Datenbanken

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Dennis Heinrich (Übungen)

und studentische Tutoren

Teilnehmerkreis und Voraussetzungen

Studiengänge

- Bachelor **Informatik**
- Bachelor/Master **Mathematik in Medizin und Lebenswissenschaften**
- Bachelor **Medieninformatik**
- Bachelor **Medizinische Ingenieurwissenschaft**
- Bachelor **Medizinische Informatik**

Voraussetzungen

- Programmierung
- Algorithmen und Datenstrukturen (Bäume)
- Lineare Algebra und Diskrete Strukturen 1

Vorteilhaft

- Einführung in die Logik

Organisatorisches: Übungen

- **Start**: Donnerstag, 20. April 2017
- **Übungen**: Donnerstags
  Anmeldung über Moodle nach dieser Veranstaltung
- **Übungsaufgaben** stehen jeweils nach jeder Vorlesung bis spätestens Mittwoch 12 Uhr über Moodle bereit
- Aufgaben sollen in einer **2-er Gruppe** bearbeitet werden
- **Abgabe der Lösungen** erfolgt bis Montag in der jeweils folgenden Woche nach Ausgabe bis 12 Uhr in der IFIS-Teeküche (1 Kasten pro Gruppe)
- Bitte unbedingt Namen, Matrikelnummern und Übungsgruppennummern auf Abgaben vermerken

Organisatorisches: Prüfung

- Die **Eintragung in den Kurs** und in eine Übungsgruppe ist **Voraussetzung**, um an dem Modul Datenbanken teilnehmen zu können und Zugriff auf die Unterlagen zu erhalten
- Am Ende des Semesters findet eine **Klausur** statt
- **Voraussetzung** zur Teilnahme an der Klausur sind mindestens **50% der gesamtmöglichen Punkte aller Übungszettel**

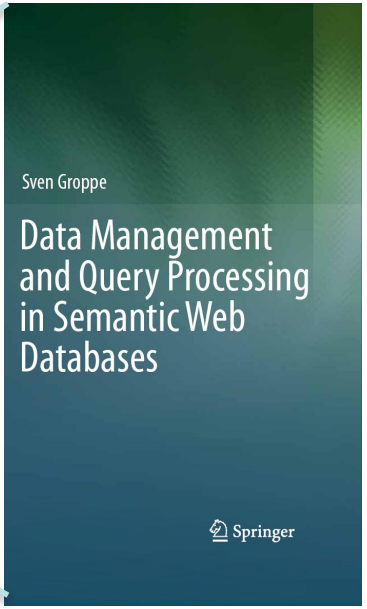
Literatur

A. Kemper, A. Eickler,
Datenbanksysteme: Eine Einführung,
9. Auflage, Oldenbourg Verlag, 2013.

R. Elmasri, S.B. Navathe,
Grundlagen von Datenbanksystemen: Bachelor-Ausabe,
3. überarbeitete Auflage, Pearson Studium, 2009.

Ausblick über IFIS Module

- **Bachelor-Programm**
  - Algorithmen und Datenstrukturen
  - Datenbanken
  - Non-Standard-Datenbanken
- **Master-Programm**
  - Webbasierte Informationssysteme
  - Datenmanagement
    - Mobile und verteilte Datenbanken
    - Semantic Web
  - Web and Data Science
    - Foundations of Ontologies and Databases for Information Systems
    - Web Mining Agents



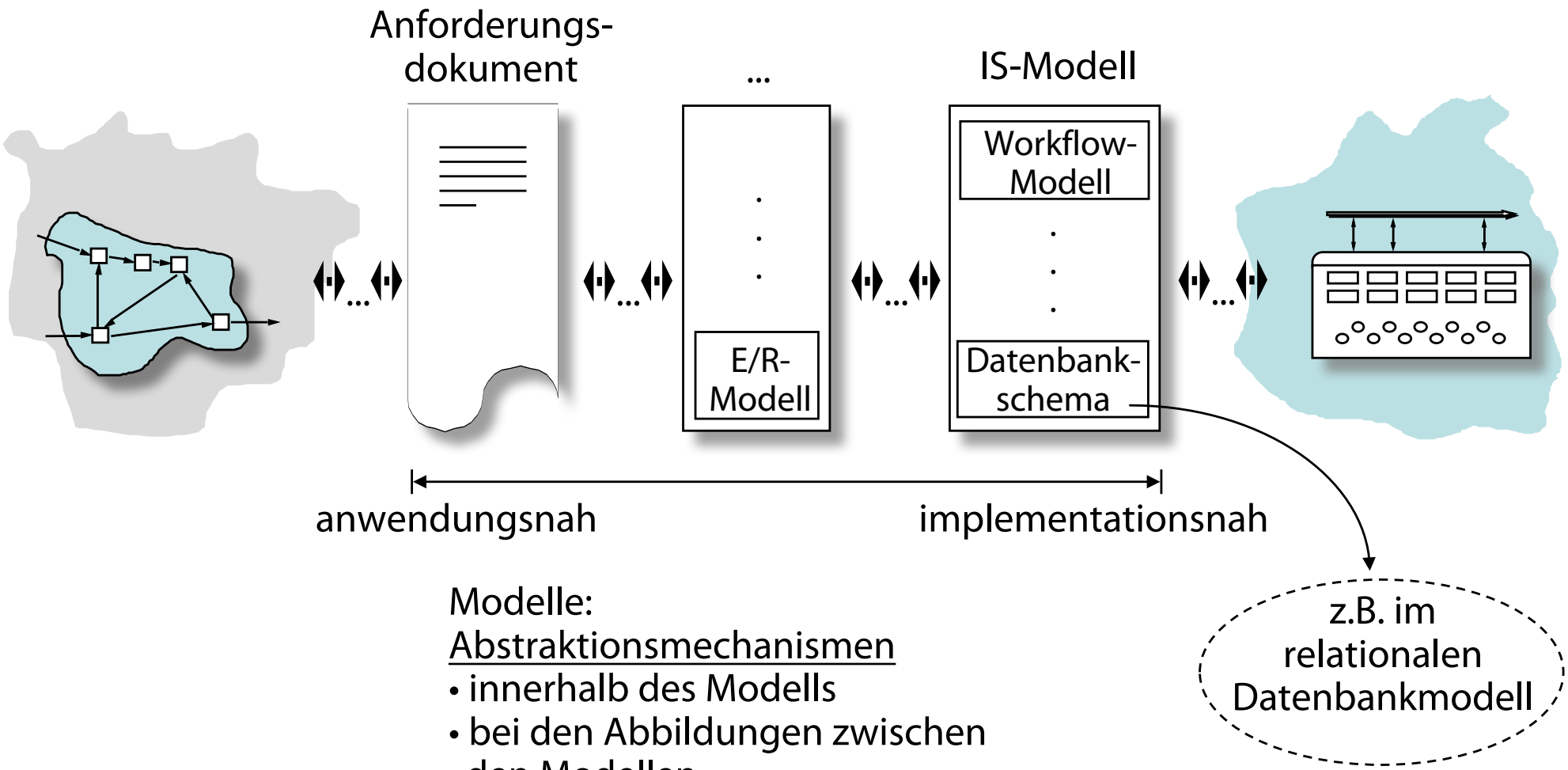
Kapitel 1: Einführung

Kennzeichen von Daten in Datenbanken

- lange Lebensdauer (Jahre, Jahrzehnte)
- reguläre Strukturen
- große Datenobjekte, große Datenmengen
- stetig anwachsende integrierte Bestände (Giga-, Tera-, Petabyte)

Decimal		
Value		SI
1000	k	kilo
1000^2	M	mega
1000^3	G	giga
1000^4	T	tera
1000^5	P	peta
1000^6	E	exa
1000^7	Z	zetta
1000^8	Y	yotta

Modelle und Abstraktion



Modelle:
Abstraktionsmechanismen
- innerhalb des Modells
- bei den Abbildungen zwischen den Modellen

P. Chen, The Entity-Relationship Model - Toward a Unified View of Data, ACM Transactions on Database Systems 1 (1), pp. 9–36, **1976**

E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387, **1970**

Erstes Beispiel einer (relationalen) Datenbank

- Datenbank für Inventar eines Weinkellers

Tabelle Weinkeller

Gestell	Sorte	Jahrgang	Anzahl_Flaschen
2	Franken	2009	5
1	Baden	2006	3
4	Rheinhessen	2007	10
1	Mosel	2013	2
2	Franken	2010	10

E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387, **1970**

Erste Anfrage

- Alle Infos zu Weinen, von denen der Weinkeller mindestens 4 Flaschen besitzt

Weinkeller

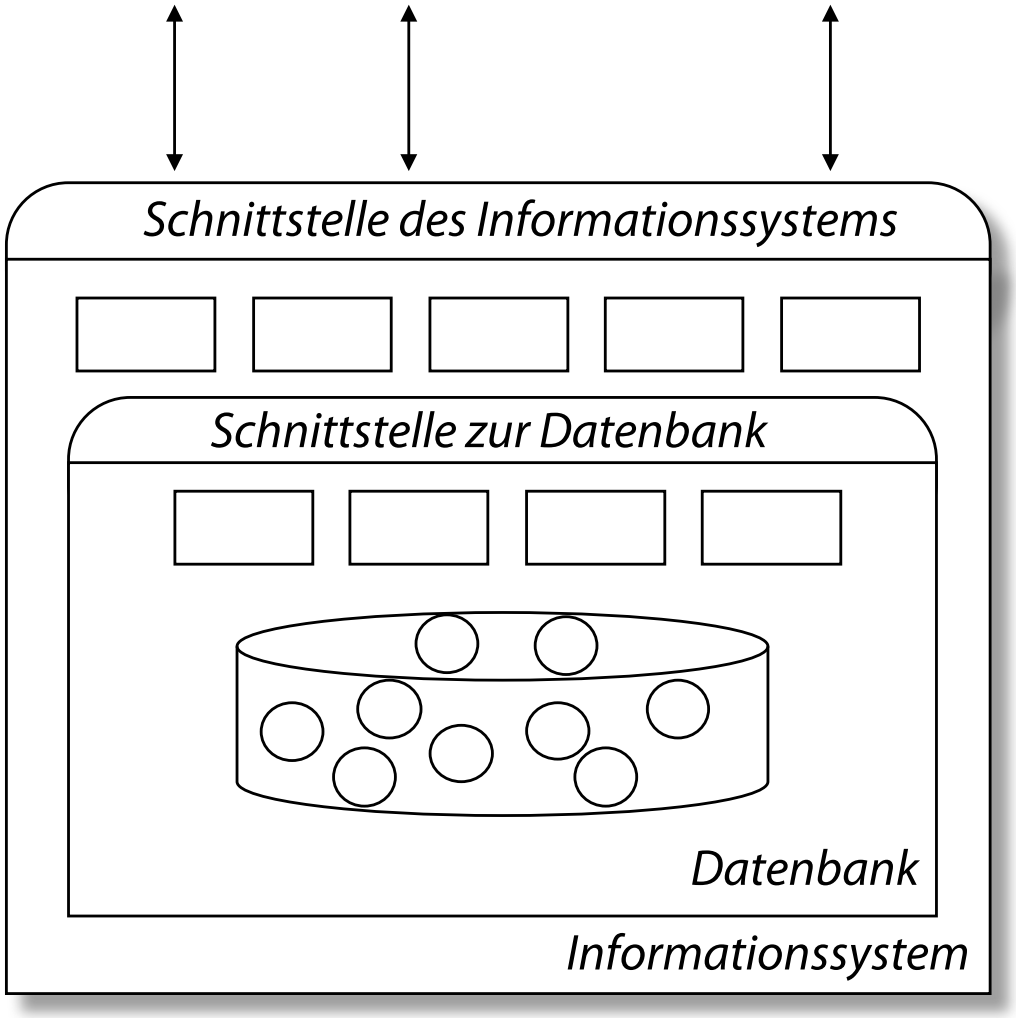
Gestell	Sorte	Jahrgang	Anzahl_Flaschen
2	Franken	2009	**5**
1	Baden	2006	3
4	Rheinhessen	2007	**10**
1	Mosel	2013	2
2	Franken	2010	**10**

```
SELECT Sorte, Gestell, Jahrgang
FROM   Weinkeller
WHERE Anzahl_Flaschen >= 4
```

