



**Hasso  
Plattner  
Institut**

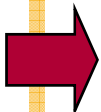
IT Systems Engineering | Universität Potsdam

Datenbanksysteme I  
Relationaler Datenbankentwurf

5.11.2007

Felix Naumann

2

- 
- Das Relationale Modell
  - Von ER-Diagrammen zu Relationenschemata
  - Konvertierung von Spezialisierung
  - Funktionale Abhängigkeiten (FDs)
  - Ableitungsregeln für FDs
  - Normalformen



# Die Relation

3

Konzeptuell ist eine Datenbank eine Menge von Tabellen.

- Relation zwischen Werten der Attributdomänen
- Tabellen = Relationen

<b>Titel</b>	<b>Jahr</b>	<b>Länge</b>	<b>Typ</b>
Basic Instinct	1992	127	Farbe
Total Recall	1990	113	Farbe
Dead Man	1995	121	s/w

Die Relation ist das einzige Konstrukt des relationalen Modells

- Sehr einfach
- Einfach in einer DB abzubilden (zwei-dimensional)
- Relationen können nicht nur Entities sondern auch Relationships darstellen.
- Entspricht oft unserer Vorstellung der Daten
- Ist das abstrakte Modell hinter SQL.

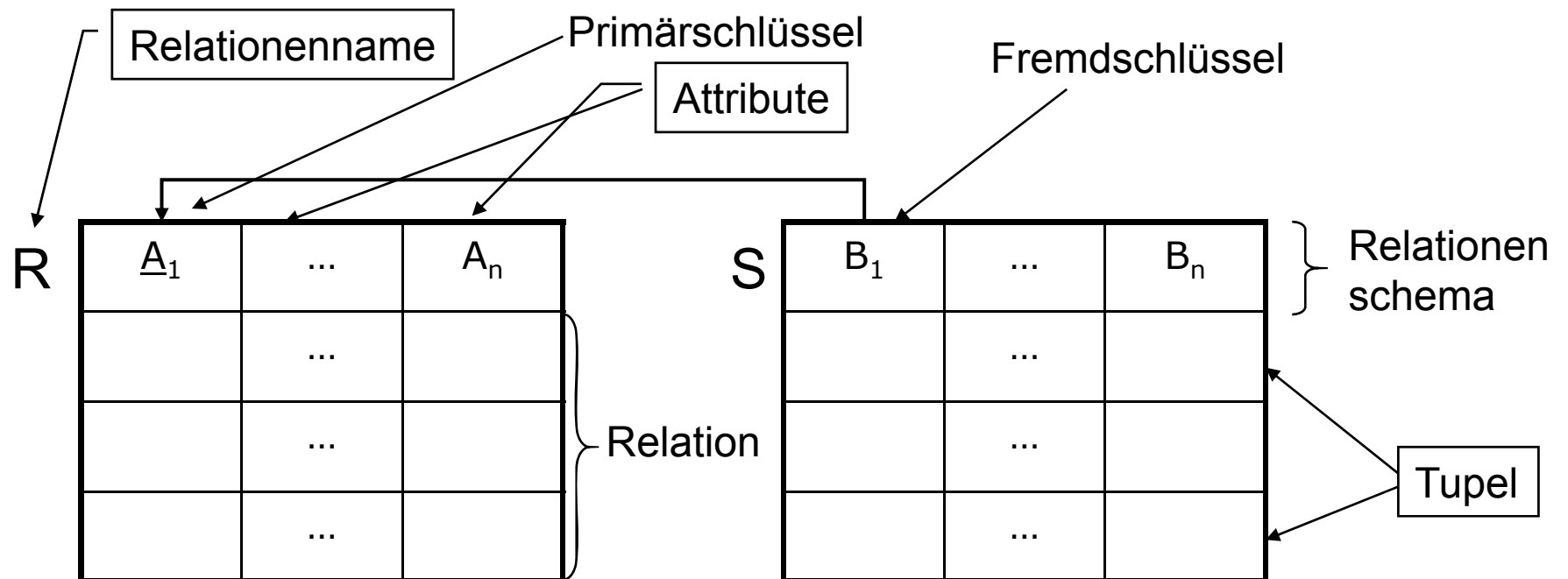
# Elemente einer Relation

4

- *Datenbankschema*
  - Besteht aus einem oder mehreren Relationenschemata.
- *Relationenschema*
  - Weitere Einträge in der Tabelle: Die „*Relation*“
  - Besteht aus keinem oder mehr Tupeln.
- Eine Zeile der Tabelle: *Tupel*
  - Tupel bilden eine Menge (nicht eine Liste).
- Eine Spaltenüberschrift: *Attribut*
  - Attribute bilden eine Menge (nicht eine Liste).
- Ein Eintrag: *Attributwert*
  - Atomar
  - Stammt aus einer elementaren Domäne (Integer, String, ...)

# Elemente einer Relation

5



# Formal

6

- Domänen  $D_1, \dots, D_n$
- Relation  $R \subseteq D_1 \times \dots \times D_n$

## Beispiel

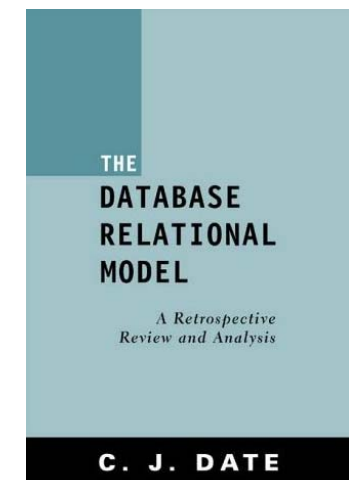
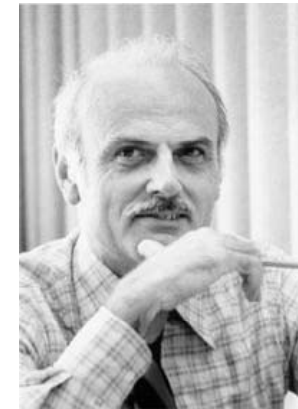
- Relationenschema: Filme(Titel, Jahr, Länge, Typ)
- Tupel: (Star Wars, 1977, 124, farbig)

# Edgar F. Codd

7

[http://en.wikipedia.org/wiki/Edgar\\_F.\\_Codd](http://en.wikipedia.org/wiki/Edgar_F._Codd)

- Promotion an der University of Michigan Ann Arbor
- Entwicklung des Relationalen Modells bei IBM (Almaden)
- „A Relational Model of Data for Large Shared Data Banks“ (1970)
- Artikelserie
- Literaturhinweis:
  - The Database Relational Model: A Retrospective Review and Analysis :
    - ◇ A Historical Account and Assessment of E. F. Codd's Contribution to the Field of Database Technology
    - ◇ Chris J. Date
  - ISBN: 0-201-61294-1 (9.99 EUR)



## *Contributions* (nach C.J.Date)

8

- Transformation von Datenmanagement zu einer Wissenschaft
  - Entsprechende Klarheit und Strenge
- Nicht nur das relationale Modell, sondern überhaupt das Konzept eines Datenmodells
  - Unterscheidung zwischen Modell und Implementierung
- Relationale Algebra und relationales Kalkül
- Informell: Anfragesprache Alpha
  - Angelehnt: SEQUEL von Chamberlin und Boyce
- Funktionale Abhängigkeiten
- Normalformen
  - Erste bis dritte Normalform



599

DERIVABILITY, REDUNDANCY AND CONSISTENCY OF RELATIONS  
STORED IN LARGE DATA BANKS

E. F. Codd  
Research Division  
San Jose, California

ABSTRACT: The large, integrated data banks of the future will contain many relations of various degrees in stored form. It will not be unusual for this set of stored relations to be redundant. Two types of redundancy are defined and discussed. One type may be employed to improve accessibility of certain kinds of information which happen to be in great demand. When either type of redundancy exists, those responsible for control of the data bank should know about it and have some means of detecting any "logical" inconsistencies in the total set of stored relations. Consistency checking might be helpful in tracking down unauthorized (and possibly fraudulent) changes in the data bank contents.

RJ 599 (# 12343)  
August 19, 1969

## INTRODUCTION

The first part of this paper is concerned with an explanation of a relational view of data. This view (or model) of data appears to be superior in several respects to the graph or network model [1, 2] presently in vogue. It provides a means of describing data with its natural structure only: that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level retrieval language which will yield maximal independence between programs on the one hand, and machine representation and organization of data on the other. A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations -- these are discussed in the second part of this paper. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations. Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present management information systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system.

1. A Relational View of Data

The term relation is used here in its accepted mathematical sense. Given sets  $S_1, S_2, \dots, S_n$  (not necessarily distinct),  $R$  is a relation on these  $n$  sets if it is a set of  $n$ -tuples, each of which has its first element from  $S_1$ , its second element from  $S_2$ , and so on. We shall refer to  $S_j$  as the  $j^{\text{th}}$  domain of  $R$ . As defined above,  $R$  is said to have degree  $n$ . Relations of degree 1 are often called unary, degree 2 binary, degree 3 ternary, and degree  $n$  n-ary.

For expository reasons, we shall frequently make use of an array representation of relations, but it must be remembered that this particular representation is not an essential part of the relational view being expounded. An array which represents an  $n$ -ary relation  $R$  has the following properties:

- (1) Each row represents an  $n$ -tuple of  $R$ ;
- (2) The ordering of rows is immaterial;
- (3) All rows are distinct;
- (4) The ordering of columns is significant - it corresponds to the ordering  $S_1, S_2, \dots, S_n$  of the domains on which  $R$  is defined;
- (5) The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

2.

The example in Figure 1 illustrates a relation of degree 4 called ship which reflects the shipments-in-progress of parts from specified suppliers to specified projects in specified quantities.

<u>ship</u>	( <u>supplier</u>	<u>part</u>	<u>project</u>	<u>quantity</u> )
	1	2	5	17
	1	3	5	23
	2	3	7	9
	2	7	5	4
	4	1	1	12

FIGURE 1: A Relation of Degree 4

One might ask: If the columns are labeled by the name of the corresponding domains, why should the ordering of columns matter? As the example in Figure 2 shows, two columns may have identical headings (indicating identical domains), but possess distinct meanings with respect to the relation. The relation depicted is called component. It is a binary relation, each of whose two domains is called part. The meaning of component (x, y) is that part x is an immediate component (or subassembly) of part y.

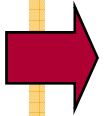
<u>component</u>	( <u>part</u>	<u>part</u> )
	1	5
	2	5
	3	5
	2	6
	3	6
	4	7
	6	7

Figure 2: A Relation with Two Identical Domains

We now assert that a data bank is a collection of time-varying relations. These relations are of assorted degrees. As time progresses, each n-ary relation may be subject to insertion of additional n-tuples, deletion of existing ones, and alteration of components of any of its existing n-tuples.

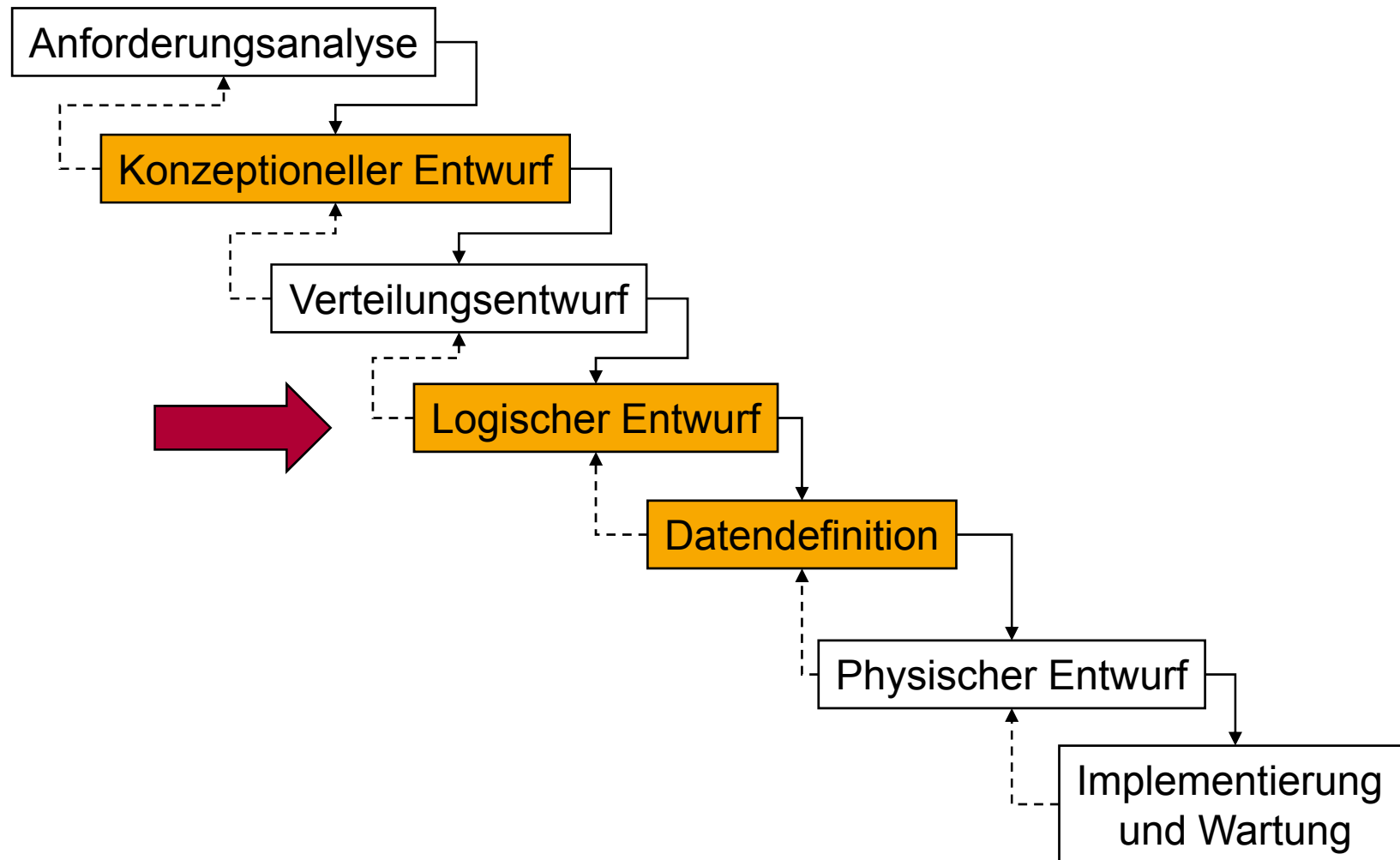
Consider, for example, a data bank which contains information about parts, projects, and suppliers. The individual description for an individual object (such as a particular part) is called an entity [3]. The prototype description for a class of objects is called an entity type. The set of entities of a given entity type can be viewed as a relation, and we shall call such a relation an entity type relation. In the example under consideration, there might be an entity type relation called part defined on the following domains:

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Funktionale Abhängigkeiten (FDs)
- Ableitungsregeln für FDs
- Normalformen



# Phasenmodell für den Datenbankentwurf

13



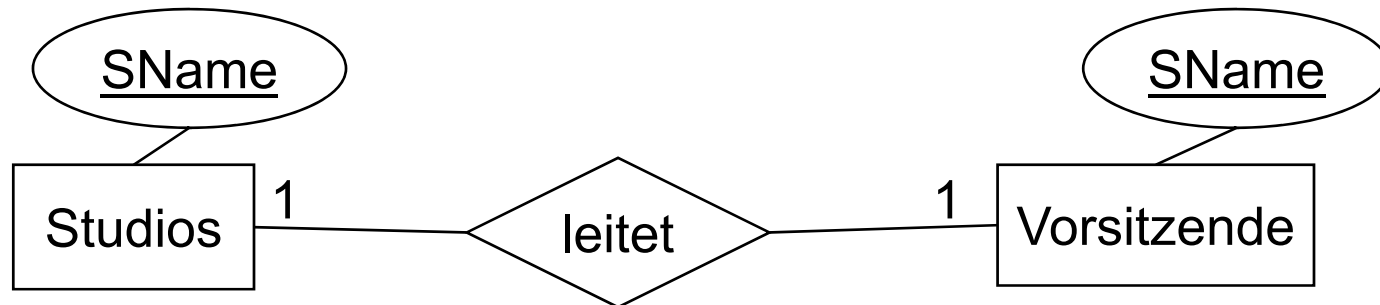
# Ziele der ER-Abbildung

14

- Darstellung aller Informationen des ER-Diagramms
- Exaktheit
  - Das Datenbankschema kann genauso viele Instanzen wie das ER-Diagramm darstellen.
  - Das Datenbankschema kann nicht mehr Instanzen als das ER-Diagramm darstellen.
- Erhaltung und Einhaltung der Informationskapazität!

