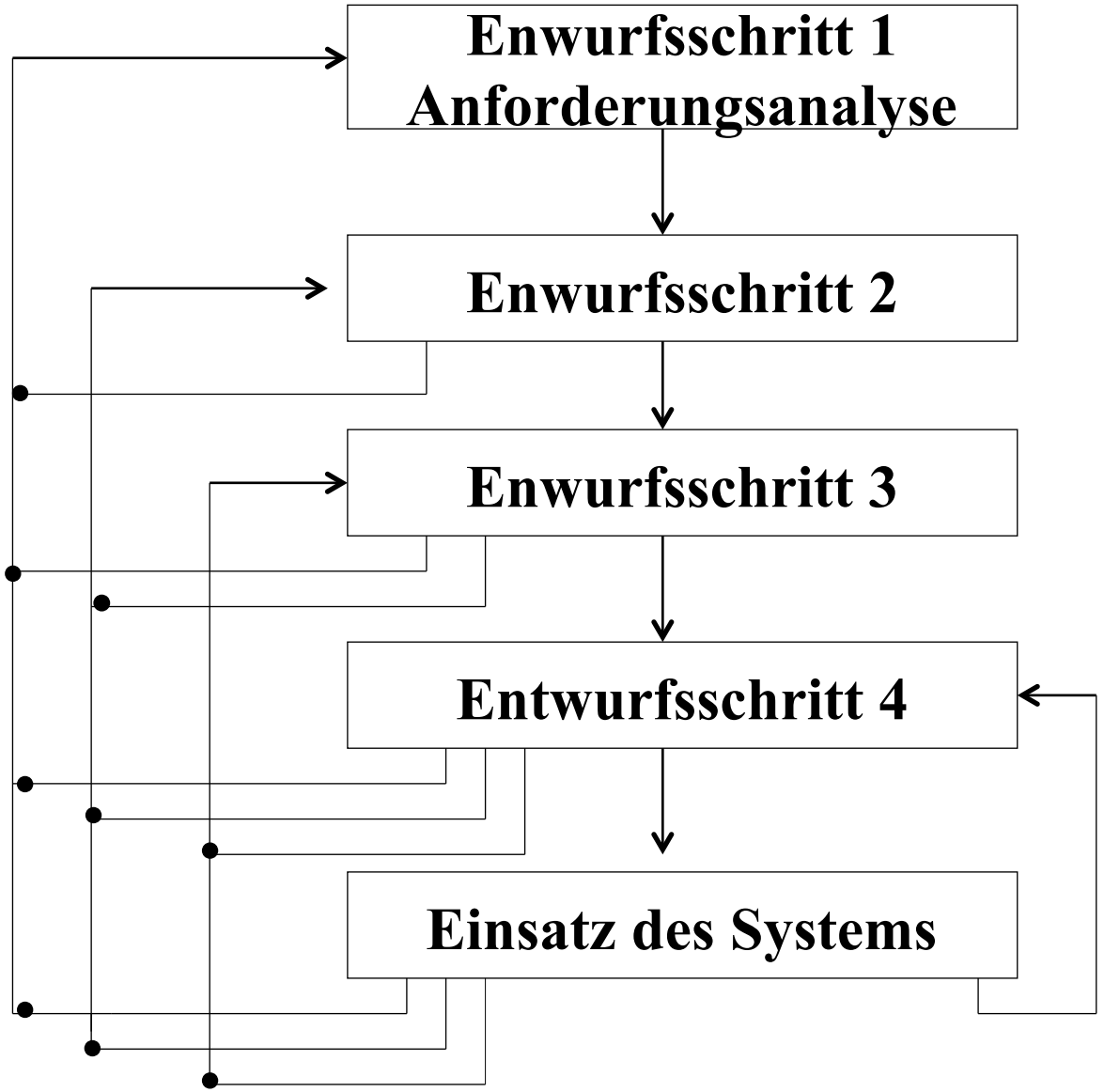
Datenbankentwurf

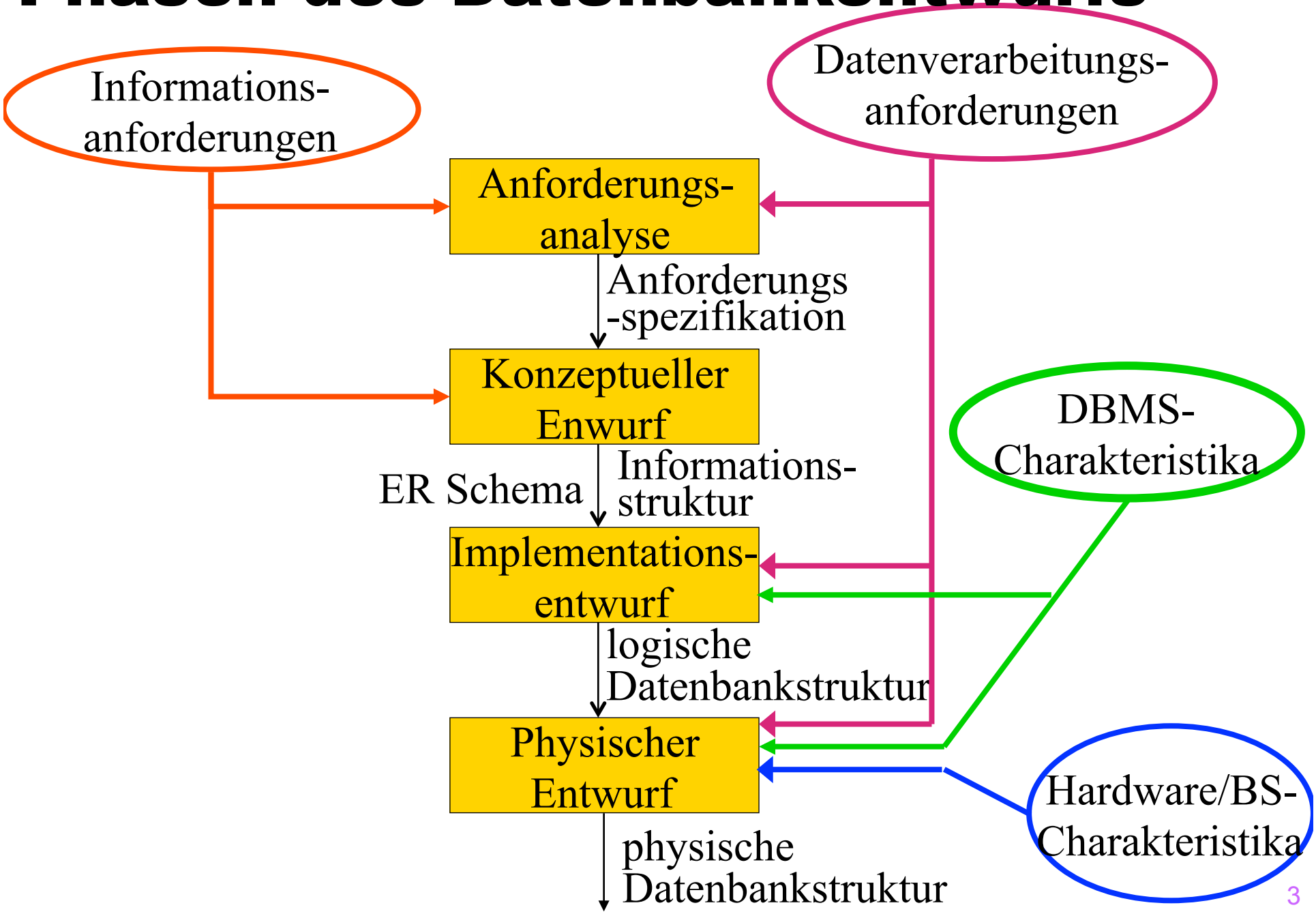
Abstraktionsebenen des Datenbankentwurfs

1. Konzeptuelle Ebene
2. Implementationsebene
3. Physische Ebene

Allgemeiner „top-down Entwurf“



Phasen des Datenbankentwurfs



Anforderungsanalyse

1. Identifikation von Organisationseinheiten
2. Identifikation der zu unterstützenden Aufgaben
3. Anforderungs-Sammelplan
4. Anforderungs-Sammlung
5. Filterung
6. Satzklassifikationen
7. Formalisierung

Objektbeschreibung

- Uni-Angestellte
  - Anzahl: 10 000
  - Attribute
    - ❖PersonalNummer
      - Typ: char
      - Länge: 9
      - Wertebereich: 0...999.999.999
      - Anzahl Wiederholungen: 0
      - Definiertheit: 100%
      - Identifizierend: ja
    - ❖Gehalt
      - Typ: dezimal
      - Länge: (8,2)
      - Anzahl Wiederholung: 0
      - Definiertheit: 10%
      - Identifizierend: nein
    - ❖Rang
      - Typ: String
      - Länge: 4
      - Anzahl Wiederholung: 0
      - Definiertheit: 100%
      - Identifizierend: nein

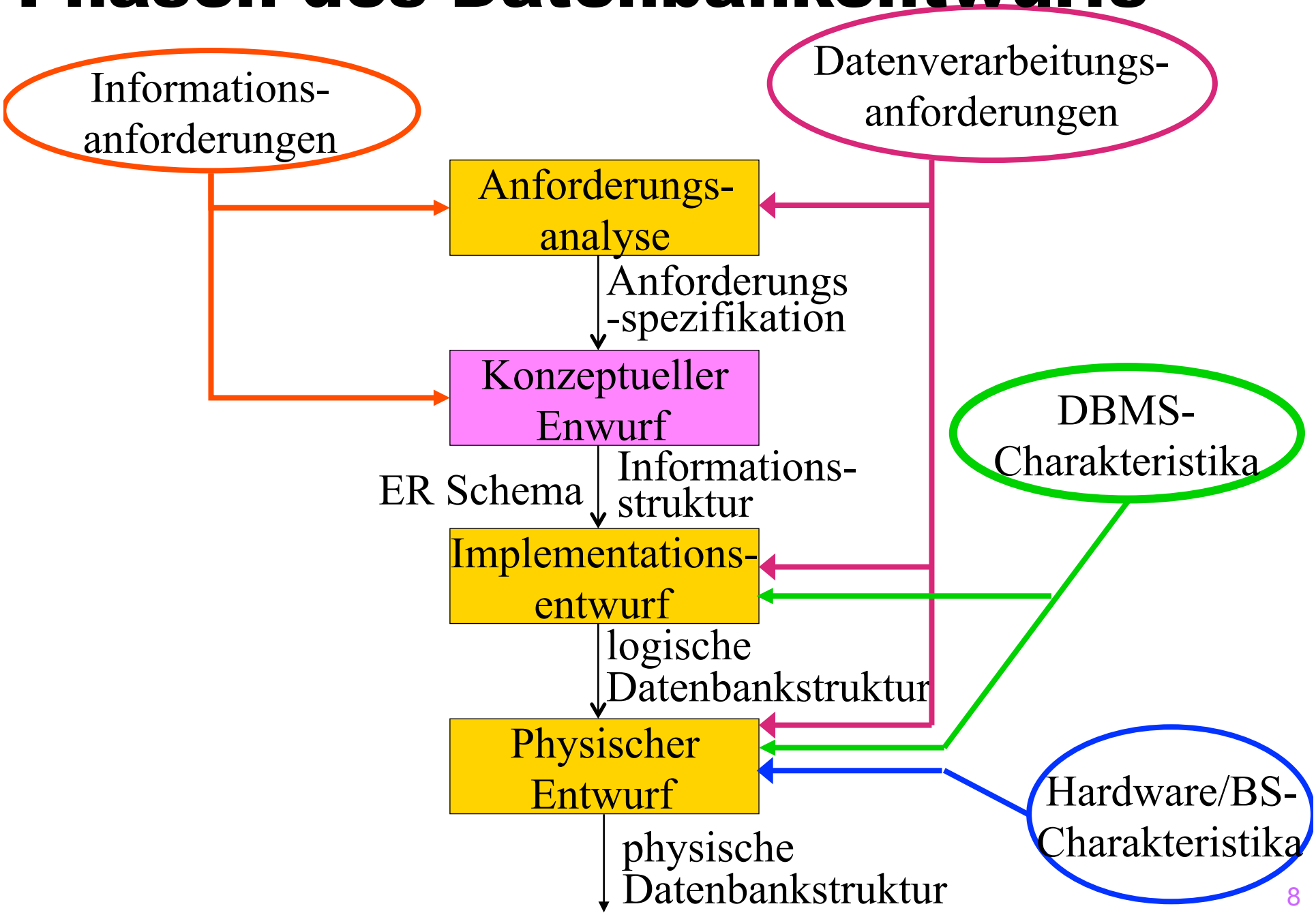
Beziehungsbeschreibung: *prüfen*

- Beteiligte Objekte:
  - Professor als Prüfer
  - Student als Prüfling
  - Vorlesung als Prüfungsstoff
- Attribute der Beziehung:
  - Datum
  - Uhrzeit
  - Note
- Anzahl: 1 000 000 pro Jahr

Prozeßbeschreibungen

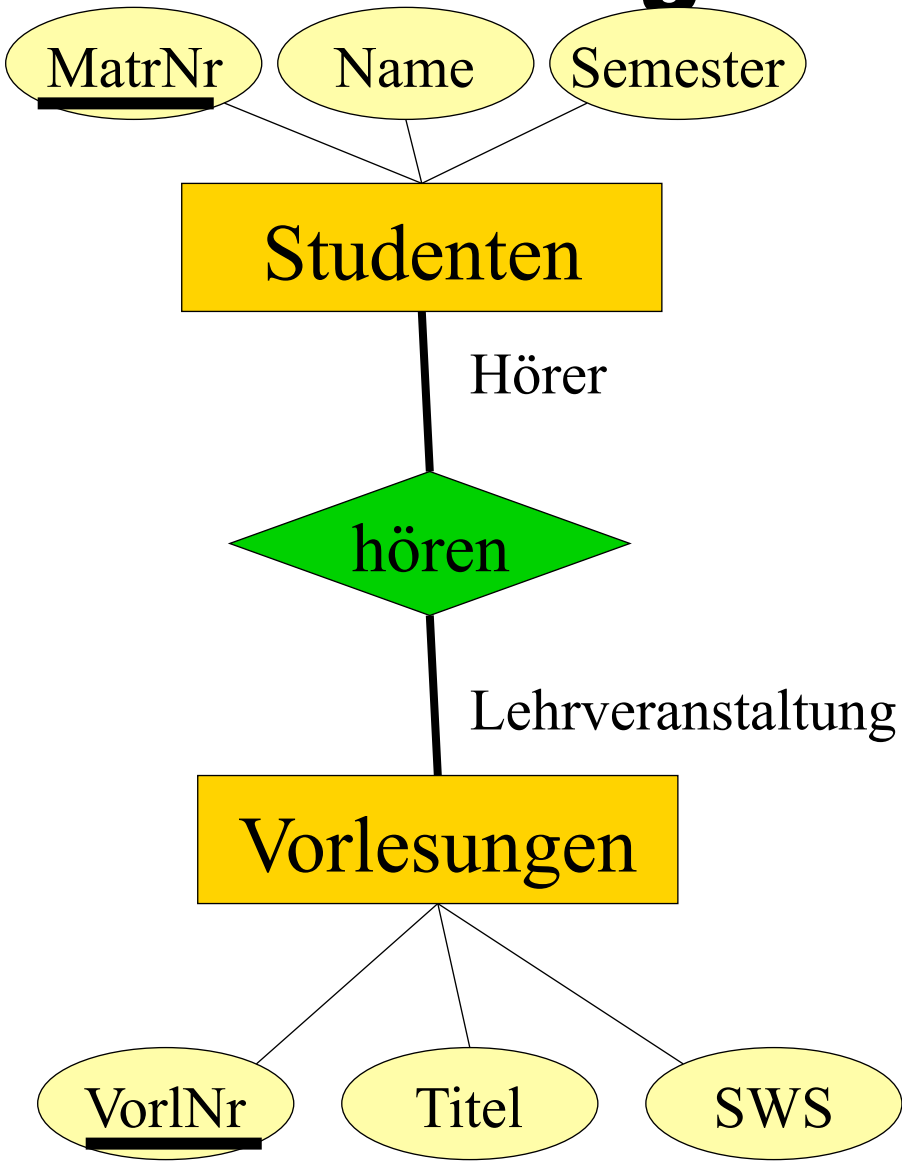
- **Prozeßbeschreibung:** *Zeugnisausstellung*
  - Häufigkeit: halbjährlich
  - benötigte Daten
    - Prüfungen
    - Studienordnungen
    - Studenteninformation
    - ...
  - Priorität: hoch
  - Zu verarbeitende Datenmenge
    - 5 000 Studenten
    - 100 000 Prüfungen
    - 100 Studienordnungen

