Synchronisation

VERTEILTER DATENBANKEN

Inhalt

- Grundlagen
  - (D- & L- & V-)Verbund
  - Fragmentierung
- Verteilung bzw. Replikation
  - Aktualisierungstechniken
  - Datenabgleich nach Fehlern
- Optimistische Synchronisation
- Mehrversionenkonzept
- Deadlock Behandlung / Verhütung

Datenverbund

- Geographische Verteilung
- Replikate

Leistungsverbund

- Speicher
- Rechenleistung

Verfügbarkeitsverbund

- Redundanz

Anforderungen an ein verteiltes Datenbanksystem

- Lokale Autonomie
- Dauerbetrieb
- Verteilungstransparenz
- Parallele Queries
- Transparenz von Hard- und Software

DDBS

Vorteile

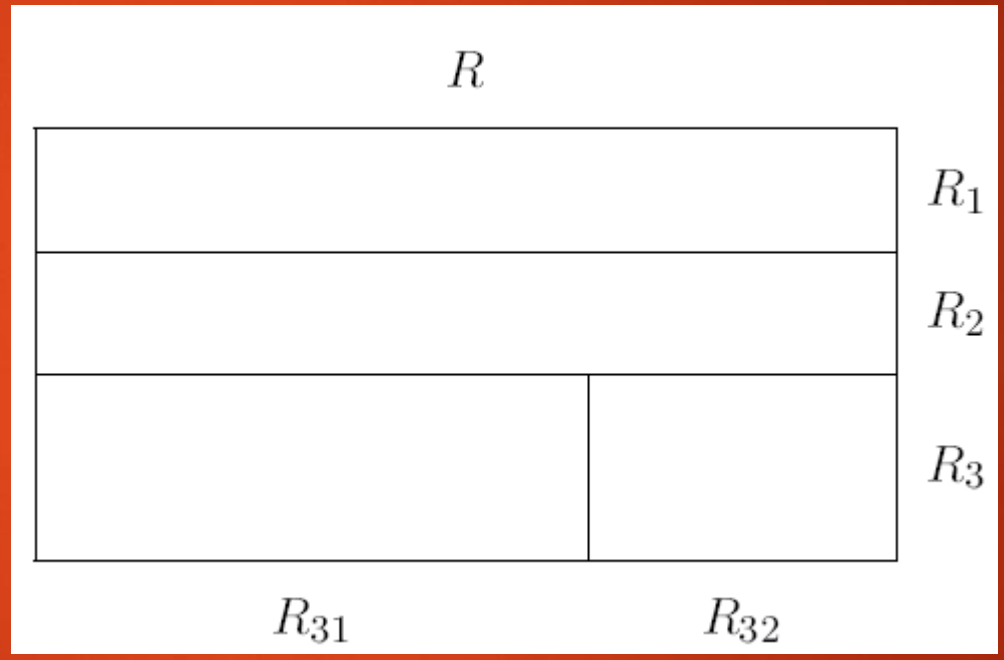
- Datenverbund
- Lokale Autonomie
- Verfügbarkeit
- Performance
- Datenintegrität
- Transparenter Zugriff

Nachteile

- Verwaltungsaufwand
- Kosten für Entwicklung
- Komplexität / Fehlerrisiko

Fragmentierung

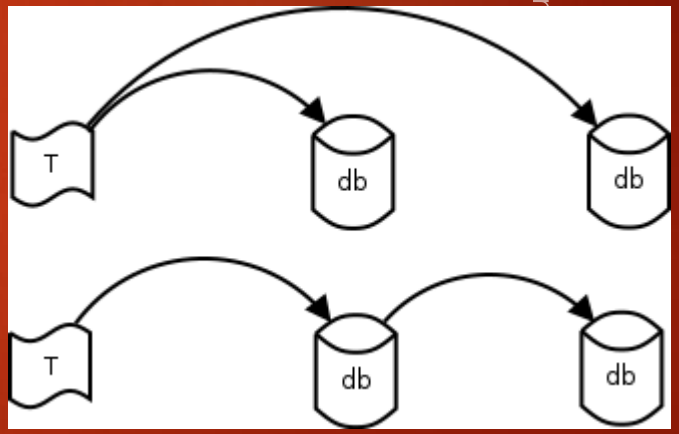
- Horizontale
- Vertikale
- Hybride



[1]

Replikat

- Aktualisierungstechniken
  - Synchron (Speicherung auf allen Repl. während Transaktion)
  - Asynchron (zeitlich versetzte Speicherung nach Abschluss der Transaktion)
- Datenabgleich nach Fehlern
  - Physisch (kopieren der korrekten Relationen)
  - Logisch (wiederholen von Transaktionen mit Hilfe des Trans.log)

