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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS17/18

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1718/grundlagen/

Blatt Nr. 11

Hausaufgabe 1

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 2

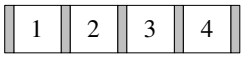
(a) Fügen Sie in einen anfänglich leeren B$^+$-Baum mit $k = 3$ und $k^* = 2$ die Zahlen eins bis fünfundzwanzig in aufsteigender Reihenfolge ein. In den Blattknoten werden TIDs verwendet. Was sind TIDs, wann lohnt sich ihre Verwendung, was ist die Alternative zu TIDs?

(b) Erläutern Sie die Vorgehensweise bei der Bearbeitung der folgenden Anfrage „Finde alle Datensätze mit einem Schlüsselwert zwischen 5 und 15.“

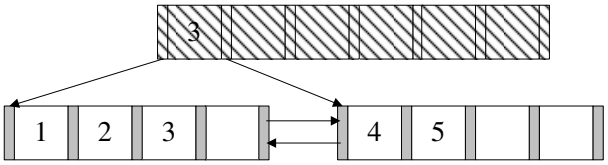
Lösung:

(a) Bei B$^+$-Bäumen unterscheiden sich die Kapazitäten von inneren Knoten und Blattknoten (angegeben durch k und k^*. Im Folgenden werden innere Knoten zur leichteren Unterscheidung schraffiert.

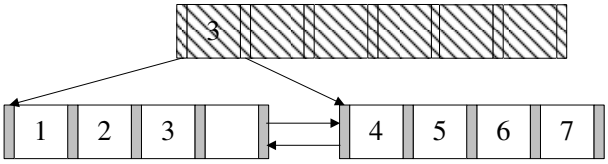
Nachdem man die Zahlen 1 bis 4 eingefügt hat, liegt folgender B-Baum vor (da die Wurzel in diesem Fall ein Blatt ist können höchstens 4 Einträge eingefügt werden):



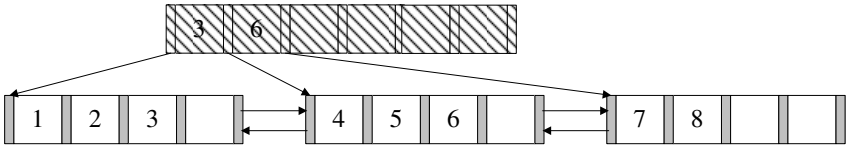
Beim Einfügen von 5 wird der Knoten gespalten. Der Referenzschlüssel 3 wandert in die neue Wurzel, deren Kapazität 6 ist. Die neuen Blattknoten haben eine Kapazität von 4 und sind untereinander verlinkt.



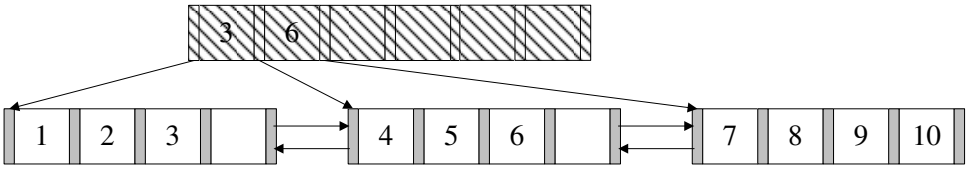
Die nächsten beiden Einträge lassen sich wieder ohne Probleme einfügen.



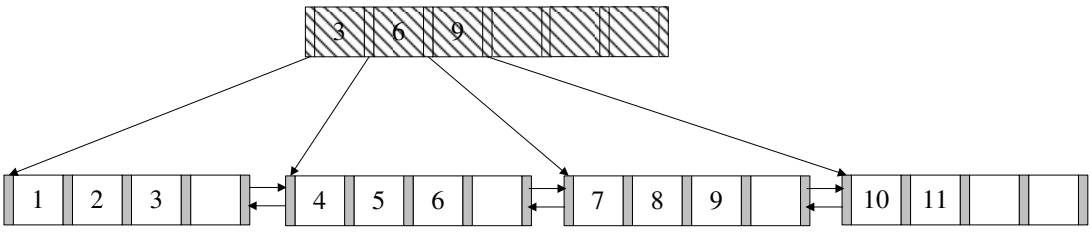
Beim Einfügen der 8 kommt es erneut zum Überlauf. Die 6 wandert in die Wurzel.



