



Skript zur Vorlesung  
**Datenbanksysteme I**  
Wintersemester 2012/13

# Kapitel 1: Einführung

Vorlesung: Prof. Dr. Christian Böhm  
Übungen: Sebastian Goebel, Nina Hubig

Skript © Christian Böhm, LMU

[http://www.dbs.ifi.lmu.de/cms/Datenbanksysteme\\_I](http://www.dbs.ifi.lmu.de/cms/Datenbanksysteme_I)



# Das Team

## Vorlesung



Christian Böhm  
Zimmer 158, Di 10-11

## Tutoren/Korrektoren:

Julius Burkart,  
Tim Jeske,  
Sinisa Kurtusic,  
Bernhard Seltl,  
Benjamin Sommer

## Übungs- betrieb



Sebastian Goebel  
Zimmer GU 108



Nina Hubig  
Zimmer GU 108



Search:

- HOME
- LEHRE (GERMAN)
  - Wintersemester 12/13
  - Sommersemester 12
  - Wintersemester 11/12
  - Sommersemester 11
  - Wintersemester 10/11
  - Sommersemester 10
  - Wintersemester 09/10
- STUDIENARBEITEN
- ABOUT THE GROUP
- RESEARCH
- PUBLICATIONS
- TEAM
- CONTACT

- Ansichten:**
- Seite
  - Diskussion
  - Bearbeiten
  - Versionen/Autoren
  - Löschen
  - Verschieben
  - Schützen
  - Beobachten

- Persönliche Werkzeuge**
- Goebl
  - Eigene Diskussion
  - Einstellungen
  - Beobachtungsliste
  - Eigene Beiträge
  - Abmelden

[Lehrstuhl](#) | [Institut](#) | [Fakultät](#) | [LMU](#)

Seitentitel (dieser Bereich ist nur für angemeldete User sichtbar):  
**Datenbanksysteme I WS1213** (diese Seite [aus dem Servercache löschen](#))

## Datenbanksysteme I im WS 2012/13 [[Bearbeiten](#)]

### Aktuelles [[Bearbeiten](#)]

- Die Übungsgruppen am 01.11.2012 **entfallen** wegen des Feiertags. Bitte besuchen Sie in dieser Woche eine andere Übungsgruppe!
- Die Übungsgruppen beginnen erst in der dritten Vorlesungswoche.

### Inhalt [[Bearbeiten](#)]

Die Vorlesung bietet eine Einführung in das Gebiet der Datenbanksysteme aus Anwendersicht. Im Mittelpunkt stehen die theoretischen Aspekte des relationalen Datenbankentwurfs anhand des relationalen Datenmodells, der relationalen Algebra und des Relationenkalküls. Es erfolgt eine ausführliche Behandlung der Anfragesprache SQL, die in den meisten relationalen Systemen implementiert ist. Des Weiteren werden Formalismen, Theorie und Algorithmen der relationalen Entwurfstheorie beschrieben und neuere Anwendungen im Bereich Datenbanken behandelt.

Die erfolgreiche Absolvierung der Vorlesung "Datenbanksysteme I" ist Voraussetzung für alle weiteren Lehrveranstaltungen (Vorlesungen, Seminare, Praktika) sowie studentische Arbeiten im Bereich Datenbanken.

### Organisation [[Bearbeiten](#)]

- **Umfang:** 3+2 Semesterwochenstunden
- **Vorlesung:** [Prof. Dr. Christian Böhm](#)
- **Übungen:** [Sebastian Goebl](#) [Nina Hubig](#)
- **Tutoren/Korrektoren:**

### Zeit und Ort [[Bearbeiten](#)]

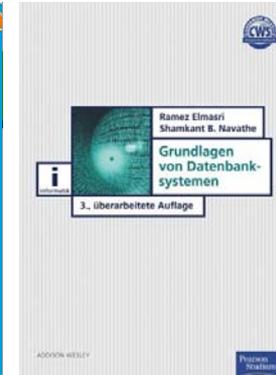
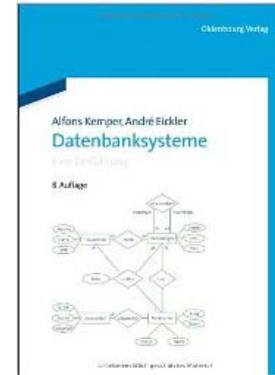
Gruppe	Veranstaltung	Zeit	Ort	Beginn
	Vorlesung	Fr, 12.00 - 15.00 Uhr	Raum M 218 (Hauptgebäude)	19.10.2012
Gruppe 01	Übung	Mo, 10.00 - 12.00 Uhr	Raum B 045 (Theresienstr. 39)	29.10.2012
Gruppe 02	Übung	Mo, 14.00 - 16.00 Uhr	Raum 003 (Schellingstr. 5)	29.10.2012
Gruppe 03	Übung	Mo, 16.00 - 18.00 Uhr	Raum 309 (Schellingstr. 3, Rg.)	29.10.2012
Gruppe 04	Übung	Di, 10.00 - 12.00 Uhr	Raum 109 (Richard-Wagner-Str. 10)	30.10.2012



# Literaturliste

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

- A. Kemper, A. Eickler:  
**Datenbanksysteme**  
Oldenbourg, 8. Auflage (2011). 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).





