

Grundlagen von Anfragen und Änderungen

- ▣▣▣▣➤ Kriterien für Anfragesprachen
- ▣▣▣▣➤ Anfragealgebren
- ▣▣▣▣➤ Anfrage-Kalküle
- ▣▣▣▣➤ Änderungsoperationen

Einführung

- bisher \rightsquigarrow Relationenschemata mit Basisrelationen, die in der Datenbank gespeichert sind
- jetzt \rightsquigarrow “Abgeleitete” Relationenschemata mit virtuellen Relationen, die aus den Basisrelationen berechnet werden (Basisrelationen bleiben unverändert)

Einführung II

- *Anfrage*: Folge von Operationen, die aus den Basisrelationen eine Ergebnisrelation berechnet. Ergebnisrelation
 - ◆ interaktiv auf dem Bildschirm anzeigen,
 - ◆ per Programm weiterverarbeiten (“Einbettung”)
- *Sicht*: Folge von Operationen, die unter einem Sichtnamen langfristig abgespeichert wird und unter diesem Namen wieder aufgerufen werden kann; ergibt eine Sichtrelation
- *Snapshot*: Ergebnisrelation einer Anfrage, die unter einem Snapshot-Namen abgelegt wird, aber nie ein zweites Mal (mit geänderten Basisrelationen) berechnet wird (etwa Jahresbilanzen)

Kriterien für Anfragesprachen

- **Ad-Hoc-Formulierung:** Der Benutzer soll eine Anfrage formulieren können, ohne ein vollständiges Programm schreiben zu müssen.
- **Deskriptivität:** Der Benutzer soll formulieren “Was will ich haben?” und nicht “Wie komme ich an das, was ich haben will?”.
- **Mengenorientiertheit:** Jede Operation soll auf Mengen von Daten gleichzeitig arbeiten, nicht navigierend nur auf einzelnen Elementen (one-tuple-at-a-time).
- **Abgeschlossenheit:** Das Ergebnis ist wieder eine Relation und kann wieder als Eingabe für die nächste Anfrage verwendet werden.

Kriterien für Anfragesprachen II

- **Adäquatheit:** Alle Konstrukte des zugrundeliegenden Datenmodells werden unterstützt.
- **Orthogonalität:** Sprachkonstrukte sind in ähnlichen Situationen auch ähnlich anwendbar.
- **Optimierbarkeit:** Die Sprache besteht aus wenigen Operationen, für die es Optimierungsregeln (die nicht Gegenstand dieses Buches sind) gibt.
- **Effizienz:** Jede Operation ist effizient ausführbar (im Relationenmodell hat jede Operation eine Komplexität $\leq O(n^2)$, n Anzahl der Tupel einer Relation).

Kriterien für Anfragesprachen III

- **Sicherheit:** Keine Anfrage, die syntaktisch korrekt ist, darf in eine Endlosschleife geraten oder ein unendliches Ergebnis liefern.
- **Eingeschränktheit:** (folgt aus Sicherheit, Optimierbarkeit, Effizienz) Die Anfragesprache darf keine komplette Programmiersprache sein.
- **Vollständigkeit:** Die Sprache muß mindestens die Anfragen einer Standardsprache (wie etwa die in diesem Kapitel einzuführende Relationenalgebra oder den sicheren Relationenkalkül) ausdrücken können.

Anfragealgebren

- Mathematik: Algebra definiert durch Wertebereich und auf diesem definierte Operatoren
- für Datenbankabfragen: Inhalte der Datenbank sind Werte, und Operatoren definieren Funktionen zum Berechnen von Abfrageergebnissen
 - ◆ Relationenalgebra
 - ◆ NF²-Algebra
 - ◆ Andere Algebra-Erweiterungen

Relationenalgebra

- **Spalten ausblenden:** Projektion, Zeichen π ; in [. . .]: welche Spalten behalten; in (. . .): auf welche Relation anwenden
- **Zeilen heraussuchen:** Selektion, Zeichen σ ; in [. . .]: unter welchen Bedingungen; in (. . .): auf welche Relation anwenden
- **Tabellen verknüpfen:** Verbund (Join), Zeichen \bowtie ; Tupel über gleichbenannten Spalten und Werten aneinanderhängen
- **Tabellen vereinigen:** Vereinigung, Zeichen \cup ; Tupel aus beiden Relationen sammeln, doppelte herauswerfen
- **Tabellen voneinander abziehen:** Differenz, Zeichen $-$; Tupel aus der ersten Relation herausnehmen, falls sie auch in der zweiten Relation vorkommen
- **Spalten umbenennen:** Umbenennung, Zeichen β ; einen Attributnamen in einen anderen umbenennen (wichtig für \bowtie und \cup , $-$)

Laufendes Beispiel

Ausleih	Invnr	Name
	4711	Meyer
	1201	Schulz
	0007	Müller
	4712	Meyer

Buch	Invnr	Titel	ISBN	Autor
	0007	Dr. No	3-125	James Bond
	1201	Objektbanken	3-111	Heuer
	4711	Datenbanken	3-765	Vossen
	4712	Datenbanken	3-891	Ullman
	4717	Pascal	3-999	Wirth

Projektion

- Syntax

π [attributmeng] (relation)

- Semantik

$$\pi_X(r) := \{t(X) \mid t \in r\}$$

für $r(R)$ und $X \subseteq R$ Attributmeng in R .

