UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Lehrstuhl Prof. Weikum

Datenbanksysteme WS 1999/2000

Übung 2 — Beispiellösung

Aufgabe 1: Relationenkalkül

Gegeben sei das aus der letzten Übung bekannte Schema einer Universitätsdatenbank:

Professor	(P_Name, Fachbereich_Nr, Gebäude, Raum, Tel)
Fachbereich	(Fachbereich_Nr, F_Name, Prodekan)
Gebäude	(Gebäude, Hausmeister)
Student	(Matrikel_Nr, S_Name, Semester, Fachbereich_Nr)
Prüfung	(Matrikel_Nr, Fach, Prüfer, Note)

Formulieren Sie die folgenden Anfragen als Ausdrücke des Tupel-Relationenkalküls und des Domain-Relationenkalküls.

a) Wer ist Prodekan des Fachbereichs 14?
Tupel-Relationenkalkül:
$\{ \text{f.Prodekan} \mid \text{f} \in \text{Fachbereich} \land \text{f.Fachbereich\_Nr} = \text{'14'} \}$
Domain-Relationenkalkül:
$\{ \text{pro} \mid \exists \text{ nr,name: Fachbereich(nr, name, pro)} \land \text{nr='14'}\}$

b) In welchen Gebäuden befinden sich Professoren des Fachbereichs 14?
Tupel-Relationenkalkül:
$\{ \text{p.Gebäude} \mid \text{p} \in \text{Professor} \land \text{p.Fachbereich\_Nr} = \text{'14'} \}$
Domain-Relationenkalkül:
$\{ \text{geb} \mid \exists \text{ name,nr,raum,tel: Professor(name,nr,geb,raum,tel)} \land \text{nr='14'}\}$

c) Welche Studenten haben sich im Fach Datenbanksysteme bei Professor Weikum prüfen lassen und haben nicht bestanden?
Tupel-Relationenkalkül:
$\{ \text{p.Matrikel\_Nr} \mid \text{p} \in \text{Prüfung} \land \text{p.Fach} = \text{'Datenbanksysteme'} \land \text{p.Prüfer} = \text{'Weikum'} \land \text{p.Note}>4.3\}$

Domain-Relationenkalkül:
$\{$ nr $|\exists$ fa, pr, no: Prüfung(nr, fa, pr, no) $\wedge$ fa='Datenbanksysteme' $\wedge$ pr = 'Weikum' $\wedge$ no $>$ 4.3 $\}$

d) An welchen Hausmeister muß sich Professor Weikum wenden, wenn er seinen Zimmerschlüssel vergessen hat?
Tupel-Relationenkalkül:
$\{$ g.Hausmeister $|$ g $\in$ Gebäude $\wedge\exists$ p: (p $\in$ Professor $\wedge$ g.Gebäude = p.Gebäude $\wedge$ p.P_Name = 'Weikum')$\}$
Domain-Relationenkalkül:
$\{$ hm $|\exists$ geb: Gebäude(hm,**geb**) $\wedge\exists$ na, fa, ra, tel: Professor(na, fa, **geb**, ra, tel) $\wedge$ na = 'Weikum' $\}$

e) Welche Studenten, die mindestens im sechsten Semester sind, haben in allen bisherigen Prüfungen mindestens die Note 2.0 erreicht?
Tupel-Relationenkalkül:
$\{$ s.S_Name $|$ s $\in$ Student $\wedge$ s.Semester $\geq$ 6 $\wedge$ $\forall$ p: ((p $\in$ Prüfung $\wedge$ s.Matrikel_Nr = p.Matrikel_Nr) $\rightarrow$ p.Note $\leq$ 2.0) $\}$

ist äquivalent zu

$\{$ s.S_Name $|$ s $\in$ Student $\wedge$ s.Semester $\geq$ 6 $\wedge$ $\forall$ p: ($\neg$ (p $\in$ Prüfung) $\vee$ s.Matrikel_Nr $\neq$ p.Matrikel_Nr $\vee$ p.Note $\leq$ 2.0) $\}$

ist äquivalent zu

$\{$ s.S_Name $|$ s $\in$ Student $\wedge$ s.Semester $\geq$ 6 $\wedge$ $\neg\exists$ p: (p $\in$ Prüfung $\wedge$ s.Matrikel_Nr = p.Matrikel_Nr $\wedge$ p.Note $>$ 2.0) $\}$

