

5 Das relationale Datenmodell

→ Codd 1970

5.1 Grundlagen	2 ▶
5.2 Relationale Algebra	33 ▶
5.3 Anmerkungen zu relationalen Sprachen.....	73 ▶
5.4 Normalformen	84 ▶

5 Das relationale Datenmodell

5.1 Grundlagen	2
5.1.1 Einige Definitionen	3
5.1.2 Attribute, Domänen, Tupel und Relationen	5
5.1.3 Schlüssel einer Relation	15
5.1.4 NULL – Werte	18
5.1.5 Die erste Normalform (1 NF)	20
5.1.6 Modellierung der Realwelt im relationalen Datenmodell	25
5.1.7 Fremdschlüssel	30
5.2 Relationale Algebra	33
5.3 Anmerkungen zu relationalen Sprachen.....	73
5.4 Normalformen	84



5.1.1 Einige Definitionen

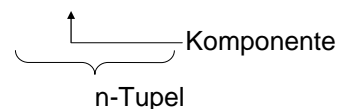
(1|2)

Kartesisches Produkt:

W_1, W_2, \dots, W_n beliebige Mengen.

$W_1 \times W_2 \times \dots \times W_n$

$::= \{(w_1, w_2, \dots, w_n) \mid w_i \in W_i (i = 1, 2, \dots, n)\}$.



Beispiel:

$W_1 = \{1, 2, 3\}, W_2 = \{a, b, c\}$

$W_1 \times W_2 = \{(1, a), (1, b), (1, c),$
 $(2, a), (2, b), (2, c),$
 $(3, a), (3, b), (3, c)\}$

(weitere Details später)

5.1.1 Einige Definitionen

(2|2)

Relation (im mathematischen 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.

Bsp.: *Telefonbuch* \subseteq *string* \times *string* \times *integer*

„Tupel“ der Relation X

$x \in X: x = (x_1, x_2, \dots, x_n),$
 $x_i \in W_i (i = 1, 2, \dots, n)$

Bsp.: *eintr* \in *Telefonbuch*: *eintr* = (James, Bond, 1234007)

5.1.2 Attribute, Domänen, Tupel und Relationen

(1|10)

Relation r im DB-Sinn: $r: (A \mid \Sigma)$

$r \subseteq \text{dom}(A)$

A = $\{a_1, a_2, \dots, a_n\}$ Menge von Attributen paarweise verschieden, ansonsten ggf. „Rollennamen“
vergeben: Z.B. NAME: ANG-NAME; ABT-NAME; ...

a_i – **Attribut**
z.B. Angest#

dom(a_i) – Wertebereich (**Domain**)
z.B. $\text{dom}(\text{Angest\#}) = \{1..999\}$

Reihenfolge der a_i : beliebig, aber i.F. fest: a_1, a_2, \dots, a_n

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

Σ - Menge von semantischen (intrarelationalen) **Integritätsbedingungen**

- schränken mögliche Tupel ein

r - Relation über den Wertebereichen der Attribute (**A**) mit gewissen einschränkenden Bedingungen (**Σ**)

5

5.1.2 Attribute, Domänen, Tupel und Relationen

(2|10)

Beispiel:

Attributmenge $A = \{a_1, \dots, a_n\}$

$\text{dom}(a_i) = \{x_i, y_i, z_i\}$

Tupel: (x_1, x_2, \dots, x_n)

	a_1	a_2	...	a_n
x:	x_1	x_2	...	x_n
y:	y_1	y_2	...	y_n
z:	z_1	z_2	...	z_n

Man beachte:

- jede Relation ist eine Menge
- „Zeilen“ (=Tupel) sind paarweise verschieden
- die Reihenfolge der Tupel ist ohne Bedeutung

6

5.1.2 Attribute, Domänen, Tupel und Relationen

(3|10)

Wir unterscheiden i.F. genauer (ähnlich wie in höheren Programmiersprachen) zwischen Relationstyp und Relation:

Relationstyp: $R = (A \mid \Sigma)$ bzw. $R = (a_1, a_2, \dots, a_n \mid \Sigma)$

- definiert durch „name=typ“, mit
- **R** – **Name** des Relationstyps
- **A** – **Format** von R; $A = \{a_1, a_2, \dots, a_n\}$
- **Σ** – Menge von semantischen **Integritätsbedingungen**

Für eine konkrete Relation r vom Typ R schreiben wir

- $r: R$ bzw.
- $r: (A \mid \Sigma)$
- $r: (A \mid \Sigma)$ wird als Schema einer Relation bezeichnet

7

5.1.2 Attribute, Domänen, Tupel und Relationen

(4|10)

Definition:

R = (A | Σ) geg.

$r: R ::= \Leftrightarrow$

(1) **val**(**R**) ::= $\{X \subseteq \text{dom}(A) \mid \Sigma(X)\}$
„value set“; mögliche Werte von R, die Integritätsbed. genügen

(2) **typ**(**r**) ::= R (oder (A | Σ))

(3) **format**(**r**) ::= A
 r ist Relation vom Format A, d.h.: $r \subseteq \text{dom}(A)$

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

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

8

5.1.2 Attribute, Domänen, Tupel und Relationen

(5|10)

Eine **semantische Integritätsbedingung** σ über der Attributmenge A

- macht eine **Aussage** über Teilmengen $X \subseteq \text{dom}(A)$
- $X \subseteq \text{dom}(A)$:
 - entweder X erfüllt σ : $\sigma(X)$
(wir sagen auch „ σ trifft zu / gilt für X “; i.Z. $\sigma(X)$)
 - oder nicht: $\neg\sigma(X)$

Relationenschema:

- $r : (A \mid \Sigma)$
- $r^t \in \text{val}(r)$ mögliche Werte zum Zeitpunkt t
- $\Sigma =$ Menge von semantischen Integritätsbedingungen über A
- in der Bedeutung:
jedes r^t erfüllt alle $\sigma \in \Sigma$
d.h. $\sigma(r)$ für alle $\sigma \in \Sigma$, kurz auch: $\Sigma(r)$

9

5.1.2 Attribute, Domänen, Tupel und Relationen

(6|10)

Anmerkungen:

- Kurzschreibweise für „ $r : (A \mid \emptyset)$ “: „ $r : (A)$ “
- Sei $r: R, \sigma, \sigma'$ spez. Integritätsbedingungen (allg. Aussagen!)
 $\sigma(r) \Leftrightarrow (\forall X \in \text{val}(R): \sigma(X))$
 $\sigma \models \sigma' \Leftrightarrow (\sigma(r) \Rightarrow \sigma'(r))$ („aus σ folgt σ' “)

Beispiel:

- Gegeben:

Relation ang-pro: (AngNr, PrNr, ProzArbZeit)

Es sei für AngNr a :

$s(a)$ = Summe von Prozent-Arbeit des Angestellten a über alle Projekte PNr, an denen a mitarbeitet

- $\sigma_1: s(a) \leq 100$ für alle a
- $\sigma_2: s(a) = 100$ für alle a
- dann gilt: $\sigma_2 \models \sigma_1$

d.h. $\forall X \in \text{val}(\text{ang-pro}): \sigma_2(X) \Rightarrow \sigma_1(X)$

10

5.1.2 Attribute, Domänen, Tupel und Relationen

(7|10)

Wunsch bzw. Vorstellung:

Entity-Typ $E: \langle A \rangle$ wird vollständig modelliert durch $r: (A \mid \Sigma)$

- d.h. wir nehmen an:
Jedes $X \subseteq \text{dom}(A)$ mit $\Sigma(X)$ stellt einen gültigen Realweltzustand (d.h. „ $E^t: X$ “) dar!

Zwei Mengen Σ, Σ' sind äquivalent ($\Sigma \sim \Sigma'$), wenn sie - in obigem Sinn - dieselbe Realwelt modellieren (d.h. dieselben gültigen Realweltzustände $X \subseteq \text{dom}(A)$ ergeben).

11

5.1.2 Attribute, Domänen, Tupel und Relationen

(8|10)

Beispiel:

„Angestellte eines Unternehmens“

- **Relationstyp:**

ANGEST=

(ANG-NR, NAME, WOHNORT, ABT-NR $\mid \Sigma_{ANG}$)

mit (etwa): $\Sigma_{ANG} = \{ \text{„ANG-NR eindeutig“} \}$

- **Relation:**

angestellte: ANGEST

12

5.1.2 Attribute, Domänen, Tupel und Relationen

(9|10)

Darstellung einer Relation:

üblicherweise als zwei-dimensionale **Tabelle**



- Spalten
 - Attribute
 - Identifizierung durch Attributnamen (manchmal auch Spalten-Nr. / z.B. DATALOG)
- Zeilen
 - Tupel der Relation
- Zeile/Spalte
 - Attributwert



13

5.1.2 Attribute, Domänen, Tupel und Relationen

(10|10)

Beispiel:

wg. „Relation = Menge“

Tabelle für angestellte

⇒ Zeilen der Tabelle paarweise verschieden

⇒ Reihenfolge der Zeilen ohne Bedeutung

angestellte			
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



14

5.1.3 Schlüssel einer Relation

(1|3)

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

Gegeben: $r: R$
 $K \subseteq A$

- (1) K **identifizierend** für r (bzw. für R) : \Leftrightarrow
 $\forall X \in \text{val}(R) \quad \forall x, y \in X$ gilt: $(x.K = y.K \Rightarrow x = y)$
(m.a.W.: : Für jeden Zeitpunkt t , $\forall x, y \in r^t$: $(x.K = y.K \Rightarrow x = y)$)
- (2) K heißt **Schlüssel** für r (bzw. R) : \Leftrightarrow
 - a) K identifizierend für r
 - b) K minimal mit dieser Eigenschaft, d.h.
 $\forall K' \subset K \exists X \in \text{val}(R) \exists x, y \in X$:
 $x \neq y$, aber $x.K' = y.K'$
(d.h., es kann kein Attribut aus K weggelassen werden, ohne die Eigenschaft "identifizierend" zu verlieren)

15

5.1.3 Schlüssel einer Relation

(2|3)

- (3) - Eine Relation kann mehrere Schlüssel(kandidaten) besitzen
 - 1 Schlüssel auszeichnen \Rightarrow **Primärschlüssel (PS)**
 - Kennzeichnung des **PS** durch Unterstreichen der zugehörigen Attribute.
- (4) $a \in A$:
 - a heißt „**Schlüsselattribut**“ : $\Leftrightarrow \exists$ Schlüssel K mit $a \in K$
(engl.: *key attribute*, auch: *prime attribute*)
 - a heißt „**Nichtschlüsselattribut (NSA)**“ sonst

Bemerkung:

- Angabe des **PS** ist eine **spezielle semantische Integritätsbedingung**
- Demgegenüber: Schlüssel für Entity-Typ: „Eigenschaft“ !

16

5.1.3 Schlüssel einer Relation

(3|3)

Beispiel:

angestellte: ANGEST

angestellte			
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
9999	Müller	Mannheim	30

{ANG-NR}
Schlüssel,
Primärschlüssel

{ANG-NR, NAME}
identifizierend,
kein Schlüssel

{NAME}
nicht identifizierend

{NAME, WOHNORT}
identifizierend?

Falls keine zusätzliche
Integritätsbedingung:
NEIN!

5.1.4 NULL – Werte

(1|2)

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 für die 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

5.1.4 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!

