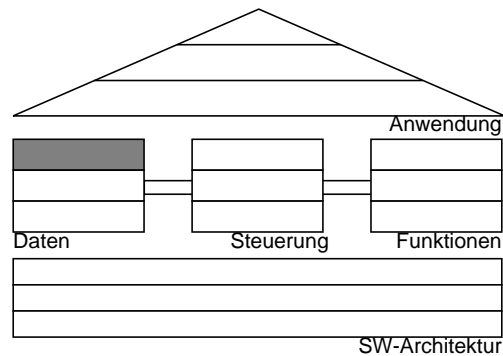


# 3. Informationsmodelle

- **GBIS-Rahmen: Einordnung**



- **Vorgehensweise bei DB-Entwurf und -Modellierung**

- Lebenszyklus
- Informationserhebung

- **Entity-Relationship-Modell (ERM)**

- Definitionen, Konzepte
- Beziehungstypen
- Diagrammdarstellung
- Beispiele

- **Erweiterungen des ERM**

- Kardinalitätsrestriktionen
- Abstraktionskonzepte

- **Abstraktionskonzepte**

- Klassifikation/Instantiierung
- Generalisierung/Spezialisierung
- Element-/Mengen-Assoziation
- Element-/Komponenten-Aggregation

# Vorgehensweise bei DB-Entwurf und -Modellierung

- **Ziel: Modellierung einer Miniwelt**

  - (Entwurf von Datenbankschemata)

  - modellhafte Abbildung eines anwendungsorientierten Ausschnitts der realen Welt (Miniwelt)
  - Nachbildung von Vorgängen durch **Transaktionen**

- **Nebenbedingungen:**

  - genaue Abbildung
  - hoher Grad an Aktualität
  - Verständlichkeit, Natürlichkeit, Einfachheit, ...

- **Zwischenziel:**

  - Erhebung der Information in der Systemanalyse (Informationsbedarf !)
  - **Informationsmodell** (allgem. Systemmodell)

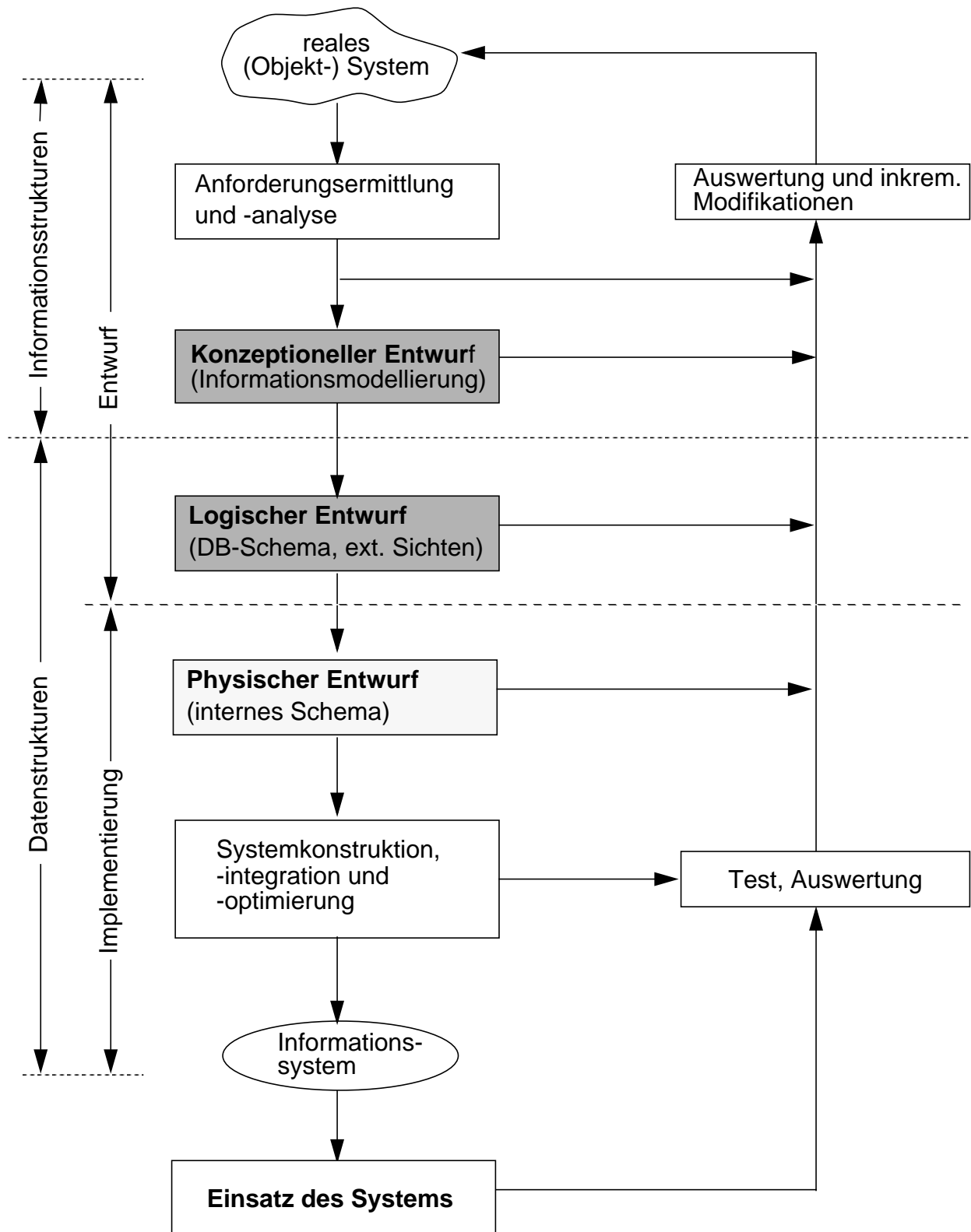
- **Bestandteile:**

  - Objekte: Entities
  - Beziehungen: Relationships

- **Schrittweise Ableitung: (Verschiedene Sichten)**

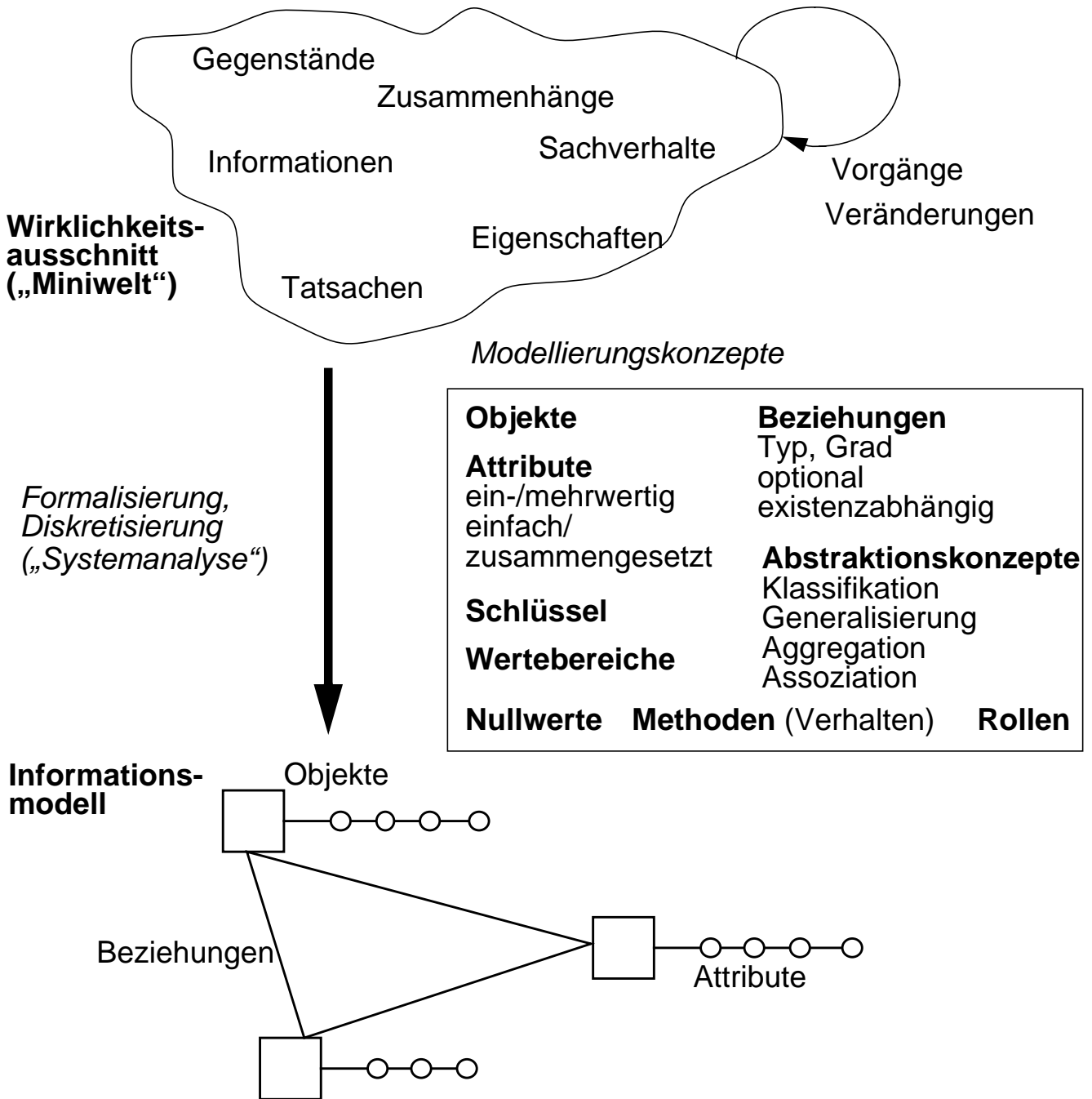
1. Information in unserer Vorstellung
2. Informationsstruktur: Organisationsform der Information
3. Logische Datenstruktur (zugriffspfadunabhängig, Was-Aspekt)
4. Physische Datenstruktur (zugriffspfadabhängig, Was- und Wie-Aspekt)

# Schritte auf dem Weg zu einem Informationssystem



**Bemerkung:** Anforderungsermittlung und -analyse sind kaum systematisiert;  
Methoden: „Befragen“, „Studieren“, „Mitmachen“

# Informationsmodelle



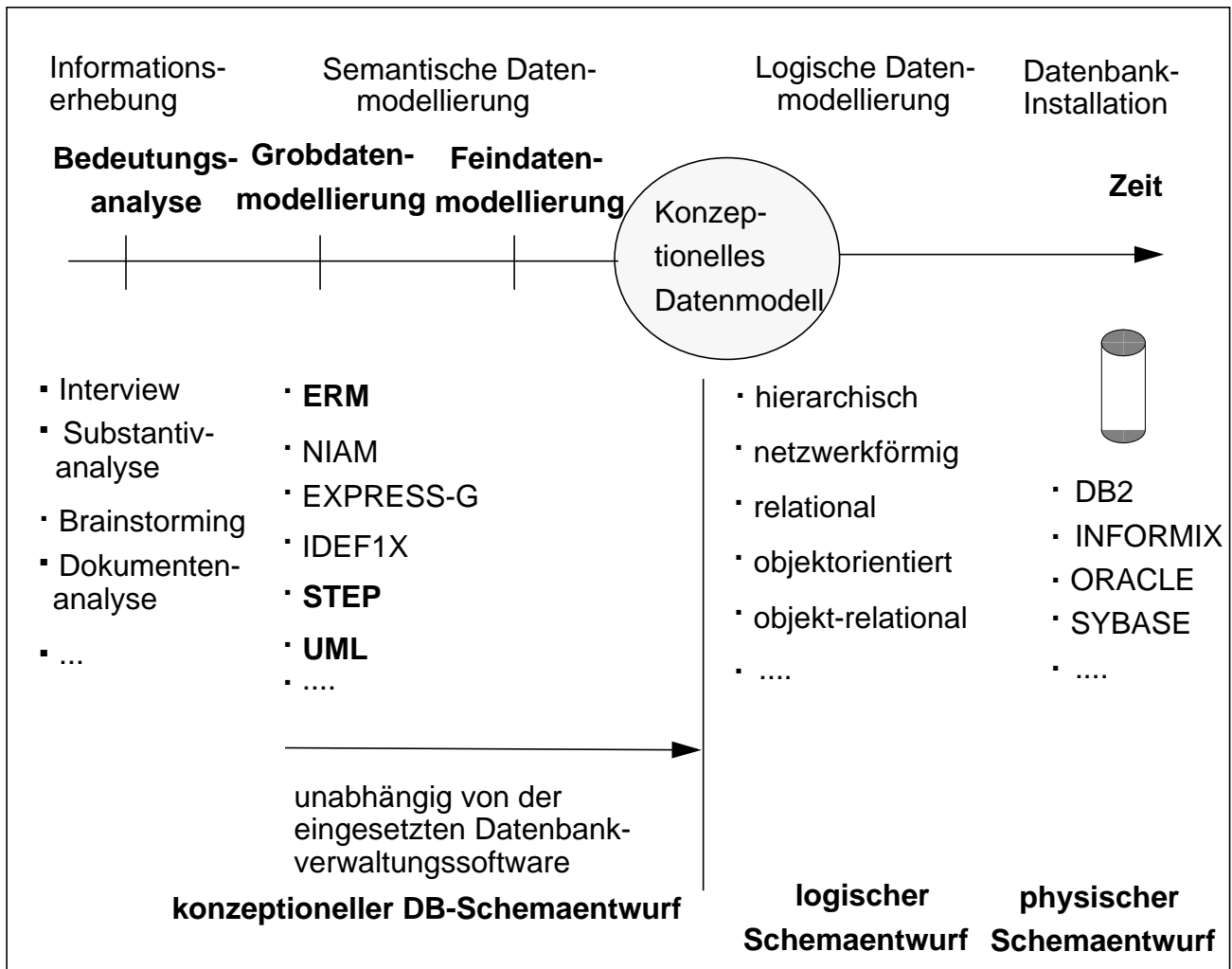
→ **Informationsmodell** (Darstellungselemente & Regeln):  
eine Art formale Sprache, um Informationen zu beschreiben

• **Informationen über Objekte und Beziehungen nur, wenn:**

- unterscheidbar und identifizierbar
- relevant
- selektiv beschreibbar

# Von der Informationserhebung zum DB-Schema

## • Prinzipielle Vorgehensweise



## • **ERM (Entity Relationship Model):**

- generell einsetzbares Modellierungswerkzeug

## • **STEP (STandard for the Exchange of Product Definition Data):**

- Modellierung, Zugriff, Austausch von **produktdefinierenden Daten** über den gesamten Produktlebenszyklus

## • **UML (Unified Modeling Language):**

- Notation und Sprache zur Unterstützung objektorientierter Softwareentwicklung

# Entity-Relationship-Modell (ERM) –<sup>1</sup>

## Überblick

- **Modellierungskonzepte**

- Entity-Mengen (Objektmengen)
- Wertebereiche, Attribute
- Primärschlüssel
- Relationship-Mengen (Beziehungsmengen)

- **Klassifikation der Beziehungstypen**

- benutzerdefinierte Beziehungen
- Abbildungstyp
  - 1 : 1
  - n : 1
  - n : m
- **Ziel:**
  - Festlegung von semantischen Aspekten
  - explizite Definition von strukturellen Integritätsbedingungen

- **Achtung**

Das ERM modelliert die Typ-, nicht die Instanzenebene; es macht also Aussagen über Entity- und Relationship-Mengen, nicht jedoch über einzelne ihrer Elemente (Ausprägungen). Die Modellierungskonzepte des ERM sind häufig zu ungenau oder unvollständig. Sie müssen deshalb ergänzt werden durch Integritätsbedingungen oder Constraints

---

1. Chen, P. P.-S.: The Entity-Relationship Model —Toward a Unified view of Data, in: ACM TODS 1:1, March 1976, pp. 9-36.

# Konzepte des ERM

- **Entities**

- wohlunterscheidbare Dinge der Miniwelt (Diskurswelt)
- „A thing that has real or individual existence in reality or in mind“ (Webster)
- besitzen Eigenschaften, deren konkrete Ausprägungen als Werte bezeichnet werden

- **Entity-Mengen (Entity-Sets)**

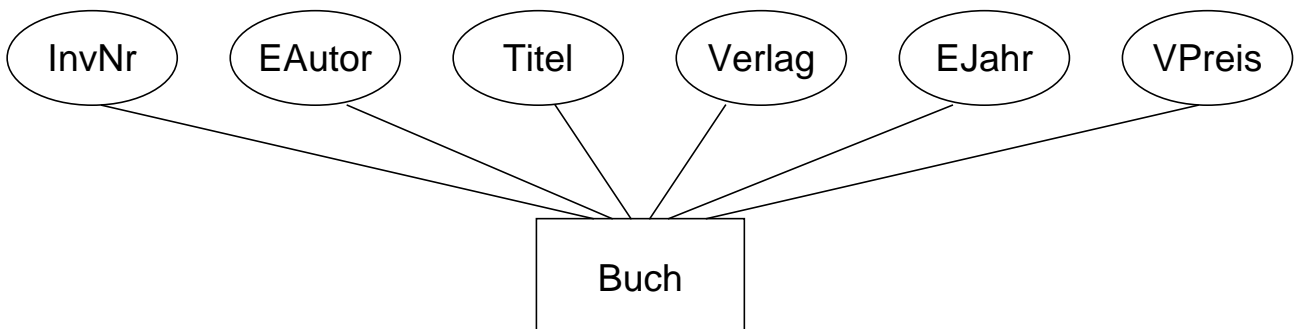
- Zusammenfassung von „ähnlichen“ oder „vergleichbaren“ Entities
- haben gemeinsame Eigenschaften
- Beispiele:
  - Abteilungen, Angestellte, Projekte, ...
  - Bücher, Autoren, Leser, ...
  - Studenten, Professoren, Vorlesungen, ...
  - Kunden, Vertreter, Wein, Behälter, ...

- **Wertebereiche und Attribute**

- Die möglichen oder „zulässigen“ Werte für eine Eigenschaft nennen wir Wertebereich (oder Domain)
- Die (bei allen Entities einer Entity-Menge auftretenden) Eigenschaften werden als Attribute bezeichnet
- Ein Attribut ordnet jedem Entity einer Entity-Menge einen Wert aus einem bestimmten Wertebereich (dem des Attributs) zu

## Konzepte des ERM (2)

- Entity-Typ Buch (in Diagrammdarstellung)



Attribut	Wertebereich
InvNr	
EAutor	
⋮	
EJahr	
VPreis	

→ Name der Entity-Menge sowie zugehörige Attribute sind **zeitinvariant**

- Entity-Menge und ihre Entities sind zeitveränderlich

$e_1 = (4711, \text{Kemper, DBS, Oldenbourg, ...})$

$e_2 = (0815, \text{Date, Intro. to DBS, Addison, ...})$

$e_3 = (1234, \text{Härder, DBS, Springer, ...})$

→ Alle Attribute sind einwertig!



