



**Übungen zur Vorlesung
"Datenbanksysteme II"**
Wintersemester 2008/2009
Manuel Mayr (manuel.mayr@uni-tuebingen.de)

8. Lösungsblatt

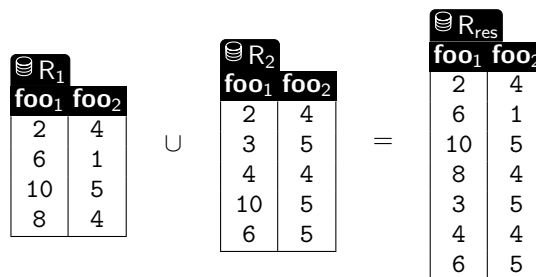
Ausgabe: 4. Dezember 2008 · Besprechung: 11. Dezember 2008

Lösung 1: Hashing and Sorting

(8 Punkte)

Hash- und sortierungsbasierte Ansätze können nicht bloß für Joins, sondern auch für andere Operatoren der relationalen Algebra eingesetzt werden. An den folgenden Beispielen soll die Semantik der beiden Operatoren *relationale Vereinigung* (\cup) und *Schnittmenge* (\cap) erläutert werden.

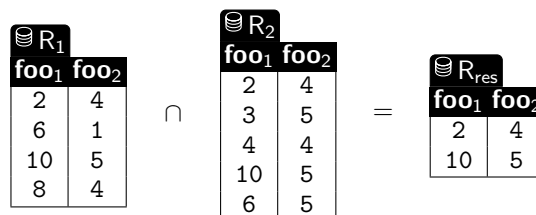
- Vereinigung (\cup)



Lösung:

Der sortierungsbasierte Ansatz zur Vereinigung zweier Relationen R und S findet sich in Algorithmus 1. Der hashbasierte Ansatz zur Vereinigung zweier Relationen findet sich in Algorithmus 3 und 2.

- Schnittmenge (\cap)



Algorithm 1 SortUnion(R, S)

Require: Zwei Relationen R und S

Ensure: $T = R \cup S$ (Duplikatfrei)

Sortiere R und S mit einem Sortieralgorithmus (bspw. External Merge Sort)
lexikografisch nach ihren Attributen

$r \leftarrow$ sei das erste Tupel in R

$s \leftarrow$ sei das erste Tupel in S

$e \leftarrow$ leer

while $r \neq$ leer **do**

while $s \neq$ leer $\wedge s < r$ **do**

$e \leftarrow e \cup \{s\}$

$s \leftarrow$ nächstes Tupel in S

end while

if $r = s$ **then**

$s \leftarrow$ nächstes Tupel in S

end if

$e \leftarrow e \cup \{r\}$

$r \leftarrow$ nächstes Tupel in R

end while

return e

Algorithm 2 HashDuplElim(R)

Require: Eine Relation R

Ensure: R hat keine Duplikate

Wähle zufällig einen Zahl k

Partitioniere R und S nach h_k

for jede Partition P **do**

if P passt in den Hauptspeicher **then**

 Führe eine In-Memory Duplikateliminierung durch

else

 HashDuplElim(P)

end if

end for

Algorithm 3 HashUnion(R, S)

Require: Zwei Relationen R und S , wobei R und S als Relationen
bereits duplikatfrei sind

Ensure: $T = R \cup S$ (Duplikatfrei)

$T \leftarrow$ Konkatenation von R und S

HashDuplElim(T)

Lösung:

Die relationale Schnittmenge zweier Relationen R und S ist äquivalent zum *Natural Join*. Voraussetzung ist hier natürlich, dass das Schema von R und S identisch ist. Dies impliziert, dass man jeden beliebigen sortierungsbasierten (MergeJoin) oder hashbasierten (HashJoin) Algorithmus verwenden kann.

Beide Varianten wurden bereits in der Vorlesung auf Folie 8.39, respektive 8.43 diskutiert und werden hier deshalb ausgespart.

Diskutieren Sie die beiden Operatoren im Kontext von Hash- und Sortierungsbasierten Operatoren. Skizzieren Sie die Algorithmen in Pseudocode!

Lösung 2: I/O Kosten**(12 Punkte)**

Es sei $R(a, b, c, d)$ eine Relation mit a als Kandidatenschlüssel. R enthält 1.000.000 Tupel, wobei 10 Tupel auf genau eine Seite passen. Die Verteilung der Werte in R ist uniform.

Folgende Informationen seien im Katalog der Datenbank gegeben:

SYSCAT			
Attribut	max Wert	min Wert	Anzahl unterschiedlicher Werte
a	2.000.000	10	1.000.000
b	10.000.000	1	30
c	1.000	-30.000	31000
d	2.500.000	27	232

Beantworten Sie die Fragen basierend auf den folgenden Situationen.

- (a) Kein Index, unsortierte Daten
- (b) Kein Index, sortierte Daten
- (c) Ein clustered B+ Baum Index, mit Schlüssel (a, b, c) und Höhe 4.
- (d) Hash Index auf dem Attribut a , mit 1.2 I/O's pro Lookup.

