Prof. Dr. Th. Härder
Universität Kaiserslautern
Fachbereich Informatik
E-mail: haerder@informatik.uni-kl.de
WWW: http://wwwdbis.informatik.uni-kl.de/

Vorlesung

Realisierung von Datenbanksystemen

SS01

Th. Härder

Vorlesung:

Mo., 10.00 - 11.30 Uhr, 46-280

Beginn: Mo., 23. 4. 2001

Ziele

- **Vermittlung von vertieften Kenntnissen**
  zu Entwurf, Aufbau, Realisierung und Programmierung von DBS, insbesondere
  - Architektur von zentralisierten und verteilten DBS (Schichtenmodelle),
  - Speichersystem, Zugriffssystem und Datensystem
  - Implementierungstechniken für Konzepte und Verfahren, mit denen die einzelnen Schichten eines DBS aufgebaut werden

- **Erarbeitung eines tiefergehendes Verständnisses**
  - für das Zusammenspiel der DBS-Schichten und -Komponenten
  - zur Erklärung des Leistungsverhaltens bei der Abwicklung von DB-Operationen
  - für das Entwerfen, Aufbauen und Warten von Datenbanken sowie für die Programmierung von DB-Anwendungen

- **Voraussetzungen für Übernahme von Tätigkeiten:**
  - DB-Systementwicklung und -implementierung
  - Entwicklung und Einsatz von DB-gestützten Anwendungen
  - Integration von kooperativen Informations- und Anwendungssystemen
  - Installation, Administration und Tuning von DB- und Transaktionssystemen
  - Systemverantwortlicher für Datenbanksysteme, insbesondere Unternehmens-, Datenbank-, Anwendungs- und Datensicherungsadministrator, Informationswirt, ...

ÜBERSICHT

1. Architekturen von DB- und TA-Systemen

- statische Abbildungshierarchie eines DBS – Schichtenmodell
- Aufbau von zentralisierten Transaktionssystemen
- Verteilte Transaktionssysteme: Einsatz von Mehrrechner-DBS
- Client/Server-DBS

2. Externspeicherverwaltung

- Datei- und Segmentkonzepte
- Maßnahmen zur Fehlertoleranz
- Einbringverfahren

3. DB-Pufferverwaltung

- Nutzung von Lokalität
- Verwaltung des DB-Puffers (Suche, Speicherzuordnung)
- Ersetzungsverfahren und Vorausplanung

4. Speicherungsstrukturen

- Freispeicherverwaltung
- Adressierung von Sätzen
- Satzabbildung, Cluster-Bildung

5. Eindimensionale Zugriffspfade

- Mehrwegbäume, statische und dynamische Hashverfahren
- Zugriffspfade für Sekundärschlüssel
- Zugriffspfade für Setstrukturen
- Verallgemeinerte Zugriffspfadstrukturen

ÜBERSICHT (2)

6. Mehrdimensionale Zugriffspfade

- Unterstützung für raumbezogene Zugriffe
- Organisation der Datensätze
- Organisation des umgebenden Datenraums
- Grid-File, R-Baum u. a.
- Vergleich verschiedener Zugriffsverfahren

7. Satzorientierte Schnittstelle

- Data-Dictionary-Funktionen
- Scan-Konzepte,
- Sortier-Operator

8. Implementierung relationaler Operatoren

- Operatoren auf einer und auf mehreren Relationen
- Implementierung der Verbundoperation

9. Mengenorientierte Schnittstelle

- Formen der Spracheinbettung
- Übersetzung von DB-Anweisungen
- Anfrageoptimierung (Standardisierung, Vereinfachung, Restrukturierung und Transformation, Kostenmodelle)
- Code-Erzeugung
- Ausführung von DB-Anweisungen

LITERATURLISTE

Härder, T., Rahm, E.: Datenbanksysteme — Konzepte und Techniken der Implementierung, Springer-Verlag, 1999 (Es sind Hörerscheine erhältlich).

Gray, J., Reuter, A.: Transaction Processing—Concepts and Techniques, Morgan Kaufmann Publishers, Inc., San Mateo, CA., 1998 (5th printing).

Ramakrishnan, R.: Database Management Systems, McGraw-Hill, Boston, 1998.

O'Neil, P.: Database—Principles, Programming, Performance, Morgan Kaufmann Publishers, Inc., San Mateo, CA., 1994.

Saake, G., Heuer, A.: Datenbanken: Implementierungstechniken, MITP-Verlag, 1999.

Mitschang, B.: Anfrageverarbeitung in Datenbanksystemen — Entwurfs- und Implementierungskonzepte, Reihe Datenbanksysteme, Vieweg, 1995.

ZEITSCHRIFTEN:

ACM TODS Transactions on Database Systems, ACM-Publikation (vierteljährl.)

THE VLDB Journal VLDB Foundation (vierteljährlich)

Information Systems Pergamon Press (8-mal jährlich)

ACM Computing Surveys ACM-Publikation (vierteljährlich)

TAGUNGSBÄNDE:

ICDE Proceedings, „International Conference on Data Engineering", jährliche Konferenz der IEEE

SIGMOD Proceedings, jährliche Konferenz der ACM Special Interest Group on Management of Data

VLDB Proceedings, jährliche Konferenz „Very Large Data Bases" (ACM, IEEE, ...)

BTW Tagungsbände der alle 2 Jahre stattfindenden Tagungen „Datenbanksysteme in Büro, Technik und Wissenschaft" der Gesellschaft für Informatik (GI),
sowie weitere Tagungen innerhalb des GI-FA 2.5 Informationssysteme

Datenbanktechnologie - Was ist das?

Konzepte, Methoden, Werkzeuge und Systeme für die

- dauerhafte Lebensdauer Daten > Dauer Erzeugungsprozeß
- zuverlässige Integrität, Konsistenz, Verlustsicherheit
- unabhängige wechselseite Änderungsimmunität AWP-DB

Verwaltung und

- komfortable „höhere" abstrakte Schnittstelle
- flexible Ad-hoc-Zugriffsmöglichkeit

Benutzung von

- großen Größe Daten >> Größe Hauptspeicher
- integrierten kontrollierte Redundanz von/für mehrere Anwendungen,
- mehrfachbenutzbaren paralleler Zugriff

