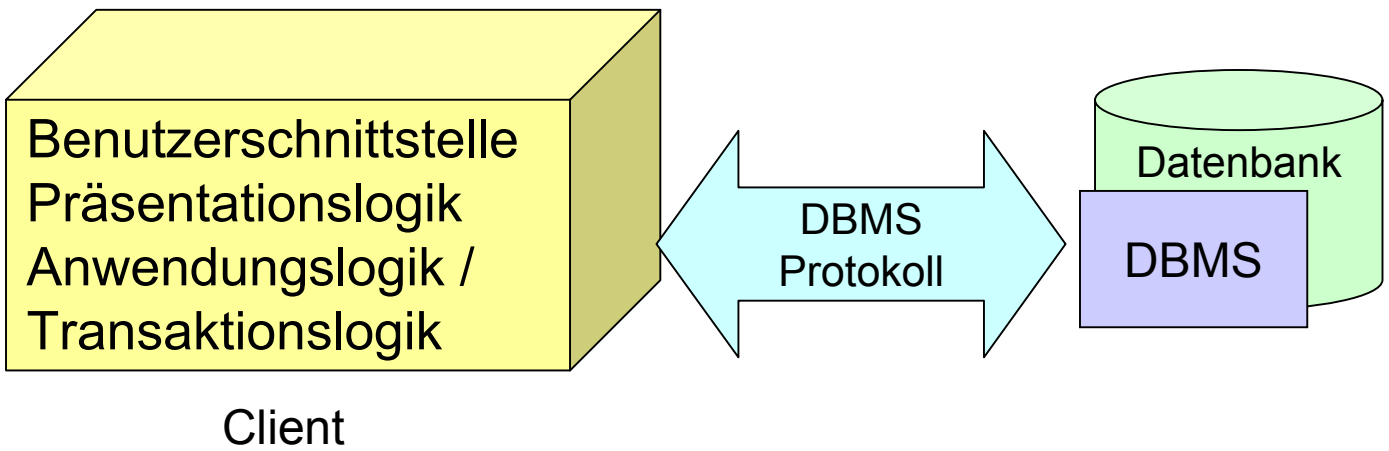
Einsatz von Applikationsservern

Untersucht am Beispiel des Sybase „Enterprise Application Server“

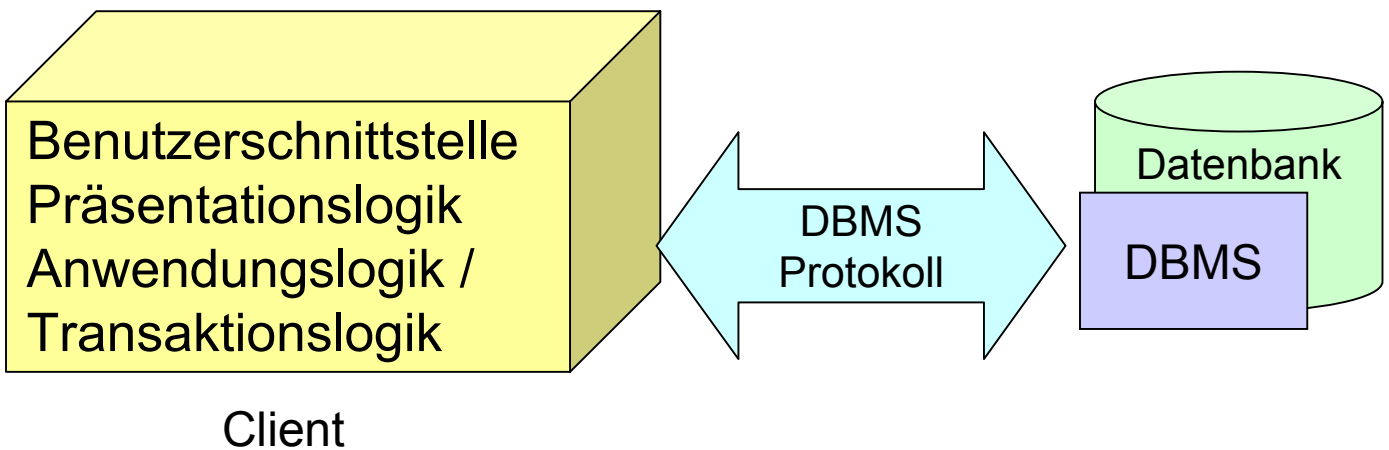
Architektur von Datenbanksystemen

- Client / Server Modell (2 Schichten Modell)