# Wovon handelt die Vorlesung?

- Bisher (Einführungsvorlesung):  
Nur Betrachtung des Arbeitsspeichers.  
Objekte werden im Arbeitsspeicher erzeugt und nach dem Programmablauf wieder entfernt
- Warum ist diese Sichtweise nicht ausreichend?
  - Anwendungen müssen Daten **persistent** speichern
  - Arbeitsspeicher ist häufig nicht **groß** genug, um z.B. alle Kundendaten einer Bank oder Patientendaten einer Klinik zu speichern
- Aus Vorlesung TGI/Betriebssysteme: Virtueller Speicher
  - Aus unserer Sicht nicht ausreichend; Notwendigkeit, die Speicherung „von der Persistenz her zu denken“



# Persistente Datenspeicherung

- Daten können auf dem sog. *Externspeicher* (Festplatte) persistent gespeichert werden

- Arbeitsspeicher:

- rein elektronisch (Transistoren und Kondensatoren)
- flüchtig
- schnell: 10 ns/Zugriff \*
- wahlfreier Zugriff
- teuer:  
59,90 € für 16 GByte\*  
(3,70 €/Gbyte)

- Externspeicher:

- Speicherung auf magnetisierbaren Platten (rotierend)
- nicht flüchtig
- langsam: 5 ms/Zugriff \*
- blockweiser Zugriff
- wesentlich billiger:  
137,-- € für 3000 GByte\*  
(4,5 ct/Gbyte)

\*Oktober 2012

- Faktor 80 bei den Kosten pro Gbyte
- Faktor 500.000 bei der Zeit für den wahlfreien Zugriff



# Beispiele

## Hardware

- » Arbeitsspeicher
- » Bürotechnik
- » Cooling
- » Drucker & Scanner
- » Eingabegeräte
- » Festplatten
  - » IDE
  - » SATA
    - 1,8 Zoll
    - 2,5 Zoll
    - 3,5 Zoll
  - » SCSI
  - » SAS
  - » USB
  - » FireWire
  - eSATA
  - Thunderbolt
  - Solid State Drives
  - Multimedia
- » **Zubehör**
- » Gehäuse
- » Grafikkarten
- » Kabel
- » Laufwerke
- » Mainboards
- » Monitore
- » Netzwerktechnik
- » PC-Audio

Hardware > Festplatten > SATA > HDS723020BLA642 2 TB

## Hitachi HDS723020BLA642 2 TB

(SATA 600, Deskstar 7K3000, 24/7)

**HITACHI**



- > Kapazität: 2000 GB
- > ms/Cache/U: -/64/7200
- > Preis pro GB: € 0,<sup>04\*</sup>

★★★★★ 40 Bewertungen lesen | bewerten

**€ 87,<sup>90\*</sup>**

Auf Lager

In den Warenkorb

Abb. kann vom Original abweichen

Details

Bewertungen

Preisentwicklung

Zubehör (64)

Video

Die DeskStar 7K3000 ist die erste Festplattenserie von Hitachi, die eine enorme Kapazität von bis zu 2 TB mit einer hohen Performance dank 7.200 U/min, 64 MB Cache und den schnellen SATA/6Gb/s-Interface verbindet. Die HDS723020BLA642 ist das 2 TB fassende Modell der Serie und ist mit einer Sektorgröße von 512 Byte auch zu älteren Systemen kompatibel. Die HDS723020BLA642 ist für den 24/7-Dauereinsatz geeignet und stellt damit eine gute Wahl für Datenbanken und Server-Systeme dar.

### Neueste User-Bewertungen



**Leise, flott, gutes...**

von Coolblue1978  
am 12.10.2011



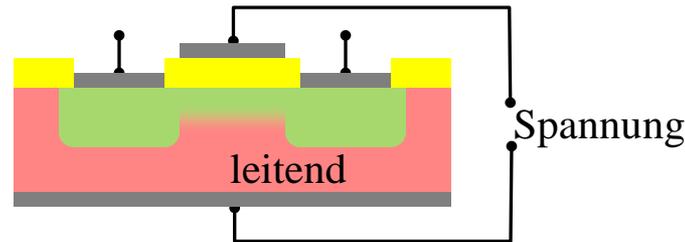
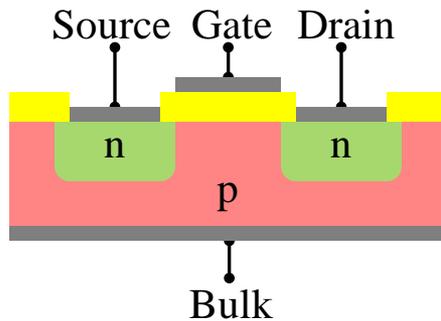
**Mahlzeit, habe mir diese FP...**

von Computer-chaos

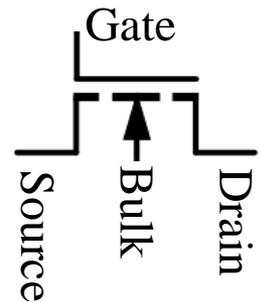


# Technologie des Hauptspeichers

- (Feldeffekt-) Transistor:  
Elektronischer Schalter bzw. Verstärker  
(mit einer schwachen Spannung kann eine höhere Spannung gesteuert werden)



Symbol:

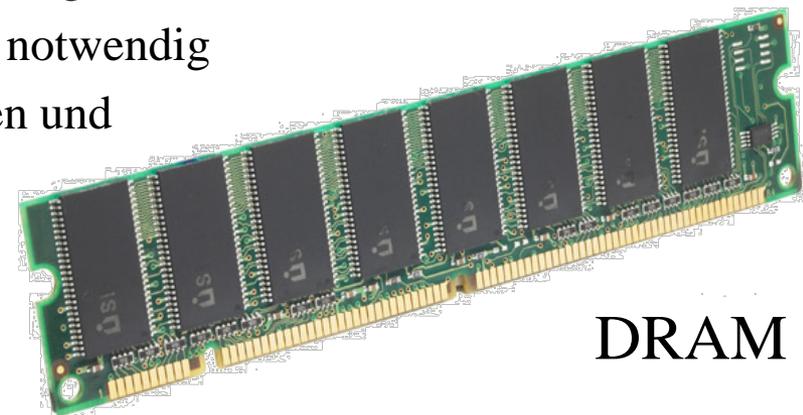
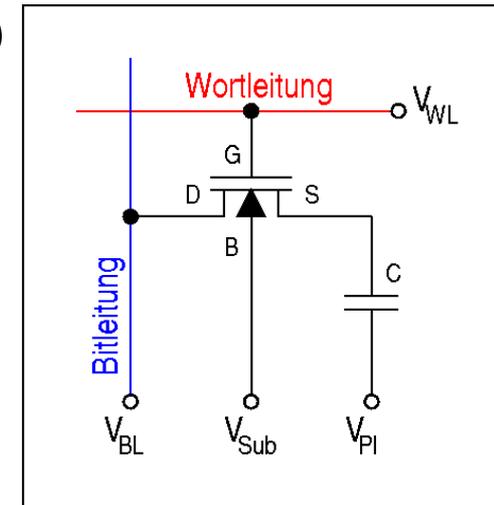


