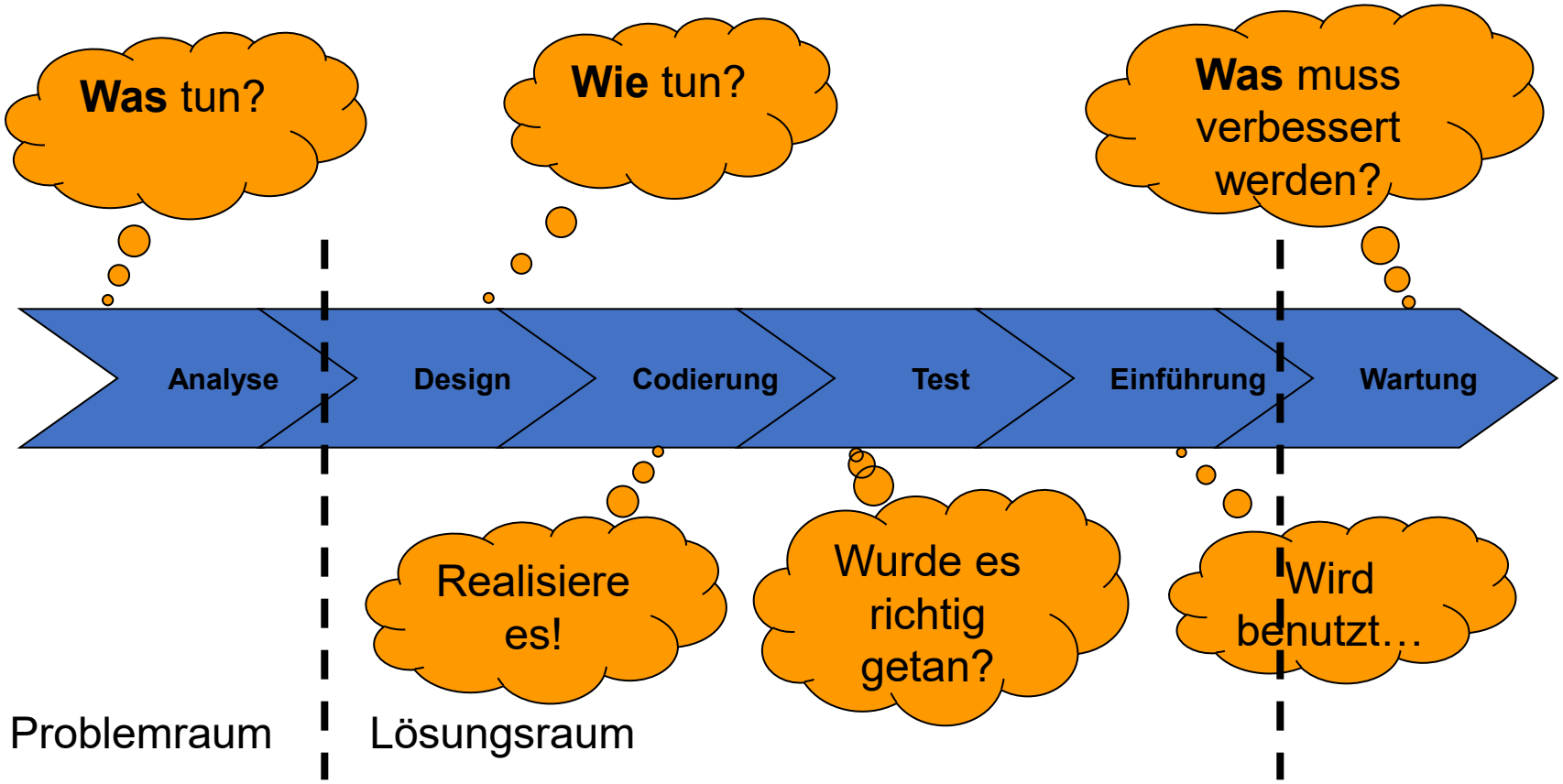
Data Engineer

Requirement Engineering:

Aufgaben, Ziele und Vorgehensweise in der Anforderungsanalyse

DB-Design Teil 1 - ERM

Aktivitäten des ….-Lebenszyklus



Definition

- Die **Anforderungsanalyse** ist üblicherweise die Startphase im ….-Lebenszyklus.

- **Requirements Engineering** steht für das systematische Vorgehen bei der **Erfassung**, **Beschreibung**, **Prüfung** und **Verfolgung** von **Anforderungen** für ein zu entwickelndes System.

- Der **Systemanalytiker** (Requirements Engineer) ist dafür zuständig.

Literatur

- Pohl, Klaus; Rupp, Chris:
  Basiswissen Requirements Engineering

  Aus und Weiterbildung zum *Certified Professional for Requirements Engineering* Foundation Level nach IREB-Standard.
  dpunkt, Heidelberg, 2009.

Anforderungsphasen

- **Kundenanforderungen**
  - Definition der Systemeigenschaften (**WAS**)

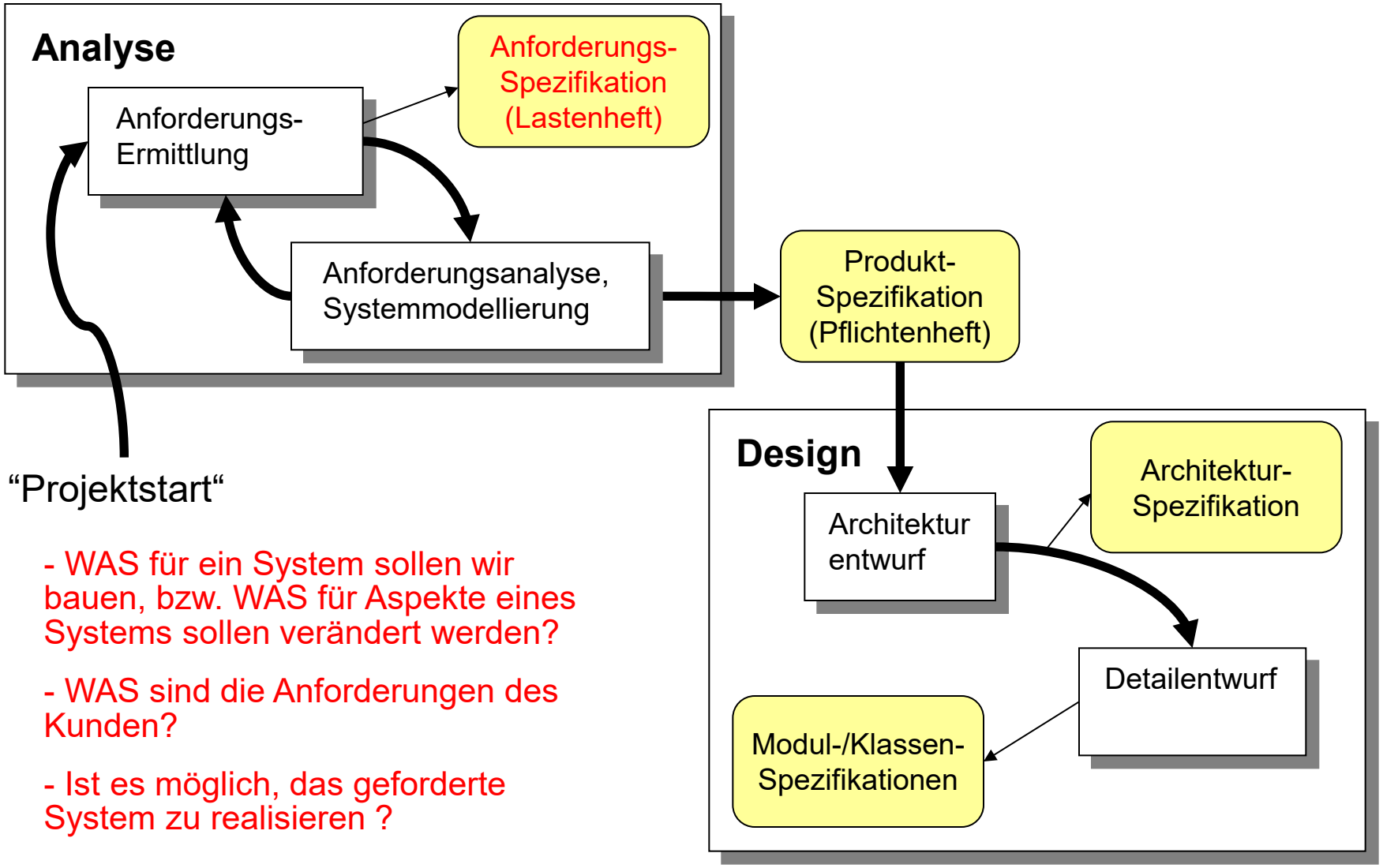
  ➔ Lastenheft

- **Systemanforderungen**
  - Technische Spezifikation des Systems (**WIE**)

  ➔Pflichtenheft

- **Pflichtenheft wird Vertragsgrundlage!**

Analysephase ➔ Designphase



- WAS für ein System sollen wir bauen, bzw. WAS für Aspekte eines Systems sollen verändert werden?

- WAS sind die Anforderungen des Kunden?

- Ist es möglich, das geforderte System zu realisieren ?

Relevanz von RE

- **Requirements Engineering** (RE) ist eine Schlüsseldisziplin der Systementwicklung und entscheidet maßgeblich über den Erfolg oder Misserfolg eines Projekts.

- Die perfekte Analyse ist nicht möglich, aber sie muss das Ziel sein !!!

- Ein guter Requirements-Engineer benötigt bestimmte persönliche Eigenschaften:

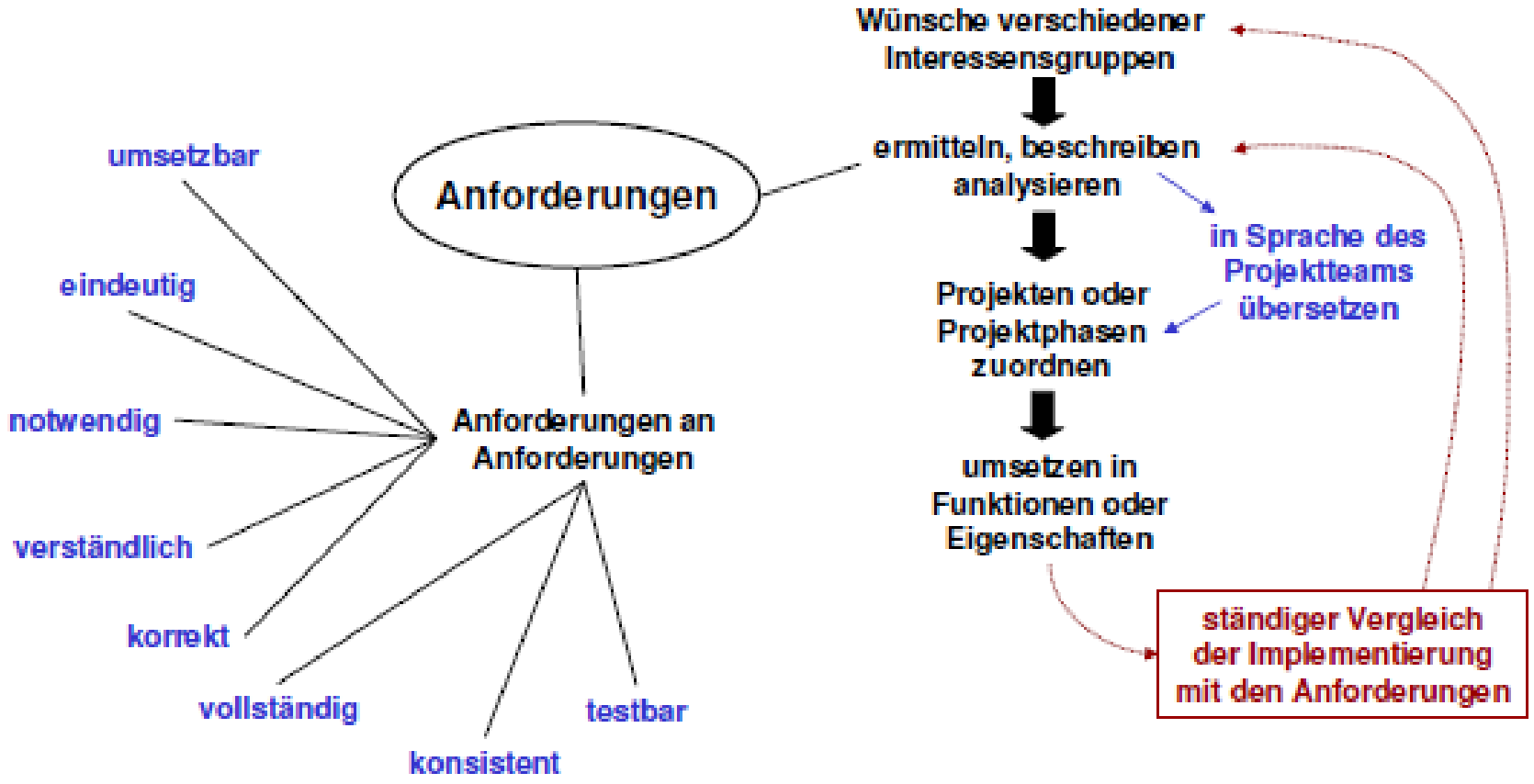
  Analytisch, Selbstbewusst, Emphatisch, Abstraktes Denken, Neugierig, Kommunikativ, Penetrant, präziser schriftlicher Ausdruck.

Definition

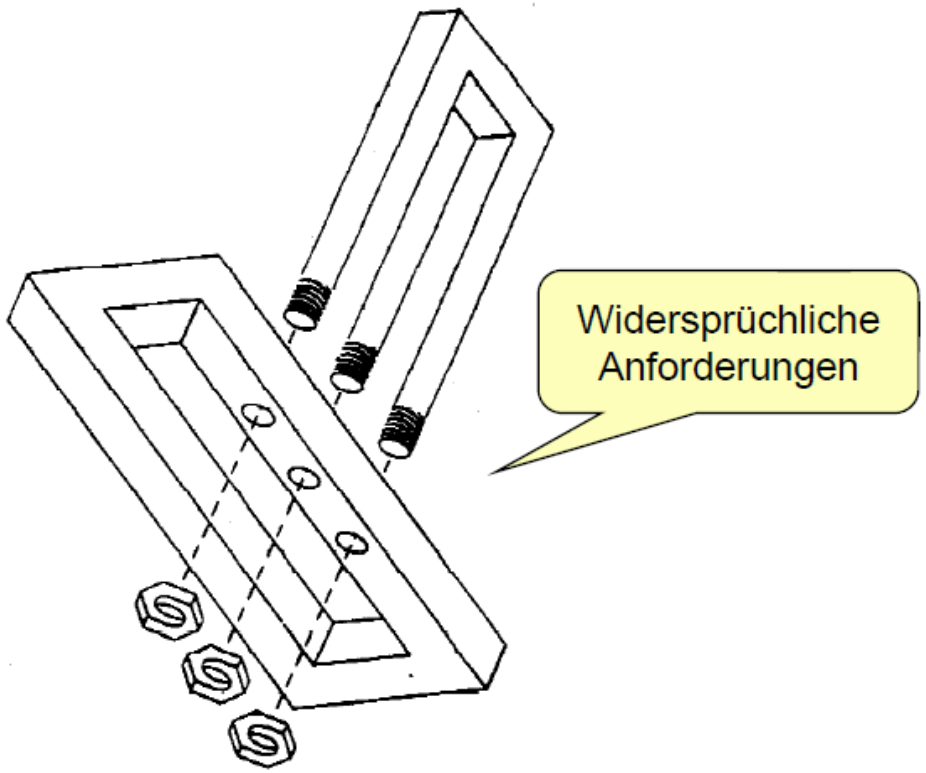
- Eine **Anforderung** (Requirement) ist eine **funktionale** oder **nichtfunktionale** Vorgabe, die ein System erfüllen soll bzw. eine technische oder formale Restriktion, die von außen vorgegeben und zu beachten ist.

