Architekturen

- Datenbankanwendungen
  - Aufgaben und Komponenten
  - Aufteilung
- Architektur Web-basierter Anwendungen
  - HTTP-basierte Architekturen
  - Applet-basierte Architekturen

DB-Anwendungen: Aufgaben

- Benutzerschnittstelle:
  - Präsentation, Benutzerinteraktion
- Anwendungslogik:
  - Implementierung der anwendungsspezifischen Funktionen
  - Bereitstellung von „Geschäftsobjekten“
- Datenmanagement:
  - Datenspeicherung und –verwaltung
  - Verarbeitung von Anfragen

Aufteilung der Funktionen

- **2 Schichten**
  - Client-Server-Architektur
  - Variante 1:
    - Client: Benutzerschnittstelle + Anwendungslogik
    - Server: DBMS
  - Variante 2:
    - Client: Benutzerschnittstelle
    - Server DBMS + Anwendungslogik

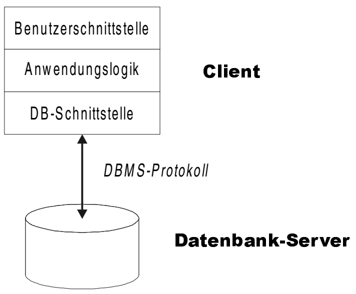
Aufteilung der Funktionen

- **3 Schichten**
  - Erweiterte Client-Server-Architektur
  - Client: Benutzerschnittstelle
  - Server: DBMS
  - Separate Schicht mit Anwendungslogik

2-Schichten-Architektur

- Server: RDBMS
- Client: RDBMS-Anwendung
- Kommunikation:
  - über DBMS-Protokoll bzw. –Schnittstelle (ODBC, JDBC)
- Objekte:
  - Relationen, Tupel
- Operationen:
  - SQL-DML

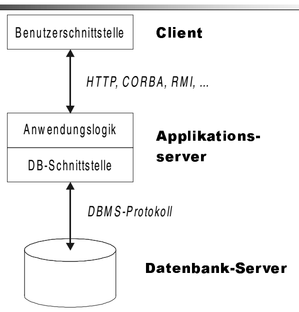
2-Schichten-Architektur



3-Schichten-Architektur

- Separater Applikationsserver mit Anwendungslogik
- Bereitstellung von Geschäftsobjekten:
  - z.B. Kunde, Produkt
- Semantisch reiche Operationen
  - z.B. Bestellung aufnehmen, Produkt ausliefern
- Kommunikation
  - Applikationsserver – DBMS: DBMS-Protokoll
  - Client – Applikationsserver: Middleware (CORBA, DCOM, RMI)

3-Schichten-Architektur

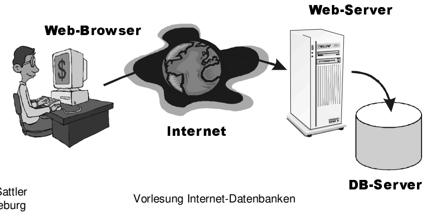


Vergleich

	2-Schichten	3-Schichten
Vorteile	▪Einfache Entwicklung ▪Optimierungs-potential (Protokoll)	▪Wiederverwendbarkeit ▪Lastverteilung ▪Wartung
Nachteile	▪Wartung, Pflege ▪Belastung Client, Server	▪Entwicklungsaufwand

Web-basierte Datenbank-Anwendungen

- Zugriff auf Daten und Dienste über Web
  - Integration von Diensten unter gemeinsamer, leicht erlernbarer Benutzeroberfläche
  - Daten als Web-Dokumente



Restriktionen

- Nutzung von
  - Standard-Komponenten
    - Web-Browser, Web-Server (HTML, JavaScript, Applets)
  - Standard-Protokollen
    - HTTP (Zustandslosigkeit, keine Transaktionen)
- Keine proprietären Erweiterungen:
  - Protokolle, Client-Programme

Architekturen für Web-DB-Anwendungen

- **Anwendungslogik**
  - Client (Browser-Erweiterungen, „aktive“ Dokumente)
  - Server (Server-Erweiterungen, Einbindung externer Dienste)
- **Kommunikation mit DB-Server**
  - Über Web-Server (HTTP)
  - Über Applikations-/Kommunikationsserver (Middleware, spezialisierte Protokolle)
  - Direkt (DBMS-Protokolle)

HTTP-basierte Ansätze

- Kommunikation zwischen Client und Server auf Basis von HTTP
  - Dienstanforderung: URL + Parameter
  - Ergebnis: HTML-Dokument
- Randbedingungen:
  - Beschränkung auf HTML zur Präsentation
  - Dynamische Erzeugung von HTML-Seiten
- Anwendungslogik im **Server**

HTTP-basierte Ansätze - Komponenten

- Browser, Web-Server
- Prozessor:
  - Verarbeitung von Makros bzw. erweiterten HTML-Seiten
- DB-Kommunikationskomponente:
  - Verbindung mit Datenbank
  - Abarbeitung der Datenbank-Befehle
  - Bsp.: ODBC, CLI