5.1.5 Die erste Normalform (1 NF)

(1|5)

Definition: 1NF-Relation

$r: (A | \Sigma)$

$a \in A: a$ „atomar“

d.h. $\forall x \in r: x.a = \text{ein Wert } \alpha \in \text{dom}(a)$

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

Potentielles Bsp. für „mengenwertiges“ Attribut:

„FÄHIGKEITEN“: Menge von Einzelfähigkeiten,

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

5.1.5 Die erste Normalform (1 NF)

(2|5)

Beispiel: Typ ANGEST erweitern um FÄHIGKEITEN → nicht 1NF-Form

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

5.1.5 Die erste Normalform (1 NF)

(3|5)

Verschiedene Lösungsmöglichkeiten für „1NF“:

Variante 1: Attribut-Verdopplung

angestellte					
ANG-NR	NAME	WOHNORT	ABT-NR	FÄHIGKEITEN1	FÄHIGKEITEN 2
3115	Meyer	Karlsruhe	35	Englisch	Stenographie
3207	Müller	Mannheim	30	C	Cobol
2814	Klein	Mannheim	32	Englisch	NULL
3190	NULL	NULL
...

⊛ Problem: 4 Fähigkeiten?



5.1.5 Die erste Normalform (1 NF)

(4|5)

Variante 2: Tupel-Verdopplung

angestellte				
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
...

⊛ Probleme: (Speicherplatz) Schlüssel ?
Update-Anomalien



5.1.5 Die erste Normalform (1 NF)

(5|5)

Variante 3: Zerlegung in 2 Relationen, PS in angestellte_2 übernehmen

angestellte			
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 für Fähigkeit NULL

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

Beispiel:

Objekte: Angestellte, Projekte

Beziehungen: Angestellte arbeiten an einem Projekt mit.

⇒ 3 Relationstypen

- **ANGEST** = (ANG-NR, NAME, WOHNORT, ABT-NR | S)
S = {"ANG-NR ist PS"}
- **PROJ** = (P-NAME, P-NR, P-FILIALE, P-LEITER | {"P-NR ist PS"})
- **ANGPRO** = (P-NR, ANG-NR, PROZ-ARBZEIT | {"P-NR, ANG-NR ist PS", ... })

- angestellte: ANGEST

- projekt: PROJEKT

- ang-pro: ANGPRO

alternativ: PS Attribute unterstreichen

25

5.1.6 Modellierung der Realwelt im relationalen Datenmodell (2|5)

PS: ANG-NR

angestellte			
<u>ANG-NR</u>	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



26

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

PS: P-NR

projekt			
P-NAME	<u>P-NR</u>	P-FILIALE	P-LEITER
P-1	761235	Karlsruhe	3115
P-2	770008	Karlsruhe	3115
P-3	770114	Heidelberg	1324
P-4	770231	Mannheim	2814

27

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

ang-pro		
<u>P-NR</u>	<u>ANG-NR</u>	PROZ-ARBZEIT
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

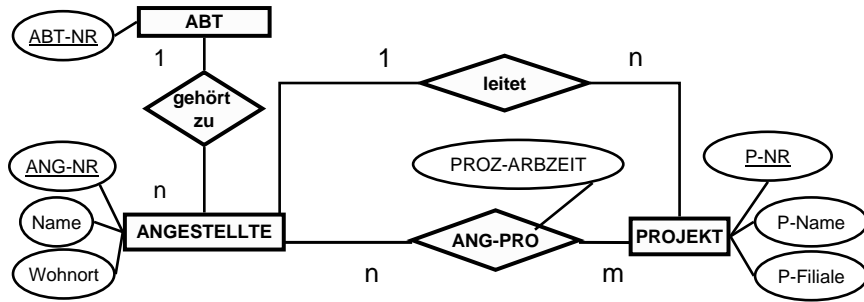
PS: { P-NR, ANG-NR }



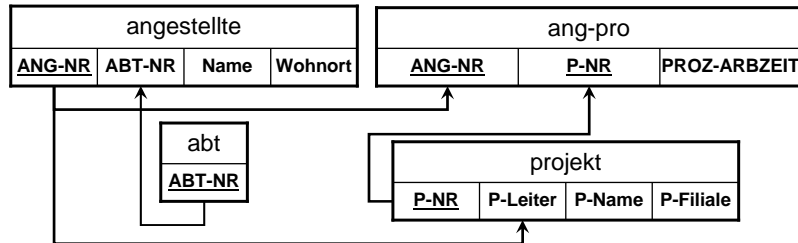
28



5.1.6 Modellierung der Realwelt im relationalen Datenmodell (5|5)



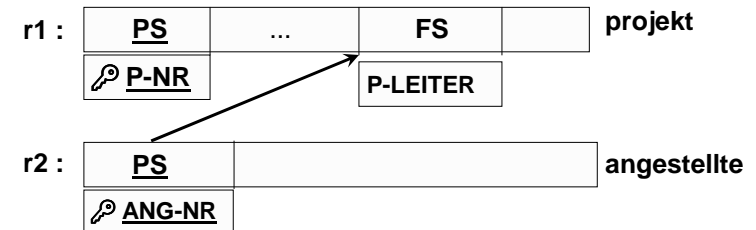
Relationen:



29

5.1.7 Fremdschlüssel (1|3)

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



Fremdschlüsselattribute im vorigen Beispiel:

- P-NR, ANG-NR in ang-pro
- P-LEITER in projekt
- ABT-NR in angestellte



30

5.1.7 Fremdschlüssel (2|3)

Man sagt auch:

die Attribute FS referenzieren r_2

$$r_1.FS \subseteq r_2.PS$$

- ist auch Integritätsbedingung (auch „referentielle Integrität“ genannt)
- Interrelationale Integrität: Integrität zwischen Relationen (Im Gegensatz zu intrarelationaler Integrität, wie z.B. PS innerhalb einer Relation)

Es gilt:

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

31

5.1.7 Fremdschlüssel (3|3)

Bemerkungen:

- Anwendungsprogrammierer muss Zusammenhänge kennen!
 - d.h. **Datenbankschema** bestehend aus Relationsschemata + intrarelationalen Integritätsbedingungen und ggf. interrelationalen Integritätsbedingungen
- Spezifikation von Fremdschlüsseln über Integritätsbedingungen.
- Namensgleichheit zwischen FS und PS nicht unbedingt notwendig.
- 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 $r_2: R_2, \dots, r_n: R_n$ referenzieren.

32

5 Das relationale Datenmodell

5.1 Grundlagen	2
5.2 Relationale Algebra	33
5.2.1 Mengenoperationen	34
5.2.2 Relationenspezifische Operationen	40
5.2.3 Ausdrücke der Relationalen Algebra	71
5.3 Anmerkungen zu relationalen Sprachen.....	73
5.4 Normalformen	84



5.2 Relationale Algebra

(1|1)

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

Relationenalgebra:

- Relationen
- Operationen auf Relationen
 - Mengenoperationen**
 - ∪ Vereinigung
 - ∩ Mengendurchschnitt
 - Mengendifferenz
 - x Kartesisches Produkt (Kreuzprodukt)
 - relationenspezifische Operationen**
 - π Projektion
 - σ Selektion
 - ⋈ oder * Join (Verbund)
 - ...

5.2.1 Mengenoperationen

(1|5)

r1: (A | ...); r2: (B | ...);

Voraussetzung: r1, r2 „kompatibel“
d.h. format(r1) = format(r2)
(d.h. A = B)

1) Vereinigung: $r1 \cup r2 ::= \{x \mid x \in r1 \vee x \in r2\}$

2) Durchschnitt: $r1 \cap r2 ::= \{x \mid x \in r1 \wedge x \in r2\}$

3) Differenz: $r1 \setminus r2 ::= \{x \in r1 \mid x \notin r2\}$

5.2.1 Mengenoperationen

(2|5)

Beispiel:

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''

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'

5.2.1 Mengenoperationen

(3|5)

4) Kartesisches Produkt:

$$r_1 \times r_2 ::= \{x \in \text{dom}(A \cup B) \mid x.A \in r_1 \wedge x.B \in r_2\}$$

Voraussetzung: $A \cap B = \emptyset$

- Verknüpft alle Tupel aus zwei verschiedenen Relationen und erzeugt neue, zusammengesetzte Tupel
- Attribute, die in r_1, r_2 gleich bezeichnet sind, müssen vor der Bildung des kartesischen Produktes umbenannt werden

37

5.2.1 Mengenoperationen

(4|5)

4) Kartesisches Produkt:

$$r_1 \times r_2 ::= \{x \in \text{dom}(A \cup B) \mid x.A \in r_1 \wedge x.B \in r_2\}$$

Voraussetzung: $A \cap B = \emptyset$

Beispiel:

r1	a	b	c
	α	β	γ
	α	β	γ'
	α'	β'	γ'

r2	d	e
	δ	ϵ
	δ'	ϵ

$r_1 \times r_2$	a	b	c	d	e
	α	β	γ	δ	ϵ
	α	β	γ	δ'	ϵ

38

5.2.1 Mengenoperationen

(5|5)

4) Kartesisches Produkt:

bei $A \cap B \neq \emptyset$ müssen Attribute qualifiziert oder umbenannt werden.

Beispiel:

r1	a	b	c
	α	β	γ
	α	β	γ'
	α'	β'	γ'

r3	a	b
	δ	ϵ
	δ'	ϵ

$r_1 \times r_3$	r1.a	r1.b	r1.c	r3.a	r3.b
	α	β	γ	δ	ϵ
	α	β	γ	δ'	ϵ

39

5.2.2 Relationenspezifische Operationen

(1|30)

A) Projektion: Auswahl von Spalten

geg: $r: (A \mid \dots)$

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

Tupel x eingeschränkt auf L

Weitere Schreibweise: $r.L$ für $\pi_{[L]}r$

Beispiel: angestellte

ANG-NR	NAME	WOHNORT	ABT-NR
3115	Meyer	Karlsruhe	35
3207	Müller	Mannheim	30
2814	Klein	Mannheim	32
3190	Maus	Karlsruhe	30
2412	Müller	Karlsruhe	32
...

40

5.2.2 Relationenspezifische Operationen

(2|30)

A) Projektion: Auswahl von Spalten

geg. $r: (A \mid \dots)$

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

Tupel x eingeschränkt auf L

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

ANG-NR	NAME
3115	Meyer
3207	Müller
2814	Klein
3190	Maus
2412	Müller
...	...

$\pi_{[NAME]} angestellte$

NAME
Meyer
Müller
Klein
Maus
Müller
...

Mengen-
eigenschaft!

41

5.2.2 Relationenspezifische Operationen

(3|30)

Eigenschaften von π :

- 1) Duplikate werden eliminiert (da Mengen)
- 2) geg. $r: (A \mid \dots)$

Falls $A1 \subseteq A2 \subseteq A$,

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

42

5.2.2 Relationenspezifische Operationen

(4|30)

B) Selektion: Auswahl von Tupeln

sei b eine Bedingung (=Prädikat)

(„Selektionsbedingung“):

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

Beispiel:

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

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

43

5.2.2 Relationenspezifische Operationen

(5|30)

Eigenschaften von σ :

geg. $r_1: (A \mid \dots), r_2: (B \mid \dots)$

- 1) $\sigma_{[b1]}(\sigma_{[b2]}(r_1)) = \sigma_{[b2]}(\sigma_{[b1]}(r_1))$
- 2) $\sigma_{[b1 \wedge b2]}(r_1) = \sigma_{[b1]}(\sigma_{[b2]}(r_1))$
- 3) $\sigma_{[b1 \vee b2]}(r) = \sigma_{[b1]}(r) \cup \sigma_{[b2]}(r)$
- 4) $\sigma_{[b]}(r_1 \cup r_2) = \sigma_{[b]}(r_1) \cup \sigma_{[b]}(r_2)$
- 5) $\sigma_{[b]}(r_1 \times r_2) = \begin{cases} \sigma_{[b]}(r_1) \times r_2 & \text{falls } b \text{ nur Attribute aus } A \\ r_1 \times \sigma_{[b]}(r_2) & \text{falls } b \text{ nur Attribute aus } B \end{cases}$

Generell:

- Für Attributnamen a in Bedingung b muss gelten $a \in A$
- Konstanten: Typ muss passen
- $X \Theta Y$: Θ muss für Werte von X, Y definiert sein

44

5.2.2 Relationenspezifische Operationen

(6|30)

Eigenschaften von σ :

- **Achtung:** σ erlaubt keine beliebige Auswahl von Zeilen!

