

# Das Datenchaos vermeiden



Wukasch, Computer sind auch Menschen, R. Müller Verlag, 1983

## Daten speichern ...

- richtig
- vollständig
- einfach auffindbar
- schnell auffindbar

# Datenbank-Anwendungen



- Tel. 111
- Bibliographische Dienste
- Kontostand abfragen
- Barcode an der Kasse lesen
- Buchhaltung führen
- ...

# °Verbreitete Datenbankprogramme

<i>Programm</i>	<i>Entwickler?</i>
<b>MS Access</b>	.....
<b>dBase</b>	.....
<b>Oracle</b>	.....
<b>Sybase</b>	.....
<b>DB2</b>	.....
<b>Notes</b>	.....

# Einordnung

## Deklarativ Anwendungen entwickeln

 [Tabellenkalkulation](#)

 [Elementare Was-Wenn-Analyse](#)

 [Lineare Optimierung](#)

## Prozedural Anwendungen entwickeln

 [Einführung](#)

 [Datentypen und Ablaufstrukturen](#)

 [Benutzerschnittstelle](#)

 [Datenfeld](#)

 [Algorithmen und Datenstrukturen](#)

## Datenbankanwendungen entwickeln

⇒ [Dateiverwaltung](#)

 [KUNDEN](#)

⇒ [Datenentwurf](#)

 [AUFTRÄGE](#)

⇒ [Datenbankverwaltung](#)

 [QBE, SQL, Relationen](#)

⇒ [Anwendungsentwicklung](#)

 [PROGRAMMZUGRIFF](#)

⇒ [\[Verteilte Datenhaltung\]](#)

⇒ [\[Objektorientierte Datenbanken\]](#)

# °Datenbanken und Informationsmanagement

Unternehmensstrategie

Informationstechnologie-Strategie

Informationstechnologie-Planung

Systementwicklung

Anwendung

# Datei bearbeiten

**Datei** :=

## 1. Datenstruktur

= offene Liste aus **Datensätzen**, zum Beispiel ...

<i>Satznummer</i>	<i>Kontonummer</i>	<i>Saldo</i>
1	10 000	2 500.-
2	1 000	100 000.-
3	10 002	-5 000.-

+

## 2. Operationen

zum Beispiel ...

- **Lies** die Sätze mit einem **Saldo** von 100 000.-
- **Schreibe** einen neuen Satz mit der **Kontonummer** 10 003
- **Lösche** den **dritten** Satz

Tabelle 1.5: Sätze und Operationen der Datei KONTO

## ANWENDUNGSENTWICKLUNG

1,2 **Vordefinierte Objektklassen**

3 **Benutzerdefinierte Objektklassen**

4,5 **Dynamische Datenstrukturen**

6 **Applikationsübergreifende Entwicklung**

## DATENBANKEN

7 **Dateiverwaltung**

8 **Datenentwurf I**

9 **Datenentwurf II**

10 **Datenbankverwaltung**

11 **Relationale Strukturen und Operationen I**

12 **Relationale Strukturen und Operationen II**

13 **Programmierender Zugriff**

14 **Repetitorium**

---

# 1 Daten und Dateien

---

⇒ 1 Daten und Dateien

- 2 Einzeldateien organisieren
- 3 Einzeldateien verwalten
- 4 Datenbanken entwerfen

## *Objekte*

Objektyp - Datei

Objekt - Satz

Attribut - Feld

Attributswerte - Daten

## *Attribute*

einfache und zusammengesetzte -

## *Daten*

numerische oder alphanumerische -

Bild- oder Ton -

## *Gültigkeitsprüfungen*

Datentypen und Masken

Verbot von Nullwerten und Duplikaten

Gültigkeitsregeln und -prozeduren

## *Datei*

Format (Textdatei, formatierte Datei)

Zugriffsart (Eingabe-, Ausgabe-)

Verwendungszweck (Stamm-, Bewegungs-)

## *Dateioperationen*

Errichten, Fortschreiben, Suchen



---

# Miniwelten beschreiben

---

## Miniwelt

- Unternehmen

## Objekt

- Konto

## Attribut (Eigenschaft)

- Kontoart

## Wert

- Privatkonto

## Wertebereich

- [Privatkonto, Lohnkonto, Sparkonto, Anlagekonto]

---

# Attributarten unterscheiden - Überblick

---

Datei aus  
Sätzen aus  
Attributen

## 1. Einfache Attribute

- Aufzählungs-  
Grundfarbe (rot, gelb, blau)
- Prädikative -  
integer (alle darstellbaren ganzen Zahlen)
- Ausschnitts-  
Note = 1..6

## 2 Zusammengesetzte Attribute

- Homogen -  
Liste von Noten
- Heterogen -  
Adresse

# Attributarten unterscheiden - Details

Attribut	Unterkategorie	Definiert durch ...	Beispiel
einfaches -	Aufzählungsattribut	Aufzählung	Grundfarbe (rot, blau oder grün)
	prädikatives Attribut	Beschreibung	integer (ganze Zahl)
	Ausschnittsattribut	Wertebereichs- Ausschnitt	Note (Zahl zwischen 1 und 6)
zusammen- gesetztes -	homogen zusammengesetztes Attribut	Komponenten aus dem gleichen Wertebereich	Notenliste (Liste von Noten des gleichen Wertebereichs)
	heterogen zusammengesetztes Attribut	Komponenten verschiedener Datentypen	Adresse (Struktur von Elementen verschiedener Datentypen)

## Übersicht 1.1: Attributarten

# Attributarten unterscheiden (Hausaufgabe 1.2)

## Aufgabe

Tag, Monat, Jahr

Datum

Umsatz

Sprachkenntnisse

Körpergrösse

Zivilstand

Vorname

## Lösung

Attribut	Typ
Tag, Monat, Jahr	Ausschnittstypen
Datum	heterogen zusammengesetztes Attribut
Umsatz	positive Zahl (cardinal)
Sprachkenntnisse	Aufzählungstyp
Körpergrösse	positive Gleitpunktzahl (positive real-Zahl)
Zivilstand	Aufzählungstyp
Vorname	Zeichenkette (string)

---

# Satz einer formatierten Datei strukturieren

---

Spesenabrechnung

% Satzname

Abrechnungsnummer

% Feldnamen

Datum

Spesenbezieher

Spesen \* (1-10)

% bis 9 Wiederholungen möglich

Betragsnummer

Produktnummer

Kostenstelle

Spesenart

Spesenbetrag

## Einrückungsliste 1.6: Satzstruktur einer formatierten Datei

1. **Unformatierte** Dateien (z.B. Textdateien)

2. **Formatierte** Dateien

a) **gleich** lange Sätze

b) **ungleich** lange Sätze

# Beschreibung eines Satzes (Hausaufgabe 1.1)

## Zahlungsbescheid

Datum

Name des Lieferanten

Adresse des Lieferanten

Checknummer |

Bankgiro

Bank des Lieferanten

Kontonummer des Lieferanten

Rechnung \* (0 - 10)

Nummer

Datum

Betrag

[Kommentar]

Zahlungstotal

# Datentypen problemangepasst verwenden

**Margaret Peacock**

Personal-Nr: 4

Vorname: Margaret

Nachname: Peacock

Position: Vertriebsmitarbeiterin

Vorgesetzte(r): Fuller, Andrew

Einstellungsdatum: 03. Mai. 93

Durchwahl Büro: 5176

Persönliche Daten

Datensatz: 4

4

von 9

---

## ° Gültigkeit von Eingabedaten prüfen

---

- Datentyp
- Nachschlagefeld
- Maske
- Obligatorische Eingabe
- Vorgaben
- Duplikateverbot
- Gültigkeitsregel
- Gültigkeitsprozedur



---

# °Gültigkeitsprüfung programmieren

---

```
Sub liesDatum()  
    Datum = InputBox("Datum:")  
    Do While Not IsDate(Datum) or Datum <> ""  
        MsgBox "Verwenden Sie das Format MM/TT/JJ"  
        Datum = InputBox("Datum:")  
    Loop  
End Sub
```

Programm 1.3: Eine Gültigkeitsprüfung in Visual Basic

---

# Datenformate unterscheiden

---

## Numerische Daten:

binär(1010000) = ?

## Alphanumerische Daten

ascii(1010000) = ?

## Bilddaten

1 = ?

0 = ?

## Tondaten

Werte von Frequenz (Tonhöhe) und Amplitutde (Lautstärke)

# °Sprachdaten codieren - BLOBs

Eine Lokalzeitung stellt ihren Lesern einen Informationsbazar zur Verfügung. Der Leser wählt eine Telefonnummer und wird aufgefordert, ein Schlüsselwort zu einem Thema, zum Beispiel zur Wettervorhersage oder zur Börse, zu sprechen. Eine Datenbank Anwendung sucht dann eine passende Datei mit Feldern des Datentyps BLOB. Die Applikation mischt schliesslich die gefundenenen **Sprachdaten** mit Werbeeinspielungen und präsentiert sie dem Anrufer per Telefon.

## Beispiel 1.4: Eine Anwendung mit Sprachdaten

**BLOB** := Binary Large OBject

**variabel lange** Zeichenfolge, die

- langen Text
- Bildfolgen
- Tonfolgen

verschlüsseln

---

## °Alphanumerische Daten codieren - **ASCII-Code**

---

0- 31	Steuerzeichen
32-47	Sonderzeichen
48-57	0..9
58-64	Sonderzeichen
65-90	A..Z (engl.)
91-96	Sonderzeichen
97-122	a..z (engl.)
123-126	Sonderzeichen
127	Steuerzeichen

---

## °Alphanumerische Daten codieren (A 1.3)

---

### Aufgabe

- a) Besonderheiten **nationaler** Zeichensätze ?
- b) Probleme beim **Vergleich von Zeichenketten** nach Anpassung nationaler Besonderheiten ?
- c) Umwandlung eines **Gross**- in einen **Klein**buchstaben ?

### Lösung

#### a) **Nationale** Zeichensätze

- z.B. Umlaute
- z.B. Akzente

#### b) **Vergleich** von Zeichenketten

- z.B. Sortierreihenfolge von ü

#### c) Umwandlung **gross - klein**

Lies Grossbuchstabe ab **Tastatur**

Übersetze in **dezimales** Äquivalent

Addiere **+32**

Übersetze in entsprechendes ASCII-Zeichen auf **Bildschirm**

# Begriff der Datei

**Datei** :=

1. **benannte** Folge von Elementen
2. mit **Zugriffsoperationen**
3. auf einem Externspeicher

## Beispiele

Datei **KONTO**

Kontonummer	Saldo
10 000	2 500.-
1'000	100'000.-
10'002	-5'000.-

## Zugriffsoperationen

**Lies** Satz mit der Kontennummer 10'002

**Schreibe** Satz mit der Kontonummer 10'003

---

# Dateien nach der Zugriffsart unterscheiden

---

## Eingabedatei

Bsp. Rabatttabelle für ein Fakturierungsprogramm

## Ausgabedatei

Bsp. Programmtext -> Objektdatei (.OBJ) -> Programmcode (.EXE)

## Eingabe-/Ausgabedatei

Bsp. Saldi der Debitorendatei *lesen* und bei Überschreiten der Kreditlimite mit dem Vermerk "Kredit ausgeschöpft" *zurückschreiben*

---

## °Dateien nach ihrer Verwendung unterscheiden

---

### **Stamm**datei (permanent)

Bsp. Lieferanten-, Personal-, Stücklisten-

### **Bewegungs**datei (vorübergehend)

Bsp. Zu- oder Abgänge von Waren und Zahlungen

### **Berichts**datei (engl. report file)

Bsp. aufbereitete Daten für Bildschirm, Drucker etc.

### **Arbeits**datei (Transferdatei)

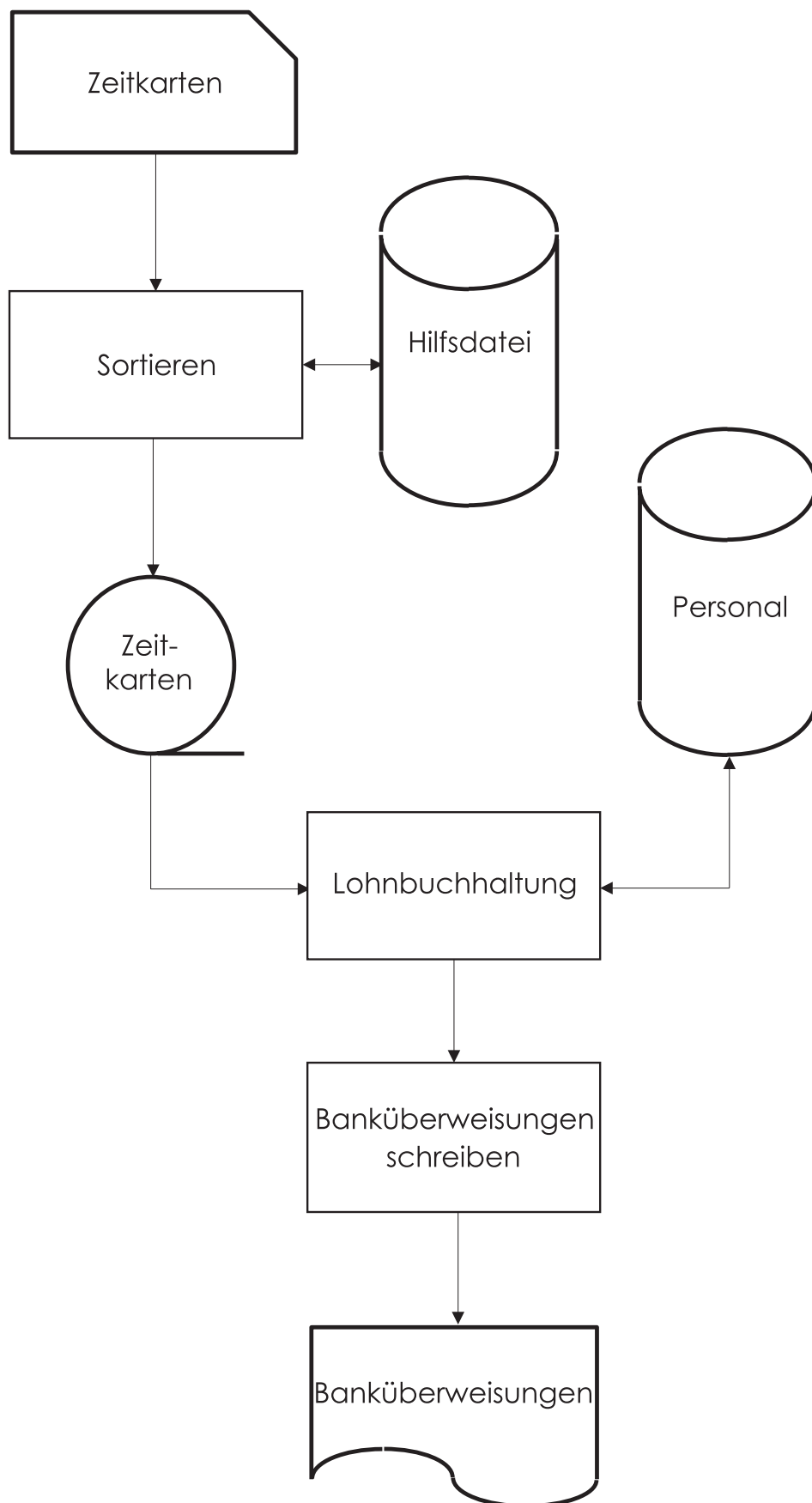
temporär

### **Programm**datei

aus Befehlen zur Verarbeitung von Daten



## °Datenfluss grafisch darstellen (A 1.4)



**Bild 1.7:** Datenfluss einer Lohnbuchhaltung

## °Dateiarten (Lösung 1.4)

<i>Dateiname</i>	<i>Zugriff</i>	<i>Verwendung</i>
Personal	Eingabe-/Ausgabedatei	Stammdatei
Zeitkarten	Eingabedatei	Bewegungsdatei
Banküberweisungen	Ausgabedatei	Bewegungsdatei
Sortierdatei	Eingabe-/Ausgabedatei	Arbeitsdatei
Lohnbuchhaltung	-	Programmdatei
Sortieren	-	Programmdatei
Per Bank überweisen	-	Programmdatei

---

# Dateien verwalten

---

## Datei errichten

- interaktiv
- stapelweise

## Satz fortschreiben

- Einfügen
- Löschen
- Ändern

## Satz suchen

### Beispiel

```
USE KONTO  
DISPLAY ALL FOR Saldo > 0
```

---

## 2 Einzeldateien organisieren

---

1 Daten und Dateien

⇒ 2 Einzeldateien organisieren

- 3 Einzeldateien verwalten
- 4 Datenbanken entwerfen

*Speicher*

externer -

interner -

*Puffer*

Datei-

Satz-

*Dateizugriff*

sequentieller -

direkter -

*Dateiorganisation*

sequentielle -

relative -

indizierte -

*Dateiverwaltung*

Datei und Satzpuffer deklarieren

Datei öffnen

Satz lesen und schreiben

Dateiende feststellen

Datei schliessen



# Bedeutung der Laufzeiteffizienz

1. Ein Compaq-Kunde sucht **telefonisch** Unterstützung
2. Der automatische Telefonbeantworter identifiziert den Kunden via Touch-Tone Input (**Automatic Number Identification**)
3. Die Applikation sucht Kundeninformation auf der **Datenbank**
4. Die Applikation sendet die Kundeninformation auf den **Bildschirm** des Sachbearbeiters.
5. Der Sachbearbeiter berät den Kunden **telefonisch**

## ***Exkurs - Automatische Ermittlung der Rufnummer***

- engl. Automatic Number Identification (ANI)
- Anrufer-**Logging**
- **Automatische Anzeige von Anruferdaten** beim ersten Läuten
- Vereinfachter **Vertragsabschluss**  
(zum Beispiel Heimlieferung ohne ausdrückliche Adressangabe)
- Vereinfachte automatische **Telefonbeantwortung**
- **Automatische Blockierung** von Anrufen

# Lebenszyklus einer Datei

**Datei** := benannte und offene Folge von Sätzen,  
auf der Zugriffsoperationen definiert sind

<i>Phase: Datei ...</i>	<i>Hauptzweck</i>
beschreiben	Datei und Satzpuffer deklarieren
öffnen	Speicherplatz reservieren
zugreifen	Sätze lesen und schreiben
schliessen	Speicherplatz freigeben, Dateiverzeichnis führen

Übersicht 2.3: Lebenszyklus einer Datei in einem Programm

# Sätze auf dem Speicher adressieren

**Direkter** (unmittelbarer) Zugriff auf eine Speicheradresse

- Lies den **dritten** Satz der Datei XY
- Zugriffsgeschwindigkeit **unabhängig** von der Satzposition

**Sequentieller** (sukzessiver) Zugriff auf einen Speicherinhalt

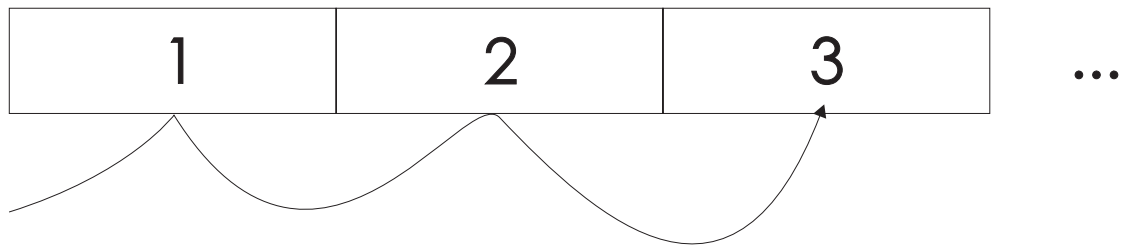
- Lies den Satz des Konteninhabers **Fridolin Müller**
- Zugriffsgeschwindigkeit **abhängig** von der Satzposition

Direkte Speicher	Sequentielle Speicher
Internspeicher	Magnetband und -kassette
Magnetplatte	Lochstreifen
Diskette	Lochkarte

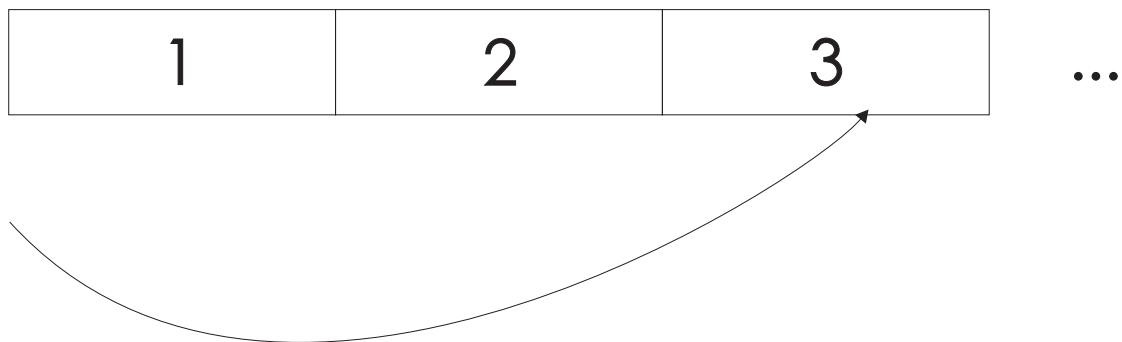
Vergleich 2.2: Direkte und sequentielle Speichermedien

# Datensätze sequentiell und direkt suchen

## 1. Sequentiell suchen



## 2. Direkt suchen



**Bild 2.1: Sequentielles und direktes Suchen**



---

## ① Eine Datei sequentiell organisieren

---

### Unterscheide!

#### Zugriff auf ein Speichermedium

- über eine Speicher**adresse**
- über einen Speicher**inhalt**

#### Organisation einer Datei (Datenstruktur + Zugriff)

- Sequentiell organisierte Datei
- Relativ organisierte Datei
- Indiziert organisierte Datei

### Sequentiell organisierte Datei

#### 1. Datenstruktur

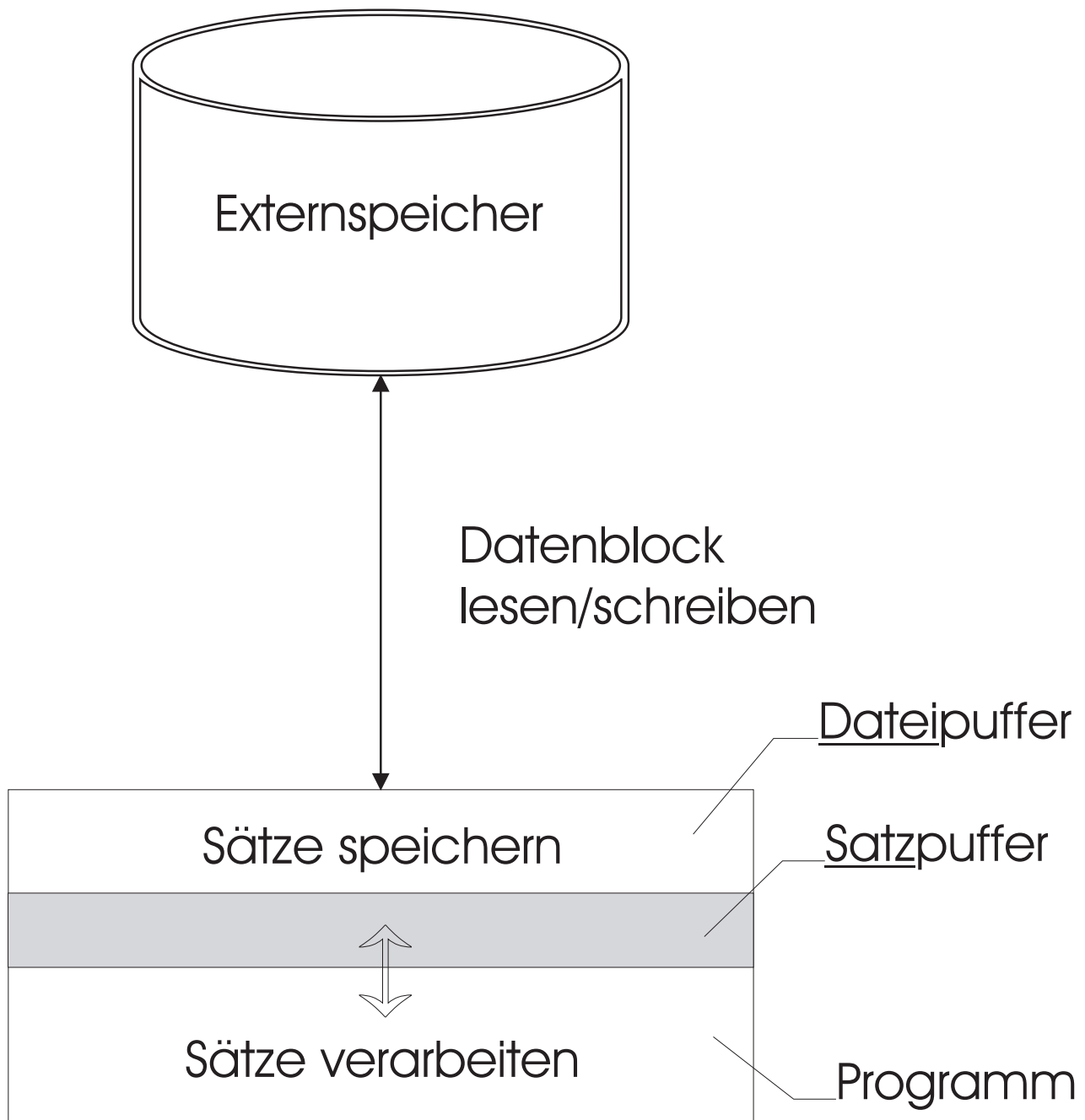
Sätze **hintereinander**

#### 2. Zugriffsooperationen

Zugriff auf den jeweils **nächsten Satz**

- a) **Lesen** des nächsten Satzes (kein Rückwärtslesen)
- b) **Schreiben** des nächsten Satzes (kein Rückwärtsschreiben)

## °Sätze intern zwischenspeichern



**Bild 2.4:** Dateipuffer und Satzpuffer

---

# °Sequentielle Dateorganisation in Pascal

---

## 1. Satzpuffer deklarieren

**type**

```
Kontentyp = record  
    Nummer: integer;  
    Saldo: real;  
end;
```

**var**

```
Kontensatz: Kontentyp;
```

## 2. Datei zum Schreiben öffnen (oder Lesen)

```
assign(Kontendatei, externerDateiname)  
rewrite(Kontendatei)
```

## 3. Sätze schreiben (oder lesen)

```
write(Kontendatei, Kontensatz)
```

## 4. Datei schliessen

```
close(Kontendatei)
```

## °Eine sequentielle Datei in Pascal erstellen

```
program   errichteSequentielleDatei;
Eingabe:   Dateispezifikation, Kontonummer, Saldo
Ausgabe:   Datei gemäss Dateispezifikation }

type
  Kontentyp = record
    Nummer: integer;
    Saldo: real end;

var
  Kontensatz: Kontentyp;
  externerDateiname: string;
  Kontendatei: file of Kontentyp;           1)

begin
  {Initialisieren}

  write('Dateispezifikation: ');
  readln(externerDateiname);
  assign(Kontendatei, externerDateiname);    2)
  rewrite(Kontendatei);                     3)
  write('Kontonummer (0 bricht ab): ');
  read(Kontensatz.Nummer);

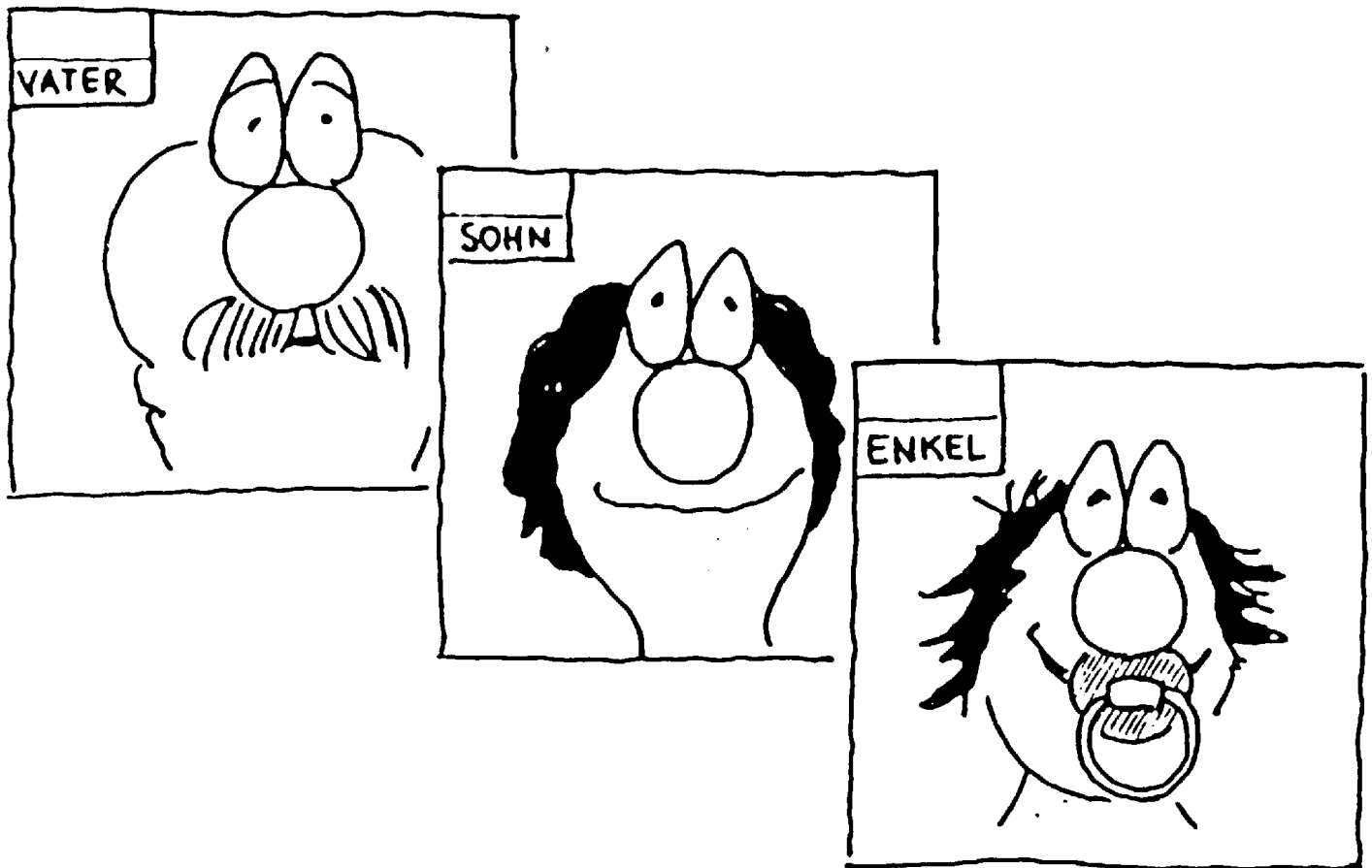
  {Verarbeiten}

  while Kontensatz.Nummer <> 0 do begin       4)
    write('Saldo: '); read(Kontensatz.Saldo);
    write(Kontendatei, Kontensatz);          5)
    write('Kontonummer (0 bricht ab): ');
    read(Kontensatz.Nummer)
  end;

  {Abschliessen}

  close(Kontendatei)                         6)
end.
```

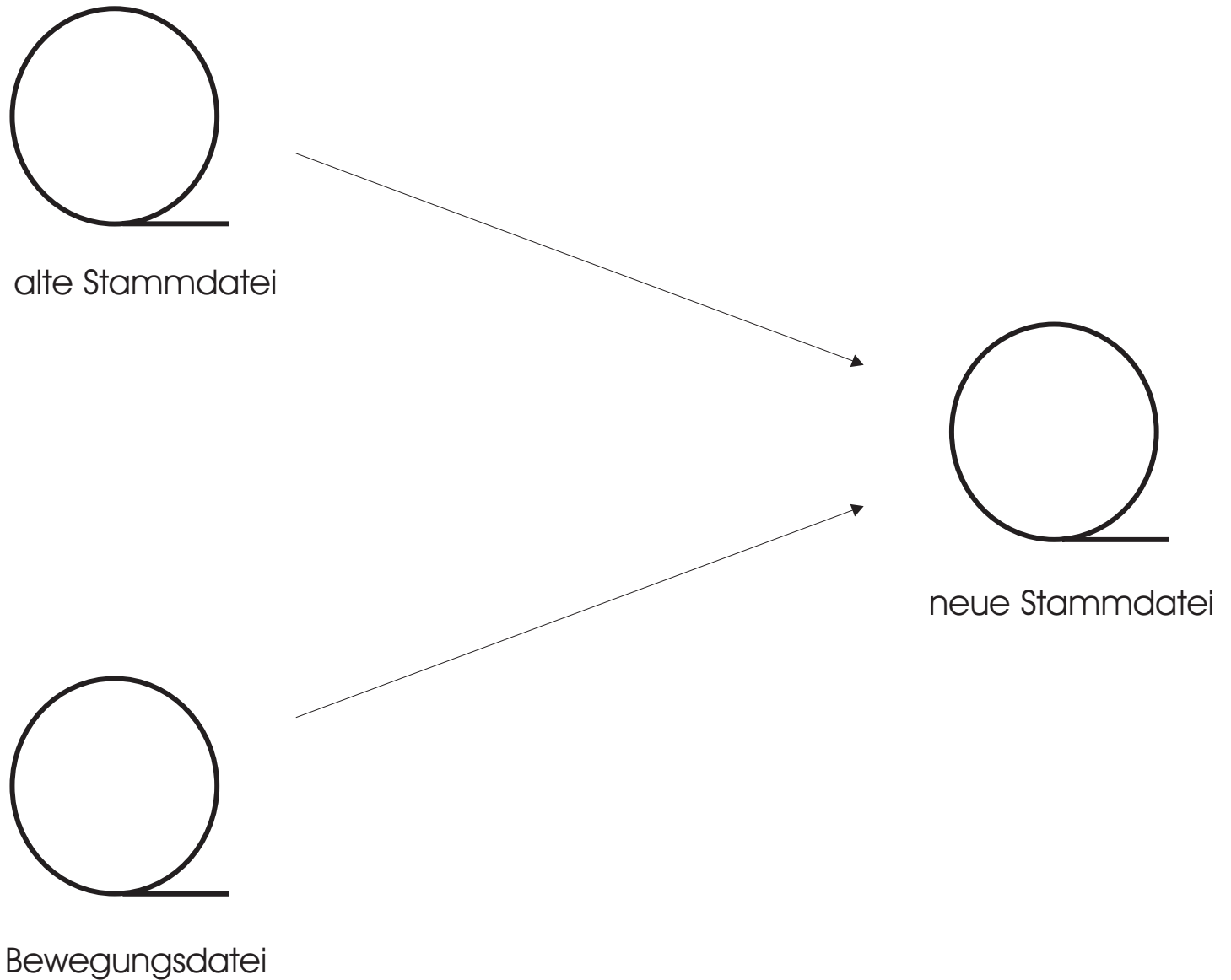
### Programm 2.5: Errichten einer sequentiellen Datei in Borland Pascal



## Vater-Sohn-Prinzip

Mehrere Dateigenerationen  
als Sicherung gegen Daten-  
verluste speichern

## °Eine sequentielle Datei fortschreiben



**Bild 2.6: Fortschreibung einer sequentiellen Datei**

---

# °Sequentiell fortschreiben - Voraussetzung

---

## 1. Sequentiell organisierte **Stammdatei**

- mit mindestens einem Satz
- mit den Feldern *Kontonummer* und *Saldo*
- nach Kontonummern aufsteigend sortiert
- ohne Sätze mit gleicher Kontonummer.

## 2. Sequentiell organisierte **Bewegungsdatei**

- mit mindestens einem Satz
- mit den Feldern *Kontonummer* und *Bewegung*
- nach Kontonummern aufsteigend sortiert
- auch mit Sätzen gleicher Kontonummer.

## 3. Sequentiell organisierte **neue Stammdatei**

- mit Sätzen für jede Kontonummer in StammAlt oder Bewegung
- nach Kontonummern aufsteigend sortiert.

## °Sequentiell fortschreiben - Beispiel

### 1. Alte Stammdatei

Kontonummer	Saldo
10 000	2 500.-
10 001	100 000.-
10 002	-5 000.-
10 011	1 000.-
10 235	500.-
10 890	3 000.-
32 767	0.- (letzter Satz)

### 2. Bewegungsdatei

Kontonummer	Bewegung
10 001	1 000.-
10 005	10 000
10 235	-1 000.-
32 767	0.-

### 3. Neue Stammdatei

Kontonummer	Bewegung
...	...
...	...
...	...
...	...
...	...
...	...
...	...



# °Sequentiell fortschreiben - Entwurfscode

Kontensatz

Nummer: integer

Betrag: real

alte Stammdatei: Datei aus Kontensätzen

Bewegungsdatei: Datei aus Kontensätzen

neue Stammdatei: Datei aus Kontensätzen

Eröffne Dateien

Lies ersten Stammsatz

Lies ersten Bewegungssatz

**Solange** Nr alter Stammsatz < letzte\_Nr

**oder** Bewegungsnr < letzte\_Nr dann

**Falls** Nr alter Stammsatz <= Bewegungsnr dann

{ alten Stammsatz kopieren }

neuer Stammsatz := alter Stammsatz

Lies nächsten Stammsatz

**sonst**

{ neuen Stammsatz einfügen }

neuer Stammsatz := Bewegungssatz

Lies nächsten Bewegungssatz

**Solange** Nr neuer Stammsatz = Bewegungsnr

**und** Bewegungsnr < letzte\_Nr dann

{ alten Stammsatz ändern bzw.

Bewegungen aggregieren }

Betrag neuer Stammsatz :=

Betrag neuer Stammsatz + Betrag Bewegungssatz

Lies nächsten Bewegungssatz

Schreibe neuen Stammsatz

Schreibe letzten Satz der neuen Stammdatei

Schliesse Dateien

## ② Eine Datei relativ organisieren

### 1. Datenstruktur

Zellen fester Länge

### 2. Operationen

Direktzugriff auf die relative Zellennummer

#### Beispiel

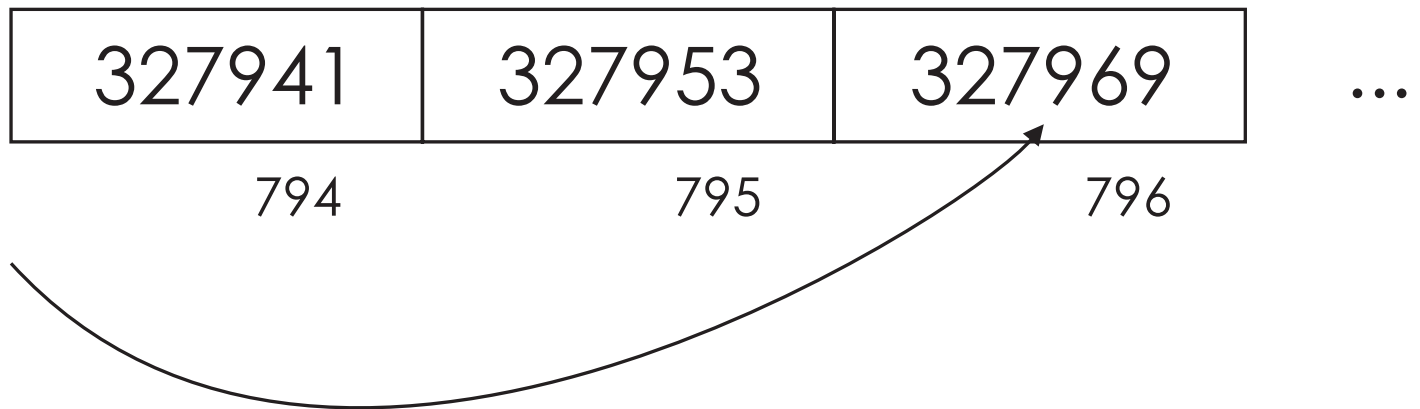
Zellennr	1	2	3	4	5
Zelleninhalt	1. Satz	leer	leer	2. Satz	3. Satz

#### Adressberechnung für den Direktzugriff

- Zugriffsadresse =  $\text{Dateiadresse} + (\text{Zellennummer} * \text{Zellengrösse})$
- Wenn Suchattribut  $\neq$  Zellennummer, dann Hash-Algorithmus

# Einen Satz mit einer Hashfunktion suchen

## 1. Versicherungsnummer suchen



## 2. Name suchen

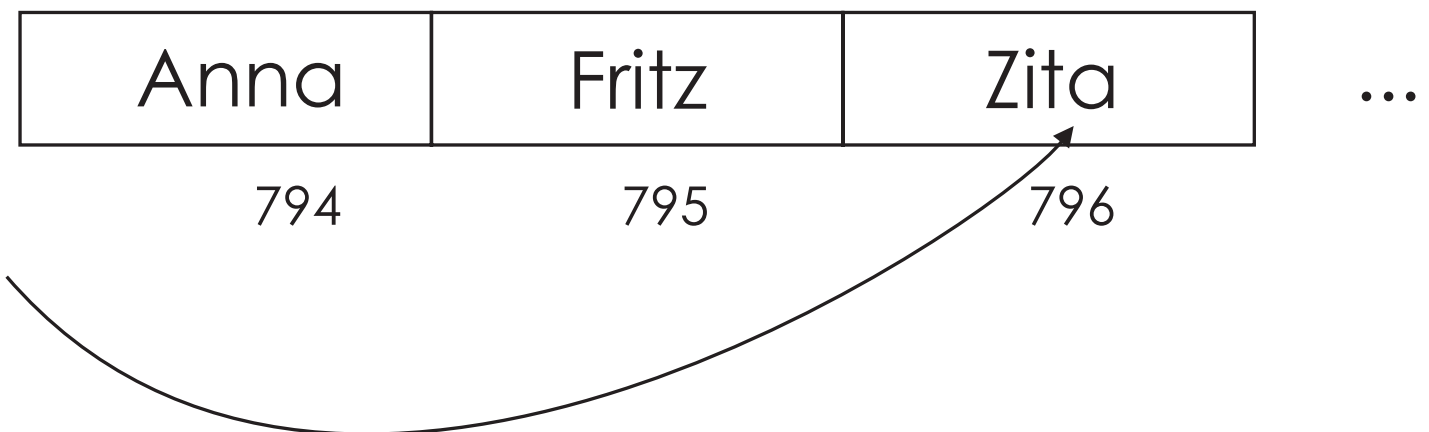


Bild 2.7: Ausgangslage des Hashzugriffs

# Beispiel einer Hashfunktion

Schlüssel	Hashfunktion	Zellennr	Bemerkung
507	$507 \bmod 1000$	507	Rest von 507 div 1000
5 076	$5\,076 \bmod 1000$	076	
309	$309 \bmod 1000$	309	
956 320	$956\,320 \bmod 1000$	320	
48 309	$48\,309 \bmod 1000$	309	Kollision

**Tabelle 2.8:** Hashfunktion  $\text{Zellennummer} := \text{Schlüssel} \bmod 1000$

## ③ Eine Datei indiziert organisieren

### Motivation

#### Beispiel einer Satzstruktur

Kontosatz

Inhabername

Saldo

Kontentyp { Spar-, Anlage-, Lohn- }

**Eignung** der sequentiellen und direkten Dateiorganisation für die folgenden Abfragen:

- Saldo des Kontos 10'003?
- Liste der Konteninhaber, sortiert nach Kontentyp



### Indiziert organisierte Datei

#### **1. Datenstruktur**

- Hauptdatei
- sortierte Indizes für Sekundärschlüssel

#### **2. Operationen**

- Direktzugriff auf indizierte Attribute

---

# Schlüsselattribute unterscheiden

---

## **Primärschlüssel** (Identifikationsschlüssel)

Attribut, das einen Satz

- **identifiziert**

Beispiel: Matrikelnummer

## **Sekundärschlüssel** (Zugriffsschlüssel)

Attribut, das hilft einen Satz zu

- **suchen** und zu
- **sortieren**

Beispiel: Nachname

# Aufbau einer Indextabelle

<i><u>Kontonummer</u></i>	<i>Adresse des Kontensatzes</i>
10 001	6
9 050	20
300	2
...	...

