3 Relationale Anfragesprachen

3.1 Relationale Algebra

Grundoperationen (Gegeben $R(A_1, ..., A_n), S(B_1, ..., B_m)$)

- Vereinigung $R \cup S$

  Voraussetzung: R und S verträglich, d.h. $Rang(R) = Rang(S)$ und A_i und B_i haben gleiche Wertebereiche für alle i

  Ergebnisschema: $R'(A_1, ..., A_n)$

  Ergebnisrelation: $R' :=$ Mengenvereinigung von R und S

- Differenz $R - S$

  Vorausetzung: wie $R \cup S$

  Ergebnisschema: wie $R \cup S$

  Ergebnisrelation: $R' :=$ Mengendifferenz von R und S

- Kartesisches Produkt

  Vorausetzung: $\{A_1, ..., A_n\} \cap \{B_1, ..., B_m\} = \emptyset$

  Ergebnisschema: $R'(A_1, ..., A_n, B_1, ..., B_m)$

  Ergebnisrelation: $R' := \{r \circ s | r \in R \wedge s \in S\}$

  $r \circ s := (r_1, ..., r_n, s_1, ..., s_m)$ für $r = (r_1, ..., r_n) \in R$ und $s = (s_1, ..., s_m) \in S$

- Projektion $\pi_{\overline{A}}(R)$

  Vorausetzung: $\overline{A} = A_{i_1}, ..., A_{i_k}$ mit $1 \leq k \leq Rang(R) = n$ und $1 \leq i_1, ..., i_k \leq Rang(R) = n$

  Ergebnisschema: $R'(A_{i_1}, ..., A_{i_k})$

  Ergebnisrelation: $R' := \{\pi_{i_1,...,i_k}(r) | r \in R\}$

  $\pi_{i_1,...,i_k}(r) := (r_{i_1}, ..., r_{i_k})$ für $r = (r_1, ..., r_n)$

  Beispiel: Mit $R(A_1, A_2, A_3)$ ergibt $\pi_{A_3,A_1}(R)$ das Ergebnisschema $R'(A_3, A_1)$; es gilt: $\pi_{A_3,A_1}(R) = \pi_{A_{i_1},A_{i_2}}(R)$ mit $k = 2, i_1 = 3, i_2 = 1$

- Selektion $\sigma_\varphi(R)$

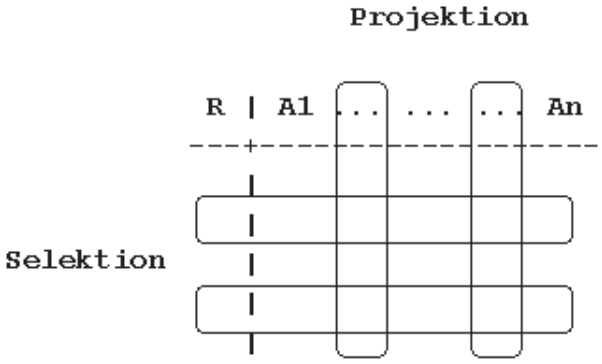
  Vorausetzung: φ ist atomare Formel $\Theta(u_1, ..., u_n)$; $u_1, ..., u_n$ sind Konstanten, Attributenamen oder Terme darüber; Θ ist boolesche Operation, wie Vergleiche $=, \neq, <, \leq, >, \geq$ oder andere boolesche Abfragen

  Beispiel: $(Gehalt * 1.04) + 120 > 4.000$, substring(Adresse,'Bremen')

  Ergebnisschema: $R'(A_1, ..., A_n)$

  Ergebnisrelation: $R' := \{r | r \in R \wedge \varphi$ ergibt nach Ersetzung der Attributnamen A_i durch Komponenten r_i den Wahrheitswert TRUE$\}$

  Beispiel: Mit $R(A, B)$ ergibt sich aus $\sigma_{B<b7}(R)$ als Ergebnis $\{r \mid r \in R \wedge r_2 < b7\}$



- Umbenennung $\delta_{C \leftarrow A_i}(R)$

  Voraussetzung: C Attributname mit $C \notin \{A_1, ..., A_n\}$

  Ergebnisschema: $R'(A_1, ..., A_{i-1}, C, A_{i+1}, ..., A_n)$

  Ergebnisrelation: $R' := R$

- Nachtrag zum Produkt:

  Falls $R \neq S$ und $\{A_1, ..., A_n\} \cap \{B_1, ..., B_m\} = \{C\}$ mit $A_i = B_j = C$ gilt, werden für $R \times S$ implizite Umbenennungen vorgenommen:

  $\delta_{R.C \leftarrow A_i}(R) \times \delta_{S.C \leftarrow B_j}(S)$

  Es entsteht das Ergebnisschema $R'(A_1, ..., R.C, ..., A_n, B_1, ..., S.C, ..., B_m)$

  Falls R=S gilt, werden für $R \times S = R \times R$ implizite Umbenennungen vorgenommen, so daß für $R(A, B, ...)$ folgendes Ergebnisschema entsteht:

  $R'(A_1, B_1, ..., A_2, B_2, ...)$

Beispiel

R	A	B
	a1	b1
	a1	b2
	a2	b1

S	A	C
	a2	b1
	a3	b3

- $R \cup S$

R'	A	B
	a1	b1
	a1	b2
	a2	b1
	a3	b3

- $R - S$

R'	A	B
	a1	b1
	a1	b2

- $R \times S$ (zu den Umbennungen bei der Diskussion von δ genaueres)

R'	R.A	B	S.A	C
	a1	b1	a2	b1
	a1	b2	a2	b1
	a2	b1	a2	b1
	a1	b1	a3	b3
	a1	b2	a3	b3
	a2	b1	a3	b3

- $\pi_A(R)$

R'	A
	a1
	a2

- $\sigma_{A<a2}(R)$

  [Es gilt die lexikographische Sortierung: $a1 < a2 < a3 < b1 < b2 < b3$]

R'	A	B
	a1	b1
	a1	b2

- $\delta_{B \leftarrow C}(S)$

S'	A	B
	a2	b1
	a3	b3

Beispielschema

- STUDENT(Matnr:int,SName:string,Fach:string,Sem:int)
- AUSLEIHE(Doknr:string,Matnr:int,Datum:string)
- BUCH(Doknr:string,Titel:string,Verlag:string,Ort:string,Jahr:int)
- AUTOREN(Doknr:string,AName:string)
- DESKRIPTOREN(Doknr:string,Schlagwort:string)

1. Namen der Studenten und Autoren

   $\pi_{AName}(AUTOREN) \cup \pi_{SName}(STUDENT)$
2. Namen der Autoren die keine Studenten sind

   $\pi_{AName}(AUTOREN) - \pi_{SName}(STUDENT)$
3. Welcher Student könnte welches Buch ausleihen?

   $STUDENT \times BUCH$
4. Welcher Student studiert im wievielten Semester?

   $\pi_{SName,Sem}(STUDENT)$

5. Bücher aus dem Jahr der Gutenberg-Bibel

   $\sigma_{Jahr=1455}(BUCH)$

6. Benenne Schlagwort in Stichwort um

   $\delta_{Stichwort \leftarrow Schlagwort}(DESKRIPTOREN)$

7. Verknüpfung von AUSLEIHE und BUCH über gemeinsame Dokumentnummern

   $\sigma_{AUSLEIHE.Doknr=BUCH.Doknr}(AUSLEIHE \times BUCH)$

   mit expliziter Umbenennung

   $\sigma_{AUSLEIHE.Doknr=BUCH.Doknr}(\delta_{AUSLEIHE.Doknr \leftarrow Doknr}(AUSLEIHE) \times \delta_{BUCH.Doknr \leftarrow Doknr}(BUCH))$

8. Erster Vergleich SQL vs RA

```
select Datum
from   STUDENT, AUSLEIHE
where  SName='Zimmermann' and STUDENT.Matnr=AUSLEIHE.Matnr
```

   $\pi_{Datum}(\sigma_{STUD.Matnr=AUS.Matnr}(\sigma_{SName='Zimmermann'}(STUD \times AUS)))$
   $\pi_{Datum}(\sigma_{SName='Zimmermann'}(\sigma_{STUD.Matnr=AUS.Matnr}(STUD \times AUS)))$
   $\pi_{Datum}(\sigma_{STUD.Matnr=AUS.Matnr}(\sigma_{SName='Zimmermann'}(STUD) \times AUS))$
   (Relationennamen abgekürzt)