Domain-Relationenkalkül:
$\{$ na $|\exists$ nr, se, fa: Student(nr, na, se, fa) $\wedge$ se $\geq$ 6 $\wedge$ $\neg$($\exists$ nr2, pfa, pr, no: Prüfung(nr2, pfa, pr, no) $\wedge$ nr2=nr $\wedge$ no $>$ 2.0) $\}$

$\{$ na $|\exists$ nr, se, fa: Student(**nr**, na, se, fa) $\wedge$ se $\geq$ 6 $\wedge$ $\neg$($\exists$ pfa, pr, no: Prüfung(**nr**, pfa, pr, no) $\wedge$ no $>$ 2.0) $\}$

f) Student Hugo Meier will den Prodekan seines Fachbereiches anrufen. Wie lautet die Telefonnummer?
Tupel-Relationenkalkül:
$\{$ p.Tel $|$ p $\in$ Professor $\wedge$ $\exists$ f: (f $\in$ Fachbereich $\wedge\exists$ s: (s $\in$ Student $\wedge$ p.Fachbereich_Nr = f.Fachbereich_Nr $\wedge$ p.P_Name = f.Prodekan $\wedge$ s.Fachbereich_Nr = f.Fachbereich_Nr $\wedge$ s.S_Name = 'Hugo Meier'))$\}$
Domain-Relationenkalkül:
$\{$ te $|\exists$ fnr, fna, pr: Fachbereich(**fnr**, fna, **pr**) $\wedge$ $\exists$ na, ge, ra: Professor(**pr**, fnr, ge, ra, te) $\wedge$ $\exists$ nr, sna, se: Student(nr, sna, se, **fnr**) $\wedge$ sna = 'Hugo Meier' $\}$

Aufgabe 2: Sicherheit von TRK-Ausdrücken

Gegeben sei das Schema der Universitätsdatenbank aus Aufgabe 1.

Geben Sie bei den folgenden Anfragen im Relationen-Tupelkalkül an, ob die Ausdrücke sicher sind oder nicht. Begründen Sie Ihre Antwort basierend auf der in der Vorlesung vorgestellten Definition zu sicheren Formeln (Skript S.35/36). Wandeln Sie außerdem unsichere Anfragen in sichere um.

a) Alle Studenten (Namen), die ausschließlich Prüfungen in Fächern gemacht haben, die Professor Weikum schon mindestens einmal geprüft hat.

$$\begin{aligned}\{s.S\_Name \mid\ & s \in Student \wedge \\ & \forall p1 : (p1 \in Prüfung \wedge p1.MNr = s.MNr \wedge \\ & \quad \exists p2 : (p2 \in Prüfung \wedge p1.Fach = p2.Fach \\ & \qquad \wedge\, p2.Prüfer = 'Weikum'\,))\}\end{aligned}$$

Der Ausdruck ist unsicher, weil in der Teilformel mit dem Allquantor alle Attribute der Tupelvariablen p1 unbeschränkt sein müssen. Durch '$p1 \in Prüfung$' sind die Attribute jedoch beschränkt, was gemäß der Definition zur Unsicherheit zu einem unsicheren Ausdruck führt. Korrekt wäre deshalb:

$$\begin{aligned}\{s.S\_Name \mid\ & s \in Student \wedge \\ & \forall p1 : (p1 \in Prüfung \wedge p1.MNr = s.MNr) \Rightarrow \\ & \quad \exists p2 : (p2 \in Prüfung \wedge p1.Fach = p2.Fach \\ & \qquad \wedge\, p2.Prüfer = 'Weikum'\,)\}\end{aligned}$$

was äquivalent ist zu

$$\begin{aligned}\{s.S\_Name \mid\ & s \in Student \wedge \\ & \forall p1 : (\neg(p1 \in Prüfung) \vee p1.MNr \neq s.MNr \vee \\ & \quad \exists p2 : (p2 \in Prüfung \wedge p1.Fach = p2.Fach \\ & \qquad \wedge\, p2.Prüfer = 'Weikum'\,))\}\end{aligned}$$

b) Alle Studenten (Namen), die mindestens einen Kommilitonen im selben Fachbereich haben, der sich genau von denselben Professoren hat prüfen lassen.

