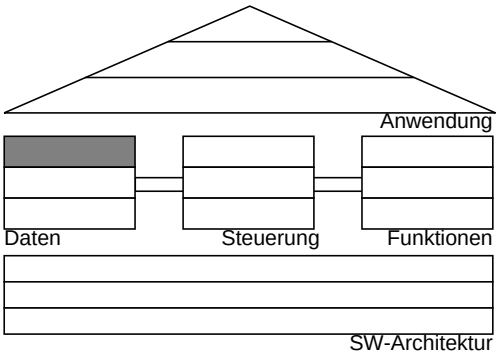
3. Informationsmodelle

- **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**

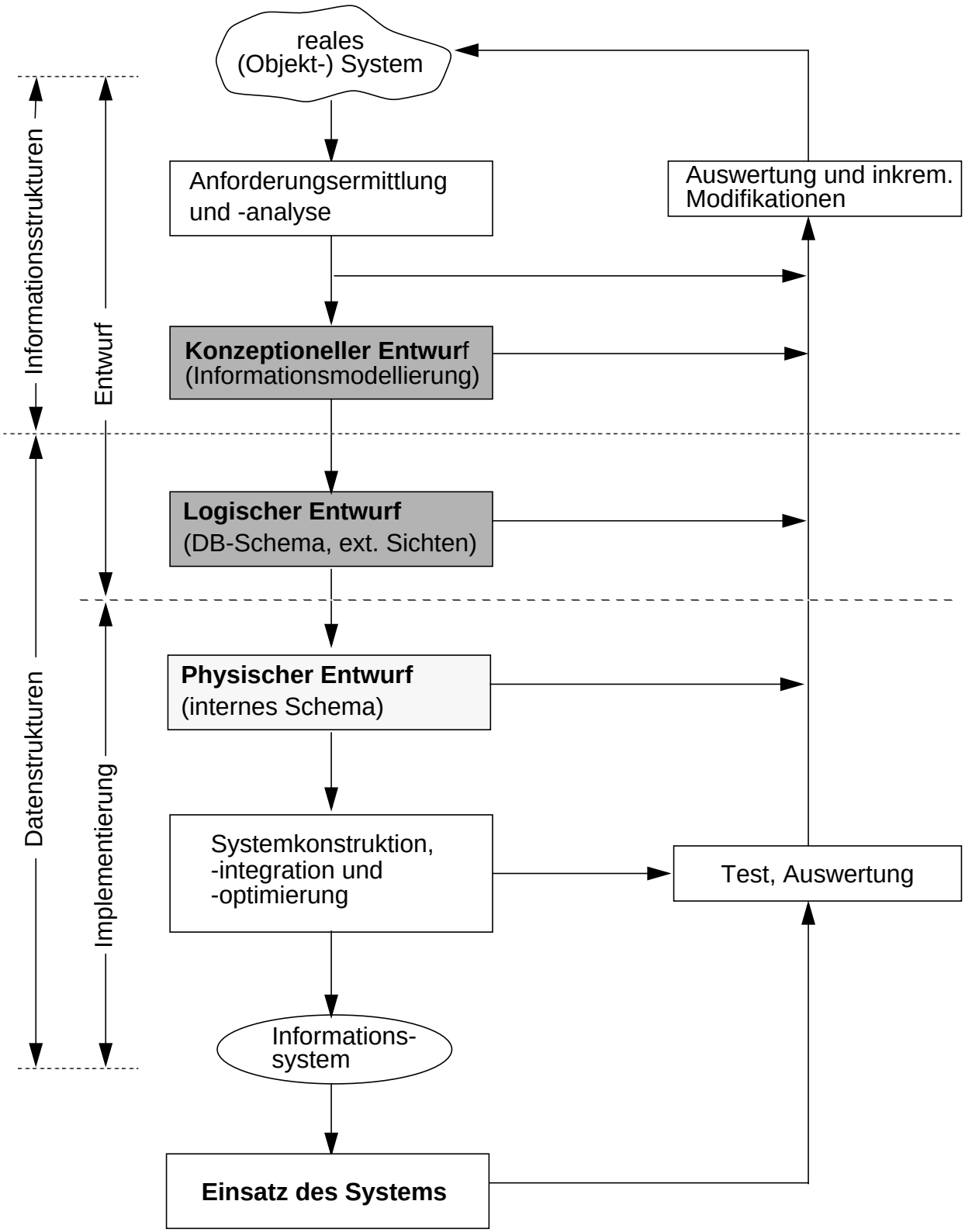
- **Nebenbedingungen:**
  - genaue Abbildung
  - hoher Grad an Aktualität
  - Verständlichkeit, Natürlichkeit, Einfachheit, ...

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

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

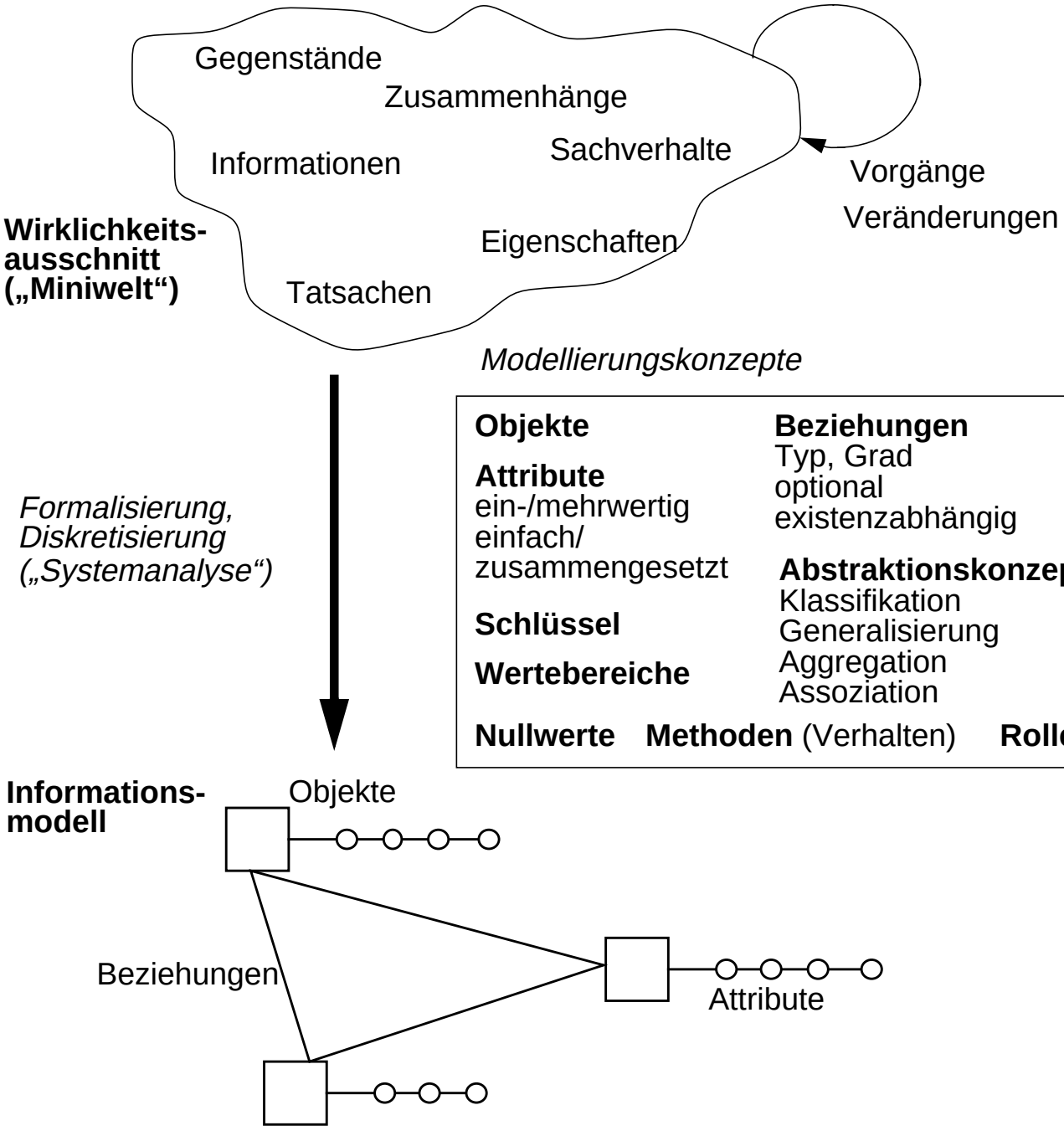
- **Schrittweise Ableitung: (Verschiedene Sichten)**
  1. Information in unserer Vorstellung
  2. Informationsstruktur: Organisationsform der Information
  3. Logische Datenstruktur (zugriffspfadunabhängig, Was-Aspekt)
  4. Physische Datenstruktur (zugriffspfadabhängig, Was- und Wie-Aspekt)

Schritte auf dem Weg zu einem Informationssystem



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

Informationsmodelle

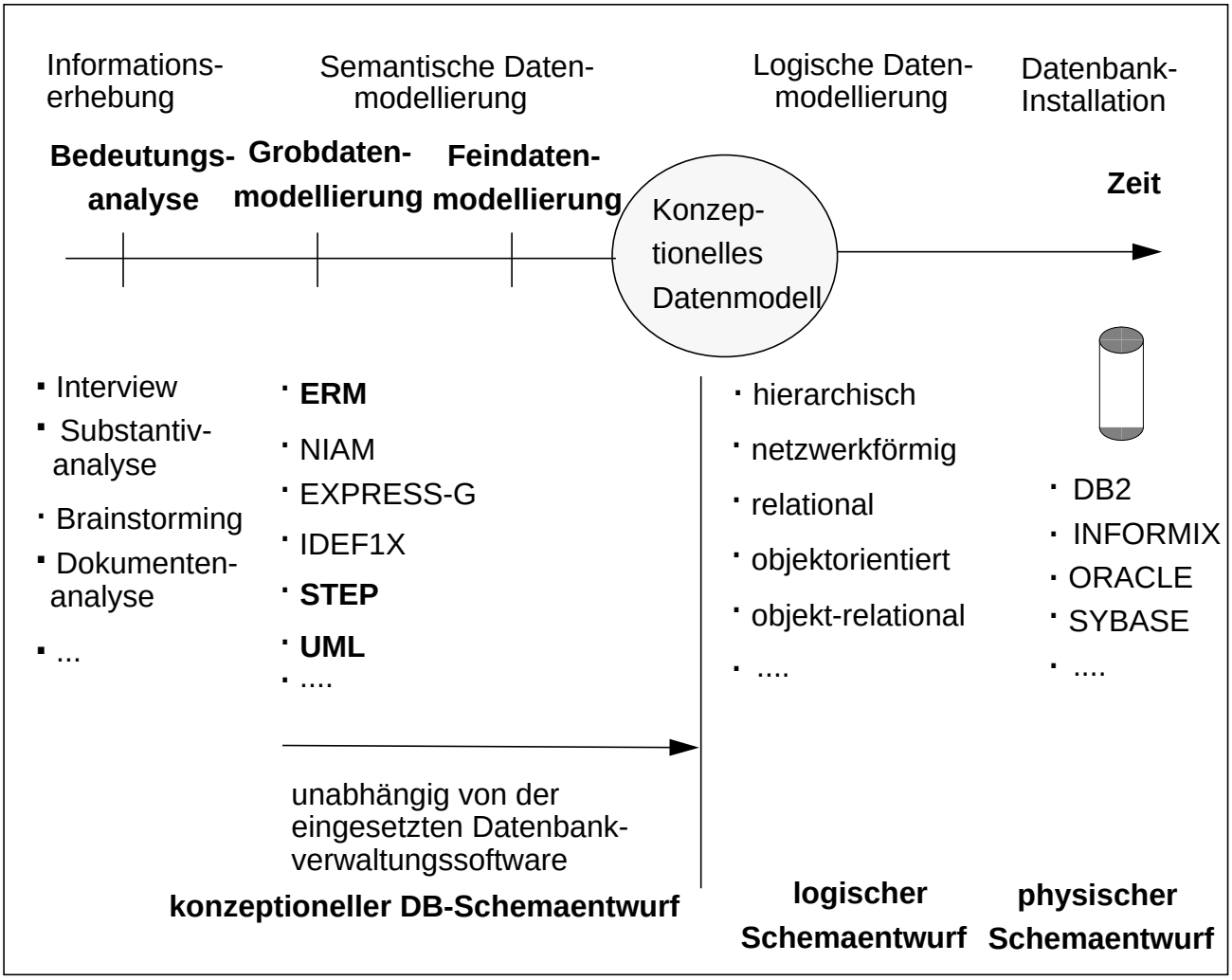


→ **Informationsmodell** (Darstellungselemente & Regeln)**:**
eine Art formale Sprache, um Informationen zu beschreiben

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

Von der Informationserhebung zum DB-Schema

- **Prinzipielle Vorgehensweise**



- **ERM** (**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)
  - 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

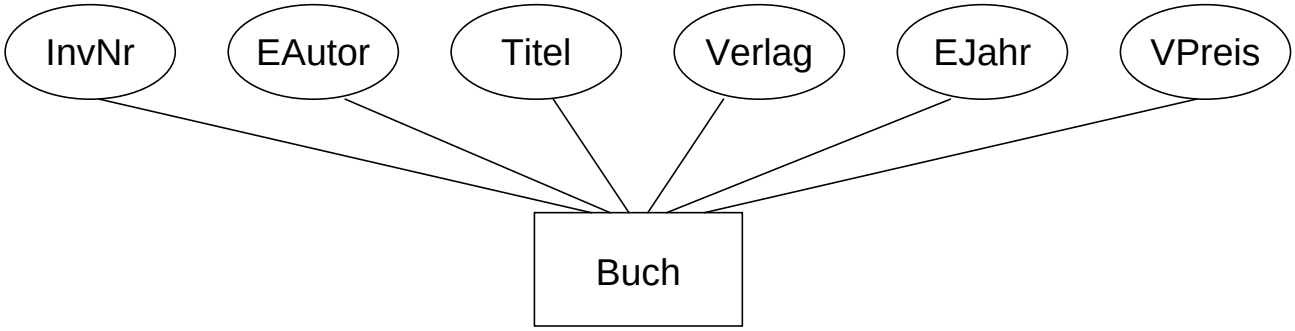
- **Entities**
  - wohlunterscheidbare Dinge der Miniwelt (Diskurswelt)
  - „A thing that has real or individual existence in reality or in mind“ (Webster)
  - besitzen Eigenschaften, deren konkrete Ausprägungen als Werte bezeichnet werden

- **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, Kemper, DBS, Oldenbourg, ...)

e_2 = (0815, Date, Intro. to DBS, Addison, ...)

e_3 = (1234, Härder, DBS, Springer, ...)

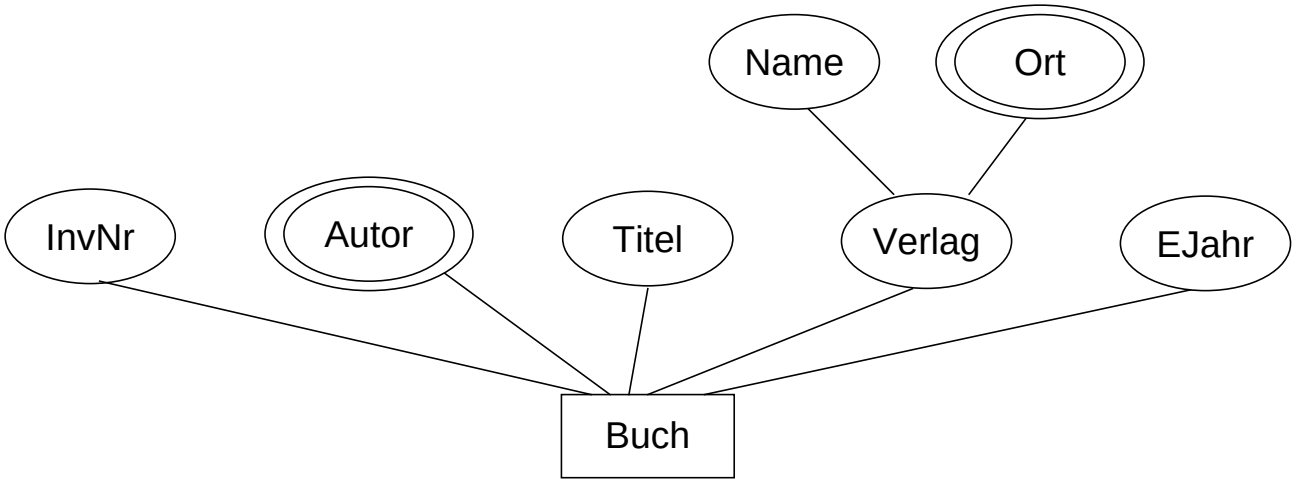
→ Alle Attribute sind einwertig!

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.)

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

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

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

  $e_i \neq e_j \rightarrow$ **K**$(e_i) \neq$ **K**(e_j)

- Mehrere Schlüsselkandidaten (SK) möglich

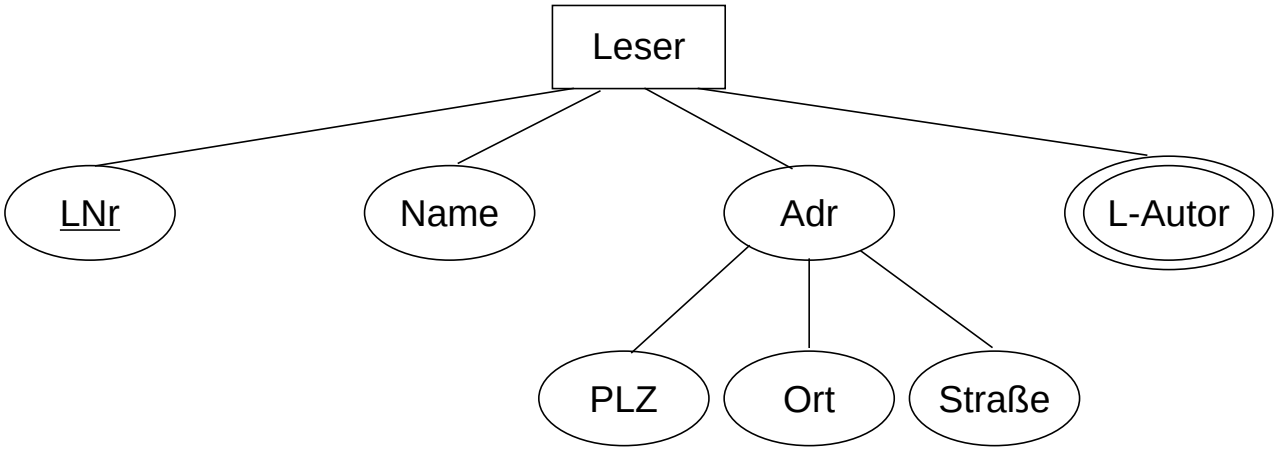
  → **Primärschlüssel** auswählen

- **Beispiel:**

  Entity-Menge **Student** mit Attributen Matnr, 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 (LNr) = int(8), W(Name) = W(L-Autor) = char(30)
  W (PLZ) = int(5), W(Ort) = char(20), W(Straße) = char(15)

  dom (LNr) = int(8)

  dom (Adr) = W(PLZ) x W(Ort) x W(Straße) = int(5) x char(20) x char(15)

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

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

  mit W(C_1), W(C_2), W(E1), W(E_2)

  dom (B) = W(C_1) x W(C_2)

  dom (D) = $2^{W(E1) \times W(E2)}$

  dom (A) = dom(B) x dom(D)

ERM - Definitionen[1]

- **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, ..., B_k)

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

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

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

  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, ..., A_m). A_i sei dom(A_i) ($1 \leq i \leq m$) zugeordnet.

  i) Ein Entity e ist ein Element des Kartesischen Produkts aller Domains, d.h.

  $$e \in dom(A_1) \times ... \times dom(A_m)$$

  ii) Eine Entity-Menge E^t (zum Zeitpunkt t) ist eine Menge von Entities, welche K erfüllt, d.h.

  $$E^t \subseteq dom(A_1) \times ... \times dom(A_m)$$

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

- **Def. 4: Relationships**

  i) Ein Relationship-Typ hat die Form R = (Ent, Y). Dabei ist R der Name des Typs (auch „Name der Beziehung“), 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.

  ii) Sei Ent = (E_1, ..., 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, ..., B_n). Eine Relationship r ist ein Element des Kartesischen Produktes aus allen E_i^t und den Domains der B_j, d.h.

  $$r \in E_1^t \times ... \times E_k^t \times dom(B_1) \times ... \times dom(B_n)$$

  bzw.

  $$r = (e_1, ..., e_k, b_1, ..., b_n)$$

  mit

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

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

  $$R^t \subseteq E_1^t \times ... \times E_k^t \times dom(B_1) \times ... \times dom(B_n).$$

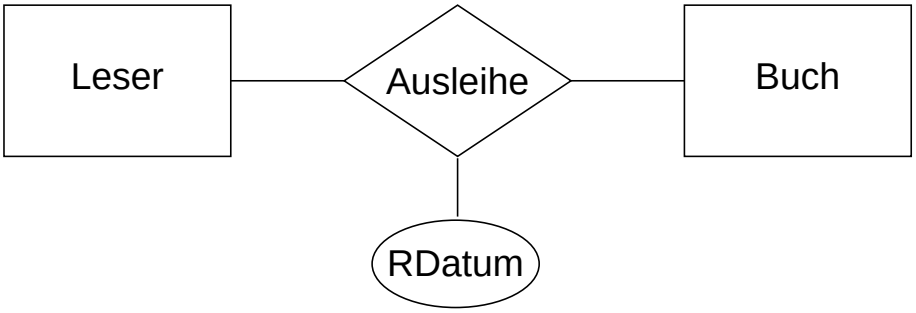
Konzepte des ERM (6)

- **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**



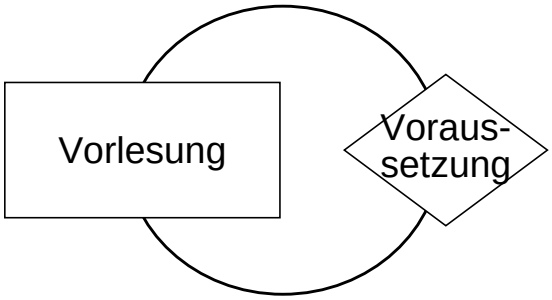
- **Relationship-Typ** R = (Ent, Y)

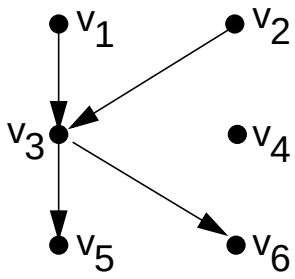
- **Eigenschaften**

  Grad:
  Existenzabhängig:
  Beziehungstyp:

Relationship-Mengen

- **Motivation für Rollennamen**





- **Definition:**

  Voraussetzung $= ((\text{Vorlesung}, \text{Vorlesung}), (\varnothing))$

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

  genauer: direkte Voraussetzung

- Einführung von Rollennamen (rn) möglich (Reihenfolge!)
  auf Typebene: (rn_1/E, rn_2/E) oder (Vorgänger/Vorlesung, Nachfolger/Vorlesung)

  auf Instanzebene: (Vorgänger/v_i, Nachfolger/v_j)

  Sprechweise: „ v_j setzt v_i voraus“

  oder „ v_i ist-Voraussetzung-für v_j“

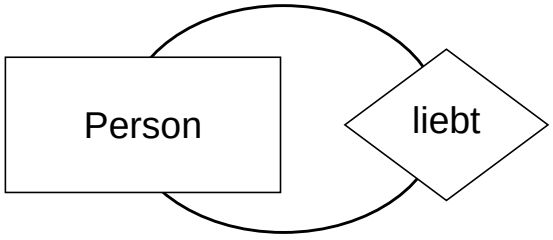
- **Eigenschaften:**

  Grad:

  Existenzabhängig:

  Beziehungstyp:

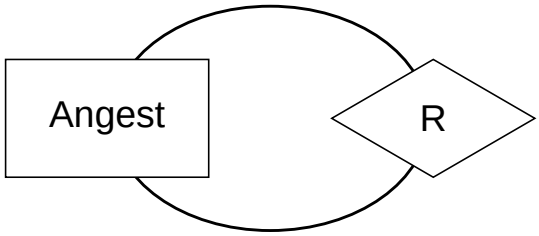
- **Transitivität gilt bei Selbstreferenz i. allg. nicht!**

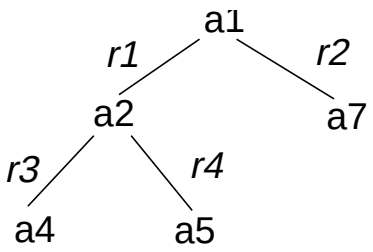


Relationship-Mengen (2)

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

  Direkter-Vorgesetzter = ((Angest/Angest, Chef/Angest), (∅))



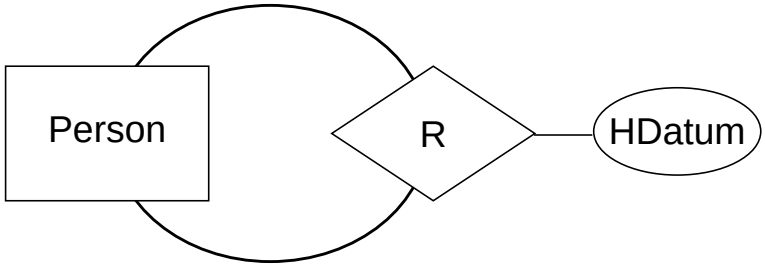


- **Eigenschaften**

  Grad:
  Existenzabhängig:
  Beziehungstyp:

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

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

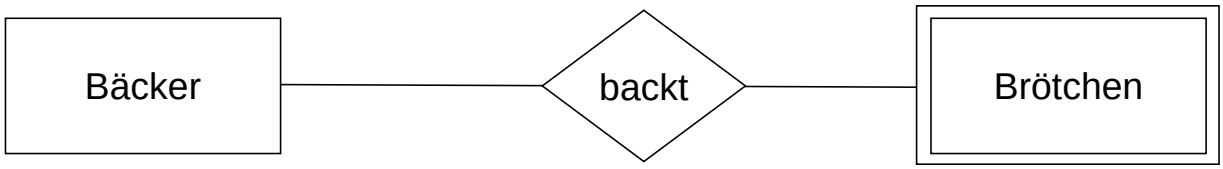


- **Eigenschaften**

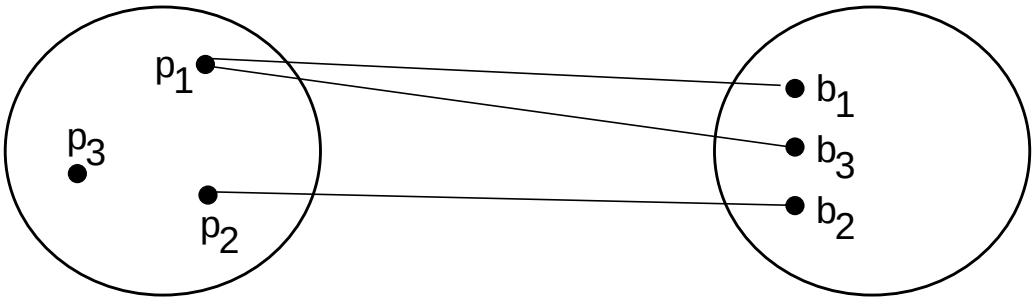
  Grad:
  Existenzabhängig:
  Beziehungstyp:

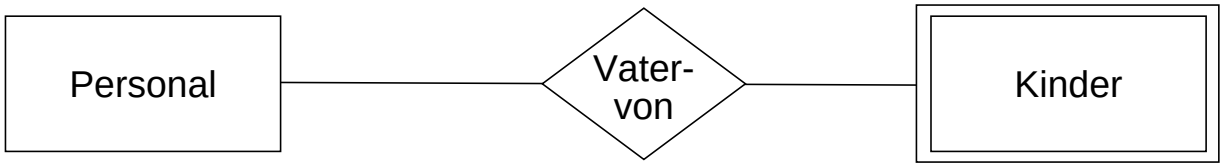
Relationship-Mengen (3)

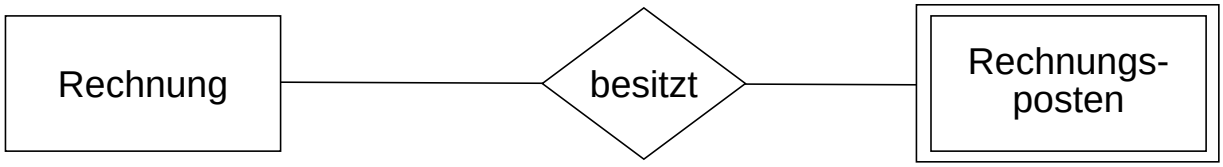
- **Existenzabhängigkeit einer Entity-Menge**
- **Beispiele**



Existenzabhängigkeit: „Relationship begründet Existenz von“





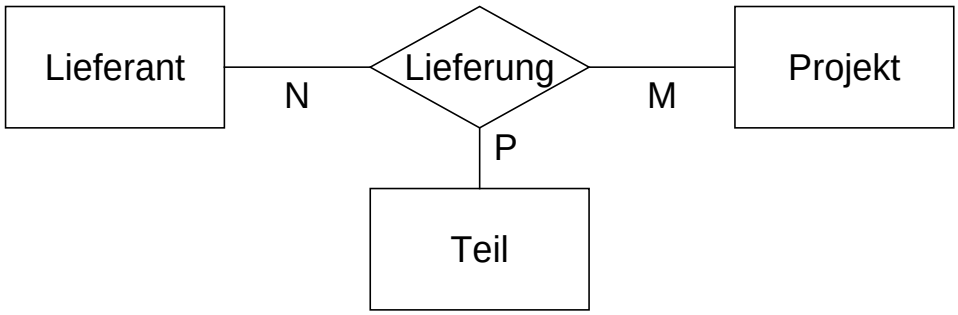


- **Eigenschaften**

Grad:	2
Existenzabhängig:	ja
Beziehungstyp:	1 : n

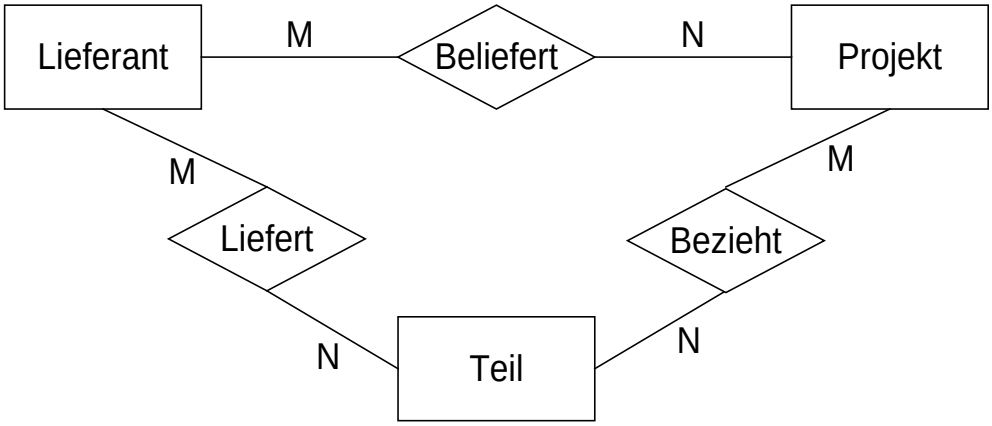
- **Bem.:** In manchen Modellen steht eine existenzabhängige Entity-Menge rechts von der selbständigen Entity-Menge und der „erzeugenden“ Relationship-Menge. Bei Mehrfachreferenzen ist eine „erzeugende” von weiteren „referenzierenden” Relationship-Mengen zu unterscheiden.