Phasen des Datenbankentwurfs

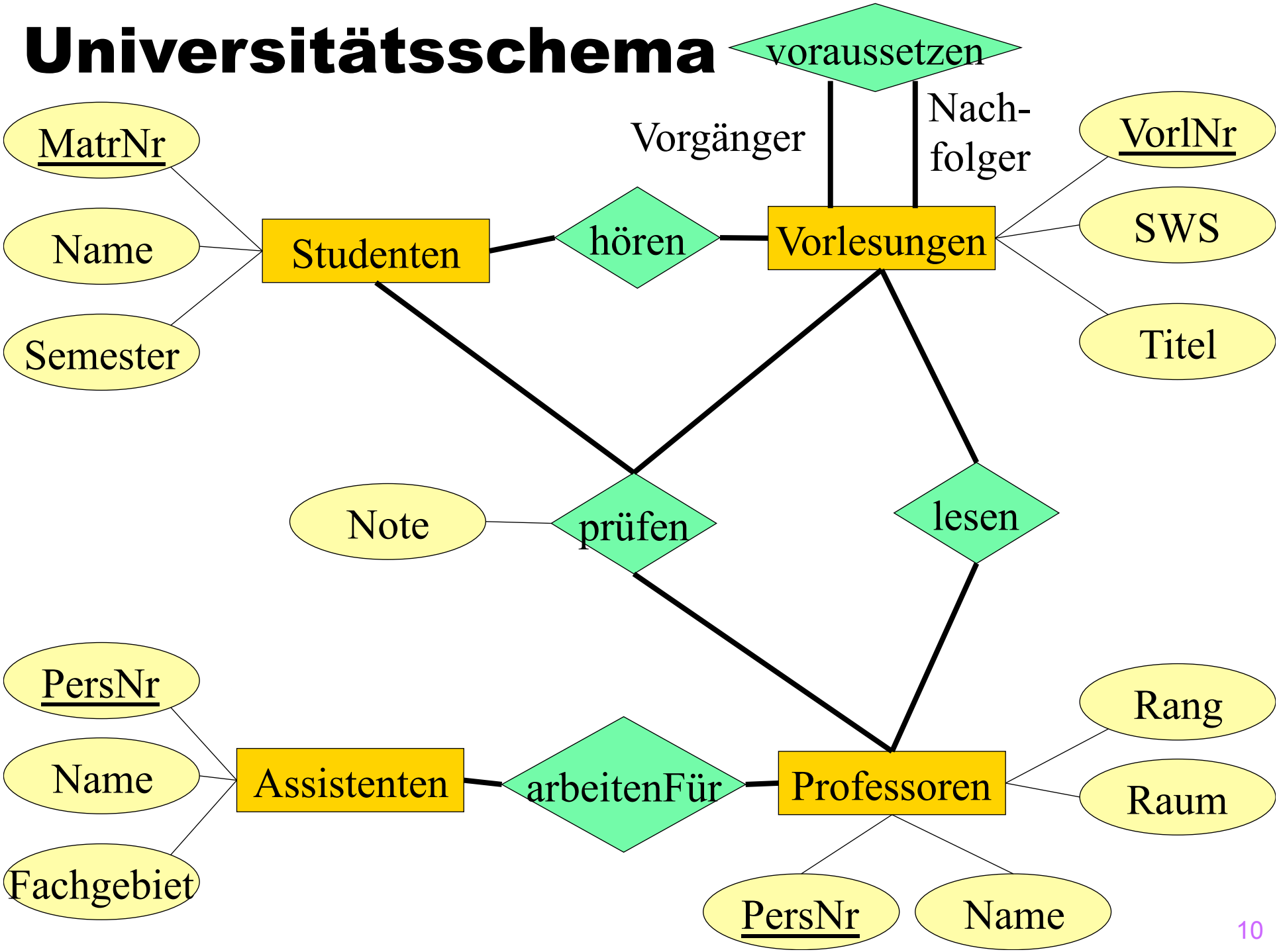


Entity/Relationship-Modellierung

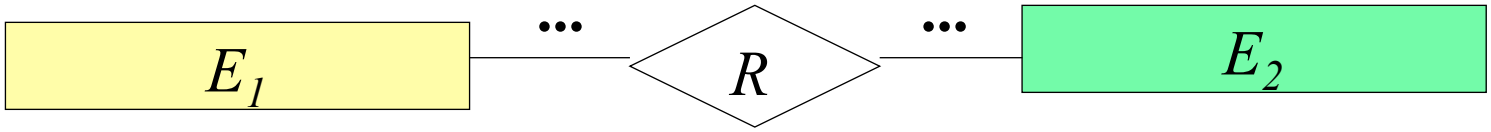
- Entity (Gegenstandstyp)
- Relationship (Beziehungstyp)
- Attribut (Eigenschaft)
- Schlüssel (Identifikation)
- Rolle



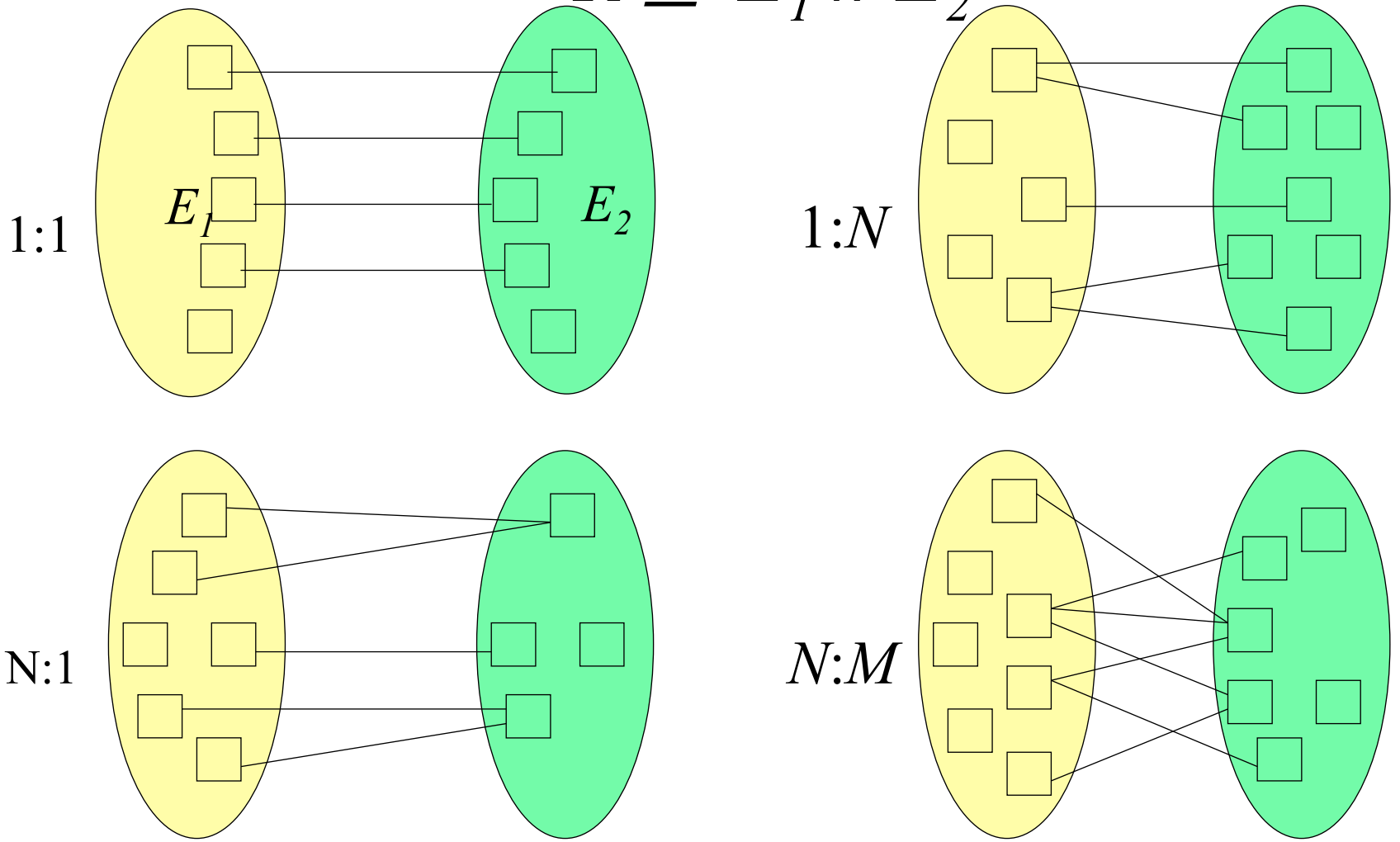
Universitätsschema



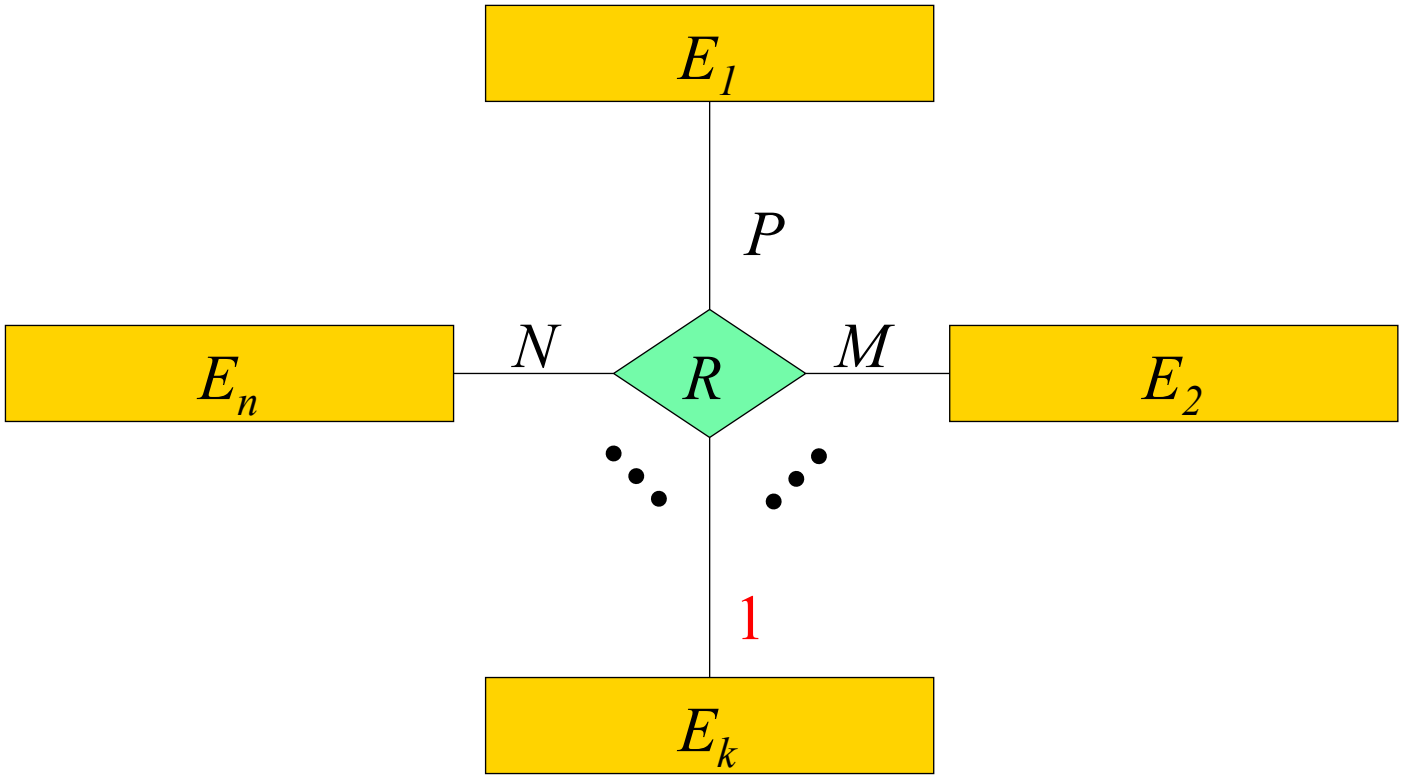
Funktionalitäten



$$R \subseteq E_1 \times E_2$$

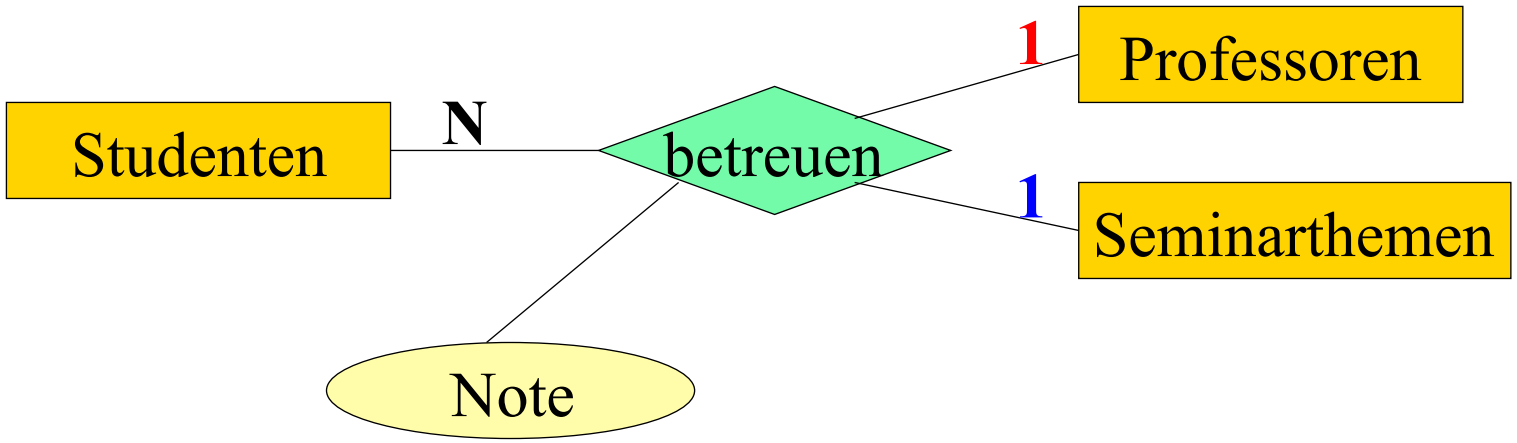


Funktionalitäten bei *n*-stelligen Beziehungen



$$R : E_1 \times \ldots \times E_{k-1} \times E_{k+1} \times \ldots \times E_n \rightarrow E_k$$

Beispiel-Beziehung: *betreuen*



betreuen : Professoren x Studenten → Seminarthemen

betreuen : Seminarthemen x Studenten → Professoren

Dadurch erzwungene Konsistenzbedingungen

1. Studenten dürfen bei demselben Professor bzw. derselben Professorin nur ein Seminarthema "ableisten" (damit ein breites Spektrum abgedeckt wird).

2. Studenten dürfen dasselbe Seminarthema nur einmal bearbeiten – sie dürfen also nicht bei anderen Professoren ein schon einmal erteiltes Seminarthema nochmals bearbeiten.