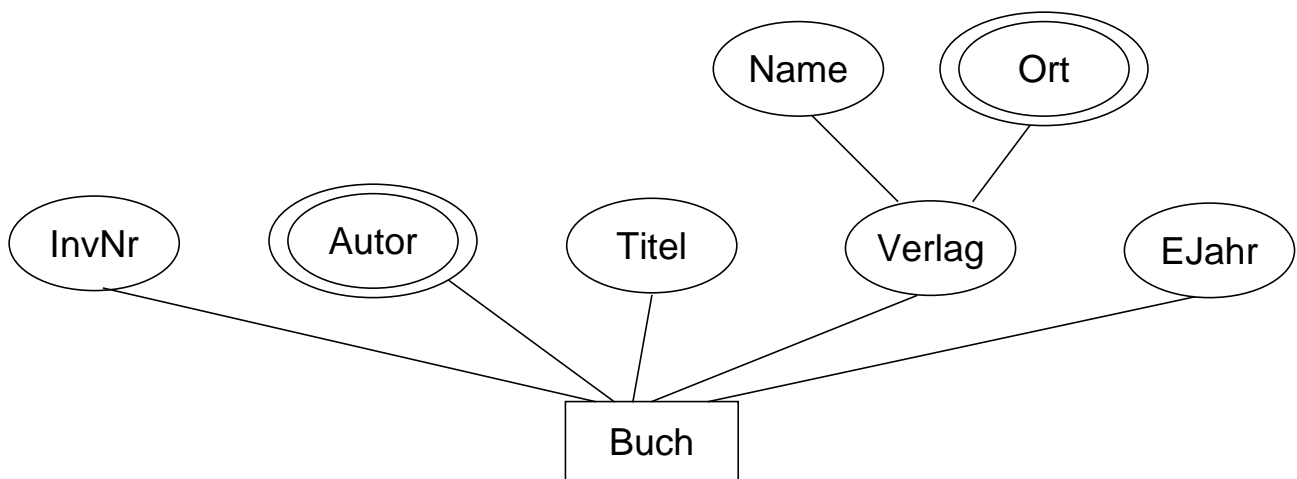
## Konzepte des ERM (3)

- **Wie wird modelliert, wenn**

- ein Buch mehrere Autoren hat
- die Verlagsinformation zusammengesetzt ist (Name, Ort)
- Eigenschaften hierarchisch gegliedert sind

- **Erhöhung der Modellierungsgenauigkeit**

- einwertige Attribute
- mehrwertige Attribute (Doppelovale)
- zusammengesetzte Attribute (hierarchisch angeordnete Ovale)  
→ Verschachtelungen sind möglich



$e_3 =$

## Konzepte des ERM (4)

- **Wie wird ein Entity identifiziert?**

- Entities müssen „wohlunterscheidbar“ sein
- Information über ein Entity **ausschließlich** durch (Attribut-) Werte

- **Identifikation** eines Entities durch Attribut (oder Kombination von Attributen)

- (1:1) - Beziehung
- ggf. künstlich erzwungen (lfd. Nr.)

- $\{A_1, A_2, \dots, A_m\} = \mathbf{A}$  sei Menge der (einwertigen) Attribute zur Entity-Menge E

$\mathbf{K} \subseteq \mathbf{A}$  heißt Schlüsselkandidat von E

$\Leftrightarrow \mathbf{K}$  irreduzibel;  $e_i, e_j \in E$  ;

$e_i \neq e_j \rightarrow \mathbf{K}(e_i) \neq \mathbf{K}(e_j)$

- Mehrere Schlüsselkandidaten (SK) möglich

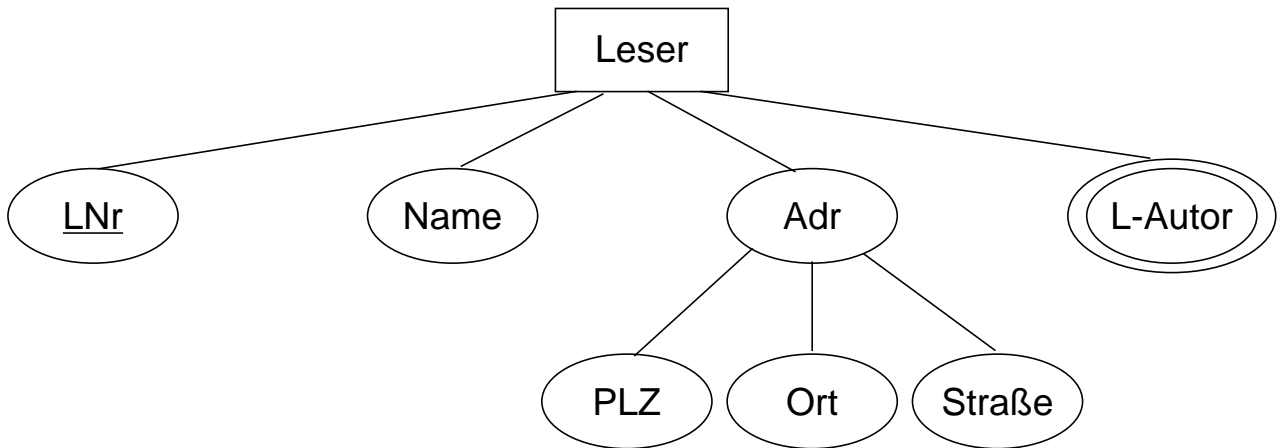
→ **Primärschlüssel** auswählen

- **Beispiel:**

Entity-Menge **Student** mit Attributen Matrnr, SVNr, Name, Gebdat, FBNr

## Konzepte des ERM (5)

- Entity-Deklaration oder **Entity-Typ** legt die **zeitinvarianten Aspekte** von Entites fest
- **Entity-Diagramm**



- **Entity-Typ  $E = (X, K)$**

Leser = ( {LNr, Name, Adr (PLZ, Ort, Straße), {L-Autor} }, {LNr} )

- **Wertebereiche**

$W(\text{LNr}) = \text{int}(8), \quad W(\text{Name}) = W(\text{L-Autor}) = \text{char}(30)$

$W(\text{PLZ}) = \text{int}(5), \quad W(\text{Ort}) = \text{char}(20), \quad W(\text{Straße}) = \text{char}(15)$

$\text{dom}(\text{Adr}) = W(\text{PLZ}) \times W(\text{Ort}) \times W(\text{Straße}) = \text{int}(5) \times \text{char}(20) \times \text{char}(15)$

$\text{dom}(\text{L-Autor}) = 2^{W(\text{L-Autor})} = 2^{\text{char}(30)}$

- **Zusammensetzung**  $A(B(C_1, C_2), \{D(E_1, E_2)\})$

mit  $W(C_1), W(C_2), W(E_1), W(E_2)$

$\text{dom}(B) = W(C_1) \times W(C_2)$

$\text{dom}(D) = 2^{W(E_1) \times W(E_2)}$

$\text{dom}(A) = \text{dom}(B) \times \text{dom}(D)$

# ERM - Definitionen<sup>1</sup>

- **Def. 1: Entity-Typ**

Ein Entity-Typ hat die Form  $E = (X, K)$  mit einem Namen  $E$ , einem Format  $X$  und einem Primärschlüssel  $K$ , der aus (einwertigen) Elementen von  $X$  besteht. Die Elemente eines Formats  $X$  werden dabei wie folgt beschrieben:

- i) Einwertige Attribute :  $A$
- ii) Mehrwertige Attribute:  $\{A\}$
- iii) Zusammengesetzte Attribute:  $A (B_1, \dots, B_k)$

- **Def. 2: Wertebereich (Domain)**

$E = (X, K)$  sei ein Entity-Typ und  $\text{attr}(E)$  die Menge aller in  $X$  vorkommenden Attributnamen. Jedem  $A \in \text{attr}(E)$ , das nicht einer Zusammensetzung voransteht, sei ein Wertebereich  $W(A)$  zugeordnet. Für jedes  $A \in \text{attr}(E)$  sei

$$\text{dom}(A) := \begin{cases} W(A) & \text{falls } A \text{ einwertig} \\ 2^{W(A)} \text{ oder } P(W(A)) & \text{falls } A \text{ mehrwertig} \\ W(B_1) \times \dots \times W(B_k) & \text{falls } A \text{ aus einwertigen} \\ & B_1, \dots, B_k \text{ zusammengesetzt} \end{cases}$$

Besteht  $A$  aus mehrwertigen oder zusammengesetzten Attributen, wird die Definition rekursiv angewendet.

- **Bem.**

Das Format  $X$  eines Entity-Typs kann formal als Menge oder als Folge dargestellt werden. Die Schreibweise als Folge ist einfacher; die Folge kann bei der Diagrammdarstellung übernommen werden.

---

1. G. Vossen: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, Oldenbourg, 4. Auflage, 2000

## ERM - Definitionen (2)

- **Def. 3: Entity und Entity-Menge**

Es sei  $E = (X, K)$  ein Entity-Typ mit  $X = (A_1, \dots, A_m)$ .  $A_i$  sei  $\text{dom}(A_i)$  ( $1 \leq i \leq m$ ) zugeordnet.

- Ein Entity  $e$  ist ein Element des Kartesischen Produkts aller Domains, d.h.  
$$e \in \text{dom}(A_1) \times \dots \times \text{dom}(A_m)$$
- Eine Entity-Menge  $E^t$  (zum Zeitpunkt  $t$ ) ist eine Menge von Entities, welche  $K$  erfüllt, d.h.

$$E^t \subseteq \text{dom}(A_1) \times \dots \times \text{dom}(A_m)$$

$E^t$  wird auch als der Inhalt bzw. der aktuelle Wert (Instanz) des Typs  $E$  zur Zeit  $t$  bezeichnet.

- **Def. 4: Relationships**

- Ein Relationship-Typ hat die Form  $R = (\text{Ent}, Y)$ . Dabei ist  $R$  der Name des Typs (auch „Name der Beziehung“),  $\text{Ent}$  bezeichnet die Folge der Namen der Entity-Typen, zwischen denen die Beziehung definiert ist, und  $Y$  ist eine (möglicherweise leere) Folge von Attributen der Beziehung.

- Sei  $\text{Ent} = (E_1, \dots, E_k)$ , und für beliebiges, aber festes  $t$  sei  $E_i^t$  der Inhalt des Entity-Typs  $E_i$ ,  $1 \leq i \leq k$ . Ferner sei  $Y = (B_1, \dots, B_n)$ . Eine Relationship  $r$  ist ein Element des Kartesischen Produktes aus allen  $E_i^t$  und den Domains der  $B_j$ , d.h.

$$r \in E_1^t \times \dots \times E_k^t \times \text{dom}(B_1) \times \dots \times \text{dom}(B_n)$$

bzw.

$$r = (e_1, \dots, e_k, b_1, \dots, b_n)$$

mit

$$e_i \in E_i^t \text{ für } 1 \leq i \leq k \text{ und } b_j \in \text{dom}(B_j) \text{ für } 1 \leq j \leq n.$$

- Eine Relationship-Menge  $R^t$  (zur Zeit  $t$ ) ist eine Menge von Relationships, d.h.,

$$R^t \subseteq E_1^t \times \dots \times E_k^t \times \text{dom}(B_1) \times \dots \times \text{dom}(B_n).$$

## Konzepte des ERM (6)

- **Relationship-Mengen**

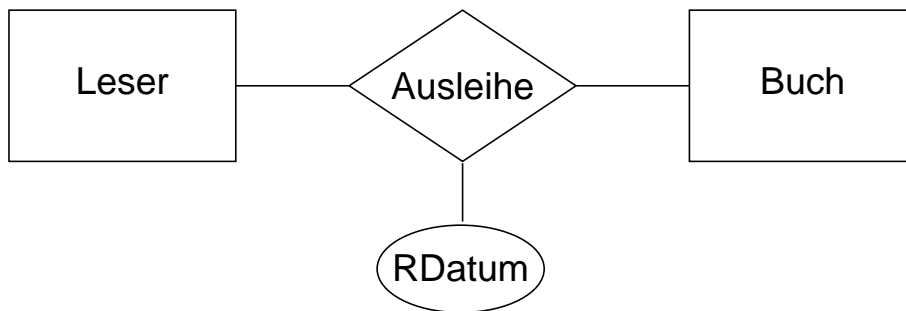
Zusammenfassung von gleichartigen Beziehungen (Relationships) zwischen Entities, die jeweils gleichen Entity-Mengen angehören

z. B. „hat ausgeliehen“ zwischen „Leser“ und „Buch“

- **Eigenschaften**

- Grad  $n$  der Beziehung (*degree*), gewöhnlich  $n=2$  oder  $n=3$
- Existenzabhängigkeit
- Beziehungstyp (*connectivity*)
- Kardinalität

- **ER-Diagramm**



- **Relationship-Typ**  $R = (\text{Ent}, Y)$

- **Eigenschaften**

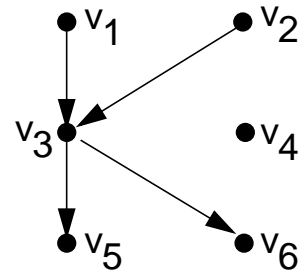
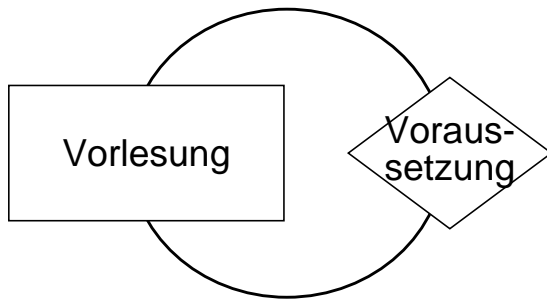
Grad:

Existenzabhängig:

Beziehungstyp:

# Relationship-Mengen

- **Motivation für Rollennamen**



- **Definition:**

Voraussetzung = ((Vorlesung, Vorlesung), ( $\emptyset$ ))

d. h.  $Voraussetzung^t = \{ (v_i, v_j) \mid v_i, v_j \in \text{Vorlesung} \}$

genauer: direkte Voraussetzung

- Einführung von Rollennamen (rn) möglich (Reihenfolge!)  
auf Typebene: (rn<sub>1</sub>/E, rn<sub>2</sub>/E) oder (Vorgänger/Vorlesung, Nachfolger/Vorlesung)

auf Instanzebene: (Vorgänger/v<sub>i</sub>, Nachfolger/v<sub>j</sub>)

Sprechweise: „v<sub>j</sub> setzt v<sub>i</sub> voraus“

oder „v<sub>i</sub> ist-Voraussetzung-für v<sub>j</sub>“

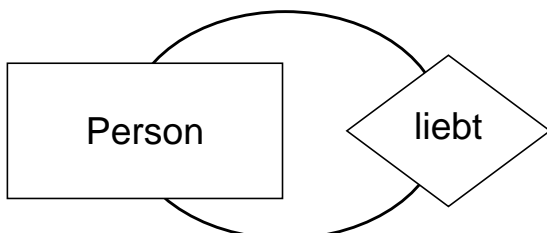
- **Eigenschaften:**

Grad:

Existenzabhängig:

Beziehungstyp:

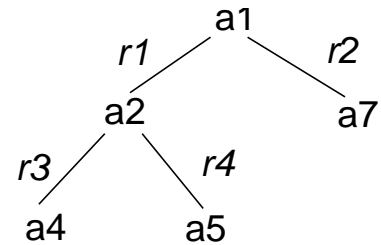
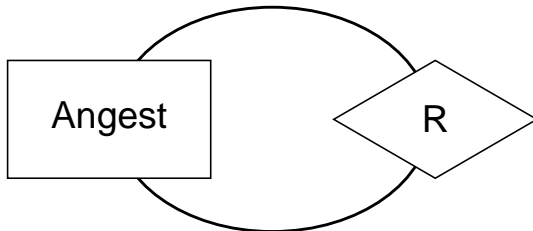
- **Transitivität gilt bei Selbstreferenz i. allg. nicht!**



## Relationship-Mengen (2)

- Keine Disjunktheit der Entity-Mengen gefordert, die an einer  $R_i$  beteiligt sind

**Direkter-Vorgesetzter** = ((Angest/Angest, Chef/Angest), ( $\emptyset$ ))



- Eigenschaften**

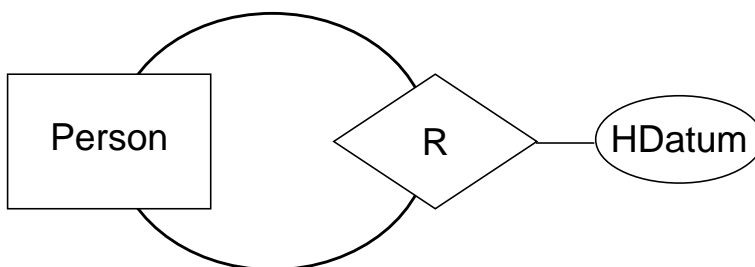
Grad:

Existenzabhängig:

Beziehungstyp:

- R sei Direkter-Vorgesetzter. Welche Beziehungen auf Angestellter sind zulässig?

- Relationship-Menge **Heirat** = ((Mann/Person, Frau/Person), (HDatum))



- Eigenschaften**

Grad:

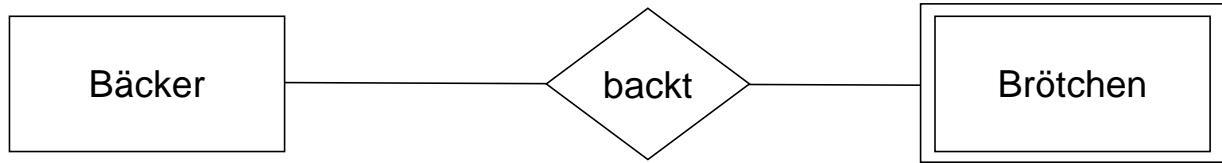
Existenzabhängig:

Beziehungstyp:

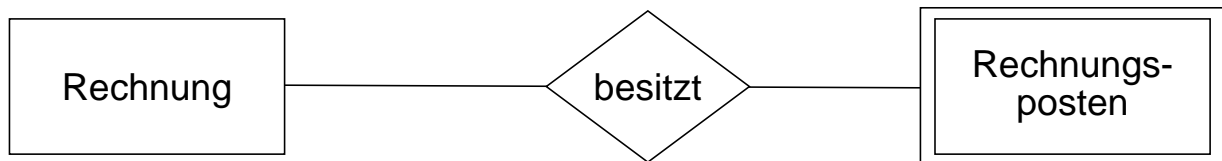
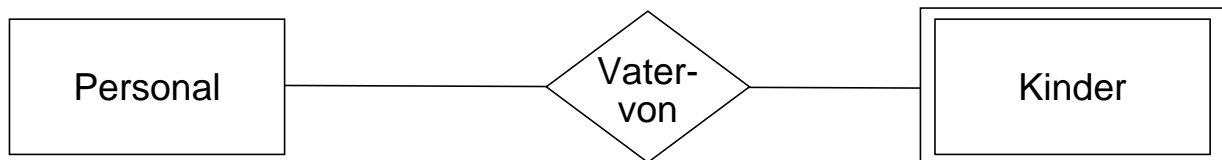
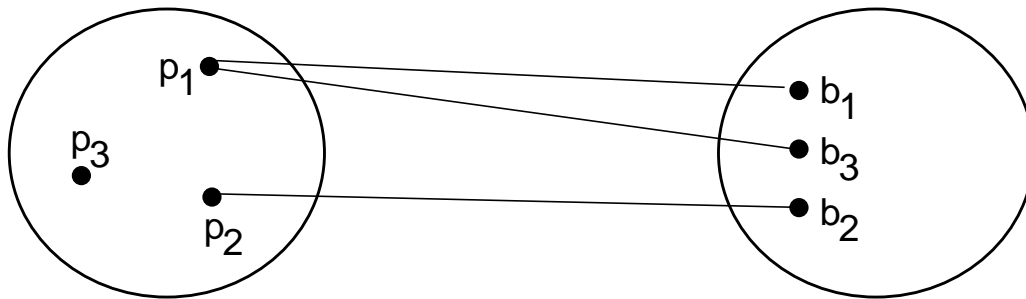


## Relationship-Mengen (3)

- Existenzabhängigkeit einer Entity-Menge
- Beispiele



Existenzabhängigkeit: „Relationship begründet Existenz von“

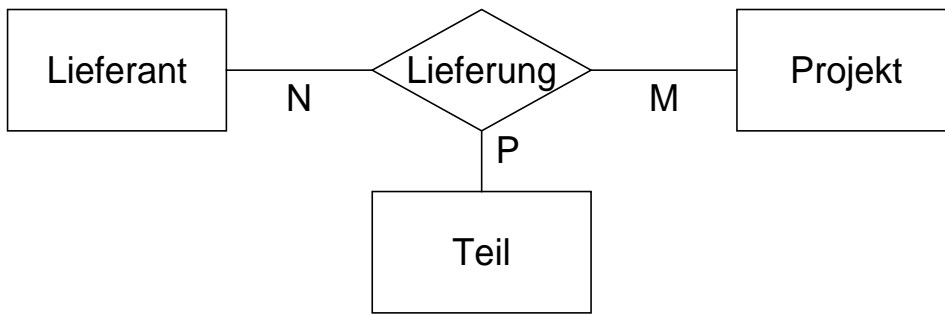


- Eigenschaften

Grad:	2
Existenzabhängig:	ja
Beziehungstyp:	1 : n

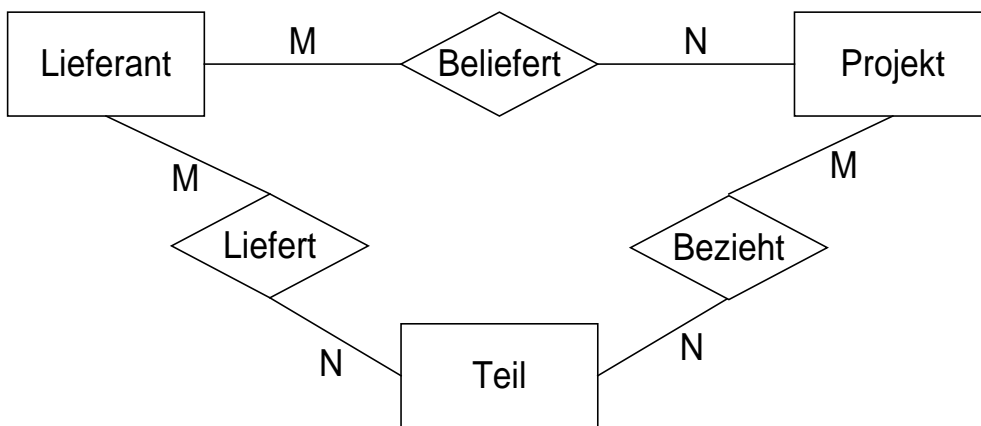
- **Bem.:** In manchen Modellen steht eine existenzabhängige Entity-Menge rechts von der selbständigen Entity-Menge und der „erzeugenden“ Relationship-Menge. Bei Mehrfachreferenzen ist eine „erzeugende“ von weiteren „referenzierenden“ Relationship-Mengen zu unterscheiden.