- Folgendes geht nicht:

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

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

oder

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

45

5.2.2 Relationenspezifische Operationen

(7a|30)

Zur Syntax von $\sigma_{[b]}$:

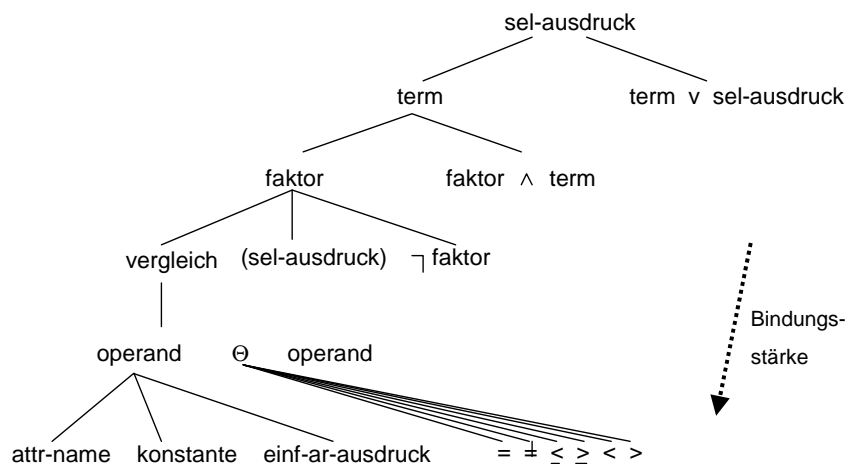
Kontextfreie Grammatik für Selektionsbedingungen b

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

46

5.2.2 Relationenspezifische Operationen

(7b|30)



47

5.2.2 Relationenspezifische Operationen

(8|30)

C) Join (Verbund)

geg.: $r_1: (A | \dots); r_2: (B | \dots)$

Verknüpfung von zwei Relationen r_1, r_2 zu einer Relation höheren Grades.

Mehrere Arten, z.B.

Natural Join (Natürlicher Verbund) : *

Theta-Join (Theta-, θ -Verbund) : *_[Bedingung]

48

5.2.2 Relationenspezifische Operationen

(9|30)

C1) Natural Join ($r_3 = r_1 * r_2$):

(auch Schreibweise $r_3 = r_1 \bowtie r_2$)

- Erzeugt eine neue Relation r_3 durch Verbinden von r_1 und r_2
 - bezüglich der **gemeinsamer Attribute** $A \cap B$ (\rightarrow „Join-Attribute“),
 - d.h. Attribute, die sowohl in r_1 , als auch in r_2 dieselbe Bezeichnung haben und über gleiche Werte verfügen.
- r_3 besitzt Attributmenge $A \cup B$

49

5.2.2 Relationenspezifische Operationen

(10|30)

Prozedurale Definition des Natural Join

geg. Relationen $r_1: (A | \dots)$, $r_2: (B | \dots)$

„Nested-Loop-Join“:

```

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
      end //for y
    end //for x
  
```

50

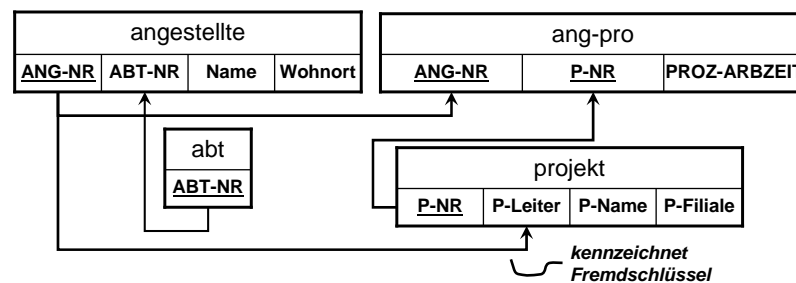
5.2.2 Relationenspezifische Operationen

(11|30)

Beispiel:

Betrachte *angestellte – projekt – ang-pro*

Gesucht: *Name & Proz-Arbeitszeit aller Angestellten, die an Projekt 770231 mitarbeiten.*



Lösung:

$\pi_{[Name, PROZ-ARBZEIT]}(\sigma_{[P-NR=770231]}(angestellte * ang-pro))$

51

5.2.2 Relationenspezifische Operationen

(12|30)

Vorgehen bei der Konstruktion des Ergebnisses:

1) $angestellte * ang-pro \rightarrow$ Zwischenergebnis1



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



3) $\pi_{[Name, PROZ-ARBZEIT]} \text{Zwischenergebnis2} \rightarrow$ Endergebnis



52

5.2.2 Relationenspezifische Operationen

(13|30)

1) Wert nach Join: $\text{angestellte} * \text{ang-pro} \rightarrow \text{Zwischenergebnis1}$

angestellte				ang-pro		
ANG-NR	Name	Wohnort	ABT-NR	P-NR	ANG-NR	PROZ-ARBEIT
1324	Schmitt	Heidelberg	35	770231	1324	100
1435	Mann	Bruchsal	32	770114	1435	60
2314	Groß	Karlsruhe	35	761235	1435	40
2412	Müller	Karlsruhe	32	770231	2314	100
2454	Schuster	Worms	31	770114	2412	100
2614	Klein	Mannheim	32	770114	2454	60
3115	Meyer	Karlsruhe	35	770008	2454	40
3190	Maus	Karlsruhe	30	770114	2614	30
3207	Müller	Mannheim	30	770008	2614	70
				770231	3115	50
				761235	3115	50
				770231	3190	50
				761235	3190	50
				761235	3207	100

Zwischenergebnis1

ANG-NR	Name	Wohnort	ABT-NR	P-NR	PROZ-ARBEIT
1324	Schmitt	Heidelberg	35	770231	100
1435	Mann	Bruchsal	32	770114	60
1435	Mann	Bruchsal	32	761235	40
2314	Groß	Karlsruhe	35	770231	100
2412	Müller	Karlsruhe	32	770114	100
2454	Schuster	Worms	31	770114	60
2454	Schuster	Worms	31	770008	40
2614	Klein	Mannheim	32	770114	30
2614	Klein	Mannheim	32	770008	70
3115	Meyer	Karlsruhe	35	770231	50
3115	Meyer	Karlsruhe	35	761235	50
3190	Maus	Karlsruhe	30	770231	50
3190	Maus	Karlsruhe	30	761235	50
3207	Müller	Mannheim	30	761235	100

5.2.2 Relationenspezifische Operationen

(14|30)

2) Wert nach Selektion der P-NR:

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

ANG-NR	Name	Wohnort	ABT-NR	P-NR	PROZ-ARBEIT
1324	Schmitt	Heidelberg	35	770231	100
2314	Groß	Karlsruhe	35	770231	100
3115	Meyer	Karlsruhe	35	770231	50
3190	Maus	Karlsruhe	30	770231	50

3) Wert nach Projektion:

$\pi_{[Name, PROZ-ARBEIT]} \text{Zwischenergebnis2} \rightarrow \text{Endergebnis}$

Name	PROZ-ARBEIT
Groß	100
Maus	50
Meyer	50
Schmitt	100

5.2.2 Relationenspezifische Operationen

(15|30)

C1) Natural Join ($r_3 = r_1 * r_2$):

Andere Definition (deskriptiv):

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

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

5.2.2 Relationenspezifische Operationen

(16|30)

Eigenschaften von *:

geg. $r_1: (A \mid \dots)$, $r_2: (B \mid \dots)$, $r_3: (C \mid \dots)$

1) $r_1 * r_2 = r_2 * r_1$ (kommutativ, bis auf Reihenfolge der Attribute)

2) $(r_1 * r_2) * r_3 = r_1 * (r_2 * r_3) = r_1 * r_2 * r_3$

3) $A \cap B = \emptyset \Rightarrow r_1 * r_2 = r_1 \times r_2$

4) $r_1 = \emptyset \Rightarrow r_1 * r_2 = \emptyset$

5) $A = B \Rightarrow r_1 * r_2 = r_1 \cap r_2$

6) $r_1 * r_1 = r_1$ (Sonderfall von 5)

7) $\sigma_{[b]}(r_1 * r_2) = \begin{cases} \sigma_{[b]}(r_1) * \sigma_{[b]}(r_2) & \text{falls } b \text{ nur Attribute aus } A \cap B \text{ besitzt} \\ \sigma_{[b]}(r_1) * r_2 & \text{falls } b \text{ nur Attribute aus } A \text{ besitzt} \\ r_1 * \sigma_{[b]}(r_2) & \text{falls } b \text{ nur Attribute aus } B \text{ besitzt} \end{cases}$

5.2.2 Relationenspezifische Operationen


(17|30)

Zusammenhänge zwischen Natural Join und Projektion

- Reihenfolge von Projektion und Join wichtig!
- Die Zerlegung einer Relation auf bestimmte Attributmengen (durch Projektion) kann verlustbehaftet sein, da man beim anschließenden Join nicht mehr exakt die Tupel erhält, die ursprünglich vorhanden waren.

a) $r : (U); U = A \cup B$


Beh.: $r \subseteq \pi_{[A]} r * \pi_{[B]} r$ („ \neq “ ist möglich!)

Beweis: 

b) $r_1 : (A); r_2 : (B)$

Beh.: $\pi_{[A]}(r_1 * r_2) \subseteq r_1$

$\pi_{[B]}(r_1 * r_2) \subseteq r_2$ („ \neq “ ist möglich!)

