3 Das relationale Datenmodell

→ Codd 1970

3 Das relationale Datenmodell

3.1.1 Einige Definitionen

Kartesisches Produkt:

$W_1, W_2, \ldots, W_n$ beliebige Mengen.

$W_1 \times W_2 \times \ldots \times W_n$

$$::= \{\underbrace{(w_1, w_2, \ldots, w_n)}_{\text{n-Tupel}} \mid w_i \in W_i \; (i = 1, 2, \ldots, n)\}.$$

Komponente

Relation (im math. Sinn)

$X \subseteq (W_1 \times W_2 \times \dots \times W_n)$

- „**n-stellige Relation** über $W_1, W_2, \dots, W_n$“
- W_i: (Werte-)Bereich; „**Domain**“
- n = **Grad** der Relation.

„Tupel“ der Relation X

$x \in X: x = (x_1, x_2, \dots, x_n)$,

$x_i \in W_i$ $(i = 1, 2, \dots, n)$

3.1.2 Attribute, Domänen, Tupel und Relationen

Einziges Strukturierungselement: „Relation“

Relation im DB-Sinn:

a – **Attribut**; Wertebereich (**Domain**) dom(a)

A = $\{a_1, a_2, \ldots, a_n\}$ Menge von Attributen (paarweise verschieden)

z.B. NAME **:** ANG-NAME; ABT-NAME; ...

Reihenfolge der a_i: beliebig, aber im folgenden fest angenommen:

$a_1, a_2, \ldots, a_n$

$\text{dom}(A) ::= \text{dom}(a_1) \times \text{dom}(a_2) \times \ldots \times \text{dom}(a_n)$

„**Relationstyp**": $(A \mid \Sigma)$ bzw. $(a_1, a_2, \ldots, a_n \mid \Sigma)$

ggf. definiert durch „name=typ", etwa

$$\mathbf{R = (A \mid \Sigma) \;\; bzw. \;\; R = (a_1, a_2, \ldots, a_n \mid \Sigma)}$$

- R – **Name** des Relationstyps
- A – **Format** von R; $A = \{ a_1, a_2, \ldots, a_n \}$
- Σ – Menge von (semantischen) **Integritätsbedingungen**

3.1.2 Attribute, Domänen, Tupel und Relationen

Konkrete „Relation **r** vom Typ R“
(bzw. vom Typ $(A \mid \Sigma)$ oder vom Typ $R(A \mid \Sigma)$)

Schreibweisen:

- $r: (A \mid \Sigma)$
- $r: R$
- $r: R(A \mid \Sigma)$

Definition:

sei $R = (A \mid \Sigma)$ Relationstyp:

val(R) $::= \{X \subseteq dom(A) \mid \Sigma\,(X)\}$.

und für r: R

typ(r) $::= R$ (oder $(A \mid \Sigma)$)

format(r) $::= A$,

d.h. $r \subseteq dom(A)$

3.1.2 Attribute, Domänen, Tupel und Relationen

Definition:

r: (A | Σ) :$\Leftrightarrow$

(1) **r** ist Relation vom Format A, d.h.: r $\subseteq$ dom(A)

(2) **r** genügt allen Integritätsbedingungen aus Σ,

In Zeichen Σ(X) (“Σ trifft auf X zu“, “Σ gilt für X“)

Dabei: Σ(r) **:**$\Leftrightarrow$ für alle $\sigma \in \Sigma$ gilt: σ(r)

r: i.a. „Variable“
(i.S. höherer Programmiersprache),
d.h. Wert kann sich ändern;
Wert von **r** zum Zeitpunkt t: $\mathbf{r}^t$

x $\in$ **r**: **Tupel**;
für B $\subseteq$ A:
x.B = Werte von x bezüglich der B-Attribute

Beispiel 3-1:

„Angestellte eines Unternehmens“

- **Relationstyp:**

  ANGESTELLTE =

  (ANG-NR, NAME, WOHNORT, ABT-NR | Σ_{ANG})

  mit (etwa): Σ_{ANG} = {„ANG-NR eindeutig“}

- **Relation:**

  angestellte1: ANGESTELLTE

3.1.2 Attribute, Domänen, Tupel und Relationen

Darstellung einer Relation:
üblicherweise als zwei-dimensionale Tabelle

- Spalten
  - Attribute
  - Identifizierung durch Attributnamen
- Zeilen
  - Tupel der Relation
- Zeile/Spalte=Zelle
  - Attributwert
- Da „Relation = Menge ...“
  ⇒ Zeilen der Tabelle paarweise verschieden,
  Reihenfolge der Zeilen ohne Bedeutung

Beispiel 3-2:

Tabelle für angestellte 1

angestellte 1			
ANG-NR	**NAME**	**WOHNORT**	**ABT-NR**
3115	Meyer	Karlsruhe	35
3207	Müller	Mannheim	30
2814	Klein	Mannheim	32
3190	Maus	Karlsruhe	30
2314	Groß	Karlsruhe	35
1324	Schmitt	Heidelberg	35
1435	Mann	Bruchsal	32
2412	Müller	Karlsruhe	32
2454	Schuster	Worms	31

3.1.3 Schlüssel einer Relation

Definition: sei R = (A | Σ)
Gegeben: **r**: R; K $\subseteq$ A.

(1) K **identifizierend** für **r** (bzw. für R!) :$\Leftrightarrow$

$\forall$ X $\in$ val(R) $\forall$ x,y $\in$ X gilt: (x.K = y.K $\Rightarrow$ x = y)

(m.a.W.: : $\forall$ Zeitpunkte t $\forall$ x,y $\in$ $\mathbf{r}^t$: (x.K = y.K $\Rightarrow$ x = y))

(2) K heißt **Schlüssel** für **r** (bzw. R) :$\Leftrightarrow$

a) K identifizierend für **r**

b) K minimal mit dieser Eigenschaft,d.h.
$\forall$ K' $\subset$ K $\exists$ X $\in$ val(R) $\exists$ x,y $\in$ X:
x $\neq$ y, aber x.K' = y.K'

(d.h., es kann kein Attribut aus K weggelassen werden, ohne die Eigenschaft "identifizierend" zu verlieren)

(3) a $\in$ A:

a „**Schlüsselattribut**“ :$\Leftrightarrow$ $\exists$ Schlüssel K mit a $\in$K

a „**Nichtschlüsselattribut**“ (**NSA**) sonst

(4) 1 Schlüssel auszeichnen: **Primärschlüssel (PS)**.

Angabe des **PS** ist eine ***spezielle Integritätsbedingung****;*

der **PS** wird i. a. gekennzeichnet durch Unterstreichen der zugehörigen Attribute.

3.1.3 Schlüssel einer Relation

Beispiel 3-3:

angestellte 1: ANGESTELLTE:

angestellte 1			
ANG-NR	**NAME**	**WOHNORT**	**ABT-NR**
3115	Meyer	Karlsruhe	35
3207	**Müller**	**Mannheim**	**30**
2814	Klein	Mannheim	32
3190	Maus	Karlsruhe	30
2314	Groß	Karlsruhe	35
1324	Schmitt	Heidelberg	35
1435	Mann	Bruchsal	32
2412	Müller	Karlsruhe	32
2454	Schuster	Worms	31
3000	**Müller**	**Mannheim**	**30**

3.1.3 Schlüssel einer Relation

{ANG-NR} - Schlüssel, Primärschlüssel

{ANG-NR, NAME} - identifizierend, kein Schlüssel

{NAME} - nicht identifizierend

{NAME, WOHNORT} - identifizierend?

Falls keine zusätzliche Integritätsbedingung: **NEIN!**

3.1.4 NULL – Werte

Problem: Attribute deren Wert

- *(noch) nicht bekannt* ist (z.B. Klausurergebnis)
  oder
- gar *nicht anwendbar* ist
  (z.B. Fax-Nr einer Person, die keinen Fax-Anschluß hat)

Lösung der Praxis:

In diesen Fällen wird der Wert **NULL** eingetragen.

NULL unterscheidet sich von allen anderen Werten einer **Domäne**;

er ist insbesondere **ungleich**

- der Zahl 0
- Leerzeichen

3.1.4 NULL – Werte (2|2)

Bemerkungen:

– Die Attributwerte des Primärschlüssels dürfen ***nie*** NULL sein!

– NULL - Werte bringen Probleme bei der Auswertung von Ausdrücken;
insbesondere bei Bool‘schen Ausdrücken.

Was ist z. B. das Ergebnis der Anfrage

„Ist ABT-NR **>** 35 ?“,

wenn der Attributwert ABT-NR den Wert NULL hat?

Hier ist Anwendung einer **dreiwertigen Logik** notwendig!