# Kapazitätserhöhende Abbildung

15



Relationenschema:  $R = \{SName, VName\}$



Schlüsselmenge:  $\{ \{SName\} \}$

Schlüsselmenge:  $\{ \{SName\}, \{SName\} \}$

SName	VName
Fox	Iger
Disney	Iger

SName	VName
Fox	Murdoch
Disney	Iger

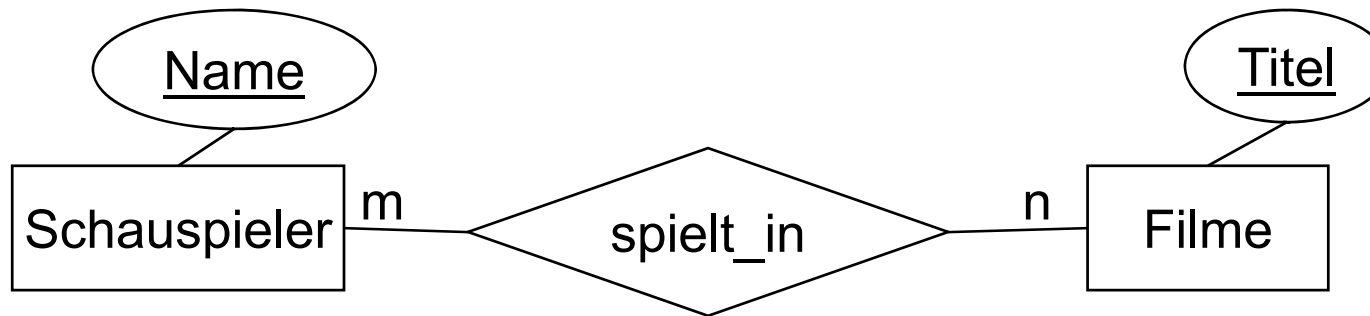
SName	VName
Fox	Murdoch
Disney	Iger

kapazitätserhöhend

kapazitätserhaltend

# Kapazitätsvermindernde Abbildung

16



Relationenschema:  $R = \{ \text{Name}, \text{Titel} \}$

Schlüsselmenge:  $\{ \{ \text{Name} \} \}$

Name	Titel
Sharon Stone	Basic Instinct
Michael Douglas	Basic Instinct

kapazitätsvermindernd

Schlüsselmenge:  $\{ \{ \text{Name}, \text{Titel} \} \}$

Name	Titel
Sharon Stone	Basic Instinct
Michael Douglas	Basic Instinct

kapazitäts-  
erhaltend

Name	Titel
Sharon Stone	Basic Instinct
Sharon Stone	Total Recall
Michael Douglas	Basic Instinct



# Grundalgorithmus

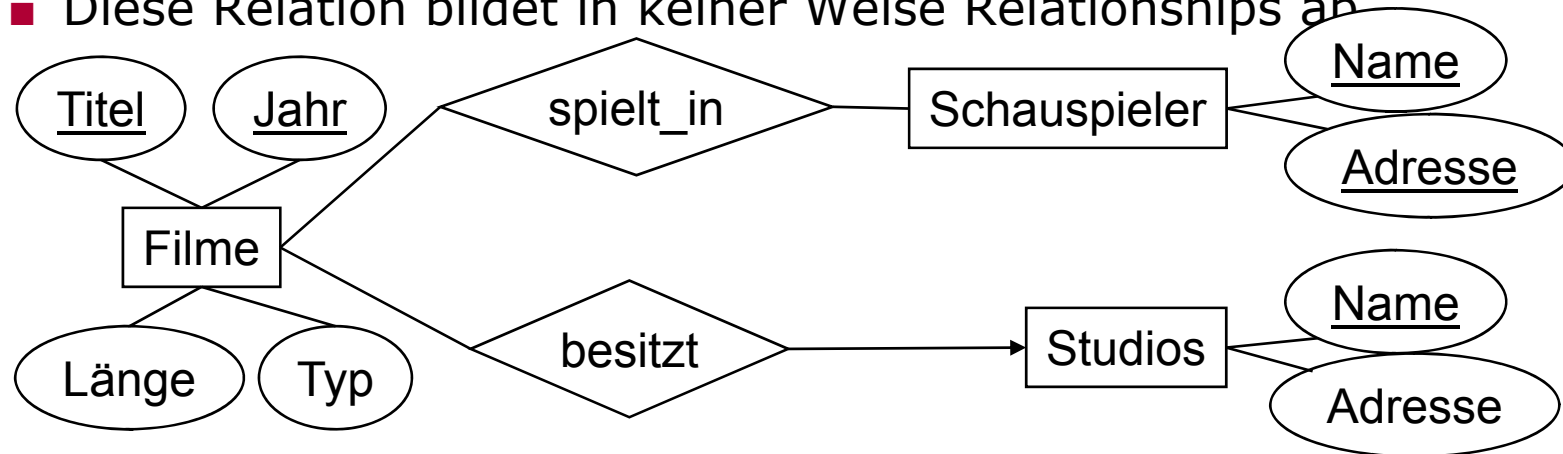
17

1. Wandle jeden Entitytypen in eine Relation mit den gleichen Attributen um.
  2. Wandle jeden Relationshiptypen in eine Relation um, deren Attribute die zugehörigen Attribute und die Schlüsselattribute der beteiligten Entitytypen sind.
  3. Verfeinere den Entwurf
    1. Zusammenlegung von Relationen
    2. Normalisierung
- Ausnahmen
    - Schwache Entitytypen
    - IST Relationships

# Entitytyp → Relation

18

- Name des Entitytyps → Name der Relation
- Attribute des Entitytyps → Attribute der Relation
- Diese Relation bildet in keiner Weise Relationships ab



- Filme(Titel, Jahr, Länge, Typ)
- Schauspieler(Name, Adresse)
- Studios(Name, Adresse)

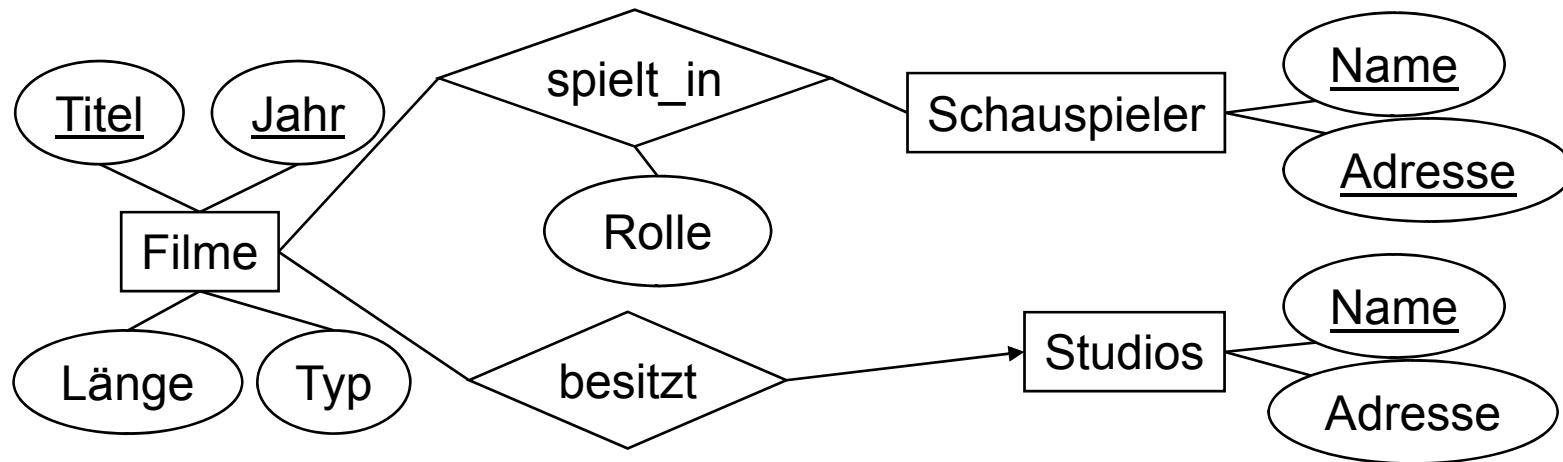
## Relationshiptyp → Relation

19

- Attribute:
  - Für jeden beteiligten Entitytypen: Füge deren Schlüsselattribut(e) als Attribute hinzu
  - Attribute des Relationshiptyps selbst
- Doppelte Attributnamen
  - Umbenennungen sind nötig!
- Falls ein Entitytyp in mehreren Rollen beteiligt ist
  - Entsprechend oft die Schlüsselattribute übernehmen
  - Geeignete Umbenennungen sind nötig

## Relationshiptyp → Relation

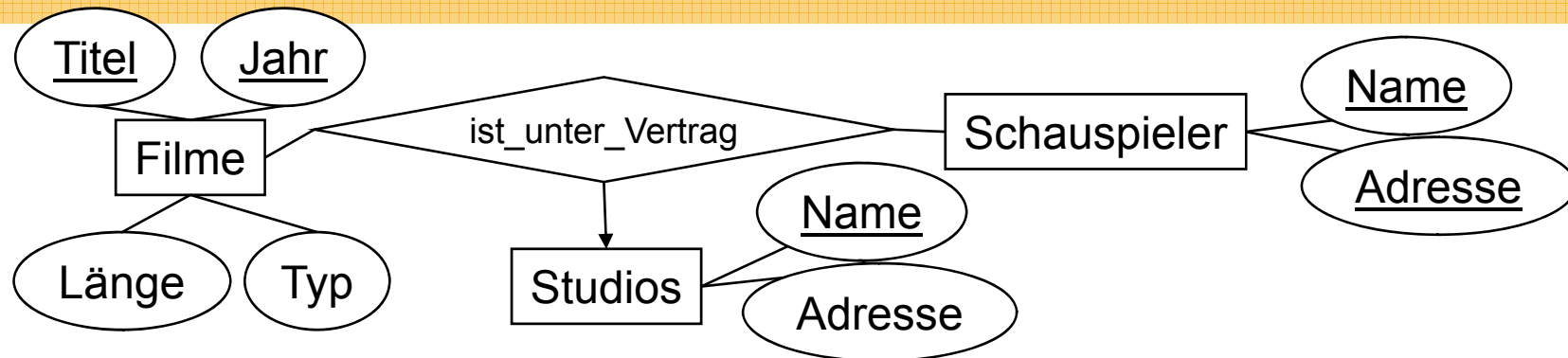
20



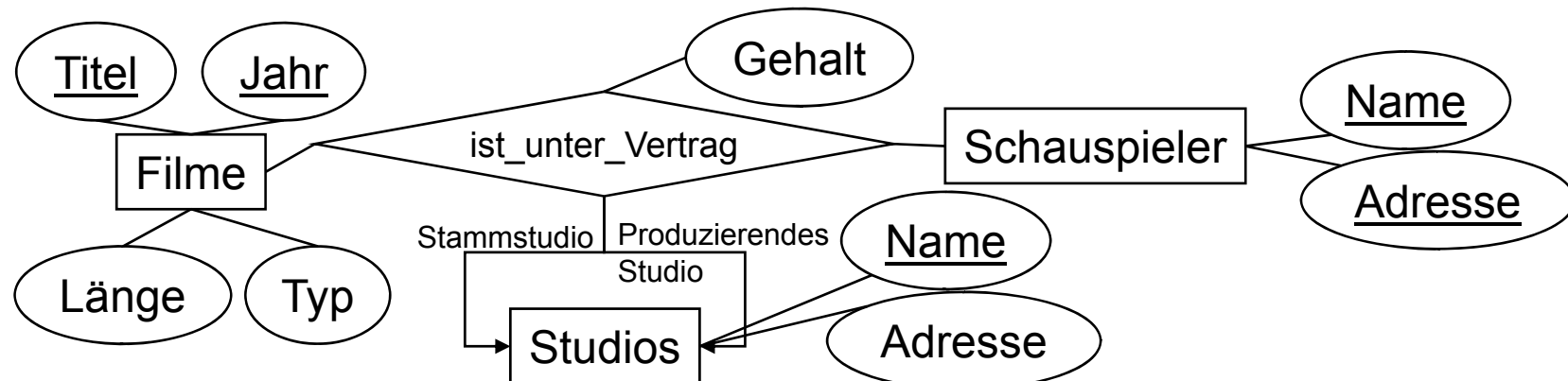
- spielt\_in(Titel, Jahr, SchauspielerName, SchauspielerAdresse, Rolle)
- besitzt(Titel, Jahr, StudioName)
  
- Umbenennungen nur zur Klarheit

# Relationshiptyp → Relation

21



- ist\_unter\_Vertrag(Titel, Jahr, SchauspielerName, SchauspielerAdresse, StudioName)



- ist\_unter\_Vertrag(Titel, Jahr, SchauspielerName, SchauspielerAdresse, Stammstudio, ProduzierendesStudio, Gehalt)

# Zusammenlegen von Relationen

22

Man kann folgende Relationen kombinieren:

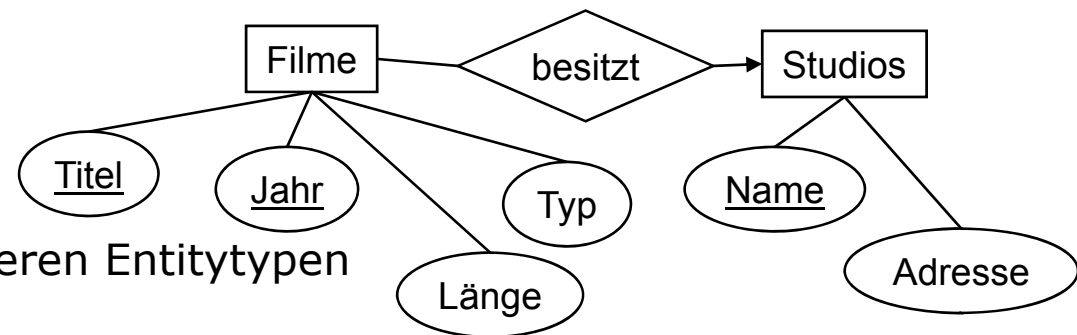
- Die Relation für einen Entitytypen  $E$
- Mit der Relation eines 1:n Relationshiptypen  $R$ , falls  $E$  auf den n-Seite liegt.

Begründung

- $R_E$  enthält den Schlüssel von  $E$ .
- $R_E$  enthält alle nicht-Schlüssel Attribute von  $E$ .
- $R_R$  enthält Schlüssel von  $E$ .
- $R_R$  enthält sonstige Attribute von  $R$ .
- $R_R$  enthält Schlüssel des anderen Entitytypen
- Sämtliche Attributwerte werden eindeutig durch den Schlüssel von  $E$  bestimmt.

