

1. Die Bedeutung des Datenmanagements
2. Datenbank-Architektur
- 3. Modellierung und Entwurf von DB-Systemen**
4. Relationale Algebra und Normalisierung
5. Definition und Abfrage von Datenbank-Systemen
6. Dateiorganisation und Zugriffsstrukturen
7. Optimierung von Anfragen
8. Transaktionen

3. Modellierung und Entwurf von DB-Systemen

1. Definitionen

1. *Modelle*
2. *Modellierungsansatz*
3. *Datenmodellierung*

2. Die Formalisierung der Realität

1. *Formalisierungsarten*
2. *Gegenstände, Begriffe, Beziehungen*
3. *Eigenschaften*

3. Das Entity-Relationship-Modell nach Chen

1. *Allgemeines*
2. *Vorgehensweise*
3. *Entitätsklassen und Entitäten*
4. *Attribute*
5. *Domänen*
6. *Beziehungen und Beziehungstypen*
7. *Charakterisierung von Beziehungstypen*
8. *Die (min, max)-Notation*
9. *Generalisierung / Spezialisierung*

3. Modellierung und Entwurf von DB-Systemen

10. *Aggregation*

11. *Konsolidierung*

3. Das Relationale DB-Modell nach Codd

1. *Überblick*

2. *12 Regeln von Codd*

3. *Begriffsdefinitionen*

4. *Schlüssel*

5. *Abbildungsregeln*

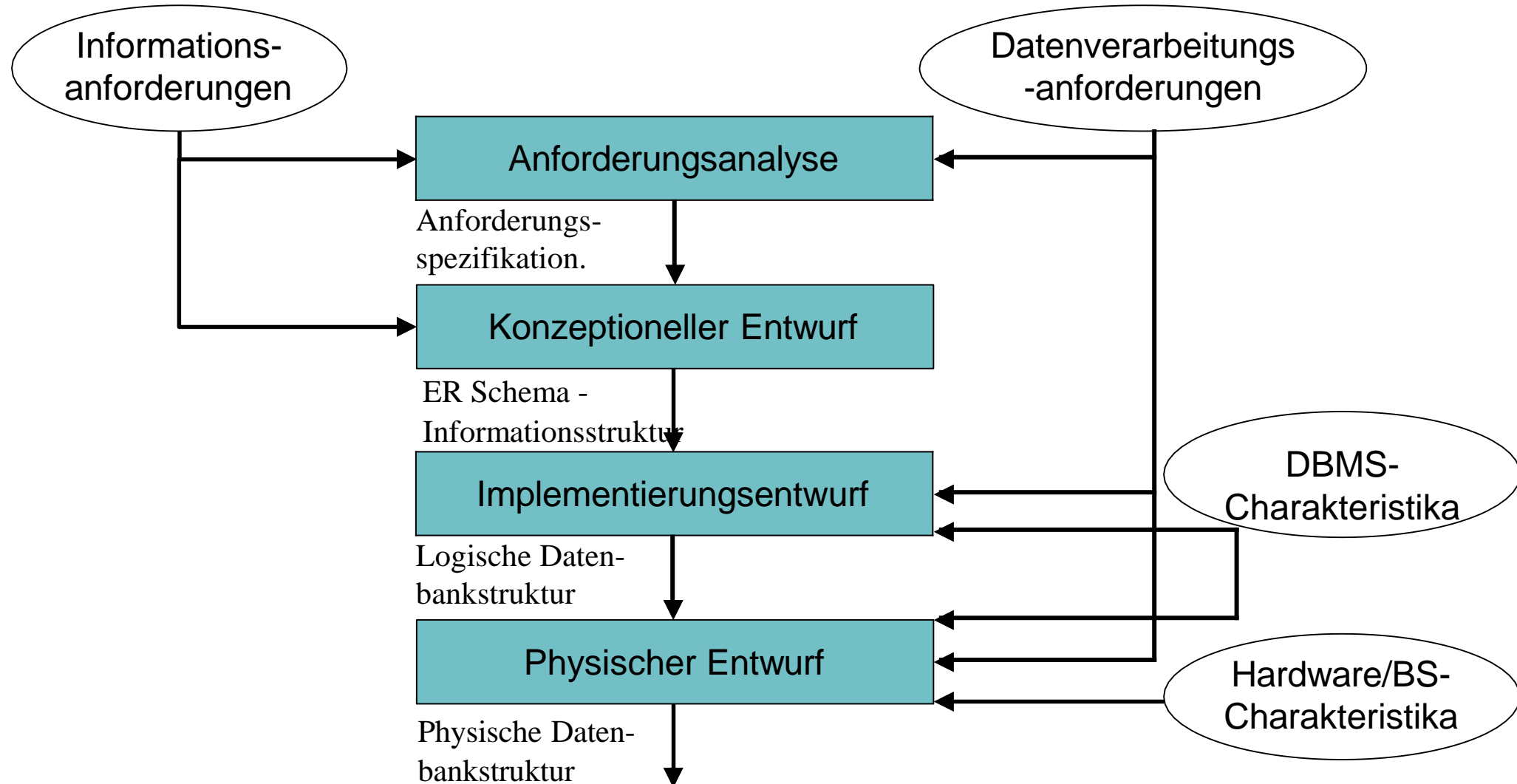
6. *Integritätsbedingungen*

Abstraktionsebenen des Datenbankentwurfs:

- 1. Konzeptuelle Ebene**
- 2. Implementationsebene**
- 3. Physische Ebene**

(in Bezug auf das 3-Ebenen-Konzept des vorherigen Kapitels betrachten wir hier nicht die externe Ebene)

3. Modellierung und Entwurf von DB-Systemen



3.1 Definitionen

- Zeitpunktbezogenes Abbild eines Realitätsausschnittes
- Ein Modell kann nie die ganze Realität abbilden
- Modelle bilden die Wirklichkeit ab durch
 - Zweckbezogene Abstraktion
 - Zweckbezogene Reduktion der Komplexität
- Modelle schaffen bessere Einsicht in die relevanten Zusammenhänge von relevanten Eigenschaften und relevanten Beziehungen von relevanten Komponenten

- **Zwischenrepräsentationen** für die Entwicklung komplexer Systeme
- **Darstellungen, Muster, oder Schemata** gegebener oder erst noch zu schaffender Phänomene
- dienen in einem gegebenen Kontext bestimmten Personen bei der **Verfolgung bestimmter Ziele und Zwecke**
- sind für **gewisse Aufgaben und innerhalb eines gewissen pragmatischen Kontextes** geschaffen
- Kombination von **Abbildungsmerkmal**, **Verkürzungsmerkmal** und **pragmatischem Merkmal**

Modellierung

- ist die methodisch geleitete Tätigkeit der Erstellung von Modellen

Modellierungsansatz

- ist eine aufeinander abgestimmte Kombination von
 - Methoden (wie ist etwas zu tun?)
 - Vorgehen (was ist wann zu tun?)
 - Werkzeugen (womit ist etwas zu tun?)

- Tätigkeit zur Strukturierung der Datenbestände
- Ziel:
 - redundanzarme, systematische Beschreibung der zur computerunterstützten Arbeit mit einem DBMS benötigten Gegenstände, Begriffe und deren Zusammenhänge
- Vorgehensweise:
 - Bestimmung der relevanten Objekte
 - Bestimmung der relevanten Eigenschaften
 - Bestimmung der Beziehungen zwischen den Objekten
 - Abbildung auf Tabellen

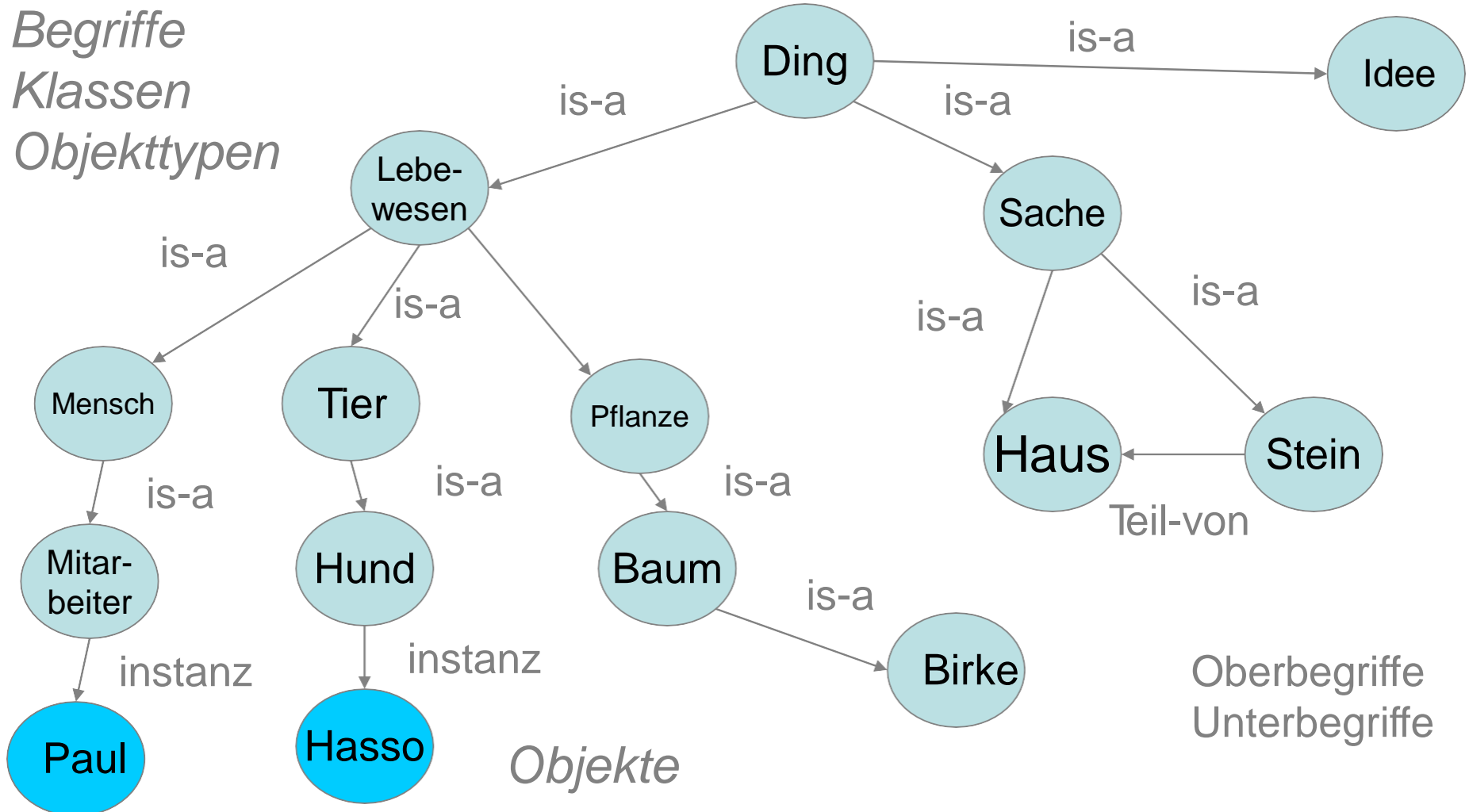
3.2 Die Formalisierung der Realität

- Formale Sprachen (Logik):
 - Alphabet (Buchstaben, Zeichen)
 - Satzbildungsregeln (Syntax)
 - Interpretation (von Sätzen in Modellen, Bedeutung, Wahrheit, Gültigkeit im Modell)
- Programmiersprachen
- Modelle (z.B. grafische Modelle)
- Theorien
- Konzepte

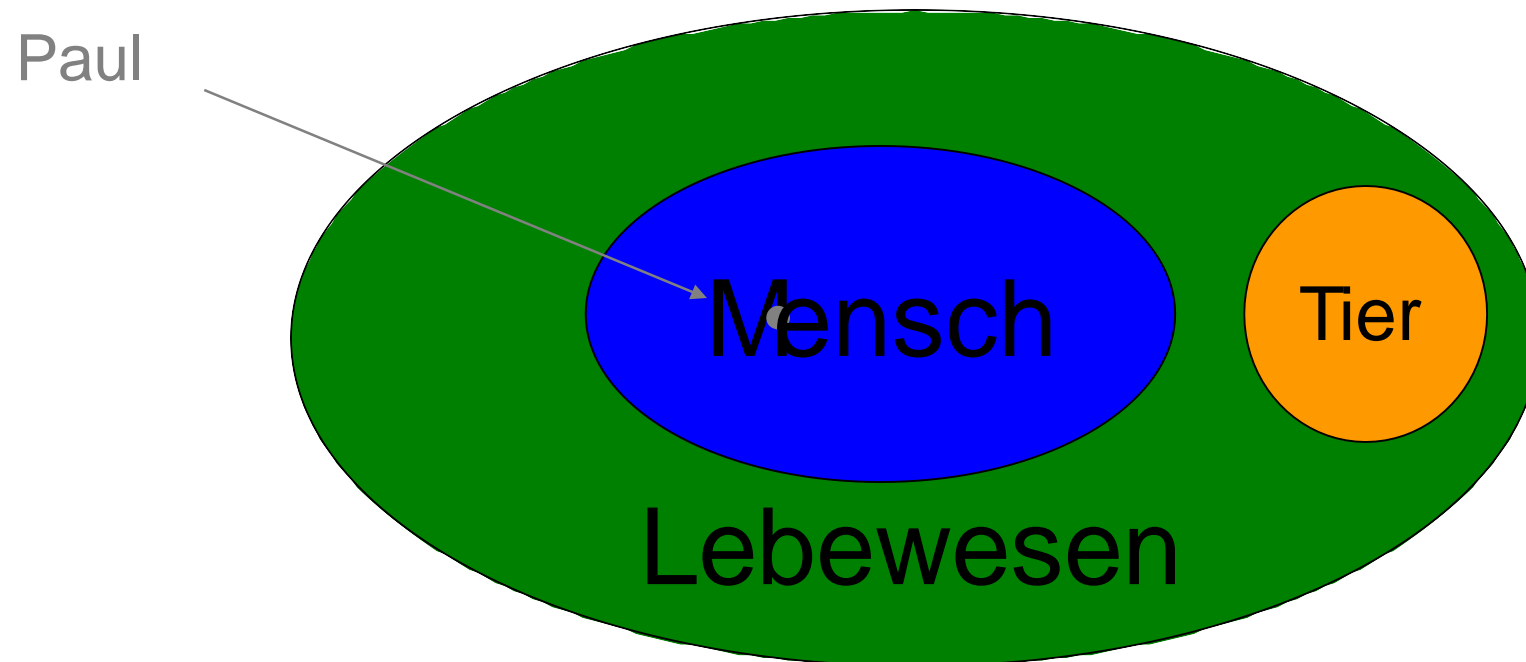
Am Anfang steht die Suche nach relevanten:

- **Gegenständen**, Individuen, Objekten
 - Eigennamen (Peter, Hasso, USA)
 - Kennzeichnungen (der Pförtner von IBM)
 - Nominalphrasen (der Chef von IBM)
- **Begriffen**, Klassen, Objekttypen, Eigenschaften, Attributen, Merkmalen (Mensch, Tier, Lebewesen, Staat)
- **Beziehungen** (z.B.: größer als, höher als)

3.2.2 Gegenstände, Begriffe, Beziehungen



„Mensch ist Teilmenge von Lebewesen“



Arten von Begriffen:

- Qualitative Begriffe bzw. klassifikatorische Begriffe
- Komparative Begriffe
- Quantitative bzw. metrische Begriffe

Denken Sie über diese 3 Punkte nach und nennen Sie einige Beispiele von qualitativen, komparativen und quantitativen Begriffen!

Eigenschaften (realer Dinge):

- Merkmal, Funktion oder Attribut das zum Wesen einer Person oder Sache gehört.
- Realisiertes Merkmal, das einer Klasse von Objekten, Prozessen, Relationen, Ereignissen einer Personengruppe gemeinsam ist und sie von anderen unterscheidet (Klassenzugehörigkeit).

Synonyme:

- verschiedene Ausdrücke, die das gleiche bezeichnen
- Beispiel:
TV-Gerät = Fernseher, Apfelsine = Orange, Pferd = Gaul

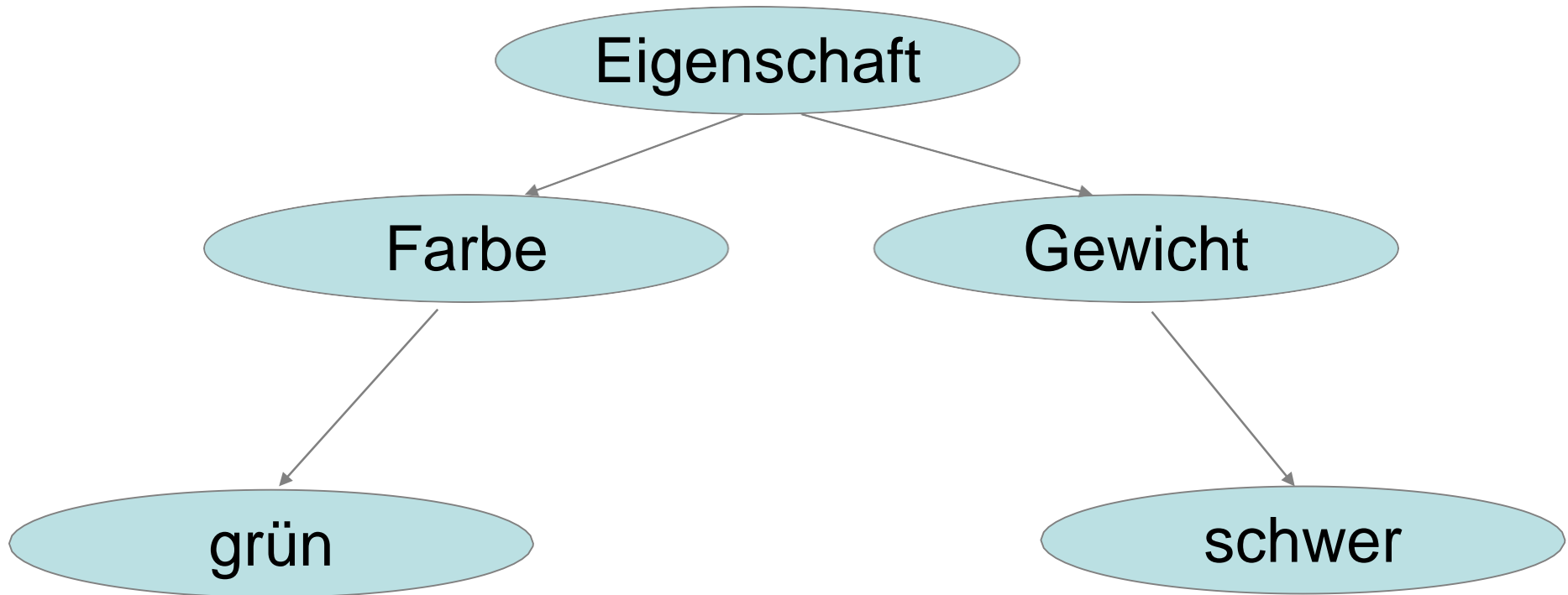
Homonyme:

- gleiche Ausdrücke, die etwas verschiedenes bezeichnen (kontextabhängig)
- Beispiel:
Bank = Geldinstitut oder Sitzgelegenheit
Schläger = Drumstic oder Krimineller
Schwiegermuttersessel = Sitzgelegenheit oder Pflanze

Darstellung in einer Tabelle

Objektname	Farbe	Gewicht
Bank1	grün	schwer

Hierarchische Darstellung



ERM oder ER-Modell

(ab und zu auch ERD genannt)

„Entity Relationship Model“

- Das Entity-Relationship-Modell (ERM) ist das populärste Entwurfs-Hilfsmittel für Datenstrukturen.
- geht auf eine von Peter Pin-Shan Chen im Jahre 1976 veröffentlichte Arbeit zurück
- Standard-Modellierungstechnik im Datenbank-Bereich
- Entwurfshilfe für die Datenmodellierung
- optimal für die Modellierung des konzeptionellen Schemas für relationale DB-Systeme
- Modellierungen im ERM können „automatisch“ in ein relationales Modell transformiert werden

- Das ERM ist nicht (nur) eine Visualisierungstechnik, sondern unterstützt mächtigere, aussagekräftigere Konstrukte.
- Das ursprüngliche ERM von Chen wurde vielfältig adaptiert und weiterentwickelt. Varianten und Erweiterungen sind u.a.
 - Binäre ERM
 - Erweiterte ERM (EERM)
 - Strukturierte ERM (SERM)
 - IDEF1X
- Es gibt keine Normierung der graphischen Darstellung des ERM → sehr viele unterschiedliche graphische Notationen.
Wir verwenden die Einfachsten.

Suchen der Objekte:

Objekt	Identität mit Eigenschaften
Objektklasse	Ansammlung von Objekten mit gleichen Eigenschaften

Definition der Objekte:

MITARB (Franz)

Gehalt, Gebdat, angestellt in Abteilung, Alter

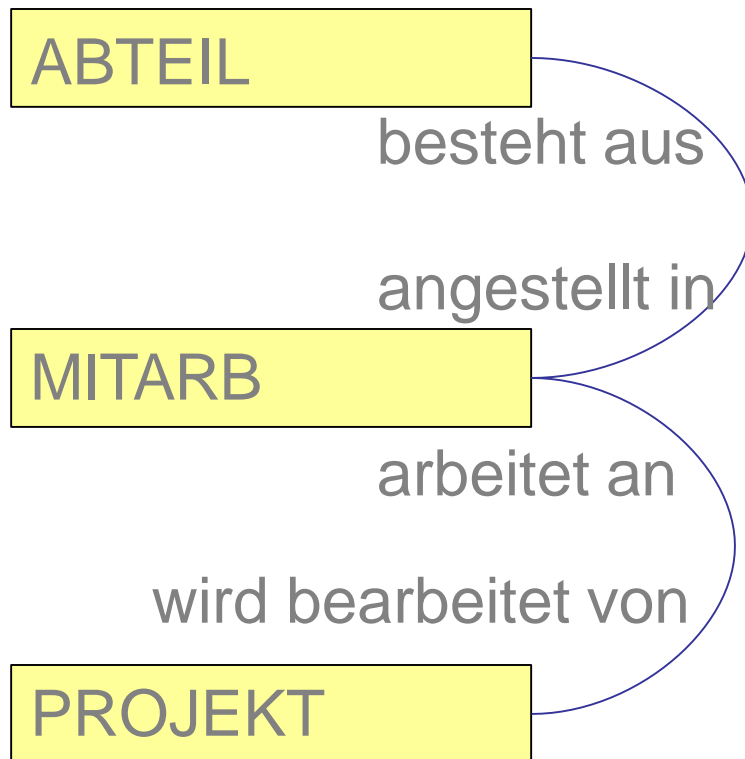
Definition der Klassen:

ABTEIL

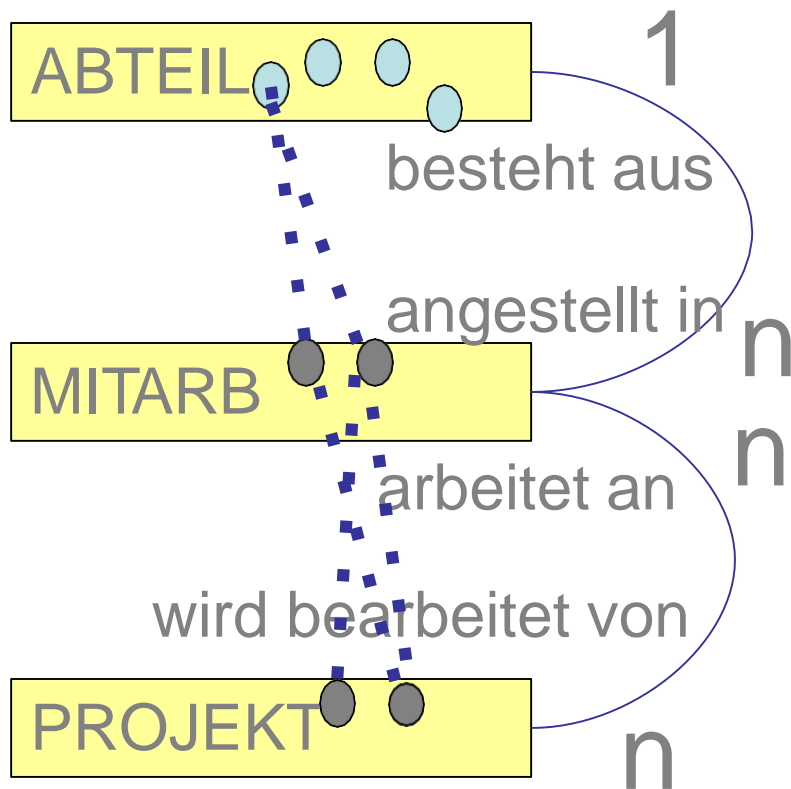
MITARB

PROJEKT

Definition der Beziehungen:



Definition der Beziehungsarten:



„Das Starterpaket“

Entitäts-Klasse (Entity Type):

- Klasse von realen oder ideellen Objekten (d.h. Objekten mit physischer oder begrifflicher Existenz) mit gleichen Attributen, denen der Datenbank-Betreiber eine selbstständige Existenz zuschreibt. Beispiel: „Mitarbeiter“, „Urheber“, „Kunden“, „Produkte“, „Personen“.

