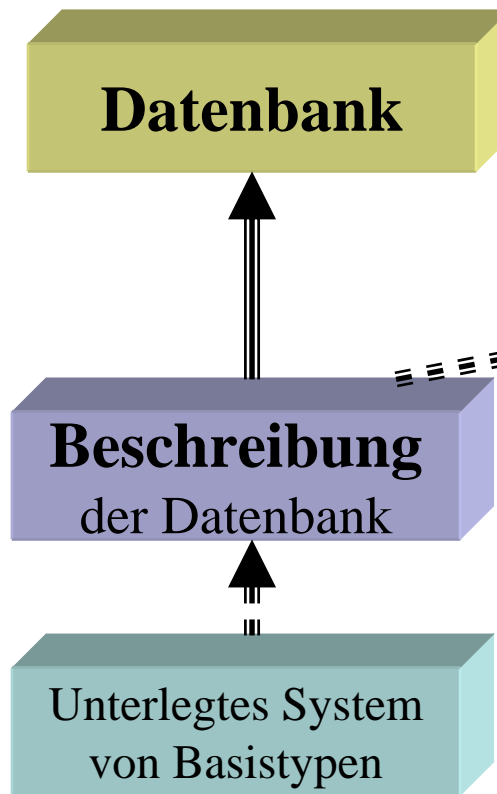


## 2. Datenbankmodelle

3 ★ Schichten  
einer Datenbank

**Data  
Dictionary**



integer, real, float, decimal,  
boolean, date, time

Konzeptionelle Schicht:  
(E)ER-Modell  
Logische (Implem.-)Schicht:  
relationales,  
hierarchisches  
Netzwerk-Modell  
Implementationsschicht:  
Physische Modelle

I.a. zur Modellierung von Struktur (+ statische Semantik) und  
Verhalten (Operationen, dynamische Semantik);  
daneben wären noch Interface-Modelle zu betrachten (mit Integration).

## Datenbankmodelle

### *A) Strukturmodelle, statische Semantik, generische Operationen*

#### 1. Klassische, implementierte Modelle

**Relationales Modell** (Relationen mit Tabellendarstellung) 

dominierendes Modell

Hierarchisches Modell (Listen mit hierarchischen Verweisen) 

Netzwerkmodell (Listen mit Graphen-Verweisen) 

#### 2. **Entity-Relationship-Modelle**

Relationentürme

### *B) Objektmodelle (Struktur, Operationen, Semantik)*

#### 1. Objektorientierte Modelle

#### 2. Verallgemeinerungen von ER-Modellen

#### 3. Objektrelationale Modelle

### *C) Prozeßmodelle*

#### 1. Workflow-Modelle

#### 2. Modelle aus dem Software-Engineering z.B.UML

# Allgemeines Herangehen (Beispiel Strukturaufbau)

## Konstruktion nach **axiomatischer Methode**

### 1. Bestimmung von **Elementareinheiten**

Integration der Einheiten in die (HW/SW)-Infrastruktur (Implementationstyp)

(Modifikations-, Durchmusterungs- ... operationen, Defaultwert, Nullwert, Repräsentation)

### 2. Definition von **Typenkonstruktoren**

Konstruktor (Menge, Tupel, Liste, Multimenge, Funktion, ...)

*mit Selektoren auf Komponenten, Modifikations-, Suchoperationen,  
Korrektheitskriterien, Default-Regeln, Nullwert,  
Benutzer-Repräsentationstypen, Implementationstypen*

**Induktive Definition** von Typen mit unterlegten Typen,  
Konstruktoren und  
Repräsentationstypen

**Objekte** haben einen (oder mehrere) Typen  
repräsentieren evt. Dinge der Realität oder Abstraktionen

Zusammenfassung von Objekten in **Klassen**

**Attribute über Basistypen  $T = \{D_i \mid 1 \leq i \leq n\}$  und Namensmenge  $N$**

**1. Atomare Attribute  $A \in N$**

$\text{dom}(A) \in T$

**2. Induktive Attributdefinition  $B \in N$  über Konstruktoren  
und komplexen Attributen  $A_1, \dots, A_m$**

$B(A_1, \dots, A_m) \quad \text{dom}(B) = \text{dom}(A_1) \times \dots \times \text{dom}(A_m)$

$B\{A_1\} \quad \text{dom}(B) = \emptyset (\text{dom}(A_1))$

$B\langle A_1 \rangle \quad \text{dom}(B) = \text{list}(\text{dom}(A_1))$

**3. Kollektionen: variable Anzahl von Komponenten**

$[], \{\}, \{\{\}\}$

Operationen auf Basistypen analog erweitert

Name(Vornamen<Rufname, Vorn1, Vorn2>, FamName, [GebName,] Titel)

$\langle \text{str}_{15} \times \text{str}_{15} \times \text{str}_{15} \rangle \times \text{str}_{25} [ \times \text{str}_{25} ] \times \text{str}_4$

(<Alex, Paul, Peter>, Buchmann, Müller, Dr.)

