Datenbanken
Wintersemester 04/05
Prof. Dr. W. May

1. Übungsblatt: ER-Modell und Relationales Modell

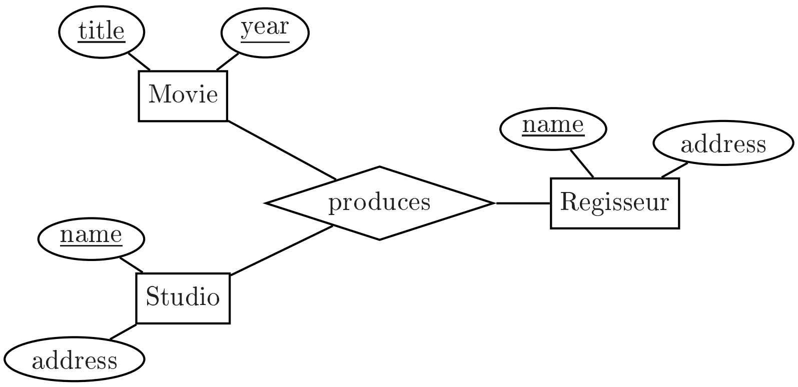
Besprechung am 4.11. Am 28.10. findet keine Übung statt.

Aufgabe 1 (ER-Modell: Film) Geben Sie ein ER-Modell für den folgenden Sachverhalt an: Filme werden in Filmstudios von Regisseuren gedreht. Filmstudios gehören einem Besitzer. In Filmen treten Schauspieler auf. Schauspieler erhalten eine Gage für jeden ihrer Verträge.

Entwickeln Sie zuerst ein einfaches Modell, und überlegen Sie dann, wie und ob Sie das Modell ergänzen könnten, um z.B. zu modellieren, dass sowohl Schauspieler als auch Regisseure und Besitzer von Filmstudios Personen sind, und manche Personen auch im selben Film oder in verschiedenen Filmen in mehreren dieser Rollen auftreten.

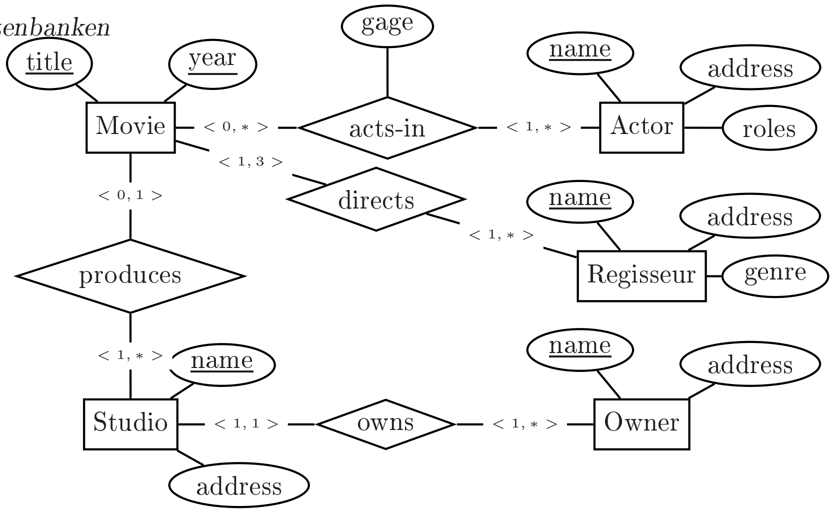
Nur die (bzw. eine) korrekte Lösung zu zeigen, wäre zu einfach und auch didaktisch nicht sinnvoll. Aus diesem Grund werden hier verschiedene Wege und Holzwege diskutiert.

Erster Ansatz: Man betrachtet den Satz “Filme werden in Filmstudios von Regisseuren gedreht.” Naheliegend ist hier eine dreistellige Beziehung:



[Präsentation: Umsetzung in Relationen.] Man sieht an der Relation, bzw. an der anzustellenden Kardinalitätsbetrachtung, dass diese Modellierung nicht davor bewahrt, für einen Film, der von mehreren Regisseuren gemeinsam gedreht wird, auch unterschiedliche Studios zu speichern. Also sollte man diese Modellierung so nicht wählen.

