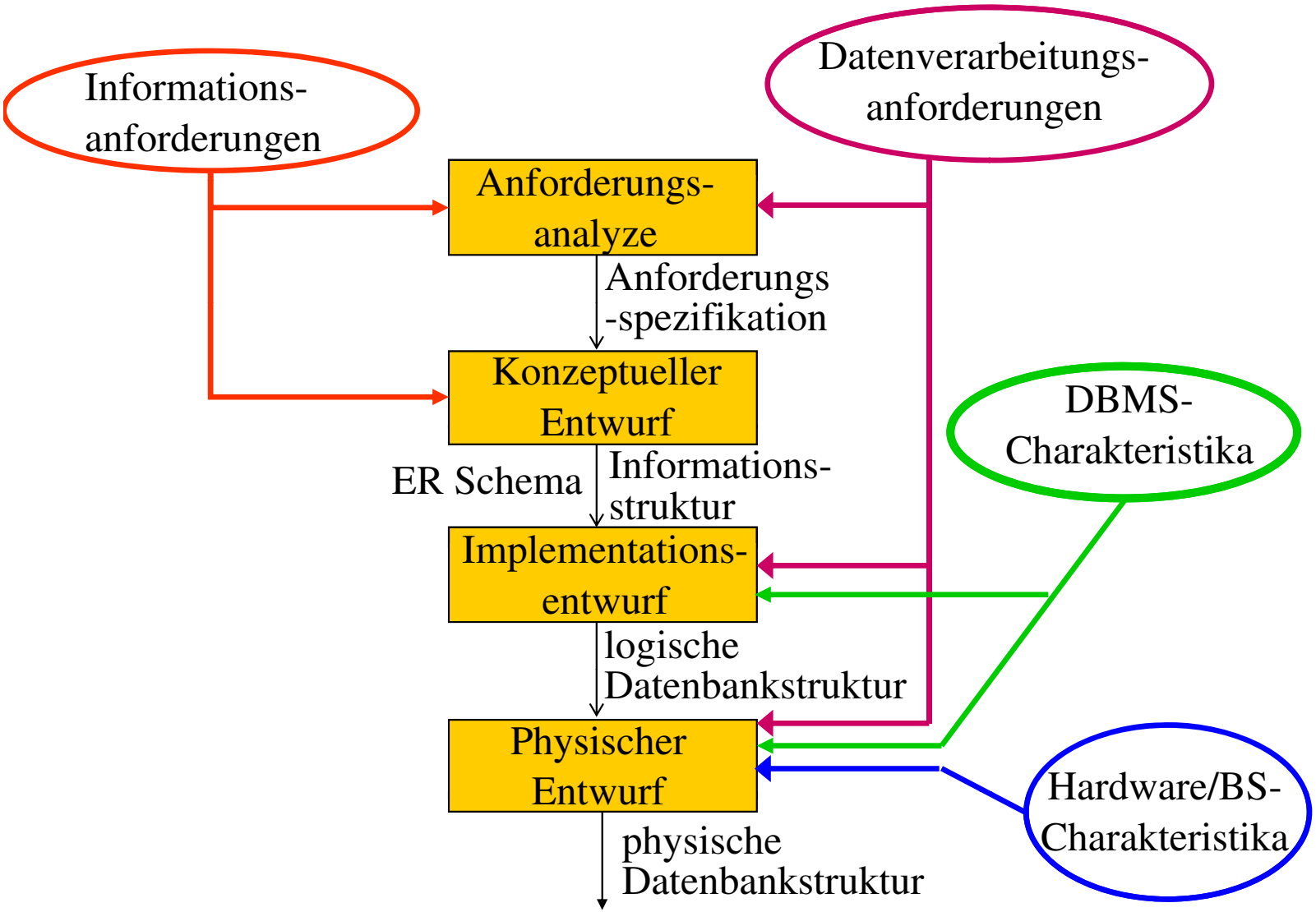
Kapitel 2

Datenbankentwurf

Phasen des Datenbankentwurfs

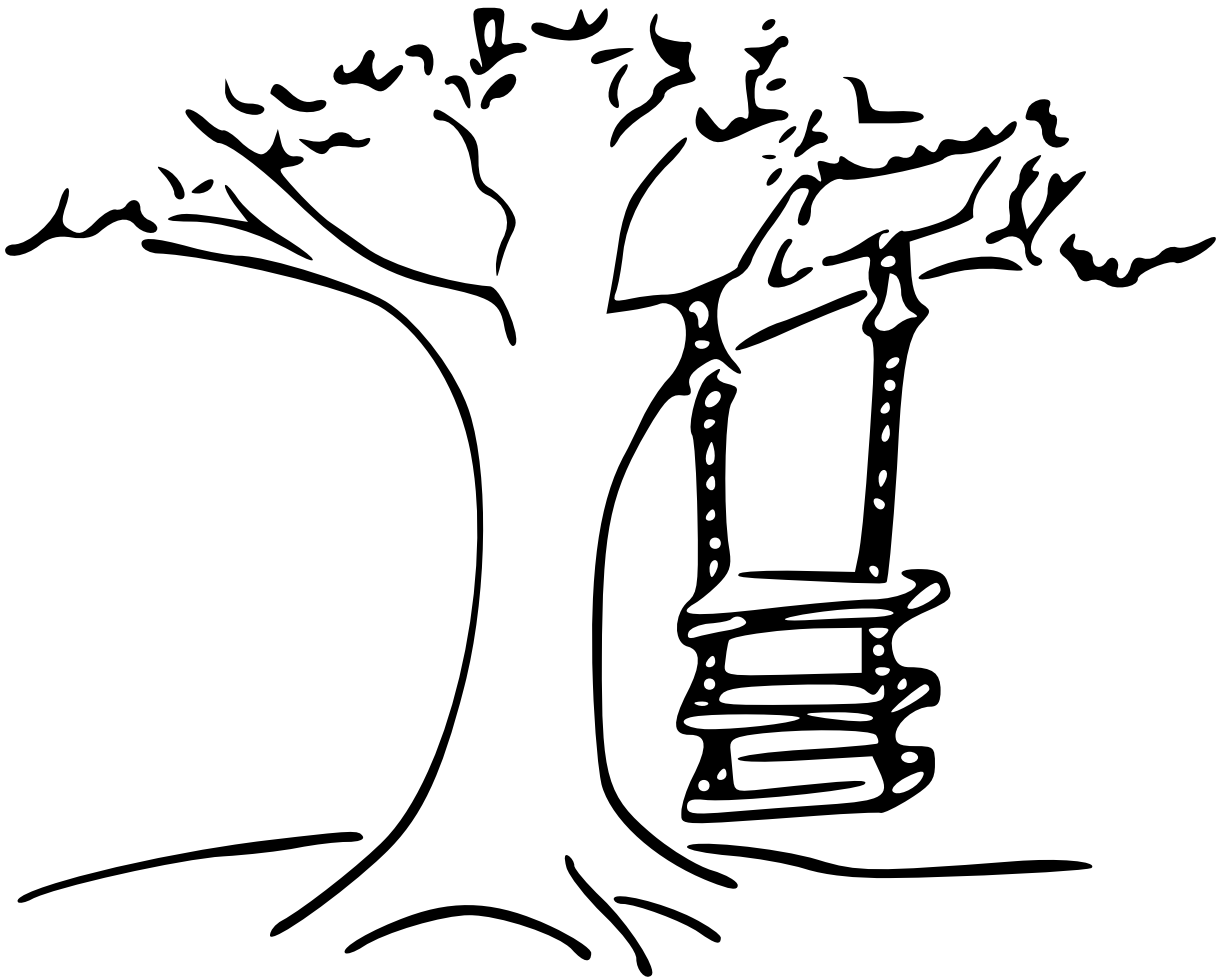


Anforderungsanalyse

- Wird detailierter in Softwaretechnikvorlesungen besprochen
- Wir streifen hier nur kurz dieses Gebiet
- Sehr wichtiger Punkt bei Projekten: Anwender und Entwickler verstehen sich gegenseitig

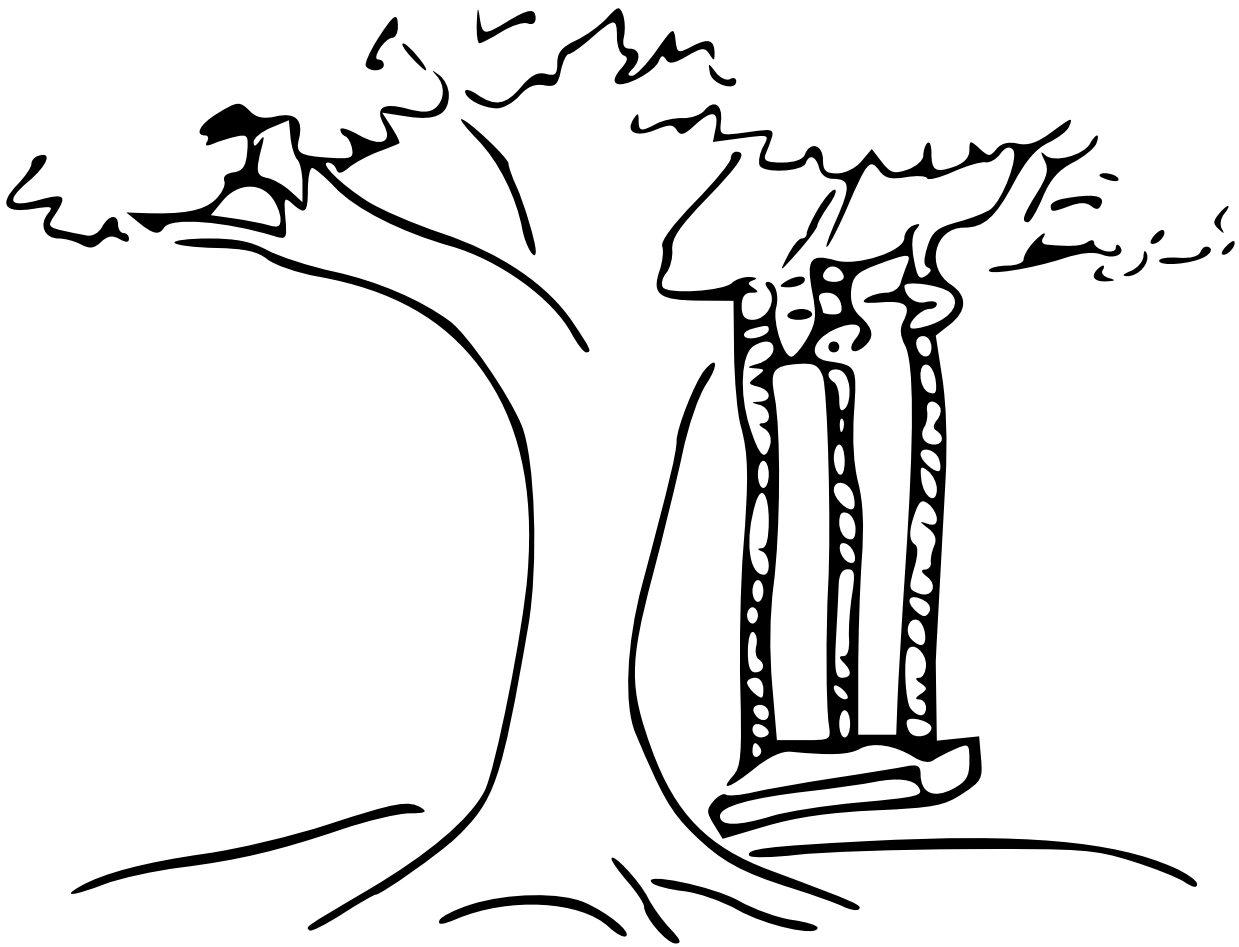
Verstehen

Wie es vom Geldgeber vorgeschlagen wurde:



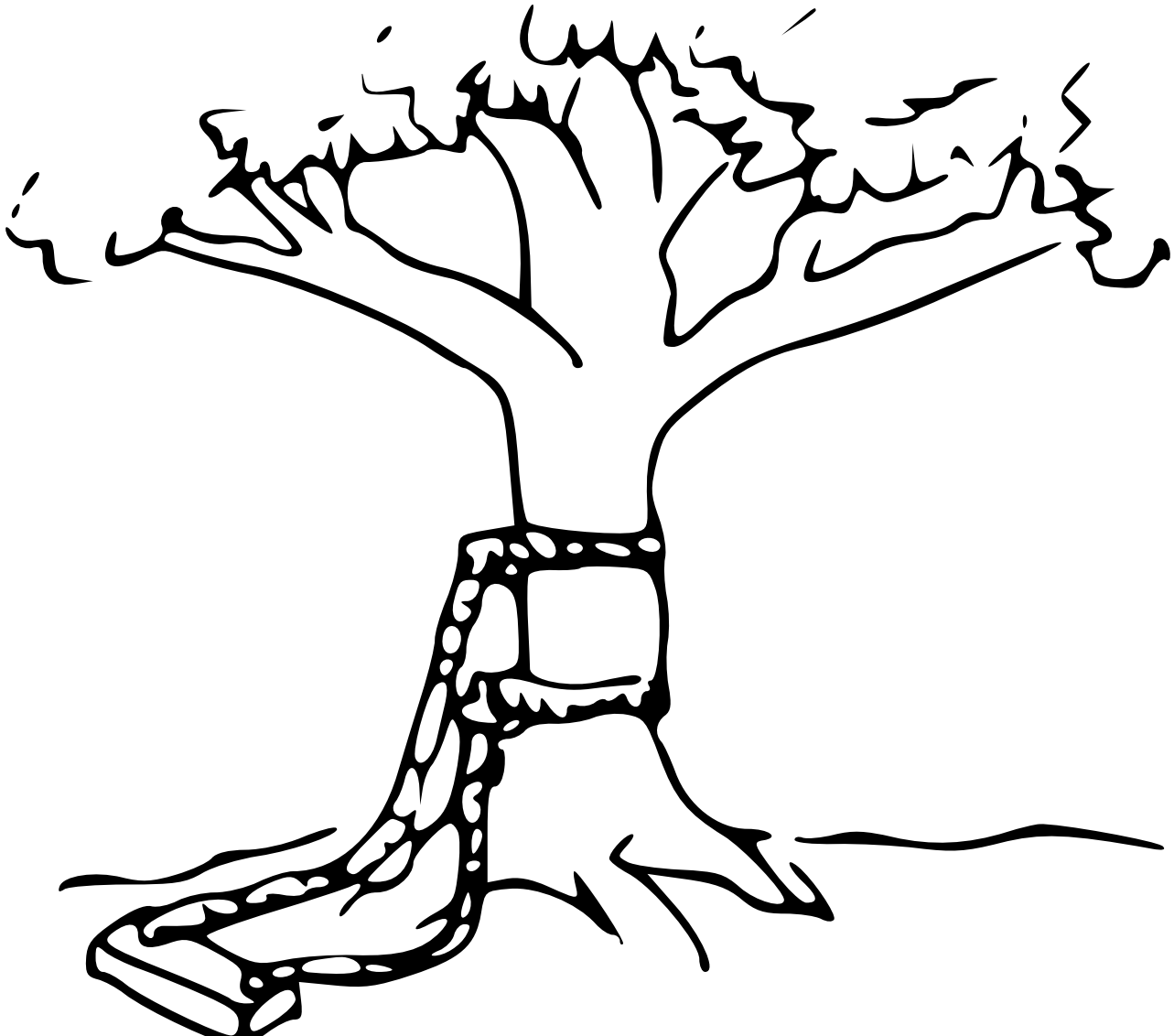
Verstehen(2)

