Einführung, Entity-Relationship Modell

9. DATENBANKSYSTEME: DAS ENTITY RELATIONSHIP MODELL

Literatur, Quellen

- Literatur: Kemper, Eickler: Datenbanksysteme: Eine Einführung. Oldenbourg Verlag, 9. Auflage, 2013.
- Quellen: Nachfolgende Folien wurden von Prof. Donald Kossmann & Martin Kaufmann freundlicherweise zur Verfügung gestellt.
- Weitere Quelle: Folien zu *Datenbanksysteme: Eine Einführung*, Lehrstuhl III Datenbansysteme, Prof. Kemper, TU München

Ziele

- Nutzen von Datenbanksystemen verstehen, Modellierungskenntnisse
  - ER Modell (Modellierung der Weltsicht)
  - Relationales Modell (Modellierung für die DB)
- Datenbanksystem anwenden
  - SQL
- Datenbanksystem von einer Programmiersprache und Umgebung heraus ansprechen
  - JDBC (SQL / Java)

Datenbankverwaltungssysteme

Ein Datenbankverwaltungssystem (DBMS) ist ein Werkzeug zur Erstellung und Ausführung datenintensiver Anwendungen

- grosse Datenbanken
- grosse Datenströme

Typische Anwendungen

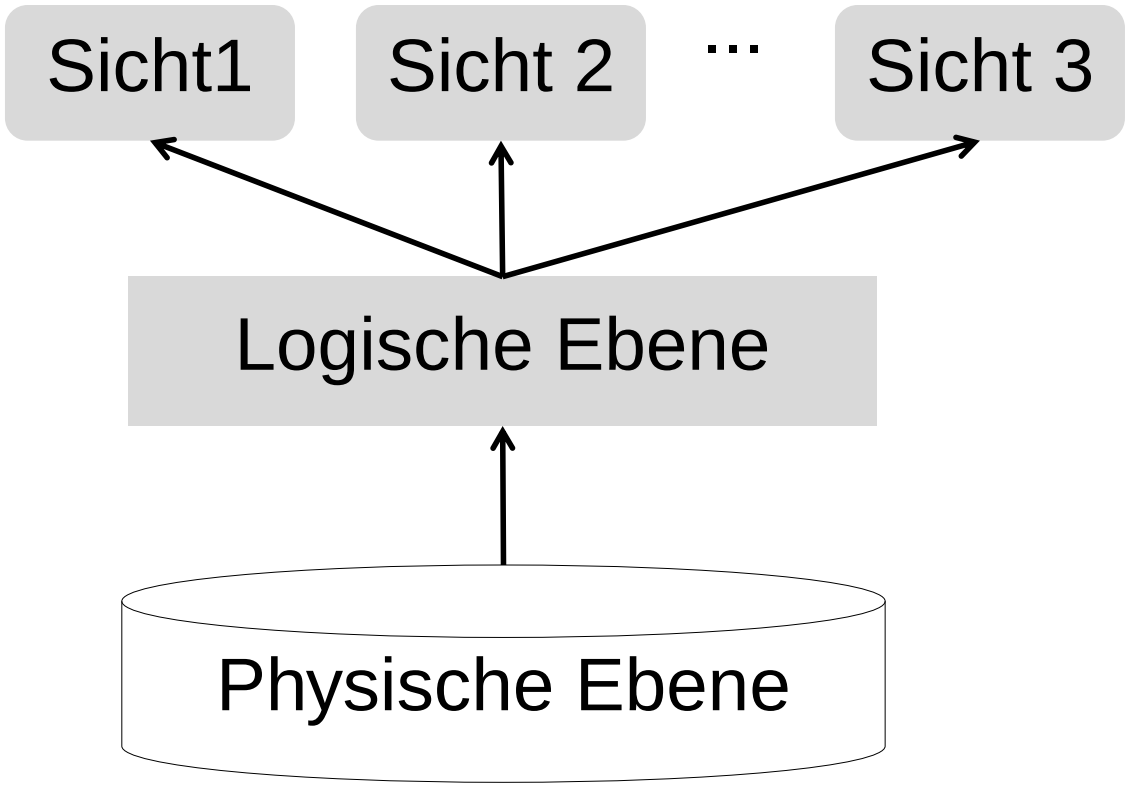
- Bank
  z.B. Konten / „Geldtransfer“
- Bibliothek
  z.B. Bücher / „Ausleihen“
- Facebook, Twitter, …
  z.B. Freunde, „Sende Tweet“
- Geoinformationssysteme
  z.B. Topographische Information, "Erzeuge Karte"

Wozu Datenbanksysteme?

- Vermeide Redundanz und Inkonsistenz
- Deklarativer Zugriff auf die Daten und Unabhängigkeit von der Implementation (physische Datenunabhängigkeit)
- Synchronisiere gleichzeitigen Datenzugriff
- Sicherheit, Vertraulichkeit

- Minimiere Kosten und Aufwand
  Ähnliche Funktionalität selbst zu implementieren würde Jahre in Anspruch nehmen

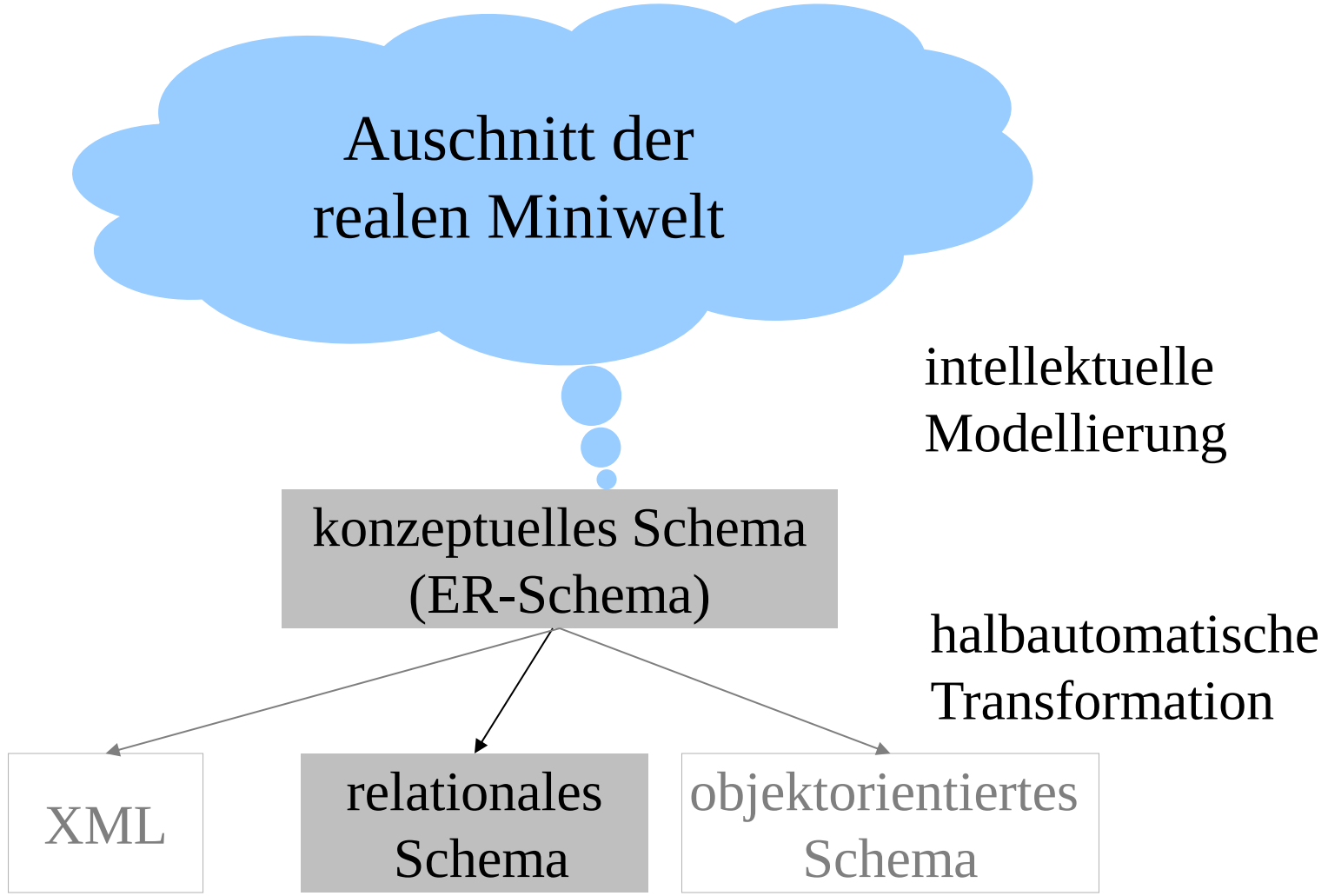
Abstraktionsebenen eines Datenbanksystems



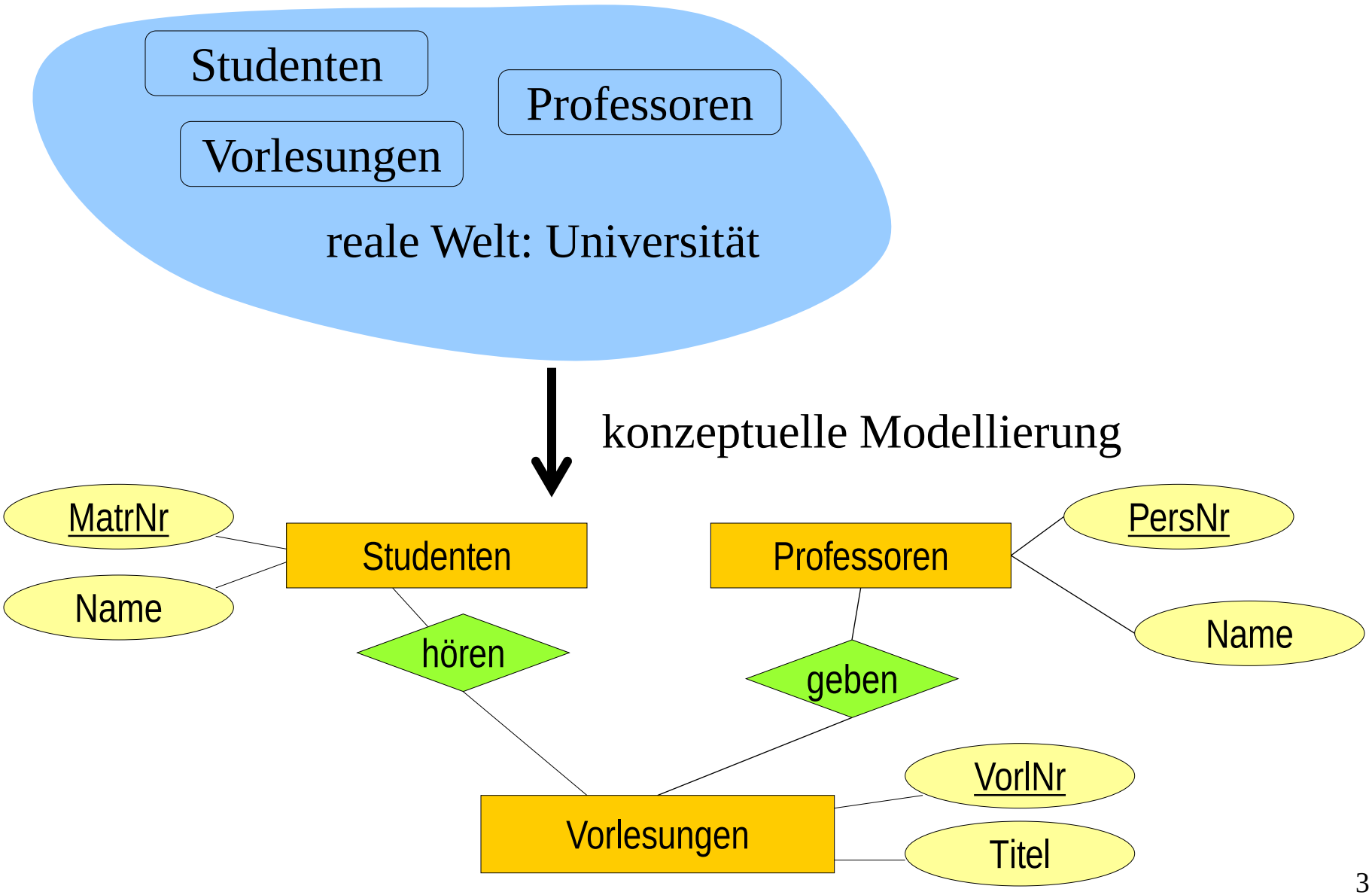
Datenunabhängigkeit:

- physische Unabhängigkeit
- logische Datenunabhängigkeit

Datenmodellierung



Beispiel: konzeptuelle Modellierung



Logisches Modell: relationales Datenmodell

Studenten	
Legi	Name
26120	Fichte
25403	Jonas
...	...

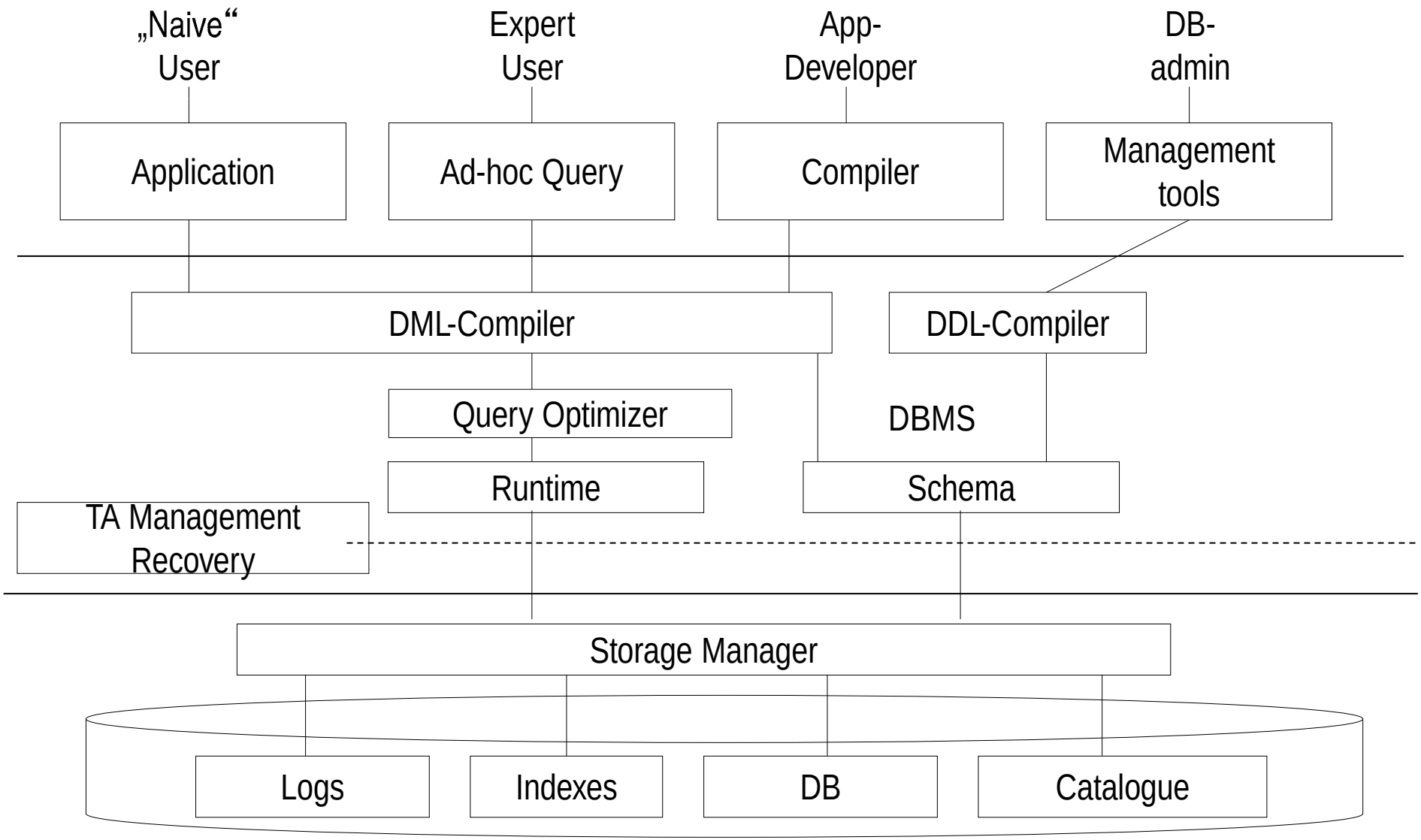
hören	
Legi	VorlNr
25403	5022
26120	5001
...	...

Vorlesungen	
VorlNr	Titel
5001	Grundzüge
5022	Glaube und Wissen
...	...

```
select Name
from   Studenten, hören, Vorlesungen
where  Studenten.Legi= hören.Legi and
       hören.VorlNr= Vorlesungen.VorlNr and
       Vorlesungen.Titel = `Grundzüge´;
```

```
update Vorlesungen
set    Title = `Grundzüge der Logik´
where  VorlNr = 5001;
```

