

# 184.686 VU Datenbanksysteme

## Das Entity-Relationship-Modell

Katja Hose

Institut für Logic and Computation

Sommersemester 2024



Informatics

# Lernziele

## Ziele

- Erstellung nicht-trivialer ER-Diagramme
- Analysieren, ob ein ER-Diagramm gut oder schlecht ist
- Relationen aus ER-Diagrammen ableiten
- Korrekte Verwendung von einer konkreten ER-Notation

## Motivation

- ER-Diagramme sind weit verbreitet.
- Das ER-Modell ist einfach zu erlernen.  
*Viel einfacher als UML.*
- Ein ER-Diagramm ist ein gutes Kommunikationsmittel.  
*Wir sprechen die gleiche Sprache.*

- 1 Datenbankentwurf
  - Schritte des Datenbankentwurfs
  - Datenbankentwurf am Beispiel
- 2 Grundkonzepte
  - Beispielszenarien
  - Entitytypen
  - Attribute
  - Beziehungstypen
- 3 Eigenschaften von Beziehungstypen
  - Stelligkeit
  - Chen-Notation (Funktionalität)
  - Participation Constraints
  - Chen-Notation (Funktionalität) bei n-stelligen Beziehungstypen
  - $[min, max]$ -Notation (Kardinalität)
- 4 Zusätzliche Konzepte
  - Schwache Entitytypen

- Der ISA-Beziehungstyp

## 5 Alternative Notationen

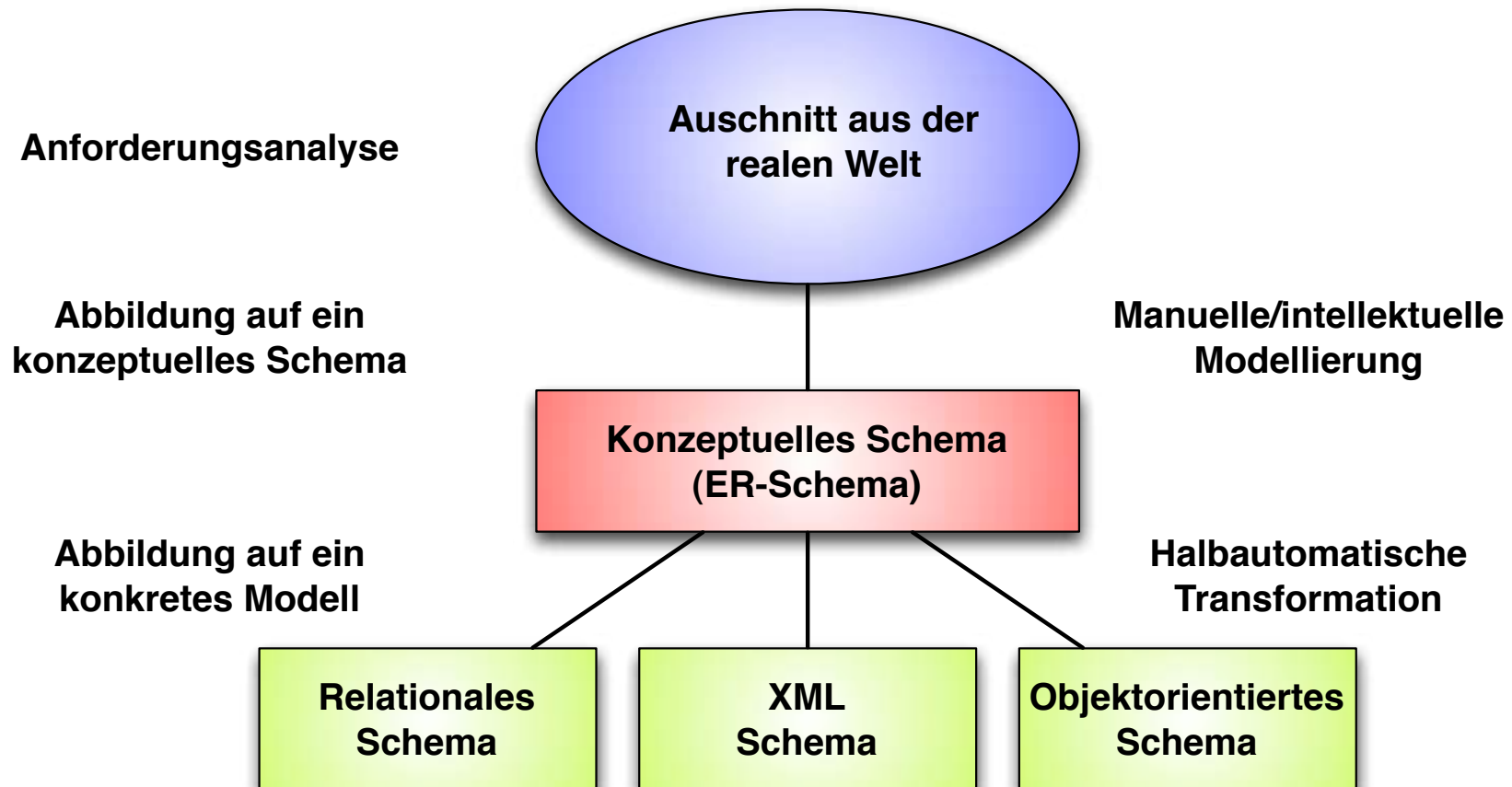
## 6 Relationen aus Grundkonzepten ableiten

- Entitytypen
- Beziehungstypen

## 7 Relation aus zusätzlichen Konzepten ableiten

- Schwache Entitytypen
- Rekursive Beziehungstypen
- N-äre Beziehungstypen
- Spezielle Attribute
- Generalsierung

# Schritte des Datenbankentwurfs



# Schritt 1: Anforderungsanalyse



## Anforderungsanalyse

### Anforderungen

- „Studierende nehmen an Vorlesungen teil“
- „Professor:innen bieten Vorlesungen an“
- „Studierende werden durch die Matrikelnummer eindeutig identifiziert“
- ...

<http://www.studyguide.aau.dk/>  
<http://en.wikipedia.org/>  
<http://da.wikipedia.org/>  
<http://www.cs.aau.dk/research/DPT/>

# Schritt 1: Anforderungsanalyse – Objektbeschreibung

## Angestellte der Universität

Attribute: PersonalNummer, Gehalt, Rang

### PersonalNummer

- Typ: char
- Länge: 9
- Wertebereich:  
0...999.999.99
- Verfügbarkeit: 100%
- Identifizierend: ja

### Gehalt

- Typ: dezimal
- Länge: (8,2)
- Verfügbarkeit: 10%
- Identifizierend: nein

### Rang

- Typ: String
- Länge: 4
- Verfügbarkeit: 100%
- Identifizierend: nein

# Schritt 1: Anforderungsanalyse – Beziehungsbeschreibung

Beziehung: „prüfen“

## Beteiligte Objekte:

- Vortragende:r als Prüfer:in
- Studierende:r als Prüfling
- Vorlesung als Prüfungsstoff

## Attribute der „prüfen“-Beziehung:

- Datum
- Uhrzeit
- Note



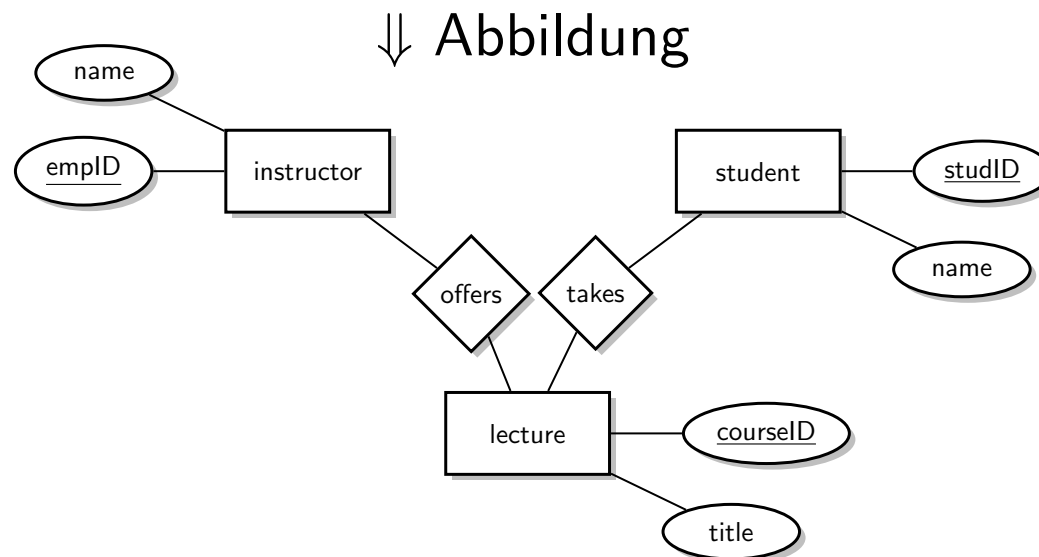
## Schritt 2: Abbildung auf ein konzeptuelles Modell

### Anforderungen

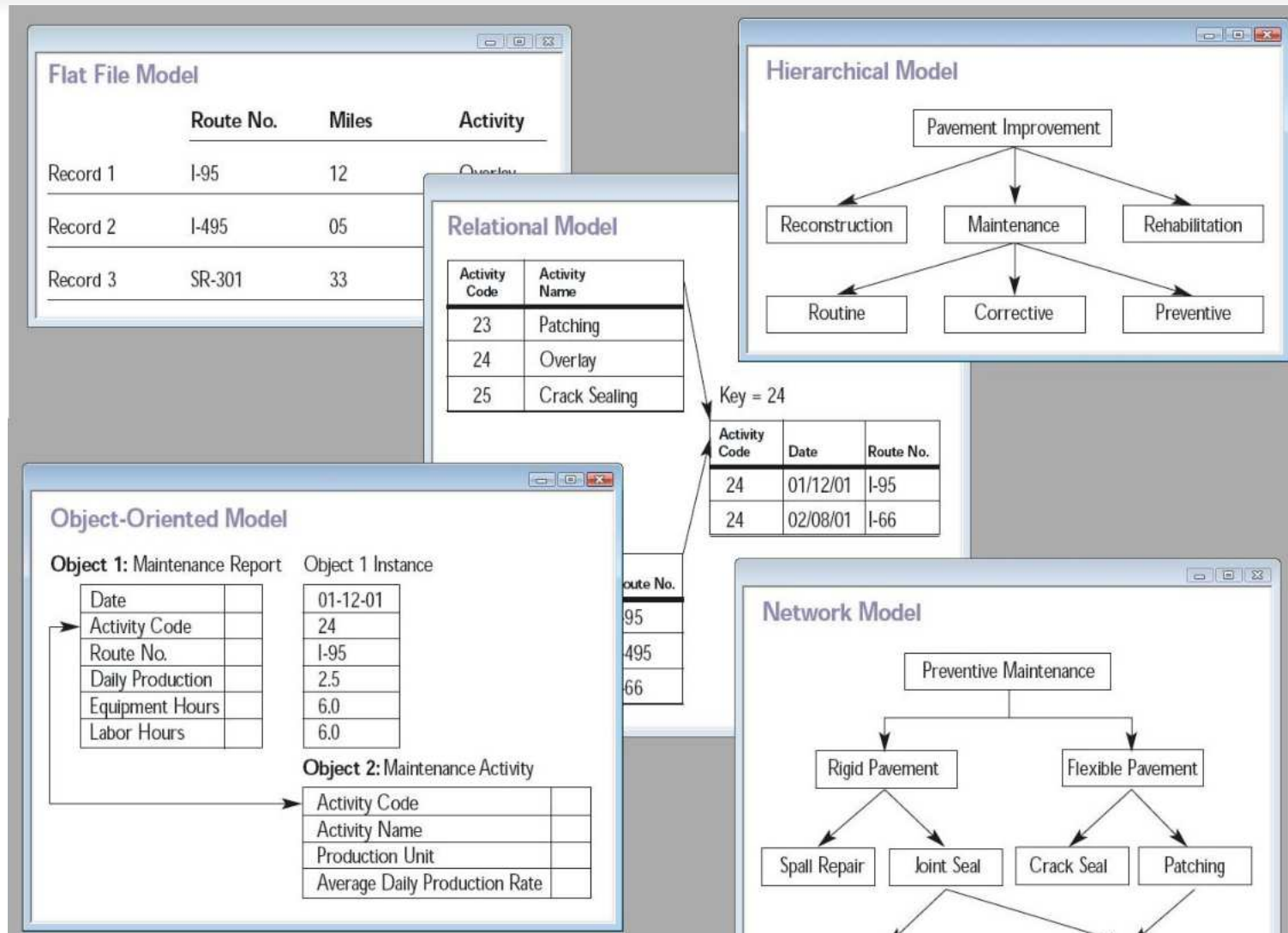
- „Studierende nehmen an Vorlesungen teil“
- „Professor:innen bieten Vorlesungen an“
- „Studierende werden durch die Matrikelnummer eindeutig identifiziert“
- ...

### Funktionale Anforderungen

- Sekretär:in muss Noten eintragen können
- ...

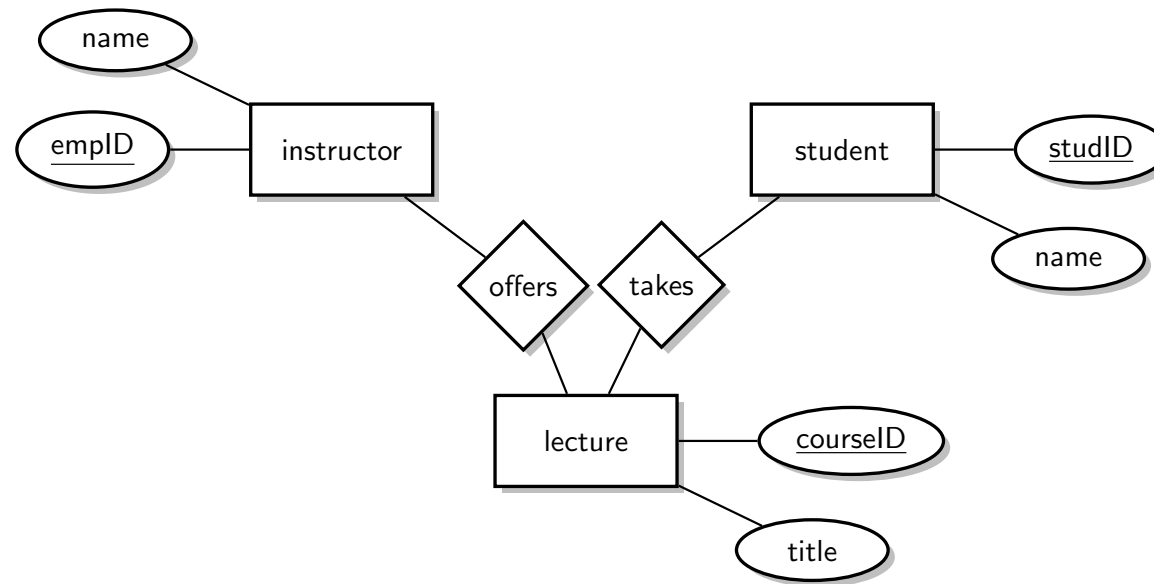


# Schritt 3: Abbildung auf ein konkretes Modell



[http://en.wikipedia.org/wiki/Database\\_model](http://en.wikipedia.org/wiki/Database_model)

## Schritt 3: Abbildung auf das relationale Modell



⇓ Abbildung

### Relationales Modell

- student (studID: integer, name: string)
- takes (studID: integer, courseID: integer)
- lecture (courseID: integer, title: string)

# Schritt 4: Praktische Umsetzung und Implementierung

## Relationales Modell

- student (studID: integer, name: string)
- takes (studID: integer, courseID: integer)
- lecture (courseID: integer, title: string)

⇓ Abbildung

## Tabellen einer DB

student		takes		lecture	
<u>studID</u>	name	<u>studID</u>	<u>courseID</u>	<u>courseID</u>	title
26120	Pedersen	25403	5022	5001	DBS
25403	Hansen	26120	5001	5022	Belief and Knowledge
...	...	...	...	...	...

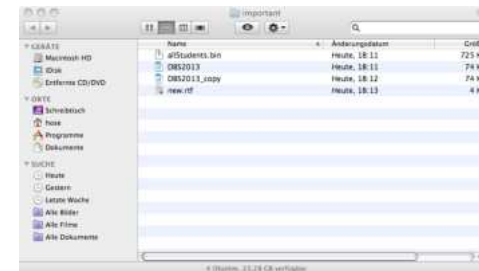
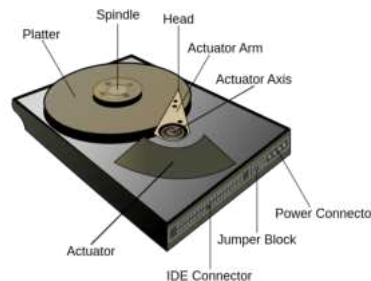
# Schritt 4: Praktische Umsetzung und Implementierung

## Tabellen einer DB

student		takes		lecture	
<u>studID</u>	name	<u>studID</u>	<u>courseID</u>	<u>courseID</u>	title
26120	Pedersen	25403	5022	5001	DBS
25403	Hansen	26120	5001	5022	Belief and Knowledge
...	...	...	...	...	...

↓ Abbildung

## Speicherseiten, Datenstrukturen, Indizes, Dateien, Geräte



<http://en.wikipedia.org>

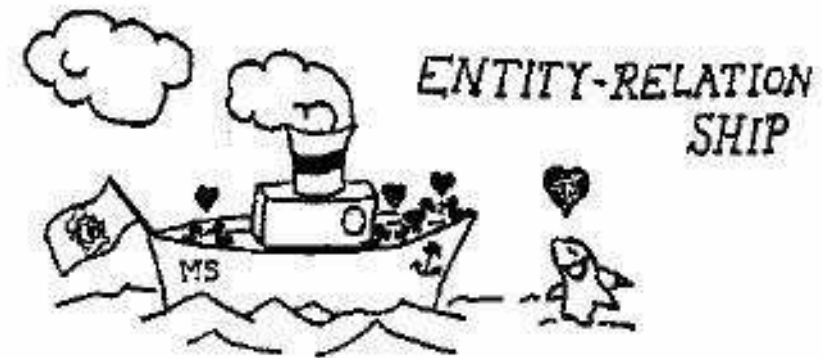
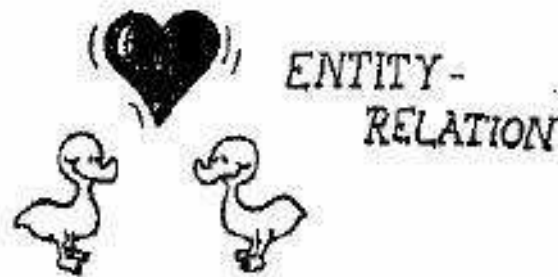
# Schritte des Datenbankentwurfs

- ① Anforderungsanalyse  
*Mit was haben wir es zu tun?*
- ② Abbildung auf ein konzeptuelles Modell (konzeptueller Entwurf)  
*Welche Daten und Zusammenhänge müssen erfasst werden?*
- ③ Abbildung auf ein konkretes Modell (logischer Entwurf)  
*Wie müssen die Daten in einem bestimmten Modell strukturiert werden (hier: das relationale Modell)?*
- ④ Umsetzung und Implementierung (Physischer Entwurf)  
*Welche Anpassungen und Optimierungen sieht ein konkretes DBMS vor?*

Ein guter Entwurf vermeidet Redundanz und Unvollständigkeit.

- 1 Datenbankentwurf
- 2 Grundkonzepte**
  - Beispielszenarien
  - Entitytypen
  - Attribute
  - Beziehungstypen
- 3 Eigenschaften von Beziehungstypen
- 4 Zusätzliche Konzepte
- 5 Alternative Notationen
- 6 Relationen aus Grundkonzepten ableiten

# Entity-Relationship-Modell (ER-Modell)



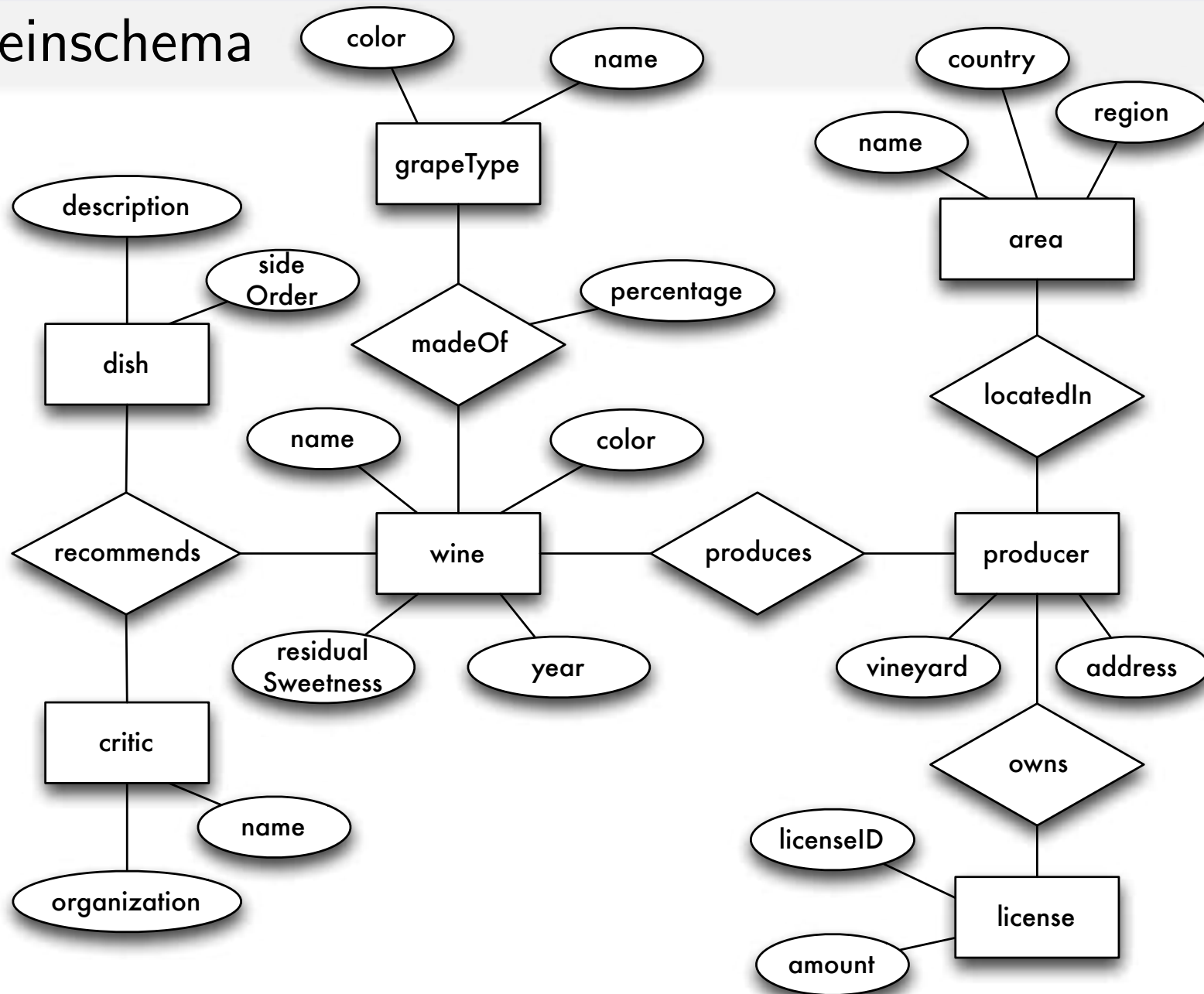


# Beispielszenarien

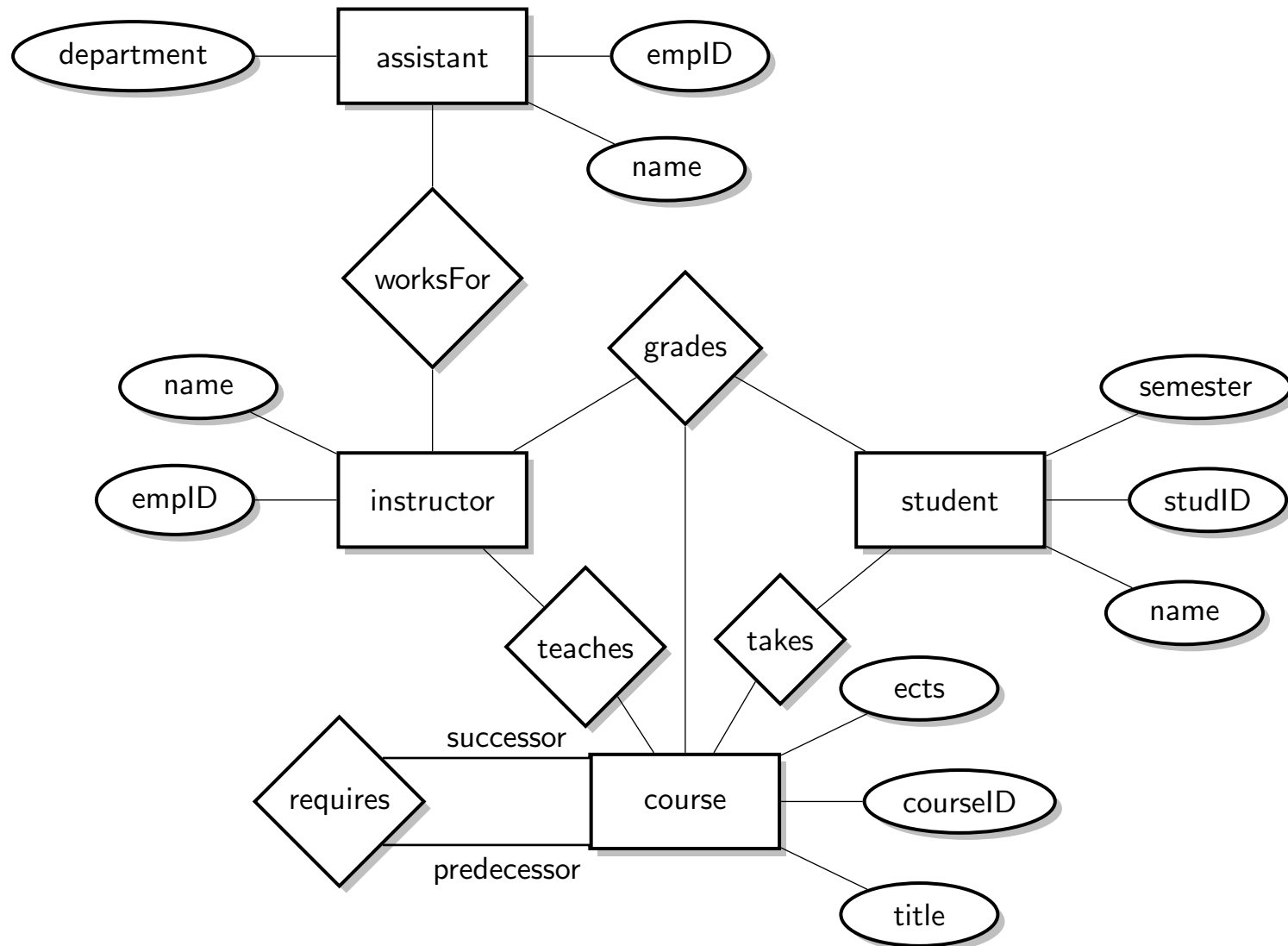
Zwei Beispielszenarien in der Vorlesung:

- Universität: students, instructors, courses,...
- Wein: wine, producers, regions,...

# Weinschema



# Universitätsschema



# Entitys und Entitytypen

- **Entitys** sind Objekte der realen Welt, über die wir Informationen abspeichern wollen.
- Nur Eigenschaften der Entitys können in einer Datenbank gespeichert werden (Beschreibung), nicht die Entitys selbst!

Entitys werden eingeteilt in **Entitytypen**.



wine

A rectangular box with a black border containing the word "wine" in a black, sans-serif font, representing an entity set.

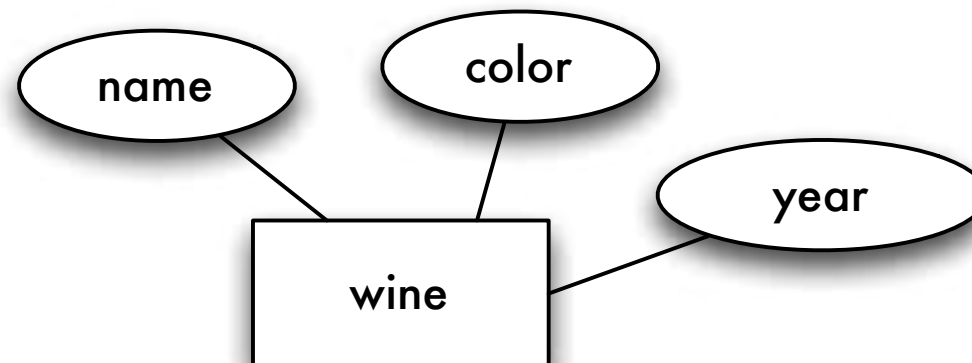
Eine **Entitymenge** ist eine konkrete Menge von Entitys des gleichen Entitytyps.

Die zwei Begriffe Entitymenge (entity set) und Entitytypen (entity type) werden oft als Synonyme verwendet.

# Attribute

**Attribute** modellieren Eigenschaften von Entitys oder auch Beziehungen.

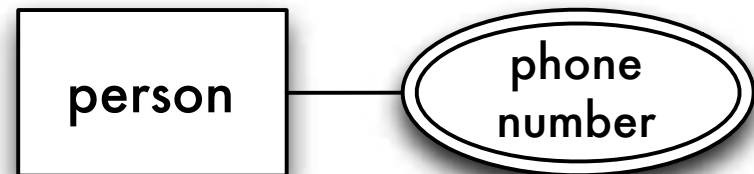
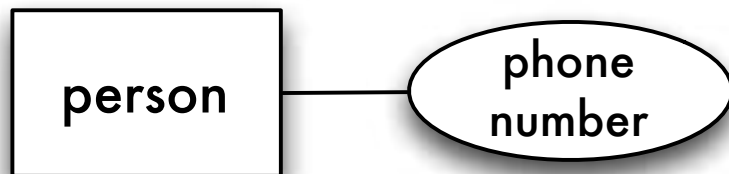
- Alle Entitys eines Entitytyps haben dieselben Arten von Eigenschaften.
- Attribute werden für Entitytypen deklariert.
- Attribute haben eine Domäne bzw. Wertemenge.



# Attribute

**Single-valued (einwertige)** vs. **multi-valued (mehrwertige)** Attribute

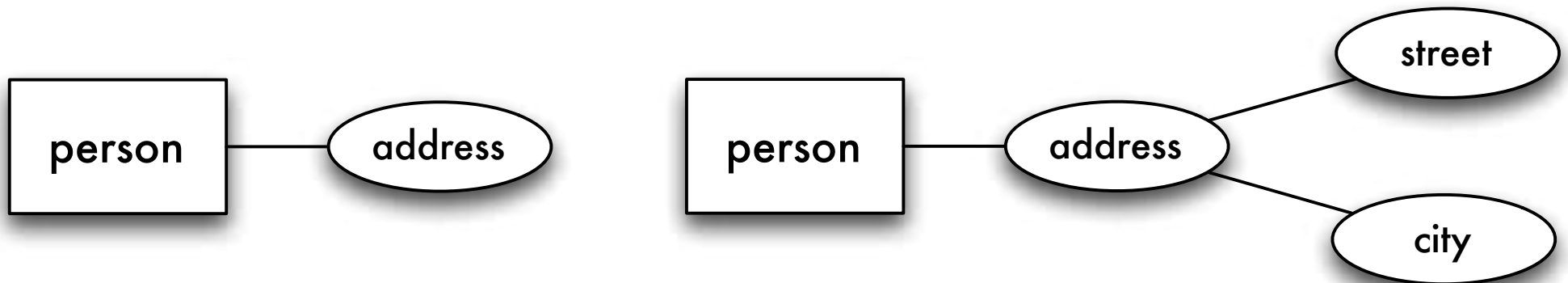
- Eine Person kann mehrere Telefonnummern haben (oder eine einzige)



# Attribute

**Simple (einfache)** Attribute vs. **composite (zusammengesetzte)** Attribute

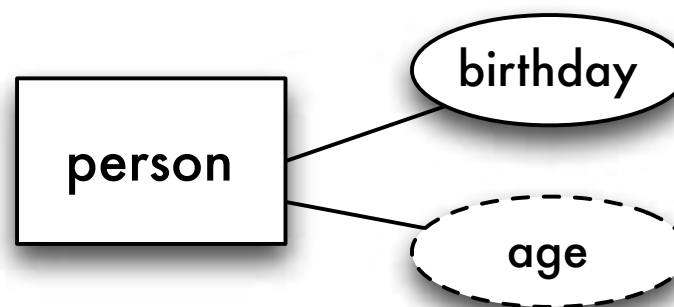
- Eine Adresse kann als String modelliert werden oder sich aus Straße und Stadt zusammensetzen



# Attribute

Gespeicherte Attribute vs. **derived (abgeleitete)** Attribute

- Zum Beispiel: Alter





# Schlüssel

Ein **(Super-)Schlüssel** besteht aus einer Untermenge von Attributen eines Entitytyps  $E(A_1, \dots, A_m)$

$$\{S_1, \dots, S_k\} \subseteq \{A_1, \dots, A_m\}$$

Die Attribute  $S_1, \dots, S_k$  eines Schlüssels werden **Schlüsselattribute** genannt.