3.1.5 Die erste Normalform (1 NF)

Definition: 1NF-Relation

r: (A | Σ)

a ∈ A: a „**atomar**“ d.h. ∀ x ∈ **r**: x.a = **ein** Wert α ∈ dom(a)

r ist in „**1. Normalform**“ oder „**1NF-Relation**“

Potentielles Bsp. für „mengenwertiges“ Attribut:

„FÄHIGKEITEN“: Menge von Einzelfähigkeiten,

z.B. „Englisch“, „COBOL“ und „Steno“, „nichts“ (∅)

3.1.5 Die erste Normalform (1 NF)

Beispiel 3-4: Typ ANGESTELLTE erweitern um FÄHIGKEITEN → nicht 1NF-Form

*angestellte 1 **		
ANG-NR	**...**	**FÄHIGKEITEN**
3115	...	„ Englisch“ „ Stenographie“
3207	...	„C“ „COBOL“
2814	...	„ Englisch“
3190	...	
...	...	...

3.1.5 Die erste Normalform (1 NF)

Verschiedene Lösungsmöglichkeiten für „1NF“:

Variante 1: Attribut-Verdopplung

angestellte 2					
ANG -NR	**NAME**	**WOHNORT**	**ABT -NR**	**FÄHIG-KEITEN1**	**FÄHIG-KEITEN 2**
3115	Meyer	Karlsruhe	35	Englisch	Stenographe
3207	Müller	Mannheim	30	C	Cobol
2814	Klein	Mannheim	32	Englisch	NULL
3190	...	...	...	NULL	NULL
...	...	...	...	...	...

? z.B. : 2314 ... Englisch & Steno & COBOL

3.1.5 Die erste Normalform (1 NF)

Variante 2: Tupel-Verdopplung

angestellte 1:				
ANG-NR	**NAME**	**WOHNORT**	**ABT-NR**	**FÄHIGKEIT**
3115	Meyer	Karlsruhe	35	Englisch
3115	Meyer	Karlsruhe	35	Stenographie
3207	Müller	Mannheim	30	C
3207	Müller	Mannheim	30	COBOL
2814	Klein	Mannheim	32	Englisch
3190	...	...	...	NULL
...	...	...	...	...

? Speicherplatz
Update-Anomalien

Schlüssel ?

3.1.5 Die erste Normalform (1 NF)

Variante 3: Zerlegung in 2 Relationen

angestellte 1:

ANG-NR	NAME	WOHNORT	ABT-NR
3115	Meyer	Karlsruhe	35
3207	Müller	Mannheim	30
2814	...	...	...
3190	...	...	...

angestellte 2:

ANG-NR	FÄHIGKEIT
3115	Englisch
3115	Stenographie
3207	C
3207	COBOL
2814	Englisch
...	...

kein 3190!

3.1.6 Modellierung der Realwelt im relationalen Datenmodell (1|5)

Beispiel 3-5:

Objekte: Angestellte, Projekte
Beziehungen: Angestellte arbeiten an einem Projekt mit.

⇒ 3 Relationstypen

- ANGESTELLTE = (**ANG-NR**, NAME, WOHNORT, ABT-NR **|** Σ)
  Σ = {"**ANG-NR** ist PS"}
- PROJEKT = (P-NAME, **P-NR**, P-FILIALE, P-LEITER | {"**P-NR** ist PS"})
- ANG-PRO = (**P-NR, ANG-NR**, PROZ-ARBZEIT| {"{**P-NR, ANG-NR**} ist PS", ... })

- angestellte1: ANGESTELLTE
- projekt1: PROJEKT
- ang-pro1: ANG-PRO

PS: ANG-NR

angestellte 1:

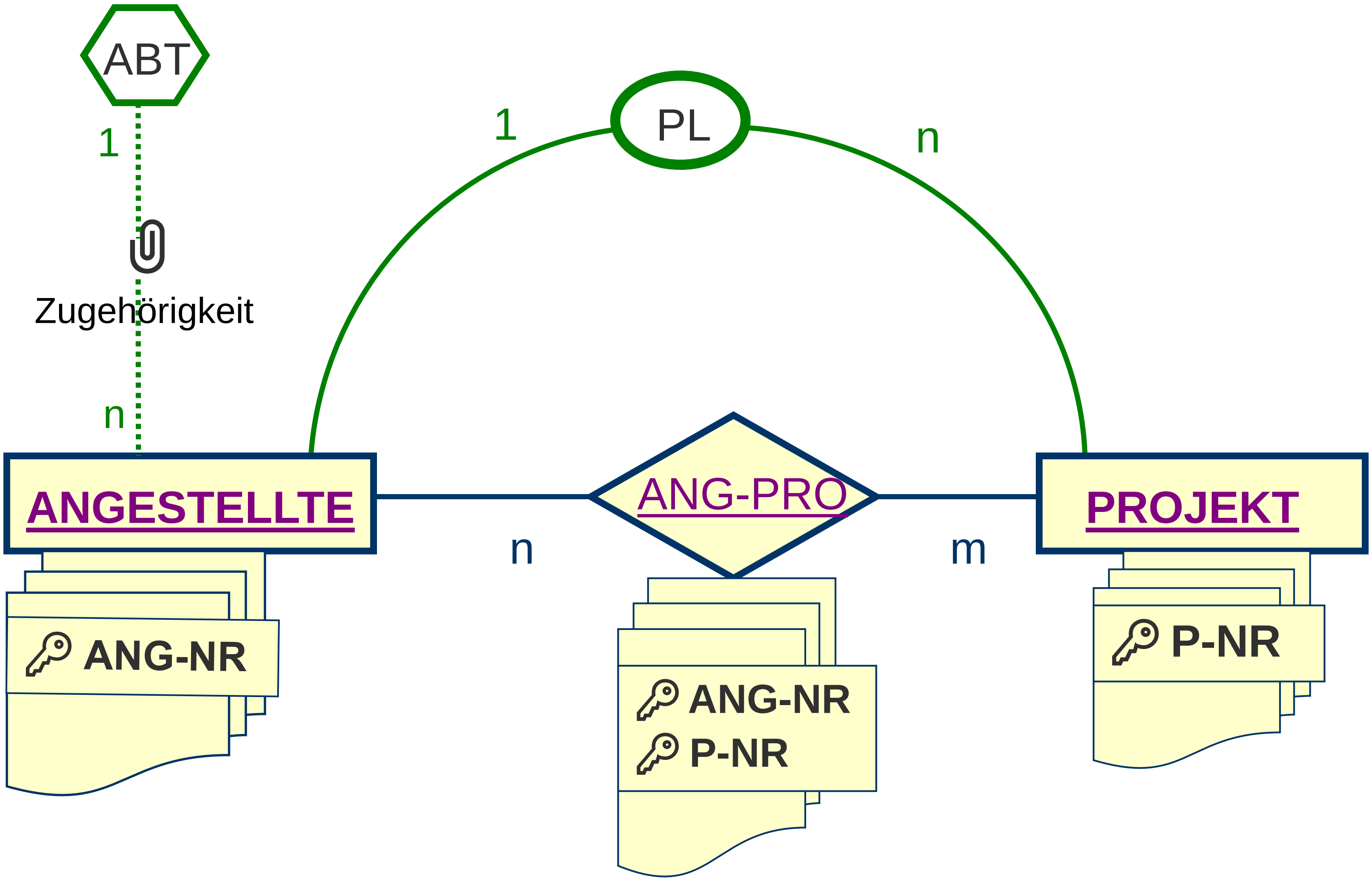
ANG-NR	**NAME**	**WOHNORT**	**ABT-NR**
3115	Meyer	Karlsruhe	35
3207	Müller	Mannheim	30
2814	Klein	Mannheim	32
3190	Maus	Karlsruhe	30
2314	Groß	Karlsruhe	35
1324	Schmitt	Heidelberg	35
1435	Mann	Bruchsal	32
2412	Müller	Karlsruhe	32
2454	Schuster	Worms	31

3.1.6 Modellierung der Realwelt im relationalen Datenmodell (3|5)

PS: P-NR

projekt 1:			
P-NAME	**P-NR**	**P-FILIALE**	**P-LEITER**
P-1	761235	Karlsruhe	3115
P-2	770008	Karlsruhe	3115
P-3	770114	Heidelberg	1324
P-4	770231	Mannheim	2814

