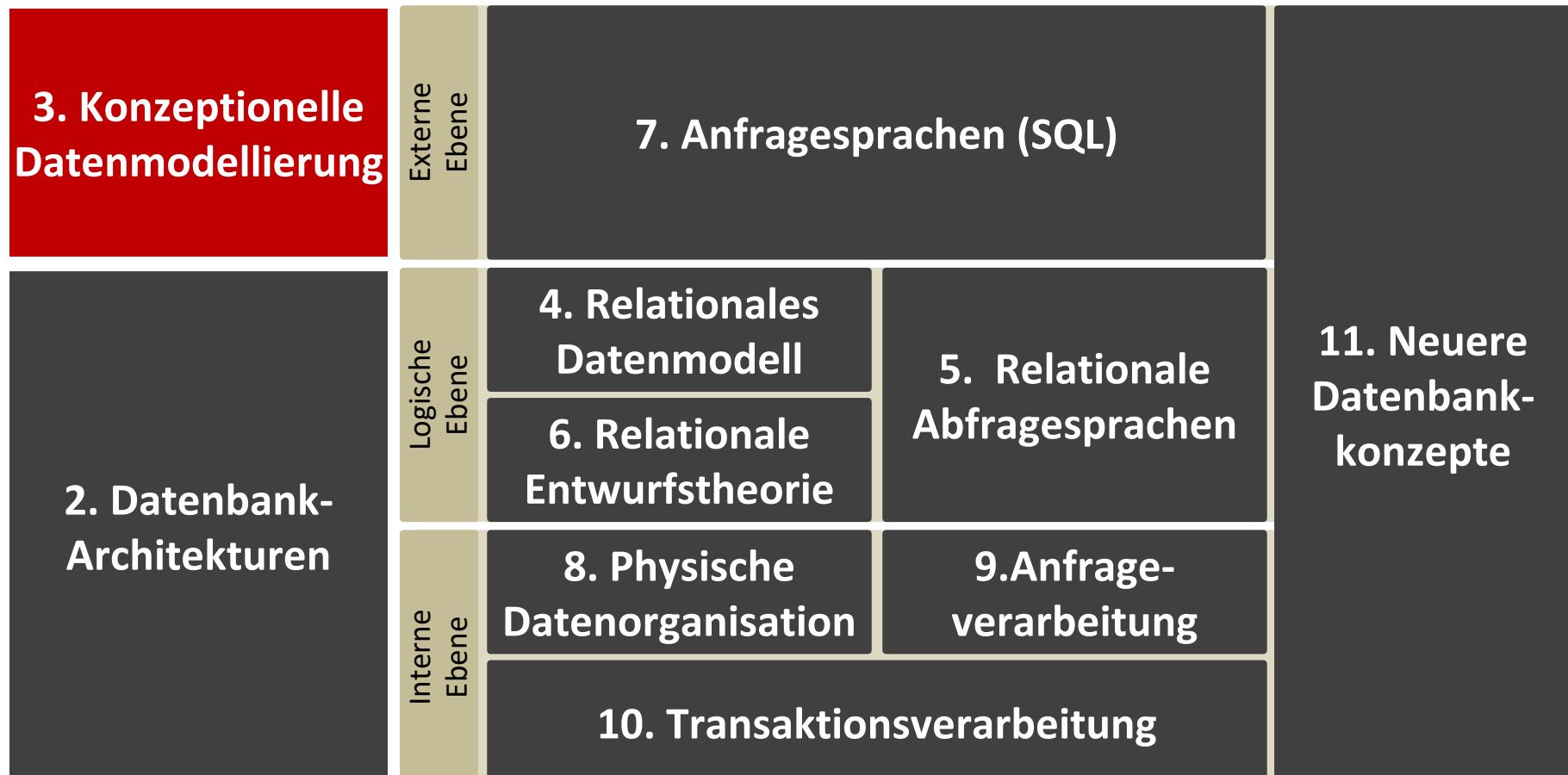


Informationsmanagement

Sommersemester 2019

Kapitel 3: **Konzeptionelle Datenmodellierung**

Themenüberblick (Teil 1)



1. Grundlagen des Informationsmanagements

Datenmodellierung

Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- (min, max)-Notation
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

Datenmodellierung

Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- (min, max)-Notation
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

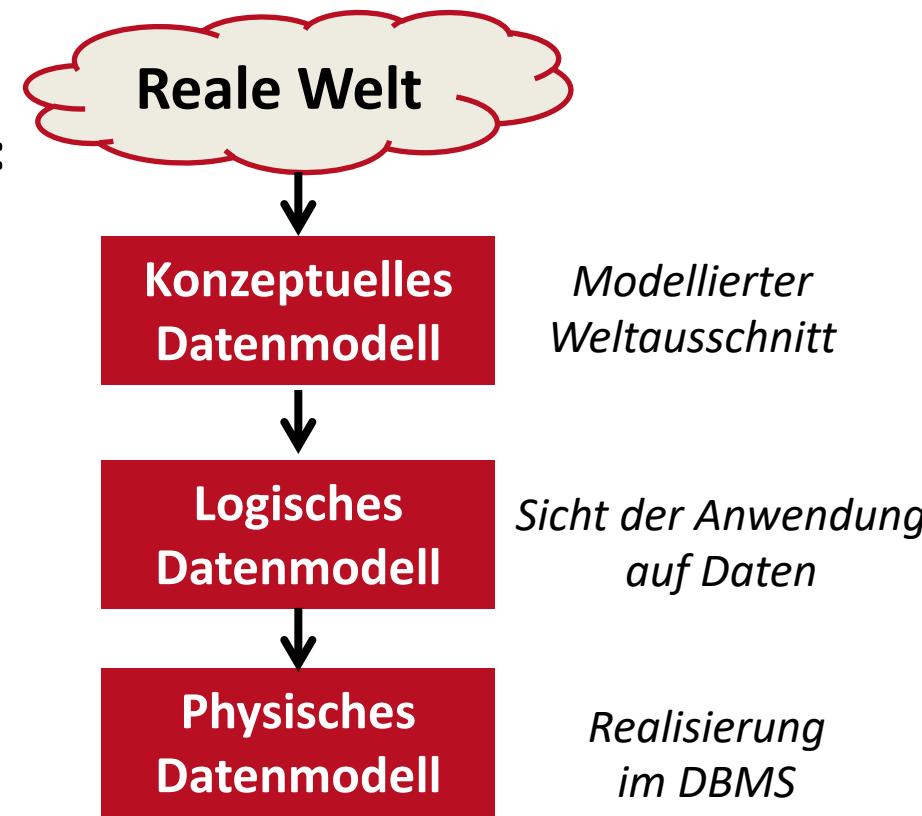
Datenmodell nach Abstraktionsebene



Datenmodell: Abstrakte Darstellung eines **Ausschnitts** der (realen) Welt mittels Daten

Unterschiedliche Abstraktionsebenen:

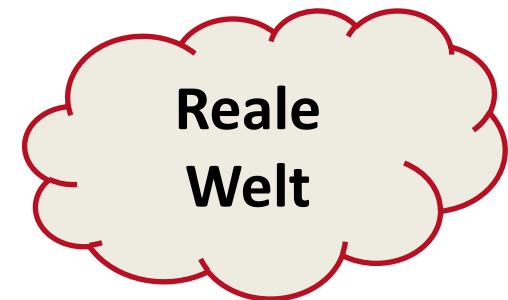
- **konzeptuell** (fachlich motiviert)
- **logisch** (DBMS-spezifisch, geräteunabhängig)
- **physisch** (geräteabhängig, technisch motiviert)





Beispiel: Vorlesung „Informationsmanagement“

- Titel der Vorlesung? CPs?
- Uhrzeiten (Begin, Ende)?
- Professor(en)?
- Raumnummer? Raumgröße?
- Dauer jeder einzelnen Vorlesung?
- Raumtemperatur pro Vorlesung? ...



Alles zu speichern ist kaum möglich und nicht sinnvoll!

We betrachten nur eine Weltausschnitt („**Modellierungswelt**“) von relevanten Objekten und Eigenschaften

Konzeptuelle Datenmodellierung

Kernfragen:

- Welche Objekte der realen Welt sollen in DB gespeichert werden?
- Zu welchen Konzepten (= Objekttypen) lassen sich die Objekte zusammenfassen?
- Welche Merkmale (=Attribute der Objekte) sollen gespeichert werden?
- Wie hängen die Objekte zusammen?
- Welche „Regeln“ müssen gelten?

