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

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe25

Alice Rey, Maximilian Reif, Tobias Goetz (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss25/impldb/

Blatt Nr. 06

Hausaufgabe 1

Gehen Sie von folgender kombinierter Fragmentierung der in Abbildung 1 dargestellten Relation *Professoren* aus:

Professoren						
PersNr	Name	Rang	Raum	Fakultät	Gehalt	Steuerklasse
2125	Sokrates	C4	226	Philosophie	85000	1
2126	Russel	C4	232	Philosophie	80000	3
2127	Kopernikus	C3	310	Physik	65000	5
2133	Popper	C3	52	Philosophie	68000	1
2134	Augustinus	C3	309	Theologie	55000	5
2136	Curie	C4	36	Physik	95000	3
2137	Kant	C4	7	Philosophie	98000	1

Abbildung 1: Beispielausprägung der um drei Attribute erweiterten Relation *Professoren*

1. Zuerst erfolgt eine vertikale Fragmentierung in

$$\begin{aligned} \text{ProfVerw} &:= \Pi_{PersNr,Name,Gehalt,Steuerklasse}(\text{Professoren}) \\ \text{Profs} &:= \Pi_{PersNr,Name,Rang,Raum,Fakultt}(\text{Professoren}) \end{aligned}$$

2. Das Fragment Profs wird weiter horizontal fragmentiert in

$$\begin{aligned} \text{TheolProfs} &:= \sigma_{\text{Fakultät = 'Theologie'}}(\text{Profs}) \\ \text{PhysikProfs} &:= \sigma_{\text{Fakultät = 'Physik'}}(\text{Profs}) \\ \text{PhiloProfs} &:= \sigma_{\text{Fakultät = 'Philosophie'}}(\text{Profs}) \end{aligned}$$

Übersetzen Sie aufbauend auf dieser Fragmentierung die folgende SQL-Anfrage in die kanonische Form.

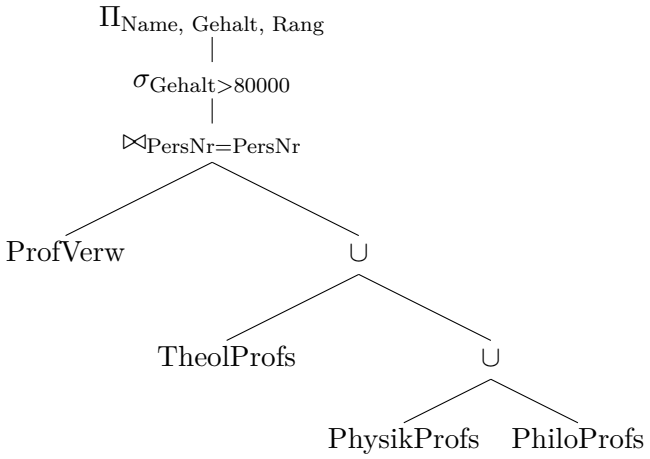
```
select Name, Gehalt Rang
from Professoren
where Gehalt > 80000;
```

Optimieren Sie diesen kanonischen Auswertungsplan durch Anwendung algebraischer Transformationsregeln (Äquivalenzen).

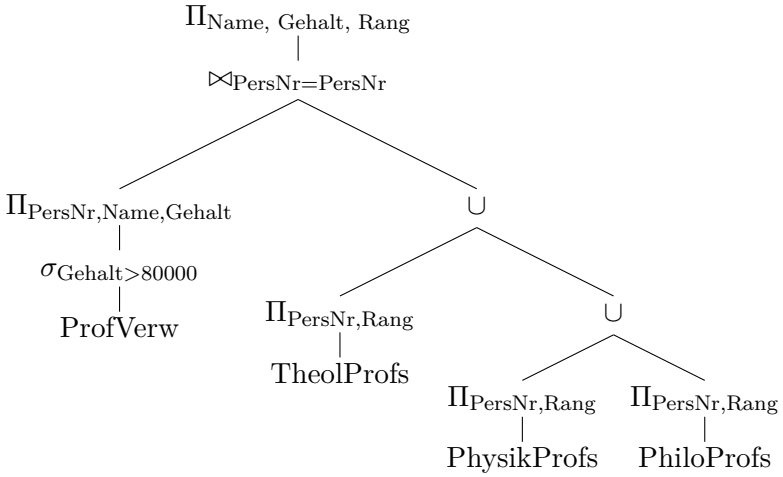
Vgl. Übungsbuch.