## Dreistellige Relationship-Mengen



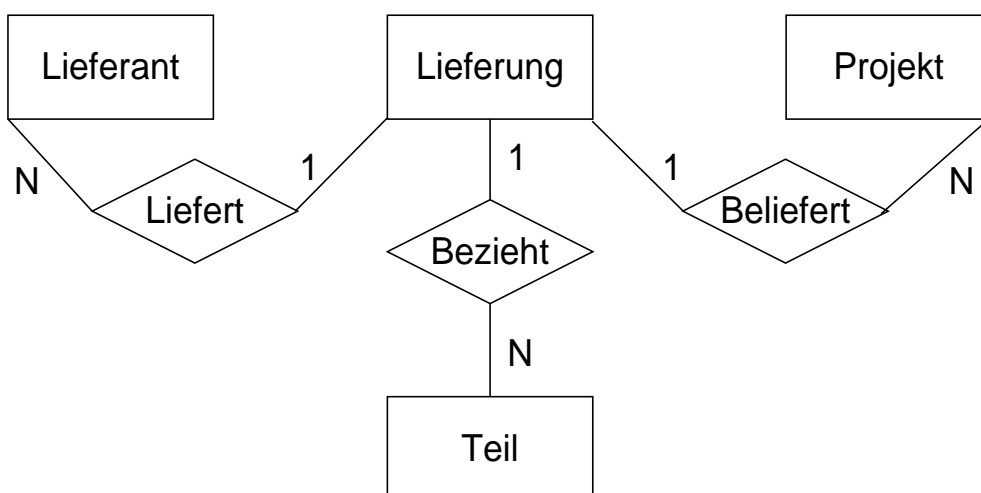
### Achtung:

Nicht gleichwertig mit drei zweistelligen (binären) Relationship-Mengen!



### Aber:

Manche Systeme erlauben nur die Modellierung binärer Relationship-Mengen!



# Klassifikation von Datenabbildungen

- **ZIEL:**

- Festlegung von semantischen Aspekten (hier: Beziehungstyp)
- explizite Definition von strukturellen Integritätsbedingungen

- **Unterscheidung von Beziehungstypen**

- $E_i - E_j$
- $E_i - E_i$

- **Festlegung der Abbildungstypen**

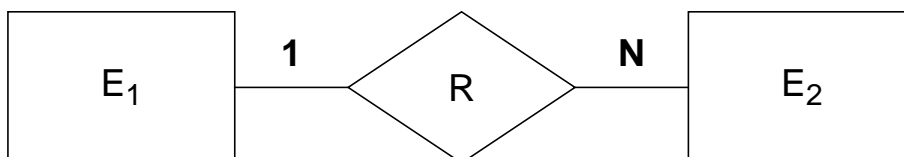
- 1:1 ... eindeutige Funktion (injektive Abbildung)
- n:1 ... math. Funktion (funktionale oder invers funktionale Abbildung)
- n:m ... math. Relation (komplexe Abbildung)

- **Beispiele zu  $E_i - E_j$**

- 1:1 ... LEITET/WIRD\_GELEITET:      PROF  $\leftrightarrow$  ARBGRUPPE
- 1:n ... ARBEITET\_FÜR/MIT:      MITARBEITER  $\rightarrow$  PROF
- n:m ... BESCHÄFTIGT/IST\_HIWI:      PROF — STUDENT

→ Abbildungstypen implizieren nicht, daß für jedes  $e_k \in E_i$  auch tatsächlich ein  $e_l \in E_j$  existiert

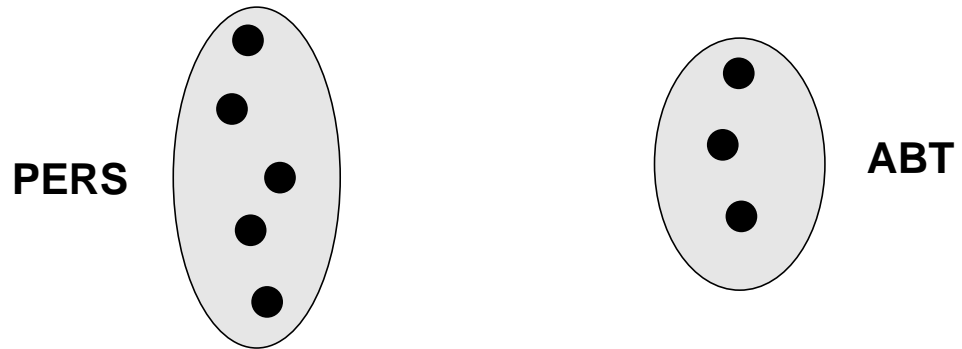
- **Diagrammdarstellung:**



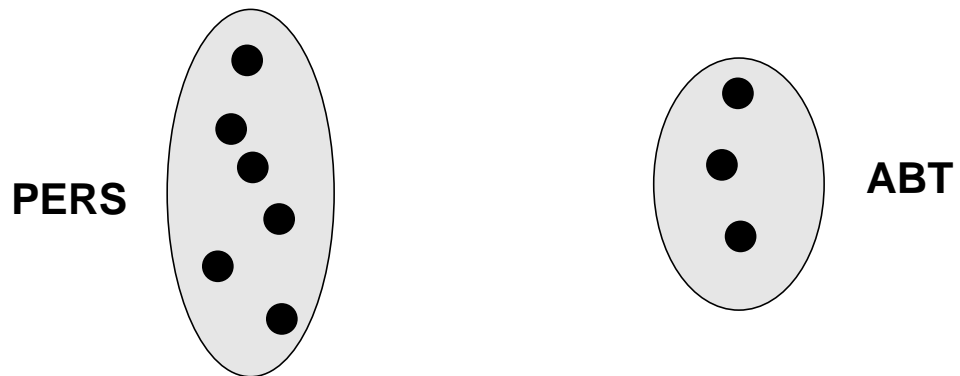
## Klassifikation von Datenabbildungen (2)

- Beispiele zu  $E_i - E_j$  (externe Klassifikation)

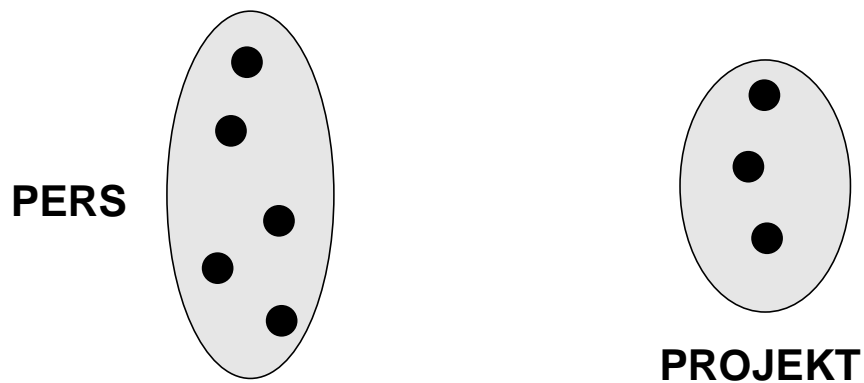
- 1:1: LEITET/WIRD\_GELEITET: PERS  $\leftrightarrow$  ABT



- n:1/1:n: ARBEITET\_FÜR/HAT\_MITARBEITER: PERS  $\rightarrow$  ABT



- n:m: ARBEITET\_FÜR/MITARBEIT: PERS — PROJEKT



# ER-Schema – Beispiel

<u>DECLARE</u>	<u>VALUE-SETS</u>	<u>REPRESENTATION</u>	<u>ALLOWABLE-VALUES</u>
	PERSONAL-NR	INTEGER(5)	(1,10000)
	VORNAMEN	CHARACTER(15)	ALL
	NACHNAMEN	CHARACTER(25)	ALL
	BERUFE	CHARACTER(25)	ALL
	PROJEKT-NR	INTEGER(3)	(1,5000)
	ANZ.-JAHRE	INTEGER(3)	(0,100)
	ORTE	CHARACTER(15)	ALL
	PROZENT	FIXED(5.2)	(0,100.00)
	ANZ.-MONATE	INTEGER(3)	(0,100)

DECLARE REGULAR ENTITY RELATION PERSONAL  
ATTRIBUTE/VALUE-SET:  
PNR/PERSONAL-NR  
NAME/(VORNAMEN,NACHNAMEN)  
KÜNSTLER-NAME/(VORNAMEN, NACHNAMEN)  
BERUF/BERUFE  
ALTER/ANZ.-JAHRE  
PRIMARY KEY:  
PNR

DECLARE REGULAR ENTITY RELATION PROJEKT  
ATTRIBUTE/VALUE-SET:  
PRO-NR/PROJEKT-NR  
PRO-ORT/ORTE  
PRIMARY KEY:  
PRO-NR

DECLARE RELATIONSHIP RELATION PROJEKT-MITARBEIT  
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES  
MITARBEITER/PERSONAL.PK/n  
PROJEKT /PROJEKT.PK/m  
ATTRIBUTE/VALUE-SET:  
ARBEITSZEITANTEIL/PROZENT  
DAUER/ANZ.-MONATE

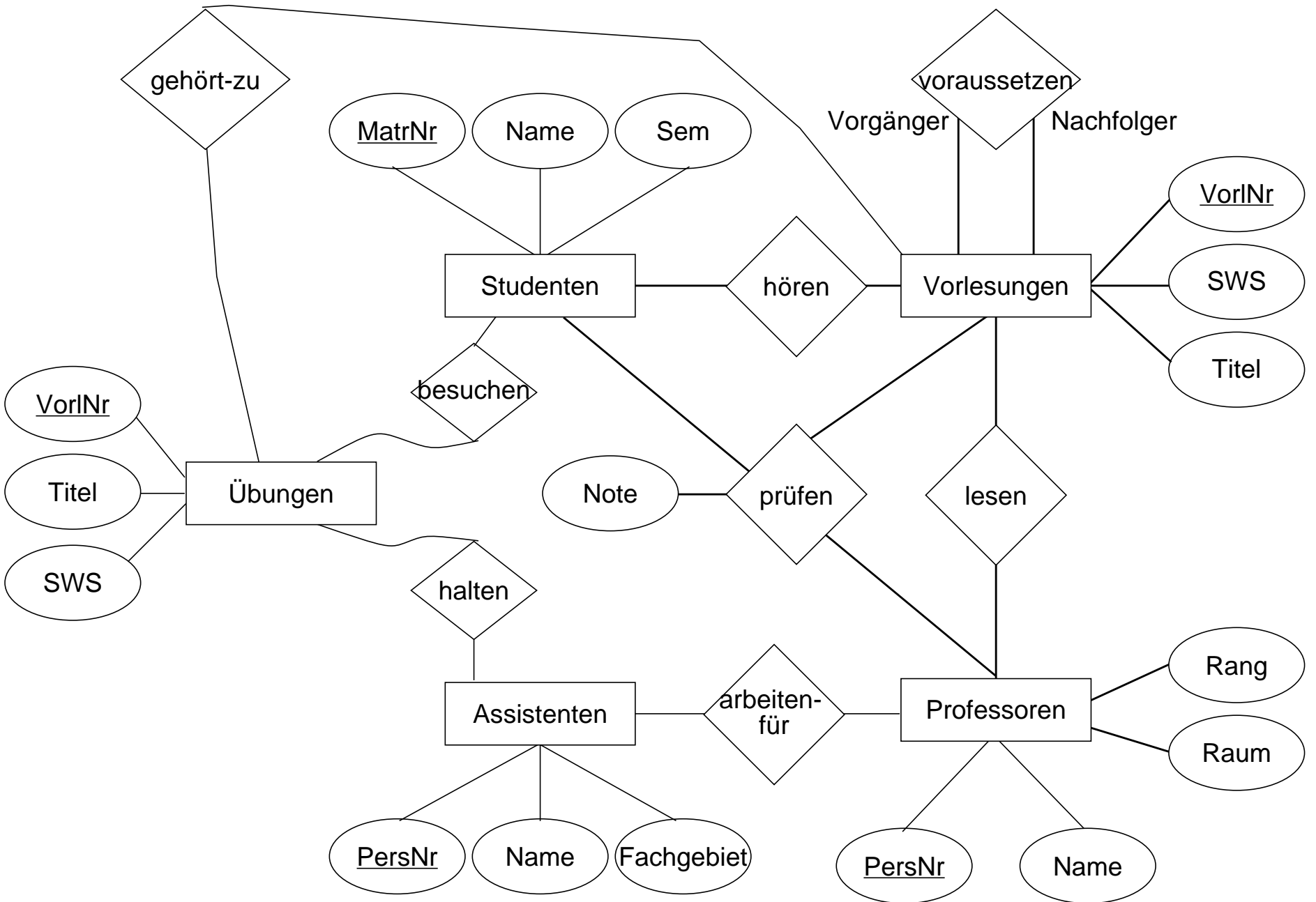
DECLARE RELATIONSHIP RELATION PERS.-ANGEHÖRIGE  
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES  
UNTERHALTSPFLICHTIGER/PERSONAL.PK/1  
KIND/ KINDER.PK/n  
EXISTENCE OF KIND DEPENDS ON  
EXISTENCE OF UNTERHALTSPFLICHTIGER

DECLARE WEAK ENTITY RELATION KINDER ATTRIBUTE/VALUE-SET:  
NAME/VORNAMEN  
ALTER/ANZ.-JAHRE  
PRIMARY KEY:  
NAME  
PERSONAL.PK THROUGH PERS-ANGEHÖRIGE

## Anwendungsbeispiel: Vorlesungsbetrieb

Stellen Sie ein ER-Diagramm für folgende Miniwelt auf:

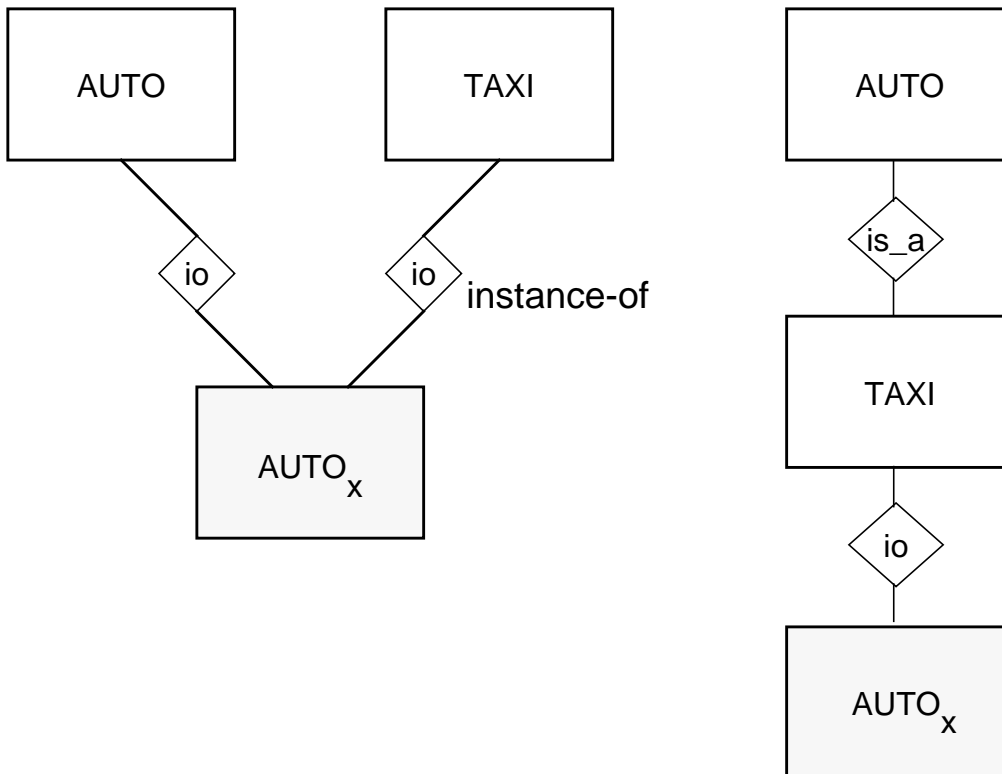
- Jeder Professor **hält** mehrere seiner Vorlesungen und **prüft** Studenten jeweils über eine dieser Vorlesungen.
- Mehrere Assistenten **arbeiten** jeweils für einen Professor und **halten** Übungen, die zu den entsprechenden Vorlesungen **gehören**.
- Mehrere Studenten **hören** jeweils eine Reihe von Vorlesungen. Übungen und Vorlesungen werden jeweils von mehreren Studenten **besucht**.
- Der Besuch von Vorlesungen **setzt** i. allg. die Kenntnis anderer Vorlesungen **voraus**.



ER-Diagramm - Vorlesungsbetrieb

# Erweiterungen des ERM

- „Alles dreht sich um die genauere Modellierung von Beziehungen“
- **Beispiel:** Unangemessene Modellierung bei überlappenden EM



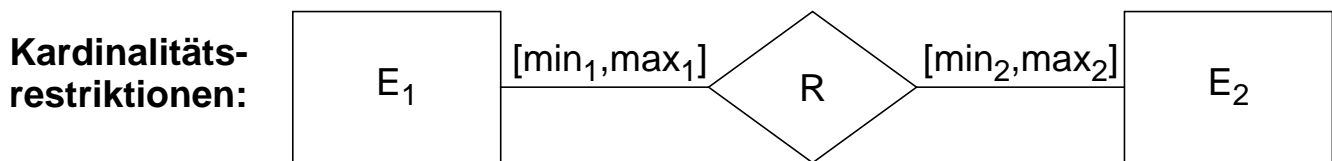
- **Ziele**
  - Verfeinerung der Abbildungen von Beziehungen durch **Kardinalitätsrestriktionen**
  - Ausprägungen (Objekte) einer EM sollen im Modell explizit dargestellt werden
  - gleichartige Darstellung von Ausprägung und Typ (EM)
  - Einführung von systemkontrollierten Beziehungen (**Abstraktionskonzepte**)



## Verfeinerung der Datenabbildung: Kardinalitätsrestriktionen

- bisher: grobe strukturelle Festlegung der Beziehungen  
z. B.: 1:1 bedeutet „höchstens eins zu höchstens eins“
- Verfeinerung der Semantik eines Beziehungstyps durch Kardinalitätsrestriktionen:  
sei  $R \subseteq E_1 \times E_2 \times \dots \times E_n$   
Kardinalitätsrestriktion  $\text{kard}(R, E_i) = [\text{min}, \text{max}]$   
bedeutet, daß jedes Element aus  $E_i$  in wenigstens min und höchstens max Ausprägungen von R enthalten sein muß (mit  $0 \leq \text{min} \leq \text{max}$ ,  $\text{max} \geq 1$ ).

