Kapitel 2: Inhaltsübersicht

- Die Ausführungen in Kapitel 2 konzentrieren sich auf das E/R-Modell, das gegenwärtig am weitesten verbreitete **semantische Datenmodell** überhaupt.

  UML ist zwar wesentlich „moderner“ und bietet wegen der Berücksichtigung objektorientierter Konstrukte einige Vorteile, hat sich in der Praxis noch nicht auf breiter Front durchgesetzt. UML wird zudem in einer anderen Vorlesung behandelt.

- Nach einigen **grundlegenden Begriffsdefinitionen** wird die „**Komplexität von Beziehungen**“ anhand einiger praktischer Beispiele detailliert behandelt.

- In Abschnitt 2.3 werden abschließend noch einige wenige **Erweiterungen des Grundmodells** von Chen behandelt.

Was ist das Entity-Relationship-Modell?

- Das **Entity-Relationship-Modell** wurde im Jahr 1976 von Chen eingeführt und dient dazu, die Ergebnisse der Analyse-phase formal zu beschreiben.
  siehe auch *Chen, P. P. : The Entity-Relationship Model - Toward a Unified View of Data, ACM Trans. Database Systems 1,1 (March 1976), pp. 9-36*

- E/R-Modelle haben sich in der Praxis beim Entwurf des konzeptionellen Schemas durchgesetzt, wobei es zahlreiche Varianten, Modifikationen und Erweiterungen gibt, z.B. bei der Notation für den Grad einer Beziehung
  - Krähenfußnotation nach Martin (z.B. im CASE-Tool ERWIN)
  - 1:n-Notation
  - (min,max)-Notation

Entities und Entity-Sets

- Ein **Entity e** ist ein konkretes Objekt der realen Welt, das von anderen Objekten eindeutig unterscheidbar ist und unabhängig von diesen existiert.

  Beispiele: _____

- Unter einem **Entity-Set E** versteht man eine Menge von Objekten, die einander nach gewissen Kriterien ähnlich sind.

  Beispiele: _____

- Mit $\mathbf{E}^t$ bezeichnet man die Menge der zu dem Zeitpunkt t tatsächlich vorhanden Entities, die dem Entity-Set E zugeordnet werden.

Attribute und Attributwerte

- Objekte der realen Welt unterscheiden sich von anderen Objekten durch bestimmte Eigenschaften bzw. Merkmale.

- Die für das konzeptionelle Datenbankschema relevanten Eigenschaften und nur diese bezeichnet man als **Attribute**, deren konkrete Ausprägung als **Attributwert**.

  Beispiele: _____

- Jedem Entity-Set E kann daher eine Menge von Attribute A zugeordnet werden. Für diese **Attributmenge** A gilt:

  E: <A> mit A = {a_1,a_2,a_3,...,a_n}

  wobei a_i ein konkretes Attribut ist.

  Anmerkung: Die Attributmenge A besteht immer aus endlich vielen Attributen.

Wertebereich von Attributen

- Mit **dom (a)** bezeichnet man den Wertebereich (engl. domain) eines Attributes a, d.h. die Menge der für dieses Attribut überhaupt möglichen bzw. zulässigen Werte.
  Beispiele: _____
- Die Domains von unterschiedlichen Attributen müssen nicht disjunkt sein; recht häufig sind sie sogar absolut identisch.
- Einen **konkreten Wert** aus dom (a) bezeichnet man als w_a
  Beispiele: _____
- Hinweis: Zum Zeitpunkt t hat ein Entity $e \in E^t$ einen Wert w_a aus dom (a). Dieser kann sich zum Zeitpunkt t+x ändern, muss es aber nicht.
  Beispiele: _____

Wertebereich von Attributmengen

- Analog bezeichnet man den **Wertebereich einer Attributmenge A** mit dom (A) und versteht darunter die Menge aller möglichen (geordneten) Tupel mit
  $w \in dom(A)$ und $dom(A) = \{(w_1, w_2, ..., w_n) \mid w_i \in dom(a_i), i= 1... n\}$

- Anders formuliert ist dom (A) das **kartesische Produkt** der Wertebereiche der einzelnen Attribute:
  $dom(A) = dom(a_1) \times dom(a_2) \times dom(a_3) \times .... \times dom(a_n)$

- Demnach ist ein Entity $e \in E^t$ durch ein geordnetes Tupel $w \in dom(A)$ eindeutig beschrieben.

- **Aufgabe**: Verdeutlichen Sie sich diese Begriffe anhand des Entity-Sets „Studenten“.

Zusammengesetzte und mehrwertige Attribute

- Ein Attribut, das sich aus mehreren, logisch unterschiedlichen Teilen zusammensetzt, wird als **zusammengesetztes** Attribut bezeichnet.
  Beispiel: _____
- Ein Attribut heißt **einwertig** (atomar, elementar), wenn jedes Entity nur genau einen Wert aus dem Wertebereich annimmt und dieser Wert nicht weiter zerlegbar ist.
  Beispiel: _____
- Wenn ein Entity in Bezug auf ein Attribut eine Teilmenge von Werten aus dem Wertebereich des Attributs annehmen kann, wird dieses Attribut als **mehrwertig** bezeichnet.
  Beispiel: _____

Schlüssel (1)

- Jedes Entity e_i vom Entity-Set E^t:<A> ist durch seine Attributwertekombination $w_i \in dom(A)$ eindeutig bestimmt, d.h. A ist **identifizierend** für E^t.

- Diese Eigenschaft kann aber auch bereits für eine Teilmenge von A gelten.

- **1. Definition Schlüssel**: Sei E:<A> ein Entity-Set und $K \subseteq A$. K wird genau dann als Schlüssel bezeichnet, wenn gilt:

  1. K ist identifizierend für E:<A>, d.h. verschiedene Objekte der realen Welt haben auch unterschiedliche Attributwerte bezüglich K.
  2. Es gibt keine echte Teilmenge $K' \subset K$, für die die Eigenschaft (1) gilt. D.h. B ist minimal bezüglich der Eigenschaft (1).

Schlüssel (2)

- Der Schlüssel besteht üblicherweise aus einem einwertigen Attribut oder aus einer Kombination von einwertigen Attributen.
- **Schlüsselkandidaten**: Alle unterschiedlichen Attribute bzw. Attributkombinationen mit der Eigenschaft „Schlüssel“.
- **Primärschlüssel** (primary key): Einziger oder ein aus den Schlüsselkandidaten fest ausgewählter Schlüssel.
- **Nichtschlüsselattribute** (Zusatz- bzw. Sekundärattribute): Alle Attribute, die nicht Teil des Schlüssels sind oder zu einem (anderen) Schlüsselkandidaten gehören.

