Enterprise JavaBeans im Praxistest

JavaForum Stuttgart
28. Juni 2000

Thomas Zink und Jörg Hettel

Inhalt

- Motivation
- Der TPC-W Benchmark
- Die J2EE Architektur
- Die Benchmark-Applikation
- Messaufbau und Messungen
- Resümee und Ausblick

Motivation

- J2EE als "Heilsversprechung" zur schnellen und komfortablen Entwicklung verteilter Systeme
- Untersuchungsgegenstand: **Enterprise JavaBeans**
  - EJB bilden vielversprechenden Ansatz für transaktionale Systeme
- Bisher nur Erfahrungen mit
  - Prototypen
    - kleine DBen
    - wenige Clients / geringe Concurrency
  - unkritische Anwendungen (Performance und Ausfallsicherheit spielen keine Rolle)

Motivation

- **Ziel:** Eigene Erfahrungen zu gewinnen, um auch quantitative Aussagen machen zu können:
  - *Wie teuer (performant) ist die Verwendung von EJBs?*
  - *Entwicklungsaufwand (Kosten-Nutzen Vergleich)?*
- **Rahmenbedingungen**
  - J2EE konforme Systemarchitektur
    - Java als Programmiersprache
  - Möglichst realistische Testanwendung
    - "große" DBen
    - viele Clients, hohe Last und viel Concurrency

Untersuchungsprojekt

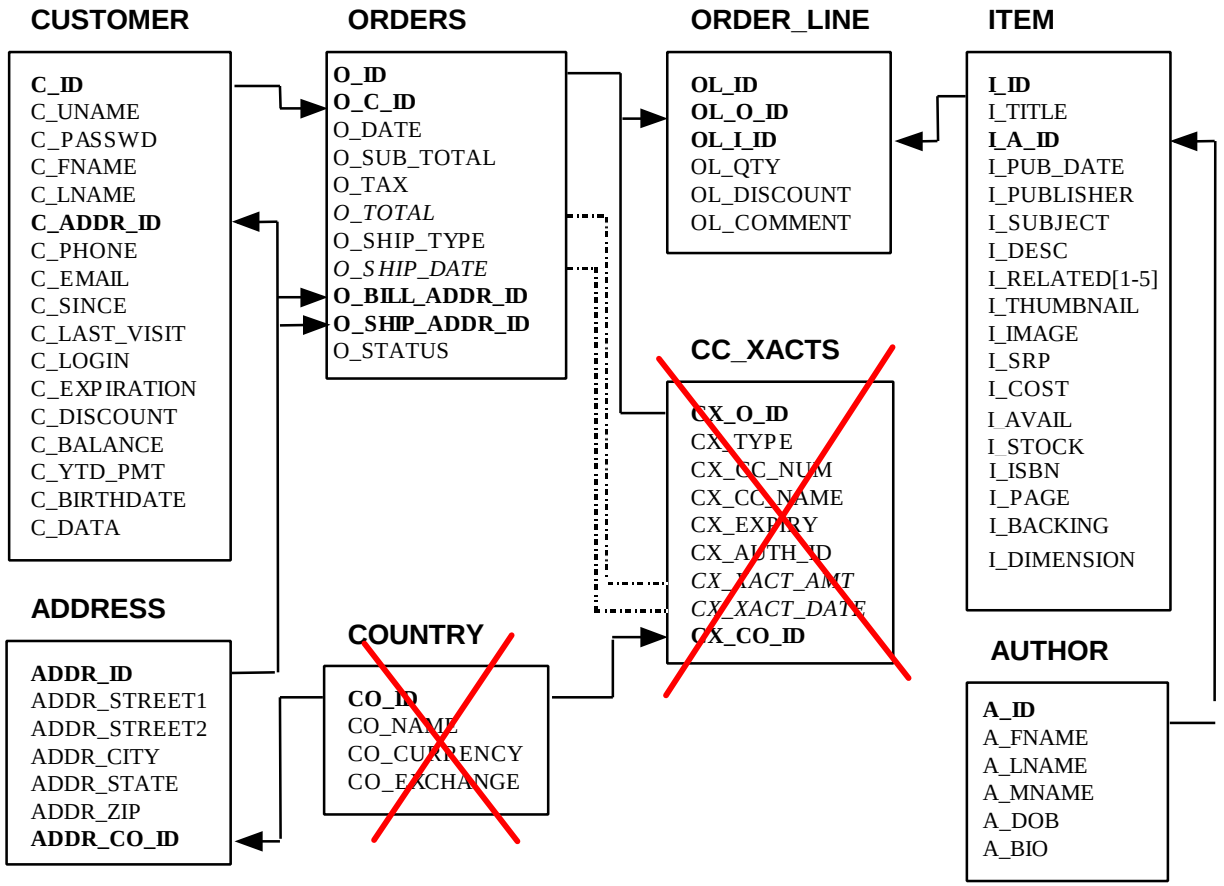
- **Idee:** Realisierung eines Ausschnittes aus dem TPC-W Benchmarks
  - Standardisierte Anwendung
  - exakte Vorgaben
  - Zwei Varianten
    - Applikation auf Basis von EJBs (container managed persistence)
    - Applikation mit herkömmlicher JDBC-Architektur

TPC-W Benchmark

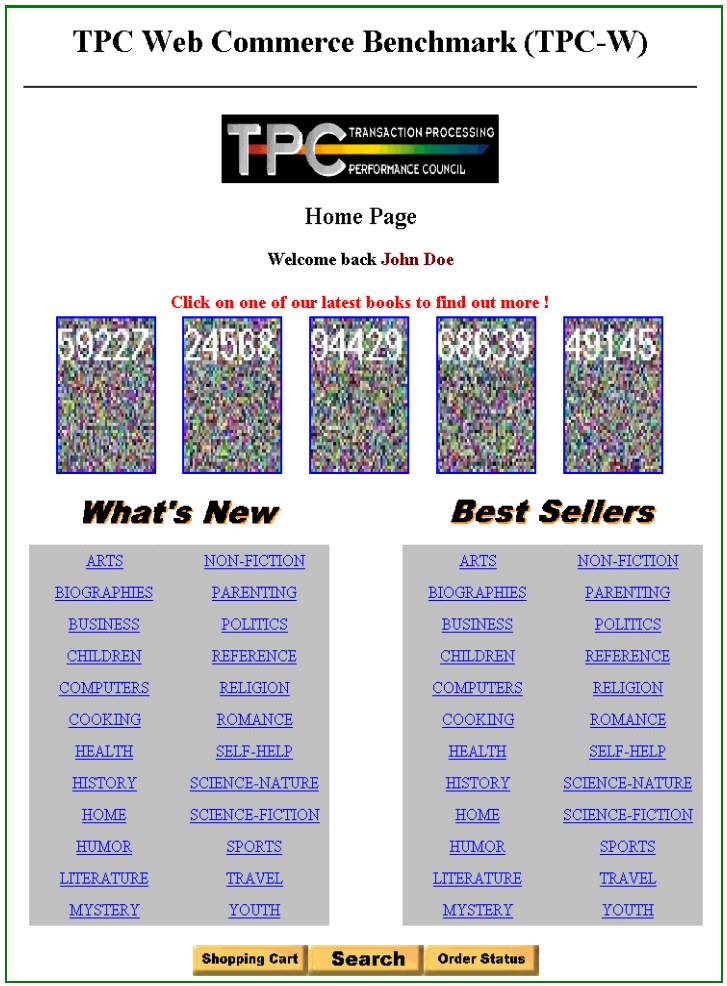
- Repräsentiert Web-Business Applikation bestehend aus
  - Browser (Client)
  - Web-Server
  - Applikations-Server
  - Datenbank
- Business-Modell: Web-Buchhandlung

