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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS20/21

Christoph Anneser, Josef Schmeißer, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2021/grundlagen/

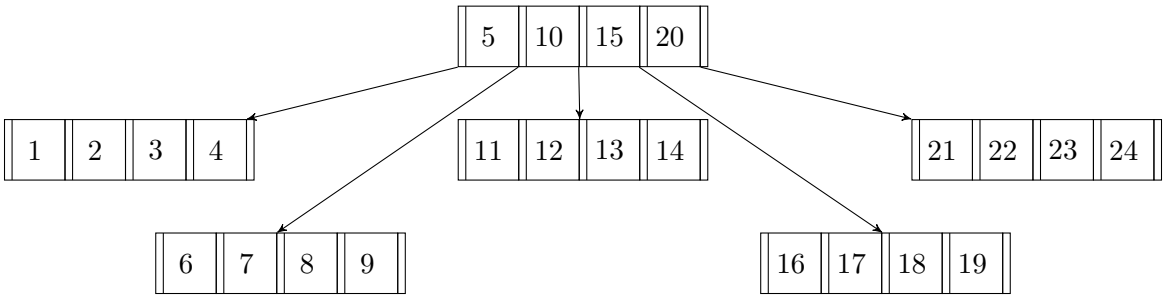
Blatt Nr. 09

Hausaufgabe 1

Geben Sie eine Permutation der Zahlen 1 bis 24 an, so dass beim Einfügen dieser Zahlenfolge in einen (anfangs leeren) B-Baum mit Grad $k = 2$ ein Baum minimaler Höhe entsteht. Skizzieren Sie den finalen Baum.

Lösung:

Damit der Baum eine minimale Höhe erreicht, muss der Füllgrad aller Knoten möglichst hoch sein. Bei $k = 2$ passen vier Elemente in jeden Knoten. Bei 24 einzufügenden Zahlen bedeutet das also, dass ein minimaler Baum insgesamt aus $\frac{24}{4} = 6$ Knoten besteht, die alle komplett gefüllt sind. Aufgrund der Invarianzen eines B-Baums muss dies ein Baum der Höhe 2 sein. Der einzig mögliche minimale Baum ist der folgende:



Damit die Zahlen 5, 10, 15 und 20 in die Wurzel geschrieben werden, muss die Einfügereihenfolge so gewählt werden, dass beim Teilen eines Knotens wegen Überlauf genau diese vier Zahlen als mittlere Elemente verwendet werden. Außerdem muss darauf geachtet werden, dass die Knoten nach dem Aufteilen wieder voll gefüllt werden.

Beispielsweise kann beim Einfügen mit den Zahlen 1, 2, 5, 6 und 7 in dieser Reihenfolge angefangen werden, wodurch die 5 dann in die Wurzel geschoben wird. Um die 10 in die Wurzel zu schreiben, können dann die Zahlen 10, 11, 12 eingefügt werden. Für die 15 dementsprechend die Zahlen 15, 16, 17 sowie für die 20 die Zahlen 20, 21, 22. Die übrigen Zahlen können dann in beliebiger Reihenfolge eingefügt werden.

Eine mögliche Permutation der Einfügereihenfolge, um einen Baum minimaler Höhe zu erhalten ist also: 1, 2, 5, 6, 7, 10, 11, 12, 15, 16, 17, 20, 21, 22, 3, 4, 8, 9, 13, 14, 18, 19, 23, 24.

Hausaufgabe 2

Fügen Sie die folgenden Tupel in eine anfangs leere erweiterbare Hashtabelle, welche 2 Einträge pro Bucket aufnehmen kann, ein. Dabei soll die Matrikelnummer als Suchschlüssel verwendet werden.

MatrNr	Name
2	Müller
8	Schmidt
19	Fischer
16	Huber
20	Bauer
34	Schneider
30	Wagner

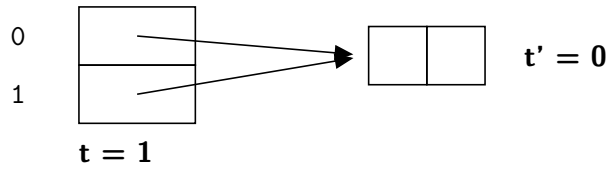
Verwenden Sie als Hashverfahren die inverse binäre Repräsentation der Matrikelnummer, wie in der Vorlesung beschrieben.

