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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS21/22

Michael Jungmair, Josef Schmeißer, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2122/grundlagen/

Blatt Nr. 12

Hausaufgabe 1

Für einen Join-Baum T sei folgende Kostenfunktion gegeben

$$C_{out}(T) = \begin{cases} 0 & \text{falls } T \text{ eine Basisrelation } R_i \text{ ist} \\ |T| + C_{out}(T_1) + C_{out}(T_2) & \text{falls } T = T_1 \bowtie T_2 \end{cases}$$

Die Kardinalität sei dabei

$$|T| = \begin{cases} |R_i| & \text{falls } T \text{ eine Basisrelation } R_i \text{ ist} \\ (\prod_{R_i \in T_1, R_j \in T_2} f_{i,j}) |T_1||T_2| & \text{falls } T = T_1 \bowtie T_2 \end{cases}$$

Sei $p_{i,j}$ das Joinprädikat zwischen R_i und R_j, dann sei

$$f_{i,j} = \frac{|R_i \bowtie_{p_{i,j}} R_j|}{|R_i \times R_j|}$$

und die Kardinalität eines Join-Resultats ist $|R_i \bowtie_{p_{i,j}} R_j| = f_{i,j}|R_i||R_j|$.

Gegeben sei eine Anfrage über die Relationen R_1, R_2, R_3 und R_4 mit $|R_1| = 10, |R_2| = 20, |R_3| = 20, |R_4| = 10$. Die Selektivitäten der Joins seien $f_{1,2} = 0.01, f_{2,3} = 0.5, f_{3,4} = 0.01$, alle nicht gegebenen Selektivitäten sind offensichtlich 1 (Warum?). Berechnen Sie den optimalen (niedrigste Kosten) Join-Tree. Als Vereinfachung reicht es, wenn Sie nur Joins mit Prädikat und keine Kreuzprodukte betrachten.

Lösung:

Es ist kein Algorithmus angegeben. Aufgrund der geringen Anzahl von Relationen ist es möglich, die Kosten aller möglichen Join-Bäume zu berechnen und den kostengünstigsten auszuwählen (Bruteforce).

Zunächst gilt es zu überlegen, für welche Join-Bäume die Kosten tatsächlich zu berechnen sind.

Left-Deep:

$$((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4 \tag{1}$$
$$((R_4 \bowtie R_3) \bowtie R_2) \bowtie R_1 \tag{2}$$
$$((R_3 \bowtie R_2) \bowtie R_1) \bowtie R_4 \tag{3}$$
$$((R_3 \bowtie R_2) \bowtie R_4) \bowtie R_1 \tag{4}$$

Bushy:

$$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4) \tag{5}$$

Alle anderen Left Deep oder Bushy Trees enthalten Kreuzprodukte oder sind im Bezug auf die Kosten äquivalent. Ersteres entsteht, wenn Relationen in einer Reihenfolge gejoint werden, in der bei einem der Joins kein Prädikt möglich ist, beispielsweise ist dies für den Left-Deep Tree

$$((R_1 \bowtie R_2) \times R_4) \bowtie R_3$$

der Fall. Im Bezug auf die Kosten bei der gegebenen Kostenfunktion äquivalent sind Join-Trees, bei denen die Kinder eines Join Operators vertauscht wurden, etwa

$$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)$$

und

$$(R_3 \bowtie R_4) \bowtie (R_1 \bowtie R_2).$$

Im Beispiel müssen lediglich die Kosten für die Join-Reihenfolgen 1, 3 und 5 berechnet werden. Dies liegt am Aufbau der Kostenfunktion sowie den symmetrischen Größen der Relationen sowie ihrer Join Selektivitäten.

Die Berechnung von $C_{out}((R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4))$ sei hier exemplarisch in epischer Breite ausgeführt (Machen Sie es selber; Sie erkennen äußerst schnell ein Muster und müssen keine derartigen Formel-Konvolute schreiben):

$$\begin{aligned}
& C_{out}((R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)) \\
= \; & |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + C_{out}(R_1 \bowtie R_2) + C_{out}(R_3 \bowtie R_4) \\
= \; & |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + |R_1 \bowtie R_2| + C_{out}(R_1) + C_{out}(R_2) + |R_3 \bowtie R_4| + C_{out}(R_3) + C_{out}(R_4) \\
= \; & |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + |R_1 \bowtie R_2| + |R_3 \bowtie R_4| \\
= \; & f_{1,3} \cdot f_{1,4} \cdot f_{2,3} \cdot f_{2,4} \cdot |R_1 \bowtie R_2| \cdot |R_3 \bowtie R_4| + |R_1 \bowtie R_2| + |R_3 \bowtie R_4| \\
= \; & 0.5 \cdot (0.01 \cdot 10 \cdot 20) \cdot (0.01 \cdot 20 \cdot 10) + (0.01 \cdot 10 \cdot 20) + (0.01 \cdot 20 \cdot 10) \\
= \; & 2 + 2 + 2 \\
= \; & 6
\end{aligned}$$

Die Ergebnisse der anderen Relevanten Join-Reihenfolgen sind:

$((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4$	24
$((R_3 \bowtie R_2) \bowtie R_1) \bowtie R_4$	222
$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)$	6

Der Bushy-Tree 5 ist also der optimale Join-Tree.