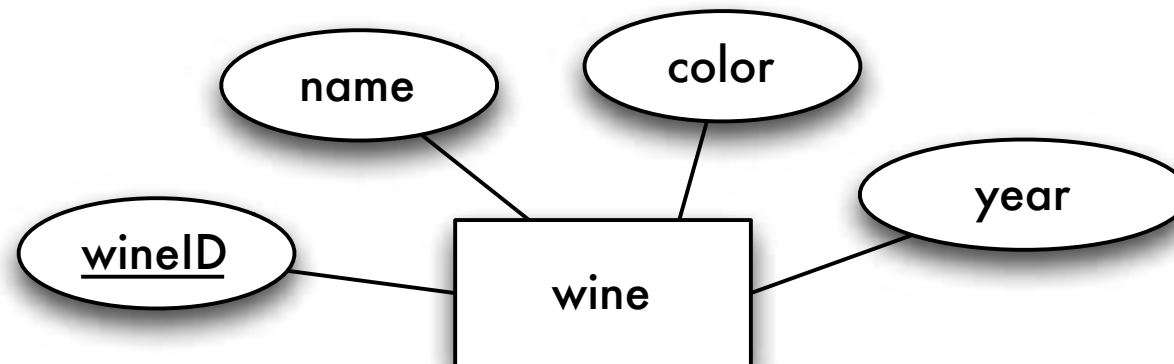
Die Werte der Schlüsselattribute identifizieren zusammen eindeutig ein bestimmtes Entity.

Ein **Schlüsselkandidat** ist ein *minimaler* Schlüssel.

Gibt es mehrere **Schlüsselkandidaten**, so wird einer als **Primärschlüssel** ausgewählt.

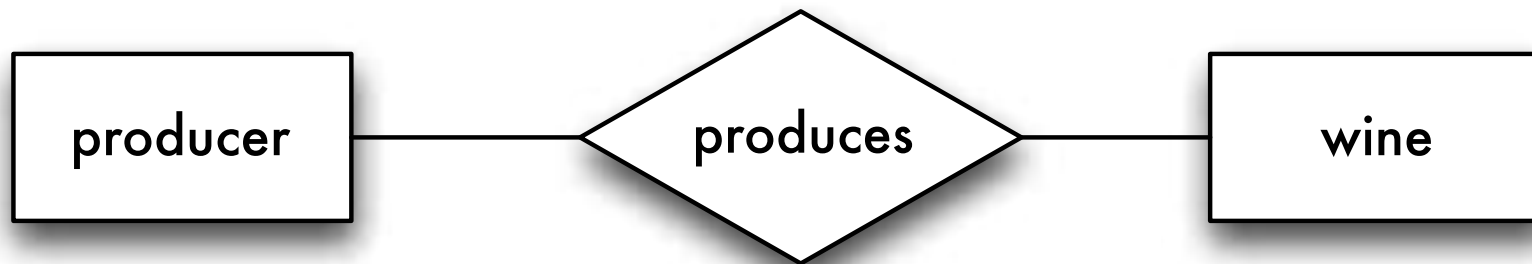
# Primärschlüssel

Attribute des Primärschlüssels werden durch Unterstreichen markiert.



# Beziehung und Beziehungstyp

- Eine **Beziehung** beschreibt die Verbindung zwischen Entitys.
- Beziehungen zwischen Entitys werden zu **Beziehungstypen** zusammengefasst.



Eine konkrete Verbindung zwischen zwei oder mehreren Entitys wird **Beziehung (-instanz)** genannt. Eine **Beziehungsmenge** ist eine Menge von Beziehungsinstanzen.

Die zwei Begriffe Beziehungsmenge (relationship set) und Beziehungstyp (relationship type) werden oft als Synonyme verwendet.

# Mathematische Auffassung einer Beziehung

Ein Beziehungstyp  $R$  zwischen den Entitytypen  $E_1, E_2, \dots, E_n$  kann als **Relation** im mathematischen Sinn angesehen werden.

**Ausprägung** des Beziehungstyps  $R$ :

$$R \subseteq E_1 \times E_2 \times \dots \times E_n$$

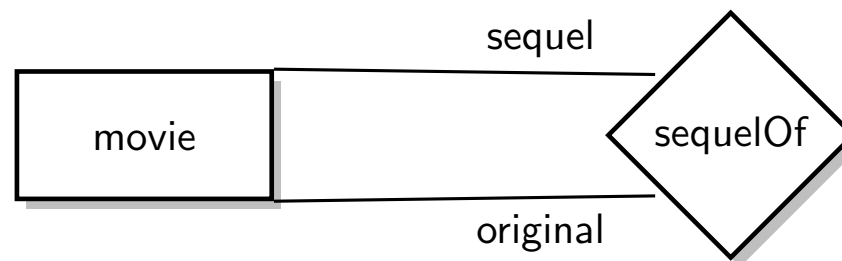
Ein Element  $(e_1, e_2, \dots, e_n) \in R$  bezeichnet man als **Instanz** des Beziehungstyps, wobei  $e_i \in E_i$  für alle  $1 \leq i \leq n$ .

Anmerkung: Diese Notation umfasst keine Attribute von Beziehungstypen.

# Rollennamen und Rekursive Beziehungstypen

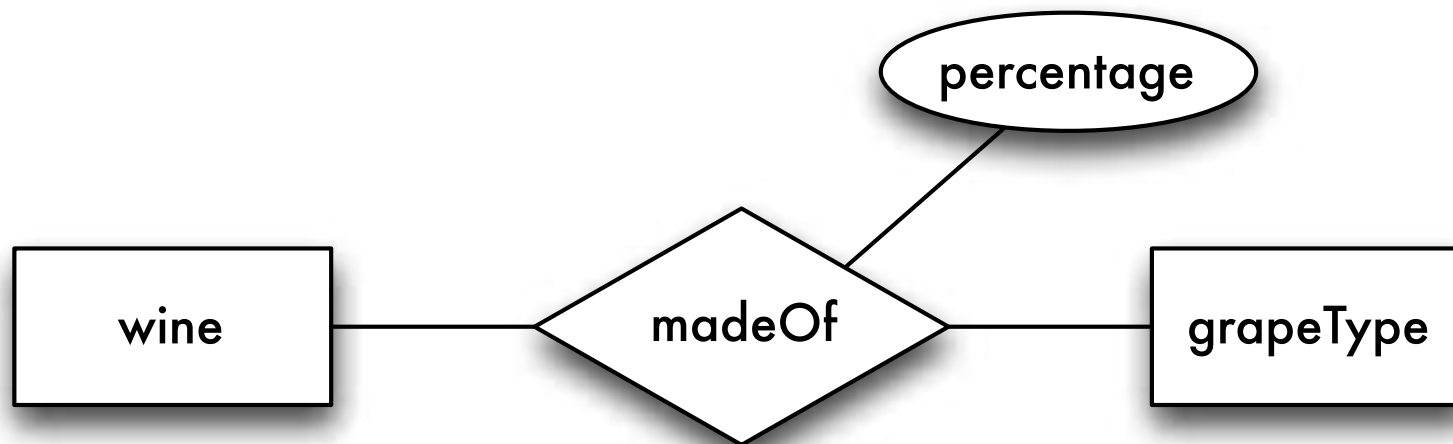
**Rollennamen** sind optional und werden zur weiteren Charakterisierung einer Beziehung verwendet.

- Vor allem sinnvoll bei einem rekursiven Beziehungstyp, bei dem ein Entitytyp mehrfach an einem Beziehungstyp beteiligt ist.



# Beziehungsattribute

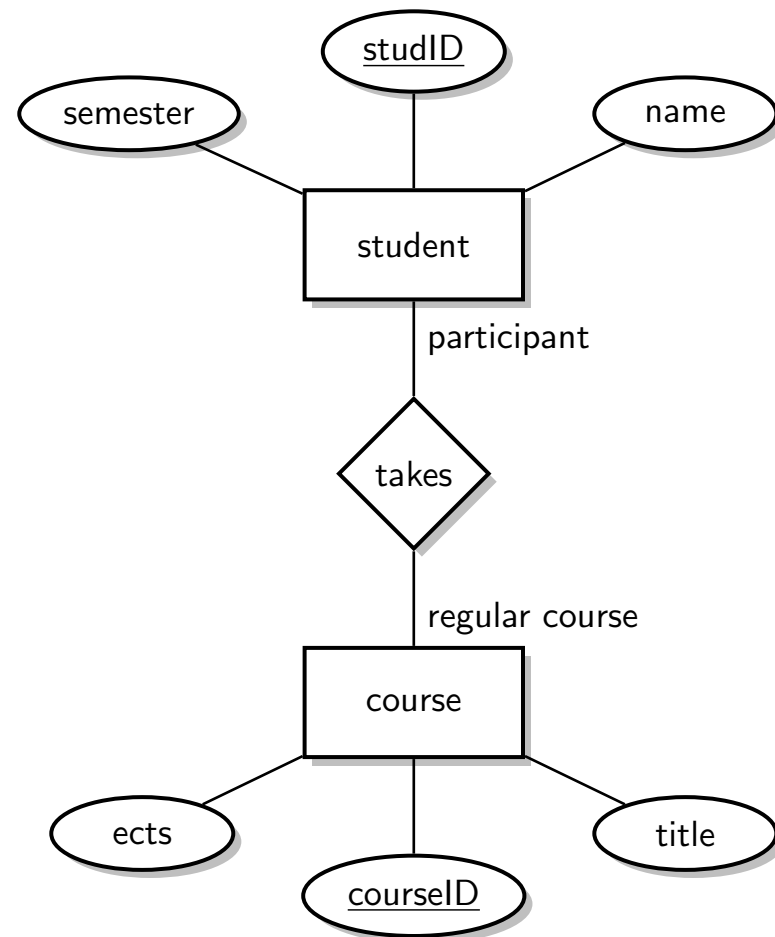
Beziehungstypen können ebenfalls Attribute besitzen.



# Zusammenfassung: Grundbegriffe

*Studierende nehmen an Vorlesungen teil*

1. Entity → Entitytyp
2. Beziehung → Beziehungstyp
3. Attribute (Eigenschaften)
4. Primärschlüssel
5. Rollen



- 1 Datenbankentwurf
- 2 Grundkonzepte
- 3 Eigenschaften von Beziehungstypen**
  - Stelligkeit
  - Chen-Notation (Funktionalität)
  - Participation Constraints
  - Chen-Notation (Funktionalität) bei n-stelligen Beziehungstypen
  - $[min, max]$ -Notation (Kardinalität)
- 4 Zusätzliche Konzepte
- 5 Alternative Notationen



# Merkmale von Beziehungstypen

## Stelligkeit bzw. Grad

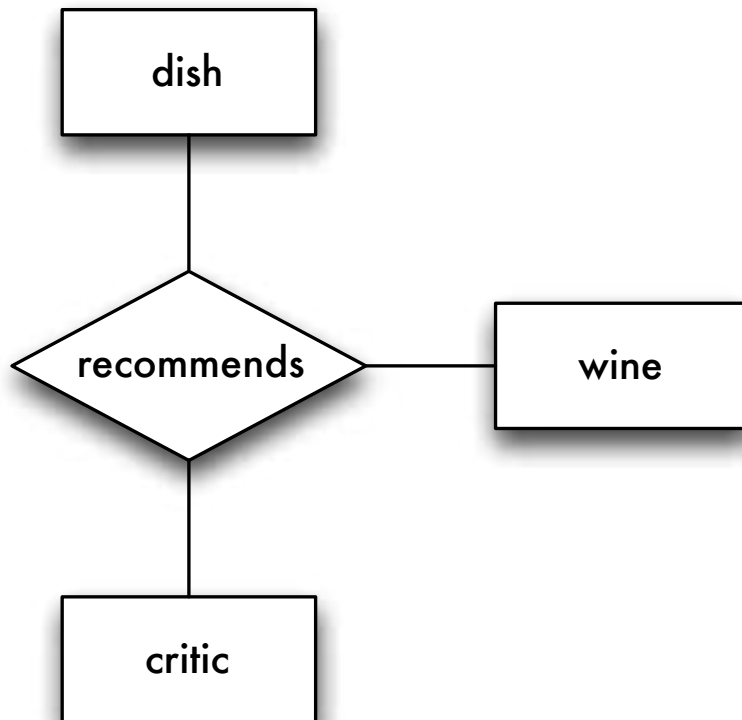
- Anzahl der beteiligten Entitytypen
- Häufig: binär
- Weniger häufig: ternär
- Allgemein: n-stellig

## Funktionalität / Kardinalität / Participation Constraints

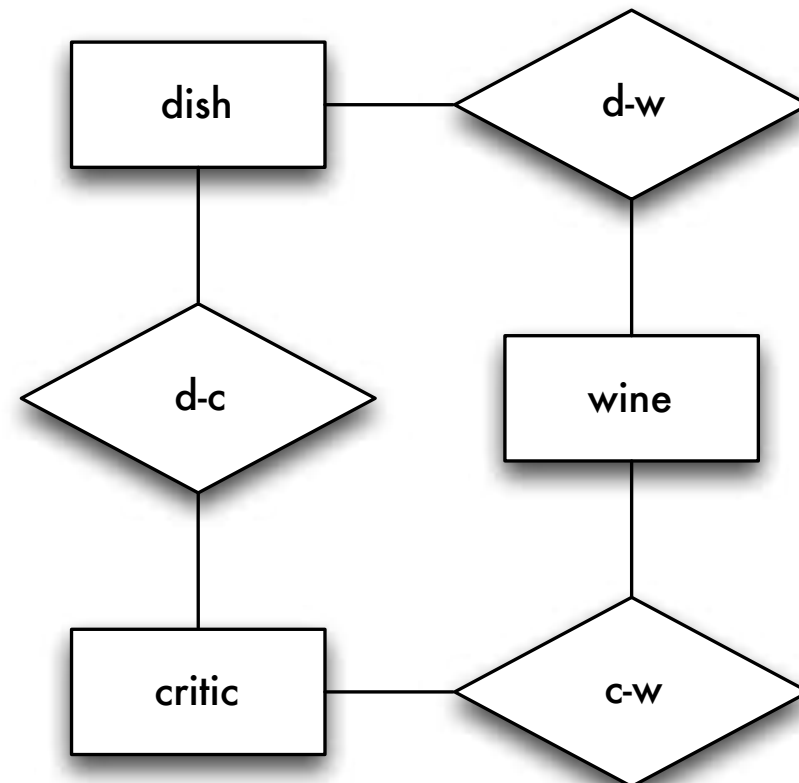
- Anzahl von Entitys, die an einer Beziehung teilnehmen
- Funktionalität (Chen-Notation): 1:1, 1:N, N:M
- Participation Constraints: partiell oder total
- Kardinalität ([min,max]-Notation): [min,max]

# Zwei- vs. mehrstellige Beziehungen

## Ternärer Beziehungstyp

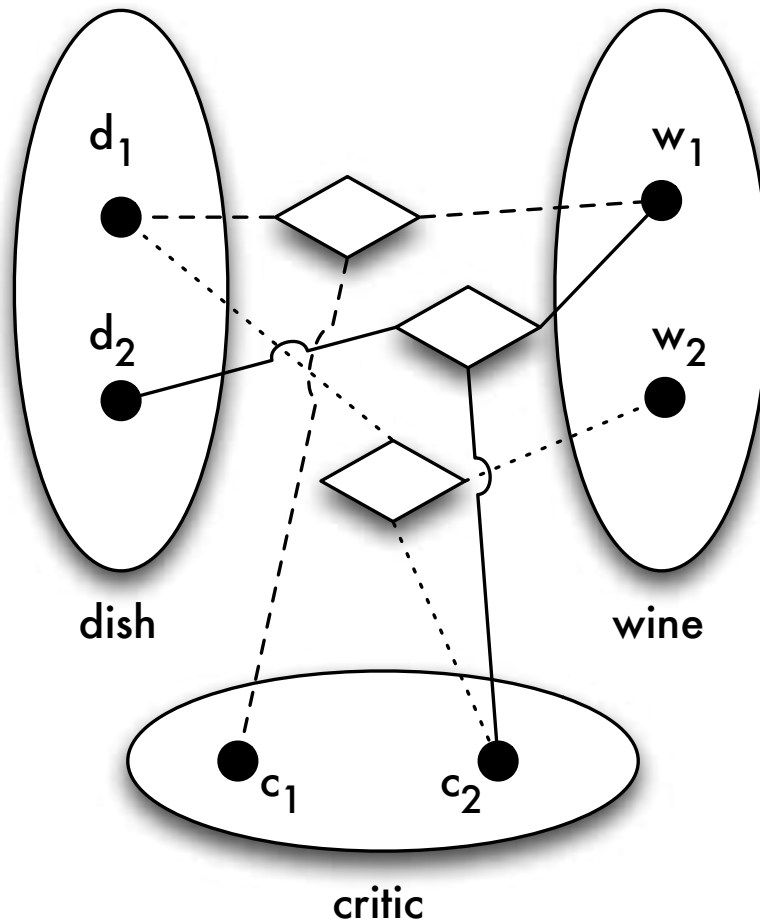


## Drei binäre Beziehungstypen

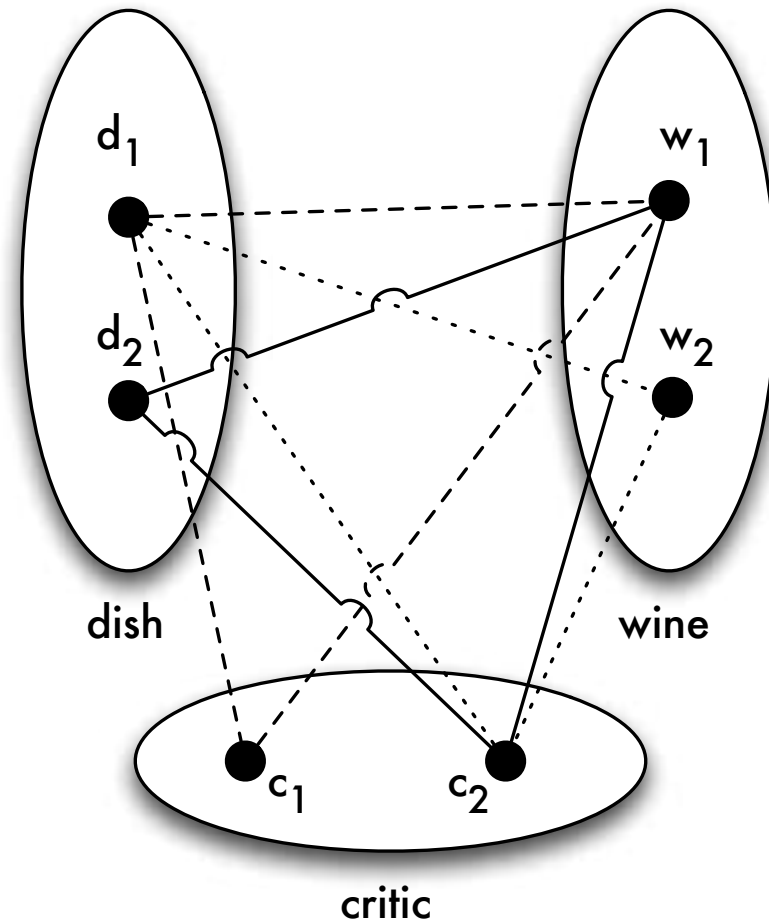


# Mehrstellige Beziehungstypen

## Ternärer Beziehungstyp

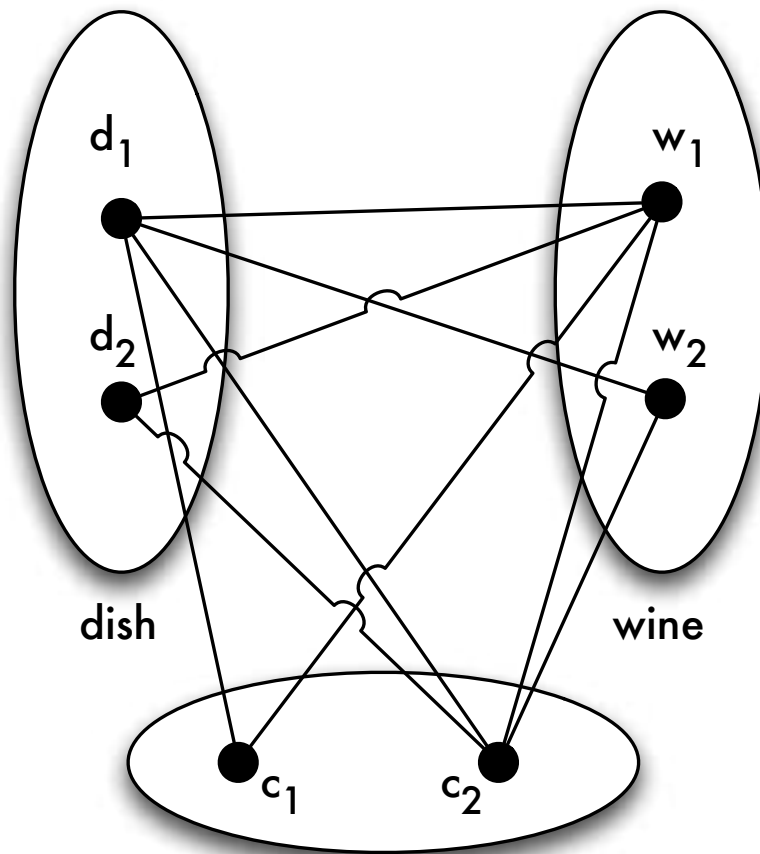


## Drei binäre Beziehungstypen



# Rekonstruktion der Ausprägungen

## Binäre Beziehungstypen

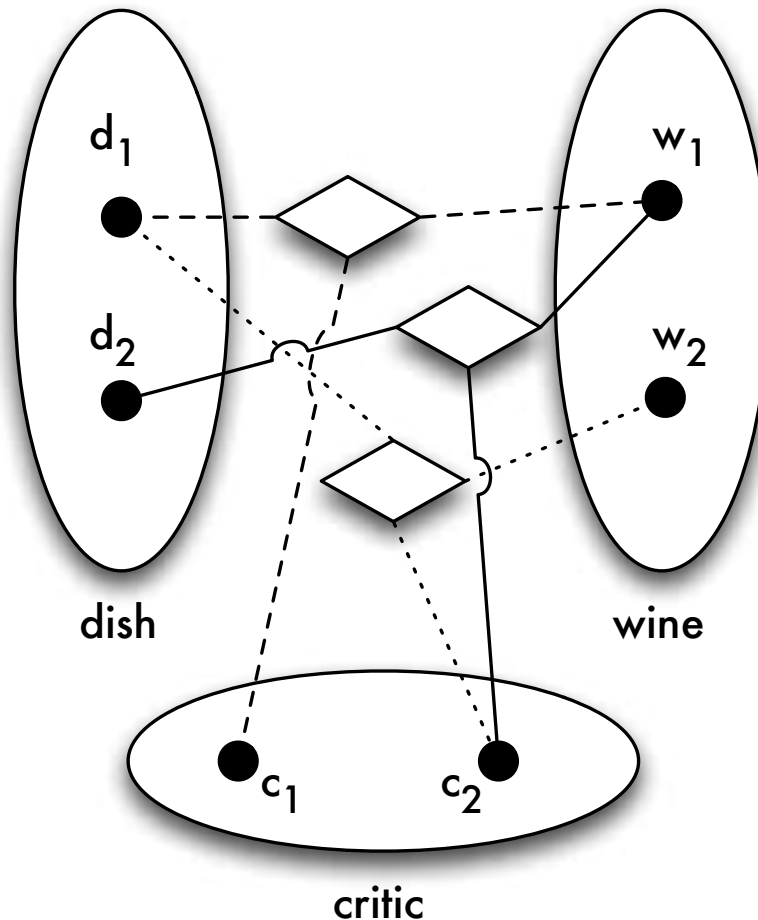


## Rekonstruierbare Beziehungen

- $d_1 - c_1 - w_1$
- $d_1 - c_2 - w_2$
- $d_2 - c_2 - w_1$
- aber auch:  $d_1 - c_2 - w_1$

# Mehrstellige Beziehungstypen

## Ternärer Beziehungstyp



Mit binären Beziehungstypen können wir die folgende Beziehung rekonstruieren

$$d_1 - c_2 - w_1$$

nicht jedoch mit einem ternären Beziehungstypen!

# Merkmale von Beziehungstypen

## Stelligkeit bzw. Grad

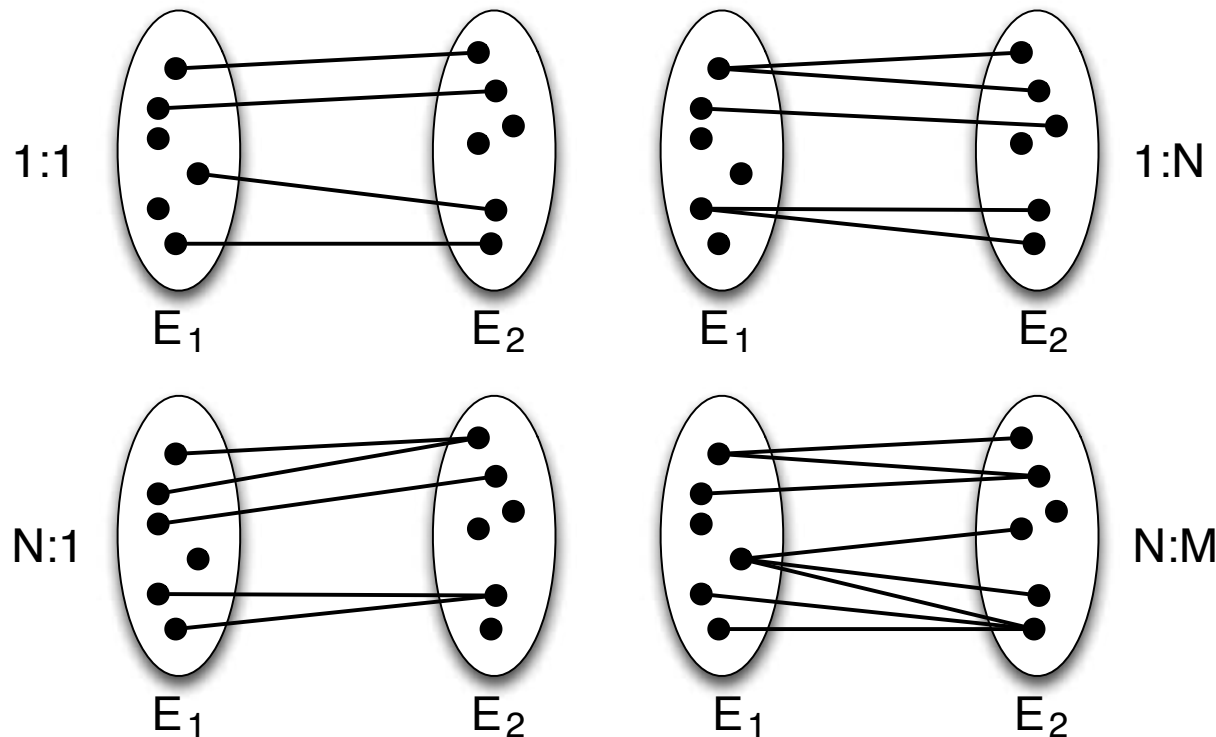
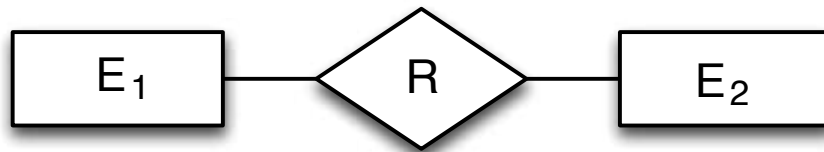
- Anzahl der beteiligten Entitytypen
- Häufig: binär
- Weniger häufig: ternär
- Allgemein: n-stellig

## Funktionalität / Kardinalität / Participation Constraints

- Anzahl der Entitys, die an einer Beziehung teilnehmen
- Funktionalität (Chen-Notation): 1:1, 1:N, N:M
- Participation Constraints: partiell oder total
- Kardinalität ([min,max]-Notation): [min,max]

# Chen-Notation (Funktionalität)

$$R \subseteq E_1 \times E_2$$



- 1: höchstens eins
- N: beliebig viele

# Funktionale Beziehungstypen

1:1, 1:N und N:1 können als **partielle Funktionen** angesehen werden (oft auch eine totale Funktionen)

1:1 Beziehungstypen:  $R : E_1 \rightarrowtail E_2$  und  $R^{-1} : E_2 \rightarrowtail E_1$

1:N Beziehungstypen:  $R^{-1} : E_2 \rightarrowtail E_1$

N:1 Beziehungstypen:  $R : E_1 \rightarrowtail E_2$  auch **funktionale Beziehung** genannt.

Die „Richtung“ ist entscheidend!

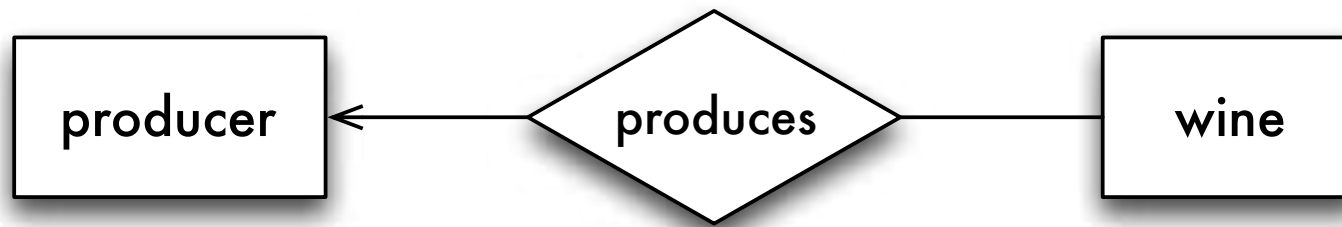
Die Funktion geht immer vom „N“-Entitytyp zum „1“-Entitytyp.

In dieser Vorlesung unterscheiden wir in diesem Zusammenhang nicht zwischen partiellen ( $\rightarrowtail$ ) und totalen Funktionen ( $\rightarrow$ ). Wir schreiben daher einfach  $\rightarrow$ .