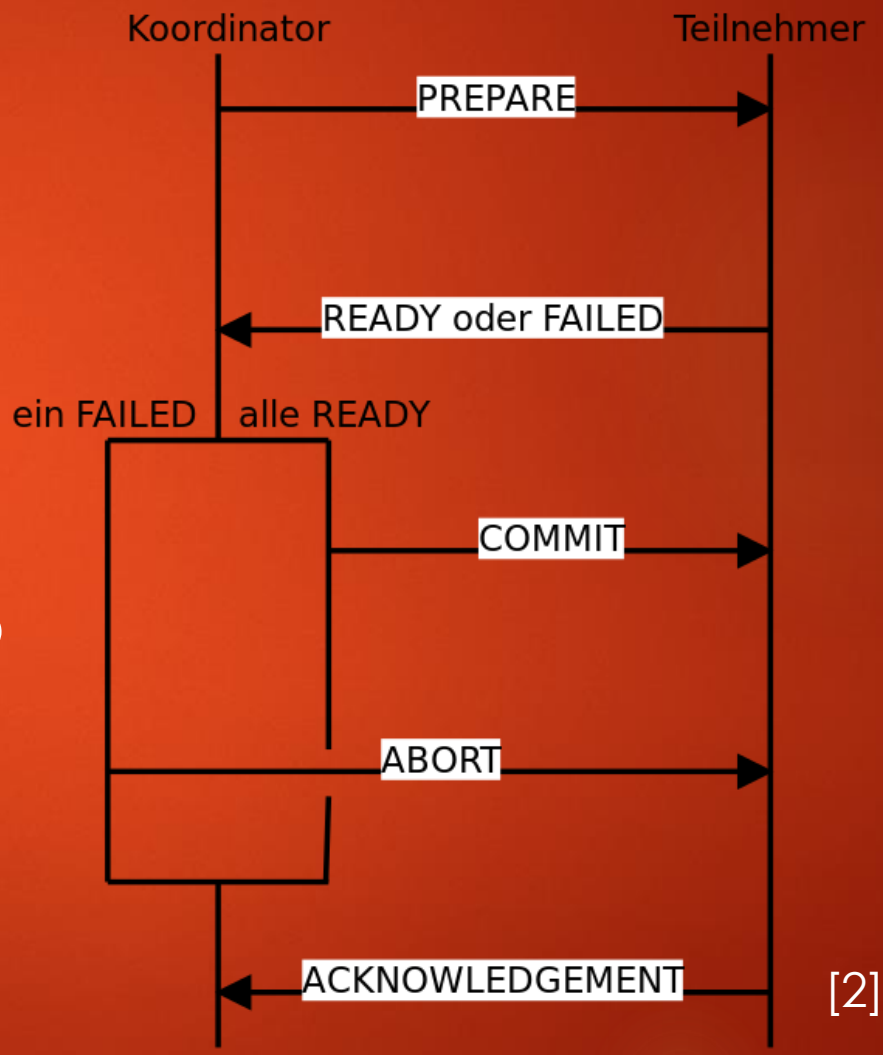


Optimistische Synchronisation

- Konflikte zwischen Trans. werden als seltene Ereignisse gesehen
  - Sperren unnötig
- Lesephase
  - Lesen und modifizieren im Trans.Puffer
- Validierungsphase (EOT)
  - Konfliktprüfung -> evtl. Auflösung der Trans.
- Schreibphase
  - Logging (Wiederholbarkeit)
  - Schreiben der modifizierten Objekten

Verteilte Validierung 2PC

- 2 Rollen (TM & S)
- TM sendet PREPARE an alle S
- S antworten mit READY oder FAILED
- TM sendet COMMIT oder ABORT



[2]

Mehrversionenkonzept

- Transaktionen müssen angeben ob sie Änderungen durchführen
- Bei Lesetransaktionen ließt die T lediglich Daten die BOT gültig sind
  - Zwischenzeitliche Änderungen bleiben unsichtbar
- Schreibetransaktionen greifen stets auf die aktuellste Version zu
  - Erstellt dabei neue Version (Versionspool)
  - Synchronisation

Deadlock Verhinderung

- Preemption durchführen
- Hold and Wait verhindern
- Mutual Exclusion beseitigen
- Circular Wait ausschließen

Deadlock Vermeidung

- Wait / die
  - T_a fordert Sperre an die von T_j gehalten wird, so wartet sie bis T_j fertig ist
  - T_j fordert Sperre an die von T_a gehalten wird, so startet T_j neu
- Wound / wait
  - T_a bricht Sperre von T_j ab (Neustart), T_a bekommt die Sperre
  - T_j wartet solange bis T_a fertig ist

Zusammenfassung

- Fragmentierung
- Aktualisierungsmöglichkeiten
- Daten-, Leistungs- & Verfügbarkeitsverbund
- 2PC & MVK
- Deadlock Behandlung

Bildquellen

[1] Sideplayer; Kapitel 15 Verteilte Datenbanken Motivation [online]
verfügbar unter: http://slideplayer.org/slide/206223/
zuletzt aufgerufen 01.03.2016

[2] Wikipedia; Commit-Protokoll [online]

verfügbar unter:
https://upload.wikimedia.org/wikipedia/commons/thumb/7/79/Zwei_phasen_commit.svg/220px-Zwei_phasen_commit.svg.png

Zuletzt aufgerufen 01.03.2016