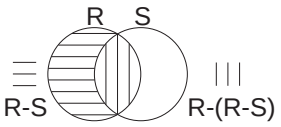
abgeleitete Operationen

- Durchschnitt $R \cap S$

  Voraussetzung: wie $R \cup S$

  Definition: $R \cap S :=$ Mengendurchschnitt von R und S

  Ableitung: $R \cap S = R - (R - S)$



- verallgemeinerte Selektion $\sigma_{\varphi}(R)$ (ÜBUNGSAUFGABE)

  Vorausetzung: φ logische Verküpfung von atomaren Formeln mittels $\wedge, \vee \neg$

  Definition: analog zur einfachen Selektion

- natürlicher Verbund $R * S$

  Voraussetzung: $C_1, ..., C_k$ seien alle gemeinsamen Attributnamen in R und S, also o.B.d.A. $R(A_1, ..., A_{n-k}, C_1, ..., C_k)$ und $S(B_1, ..., B_{m-k}, C_1, ..., C_k)$

  Definition: 'Verbinde R und S in natürlicher Weise über ihre gemeinsamen Attribute'

  Ableitung:
  $R * S = \pi_{A_1,...,A_{n-k},R.C_1,...,R.C_k,B_1,...,B_{m-k}}(\sigma_{R.C_1=S.C_1 \wedge ... \wedge R.C_k=S.C_k}(R \times S))$

- Division $R : S$

  Analogie zur Division auf den natürlichen Zahlen

  $n : m$ ist die größte natürliche Zahl mit $(n : m) * m \leq n$
  $R : S$ ist die größte Relation mit $(R : S) \times S \subseteq R$

  Voraussetzung: Attribute von S sind auch Attribute von R, also o.B.d.A.
  $R(A_1, ..., A_{n-m}, B_1, ..., B_m)$ und $S(B_1, ..., B_m)$

  Ergebnisschema: $R'(A_1, ..., A_{n-m})$

  Definition: $R : S := \{\pi_{A_1,...,A_{n-m}}(r) | r \in R \wedge \forall s \in S(\pi_{A_1,...,A_{n-m}}(r) \circ s \in R)\}$

  Ermittle alle $\pi_{A_1,...,A_{n-m}}$-Projektionen von Tupeln, die in den S-Komponenten alle in S aufgelisteten Wertkombinationen haben

  Ableitung: $R : S = \pi_{A_1,...,A_{n-m}}(R) - \pi_{A_1,...,A_{n-m}}((\pi_{A_1,...,A_{n-m}}(R) \times S) - R)$

U	A	B
4	a
4	b
5	b
5	c
6	a
6	b
6	c

V	B
b
c

U:V

U'	A
5
6

- R mod S

  Analogie zur Modulo-Operation auf den natürlichen Zahlen

  $n \ mod \ m$ ist die Differenz zwischen $(n : m) * m$ und n
  $R \ mod \ S$ ist die Differenz zwischen $(R : S) \times S$ und R

  Voraussetzung: wie bei der Division

  Ergebnisschema: $R'(A_1, ..., A_{n-m}, B_1, ..., B_m) = R'(A_1, ..., A_n)$

  Ableitung:

  $$\begin{aligned} R \ mod \ S &:= R - (R : S) \times S \\ &= R - (\pi_{A_1,...,A_{n-m}}(R) - \pi_{A_1,...,A_{n-m}}((\pi_{A_1,...,A_{n-m}}(R) \times S) - R)) \times S \end{aligned}$$

  Es gilt:

  $((n : m) * m) + (n \ mod \ m) = n \quad ((7 : 3) * 3) + (7 \ mod \ 3) = (2 * 3) + 1 = 6 + 1 = 7$
  $((R : S) \times S) \cup (R \ mod \ S) = R$

Beispiele (R und S wie oben)

R	A	B
	a1	b1
	a1	b2
	a2	b1

S	A	C
	a2	b1
	a3	b3

- $R \cap S$

R'	A	B
	a2	b1

- $\sigma_{A=a1 \vee B=b1}(R)$

R'	A	B
	a1	b1
	a1	b2
	a2	b1

- $R *_{B<C} S$

R'	R.A	B	S.A	C
	a1	b1	a3	b3
	a1	b2	a3	b3
	a2	b1	a3	b3

- $R * S$

R'	R.A	B	C
	a2	b1	b1

- $R : T = R : \pi_B(R)$

T	B
	b1
	b2

R'	A
	a1

Probe:

R' x T	A	B
	a1	b1
	a1	b2

$\subseteq$ R

- $R \; mod \; T$

R'	A	B
	a2	b1

1. Welche Informatik-Studenten haben eine Semesterzahl von mindestens 5?
   $\sigma_{Sem \geq 5 \wedge Fach='Informatik'}(STUDENT)$
2. Verbinde STUDENT und AUTOREN über die Attribute SName und AName
   $STUDENT *_{SName=AName} AUTOREN$

3. Titel der Ullman-Bücher

   $\pi_{Titel}(\sigma_{AName='Ullman'}(BUCH * AUTOREN))$

   $\pi_{Titel}(BUCH * (\sigma_{AName='Ullman'}(AUTOREN)))$ [VERSCHIEBUNG von σ]

   $\pi_{Titel}(\sigma_{AName='Ullman'}(\sigma_{BUCH.Doknr=AUTOREN.Doknr}(BUCH \times AUTOREN)))$

   Hier in der Notation eine implizite Umbenennung verwendet; eigentlich:

   $\pi_{Titel}(\sigma_{AName='Ullman'}(\sigma_{BUCH.Doknr=AUTOREN.Doknr}($
   $\delta_{BUCH.Doknr \leftarrow Doknr}(BUCH) \times \delta_{AUTOREN.Doknr \leftarrow Doknr}(AUTOREN))))$

4. Welche Mitarbeiter arbeiten in allen Projekten?

   klassisches Beispiel zur Division über dem Relationenschema
   JOB(Mitarb,Projekt)

   $JOB : \pi_{Projekt}(JOB) =$

   $\{\pi_{Mitarb}(job) | job \in JOB \wedge \forall projekt \in \pi_{Projekt}(JOB)[\pi_{Mitarb}(job) \circ projekt \in JOB]\} =$

   $$\pi_{Mitarb}(JOB) - \pi_{Mitarb}(\underbrace{\underbrace{[\pi_{Mitarb}(JOB) \times \pi_{Projekt}(JOB)] - JOB}_{\{(Mitarb,Projekt) | (Mitarb,Projekt) \notin JOB\}}}_{\{Mitarb | \exists Projekt (Mitarb,Projekt) \notin JOB\}})$$

   $JOB \ mod \ \pi_{Projekt}(JOB) =$ alle Mitarbeiter, die nicht in allen Projekten arbeiten

5. Bücher mit DB und PS (Programmiersprachen) als Schlagwort

   $BUCH * ($
   $\pi_{Doknr}(\sigma_{Schlagwort='DB'}(DESKRIPTOREN)) \cap$
   $\pi_{Doknr}(\sigma_{Schlagwort='PS'}(DESKRIPTOREN)))$

   { ('DB'), ('PS') } : S(Schlagwort) $\hat{=}$ konstante Relation über Schlagworten

   $BUCH * (DESKRIPTOREN : \{(`DB'), (`PS')\})$

6. Dokumentnummern der ältesten Bücher

   $\pi_{Doknr}(BUCH) - \pi_{Doknr}(\sigma_{Jahr_1 > Jahr_2}(\pi_{Doknr,Jahr}(BUCH) \times \pi_{Jahr}(BUCH)))$

   Rechts von $-$ werden die Dokumentnummern von Büchern berechnet, zu denen es ein Buch gibt das vor diesem Buch erschienen ist, d.h. Dokumentnummern von 'neueren' Büchern

   - DB-Zustand mit einem Dokument

BUCH	Doknr	Titel	Verlag	Ort	Jahr
	D1	...	...	...	1980

$\pi_{Doknr,Jahr}(BUCH) \times \pi_{Jahr}(BUCH)$	Doknr	$Jahr_1$	$Jahr_2$
	D1	1980	1980

   $\pi_{Doknr}(\sigma_{Jahr_1 > Jahr_2}(...)) = \emptyset$

   Ergebnis $\{D1\}$

   - DB-Zustand mit zwei Dokumenten

BUCH	Doknr	Titel	Verlag	Ort	Jahr
	D1	...	...	...	1980
	D2	...	...	...	1981

$\pi_{Doknr,Jahr}(BUCH) \times \pi_{Jahr}(BUCH)$	Doknr	Jahr_1	Jahr_2
	D1	1980	1980
	D1	1980	1981
	D2	1981	1980
	D2	1981	1981