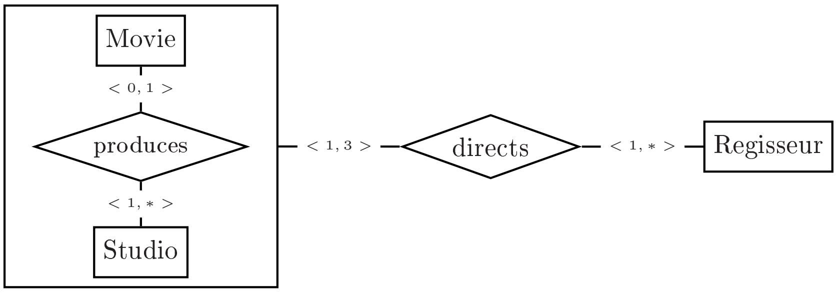
Naheliegend ist, die Beziehung in zwei zweispaltige Beziehungen (*Movie-Studio*) und (*Movie-Regisseur*) aufzulösen, womit man insgesamt die folgende “Musterlösung” erreicht:



Roles ist ein spezifisches Attribut von Schauspielern, das angibt, welche Arten von Rolle ein Schauspieler spielen kann (Erweiterung: die Beziehung *acts-in* ebenfalls *roles* zu erweitern, das angibt, welche Art von Rolle ein Schauspieler in diesem Film spielt). *Genre* ist ein spezifisches Attribute von Regisseuren, das angibt, welche Art von Filmen ein Regisseur dreht (auch hier kann man überlegen, *genre* als Attribut des Films zu nehmen).

Alternative Modellierung der linken Seite.

Eine mögliche Alternative ist, die dreistellige Beziehung *directs* aufzuspalten, indem man die Beziehung *produces* zwischen einem Studio und einem Film betrachtet, diese aggregiert (Produktion), und das Aggregat in eine *directs*-Beziehung mit Regisseuren stellt. *Owner* stehen weiterhin in Beziehung mit ihrem Studio, Schauspieler kann man wahlweise in Beziehung mit dem Film oder der Produktion stellen.



Hinweis: man erhält dasselbe relationale Modell, wenn man an der richtigen Stelle einbezieht, dass die 1:n-Beziehung zwischen Film und Studio dazu führt, dass der Schlüssel von *Production* nur (*title,year*) ist.

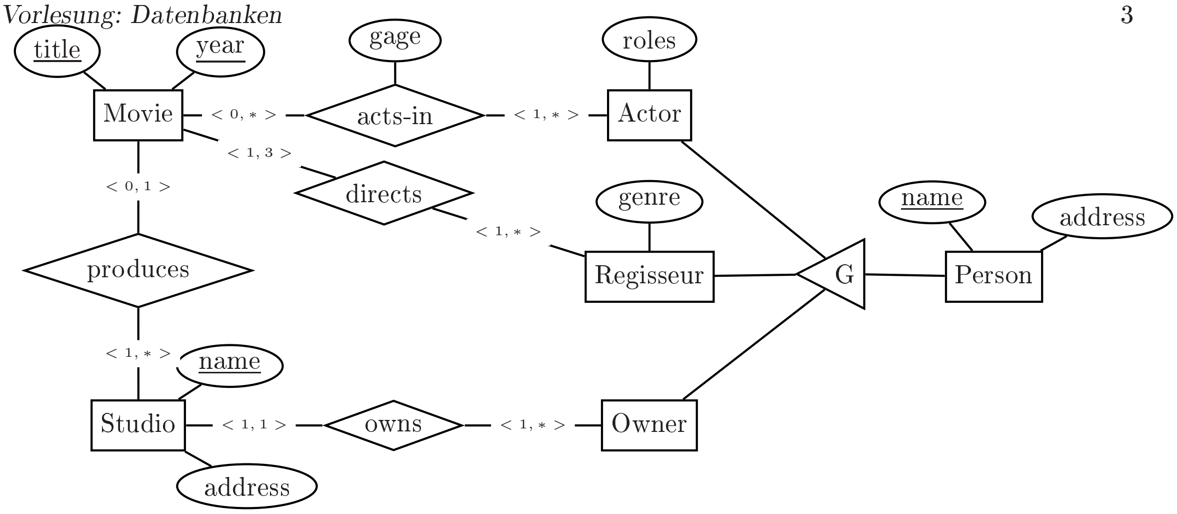
Hinweis: eine Aggregation (*Film, Regisseur*) und eine Beziehung des Aggregates zu *Studio* ist keine geeignete Modellierung, da dann für Filme, die von mehreren Regisseuren zusammen gedreht würden, auch verschiedene Studios angegeben werden könnten.

