Vorlesung Implementierung von Datenbanksystemen

13. Recovery

Prof. Dr. Klaus Meyer-Wegener

Wintersemester 2019/20

Lehrstuhl für Informatik 6 (Datenmanagement)
Friedrich-Alexander-Universität Erlangen-Nürnberg

13.1 Protokollierung und Wiederherstellung allgemein

- **Transaktionsparadigma fordert:**
  - Alles-oder-Nichts-Eigenschaft von Transaktionen
  - Dauerhaftigkeit erfolgreicher Änderungen
- **Voraussetzung:**
  - Sammeln von Informationen während des normalen Betriebs (Protokollierung, **Logging**)
  - Mechanismen zur Wiederherstellung des jüngsten transaktionskonsistenten DB-Zustands (**Recovery**):
    - Sichtbare Änderungen aller *offenen* (noch laufenden) Transaktionen rückgängig machen
    - Sichtbare Änderungen aller *abgeschlossenen* Transaktionen ggf. wiederholen
- **Globales Ziel:**
  - Erhaltung der physischen und logischen Konsistenz der Daten

Erhaltung physischer und/oder logischer Konsistenz

- **Physische Konsistenz**
  - Korrektheit der Speicherungsstrukturen
    - Alle Verweise und Adressen stimmen, alle Zugriffspfade sind vollständig usw.
  - Vollständig ausgeführte **Änderungsoperationen** (insert, update, delete, store, write, modify, ...) erhalten die physische Konsistenz.
- **Logische Konsistenz**
  - Korrektheit der Dateninhalte
    - Entsprechen einem (möglichen) Zustand der realen Welt
  - Vollständig ausgeführte **Transaktionen** erhalten die logische Konsistenz.
    - Sämtliche Änderungen abgeschlossener Transaktionen sind enthalten.
    - Keine Änderungen unvollständiger Transaktionen sind enthalten.
- **Merke:**
  - Logische Konsistenz setzt physische Konsistenz voraus!
    - Ohne physische Konsistenz ist die DB gar nicht benutzbar.

Fehlerarten

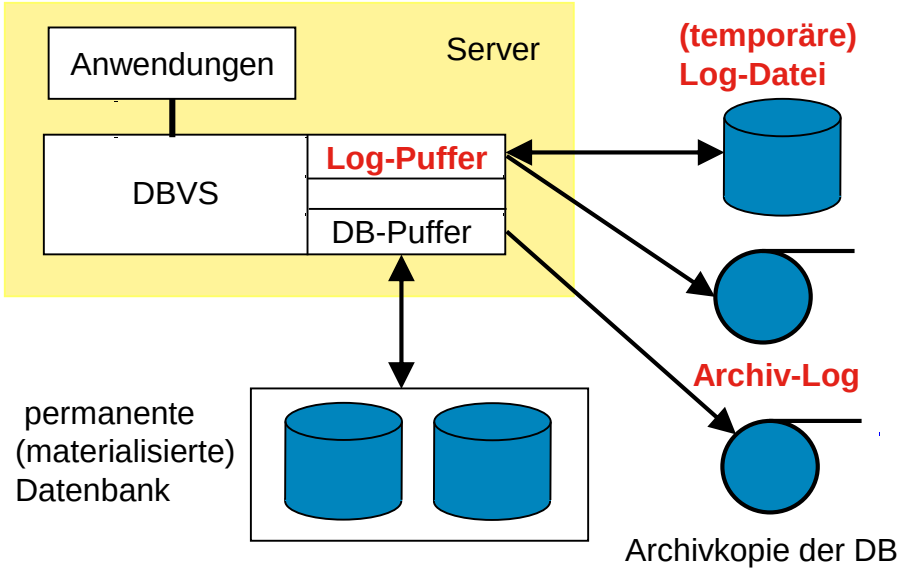
- (Siehe Kapitel 8)
- **Transaktionsfehler**
  - Verletzung von Systemrestriktionen
    - Verstoß gegen Sicherheitsbestimmungen
    - Übermäßige Betriebsmittelanforderungen / Verklemmungen
  - Anwendungsbedingte Fehler
    - Z.B. falsche Operationen und Werte
  - Aufruf von "Rollback" bzw. "Abort"
- **Systemfehler**
  - Mit Verlust aller Hauptspeicherinhalte (Puffer)
- **Gerätefehler (insbesondere Medienfehler)**
  - Zerstörung von Hintergrundspeichern (Magnetplatte)
- **(Katastrophen)**
  - Zerstörung des Rechenzentrums – hier nicht behandelt

