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

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. 13

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 Kardinaliät 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 Join Prädikat 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 Kardinaliät 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:

$$((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4 \quad (1)$$
$$((R_4 \bowtie R_3) \bowtie R_2) \bowtie R_1 \quad (2)$$
$$((R_3 \bowtie R_2) \bowtie R_1) \bowtie R_4 \quad (3)$$
$$((R_3 \bowtie R_2) \bowtie R_4) \bowtie R_1 \quad (4)$$

Bushy:

$$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4) \quad (5)$$

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):

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

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
```

Desweiteren 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 "kann" nur nested loops-Join
- Kardinalitäten: |R|=100, |S|=1000, |T|=10
- Es gibt keine Indexe

Bestimmen Sie den günstigsten QEP (query evaluation plan) auch als Baum mit Kosten-/Kardinalitäts-Abschätzungen. Verwenden Sie den in der Vorlesung gezeigten kostenbasierten DP (dynamisches Programmieren)-Optimierer.

Lösung:

In diesem einfachen Beispiel-System setzen sich Ausführungspläne aus lediglich zwei (physischen) Operatoren zusammen:

- Tablescan: scan(R_i), wobei R_i eine Basisrelation ist
- Nested loops-Join: $P_1 \bowtie^{\text{NL}} P_2$, mit den Teilplänen P_1 und P_2

D.h. dass stets alle Tupel der Basisrelationen gelesen werden und dass im Falle von Joins alle Tupel-Paare der beiden Eingaben verglichen werden. Als Kosten für die Anfragebearbeitung können also die Tupel gezählt werden, die die Query-Engine "in die Hand nehmen muss".

Kostenfunktion für Ausführungspläne:

$$C(P) = \begin{cases} |P| & \text{falls } P = \text{ scan}(...) \\ |P_1| \cdot |P_2| + C(P_1) + C(P_2) & \text{falls } P = P_1 \bowtie^{\text{NL}} P_2 \end{cases}$$

Die Kostenfunktion ist symmetrisch aufgrund der Kommutativität des NL-Joins: $C(R \bowtie^{\text{NL}} S) = C(S \bowtie^{\text{NL}} R)$

Für die Kardinaliäten kann (analog zu Hausaufgabe 1) angenommen werden, dass

$$|P| = \begin{cases} |R_i| & \text{falls } P \text{ ein scan über die Basisrelation } R_i \text{ ist} \\ (\prod_{R_i \in P_1, R_j \in P_2} f_{i,j}) |P_1||P_2| & \text{falls } P = P_1 \bowtie^{\text{NL}} P_2 \end{cases}$$

Sei $p_{i,j}$ das Join Prädikat zwischen den Relationen 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 Kardinaliät eines Join-Resultats ist $|R_i \bowtie_{p_{i,j}} R_j| = f_{i,j}|R_i||R_j|$.

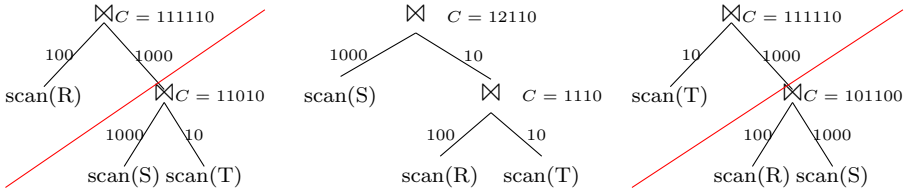
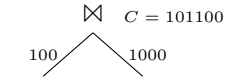
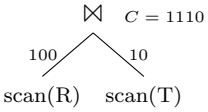
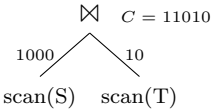
Für Equijoins über den Fremdschlüssel gilt die Abschätzung:

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

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