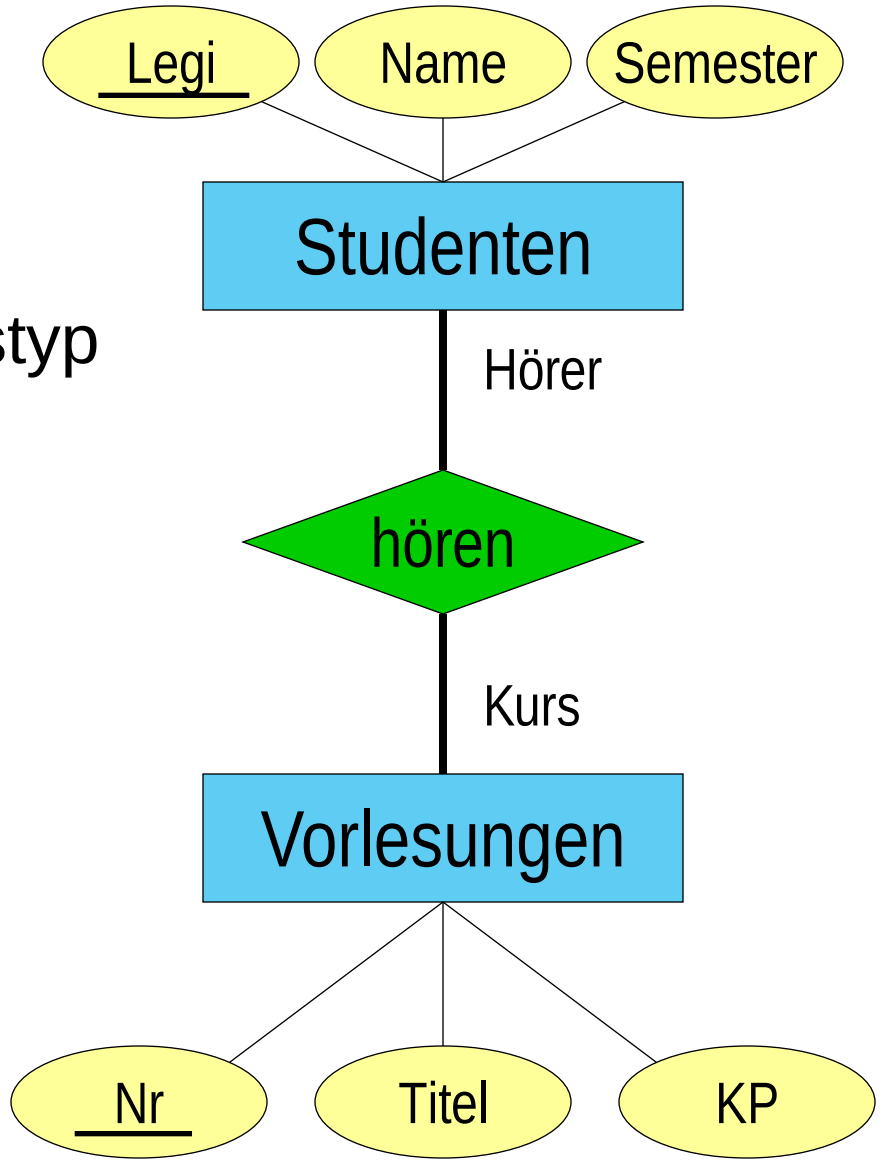
Komponenten eines DBMS*



External Storage

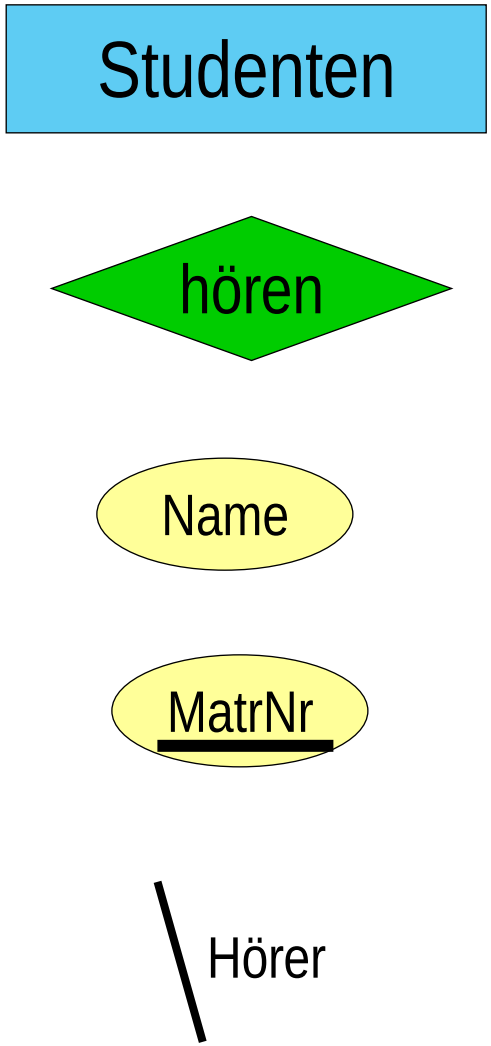
Entity/Relationship (ER) Modell

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

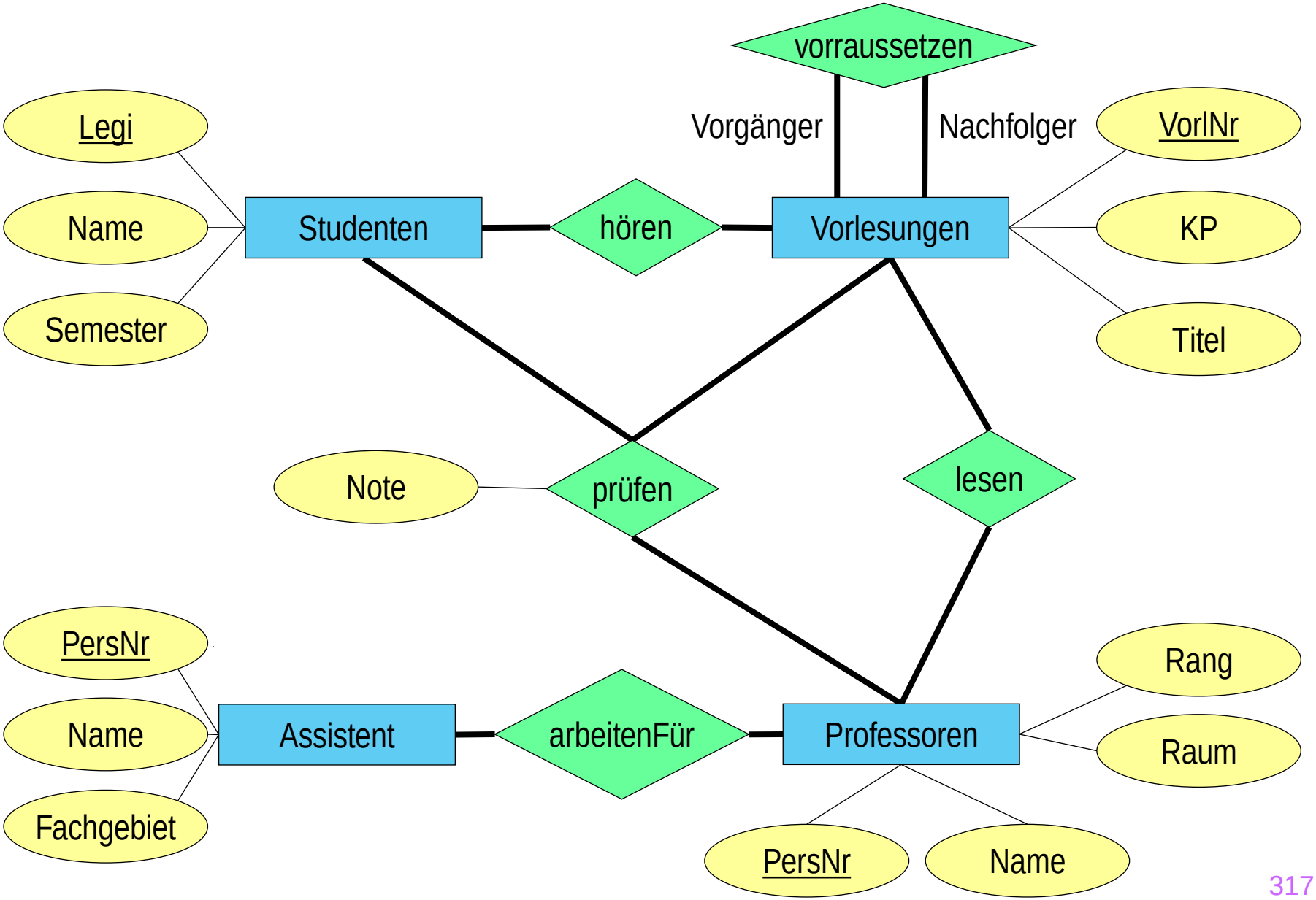


Entity/Relationship (ER) Modell

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



Modell einer Universität



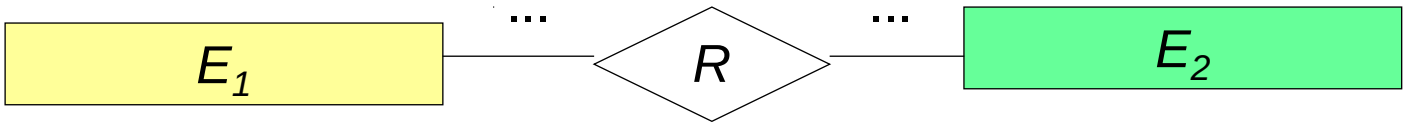
... in natürlicher Sprache

- Studenten haben LegiNr, Name und Semester. Die LegiNr identifziert einen Studenten eindeutig.
- Vorlesungen haben eine VorlNr, Kreditpunkte und einen Titel. Die VorlNr identifiziert eine Vorlesung eindeutig.
- Porfessoren haben PersNr, Name, Rang und Raum. Die PersNr identifiziert einen Professor eindeutig.
- Assistenten haben PersNr, Name und Fachgebiet. Die PersNr. identifiziert einen Assistenten eindeutig.
- Studenten hören Vorlesungen
- Vorlesungen können Voraussetzung für andere Vorlesungen sein.
- Professoren lesen Vorlesungen.
- Assistenten arbeiten für Professoren
- Studenten werden von Professoren über Vorlesungen geprüft. Studenten erhalten Noten als Teil dieser Prüfungen.
- Ist das die einzig mögliche Interpretation?

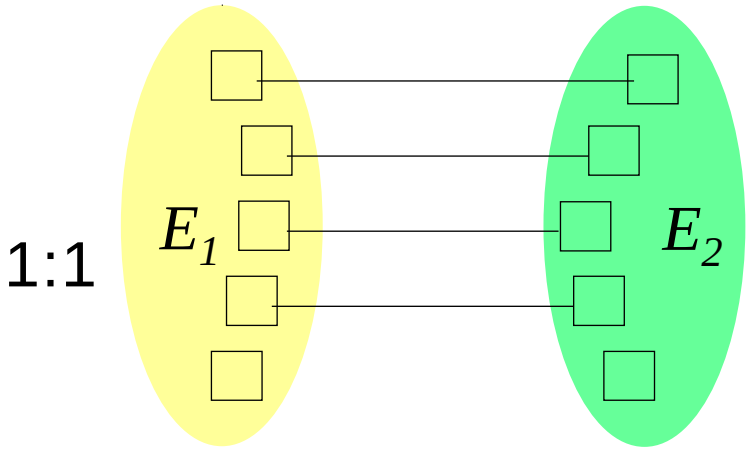
Warum ER?

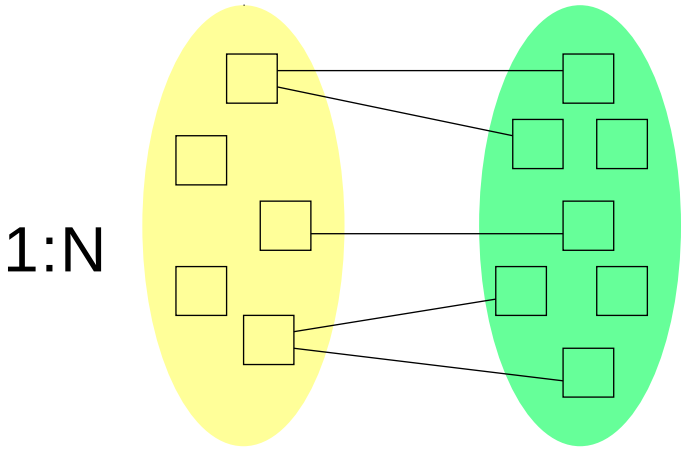
- Vorteile
  - ER Diagramme sind einfach zu erstellen und editieren
  - ER Diagramme sind aufgrund der grafischen Darstellung einfach zu verstehen (vom Laien)
  - ER Diagramme beschreiben alle Informationsanforderungen
- Allgemeines
  - Viele Tools verfügbar
  - Kontroverse, ob ER/UML in der Praxis von Nutzen ist
  - Keine Kontroverse, dass jeder ER/UML lernen sollte