Graphische Notation

Rechteck: 

Beispiel: Entitäts-Klassen

Kunden

Mitarbeiter

Produkte

Entität (Entity, Pl.: Entities):

- Reales oder abstraktes „Etwas“, das für einen betrachteten Realweltausschnitt von Interesse ist.
- ein reales oder ideelles Objekt, das einzig ist, d.h. ein individuelles Objekt eines Entity Types / einer Entitäts-Klasse, z.B. „Paul Müller“ für Entitätsklasse „Autor“.

Attribute (Attributes):

- beschreiben eine Entität, d.h. eine definierte Kombination von Attributwerten dient der Beschreibung einer Entität (z.B.: „Paul Müller“) und ferner einer Entitäts-Klasse
- Können auch Beziehungen beschreiben

Graphische Notation

Oval:



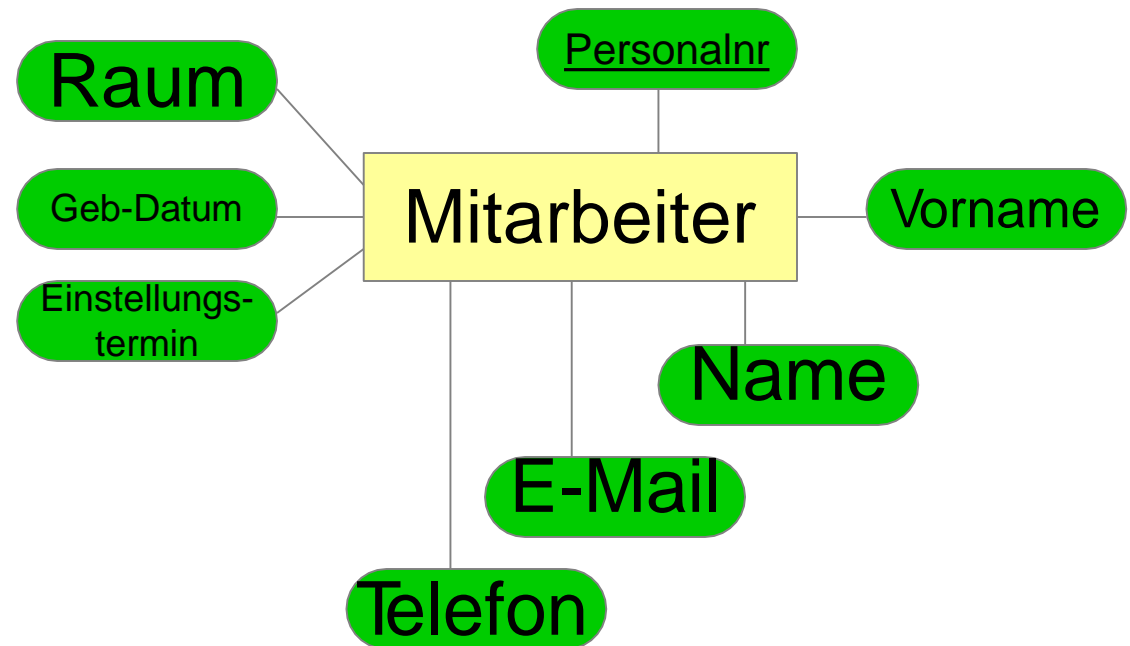
Attribute Name

Schlüssel-Attribut (Primary Key):

- das ein-eindeutige Attribut, mit dem man eine individuelle Entität eindeutig identifiziert (z.B. „Identifikationsnummer“)
- Schlüssel-Attribute werden durch Unterstreichung gekennzeichnet und in der textuellen Notation z.B. wie folgt dargestellt:

Mitarbeiter (Personalnr, Name, Vorname, Gebdatum)

Beispiel: Mitarbeitereigenschaften



Textuelle Notation:

Mitarbeiter(PersonalNr, Vorname, Name, E-Mail, Telefon, Raum, Geb-Datum, Einstellungstermin)

Domänen (Attribute Domain):

- Zulässiger Wertebereich eines Attributes
- Bereich der Werte, die ein Attribut annehmen kann (z.B. nur Zahlen größer als 0 für „Gewicht“ oder nur Ziffern und „X“ für „ISBN“)
- Standard-Datentypen für Domänen: int, string, date

Graphische Notation

Oval:



Attribute Name: Domain Name

Textuelle Notation (allgemein):

$E(A_1:D_1, \dots, A_n:D_n)$ (mit Domänen)

E : Entity Type

A_i : Attribute $i = 1 \dots n$

D_i : Domänen $i = 1 \dots N$

Kurzform:

$E(A_1, \dots, A_n)$ (ohne Domänen)

Beispiel: Angestellter

Die **Entitäts-Klasse** ANGESTELLTER:

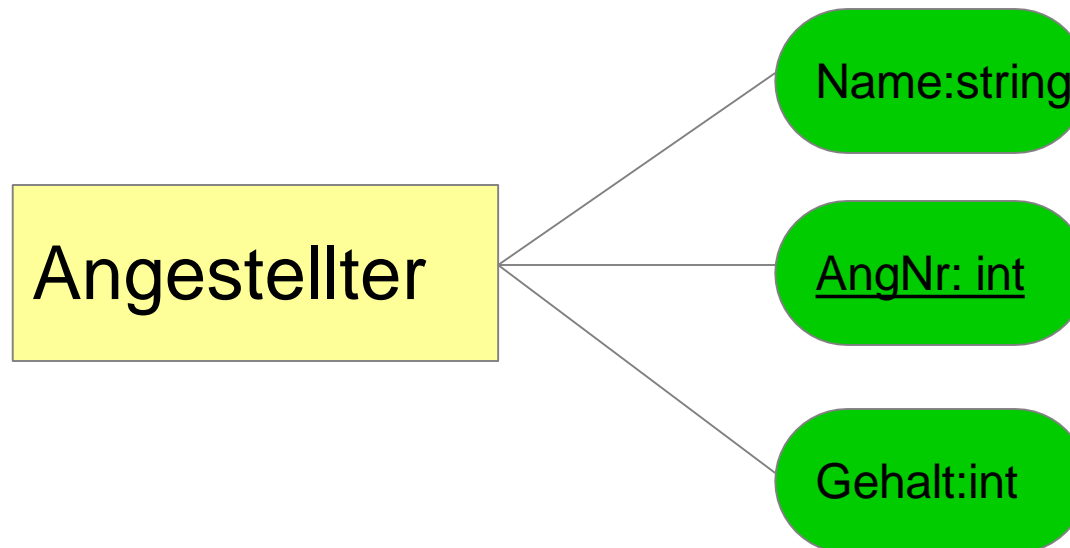
Angestellter(AngNr: int, Name: string, Gehalt: int)

Entitäten vom Typ ANGESTELLTER:

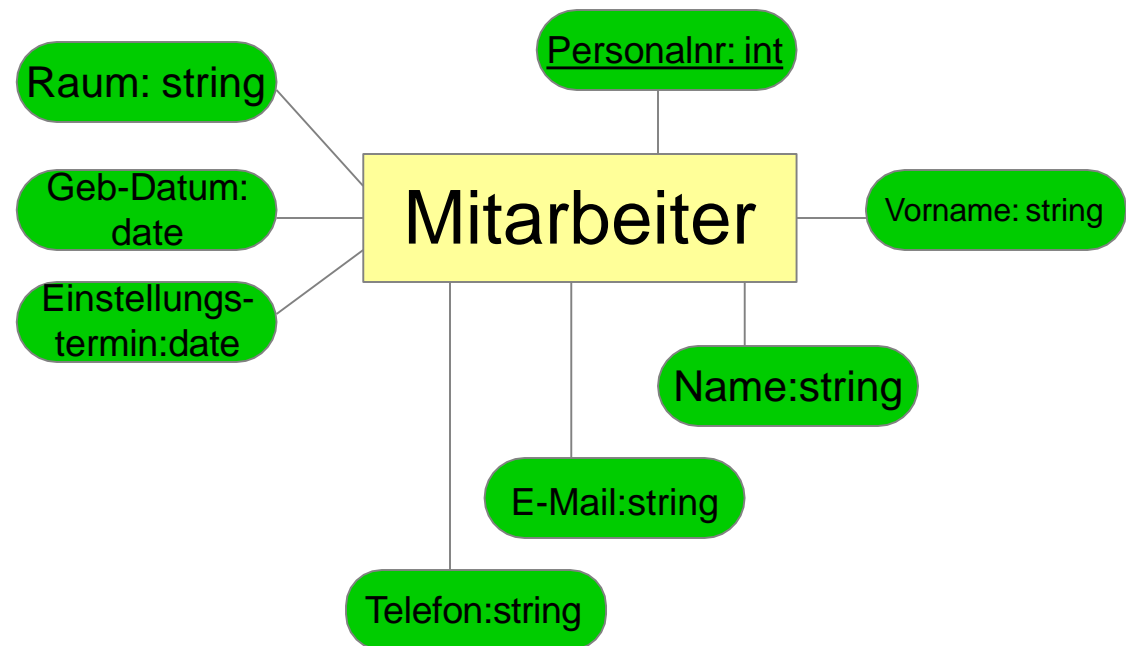
Angestellter(523, 'Bill', 90.000)

Angestellter(122, 'John', 50.500)

Graphische Notation:



Beispiel: Mitarbeitereigenschaften mit Wertebereichen



Textuelle Notation:

Mitarbeiter(Personalnr: int, Vorname:string, Name:string, E-Mail:string, Telefon:string, Raum:string, Geb-Datum:date, Einstellungstermin:date)

Beziehung (Relationship):

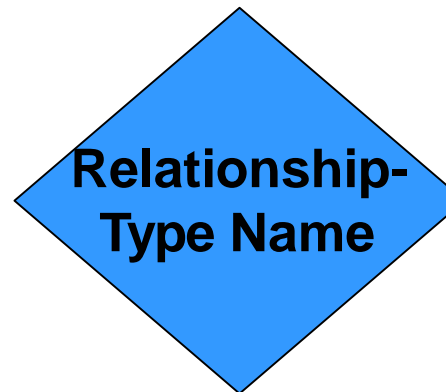
- Logische Verknüpfung oder Beziehung zwischen zwei oder mehreren Entitäten. Beispiel: „verheiratet mit“: „Paul Müller“ und „Maria Müller“.

Beziehungstypen (Relationship-Type):

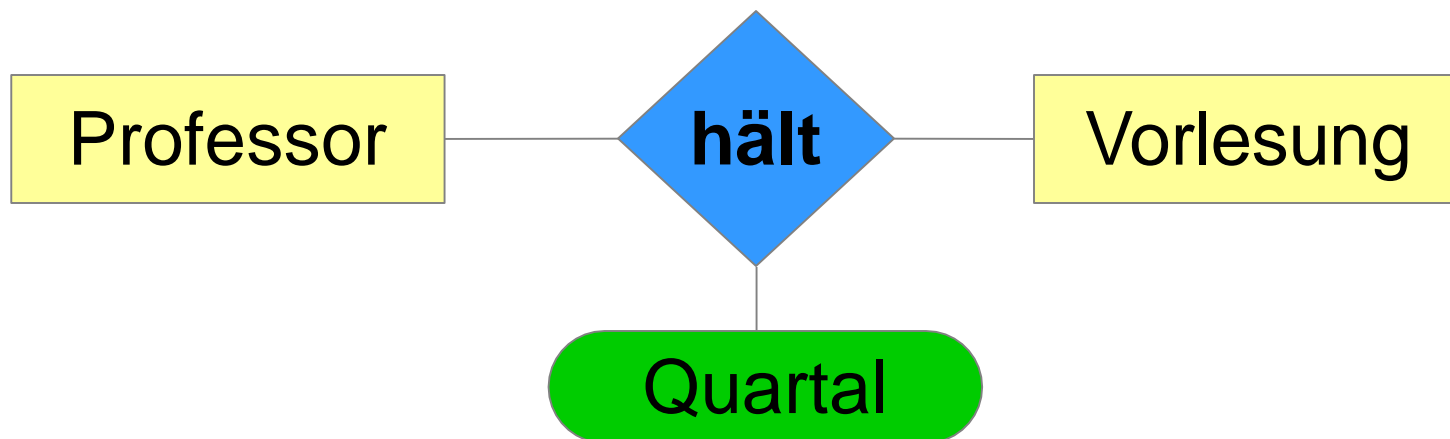
- Formale Struktur einer Beziehung einschließlich Wertebereiche der die Beziehung beschreibenden Attribute. Beispiel: zwischen „Kunden“ und „Produkten“.

Graphische Notation

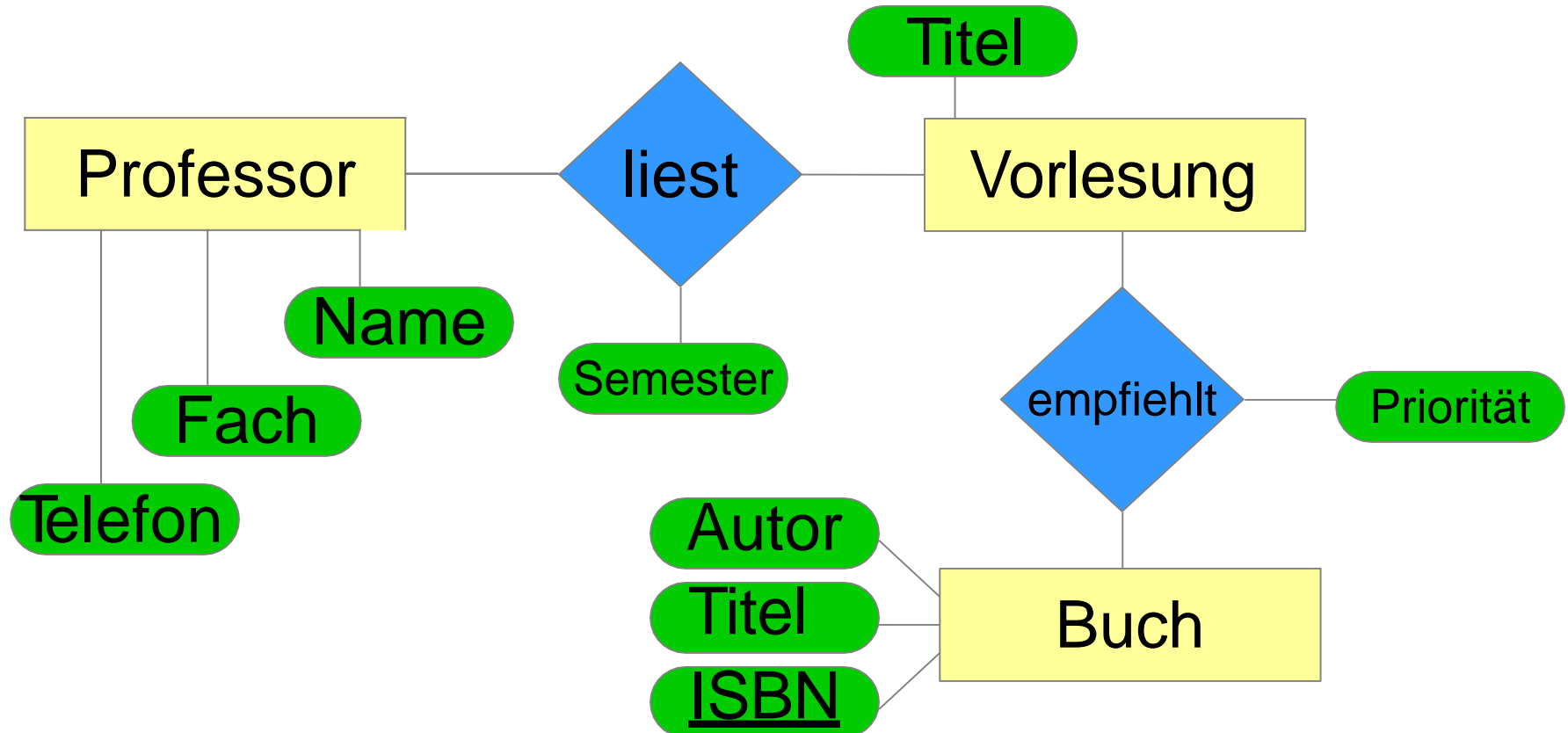
Raute:



Beispiel: Professor hält Vorlesung in einem bestimmten Quartal



Beispiel: Buchempfehlung



Aufgabe: Bibliothek

- **Stellen Sie folgenden Sachverhalt im E/R-Modell dar:**
Bücher werden von Autoren geschrieben. Diese Bücher können von Lesern ausgeliehen werden. Hierfür muss das Ausgabe- und das Rückgabedatum festgehalten werden. Jeder Leser erhält eine Lesernummer. Außerdem werden Name und Adresse festgehalten. Bücher haben eine Inventarnummer, eine ISBN-Nummer, einen Titel und einen bestimmten Verlag. Von den Autoren soll lediglich ein eindeutiger Schlüssel und der Name gespeichert werden.

„Das Komplettpaket“

Einfaches Attribut (Simple Attribute):

- ein Attribut, das aus einem einzigen unabhängigen Bestandteil besteht (z.B. „Geburtsjahr“).

Zusammengesetztes Attribut (Composite Attribute):

- ein Attribut, das mehrere unabhängige Komponenten hat (z.B. „Lebensdaten“ bestehend aus „Geburtsjahr“ und „Todesjahr“; „Name“ bestehend aus „Vornamen“ und „Nachnamen“).

Einwertiges Attribut (Single-valued Attribute):

- ein Attribut, das für jede Entität jeweils nur einen einzigen Wert (aus dem Wertbereich) annehmen kann (z.B. „Geburtsdatum“: jede Person ist nur einmal geboren)

Mehrwertiges Attribut (Multi-valued Attribute):

- ein Attribut, das für eine einzelne Entität mehrere Werte gleichzeitig haben kann (z.B. „E-Mail-Adresse“: eine Person kann mehrere E-Mail-Adressen haben)

Abgeleitetes Attribut (Derived Attribute):

- ein Attribut, dessen Wert aus anderen Attributen derselben oder anderer Entitäten abgeleitet werden kann (z.B. „Alter“ aus „Geburtsdatum“)
- Abgeleitete Attribute werden in der Regel nicht in der Datenbank gespeichert
- Attribute können auf unterschiedlichste Arten abgeleitet werden. Dies wird nicht aus dem ER-Diagramm ersichtlich.

Optional undefinierte Attribute

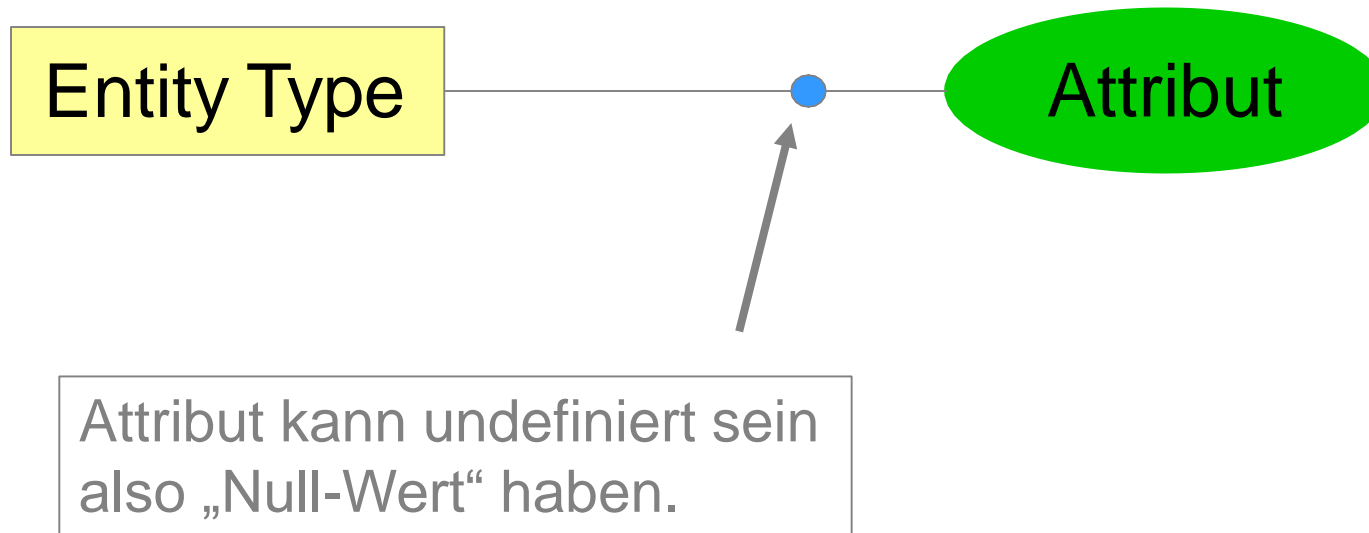
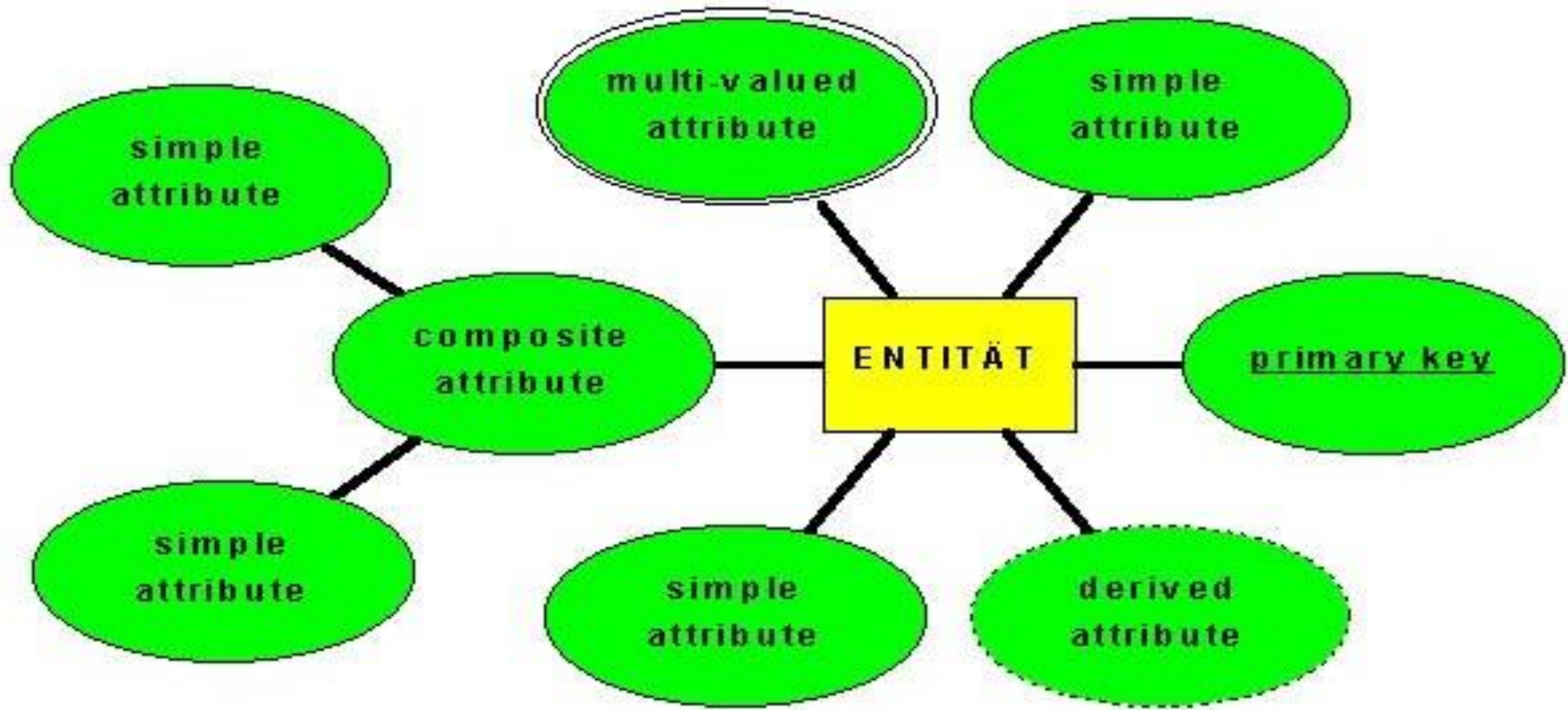


