

## 3. Informationsmodelle

Stefan DeBloch

## Überblick

- 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 (→ Vererbung)
  - Element-/Mengen-Assoziation
  - Element-/Komponenten-Aggregation
- Anhang
  - Vergleich von ERM- und UML-Konzepten

Informationssysteme

DB-Entwurf und Modellierung

ER-Modell

Erweiterungen

Abstraktionskonzepte

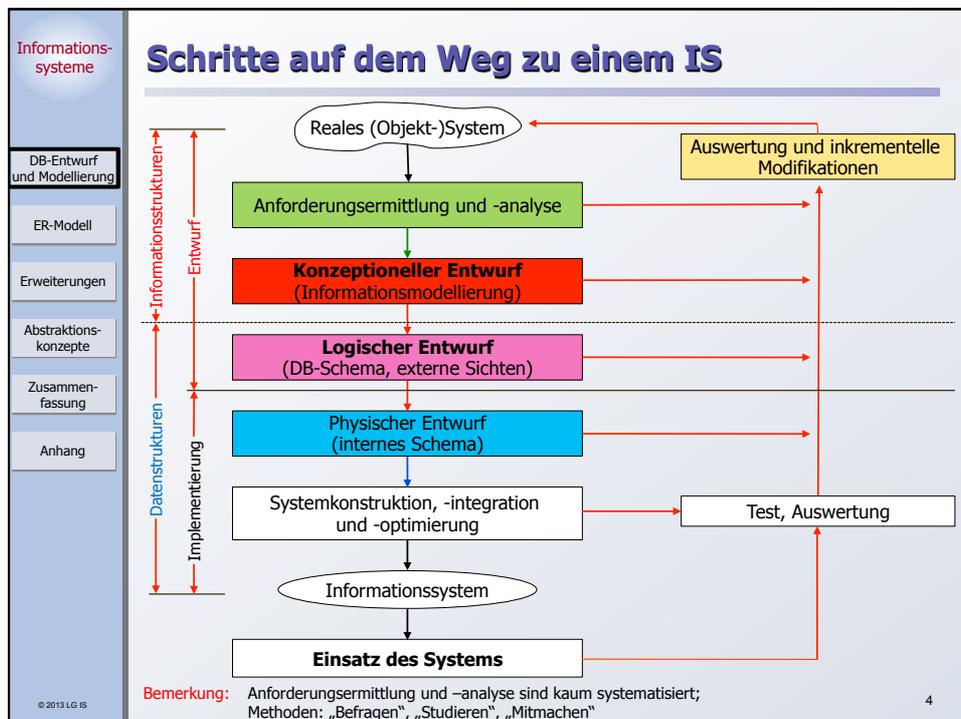
Zusammenfassung

Anhang

## 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 (allgemeines 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)

© 2013 LG IS 3



**Informationssysteme**

**Informationsmodelle**

DB-Entwurf und Modellierung

ER-Modell

Erweiterungen

Abstraktionskonzepte

Zusammenfassung

Anhang

© 2013 LG IS

5

Wirklichkeitsausschnitt („Miniwelt“)

Gegenstände  
Informationen  
Tatsachen  
Zusammenhänge  
Sachverhalte  
Eigenschaften  
Vorgänge  
Veränderungen

Formalisierung, Diskretisierung („Systemanalyse“)

Objekte  
Beziehungen  
Attribute

Informationsmodell

**Modellierungskonzepte**

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

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

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

**Informationssysteme**

**Von der Informationserhebung zum DB-Schema**

DB-Entwurf und Modellierung

ER-Modell

Erweiterungen

Abstraktionskonzepte

Zusammenfassung

Anhang

© 2013 LG IS

6

Prinzipielle Vorgehensweise

Informations-erhebung    Semantische Daten-modellierung    Logische Daten-modellierung    Datenbank-Installation

Bedeutungs-analyse    Grobdaten-modellierung    Feindaten-modellierung

Konzeptionelles Datenmodell

Zeit

- 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

Informationssysteme

---

DB-Entwurf und Modellierung

---

ER-Modell

---

Erweiterungen

---

Abstraktionskonzepte

---

Zusammenfassung

---

Anhang

---

© 2013 LG IS

## Wichtige Modellierungssprachen

---

- **ERM (Entity Relationship Model):**
  - generell einsetzbares Modellierungswerkzeug, hauptsächlich für den relationalen DB-Entwurf geeignet (Vergleich mit UML siehe Anhang)
- **UML (Unified Modeling Language):**
  - 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
- **STEP (STandard for the Exchange of Product Definition Data):**
  - Modellierung, Zugriff, Austausch von produktdefinierenden Daten über den gesamten Produktlebenszyklus

7

Informationssysteme

---

DB-Entwurf und Modellierung

---

ER-Modell

---

Erweiterungen

---

Abstraktionskonzepte

---

Zusammenfassung

---

Anhang

---

© 2013 LG IS

## Entity-Relationship-Modell (ERM)\* – Überblick

---

- **Modellierungskonzepte**
  - Entity-Mengen (Objektmengen)
  - Wertebereiche, Attribute
  - Primärschlüssel  
(**zentrales, wertbasiertes Konzept im ERM**; fehlt in UML, da Objekte immer einen systemweit eindeutigen Objektidentifikator zugeordnet bekommen)
  - 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**

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

8

## 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)

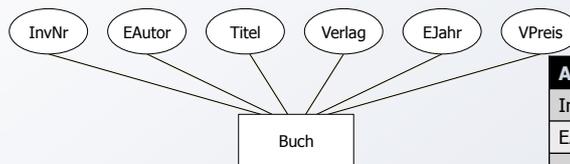
- Zusammenfassung von „ähnlichen“ oder „vergleichbaren“ Entities
- haben gemeinsame Eigenschaften
- Beispiele:
  - Abteilungen, Angestellte, Projekte, ...
  - Bücher, Autoren, Leser, ...
  - Studenten, Professoren, Vorlesungen, ...
  - Kunden, Vertreter, Wein, Behälter, ...

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

- ◇ Alle Attribute sind **einwertig!**
- ◇ Name der Entity-Menge sowie zugehörige Attribute sind **zeitinvariant**

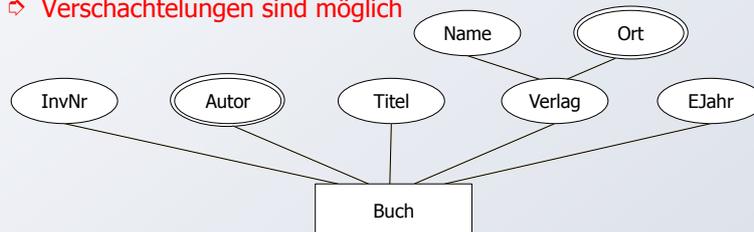
### ■ Entity-Menge und ihre Entities sind zeitveränderlich

$e_1 = (4711, \text{Kemper, DBS, Oldenbourg, ...})$   
 $e_2 = (0815, \text{Date, Intro. to DBS, Addison, ...})$   
 $e_3 = (1234, \text{Härder, DBS, Springer, ...})$

$\left. \begin{array}{l} t_1 \\ t_2 \end{array} \right\}$

## 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
  - bisher: einwertige Attribute
  - **mehrwertige Attribute** (Doppelovale)
  - **zusammengesetzte Attribute** (hierarchisch angeordnete Ovale)
  - ◇ **Verschachtelungen sind möglich**



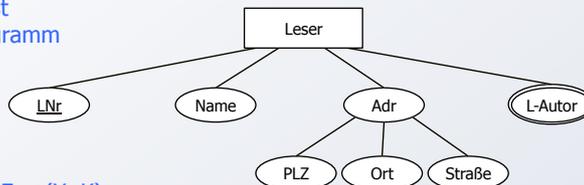
e3 = (1234, {Härder, Rahm}, DBS, (Springer, {Hd, Be}), 2001)

## 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 (z.B. laufende Nr.)
- $\{A_1, A_2, \dots, A_m\} = A$  sei Menge der (einwertigen) Attribute zur Entity-Menge  $E$ 
  - $K \subseteq A$  heißt Schlüsselkandidat von  $E$
  - ⇔  $K$  eindeutig:  $e_i, e_j \in E; e_i \neq e_j \rightarrow K(e_i) \neq K(e_j)$
  - und  $K$  irreduzibel (minimal):  $\forall K' \subset K: K'$  ist nicht eindeutig
- Mehrere Schlüsselkandidaten (SK) möglich
  - ◇ **Primärschlüssel auswählen**
  - Primärschlüsselattribute werden im ER-Diagramm durch Unterstreichen gekennzeichnet
- Beispiel:
  - Entity-Menge Student mit Attributen MatNr, SVNr, Name, Gebdat, FbNr
  - 1. MatNr 2. SVNr 3. Name, Gebdat

## Konzepte des ERM (5)

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



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

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

- Wertebereiche

$W(LNr) = int(8), \quad W(Name) = W(L-Autor) = char(30)$   
 $W(PLZ) = int(5), \quad W(Ort) = char(20), \quad W(Straße) = char(15)$   
 $dom(Adr) = W(PLZ) \times W(Ort) \times W(Straße) = int(5) \times char(20) \times char(15)$   
 $dom(L-Autor) = 2^{W(L-Autor)} = 2^{char(30)}$

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

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

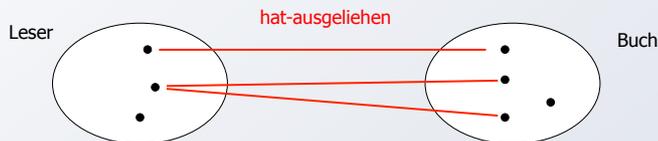
$dom(B) = W(C_1) \times W(C_2)$   
 $dom(D) = 2^{W(E_1) \times W(E_2)}$   
 $dom(A) = dom(B) \times dom(D)$

## 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“

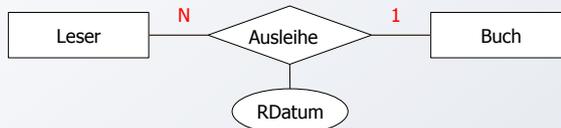


- Eigenschaften

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

## Konzepte des ERM (7)

### ER-Diagramm



Eine Beziehung R ist vom Typ **1:n**, falls (in jedem  $R^i$ ) ein Entity vom Typ  $E_1$  an  $n \geq 0$  Instanzen von R teilnimmt, d. h., mit  $n \geq 0$  Entities vom Typ  $E_2$  in Beziehung steht, andererseits jedoch jedes Entity vom Typ  $E_2$  höchstens an einer Instanz von R teilnimmt, also höchstens mit einem Entity vom Typ  $E_1$  assoziiert ist. Bei vertauschten Rollen von  $E_1$  und  $E_2$  spricht man von einer n:1-Beziehung.