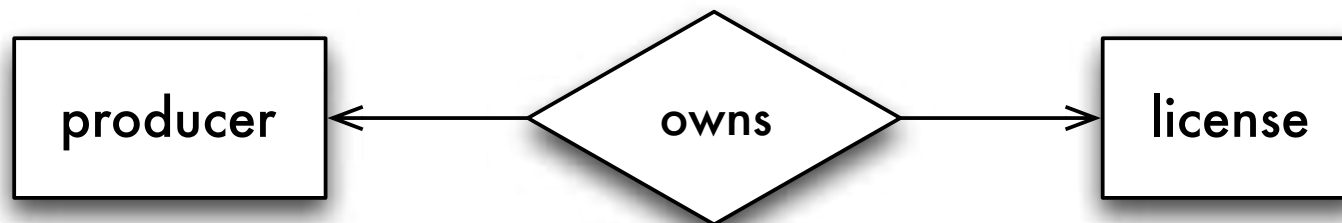


# Notation funktionaler Beziehungstypen

1:N Beziehungstyp



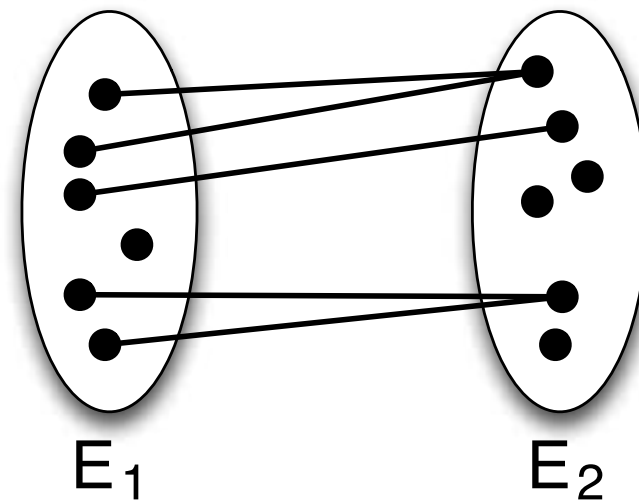
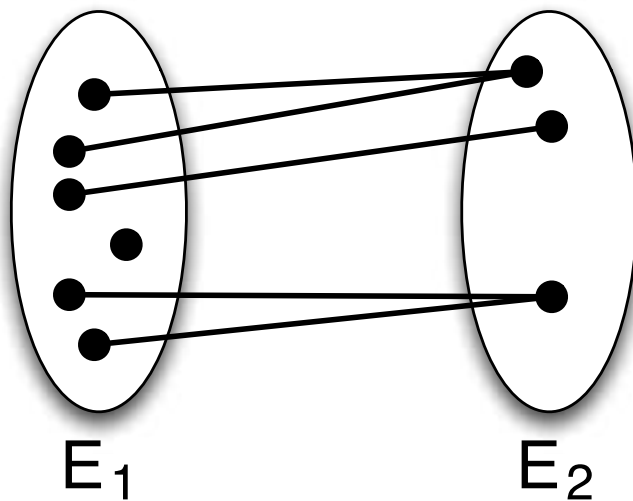
1:1 Beziehungstyp



# Participation Constraints

## Total

Jedes Entity eines Entitytyps **muss** an einer Beziehung teilnehmen, d.h. es kann nicht existieren ohne zu partizipieren ( $E_2$  im linken Beispiel).

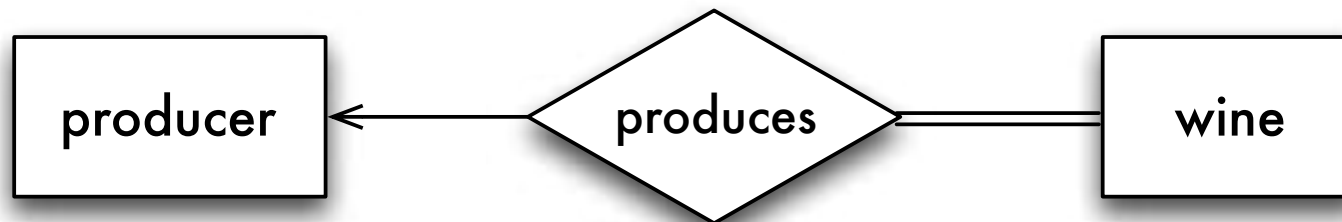


## Partiell

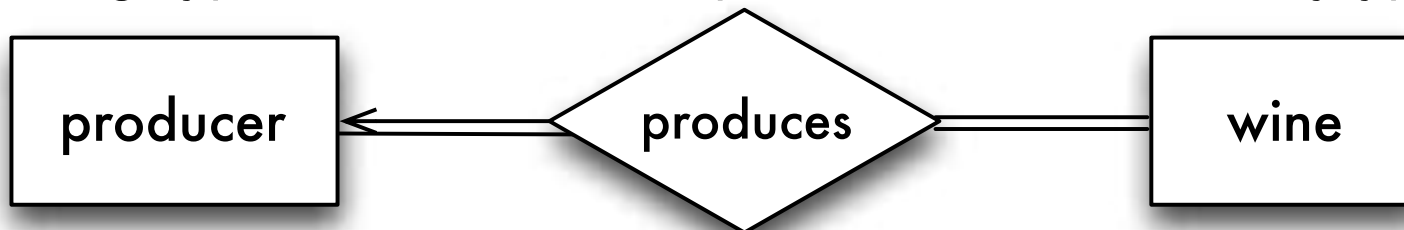
Jedes Entity eines Entitytyps **kann** an einer Beziehung teilnehmen, d.h. es kann ohne Partizipation existieren.

# Graphische Darstellung

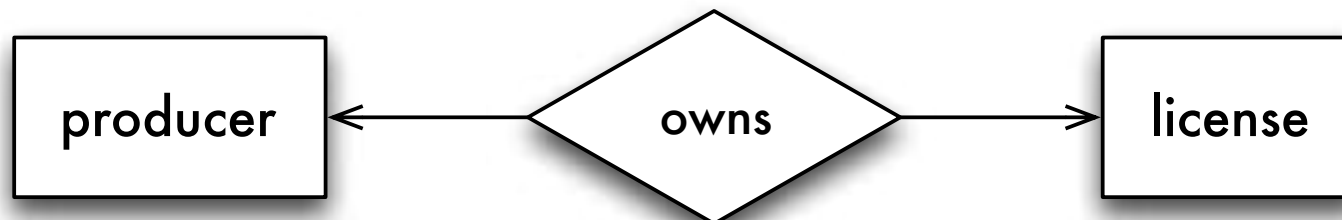
1:N Beziehungstyp mit totaler Partizipation vom Entitytyp wine



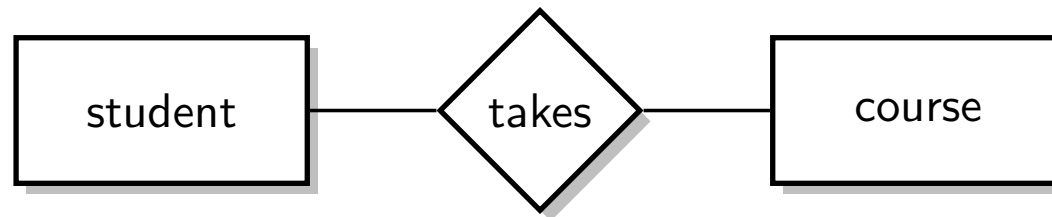
1:N Beziehungstyp mit totaler Partizipation von beiden Entitytypen



1:1 Beziehungstyp mit partieller Partizipation



# Überblick Kardinalitätsverhältnisse



Welcher Beziehungstyp ist es?

N:M

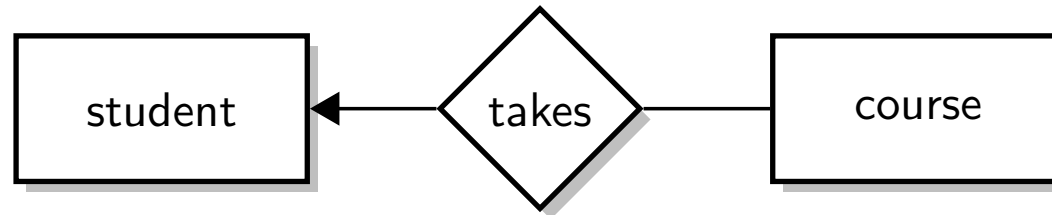
Wie viele Studierende nehmen an einer Vorlesung teil?

beliebig viele

An wie vielen Vorlesungen nimmt ein:e Studierende:r teil?

beliebig viele

# Überblick Funktionalität



Welcher Beziehungstyp ist es?

1:M

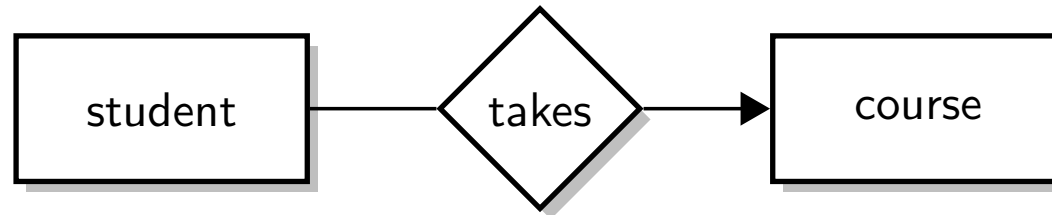
Wie viele Studierende nehmen an einer Vorlesung teil?

maximal ein:e

An wie vielen Vorlesungen nimmt ein:e Studierende:r teil?

beliebig viele

# Überblick Funktionalität



Welcher Beziehungstyp ist es?

N:1

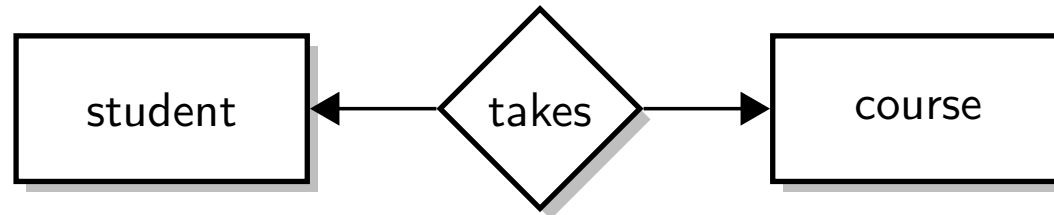
Wie viele Studierende nehmen an einer Vorlesung teil?

beliebig viele

An wie vielen Vorlesungen nimmt ein:e Studierende:r teil?

maximal eine

# Überblick Funktionalität



Welcher Beziehungstyp ist es?

1:1

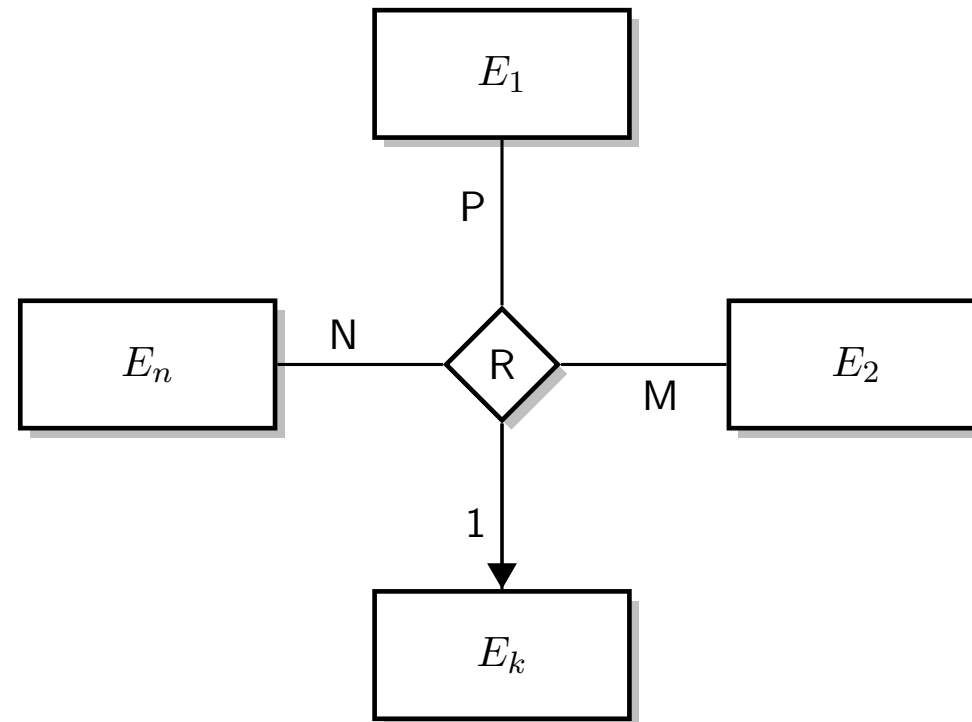
Wie viele Studierende nehmen an einer Vorlesung teil?

maximal ein:e

An wie vielen Vorlesungen nimmt ein:e Studierende:r teil?

maximal eine

# Funktionalitäten bei n-stelligen Beziehungen



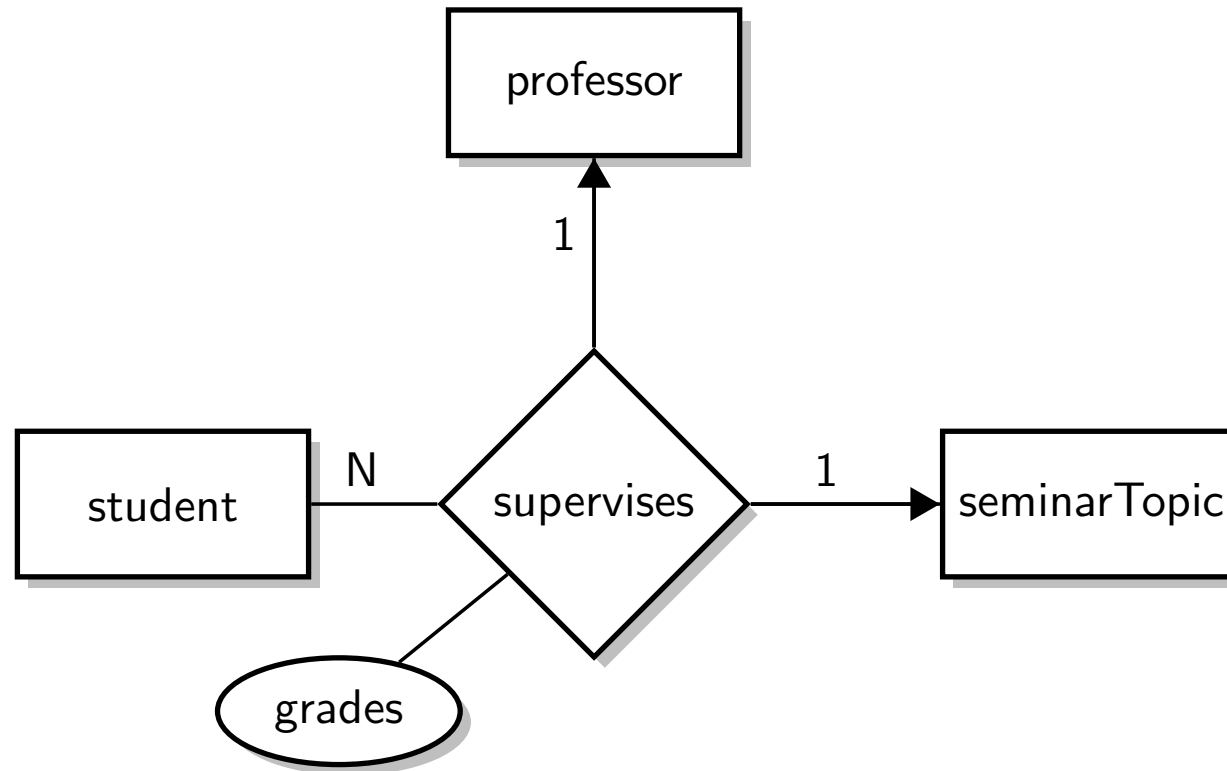
$$R : E_1 \times E_2 \times \dots \times E_{k-1} \times E_{k+1} \times \dots \times E_n \rightarrow E_k$$

## Anmerkung zur Notation im Allgemeinen

Die Verwendung von Pfeilen und 1, N, M, etc. sind äquivalent. Beide anzugeben ist nicht nötig, aber kann manchmal hilfreich sein.



# Beispielbeziehung: supervises



supervises: professor  $\times$  student  $\rightarrow$  seminarTopic

supervises: seminarTopic  $\times$  student  $\rightarrow$  professor

# Merkmale von Beziehungstypen

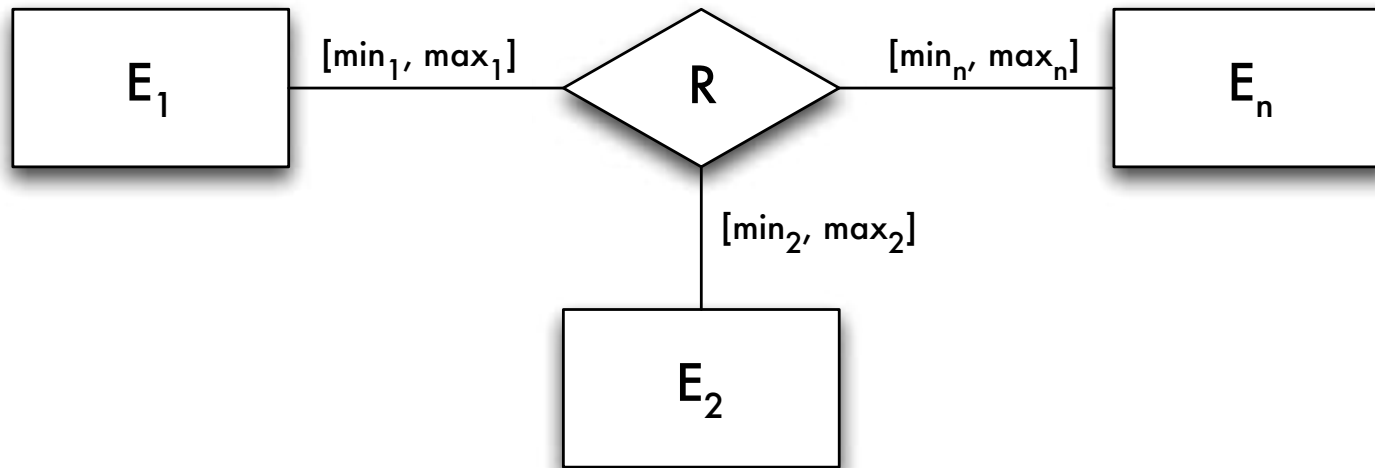
## Stelligkeit bzw. Grad

- Anzahl der beteiligten Entitytypen
- Häufig: binär
- Weniger häufig: ternär
- Allgemein: n-stellig

## Funktionalität / Kardinalität / Participation Constraints

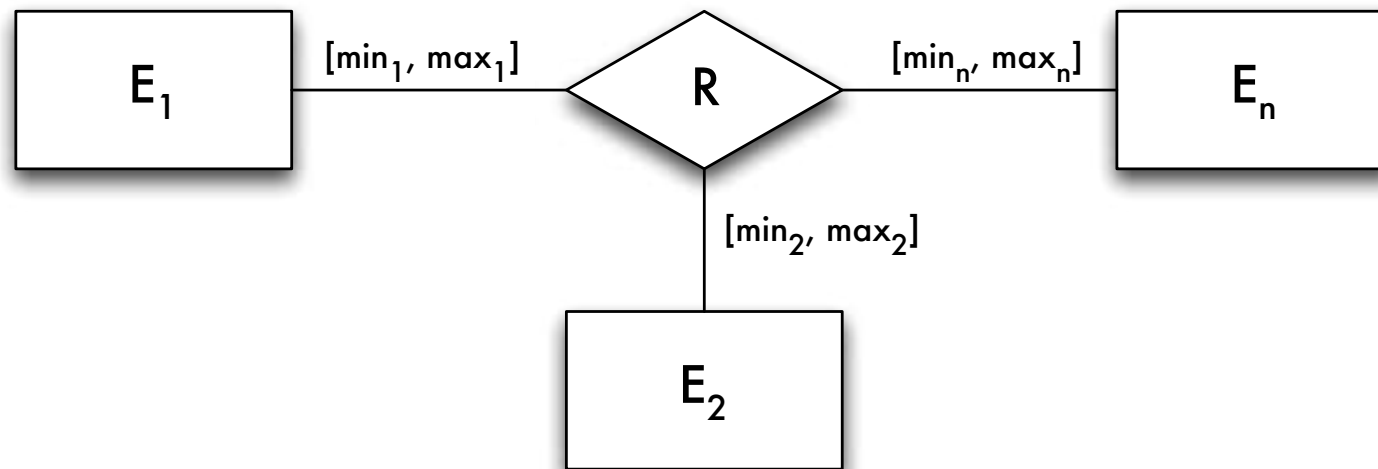
- Anzahl von Entitys, die an einer Beziehung teilnehmen
- Funktionalität (Chen-Notation): 1:1, 1:N, N:M
- Participation Constraints: partiell oder total
- **Kardinalität ( $[min, max]$ -Notation):  $[min, max]$**

# $[min, max]$ -Notation (Kardinalität)



Die  $[min, max]$ -Notation schränkt ein, wie oft ein Entity eines Entitytyps an einer Beziehung teilnehmen kann.

# [min,max]-Notation (Kardinalität)



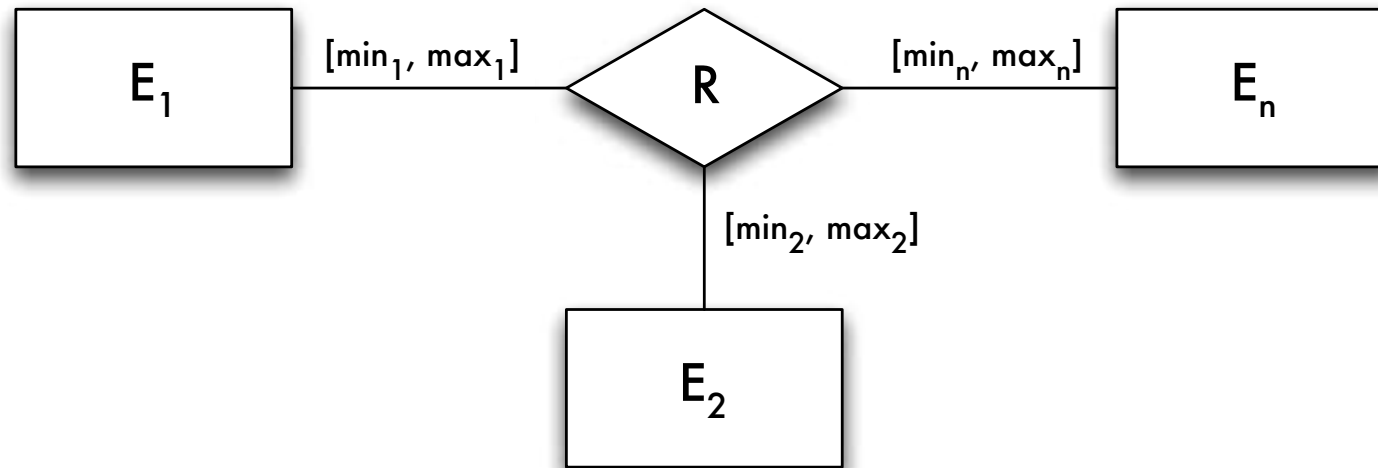
$$R \subseteq E_1 \times E_2 \times \dots \times E_i \times \dots \times E_n$$

Für jedes  $e_i \in E_i$  gibt es

- mindestens  $min_i$  Instanzen des Beziehungstyps der Art  $(\dots, e_i, \dots)$  und
- höchstens  $max_i$  viele Instanzen des Beziehungstyps der Art  $(\dots, e_i, \dots)$

Kardinalitätsbedingung:  $min_i \leq |\{r \mid r \in R \wedge r.E_i = e_i\}| \leq max_i$

# $[min, max]$ -Notation (Kardinalität)



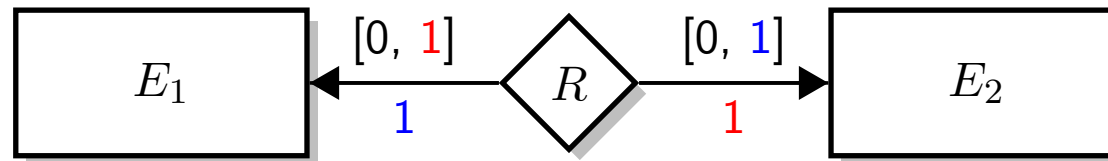
Spezielle Wertangabe für  $min_i$ : 0

Spezielle Wertangabe für  $max_i$ : \*

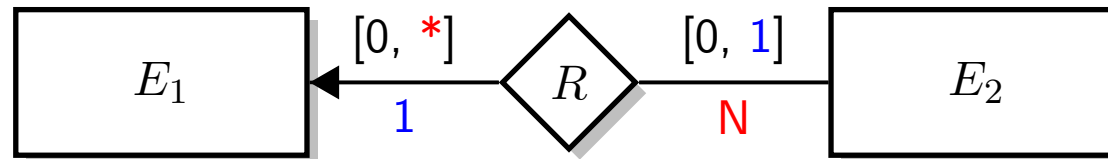
$[0, *]$  legt keine Einschränkungen fest. → Default

# Chen-Notation vs. $[min, max]$ -Notation

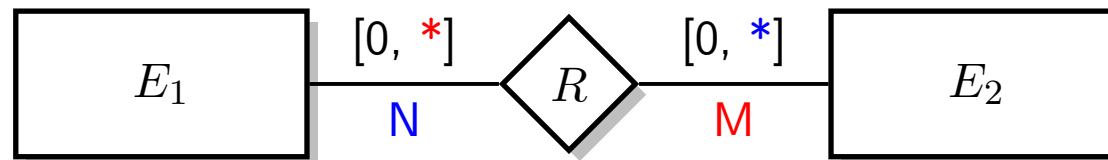
## 1:1 Beziehungstyp



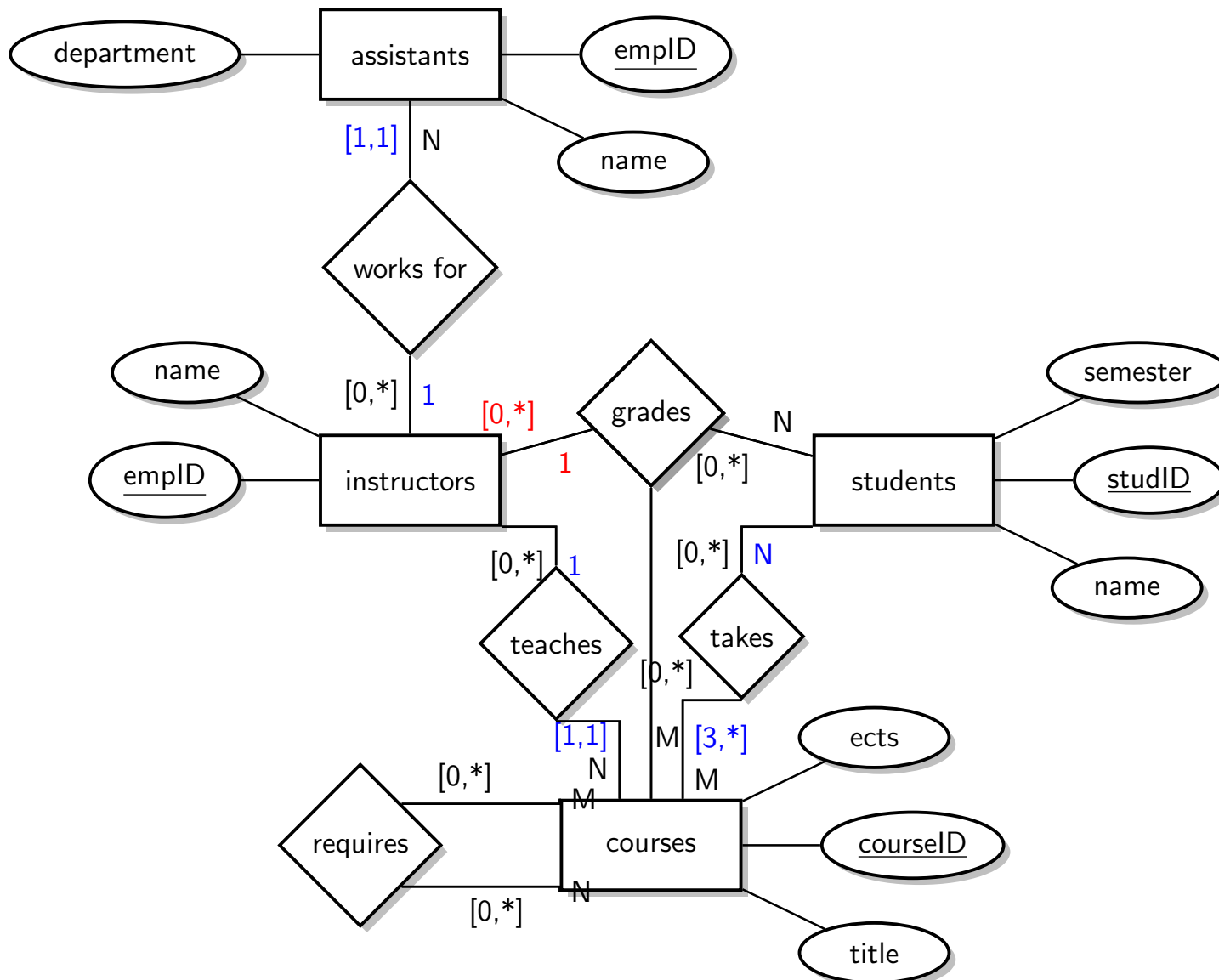
## 1:N Beziehungstyp



## N:M Beziehungstyp



# Chen-Notation vs. [min,max]-Notation



**rot:** Chen ist präziser  
**blau:** [min,max] ist präziser

# Funktionalität vs. Kardinalität

In der Literatur werden die Begriffe Funktionalität und Kardinalität oft synonym verwendet.

In dieser Vorlesung:

## Funktionalität

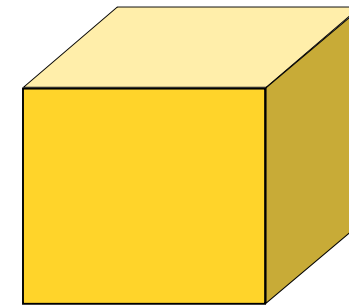
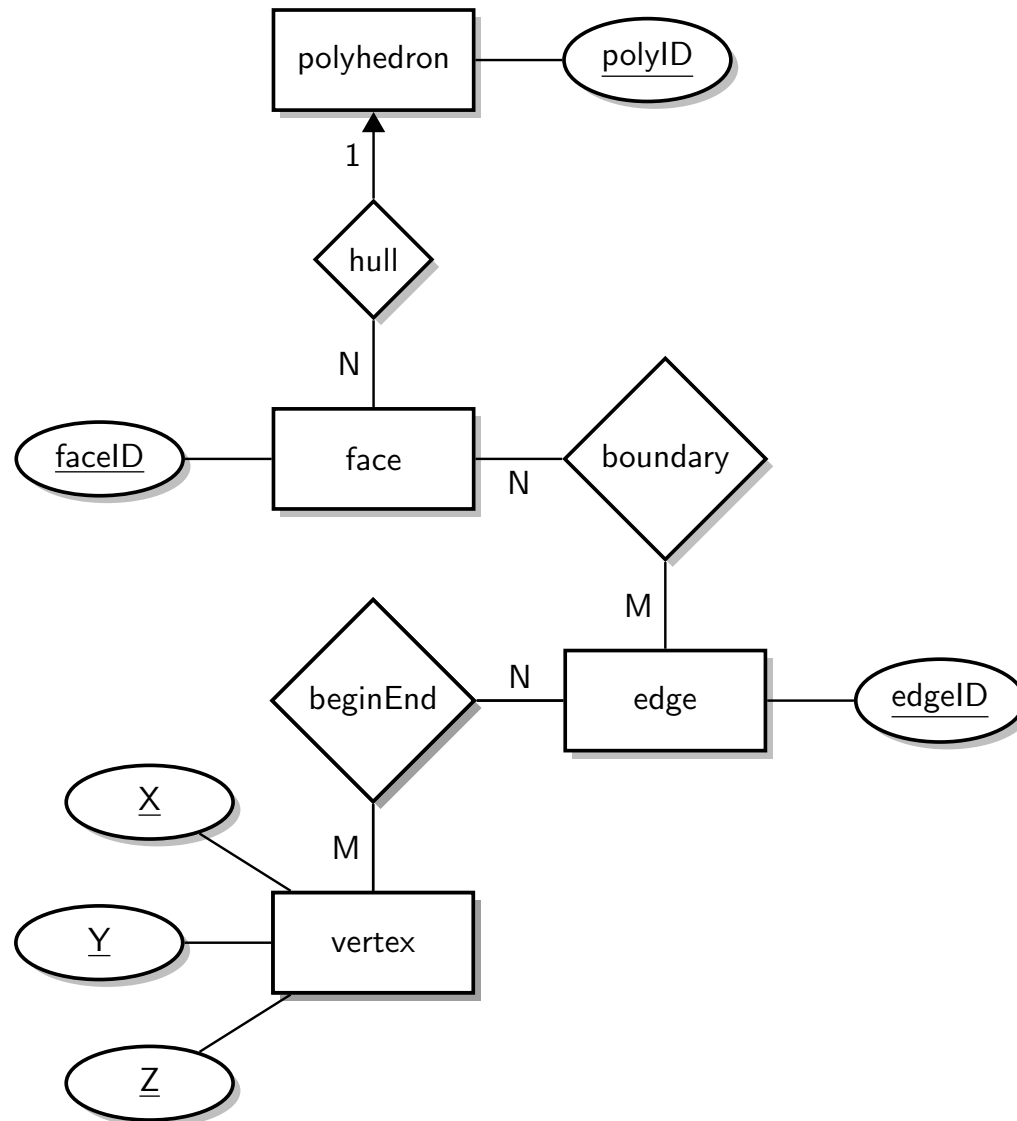
Von Funktionalitäten sprechen wir im Zusammenhang mit der Chen-Notation.