Adresse(PLZ, Ort, Straße, Hausnummer [,Etage [,Richtung]])

(03046, Cottbus, K.-Marx-Str. 17)

Mengen

bestehen aus unterscheidbaren Elementen.

Bei Anwendung des Mengenkonstruktors auf eine Kollektion entsteht eine Menge, deren Elemente unterscheidbar sind.

Mengen mit Tupelstruktur

$$M = \{ (1,a,A), (1,b,A), (2,a,B), (2,c,B), (2,d,A), (3,a,C), (3,b,B) \}$$

Elemente sind identifizierbar über ihre Werte

d.h. über 1.+2.+3. Komponenten

Elemente sind identifizierbar über einige ihrer Komponenten,

z.B. in M über 1.+2. und 2.+3. Komponenten,

nicht aber über 1.+3. Komponenten  $|\{ (1,.,A) \}| > 1$

Identifikationskomponenten nennt man **Schlüssel**

Von Interesse sind **minimale Schlüssel** .

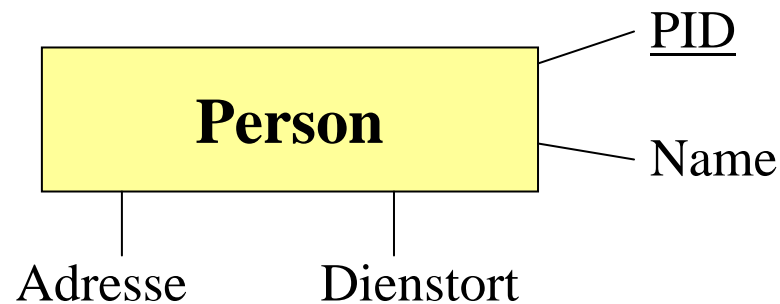
Ausgezeichneter minimaler Schlüssel: **Primärschlüssel**

## Entity-Typ $E$ für Entity-Klasse $C(E)$ (Mengensemantik)

1. Menge von (komplexen) Attributen
2. Teilmengen davon als Schlüssel  
ein Schlüssel: Primärschlüssel
3. Weitere semantische Integritätsbedingungen

$$E = ( \{A_1, \dots, A_n\}, \{ \underline{K_1}, K_2, \dots, K_m \}, \Sigma )$$

$$\text{Person} = ( \{ \text{Name}, \text{PID}, \text{Adresse}, \text{Dienstort} \}, \{ \{ \underline{\text{PID}} \}, \{ \text{Name} \} \}, \emptyset )$$



Mengen mit Tupelstruktur z.B.

über  $R = S \times T \times U$  definiert über  $\text{int} \times \text{kleinB} \times \text{großB}$

z.B.  $M = \{ (1,a,A), (1,b,A), (2,a,B), (2,c,B), (2,d,A), (3,a,C), (3,b,B) \}$

Kardinalität von Teilstrukturen

Element 1 kommt in M zweimal vor, ebenso wie 3 und b; 2 dreimal  
d.h. Elemente aus S zwei- bis dreimal

formal:

$\text{comp}(R,S) = (2,3)$  als gültige Bedingung ebenso wie  $(1,4)$

$\text{comp}(R,T) = (2,3)$  gilt dagegen für M nicht

Kardinalitätsbedingung (in der participation-Semantik)

(lookup-Semantik ist dagegen low-level, unübersichtlich, falsch interpretierbar)

Bei Darstellung einer Anwendung werden die Bedingungen  
(Integritätsbedingungen) erfaßt, die in allen möglichen Klassen  
gelten.