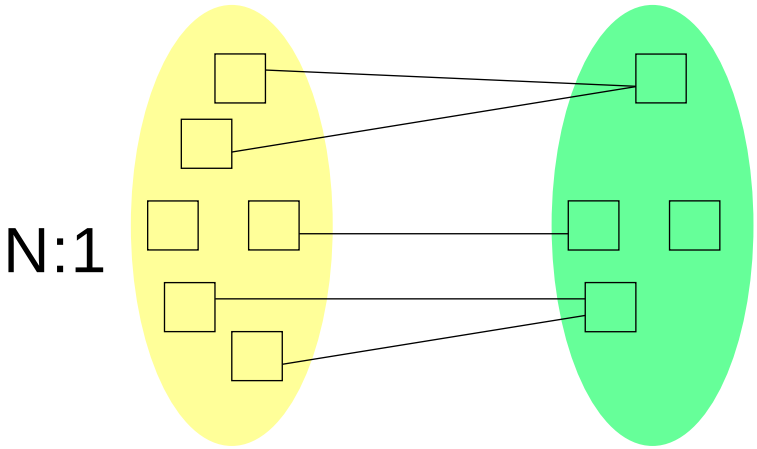
Funktionalitätsangaben

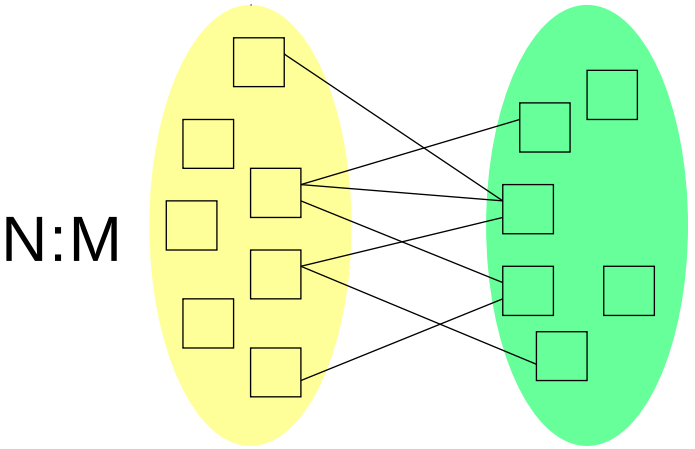


$$R \subseteq E_1 \times E_2$$

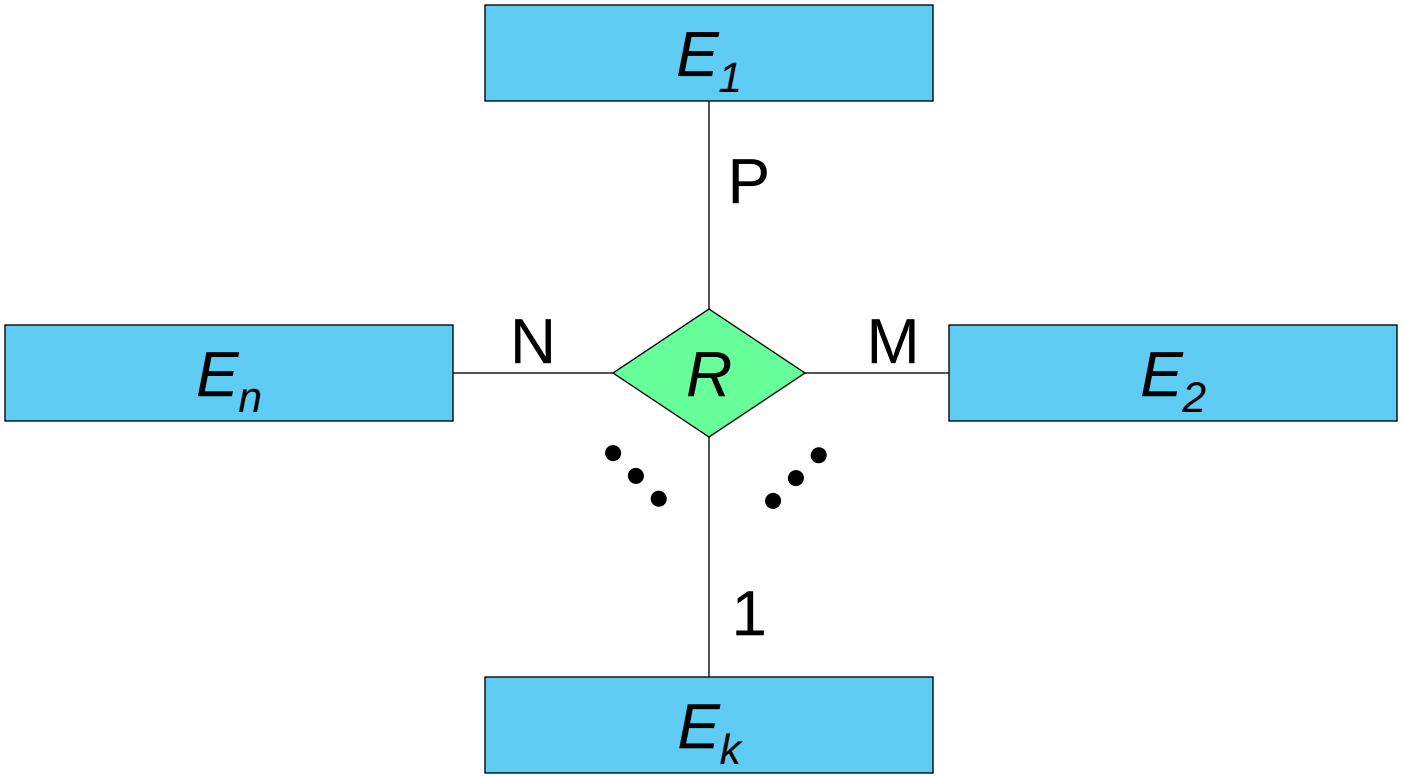






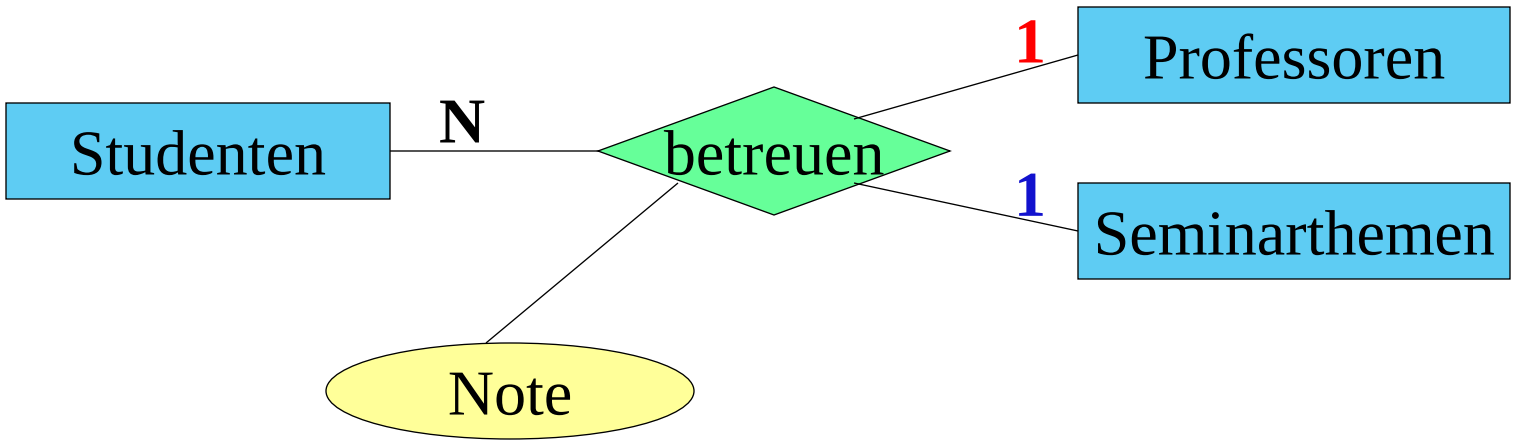


Funktionalitäten bei n-stelligen Beziehungen



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

Beispiel: Seminar



betreuen: Professoren x Studenten → Seminarthemen

betreuen: Seminarthemen x Studenten → Professoren

Konsistenzbedingungen des Seminar

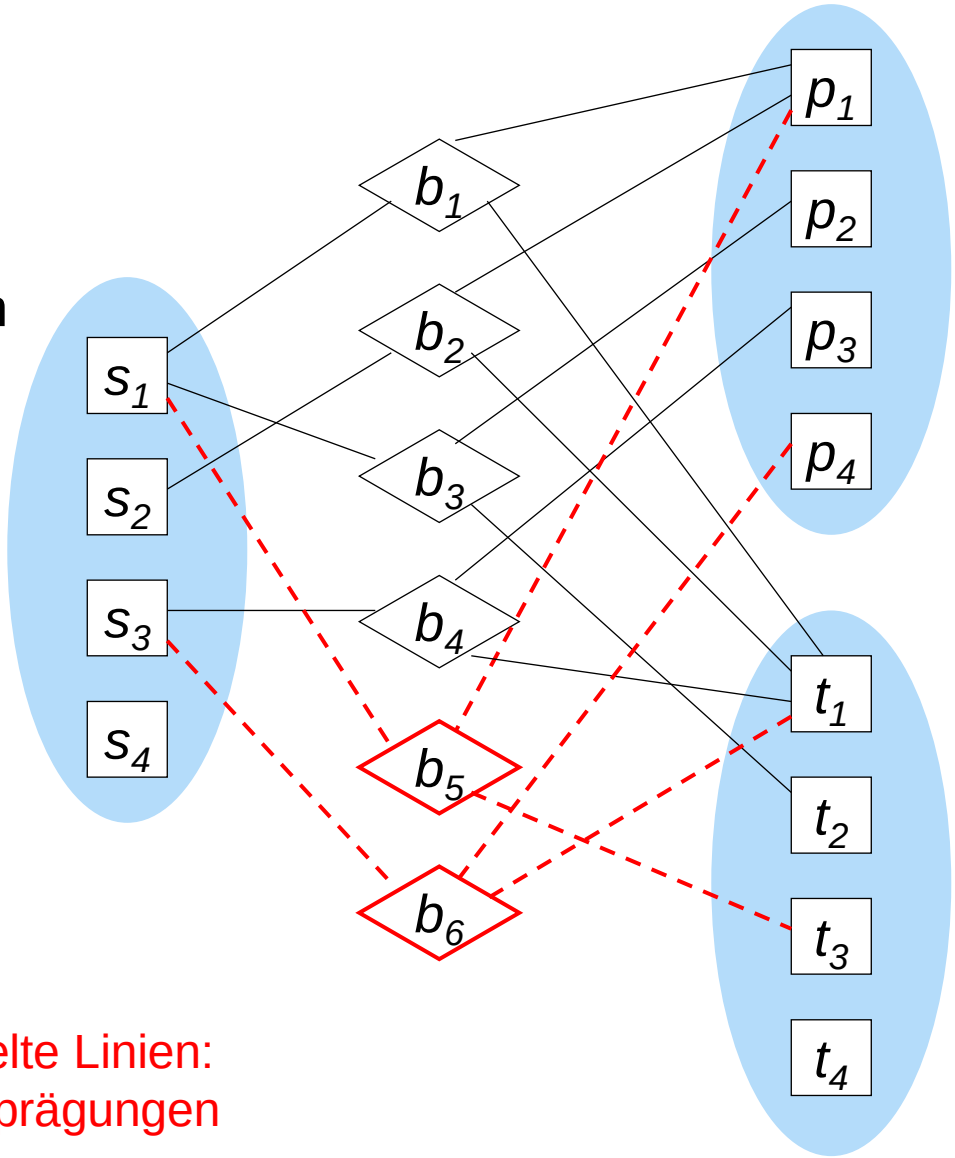
Einschränkungen

1. Studenten dürfen bei einem Professor nur ein Seminarthema bearbeiten
2. Studenten können dasselbe Seminarthema nur einmal bearbeiten