Gestell	Sorte	Jahrgang
2	Franken	2009
4	Rheinhessen	2007
2	Franken	2010

Datenbanksysteme

Realisierung eines Informationssystems mit einer Datenbank:

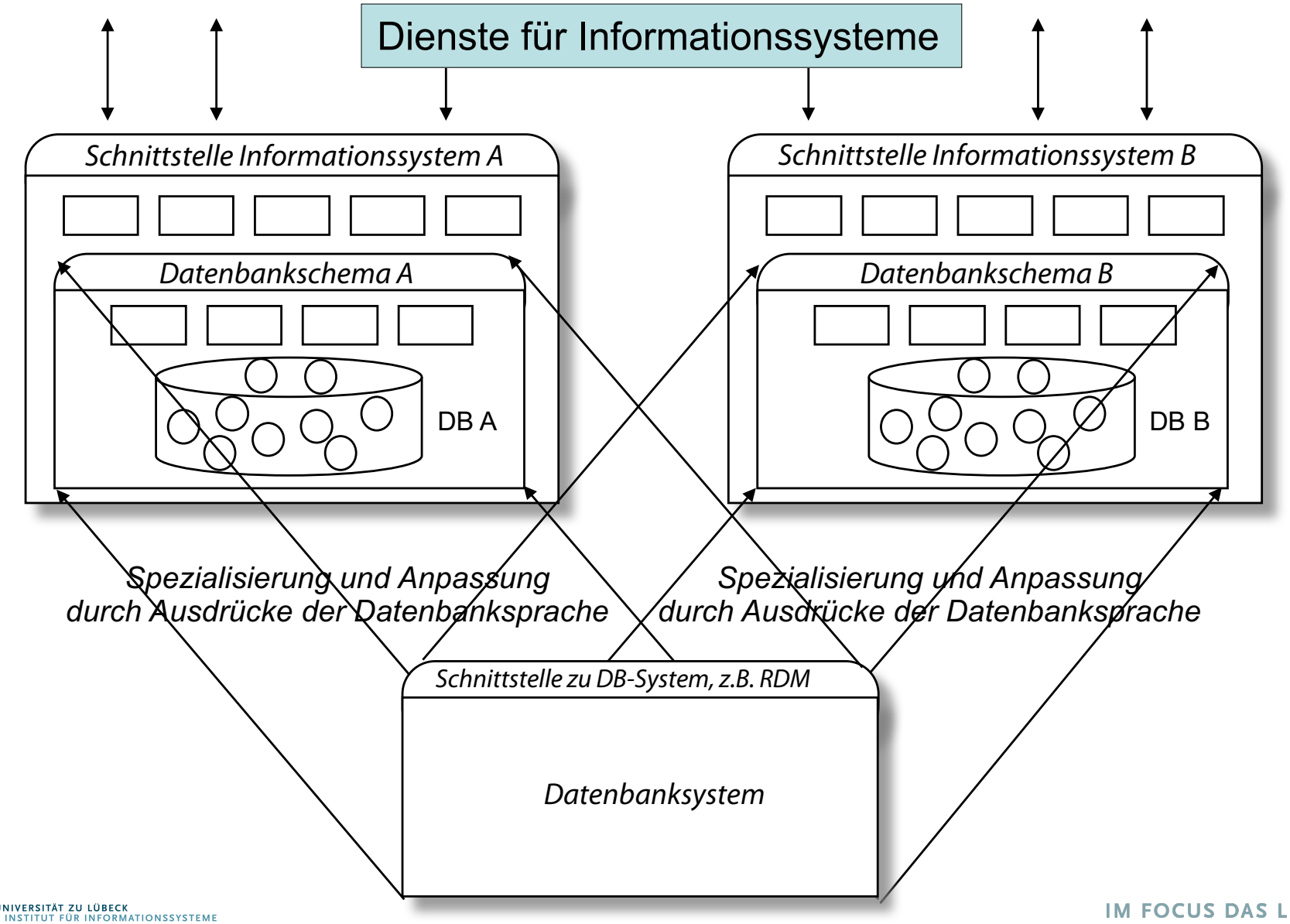


Algorithmen zur Informationsdarstellung, -verarbeitung und zur Integritätssicherung

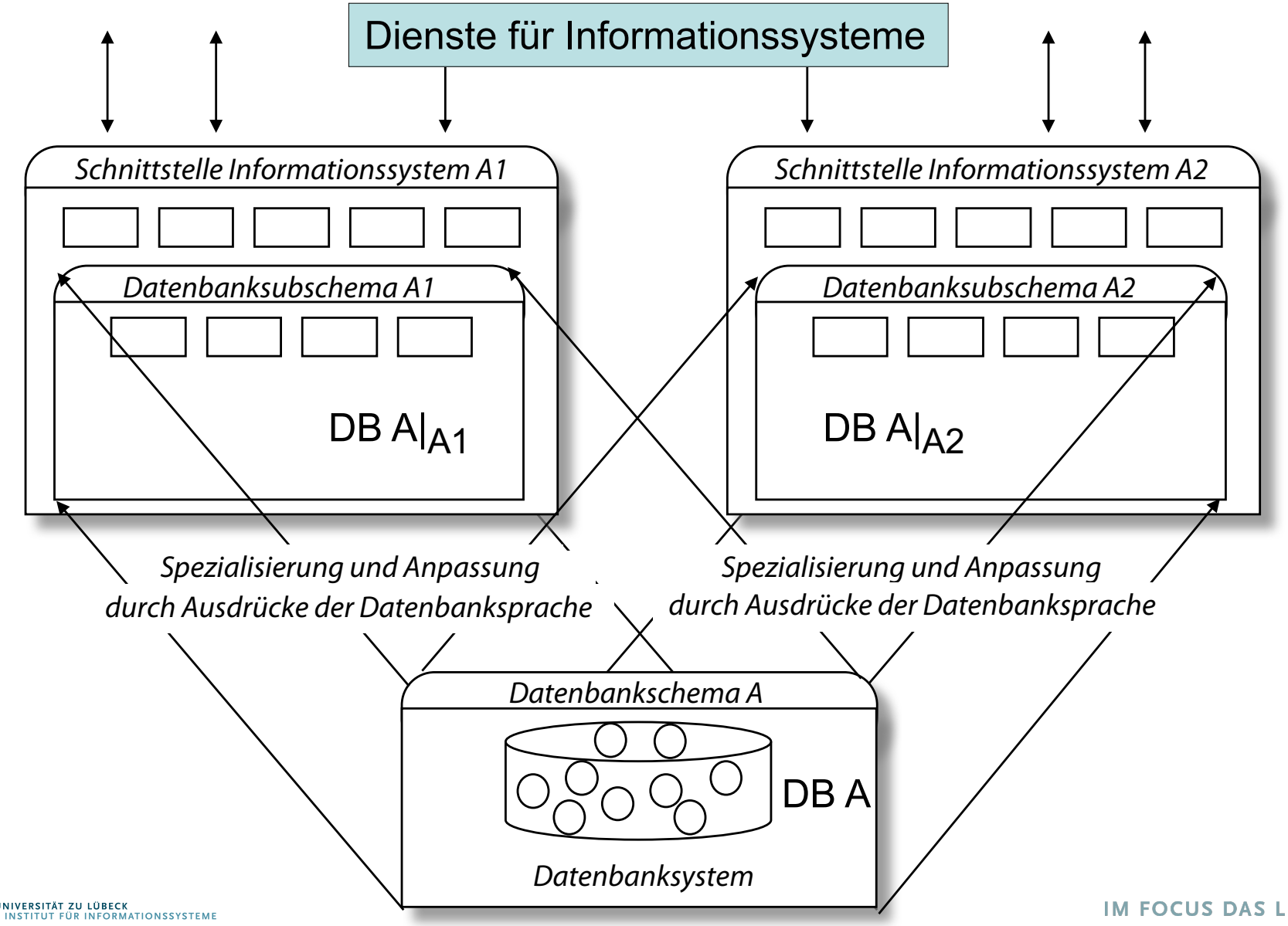
Dienste des Datenbanksystems zur Datenspeicherung, -anfrage und Integritätssicherung (Datenbankschema)

Datenbankzustand

Generisches Datenbankmodell und -system

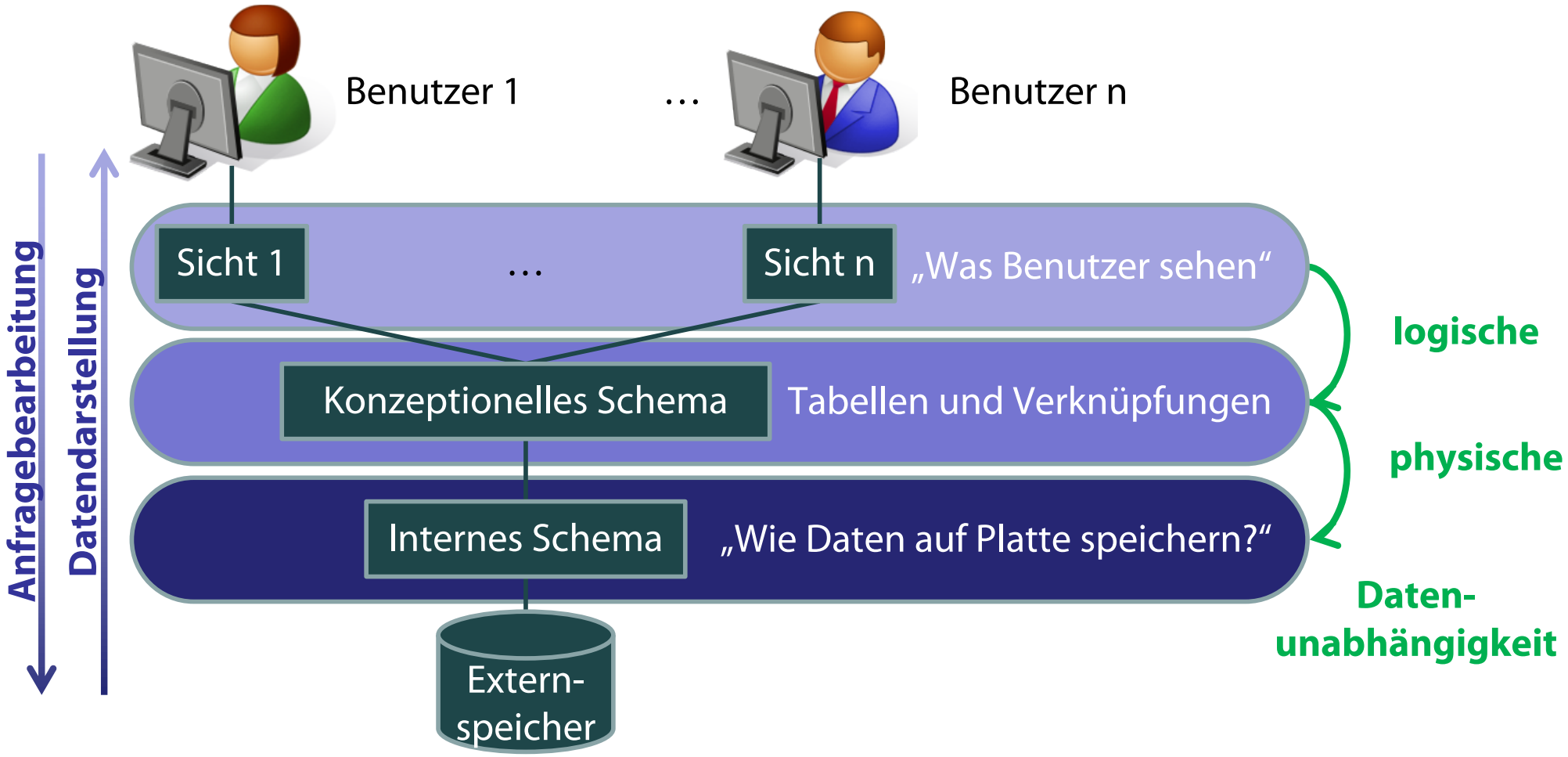


Sichten: DB-Subschemata und Subdatenbanken



ANSI-SPARC-Architektur

ANSI: **A**merican **N**ational **S**tandards **I**nstitute
SPARC: **S**tandards **P**lanning **a**nd **R**equirement **C**ommittee