- Die Definition der Anforderung muss als Übereinkunft zwischen Auftraggeber und Auftragnehmer formuliert sein. **Eindeutigkeit** ist dabei ein wesentliches Kriterium.

Anforderungen an Anforderungen

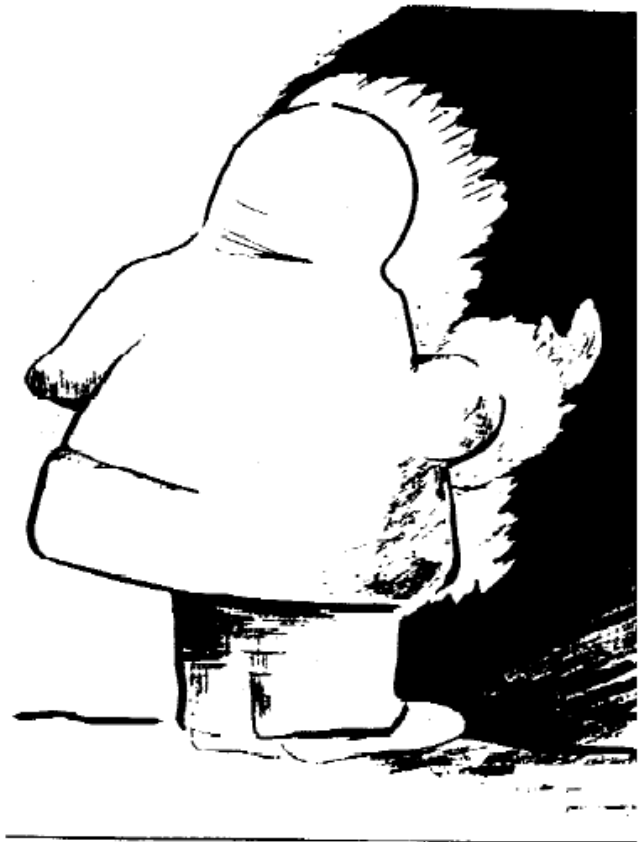


Problem: Widersprüchliche Anforderungen



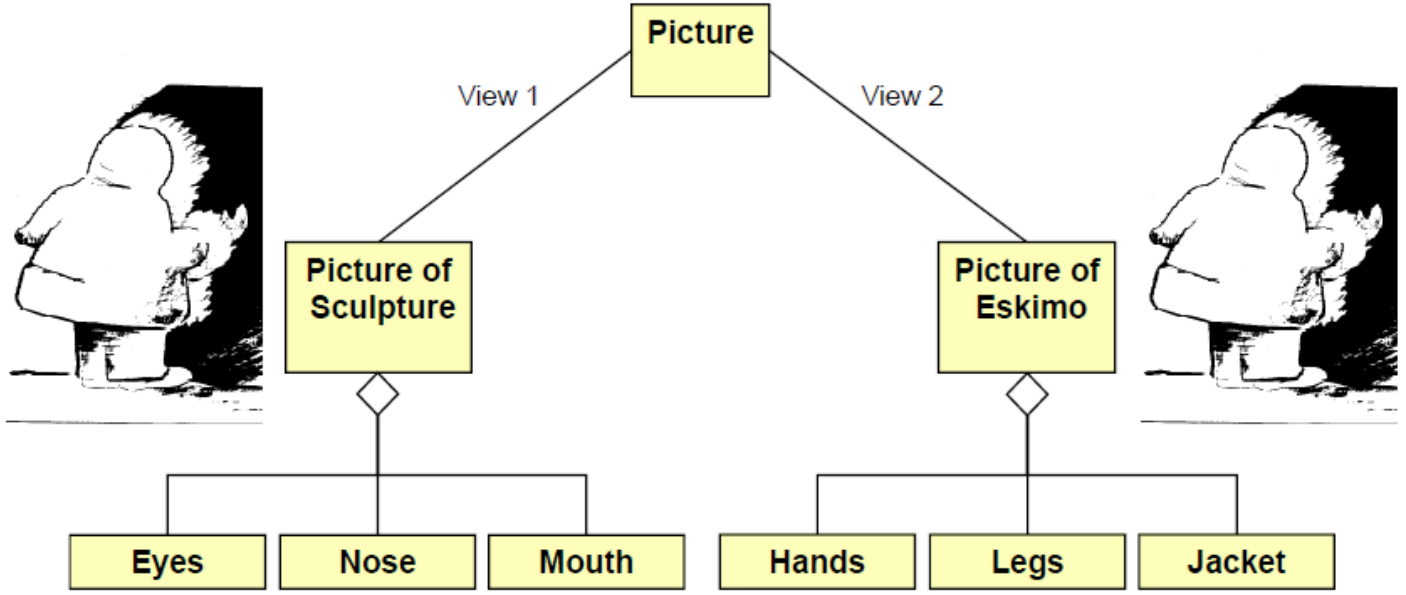
➔ Nicht realisierbar !

Problem: Mehrdeutige Anforderungen



Problem: Mehrdeutige Anforderungen

- Mehrere Interpretationen möglich



➔ Was ist richtig ?

Problem: Trennung von Anforderung und Lösung

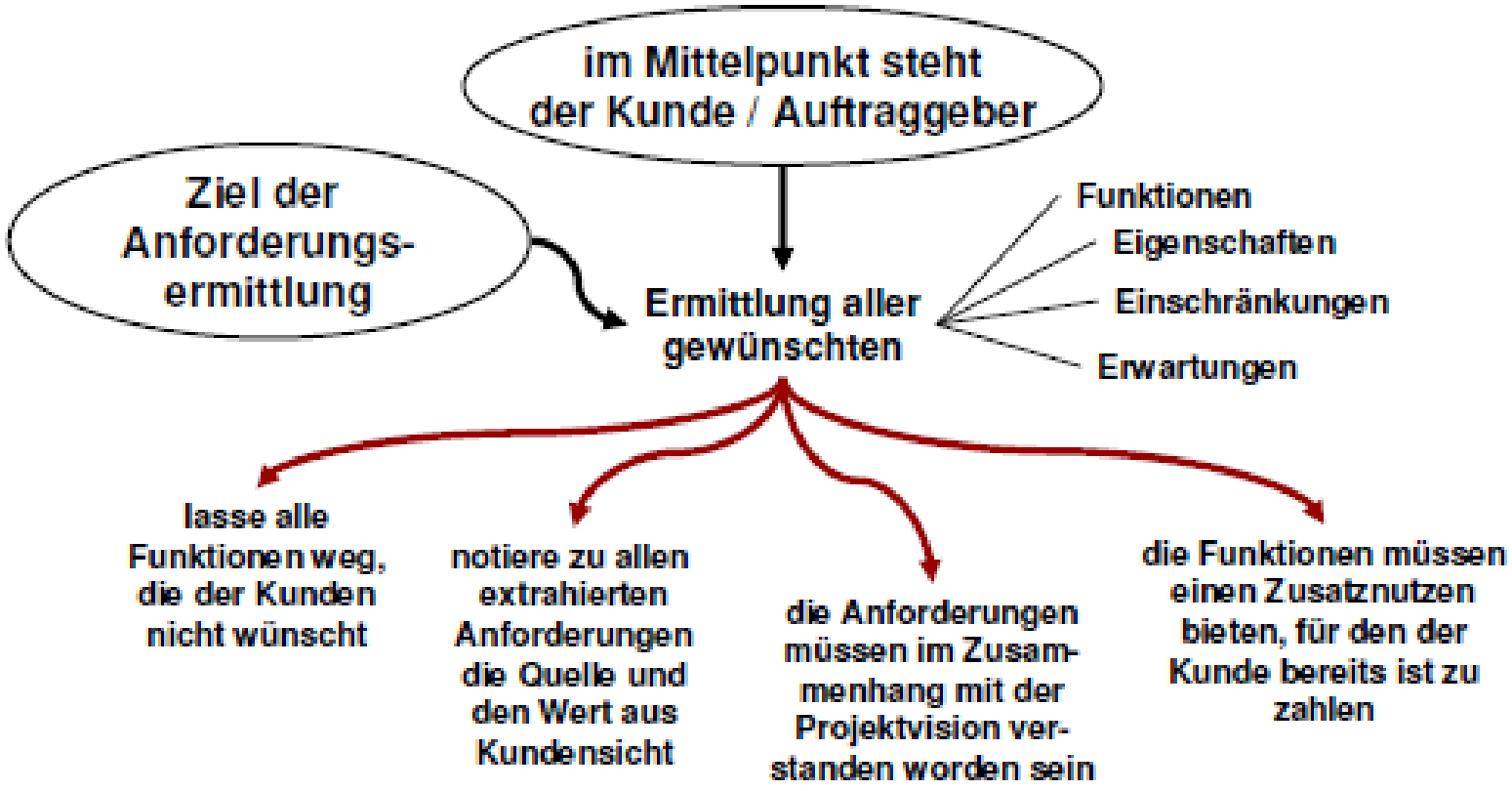
- **WAS = Spezifikation, WIE = Entwurf**
- **/REQ 1/ „Das System druckt eine wahlweise nach Namen oder Land alphabetisch sortierte Liste von Teilnehmern mit Nummer, Name, Vorname, Affiliation und Land. Auf jeder Seite sind unten links das Erstellungsdatum und unten rechts die Seitenzahl aufgedruckt.“**
- Ist dieser Satz eine Anforderung oder eine Entwurfsentscheidung?
- ➔ WAS vs. WIE ist kontextabhängig und liefert keine brauchbare Abgrenzung zwischen Anforderungen und Entwurfsentscheidungen. Die gleiche Sache kann je nach Kontext beides sein.
- **Probleme** (beschrieben als Anforderungen) und **Lösungen** (das Ergebnis von Entwurfsentscheidungen) sind **hierarchisch miteinander verzahnt**!
- **➔ WAS vs. WIE ist stufenabhängig: Eine Entwurfsentscheidung auf Stufe n wird zur Aufgabe (=Anforderung) auf Stufe n+1**

Standards

- **Standards für das Requirements Management**

- Ein sehr bekannter Standard zum Requirements Management ist der **IEEE 830** (Recommended Practice for Software Requirements Specifications). Er ist ein konkreter praxisnaher Standard für die Beschreibung und die Definition von Softwareanforderungen. Es werden Strukturen vorgeben für den Aufbau der Dokumentation, den Aufbau der einzelnen Anforderungen und sogar zur Formulierung der Anforderungen.
- Eine gute Ergänzung hierzu ist der Standard **IEEE 1362** (Guide for Information Technology – System Definition, Definition - Concept of Operations Document), der letztendlich ein Standard für Anforderungsdokumente (Lastenhefte) darstellt.
- Eine weitere sinnvolle Ergänzung zu den beiden genannten Standards sind die Spezifikationen aus **VDI 2519 – Blatt 1** (Vorgehensweise bei der Erstellung von Lasten-/Pflichtenheften). Hier wird definiert was Lasten- und Pflichtenhefte sind, was sie enthalten sollten und wie sie erstellt werden sollten.

Kundenorientierung im RE



Anforderungsarten

- Visionen und Ziele
- Rahmenbedingungen
  - z.B. Juristische Vorgaben
- Kontext und Überblick
  - Systemumgebung
- **Nichtfunktionale Anforderungen**
  - Qualitätsmerkmale
- **Funktionale Anforderungen**
  - Funktionen

Funktionale und nichtfunktionale Anforderungen

- **Funktionale Anforderungen**
  - Beschreibung der Dienste des Systems, z.B. wie es auf bestimmte Eingaben reagiert oder sich in bestimmten Situationen verhält
  - z.B. was passiert wenn ein bestimmter Knopf gedrückt wird

- **Nicht-funktionale Anforderungen**
  - Randbedingungen für die Dienste, z.B. zeitliche Einschränkungen, Entwicklungseinschränkungen, usw.
  - z.B. Lebensdauer oder Leistung

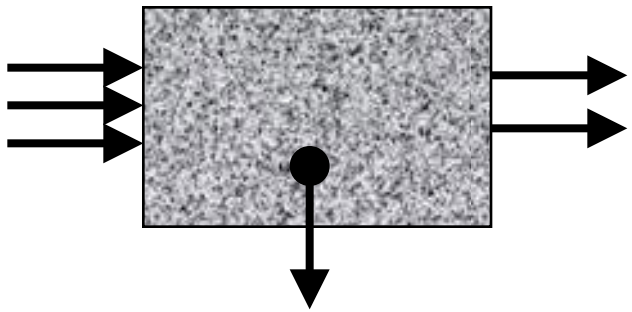
- **Domänen-Anforderungen**
  - allgemeine Anforderungen für alle Systeme einer bestimmten Anwendungsdomäne (Medizintechnik, Schienenverkehr...)
  - z.B. anwendbare Standards

