



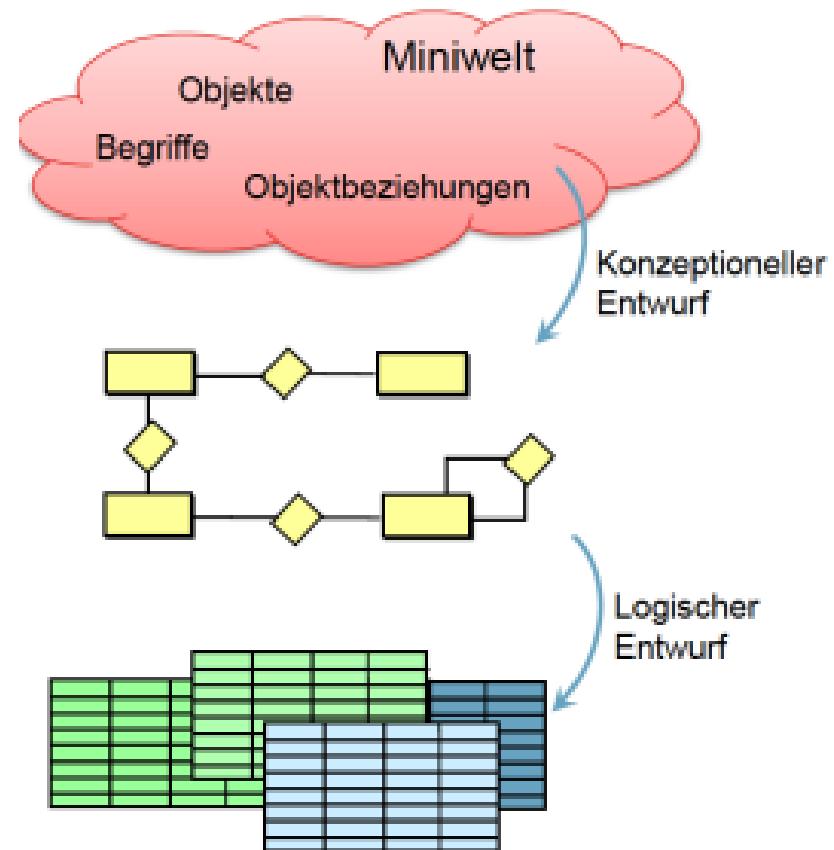
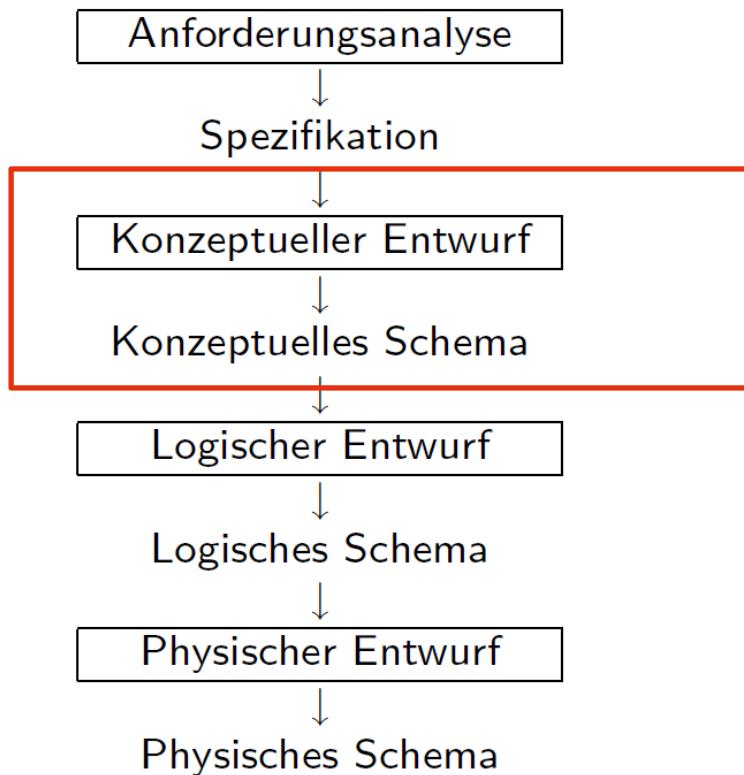
Datenbanksysteme

Debriefing/Wiederholung: Group Puzzle

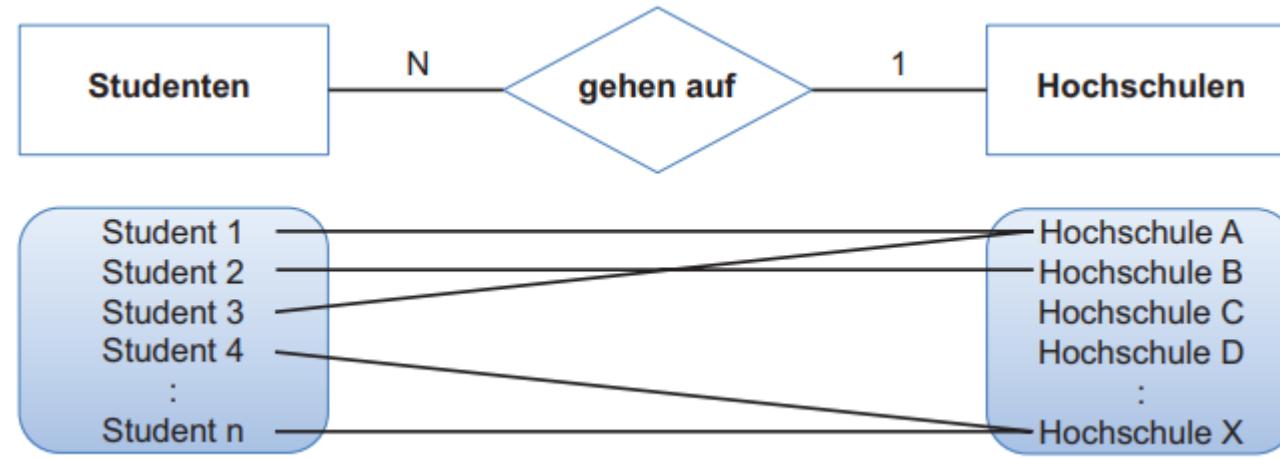
Prof. Dr. Patrick Cato

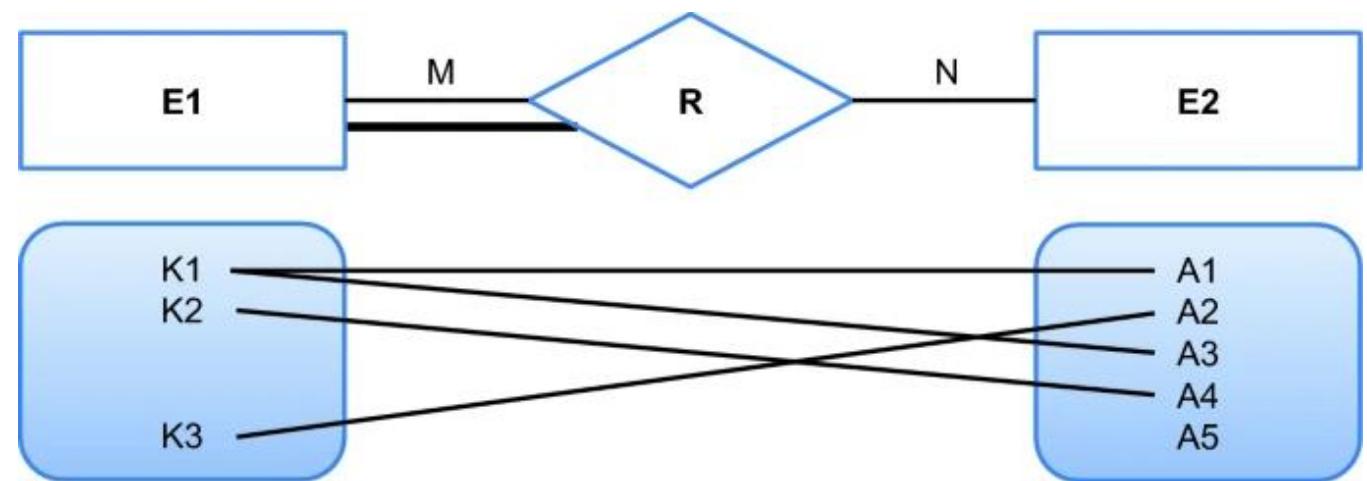
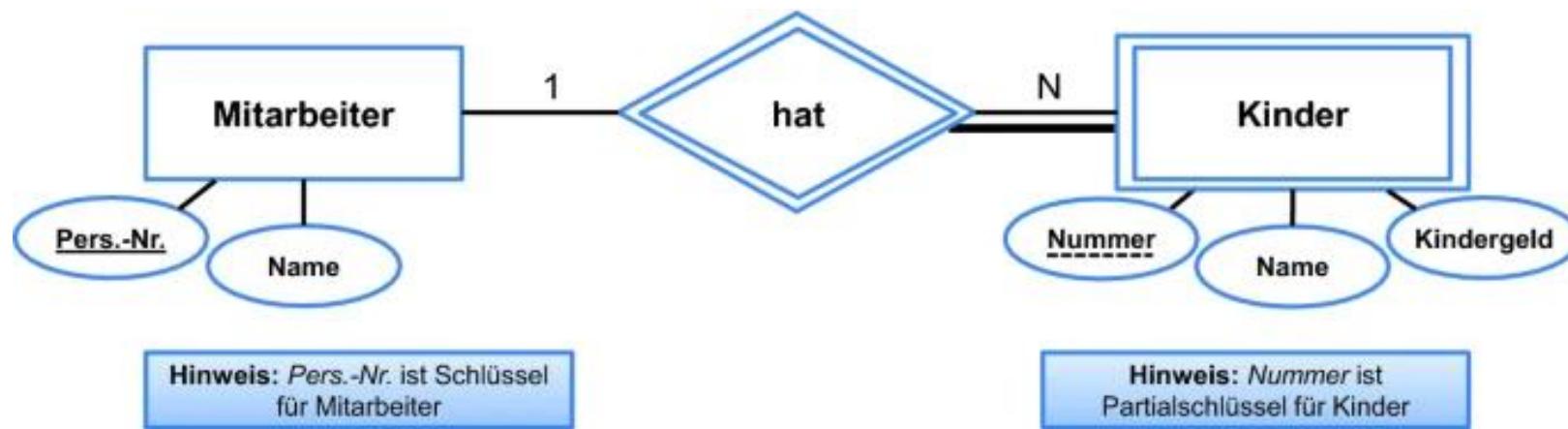
Technische Hochschule Ingolstadt 

