Ludwig Maximilians Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung

Datenbanksysteme I

Wintersemester 2009/2010

Kapitel 1: Einführung

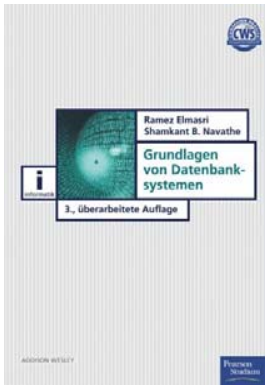
Vorlesung: PD Dr. Peer Kröger, Dr. Matthias Renz
Übungen: Andreas Zuefle

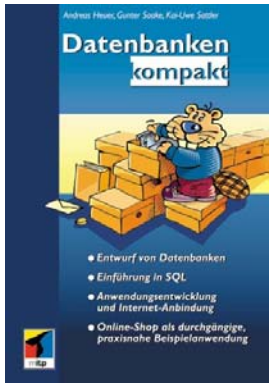
http://www.dbs.ifi.lmu.de/cms/Datenbanksysteme_I

Literaturliste

Die Vorlesung orientiert sich nicht an einem bestimmten Lehrbuch. Empfehlenswert sind aber u.a.:

- A. Kemper, A. Eickler:
  Datenbanksysteme
  Oldenbourg, 5. Auflage (2004). 39,80 €
- R. Elmasri, S. B. Navathe:
  Grundlage von Datenbanksystemen
  Pearson Studium, 3. Auflage (2004). 39,95 €
- A. Heuer, G. Saake, K.-U. Sattler:
  Datenbanken kompakt
  mitp, 2. Auflage (2003). 19,95 €
- A. Heuer, G. Saake:
  Datenbanken: Konzepte und Sprachen
  mitp, 2. Auflage (2000). 35,28 €
- R. Ramakrishnan, J. Gehrke:
  Database Management Systems
  McGraw Hill, 3. Auflage (2002).





Das Team

Vorlesung

PD Dr. Peer Kröger

Dr. Matthias Renz

Übungen

Andreas Züfle

Erich Schubert

Tutoren: Michael Mirwaldt und Nina Hubig

Wovon handelt die Vorlesung?

- Bisher (Einführungsvorlesung):
  Nur Betrachtung des Arbeitsspeichers.
  Objekte werden im Arbeitsspeicher erzeugt und nach dem Programmlauf wieder entfernt

- Warum ist dies nicht ausreichend?
  - Viele Anwendungen müssen Daten *permanent* speichern
  - Arbeitsspeicher ist häufig *nicht groß genug*, um z.B. alle Kundendaten einer Bank oder Patientendaten einer Klinik zu speichern

Permanente Datenspeicherung

- Daten können auf dem sog. *Externspeicher* (auch Festplatte genannt) permanent gespeichert werden

- Arbeitsspeicher:
  - rein elektronisch (Transistoren und Kondensatoren)
  - flüchtig
  - schnell: 10 ns/Zugriff *
  - wahlfreier Zugriff
  - teuer: 100-150 € für 1 GByte*

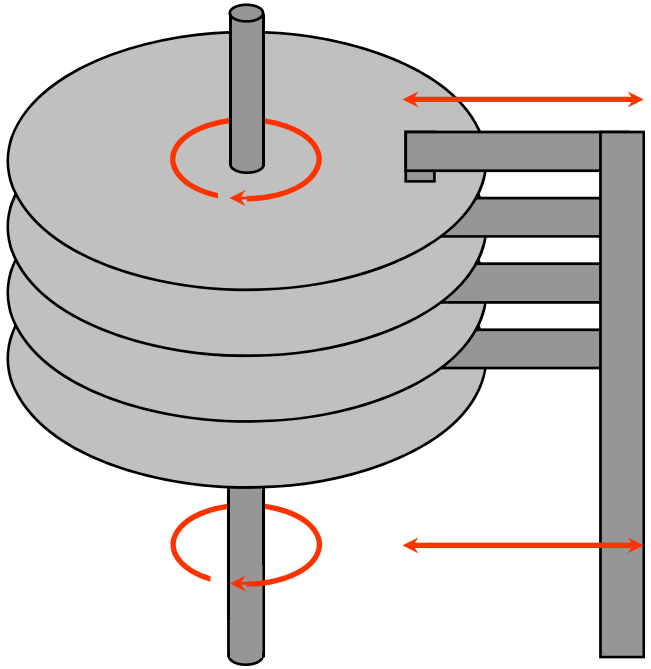
Externspeicher:

- Speicherung auf magnetisierbaren Platten (rotierend)
- nicht flüchtig
- langsam: 5 ms/Zugriff *
- blockweiser Zugriff
- wesentlich billiger: 100 € für ca. 200 GByte*

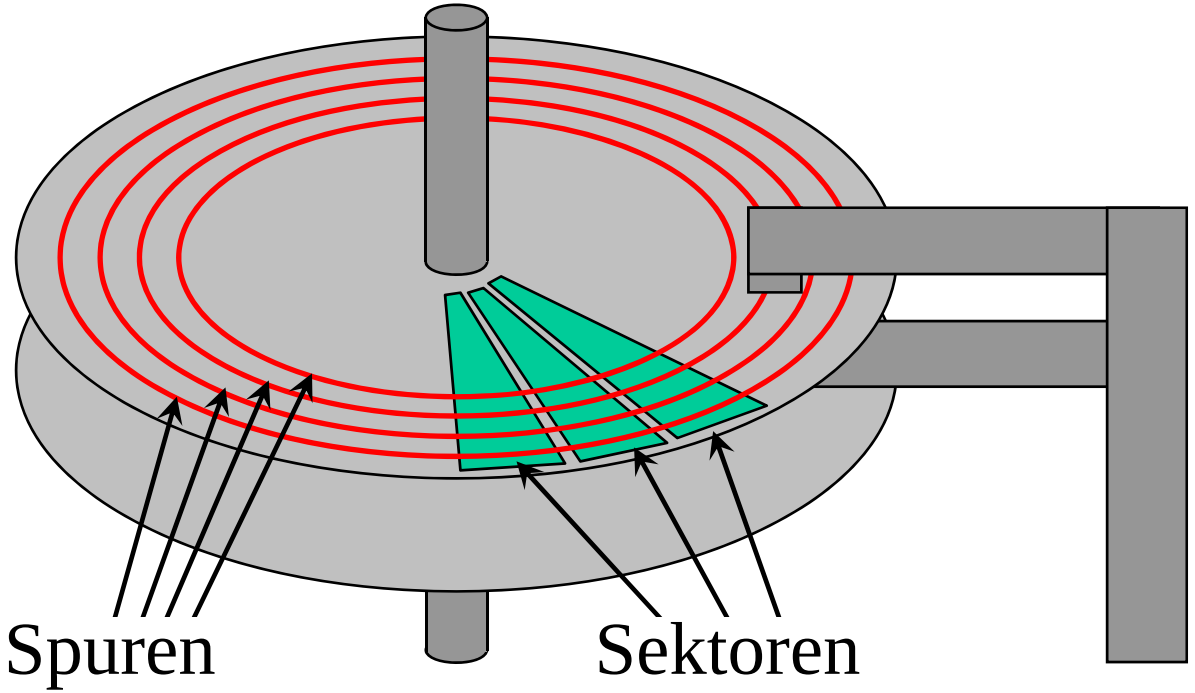
*Oktober 2005 .