Modellierung mit Generalisierung "Person".

Betrachtet man bei der obigen Modellierung die Umsetzung ins relationale Modell, so stellt man fest, Information zu Personen, die z.B. sowohl als Regisseur als auch als Schauspieler gespeichert sind, *redundant* gehalten wird. Dies kostet nicht nur Platz, sondern kann auch zu inkonsistenten Datenbankzuständen führen:

Actor(AS, Hollywood Drive 42 - LosAngeles) und Regisseur(AS, Graben 1 - Wien).

Es ist daher sinnvoll, einen Entitätstyp *Person* als Generalisierung dieser Typen einzuführen:



Generalisierung bedeutet hier, dass

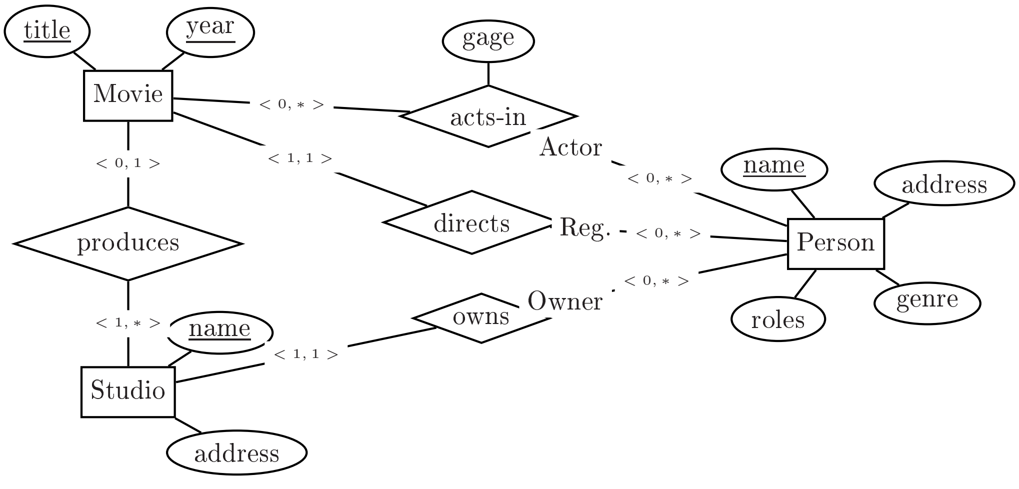
$$Persons = Actors + Regisseurs + Owners$$

ist. Würde man stattdessen eine Spezialisierung wählen, könnte man auch Personen haben, die nicht in eine dieser spezielleren Klassen fallen.

Bei einer Umsetzung ins relationale Modell erhält man je eine Tabelle für *Person(Name,Address)*, *Actor(Name,role)* (ggf. mehrwertig), *Regisseur(Name,genre)*, die jeweils den Primärschlüssel *Name* einer Person referenzieren.

Modellierung nur mit "Person" und Rollen.

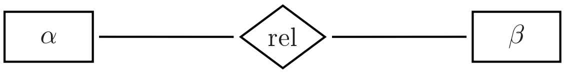
Als dritte Möglichkeit könnte man nur Personen modellieren und ihre Rollen (in der Anwendungswelt) durch Rollen(bezeichner im ER-Modell) darstellen:



Die Attribute *roles* und *genre* müssen damit alle bei Person angesiedelt sein. Damit ist nicht klar, dass z.B. nur Schauspieler das Attribut *roles* besitzen. Außerdem sind potentiell viele Nullwerte in der Tabelle enthalten.

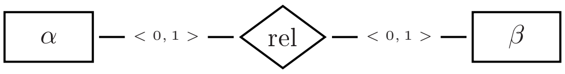
Aufgabe 2 (Komplexitäten) Betrachten Sie einen binären Beziehungstyp und die Komplexitäten $(0, 1), (1, *)$. Weisen Sie die Beziehungskomplexitäten in allen vier möglichen Weisen zu und geben Sie zu jeder Variante jeweils einen nichttrivialen Zustand an, der die Komplexitäten erfüllt, bzw. einen, der sie verletzt.

Ein binärer Beziehungstyp:



Entitätsmenge Typ1: $\{A, B, C, D, E, F\}$
Entitätsmenge Typ2: $\{1, 2, 3, 4\}$

1. Fall:



Jedes α steht mit maximal einem β in Beziehung und umgekehrt.