## Kardinalität

Von Kardinalitäten sprechen wir im Zusammenhang mit der  $[min, max]$ -Notation.



# [min,max]-Notation vs. Chen-Notation

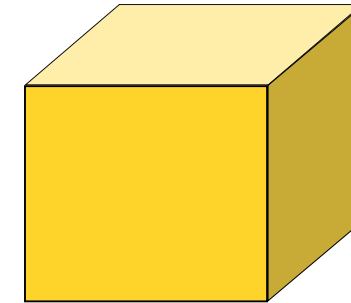
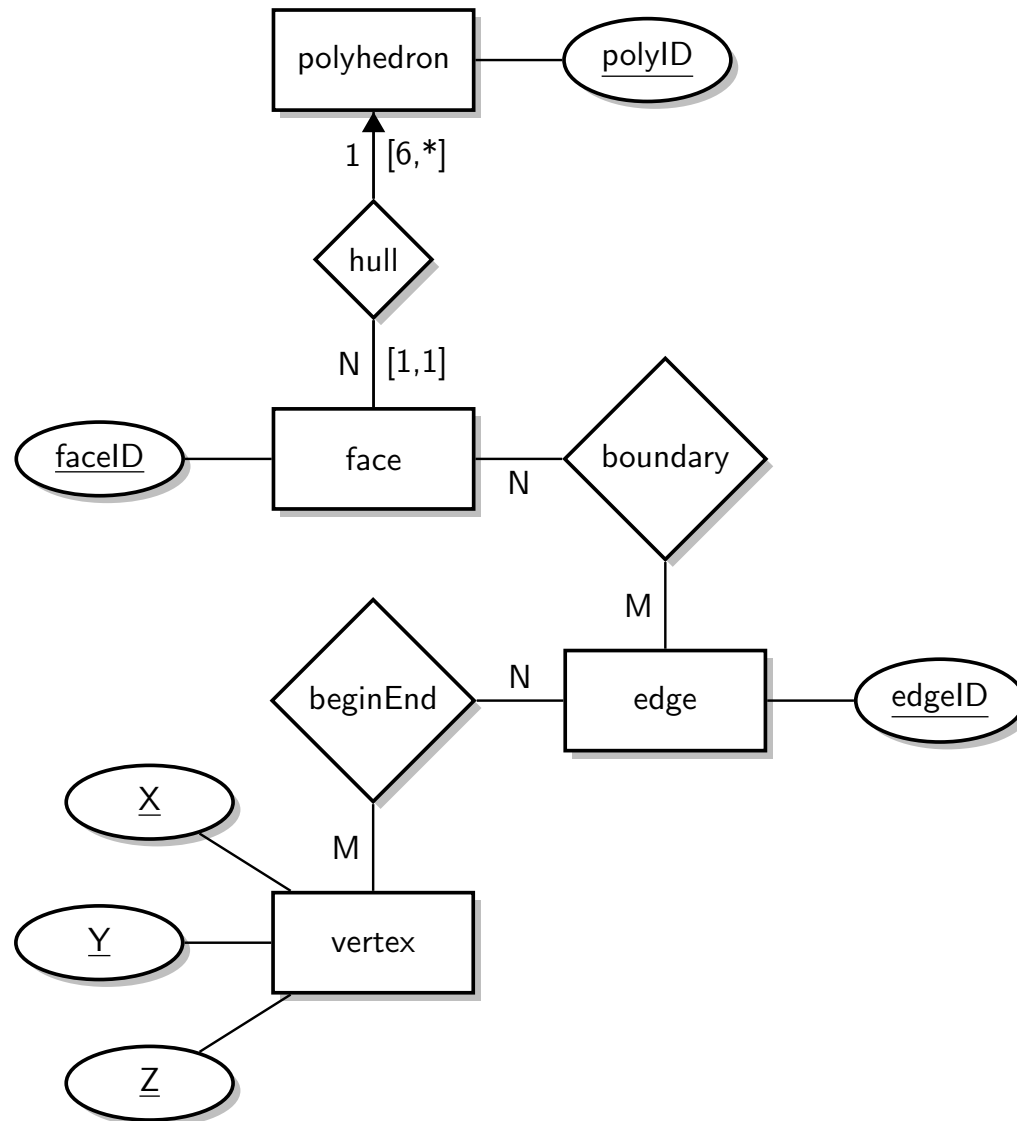


Beispiel Polyeder

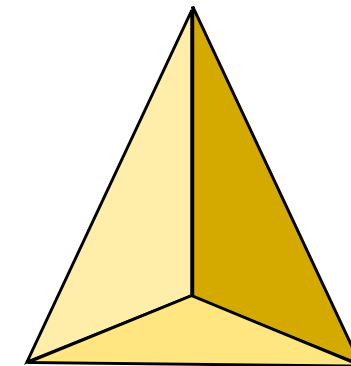
- Funktionalität: Chen-Notation
- Kardinalität: [min,max]-Notation

Funktionalität: Chen-Notation

# [min,max]-Notation vs. Chen-Notation

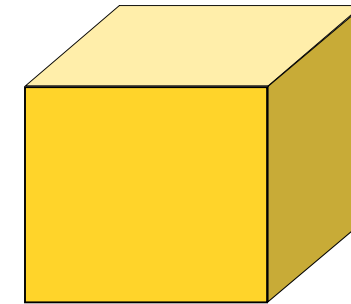
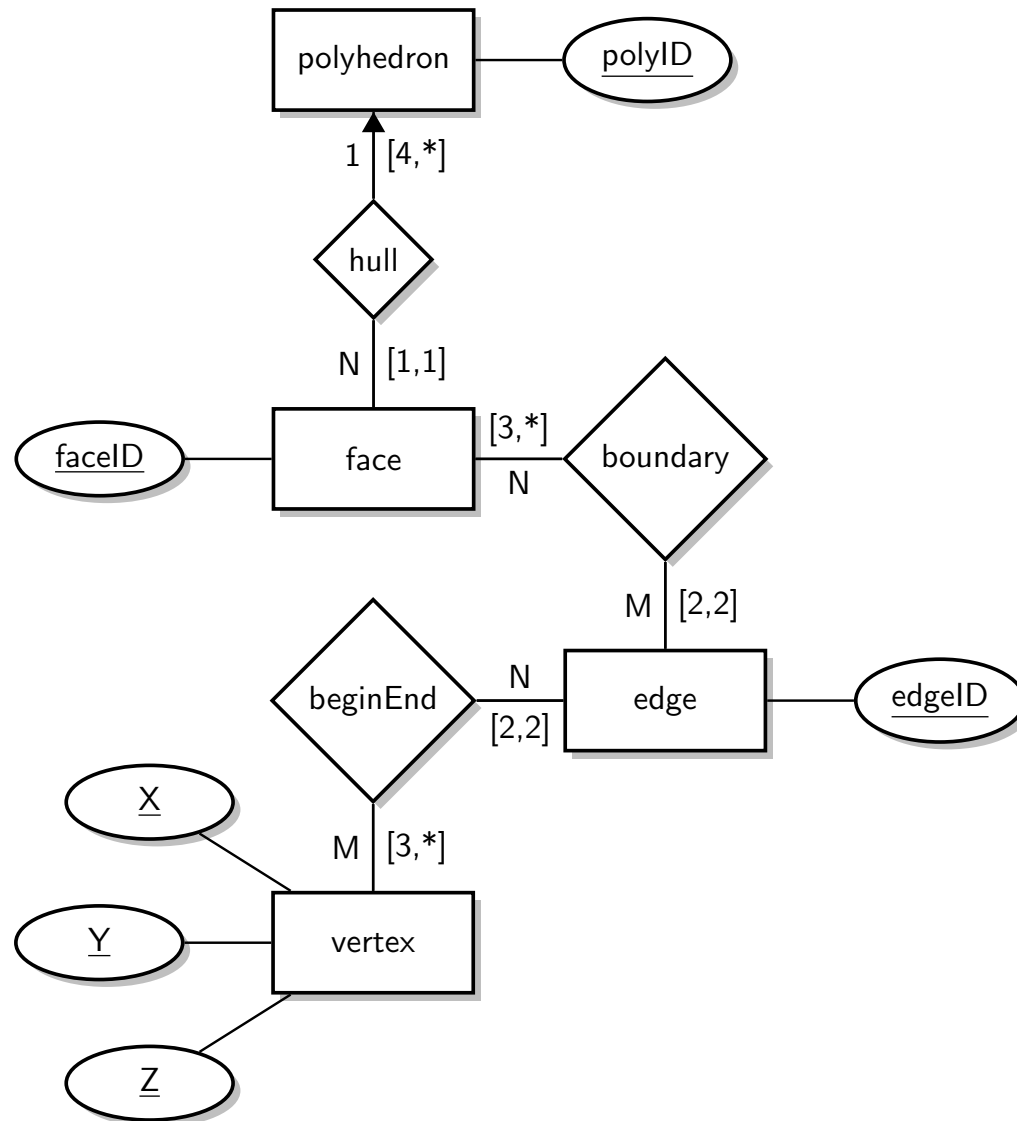


Beispiel Polyeder

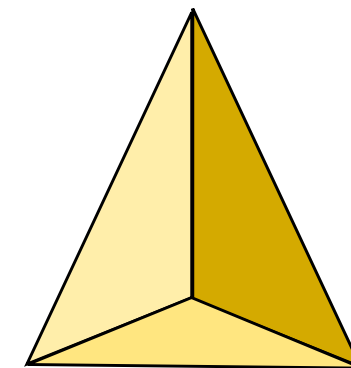


Kardinalität: [min,max]-Notation

## [min,max]-Notation vs. Chen-Notation



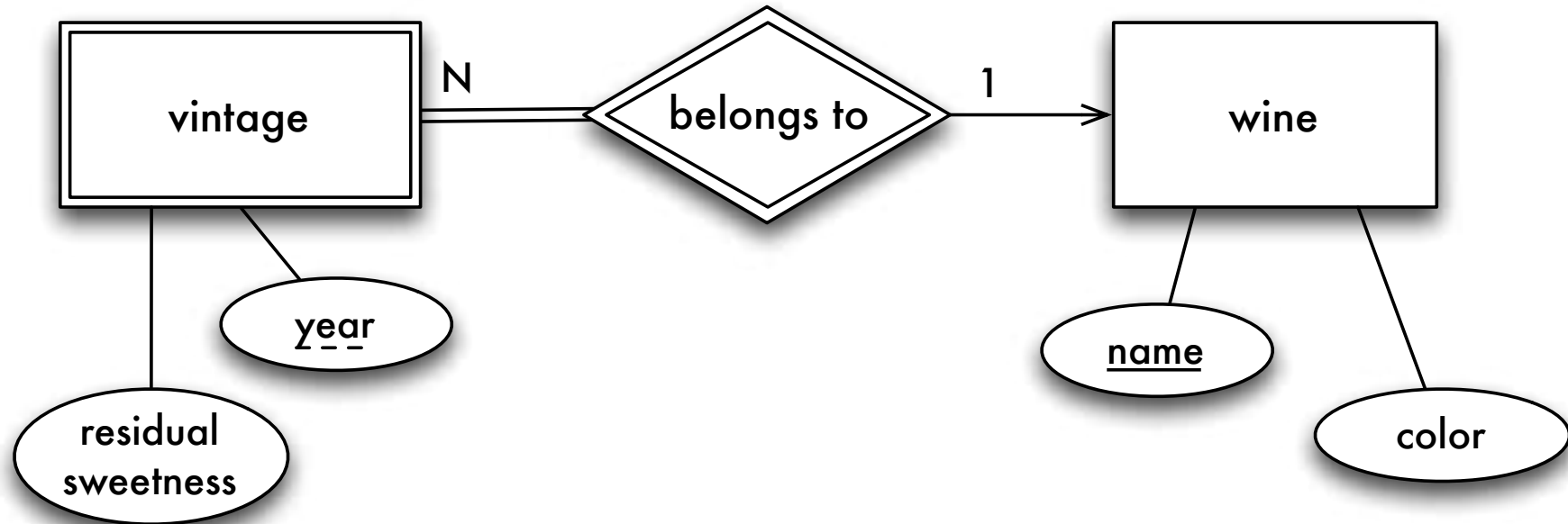
Beispiel Polyeder



Kardinalität: [min,max]-Notation

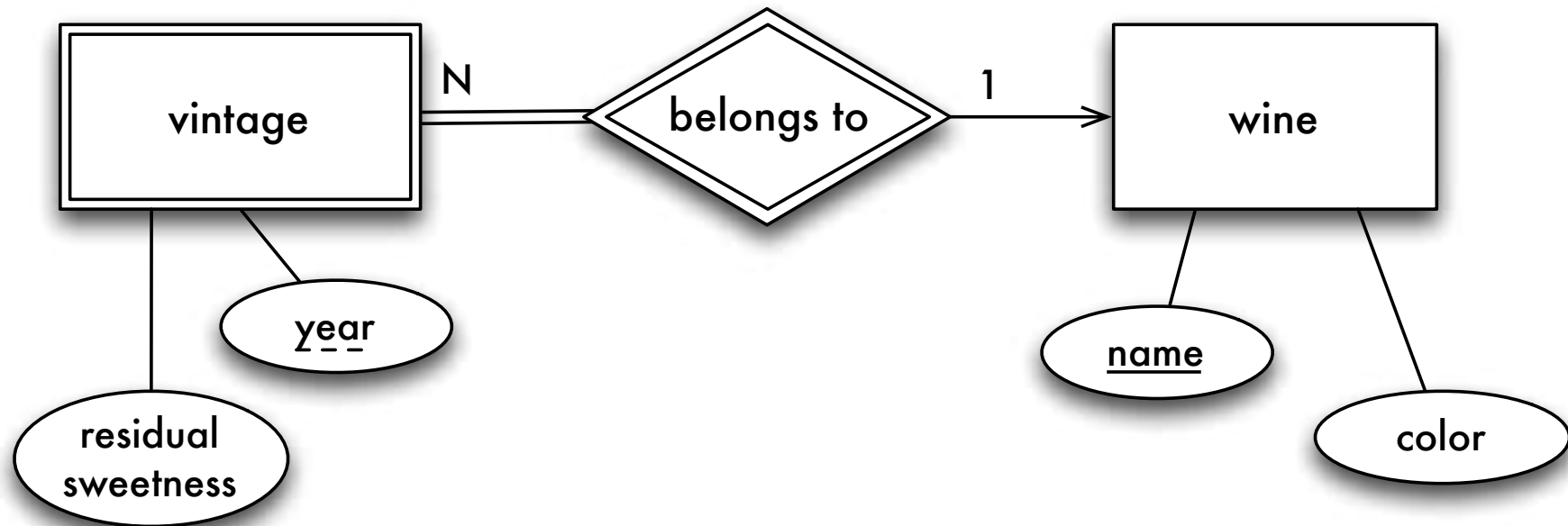
- 1 Datenbankentwurf
- 2 Grundkonzepte
- 3 Eigenschaften von Beziehungstypen
- 4 Zusätzliche Konzepte**
  - Schwache Entitytypen
  - Der ISA-Beziehungstyp
- 5 Alternative Notationen
- 6 Relationen aus Grundkonzepten ableiten
- 7 Relation aus zusätzlichen Konzepten ableiten

# Schwache Entitytypen



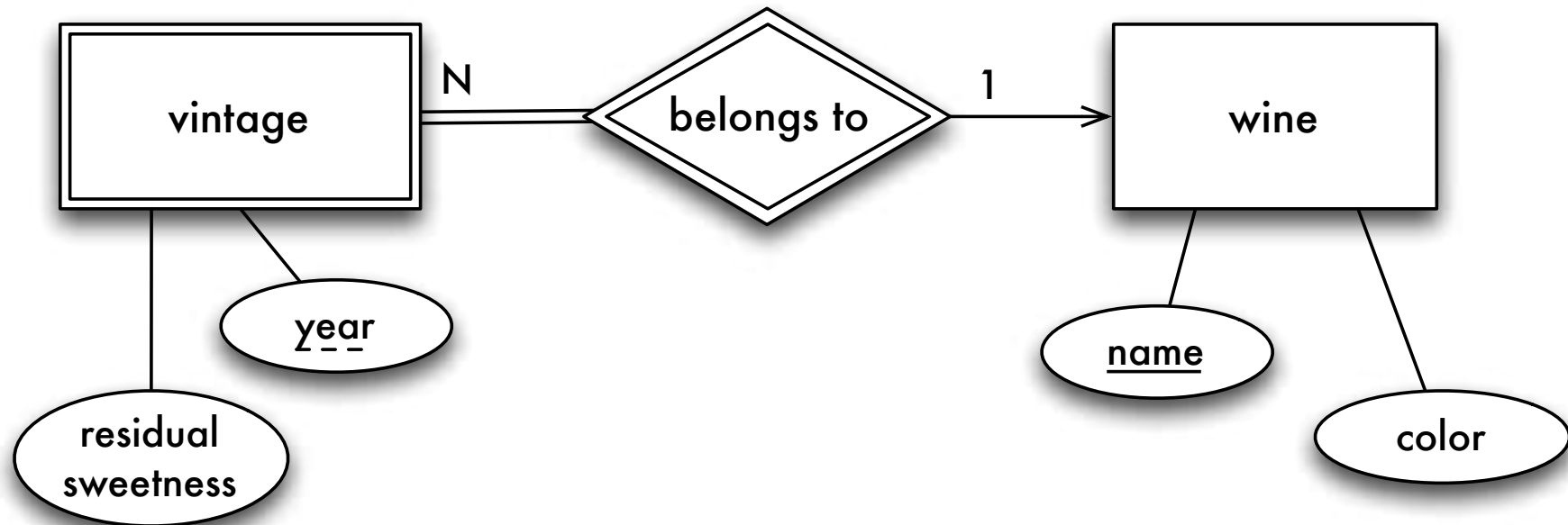
Die Existenz eines **schwachen Entitys** hängt von der Existenz eines **starken Entitys** (identifying/owning entity) ab. Sie sind durch eine identifizierende Beziehung verbunden.

# Schwache Entitytypen



- Totale Partizipation des schwachen Entitytyps.
- Nur in Kombination mit 1:N (N:1) (oder selten auch 1:1) Beziehungstypen  
Der starke Entitytyp ist immer auf der „1“-Seite!

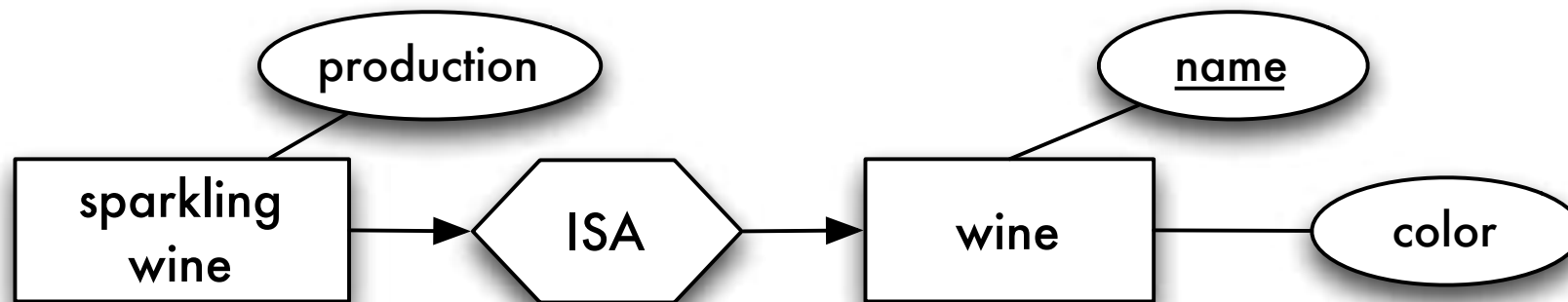
# Schwache Entitytypen



- Schwache Entitys sind oft nur mit dem Schlüssel des entsprechenden starken Entitys eindeutig identifizierbar.
- Die Schlüsselattribute des schwachen Entitytyps werden gestrichelt unterstrichen (**Teilschlüssel**).

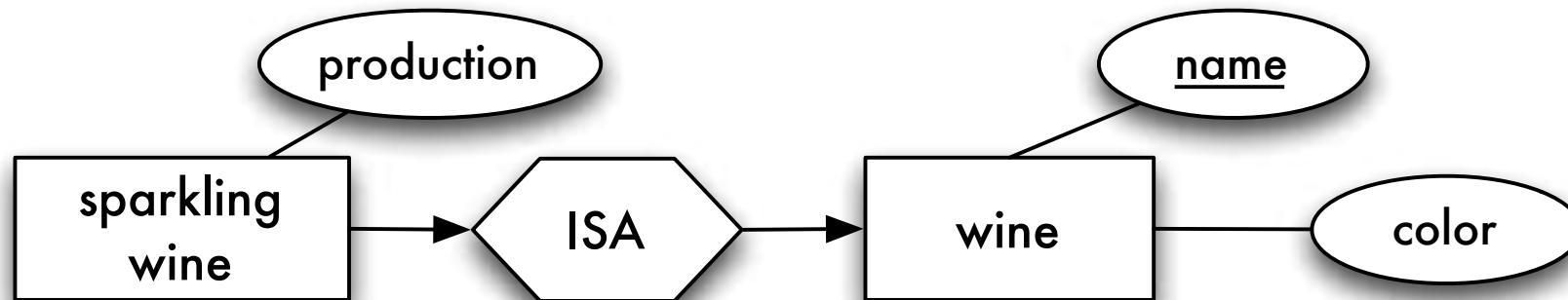
# Der ISA-Beziehungstyp

**Spezialisierung und Generalisierung** wird durch den ISA-Beziehungstyp ausgedrückt (Vererbung).





# Eigenschaften



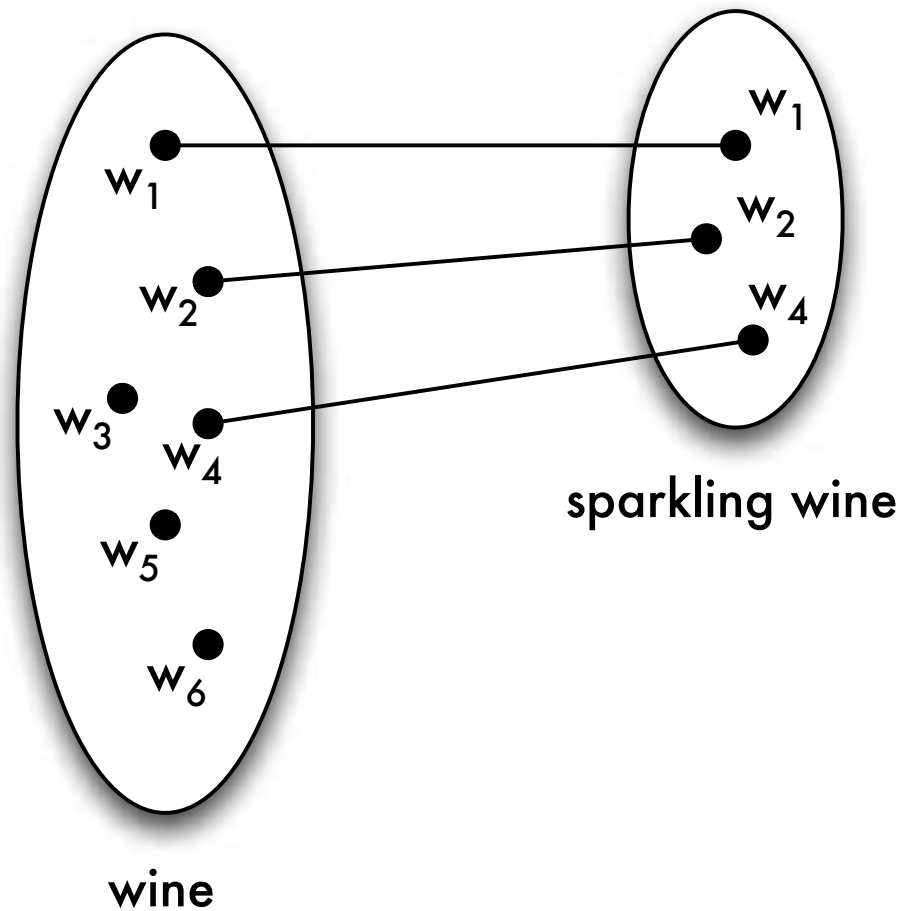
Jeder Schaumwein ist genau einem Wein zugeordnet.

↪ Schaumweine werden durch die funktionale ISA-Beziehung identifiziert.

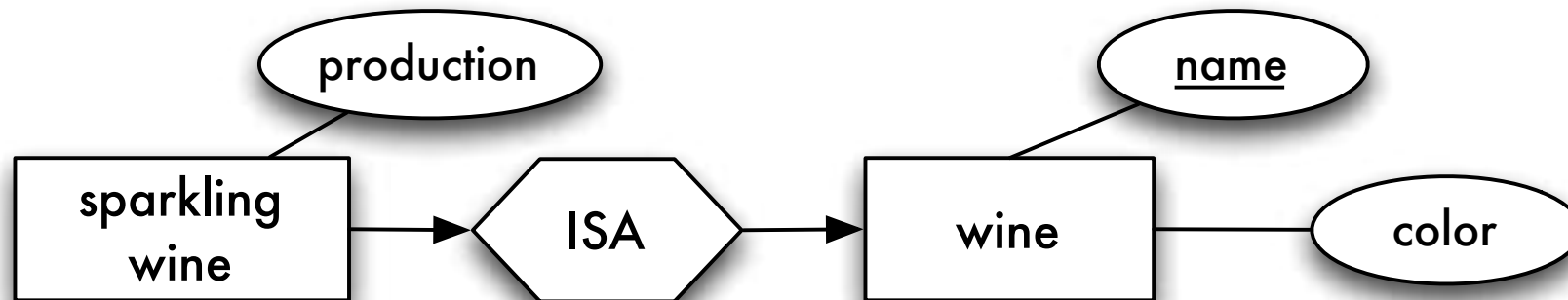
Nicht jeder Wein ist zugleich ein Schaumwein.

Attribute des Entitytyps wine werden an den Entitytyp sparkling wine vererbt.

# Eigenschaften

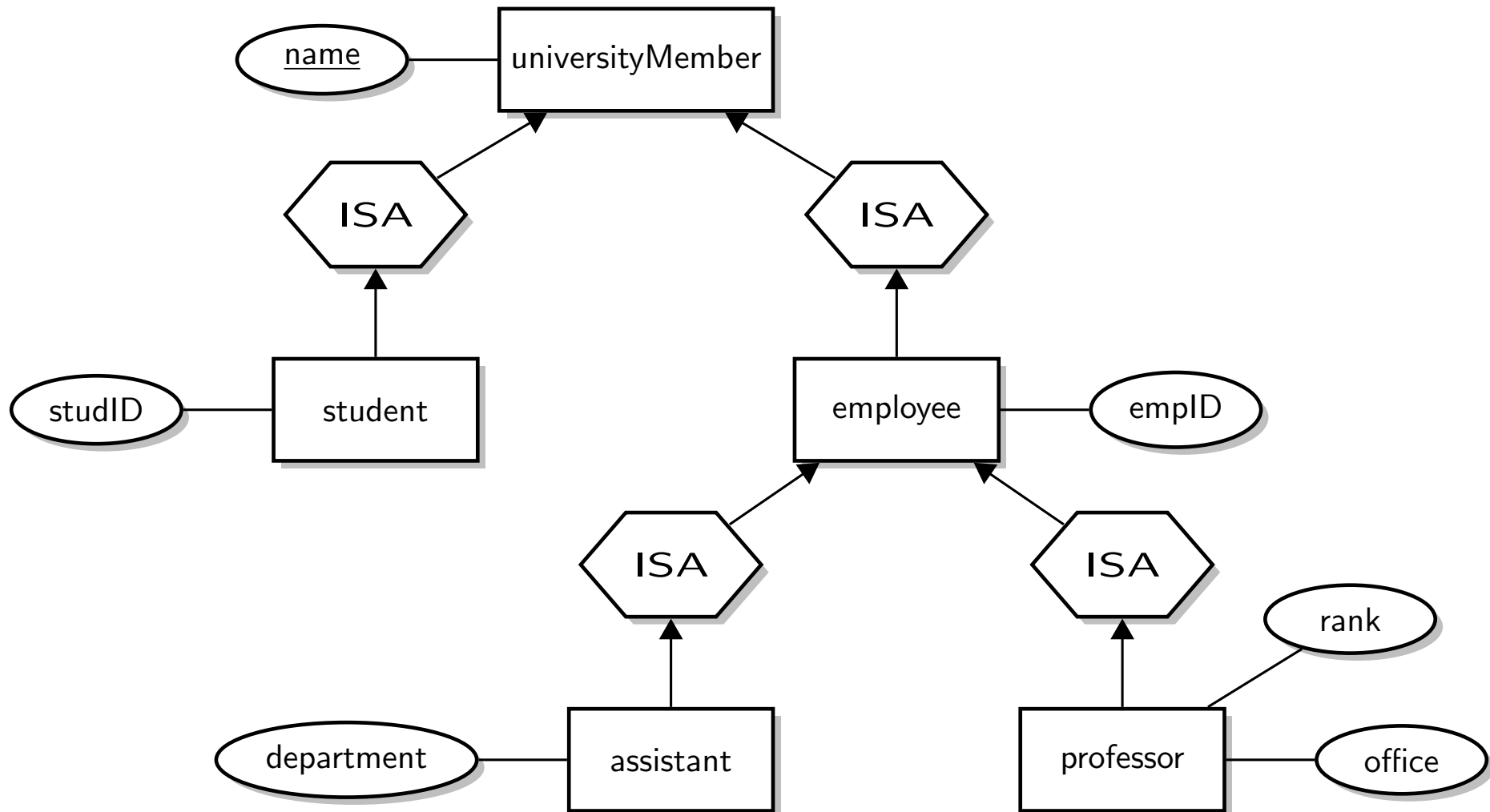


# Kardinalität



- Die Kardinalitäten sind immer
$$\text{ISA}(E_1[1, 1], E_2[0, 1])$$
- Jede Instanz von  $E_1$  (sparkling wine) nimmt genau einmal an der Beziehung teil, während Instanzen von  $E_2$  (wine) maximal einmal teilnehmen.

# Universitätsbeispiel (überlappende Spezialisierung)



# Spezielle Eigenschaften

## Überlappende Spezialisierung

Ein Entity kann zu mehreren spezialisierten Entitymengen gehören.