$$|R \bowtie S| = |S \bowtie R| = \frac{1}{|R|} \cdot |R| \cdot |S| = |S| = 1000$$
$$|R \bowtie T| = |T \bowtie R| = \frac{1}{|R|} \cdot |R| \cdot |T| = |T| = 10$$
$$|S \bowtie T| = |T \bowtie S| = \frac{1}{|T|} \cdot |S| \cdot |T| = |S| = 1000$$

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

BestePläneTabelle (DP table)		
Index	Pläne	Kosten
R,S,T		12110
R,S		101100
R,T		1110
S,T		11010
R	scan(R)	100
S	scan(S)	1000
T	scan(T)	10

Hausaufgabe 3

Bringen Sie die folgende Relation verlustlos und abhängigkeitsbewahrend in die 3. NF.

$$R(A, B, C, D, E, F)$$

FDs:

- $AB \rightarrow CD$
- $ABC \rightarrow D$
- $E \rightarrow C$
- $D \rightarrow C$
- $CDE \rightarrow AB$

Beachten Sie, dass es für die Lösung notwendig ist, einen Kandidatenschlüssel zu ermitteln, jedoch nicht alle Kandidatenschlüssel. Beachten Sie außerdem, dass die Relation das Attribut F enthält, welches bei der Zerlegung nicht wegfallen darf.

Lösung:

Der Kandidatenschlüssel **muss** bestimmt werden, anders verliert man im dritten Schritt des Synthesealgorithmus potentiell Semantik und in diesem Fall insbesondere das Attribut F. Es müssen (da nicht explizit gefordert) jedoch nicht alle Kandidatenschlüssel sondern

lediglich einer bestimmt werden. Ideales Verfahren hierzu ist, dass wir betrachten, welche Attribute nicht auf der rechten Seite von mindestens einer FD stehen. Diese **müssen** im Kandidatenschlüssel enthalten sein. Dies ist hier für E und F der Fall. Nun untersuchen wir, ob EF bereits ein Schlüssel für die Relation ist, womit wir dann bereits fertig wären. Wegen $AttrHuelle(FD, \{E, F\}) = \{E, F, C\} \neq R$ ist dies nicht der Fall. Wir versuchen nun möglichst intelligent ein Attribut hinzuzufügen, um Schlüsseleigenschaft zu erreichen. Da aus DE sofort AB folgt und E bereits definitiv im Schlüssel enthalten sein muss, fügen wir D hinzu. Damit gilt $AttrHuelle(FD, \{E, F, D\}) = \{E, F, D, C, A, B\} = R$, daher EDF ist Schlüssel. Insbesondere ist EFD auch Kandidatenschlüssel, da EF wie zuvor erklärt in jedem Schlüssel enthalten sein müssen, jedoch alleine kein Schlüssel sind, Reduktion nicht möglich, Schlüssel ist also Kandidatenschlüssel.

Ein weiterer Kandidatenschlüssel ist $ABEF$.

Nach dieser kurzen Überlegung muss lediglich der Synthesealgorithmus ausgeführt werden. Der Algorithmus ist in den Folien, im Buch und auf Wikipedia im Detail beschrieben. Sie werden massiv Punkte verlieren, wenn Sie ihn nicht beherrschen. Ein ausführliches Beispiel liegt im vorherigen Übungsblatt vor, an dieser Stelle sei lediglich zum Abgleich das Ergebnis angegeben:

Kanonische Überdeckung:

- $AB \to D$
- $E \to C$
- $D \to C$
- $DE \to AB$

Entstehende Relationen also:

- $R_1(A, B, D, E)$ mit zugeordneter FD $AB \to D$ sowie $DE \to AB$ (Zusammenfassung der Relationen, da eine in der anderen enthalten.)
- $R_2(E, C)$ mit zugeordneter FD $E \to C$
- $R_3(D, C)$ mit zugeordneter FD $D \to C$
- $R_4(E, F, D)$ ohne FD, für gewählten Kandidatenschlüssel eingefügt, da Kandidatenschlüssel nirgendwo enthalten.