Typische Modelle: ER-Diagramme, UML-Klassendiagramme

Datenmodellierung

Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- (min, max)-Notation
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

Datenmodellierung

Entity-Relationship Diagramme

- **Basiskomponenten**
- Funktionalitäten
- (min, max)-Notation
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

Entity/Relationship Diagramme (ERD)

- Einfacher Formalismus zur **konzeptuellen Modellierung** von DBs
- Eigenschaften:
 - Fokus auf **statischen Eigenschaften** (Struktur)
 - **keine dynamische Eigenschaften** (Funktionen / Methoden)

Beliebt, da oft auch **für Laien verständlich**

- Standardmodell in **frühe Entwurfsphasen (Diskussion mit Endanwendern über Anforderungen -> Software-Engineering)**
- Textuelle und graphische Darstellung (**ER-Diagramm**)

ERD - Modellierungselemente

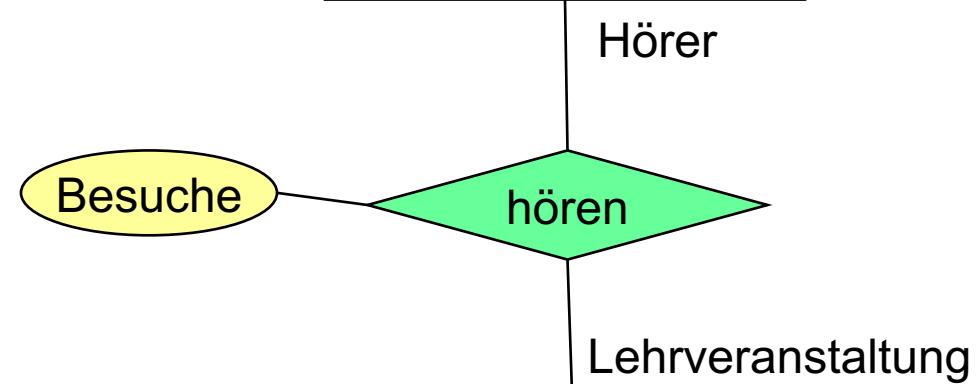


TECHNISCHE
UNIVERSITÄT
DARMSTADT

Entity (Entitätstyp, Gegenstandstyp)



Relationship (Beziehungstyp)



Attribut (Eigenschaft)

Schlüsselattribute (Identifikation)

Rolle (Beziehungsbeschreibung)

-> werden häufig nicht modelliert



Entität (*Entity*)

- Individuell identifizierbare Objekte der modellierten Welt
- *Beispiele: Carsten Binnig, Informationsmanagement (IM),...*

Entitätstyp (*Entity type / Entity set*)

- Gemeinsamer Typ von Objekten mit gleichen Merkmalen (gebildet durch Klassifikation)
- *Beispiele: Professor, Vorlesung, ...*
- Mathematisch als Menge darstellbar: *Carsten Binnig ∈ Professor*



Textuelle Notation

- $E(\dots)$

Professor(...)

Graphische Notation

- als Rechteck

Professor



Beziehung (*Relationship*)

- Beziehung zwischen zwei oder mehreren Entitäten,
- *Beispiel:* „Carsten Binnig“ **hält-Vorlesung** „IM“

Beziehungstyp (*Relationship type / Relationship set*)

- Menge von Beziehungen gleichen Typs
- *Beispiel:* die „hält-Vorlesung“-Beziehungen zwischen Prof. und Vorl.
- Mathematisch: n -stellige Relation $R \subseteq E_1 \times E_2 \times \dots \times E_n$
- *Beispiel:* $(\text{Carsten Binnig}, \text{IM}) \in R_{\text{hält-Vorlesung}} \subseteq \text{Professor} \times \text{Vorlesung}$

Beziehungstypen (BT)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

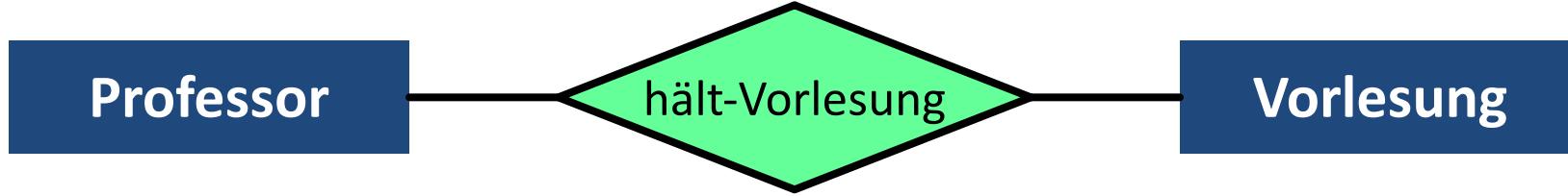
Textuelle Notation

- $R(E_1, E_2, \dots, E_n)$

hält-Vorlesung(Professor, Vorlesung)

Graphische Notation

- als Raute



Attribute



Textuelle Notation

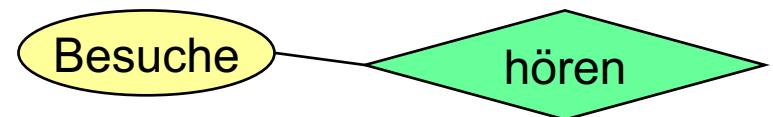
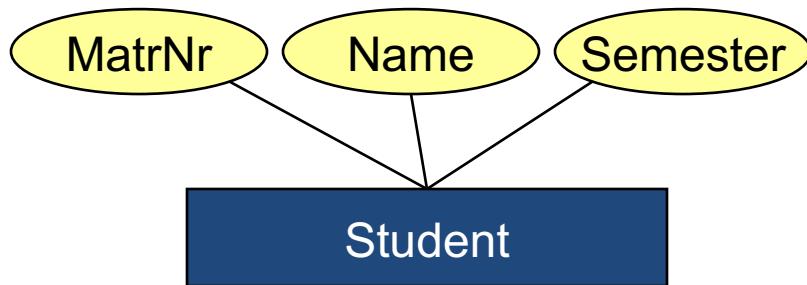
- $E(A_1, A_2, \dots, A_n)$
- $R(E_1, E_2, \dots, E_m; A_1, A_2, \dots, A_n)$

Student(Matrikelnummer,
Name, Semester)

hören(Student, Vorlesung;
Besuche)

Graphische Notation

- als Ovale

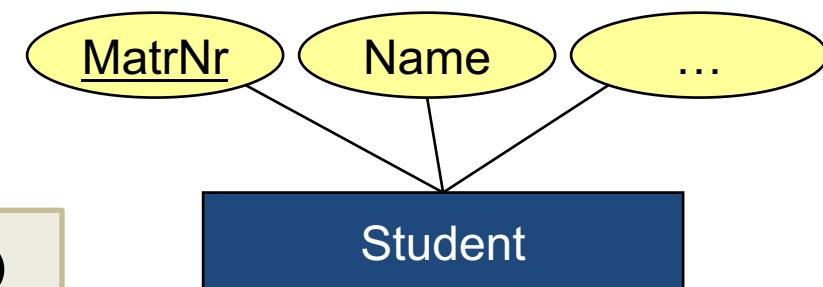


Schlüsselattribute (Keys)

- Teilmenge der Attribute, die eine Entität eindeutig identifiziert
- Für $E(A_1, A_2, \dots, A_n)$ gilt: Schlüsselattribute $\{S_1, \dots, S_k\} \subseteq \{A_1, \dots, A_n\}$
- *Beispiel: Matrikelnummer (nicht jedoch Name) kann Schlüsselattribut von Student sein*
- Beziehungstypen haben keine Schlüsselattribute!

ER-Diagramm:

- Unterstreichen der Attribute



Student(Matrikelnummer, Name,...)

Schlüsselattribute: Weitere Beispiele

- **Buch:** {ISBN}
- **Mitarbeiter:** {Sozialversicherungsnummer} oder {E-Mailadresse}?? oder {Passnummer, Land} oder ...

Wahl der Schlüsselattribute ist eine **Modellentscheidung** und benötigt Anwendungs-/Domänenwissen sowie Festlegungen (z.B. E-Mailadresse gehört genau einer Person)

Gegebenenfalls sind **künstliche Schlüsselattribute** einzuführen (z.B. PersNr bei Mitarbeiter)