3.1.6 Modellierung der Realwelt im relationalen Datenmodell (4|5)



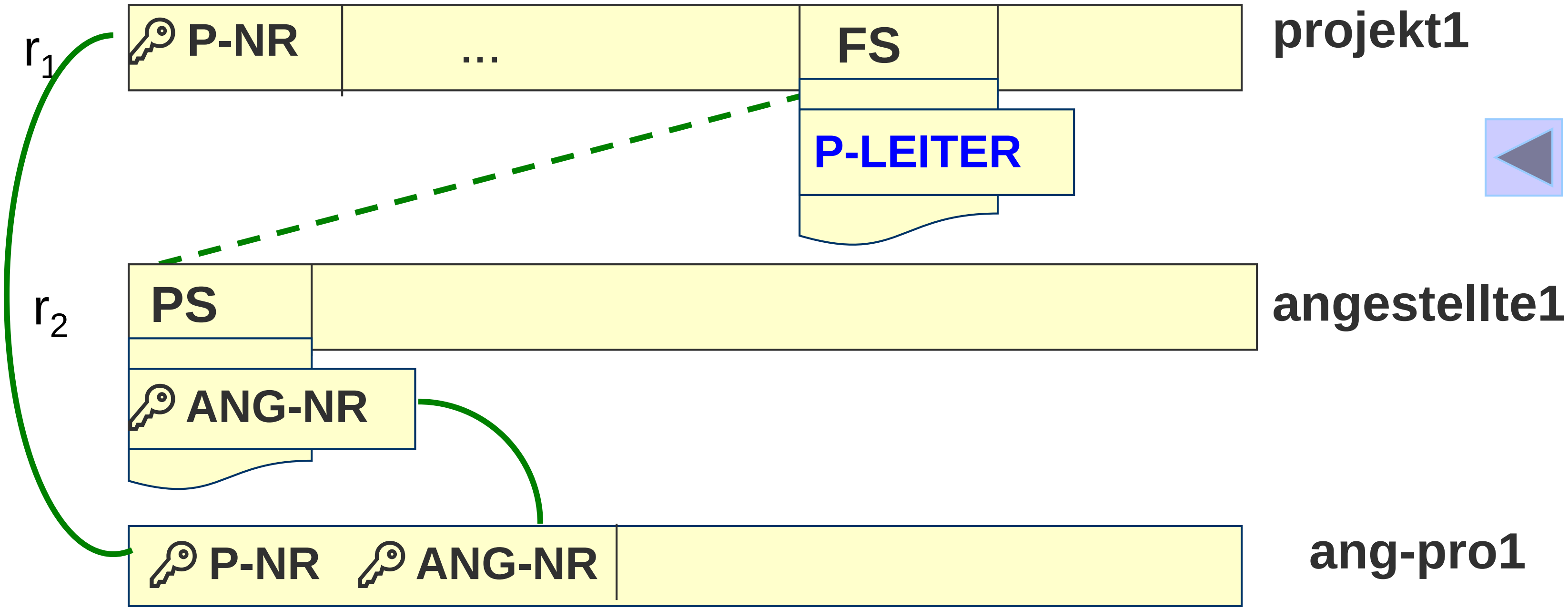
3.1.6 *Modellierung der Realwelt im relationalen Datenmodell* (5|5)

ang-pro 1		
P-NR	**ANG-NR**	**PROZ-ARBEIT**
761235	3207	100
761235	3115	50
761235	3190	50
761235	1435	40
770008	2814	70
770008	2454	40
770114	2814	30
770114	1435	60
770114	2454	60
770114	2412	100
770231	3190	50
770231	2314	100
770231	3115	50
770231	1324	100

PS: { P-NR, ANG-NR}

3.1.7 *Fremdschlüssel*

Attribute:	P-NR, ANG-NR	in ang-pro1
	P-LEITER	in projekt1
	ABT-NR	in angestellte1



Attribute **FS** eines Relationsschemas $\mathbf{r_1}$: (A), die dem Primärschlüssel (**PS**) eines Relationsschemas $\mathbf{r_2}$: (B) entsprechen (und so eine Beziehung zu $\mathbf{r_2}$ herstellen) werden als **Fremdschlüssel** bezeichnet.

3.1.7 *Fremdschlüssel*

Man sagt auch:

die Attribute FS *referenzieren* r_2

$$\mathbf{r_1.FS \subseteq r_2.PS}$$

Es gilt:

- dom(FS) = dom(PS).

- Zu jedem Attributwert von FS (ungleich NULL) in $\mathbf{r_1}$ existiert ein Tupel in $\mathbf{r_2}$, für das PS denselben Attributwert hat.

3.1.7 Fremdschlüssel

Bemerkungen:

- Anwendungsprogrammierer muss Zusammenhänge kennen!
- Namensgleichheit zwischen FS und PS nicht notwendig, verbessert aber die Lesbarkeit eines rel. Schemas.
- Ein Fremdschlüsselwert in einem Relationsschema **r**:R darf nur dann NULL sein, wenn dieser Fremdschlüssel nicht Teil des Primärschlüssels von R ist.
- Ein Relationsschema r_1: R_1 kann mehrere Fremdschlüssel haben, die unterschiedliche Relationsschemata r_2: R_2 ,…, r_n: R_n referenzieren.

3 Das relationale Datenmodell

3.2 Relationale Algebra

„Algebra": nichtleere Menge,
evtl. mit bestimmten Strukturen,
und Operationen auf dieser Menge

Relationenalgebra:

– Relationen

– Operationen auf Relationen → **Mengenoperationen** / **relationenspezifische Operationen**

3.2.1 Mengenoperationen

$\mathbf{r_1}$: (A | …); $\mathbf{r_2}$: (B | …);

Vereinigung: $r_1 \cup r_2 ::= \{x \mid x \in r_1 \vee x \in r_2\}$

Durchschnitt: $r_1 \cap r_2 ::= \{x \mid x \in r_1 \wedge x \in r_2\}$

Differenz: $r_1 \setminus r_2 ::= \{x \in r_1 \wedge x \notin r_2\}$

Voraussetzung: r_1, r_2 „kompatibel“
d.h. format(r_1) = format(r_2)
(d.h. A = B)

Beispiel 3-6.

r 1	a	b	c
	α	β	γ
	α	β	γ‘
	α‘	β‘	γ‘

r 2	a	b	c
	α‘	β‘	γ
	α‘	β‘	γ‘
	α“	β	γ“

Vikar

r1 ∪ r2	a	b	c
	α	β	γ
	α	β	γ‘
	α‘	β‘	γ‘
	α‘	β‘	γ
	α“	β	γ“

r1 ∩ r2	a	b	c
	α‘	β‘	γ‘

r1 \ r2	a	b	c
	α	β	γ
	α	β	γ‘

3.2.1 Mengenoperationen

Kartesisches Produkt:

$r_1 \times r_2 ::= \{x = x_1 x_2 \mid x_1 \in r_1 \wedge x_2 \in r_2\}$

Attributnamen:

falls nicht eindeutig (z.B. NAME in A und B)

dann eindeutig machen durch qualifizieren:

r_1.NAME, r_2.NAME   (=qualifizierter Name)

Vikar

Beispiel 3-6.

r1	a	b	c
	α	β	γ
	α	β	γ‘
	α‘	β‘	γ‘

r3	d	e
	δ	ε
	δ‘	ε

r1 × r3	a	b	c	d	e
	α	β	γ	δ	ε
	α	β	γ	δ‘	ε
	...	...	...	...	...

3.2.2 *Relationenspezifische Operationen*

(A) Projektion: *Auswahl von Spalten*

geg: **r**: (A | …)

$L \subseteq A$: $\pi_{[L]}\mathbf{r} ::= \{x.L \mid x \in \mathbf{r}\}$

Beispiel 8-7:

$\pi_{[\text{ ANG-NR,NAME }]}$ angestellte1

ANG-NR	NAME
3115	Meyer
3207	Müller
2814	Klein
3190	Maus
...	...

WOHNOR	T-NR
Karlsruhe	
Mannheim	
Mannheim	
Karlsruhe	
...	

$\pi_{[\text{ NAME }]}$ angestellte1

NAME
Meyer
Müller
Klein
Maus
...

3.2.2 *Relationenspezifische Operationen*

Weitere Schreibweise:

r.L für $\pi_{[L]}\mathbf{r}$

Eigenschaft von π :

geg. **r**: (A | …)

Falls $A1 \subseteq A2 \subseteq A$,

dann $\pi_{[A1]}(\pi_{[A2]}\mathbf{r}) = \pi_{[A1]}\mathbf{r}$

bzw. $(\mathbf{r}.A2).A1 = \mathbf{r}.A1$

3.2.2 Relationenspezifische Operationen

(B) Selektion (Restriktion): Auswahl von Tupeln

sei b = geeignete Bedingung
(„**Selektionsbedingung**“):

$\sigma_{[b]}r \ ::= \{x \in \mathbf{r} \mid x \text{ erfüllt } b\}$

Beispiel 3-8:

$\boldsymbol{\sigma_{[\text{WOHNORT} \neq \text{"Heidelberg"} \wedge \text{ NAME} = \text{"Müller"}]}\ \textbf{angestellte1}}$

ANG-NR	NAME	WOHNORT	ABT-NR
2412	Müller	Karlsruhe	32
3207	Müller	Mannheim	30

Kontextfreie Grammatik für Selektionsbedingungen

1. Θ ::= $= \mid \neq \mid \leq \mid \geq \mid < \mid >$
2. operand ::= attributname | konstante | einf-ar-ausdruck
3. vergleich ::= operand Θ operand
4. faktor ::= vergleich | (selBedingung) | $\neg$ faktor
5. term ::= faktor | faktor $\wedge$ term
6. selBedingung ::= term | term $\vee$ selBedingung

attributname: $\in$ A

konstante: Typ muss passen

einf-ar-ausdruck: einfacher arithmetischer Ausdruck

X Θ Y: Θ muss für Werte von X, Y definiert sein

! Eingeschränkte Auswahl von Zeilen !