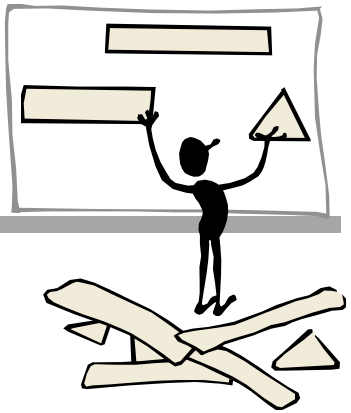


Jede Schicht ist unabhängig von deren unteren Schichten

ANSI/X3/SPARC Study Group on Data Base Management Systems, Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2, **1975**

Final report of the ANSI/X3/SPARC DBS-SG relational database task group. M. Brodie, J.W.. Schmidt (Ed.), SIGMOD Rec. 12, 4, **1982**

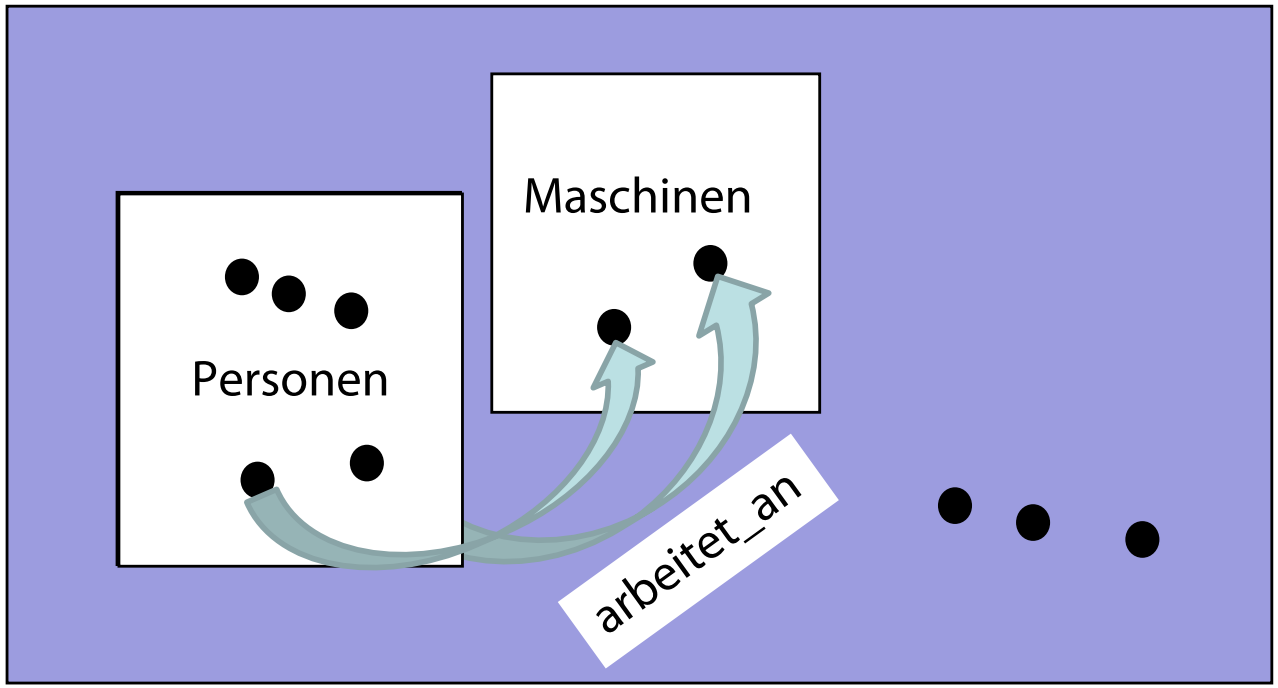
Datenunabhängigkeit



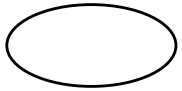
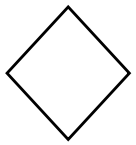
- Logisch
  - Konzeptionelles Schema kann (bedingt) geändert werden, ohne die Sichten zu verändern
    - Hinzufügen von Attributen und Tabellen zum konzeptionellen Schema
    - Verändern der Tabellenstruktur
- Physisch
  - Internes Schema kann geändert werden, ohne konzeptionelles Schema zu verändern
    - Veränderung des Speicherortes
    - Änderung des Speicherformates
    - Anlegen/Löschen von Indizes (für Anfrageoptimierung)

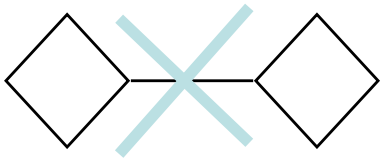
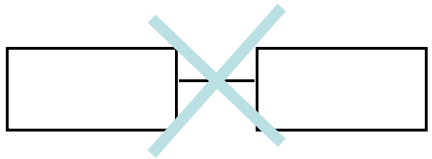
Kapitel 2: Entity-Relationship-Modellierung

- Objekte (Entitäten) mit ähnlichen Eigenschaften können zu Mengen (Entitätstypen, Klassen) zusammengefasst werden.
- Jedes Objekt ist "Instanz" einer oder mehrerer Klassen.
- Extension (Menge aller Instanzen einer Klasse)
- Objekte können in Beziehung gesetzt werden (Beziehungstyp, Relationship)



Grundlegende Elemente von ER-Diagrammen

- Objekttyp  (auch Entitätstyp oder Klasse genannt, Menge von Objekten)
- Werttyp  (für Basisdatentypen, Menge von „Werten“ bzw. Literale)
- Beziehungstyp  (Menge von Tupeln von Objekten)
- Die Elemente von ER-Diagrammen bilden einen bipartiten Graphen:

 und 

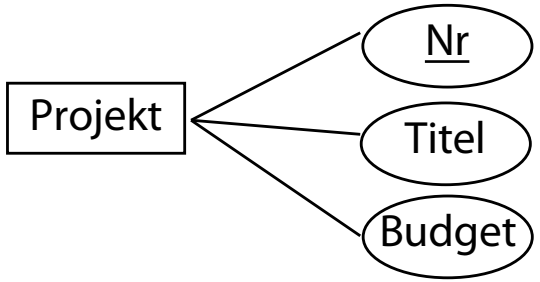
Verbindungen zwischen Symbolen der gleichen Typen sind nicht erlaubt.

P. Chen, The Entity-Relationship Model - Toward a Unified View of Data, ACM Transactions on Database Systems 1 (1), pp. 9–36, **1976**

Objekte/Entitäten und Attribute

Beispiele:

- Ein Projekt wird beschrieben durch
  - eine Nummer
  - einen Titel
  - das Budget



- Mathematische Bedeutung von Projekt: Menge von Tupeln von Werten
- Ein Tupel kann als „Aggregat“ von Basiswerten aufgefasst werden.

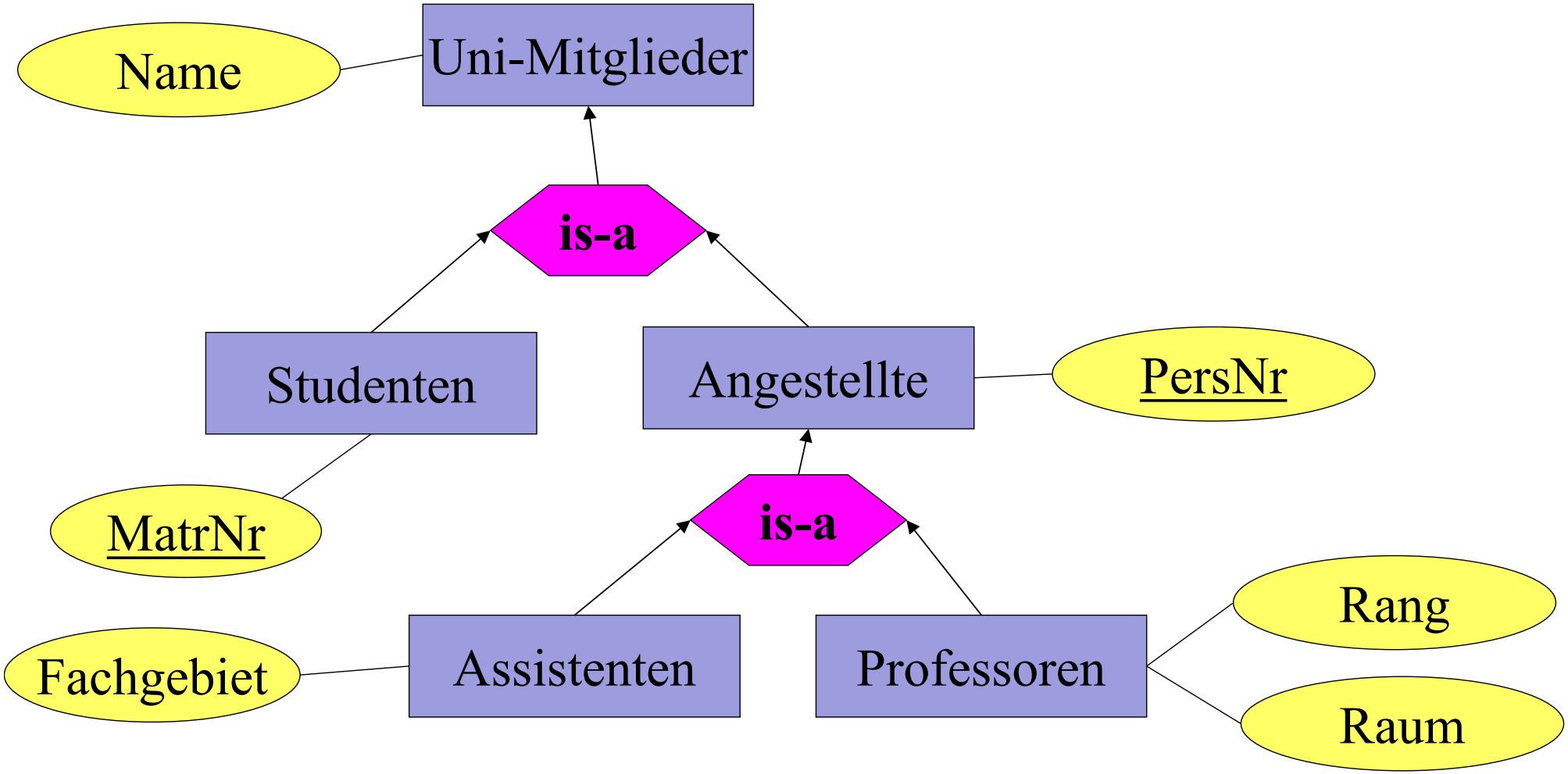
- Tupel können durch bestimmte Attribute eindeutig gekennzeichnet sein
  - In der Graphik sind diese Attribute unterstrichen
  - Wir nennen die Attribute „Schlüssel“
  - Mehr dazu gleich

Generalisierung und Spezialisierung (1)

- **Spezialisierung** bezeichnet die Verfeinerung einer Klasse (mehr Information/Anforderungen bzgl. der jeweiligen Individuen)
- **Generalisierung** ist die Vergröberung einer Klasse (weniger Information/Anforderungen bzgl. der jeweiligen Individuen)
- Spezielle Klassen (**Subklasse**) und allgemeine Klassen (**Superklasse**) bilden eine Subklassenhierarchie (→ Subtypisierung, Typhierarchie)
- **Instanzen** einer Klasse sind auch Instanzen der Superklasse
- Instanzen von Subklassen „**erben**“ die Eigenschaften der Superklasse und fügen evtl. neue hinzu (→ Vererbung von Beschreibungen für Tupelkomponenten)

Generalisierung und Spezialisierung (2)