**Achtung:** Wir schreiben die Funktionalität einer Beziehung an die Quellseite!

### Relationship-Typ $R = (Ent, Y)$ :

- Ausleihe = ((Leser, Buch), (RDatum))

### Eigenschaften

- Grad: 2
- Beziehungstyp: 1:n

## ERM – Definitionen\*

### 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:

- Einwertige Attribute: A
- Mehrwertige Attribute: {A}
- Zusammengesetzte Attribute: A (B1, ..., Bk)

### Bemerkung

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.

(\*) G. Vossen: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, Oldenbourg, 5. Auflage, 2008

## ERM – Definitionen (2)

### ■ Def. 2: Wertebereich (Domain)

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

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

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

## ERM – Definitionen (3)

### ■ Def. 3: Entity und Entity-Menge

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

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

$$e \in \text{dom}(A_1) \times \dots \times \text{dom}(A_m)$$

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

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

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

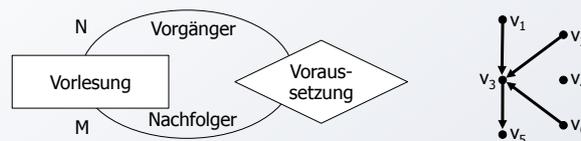
## ERM – Definitionen (4)

### Def. 4: Relationships

- i. Ein **Relationship-Typ** hat die Form  $R = (\text{Ent}, Y)$ . Dabei ist  $R$  der Name des Typs (auch „Name der Beziehung“),  $\text{Ent}$  bezeichnet die Folge der Namen der Entity-Typen, zwischen denen die Beziehung definiert ist, und  $Y$  ist eine (möglicherweise leere) Folge von Attributen der Beziehung.
- ii. Sei  $\text{Ent} = (E_1, \dots, E_k)$ , und für beliebiges, aber festes  $t$  sei  $E_i^t$  der Inhalt des Entity-Typs  $E_i$ ,  $1 \leq i \leq k$ . Ferner sei  $Y = (B_1, \dots, B_n)$ . Eine **Relationship**  $r$  ist ein Element des kartesischen Produktes aus allen  $E_i^t$  und den Domains der  $B_j$ , d. h.
 
$$r \in E_1^t \times \dots \times E_k^t \times \text{dom}(B_1) \times \dots \times \text{dom}(B_n)$$
 bzw.
 
$$r = (e_1, \dots, e_k, b_1, \dots, b_n)$$
 mit
 
$$e_i \in E_i^t \text{ für } 1 \leq i \leq k \text{ und } b_j \in \text{dom}(B_j) \text{ für } 1 \leq j \leq n.$$
- iii. Eine **Relationship-Menge**  $R^t$  (zur Zeit  $t$ ) ist eine Menge von Relationships, d. h.,
 
$$R^t \subseteq E_1^t \times \dots \times E_k^t \times \text{dom}(B_1) \times \dots \times \text{dom}(B_n).$$

## Relationship-Mengen

### Motivation für Rollennamen



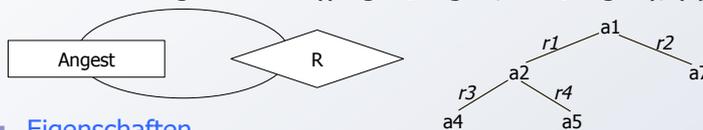
### Definition:

- $\text{Voraussetzung} = ((\text{Vorlesung}, \text{Vorlesung}), (\emptyset))$
- 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  $v_j$ “
- **Transitivität gilt bei Selbstreferenz i. Allg. nicht!**
  - Bsp: „ $p_i$  kennt  $p_j$ “

## Relationship-Mengen (2)

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

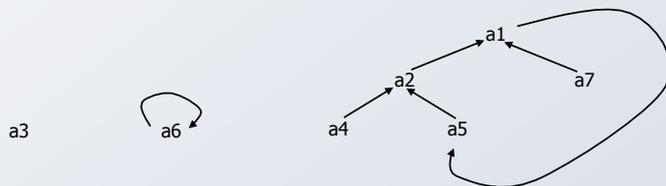
Direkter-Vorgesetzter = ((Angest/Angest, Chef/Angest), ( $\emptyset$ ))



- Eigenschaften

- Grad: 2
- Beziehungstyp: n:1

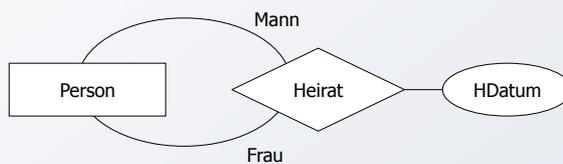
- R sei Direkter-Vorgesetzter. Welche Beziehungen auf Angestellter sind zulässig?



## Relationship-Mengen (3)

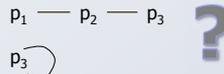
- Relationship-Menge

Heirat = ((Mann/Person, Frau/Person), (HDatum))



- Eigenschaften

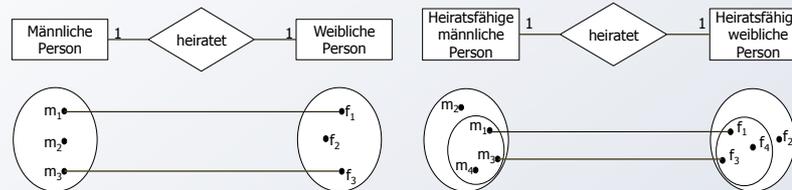
- Grad: 2
- Beziehungstyp: 1:1



Zusätzliche Integritätsbedingungen erforderlich!

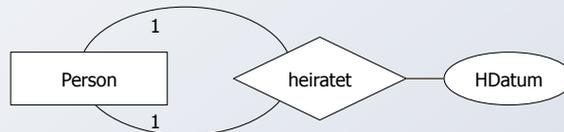
## Relationship-Mengen (4)

- Ist Anpassung der Entity-Definition an zu modellierenden Relationship-Typ sinnvoll?



- Was passiert wenn IB (versch. Geschlecht, Alter  $\geq 16$ ) nicht mehr gilt?
- Was passiert mit n anderen Beziehungstypen für „Person“

- Deshalb als Modellierung der aktuellen Sicht

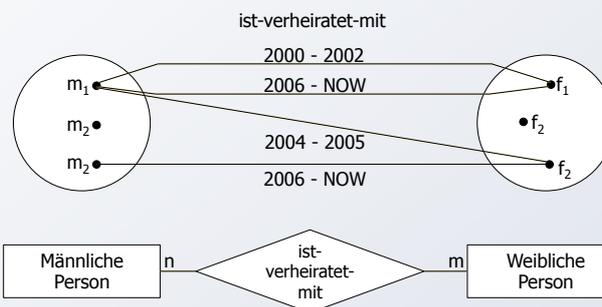


- + Alter  $\geq 16$
- + versch. Geschlecht
- + ...

## Relationship-Mengen (5)

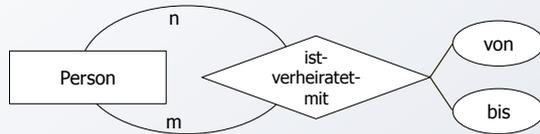
- Was passiert bei der Modellierung der Historie?

Neuer Aspekt: Gültigkeit einer Beziehung ist zeitabhängig!



## Relationship-Mengen (6)

■ Oder besser als Modellierung der historischen Sicht



■ Wie erhält man eine korrekte Abbildung der Miniwelt

- ER-Modell enthält (nur) strukturelle Integritätsbedingungen
- 1:1 Beziehungstyp bezieht sich nun auf von/bis-Zeitraum
  - deshalb **jetzt** n:m für vollständige Sicht!

⇒ Integritätsbedingungen zur Werteüberprüfung (zeitliche Überlappungsfreiheit) erforderlich!

■ Wenn nicht anders vermerkt, modellieren wir immer die aktuelle (also nicht die historische) Sicht!

