

# Datenbanksysteme

Relationales Modell

Jan Haase

2024

Abschnitt 4

---

# Themenübersicht

- Warum Datenbanken?
- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle

## **Relationales Datenbankmodell**

- Normalisierung
- Arbeiten mit relationalen Datenbanken

# Themenübersicht

- Warum Datenbanken?
- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
  - ➔ **Grundlagen**
    - Transformationen des E/R-Modells
- Normalisierung
- Arbeiten mit relationalen Datenbanken

# Phasen des Datenbankentwurfs

## Anforderungsanalyse

Informationsbedarf

Art, Menge und Qualität der Informationen, die eine Person zur Erfüllung ihrer Aufgaben in einer bestimmten Zeit benötigt

## Konzeptioneller Entwurf

Semantisches  
Modell

Kommunikationsgrundlage / unabhängig von den Eigenschaften des Ziel-Datenbanksystems,  
**Bsp.: Entity-Relationship-Modell**

## Logischer Entwurf

Logisches  
Modell

Modellierung unter Berücksichtigung der Anforderungen des Ziel-Datenbanksystems  
**Bsp.: Relationales Modell**

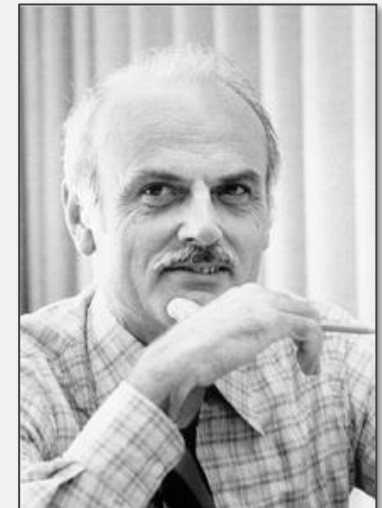
## Physischer Entwurf

Internes  
Datenmodell

Datenbank-interne Aspekte (z.B. Speicherung, Zugriffs-mechanismen/ -pfade) Überführung des log. Modells in das ausgewählte DB-System

# Relationales Modell

- In den 60er und 70er Jahren bei IBM entwickelt von Edgar F. Codd.
- Veröffentlicht 1970 im Artikel „A Relational Model of Data for Large Shared Data Banks”,  
<http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>
- Beruht auf dem mathematischen Begriff der **Relation**, den man anschaulich mit dem Begriff **Tabelle** vergleichen kann.
- Alle Informationen sind in Relationen abgelegt.
- E. F. Codd (\*1923, †2003) war ein britischer Mathematiker, der in den USA arbeitete.



# Relationales Modell: Grundlagen

Kurzer (!) Ausflug in die Mathematik

- Das Relationale Modell ist mathematisch fundiert:
  - Eine **n-stellige Relation R** ist definiert als Untermenge des kartesischen Produkts (Kreuzproduktes) der Wertebereiche der zugehörigen Attribute  $A_1, A_2, \dots, A_n$ :  
 $R \subseteq A_1 \times A_2 \times \dots \times A_n$ .
    - Beispiel: Student (MatrNr, Name, Geburtsdatum)
  - n kennzeichnet den „**Grad**“ der Relation, man spricht von einer n-stelligen Relation oder einer Menge von n-Tupeln.
  - Ein Element der Menge R wird als **Tupel** bezeichnet, d. h.  $t \in R$ .
    - Beispiel:  $t = (4711, \text{Meier}, 01.01.1985)$

# Relationales Modell: Grundlagen

- Relationen lassen sich sehr anschaulich mit folgender Zuordnung als Tabellen interpretieren:
  - die Attribute  $A_i$  sind Spaltenüberschriften
  - Tupel sind einzelne Zeilen der Tabelle („Datensätze“)
  - Relationen sind Tabellen („Dateien“)
- Beispiel: Tabellendarstellung der Relation  $Studenten \subseteq \text{Matr.Nr.} \times \text{Name} \times \text{Geburtsdatum}$

Relation (Tabelleninhalt)	<i>Studenten</i>			Attribut
	Matr.Nr.	Name	Geburtsdatum	Relationenschema
	4711	Meier	01.01.1985	
	4713	Müller	02.02.1986	Tupel (=„Datensatz“)
	...	...	...	
	5814	Schröder	03.03.1987	

# Relationales Modell: Grundlagen

## Definitionen / Begriffe:

- Die Anzahl der Zeilen (Tupel) der Tabelle heißt **Mächtigkeit** der Relation
- Die Anzahl der Spalten ist der **Grad** der Relation

## Regeln / Grundsätze:

- Jede Zeile (Tupel) ist eindeutig, d.h. unterscheidet sich von den anderen Zeilen
- Die Reihenfolge der Zeilen ist ohne Bedeutung
- Die Reihenfolge der Spalten ist ohne Bedeutung
- Die Bedeutung jeder Spalte wird durch einen Namen (den Wertebereichsnamen) gekennzeichnet.
- Alle Einträge einer Spalte sind desselben Typs



# Themenübersicht

- Warum Datenbanken?
- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
  - Grundlagen
  - ➔ **Transformationen des E/R-Modells**
  - Relationen Algebra
- Normalisierung
- Arbeiten mit relationalen Datenbanken

# Transformation des E/R-Modells in Relationen

- E/R-Modelle lassen sich leicht ohne Informationsverlust in Relationen abbilden
- Für die Transformation verschiedener Assoziationen existieren (meist) eindeutige Regeln
- Die Ansätze für die Transformation der verschiedenen Assoziationen
  - 1:1
  - 1:N
  - M:Nunterscheiden sich.