Dreistellige Relationship-Mengen



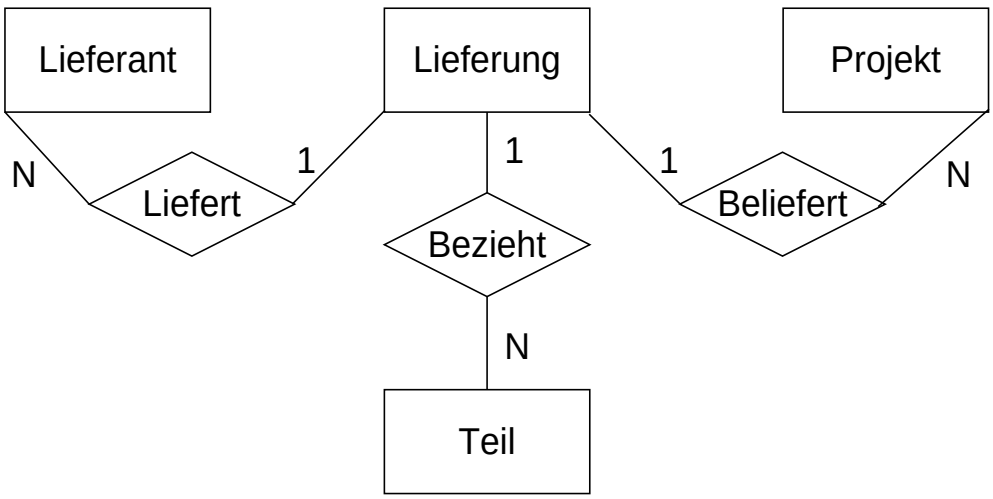
Achtung:

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



Aber:

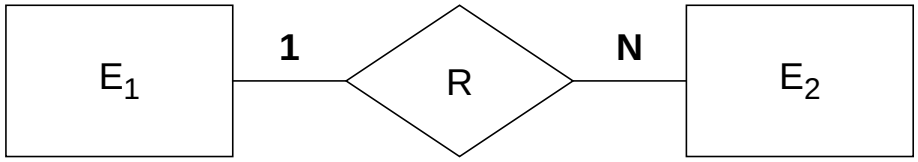
Manche Systeme erlauben nur die Modellierung binärer Relationship-Mengen!



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
  - 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
  - 1:n ... ARBEITET_FÜR/MIT: MITARBEITER $\rightarrow$ PROF
  - n:m ... BESCHÄFTIGT/IST_HIWI: PROF — STUDENT

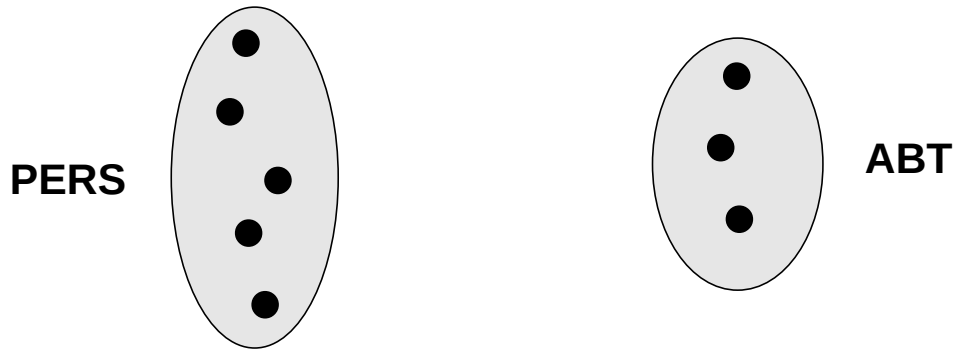
  ➞ Abbildungstypen implizieren nicht, daß für jedes $e_k \in E_i$ auch tatsächlich ein $e_l \in E_j$ existiert
- **Diagrammdarstellung**:



Klassifikation von Datenabbildungen (2)

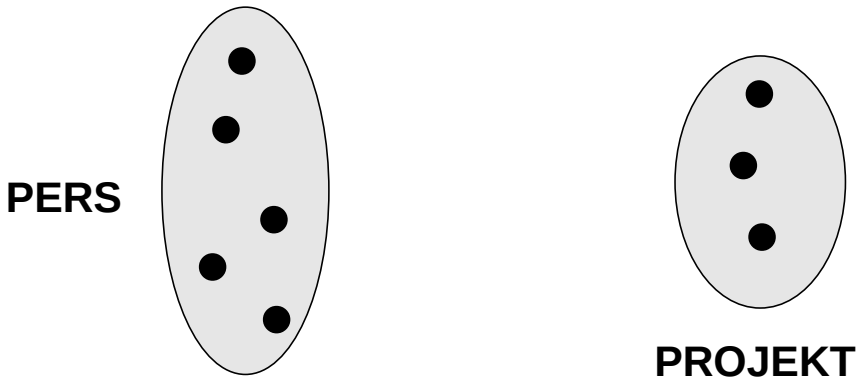
- **Beispiele zu E_i – E_j (externe Klassifikation)**

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



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

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



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

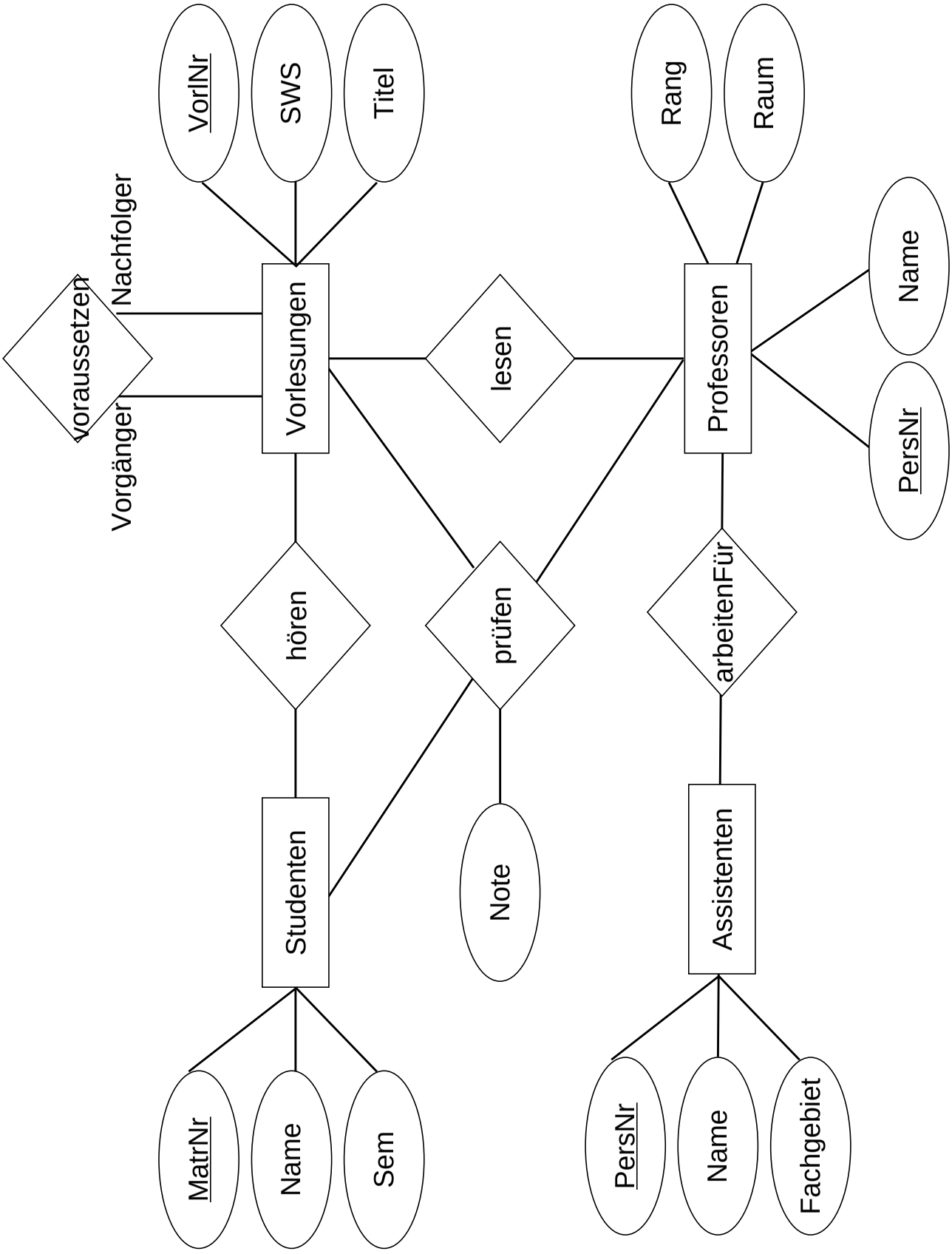
DECLARE REGULAR ENTITY RELATION PROJEKT
ATTRIBUTE/VALUE-SET:
PRO-NR/PROJEKT-NR
PRO-ORT/ORTE
PRIMARY KEY:
PRO-NR

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

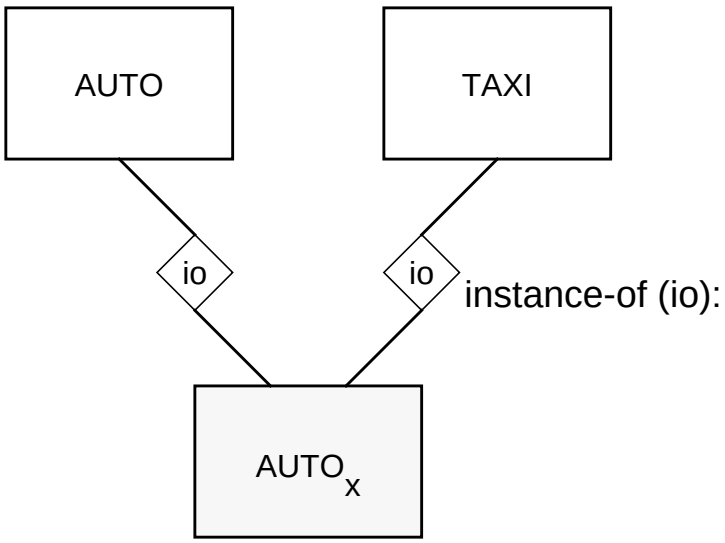
ER-Diagramm - Vorlesungsbetrieb



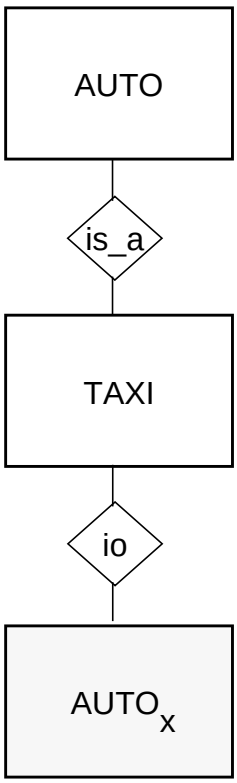
Erweiterungen des ERM

- **„Alles dreht sich um die genauere Modellierung von Beziehungen“**

- **Beispiel:** Unangemessene Modellierung bei überlappenden EM



instance-of (io):



- **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**)

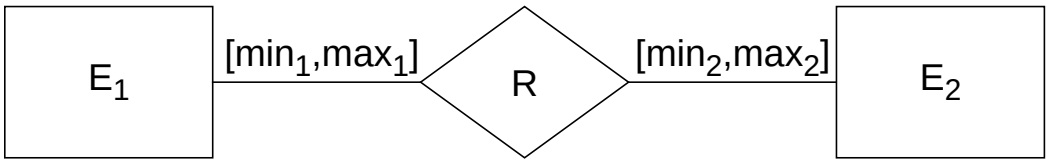
Verfeinerung der Datenabbildung: Kardinalitätsrestriktionen

- 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:



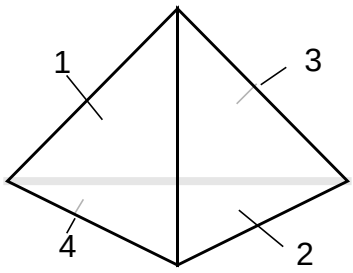
e_1 nimmt an $[min_1,max_1]$ Beziehungen von Typ R teil
e_2 nimmt an $[min_2,max_2]$ Beziehungen von Typ R teil

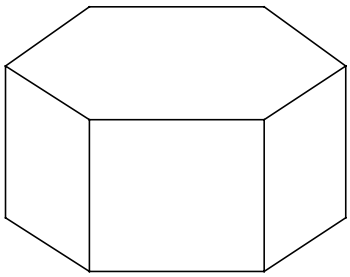
- **Beispiele**:

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		
V.Teilnahme	VORL	STUDENT		
Mitarbeit	PERS	PROJEKT		

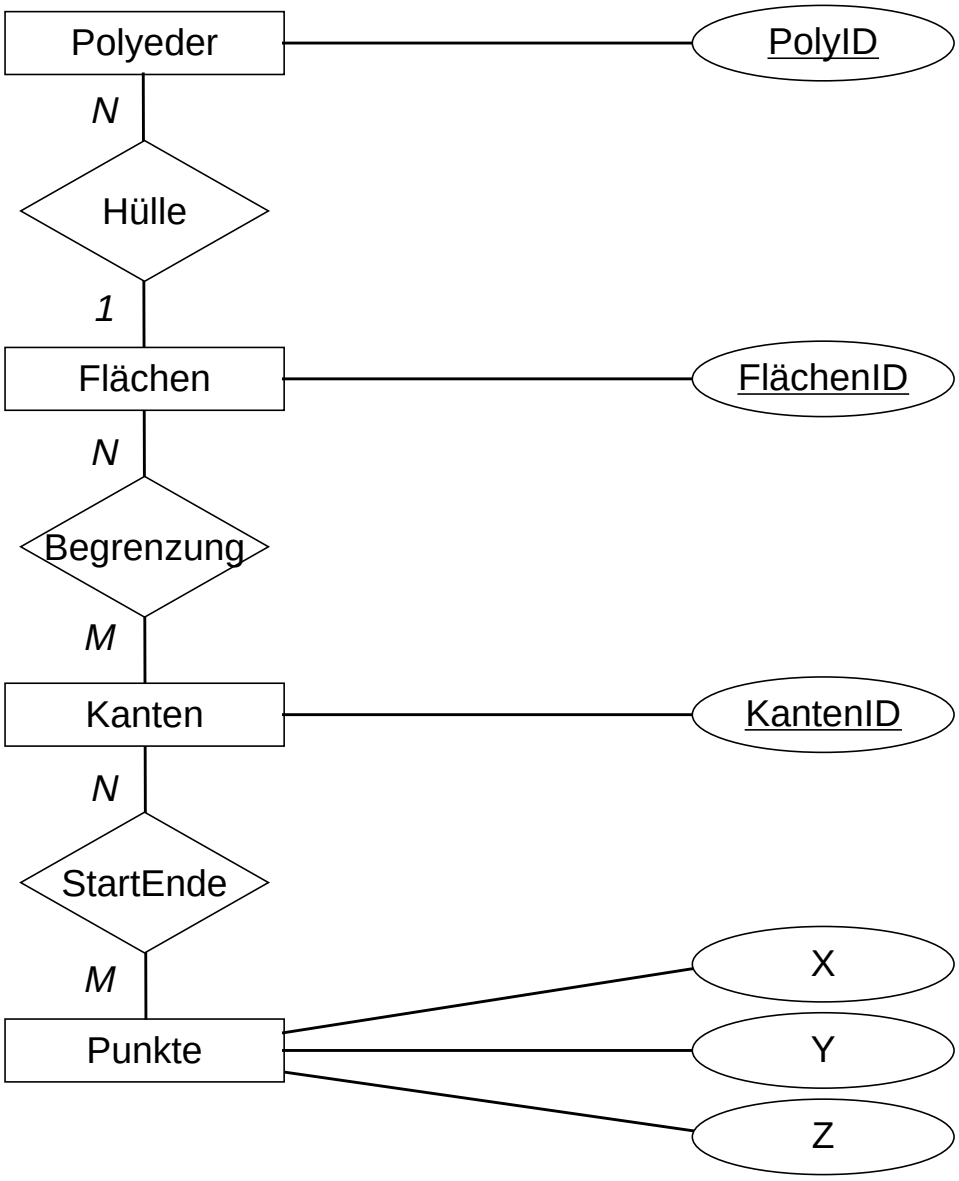
Begrenzungsflächendarstellung von Körpern

Beispiel-Körper:



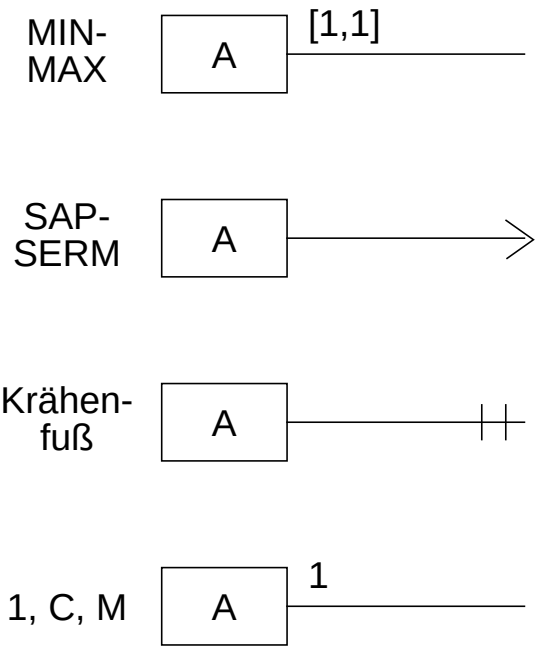


ER-Diagramm:

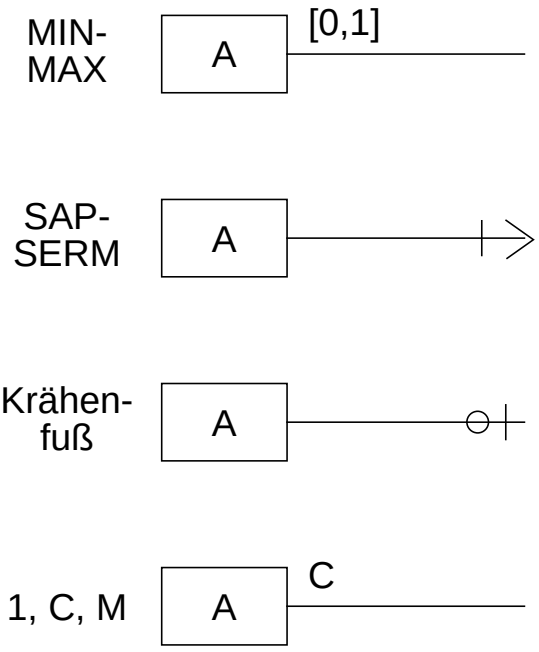


Notationen

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

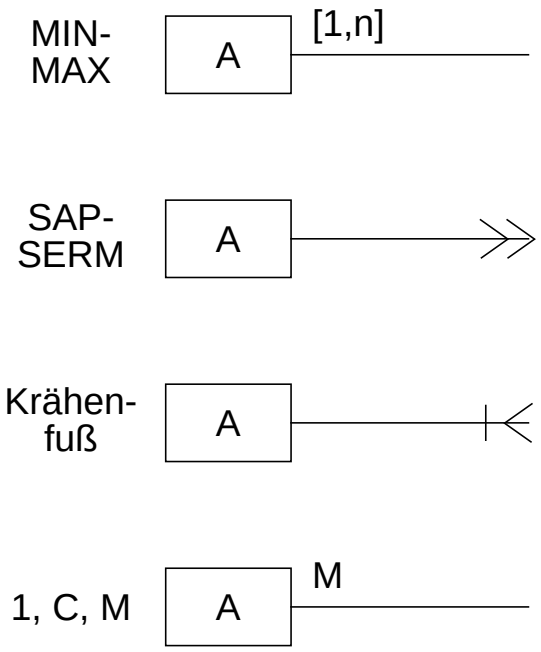


- **Jedes Element von A nimmt an höchstens einer Beziehung teil**

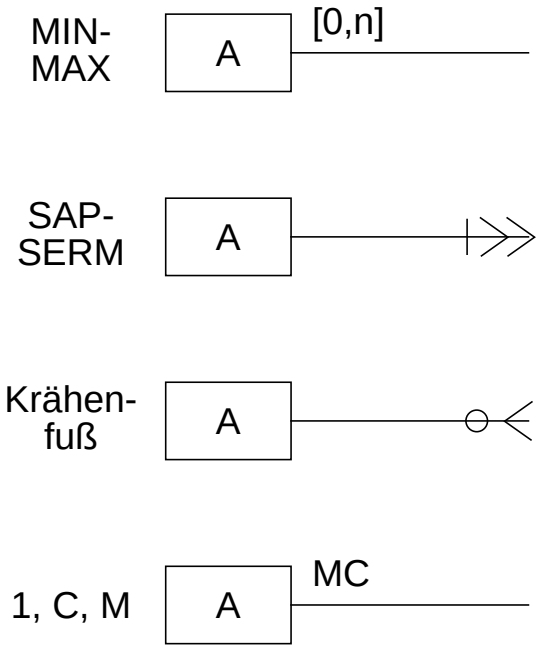


Notationen (2)

- **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 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

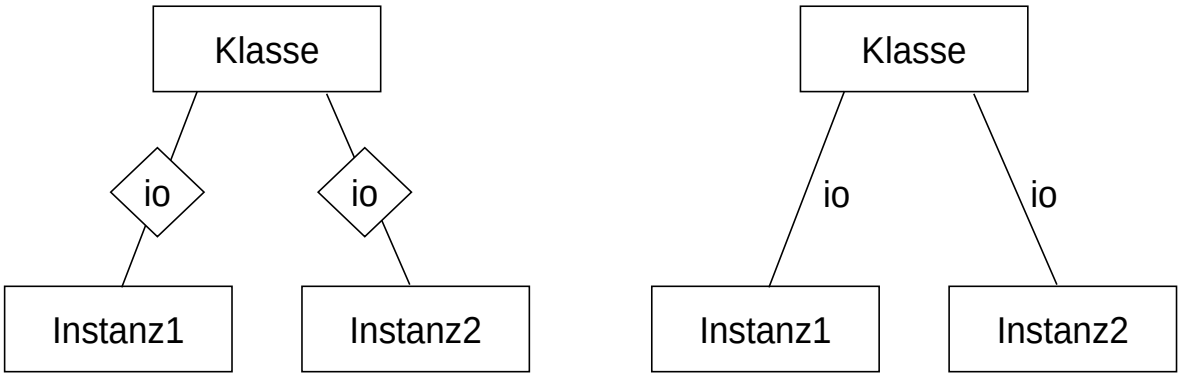
- Eine Klasse ist definiert als Zusammenfassung von Objekten
  gleichen Typs (und gleicher Repräsentation).
  Dadurch nur einmalige Definition von
  - Attributnamen und -typen
  - Methoden
  - Integritätsbedingungen

- 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**



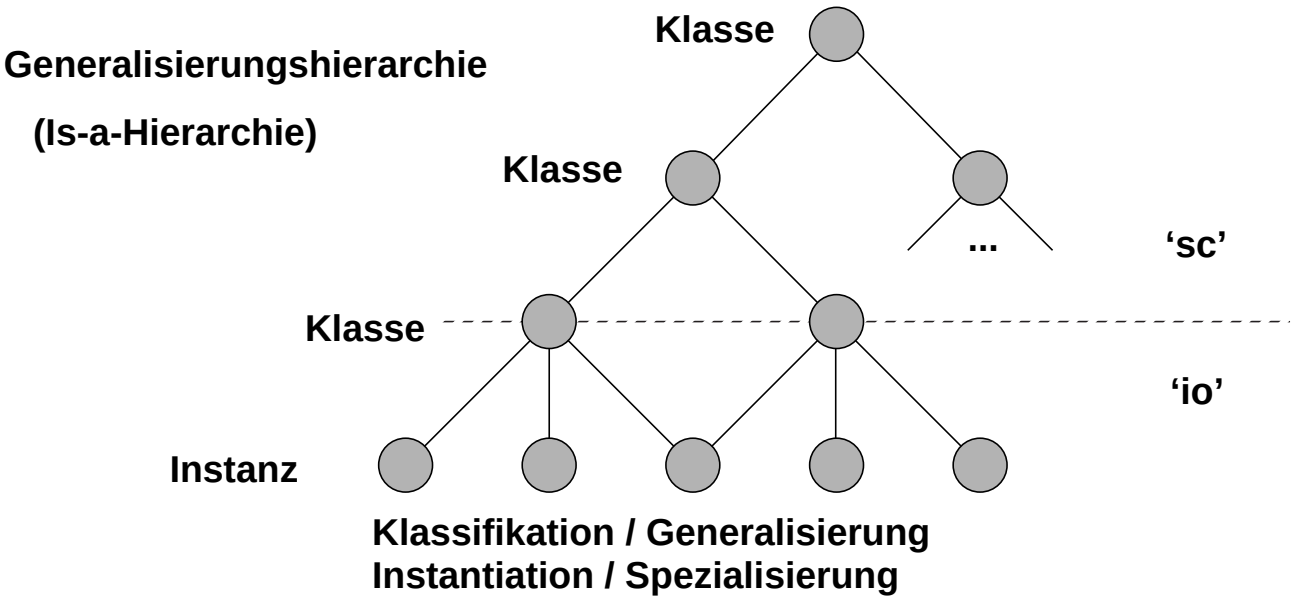
Die Darstellungen der anderen Abstraktionskonzepte erfolgen entsprechend.

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

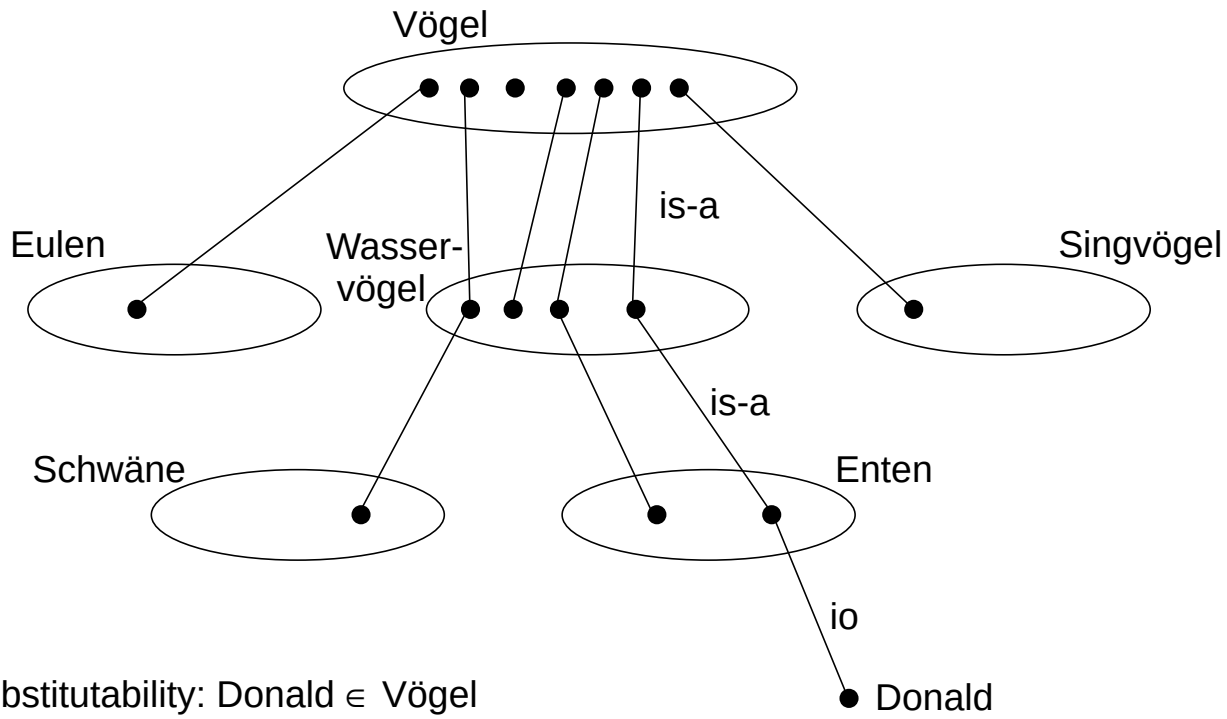
- **Anwendung**
  - Sie baut die **'subclass-of'**-Beziehung auf (**‘sc’**- oder **'is-a'**-Beziehung)
  - Sie ist rekursiv anwendbar (n-Ebenen-Beziehung) und organisiert die Klassen in einer Generalisierungshierarchie
  - Eine Superklasse ist eine Verallgemeinerung/Abstraktion der zugehörigen Subklassen. 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 ($\rightarrow$ Netzwerke, (n:m) !)
  - Zugehörigkeit eines Objektes zu einer Klasse/Superklasse wird im wesentlichen bestimmt durch **Struktur** (Attribute)**, Verhalten** (Operationen) und **Integritätsbedingungen** der Klasse/Superklasse



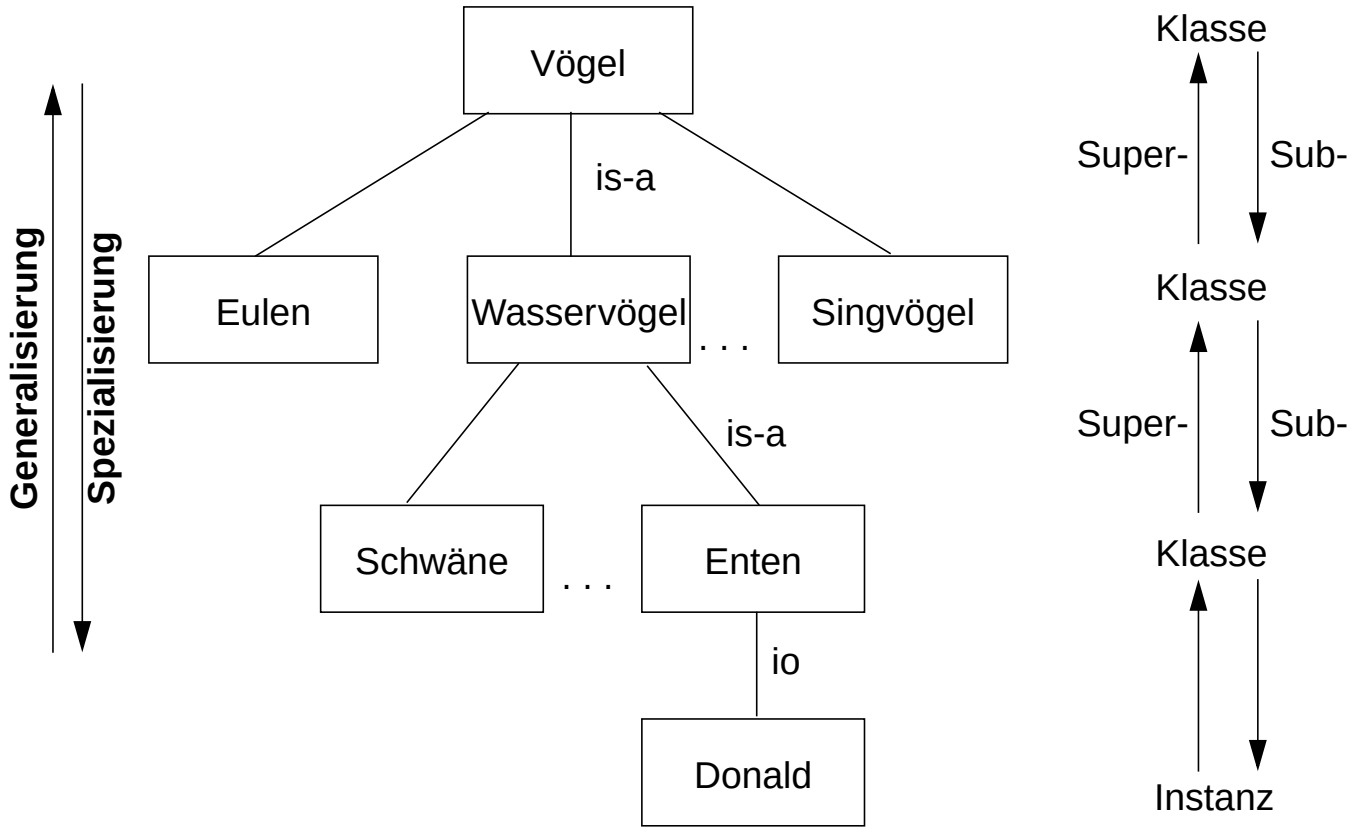
Modellierungsbeispiel zur Generalisierung

