3. Informations- und Datenmodelle

- **Vorgehensweise bei DB-Entwurf und -Modellierung**
  - Lebenszyklus
  - Informationserhebung
- **Entity-Relationship-Modell (ERM)**
  - Definitionen, Konzepte
  - Beziehungstypen
  - Diagrammdarstellung
  - Beispiele
- **Erweiterungen des ERM**
  - Schichtenmodell des ANSI/IRDS
  - Kardinalitätsrestriktionen
  - Abstraktionskonzepte
- **Abstraktionskonzepte**
  - Generalisierung / Spezialisierung
  - Spezialisierung bei Überlappungen
  - Assoziation
  - Aggregation
  - Integrierte Sichtweise

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**

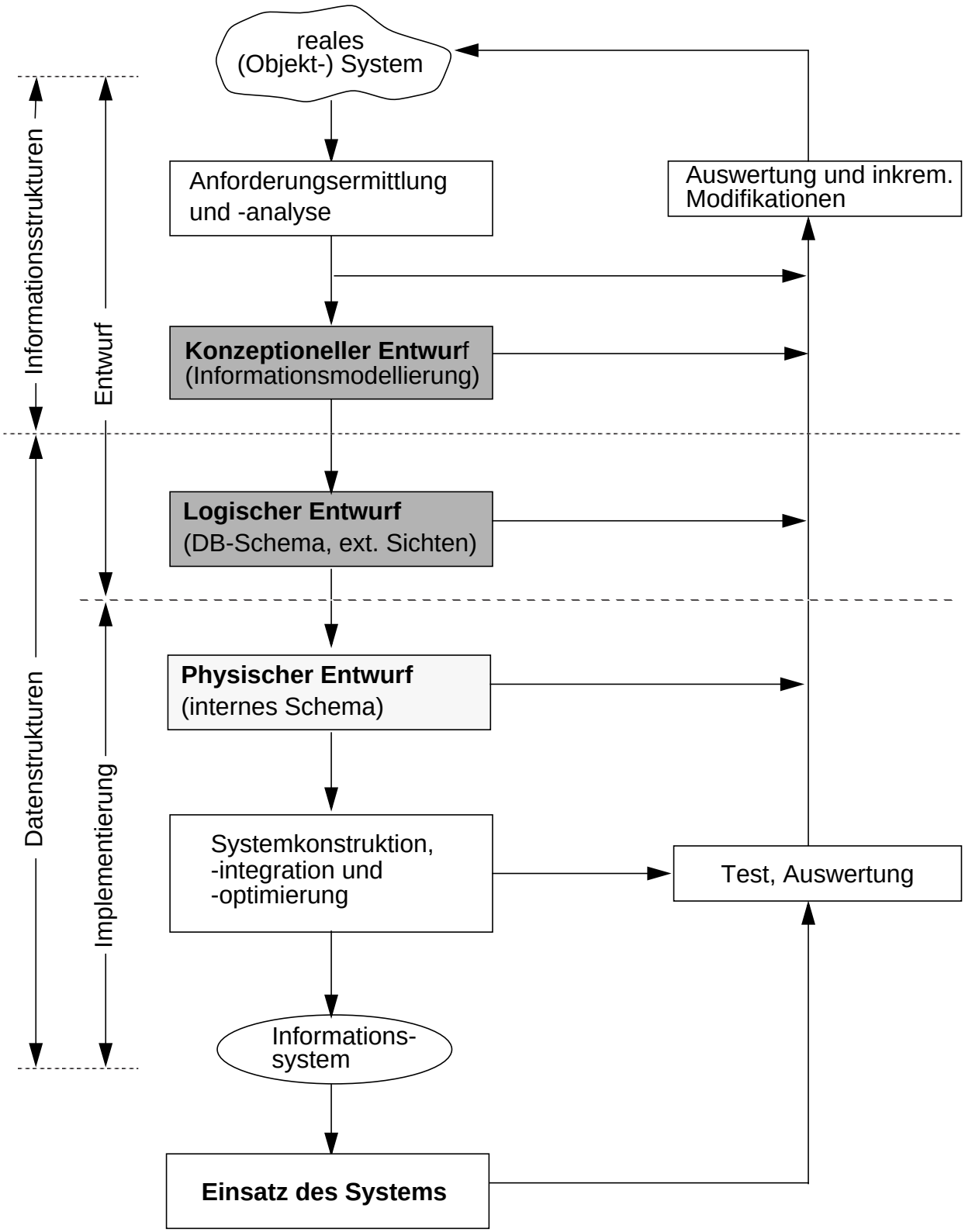
- **Nebenbedingungen:**
  - genaue Abbildung
  - hoher Grad an Aktualität
  - Verständlichkeit, Natürlichkeit, Einfachheit, ...

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

- **Bestandteile:**
  - Objekte: Entities
  - Beziehungen: Relationships

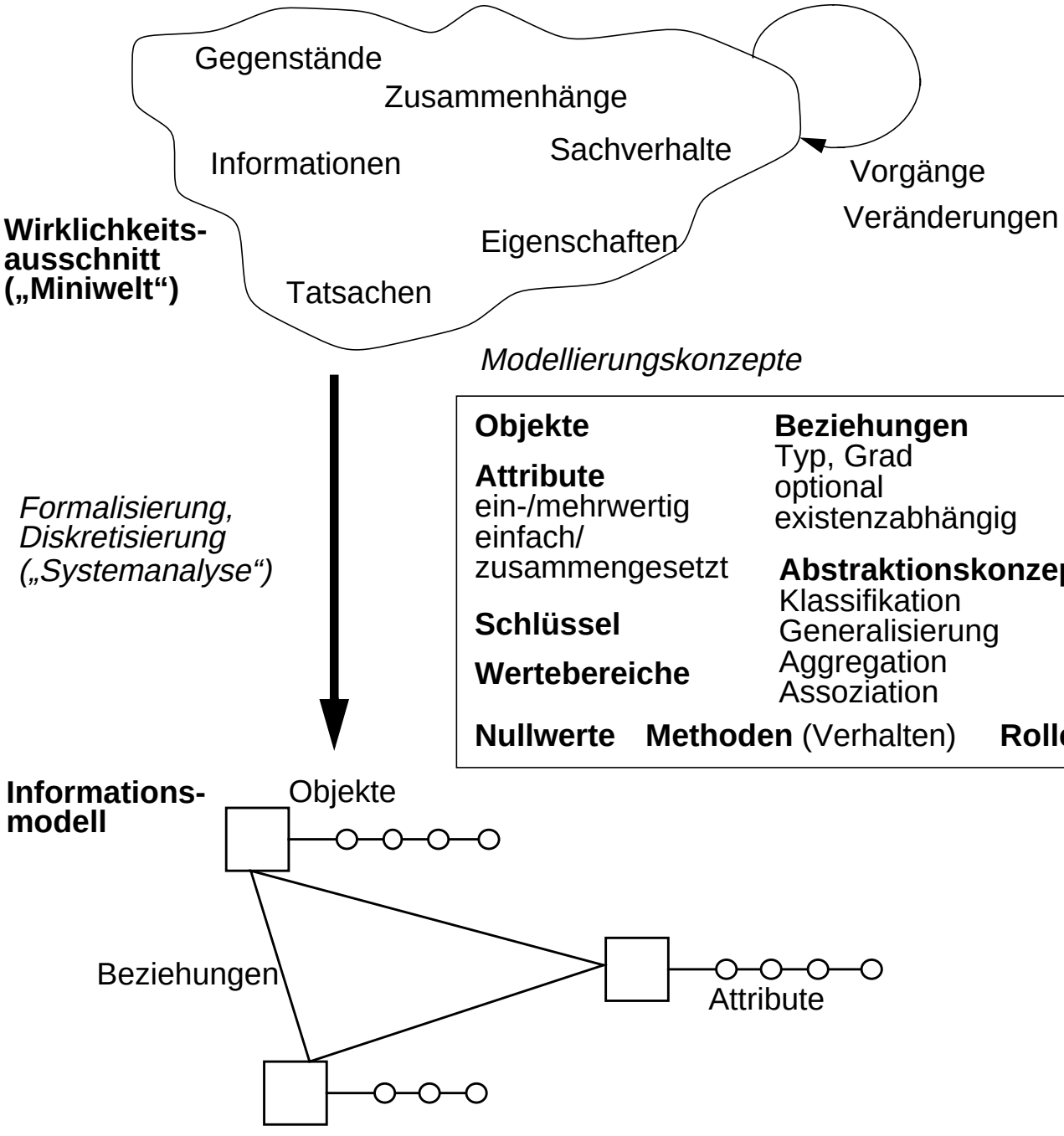
- **Schrittweise Ableitung: (Verschiedene Sichten)**
  1. Information in unserer Vorstellung
  2. Informationsstruktur: Organisationsform der Information
  3. Logische Datenstruktur (zugriffspfadunabhängig, Was-Aspekt)
  4. Physische Datenstruktur (zugriffspfadabhängig, Was- und Wie-Aspekt)

Schritte auf dem Weg zu einem Informationssystem



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

Informationsmodelle

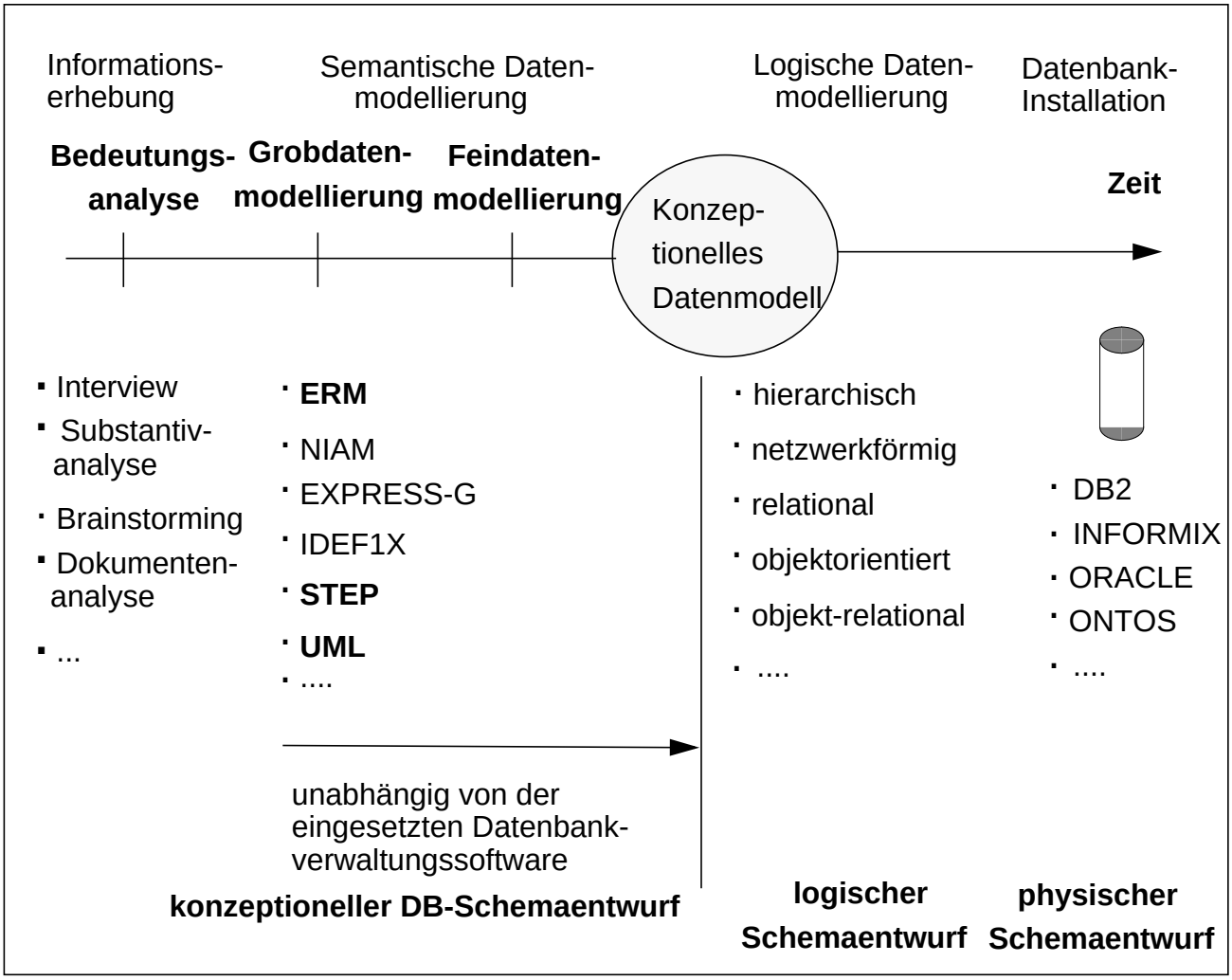


→ **Informationsmodell** (Darstellungselemente & Regeln)**:**
eine Art formale Sprache, um Informationen zu beschreiben

- **Informationen über Objekte und Beziehungen nur, wenn:**
  - unterscheidbar und identifizierbar
  - relevant
  - selektiv beschreibbar

Von der Informationserhebung zum DB-Schema

- **Prinzipielle Vorgehensweise**



- **ERM** (**E**ntity **R**elationship **M**odel)**:**
  - generell einsetzbares Modellierungswerkzeug

- **STEP** (**ST**andard for the **E**xchange of **P**roduct Definition Data)**:**
  - Modellierung, Zugriff, Austausch von **produktdefinierenden Daten** über den gesamten Produktlebenszyklus

- **UML (U**nified **M**odeling **L**anguage)**:**
  - Notation und Sprache zur Unterstützung objektorientierter Softwareentwicklung

Entity-Relationship-Modell (ERM) –[1]
Überblick

- **Modellierungskonzepte**
  - Entity-Mengen (Objektmengen)
  - Relationship-Mengen (Beziehungsmengen)
  - Wertebereiche
  - Attribute
  - Primärschlüssel

- **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

Konzept 1: **Entity-Mengen** (Objektmengen)

- **Entity:** „A thing that has real or individual existence in reality or in mind“ (Webster)

- Zusammenfassung aller Entities mit **gemeinsamen Eigenschaften**

  → Elemente einer Menge: $e \in E$

  z. B. Personen, Projekte ...
  Bücher, Autoren ...
  Kunden, Vertreter, Wein, Behälter

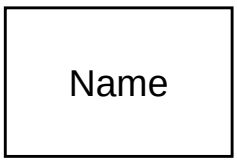
- **Zugehörigkeit** über Prädikat entscheidbar

  $e_i \in E_j \Leftrightarrow \text{is-}E_j\,(e_i)$

- DB enthält **endlich** viele Entity-Mengen
  $E_1, E_2, ..., E_n$; **nicht** notwendigerweise disjunkt

  z. B. E_1 ... Person,
  E_2 ... Student: $E_2 \subseteq E_1$

- **Symbol**:



Konzept 2: Relationship-Mengen

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

  z. B. „ist Hörer von“ zwischen „Student“ und „Vorlesung“

- **Eigenschaften**
  - Grad der Beziehung (*degree*)
  - Existenzabhängigkeit
  - Beziehungstyp (*connectivity*)
  - Kardinalität

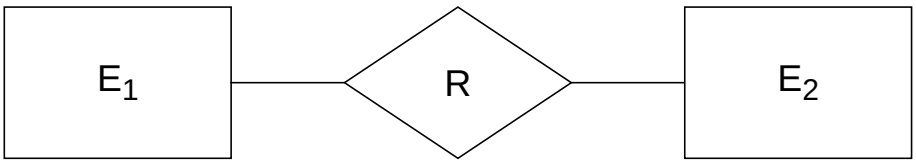
- R ... Relationship-Menge = math. Relation zwischen n E_i

  $R \subseteq E_1 \times E_2 \times ... \times E_n$

  d.h. $R = \{r = [e_1, e_2, ..., e_n] \mid e_1 \in E_1, ..., e_n \in E_n\}$

  Grad gewöhnlich: n=2 oder n=3

- **Symbol:**



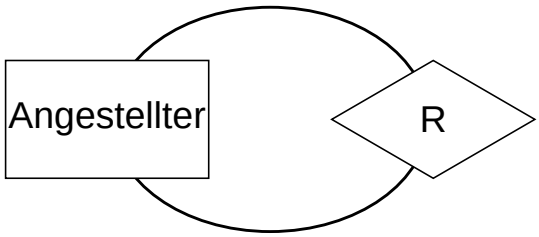
- **Eigenschaften**

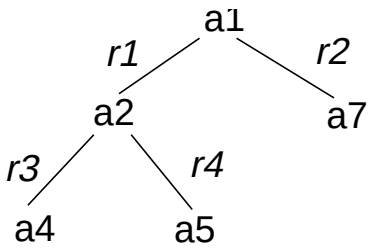
  Grad:
  Existenzabhängig:
  Beziehungstyp:

Relationship-Mengen (2)

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

  Direkter-Vorgesetzter =



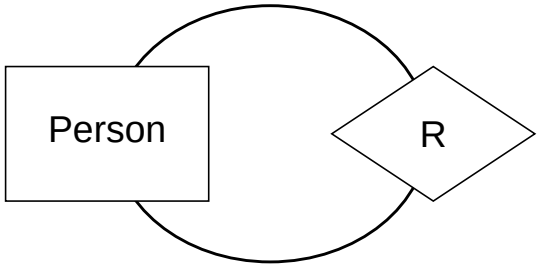


- **Eigenschaften**

  Grad:
  Existenzabhängig:
  Beziehungstyp:

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

- Relationship-Menge **Heirat**

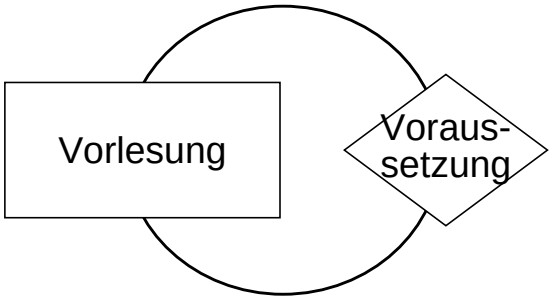


- **Eigenschaften**

  Grad:
  Existenzabhängig:
  Beziehungstyp:

Relationship-Mengen (3)

- **Motivation für Rollennamen**



- **Definition:**

  Voraussetzung $\subseteq$ Vorlesung x Vorlesung

  d. h. Voraussetzung $= \{ [v_i, v_j] \mid v_i, v_j \in$ Vorlesung $\}$

  genauer: direkte Voraussetzung

- Einführung von Rollennamen (rn) möglich (Reihenfolge!)

  z. B. $[rn_1/v_i, rn_2/v_j]$

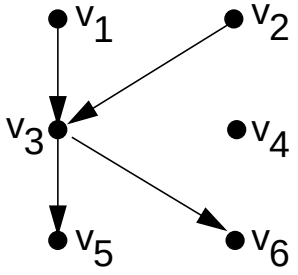
  [Vorgänger: v_i, Nachfolger: v_j]

  Sprechweise: „ v_j setzt v_i voraus“

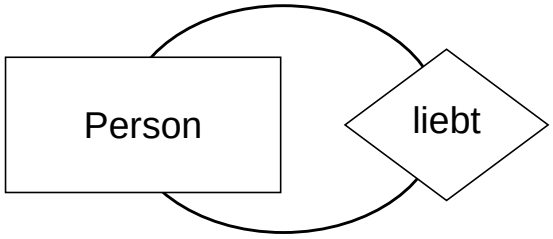
  oder „ v_i ist-Voraussetzung-für v_j“

- **Eigenschaften:**

Grad:	2
Existenzabhängig:	nein
Beziehungstyp:	n : m



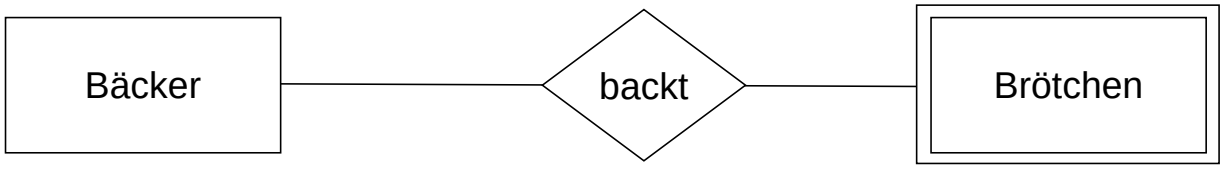
- **Transitivität gilt bei Selbstreferenz i. allg. nicht!**



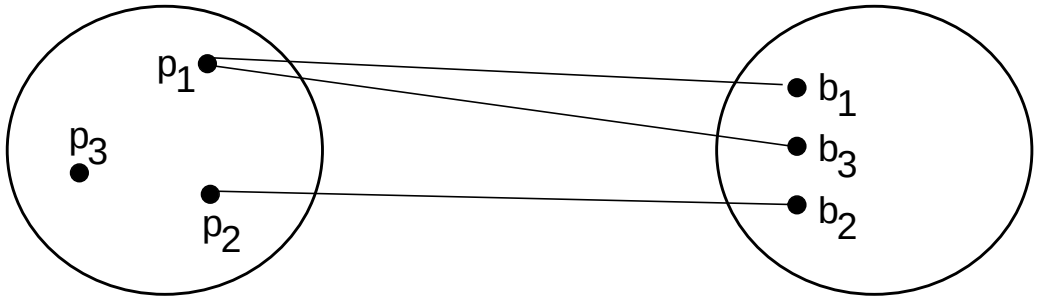
Relationship-Mengen (4)

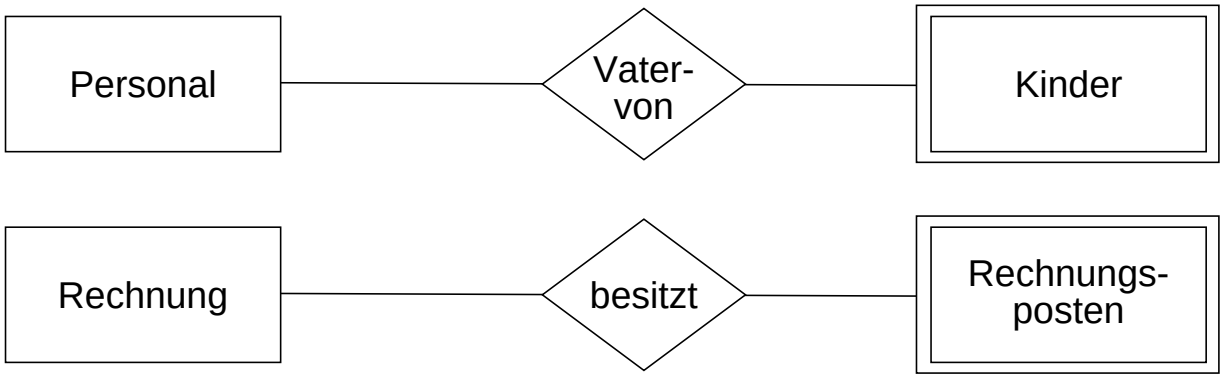
- **Existenzabhängigkeit einer Entity-Menge**

- **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

Konzept 3: Wertemengen

- Information über e_i oder r_i wird ausgedrückt durch Attribut-Wert-Paare
- Wertemengen W_i bestimmen Zulässigkeit konkreter Werte für e_i und r_i
- Definition durch Aufzählung, Prädikate ...

  z. B. PERSONALNUMMER =

  FAMILIENSTAND =

  NACHNAMEN =

- Definition von Abstrakten Datentypen (ADTs)

  z. B. DATE

  MONEY

  . . .