Möglichkeiten

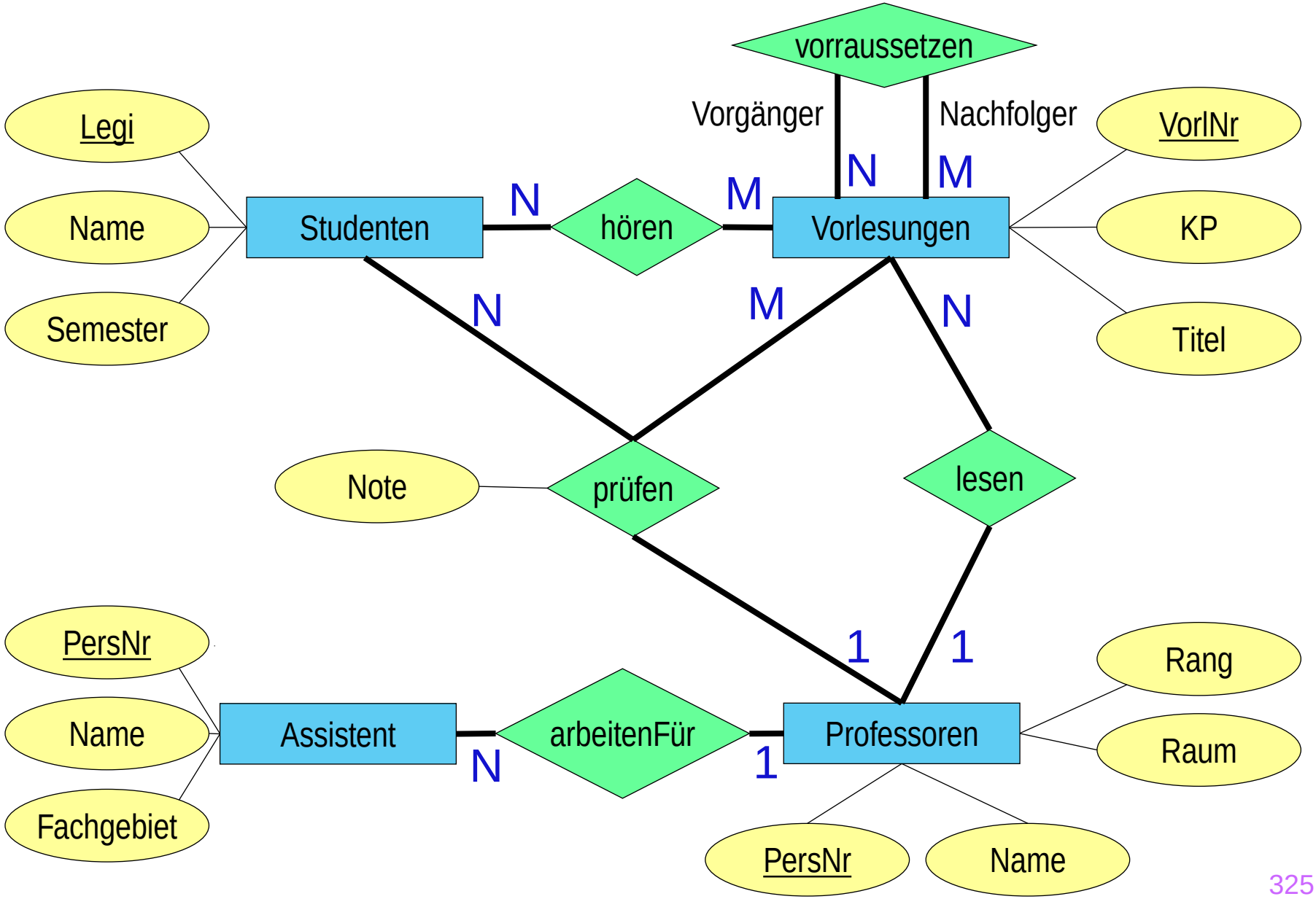
1. Professoren können das Seminarthema für andere Studenten wiederverwenden
2. Dasselbe Thema kann von verschiedenen Professoren verwendet werden

Konsistenzbedingungen: graphisch



rot gestrichelte Linien:
illegale Ausprägungen

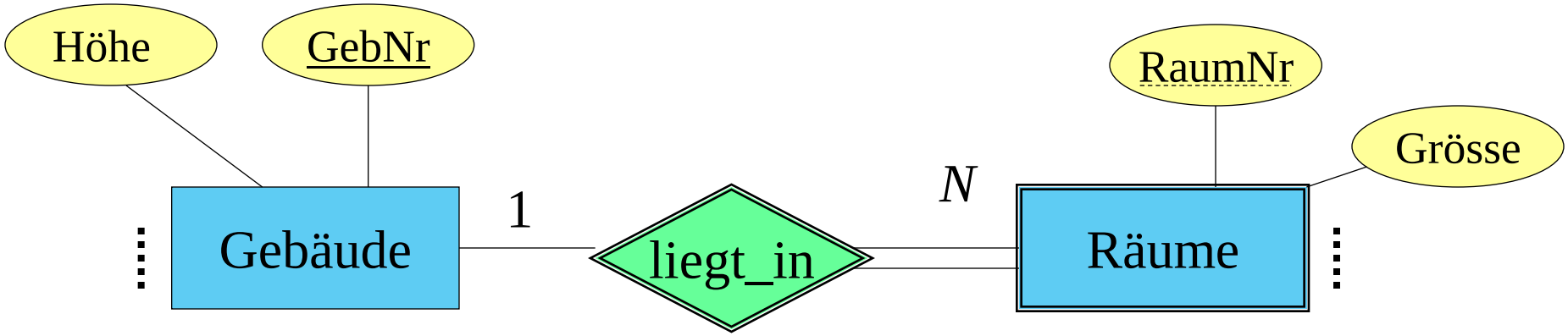
Universität mit Funktionalitäten



Daumenregeln

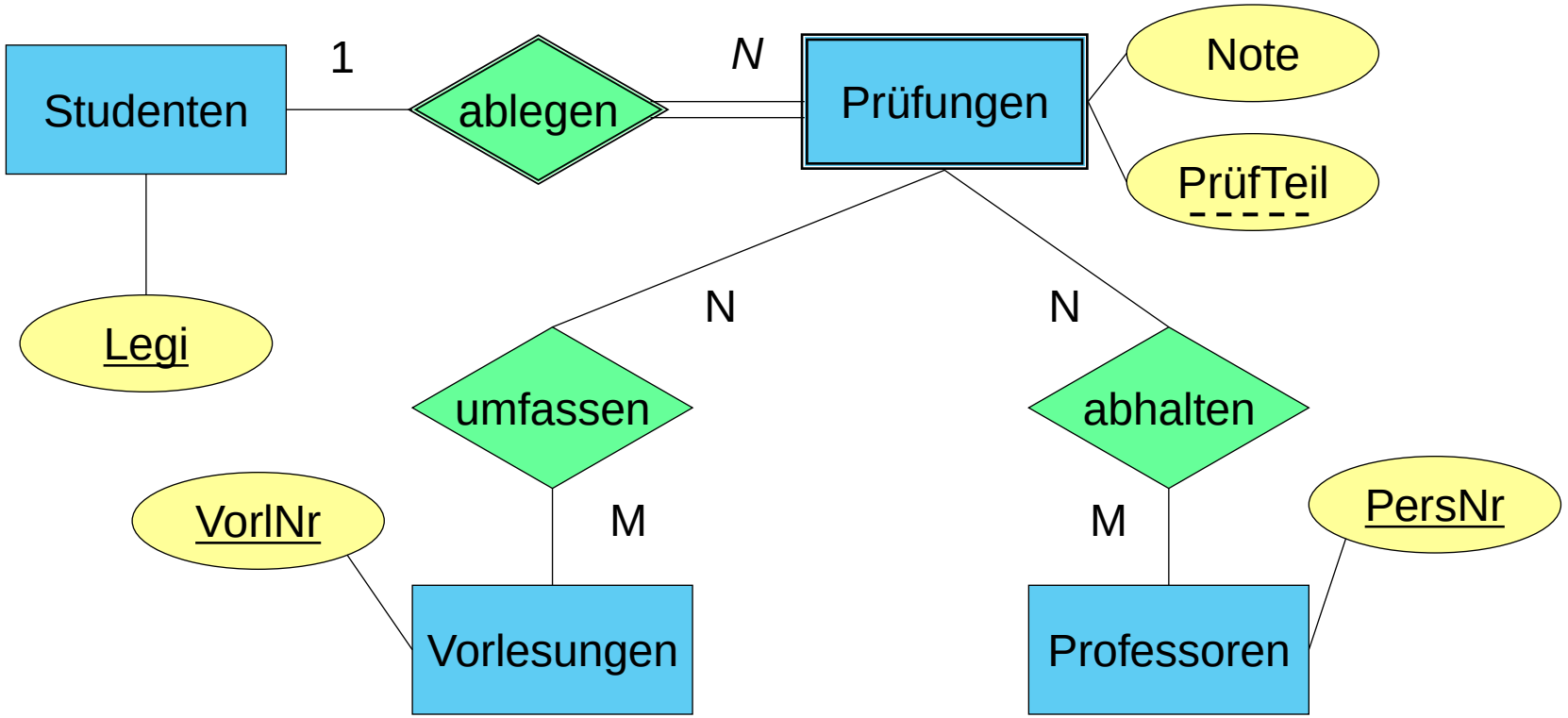
- Wann Attribut, wann Entität?
  - Entität, wenn das Konzept mehr als eine Beziehung hat
  - Attribut, wenn das Konzept nur eine 1:1 Beziehung hat
- Partitionierung von ER-Modellen
  - Realistische Modelle sind grösser als eine Seite
  - Nach Bereichen / Organisationseinheiten partitionieren
  - Kein gutes automatisches Graphenpartitionierungstool bekannt
- Tipps
  - Keine Redundanz modellieren, keine vermeintliche Leistungsverbesserung anbringen
  - Je weniger Entitäten desto besser
  - konzis, vollständig, nachvollziehbar, korrekt