### graphische Darstellung



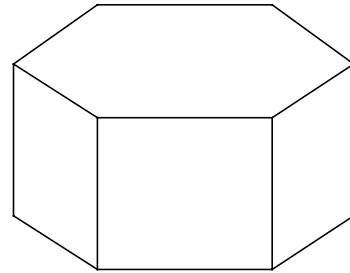
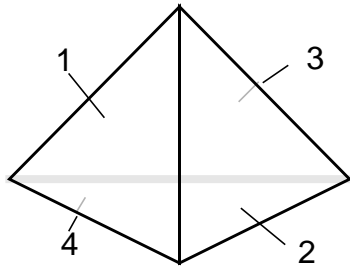
$e_1$  nimmt an  $[\text{min}_1, \text{max}_1]$  Beziehungen von Typ R teil  
 $e_2$  nimmt an  $[\text{min}_2, \text{max}_2]$  Beziehungen von Typ R teil

### Beispiele:

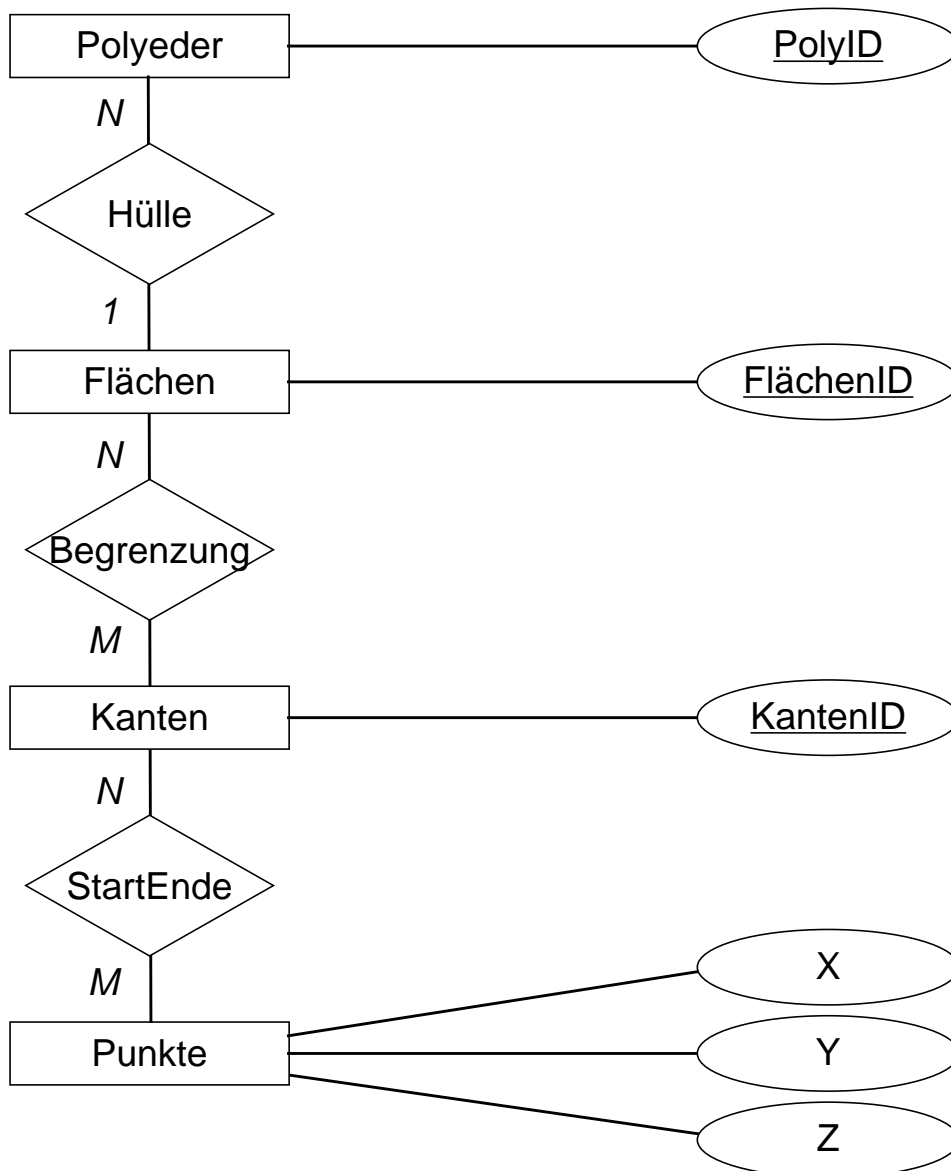
R	$E_1$	$E_2$	$\text{kard}(R, E_1)$	$\text{kard}(R, E_2)$
Abt-Leitung	ABT	PERS		
Heirat	FRAU	MANN		
Eltern	PAARE	KIND		
Abt-Angehörigk.	ABT	PERS		
V.Teilnahme	VORL	STUDENT		
Mitarbeit	PERS	PROJEKT		

# Begrenzungsflächendarstellung von Körpern

Beispiel-Körper:

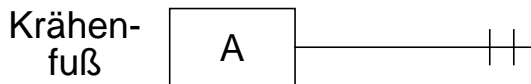
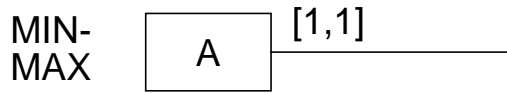


ER-Diagramm:

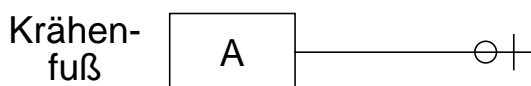
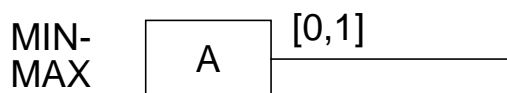


# Notationen

- Viele Systeme erlauben als Kardinalitätsrestriktionen nur 0, 1 oder n
- Jedes Element von A nimmt an genau einer Beziehung teil

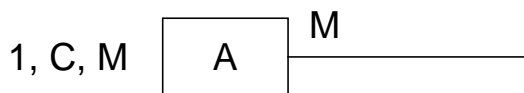
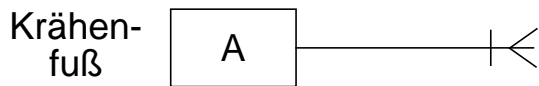
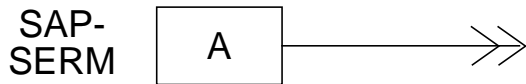
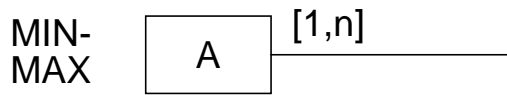


- Jedes Element von A nimmt an höchstens einer Beziehung teil

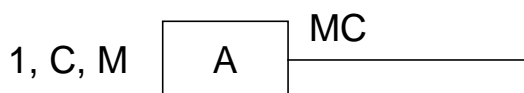
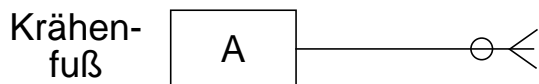
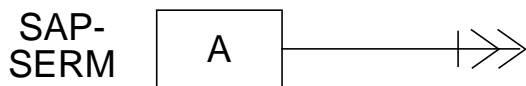


## Notationen (2)

- Jedes Element von A nimmt an mindestens einer Beziehung teil

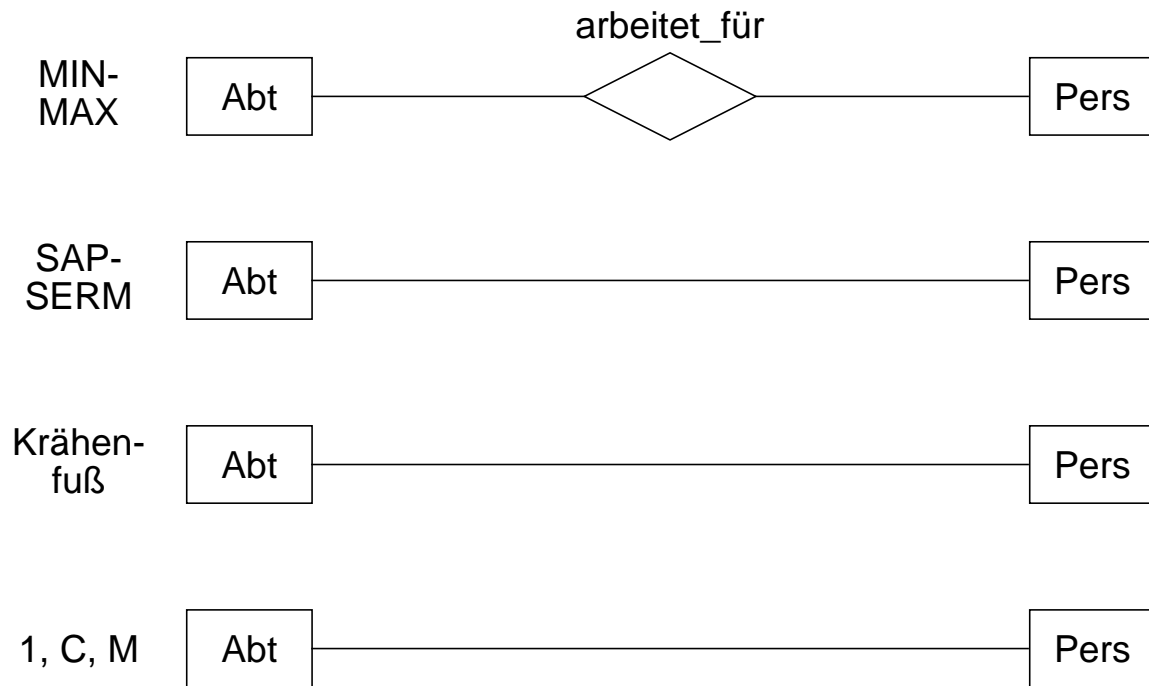


- Jedes Element von A kann an beliebig vielen Beziehungen teilnehmen

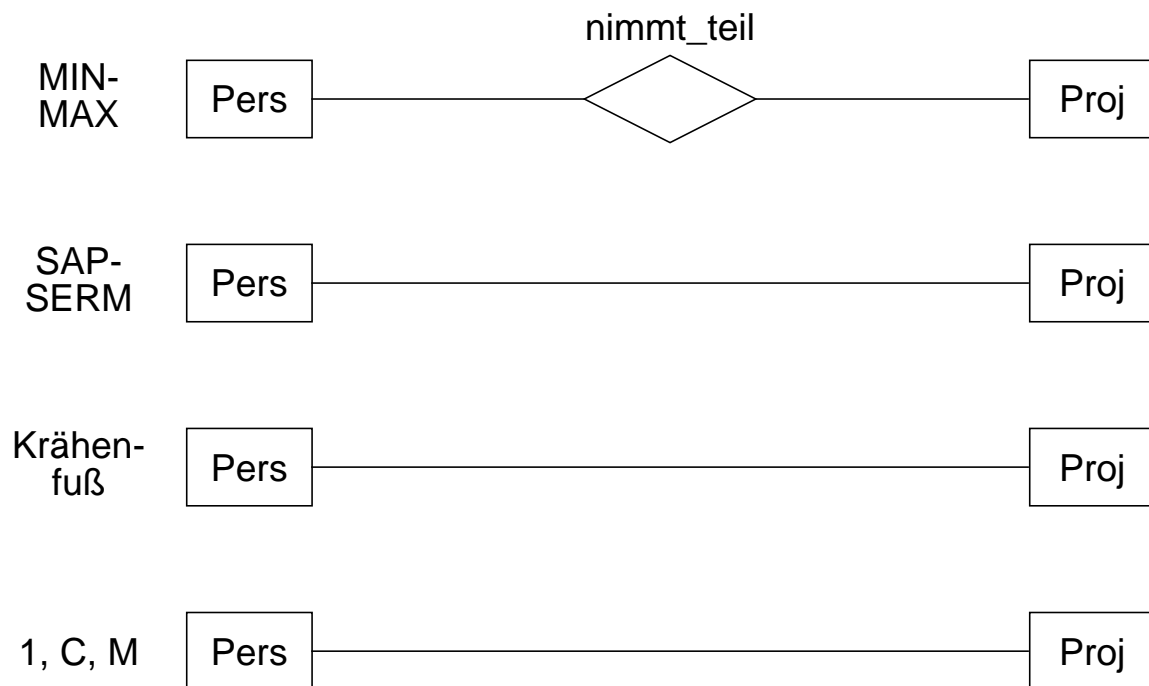


# Notationen - Beispiel

- 1:n



- n:m



# Abstraktionskonzepte<sup>1</sup>

- **Ziel:**

- Erfassung von noch mehr Semantik aus der Miniwelt durch das ERM
- Entwicklung von (Beschreibungs-)Modellen zur adäquateren Wiedergabe der ausgewählten Miniwelt (Diskursbereich)
- Definition von **systemkontrollierten Beziehungen**

- **Aufgabe:**

- Identifikation von wesentlichen Konstrukten, die vom Menschen angewendet werden, wenn er seinen Diskursbereich beschreibt.
  - Anwendung von **Abstraktion**, um die Information zu organisieren:  
“**abstraction permits someone to suppress specific details of particular objects emphasizing those pertinent to the actual view**”

- **Zwei Typen von Abstraktionen**

- von einfachen zu zusammengesetzten Objekten (*1-Ebenen-Beziehung*)
- von zusammengesetzten zu (komplexer) zusammengesetzten Objekten (*n-Ebenen-Beziehungen*)

- **Abstraktionskonzepte werden vor allem eingesetzt**

- zur Organisation der Information und damit auch
- zur Begrenzung des Suchraumes beim Retrieval sowie
- zu systemkontrollierten Ableitungen (Reasoning)

- **Übersicht**

- Klassifikation – Instantiation
- Generalisierung – Spezialisierung
- Element-/Mengenassoziation
- Element-/Komponentenaggregation

---

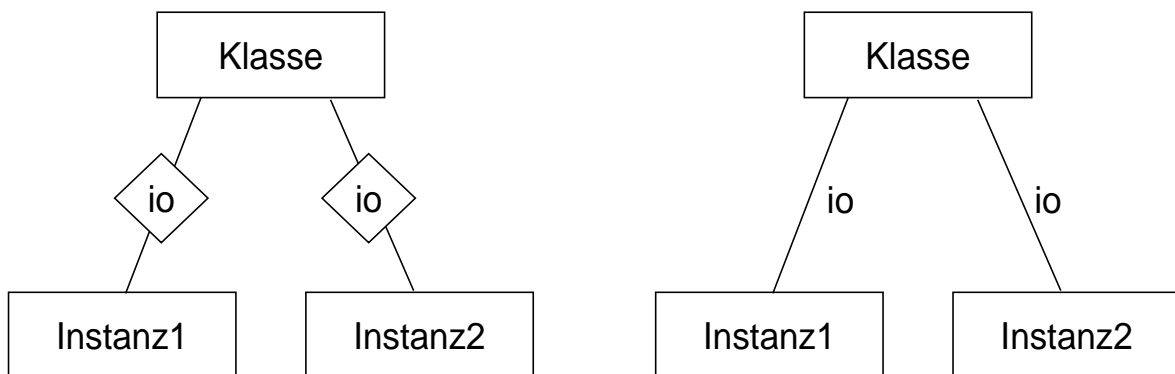
1. Mattos, N.: An Approach to Knowledge Management, LNAI 513, Springer, 1991

# Klassifikation

- Klassifikation entspricht der Bildung von Entity-Mengen:  
Sie faßt Objekte (*Entities*) mit gemeinsamen Eigenschaften zu einem neuen zusammengesetzten Objekt (Entity-Typ, **Klasse**, Klassenobjekt) zusammen
- Eine Klasse ist definiert als Zusammenfassung von Objekten **gleichen Typs** (und gleicher Repräsentation).  
Dadurch nur einmalige Definition von
  - Attributnamen und -typen
  - Methoden
  - Integritätsbedingungen
- Es wird eine '**instance-of**'-Beziehung ('**io**') als 1-Ebenen-Beziehung zu den Objekten der Klasse aufgebaut

# Instantiation

- Instantiation ist das inverse Konzept zur Klassifikation
- Sie wird benutzt, um zu Instanzen/Objekten zu gelangen, die den Eigenschaften der Klasse unterliegen
  - gleiche Struktur (Attribute)
  - gleiche Operationen
  - gleiche Integritätsbedingungen
- Klassifikation/Instantiation sind die primären Konzepte zur **Objektbildung und -strukturierung**
- **graphische Darstellung**



Die Darstellungen der anderen Abstraktionskonzepte erfolgen entsprechend.



# Generalisierung

- **Aufgabe**

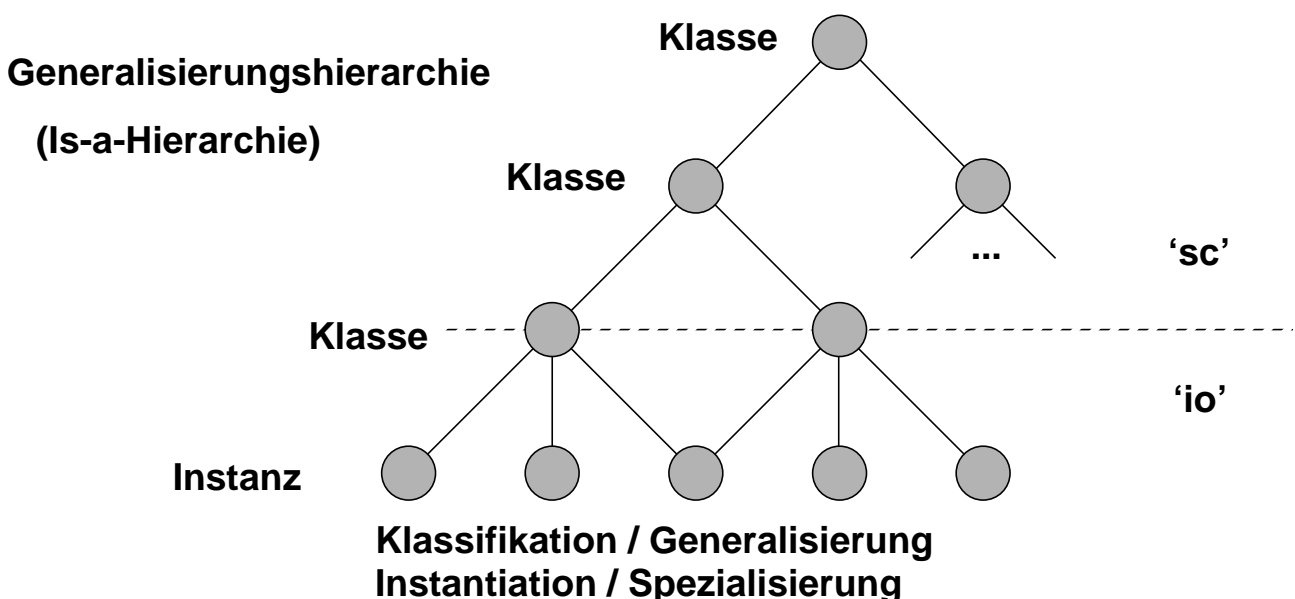
Generalisierung ist ein ergänzendes Konzept zur Klassifikation. Durch sie wird eine allgemeinere Klasse definiert, die die Gemeinsamkeiten der zugrundeliegenden Klassen aufnimmt und deren Unterschiede unterdrückt

- **Anwendung**

- Sie baut die **'subclass-of'**-Beziehung auf (**'sc'**- oder **'is-a'**-Beziehung)
- Sie ist rekursiv anwendbar (n-Ebenen-Beziehung) und organisiert die Klassen in einer Generalisierungshierarchie
- Eine Superklasse ist eine Verallgemeinerung/Abstraktion der zugehörigen Subklassen. Sie entspricht einem komplex zusammengesetzten Objekt, das gebildet wird als Kollektion von komplex zusammengesetzten Objekten (Subklassen)