Projektion II

- Beispiel 1: Projektion auf ein Attribut

π [Name] (Ausleih)

ergibt als Ergebnisrelation

Name
Meyer
Schulz
Müller

- Doppelte Ergebnistupel eliminiert

Projektion III

- Beispiel 2: Projektion auf Attributmenge

π [Invnr, ISBN] (Buch)

ergibt

Invnr	ISBN
0007	3-125
1201	3-111
4711	3-765
4712	3-891
4717	3-999

Projektion IV

- einfache Optimierungsregel: bei vielen Projektionen hintereinander reicht die zuletzt ausgeführte auch allein

π [Invnr] (π [Invnr, ISBN] (Buch))

ergibt optimiert

π [Invnr] (Buch)

Selektion

- Syntax

σ [bedingung] (relation)

- Semantik

$$\sigma_F(r) := \{t \mid t \in r \wedge F(t) = \mathbf{true}\}$$

Selektionsbedingungen

- *F* Konstanten-Selektion

Attribut θ Konstante

boolesches Prädikat θ ist $=$ oder \neq , bei linear geordneten Wertebereichen auch \leq , $<$, \geq oder $>$

- *F* Attribut-Selektion

Attribut1 θ Attribut2

- *F* logische Verknüpfung mehrerer Konstanten- oder Attribut-Selektionen mit \wedge , \vee oder \neg

Selektion II

■ Beispiel

σ [Name \leq 'N'] (Ausleih)

ergibt

Invnr	Name
4711	Meyer
0007	Müller
4712	Meyer

■ einfache Optimierungsregeln

- ◆ Selektionen lassen sich in der Reihenfolge beliebig vertauschen
- ◆ Manchmal lassen sich Projektion und Selektion vertauschen; Voraussetzung: Selektionsattribute kommen in Projektionsliste vor

Verbund

- Syntax des (natürlichen) Verbundes (engl.: natural join)

Relation1 \bowtie Relation2

- Semantik

$$r_1 \bowtie r_2 := \{t \mid t(R_1 \cup R_2) \wedge [\forall i \in \{1, 2\} \exists t_i \in r_i : t_i = t(R_i)]\}$$

- Verbund verknüpft Tabellen über gleichbenannten Spalten bei gleichen Attributwerten

Verbund: Beispiel

Ausleih \bowtie Buch

ergibt

Name	Invnr	Titel	ISBN	Autor
Müller	0007	Mr. No	3-125	James Bond
Schulz	1201	Objektbanken	3-111	Heuer
Meyer	4711	Datenbanken	3-765	Vossen
Meyer	4712	Datenbanken	3-891	Ullman

- nicht ausgeliehenes Pascal-Buch verschwindet: Tupel, die keinen Partner finden (dangling tuples), werden eliminiert
- später einführen: *outer join*, der “dangling tuples” übernimmt

Verbund II

π [Autor] (Buch) \bowtie π [Invnr] (Ausleih)

Autor	Invnr
James Bond	4711
James Bond	1201
James Bond	0007
James Bond	4712
Heuer	4711
Heuer	1201
Heuer	0007
Heuer	4712
Vossen	4711
...	...

entartet zu *kartesischem Produkt*

Eigenschaften Verbund

- aus $R_1 \cap R_2 = \{\}$ folgt $r_1 \bowtie r_2 = r_1 \times r_2$
- Projektion nicht inverse Operation zu Verbund: $\pi_{R_1}(r_1 \bowtie r_2) \subseteq r_1$
- Verbund nicht die inverse Operation zu zwei Projektionen (nur bei Verbundtreue)
- Verbund kommutativ: $r_1 \bowtie r_2 = r_2 \bowtie r_1$
- Verbund assoziativ: $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$
- Daher erlaubt:

$$\bowtie_{i=1}^p r_i$$

Mengenoperationen und Umbenennung

Buch1	<table border="1"><thead><tr><th>Autor1</th></tr></thead><tbody><tr><td>James Bond</td></tr><tr><td>Heuer</td></tr><tr><td>Vossen</td></tr><tr><td>Ullman</td></tr><tr><td>Wirth</td></tr></tbody></table>	Autor1	James Bond	Heuer	Vossen	Ullman	Wirth	Buch2	<table border="1"><thead><tr><th>Autor2</th></tr></thead><tbody><tr><td>Witt</td></tr><tr><td>Vossen</td></tr><tr><td>Silberschatz</td></tr><tr><td>Meier</td></tr><tr><td>Wirth</td></tr></tbody></table>	Autor2	Witt	Vossen	Silberschatz	Meier	Wirth
Autor1															
James Bond															
Heuer															
Vossen															
Ullman															
Wirth															
Autor2															
Witt															
Vossen															
Silberschatz															
Meier															
Wirth															

