2. Informationsmodelle

Vorlesung "Informationssysteme"

Sommersemester 2015

Technische Universität
Kaiserslautern

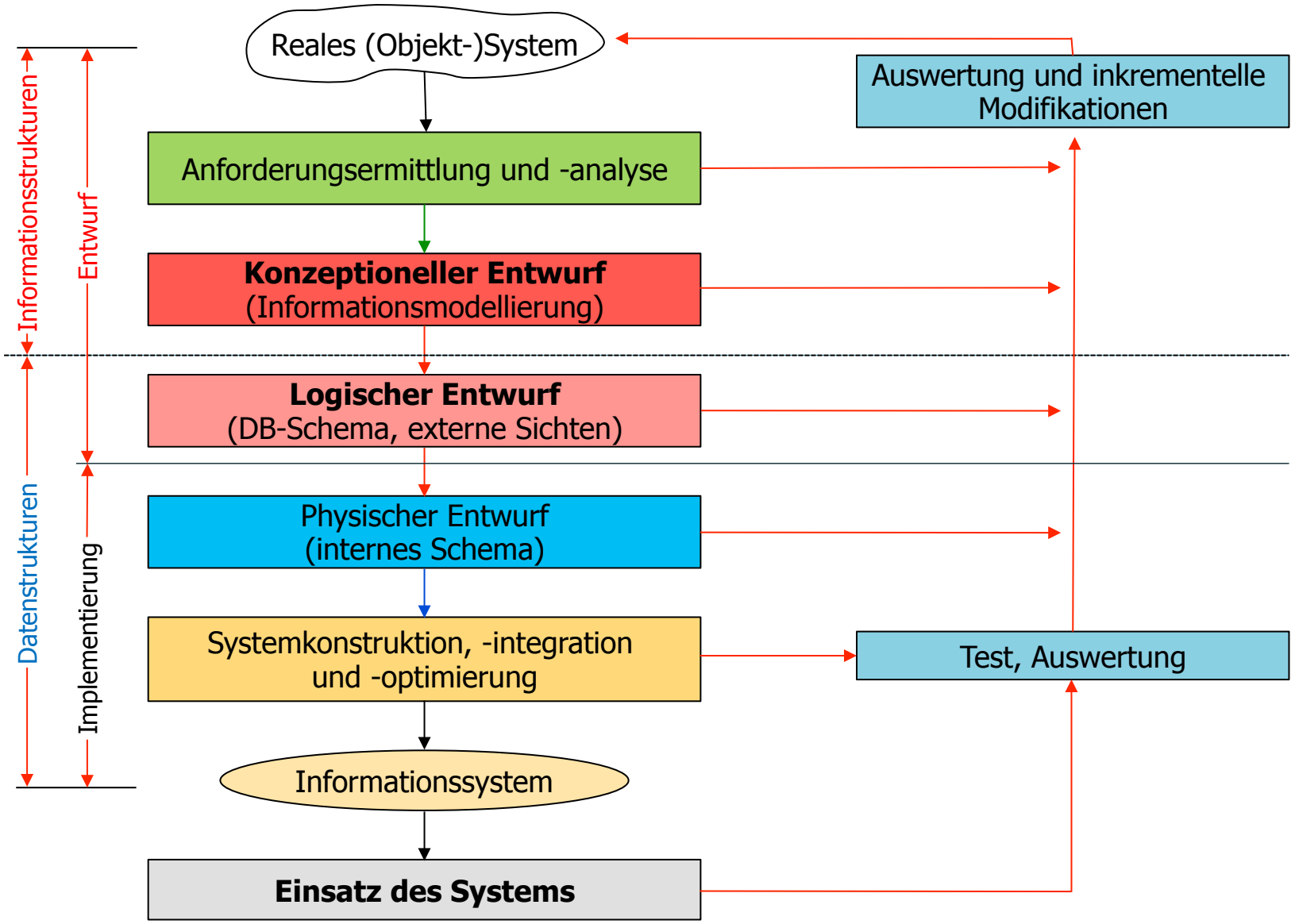
Überblick

- Vorgehensweise bei DB-Entwurf und -Modellierung
  - Lebenszyklus
  - Informationserhebung
- Entity-Relationship-Modell (ERM)
  - Definitionen, Konzepte
  - Beziehungstypen
  - Diagrammdarstellung
  - Beispiele
- Erweiterungen des ERM
  - Kardinalitätsrestriktionen
  - Abstraktionskonzepte
- Abstraktionskonzepte
  - Generalisierung
  - Aggregation

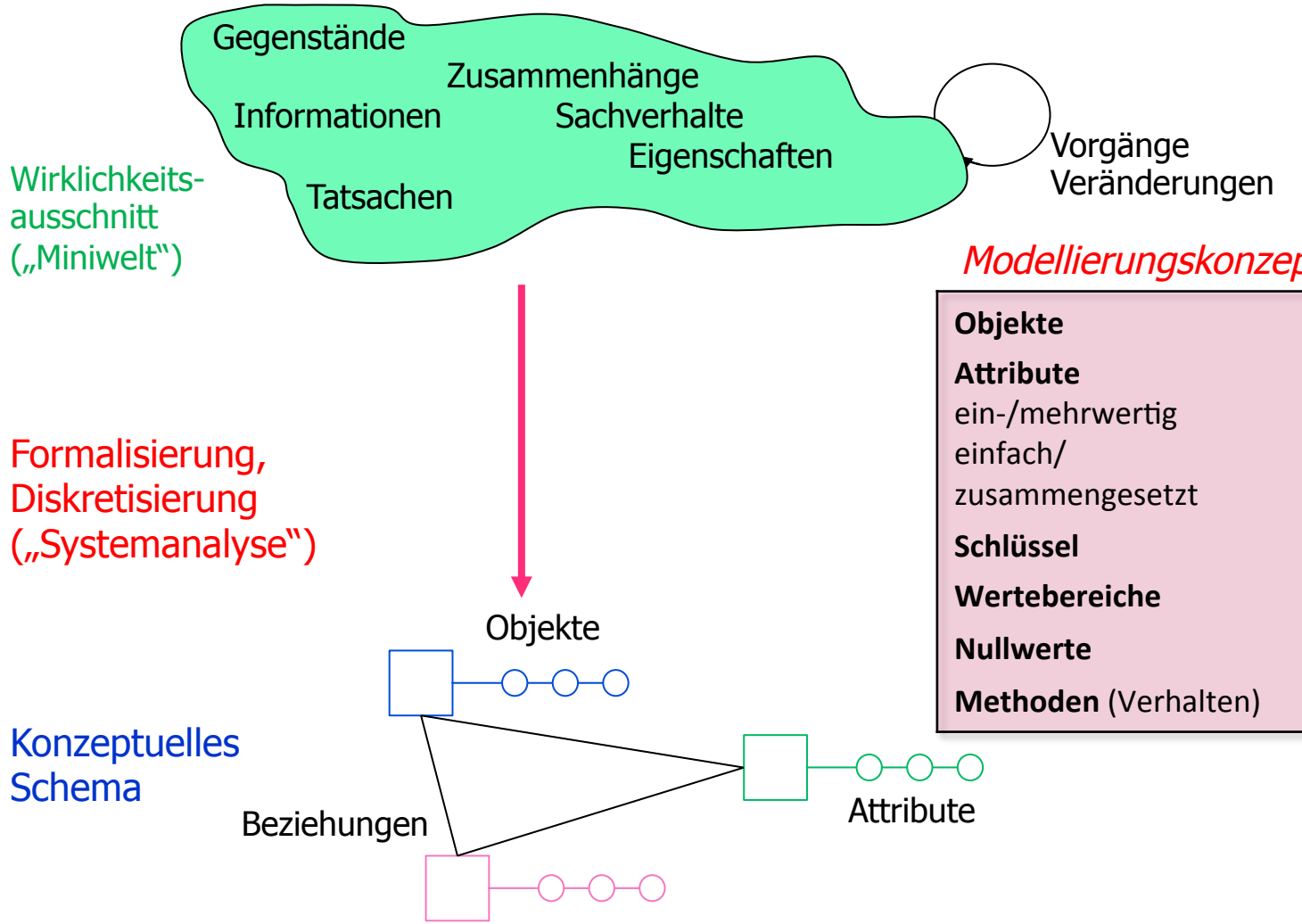
Vorgehensweise bei DB-Entwurf und -Modellierung

- Ziel: Modellierung einer Miniwelt (Entwurf von Datenbankschemata)
- Schrittweise Ableitung: ("top-down"-Entwurf)
  1. Information in unserer Vorstellung
  2. Informationsstruktur: Organisationsform der Information
  3. Logische Datenstruktur (zugriffspfadunabhängig, Was-Aspekt)
  4. Physische Datenstruktur (zugriffspfadabhängig, Was- und Wie-Aspekt)
- Nebenbedingungen
  - genaue Abbildung
  - hoher Grad an Aktualität
  - Verständlichkeit, Natürlichkeit, Einfachheit, ...

Schritte auf dem Weg zu einem IS



Informationsmodelle

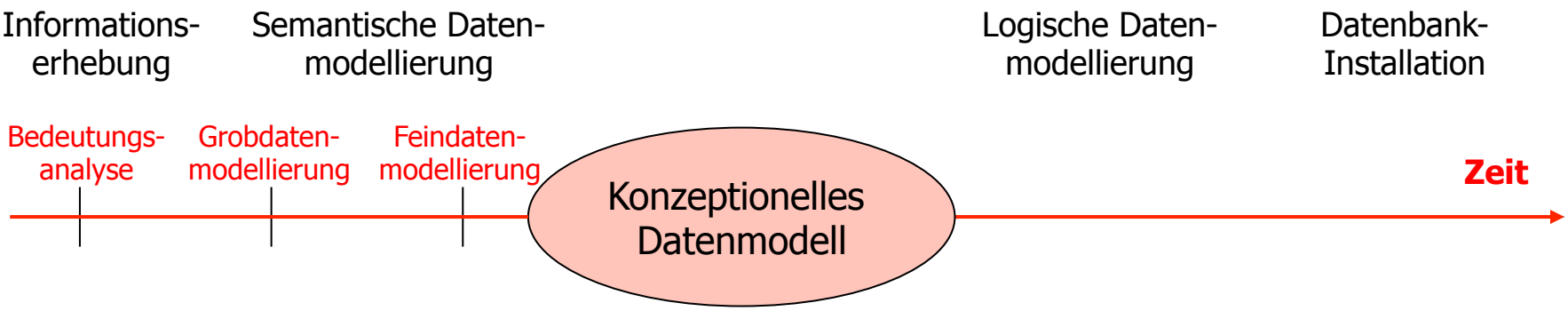


Modellierungskonzepte

Objekte	**Beziehungen**
Attribute	Typ, Grad
ein-/mehrwertig	Optional
einfach/	existenzabhängig
zusammengesetzt	**Abstraktionskonzepte**
Schlüssel	Klassifikation
Wertebereiche	Generalisierung
	Aggregation
Nullwerte	Assoziation
Methoden (Verhalten)	**Rollen**

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

Von der Informationserhebung zum DB-Schema

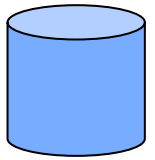


- Interview
- Substantivanalyse
- Brainstorming
- Dokumentenanalyse
- . . .

- ERM
- NIAM
- EXPRESS-G
- IDEF1X
- UML
- STEP
- . . .

Unabhängig von der Eingesetzten Datenbank-verwaltungssoftware

- hierarchisch
- netzwerkförmig
- relational
- objektorientiert
- objekt-relational
- . . .



- DB2
- ORACLE
- MS SQL Server
- MySQL
- . . .

konzeptioneller DB-Schemaentwurf

logischer Schemaentwurf

physischer Schemaentwurf

Wichtige Modellierungssprachen

- ERM (**E**ntity **R**elationship **M**odel):
  - generell einsetzbares Modellierungswerkzeug, hauptsächlich für den relationalen DB-Entwurf geeignet
- UML (**U**nified **M**odeling **L**anguage):
  - Notation und Sprache zur Unterstützung der objekt-orientierten Modellierung im Software Engineering: Es gibt sehr viele Untermodelle für den Entwurf von Softwaresystemen auf den verschiedensten Abstraktionsebenen

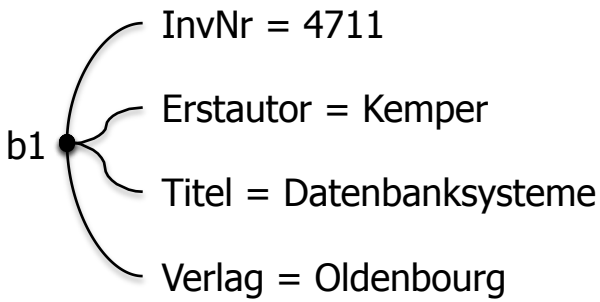
Entity-Relationship-Modell (ERM) – Überblick

- Konzepte
  - Entities und Attribute
  - Entity-Mengen/Typen
  - Primärschlüssel
  - Relationship-Mengen/Typen
- Klassifikation der Beziehungstypen
  - benutzerdefinierte Beziehungen
  - Abbildungstyp: 1:1; n:1; n:m
- Ziel: Modellbildung auf der Typebene (Schema)!
  - Festlegung von semantischen Aspekten
  - explizite Definition von strukturellen Integritätsbedingungen
  - diese gelten für alle (möglichen) Instanzen des Schemas
- ER-Diagramm: graphische Darstellung eines ER-Schemas

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

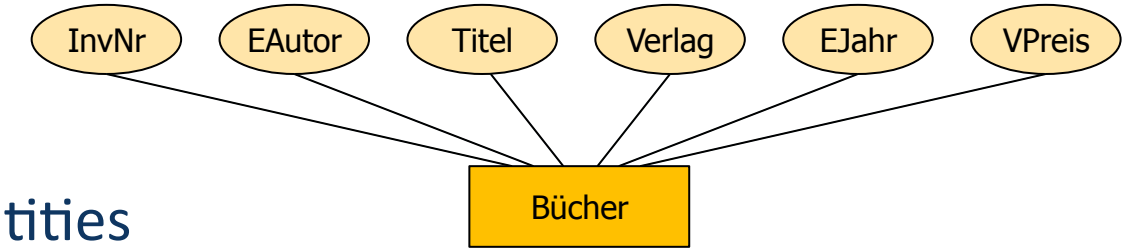
Entities und Attribute

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



Entity-Mengen und Entity-Typen

- Entity-Mengen (Entity-Sets)
  - Gruppierung von „ähnlichen“ oder „vergleichbaren“ Entities mit gemeinsamen Eigenschaften
- Entity-Typ definiert das Schema einer Entity-Menge
  - Name
  - Attribute
- Entity-Menge und ihre Entities sind zeitveränderlich (Ausprägung)

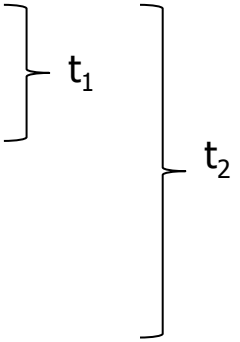


Diagrammdarstellung

e1 = (4711, Kemper, DBS, Oldenbourg, ...)

e2 = (0815, Date, Intro. to DBS, Addison, ...)