**Tabelle 2.9:** Index zu einem Primärschlüssel [Kontonummer](#)

## Wahl der Dateiorganisation (Hausaufgabe 2.8)

- 1) Ein Angestellter eines Detailhandelsgeschäfts tippt abends am **Terminal** die Zahl der verkauften Einheiten jedes Produkts ein. **Sobald** der kritische Lagerbestand unterschritten wird, schreibt das System eine **Bestellung**. Jede Woche wird ein zusammenfassender Bericht über die Lagerbewegungen geschrieben.
- 2) Ein Gaswerk liest jeden **Monat** den Zählerstand seiner Kunden in den Computer. Das System erzeugt darauf die Rechnungen und führt die Kundendatei nach. Die Kundendatei enthält unter anderem die folgenden Felder: *Kundennummer*, *Name*, *Adresse*, *Schuld*.
- 3) Ein Spital unterhält ein Patienten-**Informationssystem**. Bei der Aufnahme eines Patienten werden interessierende Daten am **Terminal** erfasst. Später kommen Informationen zur Krankengeschichte hinzu.

## Lösung

- 1) relativ, Produktenummer
- 2) sequentiell, Kundennummer
- 3) indiziert, AHV-Nummer



---

## °Binär suchen - Motivation

---

Wer im Telefonbuch blättert, ...

- springt in die **Mitte**
- schaut, ob der Suchwert in der **linken** oder **rechten** Buchhälfte liegt
- setzt die Schritte 1 und 2 in der gefundenen Hälfte fort, bis er nach weiteren Halbierungen die Seite mit dem Suchwert gefunden hat.

Die Bestimmung der Mitte und der linken bzw. rechten Hälfte setzt ein *sortiertes* Telefonbuch voraus.

## °Rekursive Binärsuche entwerfen (Abb. 2.10, 2.11)

```
Suche_binär(links, rechts, Name, Mitte)
  {-- Mitte gibt die gesuchte Zellennummer zurück}
  Mitte := (links + rechts) div 2;
  Falls links <= Mitte
    Falls Name = Index[Mitte]
      Name gefunden
    sonst falls Name < Index[Mitte]
      Suche_binär(links, Mitte-1, Name, Mitte)
    sonst {Name > Index[Mitte]}
      Suche_binär(Mitte+1, rechts, Name, Mitte)
  sonst
    Name nicht gefunden
```

Nummer	1	2	3	4	5	6	7
Index[Nr]	ANNA	BRUNO	DORA	FRANZ	KARIN	NORA	PAUL
1. Schritt	L			M			R
2. Schritt					L	M	R
3. Schritt							L,M,R

 [Code in Visual Basic](#)

 [Code in Pascal](#)

---

# °Rekursive Binärsuche in Pascal

---

```
program sucheRekursivBinaer;  
{Satznummer aufgrund des Namens rekursiv binär suchen}  
  
const  
    Anfang = 1; Ende = 7;  
    Index: array[Anfang..Ende] of string =  
        ('ANNA','BRUNO','DORA','FRANZ','KARIN','NORA','PAUL');  
  
function rekursivBinaerGefunden(  
    links, rechts: integer;  
    Name: string; var Mitte: integer): boolean;  
{ Mitte gibt die gefundene Satznummer zurück }  
begin  
    Mitte := (links + rechts) div 2;  
    if links <= Mitte then begin  
        if Name = Index[Mitte] then  
            rekursivBinaerGefunden := true  
        else if Name < Index[Mitte] then  
            rekursivBinaerGefunden :=  
                rekursivBinaerGefunden(links,Mitte-1,Name,Mitte)  
        else if Name > Index[Mitte] then  
            rekursivBinaerGefunden :=  
                rekursivBinaerGefunden(Mitte+1,rechts,Name,Mitte)  
        end  
    else  
        rekursivBinaerGefunden := false  
    end;  
  
var Nr: integer; Name: string;  
  
begin { Hauptprogramm }  
    ...  
    if rekursivBinaerGefunden(Anfang,Ende,Name,Nr) then  
        writeln('Gesuchte Satznummer = ', Nr)  
    else  
        writeln('Kein Satz mit diesem Schlüsselwert');  
    ...  
end.
```

## °Der Binärbaum - Eine Alternative zum Array

Adresse	Schlüssel	Adresse Vorgänger	Adresse Nachfolger
1	Anna	nil	2
2	Bruno	1	3
3	Dora	2	4
4	Franz	3	8
5	Karin	11	6
6	Nora	5	7
7	Paul	6	nil
8	Frida	4	9
9	Fridolin	8	10
10	Friedrich	9	11
11	Fritz	10	5

**Tabelle 2.12:** Index als Binärbaum (nil verweist nicht weiter)

## ④ Eine Datei verkettet organisieren

### Eindeutiger Schlüssel

<u>Kontennummer</u>	Speicheradresse
10'001	6
9'050	20
300	2
...	...

### Mehrdeutiger Schlüssel

<u>Inhabername</u>	Speicheradresse
Iwan Borkowski	6
Fritz Müller	12, 20, 30
...	...

# Sätze zugriffseffizient verketten

## Indexdatei

<i>Inhaber</i>	<i>Satzadressen</i>	<i>Bemerkung</i>
Iwan Borkowsky	6	eindeutige Zuordnung
Fritz Müller	12, 20, 30	mehrdeutig Zuordnung
...	...	...

**Tabelle 2.13:** Grund für die Verkettung ist ein **mehrdeutiger Schlüssel**

<i>Inhaber</i>	<i>Adresse 1. Satz</i>	<i>Bemerkung</i>
Iwan Borkowsky	6	Adresse des einzigen Satzes
Fritz Müller	12	Adresse des ersten Satzes
...	...	...

**Tabelle 2.14:** Verkettung ermöglicht eine Tabelle mit **Sätzen gleicher Länge**

## Hauptdatei

<i>Adresse</i>	<i>Kontonr</i>	<i>Inhaber</i>	<i>Saldo</i>	<i>Fortsetzungsadresse</i>
12	10 030	Fritz Müller	-2 500	20
...				
20	04 562	Fritz Müller	3 400	30
...	...	...	...	...

**Tabelle 2.15:** Indizierte Tabelle verkettet **Sätze mit gleichem Schlüsselwert**

# Dateiorganisationsformen vergleichen

<i>Kriterium</i>	<i>sequen- tiell</i>	<i>relativ</i>	<i>indi- ziert</i>
Direktzugriff	-	++	++
Löschen/Ändern an Ort	-	++	++
Einfügen an Ort	-	1)	++
Speichereffizienz	++		-
CPU-Belastung	++	++	-
variable Satzlänge	++	2)	++
Magnetbandeignung	++	-	-
freie Schlüsselwahl bei Direktzugriff	-	-	++
> 1 Zugriffsschlüssel bei Direktzugriff	-	-	++
geeignet für formatierte Dateien	++	++	++
geeignet für Textdateien	++	-	-

**Vergleich 2.16:** Sequentielle, relative und indizierte Dateien

---

## 3 Einzeldateien verwalten

---

1 Daten und Dateien

2 Einzeldateien organisieren

⇒ 3 Einzeldateien verwalten

- 4 Datenbanken entwerfen

### ***Datenverwaltung***

Dateiverwaltung (Einzeltabellen verwalten)

Datenbankverwaltung (verbundene Tabellen verwalten)

### ***Operationen***

#### **Datei**

öffnen, erstellen, indizieren, ausgeben, schliessen

#### **Satz**

suchen, löschen, einfügen, ändern





# Unverbundene Dateien - Fall Keene Sentinel

1. Ein **Telefonbenutzer** spricht ein Schlüsselwort zu einem Thema (z.B. zu Wetter oder Börse).
2. Die Applikation sucht die passende Information in **Dateien** (42 Std. Sprachdateien auf einem 486er DOS-PC mit 525 MB).
3. Die Applikation spielt Werbung ein und präsentiert die gefundene **Sprachdatei**.

# Eine Datei als Tabelle interpretieren

## KONTO

<i>Nummer</i>	<i>Inhaber</i>	<i>Typ</i>	<i>Saldo</i>
102	Schmid Willi	Sparkonto	2500
...	...	...	...

**Tabelle 3.1: Beispieltabelle KONTO**

# Eine Tabelle beschreiben

Objekt	Beispiele	Synonym	Analogie
<b>Tabelle</b>	KONTO	Relation	Datei
<b>Zeile</b>	102, Schmid Willi, Sparkonto, 2500	(Tupel)	Satz
<b>Attribut</b>	Nummer, Inhaber, Typ, Saldo	Spalte	Feld
<b>Primärschlüssel</b>	<u>Nummer</u>	Identifikation	-
<b>Sekundärschlüssel</b>	Inhaber	Zugriffsattribut	-

## Übersicht 3.2: Terminologie von Tabellen

# Benutzerschnittstellen unterscheiden

## Befehls- und Menüschnittstellen

<i><b>Befehlsschnittstelle</b></i>	<i><b>Menüschnittstelle</b></i>
schreiben	auswählen
Syntax schwierig	Syntax einfach
Beispiel DOS	Beispiel Windows

## Schnittstelle von MS Access

- **Menüs** wie in MS Excel, MS Word etc.
- **Bewegung** mit Maus oder Cursortasten
- **Query by Example**: Abfragen durch Eintragen von Auswahlbedingungen und Verknüpfungsinformation in Tabellengerüste

# Benutzerschnittstellen bewerten

<i>Kriterium</i>	<b>Befehlsschnittstelle</b>	<b>Menü-</b>
Geschwindigkeit für Experten	+	-
Flexibilität	+	-
Lernfreundlichkeit	-	+
Fehlertoleranz	-	+

Übersicht 3.3: Befehls- und Menüschnittstellen

## 3GL- und 4GL-Sprachen unterscheiden

Kriterium	3. Generation	4. Generation
<i>Beispiele</i>	<b>Cobol</b>	<b>MS Access, dBASE</b>
<i>Entwicklungsaufwand</i>	-	+
<i>Lernfreundlichkeit</i>	-	+
<i>Flexibilität</i>	+	-
<i>Effizienz</i>	+	-

# Die Struktur einer Tabelle entwerfen

<i>Satznummer</i>	<i>Name</i>	<i>Typ</i>	<i>Länge</i>	<i>Dezimalstellen</i>
1	Nummer	Zeichen	6	2
2	Inhaber	Zeichen	20	
3	Typ	Zeichen	1	
4	Saldo	Numerisch	8	

**Tabelle 3.4:** Entwurfsansicht einer dBASE-Tabelle (Benutzereingaben blau)

# Eine Tabelle errichten

## 1. Die Dateistruktur entwerfen

.create **KONTO**

...	Name	Typ	Länge	Dezimalstellen
	Nummer	Zeichen	6	
	Inhaber	Zeichen	20	
	Typ	Zeichen	1	
	Saldo	Numerisch	8	2

## 2. Die Tabelle öffnen

.use **KONTO**

## 3. Einen Satz anfügen

.append

Satznummer :1

Nummer : 10002

Inhaber : Schmid Willi

Typ : s

Saldo : 2500

(s für Sparkonto)

(Benutzereingaben blau)



## Einen Satz anzeigen

<i>Satznummer</i>	1
<i>Nummer</i>	102
<i>Inhaber</i>	Schmid Willi
<i>Typ</i>	Sparkonto
<i>Saldo</i>	2500

**Tabelle 3.5:** Datenblattansicht des ersten Satzes der dBASE-Tabelle KONTO

---

# Sätze suchen

---

## Sequentiell suchen

- `display all`
- `display for Kontentyp = "s" .and.  
Saldo > 0 fields Inhaber, Saldo`

## Direkt suchen

- `display record <Satznummer>`

## Indiziert suchen

- `index on Inhaber+Typ to INHABERINDEX`
- `find Schmid Willi  
display`

310002

Schmid Willi s 2500

---

# Satz und Struktur einer Tabelle ändern

---

## Satz **einfügen**

- `insert` % nach laufendem Satz

## Sätze **löschen**

- `delete` % lauf. Satz zum Löschen markieren
- `delete for` <Bedingung>

Bsp. `delete for Typ = "s" .or. Typ = "a"`

- `pack` % Löschen bestätigen

## Feld **ändern**

- `edit` [ <Satznummer> ] % laufenden oder n-ten Satz

## Sätze **anfügen**

- `create` % neue Satzstruktur und neue Sätze
- `append` % Satz am Dateiende anfügen

## Satz**struktur** ändern

- `modify structure` % neue Satzstruktur

## *Lernziele*


- ⇒ Begriffe Tabelle, Attribut (Feld), Primär- und Sekundärschlüssel
- ⇒ Tabellen, Formulare und Abfragen definieren
- ⇒ Daten eingeben, ändern und sortieren

## *Wiederholungsfragen*

1. Haben Sie schon einmal mit einer Datenbank gearbeitet, wenn ja mit welcher?
  - a) nein
  - b) mit MS-Access
  - c) mit einer anderen PC-basierten Datenbank
  - d) ja, mit einer Server-Datenbank (z.B. Oracle, MS SQL Server)
2. Wie unterscheiden sich Datenbank und Tabelle?
  - a) Eine Datenbank entspricht einer Tabelle.
  - b) Eine Datenbank besteht aus Tabellen.
  - c) Eine Datenbank besteht aus mehreren Dateien.
3. Welche der folgenden Aussagen ist richtig?
  - a) Eine Tabellenzeile entspricht einem Datensatz
  - b) Eine Tabellenspalte entspricht einem Feld
  - c) Eine Tabellenzeile entspricht einem Feld
4. Welchen Schlüssel benötigt eine Tabelle immer?
  - a) Primärschlüssel
  - b) Sekundärschlüssel
  - c) beide Schlüssel

# Vertiefungsfragen

## 1. Tabelle erstellen

- a) Starten Sie Access und erstellen Sie eine leere Datenbank (Datei neu / Datenbank / Speichern unter...).
- b) Definieren Sie Kundennr als *Primärschlüssel* (Rechtsklick auf einen Attributnamen / Primärschlüssel).
- c) Definieren Sie Kundenname als *Sekundärschlüssel* (Entwurfs-sicht, Feldeigenschaften, Indiziert, Ja (Duplikate möglich))
- d) Geben Sie einen beliebigen Satz ein.
- e) Verlassen Sie MS Access, ohne zu speichern. Kopieren Sie dann die Tabelle  [KUNDEN.mdb](#) auf die lokale Festplatte und entfernen Sie den Schreibschutz der Datei. Öffnen Sie die Datenbank und zeigen Sie ein beliebiges Attribut in auf- und absteigender Reihenfolge an (Rechtsklick auf einen Attributnamen / Aufsteigend oder Absteigend).

## 2. Formular erstellen

- f) Öffnen Sie das vordefinierte Kundenformular (Datenbankfenster (F11) / Formulare). Blättern Sie und fügen Sie die folgenden Datensätze ein:

8888, Müller, Malerstr. 88, 4051, Basel, 3000

9999, Wegener, Winterthurerstr. 99, 8015, Zürich, 4000

- g) Erstellen Sie mit dem Formularassistenten ein alternatives Eingabeformular (Datenbankfenster (F11) / Formulare, Neu).

## 3. Tabelle abfragen

- h) Wo wohnt der Kunde Chaplin?

- i) Wo wohnen die Kunden, deren Kreditlimite zwischen 6000 und 9000 liegt?
- j) Wie heissen die Kunden, die in Basel oder Zürich wohnen?
- k) Welche Kunden haben eine Kreditlimite von mindestens 7000 und einen Namen, der mit B oder Z beginnt?

### *Vorgehen bei Abfragen*

- 1) **Tabelle** wählen (Datenbankfenster (F11) / Register Abfragen / Neu / Entwurfsansicht / KUNDE / Hinzufügen / Schliessen)
- 2) **Abfrageattribute** wählen (Doppelklick auf jeden von der Abfrage benötigten Feldnamen)
- 3) **Anzeigeattribute** wählen (entfernen, falls das Attribut (Feld) im Ergebnis nicht angezeigt werden soll)
- 4) **Abfragekriterien** bestimmen (Zeile 'Kriterien:', Vergleichsoperatoren sind  $<$   $>$   $<=$   $>=$ , logischer Operator UND. B\* bedeutet "Texte, die mit B beginnen")
- 5) Abfrage **ausführen** (Ansicht / Datenblattsicht)
- 6) Abfrage allenfalls **speichern** (Datei / speichern)

### *Zusatzaufgaben*

#### *1. Gültigkeitsprüfung*

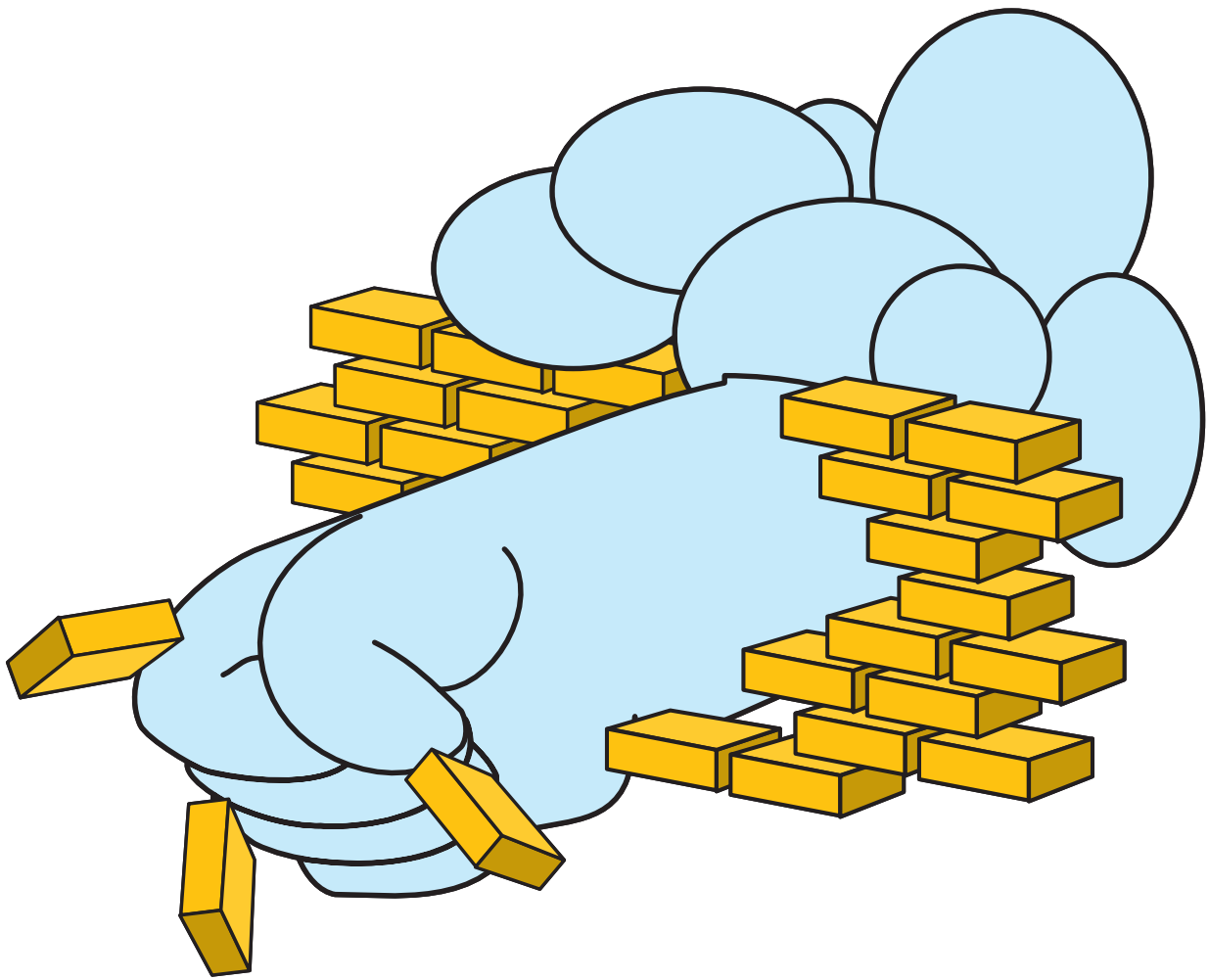
- l) Setzen Sie für das Feld Kreditlimite die Voreinstellung (Standardwert) 100 und die Gültigkeitsregel  $\geq 0$  (Ansicht / Entwurfsansicht / Feldname links klicken, Feldeigenschaften unten)

#### *2. Berechnende Abfragen*

- m) Wie hoch ist der Betrag, den Kunden schulden würden, falls alle ihre Kreditlimite voll ausschöpfen würden?
- n) Wie hoch ist die durchschnittliche Kreditlimite?

### 3. Änderungsabfragen

- o) Setzen Sie die Kreditlimite für alle Kunden auf 1000.
- p) Die Kreditlimite der Kunden aus Basel soll um 50% erhöht werden.
- q) Löschen Sie alle Kunden, die in Zürich wohnen.



## Import / Export von Dateien

## Zwischenablage (engl. clipboard)

- Copy
- Paste
- Move

## OLE (Object Linking and Embedding)

## DDE (Dynamic *Data* Exchange)



# °Zwischenablage

## Beispiel

1. Eine Adresse in der **Datenbankanwendung** markieren
2. Die markierte Adresse in die **Zwischenablage** (Clipboard) kopieren
3. Einen Brief im **Textverarbeitungssystem** öffnen
4. Die Adresse aus der **Zwischenablage** in den Brief einfügen

Beispiel 3.7: Einen Datensatz in einen Text kopieren

## Begriff

- **Zweck:** Daten **vorübergehenden** ablegen
- **Operationen:** **Ausschneiden**, **Kopieren** und **Einfügen** (cut, copy and paste)
- **Dauer:** bis zum **nächsten** Ausschneiden oder Kopieren

## °Zwischenablage - Verknüpfen und Einbetten

<i>Kommunikation</i>	<i>Objekt</i>	<i>Bearbeitbar im DBMS</i>	<i>Transfer</i>
Kopieren	Kopie	nein	einfach
Verschieben	Original	nein	einfach
Verknüpfen	Verweis auf Produzent und Original	ja	aufwendig
Einbetten	Kopie und Verweis auf Produzent	ja	aufwendig

**Übersicht 3.6:** Kommunikation mit anderen Anwendungen

# °Verknüpfen und Einbetten

## Beispiel einer Verknüpfung

1. Der Benutzer eines **Datenbanksystems** aktiviert durch einen **Doppelklick** auf dem Feld Abstract eines beliebigen Datensatzes das verknüpfte **Textprogramm**.
2. Er kann nun den Feldinhalt - ohne den Kontext des Datenbanksystems zu verlassen - mit der **gewohnten Oberfläche** des Textprogramms bearbeiten.
3. Er verlässt schliesslich das Textprogramm. Der bearbeitete Text ist nun **Teil der Datenbank**.

Beispiel 3.8: Datenfeld mit Textdokument verknüpfen

## Begriff

### **Verknüpfung**

Ein Feld **verweist** auf die erstellende Anwendung, ohne den Feldinhalt zu kopieren.

### **Einbettung**

Ein Feld **kopiert** das Ergebnis der produzierenden Anwendung, ermöglicht aber weiterhin seine Änderung mit dem Produzenten.

# °Dynamisch Daten austauschen (DDE)

## Beispiel

1. Client Word **öffnet** einen Kanal zum Server Access
2. Word **verlangt** ein bestimmtes Datenelement von Access
3. Server Access **sendet** den gewählten Adressatz an Word
4. Word **integriert** die Adresse in einen Briefkopf
5. Word **schliesst** den Kanal zu Access

## Begriff

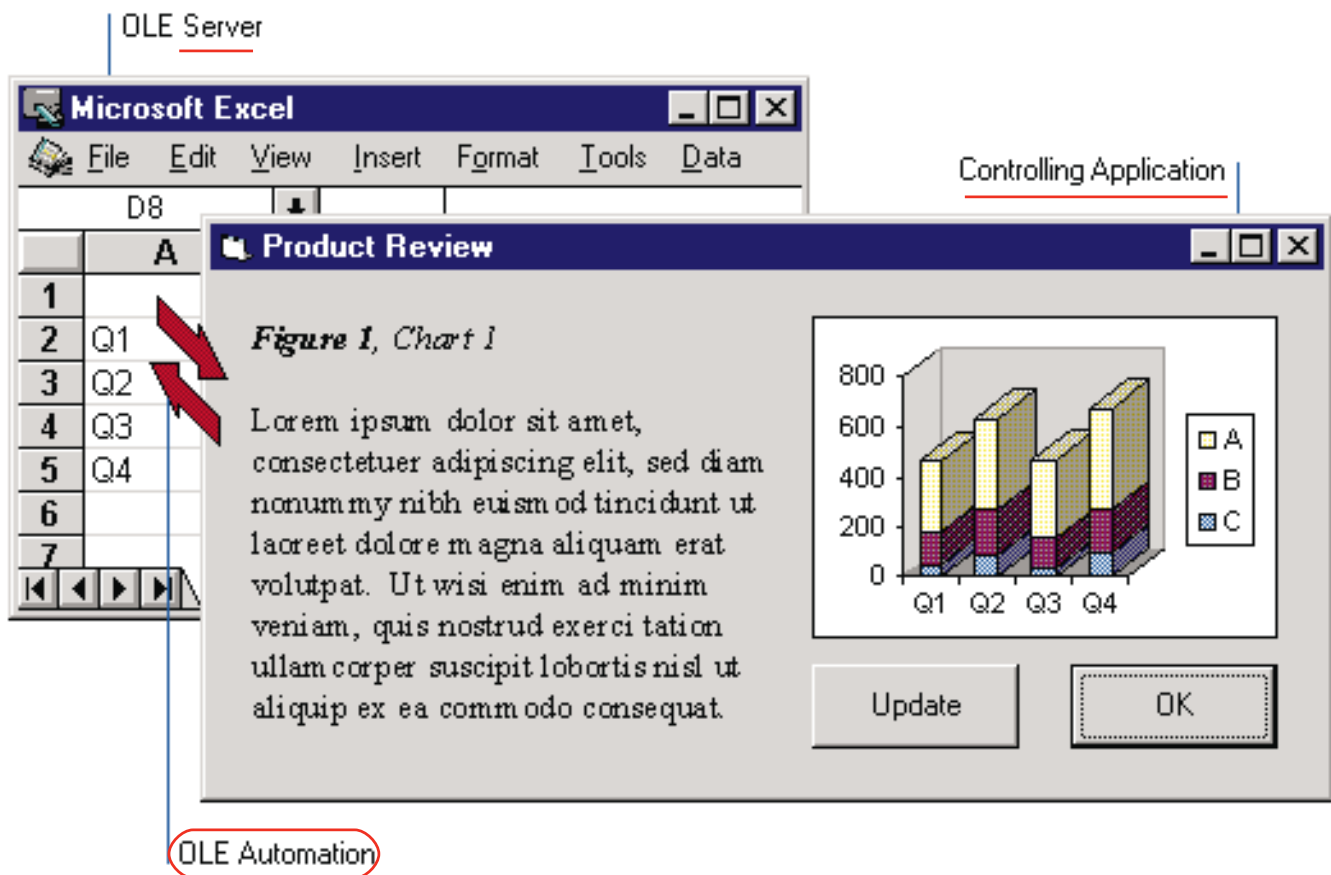
- Client und Server tauschen gleichzeitig **Daten oder Befehle**
- Client und Server **auf gleichem oder verschiedenen** Rechnern
- Datenaustausch dynamisch, weil ein Client **jederzeit neue** DDE-Verbindungen öffnen und schliessen kann

## °Zusammenfassung - Zwischenablage und OLE

<i>Kommunikation</i>	<i>Objekt</i>	<i>Bearbeitbar im DBMS</i>	<i>Transfer</i>
Kopieren	Kopie	nein	einfach
Verschieben	Original	nein	einfach
Verknüpfen	Verweis auf Produzent und Original	ja	aufwendig
Einbetten	Kopie und Verweis auf Produzent	ja	aufwendig

**Übersicht 3.6:** Kommunikation mit anderen Anwendungen

# °Objektkommunikation - Automatisierung



Serveranwendung

Steueranwendung (engl. client)

## Begriff

Der **Programmierer** verknüpft eine andere Anwendung, ohne

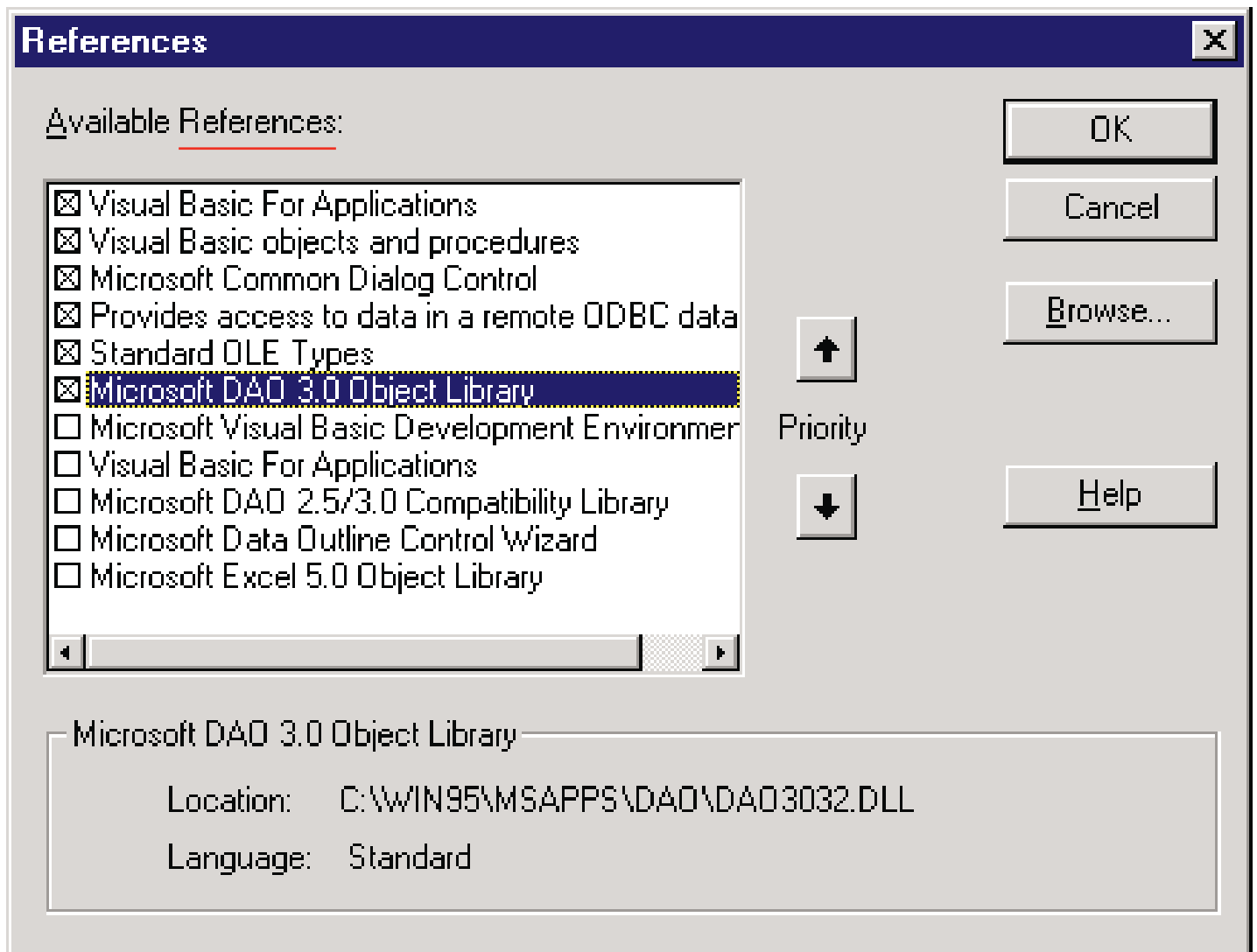
- den **Quellcode** der anderen Anwendungen zu kennen
- den **Objektcode** aus einer Programmbibliothek zu binden

## Beispiel

Ein Datenerfassungsprogramm liest Börsendaten per Modem in eine **Datenbank**. Dann verarbeitet ein **Tabellenkalkulationsprogramm** einen Teil der Datensätze und gibt die Ergebnisse an ein **Datenanalyseprogramm** weiter. Dieses erstellt mit Hilfe eines neuronalen Netzes eine Börsenvorhersage.

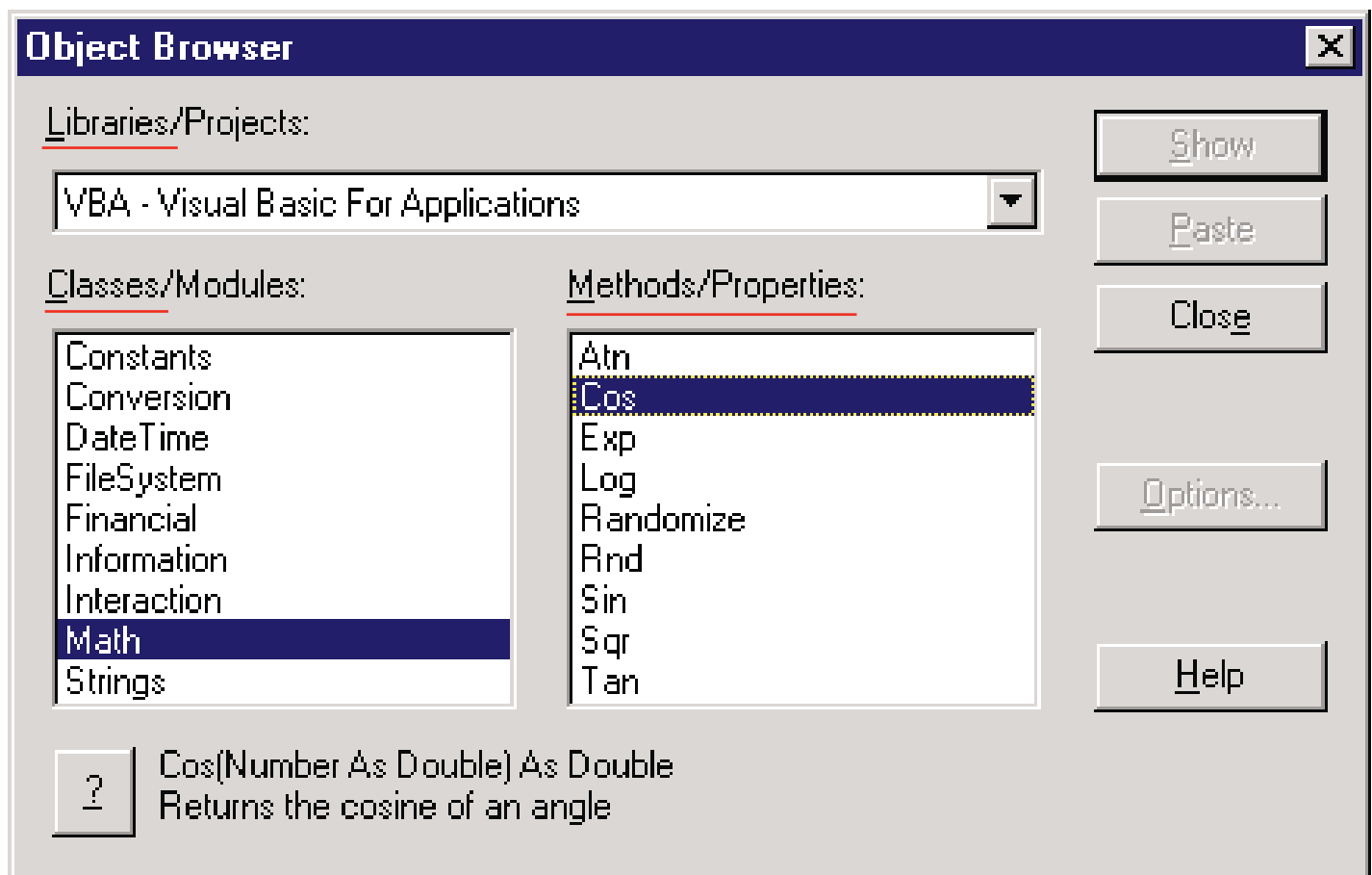
### Beispiel 3.9: OLE-Automatisierung

## °Automatisierung - Objektbibliothek wählen





## °Automatisierung - Objektbrowser konsultieren



# °Automatisierung - Programmbeispiel

```
sub zeigeVerkaufsbericht()  
  ` Access-Instanz deklarieren  
  dim AcInstanz As Object  
  
  ` Instanz erzeugen und Objektvariablen zuweisen  
  set AcInstanz = CreateObject("Access.Application")  
  
  ` Instanz-Eigenschaften und -Methoden einsetzen  
  AcInstanz.Visible = True  
  ` Datenbank VERKAUF.MDB öffnen  
  AcInstanz.OpenCurrentDatabase "c:\db\VERKAUF.MDB"  
  ` Bericht Verkaufszahlen drucken  
  AcInstanz.DoCmd.OpenReport "Verkauf",acPreview  
end sub
```

## Programm 3.10: Automatisierung aus Visual Basic

---

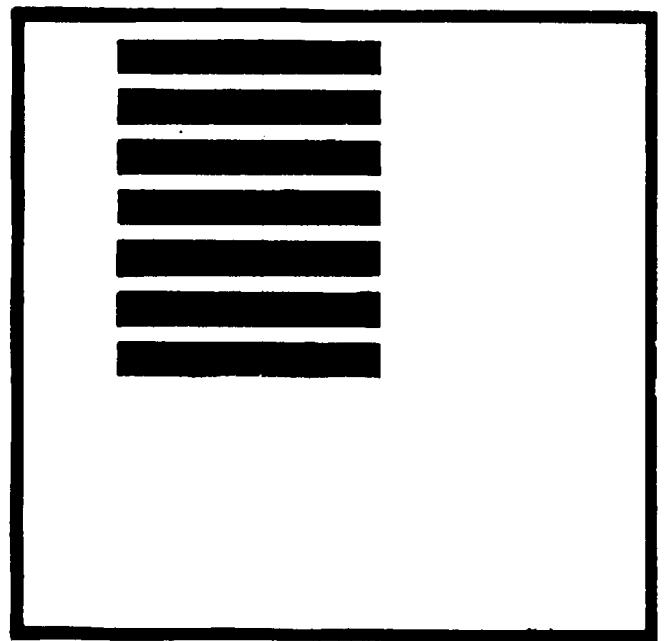
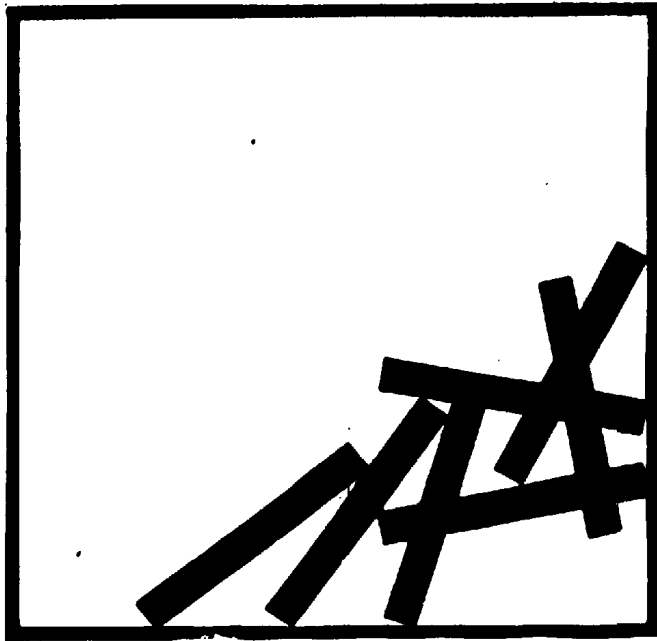
## °Mit Anwendungen kommunizieren (A 3.5)

---

Bestimmen Sie die Art der Kommunikation mit der anderen Anwendung. Begründen Sie ihre Antwort.

- a) Sie speichern das **Logo einer Lieferantenfirma**.
- b) Jeder Bericht soll das **eigene Firmenlogo** enthalten.
- c) Sie speichern eine MS **Excel-Tabelle in einem Verkaufsbericht**.
- d) Die Personaltabelle soll das **Bild jedes Mitarbeiters** enthalten.
- e) Jedes Mitarbeiterformular soll den (in einem Textverarbeitungsprogramm erstellten) **Lebenslauf** enthalten.

## 4 Daten entwerfen



Eine Untersuchung an einer amerikanischen Universität ergab, dass Studierende während des Studiums die Matrikelnummer **mehr als 300mal** nennen mussten. Fast jedes Mal mussten sie zusammen mit der Matrikelnummer Daten wie Vorname, Name und Semester angeben. Insgesamt zählte man **99 Dateien**, welche neben zusätzlicher Information all diese Daten speicherten.

Beispiel 4.1: Redundante Daten

## Abfragen auf ...

- einer Tabelle      **Datei**verwaltung
- mehreren Tabellen      **Datenbank**verwaltung

---

## 4 Datenentwurf

---

⇒ 4 Datenbanken entwerfen

- 5 Datenbanken verwalten
- 6 Relationale Datenbanken
- 7 Datenbankanwendungen entwickeln

### *Tabellenmodell*

Dateien als Tabellen

Felder als Spalten

Sätze als Zeilen

### *Entwurfsziele*

minimale Redundanz

minimale Datenabhängigkeit

### *Datenbankstrukturdiagramm*

1 : 1

1 : m

m : n

### *Entwurfsanomalien*

Einfüge -

Lösch -

Änderungs -

### *Normalisierungsschritte*

erster -

zweiter -

dritter -

# Tabelle

## Beispiel

### STUDENTEN

<u>Matrikelnr</u>	Name	Studienrichtung
17'853	Meier	BWL
...	...	...

## Schema (Skelett)

### TABELLENNAME

<u>Identifikation</u>	Attribut	...
Wert	...	...
...	...	...

**Datei = TABELLE, falls**

- jede Zeile eindeutig
- in jeder Zelle nur **ein** Wert
- jede Zeile **gleich lang**

# Erstes Entwurfsziel - Minimale Redundanz

**Implementationsbeschränkungen**  
(Sätze pro Datei, *Felder* pro Satz, *Zeichen* pro Satz)

**Notwendigkeit mehrerer Tabellen**  
+  
Tabellenmodell

## Redundanz

Je mehr **Felder** ein Satz enthält, desto eher kommt ein Teil des Satzes **auch in anderen Sätzen** der gleichen Tabelle vor.

<i>Name</i>	<i>Dozent</i>	<i>Abteilung</i>
Informatik II	Müller	BWL
Organisationslehre I	Müller	BWL
...	...	...

Redundanz minimieren durch ...

**sorgfältigen Datenentwurf !**



---

## Zweites Ziel - Minimale Datenabhängigkeit

---

Attributwert ändert sich  
(z.B. Kundenadresse)

*Mehrere Dateien ändern,  
weil ihre Daten redundant*

Dateiaufbau ändert sich  
(z.B. Adressaufbau)

*Mehrere Programme ändern  
weil von den Daten abhängig*

**Sorgfältiger Datenentwurf !**

# Normalisierung

Minimiere **Redundanz und Datenabhängigkeit**

+

Stelle Daten in **Tabellen** dar

Vermeide **Anomalien**

# Eine Datei ist nicht unbedingt eine Tabelle ...

## VERANSTALTUNG

<i>Name</i>	<i>Stunden</i>	<i>Dozent</i>	<i>Raumnummer</i>	<i>Plätze</i>	<i>Semester</i>
Recht	2	Meier	111, <b>112</b>	200, <b>150</b>	SS93, <b>SS94</b>
BWL	4	Schmid	111	200	WS94, <b>SS95</b>
VWL	3	Müller	112	150	SS94, <b>WS95</b>

**Tabelle 4.2:** Attribute mit **Wiederholungsgruppen**

1. Normalisierungsschritt

# Erster Normalisierungsschritt (1. Normalform)

**VERANSTALTUNG** als *Datei* mit variablen Satz­längen

<u>Name</u>	Stunden	Dozent	Raumnr	Plätze	Semester
Recht	2	Meier	111, <b>112</b>	200, <b>150</b>	SS87, <b>SS88</b>
BWL	4	Schmid	111	200	WS88, <b>SS89</b>
VWL	3	Müller	112	150	SS88, <b>WS89</b>

**VERANSTALTUNG** als *Relation* mit festen Satz­längen

<u>Name</u>	Stunden	Dozent	Raumnr	Plätze	<u>Semester</u>
Recht	2	Meier	111	200	SS87
Recht	2	Meier	<b>112</b>	<b>150</b>	<b>SS88</b>
BWL	4	Schmid	111	200	WS88
BWL	4	Schmid	111	200	<b>SS89</b>
VWL	3	Müller	112	150	SS88
VWL	3	Müller	112	150	<b>WS89</b>

1. Neue **Zeilen** einfügen
2. **Schlüssel** erweitern

## ① Wiederholungsgruppen eliminieren

"Tabelle" mit  
Wiederholungsgruppen

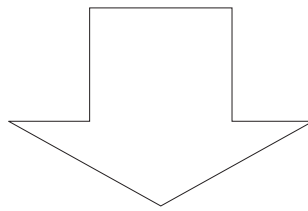


Tabelle im Sinne des  
Relationenmodells

**Bild 4.12:** 1. Normalisierungsschritt: Wiederholungsgruppen aufteilen

# Anomalien nach erstem Schritt

## VERANSTALTUNG

<u>Name</u>	<u>Semester</u>	<u>Stunden</u>	...	...	...
Recht	SS87	2 wird zu 3			
Recht	SS88	2 bleibt aus Versehen			
...	...	...			
Informatik	?				