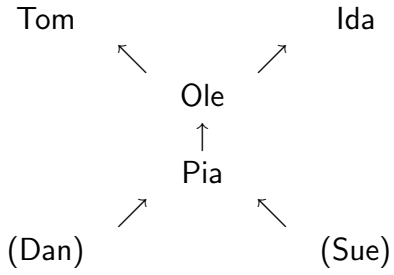
$\pi_{Doknr}(\sigma_{Jahr_1 > Jahr_2}(...)) = \{D2\}$
Ergebnis $\{D1\}$

7. Dokumentnummern der Bücher, die alle vorkommenden Schlagworte haben
   $DESKRIPTOREN : \pi_{Schlagwort}(DESKRIPTOREN)$
8. Welche Autoren liest Studi Zimmermann?
   $\pi_{AName}(\sigma_{SName=`Zimmermann'}((AUTOREN * AUSLEIHE) * STUDENT))$
   linker Verbund über Doknr; rechter Verbund über Matnr
   $\pi_{AName}((AUTOREN * AUSLEIHE) * \sigma_{SName=`Zimmermann'}(STUDENT))$

[weitere Beispiele]

- Beispiel: transitive Hülle

ELTERN	Eteil	Kind
	(Dan)	(Pia)
	(Sue)	(Pia)
	Pia	Ole
	Ole	Tom
	Ole	Ida



ELTERN* = VORFAHREN	Vorfahr	Nachkomme
	(Dan)	(Pia)
	(Sue)	(Pia)
	Pia	Ole
	Ole	Tom
	Ole	Ida
	———	———
	(Dan)	(Ole)
	(Dan)	(Tom)
	(Dan)	(Ida)
	(Sue)	(Ole)
	(Sue)	(Tom)
	(Sue)	(Ida)
	Pia	Tom
	Pia	Ida

Satz zur RA [besagt]: $\neg \exists$ Term $\tau \in$ RA mit $\tau(ELTERN) = VORFAHREN$ für alle DB-Zustände

[Es gibt allerdings Terme der RA zur Berechnung von R^k für jedes $k \in N$]

Aber: $ELTERN^2 =$
$ELTERN \cup \pi_{Eteil_1,Kind_2}(ELTERN *_{Kind_1=Eteil_2} ELTERN)$

$ELTERN^3 = ...$

[ÜBUNGSAUFGABE: Geben Sie einen Term der RA an, der $ELTERN^3$ korrekt berechnet, nicht aber $ELTERN^4$. Weiter anzugeben: (1) Zustand in dem es mindestens eine Urgrossmutter/Urgrossvater-Urenkel-Beziehung gibt (Kantenfolge der Länge 3), (2) Auswertung des Term dort, (3) Zustand in dem es mindestens eine Ururgrossmutter/Ururgrossvater-Ururenkel-Beziehung gibt (Kantenfolge der Länge 4), (4) Auswertung des Terms dort.]

- auch Integritätsbedingungen (IBen) mit RA formulierbar; IB = Gleichung oder Inklusion zwischen RA-Termen

  - STUD[ENT](Matnr:int,SName:string,Fach:string,Sem:int)
  - AUS[LEIHE](Doknr:string,Matnr:int,Datum:string)
  - BUCH(Doknr:string,Titel:string,Verlag:string,Ort:string,Jahr:int)
  - AUT[OREN](Doknr:string,AName:string)
  - DESK[RIPTOREN](Doknr:string,Schlagwort:string)

  Beispiel: Doknr aus AUS kommt in BUCH vor

  $\pi_{Doknr}(AUS) \subseteq \pi_{Doknr}(BUCH)$

  Beispiel: {Matnr} ist Schlüssel in STUD

  $\sigma_{Matnr_1=Matnr_2 \wedge (SName_1 \neq SName_2 \vee Fach_1 \neq Fach_2 \vee Sem_1 \neq Sem_2)}(STUD \times STUD) = \emptyset$

STUD	Matnr	SName	Fach	Sem
	1234	Ada	MI	5
	1234	Bob	MI	5

IB gilt hier nicht; durch Relationenmodell Zustand auch schon verboten

  Beispiel: maximal eine Ausleihe pro Doknr und Datum

  $\sigma_{Doknr_1=Doknr_2 \wedge Datum_1=Datum_2 \wedge Matnr_1 \neq Matnr_2}(AUS \times AUS) = \emptyset$

AUS	Doknr	MatNr	Datum
	D4711	M42	31.12.99
	D4711	M43	31.12.99

IB gilt hier nicht

  Beispiel: nur Piloten eingesetzt für Abflüge

  ANGESTELLTER(...,Personal#)

  PILOT(#Flugst,Personal#)

  eingesetzt_fuer(Datum,Flug#,Personal#)

  $\pi_{Personal\#}(eingesetzt\_fuer) \subseteq \pi_{Personal\#}(PILOT)$

3.2 Bereichskalkül

Motivation relationale Kalküle

- SQL-Anfragen

```
select T1, ..., Tn
from   R1, ..., Rk
where  F
```

- Formel F im where-Teil mit and, or, not und Unteranfragen: exists (select... from... where...); auch weitere Formen von Unteranfragen

  [F damit im Prinzip Formel des Tupelkalküls: $\vee$, $\wedge$, $\neg$, $\exists$, $\forall$ ($\equiv \neg\exists\neg$)]

Beispiel: Terme 123:int; 'Ullman':string; x:int; max(123+x,y):int

Beispiel: atomaren Formeln max(x,y) $<$ 100; DESK(d,'DB')

Beispiel: Formeln; Schemata R(A:int,B:int), S(C:int,B:int)

$R(4, 16) \wedge S(16, 18)$ [keine Variablen]

$\exists x(R(13, x) \wedge S(x, 14))$ [in Klammern: gebundene Vorkommen von x]

$R(4, x) \wedge \forall x(R(13, x) \Rightarrow z \leq x)$ Erinnerung: $(A \Rightarrow B)$ gdw $(\neg A \vee B)$

[1. Vorkommen von x frei, letztes und vorletztes gebunden; Vorkommen von z frei]

[Analogie: Gültigkeitsbereich von Variablen in Formeln VERSUS Verwendung von Variablen in Block-orientierten Programmiersprachen; Bindung einer Variablen mittels $\exists\ x\ (\varphi)$ oder $\forall\ x\ (\varphi)$ VERSUS Deklaration einer Variablen in einem Block]

Beispiel: Gültigkeit von Formeln

- [Der betrachtete Zustand σ ist durch folgende Tabellen festgelegt]

R	A:int	B:int	
	1	2	
	2	4	$\sigma(R)$
	3	6	

S	C:int	B:int	
	1	2	
			$\sigma(S)$
	4	4	

- [Es wird nun die Gültigkeit der folgenden Formeln φ_1 und φ_2 unter verschiedenen Belegungen β untersucht]

  $\varphi_1 \equiv R(x, y) \wedge \exists x(\ \underbrace{S(x, y) \wedge x - 1 < y}_{\varphi_2}\ ) \wedge z = \ x + y$

  [freie und gebundene Vorkommen der Variablen in obiger Formel diskutieren; alle Vorkommen von y,z frei; x frei und gebunden]

- $\beta_1 : x \mapsto 4, y \mapsto 4, z \mapsto 6$

  $\beta_2 : x \mapsto 2, y \mapsto 4, z \mapsto 6$

- Gilt $[\sigma, \beta_1] \models \varphi_2$? Antwort: ja

  Es gilt $[\sigma, \beta_1] \models S(x, y)$ und $[\sigma, \beta_1] \models (x - 1 < y)$

  $(x, y) = (4, 4) \in \sigma(S)$ und $x - 1 = 4 - 1 = 3 < 4 = y$

- Gilt $[\sigma, \beta_2] \models \varphi_1$? Antwort: ja

  $[\sigma, \beta_2] \models R(x, y)$ da $(2, 4) \in \sigma(R)$ und

  $[\sigma, \beta_2] \models \exists x(\varphi_2)$ und

  $[\sigma, \beta_2] \models (z = x + y)[\equiv (6 = 2 + 4)]$.

