Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Prof. H.-J. Schek — *A. Feßler, U. Röhm, H. Schuldt, R. Weber*

Informationssysteme IS-G SS 1997

Übung 9 — Beispiellösung

Aufgabe 1: B*-Baum

Ein B*-Baum vom Typ (k, k^*) hat folgende Eigenschaften:

1. Jeder Weg von der Wurzel zu einem Blatt hat die gleiche Länge.
2. Jeder Knoten – ausser Wurzel und Blättern – hat mindestens k und höchstens $2k$ Einträge. Blätter haben mindestens k^* und höchstens $2k^*$ Einträge. Die Wurzel hat entweder maximal $2k$ Einträge, oder sie ist ein Blatt mit maximal $2k^*$ Einträgen.
3. Jeder Knoten mit n Einträgen, ausser den Blättern, hat $n+1$ Kinder.

Geben Sie für den B*-Baum eine Formel an, mit der man die obere und untere Schranke für die Höhe des Baums bei gegebenem k, k^* und N (der Anzahl der eingetragenen Schlüssel) bestimmen kann.

Ein B-Baum erreicht für eine gegebene Anzahl N von Einträgen seine maximale Höhe, wenn alle Knoten und Blätter minimal belegt sind. Umgekehrt erhält man einen B*-Baum mit Mindesthöhe, wenn man alle Speicherkapazitäten voll ausnutzt. Es gilt also folgende Beziehung:*

$$\textit{Höhe bei maximaler Belegung} \leq h \leq \textit{Höhe bei minimaler Belegung}$$

Höhe	**Minimalbelegung**	**Maximalbelegung**
1	1	$2k^*$
2	$2k^*$	$(2k+1)2k^*$
3	$2(k+1)k^*$	$(2k+1)(2k+1)2k^*$
$\vdots$	$\vdots$	$\vdots$
h	$(k+1)^{(h-2)}2k^*$	$(2k+1)^{(h-1)}2k^*$

Wenn man N mit diesen Minimal-/Maximalbelegungen gleichsetzt und nach h auflöst, erhält man die gesuchte Formel:

$$1 + log_{2k+1}(\frac{N}{2k^*}) \leq h \leq 2 + log_{k+1}(\frac{N}{2k^*}) \qquad \text{(für } h \geq 2\text{)}$$

Aufgabe 2: Anfrageoptimierung

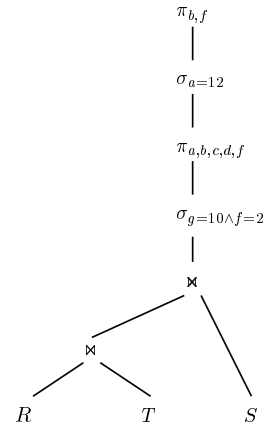
Bei komplexen Anfragen hat die Reihenfolge der Operationsausführung starken Einfluss auf die Ausführungszeit der Anfrage.

Gegeben sind die Relationen $R(\underline{a}, b, c)$, $S(\underline{c}, d, e)$ und $T(\underline{e}, f, g)$.

Berechnet werden soll die Anfrage

$$\pi_{b,f}\ \sigma_{a=12}\ \pi_{a,b,c,d,f}\ \sigma_{g=10,f=2}\ ((R \bowtie T) \bowtie S)$$

a) Zeichnen Sie den Operatorbaum für die obige Anfrage.

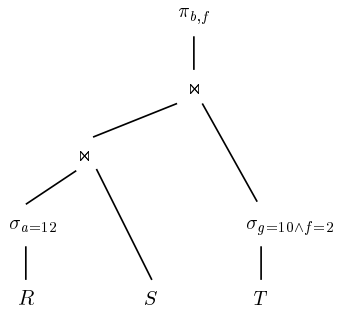


b) Spielen Sie Optimierer und formen Sie den obigen Operatorbaum algebraisch so um, dass ein möglichst günstiger Operatorbaum entsteht. Nehmen Sie Umformungen zunächst am Algebra-Ausdruck vor.

- *Selektionen sind möglichst früh auszuführen,*
- *Projektionen sollen zusammengefasst werden, und*
- *Kartesische Produkte eleminieren.*

Unter Anwendung dieser Regeln ergibt sich folgender Ausdruck bzw. Operatorbaum, wobei insbesondere zu beachten ist, dass der Join $R \bowtie T$ ein Kartesisches Produkt ist und deshalb umgeformt wird:

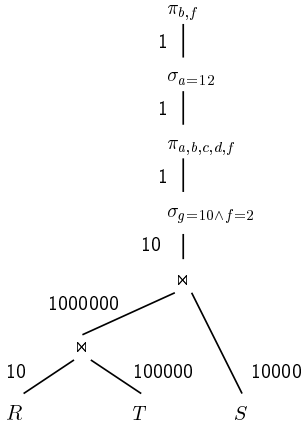
$$\pi_{b,f}\ ((\sigma_{a=12}(R) \bowtie S) \bowtie\ \sigma_{g=10,f=2}(T))$$