Rückblick / Wiederholung

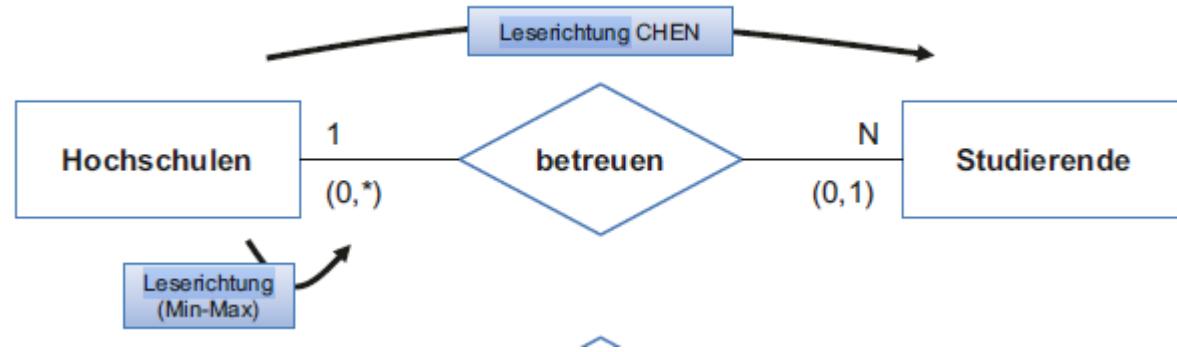


Debriefing: Entitätstyp nicht gleich Entität





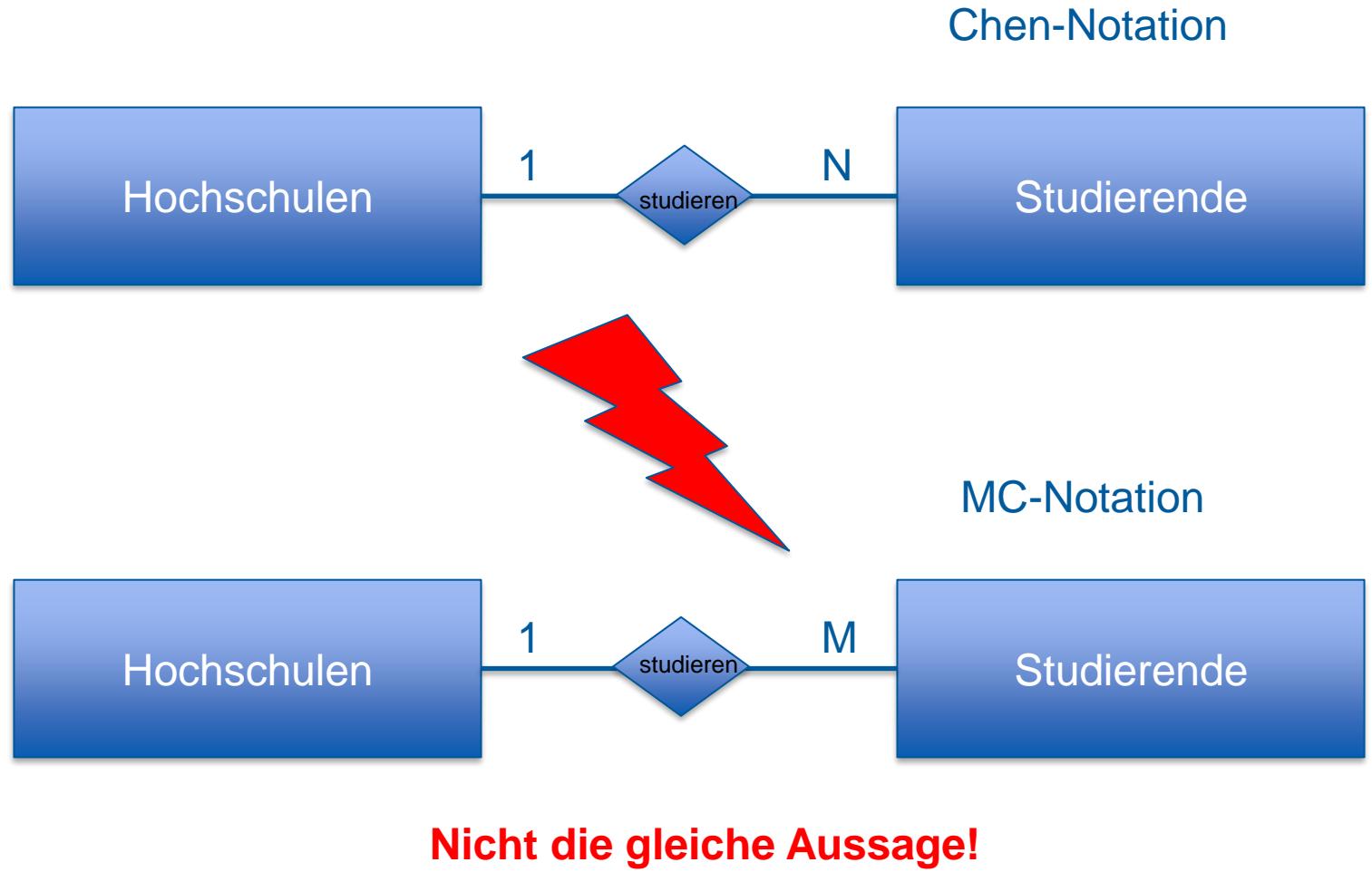
Leserichtung beachten



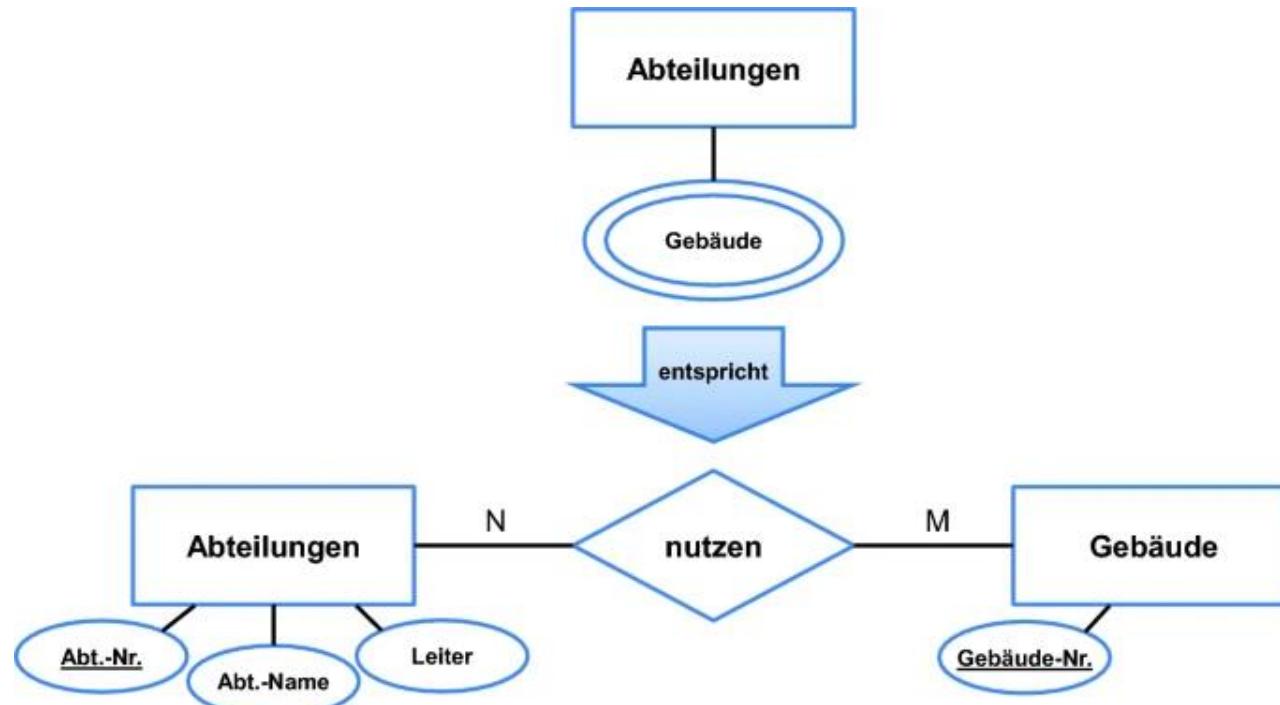
MC-Notation (Modified Chen) nicht das gleiche wie Chen-Notation

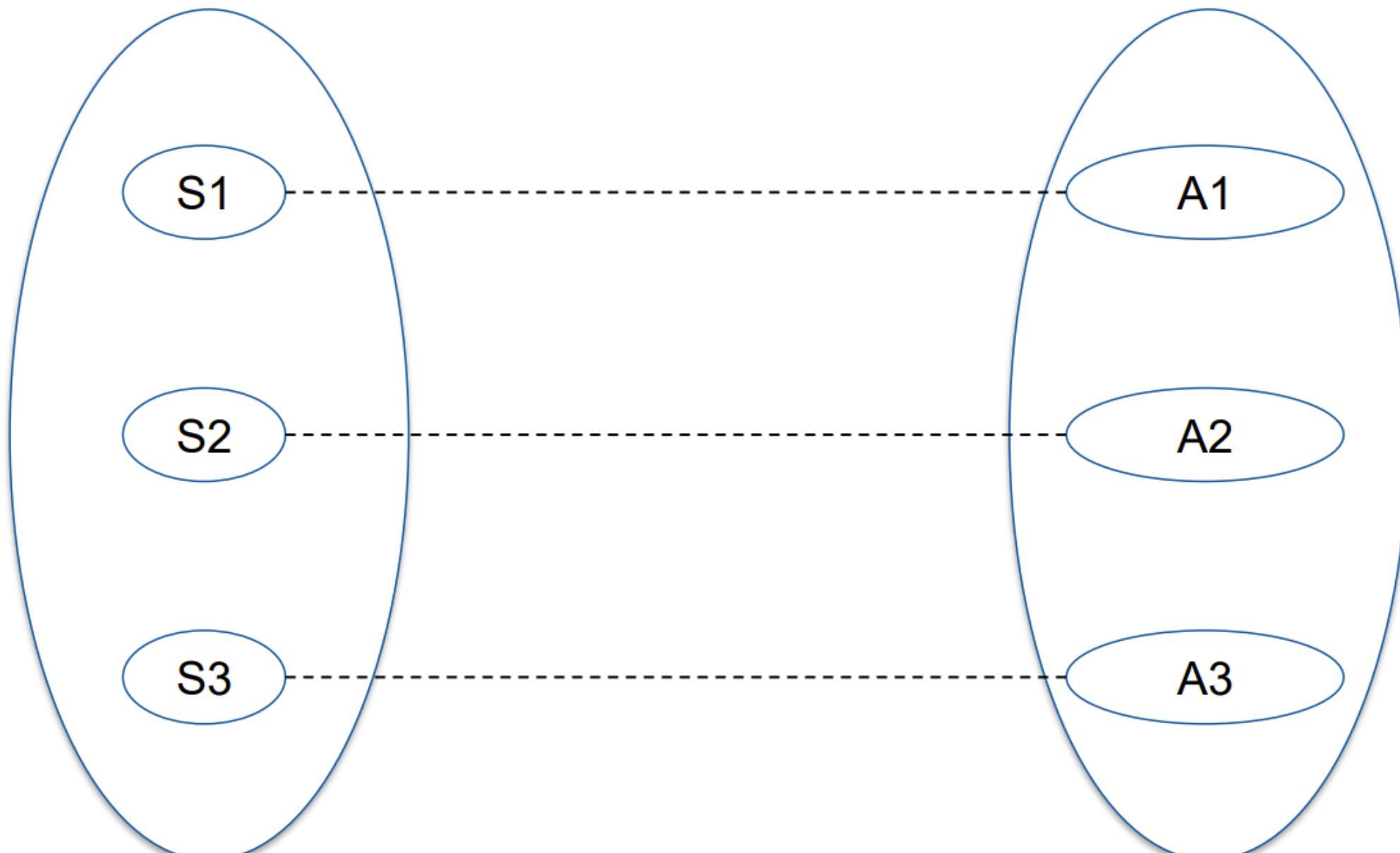


MC-Notation	Min-Max-Notation
C	(0,1)
1	(1,1)
MC	(0,*)
M	(1,*)

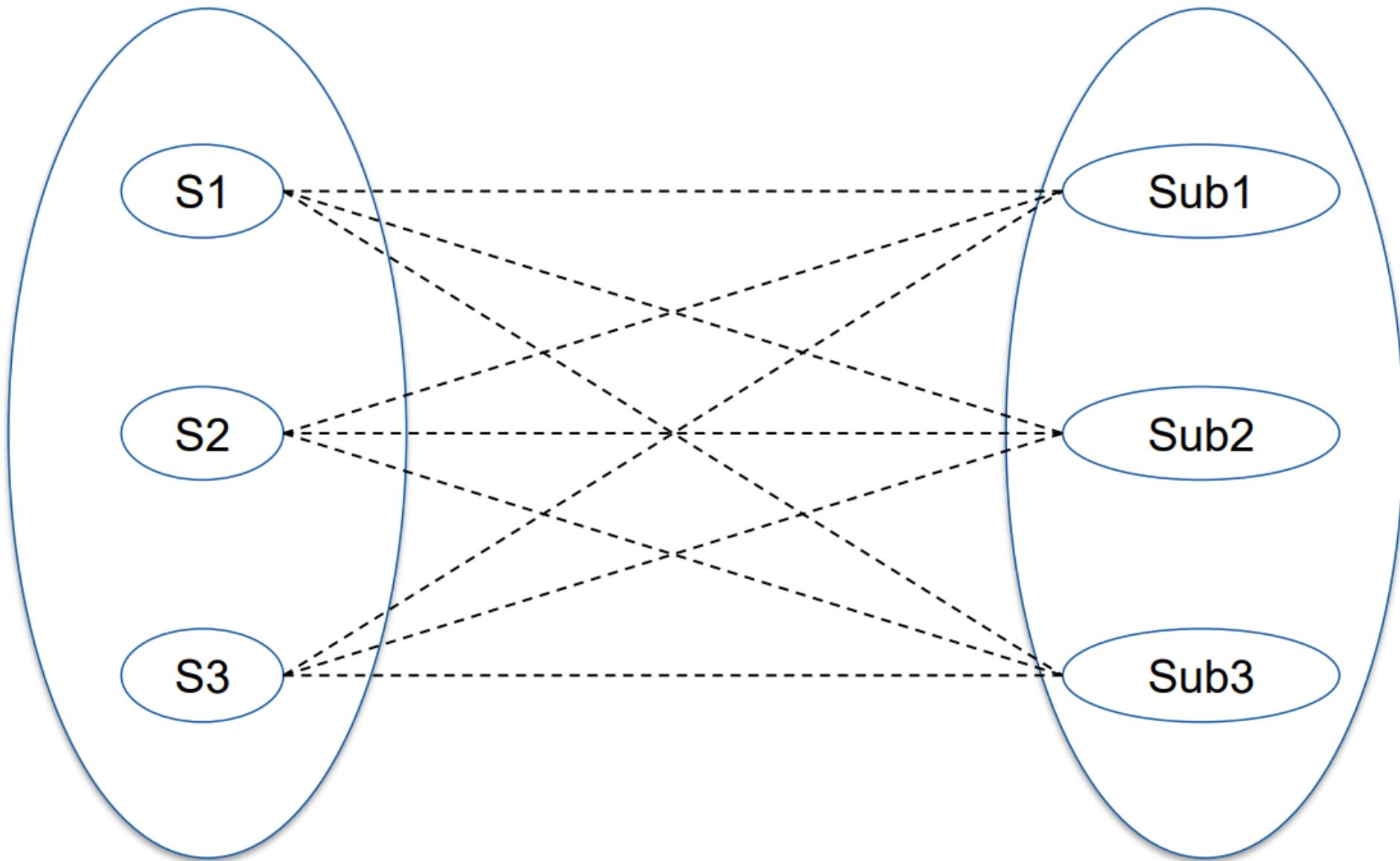


- „Die erste Normalform besagt, dass Werte nur atomar gespeichert werden dürfen. Also modelliere ich doch nicht mit mehrwertigen Attributen, sondern löse das auf“



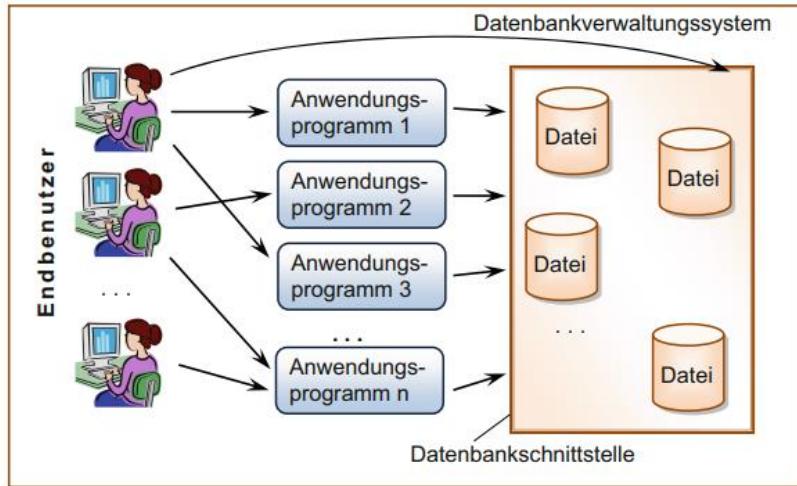


One-to-one relationship



Many-to-many relationship

„Ein Student schreibt eine oder mehrere Prüfungen und erhält je Prüfung eine Note“

