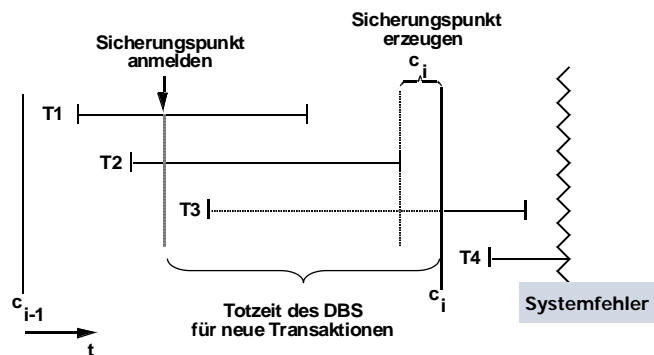
- Begriffe
- Logging
- Komponenten
- Sicherungspunkte
- Recovery



Sicherungspunkte

■ Transaktionskonsistente Sicherungspunkte

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



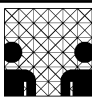
- Begriffe
- Logging
- Komponenten
- Sicherungspunkte
- Recovery



Sicherungspunkte

■ 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



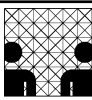
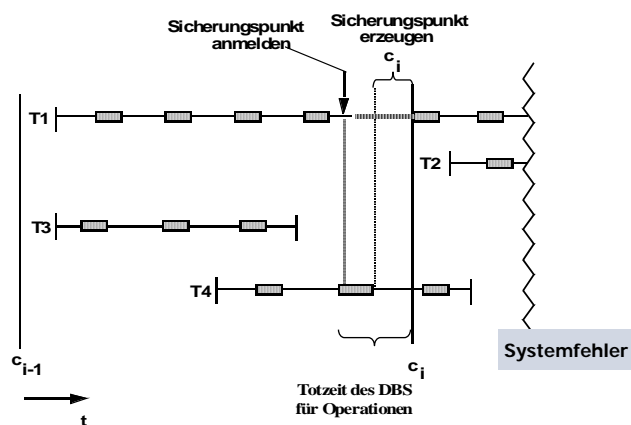
- Begriffe
- Logging
- Komponenten
- Sicherungs-
punkte**
- Recovery



Sicherungspunkte


■ Aktionskonsistente Sicherungspunkte

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




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





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




© N. Ritter

Sicherungspunkte


- 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*

N. Ritter, HMS, 13.09.2007

37



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



© N. Ritter

Sicherungspunkte

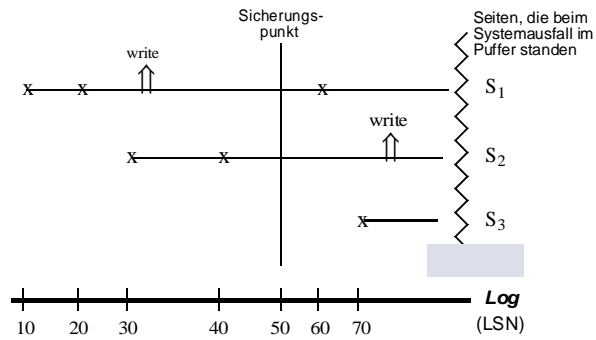
- Unscharfe Sicherungspunkte (Fuzzy Checkpoints)
 - DB auf Platte bleibt ‚fuzzy‘, nicht aktionskonsistent
 - nur bei Update-in-Place (*non-atomic*) relevant
 - Problem
 - Bestimmung der Log-Position, an der Redo-Recovery beginnen muss
 - Pufferverwalter vermerkt sich zu jeder geänderten Seite StartLSN, 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, ...

N. Ritter, HMS, 13.09.2007

38

Sicherungspunkte

■ Unscharfe Sicherungspunkte (Forts)

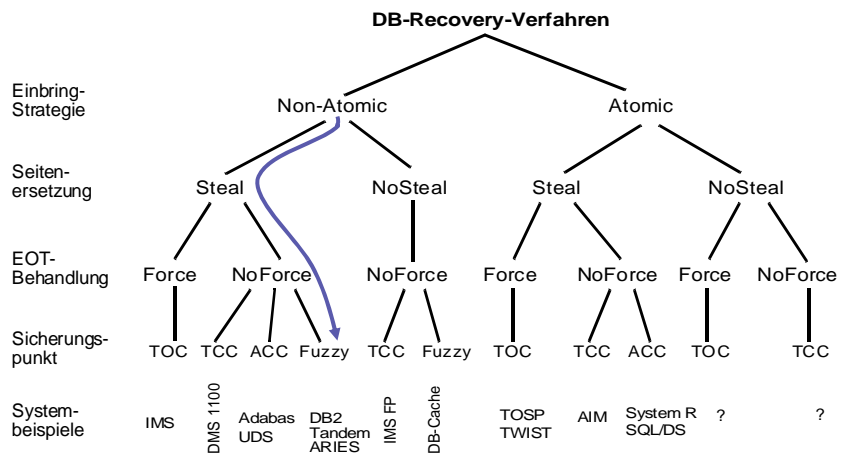



- geänderte Seiten werden asynchron ausgeschrieben
 - ggf. Kopie der Seite anlegen (für Hot-Spot-Seiten)
 - Seite ausschreiben
 - StartLSN anpassen / zurücksetzen




Sicherungspunkte

■ Kombinierbarkeit der Verfahren





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



© N. Ritter


Recovery

- 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 verhindert diese Situation