Funktionale Anforderungen

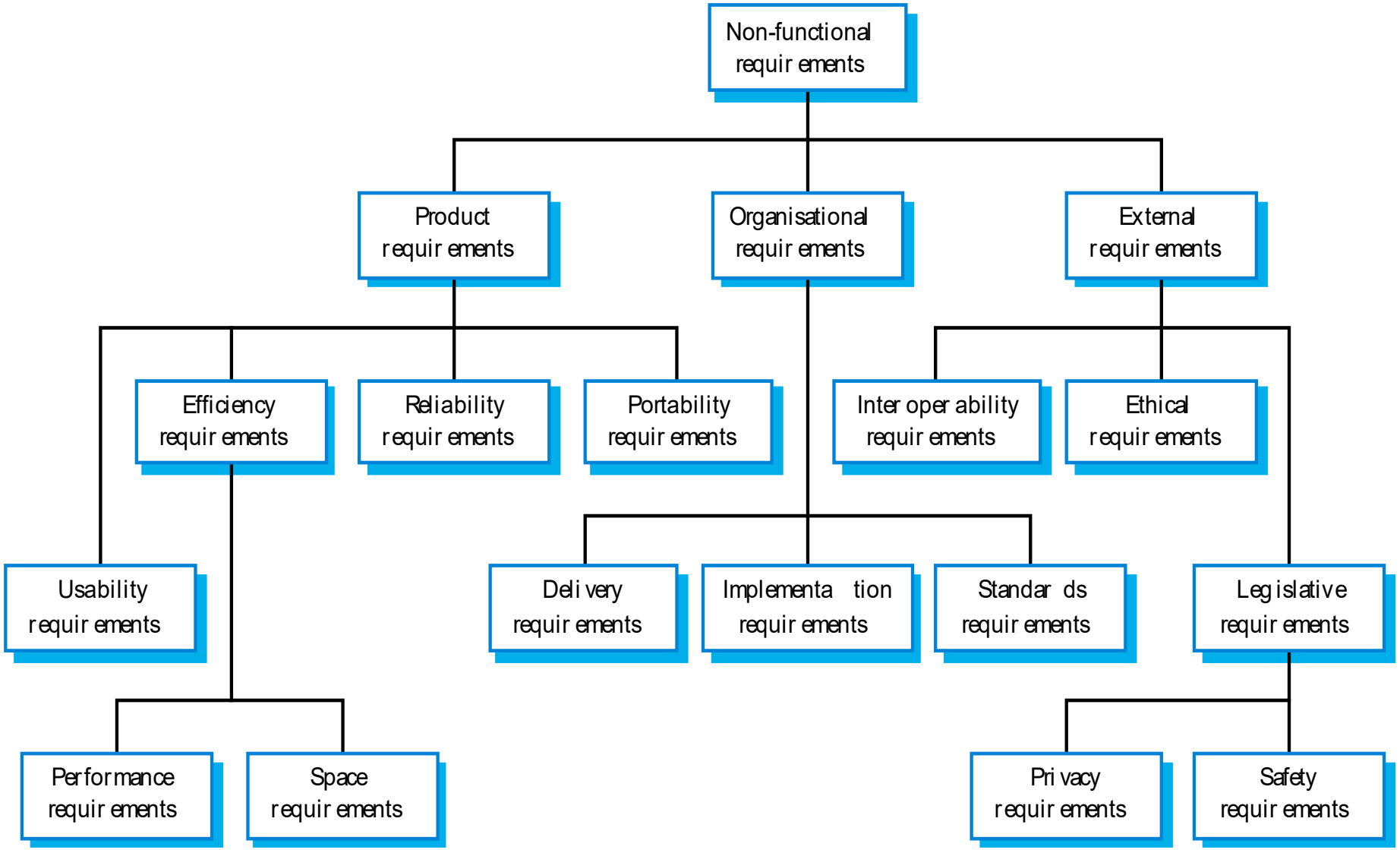
- Funktionalität oder Dienste des Systems
  - Funktionale Nutzeranforderungen: Abstrakte Außensicht
  - Funktionale Systemanforderungen: Abstrakte Innensicht

- Formulierung hängt ab von der zu erwartenden Nutzung und dem Einsatzbereich des Systems

- BLACK BOX VIEW!



Nichtfunktionale Anforderungen



Struktur einer Anforderung

Typischerweise besteht eine Anforderung aus folgenden Attributen:

- **Identifikator** *(Requirement Number):* Identifiziert die Anforderung eindeutig.
- **Beschreibung** *(Description)*
- **Problembeschreibung** *(Rationale):* Beschreibt das die Anforderung verursachende Problem.
- **Priorität** (*Priority*): Ein numerischer Wert, der die Priorität dieser Anwendung definiert und dann wichtig wird, wenn nicht alle Anforderungen erfüllt werden können.
- Quelle *(Originator):* Identifiziert die anfordernde Person oder ein Dokument, aus dem sich die Anforderung ergibt, beispielsweise eine Rechtsvorschrift.
- Abnahmekriterium *(Fit Criterion):* Beschreibt eine messbare Bedingung, mit der später geprüft wird, ob die Anforderung erfüllt wurde.
- Konflikte (*Conflicts*): Hier können Anforderungen aufgeführt werden, die dieser Anforderung widersprechen, sodass zwischen ihnen abgewägt werden muss.

Informationen zum Lebenszyklus:

- ***Status***: Identifiziert den aktuellen Zustand der Anforderung, beispielsweise ob sie vom Auftragnehmer bereits akzeptiert wurde.
- Offene Punkte: Bietet Platz für noch zu klärende Sachverhalte.
- ***History***: Versionsgeschichte der Anforderung: Wann wurde sie von wem erstmals formuliert, wann von wem geändert usw.

I

Anforderungsbeschreibung

- Verbal
  - Fließtext, strukturierte Texte
- Non-Verbal
  - Grafische Notationen
  - Formale Sprachen

Natürlichsprachliche Anforderungen

- Anforderungen werden meist in natürlicher Sprache beschrieben

- Vorteil:
  - Einfach, flexibel, universell

- Nachteil:
  - Gefahr der Mehrdeutigkeit, Vermischung etc.
    - ➔ Erstellung eines Glossars
    - ➔ Verwendung von Schablonen

DB-Design

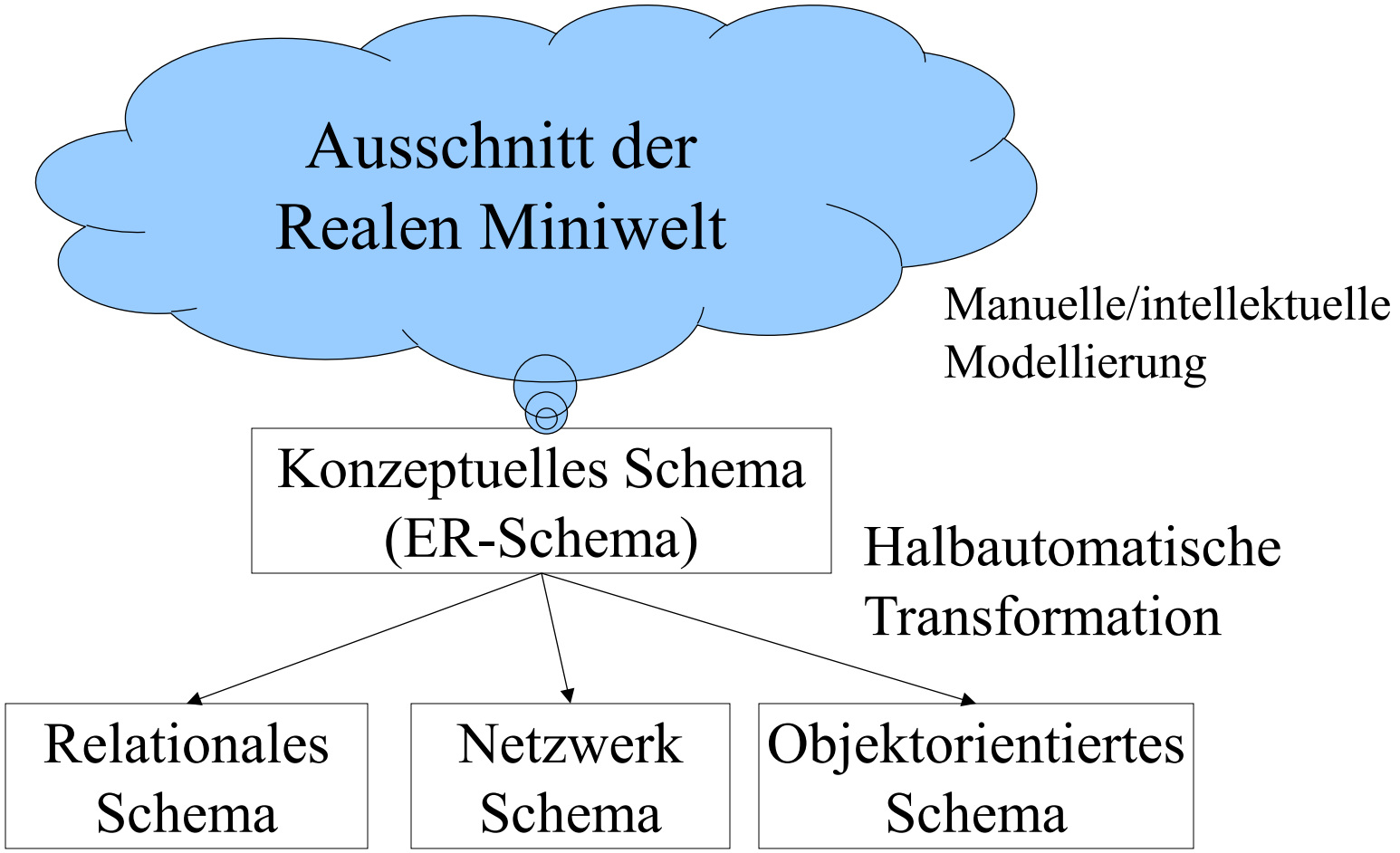
Teil 1: ER(M)

Entity-Relationship-Modell

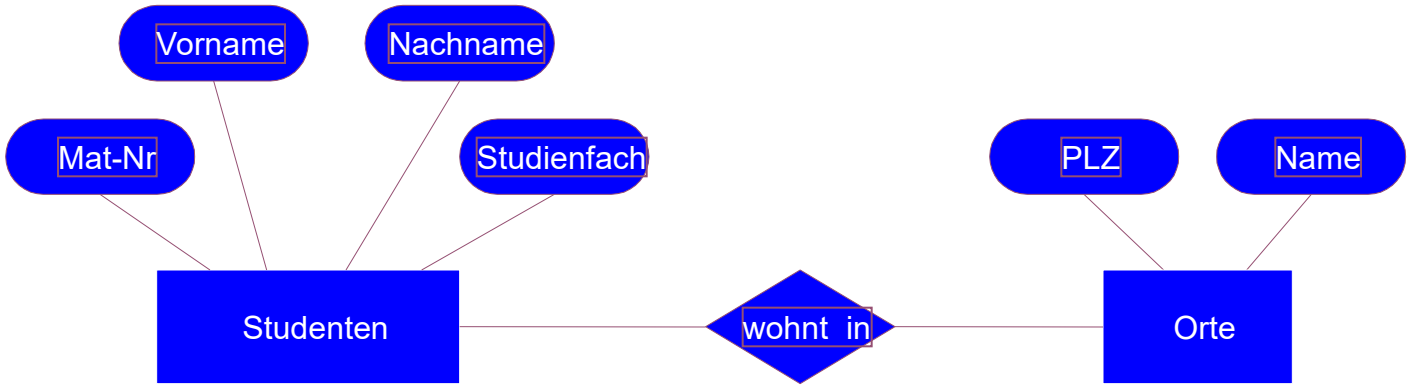
- entity:
  Gegenstand des Denkens und der Anschauung