Konzept 4: Attribute

- Attribut A zu einer Entity-Menge E oder Relationship-Menge R:
  math. Funktion

  $A : E \rightarrow W$ bzw. $W_1 \times W_2 \times ... \times W_k$

  oder

  $A : R \rightarrow W$ bzw. $W_1 \times W_2 \times ... \times W_m$

- **einfache** vs. **zusammengesetzte** Attribute

  PNR: PERSONAL → PERSONALNUMMER

  NAME: PERSONAL → VORNAMEN x NACHNAMEN

  ANSCHRIFT: PERSONAL → PLZ x STADT x STRASSE x HSNR

- **einwertige** (*single-valued*) vs. **mehrwertige** (*multivalued*) Attribute

  AUTOFARBE:

  KINDER:

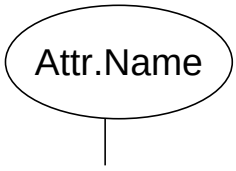
- **Nullwert**: Attributwert nicht möglich bzw. unbekannt
  A(e) = {} (z. B. private Tel. Nr.)

- Wertemengen W_i nicht notwendig verschieden

- **<u>Auch</u> Relationship-Mengen können Attribute besitzen**

  z. B. ARBEITSZEITANTEIL: PROJEKTMITARBEIT → PROZENT

- **Attributsymbol in ER-Diagrammen:**



Konzept 5: Primärschlüssel (*Entity Key*)

- 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.)
- $\{A_1, A_2, ..., A_m\}$ = **A** sei Menge der Attribute zur Entity-Menge

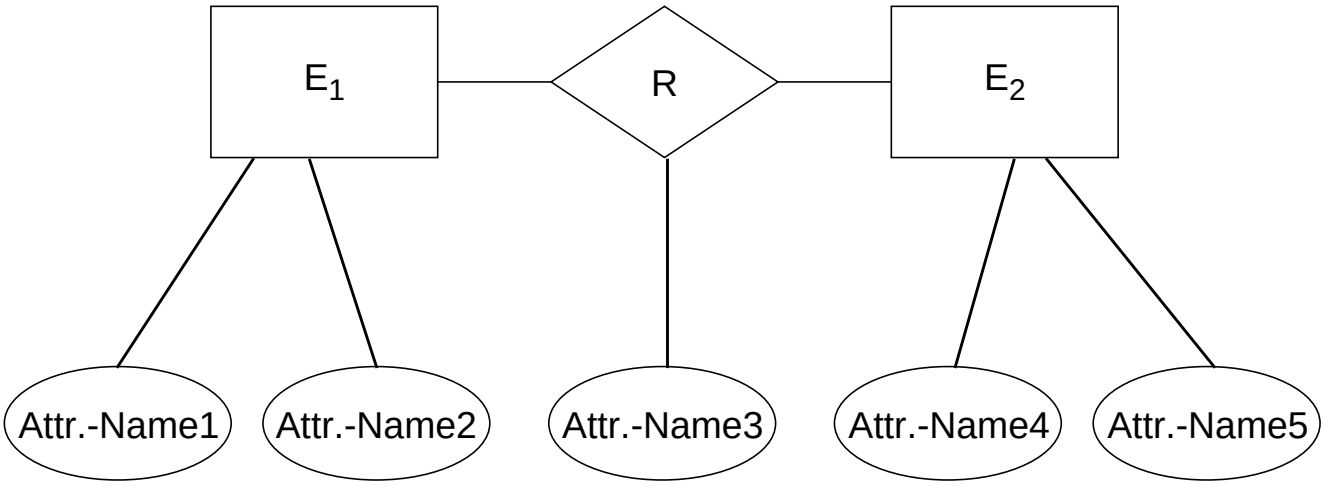
  K $\subseteq$ **A** heißt Schlüsselkandidat von E

  $\Leftrightarrow$ **K** minimal; $e_i, e_j \in E$;

  $e_i \neq e_j \rightarrow$ **K**$(e_i) \neq$ **K**(e_j)
- mehrere Schlüsselkandidaten möglich

  → **Primärschlüssel** auswählen

- Darstellung von Attributen im ER-Diagramm



Repräsentation der Miniwelt durch Werte in (regulären) Relationen

Reguläre Entity-Relation PERSONAL

Primär-schlüssel

Attribute	PNR	NAME		KÜNSTLERNAME		BERUF
Werte-Mengen	PERSONAL-NUMMER	VOR-NAMEN	NACH-NAMEN	VOR-NAMEN	NACH-NAMEN	BERUFE
Entity-Tupel	1234	PETER	MAIER	HUGO	BUG-FIGHTER	PROGRAM-MIERER
	5678	OTTO	ABEL	ERNIE	DEAD-LOCK	SYSTEM-ANALYTIKER
	⋮	⋮	⋮	⋮	⋮	⋮

Reguläre Relationship-Relation PROJEKTMITARBEIT

	← Primärschlüssel →			
Entity Relations-Name	PERSONAL	PROJEKT		
Rolle	MITARBEITER	PROJEKT		
Entity-Attribute	PNR	PRO-NR	ARBEITSZEIT-ANTEIL	DAUER
Werte-Mengen	PERSONAL-NUMMER	PROJEKT-NUMMER	PROZENT	ANZ.-MONATE
Relationship-Tupel	1234	12	30	6
	5678	78	50	12
	⋮	⋮	⋮	⋮

Repräsentation der Miniwelt durch Werte in (schwachen) Relationen

Beispiel:
Schwache Entity-Relation KINDER

Primärschlüssel

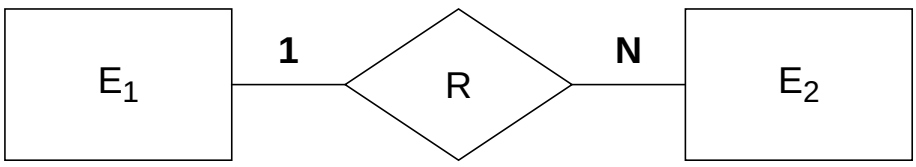
Entity-Relation-Name	PERSONAL		
Rolle	VATER-VON		
Entity-Attribute	PNR	NAME	ALTER
Werte-Menge	PERSONAL-NUMMER	VOR-NAMEN	ANZ.-JAHRE
	1234	JULIA	7
	5678	JULIA	7
	1234	PETER	2
	⋮	⋮	⋮

- Entity-Menge **ohne eigene Schlüsselkandidaten**
- Identifikation über Beziehung zu übergeordneter Entity-Menge (=> Existenzabhängigkeit!)
- Hinzunahme der Primärschlüsselattribute der „Vater“-Entity-Menge bei schwacher Entity-Menge
- **ER-Symbol:** 

Klassifikation von Datenabbildungen

- **ZIEL:**
  - Festlegung von semantischen Aspekten
  - explizite Definition von strukturellen Integritätsbedingungen
  - hier: Beziehungstyp (*connectivity*)
- **Unterscheidung von Beziehungstypen**
  - E_i - E_j
  - E_i - E_i
  - A_i - A_k (interne Klassifikation)
- **Festlegung der Abbildungstypen**
  - 1:1 ... eineindeutige 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
  - n:1 ... ARBEITET_FÜR/MIT: MITARBEITER $\rightarrow$ PROF
  - n:m ... BESCHÄFTIGT/IST_HIWI: PROF — STUDENT

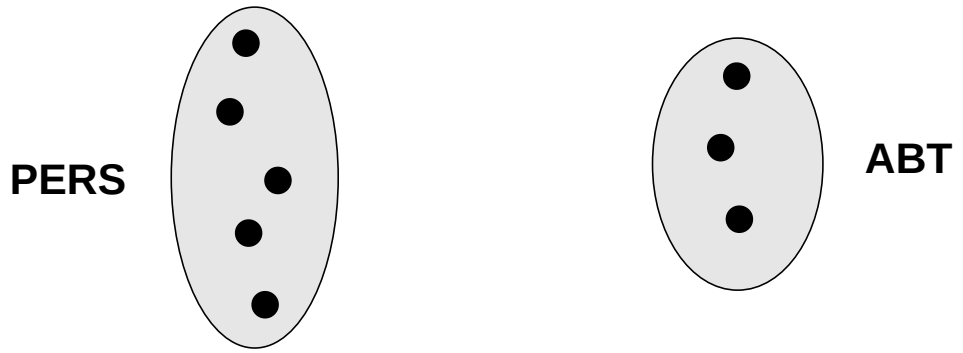
  → Abbildungstypen implizieren nicht, daß für jedes $e_k \in E_i$ auch tatsächlich ein $e_l \in E_j$ existiert
- Diagrammdarstellung:



Klassifikation von Datenabbildungen (2)

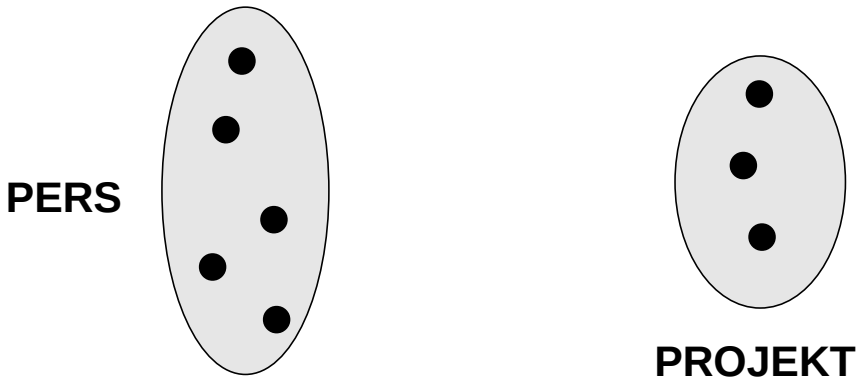
- **Beispiele zu E_i – E_j (externe Klassifikation)**

  - 1:1: LEITET/WIRD_GELEITET: PERS **<—>** ABT



  - n:1/1:n: ARBEITET_FÜR/HAT_MITARBEITER: PERS **—>** ABT

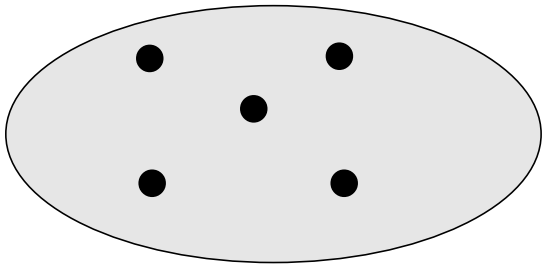
  - n:m: ARBEITET_FÜR/MITARBEIT: PERS **—** PROJEKT



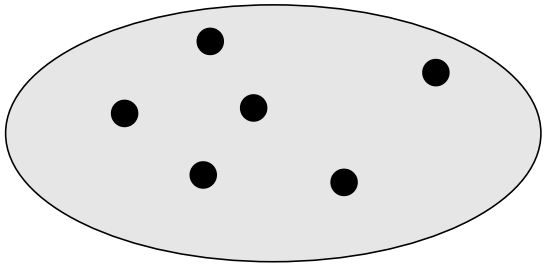
Klassifikation von Datenabbildungen (3)

- **Beispiele zu E_i – E_i (externe Klassifikation)**

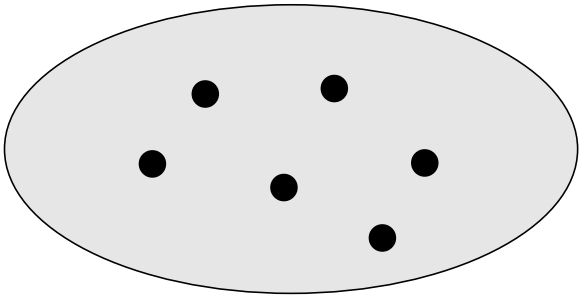
  - 1:1 VERHEIRATET MIT: PERS **<—>** PERS



  - n:1/1:n UNTERGEB./VORGESETZTER_VON: PERS **—>** PERS



  - n:m ... SETZT_SICH_ZUSAMMEN_AUS/: TEIL **—** TEIL
    GEHT_EIN_IN



Klassifikation von Datenabbildungen (4)

- **Klassifikation der Abbildungstypen für die Attribute einer Entity-Menge (E_i – W_k) – Interne Klassifikation**

  - 1:1 ... zwischen Schlüsselkandidaten: SK_i **<—>** SK_j

    Bsp. (STUDENT): MATNR **<—>** PERS-KENNZIFFER

  - n:1 ... zwischen SK und Attributen: SK_i **—>** A_k

    Bsp. (STUDENT): MATNR —> IMMAT-DATUM

  - n:m ... zwischen Schlüsselattributen: SA_i — SA_j

    Bsp. (STUDENT): NAME — GEBDAT (NAME, GEBDAT sei SK)

ER-Schema – Beispiel

DECLARE	VALUE-SETS	REPRESENTATION	ALLOWABLE-VALUES
	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

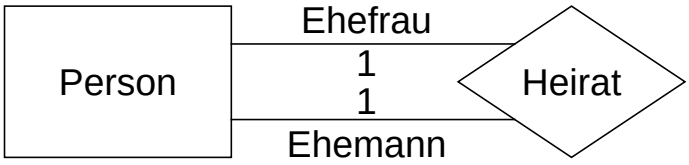
DECLARE RELATIONSHIP RELATION PROJEKT-MITARBEIT
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES
MITARBEITER/PERSONAL.PK/n
PROJEKT /PROJEKT.PK/m
ATTRIBUTE/VALUE-SET:
ARBEITSZEITANTEIL/PROZENT
DAUER/ANZ.-MONATE

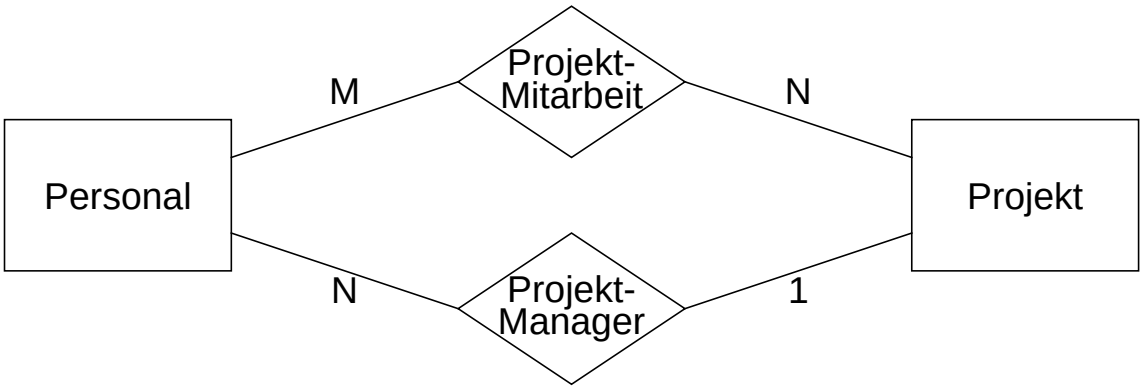
DECLARE RELATIONSHIP RELATION PERS.-ANGEHÖRIGE
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES
UNTERHALTSPFLICHTIGER/PERSONAL.PK/1
KIND/ KINDER.PK/n
EXISTENCE OF KIND DEPENDS ON
EXISTENCE OF UNTERHALTSPFLICHTIGER

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

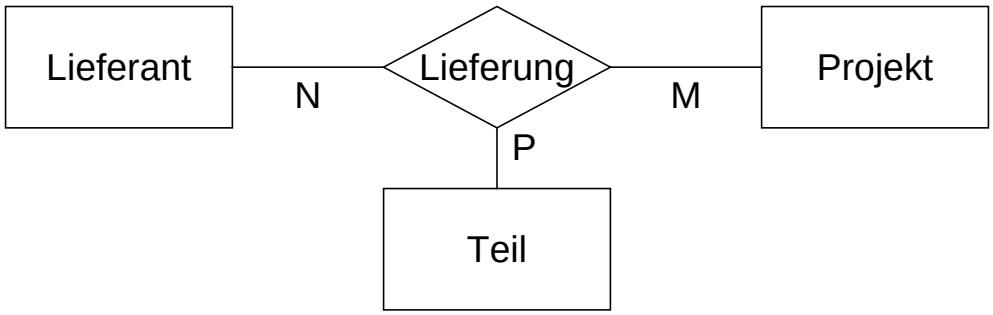
ER-Diagramme (Beispiele)