Praxishinweise zu Schlüsseln

- Häufig gefordert, dass der Primärschlüssel seinen Wert während der gesamter „Lebenszeit“ des von ihm referenzierten Objektes nicht ändert. Daher werden gerne "künstliche" Schlüssel verwendet, die der DB-Designer kontrollieren kann.
  - Matrikelnummer, Personalnummer, Steuernummer etc.
  - Diverse Artikelnummern, z.B. HAN bzw. LAN (Hersteller- / Lieferantenartikelnummer), EAN 13 (European Articel Number) und GTIN (Global Trade Item Number)
- Manchmal ist es hilfreich, in derartige Schlüssel bewusst Zusatzinformationen (sog. sprechende Schlüssel) zu codieren. Hierzu werden z.B. Nummernkreise wie folgt definiert.
  - Kundennummer 50123 bzw. 40365 bzw. 30100, wobei die ersten beiden Ziffern die Kundengruppe identifizieren, z.B. 50 = gewerblich, 40 = privat und 30 = öffentlich.
  - Länderkennziffer bei der EAN / GTIN im Einzelhandel.

Definition: Entity-Typen

- Entities aus einem Entity-Set E^t bezeichnet man als Enities vom Typ E. Ein **Entity-Typ E (A,K)** besteht aus:
  - einem Namen E,
  - einer Attributmenge A und
  - einem Primärschlüssel K ($K \subseteq A$).
- Die Attributmenge A kann enthalten:
  - einwertige Attribute: a
  - mehrwertige Attribute: a*
  - zusammengesetzte Attribute: a ($a_1,...,a_n$)
- Beispielsweise Entity-Set Studenten:

Beziehungen (relationship)

- So wie Objekte der realen Welt in vielfältigen Beziehungen zueinander stehen, spricht man im Datenbankumfeld auch von Beziehungen zwischen verschiedenen Entities.
- Typische Beispiele sind
  - Der Angestellte Jochen Müller arbeitet in der Abteilung SWE
  - Der Auftrag 4711 umfasst die Artikel mit den Nr. 0815, 0816 und 0817.
- Die Unterscheidung zwischen Entity und Beziehung ist nicht immer ganz eindeutig bzw. kann auch vom Kontext abhängig sein.

Beziehungstypen (1)

- Analog der Aggregation gleichartiger Enities zu Entity-Typen, kann man Beziehungen mit identischer Bedeutung zu einer Menge, dem sog. **Beziehungstyp R** zusammenfassen, z. B.
  - Mitarbeit — Angestellte arbeiten x % der WAZ an Projekten
  - Zugehörigkeit — Angestellte gehören zu Abteilungen

- Als **Grad eines Beziehungstyps** bezeichnet man die Anzahl der beteiligten Entity-Typen.

- Beziehungstypen können wie Entity-Typen Attribute haben, woraus dann konkrete Attributwerte zwischen den beteiligten Entities resultieren.
  Beispiel: _____

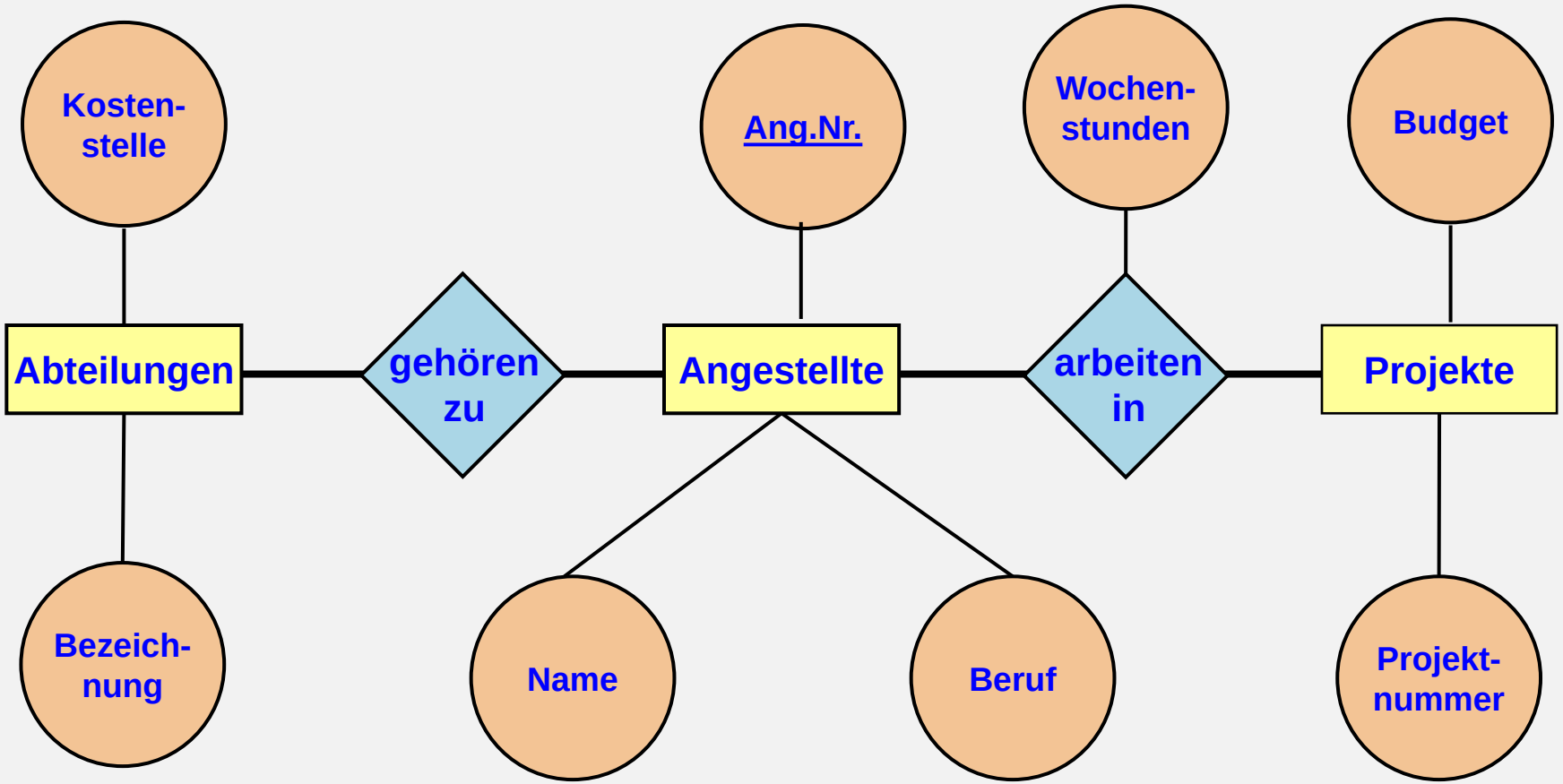
Beziehungstypen (2)