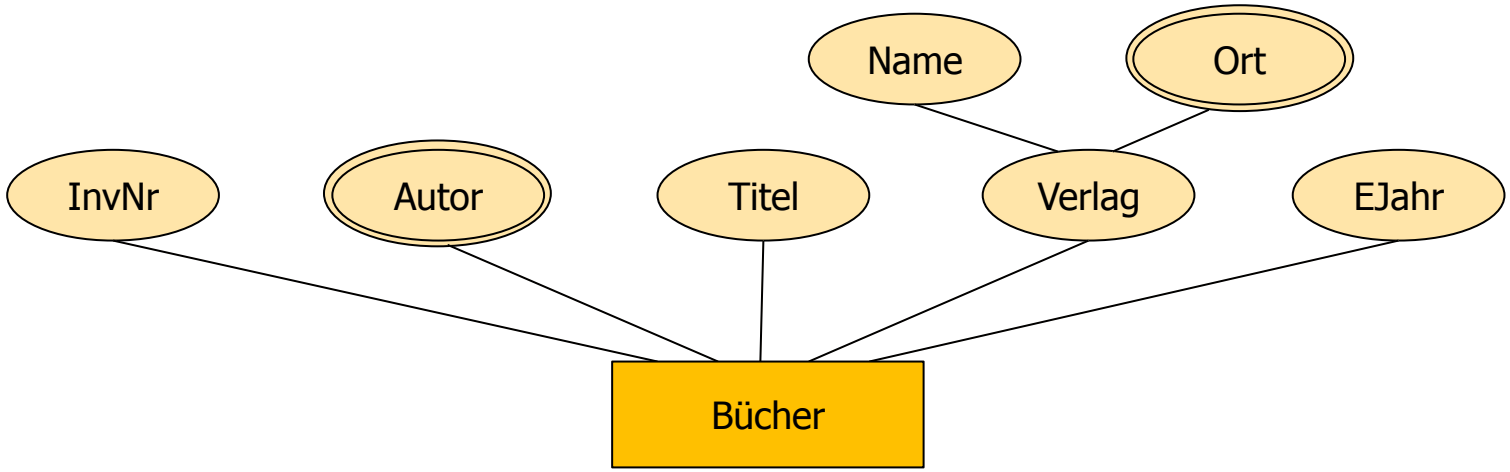
e3 = (1234, Härder, DBS, Springer, ...)



- Nochmal: ER-Schema/Diagramm enthält keine Ausprägungen!

Komplexe Attribute

- Wie wird modelliert, wenn
  - ein Buch mehrere Autoren hat
  - die Verlagsinformation zusammengesetzt ist (Name, Ort)
  - Eigenschaften hierarchisch gegliedert sind
- Erhöhung der Modellierungsgenauigkeit
  - bisher: einwertige Attribute
  - mehrwertige Attribute (Doppelovale)
  - zusammengesetzte Attribute (hierarchisch angeordnete Ovale)
  - ➪ Verschachtelungen sind möglich



e3 = (1234, {Härder, Rahm}, DBS, (Springer, {Hd, Be}), 2001)

Wertebereich/Domain eines Attributs

- Wertebereich W(A) (oder dom(A))
  - Die möglichen oder „zulässigen“ Werte für ein Attribut
  - Ein Attribut ordnet jedem Entity einer Entity-Menge einen Wert aus einem bestimmten Wertebereich (dem des Attributs) zu

Attribut	Wertebereich
InvNr	$i \in N, i \leq 10^5$
EAutor	char (40)
EJahr	{1571, ... 2012}
VPreis	DECIMAL(9, 2)

- Domain dom(A) für atomare und komplexe Attribute

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

- Wertebereiche sind nicht Teil des ER-Diagramms, können getrennt spezifiziert werden

(Primär-) Schlüssel und Schlüsselkandidat

- Wie wird ein Entity identifiziert?
  - Entities müssen anhand von Attributwerten „wohlunterscheidbar“ sein
- Identifikation eines Entities durch Attribut (oder Kombination von Attributen)
  - Attributwerte eindeutig für alle Entities derselben Entitymenge
  - ggf. künstlich erzwungen (z.B. laufende Nr.)
- $\{A_1, A_2, ..., A_m\} = A$ sei Menge der (einwertigen) Attribute zur Entity-Menge E

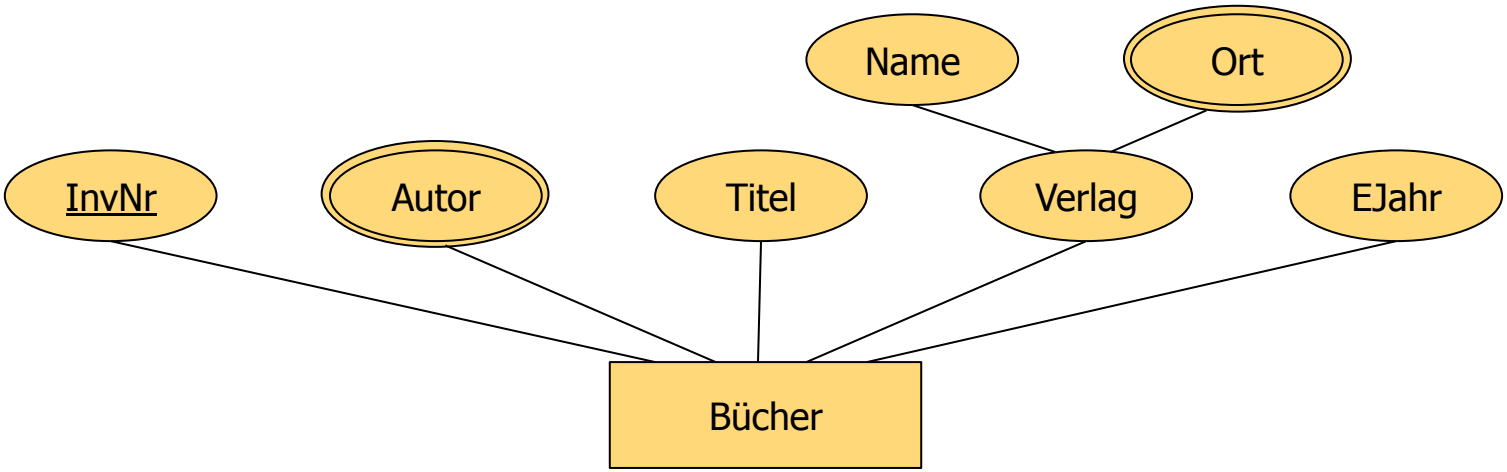
  $K \subseteq A$ heißt Schlüsselkandidat von E

  $\Leftrightarrow$ K eindeutig: $e_i, e_j \in E;\ e_i \neq e_j \rightarrow K(e_i) \neq K(e_j)$

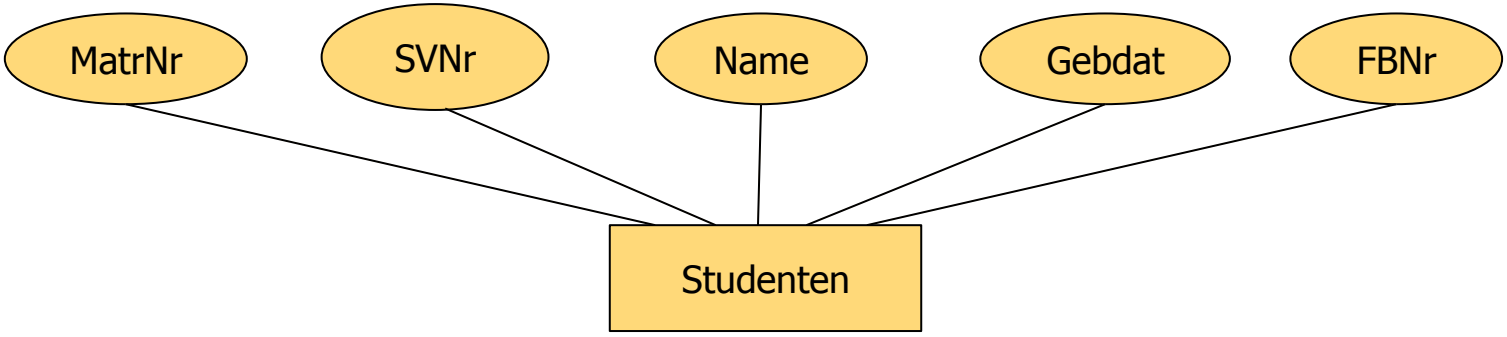
  und K minimal (irreduzibel): $\forall K' \subset K$: K‘ ist nicht eindeutig
- Falls mehrere Schlüsselkandidaten (SK): Primärschlüssel auswählen
  - im ER-Diagramm durch Unterstreichen gekennzeichnet

Beispiel

- Primärschlüssel für Entity-Typ "Buch"



- Entity-Typ Student



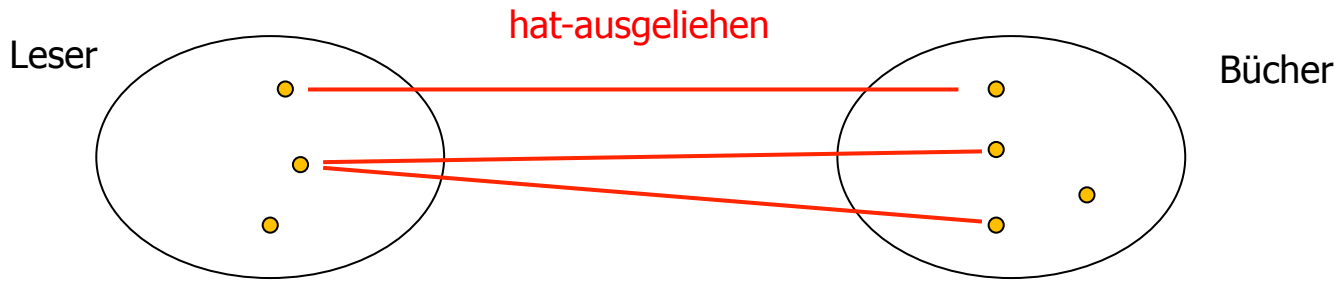
Schlüsselkandidaten? Primärschlüssel?

Relationships und Relationship-Mengen

- Relationship (Beziehung)
  - Assoziation zwischen Entities
- Relationship-Mengen

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

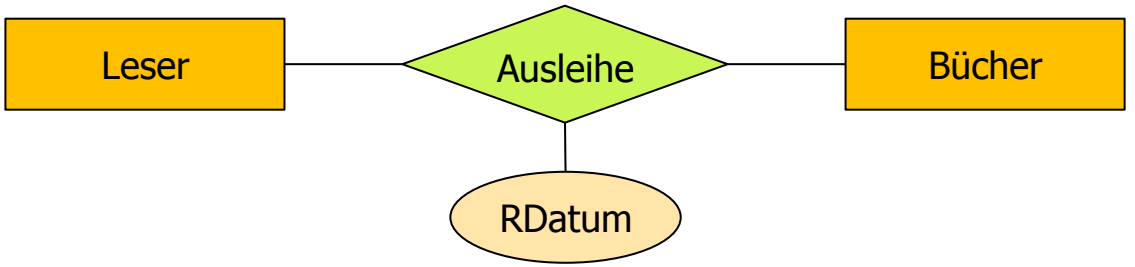
  z.B. „hat ausgeliehen“ zwischen „Leser“ und „Bücher“



Instanzbeispiel: $r_1 = (l_1, b_5)$

Relationship-Typ

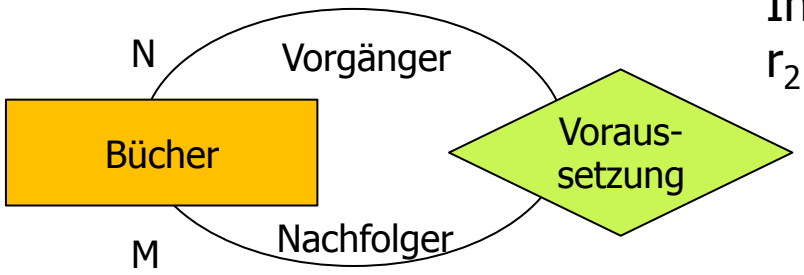
- Relationship-Typ (Beziehungstyp) definiert
  - Name der Relationship
  - beteiligte Entity-Typen und Rollennamen (und damit Grad der Beziehung)
    - Relationship ist Teilmenge des kartesichen Produkts der beteiligten Entity-Mengen
  - Attribute (optional)
    - unabhängig von Entity-Typen
  - Funktionalitäten: 1:1, 1:n/n:1, n:m
- ER-Diagramm



  - Grad?
  - Funktionalität?

Rollennamen

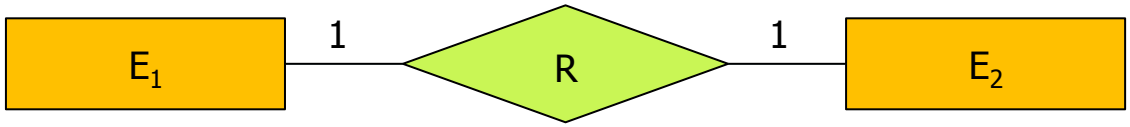
- Motivation für Rollennamen



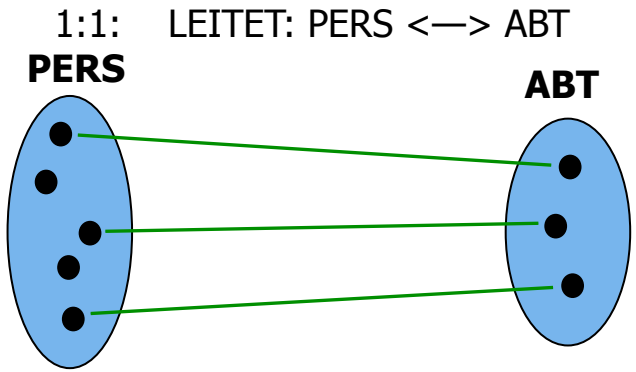
Instanzbeispiel:
r_2 = (Vorgänger: b_1, Nachfolger: b_3)

- Entity-Typ (hier "Buch") nimmt mehr als einmal an Relationship teil
  - rekursive Beziehung
  - der gleiche Entity-Typ spielt verschiedene Rollen
    - Explizite Spezifikation von Rollen erforderlich
    - Rollennamen werden im Diagramm an die Beziehungskanten geschrieben
  - Grad der Beziehung (hier: 2) durch Rollen bestimmt
- Transitivität gilt bei Selbstreferenz i.A. nicht!
  - Bsp: „p_i kennt p_j“

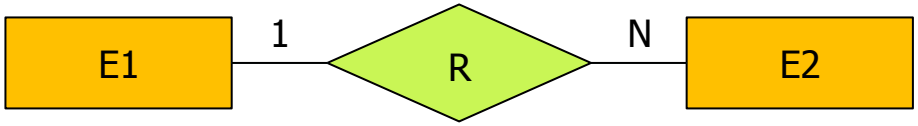
Funktionalität binärer Beziehungen – 1:1

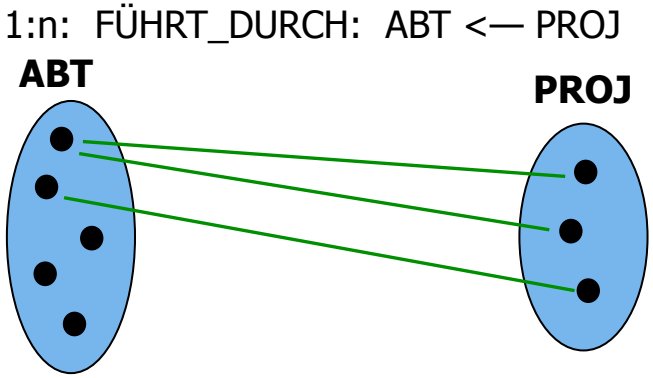


- Jedem Entity e_1 aus E_1 ist maximal 1 Entity e_2 aus E_2 zugeordnet, und umgekehrt
- Es kann auch Entities in E_1 und E_2 geben, denen kein anderes Entity zugeordnet ist