Graphische Notation variiert

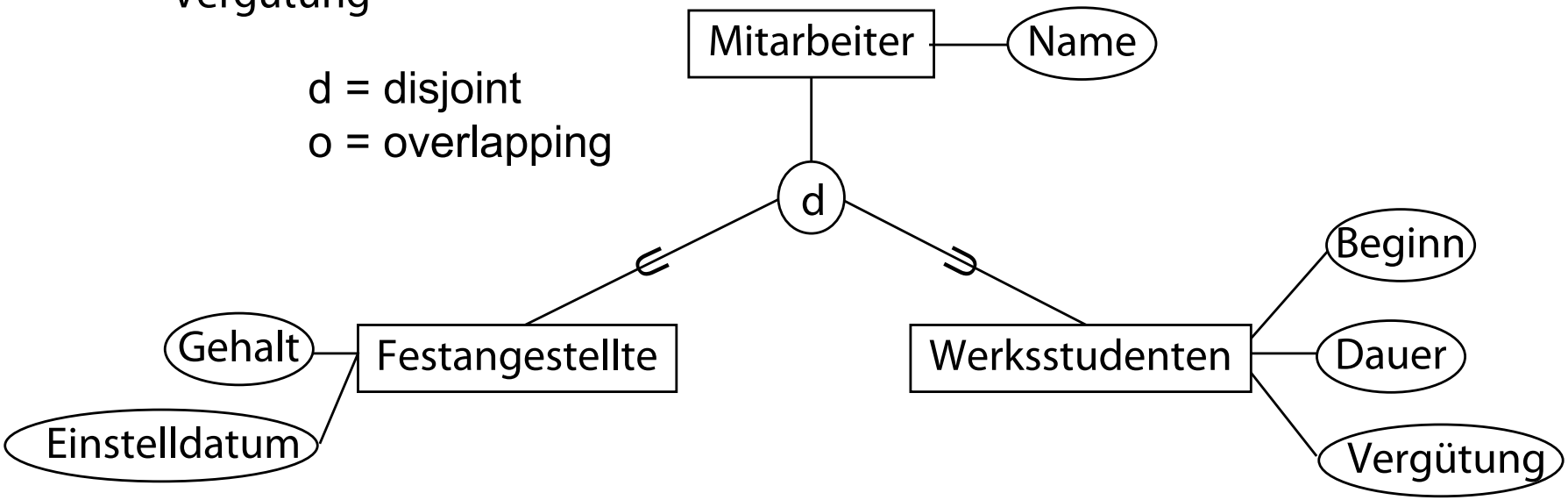


Generalisierung und Spezialisierung (3)

Erweiterte Entity-Relationship-Diagramme

Beispiel:

- Festangestellte und Werksstudenten sind Mitarbeiter. Festangestellte sind keine Werksstudenten
- Festangestellte haben die zusätzlichen Eigenschaften Gehalt und Einstelldatum.
- Werksstudenten haben die zusätzlichen Eigenschaften Beginn, Dauer und Vergütung

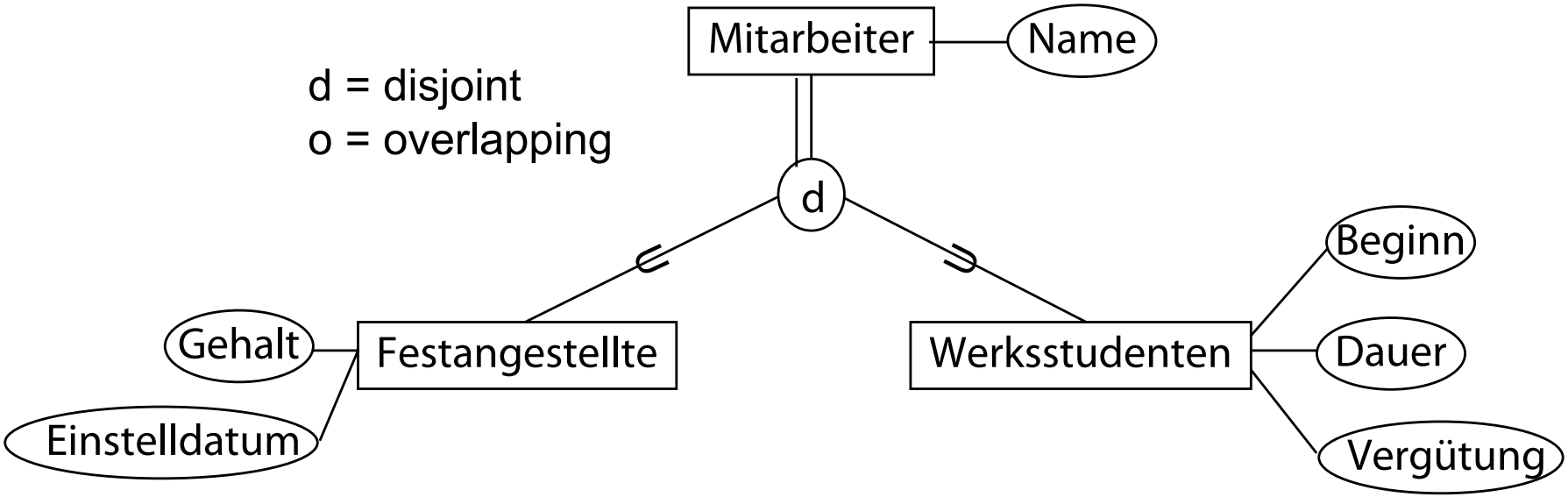


Generalisierung und Spezialisierung (4)

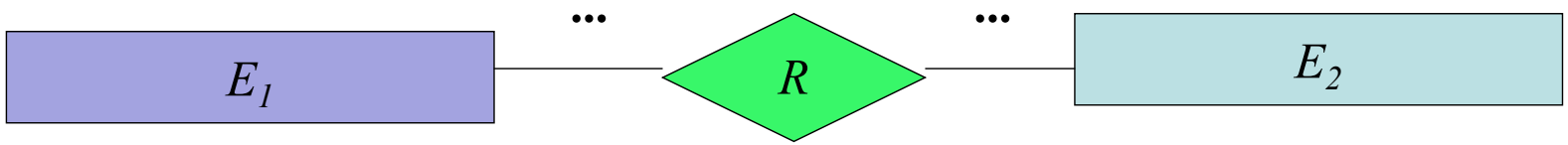
Erweiterte Entity-Relationship-Diagramme

Beispiel:

- Festangestellte und Werksstudenten sind Mitarbeiter. Festangestellte sind keine Werksstudenten
- Die Menge der Mitarbeiter ist gleich der Vereinigung der Mengen Festangestellte und Werksstudenten, d.h. ein Mitarbeiter ist entweder festangestellt oder Werksstudent

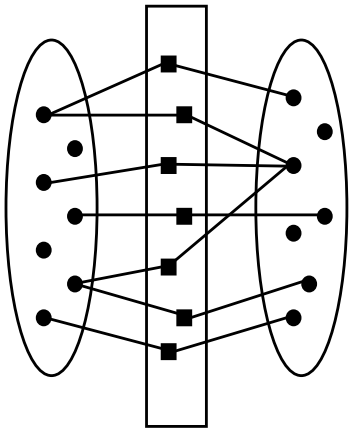


Assoziation / Relationship

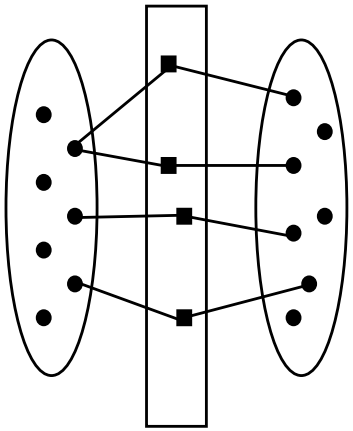


$$R \subseteq E_1 \times E_2$$

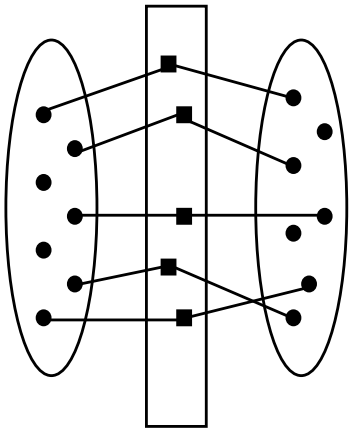
- Objekte können miteinander in Beziehung gesetzt (assoziiert) werden:
  - Binäre (ternäre, ...) Beziehungen assoziieren zwei (drei, ...) Klassen oder Objekte
  - Allgemein: n-äre Beziehungen zwischen n Klassen oder Objekten, wobei n der Grad der Beziehung ist
- Funktionalitätsangaben definieren Einschränkungen (siehe Bilder)



n:m



1:n

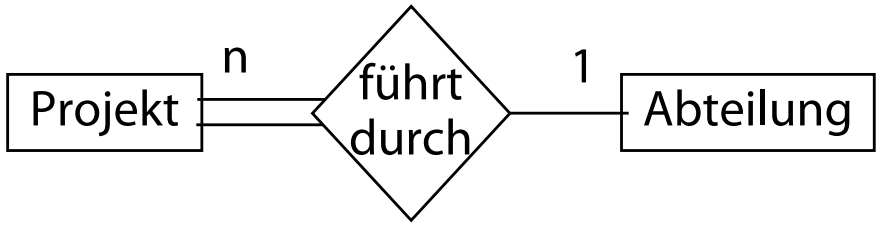


1:1

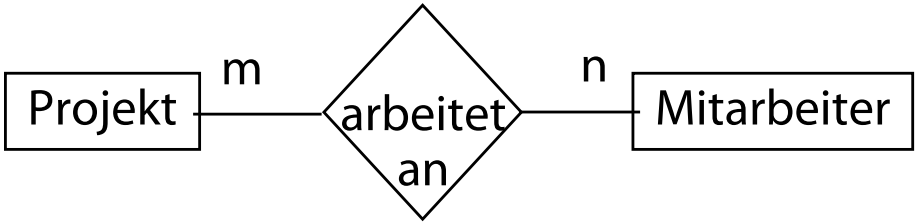
Assoziation / Relationship

Beispiele:

- Projekte werden von Abteilungen durchgeführt. Jedes Projekt muss einer Abteilung zugeordnet sein. Eine Abteilung kann mehrere Projekte ausführen.



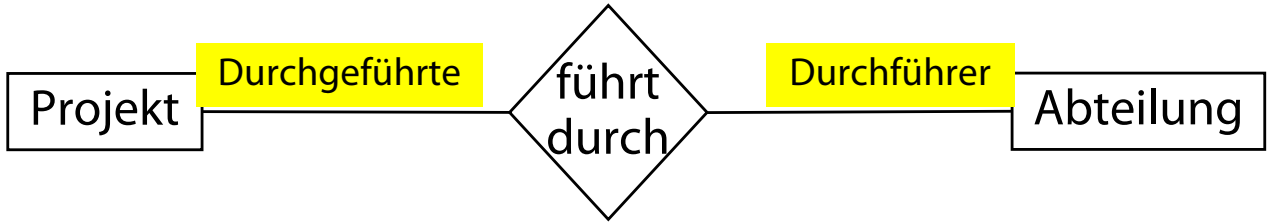
- An Projekten arbeiten Mitarbeiter. Ein Mitarbeiter kann an mehreren Projekten arbeiten. Jedes Projekt wird von beliebig vielen Mitarbeitern bearbeitet.



- Bemerkung:
  In der Literatur findet man auch andere Beschriftungsregeln.

Assoziation / Relationship

- **Totale Partizipation**: Jede Instanz einer Klasse muß mit einer Instanz der zweiten Klasse in Beziehung stehen (====)
- **Partielle Partizipation**: Eine Instanz einer Klasse kann in Beziehung zu einer Instanz der zweiten Klasse stehen (--------)
- Rollennamen identifizieren die Menge der Instanzen, die mit einer anderen Instanz in Beziehung stehen.
- Rollen können als abgeleitete Attribute verstanden werden, die die Menge der Instanzen als Attributwerte besitzen.