- Formal ist ein Beziehungstyp R charakterisiert durch
  - einen Namen B,
  - eine Menge von beteiligten Entity-Typen E_1, E_2, …, E_n und
  - einer (auch leeren) Menge von (Zusatz-) Attributen Z_1, Z_2, …, Z_n
- Folglich stellt die Ausprägung der Beziehung R eine Teilmenge des kartesischen Produktes der an der Beziehung beteiligten Entity-Typen dar und man formuliert:

$$R^t \subseteq E_1^t \times E_2^t \times .... \times E_n^t$$

- Beispiele:
  - Mitarbeit (Angestellte, Projekte / Std. / Woche) → formal R (E_1,E_2 / Z_1)
  - Zugehörigkeit (Mitarbeiter, Abteilungen / Datum)

Grafische Darstellung



- Anmerkung: Zur besseren Übersicht werden Attribute häufig ohne Kreissymbol dargestellt oder man bildet nur die Schlüsselattribute ab.

Spezialfall: Rollen

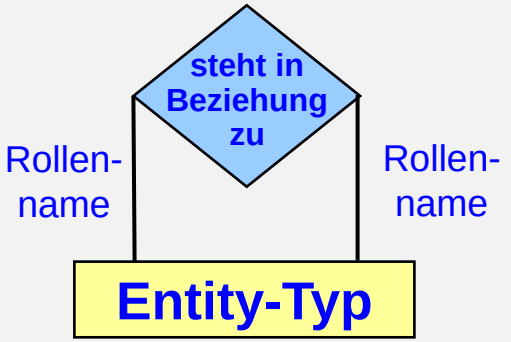
- Bisher wurde nicht ausdrücklich darauf hingewiesen, dass auch Beziehungen zwischen Entities vom selben Typ möglich sind. Die bisherigen Definitionen schließen diese Möglichkeit aber auch nicht explizit aus.

- Formal kann man diese Situation wie folgt darstellen, muss aber zur Beschreibung sogenannte Rollennamen verwenden:

$$R^t \subseteq E^t_1 \times E^t_1$$

- Beispiele:

abstrakt:



konkret:

Spezialfall: Existenzabhängige Entities

- Bisher wurde angenommen, dass Entities autonom existieren und innerhalb ihres Entity-Typs durch ihre Schlüsselattribute eindeutig identifiziert werden können.
- In der Realität gibt es jedoch gelegentlich Entities, die im Rahmen des konzeptionellen Modells nicht selbst identifizierbar sind, sondern nur aufgrund ihrer Beziehung zu einem anderen (übergeordneten) Entity. Sie besitzen daher auch keinen eigenen Schlüssel.
- Derartige Entities bezeichnet man als **existenzabhängige Entities** oder **Weak Entities**.
- Beispiel: _____

Komplexität von Beziehungen

- Die Komplexität einer Beziehung gibt an, mit wie vielen anderen Entities ein bestimmtes Entity in einer ganz konkreten Beziehung
  - stehen kann,
  - stehen darf bzw.
  - stehen muss.
- Es gibt mehrere Notationen zur Angabe der Komplexität von Beziehungen. In der Vorlesung werden zwei der gebräuchlichsten Darstellungsformen behandelt und zwar
  - 1:n – Notation nach Chen
  - (min-max) – Notation

Komplexität von binären Beziehungen (Grad = 2)

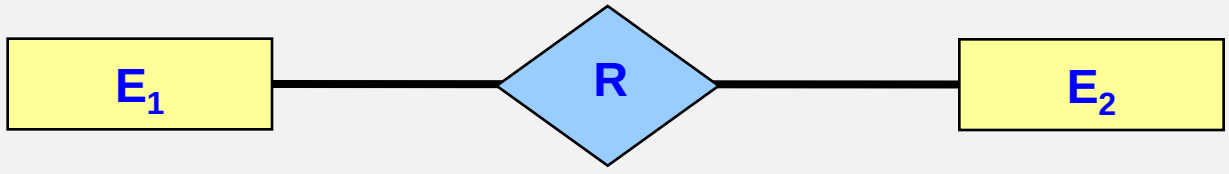
- **1:1-Beziehung**:
  Jedem Entity $e_1 \in E_1$ ist höchstens ein Entity $e_2 \in E_2$ zugeordnet und umgekehrt ist jedem Entity $e_2 \in E_2$ maximal ein Entity $e_1 \in E_1$ zugeordnet. Man beachte, dass es auch Enities geben kann, die in keiner Beziehung zu anderen Enities stehen.

- **1:n-Beziehung**:
  Jedem Entity $e_1 \in E_1$ können beliebig viele Entities $e_2 \in E_2$ zugeordnet sein. Umgekehrt aber kann jedes Entity $e_2 \in E_2$ höchstens zu genau einem Entity $e_1 \in E_1$ in einer Beziehung stehen.

- **n:1-Beziehung**: Umkehrung zu obigem.

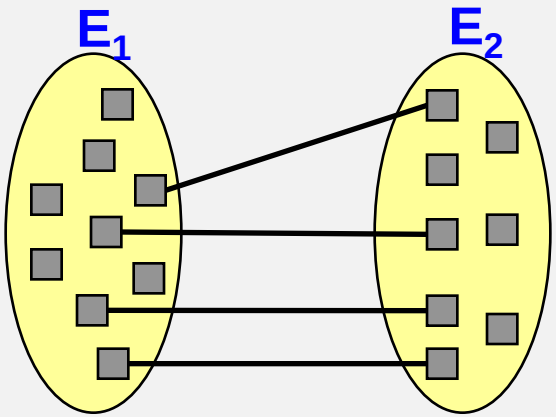
- **n:m-Beziehung**:
  Jedes Entity e_1 aus E_1 kann mit beliebig vielen Entities aus E_2 in Beziehung stehen und umgekehrt.