Schwache (existenzabhängige) Entities



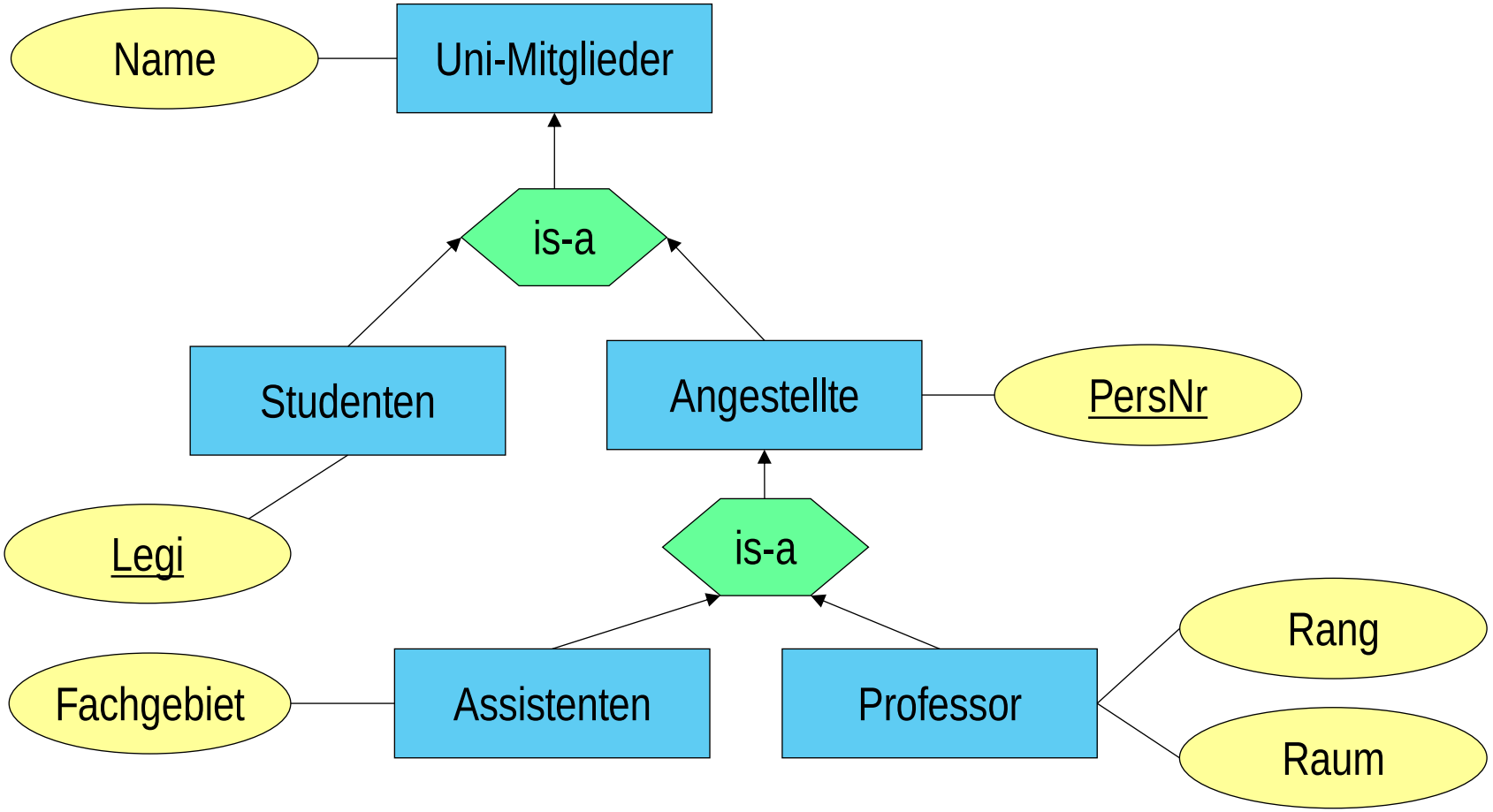
- Beziehung ist immer 1:N (oder 1:1)
- Existenz von Raum hängt ab von der Existenz des zugehörigen Gebäudes, daher kann es keine N:M Beziehung sein
- Raumnummer nur eindeutig im Gebäude
- Schlüssel eines Raumes: GebNr und RaumNr

Prüfung als schwacher Entitytyp



- Unterschied zur Modellierung mit Beziehung "prüfen"?

Generalisierung



Generalisierung

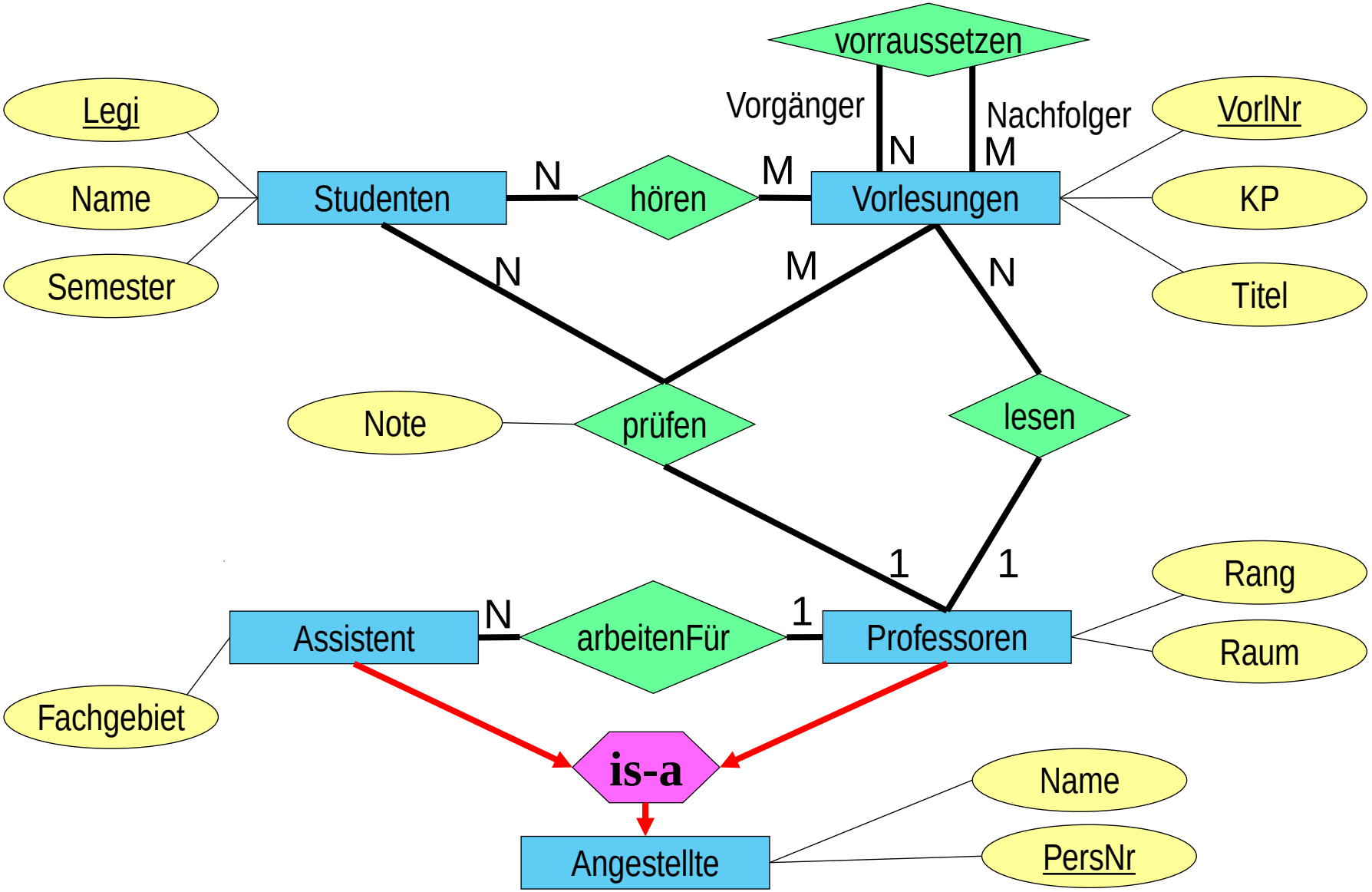
Möglich:

- Angestellte, welche weder Assistent noch Professoren sind
- Angestellter, welcher auch Assistent ist
- Angestellter, welcher auch Professor ist
- Angestellter, welcher auch Professor und Assistent ist

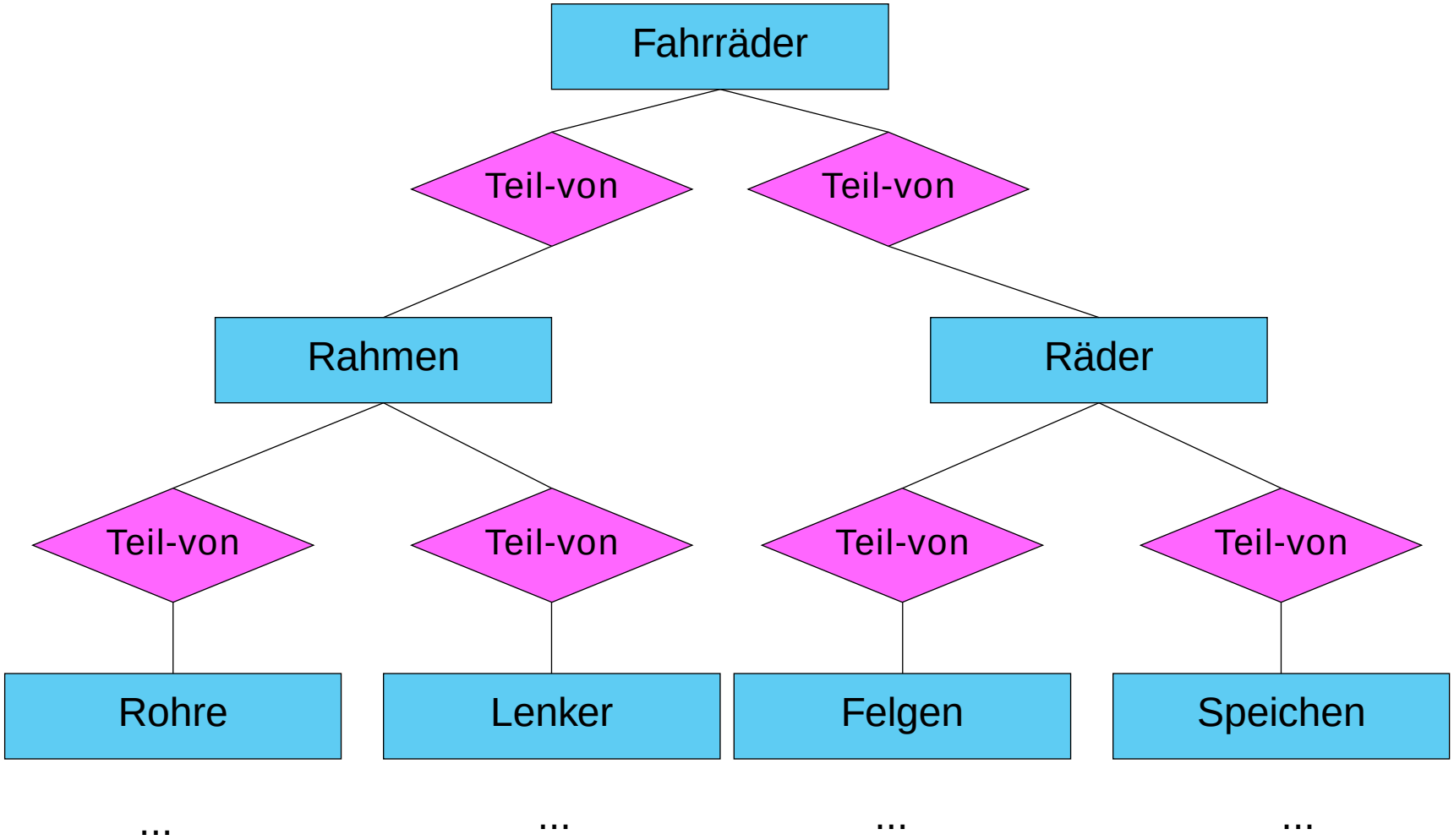
Im ER Modell kann dies nicht explizit verhindert werden

- muss separat beschrieben werden.