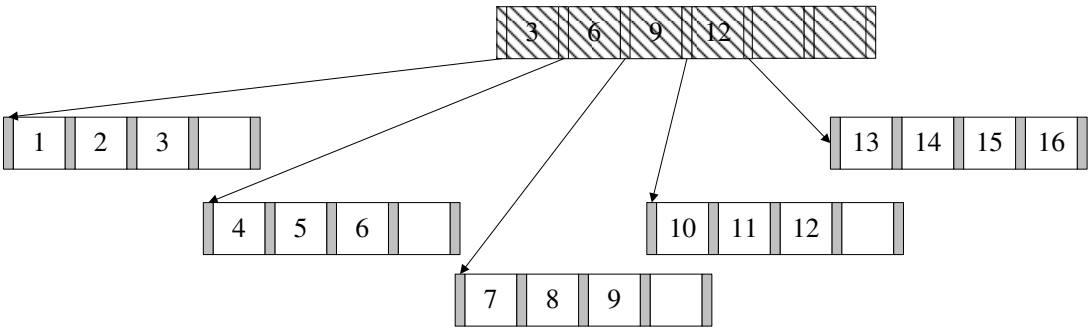
9 und 10 lassen sich wieder ohne Probleme einfügen. Bei 11 kommt es zum Überlauf.



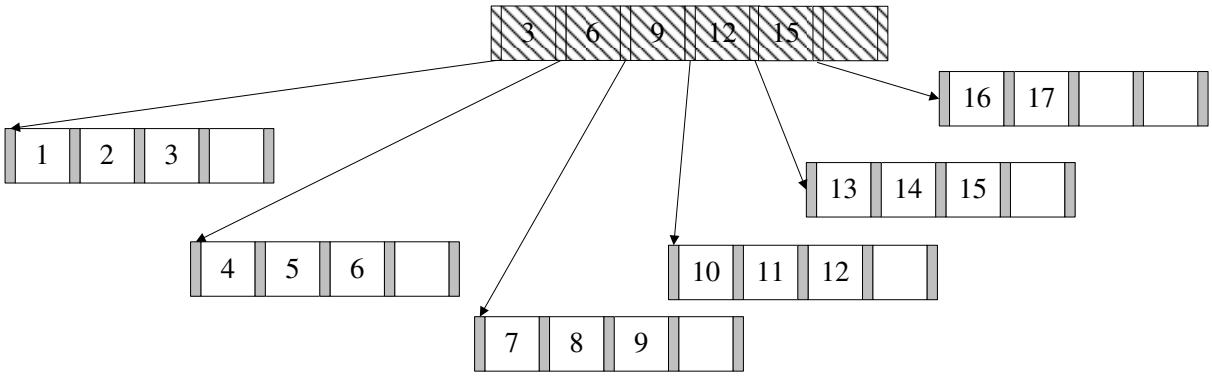
Nach dem Aufspalten erhält man dann:



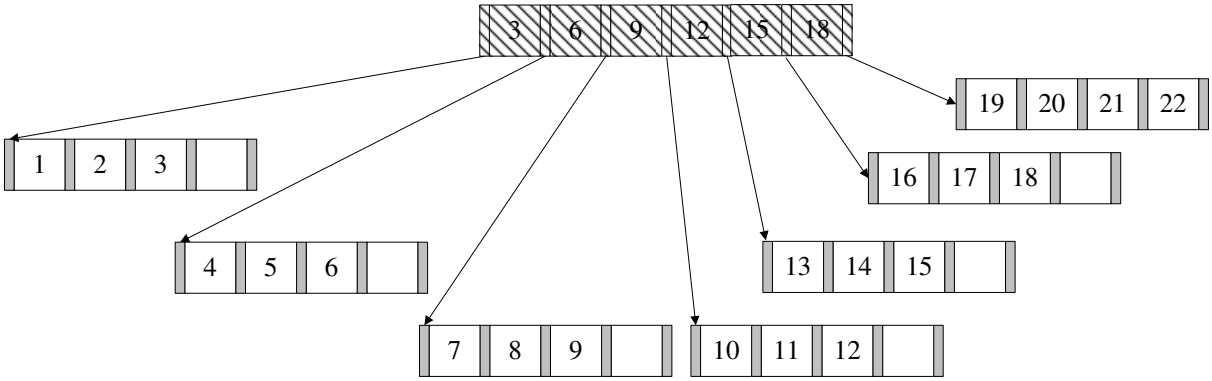
Es werden nun die nächsten Zahlen bis 16 analog eingefügt. (Die Pointer zwischen den Blattknoten existieren weiterhin, werden hier jedoch nicht mehr dargestellt.)