- *Umbenennung* β [neu \leftarrow alt] (relation)
ändert Attributnamen von alt in neu

β [Autor1 \leftarrow Autor2] (Buch2)

Durch Umbenennung nun Vereinigung, Differenz und Durchschnitt möglich

Mengenoperationen: Vereinigung

$\text{relation1} \cup \text{relation2}$

Beispiel: $\text{Buch1} \cup \beta [\text{Autor1} \leftarrow \text{Autor2}] (\text{Buch2})$

Autor1
James Bond
Heuer
Vossen
Ullman
Wirth
Witt
Silberschatz
Meier

Mengenoperationen: Differenz

relation1 – relation2

Beispiel:

Buch1 – β [Autor1 \leftarrow Autor2] (Buch2)

Autor1
James Bond
Heuer
Ullman

Mengenoperationen: Durchschnitt

$\text{relation1} \cap \text{relation2}$

Beispiel:

$\text{Buch1} \cap \beta [\text{Autor1} \leftarrow \text{Autor2}] (\text{Buch2})$

Autor1
Vossen
Wirth

Mengenoperationen, Umbenennung III

Umbenennung ermöglicht

- Verbunde, wo bisher kartesische Produkte ausgeführt wurden (unterschiedliche Attribute werden gleich benannt),
- kartesische Produkte, wo bisher Verbunde ausgeführt wurden (gleiche Attribute werden unterschiedlich genannt),
- Mengenoperationen

Mengenoperationen, Umbenennung III

Formal für $r_1(R)$ und $r_2(R)$

- Umbenennung $\beta_{B \leftarrow A}(r) := \{t' \mid \exists t \in r : t'(R - A) = t(R - A) \wedge t'(B) = t(A)\}$
- Vereinigung $r_1 \cup r_2 := \{t \mid t \in r_1 \vee t \in r_2\}$
- Durchschnitt $r_1 \cap r_2 := \{t \mid t \in r_1 \wedge t \in r_2\}$
- Differenz $r_1 - r_2 := \{t \mid t \in r_1 \wedge t \notin r_2\}$

Durchschnitt \cap wegen $r_1 \cap r_2 = r_1 - (r_1 - r_2)$ überflüssig

Unabhängigkeit und Vollständigkeit

- Minimale Relationenalgebra:

$$\Omega = \pi, \sigma, \bowtie, \beta, \cup \text{ und } -$$

- unabhängig: kein Operator kann weggelassen werden ohne Vollständigkeit zu verlieren
- andere unabhängige Menge: \bowtie und β durch \times ersetzen
- Relationale Vollständigkeit: jede andere Menge von Operationen genauso mächtig wie Ω
- Strenge relationale Vollständigkeit: zu jedem Ausdruck mit Operatoren aus Ω gibt es einen Ausdruck auch mit der anderen Menge von Operationen

Problem: Quantoren

- Allquantor in Relationenalgebra ausdrücken, obwohl in Selektionsbedingungen nicht erlaubt
- *Division* (kann aus Ω hergeleitet werden)
- $r_1(R_1)$ und $r_2(R_2)$ gegeben mit $R_2 \subseteq R_1$, $R' = R_1 - R_2$. Dann ist

$$\begin{aligned} r'(R') &= \{t \mid \forall t_2 \in r_2 \exists t_1 \in r_1 : t_1(R') = t \wedge t_1(R_2) = t_2\} \\ &=: r_1 \div r_2 \end{aligned}$$

- Division von r_1 durch r_2

$$r_1 \div r_2 = \pi_{R'}(r_1) - \pi_{R'}((\pi_{R'}(r_1) \bowtie r_2) - r_1)$$

Division: Beispiel

PILOT	FLUGZ.
Snoopy	707
Snoopy	727
Snoopy	747
Meyer	707
Meyer	727
Müller	707
Müller	727
Müller	747
Müller	777
Lüdenscheid	727

r_1

FLUGZ.
707
727
747

r_2

FLUGZ.
707
707

r_3

FLUGZ.
707

$r_1 \div r_2$ ergibt

r'

PILOT
Snoopy
Müller

$r_1 \div r_3$ ergibt

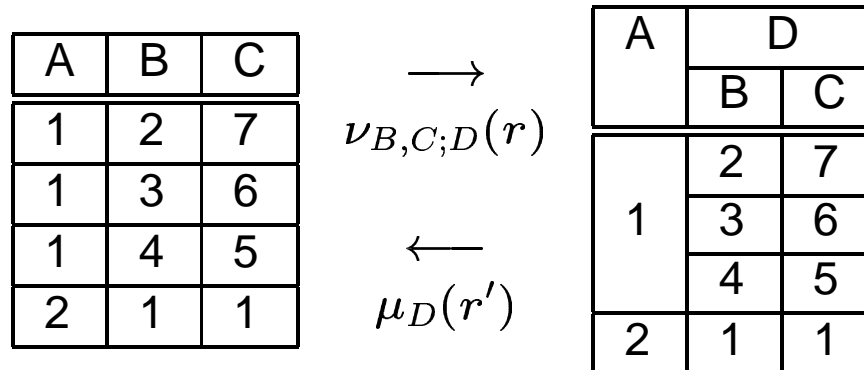
r'