Rekursion und Rollennamen

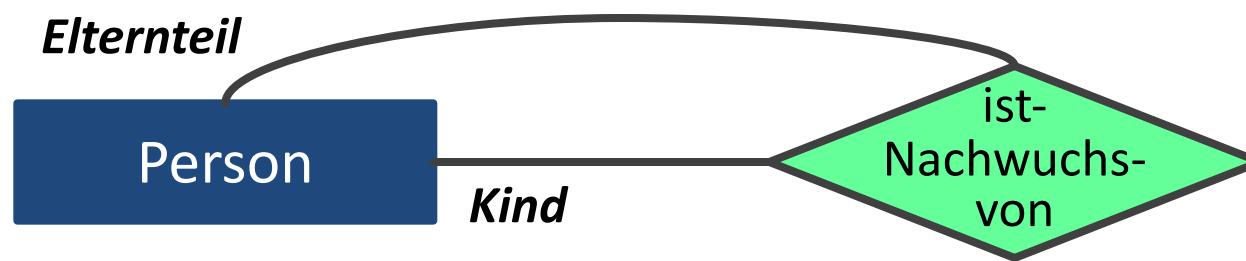


Rekursiver Beziehungstyp

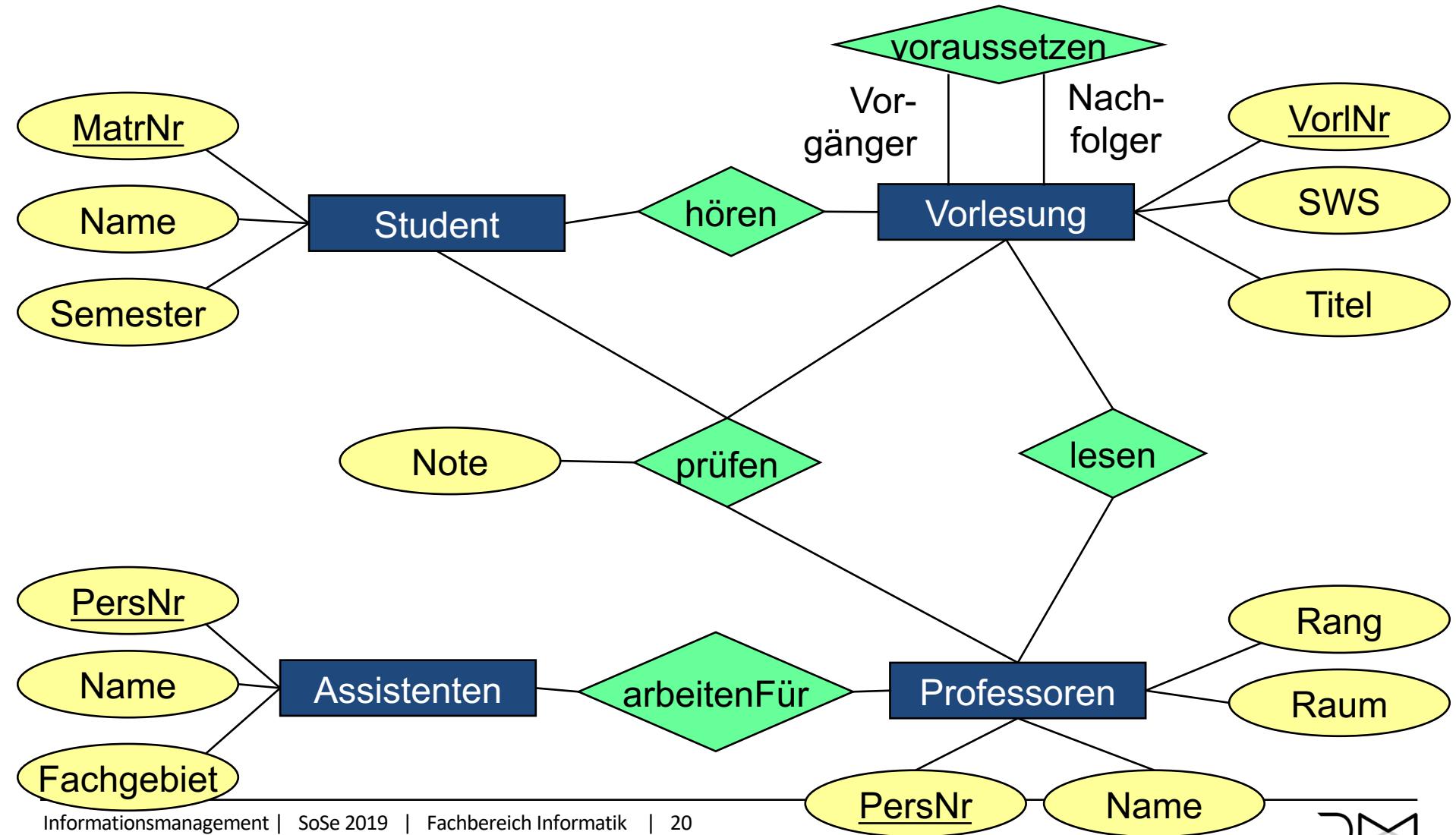
- Beziehungstypen an denen ein Entitätstyp mehrfach beteiligt ist
- Beispiel: `ist-Nachwuchs-von(Person, Person)`
- Spätestens jetzt sind **Rollennamen** nötig, z.B. Elternteil, Kind

ER-Diagramm:

- `ist-Nachwuchs-von(Elternteil: Person, Kind: Person)`



Beispiel: Universitätsschema



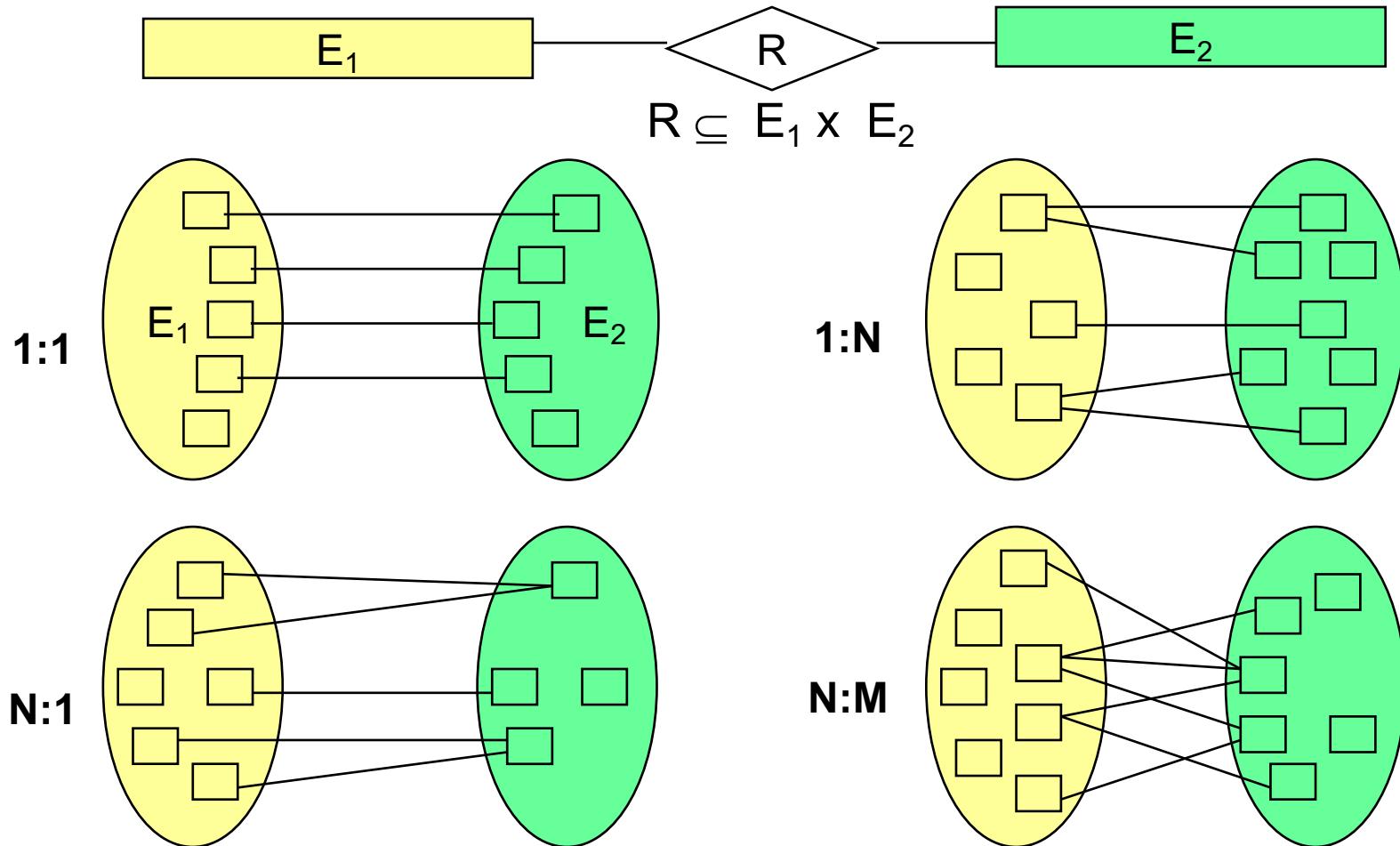
Datenmodellierung

Entity-Relationship Diagramme

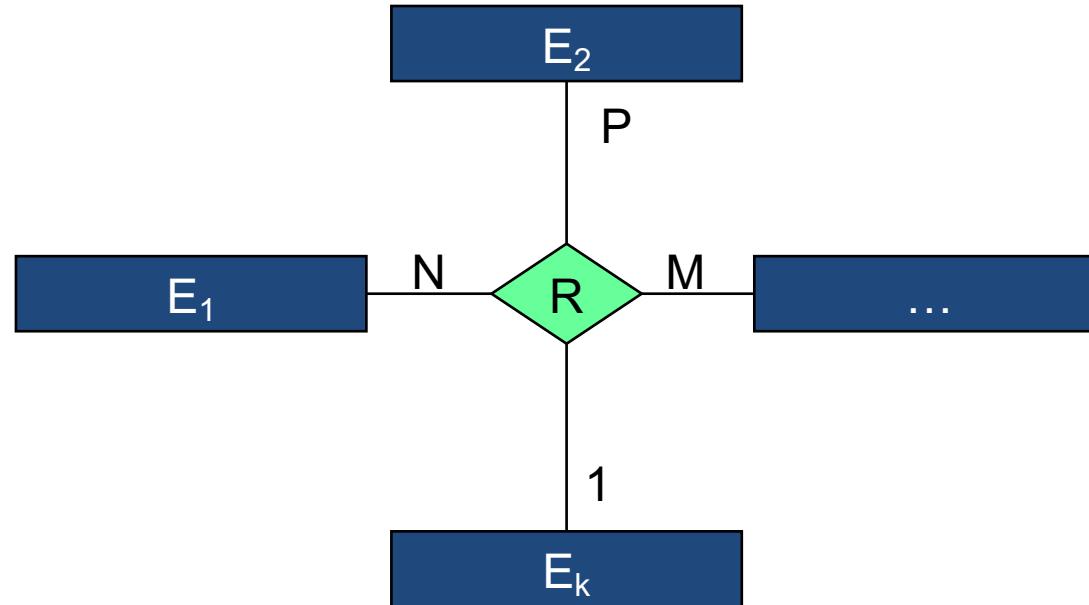
- Basiskomponenten
- **Funktionalitäten**
- (min, max)-Notation
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

