

Prof. Dr. Guido Moerkotte

Email: moer@pi3.informatik.uni-mannheim.de

Pit Fender

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

Email: pfender@pi3.informatik.uni-mannheim.de

Anfrageoptimierung
Herbst-/Wintersemester 083. Übungsblatt
08. Oktober 2008

Aufgabe 1

Gegeben sei folgendes abstraktes relationales Schema:

 $R_1(a, c, d, e)$ $R_2(a, b)$ $R_3(b, c)$ $R_4(d)$ $R_5(e)$

Geben Sie für die folgenden Anfragen den Query Graphen an. Klassifizieren Sie den Query Graphen entsprechend der Notation aus der Vorlesung. Wie groß ist jeweils der Suchraum für diese Klasse von Anfragen, wenn ein left-deep bzw. bushy Join-Tree ohne Kreuzprodukte berücksichtigt wird? Existieren effiziente Algorithmen für die Bestimmung der Join-Reihenfolge der Anfrage?

Aufgabe 1 a)

```
SELECT DISTINCT *
FROM R1, R2, R3, R4
WHERE R1.c = R3.c
      AND R2.b = R3.b
      AND R1.d = R4.d;
```

Lösung

Query Graph: chain

Suchraum für Join Trees ohne Kreuzprodukte:

links-tief: 2^{n-1} bushy: $2^{n-1}C(n-1)$

Algorithmus: polynomielle Laufzeit für join trees ohne Kreuzprodukte

Aufgabe 1 b)

```
SELECT DISTINCT *
```

```

FROM R1, R2, R3, R5
WHERE R1.a = R2.a
      AND R1.c = R3.c
      AND R1.e = R5.e
      AND R2.b = R3.b;

```

Lösung

Query Graph: cyclic
 Suchraum für Join Trees ohne Kreuzprodukte:
 Im schlimmsten Fall eine clique, also:
 links-tief: $n!$
 bushy: $n! \cdot C(n - 1)$
 Algorithmus: $NP - hard$

Aufgabe 1 c)

```

SELECT DISTINCT *
FROM R1, R2, R3, R4, R5
WHERE R1.a = R2.a
      AND R1.c = R3.c
      AND R1.d = R4.d
      AND R1.e = R5.e;

```

Lösung

Query Graph: star
 Suchraum für Join Trees ohne Kreuzprodukte:
 links-tief: $2 \cdot (n - 1)!$;
 bushy: $2^{n-1} \cdot (n - 1)!$
 Algorithmus: P ohne Kreuzprodukte, $NP - hard$ mit Kreuzprodukten

Aufgabe 1 d)

Erstellen Sie eine Tabelle, in der sie Join-Ordering-Algorithmen klassifizieren können.
 D.h. die Tabelle sollte eine Spalte für den Namen des Algorithmus und je eine Spalte
 für jedes Merkmal eines Join-Ordering-Problems enthalten.

Lösung

Name	Query Graph	Join-Tree	Kreuzprodukte	Kostenfunktion	Komplexität	optimal	Anmerkungen
Greedy-1	beliebig	left-deep	ja	beliebig	$O(n \lg n)$	nein	
Greedy-2	beliebig	left-deep	ja	beliebig	$O(n \lg n)$	nein	
Greedy-3	beliebig	left-deep	ja	beliebig	$O(n^2 \lg n)$	nein	
GOO	beliebig	bushy	ja	beliebig	$O(n^2)$	nein	
IKKBZ	azyklisch	left-deep	nein	ASI	$O(n \lg n)$	ja	1 Kostenfunktion je Pädikat

Tip: Tragen Sie die in der Vorlesung besprochenen Algorithmen in die Tabelle ein.

Aufgabe 2

Zeigen Sie, daß die rekursive Formulierung der Kostenfunktion im IKKBZ-Algorithmus korrekt und wohl definiert ist, d.h.

$Cost_H(G) = C_H(G)$ sowie $C_H(S_1S_2)$ und $C_H(S'_1S'_2)$ liefern alle Aufteilung von S in S_1S_2 und $S'_1S'_2$ dieselben Kosten.