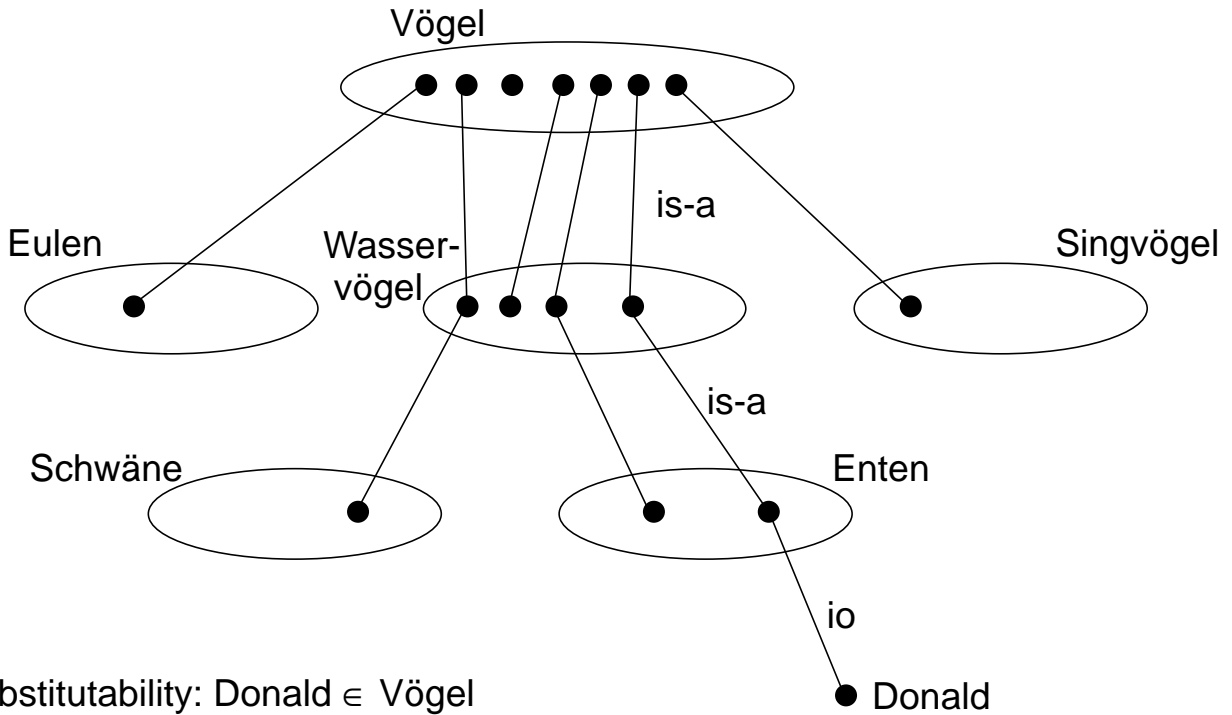
- **Struktureigenschaften der Generalisierung**

- Alle Instanzen einer Subklasse sind auch Instanzen der Superklasse
- Ein Objekt kann gleichzeitig Instanz verschiedener Klassen sein sowie auch Subklasse mehrerer Superklassen (→ Netzwerke, (n:m) !)
- Zugehörigkeit eines Objektes zu einer Klasse/Superklasse wird im wesentlichen bestimmt durch **Struktur** (Attribute), **Verhalten** (Operationen) und **Integritätsbedingungen** der Klasse/Superklasse

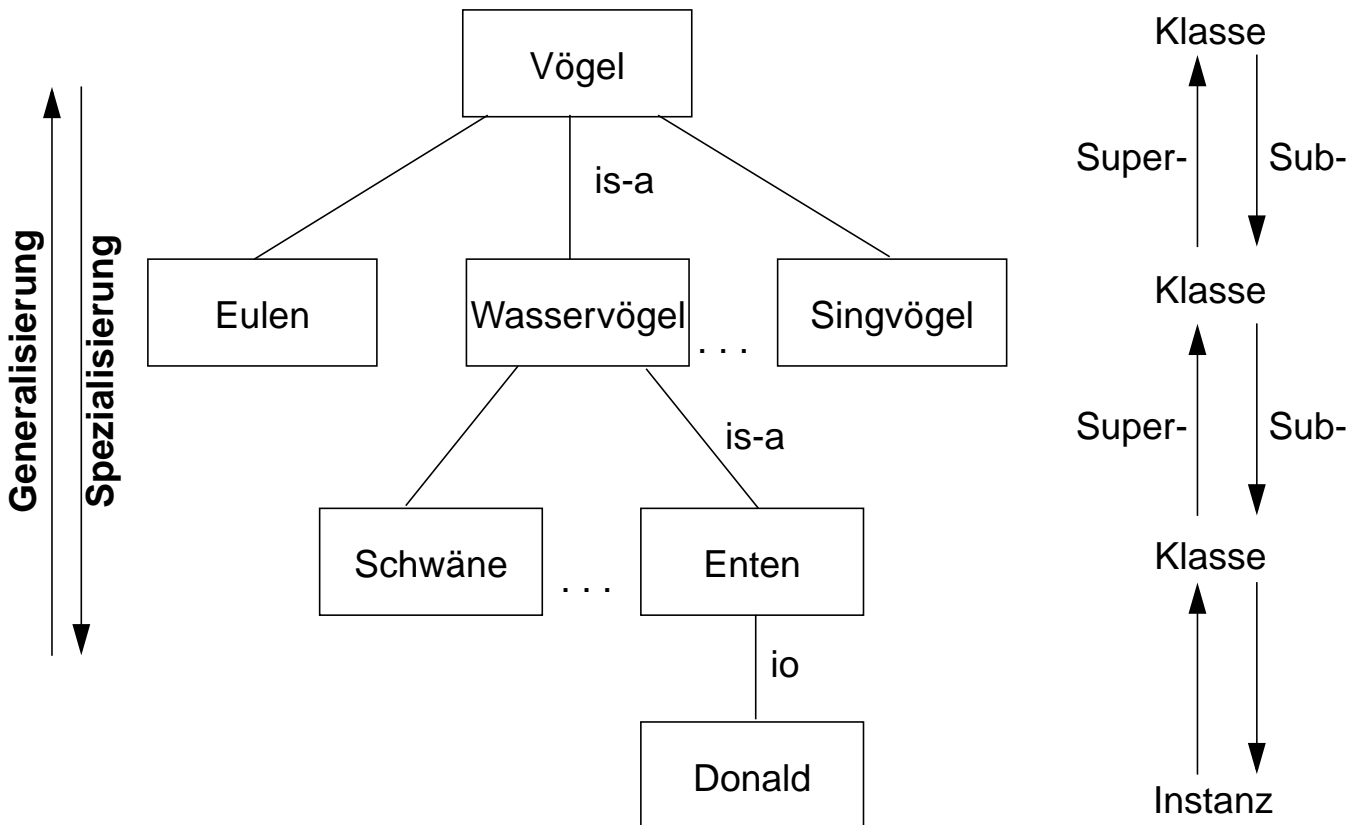


# Modellierungsbeispiel zur Generalisierung

## Instanzendarstellung

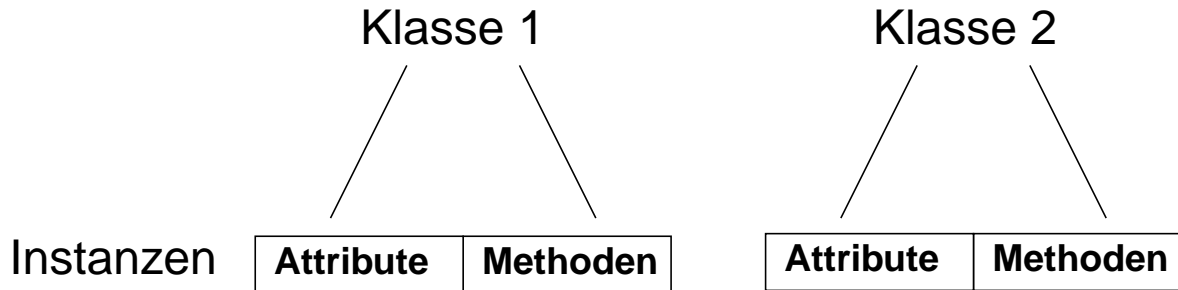


## Typdarstellung



## Generalisierung (2)

- **Objekte können gleichzeitig Instanzen mehrerer Klassen sein:**



Dabei können Attribute (und Methoden) mehrfach eingerichtet werden:

**Klasse** Autofahrer (Name, Geburtstag, Führerscheinklasse, ... )

**Klasse** Student (Name, Geburtstag, Matrikelnr, ... )

**Grund:** Autofahrer und Studenten sind beide Personen,  
und in dieser „Rolle“ haben beide Namen und Geburtstag

→ Generalisierungsschritt: **Klasse** Person einrichten

- **aber:**

Studenten oder Autofahrer, die keine Personen sind, darf es nicht geben!  
(Es kann jedoch Personen geben, die weder Studenten noch Autofahrer sind)

- **Beziehung zwischen den Klassen:**

Student (Autofahrer) ist **Subklasse** von Person

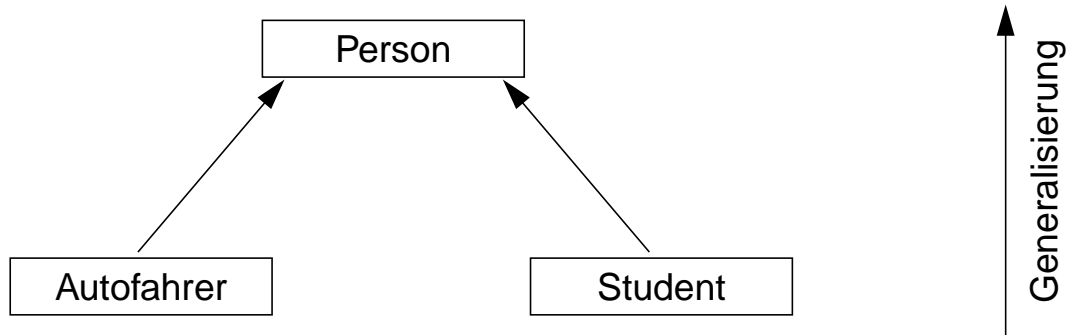
Person ist **Superklasse** von Student und Autofahrer

- jede Instanz der Subklasse ist immer auch Instanz der Klasse, aber nicht umgekehrt
- jede Methode, die auf die Instanzen einer Klasse anwendbar ist, ist damit immer auch auf die Instanzen sämtlicher Subklassen anwendbar

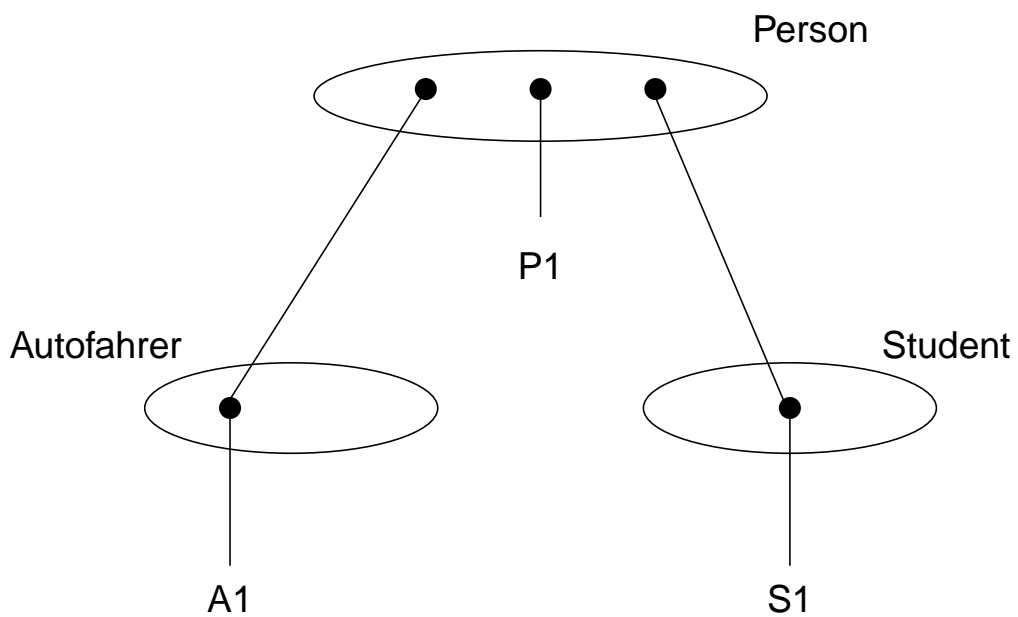
→ Garantie von bestimmten Integritätsbedingungen durch das System:  
jeder Student ist auch Person

# Generalisierung - Beispiel

- **Klassen**



- **Instanzen**

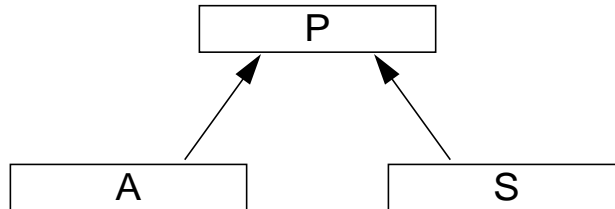


- **Subklassen sind i. allg. nicht disjunkt!**

## Generalisierung - Beispiel (2)

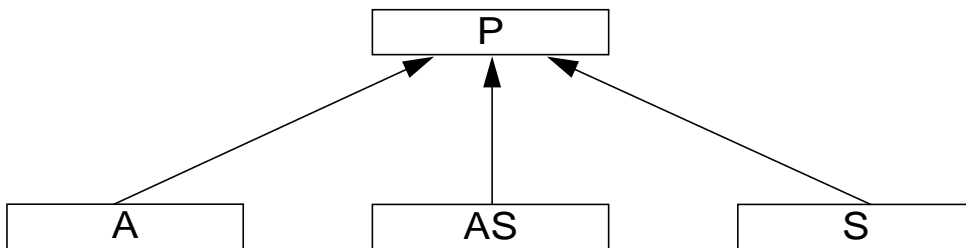
- **Überlappende Subklassen - Ansätze:**

### 1. Mehrklassenmitgliedschaft von Instanzen



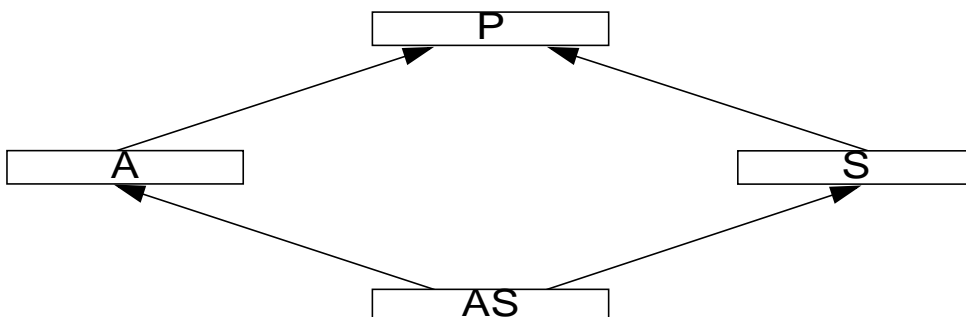
Wartung hat durch Programmierer zu erfolgen

### 2. Einklassenmitgliedschaft von Instanzen (Einfachvererbung)



Mehrfache Definition von Attributen, Integritätsbedingungen usw.,  
Wartung erfolgt durch System

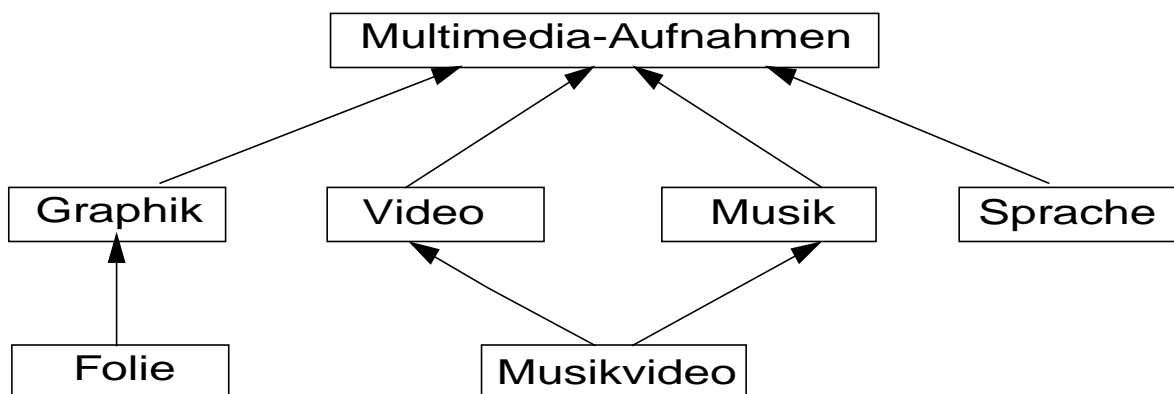
### 3. Einklassenmitgliedschaft von Instanzen (Mehrfachvererbung)



Einmalige Definition von Attributen, Integritätsbedingungen usw.,  
Wartung erfolgt durch System,  
aber: komplexere Schemastrukturen, eingeschränkte Erweiterbarkeit

## Generalisierung (3)

- **historisches Vorbild: Carl von Linné**
  - biologische Systematik
  - Reich (z.B. Tierreich) – Stamm (Chordatiere) – Klasse (Säugetiere) – Familie – Gattung – Art
  - daher auch: „Art-Gattungs-Beziehung“
- bei manchen Systemen höchstens eine Superklasse pro Klasse (Klassen bilden **Baum** bzw. Hierarchie)
- bei anderen beliebig viele („**Klassenverband**“, gerichteter azyklischer Graph)
- **Diese Restriktion bestimmt entscheidend die Definition von Klassen:**



→ Was ist zu tun, wenn nur Baumstrukturen erlaubt sind ?

- **evtl. zusätzlich spezifizieren:**
  - Subklassen müssen disjunkt sein (keine Mehrklassen-Mitgliedschaft)
  - Subklassen müssen vollständig (überdeckend) sein:  
jede Instanz der Klasse stets auch in einer der Subklassen  
(abstrakte Superklasse: hat keine direkten Instanzen)

# Spezialisierung

- **Aufgabe**

Spezialisierung ist das inverse Konzept zur Generalisierung.  
Sie unterstützt die 'top-down'-Entwurfsmethode:

- zuerst werden die allgemeineren Objekte beschrieben (Superklassen)
- dann die spezielleren (Subklassen)

- **Systemkontrollierte Ableitung**

Dabei wird natürlich das Konzept der **Vererbung** ausgenutzt:

- Superklassen-Eigenschaften werden 'vererbt' an alle Subklassen, da diese auch dort gültig sind
- **Vorteile:**
  - keine Wiederholung von Beschreibungsinformation
  - abgekürzte Beschreibung
  - Fehlervermeidung

## Spezialisierung (2)

- **Vererbung von**

- **Struktur:** Attribute, Konstante und Default-Werte
- **Integritätsbedingungen:** Prädikate, Wertebereiche usw. sowie
- **Verhalten:** Operationen (auch Methoden genannt)

⇒ Es müssen **alle Struktur-, Integritäts- und Verhaltensspezifikationen** vererbt werden. Integritätsbedingungen können **eingeschränkt**, Default-Werte können **überschrieben**, Methoden **überladen** werden.