Recovery-Klassen

- **Partial Undo (partielles Zurücksetzen / R1-Recovery)**
  - Nach Transaktionsfehler
  - Isoliertes und vollständiges Zurücksetzen der veränderten Daten in den Zustand zu Beginn der (einen) Transaktion
  - Beeinflusst andere Transaktionen nicht!
- **Partial Redo (partielles Wiederholen / R2-Recovery)**
  - Nach Systemfehler (mit Verlust des Hauptspeicherinhalts)
  - Wiederholung aller verlorengegangenen Änderungen (waren nur im Puffer) von abgeschlossenen Transaktionen
- **Global Undo (vollständiges Zurücksetzen / R3-Recovery)**
  - Nach Systemfehler (mit Verlust des Hauptspeicherinhalts)
  - Zurücksetzen aller durch den Ausfall abgebrochenen Transaktionen
- **Global Redo (vollständiges Wiederholen / R4-Recovery)**
  - Nach Gerätefehler
  - Einspielen einer Archivkopie auf neuen Datenträger und Nachvollziehen aller beendeten Transaktionen, die nach der letzten beendeten Transaktion auf der Archivkopie noch ausgeführt wurden

Arten von Sicherungsinformationen

- **Archivkopien**
  - Werden regelmäßig erstellt (hoffentlich ...), z.B. auf Magnetband
  - Auch nicht so ganz einfach:
    - "Cold Backup": Datenbanksystem muss außer Betrieb sein
    - "Hot Backup": im laufenden Betrieb – mit Beeinträchtigung
    - Datenmenge einfach zu groß
      - Ansätze dann: crash-freie Plattensysteme, RAID-Verfahren
- **Protokolldateien**
  - Protokolldateien müssen die Information enthalten, die dann R1-, R2- und R3-Recovery ermöglicht.
  - Protokollverfahren:
    - Physisches Protokollieren
    - (Logisches Protokollieren)



13.2 Einbringstrategien

- **Einbringen:**
  - Gültigmachen von Datenobjekten in der Datenbank, so dass sie auch nach Fehlern benutzt werden können
- **Problem ist der Datenbank-Puffer:**
  - Verdrängung auf Hintergrundspeicher
    - erfolgt unabhängig von Transaktionen,
    - und Inhalt des gesamten Puffers ist nach Systemfehler verloren!
  - Bereits im Zusammenhang mit Pufferverwaltung diskutiert:
    - direktes und indirektes Einbringen
- **Neue Überlegung im Zusammenhang mit Wiederherstellung:**
  - Modifikation (Einschränkung) der Pufferersetzung im Hinblick auf Transaktionsverwaltung

Einbringstrategien – Verfeinerung

WANN werden geänderte Daten aus dem Puffer auf die Platte geschrieben?

- **Steal**:
  - Bei Verdrängung aus dem Puffer,
    ggf. auch schon vor dem Ende einer Transaktion
- **NoSteal**:
  - Frühestens am Ende einer (erfolgreichen) Transaktion
    - Kein Undo erforderlich (aber sehr große Puffer)

- **NoForce**:
  - Erst bei Verdrängung aus dem Puffer,
    also i. Allg. (deutlich) nach dem Ende einer Transaktion
- **Force**:
  - Spätestens am Ende einer (erfolgreichen) Transaktion
    - Kein Partial Redo erforderlich

Einbringstrategien – Verfeinerung (2)

WIE werden geänderte Daten aus dem Puffer auf die Platte geschrieben?

- **NotAtomic:**
  - Direkte Einbringstrategie ("update in place")
  - Ist nicht ununterbrechbar zu machen
- **Atomic:**
  - Indirekte Einbringstrategie
  - Ununterbrechbares Umschalten "von alt auf neu" erreichbar
    - Z.B. mit Schattenspeicher

13.3 Protokolldaten

WAS wird in die Protokolldateien geschrieben?