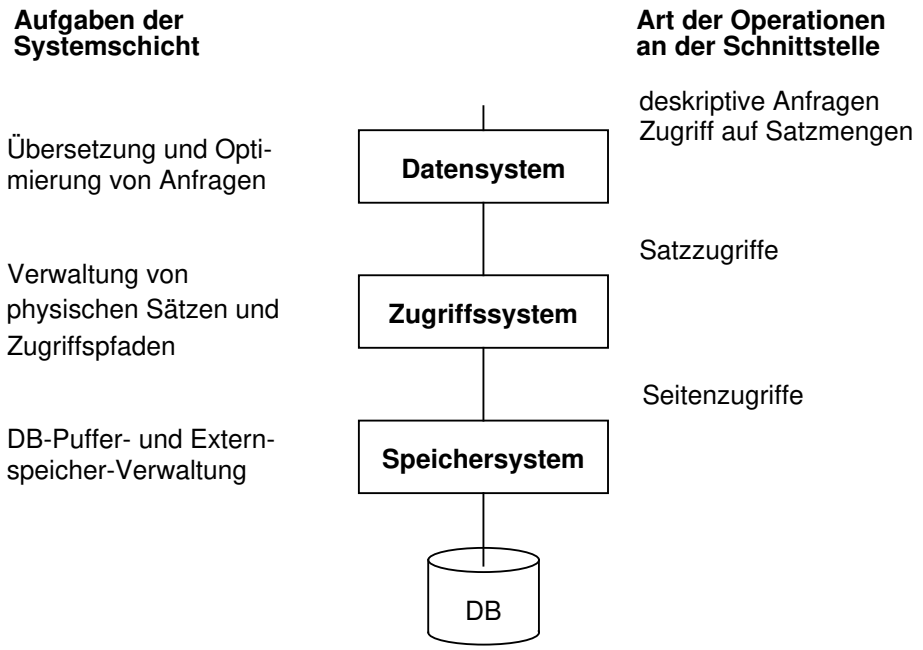
Datenbasen

1. Architekturen von DB- und TA-Systemen

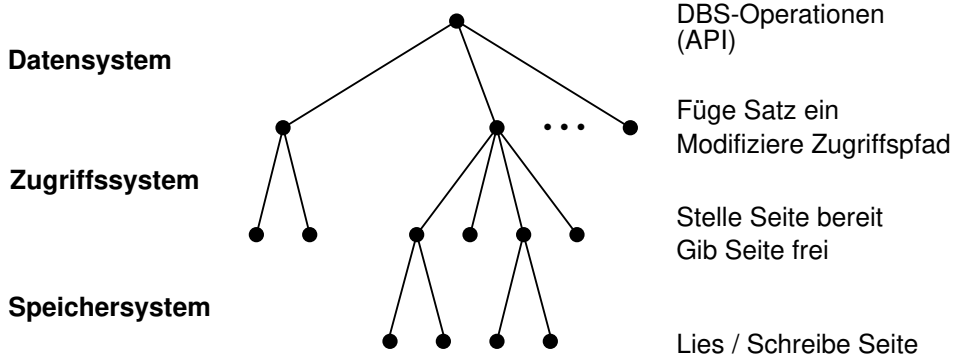
- **Schichtenmodell eines DBS**
  - schrittweise Abstraktion von einem Bitstring auf der Magnetplatte zu logischen Sichten und SQL-Operationen oder komplexen Objekten
  - Prinzipien der Schichtenbildung
  - effiziente (dynamische) Abbildung über mehrere Schichten hinweg?
- **Transaktionssysteme**
  - Grundlegende Abstraktionen und Grobaufbau
  - Schichtenmodell
  - Anforderungen
- **Verteilte Transaktionssysteme**
  - Aspekte/Prinzipien der Verteilung
  - Verteilung unter Kontrolle des TP-Monitors
  - Verteilung unter Kontrolle des DBS (Mehrrechner-DBS)
- **Client/Server-DBS**
  - „Verschicken von Anfragen" vs „Verschicken von Daten"
  - Client/Server-Architekturen für objektorientierte DBS

Schichtenmodell eines DBS – Überblick

- **Vereinfachtes Schichtenmodell**



- **Dynamischer Kontrollfluß einer Operation an das DBS**

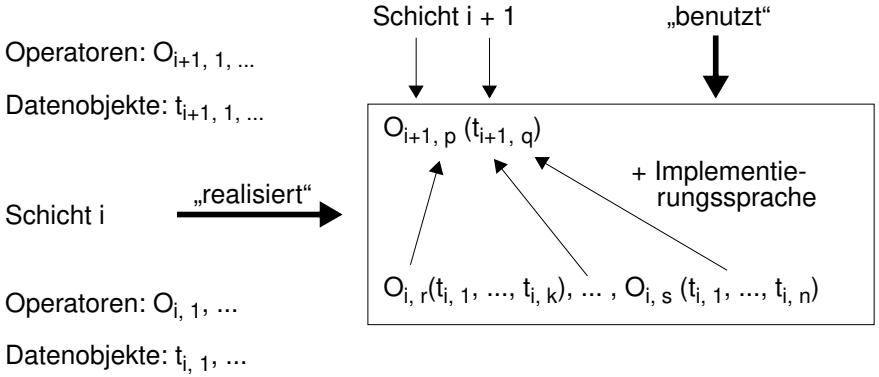


Schichtenbildung im Schichtenmodell

- **Ziel: Architektur eines datenunabhängigen DBS**
- **Wieviele Schichten braucht man?**