TPC-W Benchmark

- Datenmodell (vereinfacht für Benchmark-Zwecke)

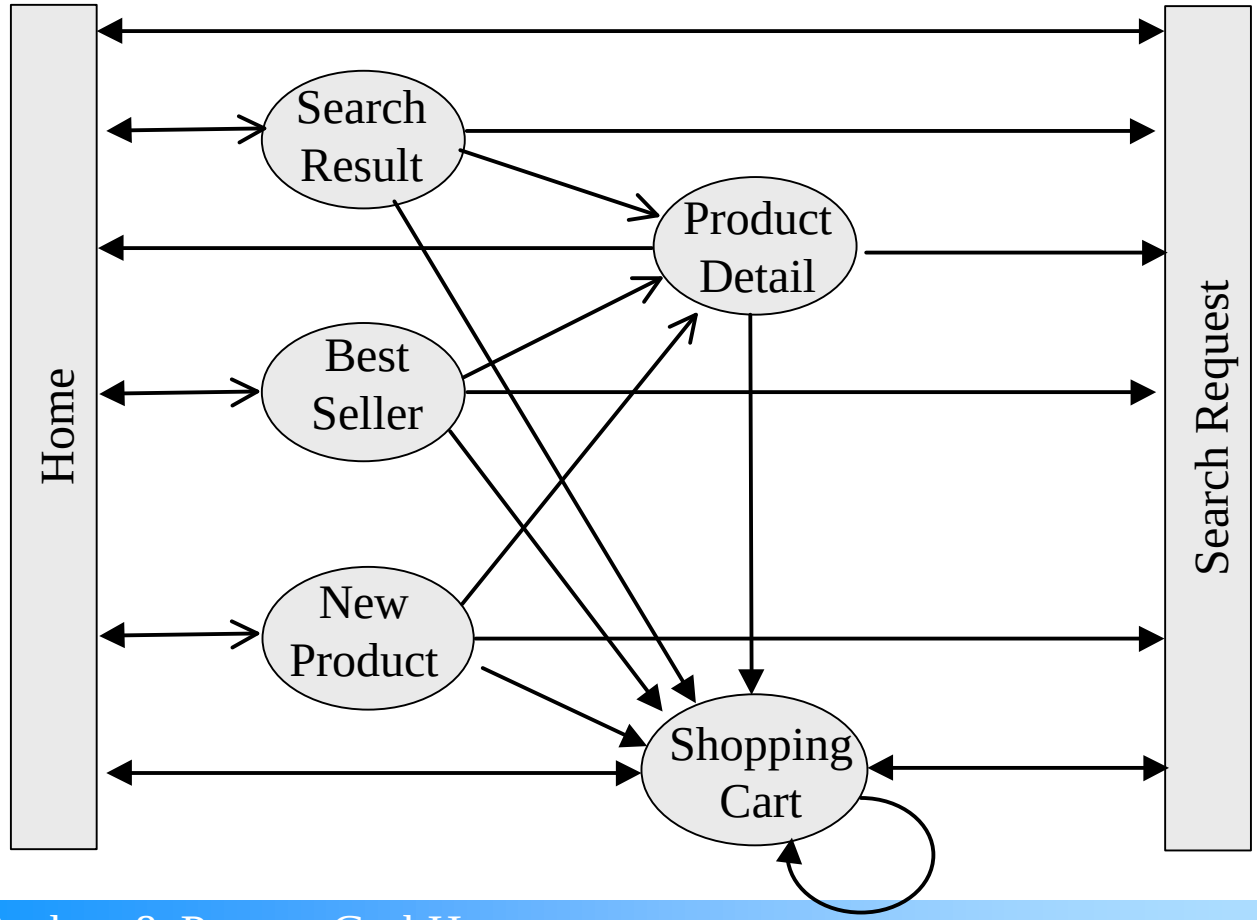


TPC-W Benchmark



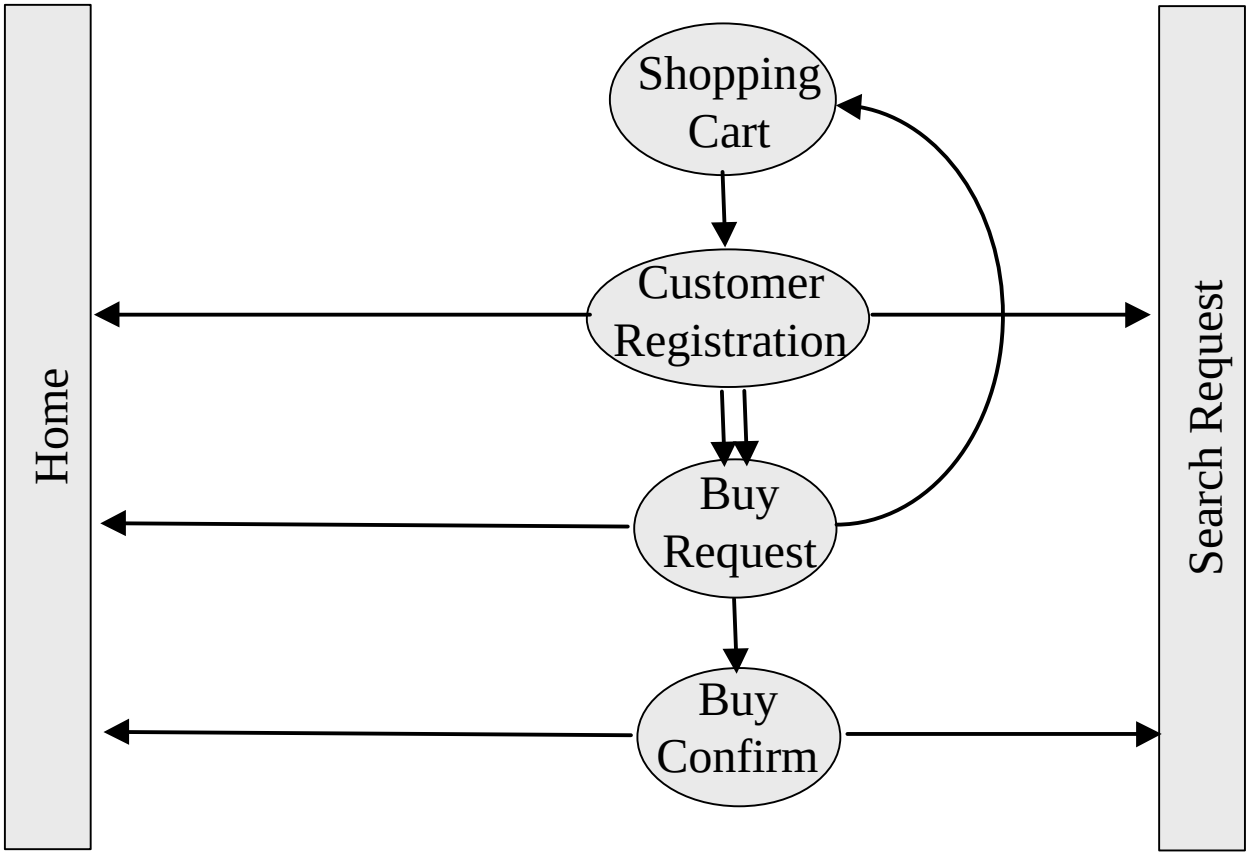
TPC-W Benchmark

- Browsing-Möglichkeiten (I)

