

# 2. Informationsmodellierung mit Entity-Relationship-Modell und UML

- Einführung Modellierung / Abstraktionskonzepte
- Entity-Relationship-Modell
  - Entity-Mengen
  - Attribute und Wertebereiche
  - Primärschlüssel
  - Relationship-Mengen
  - Klassifikation der Beziehungstypen (1:1, n:1, 1:n, n:m)
  - Schwache Entity-Mengen
- Modellierung mit UML (Klassendiagramme)
  - Kardinalitätsrestriktionen (Multiplizitäten)
  - Generalisierung / Spezialisierung
  - Aggregation



## Lernziele Kapitel 2

- Kenntnis der Vorgehensweise beim DB-Entwurf
- Grundkonzepte des ER-Modells sowie von UML-Klassendiagrammen
- Kenntnis der Abstraktionskonzepte, insbesondere von Generalisierung und Aggregation
- Fähigkeit zur praktischen Anwendung der Konzepte
  - Erstellung von ER-Modellen und -Diagrammen bzw. UML-Modellen für gegebene Anwendungsszenarien
  - Festlegung der Primärschlüssel, Beziehungstypen, Kardinalitäten, Existenzabhängigkeiten etc.
  - Interpretation gegebener ER- bzw. UML-Modelle



# Informations- und Datenmodellierung (DB-Entwurf)

## Ziele:

- modellhafte Abbildung eines anwendungsorientierten Ausschnitts der realen Welt (Miniwelt)
- Entwurf der logischen DB-Struktur (DB-Entwurf)

## Nebenbedingungen:

- Vollständigkeit
- Korrektheit
- Minimalität
- Lesbarkeit, Modifizierbarkeit

## Schrittweise Ableitung: (Verschiedene Sichten)

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

## Datenmodellierung

**Ausschnitt der  
Realen Miniwelt**

Manuelle/intellektuelle Modellierung

**Konzeptuelles Schema  
(ER-Schema bzw. UML-Klassendiagramm)**

Halbautomatische  
Transformation

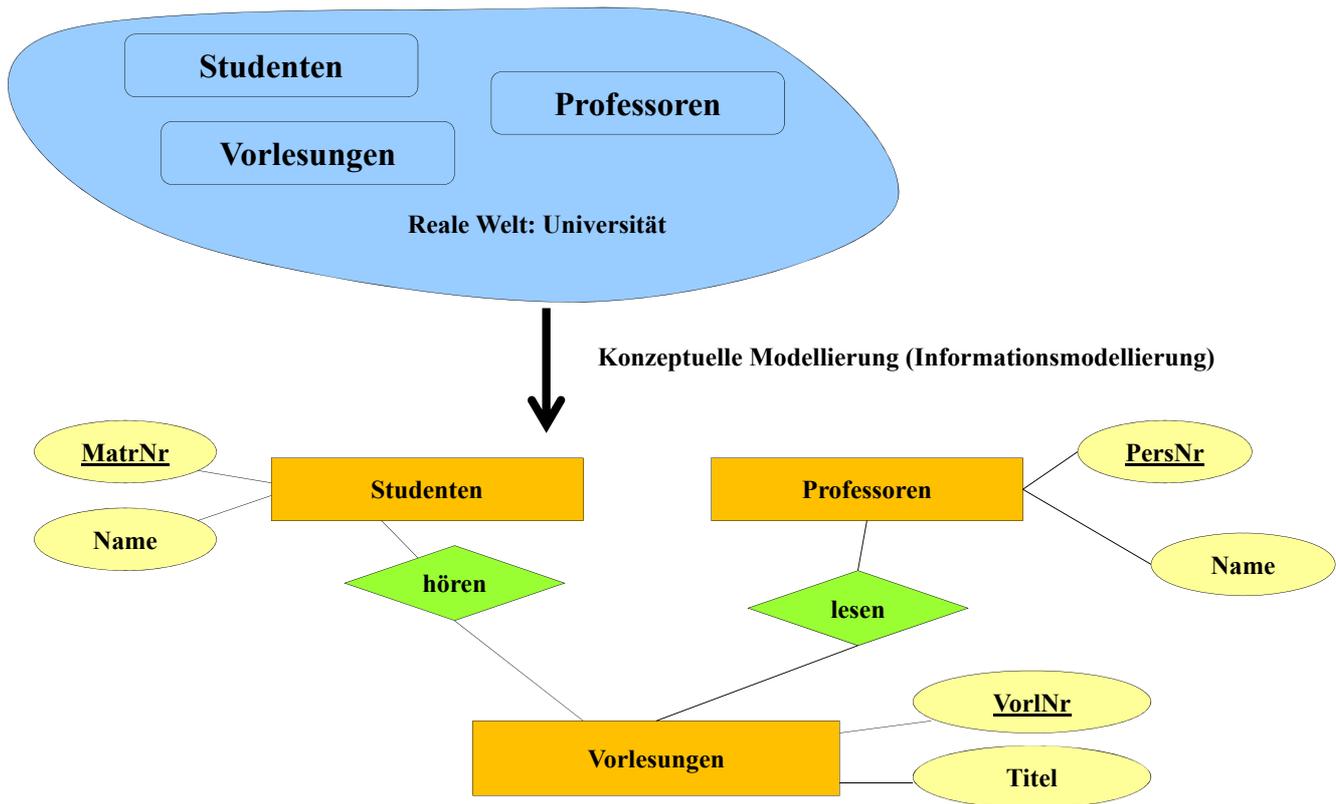
**Relationales  
Schema**

**XML  
Schema**

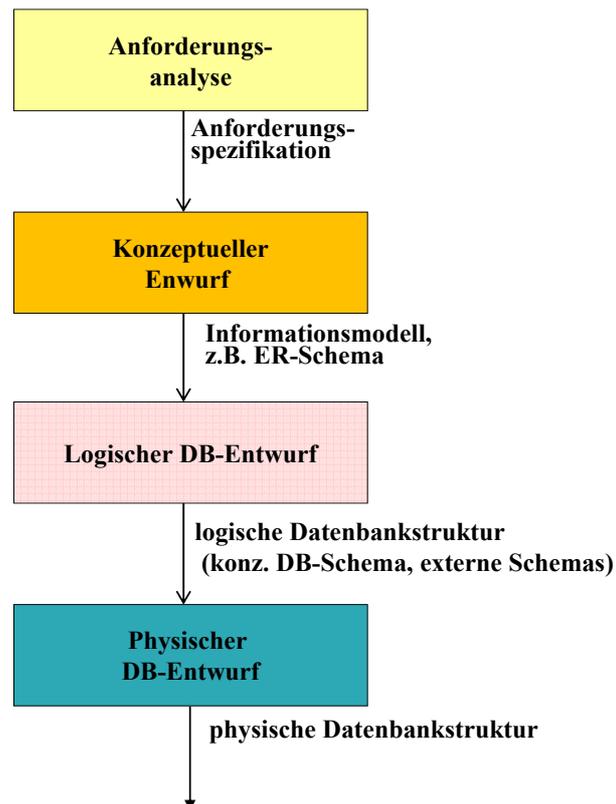
**Netzwerk  
Schema**

**Objektorientiertes  
Schema**

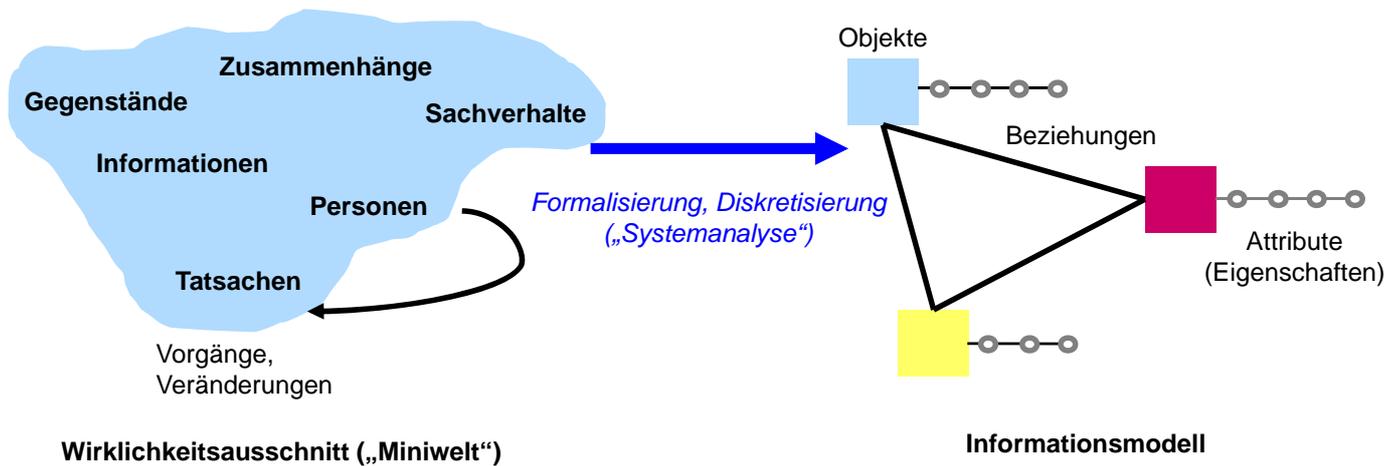
# Modellierungsbeispiel



# Phasen des DB-Entwurfs



# Informationsmodellierung



- Darstellungselemente + Regeln:
  - Objekte (Entities) und Beziehungen (Relationships)
  - Klassen von Objekten / Beziehungen
  - Eigenschaften (Attribute)
- Informationen über Objekte und Beziehungen nur wenn:
  - relevant
  - unterscheidbar und identifizierbar, selektiv beschreibbar