- **Arten der Vererbung**

- Einfach-Vererbung (eindeutig)
- Mehrfach-Vererbung

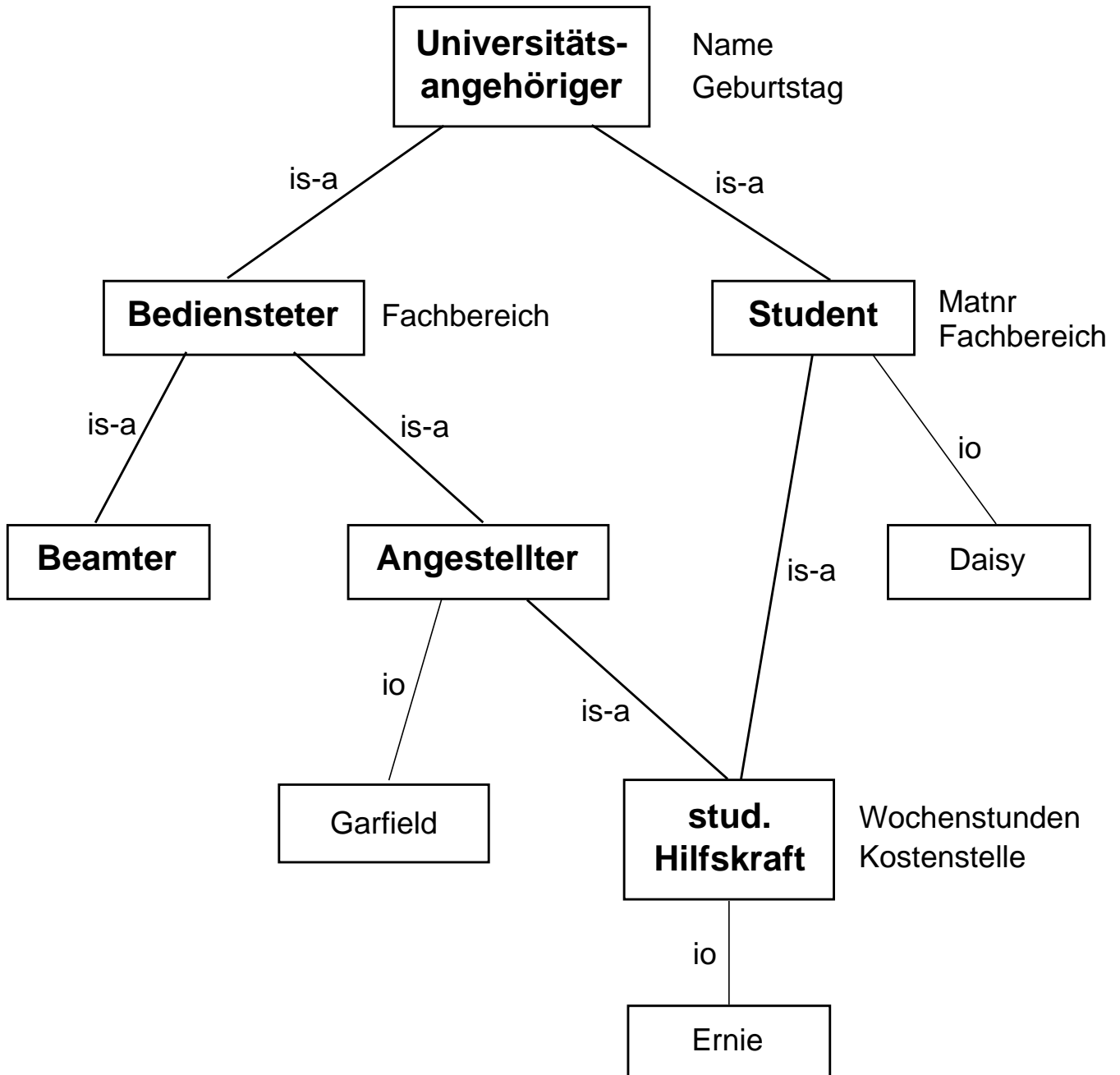
- **Schlußweise für Vererbungsregeln:**

HasAttribute (C1, A)	←	Isa (C1, C2), HasAttribute (C2, A)
HasValue (C1, A, V)	←	Isa (C1, C2), HasValue (C2, A, V)
P(..., C1, ...)	←	Isa (C1, C2), P (... , C2, ...)



# Vererbung (*Inheritance*)

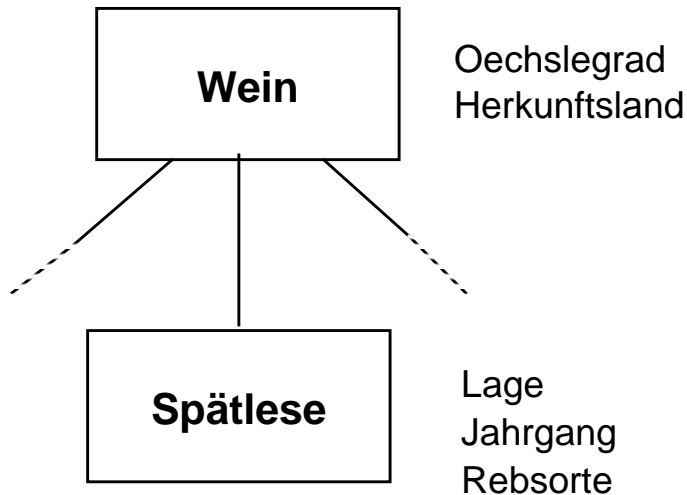
- Subklasse **erbt alle** Attribute der Superklasse



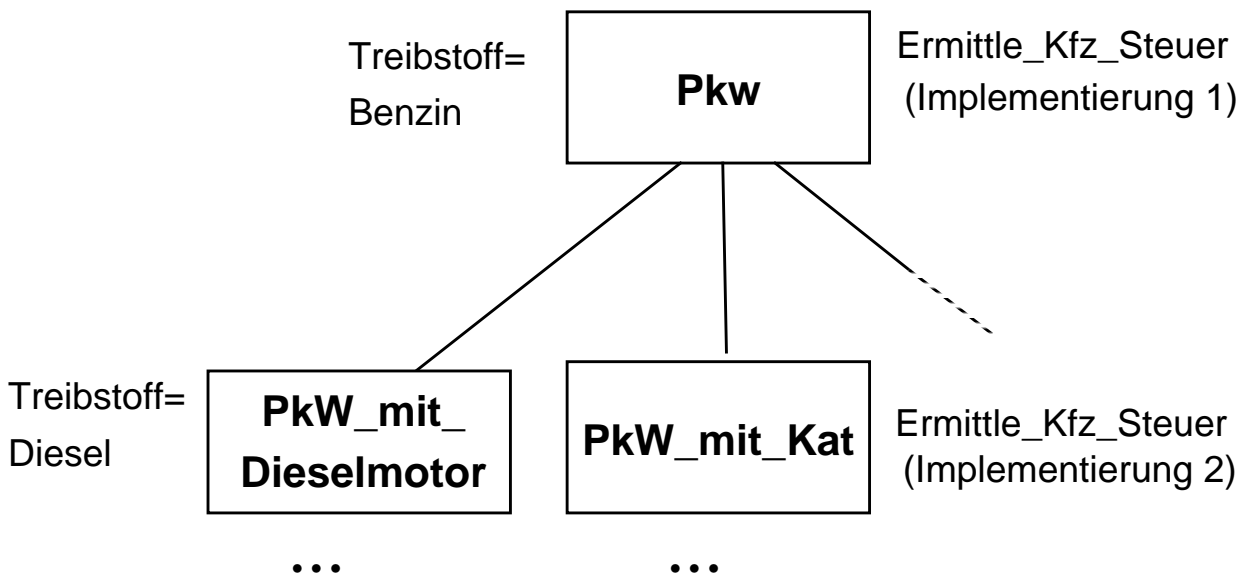
- **Mehrfach-Vererbung** (*multiple inheritance*) kann zu Konflikten führen  
→ Auflösung explizit durch den Benutzer, z. B. durch Umbenennung:  
Hiwi\_im\_Fachbereich → Fachbereich of Angestellter  
immatrikuliert\_im\_Fachbereich → Fachbereich of Student

## Vererbung (2)

- Subklasse kann den Wertebereich ererbter Attribute **einschränken**:



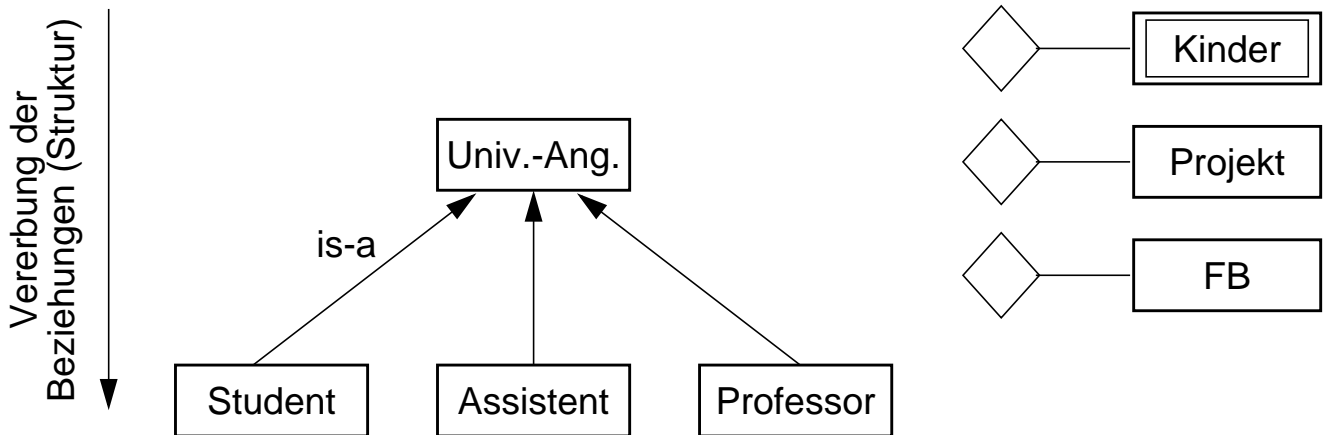
- Subklasse kann ererbte Attributwerte **überschreiben** oder Methoden **überladen**:



Methoden mit **gleichem Namen** und **unterschiedlicher Implementierung** (*Overloading*)

## Vererbung (3)

- Vererbung von Beziehungen:



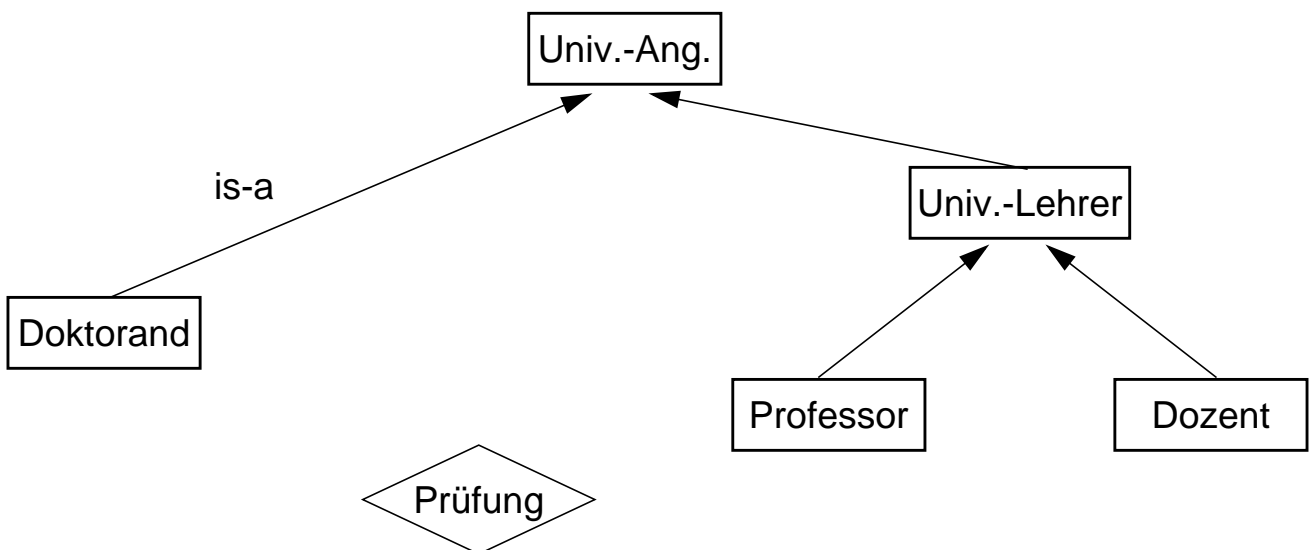
- Beispiel: Doktorprüfung

Drei-Weg-Beziehung zwischen Doktorand sowie zwei Professoren als Erst- und Zweitgutachter



- Verfeinerung von Doktorprüfung:

Erstgutachter muß Professor sein, Zweitgutachter kann Dozent sein



## Vererbung (4)

- **Mehrfach-Vererbung / mögliche Lösungen**

1. **Attribute:**

2. **Wertebereiche (zulässige Werte):**

3. **Defaultwerte:**

4. **Integritätsbedingungen (Prädikate):**

5. **Operationen (Methoden):**

### Spezialisierung: Definitionen

- **Subklasse:**

Klasse S, deren Entities eine Teilmenge einer Superklasse G sind:

$$S \subseteq G$$

d. h., jedes Element (Ausprägung) von S ist auch Element von G.

- **Spezialisierung:  $Z = \{S_1, S_2, \dots, S_n\}$**

Menge von Subklassen  $S_i$  mit derselben Superklasse G

Z heißt **vollständig (total)**, falls gilt

$$G = \cup S_i \quad (i = 1..n)$$

andernfalls **partiell**.

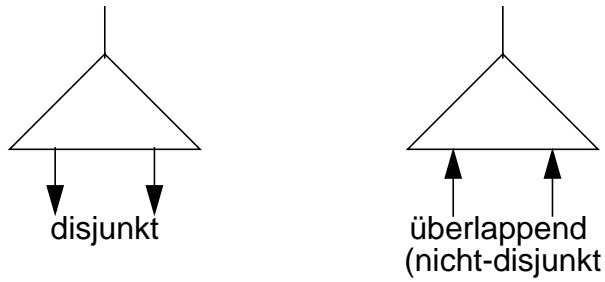
Z ist **disjunkt**, falls

$$S_i \cap S_j = \{ \} \quad \text{für } i \neq j$$

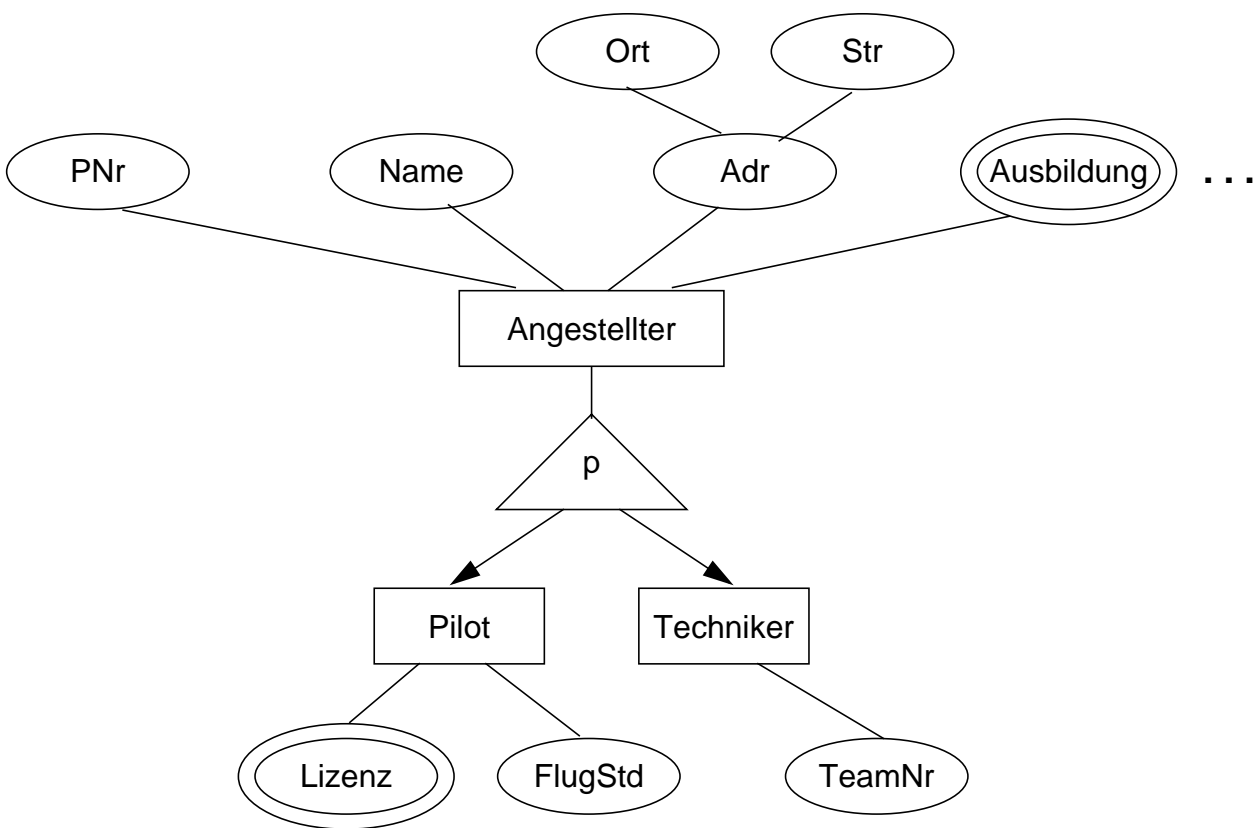
andernfalls **überlappend (nicht-disjunkt)**.

# Arten von Spezialisierungen

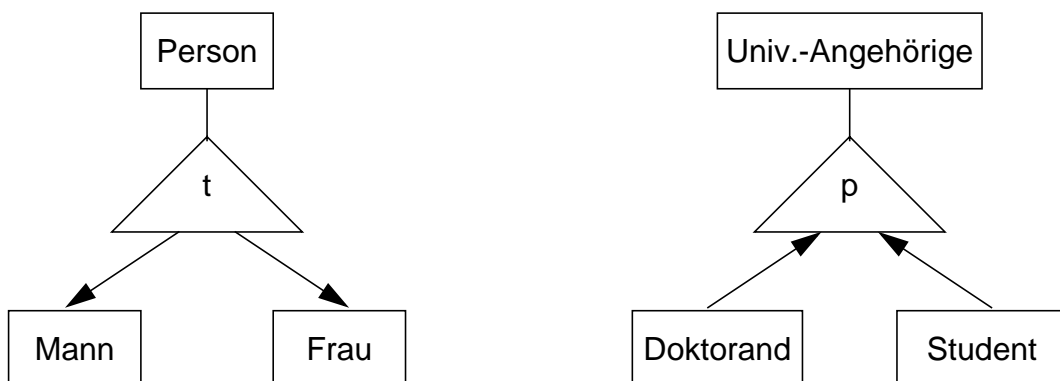
- Verfeinerung der is-a-Beziehung



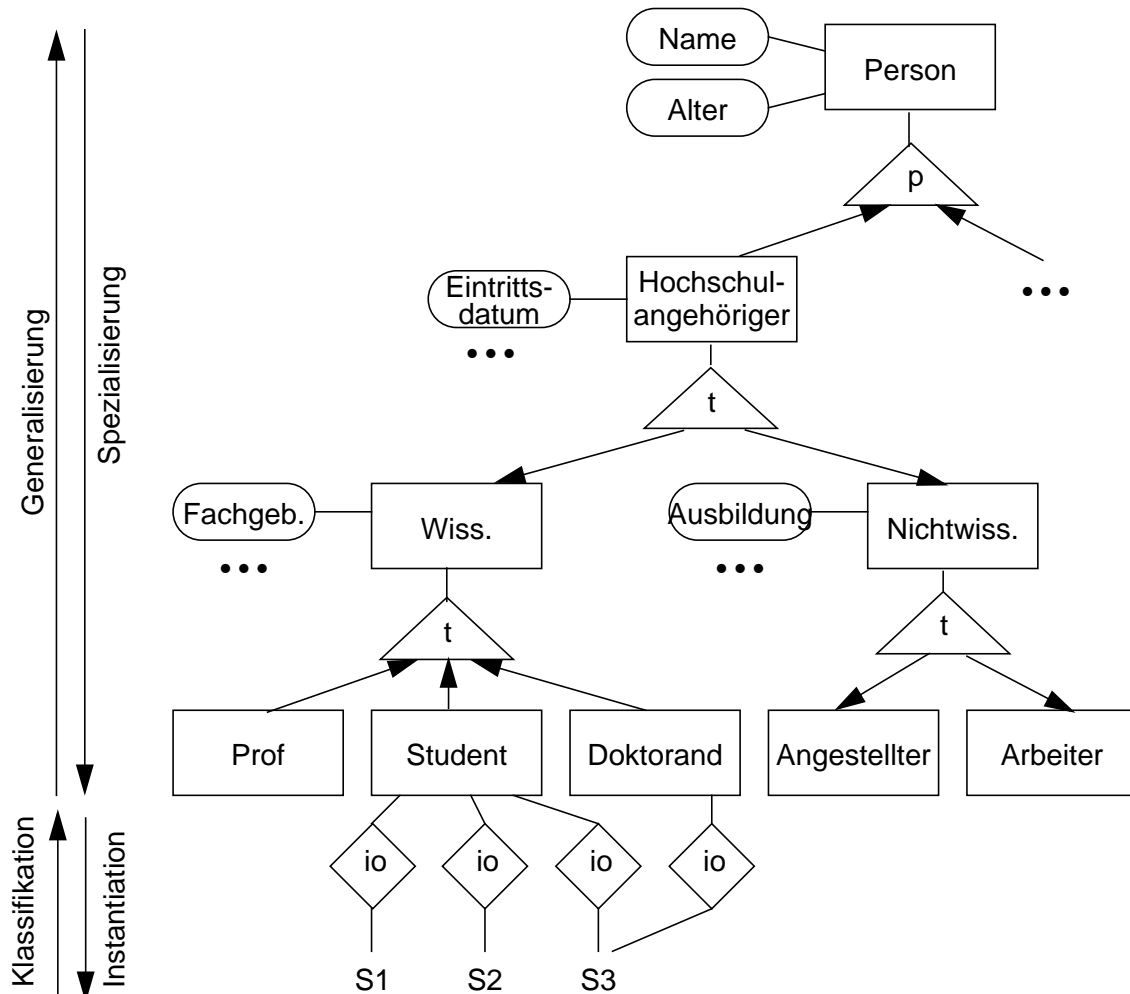
- Partielle, disjunkte Spezialisierung



- Weitere Spezialisierungen



# Abstraktionskonzept: Generalisierung/Spezialisierung



→ Generalisierungshierarchie als ER-Diagramm

- **Nutzung beim objektorientierten DB-Entwurf**

Vererbung von Typinformationen

- Strukturdefinitionen: Attribute, Defaultwerte, konstante Werte
- Integritätsbedingungen: Prädikate, Wertebereiche, Zusicherungen
- Verhalten: Operationen (Methoden) und ggf.
- Aspektdefinitionen: Kommentare, Einheiten u. a.