Entity-Menge Relationship-Menge Entity-Menge



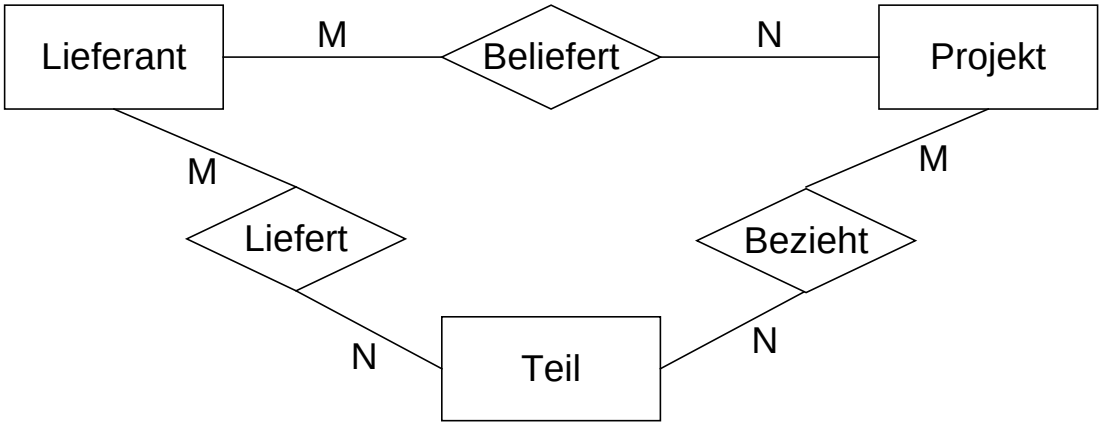


Dreistellige Relationship-Menge



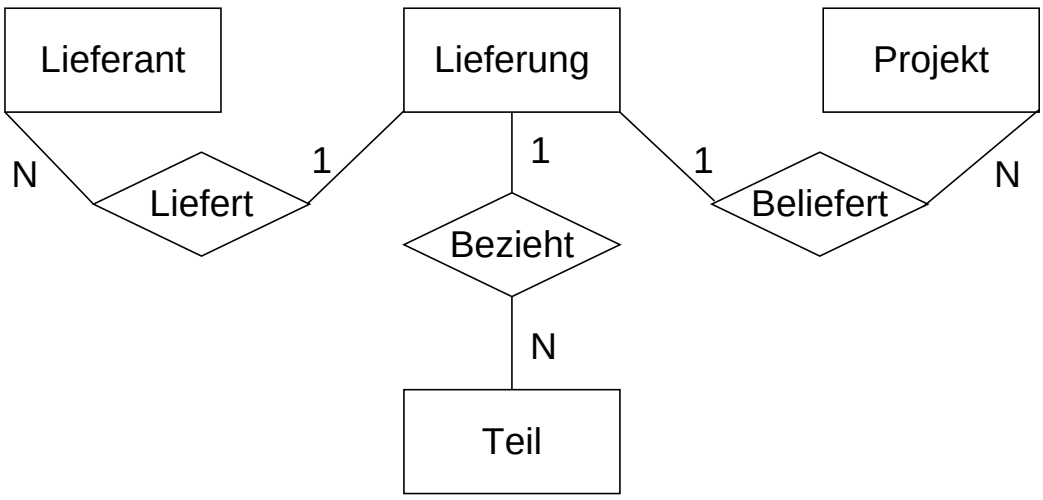
Achtung:

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

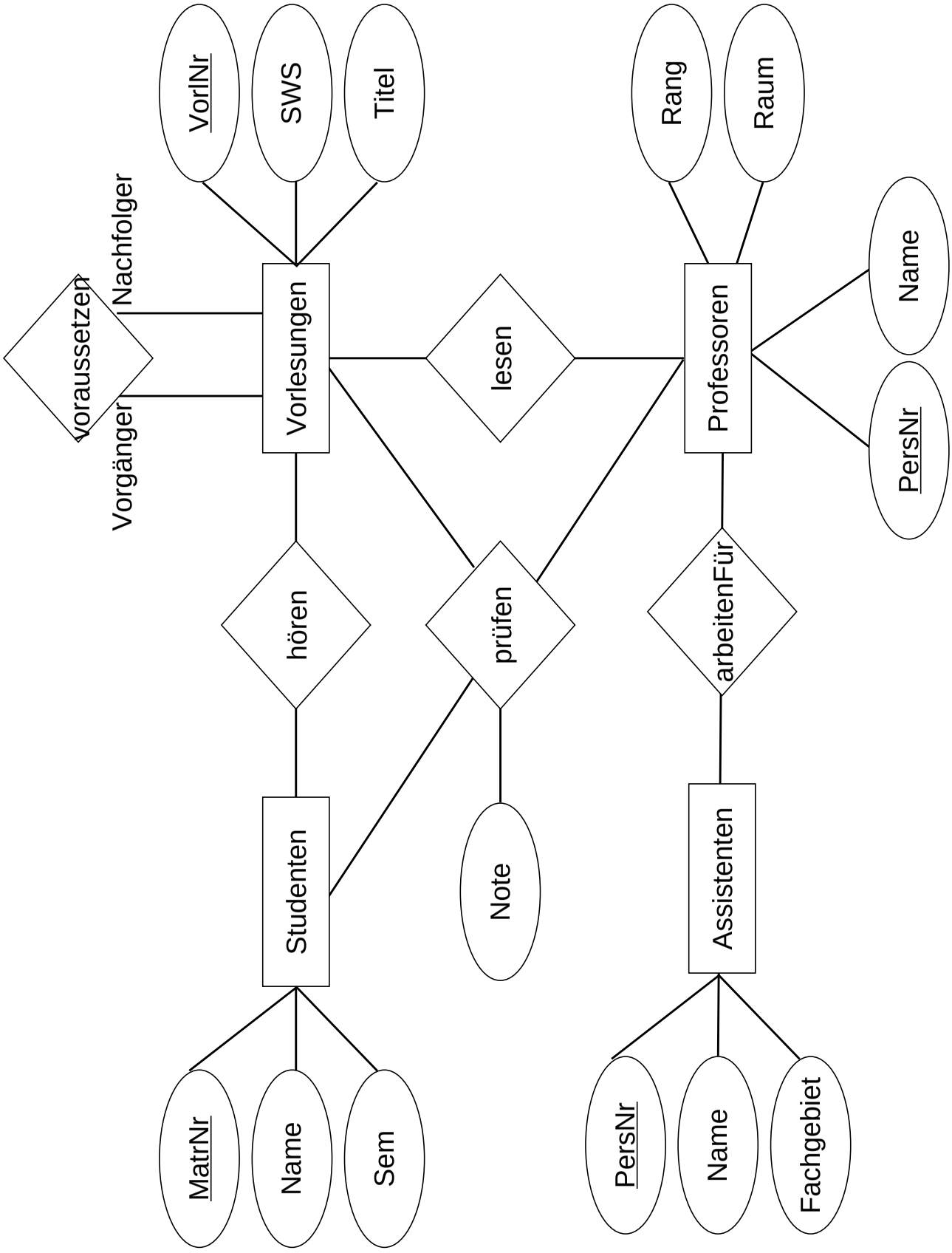


Aber:

manche Systeme erlauben nur die Modellierung binärer Relationship-Mengen!

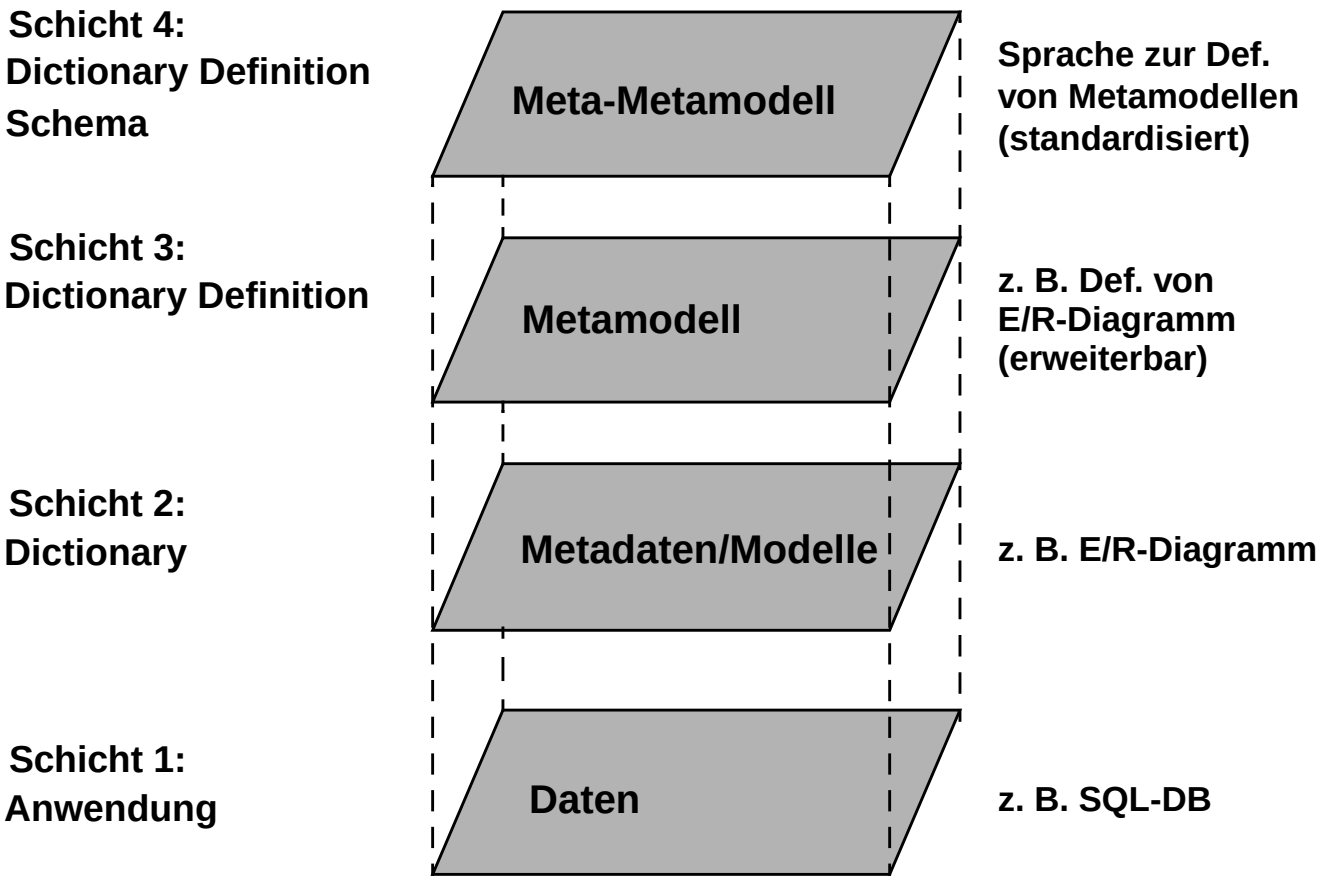


ER-Diagramm - Vorlesungsbetrieb



Erweiterungen des ERM

- **Basismodell:**
  - Festlegung des Beziehungstyps zwischen EM
  - nur benutzerdefinierte Beziehungen
  - nur Typebene dargestellt
  - unangemessene Modellierung bei überlappenden EM
- **Was bedeutet „Modellerweiterung“?**
- **Schichtenmodell des ANSI-IRDS[1]**



1. ANSI: American National Standards Institute
   IRDS: Information Repository Definition Standard

Schicht 1: Tabellen, Daten

Tabelle: KUNDE

KNR	NAME	ANSCHRIFT	...
1234	BAYER	KL ...	...
5678	SCHILCHER	SB...	...
6780	MITSCHANG	S ...	...

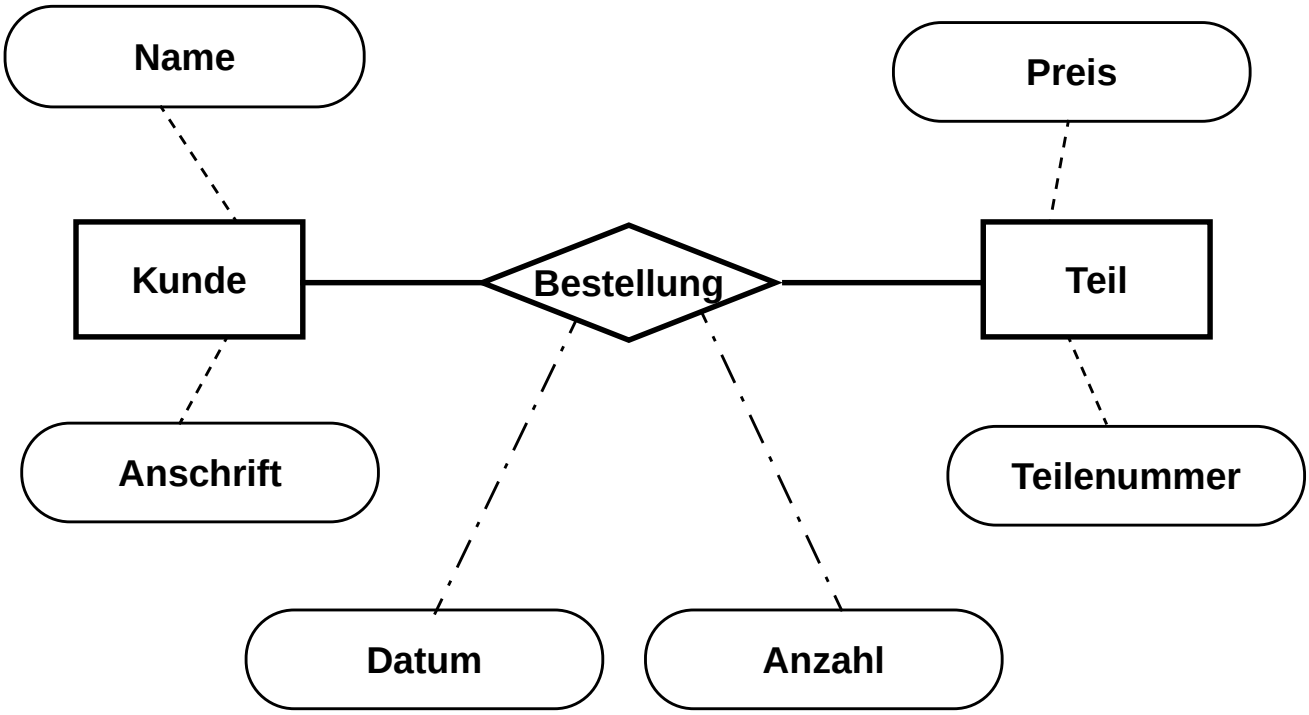
Tabelle: TEIL

TNR	PREIS	...
123 766	123.00	...
130 680	436.78	...
196 481	97.49	...

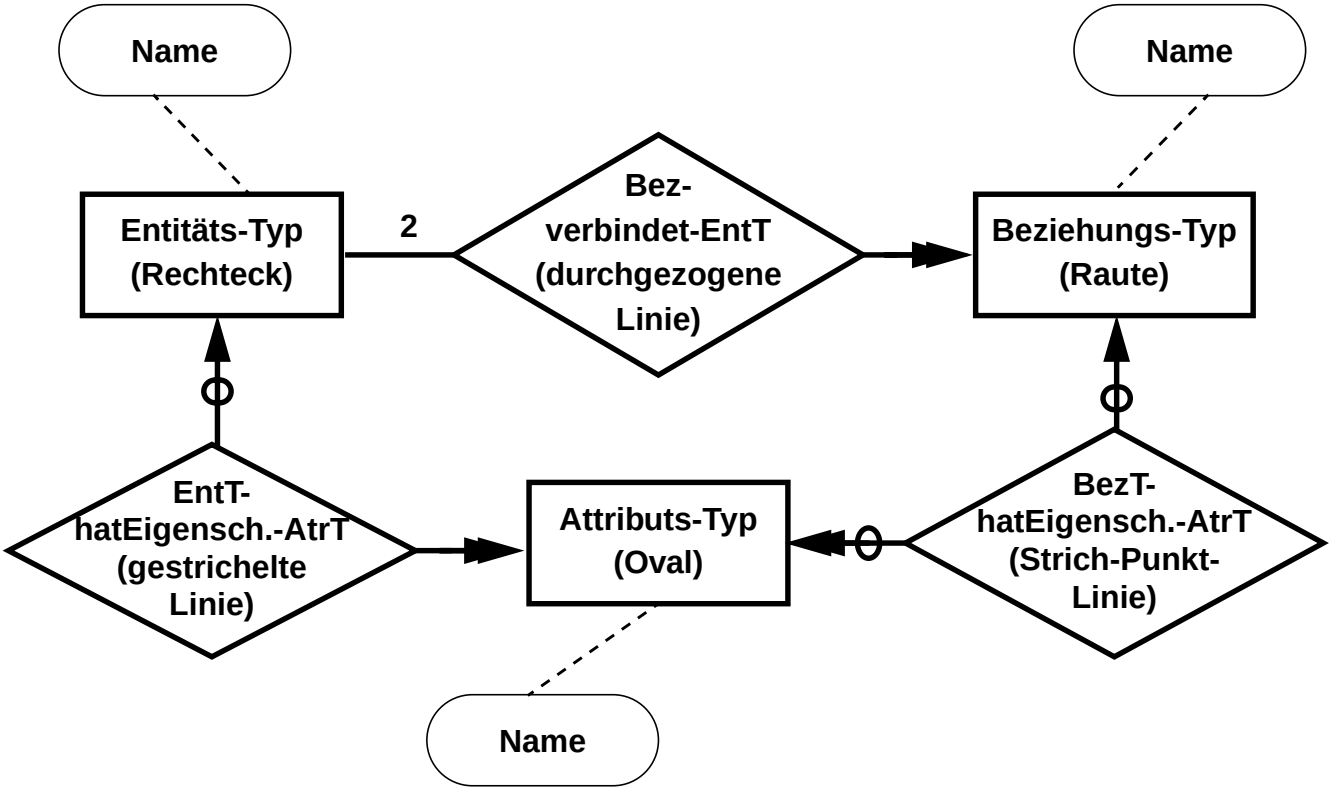
Tabelle: BESTELLUNG

PNR	TNR	ANZAHL	DATUM	...
1234	123 766	1000	1.1.87	...
1234	196 481	1500	2.7.95	...
5678	123 766	5000	4.9.93	...
5678	130 680	500	12.12.96	...
6780	130 680	3000	2.4.89	...
6780	196 481	3000	23.8.97	...

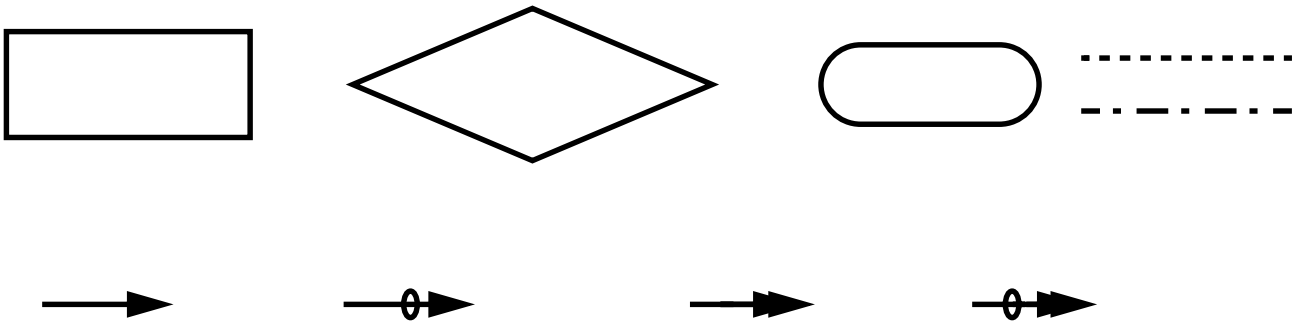
Schicht 2: ER-Diagramm



Schicht 3: Metamodell



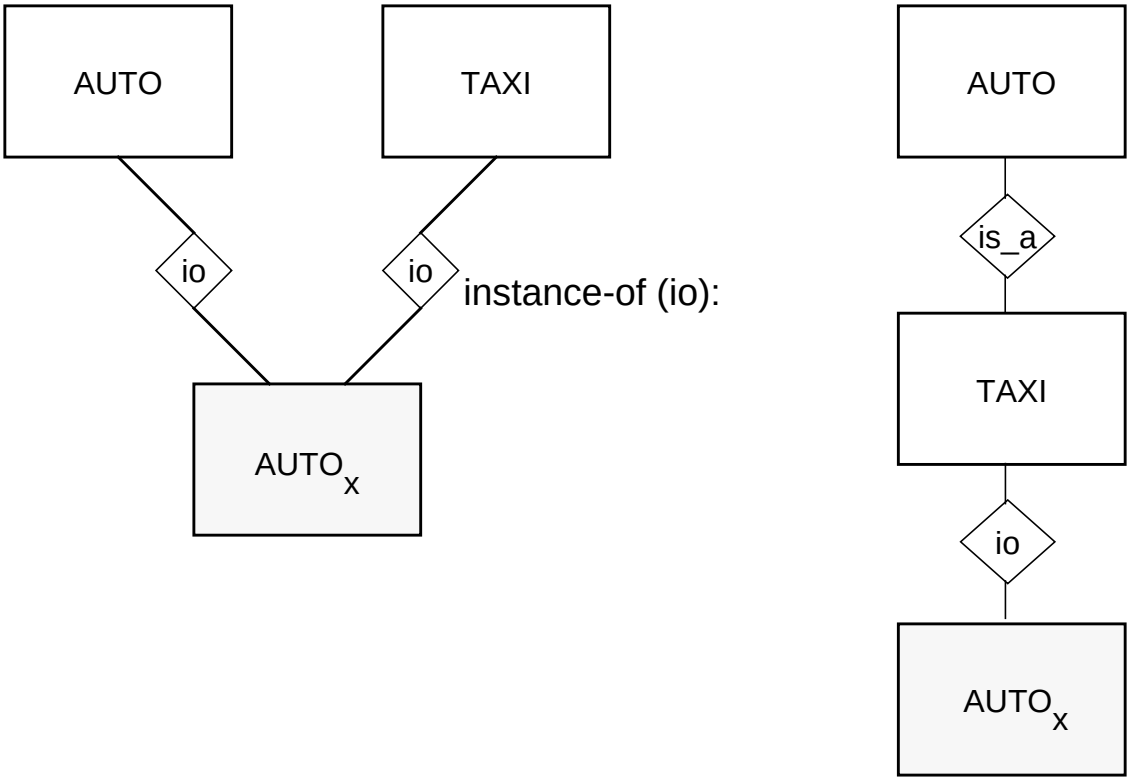
Schicht 4: Meta-Metamodell



Erweiterungen des ERM (2)

- **„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**)

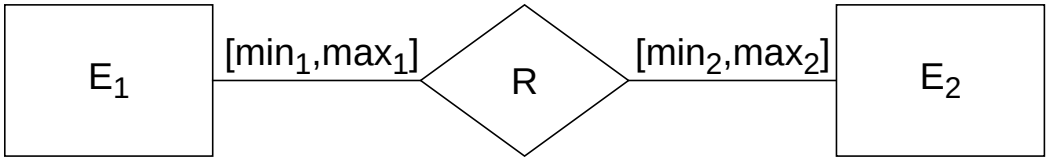
Verfeinerung der Datenabbildung: Kardinalitätsrestriktionen

- bisher: grobe strukturelle Festlegung der Beziehungen
  z. B.: 1:1 bedeutet „höchstens eins zu höchstens eins“

- Verfeinerung der Semantik eines Beziehungstyps durch Kardinalitäts-restriktionen:
  sei $R \subseteq E_1 \times E_2 \times ... \times E_n$
  Kardinalitätsrestriktion $kard(R,E_i) = [min,max]$
  bedeutet, daß jedes Element aus E_i in wenigstens min und höchstens max Ausprägungen von R enthalten sein muß (mit 0 <= min <= max, max >= 1).