## Abstraktionskonzepte

- Informations- und Datenmodelle basieren auf drei grundlegenden Abstraktionskonzepten
- **Klassifikation**: fasst Objekte (Entities, Instanzen) mit gemeinsamen Eigenschaften zu einem neuen (Mengen-) Objekt (Entity-Menge, Klasse, Objekttyp) zusammen.
  - Instanzen/Objekten einer Klasse unterliegen gleicher Struktur (Attribute), gleichen Integritätsbedingungen, gleichen Operationen
  - mathematisch: Mengengebilde
- **Aggregation**: Zusammenfassung potentiell unterschiedlicher Teilobjekte (Komponenten) zu neuem Objekt
  - mathematisch: Bildung von kartesischen Produkten
- **Generalisierung / Spezialisierung**: Teilmengenbeziehungen zwischen Elementen verschiedener Klassen
  - mathematisch: Bildung von Potenzmengen (bzw. Teilmengen)
  - wesentlich: Vererbung von Eigenschaften an Teilmengen

# Entity-Relationship-Modell

- entwickelt von P. P. Chen (ACM Transactions on Database Systems 1976)
- Konzepte:
  - Entity-Mengen
  - Beziehungsmengen (Relationship-Mengen)
  - Attribute
  - Wertebereiche
  - Primärschlüssel
- unterstützt Abstraktionskonzepte der Klassifikation und Aggregation
- zahlreiche Erweiterungsvorschläge
- graphische Darstellung durch Diagramme
- weite Verbreitung über DB-Entwurf hinaus: Systemanalyse, Unternehmensmodellierung



## Entity-Mengen

- *Entity* (Entität, Gegenstand): repräsentiert abstraktes oder physisches Objekt der realen Welt
- Gleichartige Entities (d. h. Entities mit gemeinsamen Eigenschaften) werden zu Entity-Mengen (Gegenstandstypen, Objekttypen) zusammengefasst (Klassifikation)
  - => Entities sind Elemente einer (homogenen) Menge:  $e \in E$ 
    - z. B. Personen, Projekte ...
    - Bücher, Autoren ...
    - Kunden, Vertreter, Wein, Behälter
- DB enthält endlich viele Entity-Mengen:  $E_1, E_2, \dots, E_n$ 
  - nicht notwendigerweise disjunkt

- Symbol für Entity-Menge E:



# Attribute und Wertebereiche

## ■ Attribute und Attributwerte:

- Eigenschaften von Entity-Mengen werden durch Attribute bestimmt
- Eigenschaften einzelner Entities sind durch Attributwerte festgelegt
- **Nullwert**: spezieller Attributwert, dessen Wert unbekannt oder nicht möglich ist (z. B. FaxNr)

## ■ Jedem Attribut ist ein Wertebereich (Domain) zugeordnet, der festlegt, welche Attributwerte zulässig sind (Integritätsbedingung !)

$$E (A_1: D_1, A_2: D_2, \dots A_n: D_n)$$

## ■ Attributsymbol in ER-Diagrammen:



# Attributarten

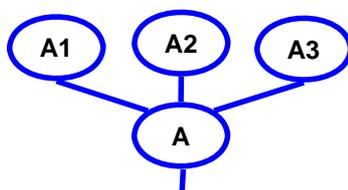
## ■ *einfache* vs. *zusammengesetzte* Attribute

- Beispiele: NAME [Vorname: char (30), Nachname: char (30) ]  
ANSCHRIFT [Strasse: char (30), Ort: char (30), PLZ: char (5) ]
- Domain für zusammengesetztes Attribut A [A<sub>1</sub>, A<sub>2</sub>, ... A<sub>k</sub>]:  
 $W (A_1) \times W (A_2) \times \dots \times W (A_k)$

## ■ *einwertige* vs. *mehrwertige* Attribute

- Beispiele: AUTOFARBE: {char (20)}  
KINDER: {[Name: char (30), Alter: int]}

## ■ Symbole:



## Schlüsselkandidat

- Schlüsselkandidat oder kurz Schlüssel (key)
  - einwertiges Attribut oder Attributkombination, die jedes Entity einer Entity-Menge eindeutig identifiziert
  - keine Nullwerte!
  - mehrere Schlüsselkandidaten pro Tabelle möglich

### ■ Definition Schlüsselkandidat

$A = \{A_1, A_2, \dots, A_m\}$  sei Menge der Attribute zu Entity-Menge  $E$

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

1.  $\forall e_i, e_j \in E$  mit  $e_i \neq e_j \rightarrow K(e_i) \neq K(e_j)$ ;
2. **K minimal**



## Primärschlüssel

- Primärschlüssel = Schlüsselkandidat
  - ggf. künstlich zu erzeugen (lfd. Nr.)
  - ggf. unter mehreren Kandidaten einen auszuwählen

Beispiel:

Prof (Zi-Nr, Sekr-TelNr, Vorname, Name, PNR)

- Primärschlüsselattribute werden durch Unterstreichung gekennzeichnet

Attr.Name



## Relationships

- Relationship-Menge: Zusammenfassung gleichartiger Beziehungen (Relationships) zwischen Entities, die jeweils gleichen Entity-Mengen angehören
- Beispiel: Beziehungen *Vorlesungsbesuch* zwischen *Student* und *Vorlesung*



## Relationships (2)

- Relationship-Menge  $R$   
entspricht mathematischer Relation zwischen  $n$  Entity-Mengen  $E_i$   
 $R \subseteq E_1 \times E_2 \times \dots \times E_n$ ,  
d. h.  $R = \{r = [e_1, e_2, \dots, e_n] \mid e_1 \in E_1, \dots, e_n \in E_n\}$   
gewöhnlich:  $n=2$  oder  $n=3$

- Symbol:



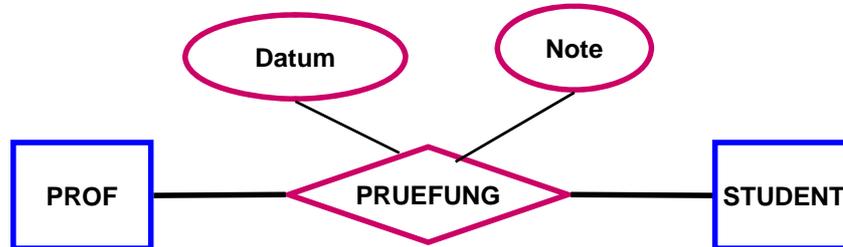
## Relationships (3)

- Relationship-Mengen können auch Attribute besitzen

$$R \subseteq E_1 \times E_2 \times \dots \times E_n \times W(A_1) \times \dots \times W(A_m)$$

$$\text{d. h. } R = \{ r = [e_1, e_2, \dots, e_n, a_1, a_2, \dots, a_m] \mid e_i \in E_i, a_j \in W(A_j) \}$$

- Beispiel



PROF (Pnr, Pname, Fach)

STUDENT (Matnr, Sname, Immdatum)

PRUEFUNG ( PROF, STUDENT, Datum, Note)

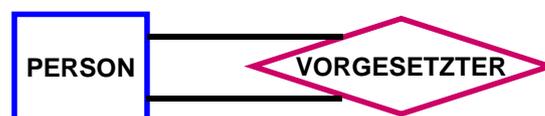
- Entities  $e_i$  können durch ihre Primärschlüssel repräsentiert werden

## Relationships (4)

- keine Disjunktheit der beteiligten Entity-Mengen gefordert (rekursive Beziehungen)

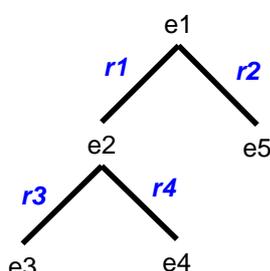
$$\text{VERHEIRATET} \subseteq \text{PERSON} \times \text{PERSON}$$

$$\text{VORGESETZTER} \subseteq \text{PERSON} \times \text{PERSON}$$



- Einführung von Rollennamen möglich (Reihenfolge !)