- relationship:
  Beziehung zwischen den entities

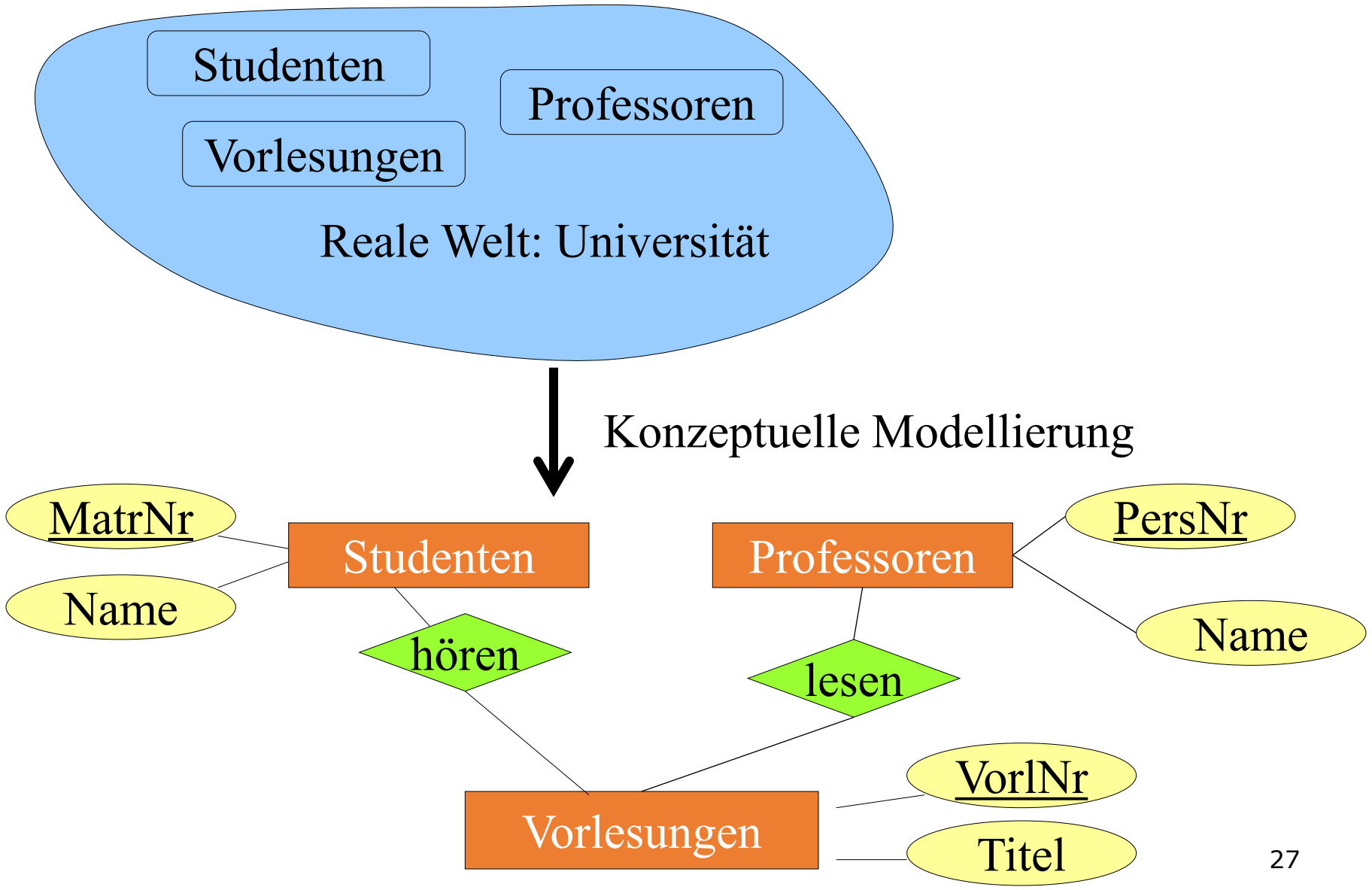
Datenmodellierung



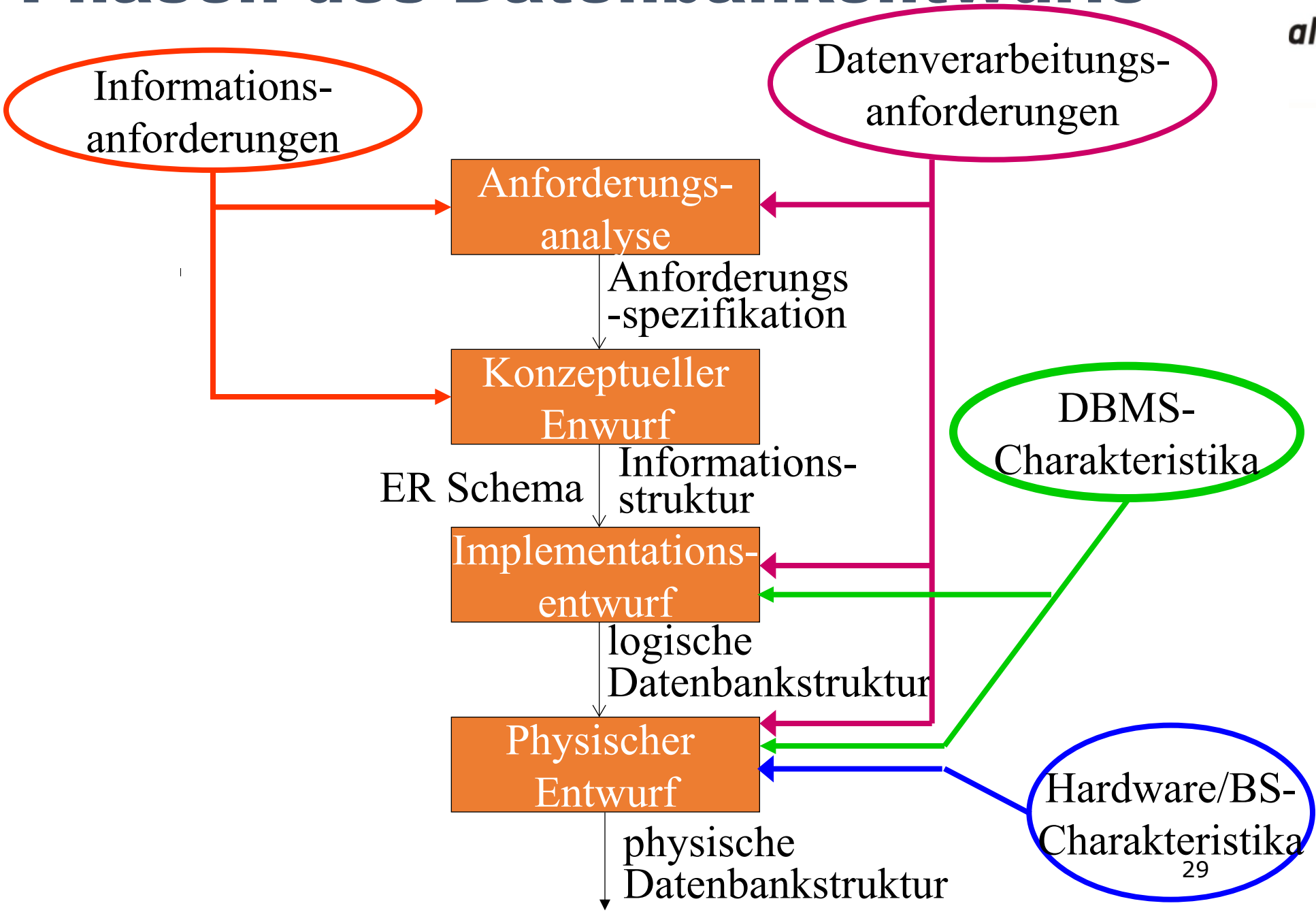
ER-Diagramm


Vorname
Nachname
Mat-Nr
Studienfach
PLZ
Name
Studenten
wohnt in
Orte

Modellierung einer kleinen Beispielanwendung



Phasen des Datenbankentwurfs



Objektbeschreibung

- Uni-Angestellte
  - Anzahl: 1000
  - Attribute
    - ❖ PersonalNummer
      - Typ: char
      - Länge: 9
      - Wertebereich: 0...999.999.99
      - 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*

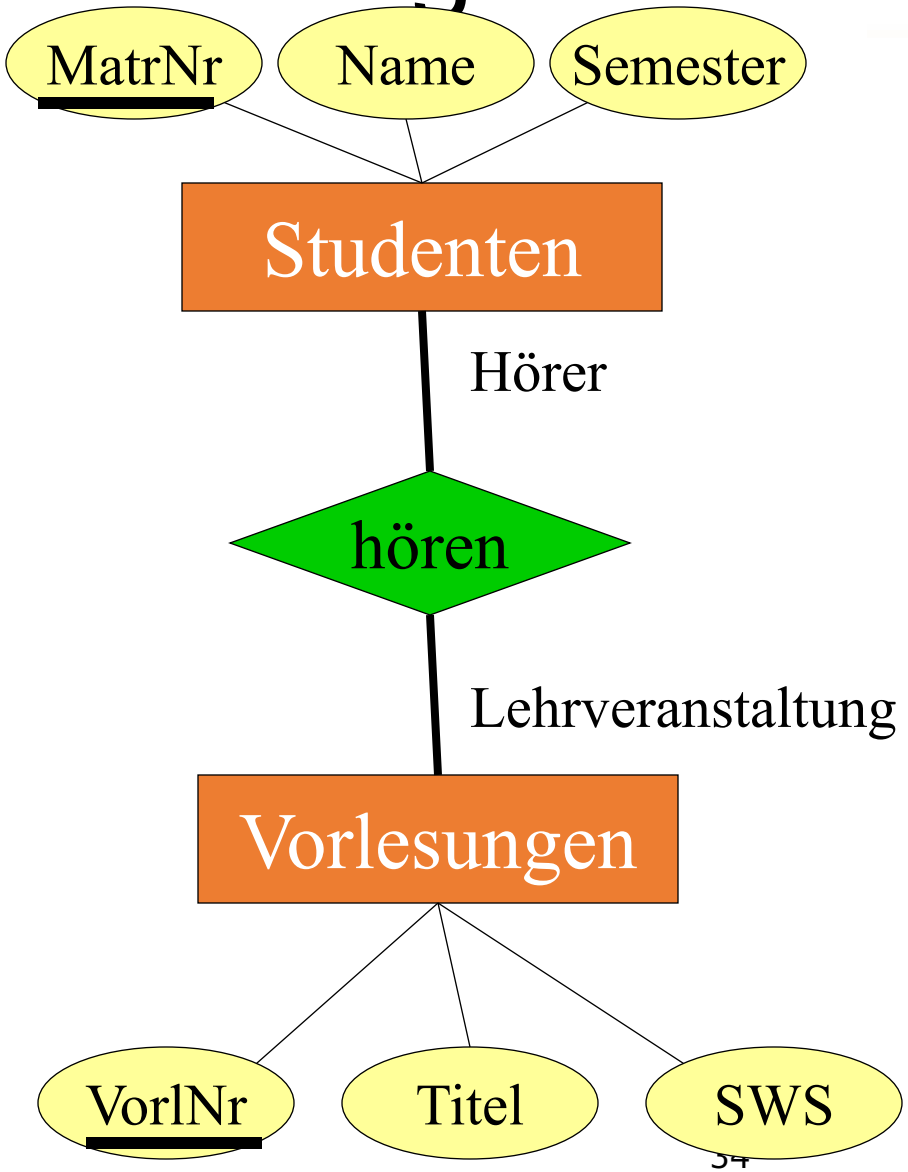
- Beteiligte Objekte:
  - Professor als Prüfer
  - Student als Prüfling
  - Vorlesung als Prüfungsstoff
- Attribute der Beziehung:
  - Datum
  - Uhrzeit
  - Note
- Anzahl: 100 000 pro Jahr

Prozeßbeschreibungen

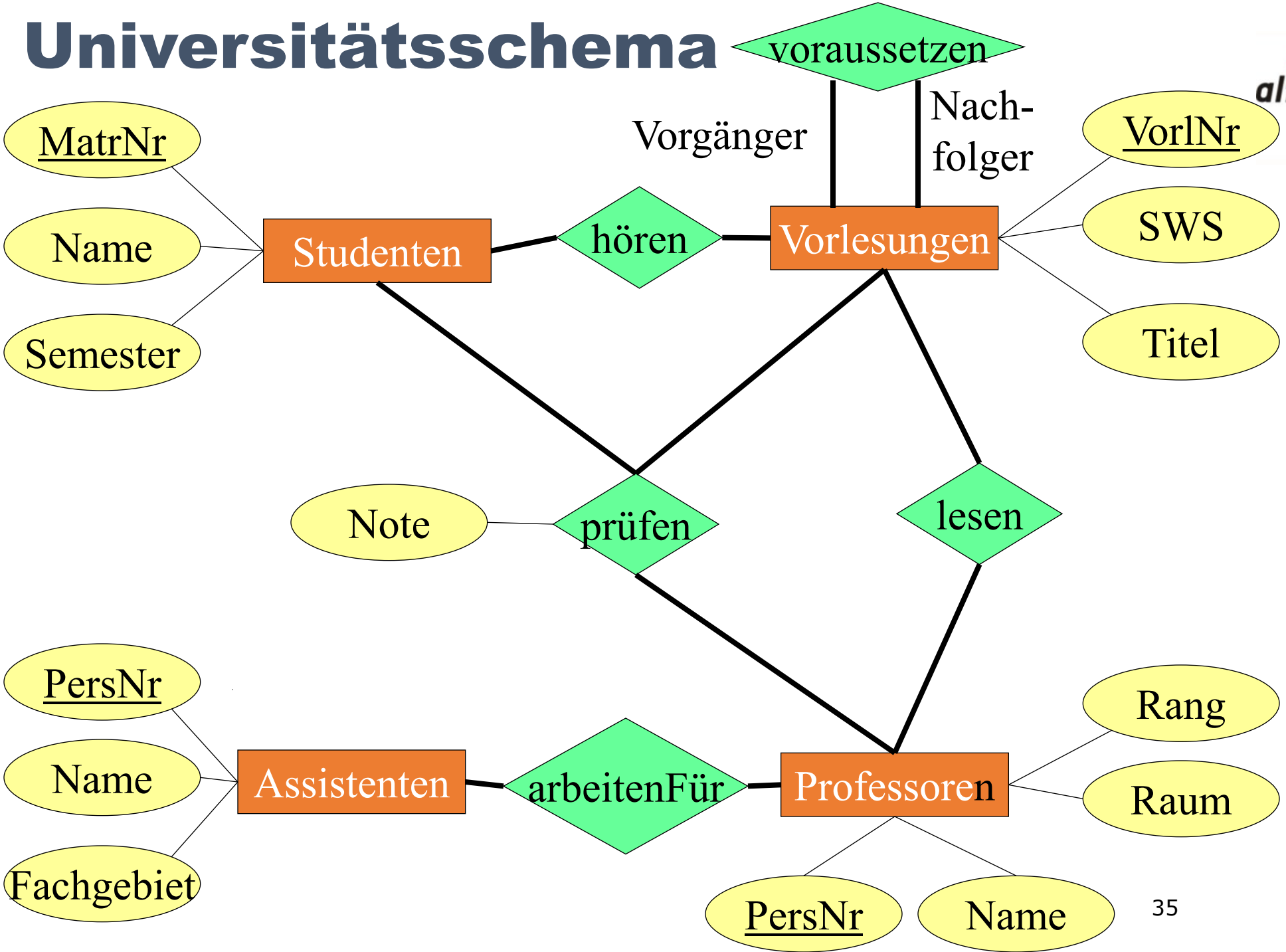
- Prozeßbeschreibung: *Zeugnisausstellung*
  - Häufigkeit: halbjährlich
  - benötigte Daten
    * Prüfungen
    * Studienordnungen
    * Studenteninformation
    * ...
  - Priorität: hoch
  - Zu verarbeitende Datenmenge
    * 500 Studenten
    * 3000 Prüfungen
    * 10 Studienordnungen

Entity/Relationship-Modellierung

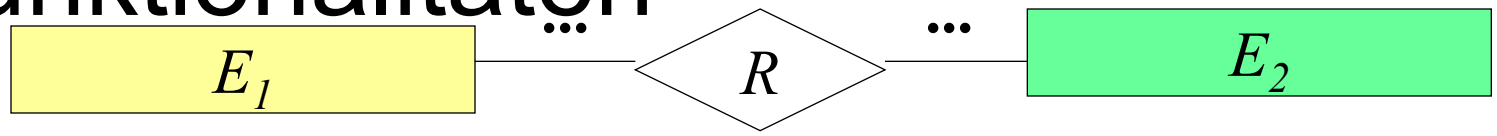
- Entity (Gegenstandstyp)
- Relationship (Beziehungstyp)
- Attribut (Eigenschaft)
- Schlüssel (Identifikation)
- Rolle