## Existenzabhängigkeit

■ Existenzabhängige (schwache) Entity-Typen

- Existenz der Entities ist abhängig von übergeordneten Entities (immer!)
  - Notation: doppelt umrandete Entity/Relationship Typen, doppelte Kante
- Entities sind oft nur in Kombination mit dem Schlüssel des übergeordneten Entitytyps eindeutig identifizierbar (optional!)
  - Notation: lokaler Schlüsselanteil ist gestrichelt unterstrichen

■ Beispiele



Existenzabhängigkeit: „Relationship begründet Existenz von“



## Existenzabhängigkeit (2)

### ■ Existenzabhängigkeit als Integritätsbedingung

Die Benutzung des Konzepts „Existenzabhängigkeit“ sollte eine **bewusste Entscheidung des Modellierers** sein, da er damit eine zusätzliche Integritätsbedingung in das Modell aufnimmt. Oft gibt die **Wahl/Zusammensetzung des Schlüssels** einen Hinweis auf eine Existenzabhängigkeit. Beispielsweise kann sich der Schlüssel eines Raumes zusammensetzen aus dem Gebäudeschlüssel (Schlüssel des übergeordneten Entitytyps) und einer lokal gültigen Raumnummer (lokaler Schlüsselanteil, gestrichelt unterstrichen im ER-Diagramm), z.B. 46, 220.

### ■ Modellierungsalternative

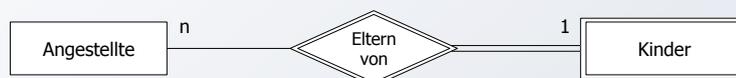
Wenn im Modell hauptsächlich Räume und beispielsweise Vorlesungen dargestellt werden sollen und Gebäude nur eine weitere Eigenschaft von Räumen sind, könnte man auf die Spezifikation von Gebäuden als Entities verzichten und ihre Nummer als **beschreibendes Attribut** von Räumen aufnehmen.

## Existenzabhängigkeit (3)

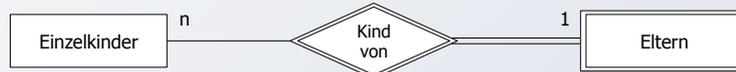
### ■ Existenzabhängigkeit kann anwendungsabhängig sein

#### ■ Beispiele

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



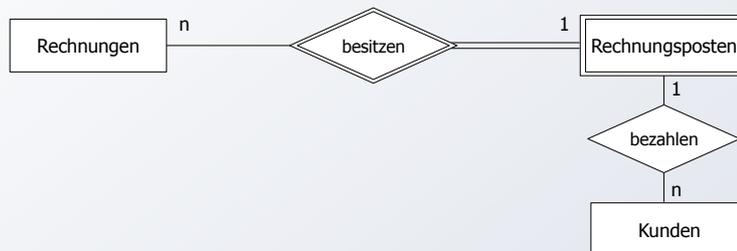
2. In einer Anwendung „Betreuung von Einzelkindern“ müssen Informationen über die Eltern ermittelt werden können:



Wenn das Haupt-Entity die Firma oder den Kindergarten verlässt, ist die Information über das abhängige Entity für die Anwendung nicht mehr von Interesse. Sie kann also **automatisch** aus dem System entfernt werden.

## Existenzabhängigkeit (4)

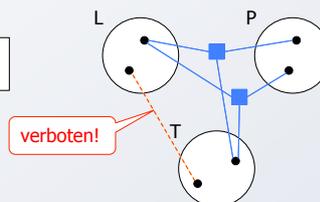
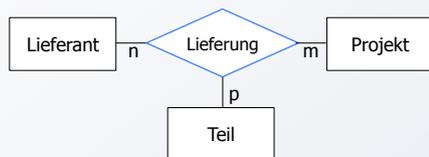
### Verfeinerung



### Bemerkung:

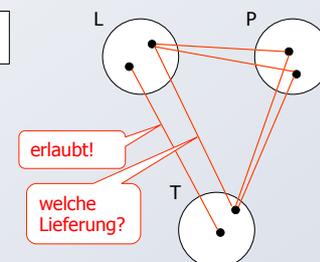
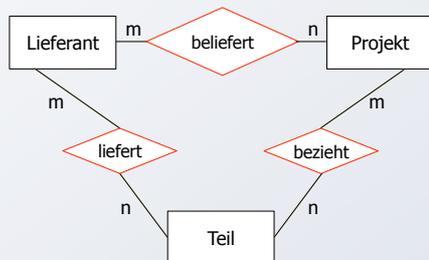
In manchen Modellen steht eine existenzabhängige Entity-Menge rechts von der selbständigen Entity-Menge und der „erzeugenden“ Relationship-Menge. Bei Mehrfachreferenzen ist eine „erzeugende“ von weiteren „referenzierenden“ Relationship-Mengen zu unterscheiden.

## Dreistellige Relationship-Mengen



### ACHTUNG:

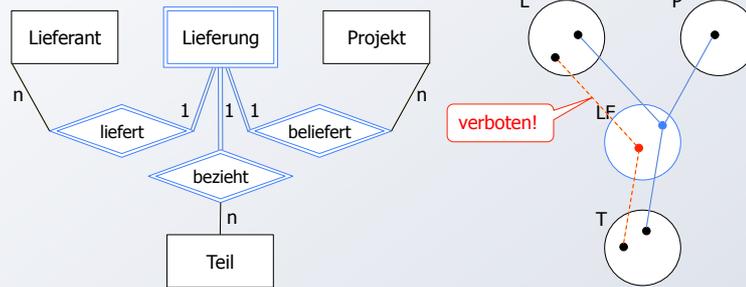
Nicht gleichwertig mit drei zweistelligen (binären) Relationship-Mengen!



## Dreistellige Relationship-Mengen (2)

**ABER:**

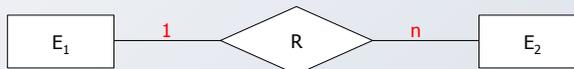
Manche Systeme erlauben nur die Modellierung binärer Relationship-Mengen



## Klassifikation von Datenabbildungen

- Ziel:
  - Festlegung von semantischen Aspekten (hier: Beziehungstyp)
  - explizite Definition von strukturellen Integritätsbedingungen
- Unterscheidung von Beziehungstypen
  - $E_i - E_j$
  - $E_i - E_i$
- Festlegung der Abbildungstypen
  - 1:1 - eindeutige Funktion (injektive Abbildung)
  - n:1/1:n - 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 ↔ ARBGRUPPE
  - n:1 ARBEITET\_FÜR/MIT: MITARBEITER → PROF
  - n:m BESCHÄFTIGT/IST\_HIWI: PROF — STUDENT

⚠ Abbildungstypen implizieren nicht, dass für jedes  $e_x \in E_i$  auch tatsächlich ein  $e_j \in E_j$  existiert
- Diagrammdarstellung:

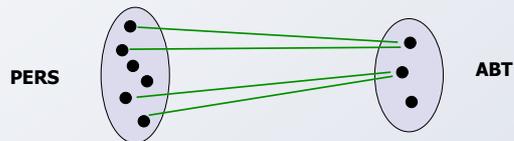


## Klassifikation von Datenabbildungen (2)

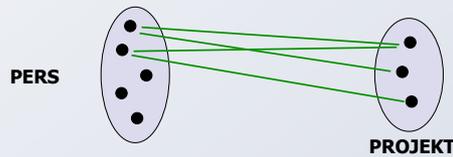
1:1: LEITET/WIRD\_GELEITET: PERS  $\leftrightarrow$  ABT



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



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



## ER-Schema – Beispiel

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

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

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

## ER-Schema – Beispiel (2)

**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

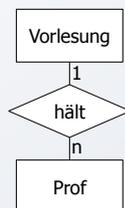
**DECLARE** RELATIONSHIP RELATION PERS.-ANGEHÖRIGE  
ROLE/ENTITY-RELATION.PK/MAX-NO-OF-ENTITIES  
 UNTERHALTSPFLICHTIGER/PERSONAL.PK/1  
 KIND/ KINDER.PK/n  
EXISTENCE OF KIND DEPENDS ON  
EXISTENCE OF UNTERHALTSPFLICHTIGER

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

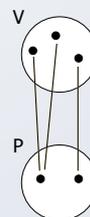
## Varianten der Modellierung – Professoren halten Vorlesungen

- Eine Vorlesung wird immer vom selben Professor gehalten; es interessiert nicht, wo und wann sie stattfindet.

ER-Diagramm

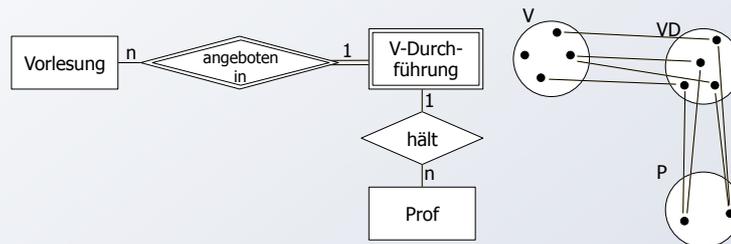


Darstellung der Beziehung auf Instanzebene



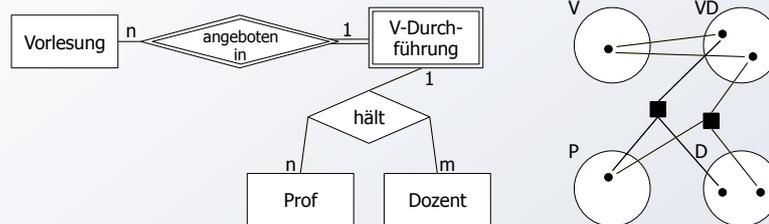
## Variante 2

- Eine Vorlesung kann wechselseitig von verschiedenen Professoren gehalten werden. Dabei soll die Durchführung einer Vorlesung (wo, wann, von wem gehalten) explizit modelliert werden.