Es geht nicht:

Bedingung, die mehrere Zeilen simultan anspricht, z.B. **ang-pro1**:

„Alle Angestellte mit zugehörigen Projekten, die zu weniger als 100% an Projekten arbeiten."

oder

„Alle Angestellten, die an mehr als 1 Projekt mitarbeiten."

(C) Join (Verbund)

Verknüpfung von 2 Relationen $\mathbf{r}_1$, $\mathbf{r}_2$ zu einer neuen Relation (im allgemeinen) höheren Grades

geg.: $\mathbf{r_1}$: (A | ...); $\mathbf{r_2}$: (B | ...)

2 Arten:

- Natural Join (Natürlicher Verbund) : $*$
- Join mit Join-Bedingung: $*_{[\textbf{\textit{Bedingung}}]}$
  Ist im Spezialfall ein Theta-Join (θ-Verbund)

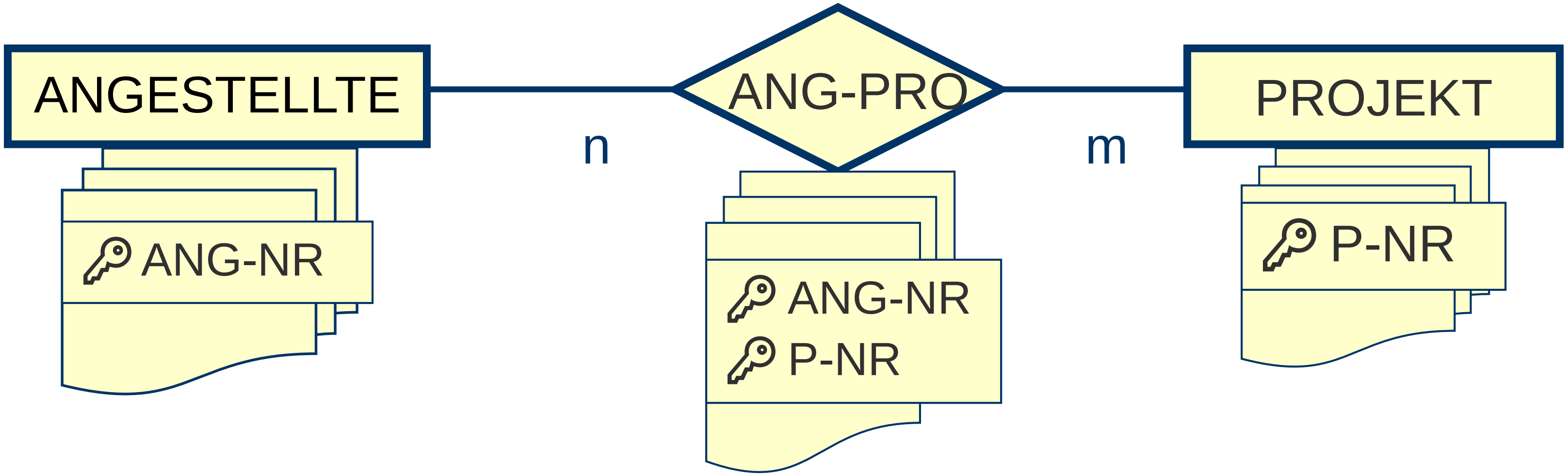
Natural Join:

Verknüpfung von $\mathbf{r_1}$ und $\mathbf{r_2}$ bezüglich
gemeinsamer Attribute $\rightarrow$ „Join-Attribute“; $A \cap B$!

Beispiel 3-9:

Betrachte *angestellte1 – projekt1 – ang-pro1*

Gesucht: Name und Proz-Arbzeit aller Angestellten, die an Projekt 770231 mitarbeiten.



3.2.2 Relationenspezifische Operationen

Vorgehen:

(1) angestellte1 $*$ ang-pro1 → Zwischenergebnis1

(2) $\sigma_{[\text{P-NR}=770231]}$ Zwischenergebnis1 → Zwischenergebnis2

(3) $\pi_{[\text{NAME, PROZ-ARBZEIT}]}$Zwischenergebnis2 → Endergebnis

(1) Wert nach Join: *angestellte1* ∗ *ang-pro1* → Zwischenergebnis1

ANG-NR	NAME	WOHNORT	ABT-NR	P-NR	PROZARBEIT
3115	Meyer	Karlsruhe	35	761235	50
3115	Meyer	Karlsruhe	35	770231	50
3207	Müller	Mannheim	30	761235	100
2814	Klein	Mannheim	32	770008	70
2814	Klein	Mannheim	32	770114	30
3190	Maus	Karlsruhe	30	761235	50
3190	Maus	Karlsruhe	30	770231	50
2314	Groß	Karlsruhe	35	770231	100
1324	Schmitt	Heidelberg	35	770231	100
1435	Mann	Bruchsal	32	761235	40
1435	Mann	Bruchsal	32	770114	60
2412	Müller	Karlsruhe	32	770114	100
2454	Schuster	Worms	31	770114	60
2454	Schuster	Worms	31	770008	40

(2) Wert nach Selection der P-NR:

$\sigma_{[\text{P-NR} = 770231]}$ **Zwischenergebnis1** → **Zwischenergebnis2**

ANG-NR	NAME	WONOHRT	ABT_NR	P_NR	PROZARBEIT
3115	Meyer	Karlsruhe	35	770231	50
3190	Maus	Karlsruhe	30	770231	50
2314	Groß	Karlsruhe	35	770231	100
1324	Schmitt	Heidelberg	31	770231	100

(3) Wert nach Projektion:

$\pi_{[\text{NAME, PROZ-ARBZEIT}]}$ Zwischenergebnis2 → **Endergebnis**

NAME	PROZARBEIT
Meyer	50
Maus	50
Groß	100
Schmitt	100

Vikar

Definition: natural join (natürlicher Verbund)

Geg.: zwei Relationen r_1: (A) und r_2: (B)

a) prozedural:

```
1 for each x ∈ r1 do
2    for each y ∈ r2 do
3       if „x und y passen“        // d.h. x.(A ∩ B) = y.(A ∩ B)
        then „übernehme x ∗ y in Ergebnistabelle“
                                   // wobei x ∗ y ::= z ∈ dom(A ∪ B)
                                   // mit z.A=x, z.B=y
        endif
     endfor y
  endfor x
```

3.2.2 *Relationenspezifische Operationen*

Definition: natural join (natürlicher Verbund)

Geg.: zwei Relationen r_1: (A) und r_2: (B)

b) deskriptiv:

$$r_1 * r_2 ::= \{z \in dom(A \cup B) \mid z.A \in r_1, z.B \in r_2\}$$

Folgerung: $r_1 * r_2 = r_1 \times r_2$, falls $A \cap B = \emptyset$
$r_1 * r_2 = r_1 \cap r_2$, falls $A = B$

Eigenschaften des natural join:

- **kommutativ**:
  $r_1 * r_2 = r_2 * r_1$
  (bis auf Reihenfolge der Attribute)
- **assoziativ**:
  $(r_1 * r_2) * r_3 = r_1 * (r_2 * r_3)$

3.2.2 *Relationenspezifische Operationen*

Join mit Join-Bedingung

Verbindung von Relationen

- bezüglich bel. Attribute
- bei beliebigen Vergleichsmöglichkeiten

Schreibweise: $\mathbf{r_1} *_{[\textit{Join-Bedingung}]} r_2$

Definition:

$$r_1 *_{[\textit{Bedingung}]} r_2 = \sigma_{[\textit{Bedingung}]}(r_1 \times r_2)$$

einfachster Fall: $a \in A, b \in B$:

$r_1 *_{[a > b]} r_2$ „θ-Join"

„θ"-Vergleichsoperator

3.2.2 Relationenspezifische Operationen

Allgemein:

Join-Bedingung - wie Selektionsbedingung

Attributnamen:

falls nicht eindeutig (z.B. NAME in A und B)
dann eindeutig machen durch qualifizieren:
r_1.NAME, r_2.NAME (=qualifizierter Name).
Vgl. kartesisches Produkt 3.2.1