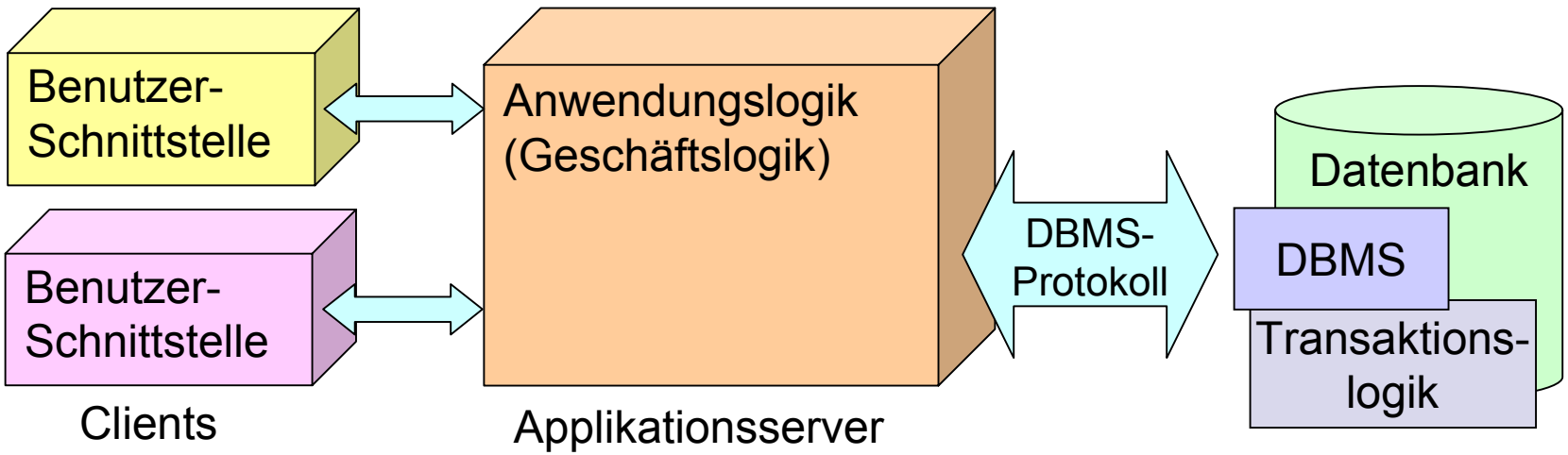
Client / Server Modell

- Nachteile:
  - Geringe Leistung / hoher Datenverkehr
  - Schwierige Erstellung / Erweiterung von Anwendungen