Wie es in der Spezifikation beschrieben wurde:



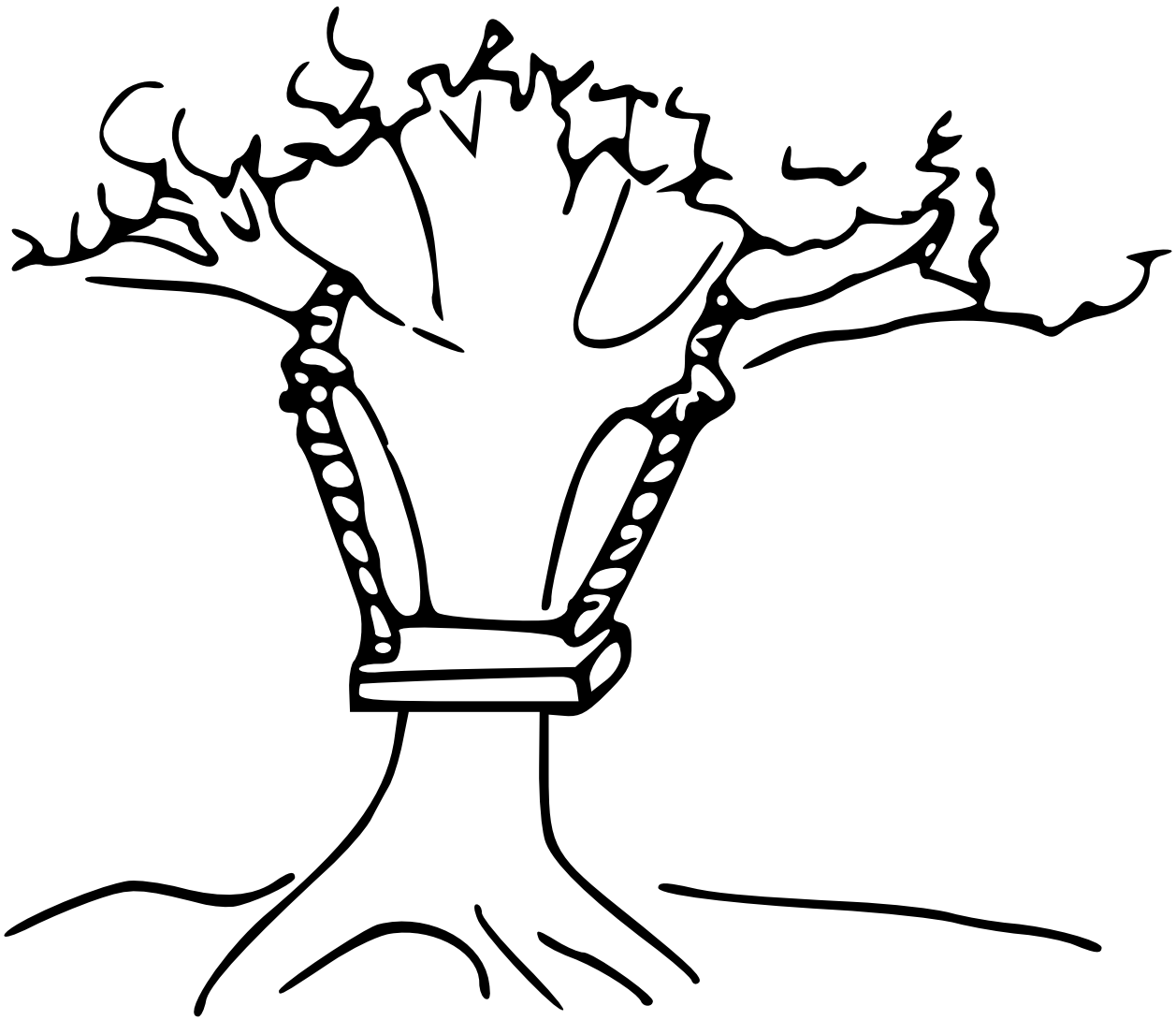
Verstehen(3)

Wie es vom Chefdesigner entworfen wurde:



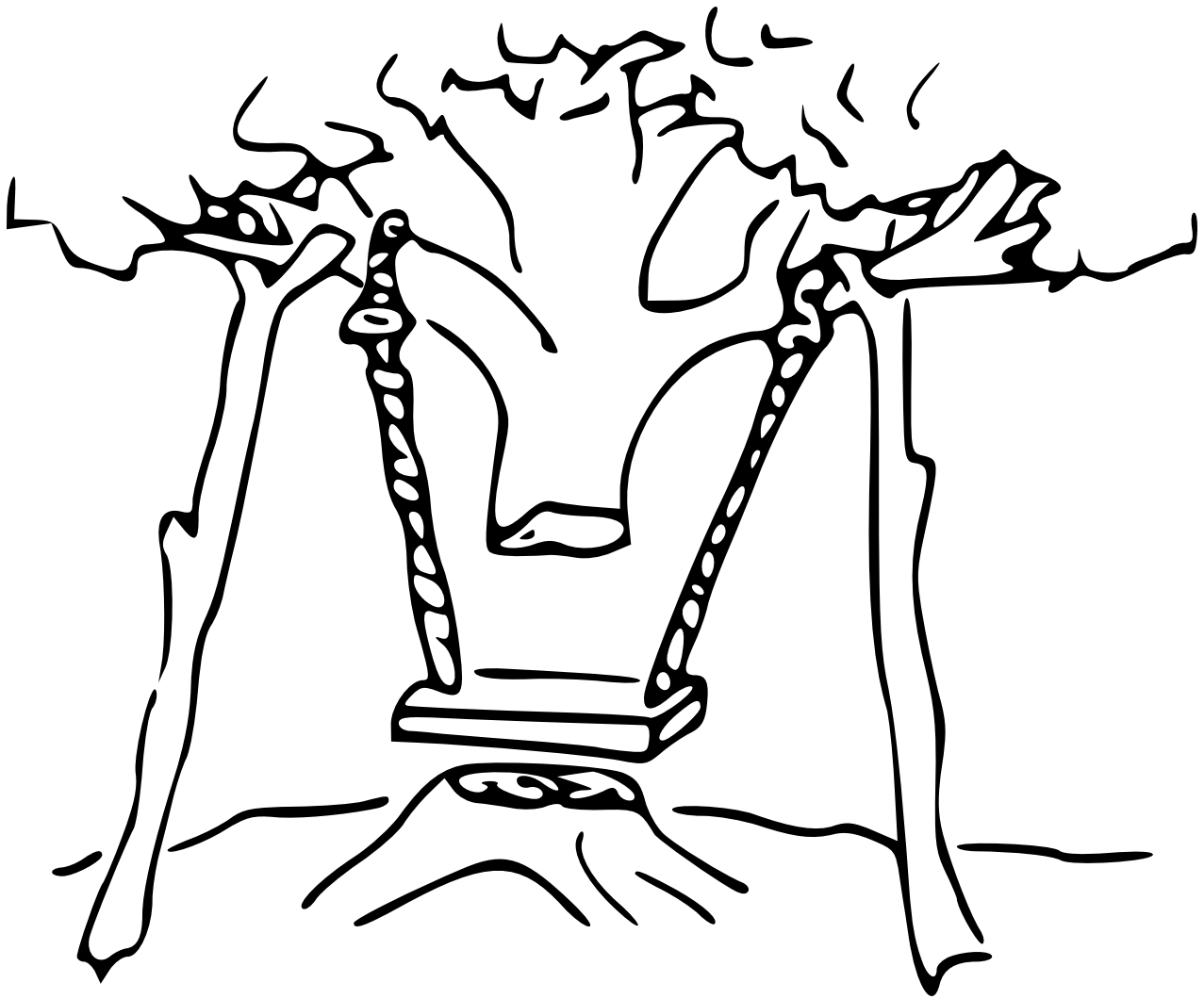
Verstehen(4)

Wie es vom Programmierer implementiert wurde:



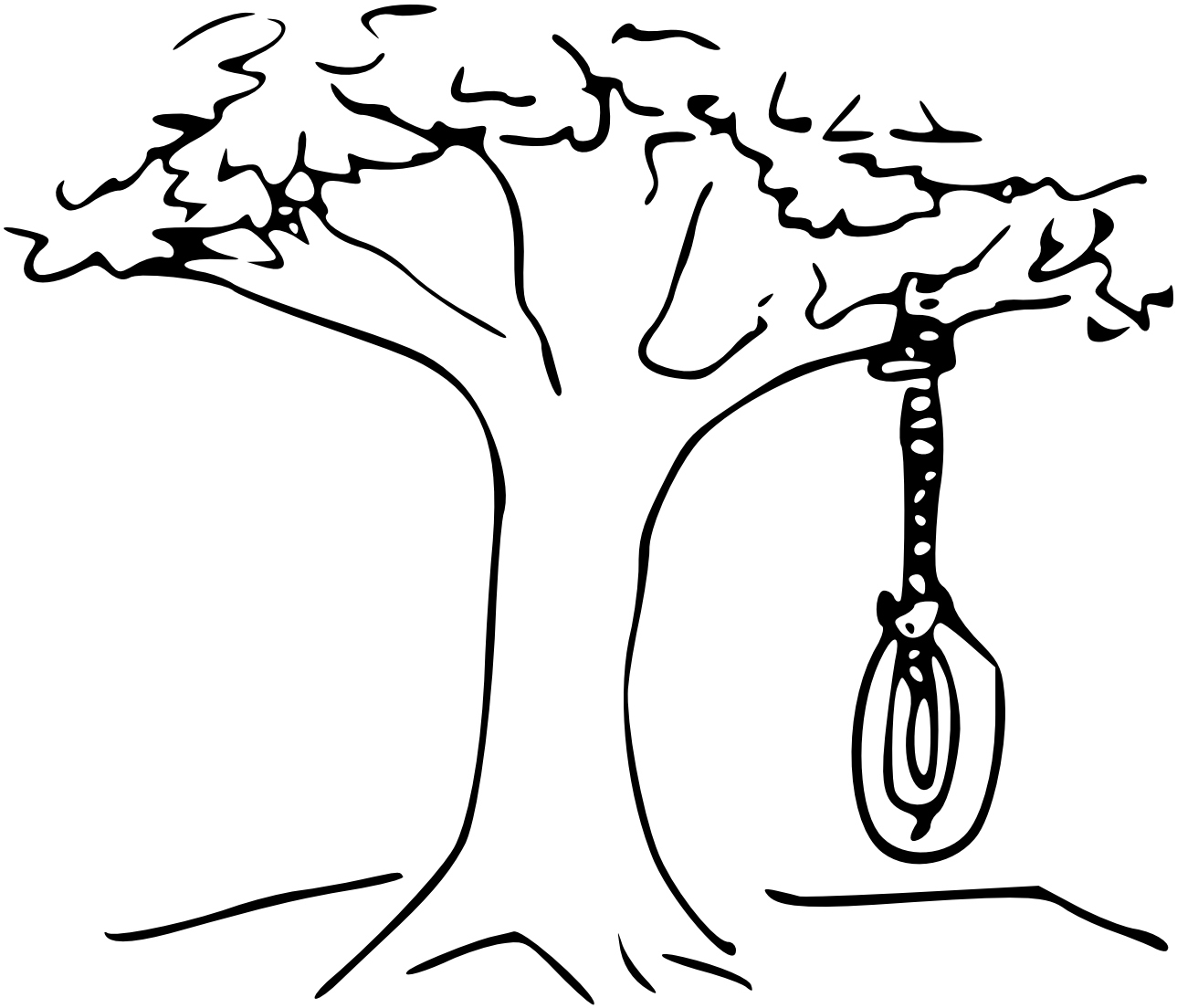
Verstehen(5)

Wie es schließlich in Betrieb genommen wurde:



Verstehen(6)

Was der Kunde eigentlich wollte:



Anforderungsanalyze

Grobes Vorgehen:

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: 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: 90
  - 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

Prozessbeschreibung: *Zeugnissaustellung*

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

Spezifikation

- Hier wird das *was* festgelegt (nicht das *wie*)
- Bestandteile der Spezifikation sind:
  - Prozeßbeschreibungen
  - Use cases
  - Benutzerschnittstelle
  - sonstige Schnittstellen (zu anderen Systemen)

Spezifikation(2)

- Die ideale Spezifikation ist
  - eindeutig
  - vollständig
  - verständlich (für alle Leser)
  - redundanzfrei
  - . . . und in der Realität nicht zu erreichen

Spezifikationsmethoden

- Es gibt eine große Auswahl an Methoden:
  - Strukturierte Analyse
  - Sachbearbeiter-Methode
  - Objektorientierte Methoden
  - ...
- Aber nicht jede Methode ist immer sinnvoll

Erstellung einer Spezifikation

- Die eigentliche Analyse ist ein iterativer (oft auch sehr politischer) Prozeß
  - Anwender erzählt Entwickler was er gern hätte
  - Entwickler schreibt alles (was er verstanden hat) in seiner "Sprache" auf ...
  - ... und übersetzt es in die "Sprache" des Anwenders
  - Dies wird dem Anwender gezeigt, mit dem Ergebnis, daß vieles noch nicht stimmt
  - Änderungswünsche werden aufgenommen
  - Zurück zum zweiten Schritt

Modellierung der Daten

- Besonders wichtig für Datenbanksysteme ist die Modellierung der Daten, die in der Datenbank gespeichert werden sollen
- Dies wird im konzeptuellen Entwurf getrennt behandelt

Konzeptueller Entwurf

- Entity-Relationship-Modell (ER-Modell) wird häufig für konzeptuellen Entwurf eingesetzt
- Ein ER-Schema ist eine graphische Repräsentation des konzeptuellen Modellierung der Daten
- Identifiziert *Entitäten* (Entities) und *Beziehungen* (Relationships) zwischen Entitäten einer Anwendungsdomäne

Konzeptueller Entwurf(2)

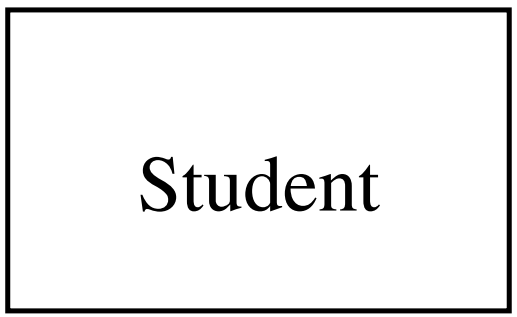
- Ein ER-Schema
  - enthält alle Informationseinheiten, die im zu implementierenden DBS enthalten sein sollen
  - dient auch der Kommunikation mit den späteren Benutzern (um Vollständigkeit und Korrektheit sicherzustellen)
  - kann (relativ) einfach in ein logisches Schema transformiert werden