- Als Protokollinformation zählt nur, was über die Einbringstrategie hinaus benötigt wird, um nach einem Systemausfall den jüngst möglichen konsistenten Zustand wiederherzustellen.

In welcher Form?	Was?	
	Zustände	Übergänge
logisch	?	Änderungs-operationen (SQL)
physisch	Before-Images, After-Images	EXOR-Differenzen

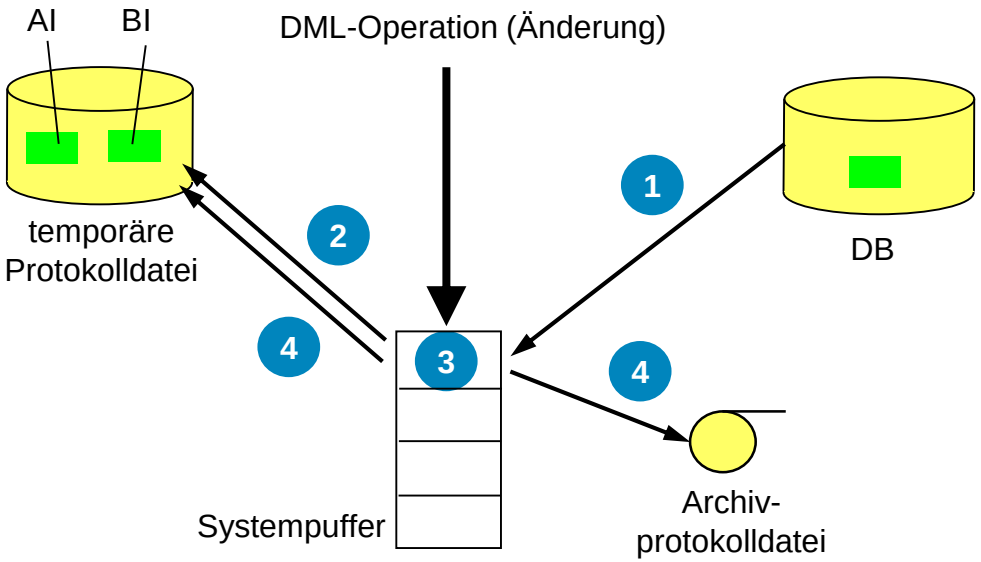
Protokolldaten (2)

WANN wird in die Protokolldatei geschrieben?

- **Undo-Information:**
  - Muss geschrieben sein, *bevor* die zugehörigen Änderungen in den Datenbestand eingebracht werden!
  - "Write Ahead Log" – **WAL-Prinzip**
  - Sonst kann Rücksetzen unmöglich sein
- **Redo-Information:**
  - Muss geschrieben sein (auf temporäre Protokolldatei und auch auf Archivprotokolldatei), *bevor* der Abschluss der Transaktion an das Programm bzw. die Benutzer gemeldet wird
    - Auch ein "WAL", aber meist nicht so genannt
  - Sonst Wiederherstellung und damit Dauerhaftigkeit der Transaktion (bzw. ihrer Ergebnisse) gefährdet

Physische Protokollierung

- **Als Zustandsprotokollierung**
  - Zustände vor bzw. nach einer Änderung werden protokolliert.
    - Alter Zustand: **Before-Image (BI),** für Undo
    - Neuer Zustand: **After-Image (AI),** für Redo
  - Einheiten der Protokollierung:
    - Seiten oder
    - Sätze (Einträge)
- **Seitenprotokollierung**
  - Für jede veränderte Seite (3) wird jeweils eine vollständige Kopie vor (2) und nach (4) der Änderung in den Log geschrieben.
    + Schnelle Recovery
    – Hoher E/A-Aufwand
  - Optimierung:
    - Nur ältestes BI per Transaktion nötig!



Physische Protokollierung (2)

- **Eintragsprotokollierung**
  - Ziel: Reduzierung des Log-Aufwands während des Normalbetriebs
  - Protokollierung nur der jeweils tatsächlich geänderten **Teile einer Seite**
    - Sätze, Index-Einträge, Freispeicher-Einträge, ...
  - Sammlung mehrerer Änderungen in einer Log-Seite und damit Pufferung im Hauptspeicher

    \+ Reduzierter E/A-Aufwand

    \+ Nutzung feinerer Sperrgranulate
      - "Whatever is logged must be locked."