Instanzendarstellung



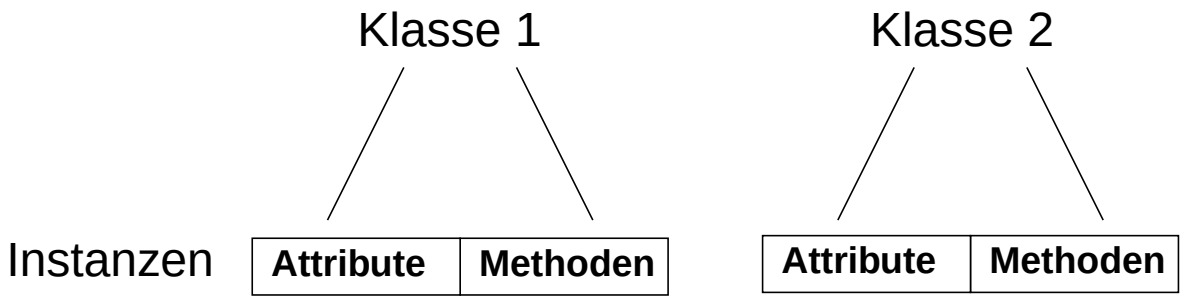
Substitutability: Donald $\in$ Vögel

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

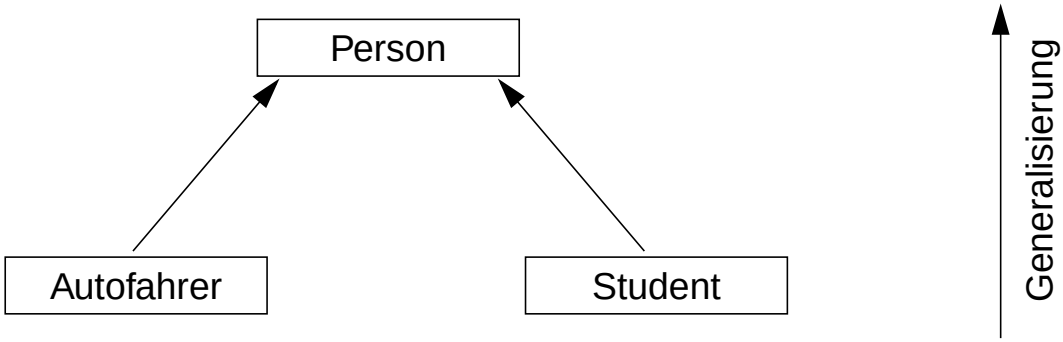
- jede Instanz der Subklasse ist immer auch Instanz der Klasse, aber nicht umgekehrt
- jede Methode, die auf die Instanzen einer Klasse anwendbar ist, ist damit immer auch auf die Instanzen sämtlicher Subklassen anwendbar

→ Garantie von bestimmten Integritätsbedingungen durch das System:

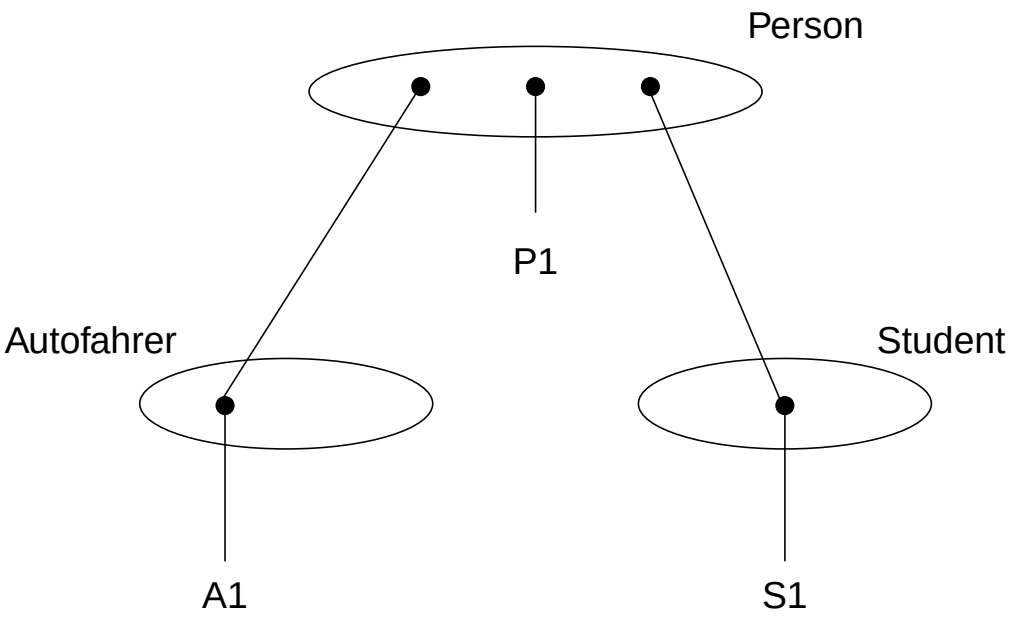
jeder Student ist auch Person

Generalisierung - Beispiel

- **Klassen**



- **Instanzen**

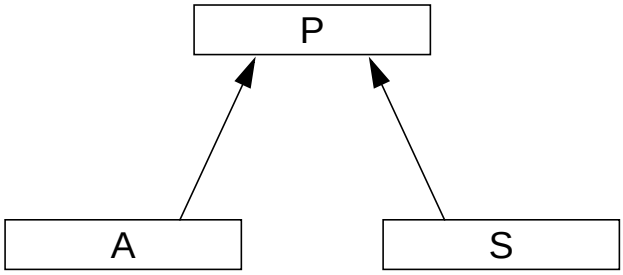


- **Subklassen sind i. allg. nicht disjunkt!**

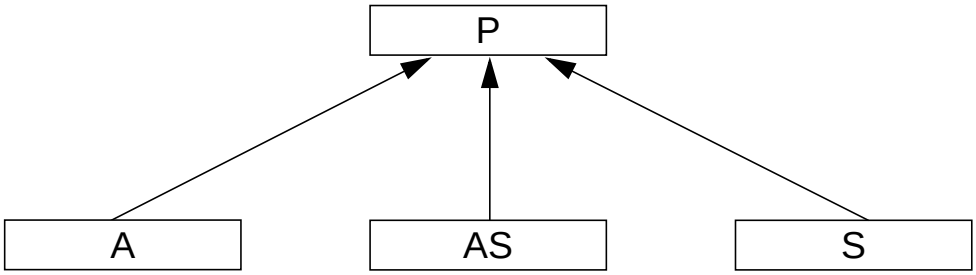
Generalisierung - Beispiel (2)

- **Überlappende Subklassen - Ansätze:**

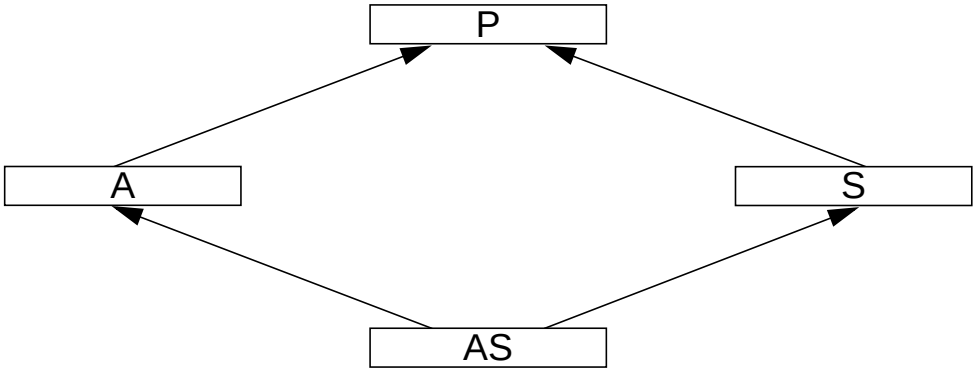
1. **Mehrklassenmitgliedschaft von Instanzen**



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



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



Generalisierung (3)

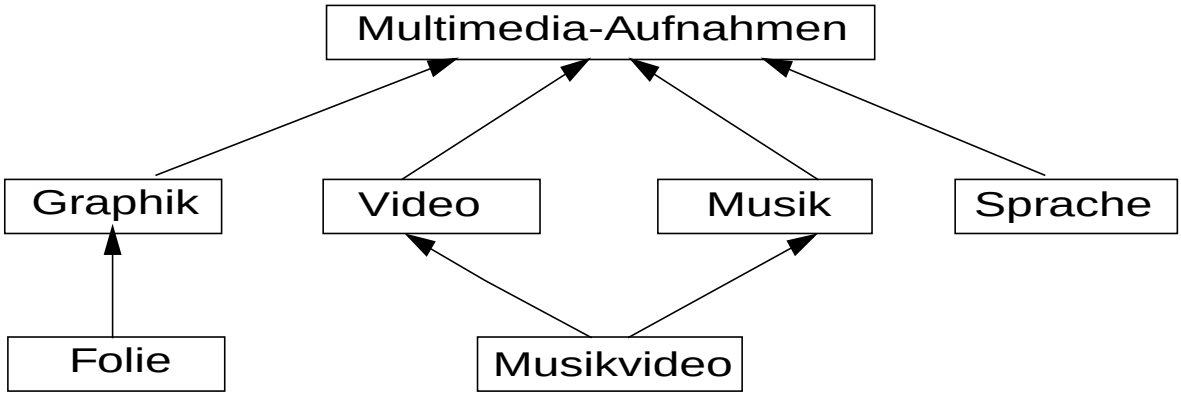
- **historisches Vorbild: Carl von Linné**
  - biologische Systematik
  - Reich (z.B. Tierreich) – Stamm (Chordatiere) – Klasse (Säugetiere) – Familie – Gattung – Art

    ➞ 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.

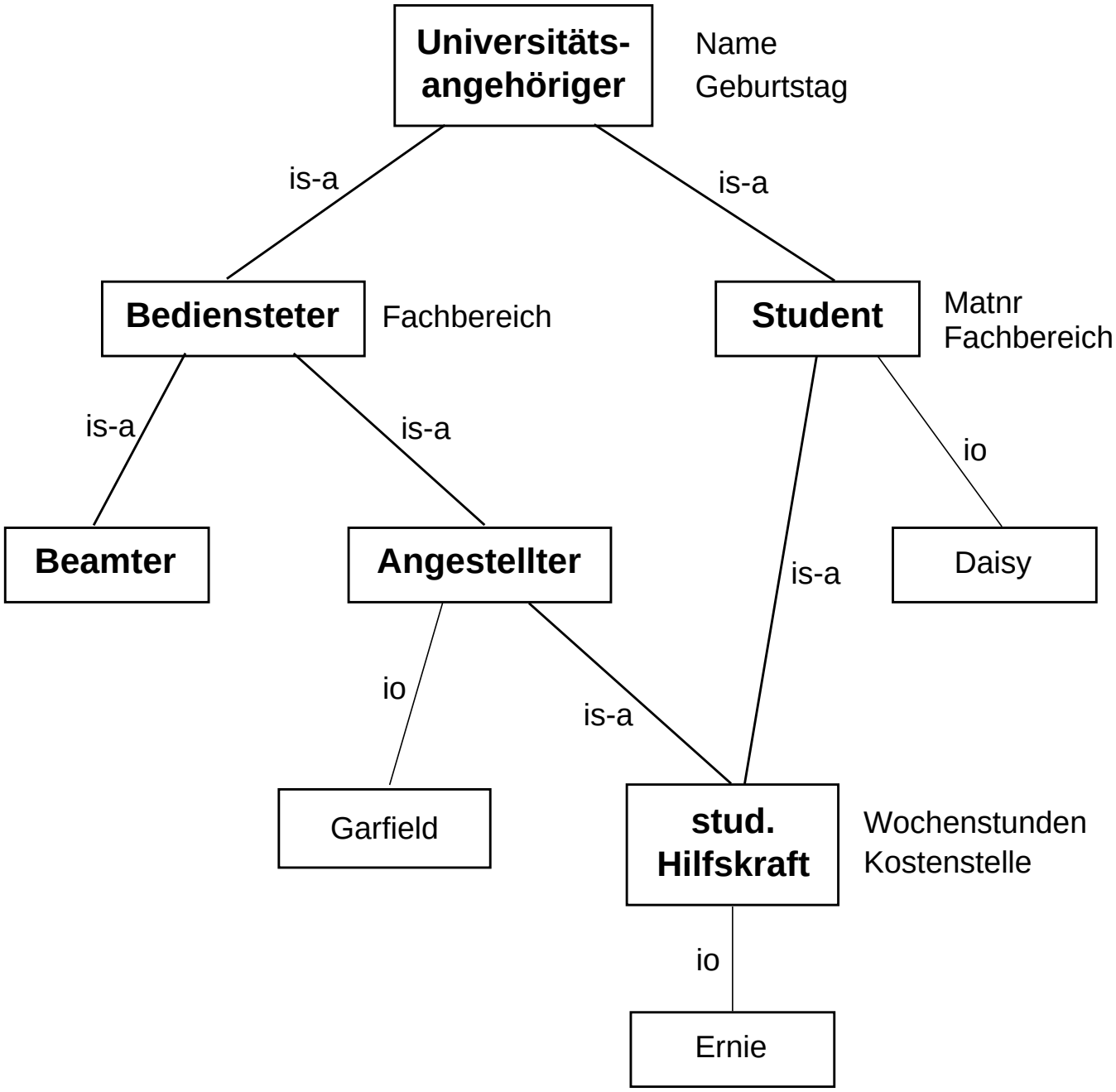
- **Arten der Vererbung**
  - Einfach-Vererbung (eindeutig)
  - Mehrfach-Vererbung

- **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

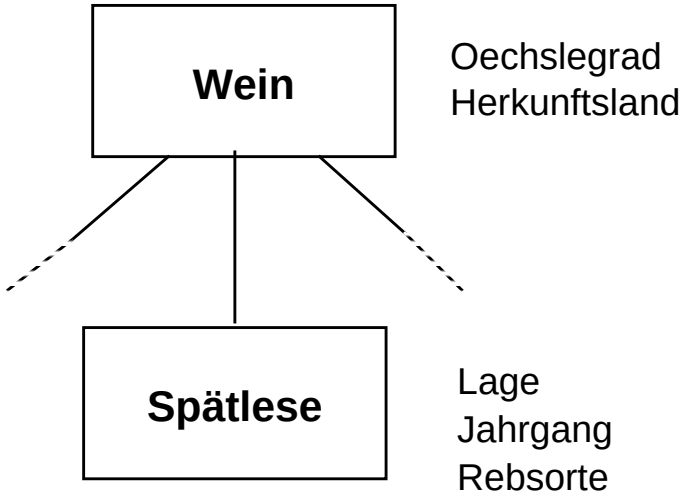


- **Mehrfach-Vererbung** (*multiple inheritance*) kann zu Konflikten führen

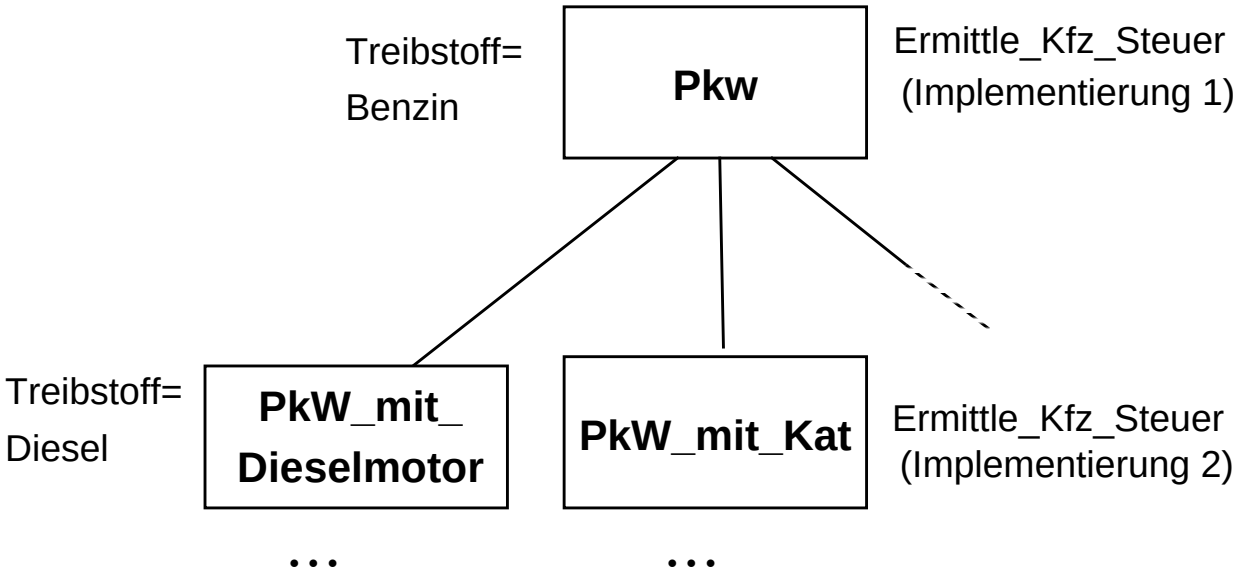
  → Auflösung explizit durch den Benutzer, z. B. durch Umbenennung:
  Hiwi_im_Fachbereich → Fachbereich of Angestellter
  immatrikuliert_im_Fachbereich → Fachbereich of Student

Vererbung (2)

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



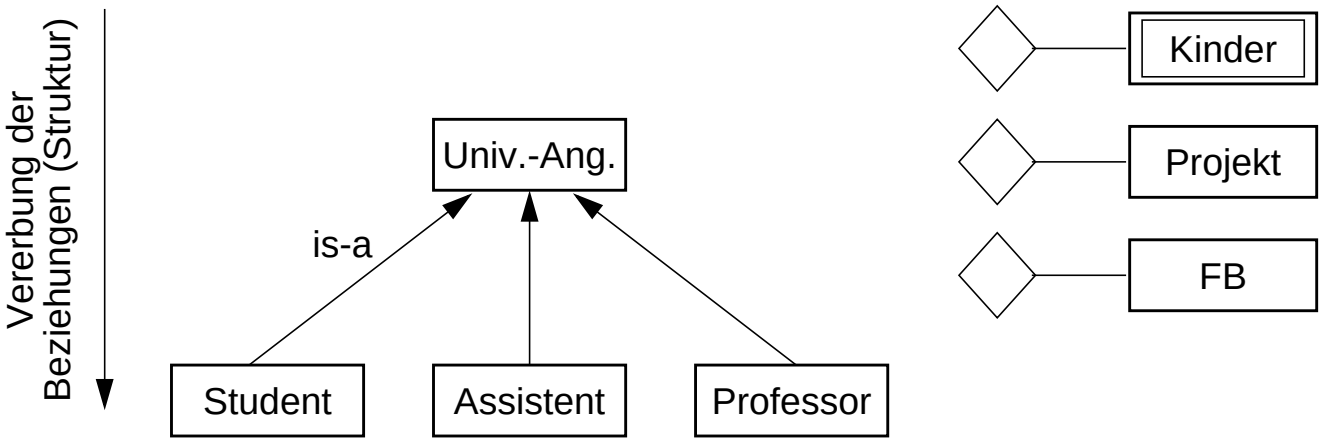
- Subklasse kann ererbte Attributwerte **überschreiben** oder Methoden **überladen:**



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

Vererbung (3)

- **Vererbung von Beziehungen:**

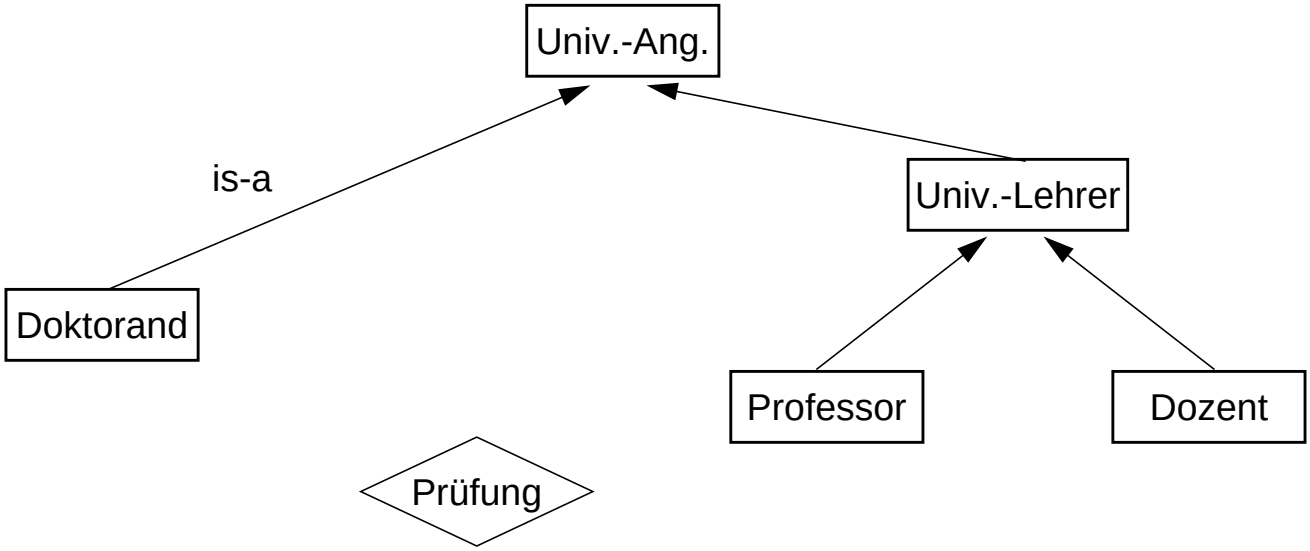


- **Beispiel: Doktorprüfung**

  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 (total)**, falls gilt

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

  andernfalls **partiell**.

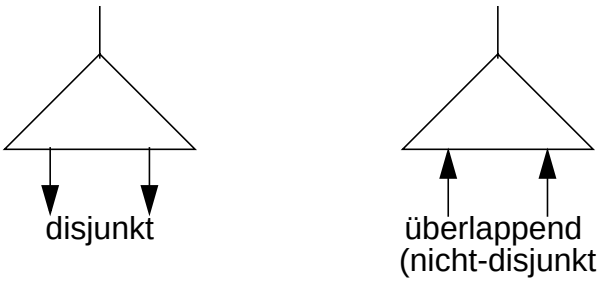
  Z ist **disjunkt**, falls

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

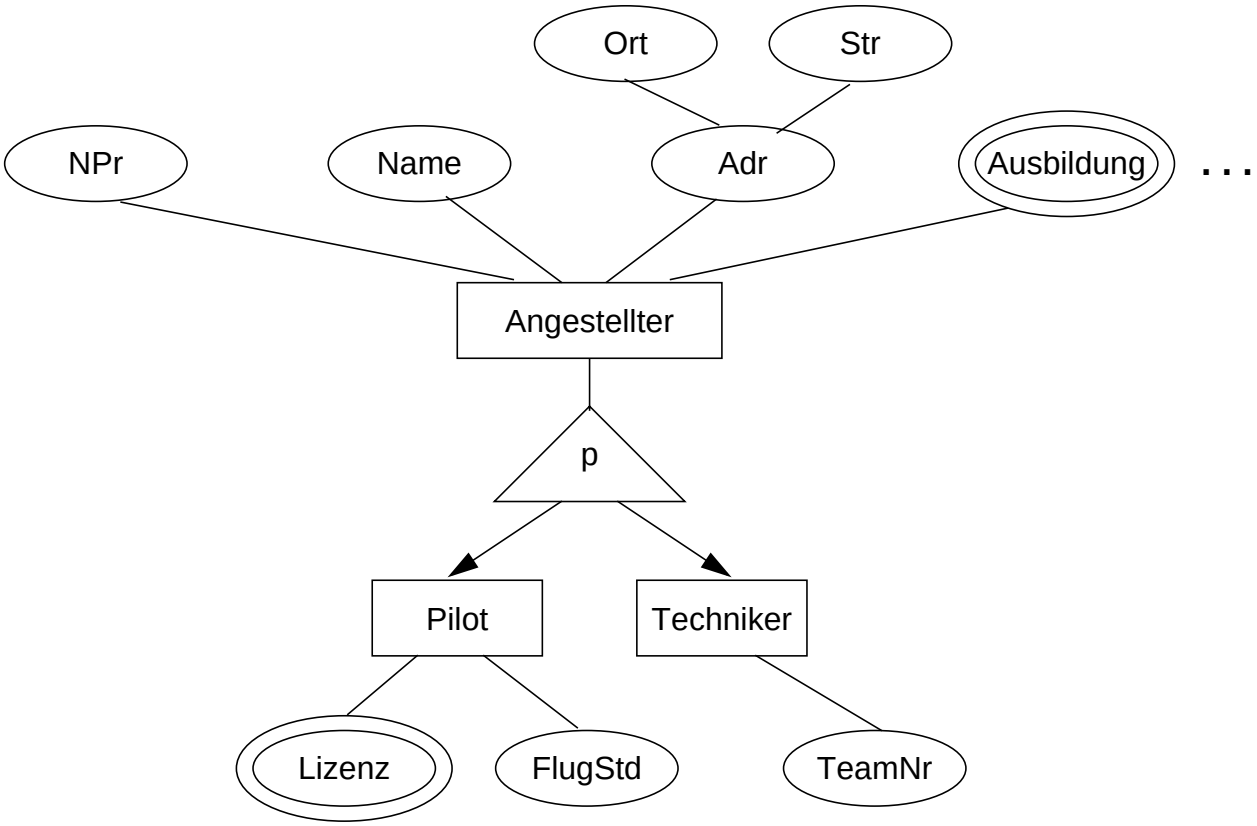
  andernfalls **überlappend (nicht-disjunkt)**.

Arten von Spezialisierungen

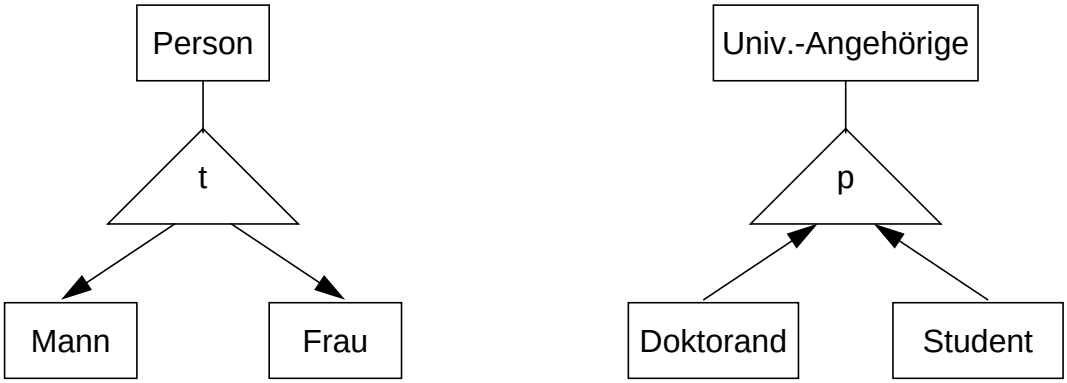
- **Verfeinerung der is-a-Beziehung**



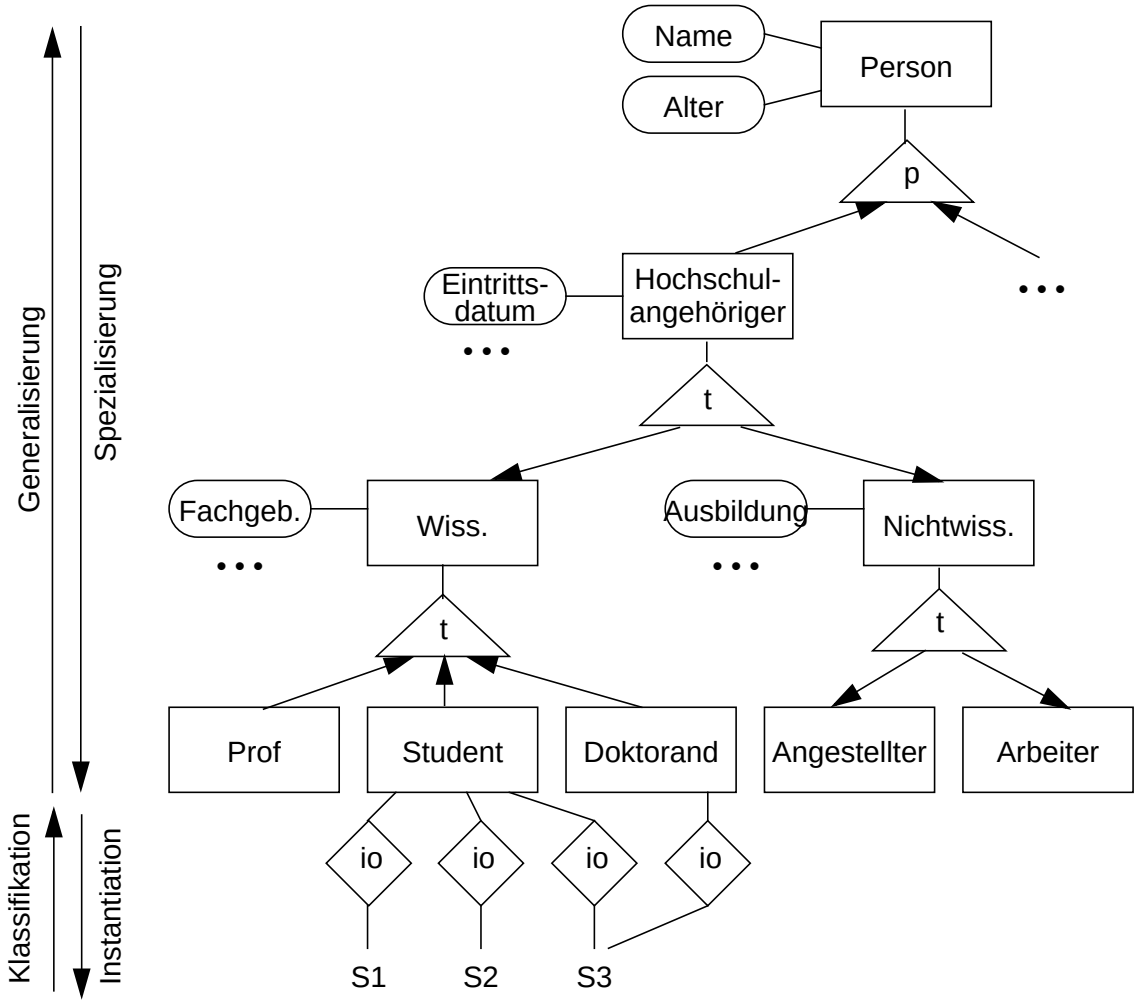
- **Partielle, disjunkte Spezialisierung**



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

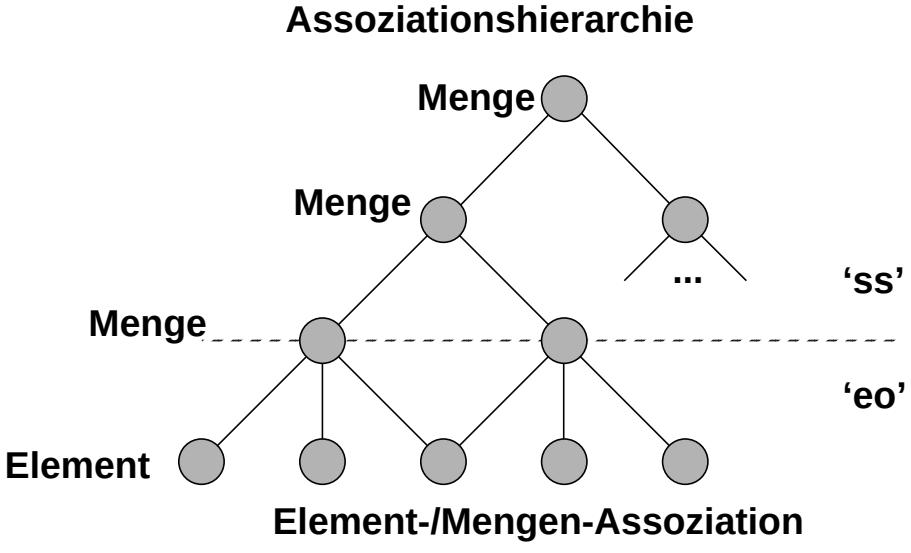
- **Anwendung**

  - 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

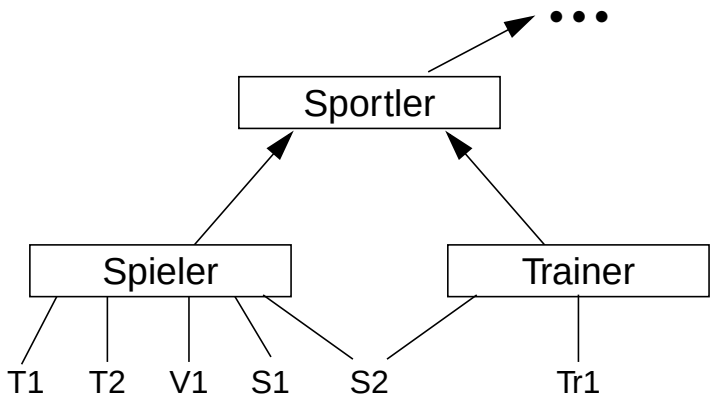
    → Netzwerke, (n:m) !