Universität mit Generalisierung

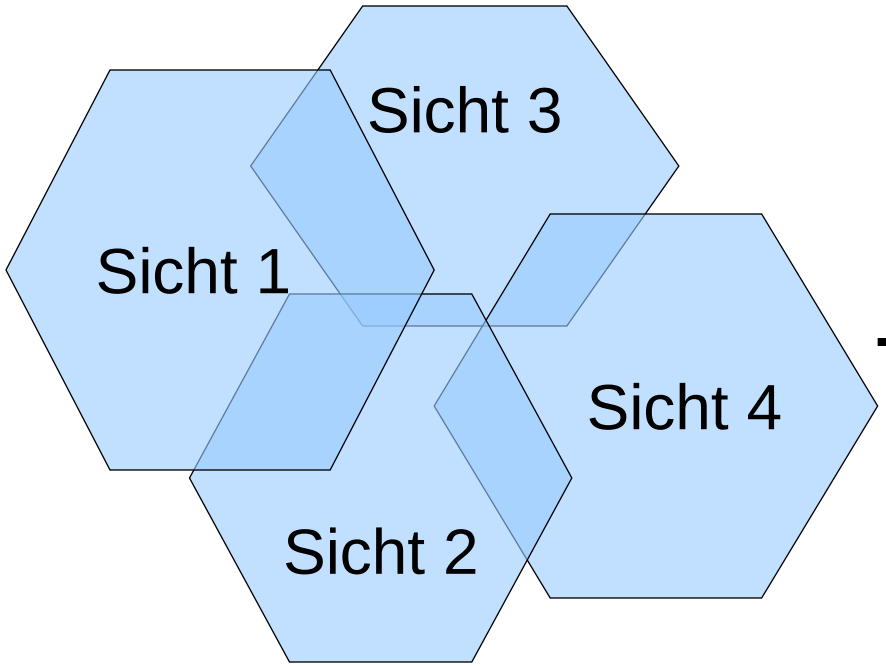


Aggregation



Konsolidierung

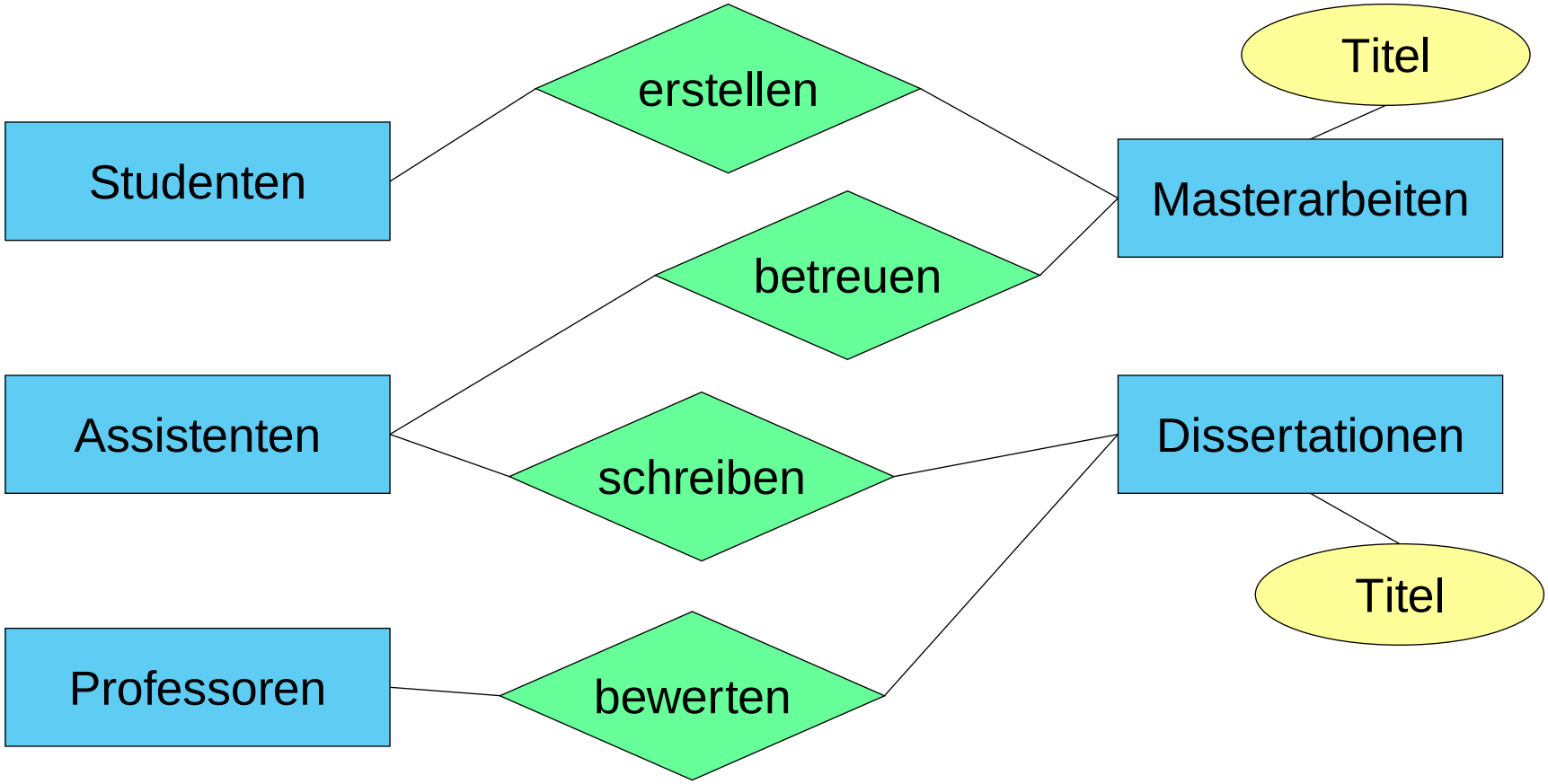
- Verschiedene Anwender haben verschiedene Sichten



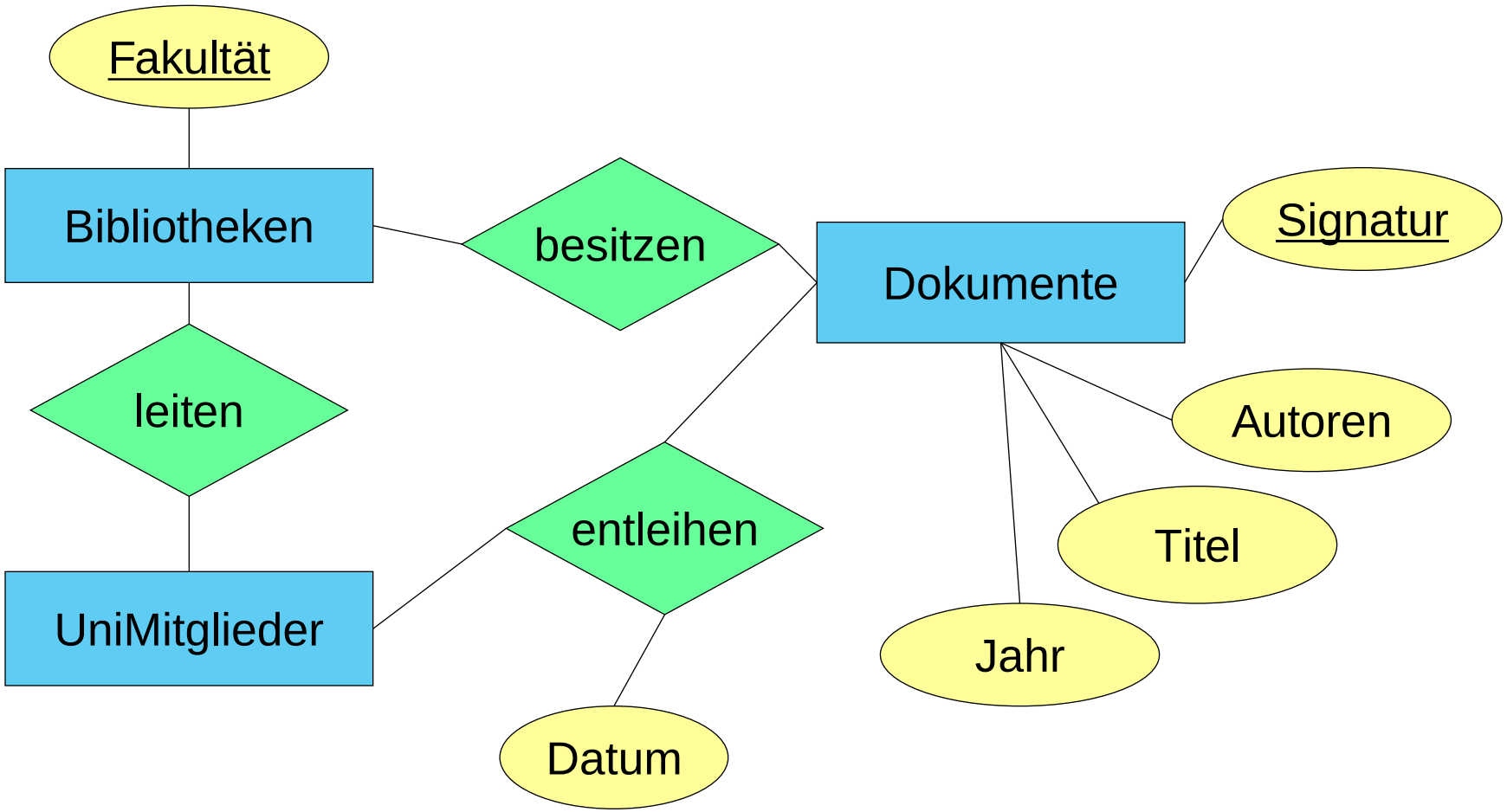
Konsoli-
dierung

Globales Schema
- redundanzfrei
- widerspruchsfrei
- Synonyme bereinigt
- Homonyme bereinigt