Vermeide **Änderungs-** und **Einfügeanomalien**

Definiere VERANSTALTUNG neu !

1. Erwähne den **Stunden**wert nur einmal pro Zeile
2. Identifiziere VERANSTALTUNG nur mit Name

Theorie der **Abhängigkeiten** zwischen Attributen

2. Normalisierungsschritt ?

# Anomalien entstehen durch Abhängigkeiten

<u>Name</u>	<u>Semester</u>	<u>Stunden</u>	...	...	...
Recht	SS87	2 wird zu 3			
Recht	SS88	2 bleibt aus Versehen			
...	...	...			
Informatik	?				

Änderungs- und Einfügeanomalie

## Beispiel

*Name* bestimmt *Stunden*

*Stunden* ist abhängig von *Name*

*Name*    *Stunden*

Wie vermeide ich Anomalien ?

## Terminologie

- *A* bestimmt *B*
- *B* ist **abhängig** von *A*
- *A*    *B*

## Abhängigkeiten (A 4.6)

---

- a) Kundennr      Kundenname?
- b) Kundenname      Kundenadresse?
- c) Angestelltenr      Abteilungsnr?
- d) Raumnummer      Platzzahl?
- e) Vorlesungsname      Dozentenname?

Attribut	Attribut
----------	----------



### ③ Teilschlüssel-Abhängigkeit abspalten

#### VERANSTALTUNG

<u>Name</u>	<u>Semester</u>	<u>Stunden</u>	...	...	...
Recht	SS87	2 wird zu 3			
Recht	SS88	2			
...	...	...			
Informatik	?				

Änderungs- und Einfügeanomalie

#### VERANSTALTUNG

<u>Name</u>	<u>Stunden</u>
Recht	2
BWL	4
VWL	3

+

#### ORGANISATION (Restanomalie)

<u>Name</u>	<u>Semester</u>	<u>Raumnr</u>	<u>Plätze</u>	<u>Dozent</u>
Recht	SS87	111	200	Meier
Recht	SS88	112	150	Meier
BWL	WS88	111	200	Schmid
BWL	SS89	111	200	Schmid
VWL	SS88	112	150	Müller
VWL	WS89	112	150	Müller

## Ergebnis nach zweitem Schritt (2. Normalform)

### VERANSTALTUNG (Abspaltung von VERANSTALTUNG)

<u>Name</u>	<u>Stunden</u>
Recht	2
BWL	4
VWL	3

### ORGANISATION (Rest von VERANSTALTUNG)

<u>Name</u>	<u>Semester</u>	<u>Raumnr</u>	<u>Plätze</u>	<u>Dozent</u>
Recht	SS87	111	200	Meier
Recht	SS88	112	150	Meier
BWL	WS88	111	200	Schmid
BWL	SS89	111	200	Schmid
VWL	SS88	112	150	Müller
VWL	WS89	112	150	Müller

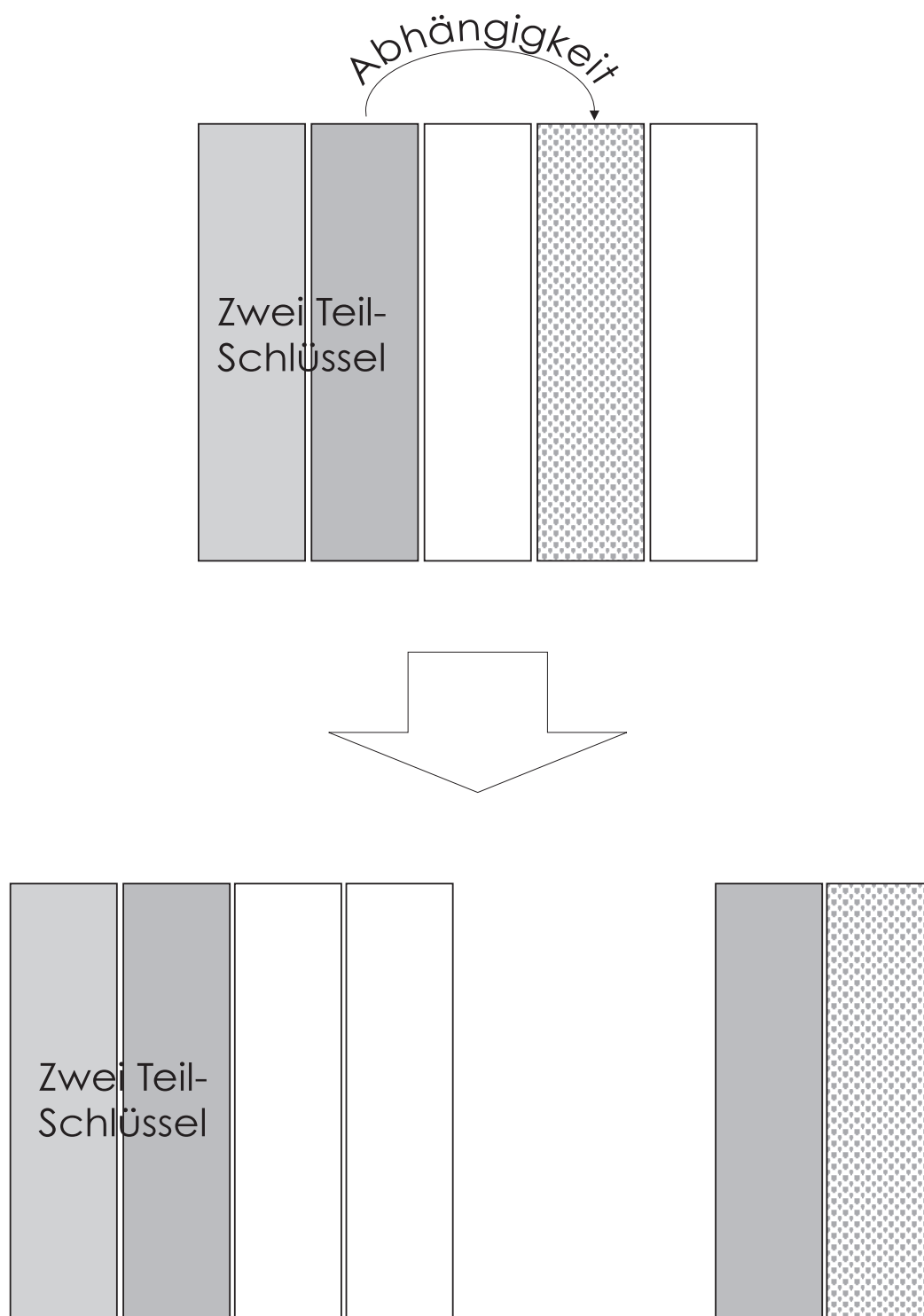
- Eine neue Veranstaltung bewirkt keine Einfügeanomalie mehr
- Eine Stundenänderung bewirkt keine Änderungsanomalien mehr

**aber ...**

- Ein **neuer Raum** bewirkt eine Einfügeanomalie
- Ein **geänderter Name** bewirkt eine Änderungsanomalie

3. Normlisierungsschritt ?

## ② Teilschlüssel-Abhängigkeiten abspalten



**Bild 4.13:** 2. Normalisierungsschritt: Teilschlüssel-Abhängigkeiten abspalten

### ③ Nichtschlüssel-Abhängigkeit abspalten

#### ORGANISATION

<u>Name</u>	<u>Semester</u>	Raumnr	Plätze	Dozent
Recht	SS87	111	200	Meier
Recht	SS88	112	150	Meier
BWL	WS88	111	200	Schmid
BWL	SS89	111	200	Schmid
VWL	SS88	112	150	Müller
VWL	WS89	112	150	Müller

#### RAUM

<u>Raumnummer</u>	Plätze
111	200
112	150

+

#### ORGANISATION (Restanomalie)

<u>Name</u>	<u>Semester</u>	Dozent	<u>Raumnummer</u>
Recht	SS87	Meier	111
Recht	SS88	Meier	112
BWL	WS88	Schmid	111
BWL	SS89	Schmid	111
VWL	SS88	Müller	112
VWL	WS89	Müller	112

## Ergebnis nach dem 3. Schritt (3. Normalform)

### ORGANISATION (Rest von ORGANISATION)

<u>Name</u>	<u>Semester</u>	Dozent	Raumnummer
Recht	SS87	Meier	111
Recht	SS88	Meier	112
BWL	WS88	Schmid	111
BWL	SS89	Schmid	111
VWL	SS88	Müller	112
VWL	WS89	Müller	112

### RAUM (Abspaltung von ORGANISATION)

<u>Raumnummer</u>	Plätze
111	200
112	150

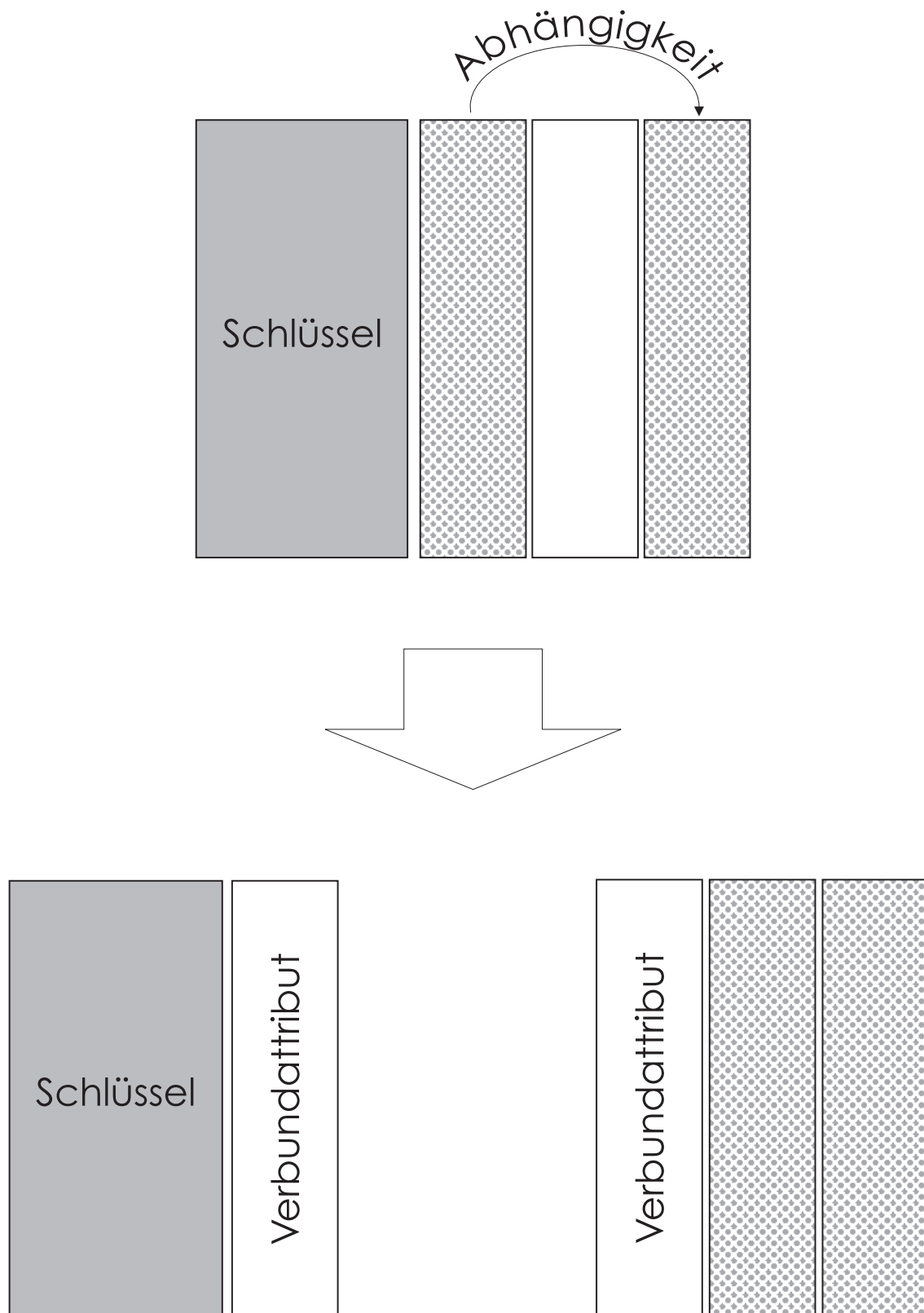
- Ein neuer Raum bewirkt *keine* Einfügeanomalie mehr
- Ein geänderter Veranstaltungsname bewirkt *keine* Update-Anomalie mehr

**aber ...**

- Ein **zusammengesetzter Schlüssel** kann noch eine Anomalie enthalten

Weitere Normalisierungsschritte ?

### ③ Nichtschlüssel-Abhängigkeiten abspalten



**Bild 4.16:** 3. Normalisierungsschritt: Nichtschlüssel-Abhängigkeiten abspalten

# Beziehungen zwischen Tabellen(sätzen)

## VERANSTALTUNG

Eine Vorlesung wird

Zu **einem** Vorlesungssatz passen **mehrere** Organisationssätze

1 :

## ORGANISATION

in mehreren Semestern von mehreren Dozenten/innen gelesen

m

## RAUM

Ein Raum dient

Zu **einem** Raum passen

1 :

## ORGANISATION

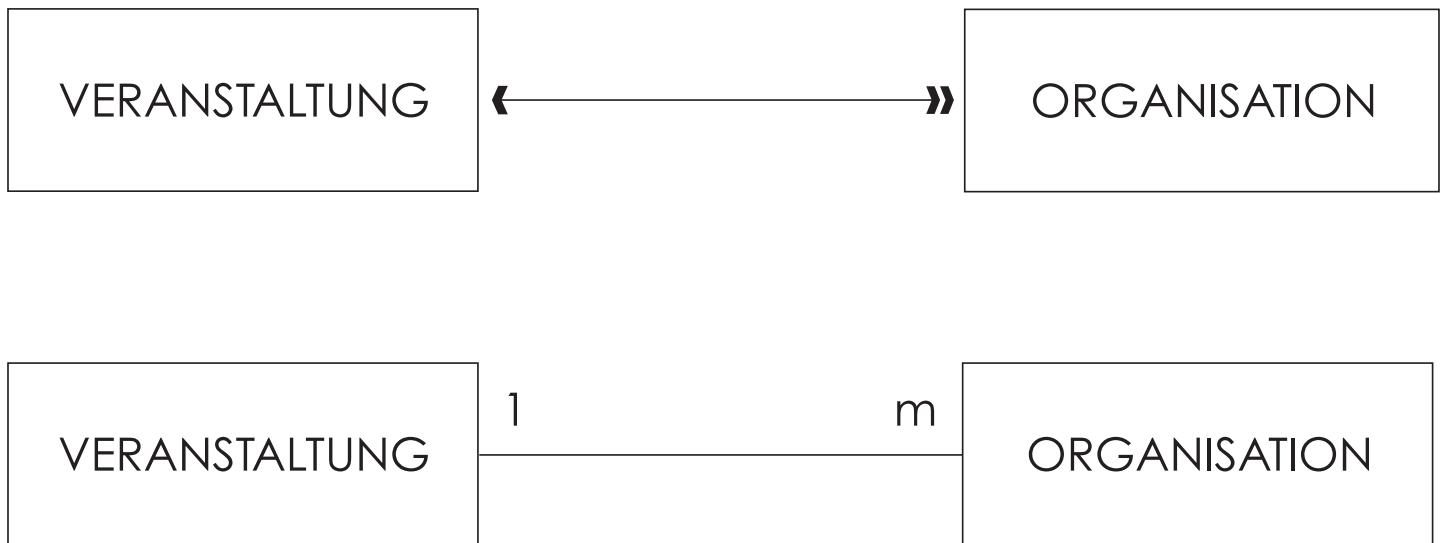
mehreren Dozenten/innen in mehreren Semestern

**mehrere** Organisationssätze

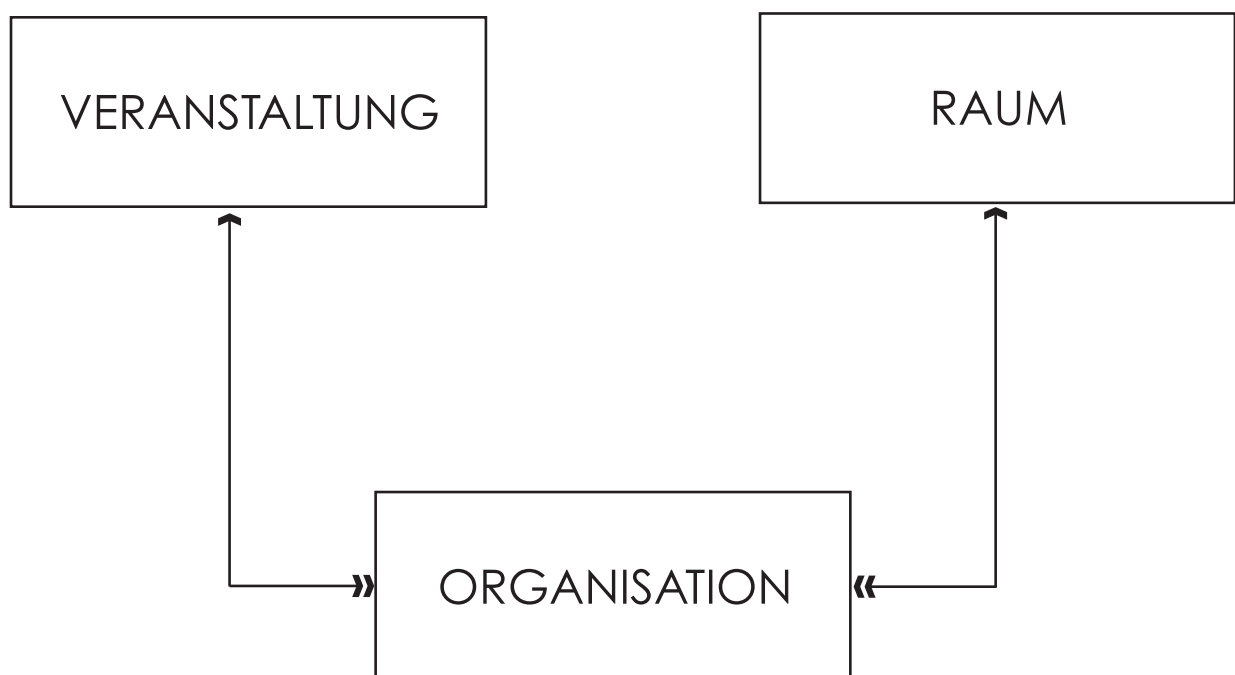
m

**Abhängigkeit ist *nicht* Beziehung !**

## Tabellenbeziehungen **graphisch** darstellen



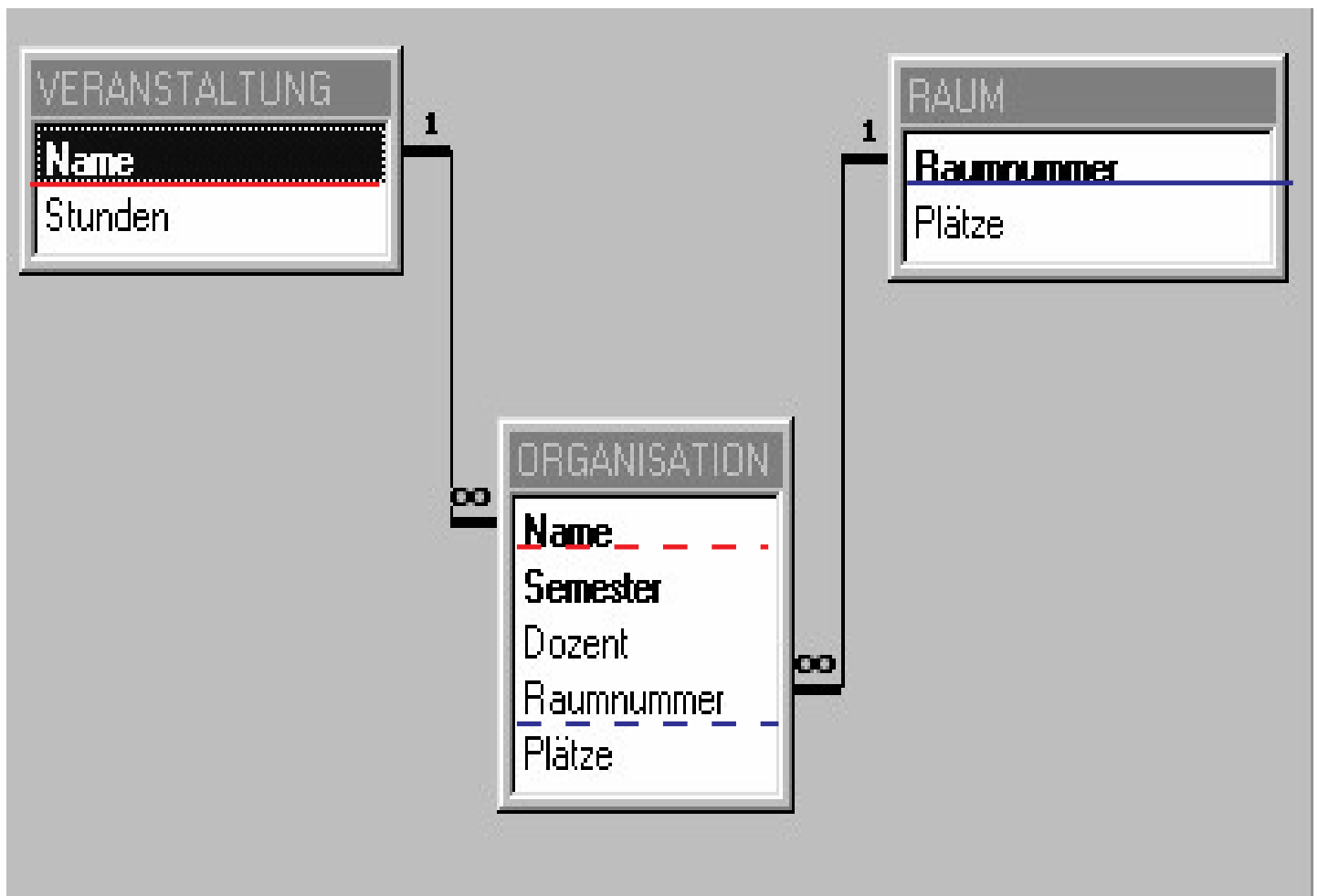
**Bild 4.8:** 1:n-Beziehung zwischen den Tabellen 4.4 und 4.5



**Bild 4.9:** 1:n-Beziehungen zwischen den Tabellen 4.4, 4.6 und 4.7



# Beziehungen in **MS Access** graphisch darstellen



**Bild 4.11:** Datenbankstrukturdiagramm von Bild 4.9 in MS Access

Beziehungen werden ...

- durch übereinstimmende Daten
- in meist gleich benannten Attributen (**Verbundattributen**)
- zwischen einer Ersttabelle und einer Zweittabelle

hergestellt

## 1:n - Beziehung als häufigster Beziehungstyp

VERANSTALTUNG (**HAUPT**TABELLE)

<u>Name</u> ( <b>Primärschlüssel</b> )	<i>Stunden</i>
<b>Recht</b>	2
BWL	4
VWL	3



ORGANISATION (**DETAIL**TABELLE)

<u>Name</u> ( <b>Fremd</b> schlüssel)	<u>Semester</u>	<i>Dozent</i>	<i>Raumnummer</i>
<b>Recht</b>	SS87	Meier	111
<b>Recht</b>	SS88	Meier	112
BWL	WS88	Schmid	111
BWL	SS89	Schmid	111
VWL	SS88	Müller	112
VWL	WS89	Müller	112

“Schlüssel” meint Primärschlüssel  
und nicht Sekundärschlüssel

# Beziehungstypen

Beziehung	Ersttabelle (... : ...)	Zweitabelle (... : ...)
<b>1 : 1</b>	<b>1 Zeile</b>	<b>1 Zeile</b>
1 : c	1 Zeile	0 oder 1 Zeile
<b>1 : m</b>	<b>1 Zeile</b>	<b>1 oder mehrere Zeilen</b>
1 : mc	1 Zeile	0, 1 oder mehrere Zeilen
<b>m : n</b>	<b>1 oder mehrere Zeilen</b>	<b>1 oder mehrere Zeilen</b>

Übersicht 4.10: Beziehungstypen (Fette Typen werden automatisch unterstützt)

- Eine 1:1-Beziehung lässt sich in einer *einzigsten* Tabelle darstellen
- In MS Access kann eine 1:m-Beziehung 1:m *oder* 1:mc sein
- Eine m:n-Beziehung erfordert eine (dritte) *Verbindungstabelle* (und besteht deshalb aus zwei 1:m-Beziehungen)

Kontrolliert das **Datenbanksystem** oder der **Programmierer** die Einhaltung eines Typs?

## Beziehungen (A 4.2)

### 1:1, 1:m, oder m:n?

- a) Mutter - Kind
- b) Dozent - Student
- c) Ehemann - Ehefrau
- d) AHV-Nummer - AHV-Versicherter
- e) Kunde - Bestellung

Dies sind Tabellen und nicht Attribute!

---

## Datenbank Zoo (Interpretation von Text 4.3)

---

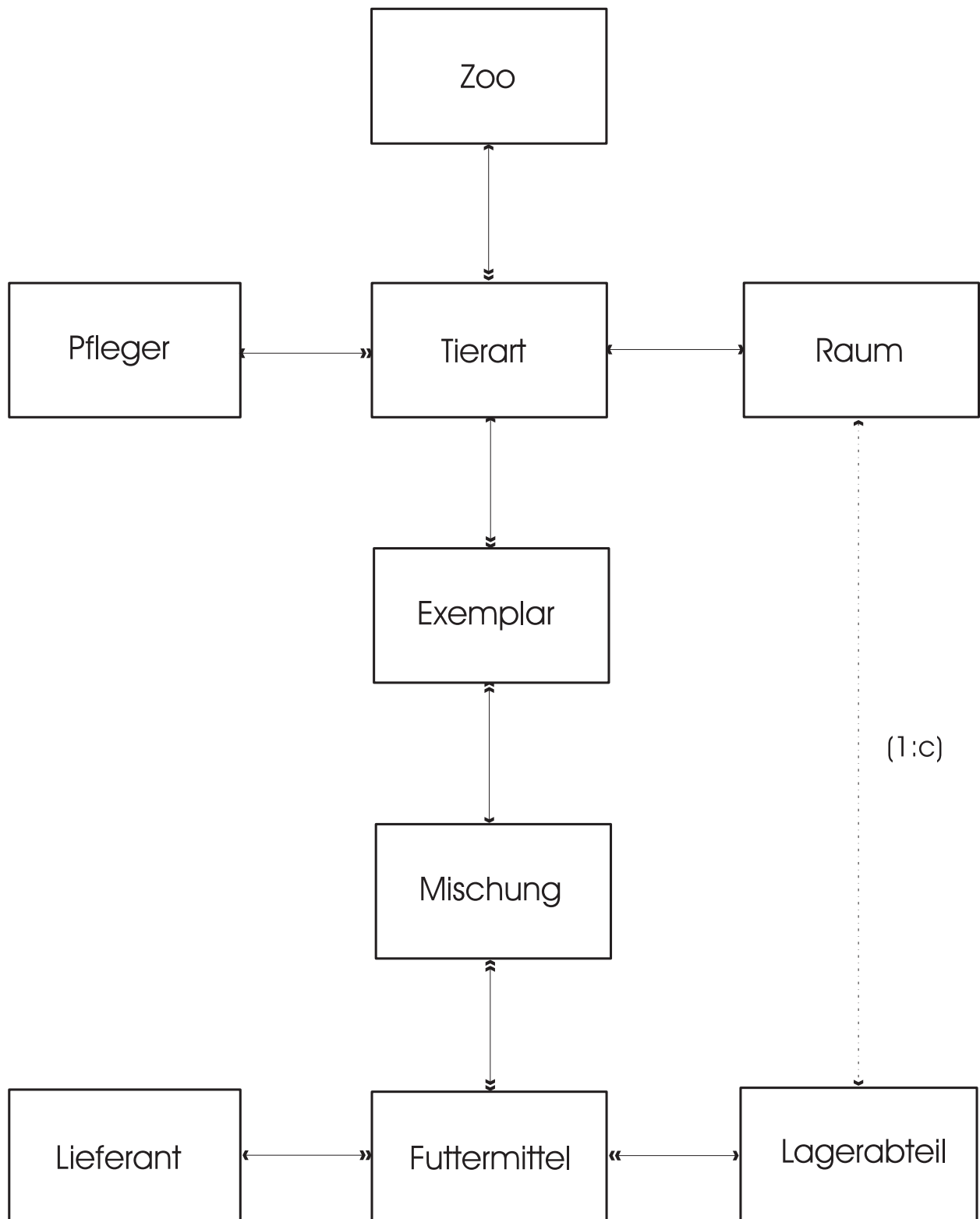
- ein Zoo - mehrere Tierarten
- eine Tierart - mehrere Exemplare
- eine Tierart - ein Pfleger
- eine Tierart - eine Futtermittelmischung
- eine Mischung - Futtermittel mehrerer *Lieferanten*
- eine Tierart - ein Raum
- ein Raum mit keinem oder einem Futtermittel-Abteil

+

Plausible Annahmen

Datenbankstrukturdiagramm

# Datenbank Zoo (Lösung 4.3)



# Beziehungen und Abhängigkeiten - Beispiele

## a) Beziehungen am Beispiel Hochschule

Beziehung	Haupttabelle	Detailtabelle	Bedeutung
1:m	VERANSTALTUNG	ORGANISATION	Eine Veranstaltung kann mehrere Organisationsformen annehmen
1:m	RAUM	ORGANISATION	Ein Raum kann in mehreren Organisationsformen vorkommen

Tabelle 4.14: Beziehungen zwischen den normalisierten Hochschultabellen

## b) Abhängigkeiten am Beispiel Hochschule

Bestimmendes Attr.	Bestimmtes Attr.	Bedeutung
Name	Stunden	Mit dem Namen einer Veranstaltung ist immer auch deren Stundenzahl bekannt
Raumnummer	Plätze	Ein bestimmter Raum hat eine bestimmte Anzahl Plätze

Tabelle 4.15: Abhängigkeiten in der Datenbank Hochschule

# Integritätsbedingungen

**Integritätsbedingung** :=

Regel, welche die *Integrität* (Vollständigkeit und Richtigkeit) einer Datenbank mit garantiert

## Wichtige Integritätsbedingungen

**Entitäts**integrität:

Jede Tabellenzeile trägt eine *Identifikation*

MS Access schreibt die Definition eines Primärschlüssels vor

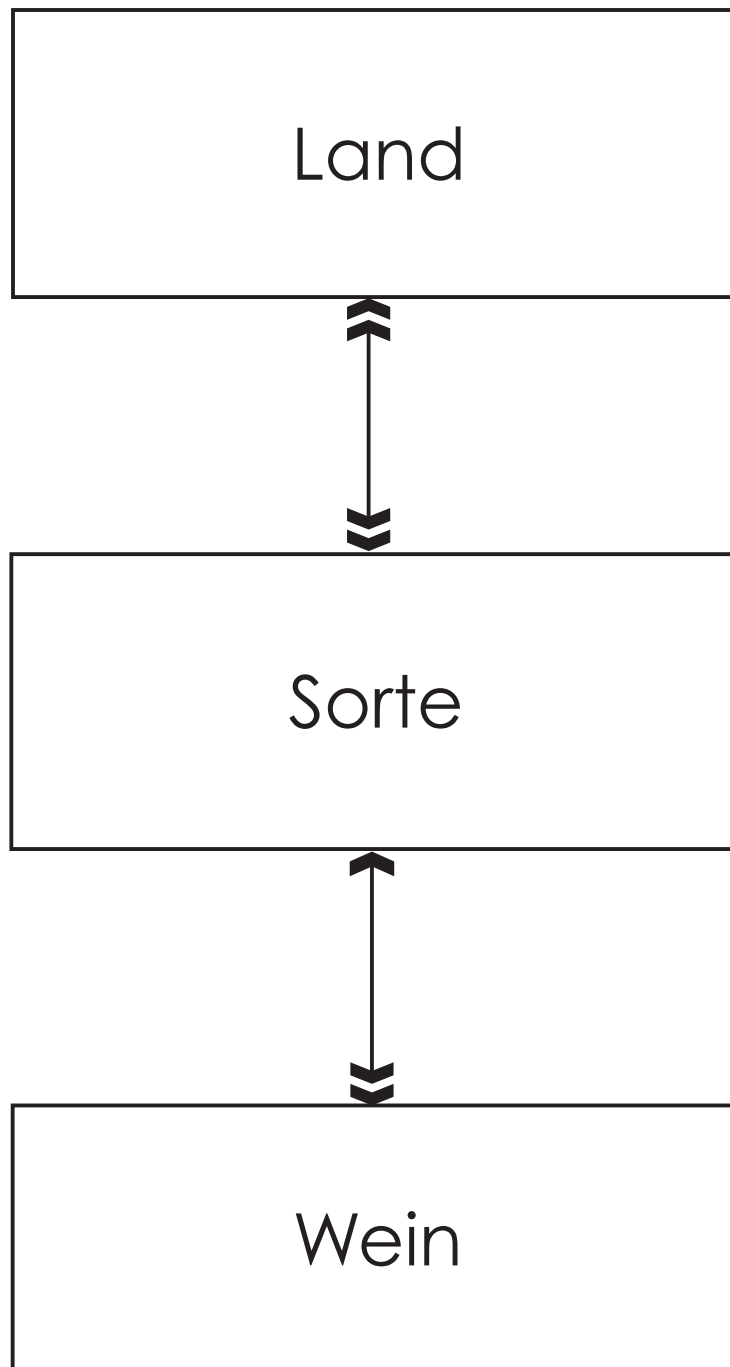
**Referentielle** Integrität:

Jeder *Fremdschlüsselwert* verweist auf eine existierende Zeile einer anderen Tabelle

MS Access erlaubt die Definition der referentiellen Integrität

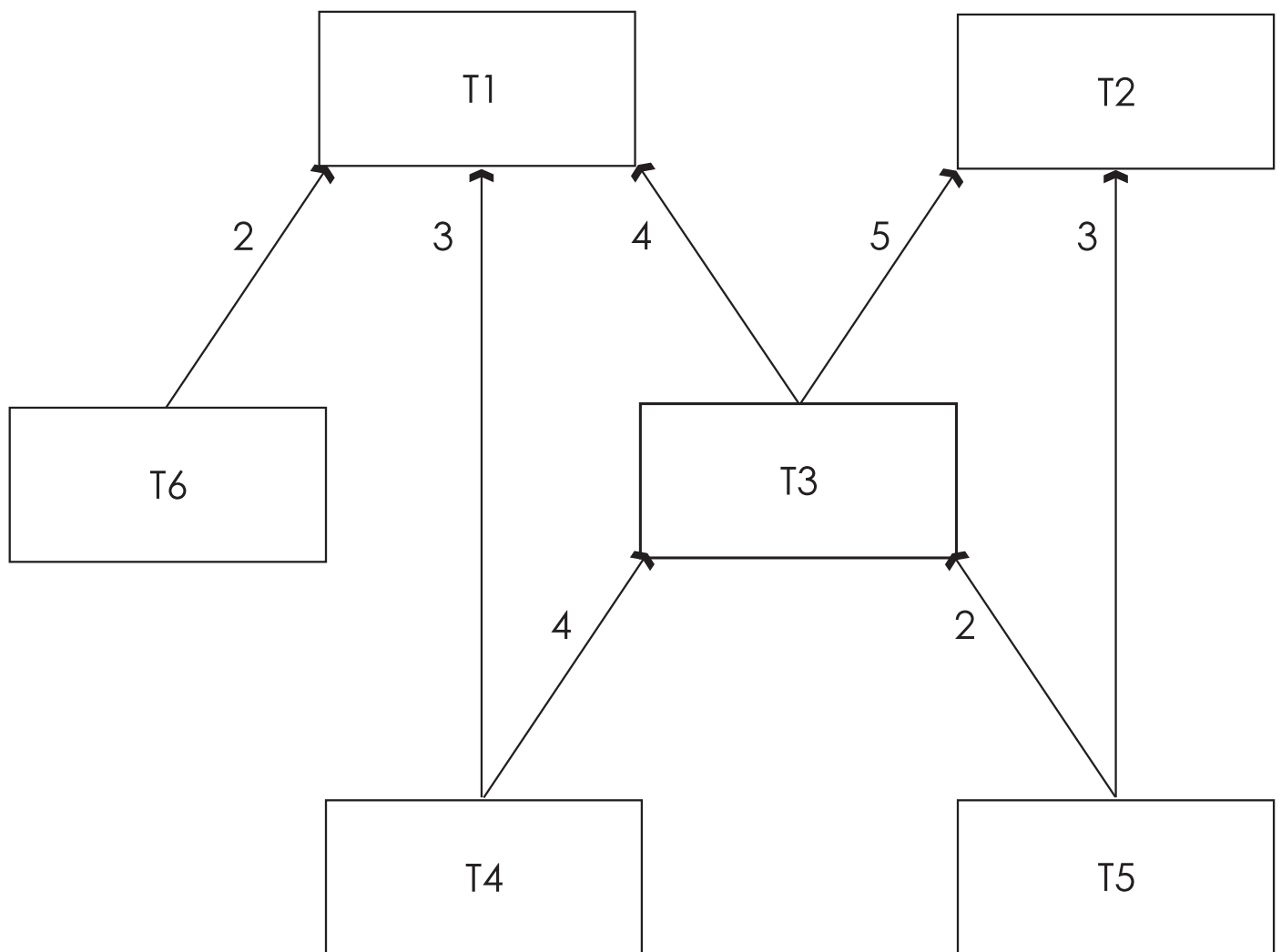


## °Beziehungen am Beispiel Wein



**Bild 4.17:** Datenbankstrukturdiagramm Weinkeller

## °Rekursive Beziehungen graphisch darstellen



**Bild 4.18:** Eine einfache Stückliste



**Bild 4.19:** Die Stückliste als rekursive m:n-Beziehung

# Phasen des Datenentwurfs - 1. Teil

- ① Erfasse die Benutzeranforderungen **verbal**
- ② Transformiere die Anforderungen in **Tabellen**

*a) Erstelle eine **Tabelle** für jeden Objekttyp*

Bsp. ANGESTELLTE, ABTEILUNGEN

*b) Bestimme **Identifikationschlüssel** (Primärschlüssel)*

Bsp. ANGESTELLTE(Personalnummer, ...)

*c) Füge weitere **Attribute** an*

Bsp. ANGESTELLTE(..., Name, Adresse, Mitarbeiterzahl, Gehalt)

## Phasen des Datenentwurfs - 2. Teil

### d) Verbinde die Tabellen durch *Fremdschlüssel*

- **1:1** - Beziehung

TABELLE1(Identifikation1, ..., Identifikation2)

TABELLE2(Identifikation2, ..., Identifikation1)

- **1:m** - Beziehung

TABELLE1(Identifikation1, ..., Identifikation2)

TABELLE2(Identifikation2, ...)

- **m:n** - Beziehung

TABELLE1(Identifikation1, ...)

TABELLE2(Identifikation2, ...)

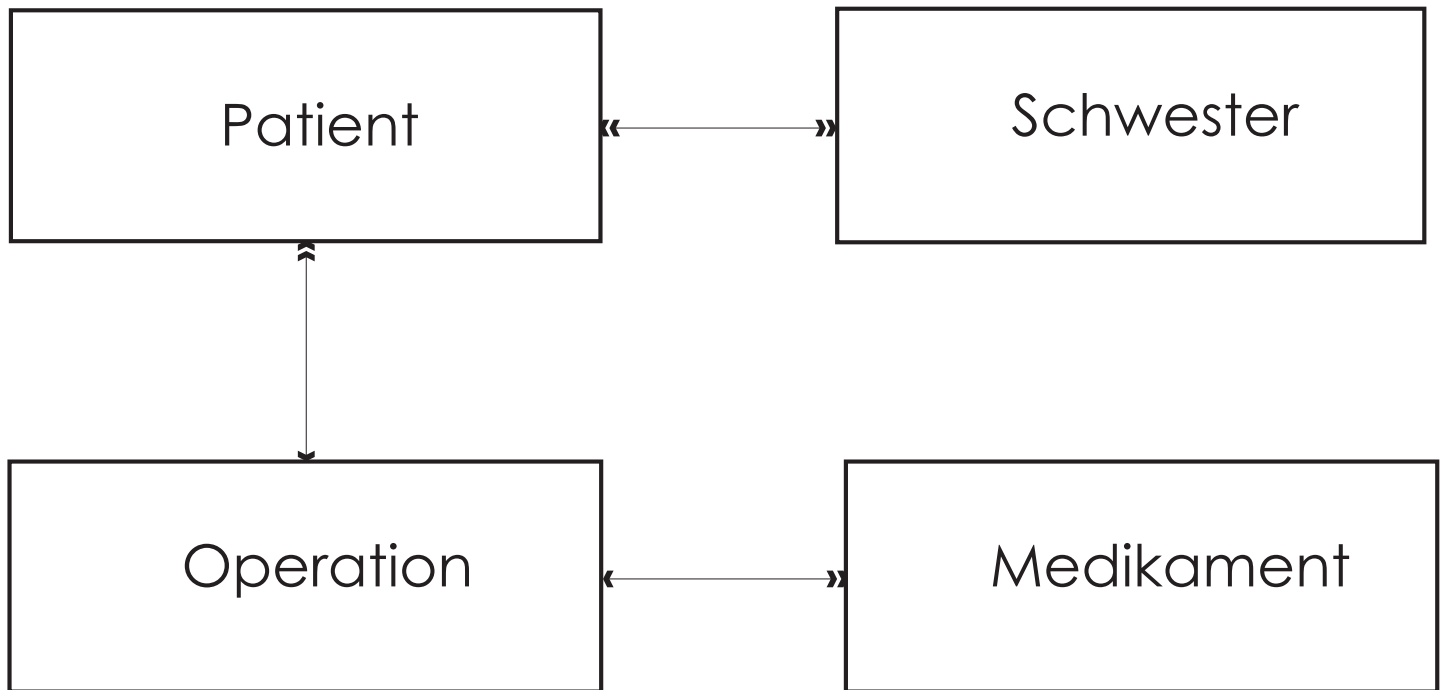
VERBINDUNG(Identifikation1, Identifikation2, ...)

