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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS13/14

Henrik Mühe (muehe@in.tum.de)
http://www-db.in.tum.de/teaching/ws1314/dbsys/exercises/

Blatt Nr. 13

Hausaufgabe 1

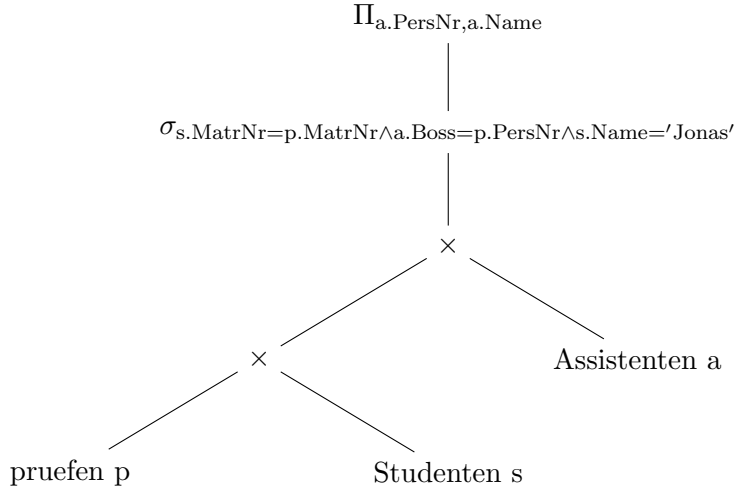
Gegeben sei die folgende SQL-Anfrage:

```
select 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.

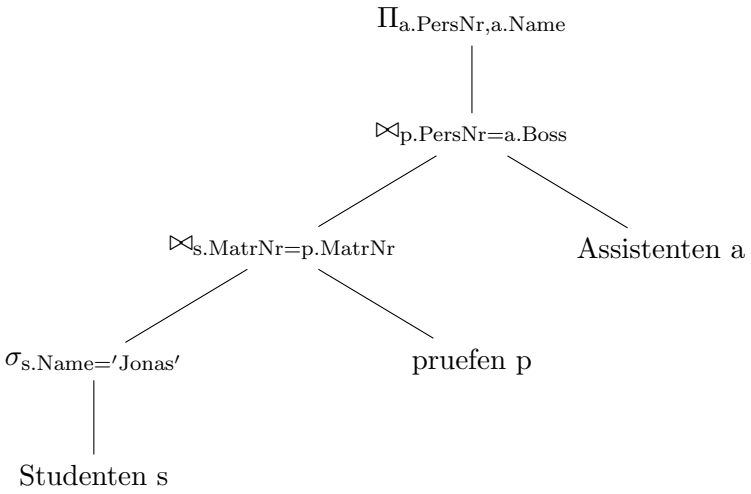
Optimieren Sie Ihren relationalen Algebraausdruck logisch. Gehen Sie dabei von **realistischen** Kardinalitäten für die relevanten Relationen aus.

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 2

(a) Was ist ein Equijoin?

(b) Was spricht gegen die Verwendung des Hashjoin-Verfahrens bei einem Join, der etwa ein <-Zeichen im Prädikat enthält?

(c) Gegeben die Relation $Profs(\underline{PersNr}, Name)$ und $Raeume(\underline{PersNr}, RaumNr)$.

1) Skizzieren Sie eine geschickte Möglichkeit, den Equijoin $Profs \bowtie Raeume$ durchzuführen.

2) Wieso ist dies hier „angenehm“ durchführbar?

3) In welchem Fall wäre selbst ein Ausdruck wie $Profs \bowtie_{Profs.Persnr<Raeume.PersNr} Raeume$ angenehm durchführbar?

(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?

(a) Ein Equijoin hat eine Äquivalenz als Joinbedingung, etwa die Gleichheit zweier Attribute.

(b) Ein Hash Join bietet sich nur für Equijoins 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) Angenehm weil wahrscheinlich Index vorhanden und Equijoin.

3) 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 $|A| * |B|$ und dieses Ergebnis kann sicher nicht schneller als in $O(|A| * |B|)$ materialisiert werden.

Hausaufgabe 3

Es sollen alle ca. 10 Milliarden Menschen in einer erweiterbaren Hashtabelle verwaltet werden. In jede Seite passen ca. 200 Einträge, durchschnittlich sind die Seiten halb voll. Je Verweis werden 4 Byte benötigt, da die Musterlösung aus einer Zeit stammt, in der es defakto nur Maschinen mit 32 bit CPU Architektur gab. Wie viel Speicherplatz verbraucht das Verzeichnis mindestens?

Das Verzeichnis enthält die Verweise auf alle Seiten (= Buckets), in dem die Einträge gehalten werden. Da pro Seite durchschnittlich 100 Einträge Platz haben, benötigen wir insgesamt $10^{10}/100 = 10^8$ Seiten. Um 10^8 Seiten zu referenzieren benötigen wir mindestens $\log_2 10^8$ Bits. Da dies eine positive ganze Zahl sein muss, ist die Anzahl der benötigten Bits $\lceil \log_2 10^8 \rceil$. Hiermit können $2^{\lceil \log_2 10^8 \rceil}$ Verweise im Verzeichnis abgelegt werden, da die Anzahl der Verweise in einem Verzeichnis immer einer 2er-Potenz entspricht. Pro Verweis werden 4 Byte benötigt, so dass das Verzeichnis eine Größe von $2^{\lceil \log_2 10^8 \rceil} \cdot 4$ Byte, also ungefähr 512 MB hat.