1. Berechnen Sie die Selektivitäten für die folgenden Prädikate.

- (a) $a = 23$
- (b) $b > 12 \wedge b > 42$
- (c) $a > 13 \wedge (c > 12 \vee b < 30)$
- (d) $a > 56 \wedge (b = 23 \wedge c = 21)$
- (e) $d = 12 \vee c < 12$

Lösung:

(a)

$$p \equiv a = 23 \Rightarrow \text{sel}(p) = \frac{1}{\text{Anzahl unterschiedlicher Werte}} = 10^{-6} \approx 0$$

(b)

$$\begin{aligned} p \equiv b > 12 \wedge b > 42 &\Rightarrow p \equiv b > 42 \\ &\Rightarrow \text{sel}(p) = \frac{\text{max Wert} - 42}{\text{max Wert} - \text{min Wert}} \approx 1 \end{aligned}$$

(c)

$$p \equiv a > 13 \wedge (c > 12 \vee b < 30)$$

$$p_1 \equiv a > 13 \Rightarrow \text{sel}(p_1) = \frac{\text{max Wert} - 13}{\text{max Wert} - \text{min Wert}} \approx 1$$

$$p_2 \equiv c > 12 \Rightarrow \text{sel}(p_2) = \frac{\max \text{ Wert} - 12}{\max \text{ Wert} - \min \text{ Wert}} \approx 32 \cdot 10^{-3}$$

$$p_3 \equiv b < 30 \Rightarrow \text{sel}(p_3) = \frac{30 - \min \text{ Wert}}{\max \text{ Wert} - \min \text{ Wert}} \approx 3 \cdot 10^{-6}$$

$$\Rightarrow \text{sel}(p) = \text{sel}(p_1) \cdot (\text{sel}(p_2) + \text{sel}(p_3) - \text{sel}(p_2) \cdot \text{sel}(p_3)) = 32 \cdot 10^{-3}$$

(d)

$$p \equiv a > 56 \wedge (b = 23 \vee c = 21)$$

$$p_1 \equiv a > 56 \Rightarrow \text{sel}(p_1) = \frac{\max \text{ Wert} - 56}{\max \text{ Wert} - \min \text{ Wert}} \approx 1$$

$$p_2 \equiv b = 23 \Rightarrow \text{sel}(p_2) = \frac{1}{\text{Anzahl unterschiedlicher Werte}} \approx 32 \cdot 10^{-3}$$

$$p_3 \equiv c = 21 \Rightarrow \text{sel}(p_3) = \frac{1}{\text{Anzahl unterschiedlicher Werte}} \approx 32 \cdot 10^{-6}$$

$$\Rightarrow \text{sel}(p) = \text{sel}(p_1) \cdot \text{sel}(p_2) \cdot \text{sel}(p_3) = 1024 \cdot 10^{-9}$$

(e)

$$p \equiv d = 12 \vee c < 12$$

$$\begin{aligned} p_1 \equiv d = 12 &\Rightarrow d < \min \text{ Wert} \\ &\Rightarrow \text{sel}(p_1) = 0 \end{aligned}$$

$$p_2 \equiv c < 12 \Rightarrow \text{sel}(p_2) = \frac{12 - \min \text{ Wert}}{\max \text{ Wert} - \min \text{ Wert}} \approx 1$$

$$\Rightarrow \text{sel}(p) = \text{sel}(p_2) + \text{sel}(p_2) - \text{sel}(p_1) \cdot \text{sel}(p_2) \approx 1$$

2. Berechnen Sie die I/O Kosten für die Selektionen (d.h. ignorieren Sie eventuelle Kosten der Operatoren \cup^{rid} und \cap^{rid}) der Prädikate in Aufgabe 2.1 basierend auf den oben genannten Situationen.

Lösung:

- (a) Die I/O Kosten, wenn kein Index vorhanden ist und keine Sortierung der Daten vorliegt wird durch die Formel $N_R + \text{sel}(p) \cdot N_R$ berechnet.

i.

$$p \equiv a = 23 \Rightarrow \left\lceil \frac{10^6}{10} \right\rceil + 0 \cdot \left\lceil \frac{10^6}{10} \right\rceil = 10^5$$

ii.

$$p \equiv b > 12 \wedge b > 42 \Rightarrow \left\lceil \frac{10^6}{10} \right\rceil + 1 \cdot \left\lceil \frac{10^6}{10} \right\rceil = 2 \cdot \left\lceil \frac{10^6}{10} \right\rceil = 2 \cdot 10^5$$

iii.

$$p \equiv a > 13 \wedge (c > 12 \vee b < 30) \Rightarrow \left\lceil \frac{10^6}{10} \right\rceil + 32 \cdot 10^{-3} \cdot \left\lceil \frac{10^6}{10} \right\rceil \approx 103,200$$

iv.

$$p \equiv a > 56 \wedge (b = 23 \vee c = 21) \Rightarrow \left\lceil \frac{10^6}{10} \right\rceil + 1024 \cdot 10^{-9} \cdot \left\lceil \frac{10^6}{10} \right\rceil \approx 10^5$$

v.