Beweis: 



57

5.2.2 Relationenspezifische Operationen

(18|30)

a) Beh.: $r \subseteq \pi_{[A]} r * \pi_{[B]} r$

Beweis:

$$\begin{array}{l} \text{zu zeigen: Tupel } x \in r \Rightarrow x \in \pi_{[A]} r * \pi_{[B]} r \\ x \in r \Rightarrow \left. \begin{array}{l} x.A \in \pi_{[A]} r \wedge \\ x.B \in \pi_{[B]} r \end{array} \right\} \begin{array}{l} \text{nach Def.} \\ \text{Projektion} \end{array} \end{array}$$

Aus der Definition des nat. Verbundes folgt dann:

$$x \in \pi_{[A]} r * \pi_{[B]} r, \text{ da } x.A * x.B = x$$

	A	B	
$x \in r$	e	g	f
$x.A = y$	e	g	$\in \pi_{[A]} r$
$x.B = z$	g	f	$\in \pi_{[B]} r$
$x.A * x.B$	e	g	= x

58

5.2.2 Relationenspezifische Operationen

(19|30)

a) Beispiel zu

$$r \not\subseteq \pi_{[A]} r * \pi_{[B]} r$$

$$A = \{e, f\}$$

$$B = \{f, g\}$$

r	e	f	g	$\pi_{[A]}$	e	f	$\pi_{[B]}$	f	g
	0	0	0		0	0		0	0
	1	0	1		1	0		0	1

$\pi_{[A]} r * \pi_{[B]} r$	e	f	g
	0	0	0
	0	0	1
	1	0	0
	1	0	1

} nicht in r!

$$\pi_{[e,f]} r * \pi_{[f,g]} r \neq r$$



59

5.2.2 Relationenspezifische Operationen

(20|30)

b) „ \subseteq “

zu zeigen: $z \in r_1 * r_2 \Rightarrow z.A \in r_1, z.B \in r_2$

$$\begin{array}{l} z \in r_1 * r_2 \\ \Rightarrow \exists x \in r_1, \exists y \in r_2 : x.(A \cap B) = y.(A \cap B), z = x * y \\ \Leftrightarrow \exists x \in r_1, \exists y \in r_2 : [z.A = x \wedge z.B = y] \\ \Rightarrow z.A (= x) \in r_1 \wedge z.B (= y) \in r_2 \end{array}$$

	A	B	
$z \in r_1 * r_2$	e	g	f
$x =$	e	g	$\in r_1$
$y =$	g	f	$\in r_2$

$x = z.A$ $y = z.B$

60

5.2.2 Relationenspezifische Operationen

(21|30)

b) Beispiel zu
 $\pi_{[B]}(r_1 * r_2) \subsetneq r_2$

A={e,f}

r ₁	e	f
	1	0
	0	0

B={f,g}

r ₂	f	g
	0	0
	1	1

r ₁ * r ₂	e	f	g
	1	0	0
	0	0	0

$\pi_{[B]}(r_1 * r_2)$	f	g
	0	0



61

5.2.2 Relationenspezifische Operationen

(22|30)

C2) Theta-Join: (θ -Verbund)

- Verbindung von Relationen bezüglich beliebiger Attribute,
- bei beliebigen Vergleichsmöglichkeiten mit Prädikat θ

Schreibweise: $r_1 * [Join-Bedingung \theta] r_2$
 oder auch: $r_1 \bowtie [Join-Bedingung \theta] r_2$

Definition: $r_1 *_{[\theta]} r_2 = \sigma_{[\theta]}(r_1 \times r_2)$

- In die Ergebnisrelation werden alle Tupel aus $r_1 \times r_2$ übernommen, für die das Prädikat θ durch Einsetzen der Werte der Tupel in den entsprechenden Variablen den Wert „wahr“ zurückliefert.

62

5.2.2 Relationenspezifische Operationen

(23|30)

Zur Syntax:

Allgemein:

- Join-Bedingung θ - wie Selektionsbedingung

Attributnamen:

- falls nicht eindeutig (z.B. NAME in A und B) dann eindeutig machen durch qualifizieren: $r_1.NAME, r_2.NAME$

Beispiel für $r_1: (A | \dots), r_2: (B | \dots), a \in A, b \in B,$
 $\theta : a > b$ (mit $\ominus : >$)

$$r_1 *_{[a > b]} r_2$$

63

5.2.2 Relationenspezifische Operationen

(24|30)

Beispiel:

$angestellte *_{[WOHNORT=P-FILIALE \wedge P-NR=761235]} projekt$

	ANG-NR	Name	Wohnort	ABT-NR	P-NAME	P-NR	P-FILIALE	P-LEITER
	2412	Müller	Karlsruhe	32	P-1	761235	Karlsruhe	3115
	2314	Groß	Karlsruhe	35	P-1	761235	Karlsruhe	3115
	3190	Maus	Karlsruhe	30	P-1	761235	Karlsruhe	3115
▶	3115	Meyer	Karlsruhe	35	P-1	761235	Karlsruhe	3115

Bemerkungen:

- Falls θ ausschließlich auf Gleichheit unter den Attributen aus A,B prüft, wird die Verbindung auch **Equi-Join** genannt (Spezialfall des Theta-Joins)
- Auch der **Natural Join** kann in relationaler Algebra durch Verwendung des Theta-Joins ausgedrückt werden (Wie?)

64

5.2.2 Relationenspezifische Operationen

(25|30)

Eigenschaften von $*_{[\theta]}$:

geg. $r_1: (A \mid \dots)$, $r_2: (B \mid \dots)$, $r_3: (C \mid \dots)$

1) $r_1 *_{[\theta]} r_2 = r_2 *_{[\theta]} r_1$ (kommutativ, bis auf Reihenfolge der Attribute)

- 2) nicht notwendigerweise assoziativ, da sich evtl. die Bedingungen in θ auf nicht vorhandene Attribute beziehen können,

z.B. $A=\{a,b\}$, $B=\{b,c\}$, $C=\{c,d\}$

$(r_1 *_{[r1.b>r2.b]} r_2) *_{[r1.a<r3.d]} r_3$ vgl. mit

$r_1 *_{[r1.b>r2.b]} (r_2 *_{[r1.a<r3.d]} r_3)$ nicht sinnvoll

5.2.2 Relationenspezifische Operationen

(26|30)

Weitere Join-Varianten

geg.: $r_1: (A \mid \dots)$, $r_2: (B \mid \dots)$

- (Left) Semi-Join

$$r_1 \ltimes r_2 = \pi_{[A]}(r_1 * r_2)$$

- wird benutzt, wenn die Information benötigt wird, welche Tupel der einen Relation (r_1) erfolgreich mit Tupeln der anderen Relation (r_2) verbunden werden können

- Anwendung in verteilten Datenbanken;
Ziel: weniger Daten (d.h. nicht komplette Relationen) übertragen

z.B. r_1 in Paris, r_2 in London; Berechne $r_1 * r_2$ in Paris

- 1) $temp_1 := \pi_{[A \cap B]}(r_1)$ in Paris
 \rightsquigarrow Projektion auf Join-Attribute
- 2) Transport von $temp_1$ nach London
- 3) $temp_2 := r_2 \ltimes r_1 = r_2 * temp_1$ in London
 $\rightsquigarrow temp_2$ enthält nur Tupel von r_2 , die passende Tupel in r_1 haben
- 4) Transport von $temp_2$ nach Paris
- 5) $r_1 * temp_2 = r_1 * r_2$ in Paris

5.2.2 Relationenspezifische Operationen

(27|30)

Weitere Join-Varianten

geg.: $r_1: (A \mid \dots)$, $r_2: (B \mid \dots)$

- Left-Outer-Theta-Join

$$r_1 \bowtie_{[\theta]} r_2 = r_1 *_{[\theta]} r_2 \cup \underbrace{(r_1 \setminus \pi_{[A]}(r_1 *_{[\theta]} r_2)) \times \{NULL\}^{|B|}}_{|B| \text{ mal}}$$

- nicht verbindbare Tupel der linksstehenden Relation (r_1) werden trotzdem ins Join-Ergebnis übernommen, und
- solche Tupel von r_1 werden mit (NULL,..., NULL) verkettet

|B| mal

5.2.2 Relationenspezifische Operationen

(28|30)

Weitere Join-Varianten

geg.: $r_1: (A \mid \dots)$, $r_2: (B \mid \dots)$

- Right-Outer-Theta-Join

$$r_1 \bowtie_{[\theta]} r_2 = r_1 *_{[\theta]} r_2 \cup \underbrace{\{NULL\}^{|A|} \times (r_2 \setminus \pi_{[B]}(r_1 *_{[\theta]} r_2))}_{|A| \text{ mal}}$$

- nicht verbindbare Tupel der rechtsstehenden Relation (r_2) werden trotzdem ins Join-Ergebnis übernommen, und
- solche Tupel von r_2 werden mit (NULL,..., NULL) verkettet

|A| mal

5.2.2 Relationenspezifische Operationen

(29|30)

Weitere Join-Varianten

geg.: $r_1: (A | \dots), r_2: (B | \dots)$

- (Full) Outer-Theta-Join

$$r_1 \bowtie_{[\theta]} r_2 = r_1 *_{[\theta]} r_2 \cup (r_1 \setminus \pi_{[A]}(r_1 *_{[\theta]} r_2)) \times \{NULL\}^{|B|} \cup \{NULL\}^{|A|} \times (r_2 \setminus \pi_{[B]}(r_1 *_{[\theta]} r_2))$$

- nicht verbindbare Tupel sowohl der linksstehenden Relation (r_1) als auch der rechtsstehenden Relation (r_2) werden trotzdem ins Join-Ergebnis übernommen

Vgl. auch Garcia-Molina, Widdom, „Database Systems“, 2002
Lang, Lockemann, Datenbankeinsatz, 1995

5.2.2 Relationenspezifische Operationen

(30|30)

Beispiele: angestellte₁:

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

t_2 :

ANG-NR	Hobby
1324	Lesen
1324	Sport
1325	Reiten

angestellte₁ $\bowtie_{[r1.ANG-NR=r2.ANG-NR]}$ t_2

angestellte1.ANG-NR	Name	Wohnort	ABT-NR	t2.ANG-NR	Hobby
3115	Meyer	Karlsruhe	35		
3207	Müller	Mannheim	30		
2814	Klein	Mannheim	32		
3190	Maus	Karlsruhe	30		
2314	Groß	Karlsruhe	35		
1324	Schmitt	Heidelberg	35	1324	Lesen
1324	Schmitt	Heidelberg	35	1324	Sport
1435	Mann	Bruchsal	32		
2412	Müller	Karlsruhe	32		
2454	Schuster	Worms	31		

angestellte₁ $\bowtie_{[r1.ANG-NR=r2.ANG-NR]}$ t_2

angestellte1.ANG-NR	Name	Wohnort	ABT-NR	t2.ANG-NR	Hobby
1324	Schmitt	Heidelberg	35	1325	Reiten
1324	Schmitt	Heidelberg	35	1324	Lesen
1324	Schmitt	Heidelberg	35	1324	Sport
1435	Mann	Bruchsal	32		
2314	Groß	Karlsruhe	35		
2412	Müller	Karlsruhe	32		
2454	Schuster	Worms	31		
2814	Klein	Mannheim	32		
3115	Meyer	Karlsruhe	35		
3190	Maus	Karlsruhe	30		
3207	Müller	Mannheim	30		

angestellte₁ $\bowtie_{[r1.ANG-NR=r2.ANG-NR]}$ t_2

angestellte1.ANG-NR	Name	Wohnort	ABT-NR	t2.ANG-NR	Hobby
1324	Schmitt	Heidelberg	35	1324	Lesen
1324	Schmitt	Heidelberg	35	1324	Sport
1325					Reiten

5.2.3 Ausdrücke der Relationenalgebra:

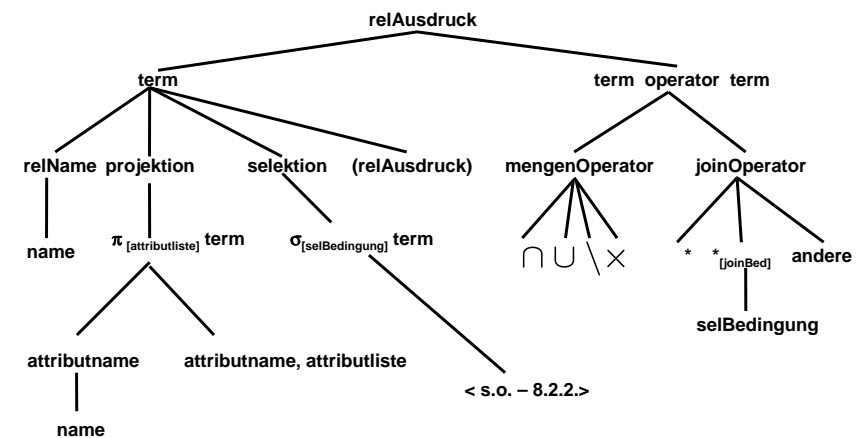
(1|2)