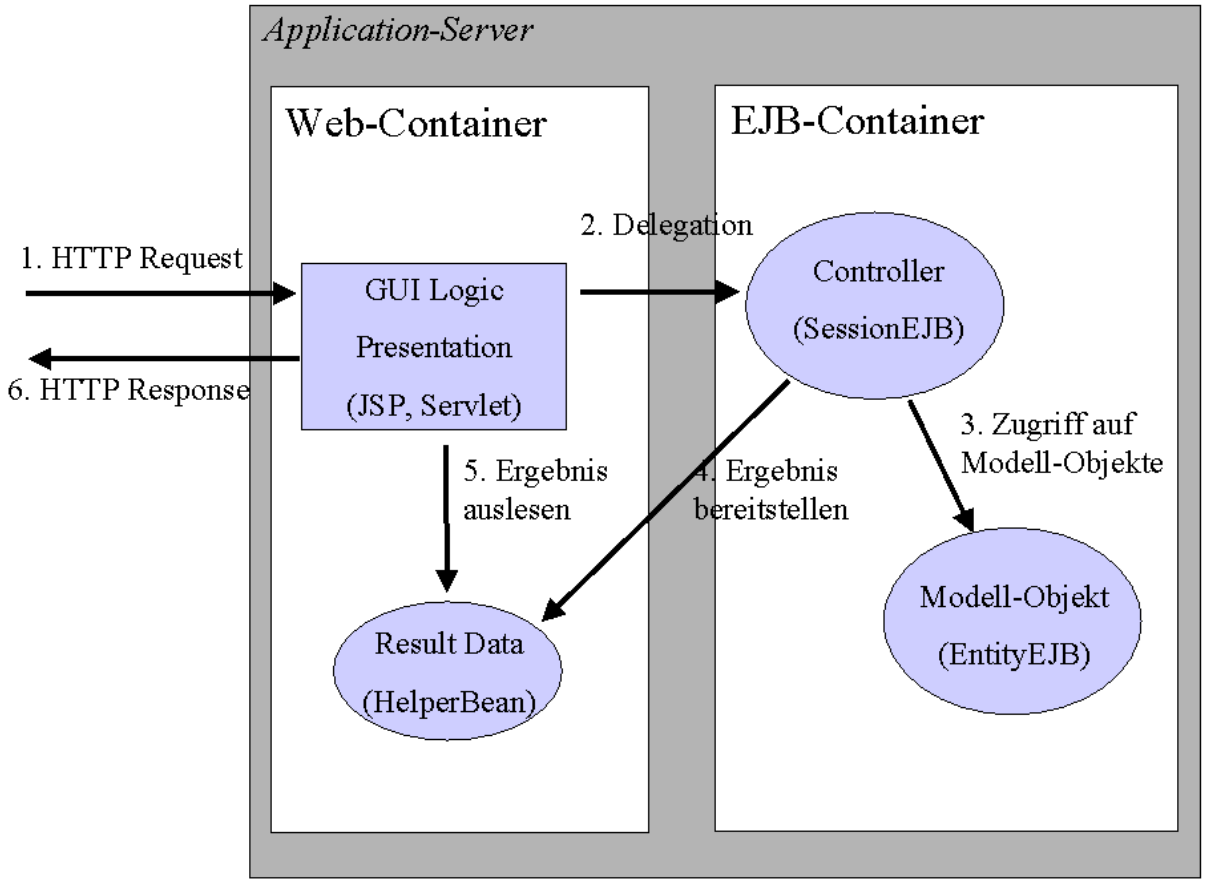


TPC-W Benchmark

- Browsing-Möglichkeiten (II)



Programmiermodell mit EJBs

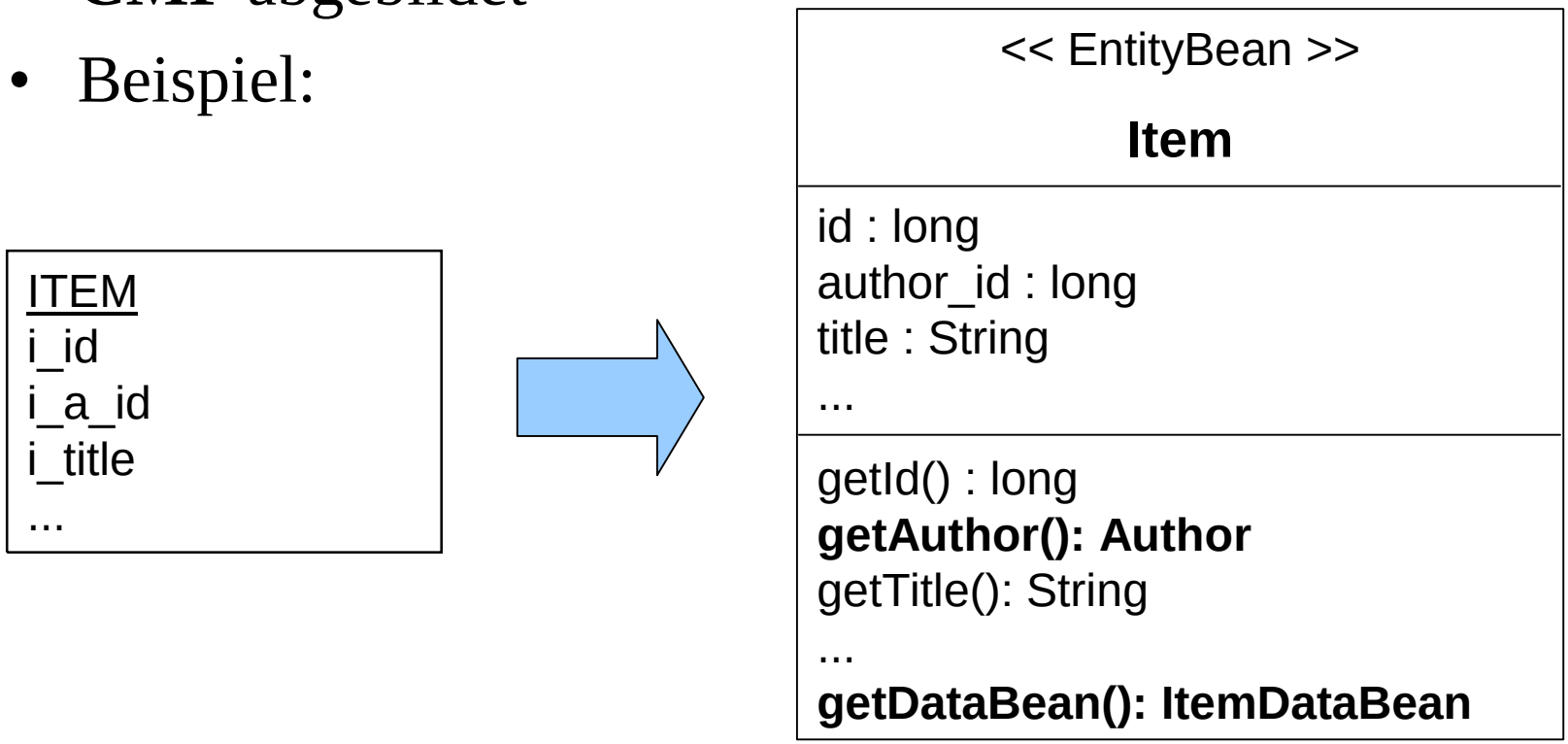


Bechmark-Implementierung (EJB)