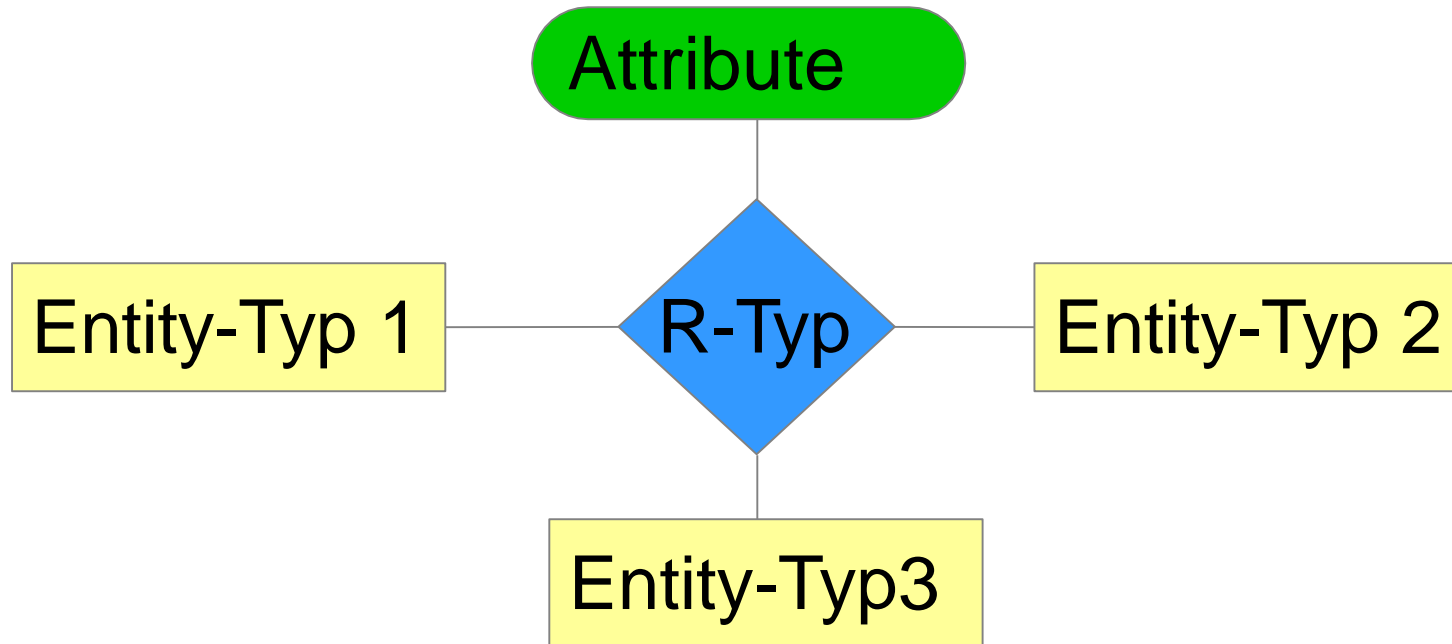
Diagramm-Symbole für Attribute



- Aufgabe: Erarbeiten Sie sich verschiedene Attributarten für den Entitätstypen „Mitarbeiter“ oder einem Entitätstypen ihrer Wahl. Setzen Sie sich dazu in Gruppen zusammen und präsentieren Sie später Ihre Lösung mit den verfügbaren Medien.

Grad einer Beziehung (Degree of a Relationship):

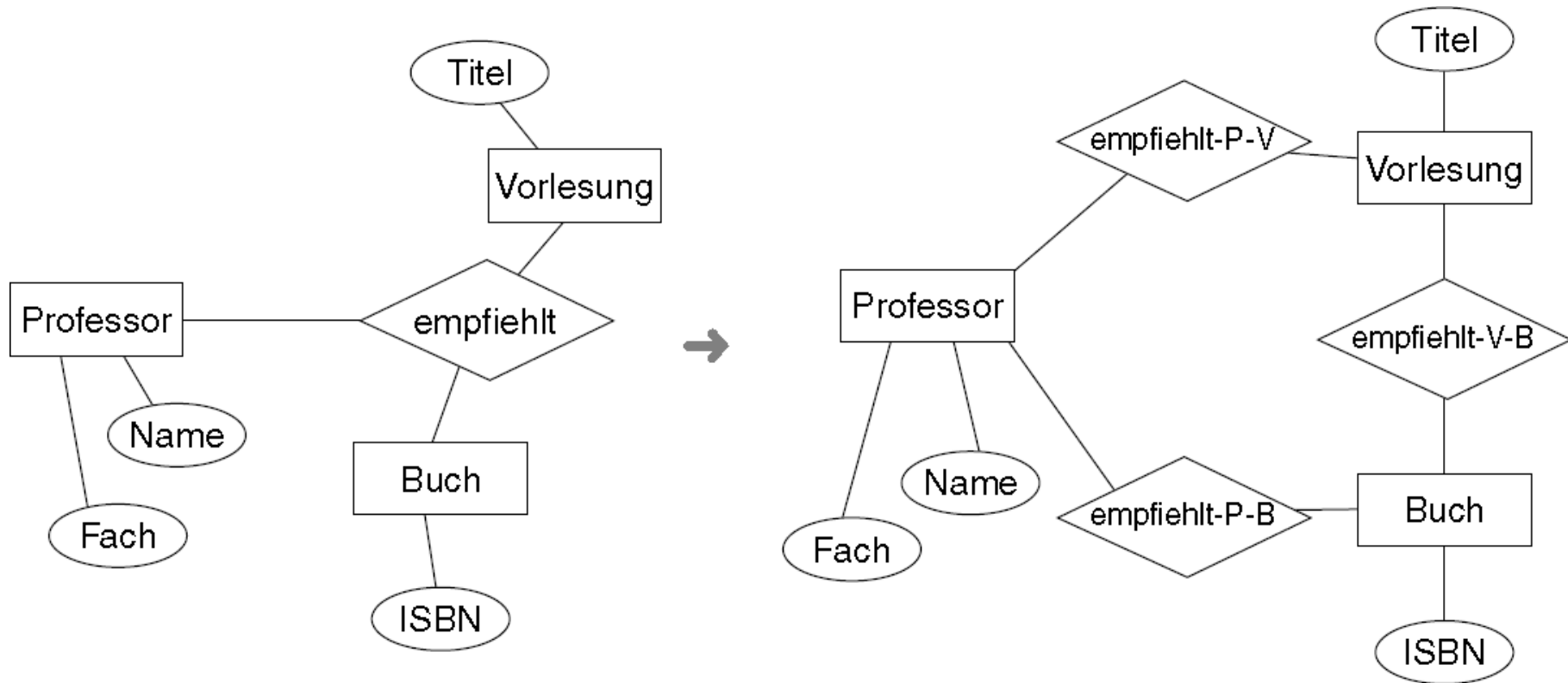
- Anzahl der Entitäts-Klassen, die an einer Beziehung beteiligt sind (z.B. Beziehung dritten Grades: z.B. bei „liefert“ sind beteiligt: „Produkt“, „Kunde“ und „Hersteller“)
- Im Gegensatz zu **mehrwertigen Beziehungstypen** gibt es auch **rekursive Beziehungstypen**, an denen nur eine Entitäts-Klasse beteiligt ist (z.B. bei „voraussetzen“ sind beteiligt: „Vorlesungen“ (als Vorgänger und Nachfolger))



- Die Darstellung repräsentiert eine dreistellige Beziehung.
- Möglich sind beliebig viele Stellen.
- Am häufigsten sind zweistellige (binäre) Beziehungen.
- Zusätzlich können Beziehungstypen auch Attribute zugeordnet werden

3.3.6 Beziehungen und Beziehungstypen (Teil 2)

Frage: Kann man jede ternäre Beziehung einfach in drei binäre Beziehungen umwandeln?



3.3.6 Beziehungen und Beziehungstypen (Teil 2)

Frage: Kann man jede ternäre Beziehung einfach in drei binäre Beziehungen umwandeln?

ursprünglicher Zustand:

empfiehl		
Professor	Vorlesung	Buch (ISBN)
Pearl	Datenbanken	0-341...
Pearl	IT-Systeme	2-305...
Graham	Datenbanken	2-305...
Graham	IT-Systeme	2-305...



Zustände der drei zweistelligen Beziehungen:

empfiehl-P-V	
Professor	Vorlesung
Pearl	Datenbanken
Pearl	IT-Systeme
Graham	Datenbanken
Graham	IT-Systeme

empfiehl-P-B	
Professor	Buch (ISBN)
Pearl	0-341...
Pearl	2-305...
Graham	2-305...

empfiehl-V-B	
Vorlesung	Buch (ISBN)
Datenbanken	0-341...
IT-Systeme	2-305...
Datenbanken	2-305...

3.3.6 Beziehungen und Beziehungstypen (Teil 2)

Antwort: Bei der Umkehrung dieser Zerlegung können zusätzliche – von der ursprünglichen Bedeutung nicht vorgesehene - Tupel entstehen → **Eigenschaft der Verlustlosigkeit**

Verloren gegangen ist die Information, dass Pearl das Buch 2-305 für IT-Systeme, aber nicht für Datenbanken empfiehlt

empfiehl-P-V	
Professor	Vorlesung
Pearl	Datenbanken
Pearl	IT-Systeme
Graham	Datenbanken
Graham	IT-Systeme

empfiehl-P-B	
Professor	Buch (ISBN)
Pearl	0-341...
Pearl	2-305...
Graham	2-305...

empfiehl-V-B	
Vorlesung	Buch (ISBN)
Datenbanken	0-341...
IT-Systeme	2-305...
Datenbanken	2-305...




empfiehl		
Professor	Vorlesung	Buch (ISBN)
Pearl	Datenbanken	0-341...
Pearl	IT-Systeme	2-305...
Graham	Datenbanken	2-305...
Graham	IT-Systeme	2-305...
Pearl	Datenbanken	2-305...

Schwache Entitäts-Klasse (Weak Entity Type):

- Schwache Entitäts-Klassen haben keinen Primärschlüssel
- Ihre Existenz hängt von einer anderen Entitäts-Klasse ab und ist nur in Kombination mit dem Schlüssel der übergeordneten Entitäts-Klasse eindeutig identifizierbar.
- Beispiel: „Angehörige“ gibt es nur, wenn es eine Entität gibt, zu denen sie verwandt sind, und sie sind nur über die zugeordnete Entität identifizierbar

Graphische Notation

Gerahmtes Rechteck:



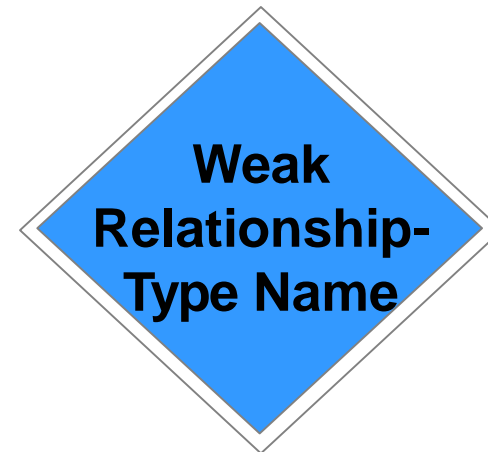
Weak Entity Type Name

Schwacher Beziehungstyp (Weak Relationship-Type):

- Eine schwacher Beziehungstyp setzt eine schwache Entitäts-Klasse in Beziehung zu einer starken Entitäts-Klasse
- (Eine starke Entitäts-Klasse ist eine mit Primärschlüssel)

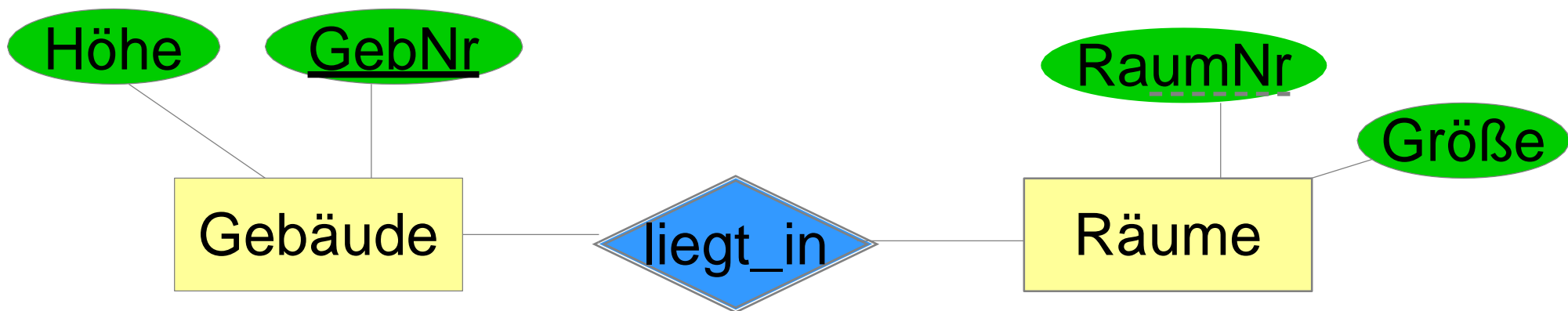
Graphische Notation

Gerahmte Raute:

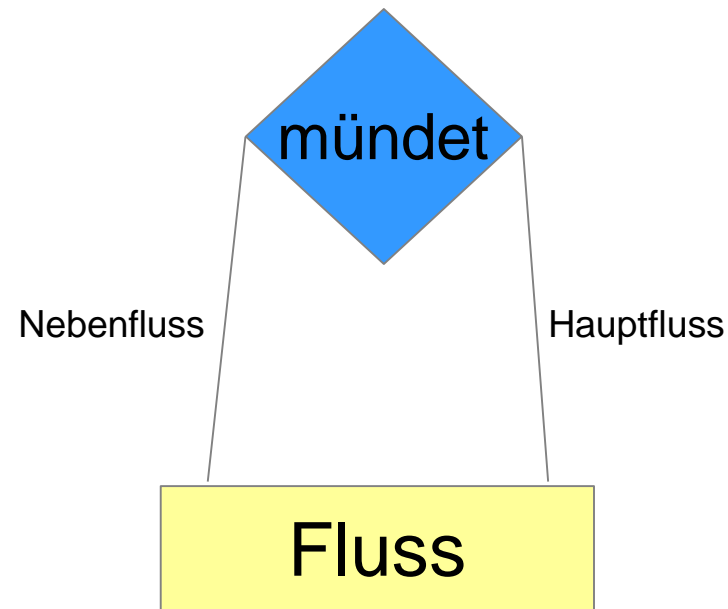


Schwacher Beziehungstyp (Weak Relationship Type):

Beispiel:



Rekursive Beziehungstypen



Bei rekursiven Beziehungstypen sollte ggf. ein Rollenname angegeben werden, um zu verdeutlichen, welche Rolle die beteiligten Entitäten hierbei einnehmen.

Charakterisierung von Beziehungstypen

Ein binärer Beziehungstyp R zwischen den Entitätstypen E_1 und E_2 heißt

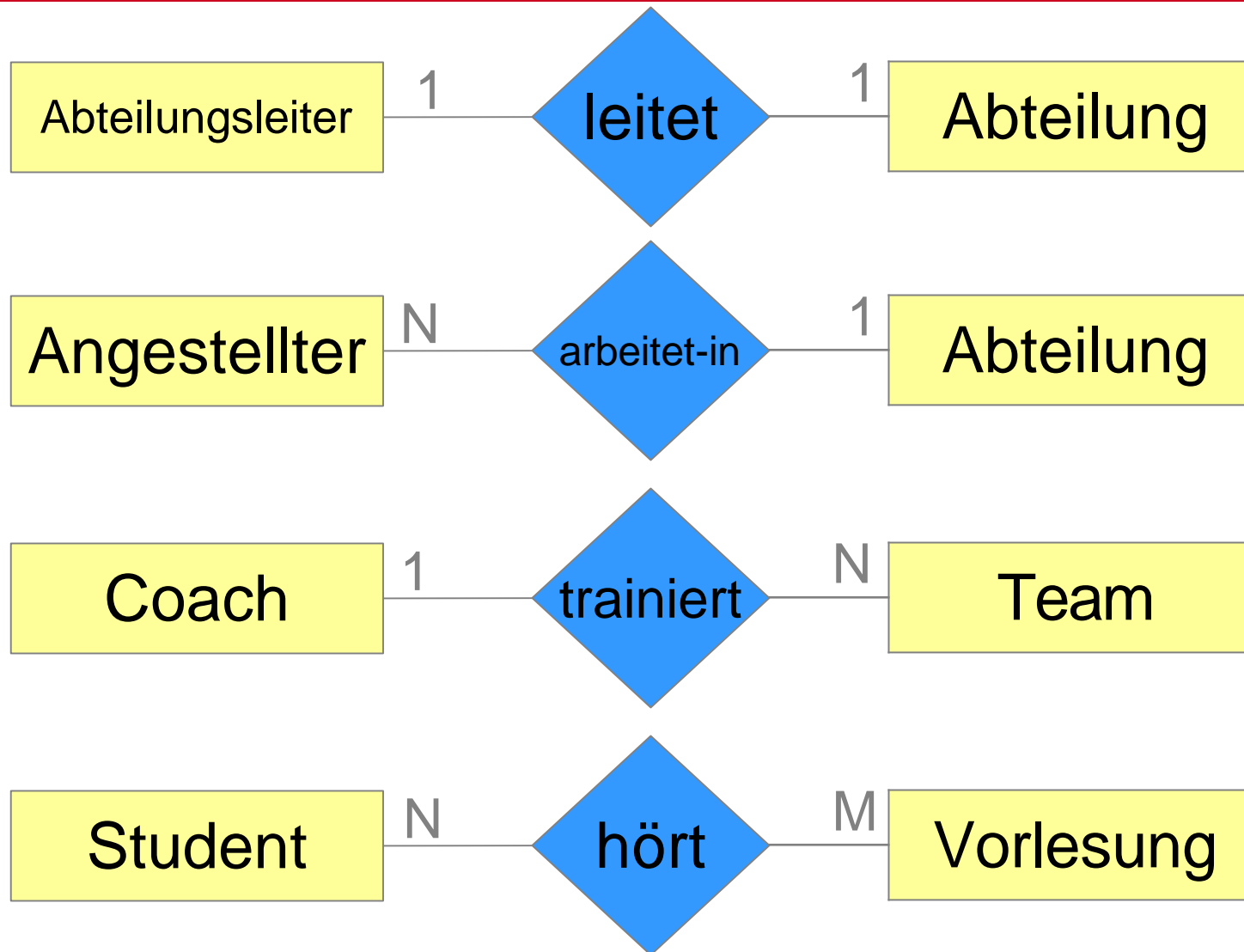
- **1:1-Beziehung (one-to-one)**, falls jeder Entität e_1 aus E_1 höchstens eine Entität e_2 aus E_2 zugeordnet ist und umgekehrt jeder Entität e_2 aus E_2 höchstens eine Entität e_1 aus E_1 zugeordnet ist. Beispiel: „verheiratet_mit“
- **1:N-Beziehung (one-to-many)**, falls jeder Entität e_1 aus E_1 beliebig viele (also keine oder mehrere) Entitäten aus E_2 zugeordnet sind, aber jeder Entität e_2 aus E_2 höchstens eine Entität e_1 aus E_1 zugeordnet ist. Beispiel: „beschäftigen“

- ***N:1-Beziehung (many-to-one)***, falls analoges zu obigem gilt.
Beispiel: „beschäftigt_bei“
- ***N:M-Beziehung (many-to-many)***, wenn keinerlei Restriktionen gelten, d.h. jede Entität aus E_1 kann mit beliebig vielen Entitäten aus E_2 in Beziehung stehen und umgekehrt kann jede Entität e_2 aus E_2 mit beliebig vielen Entitäten aus E_1 in Beziehung stehen. Beispiel: „befreundet_mit“
- In jeder dieser Beziehungstypen ist es möglich, dass keine Zuordnung von Entitäten statt findet.

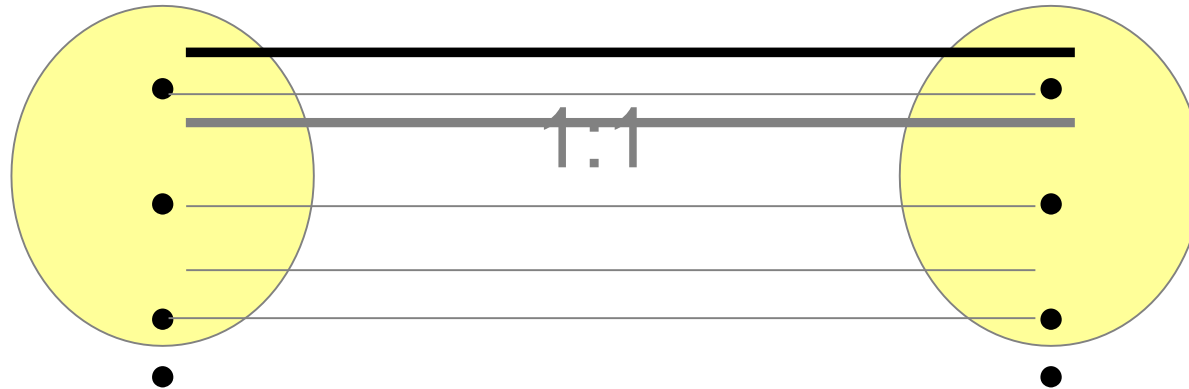
Beispiele:

- Abteilungsleiter leitet Abteilung: 1:1-Relationship
- Angestellter arbeitet in Abteilung: N:1-Relationship
- Coach trainiert ein Team : 1:N-Relationship
- Student hört Vorlesung: N:M-Relationship

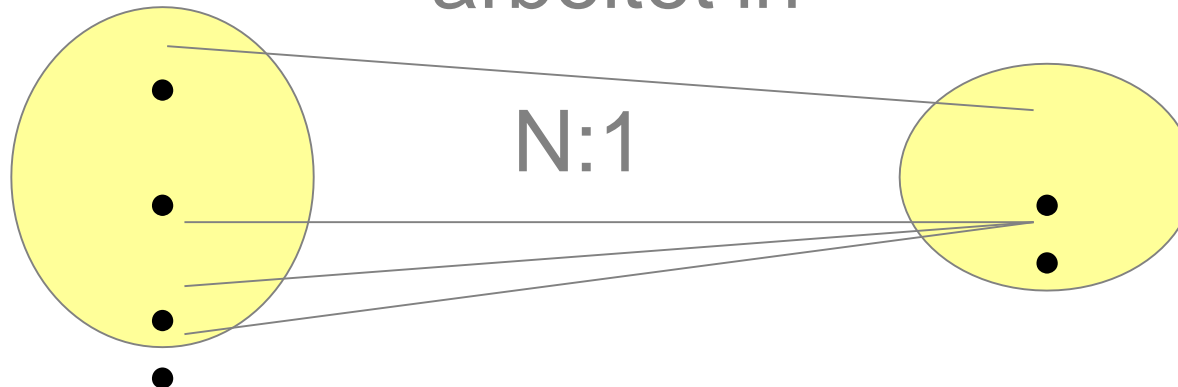
3.3.7 Charakterisierung von Beziehungstypen