Zur Syntax von Ausdrücken in Relationenalgebra:

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 | \setminus | \times
8. joinOperator ::= * | * [joinBedingung] | (andere)
9. selBedingung ::= < s.o. – 8.2.2.>
10. joinBedingung ::= selBedingung
11. relName ::= name
12. attributname ::= name

5.2.3 Ausdrücke der Relationenalgebra:

(1b|2)



5 Das relationale Datenmodell

5.1 Grundlagen	2
5.2 Relationale Algebra	34
5.3 Anmerkungen zu relationalen Sprachen.....	73
5.4 Normalformen	84



5.3.3 Anmerkungen zu relationalen Sprachen

(1|10)

Sprachvarianten für das Relationenmodell:

- Auf Relationen basierende Algebra
 - 1) **Relationenalgebra (siehe 8.2)** (prozedural)
 - 2) **Relationenkalkül / Tupelkalkül** (deskriptiv)
 - 3) **Domänenkalkül** (deskriptiv)
- Grafikorientierte Sprachen
 - 4) **Query by Example (QBE)** (deskriptiv)
- Abbildungsorientierte Sprachen
 - 5) **Structured Query Language (SQL)** (deskriptiv)

5.3.3 Anmerkungen zu relationalen Sprachen

(2|10)

zu 1) Relationenalgebra

Nicht Satz- / Tupel- / Record-weise Verarbeitung, sondern :

- mengen- und relationenorientierte Operationen!
- Relation(en) $\xrightarrow{\text{Operator}}$ Relation
- Es wird immer eine genaue Folge von Operationen für die Abarbeitung angegeben
- nur Anfragesprache

zu 2) Relationenkalkül / Tupelkalkül

Nur Angabe der Def. von Tupeln einer Relation

- zur Beschreibung: Prädikatenlogik 1. Stufe
- Variablen beziehen sich auf Tupel einzelner Relationen
- Vergleichsoperatoren, Boolesche Operatoren, Quantoren
- keine genaue Abarbeitungsreihenfolge
- Bsp: `(angestellte.Name):.
angestellte.ANG-NR=ang-pro.ANG-NR \wedge ang-pro.pnr = 770231`

5.3.3 Anmerkungen zu relationalen Sprachen

(3|10)

zu 3) Domänenkalkül

- ähnlich 2), jedoch
- Variablen beziehen sich nicht auf Tupel, sondern auf Wertebereiche (Domänen) von Attributen
- Bsp: `Aname as Zeichen(20), Aangnr as Zahl, APpnr as Zahl
(Aname)
WHERE angestellte(ANG-NR: Aangnr, Name: Aname)
 \wedge ang-pro(ANG-NR:Aangnr, P-NR: APpnr)
 \wedge APpnr=770231`

1), 2), 3) sind im folgenden Sinne äquivalent:

- zu jedem Ausdruck in 1) gibt es einen in 2), 3), der die gleiche Relation identifiziert
- zu jedem Ausdruck aus 2), 3), der eine endliche Relation bezeichnet, gibt es einen gleichwertigen in 1)

5.3.3 Anmerkungen zu relationalen Sprachen

(4|10)

zu 4) Query By Example (QBE)

- Benutzer formuliert Anfragen, indem er auf einer graphischen Benutzeroberfläche Eintragungen macht

Beispiel:

„finde die Namen aller Angestellten mit Beruf=PROG in der Abteilung 30“

ANGEST	ANG-NR	NAME	ANSCHRIFT	BERUF	ABT-NR
		<u>P.Meyer</u>		PROG	30

- PROG, 30 (nicht unterstrichen) sind Konstanten
- P.Meyer stellt Beispiel-Element (Variable) dar
 - Meyer muss nicht in Datenbank vorhanden sein
 - „P.“ bedeutet „Print“
 - alle mit P qualifizierten Daten werden ausgegeben

(vgl. auch Schlageter/Stucky)

77

5.3.3 Anmerkungen zu relationalen Sprachen

(5|10)

Einige Einschränkungen der relationalen Sprachen 1-3:

- keine Funktionen für numerische Berechnungen, insbes. **keine anwendungsspezifisch definierte Funktionen** möglich (d.h. man kann keine eigene Funktionssymbole einführen, z.B. „d“ für Durchschnitt o.ä.)
- keine Möglichkeit, Tupel nach Objektzugehörigkeit zu **gruppieren** (z.B. bei Berechnung der Umsatzsumme pro Produktgruppe ⇒ Bilde in Relation Gruppen von Produkten ⇒ Summe der Umsätze jeweils innerhalb einer Gruppe)
- **keine Rekursion** möglich (nützlich für „besteht aus“-Beziehungen, z.B. Teile eines Autos bestehen wiederum aus Teilen ⇒ Rekursion notwendig zur Ermittlung der insgesamt benötigten Teile bei der Fertigung)

Diese Einschränkungen sind z.T. durch Erweiterungen in SQL gelockert worden

78

5.3.3 Anmerkungen zu relationalen Sprachen

(6|10)

zu 5) Structured Query Language (SQL)

- aus Tupelkalkül (2) hervorgegangen
- bietet Möglichkeiten sowohl zur Definition/Verwaltung, als auch Abfrage/Veränderung von Daten mit Hilfe von
 - DML („data manipulation language“)
 - DDL („data definition language“)
- interaktiv nutzbar
- keine mathematische Notation
- in zahlreichen DBMS implementiert
- aktueller Standard: SQL2003
- aber:
SQL ist keine direkte Umsetzung des relationalen Modells. Beispielsweise sind Tabellen in SQL keine Mengen, sondern Multimengen (engl. „Bags“), d.h. identische Tupel können mehrfach auftreten.

79

5.3.3 Anmerkungen zu relationalen Sprachen

(7|10)

zu 5) Structured Query Language (SQL)

- Abbildungsorientierte Sprache
 - **Definitionsbereich**=Relation (FROM; Prädikat WHERE erlaubt Einschränkung)
 - **Zuordnungsvorschrift** besteht aus Auswahl (SELECT) von Attributen aus dem Definitionsbereich; ⇒ damit auch **Wertebereich** definiert

Beispiel: `SELECT NAME` } Wertebereich
`FROM angest` } Definitionsbereich
`WHERE ABT-NR = 20` }

(vgl. Schlageter/Stucky, 1983)

80

zu 5) Structured Query Language (SQL)

Beispiel (Equi-Join):

```
SELECT angestellte.ANG-NR, angestellte.Name
FROM angestellte, projekt
WHERE ((projekt.P-NR=761235)
      AND (angestellte.Wohnort=projekt.P-FILIALE))
```

Beispiel (SQL-DDL):

```
CREATE TABLE angestellte
(ANG-NR PersonalNr PRIMARY KEY,
 NAME VARCHAR (30),
 WOHNORT VARCHAR (30),
 ABT-NR INTEGER);

CREATE TABLE ang-pro
(P-NR ProjektNr REFERENCES projekt (P-NR),
 ANG-NR PersonalNr REFERENCES angestellte (ANG-NR),
 PROZ-ARBZEIT ProzentAngabe,
 PRIMARY KEY (P-NR, ANG-NR))
```

↙ Fremdschlüssel

zu 5) Structured Query Language (SQL)

Zusammenhang zwischen SQL und Relationenalgebra

- Anfragen in SQL sind deskriptiv, jedoch muss für die Beantwortung einer Anfrage ein konkreter Anfrageplan erstellt werden
- Dieser Anfrageplan wird mit Hilfe von Relationenalgebra formuliert
- Auf ihm können Optimierungen (z.B. durch Umformungen) basierend auf den algebraischen Gesetzen durchgeführt werden

zu 5) Structured Query Language (SQL)

Beispiel (Optimierung):

```
SELECT PROZ-ARBEIT
FROM (angest
      NATURAL JOIN
      ANGPRO)
WHERE ANG-NR=1435
```



Optimierter Anfrageplan

```
π[PROZ-ARBEIT](
  σ[ANG-NR=1435](angest)
  *
  σ[ANG-NR=1435](angpro))
```

Ergebnis hier:

Durch Umstellung der Selektion müssen weniger Tupel beim Join berücksichtigt werden!
(weniger Schleifendurchläufe beim Nested-Loop-Join)

5 Das relationale Datenmodell



5.1 Grundlagen.....	2
5.2 Relationale Algebra	34
5.3 Anmerkungen zu relationalen Sprachen.....	74
5.4 Normalformen	85
5.4.1 Ziele/Probleme beim Aufbau relationaler Datenbanken.....	85
5.4.2 Funktionale Abhängigkeiten.....	87
5.4.3 Die abgeschlossene Hülle F ⁺	97
5.4.4 Voll funktionale Abhängigkeiten.....	107
5.4.5 Zweite Normalform (2NF)	109
5.4.6 Transitive funktionale Abhängigkeiten.....	112
5.4.7 Dritte Normalform (3NF), Boyce-Codd-Normal-Form (BCNF)....	115
5.4.8 Ausblick	122

5.4.1 Ziele/Probleme beim Aufbau relationaler Datenbanken (1|2)

Ziel:

- Datenbank sollte zu **jedem Zeitpunkt**
- **ein korrektes Abbild der realen Welt** darstellen!

Relationentheorie ermöglicht eine *systematische Vorgehensweise* zur Lösung bereits dargestellter Probleme:

- Zusammenfassung von Attributen zu einer Relation
 ⇒ insertion / deletion / update anomaly
 Redundanz / unvollständige Tupel
- Zerlegung einer Relation in mehrere Relationen
 ⇒ evtl. Informationsverlust

Data Design: Bildung „korrekter“ Relationenschemata entsprechend den Regeln der Relationentheorie.

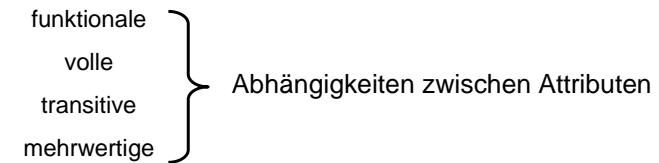
5.4.1 Ziele/Probleme beim Aufbau relationaler Datenbanken (2|2)

Vermeidung der dargestellten Probleme

- Anwendung und Einsatz geeigneter Kriterien und Methoden für
 - geeigneten Aufbau von Relationen,
 - geeignete Zerlegung einer Relation in mehrere kleinere Relationen

D.h. über 1NF hinausgehende weitere **Normalisierung** (zu 2NF / 3NF / BCNF / 4NF / ...)

grundlegende **Konzepte** dafür:



Ausgangspunkt: (beispielweise) **ER-Modell**;

darauf Anwendung der Relationentheorie und Normalisierungslehre

5.4.2 FA's: Definition, Beispiele, Eigenschaften (1|12)

geg.: U Attributmenge; $A, B, \dots \subseteq U$
 $r: (U | F)$ Relation über U

Wir betrachten **spezielle semantische Integritätsbedingungen** über U

- Form: $A \rightarrow B$ ($A, B \subseteq U$)
- Bezeichnung: „**funktionale Abhängigkeit**“ (fA)
 („B ist von A **funktional abhängig**“)
- Bedeutung für $X \subseteq \text{dom}(U)$
 $A \rightarrow B(X) : \Leftrightarrow$
 $\forall x, y \in X: (x.A = y.A \Rightarrow x.B = y.B)$

d.h.: Stimmen 2 Tupel aus X in allen A-Werten überein, so stimmen sie auch in allen B-Werten überein.