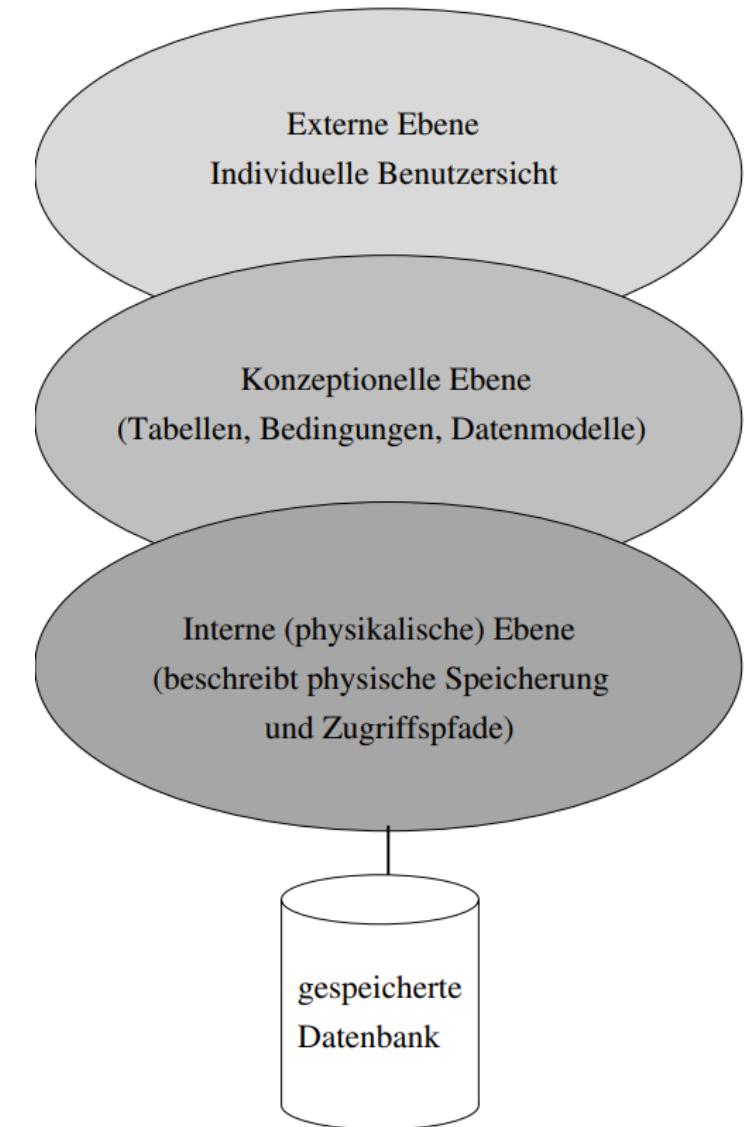
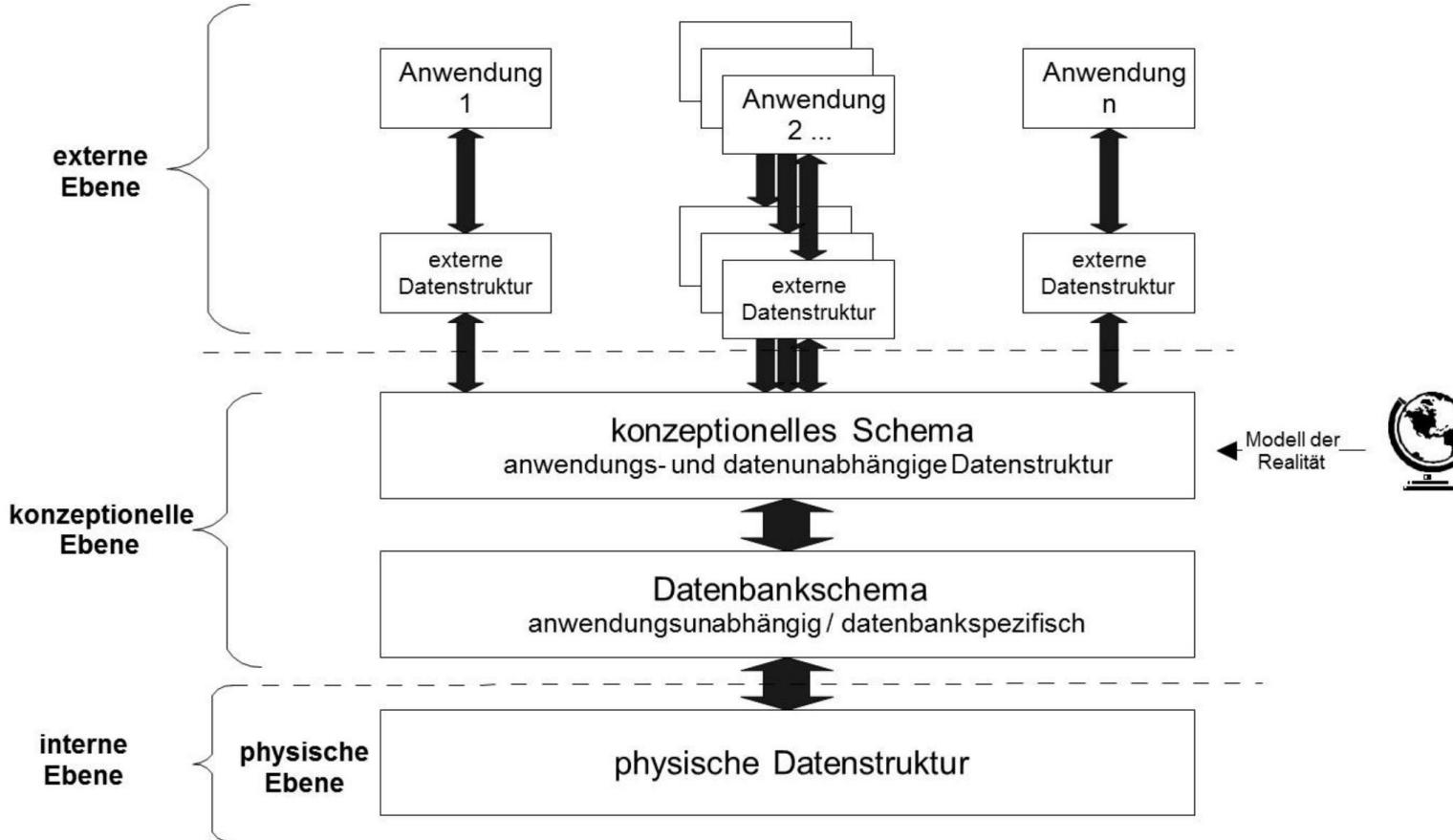


Definition Datenbank

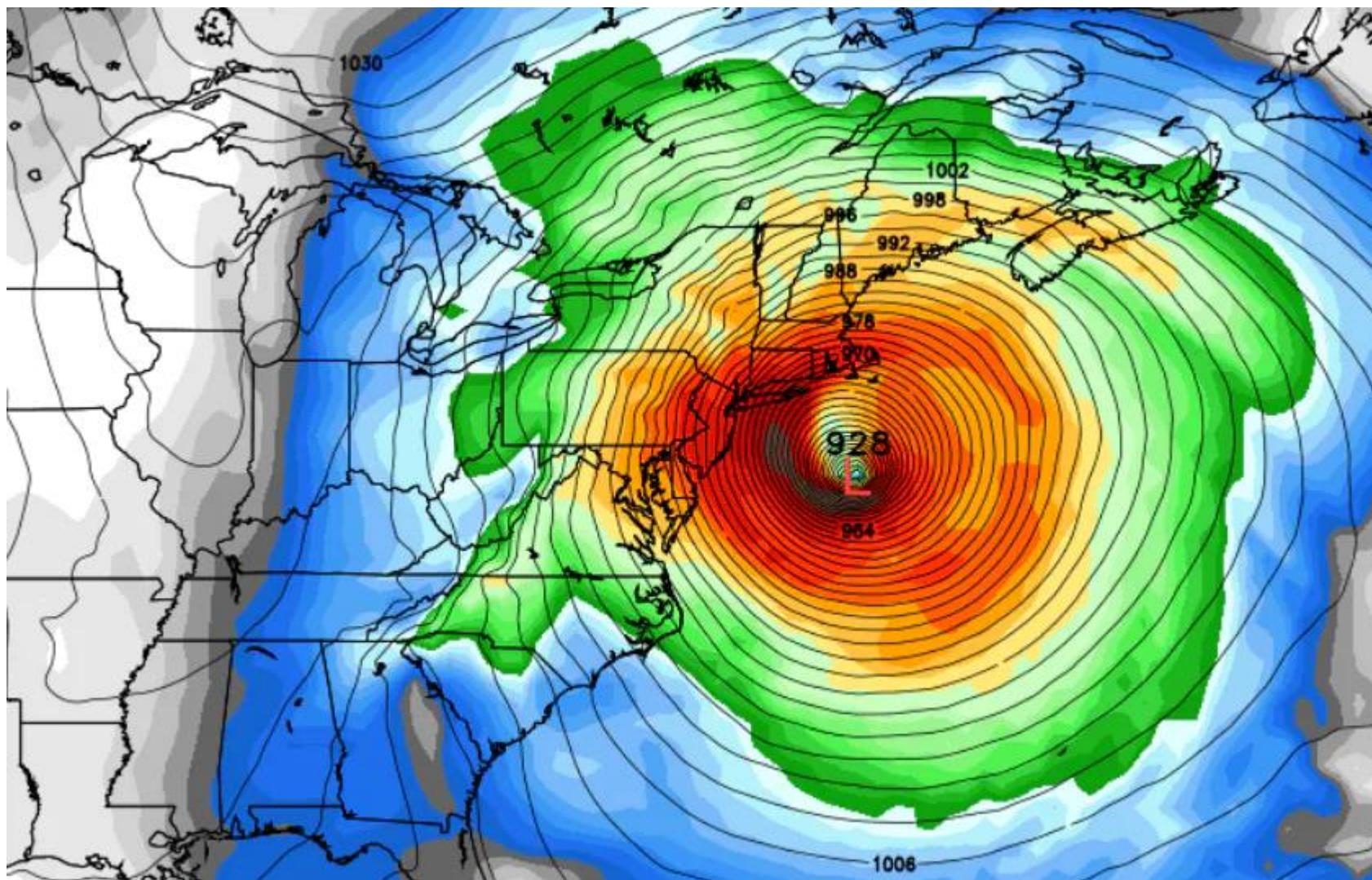
„Eine Datenbank ist eine Sammlung von Daten, [die *untereinander in einer logischen Beziehung stehen*] und von **einem eigenen Datenbankverwaltungssystem** (Database Management System, DBMS) verwaltet werden.“

Schicker (2014: 3)

Recap: ANSI-SPARC-Architektur / Drei-Schema-Architektur



Motivation for Models



Herleitung

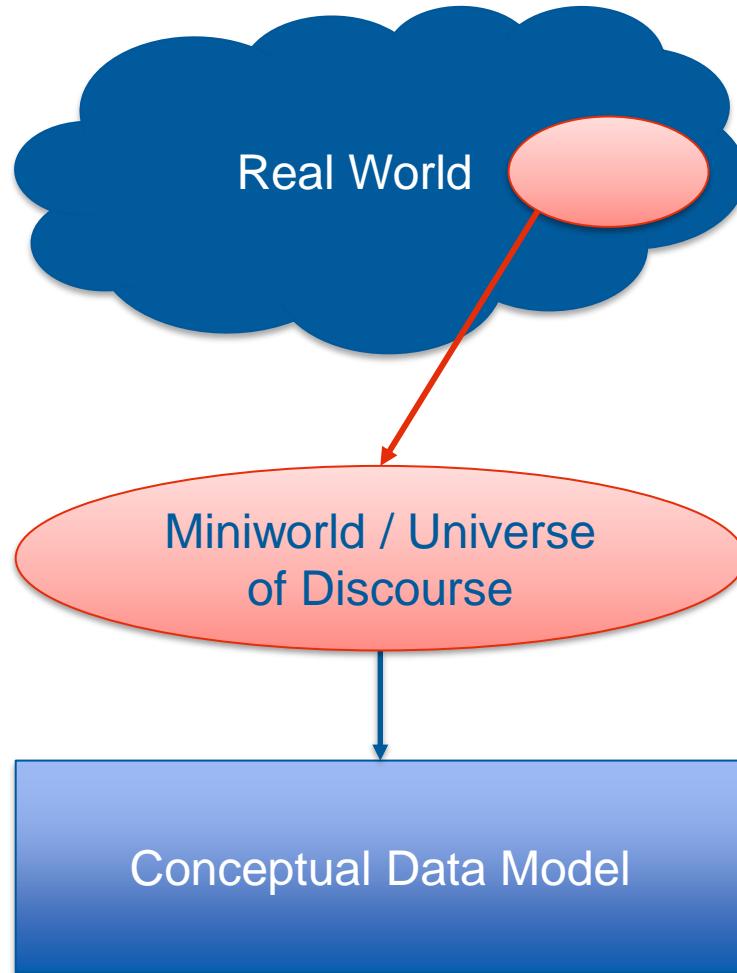


Warum benötige ich ein Modell?

- Modelle vereinfachen den Blick auf die komplexe Realität mit einem konkreten Fokus
- Modell erleichtert die Navigation, in dem man sich auf die wesentlichen Aspekte konzentriert (bspw. Geschwindigkeiten und Tunnelhöhen sind nicht relevant)

Motivation for Models

We do not model the complete world, but only the relevant context!