Lösung:

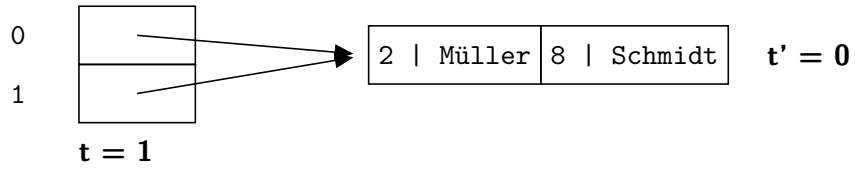
Zunächst errechnen wir den Hashwert für alle Studierende: Die inverse binäre Repräsentation der Matrikelnummer.

MatrNr	Name	Binär	Invers Binär
2	Müller	000010	010000
8	Schmidt	001000	000100
19	Fischer	010011	110010
16	Huber	010000	000010
20	Bauer	010100	001010
34	Schneider	100010	010001
30	Wagner	011110	011110

Zunächst eine leere erweiterbare Hashtabelle:

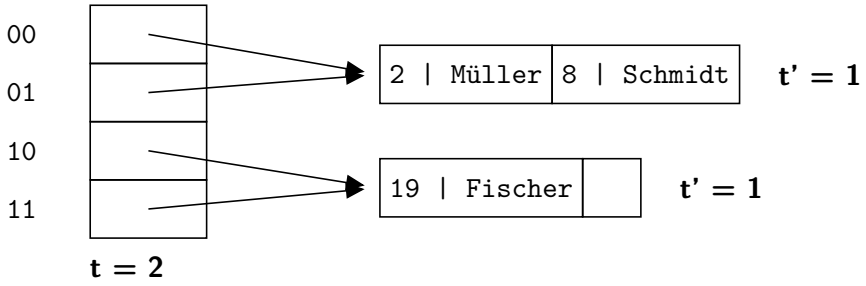


Wir fügen nun die ersten zwei Einträge ein, wonach die Hashtabelle wie folgt aussieht:

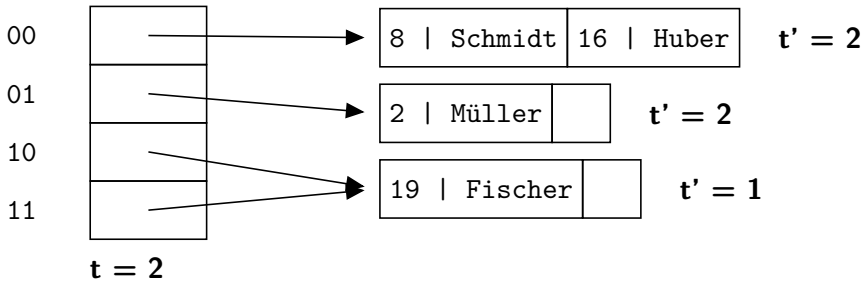


Der nächste Eintrag führt zu einem Überlauf. Da $t' < t$, können wir das Bucket teilen. Wir erhalten somit folgende Hashtabelle:

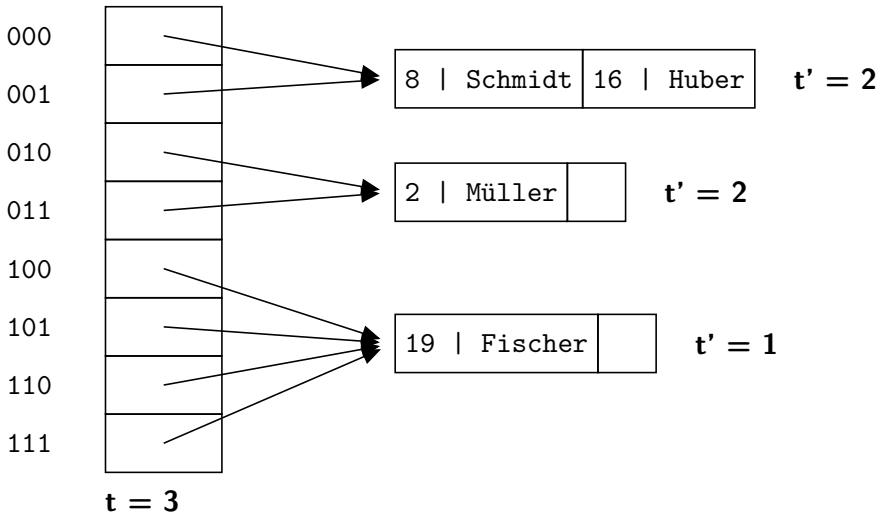
Der nächste Eintrag führt zu einem erneuten Überlauf! Leider in dem Bucket, für das $t = t'$ gilt. Wir müssen also die globale Tiefe erhöhen und erhalten somit folgende Hashtabelle:



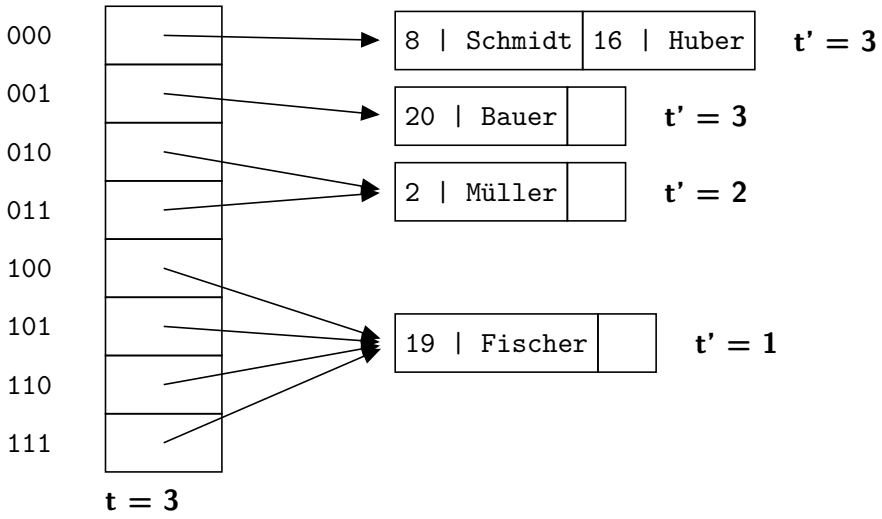
Nun können wir das erste Bucket splitten und unseren neuen Wert eintragen:



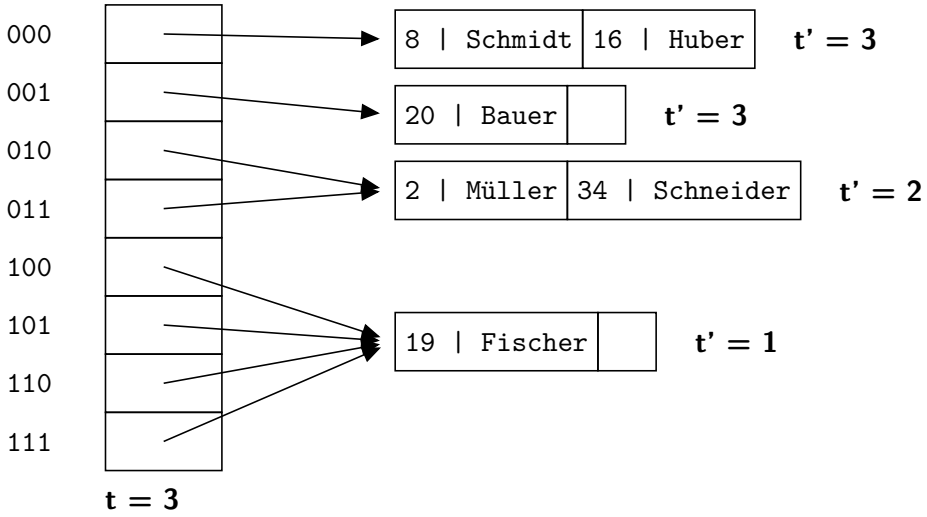
Bauer muss dem obersten Bucket hinzugefügt werden. Da dort kein Platz ist und $t = t'$ gilt, müssen wir erneut zuerst die globale Tiefe erhöhen.