- **graphische Darstellung**

Kardinalitäts-restriktionen:



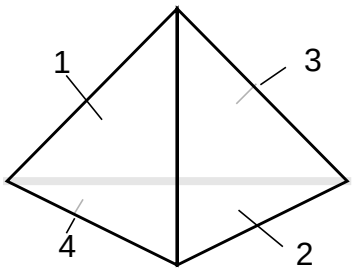
e_1 nimmt an $[min_1,max_1]$ Beziehungen von Typ R teil
e_2 nimmt an $[min_2,max_2]$ Beziehungen von Typ R teil

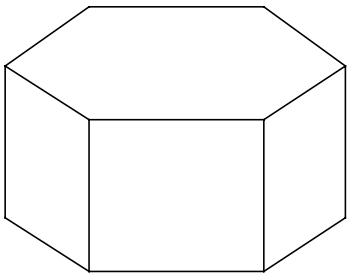
- **Beispiele**:

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

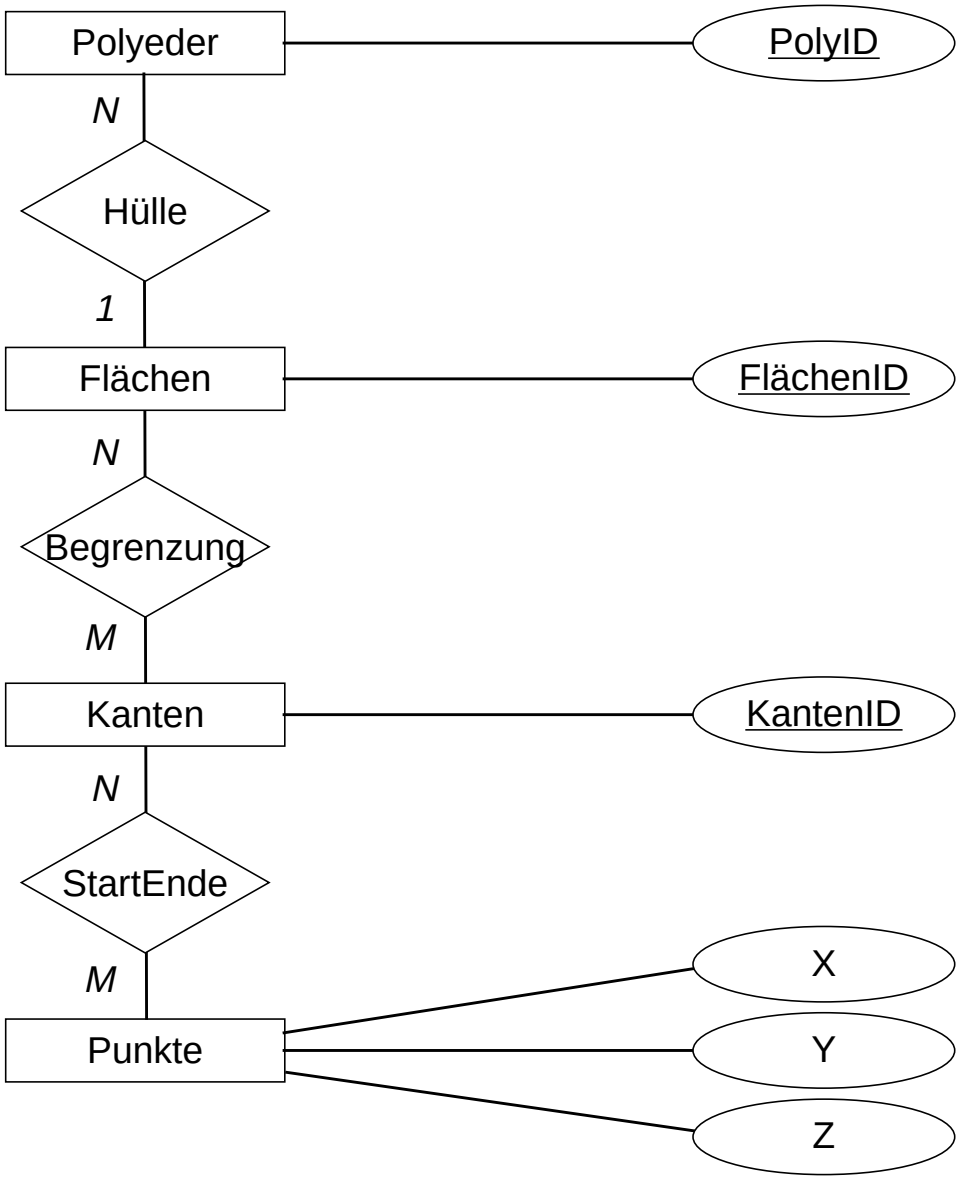
Begrenzungsflächendarstellung von Körpern

Beispiel-Körper:



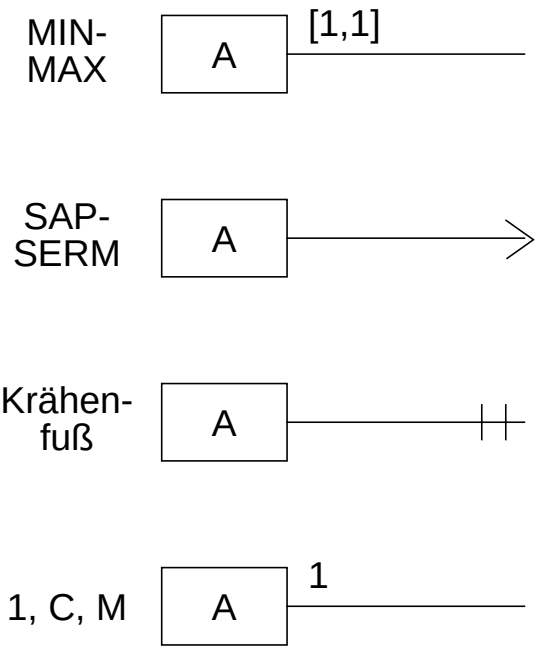


ER-Diagramm:

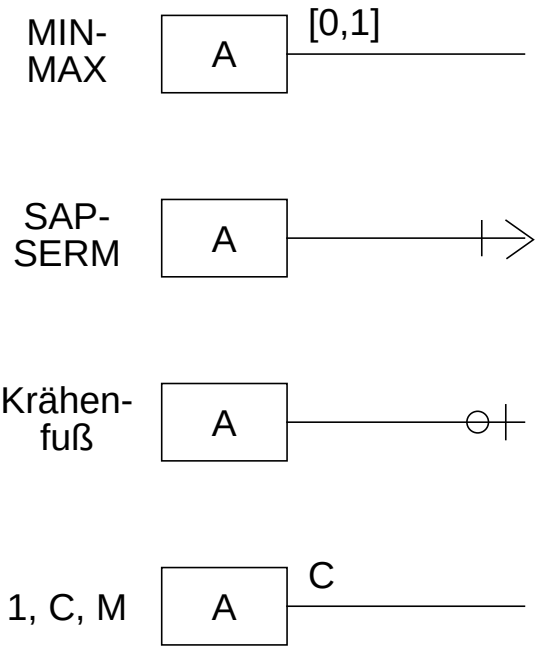


Notationen

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

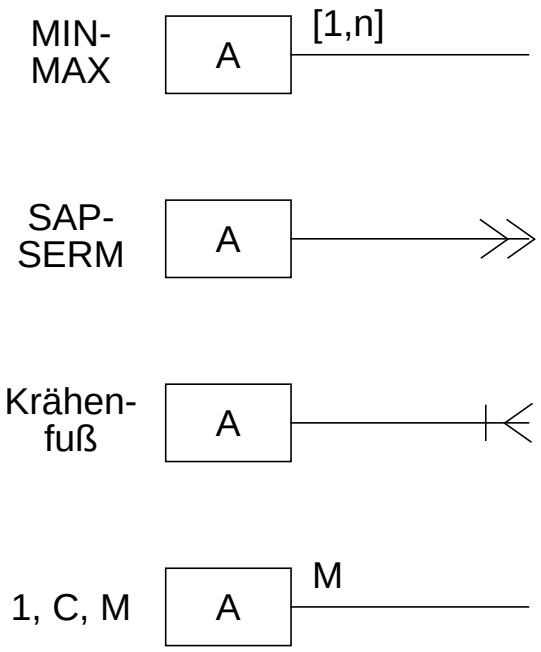


- **Jedes Element von A nimmt an höchstens einer Beziehung teil**

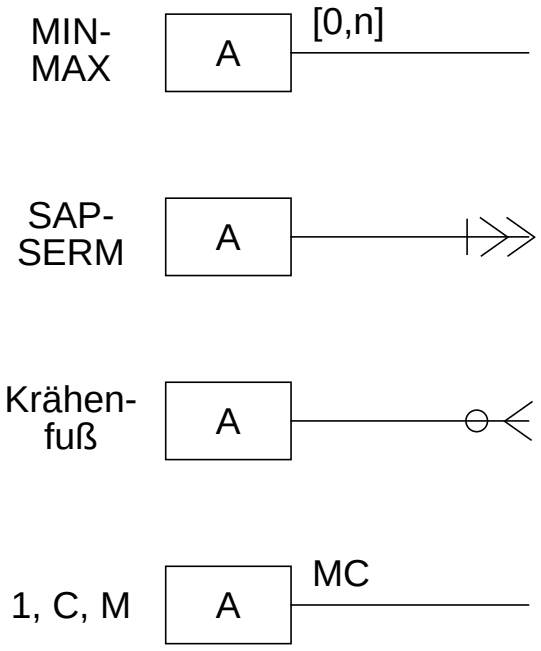


Notationen (2)

- **Jedes Element von A nimmt an mindestens einer Beziehung teil**



- **Jedes Element von A kann an beliebig vielen Beziehungen teilnehmen**



Abstraktionskonzepte[1]

- **Ziel:**
  - Erfassung von noch mehr Semantik aus der Miniwelt durch das (erweiterte) ERM
  - Entwicklung von (Beschreibungs-)Modellen, die eine weitgehend direkte Wiedergabe unseres „natürlichen“ Verständnisses der ausgewählten Miniwelt (Diskursbereich) erlauben.

- **Aufgabe:**
  - Identifikation von wesentlichen Konstrukten, die vom Menschen angewendet werden, wenn er seinen Diskursbereich beschreibt.

    → Anwendung von ***Abstraktion***, um die Information zu organisieren

also:

abstraction permits someone to suppress specific details of particular objects emphasizing those pertinent to the actual view

- **Kandidaten für eine verfeinerte Semantik:**
  - konzeptionelle **Objekte** mit Eigenschaften (Attribute)
  - **semantischere Beziehungen** zwischen den Objekten
  - **Operationen**/Aktivitäten auf diesen Objekten
  - **Integritätsbedingungen**, die diese Konzepte genauer beschreiben (um inkonsistente Zustände erkennen zu können):

  hier vor allem:

  - gleichrangige Behandlung von Entity und Entity-Menge
  - Definition von systemkontrollierten Beziehungen

---

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

Abstraktionskonzepte (2)

- **Auswirkungen**
  - Adäquate Modellierung: 1:1-Zuordnung zwischen Miniwelt- und Modell-Objekt
  - Alle relevanten Dinge in unserer Miniwelt sind Objekte des ERM (auch die Objekte, die wiederum andere Objekte beschreiben)
    - → Entity-Mengen (Klassen) und Entities (Instanzen) sind eigenständige Objekte der Modellierung
    - → Operationen unterscheiden nicht zwischen Klassen und Instanzen

- **Objekt-Eigenschaften**

  Objekte sind entweder
  - einfach, d. h. durch sich selbst vollständig definiert oder
  - zusammengesetzt, d. h. als „Abstraktion“ von anderen Objekten beschrieben

- **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

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

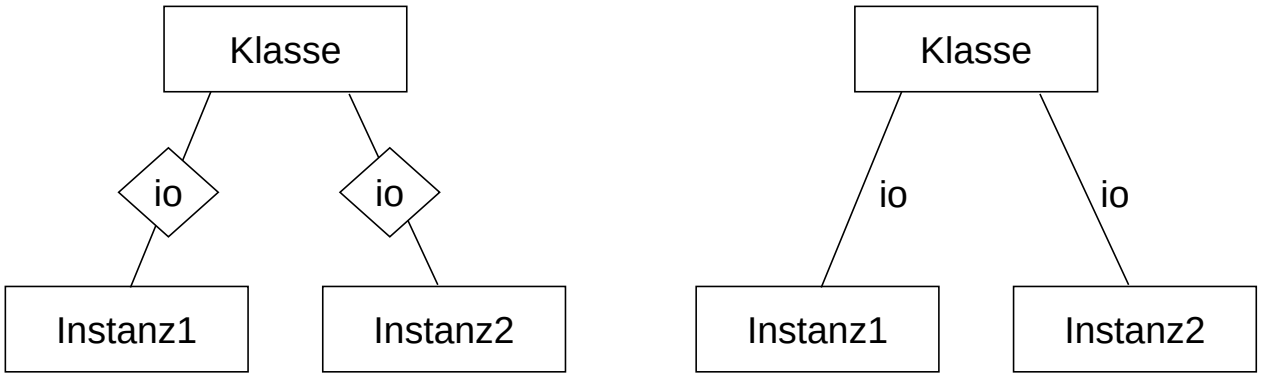
- Es wird eine **'instance-of'**-Beziehung (**‘io’**) als 1-Ebenen-Beziehung aufgebaut

- **Klasse kann selbst auch als Objekt betrachtet werden:**
  - Attribute: Definition ihrer Objekte,
    Verweise auf ihre Objekte
  - Methoden:
    „erzeuge neues Objekt“
    „suche Objekt“
  - Klasse verwaltet auch die Menge ihrer Objekte
  - Objekte einer Klasse heißen **Instanzen** dieser Klasse

    (Klasse selbst kann als Instanz einer Klasse „Klasse“ aufgefaßt werden ...)

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**
- Alternative: Einführung von klassenlosen Objekten
  (direkte Typzuordnung var X: Typ Y)
- **graphische Darstellung**



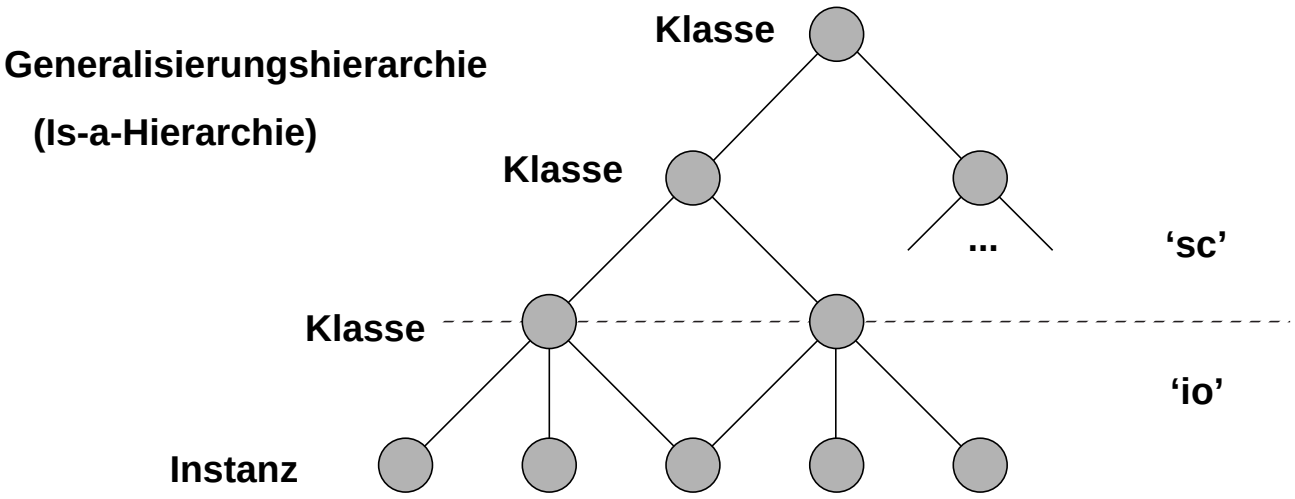
Die Darstellungen der anderen Abstraktionskonzepte erfolgen entsprechend.