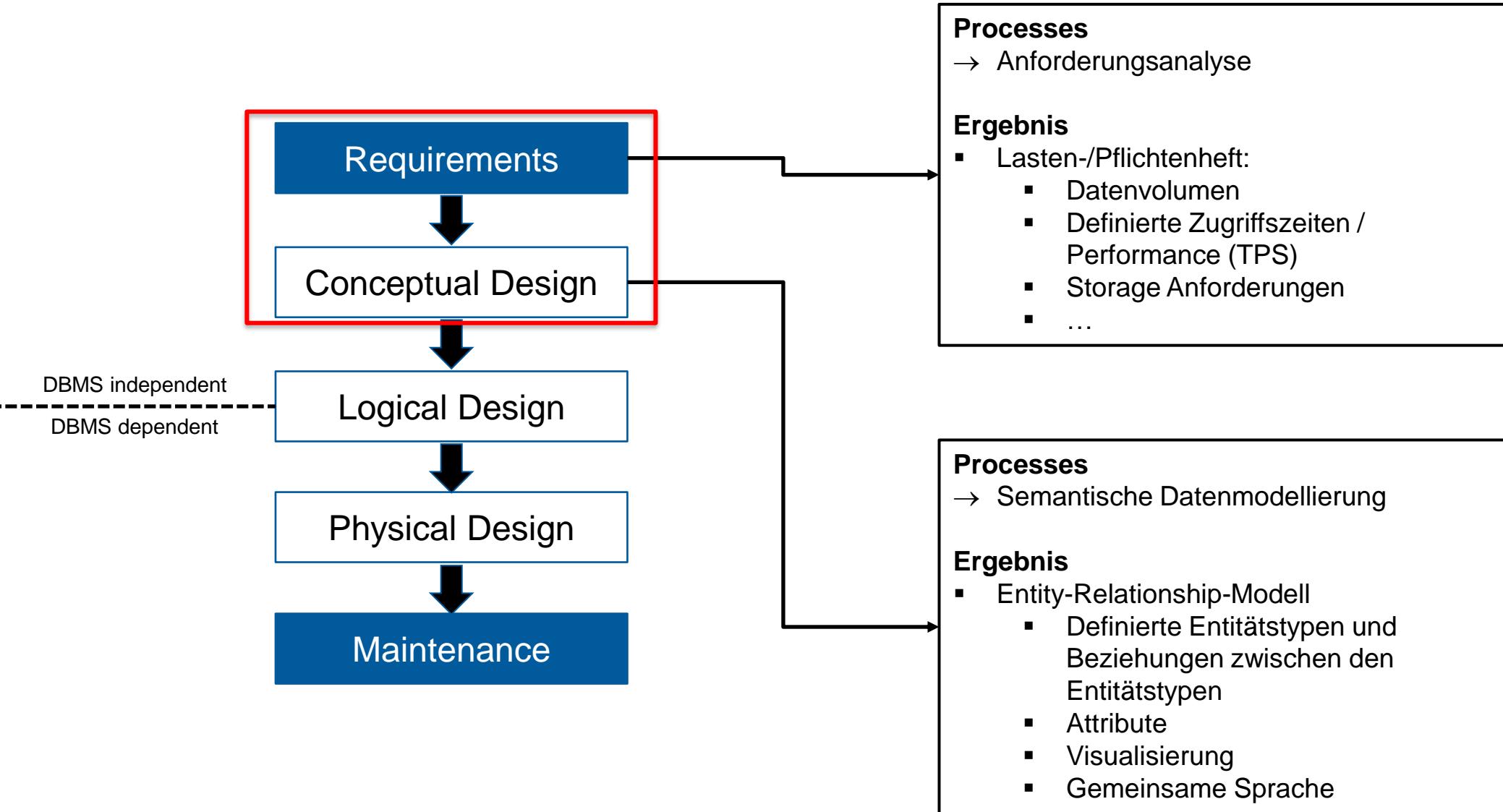


Complex Reality

Represents some aspect of the real world

High-level description of data items, characteristics, and relationships.

Recap: Database Design Phases





Datenbanksysteme

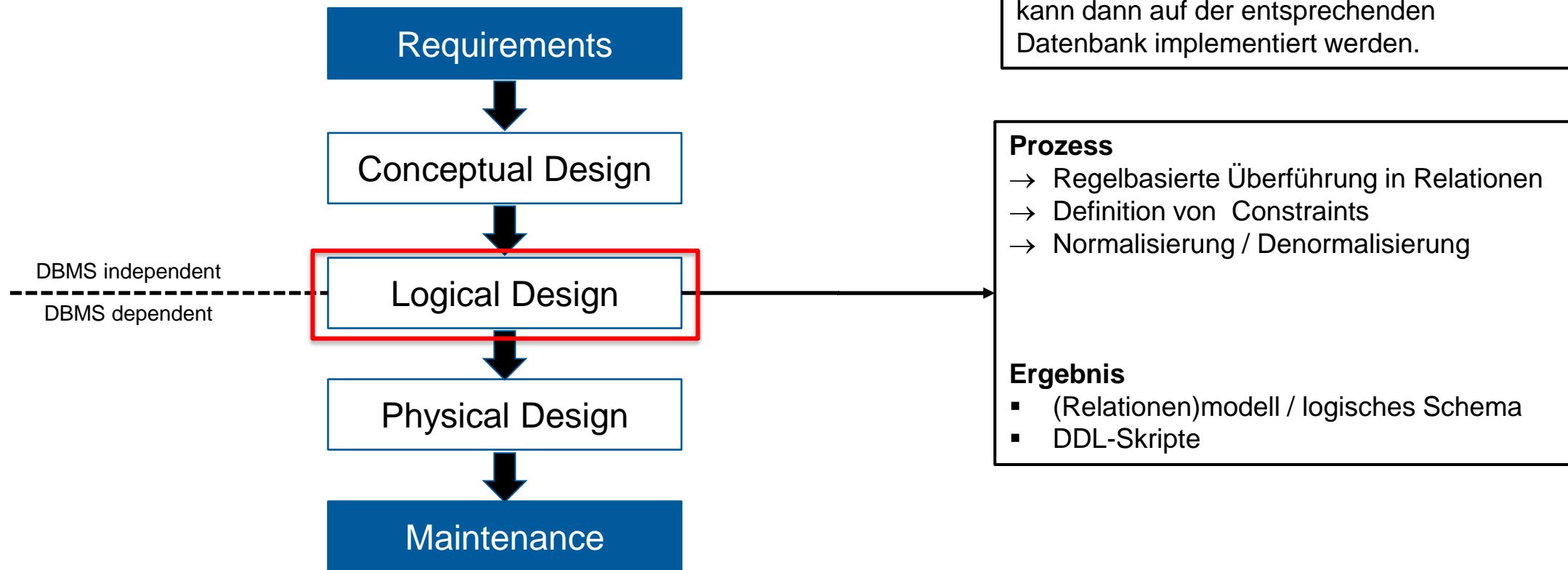
Relationales Datenmodell

Prof. Dr. Patrick Cato



Nach der heutigen Veranstaltung:

- Kann ich den Begriff Relationales Datenmodell erklären
- Kann ich ein konzeptuelles Modell (Entity-Relationship-Modell) in ein logisches Modell (relationales Modell) überführen
- Kenne ich Best Practices zur Überführung von ERM/ERDs in das Relationenmodell



A Relational Model of Data for Large Shared Data Banks

E. F. CODD

IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity

CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. INTRODUCTION

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. 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 data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

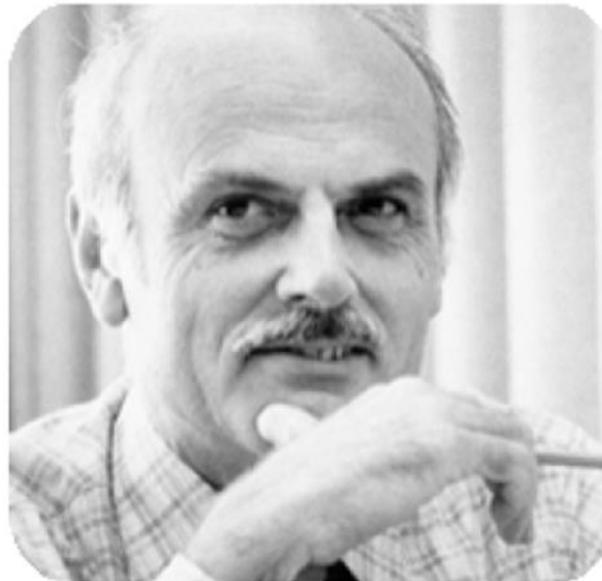
A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the “connection trap”)

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed without logically impairing some application programs is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of record from such a file is identical to (or is a subordering of) the

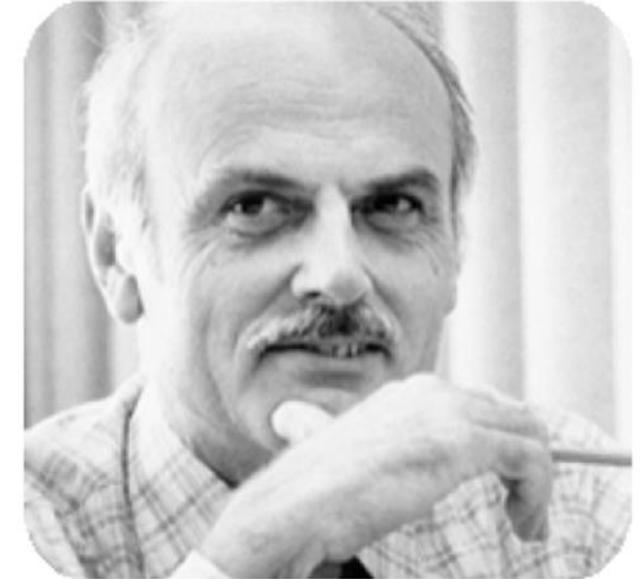


Quelle: <http://amturing.acm.org>

Edgar Codd

Inhaltspunkt

- Datenmodell hinter den heute vorherrschenden relationalen Datenbanksystemen (RDBMS)
- Entwickelt von Edgar F. Codd in den Siebzigern (er bekam den A. M. Turing Award dafür)
- Mengenorientierte Verarbeitung von Daten, die in sogenannten Relationen (Tabellen) gespeichert werden

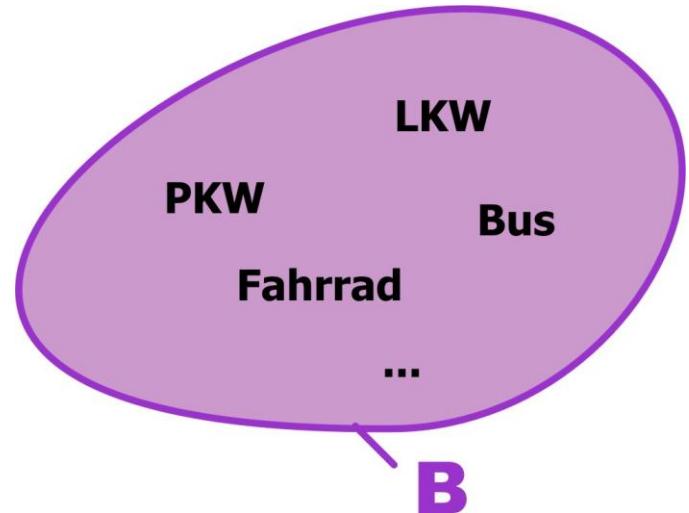
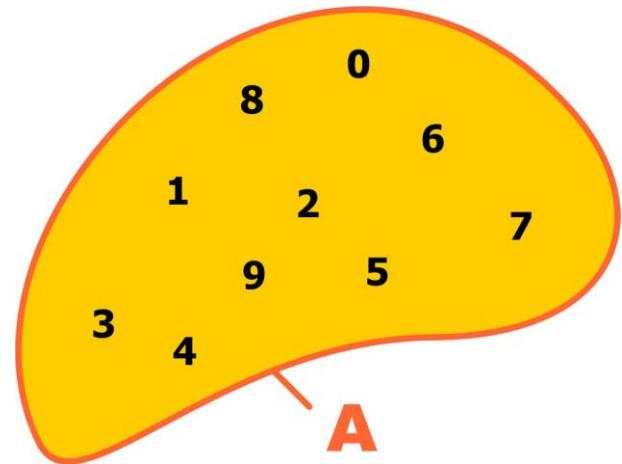


Quelle: <http://amturing.acm.org>

Menge

„Unter einer Menge verstehen wir jede Zusammenfassung M von bestimmten wohlunterschiedenen Objekten (m) unserer Anschauung oder unseres Denkens (welche die Elemente von M genannt werden) zu einem Ganzen.“

(Georg Cantor, 1845-1918)



Teilmenge

Eine Menge A heißt Teilmenge einer Menge B, wenn jedes Element von A auch zur Menge von B gehört:

$$A = \{4,5\}$$

$$B = \{1,2,3,4,5\}$$

A ist in B enthalten

$$A \subseteq B \Leftrightarrow \forall x (x \in A \Rightarrow x \in B)$$

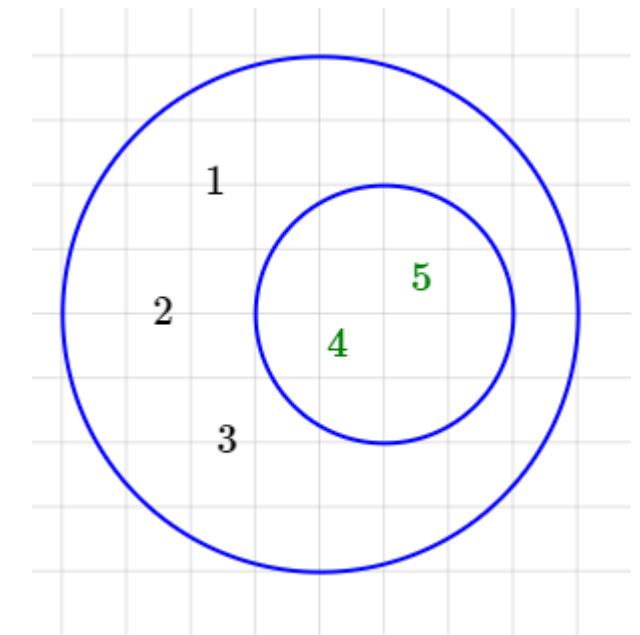
$$\underbrace{A \subseteq B}_{\text{A ist Teilmenge von B}}$$

$$\Leftrightarrow$$

$$\underbrace{\forall x}_{\text{genau dann, wenn f\"ur alle x gilt:}}$$

$$(\underbrace{x \in A \Rightarrow x \in B}_{\text{aus x ist Element von A folgt x ist Element von B}})$$

A ist Teilmenge von B genau dann, wenn f\"ur alle x gilt: aus x ist Element von A folgt x ist Element von B



Kartesisches Produkt

Das kartesische Produkt wird mittels Kreuzprodukt aus beliebigen Mengen gebildet, wobei jedes Objekt der linken Menge mit jedem weiteren Objekt übrigen Mengen kombiniert wird.