Veranschaulichung der Komplexität von binären Beziehungen

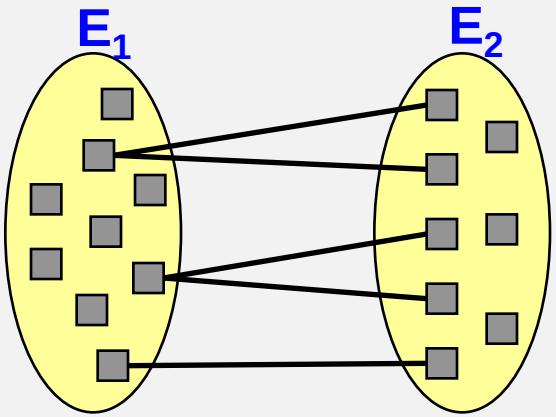


$R \in E_1 \times E_2$

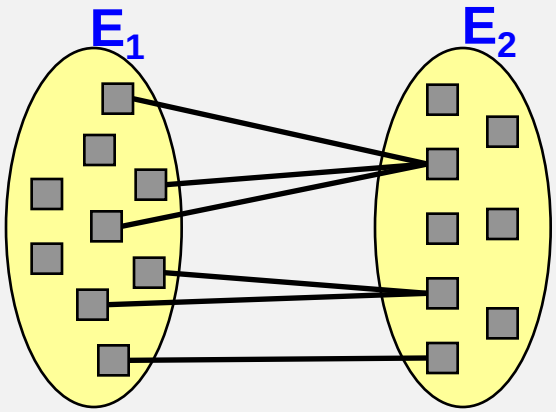
1:1



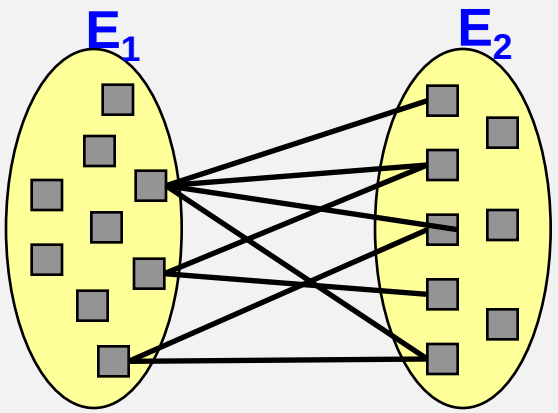
1:n



n:1



n:m



Komplexitätsangaben mit der (min,max)-Notation

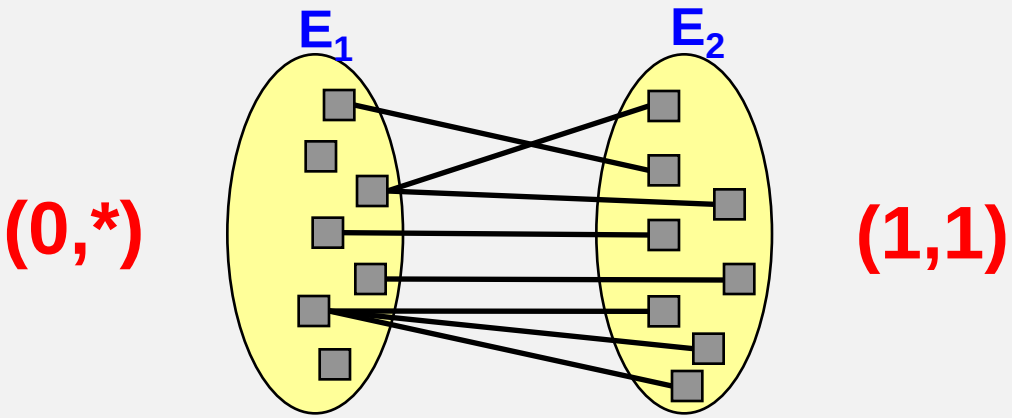
- Die 1:n-Notation hat den Nachteil, dass nicht ersichtlich ist, ob ein Entity überhaupt eine Beziehung hat.
- Eine Erweiterung des E/R-Models von Chen besteht in der sogenannten (min,max)-Notation, bei der die abzubildenden Informationen über die Beziehungen zwischen den Objekten der realen Welt manchmal präziser charakterisiert werden.
- Das Prinzip besteht darin, wann immer möglich, eine präzise Unter- und Obergrenze für die Anzahl der möglichen Beziehungen eines Entities zu anderen Entities anzugeben.

Definition (min,max)-Komplexität

- Wenn ein Entity $e_1 \in E_1$ mindestens a und höchsten b Beziehungen eines Typs haben kann, bezeichnet man dies mit der Notation (a,b), wobei folgende Bedingung erfüllt sein muss:

  $0 \leq a \leq b \leq *$ (Das Symbol „*“ steht für beliebig viele)

- Beispiel



(1,1) → ein Entity vom Typ E_2 hat genau eine Beziehung.

(0,*) → ein Entity vom Typ E_1 kann beliebig viele (also auch keine) Beziehungen haben.

Zusammenhang zwischen (min,max)- und 1:n-Notation

(min,max)-Notation E_1	1:n-Notation	(min,max)-Notation E_2
(0,1) (1,1)	1:1	(0,1) (1,1)
(0,*) (1,*)	1:n	(0,1) (1,1)
(0,*) (1,*)	n:m	(0,*) (1,*)

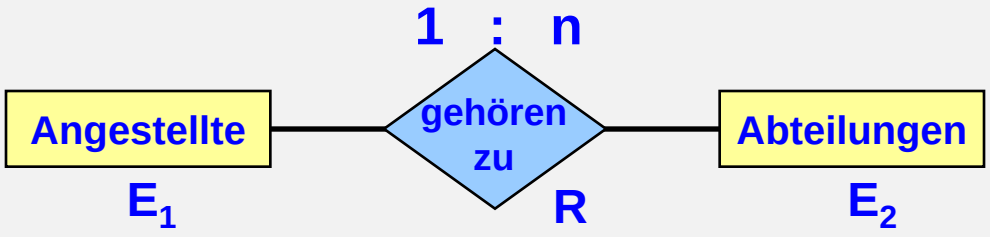
Schlüssel von Beziehungstypen

- Wie schon bei den Entity-Typen kann man auch für Beziehungstypen den Begriff Schlüssel definieren. Besonderheit hierbei ist, dass ein Beziehungstyp zwei Arten von Attributen hat und zwar
  - Schlüssel der beteiligten Entity-Typen und
  - Zusatzattribute

- Für den Schlüssel K eines Beziehungs-Typs R gilt
  $\mathbf{K(R) \subseteq \{K(E_1), K(E_2), \ldots K(E_n), Z_1, Z_2, \ldots, Z_m\}}$
  - K ist identifizierende Attributkombination für R, d. h. unterschiedliche Beziehungen haben auch unterschiedliche Attributwerte bzgl. K
  - Es gibt keine echte Teilmenge $K' \subset K$, für die Eigenschaft (1) gilt

Beispiel für Schlüssel von Beziehungstypen (1)

- Sei R:<E_1,E_2> eine 1:n-Beziehung mit Namen R, z.B.