- Der Bereich zwischen Source und Drain ist normal nicht leitend
- Durch Anlegen einer Spannung zwischen Bulk und Gate wandern Elektronen in diesen Bereich und dieser wird leitend („geschlossener Schalter“)



# Arbeitsspeicher

- DRAM (Dynamic Random Access Memory)
- Nur zwei Bauteile pro Bit:
  - 1 Feldeffekttransistor zur Steuerung
  - 1 Kondensator („Akku“ zur Speicherung von Ladung)
- Arbeitsweise:
  - Kondensator kann nur sehr wenig Ladung aufnehmen und diese nur kurzzeitig speichern
  - Wird eine Spannung auf die Wortleitung gegeben, dann kann der Speicherinhalt auf der Bitleitung ausgelesen werden.
  - Verstärker mit Zwischenspeicher notwendig
  - Inhalt muss regelmäßig ausgelesen und wieder zurückgespeichert werden (Refresh)

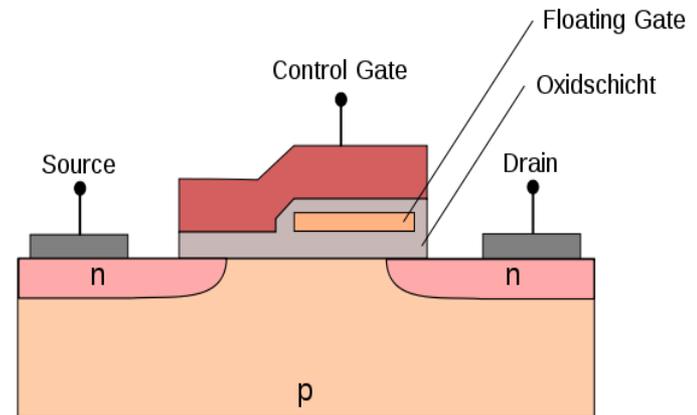


DRAM



# Flash-Memory

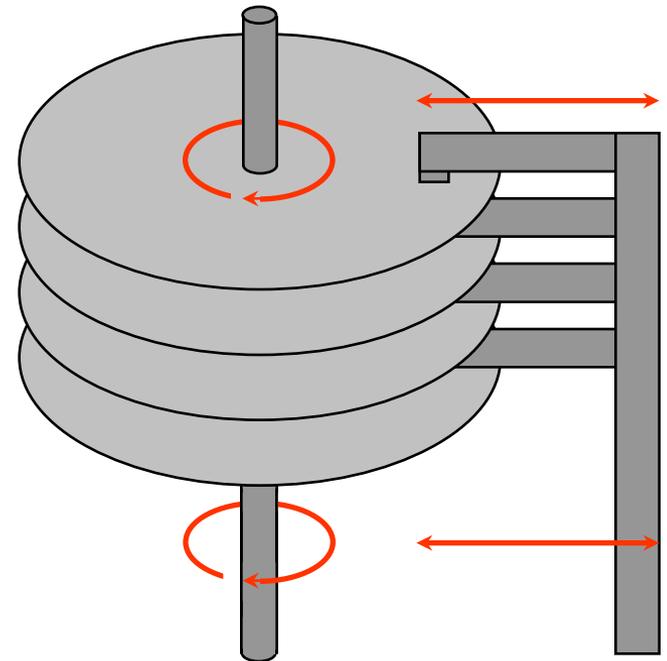
- Flash-Memory
  - Eingesetzt in Memory Sticks, Speicherkarten und sog. Solid State Disks (SSD)
  - Nicht-flüchtige Halbleiterspeicher
- Arbeitsweise
  - Gate von einer Isolationsschicht umgeben („floating gate“)
  - Durch Anlegen einer hohen Spannung können Elektronen eingebracht oder wieder herausgenommen werden („Tunneleffekt“)
- Eigenschaften:
  - Begrenzte Zahl von Schreibzyklen (100.000)
  - Daher nicht verwendbar als Arbeitsspeicher
  - + Sehr geringer Platzbedarf (geringer als DRAM)
  - + Deutlich schneller und robuster als Festplatten