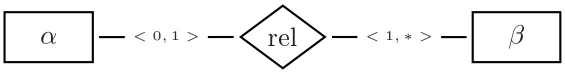
Zulässiger Zustand:

A	2
B	4
F	1

Unzulässiger Zustand:

A	2
A	4
F	1

2. Fall:



Jedes α steht mit maximal einem β in Beziehung, aber jedes β muss mindestens mit einem α in Beziehung stehen. und umgekehrt.

Zulässiger Zustand:

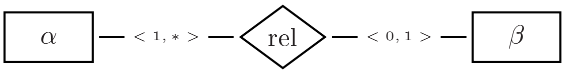
A	2
B	4
C	3
E	3
F	1

Unzulässige Zustände:

A	2
B	4
B	3
F	1

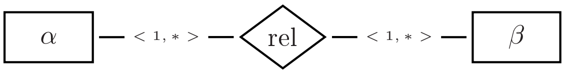
A	2
B	4
F	1

3. Fall:



Anforderungen symmetrisch; mit den gegebenen Daten aber nicht erfüllbar: jedes α muss mit mindestens einem β in Beziehung stehen, aber jedes β darf nur mit maximal einem α in Beziehung stehen.

4. Fall:



Jedes α muss mindestens mit einem β in Beziehung stehen und umgekehrt.

Zulässiger Zustand:

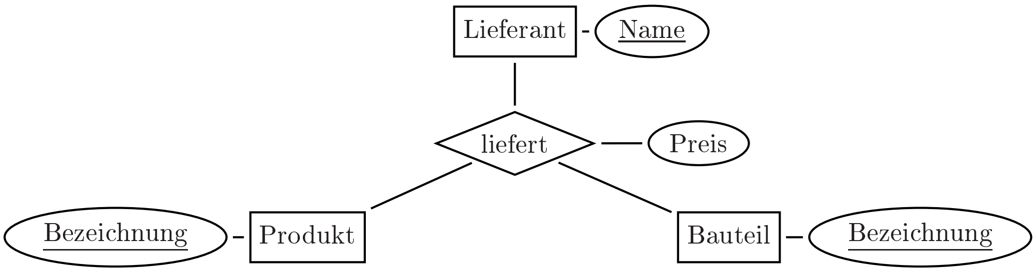
A	2
B	4
C	3
D	2
D	4
E	3
F	1

Unzulässiger Zustand:

A	2
B	4
B	3
F	1

Aufgabe 3 (Dreistellige Beziehungen (Lieferant, Produkt, Bauteil)) Gegeben sei eine dreistellige Beziehung zwischen den Entitätstypen Lieferant, Produkt und Bauteil (*Firmen liefern Bauteile für Produkte*).

a) Geben Sie ein geeignetes ER-Modell an.



b) Läßt sich dieser Sachverhalt mit ausschliesslich binären Beziehungen darstellen?

1) Zerlegung in 2 zweistellige Beziehungen:

"Lieferant L liefert Bauteil B, und Produkt P benötigt Bauteil B" bedeutet aber nicht notwendigerweise, dass P dieses Bauteil auch von L geliefert bekommt:

liefert		
L	P	B
Bosch	Golf	Einspritzpumpe
Bosch	Golf	Navigationssystem
Bosch	Corolla	Einspritzpumpe
Sony	Corolla	Navigationssystem

zerlegt in

LB	
L	B
Bosch	Einspritzpumpe
Bosch	Navigationssystem
Sony	Navigationssystem

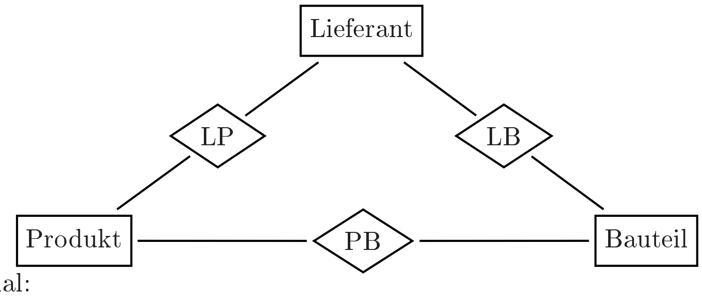
und

PB	
P	B
Golf	Einspritzpumpe
Golf	Navigationssystem
Corolla	Einspritzpumpe
Corolla	Navigationssystem

Bei Zusammenfassung erhält man auch die Tupel (Bosch, Navigationssystem, Corolla) und (Sony, Navigationssystem, Golf), die nicht in der ursprünglichen Beziehung vorhanden waren.