Ang.	Ang#	Name	Beruf
	4711	Maier	Technikerin
	4750	Weber	Buchhalter
	4912	Schmitt	DB-Designerin

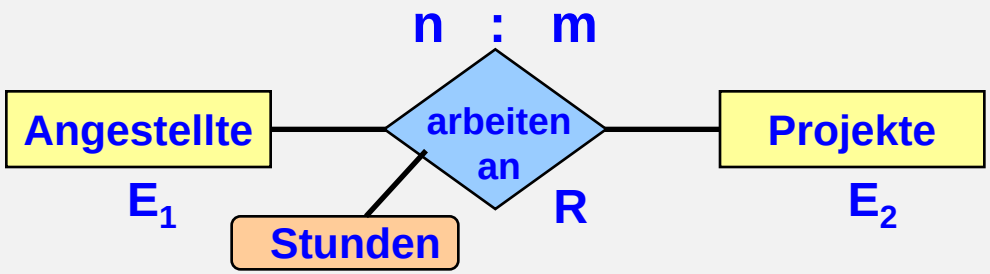
Abt.	Abt#	Bezeichnung
	01	Fibu
	02	Technik
	03	SWE

- Dann ist der Schlüssel von R der Schlüssel von E_2, auch als Fremd-schlüssel bezeichnet

Zugehörigkeit R	Ang#	Abt#
	4711	02
	4750	01

Beispiel für Schlüssel von Beziehungstypen (2)

- Sei R:<E_1,E_2> eine n:m-Beziehung mit Namen R, z.B.



Angestellte	AngNr	Name
	47110	Müller
	47120	Weber

Projekte	PNr	Bez.	Budget
	0815	Webshop	750.000 €
	0816	Neues RZ	980.000 €

- Dann setzt sich der Schlüssel von R aus den Schlüsseln der beteiligten Entity-Typen zusammen. Wieso nicht auch W-Std.?

arbeiten an	AngNr	PNr	W-Std.
	47110	0815	3
	47110	0816	7
	47120	0815	20

Schlüssel von Beziehungstypen mit Grad 3 (1)

- Manchmal ist es schwierig, Kardinalität und Schlüssel bei Beziehungstypen mit Grad > 2 zu ermitteln. Hierzu ein Beispiel:

- Angenommen, es existiert eine Beziehung "Liefermöglichkeit" zwischen den Entity-Typen "Projekt", "Bauteil" und "Lieferant„ mit folgendem Sachverhalt:
  - Es gibt mehrere Lieferanten, die jeweils verschiedene Bauteile liefern können.
  - Ein bestimmtes Bauteil kann auch von unterschiedlichen Lieferanten bezogen werden.
  - Ein Bauteil kann in verschiedenen Projekten verwendet werden.
  - In einem Projekt werden mehrere unterschiedliche Bauteile benötigt, die von verschiedenen Lieferanten geliefert werden können.

Schlüssel von Beziehungstypen mit Grad 3 (2)

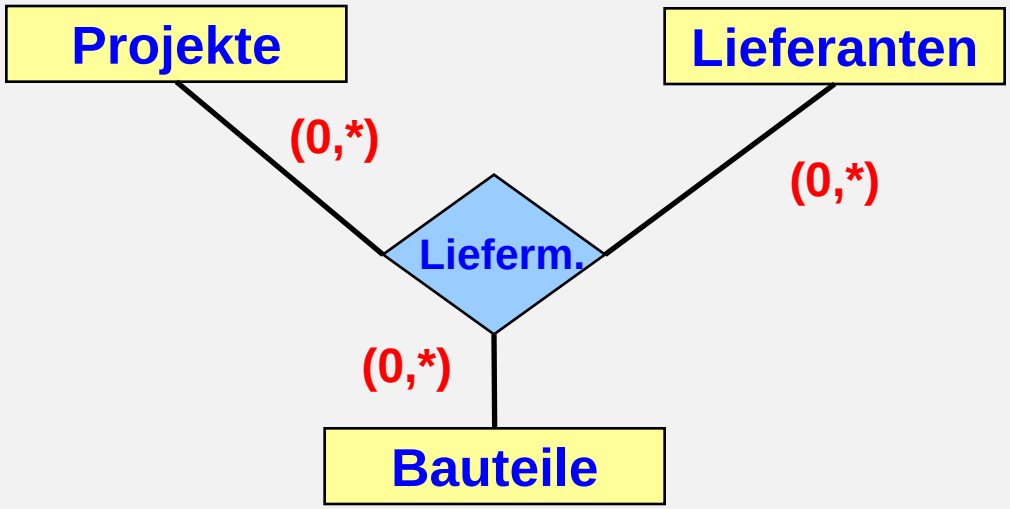
- Beispiel:

Projekt	Bauteil	Lieferant
RZ Frankfurt	Bladecenter	HP
RZ Frankfurt	Switch	HP
RZ Frankfurt	SAN	EMC
RZ Karlsruhe	Bladecenter	IBM
RZ Karlsruhe	Switch	Cisco
RZ Karlsruhe	Bladecenter	HP

- Angenommen es existiert zusätzlich die folgende, durchaus übliche Einschränkung. Was ändert sich jetzt?
  - Ein bestimmtes Bauteil für ein bestimmtes Projekt darf nur jeweils von einem einzigen Lieferanten bezogen werden.
  - Unverändert gilt, dass Bauteile zu beliebigen Projekten gehören und umgekehrt Projekte sich aus beliebigen Bauteilen zusammensetzen.

Schlüssel von Beziehungstypen mit Grad > 2 (3)

- Nachteil bei der Darstellung in der (min,max)-Notation:



- Man sieht, dass bei (min,max)-Notation nicht zwischen n:m:p- und n:m:1 unterschieden werden kann!