Funktionalitäten („Chen“-Notation)



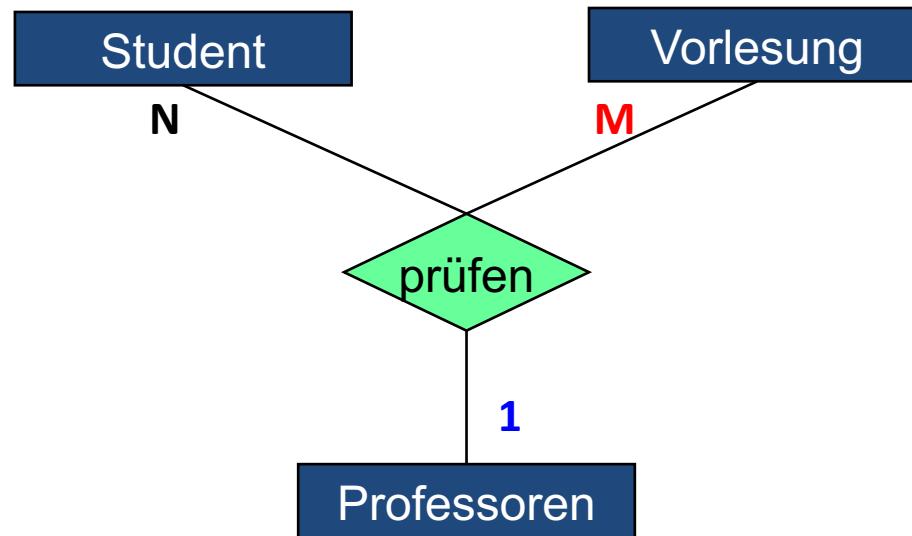
Funktionalitäten bei n-stelligen Beziehungen



Bestimmung der Funktionalität:

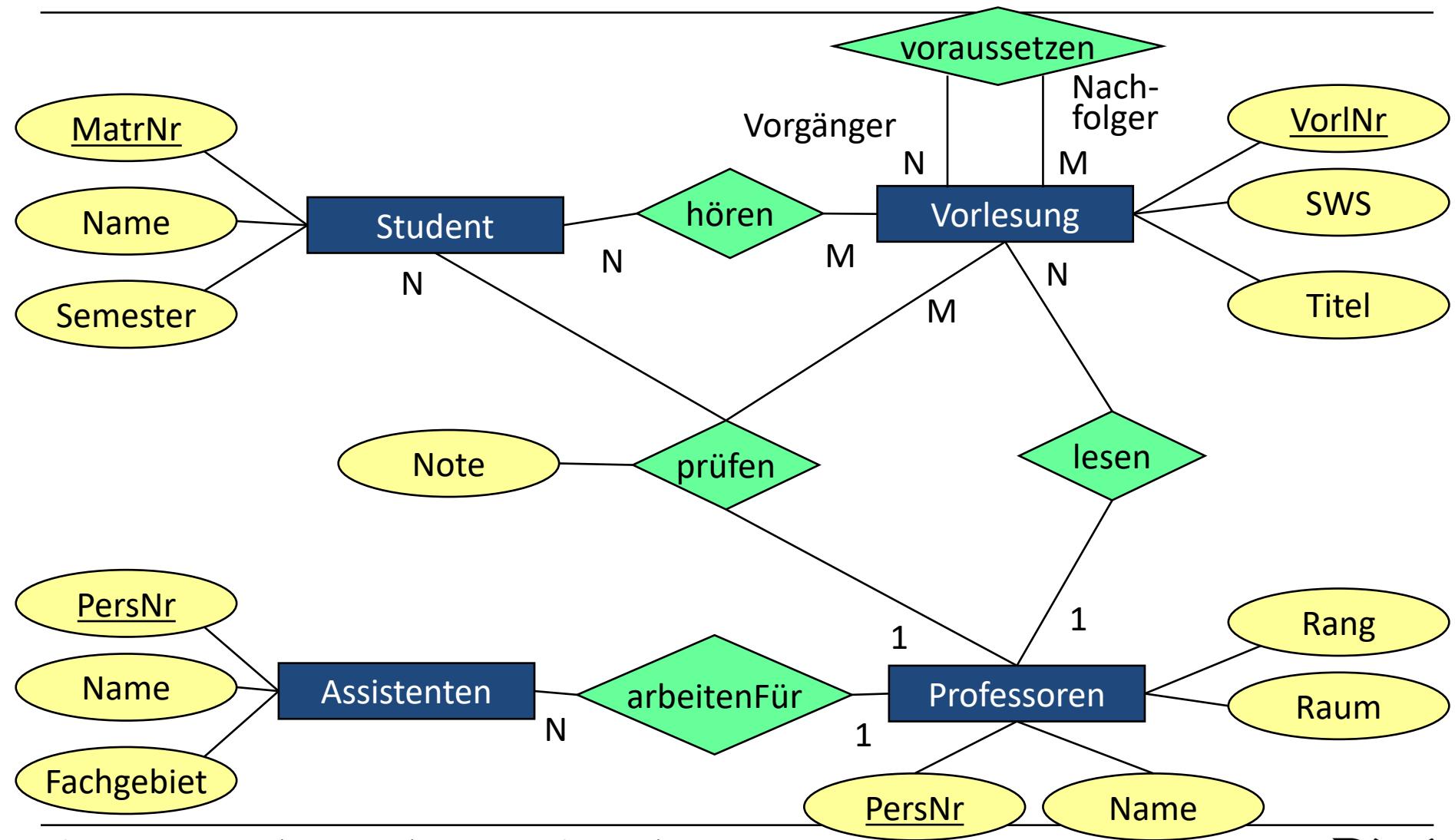
$$e_1 \in E_1 \times e_2 \in E_2 \times \dots \times e_{n-1} \in E_{n-1} \rightarrow E_k ?$$

Beispiel: Beziehung „prüfen“



$s1 \in \text{Studenten} \times v1 \in \text{Vorlesung} \rightarrow \text{Professor?}$
 $p1 \in \text{Professor} \times s1 \in \text{Student} \rightarrow \text{Vorlesung?}$
 $p1 \in \text{Professor} \times v1 \in \text{Vorlesung} \rightarrow \text{Student?}$

Beispiel: Universitätsschema



Datenmodellierung

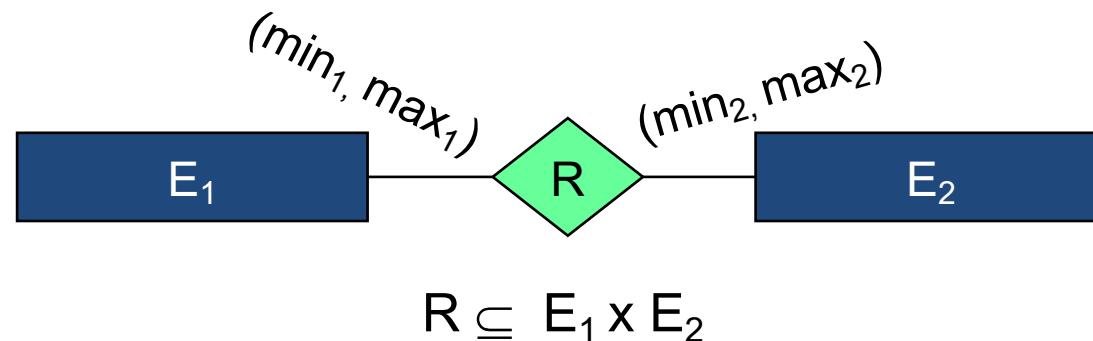
Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- **(min, max)-Notation**
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

(min, max)-Notation

(min, max)-Notation ist präziser als Funktionalitäten:
statt 1:1, 1:N, N:M werden Unter- und Obergrenze angegeben



Für jedes $e_i \in E_i$ gibt es

- Mindestens \min_i Tupel der Art $(..., e_i, ...) \in R$ und
- Höchstens \max_i viele Tupel der Art $(..., e_i, ...) \in R$