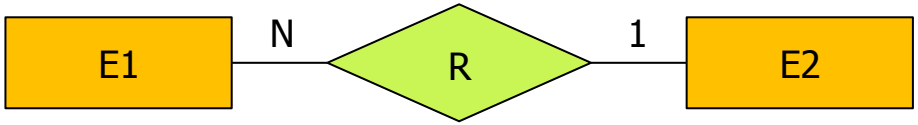


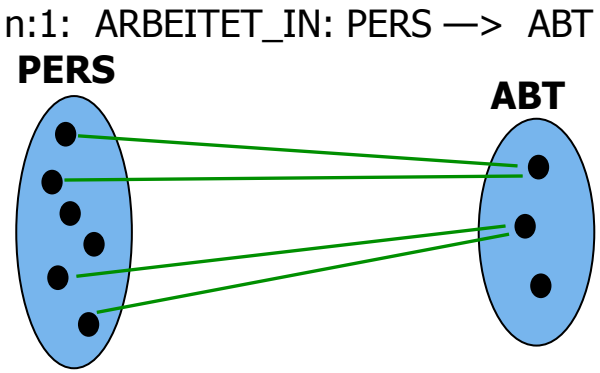
Funktionalität binärer Beziehungen – 1:N/N:1



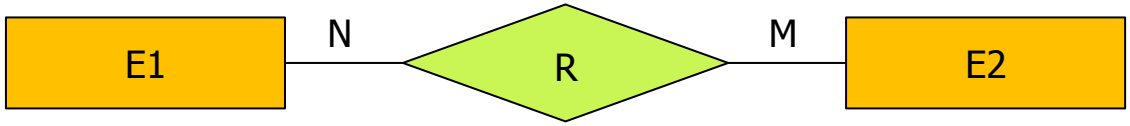


- 1:N
  - Jedem Entity e_1 aus E_1 sind beliebig viele Entities aus E_2 zugeordnet (evtl. auch garkeins)
  - Jedem Entity e_2 aus E_2 ist höchstens 1 Entity e_1 aus E_1 zugeordnet
- N:1 analog

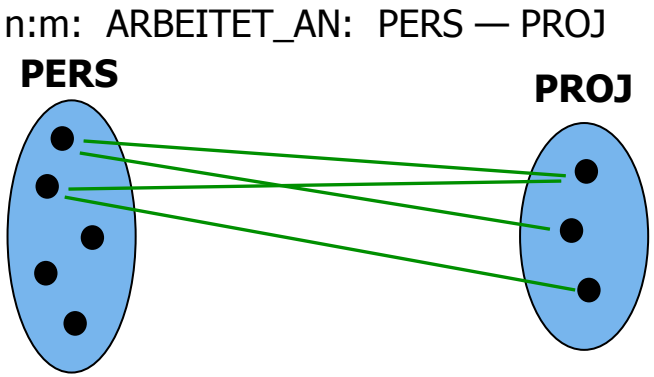




Funktionalität binärer Beziehungen – N:M



- Es gelten keine Restriktionen
- Jedes Entity aus E_1 kann mit beliebig vielen Entities aus E_2 in Beziehung stehen, und umgekehrt

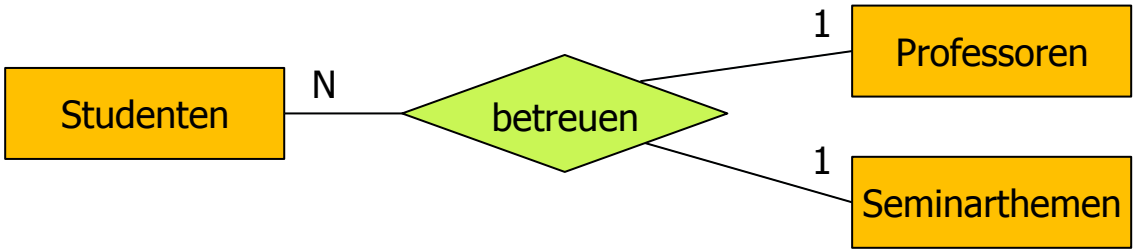


Funktionalitäten bei n-stelligen Beziehungen

- Sei R eine Beziehung zwischen Entity-Mengen E_1, ... E_n und Funktionalität ist "1" für Entity-Typ E_k
  → partielle Funktion E_1, ... E_{k-1}, E_{k+1}, ... E_n → E_k

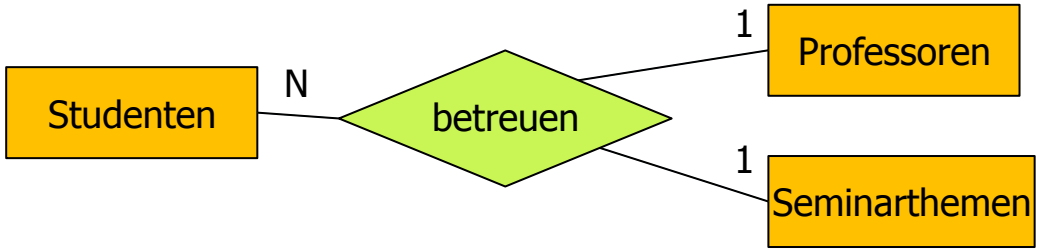
  D.h., jede beliebige Kombination von Entities aus E_1, ... E_{k-1}, E_{k+1}, ... E_n kann aufgrund von R höchstens einem e_k aus E_k zugeordnet sein

- Beispiel



  - betreuen: Professoren X Studenten → Seminarthemen
  - betreuen: Seminarthemen X Studenten → Professoren

Funktionalitäten bei n-stelligen Beziehungen (2)

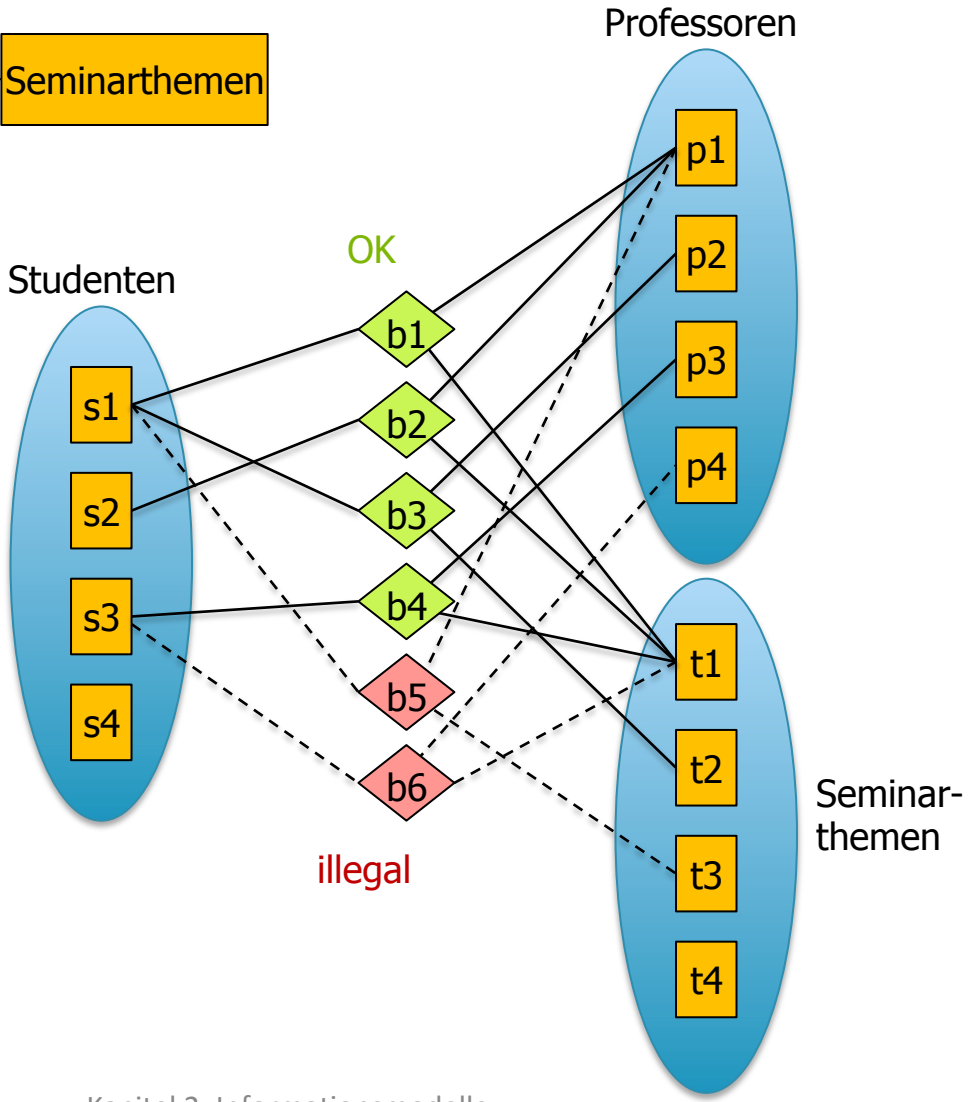


Profs dürfen dasselbe Thema wiederverwenden (bei anderen Studis)

Dasselbe Thema darf von mehreren Profs vergeben werden (an verschiedene Studis)

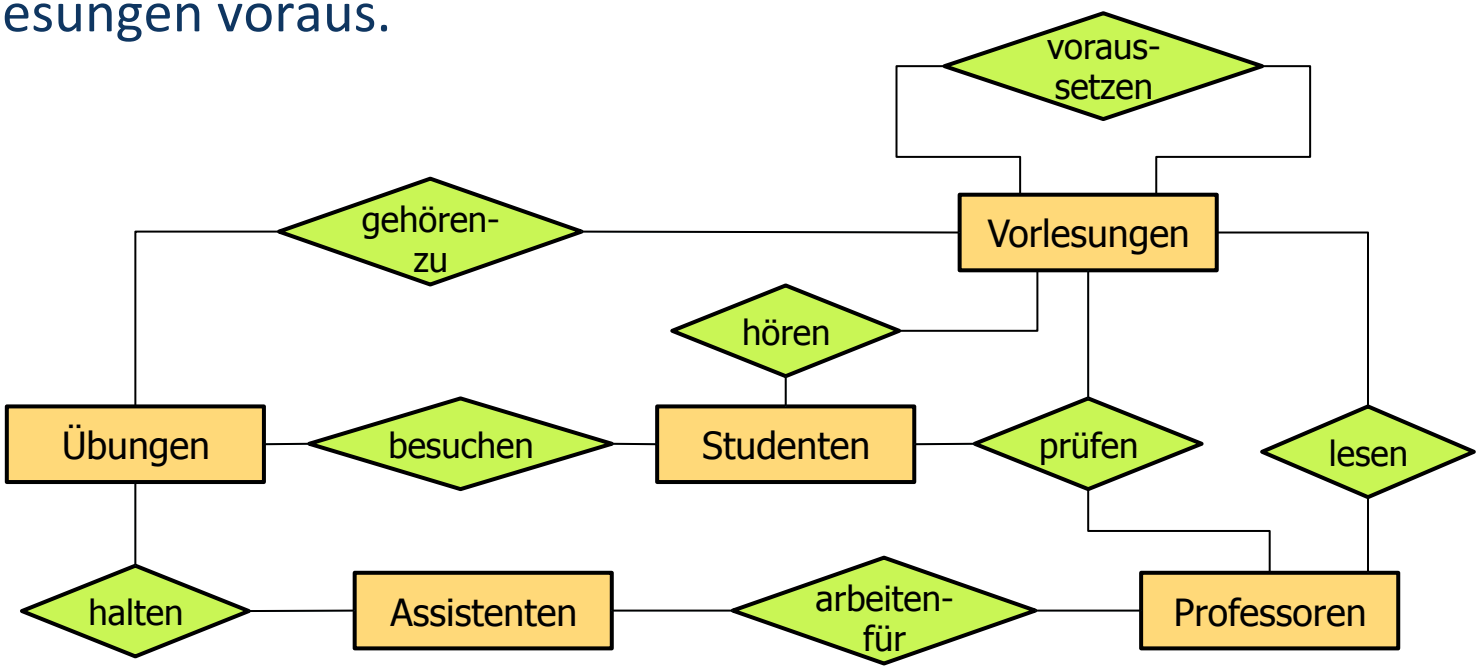
Studis dürfen beim selben Prof nur ein Seminar ableisten

Studis dürfen dasselbe Thema nur einmal bearbeiten

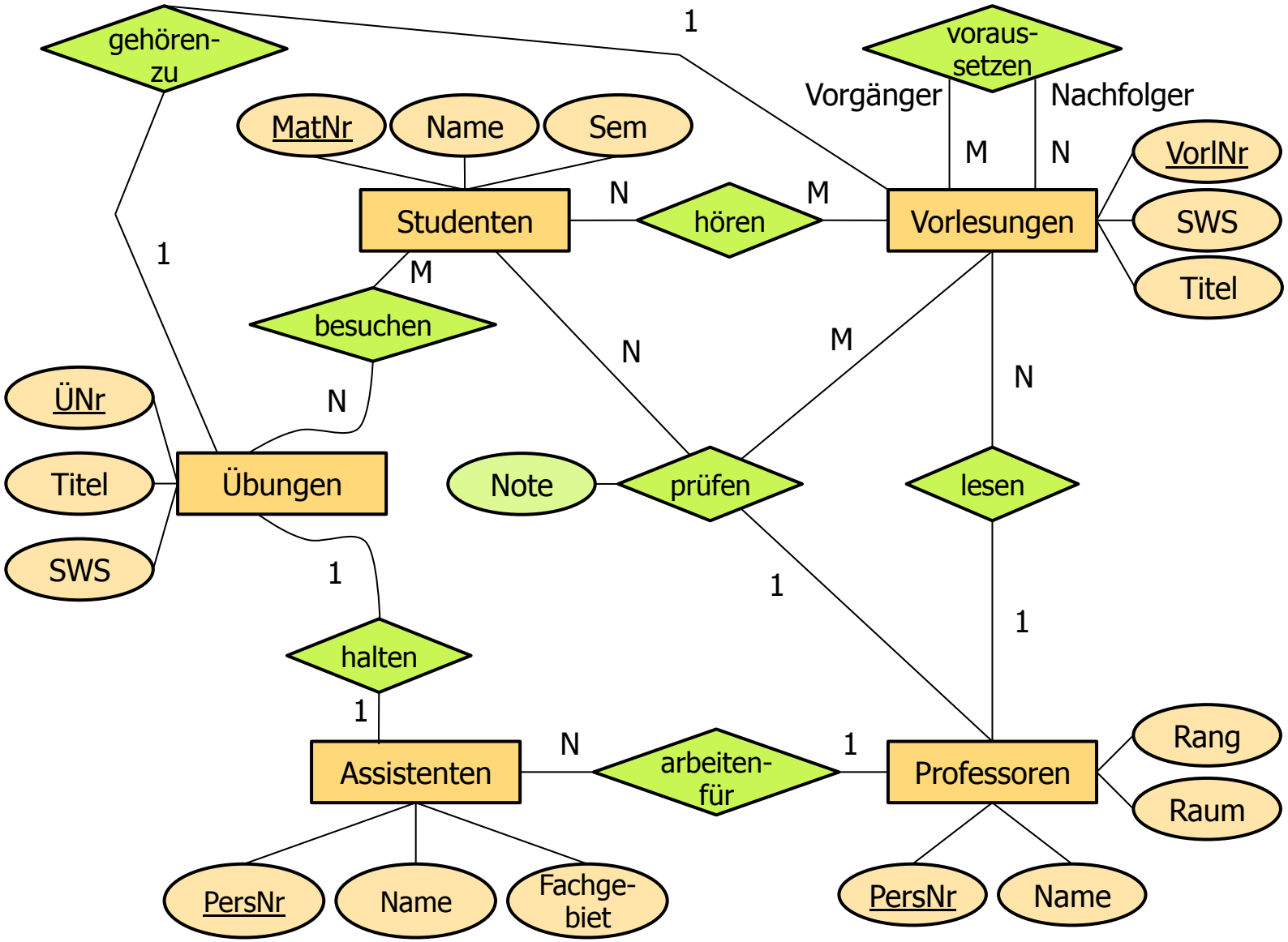


Anwendungsbeispiel: Vorlesungsbetrieb

- Jeder Professor liest mehrere seiner Vorlesungen und prüft Studenten jeweils über eine dieser Vorlesungen. Ein Student darf die gleiche Vorlesung nur von einem Professor prüfen lassen.
- Mehrere Assistenten arbeiten jeweils für einen Professor und halten Übungen, die zu den entsprechenden Vorlesungen gehören.
- Mehrere Studenten hören jeweils eine Reihe von Vorlesungen. Übungen und Vorlesungen werden jeweils von mehreren Studenten besucht.
- Der Besuch von Vorlesungen setzt i. Allg. die Kenntnis anderer Vorlesungen voraus.