## Grundsätze für die Transformation:

- Bei der Füllung der Tabellen mit Daten sind redundante Daten zu vermeiden
- NULL-Werte (d.h. leere Einträge in Tabellen) sind möglichst zu vermeiden
- Unter Berücksichtigung der ersten beiden Punkte ist eine möglichst minimale Anzahl von Tabellen anzustreben

# Transformation des E/R-Modells in Relationen

## NULL

- Null ist ein Default-Wert, sofern möglich, und nicht speziell definiert
- Annahmen: Closed-World-Assumption (CWA)
  - Das mit der DB beschriebene Modell ist vollständig
  - Beispiel:

Tabelle Uni-Angestellter	
ID	Name
1	Sokrates
2	Platon
3	Aristoteles

Tabelle Professor	
ID	
1	
2	

- In der Tabelle Professor taucht die ID = 3 nicht auf.  
→ Aristoteles ist kein Professor

# Transformation des E/R-Modells in Relationen

## NULL: Unvollständigkeit in den Daten

- Null ist ein Default-Wert, sofern möglich, und nicht speziell definiert
- Annahmen: Closed-World-Assumption (CWA)
  - Das mit der DB beschriebene Modell ist vollständig
  - Beispiel:

Tabelle Patient	
ID	Name
1	Sokrates
2	Platon
3	Aristoteles

Tabelle Blutzucker	
ID	Blutzuckerwert [30-600]
1	90
2	120

- In der Tabelle Blutzucker taucht die ID = 3 nicht auf.  
 → Aristoteles hat keinen Blutzuckerwert    ?  
    ? ?

Quelle: Ralf Möller, Datenbanken: SQL, Vorlesungsfolien Folien 20 ff., Universität zu Lübeck, Institut für Informationssysteme, 2019

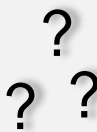
# Transformation des E/R-Modells in Relationen

## NULL ≠ NULL

- Nulls für die Modellierung von Unvollständigkeit
- Die Semantik ist nicht geklärt und wird daher häufig kritisiert
  - Beispiel:

Tabelle Patient	
ID	Name
1	Sokrates
2	Platon
3	Aristoteles

Tabelle Blutzucker	
ID	Blutzuckerwert [30-600]
1	90
2	120
3	NULL

- In der Tabelle Blutzucker taucht die ID = 3 nicht auf.  
 → Aristoteles hat einen Blutzuckerwert (30 oder 31 oder ...) 

# Transformation des E/R-Modells in Relationen

## NULL ≠ NULL

*HCG: humanes Choriongonadotropin  
Hormon, das bei Schwangerschaft gebildet wird*

- Nulls für die Modellierung von Unvollständigkeit
- Die Semantik ist nicht geklärt und wird daher häufig kritisiert
  - Beispiel:

Tabelle Patient	
ID	Name
1	Sokrates
2	Platon
3	Aristoteles
4	Xanthippe
5	Leda

Tabelle Schwangerschaft	
ID	HCG-Wert
1	NULL
2	NULL
3	NULL
4	NULL
5	130

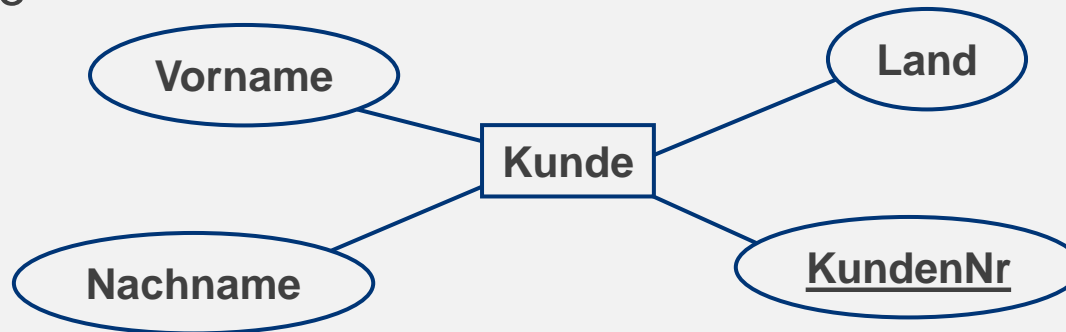
- Männliche Patienten NULL → kein HCG-Test
- Weibliche Patienten mit NULL → kein HCG-Test (aber HCG-Wert hat sie) oder HCG-Test, aber nicht bekannt

??  
??

Quelle: Ralf Möller, Datenbanken: SQL, Vorlesungsfolien Folien 20 ff., Universität zu Lübeck, Institut für Informationssysteme, 2019

# Transformation von Entities

- Entitytyp wird Tabelle
- Attribute werden Spalten
- Einzelne Entitäten entsprechen Zeilen bzw. Datensätzen
- Ein sog. „Primärschlüssel“ dient der eindeutigen Identifizierung einer Zeile



**Primär-  
schlüssel**

**Tabelle *Kunde***

<u>KundenNr</u>	Vorname	Nachname	Land
0001	Max	Meier	Österreich
0002	Nina	Niedlich	Deutschland

# Transformation von 1:1-Beziehungen

- Die Informationen werden in einer Tabelle zusammengefasst:



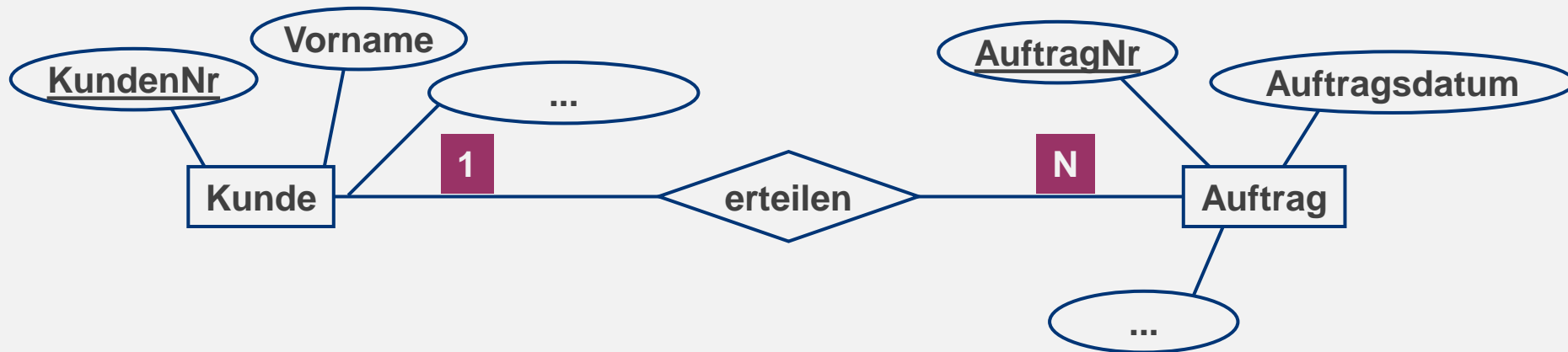
Tabelle *Kunde*

<u>KundenNr</u>	Vorname	...	Nummer	Netz
0001	Max		0664/123456	..
0002	Nina		0664/654321	..
..				



# Transformation von 1:N-Beziehungen

- Zwei Tabellen sind notwendig:
  - Tabelle Kunde
  - Tabelle Auftrag:  
enthält Primärschlüssel der übergeordneten Tabelle  
(entspricht Objekt mit Beziehung "1"), der als "Fremdschlüssel"  
bezeichnet wird (und Attribute der Assoziation)



# Transformation von 1:N-Beziehungen

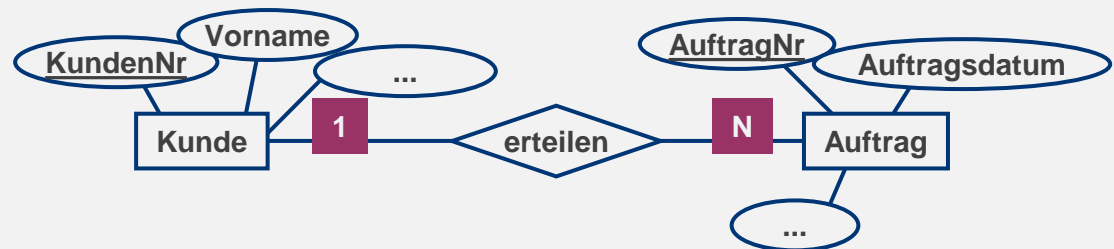
**Tabelle Auftrag**

<u>AuftragNr</u>	<u>KundenNr</u>	Auftragsdatum	...
00000001	0001	2.1.2008	
00000002	0001	3.1.2008	
00000003	0002	3.1.2008	
00000004	0007	5.1.2008	
...			

Fremdschlüssel

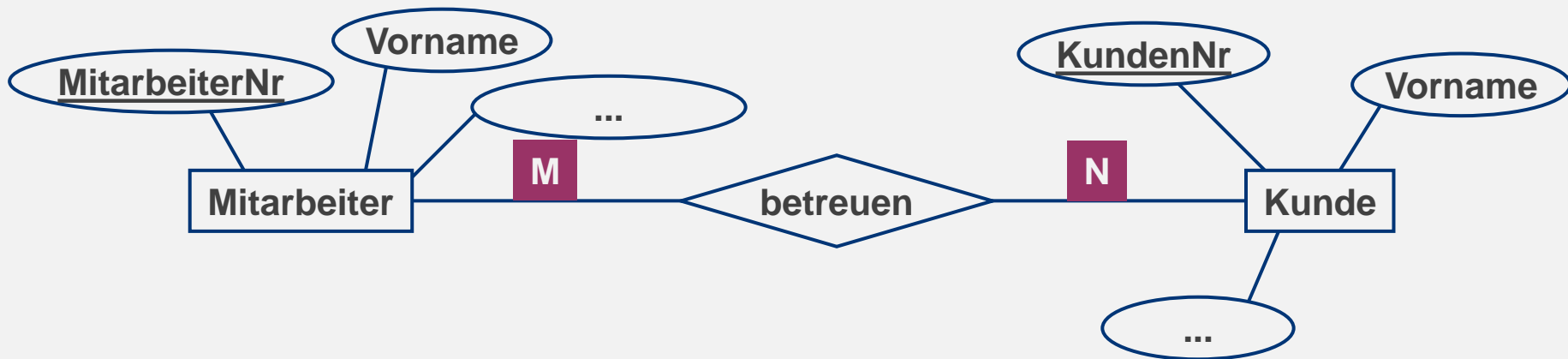
**Tabelle Kunde**

<u>KundenNr</u>	Vorname	...
0001	Max	
0002	Nina	
...		



# Transformation von M:N-Beziehungen

- Drei Tabellen sind notwendig:
  - Tabelle Mitarbeiter
  - Tabelle Kunde
  - Beziehungstabelle Kundenbetreuung: enthält Primärschlüssel der beiden Ausgangstabellen (=> Fremdschlüssel) und Attribute der Assoziation)
- Der Primärschlüssel dieser Tabelle kann, muss aber nicht aus den beiden Fremdschlüsseln zusammengesetzt sein



# Transformation von M:N-Beziehungen

Tabelle *Mitarbeiter*

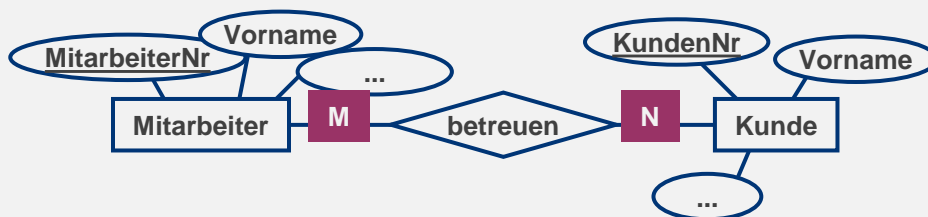
<u>MitarbeiterNr</u>	Vorname	...
01	Petra	
02	Herbert	
...		