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

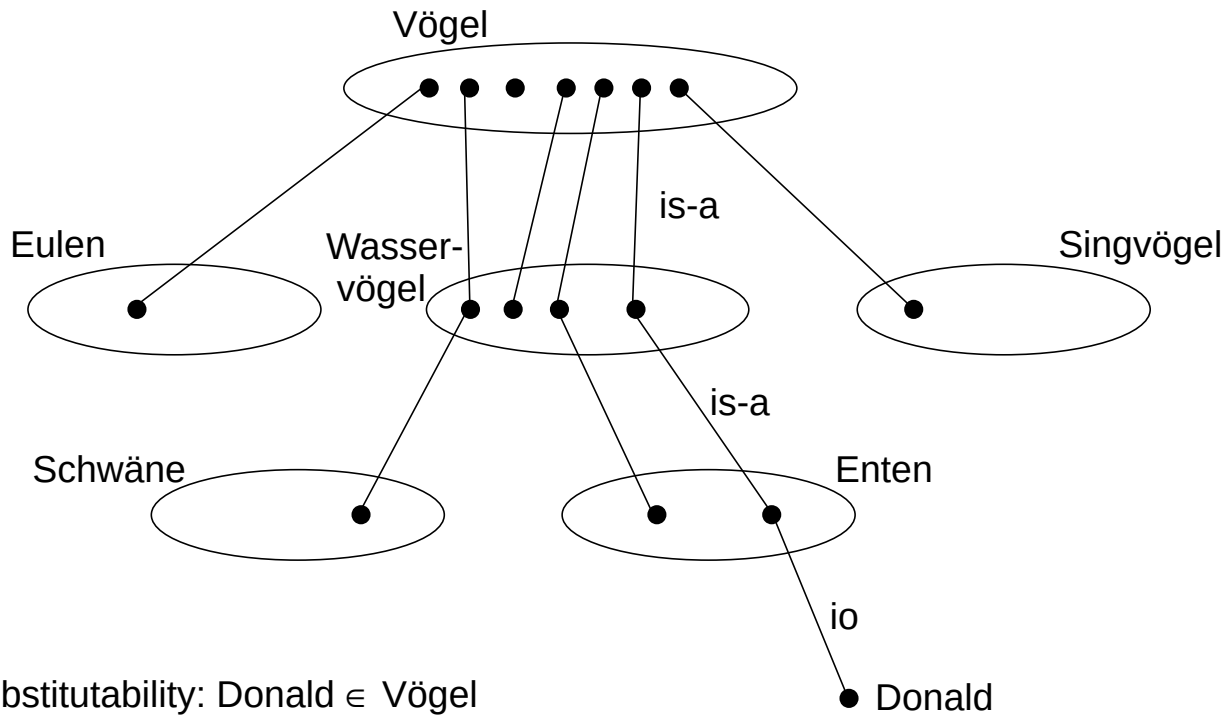
- **Anwendung**
  - Sie baut die **'subclass-of'**-Beziehung auf (**‘sc’**- oder **'is-a'**-Beziehung)
  - Sie ist rekursiv anwendbar (n-Ebenen-Beziehung) und organisiert die Klassen in einer Generalisierungshierarchie
  - Eine Superklasse ist eine Verallgemeinerung/Abstraktion der zugehörigen Subklassen. 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



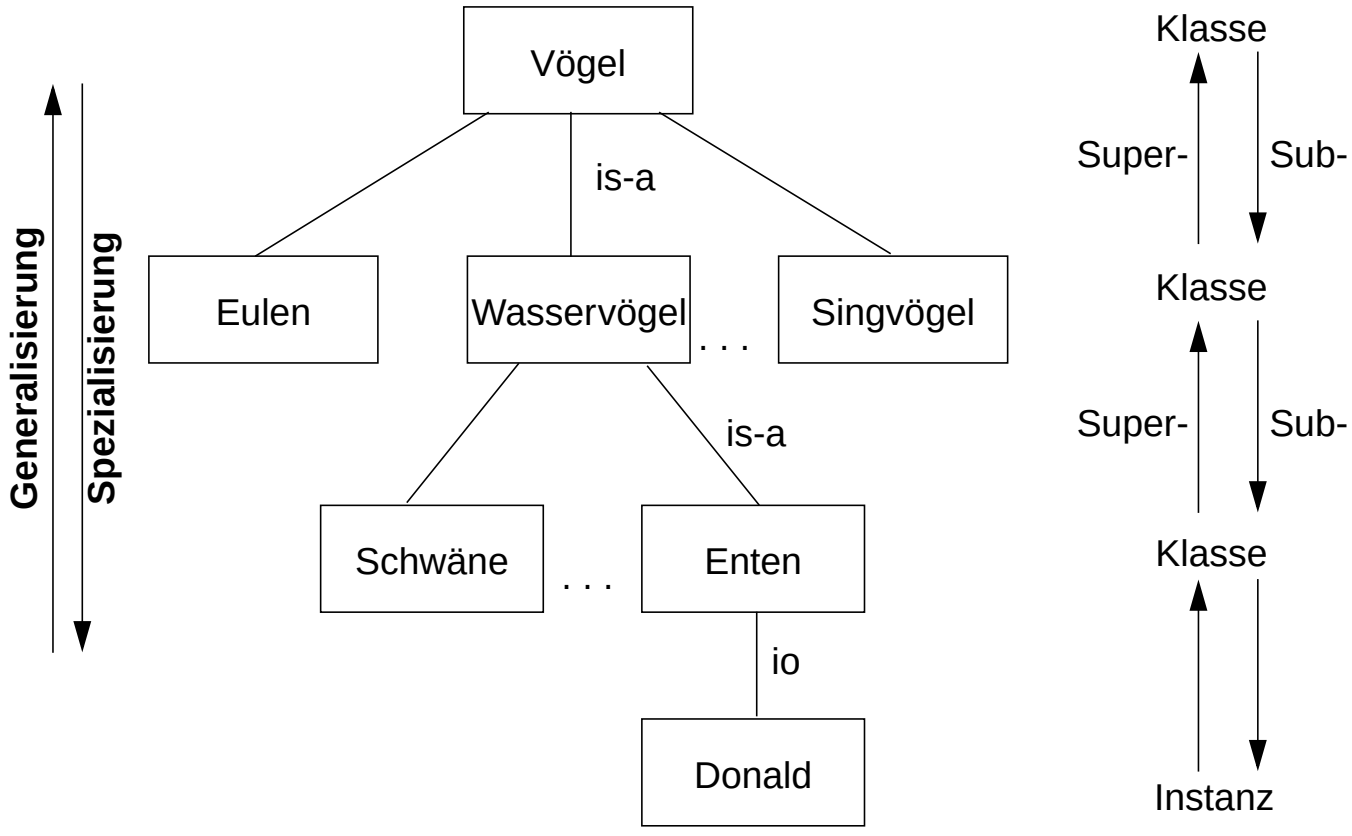
Modellierungsbeispiel zur Generalisierung

Instanzendarstellung



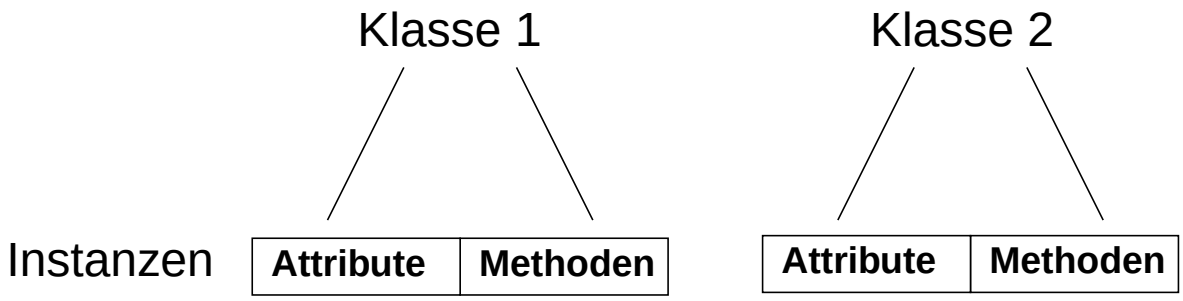
Substitutability: Donald $\in$ Vögel

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

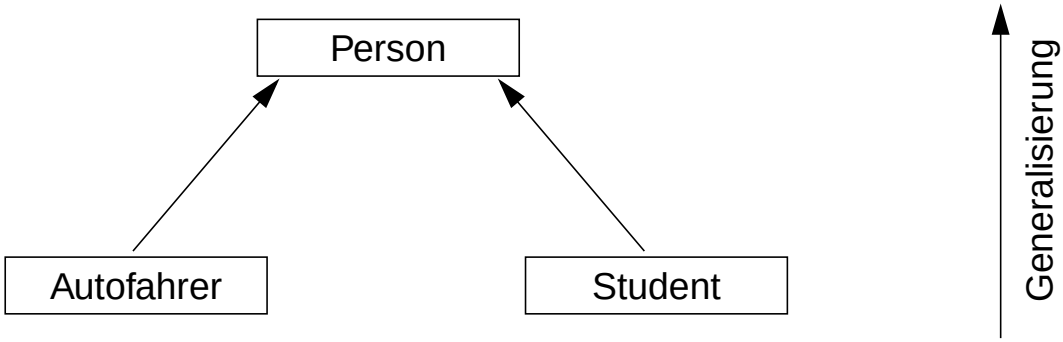
- jede Instanz der Subklasse ist immer auch Instanz der Klasse, aber nicht umgekehrt
- jede Methode, die auf die Instanzen einer Klasse anwendbar ist, ist damit immer auch auf die Instanzen sämtlicher Subklassen anwendbar

→ Garantie von bestimmten Integritätsbedingungen durch das System:

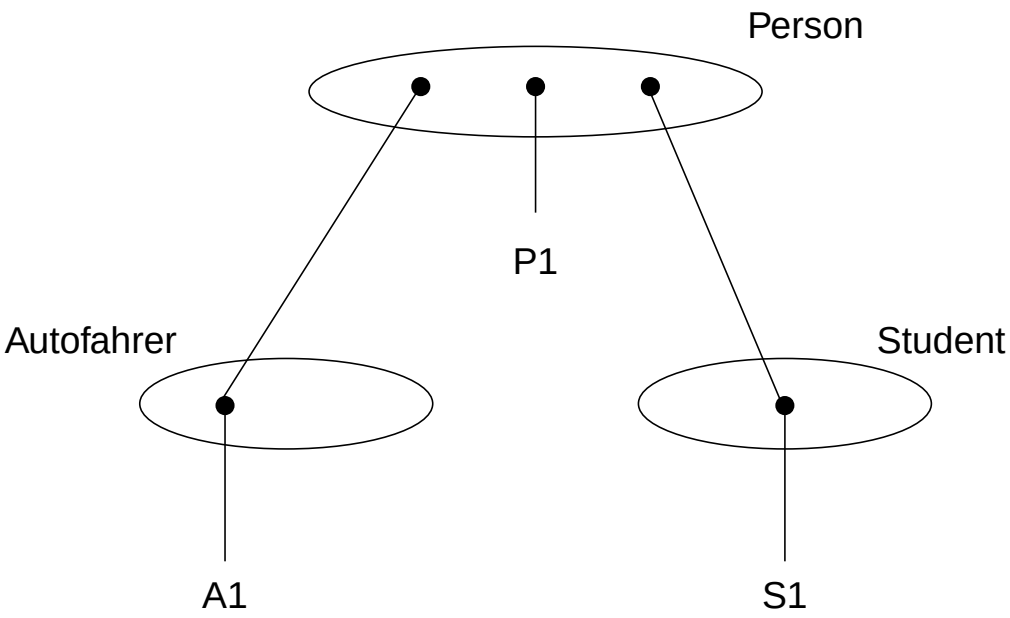
jeder Student ist auch Person

Generalisierung - Beispiel

- **Klassen**



- **Instanzen**

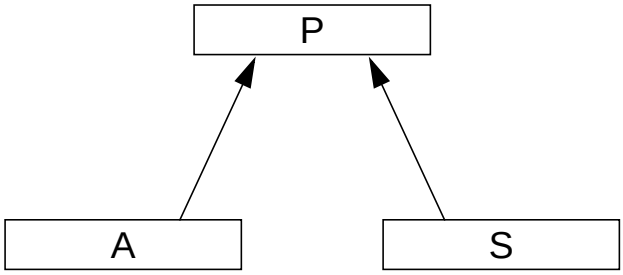


- **Subklassen sind i. allg. nicht disjunkt!**

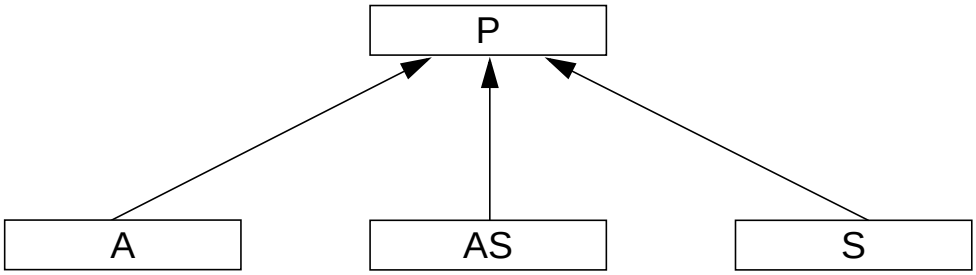
Generalisierung - Beispiel (2)

- **Überlappende Subklassen - Ansätze:**

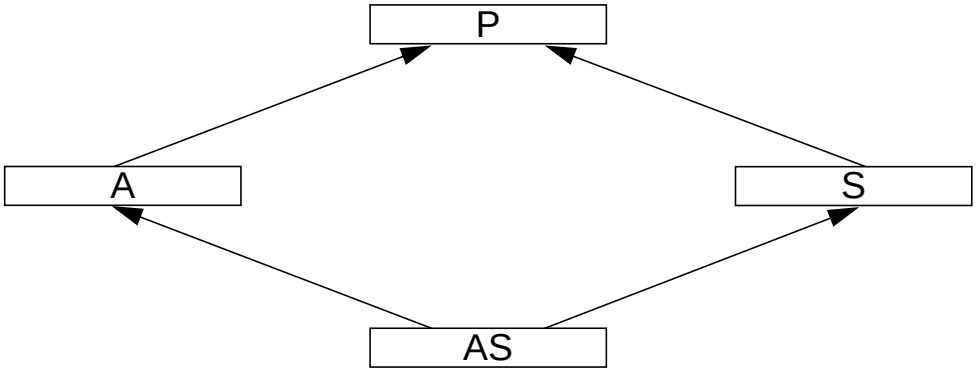
  1. **Mehrklassenmitgliedschaft von Instanzen**



  2. **Einklassenmitgliedschaft von Instanzen (Einfachvererbung)**



  3. **Einklassenmitgliedschaft von Instanzen (Mehrfachvererbung)**



Generalisierung (3)

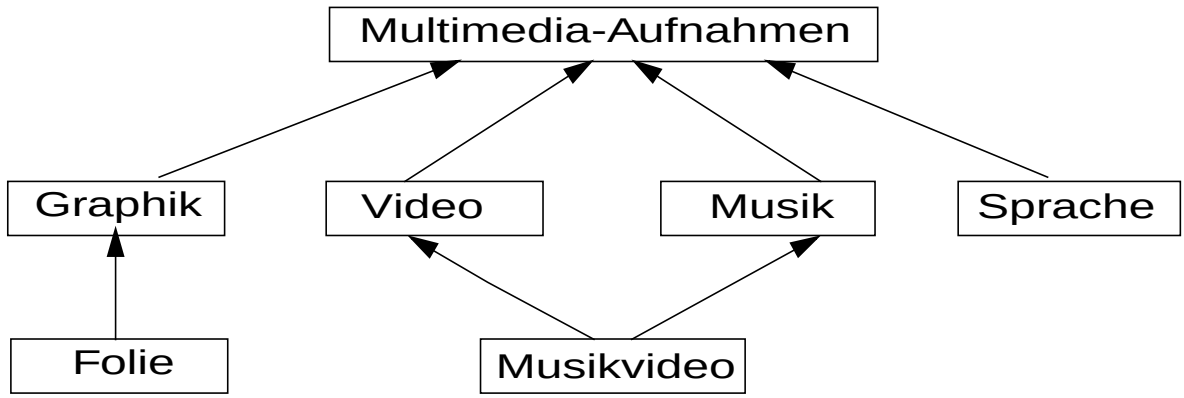
- **historisches Vorbild: Carl von Linné**
  - biologische Systematik
  - Reich (z.B. Tierreich) – Stamm (Chordatiere) – Klasse (Säugetiere) – Familie – Gattung – Art

    ➞ 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.

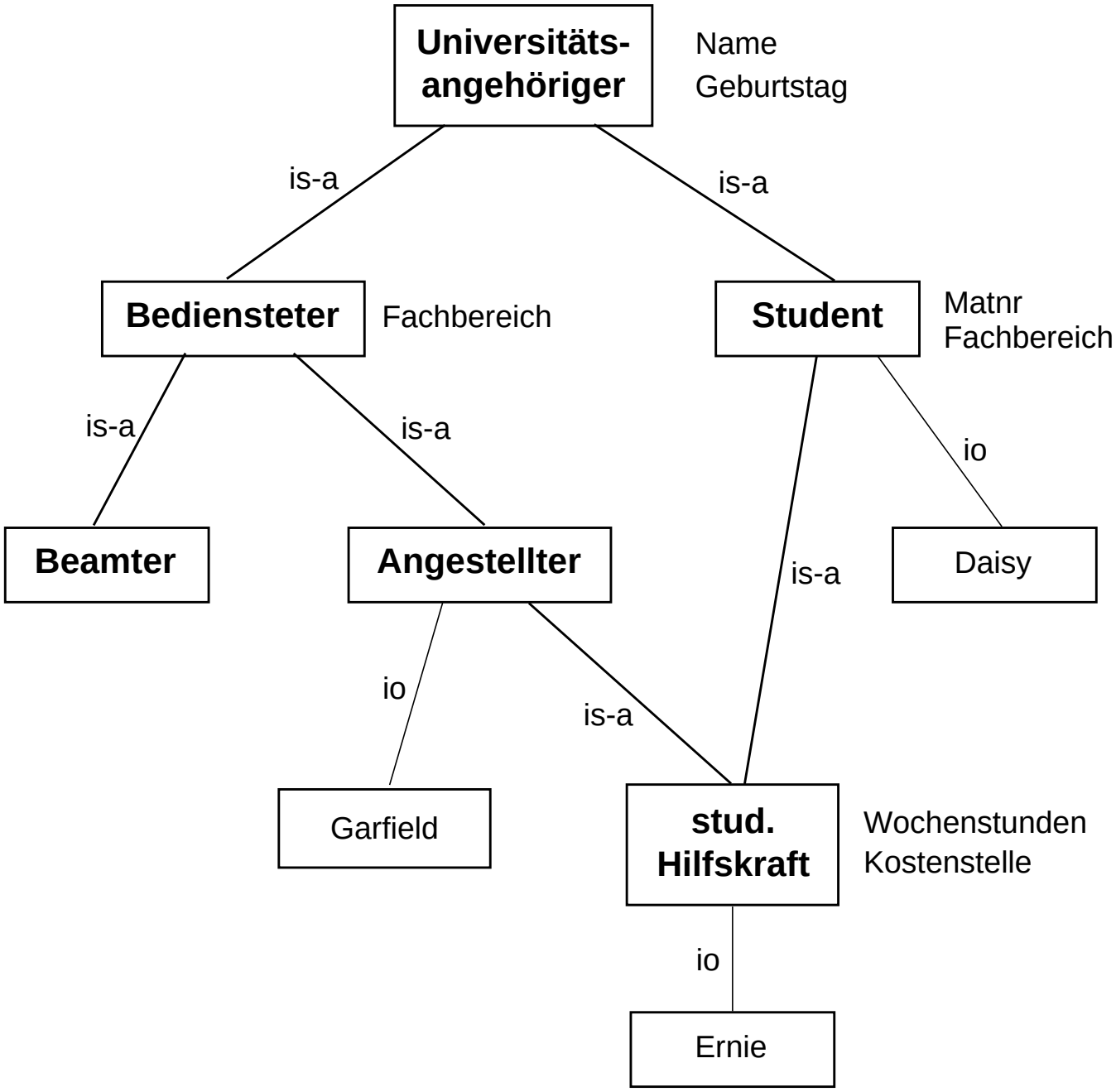
- **Arten der Vererbung**
  - Einfach-Vererbung (eindeutig)
  - Mehrfach-Vererbung

- **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

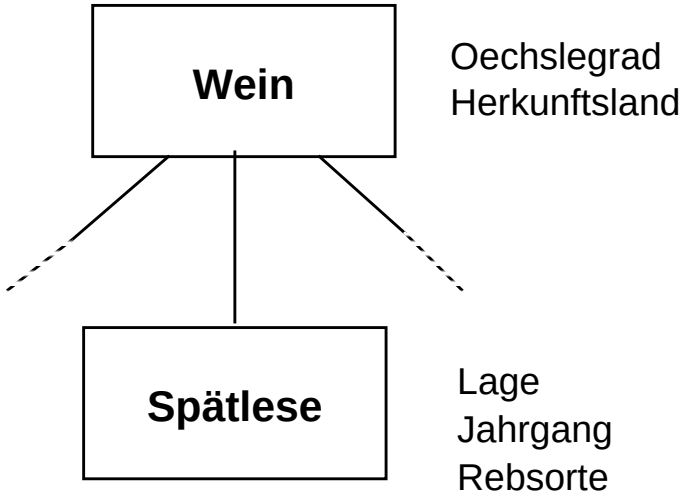


- **Mehrfach-Vererbung** (*multiple inheritance*) kann zu Konflikten führen

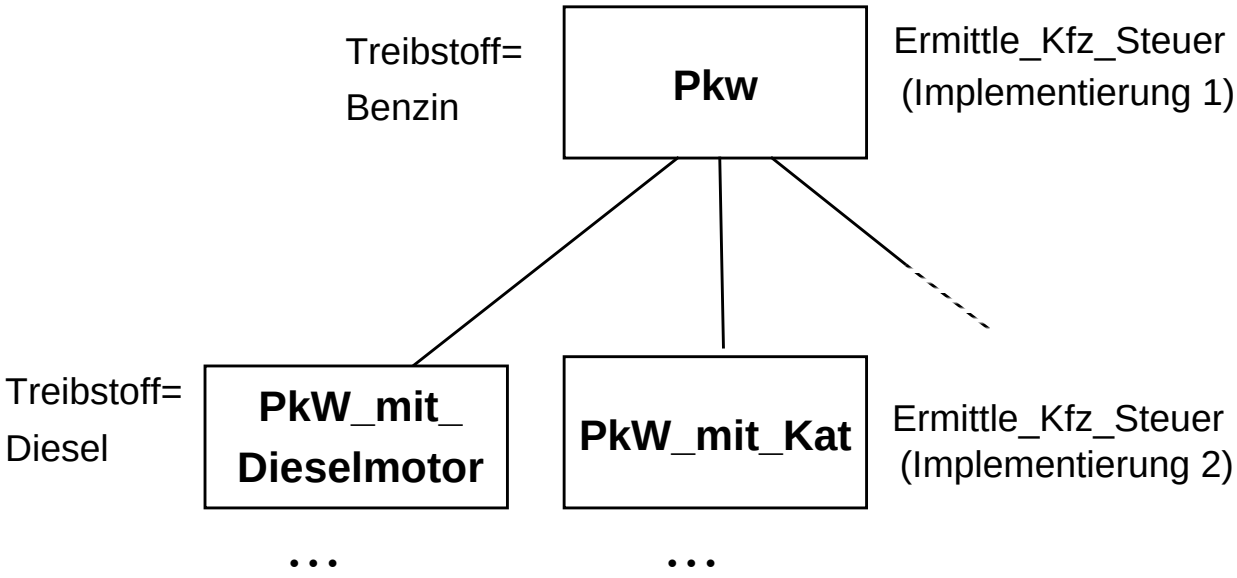
  → Auflösung explizit durch den Benutzer, z. B. durch Umbenennung:
  Hiwi_im_Fachbereich → Fachbereich of Angestellter
  immatrikuliert_im_Fachbereich → Fachbereich of Student