5.4.2 FA's: Definition, Beispiele, Eigenschaften (2|12)

Beispiel für funktionale Abhängigkeiten

Personal						
Name a	PersNr b	StOrt c	Ubereich d	Abt e	GebNr f	Gehalt g
Frits	17	Aholming	Electro	F&E	11	44.000
Frans	9133	Aholming	Electro	Contr	11	88.200
Lubbe	321	Aholming	Electro	Vertr	8	38.000
Einzian	17	München	Mechanik	F&E	3	53.000
Truhel	54	Karben	Kfz	F&E	2	43.500
Jöndhard	739	Karben	Kfz	F&E	2	45.300
Frits	17	Fürth	Mechanik	Contr	4	90.000

$U = \{a, b, c, d, e, f, g\}$
 $bc = \text{Schlüssel}$
 $c: \text{jeweils nur 1 d}$
 $e: 1 f (\text{pro } c)$

Andere Schreibweise:
 $bc \rightarrow U, b \not\rightarrow U, c \not\rightarrow U$
 $c \rightarrow d$
 $ec \rightarrow f$

Es gilt auch (z.B.):
 $a \rightarrow a$
 $ac \rightarrow ac$
 $abc \rightarrow a$

5.4.2 FA's: Definition, Beispiele, Eigenschaften

(3|12)

Beispiel

LIEF: $(P, B, L \mid PB \rightarrow L, L \rightarrow B)$

Allgemein gilt:

$PS \rightarrow$ beliebiges Attribut

(für jede Relation)

5.4.2 FA's: Definition, Beispiele, Eigenschaften

(4|12)

Bezeichnung:

$\Phi(U) ::= \{ A \rightarrow B \mid A, B \subseteq U \}$

Beispiel:

$U = \{a, b\}$; Teilmengen: $\emptyset, \{a\}, \{b\}, \{ab\}$,
(wir schreiben: \emptyset, a, b, ab)

$\Phi(U) = \{ \emptyset \rightarrow \emptyset, \emptyset \rightarrow a, \emptyset \rightarrow b, \emptyset \rightarrow ab,$
 $a \rightarrow \emptyset, a \rightarrow a, a \rightarrow b, a \rightarrow ab,$
 $b \rightarrow \emptyset, b \rightarrow a, b \rightarrow b, b \rightarrow ab,$
 $ab \rightarrow \emptyset, ab \rightarrow a, ab \rightarrow b, ab \rightarrow ab \}$

d.h.:

Ist $f \in \Phi(U)$, so ist f eine funktionale Abh. über U und hat die Form

$A \rightarrow B$ (A, B geeignet $\subseteq U$)

(d.h. $f = A \rightarrow B$).

5.4.2 FA's: Definition, Beispiele, Eigenschaften

(5|12)

Betrachte $r : (U \mid A \rightarrow B)$; dann:

- $A \rightarrow B$ (r)
Wir sagen: B ist funktional abhängig von A (in r)
A ist Determinante (von B) (in r)
A bestimmt B
- „A-Wert impliziert B-Wert“:
 - gilt zu jedem Zeitpunkt
 - bedeutet nur „dass“, nicht „wie“!

Wir schreiben:

$A \not\rightarrow B(r)$, falls „nicht $A \rightarrow B(r)$ “

$A \not\rightarrow B(r)$: Es kann $r^t = X$ geben mit $A \rightarrow B(X)$!

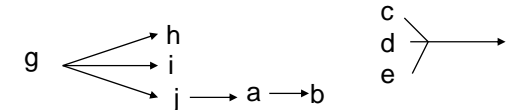
5.4.2 FA's: Definition, Beispiele, Eigenschaften

(8|12)

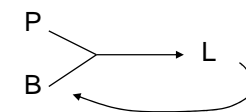
Darstellung von (kleinen) Mengen von fA's im **Abhängigkeitsgraphen**

Beispiel:

- $a, b, c, d, e, f, g, h, i, j$ seien Attribute
 $F = \{ \boxed{a \rightarrow b}, \boxed{cde \rightarrow f}, \boxed{g \rightarrow hij}, \boxed{j \rightarrow a} \} \subseteq \Phi(a, b, c, d, e, f, g, h, i, j)$



- lief: $(PBL \mid \boxed{PB \rightarrow L}, \boxed{L \rightarrow B})$
Schlüssel: PB, PL; NSA: \emptyset



5.4.2 FA's: Definition, Beispiele, Eigenschaften

(9|12)

Lemma 8.4.2: (Regeln für \rightarrow bzw. „ \rightarrow (r)“)

Vor.: $r: (U \mid F), F \subseteq \Phi(U); A, B, C, D \subseteq U$

Dann gelten die folgenden **Eigenschaften**:

- | | | | |
|------------------|---|--|---|
| Armstrong-Regeln | { | (1) $B \subseteq A \models A \rightarrow B$
<i>(insb. gilt immer: $A \rightarrow A$)</i> | Reflexivität
<i>(engl.: reflexive rule)</i> |
| | | (2) $A \rightarrow B \models AC \rightarrow BC$ | Erweiterungsregel
<i>(engl.: augmentation rule)</i> |
| | | (3) $A \rightarrow B, B \rightarrow C \models A \rightarrow C$ | Transitivität
<i>(engl.: transitive rule)</i> |
- aus (1)-(3) ableitbar:
- | | |
|--|---|
| (4) $A \rightarrow B, A \rightarrow C \models A \rightarrow BC$ | Vereinigungsregel
<i>(engl.: union/additive rule)</i> |
| (5) $A \rightarrow B, BC \rightarrow D \models AC \rightarrow D$ | Pseudotransitivität
<i>(engl.: pseudotransitive rule)</i> |
| (6) $A \rightarrow BC \models A \rightarrow B, A \rightarrow C$
<small>andere Formulierung: $A \rightarrow B, C \subseteq B \models A \rightarrow C$</small> | Zerlegungsregel
<i>(engl.: decomposition rule)</i> |

5.4.2 FA's: Definition, Beispiele, Eigenschaften

(10|12)

Beweise:

(1) Beh: $B \subseteq A \models A \rightarrow B$

Bew.: Sei $B \subseteq A$ und $x, y \in r$.

Zu zeigen: $x.A = y.A \Rightarrow x.B = y.B$

$$\begin{aligned} x.A = y.A &\Leftrightarrow x.a = y.a \text{ für alle } a \in A \\ &\Rightarrow x.a = y.a \text{ für alle } a \in B \text{ (da } B \subseteq A) \\ &\Leftrightarrow x.B = y.B \end{aligned}$$

(2) Beh: $A \rightarrow B \models AC \rightarrow BC$

Bew.: Es gelte $A \rightarrow B$ (r); $x, y \in r$.

Zu zeigen: $x.AC = y.AC \Rightarrow x.BC = y.BC$

$$\begin{aligned} x.AC = y.AC &\Leftrightarrow x.A = y.A \text{ und } x.C = y.C \\ &\Rightarrow x.B = y.B \text{ und } x.C = y.C \\ &\quad \text{A} \rightarrow \text{B} \\ &\Leftrightarrow x.BC = y.BC \end{aligned}$$

5.4.2 FA's: Definition, Beispiele, Eigenschaften

(11|12)

(3) Beh.: $A \rightarrow B, B \rightarrow C \models A \rightarrow C$

Bew.: Es gelte $A \rightarrow B$ (r) und $B \rightarrow C$ (r); $x, y \in r$.

Zu zeigen: $x.A = y.A \Rightarrow x.C = y.C$

$$\begin{aligned} x.A = y.A &\Rightarrow x.B = y.B \\ &\quad \text{A} \rightarrow \text{B} \\ &\Rightarrow x.C = y.C \\ &\quad \text{B} \rightarrow \text{C} \end{aligned}$$

d.h. $x.A = y.A \Rightarrow x.C = y.C$,
also $A \rightarrow C$

5.4.2 FA's: Definition, Beispiele, Eigenschaften

(12|12)

Hinweis: $AB = A \cup B = B \cup A = BA$

$$\left. \begin{aligned} (4) \quad A \rightarrow B &\models A \rightarrow AB \\ &\quad (2), \text{Erw. } A, AA=A \\ A \rightarrow C &\models AB \rightarrow BC \\ &\quad (2), \text{Erw. } B \end{aligned} \right\} \models A \rightarrow BC \quad (3)$$

$$\left. \begin{aligned} (5) \quad A \rightarrow B &\models AC \rightarrow BC \\ &\quad (2), \text{Erw. } C \\ BC \rightarrow D & \end{aligned} \right\} \models AC \rightarrow D \quad (3)$$

$$\left. \begin{aligned} (6) \quad A \rightarrow BC \\ \text{mit (1):} \\ BC \rightarrow B \\ BC \rightarrow C \end{aligned} \right\} \models A \rightarrow B, A \rightarrow C \quad (3)$$

5.4.3 Die abgeschlossene Hülle F^+

(1|10)

Definition: Zu $r: (U | F)$, (d.h. zu Relation r mit Attributmenge U , Menge von fA 's F)
 $F \subseteq \Phi(U)$ ist

$F^+ ::=$ Menge aller fA 's, die mit F in r gelten,
 die **abgeschlossene Hülle** von F .

Äquivalenz von fA 's:

$F, G \subseteq \Phi(U)$

F, G sind „äquivalent“ ($F \sim G$) $:\Leftrightarrow F^+ = G^+$

bei $F \sim G$ sagt man auch „ G überdeckt F “ bzw. „ G ist Überdeckung von F “

Triviale funktionale Abhängigkeit:

$f \in \Phi(U)$ ist „triviale“ fA $:\Leftrightarrow f$ gilt immer / in jeder Relation,

d.h. $f \in \emptyset^+$ (von \emptyset erzeugt)

$\emptyset^+ = \{A \rightarrow B \mid B \subseteq A \subseteq U\}$

Einfache funktionale Abhängigkeit:

$f \in \Phi(U)$ heißt „einfach“ („einfache fA “)

$:\Leftrightarrow f = A \rightarrow b$ mit $b \in U$ (d.h. b ist Attribut)

5.4.3 Die abgeschlossene Hülle F^+

(2|10)

Minimale Überdeckung:

$F, G \subseteq \Phi(U)$:

G ist eine „minimale Überdeckung“ von F

$:\Leftrightarrow$ (1) $G \sim F$

(2) G besteht nur aus *einfachen* fA 's

(3) in G ist keine fA überflüssig

(d.h.: $G' \subsetneq G \Rightarrow G'^+ \neq G^+$)

(d.h. wird eine fA aus G weggelassen, ist die resultierende Menge keine Überdeckung von F mehr)

Vorgehensweise: Minimale Überdeckung von F

- F modifizieren in nicht-triviale einfache fA 's
- ggf. fA 's einfacher machen
- überflüssige fA 's streichen

5.4.3 Die abgeschlossene Hülle F^+

(3|10)

Definition: Für $A \subseteq U$ ist A^+ die Menge aller von A funktional abhängigen Attribute:

$A^+ ::= \{b \in U \mid A \rightarrow b (r)\}$

Lemma:

Vor. $A, B \subseteq U, F \subseteq \Phi(U)$

(1) $A \rightarrow B \in F^+ \Leftrightarrow B \subseteq A^+$

(2) $F^+ = \{A \rightarrow B \mid A \subseteq U, B \subseteq A^+\}$

d.h. wenn man alle A^+ hat, hat man im wesentlichen auch F^+

- Wie aufwendig: „alle A^+ berechnen“?
Wenn $|U|=n$, dann gibt es 2^n verschiedene A^+ !
- Braucht man wirklich alle A^+ ?
Nein, im wesentlichen nur die A^+ , wo A linke Seite einer fA $f \in F$:
 $f = A \rightarrow Y \in F$ (Y geeignet).