Es sind aber folgende Datenbankzustände nach wie vor möglich:

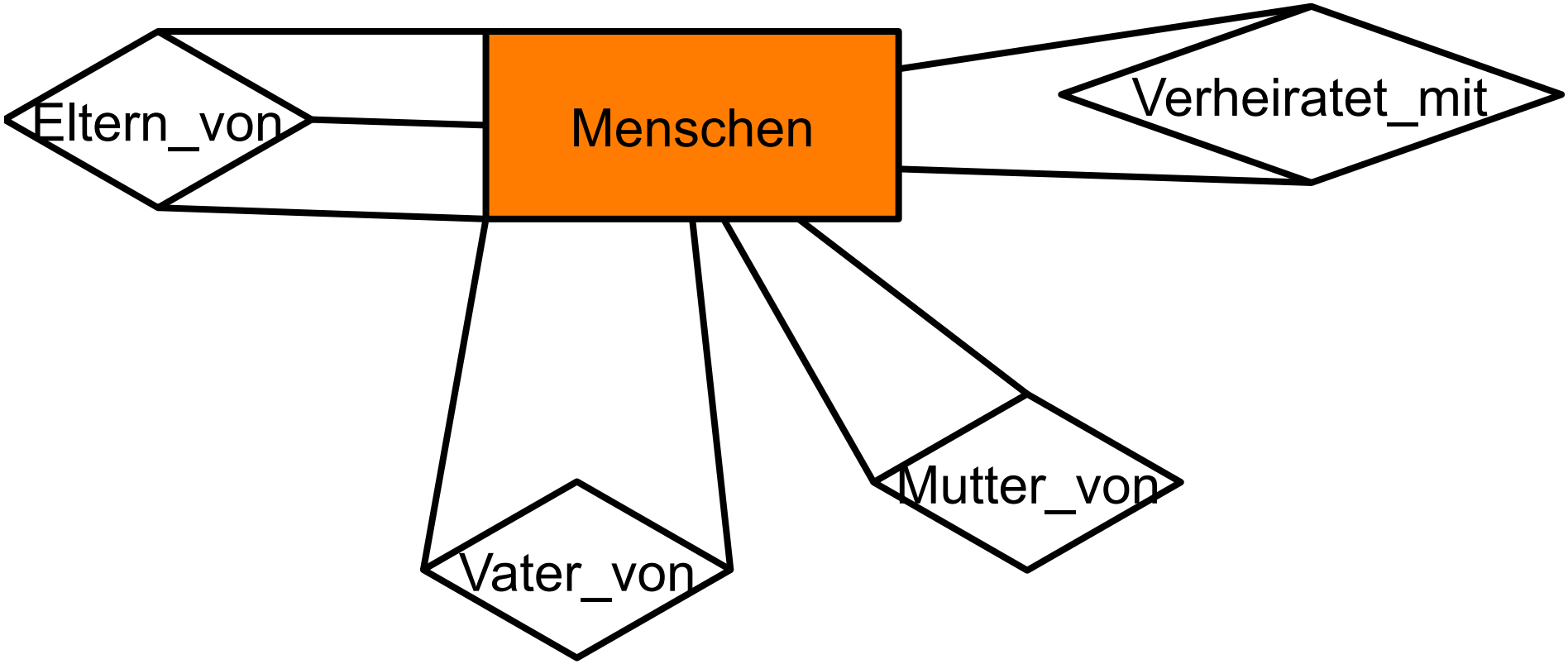
- Professoren können dasselbe Seminarthema „wiederverwenden“ – also dasselbe Thema auch mehreren Studenten erteilen.

- Ein Thema kann von mehreren Professoren vergeben werden – aber an unterschiedliche Studenten.

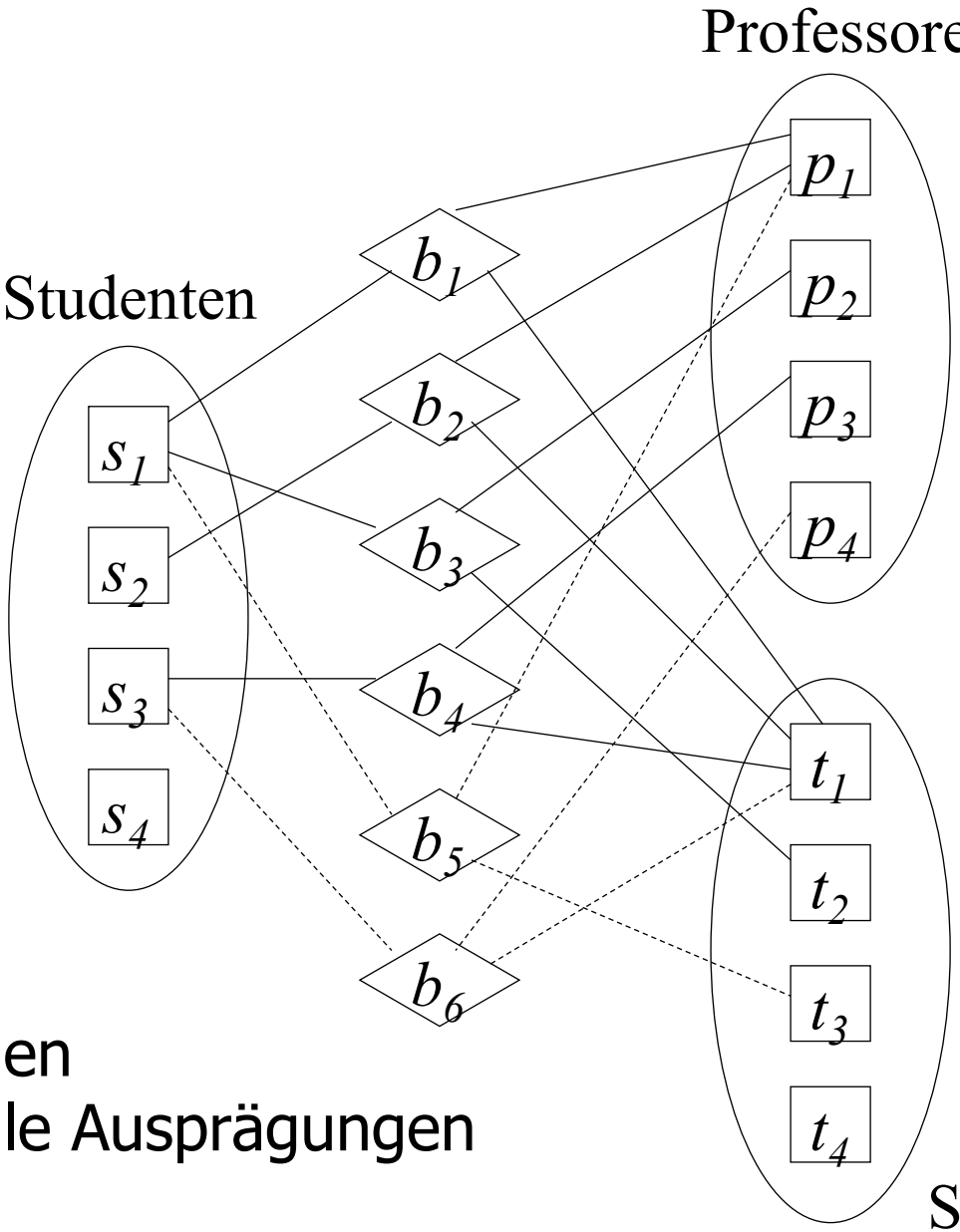
Fachschaftsvollversammlung

- Mittwoch, 30.10.2013 von 10 – 12 Uhr
- Deshalb Vorlesungsbeginn:
  - 12:00 Uhr (s.t., sharp, Punkt!)
  - Bis 13 Uhr