- Auf Basis der J2EE-Architektur und des J2EE-Programmiermodelles, d.h.
  - Business Objects = Entity Beans (6)
  - Business Logic = Session Beans (4)
  - GUI-Logic = Servlets (7)
  - Presentation = JSP (9)
- Wiederverwendung, Einfachheit, Lesbarkeit
- Umfang der Anwendung:
  - ca. 2 kLOC, ca. 20 PT Entwicklungzeit

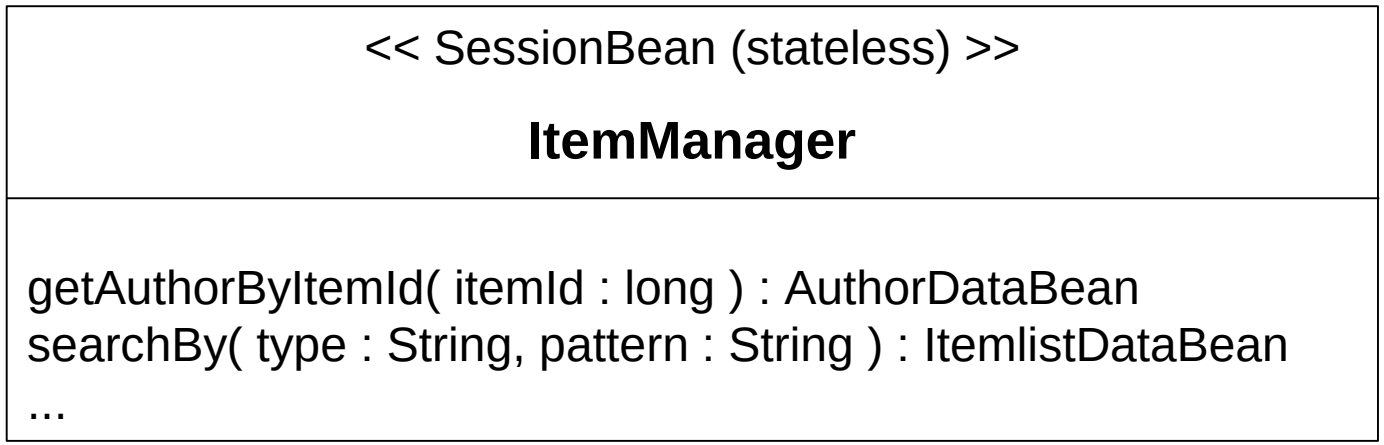
Business Objects

- „Tabellen“ der DB wurden als **Entitiy-Beans mit CMP** abgebildet
- Beispiel:



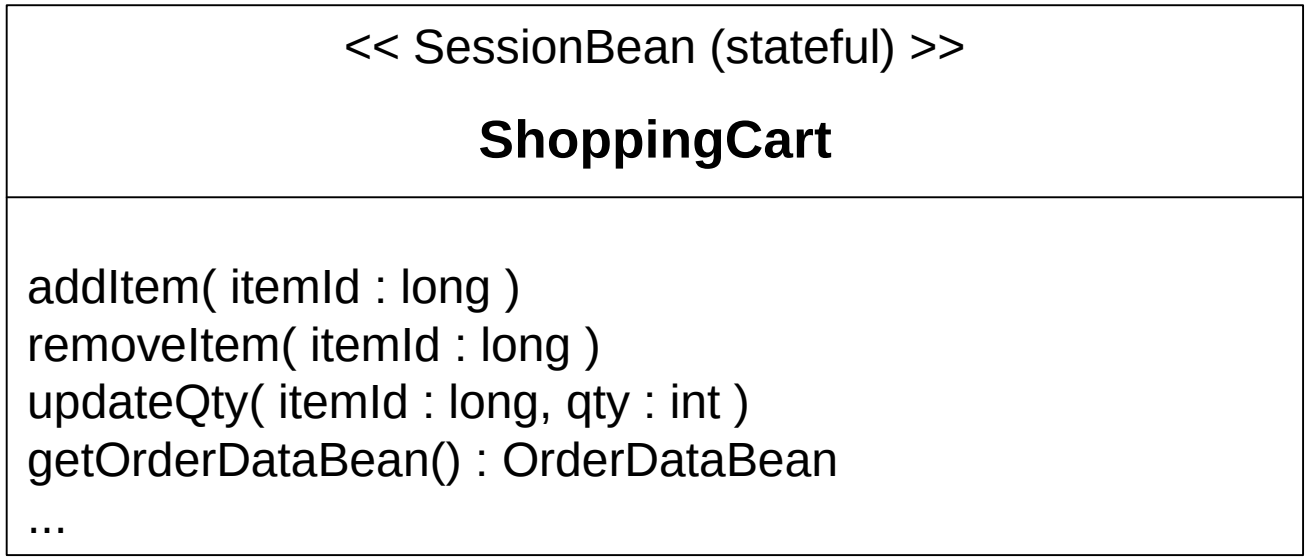
Business Logic

- Ziel: Kapselung der Anwendungslogik in **SessionBeans**
- Beispiel: Suche nach Items



Exkurs: Session Management

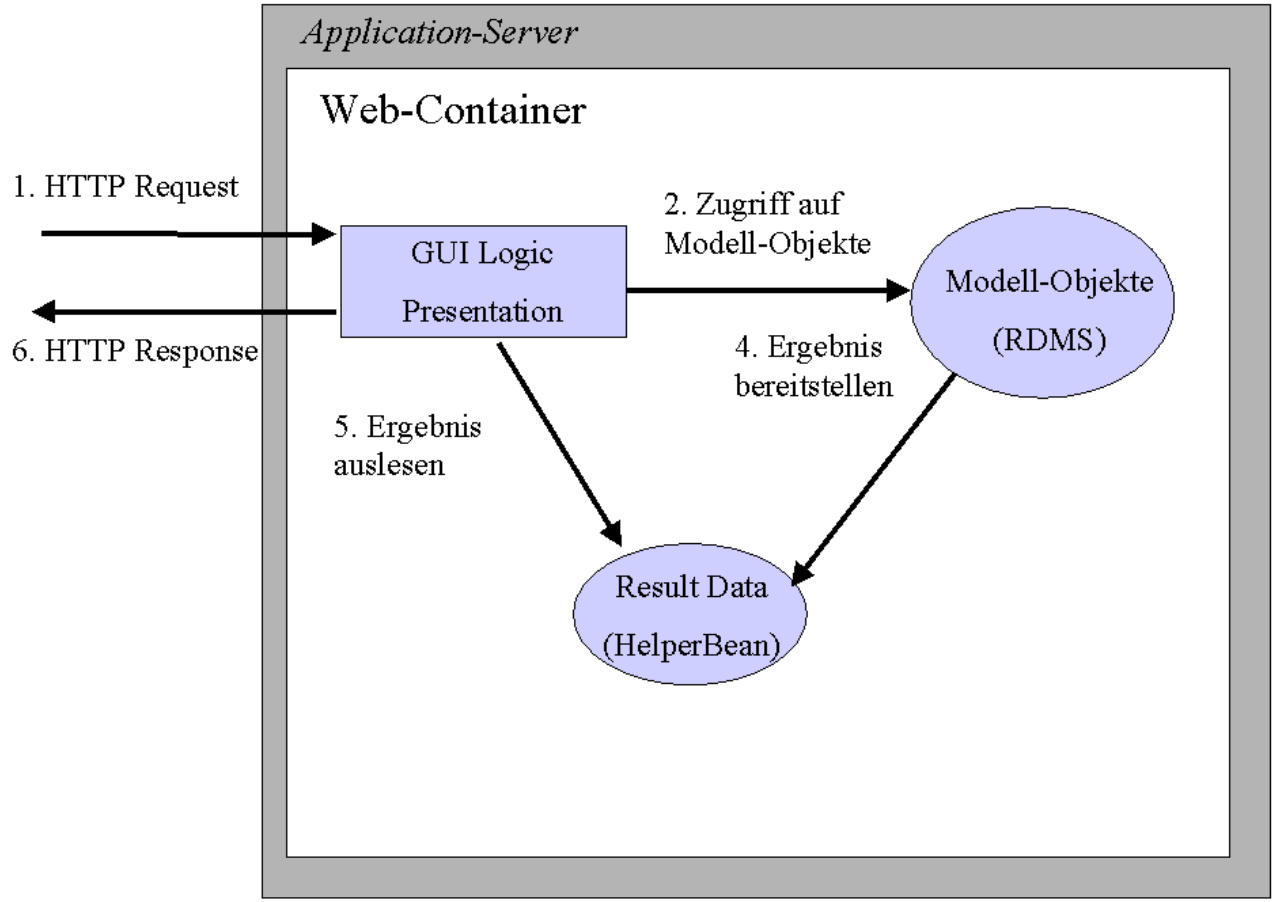
- Ziel: eigentliches Session-Management in EJB-Schicht realisieren
- Prinzip: Assoziation einer HTTP-Session (Servlets/JSP) mit einem **Stateful SessionBean**