Als Ergebnis entsteht ein n-dimensionales Tupel.

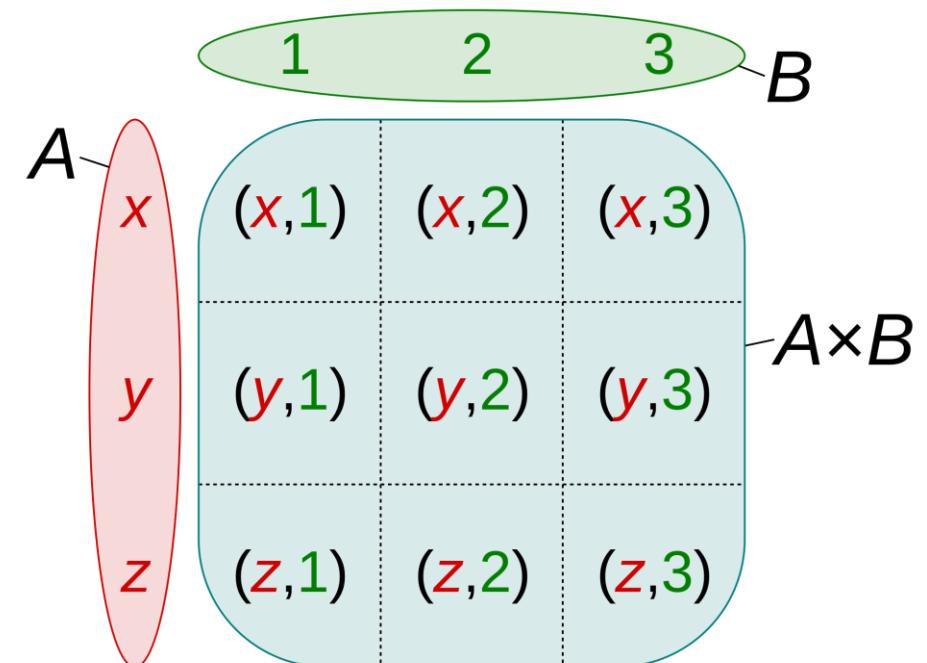
$$A = \{x, y, z\} \quad B = \{1, 2, 3\}$$

$$A \times B = \{ (x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3), (z, 1), (z, 2), (z, 3) \}$$

$$A \times B = \{ (a, b) \mid a \in A \wedge b \in B \}$$

$$\underbrace{A \times B}_{\text{A Kreuz B}} = \underbrace{\{ \quad \underbrace{(a, b)}_{\text{geordnete Paare}} \quad \mid \quad \downarrow \text{f\"ur die gilt:} \}}_{\text{die Menge aller}}$$

$$\underbrace{a \in A}_{\text{a ist Element von A}} \quad \wedge \quad \underbrace{b \in B}_{\text{b ist Element von B}} \quad \}$$

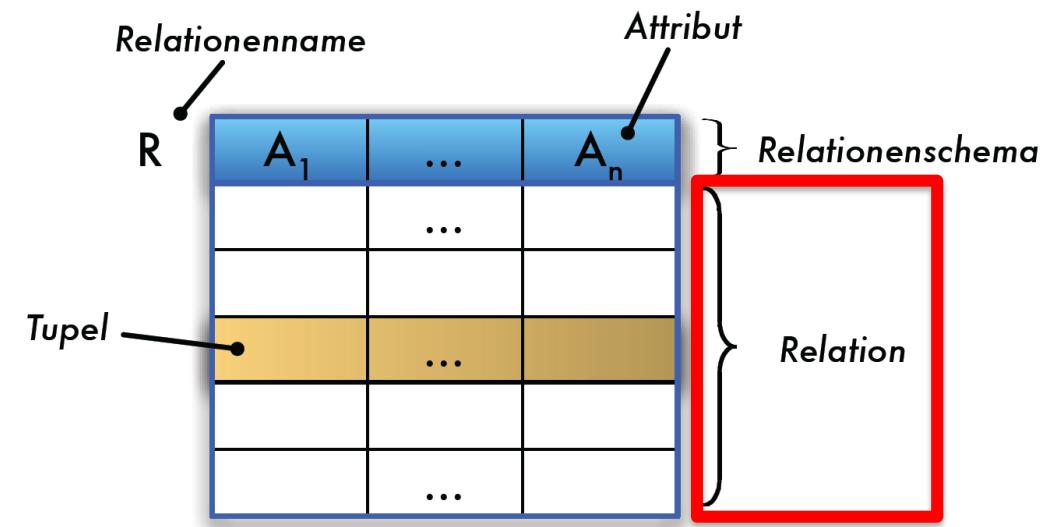


Relation

Eine Relation R ist eine Teilmenge des kartesischen Produkts (Kreuzprodukts) von n Wertebereichen D_1, \dots, D_n

$$R \subseteq D_1 \times D_2 \times \dots \times D_n$$

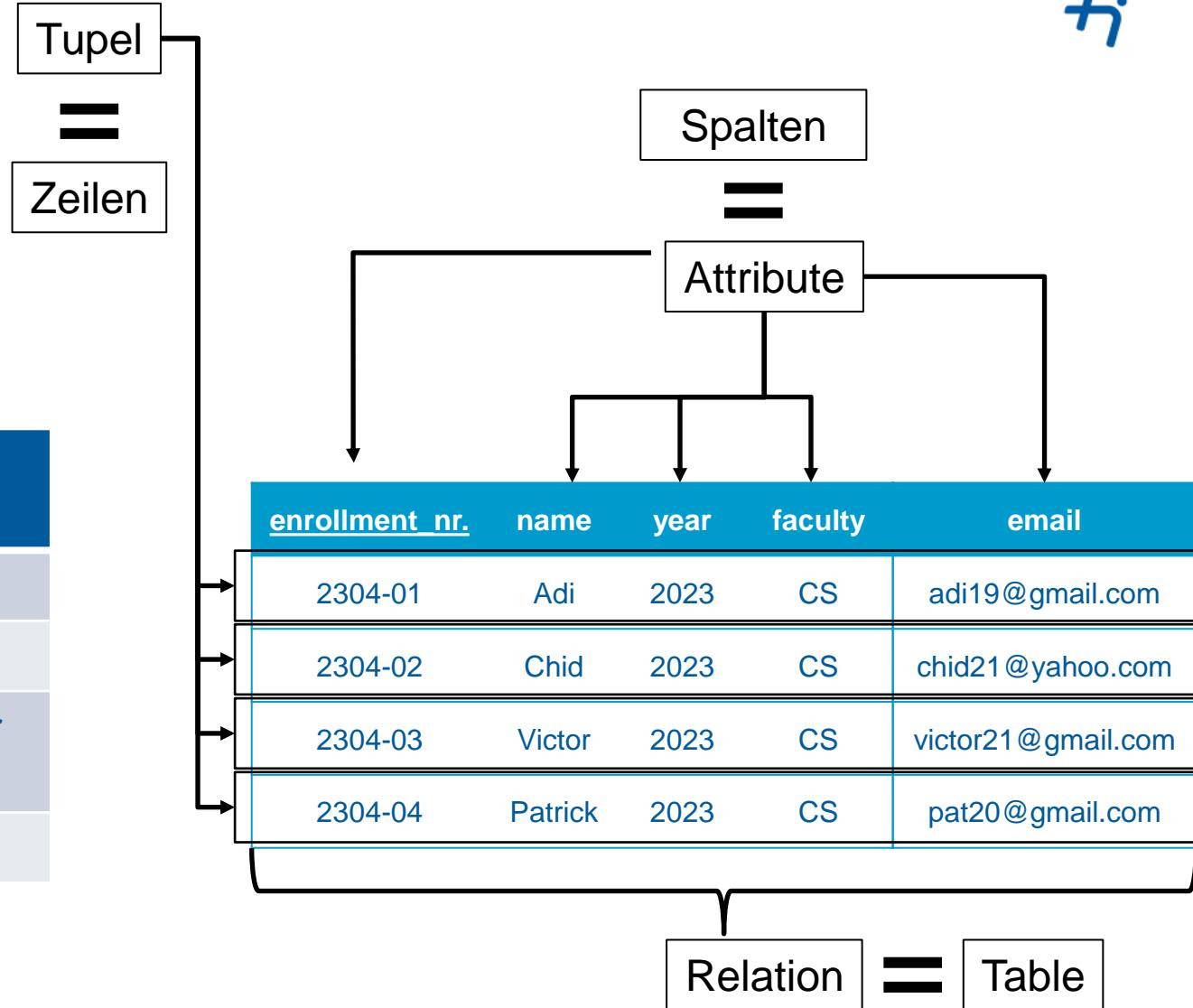
Eine Relation ist eine Menge, es kann also per Definition niemals zwei gleich Zeilen in einer Relation geben. Zudem ist die Reihenfolge der Tabellenzeilen wie Spalten unerheblich.



Begriffliche Grundlagen

Relation, Tupel, Tabelle

Formale relationale Ausdrucksweise	Informelle Ausdrucksweise
Relation	Tabelle
Tupel	Eine Zeile einer Tabelle
Attribut	Eine Spalte (Feld) einer Tabelle
Primärschlüssel	Eindeutiger Bezeichner



Definition

An **Entity** represents a thing or a concept in the real world with an independent existence. An entity has an own identity and represents just one thing.
e.g. a car, a savings account, a person, an animal

Sets of entities sharing the same characteristics or attributes are called **Entity Type**

Students relation

MNR	Vorname	...
2304-01	Adi	...
2304-02	Chid	...
2304-03	Victor	...
2304-04	Patrick	...

Beispiel: Relation Kunden

Kunden			
<u>KundenNr</u>	Vorname	Nachname	Wohnort
1	Hans	Schmidt	Ingolstadt
2	Anne	Müller	Bremen
3	Birgit	Klein	Ingolstadt

mit **vier Attributen** und **drei Tupeln**



Übersicht

- **Vereinigungsmenge:** \cup
- **Schnittmenge:** \cap
- **Differenzmenge:** \
- **Selektion:** σ
- **Projektion:** π
- **Verbund (Join):** \bowtie
- **Kartesisches Produkt:** \times
- **Umbennung:** ρ



Übersicht

Selektion: σ

Hierunter versteht man die Auswahl von Zeilen mittels Prädikats

Projektion: π

Hierunter versteht man die Auswahl von Spalten

Kartesisches Produkt: \times

Kreuzprodukt (Menge aller Tupel)

Verbund (Join): \bowtie

Hierunter versteht man die Verknüpfung von Tabellen anhand gleicher Attribute

Umbenennung: ρ

Hierunter wird die Umbenennung einer Spalte verstanden

Relationenoperationen: Selektion

Definition Selektion

Auswahl von Zeilen einer Relation über ein Prädikat

$\sigma_{\text{Prädikat}=\text{Bedingung}}$ (Relation)

Beispiel mit kanonischer Übersetzung

$\sigma_{\text{Wohnort}=\text{"Hamburg"} \wedge \text{Vorname}=\text{"Dieter"}}$ (Studenten)

Studenten

Matrikel	Vorname	Nachname	Wohnort
28749	Achmed	Barakat	Hamburg
81674	Sarah	Feldbusch	Lübeck
51896	Dieter	Müller	Hamburg

Definition Projektion

Auswahl von Spalten einer Relation:

$$\Pi_{\text{Spaltenname}}(\text{Relation})$$

Beispiel mit kanonischer Übersetzung

$$\Pi_{\text{Wohnort}}(\text{Studenten}) = \{(Hamburg), (Lübeck)\}$$

Studenten			Wohnort
Matrikel	Vorname	Nachname	
28749	Achmed	Barakat	Hamburg
81674	Sarah	Feldbusch	Lübeck
51896	Dieter	Müller	Hamburg

Definition Join

Der (natürliche) Join verknüpft Tupel, die die gleichen Werte bei Attributen mit gleichem Namen haben:

$$R \Join_{A=B} S = \sigma_{A=B}(R \times S)$$

Beispiel mit kanonischer Übersetzung

Studenten \Join Fächer = $\{(2849, \text{Achmed}, 18, 18, \text{Informatik})\}$

Fach=FID

Studenten		
Matrikel	Vorname	Fach
2849	Achmed	18
8174	Sarah	2

Fächer	
FID	Name
18	Informatik
5	Physik



Matrikel	Vorname	Fach	FID	Name
2849	Achmed	18	18	Informatik

Definition Umbenennung

Umbenennung der Spalte einer Operation.