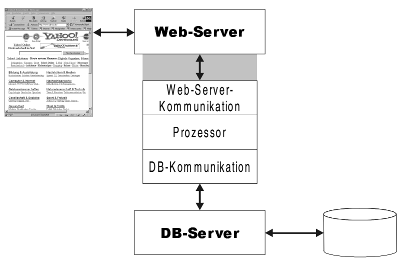
HTTP-basierte Ansätze – Komponenten

- Web-Server-Kommunikationskomponente:
  - Kommunikation zwischen Web-Server und Prozessor
  - Parameterübergabe, Ergebnisrückgabe (HTML-Dokument)
  - Bsp.: CGI, Server-Erweiterungen
- Interkomponenten-Kommunikationsmodule:
  - Kommunikation zwischen Prozessen
  - Bsp.: Sockets, CORBA, RMI

Einstufige Architektur

- Direkte Verbindung zur Datenbank
- CGI-Programm/Server-Erweiterung enthält *Prozessor*
  - Aufbau einer DB-Verbindung
  - Verarbeitung der Anforderung
  - Zurückliefern des Ergebnisses als HTML-Dokument

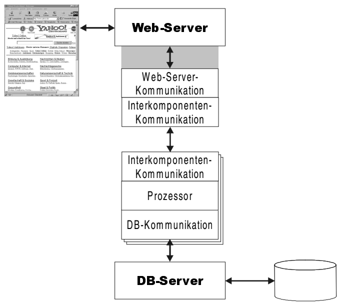
Einstufige Architektur



Zweistufige Architektur

- Trennung Prozessor und Kopplungskomponente
  - Ständig verfügbare Verarbeitungsprozesse (mit *Prozessor*) evtl. als Prozeßpool
    - Offenhalten von DB-Verbindungen
    - Lastverteilung
  - CGI-Programm/Server-Erweiterung kommunizieren mit Verarbeitungsprozeß
    - Parameterübergabe
    - Zurückliefern des HTML-Dokumentes

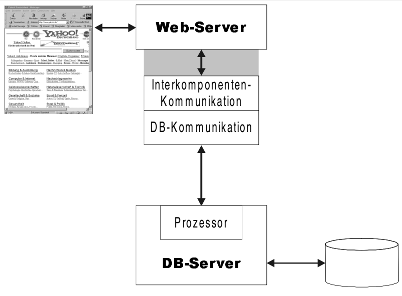
Zweistufige Architektur



Serverseitige Abarbeitung

- Ausführung von Anwendungsfunktionen und -prozeduren im DB-Server
- Verlagerung des *Prozessors* in DB-Server
- DB-Server als Web-Server
  - Verarbeitung von HTTP-Requests
  - Auslieferung von HTML-Dokumenten
- Realisierung:
  - User defined Functions, Stored Procedures

Serverseitige Abarbeitung



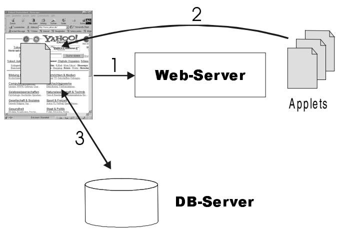
Vergleich

	einstufig	zweistufig	serverseitig
Vorteile	▪ Einfache Realisierung ▪Geringer Ressourcenbdarf ▪Für kleine Lösungen	▪Realisierung langer Transaktionen ▪Schnellere Antwortzeiten ▪Lastausgleich ▪Caching	▪Verringerte Kommunikation ▪Optimierbarkeit
Nachteile	▪Teurer Startup ▪Aufwendiger Verbindungsaufbau	▪Höherer Ressourcenbedarf ▪Aufwendige Installation und Konfiguration ▪Komplexe Entwicklung	▪Belastung des DB-Servers ▪Komplexe Entwicklung ?

Applet-basierte Ansätze

- Ausführung von Anwendungsfunktionen im Web-Browser
- Anwendungsfunktionen
  - Java-Code (Applets)
  - Visual-Basic-Code (ActiveX)
- Laden der Applets vom Web-Server
- Anwendungslogik im **Client**

Applet-basierte Ansätze



Applet-basierte Ansätze

- Kommunikation mit DBS oder Applikationsserver
  - HTTP: Dokumente laden
  - DBMS-Protokolle: JDBC
  - Middleware: CORBA, RMI
- Vorteile:
  - Reichhaltiger Funktionsumfang
    - Grafikfähigkeit, Verschlüsselung
  - Lange Transaktionen, Zustandsspeicher
  - Keine Installation notwendig

Applet-basierte Ansätze

- Probleme:
  - Sicherheitsrestriktionen
    - Kein Zugriff auf lokales Dateisystem
    - Verbindung nur zum Web-Host (Tunneling erforderlich)
  - Längere Ladezeiten
  - Anforderungen an Browser