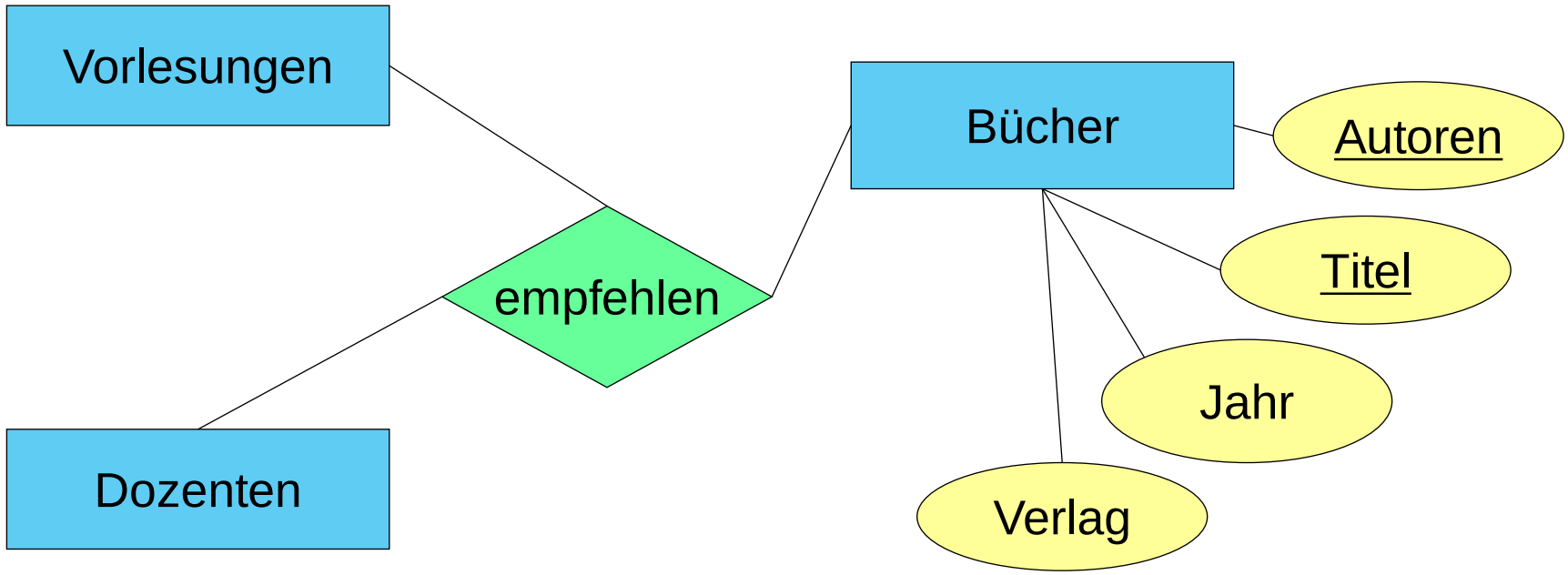
Beispiel Uni-DB, Sicht 1 der Professoren



Beispiel Uni-DB, Sicht 2 der Bibliotheksverwaltung

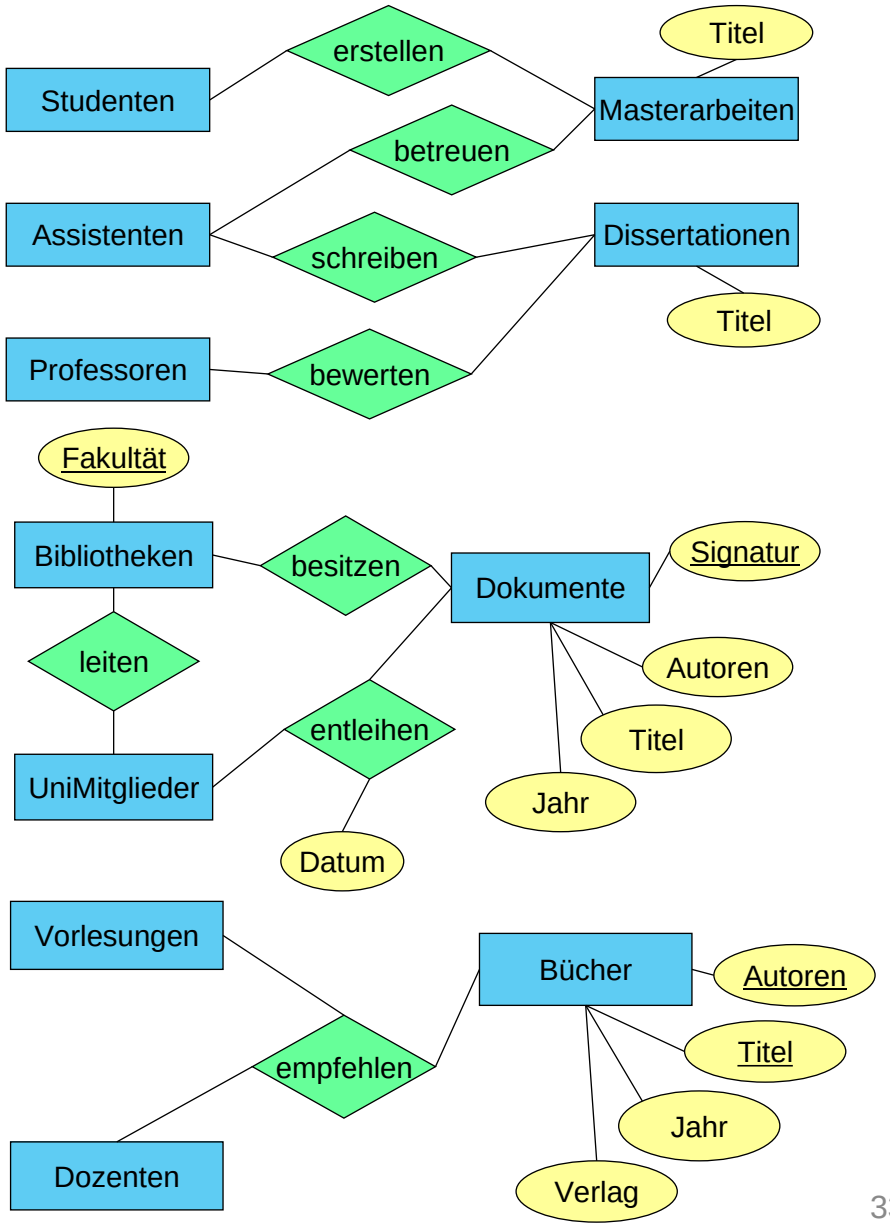


Beispiel Uni-DB, Sicht 3 der Dozenten

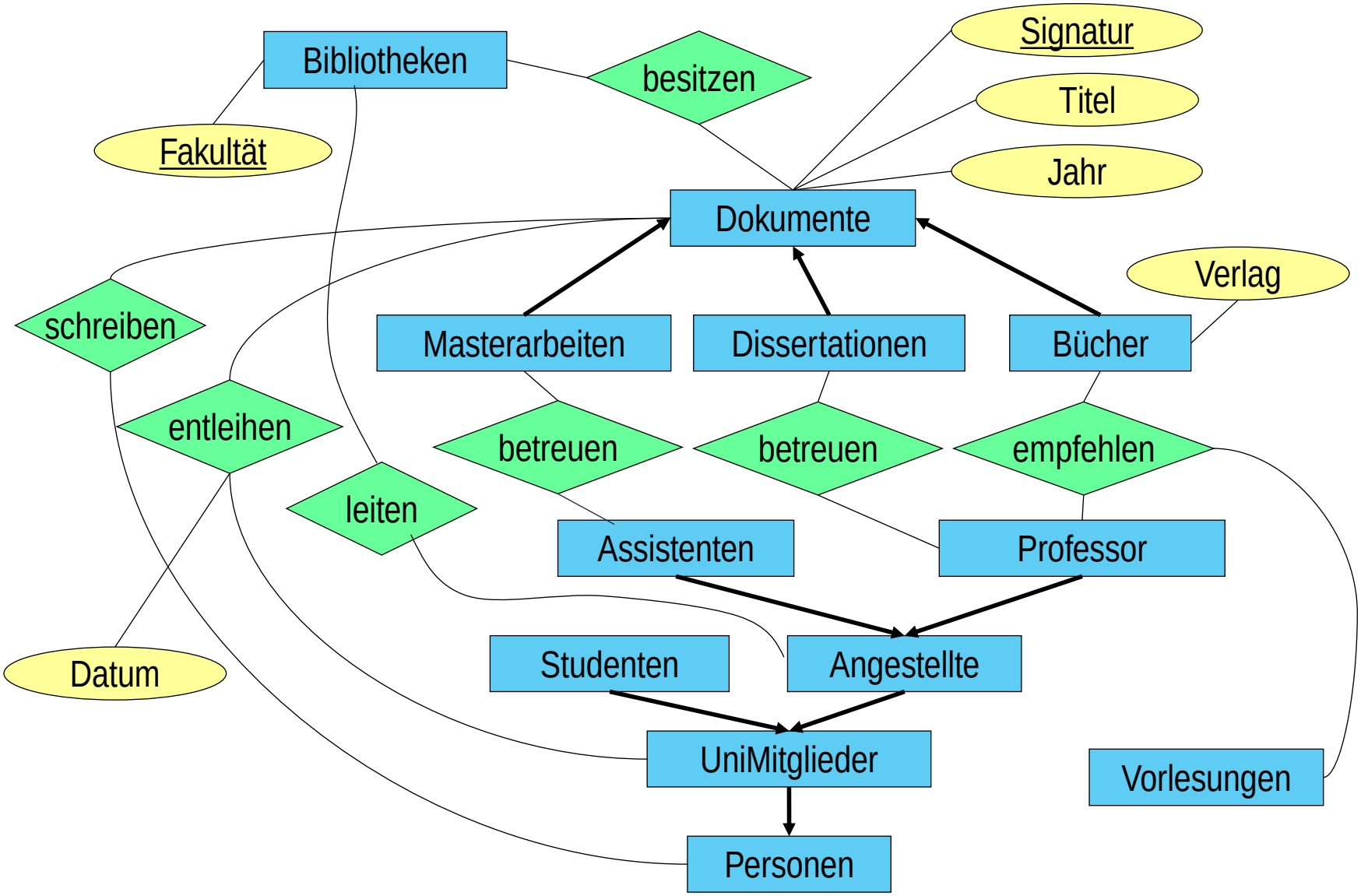


Beobachtungen

- *Dozenten* und *Professoren* sind synonym verwendet
- Entitytyp *UniMitglied* ist eine Generalisierung von *Student, Professor und Assistent.*
- Bibliotheken werden von *Angestellten* (nicht z.B. Studenten) geleitet. Sicht 2 ist hier ungenau.
- *Dissertationen*, *Masterarbeiten* und *Bücher* sind Spezialisierungen von *Dokumenten*, welche von *Bibliotheken* verwaltet werden.
- *Erstellen* und *schreiben* sind Synonyme in Sicht 1



Konsolidiertes Schema



ER Modellierung: Zusammenfassung

- ER beschreibt eine Miniwelt
  - Das "was" und die Regeln
  - ER ist statisch. Es beschreibt keine Übergänge
- Nützlich zum Erstellen von Software zur Beantwortung von (An)fragen über die Miniwelt
  - es folgt nun: ER-Modell → relationales Modell

- Ähnliche Modellierungsmöglichkeiten bietet UML (mehr auf OOP zugeschnitten)
- Auch andere graphische Darstellungen des ER Modells gebräuchlich, z.B. "Krähenfussnotation" optisch näher bei UML