Implementierung von Datenbanksystemen

Kapitel 1: Architekturen von Datenbanksystemen

Norbert Ritter
Universität Hamburg

Wintersemester 2007/2008

Kapitel 1: Architekturen

- Abbildungshierarchie eines DBS – Schichtenmodell
  - 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
- Drei-Schema-Architektur nach ANSI-SPARC
- Weitere Architekturszenarien
  - Verteilte DBS: Einsatz von Mehrrechner-DBS und FDBMS
  - Schichtenmodelle für Client/Server-DBS
  - Architektur von Transaktionssystemen

Architektur

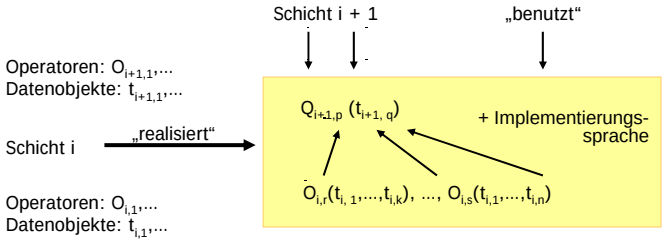
Entwurfsziele für DBS

- Integration der Daten und ihre unabhängige sowie logisch zentralisierte Verwaltung
- Datenunabhängigkeit und Anwendungsneutralität beim logischen und physischen Datenbankentwurf
- Einfache und flexible Benutzung der Daten durch geeignete Anwendungsprogrammierschnittstellen
- Zentralisierung aller Maßnahmen zur Integritätskontrolle
- Transaktionsschutz für die gesamte Datenbankverarbeitung
- Effiziente und parallele Verarbeitung von großen Datenbasen
- Hohe Verfügbarkeit und Fehlertoleranz
- Skalierbarkeit bei Wachstum der Transaktionslast und der Datenvolumina

Architektur

Schichtenbildung im Schichtenmodell

- Ziel: Architektur eines datenunabhängigen DBS
- Wie viele 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

Architektur

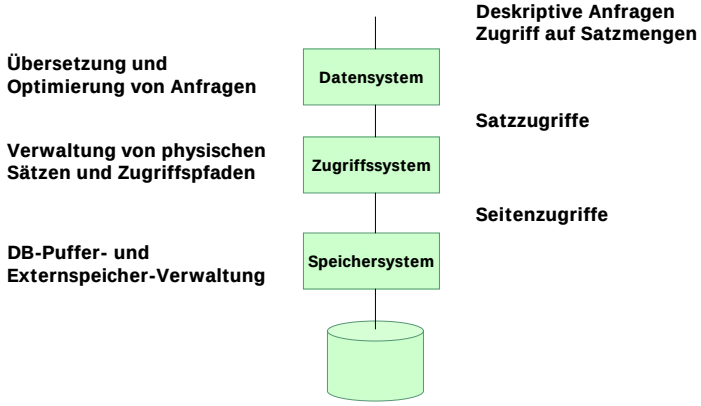
Schichtenbildung im Schichtenmodell

- Vorteile als Konsequenzen der Nutzung hierarchischer Strukturen und der „benutzt“-Relation
  - höhere Ebenen (Systemkomponenten) werden einfacher, weil sie tiefere Ebenen (Systemkomponenten) benutzen können
  - Änderungen auf höheren Ebenen sind ohne Einfluss auf tieferen Ebenen
  - höhere Ebenen können abgetrennt werden, tiefere Ebenen bleiben trotzdem funktionsfähig
  - tiefere Ebenen können getestet werden, bevor die höheren Ebenen lauffähig sind
- Jede Hierarchieebene kann als abstrakte oder virtuelle Maschine aufgefasst werden
  - Programme der Schicht i benutzen als abstrakte Maschine die Programme der Schicht i-1, die als Basismaschine dient
  - abstrakte Maschine der Schicht i dient wiederum als Basismaschine für die Implementierung der abstrakten Maschine der Schicht i+1
- Eine abstrakte Maschine entsteht aus der Basismaschine durch Abstraktion
  - einige Eigenschaften der Basismaschine werden verborgen
  - zusätzliche Fähigkeiten werden durch Implementierung höherer Operationen für die abstrakte Maschine bereitgestellt
- Programme einer bestimmten Schicht können die der nächst tieferen Schicht genau so benutzen, als sei die untere Schicht Hardware