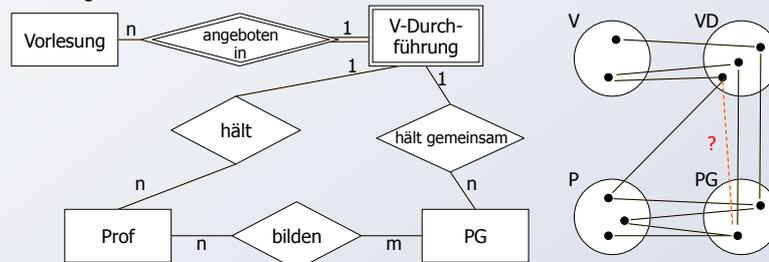


## Variante 3 – mehrere Profs halten eine VL

- a) Ein Professor und ein Dozent halten zusammen eine Vorlesung

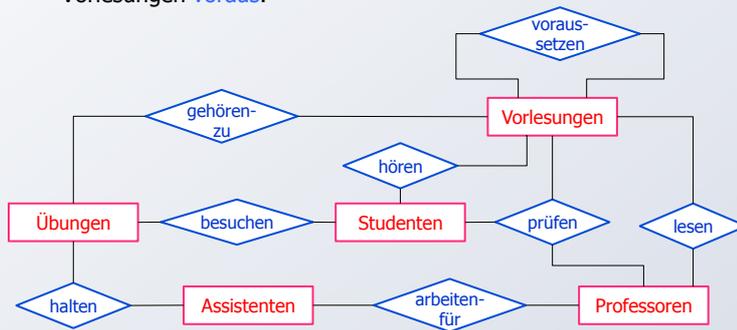


- b) Ein Professor hält oder n Professoren (Professorengruppe) halten gemeinsam eine Vorlesung

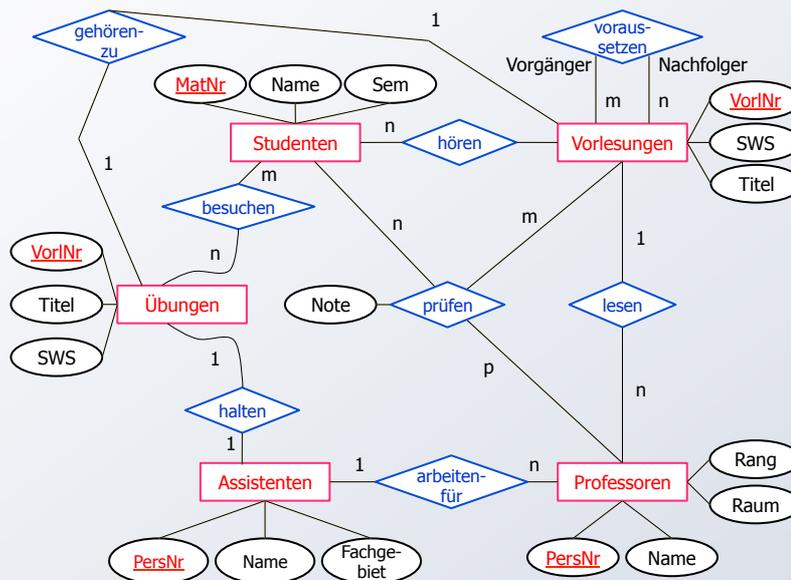


## Anwendungsbeispiel: Vorlesungsbetrieb

- Stellen Sie ein ER-Diagramm für folgende Miniwelt auf:
  - Jeder **Professor** hält mehrere seiner **Vorlesungen** und prüft **Studenten** jeweils über eine dieser Vorlesungen.
  - Mehrere **Assistenten** arbeiten jeweils für einen Professor und halten **Übungen**, die zu den entsprechenden Vorlesungen gehören.
  - Mehrere **Studenten** hören jeweils eine Reihe von Vorlesungen. Übungen und Vorlesungen werden jeweils von mehreren **Studenten** besucht.
  - Der Besuch von Vorlesungen setzt i. Allg. die Kenntnis anderer Vorlesungen voraus.



## ER-Diagramm - Vorlesungsbetrieb



## Anwendungsbeispiel: Data Warehousing

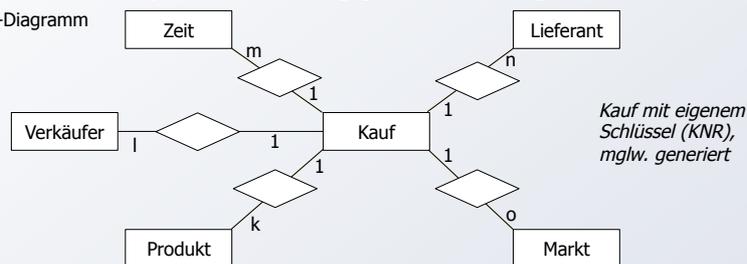
### ■ Modellierung des Kaufs von Waren in Supermärkten

- Zur Geschäftsanalyse möchte man in einer Supermarktkette die wesentlichen Informationen sammeln, die den **Kauf eines Produktes** charakterisieren
- Beteiligt sind die Entity-Typen **Verkäufer, Produkt, Lieferant, Markt und Zeit**
- Es ist möglicherweise unpraktisch, für jeden Kauf (im Extremfall werden die Daten dafür durch einen Scanner-Strich an der Kasse des Supermarktes erzeugt) einen separaten eindeutigen Schlüssel zuzuordnen. Deshalb können die Schlüssel der beteiligten Entities dazu benutzt werden.

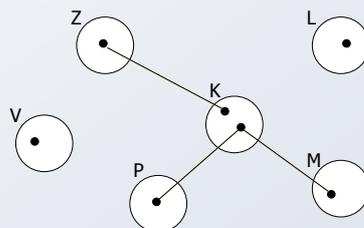
## Megabeispiel – Data Warehousing

### ■ Kauf als Entity mit unabhängigen Beziehungen

ER-Diagramm



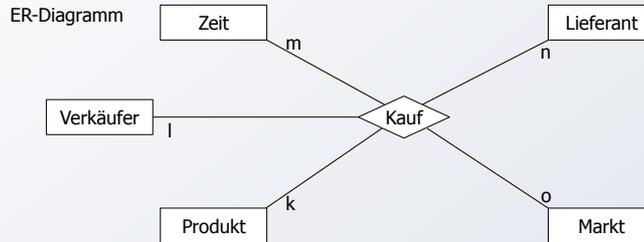
Beziehungen auf Instanzebene



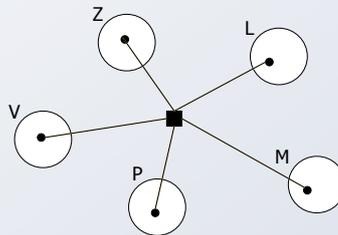
*n unabhängige Beziehungen! ausschließlich Benutzer für die Wartung der Semantik verantwortlich (insert, delete)*

## Megabeispiel – Data Warehousing (2)

### Kauf als 5-stellige Beziehung

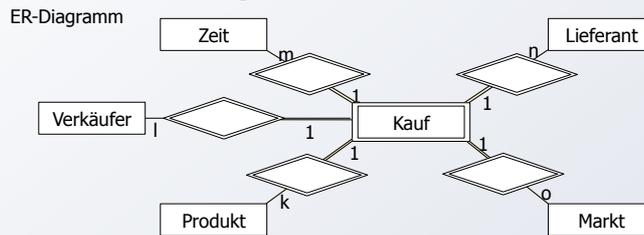


Beziehungen auf Instanzebene

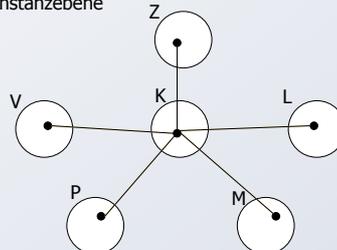


## Megabeispiel – Data Warehousing (3)

### Kauf als existenzabhängiges Entity mit 5 begründenden binären Beziehungen



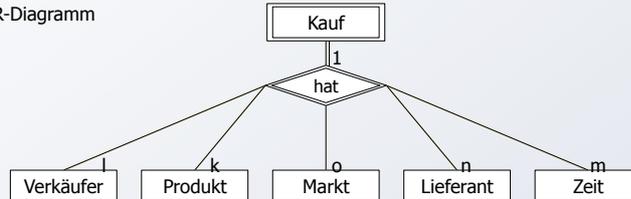
Beziehungen auf Instanzebene



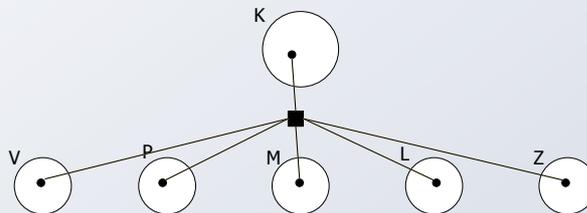
## Megabeispiel – Data Warehousing (4)

- Kauf als existenzabhängiges Entity mit einer begründenden 6-stelligen Beziehung