  ➥ Es gibt keine Architekturlehre für den Aufbau großer SW-Systeme

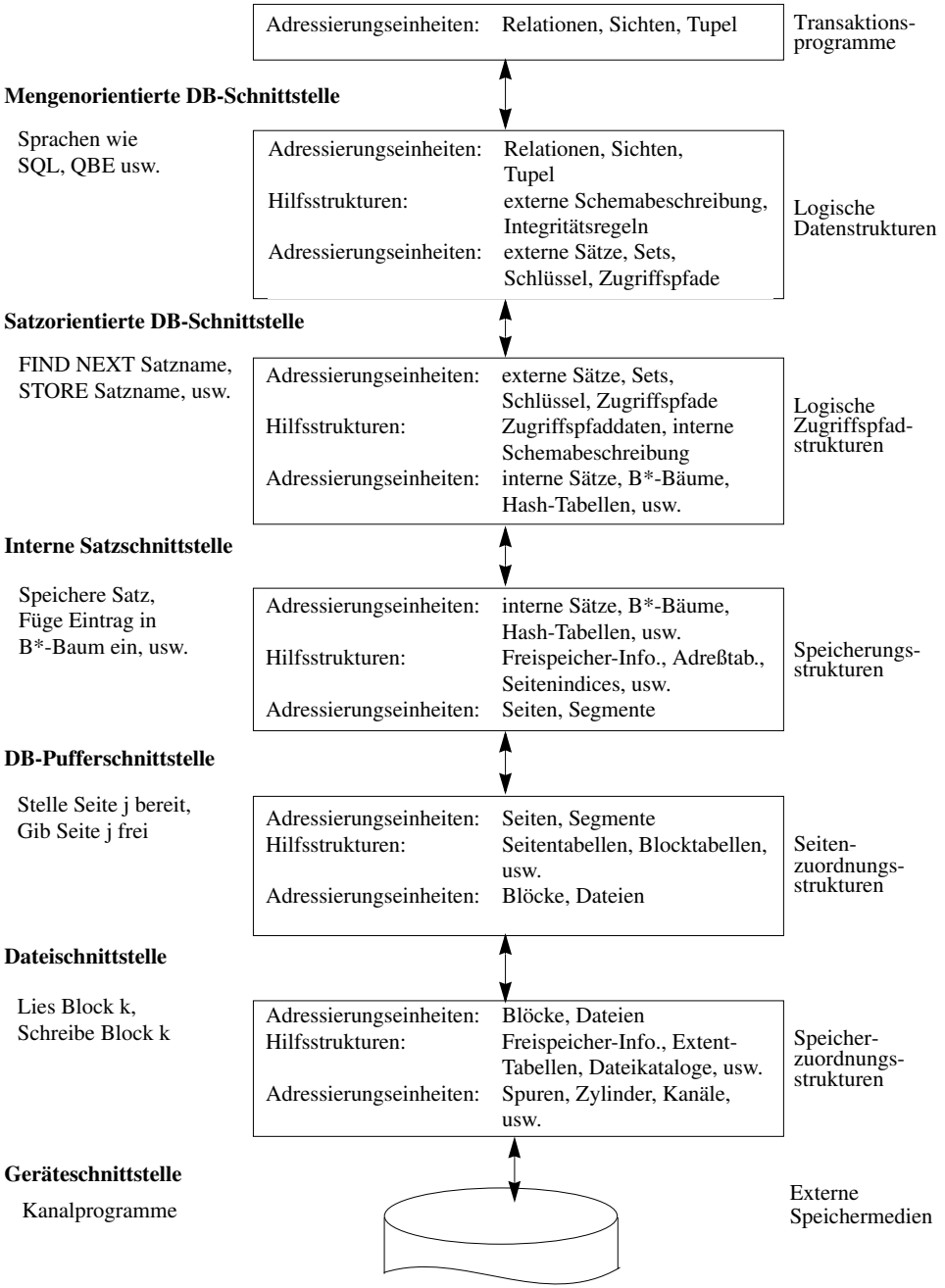
- **Empfohlene Konzepte:**
  - Geheimnisprinzip (*Information Hiding*)
  - Hierarchische Strukturierung
- **Aufbauprinzip:**



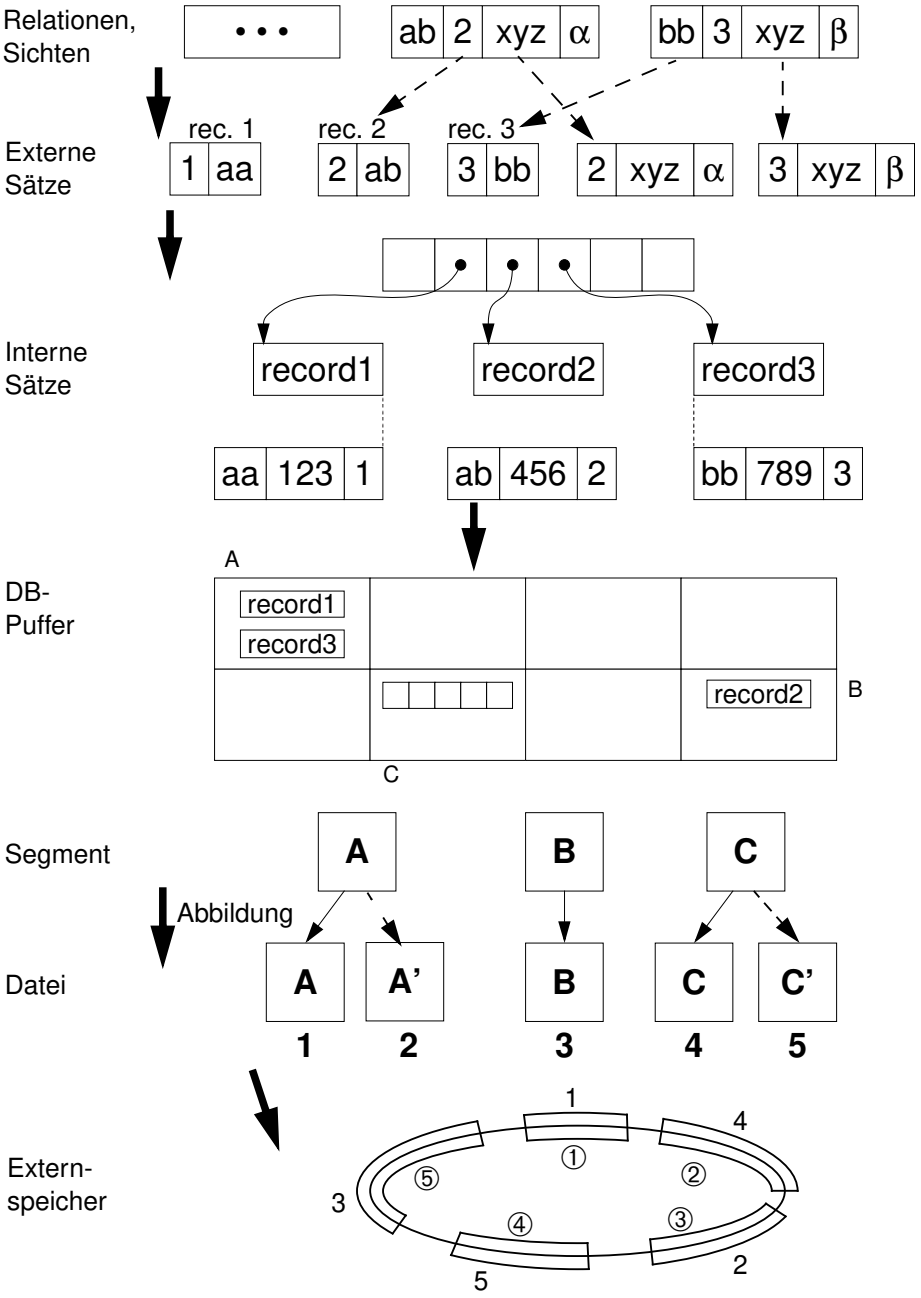
- **„benutzt“-Relation:**

  A **benutzt** B, wenn A B aufruft und die korrekte Ausführung von B für die vollständige Ausführung von A notwendig ist

