Einführung in die Java Programmierung

Kapitel 14

Transaktionen und Optimistisches Locking

Transaktionen: Warum ist das wichtig?

- Sie speichern mit Serveraufrufen häufig mehrere Objekte auf einmal:
  - Transkation = Alle Objekte werden ganz / konsistent gespeichert oder garnicht
  - Gegenbeispiel: Sie speichern halbe / inkonsistente Kunden
- Sie arbeiten über einen längeren Zeitraum mit Daten, die auch andere ändern können
  - Folge: Sie brauchen ein „Sperrkonzept“ um mit den aus DB1 bekannten Phänomenen wie Lost Update umzugehen

ACID-Eigenschaften

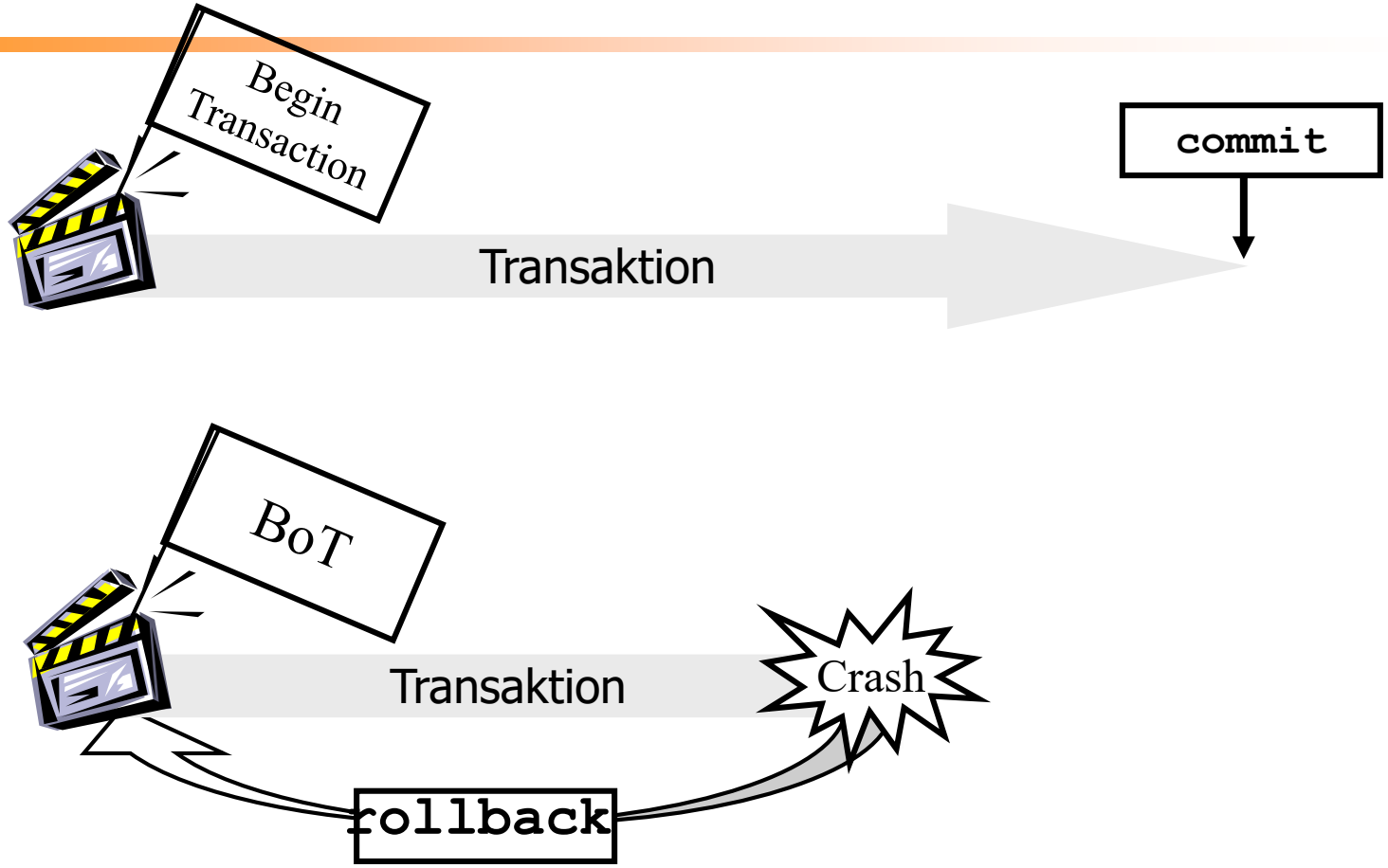
A*tomicity* (Atomarität):
Transaktion wird entweder ***ganz oder gar nicht*** ausgeführt