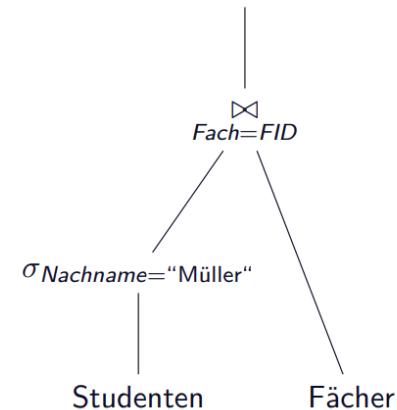
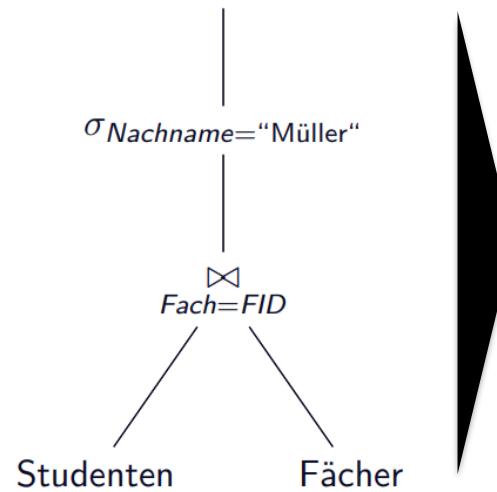
Beispiel mit kanonischer Übersetzung

$\rho_{\text{Name} \leftarrow \text{Vorname}}$ (Studenten)

Studenten

Matrikel	Name	Nachname	Wohnort
28749	Achmed	Barakat	Hamburg
81674	Sarah	Feldbusch	Lübeck
51896	Dieter	Müller	Hamburg

$\sigma_{Nachname='Müller'}(\text{Studenten} \bowtie_{Fach=FID} \text{Fächer})$



Leserichtung: Von unten nach oben

Ziel der algebraischen Optimierung

Effiziente Ausführung eines algebraischen Ausdrucks (Minimierung der Zwischenergebnisse bei gleichem Ausdruck)

Heuristiken:

- Führe Selektionen so früh wie möglich aus
- Führe Projektion so früh wie möglich aus
- Berechne gemeinsame Teilbäume nur einmal
- Verknüpfe bei Mengenoperationen immer zuerst die kleinsten Relationen

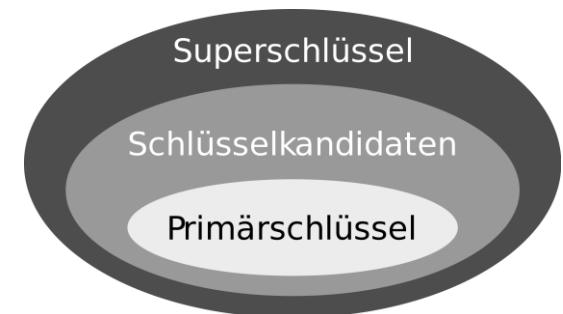
Definition Superschlüssel

Ein Superschlüssel ist eine beliebige Menge von Attributen, die die Tupel einer Relation eindeutig identifiziert. Die Kombination aller Attribute einer Relation können immer einen Superschlüssel bilden.

Beispiel Superschlüssel für Relation Literatur

$\{(ISBN), (ISBN, Autor), (Autor, Buchtitel), (ISBN, Buchtitel), (ISBN, Autor, Buchtitel)\}$

Literatur			
ISBN	Autor	Buchtitel	...
0001	Hans	X	...
0002	Peter	X	...
0003	Peter	Y	...
...



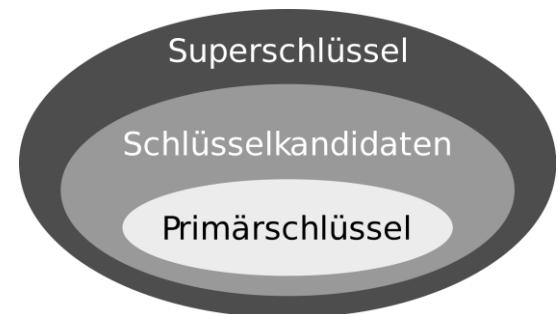
Definition Schlüsselkandidat

Ein Schlüsselkandidat ist eine minimale Menge von Attributen, die die Tupel einer Relation eindeutig identifiziert.

Beispiel für Relation Literatur

{ISBN}, {Buchtitel, Autor},

Literatur			
ISBN	Autor	Buchtitel	...
0001	Hans	X	...
0002	Peter	X	...
0003	Peter	Y	...
...	...		



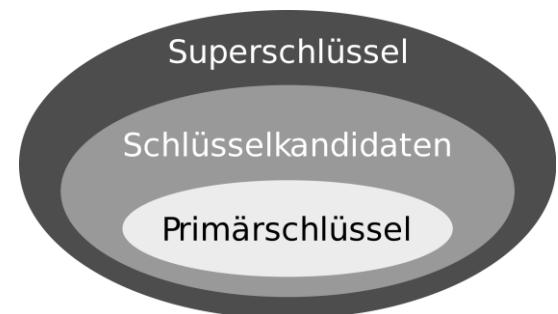
Definition Primärschlüssel

Aus den Schlüsselkandidaten wird ein Primärschlüssel ausgewählt mit dem jedes Tupel der Relation zweifelsfrei identifizieren kann.

Beispiel für Relation Literatur

{ISBN}

Literatur			
ISBN	Autor	Buchtitel ...	
0001	Hans	X	...
0002	Peter	X	...
0003	Peter	Y	...
...	...		



Definition Surrogatschlüssel

Hierbei handelt es sich um ein zusätzliches Attribut, das künstlich hinzugefügt wird, um jedes Tupel der Relation zweifelsfrei zu identifizieren. Oftmals wird dieses `_id` genannt.

Beispiel für Relation Literatur

{Literatur_ID}

Literatur				
Literatur_ID	ISBN	Autor	Buchtitel ...	
1	0001	Hans.	X	...
2	0002	Peter	X	...
3	0003	Peter	Y	...
...

Definition Fremdschlüssel

Als einen Fremdschlüssel bezeichnet man ein Attribut, das in der Regel auf einen Primärschlüsselwert einer anderen Relation verweist. Fremdschlüssel können auch so definiert werden (technologieabhängig), dass sie auf die Spalten einer anderen Tabelle verweisen wenn diese die UNIQUE-Einschränkung besitzen. Werte können dann Null Values annehmen wenn nicht ausgeschlossen (NOT NULL).

Beispiel für Relation eScooter & Employees

eScooter

ScooterID	SerialNumber	Brand	BatteryStatus	isRentable	EmployeeID
1	123asdf32	Xiaomi Mi 1S	66	1	1
2	934we4r4r	Ninebot Segway E25E	0	0	1
3	11444dfse	Ninebot Segway E25E	100	1	2

Employees

EmployeeID	Firstname	Lastname	...
1	Peter	Griffin	...
2	Bonnie	Hamilton	...
3	Mort	Goldman	...



Bei der referenziellen Integrität können Datensätze, die einen **Fremdschlüssel** aufweisen nur dann gespeichert werden, wenn der Wert des **Fremdschlüssels einmalig in der referenzierten Tabelle existiert**. Im Falle, dass ein referenzierter Wert nicht vorhanden ist, kann der Datensatz nicht gespeichert werden. Es handelt sich also um einen **Constraint in der Datenbank**.



Constraints

Definition

Constraints definieren Bedingungen, die beim Einfügen, Ändern und Löschen von Datensätzen in der Datenbank erfüllt werden müssen.

➤ Primary Key

diese enthalten per Definition keine NULL-Werte und sind eindeutig

➤ Foreign Key

in jedem Fall ist Einmaligkeit garantiert, da ein FK auf Primary Key oder eine Unique Spalte zeigt; NULL Werte sind zulässig wenn kein Constraint NOT NULL angegeben

➤ Unique

dieser Constraint definiert, dass ein Wert nur einmal in einer Spalte vorkommen darf

➤ Check

individuelle Prüfungen – z. B. Wertebereiche

➤ Not Null

definiert dass keine NULL-Werte vorkommen dürfen

Aufgabe

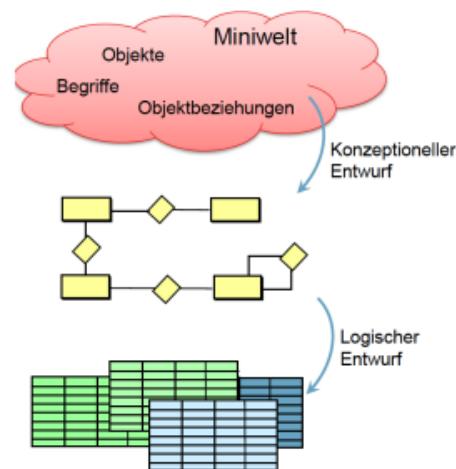
Protonen	Atomgewicht	Name	Symbol	Schmelzpkt.	Siedepkt.
1	1,008	Wasserstoff	H	-259	-253
2	4,003	Helium	He	-272	-269
3	6,941	Lithium	Li	180	1317
4	9,012	Berylium	Be	1278	2970
5	10,811	Bor	B	2300	2550
6	12,011	Kohlenstoff	C	3550	4827
7	14,007	Stickstoff	N	-210	-196
8	15,999	Sauerstoff	O	-218	-183
...		

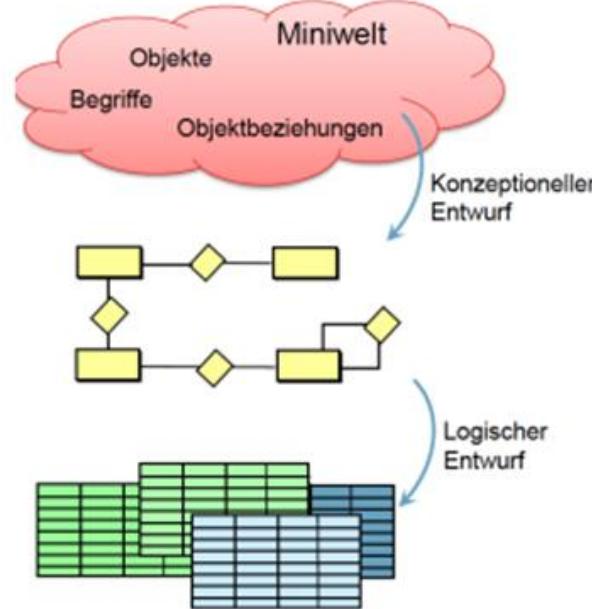
- Welches ist der eindeutige Identifikator (Primärschlüssel)?
- Was sind Superschlüssel, Schlüsselkandidaten?



- Relationen sind veränderlich, sie ändern ihren Zustand mit der Zeit (Einfügen, Löschen, Ändern von Tupeln)
- Schemas sind so gut wie unveränderlich (technisch möglich – Schemaänderungen werden jedoch selten vorgenommen)
- In welcher Reihenfolge die Attribute eines Relationenschemas definiert werden spielt keine Rolle
- Die Reihenfolge der Tupel in einer Relation spielt auch keine Rolle - in einer Menge ist keine Reihenfolge definiert

Von ER-Abbildung zur Relation





- Das Entity-Relationship-Diagramm ist ein graphisches Modell, welches Zusammenhänge anschaulich macht
- Das ER-Modell ist aber auf der konzeptuellen Ebene angesiedelt und daher nicht direkt zu verwenden
- Es ist jedoch eine Abbildung in die verschiedenartigsten logischen Ebenen möglich, wie etwa dem Relationalen Modell oder dem spaltenbasierten Modell



Was muss abgebildet werden?

Wir haben im ER-Modell zwei grundlegende Strukturierungskonzepte:

- Entitätstypen
- Beziehungstypen

Im relationalen Modell bilden wir diese Typen auf *Relationen* ab.
Die Abbildung ist nicht immer eindeutig, womit es durchaus unterschiedliche
relationale Modelle geben kann, die dieselben Sachverhalte ausdrücken.
Kardinalitäten werden durch Fremdschlüssel und Primärschlüsseldefinition
gewährleistet.



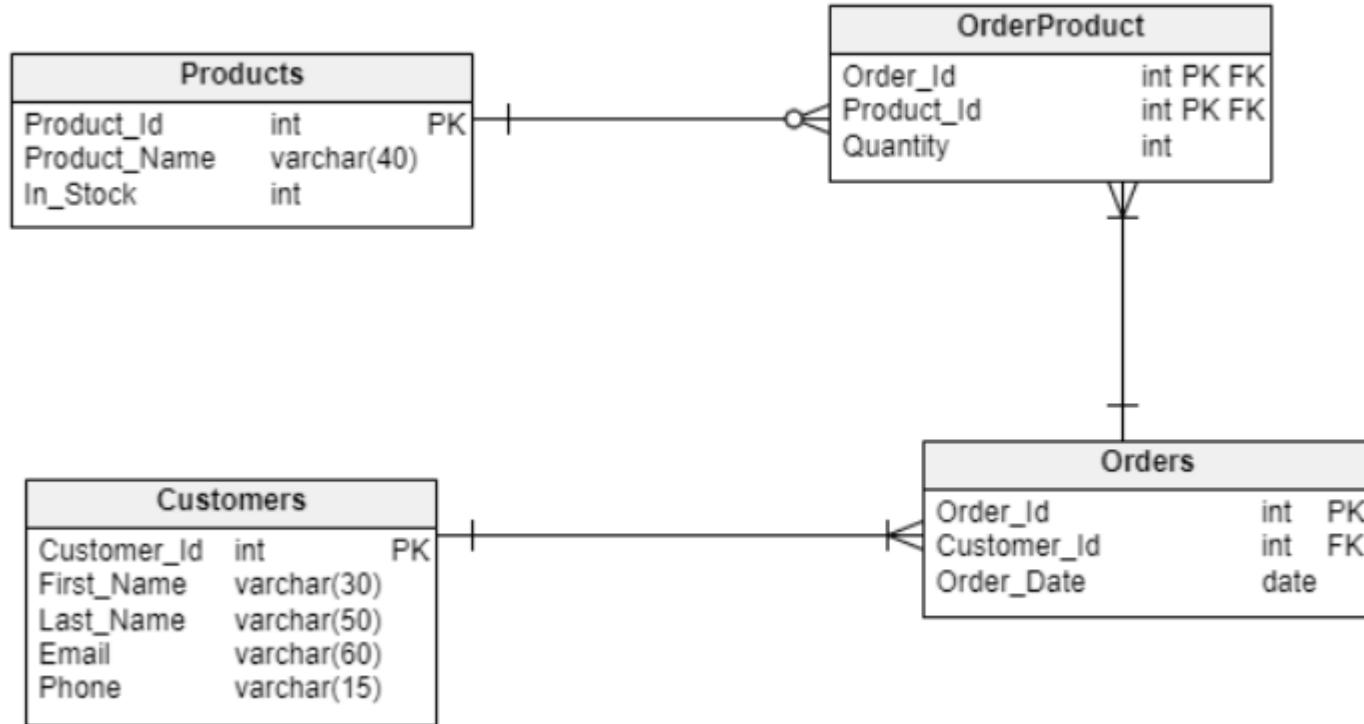
Minimalnotation:

Relationenname (Primärschlüssel, ↑ Fremdschlüssel)

Detaillierte Notation:

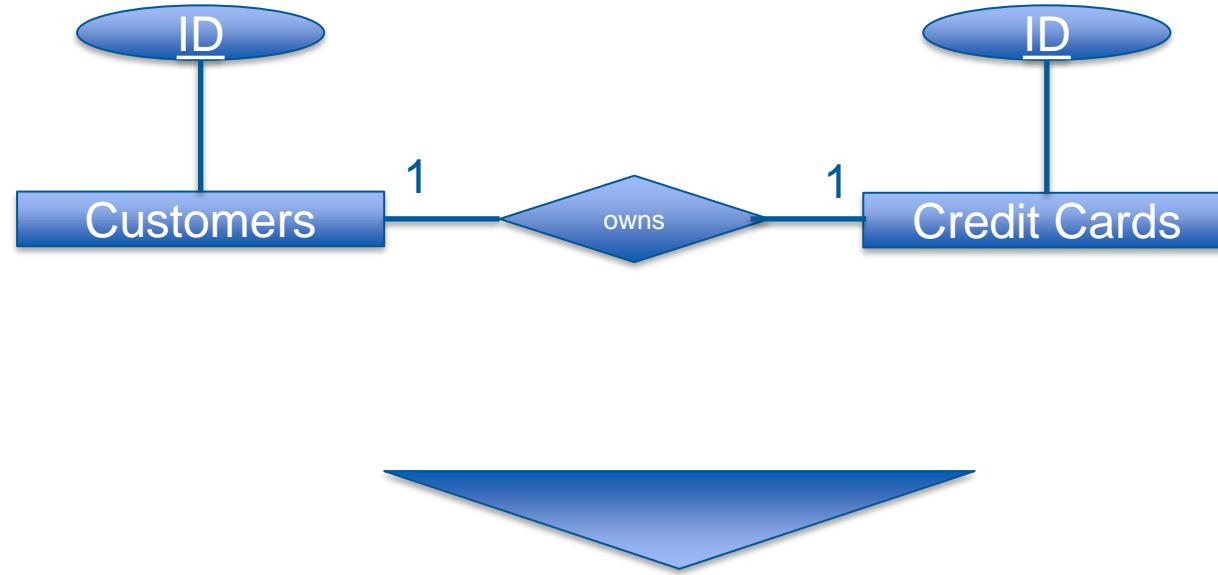
Relationenname (Primärschlüssel: datentyp, ↑ Fremdschlüssel:datentyp)

Foreign Key (Attributname(n) des Fremdschlüssel) REFERENCES Zielrelation (Attributname(n) von Zielrelation)



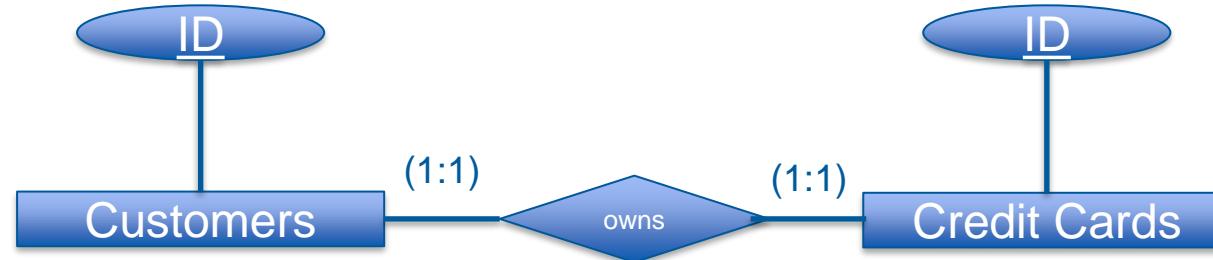
Data Model Types: An Explanation with Examples |
Vertabelo Database Modeler

Relationenmodell von 1:1 Beziehungen



owns (CustomerID, CreditID) oder
owns (CustomerID, CreditID) **Aber nicht beides**

Relational model of 1:1 relationships



Um die Kardinalität von 1 auf beiden Seiten sicherzustellen, müssen die Relationen Kunden und Kreditkarten zu einer Relation zusammengeführt werden. Entweder wird CustomerID oder CreditID als Primärschlüssel definiert. Der nicht-primäre Schlüssel wird mit der Einschränkung NOT NULL definiert.
KundenKreditkarte (KundenID, KreditkartenID)

Relationenmodell von exakte 1:1 Beziehungen

Best Practice: Um die minimale Kardinalität 1 auf beiden Seiten zu gewährleisten, werden in der Regel die Relationen zusammengefasst und die Felder entsprechend als Pflichtfelder (mit NOT NULL) definiert. Ein Attribut von beidem wird Primärschlüssel



Gutachten_Diplomarbeit (Gutachter, Autor)

Komplexer und nicht von jedem Datenbanksystem unterstützt:

Gutachten(Gutachter, ↑Autor)

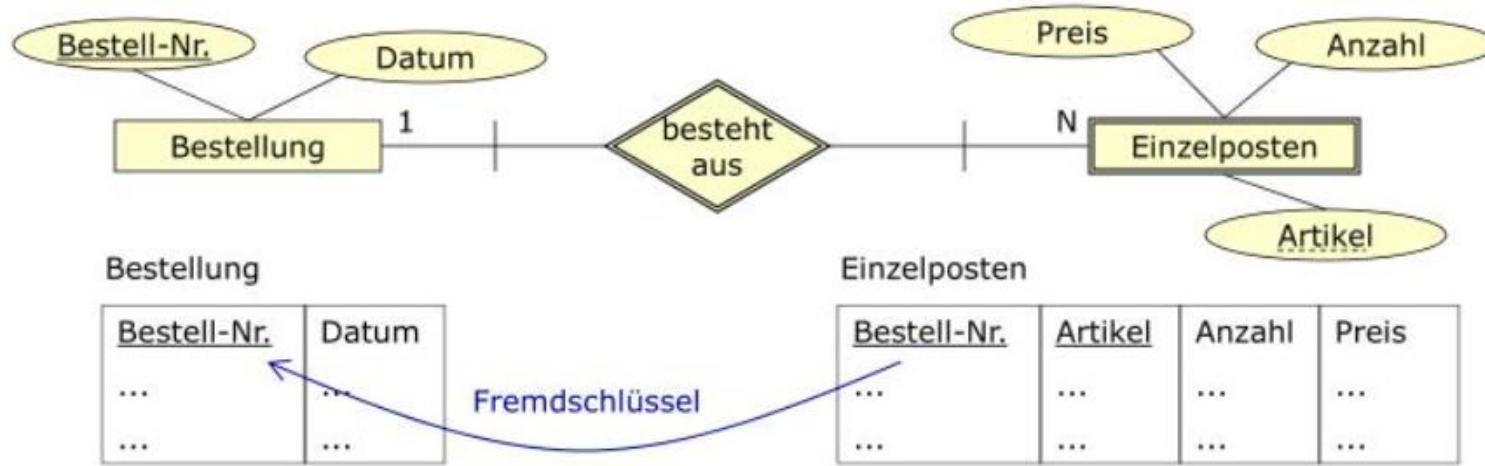
Foreign Key (Autor) REFERENCES Diplomarbeit (Autor) DEFERRABLE INITIALLY DEFERRED

Diplomarbeit (Autor, ↑Gutachter)

Foreign Key (Gutachter) REFERENCES Gutachten (Gutachter) DEFERRABLE INITIALLY DEFERRED

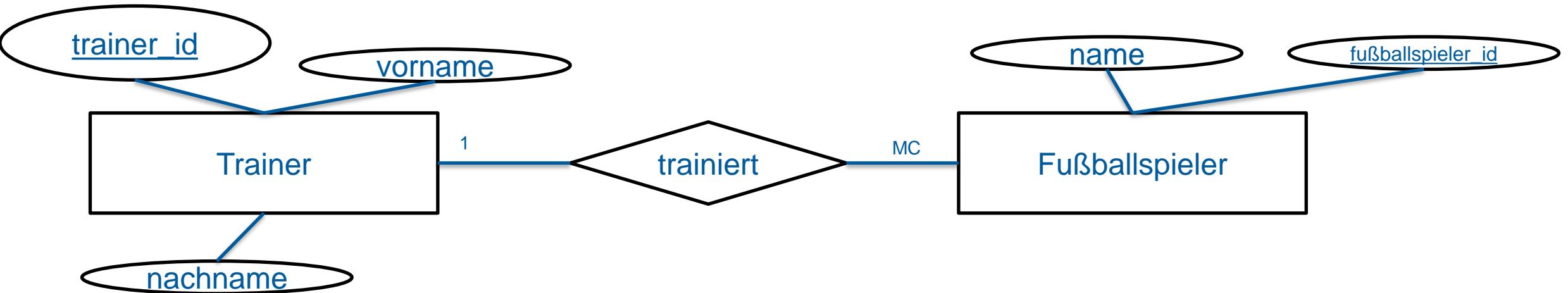
Relationenmodell von schwachen Entitätstypen

- Attribute der schwachen Entität werden um den Schlüssel der starken Entität erweitert
- Primärschlüssel → Schlüssel der starken Entität und partieller Schlüssel der schwachen Entität



Bestellung (Bestell-Nr, Datum)

Einzelposten (↑Bestell-Nr., Artikel, Anzahl, Preis)



Merke

In einer 1:N Beziehung kommt der Fremdschlüssel immer auf die Seite, wo das N steht

Beispiel (1/2)

<u>trainer_id</u>	vorname	nachname
1	Max	Maier
2	Simon	Müller

<u>fußballspieler_id</u>	vorname	↑ trainer_id
1	Jens	1
2	Stefan	1
3	Hans	2

Beispiel (2/2)

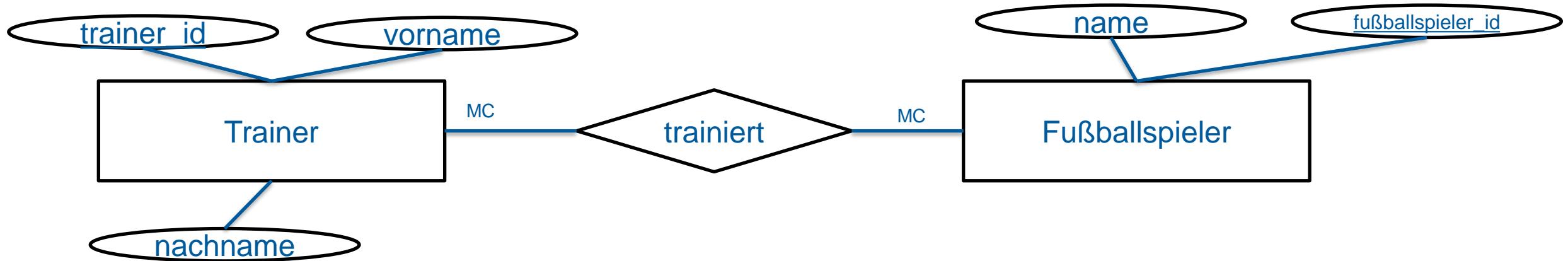
<u>trainer_id</u>	vorname	nachname	↑ fußballspieler_id
1	Max	Maier	1
2	Simon	Müller	2

<u>fußballspieler_id</u>	vorname
1	Jens
2	Stefan
3	Hans

Wenn der Fremdschlüssel in der Relation Trainer wäre, impliziert dies, dass ein Trainer nur einen Spieler haben kann. Das wäre aber eine Verletzung der semantischen Modellierung!



Fremdschlüssel bei n:m- Beziehungen



Merke

Beziehungen der Kardinalität n:m können immer nur auf drei Relationenschemata abgebildet werden. Die beiden beteiligten Entity-Typen werden auf je ein Relationenschema abgebildet. Die dritte Relation wird als **Beziehungsrelation bzw. assoziative Tabelle** bezeichnet. Der Primärschlüssel der Beziehungsrelation ist ein zusammengesetzter Primärschlüssel.