Beispiel 3-10:

angestellte1 $*_{[\text{WOHNORT=P-FILIALE} \wedge \text{P-NR=761235}]}$ projekt1

ANG-NR	NAME	**WOHN-ORT**	ABT-NR	P-NAME	**P-NR**	**P-FILIALE**	P-LEITER
3115	Meyer	Karlsruhe	35	P-1	**761235**	Karlsruhe	3115
3190	Maus	Karlsruhe	30	P-1	**761235**	Karlsruhe	3115
2314	Groß	Karlsruhe	35	P-1	**761235**	Karlsruhe	3115
2412	Müller	Karlsruhe	32	P-1	**761235**	Karlsruhe	3115

Weitere Join-Varianten

geg.: r_1: (U), r_2: (V)

- **Equi-Join**:
  - $r_1 *_{[a = b]} r_2,$
  - $r_1 *_{[a1 = b1 \wedge a3 = b3 \wedge a7 = b4]} r_2$
  - Der Equi-Join ist ein Spezialfall des Join mit Join-Bedingung
- **Semi-Join**
- **Left-Outer-Join**
- **Right-Outer-Join**
- **Full-Outer-Join**

3.2.3 *Ausdrücke der Relationenalgebra:* (1|2)

1. relAusdruck ::= term | term operator term
2. term ::= relName | projektion | selektion | (relAusdruck)
3. projektion ::= π[attributliste] term
4. selektion ::= σ[selBedingung] term
5. attributliste ::= attributname | attributname, attributliste
6. operator ::= mengenOperator | joinOperator
7. mengenOperator ::= $\cap$ | $\cup$ | \ | $\times$ |
8. joinOperator ::= * | * [joinBedingung] | (weitere join-Varianten)
9. selBedingung ::= < s.o. – 3.2.2.>
10. joinBedingung ::= selBedingung
11. relName ::= name
12. attributname ::= name

3.2.3 *Ausdrücke der Relationenalgebra:* (2|2)

Beispiel 3-11:

„Bestimme Namen und prozentuale Arbeitszeit aller Angestellten, die an Projekt Nr. 770231 mitarbeiten.“

$$\pi_{[\text{NAME, PROZ-ARBZEIT}]}\ (\sigma_{[\text{P-NR} = 770231]}\ (\text{angestellte1} * \text{ang-pro1}))$$

ANG-NR	NAME	WONOHRT	ABT_NR	P_NR	PROZARBEIT
3115	Meyer	Karlsruhe	35	770231	50
3190	Maus	Karlsruhe	30	770231	50
2314	Groß			770231	100
1324	Schmitt			770231	100

NAME	PROZARBEIT
Meyer	50
Maus	50
Groß	100
Schmitt	100

bzw: $\pi[\text{NAME, PROZ-ARBZEIT}]\ (\text{angestellte1} * \sigma[\text{P-NR} = 770231]\text{ang-pro1})$

3 Das relationale Datenmodel

3.4 Übertragung ER-Modell in relationales Datenmodell (1|1)

Problem:

Ein Datenschema, das in Form eines ER-Diagramms vorliegt, soll in ein relationales Datenschema übertragen werden.

- ER-Datenschema ⇒ relationales Datenschema
- Im ER-Modell:
  zwei verschiedene Modellierungskonstrukte:
  Entities und *Beziehungen*
- Im relationalen Modell
  nur ein Konstrukt: *Relation*

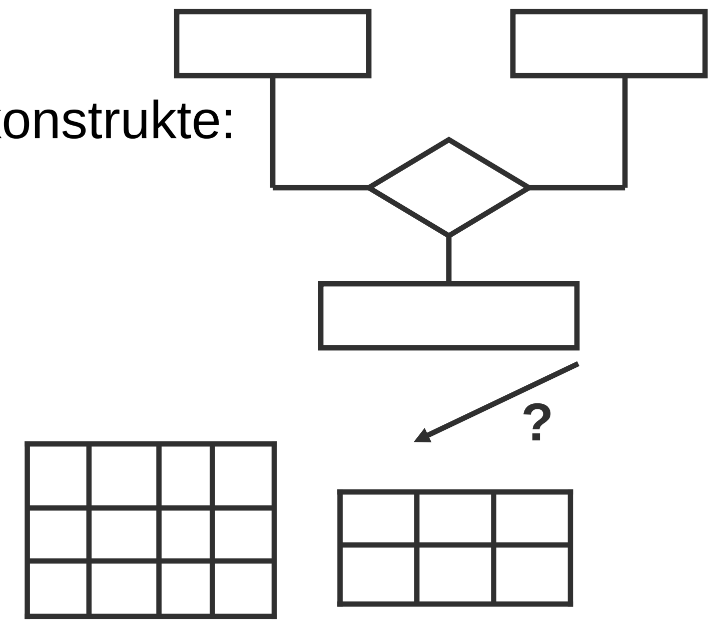


Abbildung notwendig:

Entity-Typen, Beziehungstypen → Relationsschemata, Fremdschlüssel

3.4.1 Übertragung von Entity-Typen

- <u>Entity-Typ</u> **E** :<A> mit
  - Primärschlüssel P $\subseteq$ A

  $\rightarrow$ Relationsschema $\mathbf{r}_E$: R_E mit

  $\mathbf{R}_E$ = (A | Primärschlüssel P)

  (Entity-Typ ist weder Weak Entity-Typ noch Unterentity-Typ!)

Vereinbarung für Schreibweise:

Es seien: A,B Attributmengen, a Attribut

Dann schreiben wir:

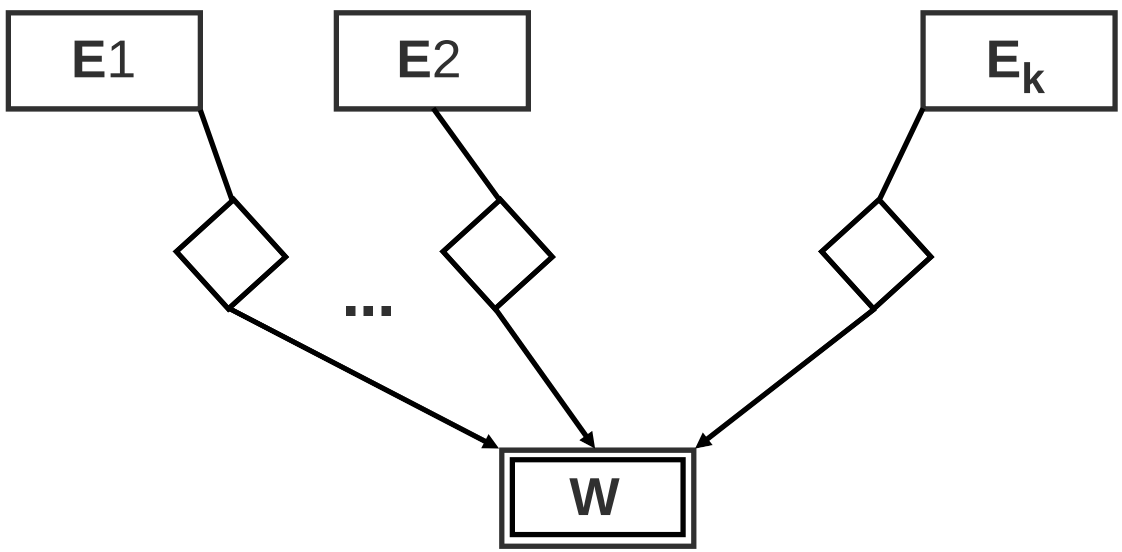
AB für A$\cup$B

aB für {a} $\cup$B

3.4.1 Übertragung von Entity-Typen

- <u>Weak Entity-Typ</u> **W:**<A>, mit
  - Schlüsselattributen $S \subseteq A$
  - Identifikationsabhängigkeit von Entity-Typen **E_1**:<A_1>, ..., **E_k**:<A_k> mit den jeweiligen

