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

# Übung zur Vorlesung *Grundlagen: Datenbanken* im WS23/24

Christoph Anneser, Michael Jungmair, Stefan Lehner, Moritz Sichert, Lukas Vogel
(gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2324/grundlagen/

**Blatt Nr. 11**

## Hausaufgabe 1

a) Was ist ein Equi-Join?

b) Bei welchen Join-Prädikaten ($<, =, >$) kann man sinnvoll einen Hashjoin einsetzen?

c) Gegeben die Relation Profs = {PersNr, Name} und Raeume = {PersNr, RaumNr}.

1) Skizzieren Sie eine geschickte Möglichkeit, den Equi-Join Profs ⋈ Raeume durchzuführen.

2) In welchem Fall wäre selbst ein Ausdruck wie

$$\text{Profs} \bowtie_{\text{Profs.Persnr} < \text{Raeume.PersNr}} \text{Raeume}$$

effizient auswertbar?

d) Der Student Maier hat einen Algorithmus gefunden, der den Ausdruck $A \times B$ in einer Laufzeit von $O(|A|)$ materialisiert. Was sagen Sie Herrn Maier?

**Lösung:**

a) Ein Equi-Join hat eine Äquivalenz als Joinbedingung, etwa die Gleichheit zweier Attribute.

b) Ein Hash Join bietet sich nur für Equi-Joins an, da lediglich ein Join-Partner mit gleichem Attributwert effizient auffindbar ist. Das Finden eines Partners, dessen Attributwert beispielsweise kleiner sein soll kann mittels Hashing i.A. nicht effizient bearbeitet werden.

c) 1) Offenbar ist das Joinattribut gerade der Primärschlüssel, womit von der Existenz eines Indexes ausgegangen werden kann. Somit bietet sich ein Index-basierter Join an, etwa dadurch, dass die eine Relation Element für Element abgearbeitet wird, während Joinpartner aus der anderen Relation mittels des Indexes gefunden werden.

2) Falls der Index sortiert ist, dies wäre etwa bei einem B-Baum der Fall. Dadurch liegen Joinpartner zumindest nacheinander im Index, anders als bei einer Implementierung des Indexes mittels Hash.

d) Dies ist mit Sicherheit nicht der Fall, da ein Algorithmus keine bessere Komplexitätsklasse haben kann als sein Ergebnis wächst. Mit anderen Worten, $A \times B$ hat eine Ergebnisgröße von $|A| \cdot |B|$ und dieses Ergebnis kann sicher nicht schneller als in $O(|A| \cdot |B|)$ materialisiert werden.

## Hausaufgabe 2

Gegeben sei die folgende SQL-Anfrage:

```
select distinct a.PersNr, a.Name
from Assistenten a, Studenten s, pruefen p
where s.MatrNr = p.MatrNr
  and a.Boss = p.PersNr
  and s.Name = 'Jonas';
```

Geben Sie die kanonische Übersetzung dieser Anfrage in die relationale Algebra an. Verwenden Sie zur Darstellung des relationalen Algebraausdrucks die Baumdarstellung.

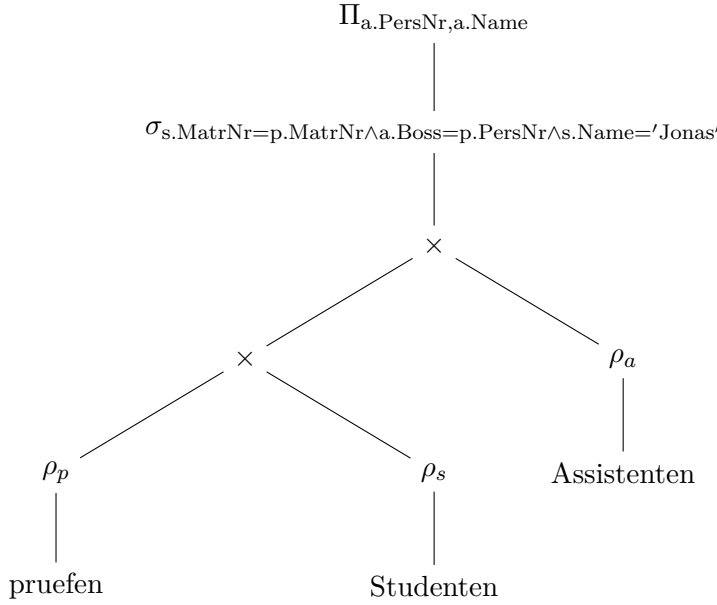
Optimieren Sie Ihren relationalen Algebraausdruck logisch. Gehen Sie dabei von **realistischen** Kardinalitäten für die relevanten Relationen aus.

Verwenden Sie hierfür die folgenden aus der Vorlesung bekannten Optimierungstechniken:

- Aufbrechen von Selektionen
- Verschieben von Selektionen nach "unten" im Plan
- Zusammenfassen von Selektionen und Kreuzprodukten zu Joins
- Bestimmung der Joinreihenfolge