Grundlegende Bausteine

- Entitätentyp
- Attribut
- Schlüssel
- Relationship (Beziehungstyp)
- Rolle

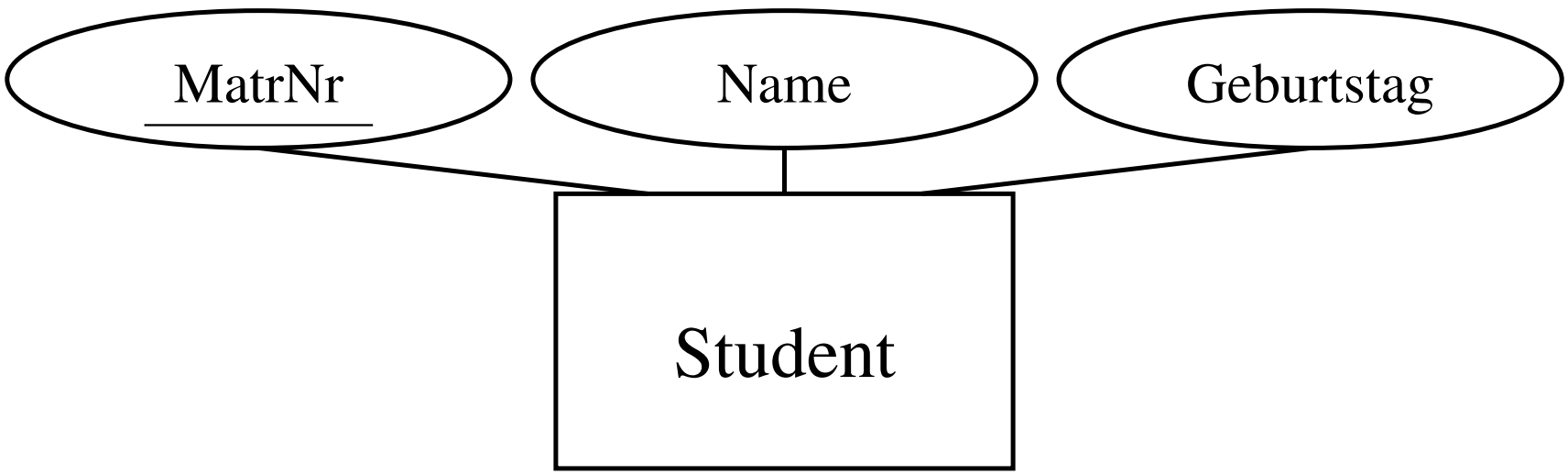
Entitäten

- Entität: ein Objekt aus der realen Miniwelt, das sich von anderen Objekten unterscheidet (z.B. Dinge, Personen, Konzepte)
- Entitätentypen werden durch Rechtecke repräsentiert:



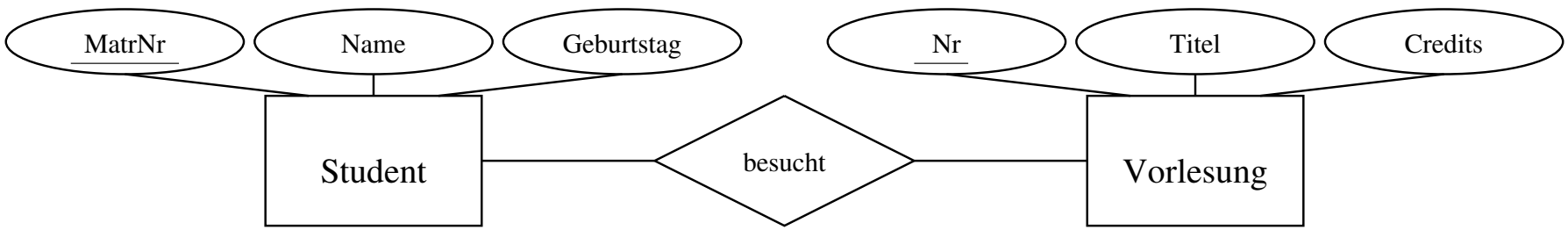
Attribute

- Eine Entität wird durch Attribute beschrieben
- Zu jedem Attribut gehört ein Wertebereich (z.B. Integer, String) der die möglichen Werte dieses Attributs beschreibt
- Ein Attribut das eine Entität eindeutig beschreibt wird *Schlüssel* genannt
- Attribute werden durch eine Ellipse repräsentiert:



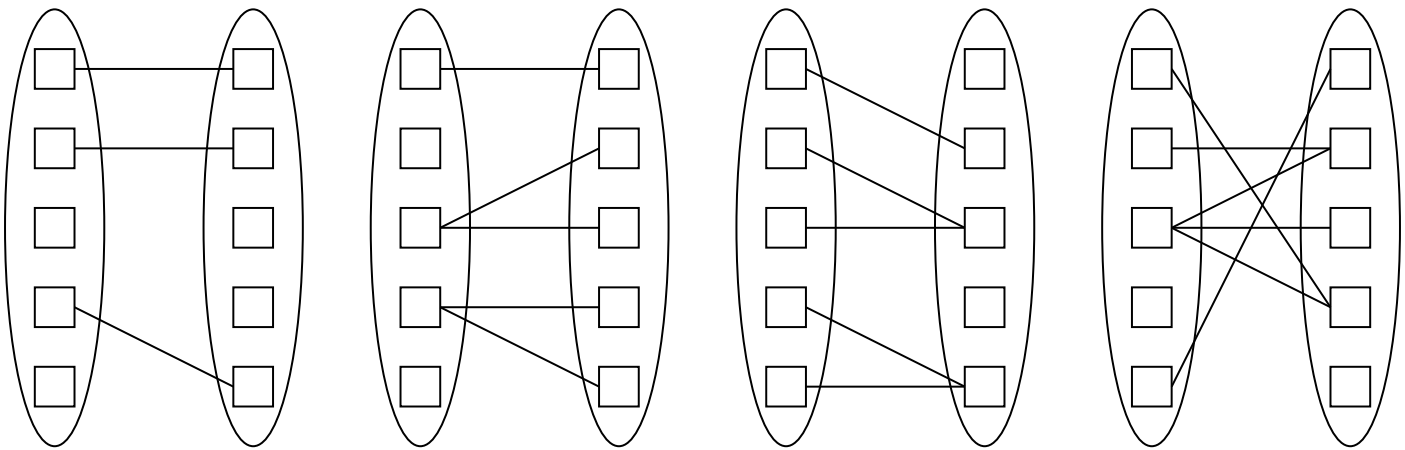
Beziehungen (Relationships)

- Relationship: verbindet zwei oder mehr Entitätstypen miteinander
- Wird durch eine Raute repräsentiert:



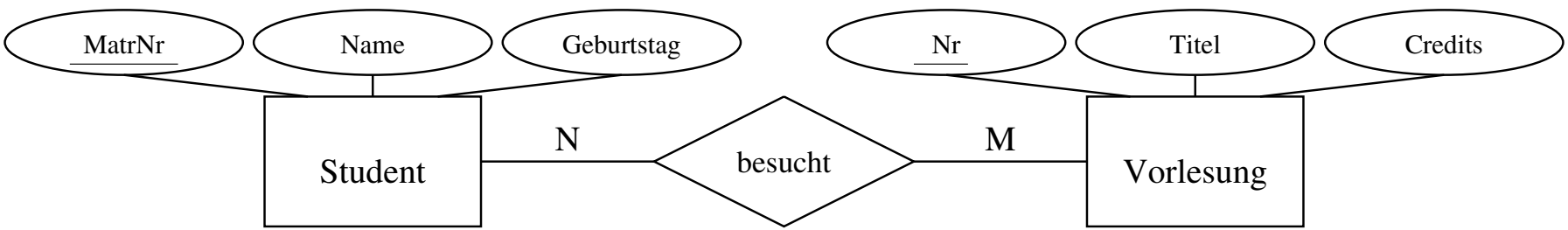
Funktionalitäten

- Weitergehende Beschreibung von Beziehungen durch Funktionalitäten
- Funktionalitäten drücken aus mit wieviel anderen Entitäten eine Entität eine Beziehung eingeht

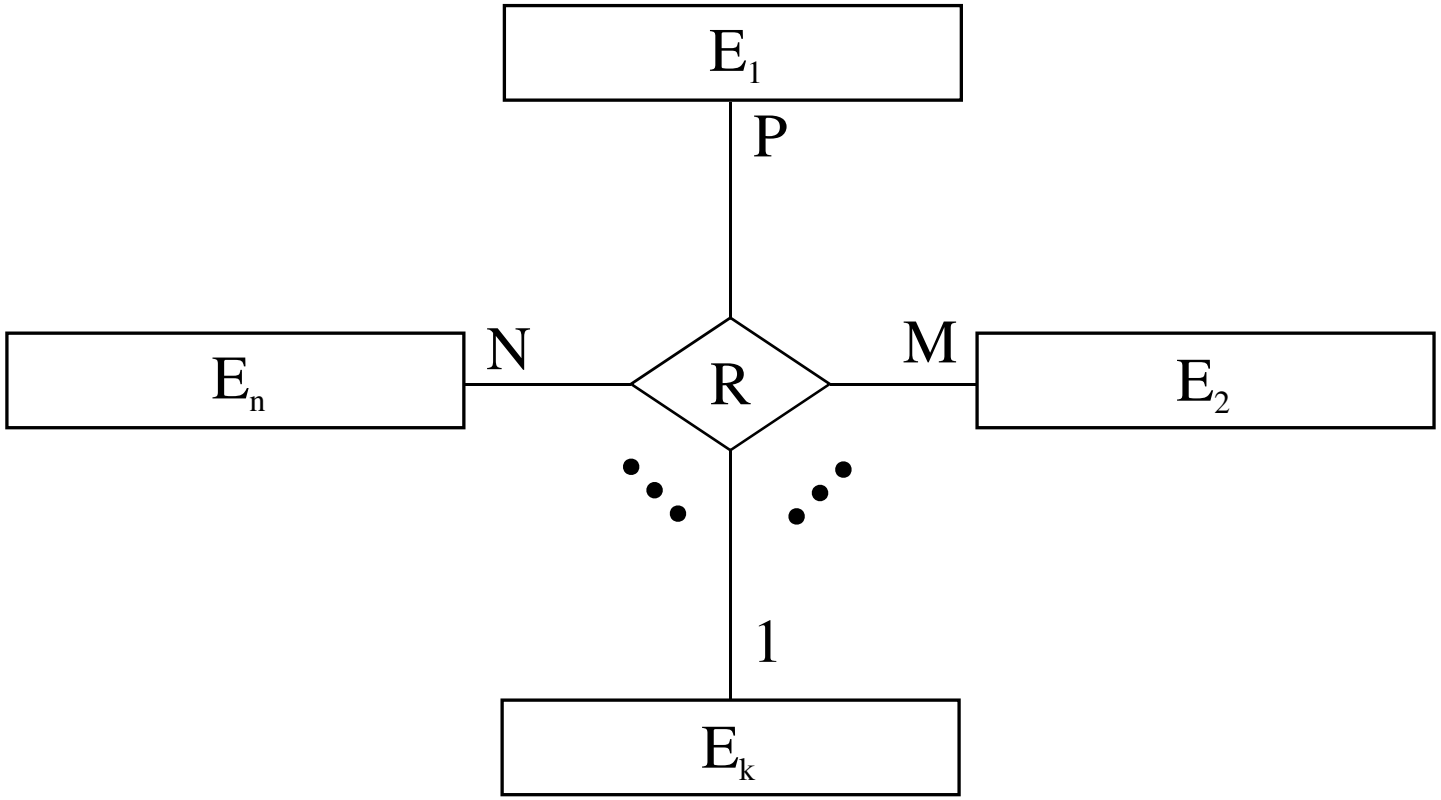


Funktionalitäten(2)

- `besucht` ist eine N:M Beziehung:



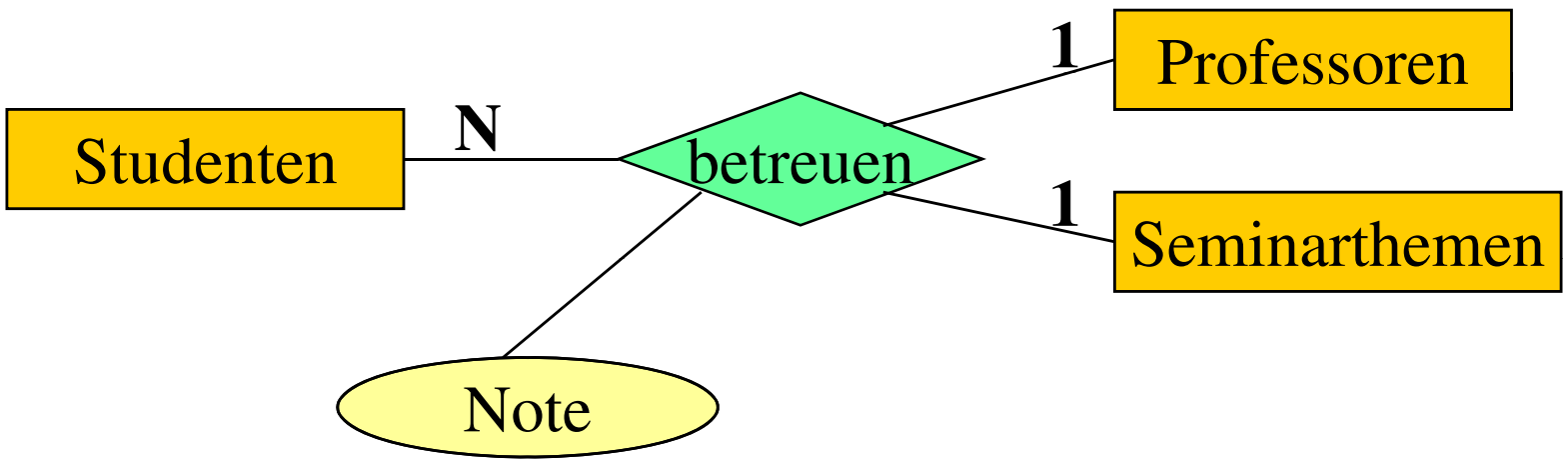
Funktionalitäten(3)



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

Funktionalitäten(4)

Beispielbeziehung: *betreuen*



$$betreuen : Professoren \times Studenten \rightarrow Seminarthemen$$

$$betreuen : Seminarthemen \times Studenten \rightarrow Professoren$$

Funktionalitäten(5)