### ③ Normalisiere die Tabellen

### ④ Bestimme die Wertebereiche der Attribute

- Bsp. Abteilungsnummer < 50
- Bsp. AHV-Nummer mit Prüfziffer

## °Datenbank Krankenhaus (Lösung 4.15)



## Datenbank Oel (A 4.8)

Oelfirma	Sitz	Direktor	Bohrloch	Ort	Bohrfirma
Westgas	Houston	Jones	23	Alaska	Drillpete
Multoil	Calgary	Smith	39	Yukon	Deepdrill
Eastpete	Chicago	Hind	41	Mexico	Borefirm
Westgas	Houston	Jones	47	Texas	Gasrig
Multoil	Calgary	Smith	58	Alaska	Drillpete
Westgas	Houston	Jones	62	North Sea	Deepdrill
Eastpete	Chicago	Hind	74	Texas	Gasrig

*Aus dem Aufgabentext ...*

- Jedes Loch wird an einem bestimmten Ort gebohrt
- Jedes Loch wird von genau einer Firma gebohrt

*Aus der Tabelle ...*

- Jede Oelfirma hat genau einen Sitz und einen Direktor

*Aufgabe*

- a) Wo Redundanzen ?
- b) Normalisiere
- c) Datenbankstrukturdiagramm

*b) Normalisierung*

1. Schlüssel ?
2. Regelwidrige Abhängigkeiten ?
3. Tabelle spalten
4. Datenbankstrukturdiagramm

# Datenbank Oel (Lösung 4.8)

## a) Redundanzen

Sitz, Direktor (von Oelfirma abhängige Teilschlüssel)

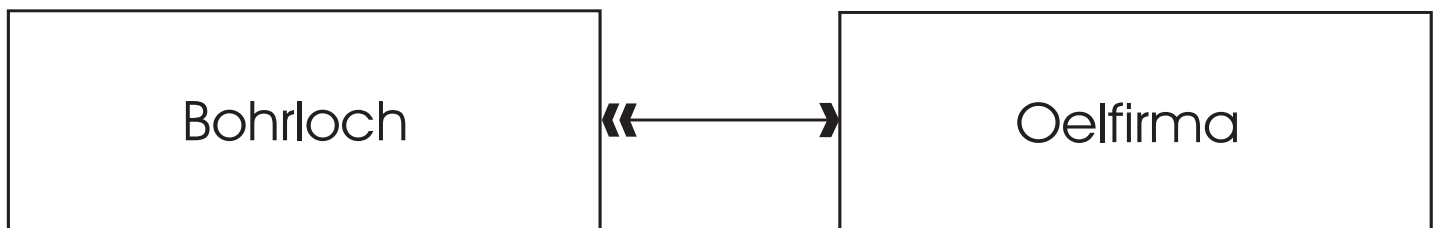
## b) Tabellenspaltung

OELFIRMA(Oelfirma, Sitz, Direktor)

BOHRLOCH(Loch, Ort, Bohrfirma, Oelfirma)

(Fremdschlüssel)

## c) Datenbankstrukturdiagramm



## Unterscheide !

1. Beziehungen zwischen **Tabellen**
2. Abhängigkeiten zwischen **Attributen**

Oelfirma	Sitz
----------	------



## °Datenbank KUNDEN (A 4.9)

### *Daten mit Wiederholungsgruppen*

<i>Bestellnr</i>	<i>Kundennr</i>	<i>Name</i>	<i>Bestelldatum</i>	<i>Produktnr</i>	<i>Menge</i>
<b>158 204</b>	<b>28 000</b>	<b>Meier</b>	<b>01-08-95</b>	<b>21 405</b>	<b>24</b>
158 204	28 000	Meier	01-08-95	<b>21 510</b>	<b>18</b>
158 204	28 000	Meier	01-08-95	<b>44 020</b>	<b>4</b>
158 204	28 000	Meier	01-08-95	<b>56 785</b>	<b>18</b>
158 204	28 000	Meier	01-08-95	<b>80 702</b>	<b>12</b>
<b>158 205</b>	<b>88 000</b>	<b>Tobler</b>	<b>02-08-95</b>	<b>21 405</b>	<b>12</b>
158 205	88 000	Tobler	02-08-95	<b>32 840</b>	<b>2</b>
158 205	88 000	Tobler	02-08-95	<b>44 020</b>	<b>8</b>
158 205	88 000	Tobler	02-08-95	<b>62 320</b>	<b>24</b>

### *Aufgabe*

In drei möglichst redundanzarme Tabellen spalten

### *Vorgehen*

1. **Erste** Normalform mit Schlüssel
2. Regelwidrige **Abhängigkeiten**?
  - a) Teilschlüssel-
  - b) Nichtschlüssel-
3. **Aufspaltung** mit Schlüssel
4. **Verbindungsattribut(e)**?

---

## °Datenbank Wälder (A 4.14)

---

Eine Datenbank soll Informationen über die Wälder einer Region enthalten. Sie enthält die folgenden Attribute:

- **Wald** (Name des Waldes)
- **Ort**
- **Eigentümer**
- **Art** (In einem Wald können mehrere Arten vorkommen, zum Beispiel Tanne und Fichte)
- **Höhe** (maximale Höhe einer Baumart)
- **Baumnr** (Einige Wälder sind Versuchswälder für Untersuchungen über den Einfluss der Luftverschmutzung. In diesen Wäldern werden geeignete Bäume ausgewählt und mit einer Nummer identifiziert)
- **Pflanzdatum** des Baumes
- **Messnr** (Von jedem Versuchsbaum werden in regelmässigen Abständen Messproben entnommen)
- **Messergebnis** (Der Inhalt der Messung ist in unserem Zusammenhang unwichtig)
- **Messdatum**

# °Datenbank Wälder (Lösung 4.14)

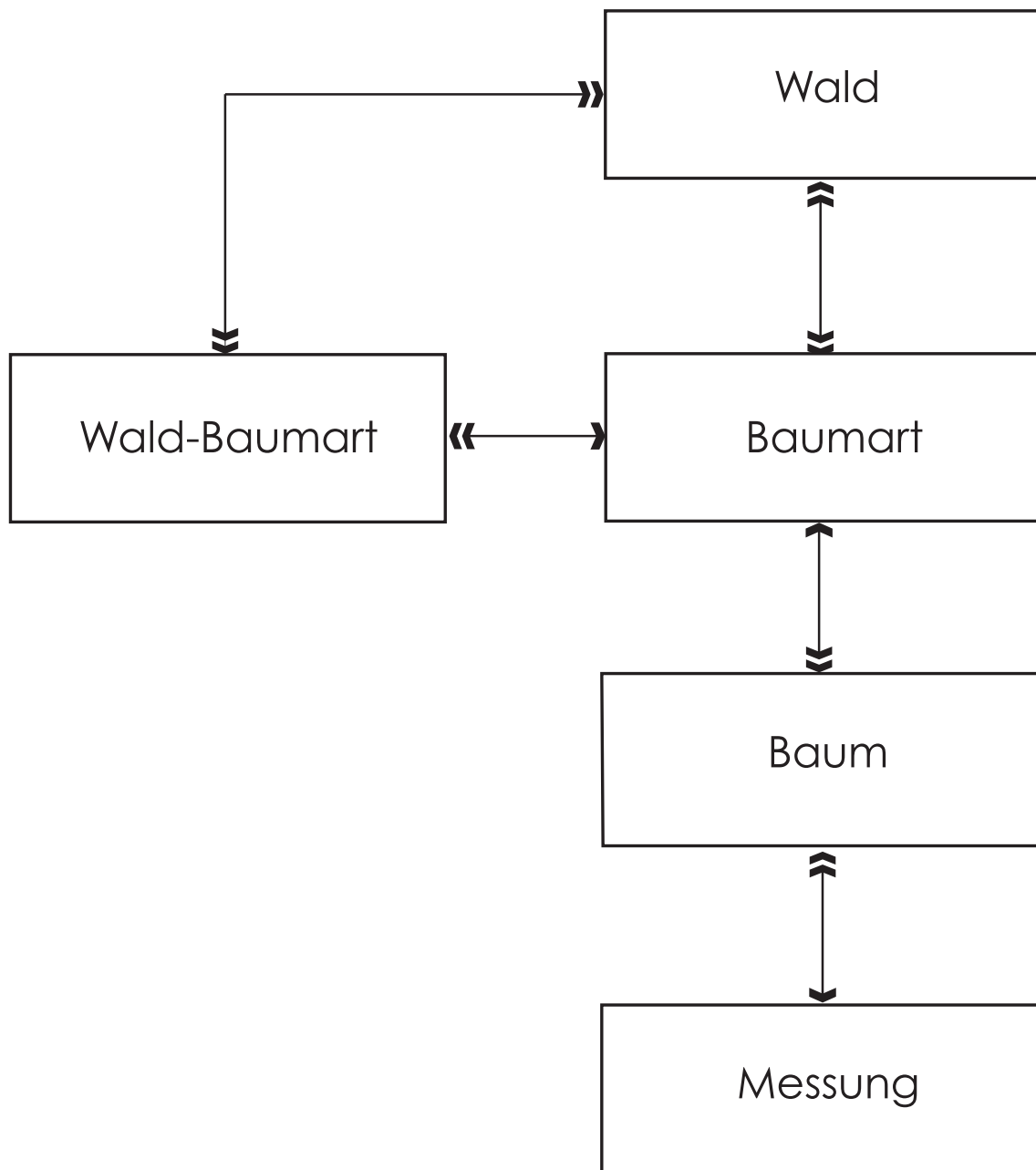
**WALD**(Name, Ort, Eigentümer)

**BAUMART**(Art, Höhe)

**VERBINDUNG**(Name, Art)

**BAUM**(Baumnr, Name, Art, Pflanzdatum)

**MESSUNG**(Messnr, Baumnr, Messergebnis, Messdatum)



*Data Base Design can be simple, but talking about is frequently not. In simple cases at least, it is often easier just to do the design than it is to try and explain exactly what it was you did.*

*C. J. Date*



## *Lernziele*

- ⇒ funktionale Abhängigkeit
- ⇒ erste bis dritte Normalform
- ⇒ Verbund
- ⇒ Datenbankstrukturdiagramm

## *Wiederholungsfragen*

1. Was ist Ziel der Normalisierung?
  - a) Redundanz maximieren
  - b) Datenunabhängigkeit minimieren
  - c) Einfüge-, Lösch und Änderungsanomalien vermeiden
2. Kennzeichen von Tabellen in erster Normalform sind ...
  - a) Wiederholungsgruppen
  - b) gleich lange Tabellenspalten
  - c) Tabellenzellen mit höchstens einem Wert
3. Welche Mängel haben Tabellen in zweiter Normalform?
  - a) Abhängigkeiten zwischen Nichtschlüsselattributen
  - b) Wiederholungsgruppen
  - c) Abhängigkeiten von einem Teilschlüssel

## *Vertiefungsfragen*

Die Kaliba AG handelt mit Kleidern und Accessoires. Die Artikel werden aus Asien importiert. Das Auftragsformular enthält das *Datum*, die *Auftragsnr.*, den *Auftragsnamen* und für jeden bestellen Artikel die *Artikelmenge* und den *Vereinbarungspreis*. Ausserdem nennt es die

*Kundennr* und den *Kundennamen*. Diese stehen zusammen mit der *Strasse*, der *PLZ*, dem *Ort* und die *Kreditlimite* auch in der Kundenkartei (vgl. 📄 KUNDEN). Eine weitere Kartei enthält schliesslich die *Artikelnr*, den *Artikelnamen*, den *Katalogpreis* und den *Lagerbestand* jedes Artikels. Der Artikelname ist - im Gegensatz zur Artikelnr nicht immer eindeutig.

## 1. Entwerfen Sie eine normalisierte Datenbank

- a) Erstellen Sie aus den erwähnten Attributen eine Ausgangstabelle in erster Normalform. Markieren Sie die Attribute mit *Wiederholungsgruppen* rot und bestimmen Sie den Identifikationsschlüssel.

*Ausgangstabelle in erster Normalform zusammenstellen*

.....

.....

.....

.....

- b) Überführen Sie die Tabelle in die zweite Normalform.

*Teilschlüssel-Abhängigkeiten identifizieren*

.....

.....

.....

*Daraus Tabellen in zweiter Normalform ableiten*

.....

.....

.....

.....

- c) Führen Sie die dritte Normalform ein. Markieren Sie die Verbundattribute grün.

*Nichtschlüssel-Abhängigkeiten identifizieren*

.....

.....

.....

*Daraus Tabellen in 3. Normalform mit Verbundattributen ableiten*

.....

.....

.....

- d) Zeichnen Sie ein *Datenbankstrukturdiagramm*

.....

.....

.....

.....

.....

.....

## 2. Passen Sie die Datenbank an Änderungen an

- e) Da sich Kleiderpreise oft ändern, notieren die Mitarbeiter für jeden Artikel das *Änderungsdatum* des Preises. Normalisieren Sie wenn nötig ein zweites Mal.

.....

.....

.....

.....

- f) Damit die Spedition weiss, ob ein Kunde eine Lastwagenzufahrt hat, kann ein Kunde zusätzlich zur Rechnungsadresse eine *Lieferadresse* und ein boolsches Attribut *LKW-Zufahrt* haben.

.....

.....

.....

.....

## Zusatzaufgabe

Physisch sind die Artikel auf die Lagerorte Rheinhafen, Flughafen und Zentrallager verteilt. Jeder Lagerort hat eine Adresse aus einem eindeutigen *Lagernamen*, einer *Strasse*, einer *PLZ* und einem *Ort*. Ein Artikel ist durch die Kombination von *Lagername*, *Hallennr*, *Gangnr* und *Fachnr* auffindbar.

Wichtig ist nicht nur der *Lagerbestand* jedes Artikels pro Lagerort, sondern auch eine Teilmenge davon, der *Vormerkbestand*. Dies ist der für Kunden reservierte, aber noch nicht ausgelieferte Bestand.

- g) Integrieren Sie die neuen Attribute in normalisierter Form.

.....

.....

.....

.....

.....

.....

.....

.....



---

## 5 Datenbanken verwalten

---

4 Datenbanken entwerfen

⇒ 5 Datenbanken verwalten

- 6 Relationale Datenbanken
- 7 Datenbankanwendungen entwickeln

① *Tabellen* **verbinden**

② *Tabellen* **abfragen**

*Abfragesprachen*

prozedurale -

nichtprozedurale -

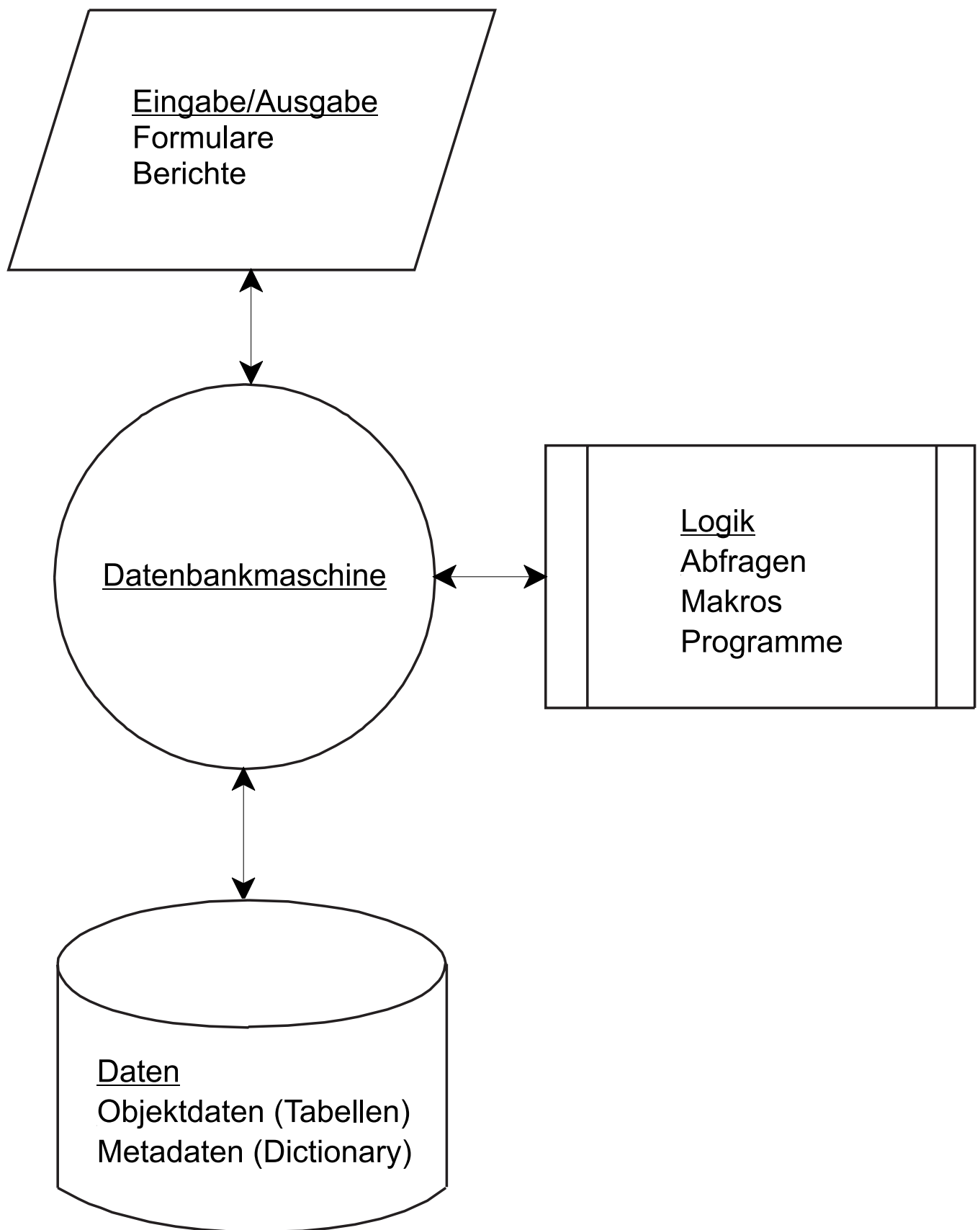
*Query by Example* als nichtprozedurale Sprache

③ *Tabellen* **ändern**

Beispiel **dBASE**

Beispiel **QBE**

# Datenbanken computergestützt verwalten



**Bild 5.1:** Aufbau einer Datenbankanwendung

# Datenbankobjekte verwalten

<i>Datenbankobjekt</i>	<i>Verwaltungswerkzeug</i>
<b>Tabelle</b>	Datenentwurfs- und -definitionshilfe
<b>Formular</b>	Formulargenerator
<b>Bericht</b>	Berichtsgenerator
<b>Abfrage</b>	Abfragesprache
<b>Programm, Makro</b>	Applikationsgenerator, Programmier- und Makrosprache

Übersicht 5.2: Datenbankobjekte und ihre Werkzeuge

# Normalisierung der Datenbank Hochschule

## VERANSTALTUNG

<u>Name</u>	<u>Stunden</u>
Recht	2
BWL	3
VWL	3

## ORGANISATION

<u>Name</u>	<u>Semester</u>	<u>Dozent</u>	<u>Raumnummer</u>
Recht	SS93	Meier	111
Recht	SS94	Meier	112
BWL	WS94	Schmid	111
BWL	SS95	Schmid	111
VWL	SS95	Müller	112
VWL	WS95	Müller	112

## RAUM

<u>Raumnummer</u>	<u>Plätze</u>
111	200
112	150

**Tabellen 5.3:** Normalisierte Datenbank Hochschule (Verbundattribute farbig)

# Tabellen spalten und verbinden

Die Normalisierung **spaltet** Tabellen

und

der Verbund **vereinigt** sie wieder

## Abfragen

- in **einer** Tabelle sind problemlos
- in **mehreren** Tabellen benötigen Verbundoperationen

# Die Normalisierung erschwert Abfragen

## VERANSTALTUNG

<u>Name</u>	<u>Stunden</u>
<b>Recht</b>	<b>2</b>
BWL	4
VWL	3

## ORGANISATION

<u>Name</u>	<u>Semester</u>	<u>Dozent</u>	<u>Raumnummer</u>
Recht	SS93	Meier	111
Recht	SS94	Meier	112
<b>BWL</b>	WS94	<b>Schmid</b>	111
<b>BWL</b>	SS95	<b>Schmid</b>	111
VWL	SS95	Müller	112
VWL	WS95	Müller	112

## RAUM

<u>Raumnummer</u>	<u>Plätze</u>
111	<b>200</b>
112	150

## Abfragen auf **einer** und **mehreren** Tabellen

- a) **Stundenzahl** der Veranstaltung **Recht** ?
- b) **Plätze** der Räume für **BWL**-Vorlesungen von **Schmid**?

# Abfragen erfordern Verbundoperationen

## ORGANISATION

<i>Name</i>	<i>Semester</i>	<i>Dozent</i>	<i>Raumnummer</i>
Recht	SS87	Meier	111
Recht	SS88	Meier	112
BWL	WS88	Schmid	111
BWL	SS89	Schmid	111
VWL	SS88	Müller	112
VWL	WS89	Müller	112

+

## RAUM

<i>Raumnummer</i>	Plätze
111	200
112	150

## VERBUND

<i>Name</i>	<i>Semester</i>	<i>Dozent</i>	<i>Raumnummer</i>	Plätze
Recht	SS87	Meier	111	200
Recht	SS88	Meier	112	150
BWL	WS88	Schmid	111	200
BWL	SS89	Schmid	111	200
VWL	SS88	Müller	112	150
VWL	WS89	Müller	112	150

---

# Verbund in dBASE

---

*Verbundattribut*

*übrige Attribute*

## Operanden

ORGANISATION (Name, Semester, Dozent, Raumnummer)

RAUM (Raumnummer, Plätze)

## Ergebnis

VERBUND (Name, Semester, Dozent, Raumnummer, Plätze)

## Operation

**use** RAUM

**select** 2

**use** ORGANISATION

**join with** RAUM **to** VERBUND ;

**for** RAUM->Raumnummer = Raumnummer ;

**fields** Name, Semester, Dozent, Raumnummer, RAUM->Plätze

**use** 1. Tabelle im ersten Speicherbereich

**select** zweiter Speicherbereich

**use** 2. Tabelle

**join with** 1. Tabelle **to** Ergebnistabelle ;

**for** Verbundbedingung ;

**fields** Attribute der Ergebnistabelle



# Verbund prozedural und deklarativ

## a) Prozedurale Abfrage (dBASE)

```
use RAUM
select 2
use ORGANISATION
join with RAUM to VERBUND;
for RAUM->Raumnummer = Raumnummer;
fields Name, Semester, Dozent,
Raumnummer, RAUM->Plätze;
```

## b) Deklarative Abfrage (QBE)

ORGANISATION

Name	Semester	Dozent	Raumnummer
			x

RAUM

Raumnummer	Plätze
x	

Vergleich 5.11: Prozedurale und deklarative Verbundanweisung

---

# Abfragen alternativ formulieren

---

## *Natürlichsprachlich*

Welche Eigenschaften charakterisieren die Wälder von Oberwil?

## *Prozedural* (dBASE)

```
use WALD
% -- temporär zwischenspeichern
copy to TMP
use TMP
delete for Gemeinde <> 'Oberwil'
% -- Duplikate eliminieren
total on WALD to ERGEBNIS
display all
```

## *Deklarativ* (SQL)

```
SELECT *
FROM WALD
WHERE Gemeinde = "Oberwil"
```

Beispiel 5.5: Natürlichsprachliche, prozedurale und deklarative Abfragen

---

# Abfragesprachen

---

## Natürlichsprachliche -

Syntax natürlich

Reihenfolge unbedeutend

Bsp. INTELLECT

## Formalsprachliche -

### Prozedurale -

Syntax formal

Reihenfolge wichtig

Bsp. Befehlssprachen dBASE oder Visual Basic für Applikationen

### Deklarative -

Syntax formal

Reihenfolge unbedeutend

Bsp. SQL (Syntax schwieriger)

Bsp. QBE (Syntax benutzerfreundlicher)

---

# Abfragen

---

## Auswahlabfragen

suchen Daten, ohne sie zu ändern

## Aktionsabfragen

schreiben Daten fort

Anfüge-

Lösch-,

Aktualisierungs-

Tabellenerstellungs-

---

# Query by Example - Abfrageschritte

---

1. **Tabellenskelette?**
2. **Ausgabespalten?**
3. **Ausgabezeilen?**
4. **Verbundattribute?**
5. Abfragename?

# QBE - Abfrage auf *einer* Tabelle

“Welcher Dozent unterrichtet Recht ?”

## ORGANISATION

Name	Semester	Dozent	Raumnummer

Tabelle 5.6: Leeres Tabellengerüst

## ORGANISATION

Name	Semester	Dozent	Raumnummer
Recht			

Tabelle 5.7: Tabellengerüst mit QBE-Einträgen

## ANTWORT

Dozent
Meier

Tabelle 5.8: Antworttabelle für die QBE-Anfrage von Tabelle 5.7

## Tabellenskelette

Ausgabespalten

Ausgabezeilen

Verbundattribute

Abfragename

# QBE - Selektionskriterien mit ODER verbinden

“Welche Dozenten lehren im Sommersemester 1993 *oder* 1995 ?”

## ORGANISATION

Name	Semester	Dozent	Raumnummer
	SS93 <i>oder</i> SS95		

## ANTWORT

Dozent
Meier
Müller

Tabellen 5.13: Disjunktion (oder)

Tabellenskelette

Ausgabespalten

Ausgabezeilen

Verbundattribute

Abfragenname

## QBE - Kriterien mit UND/NICHT verbinden

“Welche Dozenten lehren *weder* im Sommersemester 1993 *noch* im Sommersemester 1994 ?”

### ORGANISATION

Name	Semester	Dozent	Raumnummer
	nicht SS93 und nicht SS94		

### ANTWORT

Dozent
Müller
Schmid

Tabellen 5.14: Konjunktion und Negation (und, nicht)

Tabellenskelette

Ausgabespalten

Ausgabezeilen

Verbundattribute

Abfragenname



# QBE - Selektionskriterien

Selektionskriterium	Wichtige Operatoren
Exakte Vergleiche	= < > <= >=
Mustervergleiche	Platzhalter, zum Beispiel ? (?WL deckt <i>BWL</i> und <i>VWL</i> ab)
Logische Operatoren	und oder nicht
Arithmetische Operatoren	+ - * / ( )
Gruppenoperatoren	Summe, Durchschnitt, Häufigkeit, Maximum, Minimum
Übrige Operatoren	kein Eintrag, heutiges Datum

Übersicht 5.12: Weitere Auswahlkriterien für Queries by Example

# QBE über mehreren Tabellen formulieren

“Wieviele **Stunden** unterrichtet Dozent **Meier** im **Sommersemester 1993** ?”

## ORGANISATION

Name	Semester	Dozent	Raumnummer
x	SS93	Meier	

## VERANSTALTUNG

Name	Stunden
x	

## ANTWORT

Stunden
2

### Abfrage 5.9: QBE auf zwei Tabellen

Tabellenskelette

Ausgabespalten

Ausgabezeilen

Verbundattribute

Abfragename

## QBE in MS Access formulieren

The screenshot shows a database management system interface. At the top, a window titled "hochschule : Datenbank" contains a query window titled "QBE58 : Auswahlabfrage". The query window displays a database structure diagram with two tables: "ORGANISATION" and "VERANSTALTUNG". The "ORGANISATION" table has fields: Name, Semester, Dozent, and Raumnummer. The "VERANSTALTUNG" table has fields: Name and Stunden. A line connects the two tables, indicating a relationship. To the right of the diagram, the text "Datenbankstrukturdiagramm" is written in red. Below the query window, a data table is displayed with the following columns: Feld, Tabelle, Sortierung, Anzeigen, Kriterien, and oder. The table contains one row of data:

Feld:	Tabelle:	Sortierung:	Anzeigen:	Kriterien:	oder:
Stunden	VERANSTALTUNG		<input checked="" type="checkbox"/>	"SS93"	"Meier"

### Abfrage 5.10: Die QBE-Abfrage 5.9 in MS Access

---

# QBE - Berechnende Abfragen

---

## Vordefinierte Berechnungen

⇒ enthalten vordefinierte Funktionen wie

- Summe
- Mittelwert

⇒ zeigen das Ergebnis in einem automatisch erzeugten Ergebnisfeld

## Benutzerdefinierte Berechnungen

⇒ kombinieren Attributwerte aus einer oder mehreren Spalten

⇒ zeigen diese in einem benutzerdefinierten Ergebnisfeld

## *Lernziele*

- ⇒ QBE-**Auswahl**abfragen auf verbundenen Tabellen
- ⇒ QBE-**Aktions**abfragen auf verbundenen Tabellen

## *Wiederholungsfragen*

1. Wie unterscheiden sich *Auswahl-* und *Aktions*abfragen?
  - a) Aktionsabfragen erlauben die Aktualisierung von Tabellen.
  - b) Auswahlabfragen ändern Tabellen.
  - c) Aktionsabfragen sind SQL-Abfragen.
2. Die *referentielle* Integrität verlangt ...
  - a) zu jedem Fremdschlüssel- einen passenden Primärschlüsselwert.
  - b) in 1:n-Beziehungen zu jeder Haupttabelle eine Detailtabelle.
  - c) zu jedem Primärschlüssel- einen passenden Fremdschlüsselwert.
3. Ein *Verbundattribut* ...
  - a) ist ein Primärschlüsselattribut, das in zwei Tabellen gleich heisst.
  - b) ist eine Voraussetzung für eine QBE-Abfrage.
  - c) ermöglicht eine Beziehung zwischen zwei Tabellen.

## *Vertiefungsfragen*

Laden Sie die Datenbank [AUFTRÄGESkelett.mdb](#). Sie umfasst die in der ersten Frage von  [Entwurf AUFTRÄGE](#) entworfenen Tabellen.

Falls Sie direkt von der CD ROM oder vom Server laden, können Sie nicht auf die geladenen Dateien schreiben. Kopieren Sie deshalb die Datenbank auf Ihre Festplatte und entfernen Sie im Fall der CD ROM

den Schreibschutz. (Datei auswählen - Rechtsklick - Eigenschaften...).

AUFTRÄGSkelett.mdb enthält die folgenden Tabellen und Attribute:

- ARTIKEL: Artikelnr, Artikelname, Katalogpreis, Lagerbestand
- AUFTRAG: Auftragsnr, Kundennr, Datum
- AUFTRAGSPOSTEN: Auftragsnr, Artikelnr, Artikelmenge, Vereinbarungspreis
- KUNDE: Kundennr, Kundenname, Strasse, PLZ, Ort, Kreditlimite.

### 1. Abfragen auf einer einzigen Tabelle

- a) Suchen Sie alle Kunden, die mit Z beginnen (vgl. Hilfefunkt “Hilfe/Index/Platzhalterzeichen”)

### 2. Verknüpfung von zwei Tabellen

Gehen Sie wie folgt vor (vgl. Hilfe/Index/Beziehungen/Übersicht):

- Menüpunkte “Extras/Beziehungen/Tabelle anzeigen”
  - Verbundattribut der Ersttabelle auf Verbundattribut der Zweitabelle ziehen.
- b) Verknüpfen Sie die Tabellen ARTIKEL, AUFTRAG, AUFTRAGSPOSTEN und KUNDE in einem *Datenbankstrukturdiagramm*. Bilden Sie dabei die folgenden Beziehungen ab:
- Ein Auftragsatz verweist auf mehrere Auftragsposten-Sätze.
  - Ein Artikelsatz verweist auf mehrere Auftragsposten-Sätze.
  - Zu einem Kundensatz können mehrere Auftragsätze gehören.

### 3. Abfragen auf mehreren Tabellen

Gehen Sie wie folgt vor (vgl. Hilfe/Index/Abfragen/Übersicht):

- Datenbankfenster (F11)
- Register “Abfragen”
- Neu

- Entwurfsansicht
  - Tabellen wählen
  - QBE-Schema ausfüllen
  - Ein Rechtsklick und dann “Datenblattsicht” zeigt das Ergebnis
  - Abfragename vergeben.
- c) Welche Aufträge (Auftragsnr) hat Kunde Gebser erteilt?
- d) Was hat Kunde Zanforlin zwischen dem 1. und 20.11.99 bestellt (Artikelname und Menge)?

#### 4. Berechnende Abfragen

Gehen Sie wie folgt vor (vgl. Hilfe/Index/Berechnete Felder/Durchführen von Berechnungen in einer Abfrage):

- e) Wieviel wurde durchschnittlich von jedem Artikelnamen verkauft? (Beispiel einer *vordefinierten* Berechnung)
- f) Ermitteln Sie zu jedem Artikelnamen den niedrigsten und höchsten Vereinbarungspreis. (Beispiel einer *vordefinierten* Berechnung)
- g) Wie hoch ist das gesamte Auftragsvolumen? (Beispiel einer *benutzerdefinierten* Berechnung)

#### 5. Aktualisierungsabfragen

Gehen Sie wie folgt vor (vgl. Hilfe/Index/Abfragen/Aktionsabfragen/Aktualisierungsabfragen):

- Zu aktualisierende Datensätze auswählen (wie in Ziffer 3)
- Menüleiste: Abfrage/Aktualisierungsabfrage
- Aktualisierungsausdruck einsetzen (z.B. [Lagerbestand]+10)
- Menüleiste: Abfrage/Ausführen
- “Rechtsklick: Datenblattansicht” zeigt das Ergebnis
- Abfragename vergeben

h) Erhöhen Sie den Lagerbestand *aller* Artikel um 10 Stück.

- i) Erhöhen Sie den Lagerbestand an *Anzügen* um 11 Stück.

### *Zusatzaufgabe*

- j) Ermitteln Sie alle Kunden, die vom 1.11.99 bis 20.11.99 von einem bestimmten Artikel für mehr als 500.- eingekauft haben.

#### *Hinweise*

- Definieren Sie in der Entwurfsansicht unter anderem die drei Spalten: Datum, Kundenname und den mathematischen Ausdruck für das Postentotal (Artikelmenge \* Vereinbarungspreis).
  - Definieren Sie die Auftragssumme und das Kriterium Zeitraum mit dem Ausdruckseeditor (Symbol "Aufbauen").
- k) Welche Kunden erhalten vom 1.11.99 bis 20.11.99 wieviel Rabatt, wenn der Rabatt 10% vom Auftragstotal ist, falls das Total mehr als 500.- beträgt.

✓ [AUFTRÄGE.mdb](#)



## °Tabellen inklusiv und exklusiv verbinden

SORTE (Verbundwerte grün)

<u>Sorte</u>	<i>Trinktemperatur</i>
Rotwein	12-18
<b>Rosé</b>	<b>8</b>
Weisswein	8

WEIN

<u>Name</u>	<i>Sorte</i>
Dézaley	Weisswein
Dôle	Rotwein
St. Saphorin	Weisswein

*EXKLUSION (fehlender Werte)*

<i>Name</i>	<i>Sorte</i>	<i>Trinktemperatur</i>
Dézaley	Weisswein	12-18
Dôle	Rotwein	8
St. Saphorin	Weisswein	12-18

**INKLUSION** (fehlender Werte)

<i>Name</i>	<i>Sorte</i>	<i>Trinktemperatur</i>
Dézaley	Weisswein	12-18
Dôle	Rotwein	8
St. Saphorin	Weisswein	12-18
<b>?</b>	<b>Rosé</b>	<b>8</b>

Tabellen 5.15: Exklusions- und Inklusionsverknüpfungen SORTE - WEIN

## °In dBASE fortschreiben

STAMM (alt)

<i>Kontonummer</i>	<i>Betrag</i>
10 000	2 500.-
10 001	100 000.-
10 002	-5 000.-
10 011	1 000.-
10 235	500.-

BEWEGUNG

<i>Kontonummer</i>	<i>Betrag</i>
10 001	1 000.-
10 235	-1 000.-

STAMM (neu)

<i>Kontonummer</i>	<i>Saldo</i>	<i>Bemerkungen</i>
10 000	2 500.-	
10 001	101 000.-	nach Addition von 1 000.-
10 002	-5 000.-	
10 011	1 000.-	
10 235	-500.-	nach Addition von - 1 000.-

**Tabelle 5.16:** Update der Datei STAMM

weiss unveränderte Stammsätze

rot Bewegungssätze und veränderte Stammsätze

# Spreadsheets und DB - **Gemeinsames**

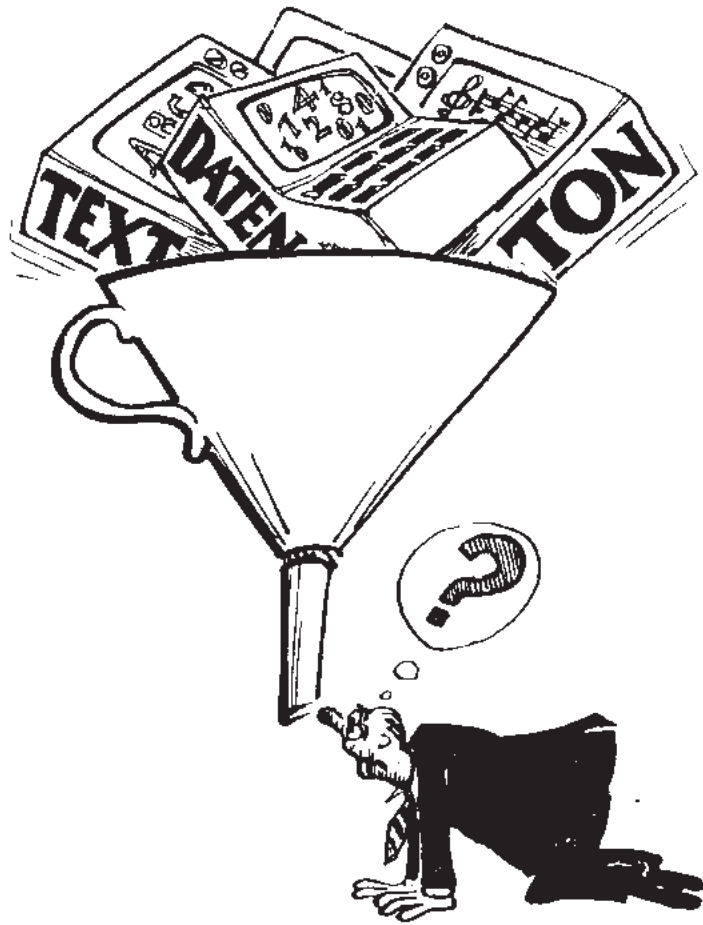
<b>Operation</b>	<b>Argumente</b>
<i>Datendarstellung</i>	Tabellen
<i>Bewegen</i>	zur folgenden/vorangehenden Zelle zur ersten/letzten Zelle zur folgenden/vorangehenden Zeile zur ersten/letzten Zeile
<i>Suchen</i>	nach Auswahlkriterien
<i>Sortieren</i>	nach Sortierkriterien
<i>Ändern</i>	von Zelleninhalten
<i>interaktiv löschen/einfügen</i>	einer Zeile oder Spalte
<i>selektiv löschen</i>	bestimmter Zeilen einer Tabelle
<i>Gültigkeitsprüfung</i>	von Eingabedaten
<i>Passwort</i>	für Anwendungsobjekte, z.B. Tabellen
<i>Makrosprache</i>	zur Applikationsentwicklung
<i>Import/Export</i>	von Fremdformaten (z. B. ASCII)

## Vergleich 5.17: Gemeinsamkeiten von Spreadsheet-Programmen und DBMS

# Spreadsheets und DB - Stärken / Schwächen

<i>Tabellenkalkulation</i>	<i>Datenbankverwaltung</i>
<ul style="list-style-type: none"><li>- Kapazitätsgrenze: i.a. Internspeicher</li><li>- Operationen auf Zellen/Zeilen/Spalten</li><li>- i.a. nur eine oder wenige Tabellen</li><li>- wenig komfortable Abfragemöglichkeiten</li><li>+ auch mehrdimensionale Tabellen</li><li>+ auch Formeln als Zelleninhalte</li><li>+ komplexe numerische Operationen</li><li>- Makrosprachen</li><li>- keine Benutzersichten (engl. views)</li><li>- programmierbare Gültigkeitstests</li></ul>	<ul style="list-style-type: none"><li>+ Kapazitätsgrenze: Externspeicher</li><li>+ auch Operationen auf Tabellen</li><li>+ leichte Verknüpfung vieler Tabellen</li><li>+ komfortable Abfragesprachen</li><li>- nur zweidimensionale Tabellen</li><li>- keine Formeln als Zelleninhalte</li><li>- i.a. einfache numerische Operationen</li><li>+ Programmiersprachen</li><li>+ Benutzersichten</li><li>+ automatische Integritätstests</li></ul>

## Vergleich 5.18: Unterschiede zwischen Spreadsheet-Programmen und DBMS



- Text
- Grafik
- Video
- Audio

# Multimedia in konventionellen Datenbanken

**BLOBs** (Binary Large OBjects) := Bitsequenzen beliebiger Länge, die auch in konventionellen Datenbanksystemen verwaltet werden können (zum Beispiel in Paradox oder MS Access)

## Arten

- Videoclip (Bildfolge)
- Audioclip (Tonfolge)
- Bitmap (Standbild, inkl. Photo CD)
- Texte beliebiger Länge

## Implementation

- Tabelle enthält BLOB-**Felder**
- BLOB-Feld verweist auf BLOB-**Datei** beliebigen Typs (Text, Grafik, Audio, Video)

## Nachteil

- Editieren oft nicht integriert (stattdessen zum Beispiel OLE)

---

# Multimedia-Datenbanken - Eigenschaften

---

## Qualität von **Speichermedien**

- Kapazität
- Transferrate
- Zugriff

## **Suche** nach Deskriptoren

- Text
- Grafik
- Video
- Ton

## **Geräte- und formatunabhängige** Schnittstelle

- unabhängig von der Gerätevielfalt
- unabhängig vom technischen Fortschritt

## Grosse **Datenmengen** und lange Transaktionen

Bsp. Filmwiedergabe

---

# 6 Relationale Systeme

---

4 Datenbanken entwerfen

5 Datenbanken verwalten

⇒ 6 Relationale Datenbanken

- 7 Datenbankanwendungen entwickeln

*Datenmodelle*

Relationenmodell

Netzwerkmodell

*Relationale Grundoperationen*

Selektion

Projektion

Verbund

*Abfragesprachen*

prozedurale - (Grundoperationen)

deklarative - (QBE, SQL)

*Kriterien relationaler Systeme*

Relationen

Nullwerte

Benutzersichten

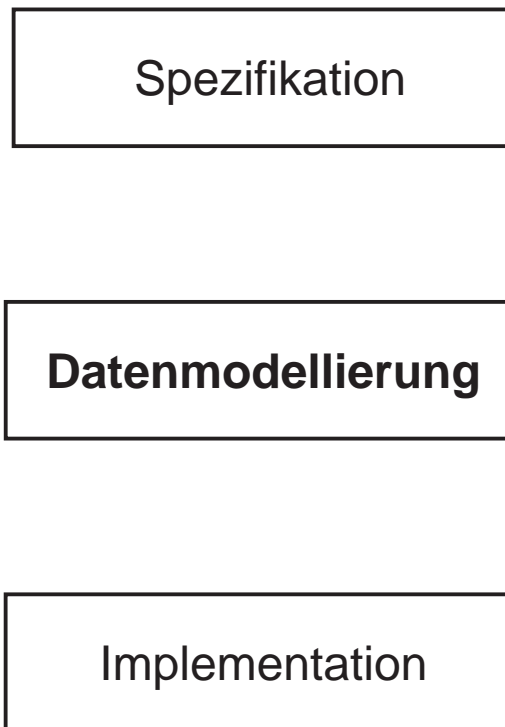
Integritätsbedingungen

Transaktionen

Tabellen als Operanden

*Netzwerkmodell*





**Bild 6.1:** Datenmodellierung als Teil der Anwendungsentwicklung

**Daten modellieren** := ausgewählte Objekte einer **Unternehmung** so abbilden, dass sie mit einem bestimmten **DBMS** verwaltet werden können

**Relationenmodell** und **Netzwerkmodell**

# Relationenmodell - Terminologie

<i>Datenmodelle im allgemeinen</i>	<i>Relationenmodell</i>
Objekttyp	Tabellengerüst
Objektmenge	Tabelle ( <b>Relation</b> )
Objekt	Zeile
Attribut	Spalte
Wert	Zelleninhalt

---

# Die Bedeutung von SQL einschätzen

---

1. Ein Kunde bestellt via **Touch-Tone** Input (Touch-Tone Input ist vor allem in den USA verbreitet und erlaubt die Eingabe über die alphanumerische Tastatur der US-Telefone ).
2. Das verwendete Telefonsystem erlaubt der Firma die **automatische Identifikation** des Kunden. Die Bestellapplikation ruft anhand dieser Identifikation mit einer **SQL**-Anweisung die Kundendaten ab.
3. Die Anwendung prüft mit einer weiteren **SQL**-Abfrage die Verfügbarkeit der bestellten Ware.
4. Die Applikation bestätigt dem Kunden die Bestellung per **Fax**.
5. Die Anwendung führt die Bestelldatenbank per **SQL** nach.

**Beispiel 6.3:** Bedeutung von SQL in betrieblichen Applikationen

# Aufgabe Relation

## STUDIERENDE

Matrikel	Name	Wohnort
34'440	A. Meyer	Basel
...	...	...