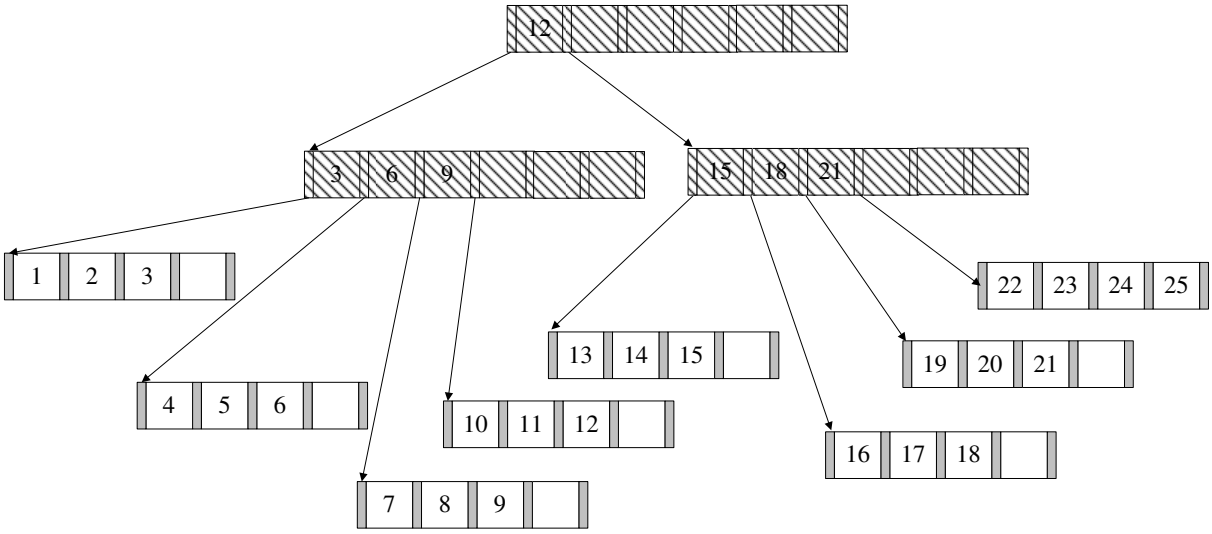
Bei 17 kommt es dann wieder zum Überlauf.



Die nächsten Zahlen werden wieder analog eingefügt. Bei 20 kommt es zum Überlauf.

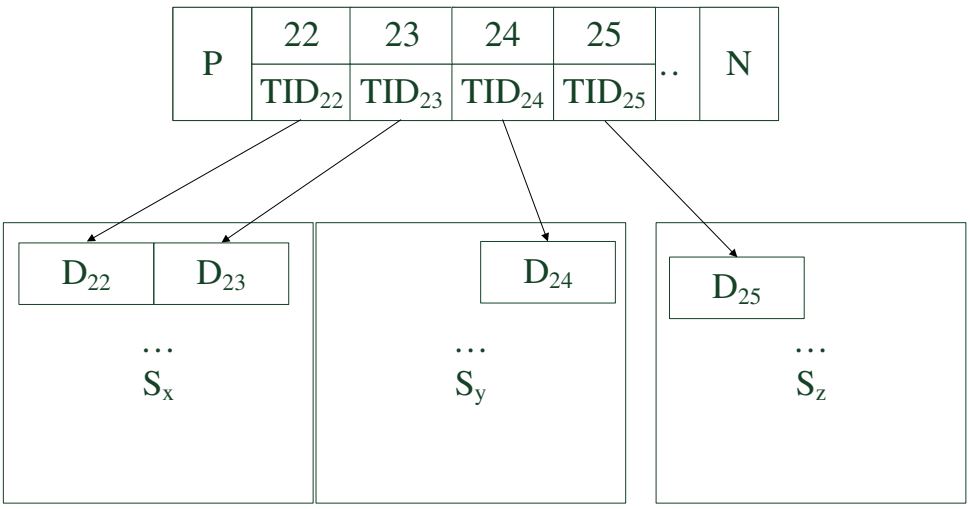


Nun fügt man noch die Zahlen 21 und 22 ein. Bei 23 kommt es erneut zum Überlauf. Die 21 wird in den Wurzelknoten kopiert, wodurch auch hier ein Überlauf stattfindet, so dass der Baum in seiner Höhe wächst. Nach dem Einfügen von 24 und 25 sieht der Baum wie folgt aus:



In den Blättern können nun Datensätze oder TIDs gespeichert werden. TIDs sind „Zeiger“ auf Tuple. Werden TIDs vewendet, wird der Index kompakter und dadurch der Baum weniger hoch. Dafür ist eine weitere Indirektion zum Auffinden der Daten erforderlich, die bei der Suche nach einem Tupel verfolgt werden muss. Der letzte

Knoten würde (im Detail) so aussehen:



S_x, S_y und S_z sind dabei beliebige Seiten im Speicher.