Beispiele: (min, max)-Notation

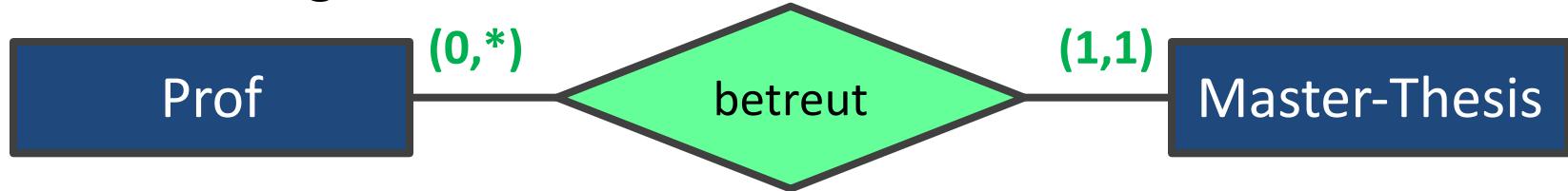


Min-Max-Notation

- 1:1-Beziehung



- 1:n-Beziehung



- m:n-Beziehung



Beispiele: (min, max)-Notation

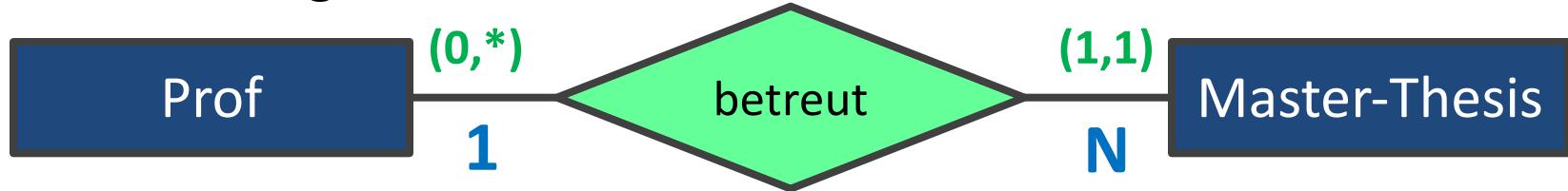


Min-Max-Notation vs. Chen-Notation

- 1:1-Beziehung



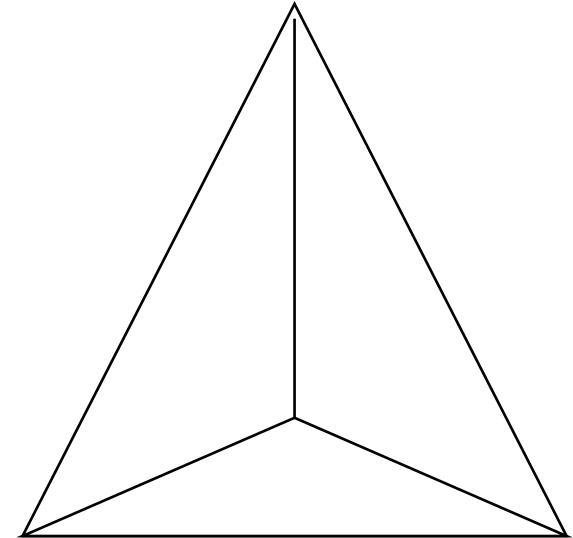
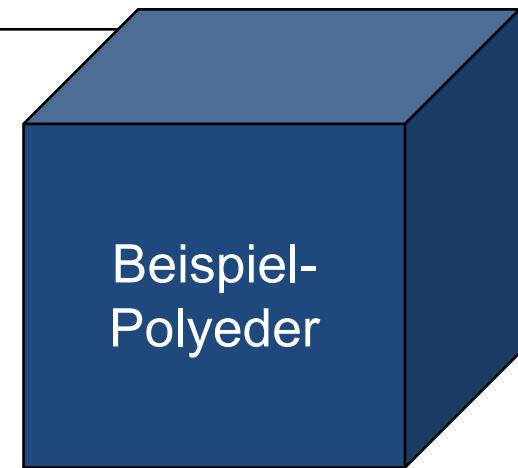
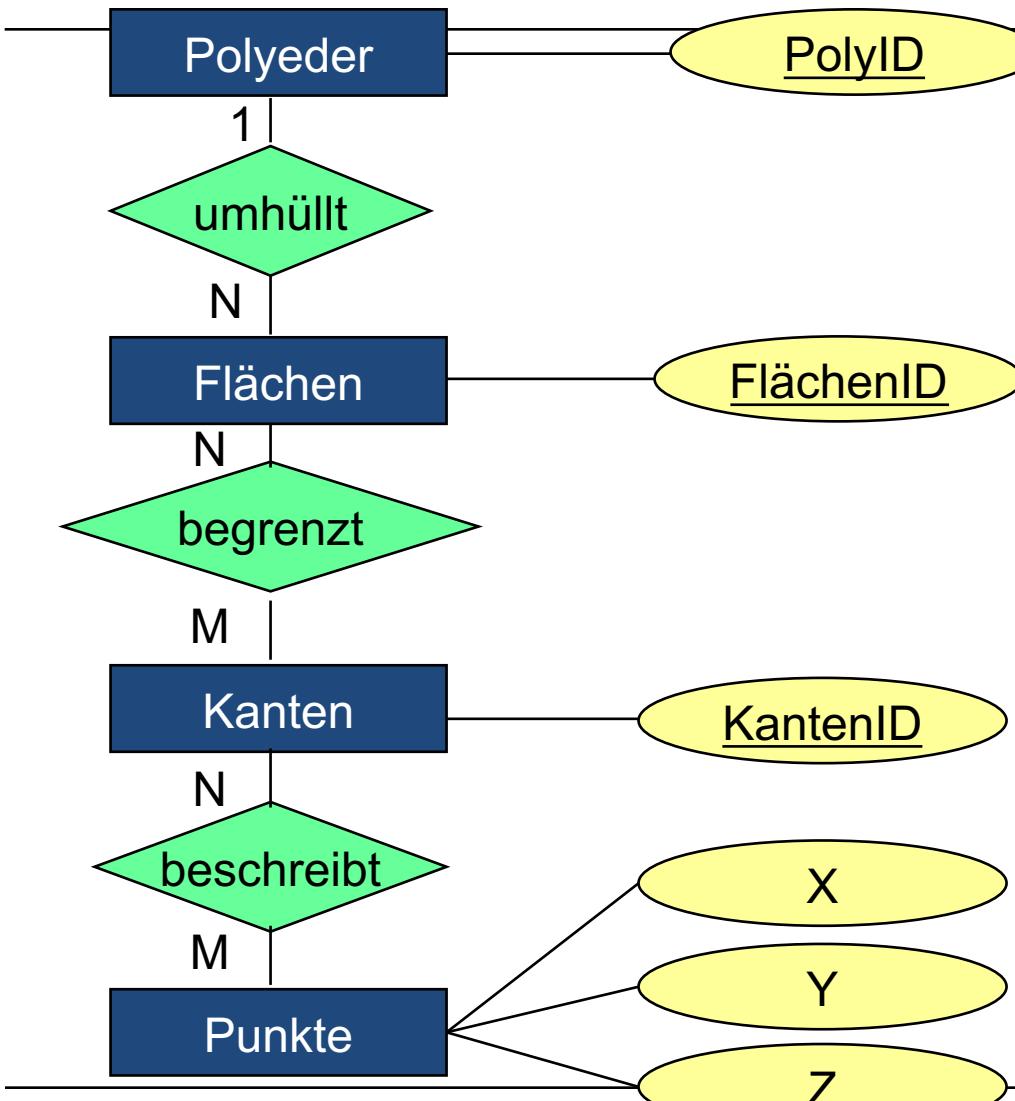
- 1:n-Beziehung