Präsentations-Logik

- Prinzip: **Servlets**
  - erhalten HTTP-Requests
  - verarbeiten die Parameter
  - greifen auf die Geschäftslogik der SessionBeans zurück
  - wählen eine *Java Server Page* aus
  - übergeben dieser Seite ggf. Daten per *Beans*
- Prinzip: **Java Server Pages (JSP)**
  - erhalten Daten per Beans
  - geben die Bean-Daten aus (erzeugen HTML)

Programmiermodell mit JDBC-Zugriff



Architektur der JDBC-Anwendung

- prinzipielle Architektur beibehalten
- aber:
  - keine EJBs, statt dessen:
    Datenbankzugriff per JDBC
  - Bessere Kontrolle über den DB Zugriff
    - „prefetching“ bei Assoziationen
  - „Pooling“ der DB-Verbindungen
  - Implementierung:
    Interfaces der SessionBeans mit „normalen“ Java-Klassen nachprogrammiert

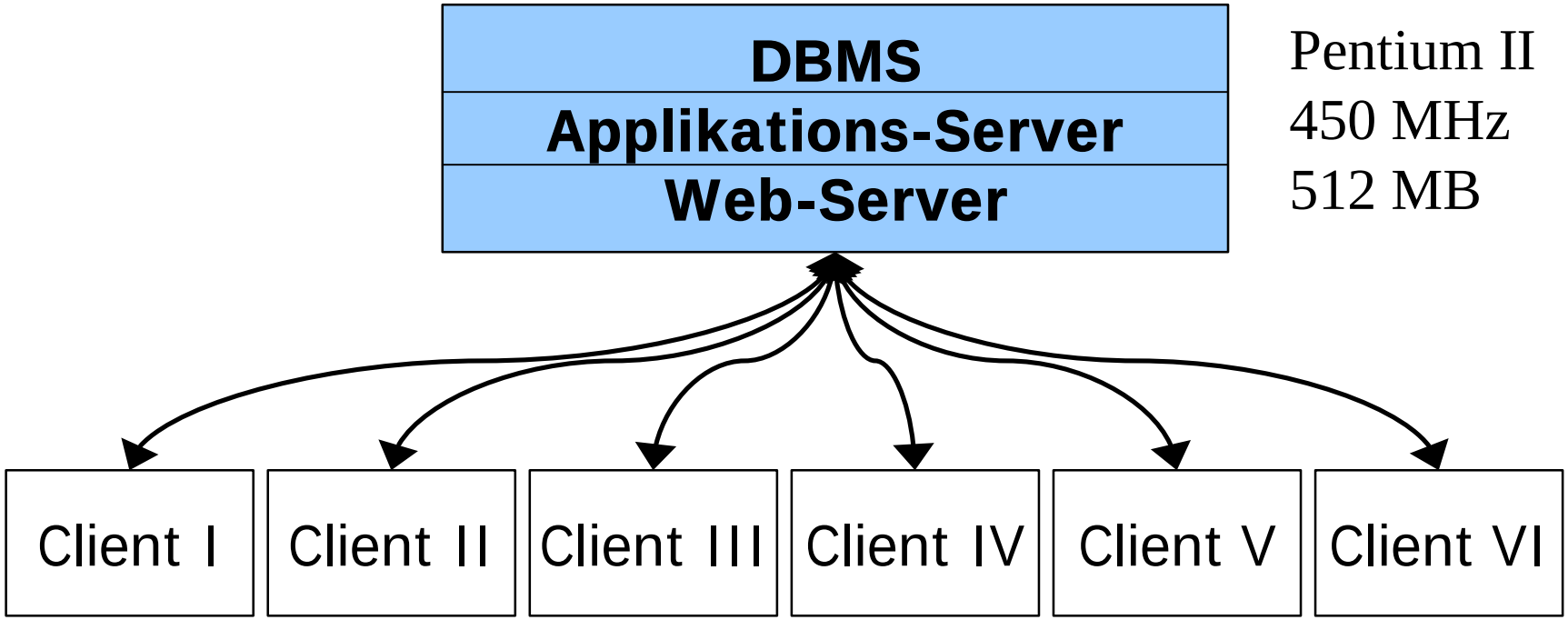
Messdaten-Erhebung

- Prinzip:
  - Benutzer-Interaktionen werden simuliert: „Emulated Browser“ (EB) setzt POST/GET-Requests ab
    - Simuliert fiktive Einkaufstour nach vorgegebenem User-Profil (festgelegte Übergangswahrscheinlichkeiten)
  - EB misst „Web Interaction Response Time“ (WIRT) und Zeitpunkt

Messdaten-Erhebung

- durch TPC-W vorgegeben:
  - mögliche Interaktionen pro Seite
  - möglicher Parameter und deren Erzeugung
  - statistische Häufigkeitsverteilung der Interaktionen (verschieden Szenarien)
  - „ThinkTime“, minimale Dauer einer Session, ...

