Transaktionsverwaltung

Transaktionsverwaltung

Transaktionsverwaltung

- Transaktionsverwaltung beinhaltet die folgenden zwei Teilgebiete
  - **Recovery**, d.h. die Behebung von eingetretenen, oft unvermeidbaren Fehlersituationen.
  - **Synchronisation** von mehreren gleichzeitig auf der Datenbank ablaufenden Transaktionen.

Beispiel für Transaktion (TA)

- Überweise Geld von Konto A nach Konto B:
  - Lies den Kontostand von A in die Variable a: **read**(A,a);
  - Reduziere den Kontostand um EURO 50,–: $a/ := a/ - 50$;
  - Schreibe den neuen Kontostand in die Datenbasis: **write**(A,a);
  - Lies den Kontostand von B in die Variable b: **read**(B,b);
  - Erhöhe den Kontostand um EURO 50,–: $b/ := b/ + 50$;
  - Schreibe den neuen Kontostand in die Datenbasis: **write**(B,b);

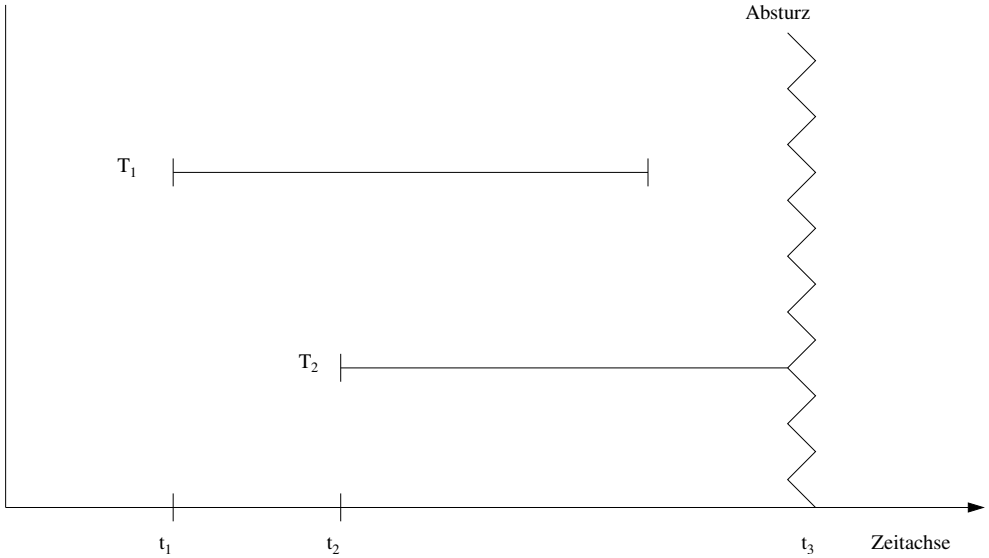
Operationen

- **begin of transaction** (**BOT**):
  - Kennzeichnet den Beginn einer Transaktion
- **commit**:
  - Erfolgreiche Beendigung einer Transaktion
  - Dauerhafte Einbringung aller Änderungen in die Datenbasis
- **abort**:
  - Selbstabbruch der Transaktion, erfolglose Beendigung
  - Zurücksetzen der Datenbasis in den Zustand vor Beginn der Transaktion

Operationen(2)

- **define savepoint**:
  - Sicherungspunkt definieren, auf den sich die (noch aktive) Transaktion zurücksetzen lässt.
- **backup transaction**:
  - Die noch aktive Transaktion wird auf den jüngsten (zuletzt angelegten) Sicherungspunkt zurückgesetzt
  - Evtl. auch Rücksetzen auf weiter zurückliegende Sicherungspunkte möglich

Systemabsturz



- Änderungen der zum Zeitpunkt t_3 abgeschlossenen TA T_1 müssen in der Datenbasis vorhanden sein
- Änderungen der zu t_3 noch nicht abgeschlossenen TA T_2 müssen vollständig aus der Datenbasis entfernt werden (Durchführung von T_2 durch Neustart)

- **commit (work)**:
  - Beende Transaktion und schreibe Änderungen fest
  - funktioniert nur, wenn keine anderen Fehler aufgetreten sind (z.B. Konsistenzverletzung durch Transaktion)
- **rollback (work)**:
  - Beende Transaktion und setze alle Änderungen zurück
  - Anders als **commit** muss DBMS die "erfolgreiche" Ausführung von **rollback** immer garantieren können

- Beispiel:

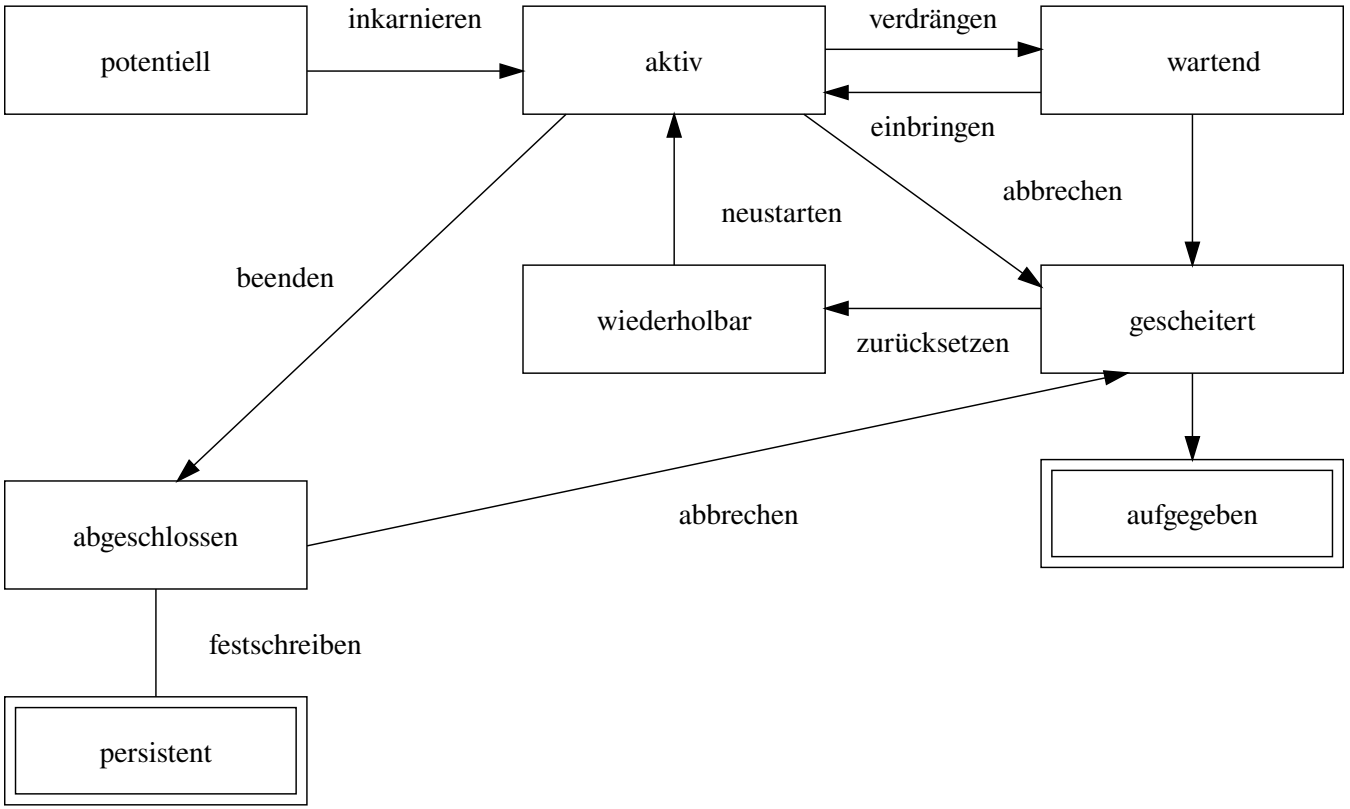
```
insert into Vorlesungen
    values (5275, 'Kernphysik', 3, 2141);

insert into Professoren
    values (2141, 'Meitner', 'C4', 205);
```

commit work

- **commit** nach dem ersten **insert** könnte nicht erfolgreich durchgeführt werden, da zu diesem Zeitpunkt referentielle Integrität verletzt

Zustandsübergänge



Zusammenfassung

- Transaktionen sollten ACID-Eigenschaften haben
- ACID:
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Zusammenfassung(2)

- Atomicity (Atomarität): “alles oder nichts”, d.h. entweder werden alle Operationen einer TA ausgeführt oder keine
- Consistency (Konsistenz): wenn eine TA auf einem konsistenten Datenbankzustand aufsetzt, ist dieser nach Beendigung der TA immer noch konsistent
- Isolation: nebenläufig ausgeführte Transaktionen dürfen keine Seiteneffekte aufeinander haben
- Durability (Dauerhaftigkeit): alle mit commit festgeschriebenen Änderungen müssen bestehen bleiben (selbst bei Systemabsturz)