Tabelle *Kunde*

<u>KundenNr</u>	Vorname	...
0001	Max	
0002	Nina	
...		

Tabelle *Kundenbetreuung*

<u>MitarbeiterNr</u>	<u>KundenNr</u>
02	0001
01	0002
02	0002
23	0963
...	



**N:M – Beziehungen werden somit in zwei 1:N – Beziehungen zerlegt:**

1. Tabelle Mitarbeiter –  
Tabelle Kundenbetreuung
2. Tabelle Kunde –  
Tabelle Kundenbetreuung

# Transformation von „C“-Stelligkeiten

- Erinnerung:
  - Eine 1:C-Beziehung bedeutet, dass ein Entity des einen Entitätstyps mit einem oder keinem Entity des anderen Entitätstyps in Beziehung steht. D.h. es handelt sich um eine „kann“-Beziehung.
- Liegt eine Kann-Beziehung vor, gibt es verschiedene Alternativen, dies in Relationen auszudrücken:
  - technisch aufwändig (und unüblich) für die Umsetzung von C ist die Aufspaltung der Relation für Elemente, die eine Beziehung haben und Elemente, die (noch) keine Beziehung haben
  - wird C als 1 interpretiert, lässt man NULL-Werte (leere Tabelleneinträge) zu, wobei NULL-Werte so lange wie möglich in der DB-Entwicklung vermieden werden sollen
  - **typisch ist die Interpretation von NC als N und von C als NC (und damit als N), da man dann die vorgestellten Übersetzungsschritte nutzen kann (und NULL-Einträge werden vermieden!)**

# Transformation von „C“-Stelligkeiten: Beispiel

- Problem bei Transformation analog einer 1:1-Beziehung (NULL-Werte)



**Tabelle *Mitarbeiter***

<u>MitarbeiterNr</u>	Vorname	...	HandyNr	Hersteller
01	Petra		4711	Nokia
02	Herbert		NULL	NULL
03	Klaus		4712	Samsung
...				

- Besser: Transformation analog einer 1:N-Beziehung

**Tabelle *Mitarbeiter***

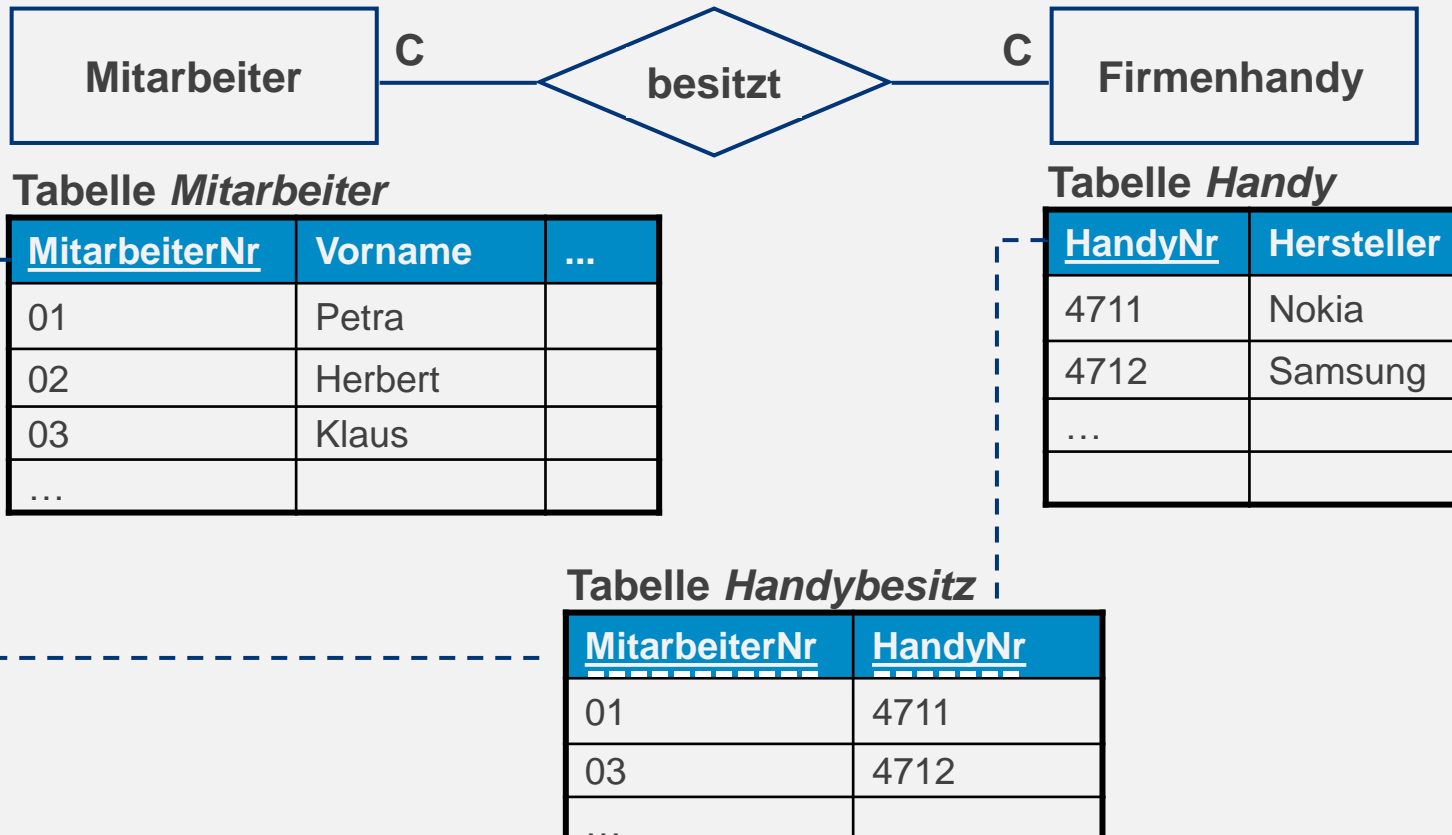
<u>MitarbeiterNr</u>	Vorname	...
01	Petra	
02	Herbert	
03	Klaus	
...		