Vererbung (2)

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



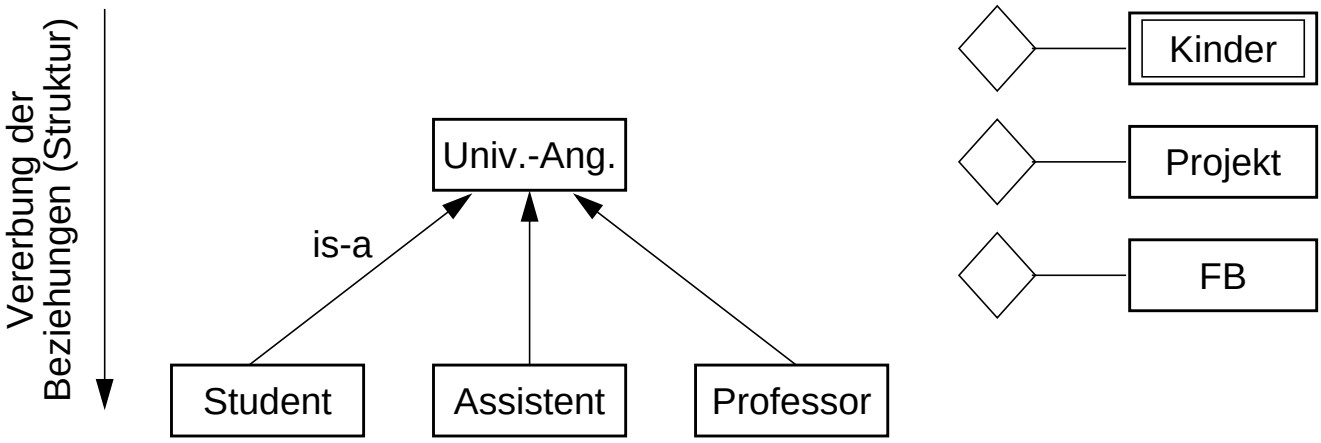
- Subklasse kann ererbte Attributwerte **überschreiben** oder Methoden **überladen:**



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

Vererbung (3)

- **Vererbung von Beziehungen:**

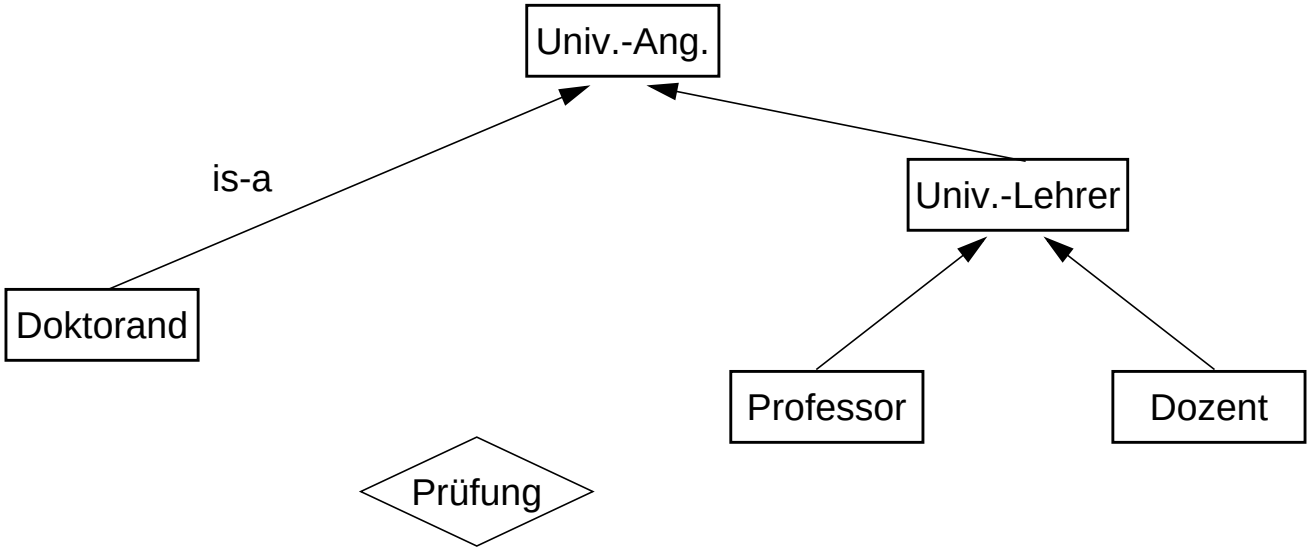


- **Beispiel: Doktorprüfung**

  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: $Z = \{S_1, S_2, \ldots S_n\}$
  Menge von Subklassen S_i mit derselben Superklasse G

  Z heißt **vollständig**, falls gilt

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

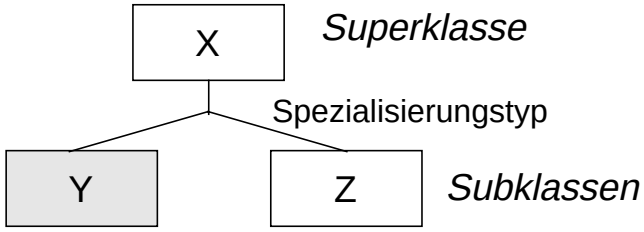
  andernfalls **partiell**.

  Z ist **disjunkt**, falls

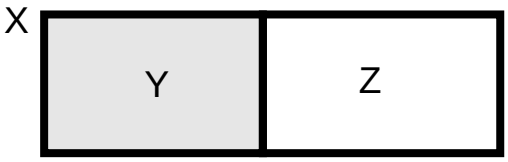
  $$S_i \cap S_j = \{\ \} \quad \text{für } i \neq j$$

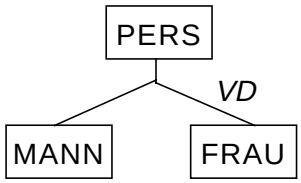
  andernfalls **überlappend**.

Arten von Spezialisierungen

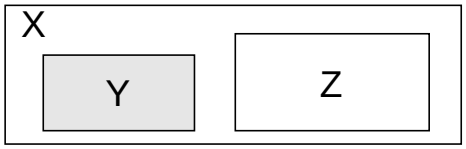


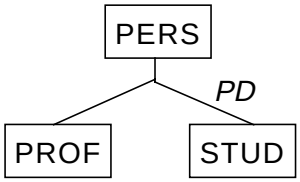
1. vollständig, disjunkt (VD)



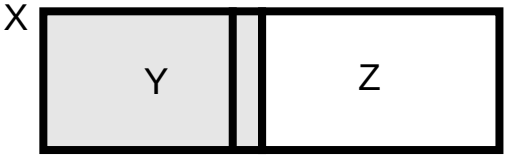


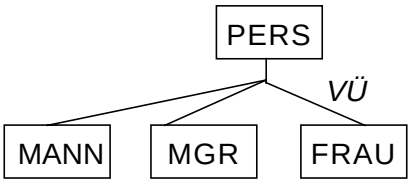
2. partiell, disjunkt (PD)



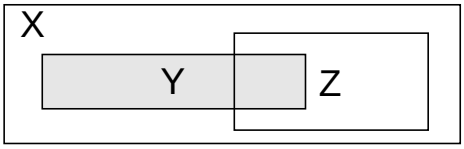


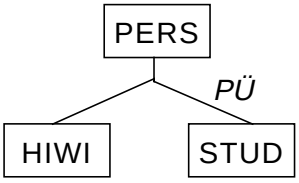
3. vollständig, überlappend (VÜ)





4. partiell, überlappend (PÜ)



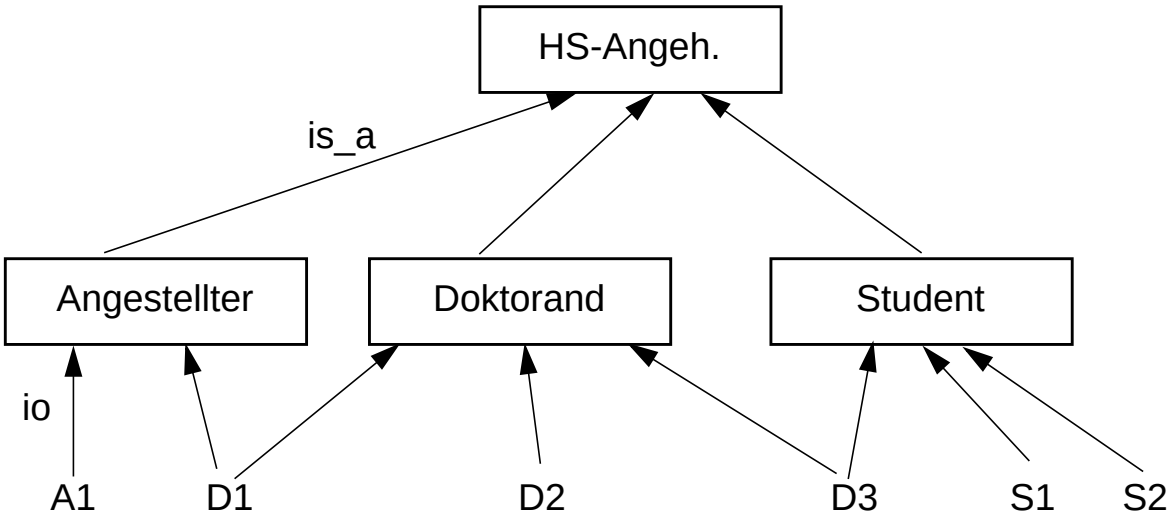


Spezialisierung bei Überlappungen

- **Entwurfsbeispiel: Hochschulangehörige (HS-Angeh.)**
  - Angestellte, Doktoranden und Studenten sind Hochschulangehörige
  - Doktoranden können zusätzlich Angestellte oder Studenten sein
  - Nachträgliche Erweiterung: Doktoranden und Studenten können zusätzlich Hiwis sein; Hiwis sind jedoch keine eigenständige Objekte

I. Überlappende Klassen –

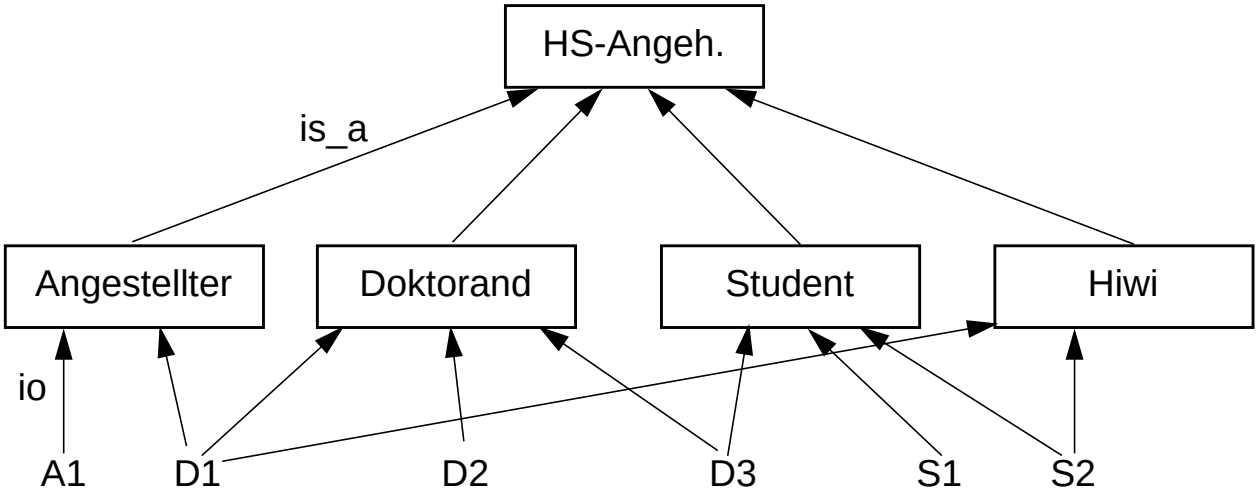
Mehrklassen-Mitgliedschaft von Instanzen



- **Entwurfsalgorithmus**
  - Zerlegung der Superklasse G in Subklassen S_i, wobei jeder Entity-Typ unabhängig von möglichen Überlappungen als Klasse repräsentiert wird
  - Instanzen werden ggf. mehreren Klassen (durch den Benutzer) zugeordnet
- **Vorteile:**
  - beschränkte Anzahl von Klassen
  - relativ einfache Erweiterung/Verfeinerung (Schema-Evolution)
  - dynamisches Ändern der Klassenmitgliedschaft ist sehr einfach

Spezialisierung bei Überlappungen (2)

- **Erweiterung des Schemas**



- **Verarbeitungsbeispiel:**
  - Einfügen einer Instanz (D1) bei Mehrklassen-Mitgliedschaft:

```
Insert   D1  INTO Angestellter    (OID = 4711, Ang.-Attr.)
Insert   D1  INTO Doktorand       (OID = 4711, Dokt.-Attr.)
Insert   D1  INTO Hiwi            (OID = 4711, Hiwi-Attr.)
```

  - Suche alle Doktoranden (auch D1):

```
Select *
From Doktorand
```

- **NACHTEILE zu I:**
  - *unnatürliche Modellierung:* Sie erzwingt z. B. „Hiwi“ als eigenständige Klasse
  - *Verlust struktureller Eigenschaften:* ER-Schema enthält nur wenige Strukturbeziehungen zwischen Entity-Typen, d. h., viele Einschränkungen (z. B. ein Hiwi ist Student oder Doktorand) sind nicht systemkontrolliert
  - *Stark eingeschränkte Vererbungsmöglichkeiten:* Braucht man z. B. in Student und Hiwi die Attribute Matnr und Semanz, so muß der Benutzer sie zweimal definieren: diese Redundanz wird vom System nicht gewartet
  - *Mehrklassen-Mitgliedschaft von Instanzen:* Sie wird ausschließlich vom Benutzer kontrolliert

Spezialisierung bei Überlappungen (3)

II. Disjunkte Klassen –
Einklassen-Mitgliedschaft von Instanzen

- **Entwurfsalgorithmus**

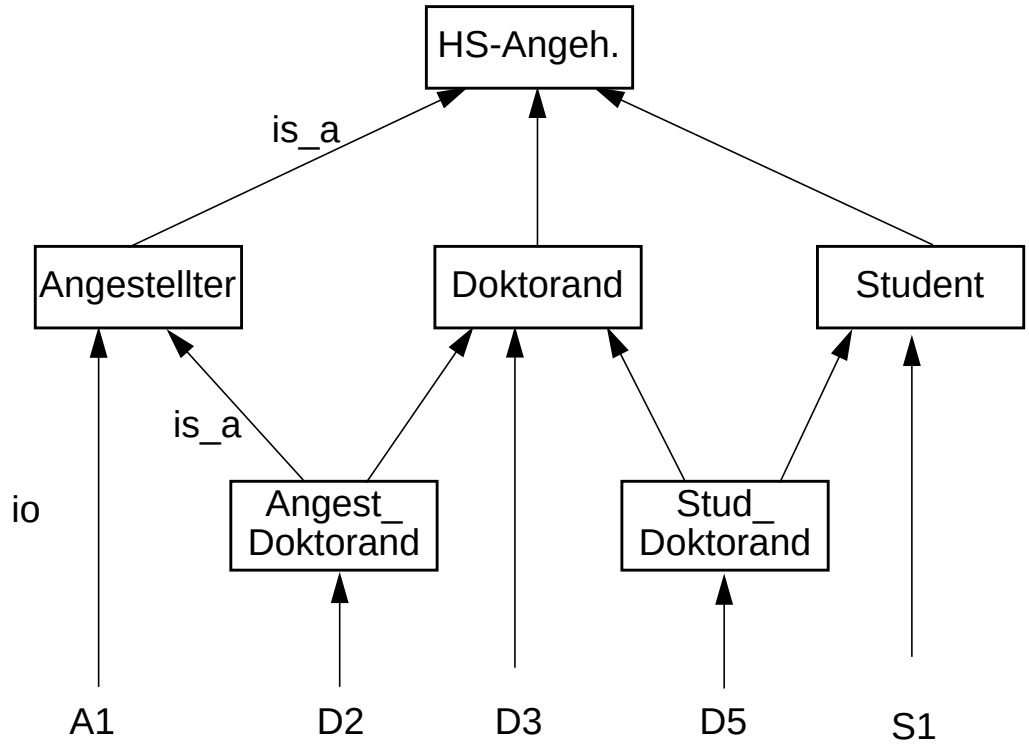
1. Zerlegung der Superklasse G in Subklassen S_i, wobei S_i eigenständige Instanzen besitzen kann

2. Bestimmung möglicher Überlappungen der Klasse S_i und Bildung einer neuen Subklasse S_j für jede Überlappung; S_j besitzt dann alle an dieser Überlappung beteiligten Klassen S als Superklassen

   Verfeinerung der Spezialisierung innerhalb von Klassen
   (z. B. Hiwi bei Student und Doktorand)

3. Für jede in den Schritten 1 und 2 erhaltene Klasse wird der Algorithmus rekursiv angewendet, wobei S_i die Rolle der Superklasse G übernimmt

- **Ausgangsschema**

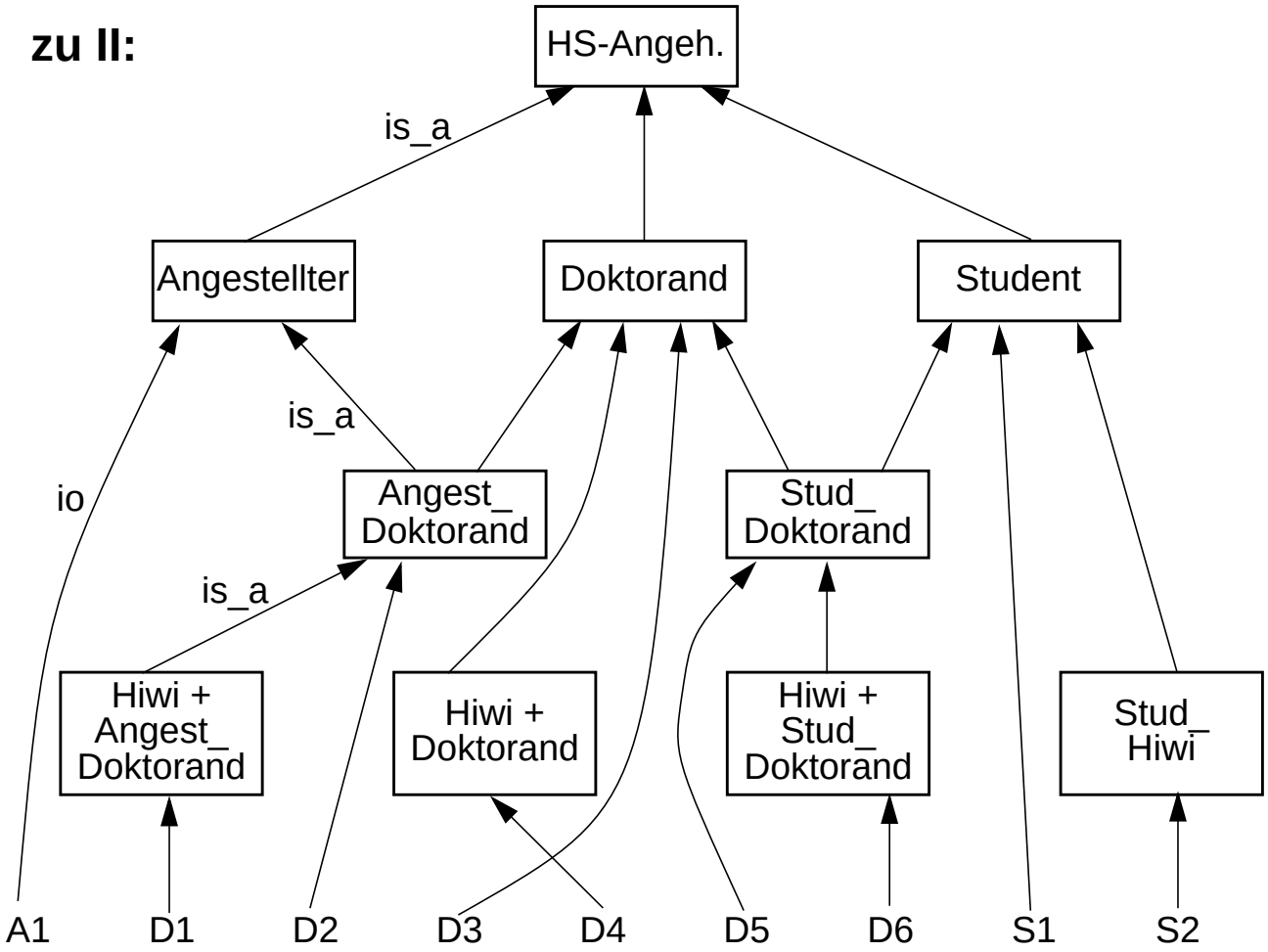


Spezialisierung bei Überlappungen (4)

- **Vorteile**
  - *keine Mehrklassen-Mitgliedschaft* von Instanzen
  - ER-Schema repräsentiert alle *strukturellen Eigenschaften*, was die volle Nutzung der Vererbungsmöglichkeiten impliziert
  - Systemkontrolle bei *strukturellen Einschränkungen*

- **Erweiterung des Schemas**

zu II:



Spezialisierung bei Überlappungen (5)

- **Nachteile**
  - „Atomisierung“ der Klassen
  - *sehr komplexe Modellierung* (10 Klassen),
    aber dafür explizite Definition struktureller Einschränkungen
  - gleichartige Verfeinerungen (z. B. Hiwi), die in mehreren Klassen isoliert stattfinden, erzwingen wiederholte Definition gleicher Attribute

- **Verarbeitungsbeispiel:**
  - Einfügen einer Instanz (D1) bei Einklassen-Mitgliedschaft:

    Insert D1 INTO (Hiwi+Angest_Doktorand)
    (OID = 4711, Hiwi + Ang. + Dokt.-Attr.)