# 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.





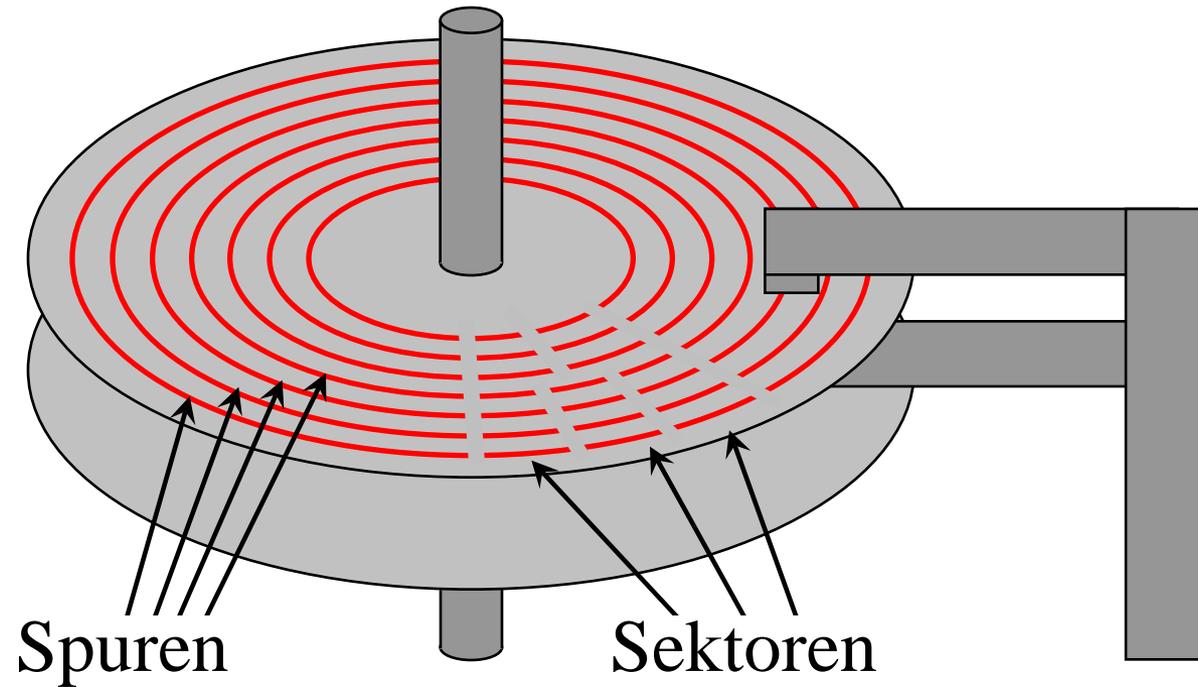
# Demonstration Festplatte

Datenbanksysteme I  
Kapitel 1: Einführung





# Einteilung der Plattenoberflächen



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



# 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



# Demonstration Schreiben/Lesen

Datenbanksysteme I  
Kapitel 1: Einführung





# Zugriffszeit auf einen Sektor

Zugriffszeit für Schreib-/Lese-Auftrag zusammengesetzt aus:

- **Suchzeit** (seek time) zur Positionierung des Kamms:  
typisch 3 ms
- **Latenzzeit** (latency time): Wartezeit wegen Rotation
  - maximal eine Umdrehung, also bei 7200 UPM: 8.3 ms
  - Im Durchschnitt die Hälfte, 4.1 ms  
(Annahme, dass Zeit zwischen zwei Aufträgen zufällig ist -- Poisson-Verteilung)
- **Transferzeit** (transfer time)  
Abhängig von der Länge des Sektors, bzw. es ist auch die Übertragung mehrerer Sektoren in einem Auftrag möglich
  - typische Transferrate: 200 Mbyte/s
  - 1 Sektor à 512 Bytes: 2,5  $\mu$ s