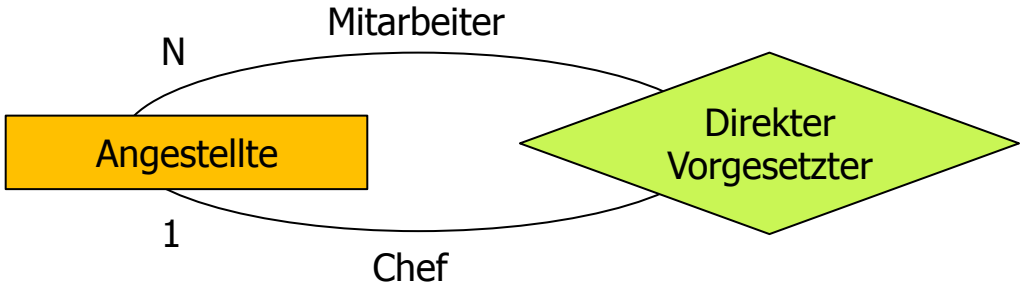


Vollständiges ER-Diagramm - Vorlesungsbetrieb

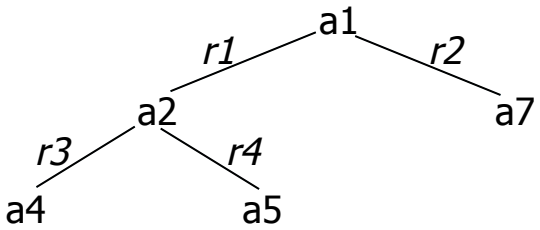


Relationships – Überlappung beteiligter Entity-Mengen

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

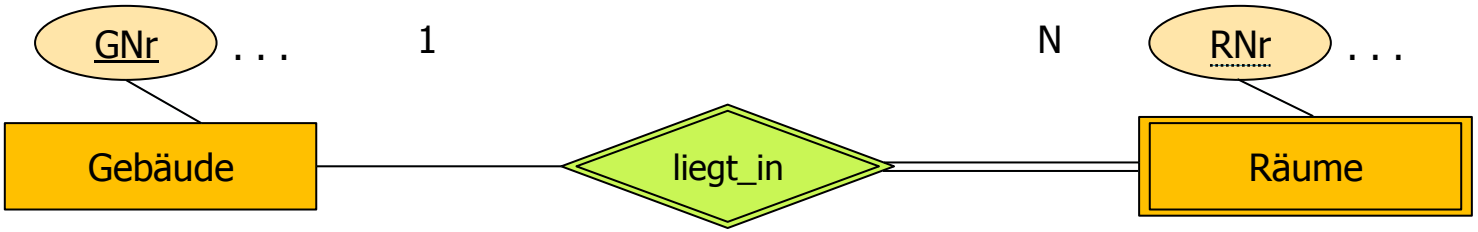


- Eigenschaften
  - Grad: 2
  - Beziehungstyp: n:1
- Welche Beziehungen auf Angestellter sind zulässig?

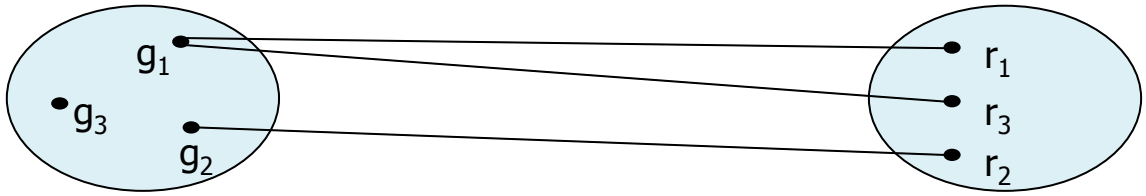


Existenzabhängigkeit

- Schwache (existenzabhängige) Entity-Typen
  - Existenz der (schwachen) Entities ist abhängig von übergeordneten Entities (identifizierender Entity-Typ)
    - Funktionalität der Beziehung: N:1 (oder 1:1)
    - Diagramm: doppelt umrandete Entity-/Relationship-Typen, doppelte Kante
  - Entities sind oft nur in Kombination mit dem Schlüssel des übergeordneten Entitytyps eindeutig identifizierbar
    - Diagramm: lokaler Schlüsselanteil ist gestrichelt unterstrichen
- Beispiel

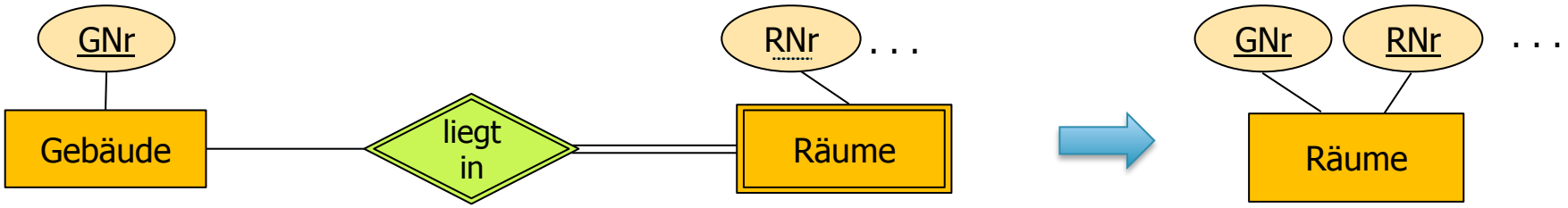


Existenzabhängigkeit: „Relationship begründet Existenz von"



Existenzabhängigkeit (2)

- Existenzabhängigkeit als Integritätsbedingung
  - ist bewusste Entscheidung des Modellierers
  - Oft ist die Wahl/Zusammensetzung des Schlüssels ein Hinweis
    - Beispiel: Primärschlüssel eines Raumes = Gebäudeschlüssel + lokal gültige Raumnummer (z.B. 46, 220)
  - Alternative: falls Gebäude ohne weitere Attribute modelliert sind und quasi zur Beschreibung von Räumen dienen, können die Entity-Typen auch zusammengefasst werden.

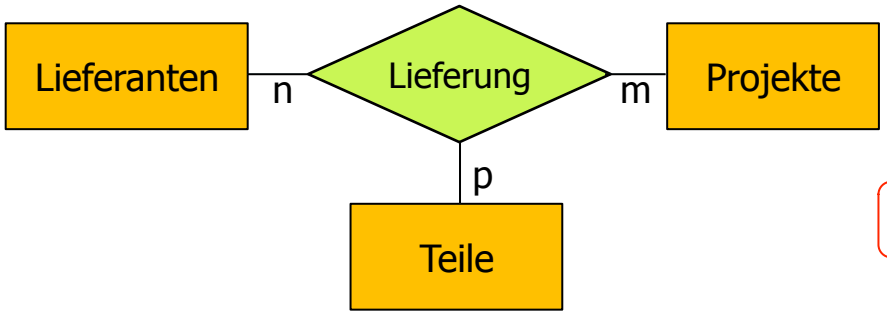


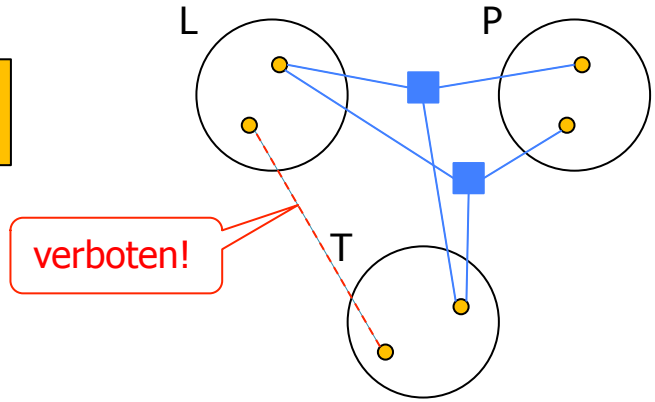
Existenzabhängigkeit kann anwendungsabhängig sein

1. In einer Firma wird Information über Kinder zur Kindergeldberechnung benötigt

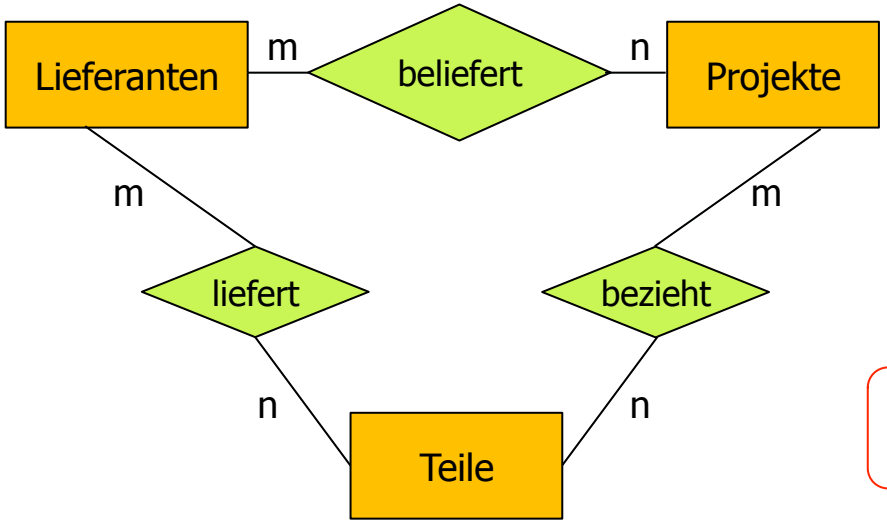
2. In einer Anwendung „Betreuung von Einzelkindern“ müssen Informationen über die Eltern ermittelt werden können:

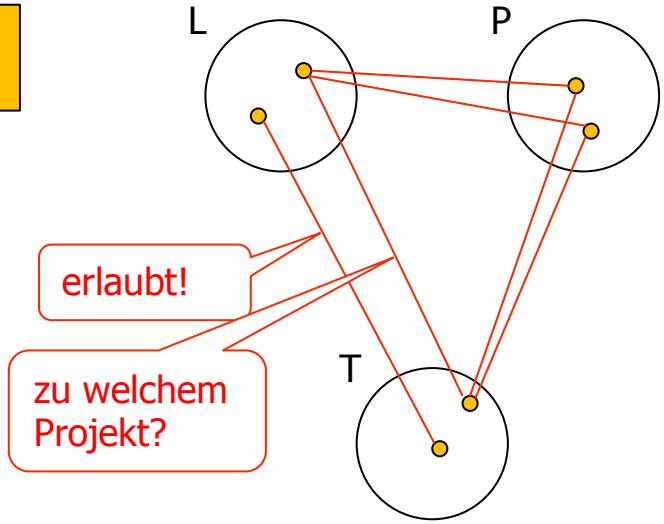
Dreistellige Relationship-Mengen





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

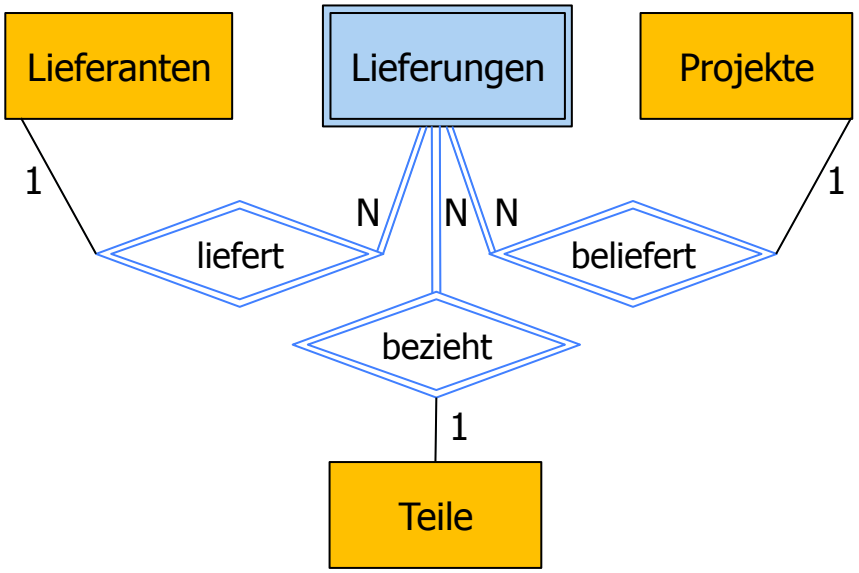


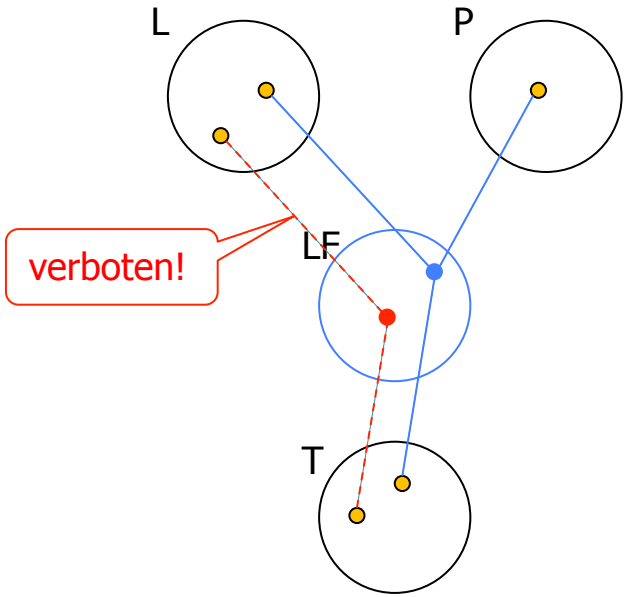


Dreistellige Relationship-Mengen (2)

ABER:
Manche Werkzeuge erlauben nur die Modellierung binärer Relationship-Mengen

Alternative:
Schwacher Entity-Typ mit drei identifizierenden Entities/Relationships





Kardinalitätsrestriktionen – (*min*, *max*)-Notation

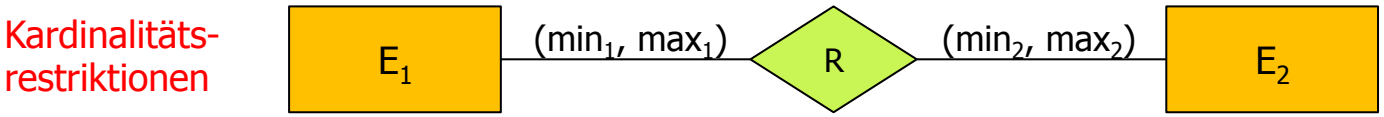
- Bisher:

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

- Verfeinerung der Semantik eines Beziehungstyps

  Sei $R \subseteq E_1 \times E_2 \times \ldots \times E_n$:
  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 $0 \leq min \leq max$, $max \geq 1$).