# Element-Assoziation

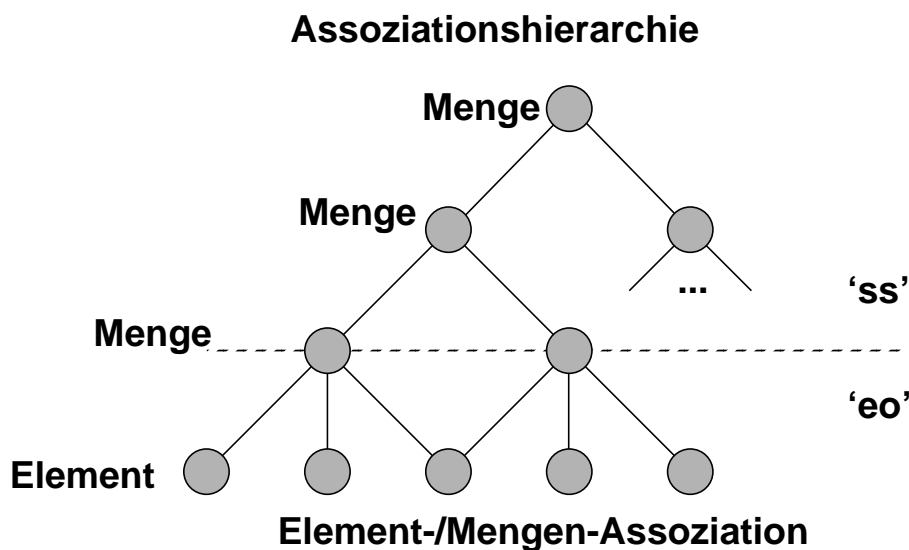
- **Aufgabe**

Die Element-Assoziation faßt Objekte (**Elemente**) zusammen, um sie im Rahmen einer Objektgruppe (**Mengenobjekt**) als Ganzes zu beschreiben. Dabei werden einerseits Details der einzelnen Elemente unterdrückt und andererseits bestimmte Eigenschaften, die die Objektgruppe charakterisieren, hervorgehoben.

- **Anwendung**

- Element-Assoziation (auch Gruppierung, Partitionierung, Überdeckungs-Aggregation genannt) baut zusammengesetzte (Mengen-)Objekte basierend auf den einfachen (Element-)Objekten auf.
- Sie verkörpert eine **'element-of'**-Beziehung (**'eo'**) als 1-Ebenen-Beziehung.
- Es können auch heterogene Objekte zu einem Mengenobjekt zusammengefaßt werden. Bei automatischer Ableitung müssen die Objekte das Mengenprädikat erfüllen. Bei manuellem Aufbau wählt der Benutzer die Objekte aus und verknüpft sie mit dem Mengenobjekt (Connect).

- **Graphische Darstellung:**



# Mengen-Assoziation

- **Aufgabe**

Die Mengen-Assoziation ist ein ergänzendes Konzept zur Element-Assoziation. Sie drückt Beziehungen zwischen zusammengesetzten Mengenobjekten aus

- **Anwendung**

- Sie baut eine '**subset-of**'-Beziehung ('**ss**') auf
- Sie ist rekursiv anwendbar und organisiert die Mengenobjekte in einer Assoziations-Hierarchie (n-Ebenen-Beziehung)

- **Struktureigenschaften der Assoziation**

- Alle Elemente eines Mengenobjekts sind auch Elemente der zugehörigen Supermenge
- Objekte können gleichzeitig Elemente verschiedener Mengenobjekte sein sowie auch Teilmenge von mehreren Supermengen  
→ Netzwerke, (n:m) !

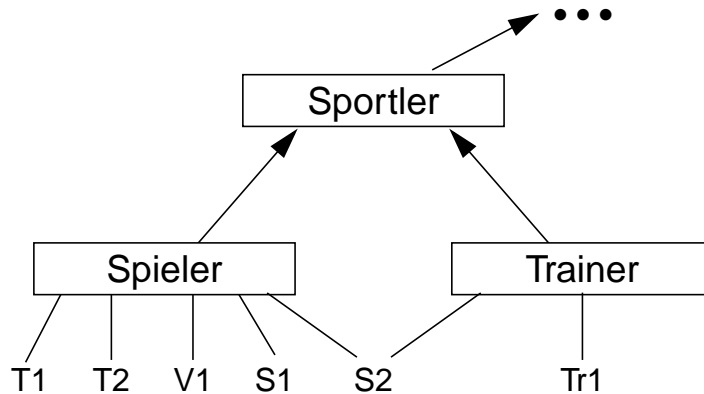
- **Systemkontrollierte Ableitungen bei der Assoziation**

- Sie unterstützt keine Vererbung, da die Mengeneigenschaften keine Element-eigenschaften sind
- Durch **Mitgliedschaftsimplication** lassen sich Eigenschaften bestimmen, die jedes gültige Element der Menge erfüllen muß
- **Mengeneigenschaften** sind Eigenschaften der Menge, die über Element-eigenschaften abgeleitet sind

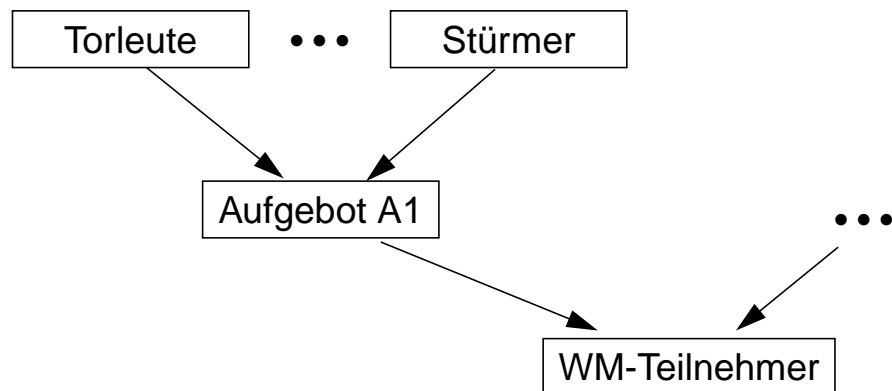


# Assoziation - Beispiel

- **Generalisierungshierarchie**



## Assoziation

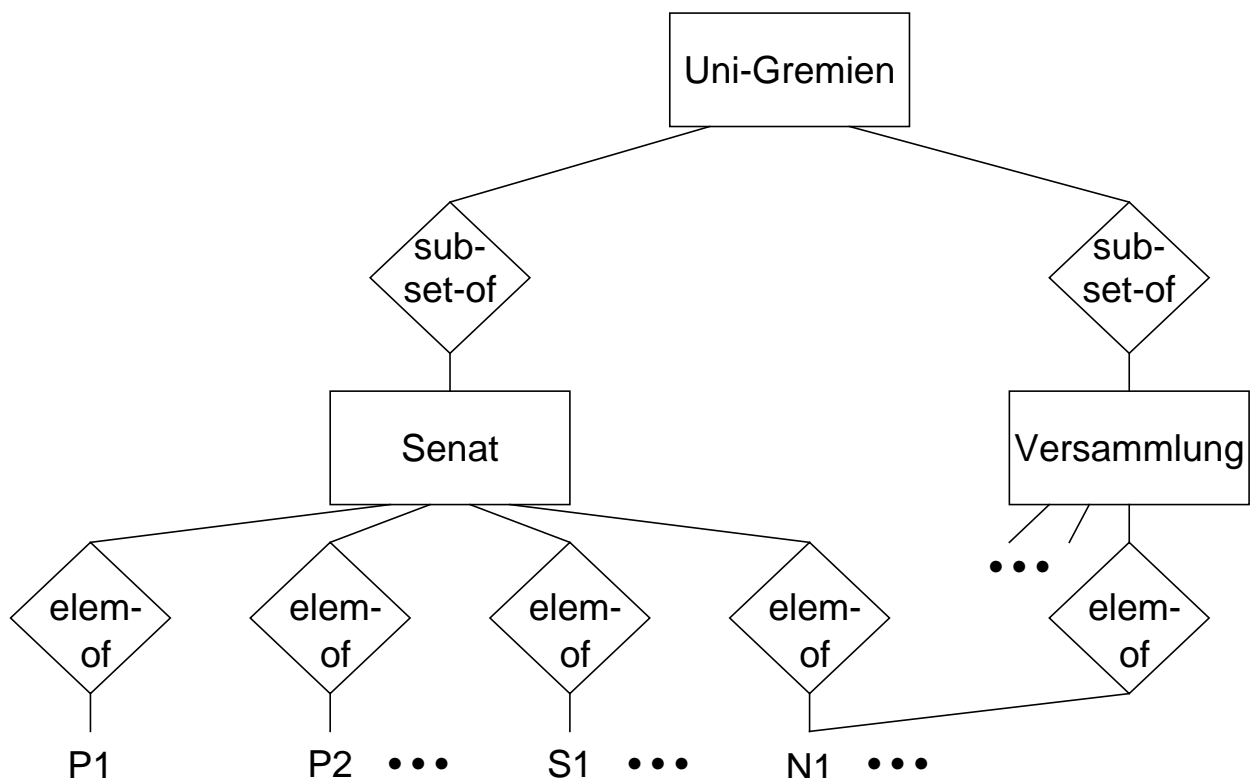


- **Operationen**

- Erzeugen/Löschen
  - Create
  - Connect/Disconnect  
(manuell oder automatisch über Mengenprädikate)
  - Delete
- Suchen
- Schlußfolgerungen
  - Mitgliedschaftsimplication
  - Mengeneigenschaften

# Abstraktionskonzept: Assoziation

- Zusammenfassung einer Menge von Objekten **potentiell verschiedenen Typs** zu einem semantisch höheren Objekt
- Neben der Element-Assoziation ist Mengen-Assoziation möglich
- „element-of“-Beziehung und „subset-of“-Beziehung



- **zusätzliche Prädikate** (z. B. GEWAEHLT) zur automatischen Bestimmung der konkreten Menge erforderlich
- Ableitung von Objekteigenschaften durch *Mitgliedschaftsimplication* (z. B. Senatsmitglied)
- Ableitung von *Mengeneigenschaften* (z. B. Anzahl der Senatsmitglieder)
- Ziel:  
**Zusammenfassung von Gruppen mit heterogenen Objekten** für einen bestimmten Kontext (Vergleiche Sichtkonzept)

# Aggregation

- **Beziehung mit spezieller zusätzlicher Bedeutung:**

Das Objekt, auf das sie verweist, soll **Bestandteil** sein  
(**Teil-Ganze-Beziehung**),

z. B.

Auto	–	Motor
Tisch	–	Tischplatte
Kante	–	Endpunkt
Bild	–	Farbtabelle

- entweder **exklusiv:**

kein anderes Objekt darf denselben Bestandteil haben

oder **gemeinsam:**

derselbe Bestandteil wird in zwei oder mehr Objekten verwendet

- entweder **abhängig:**

Bestandteil kann nicht allein existieren;

wird mit dem Objekt gelöscht

oder **unabhängig:**

Bestandteil kann auch für sich als Objekt existieren

→ Objekte mit exklusiven und/oder abhängigen Objekten heißen

**zusammengesetzte Objekte**

(„composite objects“, „komplexe Objekte“)

oder **Aggregate**

# Element-Aggregation

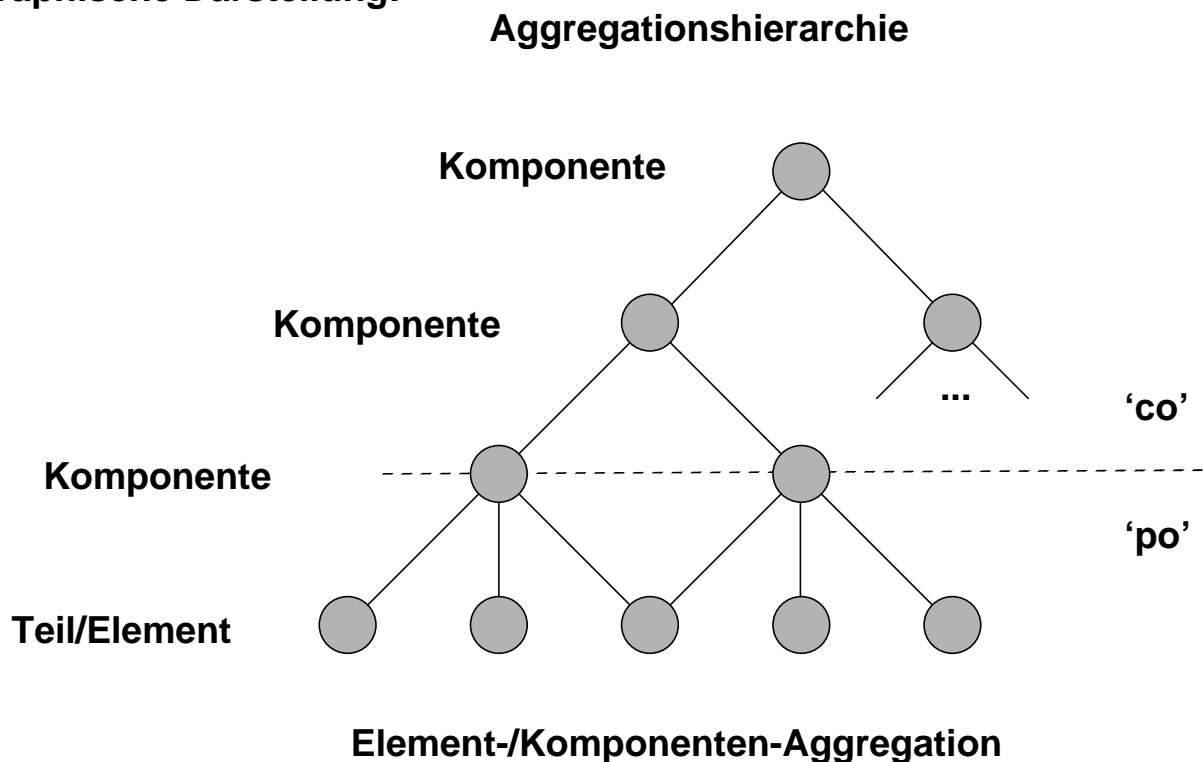
- **Aufgabe**

Die Element-Aggregation gestattet die Zusammensetzung von Objekten aus einfachen Objekten. Sie stellt die 'Teil-Ganze'-Relation für solche nicht weiter zerlegbaren Objekte her

- **Anwendung**

- Eine Kollektion von einfachen Objekten (Element-Objekt, **Teil**) wird als zusammengesetztes Objekt (**Komponentenobjekt/Aggregatobjekt**) behandelt
- Sie baut eine **'part-of'**-Beziehung (**'po'**) auf (1-Ebenen-Abstraktion). Typischerweise erzeugt der Benutzer ein Aggregat aus Teilen mit Hilfe von Connect-Anweisungen; dabei müssen Struktureigenschaften beachtet werden (z. B. Mannschaft besitzt 11 Spieler)
- Die Möglichkeit, heterogene Objekte zu aggregieren, erhöht die Anwendungsflexibilität

- **Graphische Darstellung:**



# Komponenten-Aggregation

- **Aufgabe**

Die Komponenten-Aggregation dient als ergänzendes Konzept zur Element-Aggregation. Durch sie wird die Teil-Ganze-Relation auf Komponenten angewendet

- **Anwendung**

- Zwischen den Komponentenobjekten wird eine '**component-of**'-Beziehung ('**co**') hergestellt (z. B. durch Connect-Anweisung)
- Sie ist rekursiv anwendbar und organisiert eine Aggregationshierarchie (n-Ebenen-Beziehung)

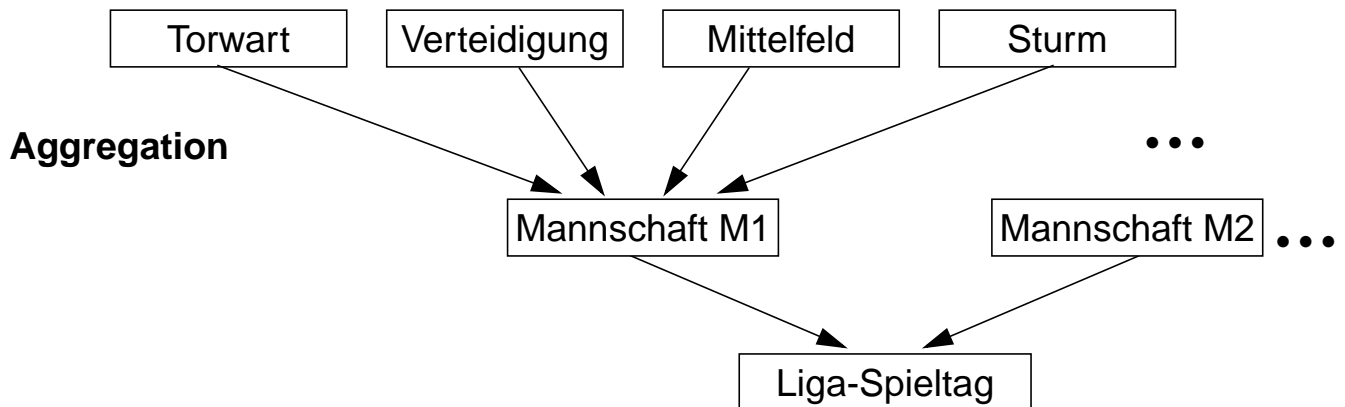
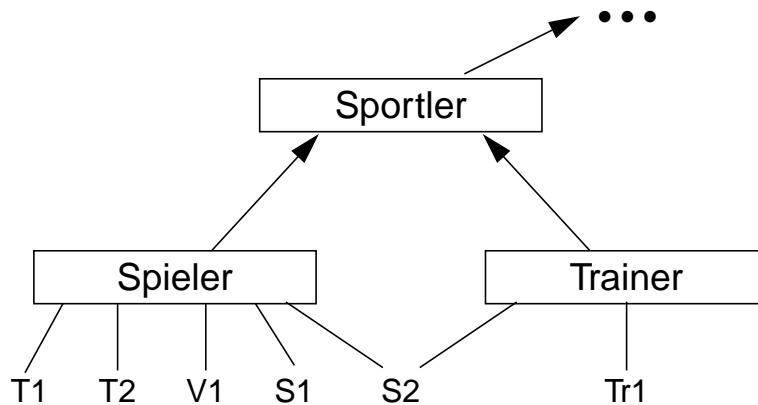
- **Struktureigenschaften bei der Aggregation**

(Aggregation bedeutet auch 'besteht-aus'/'consists-of')

- beschreibt *notwendige* Eigenschaften, die ein Objekt haben muß, um konsistent zu sein
  - Unterschied zu Klassen und Mengenobjekten, die ohne Instanzen existieren können, bzw. für die leere Mengen erlaubt sind
- Elemente einer Subkomponente sind gleichzeitig auch Elemente aller Superkomponenten dieser Subkomponente
- Objekte können gleichzeitig Elemente verschiedener Komponenten bzw. auch Subkomponente von mehreren Superkomponenten sein
  - Netzwerke, ((n:m) !)