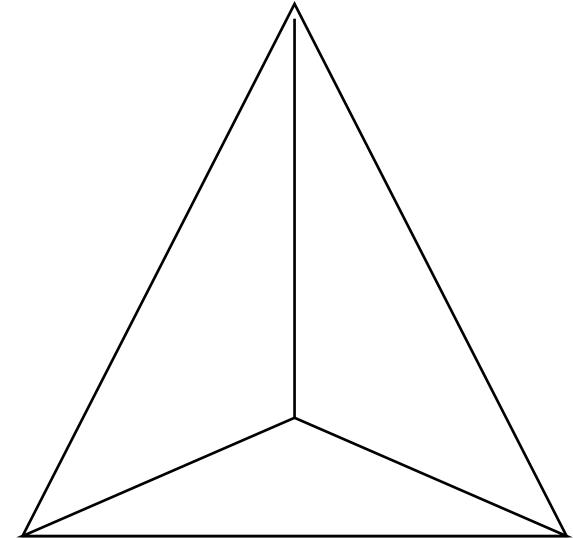
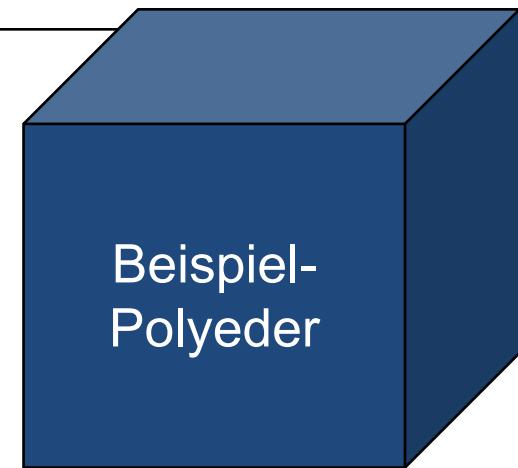
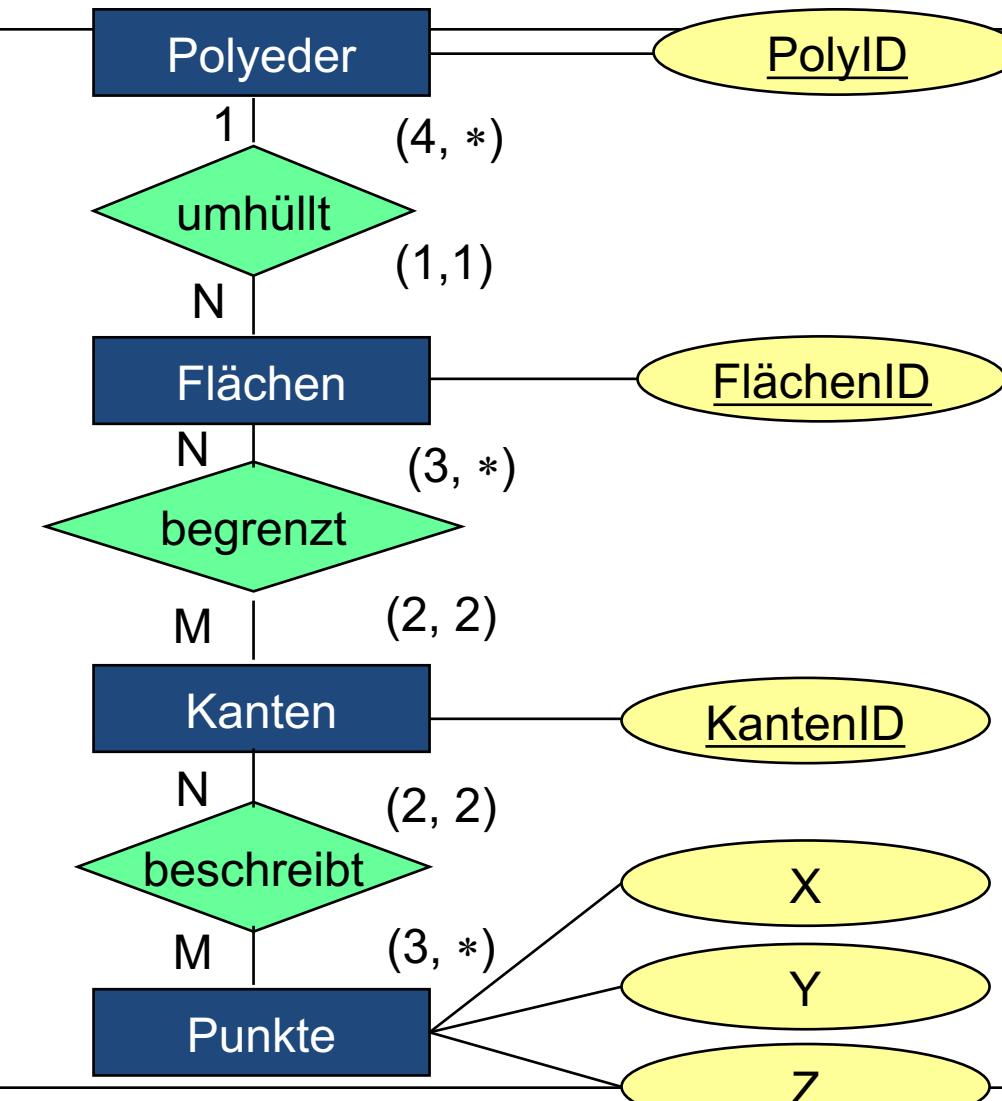
- m:n-Beziehung



Beispiel: CAD-Software



Beispiel: CAD-Software



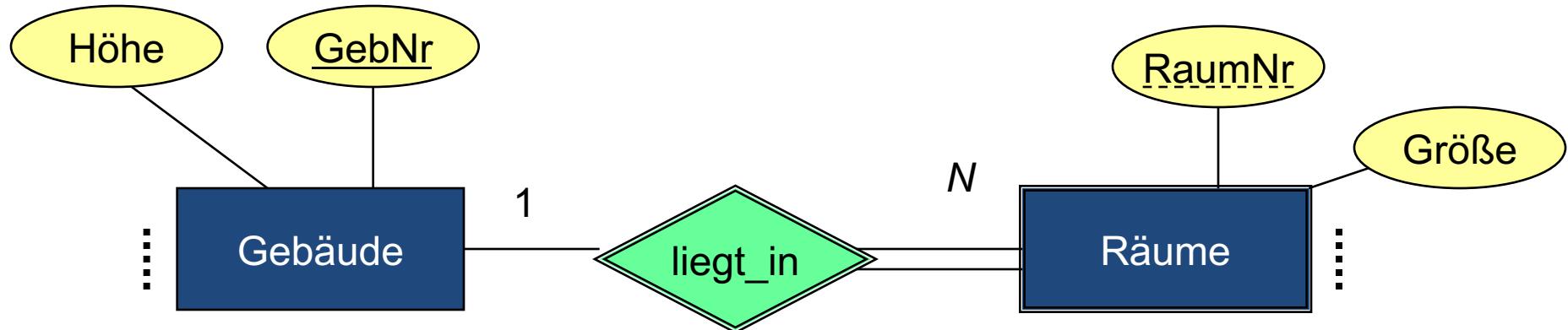
Datenmodellierung

Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- (min, max)-Notation
- **Starke und schwache Entitäten**
- Generalisierung und Aggregation

UML-Klassendiagramme

Schwache (existenzabhängige) Entitäten



Schwache Entität (z.B. Raum) existiert nur, wenn **starke Entität** (z.B. Gebäude) existiert

- **Schwache Entität und zugehörige Beziehung** werden in ER-Modell mit **doppelter Linie** umrahmt
- **Primärschlüssel** der schwachen Entität **gestrichelt unterstrichen** (z.B. RaumNr) - ist nur mit Primärschlüssel der starken Entität eindeutig

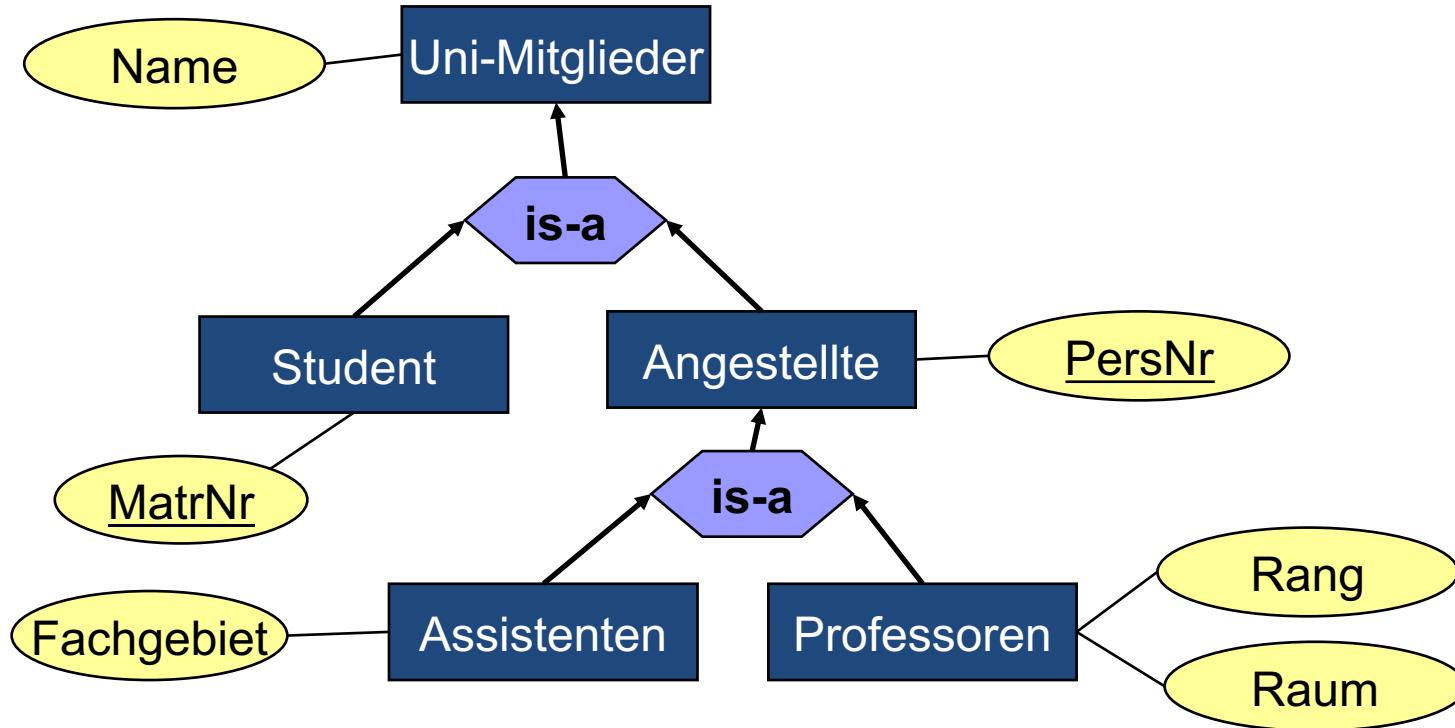
Datenmodellierung

Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- (min, max)-Notation
- Starke und schwache Entitäten
- **Generalisierung und Aggregation**