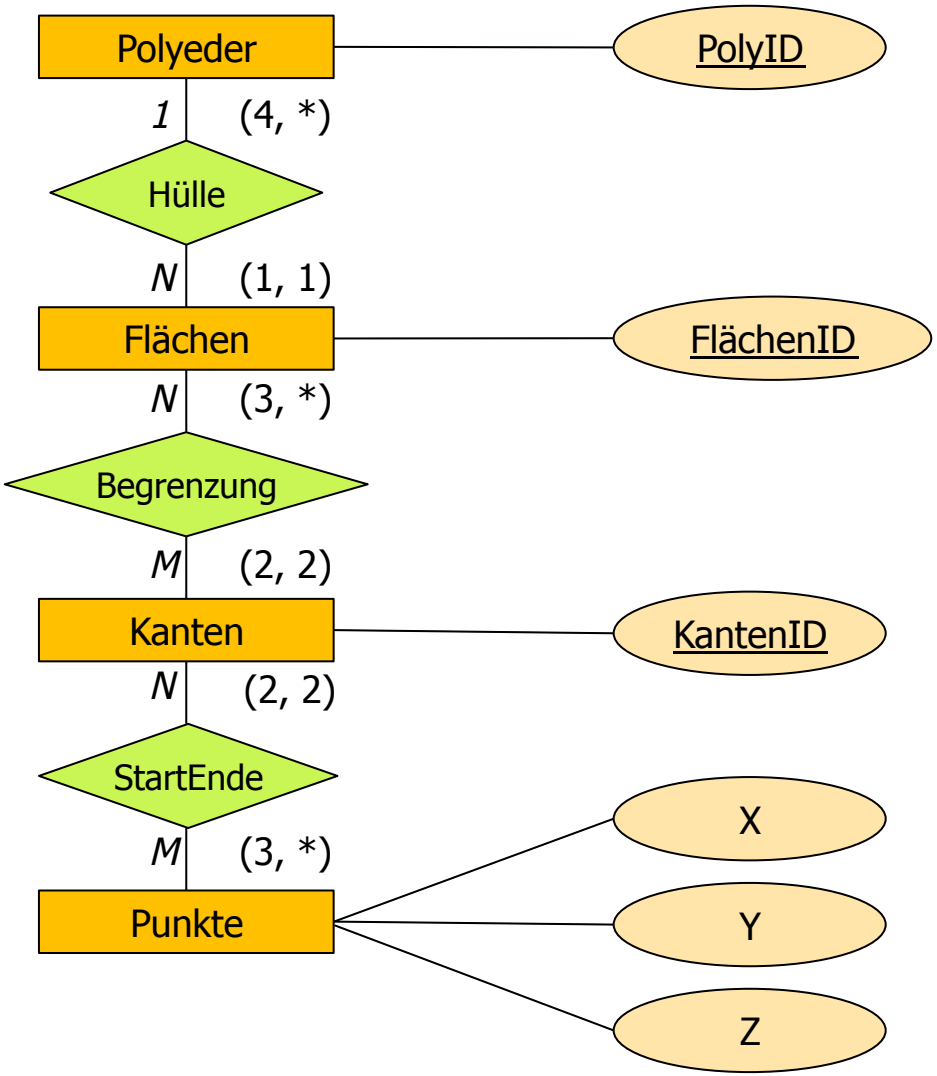
- Graphische Darstellung



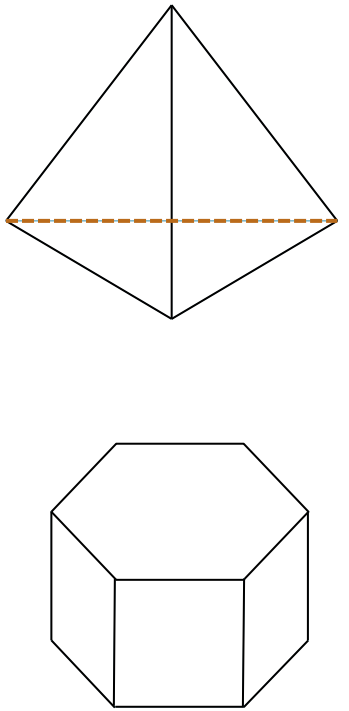
e_1 nimmt an (min_1, max_1) Beziehungen von Typ R teil
e_2 nimmt an (min_2, max_2) Beziehungen von Typ R teil

Beispiel: Begrenzungsflächendarstellung von Körpern

ER-Diagramm



Beispiel-Körper

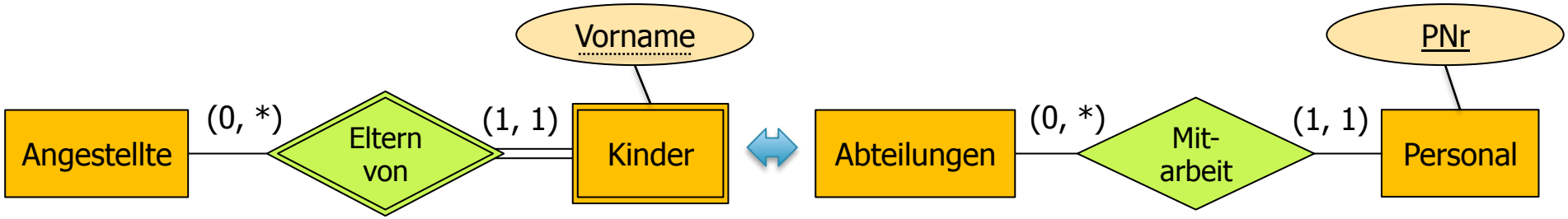


Kardinalitätsrestriktionen – weitere Beispiele

R	E_1	E_2	kard(R, E_1)	kard(R, E_2)
Abt-Leitung	ABT	PERS	(1, 1)	(0, 1)
Heirat	FRAU	MANN	(0, 1)	(0, 1)
Eltern	PAARE	KIND	(0, *)	(1, 1)
Abt-Angehörigkeit	ABT	PERS	(1, 20)	(0, 1)
V.Teilnahme	VORL	STUDENT	(3, 500)	(5, 10)
Mitarbeit	PERS	PROJEKT	(0, 3)	(1, 20)

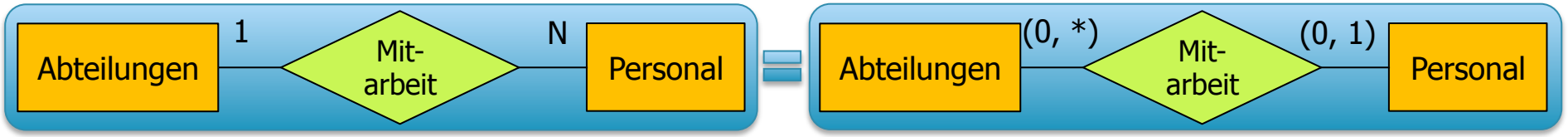
(*min*, *max*) und schwache Entity-Typen

- Wie modelliert man Existenzabhängigkeit eines Entity-Typs?
  - schwacher Entity-Typ
    - existenzabhängig von identifizierendem Entity-/Relationship-Typ
  - Kardinalitätsrestriktion (*min*, *max*)
    - min ≥ 1: jedes Entity muss an mindestens einer Beziehung teilnehmen, ist also existenzabhängig
  - "Überlappung" der Modellierungskonzepte
- Nur lokaler Schlüsselanteil vorhanden → schwacher Entity-Typ
  - auf konsistente (min, max)-Angabe achten! (1, 1)
- In anderen Fällen ist Kardinalitätsrestriktion ausreichend



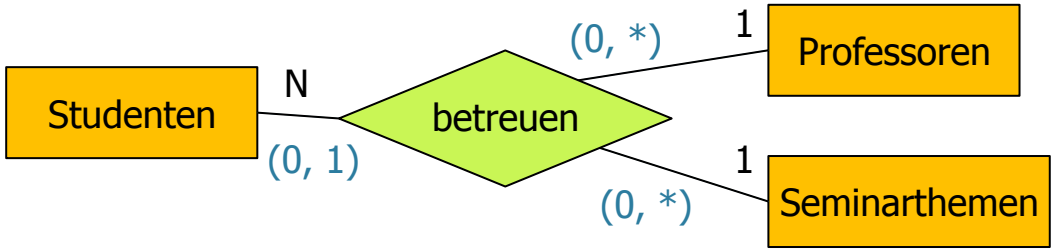
(*min*, *max*) und Funktionalitäten

- Modellierungskonzepte mit unterschiedlicher Semantik
  Beipiel: gegeben R (E_1, ..., E_k, ... E_n):
  - "(0, 1)" bei E_k: jedes e_k kommt höchstens 1 mal in R vor
  - "1" bei E_k: jede Kombination (e_1, ..., e_{k-1}, e_{k+1},... e_n) kommt höchstens 1 mal in R vor
- Spezialfall "Binäre Relationship"
  - Semantik ist vergleichbar, (min, max) ist mächtiger



- Relationship mit Grad > 2
  - Semantik/Ausdruckskraft ist nicht mehr vergleichbar
  - Es gibt Strukturbedingungen, die mit Funktionalitäten ausgedrückt werden können, aber nicht mit (min, max) – und umgekehrt!
- Beide Konzepte können kombiniert werden!

(*min*, *max*) und Funktionalitäten - Beipiel



Funktionalitäten:

Profs dürfen dasselbe Thema wiederverwenden (bei anderen Studis)

Dasselbe Thema darf von mehreren Profs vergeben werden (an verschiedene Studis)

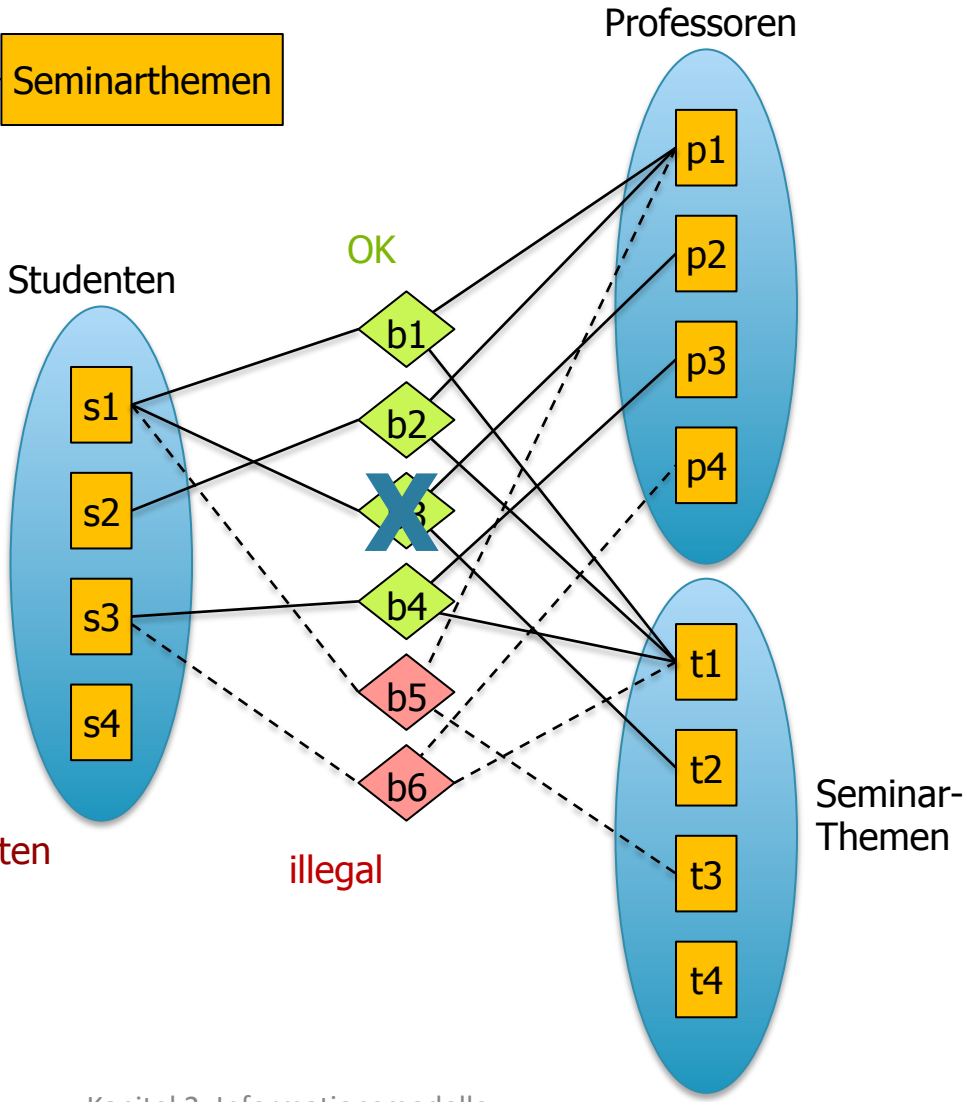
Studis dürfen beim selben Prof nur ein Seminar ableisten

Studis dürfen dasselbe Thema nur einmal bearbeiten

gefordert:

Studis dürfen maximal ein Seminar ableisten (egal bei welchem Prof)

➔ (0, 1) bei "Studenten"



Entwurfsprinzipien

- Korrektheit
  - Entwurf soll die Miniwelt korrekt darstellen
  - Erfordert instensiven Austausch mit Anwendungsexperten
- Vermeidung von Redundanz
  - Das gleiche Konzept sollte nur einmal modelliert werden
    - Bsp. "Studenten": entweder "studiert-in"-Beziehung oder "FBR"-Attribut, aber nicht beides
  - ableitbare Beziehung sollten nicht explizit gemacht werden
  - sonst Gefahr von Inkonsistenzen, Speicherplatzverschwendung
- Einfachheit ist wichtig
  - Vermeidung von unnötiger Detaillierung
  - keine Verfeinerung in Entities/Relationships, wenn nicht von der Anwendung benötigt

Abstraktionskonzepte – semantische Modellierung

- Ziel:
  - Erfassung von noch mehr Semantik aus der Miniwelt
  - 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
- Unterstützung im ER-Modell → enhanced ER-Modell (EER-Modell)
  - Auf Strukturaspekte beschränkt (kein Verhalten/Methoden)