# Aggregation - Beispiel

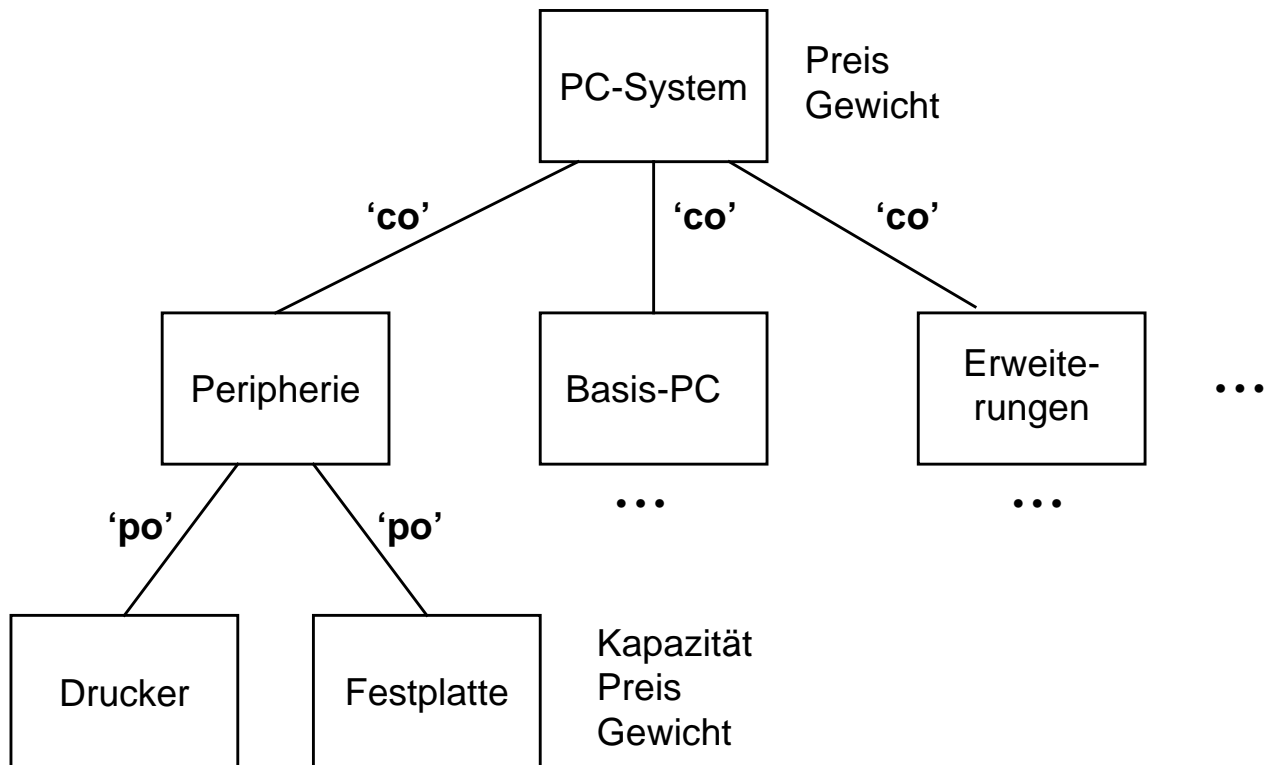
- **Generalisierungshierarchie**



- **Operationen**

- Erzeugen/Löschen
  - Create
  - Connect/Disconnect
  - Delete
- Integritätsbedingungen für Aggregatstrukturen
- Suchen (transitive Ableitung von komplexen Objekten)
- Schlußfolgerungen
  - implizierte Prädikate

## Aggregation - Beispiel (2)



- **Systemkontrollierte Ableitungen: implizierte Prädikate**

- Prädikate, die über der Aggregationshierarchie spezifiziert sind und gemeinsame Eigenschaften von Elementen/Aggregaten betreffen

- 'upward implied predicate'

Wenn  $P(x)$  wahr  $\Rightarrow$   $P$  (Aggregatobjekte  $(x)$ ) wahr

- 'downward implied predicate'

Wenn  $P(x)$  wahr  $\Rightarrow$   $P$  (Komponentenobjekte  $(x)$ ) wahr

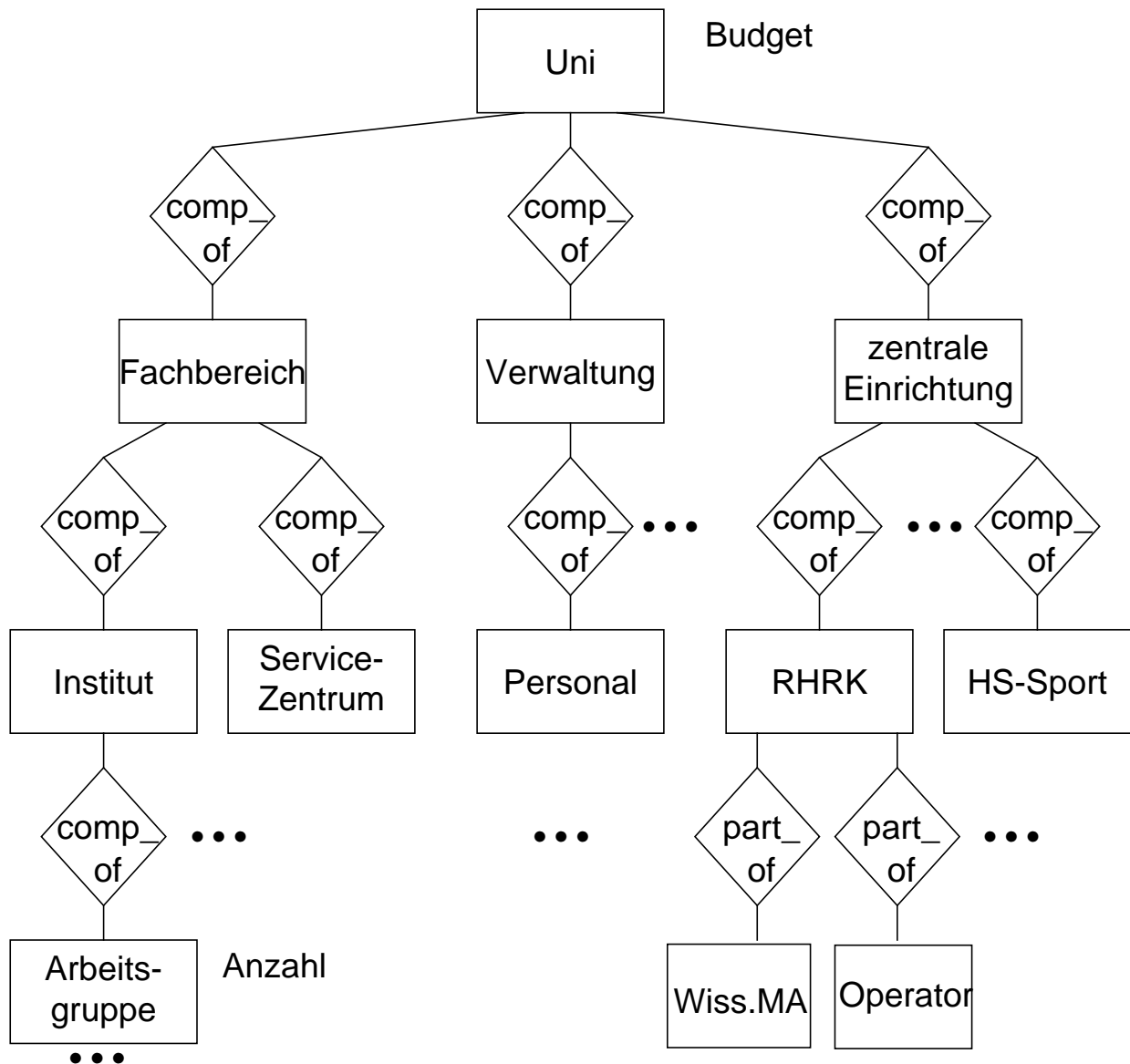
- im Beispiel:

- 'upward implied predicate': Gewicht  $>$  x

- 'downward implied predicate': Preis  $<$  y

# Abstraktionskonzept: Aggregation

- (Komponenten-) Objekte lassen sich zu einem neuen Objekt zusammenfassen
- Element- und Komponenten-Aggregation möglich
- 'part-of'-Beziehung und 'component-of'-Beziehung



- **Ableitung von Objekteigenschaften** (*implied predicates*)
  - upward implied predicate (Anzahl > x)
  - downward implied predicate (Budget < y)



# Integrierte Sichtweise – Ein Beispielobjekt

## *Feijoada*

### strukturelle Attribute

instance-of: Hauptgerichte

element-of: brasilianische Spezialitäten

has-components: schwarze Bohnen, Fleisch, Gewürze

Preis: 36

possible-values: integer > 0

cardinality: [1,1]

unit: DM

### deklarative Attribute

Vorbereitungszeit:

possible-values: integer > 0

cardinality: [1,1]

demon: Berechne-Vorbereitungszeit

geeignete-Getränke: trockener Rotwein, Bier

possible-values: instance-of Getränke

cardinality

.  
. .  
.

### prozedurale Attribute

Bestellen (Anzahl-von-Personen)

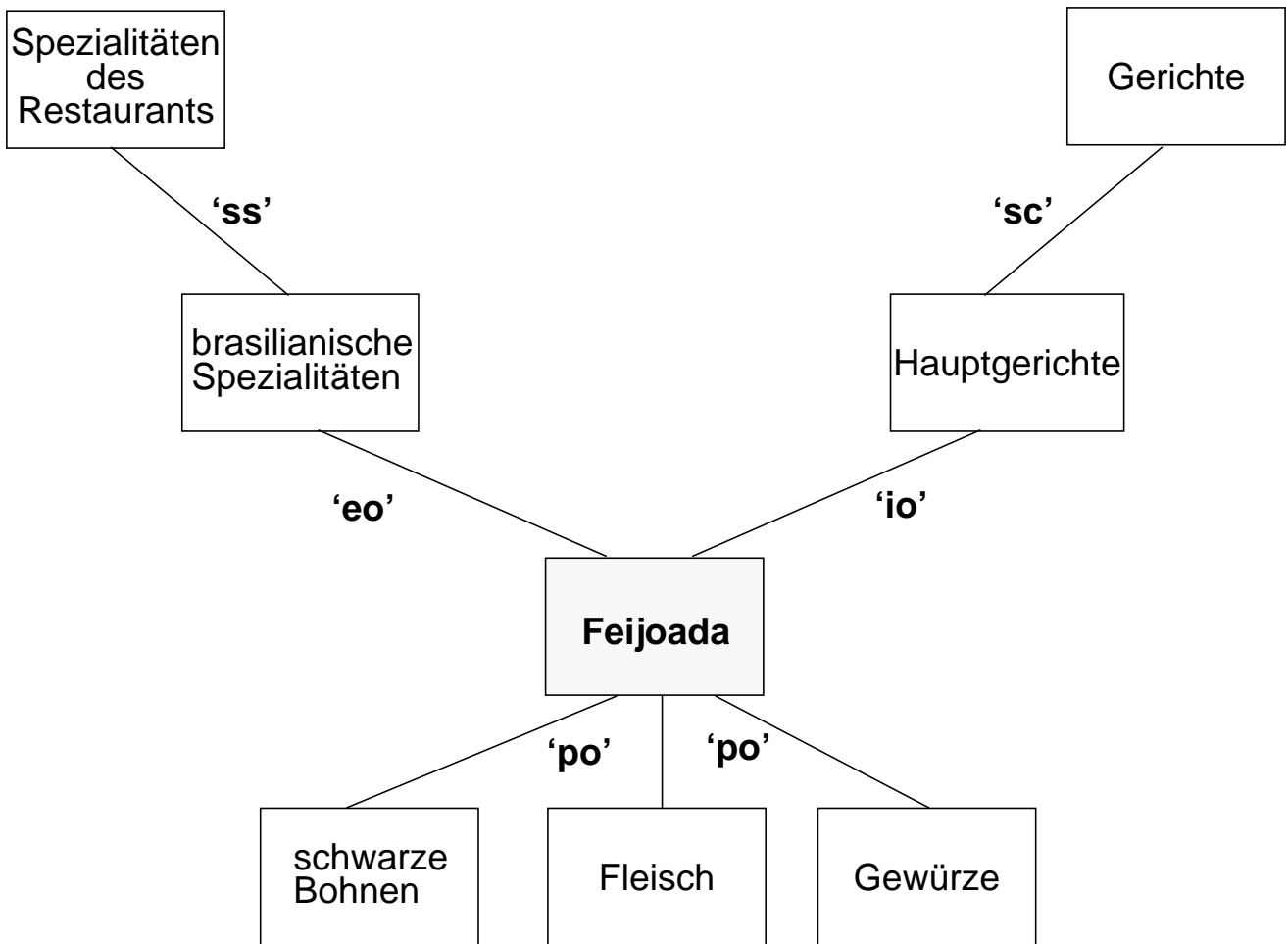
procedure BEGIN ... END

.  
. .  
.

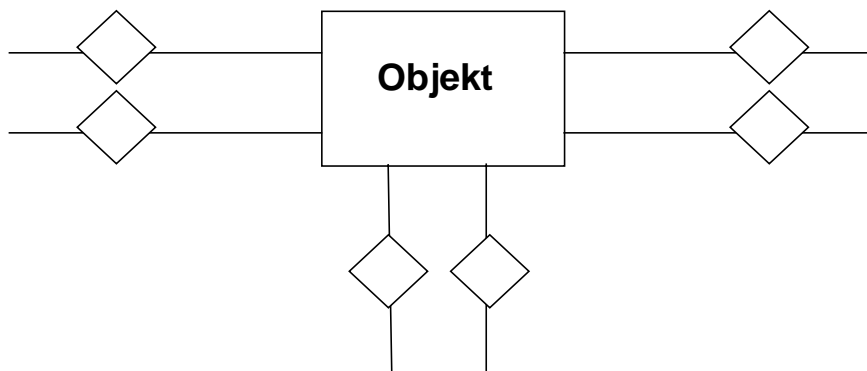
# Objektorientierte Repräsentation

- **Integration der Abstraktionskonzepte:**

- ein Objekt kann mehrere Beziehungstypen aufbauen
- entsprechend den versch. Rollen, die in den Abstraktionen vorkommen
- Objektsemantik wird bestimmt durch die Kontexte/Rollen eines Objektes



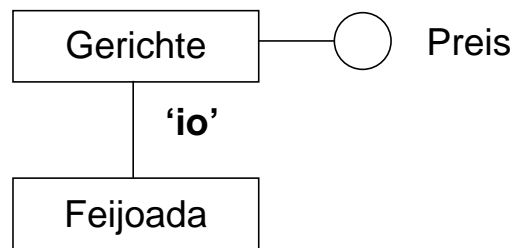
## Darstellungsprinzip:



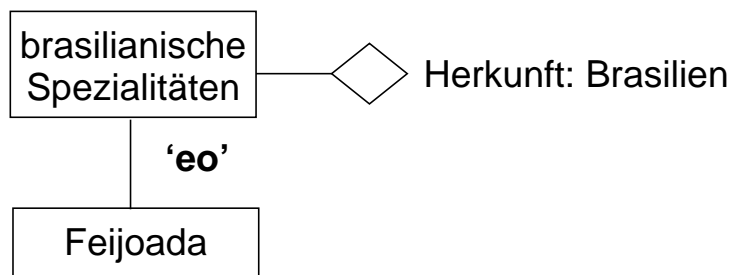
# Modellinhärentes „Reasoning“

- 3 Abstraktionskonzepte ermöglichen verschiedenartige Organisationsformen der modellierten Objekte und ihrer Beziehungen
- können für Schlußfolgerungen benutzt werden:
  - um Aussagen über Objekte und ihre Eigenschaften abzuleiten
  - als Zusatz bei Manipulations- und Retrievaloperationen

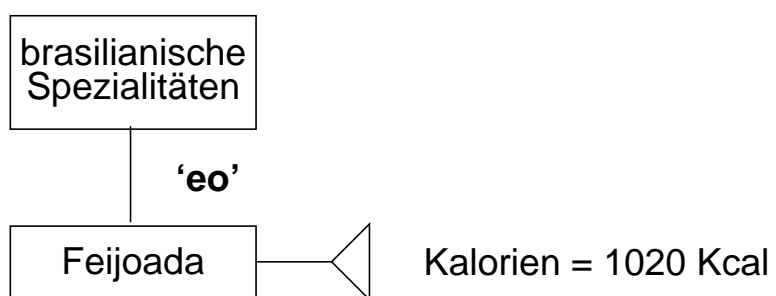
## Vererbung



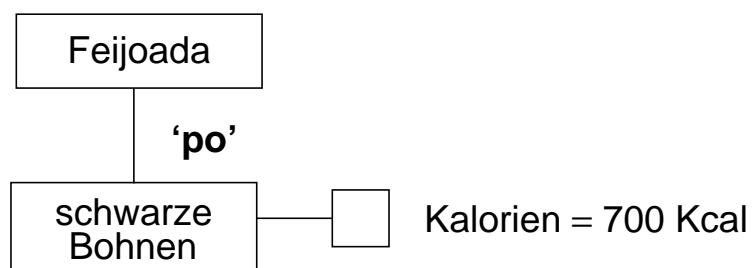
## Mitgliedschaftsimplikationen



## Mengen-eigenschaften



## implizierte Prädikate



# Zusammenfassung

- **DB-Entwurf umfaßt**

- Informationsbedarfsanalyse
- konzeptionelles DB-Schema (-> Informationsmodell)
- logisches DB-Schema
- physisches DB-Schema (nicht diskutiert)

- **ERM-Charakteristika**

- Modellierung bezieht sich auf die Typebene
- Relevante Zusammenhänge der Miniwelt werden durch Entity- und Relationship-Mengen modelliert. Sie werden genauer durch Attribute, Wertebereiche, Primärschlüssel/Schlüsselkandidaten beschrieben
- Klassifikation von Beziehungstypen dient der Spezifikation von strukturellen Integritätsbedingungen
- Anschauliche Entwurfsdarstellung durch ER-Diagramme
  - relativ karges Informationsmodell

- **Einführung weiterer Modellierungskonzepte**

- Verfeinerung von Beziehungen durch Kardinalitätsrestriktionen und vor allem Abstraktionskonzepte
- Das erweiterte ERM ist sehr mächtig und umfaßt viele bekannte Modellierungskonzepte (jedoch keine Rollen; sie lassen sich als Mehrklassen-Mitgliedschaften von Instanzen nachbilden)
- Integritätsbedingungen wurden hier nicht behandelt (-> Relationenmodell)

- **Abstraktionskonzepte und deren Implikationen**

- Generalisierung und Vererbung
- Assoziation mit Mengeneigenschaften und Mitgliedschaftsimplicationen
- Aggregation und implizierte Prädikate
- Integration der Abstraktionskonzepte mittels objektzentrierter Darstellung