    Primärschlüsseln P_1,..., P_k



3.4.1 Übertragung von Entity-Typen

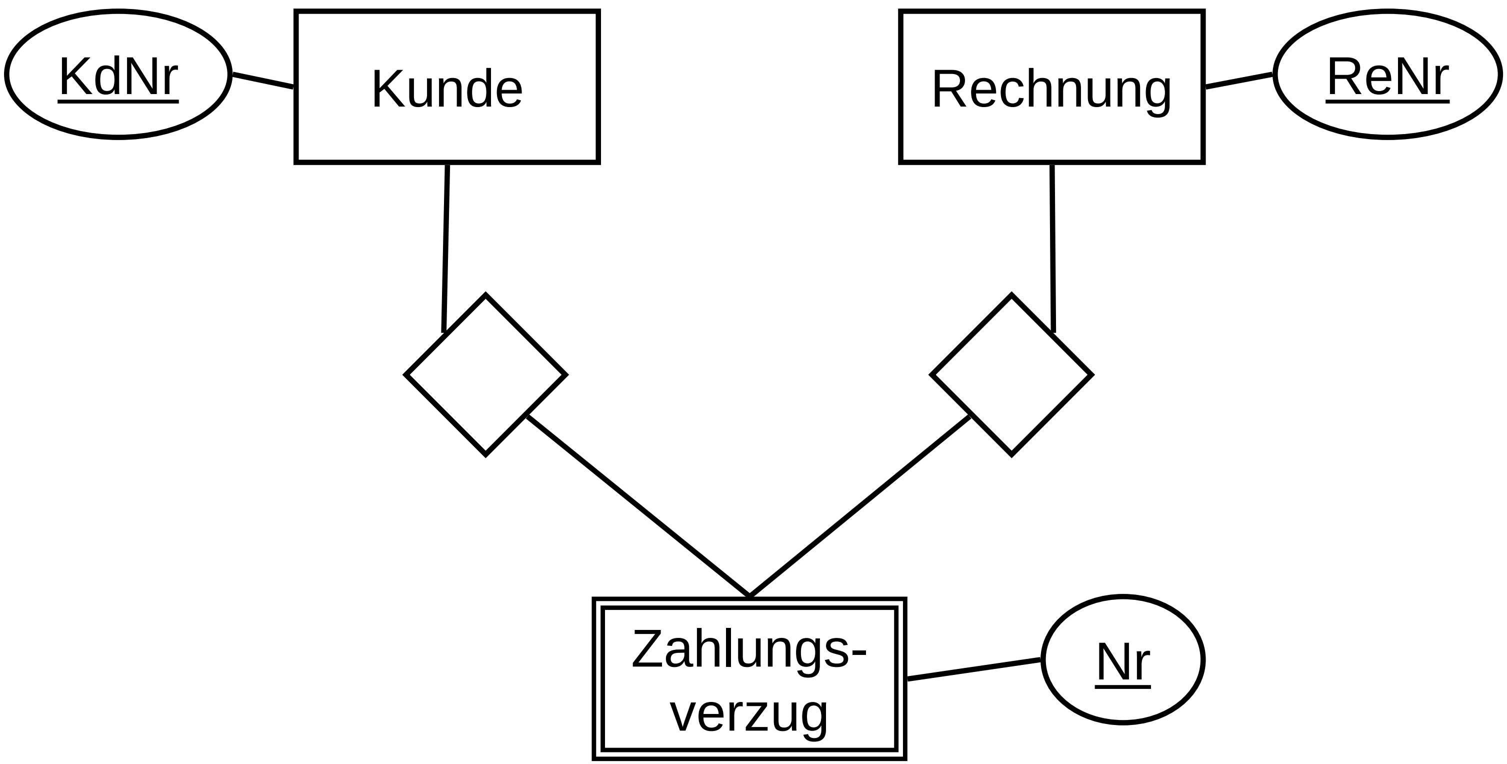
W $\rightarrow$
Relationsschema $\mathbf{r}_W$: R_W mit
$\mathbf{R}_W = (A \{P_1 \ldots P_k\} \mid \Sigma_W)$,

- $\Sigma_W =$
  S $\{P_1 \ldots P_k\}$ ist der Primärschlüssel

- Fremdschlüssel:
  $\mathbf{r}_W.P_1 \subseteq \mathbf{r}_1.P_1$,
  … ,
  $\mathbf{r}_W.P_k \subseteq \mathbf{r}_1.P_k$

3.4.1 Übertragung von Entity-Typen

Beispiel:



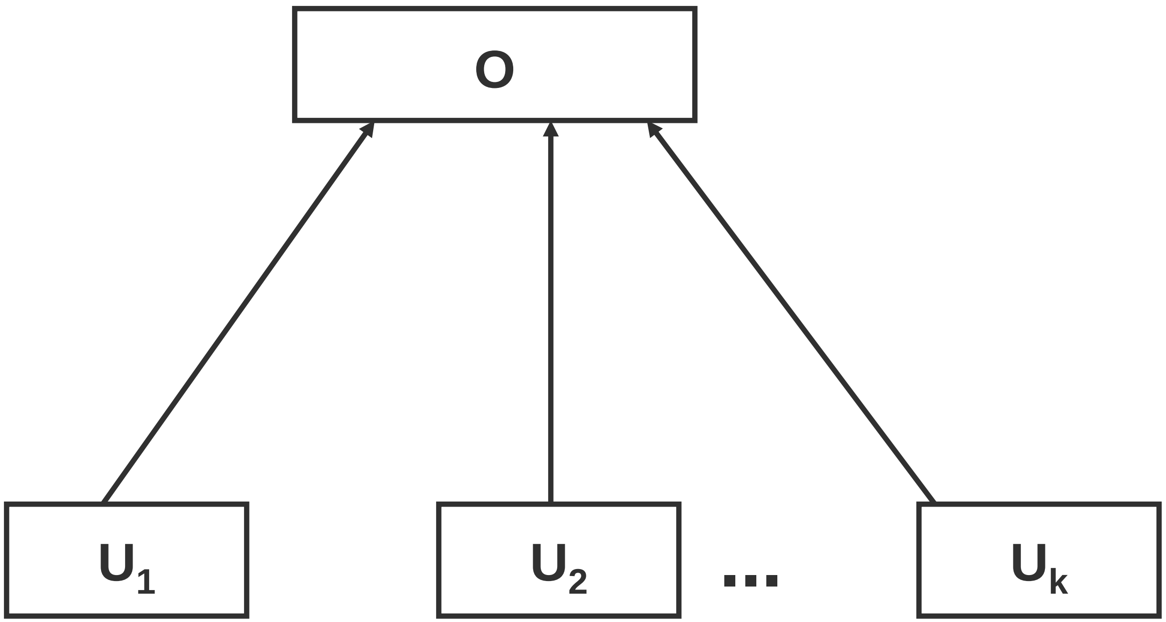
Zahlungsverzug → zverzug: (KdNr, ReNr, Nr, ...),
zverzug.{KdNr} ⊆ kunde.{KdNr}, zverzug.{ReNr} ⊆ rechnung.{ReNr}

3.4.2 Übertragung von Generalisierungen

(nur Einfachgeneralisierungen)

Unterentity-Typen: **U_1:**<A_1>, **U_2:**<A_2>, ..., **U_k:**<A_k>

Oberentity- Typ: **O:**<A> mit Primärschlüssel $P \subseteq A$



Vier Alternativen:

1. **Oberentity**-Typ **O**
   $\rightarrow$ Rel.schema $\mathbf{r}_O$:R_O mit
   $\mathbf{R}_O$ = (A | Primärschlüssel P) (wie zuvor).

   Jeder **Unterentity**-Typ $\mathbf{U}_i$
   $\rightarrow$ Rel.schema $\mathbf{r}_i$:R_i mit
   $\mathbf{R}_i$ = (P A_i | Primärschlüssel P)

   – Fremdschlüssel: $\mathbf{r}_i$.P $\subseteq$ $\mathbf{r}_O$.P

3.4.2 Übertragung von Generalisierungen

2. Für den **Oberentity**-Typ wird kein eigenes Schema erstellt:
   Jeder **Unterentity**-Typ $\mathbf{U}_i$
   $\rightarrow$ Rel.Schema $\mathbf{r}_i$:R_i mit
   $\mathbf{R}_i$ = (A A_i | Primärschlüssel P)

3. Für **O**, $\mathbf{U_1}$, $\mathbf{U_2}$, ..., $\mathbf{U_k}$
   wird ein einziges Rel.Schema **r: R** erstellt mit

   $\mathbf{R}$ = (A A_1 ... A_k {t} | Primärschlüssel P)

   – **t** ist ein zusätzliches Attribut, zur Angabe des **Untertyps**

   ! Nullwerte notwendig

4. Für **O**, $\mathbf{U_1}$, $\mathbf{U_2}$, ..., $\mathbf{U_k}$
   wird ein einziges Relationsschema **r: R** erstellt mit
   R = (A A_1 ... A_k {t_1, t_2, ...,t_k} | Primärschlüssel P)

   – $\mathbf{t}_1$, $\mathbf{t}_2$, ..., $\mathbf{t}_k$ sind **zusätzliche Attribute** vom Typ Boolean zur Angabe ob ein Entity zu einem bestimmten **Untertyp** gehört (notwendig wenn **U1**, **U2**, ..., **Uk** nicht disjunkt sind)

   ! Nullwerte notwendig

Anmerkung:

Alternative (1) ist immer verwendbar und sollte,
sofern keine sonstigen Gründe dagegen sprechen,
nach Möglichkeit verwendet werden!