    – Komplexere und zeitaufwändigere Recovery notwendig:
      - Einträge dürfen nicht mechanisch an die Stelle zurückgespeichert werden, von der sie stammen – die kann inzwischen anderweitig genutzt sein!
        - Eintrag ist evtl. inzwischen verschoben worden
      - Zurückspeichern gleicht also eher dem Neueinfügen bzw. Ändern:
        - Notfalls freien Platz suchen, TID vergeben usw.

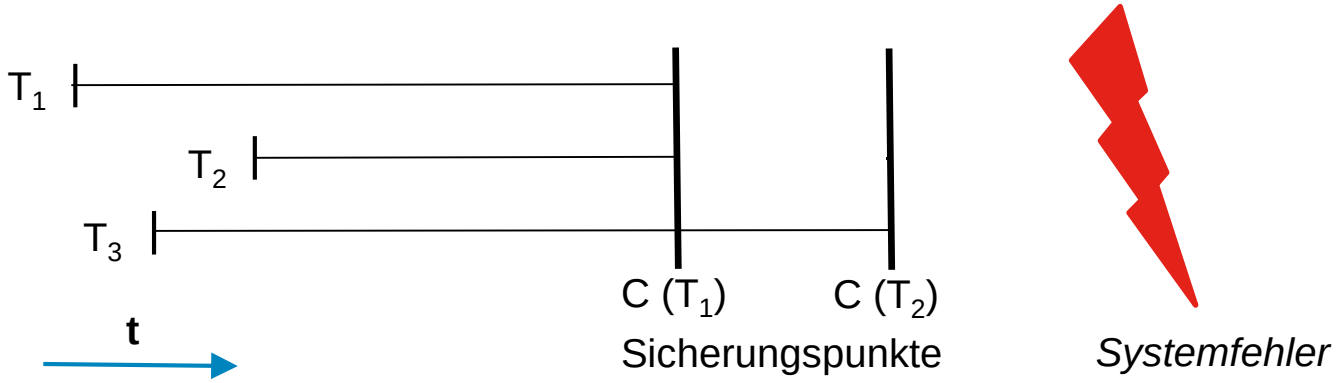
13.4 Sicherungspunkte

- **Sicherungspunkte ("checkpoints")**
  - Maßnahmen zur **Begrenzung des Redo-Aufwands** nach Systemfehlern
  - Problem bei Redo:
    - Alle erfolgreich geänderten Seiten, die zum Fehlerzeitpunkt noch im DB-Puffer vorlagen und nicht in die DB eingebracht waren, müssen rekonstruiert werden.
    - Ohne Sicherungspunkte müssten potenziell alle Änderungen seit Hochfahren des DBS wiederholt werden.
      - Nicht praktikabel! (zeitaufwändig und extremer Platzbedarf für Log)
    - Deshalb Abschnitte in der Protokolldatei markieren
- **Direkte Sicherungspunkte**
  - Ausschreiben und Einbringen aller geänderten Seiten in die Datenbank
  - Entspricht bei indirekten Einbringstrategien dem Zeitpunkt des "atomaren Umschaltens"
- **Indirekte bzw. unscharfe Sicherungspunkte ("fuzzy checkpoints")**
  - Protokollierung von Statusinformation in Log-Datei
    - Ausführliche Behandlung in der Vorl. "Transaktionssysteme"

Transaktionsorientierte Sicherungspunkte

"Transaction-Oriented Checkpoint" (TOC)