Neue Relation enthält also

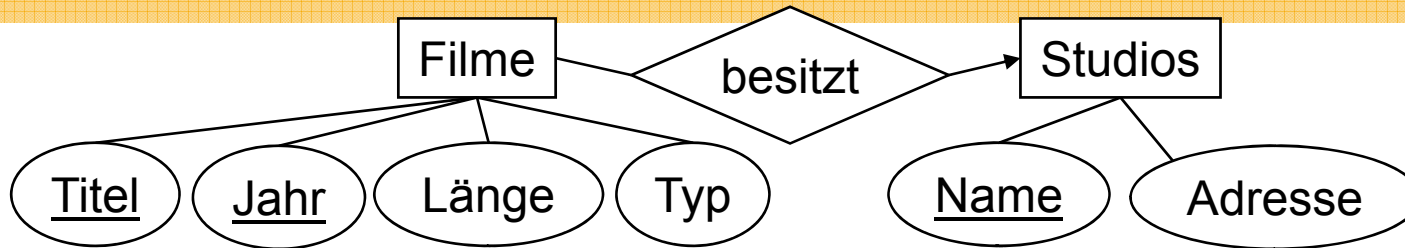
- Alle Attribute von  $E$
- Alle Attribute von  $R$ 
  - inkl. Schlüssel des anderen Entitytypen



# Zusammenlegen von Relationen

## 1:n-Relationships

23



Filme

Titel	Jahr	Länge	Typ
Basic Instinct	1992	127	Farbe
Total Recall	1990	113	Farbe
Dead Man	1995	121	s/w

besitzt

Titel	Jahr	studioName
Basic Instinct	1992	Fox
Total Recall	1990	Disney
Dead Man	1995	Paramount

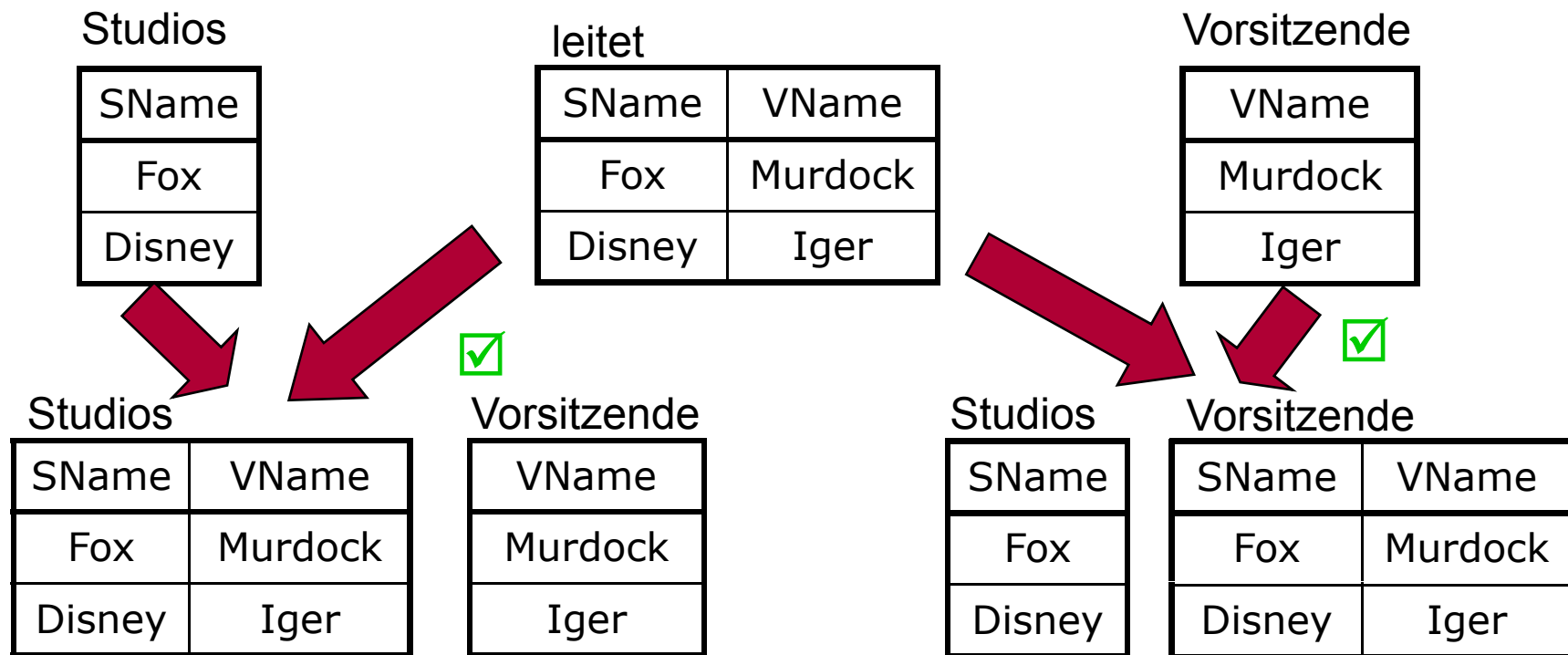
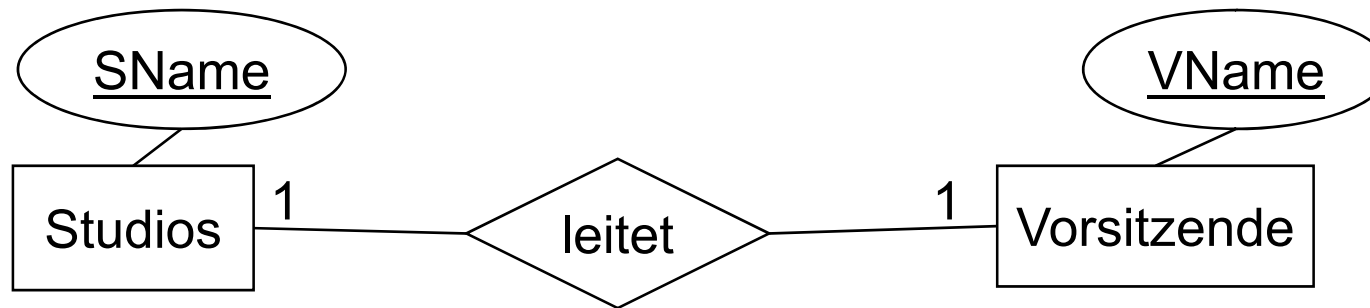
Filme

Titel	Jahr	Länge	Typ	studioName
Basic Instinct	1992	127	Farbe	Fox
Total Recall	1990	113	Farbe	Disney
Dead Man	1995	121	s/w	Paramount

# Zusammenlegen von Relationen

## 1:1-Relationships

24

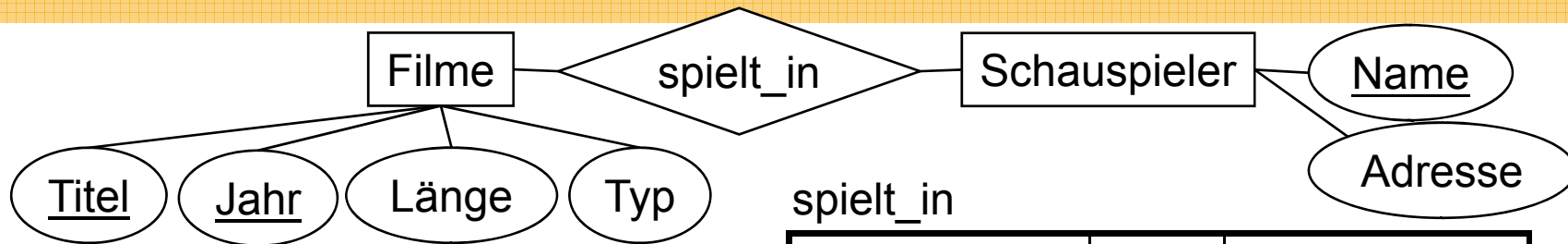




# Zusammenlegen von Relationen

## n:m-Relationships

25



Filme

Titel	Jahr	Länge	Typ
Basic Instinct	1992	127	Farbe
Total Recall	1990	113	Farbe
Dead Man	1995	121	s/w

spielt\_in

Titel	Jahr	Name
Total Recall	1990	Sharon Stone
Basic Instinct	1992	Sharon Stone
Total Recall	1990	Arnold
Dead Man	1995	Johnny Depp

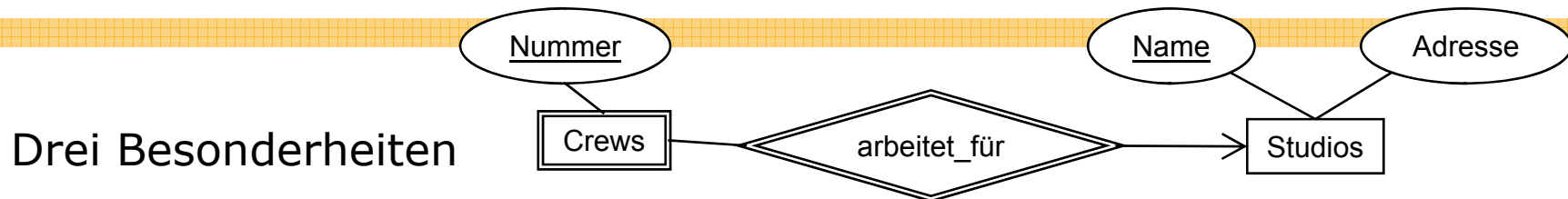
Filme

Titel	Jahr	Länge	Typ	Name
Total Recall	1990	113	Farbe	Sharon Stone
Basic Instinct	1992	127	Farbe	Sharon Stone
Total Recall	1990	113	Farbe	Arnold
Dead Man	1995	121	s/w	Johnny Depp



# Schwache Entitytypen

26

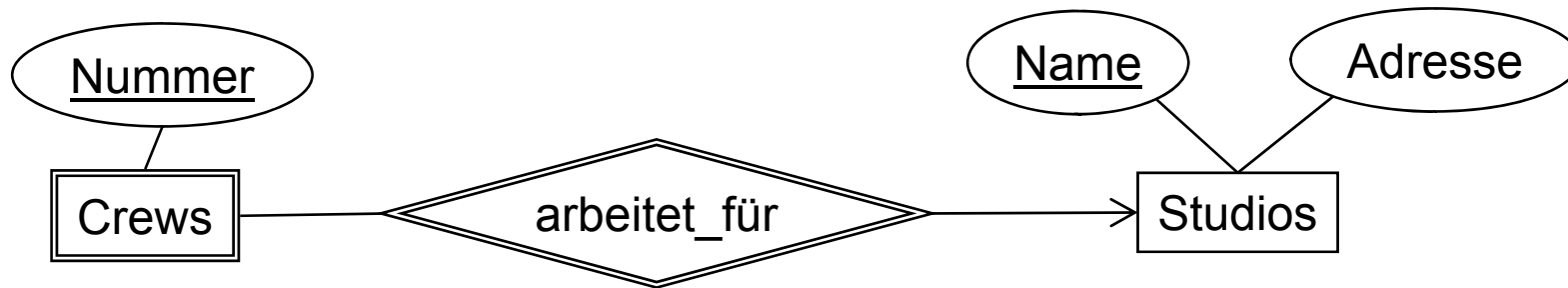


## Drei Besonderheiten

- Die Relation eines *schwachen Entitytypen*  $S$  muss nicht nur die eigenen Attribute, sondern auch die Schlüsselattribute aller Entitytypen, die über unterstützende Relationshiptypen erreicht werden, enthalten.
- Alle Relationen für *Relationshiptypen*, die  $S$  in Beziehung mit anderen Entitytypen setzen, müssen ebenfalls alle diese Attribute enthalten.
- Ein *unterstützender Relationshiptyp* muss hingegen gar nicht durch eine Relation abgebildet werden.
  - Begründung wie eben: 1:n
  - Ausnahme: Der unterstützende Relationshiptyp hat wiederum selbst Attribute.

# Schwache Entitytypen

27



- Studios(Name, Adresse)
- Crews(Nummer, Name)
- ~~arbeitet\_für(Nummer, Name1, Name2)~~

sind gleich

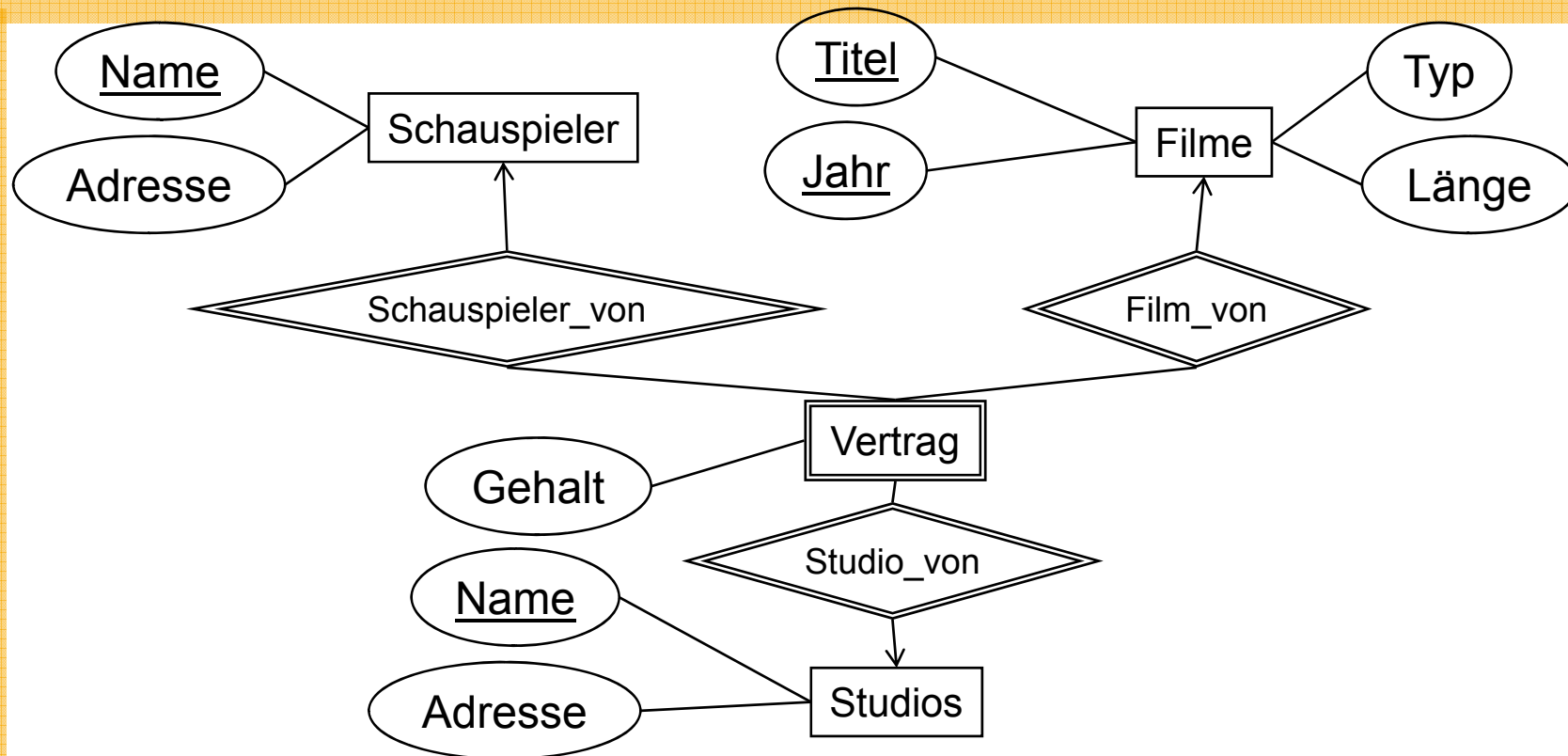
# Schema-Teilmengen

28

- Beispiel
  - Personen(Name, SSN)
  - Steuerzahler(Name, SSN, Betrag)
- Schema von Personen ist Teilmenge des Schemas von Steuerzahler.
- Aber: Instanzen können sich unterscheiden
  - Steuerzahler  $\subseteq$  Personen
  
- Beispiel
  - Schauspieler(Name, Adresse)
  - Studios(Name, Adresse)
- Schemata sind sogar identisch, aber Instanzen grundverschieden.