  - Suche alle Doktoranden (auch D1):

    Select *
    From Doktorand

    oder Suche alle angestellten Doktoranden (auch D1):

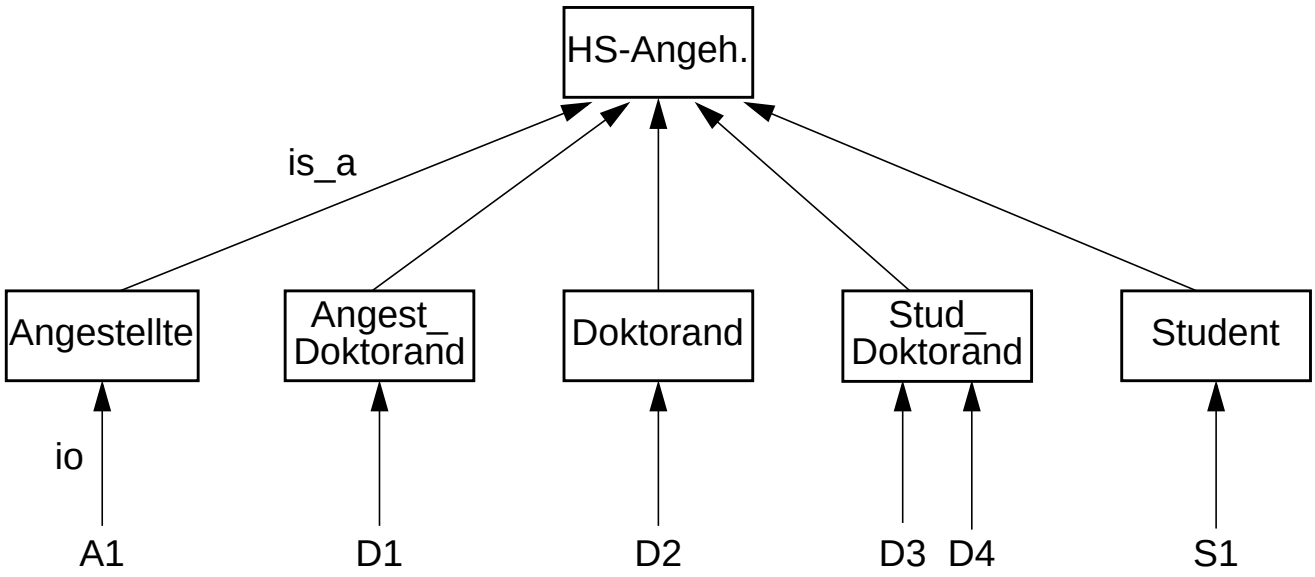
    Select *
    From Angest_Doktorand

    oder . . .

Spezialisierung bei Überlappungen (6)

III. Mischform

- **Wunsch:**
  Man möchte Einklassen-Mitgliedschaft der Instanzen und geringere Schema-Komplexität erzielen!
- **Variation von Ansatz I ergibt folgendes Schema:**



Begrenzung des Schemas auf eine einheitliche Klassentiefe

- **Welche Vor- und Nachteile ergeben sich?**
  - Was wird vererbt, was muß redundant definiert werden?
  - Wer ist für die Wartung verantwortlich?
  - Wie verhält sich die Anzahl der Klassen?

Spezialisierung bei Überlappungen (7)

- **Verarbeitungsbeispiel:**
  - Einfügen von Instanzen bei Einklassen-Mitgliedschaft:

    Insert D1 INTO Angest_Doktorand (OID = 4711, Angest_Dokt.-Attr.)

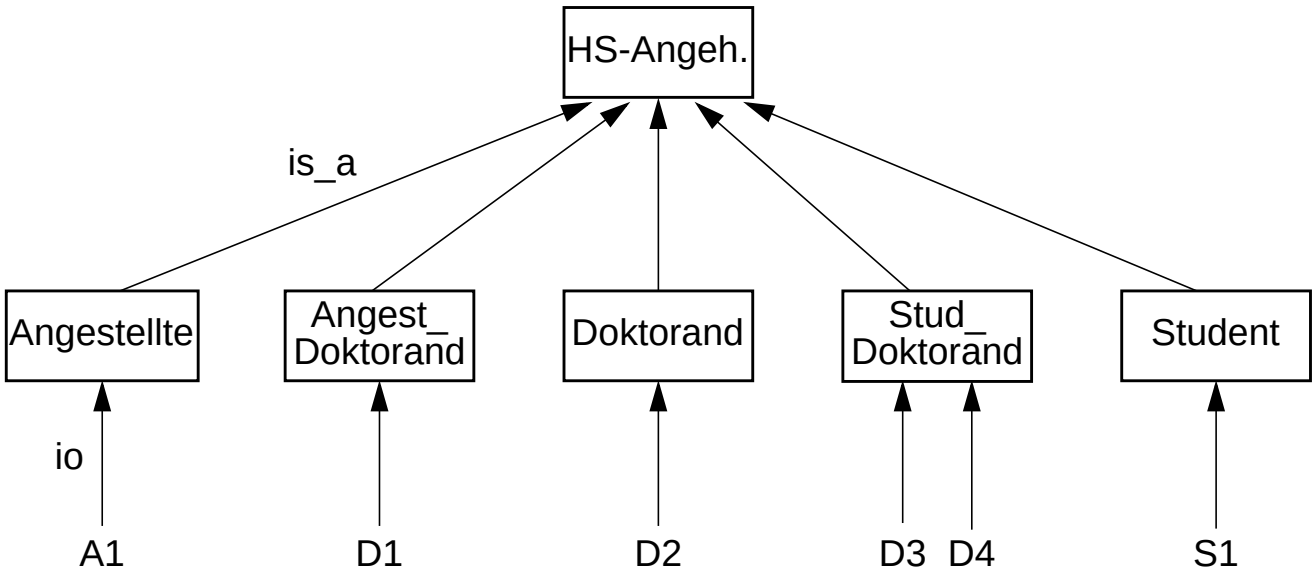
    Insert D2 INTO Doktorand (OID = 0815, Dokt.-Attr.)

    Insert D3 INTO Stud_Doktorand (OID = 1245, Stud_Dokt.-Attr.)

  - Suche alle Doktoranden (D2, aber auch D1 und D3 . . .)

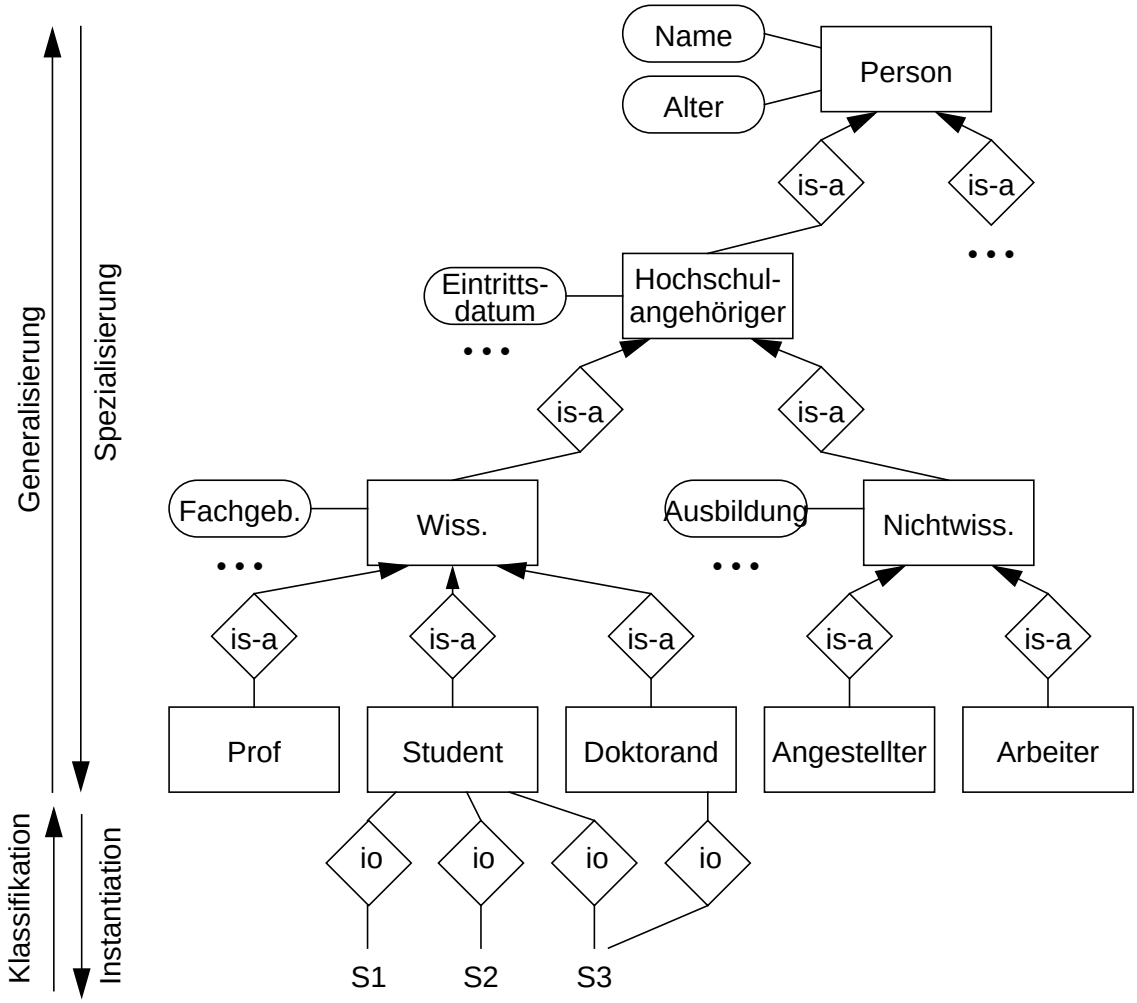
```
Select *
From Doktorand

UNION

Select *
From Angest_Doktorand

UNION

Select *
From Stud_Doktorand
```

- **Erweiterung des Schema: Wie ?**



Ist eine Erweiterung des Schemas um 4 Hiwi-Klassen sinnvoll?

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 (**Mengen**objekt) als Ganzes zu beschreiben. Dabei werden einerseits Details der einzelnen Elemente unterdrückt und andererseits bestimmte Eigenschaften, die die Objektgruppe charakterisieren, hervorgehoben.

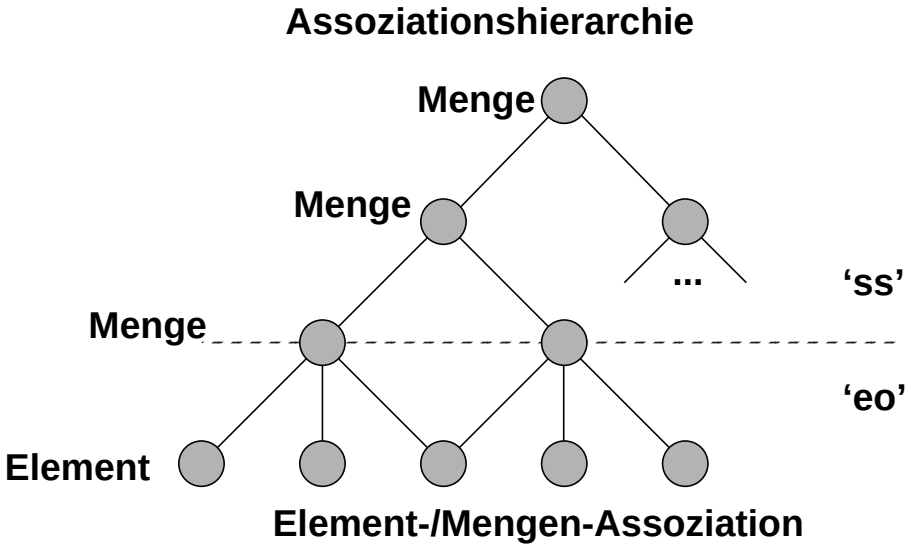
- **Anwendung**

  - 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’**) als1-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

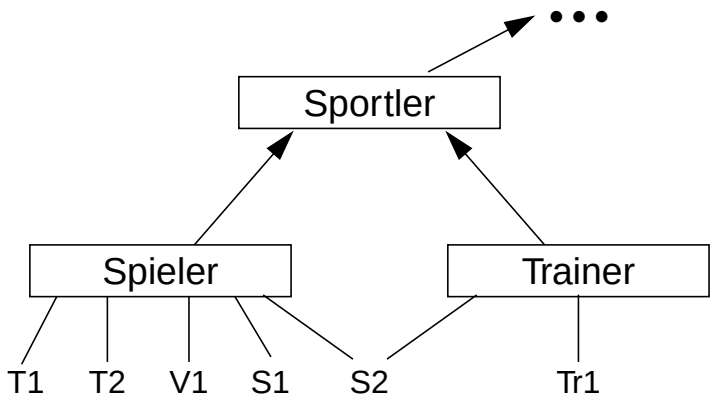
    → Netzwerke, (n:m) !

- **Systemkontrollierte Ableitungen bei der Assoziation**

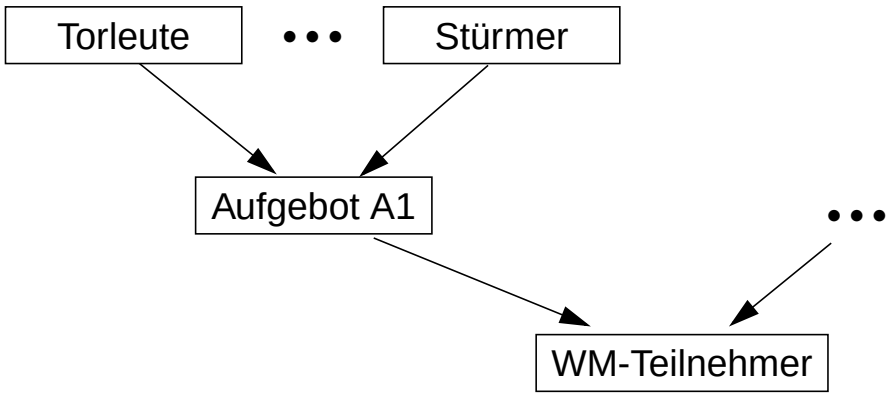
  - Sie unterstützt keine Vererbung, da die Mengeneigenschaften keine Element-eigenschaften sind
  - Durch **Mitgliedschaftsimplikation** lassen sich Eigenschaften bestimmen, die jedes gültige Element der Menge erfüllen muß
  - **Mengeneigenschaften** sind Eigenschaften der Menge, die über Element-eigenschaften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

Assoziation - Beispiel

- **Generalisierungshierarchie**



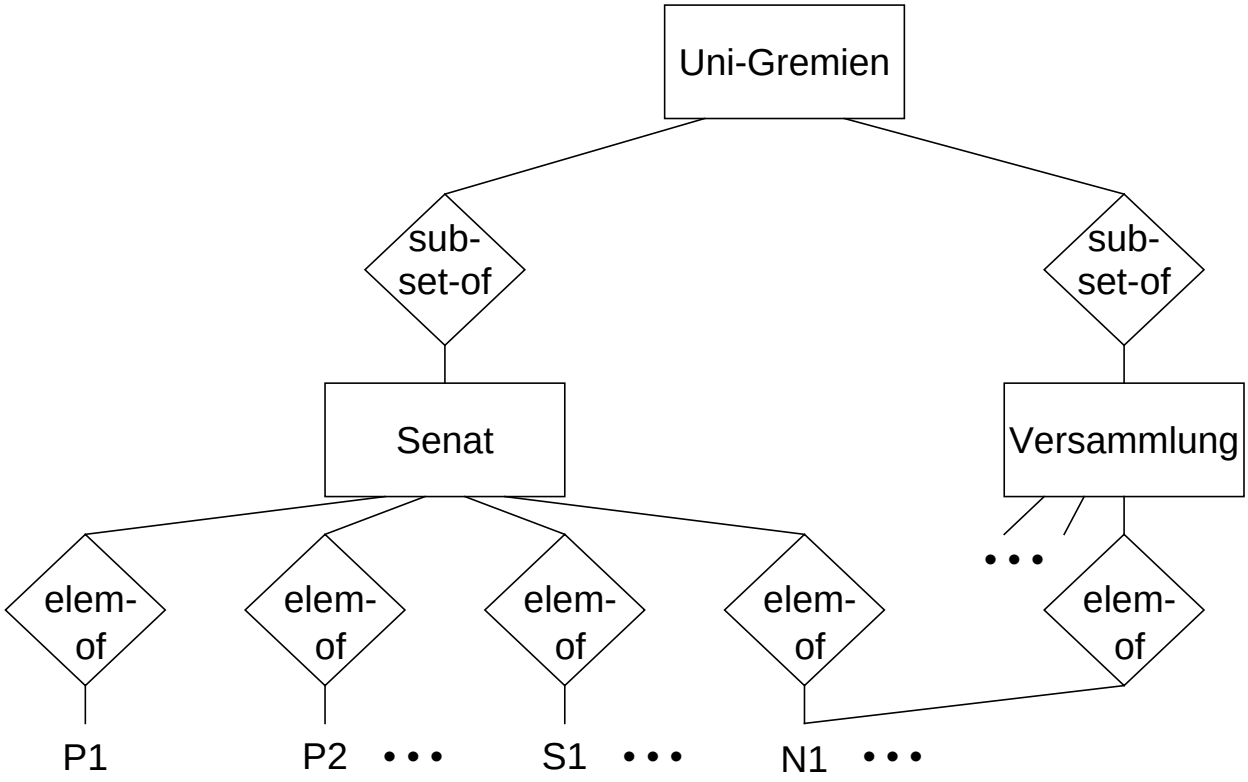
Assoziation



- **Operationen**
  - Erzeugen/Löschen
    - Create
    - Connect/Disconnect
      (manuell oder automatisch über Mengenprädikate)
    - Delete
  - Suchen
  - Schlußfolgerungen
    - Mitgliedschaftsimplikation
    - Mengeneigenschaft

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 *Mitgliedschaftsimplikation* (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**