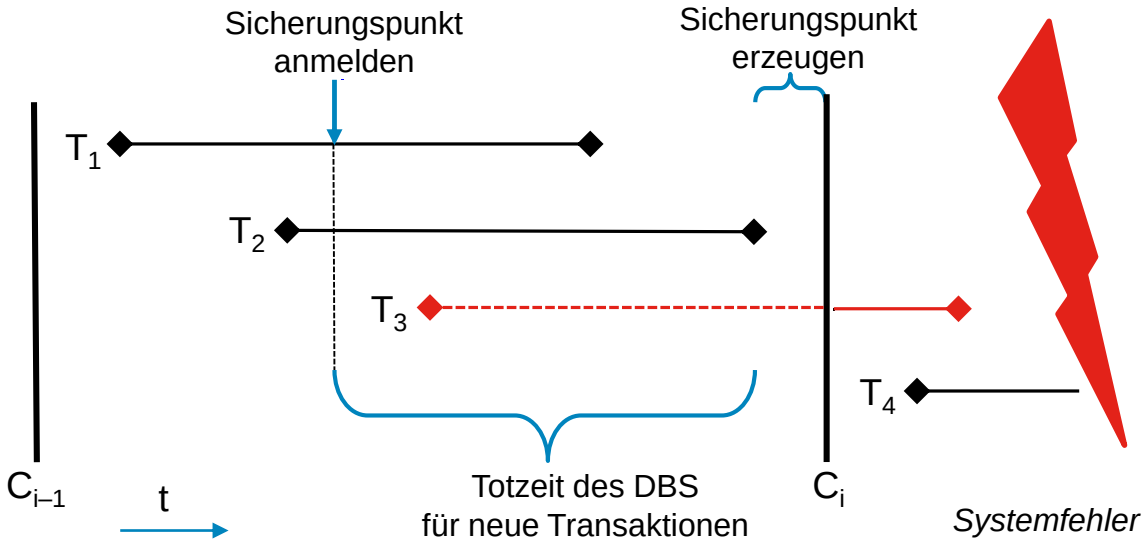
- Die geänderten Seiten einer Transaktion werden am Transaktionsende sofort in die Datenbank eingebracht (d.h. Force, siehe oben).
- Keinerlei Redo-Recovery notwendig
- Aber hohe Belastung im Normalbetrieb



Transaktionskonsistente Sicherungspunkte

"Transaction-Consistent Checkpoint" (TCC)

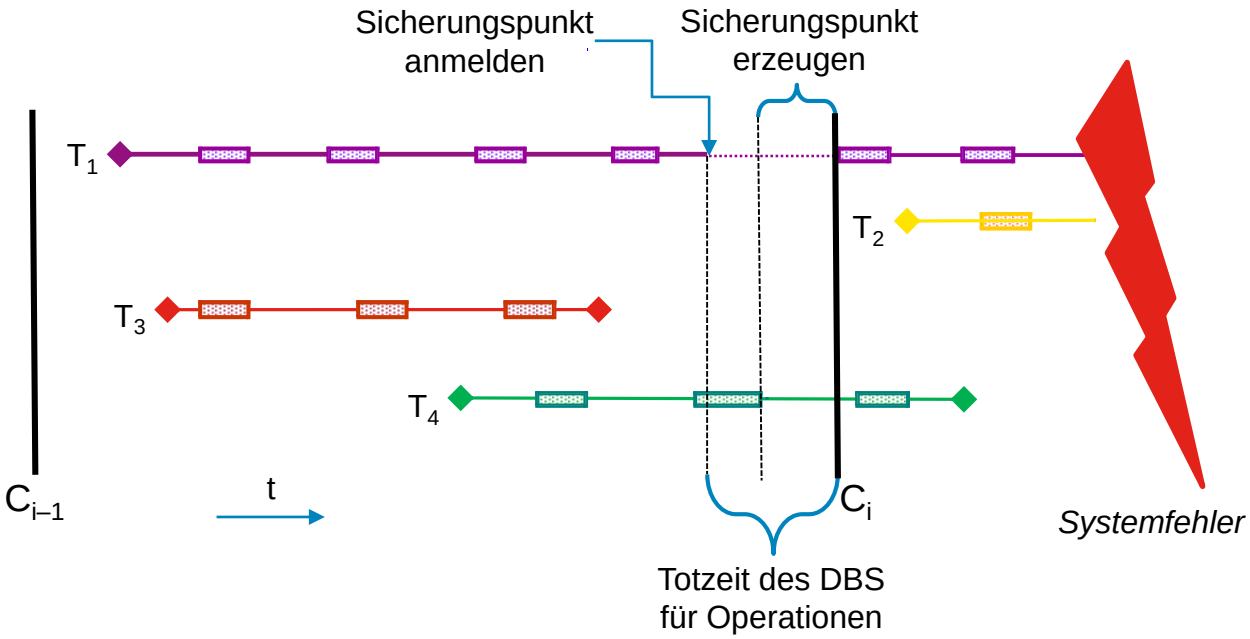
- Einbringen aller Änderungen erfolgreicher Transaktionen
- Lesesperre auf ganzer DB zur Durchführung des Sicherungspunkts
- Anmeldung eines Sicherungspunkts erzwingt Verzögerung für neue Transaktionen
- Sicherungspunkt begrenzt Undo- und Redo-Recovery