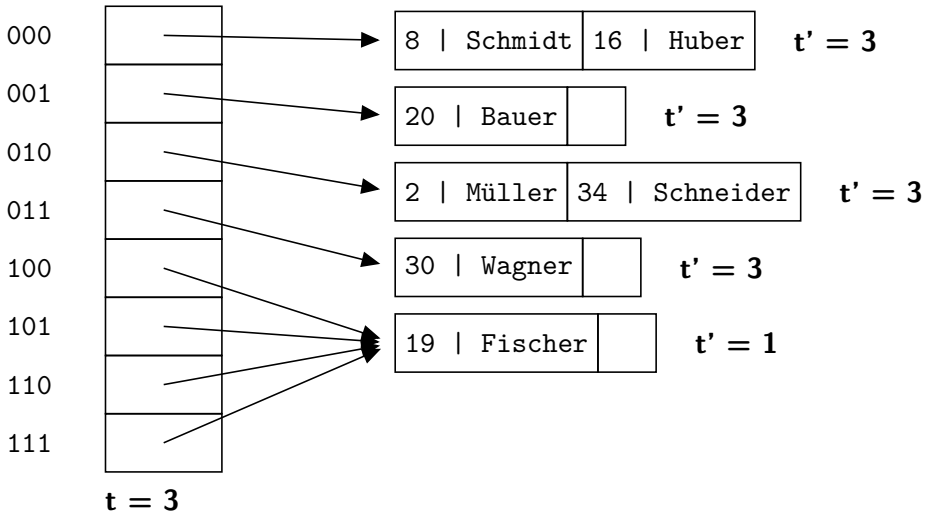
Wir können nun das oberste Bucket splitten, und Bauer hinzufügen:



Schneider passt in ein halbvolles Bucket:



Für Wagner müssen wir zuletzt noch einmal die lokale Tiefe erhöhen.



Hausaufgabe 3

Gegeben sei eine erweiterbare Hashtabelle mit globaler Tiefe t. Wie viele Verweise zeigen vom Verzeichnis auf einen Behälter mit lokaler Tiefe t'?

Lösung:

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.

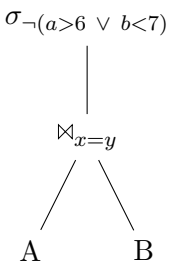
$\Rightarrow$ Es gibt somit $2^{t-t'}$ Einträge im Verzeichnis, die auf denselben Behälter mit lokaler Tiefe t' verweisen.

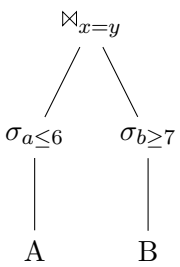
Hausaufgabe 4

Gegeben seien die Relationen

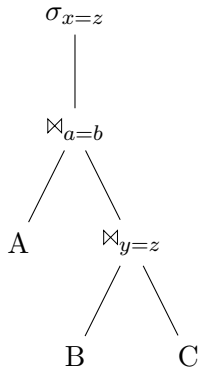
$$A : \{[a, x]\}$$
$$B : \{[b, y]\}$$
$$C : \{[c, z]\}$$

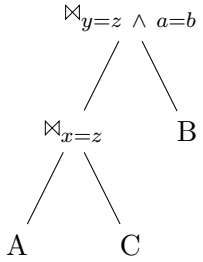
Im Folgenden sehen Sie jeweils zwei Operatorbäume der relationen Algebra. Sind diese äquivalent zueinander? Beweisen oder widerlegen Sie mithilfe der zwölf äquivalenzerhaltenden Transformationsregeln aus der Vorlesung.

a) Ist 

äquivalent zu 

?

