

Übung zur Vorlesung Grundlagen: Datenbanksysteme im WS06/07
Martin Wimmer (Martin.Wimmer@in.tum.de)

Lösung von Blatt Nr. 7

Aufgabe 1

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

Lösung

Obere Schranke für die Höhe:

Tiefe	minimale Knotenzahl in einer Tiefe
0	1
1	2
2	$2 \cdot (k + 1)$
3	$2 \cdot (k + 1)^2$
\vdots	\vdots
h	$2 \cdot (k + 1)^{h-1}$

Gesamtsumme der Knoten in einem B-Baum:

$$\begin{aligned} 1 + \sum_{i=1}^h 2 \cdot (k + 1)^{i-1} &\stackrel{\text{geometrische Reihe}}{=} 1 + 2 \frac{(k + 1)^h - 1}{k + 1 - 1} \\ &= 1 + 2 \frac{(k + 1)^h - 1}{k} \end{aligned}$$

Minimale Anzahl der Elemente im Baum:

$$\begin{aligned} 1 + k \cdot 2 \frac{(k + 1)^h - 1}{k} &= 1 + 2(k + 1)^h - 2 \\ &= 2(k + 1)^h - 1 \end{aligned}$$

Es gilt also:

$$\begin{aligned} 2(k + 1)^h - 1 &\leq n \\ 2(k + 1)^h &\leq n + 1 \\ (k + 1)^h &\leq \frac{n + 1}{2} \\ h &\leq \log_{k+1} \left(\frac{n + 1}{2} \right) \end{aligned}$$

Untere Schranke für die Höhe:

Tiefe	maximale Knotenzahl in einer Tiefe
0	1
1	$(2k + 1)$
2	$(2k + 1)^2$
3	$(2k + 1)^3$
\vdots	\vdots
h	$(2k + 1)^h$

Maximale Anzahl der Knoten im Baum:

$$\sum_{i=0}^h (2k + 1)^i \stackrel{\text{geometrische Reihe}}{=} \frac{(2k + 1)^{h+1} - 1}{2k + 1 - 1}$$

$$= \frac{(2k + 1)^{h+1} - 1}{2k}$$

Maximale Elementanzahl im Baum:

$$2k \cdot \frac{(2k + 1)^{h+1} - 1}{2k} = (2k + 1)^{h+1} - 1$$

Es gilt also:

$$n \leq (2k + 1)^{h+1} - 1$$

$$n + 1 \leq (2k + 1)^{h+1}$$

$$\log_{2k+1}(n + 1) \leq h + 1$$

$$h \geq \log_{2k+1}(n + 1) - 1$$

Aufgabe 2

- (a) Beim Hashing wird der Modulfunktion häufig eine *Faltung* vorgeschaltet. Das kann beispielsweise für Zahlen die Quersumme sein und für Zeichenketten die Summe der Buchstabenwerte. Fügen Sie die Studenten aus dem Universitätsbeispiel in eine Hashtabelle der Größe vier mit Überlaufbuckets (mit Bucketgröße zwei) ein und schalten Sie bei der Berechnung der Hashwerte zusätzlich eine Quersummenfunktion vor. Werden die Studenten jetzt gleichmäßiger verteilt?
- (b) Gegeben sei eine erweiterbare Hashtabelle mit globaler Tiefe t . Wie viele Verweise zeigen vom Verzeichnis auf einen Behälter mit lokaler Tiefe t' ?

Lösung

- (a) Zunächst fügen wir die Studenten in eine Hashtabelle der Größe vier mit Überlaufbuckets ein. Zur Berechnung der Hashwerte wird die Modulfunktion verwendet. Die Tabelle *Studenten* mit den Hashwerten ($MatrNr \bmod 4$) sieht folgendermaßen aus:

MatrNr	Student	Hashwert (Modulo)
24002	Xenokrates	2
25403	Jonas	3
26120	Fichte	0
26830	Aristoxenos	2
27550	Schopenhauer	2
28106	Carnap	2
29120	Theophrastos	0
29555	Feuerbach	3

Damit erhält man folgende Hashtabelle:

0	26120, Fichte 29120, Theophrastos	
1		
2	24002, Xenokrates 26830, Aristoxenos	27550, Schopenhauer 28106, Carnap
3	25403, Jonas 29555, Feuerbach	

Es zeigt sich, dass die Moduloberechnung unbedingt mit einer Primzahl und schon gar nicht, wie hier, mit einer geraden Zahl erfolgen sollte. So werden im Beispiel alle geraden Matrikelnummern auf Buckets gerader Kennung und alle ungeraden auf Buckets mit ungerader Kennung abgebildet. Diese Schiefelage sollte man unbedingt vermeiden.

Nun wird die angegebene Quersummenfunktion als Faltung der Hashwertbestimmung vorschaltet:

MatrNr	Student	Quersumme	Hashwert (Modulo)
24002	Xenokrates	8	0
25403	Jonas	14	2
26120	Fichte	11	3
26830	Aristoxenos	19	3
27550	Schopenhauer	19	3
28106	Carnap	17	1
29120	Theophrastos	14	2
29555	Feuerbach	26	2

Damit erhält man folgende Hashtabelle der Größe 4 mit Bucketgröße 2:

0	24002, Xenokrates	
1	28106, Carnap	
2	25403, Jonas 29120, Theophrastos	29555, Feuerbach
3	26120, Fichte 26830, Aristoxenos	27550, Schopenhauer

Die Einträge sind zwar etwas gleichmäßiger auf die Buckets verteilt, dafür sind nun aber 2 Überlaufbuckets nötig.