- **Systemkontrollierte Ableitungen bei der Assoziation**

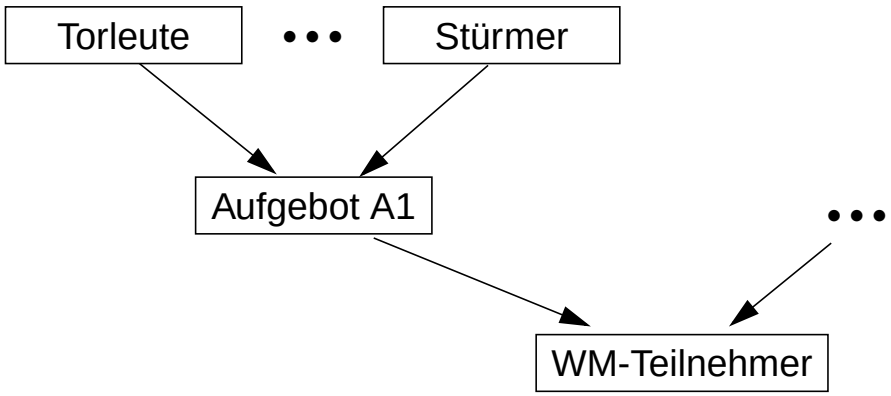
  - Sie unterstützt keine Vererbung, da die Mengeneigenschaften keine Element-eigenschaften sind
  - Durch **Mitgliedschaftsimplikation** lassen sich Eigenschaften bestimmen, die jedes gültige Element der Menge erfüllen muß
  - **Mengeneigenschaften** sind Eigenschaften der Menge, die über Element-eigenschaften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

Assoziation - Beispiel

- **Generalisierungshierarchie**



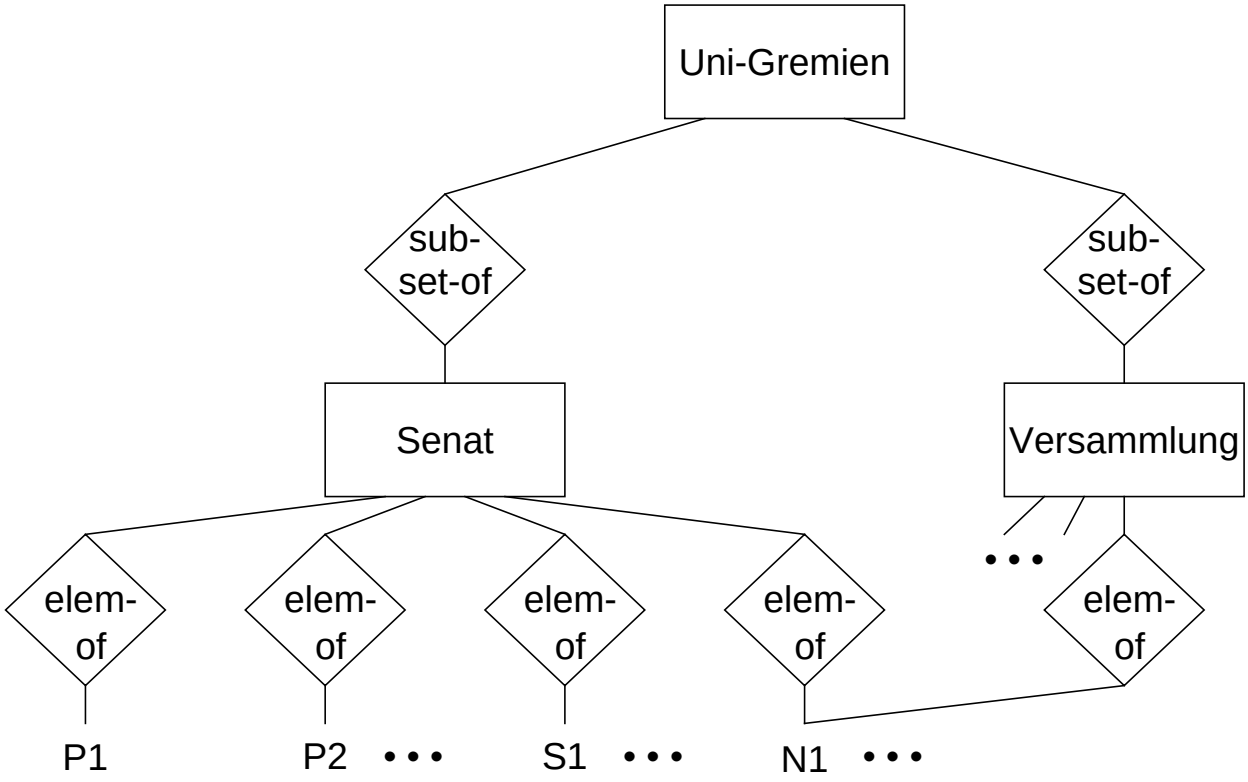
Assoziation



- **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**

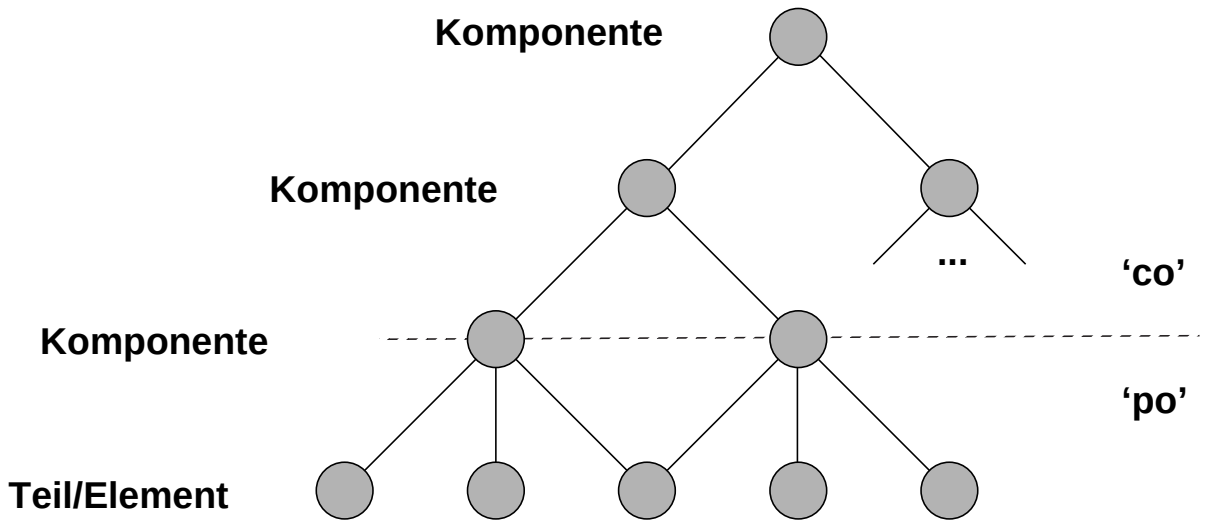
  - Eine Kollektion von einfachen Objekten (Element-Objekt, **Teil**) wird als zusammengesetztes Objekt (**Komponenten**objekt/**Aggregat**objekt) behandelt

  - 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

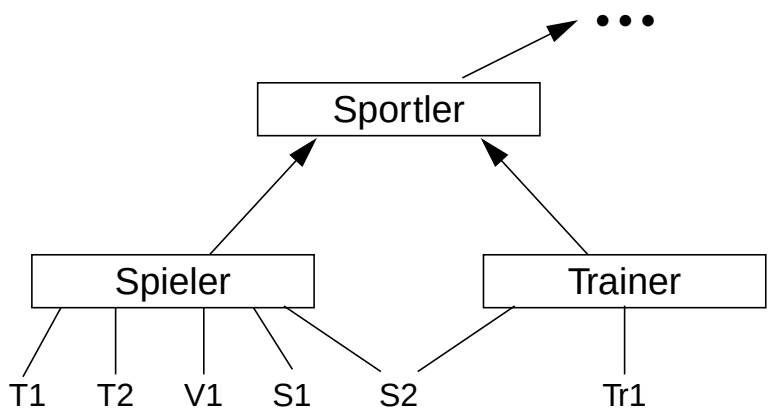
    → Unterschied zu Klassen und Mengenobjekten, die ohne Instanzen existieren können, bzw. für die leere Mengen erlaubt sind

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

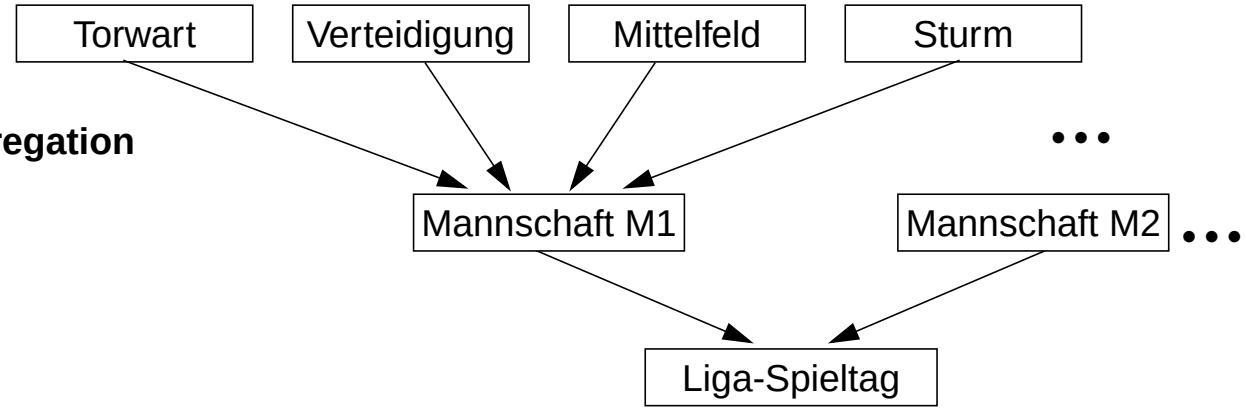
    → Netzwerke, ((n:m) !)

Aggregation - Beispiel

- **Generalisierungshierarchie**

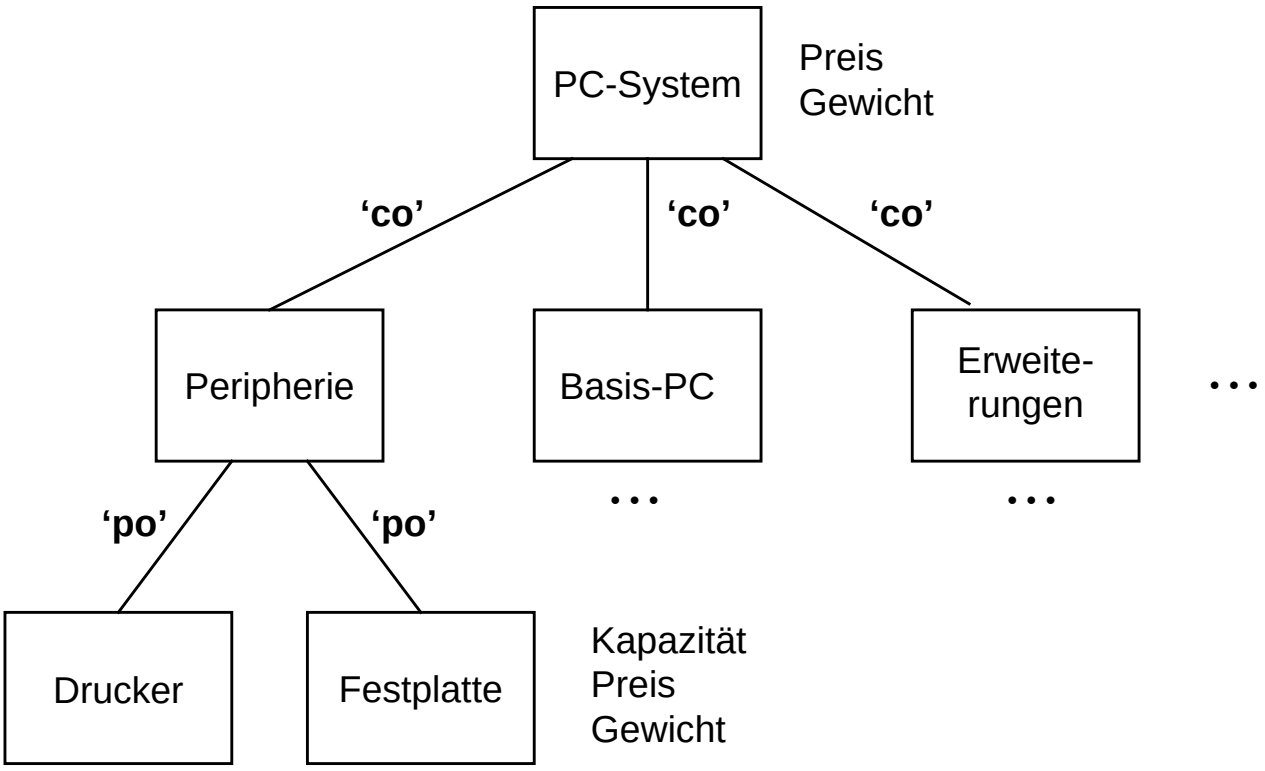


Aggregation



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

Aggregation - Beispiel (2)



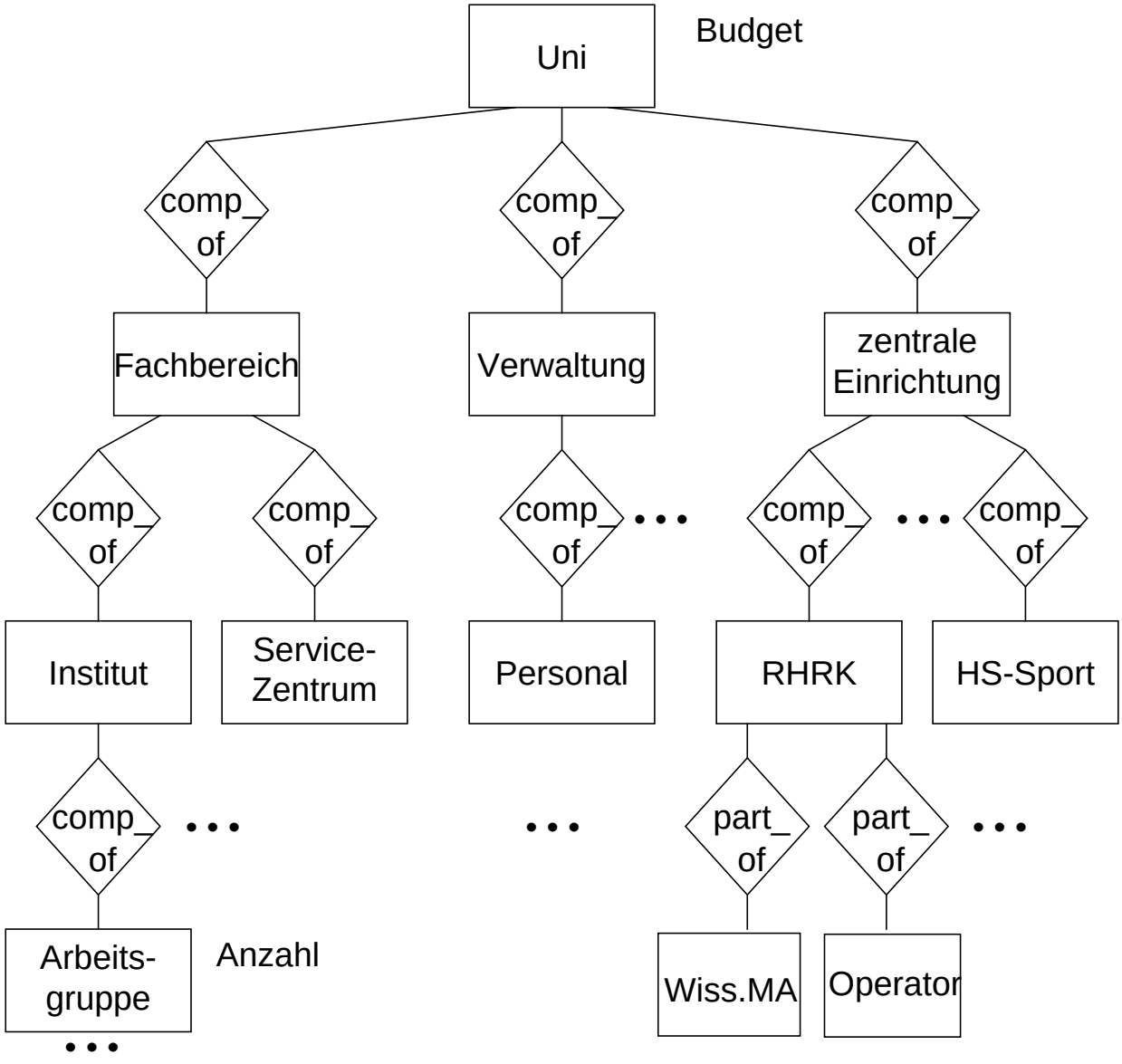
- **Systemkontrollierte Ableitungen: implizierte Prädikate**
  - Prädikate, die über der Aggregationshierarchie spezifiziert sind und gemeinsame Eigenschaften von Elementen/Aggregaten betreffen
  - ‘upward implied predicate’

    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

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

deklarative Attribute

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

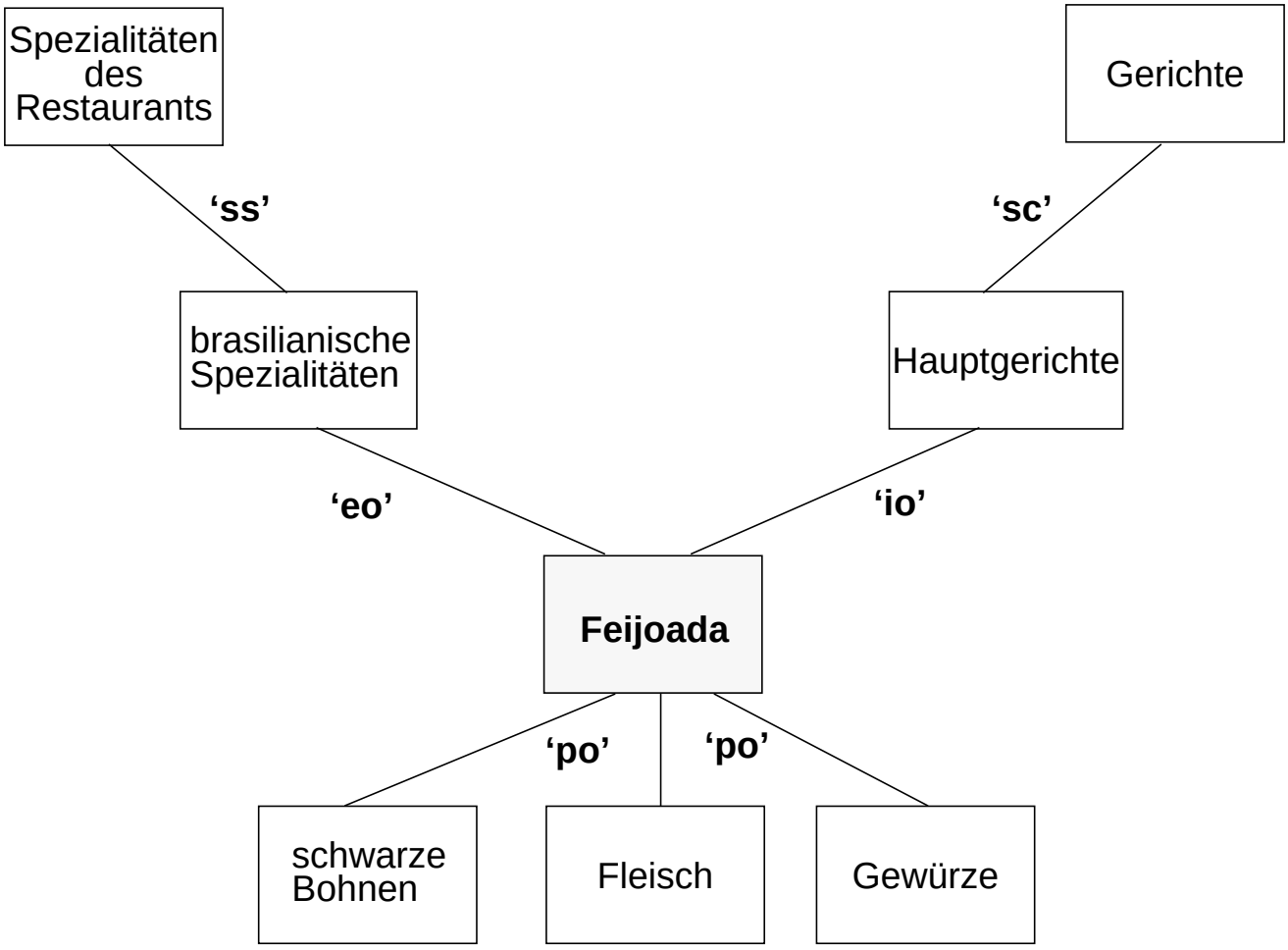
geeignete-Getränke: trockener Rotwein, Bier
- possible-values: instance-of Getränke
- cardinality
- .
- .
- .

prozedurale Attribute

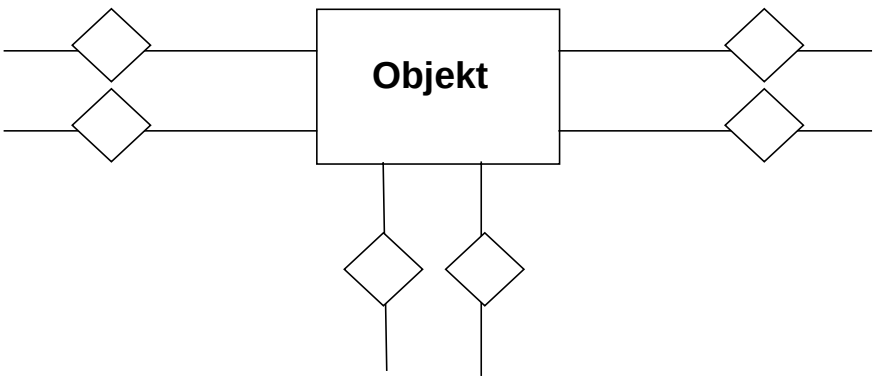
Bestellen (Anzahl-von-Personen)
- procedure BEGIN ... END
- .
- .
- .

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



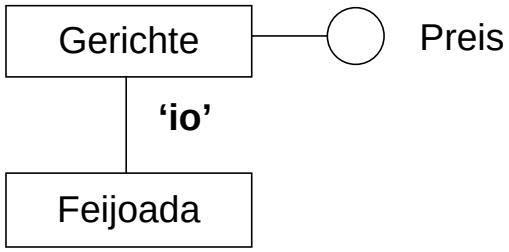
Darstellungsprinzip:



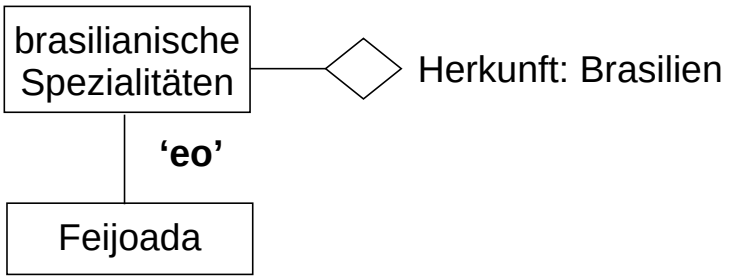
Modellinhärentes „Reasoning“

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

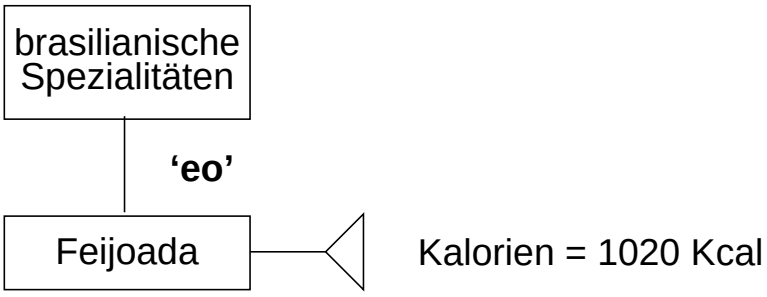
Vererbung



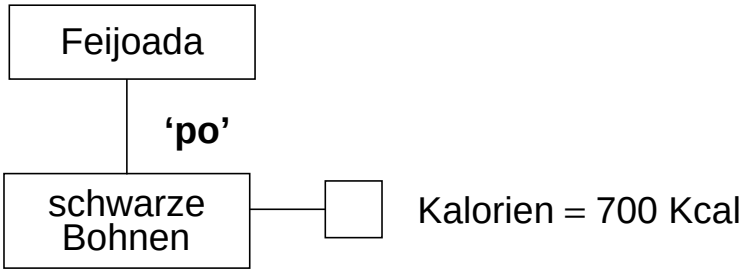
Mitgliedschafts-implikationen



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
  - 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

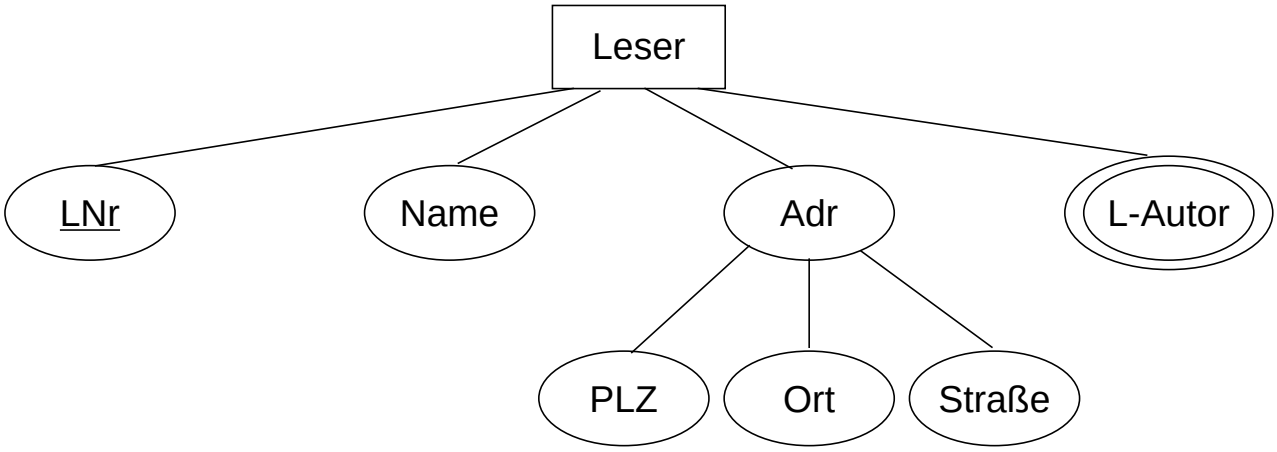
    → relativ karges Informationsmodell

- **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 Mitgliedschaftsimplikationen
  - Aggregation und implizierte Prädikate
  - Integration der Abstraktionskonzepte mittels objektzentrierter Darstellung

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 (LNr) = int(8), W(Name) = W(L-Autor) = char(30)
  W (PLZ) = int(5), W(Ort) = char(20), W(Straße) = char(15)

  dom (LNr) = int(8)

  dom (Adr) = W(PLZ) x W(Ort) x W(Straße) = int(5) x char(20) x char(15)

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

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

  mit W(C_1), W(C_2), W(E1), W(E_2)

  dom (B) = W(C_1) x W(C_2)

  dom (D) = $2^{W(E1)\ x\ W(E2)}$

  dom (A) = dom(B) x dom(D)

Zusammenfassung

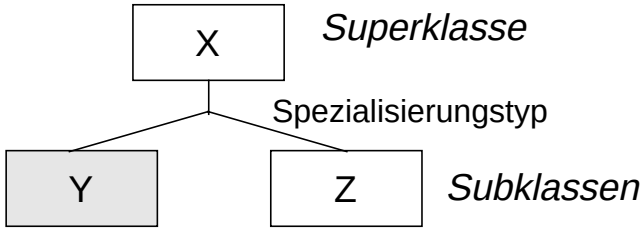
- **DB-Entwurf umfaßt**
  - Informationsbedarfsanalyse
  - konzeptionelles DB-Schema (-> Informationsmodell)
  - logisches DB-Schema
  - physisches DB-Schema (nicht diskutiert)

- **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

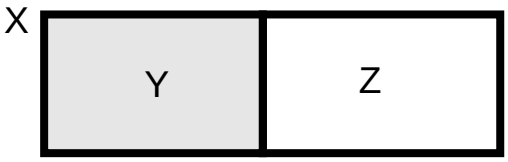
    → relativ karges Informationsmodell

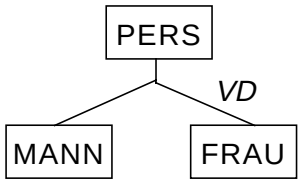
- **Einführung weiterer Modellierungskonzepte**
  - Verfeinerung von Beziehungen durch Kardinalitätsrestriktionen und vor allem Abstraktionskonzepte
    (hier insbesondere Generalisierung/Spezialisierung und Vererbung)
  - Das erweiterte ERM ist sehr mächtig und umfaßt viele bekannte Modellierungskonzepte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

Arten von Spezialisierungen

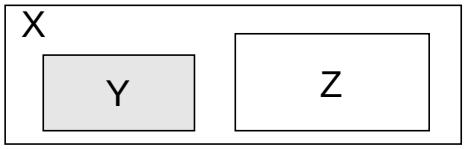


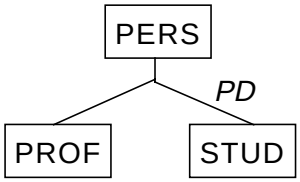
1. vollständig, disjunkt (VD)



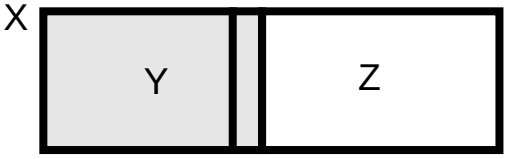


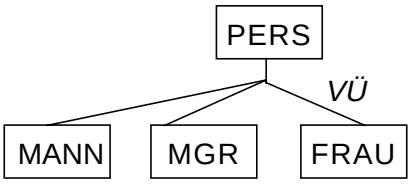
2. partiell, disjunkt (PD)



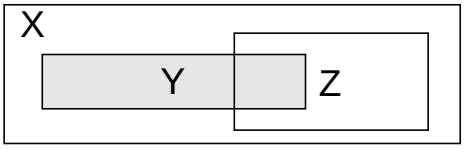


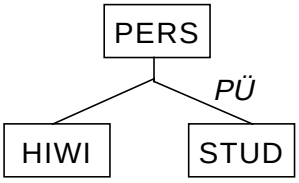
3. vollständig, überlappend (VÜ)