Unter der Annahme, dass der Preis nur vom Lieferanten und dem Bauteil abhängt (also nicht unterschiedliche Produkte/Kunden unterschiedliche Preise ausmachen können), kann er als Attribut zu *liefert* genommen werden. Anderenfalls hat man hier ein Problem, das ebenfalls darauf hindeutet, dass die Zerlegung nicht klappt.

2) Zerlegung in 3 zweistellige Beziehungen:



Zerlegung diesmal:

LB	
L	B
Bosch	Einspritzpumpe
Bosch	Navigationssystem
Sony	Navigationssystem

und

PB	
P	B
Golf	Einspritzpumpe
Golf	Navigationssystem
Corolla	Einspritzpumpe
Corolla	Navigationssystem

und

LP	
L	P
Siemens	Golf
Siemens	Corolla
Sony	Corolla

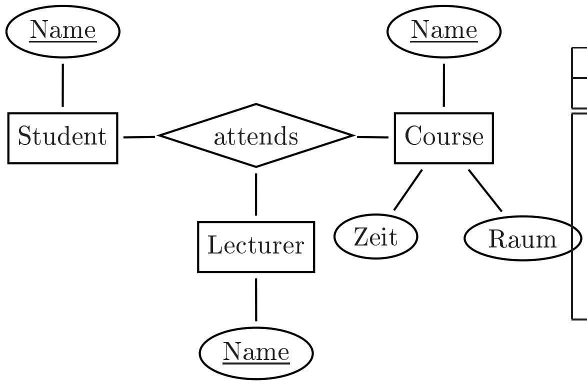
Fasst man das zusammen, erhält man auch ein Tupel (Siemens, Corolla, Navigationssystem) das nicht in der ursprünglichen Beziehung vorhanden war.

Wieder kann unter der obigen Annahme der Preis als Attribut zu *liefert* genommen werden. Anderenfalls hat man hier wieder ein Problem, das darauf hindeutet, dass die Zerlegung nicht klappt.

---

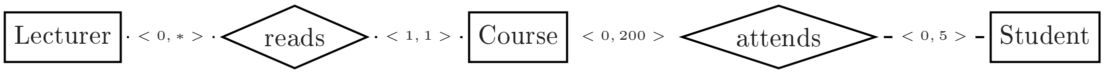
c) Betrachten Sie nun dreistellige Beziehungen wieder allgemein. Gibt es Situationen, in denen eine Darstellung durch zwei binäre Beziehungstypen möglich ist? Können diese Situationen exakt durch Beziehungskomplexitäten definiert werden?

Nehmen wir eine andere dreistellige Beziehung: Studenten hören Vorlesungen bei Dozenten. Vorlesungen findet zu einer bestimmten Zeit in einem bestimmten Raum statt.



attends		
Student	Course	Lecturer
S1	Databases	WM
S2	Databases	WM
S2	Telematics	DH
S3	Telematics	DH
S3	SQL	WM
S4	SQL	WM
S1	SQL	WM

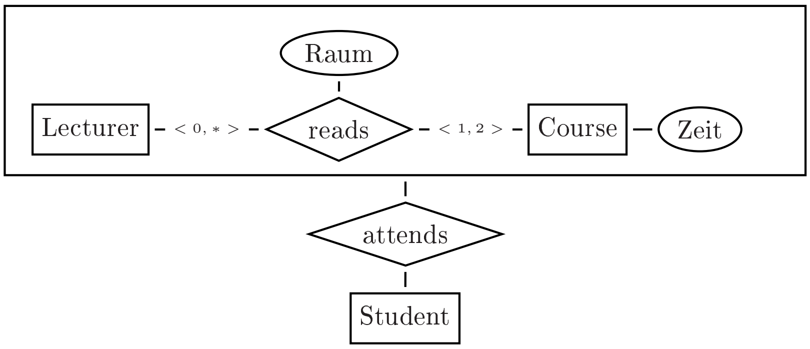
Es gibt hier eine "funktionale Abhängigkeit" *Course* → *Lecturer* (ein Dozent kann trotzdem mehrere Vorlesungen halten). Man kann sinnvoll aufteilen in "reads" und "attends":