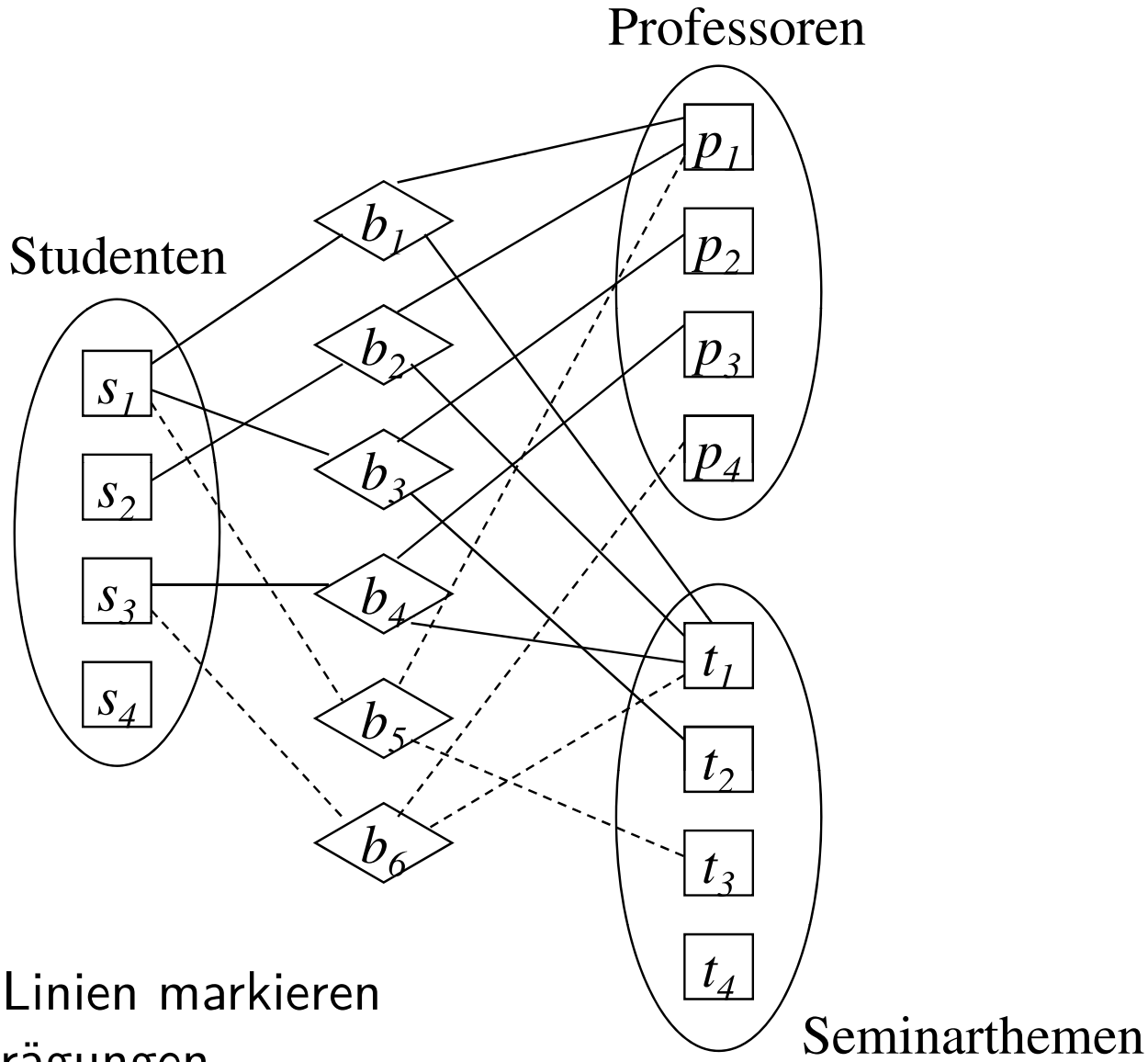
Dadurch erzwungene Konsistenzbedingungen:

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:

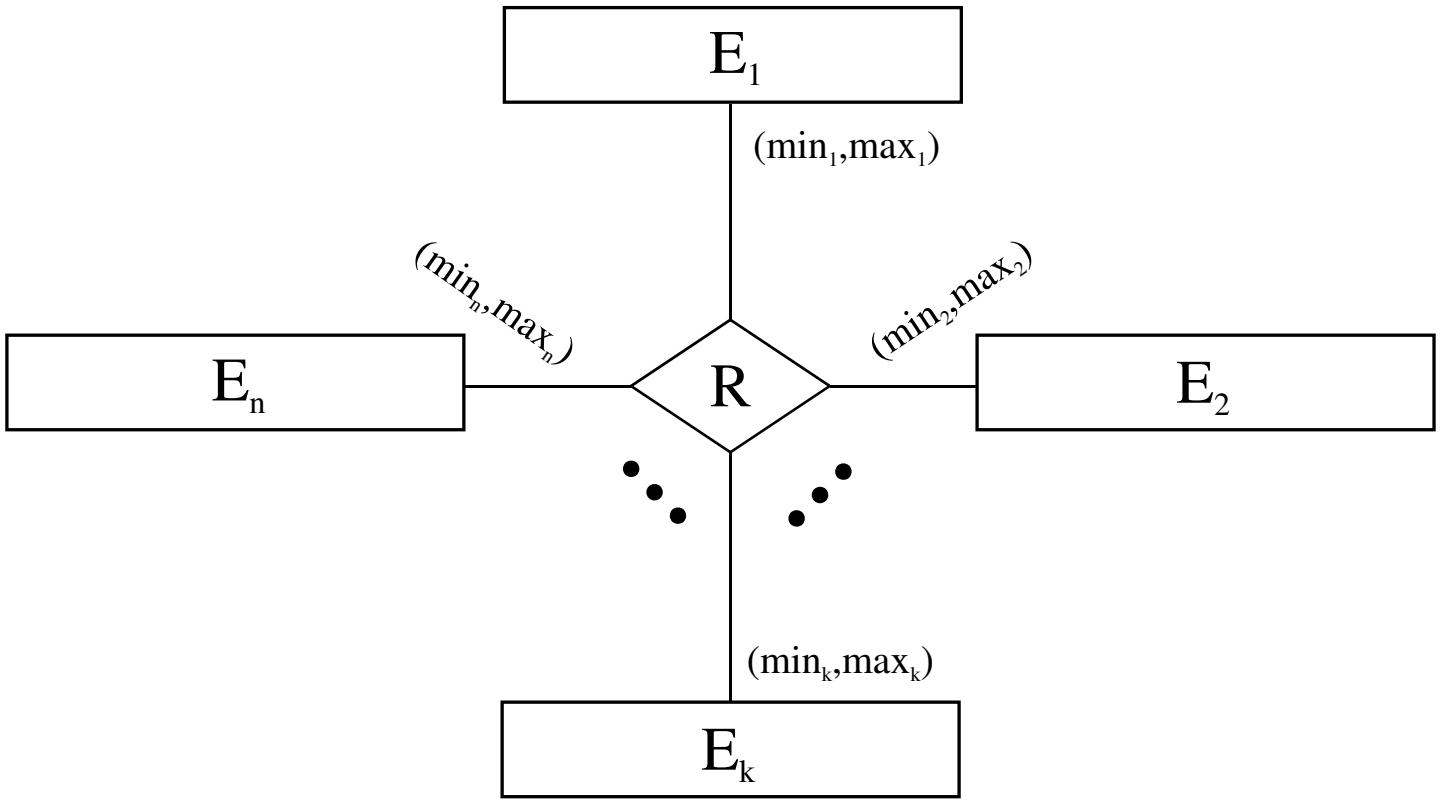
- Professoren können dasselbe Seminarthema „wiederverwenden“ – also dasselbe Thema auch mehreren Studenten erteilen.
- Ein Thema kann von mehreren Professoren vergeben werden – aber an unterschiedliche Studenten.

Funktionalitäten(6)



Gestrichelte Linien markieren illegale Ausprägungen.

Funktionalitäten(7)

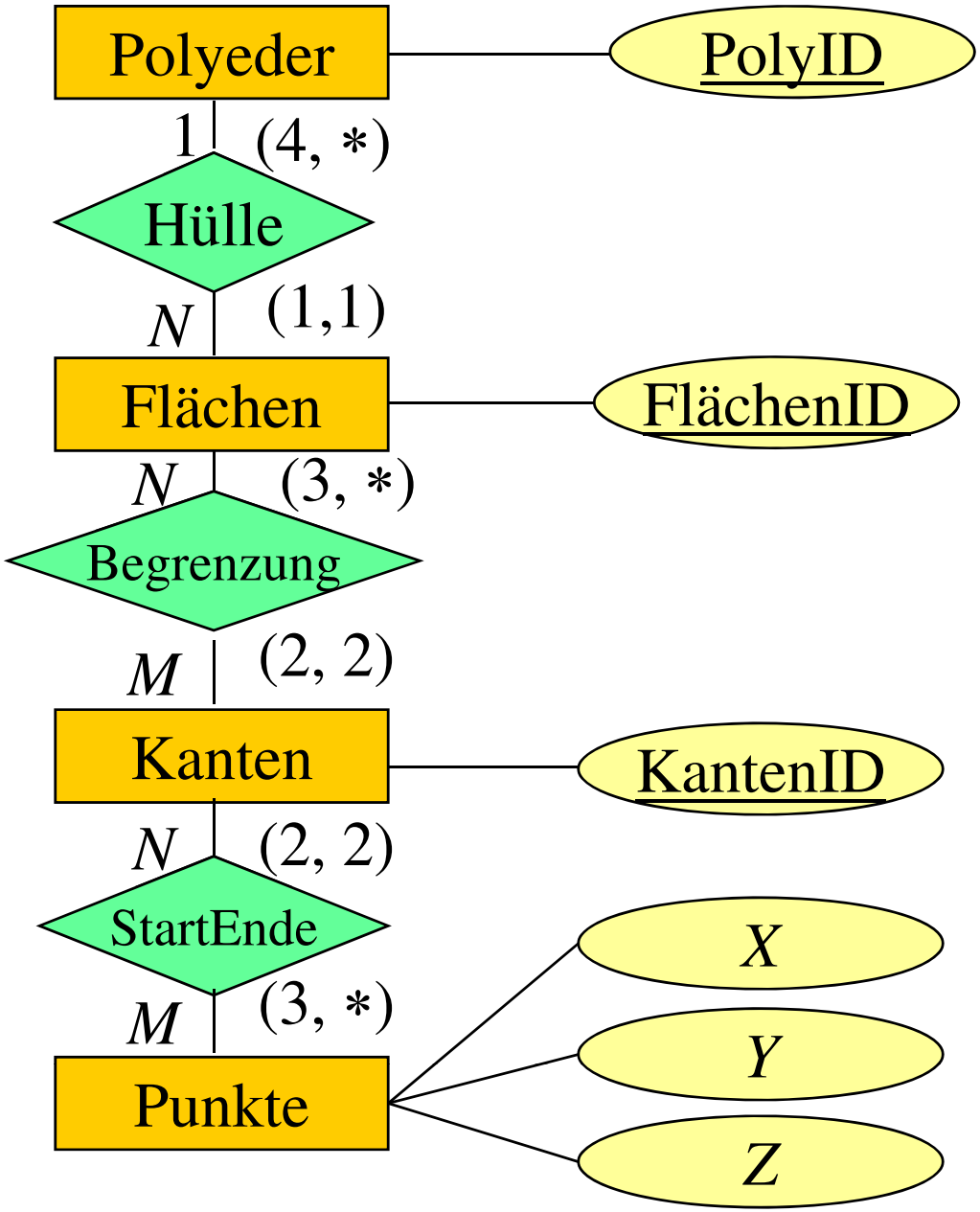


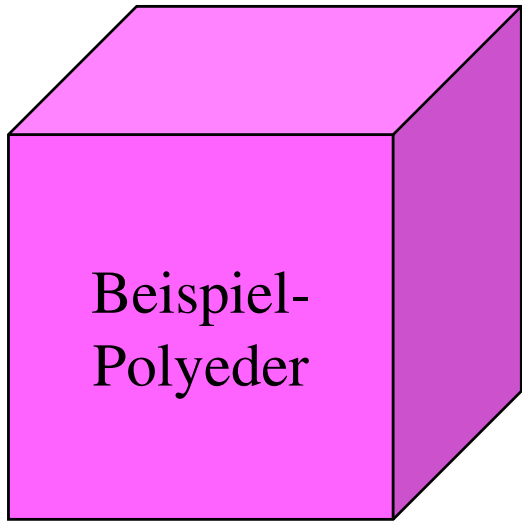
$$R \subseteq E_1 \times ... \times E_i \times ... \times E_n$$

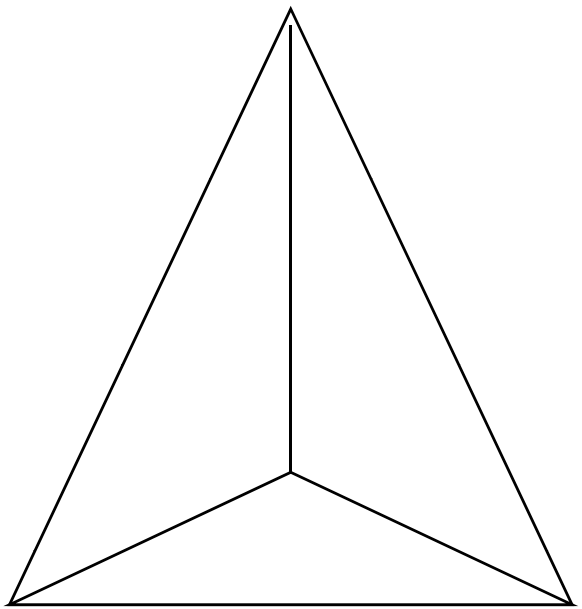
Für jedes $e_i \in E_i$ gibt es

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

Funktionalitäten(8)