3.4.3 Übertragung von Beziehungstypen

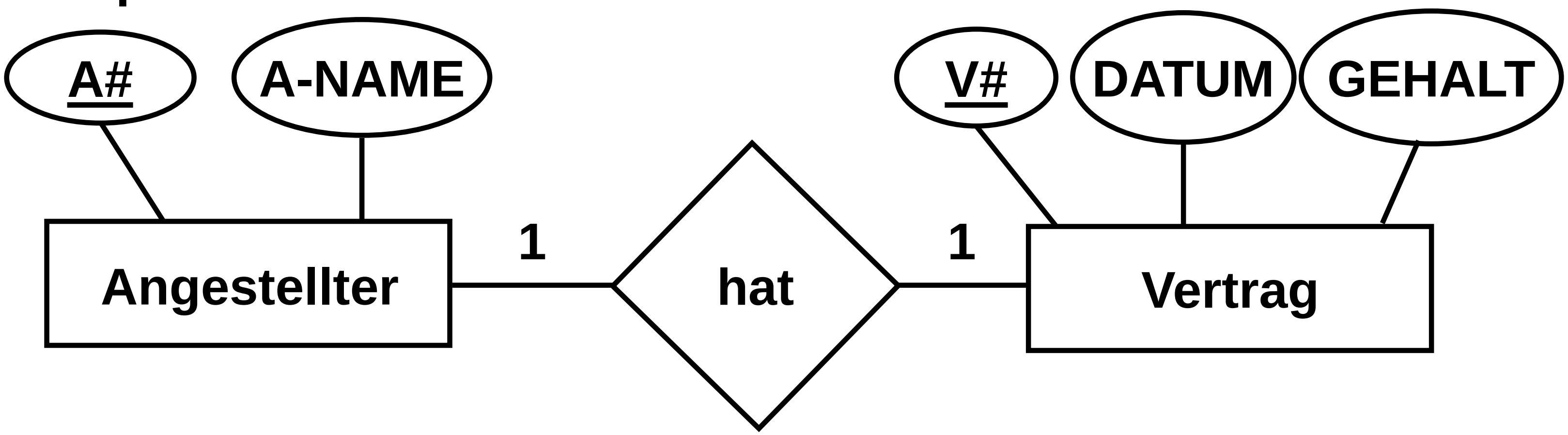
1:1 - Beziehungen

- Ein 1:1-Relationship-Typ wird i.allg. nicht zu einer eigenen Relation.
- Information wird an eine der beiden (den betroffenen Entity-Typen entsprechenden) Relationen "angehängt".

3.4.3 Übertragung von Beziehungstypen

1:1 - Beziehungen

Beispiel:



Angestellter	A#	A-NAME	V#

Vertrag	V#	DATUM	GEHALT	A#

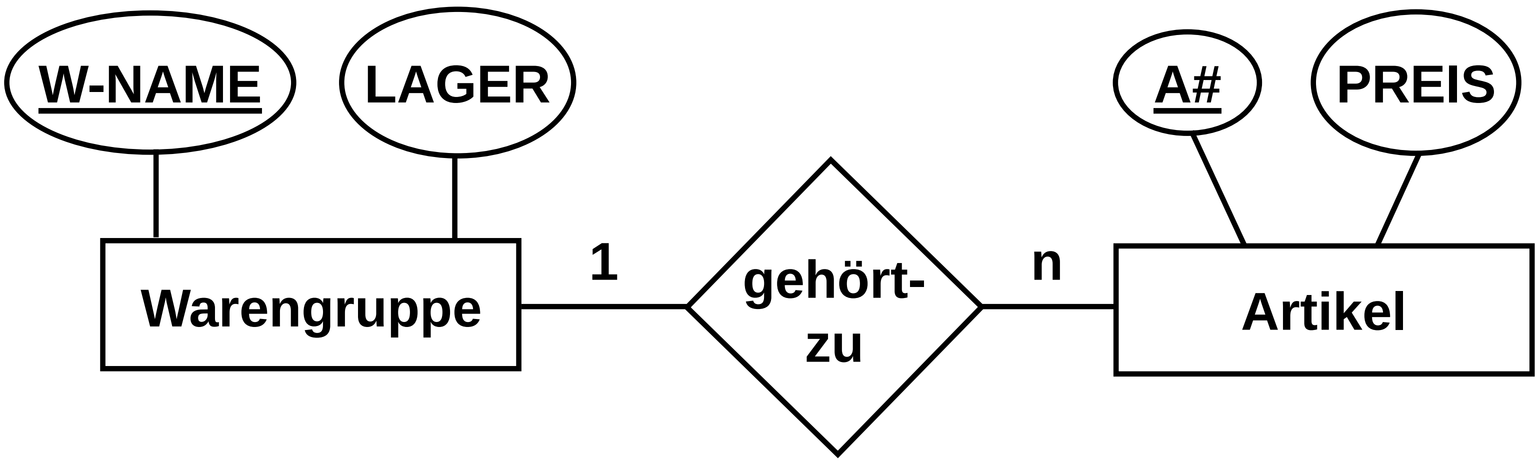
1:n - Beziehungen

- Ein 1:n-Relationship-Typ wird i.a. nicht zu einer eigenen Relation.
- Information wird an die Relation "angehängt", die dem Entity-Typ an der mit **n** beschrifteten Kante entspricht.

3.4.3 Übertragung von Beziehungstypen

1:n – Beziehungen

Beispiel: „Artikel gehört zu Warengruppe“



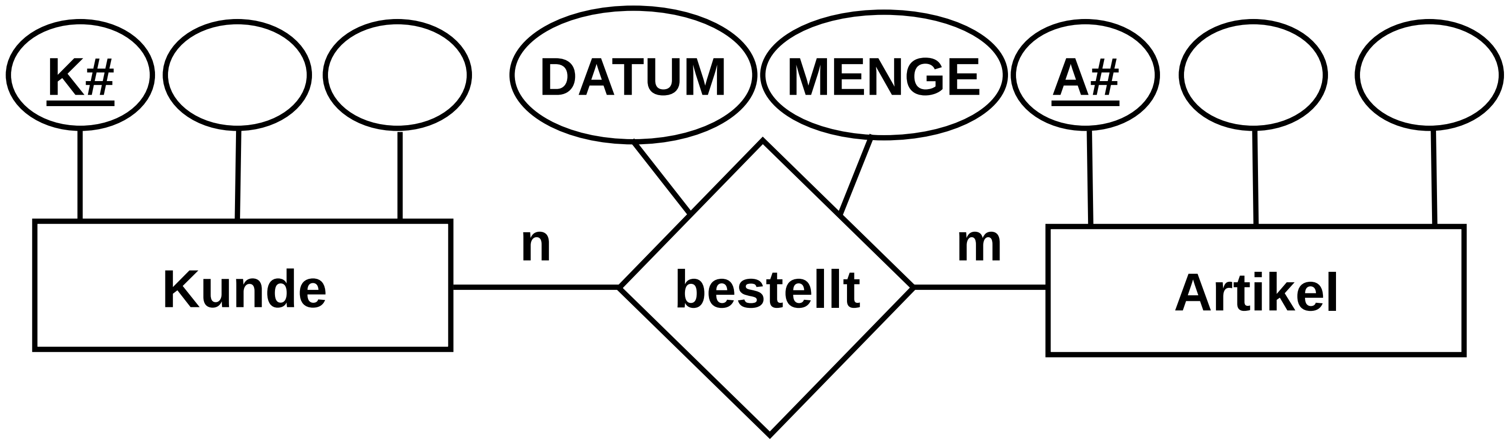
Artikel	A#	Preis	W-NAME

n:m - Beziehungen

- Aus **n:m**-Beziehungstyp wird eine zusätzliche Relation.
- Relation enthält die Schlüssel der beteiligten Entity-Typen als Attribute und zusätzlich die Attribute des Beziehungstyps.