Funktionalitäten

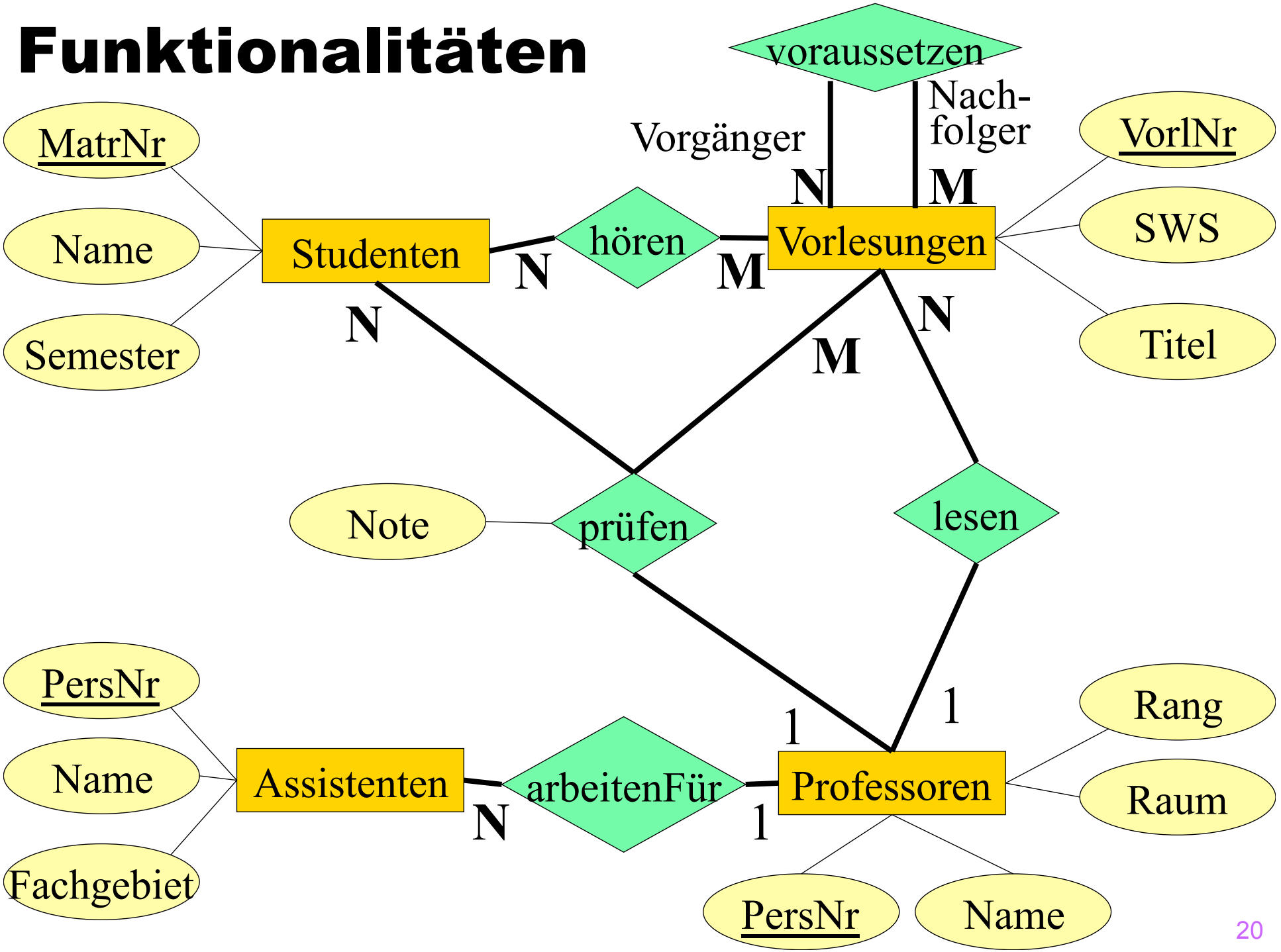


Ausprägung der Beziehung *betreuen*

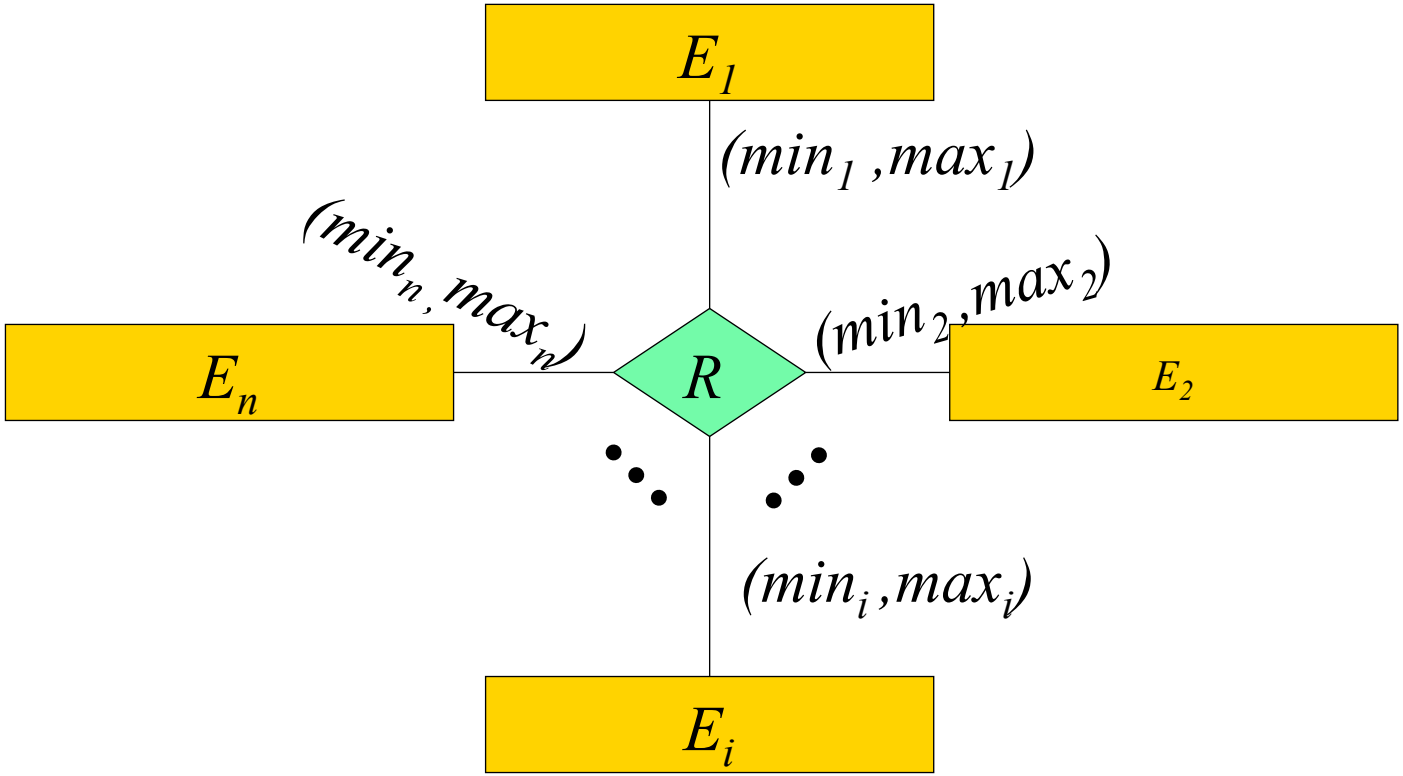


Gestrichelte Linien
markieren illegale Ausprägungen

Funktionalitäten



(min, max)-Notation

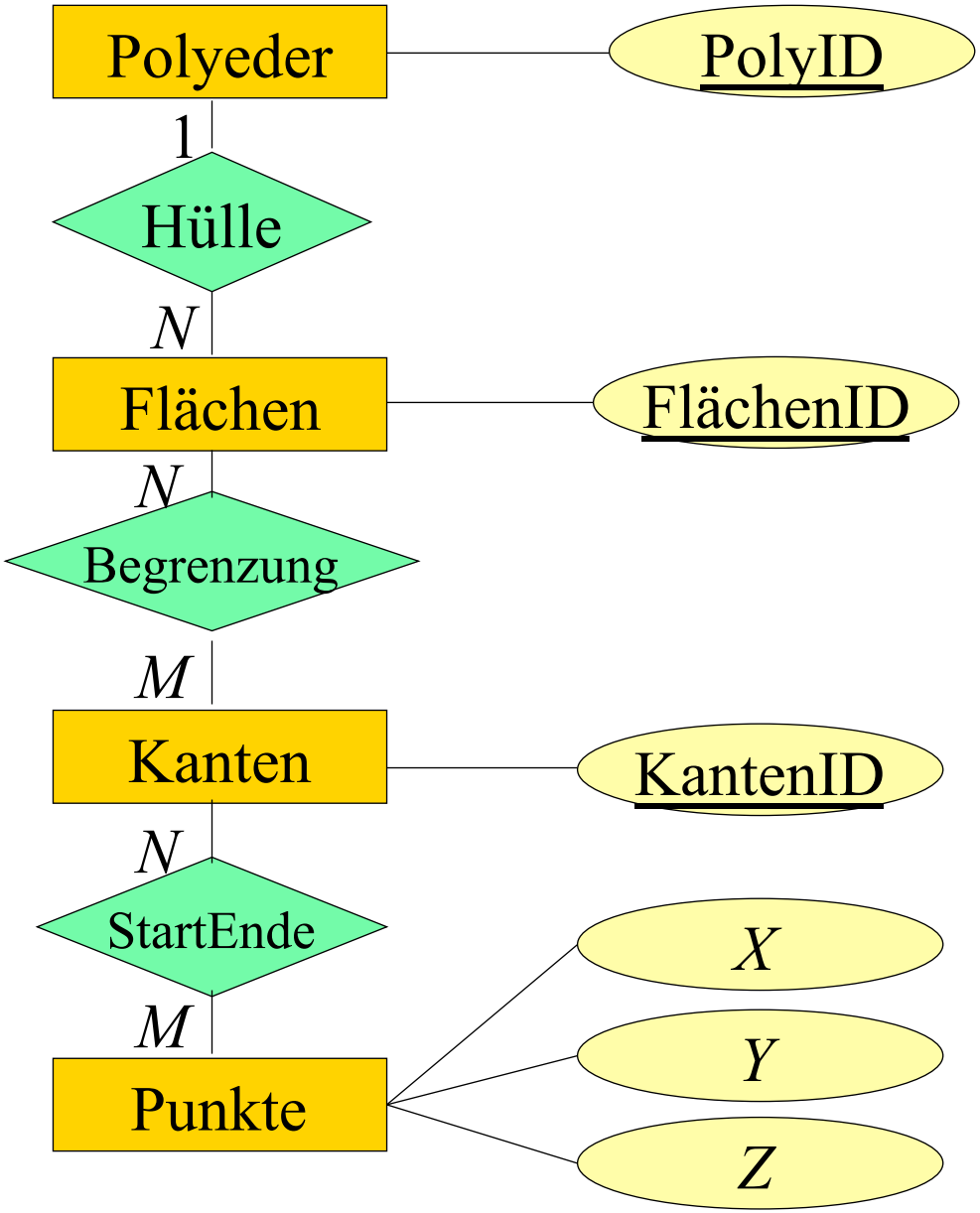


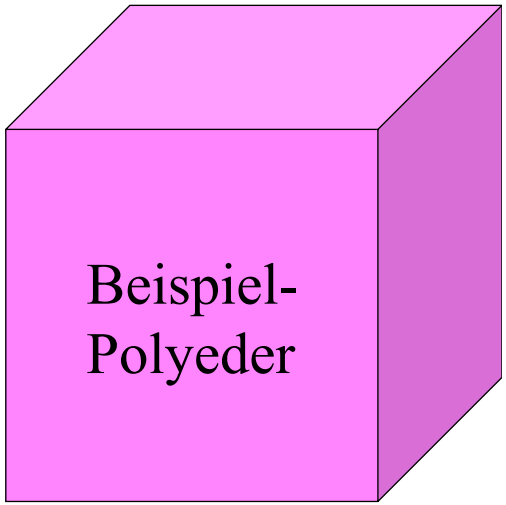
$R \subseteq E_1 \times \ldots \times E_i \times \ldots \times E_n$

Für jedes $e_i \in E_i$ gibt es

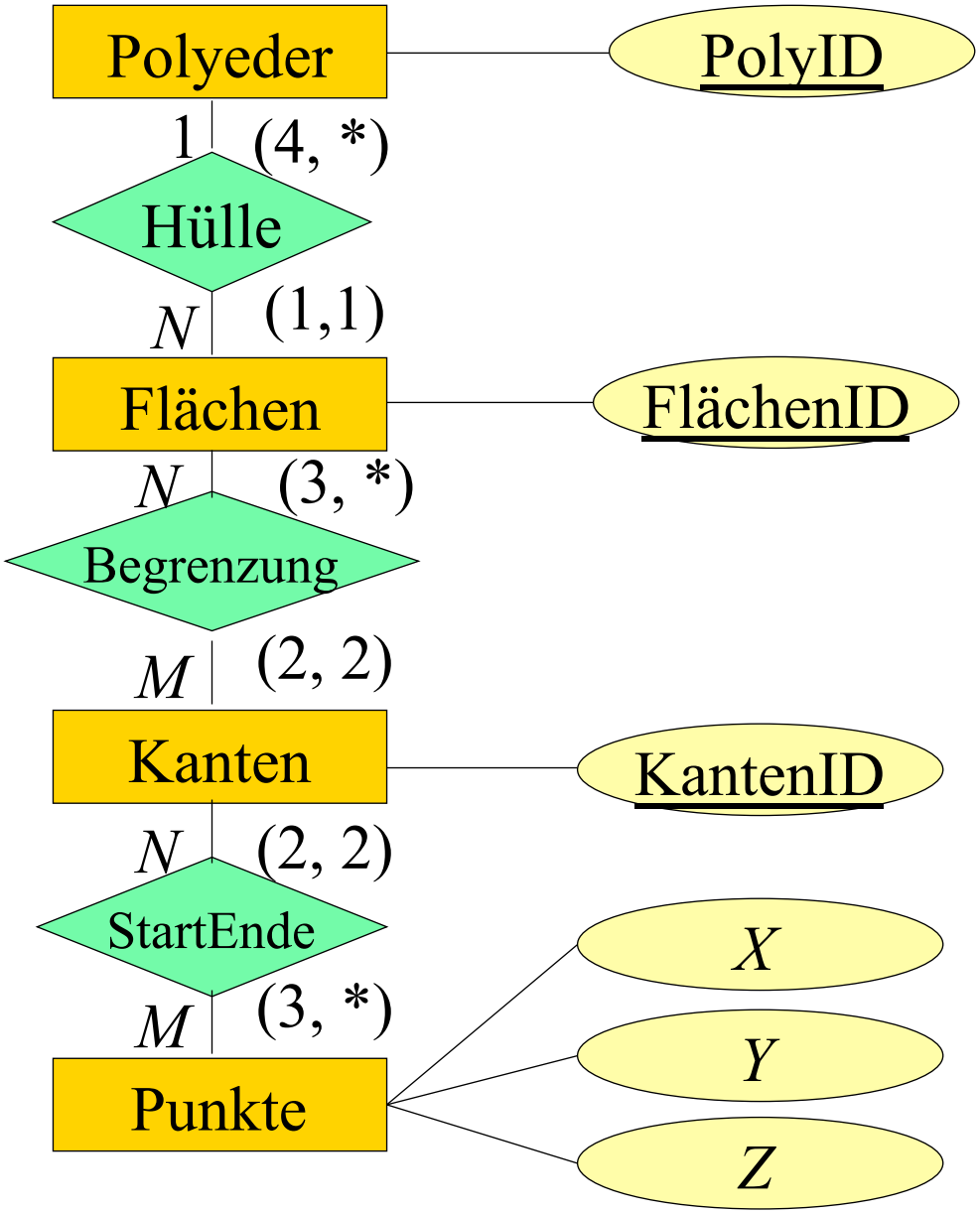
- Mindestens min_i Tupel der Art $(\ldots, e_i, \ldots) \in R$ und
- Höchstens max_i viele Tupel der Art $(\ldots, e_i, \ldots) \in R$