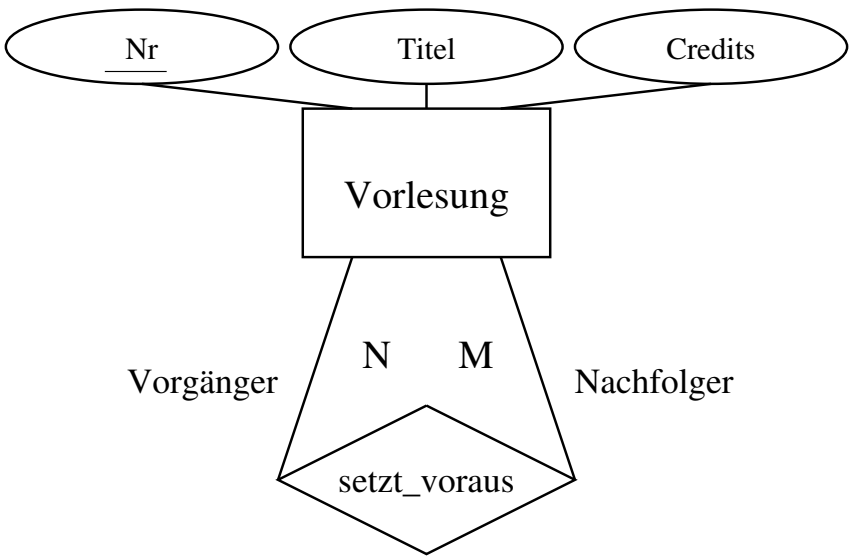




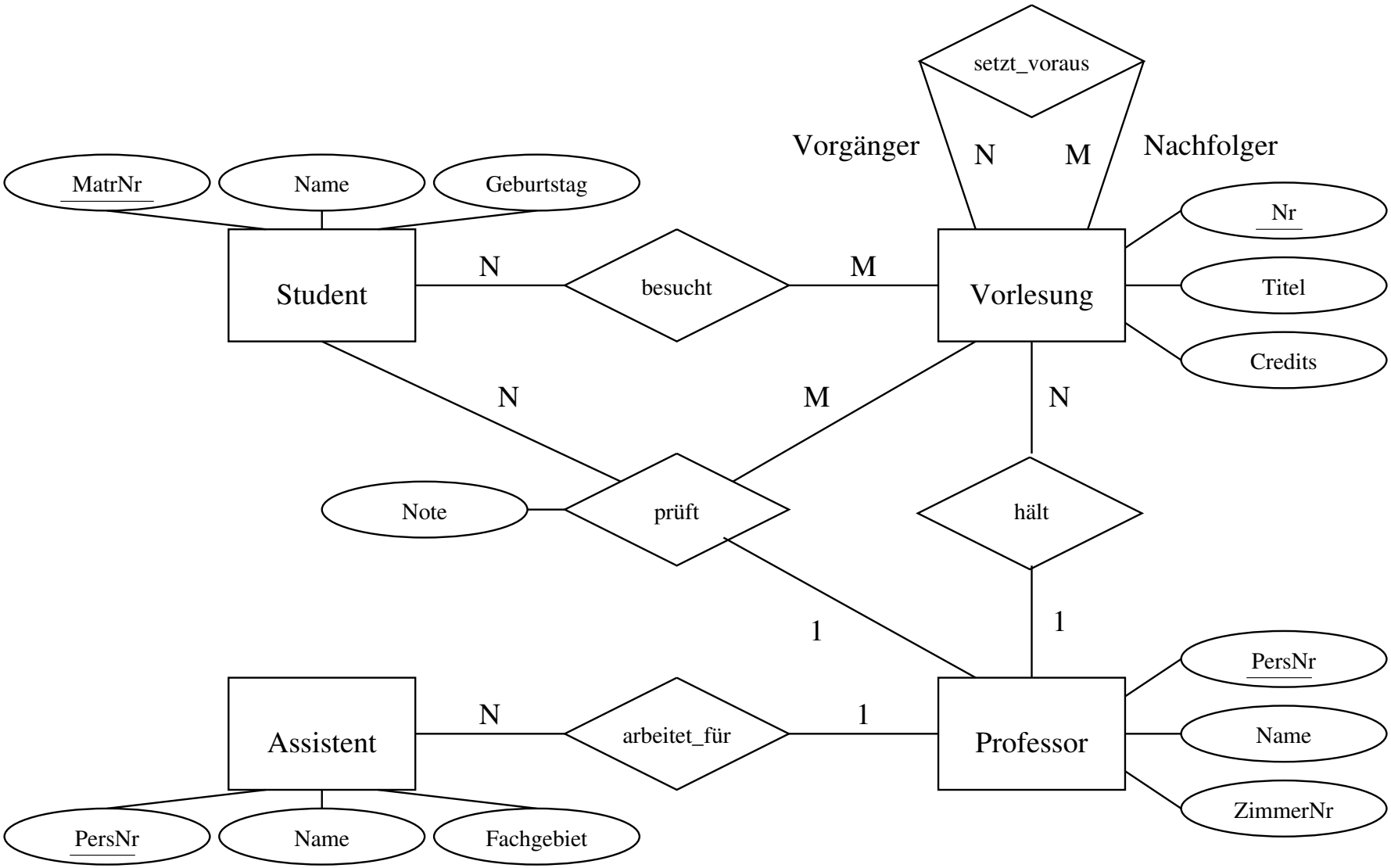


Rollen

- Rollen beschreiben in welcher Weise Entitäten an einer Beziehung teilnehmen
- Besonders nützlich zur Beschreibung rekursiver Beziehungen



Vollständiges Schema

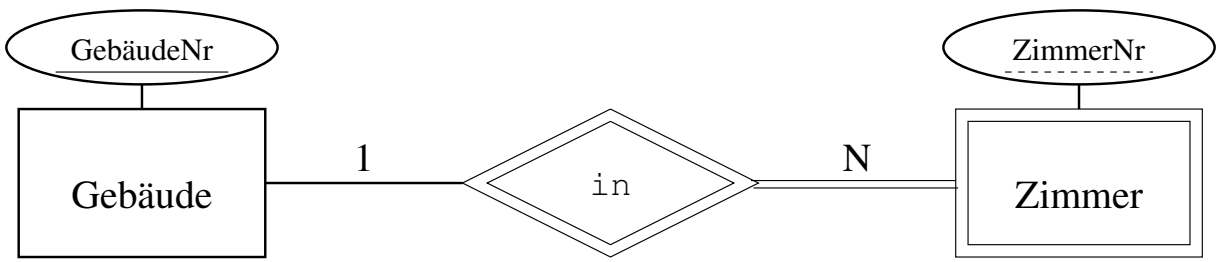


Weiterführende Konzepte

- Schwache Entitäten
- Generalisierungen
- Aggregationen

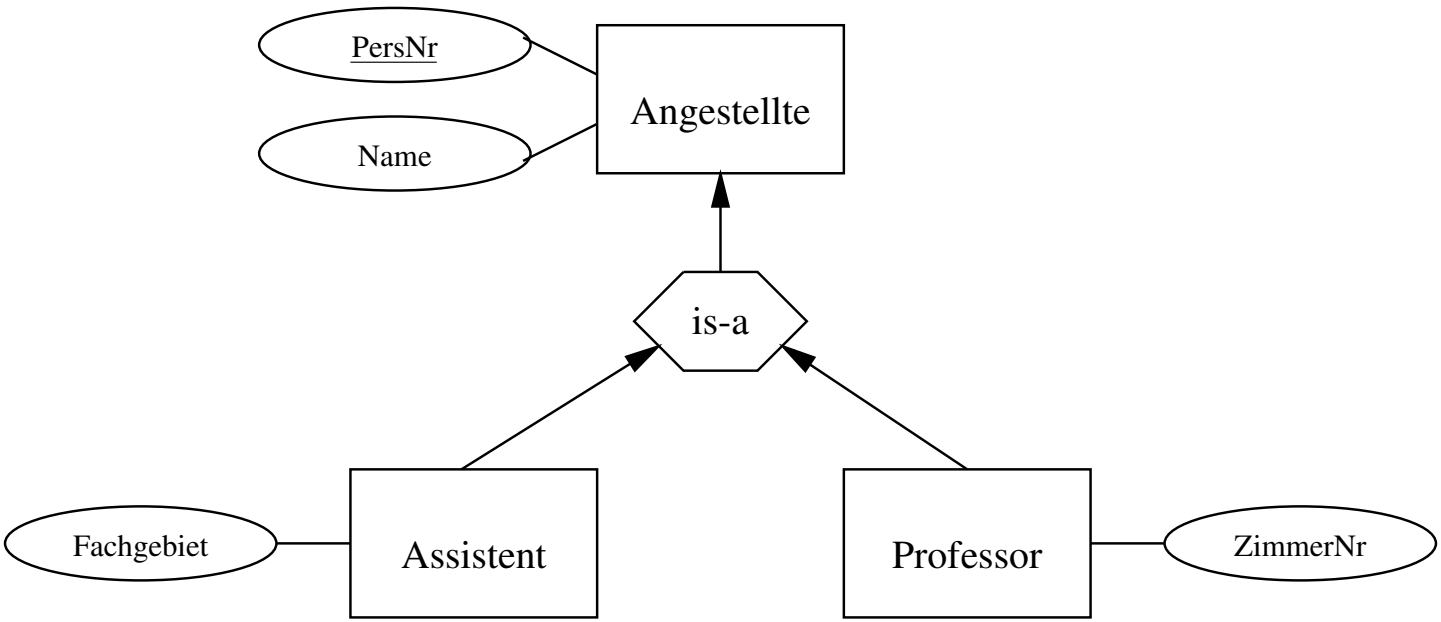
Schwache Entitäten

- Schwache Entität: kann nicht als eigenständiges Gebilde existieren
- Braucht eine "'starke"' Entität, um eindeutig identifiziert werden zu können
- Eine schwache Entität und zugehörige starke Entität gehen eine 1:N (oder seltener 1:1) Beziehung ein



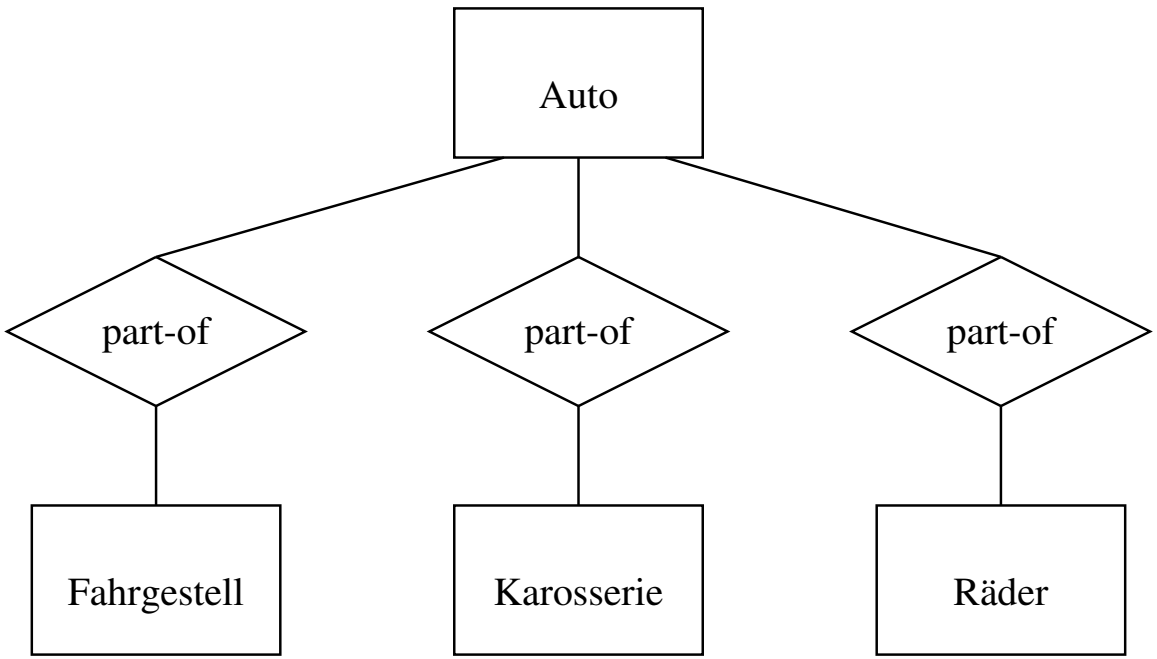
Generalisierung

- Wie in objekt-orientierten Programmiersprachen, können Attribute von einem Obertyp an einen Untertyp vererbt werden



Aggregation

- Wird bei der Modellierung von ist-Teil-von Beziehungen benutzt



Entwurfsentscheidungen

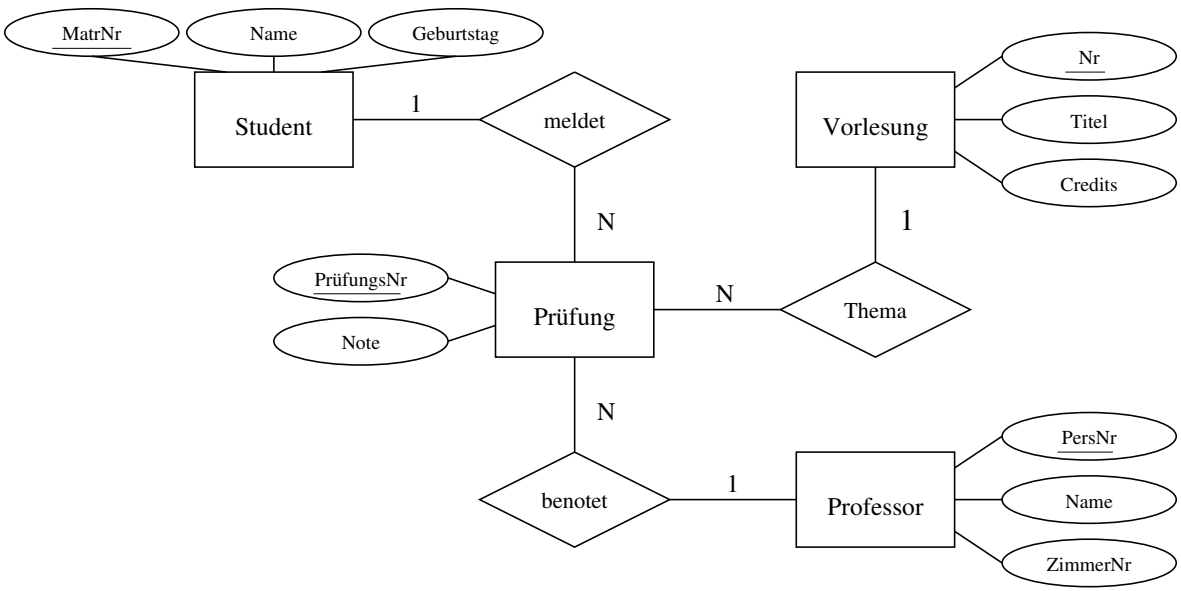
- Entität vs. Attribut
- Entität vs. Beziehung
- Binäre vs. ternäre Beziehungen
- Connection Trap

Entität vs. Attribut

- Ein eigener Entitätstyp ist flexibler, ein Attribut einfacher
- Beispiel: wir wollen Adressinformationen von Kunden speichern
  - Falls es mehrere Adressen pro Kunde geben kann, modelliere Adresse als eigene Entität (da Attribute nicht mengenwertig sein dürfen)
  - Falls eine Adresse in einzelne Bestandteile zerlegt werden soll (Straße, Hausnummer, PLZ, Stadt, usw.), dann zerlege Adresse in einzelne Attribute (diese können dann eventuell in eigene Entität ausgegliedert werden)

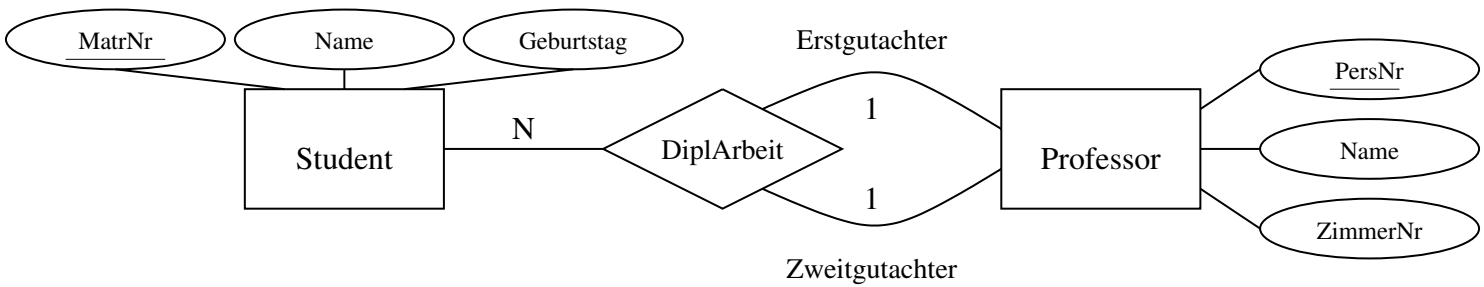
Entität vs. Beziehung

- Entität ist flexibler, Beziehung einfacher
- Beziehungen haben keinen eigene Identität: diesselben Entitäten können nicht mehr als einmal eine Beziehung eingehen
- Beispiel: mehr als eine Note für die gleiche Vorlesung vom gleichen Professor:



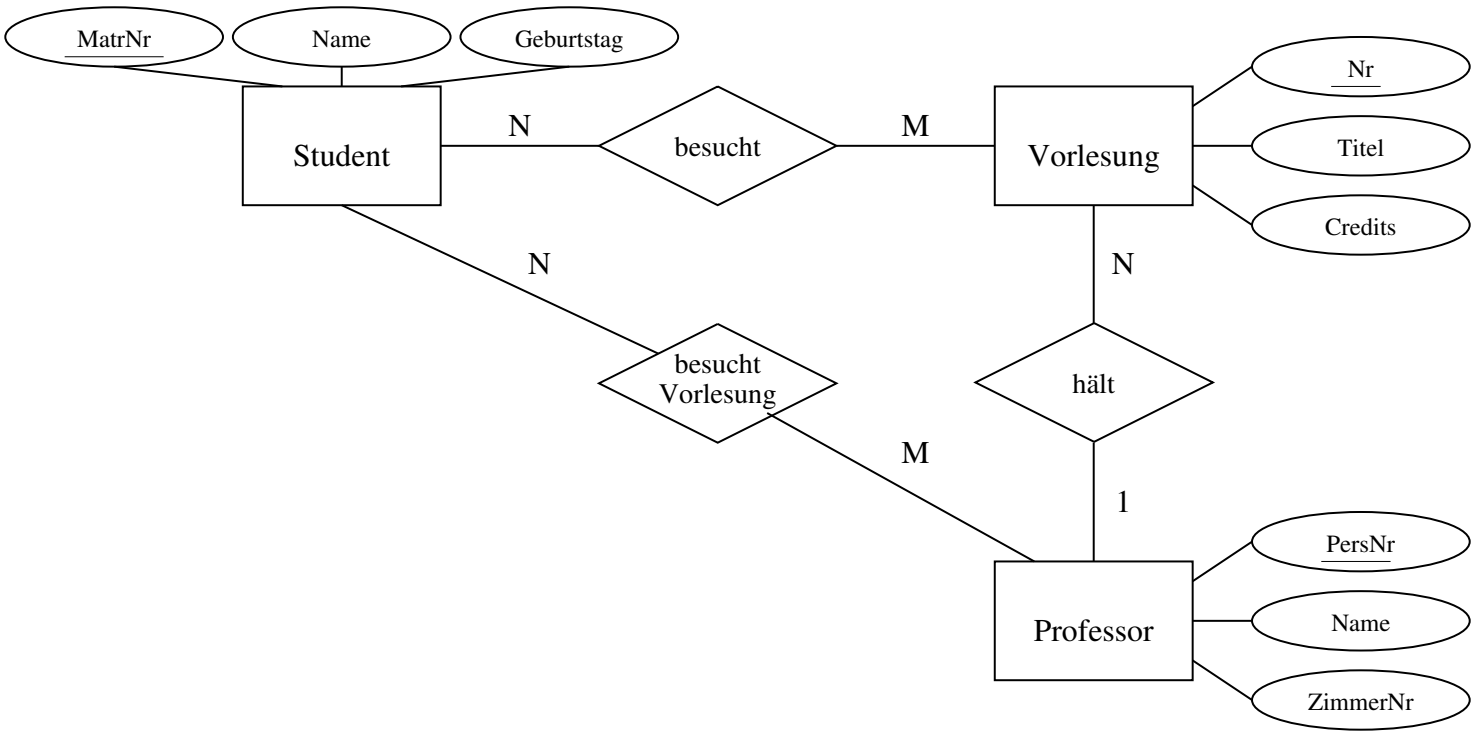
Binäre vs. ternäre Beziehung

- Echte ternäre Beziehungen benötigen alle drei teilnehmenden Entitäten, um Beziehung zu beschreiben
- Beziehung `prüft` im vollständigen Schema ist echt ternär
- Die folgende Beziehung ist es nicht:



Connection Trap

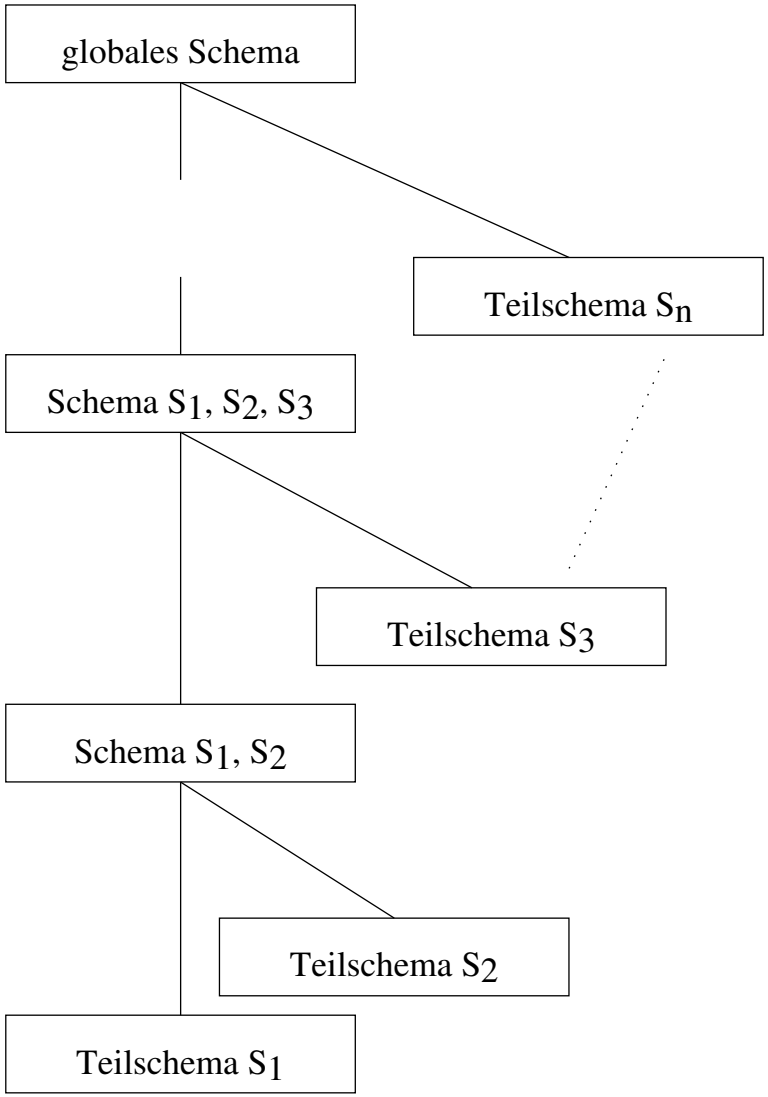
- Eine Falle in die man leicht fällt ist die connection trap: redundante, zyklische Beziehungen
- `besucht Vorlesung` ist redundant, sie wird bereits durch `besucht` und `hält` ausgedrückt



Schemakonsolidierung

- Oft werden Anwendugen von verschiedenen Personen modelliert, die auch verschiedene Blickpunkte auf die Anwendung haben
- Es entstehen Teilschemata, die in ein Gesamtschema zusammengeführt werden müssen

Schemakonsolidierung(2)

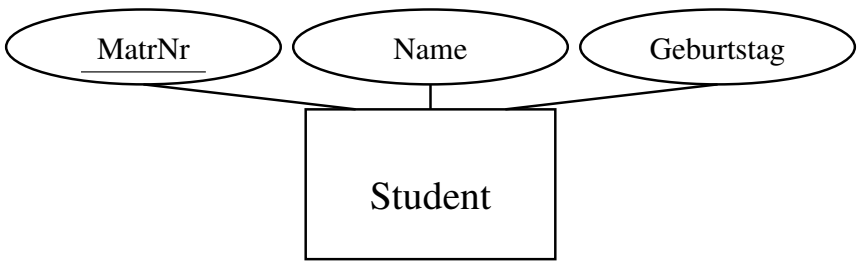


Übersetzung ER → Relational

- Für ein relationales DBMS muß ein ER-Schema in ein relationales Schema übersetzt werden
- Dies geschieht in drei Schritten:
  - Übersetzung der Entitäten
  - Übersetzung der Beziehungen
  - Vereinfachung des Schemas

Übersetzung von Entitäten

- Wandle jede Entität in eine Relation um
- Jedes Attribut der Entität wird zu einem Attribut der Relation (Schlüssel der Entität wird Primärschlüssel der Relation)



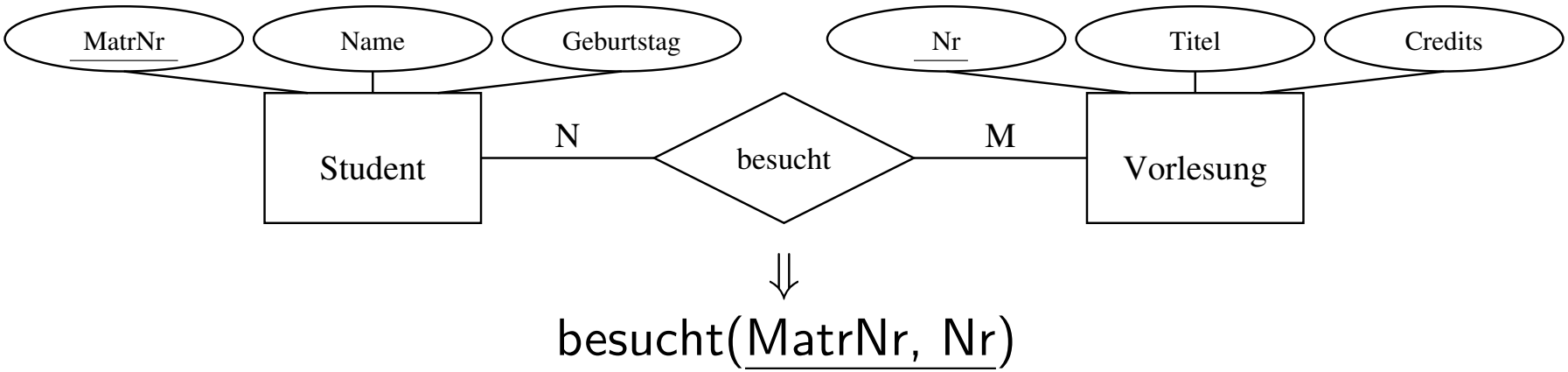
$\Downarrow$

Student(<u>MatrNr</u>, Name, Geburtstag)

Übersetzung von Beziehungen

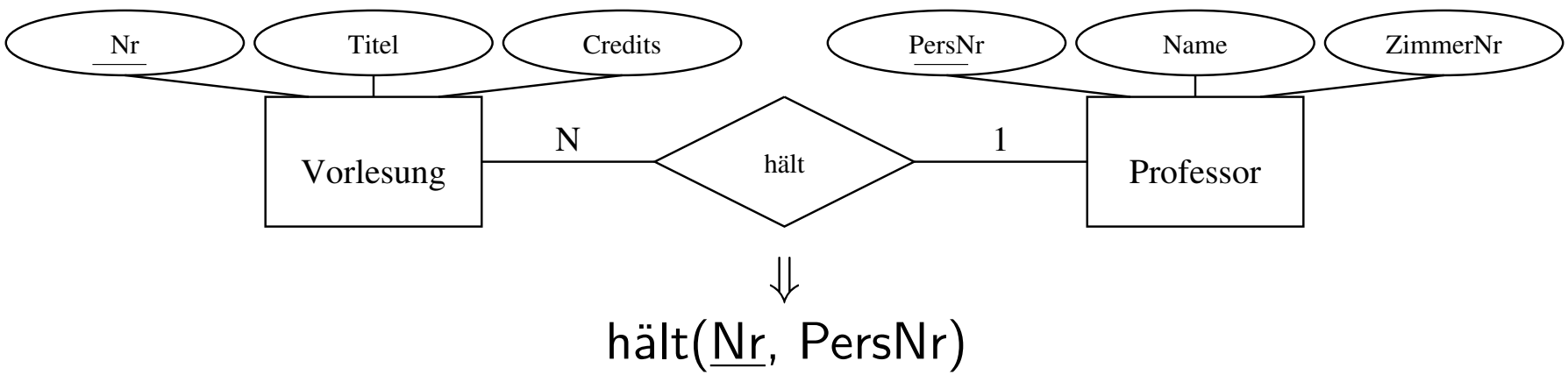
- Wandle jede Beziehung in eine Relation um
- Attribute dieser Relation sind die Schlüssel der teilnehmenden Relationen (+ Attribute der Beziehung selbst)
- Der Schlüssel einer (binären) Beziehung hängt von der Art der Beziehung ab
  - N:M - Schlüssel setzt sich aus Kombination beider Entitätsschlüssel zusammen
  - 1:N, N:1 - Schlüssel der N-Seite wird Schlüssel
  - 1:1 - einer der beiden Entitätsschlüssel kann gewählt werden

Beispiele



$$\Downarrow$$

besucht(<u>MatrNr, Nr</u>)



$$\Downarrow$$

hält(<u>Nr</u>, PersNr)

Schemavereinfachung

- Nach dem zweiten Schritt können Relationen mit dem gleichen Schlüssel existieren
- Diese Relationen werden in einer einzigen Relation zusammengefaßt
- Dies passiert bei 1:N, N:1 und 1:1 Beziehungen, da die Beziehungsrelation den gleichen Schlüssel wie eine der Entitätsrelationen hat

Beispiel

- `Vorlesung` und `hält` haben den gleichen Schlüssel:

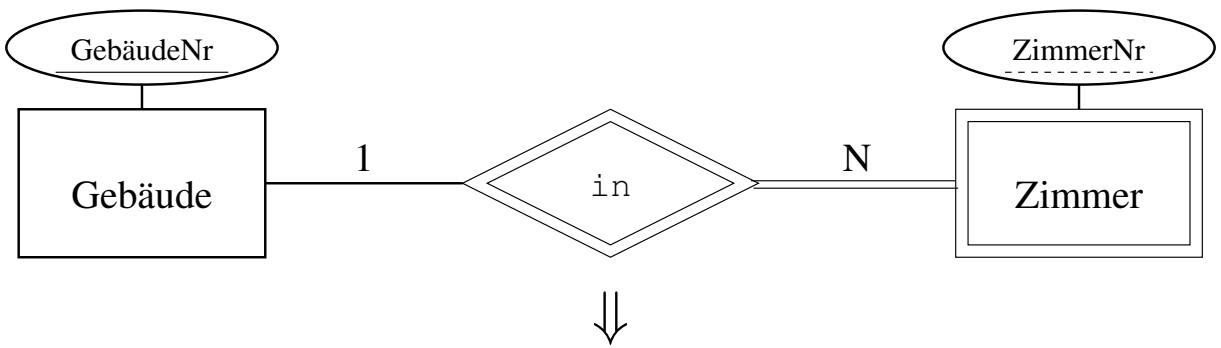
$$\text{Vorlesung}(\underline{\text{Nr}}, \text{Titel}, \text{Credits})$$
$$\text{hält}(\underline{\text{Nr}}, \text{PersNr})$$
$$\Downarrow$$
$$\text{Vorlesung}(\underline{\text{Nr}}, \text{Titel}, \text{Credits}, \text{PersNr})$$