PILOT
Snoopy
Meyer
Müller

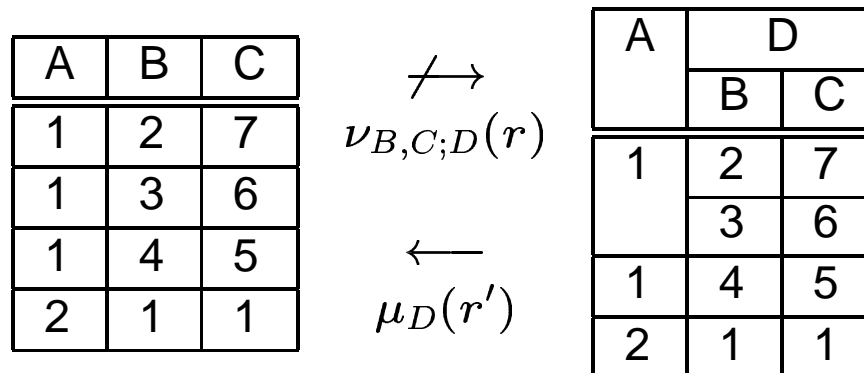
NF²-Algebra

- $\cup, -, \pi, \bowtie$ zunächst wie in Relationenalgebra
- σ -Bedingungen erweitern um:
 - ◆ Relationen als Operanden (statt nur Konstanten von Standard-Datentypen)
 - ◆ Mengenvergleiche, wie etwa $\theta: =, \subseteq, \subset, \supset, \supseteq$
- rekursiv aufgebaute Operationsparameter, etwa π und σ auch innerhalb von Projektionslisten und Selektionsbedingungen anwendbar
- zusätzliche Operationen ν (Nestung) und μ (Entnestung)

Nestung und Entnestung



Nestung nicht allgemein die Inverse der Entnestung:



Spezielle NF²-Algebren

- Minimale geschachtelte Algebra

nur ν und μ

- Orthogonale geschachtelte Algebra

Schek und Scholl: Selektion und Projektion rekursiv

- Algebren für PNF-Relationen

erhalten PNF-Eigenschaft

Andere Algebra-Erweiterungen

■ Nullwerte

r_1	<table border="1"><tr><td>A</td><td>B</td></tr><tr><td>a</td><td>b</td></tr><tr><td>a</td><td>∃</td></tr></table>	A	B	a	b	a	∃	r_2	<table border="1"><tr><td>B</td><td>C</td></tr><tr><td>b</td><td>c</td></tr><tr><td>∃</td><td>c</td></tr></table>	B	C	b	c	∃	c
A	B														
a	b														
a	∃														
B	C														
b	c														
∃	c														

∃: “Wert existiert, aber zur Zeit nicht bekannt”

Verbund

A	B	C
a	b	c

Andere Algebra-Erweiterungen II

- Nullwerte (fortg.) **definit-** und **maybe-**Markierungen

r_1	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>STATUS</th> </tr> </thead> <tbody> <tr> <td>a</td> <td>b</td> <td>d</td> </tr> <tr> <td>a</td> <td>∃</td> <td>d</td> </tr> </tbody> </table>	A	B	STATUS	a	b	d	a	∃	d
A	B	STATUS								
a	b	d								
a	∃	d								

r_2	<table border="1"> <thead> <tr> <th>B</th> <th>C</th> <th>STATUS</th> </tr> </thead> <tbody> <tr> <td>b</td> <td>c</td> <td>d</td> </tr> <tr> <td>∃</td> <td>c</td> <td>d</td> </tr> </tbody> </table>	B	C	STATUS	b	c	d	∃	c	d
B	C	STATUS								
b	c	d								
∃	c	d								

$r_1 \bowtie r_2$ liefert

A	B	C	STATUS
a	b	c	d
a	b	c	m
a	b	c	m
a	∃	c	m

- Operationen auf Wertebereichen (Spaltenerweiterung, abgeleitete Attribute)

Anfrage-Kalküle

- Ein Kalkül ist eine formale logische Sprache zur Formulierung von Aussagen.
- Ziel: Einsatz eines derartigen Kalküls zur Formulierung von Datenbank-Anfragen.
- Logikbasierter Ansatz:

Datenbankinhalte entsprechen Belegungen von Prädikaten einer Logik, Anfragen abgeleiteten Prädikaten.