$$\begin{aligned}&\{s1.S\_Name \mid \\ &\ \exists s2 : (s2 \in Student \wedge s2.MNr = s1.MNr \wedge \\ &\quad s2.FB\_Nr = s1.FB\_Nr \wedge \\ &\quad \forall t1 : (\neg(t1 \in Prüfung) \vee t1.MNr \neq s1.MNr \wedge \\ &\qquad \exists t2 : (t2 \in Prüfung \wedge t2.MNr = s2.MNr \wedge \\ &\qquad\quad t1.Prüfer = t2.Prüfer \wedge \\ &\qquad\quad \neg\exists t3 : (t3 \in Prüfung \wedge t3.MNr = s2.MNr \wedge \\ &\qquad\qquad \neg\exists t4 : (\neg(t4 \in Prüfung) \vee \\ &\qquad\qquad\quad t4.MNr = s1.MNr \wedge \\ &\qquad\qquad\quad t4.Prüfer \neq t3.Prüfer)))))\}\end{aligned}$$

Der Ausdruck ist ebenfalls unsicher, da zur Sicherheit alle Attribute aller freien Variablen beschränkt sein müssen, was für s1 nicht der Fall ist. Wir beschränken s1 nun durch $\{s1.S\_Name \mid s1 \in Student \wedge \ldots\}$

Dieser Ausdruck ist jedoch auch noch nicht sicher, da in der Teilformel $\exists t4\, F(t4)$ alle Attribute von $t4$ in F beschränkt sein müssen, was durch $\neg(t4 \in Prüfung) \vee \ldots$ nicht der Fall ist. Dieser Teil muß deshalb zunächst durch $(t4 \in Prüfung) \ldots$ ersetzt werden. Durch die folgende Disjunktion ist die Teilformel $F(t4)$ jedoch immer noch unsicher, da zur Sicherheit auch der Teil $t4.MNr = s1.MNr \wedge t4.Prüfer \neq t3.Prüfer$ in allen Attributen von t4 beschränkt sein müsste.

Um die Teilformel $F(t4)$ sicher zu machen (und um letztendlich zu dem Ergebnis zu gelangen, an dem wir interessiert sind), wird die Disjunktion zunächst in eine Konjunktion umgeformt.

Der endgültige sichere Ausdruck für unsere Query lautet also nun:

$$
\begin{array}{l}
\{s1.S\_Name \mid \mathbf{s1} {\in} \mathbf{Student} \wedge \\
\quad \exists s2 : (s2 \in Student \wedge s2.MNr \neq s1.MNr \wedge \\
\qquad\quad s2.FB\_Nr = s1.FB\_Nr \wedge \\
\qquad\quad \forall t1 : (\neg(t1 \in Prüfung) \vee t1.MNr \neq s1.MNr \wedge \\
\qquad\qquad\quad \exists t2 : (t2 \in Prüfung \wedge t2.MNr = s2.MNr \wedge \\
\qquad\qquad\qquad\quad t1.Prüfer = t2.Prüfer \wedge \\
\qquad\qquad\qquad\quad \neg\exists t3 : (t3 \in Prüfung \wedge t3.MNr = s2.MNr \wedge \\
\qquad\qquad\qquad\qquad\quad \neg\exists t4 : (\mathbf{t4} {\in} \mathbf{Prüfung}) \wedge \\
\qquad\qquad\qquad\qquad\qquad\quad t4.MNr = s1.MNr \wedge \\
\qquad\qquad\qquad\qquad\qquad\quad t4.Prüfer \neq t3.Prüfer)))))\}
\end{array}
$$

Zwar ist dieser Ausdruck nun sicher, da die Kriterien für die Sicherheit erfüllt sind, die Query liefert aber nicht das gewünschte Ergebnis. Diese Query liefert nämlich die Namen aller Studenten s1, die mindestens einen Komilitonen im selben Fachbereich haben, der alle Prüfungen, die s1 absolviert hat, ebenfalls beim selben Prüfer absolviert hat. Außerdem müssen beide mindestens eine Prüfung bei unterschiedlichen Prüfern gehabt haben.