ER-Diagramm



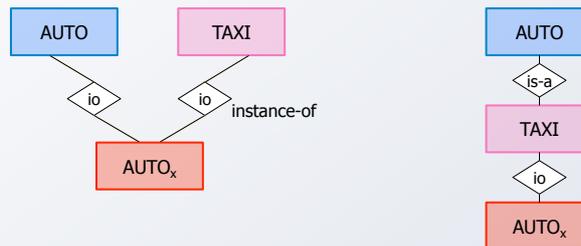
Beziehungen auf Instanzebene



## Erweiterungen des ERM

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

Beispiel: Unangemessene Modellierung bei überlappenden EM



- Ziele

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

## Verfeinerung der Datenabbildung: Kardinalitätsrestriktionen

- **Bisher:**  
grobe strukturelle Festlegung der Beziehungen  
z. B.: 1:1 bedeutet „höchstens eins zu höchstens eins“
- **Verfeinerung der Semantik eines Beziehungstyps durch Kardinalitätsrestriktionen:**  
Sei  $R \subseteq E_1 \times E_2 \times \dots \times E_n$ :  
Kardinalitätsrestriktion  $kard(R, E_i) = [min, max]$   
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**

Kardinalitätsrestriktionen



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

## Kardinalitätsrestriktionen (2)

- **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]

## Kardinalitätsrestriktionen (3)

### Caveat

- Viele Autoren (Kemper/Eickler, Elmasri/Navathe usw.), aber auch die UML-Notation, geben die **Funktionalität bei binären Beziehungen** nicht an der Quelle, sondern am Ziel der Beziehung an.
- Bei n-stelligen Beziehungen funktioniert diese Vorgehensweise nicht mehr, außer bei unspezifischen Angaben wie etwa n:m:k. Deshalb werden diese Notationen inkonsistent: der binäre wird anders als der n-stellige Fall behandelt.
- Bei der Angabe von spezifischen Kardinalitätsrestriktionen scheitern diese Ansätze ganz.

### Rechtfertigung unserer Notation

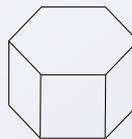
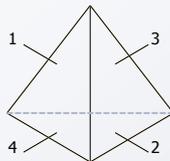
- Wir geben die **Funktionalität immer an der Quelle einer Beziehung** an. Bei binären Funktionalitätsangaben unterscheidet sich unser Ansatz nur bei 1:n (n:1).
- Dadurch erreichen wir eine konsistente Darstellung für alle Anwendungsfälle: bei binären und n-stelligen Beziehungen sowie bei Kardinalitätsrestriktionen.

### Modellieren hat Freiheitsgrade

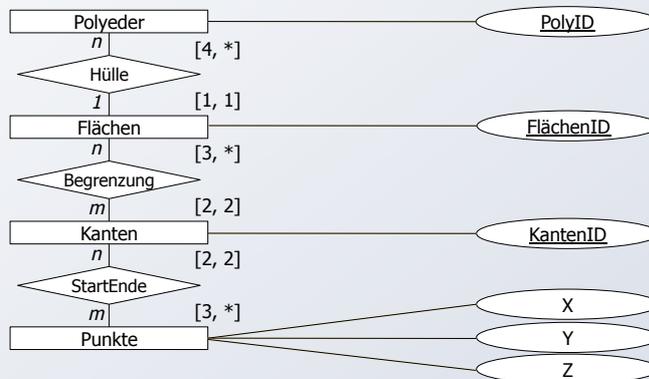
- Für eine Miniwelt, die für eine Anwendung zu modellieren ist, kann es **viele verschiedenen Lösungen geben** – gute, bessere, aber auch schlechtere oder gar falsche.
- Deshalb sollte der Entwerfer eines ER-Modells alle Freiheitsgrade nutzen, um zur besten Lösung zu kommen. Wie wir gesehen haben, hängt die Qualität der Lösung (siehe Diskussion der Existenzabhängigkeit) **stark von der Anwendung** ab. Vor- und Nachteile eines ER-Modells werden oft erst bei der Abbildung auf ein relationales DB-Schema deutlich.
- Wir verfolgen einige Konsequenzen von ER-Entwurfsentscheidungen in Kapitel 4 und 5.

## Begrenzungsflächendarstellung von Körpern

### Beispiel-Körper



### ER-Diagramm



Informations-  
systeme

---

DB-Entwurf  
und Modellierung

---

ER-Modell

---

Erweiterungen

---

Abstraktions-  
konzepte

---

Zusammen-  
fassung

---

Anhang

## Abstraktionskonzepte\*

---

- **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"

(\*) Mattos, N.: An Approach to Knowledge Management, LNAI 513, Springer, 1991

© 2013 LG IS
51

Informations-  
systeme

---

DB-Entwurf  
und Modellierung

---

ER-Modell

---

Erweiterungen

---

Abstraktions-  
konzepte

---

Zusammen-  
fassung

---

Anhang

## Abstraktionskonzepte

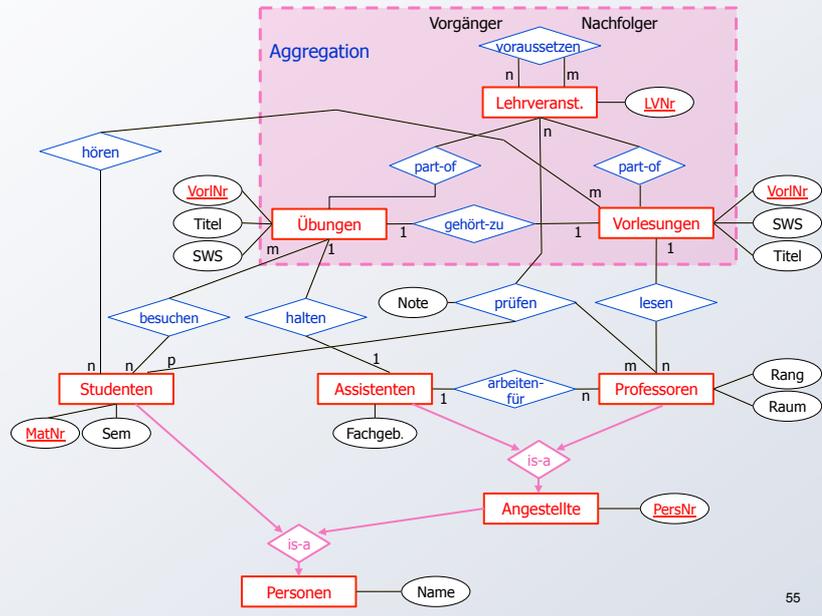
---

- **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

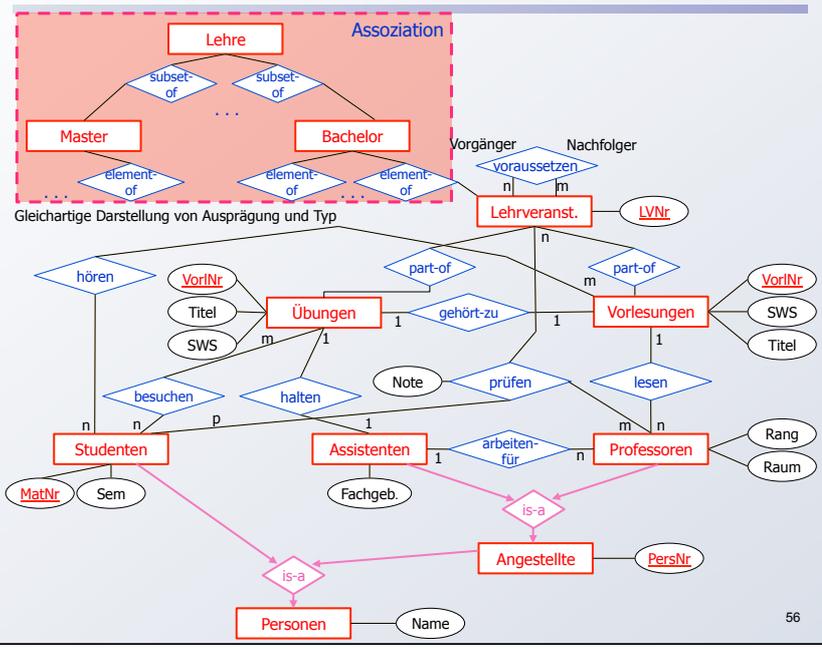
© 2013 LG IS
52



### Motivation (3)



### Motivation (4)



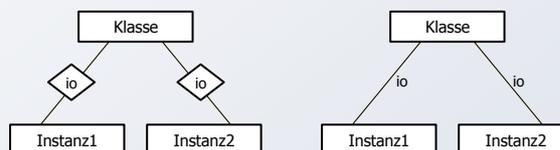
## Klassifikation

- 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

⇒ Modellierung beinhaltet Metadaten und Ausprägungen!

## 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



Wenn nicht explizit anders vermerkt, wird als Richtung in der graphischen Darstellung „von oben nach unten“ angenommen (Klasse – Instanz, Superklasse – Subklasse). Die Darstellungen der anderen Abstraktionskonzepte erfolgen entsprechend.