Ein allgemeiner Kalkül

Motivation: mathematische Notation:

$$\{x^2 \mid x \in \mathbb{N} \wedge x^3 > 0 \wedge x^3 < 1000\}$$

Eine *Anfrage* hat die Form

$$\{f(\bar{x}) \mid p(\bar{x})\}$$

■ \bar{x} bezeichnet Menge von freien Variablen

$$\bar{x} = \{x_1 : D_1, \dots, x_n : D_n\}$$

Ein allgemeiner Kalkül II

- Funktion f bezeichnet Ergebnisfunktion über \bar{x}
wichtige Spezialfälle: Angabe einer Variable selber (f ist hier die Identitätsfunktion) und Tupelkonstruktion (Ergebnis vom Typ **tuple of**)
- p Selektionsprädikat über freien Variablen \bar{x}
 - ◆ Terme aus Variablen, Konstanten und Funktionsanwendungen
 - ◆ Prädikate der Datentypen, etwa $\leq, <, >, \geq, \dots$
→ atomare Formeln über Termen
 - ◆ Bezug zur aktuellen Datenbank → Datenbankprädikate, etwa Relationennamen im Relationenmodell
 - ◆ prädikatenlogischen Operatoren $\wedge, \vee, \neg, \forall, \exists$
→ Formeln

Ergebnisbestimmung einer Anfrage

$$\bar{x} = \{x_1 : D_1, \dots, x_n : D_n\}$$

1. Bestimme alle Belegungen der freien Variablen in \bar{x} , für die das Prädikat p wahr wird.
2. Wende Funktion f auf die durch diese Belegungen gegebenen Werte an.

Problem:

Unter welchen Umständen liefern Kalkülanfragen endliche Ergebnisse?

→ *Sicherheit von Anfragen*

Relationale Kalküle

- Der *Bereichskalkül* ist dadurch gekennzeichnet, daß Variablen Werte elementarer Datentypen (*Bereiche*) annehmen.
- Im *Tupelkalkül* hingegen variieren Variablen über Tupelwerte (entsprechend den Zeilen einer Relation).

Bereichskalkül

■ *Terme:*

- ◆ Konstanten, etwa 42 oder 'MZ-4'.
- ◆ Variablen zu Datentypen, etwa x .
Die Datentypangabe erfolgt in der Regel implizit und wird nicht explizit deklariert!
- ◆ Funktionsanwendung $f(t_1, \dots, t_n)$: Funktion f , Terme t_i , etwa *plus*(12, x) bzw. in Infixnotation $12 + x$.

■ *Atomare Formeln:*

- ◆ Prädikatanwendung $\Theta(t_1, \dots, t_n)$, $\Theta \in \{<, >, \leq, \geq, \neq, =, \dots\}$ Datentypprädikat, Terme t_i . Zweistellige Prädikate wie üblich in Infix-Notation.
Beispiele: $x = y$, $42 > x$ oder $3 + 7 = 11$.

Bereichskalkül II

■ *Atomare Formeln* (fortg.):

- ◆ Prädikatanwendungen für Datenbankprädikate, notiert als $R(t_1, \dots, t_n)$ für einen Relationennamen R .

Als Voraussetzung muß n die Stelligkeit der Relation R sein und alle t_i müssen vom passenden Typ sein.

Beispiel: `bestellt('Maier', x , 100)`

■ *Formeln* wie üblich mit \wedge , \vee , \neg , \forall und \exists .

■ *Anfragen*: $\{x_1, \dots, x_n \mid \phi(x_1, \dots, x_n)\}$

ϕ ist Formel über den in der Ergebnisliste aufgeführten Variablen x_1 bis x_n . Das Ergebnis ist eine Menge von Tupeln. Die Tupelkonstruktion erfolgt implizit aus den Werten der Variablen in der Ergebnisliste.

Basiskalkül für theoretische Untersuchungen

Einschränkung des Bereichskalküls:

- Als einziger Wertebereich sind die ganzen Zahlen erlaubt.
- Datentypprädikate werden wie bei der Relationenalgebra auf die Gleichheit und die elementaren Vergleichsoperatoren eingeschränkt.
- Funktionsanwendungen sind nicht erlaubt; nur Konstanten dürfen neben Bereichsvariablen als Terme verwendet werden.

Sichere Anfragen

- *Sichere Anfragen (auch semantisch sichere Anfragen):*

Anfragen, die für jeden Datenbankzustand $\sigma(\mathcal{R})$ ein endliches Ergebnis liefern.

- Beispiel für nicht sichere Anfrage:

$$\{x, y \mid \neg R(x, y)\}$$

Einfaches Beispiel für sichere Anfrage:

$$\{x, y \mid R(x, y)\}$$

Sichere Anfragen II

- Weiteres Beispiel für sichere Anfrage:

$$\{x, y \mid y = 10 \wedge x > 0 \wedge x < 10\}$$

Sicherheit folgt direkt aus den Regeln der Arithmetik.