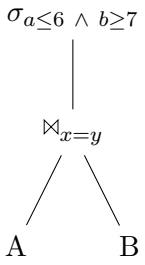
b) Ist 

äquivalent zu 

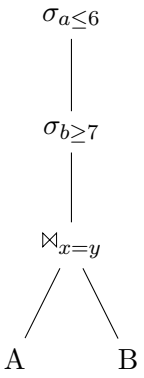
?

Lösung:

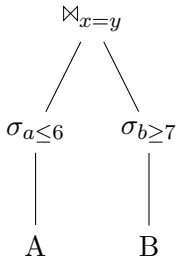
a)



(1)



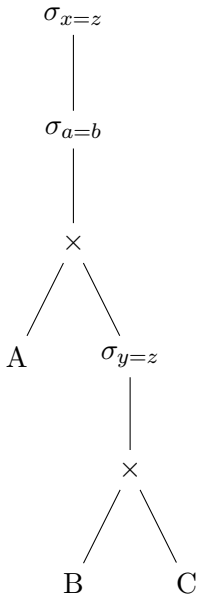
(2)



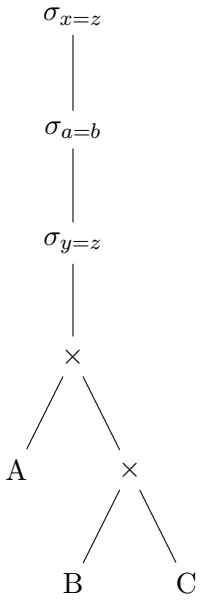
(3)

Das DeMorgansche Gesetz (Regel 11 im Foliensatz) erlaubt uns, die Disjunktion in der Selektion in eine Konjunktion umzuformen (1). Diese Konjunktion können wir nun nach Regel 1 im Foliensatz in zwei getrennte Selektionen aufbrechen (2). Diese Selektionen können wir nun nach unten propagieren (Regel 6 im Foliensatz), da a ein Attribut von A und b ein Attribut von B ist (3). Da wir lediglich Äquivalenzumformungen verwendet haben, ist dieser Operatorbaum äquivalent zum Ausgangsbaum.

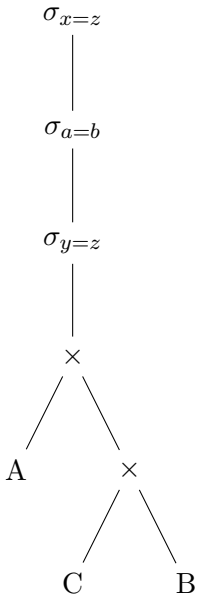
b)



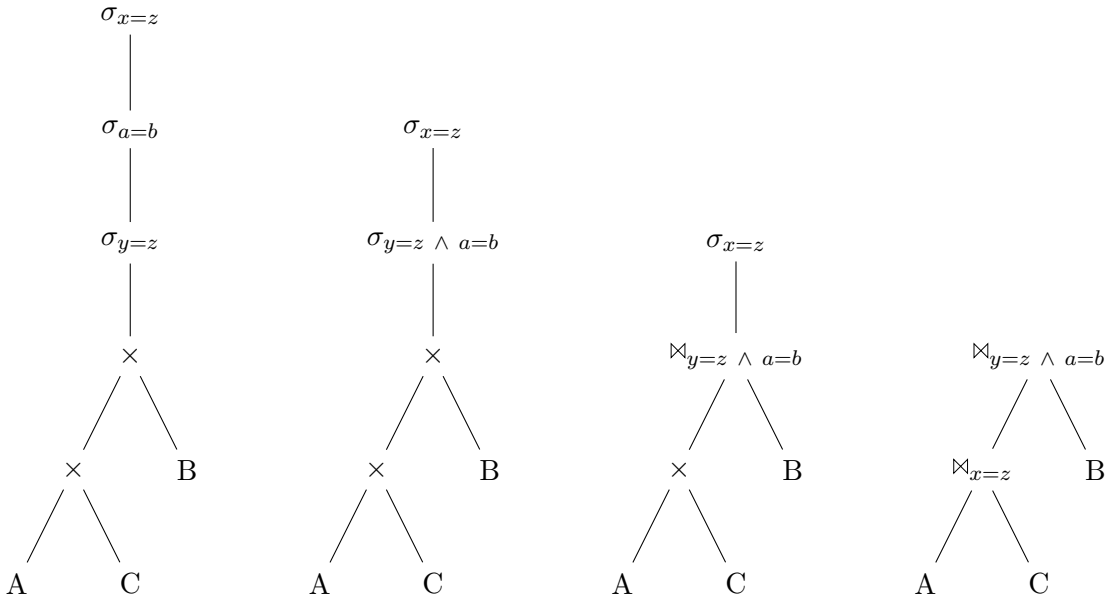
(1)