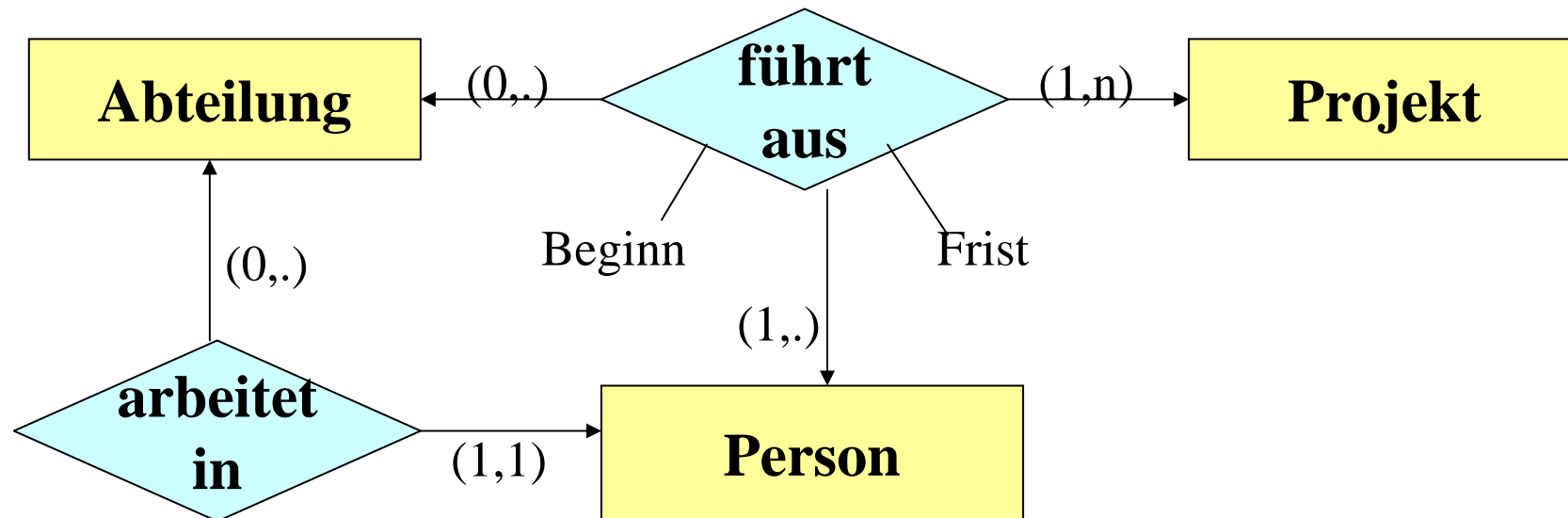
# Relationship-Typ R für Relationship-Klasse C(R) (Mengensemantik)

## 1. Komponenten

- a) Entity-Typen (evt. mit Rollen)
- b) (Komplexe) Attribute

## 2. Integritätsbedingungen

Kardinalitätsbeschränkungen (participation)



Anmerkung: oft nur binäre Relationship-Typen zugelassen

**(Fehler der meisten OO-Methoden)**

(0,.) bedeutet keine Information: kann deshalb auch weggelassen werden

(1,.) ist bessere Notation als (1,n) (letztere wird jedoch meist verwendet)

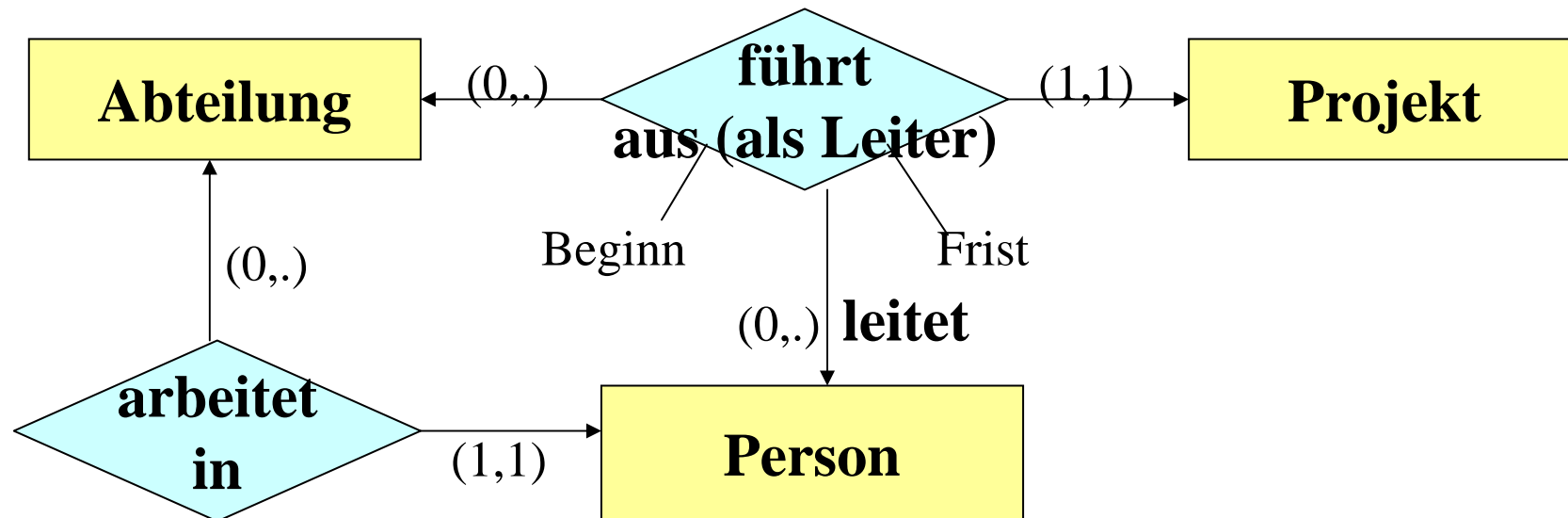
# Relationship-Typ R für Relationship-Klasse C(R) (Mengensemantik)