Lernziel 1:

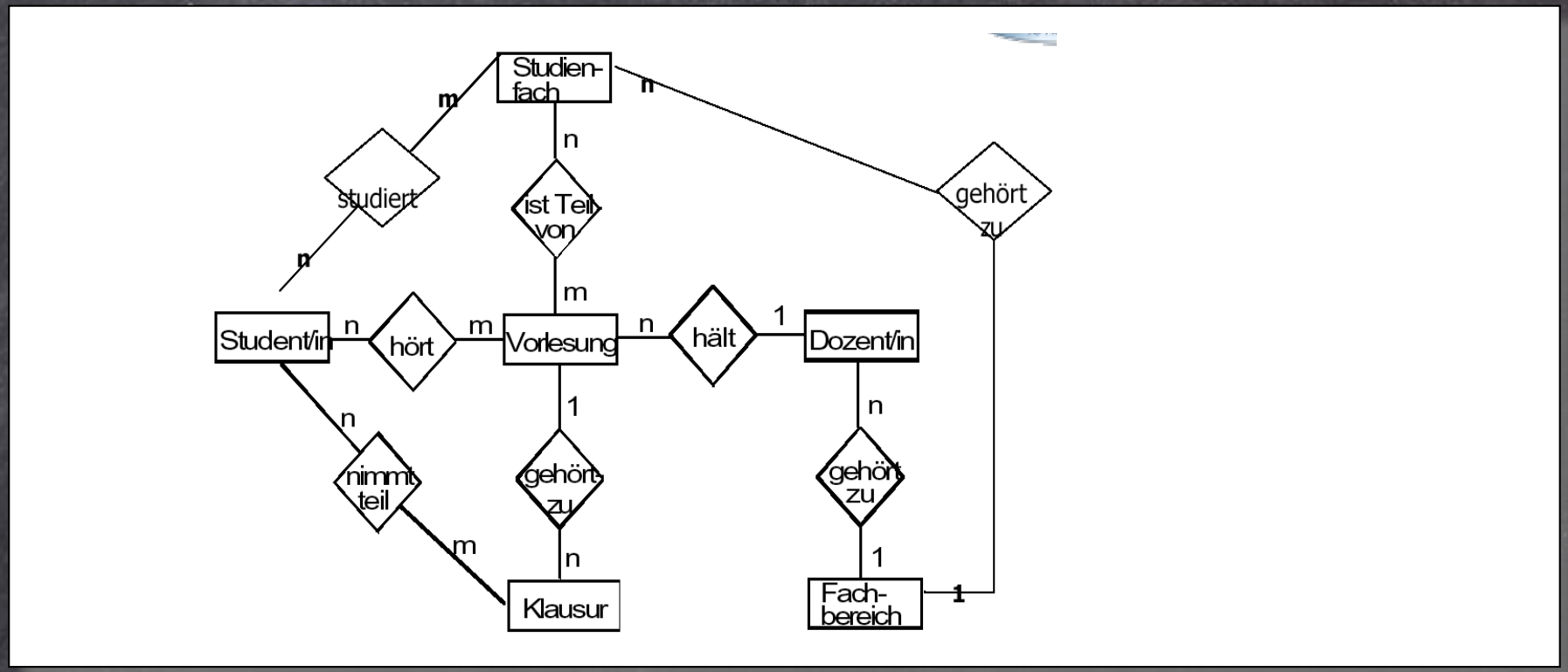
- Gegeben: "Anforderungsdefinition"

  - An einer Universität werden verschiedene Vorlesungen angeboten, die Teil mehrerer Studienfächer sind.
  - Diese Vorlesungen werden von genau einem Dozenten gehalten.
  - Jeder Dozent ist Mitglied genau eines Fachbereiches.
  - Ein Fachbereich hat mehrere Studienfächer.
  - Die Vorlesungen werden von Studenten gehört, die jeweils ein oder mehrere Studienfächer belegt haben.
  - Zu jeder Vorlesung werden mehrere Klausuren angeboten, die von den Studenten geschrieben werden.

- Gesucht: ER-Diagramm

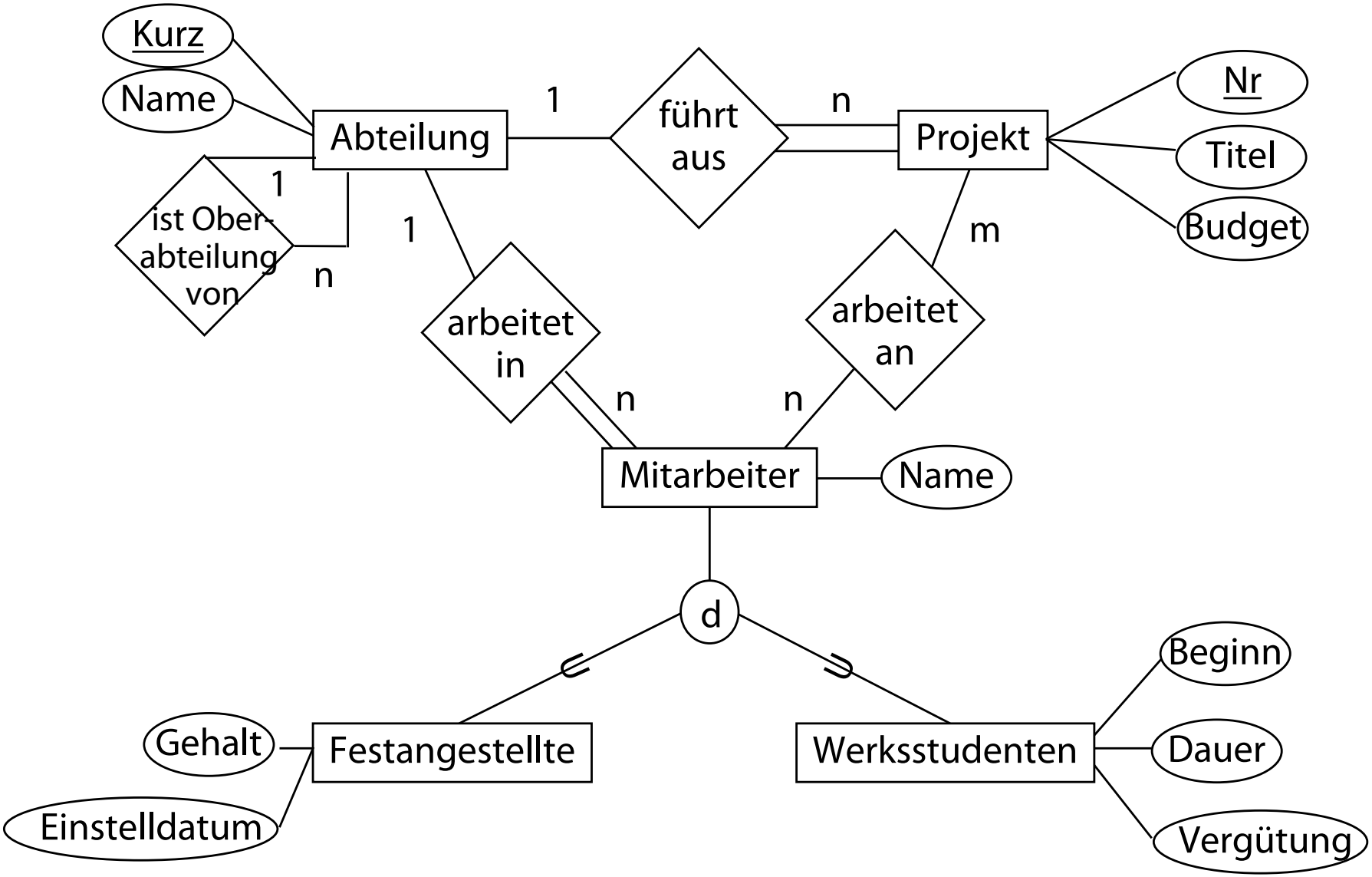
Lernziel 1:

- Gegeben: "Anforderungsdefinition"



- Gesucht: ER-Diagramm

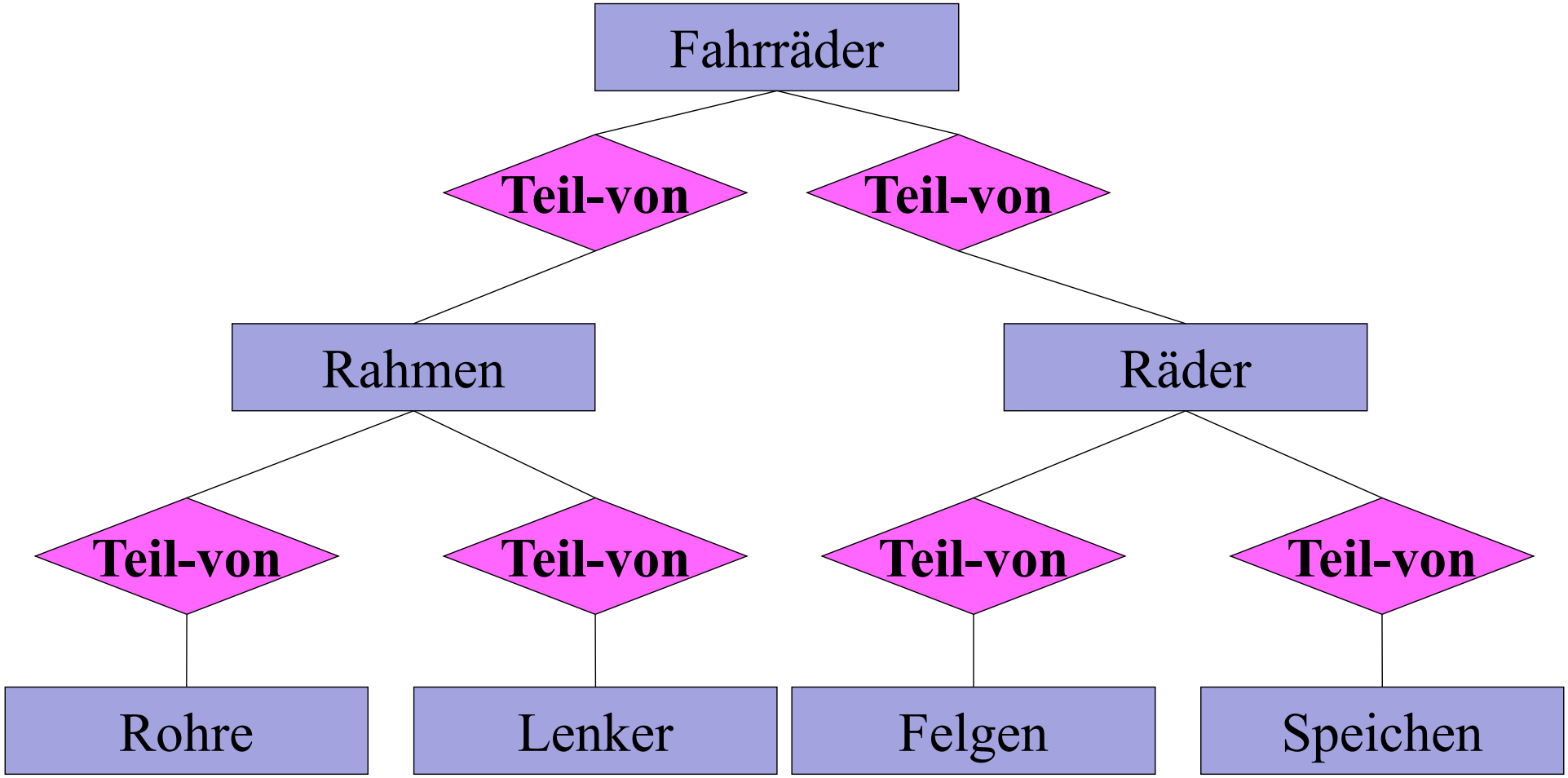
Lernziel 2: ER-Diagramm erläutern



Aggregation und Dekomposition (1)

- Objekte können zu übergeordneten Objekten aggregiert werden:
  - Beziehungen zwischen Komponenten und übergeordnetem Objekt
  - Übergeordnetes Objekt kann wiederum an Beziehungen teilnehmen.
- Im Vergleich zur "normalen" Assoziation wird die "Aggregation" in der Entity-Relationship-Modellierung nicht besonders unterstützt. Man verwende anwendungsspezifische Assoziationen

Aggregation



Identifikation und Schlüssel (1)

Zur **Identifikation** existieren zwei grundlegende Ansätze in Datenbankmodellen:

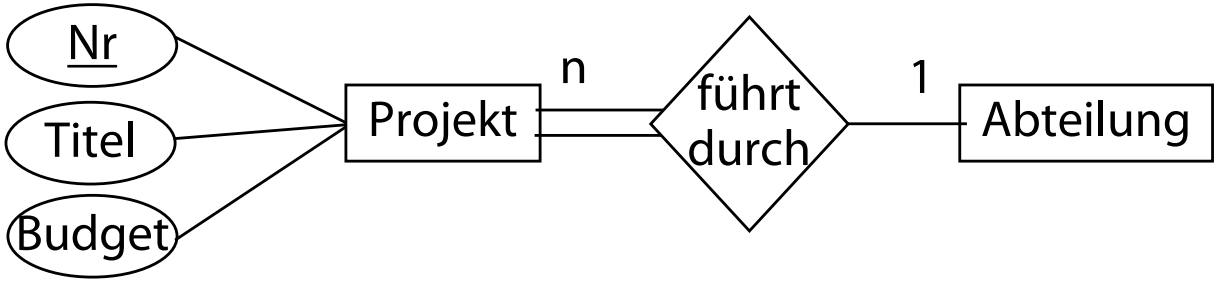
- **Referentielle Identifikation** bezeichnet direkte Verweise auf Objekte (→ Zeiger in Programmiersprachen).
- **Assoziative Identifikation** verwendet die Werte von Attributen oder Attributkombinationen, um sich eindeutig auf Objekte zu beziehen (→ Schlüssel: Ausweisnummer, Fahrgestellnummer, ...).
- In der Praxis benötigt man häufig beide Formen der Identifikation.