Hinweise:

- Im vorliegenden Fall ist die zerlegte Modellierung die klar bessere, da die Kardinalitäten der *reads*-Beziehung die funktionale Abhängigkeit ausdrücken. Bei der dreistelligen Relation wäre ein Datenbankzustand, der sowohl (S1, Info1, DH) und (S3, InfoI, WM) enthält, erlaubt.
- die Kardinalitäten können bei der dreistelligen Beziehung nicht geeignet untergebracht werden.
- Dies klappt nicht mehr, wenn grosse Vorlesungen parallel bei zwei Dozenten (selbe Zeit, unterschiedliche Räume) angeboten werden (WM, Info I) und (DH, Info I).
- Wie könnte man in diesem Fall modellieren, dass ein Student eine solche Vorlesung nur bei einem Dozenten hört?

  Aggregierung der *reads*-Beziehung zwischen Dozent und Vorlesung:

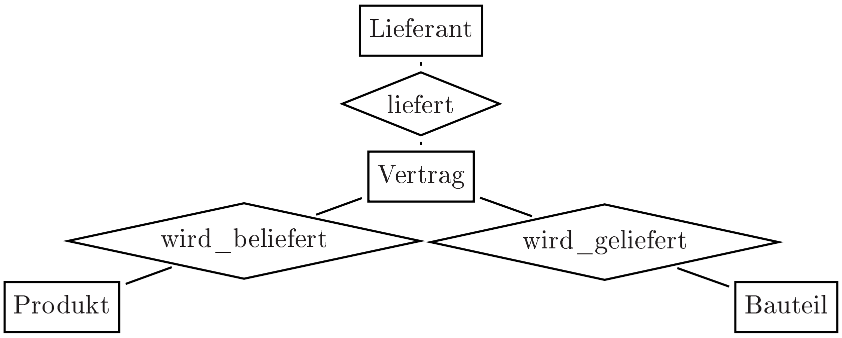


  * die Anforderung, dass ein Student eine Vorlesung nur bei einem Dozenten hört, kann so nicht beschrieben werden. Dies muss zusätzlich durch Text angegeben werden.
  * wenn man auf die ursprüngliche dreistellige Beziehung ausweichen würde, müsste die *Zeit* als Attribut zu *attends* genommen werden. Dann wären allerdings Zustände der Art attends(S1, Info1, WM, Fr.14:00) und (S2, Info1, WM, Do.16:00) in der DB möglich.

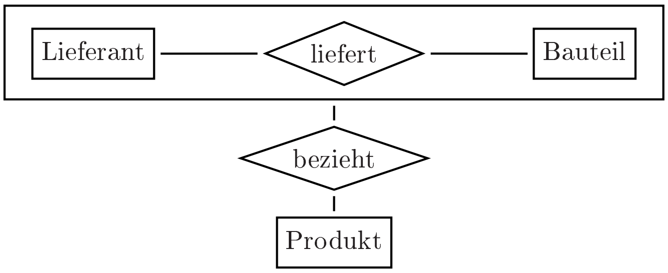
---

c) Kann man dennoch dreistellige Beziehungen generell (unter Verwendung weiterer Hilfskonstrukte) durch zweistellige Beziehungen ersetzen?

Ja. Jede Instanz der dreistelligen Beziehung wird als Entität betrachtet, z.B.



Auch hier kann man eine Aggregation von Lieferant und Bauteil bilden:



Der Preis kann hier -je nach Situation- als Attribut zu *liefert* oder zu *bezieht* hinzugenommen werden.

Nimmt man ihn zu *bezieht*, so hat man dieselbe Semantik wie bei der ursprünglichen dreistelligen Beziehung. Gehört er zu *liefert*, so hat man mit der Aggregation eine stärkere Modellierung.

Man kann die Diskussion weiter treiben, wenn man modellieren will, dass ein Produkt eine bestimmte Menge eines Bauteils (z.B. (Golf, Rad, 4) benötigt. Hier wäre dann eine Aggregation von Produkt mit Bauteil sinnvoll, die in Beziehung zu einem Lieferanten steht. Dann kann jedoch der Preis wieder für unterschiedliche Produkte auch unterschiedlich sein. Ansonsten muss man zusätzliche Bedingungen textuell festhalten.

e) Vergleichen Sie Vor- und Nachteile der verschiedenen Zerlegungen? Lassen sich die verschiedenen Integritätsbedingungen mittels Komplexitätsgraden ausdrücken?