## 1. Komponenten

- a) Entity-Typen (evt. mit Rollen)
- b) (Komplexe) Attribute

## 2. Integritätsbedingungen

Kardinalitätsbeschränkungen (participation)



**Anmerkung: Rollen werden als Markierung der Komponenten zugelassen  
Kardinalitäten haben sich geändert**

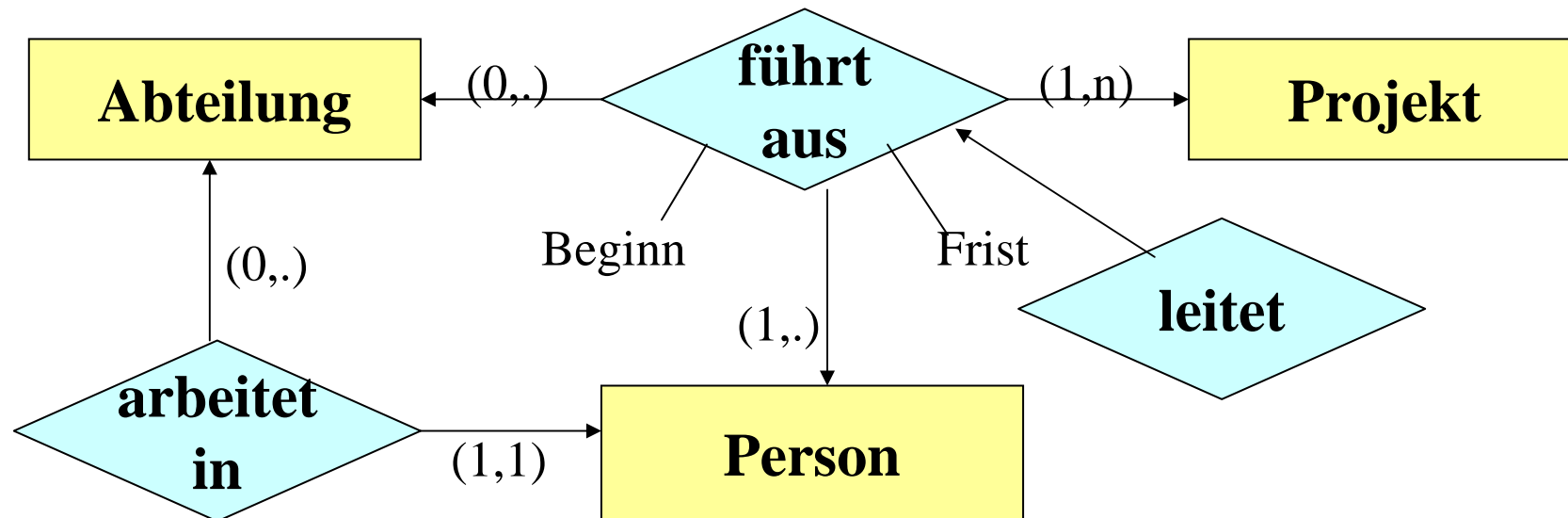
# Relationship-Typ R für Relationship-Klasse C(R) (Mengensemantik)

## 1. Komponenten

- a) Entity-Typen (evt. mit Rollen)
- b) (Komplexe) Attribute

## 2. Integritätsbedingungen

Kardinalitätsbeschränkungen (participation)



Anmerkung: unärer Relationship-Typ als Rolle

Schema gilt nur, wenn Leiter auch im Projekt mitwirkt

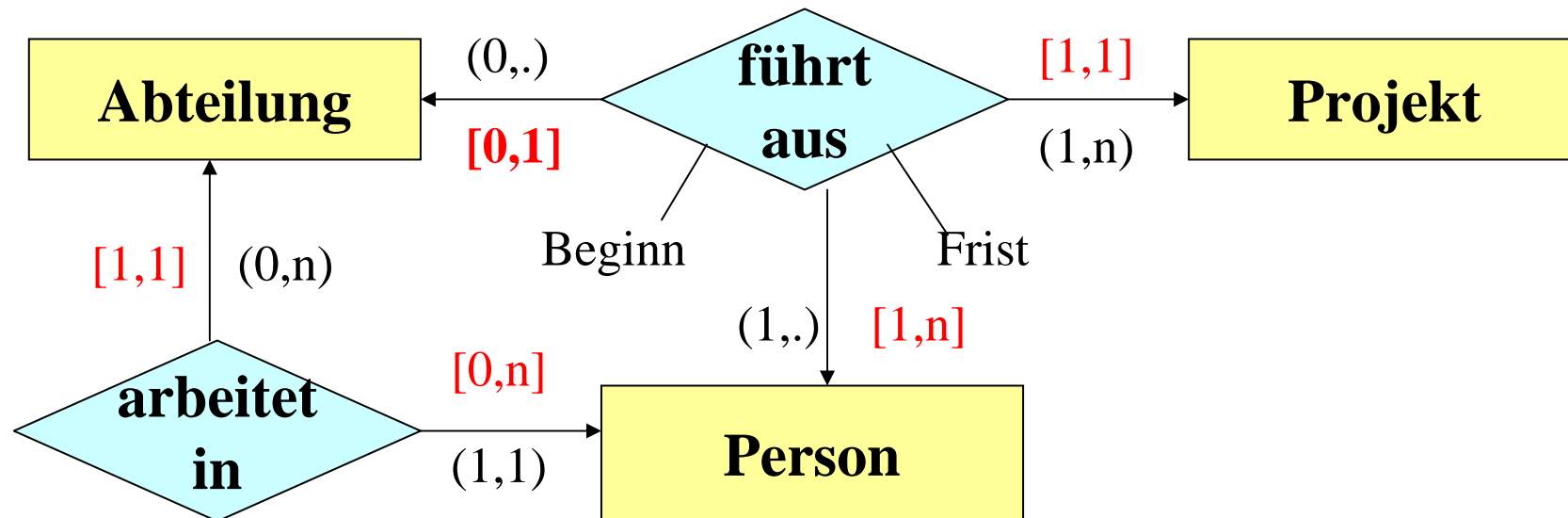
# Relationship-Typ **R** für Relationship-Klasse **C(R)** (Mengensemantik)

