LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH
DEPARTMENT INSTITUTE FOR INFORMATICS
DATABASE SYSTEMS GROUP

Skript zur Vorlesung:

Datenbanksysteme II

Sommersemester 2016

Kapitel 2
Transaktionsverwaltung

Vorlesung: PD Dr. Peer Kröger

http://www.dbs.ifi.lmu.de/cms/Datenbanksysteme_II

Dieses Skript basiert in Teilen auf den Skripten zur Vorlesung Datenbanksysteme II an der LMU München von

Prof. Dr. Christian Böhm (SoSe 2007),
PD Dr. Peer Kröger (SoSe 2008, 2013, 2014, 2015) und
PD Dr. Matthias Schubert (SoSe 2009)

2. Transaktionsverwaltung

2.1 Transaktionsbegriff

2.2 Operationen auf Transaktionsebene

2. Transaktionsverwaltung

2.1 Transaktionsbegriff

2.2 Operationen auf Transaktionsebene

2.1 Transaktionsbegriff

Motivation

- Beispiel
  - Überweisung von Huber an Meier in Höhe von 200 €
  - Mgl. Bearbeitungsplan:
    (1) Erniedrige Stand von Huber um 200 €
    (2) Erhöhe Stand von Meier um 200 €

  - Möglicher Ablauf

Konto	Kunde	Stand
	Meier	1.000 €
	Huber	1.500 €

(1) ⇒

Konto	Kunde	Stand
	Meier	1.000 €
	Huber	1.300 €

(2) ⇒ **System-absturz**

Inkonsistenter DB-Zustand darf nicht entstehen bzw. darf nicht dauerhaft bestehen bleiben !!!

2.1 Transaktionsbegriff

Transaktionskonzept

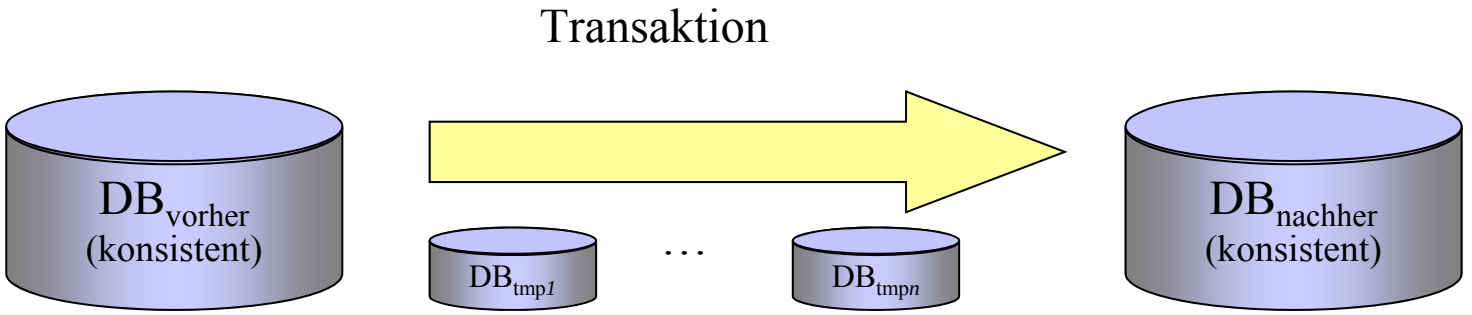
- Transaktion (TA):

  Folge von Aktionen (read, write), die die DB von einen konsistenten Zustand in einen anderen konsistenten Zustand überführt
- Anders ausgedrückt sind Transaktionen „Einheiten integritätserhaltender Zustandsänderungen einer Datenbank“
- Meist entspricht eine TA einem Benutzer(-Programm), d.h. die Aktionen der TA müssen ***nicht notwendigerweise vorher bekannt*** sein
  - Beispiel: die gesamte Interaktion eines Kunden bei einem Flugbuchungsportal ist eine Transaktion mit Operationen wie z.B.
    - verfügbare Verbindungen/Flüge abrufen (read)
    - Flug aussuchen und buchen (write)
    - Passagierdaten eingeben (write)
    - Sitze reservieren (red/write)
    - …