Aufbau einer Festplatte

- Mehrere magnetisierbare *Platten* rotieren z.B. mit 7.200 Umdrehungen* pro Minute um eine gemeinsame Achse (*z. Z. 5400, 7200, 10000 upm)
- Ein Kamm mit je zwei *Schreib-/Leseköpfen* pro Platte (unten/oben) bewegt sich in radialer Richtung.



Einteilung der Plattenoberflächen



- (interne) Adressierung einer Information:
  [Platten-Nr | Oberfl.-Nr | Spur-Nr | Sektor-Nr | Byte-Nr]
- Berechnung der Kapazität:

  # Platten * 2 * # Spuren * # Sektoren * Bytes pro Sektor

Lesen/Schreiben eines Sektors

- Positionieren des Kamms mit den Schreib-/Leseköpfen auf der Spur
- Warten bis die Platte so weit rotiert ist, dass der Beginn des richtigen Sektors unter dem Schreib-/Lesekopf liegt
- Übertragung der Information von der Platte in den Arbeitsspeicher (bzw. umgekehrt)

Achtung:

Es ist aus technischen Gründen nicht möglich, einzelne Bytes zu lesen bzw. zu schreiben, sondern mindestens einen ganzen Sektor

Speicherung in Dateien

- Adressierung mit Platten-Nr., Oberfl.-Nr. usw. für den Benutzer nicht sichtbar

- Arbeit mit Dateien:
  - Dateinamen
  - Verzeichnishierarchien
  - Die Speicherzellen einer Datei sind byteweise von 0 aufsteigend durchnummeriert.
  - Die Ein-/Ausgabe in Dateien wird gepuffert, damit nicht der Programmierer verantwortlich ist, immer ganze Sektoren zu schreiben/lesen.

Beispiel: Dateizugriff in Java

```
public static void main (String[] args) {
  try {
    RandomAccessFile f1 = new
      RandomAccessFile("test.file","rw");
    int c = f1.read() ;
    long new_position = .... ;
    f1.seek (new_position) ;
    f1.write (c) ;
    f1.close () ;
  } catch (IOException e) {
    System.out.println ("Fehler: " + e) ;
  }
}
```

Datei-Handle

Datei öffnen

ein Byte lesen

auf neue Position

ein Byte schreiben

Datei schließen

Fehlerbehandlung

Beispiel: Dateizugriff in Java

- Werden die Objekte einer Applikation in eine Datei geschrieben, ist das Dateiformat vom Programmierer festzulegen:

- Wo findet man dieses Datei-Schema im Quelltext z.B. des Java-Programms ?

  Das Dateischema wird nicht explizit durch den Quelltext beschrieben, sondern implizit in den Ein-/Auslese-Prozeduren der Datei

Logische Datenabhängigkeit

- Konsequenzen bei einer Änderung des Dateiformates (z.B. durch zusätzliche Objektattribute in einer neuen Programmversion):
  - Alte Datendateien können nicht mehr verwendet werden oder müssen z.B. durch extra Programme konvertiert werden
  - Die Änderung muss in allen Programmen nachgeführt werden, die mit den Daten arbeiten, auch in solchen, die logisch von Änderung gar nicht betroffen sind

Physische Datenabhängigkeit

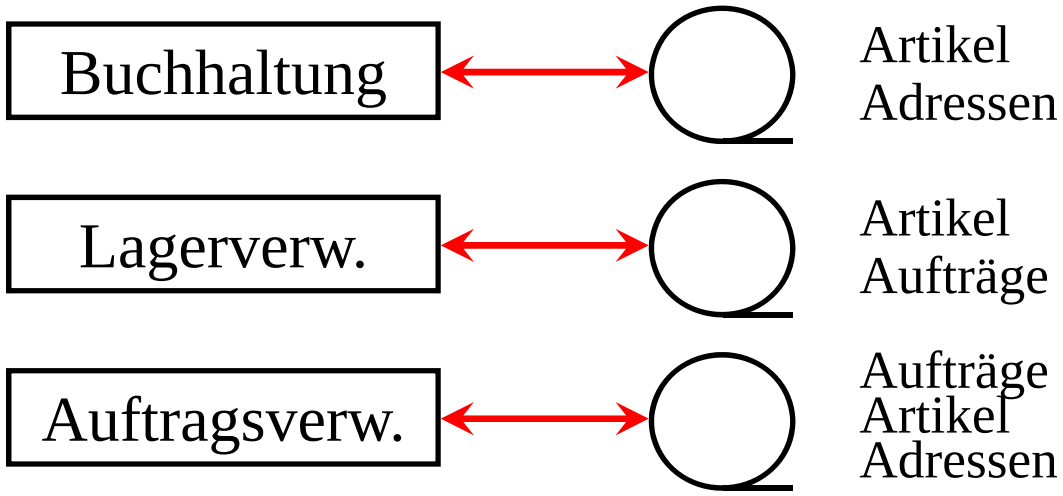
- Meist werden die Datensätze anhand ihrer Position adressiert/separiert:
  - z.B. jeder Satz hat 22 Zeichen:
  - 1. Satz: Adresse 0; 2. Satz: Adresse 22 usw.
- Suche gemäß bestimmten Attributwerten (z.B. Namen des Kunden) muss im Programm codiert werden
- Soll die Datei z.B. mit einem Suchbaum unterstützt werden, dann gleiche Konsequenzen wie bei logischer Änderung