## 1. Komponenten

- a) Entity-Typen (evt. mit Rollen)
- b) (Komplexe) Attribute

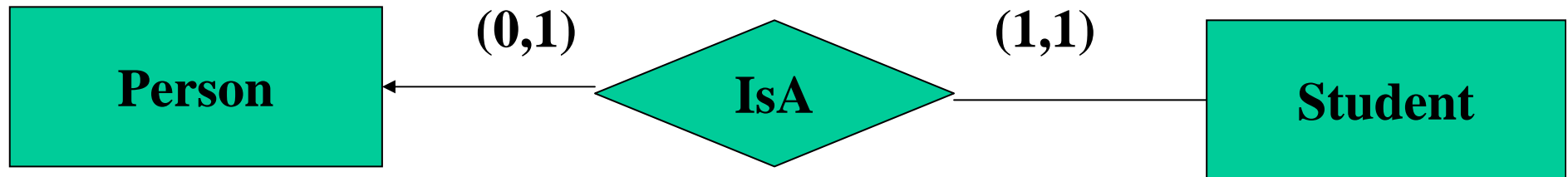
## 2. Integritätsbedingungen

Kardinalitätsbeschränkungen (**lock-up-Semantik**)



Anmerkung: Lookup-Semantik ist für den binären Fall nur die Umkehrung im n-ären Fall ( $n > 2$ ) jedoch anders

## Spezieller Relationship-Typ: Is-A-Beziehung



**Semantik:** vollständige Vererbung der Struktur  
strukturelle Verfeinerung  
vollständige Vererbung der Operationen  
operationale Verfeinerung  
i.a. Vererbung aller Beziehungen  
Kardinalitätsbeschränkungen (0,1), (1,1)  
damit hierarchische Beziehung

Erzwingung von Operationen:

Objekt Student kann gelöscht werden in Studentenkategorie unabhängig von Personenklasse, aber nicht eingefügt werden

## Weitere Konstrukte:

**Union  $\oplus$  : Vereinigung mit Identifikationsforderung**

**Komposition mit Ausschluß- und Existenzbedingung**

consists-of, part-of, has, ((0,..),(1,1))

**Aggregationen: Liste, Multimenge**

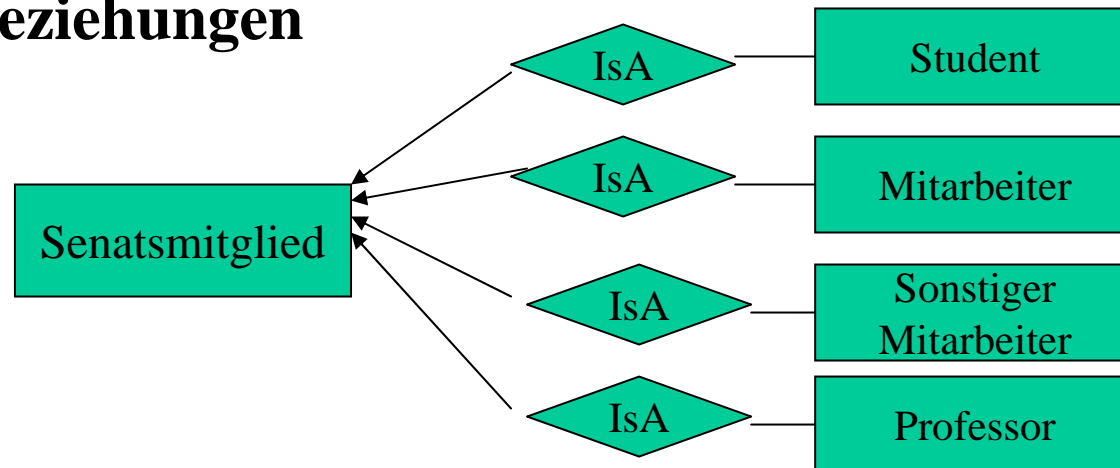
**Relationship-Typen mit Zusatzidentifikation**