- `PersNr` wird *Fremdschlüssel* von `Vorlesung` genannt, da er den Schlüssel in `Professor` referenziert (und auch die gleichen Werte enthält)
- Häufig Umbenennung, um Referenzen klarzumachen

$$\text{Vorlesung}(\underline{\text{Nr}}, \text{Titel}, \text{Credits}, \text{ProfPersNr})$$

Schwache Entitäten

- Schwache Entitäten werden im Prinzip wie 1:N Beziehungen umgesetzt
- Einziger Unterschied: die schwache Entität hat einen zusammengesetzten Schlüssel

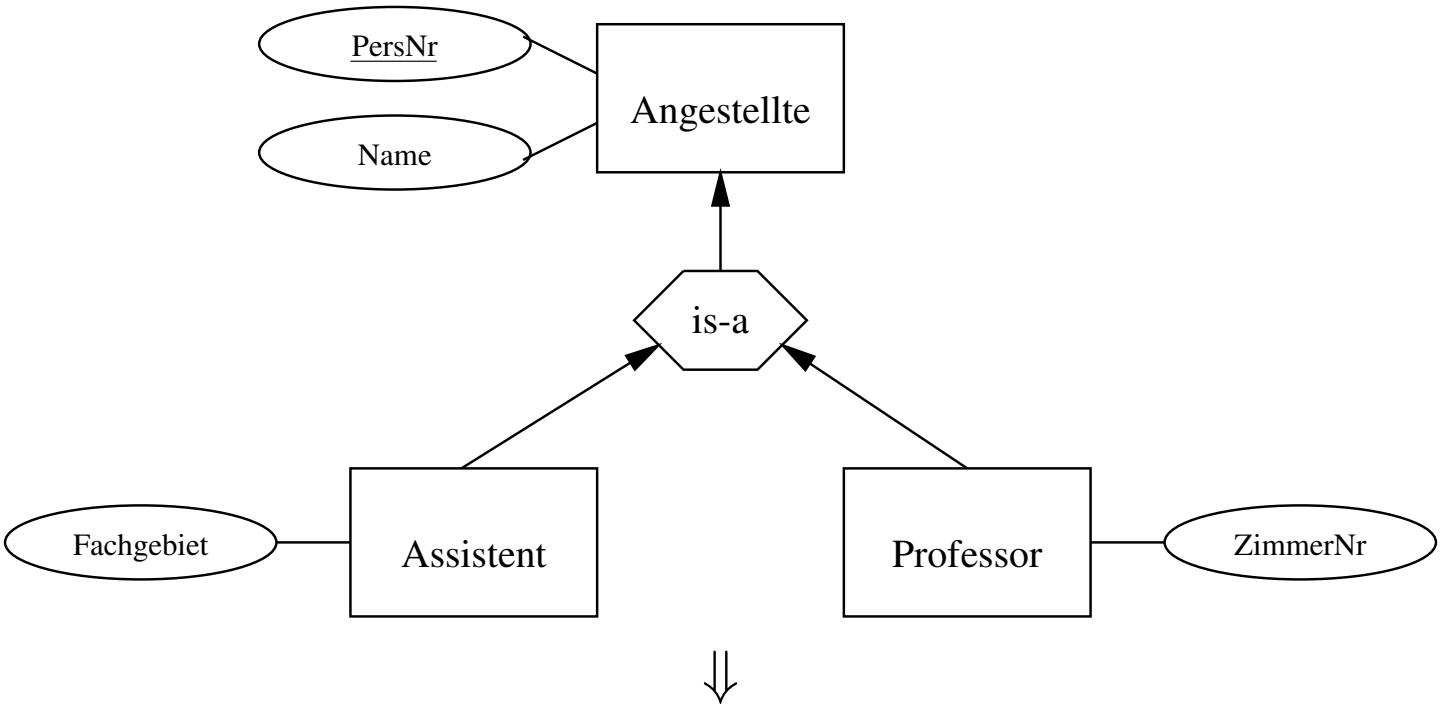


$\Downarrow$

Gebäude(<u>GebäudeNr</u>, ...) Zimmer(<u>GebäudeNr, ZimmerNr</u>, ...)

Generalisierungen

- Generalisierungen können auf drei verschiedene Weisen umgesetzt werden (es gibt keine direkte äquivalente Übersetzung ins relationale Modell



⇓

(1) Nur Obertyp, (2) nur Untertypen, (3) alle Typen

Generalisierungen(2)

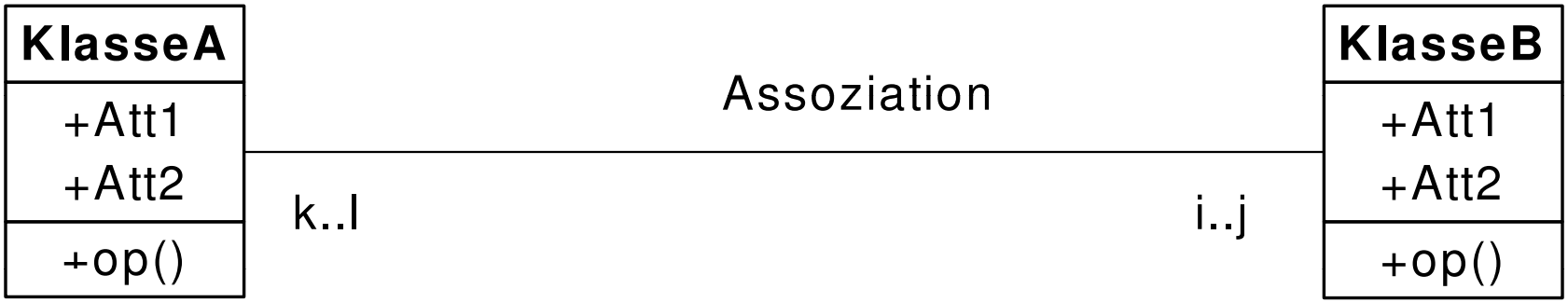
- Nur Obertyp:
  Angestellte(PersNr, Name, Fachgebiet, ZimmerNr)
  Problem: NULL-Werte
- Nur Untertypen:
  Assistent(PersNr, Name, Fachgebiet)
  Professor(PersNr, Name, ZimmerNr)
  Problem: Angestellte die weder Assistenten noch Professoren sind
- Alle Typen:
  Angestellte(PersNr, Name)
  Assistant(PersNr, Fachgebiet)
  Professor(PersNr, ZimmerNr)
  Problem: Information ist verteilt auf mehrere Relationen

Datenmodellierung mit 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)

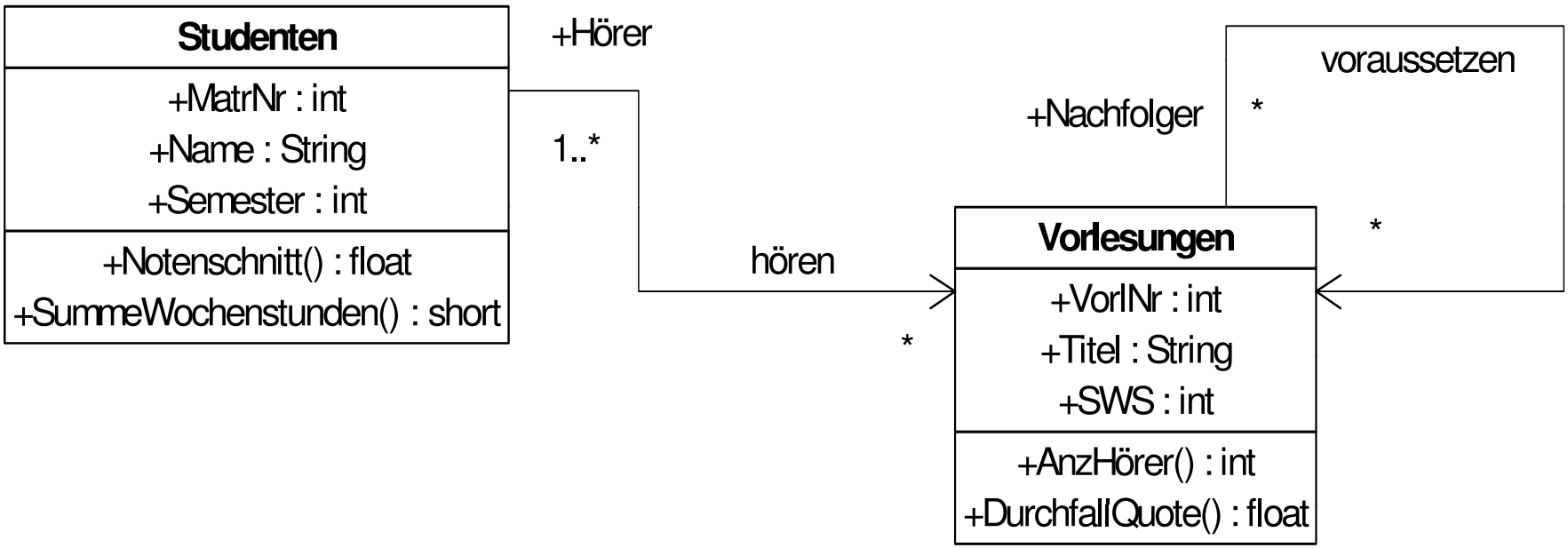
  modelliert werden
- Assoziationen zwischen Klassen entsprechen Beziehungstypen
  - Generalisierungshierarchien
  - Aggregation

Multiplizitäten

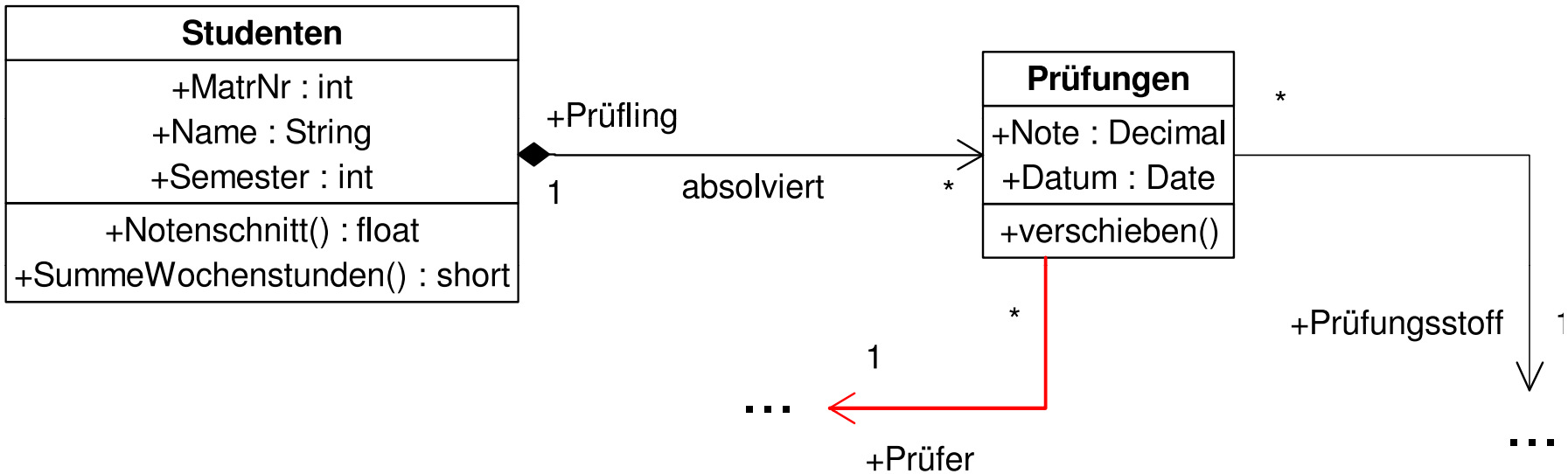


- Jedes Element von Klasse A steht mit mindestens i Elementen der KlasseB in Beziehung
- ... und mit maximal j vielen KlasseB-Elementen
- Analoges gilt für das Intervall k..l
- Multiplizitätsangabe ist analog zur Funktionalitätsangabe im ER-Modell
  - Nicht zur (min,max)-Angabe: **Vorsicht!**