Architektur

Schichtenmodell eines DBS – Überblick

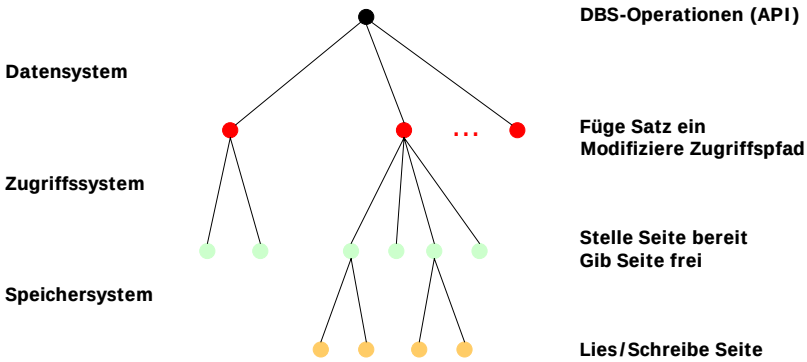
- Vereinfachtes Schichtenmodell:
  - Aufgaben der Systemschicht
  - Art der Operationen an der Schnittstelle



Architektur

Schichtenmodell eines DBS – Überblick

- Dynamischer Kontrollfluss einer Operation an das DBS

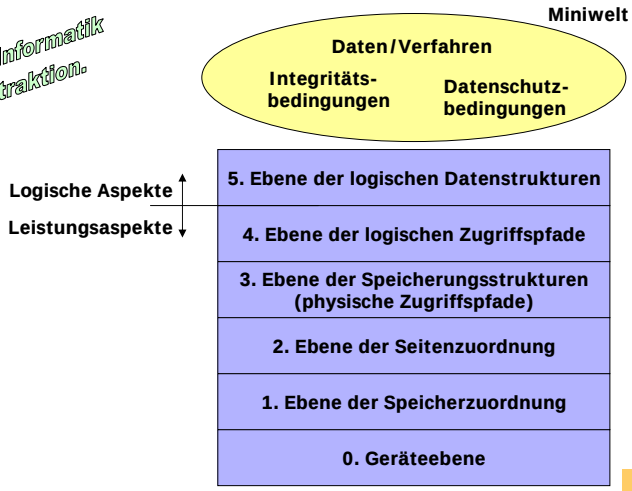


Architektur

Schichtenmodelle für DBS

- Ebenen beim Entwurf eines DBS

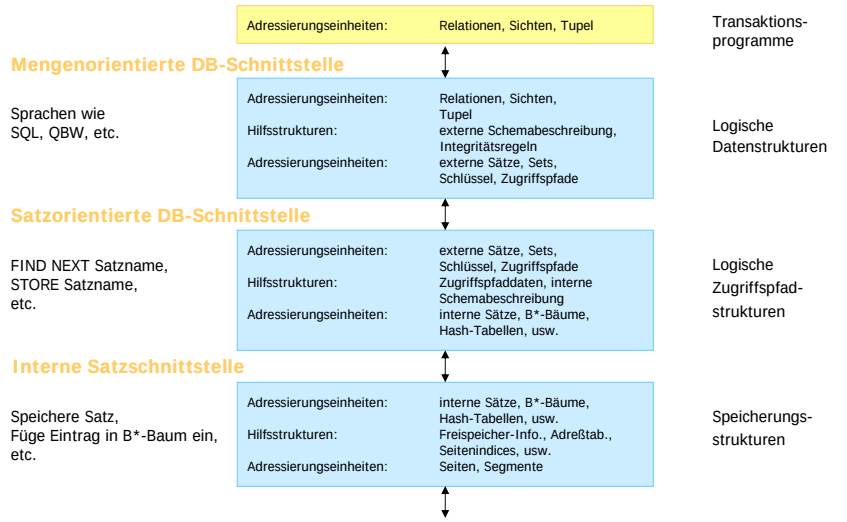
Eine Hauptaufgabe der Informatik ist systematische Abstraktion. (H. Wedekind)



Quelle: [HR01]