# Schwache Entitytypen

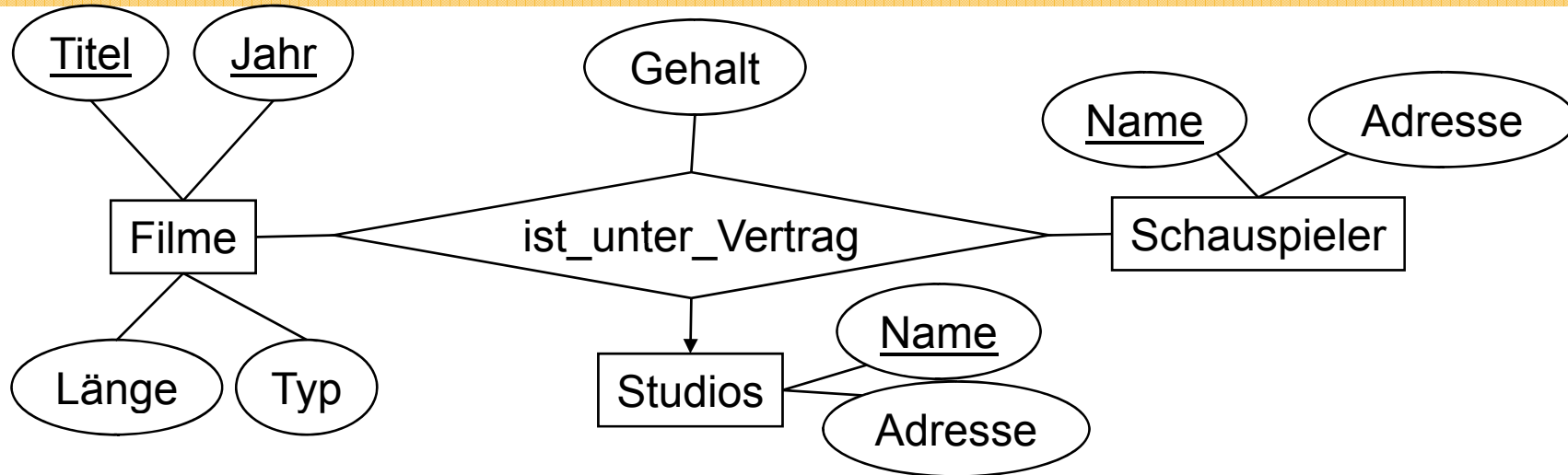
29



- Studios(Name, Adresse)
- Schauspieler(Name, Adresse)
- Filme(Titel, Jahr, Typ, Länge)
- Vertrag(SchauspielerName, StudioName, Titel, Jahr, Gehalt)

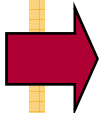
# Schwache Entitytypen

30



- Studios(Name, Adresse)
- Schauspieler(Name, Adresse)
- Filme(Titel, Jahr, Typ, Länge)
- ist\_unter\_Vertrag(SchauspielerName, StudioName, Titel, Jahr, Gehalt)
  
- Was fällt auf?

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Funktionale Abhängigkeiten (FDs)
- Ableitungsregeln für FDs
- Normalformen

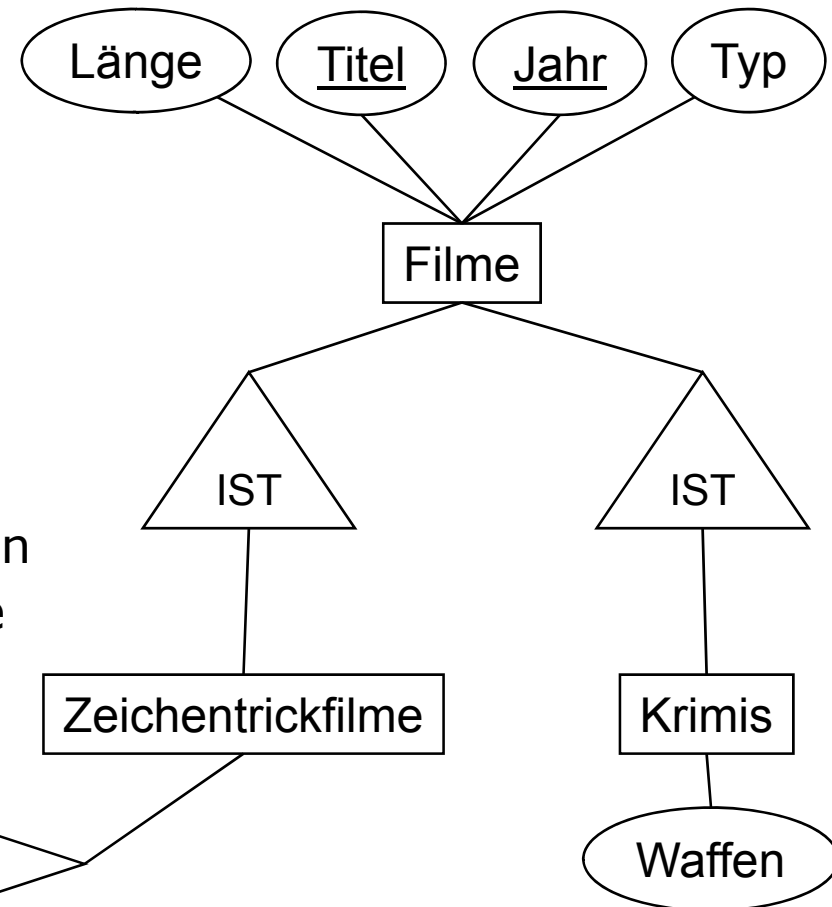


# Wdh. Spezialisierung

32

## ■ Annahmen

- Es gibt ein Wurzel-Entitytyp der IST-Hierarchie.
- Der Wurzel-Entitytyp hat einen Schlüssel, der alle Entities der gesamten Hierarchie identifiziert.
- Ein Entity kann aus mehreren Komponenten der Hierarchie bestehen.



zu Schauspielern



# Drei Strategien

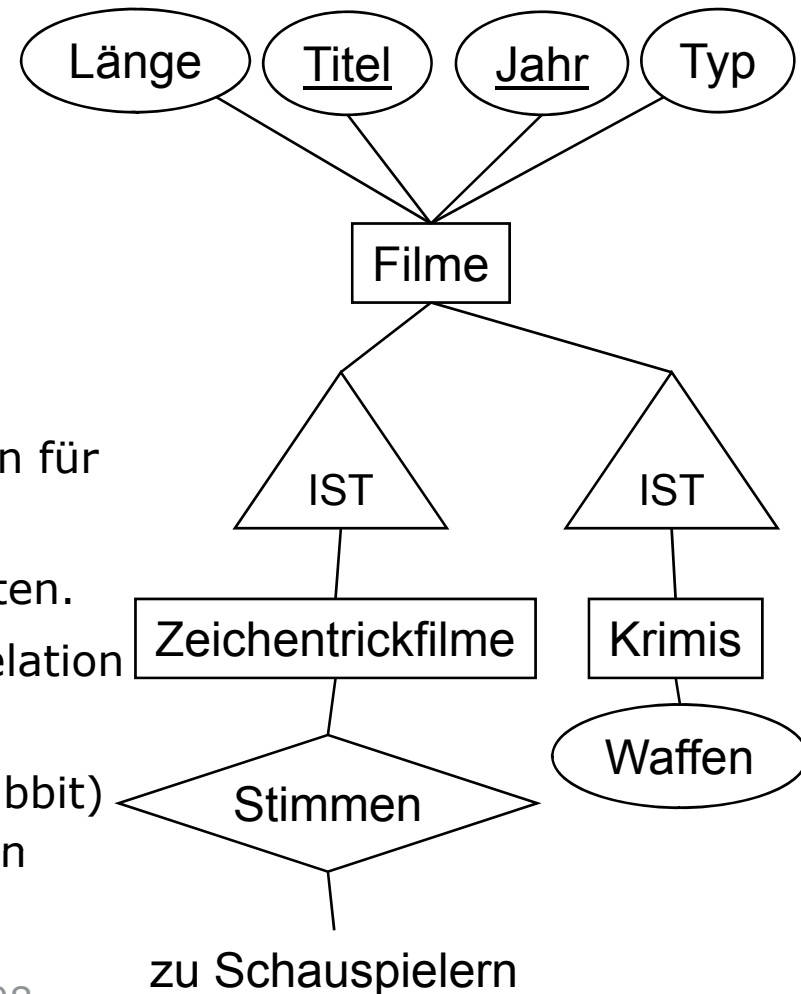
33

- Im ER-Stil
  - Für jeden Entitytypen  $E$  der Hierarchie erzeuge eine Relation mit den Schlüsselattributen des Wurzel-Entitytypen und den Attributen von  $E$ .
- Objekt-orientierter Stil
  - Ein Entity gehört zu genau einer Klasse.
  - Für jeden möglichen Teilbaum der Hierarchie, der auch die Wurzel enthält, erzeuge eine Relation mit allen Attributen der beteiligten Entitytypen.
- Mit Null-Werten
  - Erzeuge eine einzige Relation für die gesamte Hierarchie. Ein Entity wird durch ein Tupel repräsentiert mit Null-Werten für Attribute, die der Entity nicht besitzt.

# ER-Stil

34

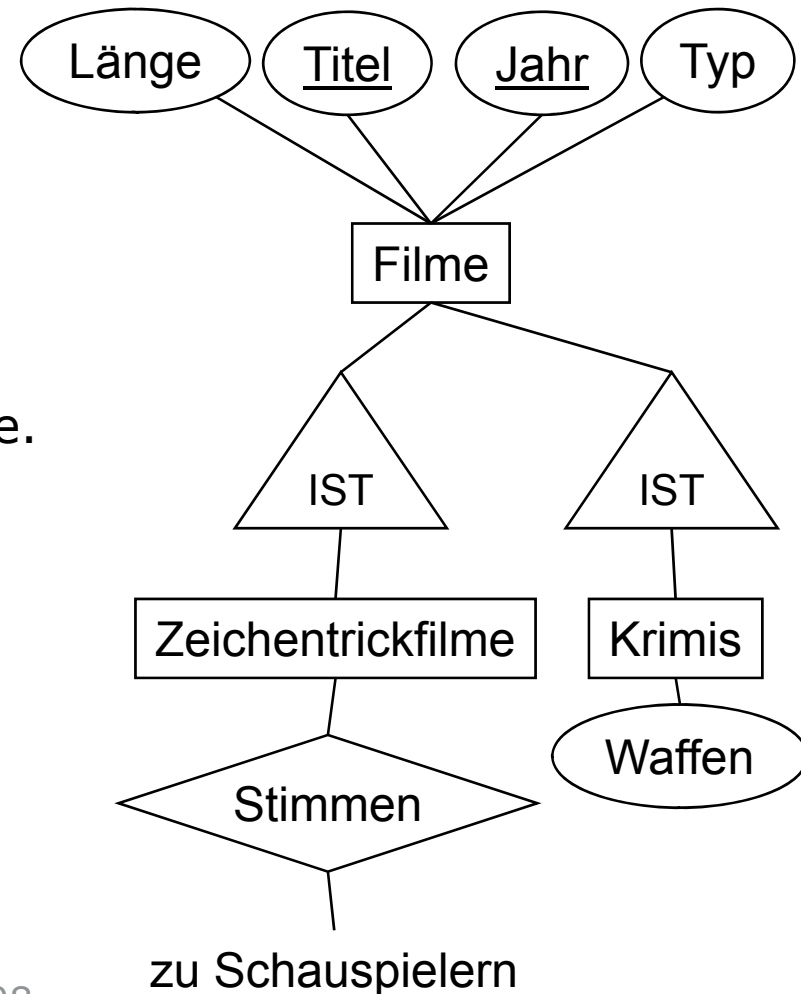
- Filme(Titel, Jahr, Länge,Typ)
- Krimis(Titel, Jahr, Waffen)
- Zeichentrickfilme(Titel, Jahr)
- Anmerkungen
  - Die IST-Relationship erhält keine Relation.
  - Geerbte Schlüsselattribute werden für weitere Beziehungen verwendet.
  - Es gibt vier verschiedene Filmsorten.
  - Jeder Film hat ein Tupel in der Relation Filme.
  - Ein konkreter Film (z.B. Roger Rabbit) kann Tupel in allen drei Relationen haben.



# ER-Stil – Feinheiten

35

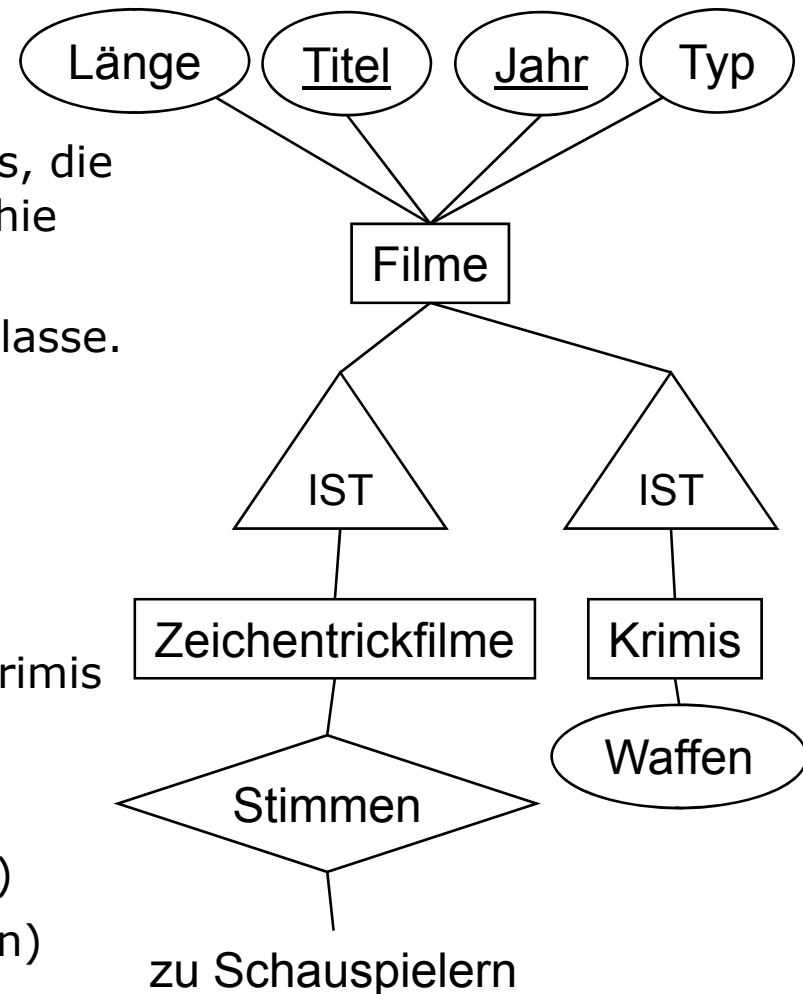
- Filme(Titel, Jahr, Länge, Typ)
- Krimis(Titel, Jahr, Waffen)
- Zeichentrickfilme(Titel, Jahr)
- Stimmen(Titel, Jahr, Name)
- Schema von Zeichentrickfilm ist Teilmenge des Schemas von Filme.
  - Kann man es weglassen?
  - Stumme Zeichentrickfilme!