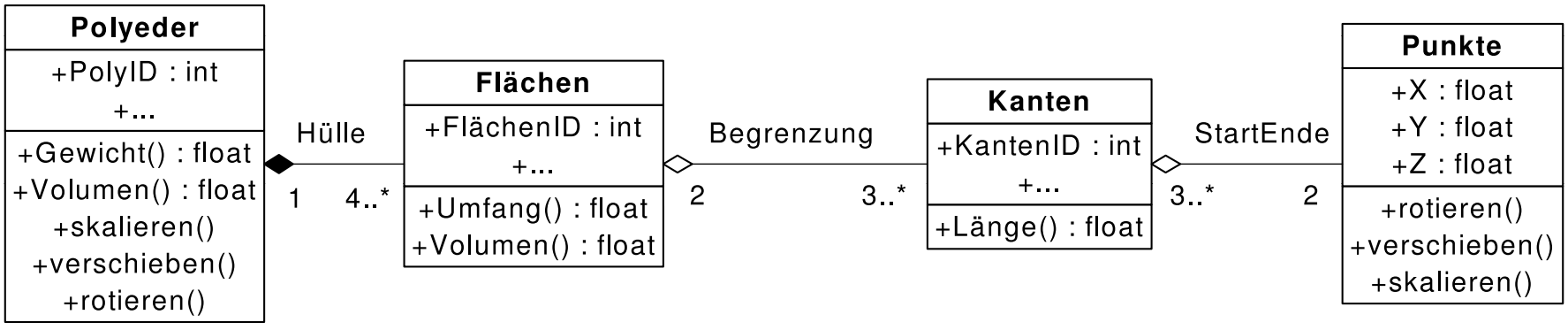
Klassen und Assoziationen

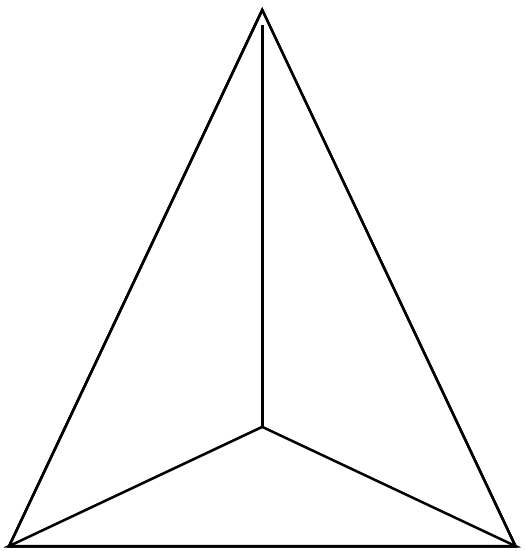


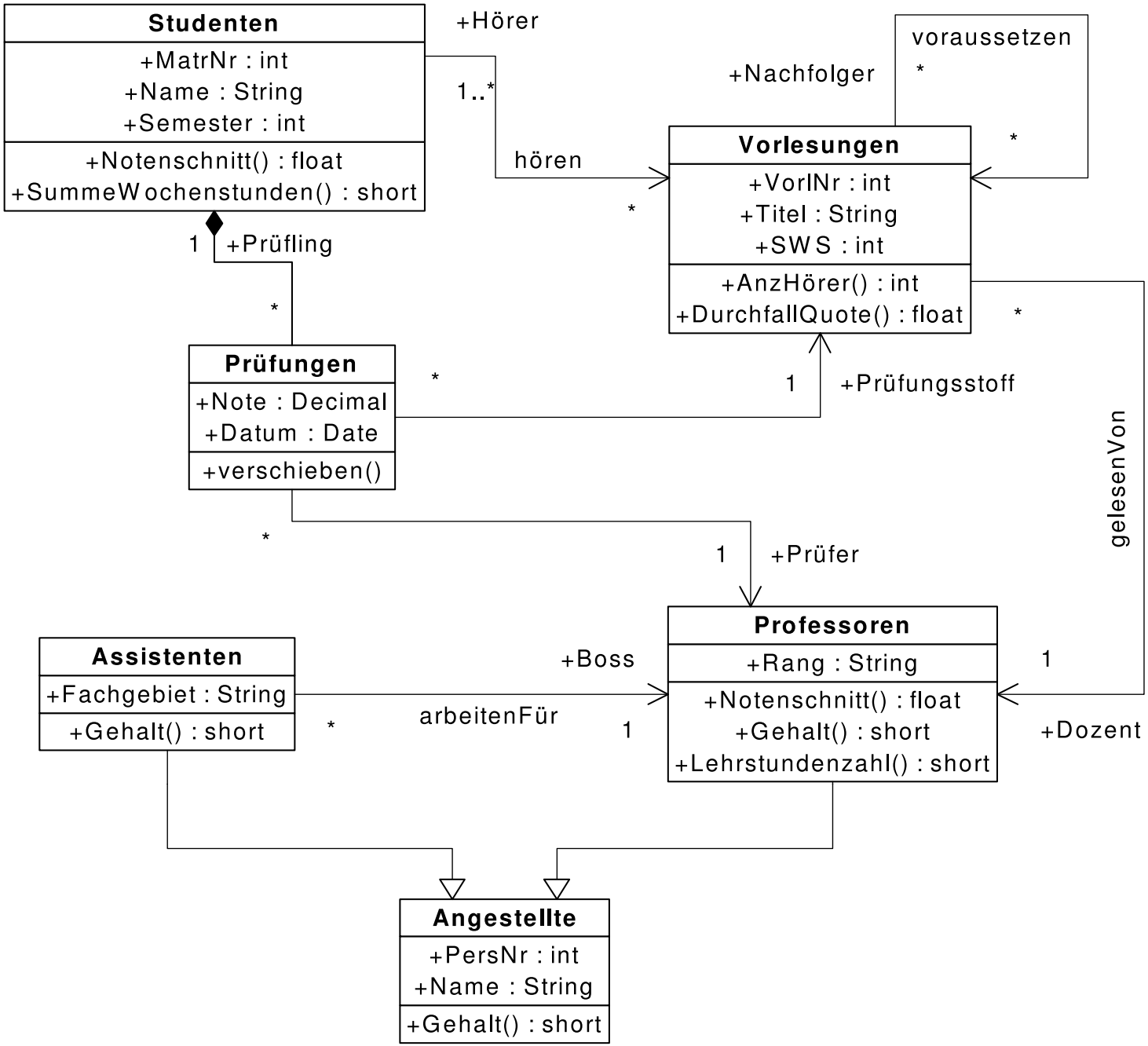
Aggregation



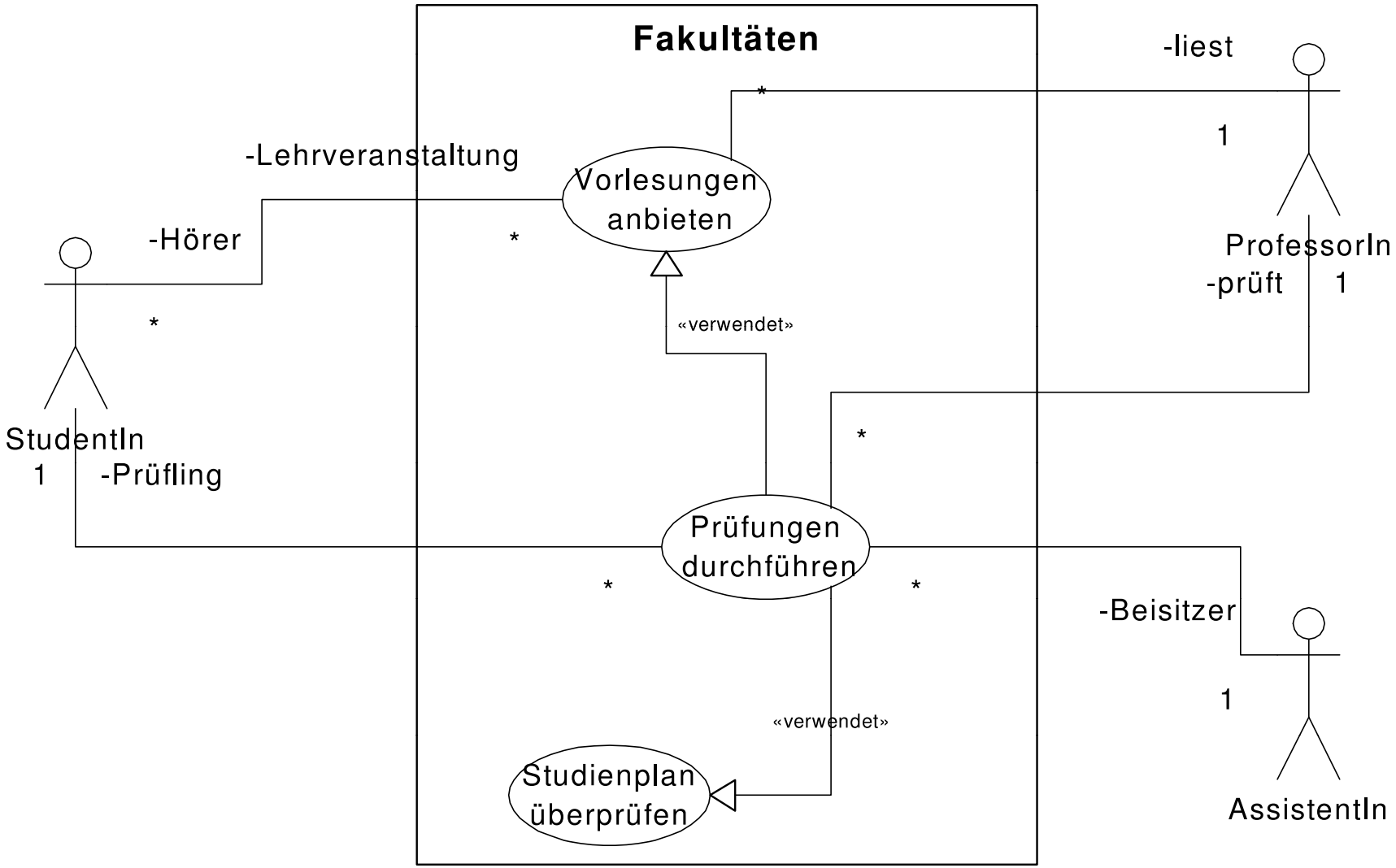
Gegrenzungsflächenmodellierung in UML



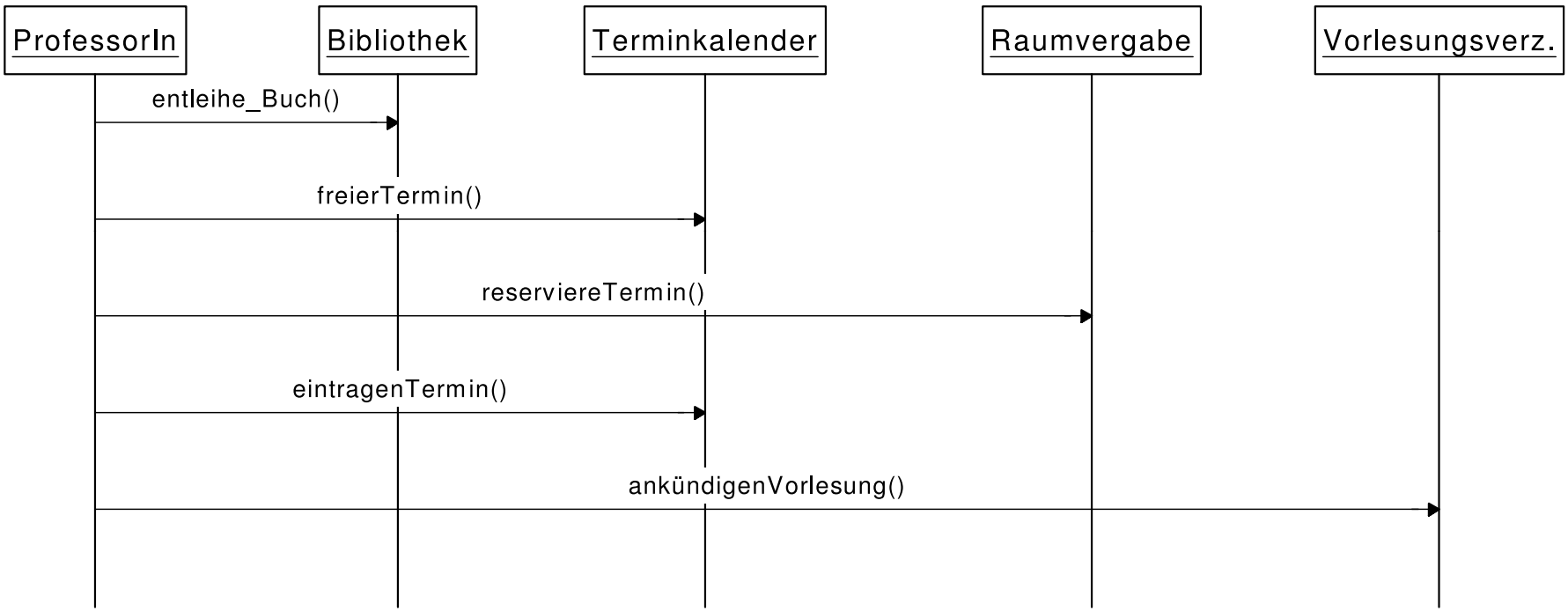



Studenten
+MatrNr : int
+Name : String
+Semester : int
+Notenschnitt() : float
+SummeWochenstunden() : short
+Hörer
1..*
hören
*
Vorlesungen
+VorlNr : int
+Titel : String
+SWS : int
+AnzHörer() : int
+DurchfallQuote() : float
+Nachfolger
voraussetzen
*
*
1
+Prüfling
*
Prüfungen
+Note : Decimal
+Datum : 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

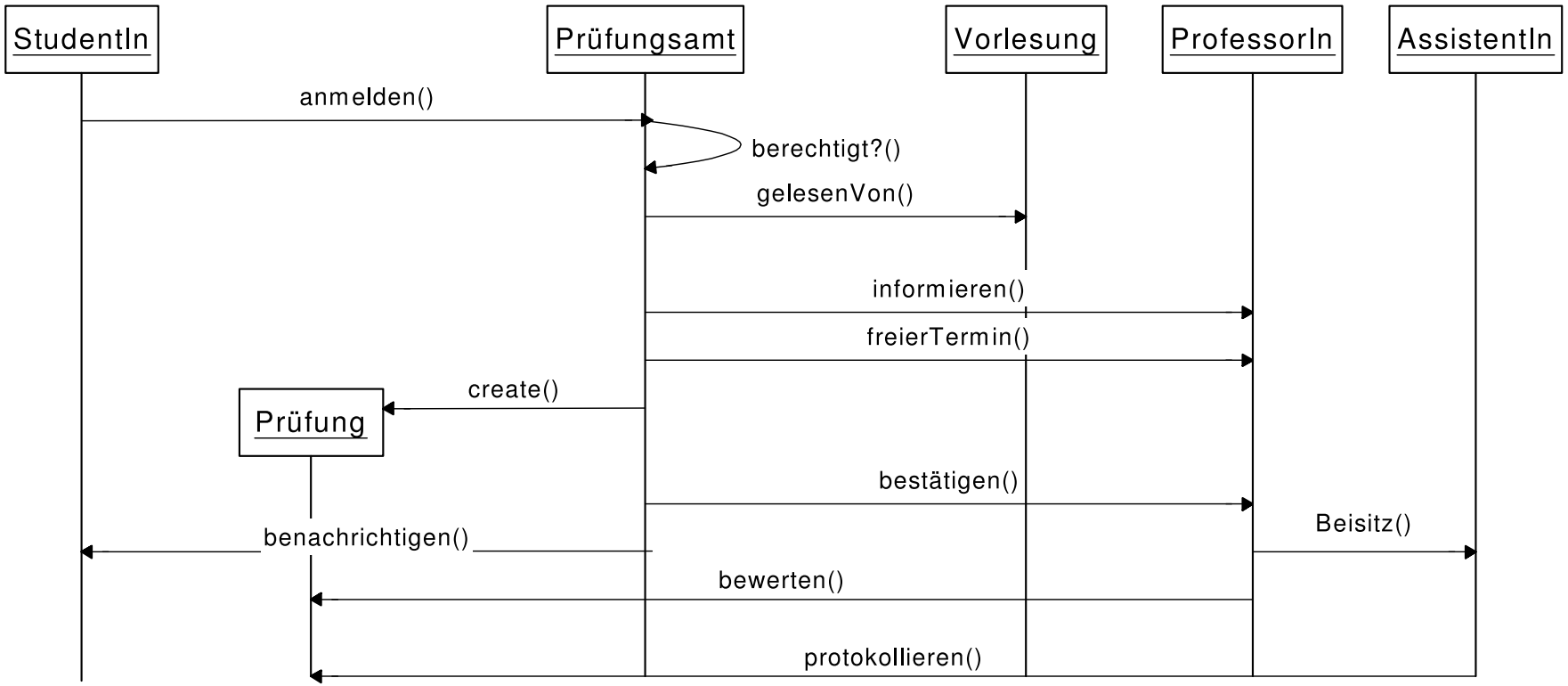
Anwendungsfälle (use cases)



Interaktions-Diagramm



Interaktions-Diagramm *Prüfungsdurchführung*



Zusammenfassung

- ER und UML sind weit verbreitete Arten der konzeptuellen Modellierung
- Die Konzepte sind nicht allzu schwer zu begreifen, d.h. der Sachverhalt kann auch mit Informatiklaien (die den Fachbereich kennen) diskutiert werden
- Entwurf ist auch immer etwas subjektives, es gibt viele verschiedene Wege einen Sachverhalt zu modellieren (sich für den passendsten zu entscheiden ist eine Sache der Erfahrung)
- Die Qualität eines konzeptuellen Schemas kann auf der logischen Ebene noch einmal geprüft werden
- Das konzeptuelle Modell wird schließlich ins relationale Modell überführt