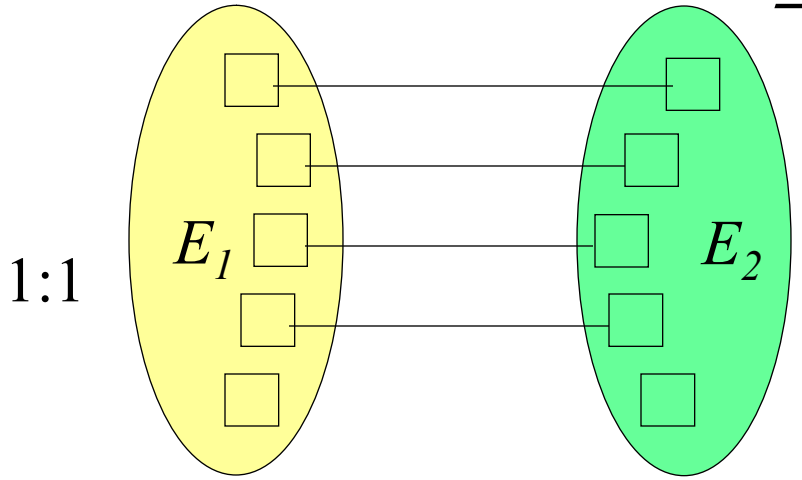
Universitätsschema

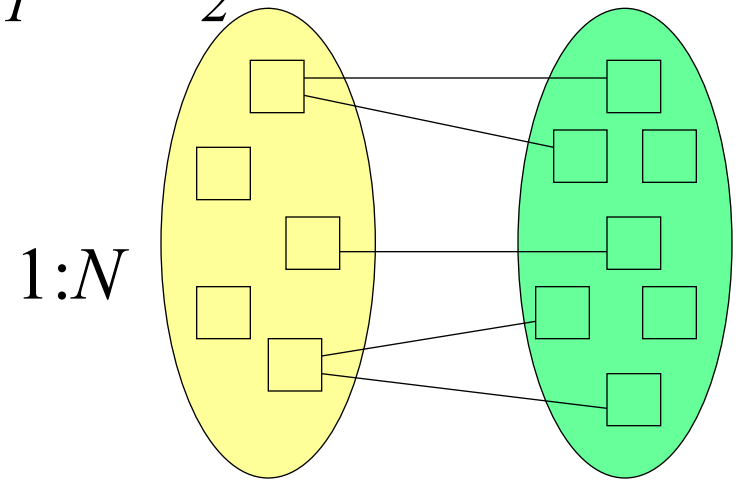


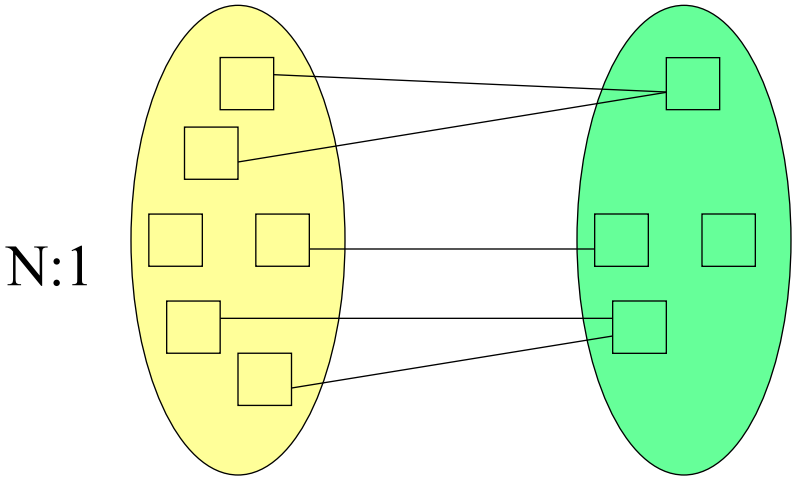
Funktionalitäten

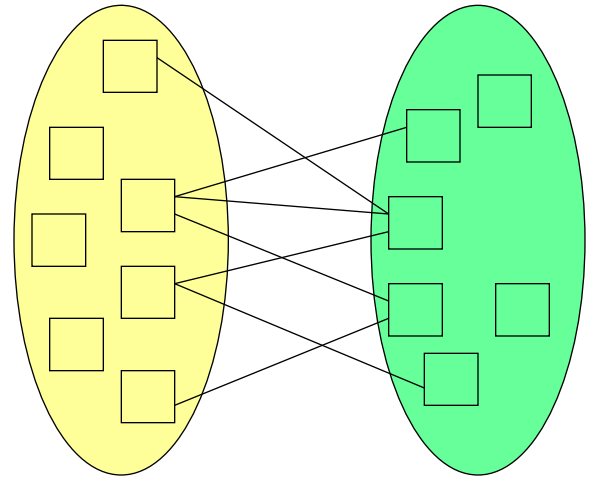


$$R \subseteq E_1 \times E_2$$

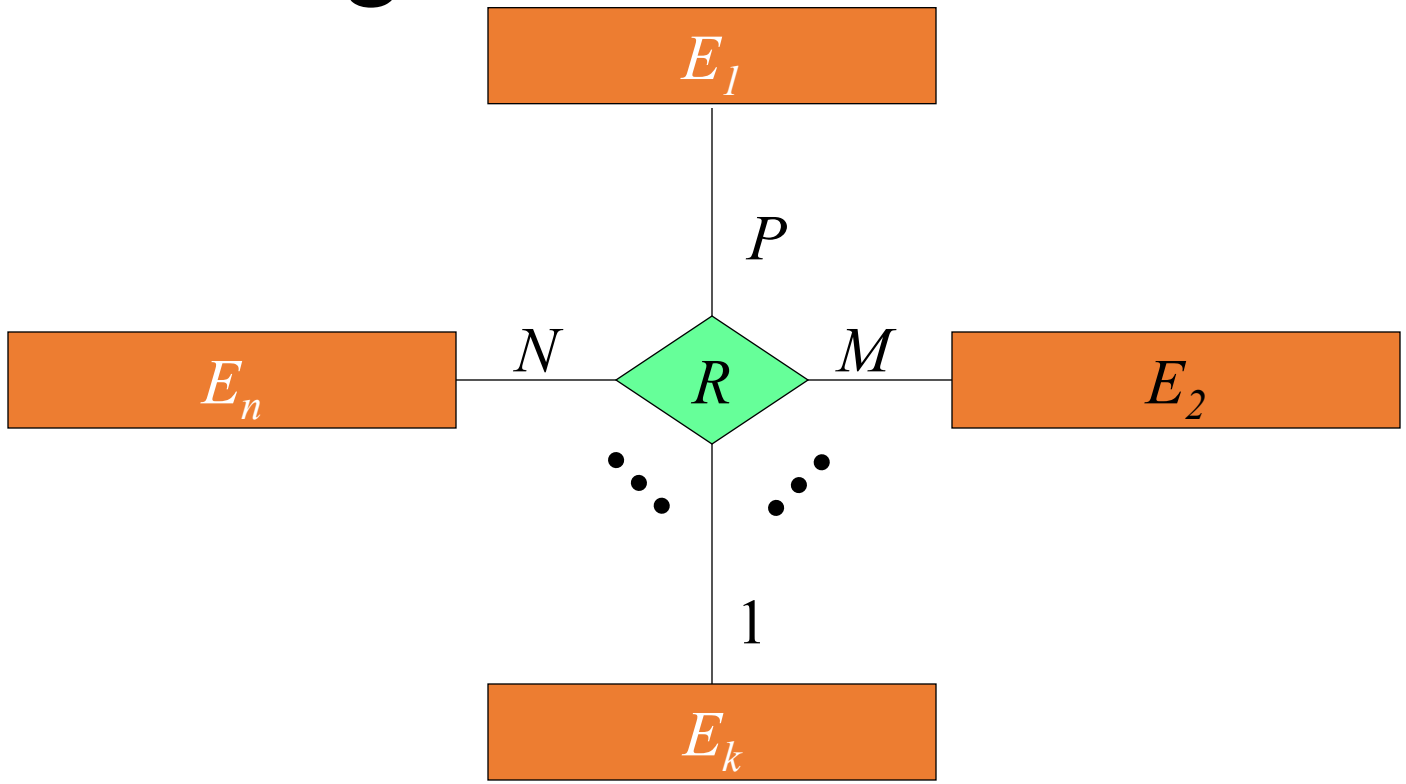






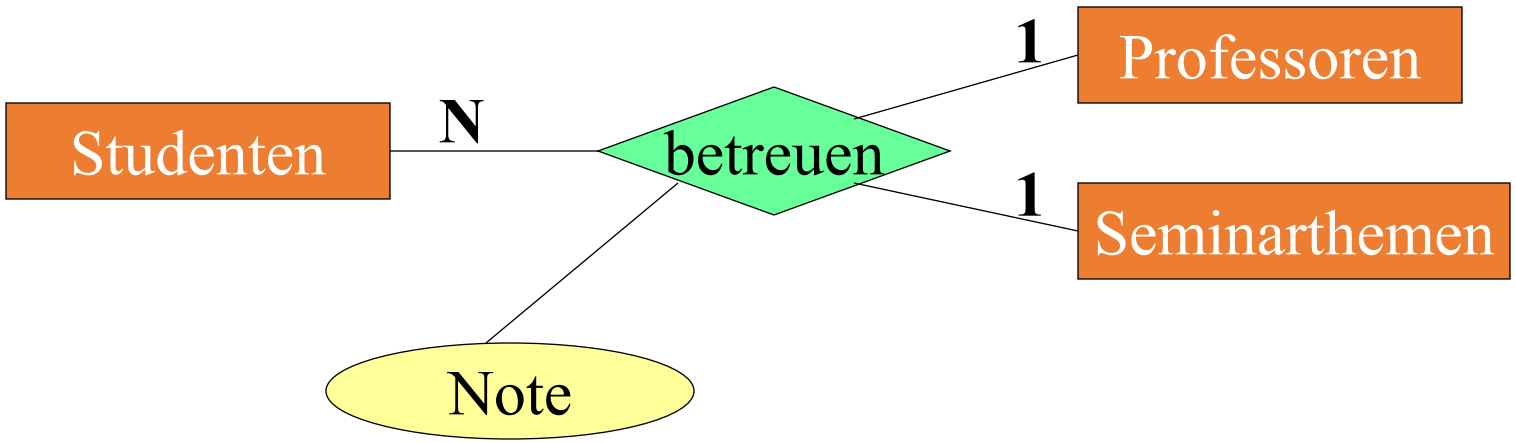


Funktionalitäten bei n-stelligen Beziehungen



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

Beispiel-Beziehung: *betreuen*



betreuen : Professoren x Studenten → Seminarthemen

betreuen : Seminarthemen x Studenten → Professoren

Dadurch erzwungene Konsistenzbedingungen

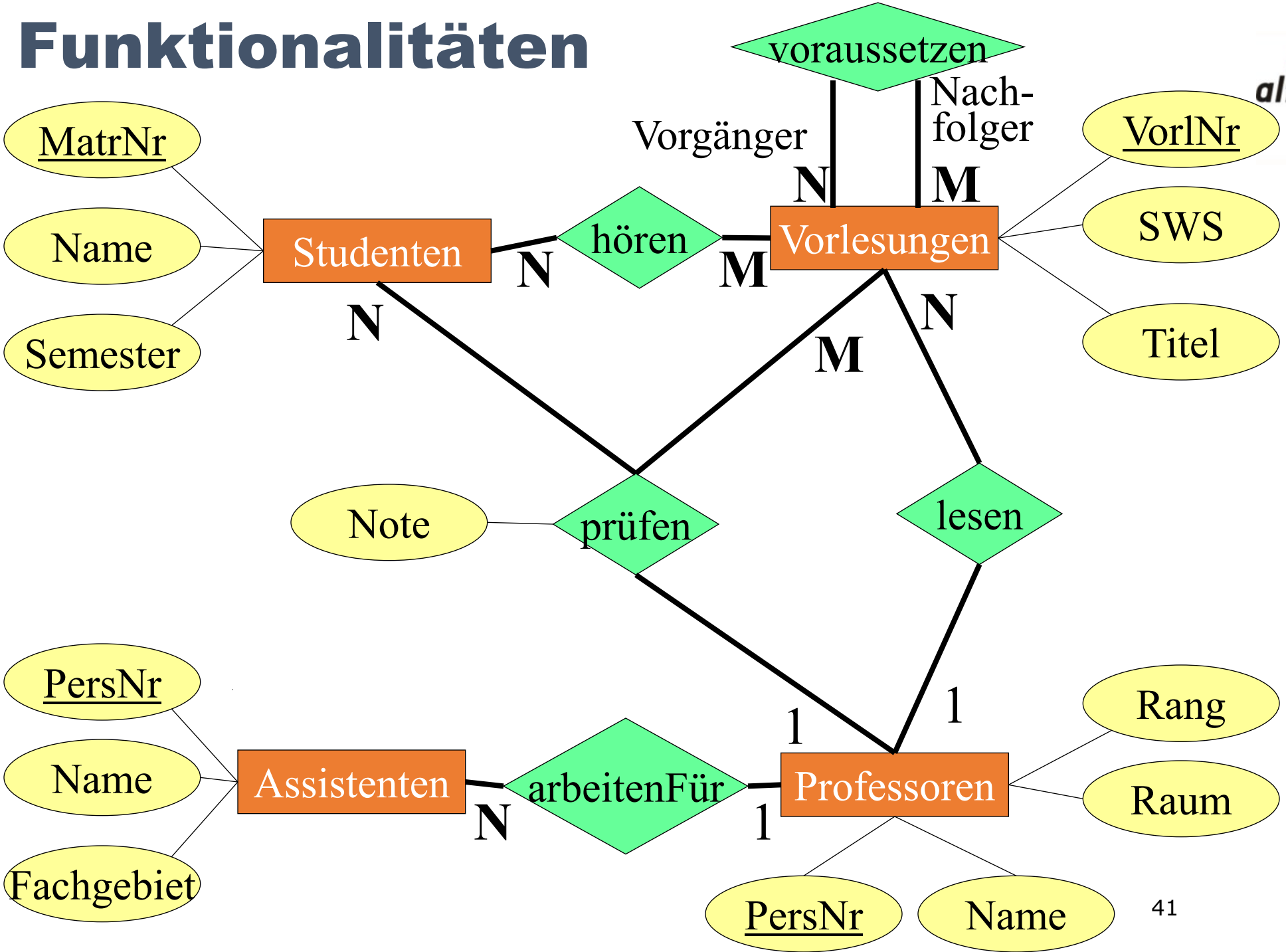
1. Studenten dürfen bei demselben Professor bzw. derselben Professorin nur ein Seminarthema "ableisten" (damit ein breites Spektrum abgedeckt wird).

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

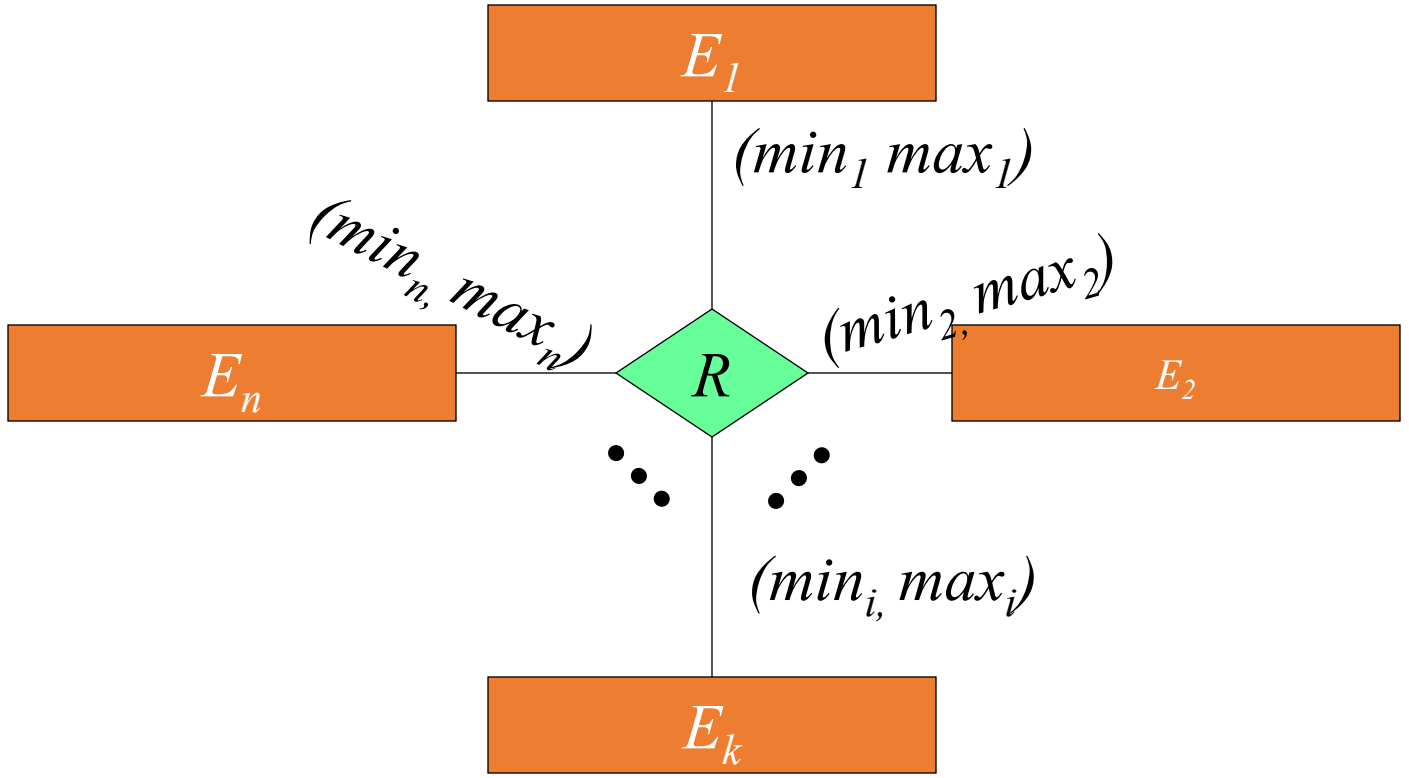
Es sind aber folgende Datenbankzustände nach wie vor möglich:

- Professoren können dasselbe Seminarthema „wiederverwenden“ – also dasselbe Thema auch mehreren Studenten erteilen.
- Ein Thema kann von mehreren Professoren vergeben werden – aber an unterschiedliche Studenten.