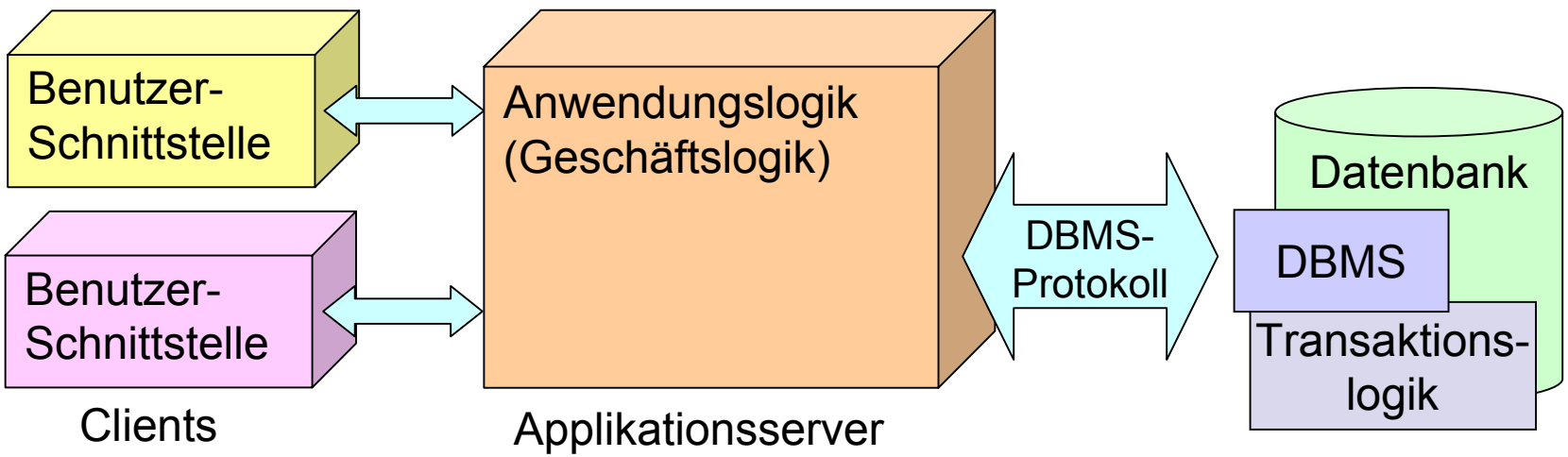
Architektur von Datenbanksystemen

- Das 3-Schichten-Modell



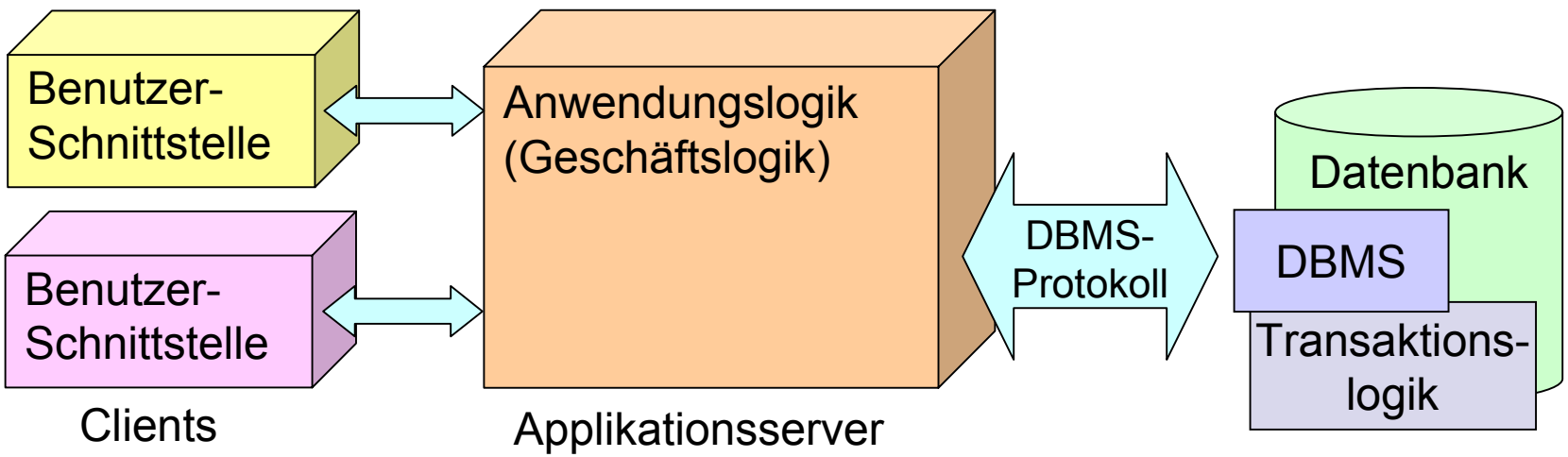
Das 3-Schichten-Modell

- Vorteile:
  - Einfache Erstellung / Erweiterung der Anwendungslogik
  - Leistung des Client ist nicht entscheidend (Thin-Clients)
  - Client benötigt keinen direkten Datenzugriff (Sicherheit)
  - Verschiedene Arten von Clients (Java, C++)
  - Steigerung der Performance und Sicherheit



Das 3-Schichten-Modell

- Nachteile:
  - Hoher Kommunikationsaufwand
  - Hoher Entwicklungsaufwand



Implementierungstechniken

- Client / Server
  - JDBC / ODBC
    - Kommunikation mit dem DBMS
  - SQL
    - Kommunikationssprache
    - Befehle zum Erstellen, Auslesen und Manipulieren
  - Stored Procedures
    - SQL mit Kontrollstrukturen und Variablen
    - Über Transaktionslogik auf den DB-Server

Implementierungstechniken

- Applikationsserver
  - Java
    - Objektorientiert
    - Wird compiliert und interpretiert (Virtual Maschine)
    - Plattformunabhängig

Implementierungstechniken

- Applikationsserver
  - Java 2 Enterprise Edition Version 1.3
    - Standard-Architektur zur Definition und Unterstützung von mehrschichtigen Programmmodellen
    - Grundlage aller auf Java basierenden Applikationsserver
    - Großer Funktionsumfang
      - Kommunikation mit Komponenten, Datenbanken und Back-End Systemen
      - Authentifizierungs- und Sicherheitsservice
      - Transaction Management
      - Mail und XML Service
      - Etc…

Implementierungstechniken

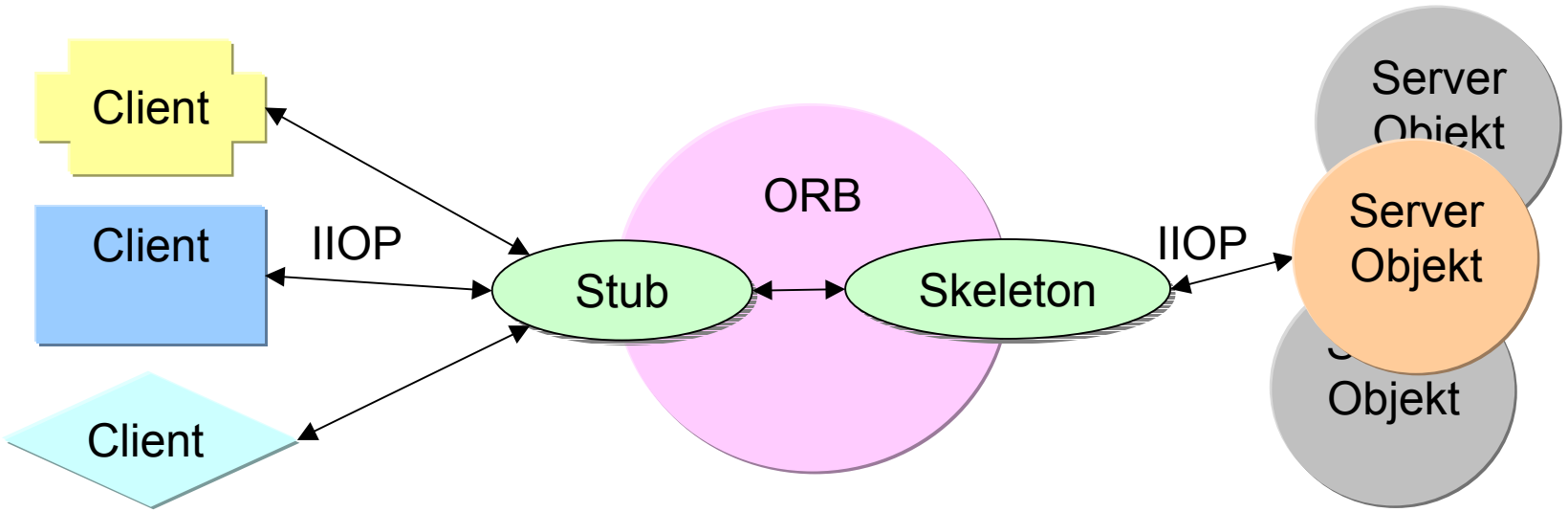
- Applikationsserver
  - Enterprise Java Beans (EJB)
    - Softwarekomponentenmodell
    - Komponenten werden auf den Server ausgeführt
    - Kommunikation ist kompatibel zum CORBA Standard

Implementierungstechniken

- Applikationsserver
  - CORBA
    - Common Object Request Broker Architecture
    - Stellt einzelne Objekte über ein Netzwerk zur Verfügung
    - Ermöglicht die Kommunikation von Applikationen, egal wo sie im Netz liegen oder womit sie entwickelt worden sind

Implementierungstechniken

- Applikationsserver
  - CORBA Architektur



Applikationsserver Systeme

- Nicht auf Java basierend
  - Serverseitige Scriptsprachen wie PHP und ASP
  - CGI Sprachen (Common Gateway Interface)
  - COM / DCOM bzw. Microsoft .NET