5.4.3 Die abgeschlossene Hülle F^+

(4|10)

Beispiel: $r: (a b c d e f g | F)$

Wdh.:

F:

1. $a \rightarrow b c$

2. $a e \rightarrow g$

3. $a g \rightarrow f$

4. $b d \rightarrow f$

5. $b g \rightarrow c$

6. $c \rightarrow a d$

7. $e \rightarrow a$

8. $g \rightarrow e$

(1) $B \subseteq A \models A \rightarrow B$

(insb. gilt immer: $A \rightarrow A$)

(2) $A \rightarrow B \models AC \rightarrow BC$

(3) $A \rightarrow B, B \rightarrow C \models A \rightarrow C$

Bsp.-Rechnungen

$(ab)^+ = abcdf$

$(a)^+ = abcdf$

$c^+ = cadbf$

$g^+ = geabcdf$

Beweis
(nächste Folie)

● $f \rightarrow acd \in F^+$? berechne f^+

$f^+ = f$

$e \rightarrow b c g \in F^+$? berechne e^+

$e^+ = eabcgfd$

Andere Fragen:

5.4.3 Die abgeschlossene Hülle F^+

(5|10)

Beweis :

$ab \rightarrow ab$ wg.(1) (aus Lemma 1.1)
fA1: $a \rightarrow bc$ (2) Erweiterungsregel mit ab
 $ab \rightarrow abc$ \downarrow

$ab \rightarrow abc$ \downarrow (3) Transitivität
fA6: $c \rightarrow ad$ (2) mit abc
 $abc \rightarrow abcd$ \downarrow

\downarrow (3)
 $ab \rightarrow abcd$
fA4: $bd \rightarrow f$ (2)
 $abcd \rightarrow abcdf$ \downarrow

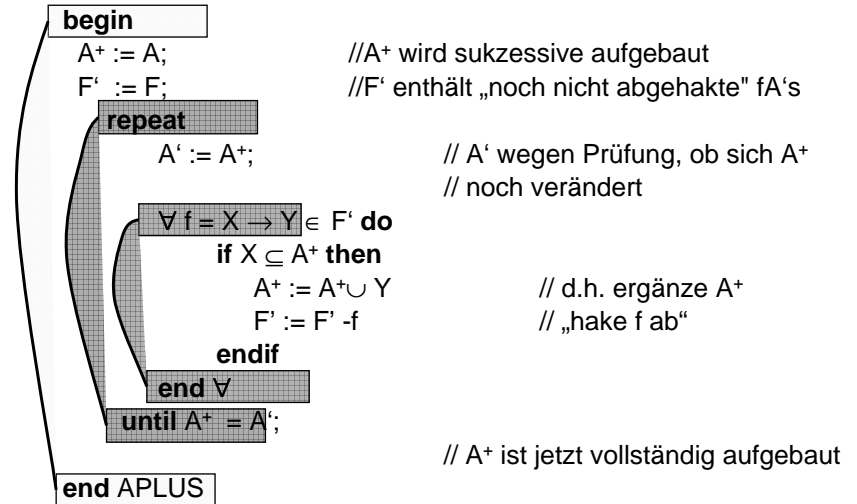
$ab \rightarrow abcdf$ \downarrow (3)

fA 2, 3, 5, 7, 8 bringen nicht mehr dazu: $(ab)^+ = abcdf$

5.4.3 Die abgeschlossene Hülle F^+

(6|10)

Algorithmus APLUS zur Berechnung von A^+



5.4.3 Die abgeschlossene Hülle F^+

(7|10)

Wozu braucht man A^+ ?

Antwort: Bestimmung aller Schlüssel

Vor.: $r: (U \mid F), F \subseteq \Phi(U)$

Definition: (Wiederholung)

$K \subseteq U$ ist „**Schlüssel**“ für r

$:\Leftrightarrow$

(1) K ist identifizierend: $K \rightarrow U(r)$, d.h. $K \rightarrow U \in F^+ \quad \underline{(K^+ = U)}$

(2) K ist minimal mit „(1)“:

$A \subset K \Rightarrow A \not\rightarrow U(r)$, d.h. $A \rightarrow U \notin F^+$

\Leftrightarrow

$K^+ = U \wedge (A \subset K \Rightarrow A^+ \neq U)$

5.4.3 Die abgeschlossene Hülle F^+

(8|10)

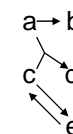
Algorithmus zur **Bestimmung aller Schlüssel** von r :

- bestimme alle notwendigen A^+ gemäß Algorithmus
- wähle alle minimalen K mit $K^+ = U$ aus.

Beispiel:

$r1: (abcde \mid F1)$

$F1 = \{a \rightarrow b, ac \rightarrow d, c \rightarrow e, e \rightarrow c\}$



Schlüssel: ac; ae
 NSA: b,d

5.4.3 Die abgeschlossene Hülle F^+

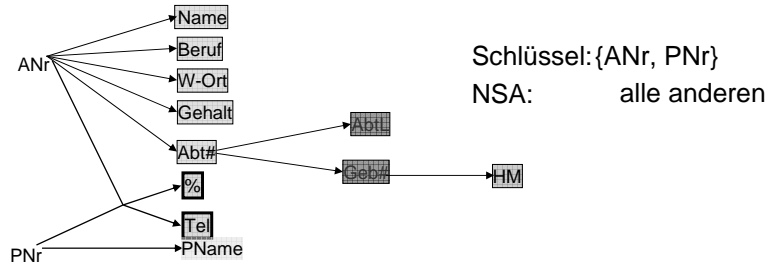
(9|10)

Beispiel

Relation angest: (U|F)

$U = \{ \text{ANr, Name, Beruf, W-Ort, Gehalt, Abt\#, AbtL, Geb\#, HM, PNr, PName, \%, TelNr} \}$

$F = \{ \text{ANr} \rightarrow \text{Name Beruf Abt\# W-Ort Gehalt}; \text{Abt\#} \rightarrow \text{AbtL Geb\#}; \text{Geb\#} \rightarrow \text{HM}; \text{PNr} \rightarrow \text{PName}; \text{ANr PNr} \rightarrow \text{\% Tel} \}$



Man beachte: alle Anomalien!

5.4.3 Die abgeschlossene Hülle F^+

(10|10)

Anmerkung:

Das Problem der *Bestimmung eines Schlüssels minimaler Kardinalität* ist vermutlich nicht effizient lösbar:

Satz

Sei F eine Menge von fAs über U und k eine positive ganze Zahl. Dann ist das Problem zu entscheiden, ob es einen Schlüssel $K \subseteq U$ gibt mit $|K| \leq k$, NP-vollständig.

(Beweis: vgl. G. Vossen, Datenmodelle, Datenbanksprachen, Datenbankmanagementsysteme, Oldenbourg, 2000, S. 163)

5.4.4 Partielle und volle funktionale Abhängigkeit

(1|2)

Geg.: $r: (U | F)$; $F \subseteq \Phi(U)$ $A, B \subseteq U$

nicht-triviale funktionale Abhängigkeit

$A \rightarrow B$ ($\in F^+$), mit ($B \not\subseteq A$)

Definition:

B ist **voll funktional** abhängig von A (in r), in Zeichen $A \twoheadrightarrow B$ wenn gilt:

(v1) $A \rightarrow B \in F^+$,

(v2) für alle $A' \subsetneq A$: $A' \rightarrow B \notin F^+$

(d.h. es gibt keine Teilmenge von A , von der B funktional abhängt)

Definition:

B ist **partiell** von A (funktional) abhängig, wenn gilt:

es gibt $A' \subsetneq A$: $A' \rightarrow B \in F^+$

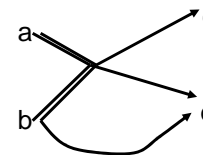
d.h. $A \supsetneq A' \rightarrow B$

5.4.4 Partielle und volle funktionale Abhängigkeit

(2|2)

Beispiel: $U = \{a, b, c, d\}$

$F = \{ \text{ab} \rightarrow \text{c}, \text{ab} \rightarrow \text{d}, \text{b} \rightarrow \text{d} \}$



$ab \twoheadrightarrow c$

$ab \rightarrow d$, aber $ab \not\rightarrow d$:

$ab \not\rightarrow b \rightarrow d$

5.4.5 Zweite Normalform (2NF)

(1|3)

$r: (U | F), F \subseteq \Phi(U); A, B, \dots, K \subseteq U; a, b, \dots \in U$

Eine Relation ist in der **Ersten Normalform (1NF)**, wenn die Attributwerte atomar sind.

Definition:

Eine 1NF-Relation r ist in der **Zweiten Normalform (2NF)**, wenn jedes Nichtschlüsselattribut von jedem Schlüssel voll funktional abhängig ist; d.h.:

r in 2NF $\Leftrightarrow \forall \text{NSA } a \forall \text{Schl. } K: K \xrightarrow{\bullet} a (r)$

r ist **nicht** in der 2NF, wenn (irgend)ein NSA von (irgend)einem Schlüssel partiell abhängig ist; d.h.:

r nicht in 2NF $\Leftrightarrow \exists \text{NSA } a \exists \text{Schl. } K: \neg(K \xrightarrow{\bullet} a)$
 $\Leftrightarrow \exists \text{NSA } a \exists \text{Schl. } K: (\exists A, A \subset K: A \rightarrow a) \neq$

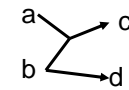
109

5.4.5 Zweite Normalform (2NF)

(2|3)

Beispiel:

(1) $r: (U | F); U = \{a,b,c,d\},$
 $F = \{ab \rightarrow c, b \rightarrow d\}$

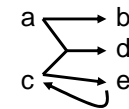


Schl.: **ab**

NSA: **c,d**

nicht in 2NF, weil d partiell vom Schlüssel ab abhängig.

(2) $r_1: (U | F); U = \{a,b,c,d,e\}$
 $F = \{a \rightarrow b, ac \rightarrow d, c \rightarrow e, e \rightarrow c\}$



Schl.: **ac; ae**

NSA: **b,d**

nicht in 2NF nur d ist voll funktional von Schlüssel ac bzw. ae abhängig

Allg.: Falls: $A \rightarrow b$ mit b NSA, $A \subsetneq$ Schlüssel K , dann: **nicht 2NF**.

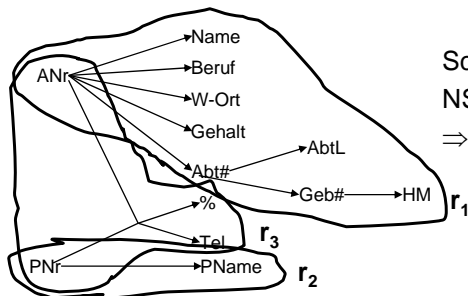
110

5.4.5 Zweite Normalform (2NF)

(3|3)

Beispiel: Relation angest: (U|F)

$U = \{ANr, Name, Beruf, W\text{-Ort}, Gehalt, Abt\#, AbtL, Geb\#, HM, PNr, PName, \%, TelNr\}$
 $F = \{ANr \rightarrow Name \text{ Beruf } Abt\# \text{ W-Ort } Gehalt; Abt\# \rightarrow AbtL \text{ Geb\#}; Geb\# \rightarrow HM;$
 $PNr \rightarrow Pname; ANr \text{ PNr} \rightarrow \% \text{ TelNr}\}$



Schlüssel: **{ANr, PNr}**

NSA: **alle anderen**

\Rightarrow Nicht in 2NF / alle Anomalien

Abhilfe 1: Zerlegung Relation angest: (U|F)
 in 3 Relationen r_1, r_2, r_3 (s.o.).