# Objekt-orientierter Stil

36

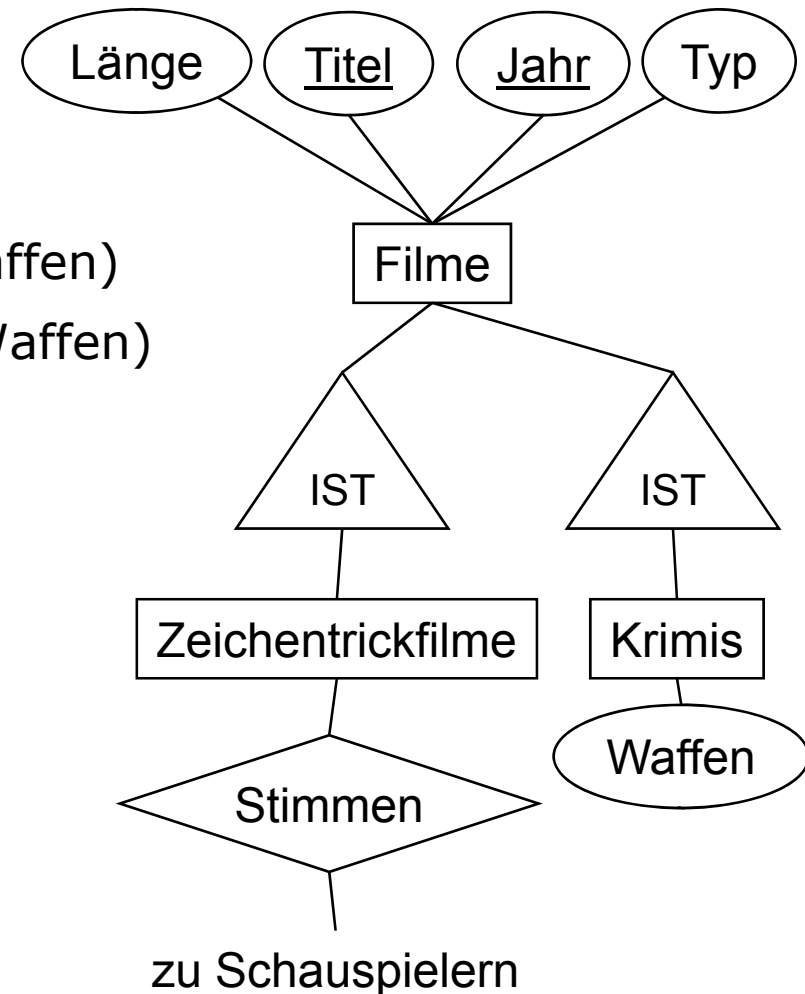
- Erzeuge Relation für jeden Teilbaum.
- Diese Relation repräsentiert die Entities, die genau diese Komponenten der Hierarchie besitzen.
  - Objekte gehören zu genau einer Klasse.
- Vier Teilbäume
  - Nur Filme
  - Filme und Zeichentrickfilme
  - Filme und Krimis
  - Filme und Zeichentrickfilme und Krimis
- Filme(Titel, Jahr, Länge, Typ)
- FilmeZ(Titel, Jahr, Länge, Typ)
- FilmeK(Titel, Jahr, Länge, Typ, Waffen)
- FilmeZK(Titel, Jahr, Länge, Typ, Waffen)



# Objekt-orientierter Stil – Feinheiten

37

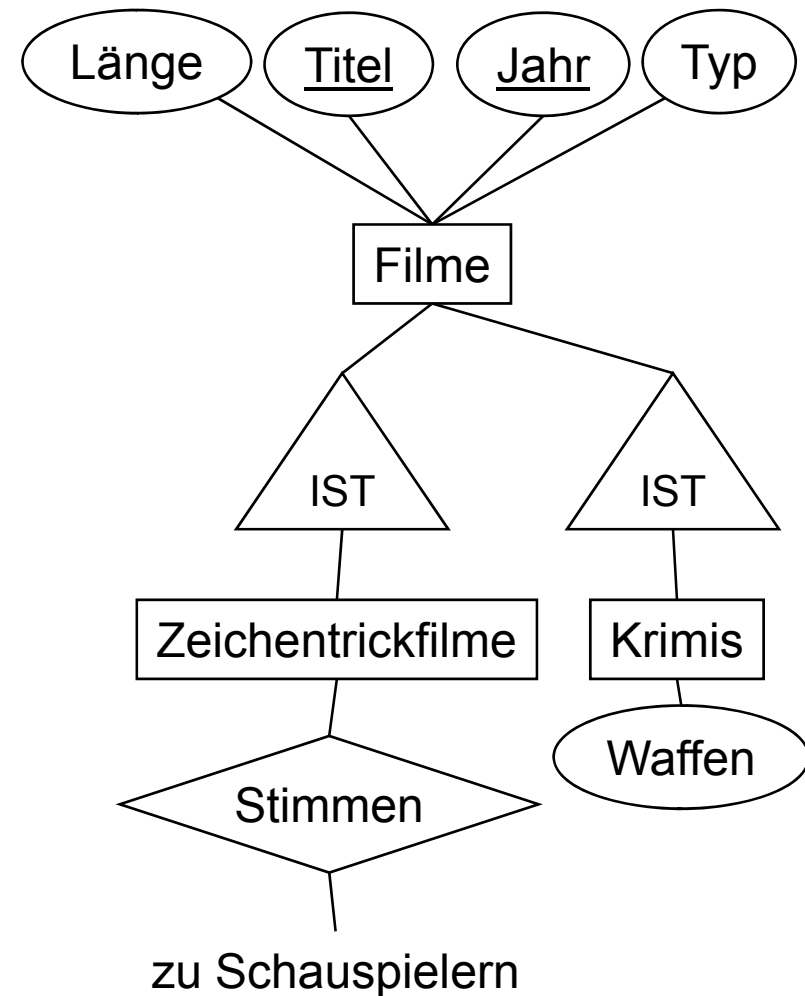
- Filme(Titel, Jahr, Länge, Typ)
- FilmeZ(Titel, Jahr, Länge, Typ)
- FilmeK(Titel, Jahr, Länge, Typ, Waffen)
- FilmeZK(Titel, Jahr, Länge, Typ, Waffen)
- Kann man Filme und FilmeZ kombinieren?
- Kann man FilmeK und FilmeZK kombinieren?
- Wie viele Relationen für Stimmen benötigt man?
  - Stimmen(Titel, Jahr, Name)



# Mit Nullwerten

38

- Eine einzige Relation mit allen Attributen.
- Filme(Titel, Jahr, Länge, Typ, Waffen)
- Nicht-Krimis haben NULL-Wert als Attributwert für Waffen.



# Vergleich der drei Stile

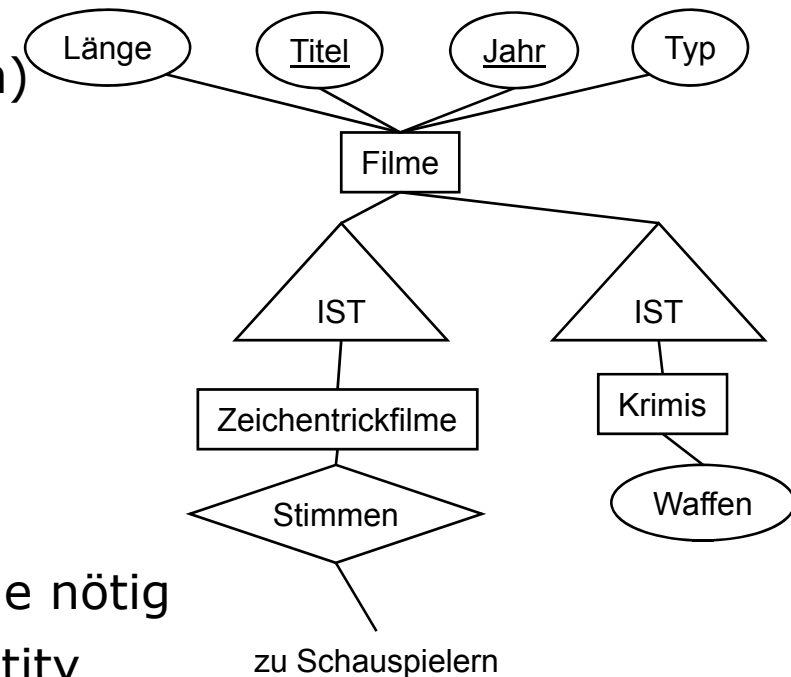
39

Anzahl an Relationen (bei  $n$  Entitytypen)

- Null-Stil: Nur eine Relation
- ER-Stil:  $n$  Relationen
- OO-Stil:  $2^{n-1}$  Relationen !

Speicherbedarf

- OO-Stil: Minimaler Speicherbedarf
  - Nur ein Tupel pro Entity
  - Jeweils nur so viele Attribute wie nötig
- Null-Stil: Auch nur ein Tupel pro Entity
  - Aber: Lange Tupel mit möglicherweise vielen Null-Werten
- ER-Stil: Viele Tupel pro Entity
  - Aber nur Schlüsselattribute werden wiederholt.

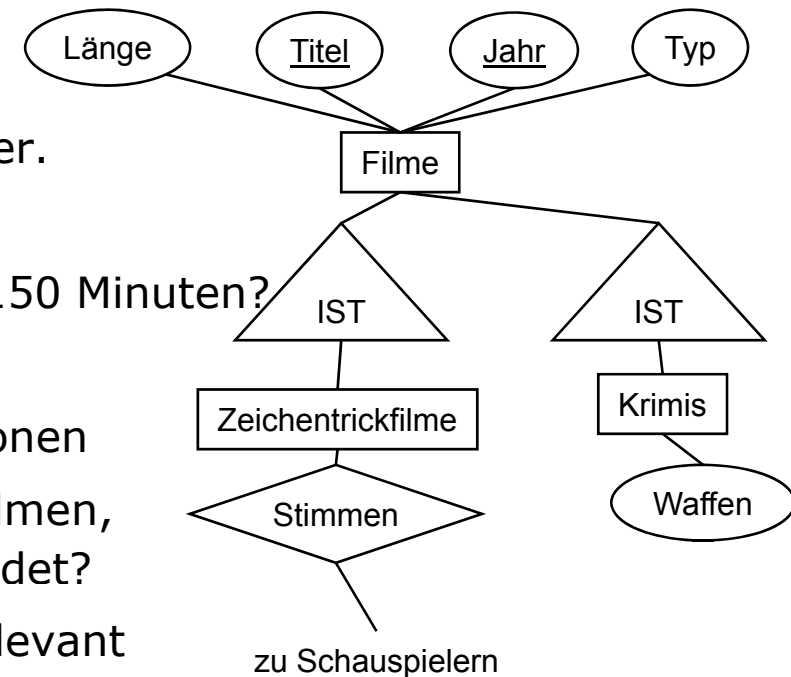


# Vergleich der drei Stile

40

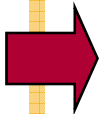
## Anfragebearbeitung

- Anfragen über viele Relationen sind teuer.
- ⇒ Null-Werte im Vorteil
- Welche Filme aus 1999 sind länger als 150 Minuten?
  - ER-Stil: Antwort direkt möglich
  - OO-Stil: Anfrage an alle vier Relationen
- Welche Waffen wurden in Zeichentrickfilmen, die länger als 150 Minuten sind, verwendet?
  - ER-Stil: Alle drei Relationen sind relevant
    - ◇ Filme für die Länge
    - ◇ Zeichentrickfilme für die Tatsache, dass es ein Zeichentrickfilm ist
    - ◇ Krimis für die Waffe
  - OO-Stil: Anfrage nur an FilmeZK()





- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Funktionale Abhängigkeiten (FDs)
- Ableitungsregeln für FDs
- Normalformen



# Motivation

42

- Bisher: Direkte Übersetzung von ER-Diagrammen in das relationale Modell
- Verbesserungen sind möglich (Verfeinern des logischen Entwurfs)
  - Aufgrund bestimmter Nebenbedingungen
  - Insbesondere: Aufgrund von funktionalen Abhängigkeiten
    - ◇ FDs: Functional dependencies
- Vermeidung von Redundanzen: Aufspalten von Relationenschemata, ohne gleichzeitig
  - semantische Informationen zu verlieren,
  - die Möglichkeit zur Rekonstruktion der Relationen zu verlieren
- Redundanzvermeidung durch Normalformen (nächster Abschnitt)

# Definition – Funktionale Abhängigkeit

43

„ $X \rightarrow A$ “ ist eine Aussage über eine Relation  $R$ , dass immer wenn zwei Tupel in den Werten der Attributmengens  $X$  übereinstimmen sie auch im Attributwert für  $A$  übereinstimmen.

- Beispiel

- Titel, Jahr  $\rightarrow$  Länge

- Notation

- ...,  $X, Y, Z$  sind Attributmengen
- $A, B, C, \dots$  sind Attribute
- $X \rightarrow A$ : „ $X$  bestimmt  $A$  funktional“.
- Kurzform:  $XYZ$  statt  $\{X, Y, Z\}$
- Kurzform: Falls
  - ◇  $X \rightarrow A, X \rightarrow B, X \rightarrow C$
  - ◇ Schreiben wir auch  $X \rightarrow ABC$  oder auch  $X \rightarrow Y$

# Beispiel

44

Titel	Jahr	Länge	Typ	StudioName	SchauspName
Total Recall	1990	113	Farbe	Fox	Sharon Stone
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone
Total Recall	1990	113	Farbe	Fox	Arnold
Dead Man	1995	121	s/w	Paramount	Johnny Depp

- Titel, Jahr → Länge
- Titel, Jahr → Typ
- Titel, Jahr → StudioName
- Titel, Jahr → Länge, Typ, StudioName
- Wenn zwei Tupel den gleichen Titel und das gleiche Jahr haben, dann haben sie auch gleiche Länge, gleichen Typ und gleichen Studionamen.
  - Klar, denn Titel und Jahr sind Schlüssel: Gegeben Titel und Jahr haben wir einen eindeutigen Film, der wohl auch eine eindeutige Länge und Typ hat.
  - Wegen 1:n Beziehung zwischen Studios und Filmen ist auch zu erwarten, dass das Studio eindeutig ist.
- Aber Titel, Jahr → SchauspName ist falsch! Warum?

# Schema vs. Instanz

45

Titel	Jahr	Länge	Typ	StudioName	SchauspName
Total Recall	1990	113	Farbe	Fox	Sharon Stone
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone
Total Recall	1990	113	Farbe	Fox	Arnold
Dead Man	1995	121	s/w	Paramount	Johnny Depp

- FDs sind Aussagen über das Schema, nicht die Instanz!
- Titel → Typ scheint zu gelten
  - Aber nur zufällig bei dieser Instanz
  - Wenn zwei Filme im Titel übereinstimmen, stimmen sie auch im Typ überein.
  - Gegenbeispiel: King Kong von 1924 vs. King Kong von 2005.

# Schlüssel als Spezialfall einer FD

46

- Eine Menge aus einem oder mehr Attributen  $\{A_1, A_2, \dots, A_n\}$  ist Schlüssel der Relation  $R$ , falls gilt:
  - Die Attribute bestimmen alle anderen Attribute funktional.
    - ◇ Anmerkung: Relationen sind Mengen, es kann also keine zwei völlig identischen Tupel geben.
  - Keine echte Teilmenge von  $\{A_1, A_2, \dots, A_n\}$  bestimmt alle anderen Attribute funktional.
    - ◇ Anmerkung: Ein Schlüssel muss also minimal sein.
- Ziel des Datenbankentwurfs: Normalisierung
  - Alle gegebenen FDs in „Schlüsselabhängigkeiten“ umformen, ohne dabei semantische Information zu verlieren.
  - Umformung durch Dekomposition von Relationen
  - Später, denn zunächst: Entdeckung aller FDs

# Schlüssel als Spezialfall einer FD

47

Titel	Jahr	Länge	Typ	StudioName	SchauspName
Total Recall	1990	113	Farbe	Fox	Sharon Stone
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone
Total Recall	1990	113	Farbe	Fox	Arnold
Dead Man	1995	121	s/w	Paramount	Johnny Depp

- {Titel, Jahr, SchauspName} ist ein Schlüssel.
- {Titel, Jahr} bestimmen Länge, Typ und StudioName funktional.
- Deshalb können keine zwei Tupel gleiche Werte für Titel, Jahr und SchauspName haben. Sie wären insgesamt identisch.
- Teilmengen?
  - {Titel, Jahr} bestimmen SchauspName nicht funktional
  - {Jahr, SchauspName} bestimmen Titel nicht funktional
  - {Titel, SchauspName} bestimmen Jahr nicht funktional
  - ◇ Beispiele?