C*onsistency* (Konsistenz oder auch Integritätserhaltung):
Datenbank ist vor Beginn und nach Beendigung einer Transaktion jeweils in einem konsistenten Zustand

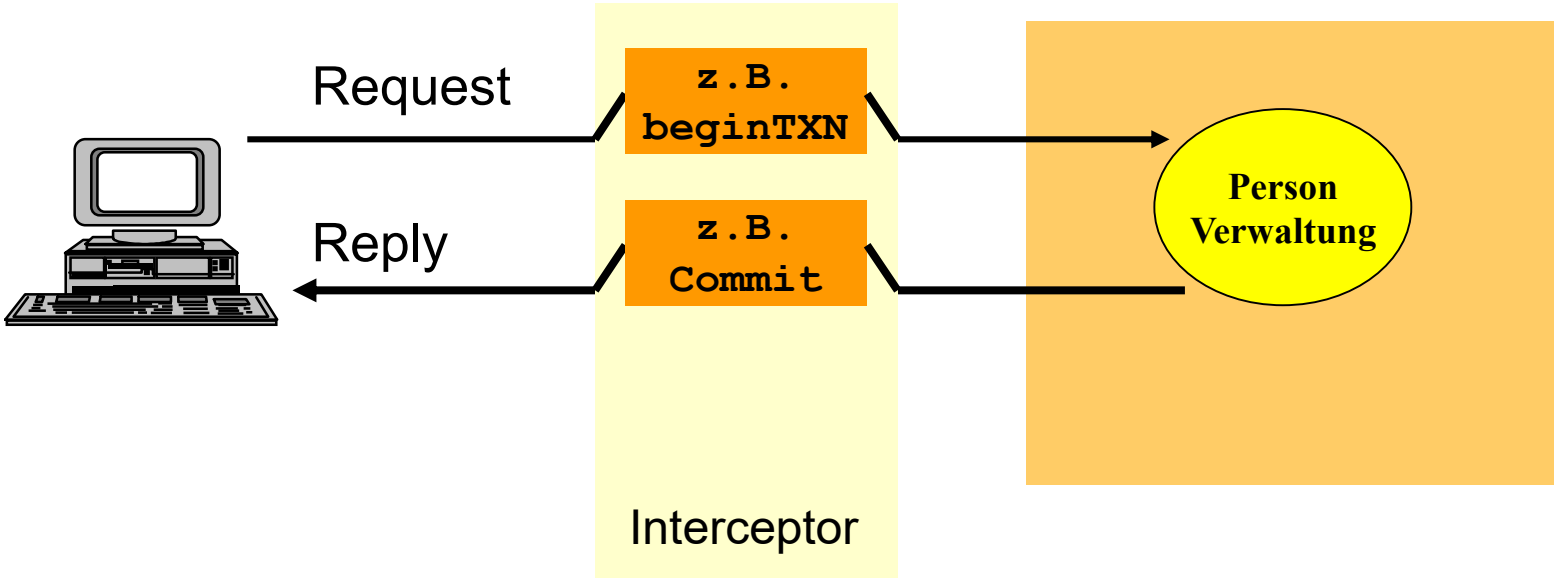
I*solation* (Isolation):
Nutzer, der mit einer Datenbank arbeitet, sollte den Eindruck haben, dass er mit dieser Datenbank alleine arbeitet

D*urability* (Dauerhaftigkeit / Persistenz):
nach erfolgreichem Abschluss einer Transaktion muss das Ergebnis dieser Transaktion „dauerhaft“ in der Datenbank gespeichert werden