  [Man beachte, daß] $[\sigma, \beta_2] \models \exists x(\varphi_2)$ gilt, da $[\sigma, \beta_1] \models \varphi_2$ gilt

  [Für die Gültigkeit von φ_1, kann man sich also eine neue, quasi lokale Belegung der Variablen x suchen]

- Gilt $[\sigma, \beta_1] \models \varphi_1$? Antwort: nein

  Es gilt $(x, y) = (4, 4) \notin \sigma(R)$ oder $z = 6 \neq 8 = 4 + 4 = x + y$.

- Gilt $[\sigma, \beta_2] \models \varphi_2$? Antwort: nein

  Es gilt $(x, y) = (2, 4) \notin \sigma(S)$.

Beispiele: Anfragen

- Titel der Ullman-Bücher

  $\{t \mid \exists d, v, o, j, d2, an(BUCH(d, t, v, o, j) \wedge AUT(d2, an) \wedge$
  $d = d2 \wedge an = `Ullman')\}$

  eigentlich: $\{t \mid \exists d\ (\exists v\ (\exists o\ (\exists j\ (\exists d2\ (\exists an\ (...)))))))\}$

  $\{t \mid \exists d, v, o, j(BUCH(d, t, v, o, j) \wedge$
  $\exists d2, an(AUT(d2, an) \wedge d = d2 \wedge an = `Ullman'))\}$

  $\{t \mid \exists d, v, o, j(BUCH(d, t, v, o, j) \wedge AUT(d, `Ullman'))\}$

  $\{t : string \mid \exists d : string, v : string, o : string, j : int$
  $(BUCH(d, t, v, o, j) \wedge AUT(d, `Ullman'))\}$

- Bücher mit DB und PS als Schlagwort

  $\{d, t, v, o, j | BUCH(d, t, v, o, j) \wedge$
  $DESK(d, `DB') \wedge DESK(d, `PS')\}$

- Dokumentnummern der ältesten Bücher

  $\{d \mid \exists t, v, o, j(BUCH(d, t, v, o, j) \wedge$
  $\forall d2, t2, v2, o2, j2(BUCH(d2, t2, v2, o2, j2) \Rightarrow j \leq j2))\}$

  Was ist das Delta zu:

  $\{d \mid \exists t, v, o, j(BUCH(d, t, v, o, j) \wedge$
  $\forall d2, t2, v2, o2, j2(BUCH(d2, t2, v2, o2, j2) \wedge j \leq j2))\}$

  $\equiv$

$\{d \mid \exists t, v, o, j(BUCH(d, t, v, o, j) \wedge \forall j2(j \leq j2 \wedge ...))\}$

ergibt $\emptyset$

- Dokumentnummern der Bücher, die alle vorkommenden Schlagworte haben

$$\{d \mid \exists sw(\quad DESK(d, sw) \wedge \\ \forall sw2((\exists d2\ DESK(d2, sw2)) \Rightarrow DESK(d, sw2)))\}$$

- Welche Autoren liest Studi Zimmermann

$\{an \mid \exists d, m, dt, f, sm$
$(AUT(d, an) \wedge AUS(d, m, dt) \wedge STUD(m, `Zimmermann', f, sm))\}$

$\{an \mid \exists d(AUT(d, an) \wedge$
$\exists m, dt(AUS(d, m, dt)) \wedge$
$\exists f, sm(STUD(m, `Zimmermann', f, sm)))\}$

$\{an \mid \exists d(AUT(d, an) \wedge$
$\exists m(AUS(d, m, *) \wedge STUD(m, `Zimmermann', *, *)))\}$

*-Notation im BK: Jeder * entpricht einer neuen, existentiell quantifizierten Bereichsvariablen mit dem kleinstmöglichen Gültigkeitsbereich

[Obacht: *-Notation im BK damit unterschiedlich von *-Notation in SQL]

sichere Ausdrücke

- Beispiel für einen nicht-sicheren Ausdruck

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

  [Ergebnis der obigen Anfrage im allgemeinen unendlich; prinzipielles Problem: x,y kommen nicht in Relationen vor]

- [Idee: Jede] Ergebnisvariable x in nichtnegierter Form an einen Wert aus einer Relation oder an Konstante binden [und so endliches Ergebnis garantieren]

  $R_1(..., x, ...) \vee ... \vee R_p(..., x, ...) \vee x = c_1 \vee ... \vee x = c_k$

  mit $p, k \in N_0$ und $p + k \geq 1$

  [aus einer Bindung dieser Gestalt ergibt sich, daß das Ergebnis immer endlich ist]

- syntaktisches Kriterium für BK-Ausdrücke

$\{x_1, ..., x_n \mid$
$[R_1(..., x_1, ...) \vee ... \vee R_p(..., x_1, ...) \vee x_1 = c_1 \vee ... \vee x_1 = c_k] \wedge$
$...$ $\equiv$ SAFE
$[S_1(..., x_n, ...) \vee ... \vee S_m(..., x_n, ...) \vee x_n = d_1 \vee ... \vee x_n = d_q] \wedge$
$\varphi\}$

mit beliebigem φ

Sicherer Ausdruck: Ausdruck der Gestalt SAFE oder durch einfache Umformungen daraus herleitbarer Ausdruck

- R(A:int,B:int), S(C:int,B:int)

  $\{x \mid R(x,*) \vee R(*,x) \vee S(x,*) \vee S(*,x) \vee x = 42 \vee x = 43\}$ erfüllt Kriterium

  R(A:int)

  $\{x \mid R(x)\}$ erfüllt Kriterium

  $\{x \mid \exists y \exists z(x = y \wedge y = z \wedge R(z))\}$ ist daraus herleitbar

  Beispiel: nicht-sicherer Ausdruck kann komplexer sein als obiges Beispiel

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

  $B \equiv R(x,y)$

  $\{x, y \mid \neg B \wedge (A \vee \neg A)\}$

  $\{x, y \mid (\neg B \wedge A) \vee (\neg B \wedge \neg A)\}$

  $\{x, y \mid \neg\neg(\neg B \wedge A) \vee \neg\neg(\neg B \wedge \neg A)\}$

  $\{x, y \mid \neg(B \vee \neg A) \vee \neg(B \vee A)\}$

  $\{x, y \mid (B \vee \neg A) \Rightarrow \neg(B \vee A)\}$

  $\{x, y \mid (A \Rightarrow B) \Rightarrow \neg(B \vee A)\}$

  $A \equiv \exists z(R(x,z))$

  $\{x, y \mid (\exists z(R(x,z)) \Rightarrow R(x,y)) \Rightarrow \neg(R(x,y) \vee \exists w(R(x,w)))\}$

- alle obigen Anfragen an Bib-DB (die Grundformen ohne die Variationen) sind sicher

- Sicherheit nur hinreichendes Kriterium für endliches Ergebnis

  Beispiel: nicht-sicherer BK-Ausdruck, der stets endliches Ergebnis liefert zum Schema $R(A : int)$

  $\{x : int \mid \exists\ min : int\ \exists\ max : int\ (\ R(min) \wedge R(max) \wedge$
  $\forall\ z : int(R(z) \Rightarrow (min \leq z \wedge z \leq max)) \wedge$
  $min \leq x \wedge x \leq max\ )\}$

Darstellung von RA-Operationen im BK; gegeben $R(A_1, ..., A_n)$ und $S(B_1, ..., B_m)$

- $R \cup S = \{x_1, ..., x_n \mid R(x_1, ..., x_n) \vee S(x_1, ..., x_n)\}$
- $R - S = \{x_1, ..., x_n \mid R(x_1, ..., x_n) \wedge \neg S(x_1, ..., x_n)\}$
- $R \times S = \{x_1, ..., x_n, x_{n+1}, ..., x_{n+m} \mid R(x_1, ..., x_n) \wedge S(x_{n+1}, ..., x_{n+m})\}$
- $\pi_{\overline{A}}(R) = \{x_{i_1}, ..., x_{i_k} \mid \exists x_{i_{k+1}}, ..., \exists x_{i_n} R(x_1, ..., x_n)\}$ mit $\overline{A} = A_{i_1}, ..., A_{i_k}$ und $\{i_1, ..., i_k, i_{k+1}, ..., i_n\} = \{1, ..., n\}$

  Beispiel: Mit $R(A_1, A_2, A_3)$ wird $\pi_{A_3,A_2}(R)$ übersetzt nach $\{x_3, x_2 \mid \exists x_1 R(x_1, x_2, x_3)\}$ oder in *-Notation $\{x_3, x_2 \mid R(*, x_2, x_3)\}$

- $\sigma_\varphi(R) = \{x_1, ..., x_n \mid R(x_1, ..., x_n) \wedge \varphi'\}$

  mit $\varphi' \equiv \varphi$ mit Variablen x_i statt Attributnamen A_i

  Beispiel: $\sigma_{A_1=42}(R) = \{x_1, x_2, x_3 \mid R(x_1, x_2, x_3) \wedge x_1 = 42\}$

3.3 Tupelkalkül

Flüge von Ullman?