Funktionalitäten



(min, max)-Notation

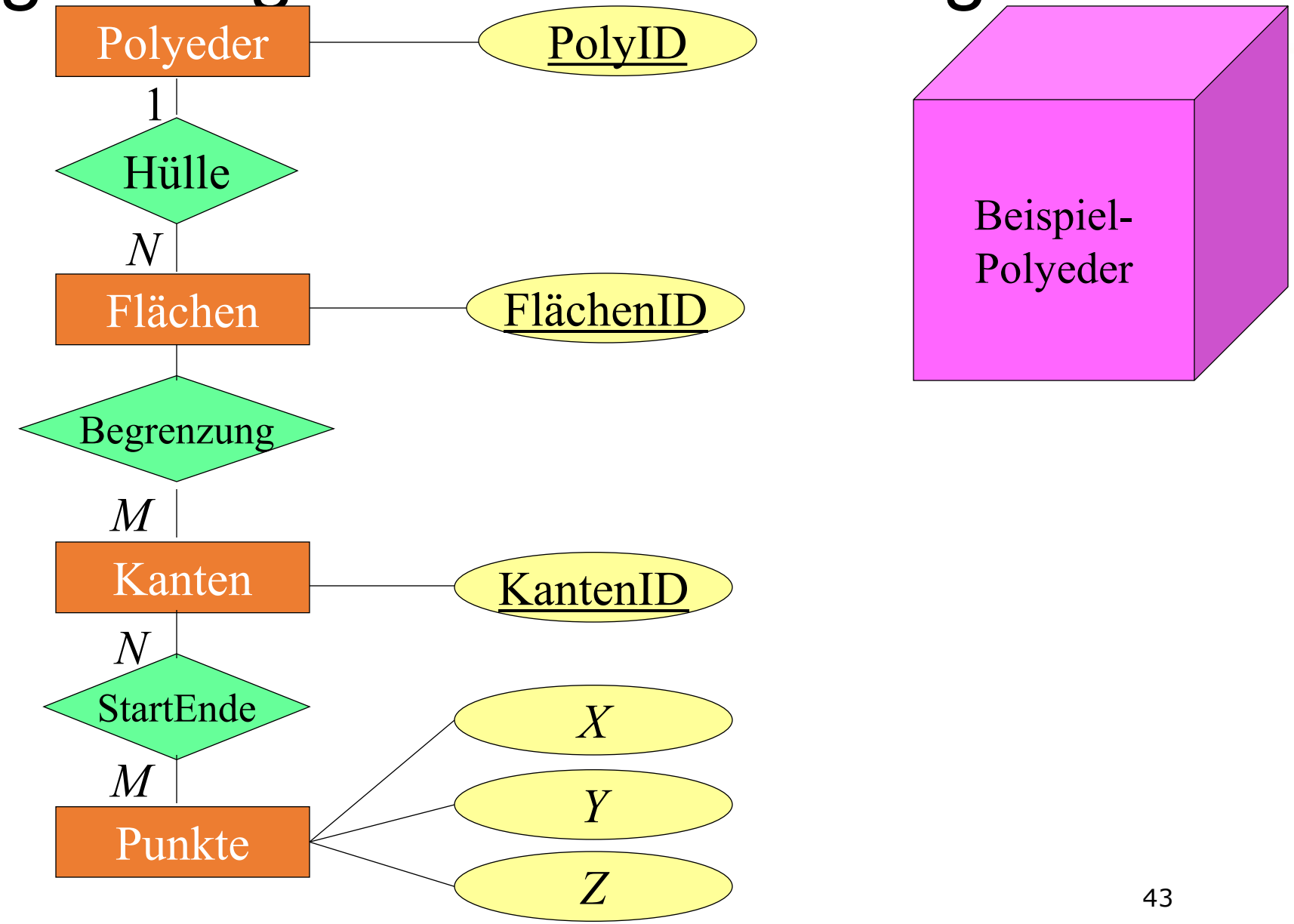


$$R \subseteq E_1 \text{ x } \ldots \text{ x } E_i \text{ x } \ldots \text{ x } E_n$$

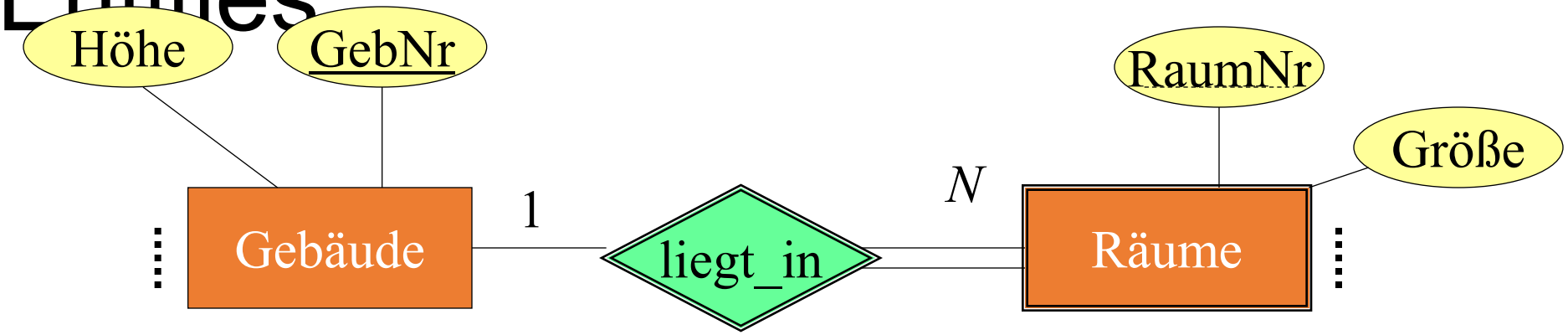
Für jedes $e_i \in E_i$ gibt es

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

Begrenungsflächendarstellung

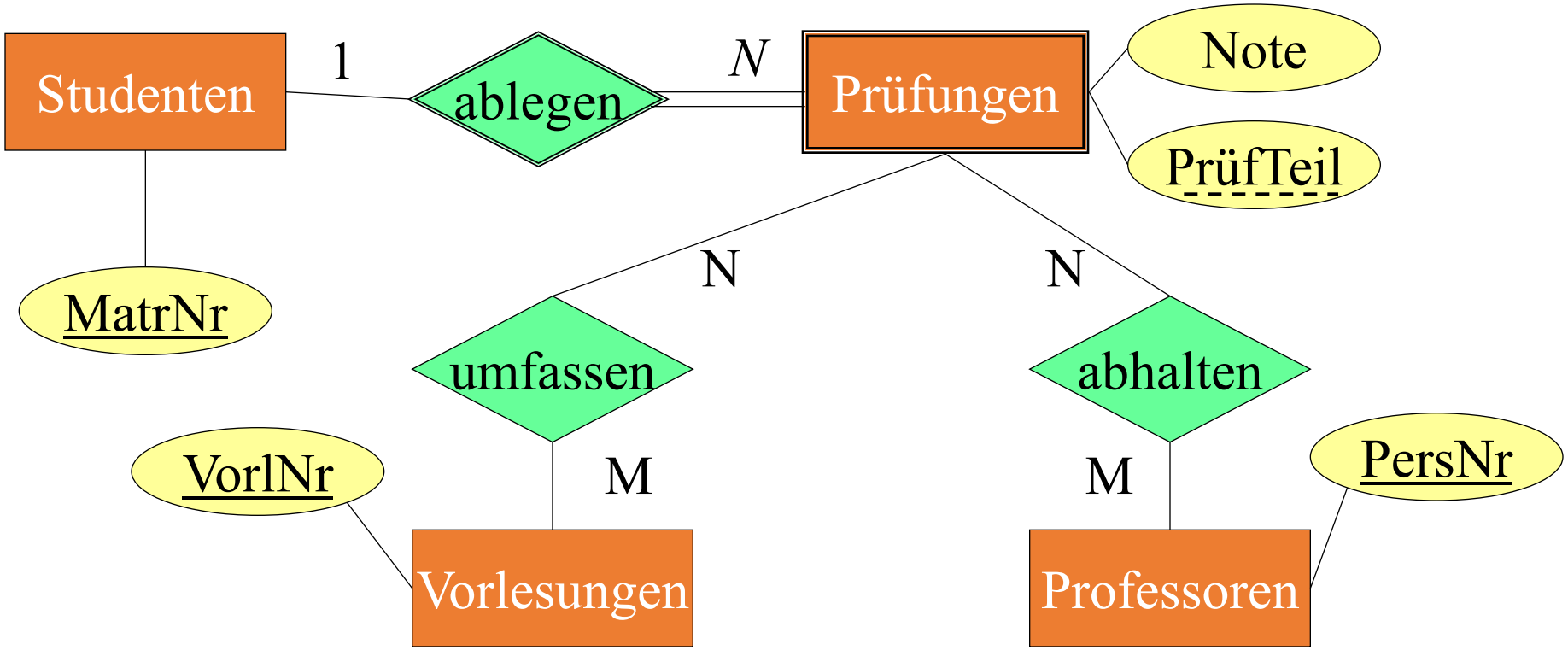


Schwache, existenzabhängige Entities



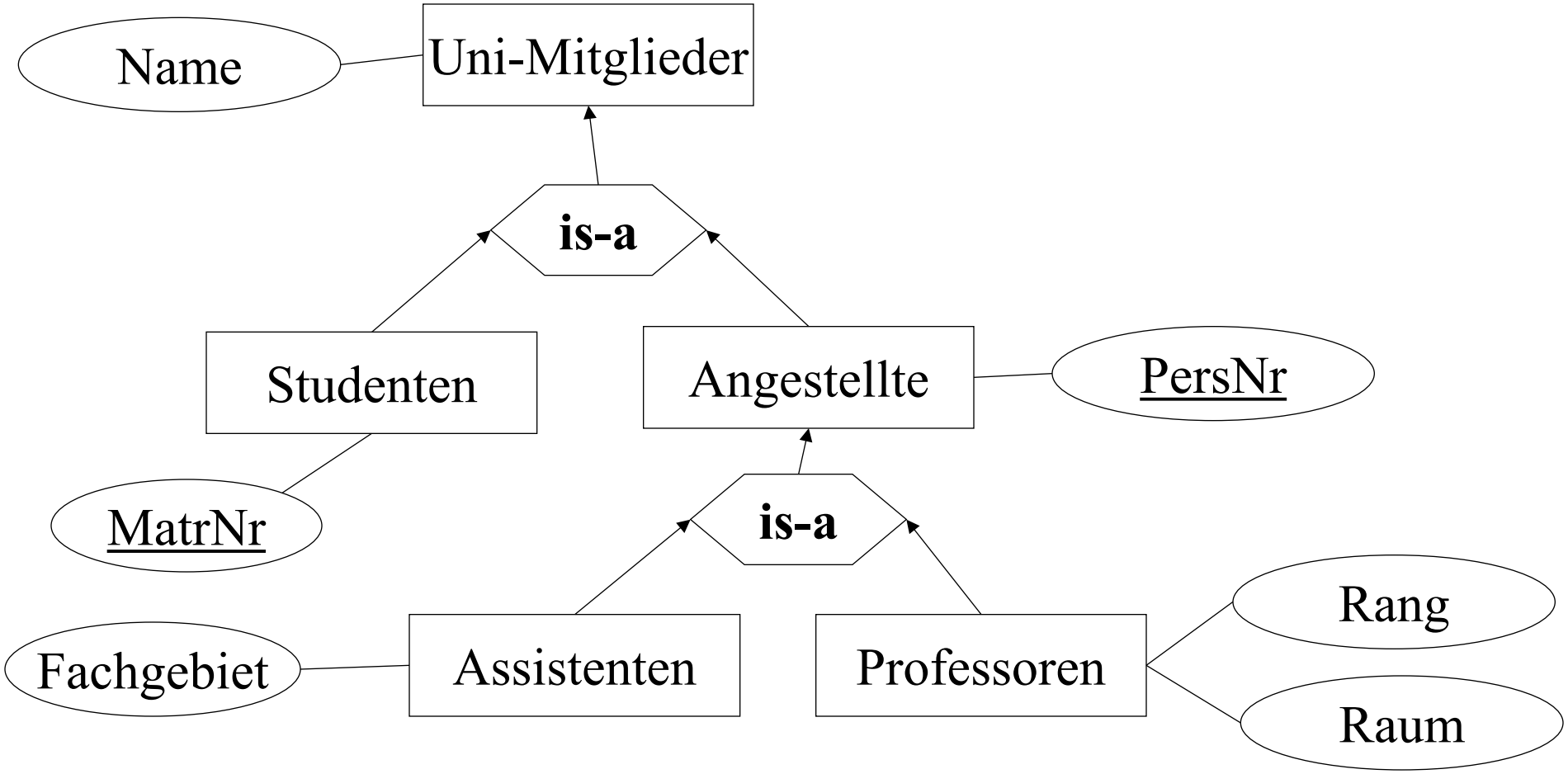
- Beziehung zwischen "starken" und schwachem Typ ist immer 1:*N* (oder 1:1 in seltenen Fällen)
- Warum kann das keine *N:M*-Beziehung sein?
- RaumNr ist nur innerhalb eines Gebäudes eindeutig
- Schlüssel ist: GebNr **und** RaumNr