Lösung

Wohldefiniertheit:

Für leere Sequenzen und Sequenzen aus einer Relation ergibt sich die Wohldefiniertheit direkt aus der Definition. Sei $S = S_1S_2 = S'_1S'_2$ mit $S_1 = R_1 \dots R_j, S_2 = R_{j+1} \dots R_n, S'_1 = R_1 \dots R_i, S'_2 = R_{i+1} \dots R_n$ und o.B.d.A. $0 \leq i < j \leq n$. Dann sei $\tilde{S} = R_i \dots R_j$.

$$\begin{aligned}
 C_H(S_1S_2) &= C_H(S_1) + T(S_1)C_H(S_2) \\
 &= C_H(S'_1) + T(S'_1)C_H(\tilde{S}) + T(S_1)C_H(S_2) \\
 &= C_H(S'_1) + T(S'_1)C_H(\tilde{S}) + \prod_{k=1}^j (s_k n_k) \cdot C_H(S_2) \\
 &= C_H(S'_1) + T(S'_1)C_H(\tilde{S}) + \prod_{k=1}^i (s_k n_k) \prod_{l=i+1}^j (s_l n_l) \cdot C_H(S_2) \\
 &= C_H(S'_1) + T(S'_1)C_H(\tilde{S}) + T(S'_1) \cdot T(\tilde{S}) \cdot C_H(S_2) \\
 &= C_H(S'_1) + T(S'_1)(C_H(\tilde{S}) + T(\tilde{S})C_H(S_2)) \\
 &= C_H(S'_1) + T(S'_1)C_H(S'_2) \\
 &= C_H(S'_1S'_2)
 \end{aligned}$$

Korrektheit: Induktion über die Länge von S .

z.Z.: $C_H(S) = Cost_H(S)$

IA:

$S = \epsilon: C_H(\epsilon) = 0 = Cost_H(\epsilon)$

$S = R_i$, Wurzel: $C_H(R_i) = 0 = Cost_H(R_i)$

IV: Die Behauptung gelte für eine Sequenz der Länge n , n beliebig und fest gewählt.

IS: $n \Rightarrow n + 1$

$$\begin{aligned}
C_H(S) &= C_H(S_1 R_{n+1}) \\
&= C_H(S_1) + T(S_1) \cdot C_H(R_{n+1}) \\
&\stackrel{IV}{=} \sum_{i=2}^n \left(\prod_{j=1}^{i-1} (s_j n_j) \cdot h_i(n_i) \right) + \prod_{j=1}^n (s_j n_j) \cdot h_{n+1}(n_{n+1}) \\
&= \sum_{i=2}^{n+1} \left(\prod_{j=1}^{i-1} (s_j n_j) \cdot h_i(n_i) \right) \\
&= Cost_H(S)
\end{aligned}$$

Aufgabe 3

Erweitern Sie das Programm vom vorherigen Übungsblatt so, dass wenn möglich Joins erzeugt werden (Klasse HashJoin in tinydb). Messen Sie den Unterschied für die Anfrage

```
select *
from lineitem l,order o
where l.l_orderkey=o.o_orderkey.
```

Lösung

siehe JoinTranslation.java
Ergebnis besteht jeweils aus 60175 Tupeln

Laufzeiten:
CanonicalTranslation: 14932s = 4h:44min:52s
JoinTranslation: 182s = 3min:2s

Aufgabe 4

Gegeben $|R_1|$, $|R_2|$, die Größe des Wertebereichs von $R_1.x$ und $R_2.y$ sowie die Information ob $R_1.x$ bzw. $R_2.y$ Schlüssel von R_1 bzw. R_2 ist.

Aufgabe 4 a)

Wie läßt sich die Selektivität von $\sigma_{R_1.x=c}$ abschätzen, wobei c eine Konstante ist?

Lösung

- wenn x Schlüssel $1/|R_1|$
- sonst $1/|\text{Wertebereich von } x|$ (identisch, wenn x dicht ist).

Aufgabe 4 b)

Wie läßt sich die Selektivität von $Join_{R_1.x=R_2.y}$ abschätzen?

Lösung

- wenn x Schlüssel $1/|R_1|$
- wenn y Schlüssel $1/|R_2|$
- sonst $1/\max(|\text{Wertbereich von } x|, |\text{Wertbereich von } y|)$