BUCHUNG(Name:string,Adr:string,Flug#:string,Datum:string)

- RA-Term

  $\pi_{Flug\#,Datum}(\sigma_{Name='Ullman'}(BUCHUNG))$

- BK-Ausdruck

  $\{f, d | \exists n, a(BUCHUNG(n, a, f, d) \wedge n = 'Ullman')\}$

  $\{f, d | BUCHUNG('Ullmann', *, f, d)\}$

- TK-Ausdruck

  $\{r : (Flug\#, Datum) \mid \exists b : (Name, Adr, Flug\#, Datum)(BUCHUNG(b) \wedge$
  $b.Name = 'Ullman' \wedge r.Flug\# = b.Flug\# \wedge r.Datum = b.Datum)\}$

  $\{r : (Flug\#, Datum) \mid \exists b : BUCHUNG$
  $(b.Name = 'Ullman' \wedge r.Flug\# = b.Flug\# \wedge r.Datum = b.Datum)\}$

Beispiel: Anfragen in TK

- Titel der Ullman-Bücher

  $\{r : (Titel) \mid \exists b : (Doknr, Titel, Verlag, Ort, Jahr), a : (Doknr, AName)$
  $(BUCH(b) \wedge AUT(a) \wedge$
  $r.Titel = b.Titel \wedge b.Doknr = a.Doknr \wedge a.AName = 'Ullman')\}$

  $\{r : (Titel) \mid \exists b : BUCH, a : AUT$
  $(r.Titel = b.Titel \wedge b.Doknr = a.Doknr \wedge a.AName = 'Ullman')\}$

- Beispiel zur Abkürzung $r : R$ mit $R(A : int, B : int)$

  - Beispielzustand

R	A	B
	1	2
	1	4

  - Für $\varphi \equiv (r.A = 1 \wedge 2 < r.B \wedge r.B < 4)$ gilt:

    $\exists\ r : (A : int, B : int)\ \varphi$ ist wahr

    $\exists\ r : R\ \varphi$ ist falsch

    $[\exists\ r : R\ \varphi] \equiv [\exists\ r : (A : int, B : int)(R(r) \wedge \varphi)]$

- Bücher mit Datenbanken (DB) und Programmierspachen (PS) als Schlagwort

  $\{r : (Titel) \mid \exists b : BUCH, d1 : DESK, d2 : DESK$
  $(r.Titel = b.Titel \wedge$
  $b.Doknr = d1.Doknr \wedge d1.Schlagwort = 'DB' \wedge$
  $b.Doknr = d2.Doknr \wedge d2.Schlagwort = 'PS')\}$

- Dokumentnummern der ältesten Bücher

  $\{r : (Doknr) \mid \exists b : BUCH(r.Doknr = b.Doknr \wedge$
  $\forall b' : BUCH(b.Jahr \leq b'.Jahr))\}$

- Dokumentnummern der Bücher die alle vorkommenden Schlagworte haben

  $\{r : (Doknr) \mid \exists d : DESK(r.Doknr = d.Doknr \wedge$
  $\forall d' : DESK(DESK(d.Doknr, d'.Schlagwort)))\}$

  ERWEITERUNG TK: Für $R(A_1, ..., A_n)$ gilt:

  $R(t_1, ..., t_n) :\Leftrightarrow \exists\ r : R\ (r.A_1 = t_1 \wedge ... \wedge r.A_n = t_n)$
  $\Leftrightarrow \exists\ r : (A_1, ..., A_n)\ (R(r) \wedge r.A_1 = t_1 \wedge ... \wedge r.A_n = t_n)$

- Welche Autoren liest Zimmermann?

  $\{r : (AName) \mid \exists aut : AUT, aus : AUS, s : STUD$
  $(r.AName = aut.AName \wedge aut.Doknr = aus.Doknr \wedge$
  $aus.Matnr = s.Matnr \wedge s.SName = `Zimmermann')\}$

  $\{r : (AName) \mid \exists aut : AUT(r.AName = aut.AName \wedge$
  $\exists aus : AUS(aut.Doknr = aus.Doknr \wedge$
  $\exists s : STUD(aus.Matnr = s.Matnr \wedge s.SName = `Zimmermann')))\}$

  - SQL-Formulierung der Beispielanfrage nach den Zimmermann-Autoren somit auch anders formulierbar
  - ```
    select AName
    from   AUT, AUS, STUD
    where  AUT.Doknr=AUS.Doknr and
           AUS.Matnr=STUD.Matnr and
           SName="Zimmermann"
    ```
  - mittels obiger Äquivalenzrechnung

    ```
    select AName
    from   AUT
    where  exists (select *
                   from   AUS, STUD
                   where  AUT.Doknr=AUS.Doknr and
                          AUS.Matnr=STUD.Matnr and
                          SName="Zimmermann")
    ```
  - oder sogar

    ```
    select AName
    from   AUT
    where  exists (select *
                   from   AUS
                   where  AUT.Doknr=AUS.Doknr and
                          exists (select *
                                  from   STUD
                                  where  AUS.Matnr=STUD.Matnr and
                                         SName="Zimmermann"))
    ```
    ```

Vergleich BK VERSUS TK [bzgl. Kriterium Bequemlichkeit]

- Vorteil TK: eine Tupelvariable statt mehrerer zugehöriger Bereichsvariablen
- Nachteil TK: Überlappungen von Tupelvariablen erfordern zusätzliche Konjunktionen von Komponentengleichungen
- Nachteil BK: In $R(x_1, ..., x_n)$ Position, nicht Name entscheidend
- Ausdrucksfähigkeit äquivalent

3.4 SQL

Beispielschema: KAL-Schema

KUNDE(KName, KAdr, Kto)

AUF[TRAG](KName, Ware, Menge)

LIEF[ERANT](LName, LAdr, Ware, Preis)

1. Welche Kunden (KName) haben ihr Konto überzogen?

   ```
   select KName
   from   KUNDE
   where  Kto<0
   ```

   äquivalenter Algebra-Term

   $\pi_{KName}(\sigma_{Kto<0}(KUNDE))$

   äquivalenter Tupelkalkül-Ausdruck

   $\{r : (KName) | \exists k : KUNDE(r.KName = k.KName \wedge k.Kto < 0)\}$

2. `select` τ `from` R_1, R_2 `where` $\varphi \equiv \pi_\tau(\sigma_\varphi(R_1 \times R_2))$

3. Welche Lieferanten (LName, LAdr) liefern Milch oder Mehl?

   SQL liefert im allgemeinen als Ergebnis keine Menge sondern eine Multimenge, d. h. eine Kollektion in der Elemente mehrmals auftreten können. Beispielsweise würde ein Lieferant der Milch und Mehl liefert im Ergebnis zweimal auftauchen. Diese Duplikate können mittels `select distinct` eliminiert werden.

   (select distinct TERM ...) : set(type)

   (select TERM ...) : bag(type)

   (...) union (...) : set(type)

   (...) union all (...) : bag(type)

   ```
   select distinct LName, LAdr
   from   LIEF
   where  Ware = 'Milch' or Ware = 'Mehl'
   ```

$\{r : (LName, LAdr) |$
$\exists l : LIEF(r.LName = l.LName \wedge r.LAdr = l.LAdr \wedge$
$(l.Ware = `Milch' \vee l.Ware = `Mehl'))\}$

äquivalent mit expliziter Tupelvariable

```
select L.LName, L.LAdr
from   LIEF L
where  L.Ware = 'Milch' or L.Ware = 'Mehl'
```

oder Relationenname als implizite Tupelvariable

```
select LIEF.LName, LIEF.LAdr
from   LIEF
where  LIEF.Ware = 'Milch' or LIEF.Ware = 'Mehl'
```

4. Welche Lieferanten (LName, Ware) aus Bremen liefern von Weiss in Auftrag gegebene Waren?

