TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS19/20

Christoph Anneser, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1920/grundlagen/

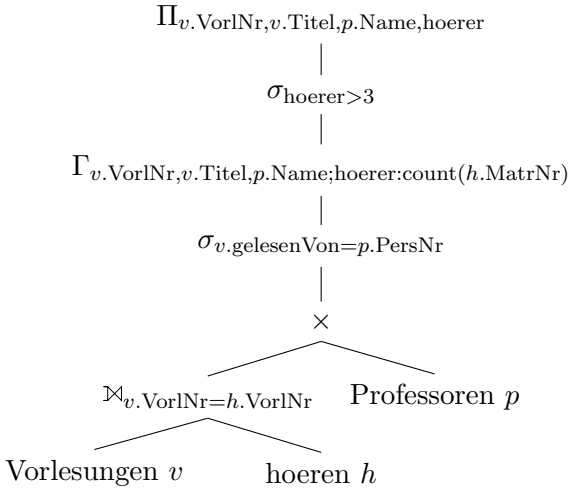
Blatt Nr. 11

Hausaufgabe 1

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

```
select v.VorlNr, v.Titel, p.Name, count(h.MatrNr) as hoerer
from
    Vorlesungen v left outer join
    hoeren h on (v.VorlNr = h.VorlNr),
    Professoren p
where
    v.gelesenVon = p.PersNr
group by v.VorlNr, v.Titel, p.Name
having count(h.MatrNr) > 3
```

Lösung:



Hausaufgabe 2

Sie verwenden ein Datenbanksystem, welches die folgenden Joinalgorithmen unterstützt:

- (Blockwise-)Nested-Loop-Join
- Index-Join
- Sort-Merge-Join
- Hash-Join

Geben Sie für jeden der vier Algorithmen ein Beispiel an, bei dem dieses Datenbanksystem diesen Algorithmus verwenden würde. Geben Sie dazu für jedes Beispiel ein Schema und eine SQL-Anfrage an. Begründen Sie, warum der gewählte Joinalgorithmus den anderen vorgezogen wird.

Lösung:

- (Blockwise-)Nested-Loop-Join: Schema: Relationen R und S. Die Attribute der Relationen sind beliebig.

  Anfrage: `select * from R, S`

  Begründung: Die Anfrage enthält ein Kreuzprodukt. Also ist die Kardinalität der Ergebnismenge zwangsläufig $|R \times S|$. Damit haben alle Joinalgorithmen die selbe (theoretische) Laufzeit. Um sich den unnötigen Overhead des Sortierens oder des Hashings zu sparen, verwendet die Datenbank daher einen Nested-Loop-Join.

- Index-Join: Schema: Relationen: $R : \{[\underline{a}]\}$ und $S : \{[b]\}$.

  Anfrage: `select * from R, S where a = b`

  Begründung: Da a in R ein Schlüssel ist, existiert bereits ein Index. Also kann dieser für einen effizienten Index-Join verwendet werden. Damit wird ebenfalls der Overhead des Sortierens oder Hashings gespart.

- Sort-Merge-Join: Schema: Relationen: $R : \{[a]\}$ und $S : \{[b]\}$.

  Anfrage: `select * from R, S where a = b order by a`

  Begründung: Da das Ergebnis ohnehin nach dem Attribut a sortiert werden muss, kann die Sortierung bereits vor dem Join durchgeführt werden, damit die Sortierung für einen Sort-Merge-Join verwendet werden kann.

- Hash-Join: Schema: Relationen: $R : \{[a]\}$ und $S : \{[b]\}$.

  Anfrage: `select * from R, S where a = b`

  Begründung: Hier wählt das Datenbanksystem den Hash-Join, da seine Komplexität die niedrigste ist.

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

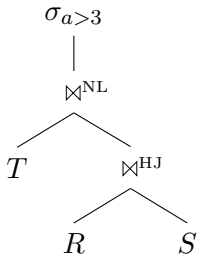
		S.B					
		5	6	7	8	8	11
R.A	0	1					
	5	2✔	3				
	7		4	5✔			
	8			6	7✔	10✔	
	8				8✔	11✔	
	10				9	12	13

Sort/Merge-Join

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

Hausaufgabe 4

Gegeben Sei der folgende physische Anfrageplan:



Hier steht $\bowtie^{\mathrm{NL}}$ für den Nested-Loop-Join und $\bowtie^{\mathrm{HJ}}$ für den Hash-Join. Geben Sie an, wie oft die `next`-Funktion von R aufgerufen wird, wenn ein Datenbanksystem diesen Anfrageplan mit dem Iteratorkonzept ausführt. Gehen Sie von folgenden Kardinalitäten aus: $|R| = 10$, $|S| = 20$ und $|T| = 5$. Gehen Sie davon aus, dass beim Nested-Loop-Join die linke Eingabe zuerst geöffnet wird und dass beim Hash-Join aus der linken Eingabe eine Hashtabelle erzeugt wird.

Lösung:

Aus dem linken Teilbaum des Hash-Joins wird eine Hashtabelle erzeugt, also wird für den Teilbaum inklusive des Hash-Joins die `next`-Funktion von R genau $|R|$ mal aufgerufen. Der rechte Teilbaum des Nested-Loop-Joins wird für jedes Tupel aus dem linken Teilbaum (also $|T|$ mal) ausgeführt. Insgesamt wird die `next`-Funktion von R also $|T| \cdot |R| = 50$ mal aufgerufen.