Semantische Sicherheit ist im allgemeinen *nicht entscheidbar!*

Syntaktisch sichere Anfragen

- *Syntaktisch sichere Anfragen*: Anfragen, die syntaktischen Einschränkungen unterliegen, um die semantische Sicherheit zu erzwingen.
- Grundidee:

Jede freie Variable x_i muß überall in $\phi(x_1, \dots)$ durch positives Auftreten $x_i = t$ oder $R(\dots, x_i, \dots)$ an endliche Bereiche gebunden werden.
- Die Bindung an endliche Bereiche muß für die ganze Bedingung, also insbesondere für alle Zweige einer Disjunktion, gelten.

Relationen für Beispiele

Kunde(KName, Adresse, Konto),

bestellt(KName, WareBez, Anzahl)

Ware(WareBez, Preis, Vorrat)

Beispiele Bereichskalkül

“Alle (Namen von) Kunden in Magdeburg”.

$$\{x \mid \text{Kunde}(x, y, z) \wedge y = 'MD'\}$$

- Vereinfachte Notation: Ansonsten ungebundene Variablen (hier y und z) im Bedingungsteil existentiell mit \exists gebunden. Vollständige Version:

$$\{x \mid \exists y \exists z \text{Kunde}(x, y, z) \wedge y = 'MD'\}$$

- Einsparung von Bereichsvariablen, indem Konstanten als Parameter des Prädikats eingesetzt werden:

$$\{x \mid \text{Kunde}(x, 'MD', z)\}$$

Beispiele Bereichskalkül II

- Abkürzung für beliebige, unterschiedliche existentiell gebundene Variablen ist $_$ Symbol:

$$\{x \mid \text{Kunde}(x, y, _) \wedge y = 'MD'\}$$

- Verschiedene Auftreten des Symbols $_$ stehen hierbei für paarweise verschiedene Variablen.

Beispiele Bereichskalkül II

“Städte mit mehr als zwei Kunden”.

$$\{y \mid \text{Kunde}(x, y, z) \wedge \text{Kunde}(x', y, z') \wedge x \neq x'\}$$

Diese Anfrage zeigt eine Verbundbildung über das zweite Attribut der Kunde-Relation. Verbundbildung kann im Bereichskalkül einfach durch die Verwendung der selben Bereichsvariablen als Parameter in verschiedenen Relationsprädikaten erfolgen.

Beispiele Bereichskalkül III

“Wer hat Waren unter 1.- DM bestellt?”

$\{x, w \mid \text{Kunde}(x, y, z) \wedge \text{bestellt}(x, w, a) \wedge \text{Ware}(w, p, v) \wedge a > 0 \wedge p < 1.00\}$

Diese Anfrage zeigt einen Verbund über drei Relationen.

Beispiele Bereichskalkül IV

“Wer hat überhaupt etwas bestellt?”

$$\{x \mid \text{Kunde}(x, y, z) \wedge \exists w \exists a (\text{bestellt}(x, w, a) \wedge a > 0)\}$$

Hier wird eine existentiell gebundene Unteranfrage eingesetzt. Derartige Unteranfragen können aufgrund der Regeln der Prädikatenlogik wie folgt aufgelöst werden:

$$\{x \mid \text{Kunde}(x, y, z) \wedge (\text{bestellt}(x, w, a) \wedge a > 0)\}$$

Beispiele Bereichskalkül V

“Wer hat nur Waren in großer Anzahl bestellt?”

$$\{x \mid \text{Kunde}(x, y, z) \wedge \forall w \forall a (\text{bestellt}(x, w, a) \implies a > 100)\}$$

Im Gegensatz zu existentiell gebundenen Unteranfragen wie im vorigen Beispiel können universell gebundene Teilformeln nicht aufgelöst werden.

Ausdrucksfähigkeit Bereichskalkül

Der Bereichskalkül ist *streng relational vollständig*, d.h. zu jedem Term τ der Relationenalgebra gibt es einen äquivalenten (sicheren) Ausdruck η des Bereichskalküls.

Umsetzung von Relationenoperationen

Seien zwei Relationenschemata $R(A_1, \dots, A_n)$ und $S(B_1, \dots, B_m)$ gegeben.

- Vereinigung (für $n = m$).

$$R \cup S \hat{=} \{x_1 \dots x_n \mid R(x_1, \dots, x_n) \vee S(x_1, \dots, x_n)\}$$

- Differenz (für $n = m$).

$$R - S \hat{=} \{x_1 \dots x_n \mid R(x_1, \dots, x_n) \wedge \neg S(x_1, \dots, x_n)\}$$

- Natürlicher Verbund.

$$R \bowtie S \hat{=} \{x_1 \dots x_n x_{n+1} \dots x_{n+m-i} \mid R(x_1, \dots, x_n) \wedge S(x_1, \dots, x_i, x_{n+1}, \dots, x_{n+m-i})\}$$

Hierbei seien die ersten i Attribute von R und S die Verbundattribute, also $A_j = B_j$ für $j = 1 \dots i$.