VORGESETZTER (Chef: PERSON, Mitarbeiter: PERSON)

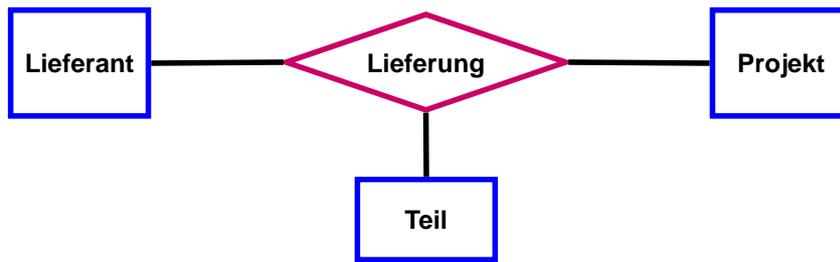


e1 ●  
e2 ●  
e3 ●  
e4 ●  
e5 ●

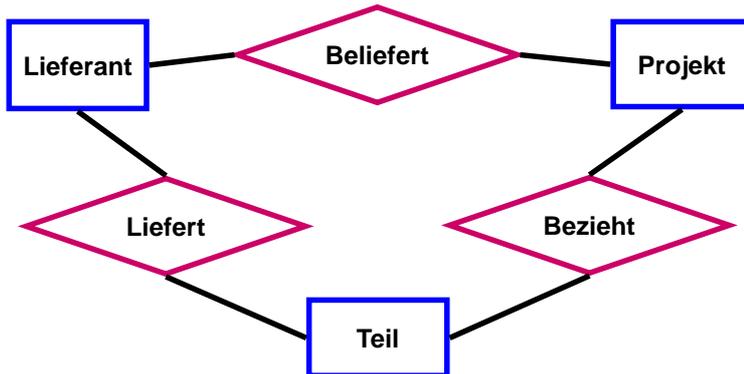
r1 ●  
r2 ●  
r3 ●  
r4 ●

## Relationships (5)

- Beispiel einer 3-stelligen Relationship-Menge



- nicht gleichwertig mit drei 2-stelligen (binären) Relationship-Mengen!



Beispiel:

L1 liefert T1,  
L1 beliefert P1,  
P1 bezieht T1

Impliziert nicht notwendigerweise

Lieferung (L1, P1, T1)

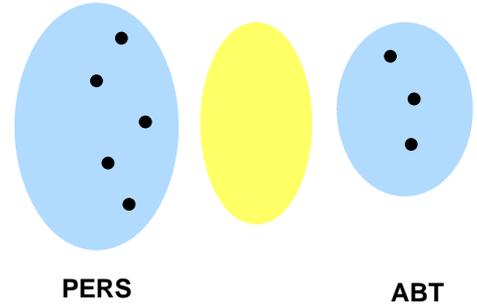
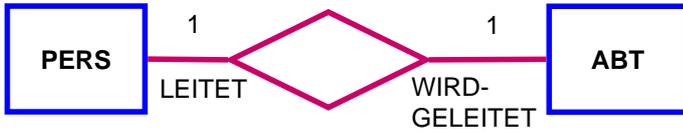
## Kardinalität von Beziehungen

- Festlegung wichtiger struktureller Integritätsbedingungen
- Unterschiedliche Abbildungstypen für binäre Beziehung zwischen Entity-Mengen  $E_i$  und  $E_j$ 
  - 1:1 eineindeutige Funktion (injektive Abbildung)
  - n:1 mathematische Funktion (funktionale Abbildung)
  - 1:n invers funktionale Abbildung
  - n:m mathematische Relation (komplexe Abbildung)
- Abbildungstypen implizieren nicht, dass für jedes  $e \in E_i$  auch tatsächlich ein  $e' \in E_j$  existiert!
  - n:1- sowie 1:1-Beziehungen repräsentieren somit i.a. nur partielle Funktionen
- Präzisierung der Kardinalitätsrestriktionen durch *Min-Max-Notation* bzw. Multiplizitäten in UML

# 1:1-Beziehungen

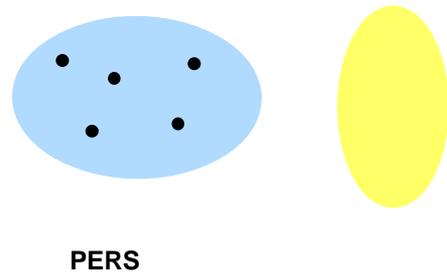
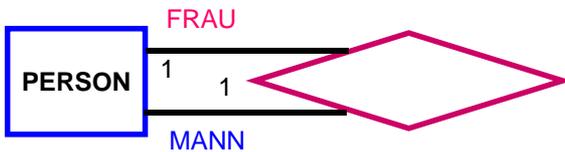
## 1:1-Beziehung zwischen unterschiedlichen Entity-Mengen

1:1 LEITET/WIRD-GELEITET: PERS ↔ ABT



## rekursive 1:1-Beziehung

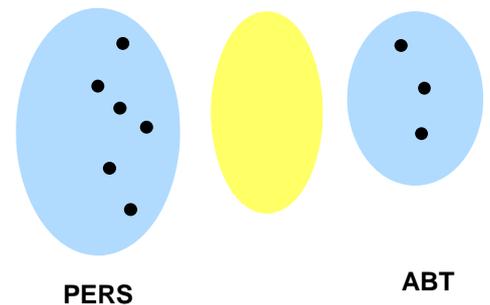
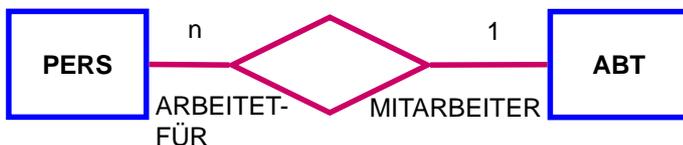
1:1 VERHEIRATET: PERS ↔ PERS



# n:1-Beziehung

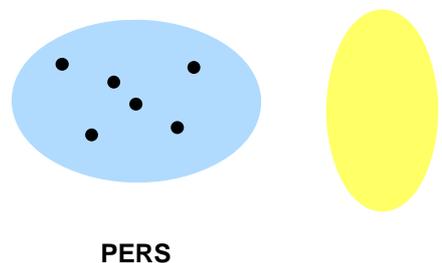
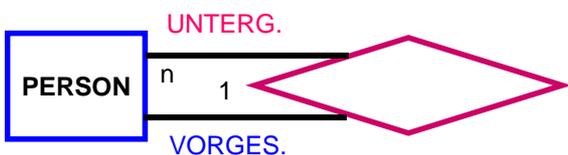
## n:1-Beziehung zwischen unterschiedlichen Entity-Mengen

n:1 ARBEITET-FÜR/MITARBEITER: PERS -> ABT



## rekursive n:1-Beziehung

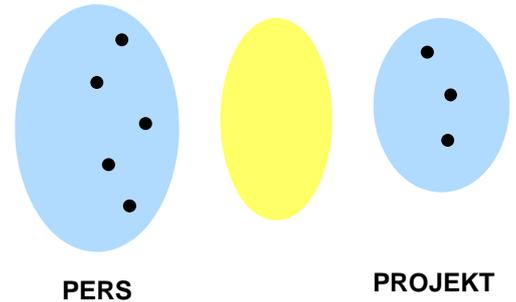
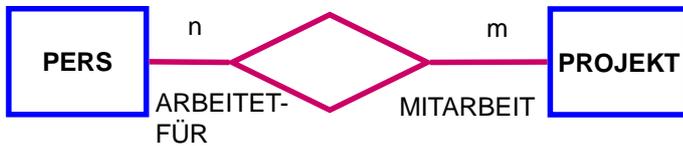
n:1 UNTERGEB./VORGESETZTER\_VON: PERS -> PERS



# n:m-Beziehung

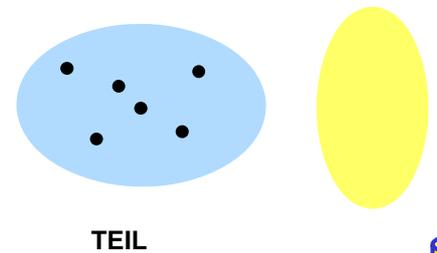
## ■ n:m-Beziehung zwischen unterschiedlichen Entity-Mengen