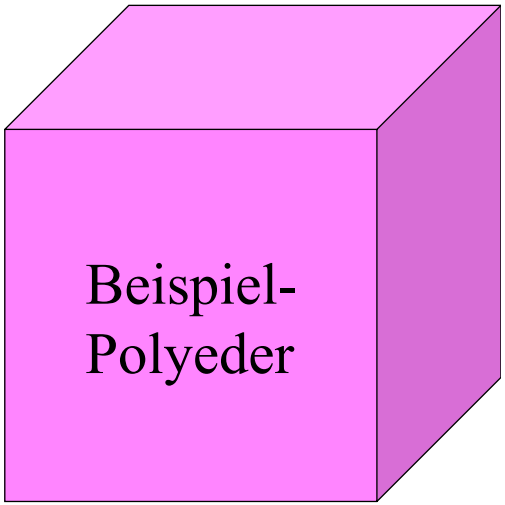
Begrenzungsflächendarstellung

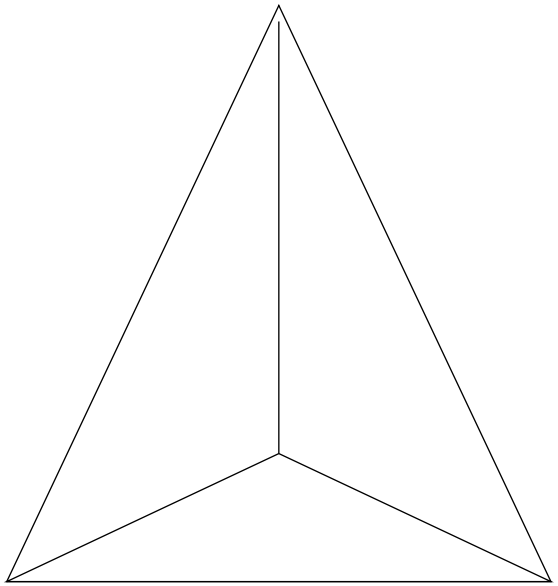




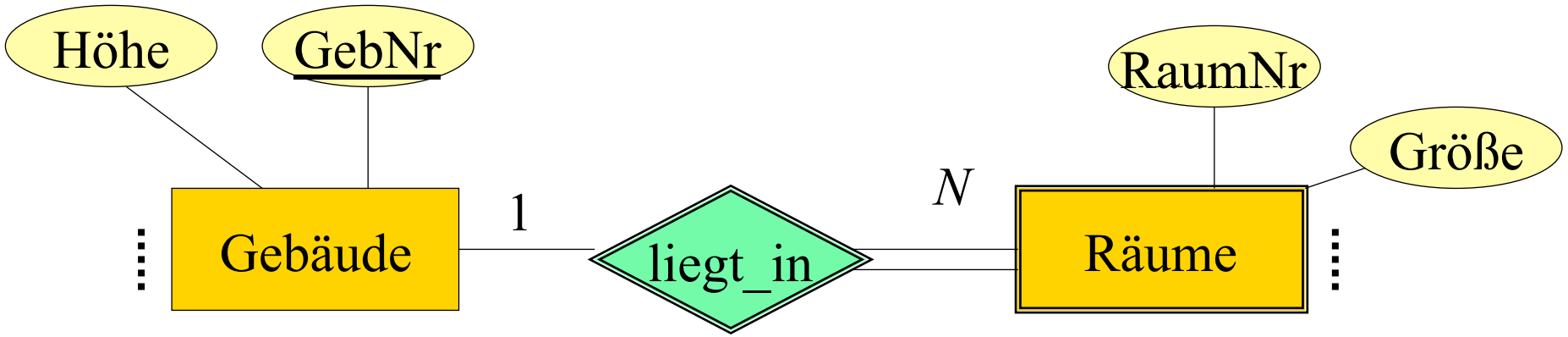
Begrenzungsflächendarstellung





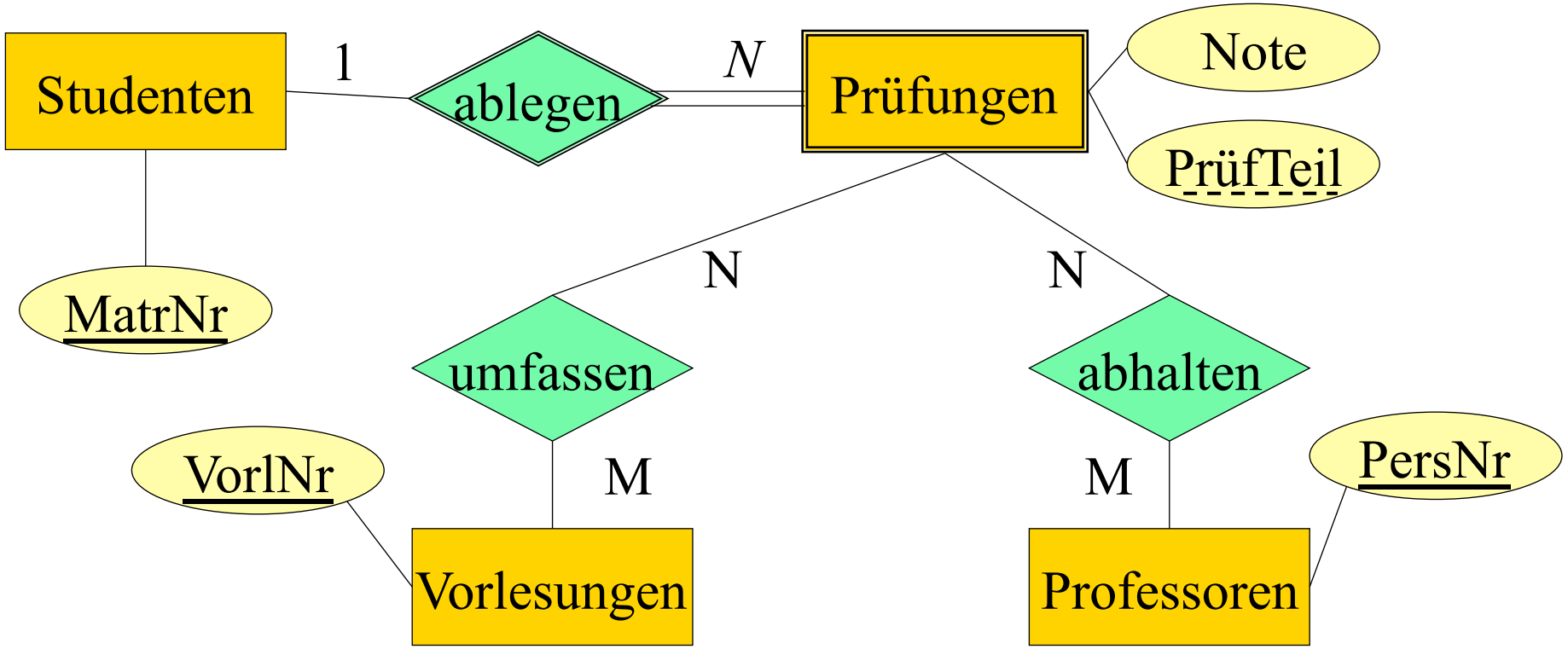


Schwache, existenzabhängige Entities



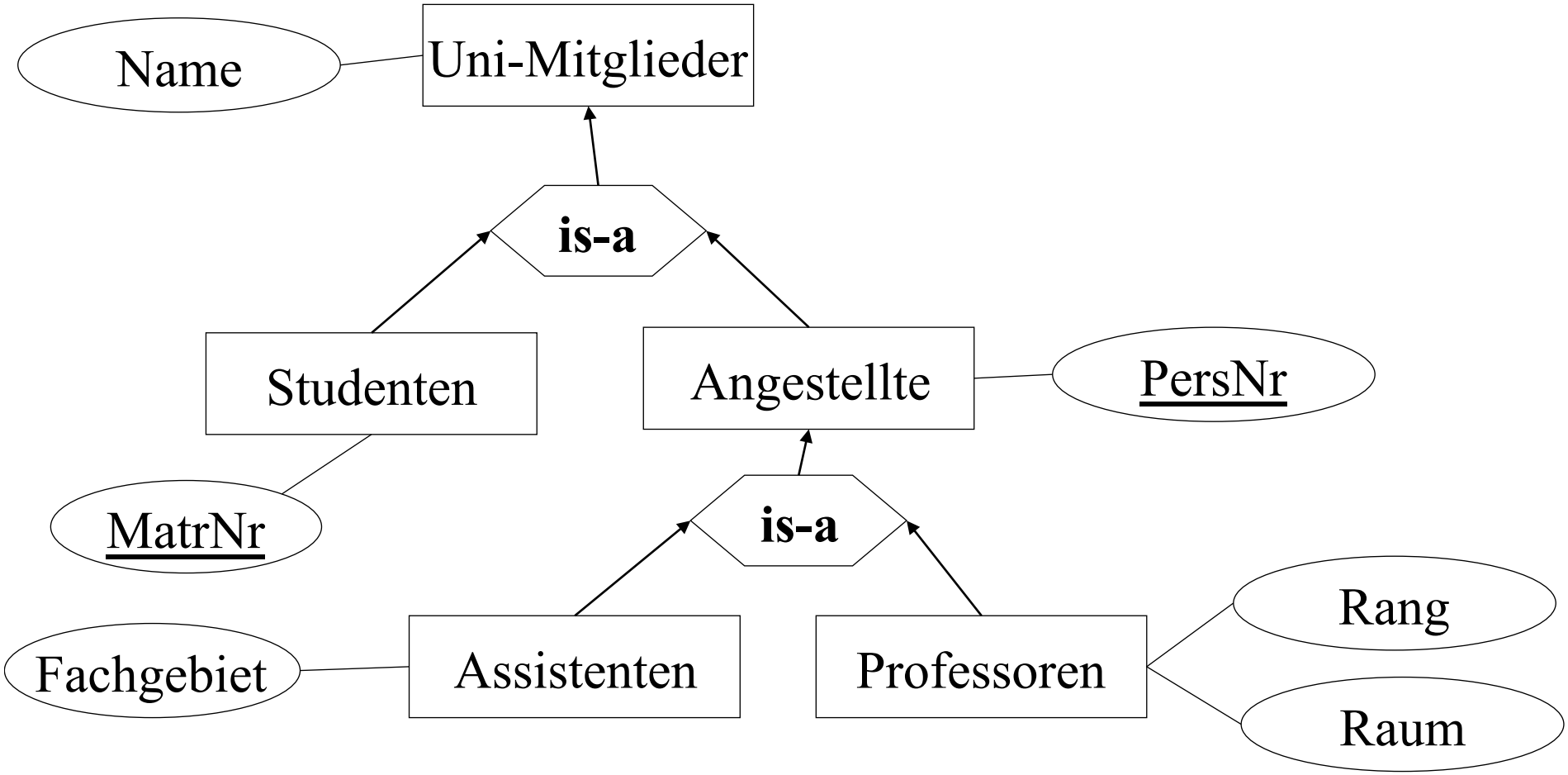
- Beziehung zwischen "starken" und schwachem Typ ist immer 1:*N* (oder 1:1 in seltenen Fällen)
- Warum kann das keine *N:M*-Beziehung sein?
- RaumNr ist nur innerhalb eines Gebäudes eindeutig
- Schlüssel ist: GebNr **und** RaumNr

Prüfungen als schwacher Entitytyp



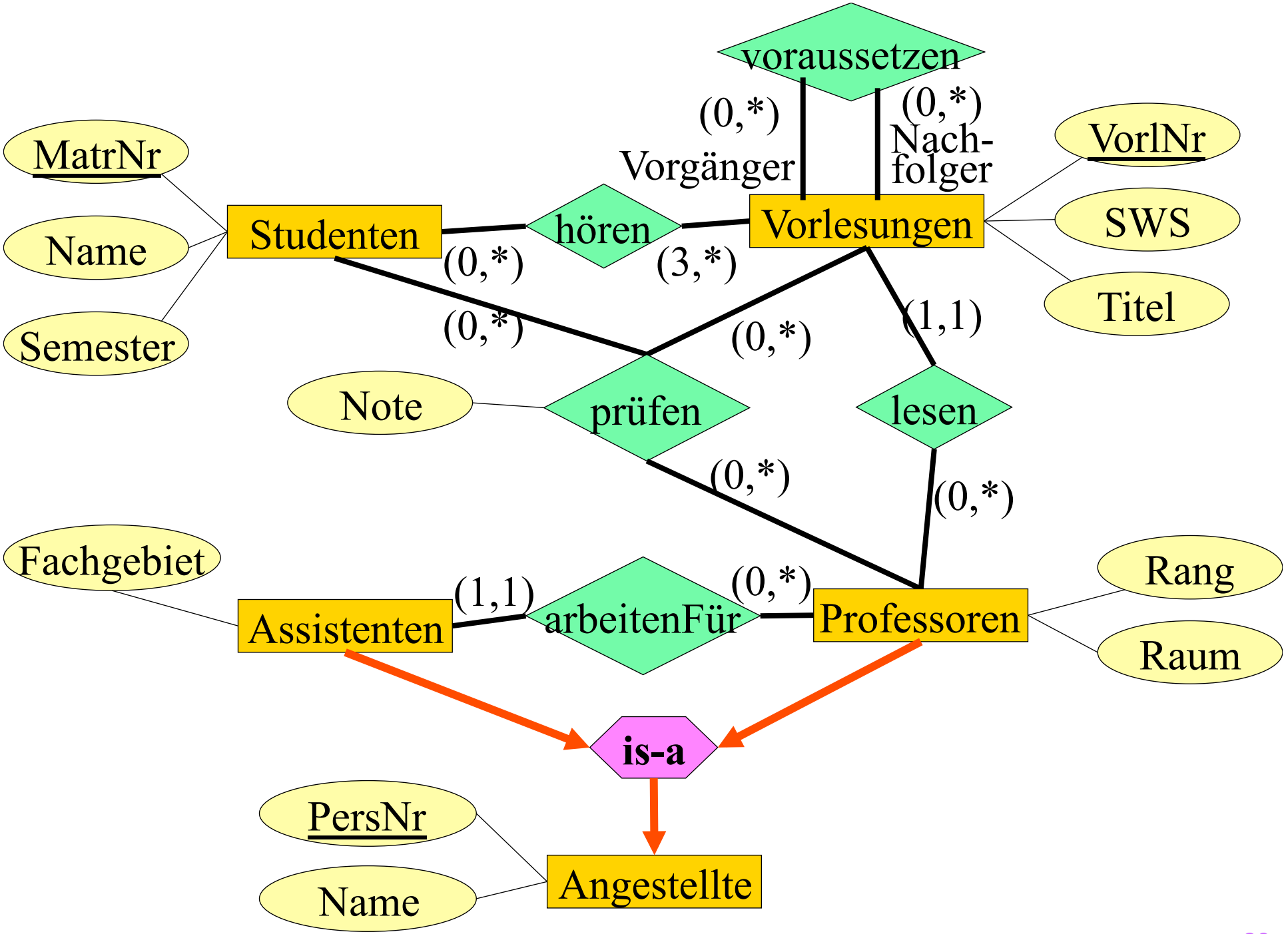
- Mehrere Prüfer in einer Prüfung
- Mehrere Vorlesungen werden in einer Prüfung abgefragt

Generalisierung

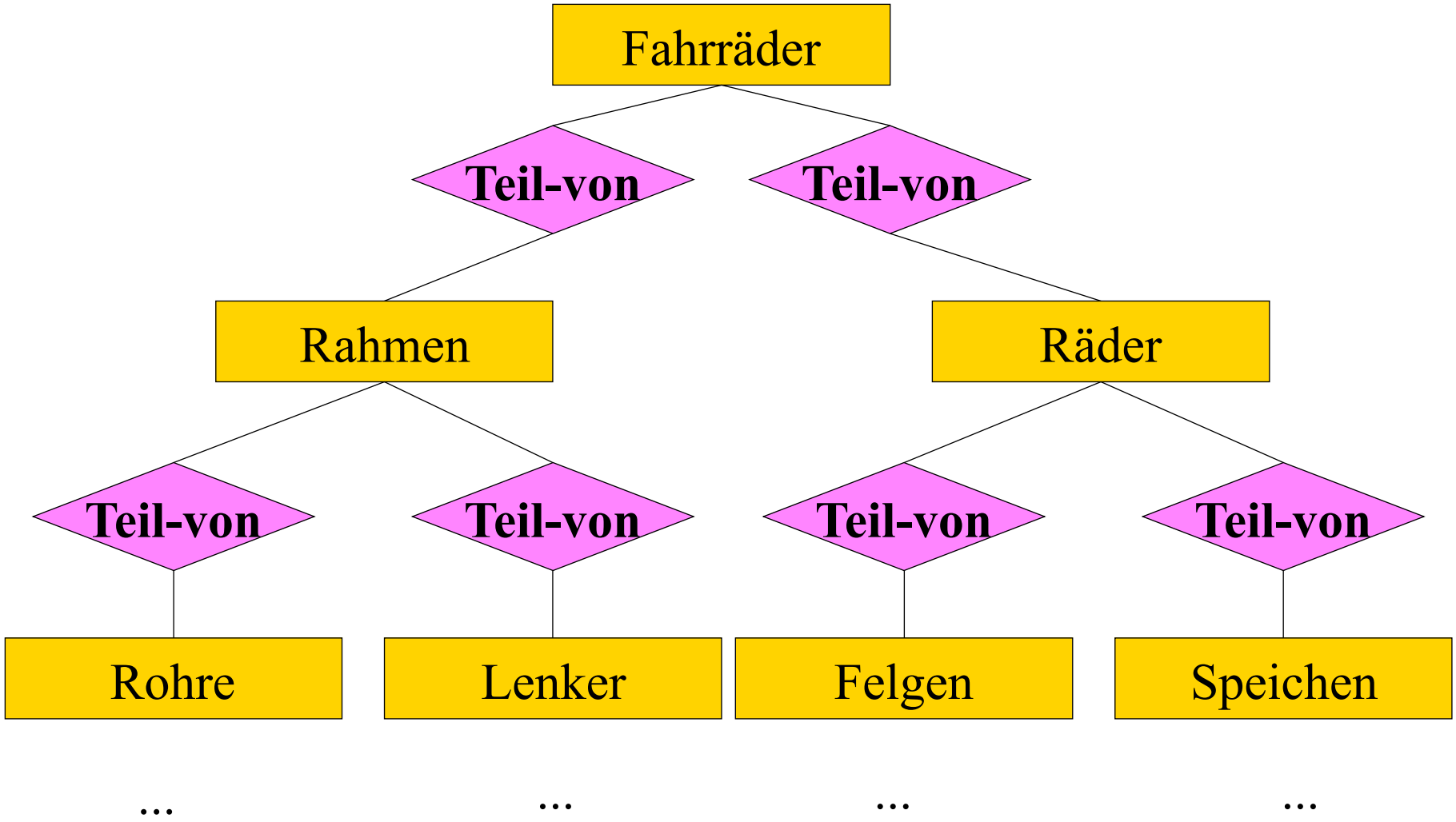


Universitätsschema mit Generalisierung und (min, max)-Markierung

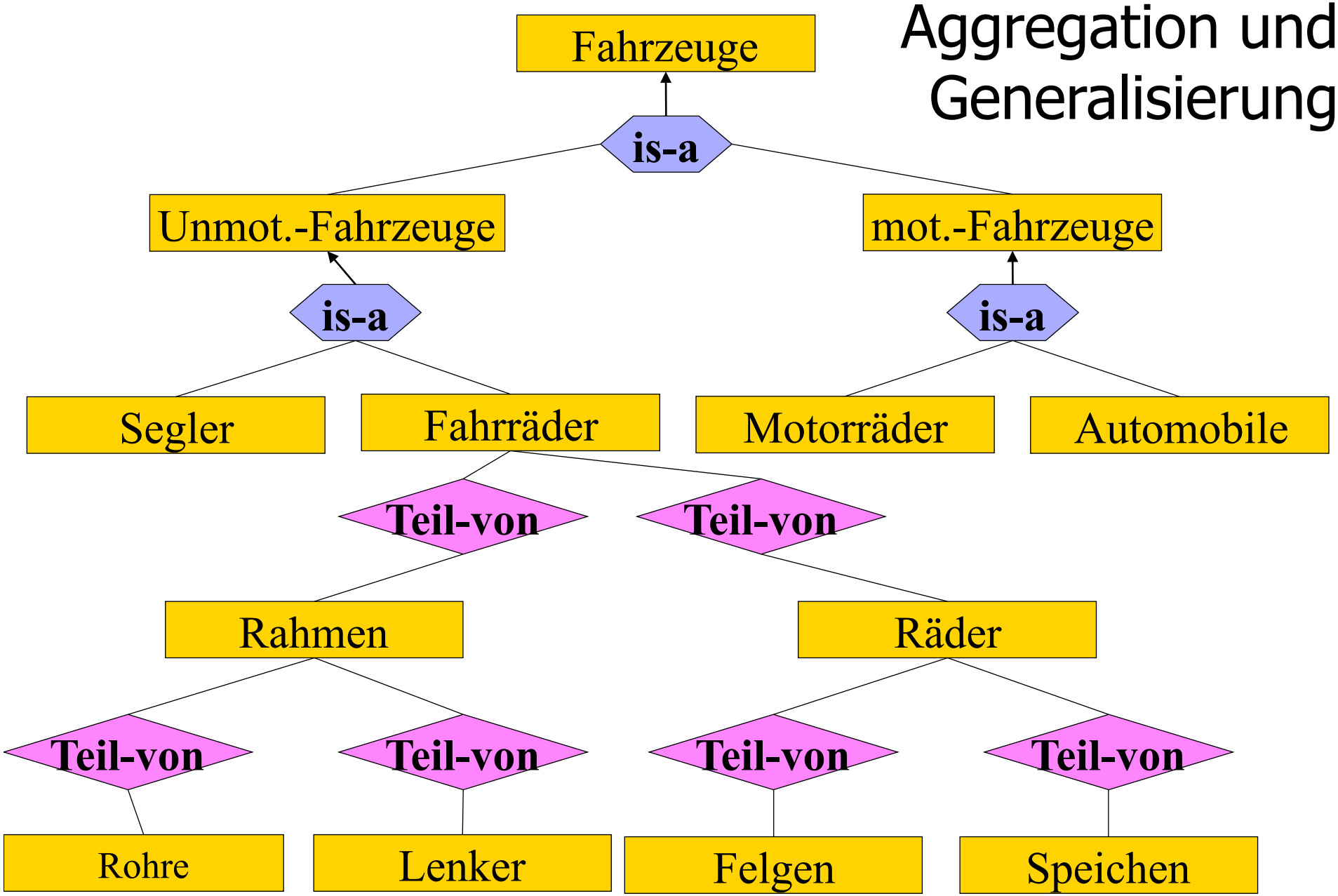
➔ Nächste Seite


voraussetzen
(0,*)
Vorgänger
(0,*)
Nach-
folger
MatrNr
Name
Semester
Studenten
hören
Vorlesungen
VorlNr
SWS
Titel
(0,*)
(3,*)
(0,*)
(0,*)
(1,1)
Note
prüfen
lesen
(0,*)
(0,*)
Fachgebiet
Assistenten
(1,1)
arbeitenFür
(0,*)
Professoren
Rang
Raum
is-a
PersNr
Name
Angestellte