$$\Pi_{\text{Name, Gehalt, Rang}}(\Pi_{\text{PersNr,Name, Gehalt}}(\sigma_{\text{Gehalt} > 80000}(\text{ProfVerw})) \bowtie_{\text{PersNr=PersNr}} (\Pi_{\text{PersNr, Rang}}(\text{TheolProfs}) \cup \Pi_{\text{PersNr, Rang}}(\text{PhysikProfs}) \cup \Pi_{\text{PersNr, Rang}}(\text{PhiloProfs})))$$

Der Baum zum kanonischen Auswertungsplan sieht wie folgt aus:



In einem ersten Schritt verschiebt man die Selektion näher an die Datenquellen, die sich nur auf *ProfVerw* bezieht. Anschließend versuchen wir, die Zwischenergebnisse so klein wie möglich zu halten, indem wir zusätzliche Projektionen einfügen. Das Attribut *Name* ist redundant in beiden vertikalen Fragmenten enthalten und wird nicht benötigt.



Hausaufgabe 2

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 3

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))

Gruppenaufgabe 4

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: $3 \cdot 8 + 1 = 25$

Gruppenaufgabe 5

Überlegen Sie sich, welche Tupel bei der Anwendung des bloomfilterbasierten Joins in Abbildung 2 übertragen werden. Markieren Sie insbesondere, welche Tupel übertragen werden, obwohl sie keinen Joinpartner finden (sog. *false drops*). Wie kann die Anzahl dieser *false*

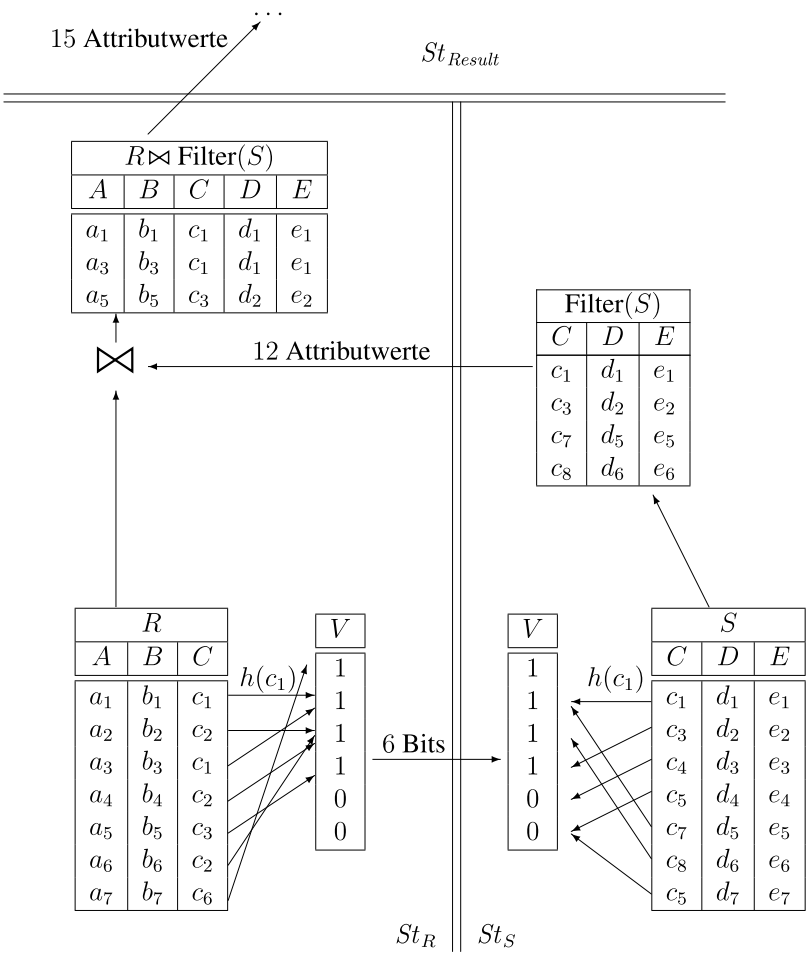


Abbildung 2: Beispiel einer verteilten Joinbearbeitung mit Bloomfilter.

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.