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

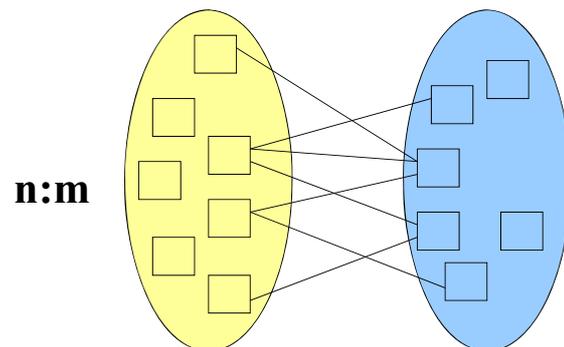
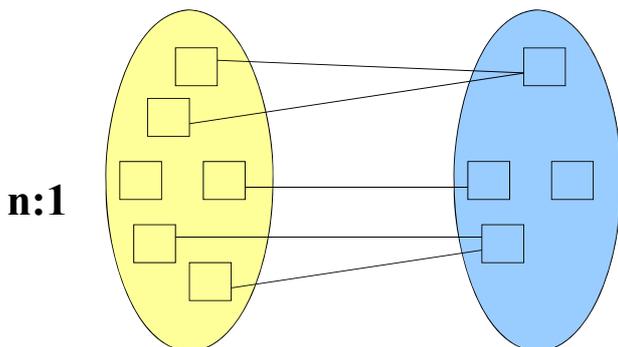
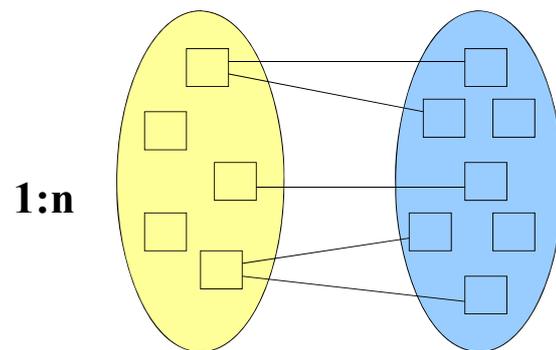
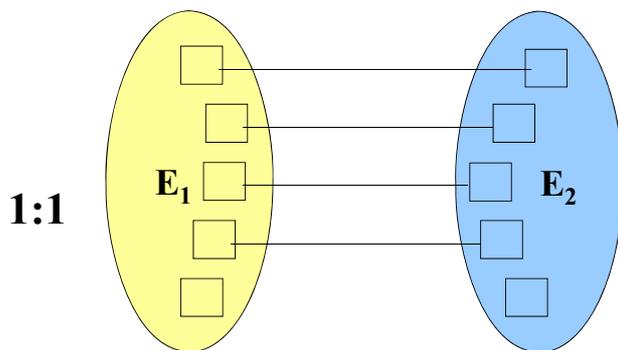
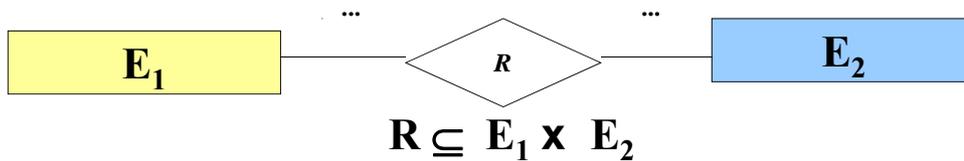


## ■ rekursive n:m-Beziehung

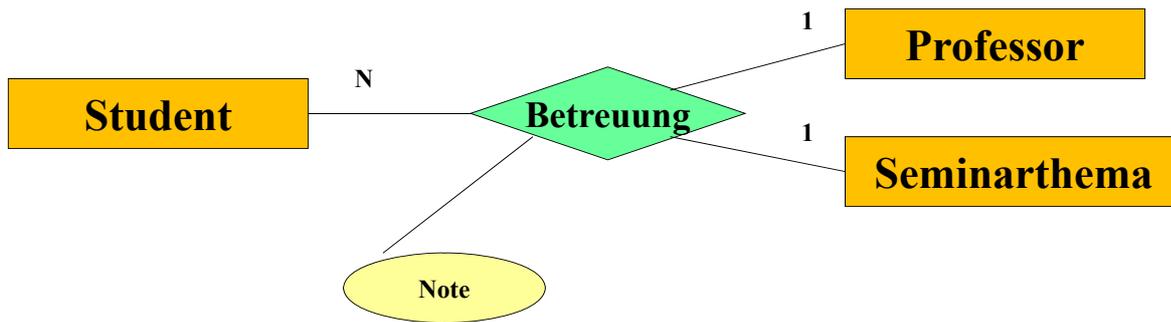
n:m SETZT\_SICH\_ZUSAMMEN\_AUS/GEHT\_EIN\_IN: TEIL ---- TEIL



# Zusammenfassender Überblick



# Kardinalitäten bei mehrstelligen Beziehungen



Betreuung: Professor  $\times$  Student  $\rightarrow$  Seminarthema

Betreuung: Seminarthema  $\times$  Student  $\rightarrow$  Professor

## Schwache Entity-Mengen (weak entities)

- Entity-Menge mit Existenzabhängigkeit zu anderer Entity-Menge
  - kein eigener Schlüsselkandidat, sondern Identifikation über Beziehung zur übergeordneten Entity-Menge
  - Bsp.: Entity-Menge *Raum* (*Nummer*, *Größe*) abhängig von *Gebäude*
- Konsequenzen
  - i.a. n:1 bzw. 1:1-Beziehung zwischen schwacher Entity-Menge und Vater-Entity-Menge
  - jedes schwache Entity muss in Relationship-Menge mit Vater-Entity-Menge vertreten sein (obligatorische Beziehungsteilnahme)
  - Primärschlüssel ist zumindest teilweise von Vater-Entity-Menge abgeleitet
- ER-Symbole:



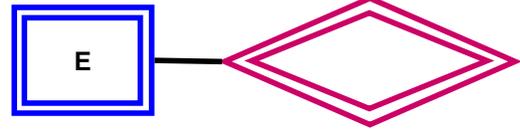
# Überblick über ER-Diagrammsymbole



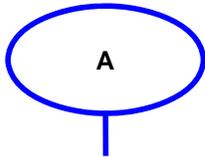
Entity-Menge



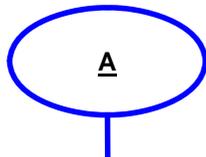
Relationship-Menge



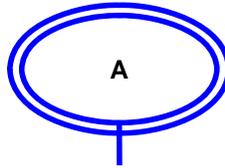
schwache Entity-Menge



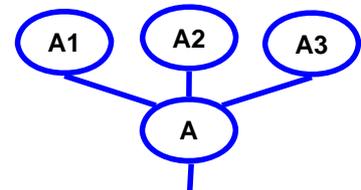
Attribut



Schlüsselattribut



mehrwertiges Attribut



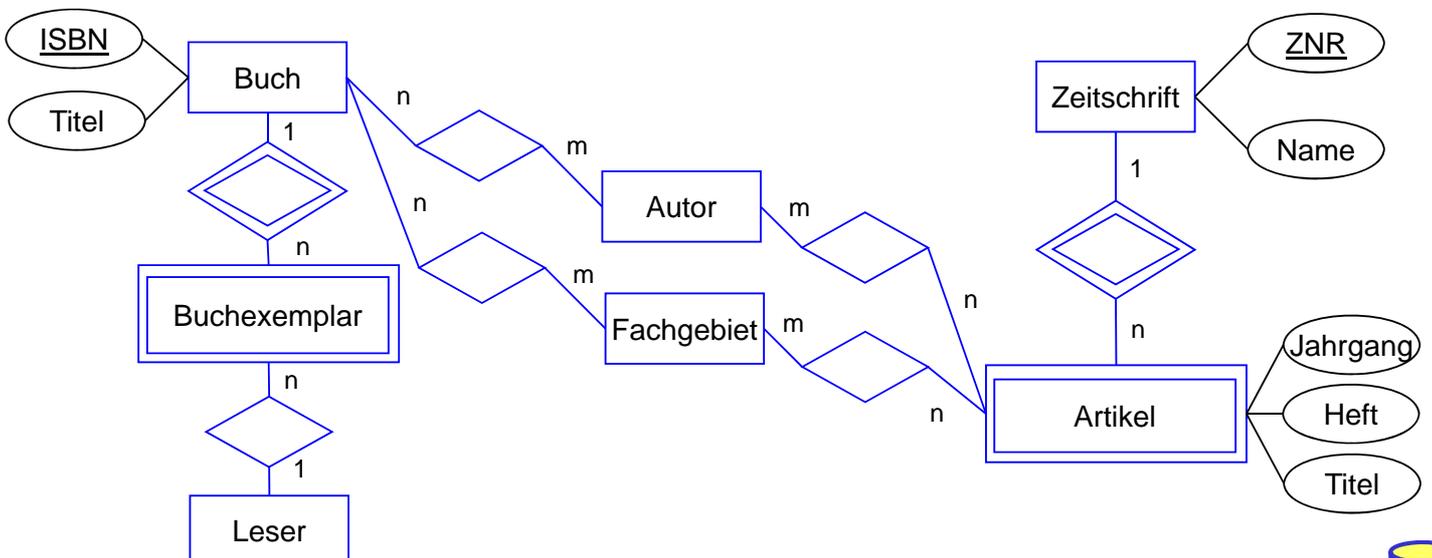
zusammengesetztes Attribut

Beziehungstypen  
(Kardinalitätsangaben)