Mess-Szenario



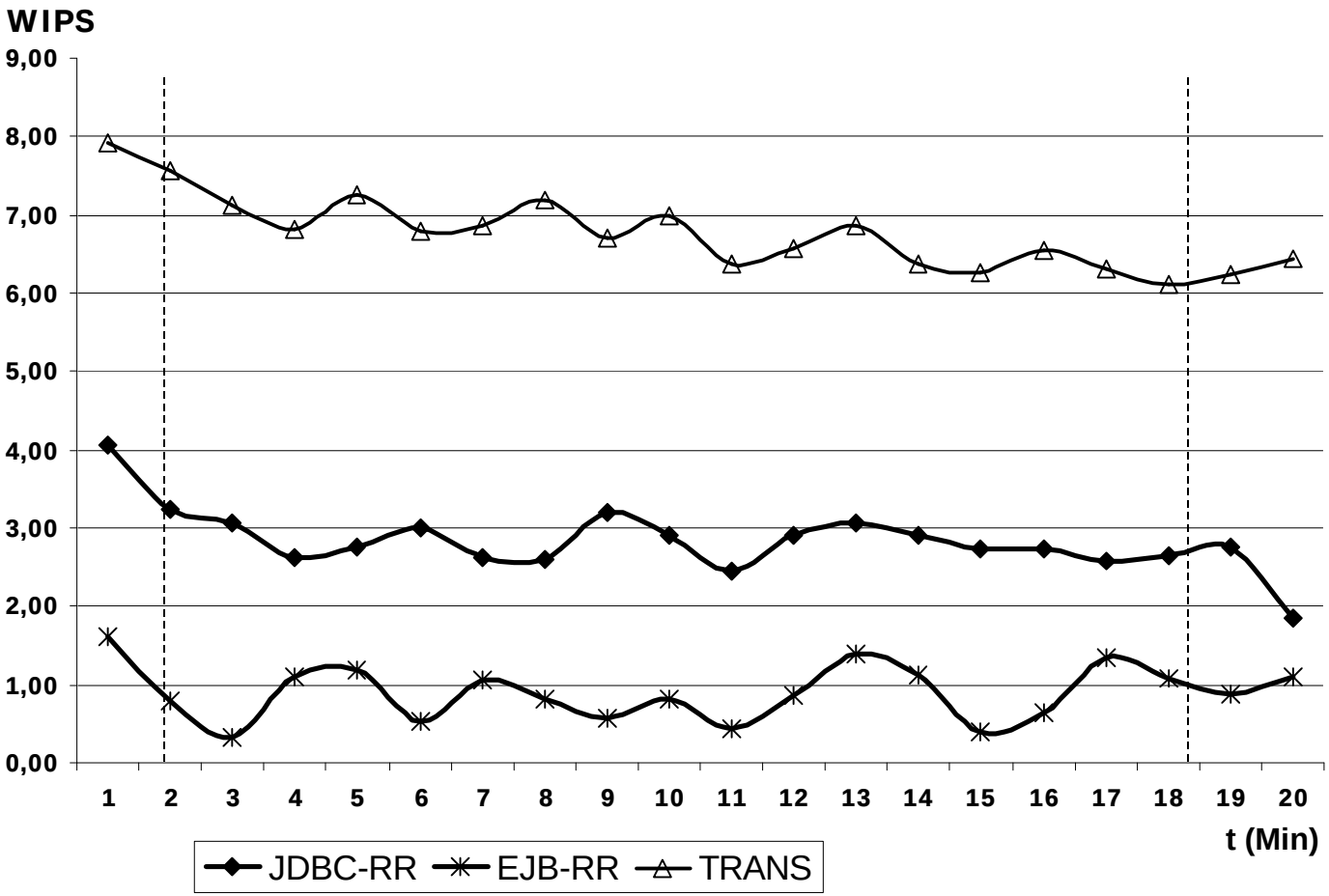
jeweils 10 Threads pro Client

✍✍✍ 60 Emulated Browsers (ThinkTime: 7sec)

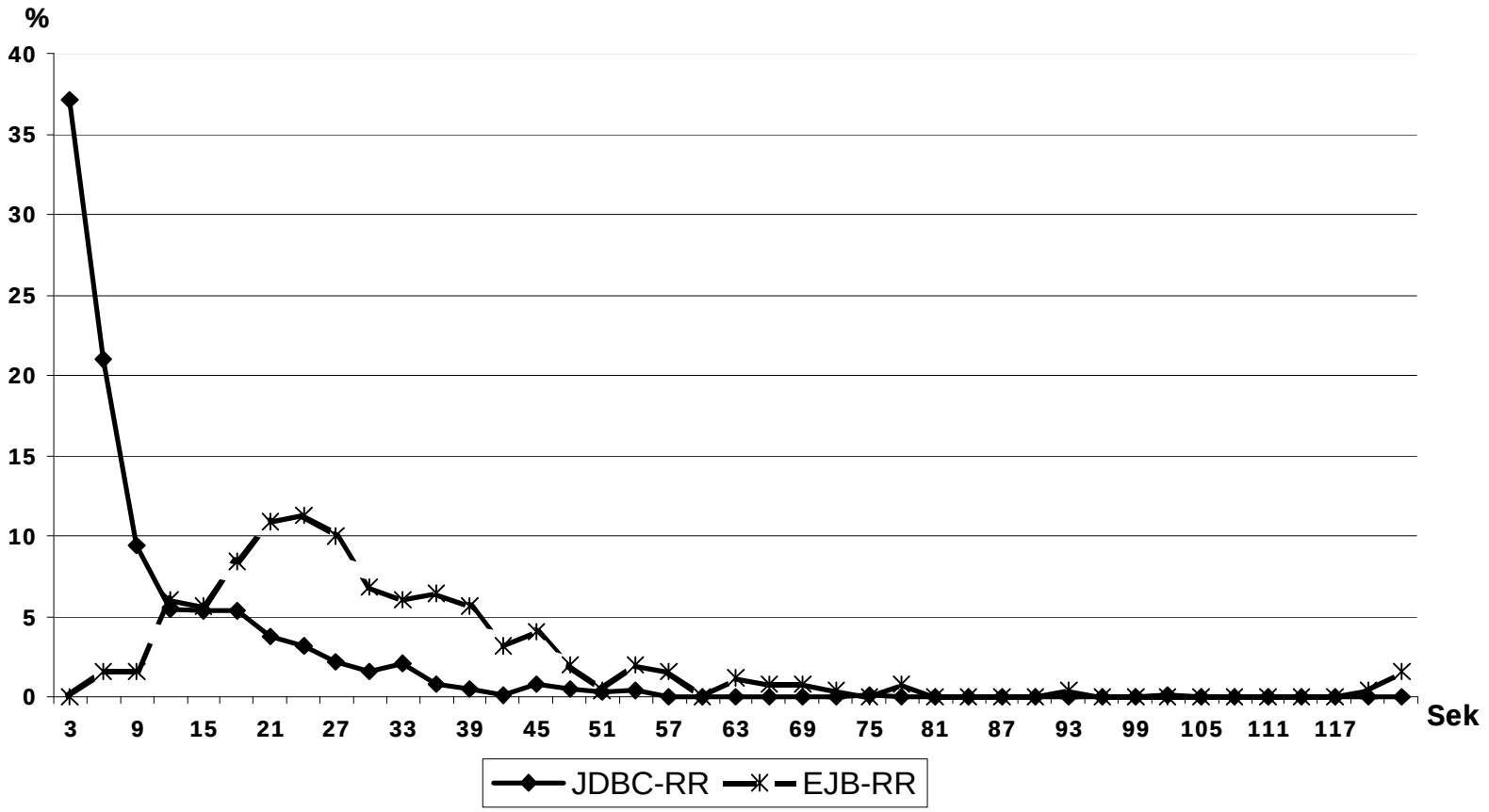
Parameter

- Datenbankfüllgrad
  - Items: 10.000, Authors: 2.500, Customers: 5.000, ...
- Messdauer:
  - jeweils 20 Minuten
  - davon: 15 Minuten analysiert
- Varianten:
  - EJB-Architektur: Repeatable Read, Repeatable Read mit Const-Flag
  - JDBC-Architektur: Repeatable Read
  - Transiente Datenhaltung