Beispiel

<u>trainer_id</u>	<u>vorname</u>	<u>nachname</u>
1	Max	Maier
2	Simon	Müller

Trainer (trainer_id, vorname, nachname)

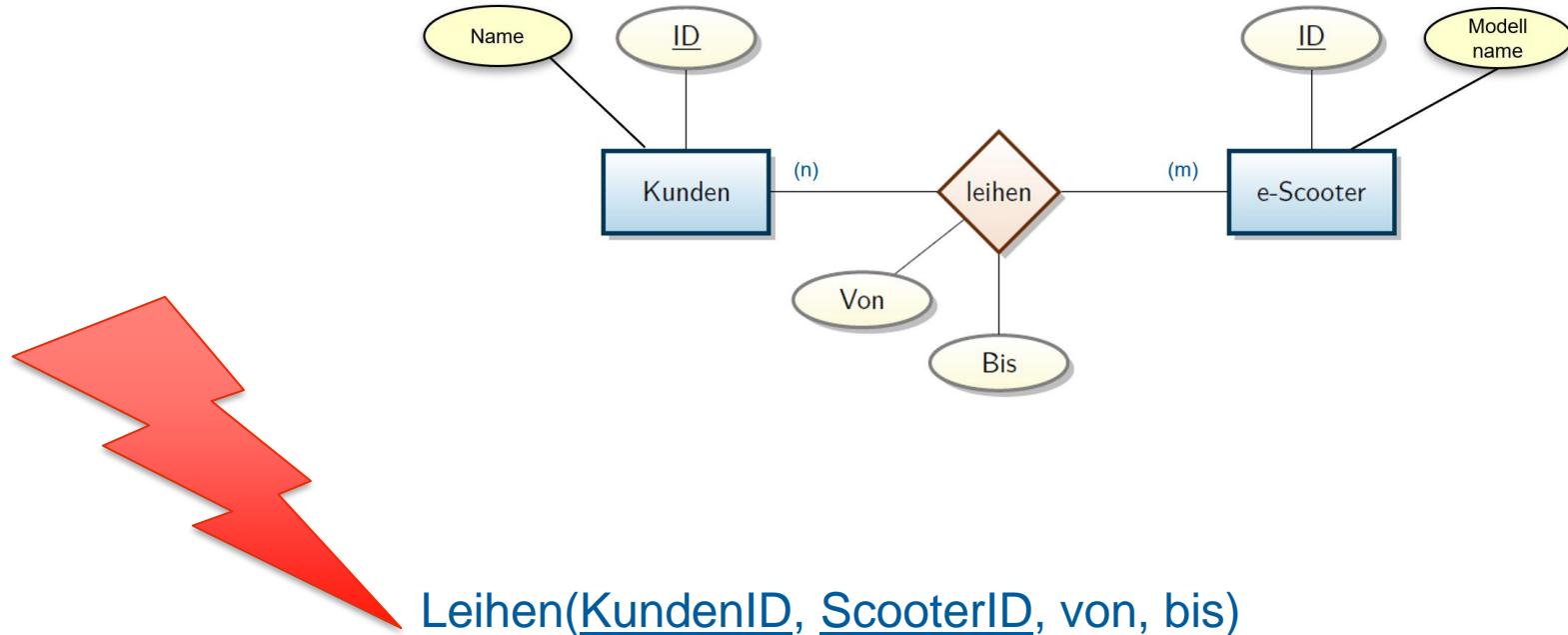
<u>fußballspieler_id</u>	<u>vorname</u>
1	Jens
2	Stefan
3	Hans

Fußballspieler(fußballspieler_id, vorname)

<u>fußballspieler_id</u>	<u>trainer_id</u>
1	1
2	1
3	2

trainer_hat_fussballspieler (fußballspieler_id, trainer_id)

Relationenmodell von n:m-Beziehungen



Leihen(KundenID, ScooterID, von, bis)

Was ist das Problem?

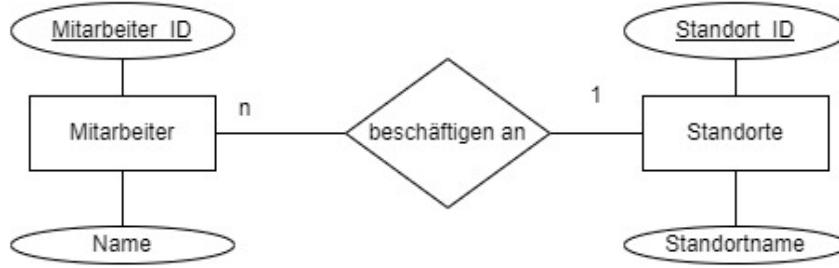
Beispiel Übersetzung von Beziehungstypen M:N

leihen (\uparrow KundenID, \uparrow ScooterID, von, bis)
Foreign Key (KundenID) REFERENCES Kunden(ID)
Foreign Key (ScooterID) REFERENCES e-Scooter(ID)



<u>KundenID</u>	<u>ScooterID</u>	<u>Von</u>	<u>Bis</u>
Jackie	123	03.02.21 14:00	03.02.21 15:00
Jackie	123	03.02.21 17:00	03.02.21 18:00
Jackie	123	01.03.21 10:00	01.03.21 11:05

N:1-Beziehungstyp: Beziehungsrelation

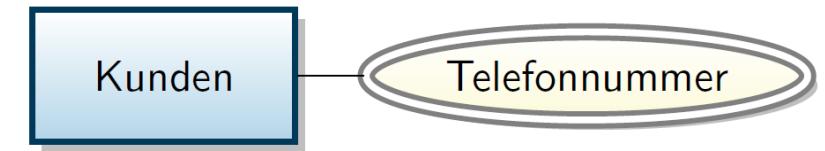


beschäftigt_an(MitarbeiterID:int, StandortID:int, Standortname:varchar)

Mehrwertige Attribute

Mehrwertige Attribute

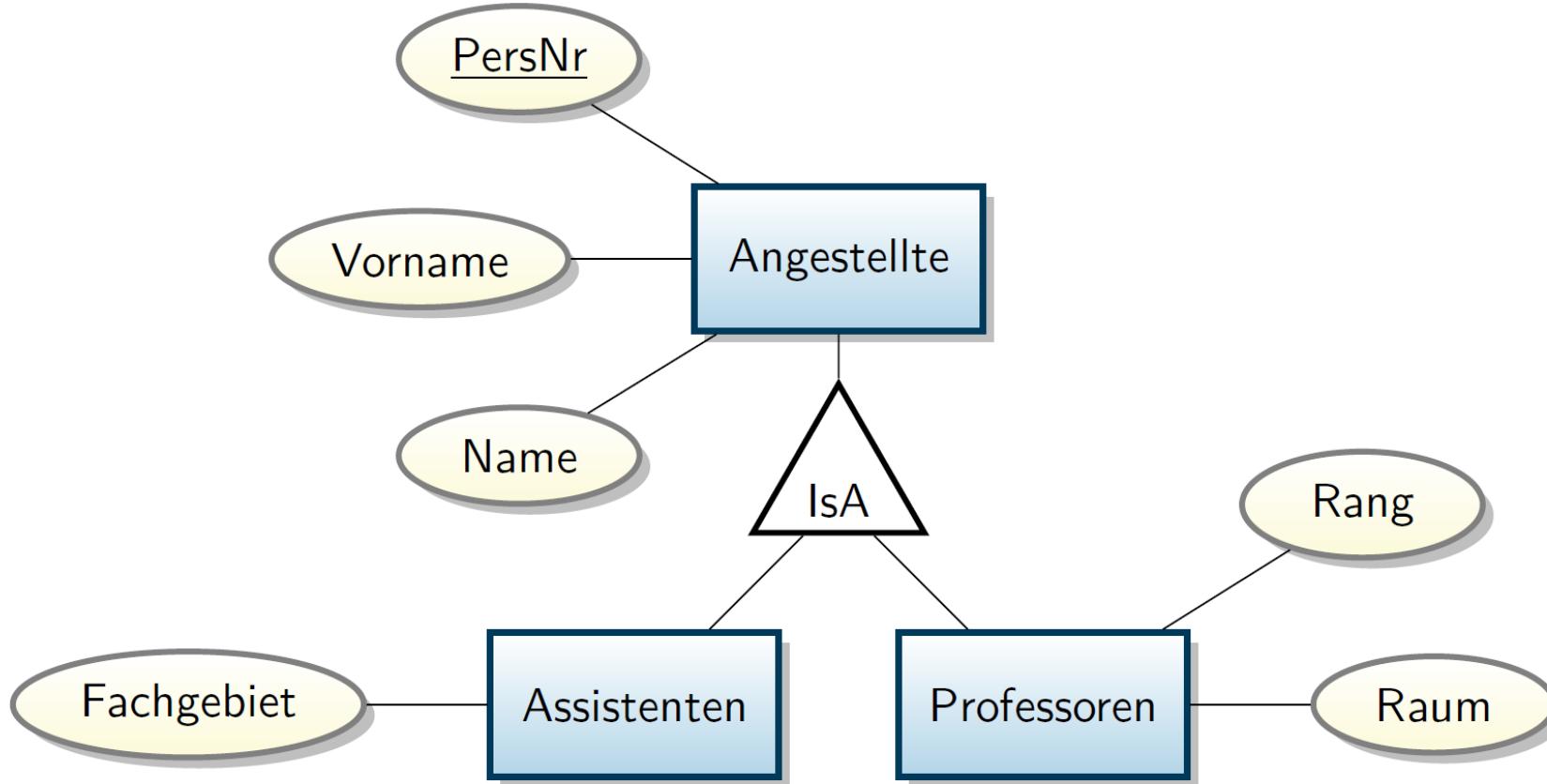
Mehrwertige Attribute können im relationalen Modell durch eine neue Relation dargestellt werden, die als Attribute das Primärschlüsselattribut des Entitätstyps und das Mehrfachattribut besitzen. Beide Attribute sind Schlüssel der Tabelle.



Kunden(ID:int, City:varchar, Name:varchar, Vorname:varchar,
isPremium:boolean, ...)

Telefonnummern(KundenID:int, Telefonnummer:int)

Umsetzung der Generalisierung (1/3)



Umsetzung der Generalisierung (2/3)

- Jeder Entitätstyp wird zu eigener Relation
 - Relation des **Obertyps** enthält **gemeinsame Attribute**
 - Relationen der **Untertypen** enthalten **eigene Attribute**
 - alle Relationen haben **gemeinsamen Schlüssel**
- Beispiel: Assistenten und Professoren als Mitarbeiter
- Nachteil: Information ist verteilt auf mehrere Relationen

```
Mitarbeiter(PersNr:int, Vorname:varchar, Name:varchar)
Professoren(PersNr:int, Rang:varchar, Raum:varchar)
Assistenten(PersNr:int, Fachgebiet:varchar)
```

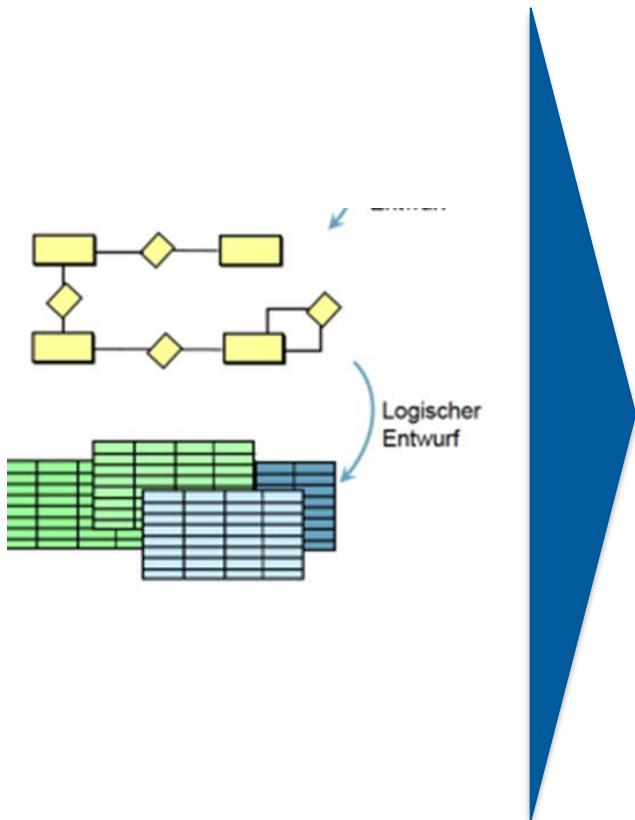
- Nur der **Obertyp** wird realisiert
- **Nachteil: NULL-Werte**

```
Mitarbeiter :{[ PersNr:int, Vorname:varchar, Name:varchar, Rang:varchar,  
Raum:varchar, Fachgebiet:varchar)}
```

- Nur **Untertypen** werden realisiert
- **Nachteil: Angestellte, die weder Assistenten noch Professoren sind**

```
Professoren(PersNr:int, Vorname:varchar, Name:varchar, Rang:varchar,  
Raum:varchar)
```

```
Assistenten(PersNr:int, Vorname:varchar, Name:varchar, Fachgebiet:varchar)
```



Im ERD	Best Practices Überführung
Entitätstyp mit Attributen und Schlüsselattributen	<ul style="list-style-type: none"> Jeder Entitätstyp wird zu einer eigenen Relation mit entsprechenden Attributen übernommen Die Schlüsselattribut(e) werden als Primärschlüssel übernommen und üblicherweise an den Anfang des Relationenschemas gestellt
Schwache Entitätstypen	<ul style="list-style-type: none"> Attribute der schwachen Entität werden, um den Schlüssel des starken Entitätstypen erweitert Primärschlüssel: Schlüssel des starken Entitätstypen und partieller Schlüssel des schwachen Entitätstypen
Beziehungstyp 1:1	<ul style="list-style-type: none"> Sind sehr selten und werden nach intensivem Review in der Regel in einer Relation dargestellt oder in zwei mit Fremdschlüsselbeziehung wenn DBMS das kann (Zirkelbezug)
Beziehungstypen n:m	<ul style="list-style-type: none"> Für n:m-Beziehungen muss eine Beziehungsrelation angelegt werden
Beziehungstyp 1:n	<ul style="list-style-type: none"> Bei zwei Relationen: In einer 1:N Beziehung kommt der Fremdschlüssel immer auf die Seite, wo das N steht Drei möglich

Übung

Die Privatbahn „Bähnle B“ plant ein Zugauskunftssystem. Dabei sollen Daten über die Züge, die Ankunfts- und Abfahrtszeiten, die Bahnhöfe und ihre Standorte sinnvoll gespeichert werden. Es ist Folgendes zu beachten: Jeder Bahnhof hat eine gewisse Anzahl von Gleisen und liegt in einer Stadt. Dabei können in einer Stadt auch mehrere Bahnhöfe liegen. Bei Städten soll außer dem Namen auch die Region gespeichert sein. Jeder Zug hat eine Bezeichnung und eine Anzahl von Wagen. Für einen Zug sollen an jedem Bahnhof die Ankunfts- und Abfahrtszeit und das Gleis gespeichert werden.

- Modellieren Sie zu dem oben beschriebenen Sachverhalt ein ER-Diagramm (Chen-Notation)
- Überführen Sie das ER-Diagramm in ein optimiertes relationales Datenbankschema.

Kennzeichnen Sie dabei Primär- und Fremdschlüssel. (Schriftnotation)