Vorteil der möglichen Zerlegung in zwei zweistellige Beziehungen:

- Redundanzvermeidung und dadurch weniger Fehlermöglichkeiten. So müßte in einer Dreierbeziehung im Fall c) zu jedem Student-Vorlesung-Dozent-Eintrag der Dozent aufgeführt werden (Fehlerquelle bei Updates!).
  Dieser Aspekt wird gegen Ende der Vorlesung (Entwurfstheorie) nochmal behandelt.
- In einer Dreierbeziehung lassen sich die Integritätsbedingungen nicht mittels Komplexitätsgraden darstellen.

Zerlegung mit Hilfs-Entitätstyp: keine Vorteile in der Ausdruckskraft. Bei der Umsetzung in das relationale Modell wird man auch sehen, dass beide Varianten dasselbe relationale Modell erzeugen.

Man muss sehr genau die Semantik der Anwendung untersuchen, um zu entscheiden, mit welcher Modellierung man sie am genauesten trifft.

Aufgabe 4 (Umsetzung in das relationale Modell: Film) In Aufgabe 1 haben Sie ein ER-Modell für eine kleine Filmdatenbank erstellt. Transformieren Sie dieses in ein relationales Modell.

Einfachste Modellierung

Annahme: roles wird als String als Kommaliste abgelegt

Movie: (title, year)

Studio: (name, address)
Actor: (name, address, roles)
Regisseur: (name, address, genre)
Owner: (name, address)

produces: (studio, title, year)
acts-in: (actor, title, year, gage)
directs: (regisseur, title, year)
owns: (owner, studio)

Alternative, um zu einem Schauspieler mehrere Rollencharakterisierungen ablegen zu können:

Actor: (name, address)
ActorRoles: (name, role)

Alternativen

Da ein Film jeweils nur in einem Studio gedreht werden kann (siehe Beziehungskomplexitäten im ER-Diagramm), kann man die *produces*-Beziehung in *Movie* mit hineinnehmen:

Movie: (title, year, studio)

Modelliert man Filmproduktion als Aggregation aus *Movie, Studio*, erhält man dafür genau die Relation

production: (title, year, studio)

Die Produktion steht nun in Beziehung zu Regisseuren, womit man hierfür die Relation

directs: (regisseur, title, year)

erhält.

Generalisierung als Personen

... verschiedene Möglichkeiten.

- Personen mit Name und Adresse, roles und genre jeweils in Relationen Actor bzw. Regisseur die als Fremdschlüssel den Namen einer Person haben. Die Fremdschlüsselbeziehungen der Relationen zu den Beziehungstypen referenzieren die Relationen Actor, Regisseur, Owner:
  Movie: (title, year)
  Studio: (name, address)
  Person: (name, address)
  Actor: (name, roles)
  Regisseur: (name, genre)
  Owner: (name)

  produces: (studio, title, year)
  acts-in: (actor, title, year, gage)
  directs: (regisseur, title, year)
  owns: (owner, studio)

  Actor.name→*Person.name* (und analog)
  acts.name→*Actor.name*
  Aufgrund der detaillierten Modellierung sind die Konsistenzbedingungen an die Datenbank sehr scharf.
- Die vier Relationen *Person*, *Actor*, *Regisseur*, *Owner* kosten relativ viel Platz, weil die Namen mehrfach abgelegt sind (als *Person(AS, LosAngeles)*, *Actor(AS, hero)* und *Regisseur(AS, action)*. Man kann sie einsparen, wenn man stattdessen *roles* und *genre* als Attribute zu *Person* nimmt und mit Nullwerten auffüllt (z.B. *Person(AS, LosAngeles, hero, action, NULL)*). Dies entspricht auch dem ER-Diagramm, wo nur der Entitätstyp *Person* verwendet wird, und mit Rollenbezeichnungen gearbeitet wird:

Movie: (title, year)
Studio: (name, address)
Person: (name, address, roles, genre)

produces: (studio, title)
acts-in: (person as actor, title, year, gage)
directs: (person as regisseur, title, year)
owns: (person as owner, studio)

acts.name→Person.name etc.

Man verliert die Integritätsbedingungen, dass nur Schauspieler das Attribut *roles* haben sowie dass die *acts*-Beziehung nur mit Schauspielern besteht (und analoges).