**Tabelle *Handy***

<u>HandyNr</u>	Hersteller	<u>MitarbeiterNr</u>
4711	Nokia	01
4712	Samsung	03
...		

# Transformation von „C“-Stelligkeiten: Beispiel

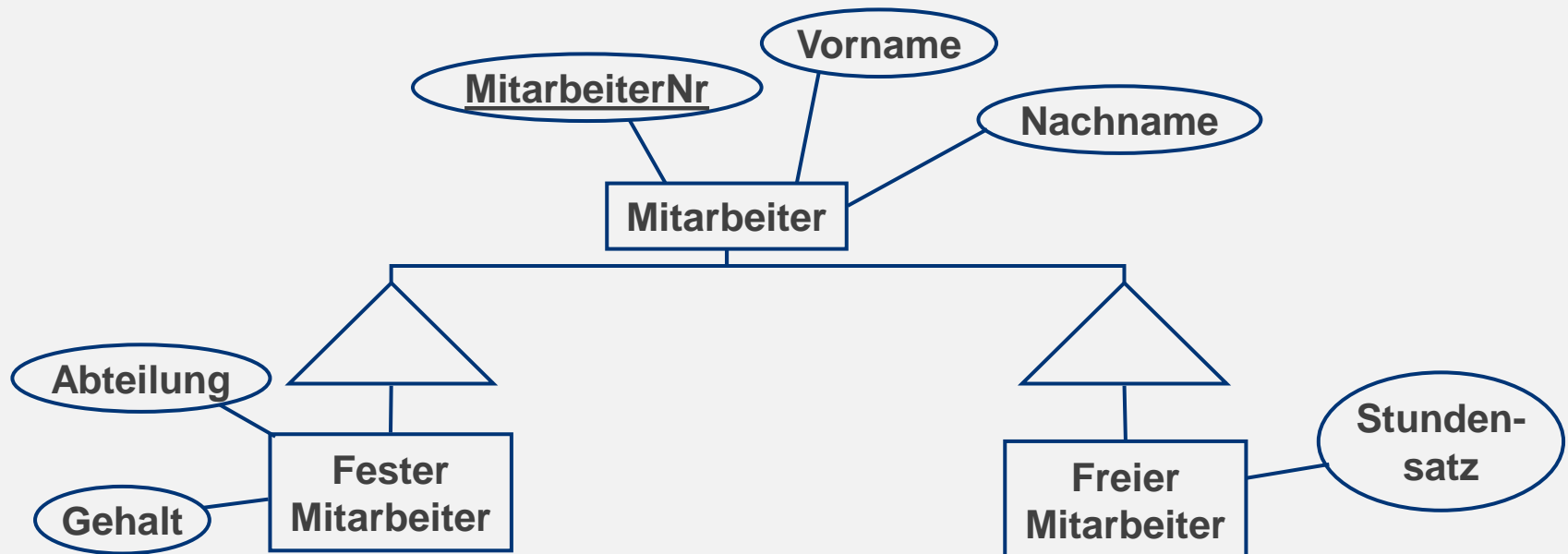
- Transformation analog einer M:N-Beziehung



**Zuordnungstabelle wird benötigt, da sonst NULL-Werte in der Handy-Tabelle stehen könnten (immer dann, wenn ein Handy keinem Mitarbeiter gehört)**

# Transformation von Generalisierungen/Spezialisierungen (1/3)

- Für Fester Mitarbeiter und Freier Mitarbeiter sind eigene Tabellen erforderlich
- Was ist mit dem Entitätstyp Mitarbeiter ?





# Transformation von Generalisierungen/Spezialisierungen (2/3)

**Tabelle *Mitarbeiter***

<u>MitarbeiterNr</u>	Vorname	...
01	Petra	
02	Herbert	
...		

**Tabelle *Fester Mitarbeiter***

<u>MitarbeiterNr</u>	Gehalt	Abteilung
01	4000	Einkauf
03	5000	Vertrieb
04	3000	Vertrieb
07	...	

**Tabelle *Freie Mitarbeiter***

<u>MitarbeiterNr</u>	Stundensatz
02	50
05	70
...	

## „Partitionierungsmodell“

Für die Spezialisierungen werden eigene Tabellen modelliert, für die Generalisierung existiert eine eigene Tabelle

**Vorteil:** Alle Entities sind in einer Tabelle abfragbar

**Nachteil:** Beim Anlegen neuer Entities sind stets mehrere Tabellen anzusprechen

# Transformation von Generalisierungen/Spezialisierungen (3/3)

**Tabelle *Fester Mitarbeiter***

<u>MitarbeiterNr</u>	Vorname	...	Gehalt	Abteilung
01	Petra		4000	Einkauf
03	...		5000	Vertrieb
04	...		3000	Vertrieb
07	...		...	
...				

**Tabelle *Freie Mitarbeiter***

<u>MitarbeiterNr</u>	Vorname	...	Stundensatz
02	Herbert		50
05			70
...			

## „Hausklassenmodell“

Es werden nur Tabellen für die Spezialisierung angelegt, die jeweils alle Attribute (auch die generellen) enthalten

**Vorteil:** Anlegen neuer Entities in nur einer Tabelle

**Nachteil:** Generelle Abfragen (z.B. Abfrage aller Mitarbeiter) werden erschwert

# Vorgehen zur Transformation in einem Relationenmodell

1. Schritt: Jeder **Entitätstyp** wird in eine Tabelle übersetzt. Attribute des Entitätstyp werden zu Spaltennamen (Attribute der Tabelle).
2. Schritt: Übersetzung von **1:1 Beziehungen**: Die betreffenden Tabellen werden zusammengeführt. Hat die Beziehung Attribute, werden diese mit in die Tabelle aufgenommen.
3. Schritt: Übersetzung von **1:N Beziehungen**: Zur Übersetzung der Beziehung werden die identifizierenden Attribute (Primärschlüssel) der übergeordneten Tabelle (1-Beziehung) als zusätzliche Attribute (Fremdschlüssel) in die untergeordnete Tabelle (N-Beziehung) übernommen.
4. Schritt: Übersetzung von **M:N Beziehungen**: Eine neue Tabelle wird angelegt, die Primärschlüssel der in Beziehung stehenden Entitäten als Fremdschlüssel enthält, sowie die Attribute der Beziehung.