**Ordnen Sie die folgenden Begriffe der Beispieltabelle zu:**

- *Tabelle* (Objektmenge)
- *Tabellengerüst* (Relation, Objekttyp)
- *Tabellenzeile* (Objekt, Tupel)
- *Tabellenspalte* (Attribut)
- *Attributname*
- *Tabellenzelleninhalt* (Attributwert)

# Relationenmodell

**Relationenmodell** definiert ...

1. Relationen
2. Grundoperationen auf Relationen
3. Integritätsbedingungen

**Integritätsbedingung** :=

Regel, welche die *Integrität* (Vollständigkeit und Richtigkeit) einer Datenbank mit garantiert

# Wichtige Integritätsbedingungen

## Entitätsintegrität

Jede Tabelle enthält einen Primärschlüssel

PRODUKTE

<u>Produktnummer</u>	...
1240	...
...	...

## Referentielle Integrität

Jeder **Fremd**schlüsselwert einer Haupttabelle verweist auf einen entsprechenden **Primär**schlüsselwert einer Detailtabelle

BESTELLUNGEN (Haupttabelle)

<u>Bestellnummer</u>	<b>Kundennummer</b>	...
150304	1240	...
...	...	...

KUNDEN (Detailtabelle)

<u><b>Kundennummer</b></u>	...
1240	...
...	...

# Abfragen maschinenlesbar machen

Benutzer

Abfragesprachen wie QBE und dBASE

Grundoperationen wie Selektion und Verbund

Physische Tabellen

**Bild 6.2:** Von der Endbenutzerabfrage zu physischen Relationen

# Grundoperationen auf Relationen

Ausgangsrelationen      Ergebnisrelationen<sup>1</sup>

## Beispiel Selektion



**Selektieren** := Zeilen kopieren (Selektion)

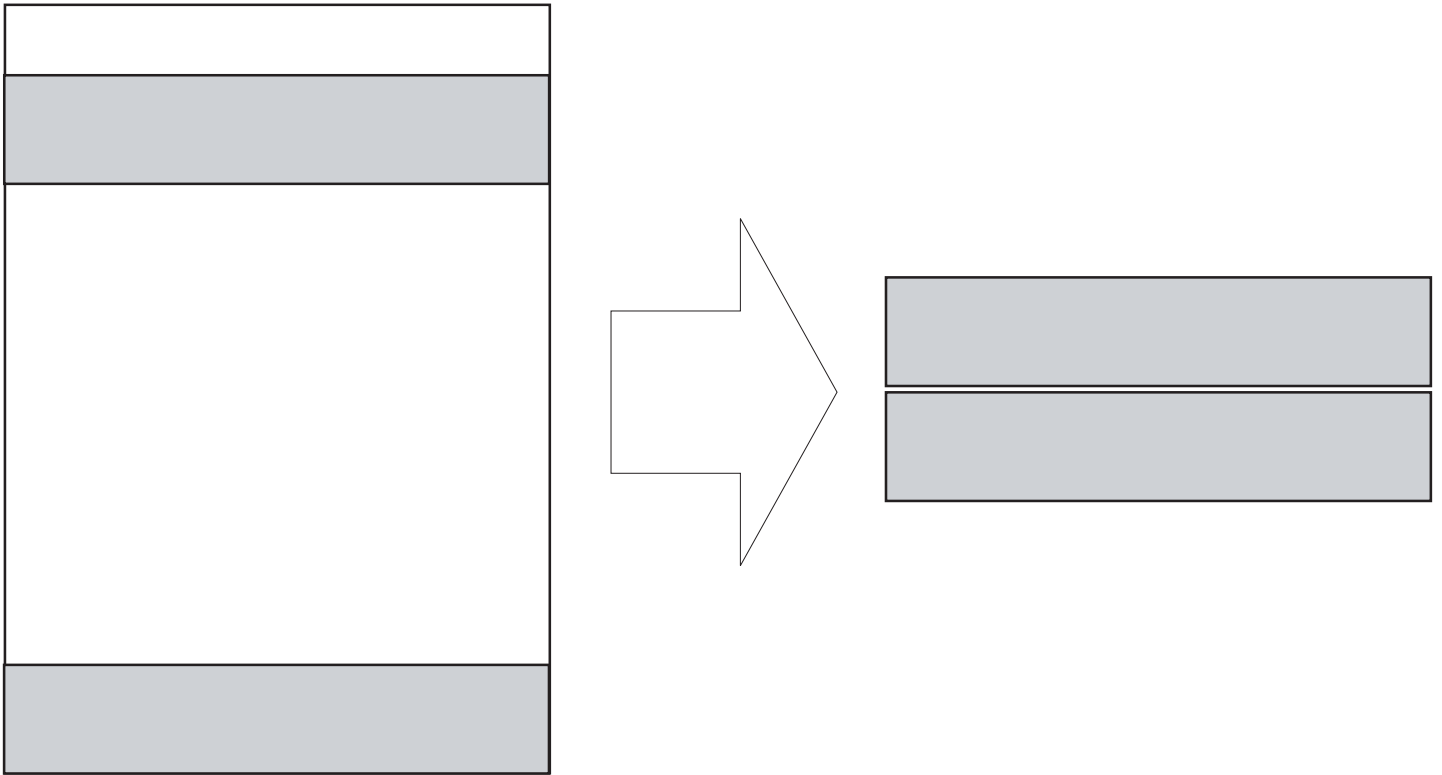
**Projizieren** := Spalten kopieren (Projektion)

**Verbinden** := Tabellen über einem Verbundattribut zusammenfügen

<sup>1</sup> in MS Access auch Domänen genannt



## Relationale Grundoperation *Zeilen selektieren*



**Bild 6.4:** Selektion (Tabellenzeilen kopieren)

# Selektion - Beispiel

## a) Ausgangstabelle

VERBINDUNG

<u>Matrikelnummer</u>	<u>Prüfungsnr</u>	<u>Prüfungsdatum</u>
34	1	30.04.1994
34	2	30.03.1994
65	3	<b>23.10.1994</b>

## b) Selektionsbedingung

**Prüfungsdatum > 30. April 1994**

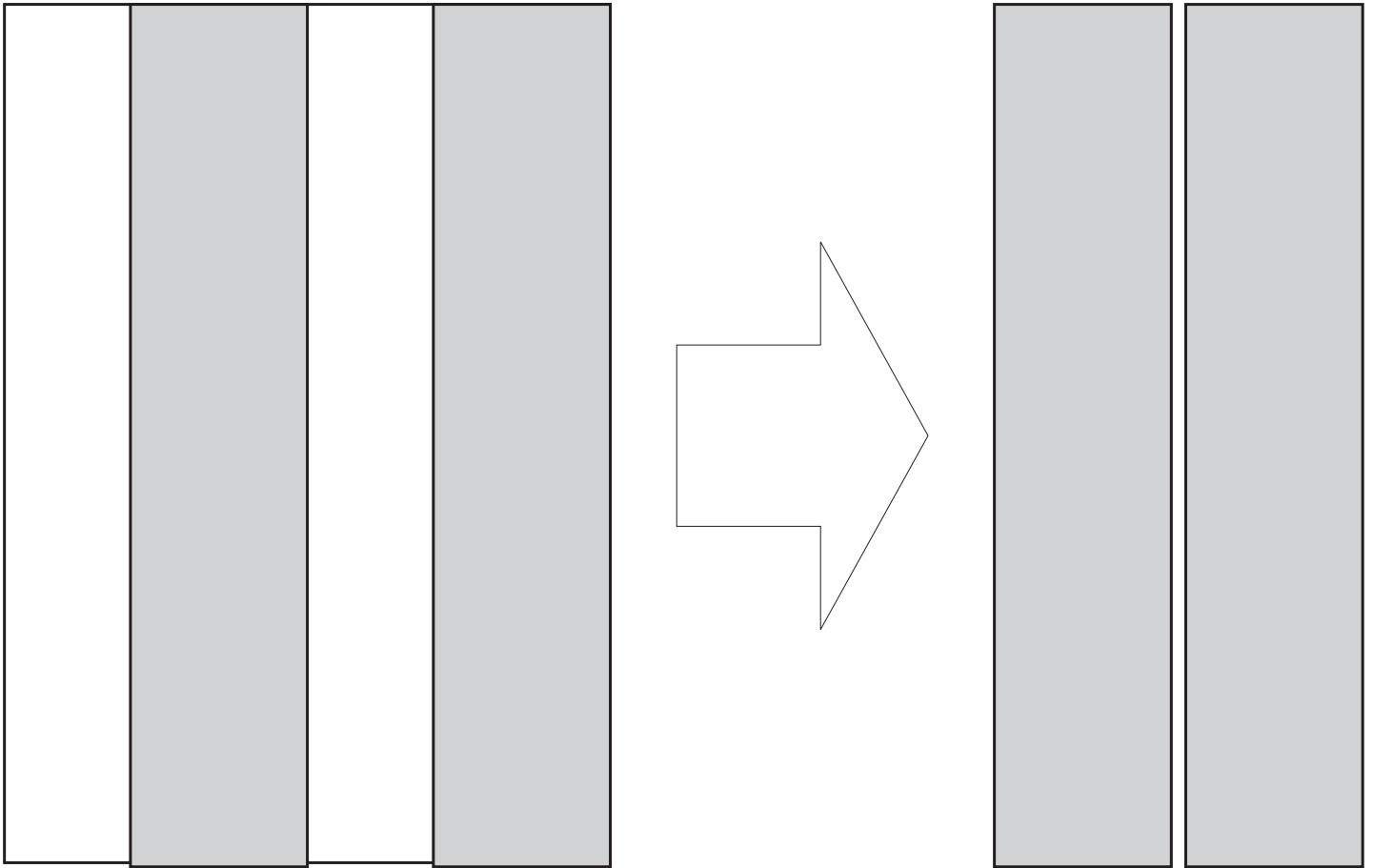
## c) Ergebnistabelle

ERGEBNIS

<u>Matrikelnummer</u>	<u>Prüfungsnr</u>	<u>Prüfungsdatum</u>
65	1	23.10.1994

Beispiel 6.5: Beispiel Selektion

## Relationale Grundoperation *Zeilen projizieren*



**Bild 6.6:** Projektion (Tabellenspalten kopieren)

# Projektion - Beispiel

## a) Ausgangstabelle

STUDENT

<i>Matrikelnummer</i>	Name	<i>Wohnort</i>
34	<b>Müller</b>	St. Gallen
35	Müller	Mörschwil
45	<b>Schmid</b>	Teufen
65	<b>Lang</b>	St. Gallen

## b) Projektionsbedingung

**Name**

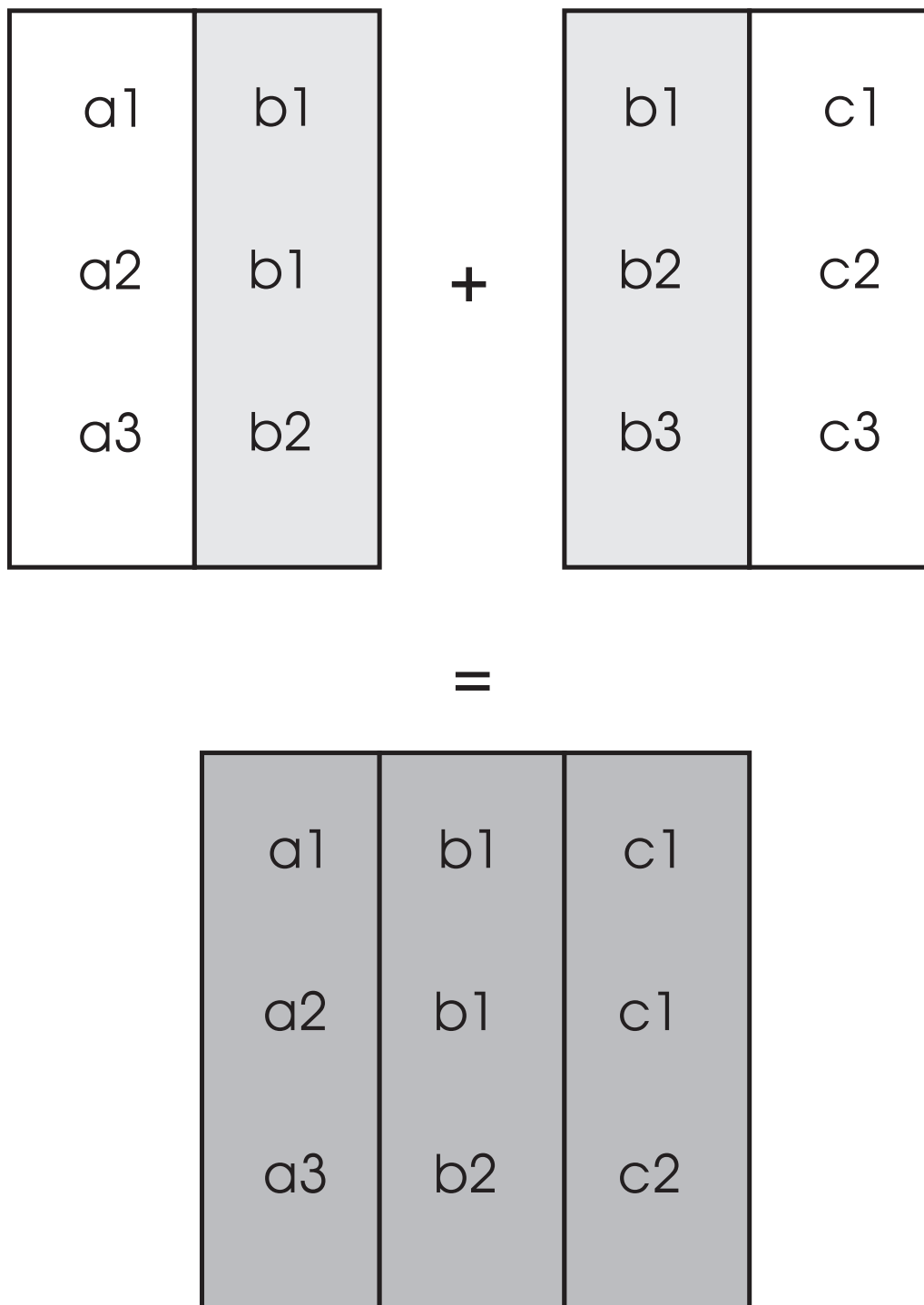
## c) Ergebnistabelle

ERGEBNIS

<i>Name</i>
Müller (Duplikat gestrichen!)
Schmid
Lang

Beispiel 6.7: Projektion

## Relationale Grundoperation *Tabellen verbinden*



**Bild 6.8:** Verbund (Tabellen zeilenweise zusammenfügen)

# Verbund - Beispiel

## a) Ausgangstabellen

### VERBINDUNG

<u>Matrikelnr</u>	<u>Prüfungsnr</u>	<u>Prüfungsdatum</u>
34	1	03.04.1995
34	2	30.03.1995
65	3	23.10.1995

### PRÜFUNG

<u>Prüfungsnr</u>	<u>Prüfungsname</u>
1	BWL
2	VWL
3	Recht

## b) Verbundattribut

**Prüfungsnr**

## c) Ergebnistabelle

### ERGEBNIS

<u>Matrikelnr</u>	<u>Prüfungsnr</u>	<u>Prüfungsname</u>	<u>Prüfungsdatum</u>
34	1	BWL	03.04.1995
34	2	VWL	30.03.1995
65	3	Recht	23.10.1995

Beispiel 6.9: Verbund (Verbundattribut grün)

# Prozedurale Abfragen mit Grundoperationen

## 1. *Natürlichsprachliche* Abfrage

“ **Matrikel und Namen** der Studierenden, die **nach dem 30.3.94** eine Prüfung bestanden haben?”

## 2. *Prozedurale* Abfrage

*Relationen*

STUDIERENDE(Matrikel, Name, Wohnort)

VERBINDUNG(Matrikel, Prüfungsnummer, Prüfungsdatum)

*Grundoperationen*

1. **Selektion** in VERBINDUNG mit **Prüfungsdatum > 30.3.94**
2. **Verbund** von STUDIERENDE und der Ergebnistabelle des ersten Schritts mit dem Verbundattribut **Matrikel**
3. **Projektion** der Ergebnistabelle von 2. auf **Matrikel und Name**



Benutzerunfreundlich!



Deklarative Abfragen mit QBE und SQL !

---

# Abfragen benutzerfreundlicher gestalten

---

## 1. Diagnose

- benutzerfeindliche Reihenfolge der Operationen
- benutzerfeindliche Verwaltung von Zwischenergebnissen

## 2. Therapie

- Weniger Tabellen durch Lockerung der Normalisierung ?
- Benutzerfreundliche deklarative Abfragesprachen (QBE, SQL) !

Deklarative Abfragen sind benutzerfreundlicher

### Beispiel SQL

```
select distinct STUDENT.Matrikel, Name  
from STUDENT, VERBINDUNG  
where STUDENT.Matrikel = VERBINDUNG.Matrikel  
and Prüfungsdatum > 30. März 1994
```



# SQL - Grobübersicht

**SQL** := Structured Query Language

## Operationen

### Datenbankobjekte definieren

- create

### Datenbankobjekte manipulieren

- insert
- update
- delete

### Datenbanken abfragen

- select

### Datenbanken schützen

- grant

## SQL ist ...

- deklarativ (WAS statt WAS UND WIE)
- interaktiv oder eingebettet

# Auswahlabfragen in SQL

**select** [distinct] **Attribut**,  
...  
**from** **TABELLE**,  
...  
**where** **Verbund-** und **Selektions**bedingungen  
**order by** *Sortierkriterium*

**select distinct** **STUDENT.Matrikel, Name**  
**from** **STUDENT, VERBINDUNG**  
**where** **STUDENT.Matrikel =**  
**VERBINDUNG.Matrikel**  
**and Prüfungsdatum > 30. März 1994**  
**order by** **Name**

“distinct” garantiert eine Ergebnistabelle ohne Dulikate

# Aufgabe zu SQL

## Wie lautet die Ergebnistabelle?

```
select  Dozent, Name, Semester
from    VORLESUNG, ORGANISATION
where   VORLESUNG.Name = ORGANISATION.Name and Stunden > 2
order by Dozent
```

### VORLESUNG

<u>Name</u>	Stunden
Recht	2
BWL	3
VWL	3

### ORGANISATION

<u>Name</u>	<u>Semester</u>	<u>Dozent</u>	<u>Raumnummer</u>
Recht	SS87	Meier	111
Recht	SS88	Meier	112
BWL	WS88	Schmid	111
BWL	SS89	Schmid	111
VWL	SS88	Müller	112
VWL	WS89	Müller	112

---

# Lösung zur Aufgabe

---

<i>Dozent</i>	<i>Name</i>	<i>Semester</i>
Müller	VWL	SS88
Müller	VWL	WS89
Schmid	BWL	SS89
Schmid	BWL	WS88

## *Lernziele*

- ⇒ Grundbegriffe des allgemeinen Datenmodells
- ⇒ Grundbegriffe des Relationenmodells
- ⇒ Relationale Grundoperationen
- ⇒ Abfrageergebnisse als Operanden

## *Wiederholungsfragen*

1. Eine Relation ...

- a) verlangt einen Primär- oder Sekundärschlüssel
- b) erlaubt keine Duplikate
- c) erlaubt keine Nullwerte


2. Welche Forderung stellt die Entitätsintegrität?

- a) Jede Tabellenzeile ist eindeutig.
- b) Zu jedem Satz einer Haupttabelle gehört ein Satz einer Detailtabelle.
- c) Jede Entität lässt sich als Tabelle darstellen.

3. Jede QBE-Abfrage ...

- a) setzt sich aus den Grundoperationen Selektion, Projektion und Verbund zusammen.
- b) ist prozedural.
- c) lässt sich auf eine Folge relationaler Grundoperationen zurückführen.

## Vertiefungsfragen

Laden Sie die Datenbank  [HOCHSCHULESkelett.mdb](#). Falls Sie direkt von der CD ROM oder vom Server laden, können Sie nicht auf die geladenen Dateien schreiben. Kopieren Sie deshalb die Datenbank auf Ihre Festplatte und entfernen Sie im Fall der CD ROM den Schreibschutz.

HOCHSCHULESkelett.mdb enthält die folgenden Tabellen und Attribute:

- STUDENTin: Matrikelnr., Name, Wohnort
- PRÜFUNG: Prüfungsnr., Prüfungsname
- VERBINDUNG: Matrikelnr., Prüfungsnr., Prüfungsdatum.

### 1. Relationale Grundbegriffe

- a) Klicken Sie im Datenbankfenster (F11) auf STUDENTin und entnehmen Sie dieser Tabelle Beispiele für die folgenden *Begriffe* von Datenmodellen, insbesondere des Relationenmodells:

Wert .....

Attribut .....

Objekt .....

Objektmenge .....

Objektyp .....

Relation .....

### 2. Integritätsbedingungen

- b) Implementieren Sie für jede Tabelle die *Entitätsintegrität*.
- c) Verknüpfen Sie die Tabellen STUDENTin, PRÜFUNG, und VERBINDUNG in einem *Datenbankstrukturdiagramm* und implementieren Sie dabei die *referentielle* Integrität.

- d) Finden Sie mit der Hilfefunktion heraus, was die Optionen “Aktualisierungsweitergabe” und “Löschweitergabe” der referentiellen Integrität bedeuten.

### 3. Relationale Grundoperationen

- e) Führen Sie die folgende Abfrage durch: Welche Studierenden sind nicht in St.Gallen wohnhaft? Um welche Grundoperation handelt es sich?
- f) Führen Sie die folgende Abfrage durch: Welche Matrikelnummern und Namen von Studierenden enthält die Tabelle STUDENTin? Um welche Grundoperation handelt es sich?
- g) Erstellen Sie eine Tabelle aller mündlichen Prüfungen. Die Tabelle soll folgende Attribute enthalten: Prüfungsnummer, -name und datum sowie Matrikelnummer des Prüflings. Um welche Grundoperation handelt es sich?
- h) Stellen Sie die folgende Abfrage: Welches sind die Matrikelnummern und Namen der Studierenden, die vor dem 23.10.98 eine Prüfung bestanden haben? Geben Sie dieser Abfrage den Namen “Deklarativ”.
- i) Wenn Sie nichts vorsehen, ist die Ergebnistabelle von Abfrage h keine Relation, weil sie Duplikate enthält. Ändern Sie dazu die Eigenschaft “Keine Duplikate” in der Entwurfssicht der Abfrage “Deklarativ”.
- j) Nennen Sie die drei Grundoperationen, aus denen sich die Abfrage h zusammensetzt:
1. ....
  2. ....
  3. ....
- k) Überprüfen Sie Ihre Antwort auf die letzte Frage, indem Sie der Reihe nach die drei Grundoperationen als QBE-Abfragen imple-

mentieren. Das Ergebnis der dritten Grundoperation muss schliesslich dem Ergebnis der Abfrage h entsprechen. Speichern Sie die Grundoperationen als Abfragen unter den Namen “DeklarativSelektion1”, “Deklarativ...2” und “Deklarativ...3” ab. Sie können so das Ergebnis der Abfrage für den Entwurf der nächsten Abfrage wieder verwenden.

## *Zusatzaufgaben*

- l) Stellen Sie die Abfrage “Deklarativ2”: Welches sind die Namen, die Semester und die DozentInnen der Vorlesungen, deren Stundendotation grösser als 2 ist. Sortieren Sie das Ergebnis nach dem Attribut DozentIn.
- m) Nennen Sie die drei Grundoperationen, aus denen sich die Abfrage i zusammensetzt:
  - 1. ....
  - 2. ....
  - 3. ....
- n) Überprüfen Sie Ihre Antwort auf die letzte Frage, indem Sie der Reihe nach die Grundoperationen als QBE-Abfragen implementieren. Das Ergebnis der letzten Grundoperation muss dem Ergebnis der Abfrage l entsprechen. Speichern Sie die Grundoperationen als Abfragen unter den Namen “Deklarativ2Selektion1”, “Deklarativ2...2” und “Deklarativ2...3” ab. Sie können so die Abfrageergebnisse wieder verwenden.

✓ [HOCHSCHULE.mdb](https://hochschule.mdb.ch)



# SQL - Aggregatfunktionen

**Aggregatfunktion** := Funktion, die eine Basis- oder Ergebnistabelle statistisch zusammenfasst

**sum** (Summe)

**avg** (Mittelwert, engl. average)

**count** (Anzahl)

**min** (Minimum)

**max** (Maximum)

**stdev** (Standardabweichung, engl. standard deviation)

## *Beispiel*

```
select count( Mitarbeiternr )  
as Mitarbeiterzahl from MITARBEITER
```

**as** definiert einen **Aliasnamen** (hier Mitarbeiternr), der über der berechneten Spalte der Ergebnistabelle steht

---

## SQL - group by in einer **Einzel**tabellenabfrage

---

*a) Umgangsprachlich*

Berechne die Durchschnittspreise  
aller ARTIKEL

gruppiert nach Lieferantennummern

*b) Auf einer Einzeltabelle*

**select**      **avg** ( Preis ) **as** Durchschnittspreis

**from**        ARTIKEL

**group by**    Lieferantennr

**group by** **Attribut** gruppiert Zeilen mit gleichem Attributwert

---

## SQL - group by in einer **Mehr**tabellenabfrage

---

*Auf mehreren Tabellen*

```
select      avg ( AUFLAGSPPOSTEN.Artikelmenge )  
              as Mengendurchschnitt,  
              ARTIKEL.Artikelname  
  
from        ARTIKEL, AUFLAGSPPOSTEN  
  
where       ARTIKEL.ArtikelNr =  
              AUFLAGSPPOSTEN.ArtikelNr  
  
group by    ARTIKEL.Artikelname
```

---

# SQL - Erweiterte select-Syntax

---

select [distinct] **Attribut** [As Aliasname], ...

from                    **TABELLE**,  
                          ...

where                    **Verbund-** und **Selektions**bedingungen

group by

order by                *Sortierkriterium*

---

## SQL - Vorteile

---

- ANSI / ISO **Standard**      Portabilität
- für **verteilte** Datenbanken geeignet
- aus **prozeduralen** Sprachen verwendbar
- anders als QBE **terminalunabhängig**

---

# Interaktives und eingebettetes SQL

---

## Interaktiver Gebrauch von SQL

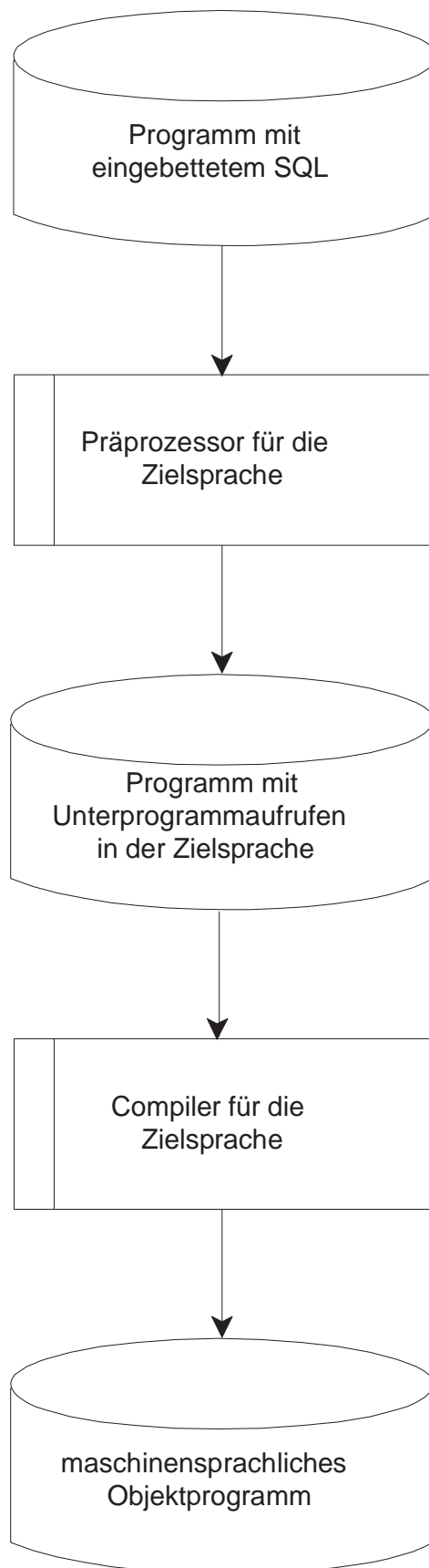
```
select Stunden  
from Vorlesung  
where Name = 'BWL'
```

## Eingebetteter Gebrauch von SQL

```
exec sql  
select Stunden into :x  
from Vorlesung  
where Name = 'BWL';
```

3 wird in der Variablen X gespeichert

# SQL eingebettet



**Bild 6.12:** Vom eingebetteten **Quellcode** zum ablauffähigen **Maschinencode**

---

# SQL - Verbreitete Anweisungen (A 6.5)

---

## Interpretieren Sie die folgenden SQL-Anweisungen !

```
create schema authorisation Müller
```

```
create table VERANSTALTUNG (  
    Name char(30) not null,  
    Stunden decimal(3),  
    unique (Name)  
)
```

```
create table ORGANISATION (...)
```

```
create table RAUM (...)
```

```
select Stunden  
from VERANSTALTUNG  
where Name = 'BWL'
```

```
insert  
into VERANSTALTUNG (Name, Stunden)  
values ('Informatik', 2)
```

```
update VERANSTALTUNG  
set Stunden = 2 * Stunden  
where Name = 'Informatik'
```

```
delete  
from VERANSTALTUNG  
where Stunden > 4
```

```
create view SCHWEIZERKUNDE (Nr, Name) as  
    select Nr, Name  
    from KUNDE  
    where Land = 'Schweiz'
```



---

# SQL - Views (Sichten)

---

## Beispiel

```
create view SCHWEIZERKUNDE (Nr, Name)
as select Nr, Name
from KUNDE
where Land = 'Schweiz'
```

## Merkmale

- Ergebnistabelle (View) **virtuell**
- Ergebnistabelle **aus** Basistabellen oder andern Views
- **Fortschreibung** einer View beschränkt möglich

# SQL - Teilsprachen

SQL-Kategorie	Zweck	Anweisungen
Data <b>Definition</b> Language (DDL)	Definition von Datenbankobjekten	create, alter, drop table, view, index
Data <b>Manipulation</b> Language (DML)	Abfrage Änderung	select insert, delete, update
Data <b>Control</b> Language (DCL)	Transaktionen Schutz / Sicherheit	lock, commit, rollback grant, revoke

Übersicht 6.10: Kategorien von SQL-Anweisungen

# SQL - Rekursive Abfragen

## MITARBEITER

Mitarbeiter	Vorgesetzter
1	null
2	1
3	1
4	2
5	2
...	...

**Tabelle 6.11: Eine rekursive 1:n-Beziehung**

```
select Vorgesetzter    % nächster Vorgesetzter
from MITARBEITER
where Mitarbeiter = 5                                % Antwort = 2

select Vorgesetzter    % übernächster Vorgesetzter
from MITARBEITER
where Mitarbeiter = (
    select Vorgesetzter                                % Unterabfrage
    from MITARBEITER
    where Mitarbeiter = 5
)                                                        % Antwort = 1

select Vorgesetzter    % alle Vorgesetzte
from MITARBEITER
connect by Mitarbeiter = prior Vorgesetzter
start with Mitarbeiter = 5                            % Antwort = 2, 1
```

## *Lernziele*

- ⇒ einfaches select
- ⇒ Unterabfrage (geschachtelte Abfrage)
- ⇒ insert
- ⇒ update
- ⇒ delete
- ⇒ Funktionen (sum, min, max, avg)

## *Wiederholungsfragen*


1. Eine select-Abfrage ist eine ...
  - a) Aktionsabfrage
  - b) Aktualisierungsabfrage
  - c) Auswahlabfrage
2. Welche Aussage ist richtig?
  - a) Jede QBE-Abfrage ist unter MS Access in SQL darstellbar.
  - b) Jede SQL-Abfrage ist unter MS Access in QBE darstellbar.
  - c) SQL und QBE sind in MS Access äquivalent.
3. Ordnen Sie die SQL-Schlüsselwörter A bis E den Argumenten 1 bis 5 zu:  
A select  
B select distinct  
C from  
D where  
E order by

- 1 wählt eine oder mehrere Tabellen
- 2 Sortierkriterium
- 3 ergibt eine Ergebnistabelle ohne Duplikate
- 4 eines oder mehrere Attribute
- 5 Verbund- oder Selektionsbedingung

## Vertiefungsfragen

Formulieren Sie alle SQL-Abfragen zunächst auf Papier und implementieren Sie sie erst dann in MS Access, und zwar in den folgenden Schritten:

- **Datenbankfenster** (F11)/Abfragen/Neu/Entwurfsansicht
- **Schliessen** (ohne eine Tabelle hinzuzufügen)
- Symbolleiste: Ansicht/**SQL**
- **Anweisung** eingeben
- Symbolleiste: Ansicht/**Datenblattansicht**
- Symbolleiste: Ansicht/**SQL**, falls Korrektur

Laden Sie die Datenbank  [KALIBASkelett.mdb](#). Falls Sie direkt von der CD ROM oder vom Server laden, können Sie nicht auf die geladenen Dateien schreiben. Kopieren Sie deshalb die Datenbank auf Ihre Festplatte und entfernen Sie im Fall der CD ROM den Schreibschutz.

Die Datenbank enthält die folgenden Tabellen und Attribute:

- ARTIKEL: Artikelnr, Name, Katalogpreis, Lagerbestand
- AUFTRAG: Auftragsnr, Kundennr, Datum
- AUFTRAGSPOSTEN: Auftragsnr, Artikelnr, Artikelmenge, Vereinbarungspreis
- KUNDE: Kundennr, Kundenname, Strasse, PLZ, Ort, Kreditlimite
- MITARBEITER: Personalnr, Name, Vorgesetztennr.

MS Access transformiert reservierte Wörter von SQL in Grossschrift (auch wenn Sie die Wörter im Entwurfseditor klein schreiben).

## 1. Abfragen auf einer einzigen Tabelle

- a) Wo wohnt der Kunde Chaplin?
- b) Wie heissen die Kunden, die in Basel oder Zürich wohnen? Vergleichen Sie diese SQL-Abfrage mit der QBE-Version.

.....

.....

.....

## 2. Abfragen auf mehreren Tabellen

- c) Welche Aufträge (Auftragsnr) hat Kunde Gebser erteilt? Vergleichen Sie diese SQL-Abfrage mit der QBE-Version.

.....

.....

- d) Was hat Kunde Zanforlin zwischen dem 1. und 20.11.99 bestellt (Artikelname und Menge)? Verwenden Sie für das Datum das folgende Format: #01/11/99# (statt 1.11.99).

## 3. Berechnende Abfragen

- e) Wieviel wurde durchschnittlich von jedem Artikelnamen verkauft?
- f) Ermitteln Sie zu jedem Artikelnamen den niedrigsten und höchsten Vereinbarungspreis. Vergleichen Sie diese SQL-Abfrage mit der QBE-Version.

.....

.....

.....

#### 4. Aktualisierungsabfragen

- g) Erhöhen Sie den Lagerbestand *aller* Artikel um 10 Stück.
- h) Erhöhen Sie den Lagerbestand an *Anzügen* um 11 Stück.

#### 5. Unterabfragen (engl. subqueries)

- i) Welches ist der nächste Vorgesetzte von Feller?
- j) Welches ist der *übemächste* Vorgesetzte von Feller? Wenn Sie Sie in der Entwurfssicht den Menüpunkt "Ansicht/Entwurfsansicht" klicken, dann sehen Sie die Übersetzung der SQL-Abfrage in die QBE-Version.
- k) Welche Kunden (Kundennr, Kundenname) wohnen am gleichen Ort wie der Kunde Zanforlin?

#### 6. Vereinigung von Tabellen (engl. union)

- l) Interpretieren Sie die folgende Abfrage:

```
select Kundenname  
from KUNDE  
union  
select MitarbeiterIn  
from MITARBEITERIN
```

Prüfen Sie Ihre Interpretation in MS Access.

- m) Prüfen Sie, ob die Abfrage l auch in QBE implementierbar ist. Welchen Schluss über das Verhältnis von QBE und SQL ziehen Sie?

#### Zusatzaufgabe

- k) Finden Sie durch Experimentieren heraus, welche Unterschiede zwischen den von Hand erstellten SQL-Abfragen und den automatischen SQL-Übersetzungen äquivalenter QBE-Abfragen bestehen.

✓ [KALIBA.mdb](#)

# RDBMS - Wichtige Codd'sche Anforderungen

Daten in Tabellen **darstellbar**

Atomarer Wert durch [Tabelle, Schlüssel, Attribut] **auffindbar**

**Nullwert** bedeutet fehlende Information

Mindestens eine befehlsgesteuerte **Abfragesprache**

## **Views**

**Interaktive** und **programmgesteuerte** Datenmanipulation

Automatische **Integritäts**kontrolle  
(zumindest die Kontrolle der Entitäts- und Beziehungsintegrität)

Automatische **Transaktions**kontrolle

**Tabellenweise** und nicht nur satzweise Verarbeitung



# Nullwerte verwalten

## ORGANISATION

<i>Name</i>	<i>Semester</i>	<i>Raum</i>	<i>Plätze</i>	<i>Dozent</i>
Recht	SS93	111	200	Meier
Recht	SS94	112	150	Meier
BWL	WS94	null	200	Schmid
BWL	SS95	111	200	Schmid
VWL	SS94	112	150	Müller
VWL	WS95	112	150	Müller

## QBE - Abfrage

<i>Name</i>	<i>Semester</i>	<i>Raum</i>	<i>Plätze</i>	<i>Dozent</i>
		not 111		

### Abfrage 6.14: Behandlung von Nullwerten

**Welche Ergebnistabellen sind möglich ?**

# Metadaten beschreiben Objektdaten

<i>Attributname</i>	<i>Attributtyp</i>
Name	alphabetisch (10)
Semester	alphabetisch (4)
Dozent	alphabetisch (10)
Raum	numerisch

**Tabelle 6.13:** Ausgewählte Metadaten der Tabelle ORGANISATION

---

# Transaktionen definieren

---

**Lies** die Daten der laufenden Bestellung vom Benutzer

**Lies** den entsprechenden Kd.satz der Tabelle KUNDE

**Wenn**  $(\text{Kd.satz.Saldo} + \text{Rechnungsbetrag}) \leq \text{Kreditlimite}$

$\text{Kd.satz.Saldo} := \text{Kd.satz.Saldo} + \text{Rechnungsbetrag}$

**Schreibe** den Kd.satz in die Tabelle KUNDE

**Schreibe** den Bestellsatz in die Tabelle BESTELLUNG

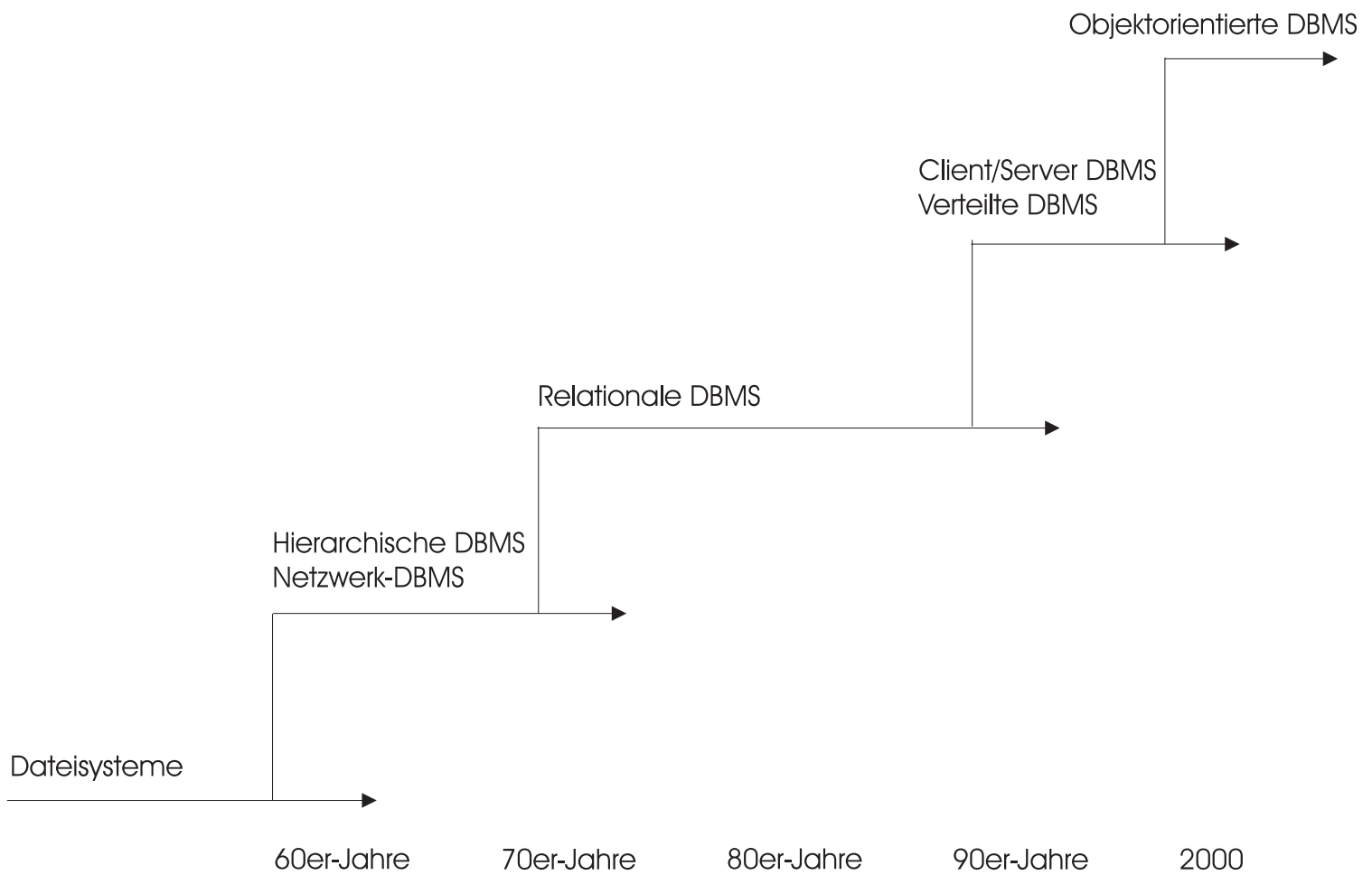
## Pseudocode 6.15: Beispiel einer Transaktion

# SQL verbessert die Integrität der Daten

<i>Integritätsregel</i>	<i>SQL</i>
Entitätsintegrität	primary key
Eindeutigkeit von Nichtschlüsseln	distinct
Beziehungsintegrität	foreign key
Verbot von Nullwerten	create/alter/drop domain, not null
Voreingestellte Werte	create/alter/drop domain
Typenprüfung	create/alter/drop domain
Wertreihenfolge	create/alter/drop domain

Übersicht 6.16: Beispiele von Integritätsregeln

# Geschichte der DBMS



**Bild 6.17:** Geschichte der DBMS-Entwicklung

## Datenbeschreibung und -manipulation **physischer**

- speicher- und laufzeiteffizienter
- Satzlängen variabel
- Beziehungen entstehen über verborgene Adressverweise