Applikationsserver Systeme

- Auf Java basierend
  - Java 2 Enterprise Edition als Grundlage
  - Teilweise oder vollständige Kompatibilität zu verschiedenen Versionen der J2EE
  - Eigene Zusatzprodukte und Funktionen

Applikationsserver Systeme

- Oracle 9iAs
  - Volle J2EE V1.3 Unterstützung
  - Unterstützt viele Sprachen (z.B. Java, Perl, C, Cobol und PL/SQL)
  - Unterstützt eine Vielzahl von Standards (z.B. J2EE1.3, Web Services, WebDAV, LDAP v3, SSL v3 und einige XML-Standards)

Applikationsserver Systeme

- BEA Weblogic
  - Gute Java Unterstützung (J2EE 1.3)
  - Mit „Tuxedo“ gutes Transaktionsmanagement
  - Unterstützt CORBA, COM und Web-Services (COM nur über Java-Wrapper)
  - Schnelle EJB-Unterstützung (mit Load Balancing und Failover)
  - Keine eigenen Entwicklungstools (Orientiert sich an Symantecs „Visual Cafe Enterprise“)

Applikationsserver Systeme

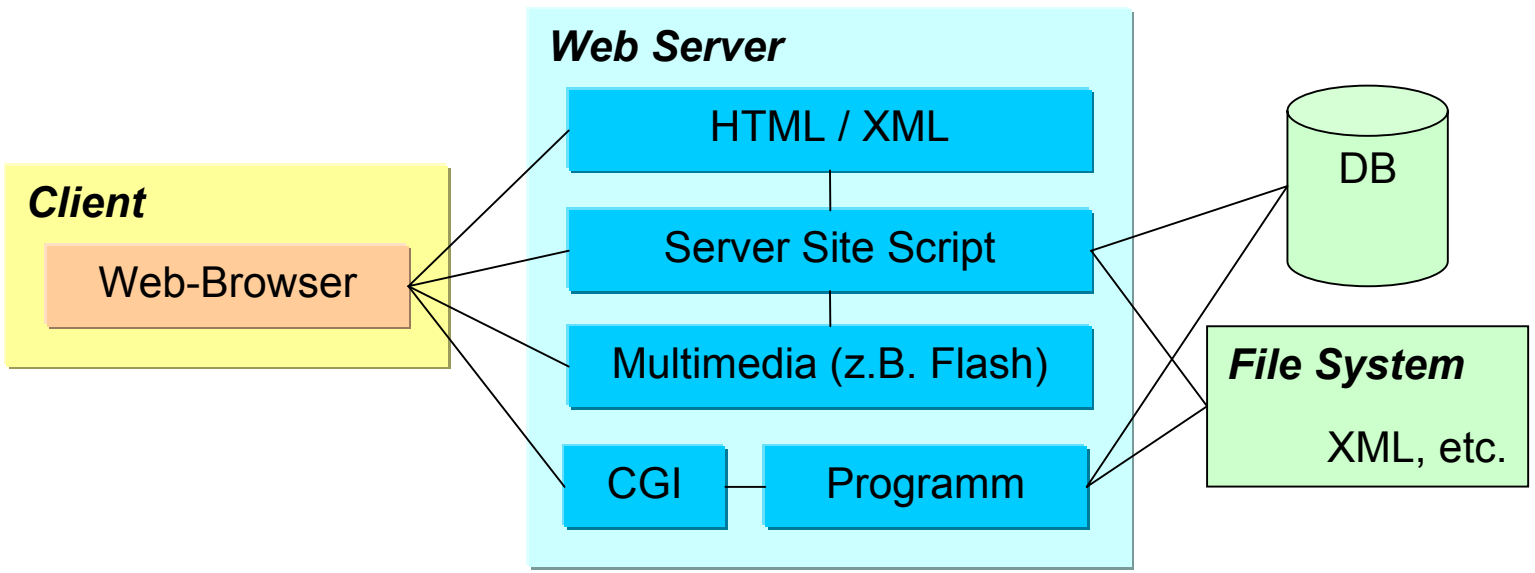
- IBM WebSphere Version 4.0
  - J2EE V1.2.1 mit einigen Features der Version 1.3 (z.B. JCA, JMS)
  - Basiert auf der Servlet-Engine
  - Volle Unterstützung von Web-Services
  - Native Unterstützung vieler Datenbanken

Applikationsserver Systeme

- Sybase Enterprise Application Server 4.1
  - Kompatibel zum J2EE V1.3 Standard
  - Unterstützt Komponenten in vielen Sprachen (Java und DCOM)
  - Eigene, serverseitige Scriptsprache PowerDynamo
  - Unterstützung von Web-Services
  - RAD Integration (PowerBuilder)