Schlüssel:

- Schlüssel sind Attribute oder Attributkombinationen mit innerhalb einer Klasse eindeutigen Werten und eignen sich deshalb zur Identifikation.
- Es kann mehrere Schlüsselkandidaten geben (Primärschlüssel, Sekundärschlüssel).
- Schlüssel stellen als Attributwerte Beziehungen zu anderen Objekten her (Fremdschlüssel).
- Durch Fremdschlüssel referenzierte Objekte müssen existieren (→ referentielle Integrität).

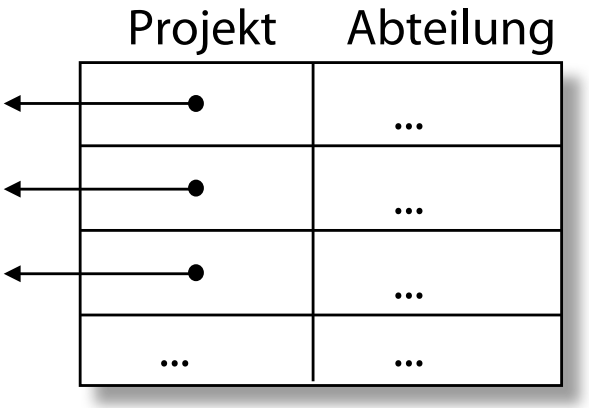
Identifikation und Schlüssel (2)

Beispiel: Projekte können durch eine Nummer eindeutig identifiziert werden.



Dabei existieren zwei Möglichkeiten zur Identifikation von Projekten innerhalb der Assoziation "führt durch":

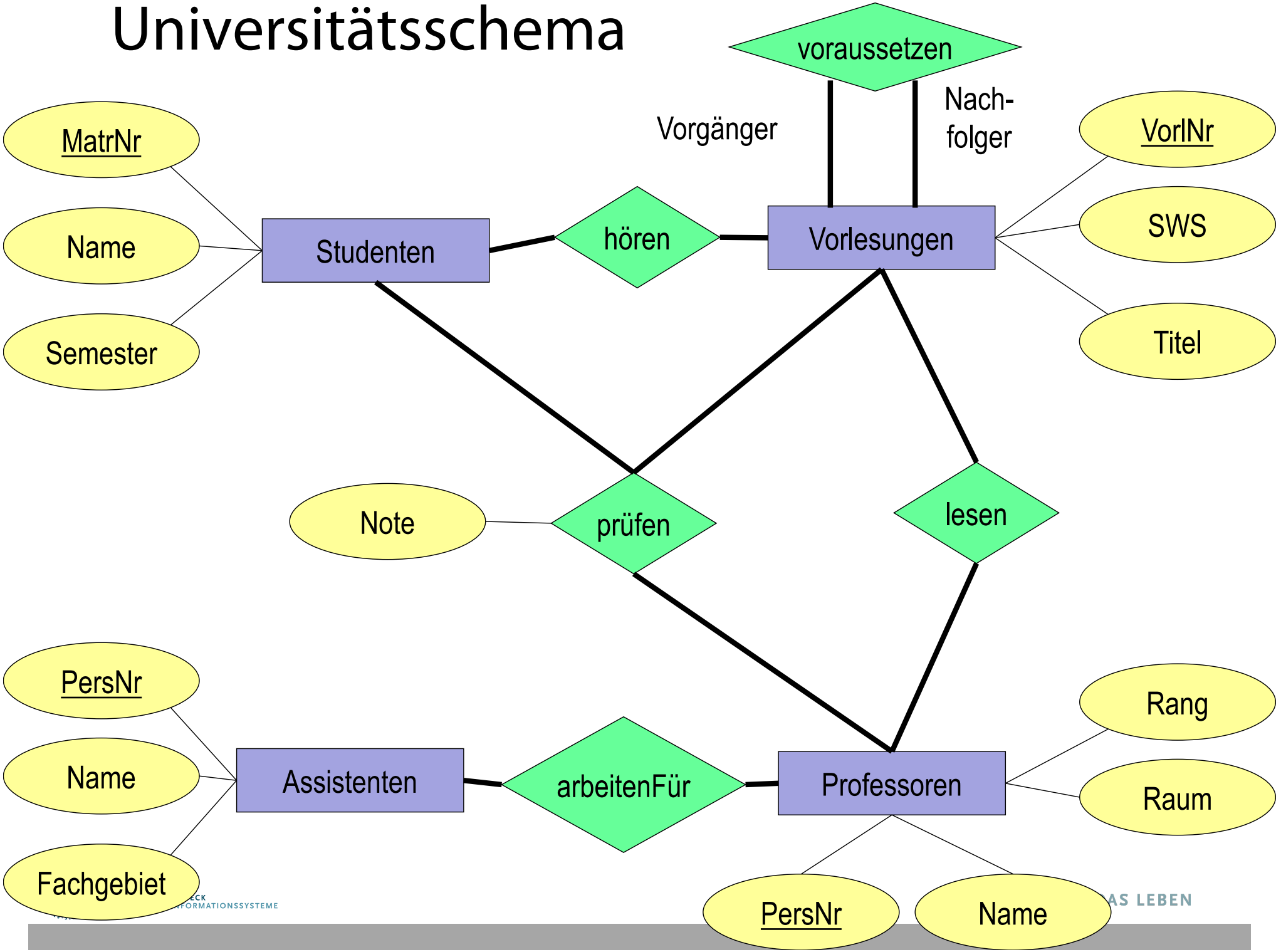
Referentielle Identifikation



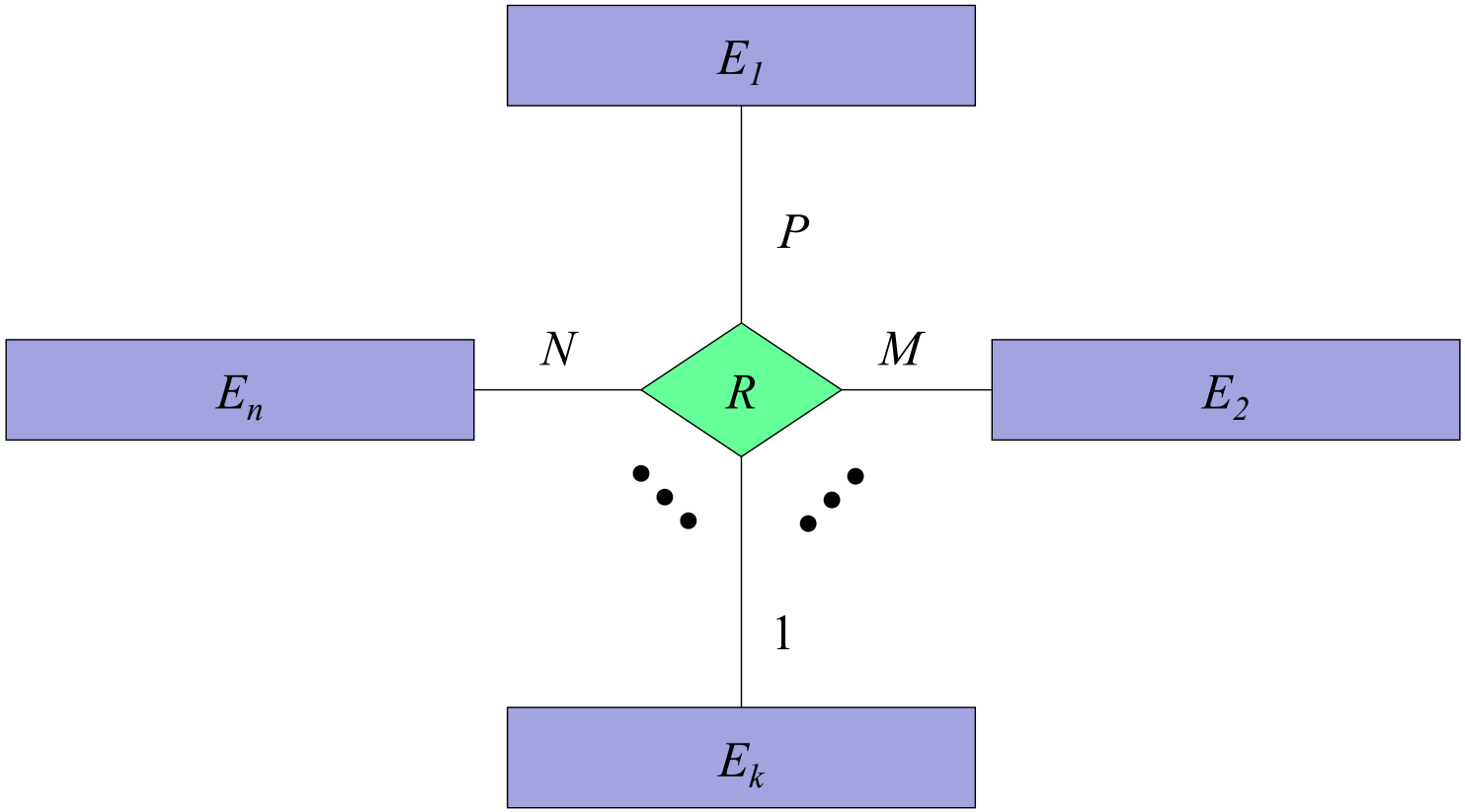
Assoziative Identifikation

Projekt	Abteilung
4711	...
4712	...
4713	...
....	...

Universitätsschema

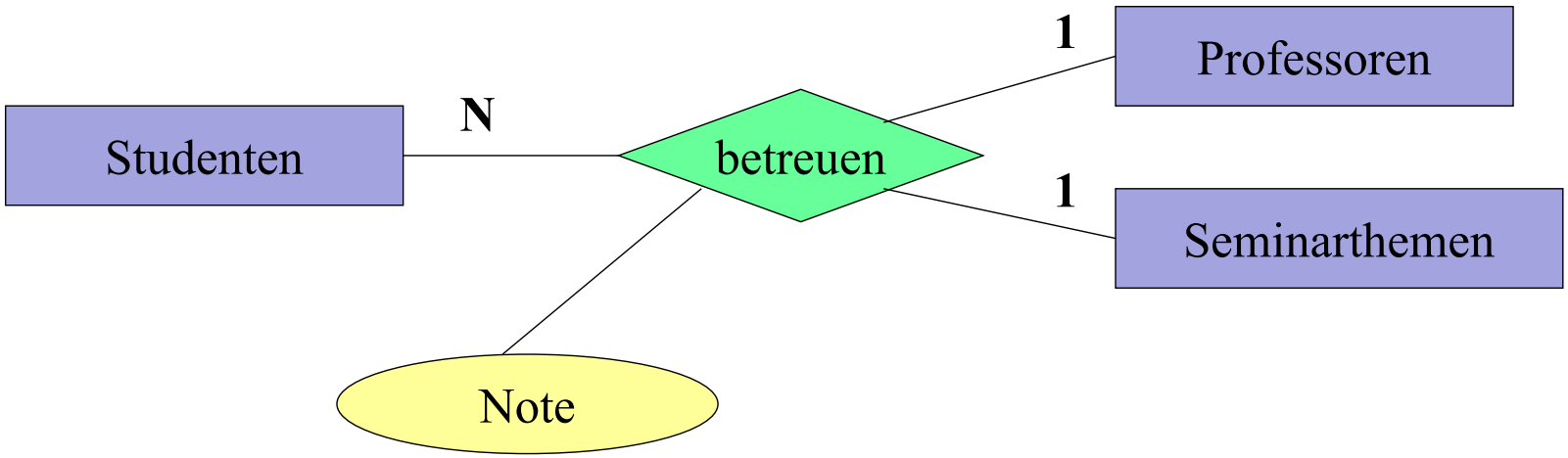


Funktionalitäten bei ***n***-stelligen Beziehungen



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

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).

