Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
Prof. Dr. Christian Böhm
Bianca Wackersreuther

Datenbanksysteme II

SS 2007

Übungsblatt 3: Recovery

Aufgaben des Logging:

Jede Änderungsoperation auf der DB im Normalbetrieb wird protokolliert

- *REDO*: Informationen zum Nachvollziehen der Änderungen erfolgreicher TAs
- *UNDO*: Informationen zum Zurücknehmen der Änderungen unvollständiger TAs

Klassifikation von Logging-Verfahren:

- *physisch*: Protokoll auf Ebene der Seiten, Datensätze, Indexeinträge
  - *Zustands-Logging*: alter Zustand *Before-Image (BFIM)*, neuer Zustand *After-Image (AFIM)*:
  - *Übergangs-Logging*: Speicherung der Zustandsdifferenz
- *logisch*: Speicherung der Änderungsoperationen mit ihren Parametern (z.B. INSERT mit Attributwerten) → kurze Logeinträge
- *physiologisch*: Kombination von physischem und logischem Logging

Struktur der Log-Einträge für Änderungen:

$$(LSN, TA\text{-}Id, Page\text{-}Id, REDO, UNDO, PrevLSN)$$

- LSN *Log Sequence Number*: eindeutige Kennung des Log-Eintrags in chronologischer Reihenfolge
- $TA\text{-}Id$: eindeutige Kennung der TA, die die Änderung durchgeführt hat
- $Page\text{-}Id$: Kennung der Seite auf der die Änderungsoperation vollzogen wurde (ein Eintrag pro geänderter Seite)
- $REDO$: gibt an, wie die Änderung nachvollzogen werden kann
- $UNDO$: beschreibt, wie die Änderung rückgängig gemacht werden kann
- $PrevLSN$: Zeiger auf vorhergehenden Log-Eintrag der jeweiligen TA (Effizienzgründe)

Log-Sätze:

Ein Log-Satz wird für jede der folgenden Aktionen geschrieben:

- *UPDATE*: nach Modifikation einer Seite wird $pageLSN$ auf LSN des UPDATE-Log-Record gesetzt
- *COMMIT*: bei Entscheidung für COMMIT wird ein COMMIT-Record mit der $TA-Id$ geschrieben und auf Platte gezwungen (erst dann ist COMMIT der TA wirklich bestätigt)
- *ABORT*: wird bei ABORT einer TA geschrieben, UNDO wird angestossen
- *END*: nach Beendigung zusätzlicher Aktionen bei COMMIT oder ABORT
- *Compensation Log Records*: beim Zurücksetzen einer TA (UNDO) müssen Aktionen rückgängig gemacht werden; geschieht durch das Schreiben einer CLR

Weitere Datenstrukturen beim Recovery:

- *TA-Tabelle*:
  - ein Eintrag pro aktive TA
  - enthält $TA\text{-}Id$, $Status$ (running/ commited/ aborted) und $lastLSN$ (letzte vergebene LSN)
- *Dirty Page-Tabelle*: Nachführen aller bereits abgeschlossenen TAs,die *noch nicht* in die DB eingebracht wurden
  - ein Eintrag pro schmutzige Seite im Puffer
  - enthält $recLSN$ (LSN des Log-Satzes, durch den *erstmals* die Seite schmutzig gemacht wurde)

Phasen der Crash-Recovery:

- ***Analysephase***: Bestimmen der Gewinner- und Verlierer-TAs seit dem Checkpoint
  Welche COMMIT?
  WELCHE ABORT?
  - *3 Aufgaben*:
    - Bestimme den Punkt im Log, an dem die REDO-Phase beginnen muss
    - Bestimme die Menge der Seiten im Puffer, die zum Crash-Zeitpunkt schmutzig waren
    - Identifiziere die TAs, die zum Crash-Zeitpunkt aktiv waren und rückgängig gemacht werden müssen (UNDO)
  - *Rekonstruiere Zustand zum Checkpoint-Zeitpunkt*
    - über $endCheckpoint$-Record
  - *Lese sequentiell vorwärts vom Checkpoint*
    - End-Record: Entferne TAs aus TA-Tabelle
    - Andere Records: Füge TAs zur TA-Tabelle hinzu, setze $lastLSN = LSN$, ändere TA-Status (COMMIT oder UNDO)
    - $Update$-Record: Wenn P nicht in $Dirty-Page$-Tabelle vorhanden: $\rightarrow$ füge P zur $Dirty-Page$-Tabelle hinzu, setze ihren $recLSN = LSN$

- ***REDO-Phase***: REDO *aller* Aktionen
  - Ablauf wird wiederholt, um den Zustand zum Zeitpunkt des Crash zu rekonstruieren:
    - Wiederholung *aller* Updates (auch der abgebrochenen TAs!), Wiederholung der CLRs
  - Vorwärtslesen vom Log-Record, dessen LSN gleich der kleinsten $recLSN$ in Dirty Page-Tabelle. Für jede LSN eines CLR oder $Update$-Log-Record, mache ein REDO der Aktion unter folgenden Voraussetzungen:
    - betroffene Seite ist nicht in Dirty Page-Tabelle, oder
    - betroffene Seite ist in nicht in Dirty Page-Tabelle, aber hat $recLSN > LSN$, oder
    - $pageLSN$ (in DB) $\geq LSN$
  - REDO einer Aktion:
    - wiederholte Ausführung der geloggten Aktion
    - setze $pageLSN$ auf LSN (kein zusätzliches Logging!)
- *UNDO-Phase*: UNDO Effekte *aller* gescheiterten TAs
  - Zurücksetzen der Verlierer-TAs in umgekehrter Reihenfolge
  - Übergehen aller Log-Einträge die zu einer Gewinner-TA gehören, für jeden Log-Eintrag, der zu einer Verlierer-TA gehört, wird die UNDO-Operation ausgeführt (egal welche LSN auf der Seite steht)
  - für jede ausgeführte UNDO-Operation wird ein CLR angelegt, der genau wie normale Log-Sätze eine eindeutige LSN zugeteilt bekommt (Wichtig für Fehlertoleranz der Recovery); der CLR enthält die folgenden Informationen:
    - LSN
    - $TA\text{-}Id$
    - $Page\text{-}IdN$
    - $REDO$: entspricht der während der UNDO-Phase ausgeführten UNDO-Operation
    - $prevLSN$
    - $UndoNxtLSN$: gewährleistet, dass CLR während nachfolgenden UNDO-Operationen übersprungen wird