**Abgeleitete Attribute (*hierarchische Definition*)**

**Optionale Typen (im Diagramm mit  $\ominus$ , im Schema mit [])**

## Spezielle Integritätsbedingungen

### Exklusivsbeziehungen



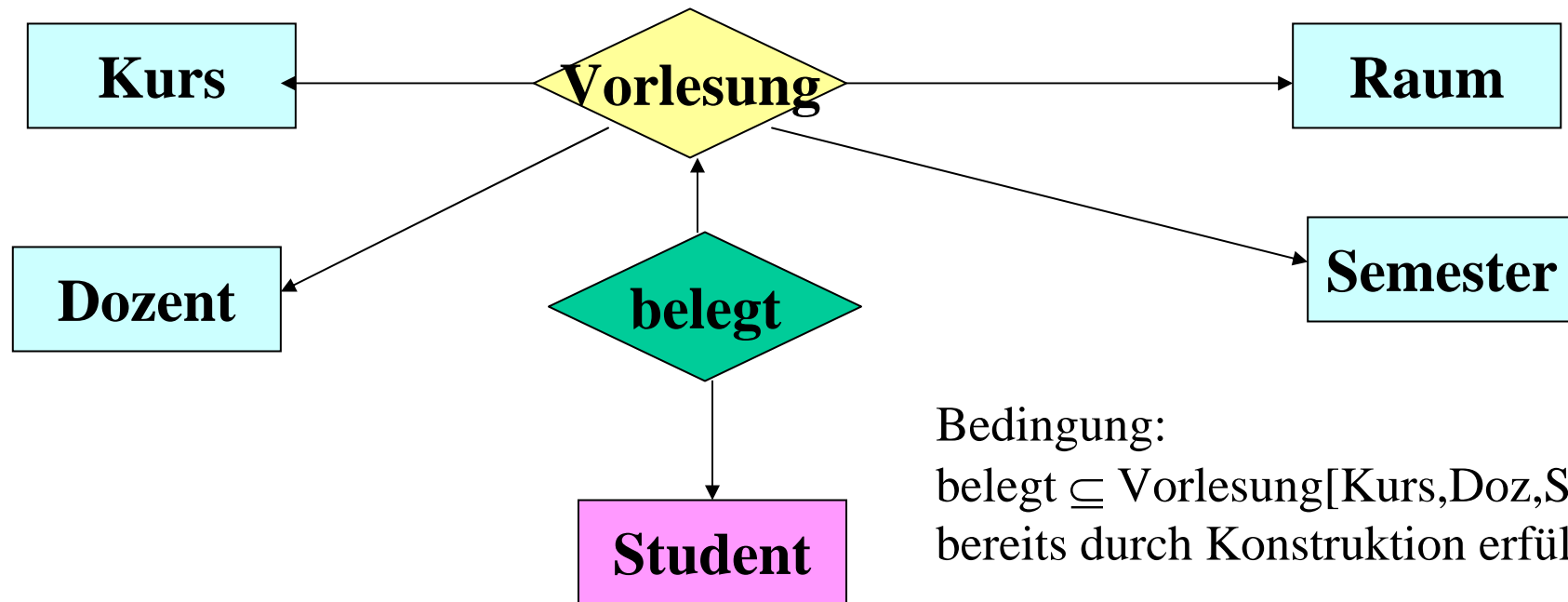
## Relationship-Typen i-ter Ordnung

1. Entity-Typen sind Relationship-Typen 0. Ordnung  
Relationship-Typen über Entity-Typen sind 1. Ordnung

2. Relationship-Typ i-ter Ordnung

Komponenten: Relationship-Typen der Ordnung  $< i$   
Attribute

Integritätsbedingungen



# **Entity-Relationship-Modell mit Mengensemantik**

**Relationship-Typen erben Identifikation ihrer  
Komponenten**

**mitunter auch mit Pointersemantik  
insbesondere für Relationship-Typen**

**Gesamtheitsregel:** abschließende Beschreibung

**Verneinungsregel:** negation as failure, closed-world-assumption

**Sichtregel:** Ist Teil aus  $R_1, \dots, R_n$  erschlossen, dann auch in Gesamtheit.

**Pragmatik: Entity-Klassen als Kernklassen**

**Rollen durch (unäre) Relationship-Typen**

Mehrfachklassifikation ohne OID-Vererbung

## **Spezielle Modellierungsbedingungen** des ER-Modelles

1. Entity-Typen mit mindestens einem Attribut
2. Jedes Attribut mindestens einmal verwendet
3. Attributtypen definiert über Konstruktor-Definition
4. Basis-Attribut-Typen haben Datentyp.
5. Schlüssel gehören zu Attributen (Komponenten) des (Entity-)Typen
6. Jeder Relationship-Typ hat mindestens eine Komponente
7. Typstruktur ist streng hierarchisch
8. Jeder Entity-Typ hat mindestens einen Schlüssel
9. Schlüssel haben mindestens ein Element
10. Elemente eines Schlüssels sind Komponenten des Typs.
11. Mengenbasierte Definition
12. OID ist nur zusätzliche Identifikation. Identifikation kann auch über Assoziationen erhalten werden.
13. Spezielle Behandlung rekursiver Typen