```
select LName, LIEF.Ware
from   LIEF, AUF
where  LAdr like '%Bremen%' and
       LIEF.Ware = AUF.Ware and
       KName = 'Weiss'
```

`like`: Substringsuche mit % $\widehat{=}$ Zeichenkette beliebiger Länge; _ $\widehat{=}$ Zeichenkette der Länge 1

SQL-Regeln für die Verwendung von Tupelvariablen (Qualifikationen)

- Tupelvariable können weggelassen werden solange ein Attribut eindeutig einer Relation bzw. einer impliziten Deklaration einer Tupelvariablen zugeordnet werden kann.
- Fehlt eine Tupelvariable, so wird angenommen, daß dort die Tupelvariable steht, die im logisch nächsten select-from-where-Block deklariert wurde.
- Der Relationenname kann als implizit deklarierte Tupelvariable verwendet werden.
- Tupelvariable müssen bei Mehrdeutigkeiten benutzt werden, falls z. B. ein Attribut in mehreren Relationen auftritt.
- Es gilt: implizite und explizite Tupelvariable können verwendet werden
- Es gilt: Tupelvariablen sind obligatorisch bei Mehrdeutigkeiten

ANFRAGE $\widehat{=}$

$\pi_{LName,LIEF.Ware}($
$\sigma_{substring(`Bremen',LAdr) \wedge LIEF.Ware=AUF.Ware \wedge KName=`Weiss'}($
$LIEF \times AUF))$

$\widehat{=}$

$\{r : (LName, Ware) | \exists l : LIEF, a : AUF$
$(r.LName = l.LName \wedge r.Ware = l.Ware \wedge$
$substring(`Bremen', l.LAdr) \wedge$
$l.Ware = a.Ware \wedge a.KName = `Weiss')\}$

äquivalent

```
select LName, Ware
from   LIEF
where  LAdr like '%Bremen%' and
       Ware = any (select Ware
                   from   AUF
                   where  KName = 'Weiss')
```

$\{r : (LName, Ware) | \exists l : LIEF$
$(r.LName = l.LName \wedge r.Ware = l.Ware \wedge$
$substring(`Bremen', l.LAdr) \wedge$
$\exists a : AUF(l.Ware = a.Ware \wedge a.KName = `Weiss'))\}$

äquivalent zu `= any` ist `in`

```
select LName, Ware
from   LIEF
where  LAdr like '%Bremen%' and
       Ware in (select Ware
                from   AUF
                where  KName = 'Weiss')
```

5. Welche Lieferanten (LName) liefern mindestens eine Ware, die auch Grau liefert?

```
select L.LName
from   LIEF L, LIEF LG
where  L.Ware = LG.Ware and LG.LName = 'Grau'
```

$\{r : (LName) | \exists l : LIEF, lg : LIEF$
$(r.LName = l.LName \wedge l.Ware = lg.Ware \wedge lg.LName = `Grau')\}$

```
select L.LName
from   LIEF L
where  exists ( select LG.Ware
                from   LIEF LG
                where  L.Ware = LG.Ware and LG.LName = 'Grau' )
```

```
select L.LName
from   LIEF L
where  exists ( select *
                from   LIEF LG
                where  L.Ware = LG.Ware and LG.LName = 'Grau' )
```

$\{r : (LName) | \exists l : LIEF(r.LName = l.LName \wedge$
$\exists lg : LIEF(l.Ware = lg.Ware \wedge lg.LName = `Grau'))\}$

6. Welche Lieferanten (alle Attribute) liefern welche Waren genau so preiswert wie alle Lieferanten, die diese Ware liefern?

```
select *
from   LIEF L
where  Preis <= all (select Preis
                     from   LIEF
                     where  Ware = L.Ware)
```

$$\{r : (LName, LAdr, Ware, Preis) | \exists l : LIEF(r = l \wedge \\ \forall l' : LIEF(l'.Ware = l.Ware \Rightarrow l.Preis <= l'.Preis))\}$$

Man beachte, daß diese Anfrage nicht kann entschachtelt werden kann. Die Angabe $*$ in der `select`-Liste liefert alle Attribute.

FRAGE: Was ist das Delta zu: obige Implikation durch Konjunktion ersetzen?

7. Stelle eine Liste von Kunden (KName, KAdr) und möglichen Lieferanten (LName, LAdr) ihrer Aufträge zusammen.

```
select KName, KAdr, LName, LAdr
from   KUNDE, LIEF
where  exists (select *
               from   AUF
               where  Ware=LIEF.Ware and KName=KUNDE.KName)
```

$$\{r : (KName, LName, LAdr) | \exists k : KUNDE, l : LIEF \\ (r.KName = k.KName \wedge r.LName = l.LName \wedge r.LAdr = l.LAdr \wedge \\ \exists a : AUF(a.Ware = l.Ware \wedge a.KName = k.KName))\}$$

Die Anfrage kann auch mit `= any` anstelle von `exists` formuliert werden:

```
select KName, KAdr, LName, LAdr
from   KUNDE, LIEF
where  Ware = any (select Ware
                   from   AUF
                   where  KName=KUNDE.KName)
```

oder

```
select KName, KAdr, LName, LAdr
from   KUNDE, LIEF
where  KName = any (select KName
                    from   AUF
                    where  Ware=LIEF.Ware)
```

8. Welche Lieferanten (LName) liefern mindestens alle Waren, die Grau liefert?

```
select LName
from   LIEF L
where  not exists (select Ware
                   from   LIEF
                   where  LName='Grau' and
                          not Ware in (select Ware
                                       from   LIEF
                                       where  LName=L.LName))
```

```
select RES.LName
from   LIEF RES
where  not exists (select GR.Ware
                   from   LIEF GR
                   where  GR.LName='Grau' and
                          not GR.Ware in (select RESWA.Ware
                                          from   LIEF RESWA
                                          where  RESWA.LName=RES.LName))
```

$\{r : (LName) | \exists res : LIEF$
$(r.LName = res.LName \wedge$
$\neg(\exists gr : LIEF(gr.LName = `Grau' \wedge$
$\neg(\exists reswa : LIEF(gr.Ware = reswa.Ware \wedge reswa.LName = res.LName)))))\}$

$\{r : (LName) | \exists res : LIEF$
$(r.LName = res.LName \wedge$
$(\forall gr : LIEF(gr.LName = `Grau' \Rightarrow$
$(\exists reswa : LIEF(gr.Ware = reswa.Ware \wedge reswa.LName = res.LName)))))\}$

RA-Term $\pi_{LName,Ware}(LIEF) : \pi_{Ware}(\sigma_{LName=`Grau'}(LIEF))$

9. Welche Waren sind in Auftrag gegeben oder lieferbar?