Informationssysteme

- Große Software-Systeme
- Viele einzelne Programme
- Programme arbeiten teils mit gemeinsamen Daten, teils mit unterschiedlichen
- Beispiele für die Programme:
  - Buchhaltung: Artikel- und Adressinformation
  - Lagerverwaltung: Artikel und Aufträge
  - Auftragsverwaltung.: Aufträge, Artikel, Adressen
  - CAD-System: Artikel, techn. Daten, Bausteine
  - Produktion, Bestelleingang, Kalkulation: ...

Redundanz

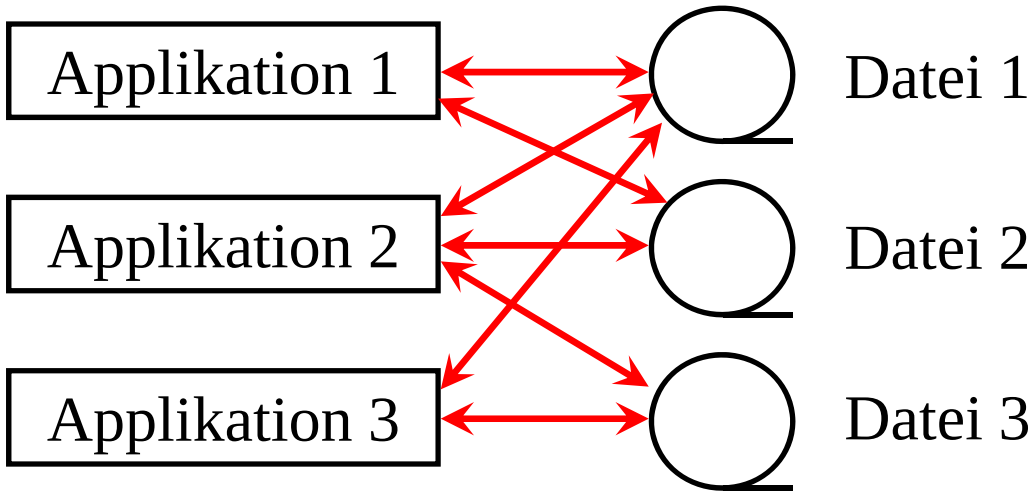
- Daten werden meist mehrfach gespeichert



- Konsequenz: u.a. *Änderungs-Anomalien*
  Bei Änderung einer Adresse müssen viele Dateien nach den Einträgen durchsucht werden
  (hierzu später mehr)

Schnittstellenproblematik

- Alternative Implementierung



- Nachteile:
  - unübersichtlich
  - bei logischen oder physischen Änderungen des Dateischemas müssen viele Programme angepasst werden

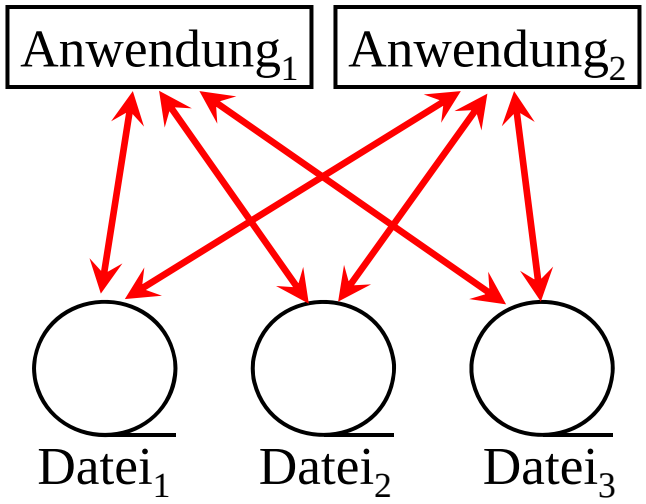
Weitere Probleme von Dateien

- In großen Informationssystemen arbeiten viele Benutzer gleichzeitig mit den Daten: Dateisysteme bieten zu wenige Möglichkeiten, um diese Zugriffe zu synchronisieren
- Dateisysteme schützen nicht in ausreichendem Maß vor Datenverlust im Fall von Systemabstürzen und Defekten
- Dateisysteme bieten nur unflexible Zugriffskontrolle (Datenschutz)

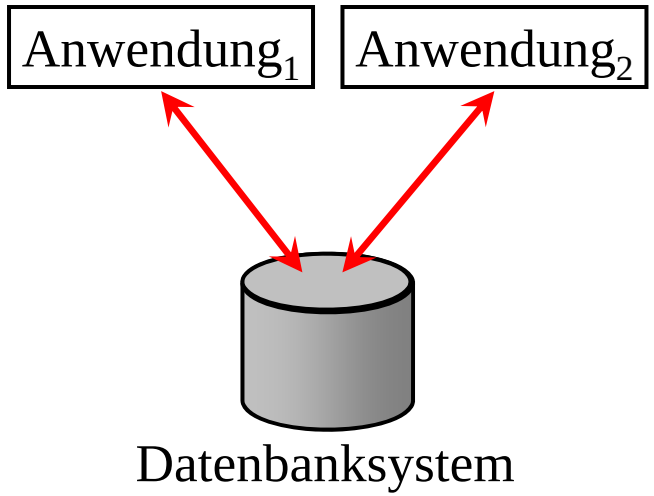
Von Dateien zu Datenbanken

- Um diese Probleme mit einheitlichem Konzept zu behandeln, setzt man Datenbanken ein:

Mit Dateisystem:



Mit Datenbanksystem:



Komponenten eines DBS

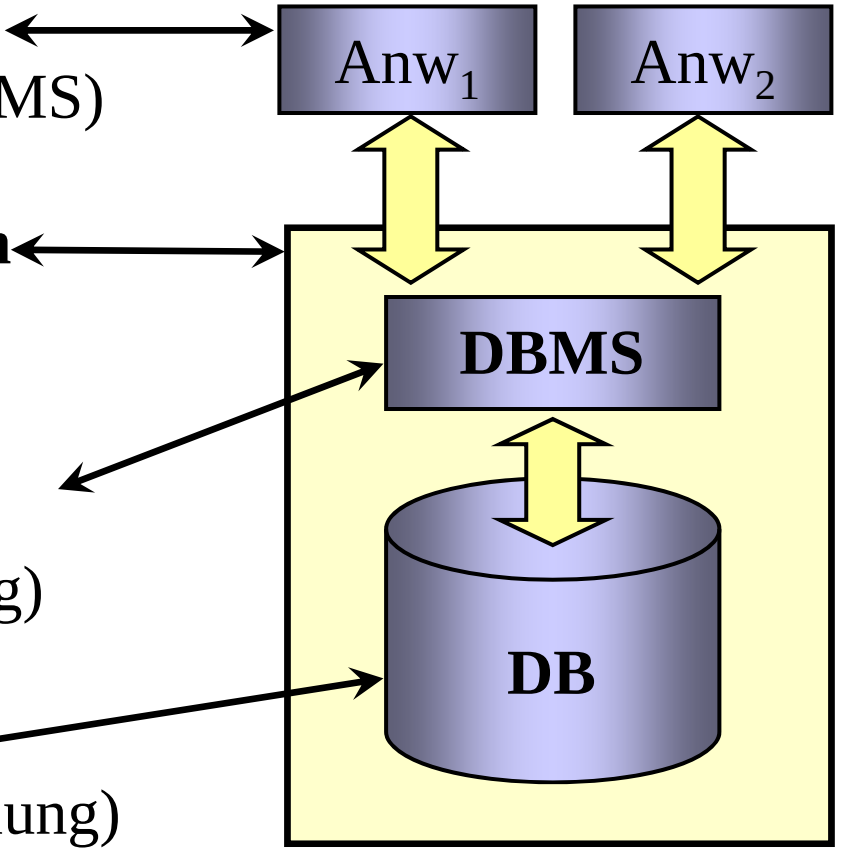
Man unterscheidet zwischen...

DB-Anwendungen
(kommunizieren mit DBMS)

DBS: Datenbanksystem
(DB + DBMS)

DBMS: Datenbank-Management-System
(Software zur Verwaltung)

DB: Datenbank
(eigentliche Datensammlung)



Typische Einsatzbereiche

- Im betriebswirtschaftlichen Bereich:
  - Banken (Kontoführung)
  - Buchhaltung und Rechnungswesen
  - Flugbuchungssysteme
  - Telefongesellschaften (Abrechnung)
  - Lagerverwaltung
- Im technisch-wissenschaftlichen Bereich:
  - CAD/CAM/CIM
  - Medizin
  - Molekularbiologie (Gendatenbanken)

Aufgaben eines DBS

Primäre Aufgabe eines DBS ist die ...

- Beschreibung
- Speicherung und Pflege
- und Wiedergewinnung

umfangreicher Datenmengen, die von verschiedenen Anwendungsprogammen dauerhaft (persistent) genutzt werden

Anforderungen an ein DBS

Liste von 9 Anforderungen (Edgar F. Codd, 1982)

- Integration
  Einheitliche Verwaltung *aller* von Anwendungen benötigten Daten.
  Redundanzfreie Datenhaltung des gesamten Datenbestandes
- Operationen
  Operationen zur Speicherung, zur Recherche und zur Manipulation der Daten müssen vorhanden sein
- Data Dictionary
  Ein Katalog erlaubt Zugriffe auf die Beschreibung der Daten
- Benutzersichten
  Für unterschiedliche Anwendungen unterschiedliche Sicht auf den Bestand
- Konsistenzüberwachung
  Das DBMS überwacht die Korrektheit der Daten bei Änderungen

Anforderungen an ein DBS

- Zugriffskontrolle
  Ausschluss unautorisierter Zugriffe
- Transaktionen
  Zusammenfassung einer Folge von Änderungsoperationen zu einer Einheit, deren Effekt bei Erfolg permanent in DB gespeichert wird
- Synchronisation
  Arbeiten mehrere Benutzer gleichzeitig mit der Datenbank dann vermeidet das DBMS unbeabsichtigte gegenseitige Beeinflussungen
- Datensicherung
  Nach Systemfehlern (d.h. Absturz) oder Medienfehlern (defekte Festplatte) wird die Wiederherstellung ermöglicht (im Ggs. zu Datei-Backup Rekonstruktion des Zustands der letzten erfolgreichen TA)

Inhalte von Datenbanken

Man unterscheidet zwei Ebenen:

- Intensionale Ebene: **Datenbankschema**
  - beschreibt *möglichen* Inhalt der DB
  - Struktur- und Typinformation der Daten (Metadaten)
  - Art der Beschreibung vorgegeben durch Datenmodell
  - Änderungen möglich, aber selten (Schema-Evolution)
- Extensionale Ebene: **Ausprägung der Datenbank**
  - *tatsächlicher* Inhalt der DB (DB-Zustand)
  - Objektinformation, Attributwerte
  - Struktur vorgegeben durch Datenbankschema
  - Änderungen häufig (Flugbuchung: 10000 TA/min)

Inhalte von Datenbanken

Einfaches Beispiel:

- Schema:

- DB-Zustand:

F	r	a	n	k	l	i	n			A	r	e	t	h	a			1	9	4	2
R	i	t	c	h	i	e				L	i	o	n	e	l			1	9	4	9

- Nicht nur DB-Zustand, sondern auch DB-Schema wird in DB gespeichert.
- Vorteil: Sicherstellung der Korrektheit der DB

