

## Übung zur Vorlesung *Einsatz und Realisierung von Datenbanksystemen* im SoSe19

Maximilian {Bandle, Schüle} (i3erdb@in.tum.de)  
<http://db.in.tum.de/teaching/ss19/impldb/>

### Blatt Nr. 05

#### 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

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

2. Das Fragment *Profs* wird weiter horizontal fragmentiert in

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

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

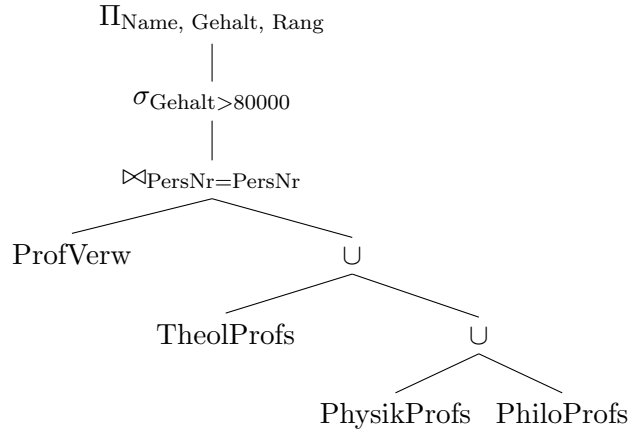
```
select Name, Gehalt Rang
from Professoren
where Gehalt > 80000;
```

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

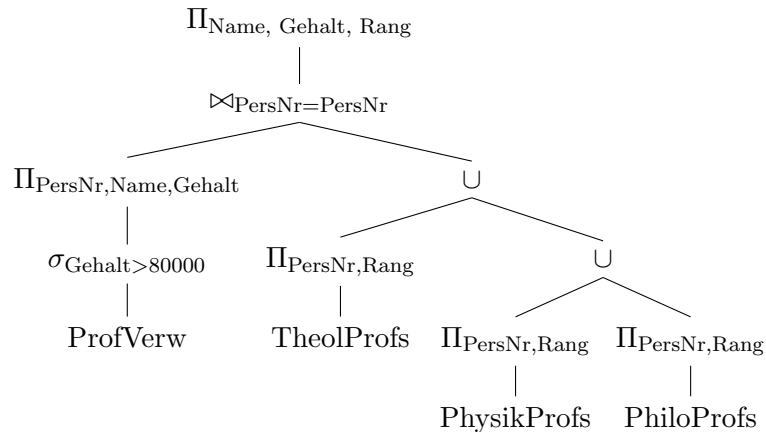
Vgl. Übungsbuch.

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

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

Für die Rekonstruierbarkeit der Originalrelation  $R$  aus vertikalen Fragmenten  $R_1, \dots, R_n$  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

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

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

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

Siehe Übungsbuch.

### 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

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?