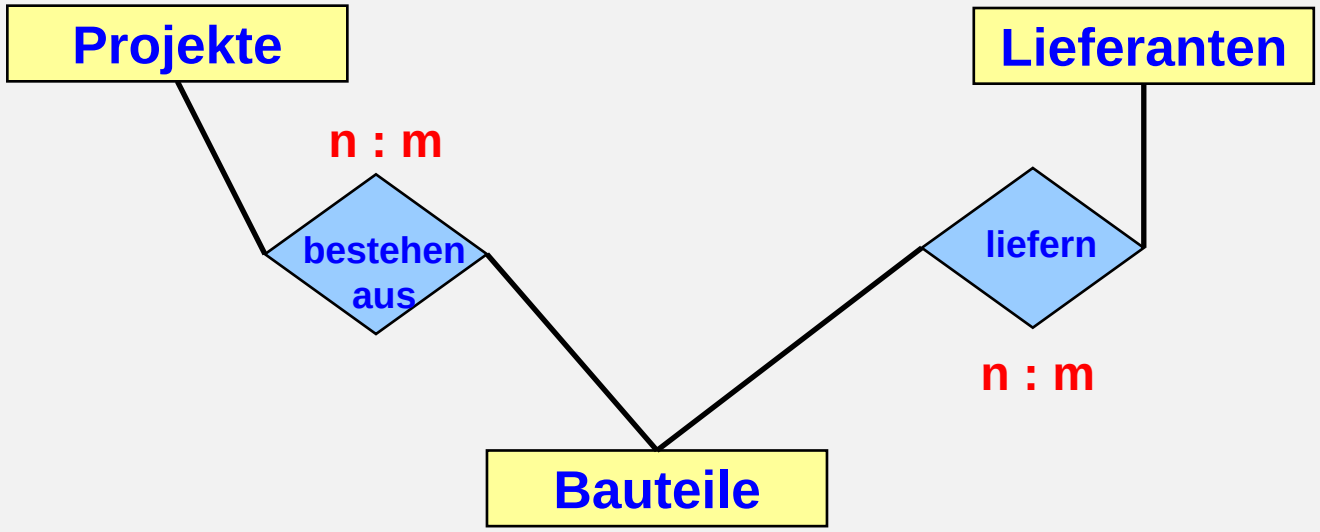
  Anmerkung: Die Angabe (0,*) legt keine Einschränkungen fest und ist somit die Standard-Annahme, wenn keine genaueren Angaben vorliegen.

Datenbankentwurf

2.2 Komplexität von Beziehungen

Frage: Lässt sich dieser Sachverhalt auch durch zwei Beziehungen vom Grad 2 darstellen?

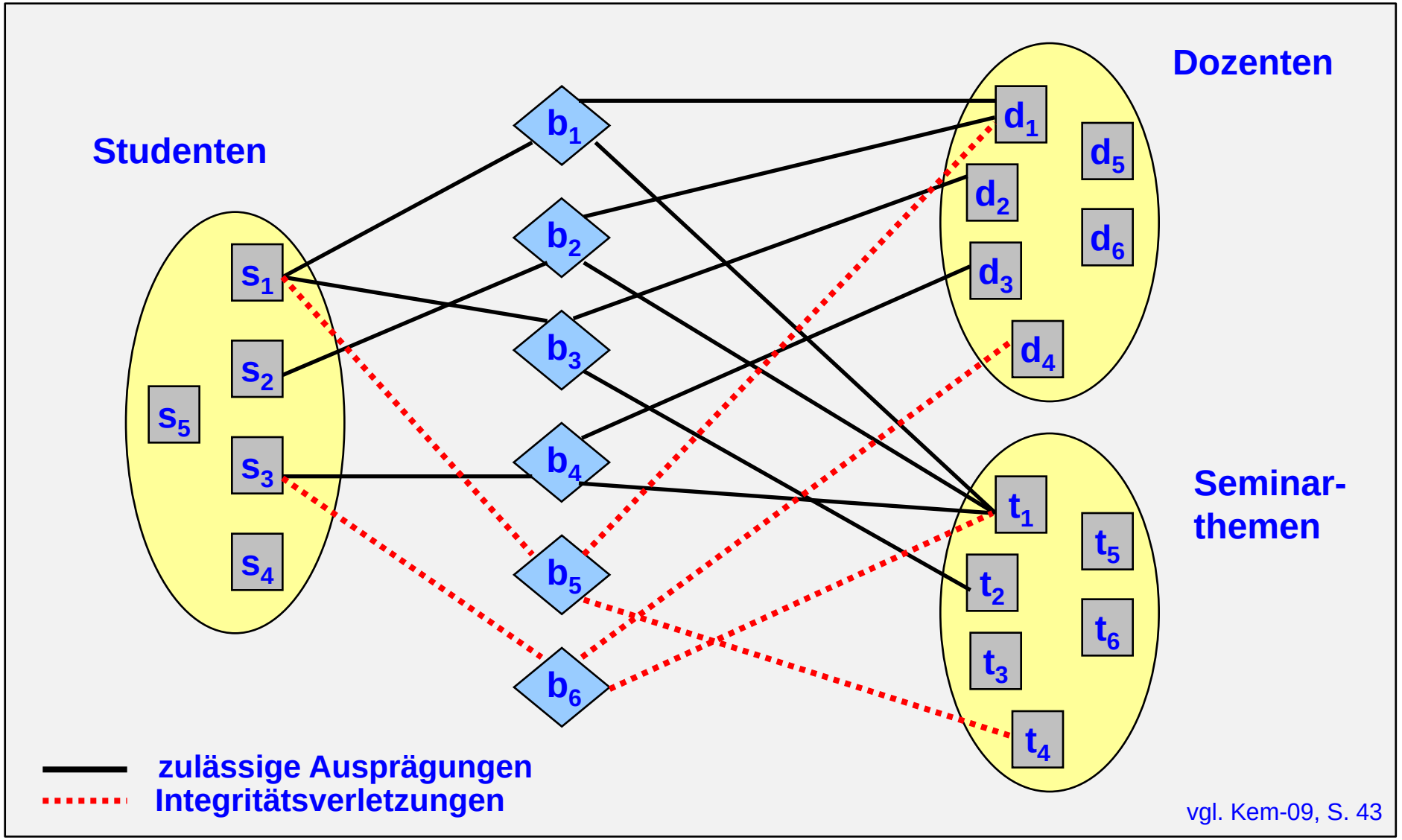
Projekt	Bauteil	Lieferant
RZ Frankfurt	Bladecenter	HP
RZ Frankfurt	Switch	HP
RZ Frankfurt	SAN	EMC
RZ Karlsruhe	Bladecenter	IBM
RZ Karlsruhe	Switch	Cisco
~~RZ Karlsruhe~~	~~Bladecenter~~	~~HP~~



Weiteres Beispiel: Vergabe Seminarthemen (1)

- Angenommen, die Prüfungsordnung der Dualen Hochschule sieht folgende Regelungen vor:
  - Studenten dürfen bei demselben Dozenten nur ein Seminarthema "ableisten" (sinnvoll, damit ein breiteres Spektrum abgedeckt wird)
  - Studenten dürfen dasselbe Seminarthema nur einmal bearbeiten, d.h. sie dürfen nicht bei einem anderen Dozenten ein von ihnen schon einmal bearbeitetes Seminarthema nochmals einreichen.
    (was natürlich ebenfalls eine äußerst sinnvolle Forderung ist :-)
- Es soll aber grundsätzlich möglich sein, dass
  - Dozenten dasselbe Seminarthema "wiederverwenden" können, d.h. ein bestimmtes Thema an mehrere Studenten vergeben können.
  - Ein Seminarthema von mehreren Dozenten vergeben wird – sofern es von unterschiedlichen Studenten bearbeitet wird.