Abteilungsleiter leitet Abteilung



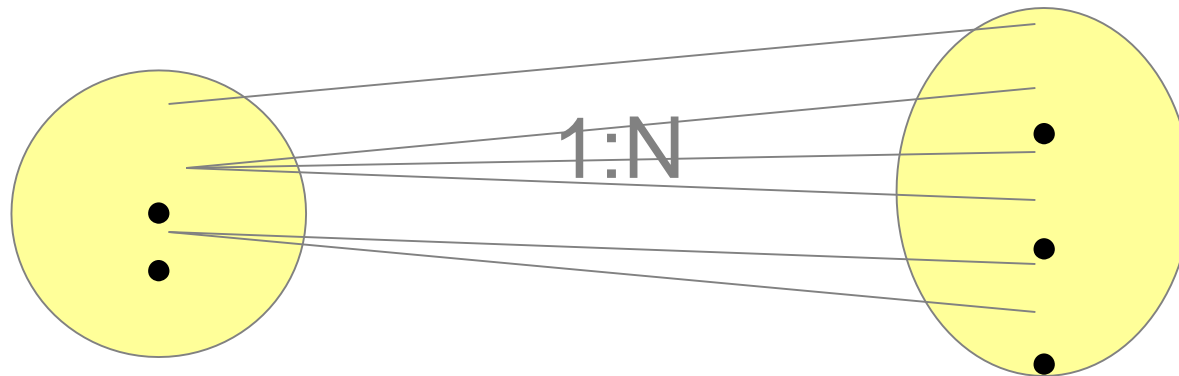
Angestellter arbeitet in Abteilung



Coach

trainiert

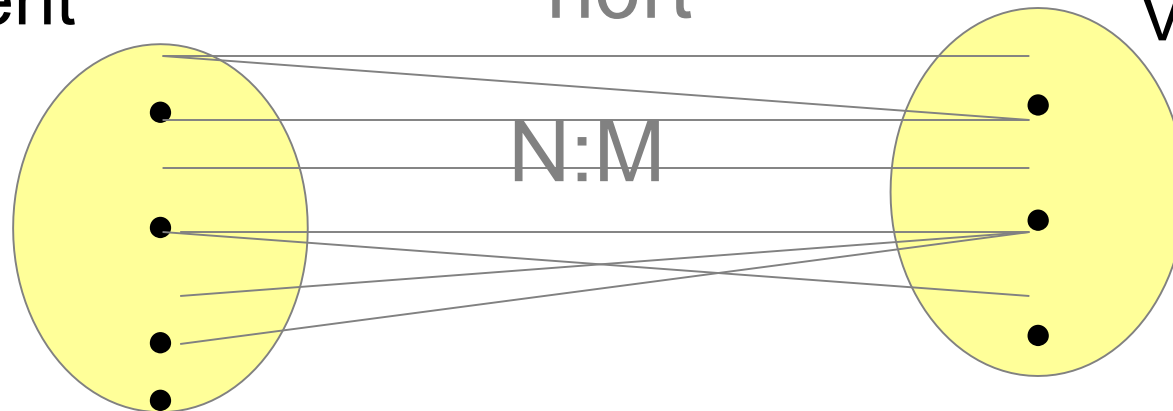
Team



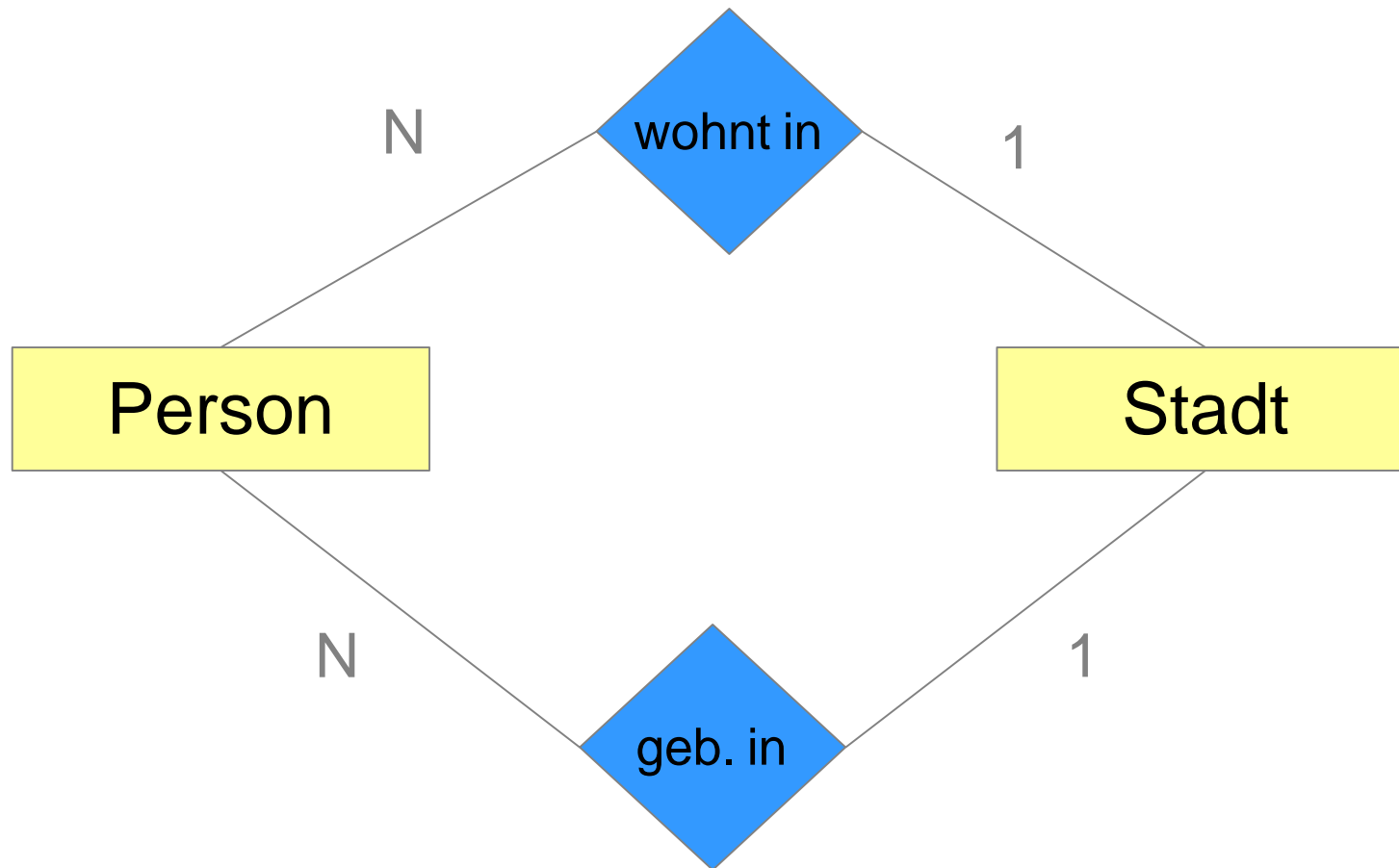
Student

hört

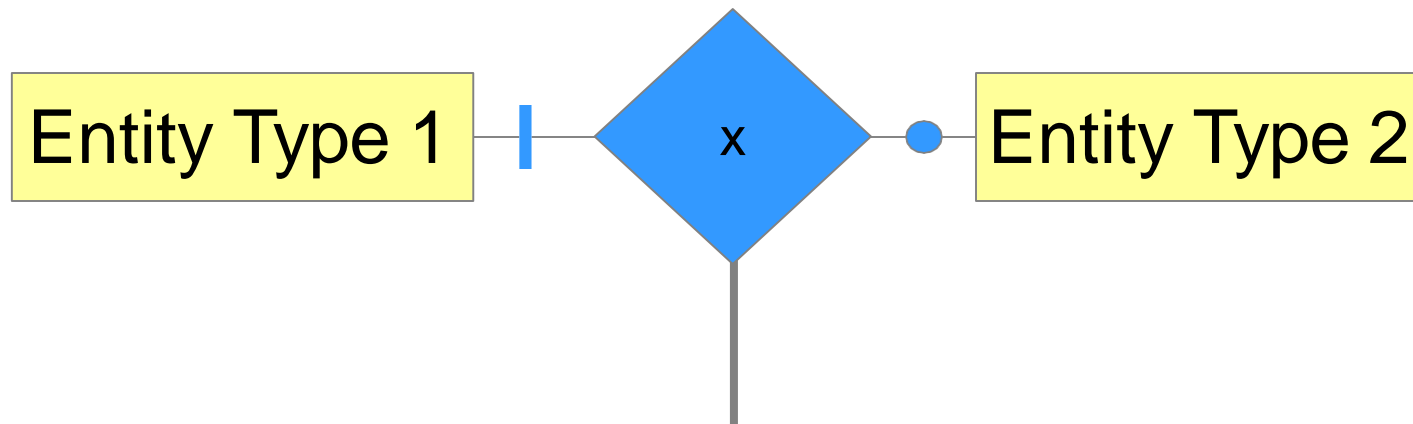
Vorlesung



3.3.7 Charakterisierung von Beziehungstypen

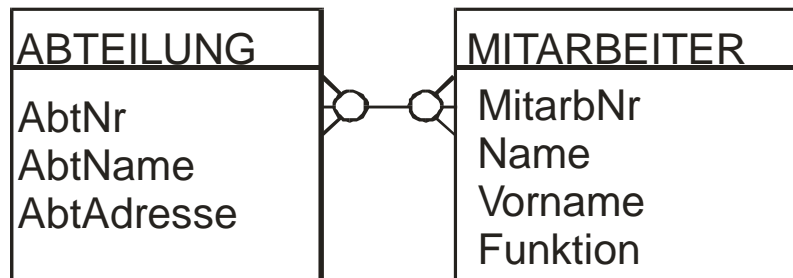


Optionalität einer Relation

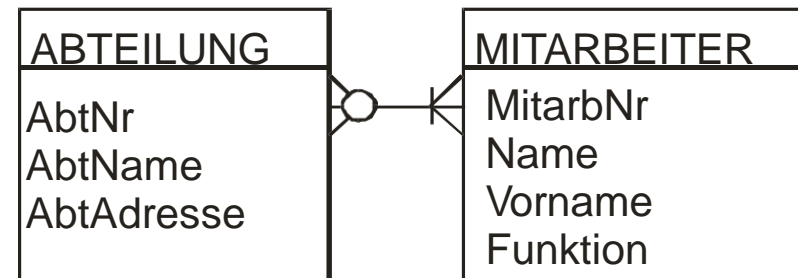


Die Beziehung zwischen zwei Entity-Types sind entweder obligatorisch oder optional.
Bsp.: Zwei Entity-Types: „Kundenkartei“ und „Rechnung“, Ein Kunde kann eine, mehrere oder auch KEINE Rechnung offen haben, während jeder Rechnung genau einem Kunden zugeordnet sein muss.

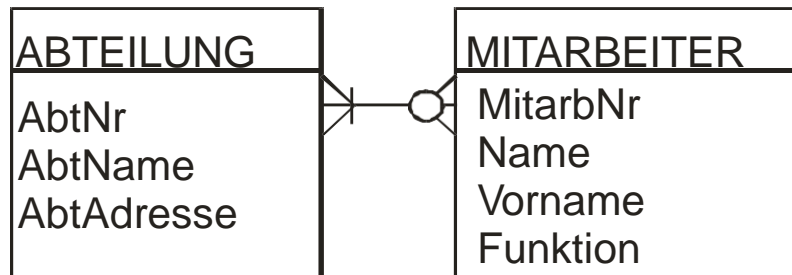
Alternative Darstellung mit der „Krähenfußnotation“



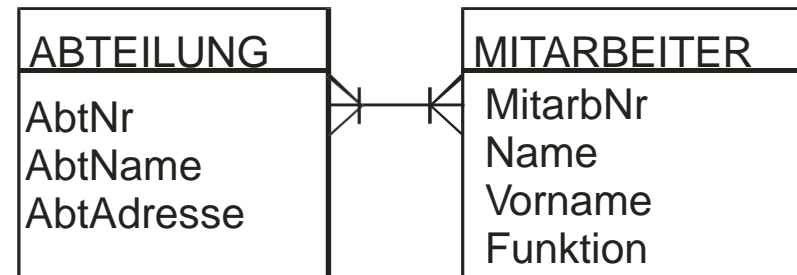
Jede Abteilung kann beliebig viele Mitarbeiter haben, jeder Mitarbeiter kann zu beliebig vielen Abteilung gehören.



Jede Abteilung muss mind. einen Mitarbeiter haben, jeder Mitarbeiter kann zu beliebig vielen Abteilung gehören.

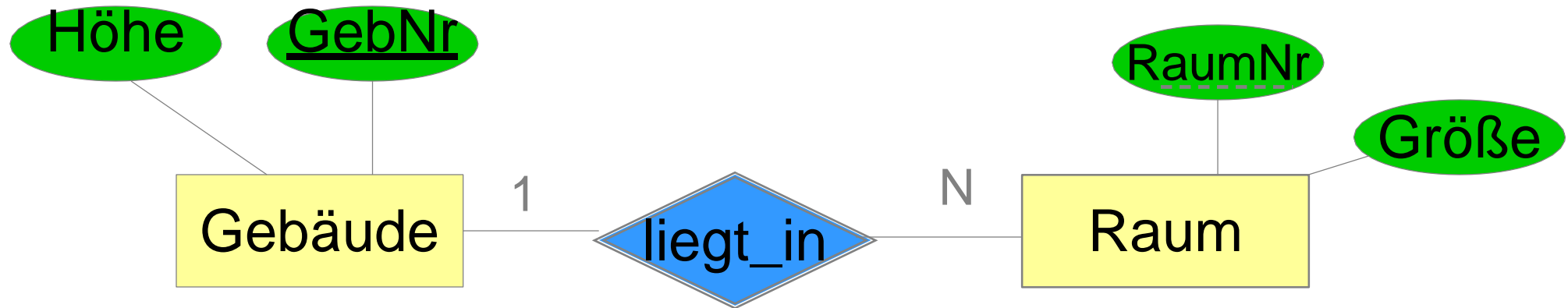


Jede Abteilung kann beliebig viele Mitarbeiter haben, jeder Mitarbeiter muss zu mind. einer Abteilung gehören.



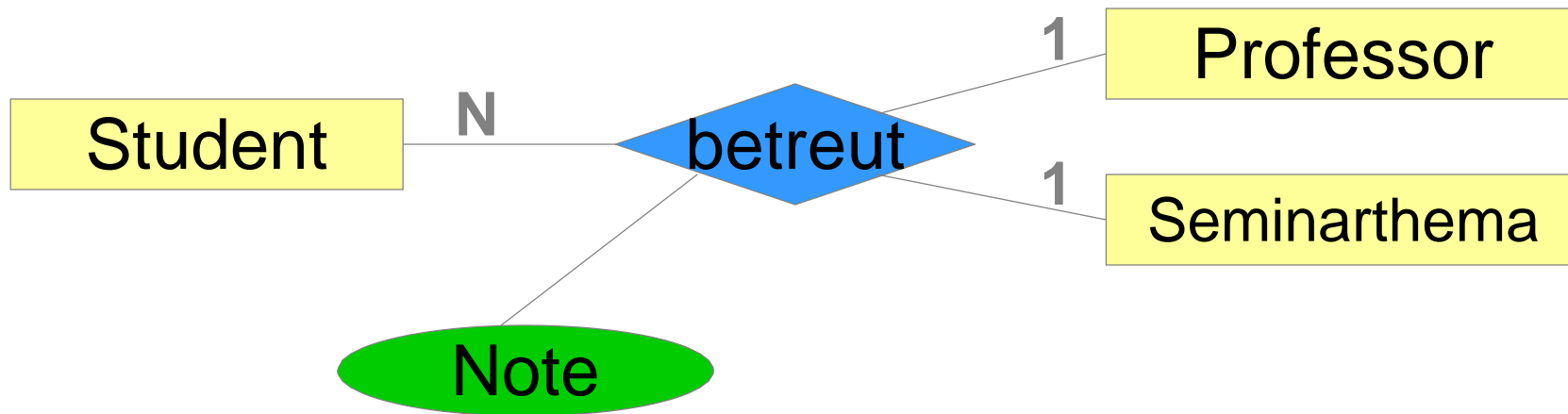
Jede Abteilung muss mind. einen Mitarbeiter haben, jeder Mitarbeiter muss zu mind. einer Abteilung gehören.

Beispiel: Schwacher Beziehungstyp



- 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

Beispiel: Mehrwertige Beziehung



betreut: Professoren x Studenten → Seminarthemen

betreut: Seminarthemen x Studenten → Professoren

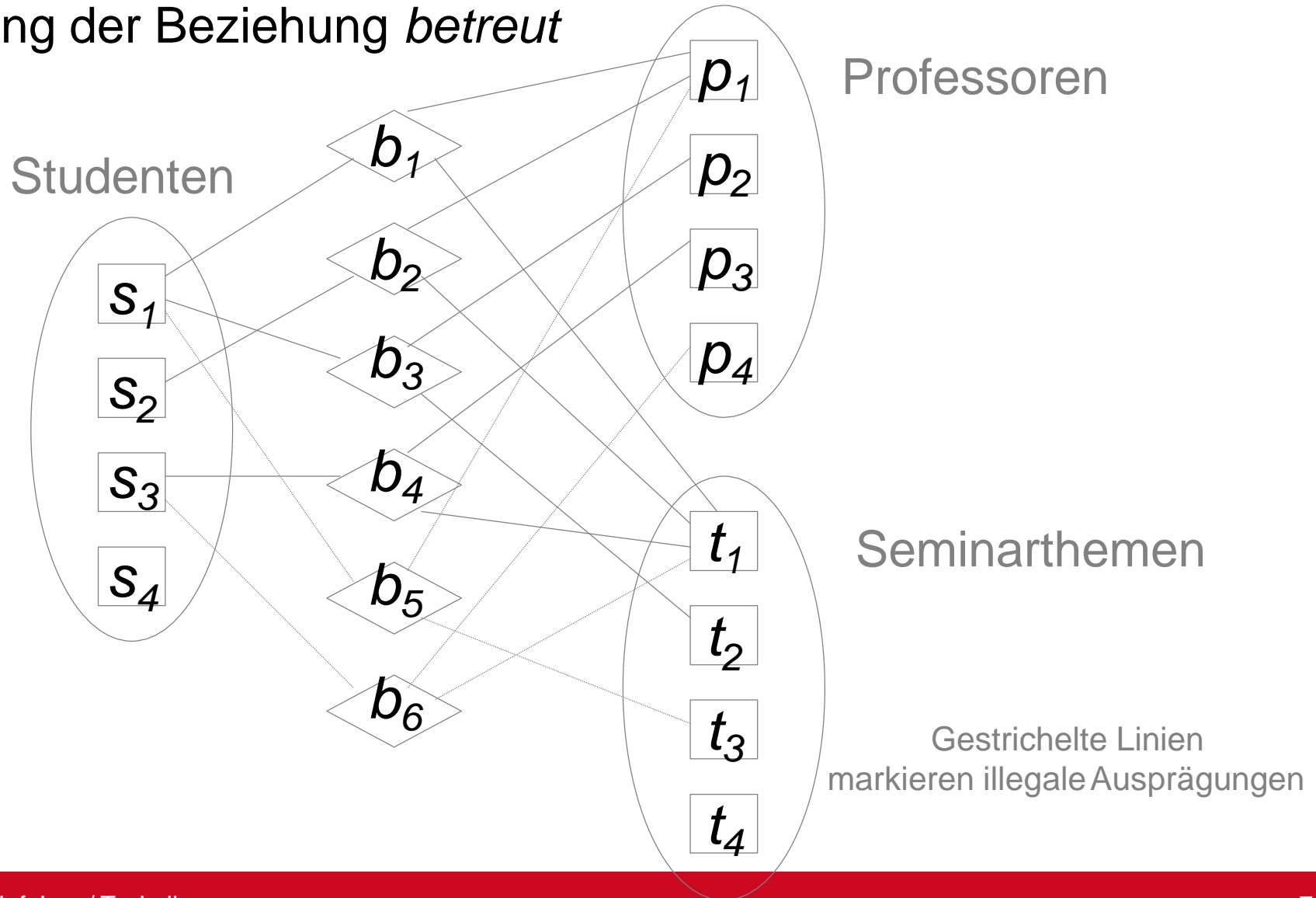
Dadurch erzwungene Konsistenzbedingungen

- Studenten dürfen bei demselben Professor bzw. derselben Professorin nur ein Seminarthema „ableisten“ (damit ein breites Spektrum abgedeckt wird).
- Studenten dürfen dasselbe Seminarthema nur einmal bearbeiten – sie dürfen also nicht bei anderen Professoren ein ihnen 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 an unterschiedliche Studenten vergeben werden

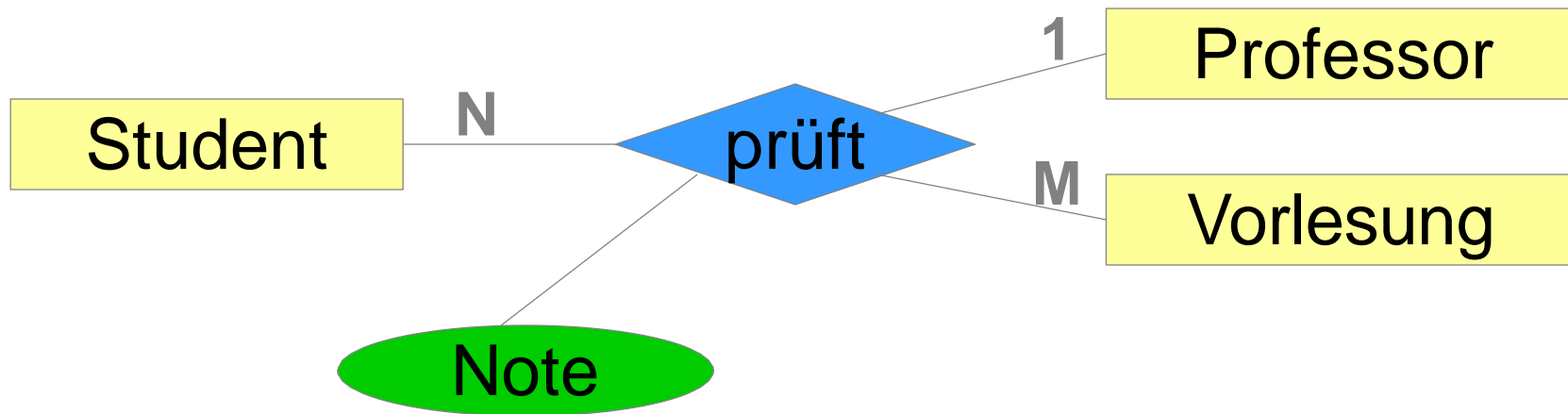
Ausprägung der Beziehung *betreut*



Abstraktion der DHBW-Organisation:

- Professoren lesen mehrere Vorlesungen, die wiederum von mehreren Studenten gehört werden.
- Die Studenten werden im Rahmen jeder Vorlesung von dem jeweiligen Dozenten geprüft unter der Vergabe einer Note.
- Weiterhin haben Professoren, einen oder mehrere Assistenten, wobei ein Assistent immer genau für einen Professor arbeitet.
- Es gibt Vorlesungen, die andere Vorlesungen voraussetzen.
- Erstellen Sie ein entsprechendes ERM und definieren Sie dabei für jede Entitätsklasse ein Schlüsselattribut und mindestens drei weitere Attribute.
- Setzen Sie die Entitätsklassen unter Verwendung der funktionalen Beziehungen „1:1“, „1:N“ bzw. „N:M“ miteinander in Verbindung.

Mehrwertige Beziehung aus Sicht des Studenten



Ein Student wird zu den verschiedenen Vorlesungen durch einen Professor geprüft.