# Schlüssel und Superschlüssel

48

- Eine Relation kann mehr als einen Schlüssel besitzen.
  - Wahl eines der Schlüssel als „Primärschlüssel“
- Eine Attributmenge, die einen Schlüssel enthält ist ein Superschlüssel.
  - {Titel, Jahr, SchauspName} ist ein Schlüssel und ein Superschlüssel
  - {Titel, Jahr, Länge, SchauspName} ist ein Superschlüssel
    - ◇ Nicht minimal
- Minimal vs. kleinster
  - Minimaler Schlüssel: Kein Attribut darf fehlen
  - Kleinster Schlüssel: Schlüssel mit wenigsten Attributen
- Alternative Begriffe:
  - Schlüssel (=Superschlüssel) und Schlüsselkandidat (=Schlüssel)



# Wo kommen FDs her?

49

- Einfach den Schlüssel  $K$  deklarieren.
  - Dann gelten einzig die FDs  $K \rightarrow A$  für jedes Attribut  $A$ .
- FDs deklarieren.
  - Dann systematisch Schlüssel ableiten.
- FDs aus der Physik:
  - Zwei Kurse können nicht zur gleichen Zeit im gleichen Raum stattfinden.
  - Zeit, Raum  $\rightarrow$  Kurs
- FDs aus dem ER-Diagramm
  - Schlüsselattribute
  - 1:n Beziehungen

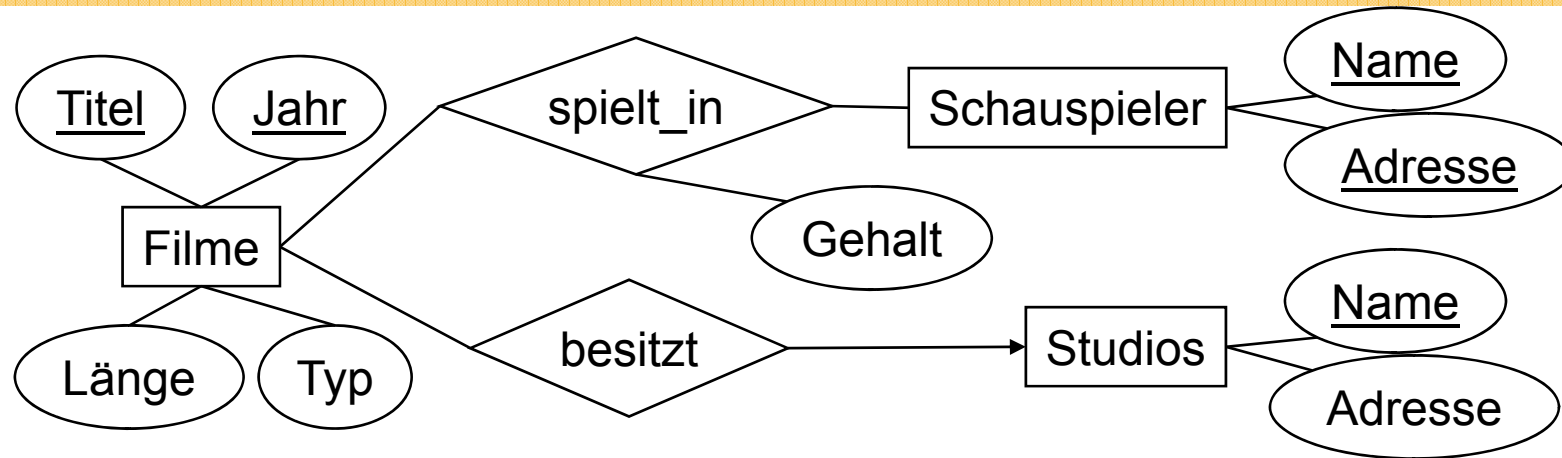
# Schlüssel aus ER-Diagrammen

50

- Falls die Relation von einem Entitytypen stammt
  - Der Schlüssel der Relation besteht aus den Schlüsselattributen des Entitytypen.
- Falls die Relation von einem Relationshiptypen stammt
  - Falls die Beziehung  $m:n$  ist, besteht der Schlüssel aus den Schlüsselattributen der verbundenen Entitytypen.
  - Falls die Beziehung  $1:n$  ist, besteht der Schlüssel aus den Schlüsselattributen des Entitytypen der  $n$ -Seite.
  - Falls die Beziehung  $1:1$  ist, besteht der Schlüssel aus den Schlüsselattributen eines der beiden beteiligten Entitytypen (egal welcher).
- Bei  $n$ -ären Relationshiptypen
  - Lage ist komplizierter
  - 1-Seite muss nie am Schlüssel beteiligt sein.

# Schlüssel aus ER-Diagrammen

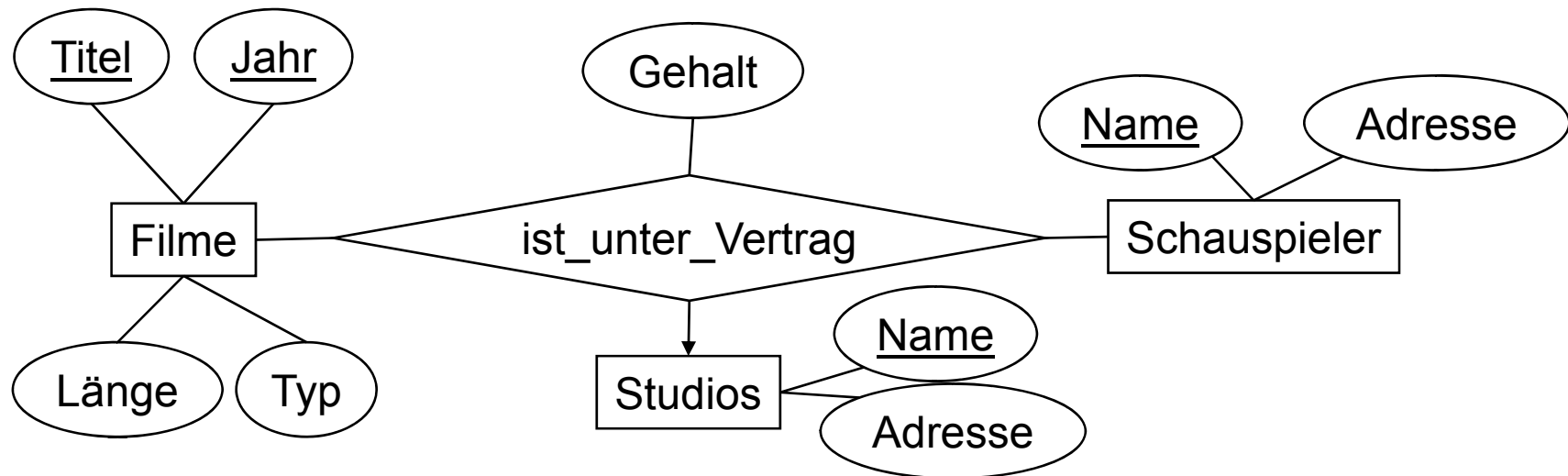
51



- Filme(Titel, Jahr, Länge, Typ)
- Schauspieler(Name, Adresse)
- Studios(Name, Adresse)
- besitzt(Titel, Jahr, Name)
- spielt\_in(Titel, Jahr, Name, Adresse, Gehalt)

# Schlüssel aus ER-Diagrammen: n-äre Relationshiptypen

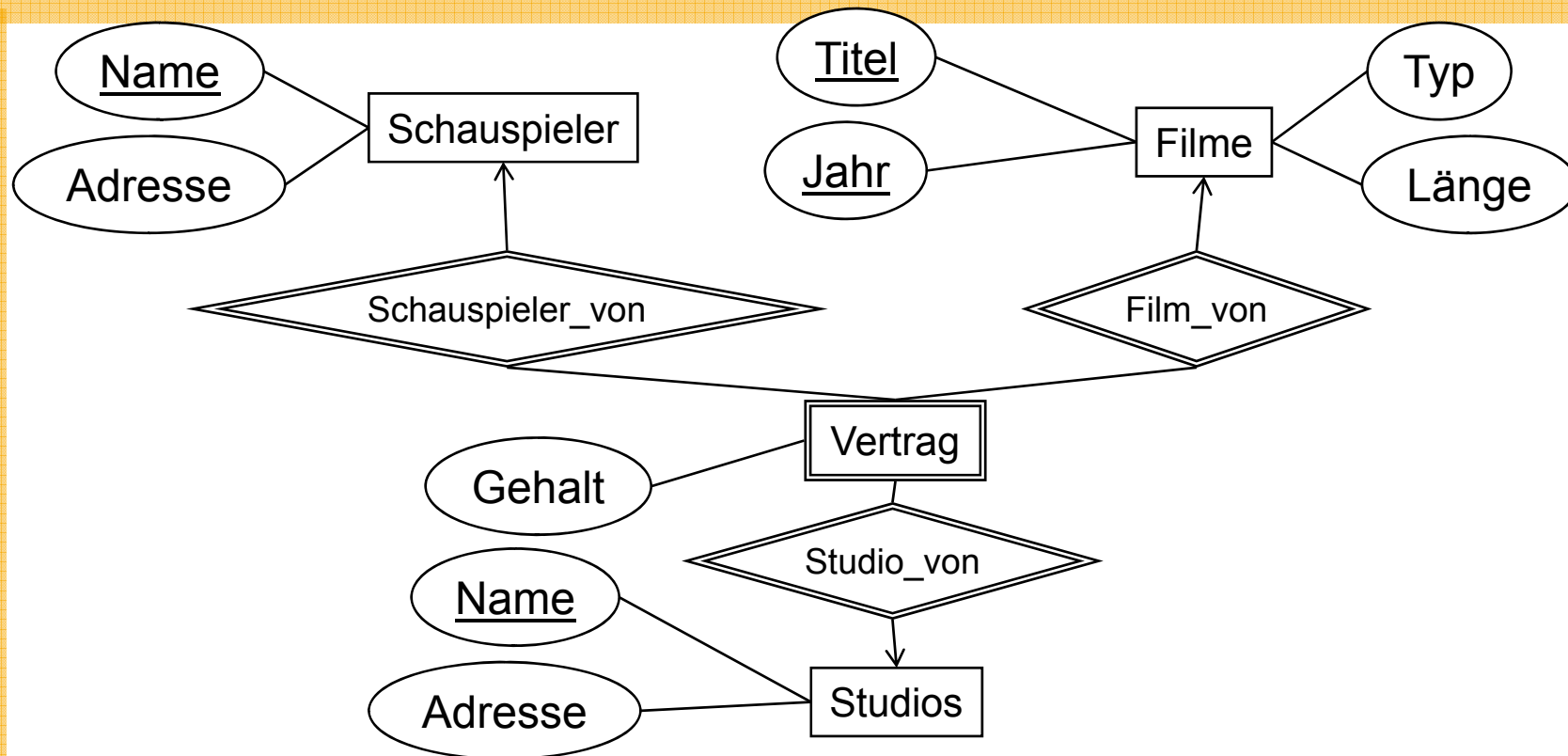
52



- Studios (Name, Adresse)
- Schauspieler (Name, Adresse)
- Filme (Titel, Jahr, Typ, Länge)
- ist\_unter\_Vertrag (SchauspielerName, StudioName, Titel, Jahr, Gehalt)

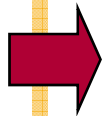
# Schlüssel aus ER-Diagrammen: n-äre Relationshiptypen

53



- Studios (Name, Adresse)
- Schauspieler (Name, Adresse)
- Filme (Titel, Jahr, Typ, Länge)
- Vertrag (SchauspielerName, StudioName, Titel, Jahr, Gehalt)

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Funktionale Abhängigkeiten (FDs)
- Ableitungsregeln für FDs
- Normalformen



# Motivation

55

- Gegeben eine Menge von FDs, kann man eventuell weitere FDs ableiten.
  - Ziel des Datenbankentwurfs:
    - ◇ Alle gegebenen FDs in „Schlüsselabhängigkeiten“ umformen, ohne dabei semantische Information zu verlieren.
    - ◇ Umformung durch Dekomposition von Relationen

# Ableitung von FDs – Beispiel

56

- Es gelte  $A \rightarrow B$  und  $B \rightarrow C$
- Dann gilt auch:  $A \rightarrow C$
- Beweis
  - Z.z.: Zwei beliebige Tupel, die in A übereinstimmen, müssen auch in C übereinstimmen.
  - Zwei solche beliebige Tupel:  $(a, b_1, c_1)$  und  $(a, b_2, c_2)$
  - $A \rightarrow B \Rightarrow (a, b, c_1)$  und  $(a, b, c_2)$
  - $B \rightarrow C \Rightarrow (a, b, c)$  und  $(a, b, c)$
  - QED
- Instanz genügt  $A \rightarrow B$  und  $B \rightarrow C$ 
  - Es gilt auch:  $A \rightarrow C$
  - nicht ableitbar:  $C \rightarrow A$ ,  $B \rightarrow A$  oder  $C \rightarrow B$

*r*

<i>A</i>	<i>B</i>	<i>C</i>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>
<i>a</i> <sub>2</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>
<i>a</i> <sub>3</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>
<i>a</i> <sub>4</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>



# FD-Mengen

57

- Zwei Mengen  $S$  und  $T$  an FDs heißen **äquivalent**, falls die Menge der gültigen Instanzen unter  $S$  die gleiche wie unter  $T$  ist.
- Eine Menge  $S$  an FDs **folgt aus** einer Menge  $T$  an FDs, falls jede unter  $T$  gültige Instanz auch unter  $S$  gültig ist.
- Hüllenbildung:
  - Ableitung aller FDs aus einer gegebenen Menge an FDs
  - Gemäß Ableitungsregeln
  - Auch: *attribute closure, closure*, Attributabschluss

# Dekomposition und Vereinigung

58

## Dekompositionsregel

- $A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$
- $\Rightarrow A_1, A_2, \dots, A_n \rightarrow B_i$  für  $i=1, \dots, m$

## Vereinigungsregel

- $A_1, A_2, \dots, A_n \rightarrow B_i$  für  $i=1, \dots, m$
- $\Rightarrow A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$

## Beispiel

- Titel, Jahr  $\rightarrow$  Länge
- Titel, Jahr  $\rightarrow$  Typ
- Titel, Jahr  $\rightarrow$  StudioName
- $\Leftrightarrow$  Titel, Jahr  $\rightarrow$  Länge, Typ, StudioName

Dekomposition funktioniert nur rechts!

- Titel, Jahr  $\rightarrow$  Länge
  - $\not\Rightarrow$  Jahr  $\rightarrow$  Länge
  - $\not\Rightarrow$  Titel  $\rightarrow$  Länge

# Triviale FDs

59

- Trivial: Attribute rechts sind Teilmenge der Attribute links
  - Titel, Jahr  $\rightarrow$  Titel
  - Es gilt immer jede triviale FD:
    - ◇ „Zwei Tupel, die in einer Menge von Attributen übereinstimmen, stimmen auch in einem dieser Attribute überein.“
- Nicht-trivial: Wenigstens ein Attribut rechts kommt links nicht vor.
  - Titel, Jahr  $\rightarrow$  Jahr, Länge
- Völlig nicht-trivial: Die Attribute links und rechts sind disjunkt.
  - Titel, Jahr  $\rightarrow$  Länge
  - Im Weiteren interessieren uns nur diese.
- Triviale-Abhängigkeitsregel
  - $A_1, A_2, \dots, A_n \rightarrow A_{i_1}, \dots, A_{i_k}, B_1, B_2, \dots, B_m$
  - $\Leftrightarrow A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$

# Hülle

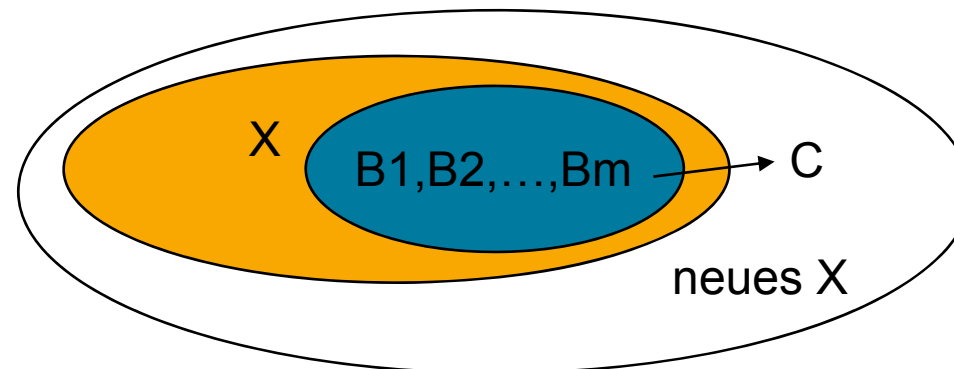
60

- Gegeben eine Menge von Attributen  $A_1, A_2, \dots, A_n$  und eine Menge  $S$  von FDs.
- Die **Hülle** von  $A_1, A_2, \dots, A_n$  unter  $S$  ist die Menge  $B$  aller Attribute für die gilt, dass für jede unter  $S$  gültige Relation auch  $A_1, A_2, \dots, A_n \rightarrow B$  gilt.
  - Menge der funktional ableitbaren Attribute
  - D.h.  $A_1, A_2, \dots, A_n \rightarrow B$  folgt aus den FDs in  $S$ .
- Notation: Hülle von  $A_1, A_2, \dots, A_n$  ist  $\{A_1, A_2, \dots, A_n\}^+$ .
- Es gilt z.B.  $A_i \in \{A_1, A_2, \dots, A_n\}^+$  für  $i=1, \dots, n$ 
  - trivialerweise

# Berechnung der Hülle

61

1. Sei  $X$  die Menge der Attribute, die später die Hülle wird. Initialisiere  $X$  mit  $\{A_1, A_2, \dots, A_n\}$ .
2. Suche wiederholt nach solchen FDs  $B_1, B_2, \dots, B_m \rightarrow C$ , dass  $B_1, B_2, \dots, B_m \in X$  aber  $C \notin X$ . Füge  $C$  zu  $X$  hinzu.
3. Wiederhole 2. bis keine Attribute mehr gefunden werden
  - Terminierung:  $X$  wächst nur, und Attributmenge ist endlich.
4.  $X$  ist schließlich die Hülle, also  $\{A_1, A_2, \dots, A_n\}^+ = X$ .