Aktionskonsistente Sicherungspunkte

"Action-Consistent Checkpoint" (ACC)

- Zum Zeitpunkt des Sicherungspunkts dürfen keine Änderungsoperationen aktiv sein.
- Kürzere Totzeit des Systems, aber auch geringere Qualität für Recovery
- Sicherungspunkt begrenzt *nur* Redo-Recovery

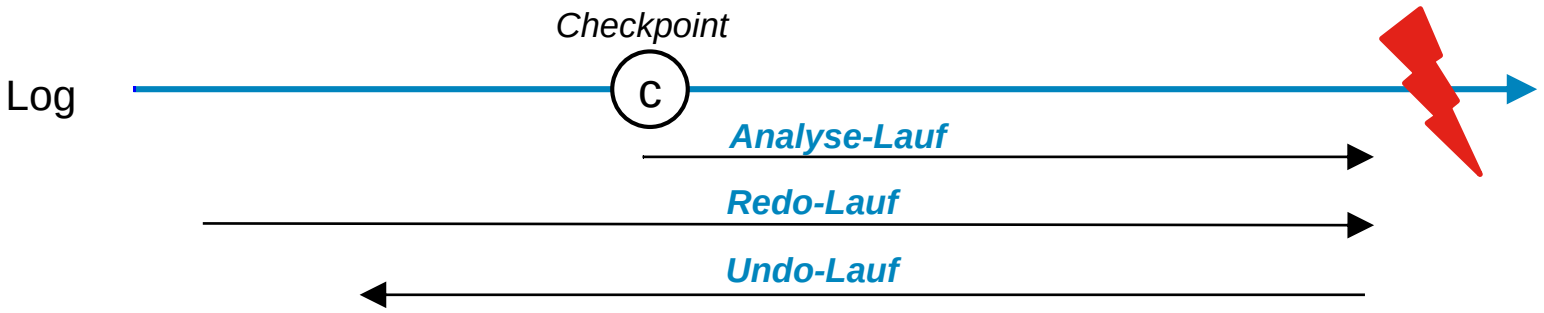


13.5 Recovery nach Systemfehler

- **Ziel**
  - Herstellung des jüngsten transaktionskonsistenten DB-Zustands aus materialisierter DB und temporärer Log-Datei
- **... bei direkter Seitenzuordnung ("update-in-place")**
  - Zustand der materialisierten DB unvorhersehbar ("chaotisch")
    - Nur physische Logging-Verfahren verwendbar
  - Ein Block der materialisierten DB ist
    - aktuell,
    - veraltet ($\rightarrow$ Redo) oder
    - "verdreckt" (geändert, aber nicht erfolgreich $\rightarrow$ Undo).
- **... bei indirekter Seitenzuordnung**
  - Materialisierte DB entspricht Zustand des letzten erfolgreichen Einbringens
    - Datenbank ist mindestens physisch konsistent
      - Logisches Logging verwendbar!
    - Je nach Art des Sicherungspunkts sogar logisch konsistent