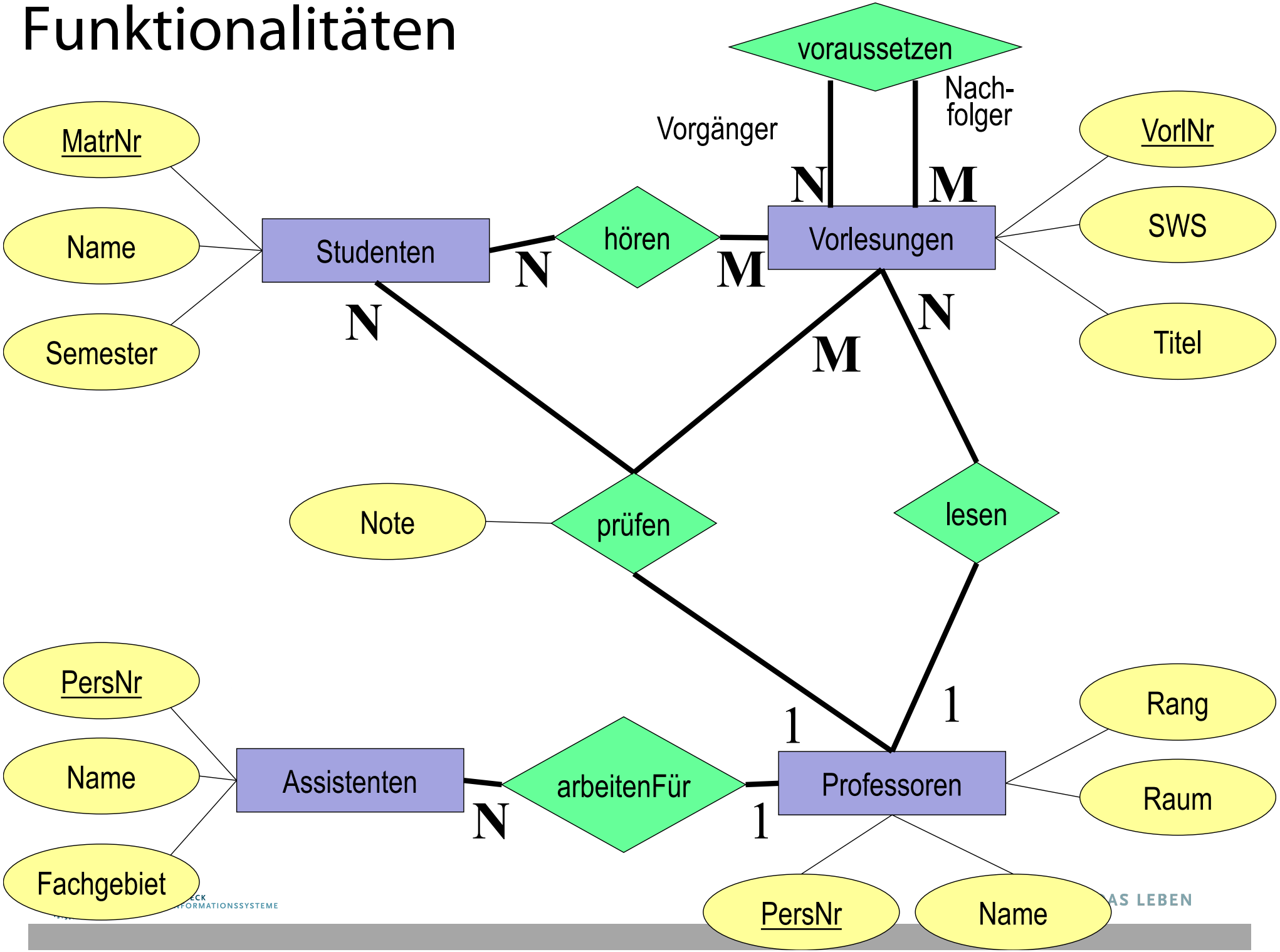
1. Studenten dürfen dasselbe Seminarthema nur einmal bearbeiten – sie dürfen also nicht bei anderen Professoren ein schon einmal erteiltes Seminarthema nochmals bearbeiten.

Folgende Datenbankzustände nach wie vor möglich:

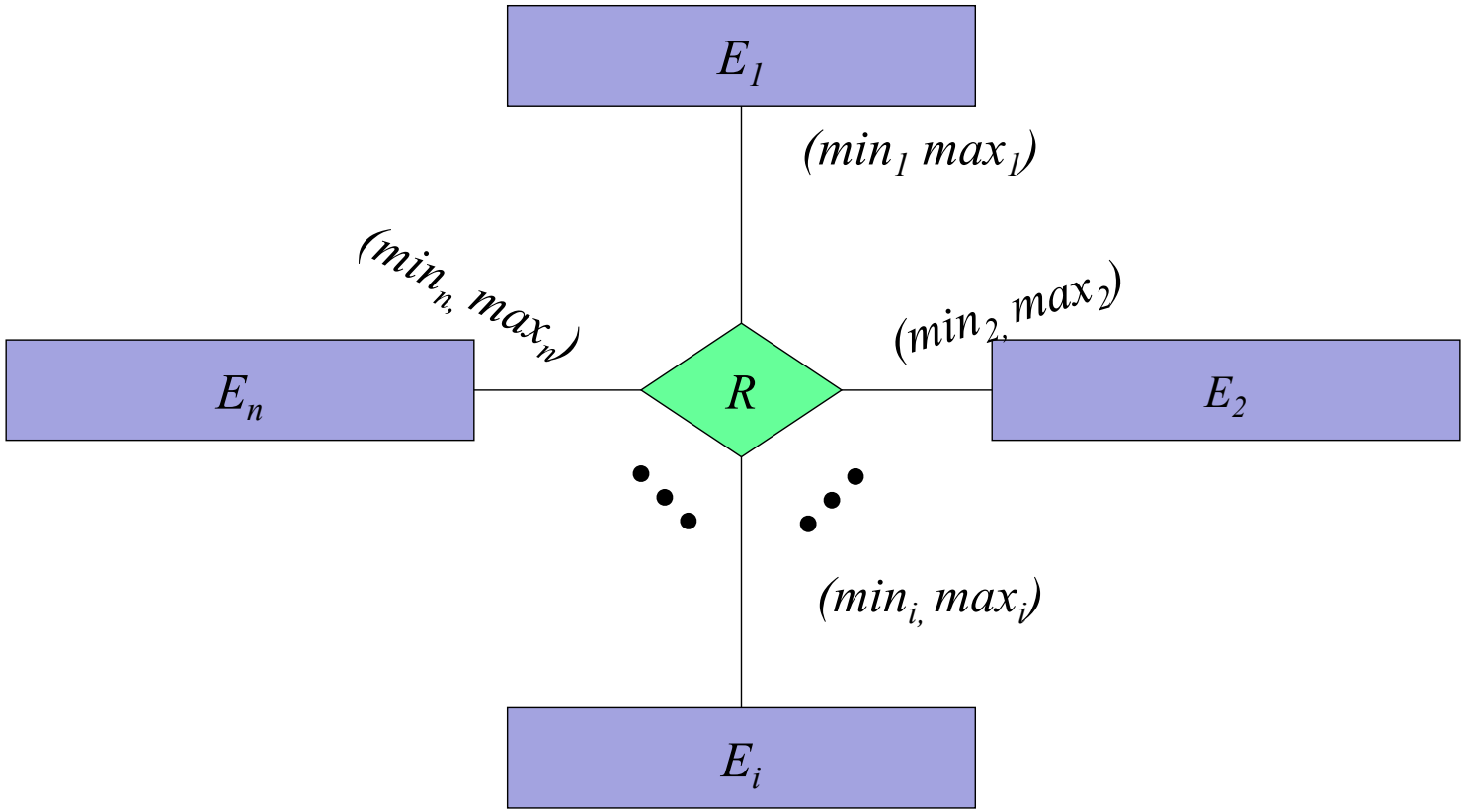
– Professoren können dasselbe Seminarthema „wiederverwenden“ – also dasselbe Thema auch mehreren Studenten erteilen.

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

Funktionalitäten



(Min, Max)-Notation / Kardinalitäten

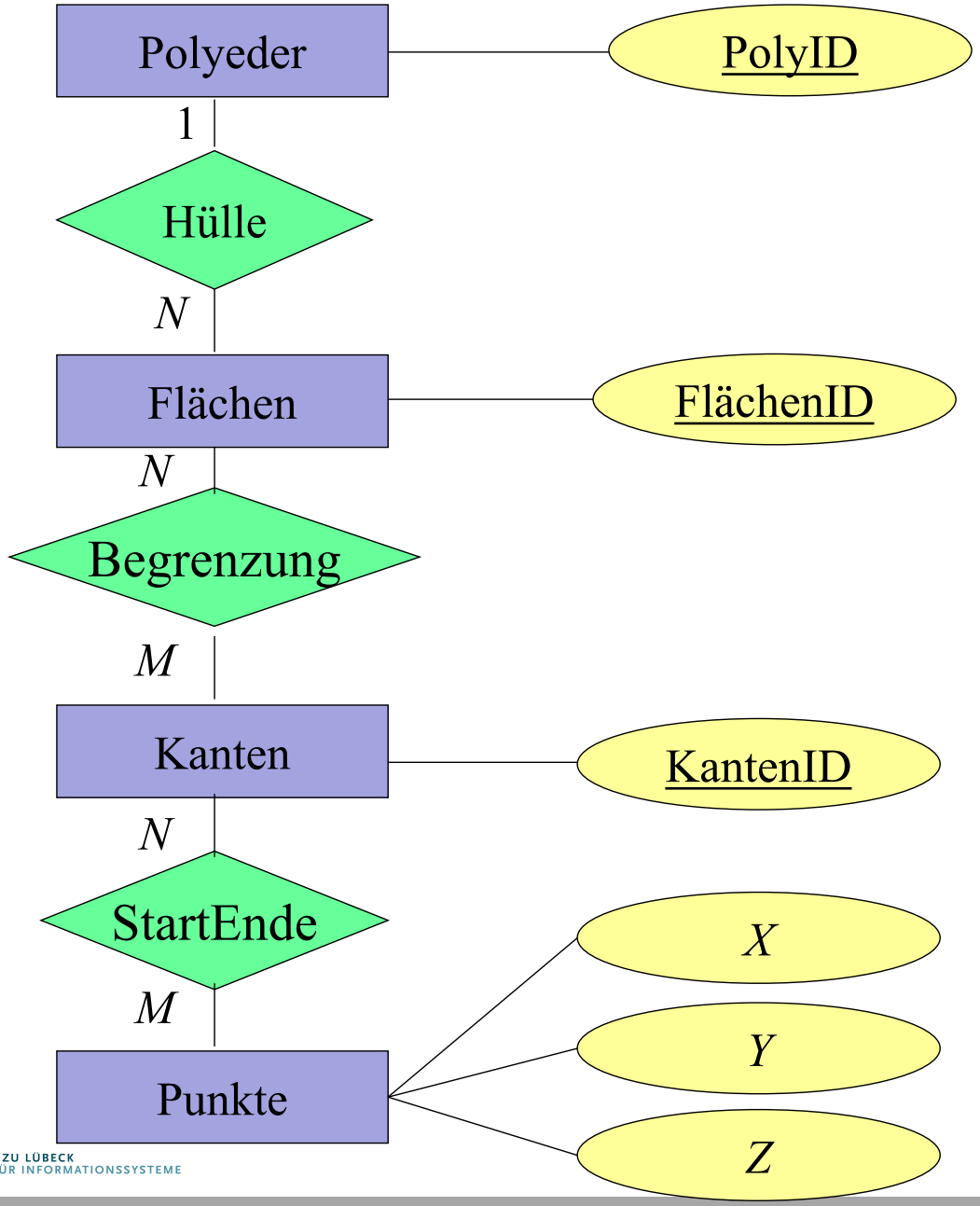


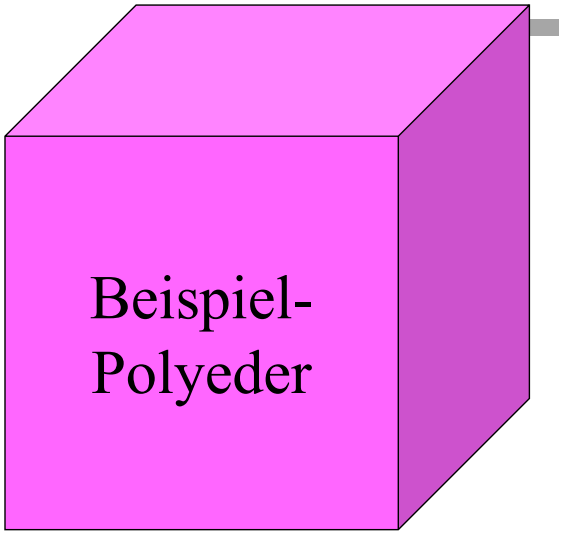
$R \subseteq E_1 \times \ldots \times E_i \times \ldots \times E_n$