- Nun hat man ziemlich viele Nullwerte in *roles*, *genre*. Die meisten Personen werden Schauspieler sein. Man kann also z.B. das Attribut *roles* bei Person lassen, und *genre* wieder in separate Relationen für Regisseure (mit nur relativ wenigen Personen) legen:

  Movie: (title, year)
  Studio: (name, address)
  Person: (name, address, roles)
  Regisseur: (name, genre)
  Owner: (name)

  produces: (studio, title, year)
  acts-in: (person, title, year, gage)
  directs: (regisseur, title, year)
  owns: (owner, studio)

Aufgabe 5 (Umsetzung in das relationale Modell: Dreistellige Relationen) In Aufgabe 3 haben Sie mehrere ER-Modelle für dreistellige Beziehungen diskutiert. Transformieren Sie die sinnvollen Modelle in relationale Modelle.

Erstes Beispiel als dreistellige Beziehung:

Lieferant: (Name, Adresse)
Bauteil: (Bezeichnung)
Produkt: (Name, Verkaufspreis)
liefert: (Lieferant, Bauteil, Produkt, Preis)

Fremdschlüssel:
liefert.Lieferant→Lieferant.Name
liefert.Produkt→Produkt.Name
liefert.Bauteil→Bauteil.Bezeichnung

Zweites Beispiel (funktionale Abhängigkeit):

Lecturer: (Name, Address)
Course: (Name, CourseNo)
Student: (Name, Addresse, StudentNo)
reads: (Lecturer, CourseNo)
attends: (StudentNo, CourseNo)

Einführung eines zusätzlichen Entitätstyps:

Lieferant: (Name, Adresse)
Bauteil: (Bezeichnung)
Produkt: (Name, Verkaufspreis)
Vertrag: (Lieferant, Bauteil, Produkt)

Bei der Umsetzung der zusaetzlichen Beziehungstypen muss jeweils der gesamte Schlüssel der Relation *Vertrag* als Fremdschlüssel aufgenommen werden. Als "automatische" Umsetzung ergibt sich damit:

liefert: (Lieferant, Lieferant, Bauteil, Produkt)
wird_geliefert(Bauteil, Lieferant, Bauteil, Produkt, Preis)
wird_beliefert(Projekt, Lieferant, Bauteil, Produkt)

Man sieht sofort, dass hier jeweils ein Attribut redundant ist, also gestrichen werden kann. Ausserdem sind alle diese Beziehungen 1:n-Beziehungen zwischen einem Vertrag und Lieferanten, Bauteilen bzw. Produkten. In der Vorlesung wurde gezeigt (Country/Capital), dass in diesem Fall die Beziehung in die auf der "1"-Seite stehende Relation (also *Vertrag*) mit hineingezogen werden kann. Damit ergibt sich in diesem Fall nicht viel neues für die *Vertrag*-Relation (da die Schlüssel der auf der "n"-Seite stehenden Entitätstypen *Lieferant*, *Bauteil* und *Produkt* bereits in *Vertrag* enthalten sind). Einzig das Attribut *Preis* kommt hinzu:

Vertrag: (Lieferant, Bauteil, Produkt, Preis)

Man erreicht dasselbe relationale Modell wie mit der ursprünglichen dreistelligen Beziehung (die Beziehung wird durch die Aufnahme der Fremdschlüssel praktisch in dieselbe Relation umgesetzt, wie der zusätzliche Entitätstyp).

Einführung einer Aggregation:

Als vierte Möglichkeit wurde eine Aggregation der "liefert"-Beziehung zwischen einem Lieferanten und einem Bauteil zu einem gegebenen Preis besprochen. Hierfür erhält man die folgenden Relationen:

Lieferant: (Name, Adresse)
Bauteil: (Bezeichnung)
Produkt: (Name, Verkaufspreis)
liefert: (Lieferant, Bauteil, Preis)
bezieht: (Lieferant, Bauteil, Produkt)

Die *bezieht*-Relation übernimmt dabei die Attribute *Lieferant* und *Bauteil* als Fremdschlüssel zur Referenzierung auf einen Eintrag in *liefert*. Wichtig ist hier, dass man eine andere referentielle Integritätsbedingung als bei der Modellierung als dreistellige Relation erhält, nämlich

bezieht.(Lieferant,Bauteil)→liefert(Lieferant,Bauteil)
bezieht.Produkt→Produkt.Name