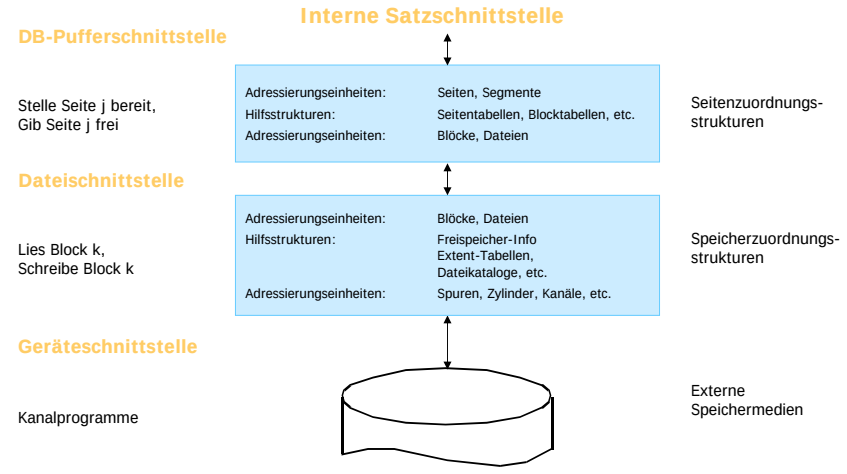
Architektur

Statisches Modell eines Datenbanksystems



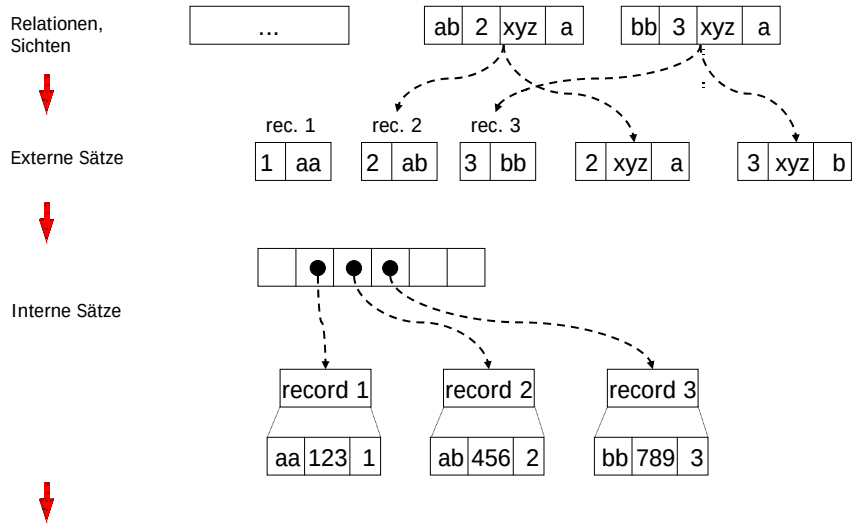
Architektur

Statisches Modell eines Datenbanksystems



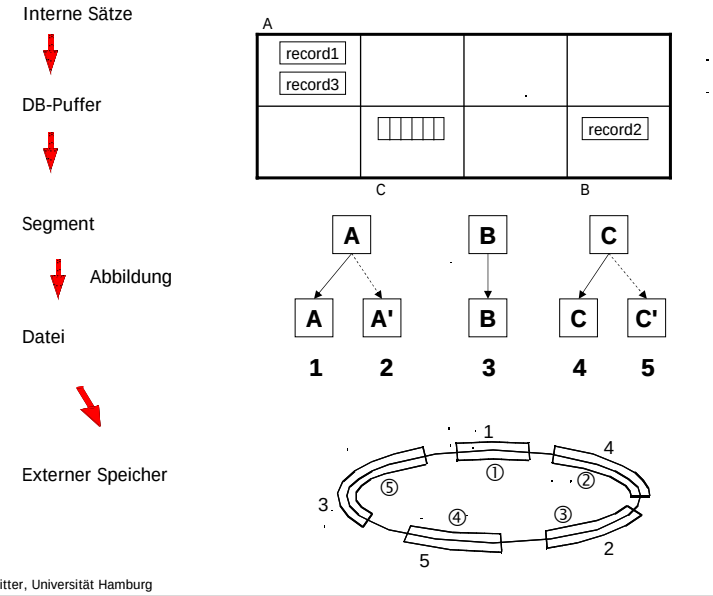
Architektur

Schichtenweise Abbildungen in einem DBS



Architektur

Schichtenweise Abbildungen in einem DBS

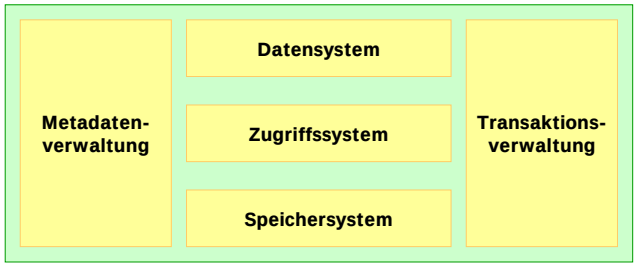


Datenunabhängigkeit im Überblick

- Ebene
  - Logische Datenstrukturen
  - Logische Zugriffspfade
  - Speicherungsstrukturen
  - Seitenzuordnungstrukturen
  - Speicherzuordnungsstrukturen
- Was wird verborgen?
  - Positionsanzeiger und explizite Beziehungskonstrukte im Schema
  - Zahl und Art der physischen Zugriffspfade; interne Satzdarstellung
  - Pufferverwaltung; Recovery-Vorkehrungen
  - Dateiabbildung, Recovery-Unterstützung durch das BS
  - Technische Eigenschaften und Betriebsdetails der externen Speichermedien

Weitere Komponenten in der DBS-Architektur

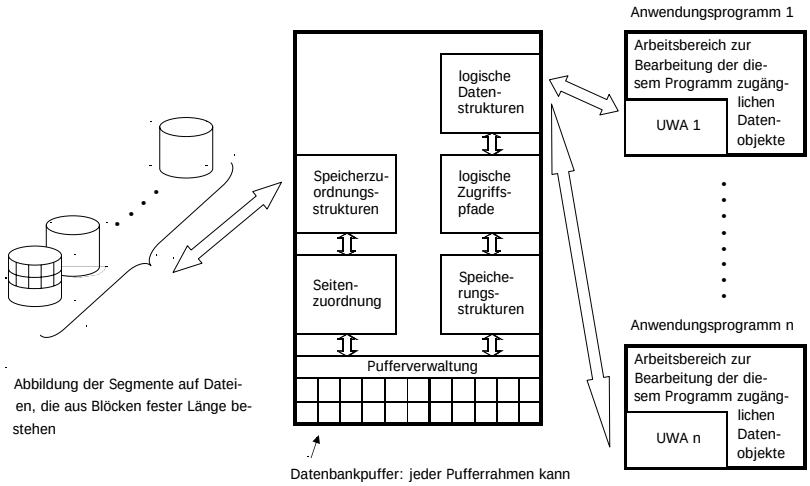
- Entwurfsziel:
  DBS sollen von ihrem Aufbau und ihrer Einsatzorientierung her in hohem Maße generische Systeme sein. Sie sind so zu entwerfen, dass sie flexibel durch Parameterwahl und ggf. durch Einbindung spezieller Komponenten für eine vorgegebene Anwendungsumgebung konfigurierbar sind.