Hausaufgabe 2

Gegeben sei die Anfrage:

```
select *
  from R, S, T
 where R.A = S.A and S.B = T.B and T.C = R.A
```

Des Weiteren soll gelten:

- `S.A` und `T.C` seien Fremdschlüssel auf `R`
- `S.B` sei Fremdschlüssel auf `T`
- `R.A`, `T.B` seien Primärschlüssel von `R` respektive `T`
- Ihre Query-Engine unterstützt nur Nested-Loop-Joins
- Kardinalitäten: $|R| = 100, |S| = 1000, |T| = 10$
- Es gibt keine Indexe

Bestimmen Sie, wie in der Vorlesung gezeigt, den optimalen Ausführungsplan als Baum mit Kosten-/Kardinalitätsabschätzungen mit Hilfe von Dynamischem Programmieren. Verwenden Sie die Kostenfunktion C_{out}.

Lösung:

Für Equijoins über den Fremdschlüssel gilt die Abschätzung:

$$f_{i,j} = \frac{1}{|R_i|}, \text{mit Fremdschlüssel in } R_j$$

Da alle Relationen über Fremdschlüssel verbunden sind, gelten (vereinfachend) folgende Kardinalitäten:

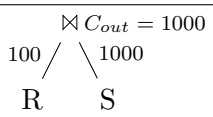
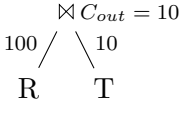
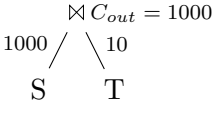
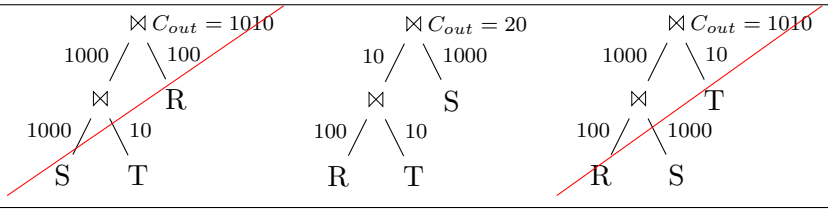
$$|R \bowtie S| = |S \bowtie R| = \tfrac{1}{|R|} \cdot |R| \cdot |S| = |S| = 1000$$

$$|R \bowtie T| = |T \bowtie R| = \tfrac{1}{|R|} \cdot |R| \cdot |T| = |T| = 10$$

$$|S \bowtie T| = |T \bowtie S| = \tfrac{1}{|T|} \cdot |S| \cdot |T| = |S| = 1000$$

$$|R \bowtie S \bowtie T| = \tfrac{1}{|R|} \cdot \tfrac{1}{|R|} \cdot \tfrac{1}{|T|} \cdot |R| \cdot |S| \cdot |T| = \tfrac{|S|}{|R|} = 10$$

Für die Berechnung der Kosten ergibt sich dann folgende DP-Tabelle:

DP-Tabelle		
Index	Pläne	Kosten
R	R	0
S	S	0
T	T	0
R,S		1000
R,T		10
S,T		1000
R,S,T		20

Hausaufgabe 3

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.

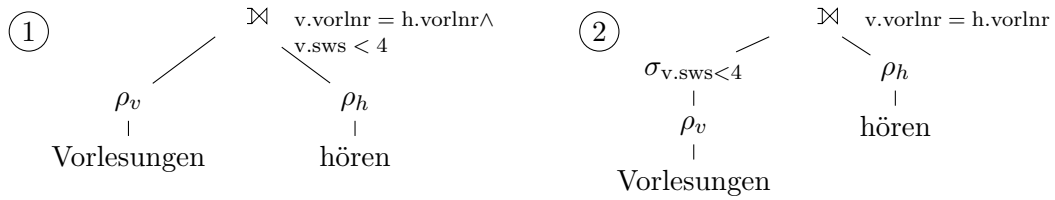
2) Falls der Index sortiert ist, dies wäre etwa bei einem B-Baum der Fall. Dadurch liegen Joinpartner zumindest nacheinander im Index, anders als bei einer Implementierung des Indexes mittels Hash.

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 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.