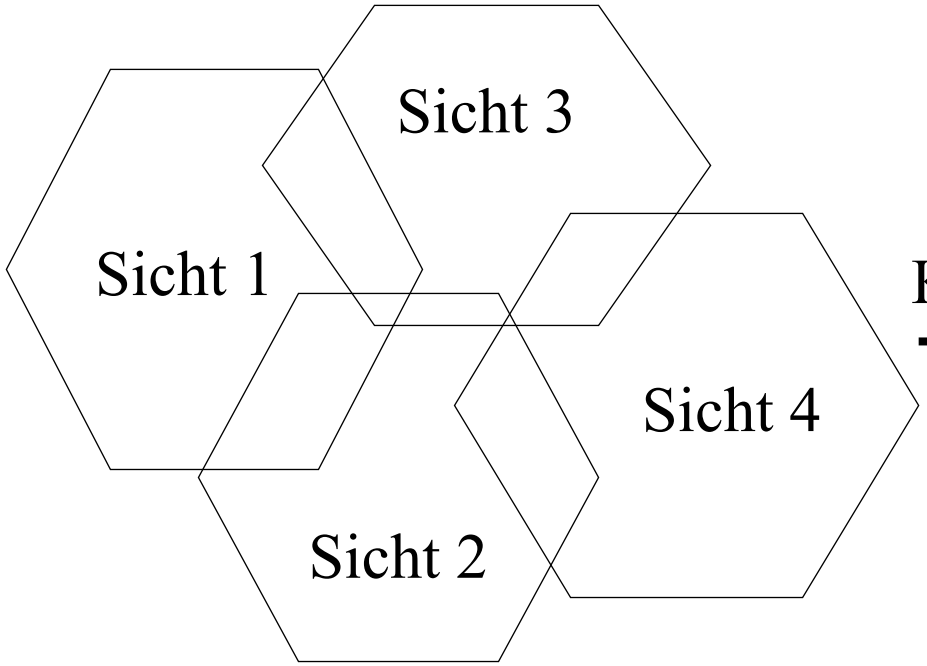
Aggregation



Aggregation und Generalisierung



Konsolidierung von Teilschemata oder Sichtenintegration

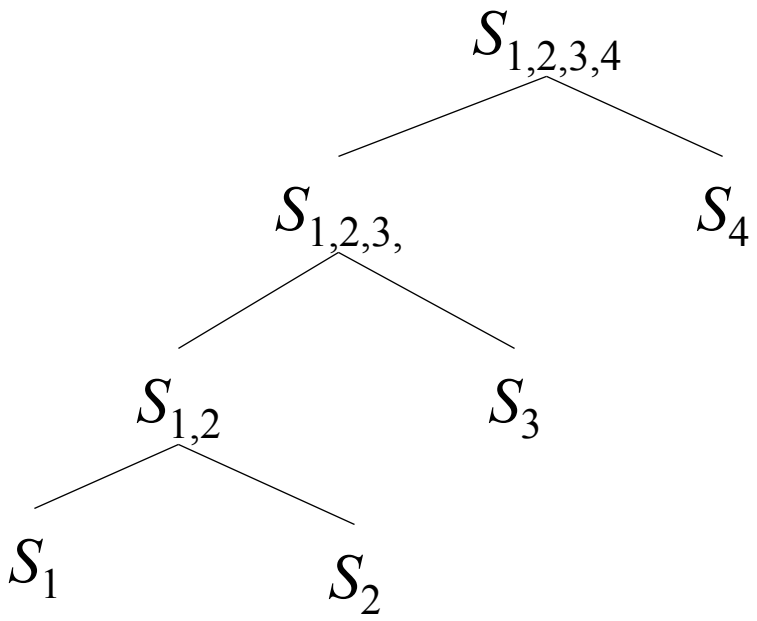


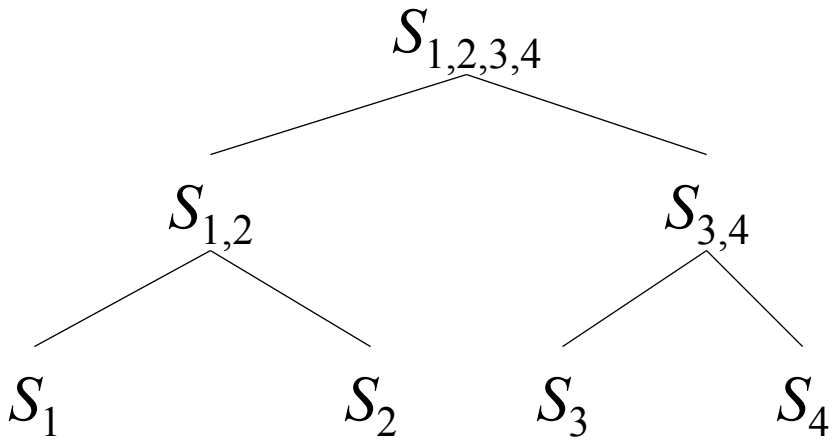
Konsoli-
dierung

Globales Schema

- **Redundanzfrei**
- **Widerspruchsfrei**
- **Synonyme bereinigt**
- **Homonyme bereinigt**

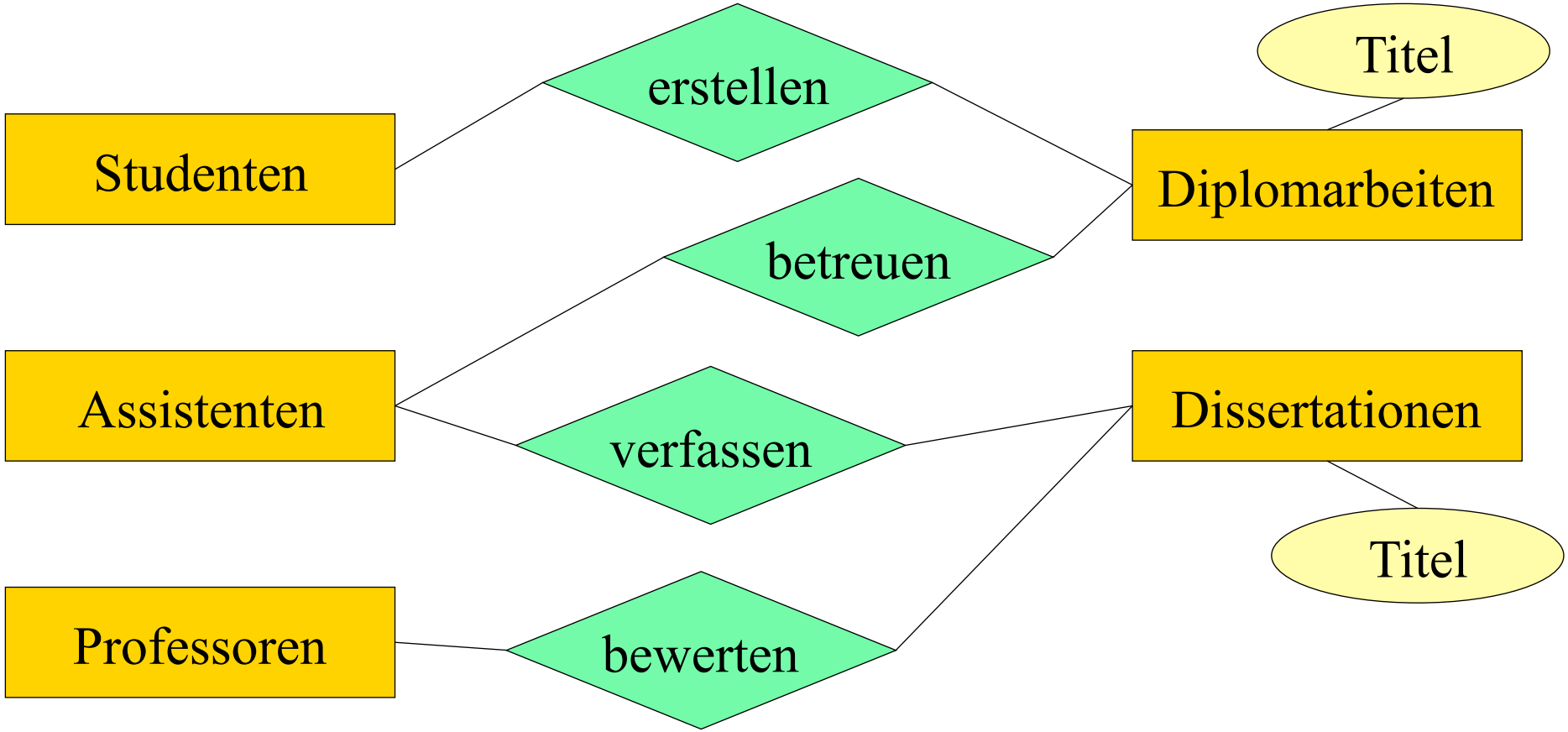
Möglicher Konsolidierungsbaum



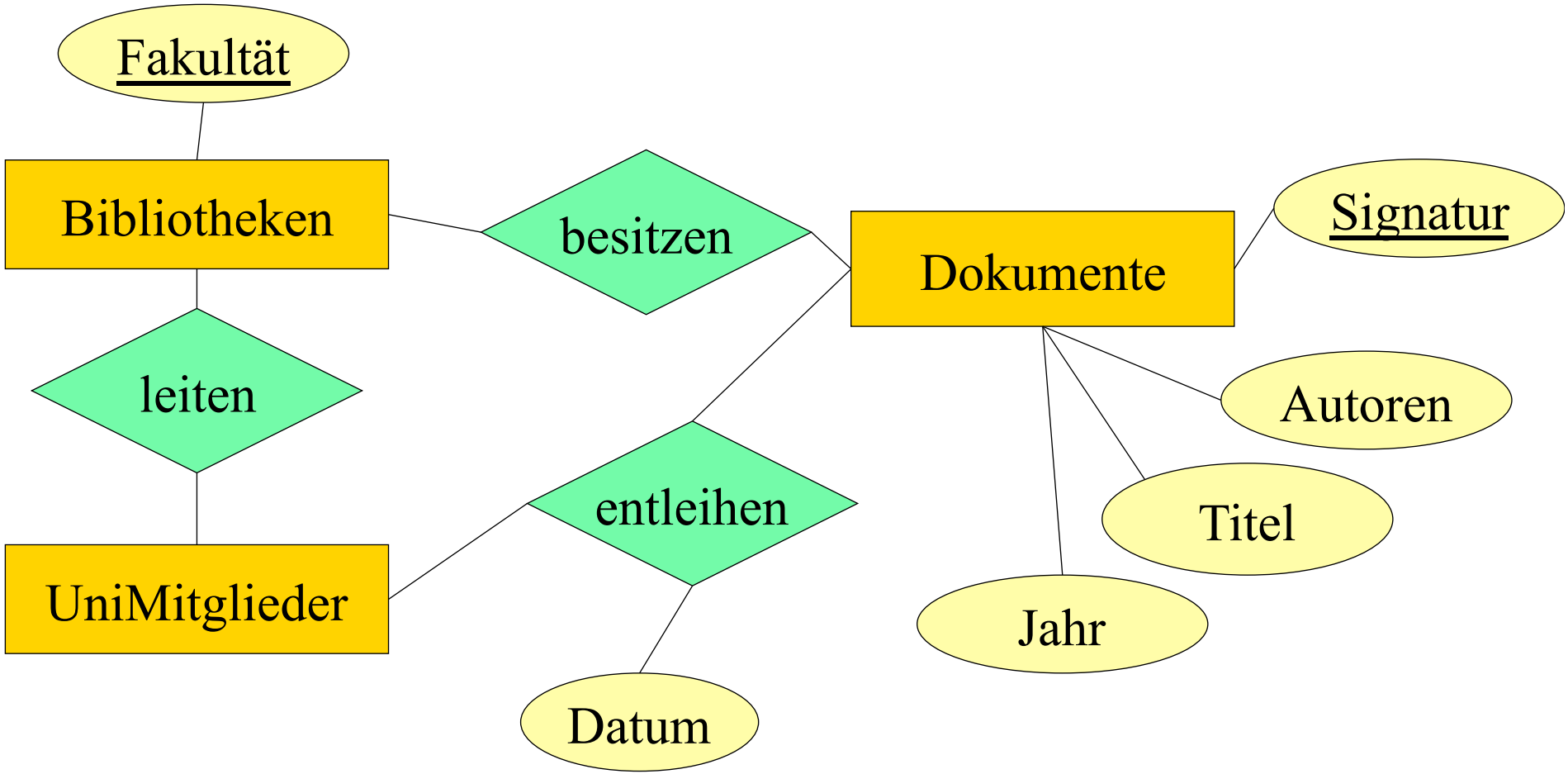


- Mögliche Konsolidierungs-bäume zur Herleitung des globalen Schemas $S_{1,2,3,4}$ aus 4 Teilschemata S_1, S_2, S_3, und S_4
  - Oben ein maximal hoher Konsolidierungsbaum
    - „links-tief“ (left-deep)
  - Unten ein minimal hoher Konsolidierungsbaum
    - Balanciert
- Beide Vorgehensweisen haben Vor- und Nachteile

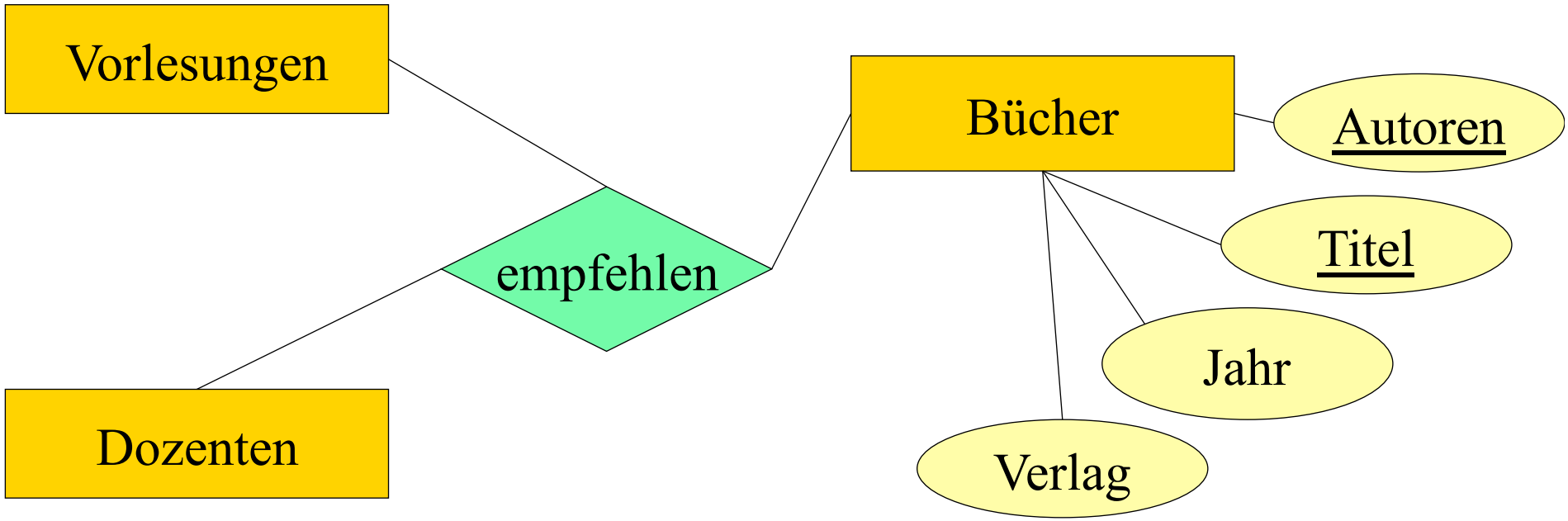
Drei Sichten einer Universitäts-Datenbank