Statisches Modell eines Datenbanksystems

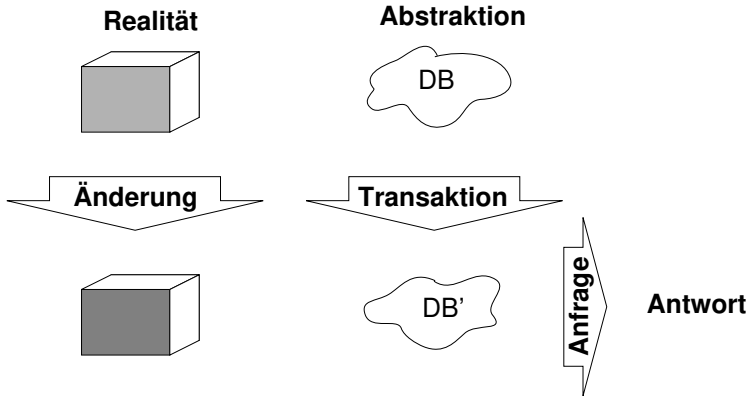


Schichtenweise Abbildungen in einem DBS



Transaktionssysteme – Grundlegende Abstraktionen

- **Abstraktion**
  - Der reale Zustand (einer Miniwelt) wird durch eine Abstraktion – DB genannt – dargestellt
  - Eine Veränderung des realen Zustands wird durch ein Programm – Transaktion genannt – in der DB nachvollzogen



- **Definitionen einer TA aus verschiedenen Sichten**
  - Mit einer Transaktion (TA) wird ein Vorgang einer Anwendung in einem Rechensystem abgewickelt.
    Ein solcher Vorgang bildet typischerweise einen **nicht-trivialen Arbeitsschritt (*unit of work*)** in betrieblichen Abläufen.
  - Eine (On-line) Transaktion ist die Ausführung eines Programmes, das mit Hilfe von Zugriffen **auf eine gemeinsam genutzte Datenbank** (DB) eine Anwendungsfunktion erfüllt.
  - Eine DB-Transaktion ist eine **ununterbrechbare Folge von DB-Operationen**, welche die Datenbank von einem logisch konsistenten in einen logisch konsistenten Zustand überführt.

ACID-Eigenschaften

Eine Transaktion ist eine **Sammlung von Aktionen** mit folgenden Eigenschaften:

- ***Atomicity***

  Die Änderungen einer TA, die den Zustand der Miniwelt betreffen, sind atomar; es gilt „Alles oder Nichts“. Zu diesen Änderungen gehören **DB-Aktualisierungen, Nachrichten und Operationen auf Steuerungsgeräten.**

- ***Consistency***

  Eine TA ist eine korrekte Transformation des Miniwelt-Zustandes. Die Operationsfolge verletzt keine der Integritätsbedingungen, die mit dem Zustand verknüpft sind. Deshalb muß eine TA ein korrektes Programm sein.

- ***Isolation***

  Trotz der konkurrenten Abwicklung von TA erscheint jeder TA T, daß andere TA entweder vor T oder nach T ablaufen, aber nicht beides zusammen.

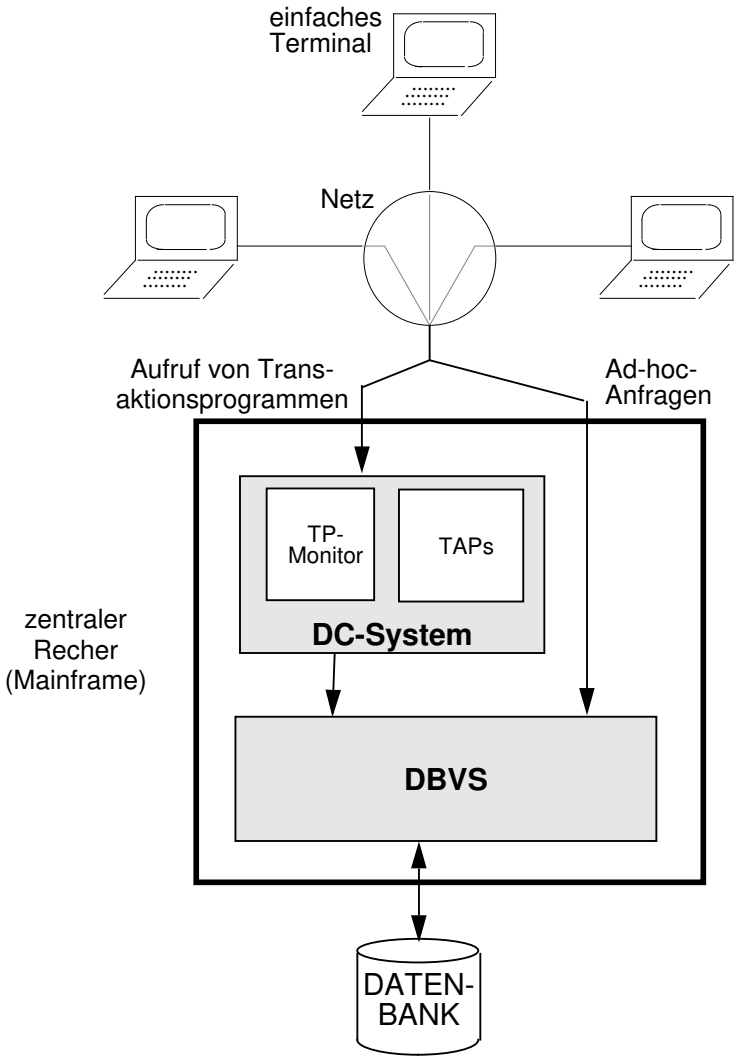
- ***Durability***

  Sobald eine TA erfolgreich beendet wird (*commit* ausführt), überleben ihre Zustandsänderungen alle erwarteten Fehler

➥ Was garantieren diese Eigenschaften beim Ablauf einer Kontenbuchungs-TA?