Das gewünschte Ergebnis bekommen wir durch die Forderung, daß t4.Prüfer=t3.Prüfer sein muß. Die domain-unabhängige Query, die das korrekte Ergebnis liefert, lautet deshalb:

$$
\begin{array}{l}
\{s1.S\_Name \mid \mathbf{s1} {\in} \mathbf{Student} \wedge \\
\quad \exists s2 : (s2 \in Student \wedge s2.MNr \neq s1.MNr \wedge \\
\qquad\quad s2.FB\_Nr = s1.FB\_Nr \wedge \\
\qquad\quad \forall t1 : (\neg(t1 \in Prüfung) \vee t1.MNr \neq s1.MNr \wedge
\end{array}
$$

$$
\begin{aligned}
\exists t2 : (&t2 \in Prüfung \wedge t2.MNr = s2.MNr \wedge \\
&t1.Prüfer = t2.Prüfer \wedge \\
&\neg \exists t3 : (t3 \in Prüfung \wedge t3.MNr = s2.MNr \wedge \\
&\qquad \neg \exists t4 : (\mathbf{t4 \in Prüfung}) \wedge \\
&\qquad\qquad t4.MNr = s1.MNr \wedge \\
&\qquad\qquad \mathbf{t4.Prüfer = t3.Prüfer})))))\}
\end{aligned}
$$

Hinweis:(siehe dazu Skriptseiten 35/36)

Das Ergebnis einer Anfrage an eine Datenbank soll immer nur vom Inhalt der Datenbank, nicht aber von den Domains der Datenbank abhängen. Man sollte deshalb immer nur domain-unabhängige Anfragen stellen, um unendlich große Ergebnismengen bzw. immer leere Ergebnismengen (und damit 'falsche' Ergebnisse) zu vermeiden. Mit den Kriterien zur Sicherheit von Ausdrücken haben wir eine formale, rein syntaktisch prüfbare Methode kennengelernt, die Domain-Unabhängigkeit von Queries nachzuweisen. Die Endlichkeit des Ergebnisses wird durch die Beschränktheit der freien Variablen sichergestellt. Um zusätzlich zu gewährleisten, daß Bedingungen in der Formel nicht von den Domains abhängen, wird zusätzlich für jede in der Query enthaltene, quantifizierte Teilformel gefordert, daß die gebundene Variable ebenfalls beschränkt ist.

Sicherheit ist aber nur eine hinreichende Bedingung für Domain-Unabhängigkeit. Es gibt also Formeln, die zwar domain-unabhängig sind, aber wo die formalen Kriterien zur Sicherheit nicht erfüllt sind. Insbesondere besteht auch die Gefahr, daß durch Äquivalenzumformungen (wie z.B. durch DeMorgansche Regeln etc.) zwar immer die Domain-Abhängigkeit bzw. -Unabhängigkeit der Formeln erhalten bleibt, die Formel aber zwischen sicher und unsicher wechseln kann, da wir die Syntax ändern.

Beispiel 1: Folgende Formel ist domain-unabhängig, aber nicht sicher.

$\{t \mid \exists s : (s \in S \wedge s = t) \wedge \exists u : (u = t)\}$

In dieser Formel ist die freie Variable t durch die erste Teilformel in Verbindung mit der Konjunktion beschränkt. Die Formel ist aber insgesamt unsicher, da die Variable u in der zweiten Teilformel nicht beschränkt ist. Um u zu beschränken, müßte t innerhalb der Teilformel beschränkt werden, was nicht der Fall ist. Der Ausdruck ist jedoch domain-unabhängig, da die beschränkte und die unbeschränkte Interpretation dasselbe Ergebnis liefern, nämlich eine Relation, die gleich der Relation S ist. Da die erste Teilformel die zweite impliziert, kann der Ausdruck in den logisch äquivalenten Ausdruck

$\{t \mid \exists s : (s \in S \wedge s = t)\}$

umgeformt werden, der jetzt sicher ist und damit auch die Domain-Unabhängigkeit impliziert.