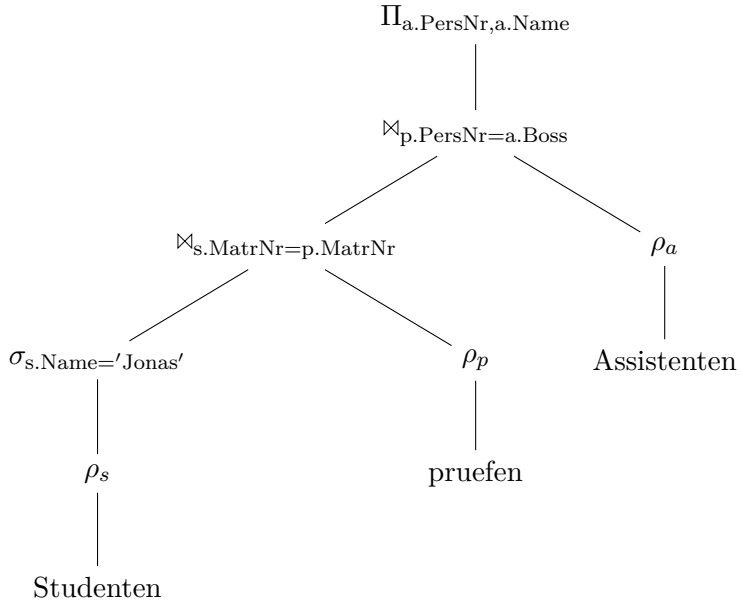
**Lösung:**

Kanonische Übersetzung:




realistische Kardinalitäten: nur ein Student mit dem Namen 'Jonas', viel mehr Assistenten, deshalb Studenten zuerst, dann über pruefen mit Assistenten joinen

logische Optimierung: Selektionen ganz nach unten, Joins statt Kreuzprodukte & in richtiger Reihenfolge




## Hausaufgabe 3

Gegeben sind die beiden Relationenausprägungen:

| $R$ | |
|---|---|
| | A |
| ... | 0 |
| ... | 5 |
| ... | 7 |
| ... | 8 |
| ... | 8 |
| ... | 10 |
| ⋮ | ⋮ |

| $S$ | |
|---|---|
| B | |
| 5 | ... |
| 6 | ... |
| 7 | ... |
| 8 | ... |
| 8 | ... |
| 11 | ... |
| ⋮ | ⋮ |

Werten Sie den Join $R \bowtie_{R.A=S.B} S$ mithilfe des Nested-Loop- sowie des Sort/Merge-Algorithmus aus. Machen Sie deutlich, in welcher Reihenfolge die Tupel der beiden Relationen verglichen werden und kennzeichnen Sie die Tupel, die in die Ergebnismenge übernommen werden. Vervollständigen Sie hierzu die beiden folgenden Tabellen:

| | | $S.B$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 6 | 7 | 8 | 8 | 11 |
| $R.A$ | 0 | 1 | 2 | 3 | | | |
| | 5 | | | | | | |
| | 7 | | | | | | |
| | 8 | | | | | | |
| | 8 | | | | | | |
| | 10 | | | | | | |

Nested-Loop-Join

| | | $S.B$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 6 | 7 | 8 | 8 | 11 |
| $R.A$ | 0 | 1 | | | | | |
| | 5 | 2✔ | | | | | |
| | 7 | | | | | | |
| | 8 | | | | | | |
| | 8 | | | | | | |
| | 10 | | | | | | |

Sort/Merge-Join

**Lösung:**

| | | *S.B* | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 6 | 7 | 8 | 8 | 11 |
| *R.A* | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| | 5 | 7✔ | 8 | 9 | 10 | 11 | 12 |
| | 7 | 13 | 14 | 15✔ | 16 | 17 | 18 |
| | 8 | 19 | 20 | 21 | 22✔ | 23✔ | 24 |
| | 8 | 25 | 26 | 27 | 28✔ | 29✔ | 30 |
| | 10 | 31 | 32 | 33 | 34 | 35 | 36 |

Nested-Loop-Join

| | | *S.B* | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 6 | 7 | 8 | 8 | 11 |
| *R.A* | 0 | 1 | | | | | |
| | 5 | 2✔ | 3 | | | | |
| | 7 | | 4 | 5✔ | | | |
| | 8 | | | 6 | 7✔ | 10✔ | |
| | 8 | | | | 8✔ | 11✔ | |
| | 10 | | | | 9 | 12 | 13 |

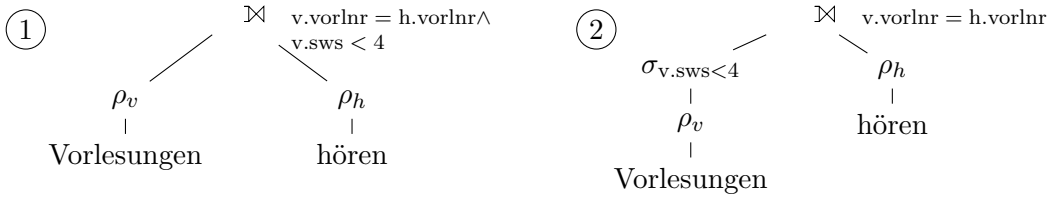
Sort/Merge-Join

Ausführliche Lösung:
`http://www-db.in.tum.de/teaching/ws1415/grundlagen/Loesung11_sort_merge_join.pdf`

## Hausaufgabe 4

*Klausuraufgabe aus dem WiSe 2018/19:*

Gegeben seien die beiden folgenden Algebraausdrücke in Operatorbaumdarstellung:




Sind die beiden Algebraausdrücke äquivalent? Begründen Sie!

**Lösung:**

Die Ausdrücke sind nicht äquivalent, da die beiden Bäume unterschiedliche Ergebnisse liefern.

Die Selektion filtert Vorlesungen aus, während der left outer join Tupel der linken Seite, die die Join-Bedingung nicht erfüllen (also z.B. von der Selektion ausgefiltert worden wären) mit null-Werten für die rechte Seite in die Ergbnismenge aufnimmt.

Mit der Beispielausprägung enthält das Ergebnis des linken Baumes z.B. die Vorlesung Ethik, der rechte Baum aber nicht.