# Generalisierung

## Aufgabe

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

## Anwendung

- Sie baut die 'subclass-of'-Beziehung auf ('sc' oder 'is-a')
- 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)

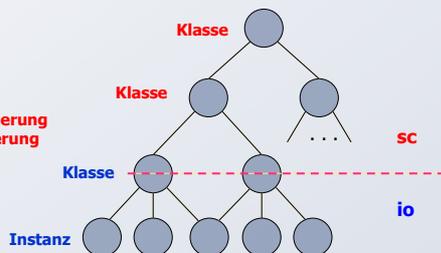
# Generalisierung (2)

## Struktureigenschaften der Generalisierung

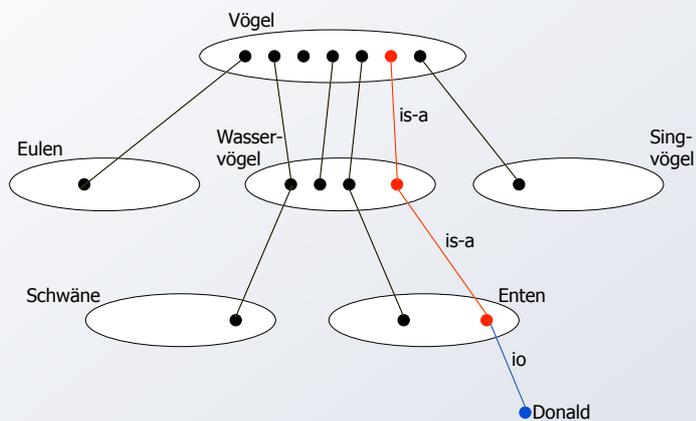
- Alle Instanzen einer Subklasse sind auch Instanzen der Superklasse
- Ein Objekt kann gleichzeitig Instanz verschiedener Klassen sein sowie auch Subklasse mehrerer Superklassen (⇨ Netzwerke, (n:m)!)
- Zugehörigkeit eines Objektes zu einer Klasse/Superklasse wird im wesentlichen bestimmt durch **Struktur**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

Generalisierungshierarchie (Is-a-Hierarchie)

↑ Klassifikation / Generalisierung  
 ↓ Instantiation / Spezialisierung

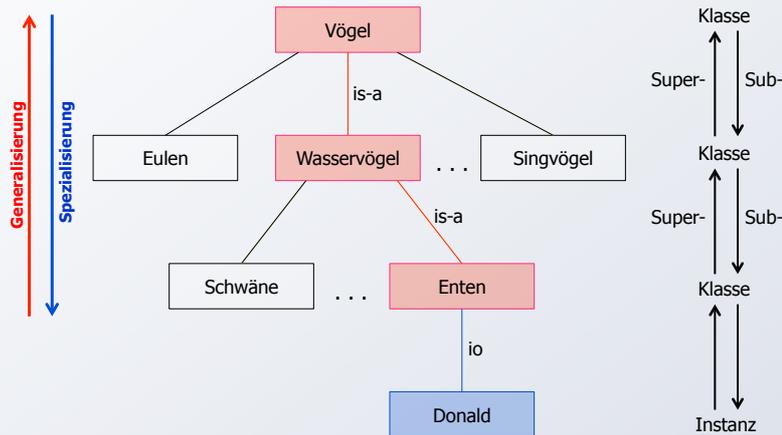


## Modellierungsbeispiel: Instanzdarstellung



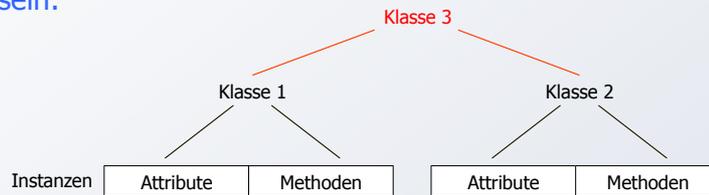
Substitutability: Donald ∈ Vögel

## Modellierungsbeispiel: Typdarstellung



## Generalisierung (3)

- 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**

## Generalisierung (4)

- 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:**

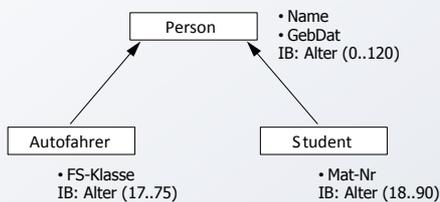
- Student (Autofahrer) ist Subklasse von Person
- Person ist Superklasse von Student und Autofahrer
  - jede Instanz der Subklasse ist immer auch Instanz der Klasse, aber nicht umgekehrt
  - jede Methode, die auf die Instanzen einer Klasse anwendbar ist, ist damit immer auch auf die Instanzen sämtlicher Subklassen anwendbar

⇒ **Garantie von bestimmten Integritätsbedingungen durch das System:**

jeder Student ist auch Person

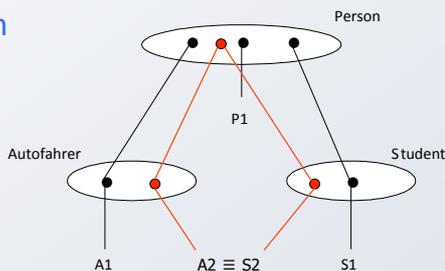
## Generalisierung – Beispiel

### ■ Klassen



↑ Generalisierung

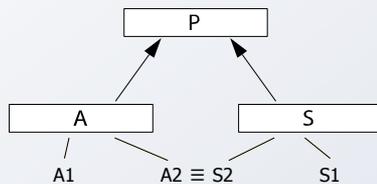
### ■ Instanzen



Subklassen sind i. Allg. nicht disjunkt!

## Überlappende Subklassen – Ansatz 1

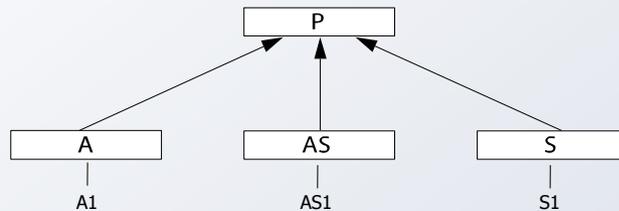
### 1. Mehrklassenmitgliedschaft von Instanzen



Wartung hat durch Programmierer zu erfolgen

## Überlappende Subklassen – Ansatz (2)

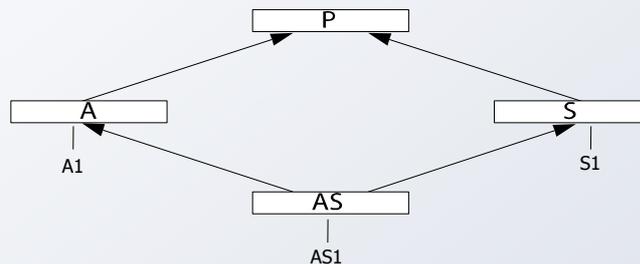
### 2. Einklassenmitgliedschaft von Instanzen (Einfachvererbung)



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

## Überlappende Subklassen – Ansatz (3)

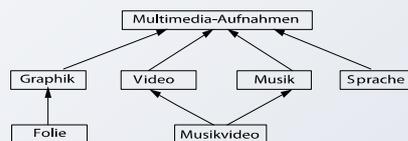
### 3. Einklassenmitgliedschaft von Instanzen (Mehrfachvererbung)



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

## Generalisierung (5)

- 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.
- **Verhalten:** Operationen (auch Methoden genannt)
- ◊ Es müssen **alle Struktur-, Integritäts- und Verhaltensspezifikationen** vererbt werden. Integritätsbedingungen können **eingeschränkt**, Default-Werte können **überschrieben**, Methoden **überschrieben** werden.

### Arten der Vererbung

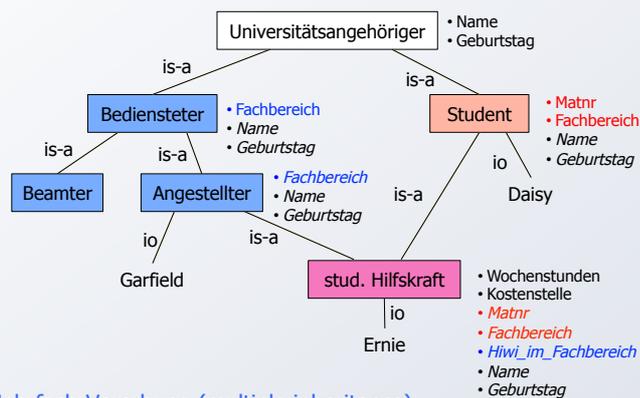
- Einfach-Vererbung (eindeutig)
- Mehrfach-Vererbung

### Schlussweise für Vererbungsregeln:

- **Attributdefinition:**  
HasAttribute (C1, A) ← Isa (C1, C2), HasAttribute (C2, A)
- **Defaultwerte:**  
HasValue (C1, A, V) ← Isa (C1, C2), HasValue (C2, A, V)
- **Substituierbarkeit:**  
P(..., C1, ...) ← Isa (C1, C2), P (..., C2, ...)