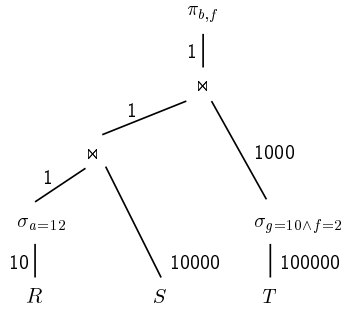


c) Nehmen Sie an, dass alle Attributwerte der Attribute a, f und g jeweils unter 10 Werten gleichverteilt und unkorreliert sind. Die Werte der Fremdschlüssel ($R.c$ und $S.e$) sollen ebenfalls in Bezug auf die Schlüsselwerte gleichverteilt sein. Die Kardinalitäten der Relationen seien mit $Card(R) = 10$, $Card(S) = 10.000$, $Card(T) = 100.000$ gegeben.

Berechnen Sie die Grösse der erwarteten Zwischenergebnismengen (Anzahl Tupel) bei der Ausführung des nichtoptimierten und des optimierten Operatorbaumes!

Die Anzahlen der maximal in den Strömen fliessenden Tupel, d.h. der Zwischenergebnismengen, sind in Schrägschrift gedruckt. Bei den Projektionen findet durch Duplikateliminierung ebenfalls eine Verringerung der Zwischenergebnismengen statt, die aber in diesem Fall nicht genau quantifizierbar ist.





Aufgabe 3: Nested Block Join

Noch einmal der Algorithmus des Nested Block Joins von Folie 8-7 in anderen Worten:
Stehen im Hauptspeicher m Seiten für die Berechnung des Joins zur Verfügung, reserviert der Nested Block Join-Algorithmus k Seiten für die innere Schleife und $m - k$ für die äussere. Die äussere Relation, nennen wir sie R, wird portionsweise in Blöcken zu $m - k$ Seiten eingelesen. Für jede dieser Portionen muss die komplette innere Relation S in Portionen zu k Seiten eingelesen werden. Alle Tupel der Relation R, die sich auf den eingelagerten $m - k$ Seiten befinden, werden dann mit allen Tupeln aus S in den k Seiten verglichen. Man kann pro Durchlauf das Einlesen einer Portion von k Seiten sparen, wenn die innere Relation im Zick-Zack-Verfahren durchlaufen wird, also abwechselnd vorwärts und rückwärts.

a) Seien b_R und b_S die Anzahl der Seiten, die R respektive S belegen. Ermitteln Sie die Kostenformel (bzgl. Seitenzugriffen) für den Nested Block Join.

Für die innere Schleife wurden k Seiten reserviert, für die äussere $m-k$. Die Relation R wird einmal vollständig durchlaufen, es ergeben sich dabei b_R Seitenzugriffe. Die innere Schleife wird $\lceil b_R/(m-k) \rceil$-mal durchlaufen. Bei jedem Durchlauf ergeben sich $b_S - k$ Zugriffe, da durch das Zick-Zack-Verfahren die letzten k Seiten des vorigen Durchlaufs wiederverwendet werden können. Lediglich der erste Durchlauf muss komplett alle b_S Seiten lesen. Insgesamt ergeben sich die Kosten zu

$$b_R + k + \lceil b_R/(m-k) \rceil \cdot (b_S - k)$$

b) Erläutern Sie, warum die Kostenformel für den Nested Block Join minimal wird, wenn $k = 1$ und R die kleinere der beiden Relationen ist.

Zuerst entledigen wir uns der Rundungsfunktion:

$$\begin{aligned} b_R + k + \lceil b_R/(m-k) \rceil \cdot (b_S - k) &\leq b_R + k + (\frac{b_R}{m-k} + 1) \cdot (b_S - k) \\ &= b_R + k + \frac{b_R b_S - b_R k}{m-k} + b_S - k \\ &= b_R + b_S + b_R \cdot \frac{b_S - k}{m-k} \\ &= b_R + b_S + b_R \cdot \frac{(m+p) - k}{m-k} \\ &= b_R + b_S + b_R \cdot (1 + \frac{p}{m-k}) \end{aligned}$$

Hierbei haben wir b_S mit $m + p$ (p beliebig positiv) ersetzt, da i.a. $b_S \gg m$. Man sieht, dass der letzte Term $(1 + \frac{p}{m-k})$ für $k = 1$ minimal wird. Die ersten beiden Summanden $b_R + b_S$ sind konstant bzgl. der Wahl der kleineren Relation als R oder als S. Jedoch wird der letzte Summand $b_R \cdot (1 + \frac{p}{m-k})$ minimal, wenn man R als kleinere der beiden Relationen wählt, da dann b_R der kleinere Wert ist, mit dem multipliziert wird.