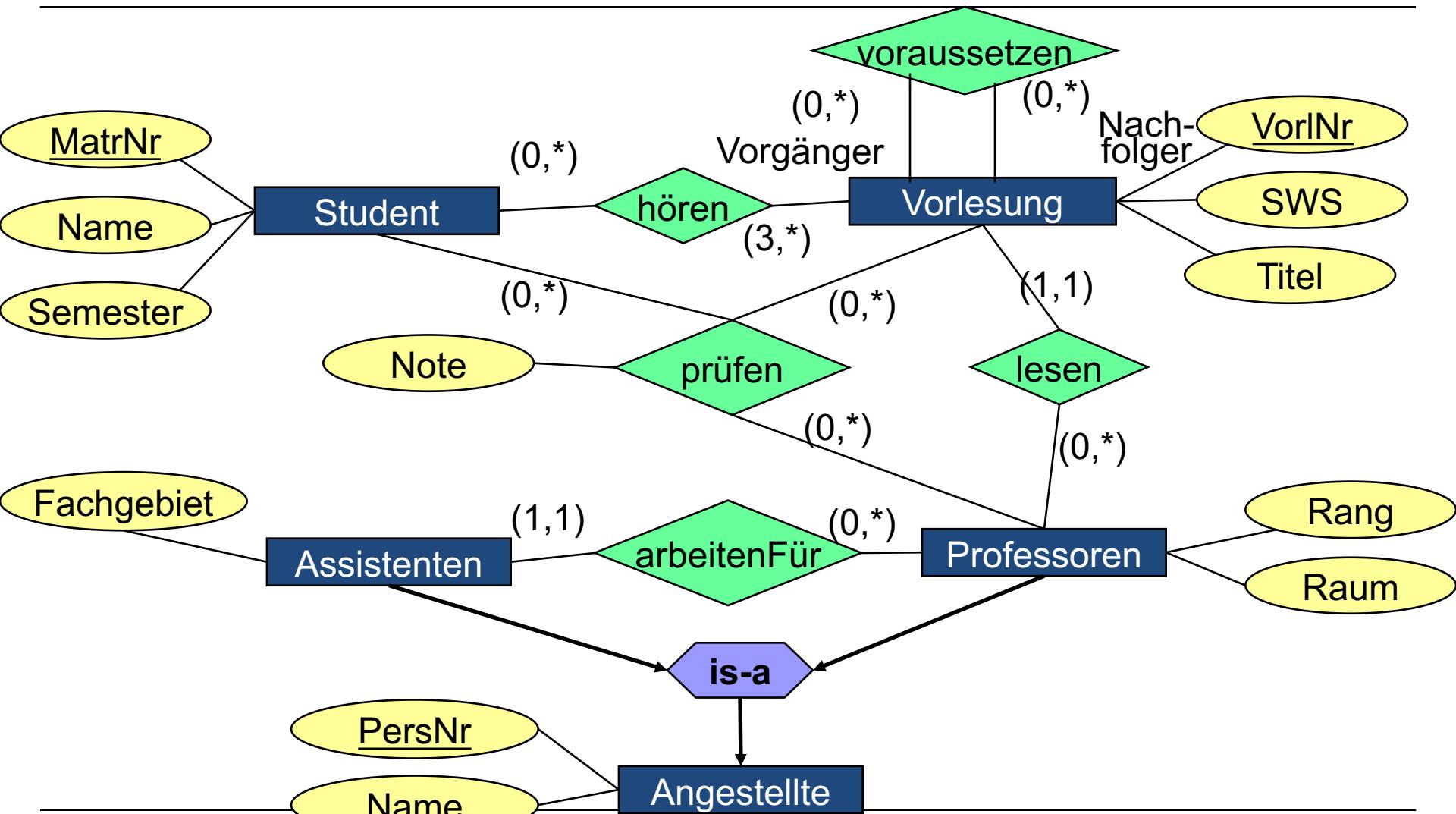
UML-Klassendiagramme

Generalisierung (Vererbung)

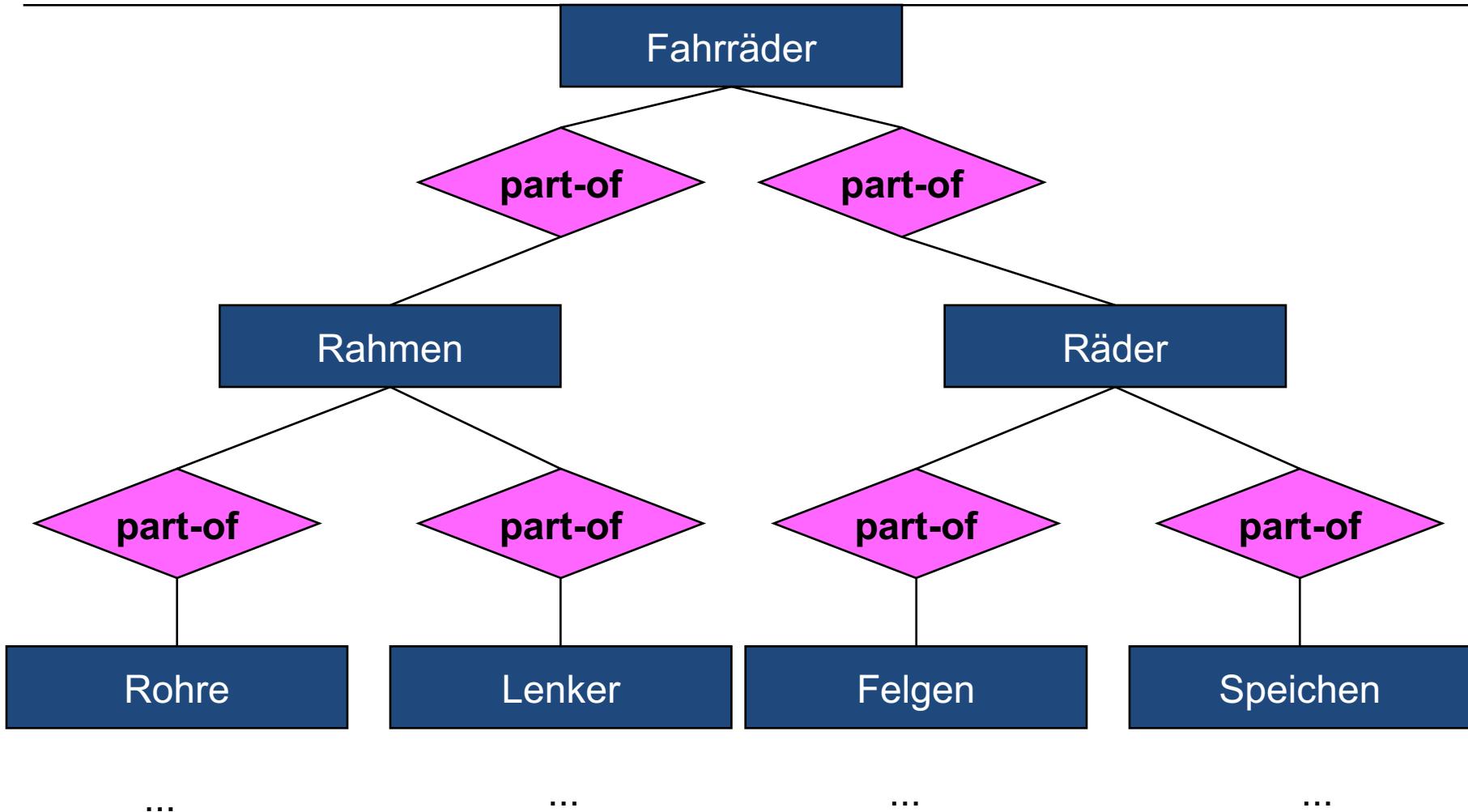


- **Gemeinsame Attribute** mehrerer Entitäten werden in **Obertyp** zusammengefasst
- **Vererbung der Attribute** des Obertyps an den Untertyp!

Beispiel: Universitätsschema



Aggregation (Teil- Ganzes Beziehung)



Datenmodellierung

Entity-Relationship Diagramme

- Basiskomponenten
- Funktionalitäten
- (min, max)-Notation
- Starke und schwache Entitäten
- Generalisierung und Aggregation

UML-Klassendiagramme

Datenmodellierung mit UML

wird nicht
abgefragt!

UML = Unified Modelling Language (UML)

De-facto Standard für den **objekt-orientierten Software-Entwurf**

Klassendiagramme dienen zur Datenmodellierung

Viele andere Diagrammarten zur Modellierung des Verhaltens:
Aktivitätsdiagramme, Sequenzdiagramme, ...

UML: Klassen und Assoziationen

wird nicht
abgefragt!