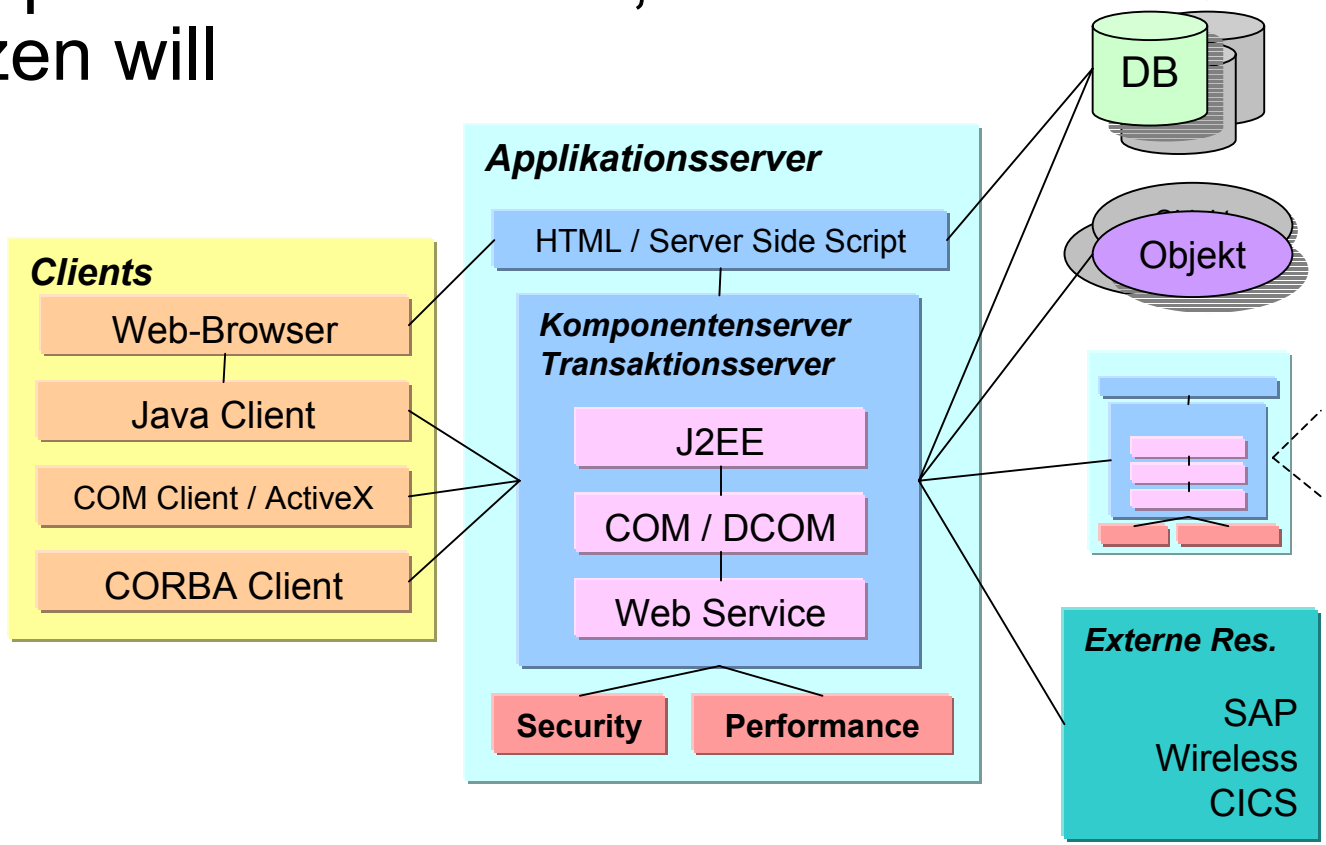
Fazit: Wer braucht welchen Server?

- Kleine, abgeschlossene Web-Anwendungen lassen sich mit einer serverseitigen Scriptsprache bzw. mit dem CGI gut umsetzten



Fazit: Wer braucht welchen Server?

- J2EE bzw. MS.NET lohnt sich nur, wenn man komponentenorientiert programmiert und diese Komponenten in vielen, verschiedenen Clients nutzen will

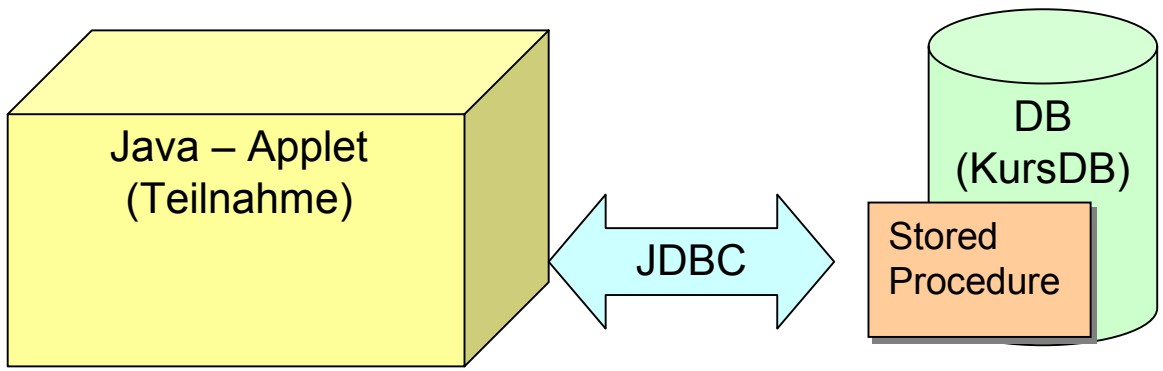


Beispielaufgabe

- Für einen Anbieter von Aufbau- und Lernkursen existiert eine Datenbank zur Speicherung der Kurse, Lehrer, Teilnehmer und Material.
- Die Teilnahme an einem Kurs soll über ein Java-Applet möglich sein, welches über das Internet aufgerufen wird.
- Lösung als:
  - Client / Server Anwendung mit Hilfe der JDBC
  - CORBA Anwendung mit Zugriff auf den Komponentenserver „Jaguar“