## ERM: Anwendungsbeispiel

Eine **Bibliothek** besteht aus Büchern und Zeitschriften. Jedes Buch kann ggf. mehrere Autoren haben und ist eindeutig durch seine ISBN gekennzeichnet. Die Bibliothek besitzt teilweise mehrere Exemplare eines Buches. Zeitschriften dagegen sind jeweils nur einmal vorhanden. Sie erscheinen in einzelnen Heften und werden jahrgangswise gebunden. Die in Zeitschriften publizierten Artikel sind ebenso wie Bücher einem oder mehreren Fachgebieten (z. B. Betriebssysteme, Datenbanksysteme, Programmiersprachen) zugeordnet. Ausgeliehen werden können nur Bücher (keine Zeitschriften).



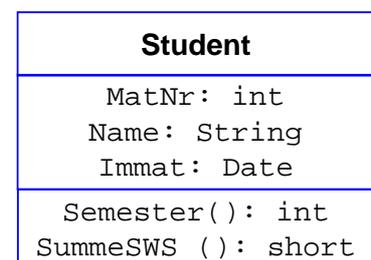
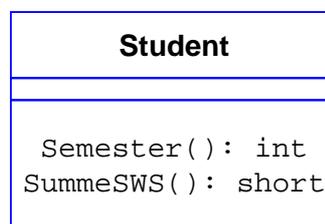
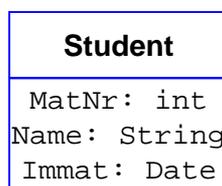
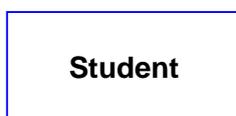
# Unified Modeling Language (UML)

- standardisierte graphische Notation/Sprache zur objektorientierten Modellierung
- Kombination unterschiedlicher Modelle bzw. Notationen, u.a.
  - Booch
  - Rumbaugh (OMT)
  - Jacobson (Use Cases)
- Standardisierung seit 1997 durch Herstellervereinigung OMG (Object Management Group)
- Mehrere UML-Teile für unterschiedliche Phasen der Systementwicklung
  - Anwendungsfälle (use cases)
  - Aktivitätsdiagramme
  - Klassendiagramme ...
- Infos: [www.uml.org](http://www.uml.org)



## Darstellung von Klassen und Objekten

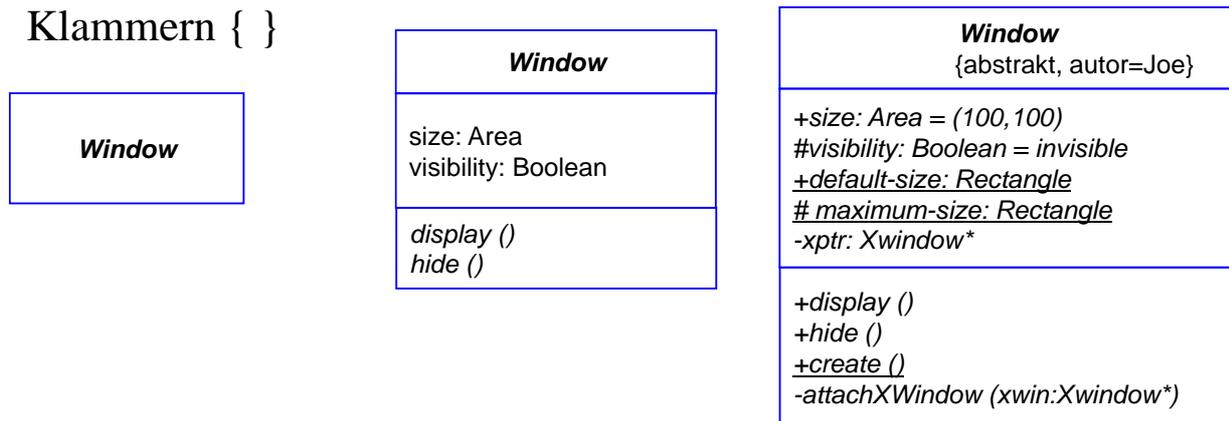
- Klassensymbol: Angabe von Klassenname, Attribute (optional), Methoden (optional)
  - i. a. werden nur relevante Details gezeigt



## Darstellung von Klassen (2)

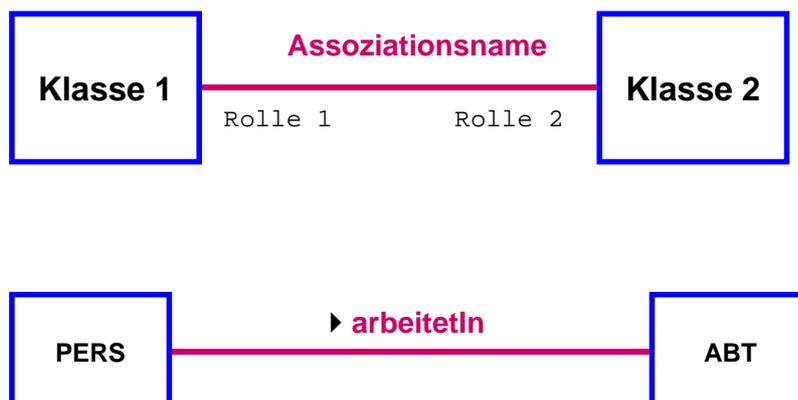
### ■ Detaildarstellung auf Implementierungsebene

- Attributspezifikation:  
*Sichtbarkeit Name: Typ = Default-Wert { Eigenschaften }*
- Operationen: *Sichtbarkeit Name (Parameterliste) : Rückgabeausdruck { Eigenschaften }*
- Sichtbarkeit: öffentlich / public (+), geschützt / protected (#), privat (-)
- unterstrichen: Klassen-Attribute / -Operationen
- Darstellung von Bedingungen (Constraints) innerhalb geschweifter Klammern { }



## Assoziationen

- Repräsentation von Beziehungen (relationships)
- optional: Festlegung eines Assoziationsnamens, seiner Leserichtung ( ▶ bzw. ◀ ), von Rollennamen, Sichtbarkeit von Rollen (+, -, #) sowie Kardinalitätsrestriktionen



# Kardinalitätsrestriktionen in UML

## ■ Verfeinerung der Semantik eines Beziehungstyps durch Kardinalitätsrestriktionen

- bisher nur grobe strukturelle Festlegungen (z. B.: 1:1 bedeutet “höchstens eins zu höchstens eins”)
- Festlegung der minimalen Kardinalität
- Erweiterung auf n-stellige Beziehungen

## ■ Definition: für binäre Assoziation $R \subseteq E_1 \times E_2$

- Multiplizität  $\min_1 .. \max_1$  ( $\min_2 .. \max_2$ ) bedeutet, dass zu jedem E2 (E1)-Element wenigstens  $\min_1$  ( $\min_2$ ) und höchstens  $\max_1$  ( $\max_2$ ) Instanzen von E1 (E2) enthalten sein müssen (mit  $0 \leq \min_i \leq \max_i$ ,  $\max_i \geq 1$ )
- Bezugnahme zur „gegenüberliegenden“ Klasse

- erlaubt Unterscheidung, ob Beziehungsteilnahme *optional* (Mindestkardinalität 0) oder *obligatorisch* (Mindestkardinalität  $\geq 1$ ) ist



