



Übung zur Vorlesung *Grundlagen: Datenbanken* im WS15/16

Harald Lang, Linnea Passing (gdb@in.tum.de)

<http://www-db.in.tum.de/teaching/ws1516/grundlagen/>

Blatt Nr. 12

Hausaufgabe 1

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.

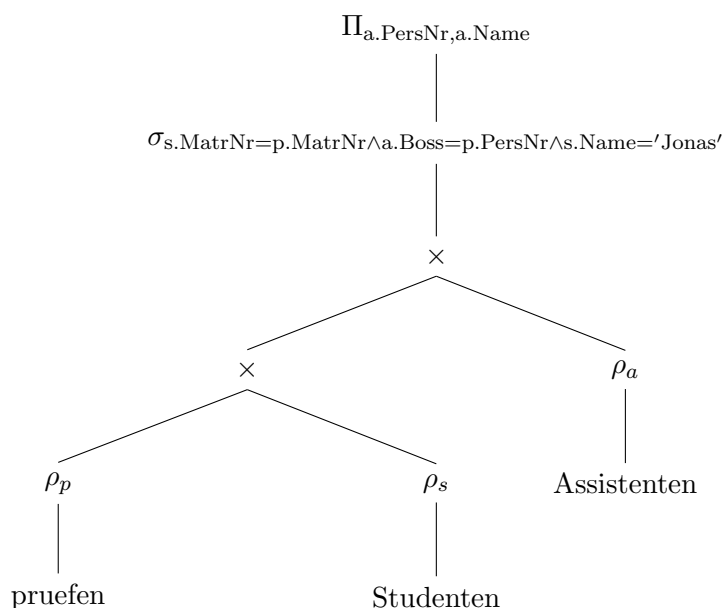
Optimieren Sie Ihren relationalen Algebraausdruck logisch. Gehen Sie dabei von **realistischen** Kardinalitäten für die relevanten Relationen aus.

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

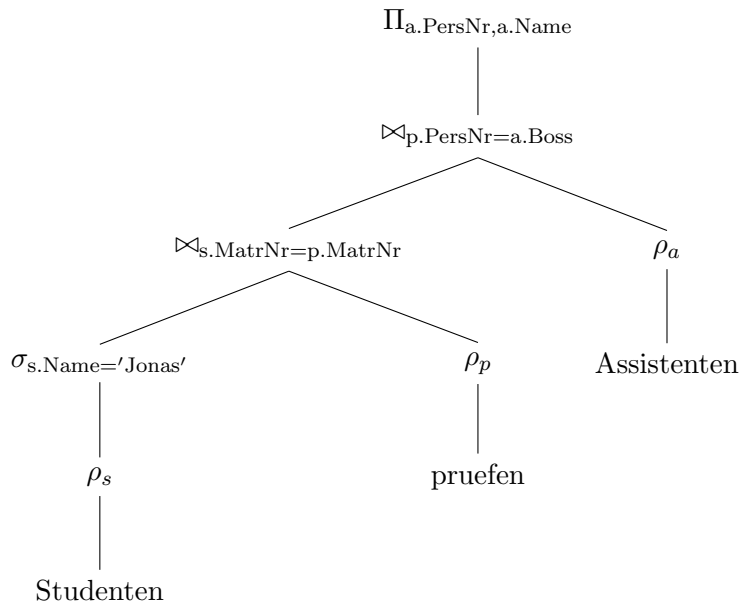
Lösung:

Kanonische Übersetzung:



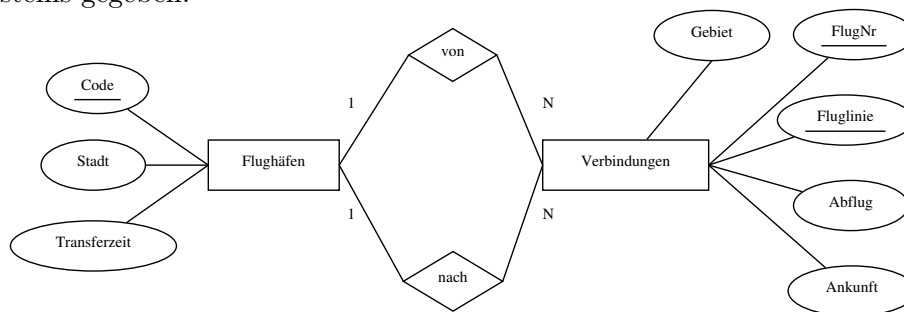
realistische Kardinalitäten: nur ein Student mit dem Namen 'Jonas', viel mehr Assistenten, deshalb Studenten zuerst, dann über pruefen mit Assistenten joinen

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



Hausaufgabe 2

Betrachten Sie die Anfrage „Finde alle Flüge von New-York nach Sydney mit einmaligem Umsteigen“. Dazu sei das nachfolgende (vereinfachte) ER-Diagramm eines Fluginformationsystems gegeben:



1. Geben Sie eine SQL-Query für die oben genannte Anfrage an.
2. Führen Sie die kanonische Übersetzung des SQL-Statements in die relationale Algebra durch.
3. Schätzen Sie die Relationsgrößen sinnvoll ab (z.B. so wie in den Beispielen der Vorlesung) und transformieren Sie den kanonischen Operatorbaum aus Teilaufgabe 2) zur optimalen Form. Wie haben sich die Kosten dabei geändert? (Kosten = Anzahl der Zwischenergebnistupel)

Lösung:

1. SQL-Anfrage:

```
SELECT DISTINCT v1.*, v2.*
```

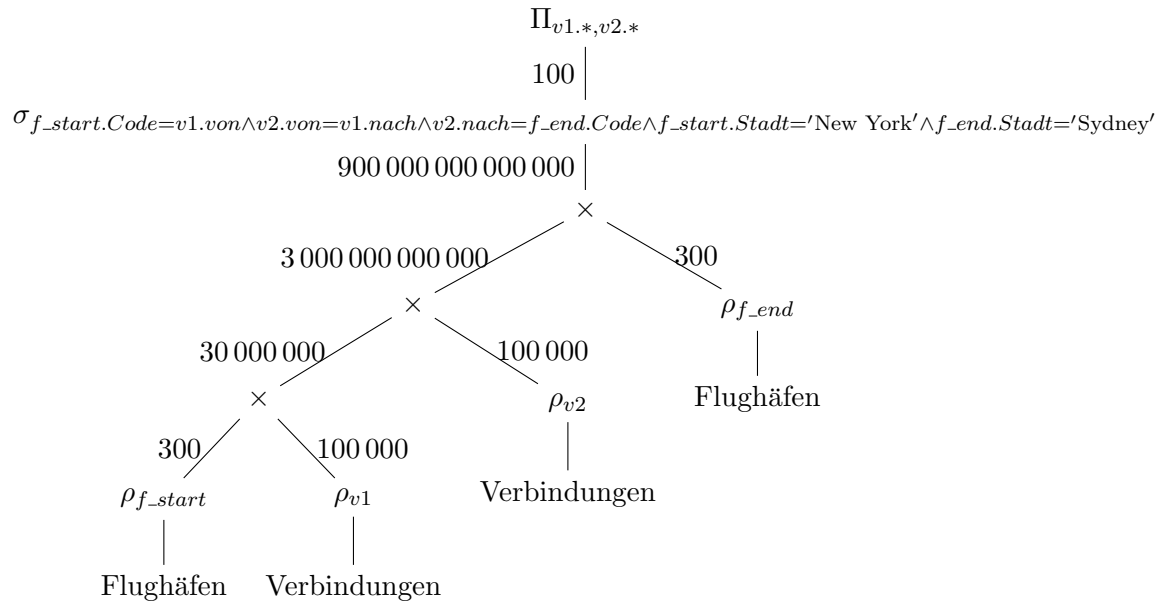
```
FROM Flughäfen f_start, Verbindungen v1, Verbindungen v2, Flughäfen f_end
```

```

WHERE f_start.Stadt = "New York"
  AND f_end.Stadt = "Sydney"
  AND v2.von = v1.nach
  AND v2.nach = f_end.Code
  AND f_start.Code = v1.von

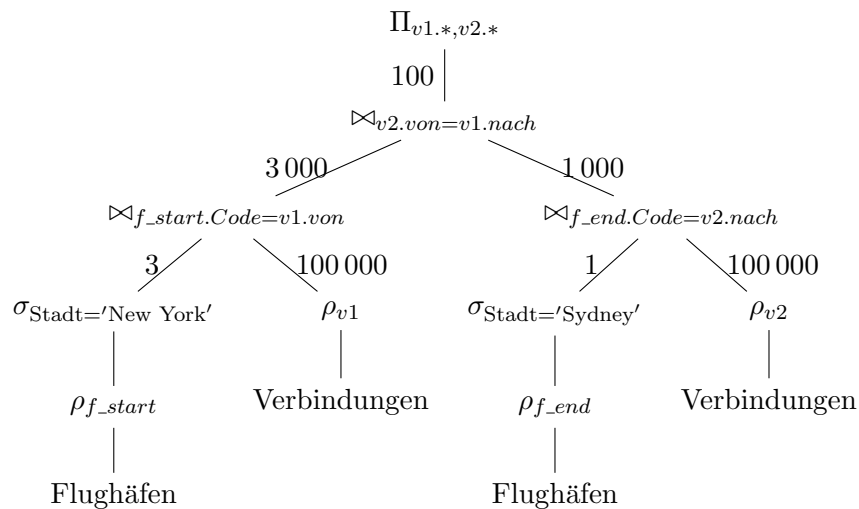
```

2. Kanonische Übersetzung:



Kosten > 900 Billionen Tupel.

3. Optimaler Ausführungsplan:



Kosten ca. 205 000 Tupel

Hausaufgabe 3

- Was ist ein Equi-Join?
- Bei welchen Join-Prädikaten ($<$, $=$, $>$) kann man sinnvoll einen Hashjoin einsetzen?

- (c) Gegeben die Relation $Prof s = \{\underline{PersNr}, Name\}$ und $Raeume = \{\underline{PersNr}, RaumNr\}$.
- 1) Skizzieren Sie eine geschickte Möglichkeit, den Equi-Join $Prof s \bowtie Raeume$ durchzuführen.
 - 2) In welchem Fall wäre selbst ein Ausdruck wie

$$Prof s \bowtie_{Prof s.Persnr < Raeume.PersNr} Raeume$$

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

Hausaufgabe 4

Gegeben sind die beiden Relationenausprägungen:

<i>R</i>	
	A
...	0
...	5
...	7
...	8
...	8
...	10
⋮	⋮

<i>S</i>	
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:

		<i>S.B</i>					
		5	6	7	8	8	11
<i>R.A</i>	0	1	2	3			
	5						
	7						
	8						
	8						
	10						

Nested-Loop-Join

		<i>S.B</i>					
		5	6	7	8	8	11
<i>R.A</i>	0	1					
	5	2✓					
	7						
	8						
	8						
	10						

Sort/Merge-Join

Lösung:

		<i>S.B</i>					
		5	6	7	8	8	11
<i>R.A</i>	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

		<i>S.B</i>					
		5	6	7	8	8	11
<i>R.A</i>	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