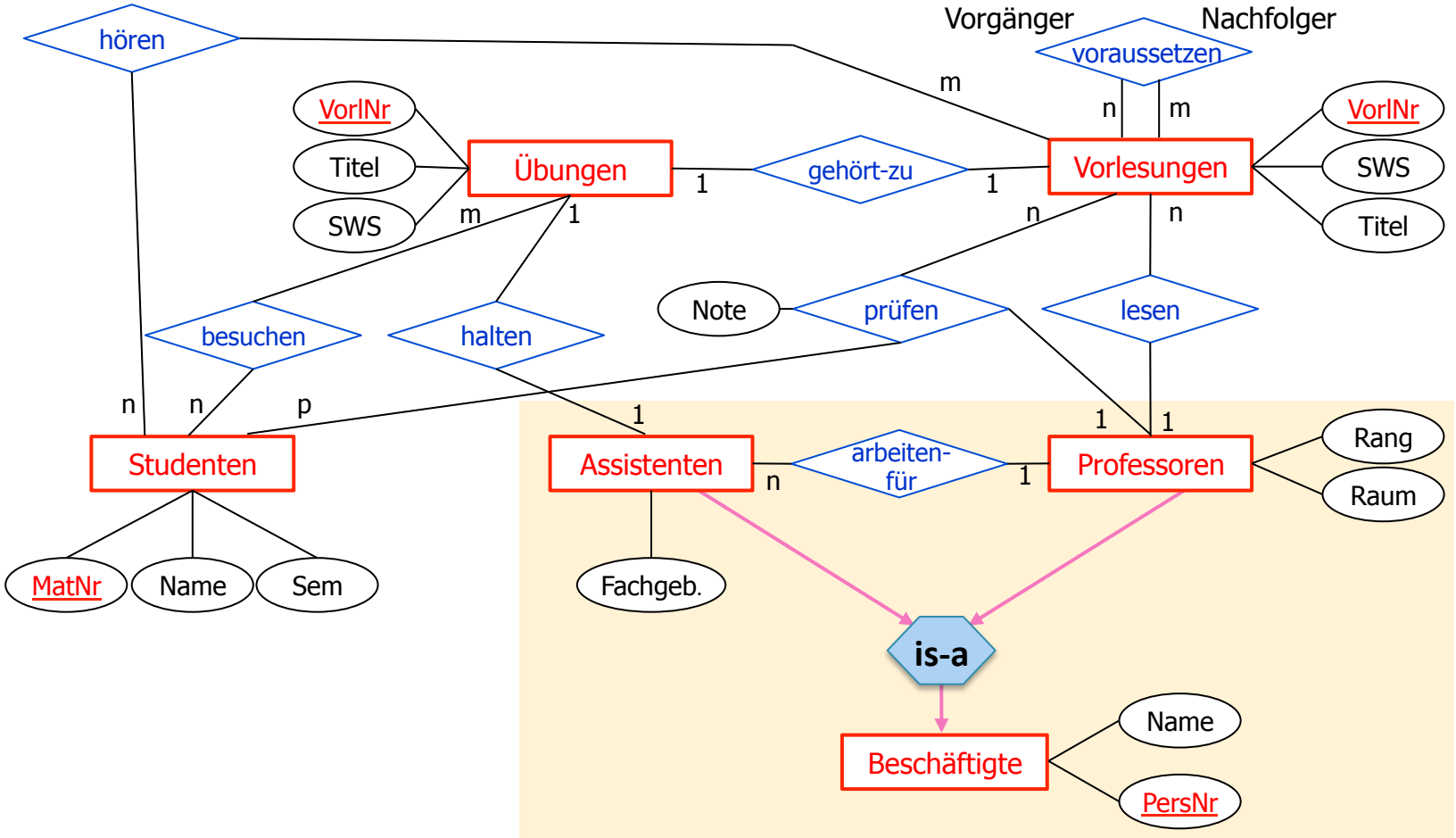
Abstraktionskonzepte in sem. Datenmodellen

siehe auch Elmasri, Navathe: Grundlagen von DBS

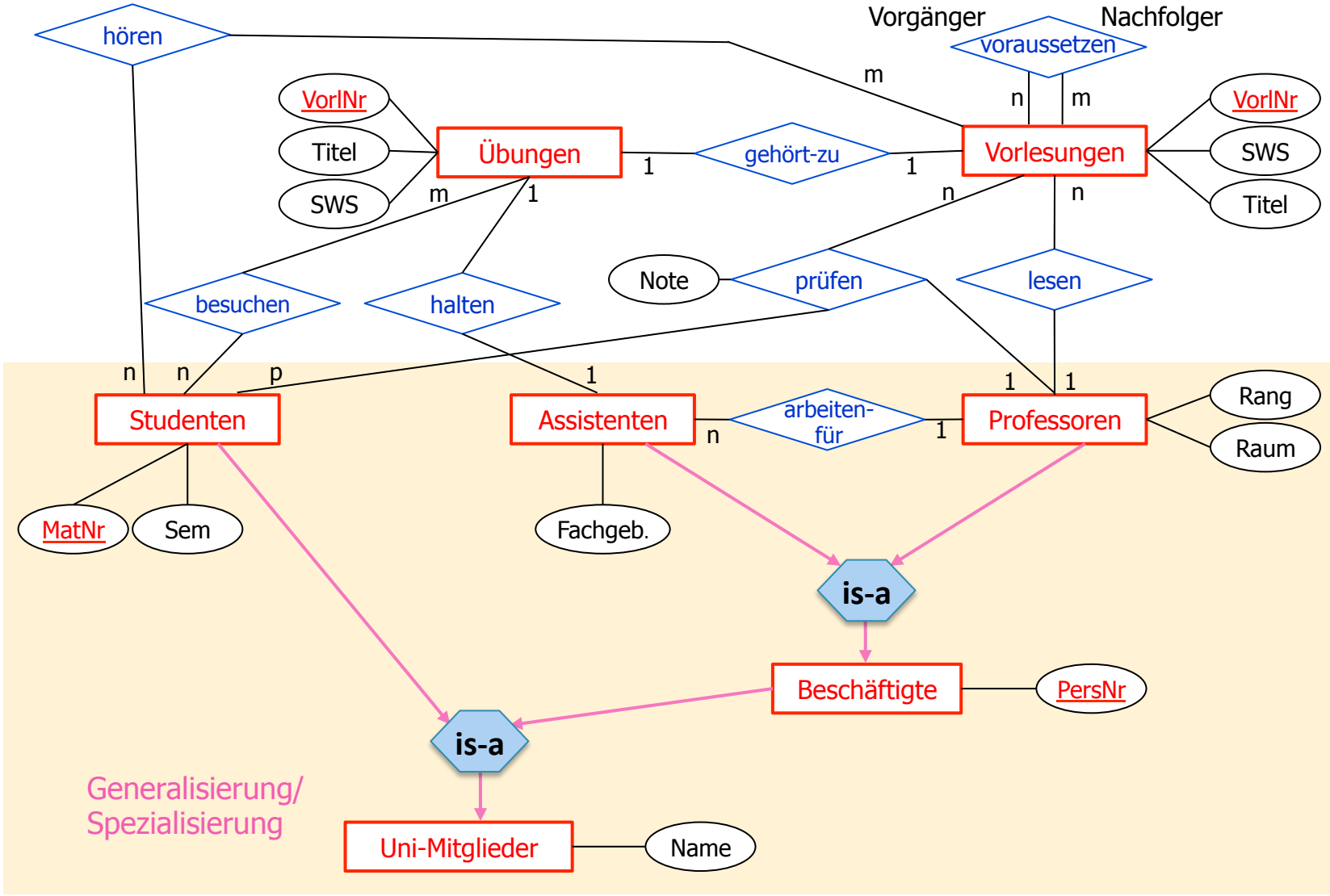
- ✓ Klassifikation/Instantiierung
  - entspricht der Definition von Entity-Typen/Mengen im ERM (Klasse ≈ Typ + Menge)
  - mögliche Erweiterung: explizite Modellierung von Instanzen (hier nicht weiter behandelt)
- ✓ Identifikation
  - Objekte haben immer einen eindeutigen Identifikator
  - im ERM durch eindeutige Namen von Entity-Typen und Definition von Primärschlüsseln
- ➢ Generalisierung/Spezialisierung ("is-a"-Beziehung)
- ➢ Aggregation ("part-of"-Beziehung)
- ▪ (Assoziation)

Motivation

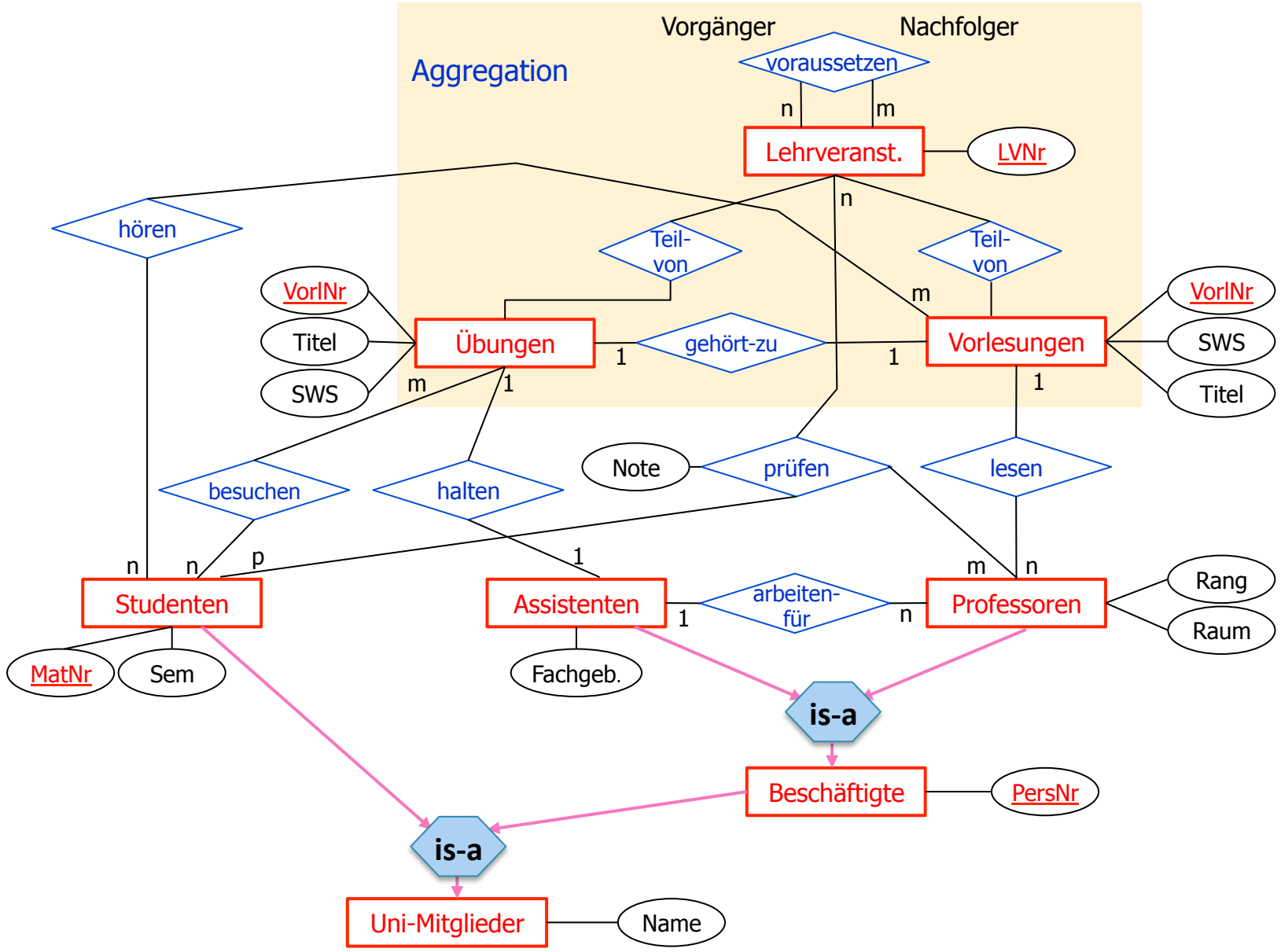
„Alles dreht sich um die genauere Modellierung von Informationsstrukturen“



Motivation (2)

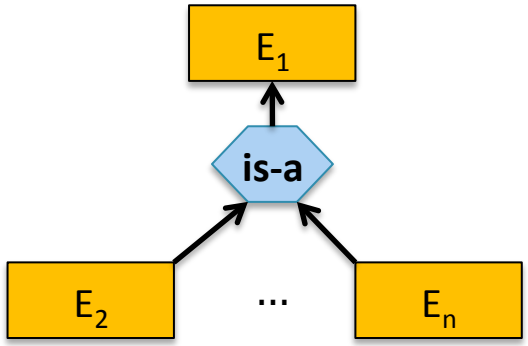


Motivation (3)

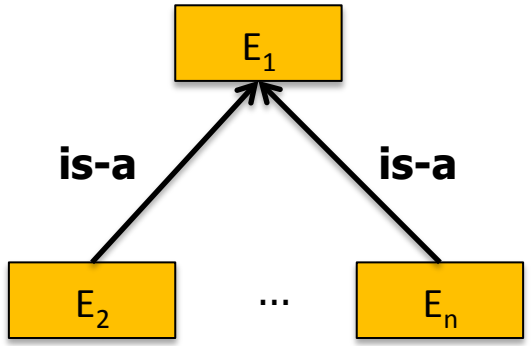


Is-a – Beziehung (Subklasse/Superklasse)

- Notation:



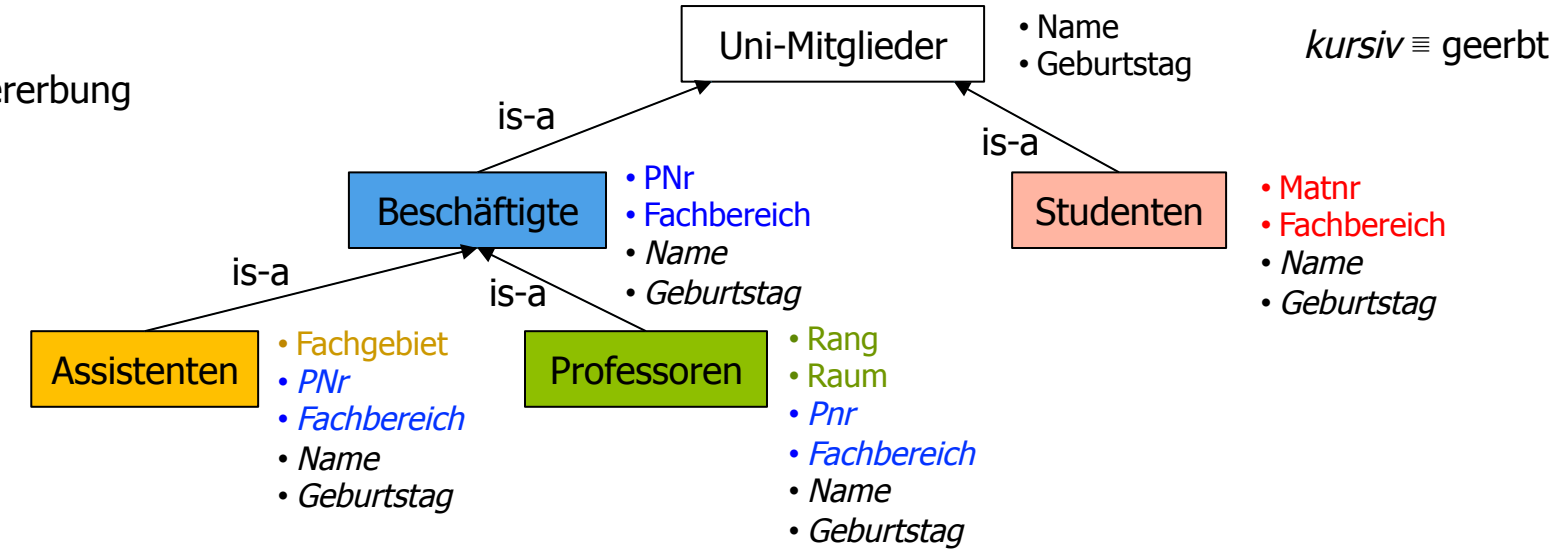
vereinfacht:



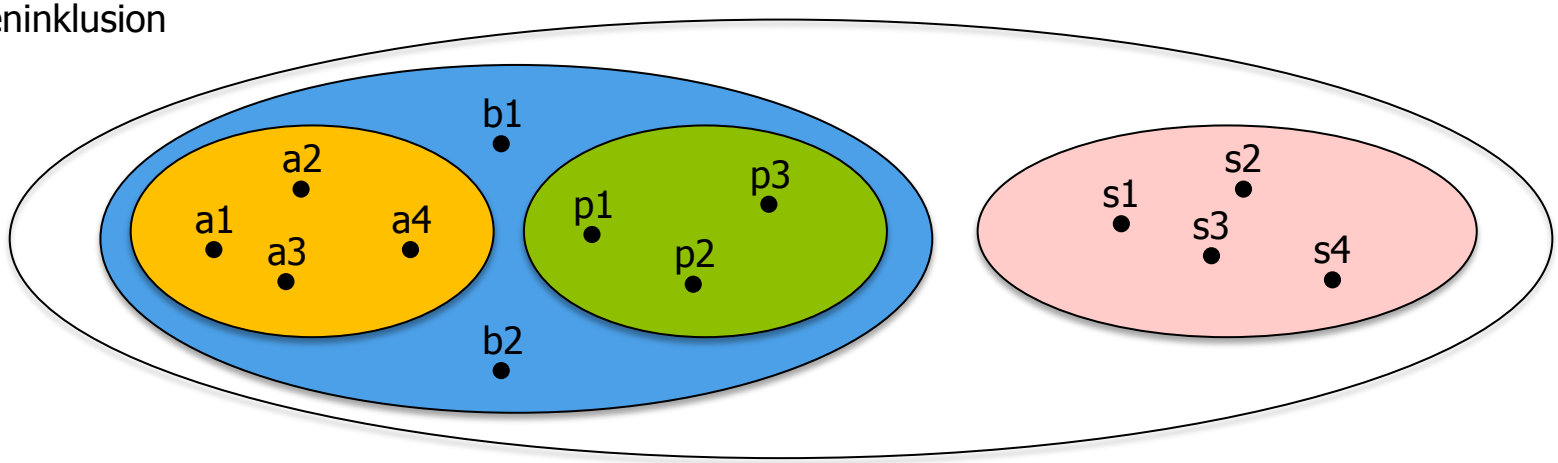
  - E_2 "ist ein" E_1; E_2 ist Subklasse von E_1; E_1 ist Superklasse von E_2
- Bedeutung (sei S "is-a" G)
  - Mengeninklusion: S ist Teilmenge von G ($S \subseteq G$)
  - Typenvererbung
    - Attribute: alle Attribute (inkl. Schlüssel!) von G beschreiben auch S (sind implizit für S definiert – werden von S "geerbt")
  - Relationships: Entities in S können an allen Relationships von G teilnehmen
  - für S können zusätzliche Attribute/Relationships definiert sein

Klassen-Hierarchie (Beispiel)

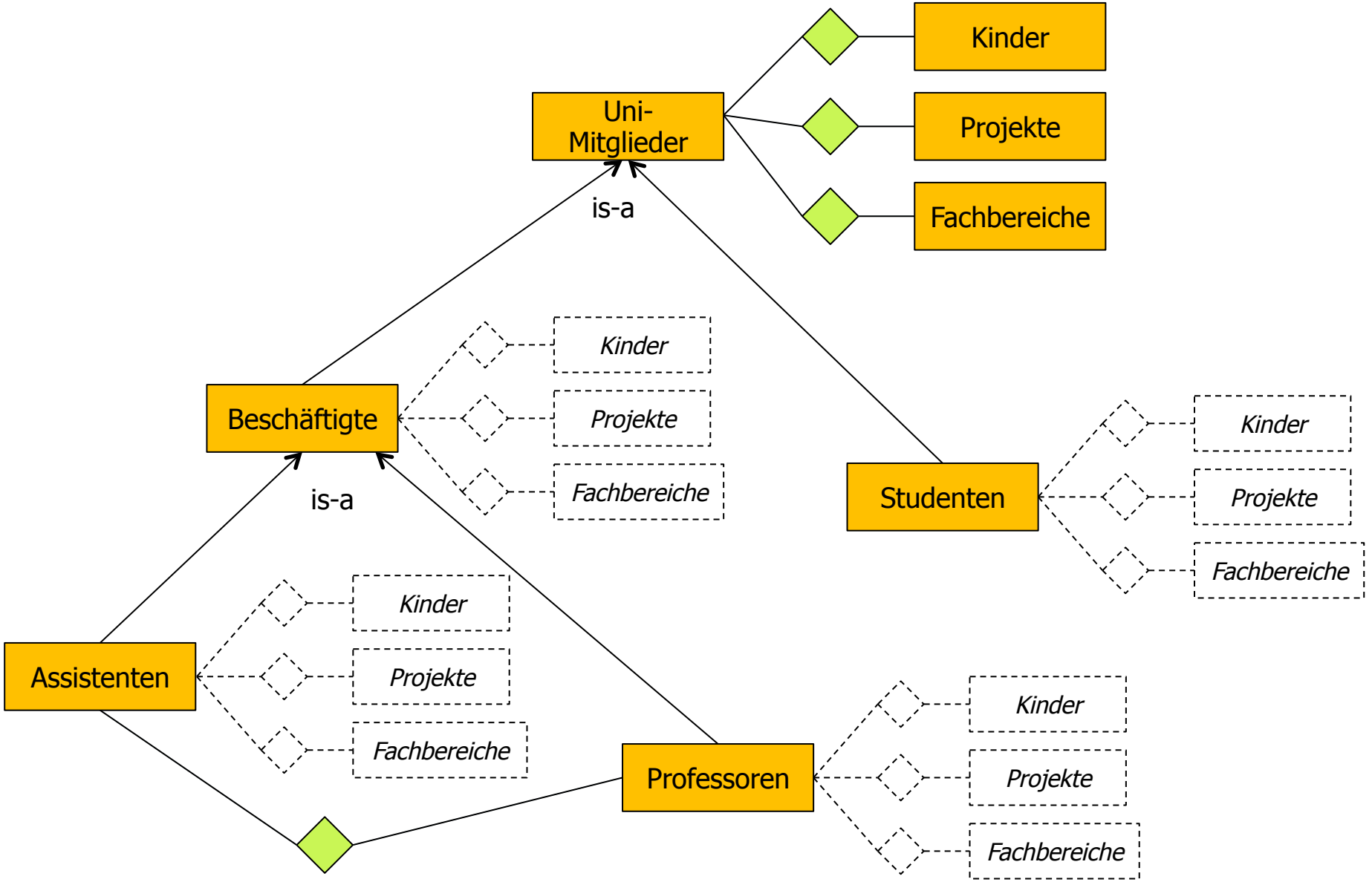
Typenvererbung



Mengeninklusion

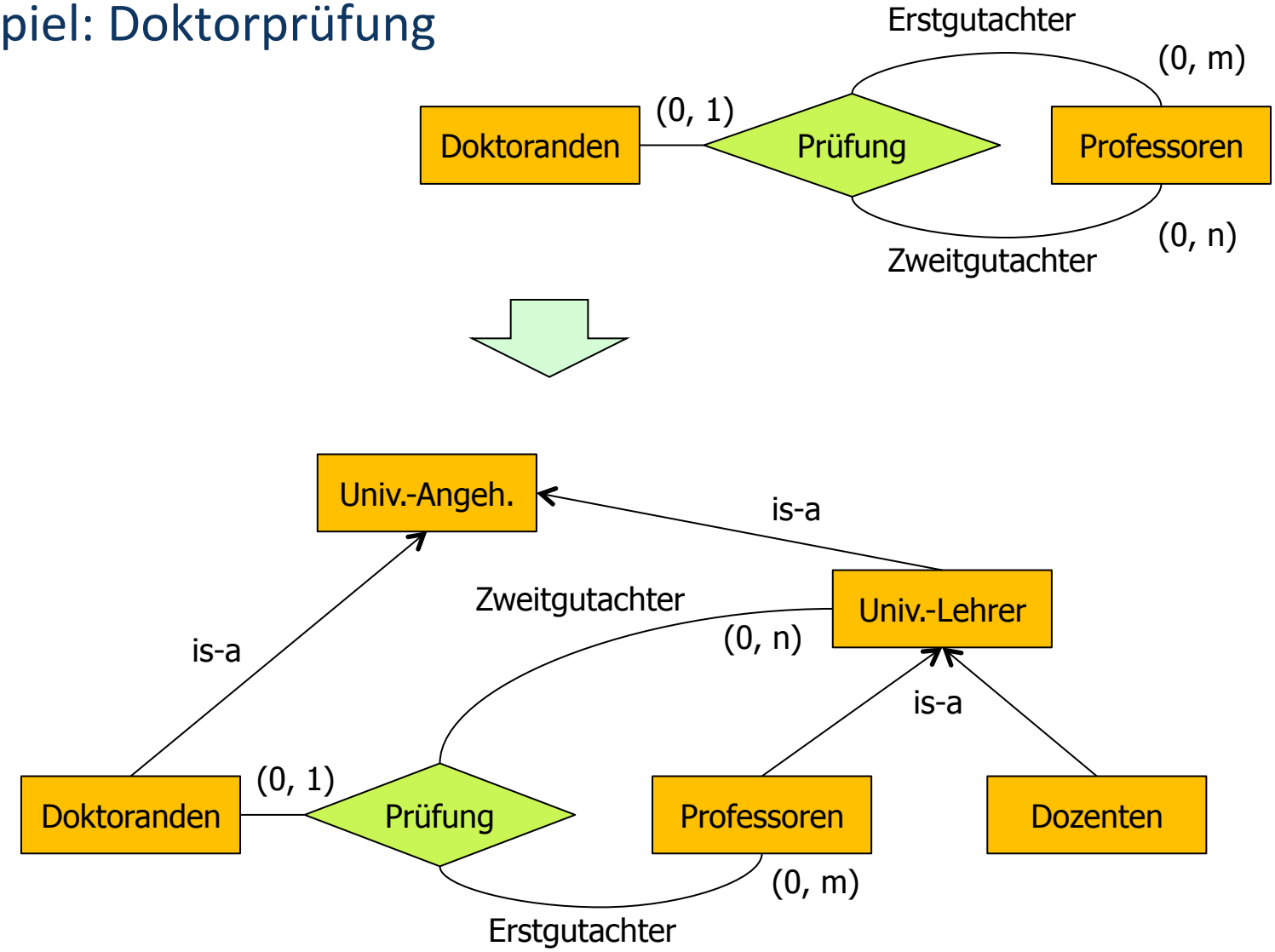


Vererbung von Beziehungen



Verfeinerung von Beziehungen

Beispiel: Doktorprüfung

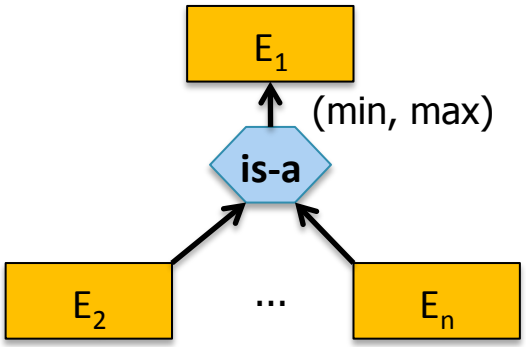


Generalisierung und Spezialisierung

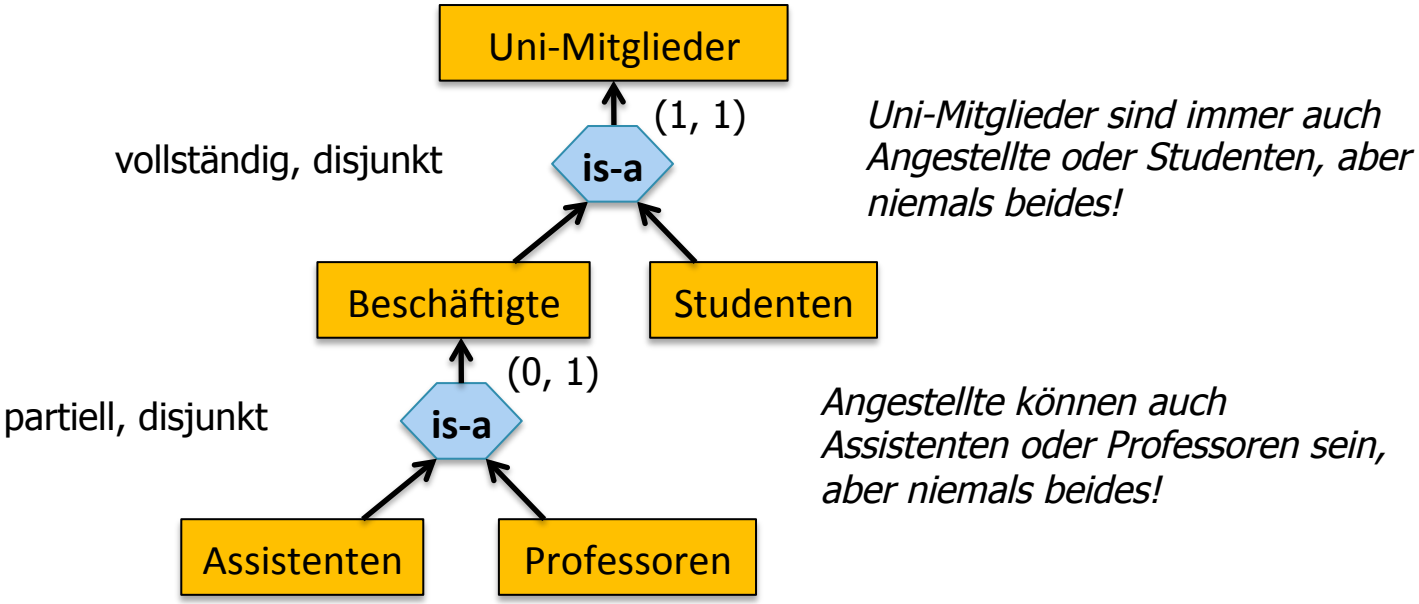
- Alternative Vorgehensweisen bei der Modellierung
- Generalisierungsprozeß
  - Gemeinsame Merkmale von Klassen werden sukzessiv identifiziert und als neue (Super-)Klasse zusammengefasst
  - Bottom-up-Vorgehen
- Spezialisierungsprozeß
  - Klasse wird sukzessiv aufgrund von Merkmalen und Struktureigenschaften durch Definition von Subklassen verfeinert
  - Top-down-Vorgehen (konzeptuelle Verfeinerung)
- Resultierende Klassenhierarchien sollten unabhängig vom Vorgehen strukturell identisch sein
- In der Praxis werden beide Vorgehensweisen kombiniert