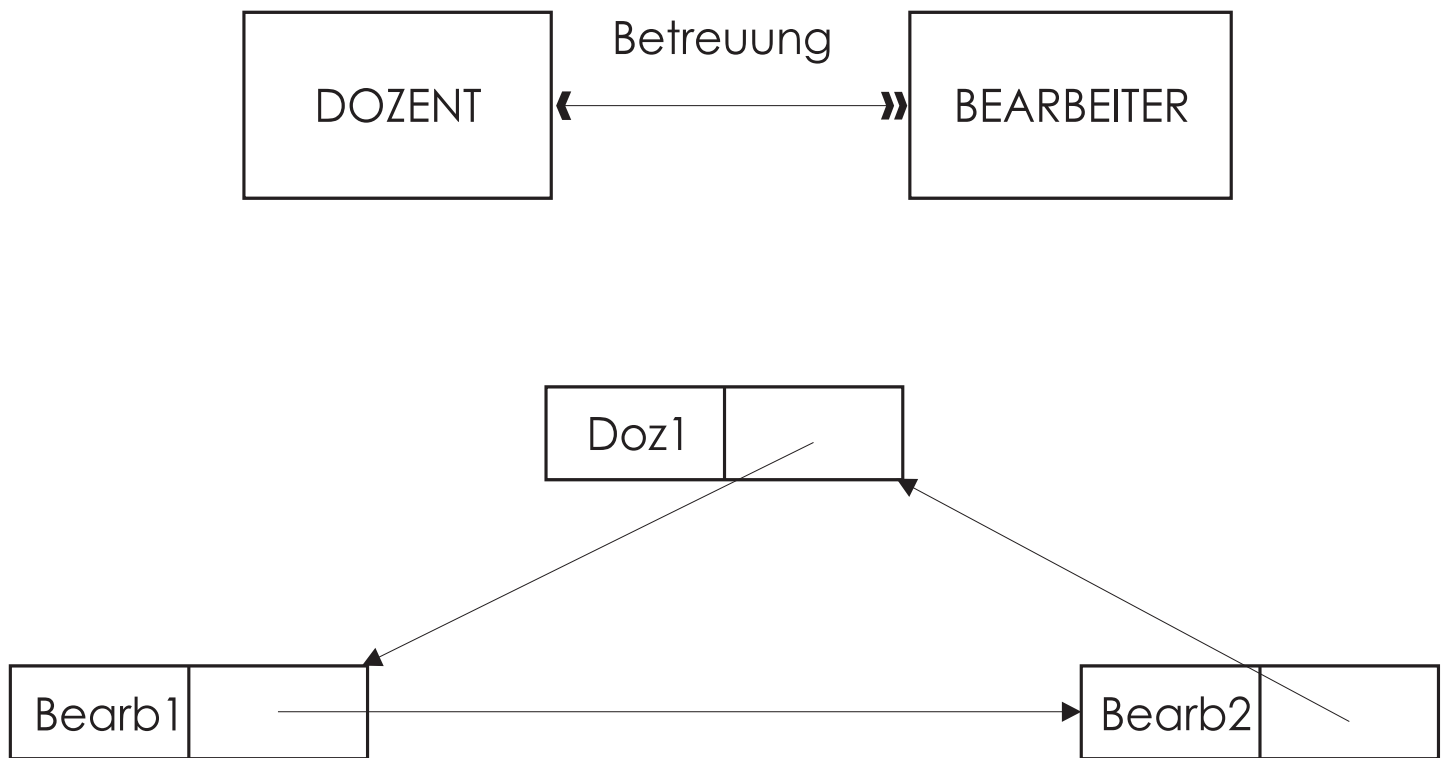
## Weniger **benutzerfreundlich**

- Datenstrukturen weniger plausibel
- Operatoren satzweise statt tabellenweise
- Abfragesprachen meist prozedural

## Weniger **flexibel**

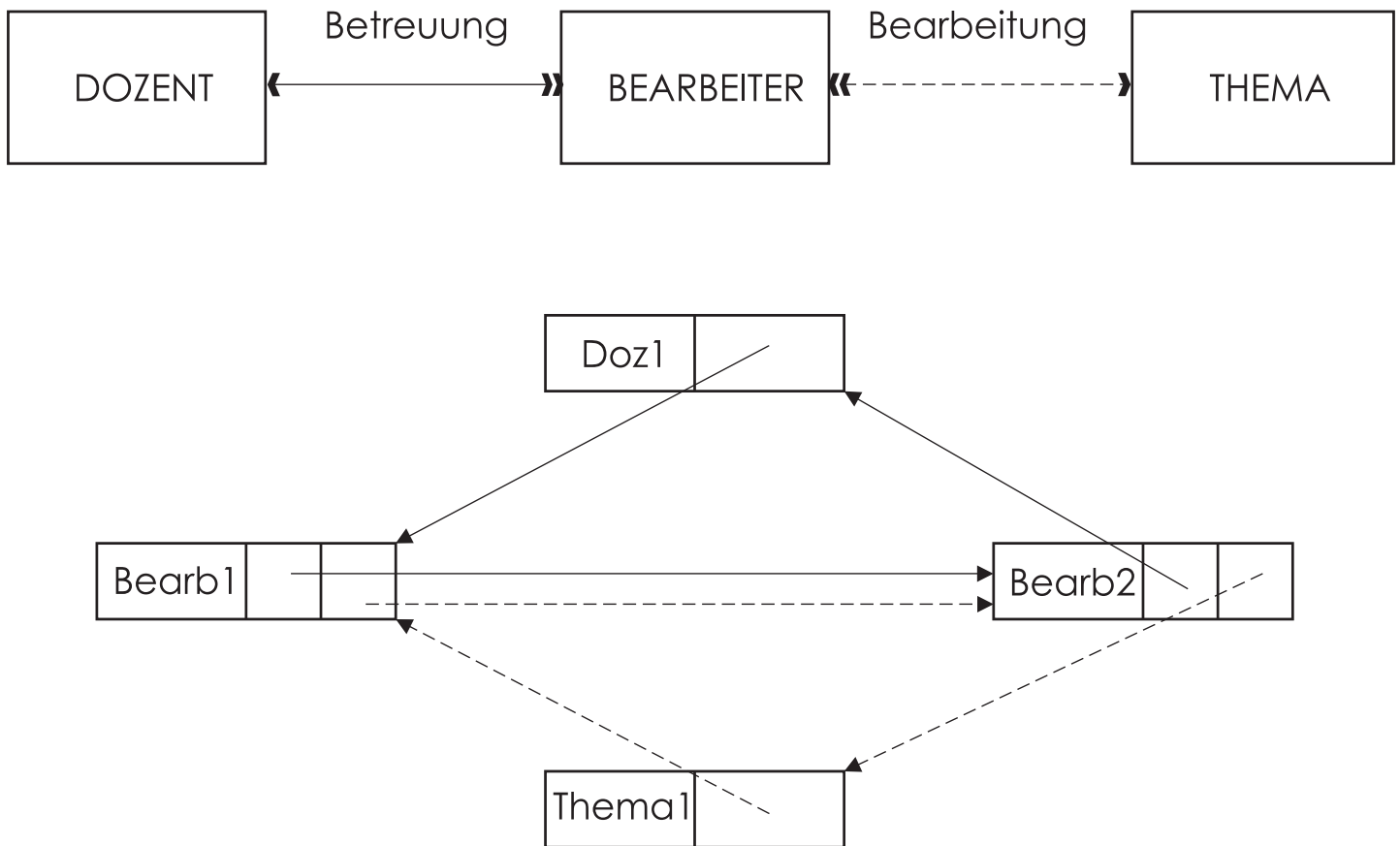
- Datenbankstruktur weniger leicht änderbar

# 1:n im Netzwerkmodell implementieren



**Bild 6.18:** Implementation einer 1:n-Beziehung

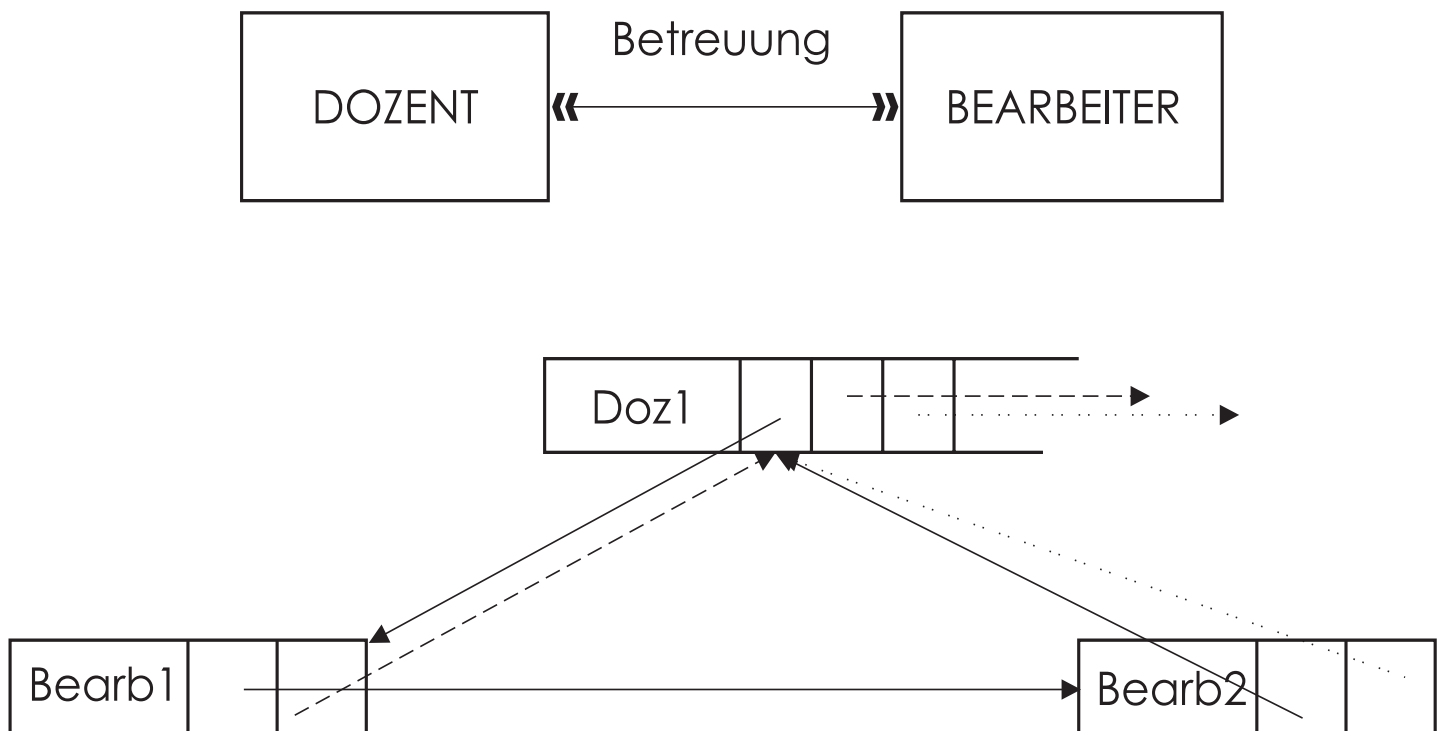
## 2 x 1:n im Netzwerkmodell implementieren



**Bild 6.19:** Implementation zweier 1:n-Beziehungen



# m:n im Netzwerkmodell implementieren



**Bild 6.20:** Implementation einer m:n-Beziehung

---

# Eine navigierende Abfrage im Netzwerkmodell

---

Suche das Diplomarbeitsthema X in der Datei THEMA  
Folge Kette *Bearbeitung* bis zum ersten Diplomanden  
**Solange** die Kette *Bearbeitung* nicht zu Ende ist  
Folge der Kette *Betreuung* bis zum ersten Dozenten  
Zeige den Dozentennamen  
Lies nächsten Bearbeitersatz der Kette *Bearbeitung*

**Entwurfscod** 6.21: Eine navigierende Abfrage zu Bild 6.19

---

# 7 Anwendungen entwickeln

---

4 Datenbanken entwerfen

5 Datenbanken verwalten

6 Relationale Datenbanken

⇒ 7 Datenbankanwendungen entwickeln

## *Benutzerschnittstellen*

Befehlschnittstelle

Menüschnittstelle

Direkte Manipulation

## *Programmiersprachen*

eingebaute - (i.d.R. Viertgenerationssprachen)

externe -

## *Programmentwicklung*

Spezifikation

Entwurf

Implementation

Installation und Wartung

---

# Phasen der Applikationsentwicklung

---

## 1. Spezifikation

## 2. Entwurf

- Datenentwurf
- Dialogentwurf
- Algorithmischer Entwurf

## 3. Implementation

- Codieren
- Testen

## 4. Installation und Wartung

---

# Ein Literaturverwaltungssystem spezifizieren

---

## Daten

### Dokumente mit den bibliographischen Angaben

- Nummer
- Autor
- Titel
- Verlag
- Erscheinungsjahr

### Schlagworte für jedes Dokument

## Operationen

### Suchen nach . . .

- bibliographischen Angaben
- Schlagworten

### Ändern von Dokumenten

- Anfügen
- Editieren
- Löschen

# Als Formular ausgeben

**Margaret Peacock**

**Personal-Nr:** 4

**Vorname:**


**Nachname:**

**Position:**

**Vorgesetzte(r):**

**Einstellungsdatum:**

**Durchwahl Büro:**



**Persönliche Daten**

Datensatz:       von 9

**Bild 7.1:** Beispiel eines Formulars

# Als Bericht ausgeben

## *Umsatzsummen nach Anzahl*

04. Jun. 96

Umsatz:	Bestell-Nr:	Firma:	Zähler:
SFr. 11'188	10417	Simons bistro	1
SFr. 10'165	10691	QUICK-Stop	2
SFr. 10'192	10540	QUICK-Stop	3
SFr. 10'496	10479	Rattlesnake Canyon Grocery	4
SFr. 9'921	10515	QUICK-Stop	5
SFr. 9'195	10424	Mère Paillarde	6
SFr. 8'623	10514	Ernst Handel	7
SFr. 6'635	10776	Ernst Handel	8
SFr. 6'475	10607	Save-a-lot Markets	9
SFr. 6'375	10612	Save-a-lot Markets	10

**Bild 7.2:** Beispiel eines Berichts

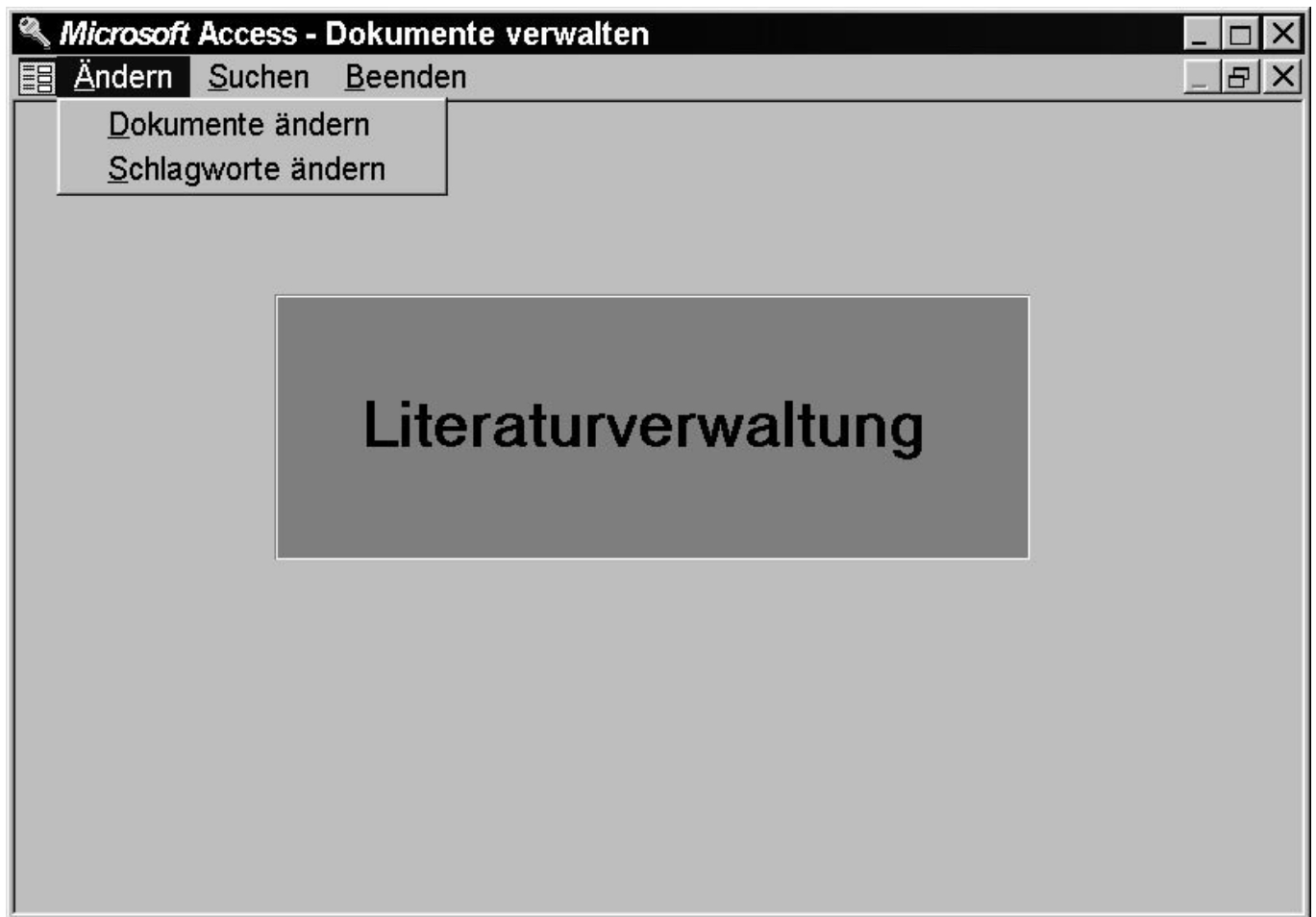
## Verfahren

1. Mitarbeiter erstellt am Bildschirm ein **Formularskelett**.
2. Kunde füllt das ausgedruckte Formularskelett **handschriftlich** aus.
3. Kunde übermittelt das ausgefüllte Formular per **Fax**.
4. Rechner speichert das übermittelte Formular als **graphische Datei**.
5. Formularsoftware übersetzt die Faxdatei in eine Textdatei (**OCR**).
6. Formularsoftware übergibt Textdatei einer Paradox-**Datenbank**.
7. Paradox konvertiert Textformat in sein Datenbank**format**.

### Beispiel 7.3: Automatische Formularverwaltung



# Menüoptionen in einem Formular darstellen

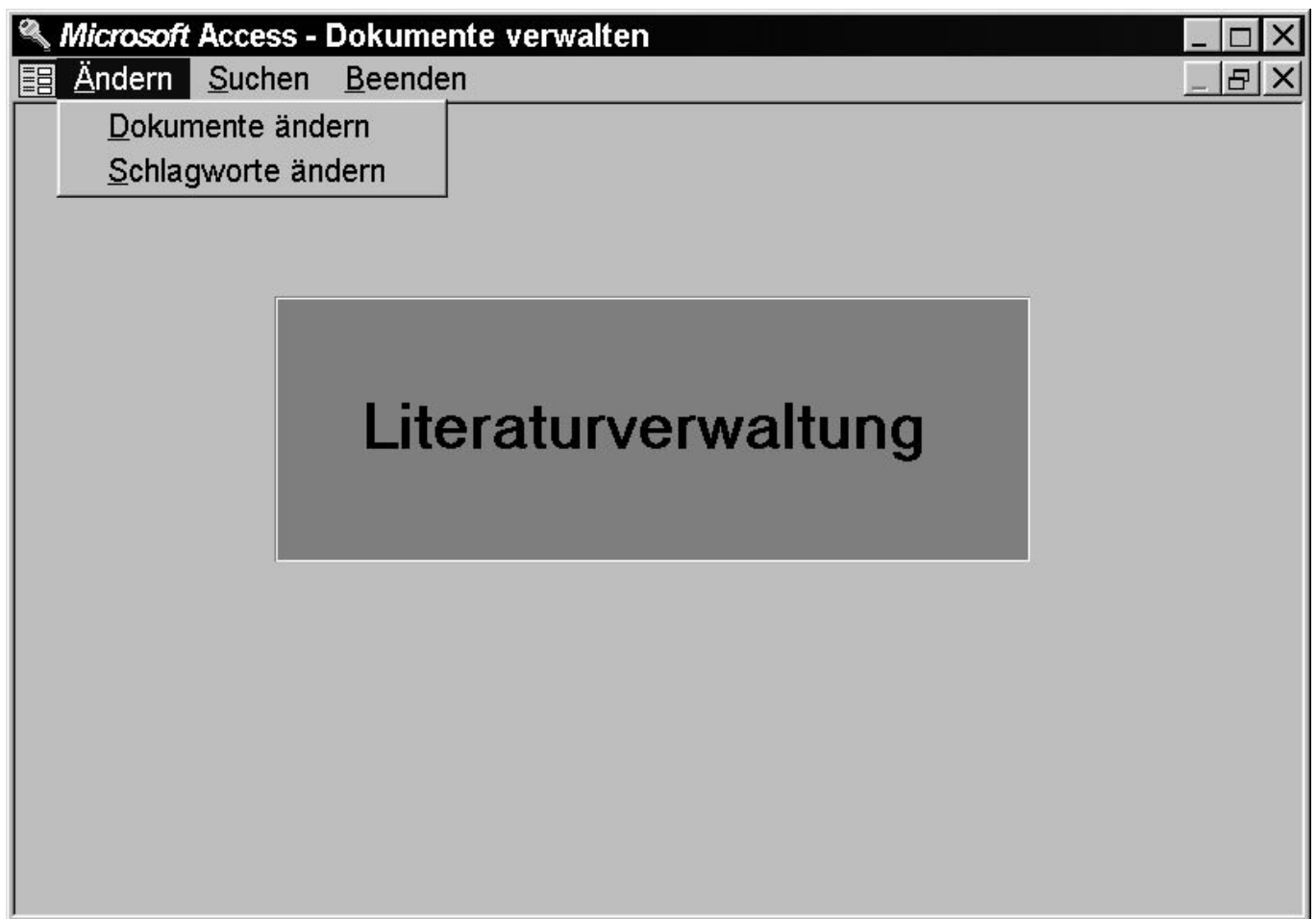


**Formular 7.4: Dokumente\_verwalten**

## Lokalisieren Sie ...

- Menü
- Menüleiste

# Aus einem Unterformular auswählen



Formular 7.5: Dokumente\_ändern

## Lokalisieren Sie ...

- Bezeichnungsfelder
- Textfelder (Bearbeitungsfelder)
- Unterformular

# Schlagworte ändern

The screenshot shows the 'Microsoft Access - Dokumente ändern' window. The title bar includes the Microsoft Access icon and standard window controls. Below the title bar is a 'Zurück' button. The main area is titled 'Dokumente ändern'. It contains several fields: 'Dokumentnummer' (1), 'Autor' (Wiederhold, Gio), 'Titel' (Dateiorganisation in Datenbanken), 'Verlag' (McGraw-Hill), and 'Jahr' (1989). The 'Schlagworte' field is open, showing a list of keywords: 'Dateien', 'Datenbanken', 'Programmierung' (selected), 'Client/Server-Systeme', 'Dateien', 'Datenbanken', 'Datenentwurf', 'Programmierung', and 'Verteilte Datenbanken'. At the bottom, there is a 'Datensatz:' section with navigation buttons and a page indicator '1 von 8'.

**Microsoft Access - Dokumente ändern**

Zurück

**Dokumente ändern**

**Dokumentnummer** 1

**Autor** Wiederhold, Gio

**Titel** Dateiorganisation in Datenbanken

**Verlag** McGraw-Hill

**Jahr** 1989

**Schlagworte**

- Auswählen...
- Dateien
- Datenbanken
- Programmierung
- \* Client/Server-Systeme
- Dateien
- Datenbanken
- Datenentwurf
- Programmierung
- Verteilte Datenbanken

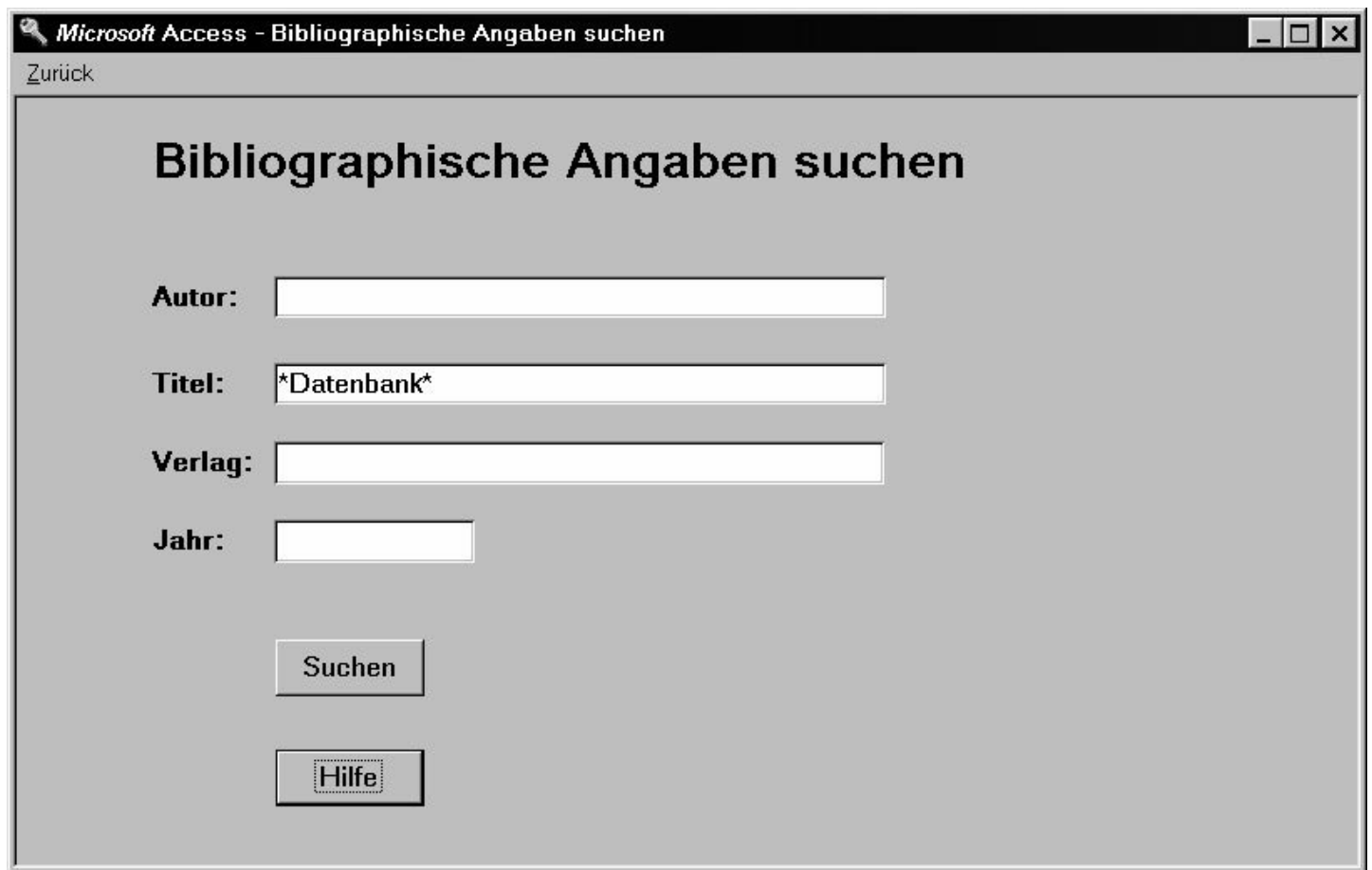
Datensatz: 1 von 8

**Formularausschnitt 7.6: Schlagworte ändern**

## Lokalisieren Sie ...

- Textfelder

# Mit Platzhaltern suchen - Ereignisprozedur



The screenshot shows a Microsoft Access window titled "Microsoft Access - Bibliographische Angaben suchen". Inside the window, there is a form with the title "Bibliographische Angaben suchen". The form contains four input fields with labels: "Autor:", "Titel:", "Verlag:", and "Jahr:". The "Titel:" field contains the text "\*Datenbank\*". Below the input fields are two buttons: "Suchen" and "Hilfe". A "Zurück" button is located in the top left corner of the form area.

Formular 7.7: Bibliographisches\_suchen

## Lokalisieren Sie ...

- Schaltflächen

# Aus Listenfeldern wählen - Ereignisprozedur

**Schlagworte suchen**

Client/Server-Systeme
<b>Dateien</b>
<b>Datenbanken</b>
Datenentwurf
Programmierung
Verteilte Datenbanken

Operator

☒ UND

☐ ODER

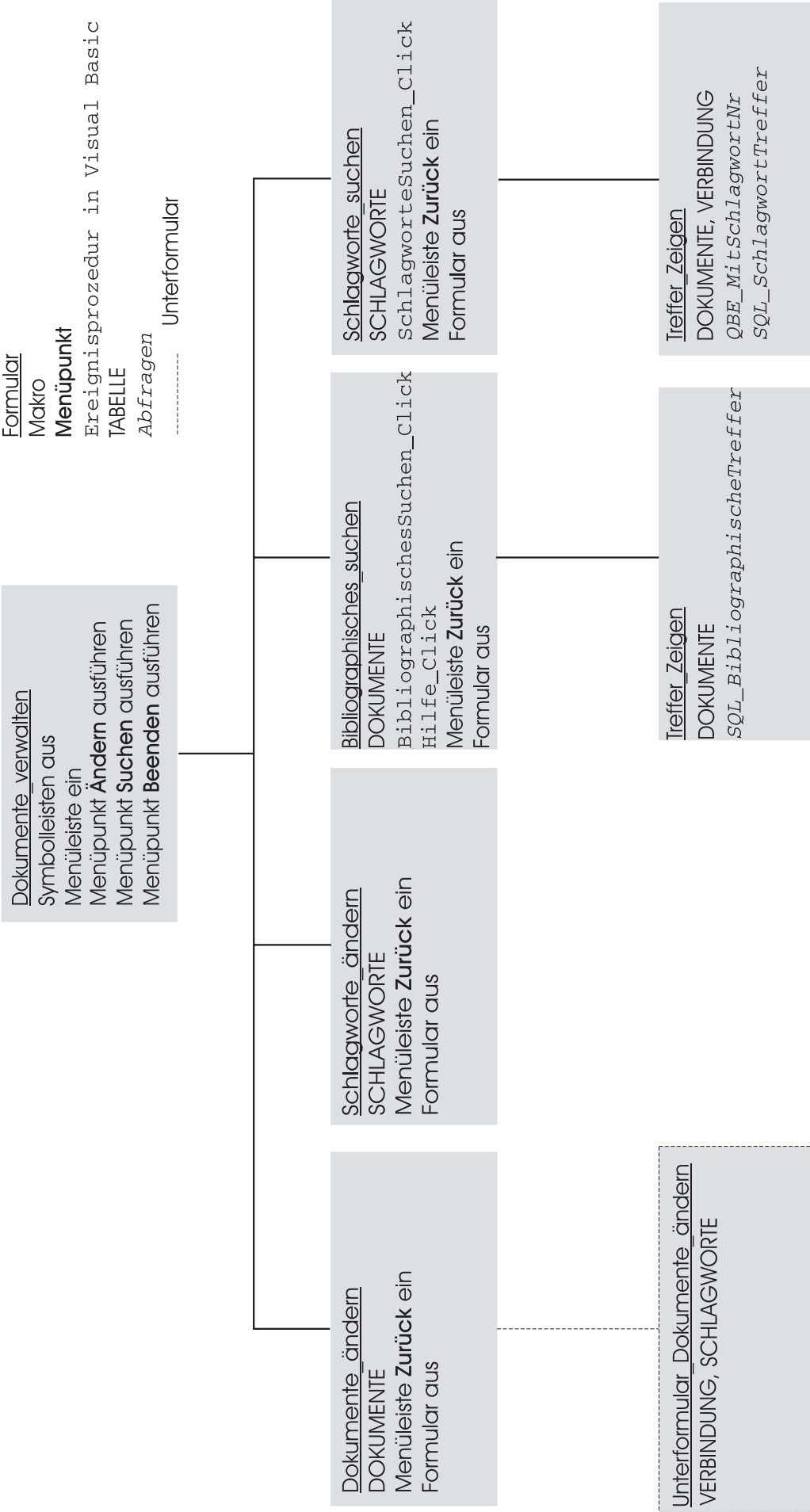
Suchen

Formularausschnitt 7.8: Schlagworte\_suchen

## Lokalisieren Sie ...

- Listenfeld
- Optionsgruppe

# Datenbankobjekte hierarchisch darstellen



# Makrobefehle zur Literaturverwaltung

<i>Makrobefehl</i>	<i>Aufgabe</i>
<b>Symbolleisten_aus</b>	blendet unnötige Symbole (engl. icons) aus
<b>Menüpunkt_Ändern_ausführen</b>	öffnet je nach Wahl die Formulare <i>Dokumente_ändern</i> oder <i>Schlagworte_ändern</i>
<b>Menüpunkt_Suchen_ausführen</b>	öffnet je nach Wahl die Formulare <i>Bibliographisches_suchen</i> oder <i>Schlagworte_suchen</i>
<b>Menüpunkt_Beenden_ausführen</b>	verlässt die Anwendung
<b>Menüleiste_ein</b>	fügt die Menüs Ändern, Suchen und Beenden in die Menüliste ein
<b>Menüleiste_Zurück_ein</b>	fügt das Menü Zurück in die Menüleiste ein
<b>Formular_aus</b>	schaltet das laufende Formular aus

Übersicht 7.10: Datenbankobjekte "Makros"

# Tabellenentwurf

<i>Name</i>	<i>Typ</i>	<i>Eingabe erforderlich</i>	<i>Leere Zei- chenfolge</i>	<i>indiziert</i>
<u>Dokumentnummer</u>	Zahl			( )
Autor	Text			
Titel	Text			
Verlag	Text			
Jahr	Text			

**Tabelle 7.11: DOKUMENTE**

<i>Attributname</i>	<i>Typ</i>	<i>Eingabe erforderlich</i>	<i>Leere Zei- chenfolge</i>	<i>indiziert</i>
<u>Schlagwortnummer</u>	AutoWert	(generiert)	(generiert)	( )
Schlagwort	Text			

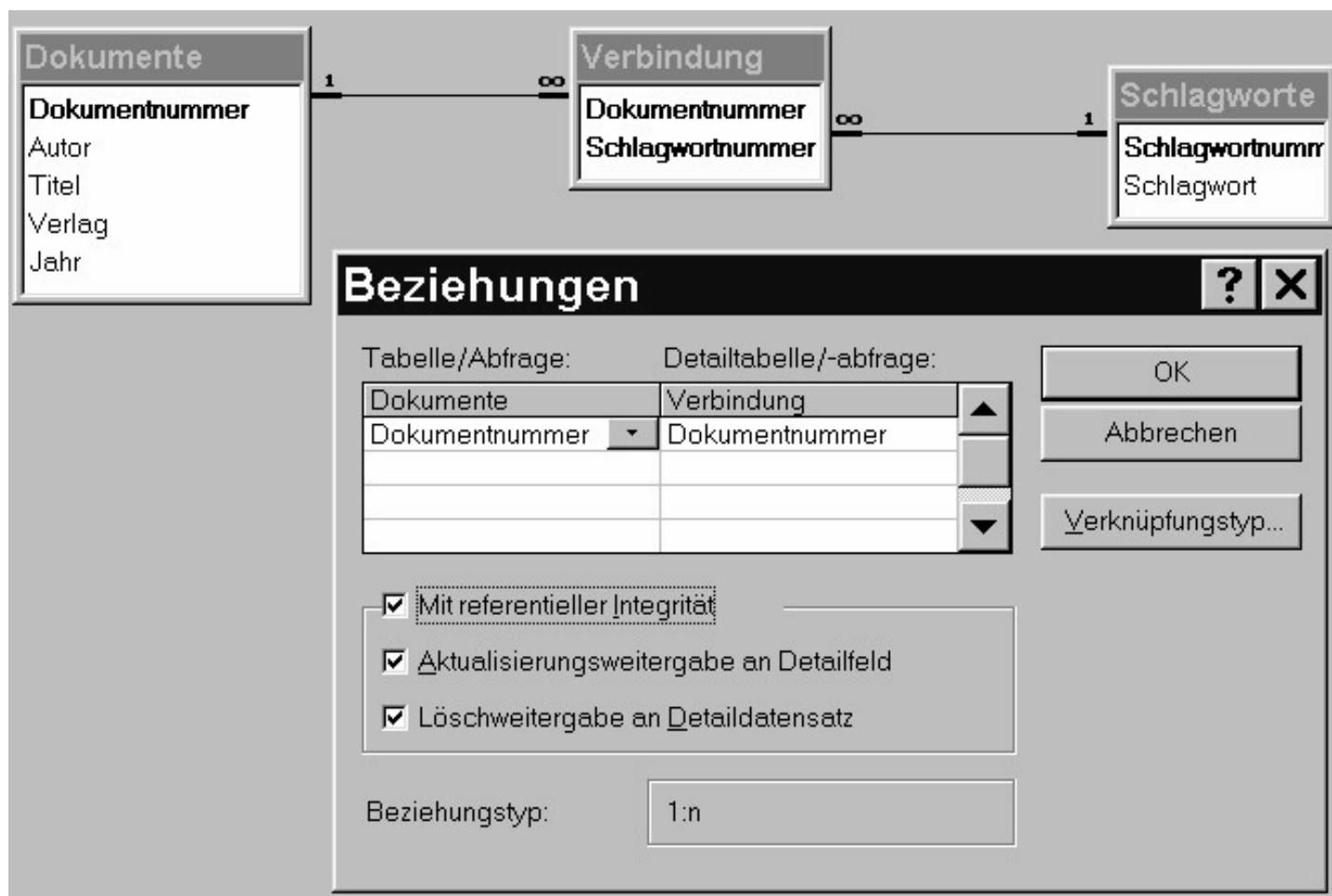
**Tabelle 7.12: SCHLAGWORTE**

<i>Attributname</i>	<i>Typ</i>
<u>Dokumentnummer</u>	Zahl
<u>Schlagwortnummer</u>	Zahl

**Tabelle 7.13: VERBINDUNG**



# Beziehungen in MS Access verwalten



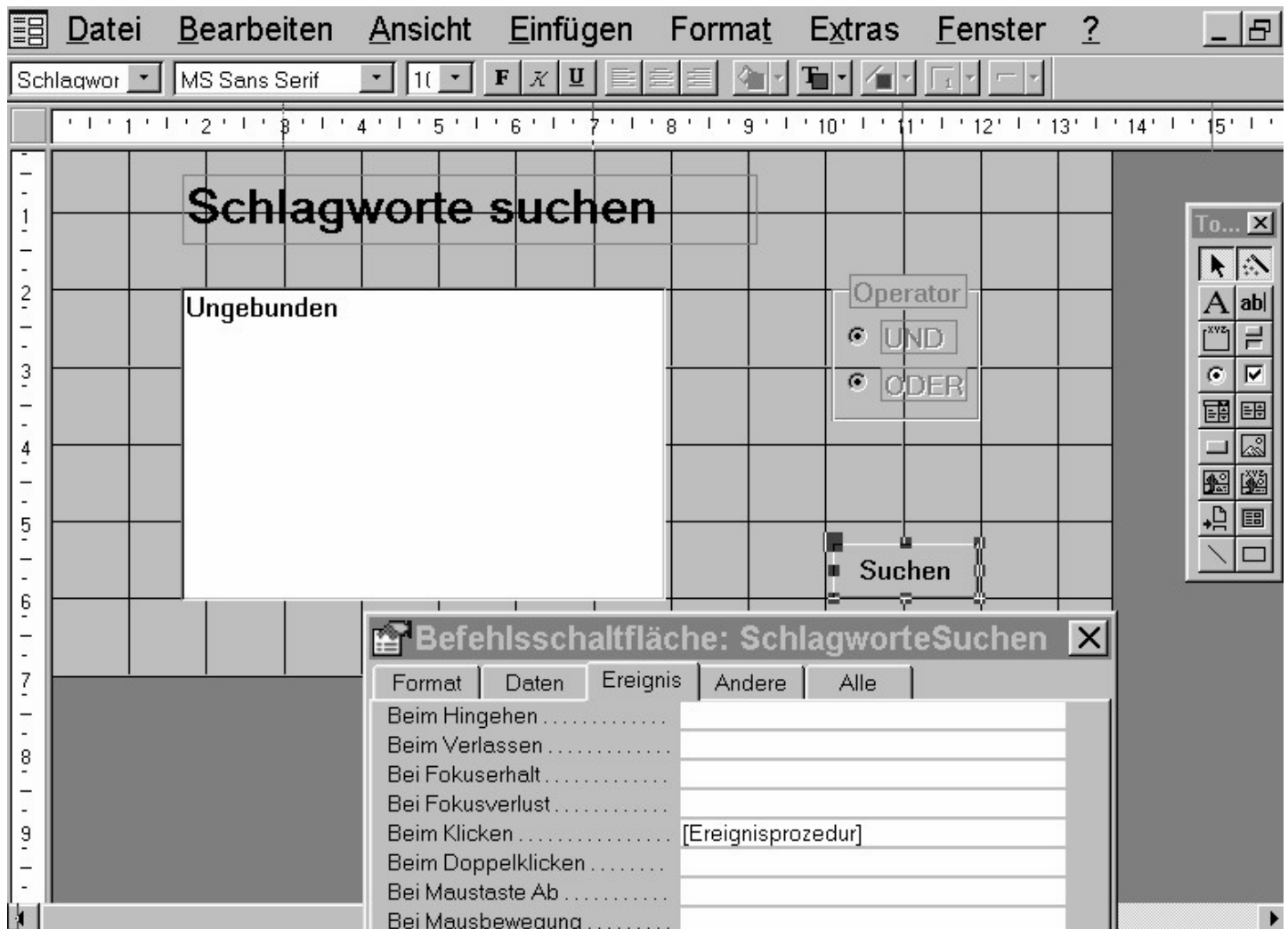
**Datenbankstrukturdiagramm 7.14:** Anwendung Literaturverwaltung

# Formular mit einer SQL-Abfrage öffnen

Dokumentnummer	Autor	Titel	Verlag	Jahr
2	Ullman, J.D.	Database and Knowledge Base	Computer Science Pres	1988
1	Wiederhold, Gio	Dateiorganisation in Datenbanken	McGraw-Hill	1989
4	Elder, J.	Construction of Data Processing	Prentice-Hall	1984
0				

**Formular 7.15: Treffer\_zeigen**

# Formular entwerfen - Ereignis identifizieren



**Formular 7.16:** Schlagworte suchen in der Entwurfssicht

# Ereignisprozeduren entwerfen

Formulare und Steuerelemente anordnen

*Bsp. Bezeichnungs- und Textfelder sowie Schaltflächen*

Ereignisse identifizieren

(Benutzeraktion, Programmaktion oder Systemaktion)

*Bsp. Klick auf ein Steuerelement*

Reaktionen auf Ereignisse als Makro- oder VBA-Ereignisprozeduren programmieren

---

# Ereignisorientiert programmieren

---

## Ereignisprozedur **SchlagworteSuchen\_Click()**

```
Falls Zahl der gewählten Schlagworte > 0
  Falls UND-Verknüpfung gewählt
    Erzeuge aus den Schlagworten einen UND-Befehl
  sonst % falls ODER-Verknüpfung gewählt
    Erzeuge aus den Schlagworten einen ODER-Befehl
  Öffne Formular Treffer_zeigen mit dem erzeugten Befehl
```

## Ereignisprozedur **Schlagwortliste\_AfterUpdate()**

```
Falls Zahl der gewählten Schlagworte > 1
  Aktiviere die Optionsgruppe
sonst
  Deaktiviere die Optionsgruppe
```

Ausgewählte Ereignisprozeduren

# Ereignisprozedur in VBA deklarieren - 1. Teil

```
private sub SchlagworteSuchen_Click()  
const UND = 1  
  
const undStart = _ 3)  
    "select Dokumentnummer,Autor,Titel,Verlag,Jahr &_  
    from QBE_MitSchlagwortNr &_  
    where QBE_MitSchlagwortNr.Schlagwortnummer = "  
  
const undMitte = _  
    "and Dokumentnummer in &_  
    (select Dokumentnummer from QBE_MitSchlagwortNr &_  
    where QBE_MitSchlagwortNr.Schlagwortnummer = "  
  
const oderStart = "select distinct &_  
    Dokumentnummer, Autor, Titel, Verlag, Jahr &_  
    from QBE_MitSchlagwortNr &_  
    where Schlagwortnummer = "  
  
dim lfDb as database  
dim Snapshot as recordset 1)  
  
    ' Listenfeld Schlagwortliste 2)  
dim i as variant  
dim NrSchlagwort as integer  
  
dim SQL as string
```