```
(select Ware from AUF)
union
(select Ware from LIEF)
```

$\{r : (Ware) | \quad \exists a : AUF(r.Ware = a.Ware) \vee$
$\exists l : LIEF(r.Ware = l.Ware)\}$

Diese Anfrage kann in SQL nicht ohne `union` formuliert werden.

Generelle Anforderungen (τ, τ_1 und τ_2 sind Terme und φ, φ_1 and φ_2 Formeln):

- Eine gegebene SQL Anfrage 'SELECT τ_1, ..., τ_n FROM R_1 s_1, ..., R_m s_mWHERE φ' und alle Unteranfragen sind vollständig mit expliziten Tupelvariablen qualifiziert: Falls ein Attribut A_j in einem Resultatsterm τ_i oder in einer Formel φ auftaucht, ist dieses Auftreten von der Form $s_i.A_j$, wobei s_i in der FROM Klausel (oder im Falle einer Unteranfrage vielleicht im FROM-Teil einer äußeren Anfrage) auftaucht und A_j ein Attribut einer entsprechenden Relation R_i ist.

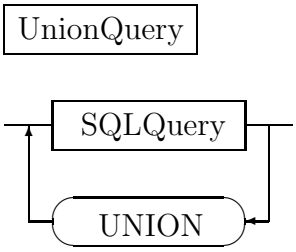


Abbildung 1: Syntax von UnionQuery.

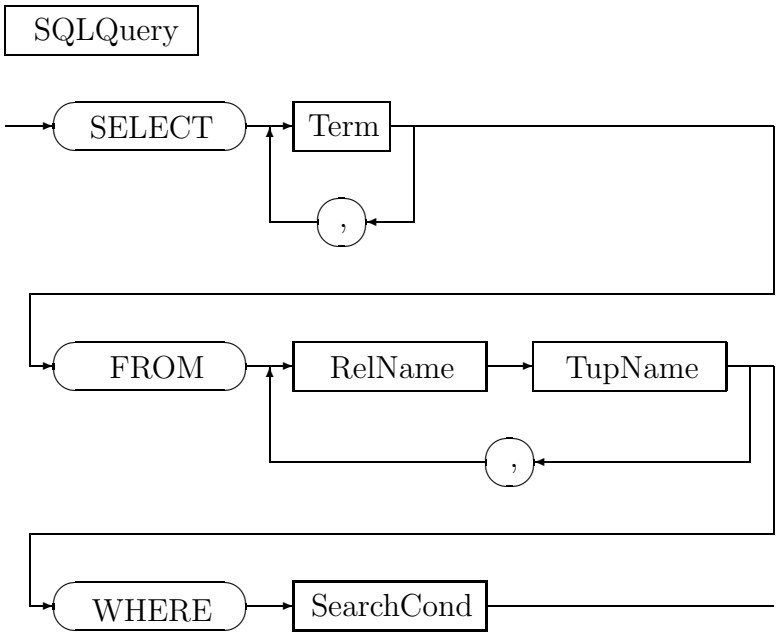


Abbildung 2: Syntax von SQLQuery.

- Die Namen von Tupelvariablen sind eindeutig ($i \neq j \Rightarrow s_i \neq s_j$) und nur einmal deklariert.
- Die Resultatsterme und auch die Formel φ verwenden nur deklarierte Tupelvariable.
- Die Resultatsterme τ_i können auch Konstanten sein.
- Bei Vergleichen wie '$\tau_1 = \tau_2$' oder 'τ_1 = ANY (SELECT τ_2 FROM ... WHERE ...)' haben τ_1 and τ_2 den gleichen Datentyp.
- Für UNION-Ausdrücke nehmen wir an, daß τ_i und τ_i' den gleichen Datentyp haben (i∈1..n).

sql2tc⟦ SELECT τ_1, ..., τ_n FROM R_1 s_1, ..., R_m s_m WHERE φ ⟧ :=
{ r : (Res_1, ..., Res_n) | (∃ s_1:R_1, ..., s_m:R_m)
(r.Res_1 = τ_1 ∧ ... ∧ r.Res_n = τ_n ∧ **sql2tc**⟦ φ ⟧) }

sql2tc⟦ (NOT φ) ⟧ := (¬ **sql2tc**⟦ φ ⟧)

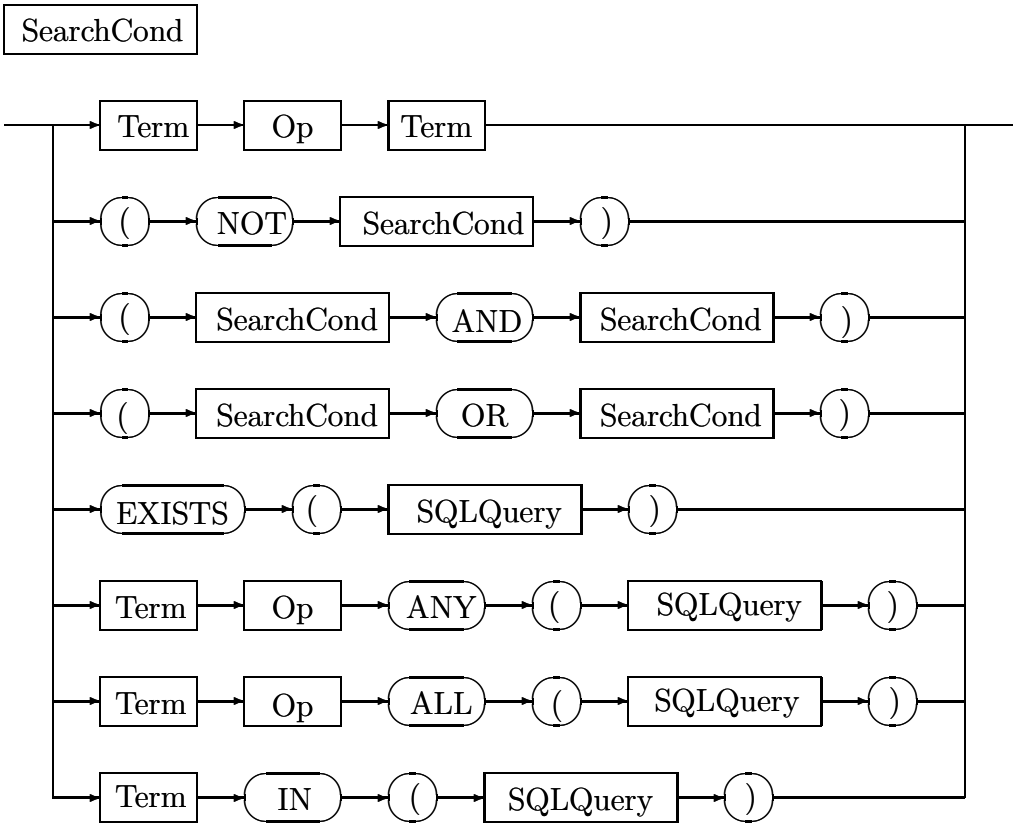


Abbildung 3: Syntax von SearchCond.

sql2tc⟦ (φ_1 AND φ_2) ⟧ := (**sql2tc**⟦ φ_1 ⟧ ∧ **sql2tc**⟦ φ_2 ⟧)

sql2tc⟦ (φ_1 OR φ_2) ⟧ := (**sql2tc**⟦ φ_1 ⟧ ∨ **sql2tc**⟦ φ_2 ⟧)

sql2tc⟦ $\tau_1 \omega \tau_2$ ⟧ := $\tau_1 \omega \tau_2$

sql2tc⟦ τ ω ALL (SELECT s_i.A FROM R_1 s_1, ..., R_i s_i, ..., R_m s_m WHERE φ) ⟧ :=
(∀ s_i:R_i) (((∃ s_1:R_1, ..., s_{i-1}:R_{i-1}, s_{i+1}:R_{i+1}, ..., s_m:R_m) **sql2tc**⟦ φ ⟧) ⇒ τ ω s_i.A)

sql2tc⟦ τ ω ANY (SELECT s_i.A FROM R_1 s_1, ..., R_i s_i, ..., R_m s_m WHERE φ) ⟧ :=
(∃ s_i:R_i) (((∃ s_1:R_1, ..., s_{i-1}:R_{i-1}, s_{i+1}:R_{i+1}, ..., s_m:R_m) **sql2tc**⟦ φ ⟧) ∧ τ ω s_i.A)

sql2tc⟦ τ IN (SELECT s_i.A FROM R_1 s_1, ..., R_i s_i, ..., R_m s_m WHERE φ) ⟧ :=
(∃ s_i:R_i) (((∃ s_1:R_1, ..., s_{i-1}:R_{i-1}, s_{i+1}:R_{i+1}, ..., s_m:R_m) **sql2tc**⟦ φ ⟧) ∧ τ = s_i.A)

sql2tc⟦ EXISTS (SELECT r_1.A_1, ..., r_n.A_n FROM R_1 s_1, ..., R_m s_m WHERE φ) ⟧ :=
(∃ s_1:R_1, ..., s_m:R_m) **sql2tc**⟦ φ ⟧

sql2tc⟦ SELECT τ_1, ..., τ_n FROM R_1 s_1, ..., R_m s_m WHERE φ
UNION
SELECT τ_1', ..., τ_n' FROM R_1' s_1', ..., R_k' s_k' WHERE φ' ⟧ :=

$\{$ r : (Res_1, ..., Res_n) |
($\exists$ s_1:R_1, ..., s_m:R_m) (r.$\mathsf{Res}_1 = \tau_1 \wedge$ ... $\wedge$ r.$\mathsf{Res}_n = \tau_n \wedge$ **sql2tc**$[\![\varphi]\!]$) $\vee$
($\exists$ s_1':R_1', ..., s_k':R_k') (r.$\mathsf{Res}_1 = \tau_1$'$\wedge$ ... $\wedge$ r.$\mathsf{Res}_n = \tau_n$' $\wedge$ **sql2tc**$[\![\varphi$' $]\!]$) $\}$

Die UNION-Regel kann auf den Fall mit mehr als 2 Operanden verallgemeinert werden.

Beispiel

$$
\begin{array}{l}
select\ * \\
from\ LIEF\ L \\
where\ Preis <= all\ (\ select\ Preis \\
\qquad\qquad\qquad\qquad from\ LIEF \\
\qquad\qquad\qquad\qquad where\ Ware = L.Ware\ )
\end{array}
$$

$$
\begin{array}{l}
\mathbf{sql2tc}[\![\ select\ l1.LName, l1.LAdr, l1.Ware, l1.Preis \\
\qquad\quad from\ LIEF\ l1 \\
\qquad\quad where\ l1.Preis <= all\ (\ select\ l2.Preis \\
\qquad\qquad\qquad\qquad\qquad\qquad from\ LIEF\ l2 \\
\qquad\qquad\qquad\qquad\qquad\qquad where\ l2.Ware = l1.Ware)]\!]
\end{array}
$$

$$
\begin{array}{l}
\{r : (Res_1, Res_2, Res_3, Res_4) | \\
\quad \exists l1 : LIEF(r.Res_1 = l1.LName \wedge r.Res_2 = l1.LAdr \wedge r.Res_3 = l1.Ware \wedge r.Res_4 = l1.Preis \wedge \\
\quad \mathbf{sql2tc}[\![l1.Preis <= all\ (\ select\ l2.Preis \\