2.1 Transaktionsbegriff

Hauptaufgaben der Transaktions-Verwaltung

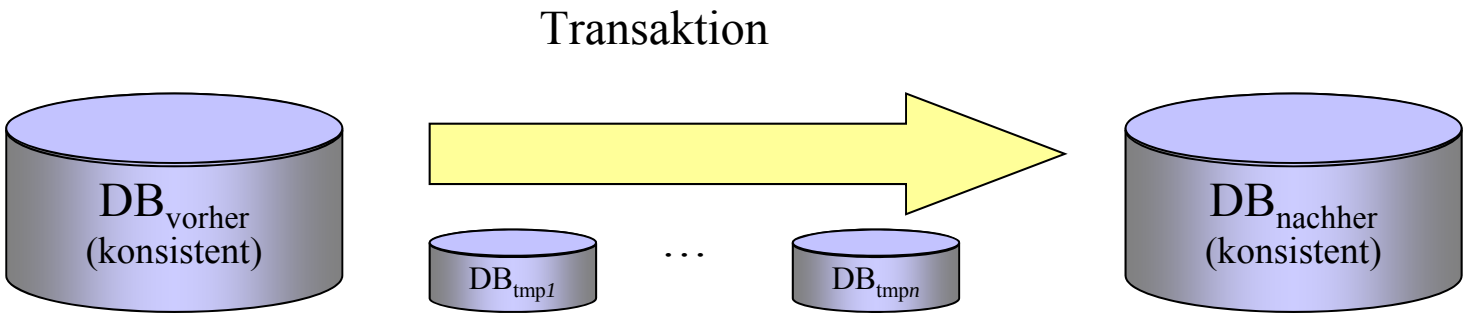
- Synchronisation (Koordination mehrerer Benutzerprozesse, ähnlich der Prozessverwaltung in Betriebssystemen)
  - i.d.R. viele Benutzer => viele gleichzeitig ablaufende TAs
  - Unkontrollierte Nebenläufigkeit verhindern
- Recovery (Behebung von Fehlersituationen)
  - Schutz gegen Fehler (in HW und/oder SW)
  - Transaktionsorientiert



2.1 Transaktionsbegriff

Zusätzlich:

- Datenintegrität
  - Integritätsbedingungen, die aus der Semantik der realen Welt abgeleitet werden
  - Bedingungen, die von einer Datenbank zu jedem Zeitpunkt erfüllt sein müssen
- Datenschutz
  - vor beabsichtigten Schäden



2.1 Transaktionsbegriff

ACID-Prinzip

- Atomicity (Atomarität)

  Effekt einer TA kommt entweder ganz oder gar nicht zum Tragen.

- Consistency (Konsistenz, Integritätserhaltung)

  Durch eine TA wird ein konsistenter DB-Zustand wieder in einen konsistenten DB-Zustand überführt.

- Isolation (Isoliertheit, logischer Einbenutzerbetrieb)

  Innerhalb einer TA nimmt ein Benutzer Änderungen durch andere Benutzer nicht wahr.

- Durability (Dauerhaftigkeit, Persistenz)

  Der Effekt einer abgeschlossenen TA bleibt dauerhaft in der DB erhalten.

- Weitere wichtige Forderung:

  TA soll in endlicher Zeit bearbeitet werden können

2. Transaktionsverwaltung

2.2 Operationen auf Transaktionsebene

Steuerung von TAs

- Begin of transaction (BOT)
  markiert den Anfang einer Transaktion

- End of transaction / commit
  markiert das Ende einer Transaktion
  alle Änderungen seit dem letzten BOT werden festgeschrieben

- Abort
  markiert den Abbruch einer Transaktion
  die Datenbasis wird in den Zustand vor BOT zurückgeführt

2.2 Operationen auf Transaktionsebene

TA-Verwaltung in SQL

- Begin of transaction (BOT)
  Transaktionen werden implizit begonnen, es gibt kein `begin work` o.ä.
- End of transaction / commit
  `commit` oder `commit work`
- Abort
  `rollback` oder `rollback work`

- Beispiel

```
UPDATE Konto SET Stand = Stand-200 WHERE Kunde = ‘Huber’;

UPDATE Konto SET Stand = Stand+200 WHERE Kunde = ‘Meier’;

COMMIT;
```

2.2 Operationen auf Transaktionsebene

Unterstützung langer Transaktionen

- Motivation

  BOT, COMMIT, ABORT reichen normalerweise zur TA-Verwaltung aus; meist dauern TAs aber sehr lange (viele read/write Operationen); dann ist es sinnvoll, zwischenzeitlich ***Sicherungspunkte*** setzen zu können