Student 1 wird zu Vorlesung AB durch mehrere Profs geprüft wird → **geht nicht**

Student 1 wird zu Vorlesung AB und CD durch Prof XY geprüft → **geht**

Mehrwertige Beziehung

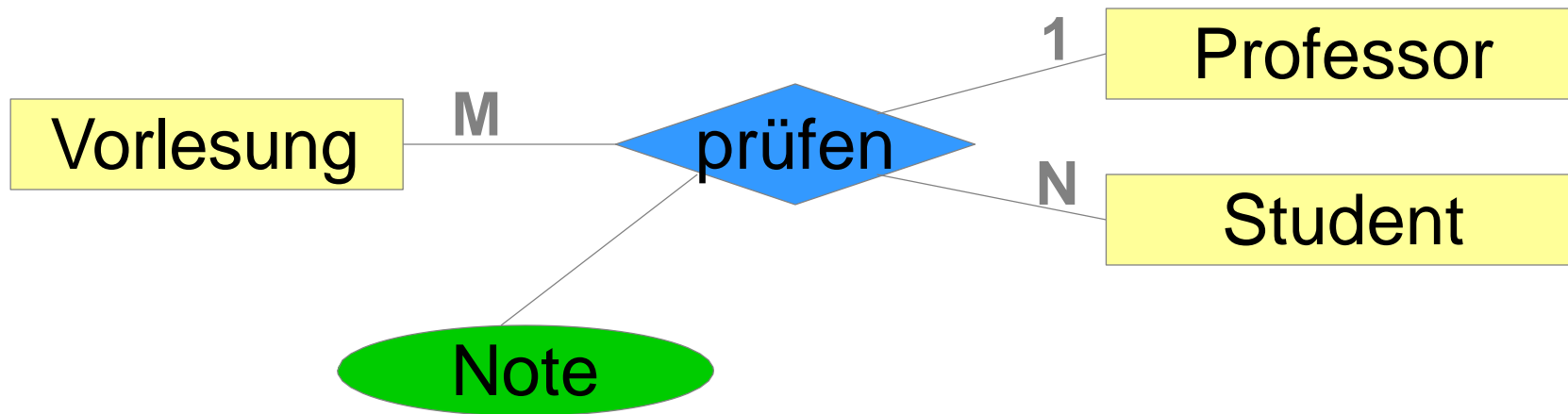
Student	Vorlesung	Professor	Note
S1	AB	XY	1
S1	AB	ZZ	2
S1	CD	XY	5

Ein Student wird zu den verschiedenen Vorlesungen durch einen Professor geprüft.

Student wird zu Vorlesung AB durch mehrere Profs geprüft wird → **geht nicht**

Student wird zu Vorlesung AB und CD durch Prof XY geprüft → **geht**

Mehrwertige Beziehung aus Sicht der Vorlesung



Eine Vorlesung wird von einem Prof an mehreren Studenten abgeprüft.

Vorlesung AB wird durch Prof XY und ZZ an Student 1 abgeprüft wird → **geht nicht**

Vorlesung AB wird durch Prof XY an Student 1 und Student 2 abgeprüft → **ok**

Vorlesung AB wird durch Prof XY an Student 1 und durch Prof ZZ an Student 2 abgeprüft → **ok**

Mehrwertige Beziehung

Student	Vorlesung	Professor	Note
S1	AB	XY	1
S1	AB	ZZ	2
S2	AB	XY	2
S3	AB	ZZ	1

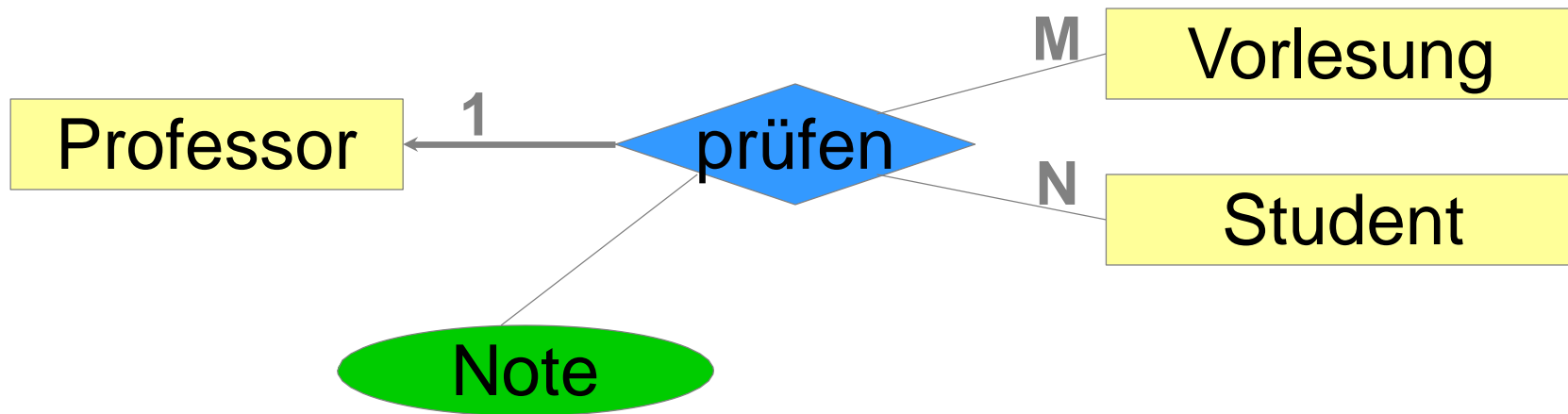
Eine Vorlesung wird von einem Prof an mehreren Studenten abgeprüft.

Vorlesung AB wird durch Prof XY und ZZ an Student 1 abgeprüft wird → **geht nicht**

Vorlesung AB wird durch Prof XY an Student 1 und Student 2 abgeprüft → **ok**

Vorlesung AB wird durch Prof XY an Student 1 und durch Prof ZZ an Student 2 abgeprüft → **ok**

Mehrwertige Beziehung aus Sicht des Professors



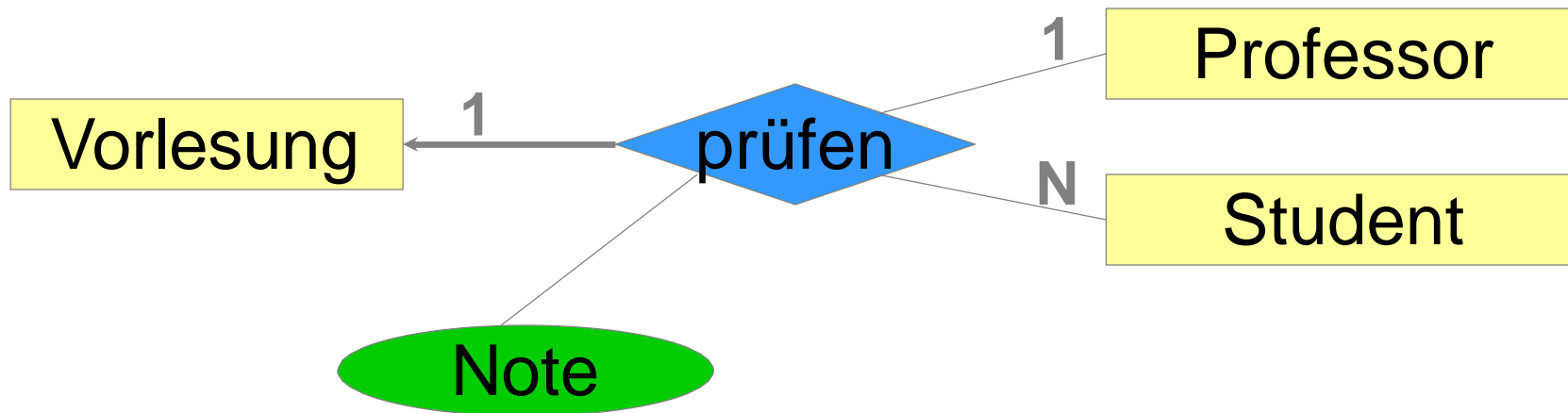
Ein Professor prüft seine Vorlesungen jeweils an mehreren Studenten ab.

Vorlesung x Student → Professor

Studenten 1, 2 und 3 in Vorlesung AB → werden durch Prof XY geprüft → ok

Student 1 in Vorlesung AB und Vorlesung CD → werden durch Prof XY geprüft → ok

Wie wäre es hier?



Ein Student wird durch den selben Prof nur einmal geprüft.

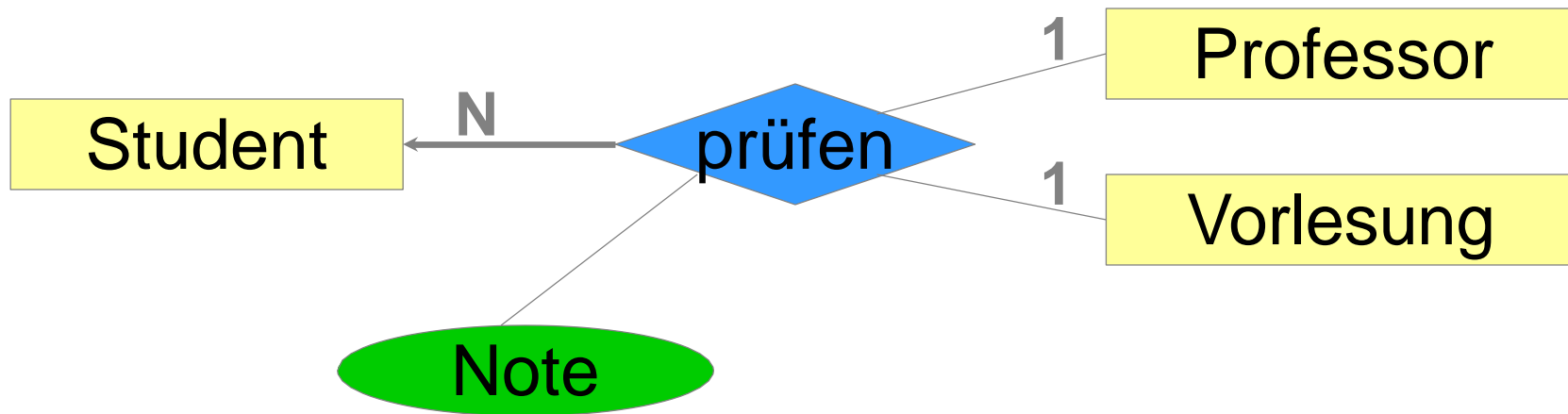
1 Professor x N Student → 1 Vorlesung

Prof XY prüft Student 1 und Student 2 → zu Vorlesung AB → ok

Prof XY prüft Student 1 → zu Vorlesung AB und CD → geht nicht

Prof XY und ZZ prüfen Student 1 → zu Vorlesung AB → geht nicht

Wie wäre es hier?



Ein Student wird durch den selben Prof nur einmal geprüft.

1 Professor x 1 Vorlesung → N Studenten

Prof XY in Vorlesung AB u. Prof ZZ in Vorlesung CD → prüfen Student 1 → **ok**

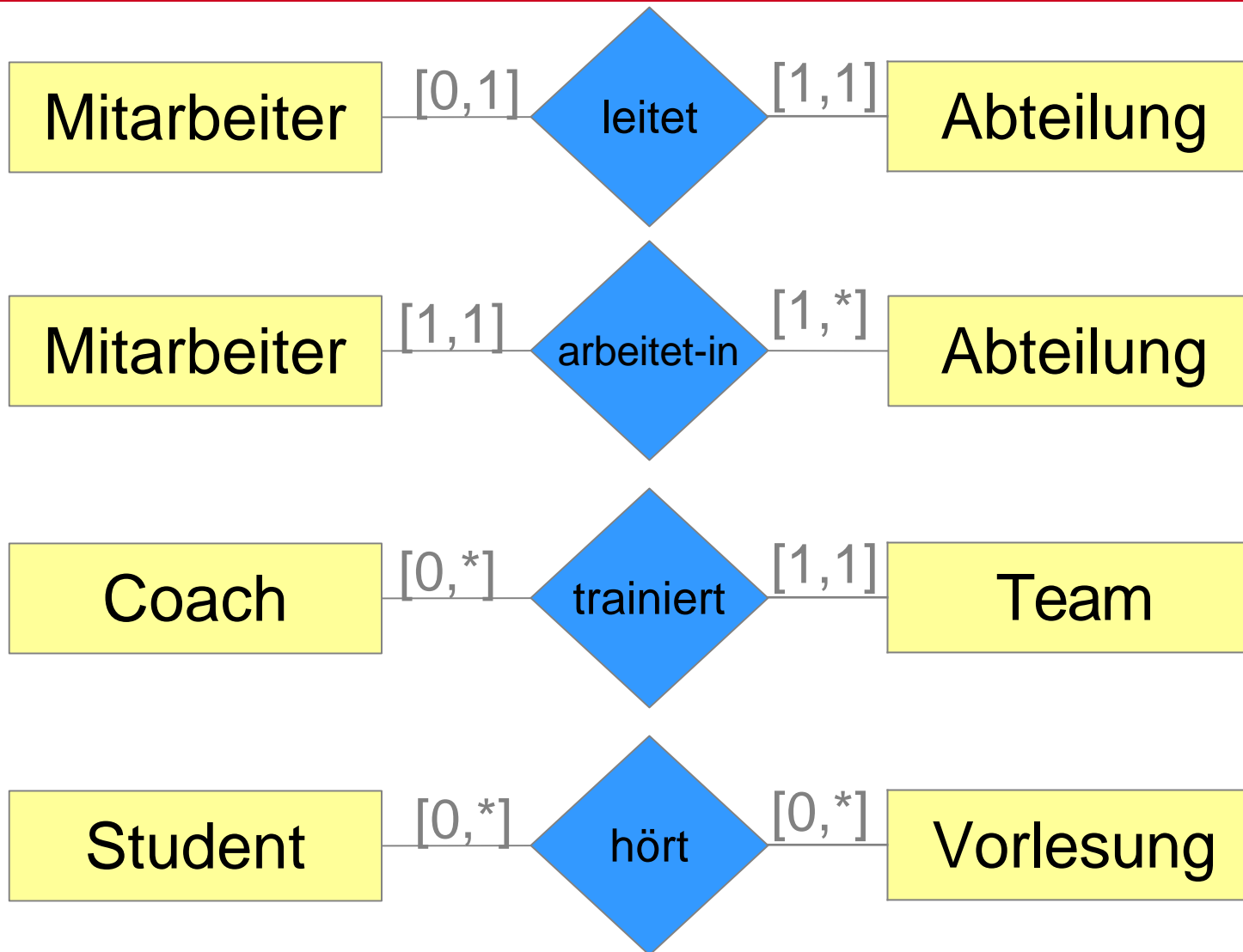
Prof XY in Vorlesung AB und CD → prüft Student 1 → **geht nicht**

Die (min, max)-Notation

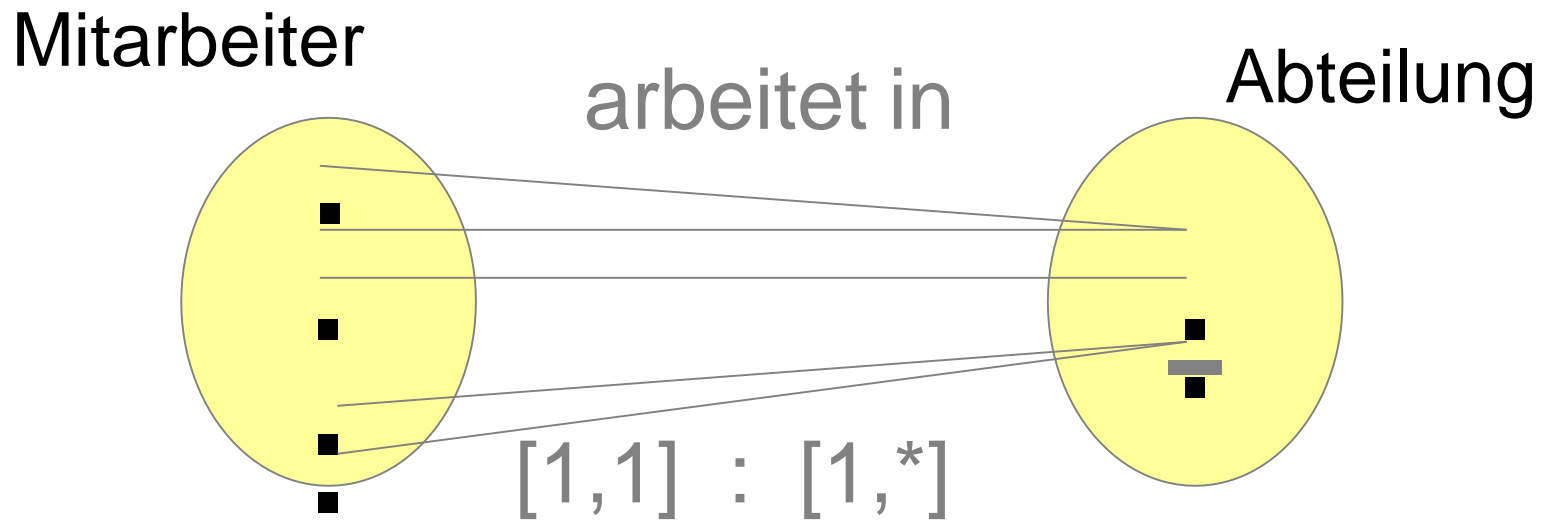
- Es gibt in der Datenbank-Literatur auch noch eine andere entgegengesetzte Definition / Interpretation der funktionalen Beziehung (1:N)
- Gemäß dieser Definition schränkt die Kardinalitätsangabe die möglichen Teilnahmen der Instanzen des zugehörigen Entity-Types an der Beziehung entsprechend dem angegebenen Wert ein

- Bei der *(min, max)*-Notation schränkt die Angabe $[min, max]$ die möglichen Teilnahmen von Entitäten der Entitäts-Klassen ein.
- Für jede an einem Beziehungstyp beteiligte Entitäts-Klasse wird ein Paar von Zahlen, nämlich *min* und *max* angegeben.
- Das Zahlenpaar sagt aus, dass jede Entität dieser Klasse mindestens *min*- und höchstens *max*-mal in der Beziehung steht.
- Wenn es Entitäten geben darf, die gar nicht an der Beziehung teilnehmen, so wird *min* mit 0 angegeben; wenn eine Entität beliebig oft an der Beziehung teilnehmen darf, so wird die *max*-Angabe durch * ersetzt. Somit ist $(0,*)$ die allgemeinste Aussage.

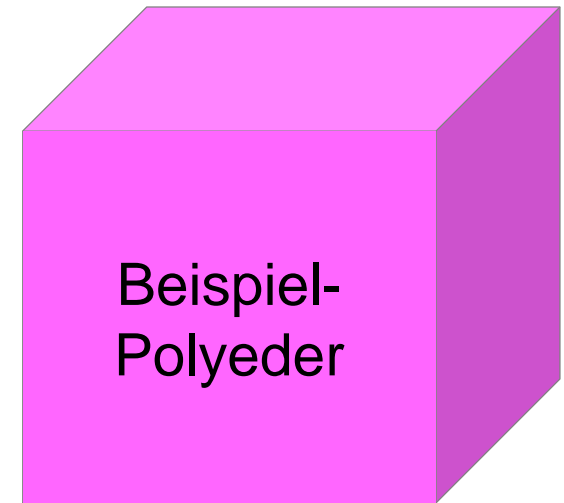
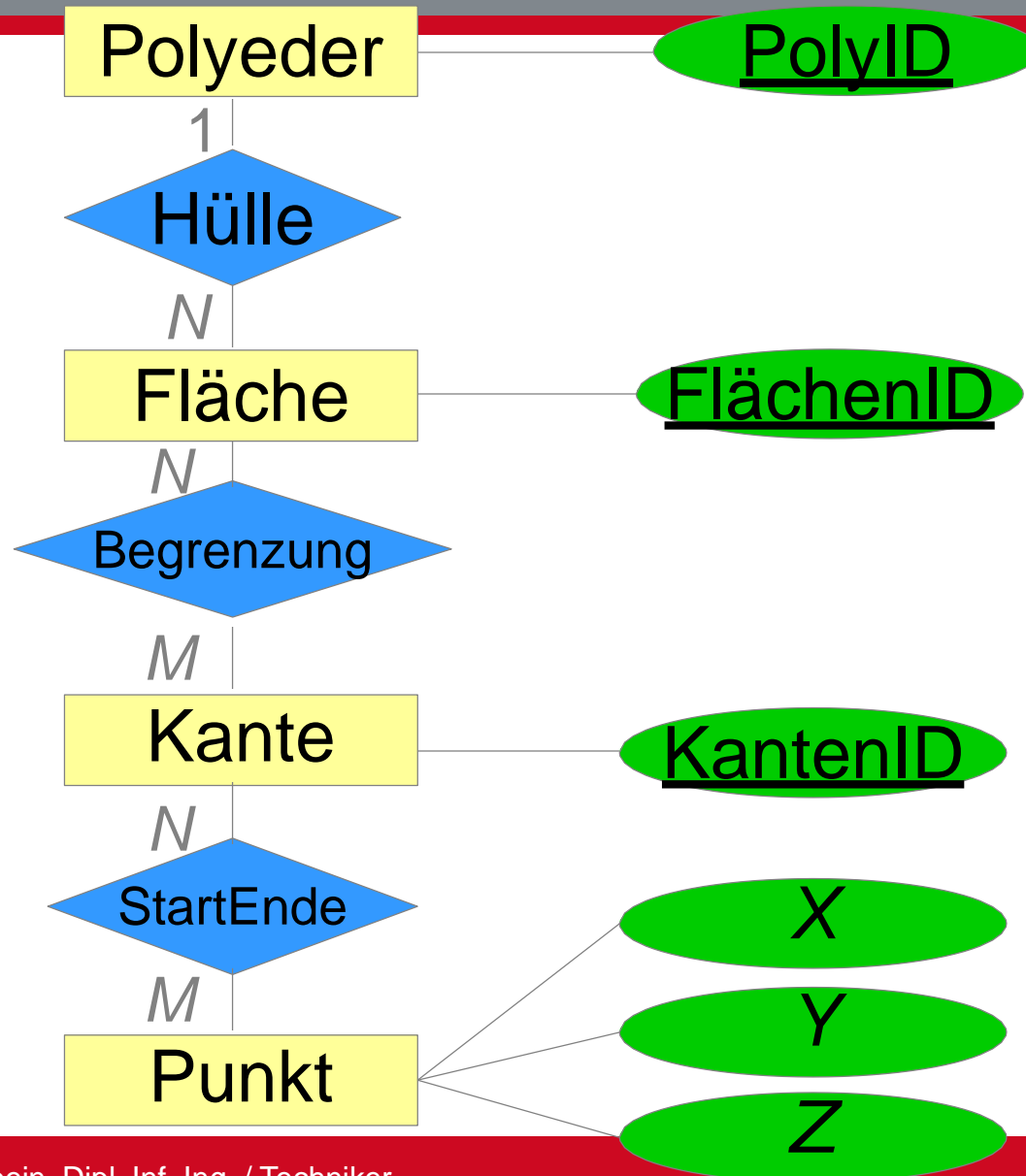
3.3.8 Die (min, max)-Notation



