Datenbanksysteme WS 2017/18
Prof. Dr.-Ing. Sebastian Michel
Manuel Hoffmann, Marc Dörr
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 11: Ausgabe 23.01.2018, Abgabe 29.01.2018 16:00 MEZ

Aufgabe 1: Nebenläufige Transaktionen (1 P.)

Benutzen Sie eine Programmiersprache Ihrer Wahl für diese Aufgabe, Sie dürfen das Java Template auf der Website verwenden. Schreiben Sie ein Programm, dass sich mit Postgres verbindet und dabei unterschiedliche Isolationsstufen benutzt.

a) Das Programm soll demonstrieren welche Kombinationen der Isolationsstufen *read committed*, *repeatable read* und *serializable* ein Phantom-Problem haben. Ein Phantom-Problem tritt z.B. auf, wenn eine Transaktion mehrfach `COUNT(*)` auf einer Tabelle ausführt, aber unterschiedliche Ergebnisse bekommt.

b) Benutzen Sie ihr Programm dann um herauszufinden, ob in Postgres das Lost-Update Problem auftreten kann.

Aufgabe 2: 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 3: 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 4: Dies und Das (1 P.)

Beantworten Sie zehn der folgenden Fragen:

a) Kann man bei einem Quadtree die Himmelsrichtungen N, S, E und W hinzufügen, um einen höheren Fanout zu bekommen?

b) Beschreiben Sie eine Möglichkeit, den kd-Baum so zu erweitern, dass mehrere Datenpunkte pro Knoten abgespeichert werden können. Erklären Sie, wie dann eingefügt, gesucht und gelöscht wird.

c) Im R-Baum müssen die MBRs aller Kindknoten komplett im MBR des Elternknotens liegen. Könnte man diese Forderung auch abschwächen und sagen, die Kind-MBRs müssen das Eltern-MBR nur schneiden?

d) Wie würden Sie die Pivot-Objekte für den GH-Tree auswählen?

e) Beschreiben Sie, wie Sie Einfügen und Löschen für den GH-Tree implementieren würden.

f) Beschreiben Sie anhand der Illustrationen im Script genau, wie die Suche nach einem Schlüssel bei Erweiterbarem Hashing funktioniert.

g) Was ist für leseintensive Workloads bei Linarem Hashing besser: kontrollierte Splitting mit niedrigem β_s, mit hohem β_s oder unkontrolliertes Splitting?

h) Wann ist eine Aggregationsfunktion (im Kontext von Top-K Algorithmen) monoton?

i) Ist der Threshold Algorithmus immer besser als Fagins Algorithmus? Welches Maß für "besser" benutzen Sie?

j) Sie haben zweidimensionale Datenpunkte in einer Datenbank gespeichert, in der Sie keine Nächste-Nachbar-Suche durchführen können. Dafür können Sie Anfragen vom Typ "Gib mir alle Punkte in folgendem Rechteck" ausführen. Wie können Sie auf dieser Datenbank trotzdem effizient die Skyline berechnen?

k) Für jeden der Buchstaben ACID: Welche Komponenten im Datenbanksystem wird nicht gebraucht, wenn der Buchstabe nicht mehr garantiert wird?

l) Beschreiben Sie, wie sich das Vorhandensein von sekundären Indexen auf die Transaktionsverwaltung und die möglichen Anfragen auswirkt.

m) Bei der Kombination steal und ¬force, geben Sie an wie viele Redo- und wie viele Undo-Aktionen ausgeführt werden müssen.

n) Der Log-Ringpuffer fasst 2^{15} Einträge. Wie viele Transaktionen können gleichzeitig im Datenbanksystem ablaufen?

o) Beschreiben Sie ein Deadlock-Vermeidungsverfahren, bei dem jeder Transaktion eine Priorität $\in \mathbb{N}$ zugeordnet ist und indem bei der Vermeidung eines Deadlocks die Summe der Prioritäten der neugestarteten TA minimiert werden soll.

p) Erweitern Sie das in der Vorlesung vorgestellte Zeitstempelverfahren um mehrere Versionen eines Datensatzes. Dabei werden Leseoperationen erweitert, sodass mit $r(x, t)$ angegeben wird, dass das Objekt x im Zustand vom Zeitpunkt t gelesen wird. Wie sieht ein Schreibzugriff aus? Wie viele Versionen eines Objektes müssen gespeichert sein? Wird so noch ACID garantiert?