Vergleich bzgl. des Schemas

- Datenbanken
  - Explizit modelliert (Textdokument oder grafisch)
  - In Datenbank abgespeichert
  - Benutzer kann Schema-Informationen auch aus der Datenbank ermitteln: *Data Dictionary*, *Metadaten*
  - DBMS überwacht Übereinstimmung zwischen DB-Schema und DB-Zustand
  - Änderung des Schemas wird durch DBMS unterstützt (Schema-Evolution, Migration)

Vergleich bzgl. des Schemas

- Dateien
  - Kein Zwang, das Schema explizit zu modellieren
  - Schema implizit in den Prozeduren zum Ein-/Auslesen
  - Schema gehört zur Programm-Dokumentation
  - oder es muss aus Programmcode herausgelesen werden.
    Hacker-Jargon: Entwickler-Doku, RTFC (read the f...ing code)
  - Fehler in den Ein-/Auslese-Prozeduren können dazu führen, dass gesamter Datenbestand unbrauchbar wird:

F	r	a	n	k	l	i	n			A	r	e	t	h	a		1	9	4	2	R
i	t	c	h	i	e				L	i	o	n	e	l			1	9	4	9	

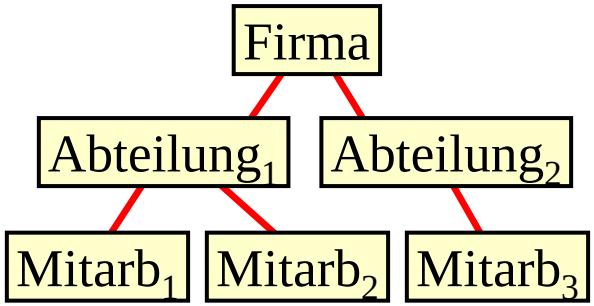
  - Bei Schema-Änderung müssen Migrations-Prozeduren programmiert werden, um bestehende Dateien auf das neue Format umzustellen

Datenmodelle

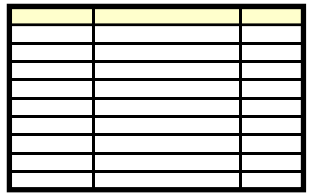
- Formalismen zur Beschreibung des DB-Schemas
  - Objekte der Datenbank
  - Beziehungen zwischen verschiedenen Objekten
  - Integritätsbedingungen

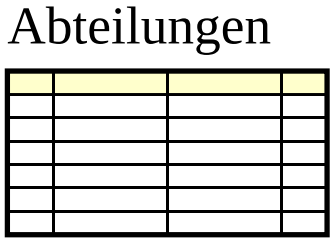
- Verschiedene Datenmodelle unterscheiden sich in der Art und Weise, wie Objekte und Beziehungen dargestellt werden:

Hierarchisch: Baum



Relational: Tabellen





Datenmodelle

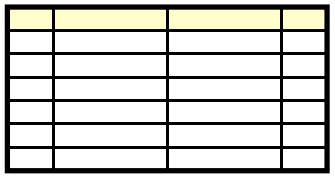
- Weitere Unterschiede zwischen Datenmodellen:
  - angebotene Operationen (insbes. zur Recherche)
  - Integritätsbedingungen

- Die wichtigsten Datenmodelle sind:
  - Hierarchisches Datenmodell
  - Netzwerk-Datenmodell
  - Relationales Datenmodell
  - Objektorientiertes Datenmodell
  - Objekt-relationales Datenmodell

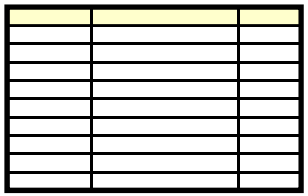
Relationales Modell

- Alle Informationen werden in Form von Tabellen gespeichert
- Die Datenbank besteht aus einer Menge von Tabellen (Relationen)
- Im Beispiel enthält die Tabelle „Mitarbeiter“ Informationen über die Mitarbeiter des Betriebes
- In jeder Zeile (Tupel) Information über einen
- Mitarbeiter (die Zeilen sind strukturell gleich)
- Die Spalten (Attribute) haben einen Namen (z.B. *Personalnr*, *Name*, *Vorname*, *Geburtsdatum*, etc.).
  Sie sind strukturell (Typ, Anzahl Zeichen) verschieden.

Abteilungen



Mitarbeiter



Relationales Modell

- Die Attribute der Tupel haben primitive Datentypen wie z.B. String, Integer oder Date
- Komplexe Sachverhalte werden durch Verknüpfung mehrerer Tabellen dargestellt
- Beispiel:

Mitarbeiter

PNr	Name	Vorname	ANr
001	Huber	Erwin	01
002	Mayer	Hugo	01
003	Müller	Anton	02

Abteilungen

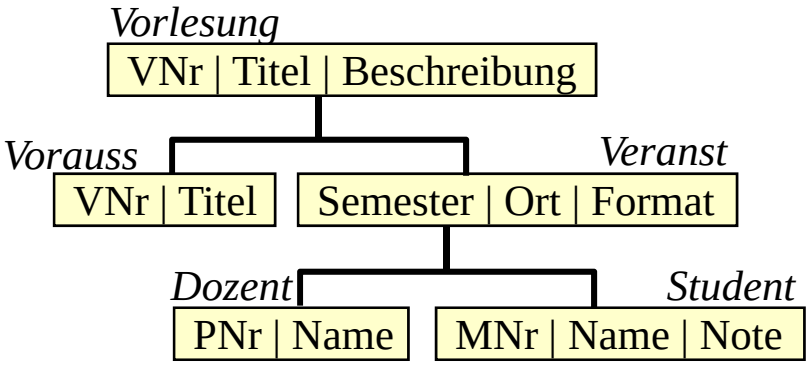
ANr	Abteilungsname
01	Buchhaltung
02	Produktion
03	Marketing