## Berechnung der Hülle – Beispiel

62

- Relation mit Attributen A, B, C, D, E, F.
- Gegeben FDs
  1.  $AB \rightarrow C$
  2.  $BC \rightarrow AD$
  3.  $D \rightarrow E$
  4.  $CF \rightarrow B$
- Gesucht: Hülle von  $\{A, B\}$ , also  $\{A, B\}^+$ .
  - FD 1:  $X = \{A, B, C\}$
  - FD 2:  $X = \{A, B, C, D\}$
  - FD 3:  $X = \{A, B, C, D, E\}$       ( =  $\{A, B\}^+$  )

# Membershipproblem

63

- Kann eine bestimmte FD  $X \rightarrow Y$  aus der gegebenen FD Menge abgeleitet werden?
- Berechne Hülle von  $X$  und teste ob  $Y$  darin enthalten ist.
- Beispiel:
  - $AB \rightarrow C$  und  $BC \rightarrow AD$  und  $D \rightarrow E$  und  $CF \rightarrow B$
  - Kann  $AB \rightarrow D$  abgeleitet werden?
    - ◇  $\{AB\}^+ = \{A, B, C, D, E\}$
    - ◇  $D \in \{A, B, C, D, E\}$ , also JA!
  - Kann  $D \rightarrow A$  abgeleitet werden?
    - ◇  $\{D\}^+ = \{D, E\}$
    - ◇  $A \notin \{D, E\}$ , also NEIN!

# Analyse des Algorithmus zur Hüllenbildung

64

- Nur Beweisidee
  - Korrektheit: Es werden keine ungültigen FDs erzeugt.
    - ◇ Induktion über Anzahl der Operationen
    - ◇ Transitivität bzw. Argumentation über die Tupel.
  - Vollständigkeit: Es werden alle gültigen FDs erzeugt.
    - ◇ Annahme des Gegenteils, d.h. es gibt eine FD  $X \rightarrow Y$ , die nicht gefunden wird.
    - ◇ Konstruktion einer Instanz, die für FDs, aber nicht für  $X \rightarrow Y$  gültig ist.



# Transitivitätsregel

65

- Falls  $A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$  und  $B_1, B_2, \dots, B_m \rightarrow C_1, C_2, \dots, C_k$
- $\Rightarrow A_1, A_2, \dots, A_n \rightarrow C_1, C_2, \dots, C_k$
- Beispiel

Titel	Jahr	Länge	Typ	StudioName	StudioAdresse
Total Recall	1990	113	Farbe	Fox	Hollywood
Basic Instinct	1992	127	Farbe	Disney	Buena Vista
Total Recall	1993	113	Farbe	Fox	Hollywood
Dead Man	1995	121	s/w	Paramount	Buena Vista

- Titel, Jahr  $\rightarrow$  StudioName
  - ◇ gilt wegen n:1 von besitzt-Beziehung
- StudioName  $\rightarrow$  StudioAdresse
  - ◇ gilt wegen Schlüsseleigenschaft von Studioname
- Transitivität: Titel, Jahr  $\rightarrow$  StudioAdresse

# Die „Basis“

66

- Unterscheidung zwischen gegebenen FDs und abgeleiteten FDs
- Wahl welche FDs zur Repräsentation aller FDs verwendet werden.
  - Eine Menge an FDs, aus der alle anderen FDs abgeleitet werden können, heißt Basis.
  - Falls keine echte Teilmenge der Basis wiederum selbst eine Basis ist, ist die Basis minimal.
- Beispiel
  - $R(A, B, C)$ ; jedes Attribut bestimmt funktional die anderen beiden.
  - Welche FDs gelten?
  - $A \rightarrow B, A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B$
  - Abgeleitet:  $AB \rightarrow C, AC \rightarrow B, BC \rightarrow A$
  - Kurzformen:  $A \rightarrow BC, B \rightarrow AC, C \rightarrow AB$
  - Triviale FDs:  $A \rightarrow A, B \rightarrow B, C \rightarrow C$
  - Nicht-triviale FDs:  $AB \rightarrow BC, AC \rightarrow BC, \dots$
  - Minimale Basis:  $\{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B\}$
  - Minimale Basis:  $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

# Armstrong Axiome und weitere Ableitungsregeln

67

- R1 Reflexivität  $X \supseteq Y \Rightarrow X \rightarrow Y$  (insbes.  $X \rightarrow X$ )
    - Triviale FDs
  - R2 Akkumulation  $\{X \rightarrow Y\} \Rightarrow XZ \rightarrow YZ$ 
    - Auch: Augmentation
  - R3 Transitivität  $\{X \rightarrow Y, Y \rightarrow Z\} \Rightarrow X \rightarrow Z$
  - R1-R3 bekannt als *Armstrong-Axiome*
    - Sound and complete
  
  - R4 Dekomposition  $\{X \rightarrow YZ\} \Rightarrow X \rightarrow Y$
  - R5 Vereinigung  $\{X \rightarrow Y, X \rightarrow Z\} \Rightarrow X \rightarrow YZ$
  - R6 Pseudotransitivität  $\{X \rightarrow Y, WY \rightarrow Z\} \Rightarrow WX \rightarrow Z$
- } Herleitung mit Armstrong Axiomen?

# Armstrong Axiome

68

Die Menge der Armstrong-Axiome ist

- Gültig (*sound*)
  - Es wird nichts nicht-ableitbares abgeleitet.
- Vollständig (*complete*)
  - Durch diese Regeln können alle ableitbaren FDs abgeleitet werden.
- Minimal
  - Keine Regel kann weggelassen werden.

# FDs nach Projektionen

69

- Motivation: Normalisierung bricht eine Relation in mehrere Teile.
- Gegeben eine Relation  $R$  mit FD-Menge  $F$ . Sei  $S$  das Ergebnis nach Entfernung einiger Attribute aus  $R$  („Projektion“).
- Welche FDs gelten noch für  $S$ ?
  - Alle FDs, die aus  $F$  folgen,
  - die nur Attribute aus  $S$  verwenden.
- Beispiel:  $R(A, B, C, D)$ 
  - FDs:  $\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
  - Projektion von  $B$ :  $S(A, C, D)$
- Algorithmus: Berechne Hülle jeder Teilmenge
  - Trick 1: Hülle der leeren und Hülle der Menge aller Attribute muss nicht gebildet werden.
  - Trick 2: Falls die Hülle von  $X$  bereits alle Attribute enthält, müssen die Supermengen von  $X$  nicht mehr geprüft werden.
    - ◇ Deshalb: Beginnen mit kleinsten Teilmengen

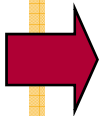
# FDs nach Projektionen

70

- Beispiel:  $R(A, B, C, D)$ 
  - FDs:  $\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
  - Projektion von B:  $S(A, C, D)$
- Sei  $\{X\}^+$  die Hülle der Teilmenge  $X$ .
- Dann gelten FDs  $X \rightarrow E$  für jedes  $E \in \{X\}^+$  und  $E \in S$  und  $E \notin X$ .
- $\{A\}^+ = \{A, B, C, D\}$ 
  - $A \rightarrow C$  und  $A \rightarrow D$
  - $A \rightarrow B$  stimmt zwar auch, aber  $B$  nicht in  $S$ .
  - Enthält bereits alle Attribute aus  $S$ , deshalb werden Supermengen nicht berücksichtigt.
- $\{C\}^+ = \{C, D\}$ 
  - $C \rightarrow D$
- $\{D\}^+ = \{D\}$
- $\{C, D\}^+ = \{C, D\}$
- Ergebnis:  $A \rightarrow C, \del{A \rightarrow B} und  $C \rightarrow D$$

71

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Funktionale Abhängigkeiten (FDs)
- Ableitungsregeln für FDs
- Normalformen



# Schema Design – Überblick

72

1. Anomalien durch schlechtes Design
2. Dekomposition (Zerlegung) von Relationen
3. Boyce-Codd-Normalform (BCNF)
4. Zerlegung zur Erreichung der BCNF
5. Andere Normalformen
  - Insbesondere 3NF



# Anomalien

73

Titel	Jahr	Länge	Typ	StudioName	SchauspName	SchauspAdresse
Total Recall	1990	113	Farbe	Fox	Sharon Stone	Hollywood
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone	Hollywood
Total Recall	1990	113	Farbe	Fox	Arnold	Sacramento
Dead Man	1995	121	s/w	Paramount	Johnny Depp	Paris

- Redundanz
  - Länge und Typ eines Films sind mehrfach dargestellt.
  - Unnötige Speicherplatzverschwendung
- Update Anomalien
  - Falls Total Recall doch 114 Minuten lang ist, muss man dies an mehreren Stellen ändern.
  - Durch Normalisierung kann dies verhindert werden.
- Insert Anomalien
  - Neues Tupel: (Sleepy Hollow, 1999, 105, Farbe, Fox, Johnny Depp, Dallas)
- Delete Anomalien
  - Bei Löschen gehen mehr Informationen verloren.
  - Falls Johnny Depp als letzter Schauspieler aus dem Film entfernt würde, gingen auch die Filmdaten verloren.

# Dekomposition von Relationen

74

- Elimination der Anomalien durch Dekomposition der betroffenen Relationen
- Dekomposition
  - Aufteilung der Attribute in zwei Relationen
  - Erzeugung der Tupel in den zwei neuen Relationen
- $R(A_1, A_2, \dots, A_n)$  kann in  $S(B_1, B_2, \dots, B_m)$  und  $T(C_1, C_2, \dots, C_k)$  dekomponiert werden, falls
  - $\{A_1, A_2, \dots, A_n\} = \{B_1, B_2, \dots, B_m\} \cup \{C_1, C_2, \dots, C_k\}$
  - Tupel in S sind die Projektion aller Tupel in R auf  $\{B_1, B_2, \dots, B_m\}$ 
    - ◇ Insbesondere: Duplikate werden entfernt
    - ◇ Dadurch: Verminderung der Redundanz
  - Tupel in T sind Projektionen auf  $\{C_1, C_2, \dots, C_k\}$ .

Anmerkung: Schemata können sich überlappen

# Dekomposition von Relationen

75

## Filme

Titel	Jahr	Länge	Typ	StudioName	SchauspName
Total Recall	1990	113	Farbe	Fox	Sharon Stone
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone
Total Recall	1990	113	Farbe	Fox	Arnold
Dead Man	1995	121	s/w	Paramount	Johnny Depp

- Vorschlag zur Dekomposition

- Filme1(Titel, Jahr, Länge, Typ, StudioName)
- Filme2(Titel, Jahr, SchauspName)

## Filme1

Titel	Jahr	Länge	Typ	StudioName
Total Recall	1990	113	Farbe	Fox
Basic Instinct	1992	127	Farbe	Disney
Dead Man	1995	121	s/w	Paramount

## Filme2

Titel	Jahr	SchauspName
Total Recall	1990	Sharon Stone
Basic Instinct	1992	Sharon Stone
Total Recall	1990	Arnold
Dead Man	1995	Johnny Depp

- Anomalien sind beseitigt:

- Redundanz ist verschwunden
- Update Anomalie
- Insert Anomalie
- Delete Anomalie

Redundanz?

# Boyce-Codd-Normalform (BCNF)

76

BCNF ist eine Bedingung zur Eliminierung der Anomalien

- Eine Relation  $R$  ist in BCNF genau dann wenn:
- Für jede nicht-triviale FD  $A_1A_2\dots A_n \rightarrow B$  für  $R$  ist  $\{A_1, A_2, \dots, A_n\}$  ein Superschlüssel für  $R$ .
- Reminder:
  - Nicht-trivial: Wenigstens ein Attribut rechts kommt links nicht vor. D.h. hier:  $B \notin \{A_1, A_2, \dots, A_n\}$
  - Superschlüssel: Schlüssel oder Supermenge eines Schlüssels
- Anders: Die linke Seite jeder gültigen, nicht-trivialen FD muss ein Superschlüssel sein.
- Nochmal anders: Die linke Seite jeder gültigen, nicht-trivialen FD muss einen Schlüssel enthalten.
- Was darf also nicht gelten?
  - Motivation: FDs zu Schlüsselabhängigkeiten machen.

# Boyce-Codd-Normalform (BCNF)

77

## Allgemeinere (und praktischere) Formulierung

- Eine Relation  $R$  ist in BCNF genau dann wenn:
- Für jede nicht-triviale FD  $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$  für  $R$  ist  $\{A_1, A_2, \dots, A_n\}$  ein Superschlüssel für  $R$ .
- Äquivalent?
  - $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$  ist Abkürzung für  $m$  FDs.
  - Wegen nicht-Trivialität gibt es mindestens ein  $B_i \notin \{A_1, A_2, \dots, A_n\}$
  - $A_1A_2\dots A_n \rightarrow B_i$  ist dann die Verletzung der BCNF
- Motivation: Dekomposition mit allen Attributen, die auf der rechten Seite sind.

# BCNF – Beispiel

78

Titel	Jahr	Länge	Typ	StudioName	SchauspName
Total Recall	1990	113	Farbe	Fox	Sharon Stone
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone
Total Recall	1990	113	Farbe	Fox	Arnold
Dead Man	1995	121	s/w	Paramount	Johnny Depp

- Tabelle ist nicht in BCNF
- Prüfung
  - Einziger Schlüssel?
    - ◇ {Titel, Jahr, SchauspName}
  - Superschlüssel enthalten also mindestens diese drei Attribute.
  - Eine FD: Titel, Jahr → Länge, Typ, StudioName
    - ◇ Titel, Jahr → Länge
    - ◇ Titel, Jahr → Typ
    - ◇ Titel, Jahr → StudioName
  - ⇒ nicht BCNF


# BCNF – Beispiel

## Filme

79

Titel	Jahr	Länge	Typ	StudioName	SchauspName
Total Recall	1990	113	Farbe	Fox	Sharon Stone
Basic Instinct	1992	127	Farbe	Disney	Sharon Stone
Total Recall	1990	113	Farbe	Fox	Arnold
Dead Man	1995	121	s/w	Paramount	Johnny Depp

### Filme1



Titel	Jahr	Länge	Typ	StudioName
Total Recall	1990	113	Farbe	Fox
Basic Instinct	1992	127	Farbe	Disney
Dead Man	1995	121	s/w	Paramount

### Filme2



Titel	Jahr	SchauspName
Total Recall	1990	Sharon Stone
Basic Instinct	1992	Sharon Stone
Total Recall	1990	Arnold
Dead Man	1995	Johnny Depp



■ Filme1 ist in BCNF

- Titel, Jahr → Länge, Typ, StudioName
- {Titel, Jahr} ist einziger Schlüssel
- Jede (nicht-triviale) FD hat mindestens Titel und Jahr auf der linken Seite.