Allgemeine Restart-Prozedur

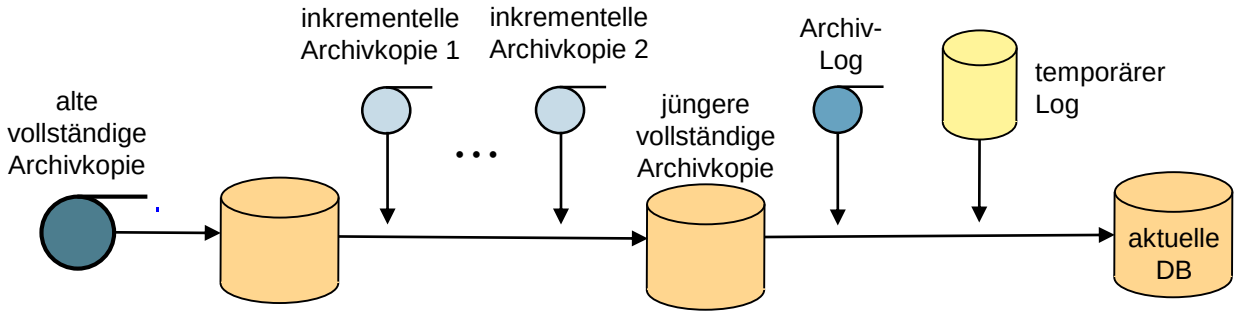
- **3-phasiger Ansatz (Lesen der temporären Log-Datei)**
  - **Analyse-Lauf**
    - Vom letzten Checkpoint vorwärts bis zum Log-Ende
    - Bestimmung der "Gewinner"- und "Verlierer"-Transaktionen
  - **Redo-Lauf**
    - Vorwärtslesen des Logs (Startpunkt abhängig vom Checkpoint-Typ)
    - Änderungen der "Gewinner"-Transaktionen ggf. wiederholen
  - **Undo-Lauf**
    - Rücksetzen der "Verlierer"-Transaktionen
      durch Rückwärtslesen des Logs
      bis zum BOT-Satz der ältesten "Verlierer"-Transaktion



(siehe auch Anhang)

13.6 Geräte-Recovery

- **Vorbereitung**
  - Die Wahrscheinlichkeit eines Gerätefehlers so weit wie möglich reduzieren (RAID-Systeme, Hot-Standby-Konfigurationen, ...)
- **Meist Plattendefekte!**
- **Datenbankrekonstruktion basierend auf**
  - Archiv-Log
    - "Full Backup" versus "Incremental Backup"
  - Temporärem Log
    - Was passiert, wenn die dafür verwendete Platte auch defekt ist?



13.7 Zusammenfassung Recovery

- **Fehlerarten und Arten der Recovery**
  - Transaktionsfehler: **Partial Undo** (R1)
  - Systemfehler: **Partial Redo** (R2) und **Global Undo** (R3)
  - Medienfehler: **Global Redo** (R4)
- **Protokollierungsverfahren**
  - Zustandsprotokollierung: physisches Logging (Seiten bzw. Einträge)
- **Sicherungspunkte**
  - Transaktionsorientierte Sicherungspunkte
  - Transaktionskonsistente Sicherungspunkte
  - Ablaufkonsistente Sicherungspunkte
- **Drei Phasen der allgemeinen Recovery-Prozedur**

Anhang

- **Die folgende Folie war bis zum Wintersemester 2015/16 als Folie 13 - 19 im Einsatz. Wir haben sie jetzt geändert, damit sie der Abb. 15.16 auf Seite 481 des Buches von Härder und Rahm entspricht.**
  - Dazu wurde die Reihenfolge von Redo-Lauf und Undo-Lauf vertauscht.
- **Beide Versionen sind richtig.**
  - Der Analyse-Lauf trennt Gewinner- und Verlierer-Transaktionen voneinander, so dass sie sich bei der Recovery nicht beeinflussen können.
- **Wir wollten diese Änderung aber nicht ohne einen Hinweis machen, weil das beim Vergleich zweier Versionen dieses Foliensatzes zu Verwirrung hätte führen können.**

Allgemeine Restart-Prozedur

- **3-phasiger Ansatz (Lesen der temporären Log-Datei)**
  - Analyse-Lauf
    - Vom letzten Checkpoint bis zum Log-Ende
    - Bestimmung von Gewinner- und Verlierer-Transaktionen
  - Undo-Lauf
    - Rücksetzen der Verlierer-Transaktionen durch Rückwärtslesen des Logs bis zum BOT-Satz der ältesten Verlierer-Transaktion
  - Redo-Lauf
    - Vorwärtslesen des Logs (Startpunkt abhängig vom Checkpoint-Typ)
    - Änderungen der Gewinner-Transaktionen werden ggf. wiederholen