Weiteres Beispiel: Vergabe Seminarthemen (2)



vgl. Kem-09, S. 43

Generalisierung und Spezialisierung (1)

- Um im konzeptionellen Entwurf eine natürlichere und auch übersichtlichere Strukturierung der Entity-Typen zu erzielen, werden diese auf Typebene weiter abstrahiert und zudem das Prinzip der Vererbung eingeführt.

- **Generalisierung**: Gleichartige Entity-Typen werden zu einem übergeordneten Entity-Typ (sog. Obertyp) zusammengefasst.
  - Hierzu werden die gemeinsamen Eigenschaften (Attribute) der Untertypen "herausgezogen" und dem gemeinsamen Obertyp zugeordnet. Die restlichen (spezifischen) Attribute verbeiben bei den Untertypen.

- **Spezialisierung**: Umkehrung der Generalisierung, d.h. ein Entity-Typ wird in unterschiedliche Teilmengen „zerlegt“.
  - Sinnvoll ist dies z.B. wenn nur eine Teilmenge Beziehungen zu einem anderen Entity-Typ hat.

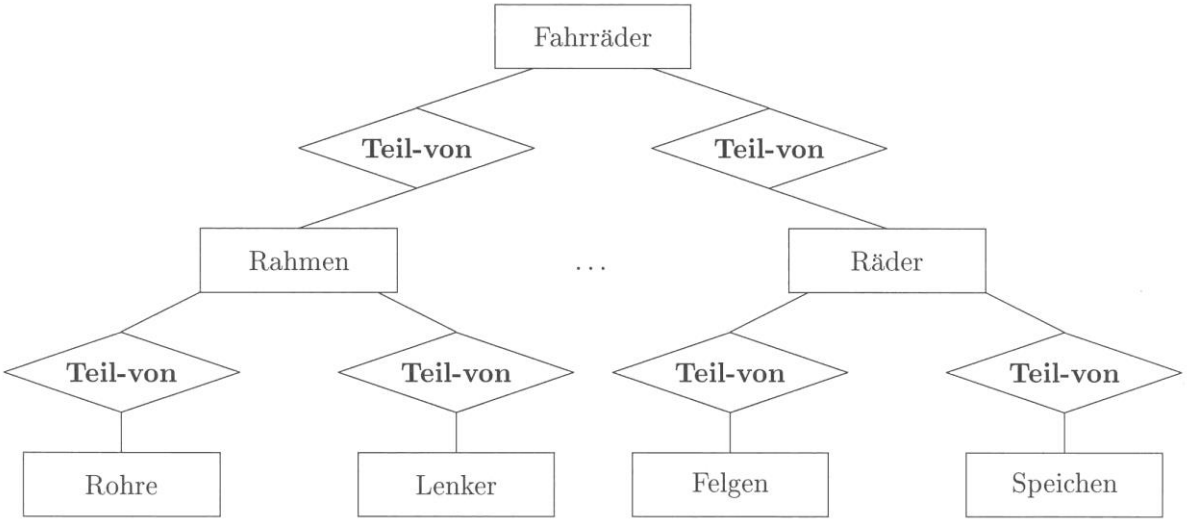
Generalisierung und Spezialisierung (2)

- E ist genau dann eine Generalisierung von $E_1,E_2,...,E_n$, wenn jedes e $\in E^t$ auch Element von einer Menge $E_1^t,...,E_n^t$ ist.
- Zu jedem Subtype E_i von E gilt
  - $E_i^t \in E^t$ zu jedem Zeitpunkt
  - E_i hat denselben Primärschlüssel wie E
  - Der Subtype E_i hat mindestens soviel Attribute wie E; i.a. jedoch mehr
- Spezialfall **disjunkte Spezialisierung**: Die Entity-Mengen E_1, $E_2,...,E_n$ sind paarweise disjunkt, d.h. jedes e $\in E^t$ ist auch Element von genau einer der Mengen $E_1^t,...,E_n^t$.
- Anmerkung: Eine vollständige Spezialisierung ist dann gegeben, wenn die Entitymenge des Obertyps keine "direkten" Entities mehr enthält, d.h. alle Entities auf Untermengen "verteilt" wurden.

Aggregation

- Während man bei der Generalisierung gleichartige Entity-Typen strukturiert, werden bei der **Aggregation** unterschiedliche Entity-Typen zusammengefasst und zueinander in Beziehung gesetzt.

