TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe20

Maximilian {Bandle, Schüle}, Josef Schmeißer (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss20/impldb/

Blatt Nr. 05

Hinweise Die Datalogaufgaben können auf `https://datalog.db.in.tum.de/` getestet werden. Auf der Seite kann unter *examples* ein entsprechender Datensatz geladen werden. Die neuen IDB Regeln sollten am Ende der EDB definiert und dann im Query-Eingabefeld abgefragt werden.

Zusätzlich zu der in der Vorlesung vorgestellten Syntax hier noch eine Kurzübersicht der Vergleichsoperatoren: $X < Y, Y > X$ (kleiner, größer), $X =< Y, X >= Y$ (kleiner oder gleich, größer oder gleich), $X = Y, X \backslash = Y$ (gleich, ungleich), $not(pred(X, Y))$ (existiert nicht pred(X,Y)).

Hausaufgabe 1

Definieren Sie das Prädikat `sg(X,Y)` das für "same generation" steht. Zwei Personen gehören zur selben Generation, wenn Sie mindestens je ein Elternteil haben, das derselben Generation angehört.

Verwenden Sie beispielsweise die folgende Ausprägung einer ElternKind Relation. Das erste Element ist hier das Kind, das Zweite ein Elternteil.

```
parent(c,a).
parent(d,a).
parent(d,b).
parent(e,b).
parent(f,c).
parent(g,c).
parent(h,d).
parent(i,d).
parent(i,e).
parent(f,e).
parent(j,f).
parent(j,h).
parent(k,g).
parent(k,i).
```

a) Definieren Sie das Prädikat in Datalog.

```
sg(X,Y) = :- parent(Z,X),X=Y. % X als Elternteil
sg(X,Y) = :- parent(X,Z),X=Y. % X als Kind
% X,Y Kind von U und V, U und V gleiche Generation
sg(X,Y) = :- sg(U,V),parent(X,U),parent(Y,V).
```

b) Demonstrieren Sie die naive Ausführung des Prädikats.

c) Erläutern Sie das Vorgehen bei der seminaiven Auswertung.

Siehe Übungsfolien.

Gruppenaufgabe 2

Ist folgendes Datalog-Programm stratifiziert?

$$\begin{aligned} p(X,Y) &:- q_1(Y,Z), \neg q_2(Z,X), q_3(X,P). \\ q_2(Z,X) &:- q_4(Z,Y), q_3(Y,X). \\ q_4(Z,Y) &:- p(Z,X), q_3(X,Y). \end{aligned}$$

Ist das Programm sicher – unter der Annahme, dass p, q_1, q_2, q_3, q_4 IDB- oder EDB-Prädikate sind?

Loesung: Vgl. Übungsbuch. Das Programm ist **nicht stratifizert**, aber **sicher**. Es ist nicht stratifiziert, weil q_2 von p abhängt, aber negiert in p vorkommt. Es ist sicher, weil alle Variablen in IDB- oder EDB-Prädikaten gebunden sind.

Zur Wiederholung: Eine Regel ist **sicher** gdw. alle Variablen **eingeschränkt** sind. Variable X ist in einer Regel **eingeschränkt**, falls sie im Rumpf enthalten ist und sie:

- in einem positiven Prädikat vorkommt (nicht Vergleichsprädikat),
- $X = c$ (Konstante) oder
- $X = Y$, wenn Y bereits nachgewiesen ist.

Jede Variable eines negierten Prädikates muss bereits eingeschränkt sein.[ab]

Ein Datalog-Programm ist **stratifiziert**, wenn in einer Regel p alle negierten Prädikate $not(q_i)$ nicht von p abhängen (kein Zyklus) und alle positiven Prädikate vorher oder unabhängig von der Reihenfolge (wie bei Rekursion) ausgewertet werden. Formal ausgedrückt können wir jedem Prädikat eine sogenannte Stratifikationsnummer zuordnen. Negierte Prädikate müssen eine echt kleinere Stratifikationsnummer besitzen, positive Prädikate eine maximal so große Stratifikationsnummer wie das davon abhängige Prädikat.

[a] `https://www.dbis.informatik.uni-goettingen.de/Teaching/DB/db-datalog.pdf`
[b] `http://pages.cs.wisc.edu/~paris/cs784-s17/lectures/lecture9.pdf`

Hausaufgabe 3

Gegeben seien die Tabellen `Studenten` und `Punkte` mit Schlüssel `MatrNr`, wobei `Punkte` auf einem separaten Rechner gespeichert ist. Es soll folgende Anfrage ausgeführt werden:

```
SELECT Name, Bonus FROM Student s, Punkte p WHERE s.MatrNr = p.MatrNr;
```

Der Datenbankadministrator entscheidet sich für einen Bloom-Filter zur Vorauswahl der Tupel. Auf `MatrNr` wird die Hash-Funktion $h(x) = x \mod 5$ angewendet.