- Später ausführliche Behandlung

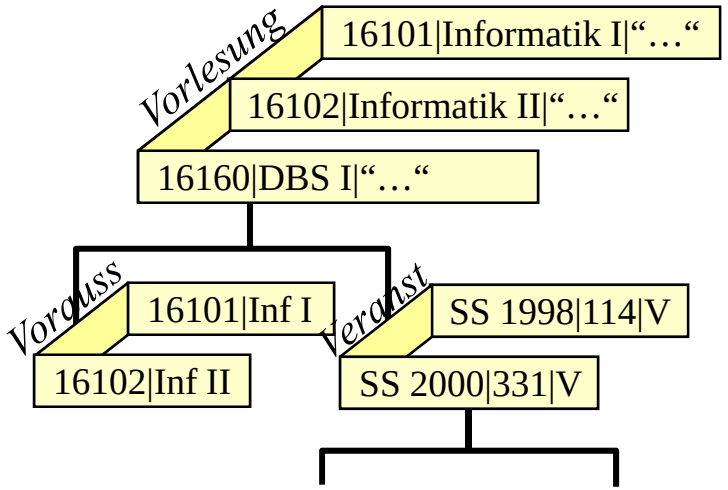
Hierarchisches Datenmodell

- Schema + Daten werden durch Baum strukturiert
- Der gesamte Datenbestand muss hierarchisch repräsentiert werden (oft schwierig)
- Beispiel Lehrveranstaltungen:

Schema:



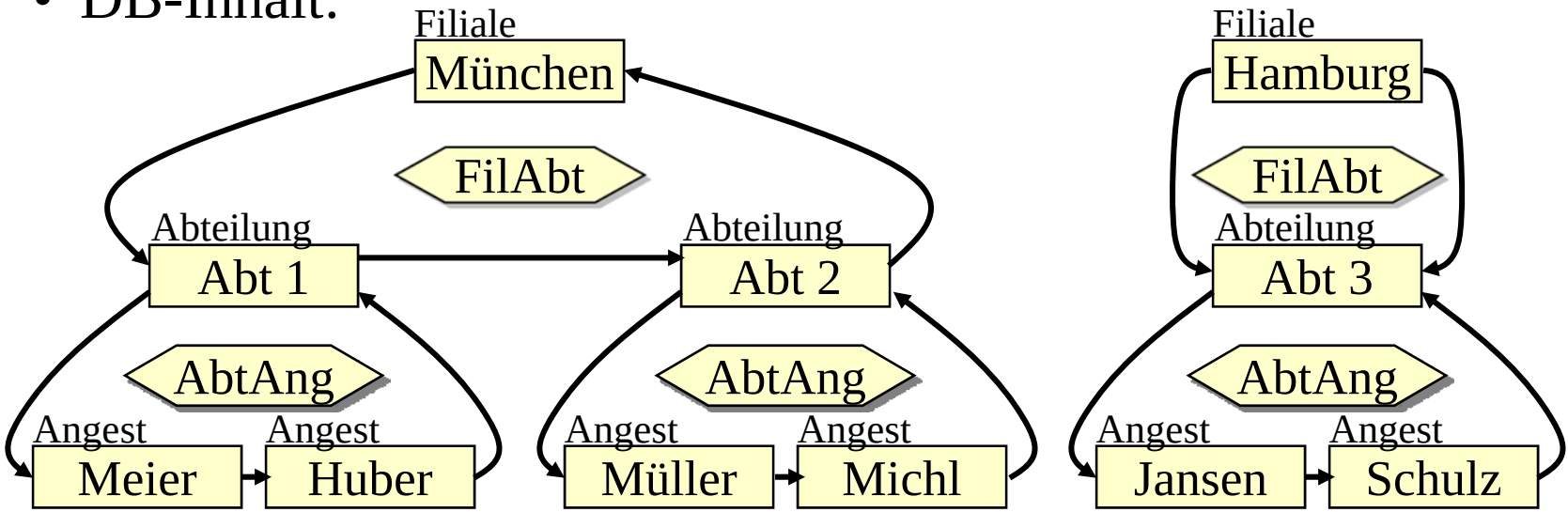
Inhalt:



Netzwerk-Datenmodell

- Schema und Daten werden durch Graphen (Netzwerke) repräsentiert
- Schema:

- DB-Inhalt:



Objekt-Orientiertes Datenmodell

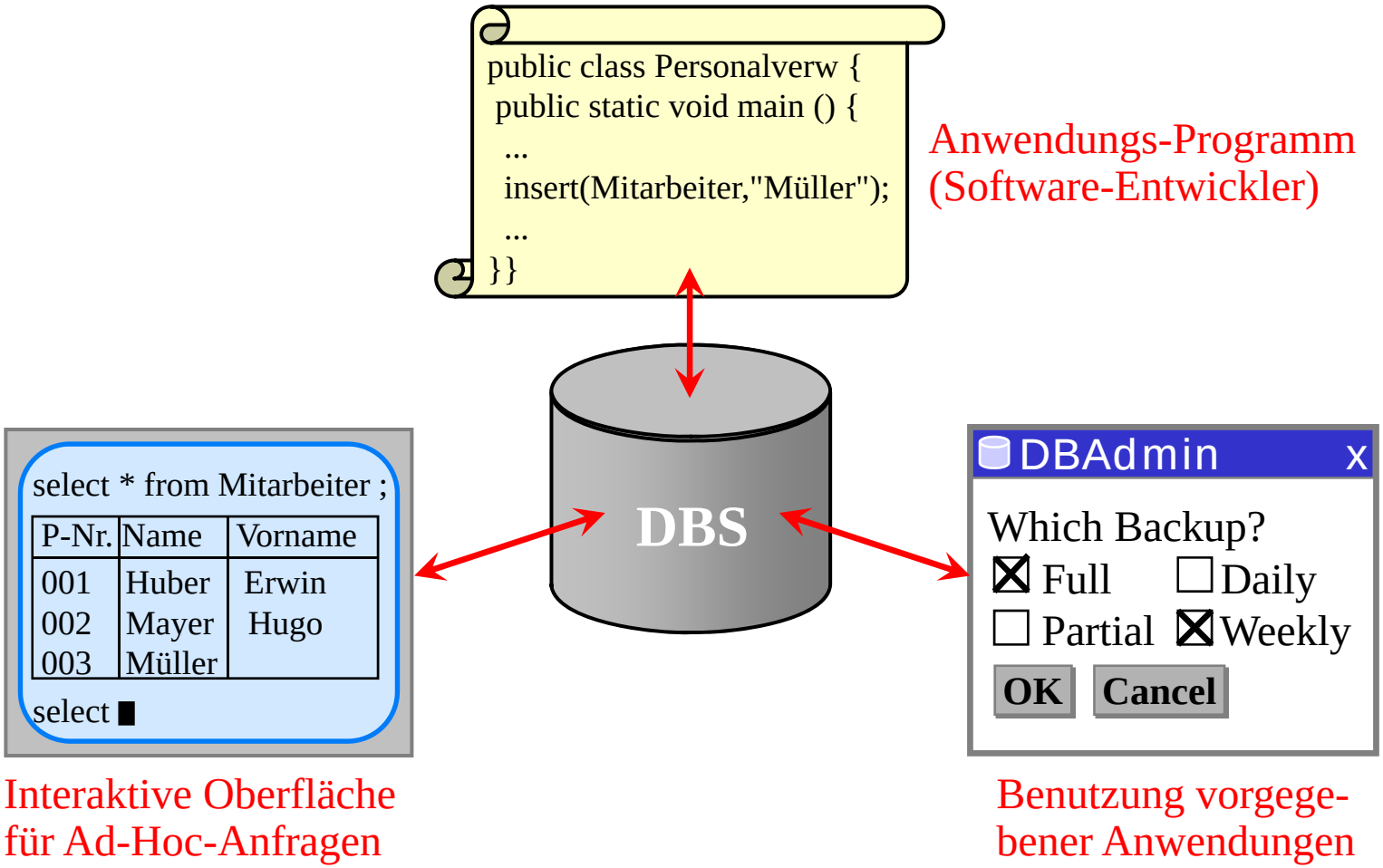
- In der Datenbank werden Objekte, d.h. Ausprägungen von Klassen, die zueinander in verschiedenen Beziehungen stehen (z.B. auch Vererbungsbeziehung), persistent gespeichert.
- Rein objektorientierte Datenbanken haben sich kaum durchgesetzt
- Relationale Datenbanken haben die Idee aufgenommen und erlauben jetzt auch Speicherung komplexer Objekte (incl. Vererbung) in Relationen