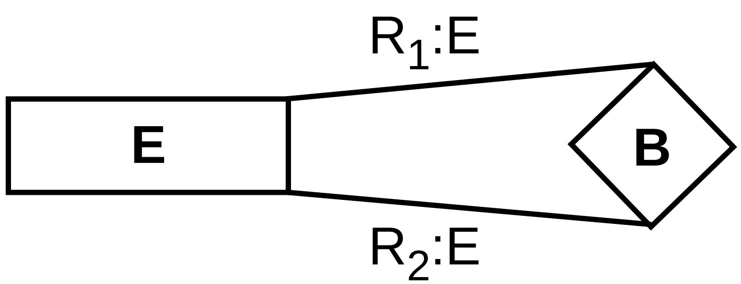
n:m - Beziehungen

Beispiel:



bestellt	K#	A#	DATUM	MENGE

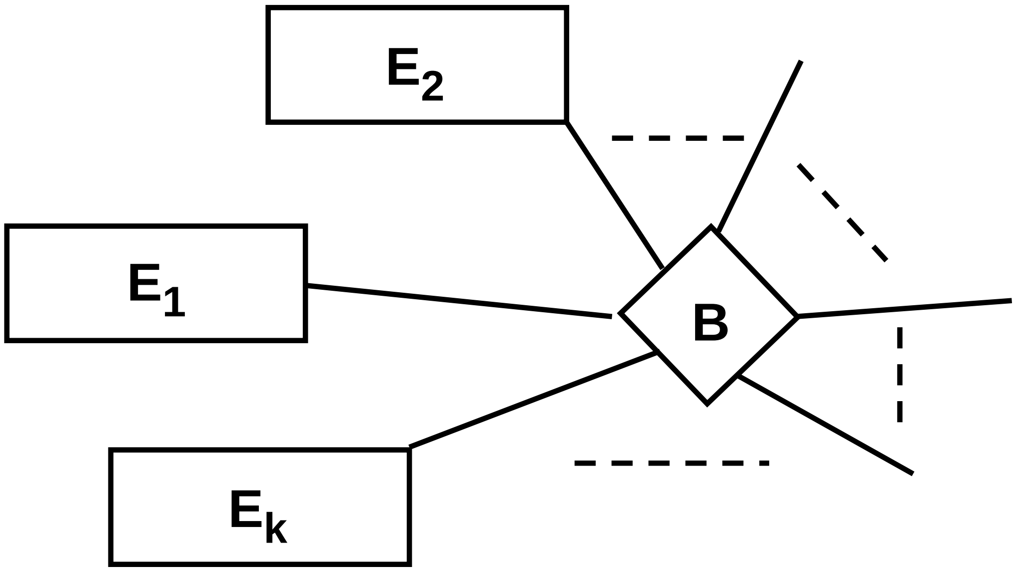
Rekursive Beziehungen (Selbstbezüge)



analog zu normalen 1:1 -, 1:n - und n:m - Beziehungen

Es wird empfohlen, die Rollennamen (vgl. 2.1.6) in die Namen der entsprechenden Attribute aufzunehmen.

Beziehungen vom Grad > 2



B wird eigenes Relationsschema $\mathbf{r}_B$: R_B:

B: <E_1, ..., E_k / A_z>,
Relationen für die Entity-Typen:
$\mathbf{r}_1$: R_1 mit $\mathbf{R}_1 = (A_1 \mid \Sigma_1)$, ($P_1$ ist Pr.schlüssel) $\in \Sigma_1$, ...,
$\mathbf{r}_k$: R_k mit $\mathbf{R}_k = (A_k \mid \Sigma_k)$, ($P_k$ ist Pr.schlüssel) $\in \Sigma_k$
$\rightarrow$

Beziehungen vom Grad > 2

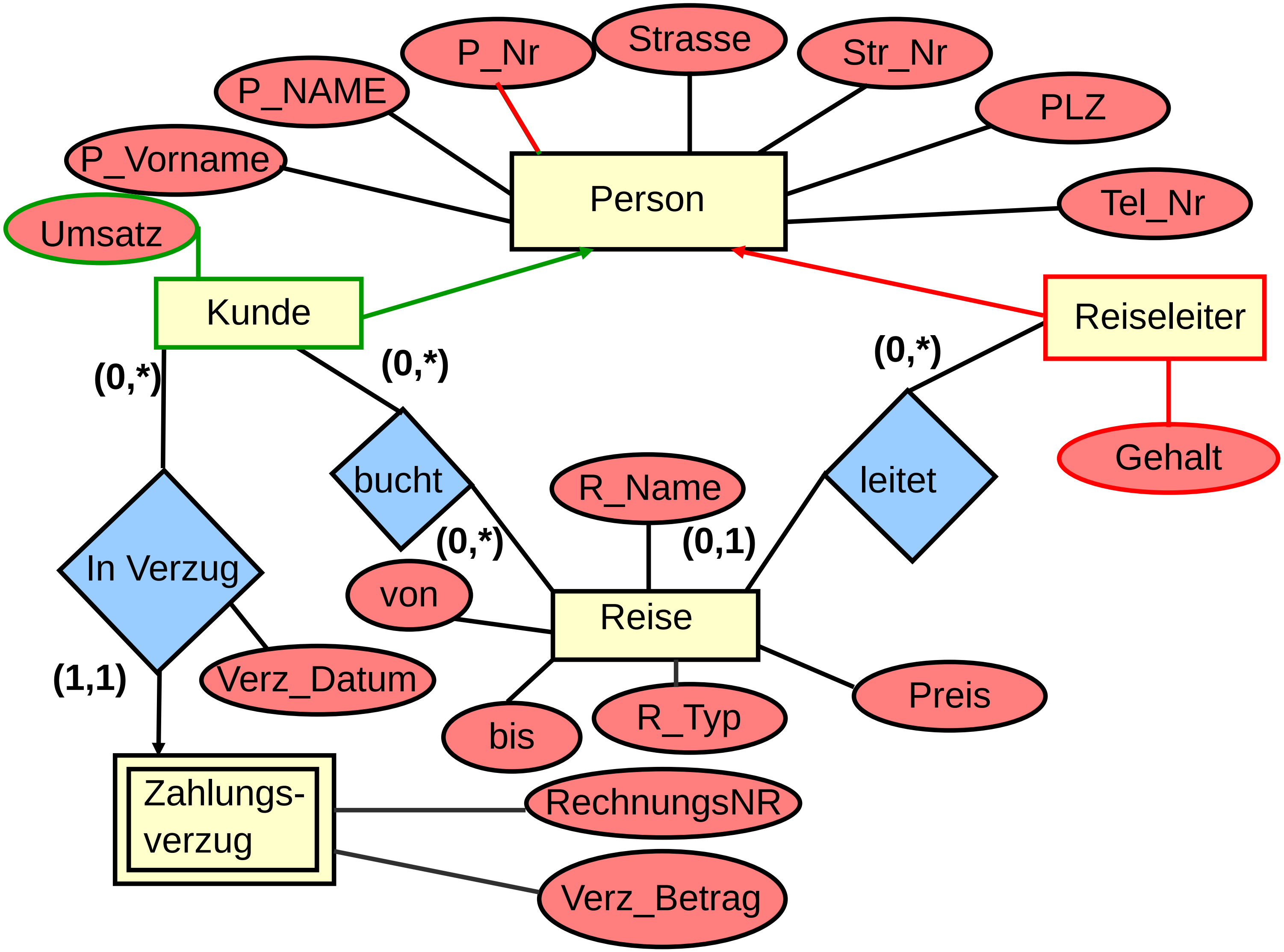
Relation für die Beziehung:

$r_B : R_B$,

- $r_B{:}R_B$ mit $\mathbf{R_B} = (P_1 \ldots P_k\ A_z \mid \Sigma_B)$,
- Fremdschlüssel:
  $\mathbf{r}_B.P_1 \subseteq \mathbf{r}_1.P_1, \ldots , \mathbf{r}_B.P_k \subseteq \mathbf{r}_k.P_k$
- Falls 1 die Mindestkardinalität von $\mathbf{E}_j$ bzgl. **B** ist zusätzlich $\mathbf{r}_j.P_j \subseteq \mathbf{r}_B.P_j$

$\Sigma_{\mathbf{B}}$: Schlüssel ergibt sich aus der Vereinigung der Primärschlüssel P_j der Relationsschemata, für welche die entsprechenden Entity-Typen (gemeinsam) einen Schlüssel von **B** darstellen.

Beispiel 3-15: Datenmodell eines Reiseveranstalters



Normale Entity-Typen:

- **Person** → person: (P_NR, P_Vname, P_Nname, Strasse, Str_Nr, PLZ, Tel_Nr)
- **Reise** → reise: (R_Name, von, bis, Preis, R_Typ)

Weak Entity-Typen und Unterentity-Typen:

- **Kunde** → kunde: (P_Nr, Umsatz),
  kunde.{P_Nr} ⊆ person.{P_Nr}
- **Reiseleiter** → rleiter: (P_Nr, Gehalt),
  rleiter.{P_Nr} ⊆ person.{P_Nr}
- **Zahlungsverzug** → zverzug: (P_Nr, RechnungsNr, Verz_Datum, Verz_Betrag),
  zverzug.{P_Nr} ⊆ kunde.{P_Nr}

NULL-Werte sind nicht erlaubt.

3.4.3 Übertragung von Beziehungstypen

1:n- Beziehungen

- „in Verzug" wurde schon durch den **Weak Entity-Typ** zverzug realisiert

- „leitet": Ergänzen von **Reise** um die Personennummer des Reiseleiters
  → reise: (R_Name, von, bis, Preis, R_Typ, P_Nr)
  reise.{P_Nr} ⊆ rleiter.{P_Nr},

NULL-Werte sind erlaubt.

m:n-Beziehungen

- „bucht" → buchung: (P_Nr, R_Name, von, bis),
  buchung.{P_Nr} ⊆ kunde.{P_Nr}
  buchung.{R_Name, von, bis} ⊆ reise.{R_Name, von, bis}