(2)



(3)



Nach Regel 12 können wir einen Join in ein kartesisches Produktund eine Selektion transformieren (1). Regel 6 erlaubt es uns dann, alle Selektionen nach oben zu propagieren (2). Wir können nun erst die Kommutativität (Regel 5) (3) und dann die Assoziativität (Regel 8) (4) des Kreuzproduktes verwenden. Nach Regel 1 können wir zwei Selektionen zusammenführen (5) und nach Regel 12 erneut in einen Join umwandeln (6) Ein erneutes Anwenden von Regeln 6 und 12 führt uns zum Ziel (7). Auch diese beiden Operatorbäume sind äquivalent.

Hausaufgabe 5

Gegeben sei die folgende SQL-Anfrage:

```
select distinct a.PersNr, a.Name
from Assistenten a, Studenten s, pruefen p
where s.MatrNr = p.MatrNr
  and a.Boss = p.PersNr
  and s.Name = 'Jonas';
```

Geben Sie die kanonische Übersetzung dieser Anfrage in die relationale Algebra an. Verwenden Sie zur Darstellung des relationalen Algebraausdrucks die Baumdarstellung.

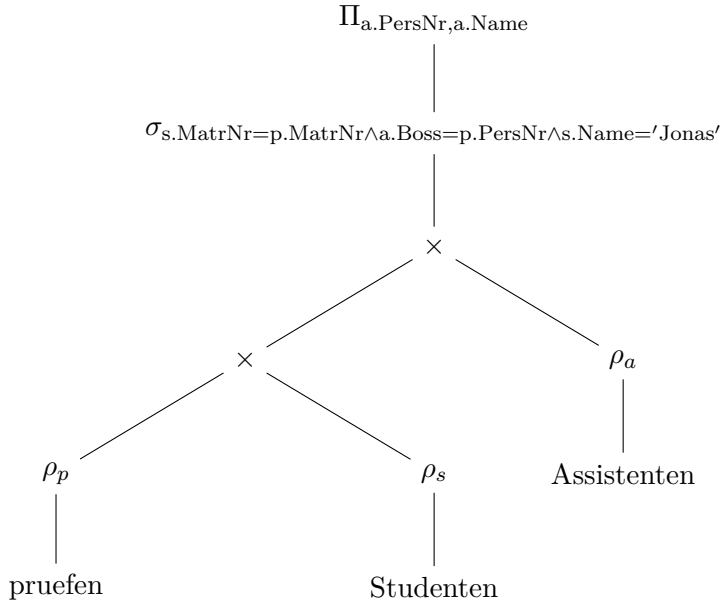
Optimieren Sie Ihren relationalen Algebraausdruck logisch. Gehen Sie dabei von **realistischen** Kardinalitäten für die relevanten Relationen aus.

Verwenden Sie hierfür die folgenden aus der Vorlesung bekannten Optimierungstechniken:

- Aufbrechen von Selektionen
- Verschieben von Selektionen nach “unten” im Plan
- Zusammenfassen von Selektionen und Kreuzprodukten zu Joins
- Bestimmung der Joinreihenfolge

Lösung:

Kanonische Übersetzung:



realistische Kardinalitäten: nur ein Student mit dem Namen 'Jonas', viel mehr Assistenten, deshalb Studenten zuerst, dann über pruefen mit Assistenten joinen

logische Optimierung: Selektionen ganz nach unten, Joins statt Kreuzprodukte & in richtiger Reihenfolge