© N. Ritter, HMS, 13.09.2007 41



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



© N. Ritter

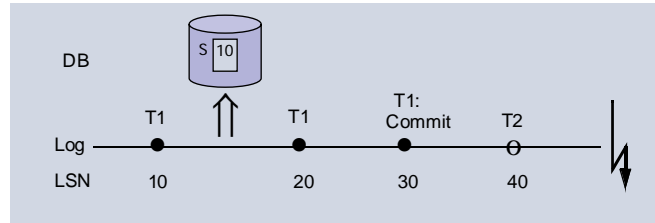
Recovery

- 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

© N. Ritter, HMS, 13.09.2007 42

Recovery

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

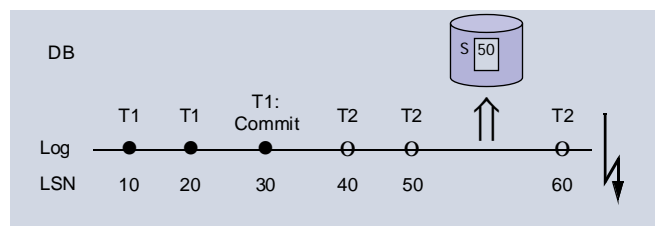


- Redo von T1: $S(10) = T1(10)$: -
 $S(10) < T1(20)$: Redo, $S(20)$
- Undo von T2: $S(20) < T2(40)$: -
- Seiten-LSN wird demnach bei Redo aktualisiert (wächst monoton)




Recovery

- 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





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




© N. Ritter

Recovery


- 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, HMS, 13.09.2007

45



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




© N. Ritter

Recovery


- 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 (*noforce*) → Redo
 - oder ‚schmutzig‘ (*steal*) → Undo

© N. Ritter, HMS, 13.09.2007

46



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




© N. Ritter

Recovery


- 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

© N. Ritter, HMS, 13.09.2007

47



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



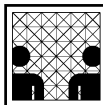
© N. Ritter

Recovery

- 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

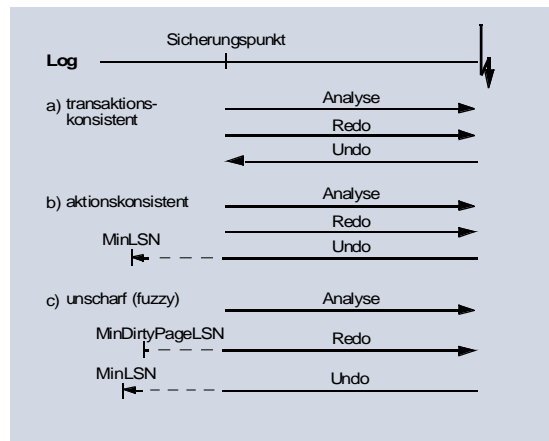
© N. Ritter, HMS, 13.09.2007

48

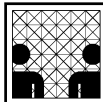


Recovery

■ Allgemeine Restart-Prozedur (Forts.)



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



Recovery

■ Restart-Prozedur bei Update-in-Place

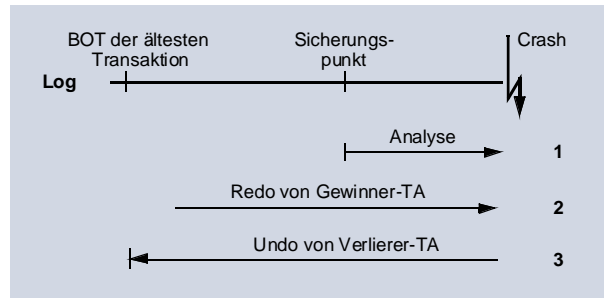
- Attribute: non-atomic, steal, noforce, fuzzy checkpoints
- Ablauf
 1. Analyse-Phase
 - vom letzten Sicherungspunkt bis zum Log-Ende
 2. Redo-Phase
 - Startpunkt abhängig vom Sicherungspunkt-Typ: hier MinDirtyPageLSN
 - selektives Redo: nur Wiederholung der Änderungen der Gewinner-TA
 3. Undo-Phase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

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



Recovery

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

N. Ritter, HMS, 13.09.2007

51

Recovery

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)



© N. Ritter

N. Ritter, HMS, 13.09.2007

52

Recovery

- 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!
 - 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“)

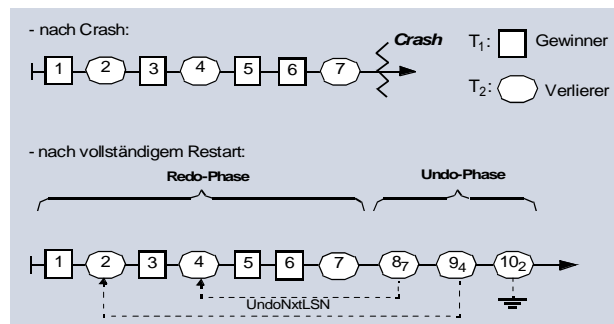


- Begriffe
- Logging
- Komponenten
- Sicherungs-punkte
- Recovery



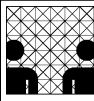
Recovery

- Compensation Log Records (CLR)
 - Optimierte Lösung (Forts.)



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





Recovery



Begriffe

Logging

Komponenten

Sicherungs-
punkte

Recovery

- 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's benötigen keine Undo-Information, da sie während nachfolgender Undo-Phasen übersprungen werden (UndoNxtLSN)



© N. Ritter

N. Ritter, HMS, 13.09.2007

55