Prüfungen als schwacher Entitytyp



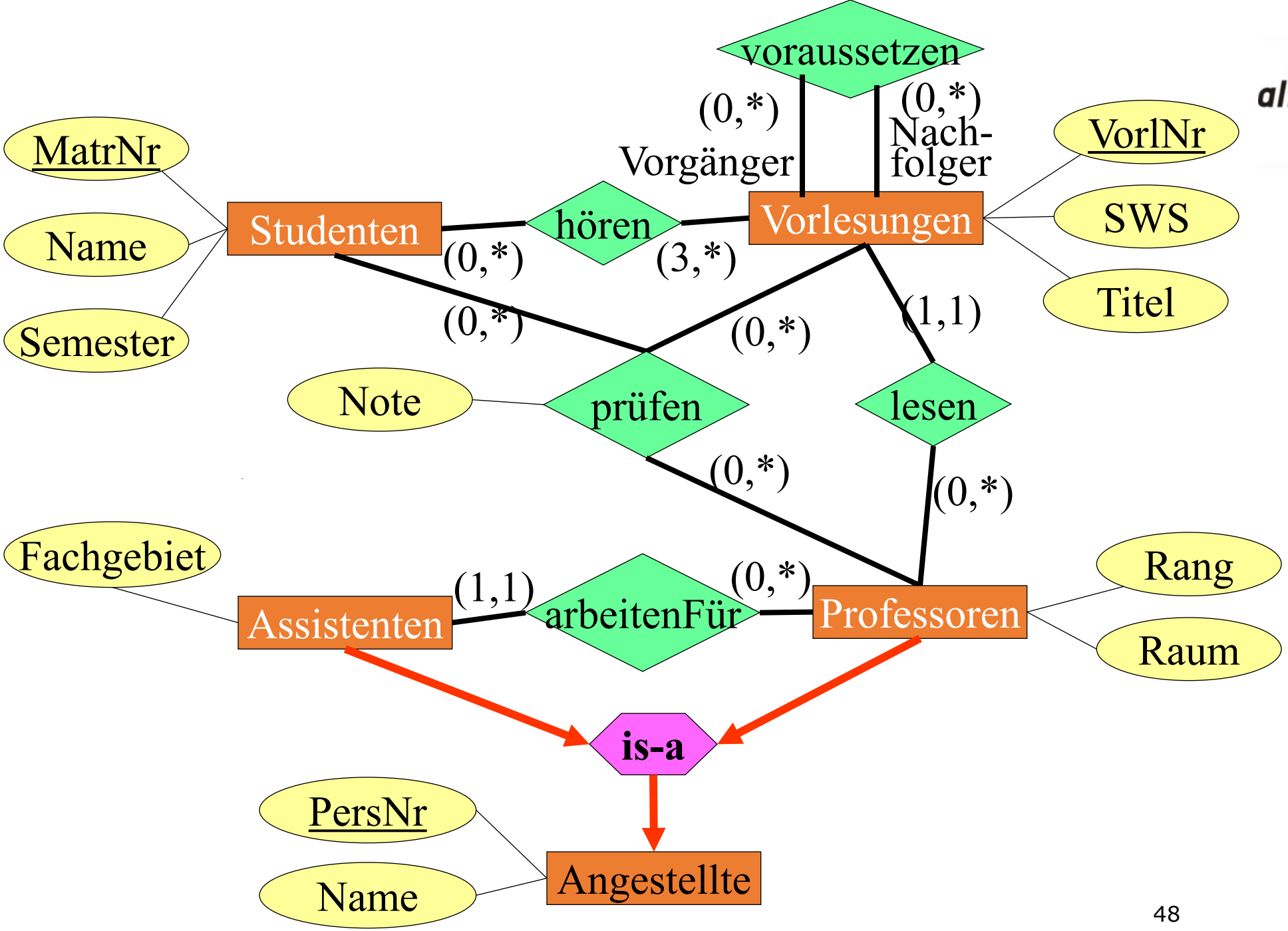
- Mehrere Prüfer in einer Prüfung
- Mehrere Vorlesungen werden in einer Prüfung abgefragt