7200 Umdr./min =  
= 120 Umdr./sec;  
1/120 = 0.0083



# 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"); ← Datei öffnen  
        int c = f1.read() ; ← ein Byte lesen  
        long new_position = .... ;  
        f1.seek (new_position) ; ← auf neue Position  
        f1.write (c) ; ← ein Byte schreiben  
        f1.close () ; ← Datei schließen  
    } catch (IOException e) { ← Fehlerbehandlung  
        System.out.println ("Fehler: " + e) ;  
    }  
}
```



# Beispiel: Dateizugriff in Java

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

Name (10 Zeichen)      Vorname (8 Z.)      Jahr (4 Z.)  

F	r	a	n	k	l	i	n			A	r	e	t	h	a			1	9	4	2
---	---	---	---	---	---	---	---	--	--	---	---	---	---	---	---	--	--	---	---	---	---

- 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



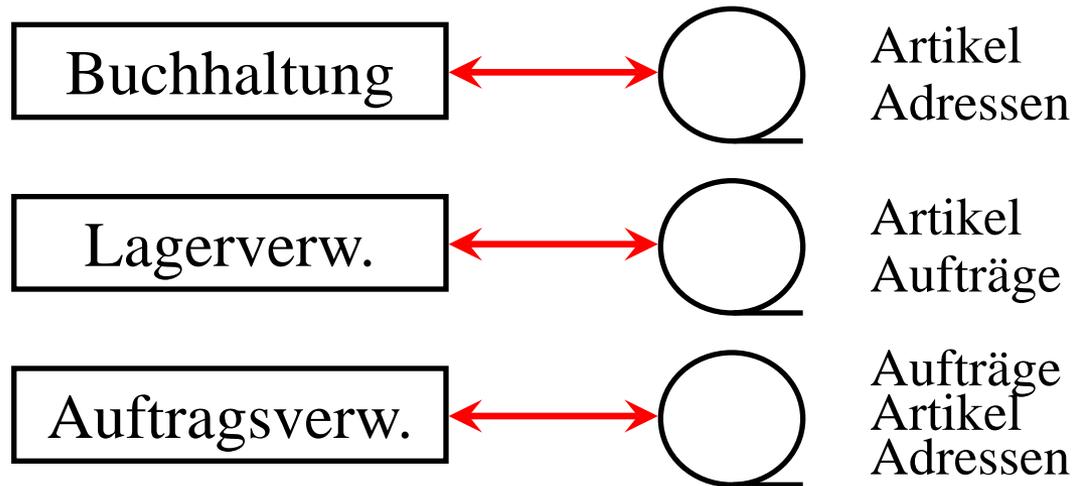
# Mangelnde „Datenunabhängigkeit“

- 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



# Redundanzproblem bei Dateien

- Daten werden mehrfach gespeichert

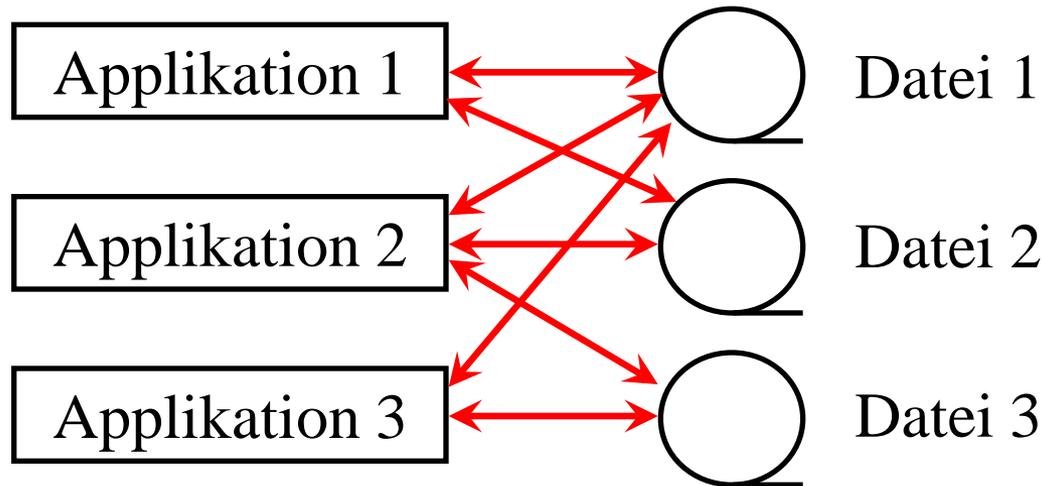


- Konsequenz: u.a. *Änderungs-Anomalien*  
Bei Änderung einer Adresse müssen viele Dateien nach den Einträgen durchsucht werden  
(hierzu später mehr)



# Schnittstellenproblematik

- Alternative Implementierung



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



# Weitere Probleme von Dateien

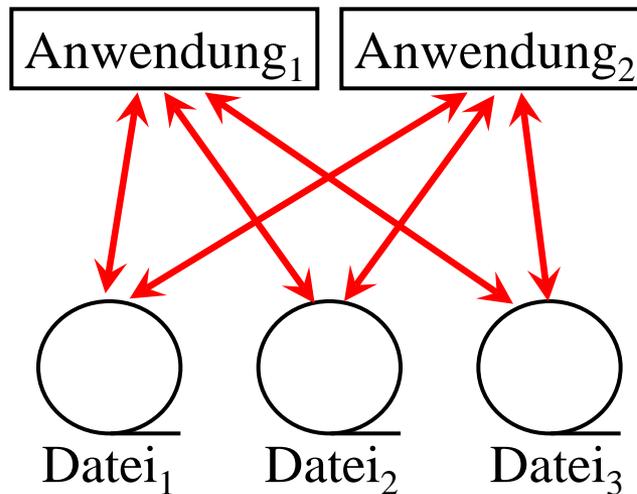
- In großen Informationssystemen arbeiten viele Benutzer gleichzeitig mit den Daten: Dateisysteme bieten zu wenige Möglichkeiten, um diese Zugriffe zu synchronisieren
- Dateisysteme schützen nicht in ausreichendem Maß vor Datenverlust im Fall von Systemabstürzen und Defekten
- Dateisysteme bieten nur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



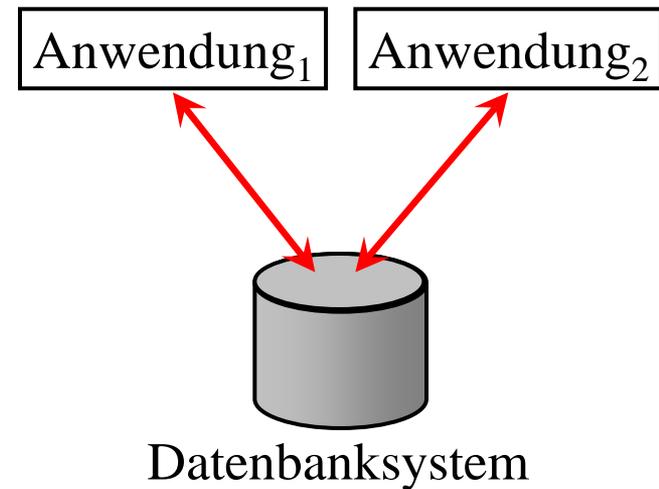
# Von Dateien zu Datenbanken

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

Mit Dateisystem:



Mit Datenbanksystem:



- Einheitliche Speicherung **aller** Daten, die z.B. in einem Betrieb anfallen



# Komponenten eines DBS

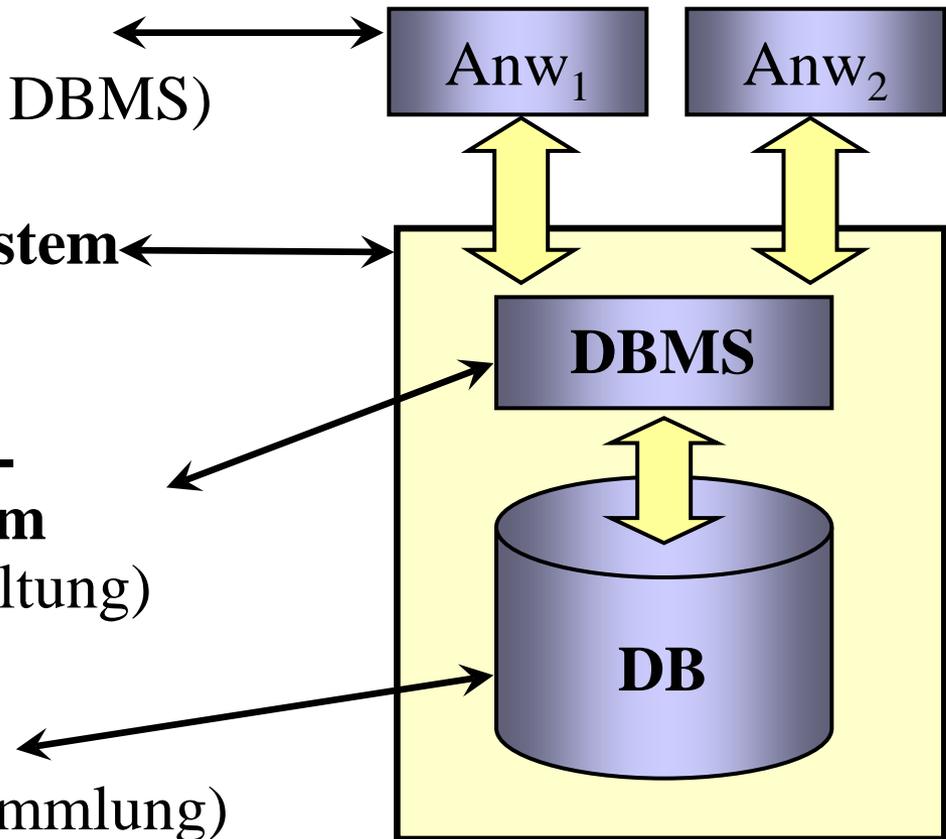
Man unterscheidet zwischen...

**DB-Anwendungen**  
(kommunizieren mit DBMS)

**DBS: Datenbanksystem**  
(DB + DBMS)

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

**DB: Datenbank**  
(eigentliche Datensammlung)





# Aufgaben eines DBS

Primäre Aufgabe eines DBS ist die ...

- Beschreibung
- Speicherung und Pflege
- und Wiedergewinnung

umfangreicher Datenmengen, die von verschiedenen Anwendungsprogrammen 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 unauthorisierter 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:

Name (10 Zeichen)	Vorname (8 Z.)	Jahr (4 Z.)
<input type="text"/>	<input type="text"/>	<input type="text"/>

- DB-Zustand:

<input type="text" value="F"/> <input type="text" value="r"/> <input type="text" value="a"/> <input type="text" value="n"/> <input type="text" value="k"/> <input type="text" value="l"/> <input type="text" value="i"/> <input type="text" value="n"/> <input type="text"/>	<input type="text" value="A"/> <input type="text" value="r"/> <input type="text" value="e"/> <input type="text" value="t"/> <input type="text" value="h"/> <input type="text" value="a"/> <input type="text"/>	<input type="text" value="1"/> <input type="text" value="9"/> <input type="text" value="4"/> <input type="text" value="2"/>
<input type="text" value="R"/> <input type="text" value="i"/> <input type="text" value="t"/> <input type="text" value="c"/> <input type="text" value="h"/> <input type="text" value="i"/> <input type="text" value="e"/> <input type="text"/>	<input type="text" value="L"/> <input type="text" value="i"/> <input type="text" value="o"/> <input type="text" value="n"/> <input type="text" value="e"/> <input type="text" value="l"/> <input type="text"/>	<input type="text" value="1"/> <input type="text" value="9"/> <input type="text" value="4"/> <input type="text" value="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	

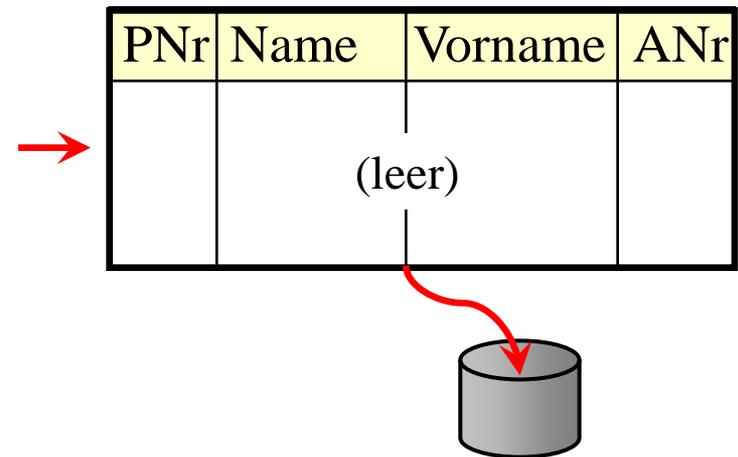
- Bei Schema-Änderung müssen Migrations-Prozeduren programmiert werden, um bestehende Dateien auf das neue Format umzustellen



# 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-Tabelle S. 33)