- Umsetzung
  - Define savepoint (SQL: `savepoint <identifier>`)
    - markiert Sicherungspunkt, auf den sich eine noch aktive TA zurücksetzen lässt
    - Änderungen dürfen noch nicht festgeschrieben werden, da die Transaktion noch (als Ganzes) scheitern bzw. zurückgesetzt werden kann
  - Backup transaction (SQL: `rollback to <identifier>`)
    - setzt die Datenbasis auf einen definierten Sicherungspunkt zurück
  - Bemerkungen:
    - In manchen Systemen kann nur auf den jüngsten SP zurückgesetzt werden
    - Anders als der `commit`-Befehl muss das DBMS die „erfolgreiche" Ausführung eines `rollback`-Befehls immer garantieren können

2.2 Operationen auf Transaktionsebene

Abschluss von Transaktionen

- Erfolgreich durch COMMIT

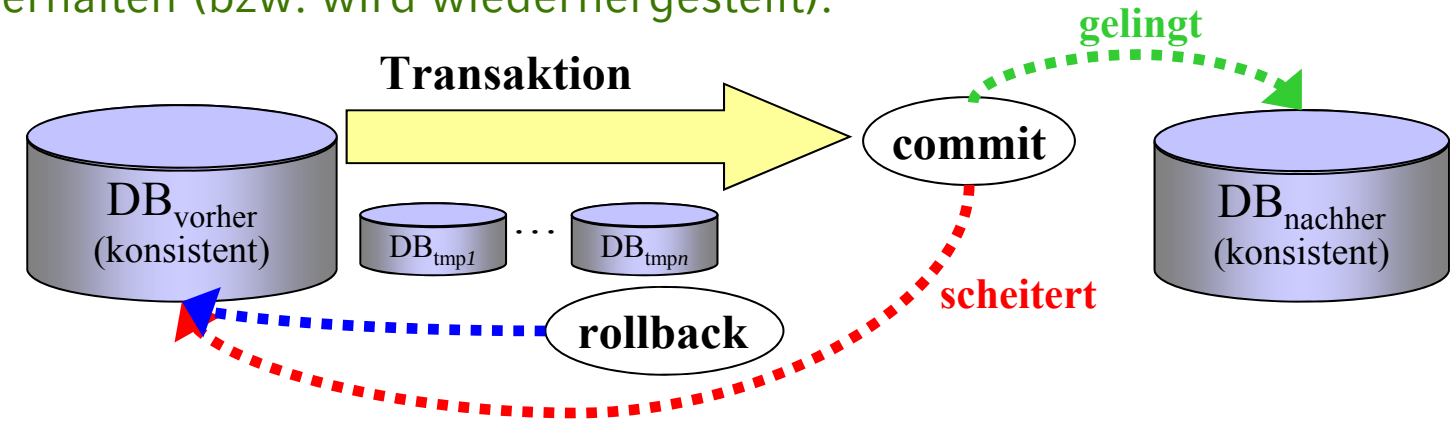
  => der neue DB-Zustand wird dauerhaft gespeichert

- Erfolglos durch ABORT

  => der ursprüngliche Zustand wie zu Beginn der Transaktion bleibt erhalten (bzw. wird wiederhergestellt). Ein COMMIT kann z.B. scheitern, wenn die Verletzung von Integritätsbedingungen erkannt wird.