- Überblick:



Weitere Komponenten in der DBS-Architektur

- Metadaten
  - Metadaten enthalten Informationen über die zu verwaltenden Daten
  - sie beschreiben also diese Daten (Benutzerdaten) näher hinsichtlich Inhalt, Bedeutung, Nutzung, Integritätsbedingungen, Zugriffskontrolle usw.
  - die Metadaten lassen sich unabhängig vom DBVS beschreiben (siehe internes, konzeptionelles und externes Schema)
  - dadurch erfolgt das „Zuschneidern eines DBS“ auf eine konkrete Einsatzumgebung; die Spezifikation, Verwaltung und Nutzung von Metadaten bildet die Grundlage dafür, dass DBS hochgradig „generische“ Systeme sind
  - Metadaten fallen in allen DBS-Schichten an
  - Metadatenverwaltung, DB-Katalog, Data-Dictionary-System, DD-System, ...
- Transaktionsverwaltung
  - Realisierung der ACID-Eigenschaften
  - Synchronisation
  - Logging/Recovery
  - Integritätssicherung

Bearbeitung einer DB-Anfrage – dynamischer Ablauf

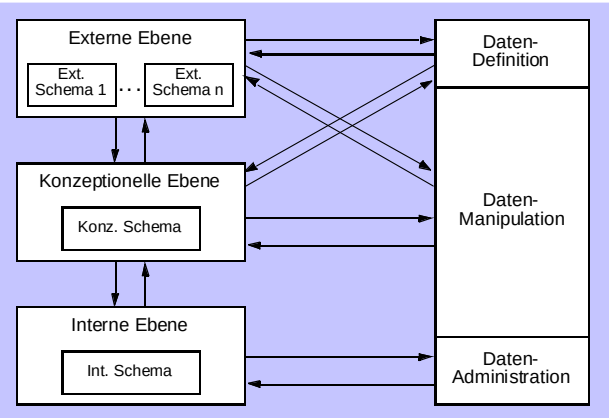


Kapitel 1: Architekturen

- Abbildungshierarchie eines DBS – Schichtenmodell
  - 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
- Drei-Schema-Architektur nach ANSI-SPARC
- Weitere Architekturszenarien
  - Verteilte DBS: Einsatz von Mehrrechner-DBS
  - Schichtenmodelle für Client/Server-DBS
  - Architektur von Transaktionssystemen