## Operationen über Typen $R = (\text{compon}(R), \text{keys}(R), \Sigma(R))$

1. **Projektion**  $\Pi_X(R) = (\text{comp}(R) \cap X, \{K \cap X \mid K \subseteq X \wedge K \in \text{keys}(R)\}, \{Z \rightarrow Y \mid Z, Y \subseteq X, Z \rightarrow Y \in \Sigma(R)\})$

$$\Pi_X(C(R)) = \{t[X] \mid t \in C(R)\}, \quad t[X] = (t_{A_1}, \dots, t_{A_n}), \quad X = \{A_1, \dots, A_n\}$$

2. **Verbund** (Konjunktion der Prädikate) (Es entsteht neuer Typ aus beiden Typen.)

$$C(R_1) * C(R_2) = \{t \mid \exists t' \in C(R_1) \wedge \exists t'' \in C(R_2) : t'[comp(R_1)] = t[comp(R_1)] \wedge t''[comp(R_2)] = t[comp(R_2)]\}$$

3. **Selektion** (Herausschneiden von Tupeln)

$$\sigma_\alpha(C(R)) = \{t \in C(R) \mid \alpha(t) = \text{true}\}$$

Selektionsbedingung: 1.  $\alpha \equiv A = a, \quad a \in \text{dom}(A), A \in \text{comp}(R)$

2.  $\alpha \equiv A = B, \quad A, B \in \text{comp}(R), \text{dom}(A) \cong \text{dom}(B)$

3.  $\alpha \equiv \alpha_1 \wedge \alpha_2, \quad \text{analog } \vee, \neg$

4. **Mengenoperationen** über Klassen gleichen Typs

Vereinigung, Durchschnitt, Differenz

5. **Umbenennung** Komponente A wird umbenannt zu neuer Komponente B mit gleichem Wertebereich

Achtung: Nach Anwendung der Operationen müssen Integritätsbedingungen nicht mehr gelten.

## Spezielle ER-Modelle

1. **Relationales Modell:** nur Entity-Typen mit atomaren Attributen

Vorausgesetzt wird Mengensemantik im ER-Modell.

Bei Verzicht auf Verbot von atomaren Attributen: **genestetes relationales Modell.**

2. **Hierarchisches Modell:** Entity-Typen mit Listensemantik

binäre  $((1,1),(0,n))$ -Relationship-Typen erster Ordnung mit Pointersemantik

Darstellung als Wald (Menge gerichteter Bäume) mit virtuellen Knoten

3. **Netzwerkmodell:** Entity-Typen(Record-Typ (logical record)(Listensemantik),

binäre  $(0,1)(0,.)$  Relationship-Typen (Link oder Set-Typ) mit Pointersemantik

Darstellung als gerichteter Graph

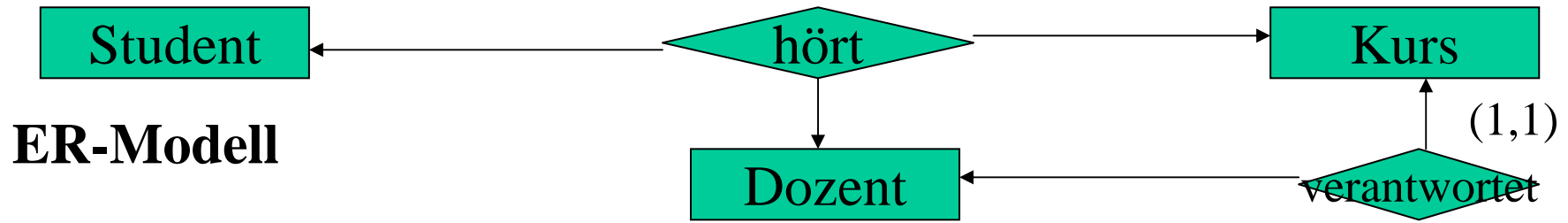
4. **Binäres ER-Modell:** nur binäre Relationship-Typen erster Ordnung