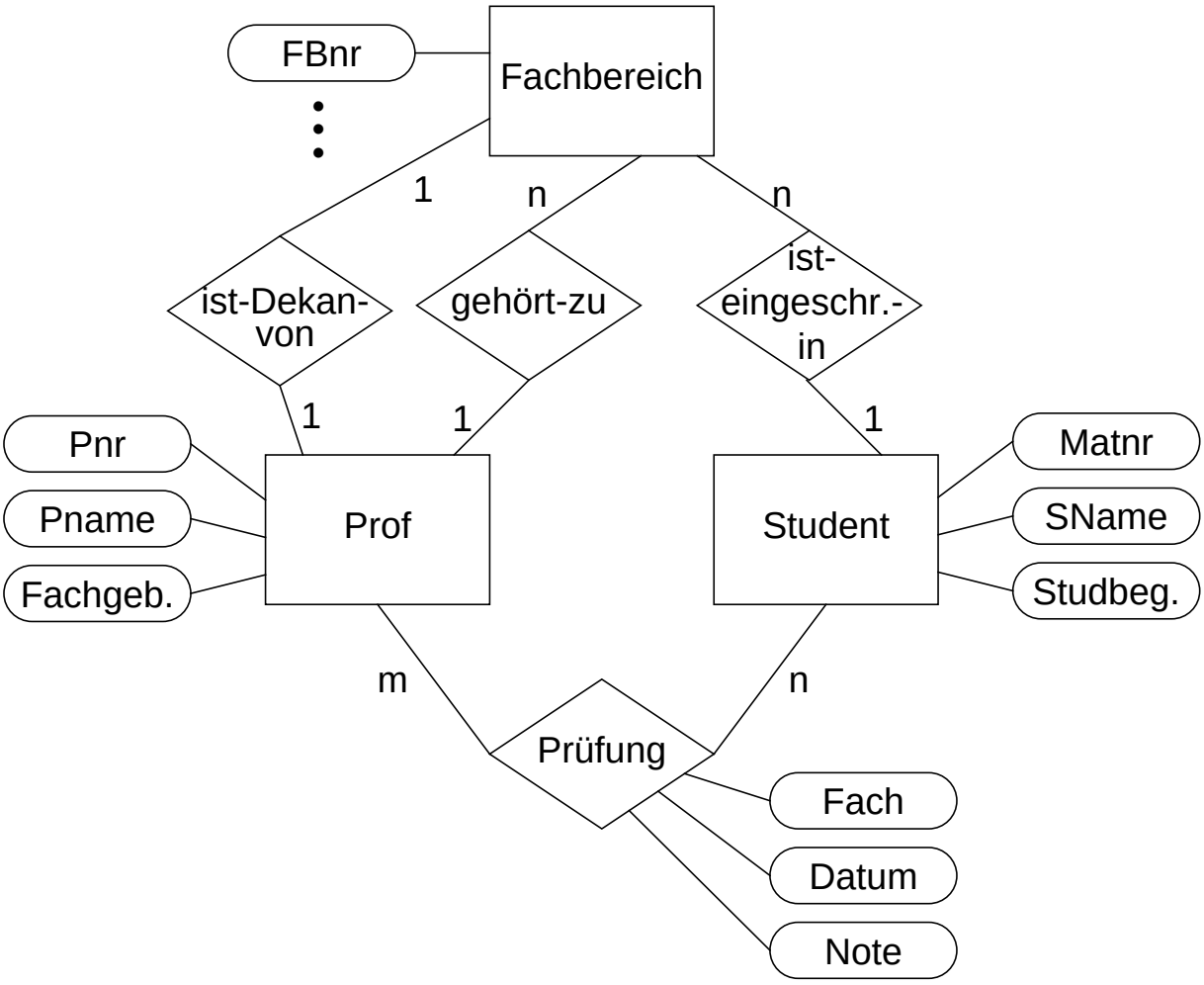
4. partiell, überlappend (PÜ)





Überblick über die wichtigsten Datenmodelle

- **Entity/Relationship-Diagramm der Beispiel-Miniwelt**



- Spezifikation benutzerdefinierter Beziehungen
- Klassifikation der Beziehungstypen (1:1, 1:n, n:m)
- Verfeinerung durch Kardinalitätsrestriktionen ([1,1]:[1,10], [0:1]:[0:*])
- relativ semantikarme Darstellung vom Weltausschnitt

Relationenmodell

Schema

FB

FBNR	FBNAME	DEKAN

PROF

PNR	PNAME	FBNR	FACHGEB

STUDENT

MATNR	SNAME	FBNR	STUDBEG

PRÜFUNG

PNR	MATNR	FACH	DATUM	NOTE

Ausprägungen

FB	FBNR	FBNAME	DEKAN
	FB 9	WIRTSCHAFTSWISS	4711
	FB 5	INFORMATIK	2223

PROF	PNR	PNAME	FBNR	FACHGEB
	1234	HÄRDER	FB 5	DATENBANKSYSTEME
	5678	WEDEKIND	FB 9	INFORMATIONSSYSTEME
	4711	MÜLLER	FB 9	OPERATIONS RESEARCH
	6780	NEHMER	FB 5	BETRIEBSSYSTEME

STUDENT	MATNR	SNAME	FBNR	STUDBEG
	123 766	COY	FB 9	1.10.95
	225 332	MÜLLER	FB 5	15. 4.87
	654 711	ABEL	FB 5	15.10.94
	226 302	SCHULZE	FB 9	1.10.95
	196 481	MAIER	FB 5	23.10.95
	130 680	SCHMID	FB 9	1. 4.97

PRÜFUNG	PNR	MATNR	FACH	PDATUM	NOTE
	5678	123 766	BWL	22.10.97	4
	4711	123 766	OR	16. 1.98	3
	1234	654 711	DV	17. 4.97	2
	1234	123 766	DV	17. 4.97	4
	6780	654 711	SP	19. 9.97	2
	1234	196 481	DV	15.10.97	1
	6780	196 481	BS	23.12.97	3

Anfragebeispiele

Q1: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 1990 begonnen haben.

```
SELECT   *
FROM     STUDENT
WHERE    FBNR = ’FB5’ AND STUDBEG < ’1.1.90’
```

Q2: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

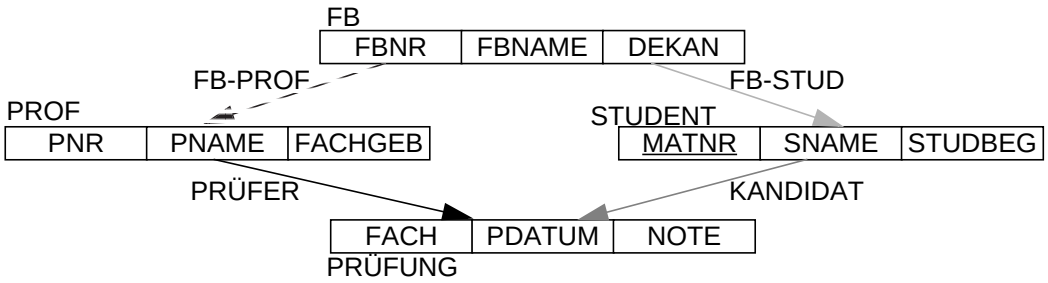
```
SELECT   *
FROM     STUDENT
WHERE    FBNR = ’FB5’ AND MATNR IN
         (SELECT   MATNR
         FROM      PRÜFUNG
         WHERE     FACH = ’DV’ AND NOTE ≤ ’2’)
```

Q3: Finde die Durchschnittsnoten der DV-Prüfungen für alle Fachbereiche mit mehr als 1000 Studenten
(höchstens eine DV-Prüfung pro Student).

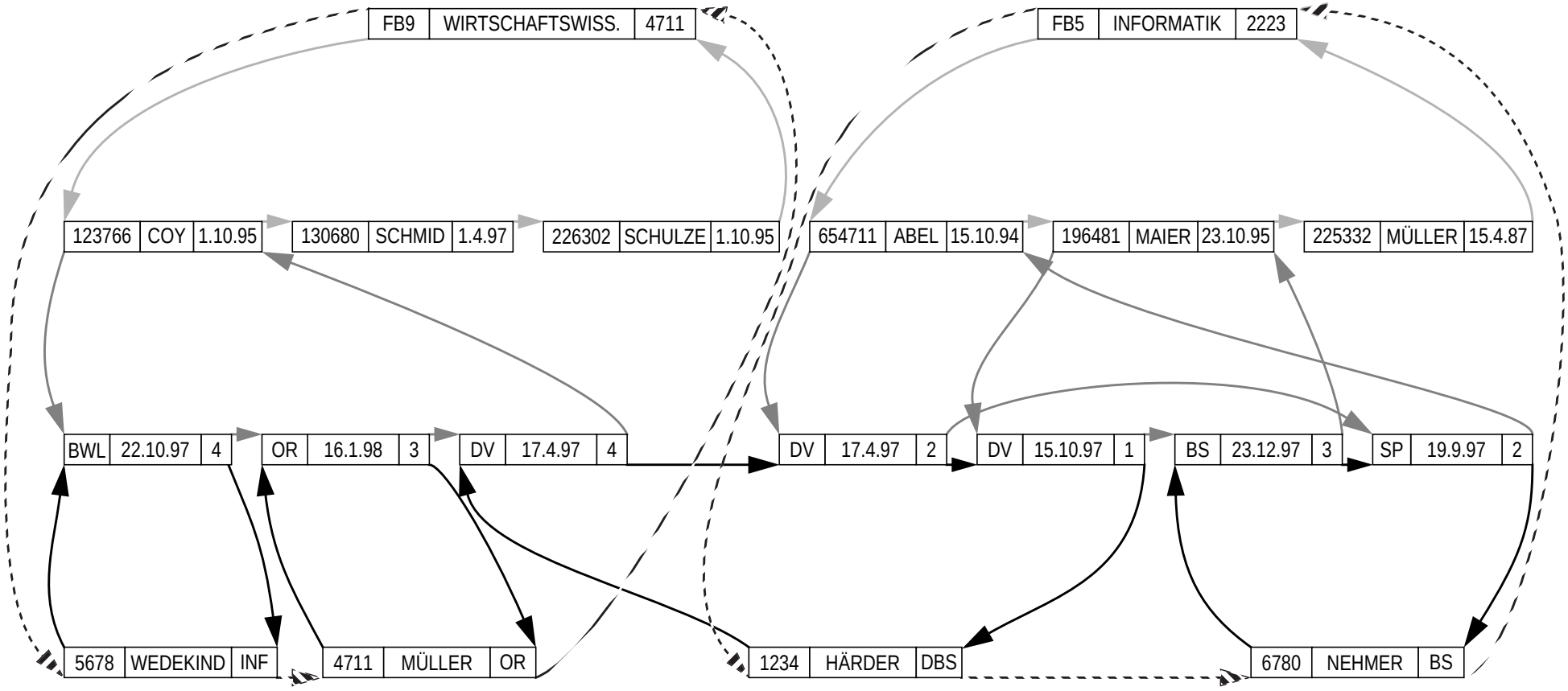
```
SELECT   F.FBNR, AVG (P.NOTE)
FROM     PRÜFUNG P, STUDENT F
WHERE    P.FACH = ’DV’ AND P.MATNR = F.MATNR
GROUP BY F.FBNR
HAVING   (COUNT(*) > 1000)
```

Netzwerkmodell

Schema



Ausprägungen



Anwendung der DML des Netzwerkmodells

- **Anfragebeispiele**

Q1: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 1990 begonnen haben.

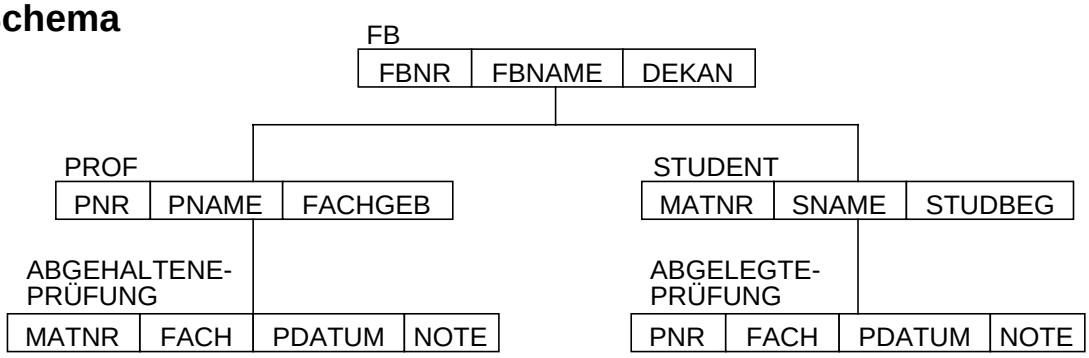
```
                MOVE 'FB5' TO FBNR OF FB;
                FIND FB USING FBNR;
NEXT_STUDENT:   FETCH NEXT STUDENT WITHIN FB-STUD SET;
                IF   END_OF_SET THEN EXIT;
                IF   STUDBEG < '1.1.90' THEN
                     PRINT STUDENT RECORD;
                GOTO NEXT_STUDENT;
```

Q2: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

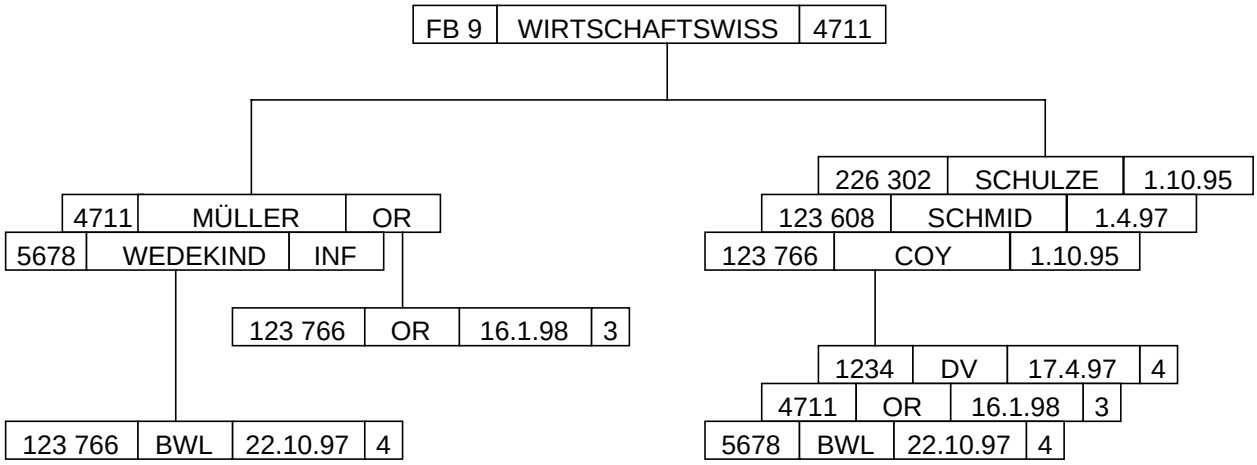
```
                MOVE 'FB5' TO FBNR OF FB;
                FIND FB USING FBNR;
NEXT_STUDENT:   FETCH NEXT STUDENT WITHIN FB-STUD SET;
                IF   END_OF_SET THEN EXIT;
NEXT_PRÜFUNG:   FETCH NEXT PRÜFUNG WITHIN KANDIDAT SET;
                IF   END_OF_SET THEN GOTO NEXT_STUDENT;
                IF   FACH = 'DV' AND NOTE ≤ '2'
                DO;
                     PRINT STUDENT RECORD;
                     GOTO NEXT_STUDENT;
                END;
                GOTO NEXT_PRÜFUNG;
```

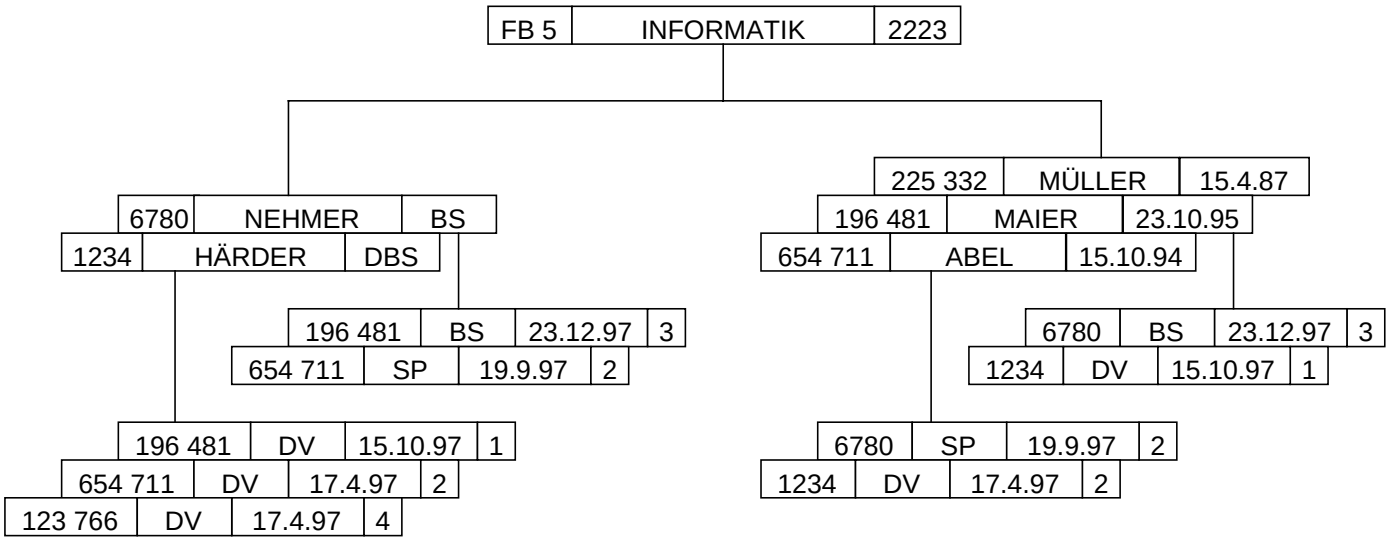
Hierarchisches Datenmodell

Schema



Ausprägungen





Anwendung der Sprache DL/1 auf eine Datenbank nach dem hierarchischen Datenmodell (IMS)

- **Anfragebeispiele:**

Q1: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 1990 begonnen haben.

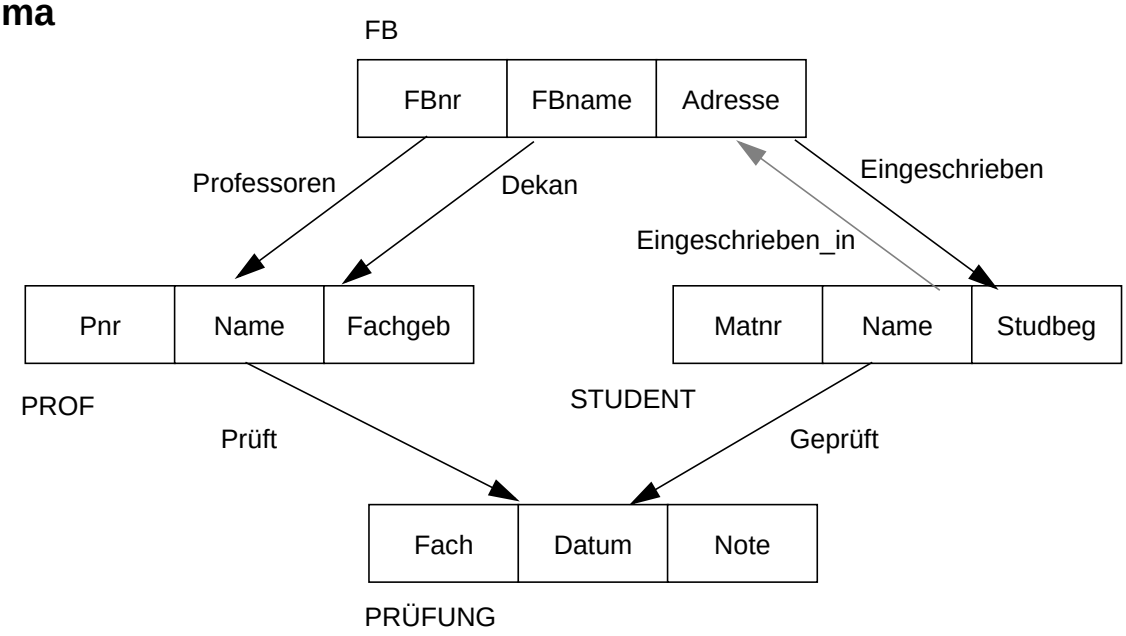
```
                    GU FB (FBNR = 'FB5');
NEXT_STUDENT:       GNP STUDENT (STUDBEG < '1.1.90');
                    IF END_OF_PARENT THEN EXIT;
                    PRINT STUDENT RECORD;
                    GOTO NEXT_STUDENT;
```

Q2: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
                    GU FB (FBNR = 'FB5');
NEXT_STUDENT:       GNP STUDENT;
                    IF END_OF_PARENT THEN EXIT;
                    GNP ABGELEGTE_PRÜFUNG   (FACH = 'DV') AND
                                            (NOTE ≤ '2');
                    IF END_OF_PARENT THEN GOTO NEXT_STUDENT;
                    PRINT STUDENT RECORD;
                    GOTO NEXT_STUDENT;
```

Objektorientiertes/objektrelationales Datenmodell

Schema



- **Typdefinitionen**

```
CREATE TYPE ADRESSEN_T
        (Strasse            char(30),
        Hausnummer          integer,
        PLZ                 integer,
        Ort                 char(30));

CREATE TYPE DATE_T
        (Jahr               integer,
        Monat               integer,
        Tag                 integer);

CREATE TYPE PERSON_T
        (Name               char(30)
         Adresse            ADRESSEN_T
        ...    (* weitere allg. Personendaten *)
        ... );

CREATE FUNCTION  . . . ;   (* Definition von zugehörigen Methoden *)
```

Objektorientiertes/objektrelationales Datenmodell (2)

- **Typdefinitionen**

CREATE TYPE PRÜFUNG_T
(Fach char(30),
Pdatum DATE_T,
Note integer)

CREATE FUNCTION Prüfungsbericht (Prüfung REF(PRÜFUNG_T))
RETURNS integer ... ;

CREATE TYPE FB_T
(FBnr char(30),
FBname char(30),
Adresse ADRESSEN_T,
Dekan REF (PROF_T),
Eingeschrieben SET (REF(STUDENT_T))
Professoren SET (REF(PROF_T)));

CREATE FUNCTION Durchschnittsnote.pro.Student (FB_T) ...;
CREATE FUNCTION Durchschnittsnote.pro.Fach (FB_T) ...;

CREATE TYPE PROF_T **UNDER** PERSON_T
(Pnr integer,
Fachgeb char(30),
Prüft SET (REF (PRÜFUNG_T)));

CREATE FUNCTION
Prüfungsergebnis (PROF_T, STUDENT_T, PRÜFUNG_T), ... ;

CREATE TYPE STUDENT_T **UNDER** PERSON_T
(Matnr integer,
Studbeg DATE_T,
Eingeschrieben_in REF (FB_T)
Geprüft SET (REF (PRÜFUNG_T)));

CREATE FUNCTION
Prüfungsanmeldung (PROF_T, STUDENT_T, PRÜFUNG_T) ...;
Prüfungsverlegung (PROF_T, STUDENT_T, PRÜFUNG_T) ...;

Objektorientiertes/objektrelationales Datenmodell (3)

- **Tabellendefinitionen**

```
CREATE TABLE FB OF FB_T
   (PRIMARY KEY FBnr
    Dekan WITH OPTIONS SCOPE Prof,
    Professoren WITH OPTIONS SCOPE Prof,
    Eingeschrieben WITH OPTIONS SCOPE Student);

CREATE TABLE Prof OF PROF_T
   (PRIMARY KEY Pnr
    Prüft WITH OPTIONS SCOPE Prüfung);

CREATE TABLE Student OF STUDENT_T
   (PRIMARY KEY Matnr
    Eingeschrieben_in WITH OPTIONS SCOPE FB,
    Geprüft WITH OPTIONS SCOPE Prüfung);

CREATE TABLE Prüfung OF PRÜFUNG_T
   (PRIMARY KEY Fach, Pdatum, Note);
```

Objektorientiertes/objektrelationales Datenmodell – Ausprägungenl

:B	FBnr	FBname	Dekan	Adresse	Professoren	Eingeschrieben
@1	FB 9	WIRTSCHAFTSWISS	@5	@201	@4, @5, ...	@7, @10, ...
@2	FB 5	INFORMATIK	@111	@202	@3, @6, ...	@8, @9, ...

PROF	Pnr	Name	Fachgeb	Prüft
@3	1234	HÄRDER	DATENBANKSYSTEME	@15, @16, ...
@4	5678	WEDEKIND	INFORMATIONSSYSTEME	@13, ...
@5	4711	MÜLLER	OPERATIONS RESEARCH	@14
@6	6780	NEHMER	BETRIEBSSYSTEME	@17, @19

STUDENT	Matnr	Sname	Studbeg	Eingeschrieben_in	Geprüft
@7	123 766	COY	1.10.95	@1	@13, @16, ...
@8	225 332	MÜLLER	15.4.87	@2	-
@9	654 711	ABEL	15.10.94	@2	@15, @17
@10	226 302	SCHULZE	1.10.95	@1	-
@11	194 481	MAIER	23.10.95	@2	@18, @19
@12	130 680	SCHMID	1.4.97	@1	-

PRUEFUNG	Fach	Pdatum	Note
@13	BWL	22.10.97	4
@14	OR	16.1.98	3
@15	DV	17.4.97	2
@16	DV	17.4.97	4
@17	SP	19.9.97	2
@18	DV	15.10.97	1
@19	BS	23.12.97	3

Objektorientiertes Datenmodell –

Anfragebeispiele

Q1: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 1990 begonnen haben.

```
SELECT *
FROM     STUDENT S
WHERE S.Eingeschrieben_in->FBnr = 'FB5'
            AND S.Studbeg < '1.1.90';
```

Q2: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
SELECT *
FROM     STUDENT S
WHERE S.Eingeschrieben_in->FBnr = 'FB5' AND EXISTS
            (SELECT *
            FROM TABLE (S.Geprüft) P
            WHERE P.Fach = 'DV' AND P.Note <= 2);
```

Q3: Finde die Durchschnittsnoten der DV-Prüfungen für alle Fachbereiche mit mehr als 1000 Studenten.

```
SELECT F.FBnr, Durchschnittsnote.pro.Fach (F, ‘DV’)
FROM  FB  F,
WHERE 1000 < (SELECT COUNT(*)
                        FROM TABLE (F.Eingeschrieben));
```

Bewertung – Relationenmodell

- **Informationen des Benutzers**
  - ausschließlich durch den Inhalt der Daten
  - keine physischen Verbindungen
  - keine bedeutungsvolle Ordnung

- **deskriptive Sprachen**
  - hohes Auswahlvermögen
  - leichte Erlernbarkeit auch für den DV-Laien

- **Vorteile**
  - strenge theoretische Grundlage
  - einfache Informationsdarstellung durch Tabellen
  - keine Bindung an Zugriffspfade oder Speichertechnologie
  - keine Aussage über die Realisierung
  - hoher Grad an Datenunabhängigkeit
  - symmetrisches Datenmodell; d.h. es gibt keine bevorzugte Zugriffs- oder Auswertungsrichtung

- **Hauptnachteil**
  - Einsatz von nicht-prozeduralen Sprachen soll Ineffizienz implizieren
  - aber: Optimierung der Anforderungen liegt in der Verantwortung des Systems

Bewertung – Hierarchische Datenmodelle und Netzwerkmodelle

- **Informationen des Benutzers**
  - Darstellung durch „Verbindungen“ von Datensätzen
  - Berücksichtigung ihrer Anordnung in Speicherungsstrukturen

- **prozedurale Sprachen**
  - Programmierer als Navigator
  - satzweiser Zugriff über vorhandene Zugriffspfade

- **Vorteile**
  - Entwurf effizienter Programme durch den Anwendungsprogrammierer
  - höheres Leistungsvermögen bei zugeschnittenen Zugriffspfadstrukturen

- **Nachteile**
  - starke Bindung an Zugriffspfade
  - geringerer Grad an Datenunabhängigkeit
  - beschränkt auf die Benutzerklassen „Anwendungsprogrammierer und “Systempersonal“
  - bei hierarchischen Datenmodellen
    - unnatürliche Organisation bei komplexen Beziehungen (n:m)
    - Auswertungsrichtung ist vorgegeben

Bewertung – Objektorientierte Modelle

- **Informationen des Benutzers**
  - Darstellung durch „Verbindungen“ von Datensätzen
  - sowie durch den Inhalt der Daten

- **prozedurale und deskriptive Sprachen**
  - Programmierer als Navigator
  - hohes Auswahlvermögen
  - satzweiser Zugriff (über vorhandene Zugriffspfade)

- **Vorteile**
  - Entwurf effizienter Programme durch den Anwendungsprogrammierer
  - Leistungsvermögen aufgrund der Nutzung zugeschnittener Zugriffspfadstrukturen

- **Nachteile**
  - kein einheitliches Datenmodell
  - keine theoretische Grundlage
  - Informationsdarstellung durch ‘Tabellen’ und ‘Verbindungen’
  - Bindung an Zugriffspfade
    - Symmetrie des Datenmodells?
  - Grad an Datenunabhängigkeit?

Zusammenfassung

- **ERM-Charakteristika**
  - 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

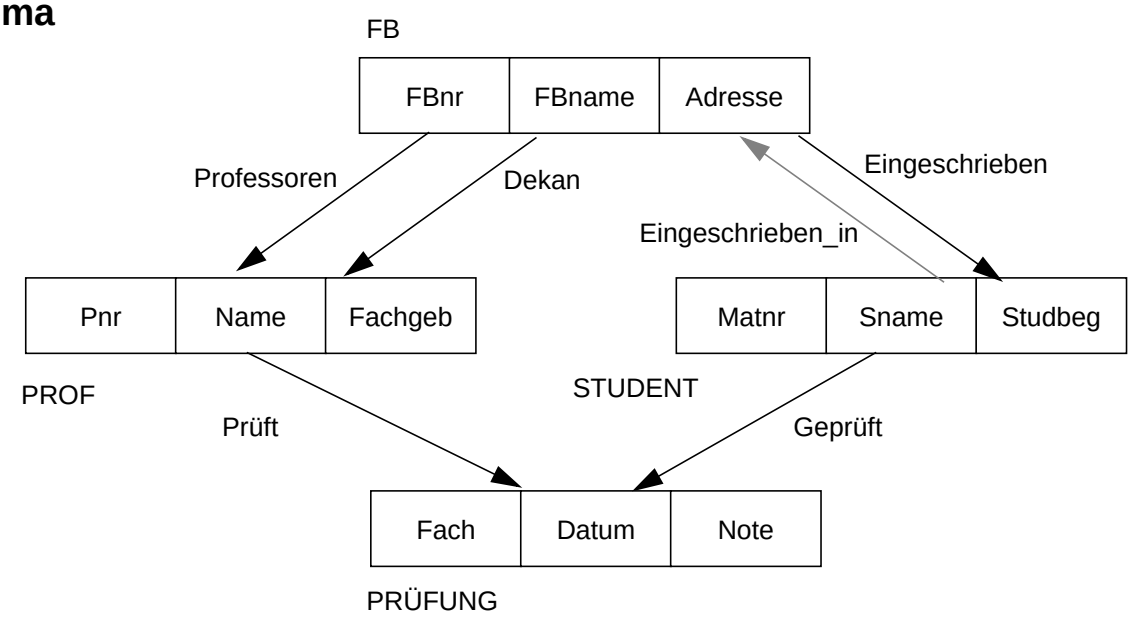
- **Datenmodelle**
  - Relationenmodell mit hohem Grad an Datenunabhängigkeit
  - Netzwerkmodell mit navigierendem, satzweisem Zugriff
  - hierarchisches Datenmodell mit unnatürlicher Organisationsform bei komplexen Beziehungen
  - objektorientiertes Datenmodell natürliche Organisationsform bei vorwiegend navigierendem, satzweisen Zugriff

Ausblick: Objektorientierter DB-Entwurf

- **Verfahren der objektorientierten Modellierung**
  - erweiterte ER-Modelle können nur Einstiegspunkt sein
  - Was soll modelliert werden?
    - Objektorientierte Analyse (OOA) in 5 Schritten:
      1. Finden von Klassen und Objekten
      2. Erarbeiten von Strukturen
      3. Aufteilen in Themen und Subthemen
      4. Definieren der Attribute
      5. Definieren der Methoden
    - Beschreibung mittels 5-Ebenen-Diagramm
      1. Subjektebene
      2. Klassen- und Objektebene
      3. Strukturebene
      4. Attributebene
      5. Service-Ebene
  - Wie soll es modelliert werden?
    - Objektorientiertes Design (OOD)
      Spezifikation einer Implementierung mittels eines OO-DM
  - ➥ daten- oder funktionsorientierte Verfahren, Mischung aus beiden u. a.