Drei-Schema-Architektur nach ANSI-SPARC (1)

- bisher Realisierungssicht (5-Schichtenmodell), nun Benutzungssicht

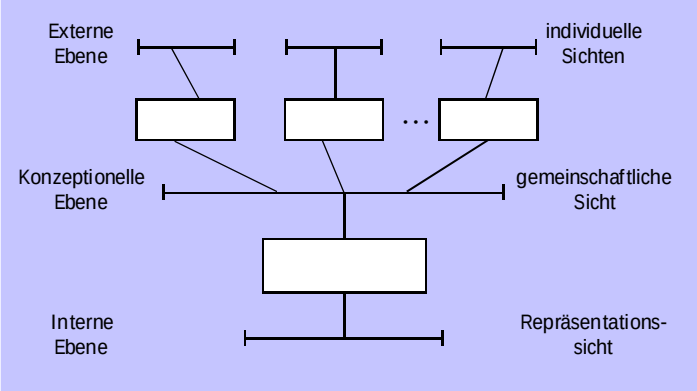


ANSI: American National Standards Institute, SPARC-Komitee:
Study Group on Database Management Systems, http://www.ansi.org

Quelle: [TK78]

Architektur

Drei-Schema-Architektur nach ANSI-SPARC (2)



Architektur

Drei-Schema-Architektur nach ANSI-SPARC (3)

- Konzeptionelles Schema
  - (zeitvariante) globale Struktur; neutrale und redundanzfreie Beschreibung in der Sprache eines spezifischen Datenmodells
  - Beschreibungssprache: DDL (Data Definition Language)
- Externes Schema
  - Definition von zugeschnittenen Sichten auf Teile des konzeptionellen Schemas für spezielle Anwendungen (Benutzer)
  - Sichtenbildung
    - Anpassung der Datentypen an die der Wirtssprache (DBS ist „multi-lingual“)
    - Zugriffsschutz
    - Reduktion der Komplexität
- Internes Schema
  - legt physische Struktur der DB fest (Satzformate, Zugriffspfade etc.)
  - Beschreibungssprache: SSL (Storage Structure Language)

Architektur

Drei-Schema-Architektur nach ANSI-SPARC (4)

- Beispiel

Extern (PL/1)

```
DCL   1 PERS,
      2 PNR CHAR(6),
      3 GEH FIXED BIN(31)
      4 NAME CHAR(20)
```

Extern (COBOL)

```
01    PERSC.
      02   PERSNR PIC X(6).
      02   ABTNR PIC X(4).
```

Konzeptionelles Schema

```
ANGESTELLTER
        ANG_NUMMER      CHARACTER   (6)
        NAME            CHARACTER   (20)
        ABT_NUMMER      CHARACTER   (4)
        GEHALT          NUMERIC     (5)
```

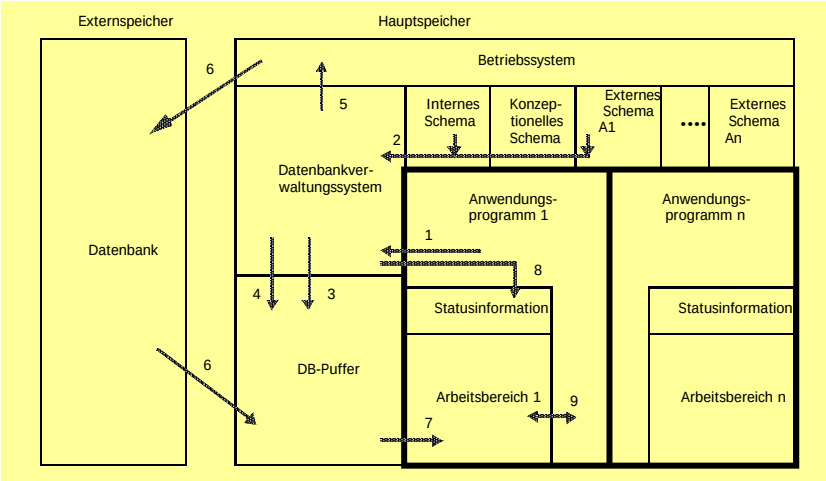
Internes Schema

```
SPEICHERUNG_PERS   LENGTH=40
        PREFIX     TYPE=BYTE(6), OFFSET=0
        PNR        TYPE=BYTE(6), OFFSET=6, INDEX=PNR
        NAME       TYPE=BYTE(20), OFFSET=12
        ANR        TYPE=BYTE(4), OFFSET=32
        GEHALT     TYPE=FULLWORD, OFFSET=36
```