## Vererbung (Inheritance)

### Subklasse erbt alle Attribute der Superklasse

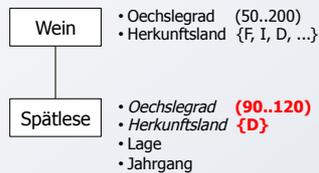


### Mehrfach-Vererbung (multiple inheritance) kann zu Konflikten führen

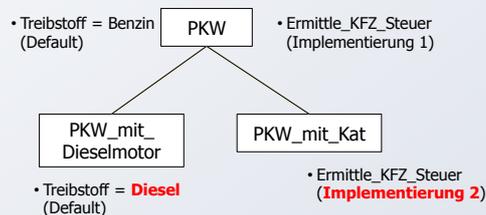
- ◊ **Auflösung explizit durch den Benutzer, z. B. durch Umbenennung:**
  - Fachbereich of Angestellter → Hiwi\_im\_Fachbereich
  - Fachbereich of Student → immatrikuliert\_im\_Fachbereich

## Vererbung (2)

- Subklasse kann den Wertebereich ererbter Attribute einschränken:



- Subklasse kann ererbte Methoden überschreiben oder Defaultwerte bei Attributen ersetzen:



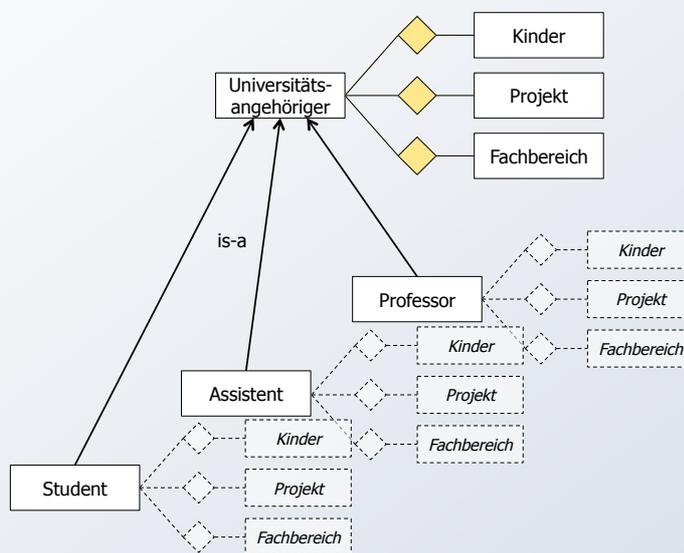
Methoden mit **gleichem Namen und unterschiedlicher Implementierung** (Overriding).  
 Der Begriff **überschreiben** (engl. override, wörtlich außer Kraft setzen, überwinden) beschreibt eine Technik, die es einer Unterklasse erlaubt, eine eigene Implementierung einer geerbten Methode zu definieren. Die überschreibende Methode ersetzt dabei die überschriebene Methode der Oberklasse.

## Vererbung (3)

- Mehrfach-Vererbung / mögliche Lösungen

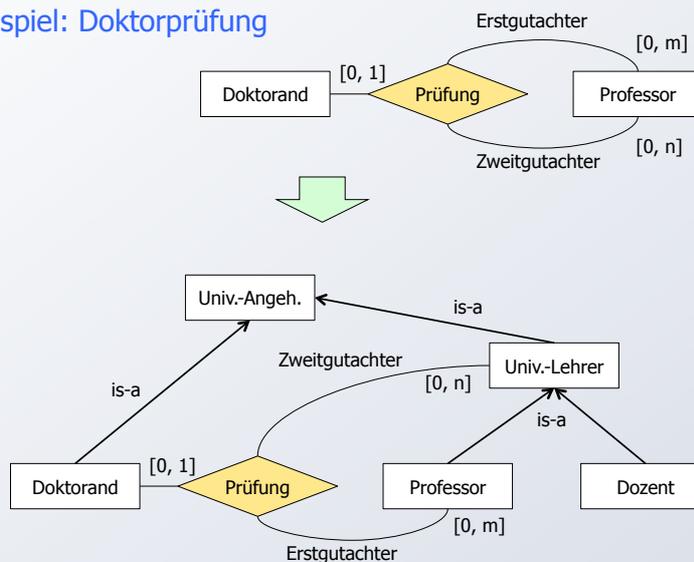
- Attribute:  
Vereinigung; bei Konflikt: Umbenennung
- Wertebereiche (zulässige Werte):  
Schnittmenge
- Defaultwerte: benutzerdefiniert (Ersetzung)
- Integritätsbedingungen (Prädikate):  
benutzerkontrolliert (Spezialisierung)
- Operationen (Methoden):  
benutzerkontrolliert (Überschreiben)

## Vererbung von Beziehungen



## Verfeinerung von Beziehungen

### ■ Beispiel: Doktorprüfung



## Spezialisierung: Definitionen

### Subklasse:

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

$$S \subseteq G$$

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

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

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

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

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

andernfalls **partiell**.

- $Z$  ist **disjunkt**, falls

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

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

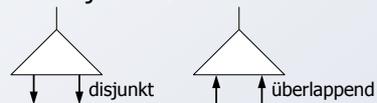
## Arten von Spezialisierungen

### Verfeinerung der is-a-Beziehung

Vollständigkeit:

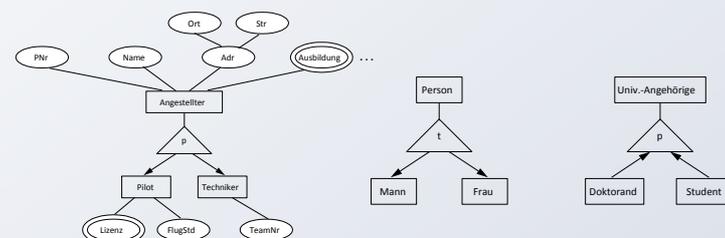


Disjunktheit:



### Beispiele:

- (1) partiell, disjunkt    (2) total, disjunkt    (3) partiell, überlappend

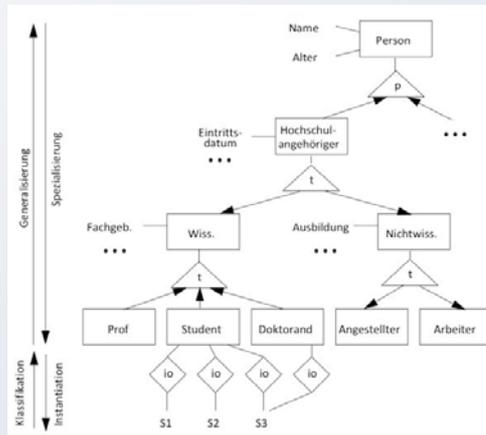


## Abstraktionskonzept: Generalisierung/Spezialisierung

### Nutzung beim objektorientierten DB-Entwurf

Vererbung von Typinformationen

- Strukturdefinitionen:
  - Attribute
  - Defaultwerte
  - konstante Werte
- Integritätsbedingungen:
  - Prädikate
  - Wertebereiche
  - Zusicherungen
- Verhalten:
  - Operationen
- Aspektdefinitionen:
  - Kommentare
  - Einheiten u. a.



## Element-Assoziation

### Aufgabe

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

### Anwendung

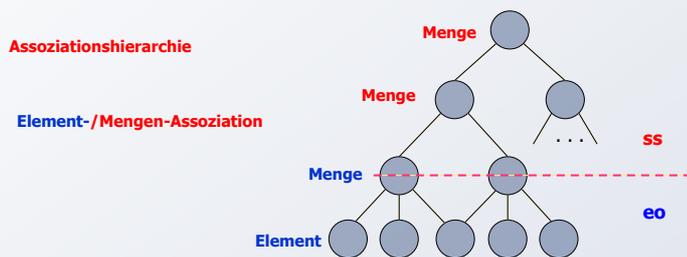
- Element-Assoziation (auch Gruppierung, Partitionierung, Überdeckungs-Aggregation genannt) baut zusammengesetzte (Mengen-)Objekte basierend auf den einfachen (Element-)Objekten auf.
- Sie verkörpert eine 'element-of'-Beziehung ('eo') als 1-Ebenen-Beziehung.
- Es können auch **heterogene Objekte zu einem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

# 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 so-wie auch Teilmenge von mehreren Supermengen
  - ↳ **Netzwerke, (n:m)!**
- **Systemkontrollierte Ableitungen bei der Assoziation**
  - Sie unterstützt keine Vererbung, da die Mengeneigenschaften keine Elementeigenschaften sind
  - Durch **Mitgliedschaftsimplication** lassen sich Eigenschaften bestimmen, die jedes gültige Element der Menge erfüllen muss
  - **Mengeneigenschaften** sind Eigenschaften der Menge, die über Elementeigenschaften abgeleitet sind

# Assoziation

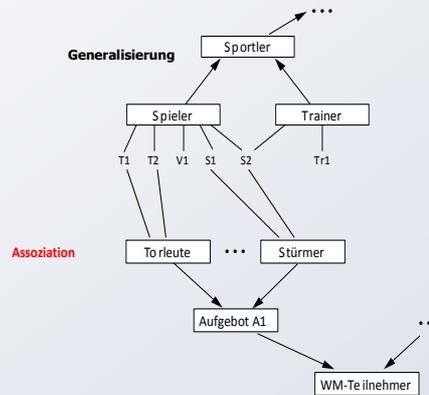
- **Graphische Darstellung:**



## Assoziation – Beispiel

### Operationen

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

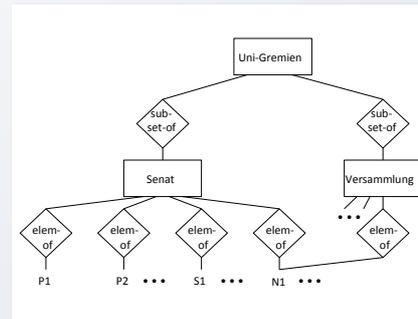


## Abstraktionskonzept: Assoziation

Zusammenfassung einer Menge von Objekten potentiell verschiedenen Typs zu einem semantisch höheren Objekt

Neben der Element-Assoziation ist Mengen-Assoziation möglich

- „element-of“-Beziehung und „subset-of“-Beziehung



- zusätzliche Prädikate (z. B. GEWAEHHLT) 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\*:** *Auto – Motor*  
kein anderes Objekt darf denselben Bestandteil haben
- **oder gemeinsam:** *Tisch – Tischplatte*  
derselbe Bestandteil wird in zwei oder mehr Objekten verwendet
- **Entweder abhängig:** *Kante – Endpunkt*  
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“.