- **Struktur eines TA-Programms**
  - BEGIN_WORK() → alle nachfolgenden Operationen gehören zur TA
  - COMMIT_WORK() → neuer konsistenter Zustand wird persistent (*Durability*)
  - ROLLBACK_WORK() → Fehler erzwingt rückwirkungsfreies Rücksetzen der TA (*Atomicity*)

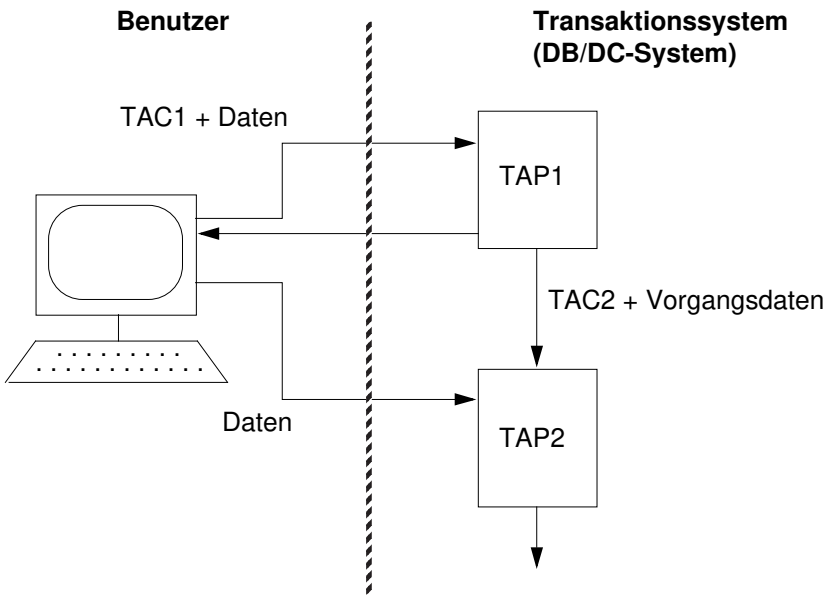
Grobaufbau eines zentralisierten Transaktionssystems



TP-System = Transaktionssystem (*Transaction Processing System*)
TAP = TA-Programm/Anwendungsprogramm
TP-Monitor = *Transaction Processing Monitor*
DC-System = Datenkommunikationssystem

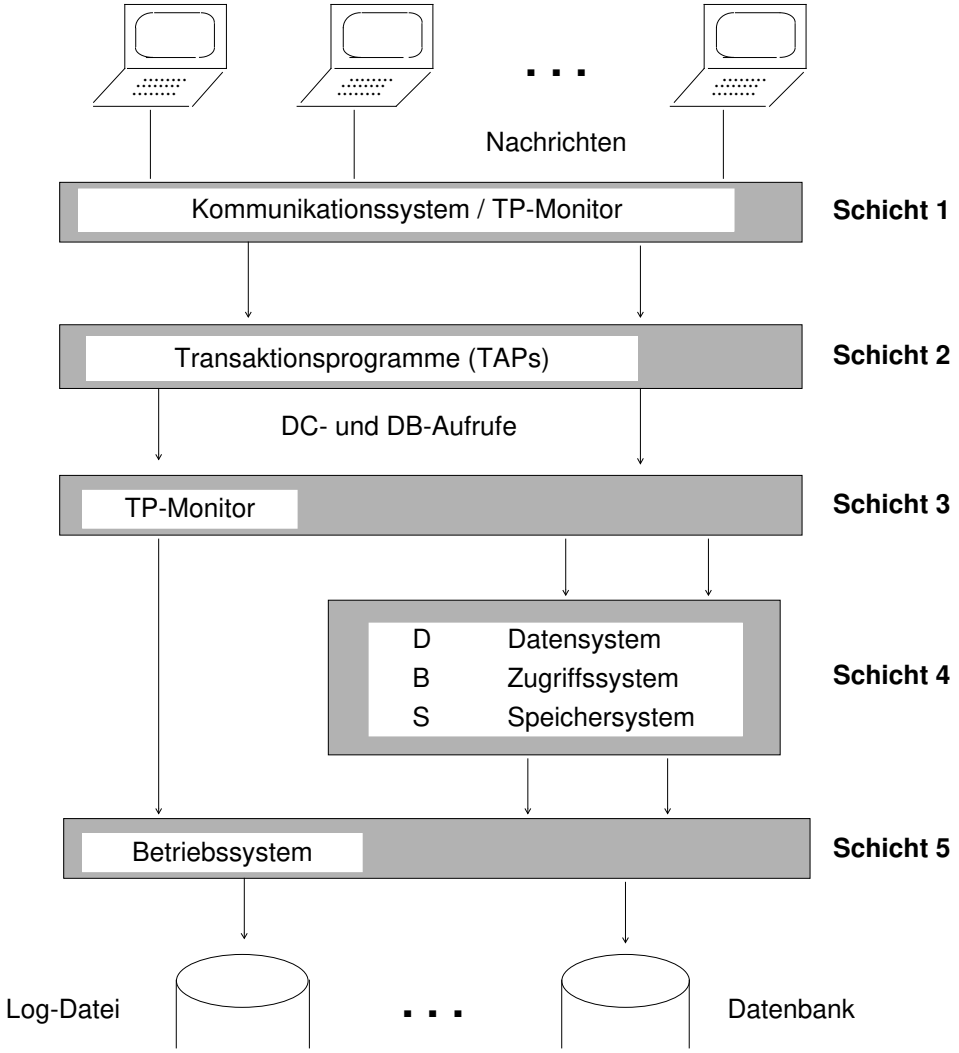
Transaktionssysteme – operationale Eigenschaften

- **Prinzipieller Ablauf**



- **System- und Betriebsmerkmale**
  - Benutzung im Dialog: „parametrischer“ Benutzer
  - wenige kurze Transaktionstypen: hohe Wiederholrate
  - sehr viele Benutzer gleichzeitig
  - gemeinsame Datenbestände mit größtmöglicher Aktualität
  - kurze Antwortzeiten als Optimierungsziel vor Auslastung
  - stochastisches Verkehrsaufkommen
  - hohe Verfügbarkeit des Systems