Objektorientiertes Datenmodell

Schema



- **Typdefinitionen**

```
Type ADRESSEN_T =                          (* VALUE TYPE *)
                (Strasse: string,
                Hausnummer: integer,
                PLZ: integer,
                Ort: string)
End ADRESSEN_T

Type DATE_T =                              (* VALUE TYPE *)
                (Jahr: integer,
                Monat: integer,
                Tag: integer)
End DATE_T

Type PERSON_T =                            (* OBJECT TYPE *)
  IsA OBJECT_T
  Structure
                (Person-Daten              : . . .)
  Methods     . . .
End PERSON_T
```

Objektorientiertes Datenmodel (2)

- **Typdefinitionen**

```
Type FB_T =
  IsA OBJECT_T
  Structure
            (FBnr                 : integer,
            FBname                : string,
            Adresse               : ADRESSEN_T,
            Dekan                 : REF ( PROF_T),
            Eingeschrieben        : SET_OF ( REF( STUDENT_T))
                                    INVERSE Eingeschrieben_in,
            Professoren           : SET_OF ( REF( PROF_T )))
  Methods   Durchschnittsnote.pro.Student(),
            Durchschnittsnote.pro.Fach(Fach), . . .
End FB_T

Type PROF_T =
  IsA PERSON_T
  Structure
            (Pnr                  : integer,
            Fachgeb               : string,
            Prüft                 : SET_OF (REF ( PRÜFUNG_T)))
  Methods   Prüfungsergebnis (Matnr, Pnr, Fach, Datum, Note), . . .
End PROF_T

Type STUDENT_T =
  IsA PERSON_T
  Structure
            (Matnr                : integer,
            Sname                 : string,
            Studbeg               : DATE_T,
            Eingeschrieben_in     : REF ( FB_T )
                                    INVERSE Eingeschrieben,
            Geprüft               : SET_OF (REF ( PRÜFUNG_T)))
  Methods   Prüfungsanmeldung (Matnr, Pnr, Fach, Datum),
            Prüfungsverlegung (...), . . .
End STUDENT_T
```

Objektorientiertes Datenmodel (3)

- **Typdefinition**

```
Type PRÜFUNG_T =
  IsA OBJECT_T
  Structure
            (Fach                 : string,
            Datum                 : DATE_T,
            Note                  : integer)
  Methods    . . .
End PRÜFUNG_T
```

- **Klassendefinitionen**

```
CREATE Class Fb : FB_T,
  Fb.Dekan REFERENCES Prof,
  Fb.Professoren REFERENCES Prof,
  Fb.Eingeschrieben REFERENCES Student;

CREATE Class Prof : PROF_T,
  Prof.Prüft REFERENCES Prüfung;

CREATE Class Student : STUDENT_T,
  Student.Eingeschrieben_in REFERENCES Fb,
  Student.Geprüft REFERENCES Prüfung;

CREATE Class Prüfung : PRÜFUNG_T;
```

Objektorientiertes Datenmodell –

Ausprägungen

FB	FBnr	FBname	Dekan	Adresse	Professoren	Eingeschrieben
@1	FB 9	WIRTSCHAFTSWISS	@5	@201	@4, @5, ...	@7, @10, ...
@2	FB 5	INFORMATIK	@111	@202	@3, @6, ...	@8, @9, ...

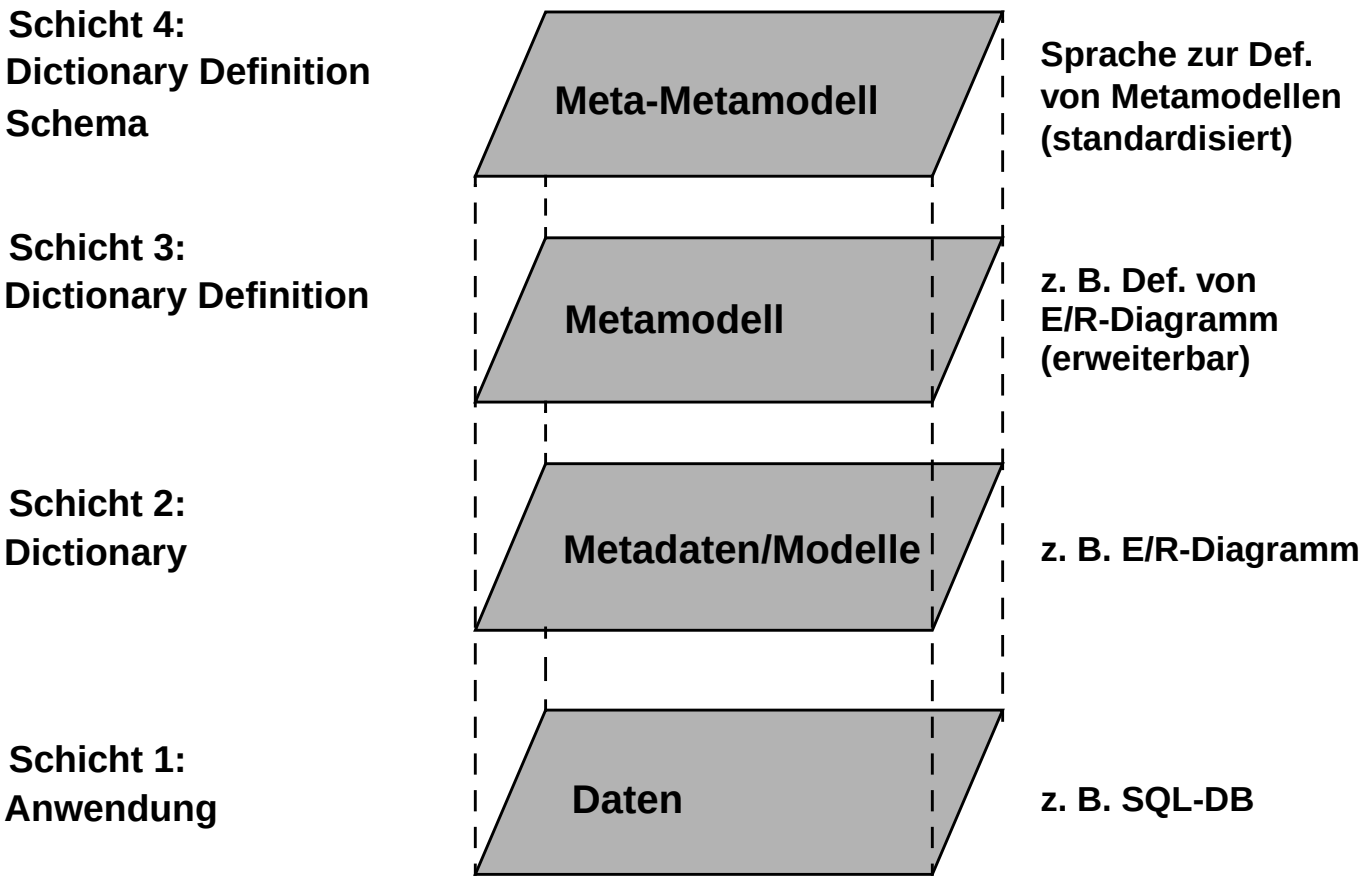
PROF	Pnr	Name	Fachgeb	Prüft
@3	1234	HÄRDER	DATENBANKSYSTEME	@15, @16, ...
@4	5678	WEDEKIND	INFORMATIONSSYSTEME	@13, ...
@5	4711	MÜLLER	OPERATIONS RESEARCH	@14
@6	6780	NEHMER	BETRIEBSSYSTEME	@17, @19

STUDENT	Matnr	Sname	Studbeg	Eingeschrieben_in	Geprüft
@7	123 766	COY	1.10.95	@1	@13, @16, ...
@8	225 332	MÜLLER	15.4.87	@2	-
@9	654 711	ABEL	15.10.94	@2	@15, @17
@10	226 302	SCHULZE	1.10.95	@1	-
@11	194 481	MAIER	23.10.95	@2	@18, @19
@12	130 680	SCHMID	1.4.97	@1	-

PRUEFUNG	Fach	Pdatum	Note
@13	BWL	22.10.97	4
@14	OR	16.1.98	3
@15	DV	17.4.97	2
@16	DV	17.4.97	4
@17	SP	19.9.97	2
@18	DV	15.10.97	1
@19	BS	23.12.97	3

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 ...	...

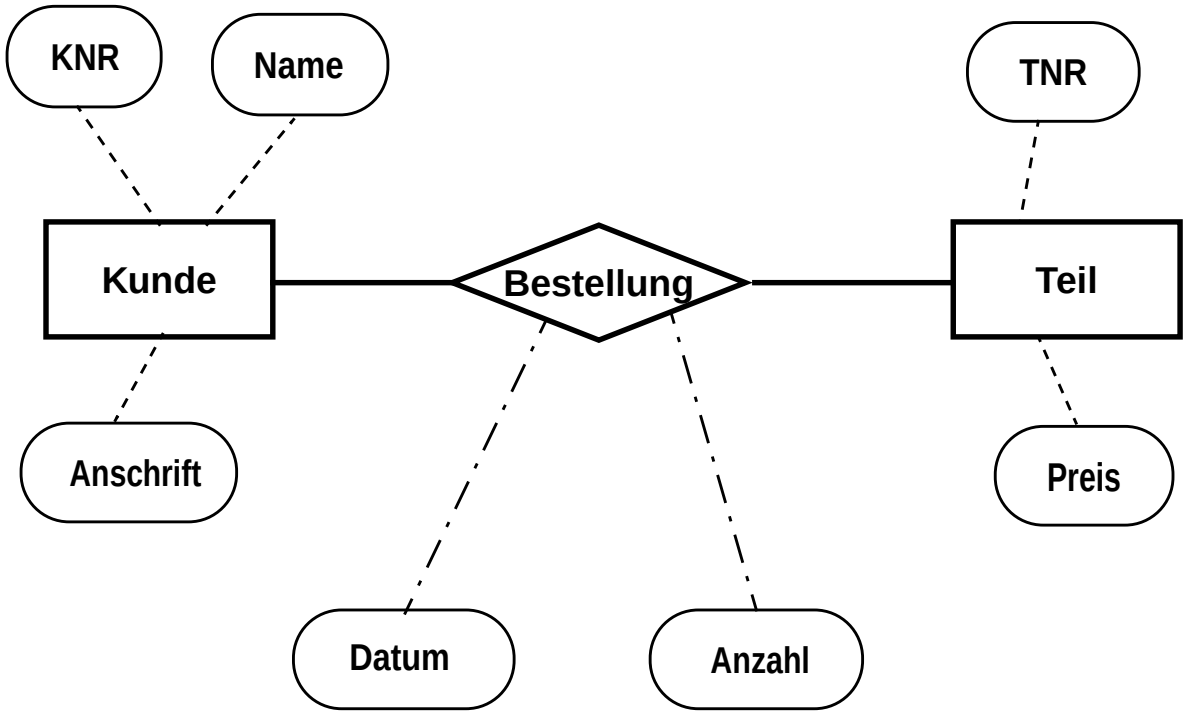
Tabelle: TEIL

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

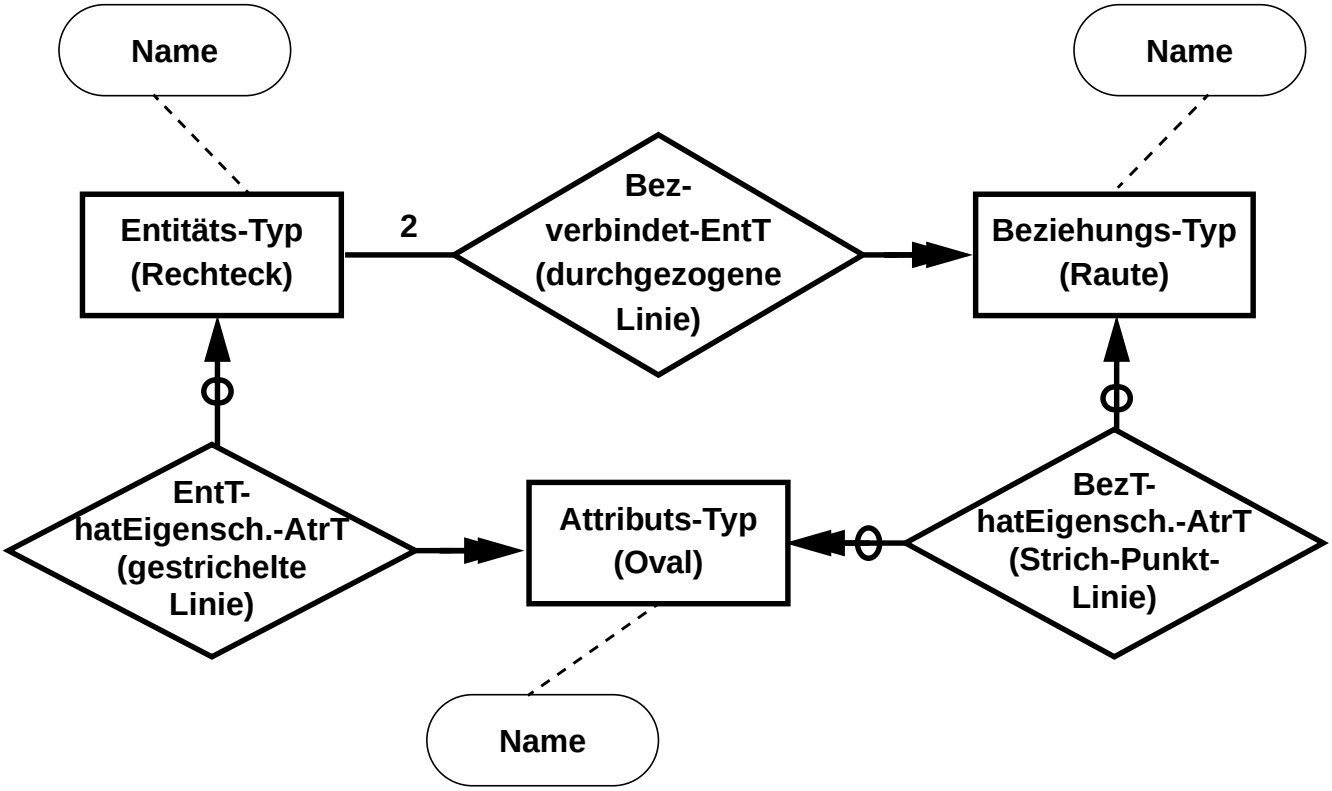
Tabelle: BESTELLUNG

KNR	TNR	ANZAHL	DATUM	...
1234	123 766	1000	1.2.00	...
1234	196 481	1500	2.7.99	...
5678	123 766	5000	4.9.98	...
5678	130 680	500	12.12.96	...
6780	130 680	3000	2.4.00	...
6780	196 481	3000	23.8.99	...

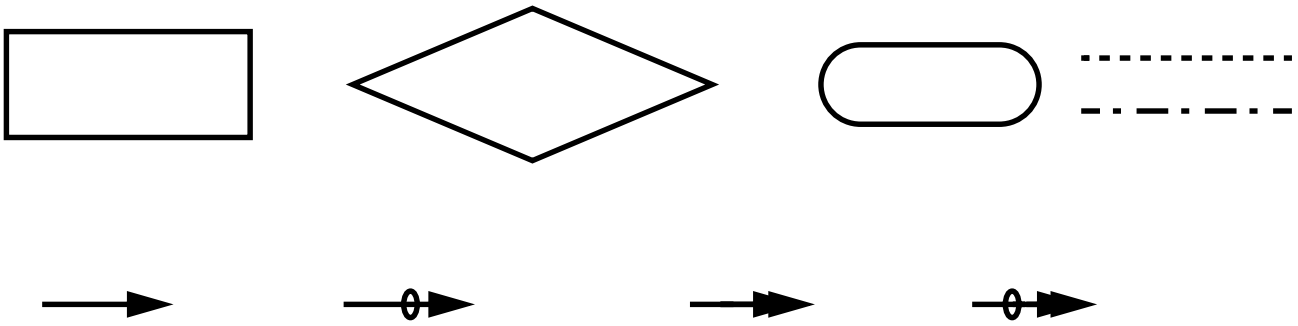
Schicht 2: ER-Diagramm



Schicht 3: Metamodell

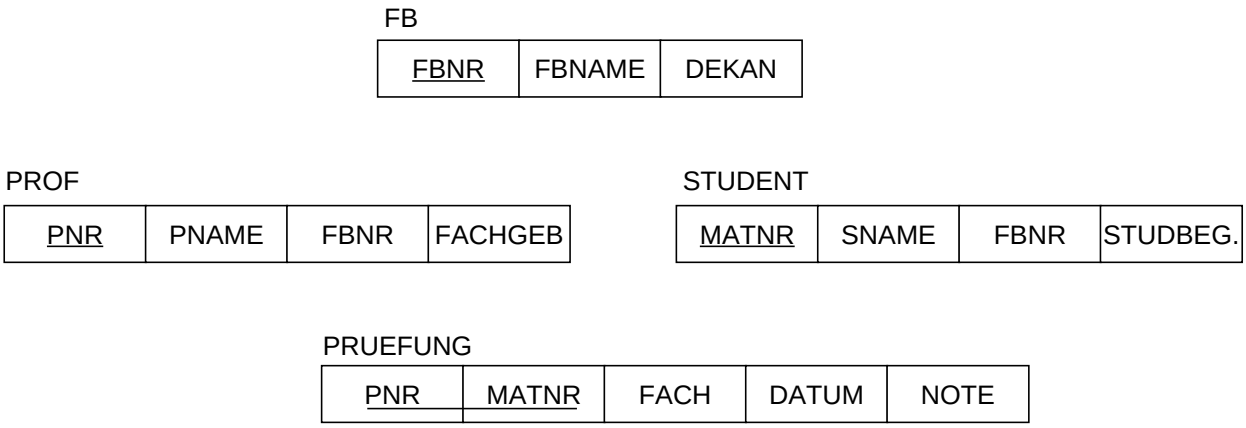


Schicht 4: Meta-Metamodell

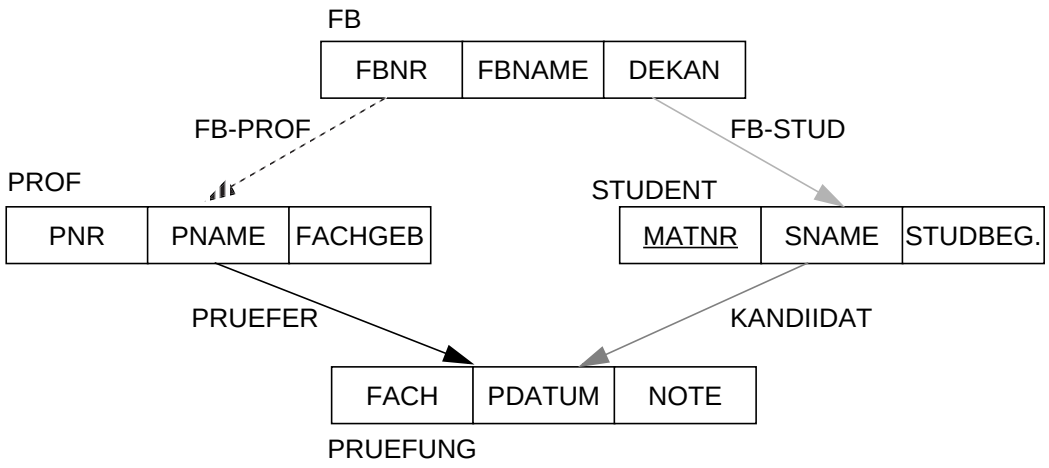


DB-Schemata

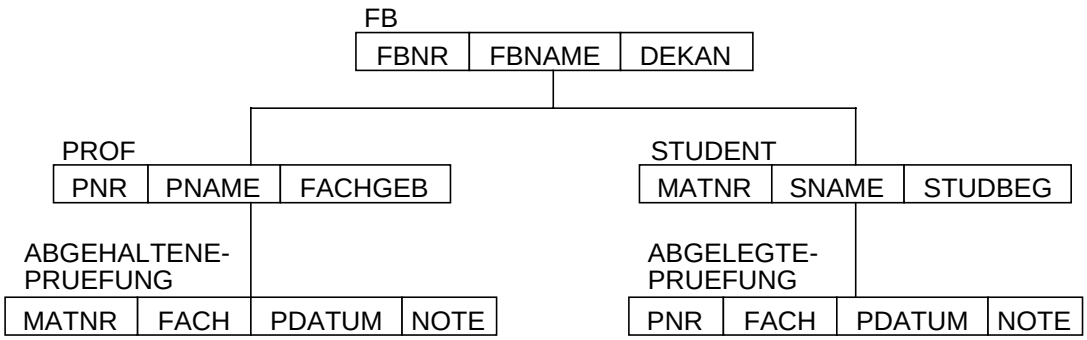
a) nach dem Relationenmodell



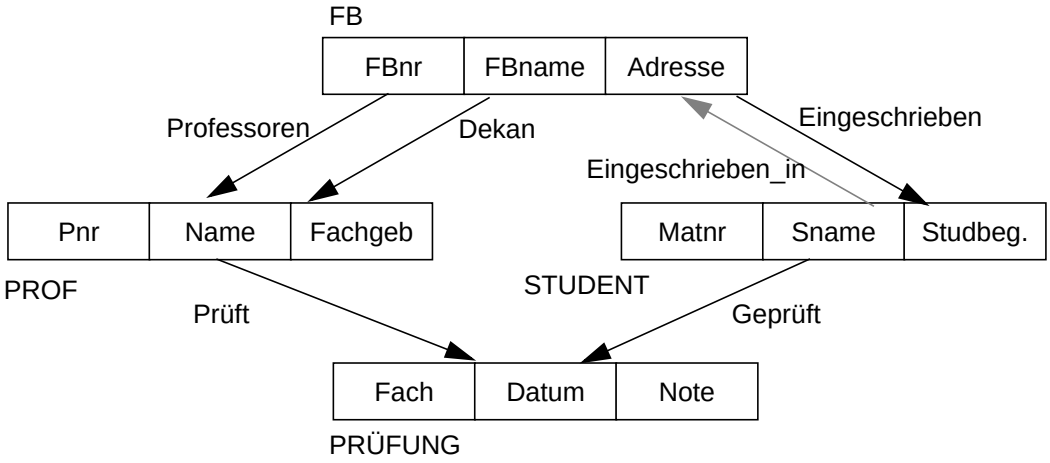
b) nach dem Netzwerkmodell

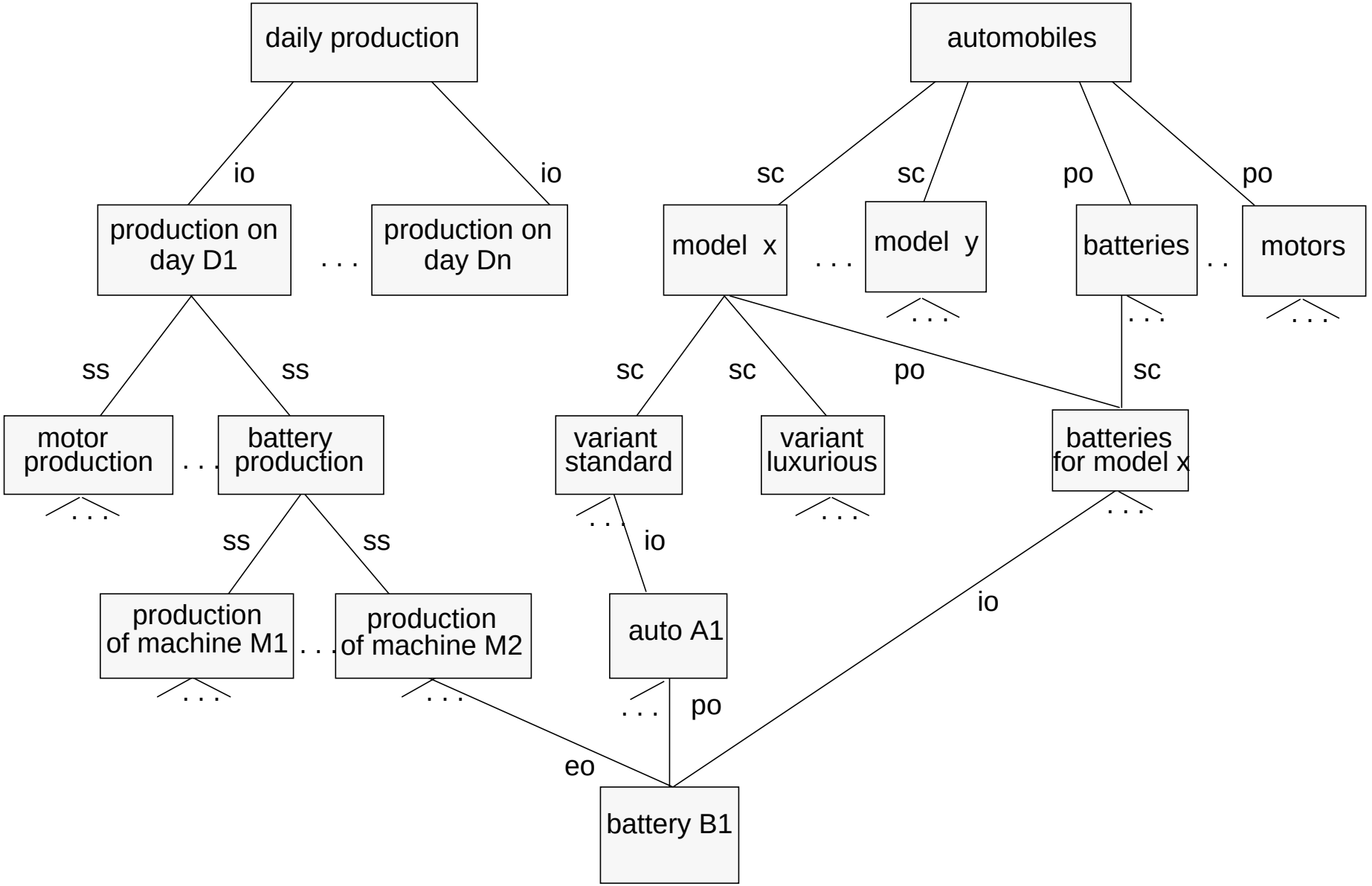


c) nach dem Hierarchiemodell



d) nach dem objektorientierten Modell




daily production
automobiles
io
io
sc
sc
po
po
production on day D1
production on day Dn
model x
model y
batteries
motors
ss
ss
sc
sc
po
sc
motor production
battery production
variant standard
variant luxurious
batteries for model x
ss
ss
io
io
production of machine M1
production of machine M2
auto A1
po
eo
battery B1

Information über Entities in einer

Entity-Menge

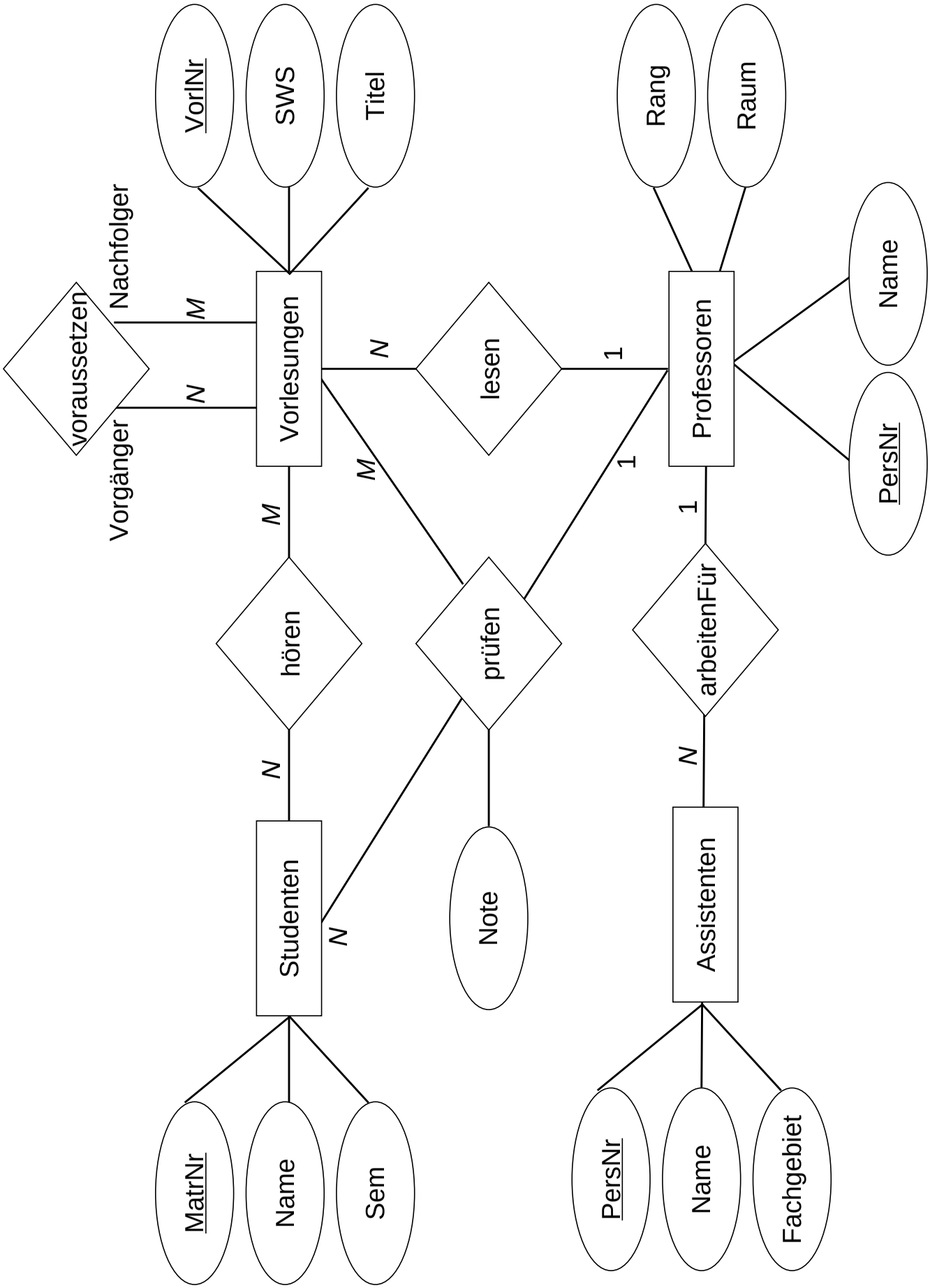
Attribute	F_1 (PERSONAL)	F_2 (NAME)		F_3 (KÜNSTLER-NAME)		F_4 (BERUF)
Entity-Menge E_1 (PERSONAL)	W_1 (PERSONAL-NUMMER)	W_2 (VOR-NAMEN)	W_3 (NACH-NAMEN)	W_2 (VOR-NAMEN)	W_3 (NACH-NAMEN)	W_4 (BERUF)
e_1	w_{11} (1234)	w_{21} (PETER)	w_{31} (MAIER)	w_{22} (HUGO)	w_{32} (BUG-FIGHTER)	w_{41} (POGRAM-MIERER)
e_2	w_{12} (5678)	w_{23} (OTTO)	w_{33} (ABEL)	w_{24} (ERNIE)	w_{34} (DEAD-LOCK)	w_{42} (SYSTEM-ANALYTIKER)
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Werte-Mengen