# Themenübersicht

- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
  - Grundlagen
  - Transformationen des E/R-Modells
- ➡ **Relationen-Algebra**
- Normalisierung
- Arbeiten mit relationalen Datenbanken

# Relationen-Algebra

- Theoretische Grundlage relationaler Datenbanken

- Anfragen formulieren
- Informationen zusammenstellen

- **Klassische Operationen**



- **Spezielle Operationen**

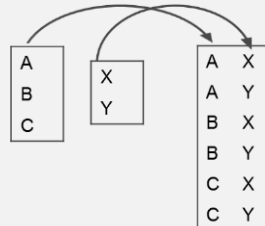
- Selektion



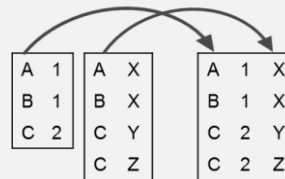
Projektion



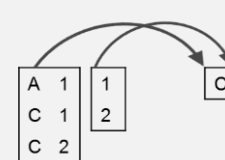
- Kartesisches Produkt



Natural Join



Division



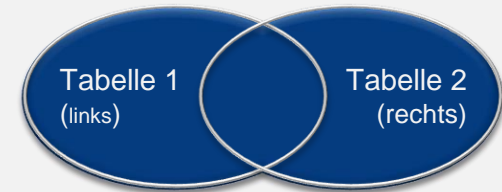
## Vereinigung – Durchschnitt – Differenz

- Klassische Operationen nach der Mengenlehre
- Zwei Relationen mit dem gleichen Schema
  - Anzahl der Attribute
  - Tupel-Typen der Attribute
  - Attribut-Reihenfolge

Vereinigungsverträgliche  
Relationen

# Vereinigung

- Vereinigung zweier Mengen, wobei Duplikate entfernt werden.
- Einträge, die in der einen oder der anderen Relation vorkommen.



$$R \cup S = \{ r \mid r \in R \text{ oder } r \in S \}$$

VK1

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

VK2

Verkäufer	Produkt	Käufer
Müller	Hemd	Schmidt
Müller	Rock	Schmidt
Meier	Rock	Schulz

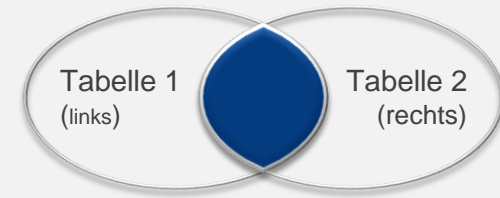


VK1 U VK2

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz
Müller	Hemd	Schmidt
Meier	Rock	Schulz

# Durchschnitt

- Durchschnitt zweier Mengen
- Einträge, die sowohl in der einen als auch in der anderen Relation vorkommen.



$$R \cap S = \{ r \mid r \in R \text{ und } r \in S \}$$

VK1

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

VK2

Verkäufer	Produkt	Käufer
Müller	Hemd	Schmidt
Müller	Rock	Schmidt
Meier	Rock	Schulz



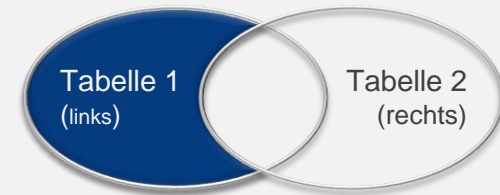
VK1 ∩ VK2

Verkäufer	Produkt	Käufer
Müller	Rock	Schmidt



# Differenz

- Differenz zweier Mengen
- Einträge, die nur in der ersten, aber nicht in der zweiten Relation enthalten sind.



$$R - S = \{ r \mid r \in R \text{ und nicht } r \in S \}$$

VK1

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

VK2

Verkäufer	Produkt	Käufer
Müller	Hemd	Schmidt
Müller	Rock	Schmidt
Meier	Rock	Schulz



VK1 - VK2

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Meier	Hose	Schulz

**Unterschiedliche Ergebnisse  
für  $R - S$  oder  $S - R$ !**

# Selektion

- Die Selektion ist eine Abbildung einer Relation R aufgrund der Bedingung B

$$R_n \rightarrow R_n$$

$$R \rightarrow \text{Selektion}(R, B)$$

## Selektion


VK

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz



*Selektion(VK, Verkäufer = 'Meier')*

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Meier	Hose	Schulz

# Selektion

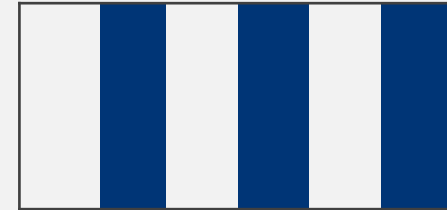
- Bestandteile eines Selektionsprädikates
  - Attribute einer Relation und Konstanten als Operanden
  - Vergleichsoperatoren  
=, <, ≤, >, ≥, und <>
  - Logische Operatoren  
UND, ODER, NICHT
  - Kombination aus allen Möglichkeiten, die durch Klammerung erzeugt werden.

## Selektion


# Projektion

- Die Projektion extrahiert bestimmte Attribute (Spalten), vertauscht ggfs. ihre Reihenfolge und kann den Attributnamen ändern.

