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

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe22

Alice Rey, Maximilian {Bandle, Schüle}, Michael Jungmair (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss22/impldb/

Blatt Nr. 06

Hausaufgabe 1

Gegeben sei folgende Relation `Klausur` mit Schlüssel `MatrNr`:

MatrNr	Name	Note	Standort
10101	Philipp	1,0	München
10102	Magdalena	1,0	Garching
10103	Erik	1,0	Garching
10104	Josef	1,0	Garching
10105	Alex	1,0	Garching
10106	Maxmilian	1,0	München

Für eine verteilte Datenbank soll die Tabelle geeignet fragmentiert werden. Ziel ist, Namen mit Standort der Studenten lokal und die Noten getrennt abzuspeichern.

1) Fragmentieren Sie die Relation geeignet *vertikal*.
   a) Geben Sie das Schema für die zwei resultierenden Relationen $KlausurV_1$ und $KlausurV_2$ an. Unterstreichen Sie jeweils den Primärschlüssel.

      $KlausurV_1 : \{[\underline{MatrNr}, Note]\}, KlausurV_2\{[\underline{MatrNr}, Name, Standort]\}$

   b) Geben Sie in SQL-92 die zwei resultierenden Relationen `KlausurV1` und `KlausurV2` als Hilfstabellen (mittels `with`) an.

```
with KlausurV1 as (SELECT MatrNr,Note FROM Personen),
     KlausurV2 as (SELECT MatrNr,Name,Standort FROM Personen)
```

2) Die geeignetere der beiden resultierenden Relationen soll *horizontal* fragmentiert werden.
   a) Geben Sie das Prädikat der Selektion an, mit dem fragmentiert wird.

      Standort='Garching' oder Standort='München'

   b) Geben Sie in SQL-92 die zwei resultierenden Relationen `KlausurH1` und `KlausurH2` als Hilfstabellen (mittels `with`) an.

```
with KlausurH1 as(SELECT * FROM KlausurV2 WHERE Standort='Garching'),
     KlausurH2 as(SELECT * FROM KlausurV2 WHERE Standort<>'Garching')
```

3) Schreiben Sie eine SQL-Abfrage, die die Ursprungsrelation aus den Teilrelationen zusammensetzt.

```
select KlausurV2.*, KlausurV1.Note
from KlausurV1,
  (select * from KlausurH1 union select * from KlausurH2) as KlausurV2
where KlausurV1.MatrNr=KlausurV2.MatrNr
```

Hausaufgabe 2

Für die Rekonstruierbarkeit der Originalrelation R aus vertikalen Fragmenten $R_1, \ldots, R_n$ reicht es eigentlich, wenn Fragmente paarweise einen Schlüsselkandidaten enthalten. Illustrieren Sie, warum es also nicht notwendig ist, dass der Durchschnitt aller Fragmentschemata einen Schlüsselkandidaten enthält. Es muss also nicht unbedingt gelten

$$R_1 \cap \cdots \cap R_n \supseteq \kappa,$$

wobei κ ein Schlüsselkandidat aus R ist.

Geben Sie ein anschauliches Beispiel hierfür – am besten bezogen auf unsere Beispiel-Relation *Professoren*.

Lsg: Vgl. Übungsbuch, Primärschlüssel unterstrichen: RaumF: $\{[\underline{Raum}, Fakultaet]\}$
Professoren: $\{[\underline{PersNr}, Name, Raum]\}$, ProfessorenR: $\{[\underline{PersNr}, Rang]\}$
ProfessorenF: $\{[\underline{PersNr}, Name, Rang, Raum, Fakultaet]\}$
ProfessorenF = RaumF $\bowtie_{\text{Raum=Raum}}$ (Professoren $\bowtie_{\text{PersNr=PersNr}}$ (ProfessorenR))

Hausaufgabe 3

Zeigen Sie, dass die *write-all / read-any* Methode zur Synchronisation replizierter Daten einen Spezialfall der *Quorum-Consensus*-Methode darstellt.

- Für welche Art von Workloads eignet sich dieses Verfahren besonders gut?
- Wie werden Stimmen zugeordnet um *write-all / read-any* zu simulieren?
- Wie müssen die Quoren Q_w und Q_r vergeben werden?

Dieses Verfahren fordert einen sehr großen Aufwand beim Schreiben, aber nur minimalen Aufwand beim Lesen. Daher eignet es sich besonders gut für Workloads in denen wesentlich mehr Daten gelesen als geschrieben werden. Dies entspricht z.B. analytischen Anfragen.
Siehe Übungsbuch.

Hausaufgabe 4

Um Ausfallsicherheit zu garantieren, ist ein Datenwert 'A' auf vier Rechnern verteilt. Jeder Rechner hält dabei eine vollständige Kopie von 'A'. Um Konsistenz zu garantieren, wird das Quorum-Consensus-Verfahren eingesetzt. Dabei ist jedem Rechner ein Gewicht $w_i(A)$ wie folgt zugewiesen:

Rechner	Kopie	Gewicht
R_1	A_1	3
R_2	A_2	1
R_3	A_3	2
R_4	A_4	2

Das Lesequorum ist $Q_r(A) = 4$ und das Schreibquorum is $Q_w(A) = 5$.

a) Geben Sie **alle** Lesemöglichkeiten für eine Transaktion auf dem Datum 'A' nach dem Quorum-Consensus-Protokoll an.
A_1,A_2
A_1,A_2,A_3
A_1,A_2,A_3,A_4
A_1,A_2,A_4
A_1,A_3
A_1,A_3,A_4
A_1,A_4
A_2,A_3,A_4
A_3,A_4

b) Geben Sie **alle** Schreibmöglichkeiten für eine Transaktion auf dem Datum 'A' nach dem Quorum-Consensus-Protokoll an.
A_1,A_2,A_3
A_1,A_2,A_3,A_4
A_1,A_2,A_4
A_1,A_3
A_1,A_3,A_4
A_1,A_4
A_2,A_3,A_4

c) Zeigen Sie für dieses Beispiel, dass, während eine Transaktion T_1 ein Schreibquorum auf A hält, es für andere Transaktionen T_x nicht möglich ist, ein Lesequorum für A zu bekommen.
Zum Schreiben muss die Transaktion T_1 mindestens Kopien mit einem Gesamtgewicht von 5 gesperrt haben. Insgesamt haben alle Kopien zusammen das Gewicht 8. Somit können maximal Kopien mit einem Gewicht von zusammen 3 übrig bleiben, womit das Lesequorum von 4 nicht mehr erfüllt werden kann.

Hausaufgabe 5

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	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

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

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: $3 \cdot 8 + 1 = 25$