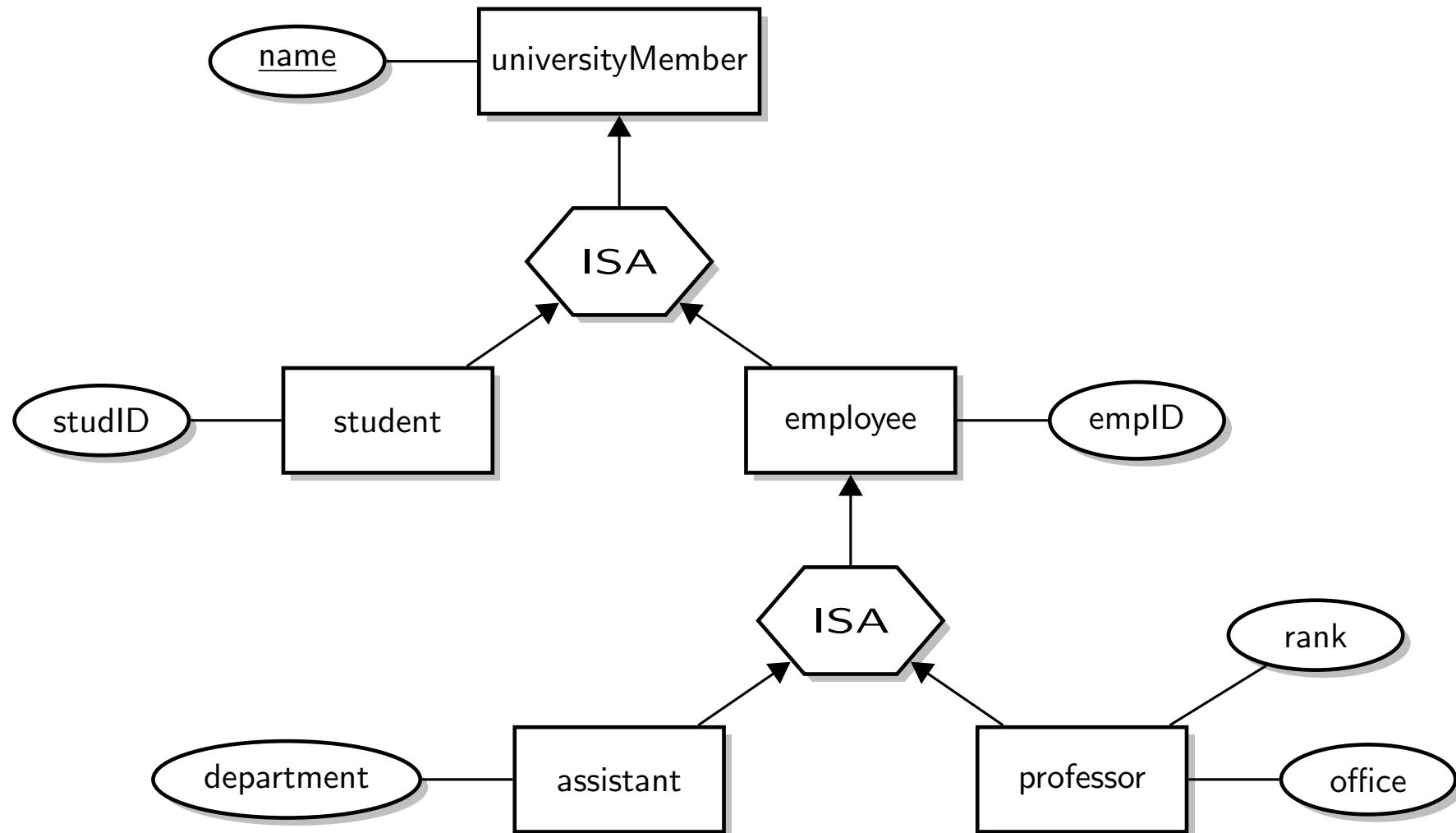
→ Separate ISA-Symbole verwenden

## Disjunkte Spezialisierung

Ein Entity kann zu höchstens einer spezialisierten Entitymengen gehören.

→ Ein gemeinsames ISA-Symbol verwenden

# Beispiel



# Attribute und Beziehungstypen

Spezialisiertere Entitytypen erben

- Attribute von weniger spezialisierten Entitytypen
- Teilnahme in Beziehungstypen von weniger spezialisierten Entitytypen

Spezialisiertere Entitytypen können

- zusätzliche Attribute haben
- an Beziehungstypen teilnehmen, an denen die weniger spezialisierten Entitytypen nicht teilnehmen.

# Participation Constraints

## Totale Generalisation/Spezialisierung

Jeder weniger spezialisierte Entitytyp **muss** zu einem spezialisierten Entitytyp gehören.

Notation: doppelte Linie

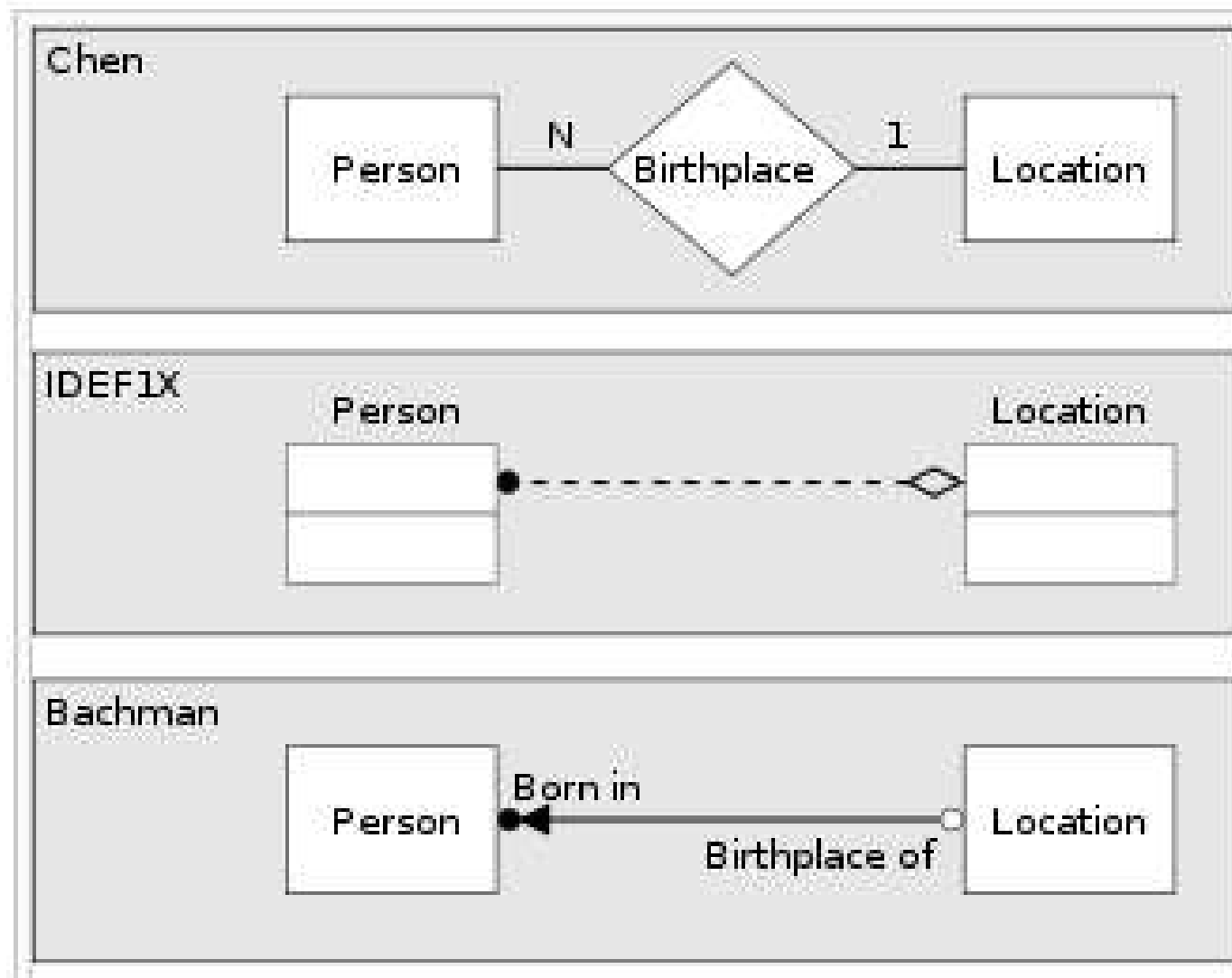
## Partielle Generalisation/Spezialisierung (default)

Jeder weniger spezialisierte Entitytyp **kann** (aber muss nicht) zu einem spezialisierten Entitytyp gehören.



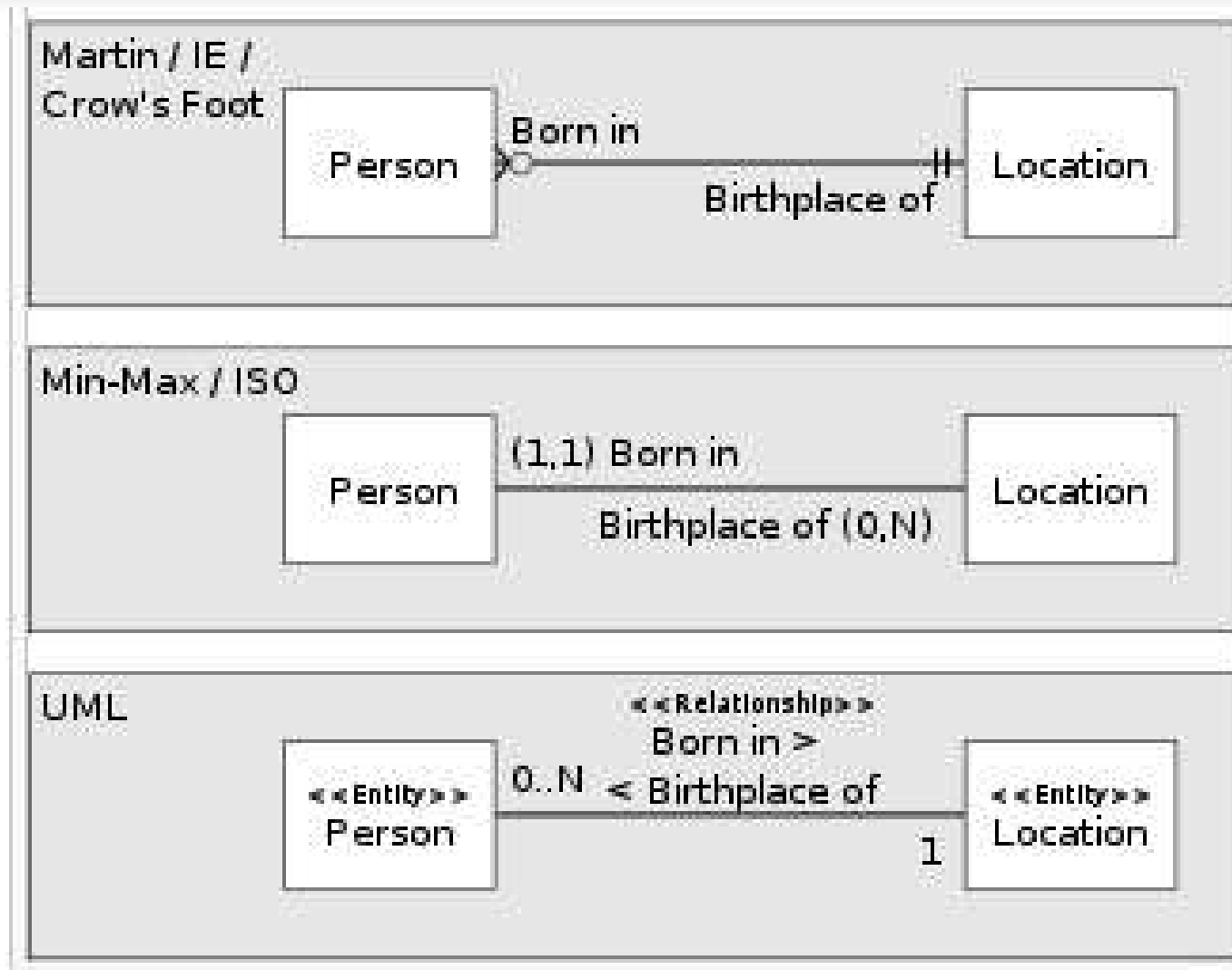
- 1 Datenbankentwurf
- 2 Grundkonzepte
- 3 Eigenschaften von Beziehungstypen
- 4 Zusätzliche Konzepte
- 5 Alternative Notationen**
- 6 Relationen aus Grundkonzepten ableiten
- 7 Relation aus zusätzlichen Konzepten ableiten

# Alternative Notationen



[http://en.wikipedia.org/wiki/Entity-relationship\\_model](http://en.wikipedia.org/wiki/Entity-relationship_model)

# Alternative Notationen



[http://en.wikipedia.org/wiki/Entity-relationship\\_model](http://en.wikipedia.org/wiki/Entity-relationship_model)

# Zusammenfassung bisher

- Entity-Relationship-Diagramme (ERDs) beschreiben das konzeptuelle Schema einer Datenbank.
- Weitere Konstrukte im erweiterten ER-Modell.
- Grundlegende ER-Konzepte (Entitytypen, Beziehungstypen, Attribute)
- Stelligkeit von Beziehungstypen
- Funktionalität / Kardinalität (Chen, [min,max], totale/partielle Partizipation)
- Schwache Entitytypen
- ISA-Beziehungstyp

- 1 Datenbankentwurf
  - Schritte des Datenbankentwurfs
  - Datenbankentwurf am Beispiel
- 2 Grundkonzepte
  - Beispielszenarien
  - Entitytypen
  - Attribute
  - Beziehungstypen
- 3 Eigenschaften von Beziehungstypen
  - Stelligkeit
  - Chen-Notation (Funktionalität)
  - Participation Constraints
  - Chen-Notation (Funktionalität) bei n-stelligen Beziehungstypen
  - $[min, max]$ -Notation (Kardinalität)
- 4 Zusätzliche Konzepte
  - Schwache Entitytypen

- Der ISA-Beziehungstyp

## 5 Alternative Notationen

## 6 Relationen aus Grundkonzepten ableiten

- Entitytypen
- Beziehungstypen

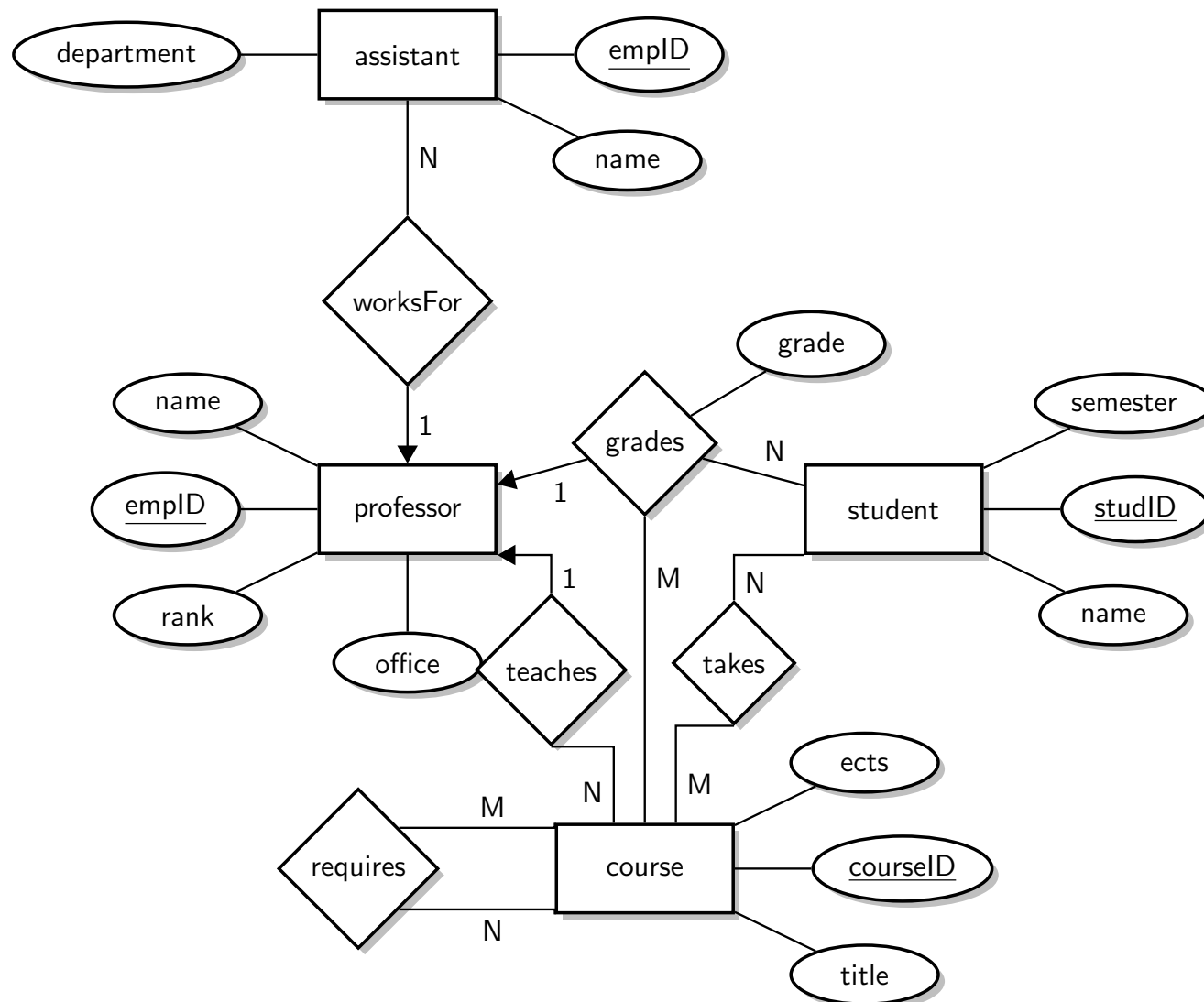
## 7 Relation aus zusätzlichen Konzepten ableiten

- Schwache Entitytypen
- Rekursive Beziehungstypen
- N-äre Beziehungstypen
- Spezielle Attribute
- Generalsierung

# Entwurfsanmerkungen

- Entitys entsprechen Substantiven, Beziehungen entsprechen Verben.
- Jede Aussage in den Anforderungen sollten irgendwo im ER-Schema widergespiegelt werden.
- Jedes ER-Diagramm (ERD) sollte sich irgendwo in den Anforderungen wiederfinden.
- Ein konzeptueller Entwurf enthüllt oft Inkonsistenzen und Mehrdeutigkeiten in den Anforderungen, welche zuerst geklärt werden müssen.

# Universitätsschema mit Funktionalitäten



Wie erstellt man systematisch Relationen, die alle Informationen des ER-Diagramms beinhalten?



# Entitytypen

- **student**

{[ studID: integer, name: string, semester: integer ]}

- **course**

{[ courseID: integer, title: string, ects: integer ]}

- **professor**

{[ emplID: integer, name: string, rank: string, office: integer ]}

- **assistant**

{[ emplID: integer, name: string, department: string ]}

## Grundsätzliches Vorgehen

- Für jeden Entitytyp → Relation
- Name des Entitytypen → Name der Relation
- Attribute des Entitytypen → Attribute der Relation
- Schlüssel des Entitytypen → Schlüssel der Relation

# Entitytypen

- **student**

{[ studID: integer, name: string, semester: integer ]}

- **course**

{[ courseID: integer, title: string, ects: integer ]}

- **professor**

{[ emplID: integer, name: string, rank: string, office: integer ]}

- **assistant**

{[ emplID: integer, name: string, department: string ]}

## Notation von Relationenschemata

student (studID: integer, name: string, semester: integer)

student: {[ studID: integer, name: string, semester: integer ]}

Die Reihenfolge der Attribute ist **in diesem Kontext** egal!

# Entitytypen

- **student**

{[ studID: integer, name: string, semester: integer ]}

- **course**

{[ courseID: integer, title: string, ects: integer ]}

- **professor**

{[ emplID: integer, name: string, rank: string, office: integer ]}

- **assistant**

{[ emplID: integer, name: string, department: string ]}

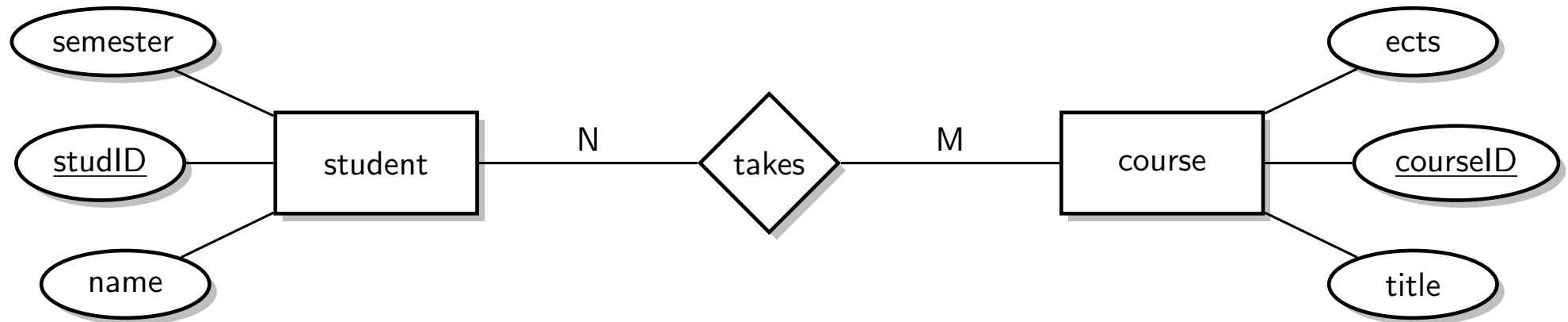
## Notation von Relationenschemata

student (studID, name, semester)

student: {[ studID, name, semester ]}

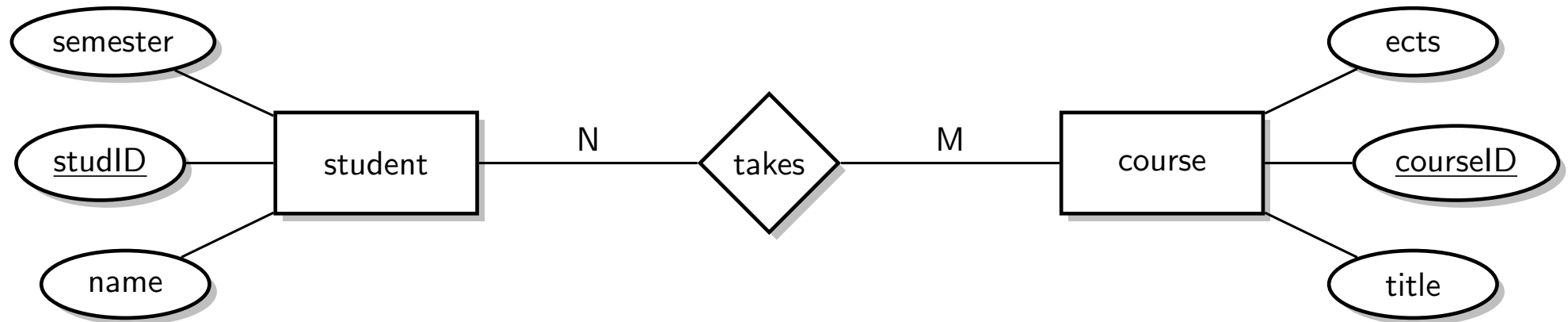
Und im Moment ist auch die Domäne der Attribute nicht wichtig.

# Abbildung von N:M-Beziehungstypen



Wie bilden wir diese Information auf Relationen ab?

# Abbildung von N:M-Beziehungstypen

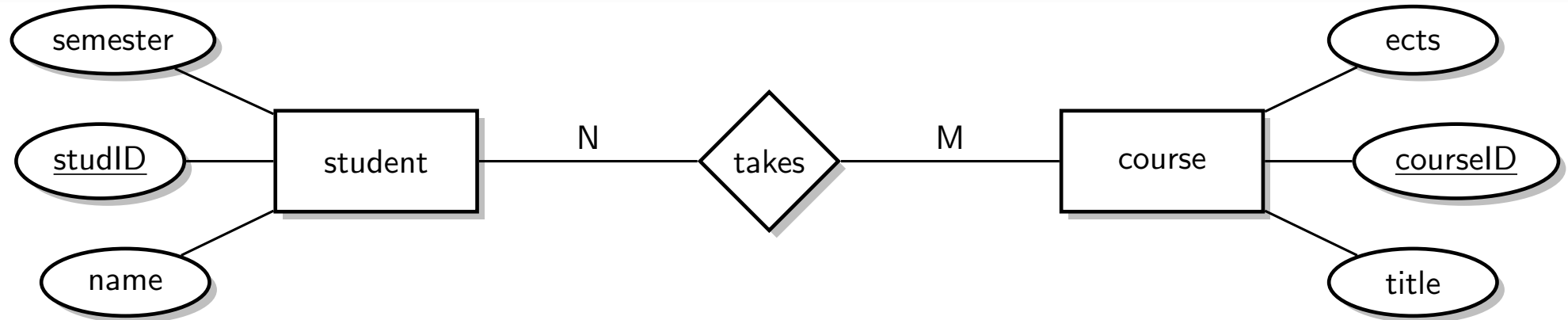


## Grundsätzliches Vorgehen

- Neues Relationenschema mit allen Attributen des Beziehungstyps
- Übernahme aller Primärschlüssel der beteiligten Entitytypen
- Primärschlüssel der beteiligten Entitytypen zusammen werden Schlüssel im neuen Relationenschema

**takes:** {[ studID → student, courseID → course ]}

# Abbildung von N:M-Beziehungstypen



## Grundsätzliches Vorgehen

- Neues Relationenschema mit allen Attributen des Beziehungstyps
- Übernahme aller Primärschlüssel der beteiligten Entitytypen
- Primärschlüssel der beteiligten Entitytypen zusammen werden Schlüssel im neuen Relationenschema

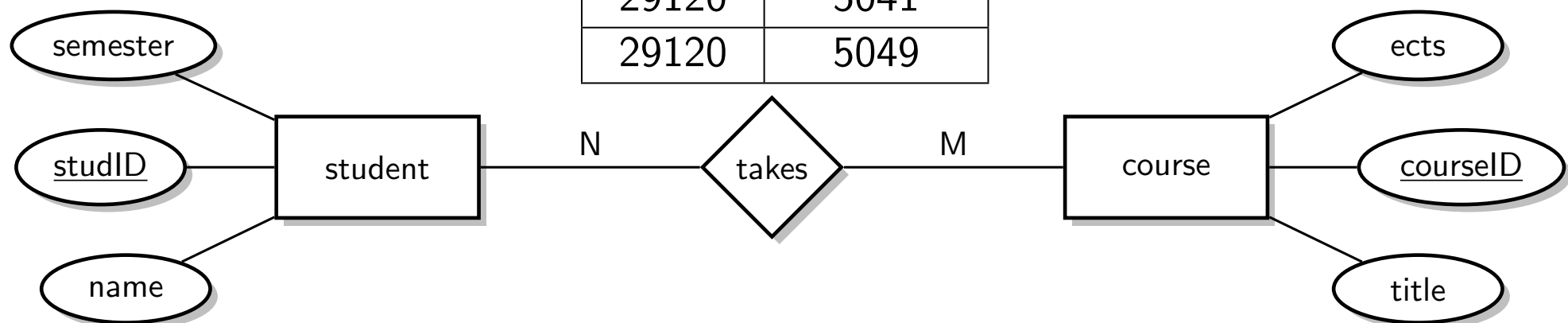
„Importierte“ Schlüsselattribute der beteiligten Entitytypen werden **Fremdschlüssel** genannt.

# Abbildung von N:M-Beziehungstypen

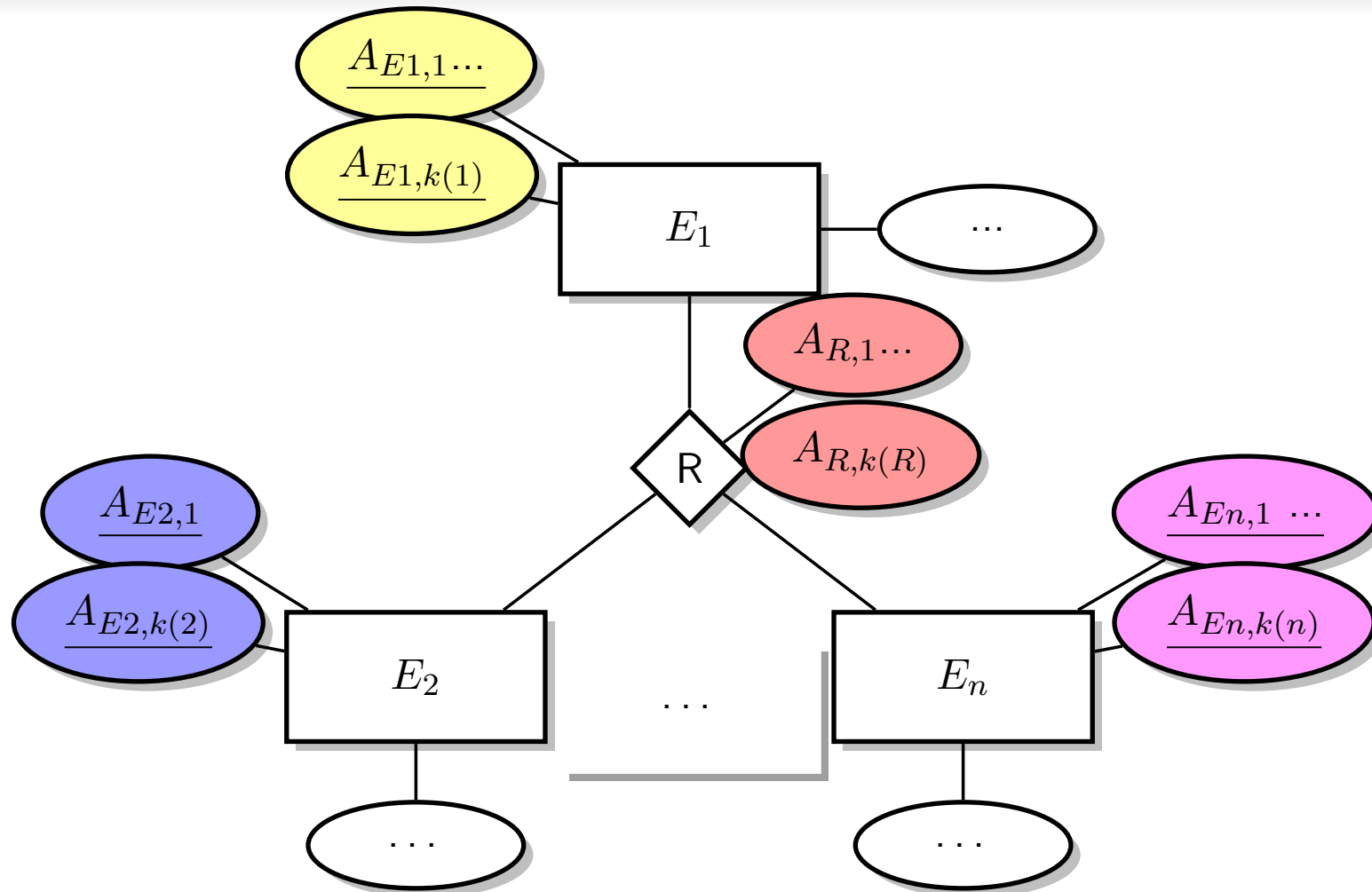
student	
<u>studID</u>	...
26120	...
27550	...
...	...

takes	
<u>studID</u>	<u>courseID</u>
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049

course	
<u>courseID</u>	...
5001	...
4052	...
...	...