Schichtenmodell eines Transaktionssystems



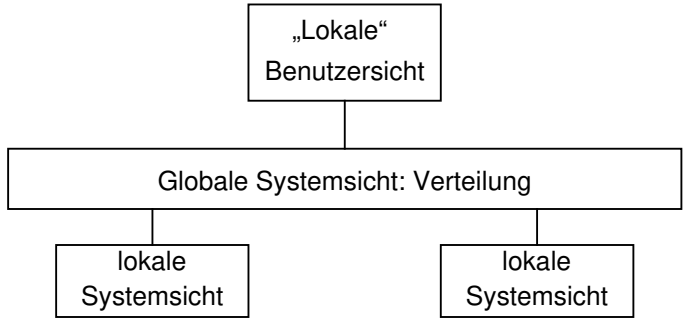
Anforderungen an Transaktionssysteme

- **Abwicklung sehr hoher TA-Raten**
  - TA-Typ „Kontenbuchung“: Maßeinheit bei den Benchmarks TPC-A, TPC-B
    ➥ n Ktps vom Typ TPC-B?
  - komplexere Transaktionen „Abwicklung von Bestellungen“ beim TPC-C
    ➥ n*100 Ktpm vom Typ TPC-C
  - Es ist immer mehr Funktionalität gefragt!
    (benutzerdefinierte Datentypen, große Objekte, Multimedia, ...)
- **Gewährleistung hoher Verfügbarkeit**
  - Vermeidung eines Systemausfalls
  - Verfügbarkeit aller Daten
  - **Mehrrechner-Architekturen** erforderlich
- **Modulares Systemwachstum**
  - Subsysteme als Einheiten des Entwurfs, der Kontrolle und des Ausfalls
  - annähernd lineare Durchsatzerhöhung
  - Zuordnung von Prozessoren und **Daten** (Platten)
- **Einbindung in Client/Server-Architekturen**
  - 2-stufig: Präsentation - Anwendung/Datenhaltung
  - 3-stufig: Präsentation - Anwendung - Datenhaltung (Skalierbarkeit!)
  - 4-stufig: Einbezug von Web-Servern

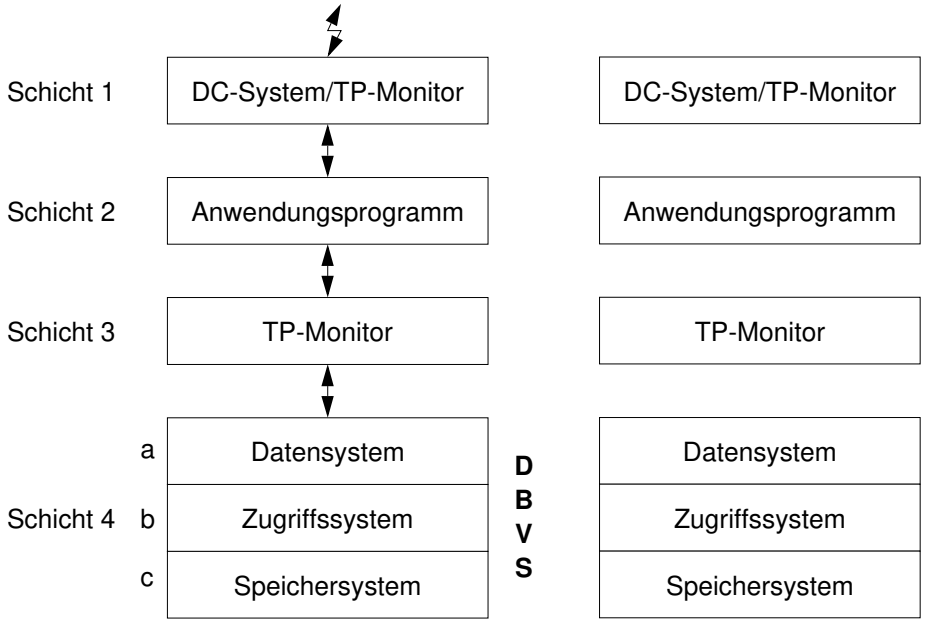
➥**Transaktionssysteme sind** i. allg. horizontal und/oder vertikal **verteilt**!

Klassifikation verteilter Transaktionssysteme

- **Vorgehensweise bei der Verteilung**
  - Partitionierung der Daten (ggf. mit Replikation)
  - Kommunikation zwischen Prozessen zum Zugriff auf jeweils lokalen Daten
- **gewünschte Sichtbarkeit**



- **Homogenes Systemmodell**



Klassifikation verteilter Transaktionssysteme (2)

- **Realisierung der globalen Systemsicht: wo?**
  - Forderung: lokale Sicht des Benutzers (Ortstransparenz, „single system image“)
  - Abbildung auf lokale Sichten der Knoten
- **Verteilung unter Kontrolle des TP-Monitors *(Verteilte DC-Systeme)***
  - TP-Monitor verbirgt weitgehend Heterogenität bezüglich Kommunikationsprotokollen, Netzwerken, BS und Hardware
  - DBS bleiben weitgehend unabhängig (keine Kooperation zwischen DBS)
  - heterogene DBS möglich

    ➥ Einsatz von DBS-Gateways
  - geringe Implementierungskomplexität
- Drei wesentliche Alternativen (werden hier nicht vertieft):

1. ***Transaction Routing***
   - globale Sicht in Schicht 1 (Kommunikationssystem)
   - Einheit der Verteilung ist die Transaktion (TA-Typ)

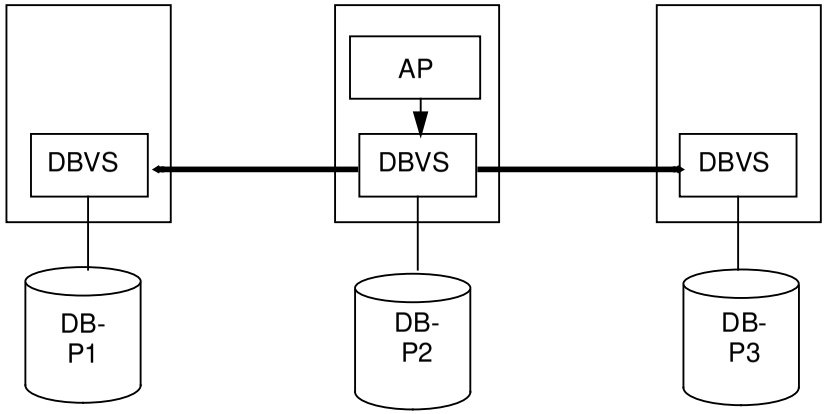
2. **Programmierte Verteilung**
   - globale Sicht in Schicht 2 (im AP)
   - Einheit der Verteilung ist eine Teil-Transaktion (Programmfragment)

3. **Verteilung von DB-Operationen (*Function Request Shipping*)**
   - globale Sicht in Schicht 3 (TP-Monitor)
   - Einheit der Verteilung ist ein DML-Befehl

Klassifikation verteilter Transaktionssysteme (3)

- **Verteilung unter Kontrolle des DBS (→ Mehrrechner-DBS)**
  - vollständige Verteilungstransparenz für AP
  - setzt eine logische Datenbank voraus
  - geringe Eignung für Knotenautonomie und Heterogenität
- **mehrere (homogene) Realisierungsalternativen**
  - globale Sicht in Schicht 4
  - Verteilung ganzer DML-Befehle bzw. von Teiloperationen (Schicht 4a)
  - Verteilung von internen Satzoperationen (Schicht 4b)
  - Verteilung von Seitenzugriffen (Schicht 4c)
- **gleichgestellte Verarbeitungsrechner vs. Spezialisierung (Client/Server-Konfigurationen)**
- **Realisierung allgemeiner Resource-Manager-Architekturen**
  - Resource Manager (RMs) sind Systemkomponenten, die **Transaktionsschutz für ihre gemeinsam nutzbaren Betriebsmittel** (BM) bieten
  - Sie gestatten die externe Koordination von BM-Aktualisierungen durch ein 2PC-Protokoll
  - Beispiele: DBS, Dateisysteme, PS-Laufzeitsystem, Mail-Service, Window-Manager, ...