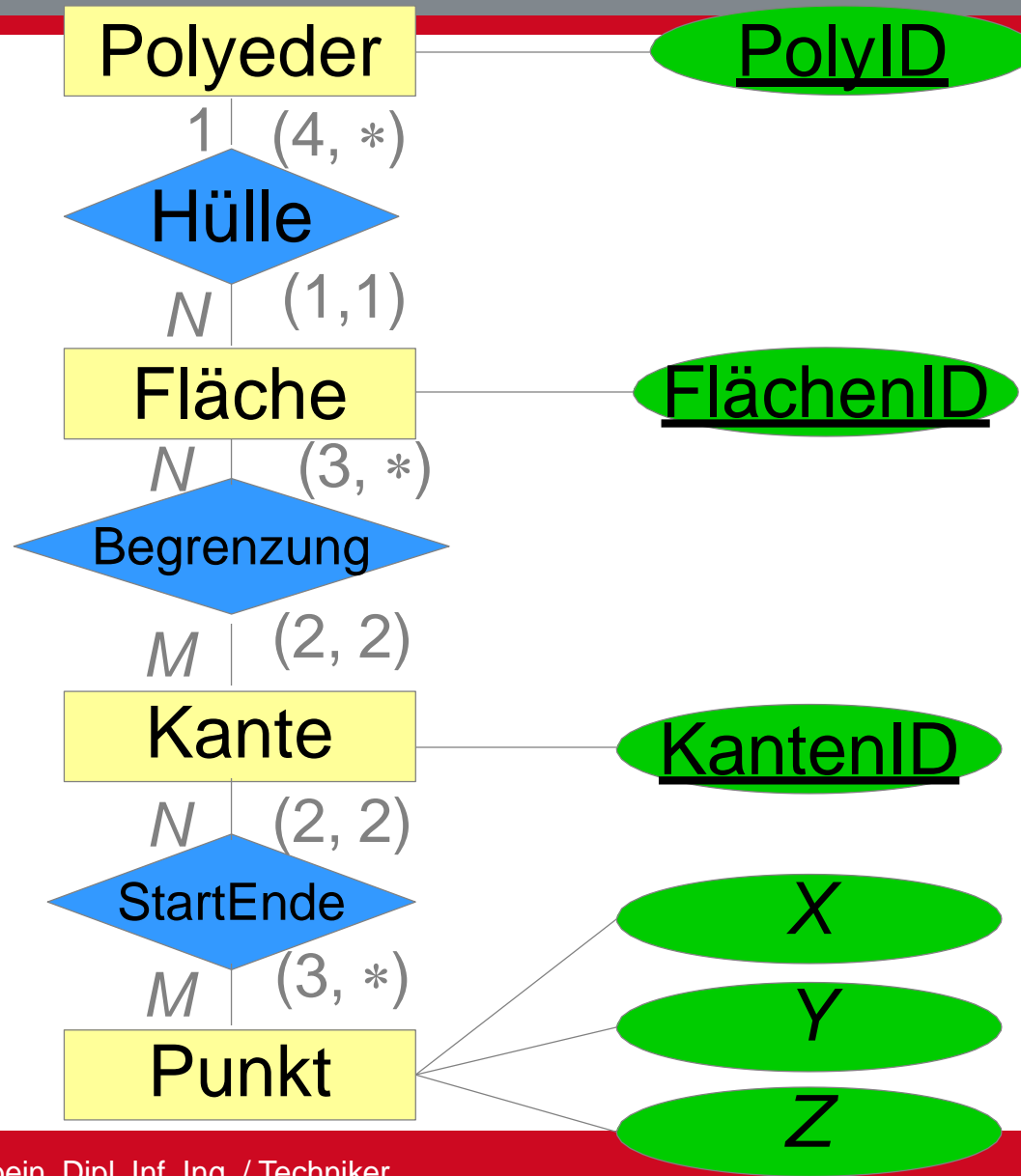
3.3.8 Die (min, max)-Notation



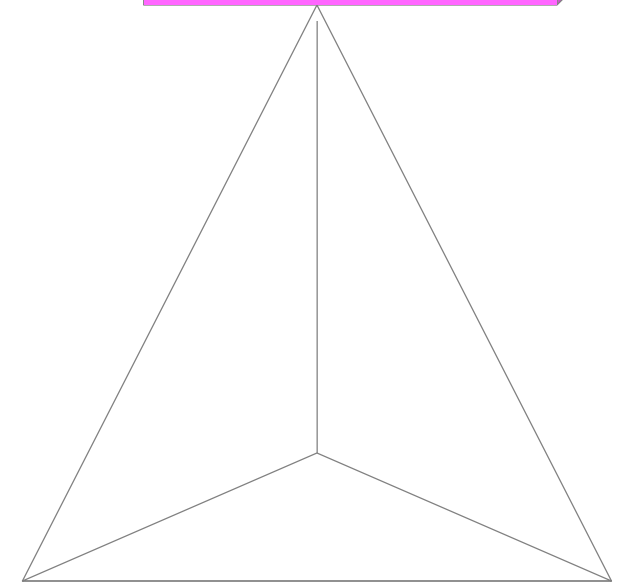
3.3.8 Die (min, max)-Notation

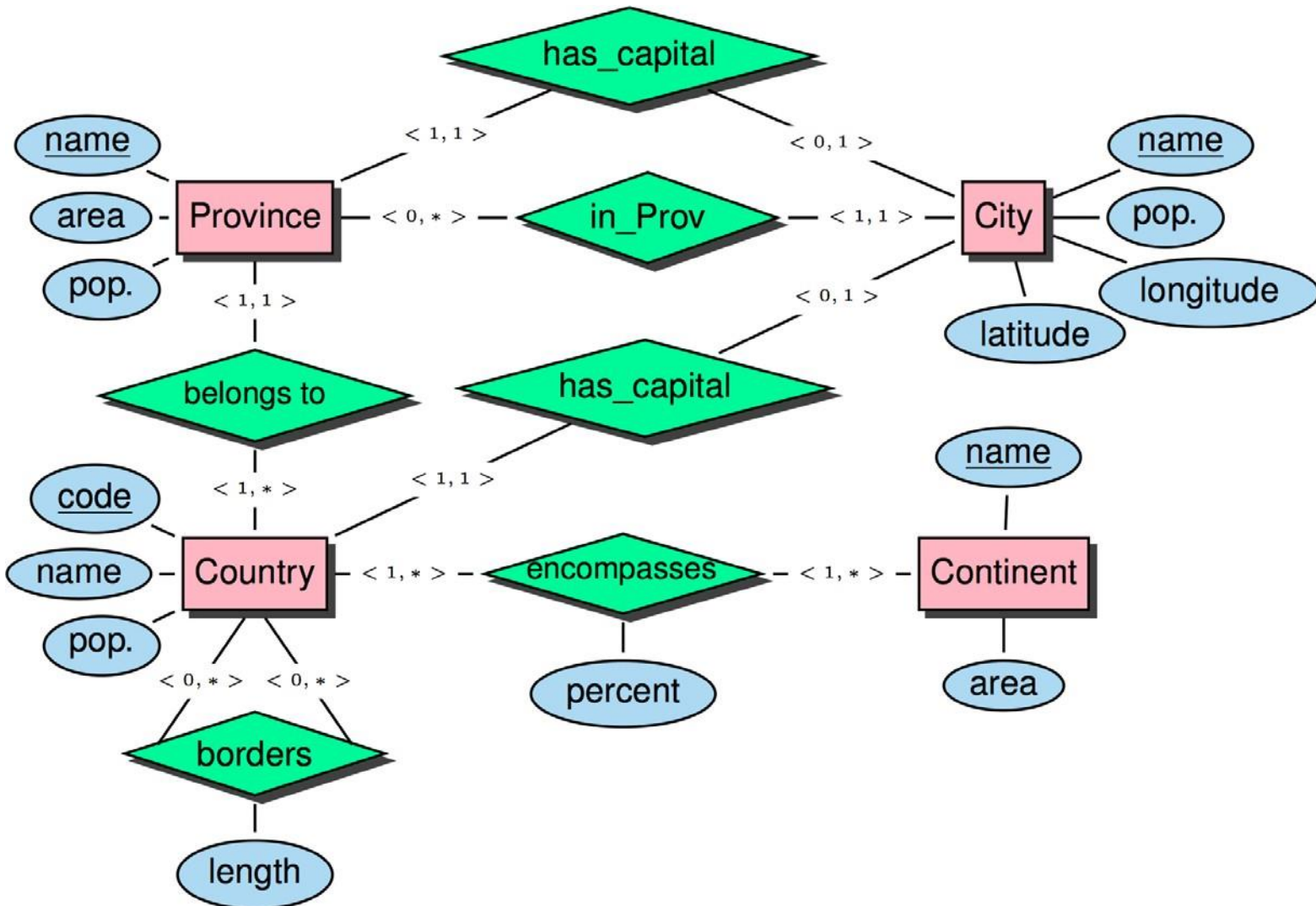


3.3.8 Die (min, max)-Notation



Beispiel-
Polyeder





„Das Erweiterungspaket“

- Zur weiteren Strukturierung der Entitätstypen wird die **Generalisierung** eingesetzt.
- Hierbei werden Eigenschaften von ähnlichen Entitätstypen einem gemeinsamen **Obertyp** zugeordnet.
- Bei dem jeweiligen **Untertyp** verbleiben nur die nicht faktorisierten Attribute. Somit stellt der Untertyp eine **Spezialisierung** des Obertyps dar.
- Spezialisierung wird durch eine Beziehung mit dem Namen **is-a** (ist ein) ausgedrückt, welche durch ein Sechseck mit gerichteten Pfeilen symbolisiert wird.

Definition

Seien E_1, E_2 zwei Entitätstypen.

Der Beziehungstyp

E_1 *isa* E_2

besteht genau dann, wenn E_1 eine Spezialisierung von E_2 ist.

E_1 heißt *Subtyp* des *Supertyps* E_2 .

Synonyme

- Spezialisierung / Generalisierung
- Unter- / Oberbegriff
- IST-Beziehung
- is-a-Relationship
- isa-Relationship

Besteht der Entitätstyp E_1 *isa* E_2 , so **erbt** E_1 die Attribute von E_2 . Dies hat folgende Konsequenzen:

- Für E_1 ist lediglich die Angabe zusätzlicher Attribute notwendig.
- Die Schlüsselattribute von E_2 sind auch die Schlüsselattribute von E_1 .
- Auf der Instanzenebene gilt: eine Entität von E_1 erbt die zugehörigen Werte aus E_2 .

Beispiel:

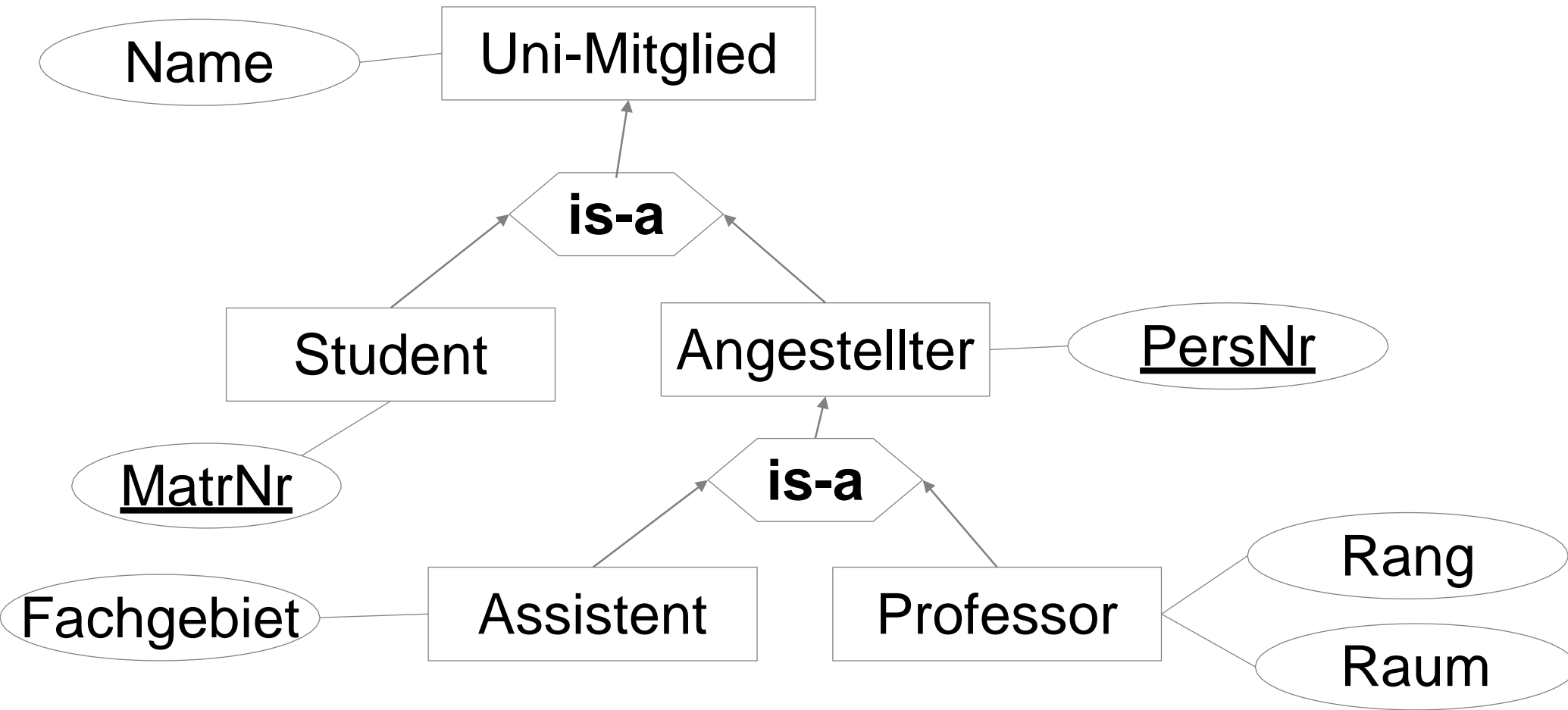
ABTEILUNGSLEITER *isa* ANGESTELLTER

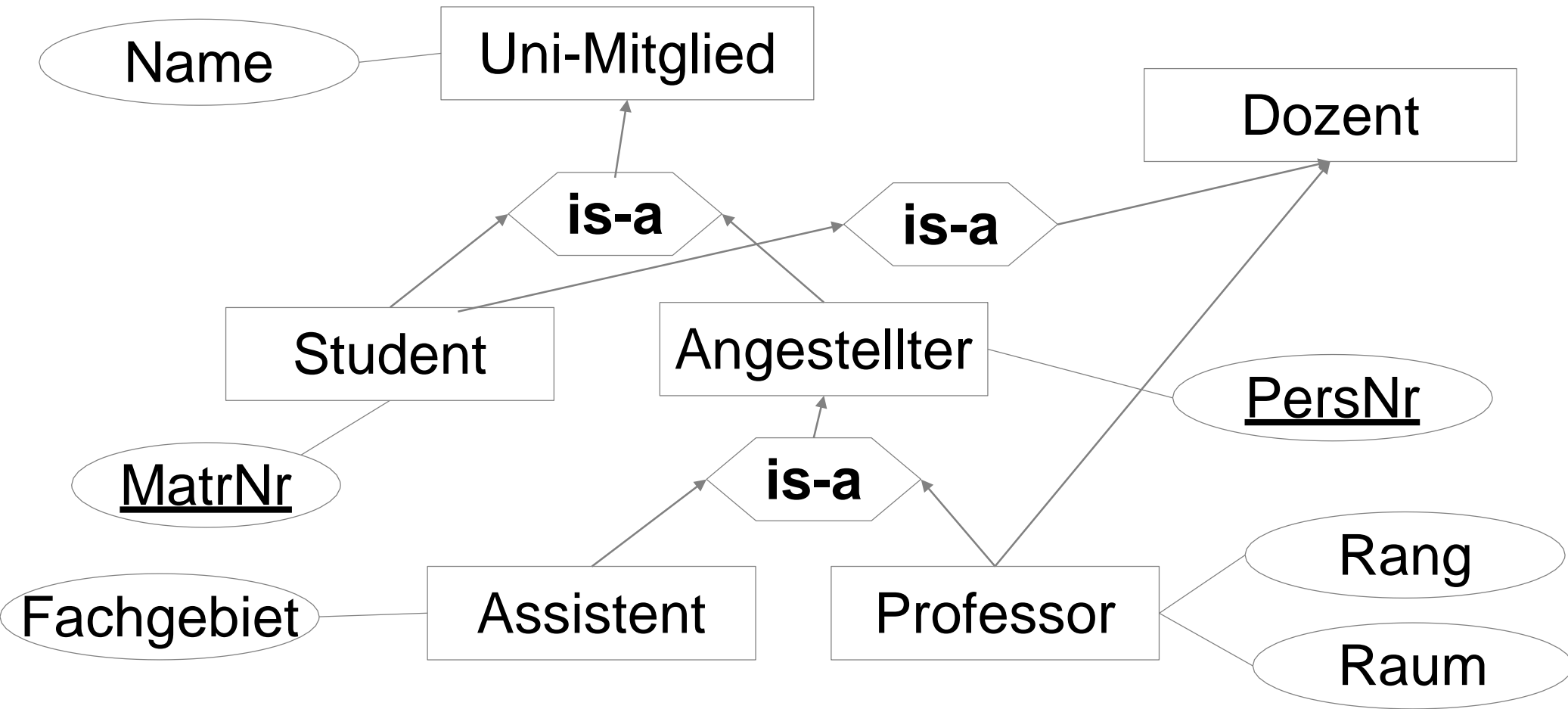
ABTEILUNGSLEITER (Dienstwagen: string)

ANGESTELLTER (Personalnr: int, Name: string)

In diesem Fall erbt ABTEILUNGSLEITER alle Attribute von ANGESTELLTER (also „Personalnr“ und „Name“) und hat das zusätzliche Attribut „Dienstwagen“.

Es besteht auch die Möglichkeit der *Mehrfachvererbung*.





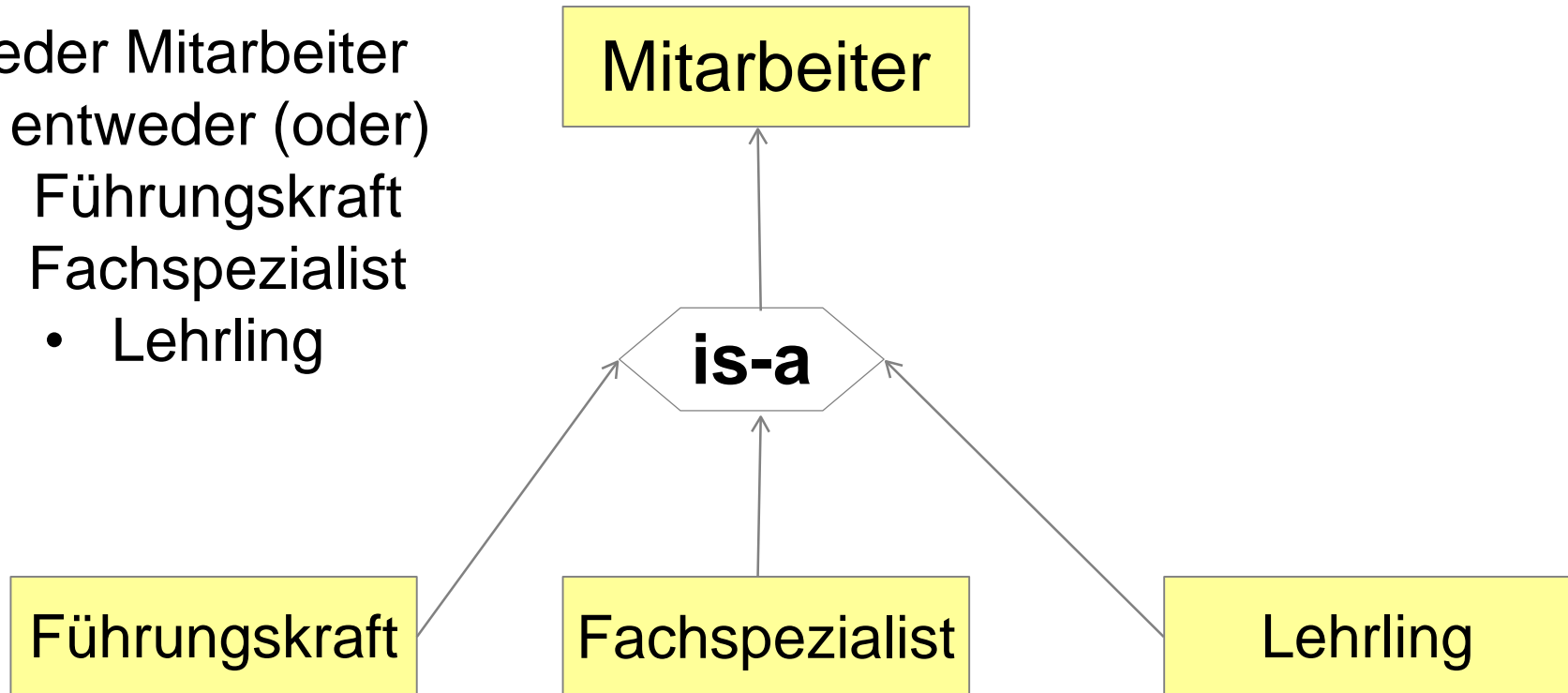
Bezüglich der Teilmengensicht ist von Interesse:

- die ***disjunkte*** Spezialisierung: die Schnittmenge der Entitätsmengen der Untertypen ergibt eine leere Menge
- die ***nicht disjunkte*** Spezialisierung: die Schnittmenge der Entitätsmengen der Untertypen enthält Elemente
- die ***vollständige (totale)*** Spezialisierung: die Obermenge enthält keine direkten Elemente, sondern setzt sich komplett aus der Vereinigung der Entitätsmengen der Untertypen zusammen (total vs. partiell)

Beispiel:

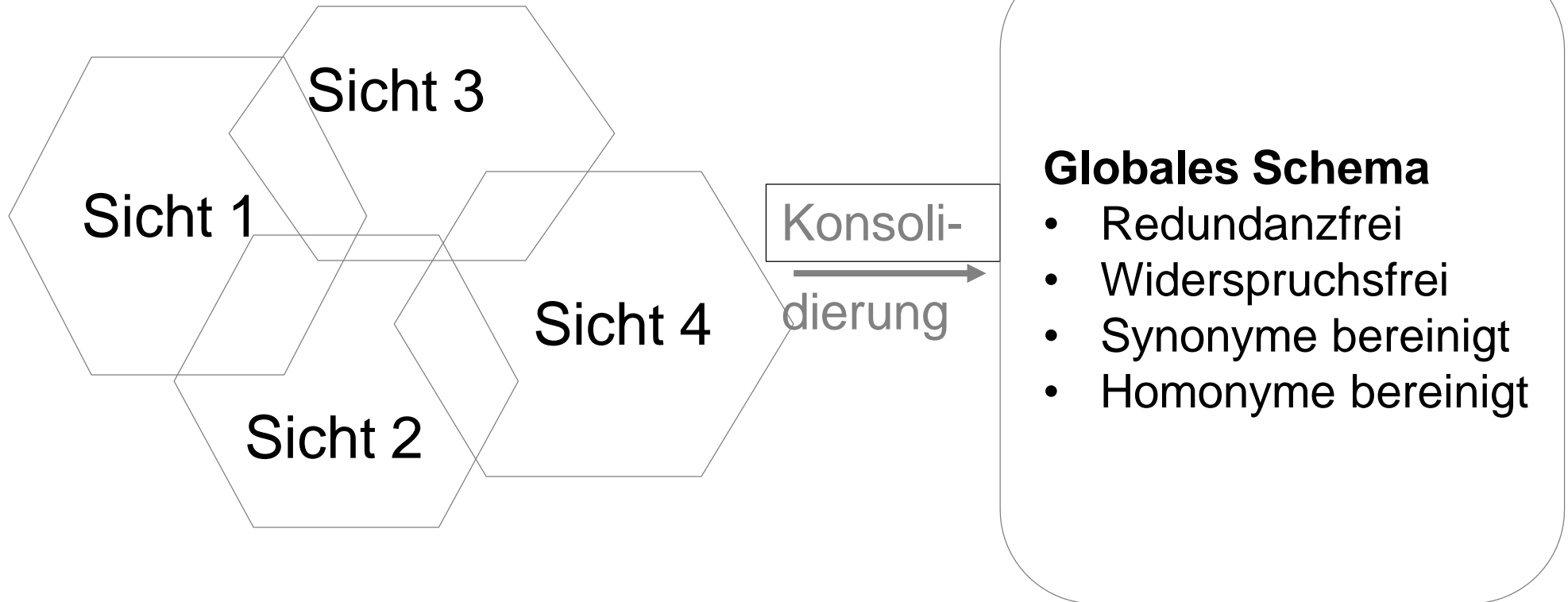
Jeder Mitarbeiter
ist entweder (oder)

- Führungskraft
- Fachspezialist
 - Lehrling



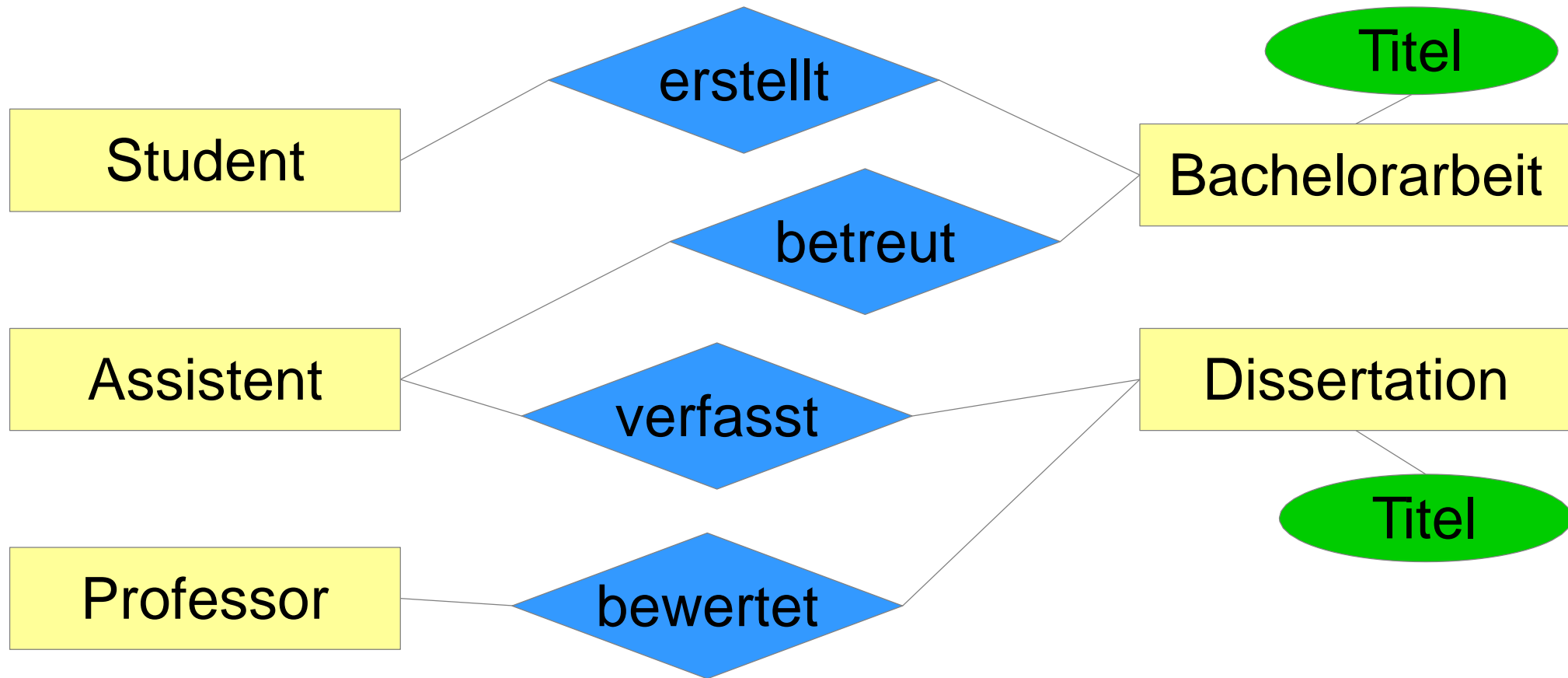
Schemakonsolidierung

- Bei der Modellierung eines komplexeren Sachverhaltes bietet es sich an, den konzeptuellen Entwurf zunächst in verschiedene Anwendersichten aufzuteilen.
- Nachdem die einzelnen Sichten modelliert sind, müssen sie zu einem globalen Schema zusammengefasst werden.