Transaktionen: *Alles oder Nichts*



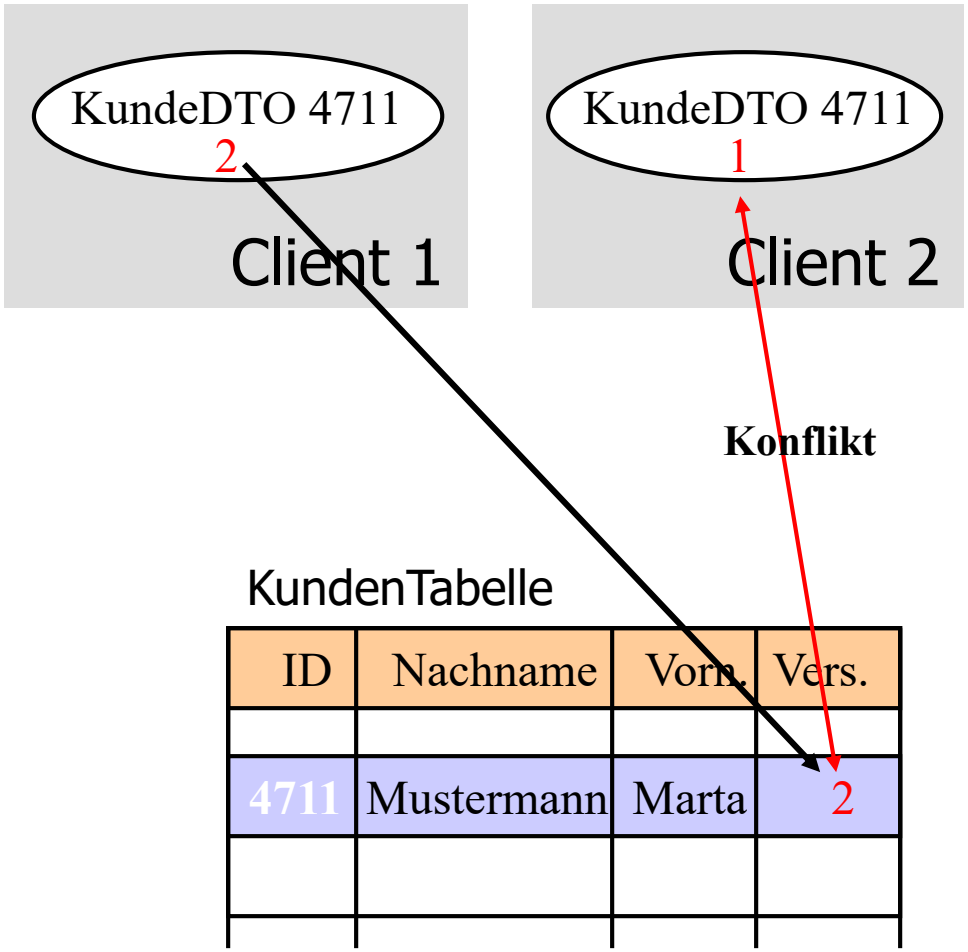
Das Interceptor Pattern



Lost Update Problem

- Problem:
  - Client hat nur ***Kopie*** von Server-Daten (z.B. Kunde)
  - Was geschieht, wenn ein anderer Client denselben Kunden parallel ändert?
- Idee: DTO bekommt ***Versionszähler*** mit dem Konflikte erkannt werden können
  - Versionszähler (z.B. Int) in DTO und in Datenbank
  - Bei jeder Änderung der Daten
    1. Eigenen Versionszähler mit Versionszähler in DBMS vergleichen
       - Wenn gleich, dann weiter bei 2
       - Wenn ungleich, Konflikt melden und Abbruch
    2. Versionszähler um eins inkrementieren
    3. Daten in die Datenbank schreiben

Beispiel



- Jedes DTO bekommt einen ***Versions-zähler***
- Datensatz ist **nicht am Server/im DBMS gesperrt**
- Andere Clients können lesen und schreiben
- Konflikte werden über Versionsvergleich erkannt

Implementierung des Versionszählers mit JPA

```
@Entity
   public class Project {
        @Id
        @GeneratedValue
        private Long id;
        private String name;

        @Version
        private Long version;
}
```

Versionszähler in den Requests

- Profiversion: Parameter im HTTP-Header
  - Etag
  - IF-MATCH
- Vereinfachte Version: Versionszähler im DTO