Ergebnis: r_1, r_2, r_3 in 2NF;
 trotzdem: alle Anomalien (in r_1), Grund?

111

5.4.6 Transitiv (funktionale) Abhängigkeiten

(1|3)

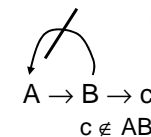
Geg.: $r: (U | F); A, B \subseteq U; c \in U$

Das Attribut c ist **transitiv abhängig** von A , wenn es ein $B \subseteq U$ gibt, so dass folgende Bedingungen erfüllt sind:

(t1) $A \rightarrow B \rightarrow c$

(t2) $B \not\rightarrow A$ verhindert, daß $A = B$ bzw. A äquivalent B

(t3) $c \notin AB$ verhindert, daß c trivialerweise (Reflexivität) von A oder B abhängt.



wir schreiben: $A \mapsto c$

c transitiv funktional abhängig von A

112

5.4.6 Transitiv (funktionale) Abhängigkeiten

(2|3)

Voraussetzungen wie oben: $r:(U,F), \dots$

Bezeichnung:

Wir nennen c **direkt (funktional) abhängig** von A , wenn

(d1) c funktional, aber

(d2) nicht transitiv funktional abhängig ist von A ;

d.h.:

(d1) $A \rightarrow c(r)$

(d2) $A \rightarrow B \rightarrow c(r) \Rightarrow (B \rightarrow A(r) \text{ oder } c \in AB)$

Wir schreiben: $A \bullet \rightarrow c(r)$

Folgerung / Lemma 1.4:

Für jeden Schlüssel K in r und jedes Attribut $c \notin K$ gilt:

$$K \bullet \rightarrow c(r) \Rightarrow K \bullet \rightarrow c(r)$$

113

5.4.6 Transitiv (funktionale) Abhängigkeiten

(3|3)

Beweis:

Annahme: $\neg (K \bullet \rightarrow c(r))$

Dann gibt es B mit:

$$B \subsetneq K \text{ (somit } K \rightarrow B(r) \text{) und } B \rightarrow c(r)$$

Da K Schlüssel, gilt $B \not\rightarrow K(r)$.

Somit:

$$\begin{array}{c} \curvearrowright \\ K \rightarrow B \rightarrow c \\ c \notin BK (= K) \end{array}$$

d.h. aber $K \not\rightarrow c(r), \neg (K \bullet \rightarrow c(r))$

114

5.4.7 Dritte Normalform (3NF)

(1|5)

Definition: Eine 1NF-Relation ist in der **dritten Normalform (3NF)**, wenn **kein Nichtschlüsselattribut** von (irgend)einem **Schlüssel transitiv** abhängig ist; m.a.W.:

wenn **jedes NSA** von **jedem Schlüssel** **direkt** abhängig ist; d.h.:

$$r \text{ in 3NF} \Leftrightarrow \forall \text{ NSA } a \forall \text{ Schl. } K: \neg (K \rightarrow a(r))$$

$$\Leftrightarrow \forall \text{ NSA } a \forall \text{ Schl. } K: K \bullet \rightarrow a(r)$$

Somit gilt entsprechend:

$$r \text{ nicht in 3NF} \Leftrightarrow \exists \text{ NSA } a \exists \text{ Schl. } K: K \rightarrow a(r)$$

115

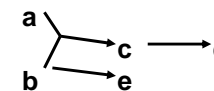
5.4.7 Dritte Normalform (3NF)

(2|5)

Beispiel:

$s:(U | F)$

$U = \{a, b, c, d, e\}; F = \{ab \rightarrow c, c \rightarrow d, b \rightarrow e\}$



Schlüssel: ab

NSA: c, d, e

$ab \rightarrow c \rightarrow d, ab \rightarrow b \rightarrow e$

d und e sind transitiv Abh. vom Schlüssel

\Rightarrow **nicht 3NF**

$ab \supset b \rightarrow e \Rightarrow$ **nicht 2NF**

Bemerkung:

Allgemein gilt:

r nicht in 2NF $\Rightarrow r$ nicht in 3NF

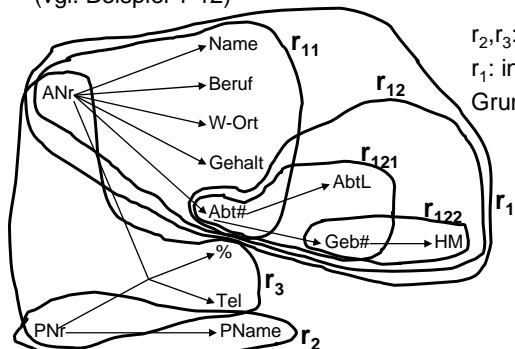
bzw. r in 3NF $\Rightarrow r$ in 2NF

116

5.4.7 Dritte Normalform (3NF)

(3|5)

Beispiel1-14: (Angestellten-Beispiel 1-3) Relation angest: (U|F)
(vgl. Beispiel 1-12)



r_2, r_3 : in 3NF.
 r_1 : in 2NF, nicht in 3NF / Anomalien
Grund: (Schl.) $ANr \twoheadrightarrow AbtL$ (NSA)

Abhilfe: Zerlegung r_1 in r_{11}, r_{12} :

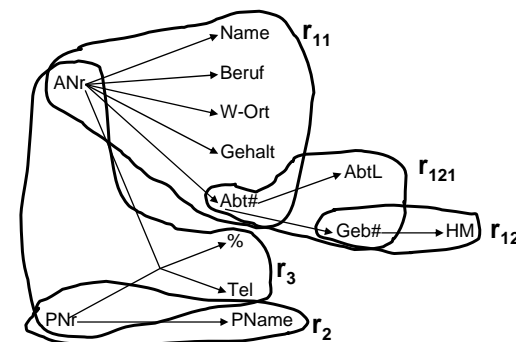
r_{11} in 3NF;
 r_{12} in 2NF, nicht in 3NF / Anomalien!
Grund: (Schl.) $Abt\# \twoheadrightarrow HM$ (NSA);

Abhilfe: Zerlegung r_{12} in r_{121}, r_{122} : beide in 3NF

5.4.7 Dritte Normalform (3NF)

(4|5)

Ergebnis:



angest: (U|F) wird zerlegt in 5 Relationen:
 $r_{11}, r_{121}, r_{122}, r_2, r_3$ (alle sind in 3NF)!

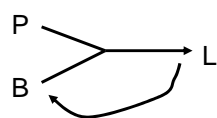
5.4.7 Dritte Normalform (3NF)

(5|5)

Beispiel: („Lieferbeziehung“):

Projekte (P) — Bauteile (B) — Lieferanten (L)

lief: LIEF(PBL | PB → L, L → B)



Schlüssel: PB; PL
NSA: \emptyset
 \Rightarrow 2NF, 3NF

Schl. $PB \rightarrow L \rightarrow B$; $B \in PBL$

aber

Schl. $PL \rightarrow L \rightarrow B$; $B \notin PL$ **B kein NSA**

Trotzdem (obwohl 3NF): **Anomalien!**

5.4.7 BCNF – Boyce-Codd-Normalform

(1|2)

r in 3NF $\Leftrightarrow \forall$ NSA $a \forall$ Schl. $K: K \twoheadrightarrow a$ (r)

Definition: Eine einfache fA $A \rightarrow b$ ($\in F^+$) heißt *elementar*
 \Leftrightarrow fA ist nicht-trivial (d.h. $b \notin A$)
und voll funktional abhängig ($A \twoheadrightarrow b$)

Umformulierung mittels „elementarer fA’s“:

r in 3NF \Leftrightarrow Für jede elementare fA $A \rightarrow b$ gilt:
entw. A ist Schlüssel
oder b ist Schlüsselattribut

5.4.7 BCNF – Boyce-Codd-Normalform

(2|2)

Wdh.:

r in 3NF \Leftrightarrow Für jede elementare fA $A \rightarrow b$ gilt:
 entw. A ist Schlüssel
 oder b ist Schlüsselattribut

r ist in BCNF (Boyce-Codd-Normalform)

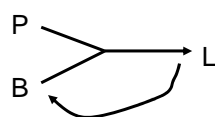
\Leftrightarrow Für jede elementare fA $A \rightarrow b$ gilt:
 A ist Schlüssel.

d.h. **alle** elementaren fA's gehen von Schlüsseln aus!

Folgerung: r in BCNF \Rightarrow r in 3NF

Aber: Umkehrung gilt i.A. nicht!

\Rightarrow **Verschärfung der 3NF**



Schlüssel: PB; PL
 NSA: \emptyset
 \Rightarrow **2NF, 3NF**

5.4.8 Ausblick

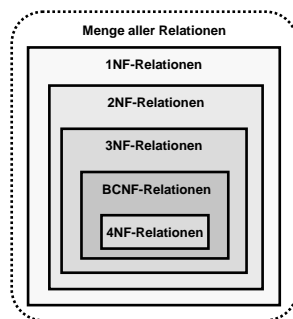
(1|2)

- Es gibt noch andere Arten von Abhängigkeiten, z.B.
 - mehrwertige Abhängigkeiten („multivalued dependencies“)
 - „join dependencies“ (JD)
- Einige weitere Normalformen
 - 4NF
(Relation in BCNF hat außer fA's keine mehrwertigen Abhängigkeiten)
 - 5NF/ Project-Join Normal Form
(Jede JD einer Relation r wird allein von den Schlüsselabhängigkeiten von r impliziert)
 - Elementary Key Normal Form (EKNF)
 - Domain-Key Normal Form (DKNF)
 - Object Normal Form (ONF)
- prinzipiell: weniger Redundanz/Anomalien vs. Anzahl Relationen

(Details: s. angegebene Literatur)

5.4.8 Ausblick

(2|2)



← Praxis

↓
 Verlustfreie Zerlegung möglich

- **1NF:** Atomarität der Attribute
- **2NF:** jedes Nichtschlüsselattribut von jedem Schlüssel voll funktional abhängig
- **3NF:** Verbietet zusätzlich fA's in Menge der NSA
- **BCNF:** Für jede elementare fA $A \rightarrow b$ gilt:
 A ist Schlüssel
- **4NF:** Verbietet zusätzlich mehrwertige Abhängigkeiten

Literatur

- G. Schlageter, W. Stucky. **Datenbanksysteme: Konzepte und Modelle.** B. G. Teubner, Stuttgart 1983 (2. Auflage)
- R. Elmasri, S. B. Navathe. **Fundamentals of Database Systems.** Pearson Education, 4. Aufl., 2004, ISBN 0321204484
- H. Garcia-Molina, J.D. Ullman, J. Widom. **Database Systems. The Complete Book.** Prentice Hall, 2002, ISBN 0130319953
- G. Vossen. **Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme.** Oldenbourg, München, 2000, ISBN: 3486253395.
- S. M. Lang, P. C. Lockemann. **Datenbankeinsatz.** Springer, 1995, ISBN: 3-540-58558-3

Web

- Interaktive SQL-Schnittstelle
<http://www.db.fmi.uni-passau.de/local/db2sql/>