```
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

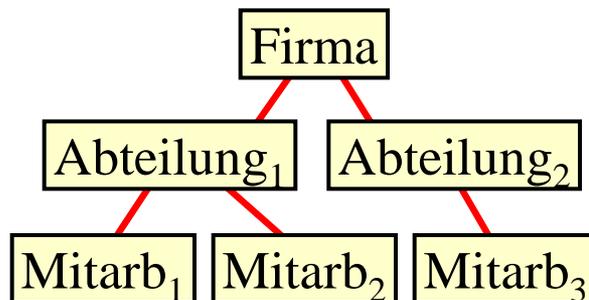
Ergebnis einer Anfrage ist immer eine (neue) Tabelle



# 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

Mitarbeiter



Abteilungen



# 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				Abteilungen	
PNr	Name	Vorname	ANr	ANr	Abteilungsname
001	Huber	Erwin	01	01	Buchhaltung
002	Mayer	Hugo	01	02	Produktion
003	Müller	Anton	02	03	Marketing

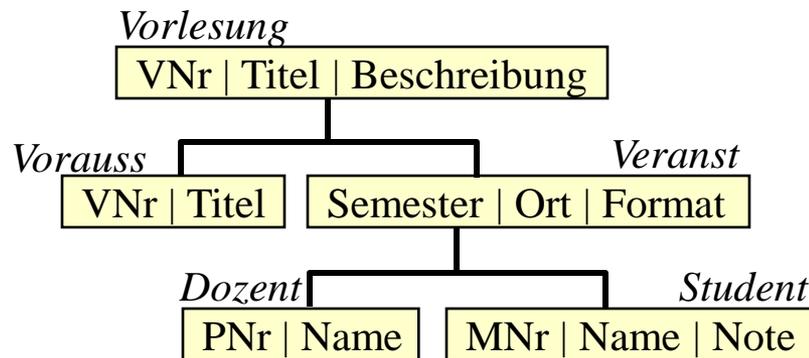
- Ausführliche Behandlung im nächsten Kapitel