Projektion



VK

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

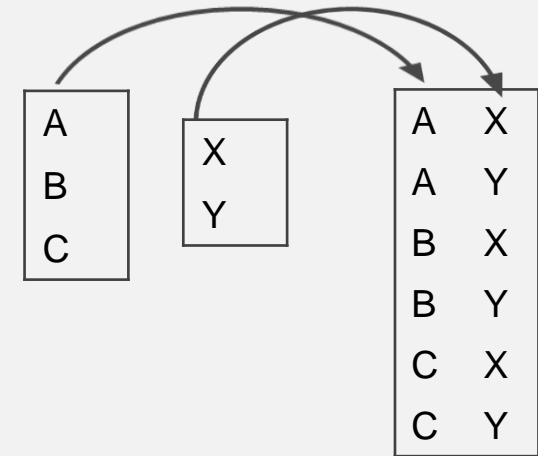


*Projektion(VK[Käufer, Produkt])*

Käufer	Produkt
Schmidt	Hose
Schmidt	Rock
Schulz	Hose

# Kartesisches Produkt

- Das Kartesische Produkt (Kreuzprodukt) ist die Menge aller Paare aus Tupeln der ersten Relation verknüpft mit Tupeln der zweiten Relation.



*VK*

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

*PL*

Produkt	Preis	Klasse
Hose	100	B
Rock	200	A



*VK × PL*

Verkäufer	Produkt	Käufer	Produkt	Preis	Klasse
Meier	Hose	Schmidt	Hose	100	B
Meier	Hose	Schmidt	Rock	200	A
Müller	Rock	Schmidt	Hose	100	B
Müller	Rock	Schmidt	Rock	200	A
Meier	Hose	Schulz	Hose	100	B
Meier	Hose	Schulz	Rock	200	A

# JOIN-Operatoren

- JOIN-Operatoren verbinden ähnlich wie das kartesische Produkt zwei Relationen. Dabei werden aber nur solche Tupel ausgewählt, die in einer Beziehung zueinander stehen.
- Der **THETA-JOIN** ist eine Operation bei der zuerst das kartesische Produkt und auf die Ergebnismenge die Selektion ausgeführt wird.

***THETA-JOIN(VK, PL, VK.Produkt = PL.Produkt)***

Verkäufer	Produkt	Käufer	Produkt	Preis	Klasse
Meier	Hose	Schmidt	Hose	100	B
Müller	Rock	Schmidt	Rock	200	A
Meier	Hose	Schulz	Hose	100	B

***THETA-JOIN(VK, PL, VK.Produkt = PL.Produkt AND Preis < 200)***

Verkäufer	Produkt	Käufer	Produkt	Preis	Klasse
Meier	Hose	Schmidt	Hose	100	B
Meier	Hose	Schulz	Hose	100	B

# JOIN-Operatoren

- Der **EQUI-JOIN** entspricht einem Theta-Join, der nur den Vergleichsoperator "=" im Selektionsprädikat zulässt.
- Der Theta-Join ( $R_1, R_2, A_1 = B_1$ ) ist auch ein Beispiel für einen EQUI-JOIN. Damit ist der EQUI-JOIN ein Spezialfall des Theta-Joins.

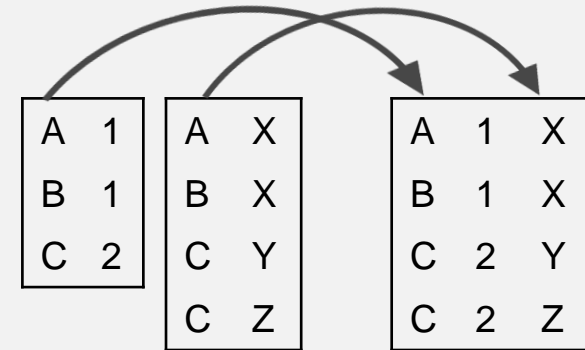
*THETA-JOIN(VK, PL, VK.Produkt = PL.Produkt)*

*EQUI-JOIN(VK, PL)*

Verkäufer	Produkt	Käufer	Produkt	Preis	Klasse
Meier	Hose	Schmidt	Hose	100	B
Müller	Rock	Schmidt	Rock	200	A
Meier	Hose	Schulz	Hose	100	B

# JOIN-Operatoren

- Beim **NATURAL JOIN** werden automatisch alle gleich lautenden Spalten verglichen und doppelte Spalten entfernt.
- Der **NATURAL JOIN** entspricht also dem kartesischen Produkt mit anschließender Selektion und Projektion.



VK

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

PL

Produkt	Preis	Klasse
Hose	100	B
Rock	200	A



**NATURAL JOIN (VK, PL)**

Verkäufer	Produkt	Käufer	Preis	Klasse
Meier	Hose	Schmidt	100	B
Müller	Rock	Schmidt	200	A
Meier	Hose	Schulz	100	B



# JOIN-Operatoren

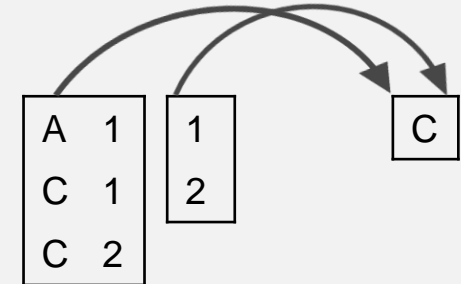
- Die **OUTER JOINS** ermöglichen es, Tupel im Ergebnis mit aufzunehmen, die beim Natural Join herausfallen. Diejenigen Tupel, die nur in einer Relation vorkommen, werden mit NULL aufgefüllt.
  - **OUTER JOIN**( $R_1, R_2$ )  
alle Tupel aus der rechten **und** der linken Relation werden aufgeführt und mit NULL aufgefüllt.
  - **LINKER OUTER JOIN** ( $R_1, R_2$ )  
alle Tupel aus der linken Relation ( $R_1$ ) werden aufgeführt, Spalten aus der rechten Relation ( $R_2$ ) werden mit NULL aufgefüllt.
  - **RECHTER OUTER JOIN** ( $R_1, R_2$ )  
alle Tupel aus der rechten Relation ( $R_2$ ) werden aufgeführt, Spalten aus der linken Relation ( $R_1$ ) werden mit NULL aufgefüllt.