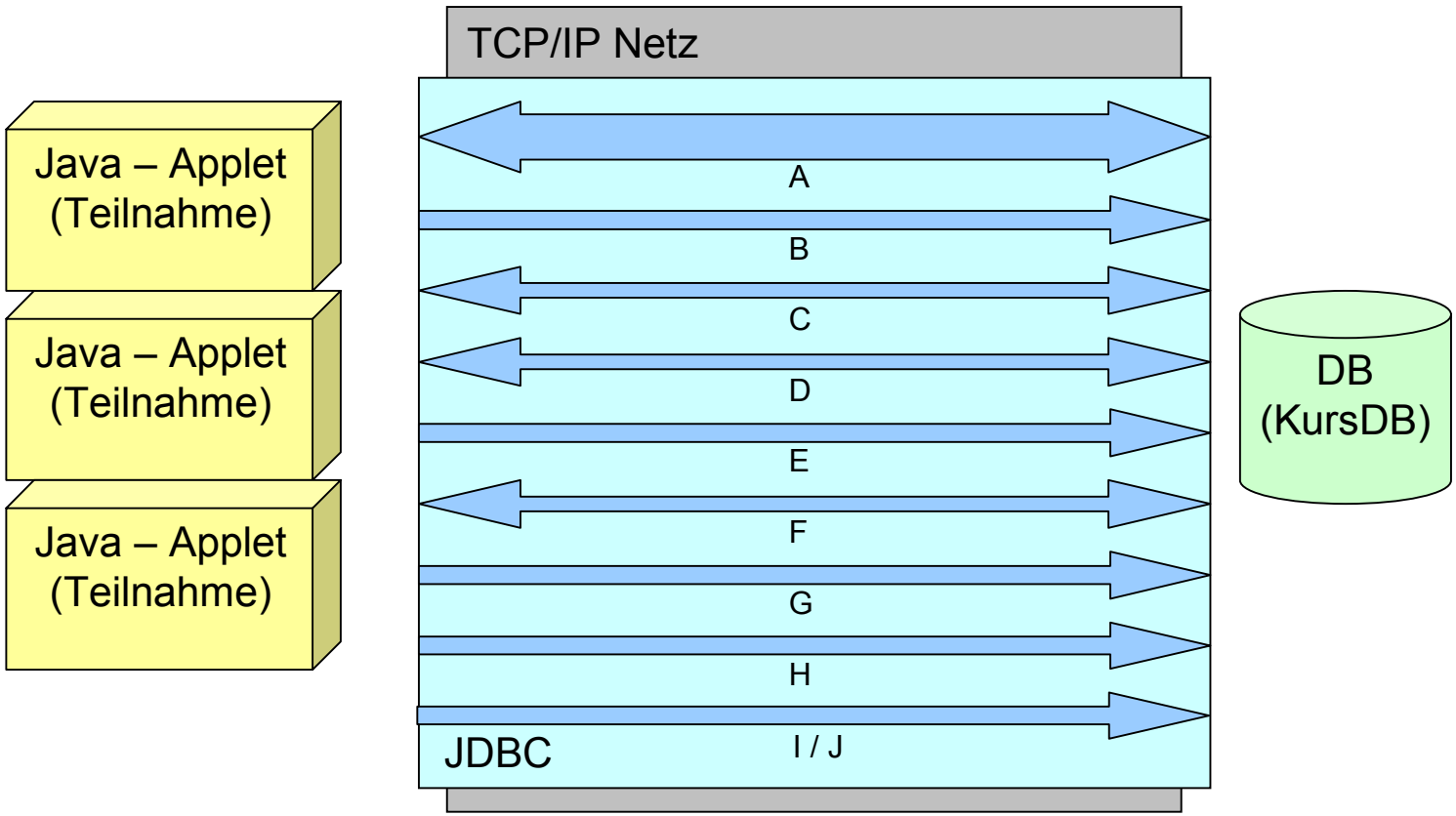
Client / Server Lösung

- Verbindung zum Datenbankserver über die JDBC-Schnittstelle
- Transaktionslogik in der Applikation
- Transaktionslogik in Stored Procedures