Architektur

Drei-Schema-Architektur nach ANSI-SPARC (5)

- Bearbeitung einer DB-Anfrage – dynamischer Ablauf



Drei-Schema-Architektur nach ANSI-SPARC (6)

- Dynamischer Ablauf (Forts.)
  - Interne Bearbeitungsschritte
    1. **SELECT * FROM PERS' WHERE PNR = '12345'**
    2. Vervollständigen der Verarbeitungsinformation aus Konzeptionellem und Internem Schema;
       Ermittlung der Seiten# (z. B. durch Hashing)
    3. Zugriff auf DB-Puffer: erfolgreich oder
    4. Zugriff auf DB über DB-Pufferverwaltung/Betriebssystem
    5. Durchführen des E/A-Auftrages
    6. Ablegen der Seite im DB-Puffer
    7. Übertragen in Arbeitsbereich
    8. Statusinformation: Return-Code, Cursor-Info
    9. Manipulation mit Anweisungen der Programmiersprache

Kapitel 1: Architekturen

- Abbildungshierarchie eines DBS – Schichtenmodell
  - 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
- Drei-Schema-Architektur nach ANSI-SPARC
- Weitere Architekturszenarien
  - Verteilte DBS: Einsatz von Mehrrechner-DBS und FDBMS
  - Schichtenmodelle für Client/Server-DBS
  - Architektur von Transaktionssystemen

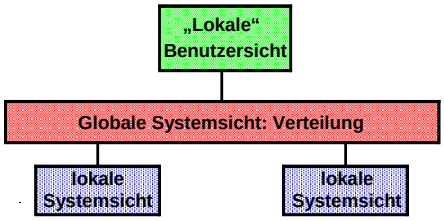
Architektur

Klassifikation verteilter Datenbanksysteme

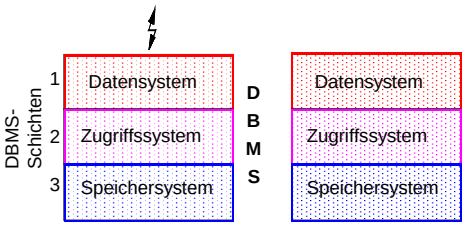
Vorgehensweise bei der Verteilung

- Partitionierung der Daten (ggf. mit Replikation)
- Kommunikation zwischen Prozessen zum Zugriff auf jeweils lokalen Daten

Gewünschte Sichtbarkeit



Homogenes Systemmodell



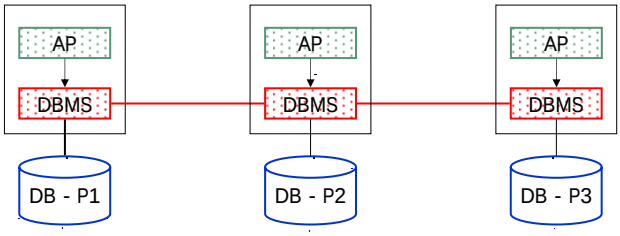
Mehrere (homogene) Realisierungsalternativen

- Globale Systemsicht wird in einer (zusätzlichen) DBMS-Schicht hergestellt
- Verteilung ganzer DML-Befehle bzw. von Teiloperationen (Schicht 1)
- Verteilung von internen Satzoperationen (Schicht 2)
- Verteilung von Seitenzugriffen (Schicht 3)

Quelle: [Ra94]