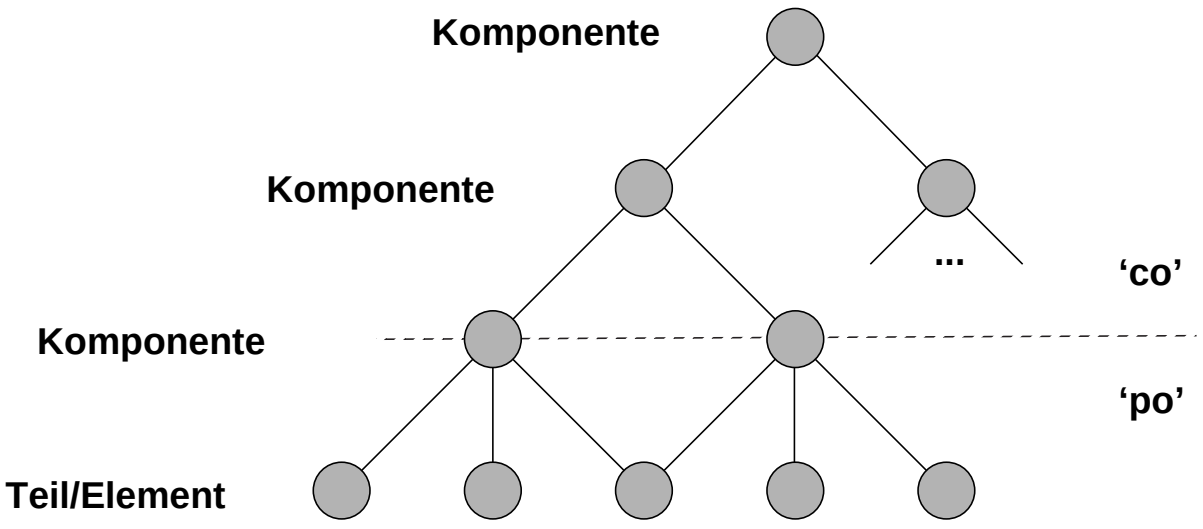
  - Eine Kollektion von einfachen Objekten (Element-Objekt, **Teil**) wird als zusammengesetztes Objekt (**Komponenten**objekt/**Aggregat**objekt) behandelt

  - 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:**

Aggregationshierarchie



Element-/Komponenten-Aggregation

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

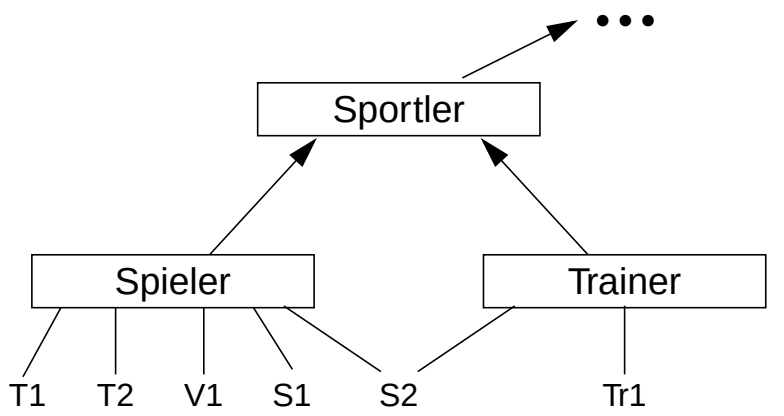
    → Unterschied zu Klassen und Mengenobjekten, die ohne Instanzen existieren können, bzw. für die leere Mengen erlaubt sind

  - Elemente einer Subkomponente sind gleichzeitig auch Elemente aller Superkomponenten dieser Subkomponente
  - Objekte können gleichzeitig Elemente verschiedener Komponenten bzw. auch Subkomponente von mehreren Superkomponenten sein

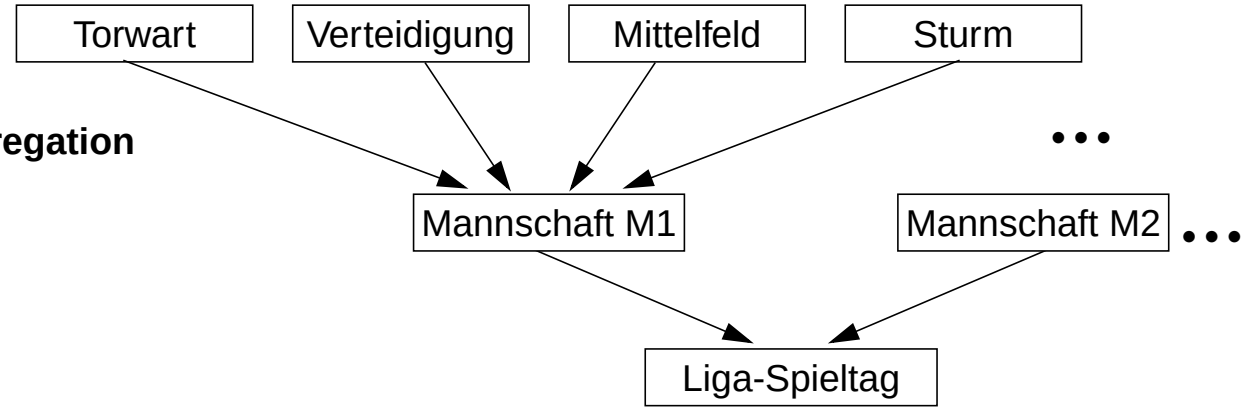
    → Netzwerke, ((n:m) !)

Aggregation - Beispiel

- **Generalisierungshierarchie**

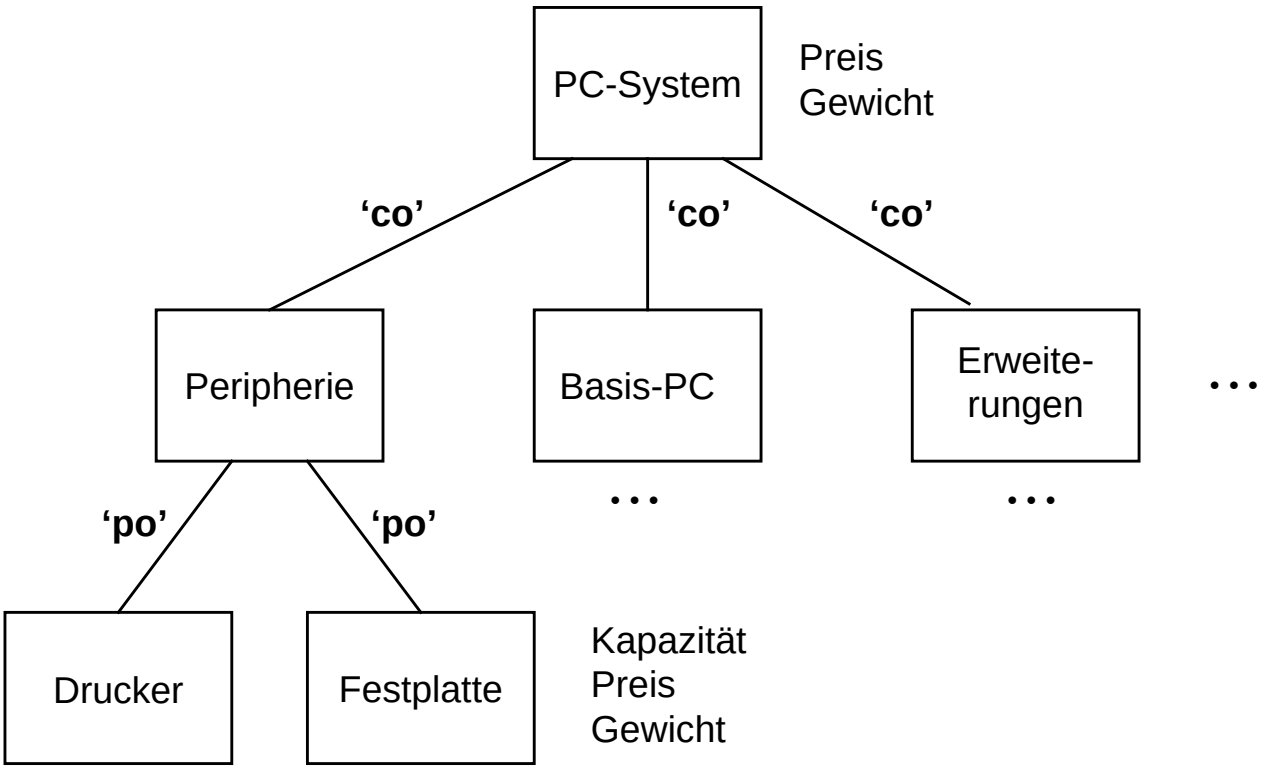


Aggregation



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

Aggregation - Beispiel (2)



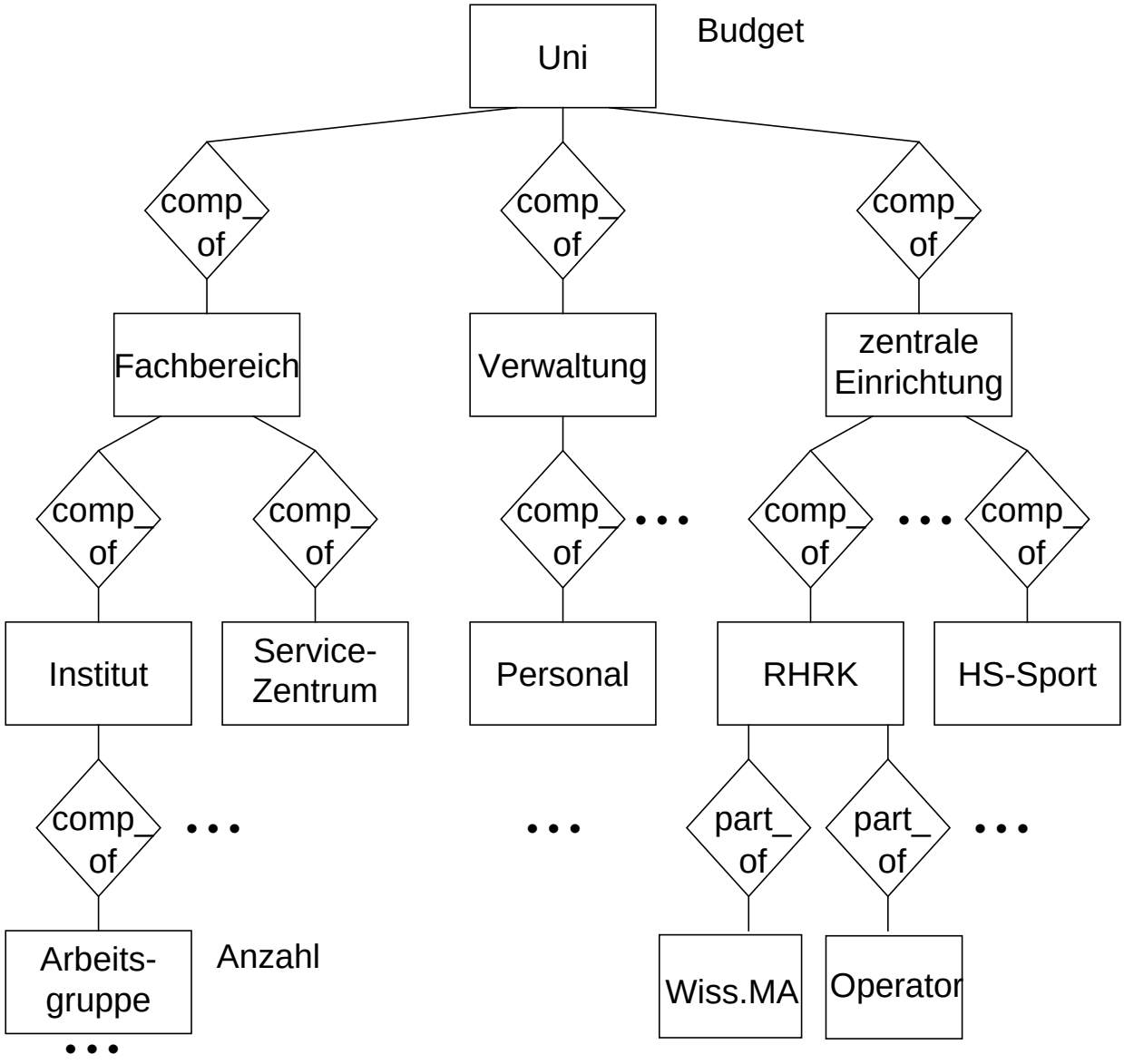
- **Systemkontrollierte Ableitungen: implizierte Prädikate**
  - Prädikate, die über der Aggregationshierarchie spezifiziert sind und gemeinsame Eigenschaften von Elementen/Aggregaten betreffen
  - ‘upward implied predicate’

    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

deklarative Attribute

Preis: 36
- possible-values: integer > 0
- cardinality: [1,1]
- unit: DM

Vorbereitungszeit:
- possible-values: integer > 0
- cardinality: [1,1]
- demon: Berechne-Vorbereitungszeit

geeignete-Getränke: trockener Rotwein, Bier
- possible-values: instance-of Getränke
- cardinality

.
.
.

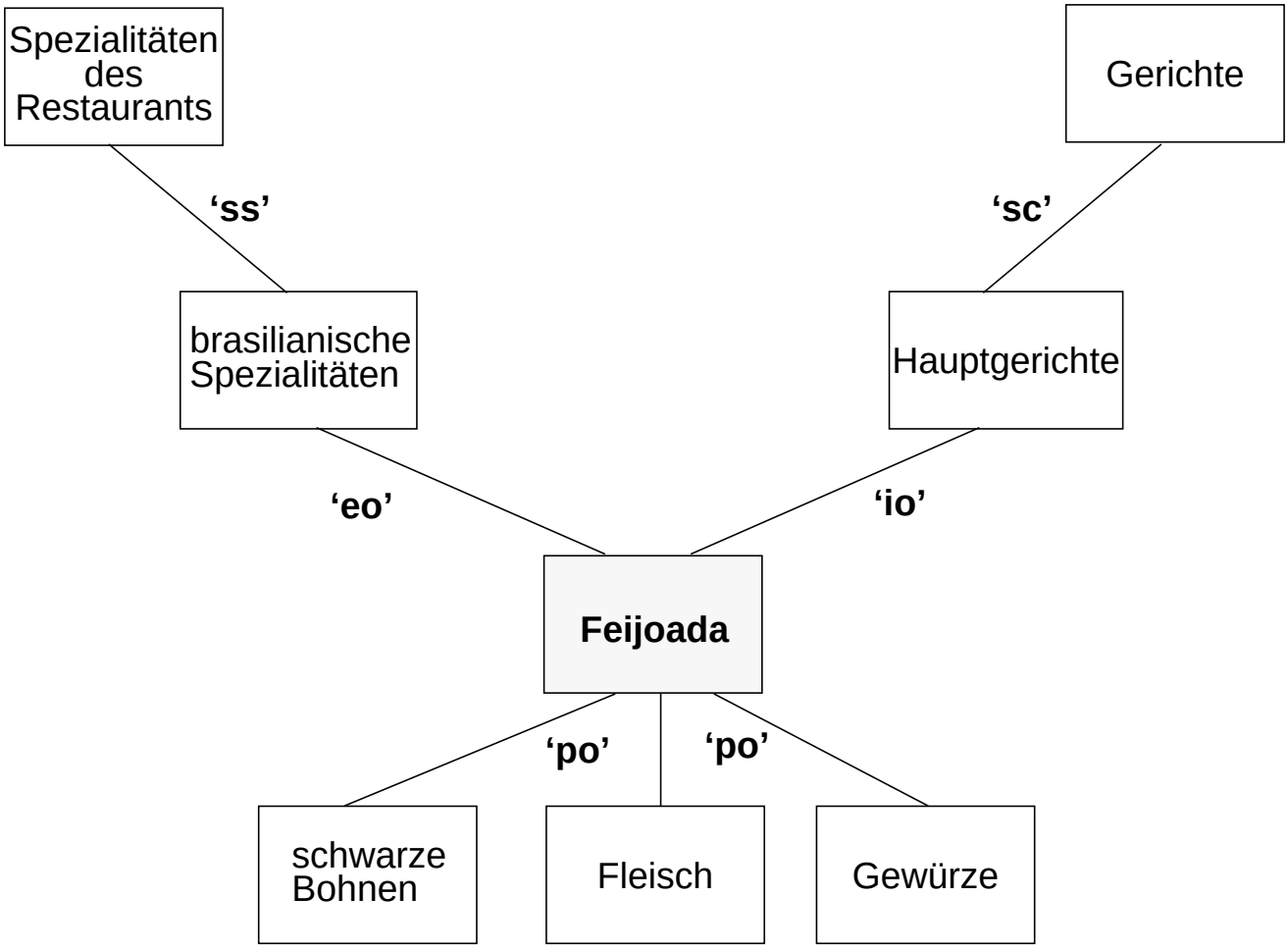
prozedurale Attribute

Bestellen (Anzahl-von-Personen)
- procedure BEGIN ... END

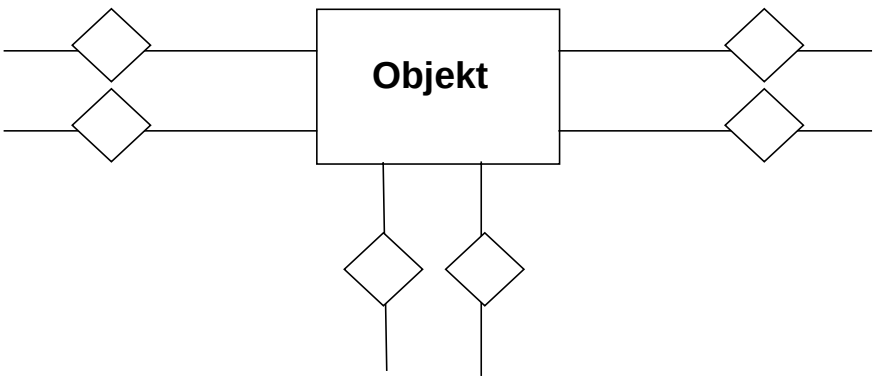
.
.
.

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



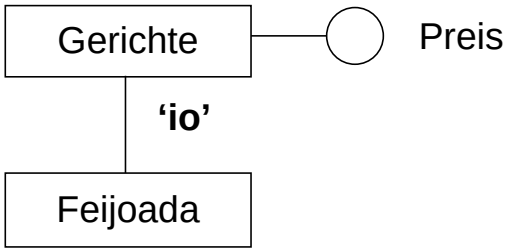
Darstellungsprinzip:



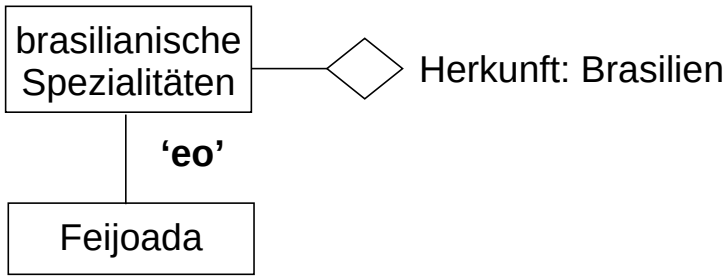
Modellinhärentes „Reasoning“

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

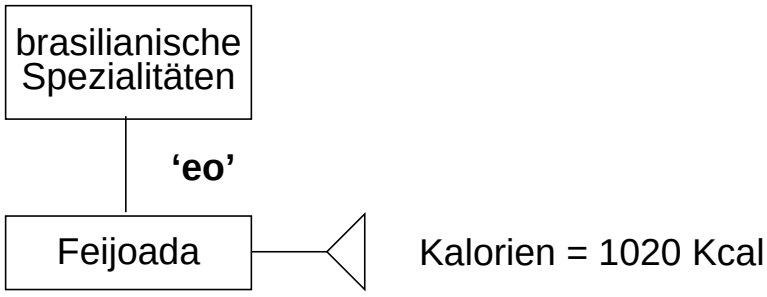
Vererbung



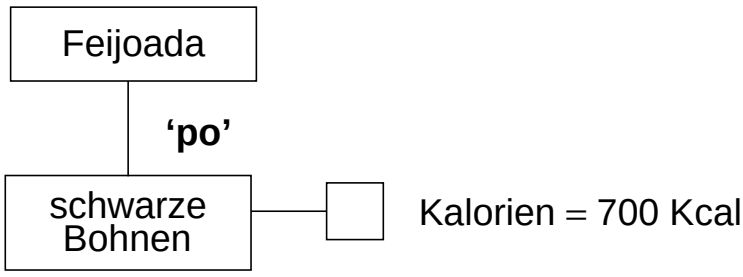
Mitgliedschafts-implikationen



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
  - Entity- und Relationship-Mengen
    (Attribut, Wertebereich, Primärschlüssel)
  - Klassifikation von Beziehungstypen
  - 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 (siehe Relationenmodell)

- **Abstraktionskonzepte und deren Implikationen**
  - Generalisierung und Vererbung
  - Assoziation mit Mengeneigenschaften und Mitgliedschaftsimplikationen
  - Aggregation und implizierte Prädikate
  - Integration der Abstraktionskonzepte mittels objektzentrierter Darstellung