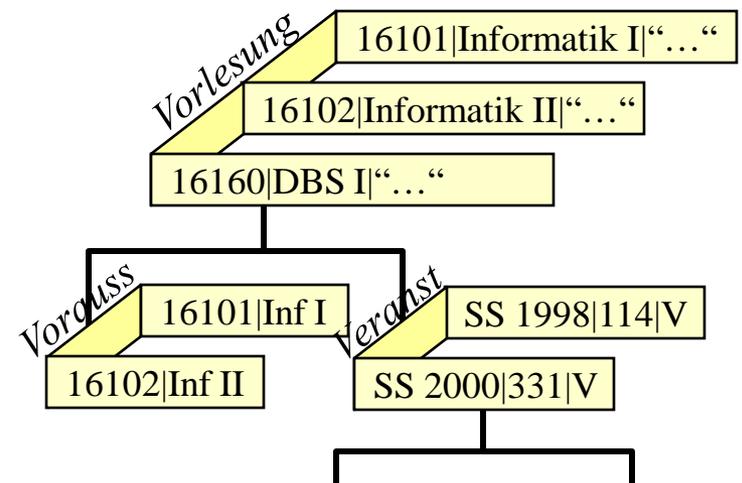
# Hierarchisches Datenmodell

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

## Schema:



## Inhalt:





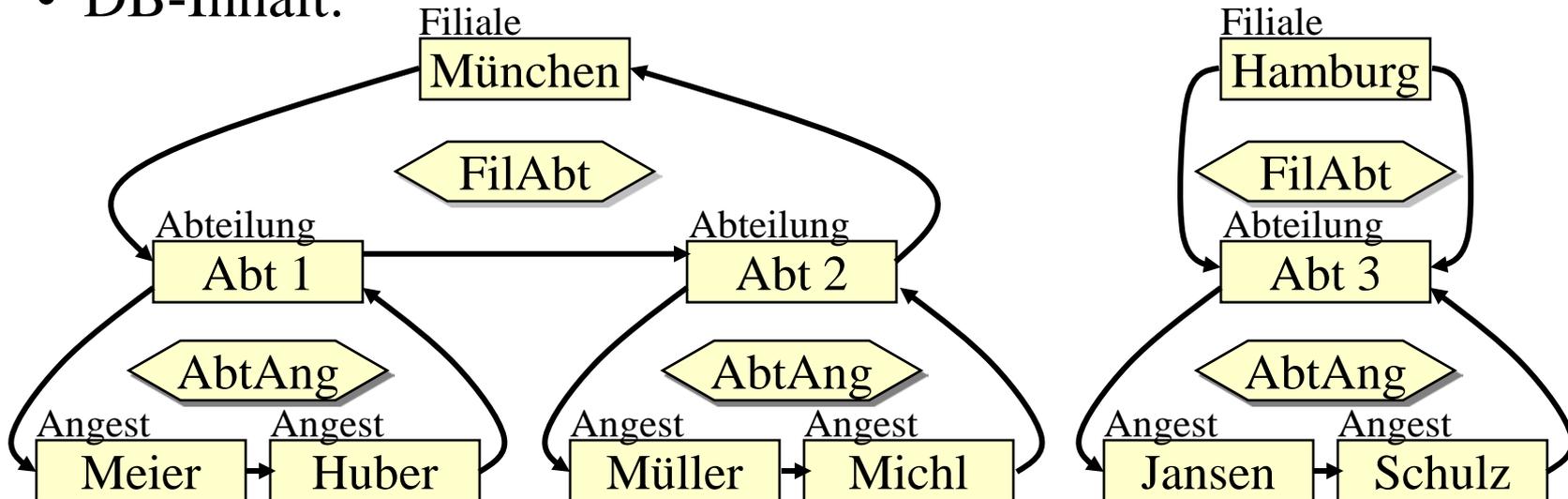
# Netzwerk-Datenmodell

- Schema und Daten werden durch Graphen (Netzwerke) repräsentiert

- Schema:



- DB-Inhalt:



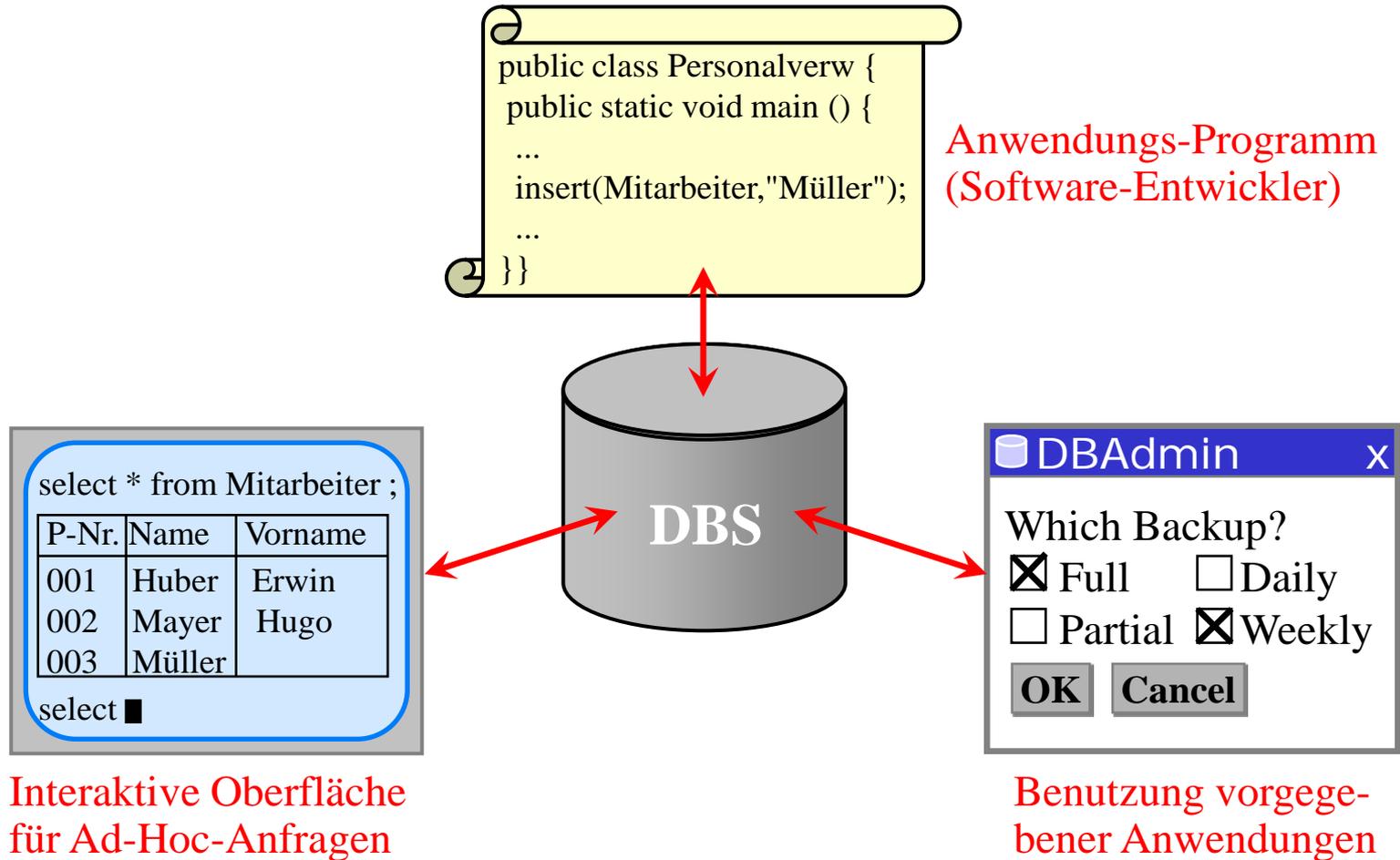


# Objekt-Orientiertes Datenmodell

- 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**



# Verwendung eines DBS



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



# 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

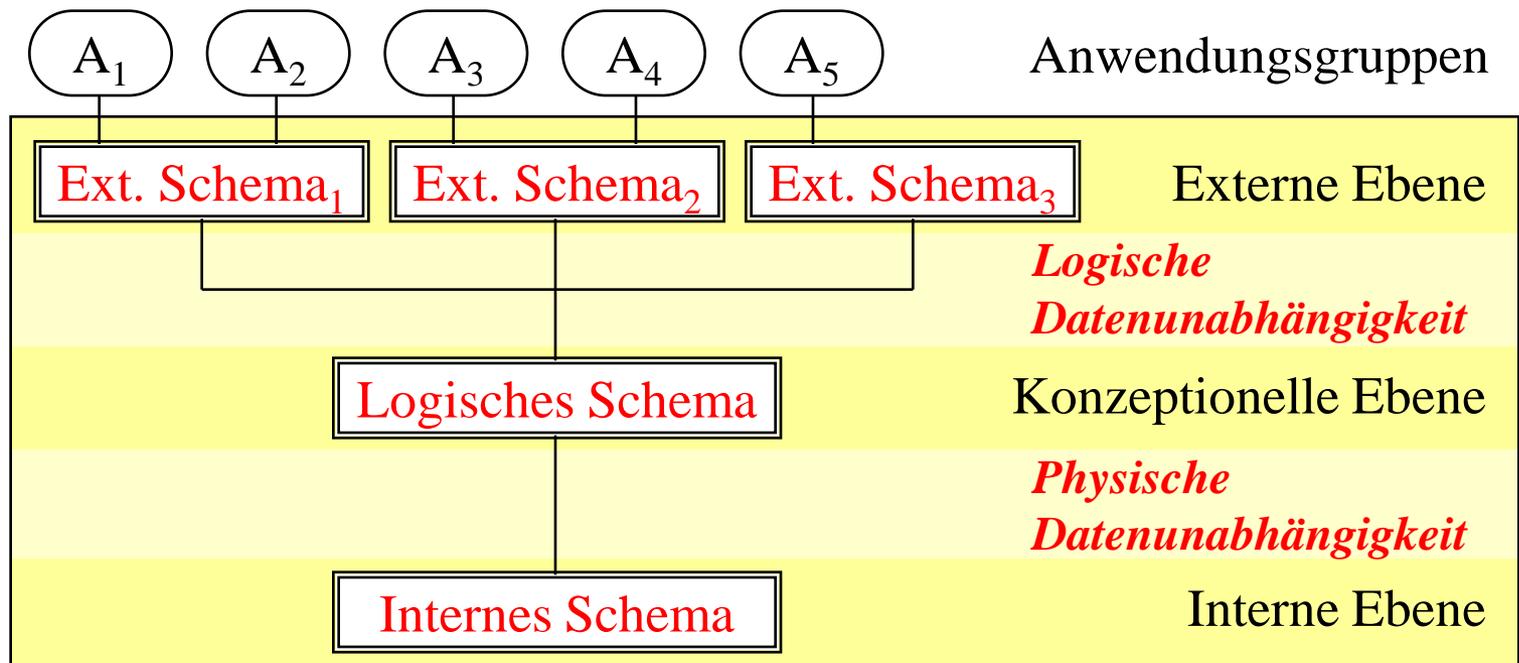


# Architektur eines DBS

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)



# Überblick über die Vorlesung

Datenbanksysteme I  
Kapitel 1: Einführung

Kapitel	Inhalt
1	Einführung
2	Das Relationale Datenmodell
3	Die Relationale Algebra, (SQL, Teil 1)
4	Tupel- und Bereichskalkül (SQL, Teil 2)
5	Sortieren, Gruppieren und Views (SQL, Teil 3)
6	Datenbank-Modellierung mit dem E/R-Modell
7	Normalformen
8	Transaktionen
9	Index-Strukturen
10	Relationale Anfragebearbeitung
11	Anwendungsentwicklung