Sicht 1: Erstellung von Dokumenten als Prüfungsleistung



Sicht 2: Bibliotheksverwaltung

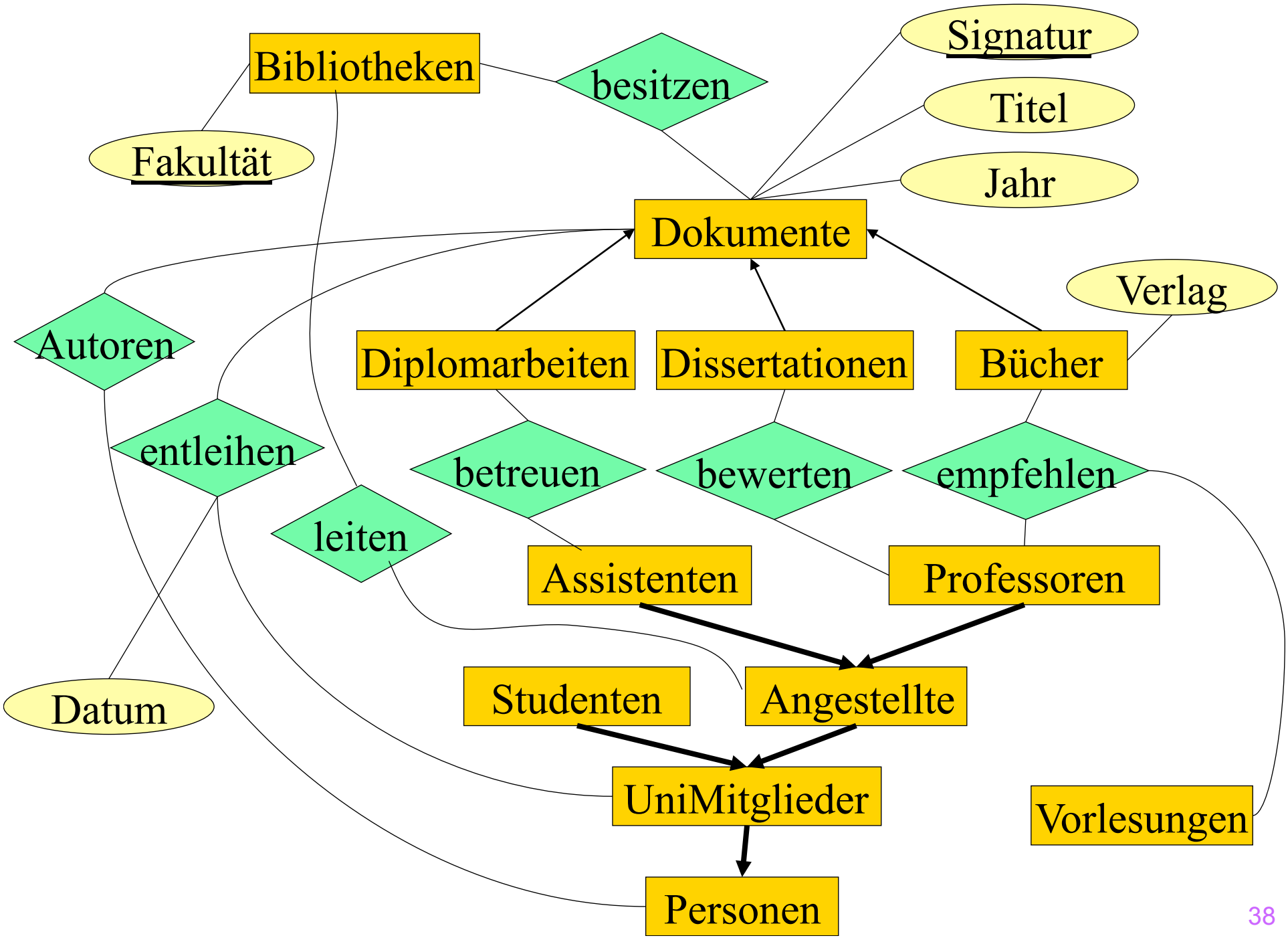


Sicht 3: Buchempfehlungen für Vorlesungen

Beobachtungen

- Die Begriffe *Dozenten* und *Professoren* sind synonym verwendet worden.
- Der Entitytyp *UniMitglieder* ist eine Generalisierung von *Studenten, Professoren* und *Assistenten*.
- Fakultätsbibliotheken werden sicherlich von *Angestellten* (und nicht von *Studenten*) geleitet. Insofern ist die in Sicht 2 festgelegte Beziehung *leiten* revisionsbedürftig, sobald wir im globalen Schema ohnehin eine Spezialisierung von *UniMitglieder* in *Studenten* und *Angestellte* vornehmen.
- *Dissertationen, Diplomarbeiten* und *Bücher* sind Spezialisierungen von *Dokumenten*, die in den *Bibliotheken* verwaltet werden.

- Wir können davon ausgehen, dass alle an der Universität erstellten *Diplomarbeiten* und *Dissertationen* in *Bibliotheken* verwaltet werden.
- Die in Sicht 1 festgelegten Beziehungen *erstellen* und *verfassen* modellieren denselben Sachverhalt wie das Attribut *Autoren* von *Büchern* in Sicht 3.
- Alle in einer Bibliothek verwalteten Dokumente werden durch die *Signatur* identifiziert.

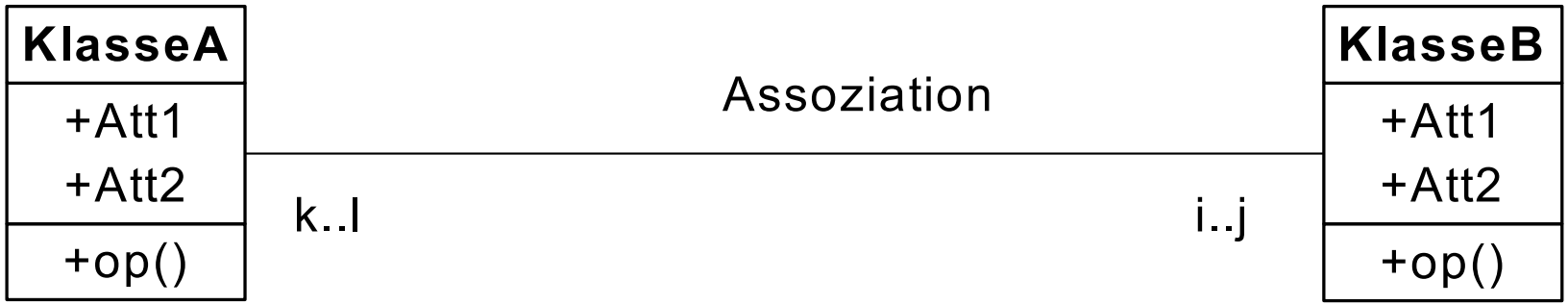

Bibliotheken
besitzen
Signatur
Titel
Jahr
Fakultät
Dokumente
Verlag
Autoren
Diplomarbeiten
Dissertationen
Bücher
entleihen
betreuen
bewerten
empfehlen
leiten
Assistenten
Professoren
Datum
Studenten
Angestellte
UniMitglieder
Vorlesungen
Personen

Datenmodellierung mit UML

- Unified Modelling Language UML
- De-facto Standard für den objekt-orientierten Software-Entwurf
- Zentrales Konstrukt ist die Klasse (class), mit der gleichartige Objekte hinsichtlich
  - Struktur (~Attribute)
  - Verhalten (~Operationen/Methoden)

  modelliert werden
- Assoziationen zwischen Klassen entsprechen Beziehungstypen
- Generalisierungshierarchien
- Aggregation

Multiplizität

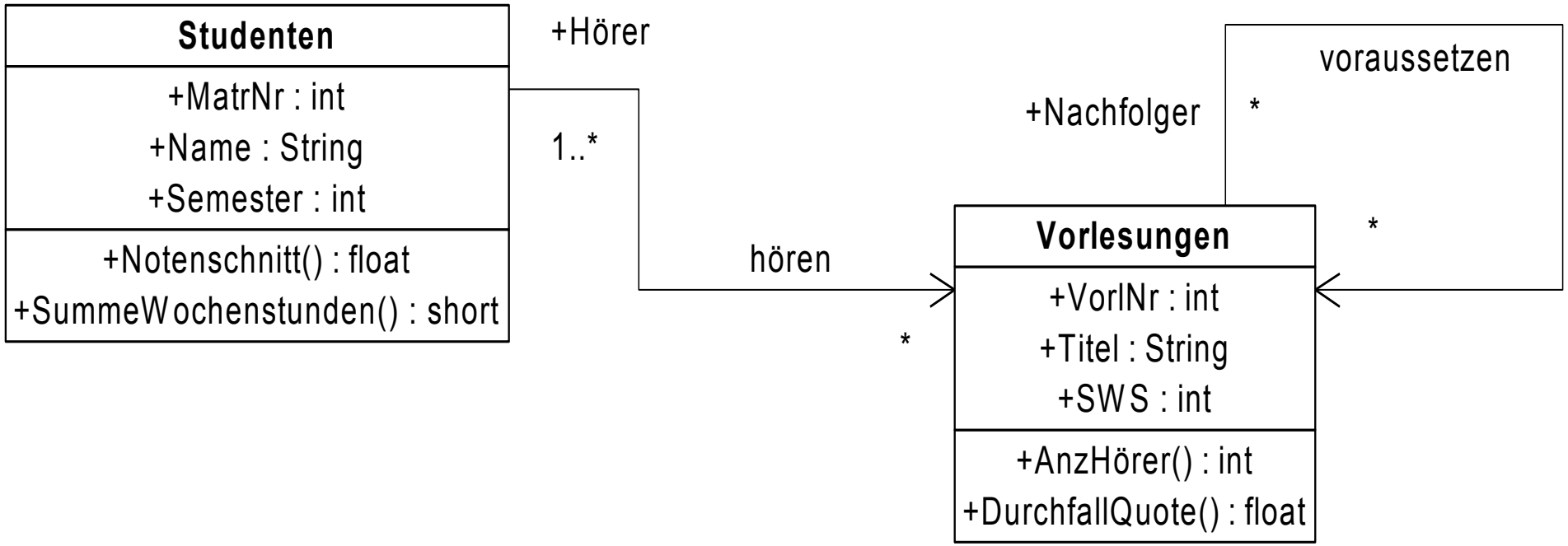


- Jedes Element von KlasseA steht mit mindestens i Elementen der KlasseB in Beziehung
- ... und mit maximal j vielen KlasseB-Elementen

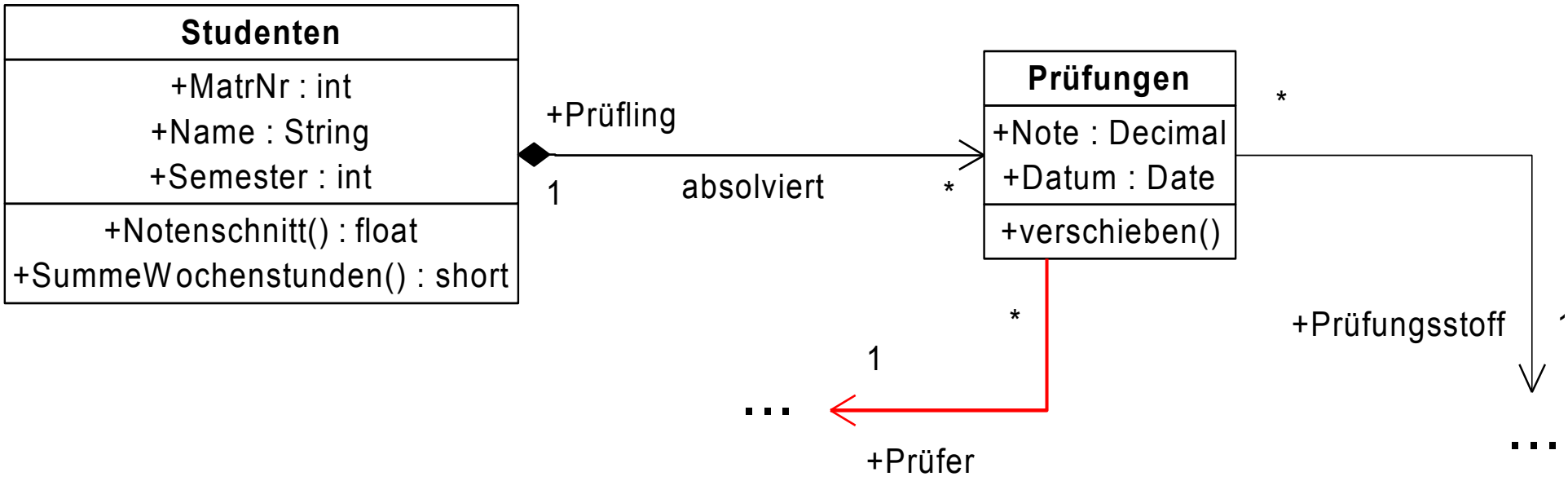
- Analoges gilt für das Intervall k..l

- Multiplizitätsangabe ist analog zur Funktionalitätsangabe im ER-Modell
  - **Nicht** zur (min,max)-Angabe: **Vorsicht!**