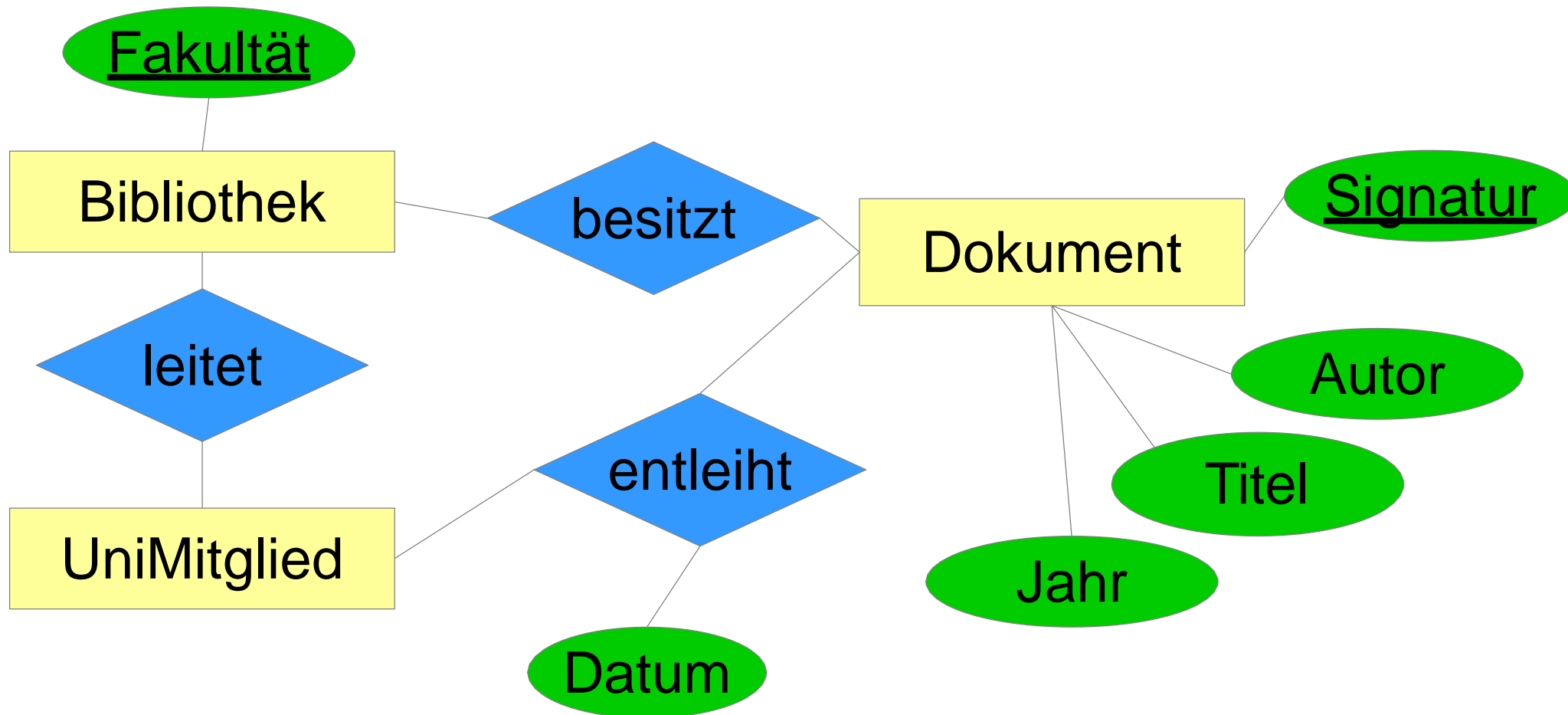


- Probleme entstehen dadurch, dass sich die Datenbestände der verschiedenen Anwender teilweise überlappen. Daher reicht es nicht, die einzelnen konzeptuellen Schemata zu vereinen, sondern sie müssen **konsolidiert** werden.
- Darunter versteht man das Entfernen von Redundanzen und Widersprüchen.
- Widersprüche entstehen durch **Synonyme** (gleiche Sachverhalte wurden unterschiedlich benannt) und durch **Homonyme** (unterschiedliche Sachverhalte wurden gleich benannt)
- Widersprüche entstehen auch durch unterschiedliches Modellieren desselben Sachverhalts zum einen über Beziehungen, zum anderen über Attribute.
- Bei der Zusammenfassung von ähnlichen Entity-Types zu einem Obertyp bietet sich die Generalisierung an.

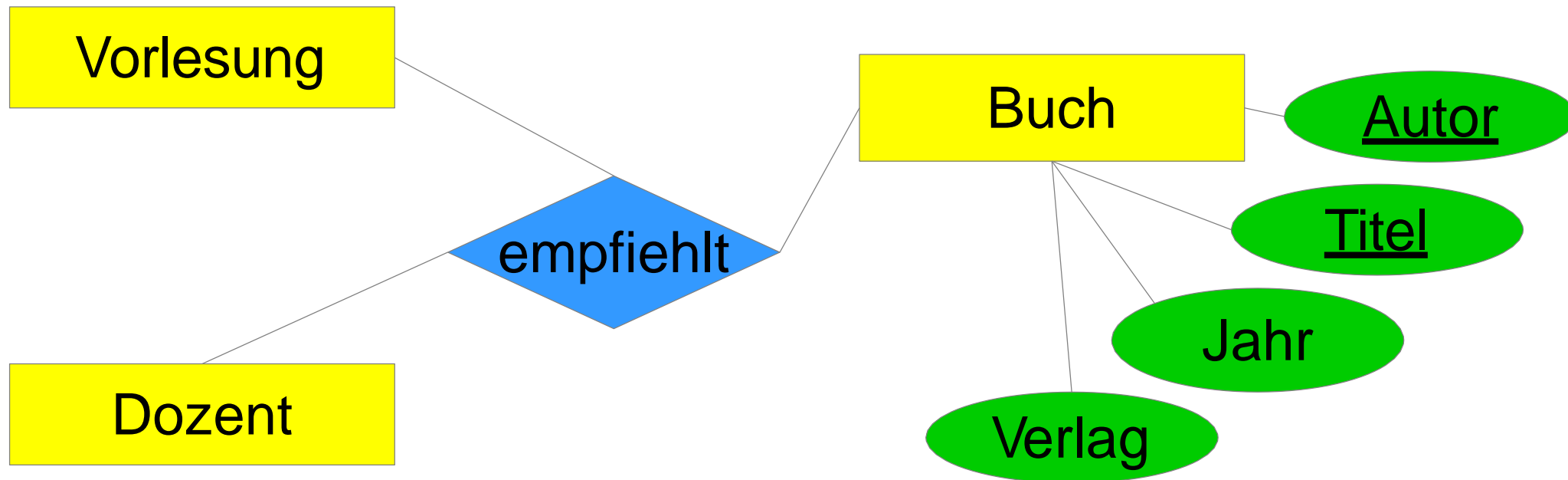
Sicht 1: Erstellung von Dokumenten als Prüfungsleistung



Sicht 2: Bibliotheksverwaltung



Sicht 3: Buchempfehlungen für Vorlesungen



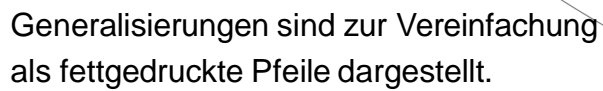
- Ausdrucken für Gruppenarbeit!
- → Ergebnis diskutieren

Beobachtungen:

- Die Begriffe *Dozenten* und *Professoren* werden synonym verwendet.
- Der Entitytyp *UniMitglied* ist eine Generalisierung von *Student*, *Professor* und *Assistent*.
- Fakultätsbibliotheken werden 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 *UniMitglied* in *Student* und *Angestellter* vornehmen.
- *Dissertationen*, *Bachelorarbeiten* und *Bücher* sind Spezialisierungen von *Dokumenten*, die in den *Bibliotheken* verwaltet werden.

Beobachtungen:

- Wir können davon ausgehen, dass alle an der Universität erstellten *Bachelorarbeiten* 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.



3.4 Das Relationale DB-Modell nach Codd

- Das Relationale Modell wurde 1970 von E.F. Codd entwickelt
- Die Version 2 wurde 1990 veröffentlicht. Ein DBMS muss danach theoretisch 333 Kriterien erfüllen, um relational zu sein.
- Das RM besteht aus der Definition von Objekten, Operatoren und Regeln. Die Operatoren definieren eine relationale Algebra mit der die Objekte bearbeitet werden können.
- Basiert auf der Mengenlehre (Relationale Algebra)
- Zugriff auf die Datensätze wird über die Feldinhalte ermöglicht.
- Dementsprechend arbeitet der Benutzer nur mit logischen, mengenorientierten Abfragen, wobei die physische Speicherung und der Datenzugriff für ihn im Hintergrund bleiben.

- Daten werden in Tabellen (Relationen) dargestellt mit:
 - Eindeutige Namen
 - Menge von Datensätzen (Tupel)
 - Menge von Attributen, deren Wertebereiche als Domänen bezeichnet werden
- Die Datensätze bilden die Zeilen und die Merkmale des Objektes bzw. die Datenfelder entsprechen den Spalten der Tabelle.
- Beim relationalen Datenmodell stehen als Strukturelemente ausschließlich Relationen, die sich durch Tabellen darstellen lassen, zur Verfügung.
- Alle Daten werden ausschließlich über inhaltliche Angaben aufeinander bezogen; die Beziehungen zwischen beliebigen Datensätzen werden über gleiche Feldinhalte hergestellt.

1. Informationsregel

Darstellung von Informationen ausschließlich auf logischer Ebene.
Beschreibung von DB und Relationen wiederum in Relationen.

2. Garantierter Zugriff

Zugriff auf jedes Element einer relationalen DB durch Tabellennamen,
Primärschlüssel und Spaltennamen.

3. Systematische Behandlung von NULL-Werten

Null-Werte müssen unterstützt werden.

4. Integrierter Datenkatalog

Zugriff auf Datenkatalog mit derselben Sprache wie auf reguläre Daten

5. Umfassende Datenbanksprache

6. Datenmanipulation über logische Sichten

Datenmanipulationen müssen auch über Views möglich sein.

7. Mengenorientiertes Einfügen, Ändern und Löschen

8. Physische Datenunabhängigkeit

9. Logische Datenunabhängigkeit

10. Integritätsunabhängigkeit

Integritätsbedingungen müssen Bestandteil der relationalen Datenbanksprache sein. Sie müssen im Systemkatalog abgelegt werden.

11. Verteilungsunabhängigkeit

Unabhängigkeit gegenüber verteilten Datenbeständen.

12. Unterwanderungsverbot

Regeln 1-11 dürfen nicht durch andere Mechanismen sichergestellt werden.

Datenbankschema:

- Unter einem **Datenbankschema** versteht man eine Spezifikation der Datenstrukturen einer Datenbank mit den zugehörigen Integritätsbedingungen.
- Ein Datenbankschema enthält die Definition der
 - Tabellen
 - Attribute
 - Primärschlüssel
 - Integritätsbedingungen (Einschränkungen der Wertebereiche der Attribute)

Relationales DB-Schema:

- Menge aller Relationenschemata in der Datenbank

Relationale DB:

- Das relationale DB-Schema zusammen mit den momentanen Werten der Relationen

Tabellen:

- Tabellename
- Zeilen → Datensätze
- Spalten → Merkmale, Attribute, (Felder)
- Zellen → einzelne Datenelemente

Tabellendefinition:

- Eindeutiger Tabellename
- Eindeutige Merkmalsnamen (Attribute) pro Tabelle
- Reihenfolge der Merkmale ist egal
- Die Reihenfolge der Datensätze ist beliebig
- Anzahl der Merkmale ist beliebig (endlich)
- Anzahl der Datensätze ist beliebig (endlich)
- Mit jedem Merkmal wird ein Datentyp verknüpft
- Schlüsselfeld dient der eindeutigen Identifikation eines Datensatzes
- Es gibt keine 2 Datensätze mit identischem Schlüsselwert

3.4.3 Begriffsdefinitionen

Relation / Entitätstyp / Tabelle
Relationenname

Attribute Spalte

Kunde

Kundennummer	Name	Str.	PLZ	Ort	Kontonr.	BLZ	Rabatt in %
25	Honeywell	Lindenstr. 1	08150	Adorf	0312951252	45090000	20
64	IBM	Buchenstr. 2	12345	Berlin	3251880820	35070000	10
34	Sony	Eichenweg 3	67895	Frankfurt	2365800150	51090000	23
54	Schenker	Eschenplatz 4	15007	Ratingen	0254650500	61011000	0
23	Epson	Kaktusallee 5	32489	München	1259025000	51090024	NULL
53	Microsoft	Tulpengasse 7	15894	Hamburg	1584034500	75030000	10

Relationschema

Entität /
Tupel
Datensatz

Relation

Attributwert / Datenelement / Datenwert

• Relation	Tabelle
• Attribut	Spalte
• Tupel	Datensatz
• Domain	Wertebereich
• Candidate Key	eindeutiger Schlüssel, möglicher Hauptschlüssel
• Primary Key	Hauptschlüssel
• Alternate Key	Alternativschlüssel
• Foreign Key	Fremdschlüssel
• Degree	Ausdehnungsgrad (Anzahl der Attribute)
• Kardinalität	Mächtigkeit der Tabelle (Anzahl der Tupel)

Schlüssel (Key):

- Zur Identifizierung von Datensätzen innerhalb einer Tabelle:
 - Merkmalschlüssel (z.B. Name)
 - Künstlicher Schlüssel (z.B. Liefernummer)
 - Kombination aus Schlüsseln
- Jede Teilmenge der Attributmenge eines Entitätstypen, mit der die Entitäten dieses Typs eindeutig identifizierbar sind, heißt **Schlüssel** dieses Entitätstypen.
- Jeder Schlüssel stellt einen speziellen Attributtypen dar
- Jede minimale Menge von Schlüsselattributen heißt **Schlüsselkandidat**.

- **Schlüsselkandidat:** Attribute oder Attributkombinationen, die geeignet sind, den Datensatz zu identifizieren.
- **Primärschlüssel:** Der beim Entwurf der DB zur eindeutigen Identifikation und logischen Verknüpfung der Datensätze ausgewählte Schlüsselkandidat.
- **Zusammengesetzter Schlüssel:** Schlüssel, der aus mehreren Attributen besteht.
- **Fremdschlüssel:** Attribut (oder Attributskombination), das (bzw. die) in einer anderen Tabelle Primärschlüssel ist.
- **Sekundärschlüssel:** Ein oder mehrere Attribute eines Entitätstypen, die als zusätzliches Suchkriterium zum Auffinden von einem oder mehreren Datensätzen verwendet werden kann. Er ist nicht notwendigerweise eindeutig und daher kein Schlüssel im engeren Sinn.

3.4.4 Schlüssel

Kunde

<u>Kunden-nummer</u>	Name	Str.	PLZ	Ort	Kontonr.	BLZ	Rabatt in %
25	Honeywell	Lindenstr. 1	08150	Adorf	0312951252	45090000	20
64	IBM	Buchenstr. 2	12345	Berlin	3251880820	35070000	10
34	Sony	Eichenweg 3	67895	Frankfurt	2365800150	51090000	23

Bank



<u>BLZ</u>	Name der Bank
45090000	Deutsche Bank
35070000	Dresdener Bank
51090000	Commerzbank



Vom ER-Modell zum relationalen Modell

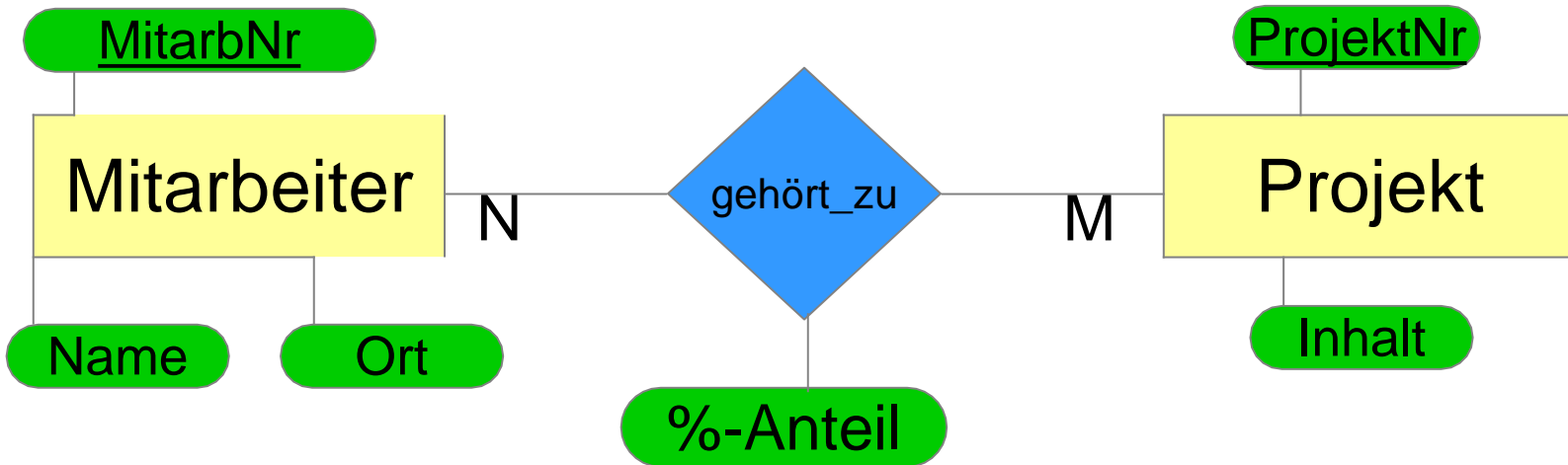
Regel 1:

- Jeder **Entitätstyp** muss als **eigenständige Tabelle** mit **eindeutigem Primärschlüssel** definiert werden.

Regel 2 (N:M-Beziehung):

- Jede **N:M** Beziehung muss als **eigenständige Tabelle** definiert werden.
- Die **Primärschlüssel** der zugehörigen Entitätstypen treten als **Fremdschlüssel** in dieser Tabelle auf.
- Der **Primärschlüssel der Beziehung** setzt sich aus den enthaltenen Fremdschlüsseln zusammen oder ist ein anderer Schlüsselkandidat (z.B. ein neuer künstlich eingeführter Schlüssel).

Regel 2 (N:M-Beziehung):



Mitarbeiter

<u>MitarbNr</u>	Name	Ort
r		

Zugehörigkeit

MitarbNr	ProjektNr	%-Anteil

Projekt

<u>ProjektNr</u>	Inhalt

Regel 2 (N:M-Beziehung):

Die Beziehung Mitarbeiter zu Projekt muß über die Relation Zugehörigkeit hergestellt werden.

BSP: Ein Mitarbeiter kann an mehreren Projekten beteiligt sein und ein Projekt kann von mehreren Mitarbeitern durchgeführt werden.

Tab. Mitarbeiter

<u>MNr.</u>	Name
122	Meier
124	Müller
169	Huber

Tab. Projekt

<u>PNr.</u>	Name
1	Planung
2	Kurs
3	Fete

Regel 2 (N:M-Beziehung):

Die Beziehung Mitarbeiter zu Projekt muß über die Relation Zugehörigkeit hergestellt werden.

BSP: Ein Mitarbeiter kann an mehreren Projekten beteiligt sein und ein Projekt kann von mehreren Mitarbeitern durchgeführt werden.