# BCNF – zwei Attribute

80

Jede Relation mit nur zwei Attributen ist in BCNF.

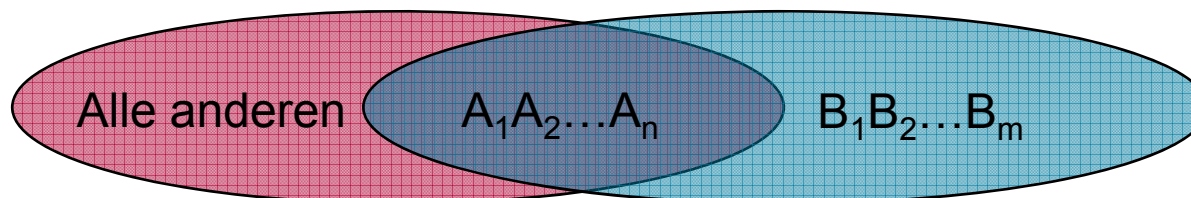
- Analyse aller FDs mit nur einem Attribut rechts.
- Fallunterscheidung
  - Keine FDs  $\Rightarrow$  BCNF, da nur FDs dies verletzen können.
  - $A \rightarrow B$ , aber nicht  $B \rightarrow A$ 
    - ◇ A ist einziger Schlüssel
    - ◇ Jede (nicht-triviale) FD hat A auf der linken Seite
  - $B \rightarrow A$ , aber nicht  $A \rightarrow B$ 
    - ◇ Analog
  - $A \rightarrow B$  und  $B \rightarrow A$ 
    - ◇ A und B sind jeweils Schlüssel 
    - ◇ Jede FD hat einen der beiden Schlüssel auf der linken Seite. 



# Dekomposition zu BCNF

81

- Ziel: Wiederholte Dekomposition von Relationen
  - Zur Erreichung der BCNF
  - Unter Garantie der Wiederherstellbarkeit der ursprünglichen Relation
- Dekomposition in lauter 2er Relationen garantiert nicht Wiederherstellbarkeit
- FDs helfen
- Grundalgorithmus
  - Suche BCNF-verletzende FDs ( $A_1A_2\dots A_n \rightarrow B_1B_2\dots B_m$ ).
  - Füge auf der rechten Seite so viele Attribute wie möglich hinzu.
  - Erzeuge zwei neue Relationen:



# Dekomposition zu BCNF – Beispiel 1

82

- Filme(Titel, Jahr, Länge, Typ, StudioName, SchauspName)
- Titel, Jahr  $\rightarrow$  Länge, Typ, Studioname verletzt BCNF
- Neue Relationen
  - Filme1(Titel, Jahr, Länge, Typ, StudioName)
  - Filme2(Titel, Jahr, SchauspName)
  - Beide sind in BCNF.

# Dekomposition zu BCNF – Beispiel 2

83

Titel	Jahr	Länge	Typ	StudioName	StudioAdresse
Total Recall	1990	113	Farbe	Fox	Hollywood
Basic Instinct	1992	127	Farbe	Disney	Buena Vista
Total Recall	1990	113	Farbe	Fox	Hollywood
Dead Man	1995	121	s/w	Paramount	Buena Vista

- Titel, Jahr → Länge, Typ, StudioName
- StudioName → StudioAdresse
- Transitivität: Titel, Jahr → StudioAdresse
- ⇒ {Titel, Jahr} ist Schlüssel
- StudioName → StudioAdresse verletzt also BCNF
- Zwei neue Relationen
  - Filme1(Titel, Jahr, Länge Typ, StudioName)
  - Filme2(StudioName, StudioAdresse)

# Dekomposition zu BCNF – Beispiel 2

84

Titel	Jahr	Länge	Typ	StudioName	StudioAdresse
Total Recall	1990	113	Farbe	Fox	Hollywood
Basic Instinct	1992	127	Farbe	Disney	Buena Vista
Total Recall	1990	113	Farbe	Fox	Hollywood
Dead Man	1995	121	s/w	Paramount	Buena Vista



Titel	Jahr	Länge	Typ	StudioName
Total Recall	1990	113	Farbe	Fox
Basic Instinct	1992	127	Farbe	Disney
Total Recall	1990	113	Farbe	Fox
Dead Man	1995	121	s/w	Paramount

StudioName	StudioAdresse
Fox	Hollywood
Disney	Buena Vista
Paramount	Buena Vista

- Titel, Jahr → Länge, Typ, StudioName
- {Titel, Jahr} ist Schlüssel

- StudioName → StudioAdresse
- {StudioName} ist Schlüssel

# Vergleich der beiden Beispiele

85

- Beispiel 1
  - BCNF-Verletzung durch Darstellung einer n:m Beziehung in einer einzigen Relation (Filme, Schauspieler).
  - Zusammen mit anderen Informationen über Filme
- Beispiel 2
  - BCNF-Verletzung durch Darstellung einer n:1 Beziehung in einer einzigen Relation (Filme, Studios)
  - Zusammen mit anderen Informationen über Studios
  - Abhängigkeit ist nur transitiv
    - ◇ Titel, Jahr → Länge, Typ, StudioName
    - ◇ StudioName → StudioAdresse

## Dekomposition zu BCNF – Beispiel 3

86

- Filme(Titel, Jahr, StudioName, Präsident, PräsAdresse)
  - Titel, Jahr → StudioName
  - StudioName → Präsident
  - Präsident → PräsAdresse
  - $\Rightarrow$  {Titel, Jahr} ist Schlüssel
- Erste Dekomposition anhand von StudioName → Präsident
  - Hinzufügen von möglichst vielen Attributen auf der rechten Seite: StudioName → Präsident, PräsAdresse
  - Filme1(Titel, Jahr, StudioName)
  - Filme2(StudioName, Präsident, PräsAdresse)
    - ◇ Hier gilt weiter Präsident → PräsAdresse
    - ◇ BCNF Verletzung
- Zweite Dekomposition
  - Filme2 wird zu Filme2(StudioName, Präsident)
  - Filme3(Präsident, PräsAdresse)
- Verfahren terminiert, da jede neue Relation kleiner wird und 2er-Relationen garantiert in BCNF sind.

# Wiederherstellung von Informationen

87

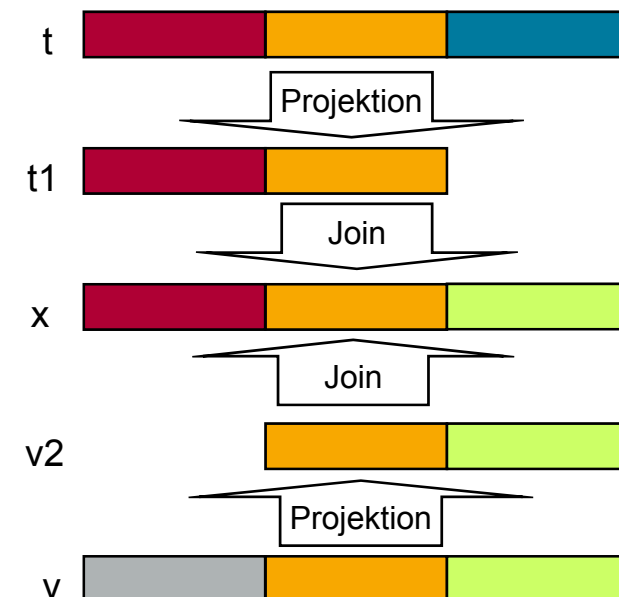
- Angenommen:  $R(A,B,C)$  mit  $B \rightarrow C$  als BCNF-Verletzung
  - Z.B. weil  $A \rightarrow B$  gilt, und somit  $\{A\}$  Schlüssel ist
  - Oder z.B. weil  $B \rightarrow C$  die einzige FD ist, und somit  $\{A,B\}$  Schlüssel ist.
- Dekomposition:  $R_1(A,B)$  und  $R_2(B,C)$
- Sei  $t = (a,b,c)$  ein Tupel in  $R$ 
  - Wird zu  $t_1(a,b)$  in  $R_1$  und  $t_2(b,c)$  in  $R_2$
- Wiederherstellung durch „Join“ (Verbund).
  - Vollständigkeit
    - ◇ Kombination von Tupeln zweier Relationen, die in den Werten für gemeinsame Attribute übereinstimmen.
    - ◇  $t_1(a,b)$  verknüpft mit  $t_2(b,c)$  wird zu  $t(a,b,c)$
  - Korrektheit
    - ◇ Nächste Folie

# Wiederherstellung von Informationen

88

## Korrektheit

- Seien  $t(a,b,c)$  und  $v(d,b,e)$  zwei Tupel in  $R$
- Dekomposition mit Projektion
- In  $R_1$ :  $t_1(a,b)$  und  $v_1(d,b)$
- In  $R_2$ :  $t_2(b,c)$  und  $v_2(b,e)$
- Join ergibt Tupel
  - $t(a,b,c)$
  - $x(a,b,e)$
  - $v(d,b,e)$
- Ist  $x(a,b,e)$  ein Fehler?
- Nein, denn  $B \rightarrow C$ 
  - D.h.  $c = e$
- Dies gilt auch allgemeiner für Attributmengen





# Wiederherstellung von Informationen

89

## Dekomposition ohne FD

- Angenommen  $R(A,B,C)$  ohne  $B \rightarrow C$
- Projektionen auf  $R_1(A,B)$  und  $R_2(B,C)$

A	B
1	2
4	2

B	C
2	3
2	5

A	B	C
1	2	3
4	2	5

- Wiederherstellung durch Join über B

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

Falsch!

# Weitere Normalformen

90

- 1. Normalform (1NF)
  - Nur atomare Werte
- 2. Normalform (2NF)
  - 1NF und keine Abhängigkeiten von einem Teil eines Schlüssels
- 3. Normalform (3NF)
  - 2NF und zusätzlich keine transitiven Abhängigkeiten
- Boyce-Codd Normalform (BCNF)
  - 3NF und keine transitiven Abhängigkeiten auch innerhalb des Schlüssels
- 4. Normalform (4NF)
  - Mehrwertige Abhängigkeiten

# 1. Normalform

91

1NF: Nur atomare Werte

- Relation nicht in 1NF:
  
- Umgewandelte Relation in 1NF:
  
- Andere Umwandlungsmöglichkeit
  - R(Vater, Mutter, Kind1, Kind2)

<i>Vater</i>	<i>Mutter</i>	<i>Kinder</i>
Johann	Martha	{Else, Lucie}
Johann	Maria	{Theo, Josef}
Heinz	Martha	{Cleo}

<i>Vater</i>	<i>Mutter</i>	<i>Kind</i>
Johann	Martha	Else
Johann	Martha	Lucie
Johann	Maria	Theo
Johann	Maria	Josef
Heinz	Martha	Cleo

Beispiel nach Alfons Kemper (TU München)

## 2. Normalform

92

1NF und keine Abhängigkeiten von Nicht-Schlüssel-Attributen von einem Teil eines Schlüssels

- Matr → Name
  - Aber Matr ist nicht vollständiger Schlüssel
  
- Abhilfe: Dekomposition
  - R1(MatrnNr, VorlNr)
  - R2(MatrnNr, Name, Semester)

<u>MatrnNr</u>	<u>VorlNr</u>	Name	Semester
26120	5001	Fichte	10
27550	5001	Schopenhauer	6
27550	4052	Schopenhauer	6
28106	5041	Carnap	3
28106	5052	Carnap	3
28106	5216	Carnap	3
28106	5259	Carnap	3
...	...	...	...

## 3. Normalform

93

### Kinoaufführungen

- R(Titel, Kino, Stadt)
- FDs
  - Kino → Stadt (ein Kino steht in nur einer Stadt)
  - Titel, Stadt → Kino (Annahme: Ein Film wird nicht zweifach in der gleichen Stadt aufgeführt)
- Schlüssel
  - Einzelne Attribute sind nicht Schlüssel
  - {Titel, Stadt} ist Schlüssel, da sie funktional alle anderen Attribute bestimmt.
  - {Kino, Titel} ist auch Schlüssel, da Kino → Stadt augmentiert werden kann zu Kino, Titel → Stadt
- BCNF-Verletzung:
  - Kino → Stadt (da Kino nicht Superschlüssel ist)

# 3. Normalform

94

## Dekomposition

- Verletzende FD: Kino → Stadt
- Dekomposition
  - R1(Kino, Stadt)
  - R2(Kino, Titel)
- Problem:
  - Titel, Stadt → Kino kann nicht mehr sichergestellt werden.

□ Gültige Beispielrelationen:

Kino	Stadt	Kino	Titel
Ufa	Berlin	Ufa	Star Wars
Babylon	Berlin	Babylon	Star Wars

□ Aber Wiederherstellung:

Kino	Stadt	Titel
Ufa	Berlin	Star Wars
Babylon	Berlin	Star Wars

### 3. Normalform

95

Lösung des Problems durch Relaxierung der BCNF

- Eine Relation  $R$  ist in 3. Normalform genau dann wenn:
- Für jede nicht-triviale FD  $A_1A_2...A_n \rightarrow B$  für  $R$  ist
  - entweder  $\{A_1, A_2, \dots, A_n\}$  ein Superschlüssel für  $R$ ,
  - oder  $B$  ist Teil eines Schlüssels für  $R$ .
- Kurz: Für jede FD ist entweder die linke Seite ein Superschlüssel oder die rechte Seite Teil eines Schlüssels.
- Am Beispiel
  - $R(\text{Titel}, \text{Kino}, \text{Stadt})$  mit FDs
    - ◇  $\text{Kino} \rightarrow \text{Stadt}$
    - ◇  $\text{Titel}, \text{Stadt} \rightarrow \text{Kino}$

Verletzt nicht 3.  
Normalform, da Stadt Teil  
eines Schlüssels ist.

## 3NF vs. BCNF

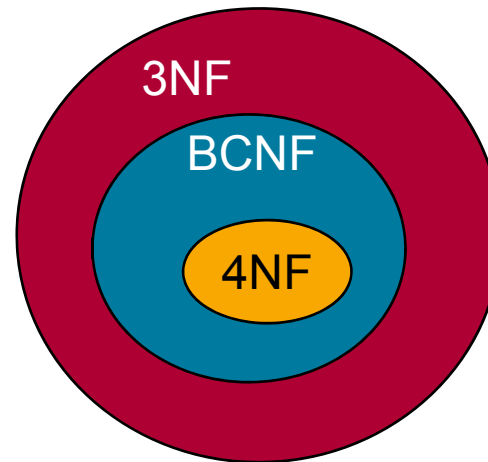
96

- Wichtige Eigenschaften der Dekomposition
  1. Wiederherstellbarkeit
    - ◇ Projektion der ursprünglichen Relation auf die neuen Relationen und dann Rekonstruktion der ursprünglichen Relation (mittels Join).
  2. Bewahrung der FDs
    - ◇ Prüfbarkeit aller FDs in den neuen Relationen
- BCNF garantiert 1.
- 3NF garantiert 1. und 2.



# Zusammenfassung – Normalformen

97



Eigenschaft	3NF	BCNF	4NF
Eliminiert Redundanzen aus FDs	Die meisten	Ja	Ja
Eliminiert Redundanzen aus MVDs	Nein	Nein	Ja
Erhält FDs	Ja	Vielleicht	Vielleicht
Erhält MVDs	Vielleicht	Vielleicht	Vielleicht

- Relationales Modell
- Schemata
- Von Entitytypen zu Relationen
- Von Relationshiptypen zu Relationen
- Von IST-Hierarchien zu Relationen
- Funktionale Abhängigkeiten
- Schlüssel einer Relation
- Regeln für FDs
- Dekomposition
- BCNF
- 3NF
- MVDs und 4NF