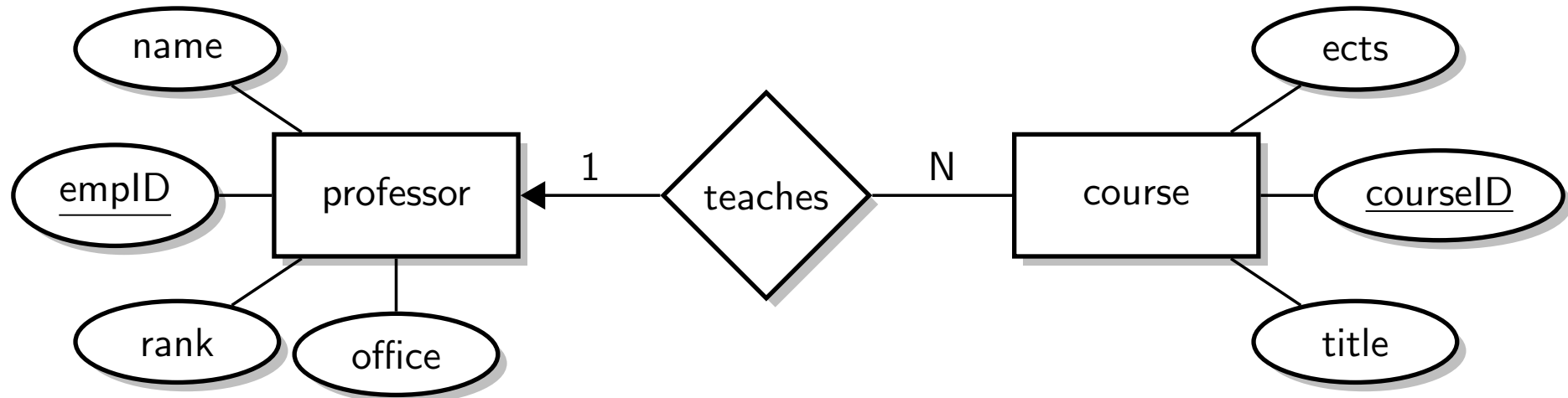
## Abbildung von N:M-Beziehungstypen allgemein



$R: \{ [ \underbrace{A_{E1,1}, \dots, A_{E1,k(1)}}_{\text{Schlüssel von } E_1}, \underbrace{A_{E2,1}, \dots, A_{E2,k(2)}}_{\text{Schlüssel von } E_2}, \dots, \underbrace{A_{En,1}, \dots, A_{En,k(n)}}_{\text{Schlüssel von } E_n}, \underbrace{A_{R,1}, \dots, A_{R,k(R)}}_{\text{Attribute von } R} ] \}$

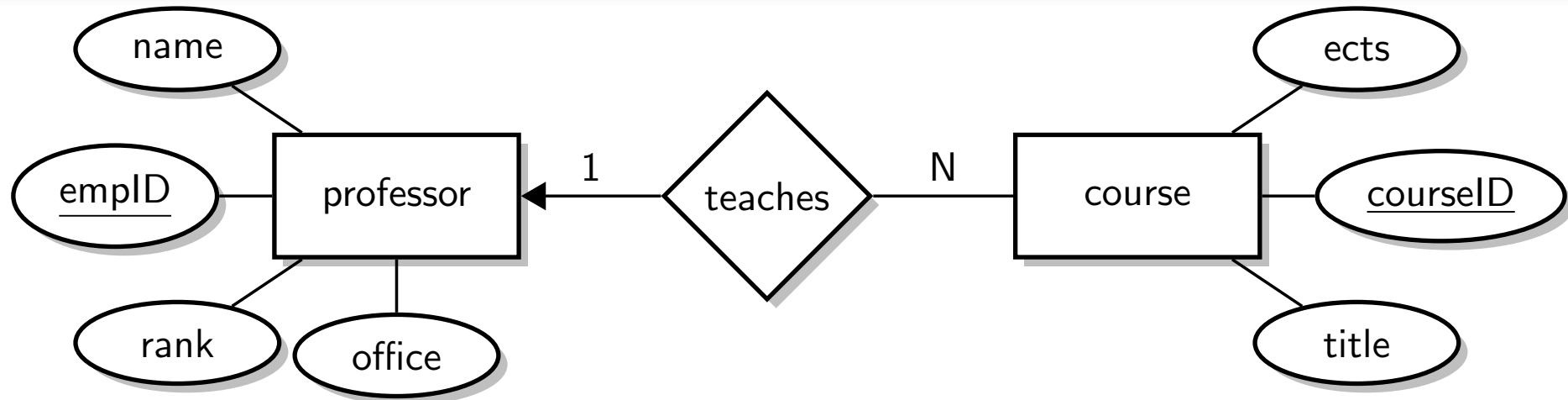


# Abbildung von 1:N-Beziehungstypen



Soll dieser Beziehungstyp genauso wie ein N:M-Beziehungstyp abgebildet werden?

# 1:N-Beziehungstypen

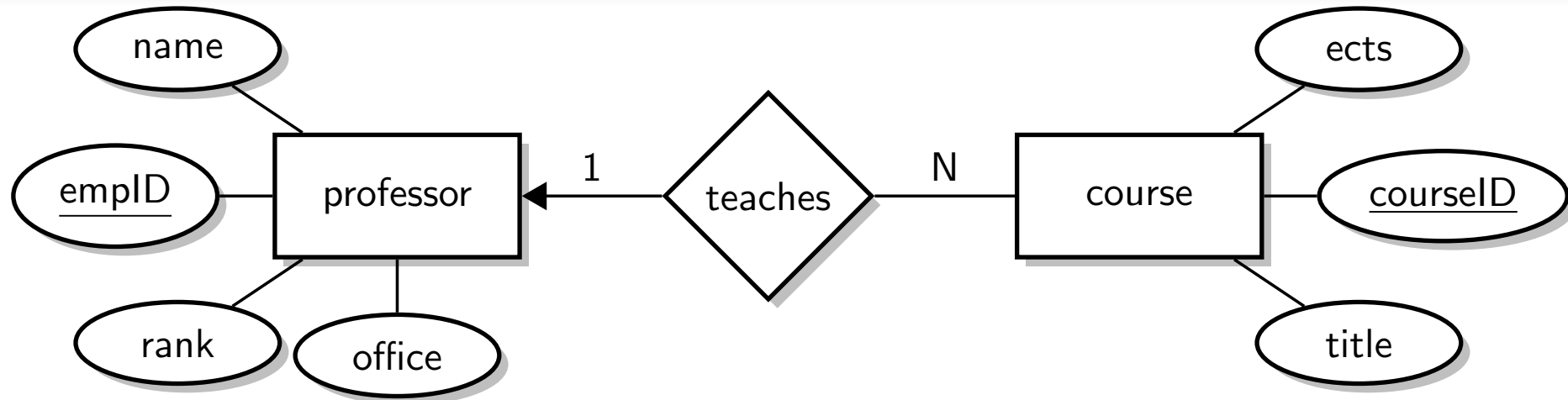


- Neues Relationenschema mit allen Attributen des Beziehungstyps
- Übernahme aller Primärschlüssel der beteiligten Entitytypen
- Primärschlüssel ...

## Initialentwurf!

- **course:** {[ courseID, title, ects ]}
- **professor:** {[ emplID, name, rank, office ]}
- **teaches:** {[ courseID → course, emplID → professor ]}

# 1:N-Beziehungstypen



- Neues Relationenschema mit allen Attributen des Beziehungstyps
- Übernahme aller Primärschlüssel der beteiligten Entitytypen
- Primärschlüssel der N-Seite wird Schlüssel im neuen Relationenschema

## Initialentwurf!

- **course:** {[ courseID, title, ects ]}
- **professor:** {[ emplID, name, rank, office ]}
- **teaches:** {[ courseID → course, emplID → professor ]}

## Abbildung von 1:N-Beziehungstypen

### Initialentwurf

- **course:** {[ courseID, title, ects ]}
- **professor:** {[ emplID, name, rank, office ]}
- **teaches:** {[ courseID → course, emplID → professor ]}

### Verbesserung durch Zusammenfassung (finale Abbildung)

- **course:** {[ courseID, title, ects, taughtBy → professor ]}
- **professor:** {[ emplID, name, rank, office ]}

**taughtBy** ist ein **Fremdschlüssel** und referenziert den Primärschlüssel der Relation professor.

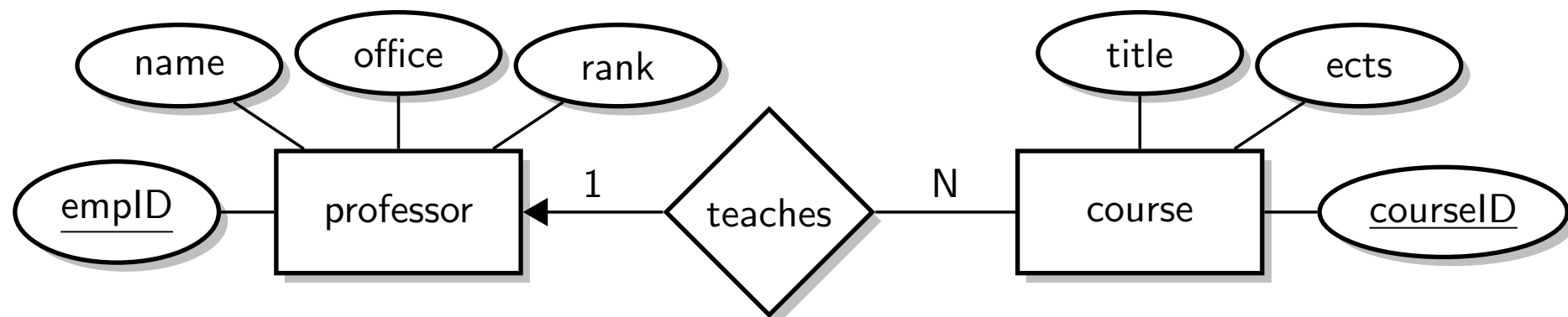
Die Werte von **taughtBy** entsprechen den Werten von *emplID* in der Relation professor.

Relationen mit denselben Schlüsseln können und **sollten immer** kombiniert werden... **aber nur diese und keine anderen!**

# professor und course

professor			
empID	name	rank	office
2125	Socrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

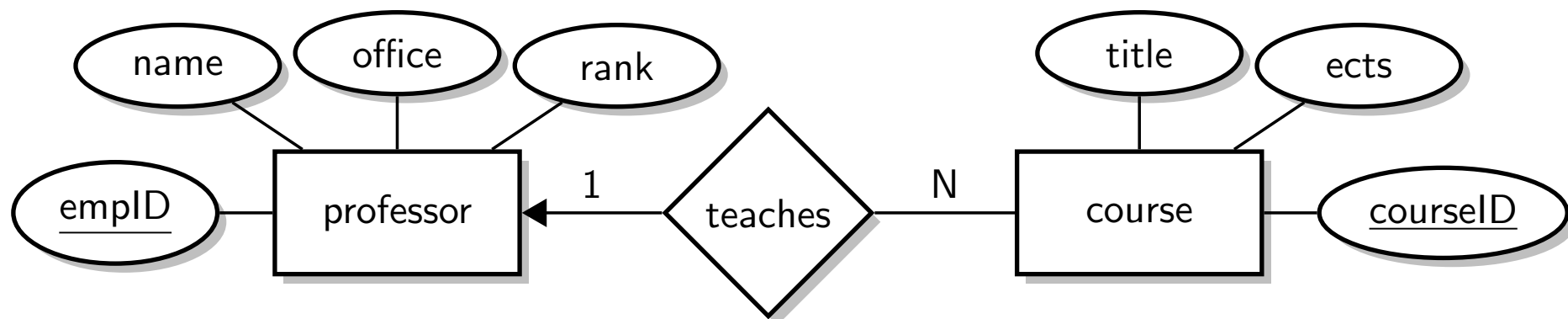
course			
courseID	title	ects	taughtBy
5001	DBS	4	2137
5041	Robotics	4	2125
5043	Software Engineering	3	2126
5049	Ethics	2	2125
4052	Logic	4	2125
5052	Theory of Science	3	2126
5216	Bioethics	2	2126
5259	Chemistry	2	2133
5022	Belief and Knowledge	2	2134
4630	Physics	4	2137



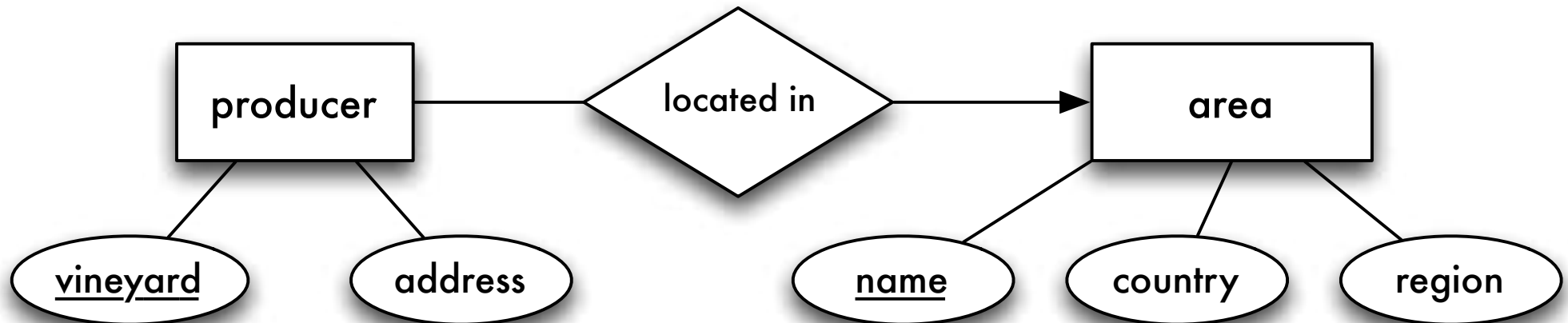
Achtung: das funktioniert **nicht**

professor				
emplID	name	rank	office	teaches
2125	Socrates	C4	226	5041
<b>2125</b>	<b>Socrates</b>	<b>C4</b>	<b>226</b>	<b>5049</b>
<b>2125</b>	<b>Socrates</b>	<b>C4</b>	<b>226</b>	<b>4052</b>
...	...	...	...	...
2134	Augustinus	C3	309	<b>5022</b>
2136	Curie	C4	36	<b>??</b>
...	...	...	...	...

course		
courseID	title	ects
5001	DBS	4
5041	Robotics	4
5043	Software Engineering	3
5049	Ethics	2
4052	Logic	4
5052	Theory of Science	3
5216	Bioethics	2
5259	Chemistry	2
5022	Belief and Knowledge	2
4630	Physics	4



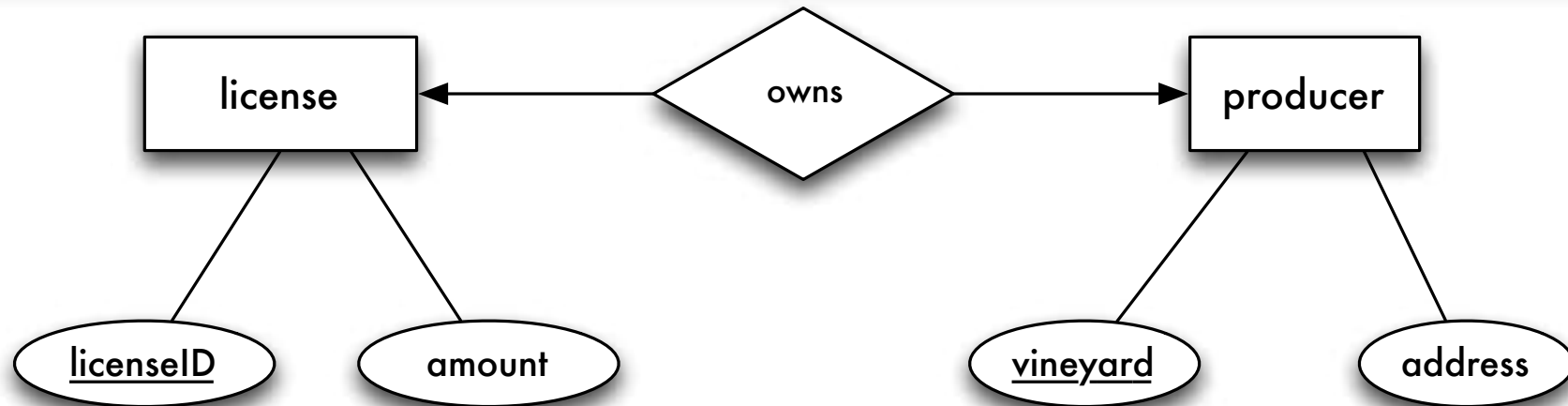
## Zusammenfassung: N:1-Beziehungstypen



Wie bildet man dieses ER-Diagramm auf Relationen ab?

- **producer:** {[ vineyard, address, locatedIn → area ]}
- **area:** {[ name, country, region ]}

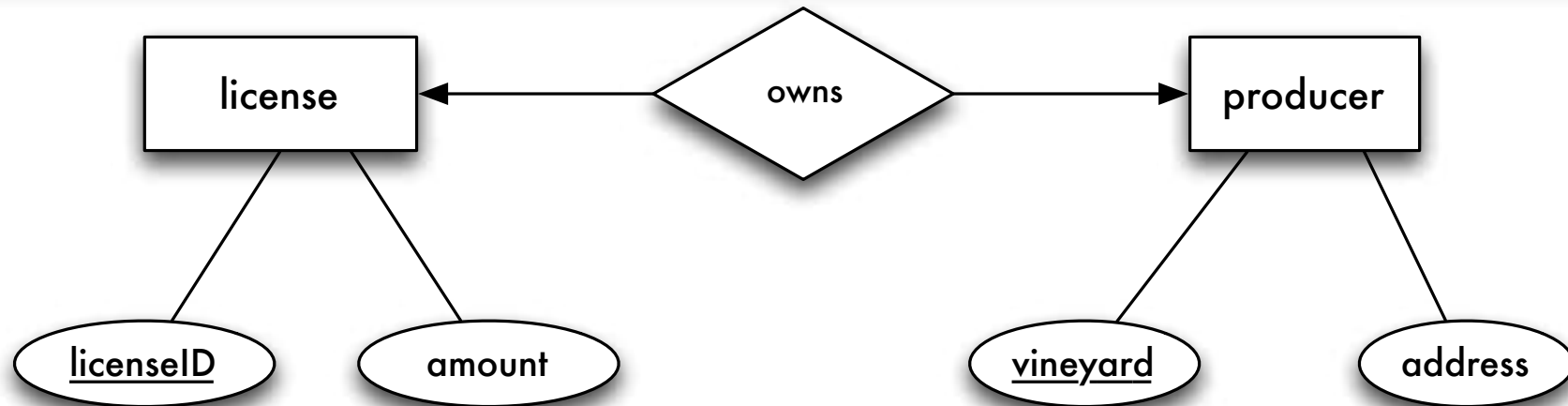
# 1:1-Beziehungstypen



Unterscheidet sich die Herangehensweise hier zu einem 1:N-Beziehungstypen?



# 1:1-Beziehungstypen



- Neues Relationschema mit allen Attributen des Beziehungstyps
- Übernahme aller Primärschlüssel der beteiligten Entitytypen
- Irgendein Primärschlüssel der involvierten Entitytypen wird der Primärschlüssel im neuen Relationenschema

## Initialentwurf!

- **license:** {[ licenseID, amount ]}
- **producer:** {[ vineyard, address ]}
- **owns:** {[ licenseID → license, vineyard → producer ]} oder  
**owns:** {[ licenseID → license, vineyard → producer ]}

# 1:1-Beziehungstypen

## Initialentwurf

- **license:** {[ licenseID, amount ]}
- **producer:** {[ vineyard, address ]}
- **owns:** {[ licenseID → license, vineyard → producer ]} oder  
**owns:** {[ licenseID → license, vineyard → producer ]}

## Verbesserung durch Zusammenfassung

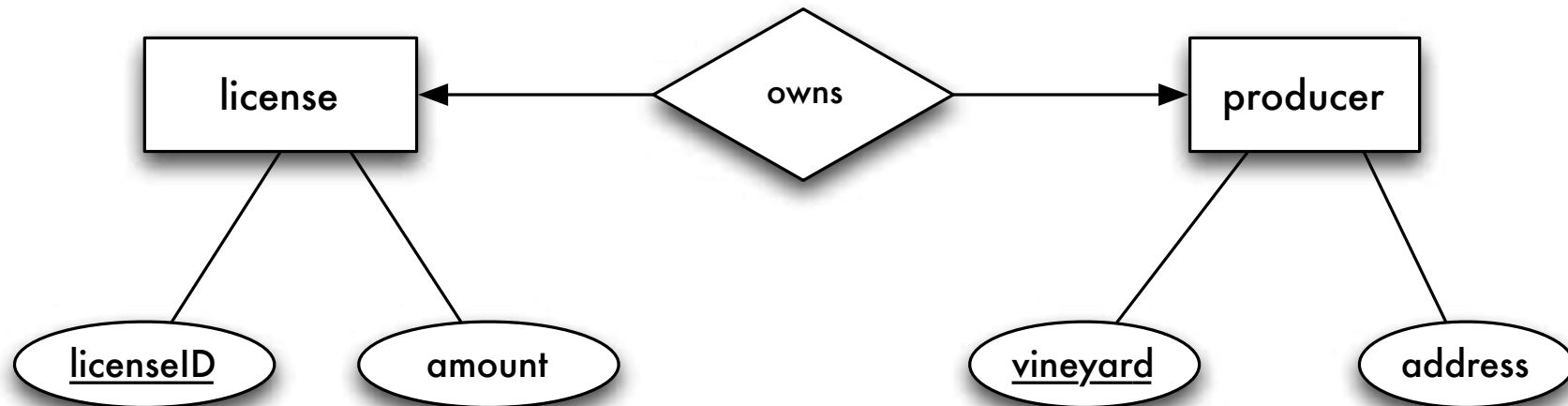
- **license:** {[ licenseID, amount, ownedBy → producer ]}
- **producer:** {[ vineyard, address ]}

oder

- **license:** {[ licenseID, amount ]}
- **producer:** {[ vineyard, address, ownsLicense → license ]}

Am besten erweitert man eine Relation mit **totaler Partizipation**.

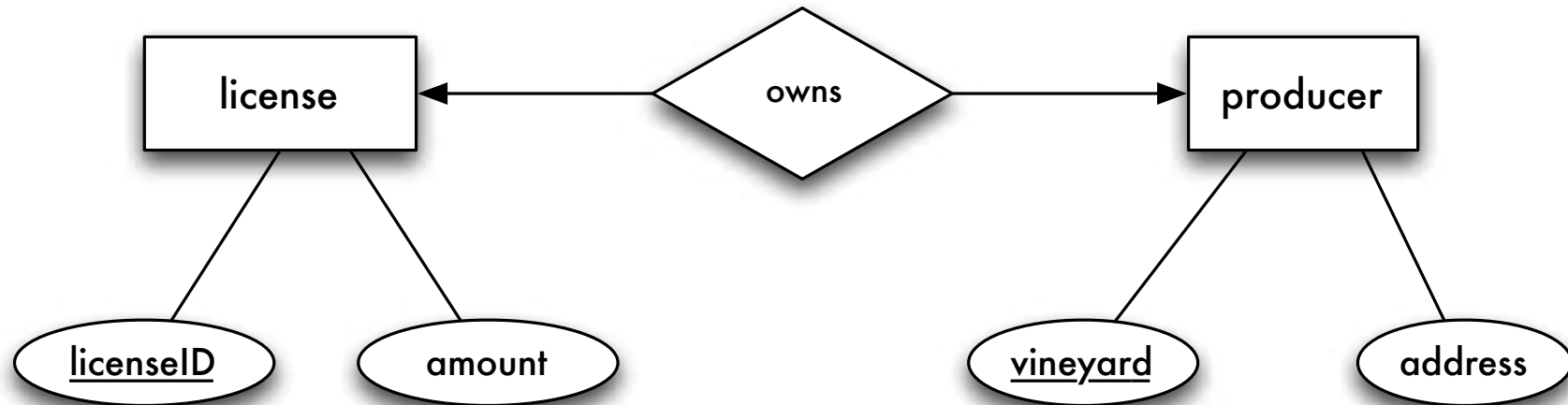
# Warum nicht eine einzige Relation?



producer	vineyard	address	licenseID	amount
	Rotkäppchen	Freiberg	42-007	10.000
	Weingut Müller	Dagstuhl	42-009	250

Wäre nur korrekt bei **totaler Partizipation** ([1,1]) beider beteiligter Entitytypen.

# Warum nicht eine einzige Relation?



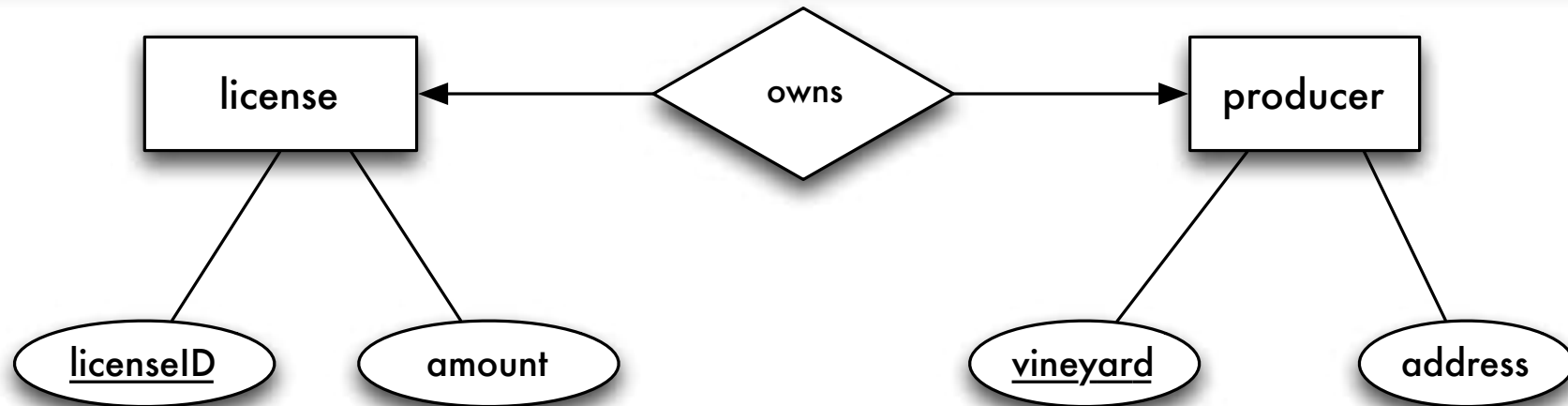
Erzeuger ohne Lizenz würden null-Werte benötigen

producer	vineyard	address	licenseID	amount
	Rotkäppchen	Freiberg	42-007	10.000
	Weingut Müller	Dagstuhl	⊥	⊥

Theoretisch möglich (aber unüblich) bei **partieller Partizipation** ( $[0,1]$ ) einer der beteiligten Entitytypen und **totaler Partizipation** des anderen.

→ führt zu null-Werten

# Warum nicht eine einzige Relation?



Freie Lizenzen führen zu noch mehr null-Werten

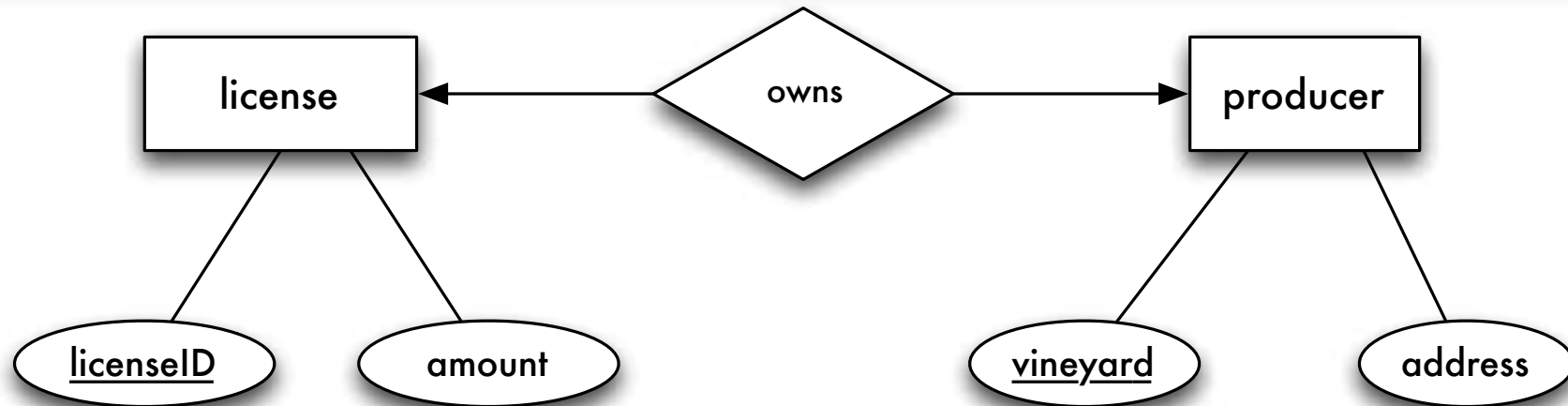
producer	vineyard	address	licenseID	amount
	Rotkäppchen	Freiberg	42-007	10.000
	Weingut Müller	Dagstuhl	⊥	⊥
	⊥	⊥	42-003	100.000

**Partielle Partizipation** beider Entitytypen

→ führt zu null-Werten in allen Attributen

→ Bestimmung eines Primärschlüssel nicht möglich,  
Speicherplatzverschwendung

# Warum nicht eine einzige Relation?



Freie Lizenzen führen zu noch mehr null-Werten

producer	vineyard	address	licenseID	amount
	Rotkäppchen	Freiberg	42-007	10.000
	Weingut Müller	Dagstuhl	⊥	⊥
	⊥	⊥	42-003	100.000

- Im Allgemeinen: kein Zusammenfassen zu einer einzigen Relation!
- Standardansatz: 2 Relationen (ein paar null-Werte sind kein Problem)

# Zusammenfassung: Beziehungstypen auf Relationen abbilden

## M:N-Beziehungstyp

- Neue Relation mit Attributen des Beziehungstyps
- Attribute hinzufügen, die die Primärschlüssel der involvierten Entitytypen referenzieren (Fremdschlüssel)
- Primärschlüssel: Menge der Fremdschlüssel

## 1:N-Beziehungstyp

- Attribute zur Relation des Entitytyps auf der „N“-Seite hinzufügen:
  - Fremdschlüssel, der den Primärschlüssel des Entitytyps auf der „1“-Seite referenziert
  - Attribute des Beziehungstyps hinzufügen

## 1:1-Beziehungstyp

- Attribute zur Relation eines der involvierten Entitytypen hinzufügen:
  - Fremdschlüssel, der den Primärschlüssel des Entitytypen der anderen Seite referenziert
  - Attribute des Beziehungstyps hinzufügen

# Fremdschlüssel

Ein **Fremdschlüssel** ist ein Attribut (oder eine Kombination von Attributen) einer Relation, das auf den Primärschlüssel (oder Schlüsselkandidaten) einer anderen Relation verweist.

## Beispiel

- **course:** {[ courseID, title, ects, **taughtBy** ]}
- **professor:** {[ emplID, name, rank, office ]}

taughtBy ist ein Fremdschlüssel, der auf die Relation professor verweist.

## Notation

- **course:** {[ courseID, title, ects, **taughtBy** → **professor** ]}
- **professor:** {[ emplID, name, rank, office ]}



# Fremdschlüssel

Ein **Fremdschlüssel** ist ein Attribut (oder eine Kombination von Attributen) einer Relation, das auf den Primärschlüssel (oder Schlüsselkandidaten) einer anderen Relation verweist.