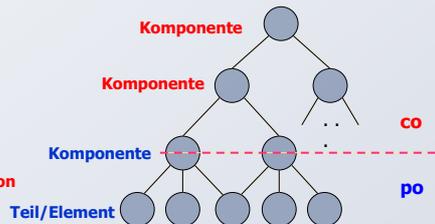
## Element-Aggregation

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

■ **Graphische Darstellung:**

**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

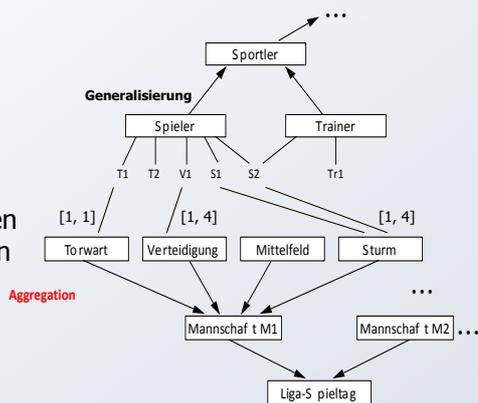
(Aggregation bedeutet auch 'besteht-aus'/'consists-of')

- 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)!)

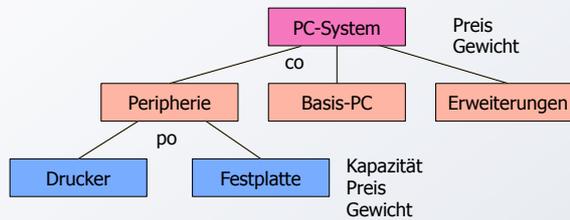
# Aggregation – Beispiel

## Operationen

- Erzeugen/Löschen
  - Create
  - Connect/Disconnect
  - Delete
- Integritätsbedingungen für Aggregatstrukturen
- Suchen (transitive Ableitung von komplexen Objekten)
- Schlussfolgerungen implizierte Prädikate
  - S2 = 10 Tore
  - M1 ≥ 10 Tore



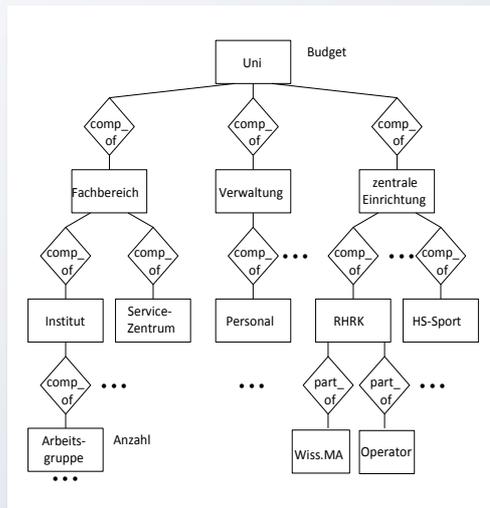
## Aggregation – Beispiel (2)



- **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 – Beispielobjekt

```

Feijoada
instance-of: Hauptgerichte
element-of: brasilianische Spezialitäten
has-components: schwarze Bohnen, Fleisch, Gewürze

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

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

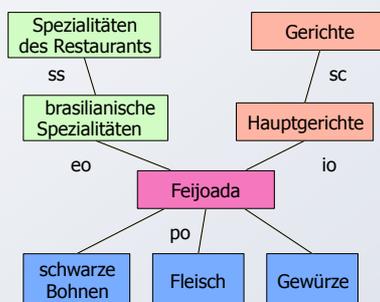
geeignete-Getränke: trockener Rotwein, Bier
possible-values: instance-of Getränke
cardinality
.
.
.

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

## Objektorientierte Repräsentation

### Integration der Abstraktionskonzepte:

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



## Modellinhärentes „Reasoning“

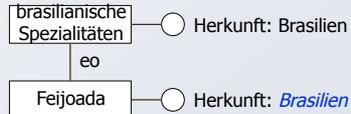
### 3 Abstraktionskonzepte

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

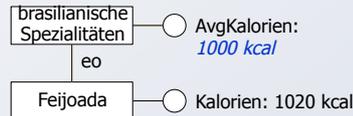
#### Vererbung



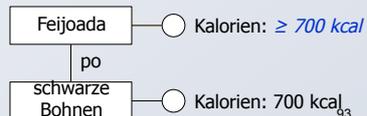
#### Mitgliedschaftsimplicationen



#### Mengeneigenschaften



#### implizierte Prädikate



## Zusammenfassung

### DB-Entwurf umfasst

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

### ERM-Charakteristika

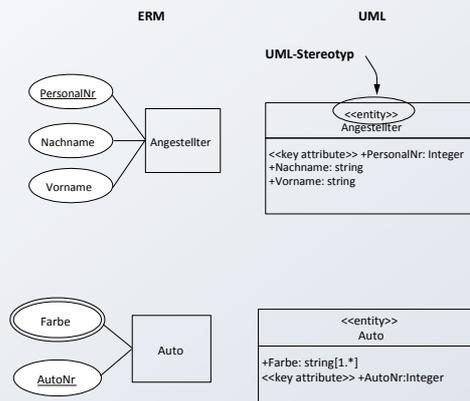
- Modellierung bezieht sich auf die Typebene
  - Relevante Zusammenhänge der Miniwelt werden durch Entity- und Relationship-Mengen modelliert. Sie werden genauer durch Attribute, Wertebereiche, Primärschlüssel/ Schlüsselkandidaten beschrieben
  - Klassifikation von Beziehungstypen dient der Spezifikation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

## 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 (jedoch keine Rollen; sie lassen sich als Mehrklassen-Mitgliedschaften von Instanzen nachbilden)
  - Integritätsbedingungen wurden hier nicht behandelt (-> Relationenmodell)
- Abstraktionskonzepte und deren Implikationen
  - Generalisierung und Vererbung
  - Assoziation mit Mengeneigenschaften und Mitgliedschaftsimplicationen
  - Aggregation und implizierte Prädikate
  - Integration der Abstraktionskonzepte mittels objektzentrierter Darstellung

## Vergleich der Modellierung – ERM und UML

- Entity-Typ
  - Angestellter mit Personalnummer, Nachname und Vorname
- Mehrwertige Attribute
  - Auto mit Farbe(n) und Autonummer



## Vergleich der Modellierung – ERM und UML (2)

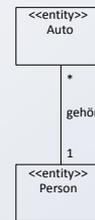
### Relationship-Typ

- Auto gehört Person

ERM

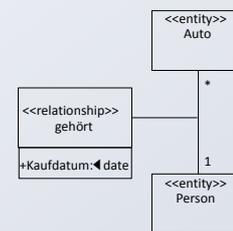
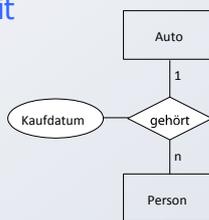


UML



### Relationship-Typ mit Attributen

- Auto gehört Person ab einem Kaufdatum

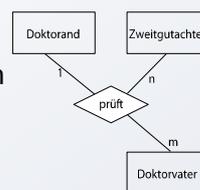


## Vergleich der Modellierung – ERM und UML (3)

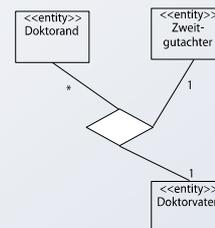
### n-stelliger Relationship-Typ

- Professoren prüfen Studenten über Vorlesungen an Datum

ERM

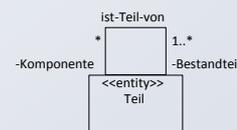
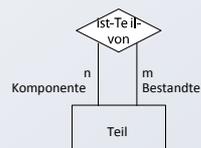


UML



### Relationship-Typ auf einem Entity-Typ

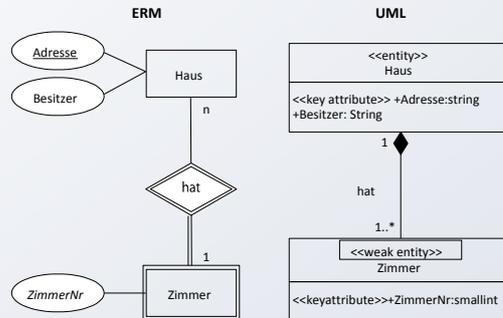
- Stücklistenbeziehung: Teile sind Teile von Teilen



## Vergleich der Modellierung – ERM und UML (4)

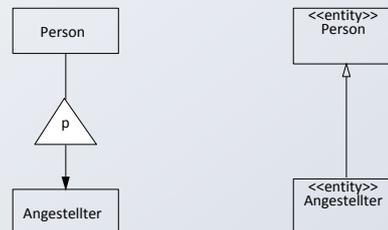
### Existenzabhängigkeit

- Ein Haus hat eine Adresse und einen Besitzer und hat mehrere Zimmer



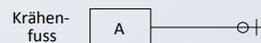
### Generalisierung

- Angestellte sind Personen



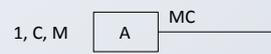
## Notationen

- Viele Systeme erlauben als Kardinalitätsrestriktionen nur 0, 1 oder n sowie ausschließlich binäre Beziehungen
- Jedes Element von A nimmt an genau einer Beziehung teil | höchstens einer Beziehung teil



## Notationen (2)

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



## Notationen – Beispiel

- 1:n

- n:m