Vollständige und disjunkte Spezialisierungen

- S_i "is-a" G ($1 \leq i \leq n$); immer: S_i ist Teilmenge von G ($S_i \subseteq G$)
- Zusätzliche Einschränkungsmöglichkeiten der Subklassenbeziehung
  - disjunkte Spezialisierung
    - S_i sind paarweise disjunkt: $S_i \cap S_j = \varnothing$ für $i \neq j$
    - ein Entity aus G kann zus. in max. einer der Subklassen enthalten sein
  - vollständige Spezialisierung
    - Die Vereinigung aller S_i ergibt G: $G = \cup S_i$ ($1 \leq i \leq n$)
    - ein Entity aus G ist immer auch in einer der Subklassen enthalten
- Im Diagramm: wie Kardinalitätsrestriktionen der Superklasse
  - vollständig: min = 1; disjunkt: max = 1
- Vier mögliche Kombinationen
  - partiell + überlappend: (0, *) - *default*
  - vollständig + disjunkt : (1, 1)
  - partiell + disjunkt: (0, 1)
  - vollständig + überlappend: (1, *)



Vollständige und disjunkte Spezialisierungen (Beispiel)

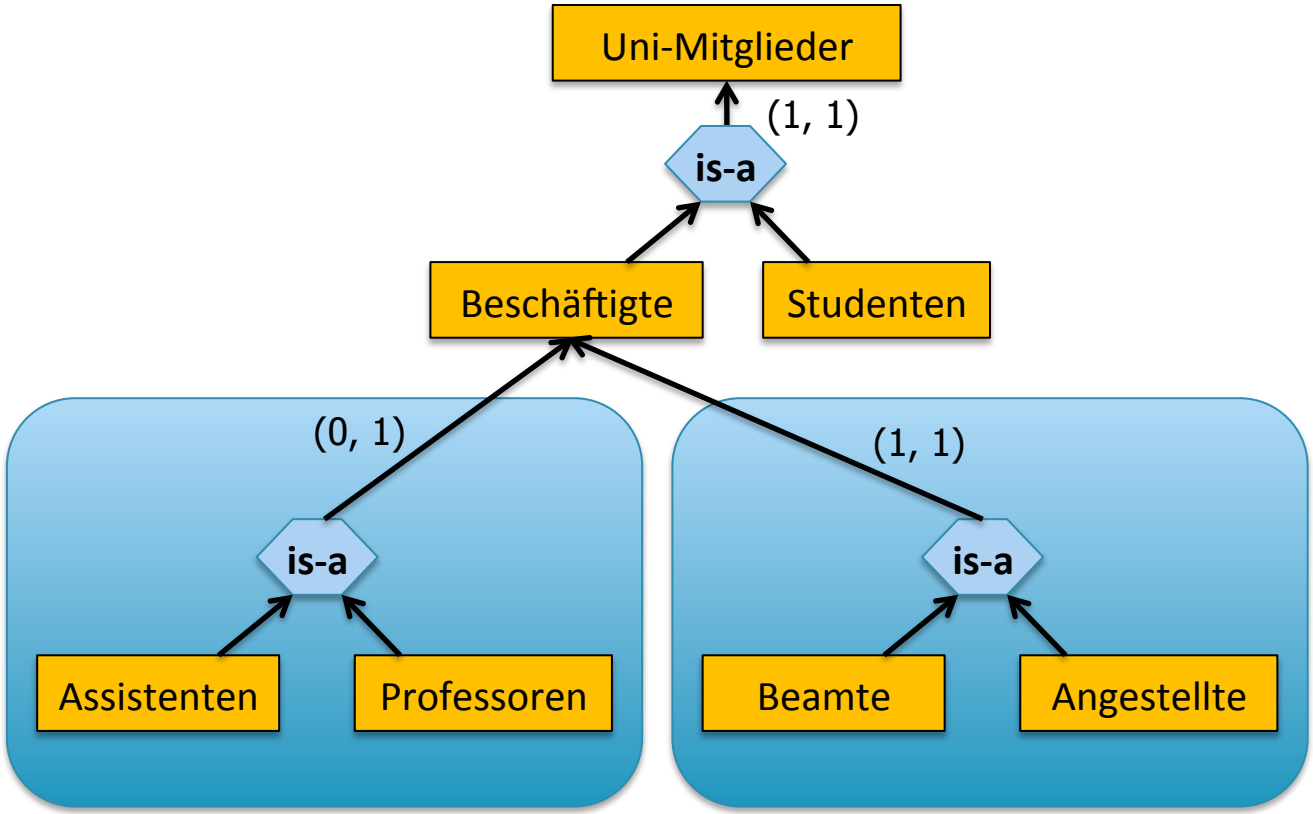


Explizite Kardinalitätsrestriktionen für Subklassen sind nicht sinnvoll! Warum?

Wo müssen Primärschlüssel definiert werden?

Mehrere Spezialisierungen für einen Entity-Typ

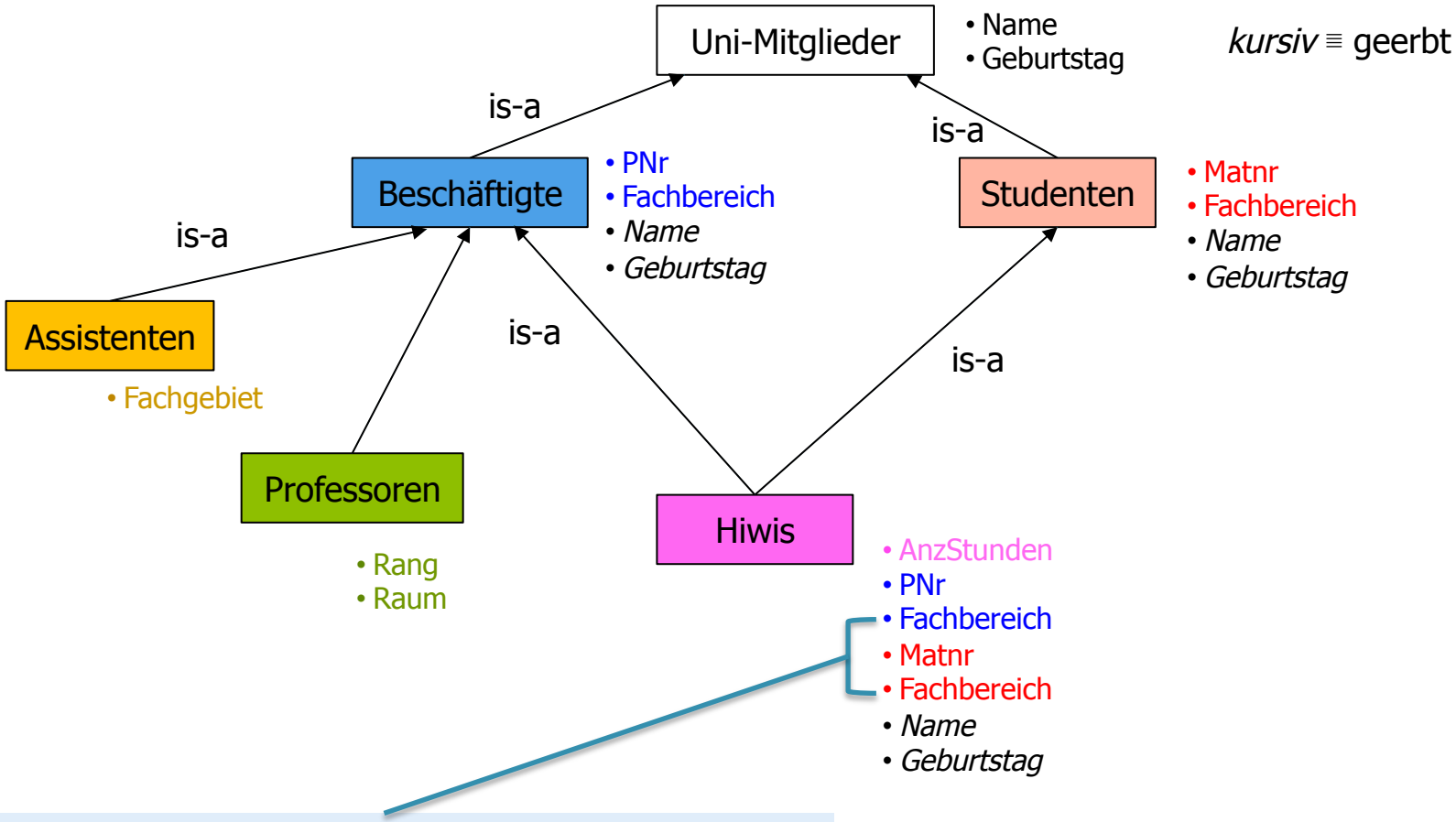
- Mehrere Spezialisierungen nach verschiedenen Kriterien
  - werden durch eigene "is-a"-Boxen dargestellt
  - eigene Angaben zu Vollständigkeit, Disjunktheit möglich



Mehrfach-Vererbung (*multiple inheritance*)

- Bisher: eine Klasse hat höchstens eine Superklasse
  - oft zu restriktiv (z.B., Hiwis sind Studenten und Beschäftigte)
- Verallgemeinerte "is-a"-Beziehung erlaubt mehrere Superklassen
  - Vererbung
    - erfolgt nun von mehreren (Super-)Klassen
    - es kann mehrere "is-a"-Pfade zur gleichen (Super-)Klasse geben
    - Attribute der Klasse werden nur einmal vererbt (also nur eine Kopie des Attributes vorhanden)
  - Vererbungskonflikte
    - können auftreten, wenn in mehreren verschiedenen Superklassen Attribute mit gleichem Namen definiert werden
    - müssen manuell durch Umbenennung von Attributen aufgelöst werden
- Vorsicht: auf Konsistenz mit Disjunktheitsangaben achten!
  - Superklassen müssen überlappen

Mehrfachvererbung (Beispiel)



Vererbungskonflikt
mögliche Umbenennungen:
Fachbereich of Angestellter → Hiwi_im_Fachbereich
Fachbereich of Student → immatrikuliert_im_Fachbereich

Aggregation

- Beziehung mit spezieller zusätzlicher Bedeutung:

  Ein Objekt ist aus **Bestandteilen** zusammengesetzt (Teil-Ganze-Beziehung), z. B.

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

- Entweder exklusiv*: *Auto – Motor*

  kein anderes Objekt darf denselben Bestandteil haben

  oder gemeinsam: *Kante – Endpunkt*

  derselbe Bestandteil wird in zwei oder mehr Objekten verwendet

- Entweder abhängig: *Tisch – Tischplatte*

  Bestandteil kann nicht allein existieren; wird mit dem Objekt gelöscht

  oder unabhängig: *Bild – Farbtabelle*

  Bestandteil kann auch für sich als Objekt existieren

- Objekte mit exklusiven und/oder abhängigen Objekten heißen

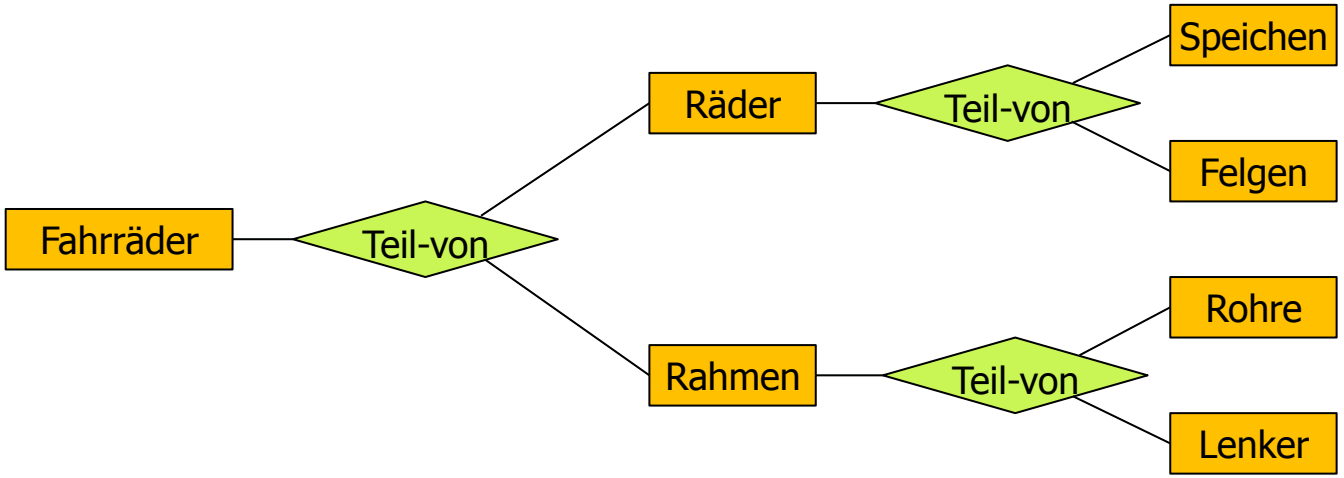
  zusammengesetzte Objekte (composite objects, komplexe Objekte)

  oder Aggregate

(*) Die exklusive Aggregation wird in UML als Komposition bezeichnet, die gemeinsame als „Aggregation".

Aggregation im ER-Modell

- Attribute sind Bestandteile ihrer Entities
  - immer exklusiv, abhängig
- Beliebige Relationship zur Modellierung von Aggregation
  - Kardinalitätsrestriktionen zur Modellierung von
    - Exklusivität: Bestandteil hat max=1
    - Abhängigkeit: Bestandteil hat min=1
- "Teil-von"-Relationship (oder "part-of")
  - erfordert in gleicher Weise Angabe von Kardinalitäten



Mögliche Erweiterungen

- Verschiedene Vorschläge zur Erweiterung der "*is-a*"-Beziehung
  - Definition/Vererbung/Verfeinerung von
    - Funktionen/Methoden
    - Wertebereichen und Constraints
  - Bedingungen zur automatisch Ableitung von Subklassen
    - prädikatdefinierte Subklassen
    - attributdefinierte Subklassen
- Erweiterung der Aggregation (implizierte Prädikate)
- Assoziationskonzept
  - Element-/Teilmengenbeziehung über beliebige Klassen
  - Mitgliedschaftsbedingungen und Ableitungsregeln
- Alternative Notationen für die in diesem Kapitel eingeführten Konzepte

➔ Werden an dieser Stelle nicht weiter vertieft

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

Zusammenfassung (2)

- Einführung weiterer Modellierungskonzepte
  - Verfeinerung von Beziehungen durch Kardinalitätsrestriktionen und vor allem Abstraktionskonzepte
  - Das erweiterte ERM ist sehr mächtig und umfasst viele bekannte Modellierungskonzepte
- Abstraktionskonzepte erlauben die genauere Modellierung und Organisation von Informationen
  - Generalisierung/Spezialisierung mit Vererbung, Mengeninklusion (Disjunktheit, Vollständigkeit)
  - Aggregation (Teile-Ganzes-Beziehung)
- Integritätsbedingungen
  - (E)ER-Modell ermöglicht Definition von strukturellen Bedingungen
  - beliebige logische Bedingungen nicht Teil des (formalen) ER-Modells
    - höchstens über Kommentare/Annotationen spezifizierbar