Umsetzung von Relationenoperationen II

■ Projektion.

$$\pi_{\bar{A}}(R) \hat{=} \{y_1 \dots y_k \mid \exists x_1 \dots \exists x_n (R(x_1, \dots, x_n) \wedge y_1 = x_{i_1} \wedge \dots \wedge y_k = x_{i_k})\}$$

Hierbei ist die Attributliste der Projektion wie folgt gegeben: $\bar{A} = (A_{i_1}, \dots, A_{i_k})$

■ Selektion.

$$\sigma_{\phi}(R) \hat{=} \{x_1 \dots x_n \mid R(x_1, \dots, x_n) \wedge \phi'\}$$

Die Formel ϕ' wird hierbei aus ϕ gewonnen, indem Variable x_i an Stelle der Attributnamen A_i eingesetzt werden.

Tupelkalkül

- Anfragen im Tupelkalkül werden analog zu denen des Bereichskalküls aufgebaut, aber mit folgenden Unterschieden:

- ◆ Variablen sind tupelwertig.
- ◆ $R(t)$ bedeutet “ t ist in Relation R ”. Alternative Notation ist $t \in R$.
- ◆ $t.A_i$ bzw. $t[i]$ ermöglichen Zugriff auf die i -te Tupelkomponente. Eine alternative verbreitete Notation ist $t(i)$.
- ◆ Anfragen haben die Form

$$\{u \mid \phi(u)\}$$

wobei u eine Tupelvariable ist.

- Vereinfachte Notation: Beliebige Terme als Zielfunktion.

Der Basis-Tupelkalkül

$$\{t \mid \phi(t)\}$$

t *Tupelvariable*, $\phi(t)$ spezielle Formel der Prädikatenlogik 1. Ordnung

■ *Atome* des Basis-Tupelkalküls:

- ◆ $s \in r$ mit s *Tupelvariable* und r *Relation*.
- ◆ $s_1.A\theta s_2.B$ mit s_1, s_2 *Tupelvariablen* und A, B *Attribute*. s_1 muß über A und s_2 über B definiert sein.
 θ wie in *Relationenalgebra*: $=, \neq, <, >, \geq, \leq$.
- ◆ $s.A\theta c$ mit s *Tupelvariable*, s ist über A definiert, und $c \in \text{dom}(A)$ (c hat also einen passenden Typ).

Der Basis-Tupelkalkül II

■ *Formeln* des Basis-Tupelkalküls:

- ◆ Ein Atom ist eine Formel.
- ◆ Sind ϕ und ψ Formeln, dann sind auch $\neg\phi$, (ϕ) , $\phi \wedge G$ und $\phi \vee \psi$ Formeln.
- ◆ Ist $\phi(s)$ eine Formel und s eine freie (d.h. noch nicht durch einen Quantor gebundene) Tupelvariable in $\phi(s)$, so sind auch $\forall s : \phi(s)$ und $\exists s : \phi(s)$ Formeln.

Erweiterungen in der Notation

- Implizite Tupelbildung analog zum Bereichskalkül:

$$\{u.A_1, v.B_2 \mid R(u) \wedge S(v)\}$$

Anfrage ist äquivalent zu:

$$\{w \mid \exists u \exists v R(u) \wedge S(v) \wedge u.A_1 = w[1] \wedge v.B_2 = w[2]\}$$

- Definition von Attributnamen in der Zielliste und Verwendung von Termen in der Zielliste:

$$\{u.\text{Name}, \text{Jahresgehalt} : u.\text{Gehalt} * 13 \mid \text{Angestellte}(u)\}$$

Beispiele Tupelkalkül

- *Alle Kunden mit Namen 'Mueller'.*

$$\{t \mid \text{Kunde}(t) \wedge t.\text{KName} = \text{'Mueller'}\}$$

Hier werden alle Attribute von Kunden mit im Ergebnis ausgegeben.

- *Die Adressen von Kunden mit Namen 'Mueller'.*

$$\{t.\text{Adresse} \mid \text{Kunde}(t) \wedge t.\text{KName} = \text{'Mueller'}\}$$

- *Namen und Adressen von Kunden, die Papier bestellt haben.*

$$\{t.\text{KName}, t.\text{Adresse} \mid \text{Kunde}(t) \wedge \exists b(\text{bestellt}(b) \wedge b.\text{WName} = \text{'Papier'} \wedge t.\text{KName} = b.\text{KName})\}$$

EER-Kalkül

- Anfragen sind Multimengenanfragen: $\{\{\dots \mid \dots\}\}$
- Variablen werden positiv an *endliche Bereiche* gebunden. Bindung an Bereiche in *Deklarationen* δ_i .

$$\{\{t_1, \dots, t_n \mid \delta_1 \wedge \dots \wedge \delta_k \wedge \phi\}\}$$