# Kardinalitätsrestriktionen in UML (2)

## ■ Zulässige Multiplizitätsfestlegungen

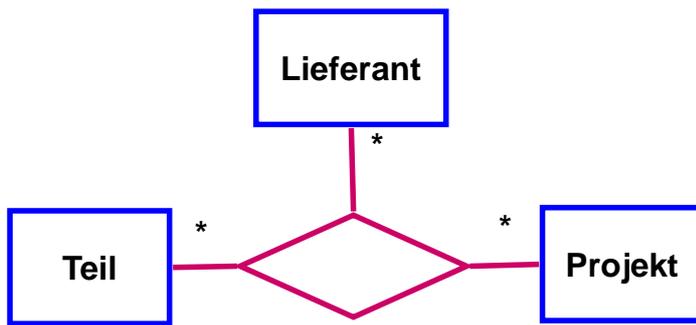
- x..y      mindestens x, maximal y Objekte nehmen an der Beziehung teil
- 0..\*      optionale Teilnahme an der Beziehung
- 1..\*      obligatorische Teilnahme an Beziehung
- 0..1
- 1          genau 1
- \*          „viele“



## ■ Beispiele

R	E <sub>1</sub>	E <sub>2</sub>	klassischer Beziehungstyp	min1..max1	min2..max2
Abt.Leutung	ABT	PERS			
Parteimitglied	PARTEI	PERS			
Verheiratet	FRAU	MANN			
V.teilname	VORL	STUDENT			
Belegung	PERS	ZIMMER			

# N-stellige Assoziationen



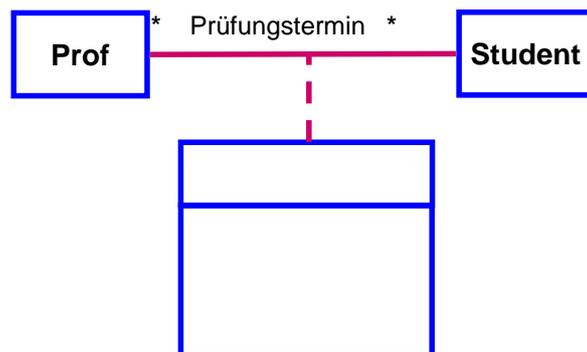
## ■ Multiplizitätsangabe einer Klasse

- regelt bei n-stelligen Beziehungen die Anzahl möglicher Instanzen (Objekte) zu einer fixen Kombination von je einem Objekt der übrigen n-1 Assoziationsenden

# Weitere Assoziationen

## ■ Assoziations-Klassen

- notwendig für Beziehungen mit eigenen Attributen
- gestrichelte Linie
- Name der A.-Klasse entspricht dem der Assoziation



## ■ gerichtete Assoziation

- Einschränkung der Navigierbarkeit: keine direkte Navigationsmöglichkeit in umgekehrter Richtung (einfachere Implementierung)
- auf konzeptioneller Ebene nicht notwendigerweise festzulegen



# Is-A-Beziehungen

## ■ Is-A-Beziehung zwischen Klassen (Entity-Mengen)

- $E_1$  is-a  $E_2$  bedeutet, dass jedes Objekt aus  $E_1$  auch ein Objekt aus  $E_2$  ist, jedoch mit zusätzlichen strukturellen Eigenschaften
- *Substitutionsprinzip*: alle Instanzen einer Subklasse sind auch Instanzen der Superklasse

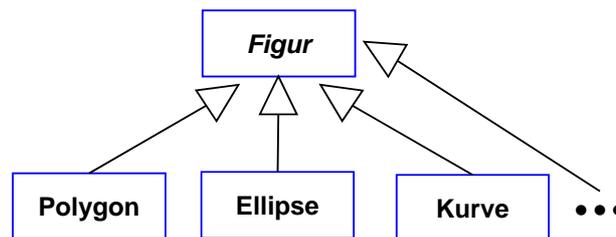
## ■ Vererbung von Eigenschaften (Attribute, Integritätsbedingungen, Methoden ...) der Superklasse an alle Subklassen

- Wiederverwendbarkeit, Erweiterbarkeit
- keine Wiederholung von Beschreibungsinformation, Fehlervermeidung

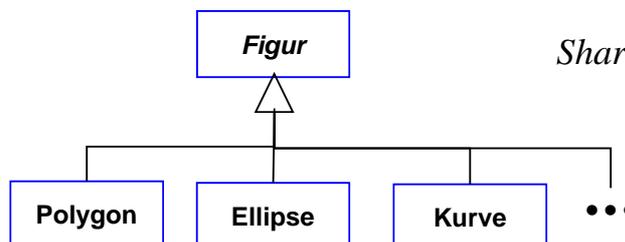


# UML-Darstellungsvarianten Generalisierung

*Separate Target Style* (Pfeil pro Verbindung)



*Shared Target Style*

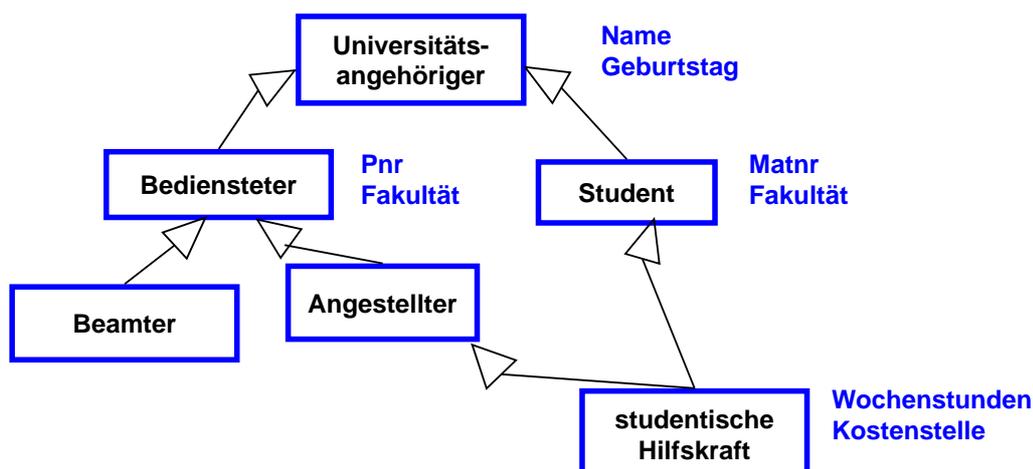


## Generalisierung/Spezialisierung

- Is-A-Beziehungen realisieren Abstraktionskonzepte der Generalisierung und Spezialisierung
- Generalisierung: Bottom-Up-Vorgehensweise
  - Bildung allgemeinerer Superklassen aus zugrundeliegenden Subklassen
  - Übernahme gemeinsamer Eigenschaften und Unterdrückung spezifischer Unterschiede
  - rekursive Anwendbarkeit => Generalisierungshierarchie
- Spezialisierung: Top-Down-Vorgehensweise
  - zuerst werden die allgemeineren Objekte (Superklassen), dann die spezielleren (Subklassen) beschrieben

## Generalisierung/Spezialisierung (2)

- oft keine reine Hierarchien, sondern Netzwerke (n:m)
  - eine Klasse kann Subklasse mehrerer Superklassen sein
  - ein Objekt kann gleichzeitig Instanz verschiedener Klassen sein
  - Zyklen nicht erlaubt/sinnvoll (A is-a B, B is-a A)
- führt zum Problem der **Mehrfach-Vererbung**
  - Namenskonflikte möglich
  - benutzergesteuerte Auflösung, z. B. durch Umbenennung



## Spezialisierung: Definitionen

- **Klasse:** Menge von Entities (Entity-Mengen, Superklassen, Subklassen)
- **Subklasse:** Klasse  $S$ , deren Entities eine Teilmenge einer Superklasse  $G$  sind (is-a-Beziehung), d. h.  $S \subseteq G$   
d. h. jedes Element (Ausprägung) von  $S$  ist auch Element von  $G$ .
- **Spezialisierung:**  $Z(G) = \{S_1, S_2, \dots, S_n\}$

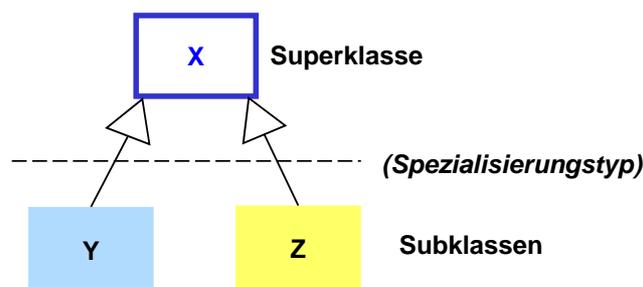
Menge von Subklassen  $S_i$  mit derselben Superklasse  $G$

- Zusätzliche Integritätsbedingungen: Vollständigkeit (Überdeckung) und Disjunktheit von Spezialisierungen

$Z$  heisst **vollständig (complete)**, falls gilt:  $G = \cup S_i$  ( $i = 1..n$ )  
andernfalls **partiell (incomplete)**.

$Z$  ist **disjunkt (disjoint)**, falls  $S_i \cap S_j = \{ \}$  für  $i \neq j$   
andernfalls **überlappend (overlapping)**.