Generalisierung

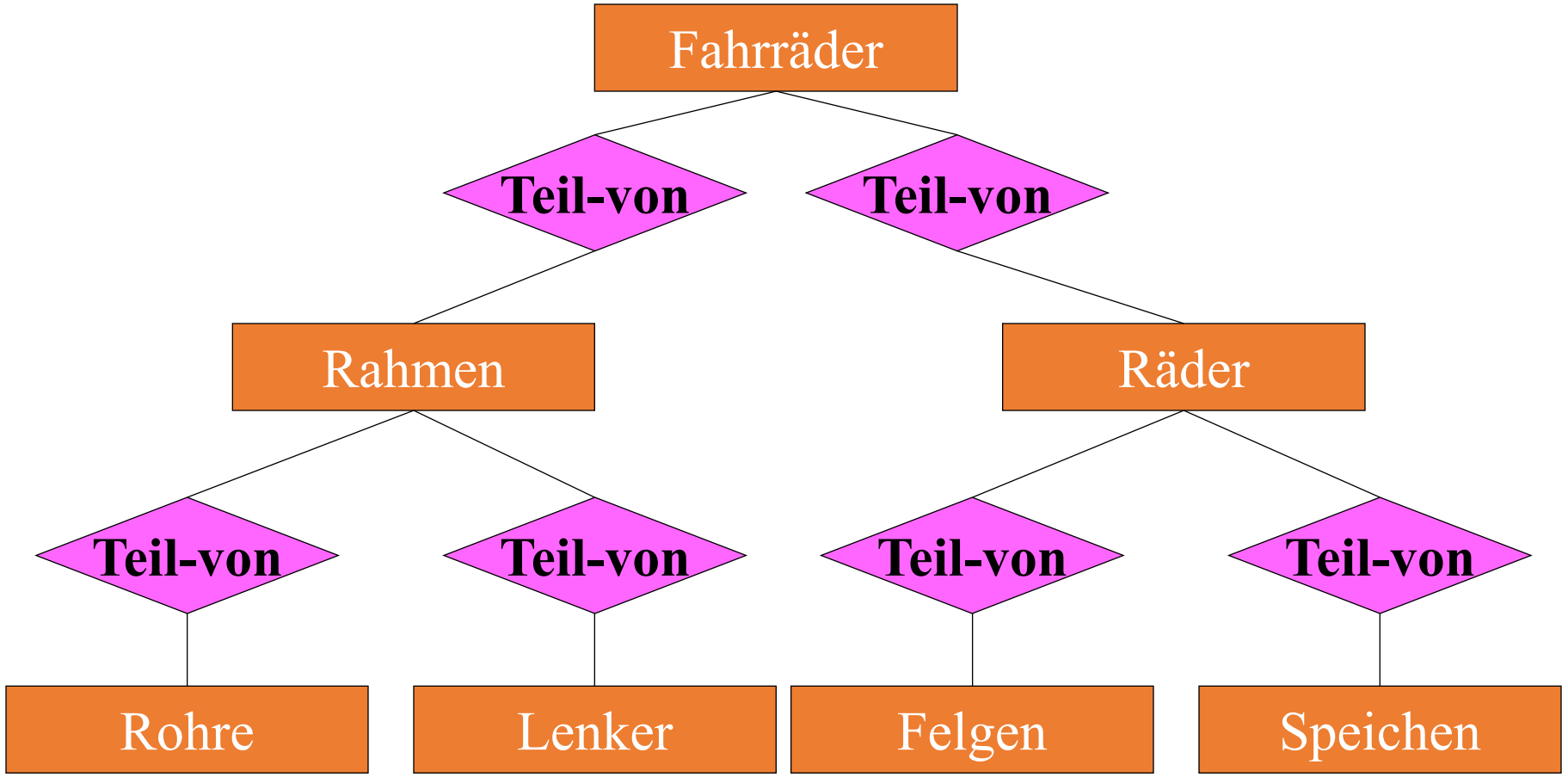


Universitätsschema mit Generalisierung und (min, max)-Markierung

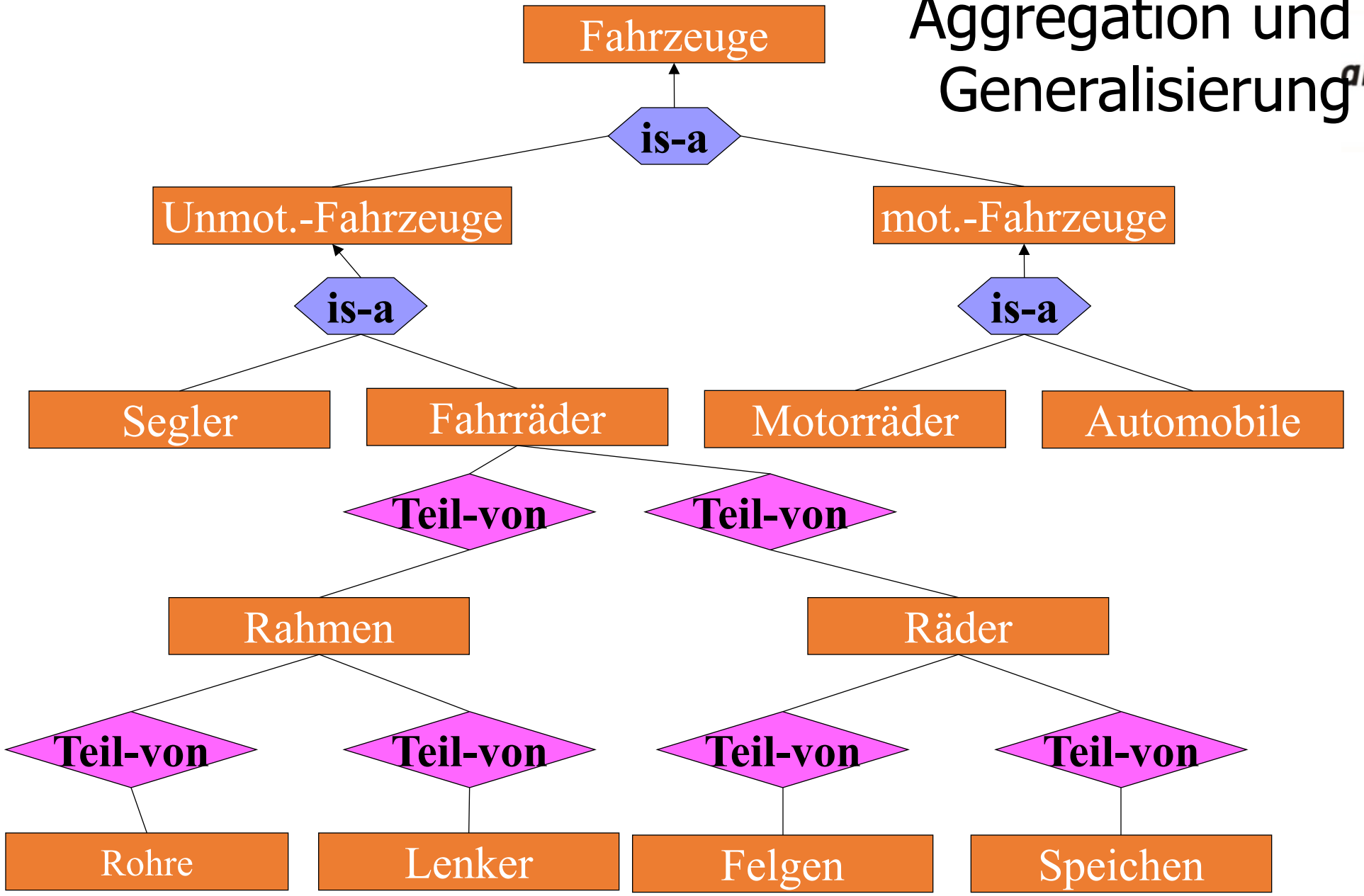
➔ Nächste Seite


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

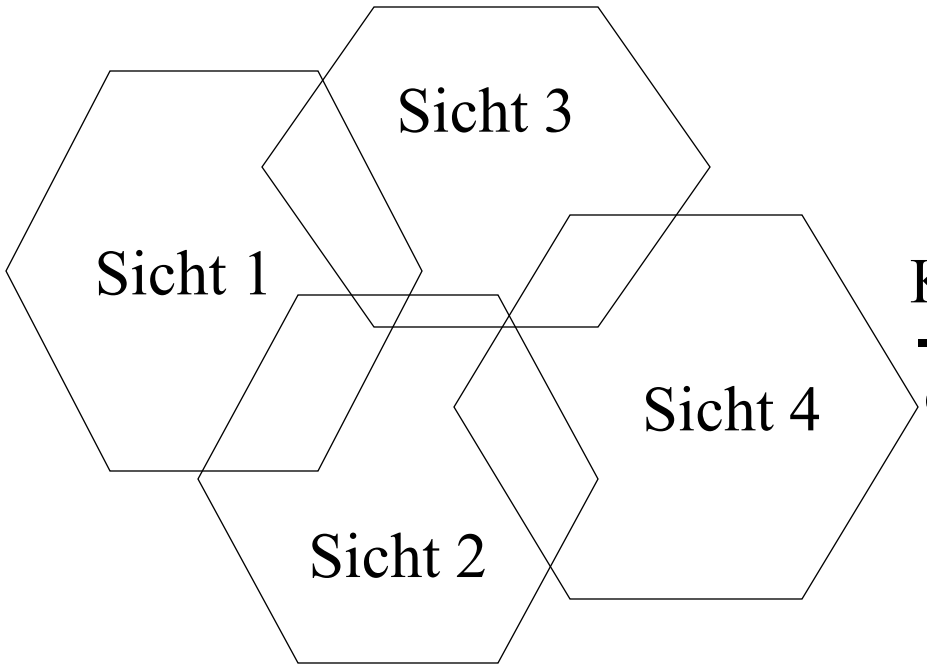
Aggregation



Aggregation und Generalisierung



Konsolidierung von Teilschemata oder Sichtenintegration

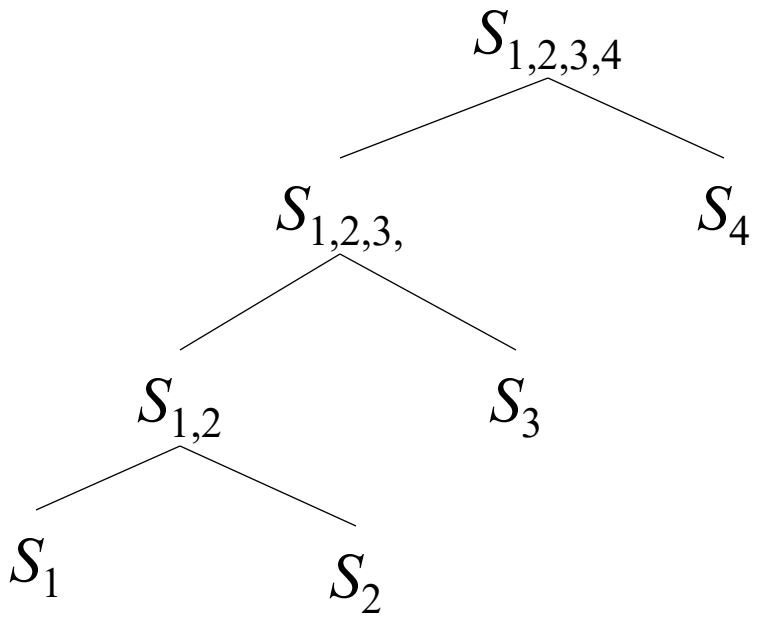


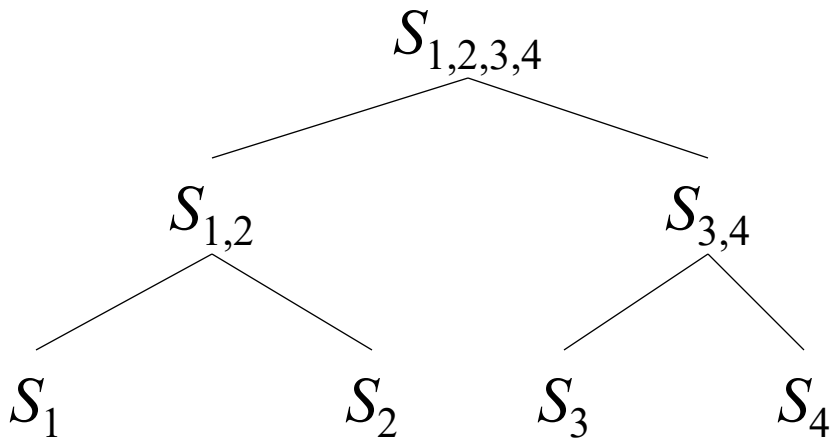
Konsoli-
dierung

Globales Schema

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

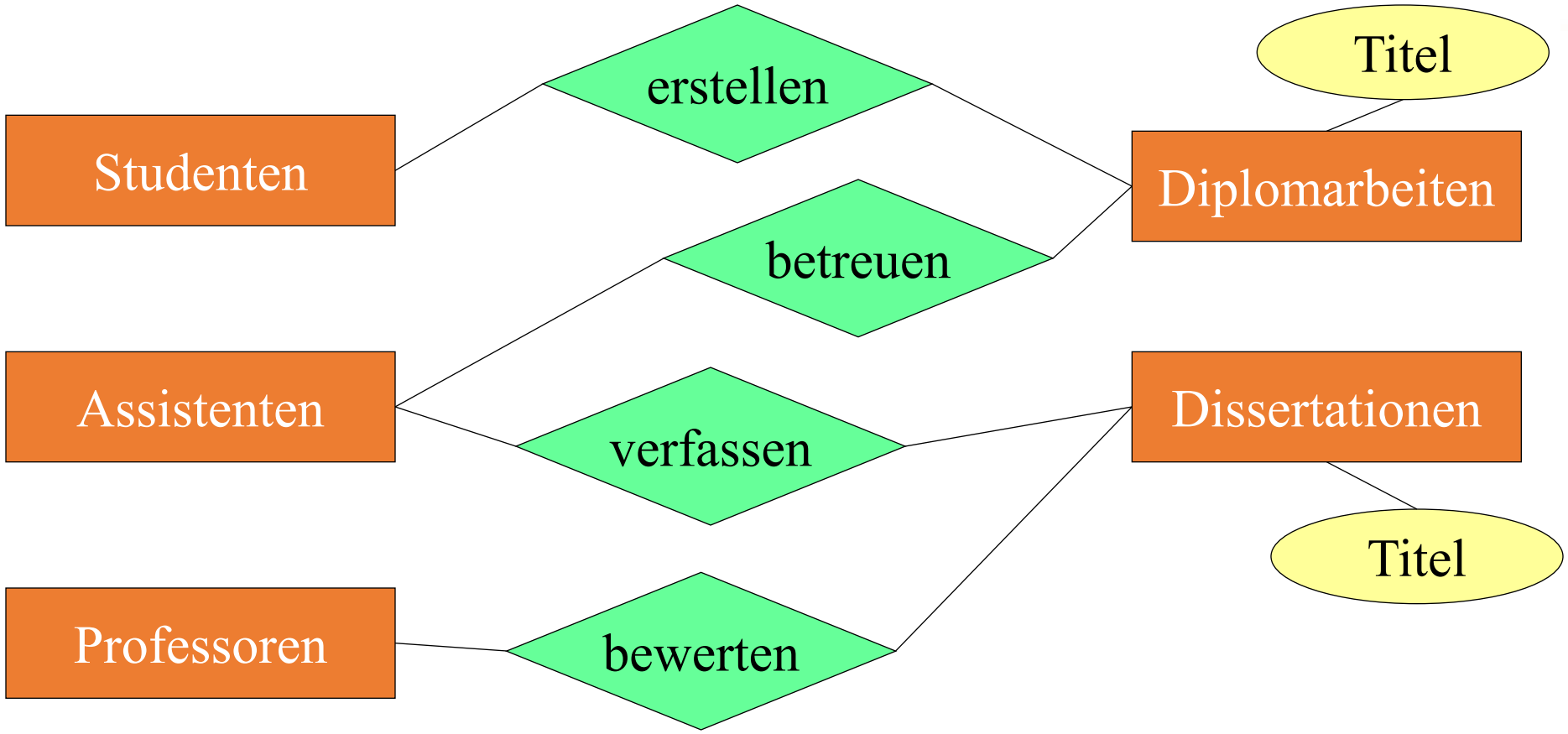
Möglicher Konsolidierungsbaum



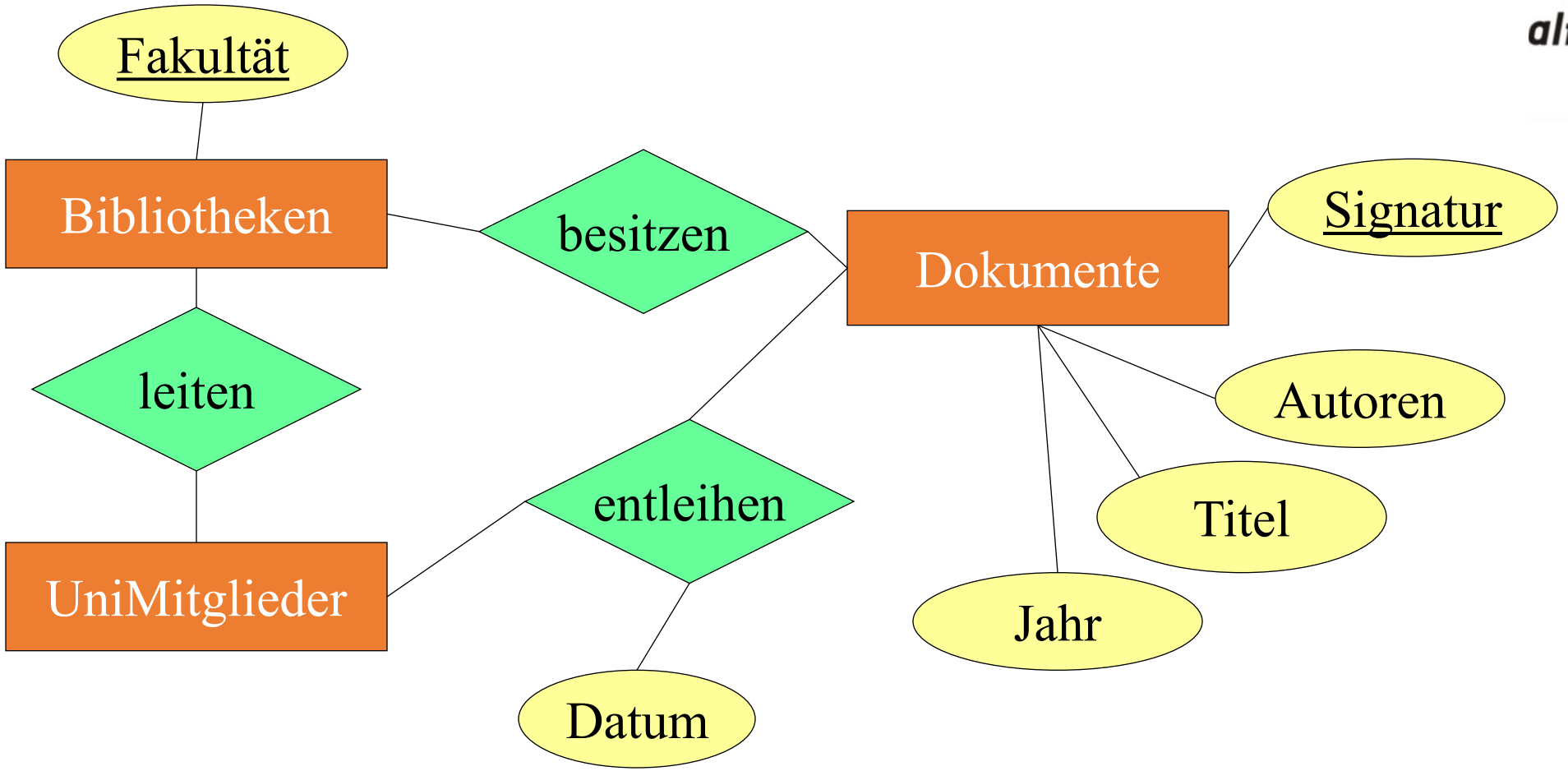


- Mögliche Konsolidierungs-bäume zur Herleitung des globalen Schemas $S_{1,2,3,4}$ aus 4 Teilschemata S_1, S_2, S_3, und S_4
  - Oben ein maximal hoher Konsolidierungsbaum
    - „links-tief“ (left-deep)
  - Unten ein minimal hoher Konsolidierungsbaum
    - Balanciert
- Beide Vorgehensweisen haben Vor-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

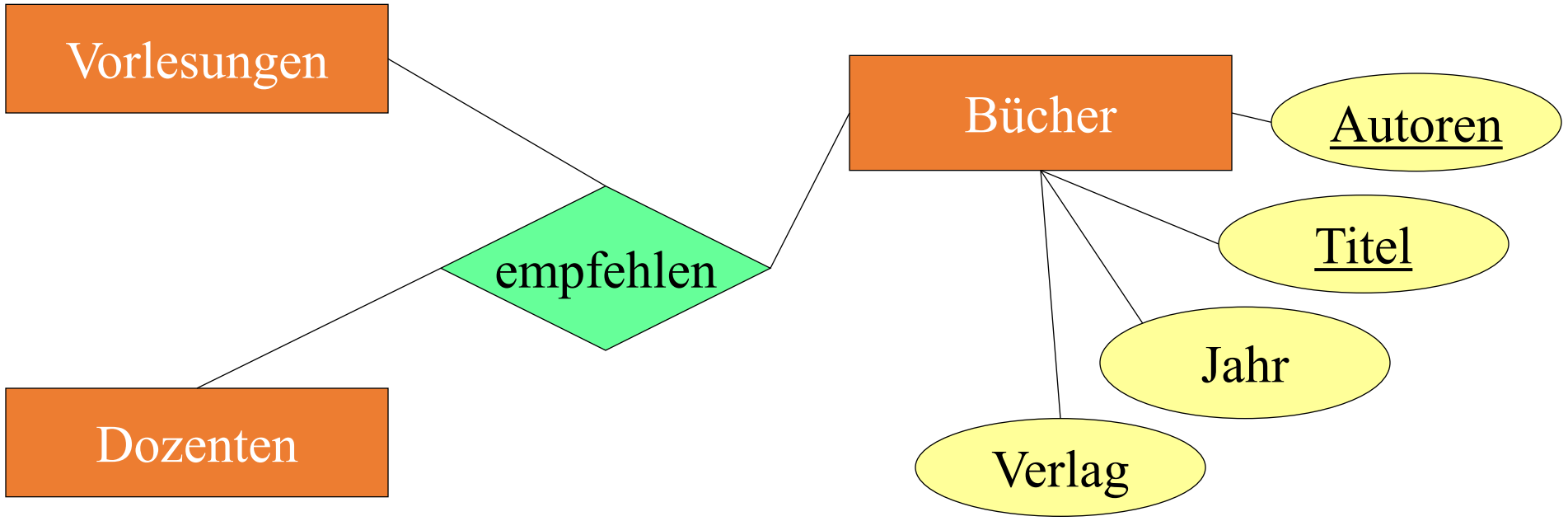
Drei Sichten einer Universitäts-Datenbank



Sicht 1: Erstellung von Dokumenten als Prüfungsleistung



Sicht 2: Bibliotheksverwaltung

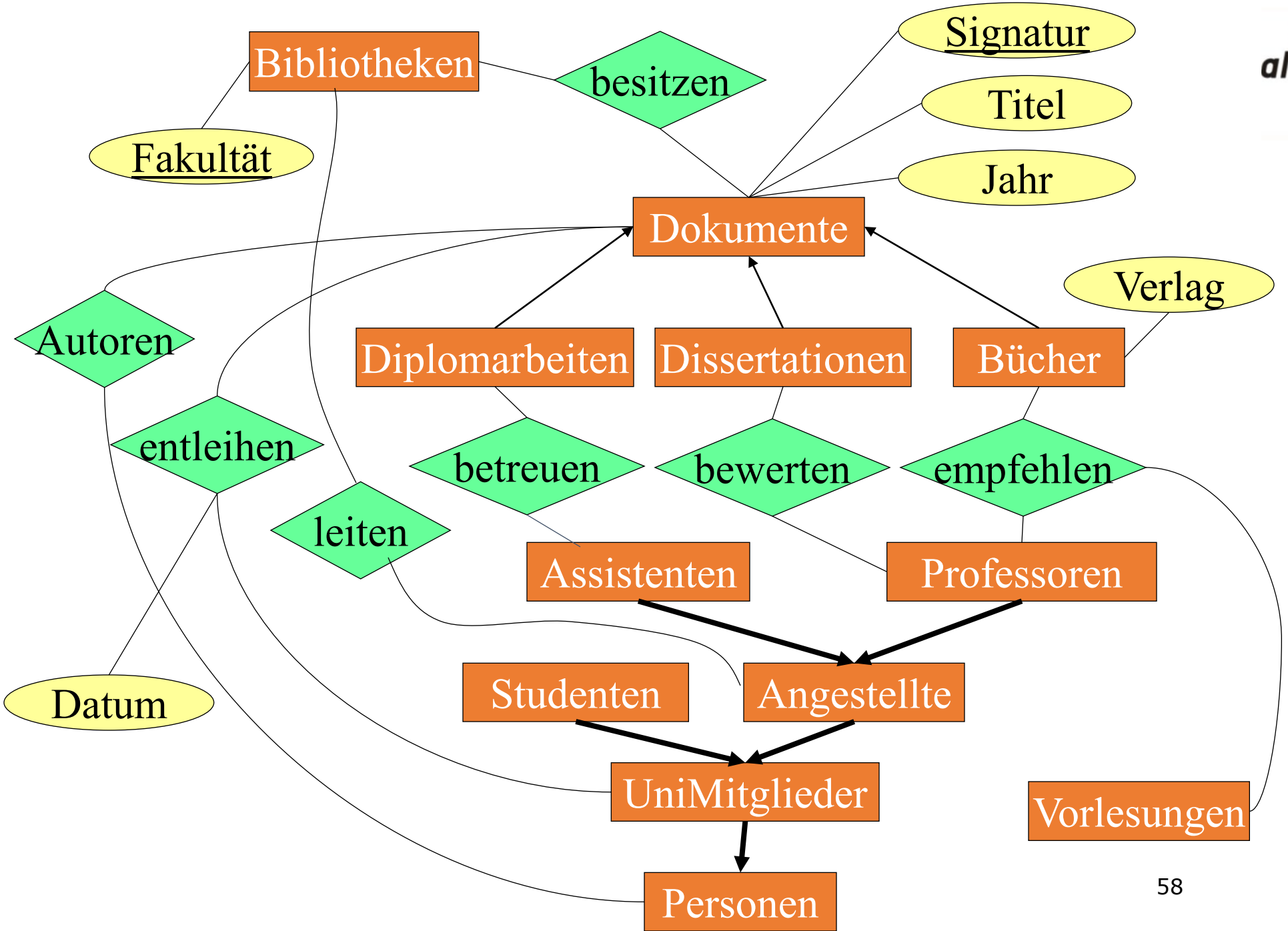


Sicht 3: Buchempfehlungen für Vorlesungen

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