## Notation

- **course:** {[ courseID, title, ects, **taughtBy** → **professor** ]}
- **professor:** {[ emplID, name, rank, office ]}

## Alternative Notation

- **course:** {[ courseID, title, ects, **taughtBy** ]}
- **professor:** {[ emplID, name, rank, office ]}

**Foreign key: course.taughtBy → professor.emplID**

Notation für zusammengesetzte Schlüssel:  $\{R.A_1, R.A_2\} \rightarrow \{S.B_1, S.B_2\}$

- 1 Datenbankentwurf
  - Schritte des Datenbankentwurfs
  - Datenbankentwurf am Beispiel
- 2 Grundkonzepte
  - Beispielszenarien
  - Entitytypen
  - Attribute
  - Beziehungstypen
- 3 Eigenschaften von Beziehungstypen
  - Stelligkeit
  - Chen-Notation (Funktionalität)
  - Participation Constraints
  - Chen-Notation (Funktionalität) bei n-stelligen Beziehungstypen
  - $[min, max]$ -Notation (Kardinalität)
- 4 Zusätzliche Konzepte
  - Schwache Entitytypen

- Der ISA-Beziehungstyp

## 5 Alternative Notationen

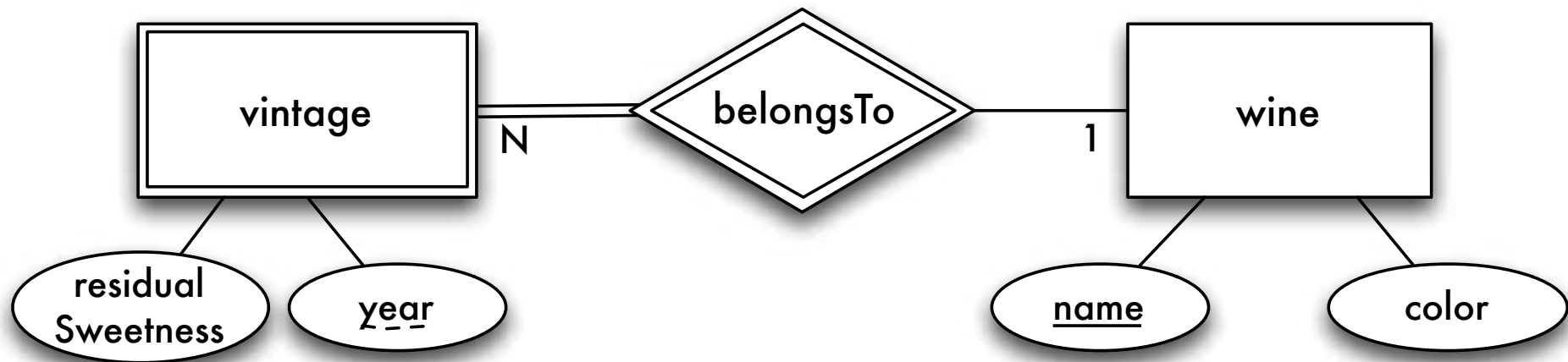
## 6 Relationen aus Grundkonzepten ableiten

- Entitytypen
- Beziehungstypen

## 7 Relation aus zusätzlichen Konzepten ableiten

- Schwache Entitytypen
- Rekursive Beziehungstypen
- N-äre Beziehungstypen
- Spezielle Attribute
- Generalsierung

## Schwache bzw. (existenz-)abhängige Entitytypen

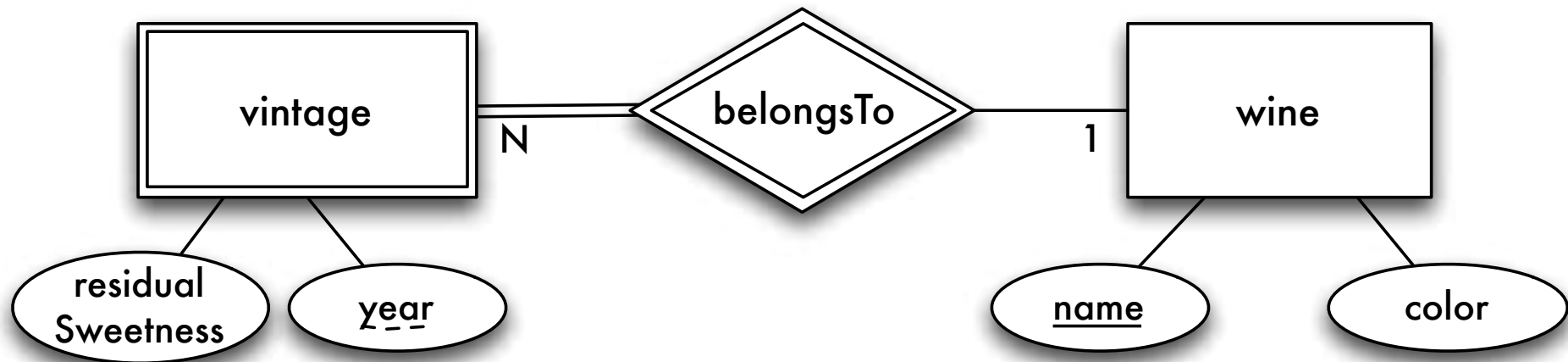


Entitäts eines schwachen Entitytyps sind

- in ihrer Existenz von einem anderen, übergeordneten (starken) Entitytypen abhängig
- normalerweise nur in Kombination mit dem Schlüssel des übergeordneten Entitytypen eindeutig identifizierbar

Wie kann man schwache Entitytypen als Relationen abbilden?

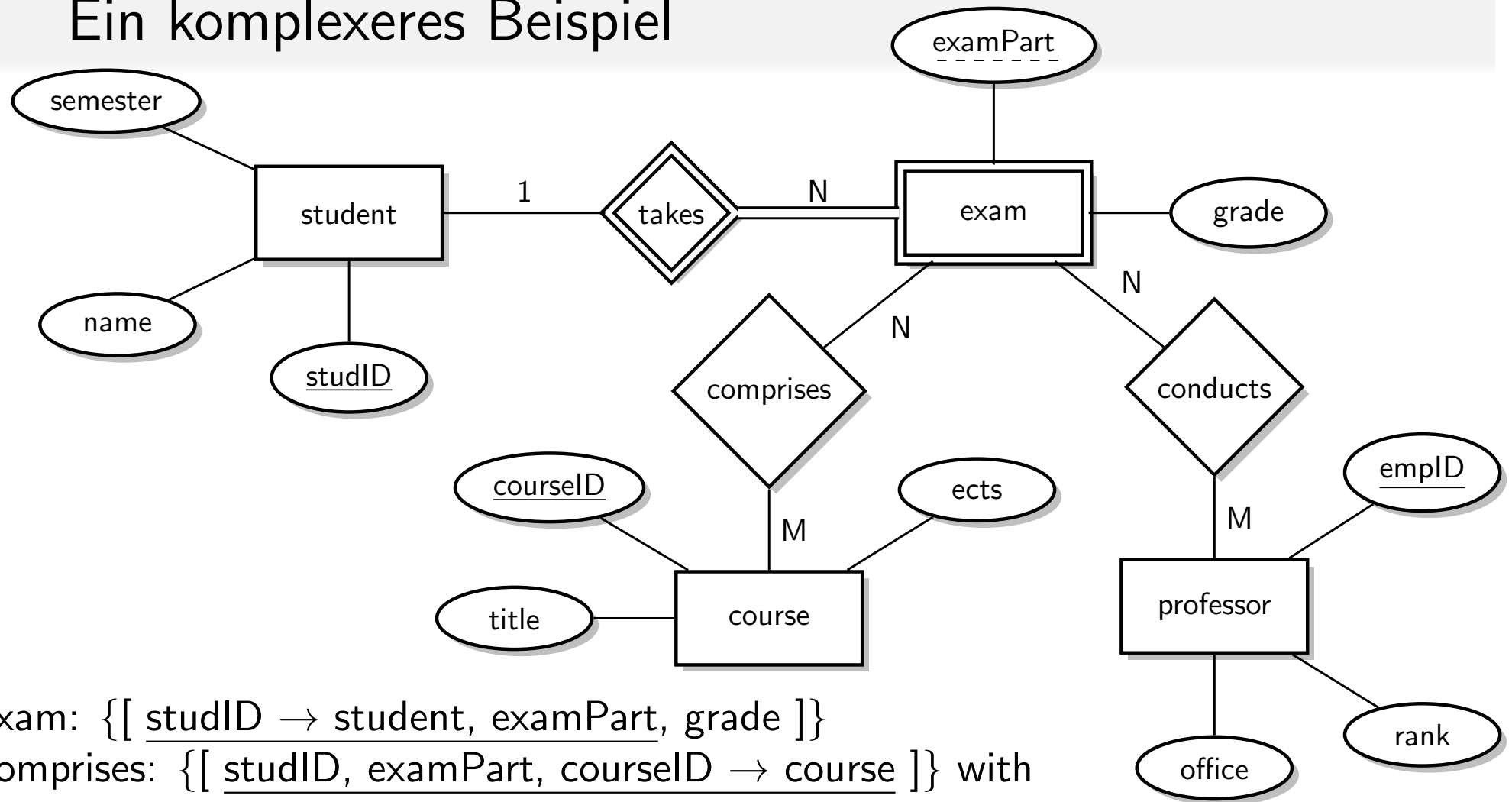
# Schwache Entitytypen



Schwache Entitytypen und deren identifizierende Beziehungstypen können **immer** zusammengeführt werden.

- wine: {[ color, name ]}
- vintage: {[ name → wine, year, residualSweetness ]}

# Ein komplexeres Beispiel



exam: {[ studID → student, examPart, grade ]}

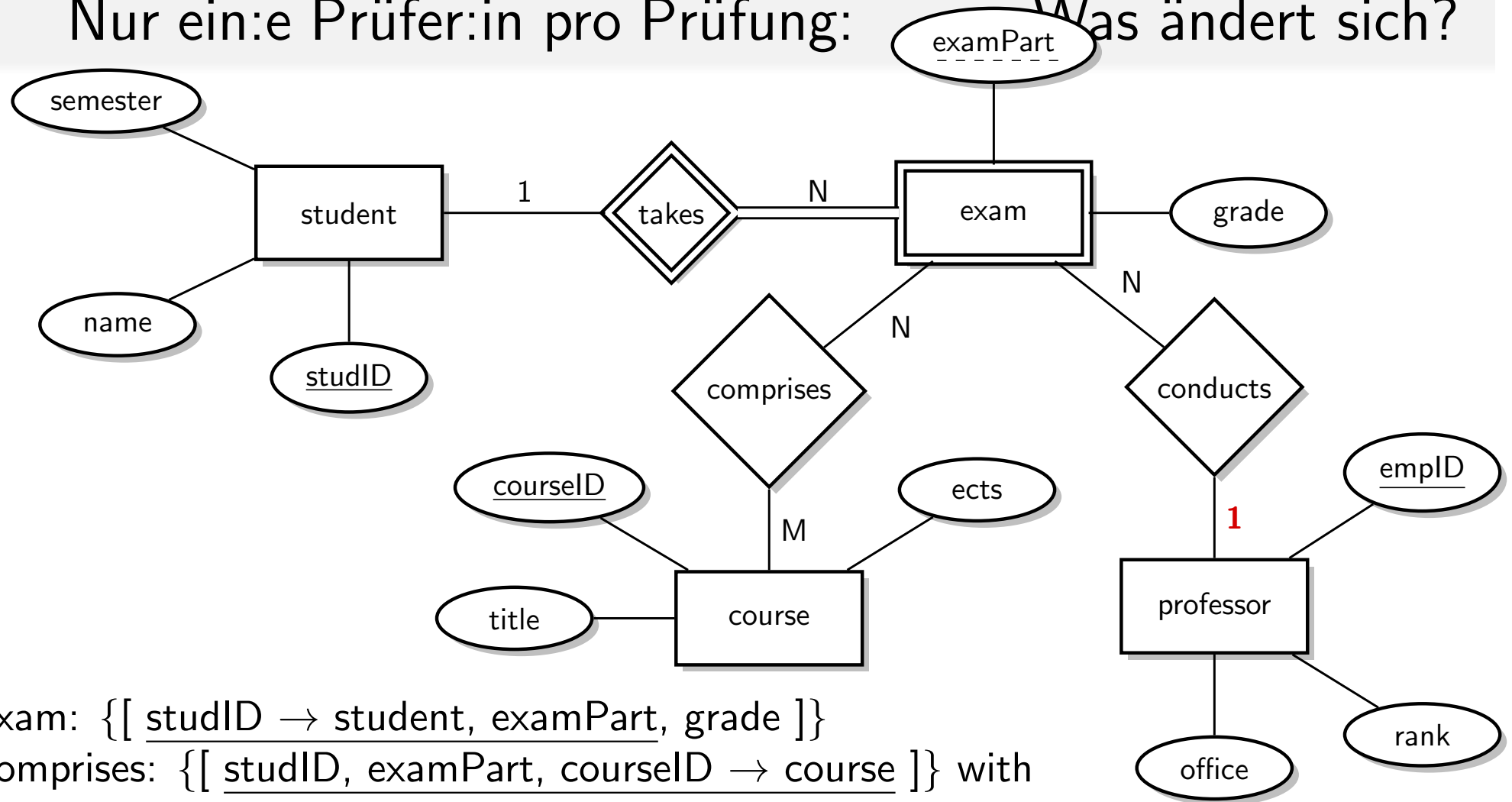
comprises: {[ studID, examPart, courseID → course ]} with

Foreign key: {*studID*, *examPart*} → {*exam.studID*, *exam.examPart*}

conducts: {[ studID, examPart, emplID → professor ]} with

Foreign key: {*studID*, *examPart*} → {*exam.studID*, *exam.examPart*}

# Nur ein:e Prüfer:in pro Prüfung: Was ändert sich?



exam: {[ studID → student, examPart, grade ]}

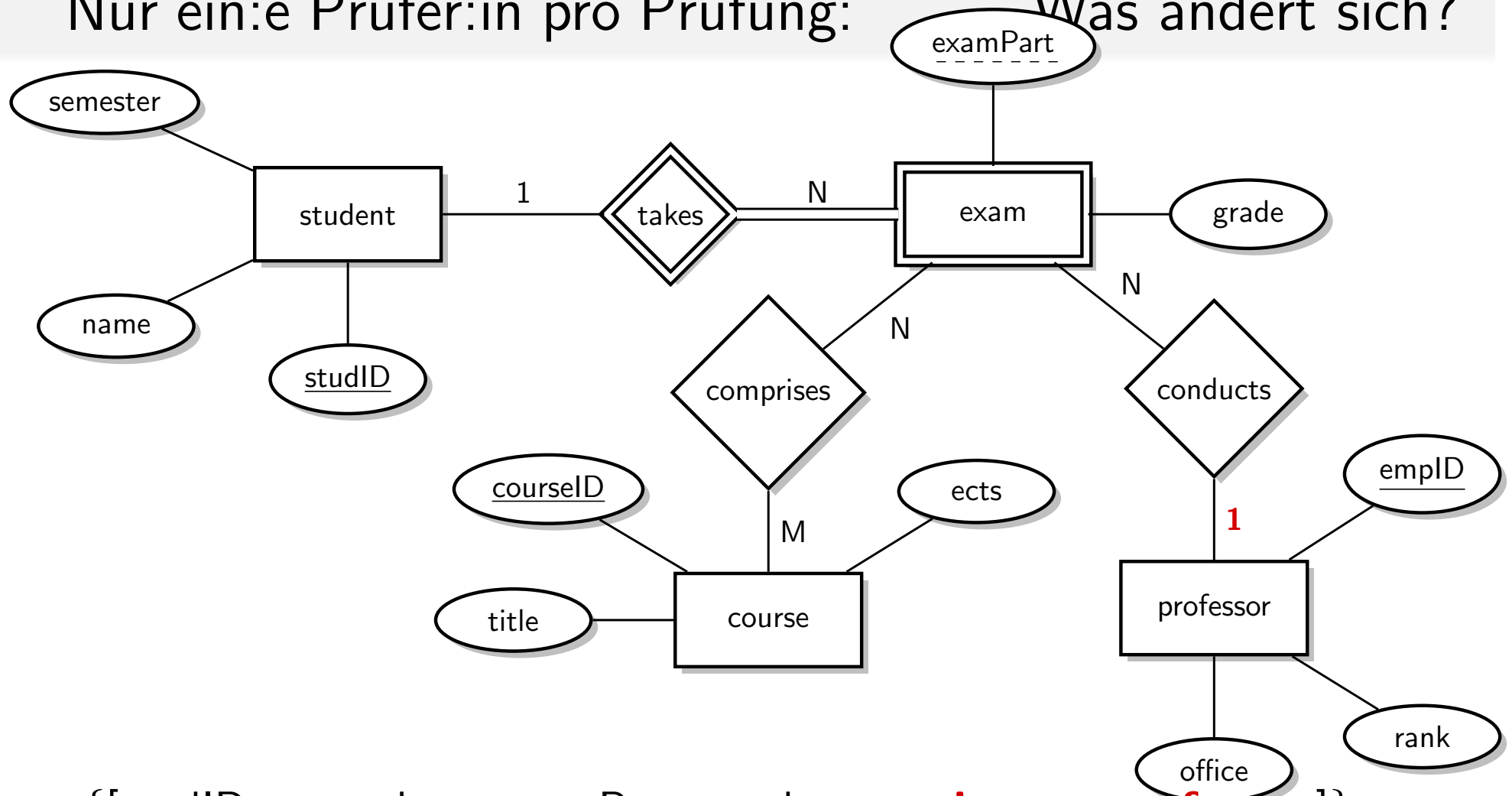
comprises: {[ studID, examPart, courseID → course ]} with

Foreign key: {studID, examPart} → {exam.studID, exam.examPart}

conducts: {[ studID, examPart, emplID → professor ]} with

Foreign key: {studID, examPart} → {exam.studID, exam.examPart}

Nur ein:e Prüfer:in pro Prüfung: Was ändert sich?



exam: {[studID → student, examPart, grade, **examiner** → **professor** ]}

comprises: {[studID, examPart, courseID → course ]} with

Foreign key: {*studID*, *examPart*} → {*exam.studID*, *exam.examPart*}



# Rekursive Beziehungstypen

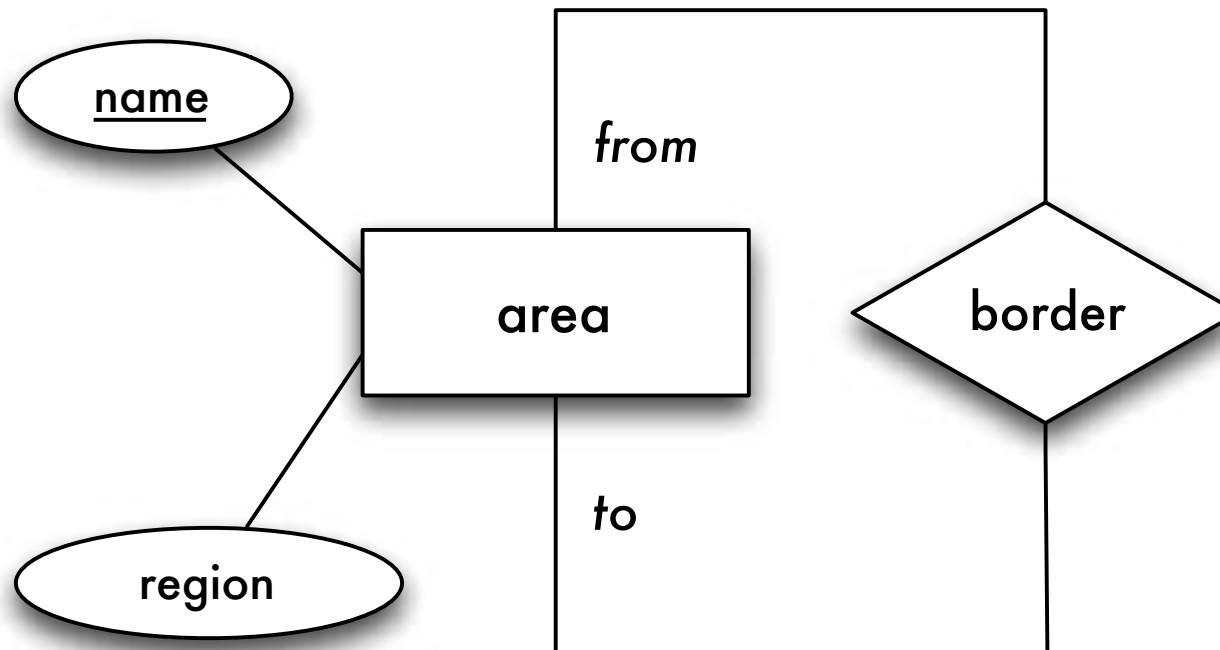


Abbildung wie normale N:M-Beziehungstypen mit Umbenennung der Fremdschlüssel

- area: {[ name, region ]}
- border: {[ from → area, to → area ]}

# Rekursive funktionale Beziehungstypen

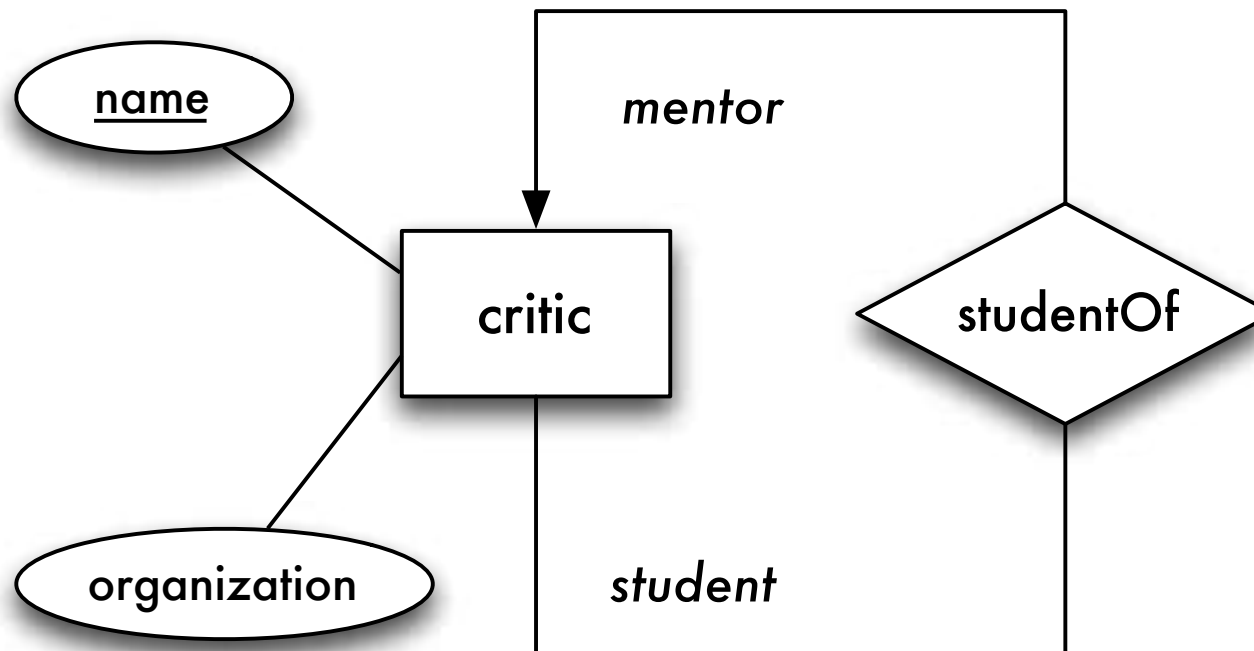
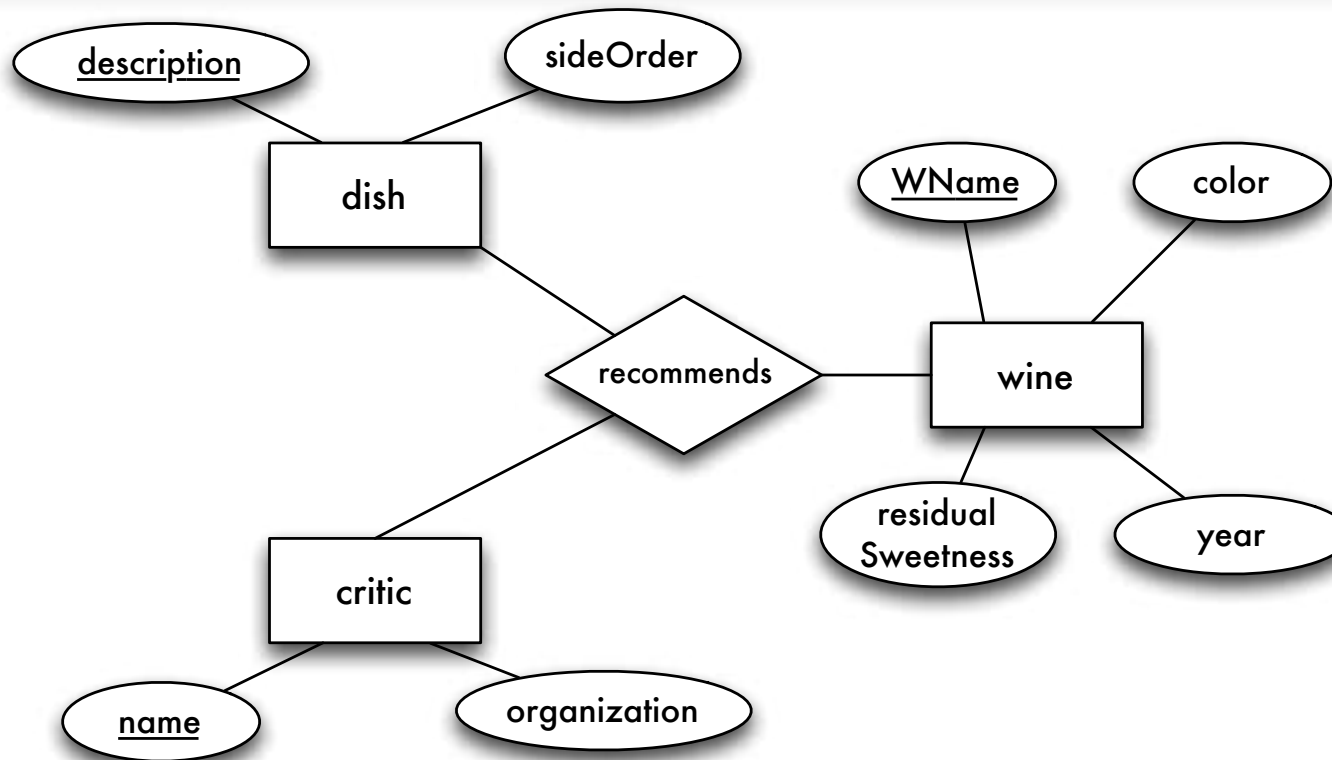


Abbildung wie normale 1:N.Beziehungstypen inklusive Zusammenführung

- critic:  $\{ [ \text{name}, \text{organization}, \text{mentor} \rightarrow \text{critic} ] \}$

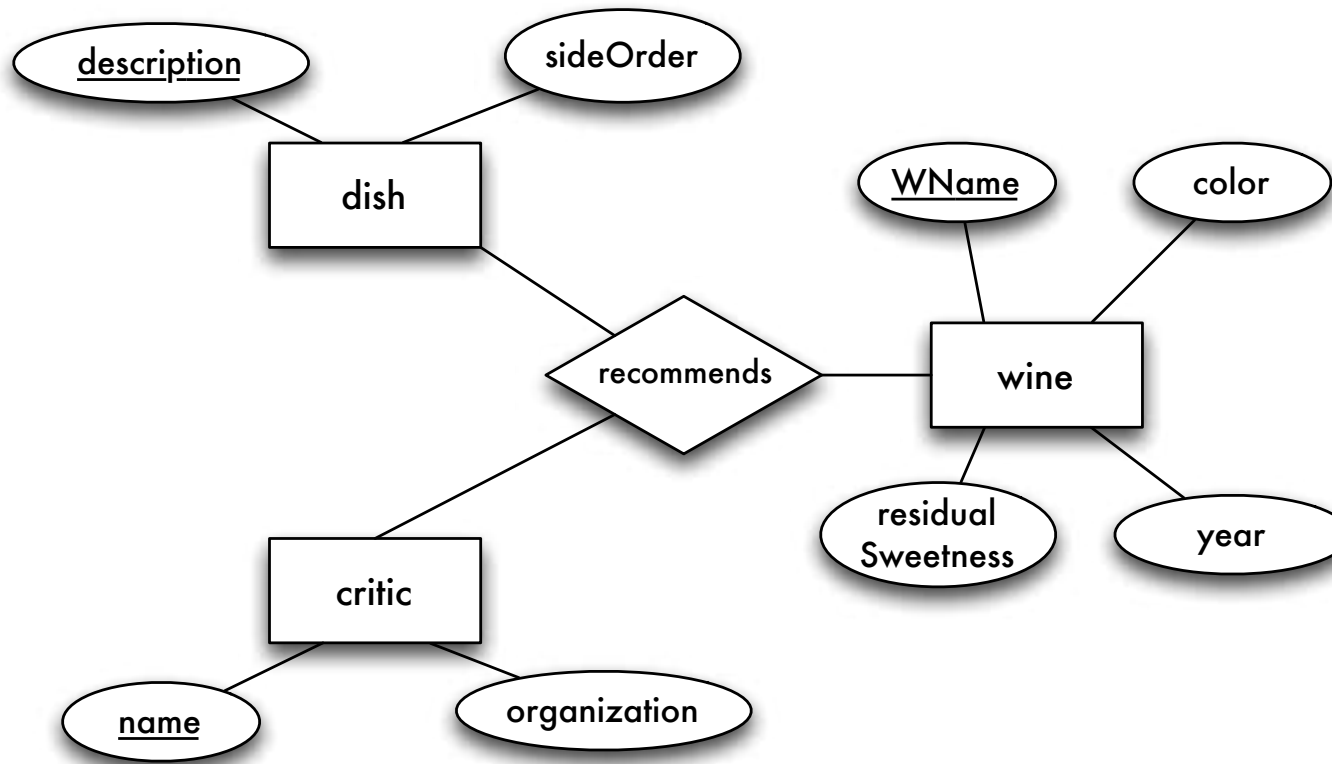
# N-äre Beziehungstypen



Entitytypen werden zunächst nach den Standardregeln abgebildet.