## Arten von Spezialisierungen

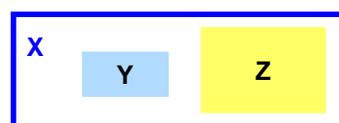


- disjunkte Spezialisierungen (Partitionierung)

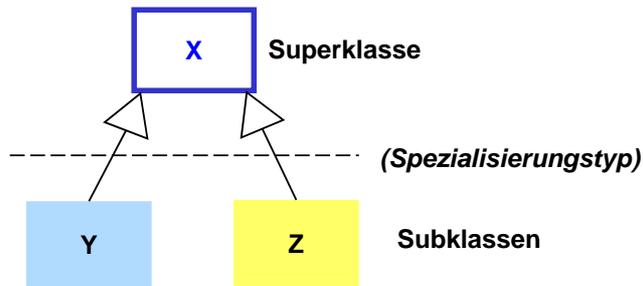
vollständig, disjunkt (*complete, disjoint*)



partiell, disjunkt (*incomplete, disjoint*)



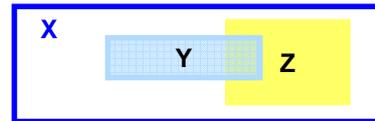
# Arten von Spezialisierungen (2)



## überlappende Spezialisierungen

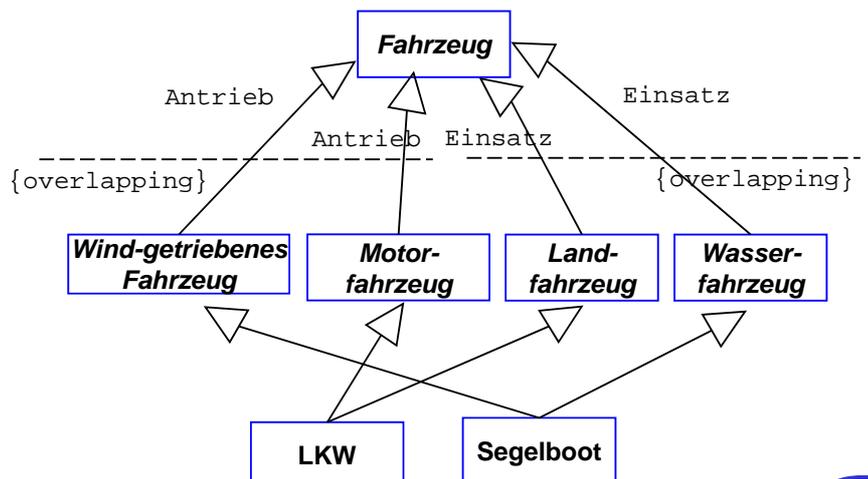
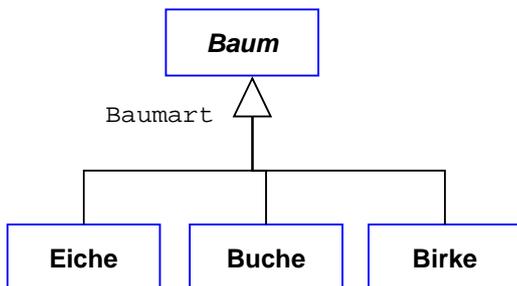
vollständig, überlappend (*complete, overlapping*)

partiell, überlappend (*incomplete, overlapping*)



# Spezialisierung (3)

- Angabe von Diskriminatoren (zB Attribut der Superklasse, das Zuordnung zu Subklassen bestimmt)

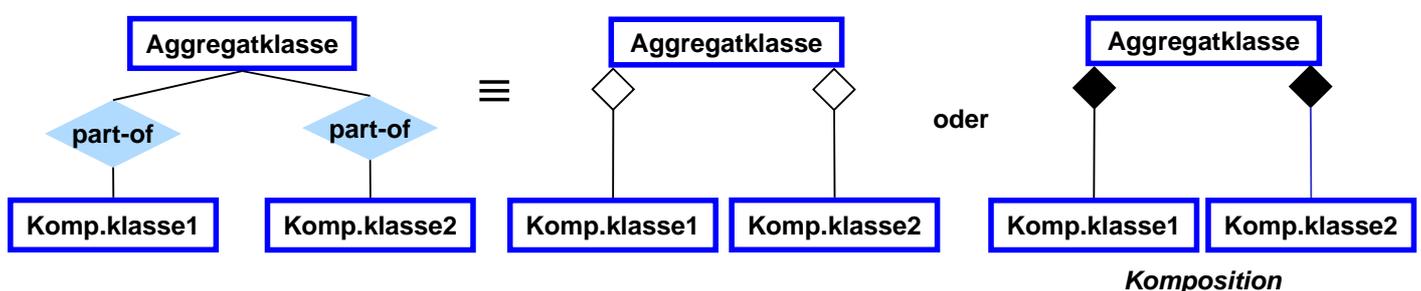


# Aggregation

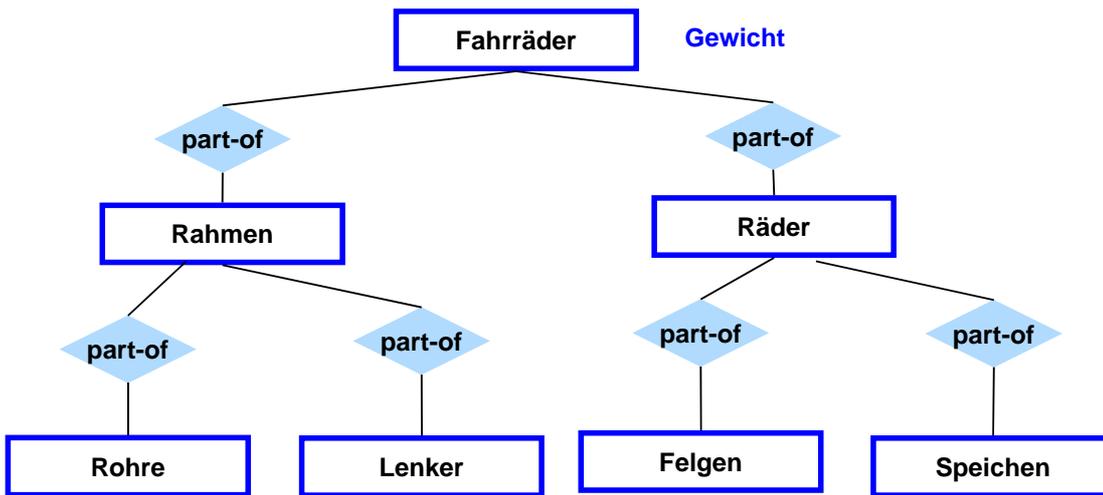
- Objekte werden als Zusammensetzung von anderen Objekten angesehen
  - eine Kombination von einfachen (atomaren, d.h. nicht weiter zerlegbaren) Objekten (Element, **Teil**) wird betrachtet als zusammengesetztes Objekt (**Aggregatobjekt**)
  - rekursive Anwendung des Aggregatsprinzips: Aggregationsobjekte mit komplexen Komponenten
- Einfache Formen der Aggregation:
  - zusammengesetzte Attribute
  - Entity-Menge als Aggregation verschiedener Attribute
- Erweiterung auf Part-of-Beziehung zwischen Entity-Mengen / Klassen

## Aggregation (2)

- **Part-of-Beziehung** (Teil-von-Beziehung) zwischen Komponenten- und Aggregatobjekten
  - Elemente einer Subkomponente sind auch Elemente aller Superkomponenten dieser Subkomponente
  - **Referenzsemantik** ermöglicht, dass ein Objekt gleichzeitig Element verschiedener Komponenten bzw. Subkomponente von mehreren Superkomponenten ist -> Netzwerke, (n:m) !
  - **Wertesemantik (Komposition)**: Teil-Objekt gehört genau zu einem Aggregat-Objekt; Existenzabhängigkeit!



# Aggregation (3)



- Unterstützung komplex-strukturierter Objekte
- heterogene Komponenten möglich
- keine Vererbung !