Tab. Mitarbeiter

<u>MNr.</u>	Name
122	Meier
124	Müller
169	Huber

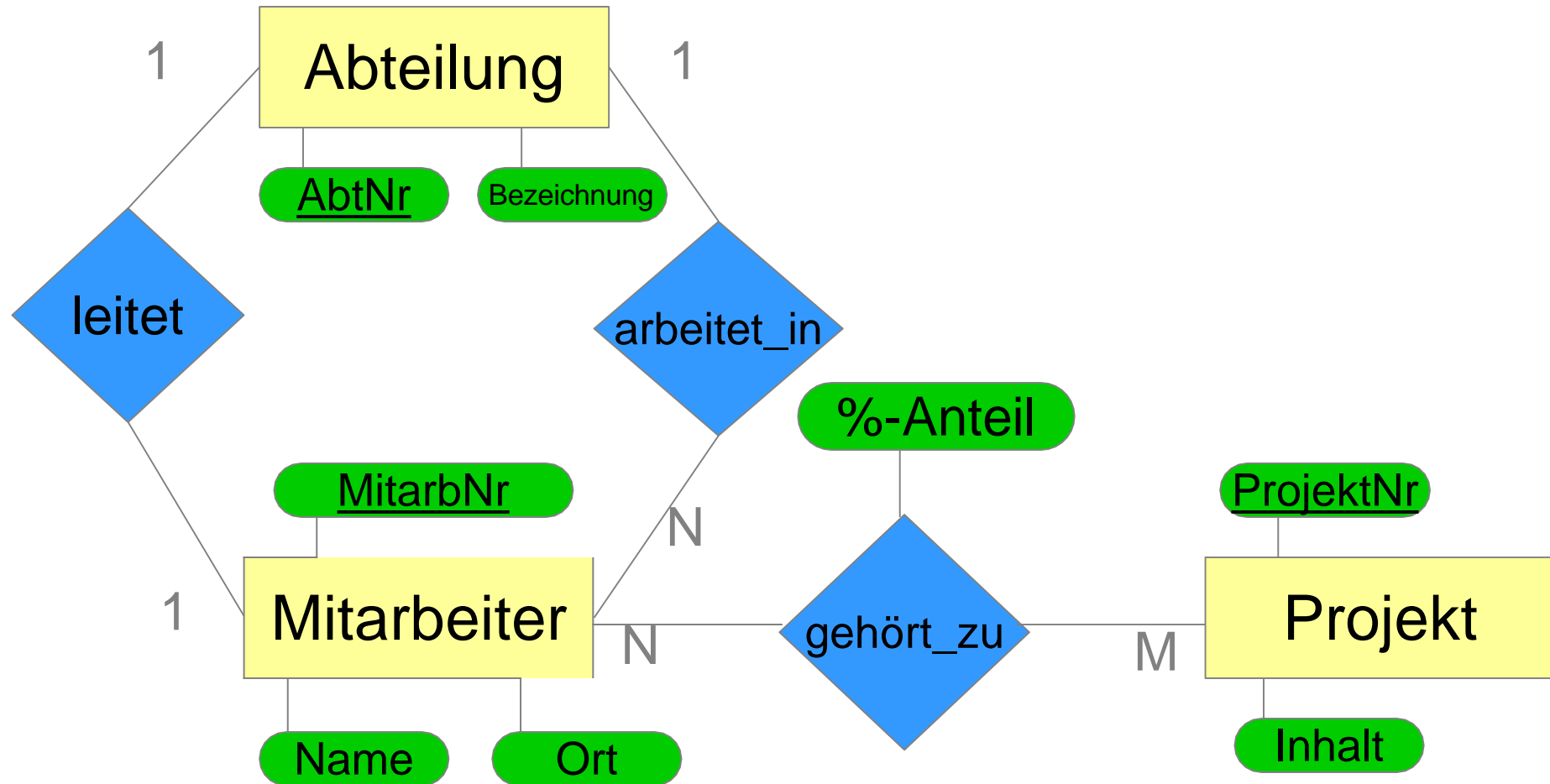
Tab. Zugehörigkeit

<u>MNr.</u>	<u>PNr.</u>
122	1
122	2
124	3
169	3

Tab. Projekt

<u>PNr.</u>	Name
1	Planung
2	Kurs
3	Fete





3.4.5 Abbildungsregeln

Abteilung

<u>AbtNr</u>	Bezeichnung

Projekt

<u>ProjektNr</u>	Inhalt

Mitarbeiter

<u>MitarbNr</u>	Name	Ort

Abteilungsleiter

<u>AbtNr</u>	MitarbNr

Unterstellung

<u>MitarbNr</u>	AbtNr

Zugehörigkeit

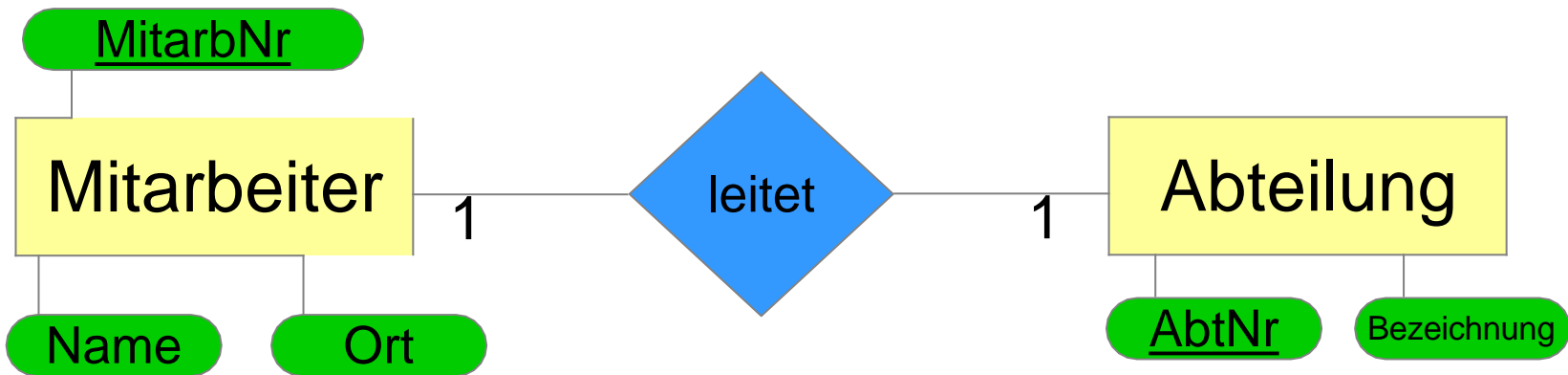
<u>MitarbNr</u>	<u>ProjektNr</u>	%-Anteil

- Im ER-Modell wurden Verben für Beziehungstypen verwendet. Sie wurden mit passenden Substantiven für Tabellennamen ersetzt.
- Da zu jeder Abteilung genau ein Abteilungsleiter gehört, genügt die Abteilungsnummer in der Tabelle „Abteilungsleiter“ als Primärschlüssel.
- Gleiches gilt für die Mitarbeiternummer in der Tabelle „Unterstellung“.
- In der Tabelle „Zugehörigkeit“ müssen die Fremdschlüssel „MitarbNr“ und „ProjektNr“ zusammen als Primärschlüssel definiert werden.

Regel 3 (1:1- Beziehung):

- Jede **1:1** Beziehung kann ohne zusätzliche eigenständige Tabelle definiert werden.
- Dazu wird **in einer** der beiden Tabellen mit Beziehungstyp **1** ein Fremdschlüssel auf die damit verknüpfte Tabelle geführt.
- Die Fremdschlüsselbeziehung wird durch ein Attribut gegeben, dessen Name sich z.B. aus dem Namen des entliehenen Primärschlüssels und einer Erläuterung der Beziehung zusammensetzen kann (ein Beispiel folgt).

Regel 3 (1:1- Beziehung):



Mitarbeiter

<u>MitarbNr</u>	Name	Ort

Abteilung

<u>AbtNr</u>	Bezeichnung	MitarbNr_Abteilungsleiter

Fremdschlüssel-Beziehung

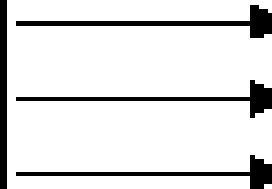
Regel 3 (1:1- Beziehung):

Mitarbeiter

<u>MNr.</u>	Name	PLZ	Strasse	Abt.
1	Meier	97070	aaa	A1
2	Müller	97082	bbb	A2
3	Huber	97090	ccc	A3

Abteilung

<u>Abteilung</u>	Bezeichnung
A1	Produktion
A2	Marketing
A3	Finanzen

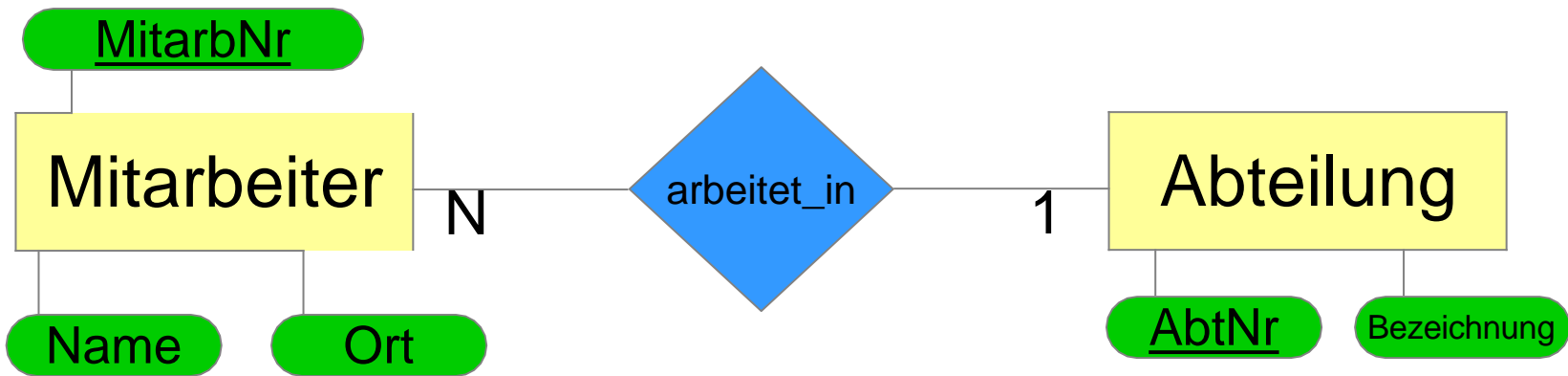


Dies zeigt eine mögliche Umsetzung in die „andere Richtung“.

Regel 4 (1:N - Beziehung):

- Jede **1:N** Beziehung **kann** ohne zusätzliche eigenständige Tabelle definiert werden.
- Dazu wird **in** der Tabelle mit Beziehungstyp **N** ein Fremdschlüssel auf die damit verknüpfte Tabelle geführt.
- Die Fremdschlüsselbeziehung wird durch ein Attribut gegeben, dessen Name sich z.B. aus dem Namen des entliehenen Primärschlüssels und einer Erläuterung der Beziehung zusammensetzen kann (ein Beispiel folgt)

Regel 4 (1:N - Beziehung):



Mitarbeiter

<u>MitarbNr</u>	Name	Ort	AbtNr_Unterstellung

Abteilung

<u>AbtNr</u>	Bezeichnung

Fremdschlüssel-Beziehung

Regel 4 (1:N - Beziehung):

Bsp.: Eine Abteilung hat mehr als einen Mitarbeiter.

Tab. Mitarbeiter

<u>MNr.</u>	Name	PLZ	Strasse	Abt.
1	Meier	97070	aaa	A1
2	Müller	97082	bbb	A3
3	Huber	97090	ccc	A1
4	Michler	97082	ddd	A2

Tab. Abteilung

<u>Abteilung</u>	Bezeichnung
A1	Produktion
A2	Marketing
A3	Finanzen

n:1

Ein auf dem ER-Modell für Mitarbeiter basierendes relationales Modell wird am Whiteboard zusammengefasst.

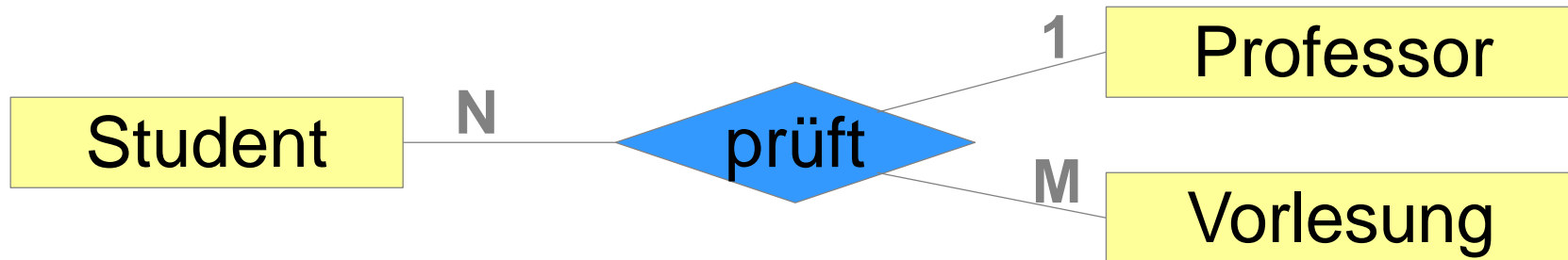
Abstraktion der DHBW-Organisation:

Setzen Sie nun Ihr entworfenes E/R-Modell gemäß der Abbildungsregeln nach Codd ins relationale Modell mit Tabellen um.

Regel 5 (n-äre Beziehung):

- Jede **n-äre** Beziehung mit mehr als zwei beteiligten Entitätstypen muss als **eigenständige Tabelle** definiert werden.
- Die **Primärschlüssel** der zugehörigen Entitätstypen treten in der Beziehung als **Fremdschlüssel** auf.
- Der **Primärschlüssel der Beziehung** setzt sich normalerweise aus den enthaltenen Fremdschlüsseln zusammen.
Wenn jedoch die Kardinalitätseinschränkungen **EINES** der Entitätstypen 1 ist, dann sollte der Primärschlüssel dieses Fremdschlüsselattribut nicht beinhalten.

Beispiel am Whiteboard entwickeln
(Professor prüft Student in Vorlesung).

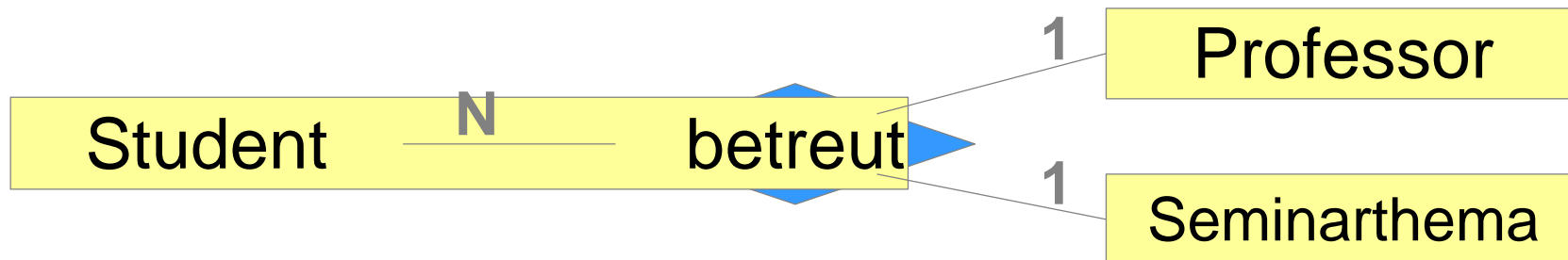


prüft: Student x Vorlesung → Professor

„Ein Student wird für eine Vorlesung durch höchstens einen Professor geprüft.“

In einer Tabelle „Prüfung“ bilden nur die Fremdschlüsselattribute von „Student“ und „Vorlesung“ den Primärschlüssel.

Beispiel am Whiteboard entwickeln
(Student wird von Professor bei einem Thema betreut).



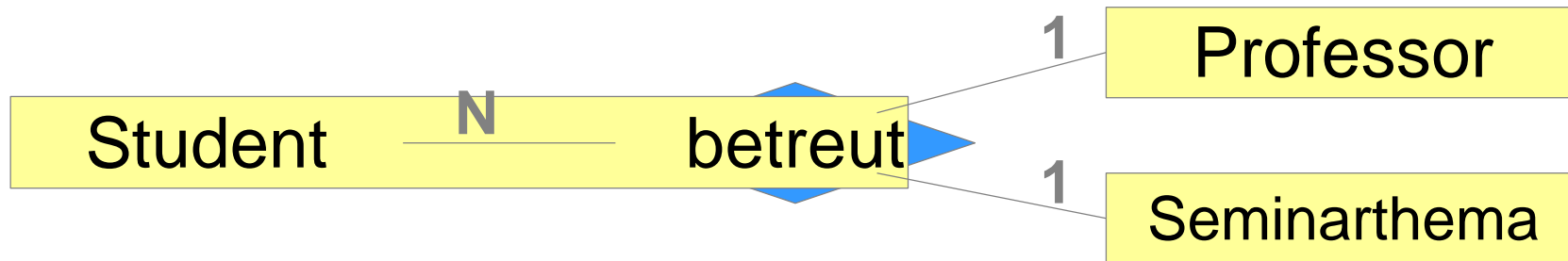
betreut: Professor x Student → Seminarthema

betreut: Seminarthema x Student → Professor

„Ein Student kann ein Seminarthema bei höchstens einem Professor bearbeiten und höchstens ein Thema beim selben Professor ableisten.“

In einer Tabelle „Betreuung“ kann kein Primärschlüssel erzeugt werden!

Mögliche und verbotene Zustände (ohne Primärschlüssel)

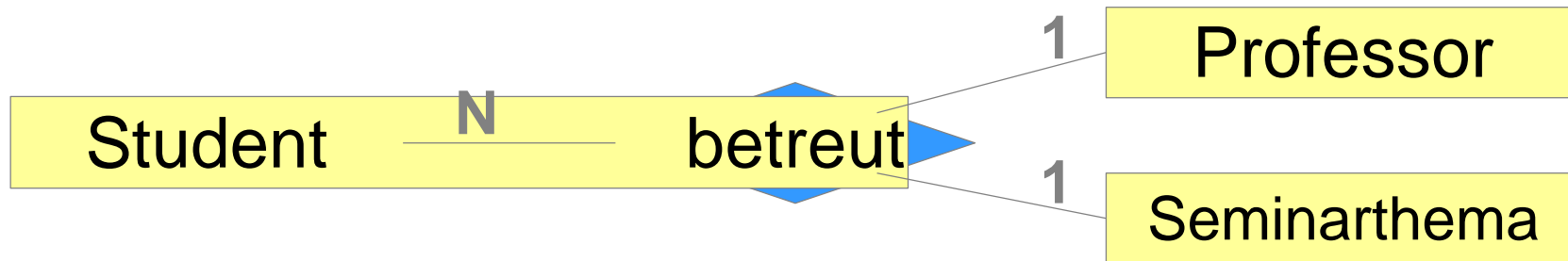


Student	Professor	Seminarthema
123	Prof1	NoSQL
123	Prof1	Android
123	Prof2	NoSQL
 	 	

Die durchgestrichenen Zeilen werden durch die

1-1-Beziehung verhindert.

Mögliche und verbotene Zustände (mit Primärschlüsseln)



<u>Student</u>	<u>Professor</u>	<u>Seminarthema</u>
123	Prof1	NoSQL
123	Prof1	Android
123	Prof2	NoSQL

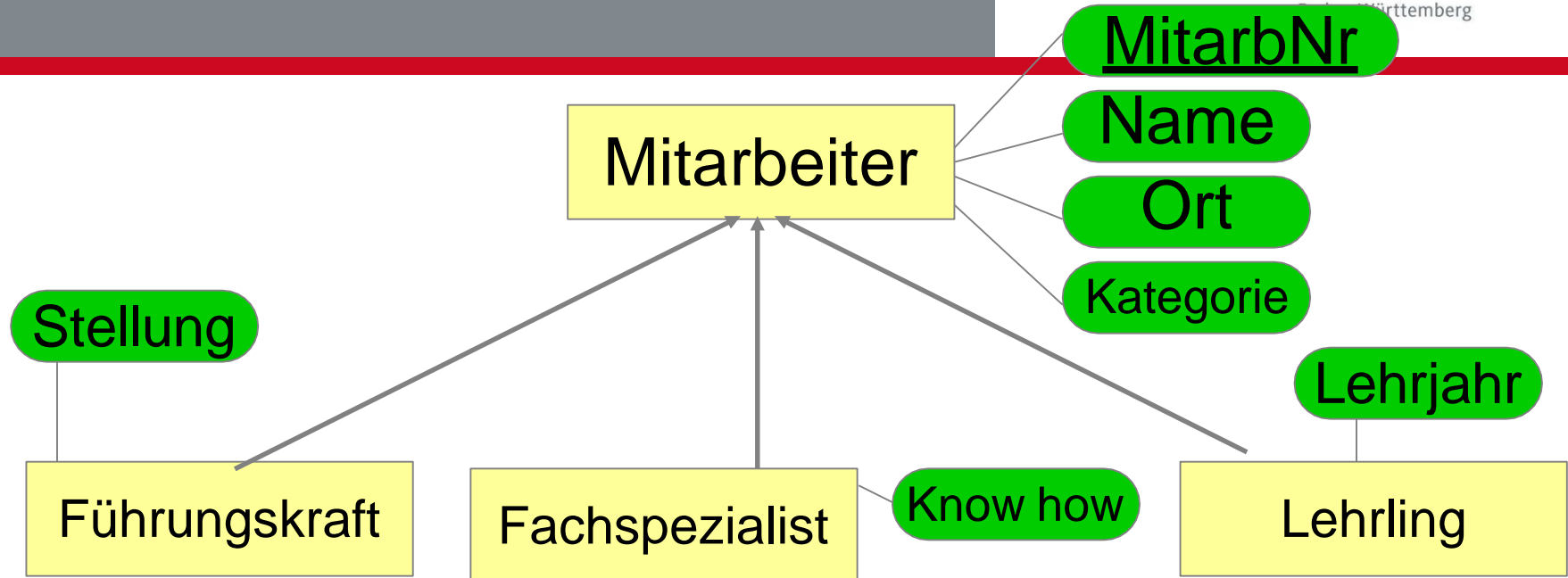
<u>Student</u>	<u>Professor</u>	<u>Seminarthema</u>
123	Prof1	NoSQL
123	Prof1	Android
123	Prof2	NoSQL

Die roten Zeilen wären mit den jeweiligen Primärschlüsseln erlaubt, verstoßen aber gegen die 1-1-N-Beziehung. Es kann kein Primärschlüssel erzeugt werden!

Regel 6 (Generalisierung):

- Jeder Entitätstyp einer Generalisationshierarchie kann über eine eigenständige Tabelle abgebildet werden, wobei der Primärschlüssel der übergeordneten Tabelle auch Primärschlüssel der untergeordneten Tabelle wird.

3.4.5 Abbildungsregeln



Mitarbeiter

<u>MitarbNr</u>	Name	Ort	Kategorie

Führungskraft

MitarbNr	Stellung

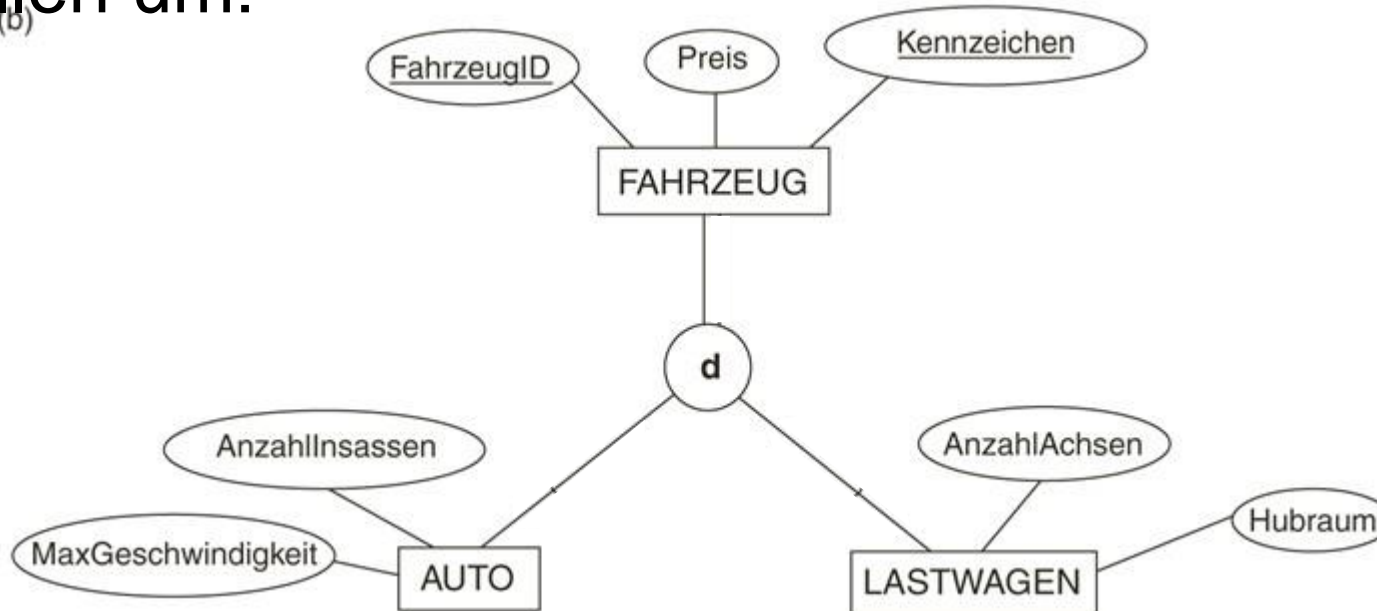
Fachspezialist

<u>MitarbNr</u>	Know-how

Lehrling

<u>MitarbNr</u>	Lehrjahr

Setzen Sie folgendes Modell gemäß Regel 7 in Tabellen um:
(b)



Frage: Welche Varianten sind darüber hinaus denkbar?

- TODO: Tabellen und Aufzählung [Student(MatNr,...)]
- Fremdschlüssel überstreichen
- Durch beispielhafte Daten, die einen gültigen Datenbankzustand darstellen, kann die Korrektheit der Fremdschlüsselbeziehungen abgeleitet werden.

Entitäts-Integrität

- Ein Attribut, das Komponente eines Primärschlüssels ist, darf zu keinem Zeitpunkt den Wert „NULL“ annehmen.

Semantische Integrität

- Soll gewährleisten, dass die Datenbank nur zulässige Sachverhalte im Sinne der Anwendung widerspiegelt. Zum Beispiel dürfen Autos nur Reifen zugeordnet werden, die geeignet sind (Prüfregeln).

Referentielle Integrität

- Soll sicherstellen, dass jeder Fremdschlüsselwert einer Tabelle auch einem Schlüsselwert in der referenzierten Tabelle entspricht (Dead Link-Vermeidung)

**Vielen Dank für Ihre Aufmerksamkeit
bisher. Wir befinden uns am Ende des
dritten Kapitels!**