`Studenten`

MatrNr	**Name**	**Hashwert**
27	Magda	
4	Josef	
19	Erik	
95	Philipp	

`Punkte`

MatrNr	**Bonus**	**Hashwert**
27	ja	
16	nein	
25	nein	
95	ja	

a) Berechnen Sie die Hash-Werte und tragen Sie diese in die obige Tabelle ein.

`Studenten`

MatrNr	**Name**	**Hashwert**
27	Magda	2
4	Josef	4
19	Erik	4
95	Philipp	0

`Punkte`

MatrNr	**Bonus**	**Hashwert**
27	ja	2
16	nein	1
25	nein	0
95	ja	0

b) Geben Sie den von `Studenten` zu übertragenden Bitvektor an.

Bitvektor: 10101
$h(x)$ hat fünf verschiedene Ausgaben. D.h. Vektor ist min. 5 Bits lang. Ein Bit wird dann gesetzt, wenn die Hashfunktion es einmal ausgegeben hat.

c) Geben Sie basierend auf dem Bitvektor an, welche Tupel aus `Punkte` übertragen werden.

27, 25, 95

d) Geben Sie die Falsch-Positiv-Rate (false positive rate) an.

33%

e) Nehmen Sie an, dass jedes Tupel 8 Byte und der Bloomfilter selbst 1 Byte groß ist. Berechnen Sie zunächst die übertragenen Bytes ohne und mit Einsatz des Bloom-Filters.
Ohne Filter: $4 \cdot 8 = 32$
Mit Filter: $3 \cdot 8 + 1 = 25$

Hausaufgabe 4

Überlegen Sie sich, welche Tupel bei der Anwendung des bloomfilterbasierten Joins in Abbildung 1 übertragen werden. Markieren Sie insbesondere, welche Tupel übertragen werden, obwohl sie keinen Joinpartner finden (sog. *false drops*). Wie kann die Anzahl dieser *false drops* verringert werden? Welche Eigenschaften sollte die Hashfunktion $h(c)$ die bei dieser Joinbearbeitung verwendet wird erfüllen?

- False Drops sind c7 und c8 die übertragen werden, obwohl sie keinen Joinpartner finden.
- Bei sinnvoller Hashfunktion weniger False Drops je größer der Vektor ist. Gute Hashfunktion! Mehrere Hashfunktionen zu benutzen verbessert auch die Effizienz des Bloomfilters
- Hashfunktion sollte möglichst gleichmäßig verteilen.

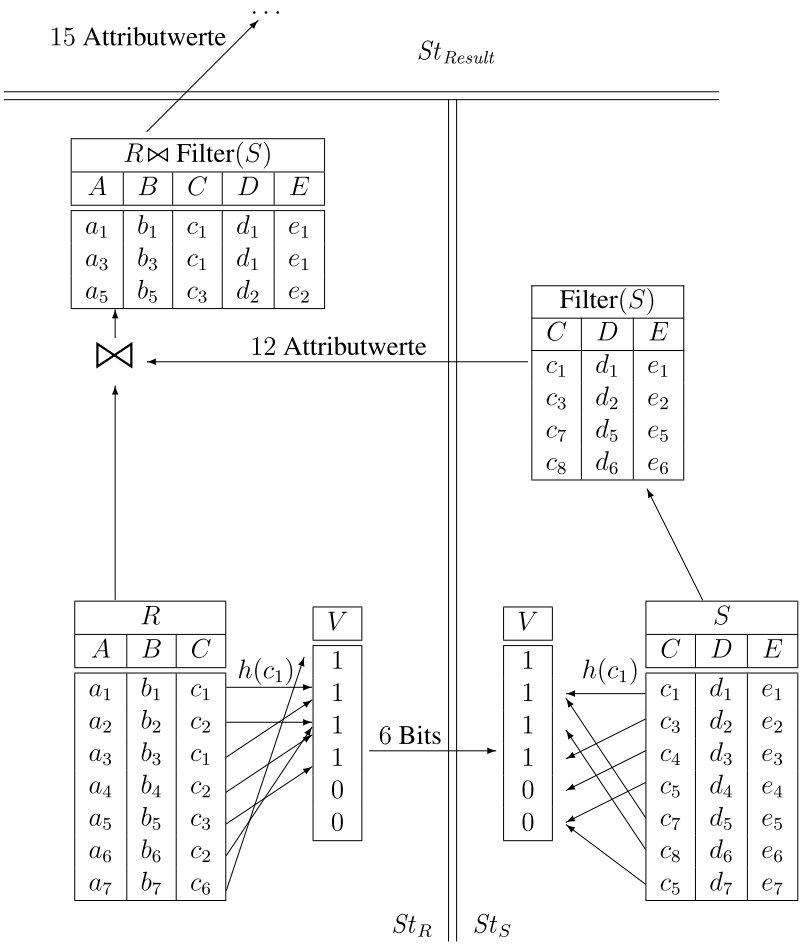

Abbildung 1: Beispiel einer verteilten Joinbearbeitung mit Bloomfilter.