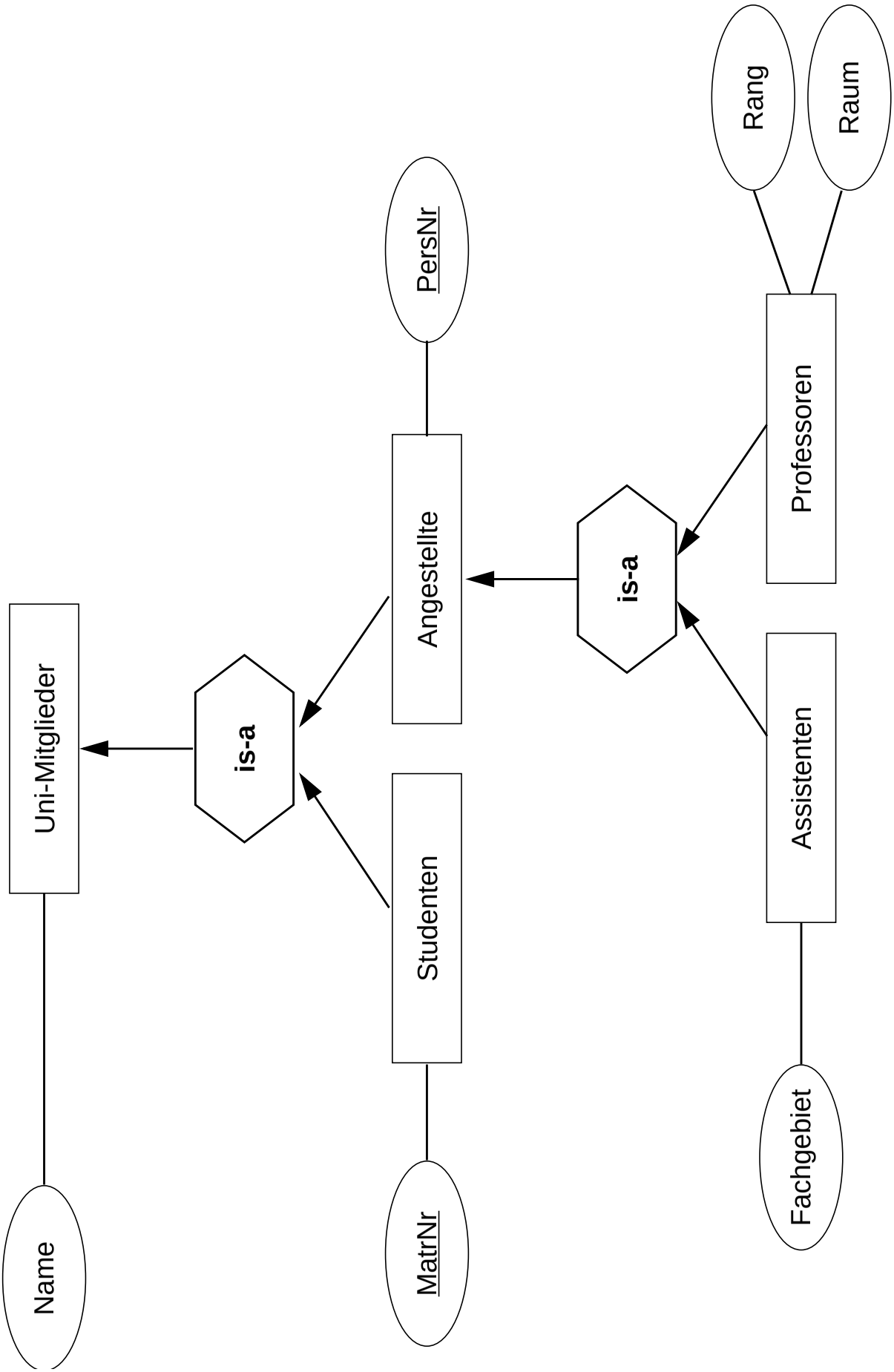
Information über Relationships in einer Relationship-Menge

Rolle	MIT-ARBEITER	PROJEKT			
			F_1 (ARBEITS-ZEITANTEIL)	F_2 (DAUER)	Attribute
Entity-Mengen	E_i (PERSONAL)	E_j (PROJEKT)	W_n (PROZENT)	W_m (ANZ.-MONATE)	
	e_{i1}	e_{j1}	w_{n1} (30)	w_{m1} (6)	
	e_{i2}	e_{j2}	w_{n2} (50)	w_{m2} (12)	
	⋮	⋮	⋮	⋮	

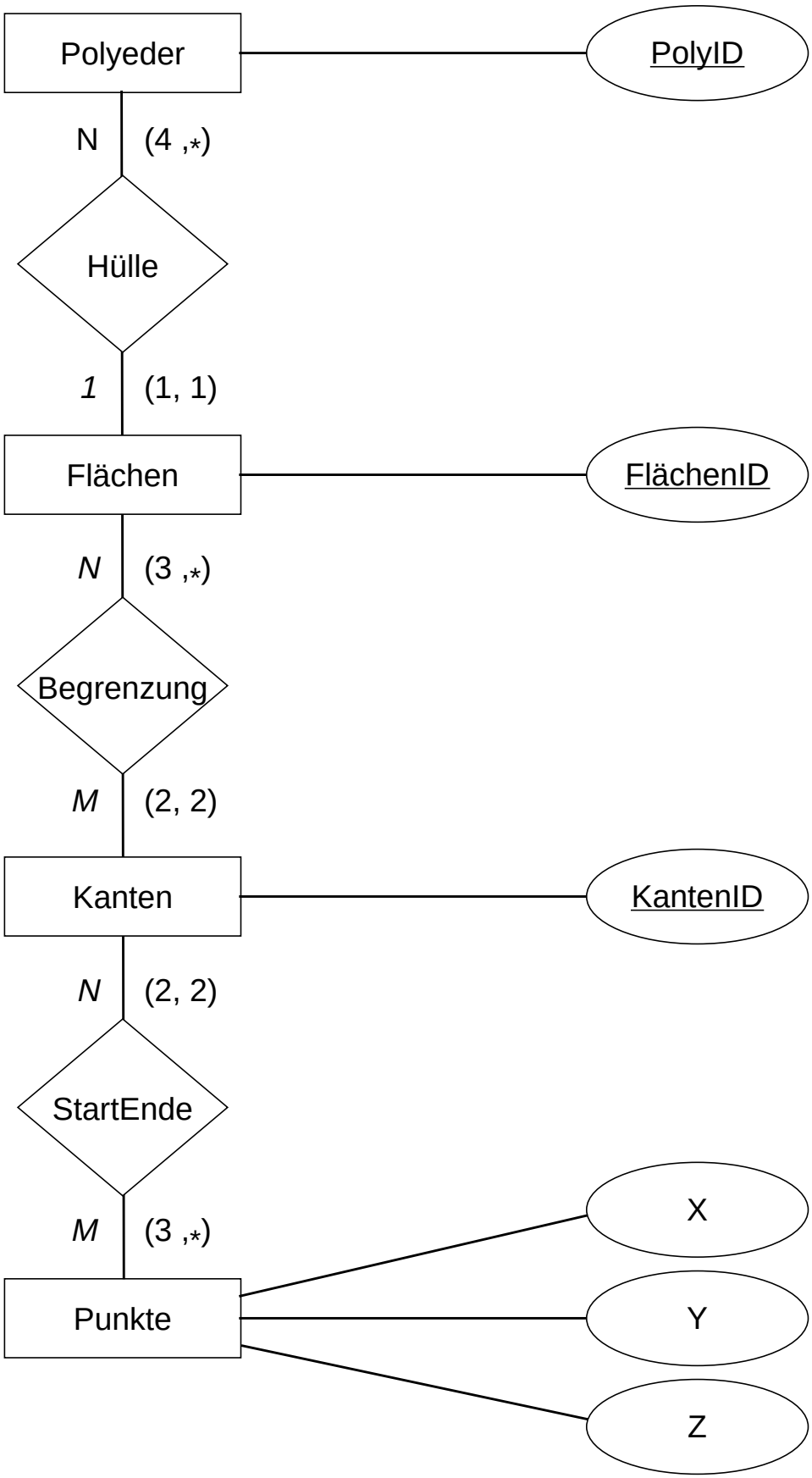
Markierung der Funktionalitäten



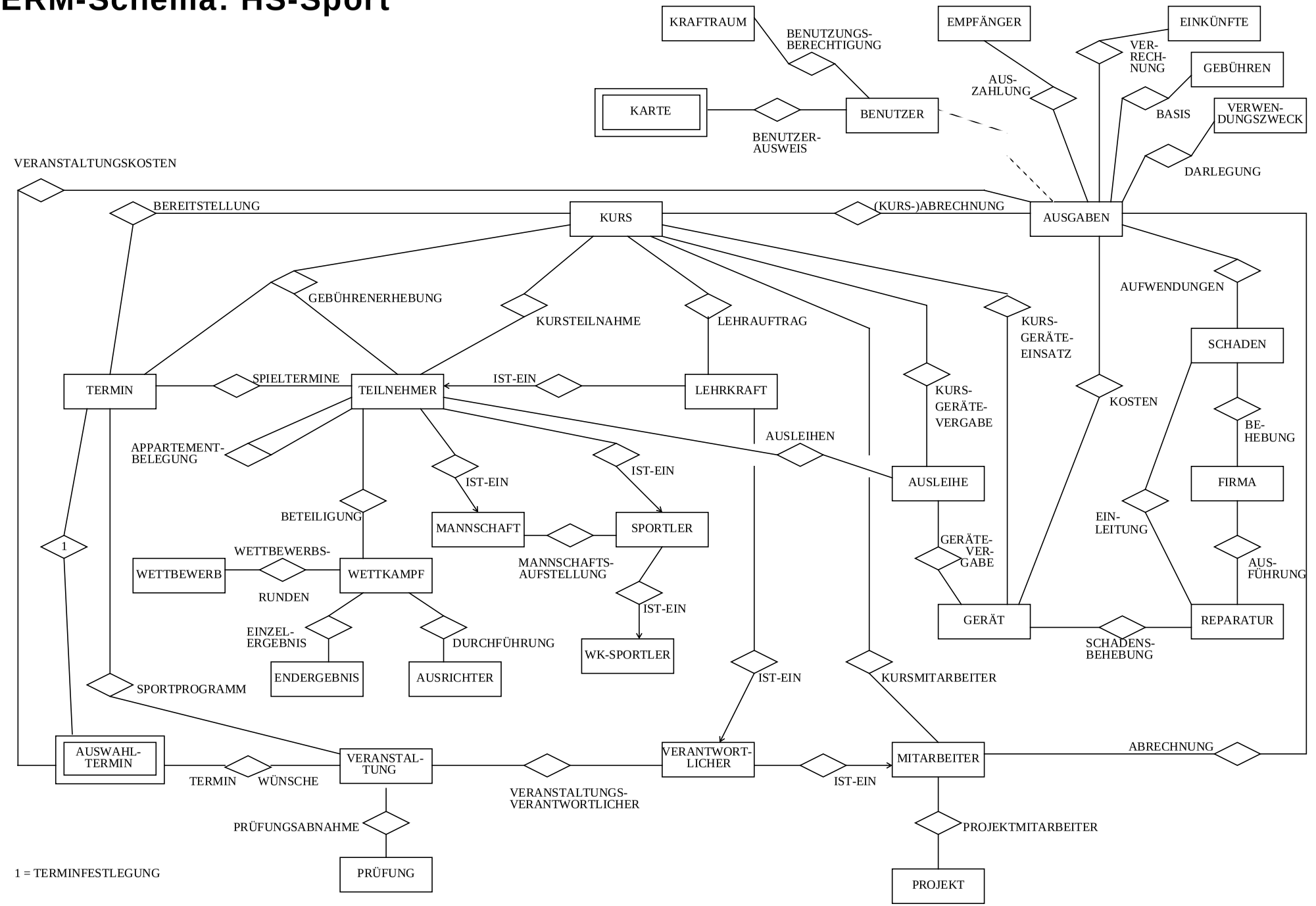
Generalisierung



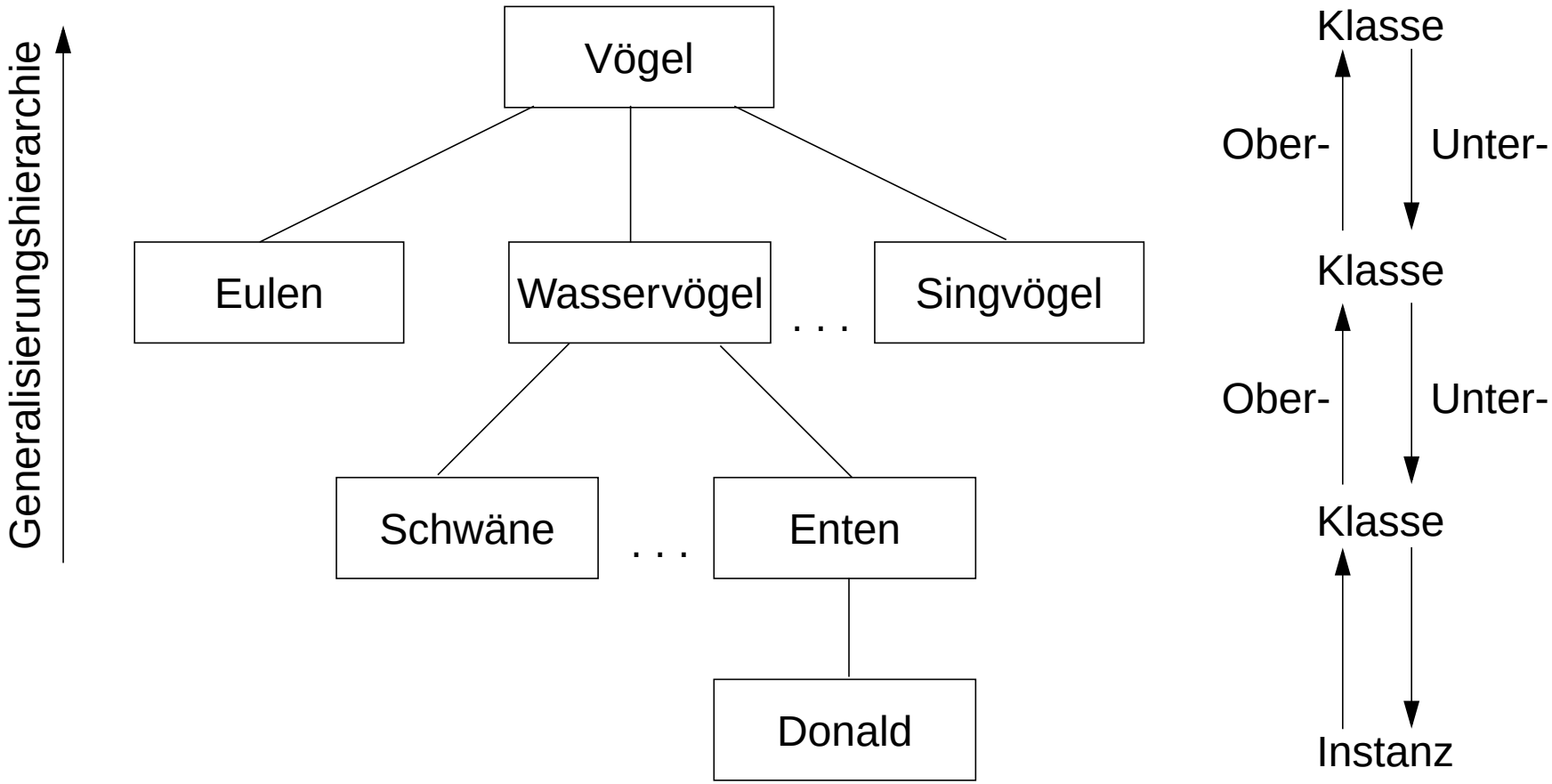
Begrenzungsflächendarstellung



ERM-Schema: HS-Sport



Modellierungsbeispiel zur Generalisierung


Generalisierungshierarchie
Vögel
Eulen
Wasservögel
. . .
Singvögel
Schwäne
. . .
Enten
Donald
Klasse
Ober-
Unter-
Klasse
Ober-
Unter-
Klasse
Instanz

Probleme bei der Informationserhebung

- **Ziel: Ungeschickte und ungenaue Begriffe in Anwendungsbereichen aufdecken und beseitigen.**

1. Synonyme kontrollieren!

Wörter, die dieselbe Bedeutung haben und gegeneinander ausgetauscht werden können.

z. B.: MITGLIED und GENOSSE bedeutet im Steuerberatungsunternehmen dasselbe.

2. Homonyme beseitigen!

Wörter, die gleich geschrieben und gesprochen werden, aber eine deutlich andere Bedeutung haben.

z. B.: STEUER als Lenkvorrichtung im Gegensatz zur STEUER als Abgabe an den Staat.

3. Äquipollenzen aufdecken!

Dieselben Objekte werden unter verschiedenen Blickwinkeln betrachtet.

z. B.: LAGERBESTAND als mengenmäßige und WARENKONTO als wertmäßige Rechnung über den Artikelbestand einer Firma.

4. Vagheiten klären!

Da inhaltlich keine klaren Abgrenzung (Definition) der Begriffe erfolgt, treten hinsichtlich der Objekte, die unter diesen Begriff fallen, Unklarheiten und Unsicherheiten auf.

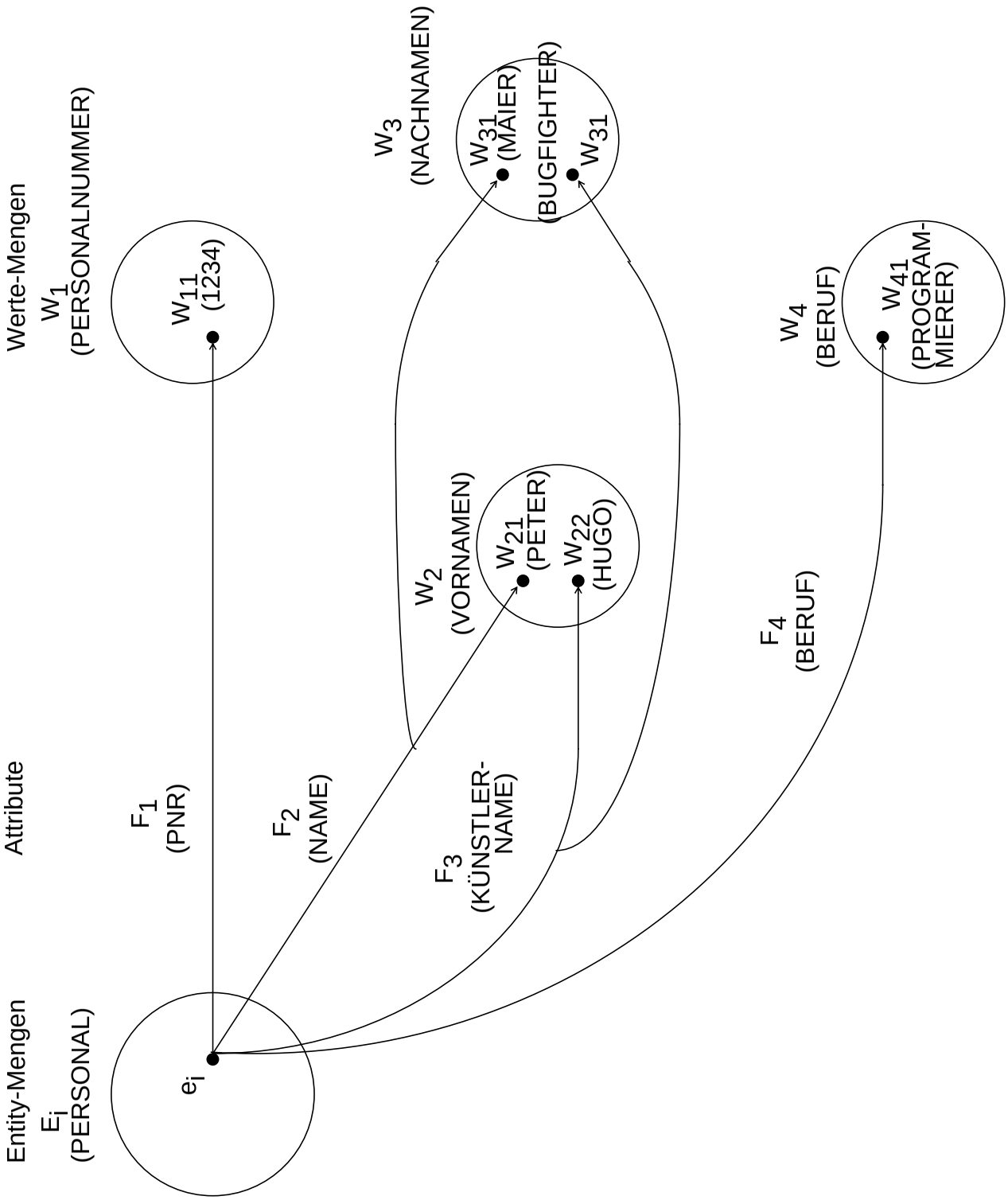
z. B.: Gehört WOHNSITZ als Ort der Berufsausübung eines Beraters noch zum Begriff UNTERNEHMEN.

5. Falsche Bezeichner ersetzen!

Abweichung der tatsächlichen von der zunächst suggerierten Wortbedeutung eines Begriffes.

z. B.: Bei DATEV hat die BERATERNUMMER nicht die Funktion SteuerBERATER zu identifizieren, sondern sie identifiziert eine von mehreren NUTZUNGSBERECHTIGUNGEN, die ein SteuerBERATER hat.

Attribute der Entity-Menge PERSONAL



Attribute der Relationship-Menge

PROJEKTMITARBEITER

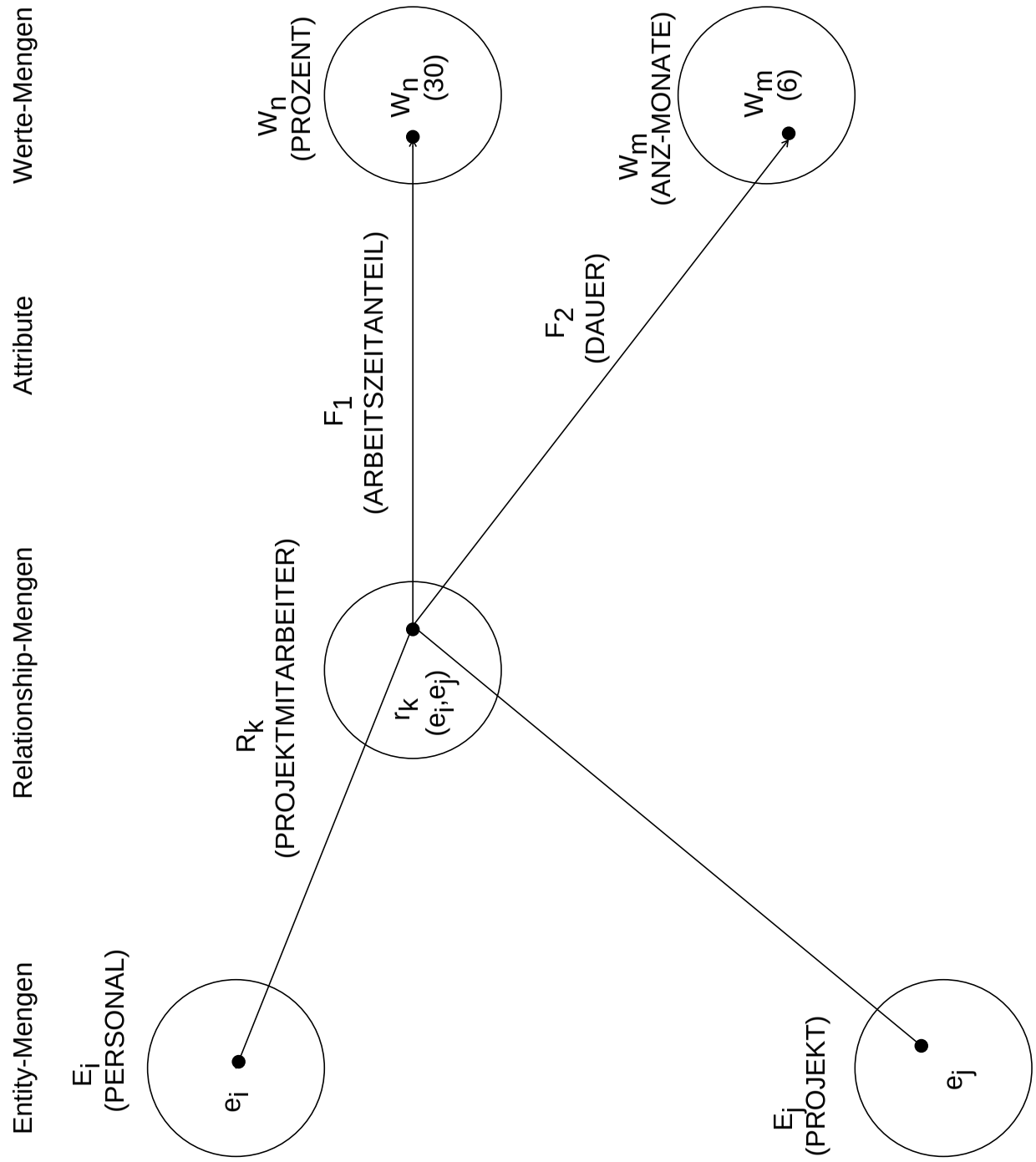
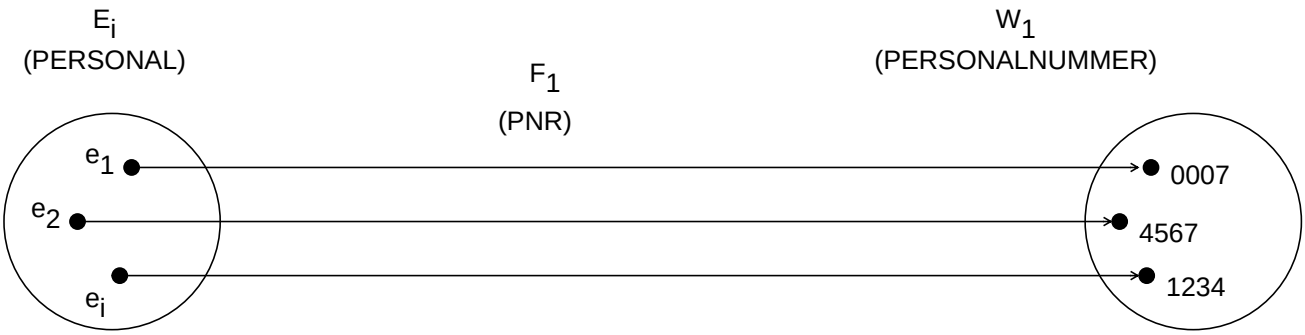


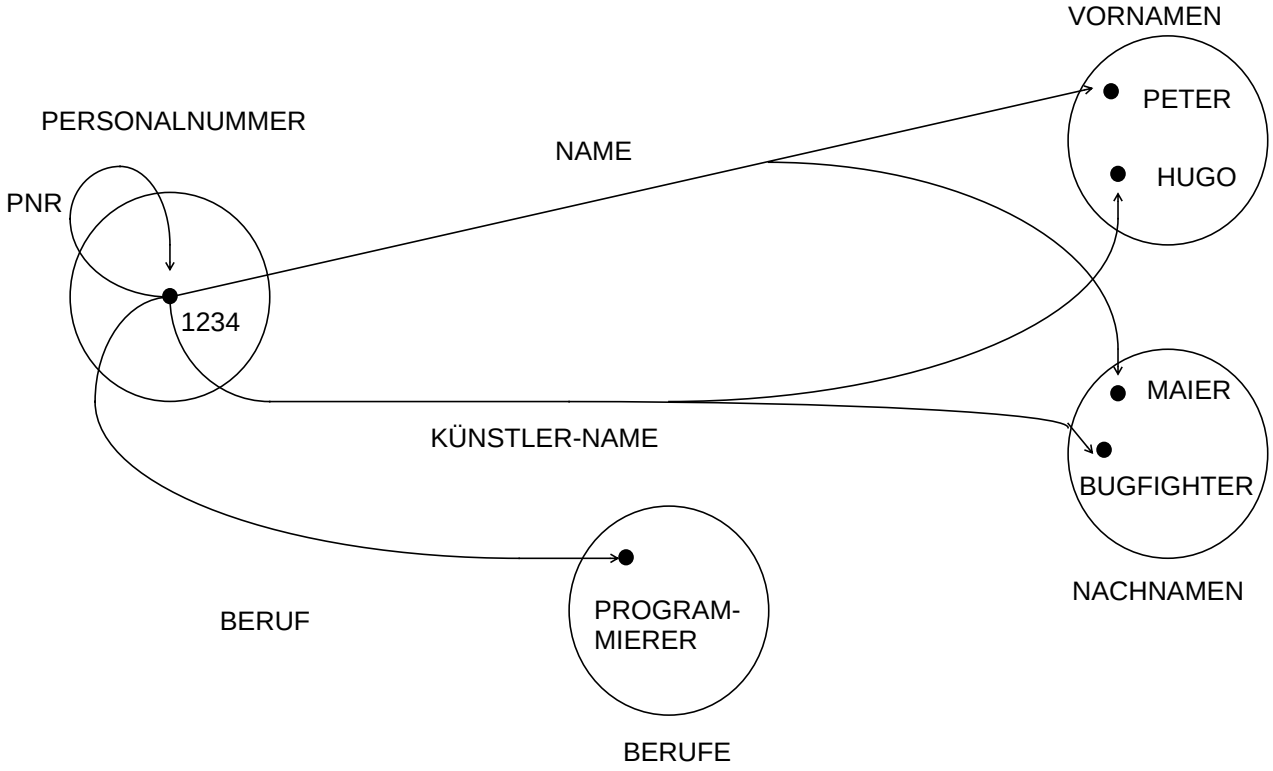
Abbildung der Entity-Menge E_i durch ihre Werte-Menge

für das Primärschlüssel-Attribut

1:1-Abbildung zwischen Entity-Menge E_i und Werte-Menge für Primärschlüssel-Attribut



Ersetzung der Entity-Menge E_i durch ihre Werte-Menge für den Primärschlüssel



Repräsentation der Miniwelt durch Werte in (regulären) Relationen

Reguläre Entity-Relation PERSONAL

Primär-schlüssel

Attribute	PNR	NAME		KÜNSTLERNAME		BERUF
Wertebereiche	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
Wertebereiche	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

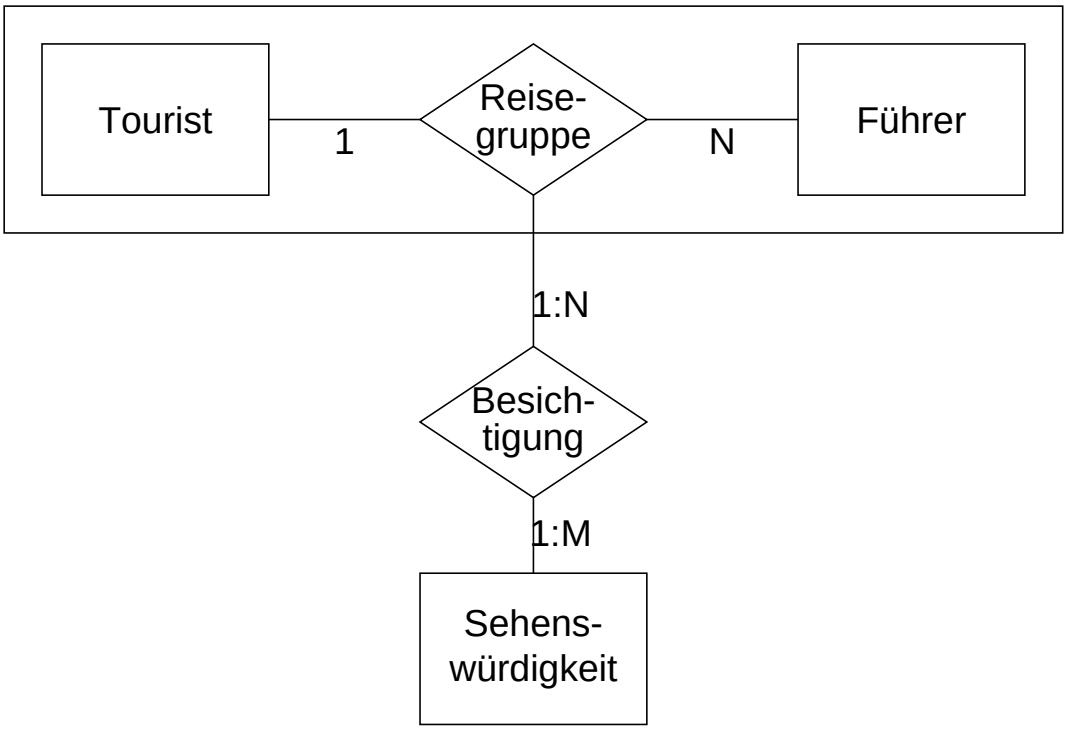
Beispiel:
Schwache Entity-Relation KINDER

Primärschlüssel

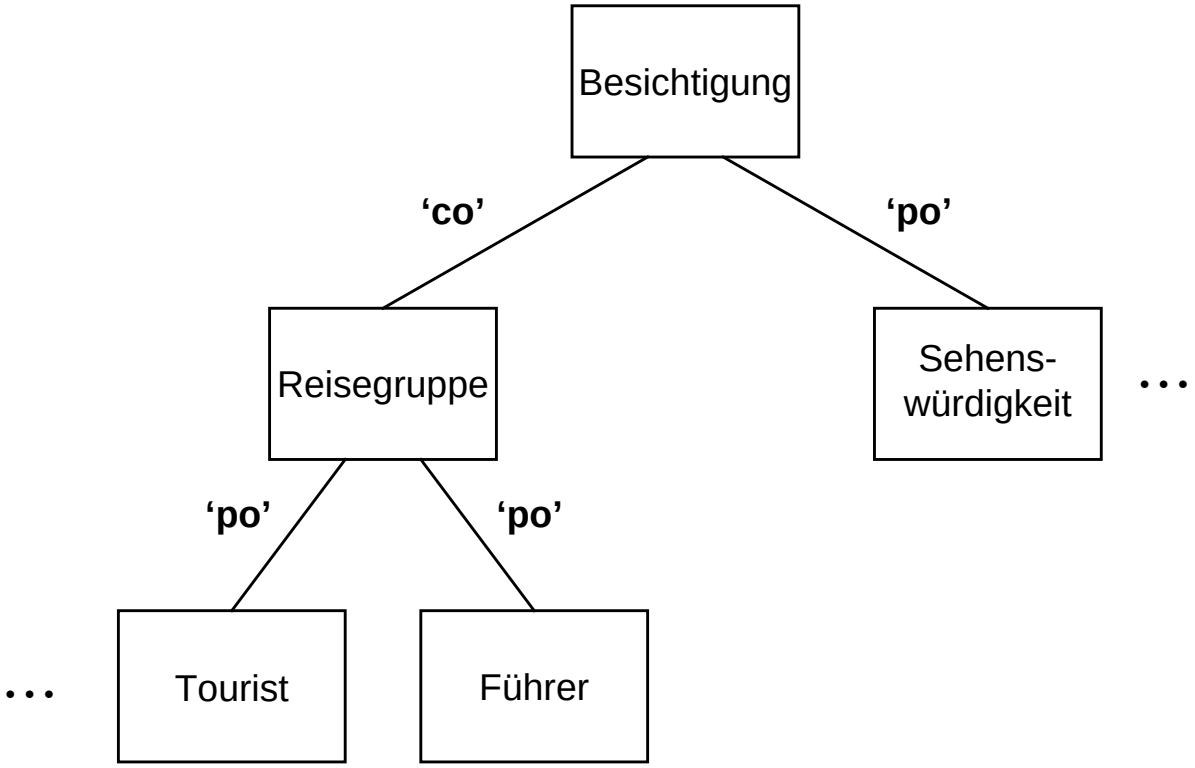
Entity-Relation-Name	PERSONAL		
Rolle	VATER-VON		
Entity-Attribute	PNR	NAME	ALTER
Wertebereiche	PERSONAL-NUMMER	VOR-NAMEN	ANZ.-JAHRE
	1234	JULIA	7
	5678	JULIA	7
	1234	PETER	2
	⋮	⋮	⋮

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

Aggregation: Beispiel (2)



Welche Lösung besitzt mehr Semantik?