\qquad\qquad\qquad\qquad\qquad\qquad from\ LIEF\ l2 \\
\qquad\qquad\qquad\qquad\qquad\qquad where\ l2.Ware = l1.Ware)]\!])\}
\end{array}
$$

$$
\begin{array}{l}
\{r : (Res_1, Res_2, Res_3, Res_4) | \\
\quad \exists l1 : LIEF(r.Res_1 = l1.LName \wedge r.Res_2 = l1.LAdr \wedge r.Res_3 = l1.Ware \wedge r.Res_4 = l1.Preis \wedge \\
\qquad \forall l2 : LIEF(l2.Ware = l1.Ware \Rightarrow l1.Preis <= l2.Preis))\}
\end{array}
$$

Theorem:

(a) τ_1 IN (SELECT τ_2 FROM ρ WHERE φ) $\Leftrightarrow$
$\tau_1 =$ ANY (SELECT τ_2 FROM ρ WHERE φ)

(b) NOT (τ_1 ω ALL (SELECT τ_2 FROM ρ WHERE φ)) $\Leftrightarrow$
τ_1 $\overline{\omega}$ ANY (SELECT τ_2 FROM ρ WHERE φ)

(c) NOT (τ_1 ω ANY (SELECT τ_2 FROM ρ WHERE φ)) $\Leftrightarrow$
τ_1 $\overline{\omega}$ ALL (SELECT τ_2 FROM ρ WHERE φ)

(d) τ_1 ω ANY (SELECT τ_2 FROM ρ WHERE φ) $\Leftrightarrow$
EXISTS (SELECT $\tau(\rho)$ FROM ρ WHERE (φ) AND $\tau_1 \omega \tau_2$)

(e) τ_1 ω ALL (SELECT τ_2 FROM ρ WHERE φ) $\Leftrightarrow$
NOT EXISTS (SELECT $\tau(\rho)$ FROM ρ WHERE (φ) AND $\tau_1 \overline{\omega} \tau_2$)

(f) EXISTS (SELECT τ FROM ρ WHERE φ) $\Leftrightarrow$ EXISTS (SELECT * FROM ρ WHERE φ)

$\overline{\omega}$ die Negation von ω, e.g. $\overline{\leq} :=>$. $\tau(\rho)$ bezieht sich auf alle Attribute in ρ. Es gilt $\rho \equiv$
R_1 s_1, ... , R_m s_m.

Beweis: Es soll $\rho_i \equiv$ s_1:R_1, ..., s_{i-1}:R_{i-1}, s_{i+1}:R_{i+1}, ..., s_m:R_m gelten.

zu (a)

$\mathbf{sql2tc}[\![$ τ_1 IN (SELECT τ_2 FROM ρ WHERE φ) $]\!] \Leftrightarrow$
$(\ \exists\ s_i{:}R_i\ )\ (\ (\ (\ \exists \rho_i\ )\ \mathbf{sql2tc}[\![\varphi]\!]\ ) \wedge \tau_1 = \tau_2) \Leftrightarrow$
$\mathbf{sql2tc}[\![$ $\tau_1 =$ ANY (SELECT τ_2 FROM ρ WHERE φ) $]\!]$

zu (b)

$\mathbf{sql2tc}[\![$ NOT ($\tau_1\ \omega$ ALL (SELECT τ_2 FROM ρ WHERE φ)) $]\!] \Leftrightarrow$
$\neg\ (\ (\ \forall\ s_i{:}R_i\ )\ (\ (\ \exists \rho_i\ )\ \mathbf{sql2tc}[\![\varphi]\!]\ ) \Rightarrow \tau_1 \omega \tau_2\ ) \Leftrightarrow$
$\neg\ (\ (\ \forall\ s_i{:}R_i\ )\ (\ (\ \forall \rho_i\ )\ \neg\ \mathbf{sql2tc}[\![\varphi]\!]\ ) \vee \tau_1 \omega \tau_2\ ) \Leftrightarrow$
$\neg\ (\ (\ \forall\ s_i{:}R_i\ )\ (\ \forall \rho_i\ )\ (\ \neg\ \mathbf{sql2tc}[\![\varphi]\!] \vee \tau_1 \omega \tau_2\ )\ ) \Leftrightarrow$
$(\ \exists\ s_i{:}R_i\ )\ (\ \exists \rho_i\ )\ (\ \mathbf{sql2tc}[\![\varphi]\!] \wedge \tau_1 \overline{\omega} \tau_2\ ) \Leftrightarrow$
$\mathbf{sql2tc}[\![$ $\tau_1\ \overline{\omega}$ ANY (SELECT τ_2 FROM ρ WHERE φ) $]\!]$

(c), (d), (e) und (f) analog. q.e.d.

Beispiel: Anwendung $\mathbf{sql2ra}[\![...]\!]$

- KUNDE(KNr,KName), BEST(KNr,Ware)
- Beispiel 1

  $\mathbf{sql2ra}[\![select\ \ k.KName$
  $from\ \ KUNDE\ k, BEST\ b$
  $where\ k.KNr = b.KNr\ and\ b.Ware = 42]\!] =$

  $\pi_{k.KName}(\sigma_{k.KNr=b.KNr}(K \times B) \cap \sigma_{b.Ware=42}(K \times B))$

  $K \times B =$

  $\delta_{k.KNr \leftarrow KNr, k.KName \leftarrow KName}(KUNDE) \times \delta_{b.KNr \leftarrow KNr, b.Ware \leftarrow Ware}(BEST)$
- Beispiel 2

  $\mathbf{sql2ra}[\![select\ \ k.KName$
  $from\ \ KUNDE\ k$
  $where\ not\ \underbrace{exists\ (select\ *\ from\ BEST\ b\ where\ k.KNr = b.KNr)}_{EXISTS'}]\!] =_{(Regel\ SELECT)}$

  $\pi_{k.KName}(\mathbf{sql2ra}[\![not\ EXISTS']\!](\ \underbrace{rename[\![KUNDE\ k]\!]}_{KUNDE' = \delta_{k.KNr \leftarrow KNr, k.KName \leftarrow KName}(KUNDE)}\ )) =_{(Regel\ NOT)}$

  $\pi_{k.KName}(\mathbf{sql2ra\text{-}}[\![EXISTS']\!](KUNDE')) =_{(Regel\ \mathbf{sql2ra\text{-}})}$

  $\pi_{k.KName}(KUNDE' -$
  $\pi_{k.KNr,k.KName}(\mathbf{sql2ra}[\![EXISTS']\!](KUNDE'))) =_{(Regel\ EXISTS)}$

  $\pi_{k.KName}(KUNDE' -$
  $\pi_{k.KNr,k.KName}($
  $\mathbf{sql2ra}[\![k.KNr = b.KNr]\!](KUNDE'\ \times\ \underbrace{rename[\![BEST\ b]\!]}_{BEST' = \delta_{b.KNr \leftarrow KNr, b.Ware \leftarrow Ware}(BEST)}\ ))) =_{(Regel\ \tau_1\ \omega\ \tau_2)}$

  $\pi_{k.KName}(KUNDE' -$
  $\pi_{k.KNr,k.KName}(\sigma_{k.KNr=b.KNr}(KUNDE' \times BEST')))$