<u>**Beispiel 2:**</u> Man betrachte folgende unsichere Anfrage.

$$\{t \mid t \in T \wedge \exists s : (s \in S \vee \exists u : (u \in U \wedge u = \langle 1, 2, 3\rangle))\}$$

Die Variable s ist in der Teilformel $(s \in S \vee \exists u : (u \in U \wedge u = \langle 1, 2, 3\rangle))$ nicht beschränkt, da sie durch die Disjunktion in beiden Teilen beschränkt sein müßte, in $\exists u : (u \in U \wedge u = \langle 1, 2, 3\rangle)$ aber nicht beschränkt ist. Formt man die Anfrage in einen äquivalenten Ausdruck der Form

$$\{t \mid t \in T \wedge (\exists s : (s \in S) \vee \exists u : (u \in U \wedge u = \langle 1, 2, 3\rangle))\}$$

um, so ist die Anfrage sicher geworden. Wir haben hier also ein Beispiel für eine Umformung, die die Eigenschaft 'Sicherheit' nicht erhält. Der Ausdruck muß natürlich stets domain-unabhängig sein, da aus der Sicherheit die Domain-Unabhängigkeit folgt.

Aufgabe 3: Datalog I

Betrachten Sie eine deduktive Datenbank mit den folgenden Fakten und Regeln:
(siehe Übungsblatt).

Berechnen Sie das Resultat der Anfrage $SelbeGeneration(Bob, X)$ nach der naiven und der semi-naiven Bottom-up-Evaluationsmethode.

Naiv:

i	P_i		
0	{SG(Adam,Adam),	SG(Abigail,Abigail)	}
1	$P_0 \cup$	{SG(Bonnie,Bonnie),	SG(Bob,Bob),
		SG(Bonnie,Bob),	SG(Bob,Bonnie),
		SG(Bonnie,Bill),	SG(Bill,Bonnie),
		SG(Bob,Bill),	SG(Bill,Bob),
		SG(Bill,Bill)	}
2	$P_1 \cup$	{SG(Clyde,Clyde),	SG(Claire,Claire)
		SG(Clyde,Claire),	SG(Claire,Clyde),
		SG(Cerise,Cerise),	SG(Charly,Charly),
		SG(Cecilia,Cecilia),	
		SG(Cerise,Charly),	SG(Charly,Cerise),
		SG(Cerise,Cecilia),	SG(Cecilia,Cerise),
		SG(Charly,Cecilia),	SG(Cecilia,Charly),
		SG(Clyde,Cerise),	SG(Cerise,Clyde),
		SG(Clyde,Charly),	SG(Charly,Clyde),
		SG(Clyde,Cecilia),	SG(Cecilia,Clyde),
		SG(Claire,Cerise),	SG(Cerise,Claire),
		SG(Claire,Charly),	SG(Charly,Claire),
		SG(Claire,Cecilia),	SG(Cecilia,Claire) }
3	wie P_2		

Semi-Naiv:

i	P_i	Δ_i	
0	$=\Delta_0$	{SG(Adam,Adam),	SG(Abigail,Abigail)}
1	wie P_1^{naiv}	{SG(Bonnie,Bonnie),	SG(Bob,Bob),
		SG(Bonnie,Bob),	SG(Bob,Bonnie),
		SG(Bonnie,Bill),	SG(Bill,Bonnie),
		SG(Bob,Bill),	SG(Bill,Bob),
		SG(Bill,Bill)	}
2	wie P_2^{naiv}	{SG(Clyde,Clyde),	SG(Claire,Claire),
		SG(Clyde,Claire),	SG(Claire,Clyde),
		SG(Cerise,Cerise),	SG(Charly,Charly),
		SG(Cecilia,Cecilia),	
		SG(Cerise,Charly),	SG(Charly,Cerise),
		SG(Cerise,Cecilia),	SG(Cecilia,Cerise),
		SG(Charly,Cecilia),	SG(Cecilia,Charly),
		SG(Clyde, Cerise),	SG(Cerise, Clyde),
		SG(Clyde,Charly),	SG(Charly,Clyde),
		SG(Clyde,Cecilia),	SG(Cecilia,Clyde),
		SG(Claire,Cerise),	SG(Cerise,Claire),
		SG(Claire,Charly),	SG(Charly,Claire),
		SG(Claire,Cecilia),	SG(Cecilia,Claire) }
3	wie P_2	$\emptyset$	

Als Ergebnis der Anfrage erhält man in beiden Fällen:

SG(Bob,Bob), SG(Bob,Bonnie), SG(Bob,Bill)

Ehepaare sind nach den angegebenen Regeln nicht in derselben Generation. Dies liegt daran, daß man bei Berücksichtigung der eingeheirateten Personen einen gerichteten Graphen anstelle eines Baumes erhält. Dies ist unter Umständen sinnvoll, da zwei gleichalte Personen durchaus verschiedenen Generationen gegenüber dem "Urpaar" angehören können.

Aufgabe 4: Datalog II

Zeigen Sie, daß die semi-naive Bottom-Up-Auswertung für Datalog-Programme ohne Negation zur naiven Bottom-up-Auswertung äquivalent ist. Der Iterationsschritt der naiven Auswertung läßt sich relationsalgebraisch wie folgt schreiben:

$$P_i^N = P_{i-1}^N \cup (R \bowtie P_{i-1}^N) \ (1)$$

Aus dem Algorithmus der seminaiven Vorgehensweise kann man entsprechend folgende Invariante ableiten:

$$P_i^S = P_{i-1}^S \cup (R \bowtie \Delta_{i-1}^S) \ (2)$$

Definitionsgemäß gilt für die semi-naive Auswertung:

$$\Delta_i^S = P_i^S - P_{i-1}^S \ (3)$$

Gleichung (3) läßt sich wie folgt umformen:

$$P_i^S = \Delta_i^S \cup P_{i-1}^S \ (3')$$

Aus (1) folgt:

$$R \bowtie P_{i-1}^N \subseteq P_i^N \ (1')$$

Induktion über die Anzahl der Schleifenausführungen:

$$\begin{array}{llcll}
i = 0: & P_0^N & = & P_0 & \text{siehe naives Prorgamm} \\
 & P_0^S & = & \Delta_0^S = P_0 & \text{siehe seminaives Progr.} \\
 & & & & \\
i \rightarrow i+1: & P_{i+1}^N & = & P_i^N \cup (R \bowtie P_i^N) & (1) \\
 & & = & P_i^N \cup (R \bowtie P_i^S) & \text{Ind. Voraussetzung} \\
 & & = & P_i^N \cup (R \bowtie (\Delta_i^S \cup P_{i-1}^S)) & (3') \\
 & & = & P_i^N \cup (R \bowtie (\Delta_i^S \cup P_{i-1}^N)) & \text{Ind. Voraussetzung} \\
 & & = & P_i^N \cup ((R \bowtie \Delta_i^S) \cup (R \bowtie P_{i-1}^N)) & \text{Distrib. von } \bowtie \text{ und } \cup \\
 & & = & P_i^N \cup ((R \bowtie P_{i-1}^N) \cup (R \bowtie \Delta_i^S)) & \text{Komm. von } \cup \\
 & & = & (P_i^N \cup (R \bowtie P_{i-1}^N)) \cup (R \bowtie \Delta_i^S) & \text{Assoz. von } \cup \\
 & & = & P_i^N \cup (R \bowtie \Delta_i^S) & (1') \\
 & & = & P_i^S \cup (R \bowtie \Delta_i^S) & \text{Ind. Voraussetzung} \\
 & & = & P_{i+1}^S & (2)
\end{array}$$

q.e.d.