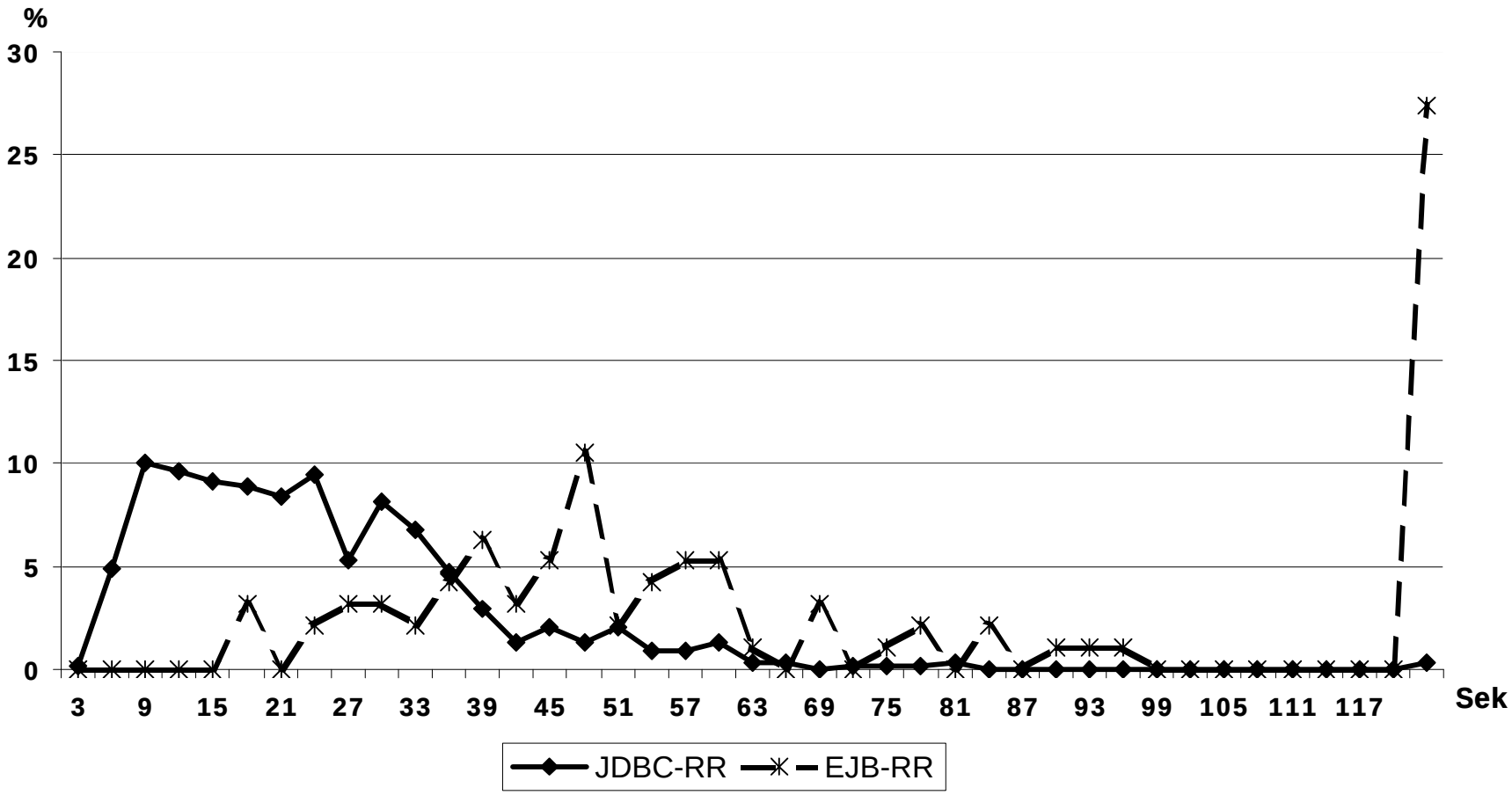
Web Interactions per Second (WIPS)