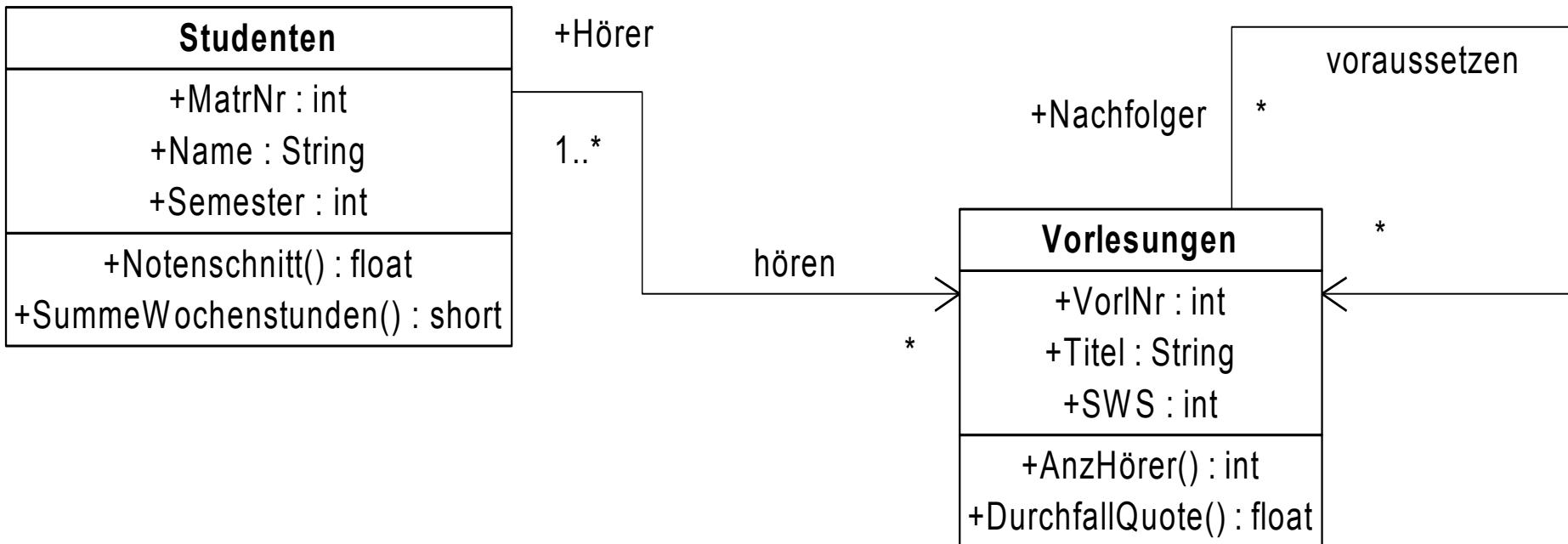
Klassen (ähnlich Entitätstyp): beschreiben Daten aber auch Verhalten oder abgeleitet Eigenschaften durch Methoden

Assoziationen (ähnlich Relationships): beschreiben Beziehung zwischen Klassen

- **Multiplizitätsangabe:** Jedes Element von Klasse A steht mit mindestens i Instanzen der Klasse B... und mit maximal j
- **Position der Multiplizitätsangabe** ist analog zur Position Funktionalitätsangabe im ER-Modell (nicht analog zu min-max!)

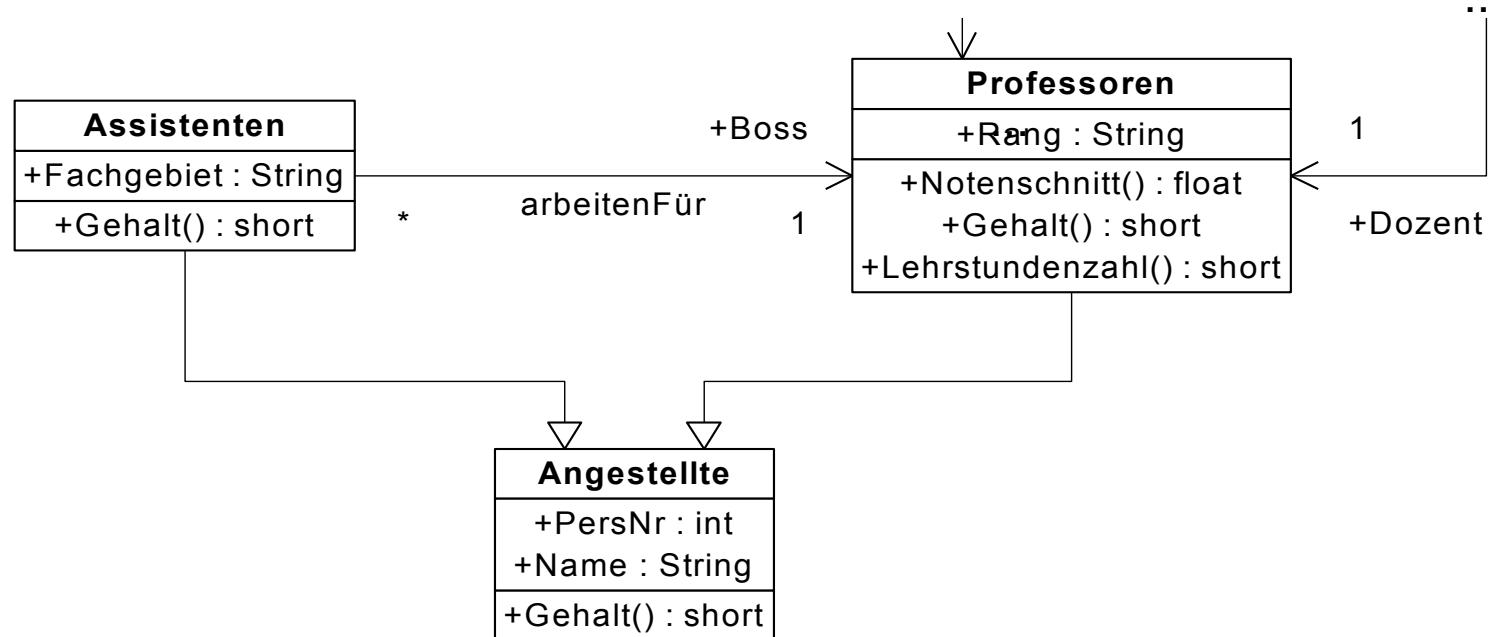
Beispiel: Universitätswelt

wird nicht
abgefragt!



UML: Generalisierung (Vererbung)

wird nicht
abgefragt!

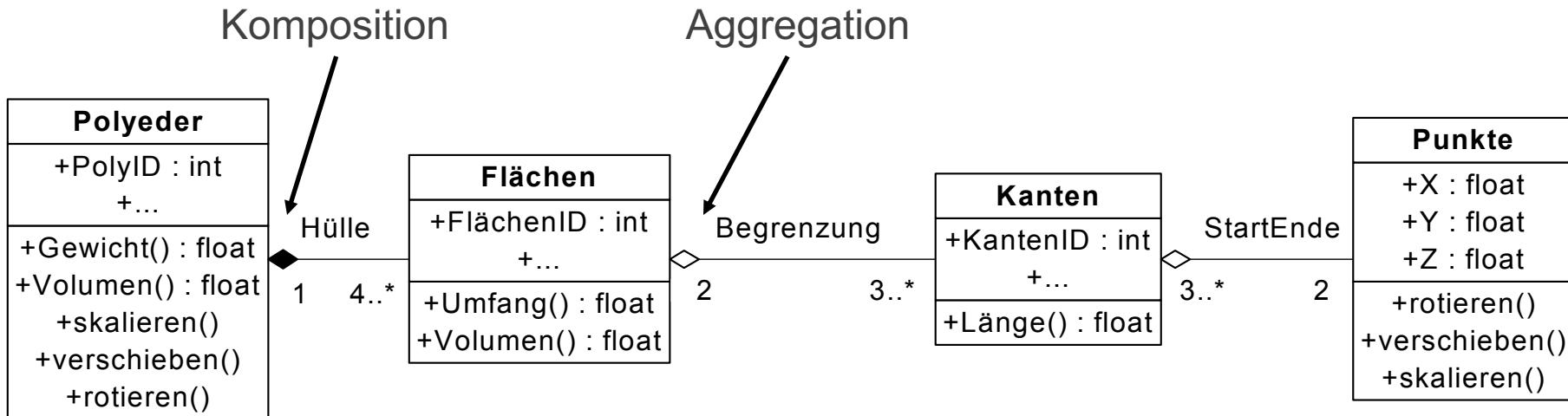


Generalisierung ist ähnlich wie in ER-Diagrammen

- Gemeinsame Attribute und Methoden werden in **Obertyp** zusammengefasst
- Vererbung der Attribute des Obertyps an den Untertyp!

UML: Aggregation und Komposition

wird nicht
abgefragt!



Aggregation: Teil-Ganzes Beziehung

Komposition: Spezialfall der Aggregation bei der Teil nur existiert wenn Ganzes existiert (ähnlich „schwacher Entität“ in ER)



Fragen?

Referenzen

- Alfons Kemper: **Datenbanksysteme - Eine Einführung**, 10. Auflage. De Gruyter Studium, de Gruyter Oldenbourg 2015, ISBN 978-3-11-044375-2
- Theo Härder: Realisierung von operationalen Schnittstellen. In: *Datenbank-Handbuch*, S. 163–335, Berlin: Springer, 1987.
- Gunter Saake, Kai-Uwe Sattler, Andreas Heuer: *Datenbanken: Konzepte und Sprachen*. 4. Auflage, Heidelberg: mitp, 2010.
- Gottfried Vossen: *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*. 5. Auflage. München: Oldenbourg, 2008.