Für jedes $e_i \in E_i$ gibt es

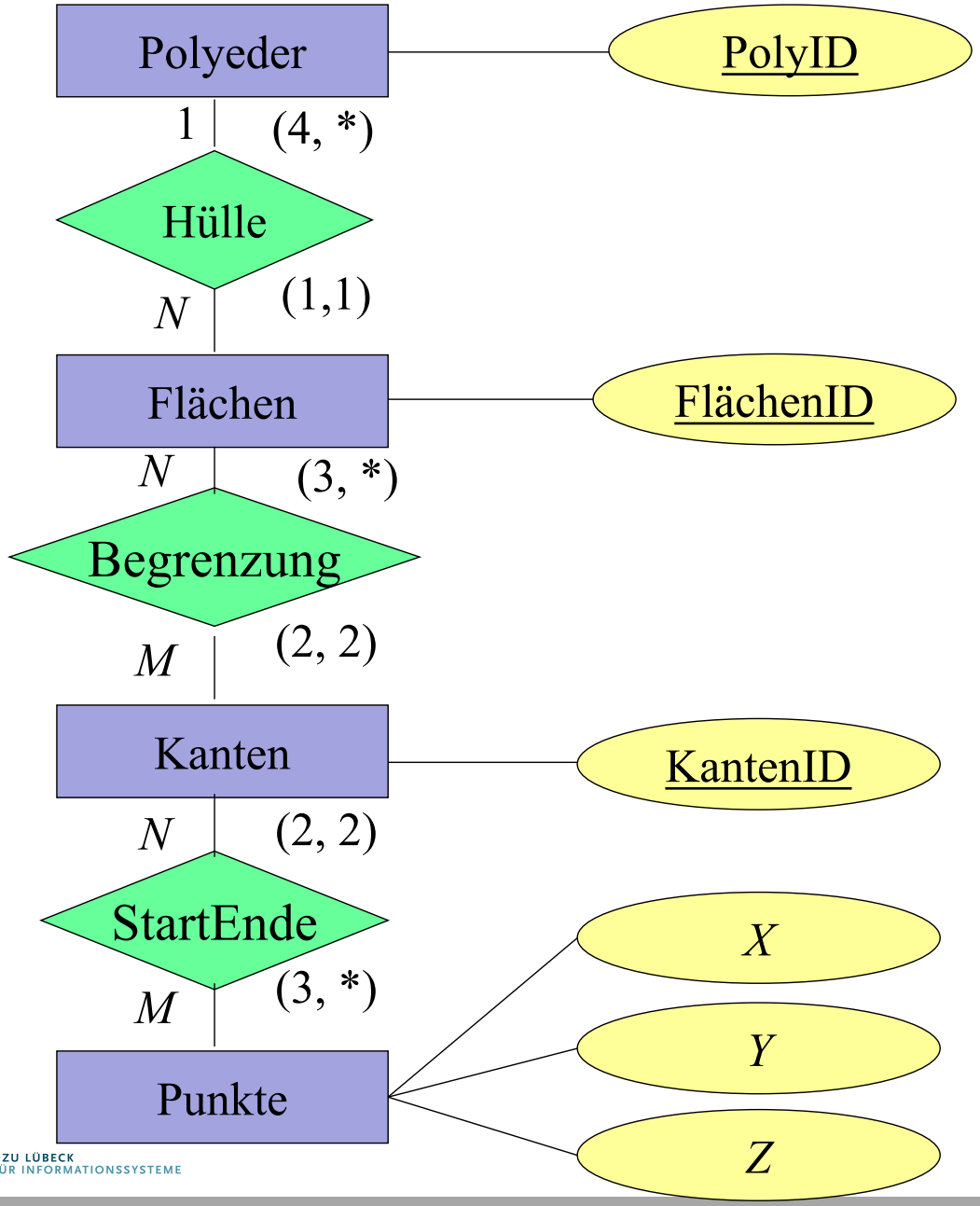
- Mindestens min_i Tupel der Art $(..., e_i, ...)$ und
- Höchstens max_i viele Tupel der Art $(..., e_i, ...) \in R^n$

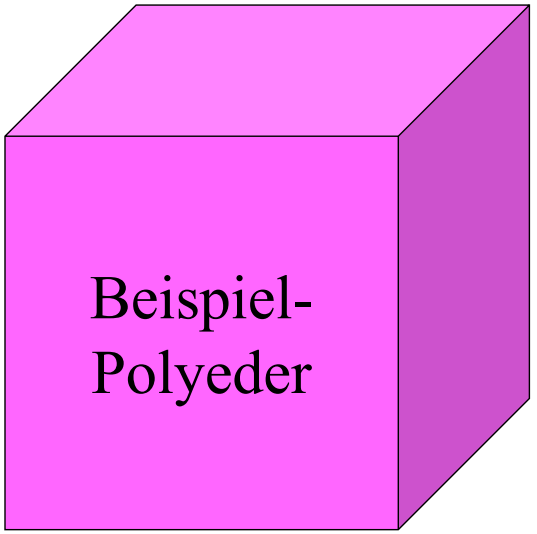
Komplex-strukturierte Entities

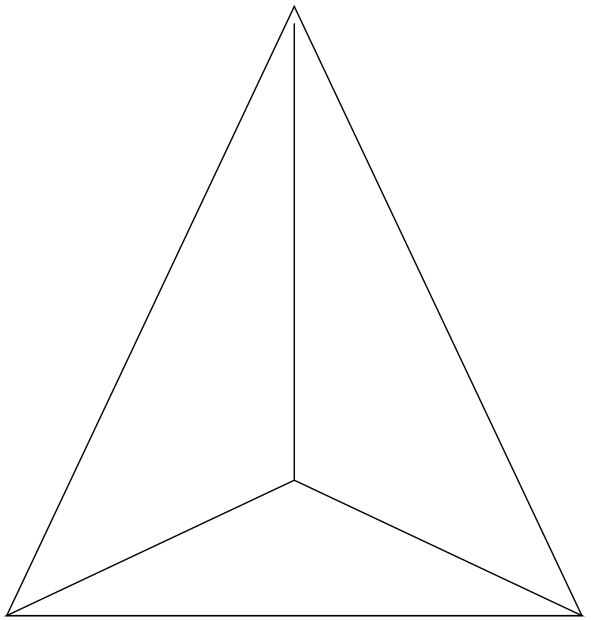




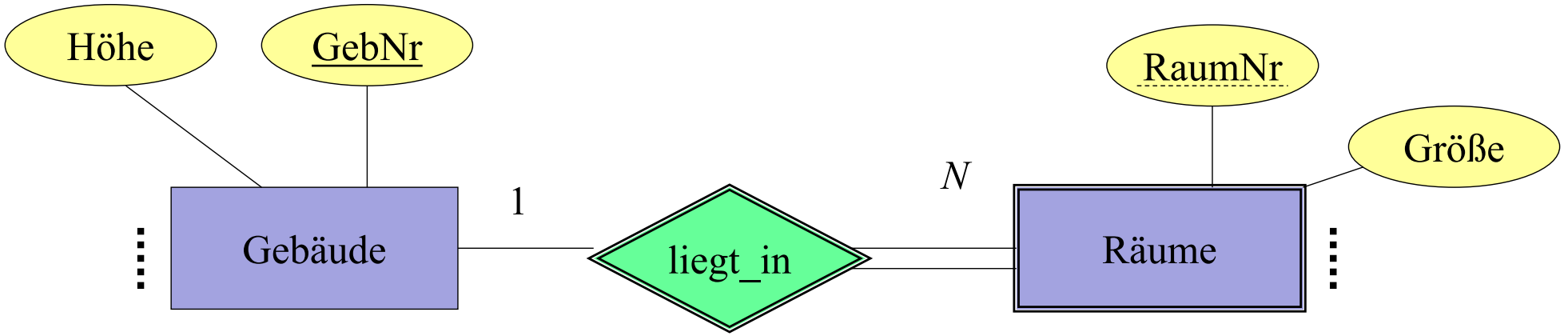
Komplex-strukturierte Entities







Schwache, existenzabhängige Entities



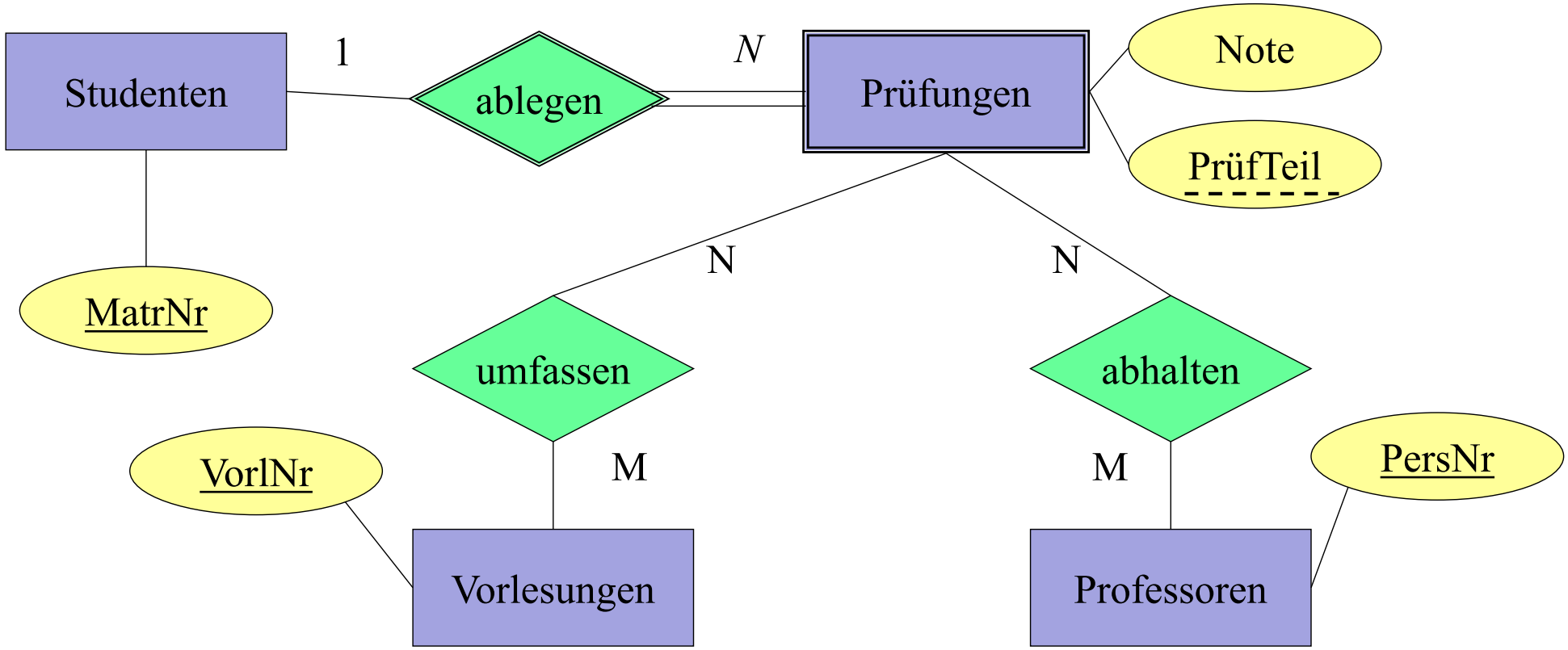
Beziehung zwischen "starken" und schwachem Typ ist immer 1:*N* (oder 1:1 in seltenen Fällen)

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

Zusammenfassung, Kernpunkte

Grundlagen von Datenbanksystemen

- Grob-Architektur eines Datenbanksystems
- Logisch-konzeptuelle Entwurfsebene: Entity-Relationship-Modell