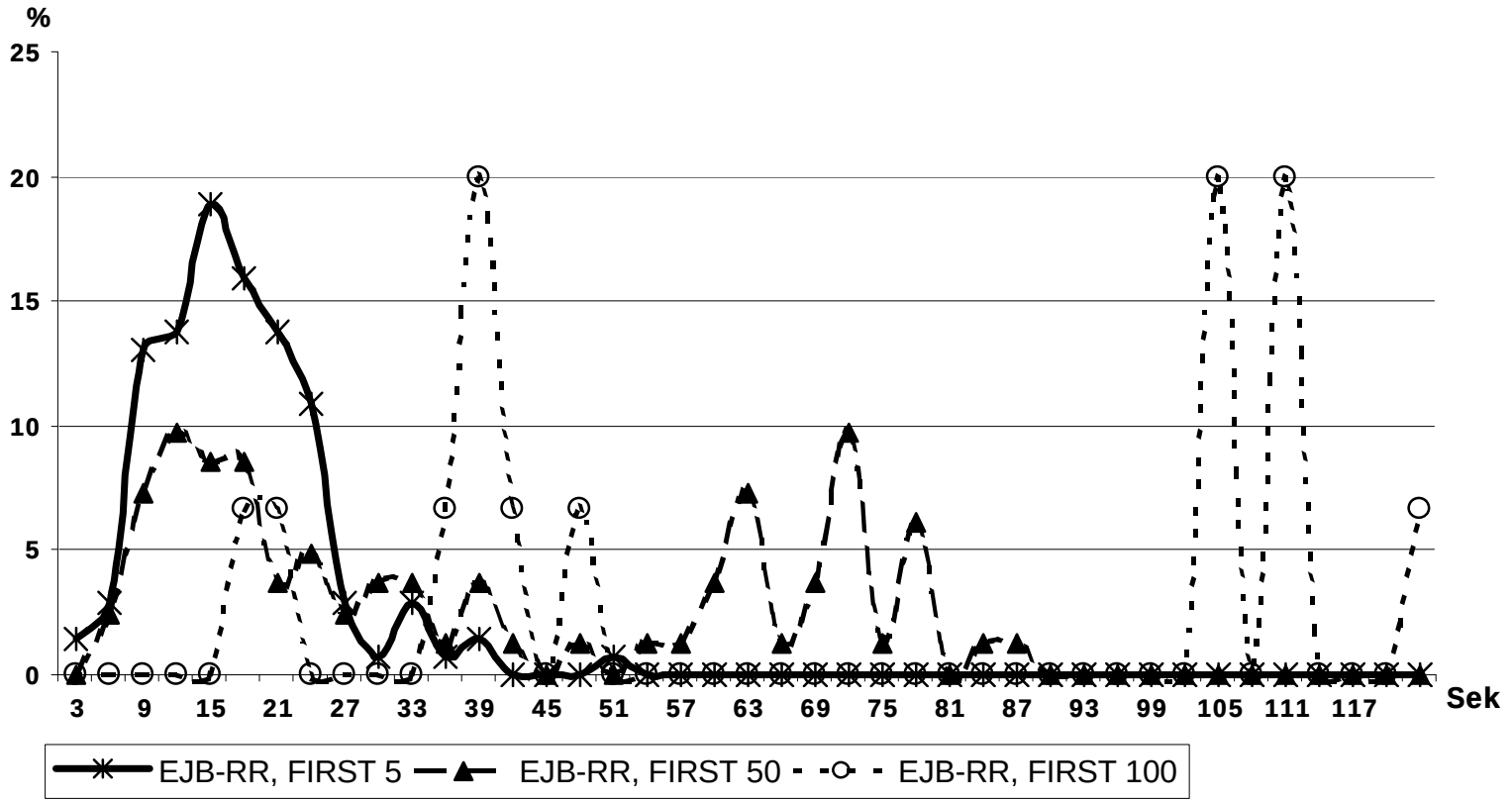
WIRT-Verteilung für Home



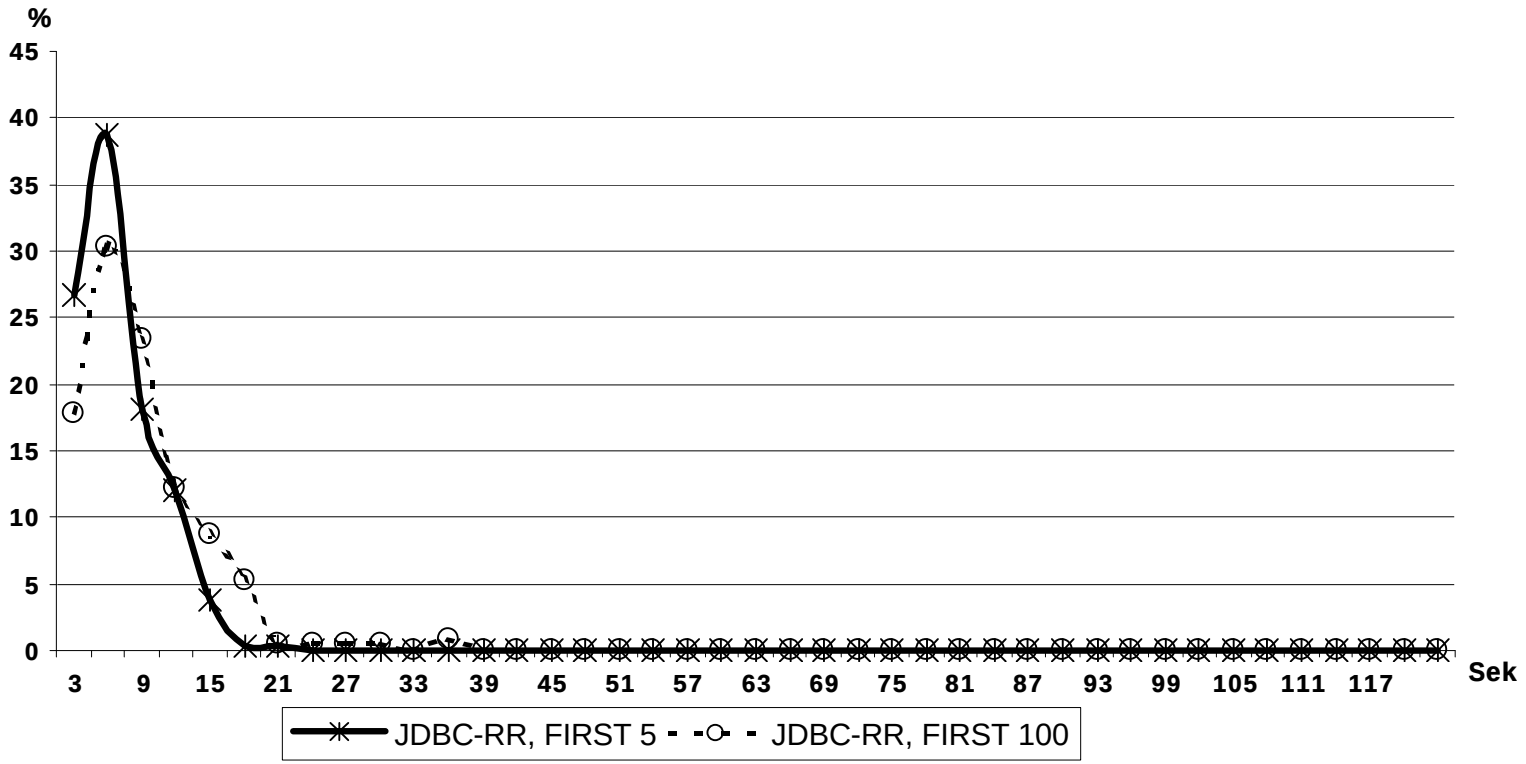
WIRT-Verteilung für SearchResult



SearchResult mit variierendem Fetch First (EJB-Variante)



SearchResult mit variierendem Fetch First (JDBC-Variante)

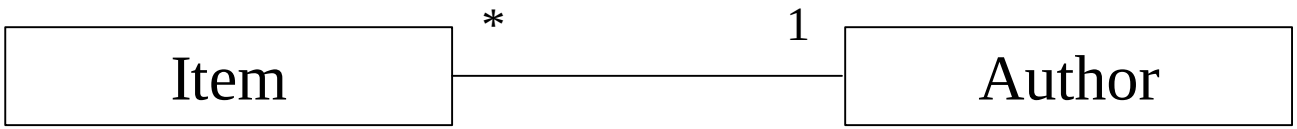


Bemerkungen

- Ein "Home-Page" Zugriff entspricht 5 DB-Zugriffe
- Beim "SearchResult" werden bis zu 50 Assoziationen aufgelöst
  - d.h. bei EJB-Variante 50 findByPrimaryKey() Aufrufe
  - bei JDBC-Variante wurde JOIN verwendet

Auflösen von Assoziationen (EJB)

- Assoziationen können nicht performant abgebildert werden



  - Beispiel: ItemHome::findBySubject(...)

```
public Enumeration findBySubject(....) throws FinderException, ... {
        // liefert Enumeration über Item-Objekte
}
```

Auflösen von Assoziationen (EJB)

- Zugriff auf EJBs immer über Home-Lookup
  - Beispiel: Item::getAuthor()

```
public Author getAuthor() throws RemoteException {
        ....
        InitialContext  initCtxt = ....

        Object tmpObj = initCtxt.lookup( "Author" );
        authorHome = (AuthorHome)
                javax.rmi.PortableRemoteObject.narrow(
                        tmpObj, AuthorHome.class );

        author = authorHome.findByPrimaryKey( ... );
        ....
        return author;
}
```

Auflösen von Assoziationen (JDBC)

- Auflösen der Assoziation über JOIN

```
public Enumeration findBySubject(....) {
      ....
      String queryString =
            "SELECT * FROM ITEM I, AUTHOR A
            WHERE (A.A_LNAME LIKE ?) AND (I.I_A_ID = A.A_ID)
            FETCH FIRST 50";

     java.sql.PreparedStatement pstmt =
                                    con.prepareStatement(queryString);
      .....
      // Auslesen des ResultSets und notwendige DataBeans
      // erzeugen
     ......
}
```

Mess-Ergebnisse

- Resultate beziehen sich auf container managed persistence
- Faktor 3 bei der WIRT zwischen JDBC und EJB-Variante liegt an der großen Differenz bei Seiten wie "SearchResult"
- Im Einzelfall teilweise extrem lange Antwortzeiten und hoher Durchschnitt
- Auflösen von Assoziationen teuer

Fazit

- Schnelle Entwicklung (ca. 20 Personentage)
- An Technologie angepasstes Design liefert akzeptable Performance
  - z.B. Listenzugriffe in SessionBeans über JDBC
- Durch TPC-W wird nur ein Teil der Verteilungsaspekte der EJB-Architektur abgedeckt