## Programm 7.17: Deklarationen der Ereignisprozedur der Schaltfläche Suchen

## Ereignisprozedur in VBA programmieren - 2.

```
if Schlagwortliste.ItemsSelected.Count = 0 then 2)
    MsgBox "Schlagwort wählen"
else 3)
    SQL = ""
    if Forms![Schlagworte_suchen]!Optionsgruppe=UND then
        for each i in Schlagwortliste.ItemsSelected
            NrSchlagwort = Schlagwortliste.ItemData(i)
            if SQL = "" then
                SQL = NrSchlagwort
            else
                SQL = SQL & undMitte & NrSchlagwort & ")"
            end if
        next
        SQL = undStart & SQL & ";"
    else '-- ODER 3)
        for each i in Schlagwortliste.ItemsSelected
            NrSchlagwort = Schlagwortliste.ItemData(i)
            if SQL = "" then
                SQL = NrSchlagwort
            else
                SQL = SQL & " OR Schlagwortnummer=" & NrSchlagwo
            end if
        next
        SQL = oderStart & SQL & ";"
    end if

    set lfDb = CurrentDb()
    set Snapshot=lfDb.OpenRecordset(SQL, dbOpenSnapsh) 1)
    if not Snapshot.EOF then
        DoCmd.OpenForm "TrefferZeigen" 4)
        Forms!Treffer_zeigen.RecordSource = SQL
    else
        MsgBox "Keine Datensätze gefunden"
    end if
end if
```

# Listenfeld mit einer SQL-Abfrage öffnen

Ereignisprozedur `Schlagwortliste_AfterUpdate()`

**Falls** Zahl der gewählten Schlagworte > 1

Aktiviere die `Optionsgruppe`

**sonst**

Deaktiviere die `Optionsgruppe`



## Schlagworte suchen

Client/Server-Systeme
Dateien
Datenbanken
Datenentwurf
Programmierung
Verteilte Datenbanken

Operator

☒ UND

☐ ODER

Suchen

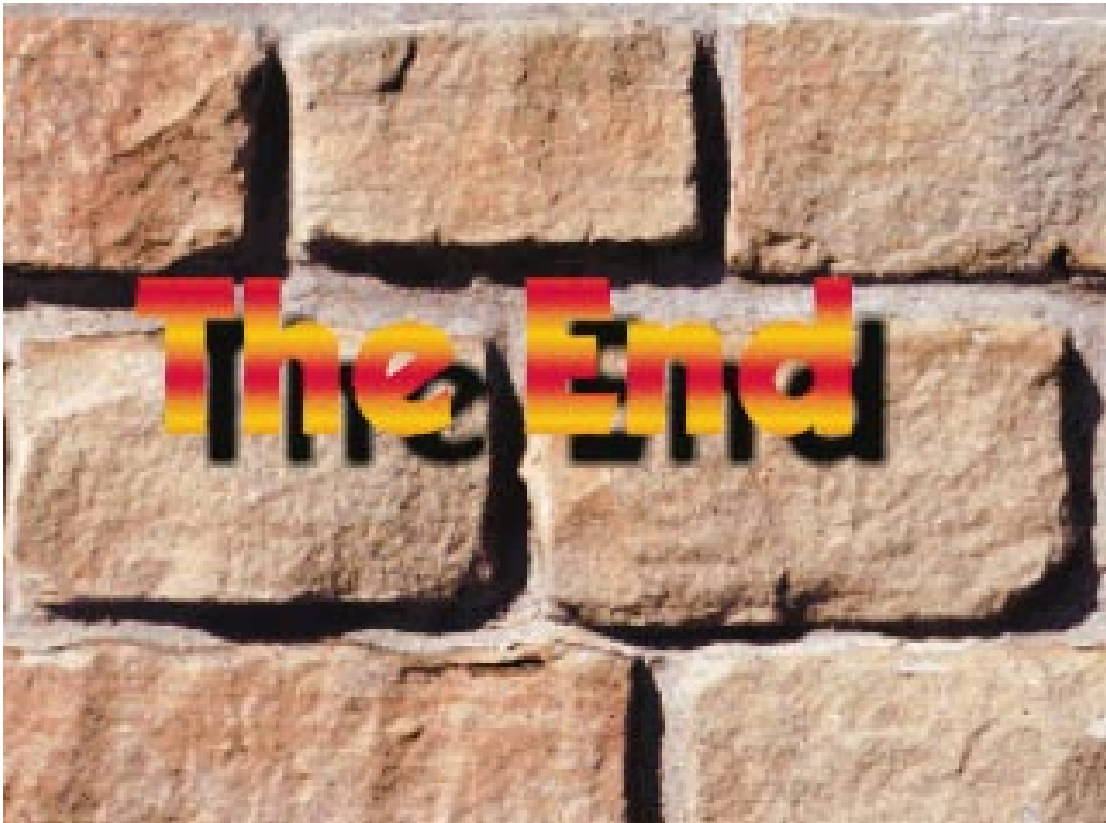


## Formular mit einer SQL-Unterabfrage öffnen

```
select Dokumentnummer, Autor, Titel, Verlag, Jahr
from QBE_MitSchlagwortNr
where QBE_MitSchlagwortNr.Schlagwortnummer=<Nr1>
and Dokumentnummer in (
select Dokumentnummer
from QBE_MitSchlagwortNr
where QBE_MitSchlagwortNr.Schlagwortnumme=<Nr2>
);
```

Unterabfrage rot

Eine Aufgabe wird erst in der Übung ausgeteilt



## Oberstufe

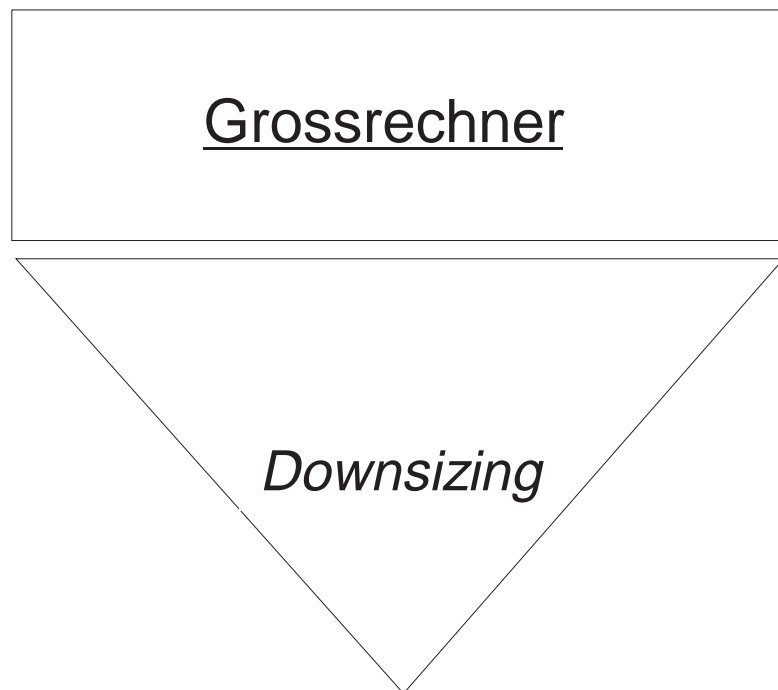
- Informationsmanagement
- Datenbanken (Projektarbeit)
- Softwareentwicklung (Projektarbeit)
- Entscheidungsunterstützende Systeme I und II
- Seminar zu aktuellen Themen der angewandten Informatik

[Web Quiz](#)

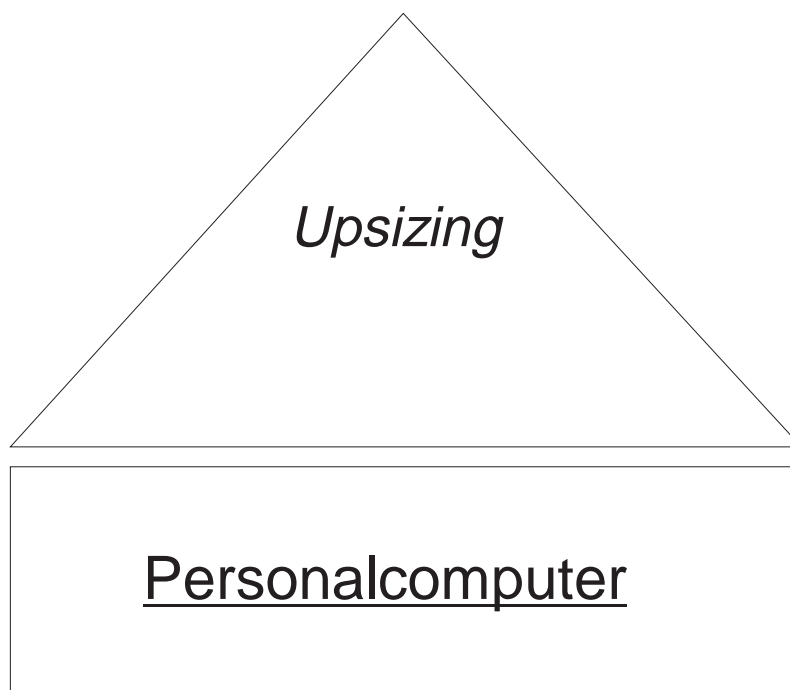
---

## °DV zentralisieren oder verteilen ?

---



## Client/Server-Systeme



**Abb. 8.1:** Downsizing und Upsizing

---

# °Client/Server-Systeme

---

⇒ 8 Client/Server-Systeme

- 9 Verteilte Datenbanksysteme
- 10 Objektorientierte Datenbanksysteme

## *Migration*

Right-, Down- und Upsizing

## *Mehrbenutzerbetrieb*

Time Sharing -, Peer to Peer -, Client/Server -

## *Client/Server-System*

Client, Netz, Server

## *ISO-Kommunikationsmodell*

Schichten, Protokolle

## *Aufgabenverteilung zwischen Client und Server*

Benutzerschnittstelle, Anwendungslogik, Datenbankverwaltung

## *Datenbankprozeduren*

gespeicherte Prozeduren, Trigger

## *Datenschutz*

lokale Datenbanken, Client/Server-Datenbanken

## °DV zentralisieren oder verteilen ?

Ein Unternehmen hat eigens für das Personal- und Rechnungswesen einen **Grossrechner** angeschafft. Die gesamte Datenverarbeitung besteht aus 400 **Arbeitsplatzrechnern**, 20 **Netzrechnern** und dem Grossrechner. Der Grossrechner verursacht zwei Drittel der **Kosten**. Ausserdem sind die Mainframe-Anwendungen **unflexibel**. Zum Beispiel können Reisekostenabrechnungen nach Mitte Monat nicht mehr in die ordentliche Gehaltsabrechnung integriert werden.

Das Unternehmen beschliesst deshalb, die Grossrechner-Anwendungen auf eine **Client/Server**-Umgebung zu portieren. Ein externes Beratungsunternehmen wird mit der Evaluation eines Unix-Systems und der **Neuverkabelung** des Unternehmens beauftragt. Nach einem Jahr ist die Umstrukturierung abgeschlossen. Die Datenhaltung ist weitgehend dezentralisiert, und der Grossrechner kann an den Lieferanten zurückgegeben werden. Die Gehaltsabrechnung ist flexibler, und die Client/Server-Lösung erweist sich auch in anderer Hinsicht als effektiver: Das mittlere Kader und die Unternehmensleitung können **selbständig Anfragen** an das Server-Datenbanksystem richten und **ad hoc**-Auswertungen vornehmen.

### Beispiel 8.2: Downsizing (nach SCHI90)

## °Zwischen Rechnern kommunizieren

<i>System</i>	<i>Komponenten</i>
Time Sharing-System	“intelligenter” Grossrechner und “dumme” Terminals
Peer-to-Peer-Netz	gleichberechtigte Rechner
Client/Server-System	Anwendungs- und Dienstleistungsrechner

### Vergleich 8.3: Lokale Kommunikation zwischen Rechnern

---

# °Rechnernetze

---

## Arten

- **Lokales** Netz (LAN: Local Area Network)
- **Verbund**netz (WAN: Wide Area Network)

## Zweck ist das Teilen von ...

- **Hardware** (Drucker, Modem, Faxgerät, etc.)
- **Daten** (Kundendaten, Personaldaten, etc.)
- **Rechenleistung** (z.B. Hochleistungsrechner)
- **Zugängen** zu andern Netzen (Internet, CompuServe, etc.)

## Komponenten

- **Clients** (Arbeitsplatzrechner )
- **Netzkarten**
- **Netzsoftware**
- **Server** (Dienstleistungsrechner)
- **Leitungen** (Telefonkabel, Koaxialkabel, Glasfaserkabel)



**Kommunikation**

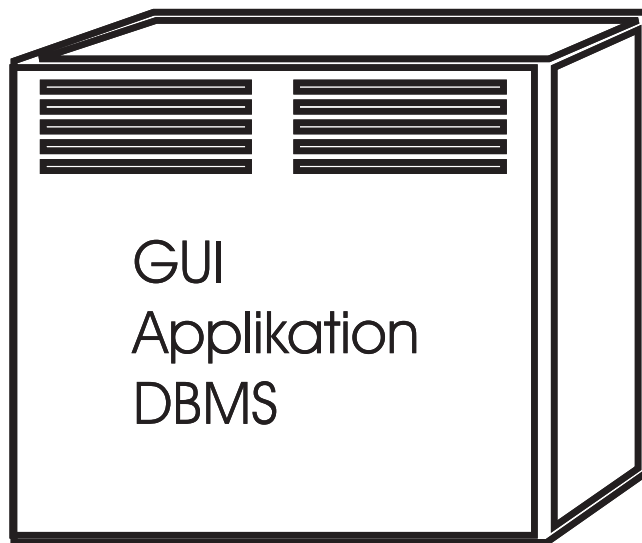
+

**Produktvielfalt**

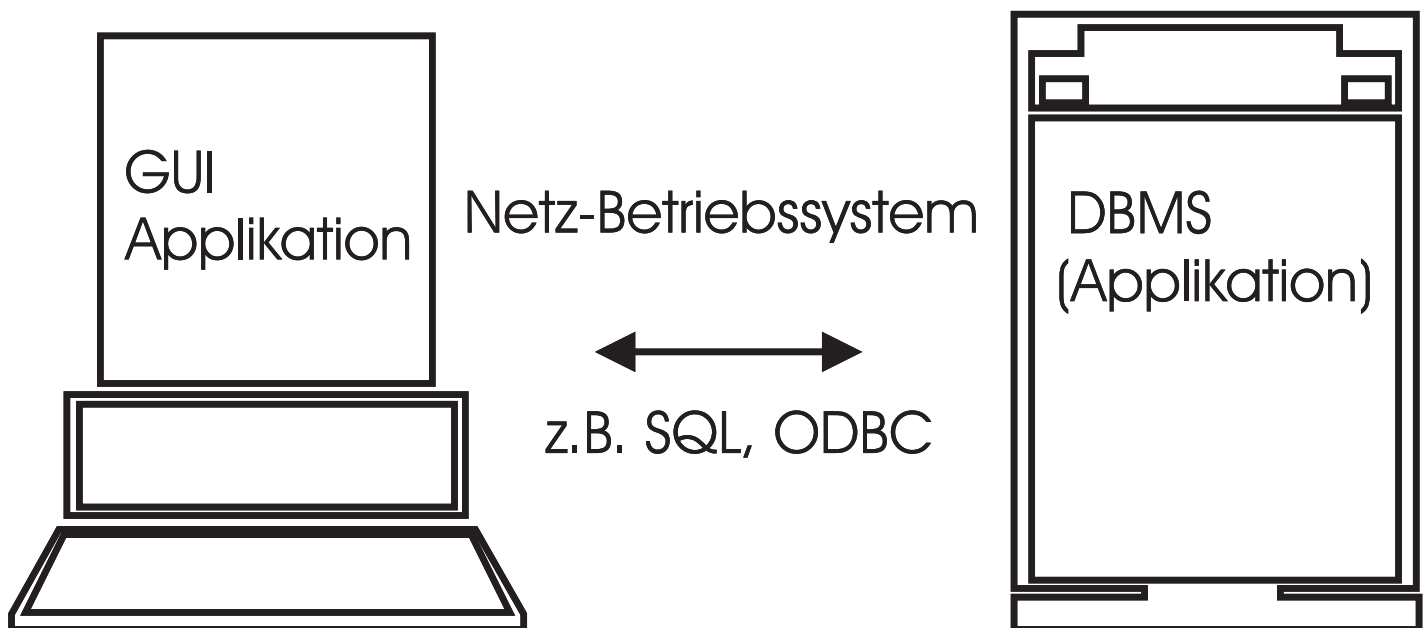
**Notwendigkeit der Standardisierung**

**Schichtenmodell der ISO**

## °Ressourcen dezentralisieren



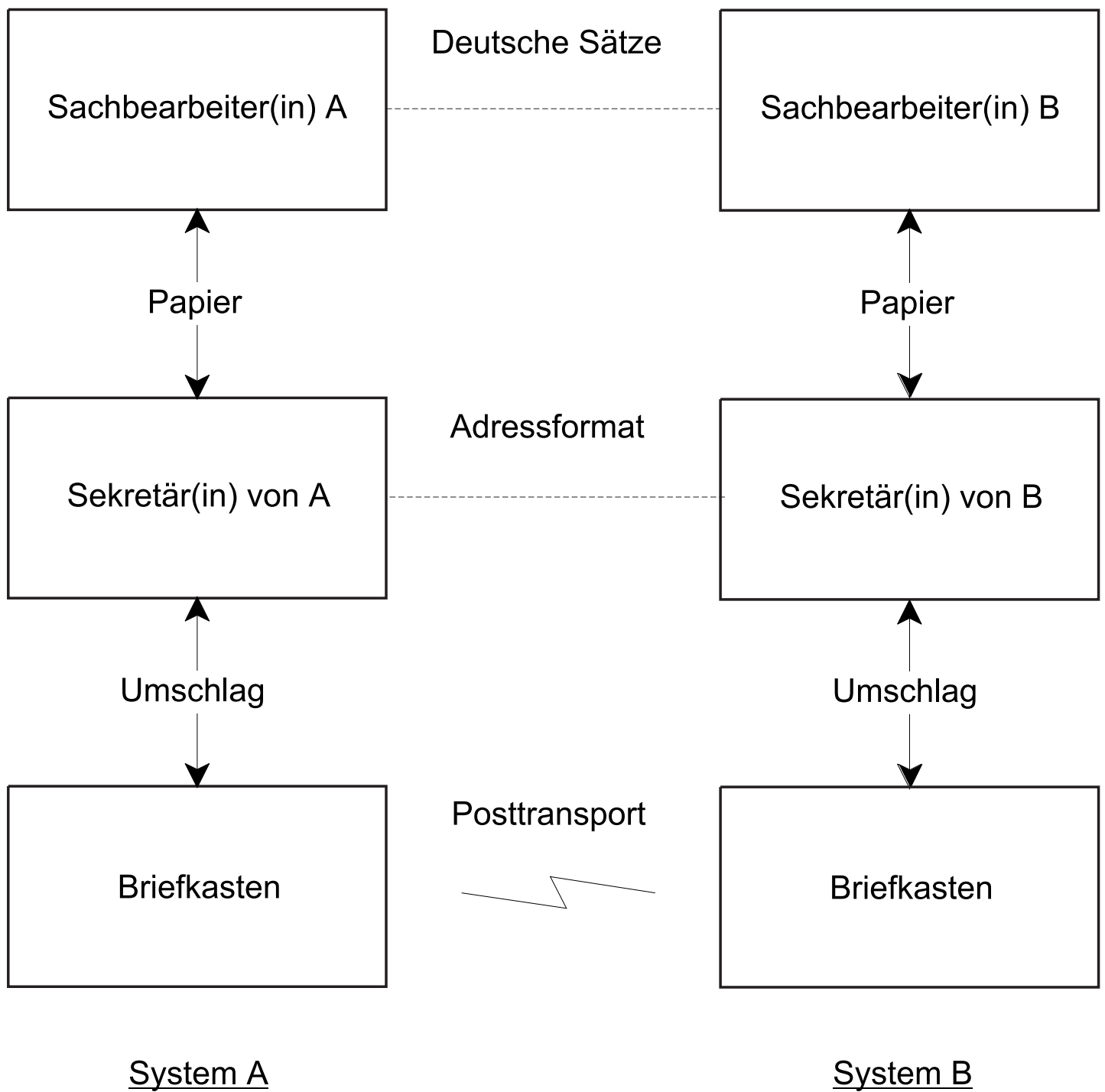
Time Sharing



Client Server

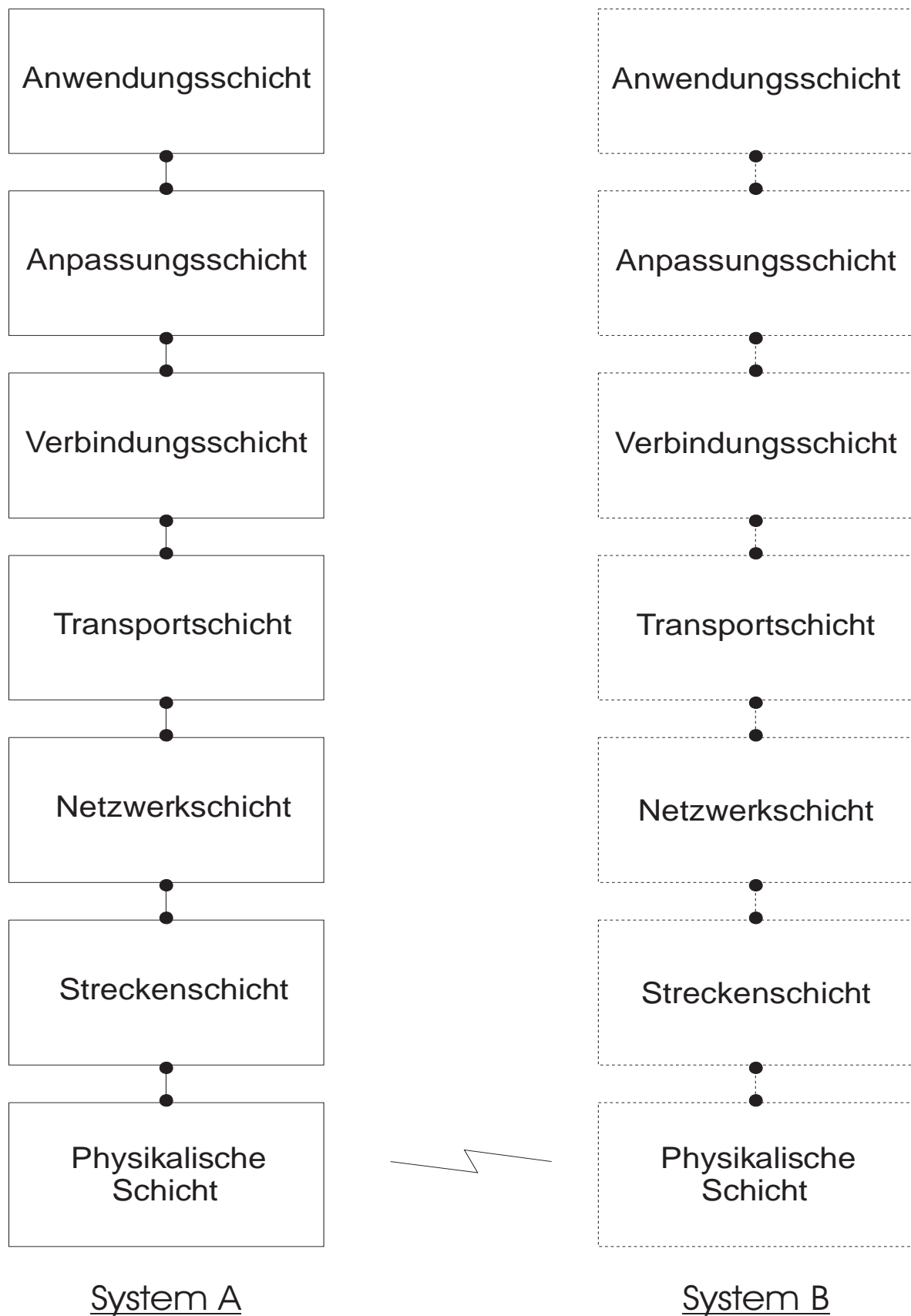
**Bild 8.4:** Vom Grossrechner zu Client/Server-Systemen

# °Das ISO-Schichtenmodell veranschaulichen



**Bild 8.5:** Eine Analogie zum Schichtenmodell von ISO

# °Das ISO-Schichtenmodell - Architektur



**Bild 8.6:** Vertikale und horizontale Kommunikation im Schichtenmodell von ISO

# °Das ISO-Schichtenmodell - Aufgaben

Schicht	regelt zum Beispiel	Bsp.	TCP/IP
Anwendungs-	Dateiübertragung, Terminalemulation	X.400	
Anpassungs-	Formatkonversion (z.B. ASCII ANSI)		Anwendungsschicht
Verbindungs-	Sicherheit (z.B. Passwörtererkennung)		
Transport-	Nachrichten	TCP	Transmission Control Protocol
Netzwerk-	Adressierung (engl. routing)	X.25	Internet Protocol
Strecken-	Nachrichtenformat auf Bitebene	HDL C	Ethernet, Token Ring

**Tabelle 8.7:** Schichtenmodell von ISO

---

# °Client/Server - System

---

Verbund von Hardware und Software, der Aufträge von **Clients** (Arbeitsplatz-Rechnern) durch **Server** (Dienstleistungsrechner) ausführen lässt

## °Client / Server - Aufgabenverteilung

<b>Client</b>	<b>Server</b>
Benutzerschnittstelle verwalten	Daten empfangen/sendern
Benutzereingaben verwalten	Datenbankzugriffe durchführen
Anwendungslogik abarbeiten	Transaktionen steuern
Daten senden/empfangen	Datenintegrität gewährleisten
Ergebnisse formatieren und ausgeben	Sicherheit/Datenschutz gewährleisten

**Übersicht 8.8:** Ein Modell der Aufgabenteilung zwischen Client und Server

## °Client/Server - Aufgaben alternativ verteilen

s e r v e r - zentriert			c l i e n t - zentriert	
<i>verteilte Präsentation</i>	<i>entfernte Präsentation</i>	verteilte Anwendung	<i>entfernte Daten</i>	verteilte Daten
Daten	Daten	Daten	<b>Daten</b>	<b>Daten</b>
Anwendung	Anwendung	<b>Anwendung</b>	Netzwerk	Netzwerk
<b>Präsentation</b>	Netzwerk	Netzwerk	Anwendung	<b>Daten</b>
Netzwerk	<b>Präsentation</b>	<b>Anwendung</b>		Anwendung
<b>Präsentation</b>		Präsentation	Präsentation	Präsentation

Übersicht 8.9: Alternative Modelle der Aufgabenteilung



# °Client/Server - Produkte

## Server - Datenbanksysteme

- Bsp. Oracle
- Bsp. Sybase

## Client - Entwicklungsumgebungen

- **3GL**-Sprachen wie C
- **4GL**-Sprachen wie VBA
- **Objektorientierte** Sprachen wie Delphi Pascal

## Vorteile

- **Aufgabenteilung** unter Minimierung der Netzlast
- **Preis-/Leistungsverhältnis** im Vergleich zum Mainframe
- **Modularität** und **Skalierbarkeit** von Servern und Clients
- **Offenheit** gegenüber Hardware- und Software-Herstellern

## Anspruch und Wirklichkeit

<i><b>Anspruch</b></i>	<i><b>Aber ...</b></i>	<i><b>Beispiel</b></i>
Aufgabenteilung Anwendung - Datenbank	Überschneidung	Validierung auf dem Client
SQL auf dem Server 4GL auf dem Client	Embedded SQL Datenbankprozeduren	Trigger
Offenes System mit Standards	proprietäre Erweiterungen	Datenbankprozeduren

## °Server - Datenbanksysteme

<i>Firma</i>	<i>DBMS</i>	<i>Entwicklungs- werkzeuge</i>	<i>Bemerkungen</i>
Oracle	Oracle	PL/SQL, CDE2	Marktführer, viele Plattformen, CASE
Sybase	SQL Server	PowerBuilder	Upsizing-Tool, CASE
IBM	DB2	Visual Age	seit 1984, gute Grossrech- ner- und SQL-Unterstützung
Informix	Informix SE/Online	NewEra	Multimedia-Unterstützung
Gupta (Centura)	SQLBase	SQLWindows (Centura)	optimiert für MS Windows
Microsoft	SQL Server	Visual C++, Visual Basic	Sybase-Lizenz
Borland	Interbase	Delphi Client/Server	effiziente und flexible visuelle Programmierung in Object Pascal

Übersicht 8.10: Server-Datenbanksysteme

---

## °Geschäftsprozesse automatisieren

---

1. Ein **Mitarbeiter** füllt einen **Reiseantrag** aus.
2. Sein **Vorgesetzter** prüft den Antrag.
3. Das interne “**Reisebüro**” bearbeitet den Antrag.
4. Die **Buchhaltung** erhält eine Kopie.
5. Der **Antragsteller** erhält den vollständig ausgefüllten Antrag zurück.

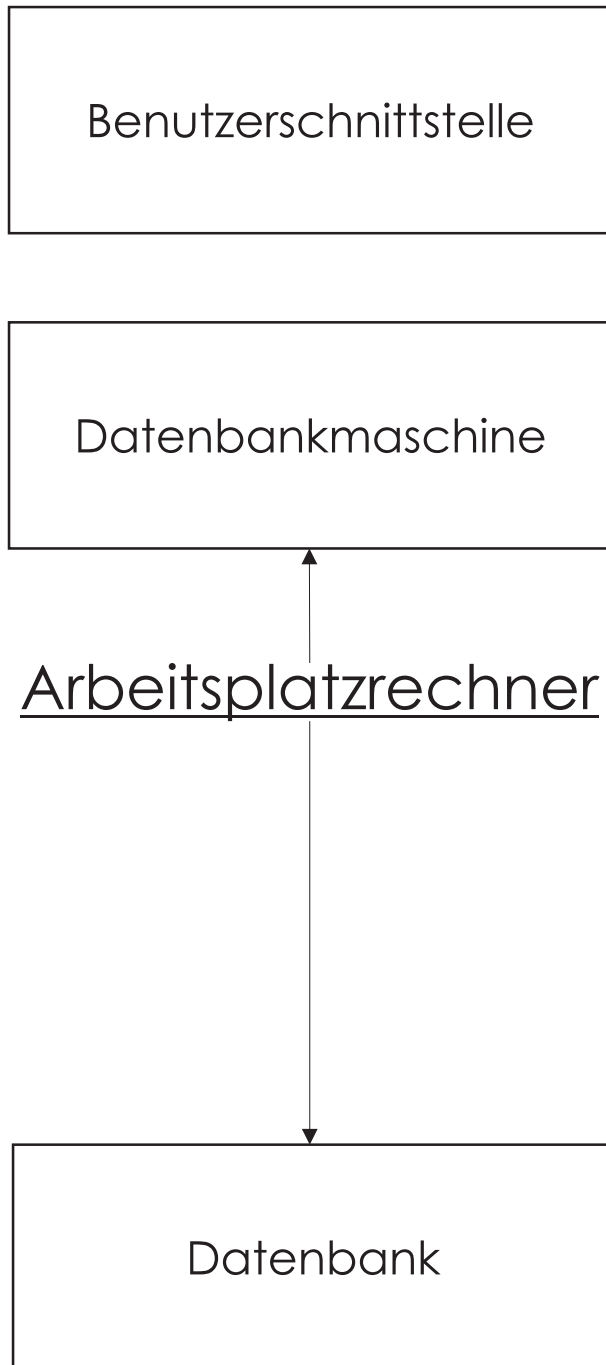
Die Automatisierung einer bislang manuellen Formularverwaltung nutzt die folgenden Möglichkeiten von Notes: die **Dokumentenbank**, die **elektronische Post**, **elektronische Unterschriften** und die **Entwicklungsumgebung**. Der automatisierte Prozess zeichnet sich durch die folgenden Eigenschaften aus:

- Jeder Mitarbeiter findet unter seinen persönlichen Notes-Dokumenten eine Reiseantragsmaske.
- Der Antrag gelangt per elektronische Post an die verschiedenen Adressaten (engl. routing).
- Der Antragsteller erhält periodische Post über den Status der Formularbearbeitung.
- Das Formular gelangt - mit zusätzlichen Informationen über die Reisemodalitäten - wieder an den Mitarbeiter.

### Beispiel 8.11: Eine Workflow-Anwendung mit Lotus Notes

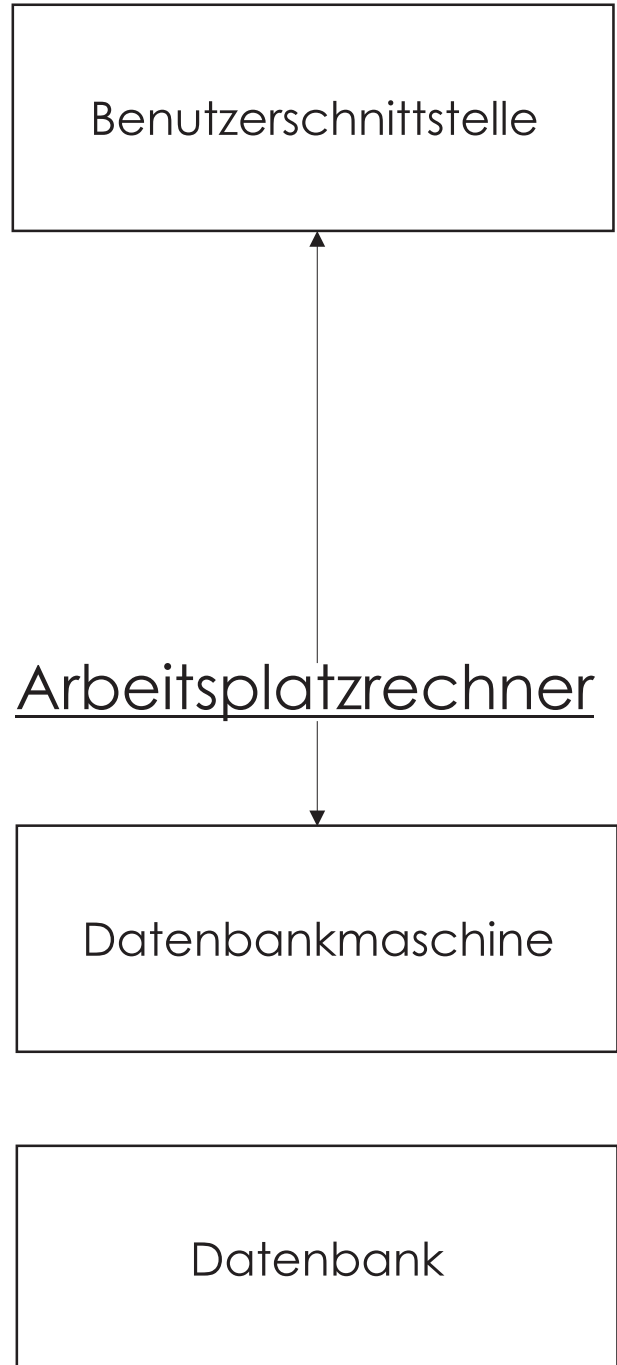
# °Fileserver - und Client/Server - Architekturen

## FILESERVER-MODELL



Server

## CLIENT/SERVER-MODELL



Server

**Bild 8.12:** Client/Server- und Fileserver-Architekturen

# °Fileserver-Transaktionen programmieren

**begintrans ()**

```
Transaktionsfehler = .F.          && F heisst False
on error do Transaktionsfehler && Fehlerbehandl. ein
use N:\FIRMA                      && N ist Netz-Laufwerk
replace all Umsatz with 0         && Umsatz: Feld von FIRMA
on error                          && Fehlerbehandlung aus
if Transaktionsfehler
    rollback()
else
    commit()                      && Transaktionsende
endif
```

```
proc Transaktionsfehler          && Fehlerbehandlung
wait "Transaktionsfehler!"        && Warte auf Taste
Transaktionsfehler =. T.          && T heisst True
```

**Programm 8.13: dBASE-Transaktionen unter einer Fileserver-Architektur**

**grosse Netzbelastung**

## ° **Fileserver** - und **Client/Server** - Netzbelastung

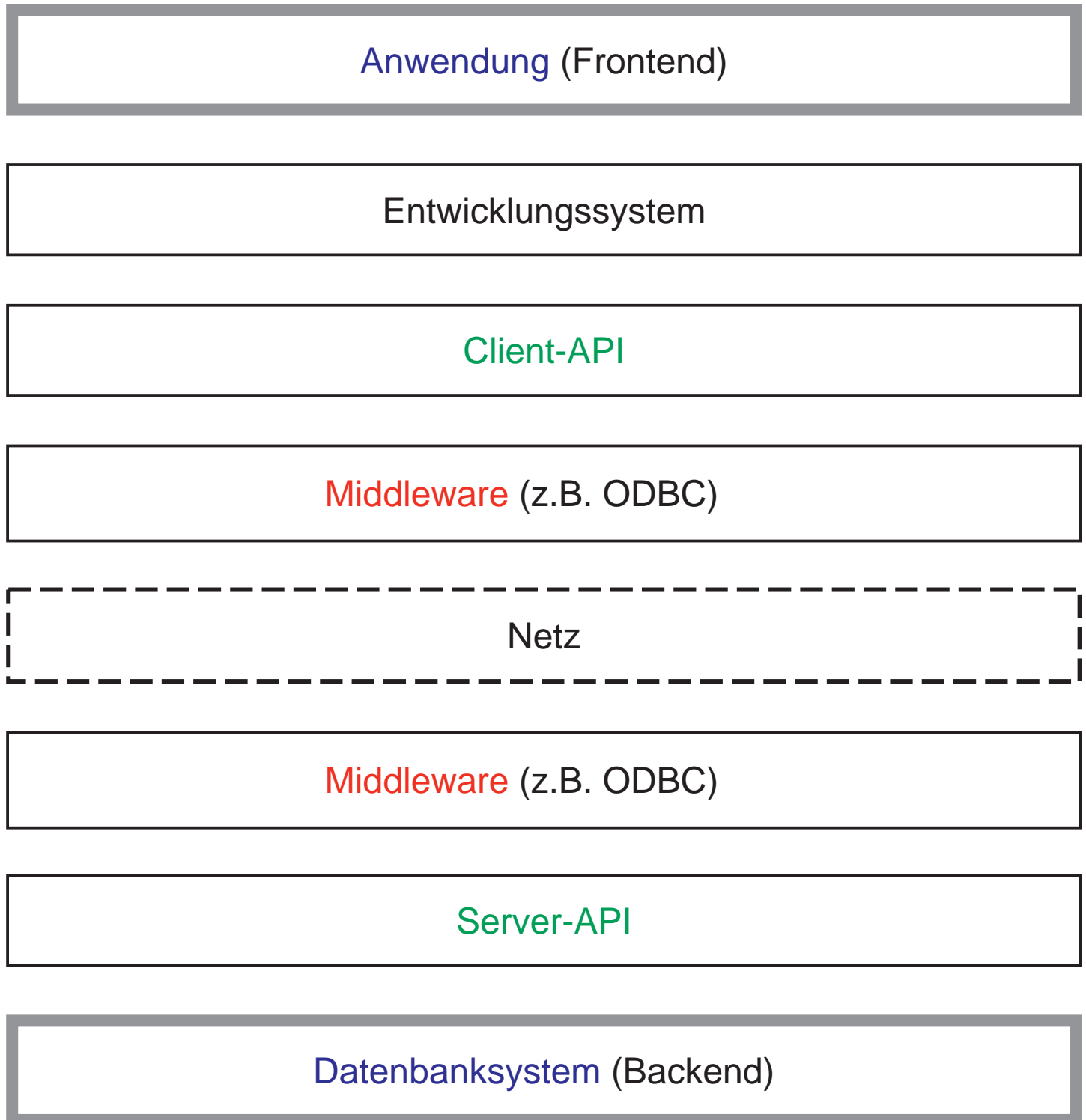
<i>Passiver Fileserver</i>	<i>Netzbelastung</i>
1. Das lokale DBMS fordert Tabelle X und Index	Befehl
2. Der Server sendet Tabelle X und Index	<b>Zwei Dateien</b>
3. Das <b>lokale DBMS</b> sucht die Antwort	-

**Tabelle 8.14:** Suche in einem **passiven** Fileserver-System

<i>Aktiver Server</i>	<i>Netzbelastung</i>
1. Der Client sendet eine SQL-Abfrage	Befehl
2. Das <b>Server-DBMS</b> sucht die Antwort	-
3. Der Server sendet die Antwort	(kleine) Antworttabelle

**Tabelle 8.15:** Suche in einem **aktiven** Client/Server-System

## °Client/Server-Schnittstellen standardisieren



**Bild 8.16:** Softwareschichten in einem Client/Server-System

## °Clients und Server durch ODBC verbinden

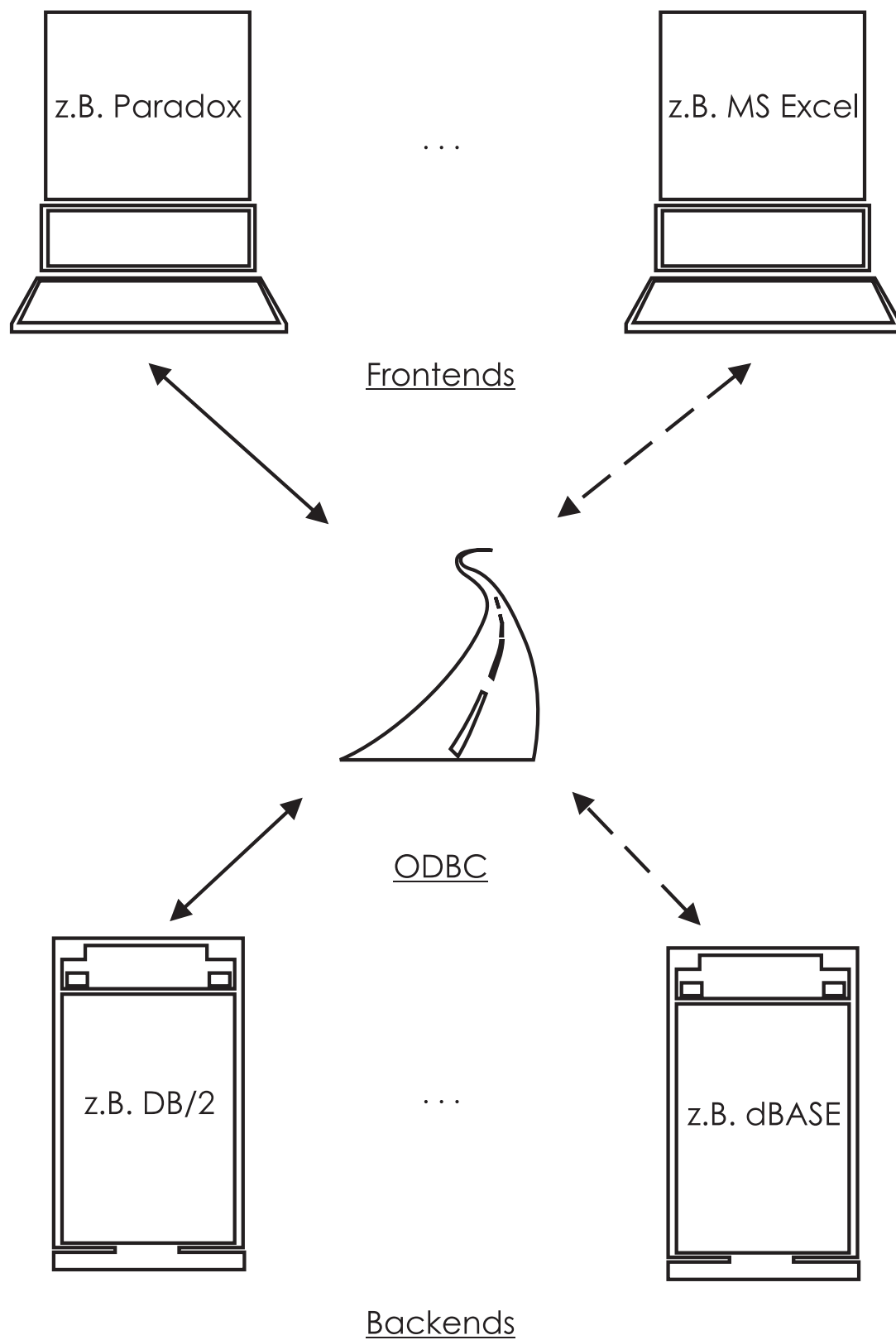
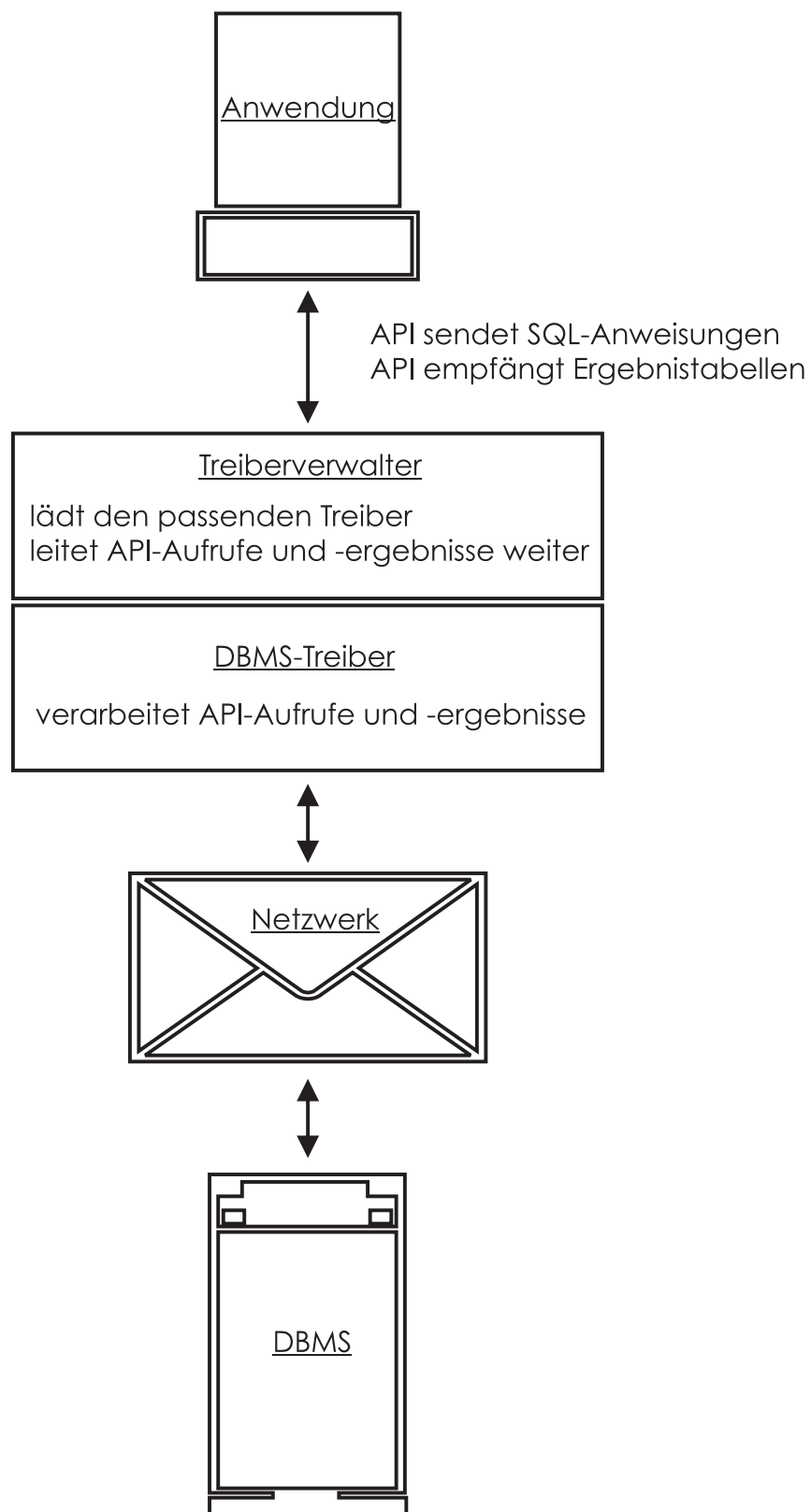


Bild 8.17: ODBC - Herstellerunabhängige Client/Server-Schnittstelle



# °Clients und Server durch ODBC verbinden



**Bild 8.18:** ODBC - Architektur

# °ODBC für Client/Server-Datenbanken

**ODBC** (Open DataBase Connection) :- betriebssystemübergreifender Industriestandard, der die Schnittstelle zwischen **Frontend** (Benutzeroberfläche) und **Backend** (Datenbanksoftware) normiert

## Beispiele

- **Frontend**

Textverarbeitungssoftware

Tabellenkalkulationssoftware

Selbst programmierte Anwendungsschnittstelle

- **Backend**

Bsp. dBase

Bsp. Oracle

- **Anwendungen**

- dBase-Adresse lesen und in einen Brief einfügen

- Finanzkennziffern aus einer Buchhaltungs-Datenbank im Tabellenkalkulationspaket Lotus 1-2-3 berechnen

- **Beispiel einer Netzkonfiguration**

1. QBE-Abfrage unter MS Access auf einem **Windows-Client** eines ersten LAN

2. Versand der Nachricht über **ISDN**

3. Antwort von **Sun-Server** unter Oracle auf einem zweiten LAN

---

## °Gespeicherte Prozeduren entlasten die Clients

---

```
create procedure Loehne(  
    NrAbt char(2))      /* Attribut von MITARBEITER */  
returns(                /* Ausgabeparameter */  
    Lohnsumme decimal(12, 2),  
    Lohndurchschnitt decimal(12, 2)),  
as  
begin  
    select sum(Lohn), avg(Lohn)  
    from MITARBEITER  
    where NrAbt = :NrAbt  
    into :Lohnsumme, :Lohndurchschnitt  
end
```

**Programm 8.19:** Eine gespeicherte Prozedur in Interbase (Borland)

---

# °Legacy- auf Client/Server-Systeme migrieren

---

## **Strategie**

- Wichtig sind die **Erfordernisse des Unternehmens** und nicht die Möglichkeiten der Informationstechnik.
- Migration ist eine laufende Aufgabe. Entwerfe deshalb Informationssysteme auf **Flexibilität**. Ein ideales Informationssystem wird objektorientiert, komponentenorientiert (zum Beispiel OLE-fähig) und portabel sein. Viele Systeme werden ausserdem heterogen und verteilt sein.
- Komplexe Migrationsprozesse können mehrere **Jahre** dauern.
- Gehe schrittweise vor. Globale Entwürfe unterschätzen die Komplexität der Migration.

