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).

## 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.
   b) Geben Sie in SQL-92 die zwei resultierenden Relationen `KlausurV1` und `KlausurV2` als Hilfstabellen (mittels `with`) an.
2) Die geeignetere der beiden resultierenden Relationen soll *horizontal* fragmentiert werden.
   a) Geben Sie das Prädikat der Selektion an, mit dem fragmentiert wird.
   b) Geben Sie in SQL-92 die zwei resultierenden Relationen `KlausurH1` und `KlausurH2` als Hilfstabellen (mittels `with`) an.
3) Schreiben Sie eine SQL-Abfrage, die die Ursprungsrelation aus den Teilrelationen zusammensetzt.

## 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 $\kappa$ ein Schlüsselkandidat aus $R$ ist.

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

## 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 | |
| 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.

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

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

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

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.

## 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 drops* verringert werden? Welche Eigenschaften sollte die Hashfunktion $h(c)$ die bei dieser Joinbearbeitung verwendet wird erfüllen?

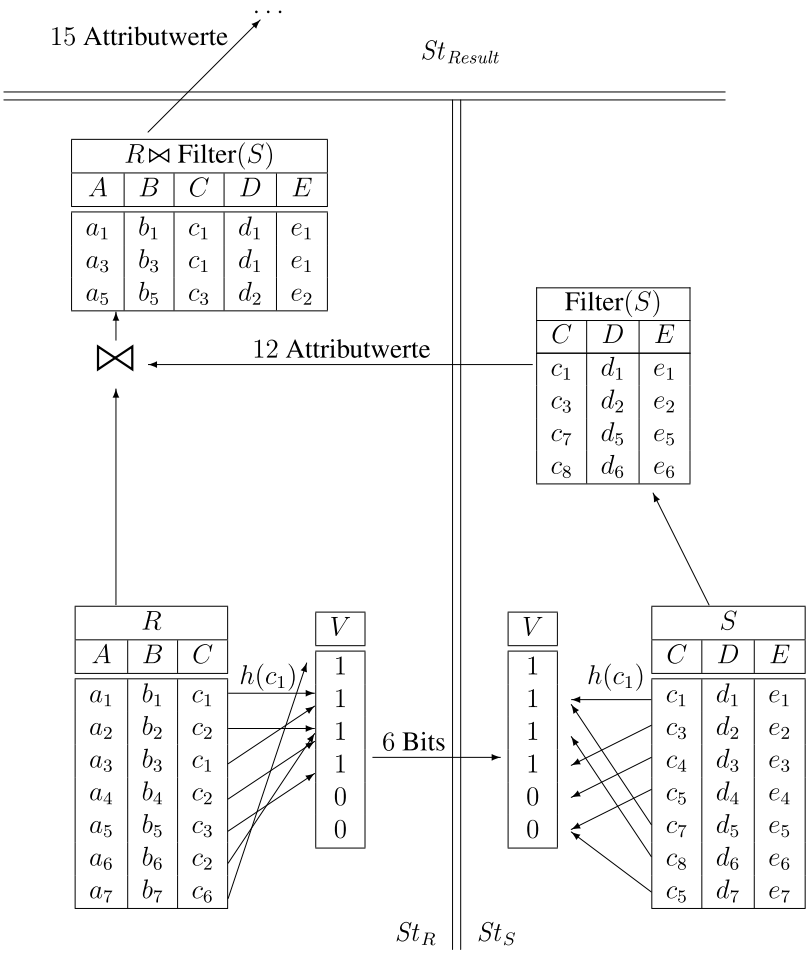


Abbildung 2: Beispiel einer verteilten Joinbearbeitung mit Bloomfilter.