Mehrrechner-DBS



- **Verteilung unter Kontrolle der DBVS**
  - Kooperation zwischen (homogenen) DBVS
  - Ortstransparenz für AP (ein DB-Schema)*: single system image*
  - höchste Flexibilität für Daten- und Lastverteilung
- **Architekturklassen**

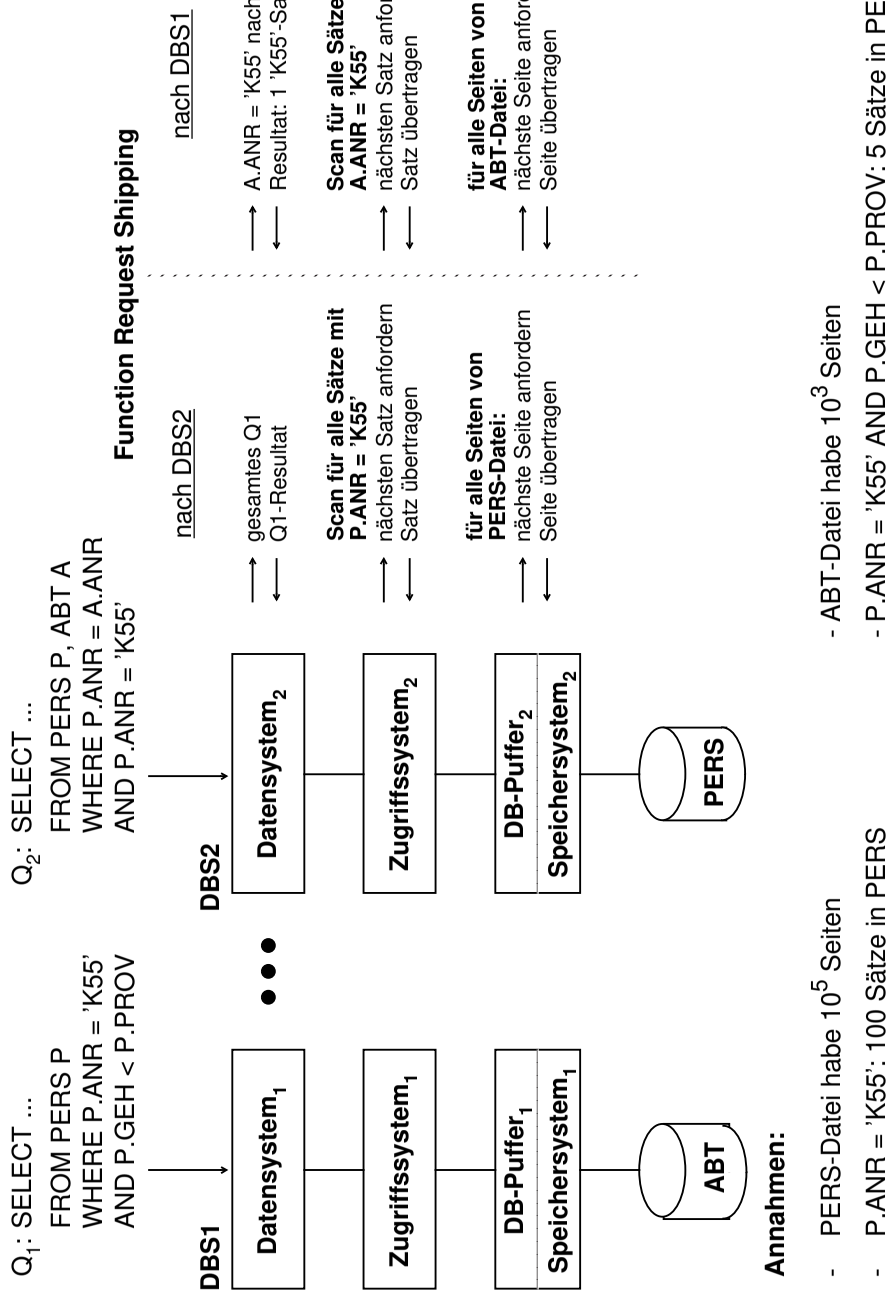
1. DB-Partitionierung (*Shared Nothing*, VDBS)

- Jeder Knoten besitzt volle Funktionalität und verwaltet eine Datenpartition
- Datenreplikation erhöht Verfügbarkeit und Zuverlässigkeit: Minimierung der Auswirkung von „entfernten“ Fehlern, Fehlermaskierung durch Redundanz
- Verarbeitungsprinzip: **die Last folgt den Daten**

2. DB-Sharing (*Shared Disk*)

- lokale Ausführung von DML-Befehlen
- Verarbeitungsprinzip: **Datentransport zum ausführenden Rechner**
- lokale Erreichbarkeit der Externspeicher

Verarbeitungsaufwand bei VDBS



Transaktionsverarbeitung [1] in Client/Server-DBS

- **Entscheidend für die DB-bezogene Leistungsfähigkeit**
  - Art und Komplexität der DB-Operationen (mengenorientiert, navigierend)
  - Nutzung der Referenzlokalität bei den Datenzugriffen

- **Bisheriges Verarbeitungskonzept**

  Bei allen bisherigen Architekturvorschlägen wurde ein „Verschicken der Anfragen“ zum Server-DBMS unterstellt (query shipping approach):
  - Die DML-Anweisung wird zum Server geschickt
  - Nach ihrer Verarbeitung wird das Ergebnis der AW (Client) zur Verfügung gestellt.

  ➥ keine Nutzung vorhandener Referenzlokalität im Client

- **Weiteres Verarbeitungskonzept**

  Folgende Vorgehensweise wird von den meisten OODBS verfolgt (data shipping approach):
  - Alle Daten, die zur Ausführung einer Anfrage benötigt werden, werden zum Client geholt und dort gepuffert
  - Danach wird die Anfrage (oder mehrere Anfragen hintereinander) im Client-Puffer ausgewertet

  ➥ Dieses Konzept ist vor allem bei hoher Referenzlokalität vorteilhaft. Es wird durch das **Checkout/Checkin-Konzept** genutzt.