Architektur

Mehrrechner-DBS



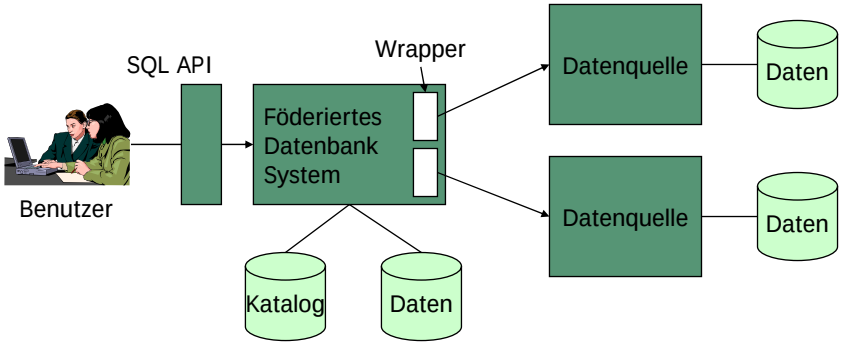
Architekturklassen

1. **DB-Partitionierung** (***Shared Nothing***, VDBS)
   - Jeder Knoten besitzt volle Funktionalität und verwaltet eine DB-Partition
   - 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

Architektur

Föderierte Datenbanksysteme (FDBS)

- Transparenter Zugriff auf mehrere heterogene und weitgehend autonome Datenquellen
- Vollständiges DBMS
  - Kompensiert Funktionalität, die in einzelnen Datenquellen fehlt
  - globale Anfrageoptimierung



Architektur

Schichtenmodelle für Client/Server-DBS

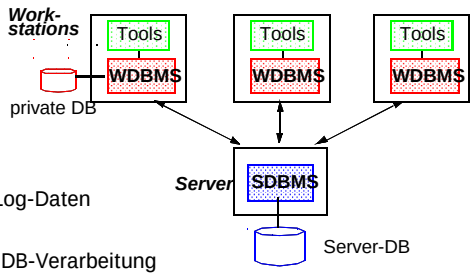
- Entscheidend für die DB-bezogene Leistungsfähigkeit:
  - Art und Komplexität der DB-Operationen (mengenorientiert, navigierend)
  - Nutzung der Referenzlokalität bei den Datenzugriffen
- Bisheriges Verarbeitungskonzept
  - Bei allen Architekturvorschlägen wurde ein „Verschicken der Anfragen“ zum (Server)-DBMS unterstellt (query shipping approach):
  - Die DML-Anweisung wird zum Server geschickt
  - Nach ihrer Verarbeitung wird das Ergebnis der AW (Client) zur Verfügung gestellt
- Wichtige Eigenschaft:
  - Es gibt keine Replikation von DB-Daten
  - keine Nutzung vorhandener Referenzlokalität im Client

Schichtenmodelle für Client/Server-DBS

- Weiteres Verarbeitungskonzept: Folgende Vorgehensweise wird von den meisten Objektorientierten DBS (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
  - Achtung: Die Komplexität der Anfragen ist stark eingeschränkt
    - Typisch sind Navigationsoperationen
    - Anfragen mit einfachen Suchargumenten (SSA: simple search arguments) werden manchmal unterstützt, jedoch keine Verbundoperationen, Aggregationen, ...
    - Dieses Konzept ist vor allem bei hoher Referenzlokalität vorteilhaft. Es wird durch das Check-out/Check-in-Konzept genutzt.

Schichtenmodelle für Client/Server-DBS

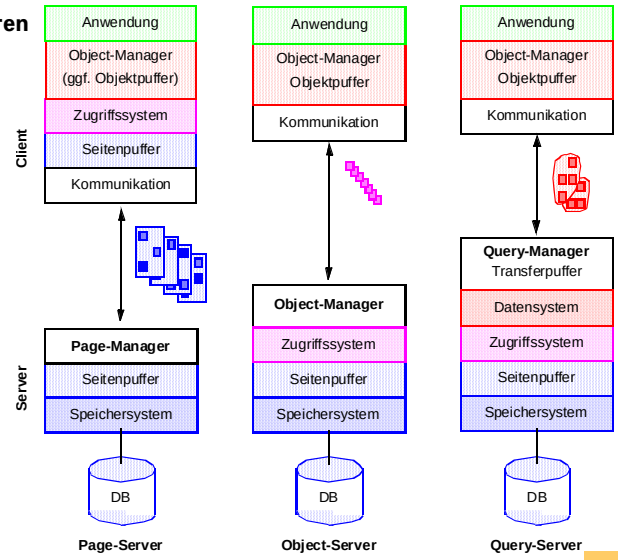
- Vorteile des „Verschickens von Daten“
  - Lokale Datenpufferung reduziert die Interaktionen zwischen Client und Server (Verminderung der Netzbelastung; Vermeidung von wiederholten Server-Anfragen)
  - Aggregierte Rechnerleistung und die insgesamt verfügbare Speicherkapazität des Systems erhöht sich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



- Client/Server-Architektur: Prinzip
  - Lokale Platten für private DB und Log-Daten
  - Steigerung der Leistungsfähigkeit:
  - verteiltes Server-System, parallele DB-Verarbeitung

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!