- (b) In dem Verzeichnis einer Hashtabelle mit globaler Tiefe t werden t Bits eines Hashwerts für die Identifizierung eines Verzeichniseintrags verwendet. Für einen Behälter mit lokaler Tiefe t' sind hingegen nur die ersten t' Bits dieses Bitmusters relevant.

Mit anderen Worten bedeutet dies, dass alle Einträge, die einen Behälter mit lokaler Tiefe t' referenzieren, in den ersten t' Bits übereinstimmen. Da alle Bitmuster bis zur Länge t in dem Directory aufgeführt sind, unterscheiden sich diese Einträge in den letzten $t - t'$ Bits.

⇒ Es gibt somit $2^{t-t'}$ Einträge im Verzeichnis, die auf denselben Behälter mit lokaler Tiefe t' verweisen.

Aufgabe 3

An das Informationssystem der Universitätsverwaltung wird folgende Anfrage gestellt:

```

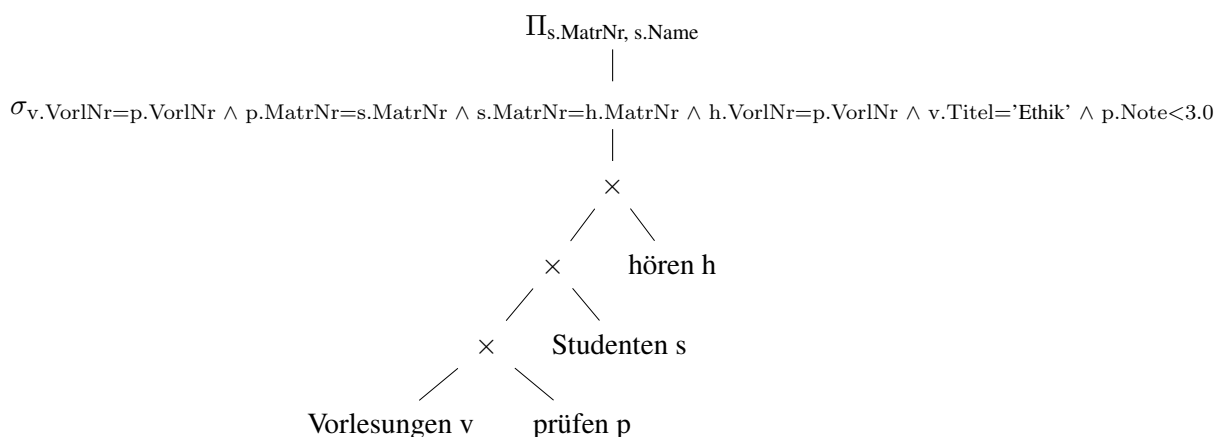
select s.MatrNr, s.Name
from Vorlesungen v, prüfen p, Studenten s, hören h
where v.VorlNr = p.VorlNr and p.MatrNr = s.MatrNr
and s.MatrNr = h.MatrNr and h.VorlNr = p.VorlNr
and v.Titel = 'Ethik' and p.Note < 3.0;

```

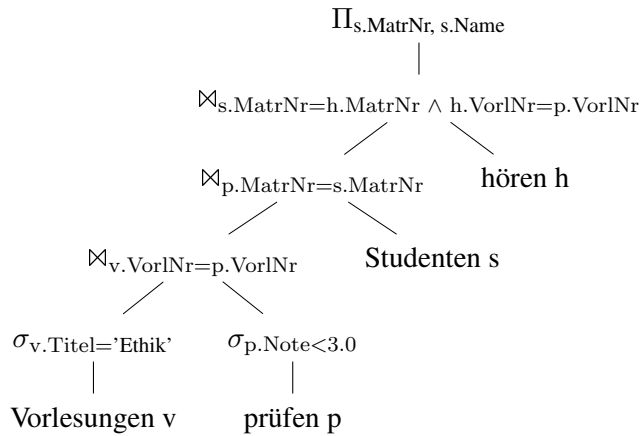
- (a) Erläutern Sie (in natürlicher Sprache) das Ergebnis dieser Anfrage.
- (b) Geben Sie die kanonische Übersetzung dieser Anfrage in die relationale Algebra an. Verwenden Sie zur Darstellung des relationalen Algebraausdrucks die Baumdarstellung.
- (c) Optimieren Sie Ihren relationalen Algebraausdruck (logisch). Dokumentieren Sie Ihre Schritte bei der Optimierung.

Lösung

- (a) Finde die Studenten, die Ethik gehört haben und darüber eine Prüfung mit einer Note besser als drei abgelegt haben.
- (b) Operatorbaum



- (c) Optimieren:
Schritt 1: Aufbrechen der Selektionen und Zusammenfassen zu Joins



Schritt 2: Anpassen der Join-Reihenfolge

Es ist davon auszugehen, dass $|v| < |s| < |p| < |h|$

Damit ist die Join-Reihenfolge schon passend!

Schritt 3: Einfügen weiterer Projektionen, z.B.