- critic: {[ name, organization ]}
- dish: {[ description, sideOrder ]}
- wine: {[ color, WName, year, residualSweetness ]}

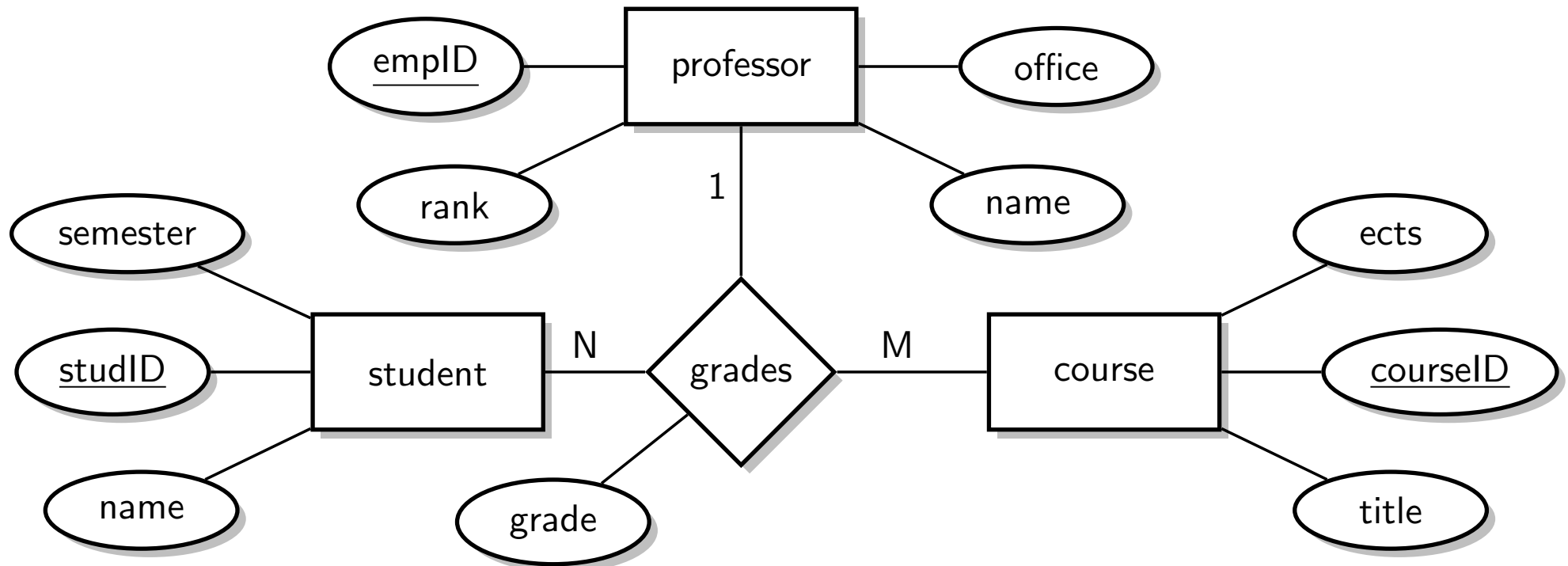
# N-äre Beziehungstypen



N-äre Beziehungstypen (N:M:P)

recommends: {[ WName → wine, description → dish, name → critic ]}

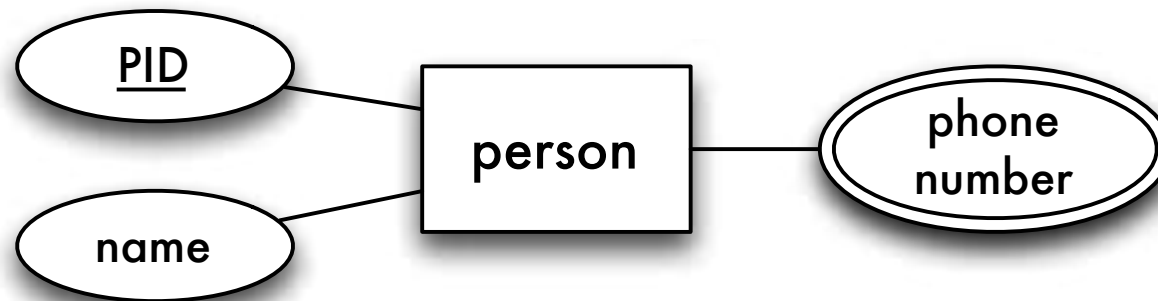
# N:M:1-Beziehungstyp



## Relationen

- student:  $\{[\text{studID}, \text{name}, \text{semester}]\}$
- course:  $\{[\text{courseID}, \text{title}, \text{ects}]\}$
- professor:  $\{[\text{emplID}, \text{name}, \text{rank}, \text{office}]\}$
- grades:  $\{[\text{studID} \rightarrow \text{student}, \text{courseID} \rightarrow \text{course}, \text{emplID} \rightarrow \text{professor}, \text{grade}]\}$

# Mehrwertige Attribute

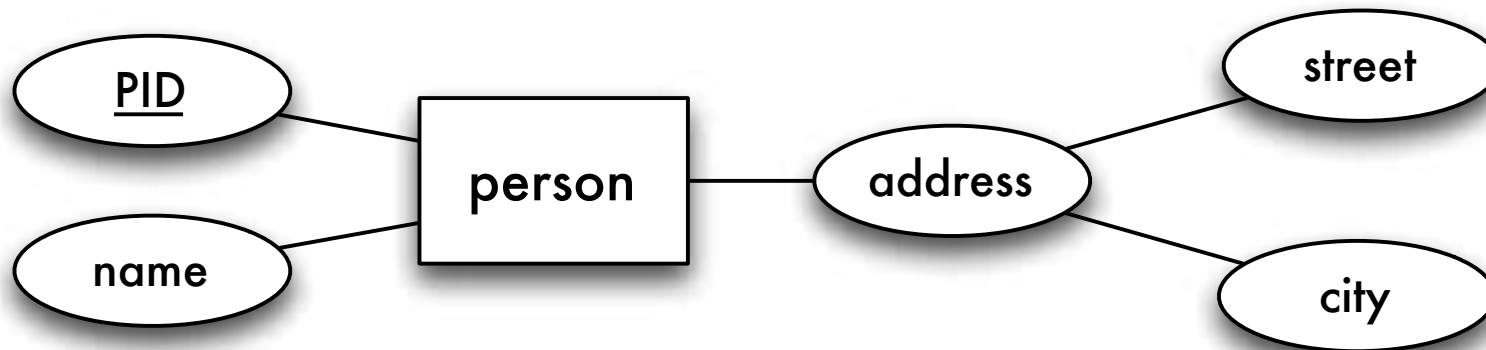


Es wird eine separate Relation für jedes mehrwertige Attribut erstellt.

## Relationen

- person: {[ PID, name ]}
- phoneNumber: {[ PID → person, number ]}

# Zusammengesetzte Attribute

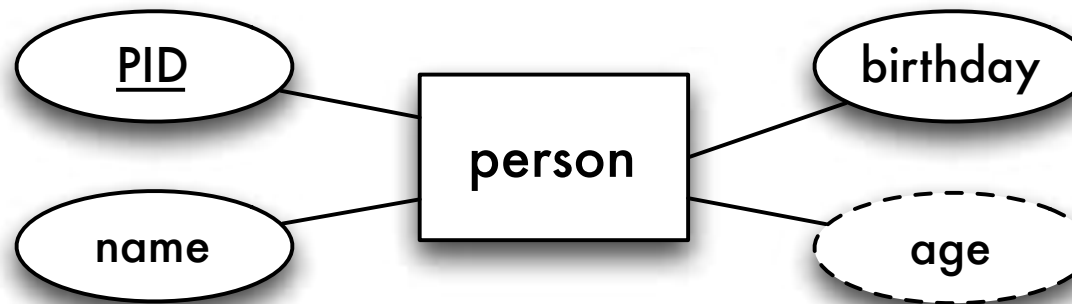


Zusammengesetzte Attribute werden der Relation des zugehörigen Entitytypen hinzugefügt.

Relation

- person: {[ PID, name, street, city ]}

# Abgeleitete Attribute



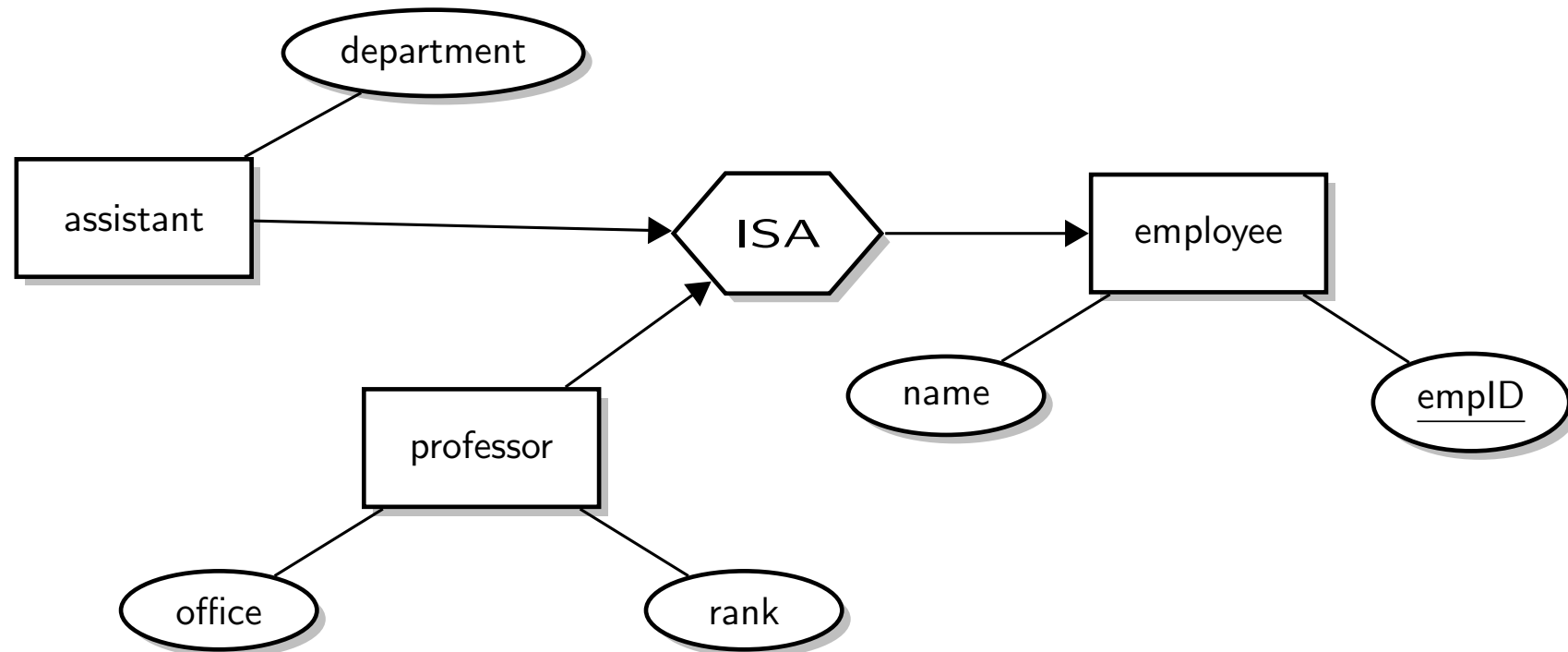
Diese werden bei der Abbildung in Relationen ignoriert und später mittels Views hinzugefügt.



# Überblick der Schritte

- ① Regulärer Entitytyp  
Relation erstellen, spezielle Attributtypen beachten
- ② Schwacher Entitytyp  
Relation erstellen
- ③ 1:1 binärer Beziehungstyp  
Erweitern einer Relation mit Fremdschlüssel
- ④ 1:N binärer Beziehungstyp  
Erweitern einer Relation mit Fremdschlüssel
- ⑤ N:M Beziehungstyp  
Erstellen einer Relation
- ⑥ N-ärer Beziehungstyp  
Erstellen einer Relation

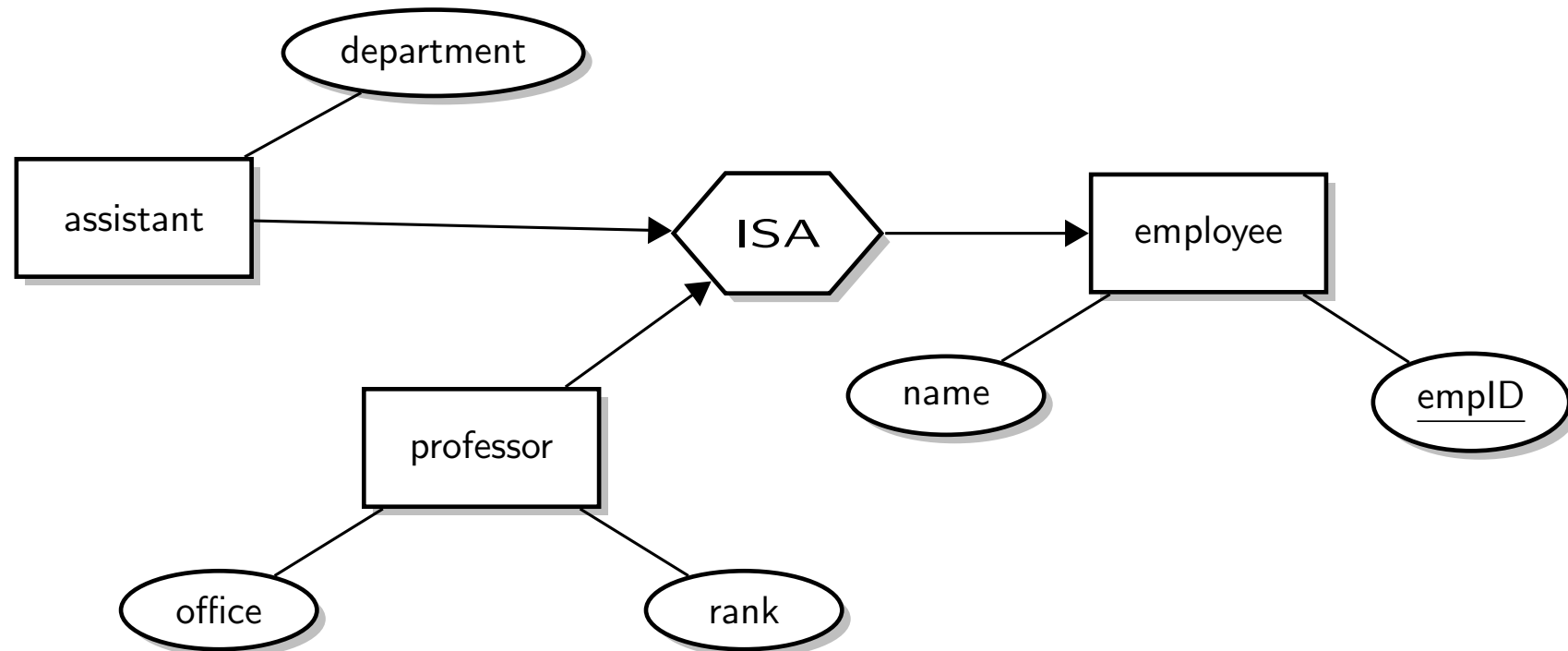
# Relationale Modellierung der Generalsierung



Das relationale Modell unterstützt keine Generalisierung und kann daher keine Vererbung ausdrücken.

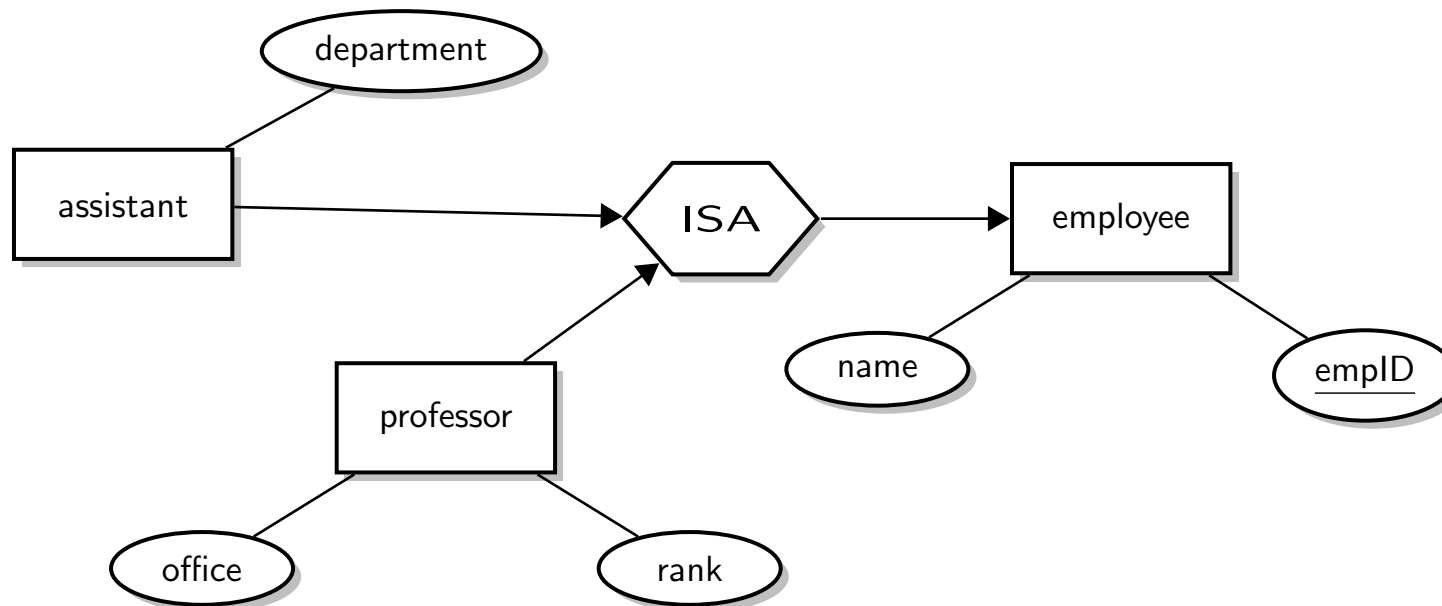
→ Generalisierung wird **simuliert**.

# Relationale Modellierung der Generalsierung



Wie kann dieses ER-Diagramm auf Relationen abgebildet werden?

## Alternative 1: Hauptklassen



Ein bestimmtes Entity wird abgebildet als **ein Tupel** in einer einzigen Relation (zur zugehörigen Hauptklasse).

- employee: {[ empID, name ]}
- professor: {[ empID, name, rank, office ]}
- assistant: {[ empID, name, department ]}

# Alternative 1: Hauptklassen

employee	
<u>emplID</u>	name
2123	P. Müller
2124	A. Schmidt

employee:  
 {[ emplID, name ]}

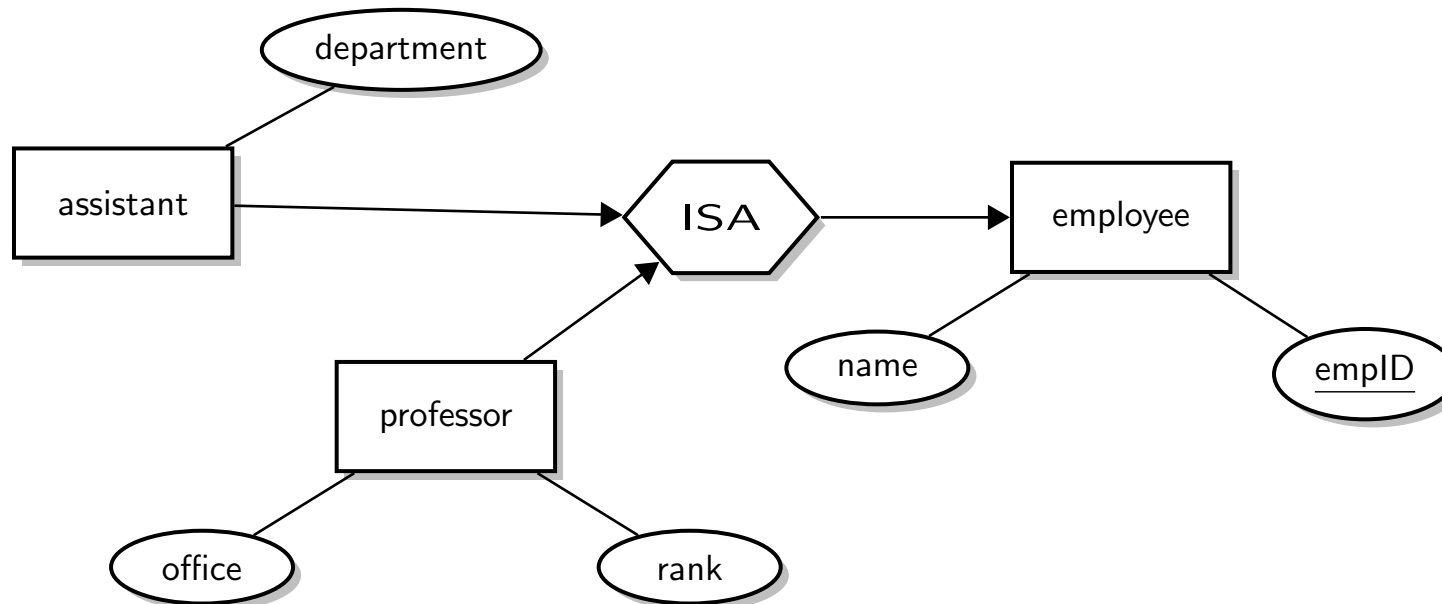
professor			
<u>emplID</u>	name	rank	office
2125	Socrates	C4	226
2126	Russel	C3	232
2127	Kopernikus	C3	310
2128	Curie	C4	36

professor:  
 {[ emplID, name, rank, office ]}

assistant		
<u>emplID</u>	name	department
2150	C. Meyer	DBS
2151	B. Fischer	Physics

assistant:  
 {[ emplID, name, department ]}

## Alternative 2: Partitionierung



**Teile eines** bestimmten Entitys werden in mehreren Relationen abgebildet, der Schlüssel wird dupliziert.

- employee: {[ empID, name ]}
- professor: {[ empID → employee, rank, office ]}
- assistant: {[ empID → employee, department ]}

## Alternative 2: Partitionierung

employee	
<u>empID</u>	name
2123	P. Müller
2124	A. Schmidt
2125	Socrates
...	...
2150	C. Meyer
2151	B. Fischer

employee:  
 $\{[ \underline{\text{empID}}, \text{name} ]\}$

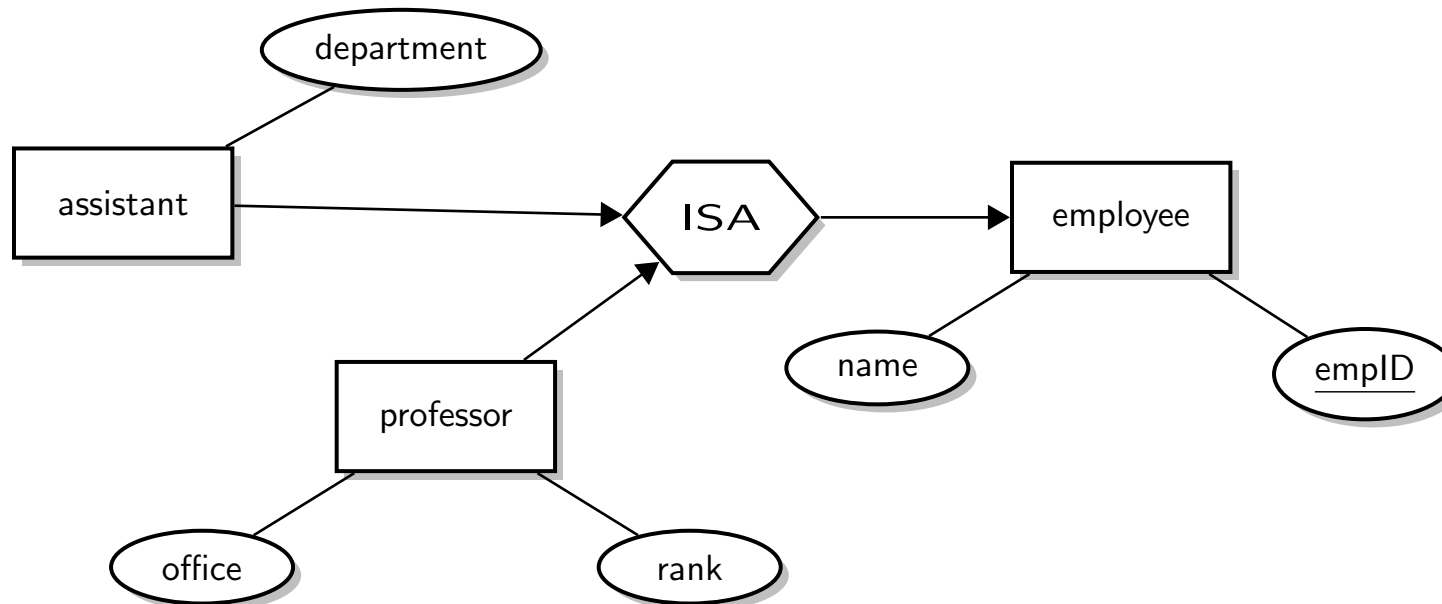
professor		
<u>empID</u>	rank	office
2125	C4	226
...	...	...

professor:  
 $\{[ \underline{\text{empID}} \rightarrow \text{employee}, \text{rank}, \text{office} ]\}$

assistant	
<u>empID</u>	department
2150	DBS
2151	Physics

assistant:  
 $\{[ \underline{\text{empID}} \rightarrow \text{employee}, \text{department} ]\}$

## Alternative 3: Vollständige Redundanz



Ein bestimmtes Entity wird **redundant** in mehreren Relationen gespeichert inklusive aller geerbten Attribute.

- employee: {[ empID, name ]}
- professor: {[ empID, name, rank, office ]}
- assistant: {[ empID, name, department ]}



## Alternative 3: Vollständige Redundanz

employee	
<u>emplID</u>	name
2123	P. Müller
2124	A. Schmidt
2125	Socrates
...	...
2150	C. Meyer
2151	B. Fischer

employee:  
 $\{[ \underline{\text{emplID}}, \text{name} ]\}$

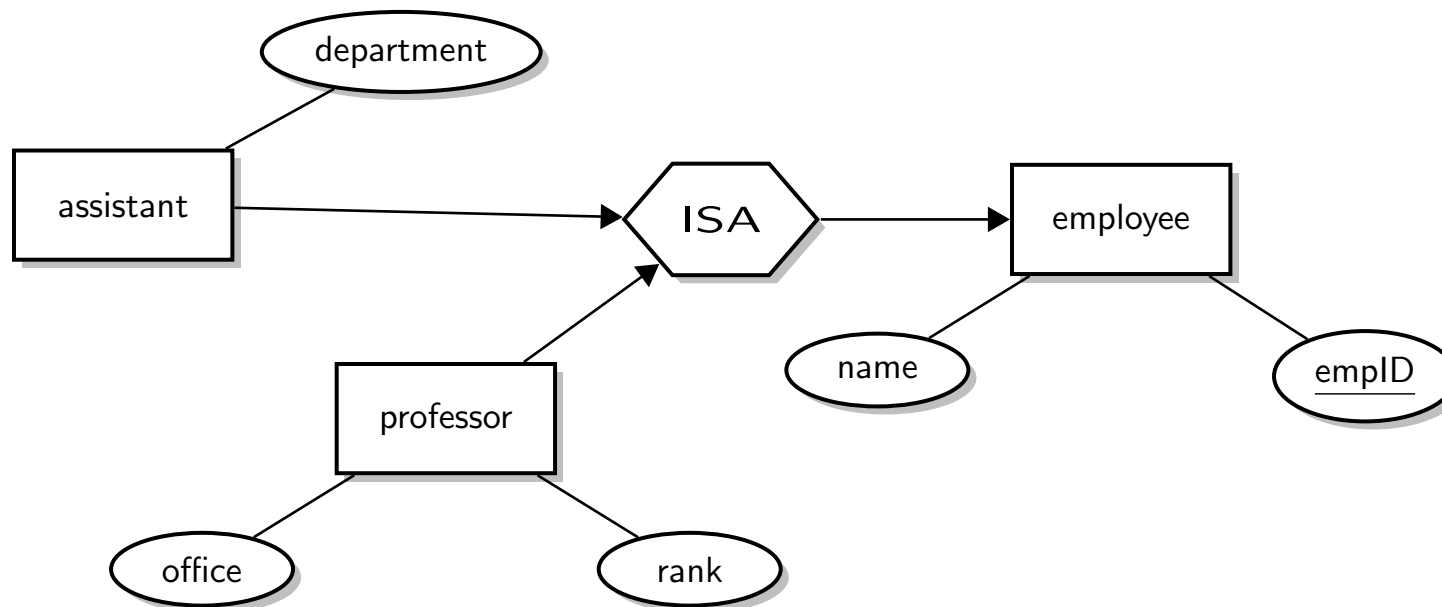
professor			
<u>emplID</u>	name	rank	office
2125	Socrates	C4	226
...	...	...	...

professor:  
 $\{[ \underline{\text{emplID}}, \text{name}, \text{rank}, \text{office} ]\}$

assistant		
<u>emplID</u>	name	department
2150	C. Meyer	DBS
2151	B. Fischer	Physics

assistant:  
 $\{[ \underline{\text{emplID}}, \text{name}, \text{department} ]\}$

## Alternative 4: Eine einzige Relation



Alle Entitäts werden in **einer einzigen Relation** gespeichert und ein besonderes Attribut hinzugefügt, welches die Zugehörigkeit zu einem bestimmten Entitytyp angibt.

- employee: {[ emplID, name, **type**, rank, office, department ]}

## Alternative 4: Eine einzige Relation

employee: {[ emplD, name, type, rank, office, department ]}

employee					
<u>emplD</u>	name	type	rank	office	department
2123	P. Müller	employee	⊥	⊥	⊥
2124	A. Schmidt	employee	⊥	⊥	⊥
2125	Socrates	professor	C4	226	⊥
2126	Russel	professor	C3	232	⊥
2127	Kopernikus	professor	C3	310	⊥
2128	Curie	professor	C4	36	⊥
2150	C. Meyer	assistant	⊥	⊥	DBS
2151	B. Fischer	assistant	⊥	⊥	Physics

# Zusammenfassung

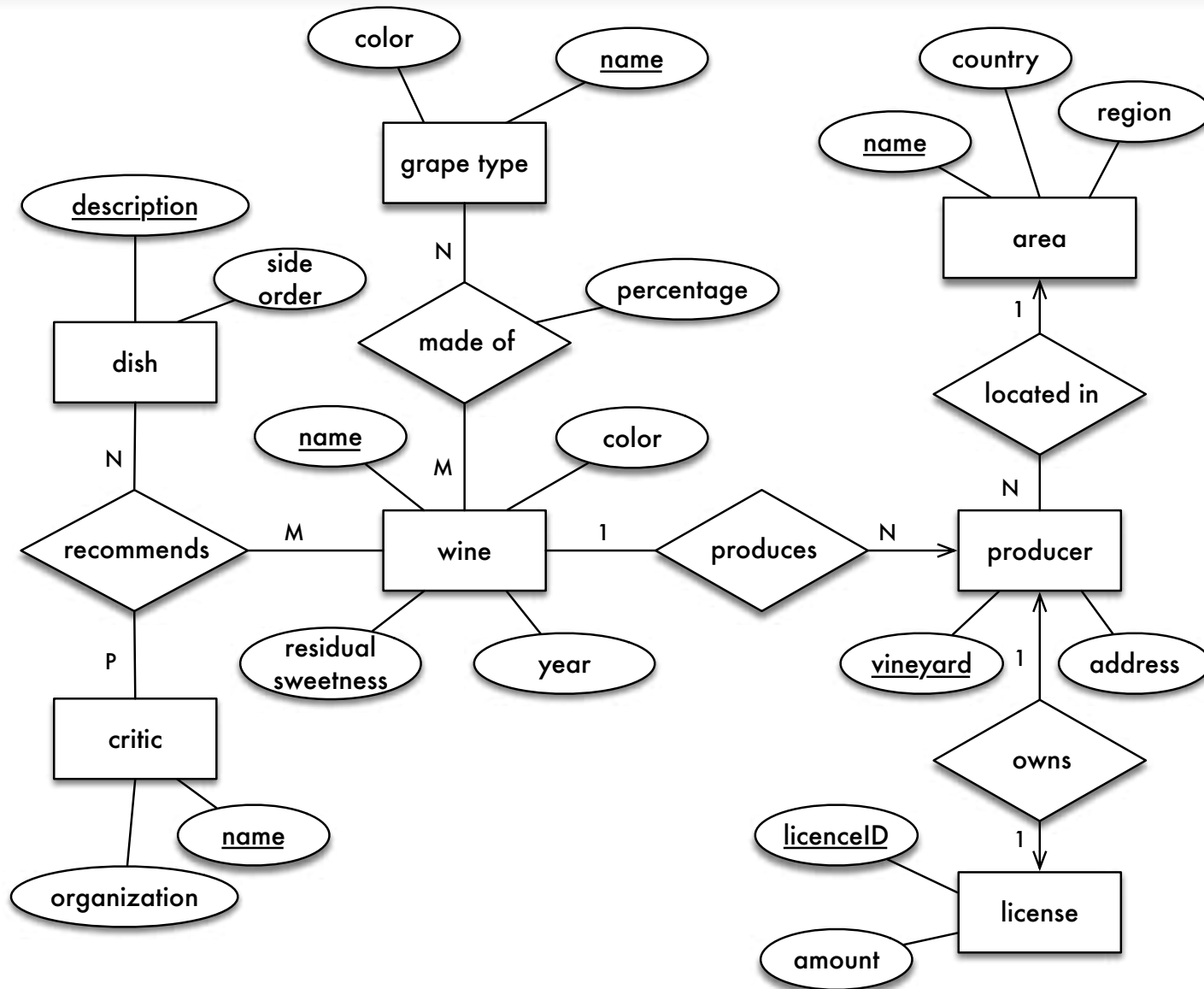
## ER-Diagramme auf Relationen abbilden

- Entitytypen
- Binäre Beziehungstypen
- N-stellige Beziehungstypen
- Schwache Entitytypen
- Rekursive Beziehungstypen
- Generalisation
  - Die „Partitionierungsvariante“ wird in den meisten Anwendungen bevorzugt.
- Die behandelten „Abbildungsregeln“ haben das Ziel, die Anzahl von Relationen zu minimieren, nicht notwendigerweise die null-Werte.

# Appendix

## 8 Beispiele

# Wine-Schema mit Chen-Notation



# Universitätsschema mit Chen-Notation