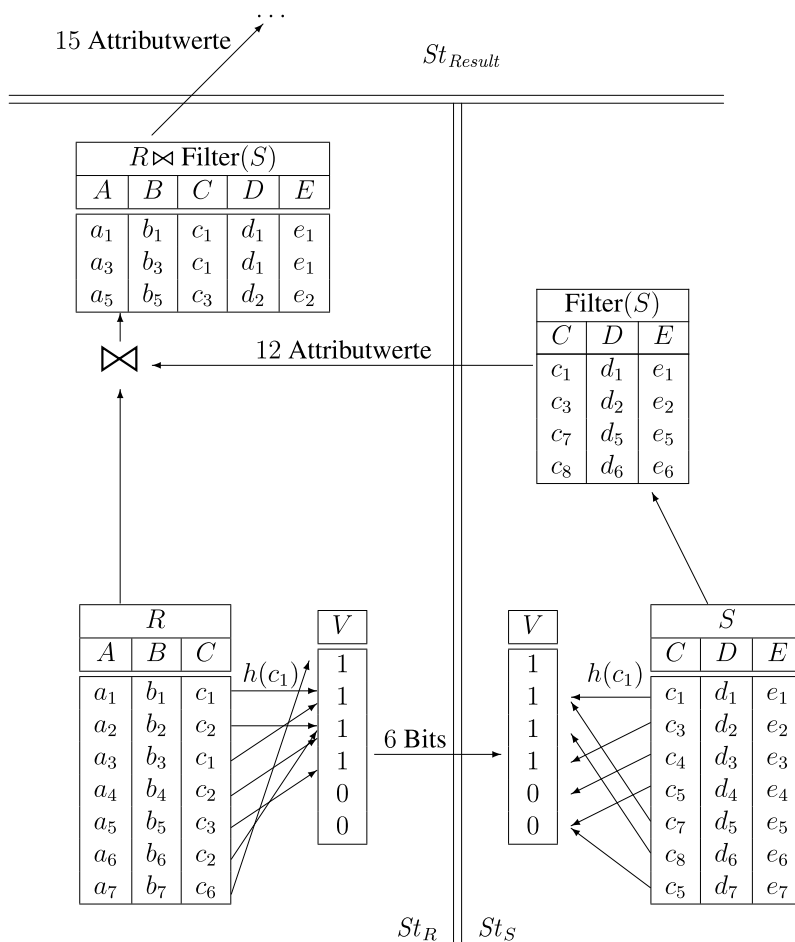


Abbildung 2: Beispiel einer verteilten Joinbearbeitung mit Bloomfilter.

- 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.

### Hausaufgabe 6

Zeigen Sie, dass die Suche in einem Chord-Overlaynetzwerk durch die Nutzung der FingerTabellen in maximal logarithmisch vielen Schritten zur Größe des Zahlenrings (bzw. der Anzahl der Stationen) durchgeführt werden kann. Verwenden Sie die Suche nach K57 beginnend an Station P11 (siehe Abbildung 3) zur Illustration.

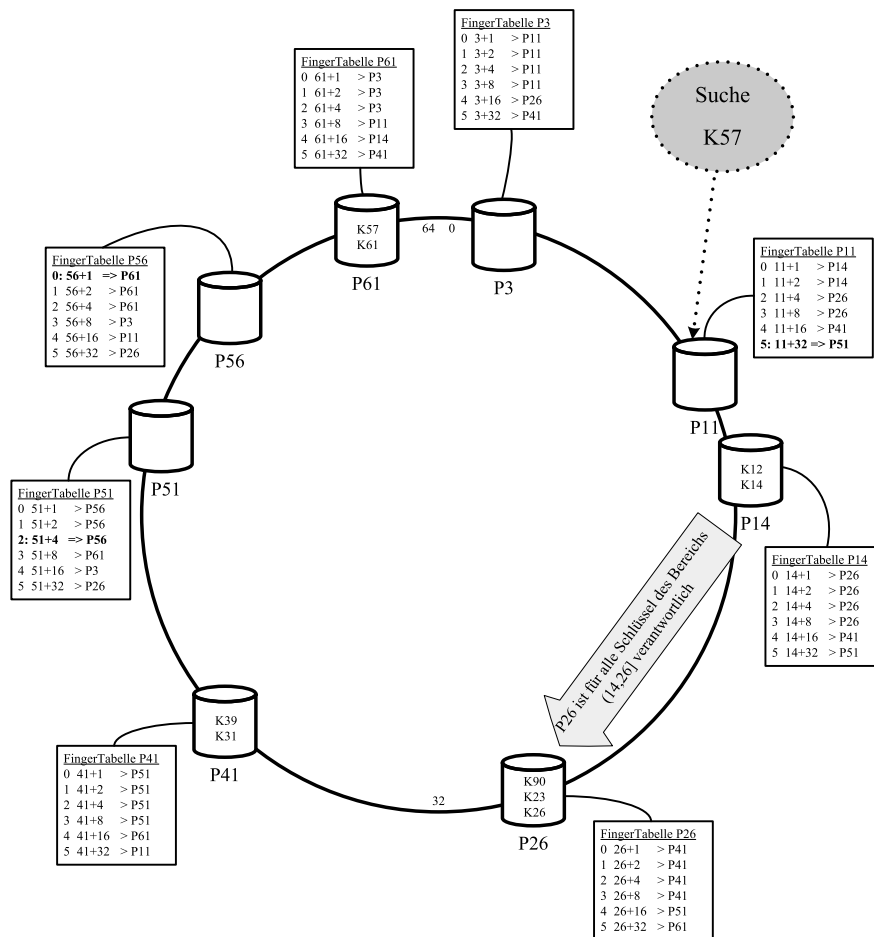


Abbildung 3: Beispiel eines Chord-Overlaynetzwerks.