Beispiel für Deklaration: $(a: \text{ANGESTELLTE})$

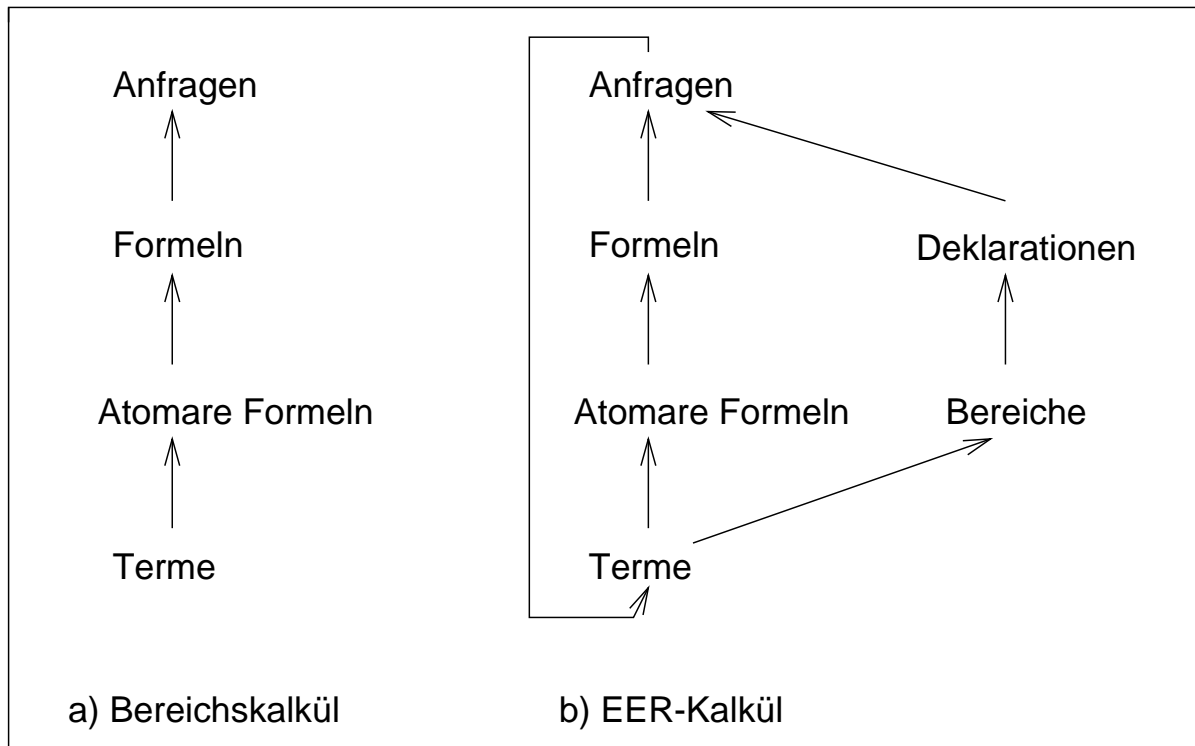
- Anfragen als Bereiche \rightarrow *rekursiver Anfrageaufbau*.
- *Aggregatfunktionen* wie Summenbildung, Durchschnittswerte, Zählen als Funktionen:

$$\text{AVG}(\{\{Preis(b) \mid (b: \text{BUCH}) \wedge \text{Verlag}(b) = \text{'Thomson'}\}\})$$

EER-Kalkül II

- Vereinigung, Schnittmenge etc. auf Anfragetermen
- Funktion **bts** (für *bag-to-set*): entfernt Duplikate aus Multimengen.
- Konzepte des EER-Modells → Konstrukte des EER-Kalküls:
 - ◆ Entity-Typen: Bereiche für Variablen.
 - ◆ Beziehungstypen: Prädikatsymbole oder Bereiche.
 - ◆ Attribute von Entity-Typen: Funktionen.

Formelaufbau in Bereichskalkül und EER-Kalkül



Änderungsoperationen

- Die Operation $u(d) := \text{insert } t \text{ into } r_i(\mathcal{R}_i)$ ist definiert durch

$$d \mapsto \begin{cases} d' := \{r_1, \dots, r_i \cup \{t\}, \dots, r_p\} & \text{falls } d' \in \mathbf{DAT}(\mathcal{S}) \\ d & \text{sonst} \end{cases}$$

- Die Operation $u(d) := \text{delete } t \text{ from } r_i(\mathcal{R}_i)$ ist definiert durch

$$d \mapsto \begin{cases} d' := \{r_1, \dots, r_i - \{t\}, \dots, r_p\} & \text{falls } d' \in \mathbf{DAT}(\mathcal{S}) \\ d & \text{sonst} \end{cases}$$

- Die Operation $u(d) := \text{replace } t \rightarrow t' \text{ in } r_i(\mathcal{R}_i)$ ist definiert durch

$$d \mapsto \begin{cases} d' := \{r_1, \dots, (r_i - \{t\}) \cup \{t'\}, \dots, r_p\} & \text{falls } d' \in \mathbf{DAT}(\mathcal{S}) \\ d & \text{sonst} \end{cases}$$