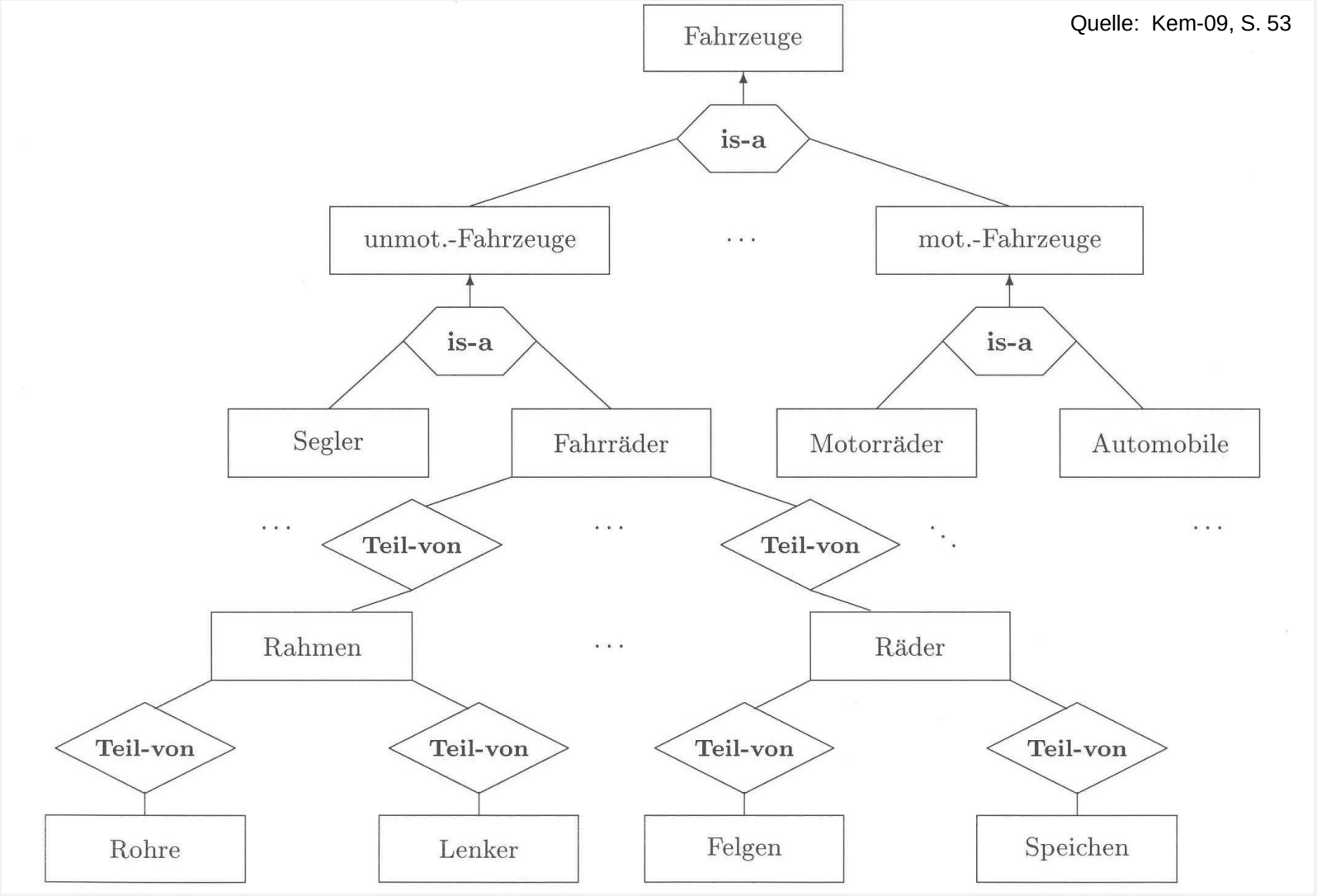
- Insofern ist die Aggregation ein besonderer Beziehungstyp, der einem übergeordneten Entity-Typ mehrere untergeordnete Entity-Typen zuordnet.
  Dies wird auch als **(part-of)**-Beziehung bezeichnet im Gegensatz zur **(is-a)**-Beziehung.



Quelle: Kem-09, S. 52

Kombination von Generalisierung und Aggregation

Quelle: Kem-09, S. 53



Ein EER-Modell (1)

- Nachfolgend wird (auszugsweise) eine (von mehreren) Erweiterung des klassischen E/R-Modells vorgestellt. Diese basiert überwiegend auf den Veröffentlichungen von Hohenstein und Gogolla und entstand in den Jahren 1993/94 im Rahmen eines Projekts zur Entwicklung einer neuen Datenbankentwurfsumgebung.
- Unverändert übernommen wurden dabei
  - Entity-Typen bzw. Entities
  - Beziehungstypen bzw. Beziehungen
  - Attribute
- Neu sind
  - Erweiterung der Wertebereiche für Attribute.
  - Die "is a"-Beziehung wird durch einen sog. Typenkonstruktor ersetzt.
  - Erweiterung des Schlüsselkonzepts
  - Abhängige Entity-Typen fallen komplett weg und werden durch das neue Schlüsselkonzept modelliert.

Erweiterungen bei Attributen

- Im klassischen E/R-Modell sind nur Attribute möglich, deren Werte Standarddatentypen wie *String* oder *Integer* annehmen können.
- Eine naheliegende Erweiterung besteht darin, auch strukturierte Attributwerte (vergleichbar dem Datentyp *Record* bei prozeduralen Programmiersprachen) zuzulassen.
- Eine zweite Erweiterung stellen die mehrwertigen Attribute dar. D.h. es wird angenommen, dass konkrete Entities auch mehrere unterschiedliche Werte bei diesem Attribut haben können.
- Weiterhin kann man noch abgeleitete Attribute zulassen. Dies sind Attribute, deren Werte nicht in der Datenbank abgespeichert werden müssen, sondern aufgrund einer Herleitungsregel bei einer Anfrage an die DB berechnet bzw. bestimmt werden können.

Vorgehensweise beim Aufbau von E/R-Diagrammen (1)

1. Man beginne mit den leicht erkennbaren natürlichen Objekten wie beispielsweise Personen und konkreten Gegenständen und fasse diese anhand gemeinsamer Merkmale zu Entity-Typen zusammen.
2. Man sammle zu jedem Entity-Typ alle relevanten Attribute und definiere deren Wertebereiche.
3. Man bestimme den Schlüssel der einzelnen Entity-Typen.
   - Alle Attribute mit 1:1 Beziehung sind Schlüsselkandidaten!
   - Gibt es keine solchen Attribute, so wähle man eine Attributkombination mit Schlüsseleigenschaften oder führe ein zusätzliches Attribut zur Bildung des Schlüssels ein
4. Stellt man fest, dass zwischen den Werten eines Attributes und den Entities eines Entity-Typs eine n:m Beziehung besteht, dann
   - fasse man dieses Attribut als neuen Entity-Typ auf oder
   - führe zusätzliche Attribute ein

Vorgehensweise beim Aufbau von E/R-Diagrammen (2)

5. Stellt man fest, dass ein vermeintliches Attribut selbst Attribute hat, so ist dieses Attribut als Entity-Typ zu modellieren.

6. Bilden bzw. Ergänzen der Beziehungstypen

7. Man gebe die Komplexitäten in 1:n- und (min,max)-Notation an.

8. Man überprüfe das Schema auf Redundanzen und auf Normalfor-men. Z.B. können Objekt- und Beziehungstypen mit demselben Schlüssel zusammengefasst werden.

- **Vorschläge zur grafischen Darstellung**
  - Beziehungstypen durchnummerieren und in einer gesonderten Liste aufführen.
  - Attribute bei großen ER-Diagrammen weglassen oder nur Schlüssel angeben.