1. Härder, T., Meyer-Wegener, K.: Transaktionssysteme in Workstation/Server-Umgebungen, Informatik - Forschung und Entwicklung (1990), 5: 127-143.

Transaktionsverarbeitung[1] in Client/Server-DBS (2)

- **Vorteile des „Verschickens von Daten“**
  - Datenpufferung reduziert die Interaktionen zwischen Client und Server (Verminderung der Netzbelastung; Vermeidung von wiederholten Server-Anfragen)
  - Es erhöhen sich die aggregierte Rechnerleistung und die insgesamt verfügbare Speicherkapazität des Systems
  - Die Server-Last wird reduziert
  - Die Skalierbarkeit des Systems wird verbessert

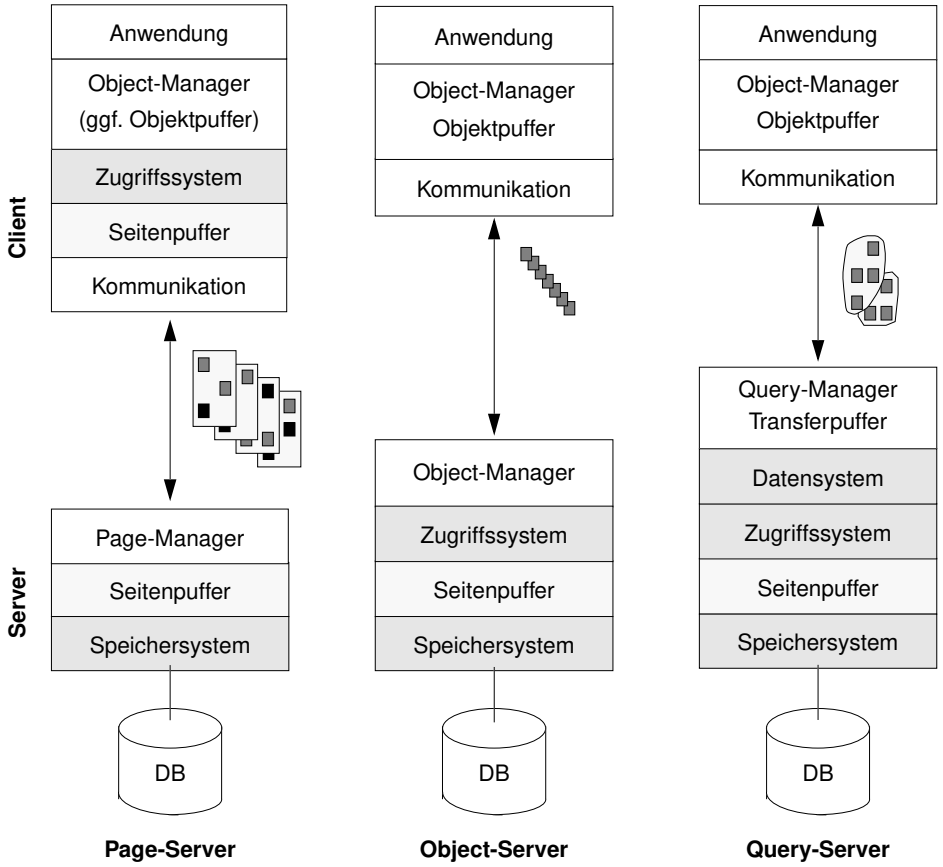
- **Ingenieur-AW brauchen eine lokale Datenhaltung**
  - Ausnutzung der Referenzlokalität bei der Verarbeitung
  - Lange Speicherung (Objektpuffer) großer Datenmengen im HSP
  - Lokale Darstellung der Objekte am Bildschirm
  - Schnelle Transformation und Verarbeitung komplexer Objekte
  - Erforderliche Eigenschaften auf Client-Seite, insbesondere
    - große Hauptspeicherkapazität
    - hohe Verarbeitungsleistung
    - breitbandige Datenübertragung
    - hochauflösende Darstellungsqualität
    - ggf. lokale Externspeicheranbindung

  ➥ DBS, die Entwurfsumgebungen unterstützen sollen, benötigen auf Client- und Server-Seite geeignete Funktionalität zur Datenverwaltung/Transaktionsunterstützung.

1. Neben dem Begriff Client/Server-Architektur wird auch der Begriff „**Workstation/Server-Architektur**“ benutzt; er soll explizit machen, daß auf Client-Seite Workstation-Eigenschaften benötigt werden.

Schichtenmodelle für Client/Server-DBS

- **Client/Server-Architekturen für objektorientierte DBS**
  - File-Server, Page-Server
  - Object-Server
  - Query-Server



➥ Im Objektpuffer erfolgt typischerweise navigierende Verarbeitung!

Zusammenfassung

- **Allgemeine Aspekte des Schichtenmodells**
  - Schichtenmodell ist allgemeines Erklärungsmodell für die DBS-Realisierung
  - Schichtenbildung läßt sich zweckorientiert verfeinern/vergröbern: Anwendbarkeit für TA-Systeme, verteilte DBS, Client/Server-DBS, ...
  - Entwurf geeigneter Schnittstellen erfordert große Implementierungserfahrung
  - Konkrete Implementierungen verletzen manchmal die Isolation der Schichtenbildung aus Leistungsgründen (→ kritische Funktionen haben „Durchgriff“)
- **Implementierungskonzepte**
  - Die grundlegenden Implementierungskonzepte zentralisierter DBS finden sich auch in jedem Knoten eines verteilten DBS
  - Die Kenntnis der Implementierungskonzepte ermöglicht es, existierende DBS objektiv zu beurteilen und zu vergleichen
  - Ihre genaue Kenntnis ist Voraussetzung für Leistungsanalysen bzw. Abschätzungen des Systemverhaltens
  - Durch Identifizieren weniger, grundlegender Konzepte gelangt man zu einem tieferen Verständnis dessen, was mit „Datenunabhängigkeit“ gemeint ist
- **Klassifikation Verteilter Transaktionssysteme**
  - Transaction Routing, Programmierte Verteilung, Aufruf von DB-Operationen
  - Mehrrechner-DBS als Shared-Nothing oder Shared-Disk-Systeme
- **Client/Server-Systeme**
  - Data Shipping ist vor allem bei hoher Referenzlokalität vorteilhaft. Es wird durch das Checkout/Checkin-Konzept genutzt.
  - Ingenieur-AW brauchen große Datenmengen (lokale Pufferung der von der AW benötigten DB-Objekte, Verarbeitung dieser Objekte in HSP-Strukturen, Objektdarstellung am Bildschirm)
  - Nutzung von Lokalität in einem Objektpuffer im Client