# JOIN-Operatoren

- Eigenschaften der JOIN-Operatoren
  - Attribute für Joins müssen keine Schlüsselattribute sein.
  - Join-Attribute der Relationen müssen nicht den gleichen Namen haben (ausgenommen beim Natural-Join)
  - Jede Relation kann mit einer anderen Relation gejoint werden – auch mit sich selbst.
  - Alle Join-Operatoren lassen sich aus Selektion, Projektion und kartesischem Produkt ableiten.

# Division

- Division zwei Relationen  $R_1$  und  $R_2$ .
- Die Attribute von  $R_2$  sollen in  $R_1$  enthalten sein.
- Lässt sich aus Selektion, kartesischem Produkt und Differenz ableiten.
- All-Quantor-Abfrage: Gibt die Tupel aus, die mit allen Tupeln von  $R_1$  und  $R_2$  verknüpft sind.



VK

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz
Meier	Rock	Schmidt

„Welcher Verkäufer verkaufte an den gleichen Käufer **alle** Produkte ?“

VK

Verkäufer
Meier



PL

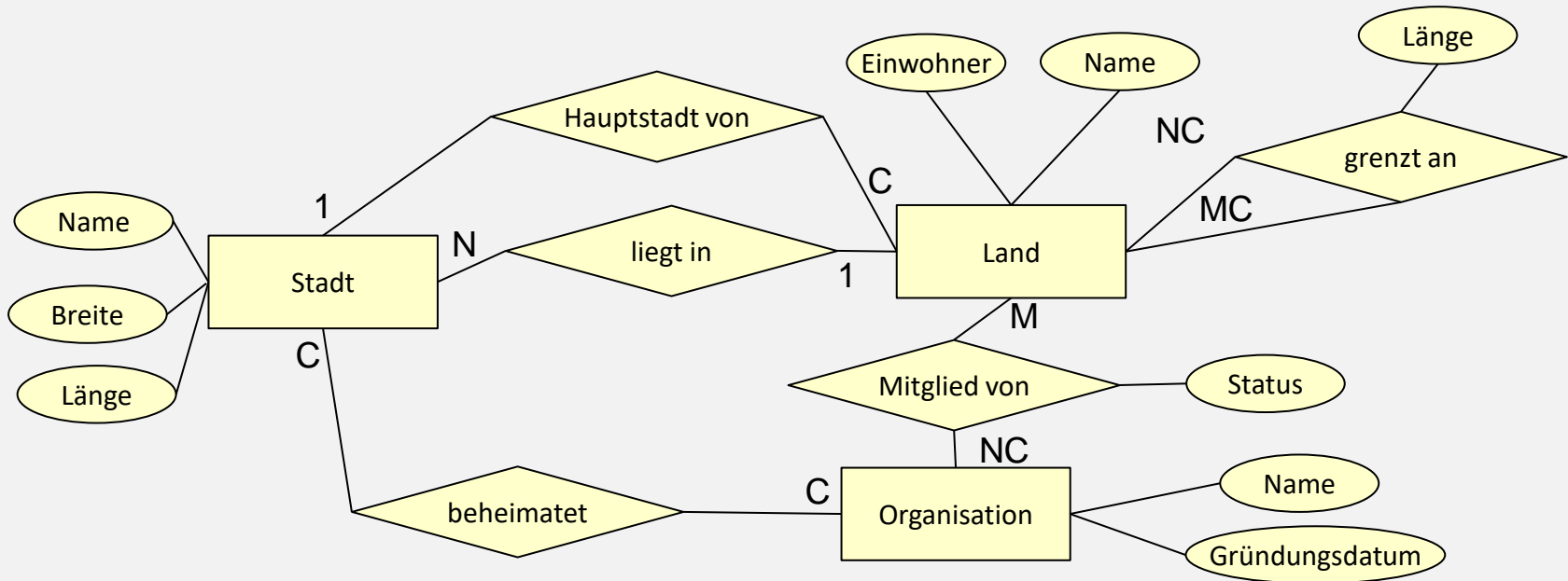
Produkt
Hose
Rock

# Zusammenfassung (1/2)

- Phasen des Datenbankentwurfs
  - Logischer Entwurf
    - Logisches Modell, z.B. Relationales Modell
- Relationales Modell
  - Grundelemente eines Relationalen Modells
    - Relation
    - Relationsschema
    - Attribut
    - Tupel
    - Primärschlüssel
    - Fremdschlüssel
  - Begriffe
  - Regeln

## Zusammenfassung (2/2)

- Transformation ER-Modell → Relationales Modell
  - Grundsätze / allgemeine Regeln
  - Bedeutung von Null-Werten
  - 1:1-Beziehungen
  - 1:N-Beziehungen
  - M:N-Beziehungen
  - „C“-Stelligkeiten
  - Generalisierung/ Spezialisierung
    - Partitionierungsmodell
    - Hausklassenmodell
- Relationen Algebra
  - Anfragen formulieren und Informationen zusammenstellen
  - Klassische Mengenoperationen
    - Vereinigung, Schnittmenge, Differenz
  - Spezielle Mengenoperationen
    - Selektion, Projektion, Join, Division



1. Leiten Sie aus dem Diagramm Tabellen ab und markieren Sie einen Schlüsselkandidaten in jeder Tabelle. Vermeiden Sie die Ableitung überflüssiger Einzeltabellen.
2. Nennen Sie zwei Korrekturen, die Sie im ER-Diagramm durchführen würden, um das Modell realistischer zu machen, die sich nicht auf Attribute beziehen.