**SERM (Structured ER Modell):** nur binäre  $((1,1),(0,.)$ -Relationship-Typen

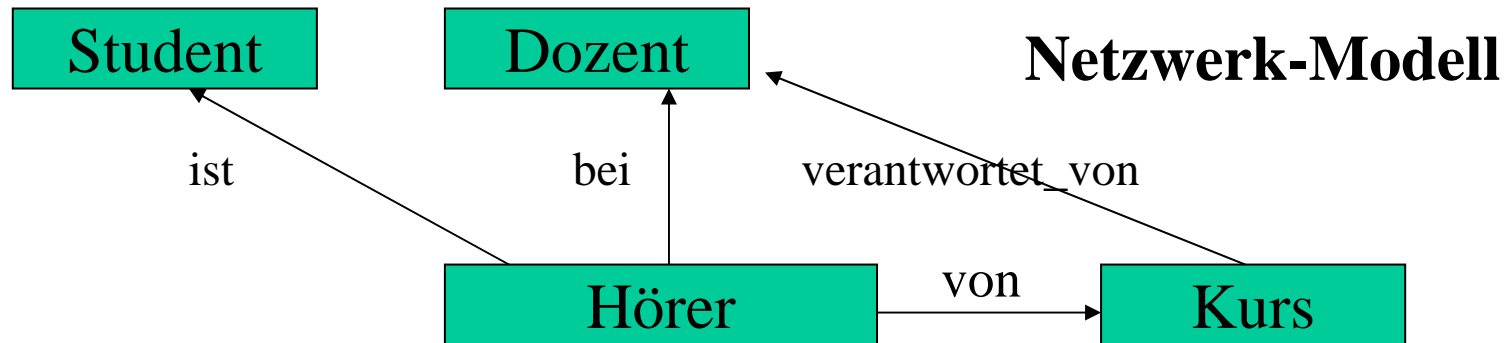
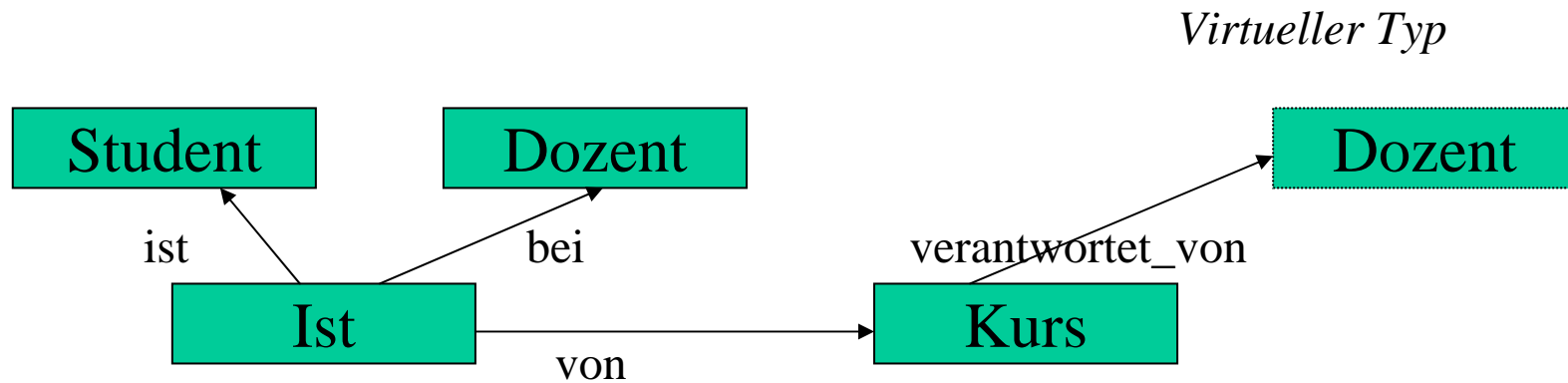
erster Ordnung, Diagramme mit Pfeilen nur von-links-nach-rechts und

von-oben-nach-unten

5. **ORM (Object-Role-Model):** keine Entity-Typen



**Hierarchisches Modell**



# Abbildung von ER-Konstrukten auf andere Modelle

**ER → Relational:** Verflachen der Attribut-Typen  
Entity-Typen in relationale Typen,  
Relationship-Typen entweder eigenständig oder  
durch Einlagerung ((1,1)-Rolle) schrittweise über gesamte Hierarchie  
Speziell bei rekursiven Typen, IsA-Typen, Generalisierung, Spezialisierung

**ER → Netzwerk:** Verflachen der Ordnung,  
Attribute in Relationship-Typen überführen  
Binarisierung aller n-stelligen Relationship-Typen  
Umwandlung in gerichtete binäre Relationship-Typen

**ER → Hierarchisch:** wie bei der Übersetzung ins Netzwerk-Modell,  
Herstellen von Baumstrukturen durch Einführung virtueller Typen

**ER → XML:** wie bei der Übersetzung ins hierarchische Modell,  
Herstellen von Baumstrukturen durch Einführung virtueller Typen