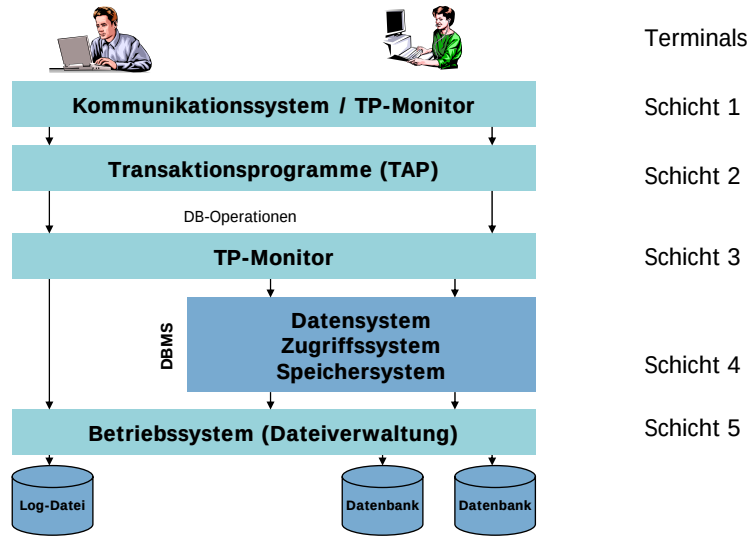


Transaktionsverarbeitung

- Drei Aspekte:
  - Mit einer Transaktion (TA) wird ein Vorgang einer Anwendung in einem Rechnersystem abgewickelt. Ein solcher Vorgang bildet typischerweise einen nicht-trivialen Arbeitsschritt in betrieblichen Abläufen.
  - Eine Online-Transaktion ist die Ausführung eines Programms, das mit Hilfe von Zugriffen auf eine gemeinsam genutzte Datenbank eine Anwendungsfunktion erfüllt.
  - Eine Transaktion ist eine ununterbrechbare Folge von DB-Operationen, die die Datenbank von einem logisch konsistenten Zustand in einen anderen logisch konsistenten Zustand überführt.

Anwendungs-bereich	Exemplarische Transaktionen
Banken	Kontenbuchungen, Zinsberechnungen
Aktienhandel	Kauf von n Aktien einer bestimmten AG
Versicherungen	Versicherungsprämie bezahlen
Lagerverwaltung	Wareneingang registrieren, Warenbestand anpassen
Handel	Verkauf registrieren
Verwaltung	KFZ-Zulassung, An- und Abmeldung
Electronic Commerce	Online-Bestellung, Suche in Online-Katalog
Telekommunikation	Verbindungsnachweise
...	

Transaktionssysteme



Quelle: [Ra93]

Zusammenfassung

- Allgemeine Aspekte des Schichtenmodells
  - Schichtenmodell ist allgemeines Erklärungsmodell für die DBS-Realisierung
  - Schichtenbildung lässt sich zweckorientiert verfeinern/vergröbern:
    - Anwendbarkeit für Verteilte DBS, Client/Server-DBS, ...
  - Entwurf geeigneter Schnittstellen erfordert große Implementierungserfahrung
  - Konkrete Implementierungen verletzen manchmal die Isolation der Schichtenbildung aus Leistungsgründen (d.h. 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

Ergänzende Literatur zu diesem Kapitel

[Hä05] Härder, T.: DBMS Architecture - The Layer Model and its Evolution. In: Datenbank-Spektrum, Heft 13, Mai 2005, dpunkt-Verlag.

[Ri07] Ritter, N.: Verteilte und föderierte Datenbanksysteme. Kapitel 12 in: Kudraß (Hrsg.): Taschenbuch Datenbanken, Hanser, 2007.

[Ra93] Rahm, E.: Hochleistungs-Transaktionssysteme, vieweg, 1993

[Ra94] Rahm, E.: Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenbankverarbeitung. Addison-Wesley, 1994.

[TK78] Tsichritzis, D. C., Klug, A.: The ANSI/X3/Sparc DBMS Framework Report of the Study Group on Database Management Systems. In: Information Systems 3:3, 1978.