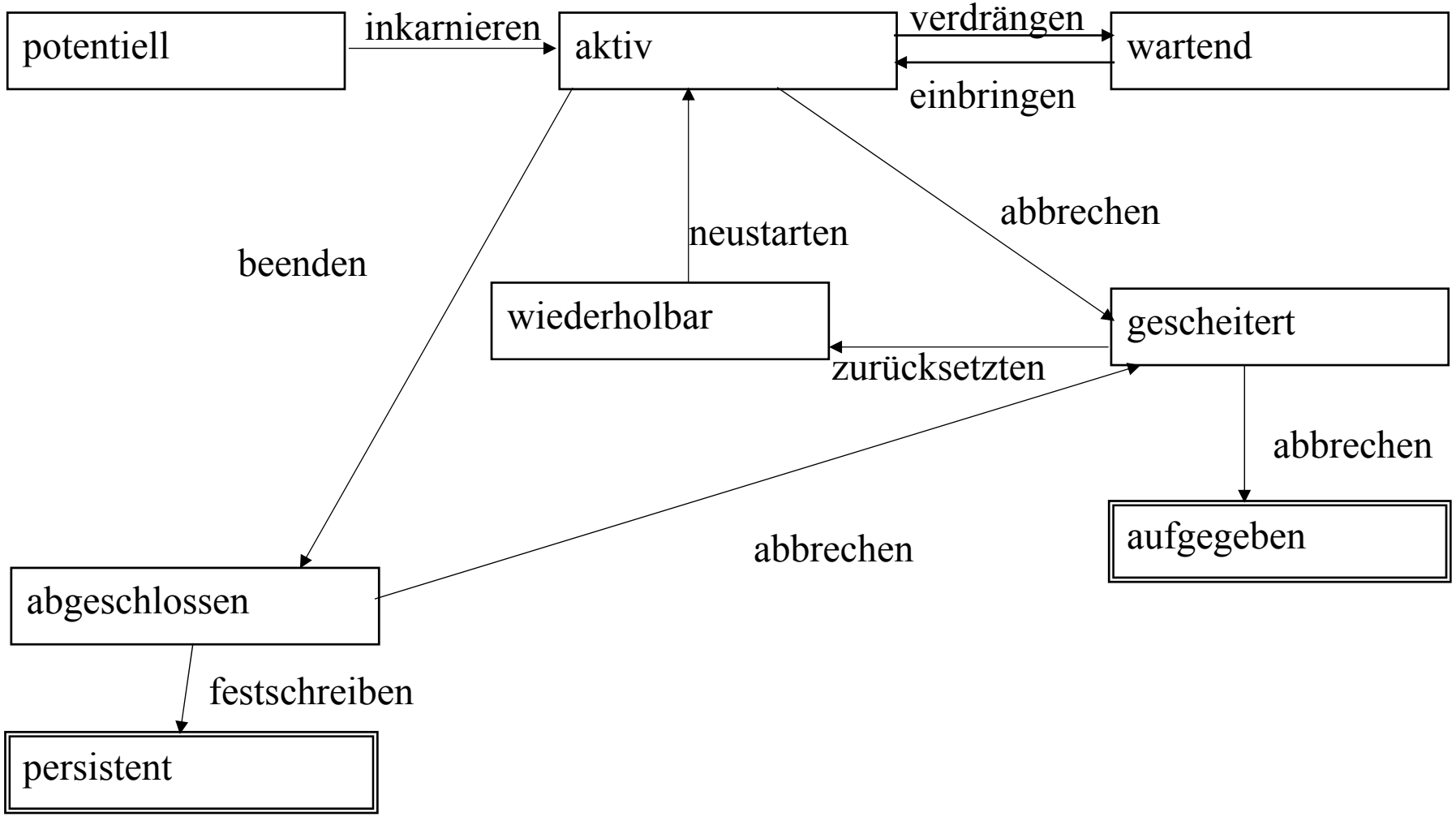
- Benutzer widerruft durch ROLLBACK

  => der Zustand zum Zeitpunkt des aufgerufenen Sicherungspunktes bleibt erhalten (bzw. wird wiederhergestellt).



2.2 Operationen auf Transaktionsebene

Zustände einer TA (aus: Kemper, Eickler: Datenbanksysteme)



2.2 Operationen auf Transaktionsebene

Zustände einer TA

- Potentiell: TA ist „gecodet“ und wartet auf ihre Abarbeitung
- Aktiv: TA wird abgearbeitet und konkurriert mit den anderen aktiven TAs um die Betriebsmittel
- Wartend: TA ist aus dem Puffer verdrängt um Platz im Puffer zu schaffen
- Abgeschlossen: eine TA ist durch ihr COMMIT abgeschlossen aber ihr Ergebnis möglicherweise noch nicht in die DB eingebracht
- Persistent: Ergebnis der TA ist dauerhaft in der DB gespeichert
- Gescheitert: TA ist gescheitert (z.B. durch ABORT, ROLLBACK, Verletzung von Konsistenzbedingung, HW-Fehler, Deadlock, etc.)
- Wiederholbar: gescheiterte TAs sind u.U. wiederholbar (z.B. durch Deadlock-Behebung zurückgesetzte TAs)
- Aufgegeben: gescheiterte Tas können auch als „hoffnungslos“ aufgegeben werden