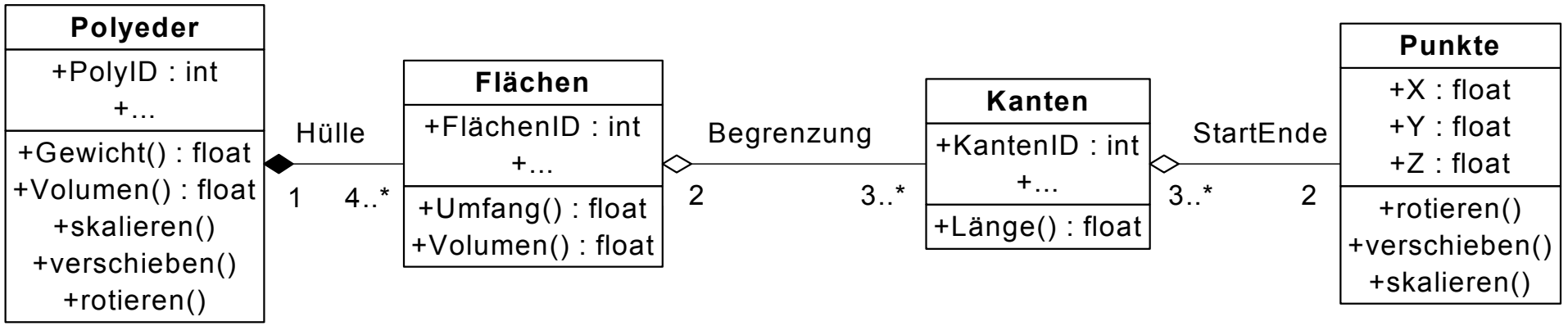
Klassen und Assoziationen

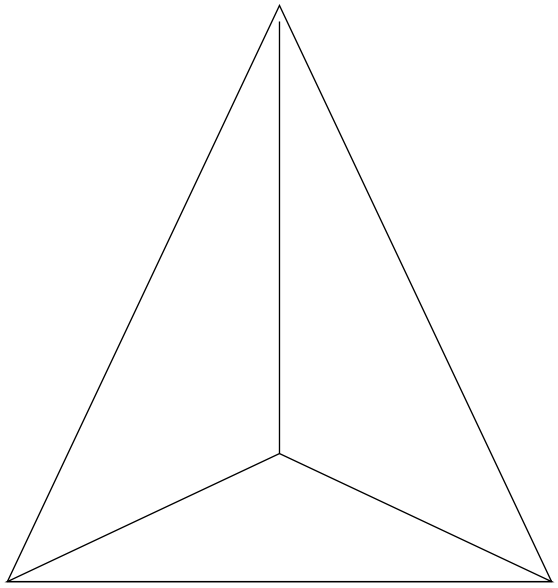


Aggregation

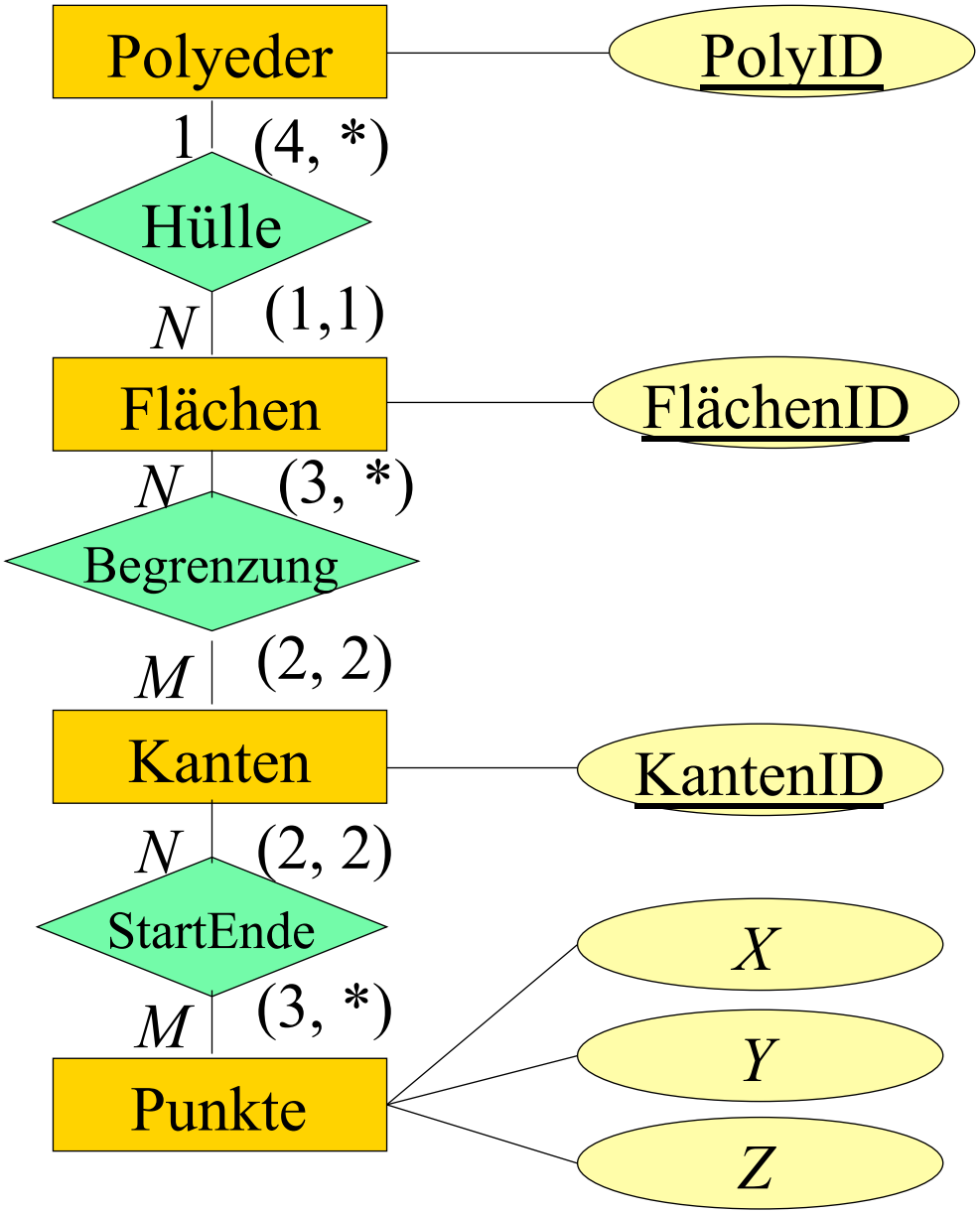


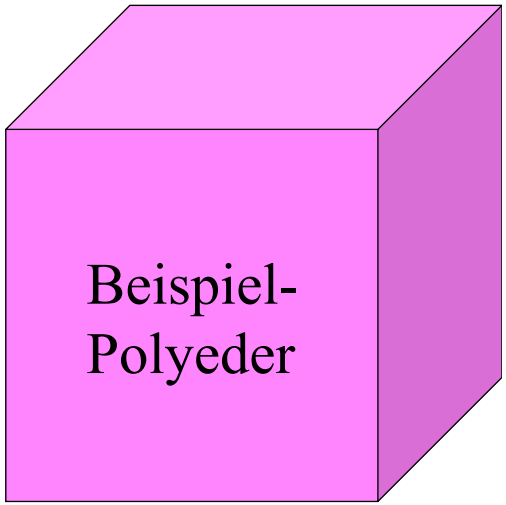
Begrenzungsflächenmodellierung von Polyedern in UML

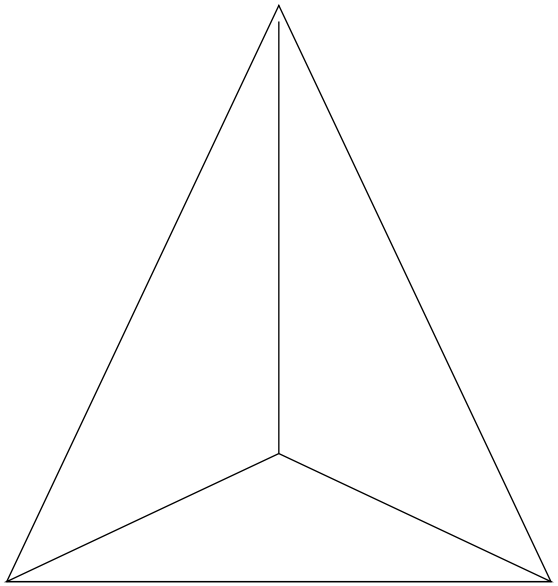


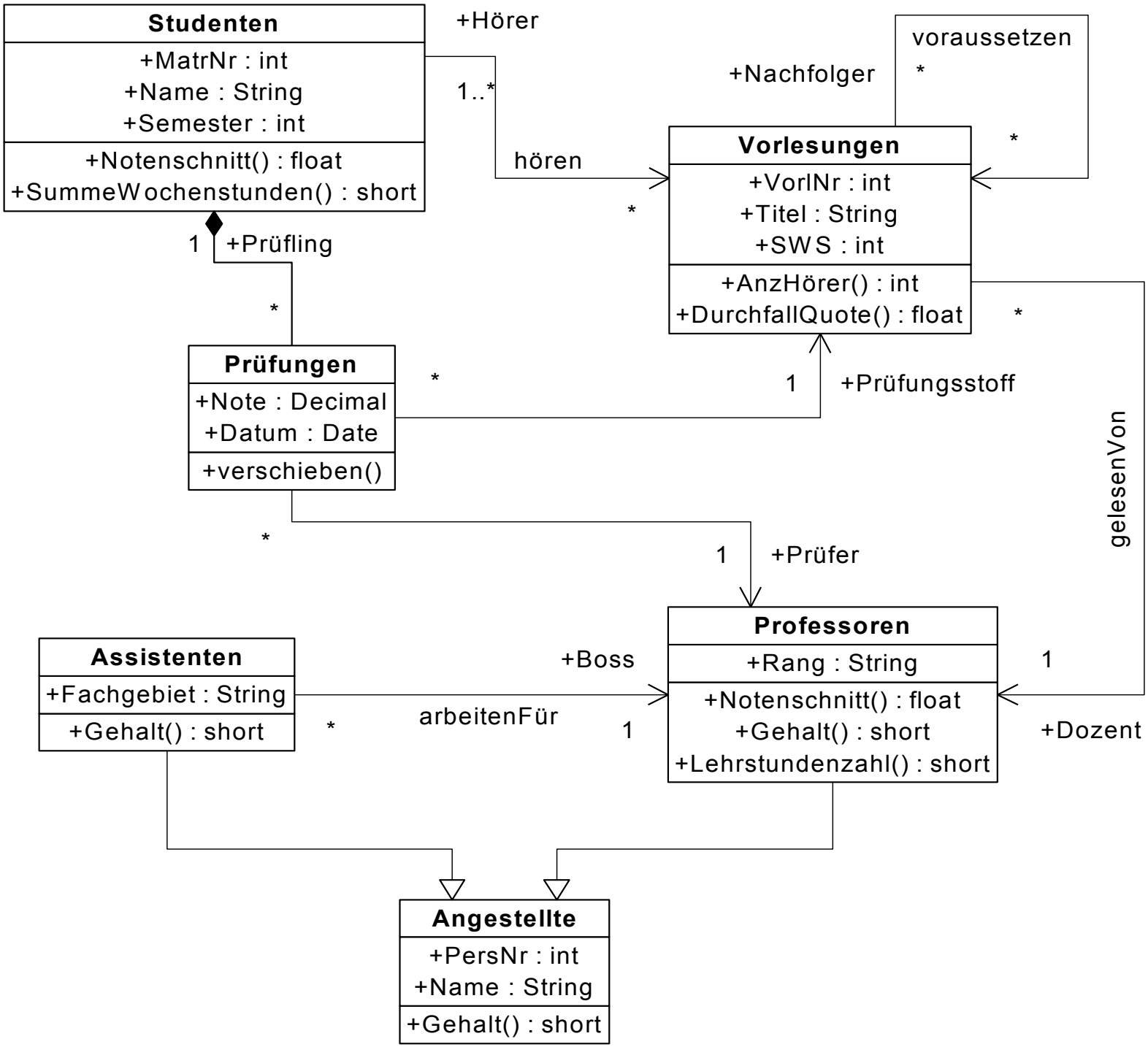


Begrenzungsflächendarstellung

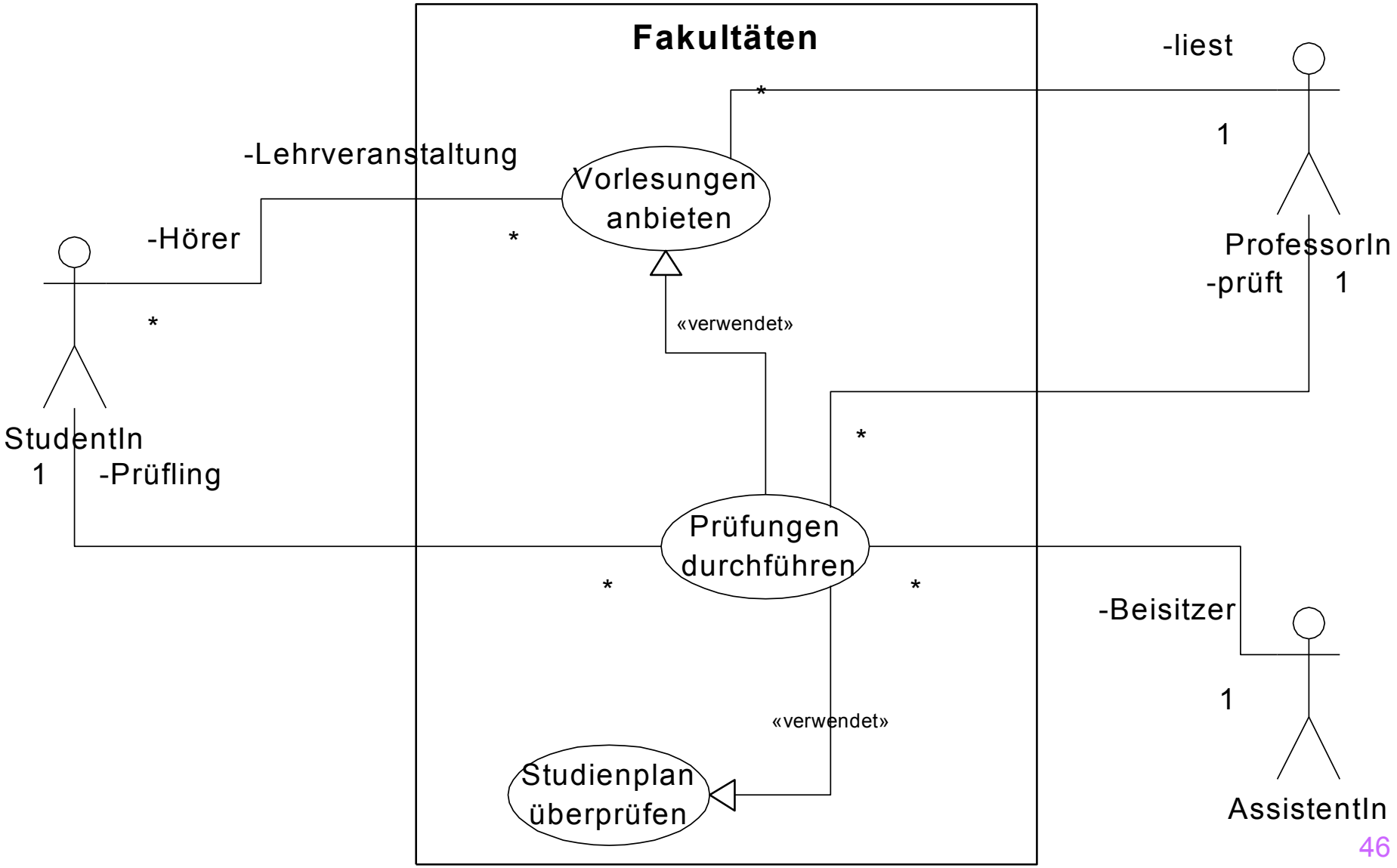




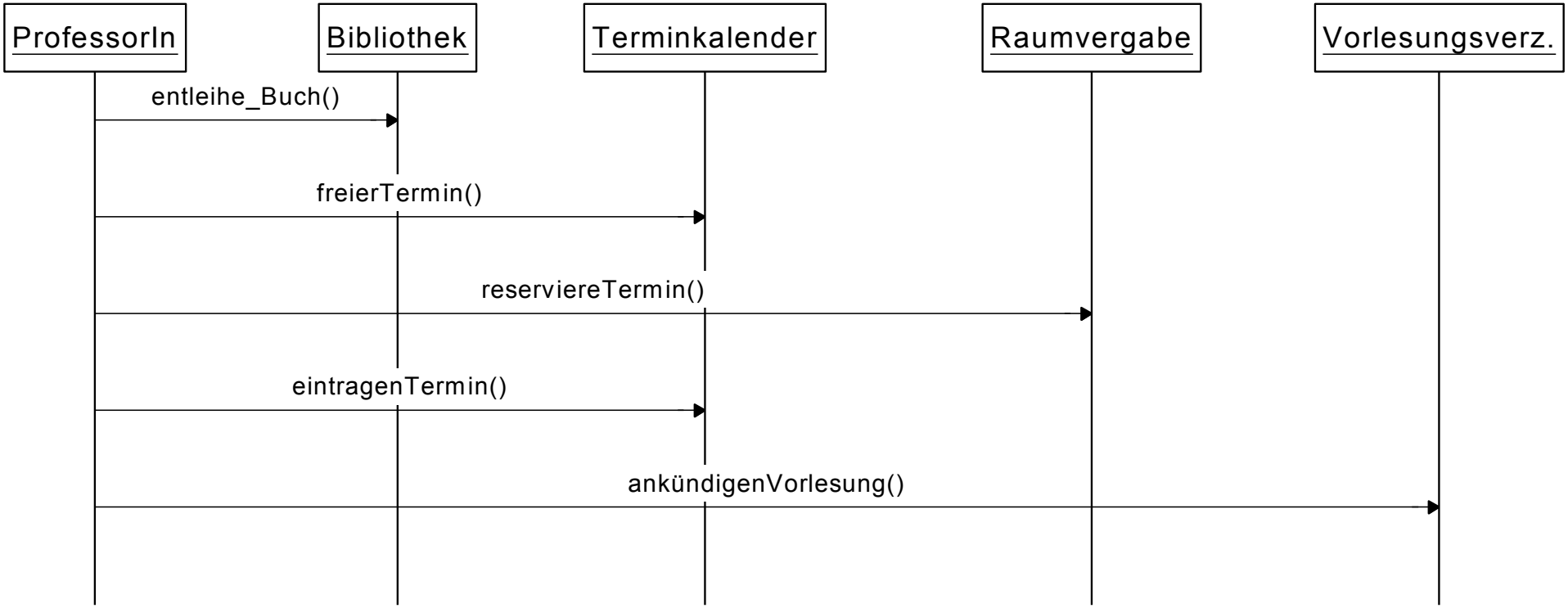



Studenten
+MatrNr : int
+Name : String
+Semester : int
+Notenschnitt() : float
+SummeWochenstunden() : short
+Hörer
1..*
hören
*
+Nachfolger
voraussetzen
*
*
Vorlesungen
+VorlNr : int
+Titel : String
+SWS : int
+AnzHörer() : int
+DurchfallQuote() : float
1
+Prüfling
*
Prüfungen
+Note : Decimal
+Datum : Date
+verschieben()
*
1
+Prüfungsstoff
*
gelesenVon
*
1
+Prüfer
Professoren
+Rang : String
+Notenschnitt() : float
+Gehalt() : short
+Lehrstundenzahl() : short
1
+Dozent
Assistenten
+Fachgebiet : String
+Gehalt() : short
+Boss
*
arbeitenFür
1
Angestellte
+PersNr : int
+Name : String
+Gehalt() : short

Anwendungsfälle (use cases)



Interaktions-Diagramm:
Modellierung komplexer Anwendungen



Interaktions-Diagramm: *Prüfungsdurchführung*