Vgl. Übungsbuch.

**z.z.:** Mit jedem Sprung halbiert sich die Anzahl der Stationen.

**Beweis durch vollständige Induktion** über die Anzahl der Sprünge  $i$ .

**Induktionsanfang:** Sind wir direkt vor dem gesuchten Peer, also  $i = 0$ , müssen wir nicht mehr springen, es verbleiben  $\ln(1) = \ln(2^0) = 0$  Sprünge.

**Induktionsschritt:** Vom  $i$ -ten zum  $i + 1$ -Sprung halbiert sich die Anzahl verbleibender Stationen von  $2^{n-i}$  nach  $2^{n-i-1}$

*Induktionsvoraussetzung:* Nach  $i$  Sprüngen reduziert sich die Anzahl verbleibender Stationen auf  $2^{n-i}$ .

Wir wissen, dass sich innerhalb eines Intervalls  $[p + 2^i, p + 2^{i+1})$  genau  $2^i$  Stationen befinden, da wir sonst bereits zum nächsten Peer gesprungen wären. Damit haben wir bereits den Abstand auf  $2^{n-i}$  Stationen halbiert (Induktionsvoraussetzung). Jetzt springen wir in das Intervall  $[(p + 2^i) + 2^{i-1}, (p + 2^i) + 2^i)$  mit  $2^{i-1}$  möglichen Stationen. Für den maximalen Abstand zum Ziel-Peer verbleiben nach dem  $i + 1$ -ten Sprung maximal  $2^{n-(i+1)}$  Stationen, wenn wir  $n - (i + 1)$  mal springen. Damit halbiert sich die Zahl verbleibender Sprünge auf  $2^{n-(i+1)} = 2^{n-i-1}$

## Hausaufgabe 7

Skizzieren Sie die Vorgehensweise beim Hinzufügen eines neuen Peers im Chord Netzwerk. Als Beispiel nehmen Sie die Hinzunahme eines Peers P33 in dem Beispiel-Netzwerk aus Abbildung 3.

Vgl Übungsbuch: Zuerst kontaktiert der neue Peer einen bekannten Peer im Chord Netzwerk. Dieser leitet ihn mittels der Hash-Funktion an den direkten Vorgänger des neuen Peers weiter. Von diesem übernimmt er die Fingertabelle und sein direkter Nachfolger wird ihm bekannt. Von diesem wiederum übernimmt er alle relevanten Daten, für die er nun zuständig ist.

Nach dem Einfügen eines neuen Peers Pk in ein Chord-Netzwerk müssen die Fingertabellen anderer Peers, „die bislang Pj referenziert haben, jetzt möglicherweise auf Pk geändert werden“<sup>a</sup>. Dies passiert allerdings nicht beim Einfügen, sondern beim Stabilisieren<sup>b</sup>. Der Stabilisierungsalgorithmus wird in 30s Abständen aufgerufen<sup>c</sup>, bzw. beim Einfügen eines neuen Peers<sup>d</sup>. Beim Einfügen wird nur der direkte Vorgänger über seinen neuen Nachfolger informiert.

---

<sup>a</sup>Kemper, Eickler: Datenbanksysteme, 10. Auflage

<sup>b</sup>[https://en.wikipedia.org/wiki/Chord\\_\(peer-to-peer\)#Pseudocode](https://en.wikipedia.org/wiki/Chord_(peer-to-peer)#Pseudocode)

<sup>c</sup><http://graal.ens-lyon.fr/~abenoit/reso06/papier/chord1.pdf>

<sup>d</sup>[http://sarwiki.informatik.hu-berlin.de/Chord#Eintritt\\_neuer\\_Knoten](http://sarwiki.informatik.hu-berlin.de/Chord#Eintritt_neuer_Knoten)