Datenbanksysteme WS 2016/17
Prof. Dr.-Ing. Sebastian Michel
BSc. Peter Brucker
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 11: Ausgabe 24.01.2017, Abgabe 30.01.2017

Aufgabe 1: Prob. Datenbanken - Intensionale Auswertung (1 P.)

Gegeben eine probabilistische Datenbank D mit zwei Relationen S und T.

$S=$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	2	0.5

$T=$

	C	D	
t_1	1	'p'	0.3
t_2	1	'q'	0.2
t_3	2	'q'	0.5

Wir betrachten folgende Anfrage

$$q(z) \leftarrow S(x,y), T(y,z)$$

Bestimmen Sie mögliche Ergebnistupel und deren Wahrscheinlichkeiten anhand der intensionalen Auswertung. Berechnen Sie dazu die aussagenlogischen Formeln (über den Termen s_i und t_j), die die Ergebnistupel beschreiben, anschließend für jede Formel deren Wahrheitstabelle und für jede gültige Belegung die Wahrscheinlichkeit dieser Welt, etc. (analog zum Beispiel aus der Vorlesung).

Hinweis: Einen Truth-Table-Generator finden Sie unter `https://web.stanford.edu/class/cs103/tools/truth-table-tool/`

Aufgabe 2: Intensionale vs. Extensionale Auswertung, Sichere Pläne, Prob. Datenbanken und Postgresql (1 P.)

Beschuldigt=

Zeuge	Verdächtiger	
Mary	Paul	p_1
Mary	John	p_2
Susan	John	p_3

Alibi=

Verdächtiger	Behauptung	
Paul	Kino	q_1
Paul	Freund	q_2
John	Bar	q_3

Gegeben die Anfrage

$$\exists s\ \exists x\ :\ \mathit{Beschuldigt}(w,s) \wedge \mathit{Alibi}(s,x)$$

d.h. "Wer beschuldigt jemanden, der ein Alibi hat?".

(a) Bestimmen Sie die Ergebnisse und deren Wahrscheinlichkeiten (Formel) anhand der intensionalen Anfrageausführung.

(b) Betrachten Sie die folgenden beiden Anfragepläne

(i) $\pi_{\text{Zeuge}}(\text{Beschuldigt} \bowtie \text{Alibi})$

(ii) $\pi_{\text{Zeuge}}(\text{Beschuldigt} \bowtie \pi_{\text{Verdächtiger}}\text{Alibi})$

ob einer dieser Pläne die korrekten Wahrscheinlichkeiten des Ergebnisses liefert, d.h. mit der intensionalen Formel übereinstimmt.

(c) Benutzen Sie Postgresql, um diese Aufgabe zu lösen: Die Statements unten erzeugen und füllen zwei Tabellen R und S. Betrachten Sie die Anfrage $Q(z) = R(z,x), S(x,y)$. Geben Sie dazu entsprechende SQL Anfragen an, sowie deren Ergebnisse, die jeweils einen unsicheren und einen sicheren extensionalen Plan darstellen. Erweitern Sie dazu Postgresql durch die Aggregationsfunktion `prod`, die in der Vorlesung angegeben wurde.

```
create table R(z char(8), x char(8), p float);
create table S(x char(8), y char(8), p float);
insert into R values('c', 'a1', 0.5);
insert into R values('c', 'a2', 0.5);
insert into R values('c', 'a3', 0.5);
insert into S values('a1', 'b1', 0.5);
insert into S values('a1', 'b2', 0.5);
insert into S values('a2', 'b2', 0.5);
insert into S values('a2', 'b3', 0.5);
insert into S values('a2', 'b4', 0.5);
```

Hinweise: (i) Das Ergebnistupel hat laut (korrekter) intensionaler Auswertung die Wahrscheinlichkeit ≈ 0.648. (ii) Um einen bestimmen Plan zu "erzwingen", können Sie Teilanfragen explizit via `WITH .... AS` Statements "vorberechnen".

Aufgabe 3: MapReduce (1 P.)

(a) Wir betrachten Textdokumente, in denen jeder Satz in einer neuen Zeile gespeichert ist, etwa:

```
Dieses Buch richtet sich vor allem an Informatikstudenten der ersten Semester.
Es ist hervorgegangen aus Vorlesungen, die von den Autoren der TU Kaiserslautern gehalten wurden.
Der Inhalt und Stoffumfang wurden mehrfach in der Praxis erprobt.
...
```

Um die Sprache in diesen Texten zu verstehen, erstellen wir einen gerichteten Graphen $G = (V, E)$ wobei V aus allen Worten der Textdokumente besteht und eine Kante $(v_1, v_2) \in E$ falls in einem Satz $v_1 v_2$ direkt nacheinander vorkommen. Zum Beispiel gibt es die Knoten "der", "ersten", "TU", "Praxis" und Kanten von "der" zu "ersten", von "der" zu "TU" und von "der" zu "Praxis".

(i) Geben Sie für einen MapReduce-Job Mapper und Reducer in Pseudocode an, die aus einem Dokumentkorpus den Graphen G konstruieren. Gehen Sie davon aus, dass ein Mapper jeweils eine Zeile des Textes einliest, und der Reducer mittels der Funktion `add_edge(v1, v2)` eine Kante in den Graphen einfügen kann.

(ii) Erweitern Sie den MapReduce-Job derart, dass Sie zu jedem Knoten zählen, wie oft das Wort vorkommt, und zu jeder Kante, wie oft das entsprechende Wort dem ersten folgt. Benutzen Sie dazu die Funktionen `add_weight(v, w)`, die dem Knoten v das Gewicht w zuordnet und `add_edge(v1, v2, w)`, die der Kante (v_1, v_2) das Gewicht w zuordnet.

(b) Gegeben ist folgende CSV-Datei mit Personaldaten (Name, Geschlecht, Firma, Gehalt) als Eingabedokument:

```
Paula,w,IBM,50000
Heinrich,m,SAP,40000
Gabriele,w,IBM,60000
Franka,w,Google,60000
...
```

Skizzieren Sie zu folgenden SQL-Anfragen äquivalente MapReduce-Programme:

(i)
```
SELECT firma, AVG(gehalt) FROM pers WHERE geschlecht='w' GROUP BY firma
HAVING COUNT(*)>=10
```

(ii)
```
SELECT firma, name, gehalt FROM pers a WHERE NOT EXISTS (SELECT * FROM pers b
WHERE b.firma = a.firma AND b.gehalt>a.gehalt)
```