Subklassen/Superklassen-Einschränkungen

- Eine Instanz in einer Subklasse erfordert eine zugehörige Instanz in jeder ihrer Superklassen. Eine Instanz in einer Superklasse erfordert eine oder null zugehörige Instanzen in jeder ihrer Subklassen.

- **Abstrakte Superklassen** besitzen keine direkten Instanzen. Alle ihre Instanzen sind auch in den Subklassen.

- **ONEOF**

  Eine Instanz in einer Superklasse ist nur mit einer Instanz aus einer ihrer Subklassen verknüpft (Is-A).

  CLASS Angestellter

    ABSTRACT SUPERCLASS OF

        (ONEOF(Ingenieur,Techniker, Sekretärin));

    . . .

  CLASS Ingenieur

    SUBCLASS OF (Angestellter); . . .

- **ANDOR**

  Die Instanzen der Subklassen können überlappen.

  CLASS Hochschulangehöriger

    SUPERCLASS OF (Student ANDOR Doktorand); . . .

- **AND**

  Falls die Instanzen nicht überlappen (ONEOF), können mit AND

  Beziehungen zwischen den disjunkten Gruppen hergestellt werden:

    CLASS Hochschulangehöriger

      SUPERCLASS OF (ONEOF (Männlich, Weiblich) AND

                                    (ONEOF (Beamter, Angestellter));

      . . . .

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

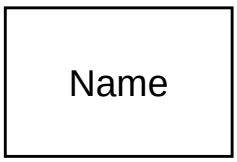
- **Zugehörigkeit** über Prädikat entscheidbar

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

- 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“

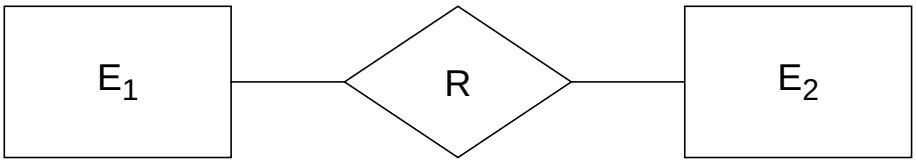
- **Eigenschaften**
  - Grad der Beziehung (*degree*)
  - Existenzabhängigkeit
  - Beziehungstyp (*connectivity*)
  - Kardinalität
- R ... Relationship-Menge = math. Relation zwischen n E_i

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

  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:

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 $\rightarrow$ PERSONALNUMMER

  NAME: PERSONAL $\rightarrow$ VORNAMEN x NACHNAMEN

  ANSCHRIFT: PERSONAL $\rightarrow$ 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

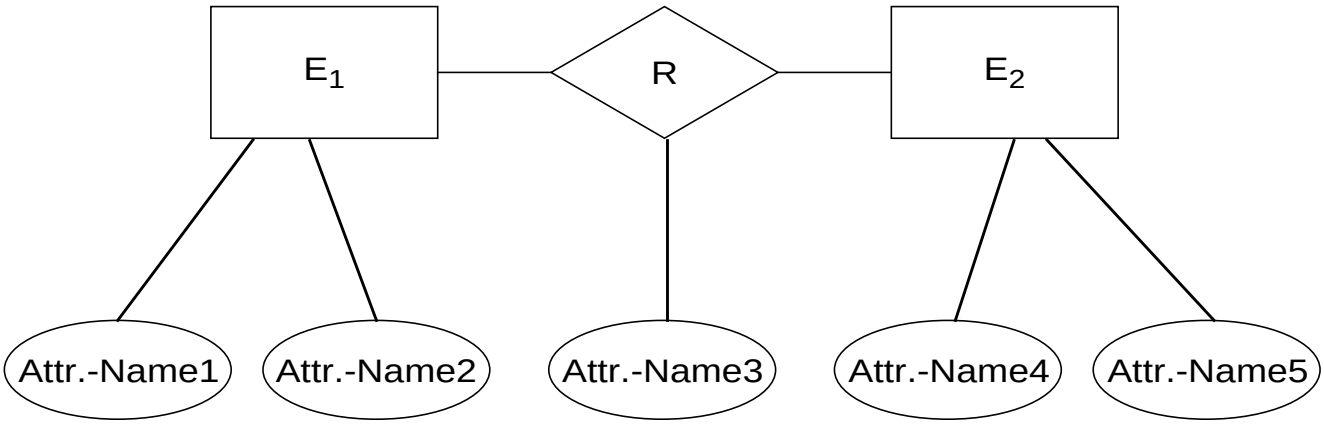
  $\Leftrightarrow$ **K** minimal; $e_i, e_j \in E$;

  $e_i \neq e_j \rightarrow$ **K**$(e_i) \neq$ **K**(e_j)

- mehrere Schlüsselkandidaten (SK) möglich

  → **Primärschlüssel** auswählen

- Darstellung von Attributen im ER-Diagramm

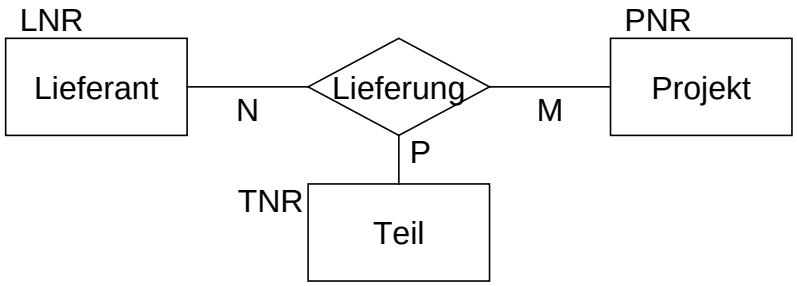


Modellierung dreistelliger Relationship-Mengen

- **Geg.**

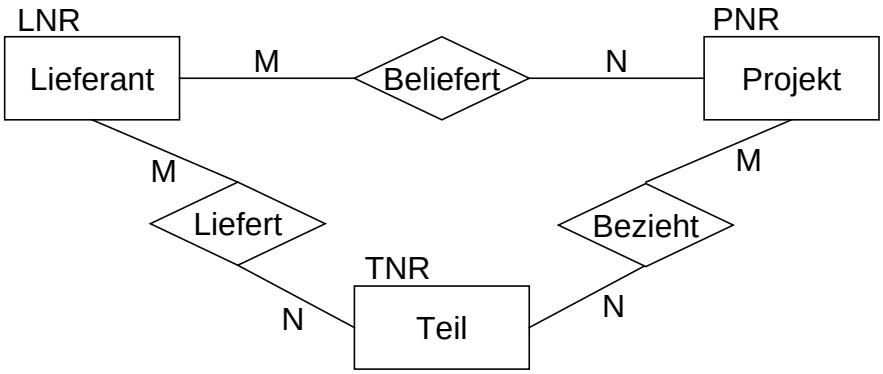
  Entity-Mengen
  Teil mit Attributen TNR, TBez, ...,
  Lieferant mit Attributen LNR, Firma, ...,
  Projekt mit Attributen PNR, PName, ...,

Modellierung der Relationship-Menge Lieferung mit Attribut LTermin



Angabe einiger zulässiger Instanzen (Tupel) mit t_1 , $t_2 \in$ TNR, l_1 , $l_2 \in$ LNR, p_1 , $p_2 \in$ PNR in den zugehörigen E- und R-Relationen

- **Das nachfolgende ER-Diagramm modelliert eine ganz andere Miniwelt!**



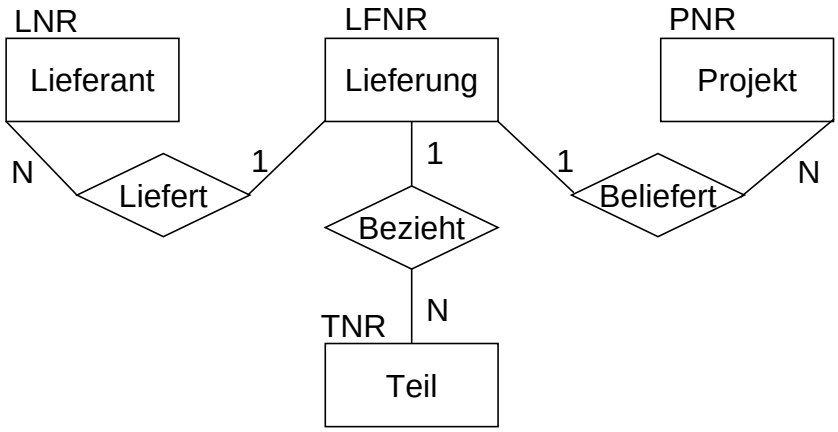
Angabe einiger zulässiger Instanzen (Tupel) mit t_1 , $t_2 \in$ TNR, l_1 , $l_2 \in$ LNR, p_1 , $p_2 \in$ PNR in den zugehörigen E- und R-Relationen

Was wird dargestellt?

- **Modellierung mit binären R-Mengen**

Lieferung sei E-Menge mit mit Attributen LFNR, LTermin, ...,

Alle Nicht-Schlüsselattribute dürfen Nullwerte annehmen

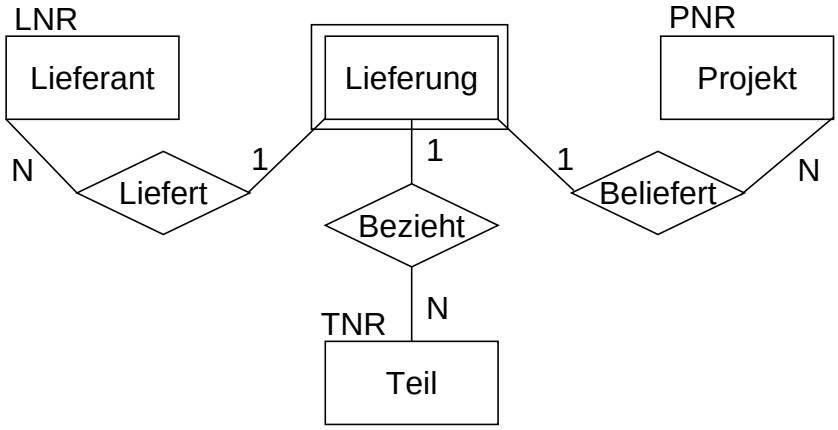


Angabe einiger zulässiger Instanzen (Tupel) mit t_1 , $t_2 \in$ TNR, l_1 , $l_2 \in$ LNR, p_1 , $p_2 \in$ PNR, f_1 , ..., $f_n \in$ LFNR in den zugehörigen E- und R-Relationen

Welche unerwünschten (falschen) Zustände sind möglich?

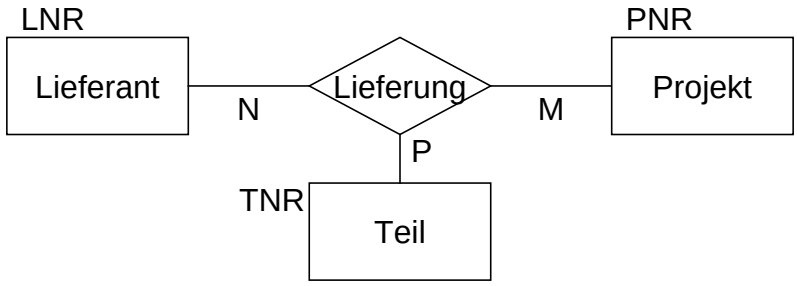
- **Existenzabhängigkeit ist zunächst nur für eine die „Existenz begründende“ E-Menge definiert.**

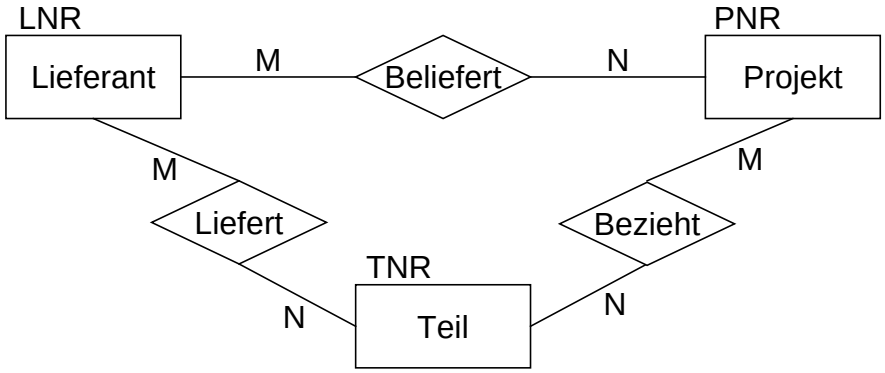
  Was passiert, wenn Lieferung als schwache (von mehreren abhängige) E-Menge definiert wird?

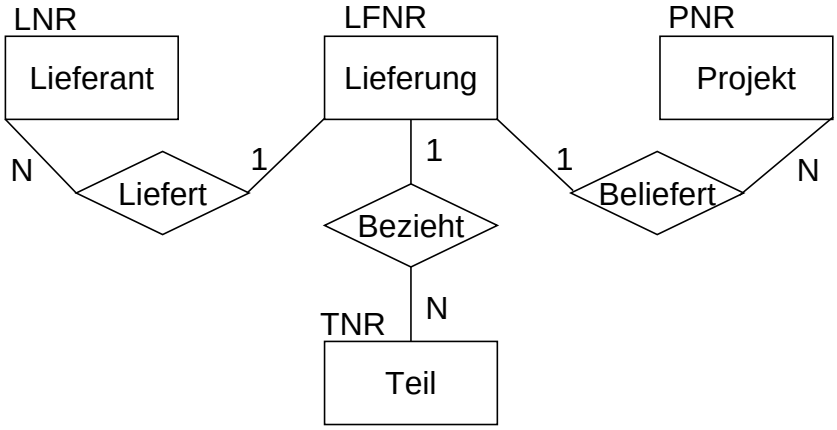


Angabe einiger zulässiger Instanzen (Tupel) mit t_1 , $t_2 \in$ TNR, l_1 , $l_2 \in$ LNR, p_1 , $p_2 \in$ PNR in den zugehörigen E- und R-Relationen

Welche unerwünschten (falschen) Zustände sind möglich?

- 
LNR
Lieferant
N
Lieferung
M
PNR
Projekt
P
TNR
Teil


LNR
Lieferant
M
Beliefert
N
PNR
Projekt
M
Liefert
Bezieht
M
TNR
N
Teil
N

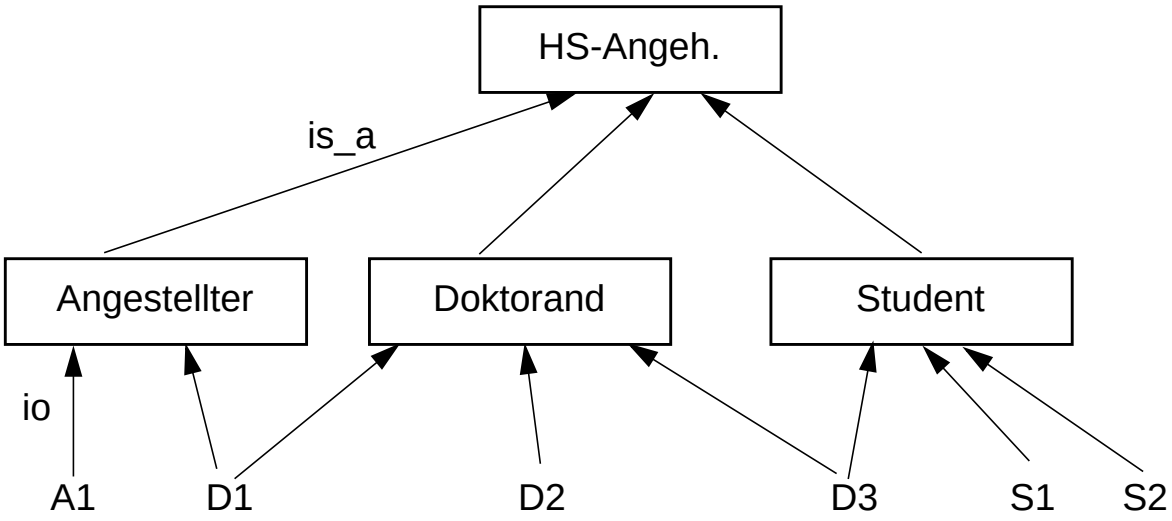

LNR
Lieferant
LFNR
Lieferung
PNR
Projekt
N
Liefert
1
1
Bezieht
1
Beliefert
N
N
TNR
Teil

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 –

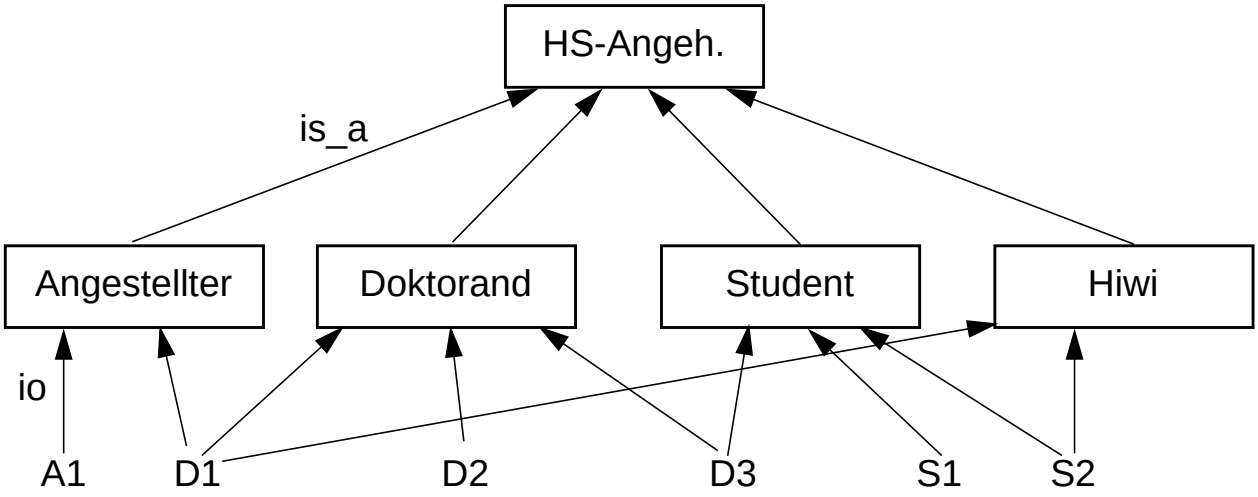
Mehrklassen-Mitgliedschaft von Instanzen



- **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**

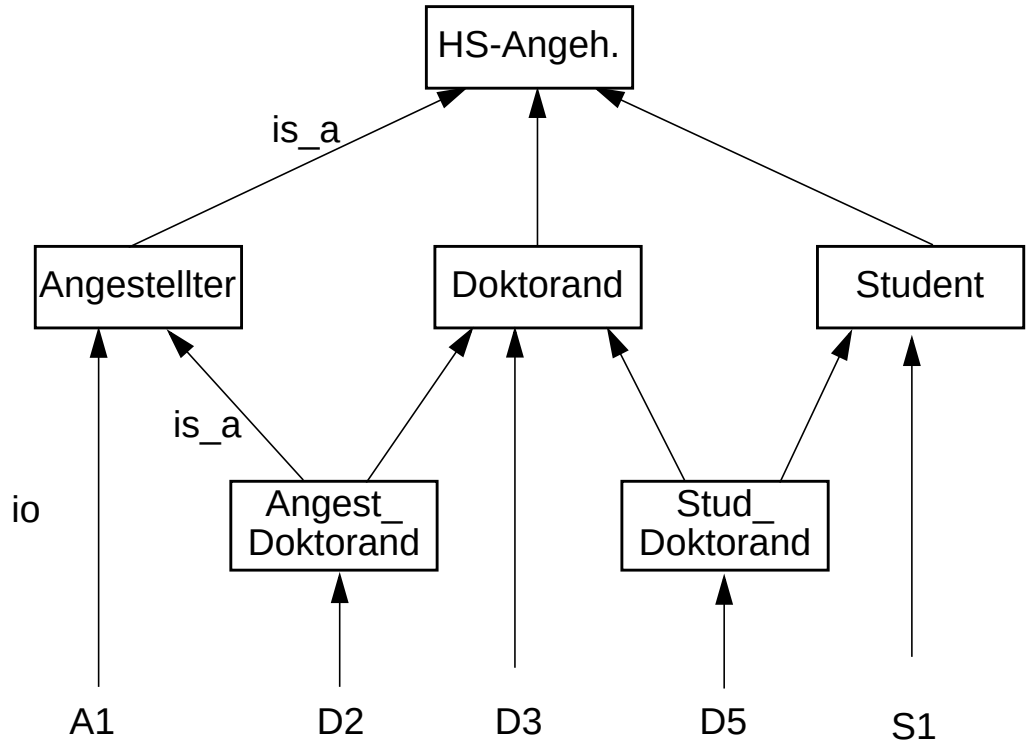
1. Zerlegung der Superklasse G in Subklassen S_i, wobei S_i eigenständige Instanzen besitzen kann

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**

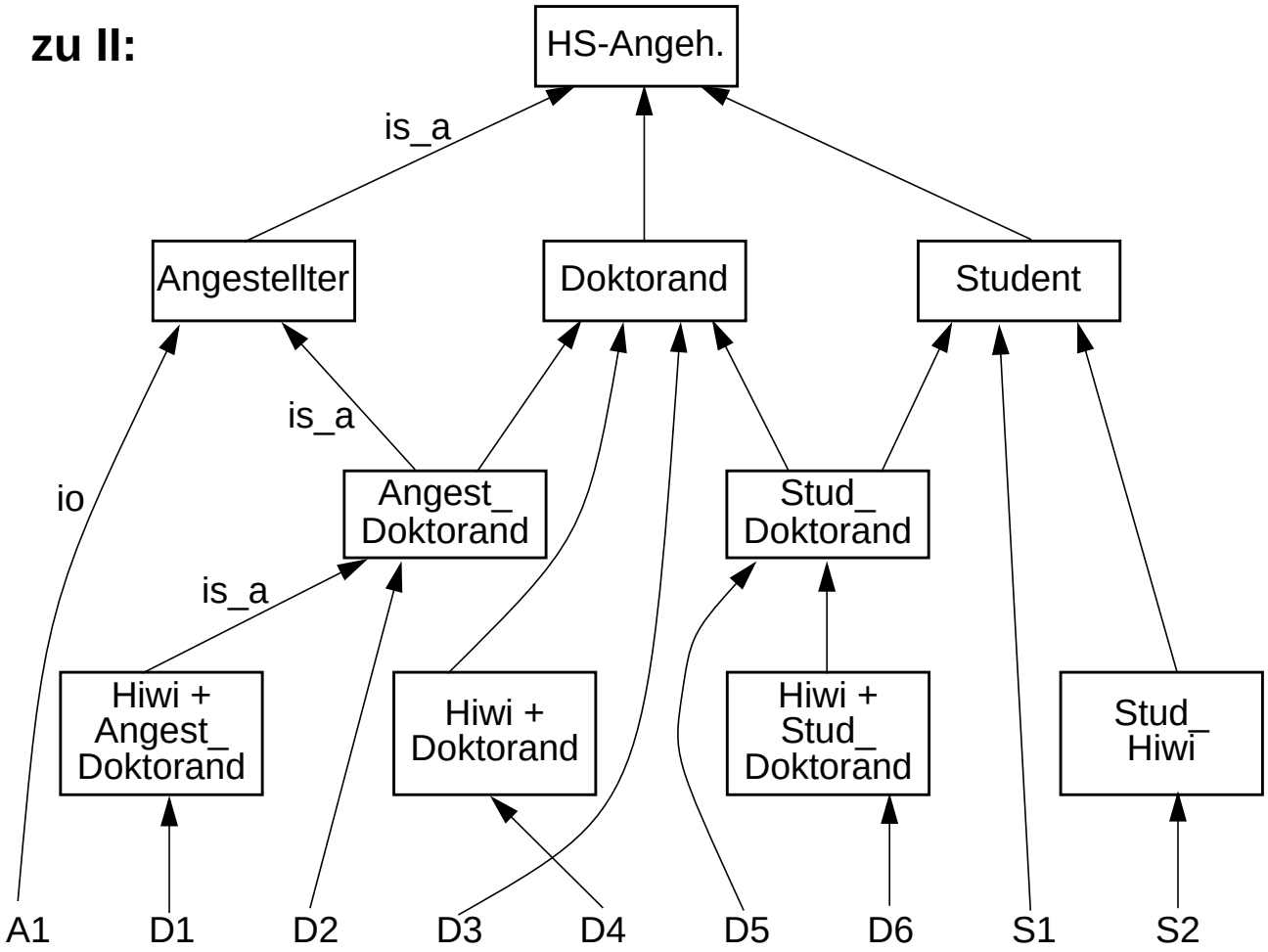


Spezialisierung bei Überlappungen (4)

- **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):

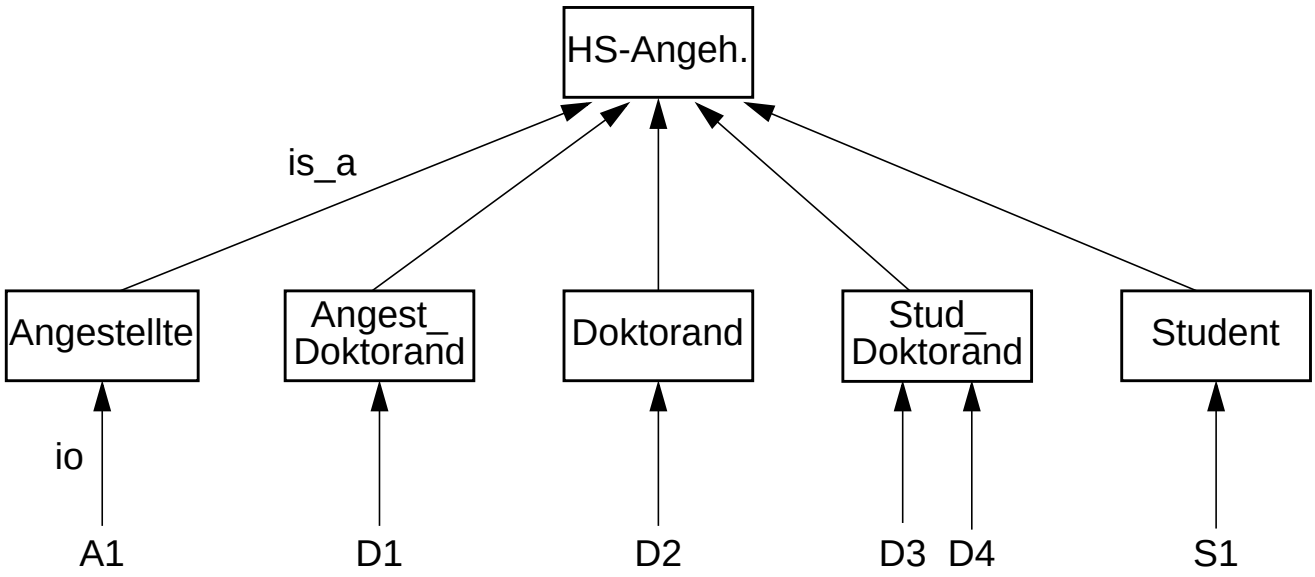
    Select *
    From Angest_Doktorand

    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.)

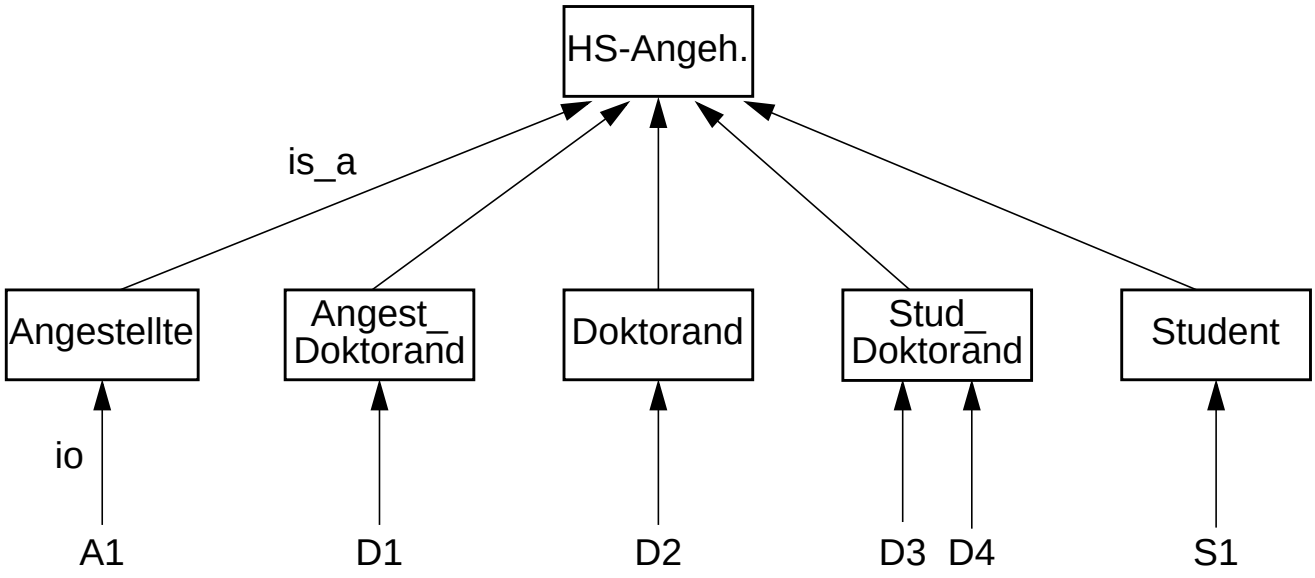
    Insert D2 INTO Doktorand (OID = 0815, Dokt.-Attr.)

    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?