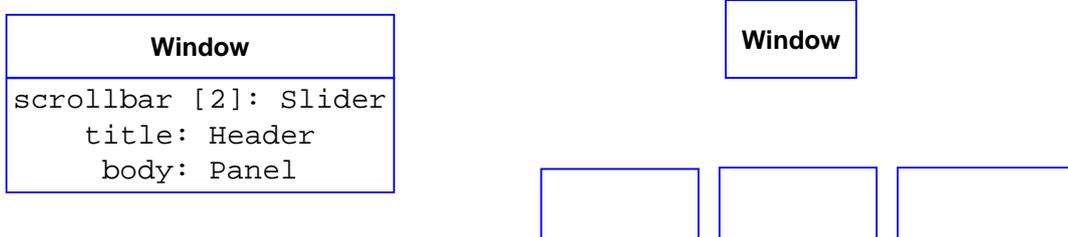
# Aggregation (UML)

- **Aggregation:** spezielle Assoziation zwischen 2 Klassen, in der eine Klasse eine andere vollständig (whole) enthält

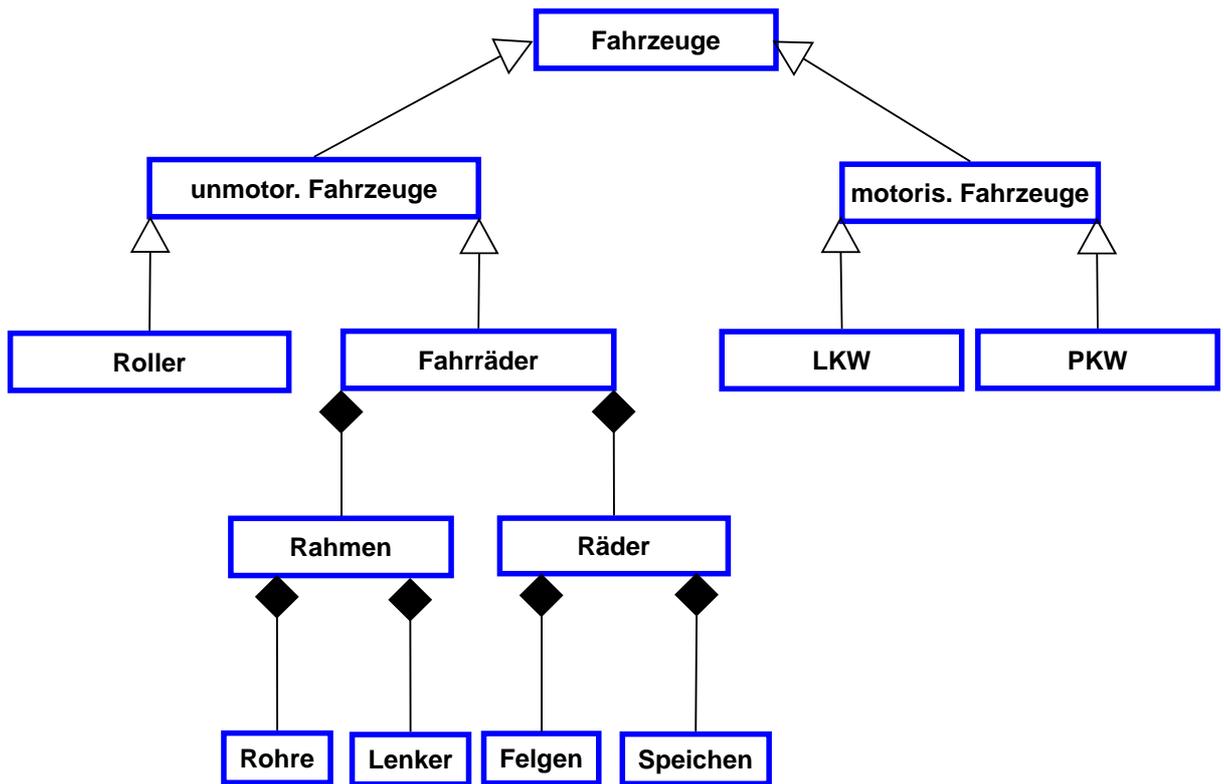


## ■ Komposition

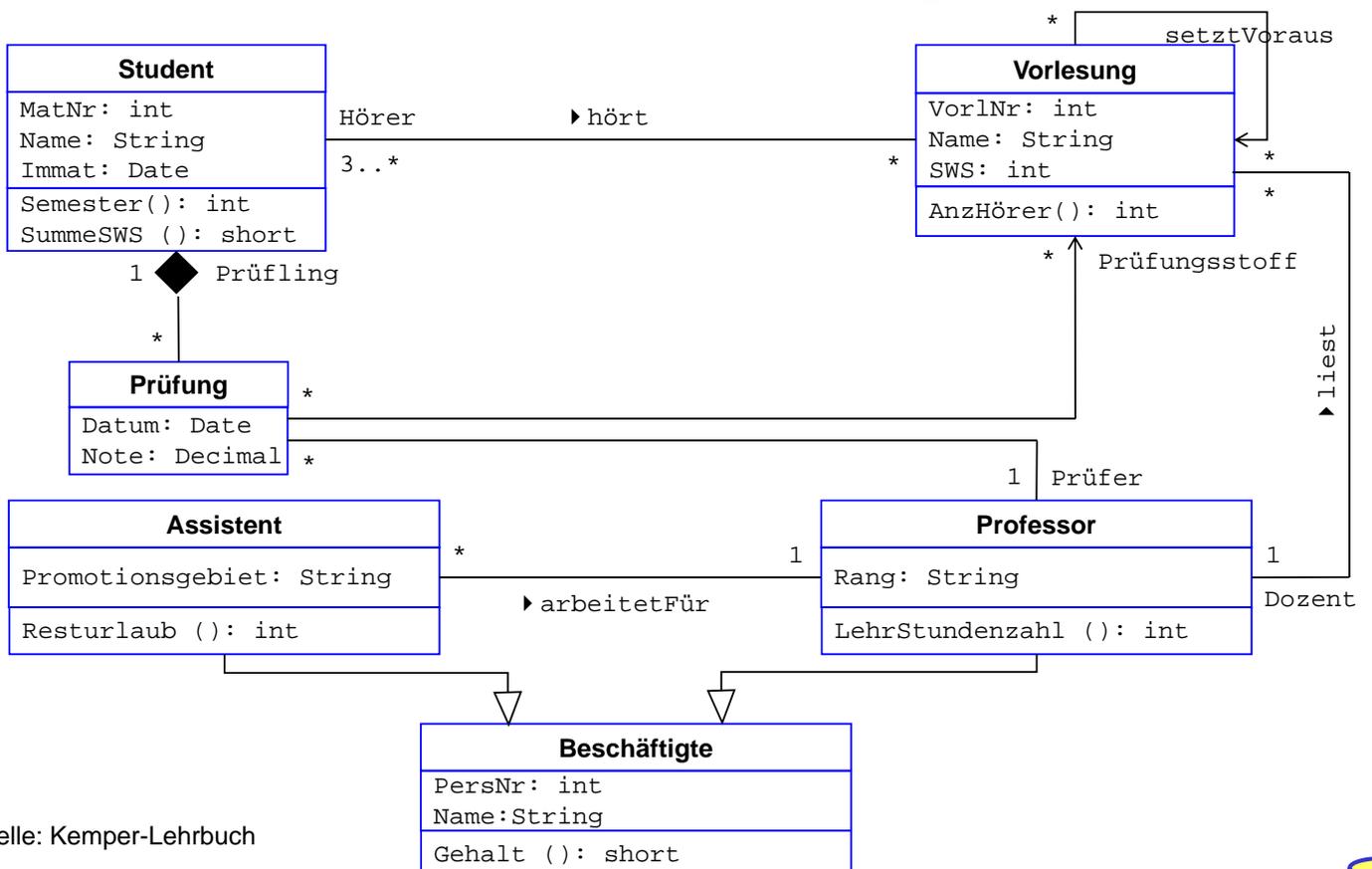
- Komponenten-Klasse einer Aggregatklasse A entspricht einem Attribut von A
- Äquivalente Repräsentationen:



# Kombination von Generalisierung und Aggregation



## Beispiel UML-Klassendiagramm



# Zusammenfassung

- DB-Entwurf umfasst
  - Informationsanalyse
  - konzeptueller Entwurf (-> Informationsmodell)
  - logischer Entwurf (-> logisches DB-Schema)
  - physischer Entwurf (-> physisches DB-Schema)
- Informationsmodellierung mit dem ER-Modell
  - Entity-Mengen und Relationship-Mengen
  - Attribute, Wertebereiche, Primärschlüssel
  - Beziehungstypen (1:1, n:1, n:m)
  - Diagrammdarstellung
- UML-Klassendiagramme: Unterschiede zu ER-Modell
  - standardisiert
  - Spezifikation von Verhalten (Methoden), nicht nur strukturelle Aspekte
  - genauere Kardinalitätsrestriktionen (Multiplizitäten)
  - Unterstützung der Abstraktionskonzepte der Generalisierung / Spezialisierung, Aggregation / Komposition
- keine festen Regeln zur eigentlichen Informationsmodellierung (i.a. viele Modellierungsmöglichkeiten einer bestimmten Miniwelt)