## **Technik**

- Migriere **Daten**, aber nicht Programme. Veraltete Programme, vor allem benutzerzentrierte, schreibt man besser neu. Moderne 4GL sind meist um vieles produktiver als ältere 3GL.
- Migriere nur die **notwendigsten** Daten.
- Glaube **Anbietern von Migrationswerkzeugen** nur dann, wenn sie demonstrieren, dass ihre Werkzeuge in Deiner Umgebung laufen.

## **Metadaten**

- Achte auf **Spezifikationen** von Daten, Funktionen, Schnittstellen.
- Bestimme einen **Verantwortlichen** für die sorgfältige Verwaltung bestehender und neu anfallender Metadaten.

**Bild 8.20:** Erfahrungen mit Reengineering-Projekten (nach BRO95)

# °Migrationswerkzeuge

<i>Werkzeuge und ihre Funktionen</i>	<i>Beispiele</i>
<b>Gateways</b> verbinden Legacy- und Zielsysteme zur Laufzeit	Textschnittstelle ↔ GUI, SQL ↔ Sprache XY
<b>Analysewerkzeuge</b> analysieren und dokumentieren Daten und Code	Flow Charts aus Code erzeugen
<b>Spezifikationsgeneratoren</b> extrahieren Spezifikationen aus Daten und Code (Reverse Engineering)	Datenbankschema extrahieren
<b>Migrationswerkzeuge</b> i.e.S. portieren Daten und Code in die Zielumgebung	Datendefinitionen eines hierarchischen DBMS in ein relationales DBMS überführen
<b>Testwerkzeuge</b> unterstützen Modul- und Systemtest der neuen Lösung	repräsentative Testdaten erzeugen
<b>Konfigurationshilfen</b> konfigurieren System aus Komponenten automatisch	Versionskontrolle

**Übersicht 8.21: Migrationswerkzeuge**

---

## °Widerstände gegen Client/Server-Migration

---

- fehlende **Motivation** (Mainframe- und COBOL-Umgebungen)
- ungenügende **PC-Hardware** (OS/2 vs. 3270-Terminal)
- keine **Erfahrung** mit kleinen Projekten
- schwierige **Migration von Daten**
- **Abhängigkeiten** von Betriebssystem, GUI und DBMS
-

---

## °9 Verteilte Datenbanksysteme

---

- ⇒ 8 Client/Server-Systeme
- 9 Verteilte Datenbanksysteme
- 10 Objektorientierte Datenbanksysteme

### *Datenbanksystem*

zentrales -

dezentrales -

lokales -

verteiltes - (globales -)

### *Datenunabhängigkeit*

Fragmentierungsunabhängigkeit

Ortsunabhängigkeit

Replikationsunabhängigkeit

### *Ablauf einer globalen Abfrage*

### *Verteilte Transaktionen*

Atomizität

Konsistenz

Replikation

Isolierbarkeit

Serialisierbarkeit

Isolierungsgrade

Dauerhaftigkeit

# ° Daten verteilen

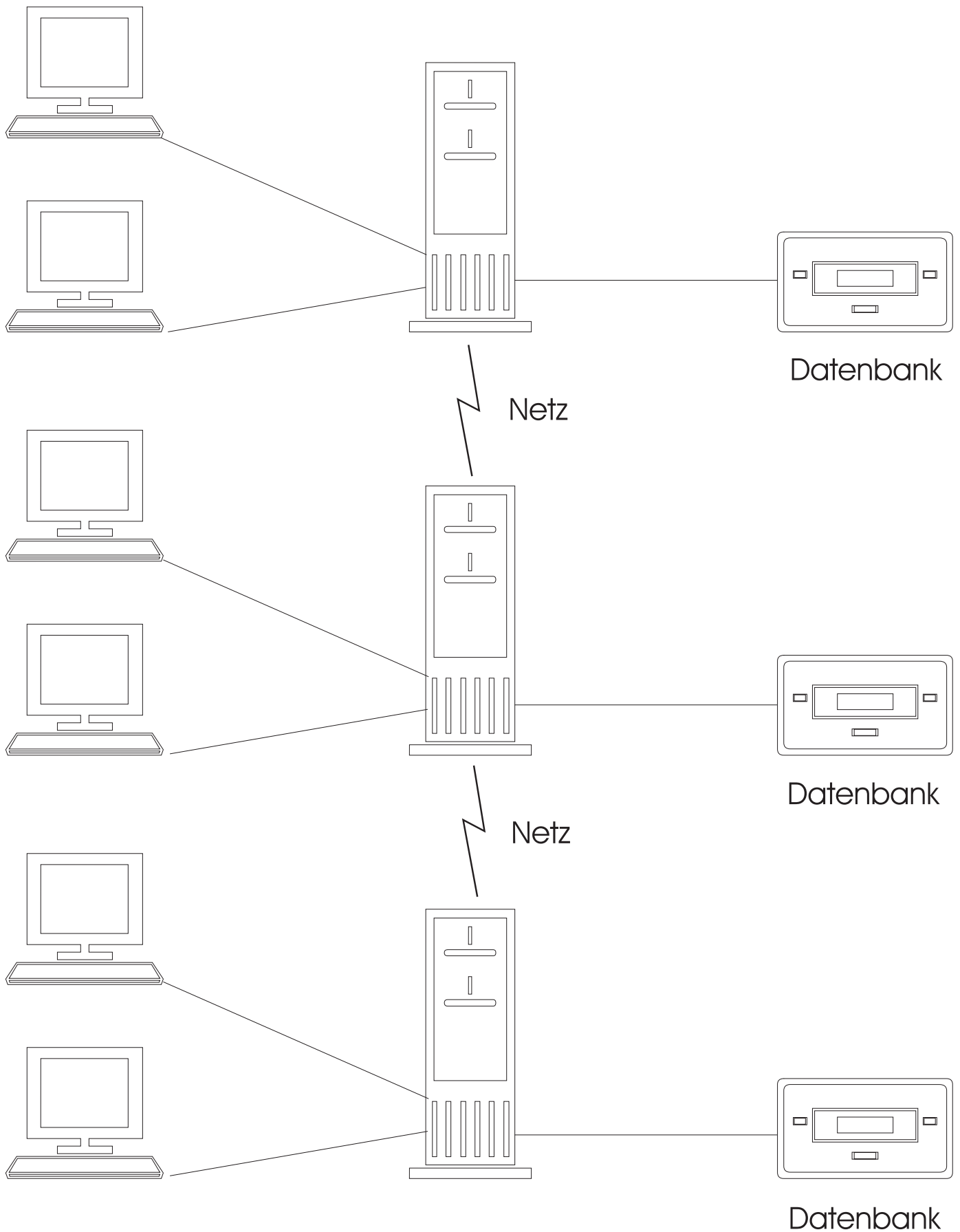


Bild 9.1: Aufbau eines verteilten Datenbanksystems



---

## °Zentralisieren oder verteilen?

---

Eine Warenhauskette koordiniert die lokalen Datenbanksysteme der einzelnen Filialen wie folgt: Jedes Warenhaus verwaltet die täglichen Daten - zum Beispiel Lager-, Verkaufs-, Kunden- und Personaldaten - **lokal**. Über Nacht werden Daten aggregiert, die auch für andere interessierte Filialen und die Zentrale wichtig sind. Auf diese gemeinsam interessierenden Daten kann über ein **verteiltes** Datenbanksystem global zugegriffen werden.

Eine **zentrale** Lösung würde die Daten aller Filialen in einem einzigen entfernten DMBS vereinen. Die wahrscheinlicheren Abfragen auf die eigenen Daten würden also gleich behandelt wie die weniger wahrscheinlichen Fragen an die Daten der andern Filialen. Die Benutzer würden sich wohl bald über hohe Kommunikationskosten und lange Wartezeiten beklagen.

### Beispiel 9.2: Verteilte und zentrale Datenhaltung

# °Client/Server- und verteilte Datenbanksysteme

Vergleichskriterium	Client/Server	Verteilte DBMS
Vernetzung mehrerer Rechner		
Verteiltes Rechnen		
Verteilte Daten	-	
Zahl der Datenbankrechner	i.d.R. einer	mehrere
Autonomie der Rechner (Lokalität)	klein	entfernte gross

**Vergleich 9.3:** Client/Server- und verteilte Datenbanksysteme

## °Zentralisieren oder verteilen?

<i>Vergleichskriterium</i>	<i>Verteilte DBMS</i>	<i>Zentrale DBMS</i>
Lokale Autonomie	+	-
Laufzeiteffizienz	+	-
Speichereffizienz	-	+
Verfügbarkeit	+	-
Übermittlungskosten	+	-
Hardwarekosten	-	+
Erweiterbarkeit	+	-
Technologische Reife	-	+
Komplexität	-	+
Datenschutz	-	+

**Vergleich 9.4:** Verteilte und zentrale Datenbanksysteme (+ Vorteil, - Nachteil)

---

# °Datenunabhängigkeit

---

## a) datenunabhängige Variante

```
select Gehalt  
from ANGESTELLTE  
where Name="Müller"
```

## b) fragmentierungsabhängige Variante

```
select Gehalt  
from ANGESTELLTE_1  
where Name="Müller"
```

### union

```
select Gehalt  
from ANGESTELLTE_2  
where Name="Müller"
```

## c) ortsabhängige Variante

```
select Gehalt  
from ANGESTELLTE_1_AUF_A  
where Name="Müller"
```

### union

```
select Gehalt  
from ANGESTELLTE_2_AUF_B  
where Name="Müller"
```

# °Globale Abfragen beantworten

## 1. Globale Abfrage formulieren

ANGESTELLTE

<u>Nr</u>	Name	Gehalt
	Müller	

## 2. Globale Abfrage in einem Standardformat

```
select Gehalt from ANGESTELLTE where Name = "Müller"
```

## 3. Lokale Abfragen

```
select Gehalt  
from ANGESTELLTE_1_AUF_A  
where Name = "Müller"
```

```
select Gehalt  
from ANGESTELLTE_2_AUF_B  
where Name = "Müller"
```

## 4. Ausführbare Befehle

### a) Erste lokale Abfrage im Dateisystem A

Öffne die Datei ANGESTELLTE\_1\_AUF\_A

**Solange** nicht am Dateiende

Lies einen Satz aus der Datei ANGESTELLTE\_1\_AUF\_A

**Falls** ANGESTELLTE\_1.Name = "Müller"

Schreibe ANGESTELLTE\_1.Gehalt in die Ergebnisdatei

Schliesse die Datei ANGESTELLTE\_1\_AUF\_A

### b) Zweite lokale Abfrage im SQL-System B

## 5. Ergebnistabelle ausgeben

---

# °Globale in lokale Abfragen übersetzen I

---

## 1. Ziel

Antwortzeit, Speicherbedarf und Kommunikationskosten minimieren

## 2. Gegeben

### a) Relationales Schema

KUNDE (Kundennr, Kundenname, Adresse)

BESTELLUNG (Kundennr, Produktnr, Datum, Menge)

PRODUKT (Produktnr, Produktname, Preis).

### b) Fragmentierung

KUNDE seiFragmentiertIn

KUNDE\_1\_auf\_A

KUNDE\_2\_auf\_B

BESTELLUNG seiFragmentiertIn

BESTELLUNG\_1\_auf\_A

BESTELLUNG\_2\_auf\_B

PRODUKT seiFragementiertUndRepliziertIn

PRODUKT\_1\_auf\_A\_und\_B

PRODUKT\_2\_auf\_A\_und\_B

### c) Globale Abfrage

```
select Kundenname
from KUNDE, BESTELLUNG, PRODUKT
where KUNDE.Kundennr = BESTELLUNG.Kundennr
and BESTELLUNG.Produktnr = PRODUKT.Produktnr
and Produktname = "Agenda"
and Datum = "1/12/95"
```

# °Globale in lokale Abfragen übersetzen II

## 3. Übersetzung

a) Globale SQL-Abfrage als *relationale Operationen*

**Projektion**(Kundenname)

**Selektion**(Produktname = "Agenda" and Datum = "1/12/95")

**Verbund**(KUNDE.Kundennr = BESTELLUNG.Kundennr)

KUNDE(Kundennr, Kundenname, Adresse)

**Verbund**(PRODUKT.Produktnr=BESTELLUNG.Produktnr)

PRODUKT(Produktnr, Produktname, Preis)

BESTELLUNG(Kundennr, Produktnr, Datum, Menge)

b) Globale Tabellen durch ihre *Fragmente* ersetzen

Projektion

Selektion

Verbund

**Vereinigung**

KUNDE\_1\_AUF\_A

KUNDE\_2\_AUF\_B

Verbund

**Vereinigung**

PRODUKT\_1\_AUF\_A

(oder das Replikat PRODUKT\_1\_AUF\_B)

PRODUKT\_2\_AUF\_A

(oder das Replikat PRODUKT\_2\_AUF\_B)

**Vereinigung**

BESTELLUNG\_1\_AUF\_A

BESTELLUNG\_2\_AUF\_B

---

## °Globale in lokale Abfragen übersetzen III

---

c) *Relationale Operationen* den *Komponenten-DB* zuordnen

Projektion **auf A**

Selektion **auf A**

Verbund **auf A**

Vereinigung **auf A**

KUNDE\_1\_AUF\_A

KUNDE\_2\_AUF\_B

Verbund **auf A**

Vereinigung **auf A**

PRODUKT\_1\_AUF\_A

PRODUKT\_2\_AUF\_B

Vereinigung **auf B**

BESTELLUNG\_1\_AUF\_A

BESTELLUNG\_2\_AUF\_B

## Freiheitsgrade der Übersetzung

- alternative Reihenfolgen der Schritte a) bis c)
- alternative Algorithmen und Heuristiken *innerhalb* der Schritte a) bis c)
- Durchführung zur Übersetzungszeit oder/und zur Laufzeit.



# °Optimierungsheuristiken

## Heuristik A: **Führe Selektionen möglichst früh** aus

Projektion(Kundenname)

**Selektion**(Produktname = "Agenda" AND Datum = "1/1/96")

Verbund(KUNDE.Kundenr = BESTELLUNG.Kundenr)

KUNDE(Kundenr, Kundenname, Adresse)

Verbund(PRODUKT.Produktnr =

BESTELLUNG.Produktnr)

PRODUKT(Produktnr, Produktname, Preis)

BESTELLUNG(Kundenr, Produktnr, Datum, Menge)



Projektion(Kundenname)

Verbund(KUNDE.Kundenr = BESTELLUNG.Kundenr)

KUNDE(Kundenr, Kundenname, Adresse)

Verbund(PRODUKT.Produktnr = BESTELLUNG.Produktnr)

**Selektion**(Produktname = "Agenda")

PRODUKT(Produktnr, Produktname, Preis)

**Selektion**(Datum = "1/1/96")

BESTELLUNG(Kundenr, Produktnr, Datum, Menge)

## Heuristik B: **Verteile aggregierende Funktionen**

# °Verteilte Transaktionen - ACID-Prinzip

**A**tomic

nicht unterbrechbar (rollback)

**C**onsistent

integritätserhaltend

**I**solated

von konkurrierenden Transaktionen abgeschottet

**D**urable

dauerhaft (commit)

## °Verteilte Transaktionen - Pseudocode 9.5

```
Transaktion Abhebung(Kontonummer, Betrag)
    select Saldo                                % Leseoperation
    from KONTO
    where Kontonr = Kontonummer;
    falls Saldo nicht gefunden dann rollback;
    update KONTO                                % Schreiboperation
    set Saldo = Saldo - Betrag
    where Kontonr = Kontonummer;
    falls Saldo < 0 dann rollback;
    commit;                                     % Transaktionsende

Transaktion Transfer(Kontonr_1, Kontonr_2, Betrag)
    % -- Lastschrift auf Konto 1
    select Saldo                                % Leseoperation
    from KONTO
    where Kontonr = Kontonr_1;
    falls Saldo nicht gefunden dann rollback;
    update KONTO                                % Schreiboperation
    set Saldo = Saldo - Betrag
    where Kontonr = Kontonr_1;
    falls Saldo < 0 dann rollback;
    % -- Gutschrift auf Konto 2
    select Saldo                                % Leseoperation
    from KONTO
    where Kontonr = Kontonr_2;
    falls Saldo nicht gefunden dann rollback;
    update KONTO                                % Schreiboperation
    set Saldo = Saldo + Betrag
    where Kontonr = Kontonr_2;
    commit;                                     % Transaktionsende
```

# °Serialisierbare Geschichten sind isoliert

**Geschichte** := Folge von Lese- und Schreiboperationen aus der gleichen oder verschiedenen Transaktionen

## a) Serielle Geschichte

Transfer.lies (Kontonr\_1.Saldo)  
Transfer.schreibe (Kontonr\_1.Saldo)  
Transfer.lies (Kontonr\_2.Saldo) B  
Transfer.schreibe (Kontonr\_2.Saldo)  
Abhebung.lies (Kontonr\_1.Saldo) C  
Abhebung.schreibe (Kontonr\_1.Saldo)

## b) Serialisierbare Geschichte

Transfer.lies (Kontonr\_1.Saldo) A  
Transfer.schreibe (Kontonr\_1.Saldo)  
Abhebung.lies (Kontonr\_1.Saldo) C  
Abhebung.schreibe (Kontonr\_1.Saldo)  
Transfer.lies (Kontonr\_2.Saldo)  
Transfer.schreibe (Kontonr\_2.Saldo)

## c) Inkonsistente Geschichte

Transfer.lies (Kontonr\_1.Saldo)  
Abhebung.lies (Kontonr\_1.Saldo)  
Transfer.schreibe (Kontonr\_1.Saldo)  
Transfer.lies (Kontonr\_2.Saldo)  
Transfer.schreibe (Kontonr\_2.Saldo)  
Abhebung.schreibe (Kontonr\_1.Saldo)

---

# °Serialisierbare Geschichten durch Sperren

---

## Sperrobjekte (Granularität)

Datenbank

Tabelle

Speicherseite (engl. page)

Satz

Feld

## Gesperrte Operationen

Leseoperation

Schreiboperation

## Sperrgrund

implizite -

explizite -

# °Sperrern isolieren - Beispiel

## a) Inkonsistente Geschichte

Transfer.lies (Kontonr\_1.Saldo)  
Abhebung.lies (Kontonr\_1.Saldo)  
Transfer.schreibe (Kontonr\_1.Saldo)  
Transfer.lies (Kontonr\_2.Saldo)  
Transfer.schreibe (Kontonr\_2.Saldo)  
Abhebung.schreibe (Kontonr\_1.Saldo)

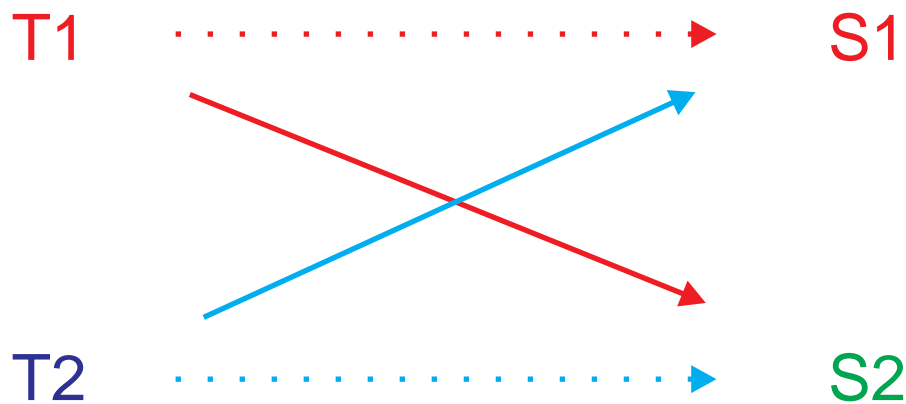
## b) Geschichte mit Sperren

Transfer.lies (Kontonr\_1.Saldo)  
% — Sperre Kontonr\_1.Saldo  
Abhebung.lies (Kontonr\_1.Saldo)  
% wartet auf Freigabe  
Transfer.schreibe (Kontonr\_1.Saldo)  
% — Entsperrung Kontonr\_1.Saldo  
Transfer.lies (Kontonr\_2.Saldo)  
Transfer.schreibe (Kontonr\_2.Saldo)  
Abhebung.schreibe (Kontonr\_1.Saldo)

## c) Neu geordnete Geschichte

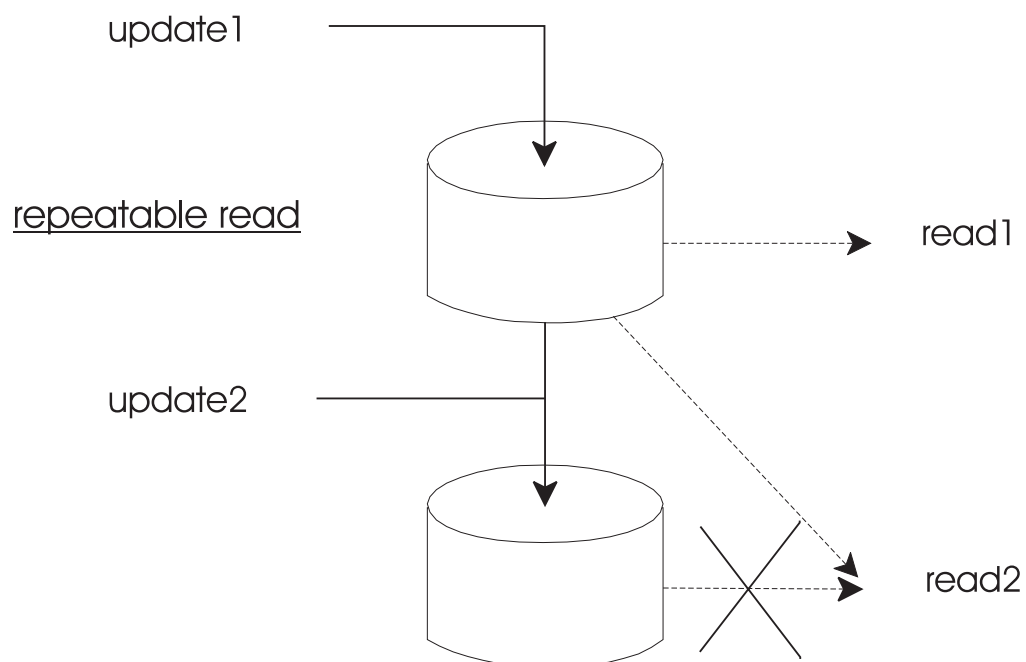
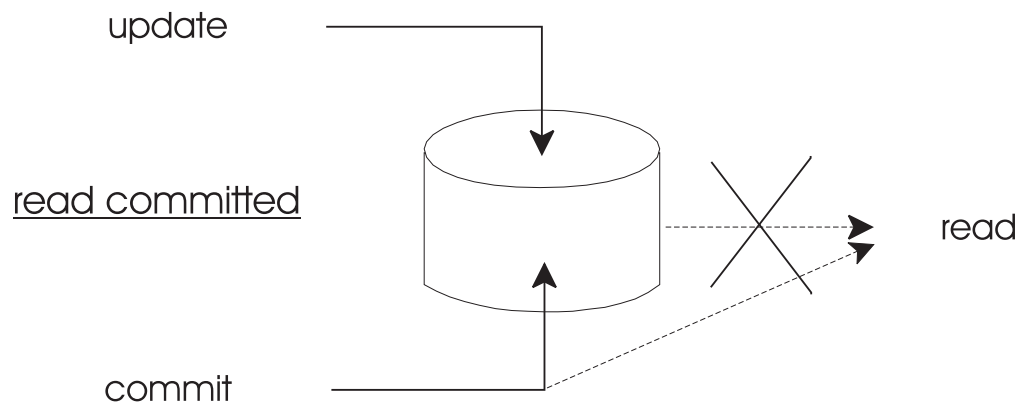
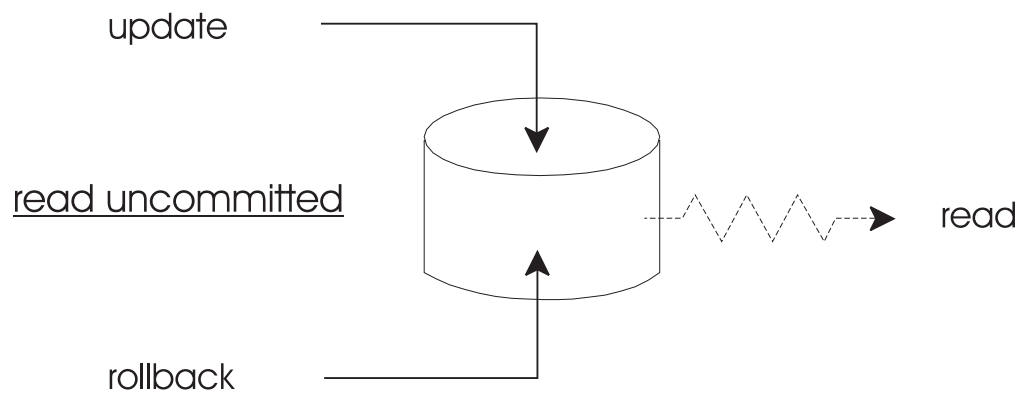
Transfer.lies (Kontonr\_1.Saldo)  
Transfer.schreibe (Kontonr\_1.Saldo)  
Abhebung.lies (Kontonr\_1.Saldo)  
Transfer.lies (Kontonr\_2.Saldo)  
Transfer.schreibe (Kontonr\_2.Saldo)  
Abhebung.schreibe (Kontonr\_1.Saldo)

## °Deadlocks (Verklemmungen) vermeiden



- 1) Transaktion **T1** sperrt den Satz **S1**.
- 2) Transaktion **T2** sperrt den Satz **S2**.
- 3) Transaktion **T1** beansprucht Satz **S2** auch. Sie versucht ihn zu sperren, muss aber abwarten, weil **S2** bereits von **T2** gesperrt ist.
- 4) Transaktion **T2** beansprucht Satz **S1** auch. Sie versucht ihn zu sperren, muss aber abwarten, weil **S1** bereits von **T1** gesperrt ist.

# °Isolierungsgrade



**Bild 9.6:** Isolierungsgrade



## °10 RDBMS mit ODBMS vergleichen

Kriterium	RDBMS	OODBMS
Schwerpunkt	administrativ	technisch
Datentypen	einfach, vordefiniert	komplex, benutzerdef.
Vererbung	-	Daten und Methoden
gleichz. Benutzer	viele	wenige (Arbeitsgruppe)
Werkzeuge	endbenutzerorientiert	programmiererorientiert
Transaktionen	i.d.R. kurz	oft lang
Prog.sprachen	3GL, 4GL	objektorientierte

**Tabelle 10.1:** Relationale und objektorientierte Datenbanksysteme

## °Daten strukturiert darstellen

<i>Konstruktor</i>	<i>Eigenschaften</i>	<i>Beispiel</i>	<i>Beispieloperation</i>
<b>Record</b>	feste Zahl von Elementen unterschiedlichen Typs	Anschrift	=
<b>Menge</b>	beliebig viele ungeordnete Unikate des gleichen Typs	Grundfarben	(Vereinigung)
<b>Liste</b>	beliebig viele geordnete Elemente (ev. Duplikate) des gleichen Typs	Adressliste	nächstes Element

**Tabelle 10.2:** Wichtige Typkonstruktoren in objektorientierten Datenmodellen

# °Daten / Operationen **objektorientiert** darstellen

**Objekttyp** HOCHSCHULANGEHÖRIGER

**Record aus** % geschachtelter Record

Versicherungsnummer: integer

Name: **Record aus**

Vorname: string

Nachname: string

Anschrift: **Record aus**

PLZ: integer

Ort: string

Strasse: string

Nummer: integer

Telefonnummern: **Menge aus** % siehe Tabelle 10.3

Telefon: string

**Objekttyp** STUDENT **ist** HOCHSCHULANGHÖRIGER % Vererbung

**Record aus**

Matrikelnummer: integer

...

Referent: DOZENT % Verweis auf benutzerdef. Typ

Matura: **Liste aus** % siehe Tabelle 10.3

Fach: string

Note: real

**Methoden**

Maturadurchschnitt: real % *statisch* gebunden

**Objekttyp** ASSISTENT **ist** HOCHSCHULANGEHÖRIGER % Vererbung

**Record aus** ...

Vorgesetzter: DOZENT % Verweis auf benutzerdef. Typ

Gehaltskonto: string

**Methoden**

Gehalt: real % *dynamisch* gebunden (polymorph)

**Objekttyp** HILFSASSISTENT **ist** STUDENT % Vererbung

**Record aus** ...

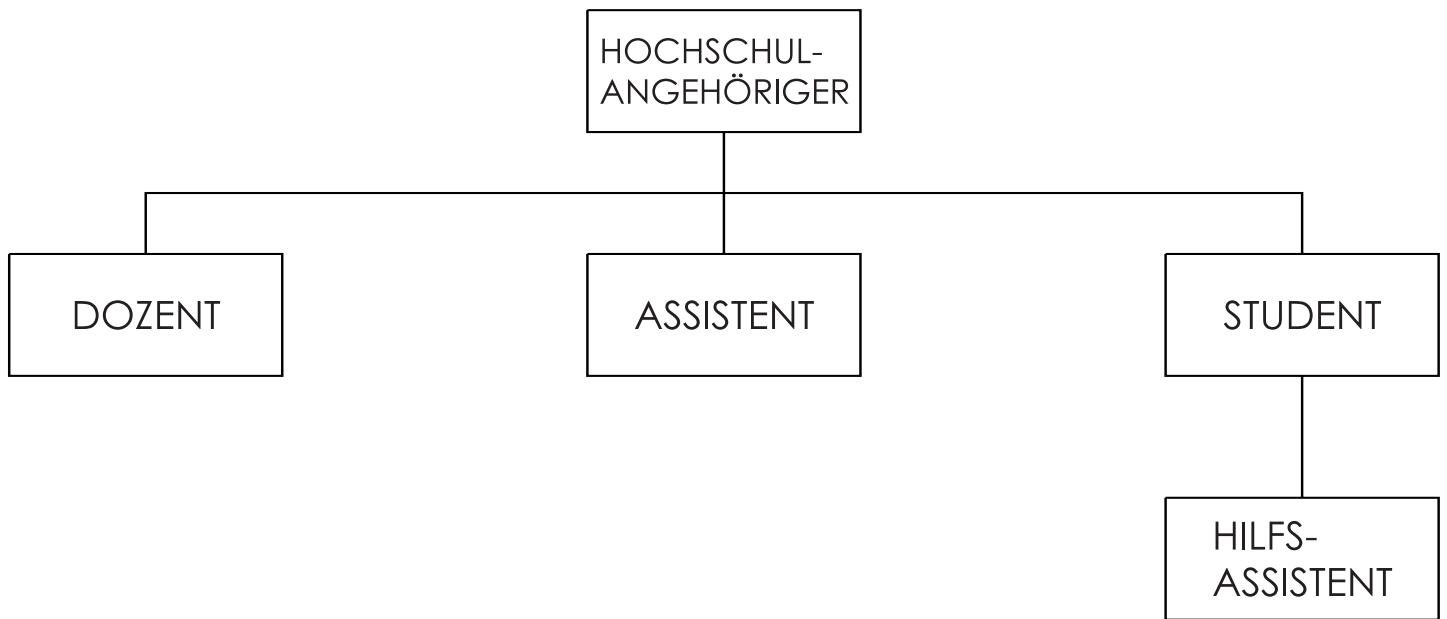
Gehaltskonto: string

**Methoden**

Gehalt: real % *dynamisch* gebunden (polymorph)

## Beispiel 10.3: Ausschnitt aus einem objektorientierten Datenmodell

## °Objekttypen hierarchisieren



**Bild 10.4:** Hierarchie zwischen den Objekttypen des Beispiels 10.2

## °11 Relationale DBMS bewerten

<i>Informationsquelle</i>	<i>Vorteile</i>	<i>Nachteile</i>
<i>Anbieterinformation</i>	Motivation des Informanten	Subjektivität
<i>Literaturanalyse</i>	Objektivität	Aktualität?
<i>Benutzerbefragung</i>	Objektivität	Froschperspektive Motivation des Informanten?
<i>Benchmarks</i>	Objektivität	Aufwand Repräsentativität?

**Vergleich 11.1:** Informationsquellen zur Systemauswahl

# °Relationale DBMS bewerten

<b>Oberkriterium</b>	<b>Kriterium</b>
<b>Modelltreue</b>	
	Sind alle Objekt- und Metadaten in Tabellenform und lassen sich darauf Tabellenoperationen ausführen? (Katalog)
	Lassen sich fehlende Werte einheitlich als Nullwerte darstellen?
	Unterstützt die Datenbanksprache die relationalen Grundoperationen?
	Abhängigkeit von Änderungen der physischen und logischen Datenorganisation? (zum Beispiel nach Optimierungen oder Änderungen von Basistabellen)
<b>Integrität</b>	
<b>deklarative -</b>	Welche Integritätsbedingungen lassen sich mit einer deklarativen Datenbeschreibungssprache im Katalog (Data Dictionary) festhalten?
	Entitäts- und Beziehungsintegrität automatisch ?
<b>prozedurale -</b>	Programmierung von Integritätsbedingungen als gespeicherte Prozeduren und Trigger ?
<b>Sprache</b>	
	Benutzerfreundlichere Art der deklarativen Datendefinition, -abfrage, -manipulation und -kontrolle als SQL ? (zum Beispiel QBE)
	Wie umfangreich sind DDL, DML und DCL?
	Zusammengesetzte Schlüssel definierbar ?
	Operationen auf Views ?
	Wieviele Tabellen und Schachtelungen unterstützten die Abfragesprachen?

<b>Oberkriterium</b>	<b>Kriterium</b>
	Für welche Objekte und Eigenschaften lassen sich Synonyme vergeben?
	Mehrere Versionen von Datenbankobjekten ? (engl. versioning)
<b>Entwicklung</b>	
<b>Werkzeug- umfang</b>	Autonome Entwicklung ganzer Anwendungen?
	Werden alle Entwicklungsphasen unterstützt? (insbesondere auch der Datenentwurf)
	Strukturierte Programmierung? (z.B. durch Steuerkonstrukte, vollständige Datentypen, lokale und globale Objekte)
	Objektorientierte Anwendungsentwicklung? (Kapselung, Vererbung und Polymorphie)
	Programmierungsumgebung ? (zum Beispiel Editor, Debugger, Browser und Versioning)
	Wie vollständig und programmierfreundlich lassen sich die Datenbankfunktionen aus externen 3GL-Sprachen aufrufen?
<b>Werkzeug- komfort</b>	Kann der Entwickler die Benutzerschnittstelle deklarativ gestalten? (Formular- und Berichtsgeneratoren)
	Lassen sich Standardanwendungen deklarativ entwickeln? (z.B. durch Applikationsgeneratoren)
<b>Kompatibilität</b>	
<b>Portabilität</b>	Welche Betriebssysteme und Hardwareware-Plattformen unterstützt das DBMS?
	Werden Multiprozessor-Systeme unterstützt?
	Lassen sich Schreiboperationen aus Sicherheitsgründen gleichzeitig auf mehreren Geräten speichern? (Plattenspiegelung)

<b>Oberkriterium</b>	<b>Kriterium</b>
<i>Kommunikation</i>	Anwendungsübergreifend ? (zum Beispiel über Import/Export, OLE oder DDE)
	Welche Datentypen unterstützt das DBMS? (zum Beispiel BLOBS, Memofelder, OLE)
<i>Standards</i>	Portabilität des SQL-Dialekts ?
	Wie nahe kommt die eingebaute prozedurale Programmiersprache an eine verbreitete 3GL- oder 4GL-Sprache?
<i>Datensicherheit</i>	
<i>Verfügbarkeit</i>	Ausfallsicherheit
<i>Backup</i>	Backup während des laufenden Betriebs ?
<i>Recovery</i>	Automatischer und manueller Recovery ?
	Komfort der Erstellung von Log-Dateien ?
<i>Transaktionssteuerung</i>	Welches Sperrprotokoll steuert den konkurrierenden Zugriff mehrerer Transaktionen auf gemeinsame Objekte ?
	Welche Isolierungsgrade werden unterstützt ?
	Welches ist die Granularität der Sperroperationen? (Datenbank, Tabelle, Page, Satz, Feld)
	Deadlocks automatisch erkannt und aufgelöst?
<i>Datenschutz</i>	
	Datenschutzmassnahmen (z.B. Passwortschutz, Verschlüsselung, Zugriffsprotokolle)
	Granularität der Datenschutzmassnahmen ? (System, Datenbank, Tabelle, Page, Satz, Feld)
	Hiding von Datenbankobjekten ?
	Lassen sich Benutzergruppen bilden?
	Lassen sich Berechtigungen vererben?
<i>Netzfähigkeit</i>	
	Welche Netzwerkprotokolle werden unterstützt?



<b>Oberkriterium</b>	<b>Kriterium</b>
	Verteilte Architekturen ? (Fileserver-, Client/Server-, verteilte Datenbanken)
	Erlaubt das DBMS heterogen verteilte Daten?
	Ist Replikation möglich?
	Qualität der Web-Einbindung? (z.B. E-Commerce)
<b>Physische Grenzen</b>	
	z.B. Speicherbedarf; Anzahl Netzknoten; Länge von Feld, Schlüssel, Satz, Namen, Datei, Datenbank und Programm?
<b>Effizienz</b>	
<b>Speicher-</b>	Wie flexibel lässt sich der Cache (Datenpuffer) verwalten?
	Lassen sich Sätze physisch zusammenfassen und als Cluster speichern?
	Lassen sich die Daten komprimieren?
<b>Laufzeit-</b>	Wie kann der Zugriff optimiert werden? (zum Beispiel durch B-Bäume und Hashfunktionen)
	Werden Suchabfragen automatisch optimiert?
	Erzeugt der Übersetzer der prozeduralen Sprache Maschinen- oder Zwischencode?
	Multiprozessorfähigkeit?
<b>Komfort</b>	
	Bietet das System vollständige Einführungs- und Referenzhilfen an? (on line, schriftlich, Schulung)
	Wie gut ist die laufende Unterstützung durch den Anbieter? (technischer Support)
	Existieren Benutzervereinigungen?
<b>Anbieter- und Produktqualität</b>	

<b>Oberkriterium</b>	<b>Kriterium</b>
	Produktumstände wie Reifegrad oder Häufigkeit und Aufwand von Versionswechseln ?
	Herstellerfirma ? (zum Beispiel ihre wirtschaftliche Lage, die lokale und globale Marktpräsenz und die eingeholten Referenzen)
	In welches Abhängigkeitsverhältnis gegenüber dem Anbieter begeben wir uns?

Web Quiz

Musterklausur, 2. Teil