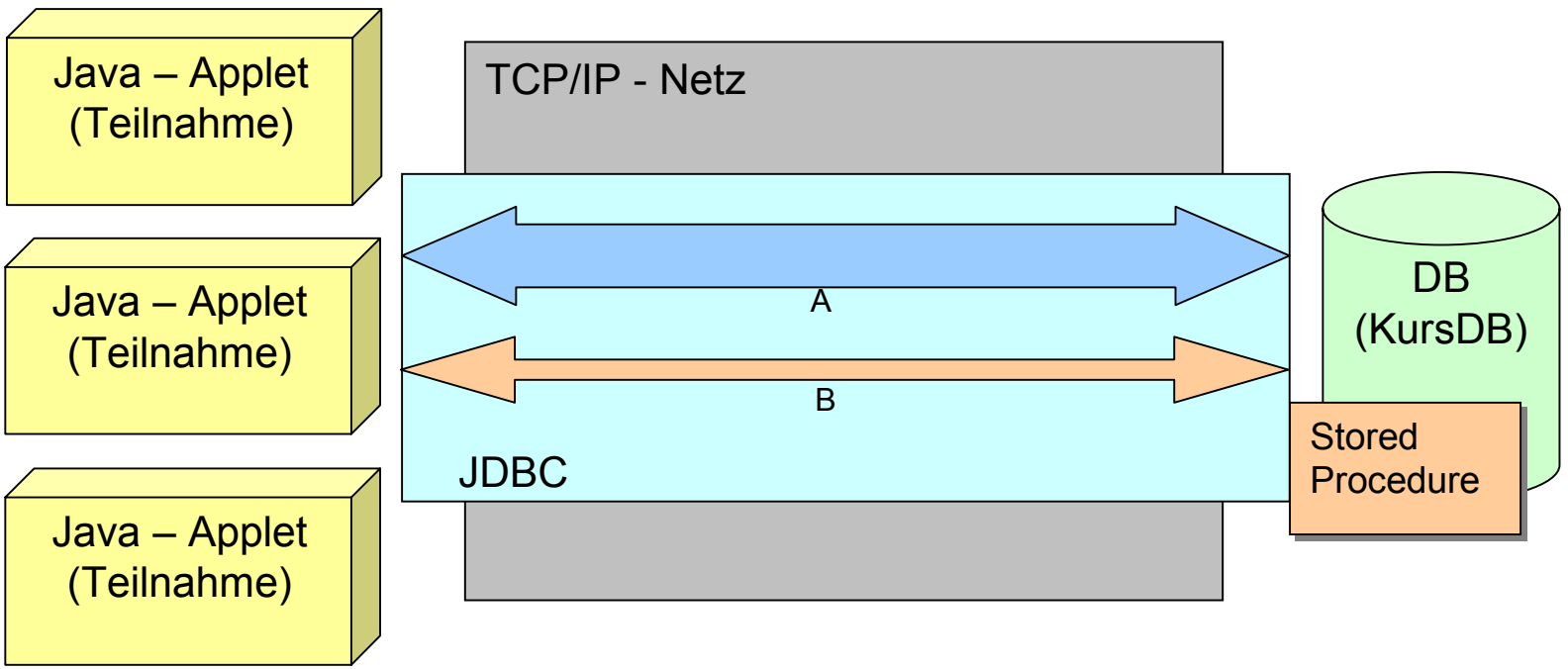
Client / Server Lösung

- Transaktionslogik in der Applikation



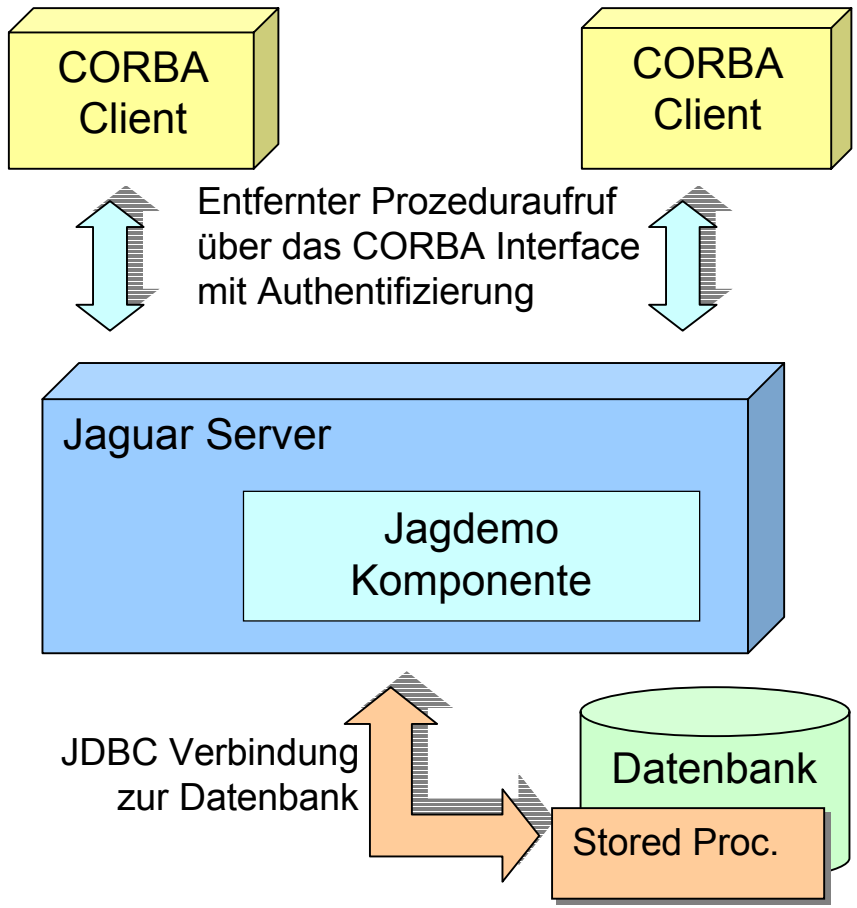
Client / Server Lösung

- Transaktionslogik in Stored Procedures



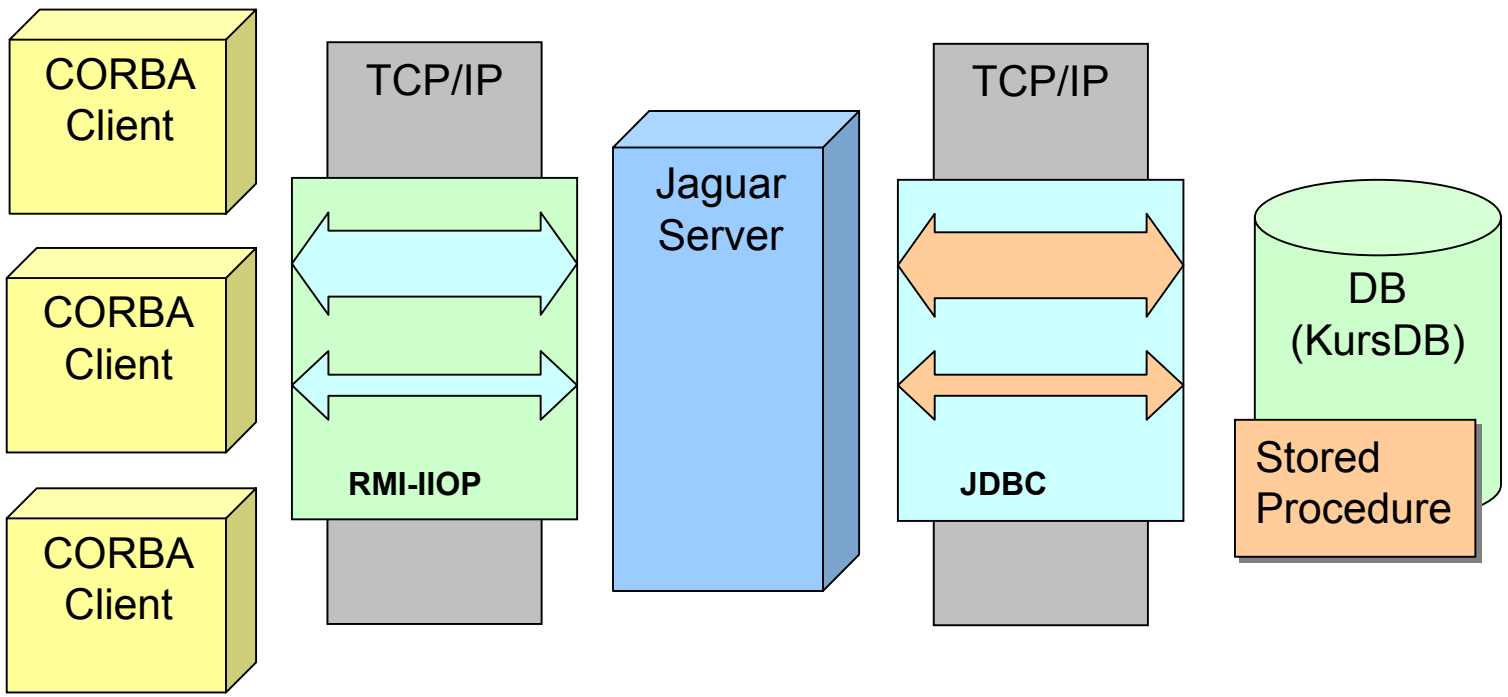
Komponentenbasierte Lösung

- keine JDBC-Schnittstelle
- Methoden werden über das CORBA Interface aufgerufen:
  - Die Ausgabe der Angebote (Angebot_holen)
  - Die Teilnahme (Teilnahme)



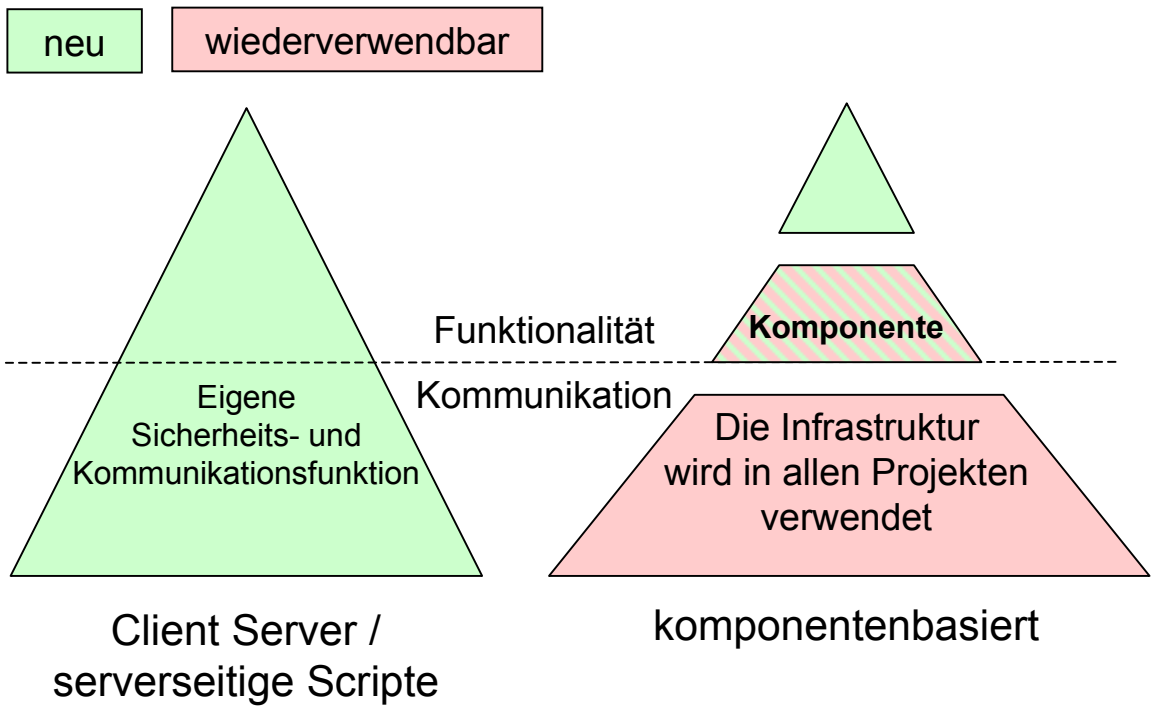
Komponentenbasierte Lösung

- Erhöhter Kommunikationsaufwand



Praktische Erfahrungen

- Die vielfältige Infrastruktur und Kommunikation ist gewöhnungsbedürftig
- Bietet gute Wiederverwendbarkeit, Performance und Sicherheit



Aussichten

- In großen Firmen ist die Nutzung von Applikationsservern schon Standard
- Immer mehr Applikationsserver-Anwendungen auch für Privatnutzer
- Bei immer schnelleren Netzanbindungen sind auch komplette Desktopanwendungen über das Internet denkbar
- J2EE großer Standard
- Microsoft .Net ist eher eine Softwarelösung