→ Objekt-Relationale Datenbanken

Produkte

- (Objekt-) Relationale Datenbanken:
  - Oracle (Marktführer)
  - IBM DB2
  - Microsoft SQL Server
  - MySQL (Open Source)
  - PostgreSQL (Open Source)
  - keine vollwertigen Datenbanksysteme: dBase, FoxPro, ACCESS
- Nicht-Relationale Datenbanken
  - IMS: Hierarchisches Datenbanksystem (IBM)
  - UDS: Netzwerk-Datenbanksystem (Siemens)
  - Verschiedene objektorientierte DB-Produkte

Verwendung eines DBS

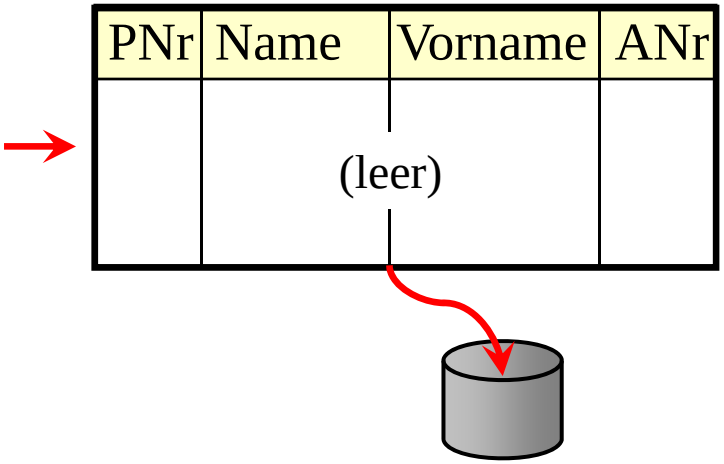


Aus technischer Sicht ist die interaktive Oberfläche ebenfalls ein Anwendungsprogramm, das auf dem DBS aufsetzt

Datenbank-Sprachen

- Data Definition Language (DDL)
  - Deklarationen zur Beschreibung des Schemas
  - Bei relationalen Datenbanken:
    Anlegen und Löschen von Tabellen, Integritätsbedingungen usw.

```
CREATE TABLE Mitarbeiter
  (
    PNr      NUMBER (3),
    Name     CHAR (20),
    Vorname  CHAR (20),
    Anr      NUMBER (2)
  )
```



Datenbank-Sprachen

- Data Manipulation Language (DML)
  - Anweisungen zum Arbeiten mit den Daten in der Datenbank (Datenbank-Zustand)
  - lässt sich weiter unterteilen in Konstrukte
    - zum reinen Lesen der DB (Anfragesprache)
    - zum Manipulieren (Einfügen, Ändern, Löschen) des Datenbankzustands
  - Beispiel: SQL für relationale Datenbanken:

```
SELECT *
FROM Mitarbeiter
WHERE Name = 'Müller'
```

Datenbank-Sprachen

- Wird das folgende Statement (Mitarbeiter-Tab. S. 29)

```
SELECT *
FROM Mitarbeiter
WHERE ANr = 01
```

in die interaktive DB-Schnittstelle eingegeben, dann ermittelt das Datenbanksystem alle Mitarbeiter, die in der Buchhaltungsabteilung (ANr = 01) arbeiten:

PNr	Name	Vorname	ANr
001	Huber	Erwin	01
002	Mayer	Hugo	01

Ergebnis einer Anfrage ist immer eine (neue) Tabelle

Verbindung zur Applikation

- Verwendung einer Programmierbibliothek
  - Dem Programmierer wird eine Bibliothek von Prozeduren/Funktionen zur Verfügung gestellt (Application Programming Interface, API)
  - DDL/DML-Anweisungen als Parameter übergeben
  - Beispiele:
    - OCI: Oracle Call Interface
    - ODBC: Open Database Connectivity
      gemeinsame Schnittstelle an alle Datenbanksysteme
    - JDBC: Java Database Connectivity

Verbindung zur Applikation

- Beispiel: JDBC

```
String q    = “SELECT * FROM Mitarbeiter “ +
              “WHERE Name = 'Müller'“ ;
Statement s = con.createStatement () ;
ResultSet r = s.executeQuery (q) ;
```

