Kapitel 2
Transaktionsverwaltung

Skript zur Vorlesung: Datenbanksysteme II
Sommersemester 2008, LMU München

Dieses Skript basiert zu einem Teil auf dem Skript zur Vorlesung Datenbanksysteme II von Prof. Dr. Christian Böhm gehalten im Sommersemester 2007 an der LMU München

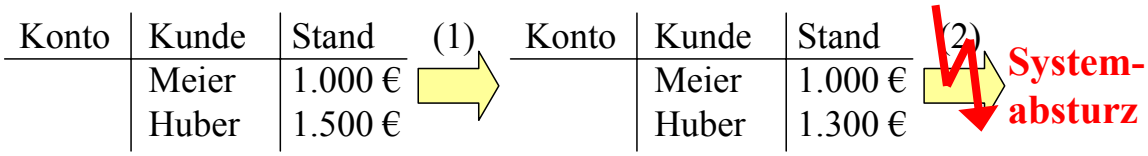
Übersicht

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



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

Transaktionskonzept

– Transaktion:

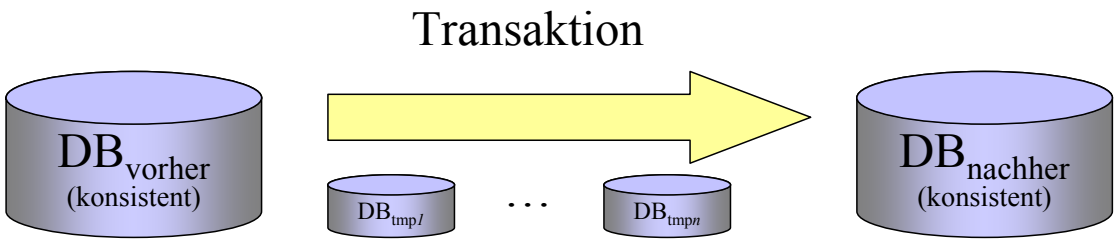
  Folge von Aktionen (*read, write*), die die DB von einen konsistenten Zustand in einen anderen konsistenten Zustand überführt

– Transaktionen:

  Einheiten integritätserhaltender Zustandsänderungen einer Datenbank

Hauptaufgaben der Transaktions-Verwaltung

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



ACID-Prinzip

- **A**tomicity (*Atomarität*)
  Effekt einer TA kommt entweder ganz oder gar nicht zum Tragen.
- **C**onsistency (*Konsistenz*, *Integritätserhaltung*)
  Durch eine TA wird ein konsistenter DB-Zustand wieder in einen konsistenten DB-Zustand überführt.
- **I**solation (*Isoliertheit*, *logischer Einbenutzerbetrieb*)
  Innerhalb einer TA nimmt ein Benutzer Änderungen durch andere Benutzer nicht wahr.
- **D**urability (*Dauerhaftigkeit*, *Persistenz*)
  Der Effekt einer abgeschlossenen TA bleibt dauerhaft in der DB erhalten.
- Weitere wichtige Forderung:
  TA in endlicher Zeit bearbeitet werden können

Übersicht

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

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;
```


Unterstützung langer Transaktionen

- Motivation
  - Die drei Befehle BOT, COMMIT, ABORT reichen normalerweise zur TA-Verwaltung aus
  - In modernen Anwendungen dauern TAs aber meist sehr lange (viele read/write Operationen)
  - In diesem Fall ist es sinnvoll, zwischenzeitlich Sicherungspunkte setzen zu können

Unterstützung langer Transaktionen (cont.)

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

Abschluss von Transaktionen

- Abschluss einer TA
  - 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 Sicherungspunkt bleibt erhalten (bzw. wird wiederhergestellt).