$$p \equiv d = 12 \vee c < 12 \Rightarrow \left\lceil \frac{10^6}{10} \right\rceil + 1 \cdot \left\lceil \frac{10^6}{10} \right\rceil \approx 2 \cdot 10^5$$

(b) Die Annahme, die wir hier treffen wollen ist, dass die Sortierung der lexikografisch zuerst nach a , dann nach b usw. auf den Daten gegeben ist. Eine andere Reihenfolge ist hier natürlich auch korrekt, falls die nachfolgenden Lösungen stimmig sind. Die entsprechenden I/O Kosten werden mit der Formel $ld(N_R) + sel(p) \cdot 2$ berechnet.

i. Da die Sortierung zuerst nach die Sortierung zuerst nach a erfolgt ist können wir das Prädikat $p \equiv a = 23$ mittels binärer Suche optimal filtern.

$$ld\left(\left\lceil \frac{10^6}{10} \right\rceil\right) \cdot 0 \cdot \left\lceil \frac{10^6}{10} \right\rceil \cdot 2 = 16.6$$

ii. Das Prädikat $p \equiv b > 12 \wedge b > 42$ kann von der gegebenen Sortierung nicht profitieren, sodass wir auf den Fall *unsortiert, kein Index* zurückgreifen müssen.

iii.

iv.

v. Auch das Prädikat $p \equiv d = 12 \vee c < 12$ kann nicht von der Sortierung profitieren, sodass wir wieder auf den Fall *unsortiert, kein Index* zurückgreifen müssen.

(c) Hier muss natürlich erst einmal unterschieden werden, welche Prädikate der B+ Baum effizient verarbeiten kann. Sollte der B+ Baum einsetzbar, dann berechnen sich die I/O Kosten aus $h_{B+ \text{ Baum}} + 2sel(p)N_R$

i. Bei $p \equiv a = 23$ kann der B+ Baum, das Prädikat effizient filtern.

$$p \equiv a = 23 \Rightarrow 4 + 2 \cdot 0 \cdot \left\lceil \frac{10^6}{10} \right\rceil = 4$$

ii. Das Prädikat $p \equiv b > 12 \wedge b > 42$ kann nicht vom B+ Baum beantwortet werden, da es keinen Präfix des Suchschlüssels darstellt. Hier sind die Kosten also die selben, wie bei *unsortiert, kein Index*.

iii.

Das Prädikat $p \equiv a > 13 \wedge (c > 12 \vee b < 30)$ kann super vom Index B+ Baum verwendet werden. Wir nutzen hier erstmal die Distributivität aus sodass $p \equiv a > 13 \wedge (c > 12 \vee b < 30) \Rightarrow (a > 13 \wedge c > 12) \vee (a > 13 \wedge b < 30)$. Hier haben wir also zwei Prädikate die vom Index beantwortet werden können. Einerseits $(a > 13 \wedge b < 30)$, das ein Präfix des B+ Baumes darstellt, und $(a > 13 \wedge c > 12)$. Bei $(a > 13 \wedge c > 12)$ kann der B+ Baum erstmal verwendet werden, um $a > 13$ zu finden, beim Scan des *Sequence Set* kann allerdings, das Prädikat $c > 12$ einfach mitüberprüft werden (*index sargable predicate*).

Für $(a > 13 \wedge b < 30)$ ergibt sich hier die Selektivität $1 \cdot 3 \cdot 10^{-6}$. Daraus ergeben sich die I/O Kosten $4 + 2 \cdot 1 \cdot 3 \cdot 10^{-6} \cdot 10^5 = 4.6$

iv. Das Prädikat $p \equiv a > 56 \wedge (b = 23 \wedge c = 21)$, kann optimal vom B+ Baum beantwortet werden.

$$p \equiv a > 56 \wedge (b = 23 \wedge c = 21) \Rightarrow 4 + 2 \cdot 1024 \cdot 10^{-9} \cdot \left\lceil \frac{10^6}{10} \right\rceil = 42048 \cdot 10^{-4}$$

v. Auch das Prädikat $p \equiv d = 12 \vee c < 12$ kann leider nicht vom B+ Baum Index beantwortet werden. Die Kosten müssen wir also wieder auf *unsortiert, kein Index*.

(d) Auch hier muss wieder drauf geachtet werden, welches Prädikat der Index überhaupt beantworten kann. Die Formel hierbei ist $|R| + \text{sel}(p) \cdot N_R$.

i. Das Prädikat $p \equiv a = 23$ ist für den Index natürlich zugeschnitten und kann optimal beantwortet werden. Die I/O Kosten belaufen sich hier auf 1.2.

ii. Leider gibt es für den Index hier nichts zu tun, und wir müssen, wieder auf den Fall *unsortiert, kein Index* zurückgreifen.

iii. Auch hier hilft unser Index nicht weiter \Rightarrow *unsortiert, kein Index*

iv. Keine Aufgabe für den Index auch hier. Deshalb wird wieder der Fall *unsortiert, kein Index*.

v. Unser Index kann auch hier nicht zur Rate gezogen werden \Rightarrow *unsortiert, kein Index*.