- Die Ergebnistabelle wird an das Java-Programm übergeben.
- Ergebnis-Tupel können dort verarbeitet werden

Verbindung zur Applikation

- Einbettung in eine Wirtssprache
  - DDL/DML-Anweisungen gleichberechtigt neben anderen Sprachkonstrukten
  - Ein eigener Übersetzer (Precompiler) wird benötigt, um die Konstrukte in API-Aufrufe zu übersetzen
  - Beispiele:
    - Embedded SQL für verschiedene Wirtssprachen, z.B. C, C++, COBOL, usw.
    - SQLJ oder JSQL für Java

Verbindung zur Applikation

- Beispiel in SQLJ:

```
public static void main () {
  System.out.println (“Hallöchen“) ;
  #sql {SELECT * FROM Mitarbeiter
        WHERE Name = 'Müller'}
  ...
}
```

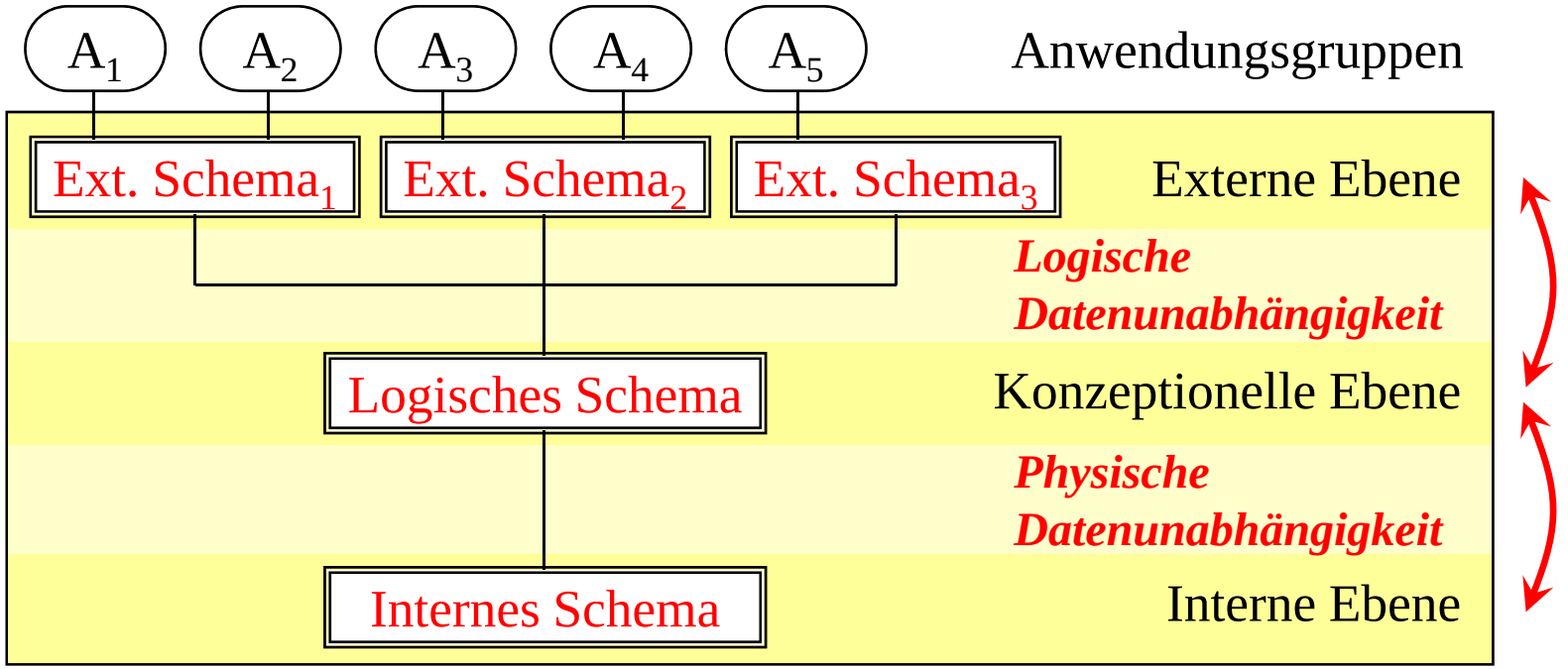
- Die Ergebnistabelle wird an das Java-Programm übergeben.
- Ergebnis-Tupel können dort verarbeitet werden

Architektur eines DBS

Drei-Ebenen-Architektur zur Realisierung von

- **physischer**
- **und logischer**

Datenunabhängigkeit (nach ANSI/SPARC)



Konzeptionelle Ebene

- Logische Gesamtsicht *aller* Daten der DB unabhängig von den einzelnen Applikationen
- Niedergelegt in konzeptionellem (logischem) Schema
- Ergebnis des (logischen) Datenbank-Entwurfs (siehe Kapitel 6)
- Beschreibung aller Objekttypen und Beziehungen
- Keine Details der Speicherung
- Formuliert im Datenmodell des Datenbanksystems
- Spezifiziert mit Hilfe einer Daten-Definitionssprache (Data Definition Language, DDL)

Externe Ebene

- Sammlung der individuellen Sichten aller Benutzer- bzw. Anwendungsgruppen in mehreren externen Schemata
- Ein Benutzer soll keine Daten sehen, die er nicht sehen will (Übersichtlichkeit) oder nicht sehen soll (Datenschutz)
  - Beispiel: Das Klinik-Pflegepersonal benötigt andere Aufbereitung der Daten als die Buchhaltung
- Datenbank wird damit von Änderungen und Erweiterungen der Anwenderschnittstellen abgekoppelt (logische Datenunabhängigkeit)

Interne Ebene

- Das interne Schema beschreibt die systemspezifische Realisierung der DB-Objekte (physische Speicherung), z.B.
  - Aufbau der gespeicherten Datensätze
  - Indexstrukturen wie z.B. Suchbäume
- Das interne Schema bestimmt maßgeblich das Leistungsverhalten des gesamten DBS
- Die Anwendungen sind von Änderungen des internen Schemas nicht betroffen (physische Datenunabhängigkeit)