(b) Um eine Bereichsanfrage zu beantworten geht man wie folgt vor:

1. Zunächst sucht man nach der unteren Schranke der Anfrage, in diesem Fall nach der 5. Dies geschieht genauso wie beim B-Baum. Die Suche endet in einem Blattknoten.
2. Anschließend liest man alle sukzessiven Einträge bis zur oberen Schranke der Anfrage, in diesem Fall 15. Hierbei nutzt man die *Next*-Verlinkungen der Blattknoten untereinander.

Natürlich wäre es umgekehrt auch möglich, nach der oberen Schranke zu suchen und dann den *Previous*-Pointern zu folgen.

Hausaufgabe 3

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.

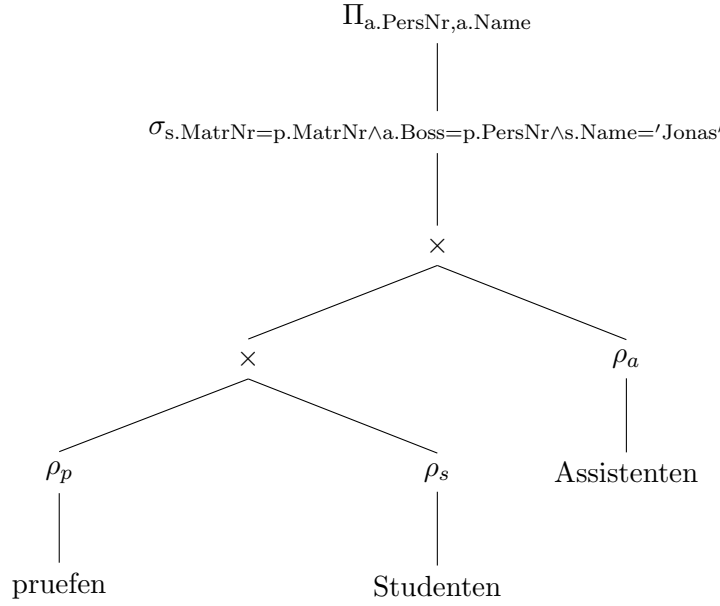
Optimieren Sie Ihren relationalen Algebraausdruck logisch. Gehen Sie dabei von **realistischen** Kardinalitäten für die relevanten Relationen aus.

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

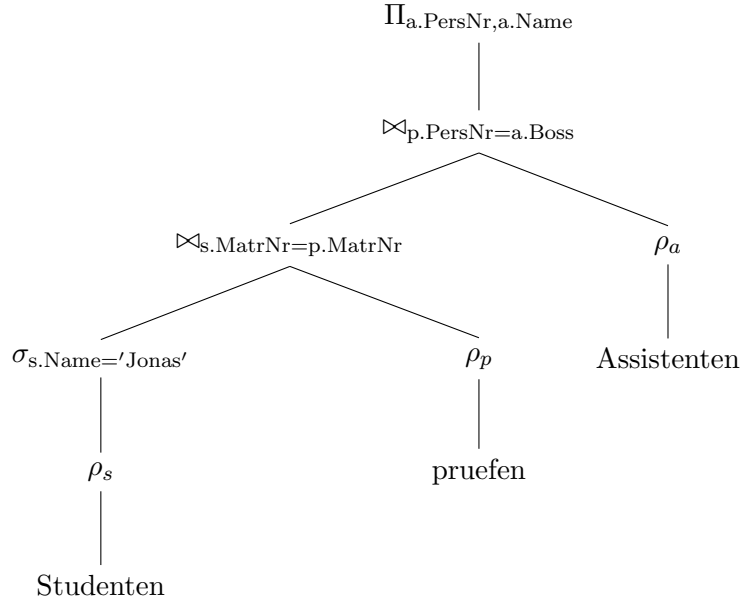
Lösung:

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



Hausaufgabe 4

Betrachten Sie ein abstraktes Relationenschema $\mathcal{R} = \{A, B, C, D, E, F, G\}$ mit den FDs

$$\begin{aligned} A &\rightarrow BC \\ DE &\rightarrow B \\ F &\rightarrow A \\ E &\rightarrow BF \\ A &\rightarrow DE \\ C &\rightarrow A. \end{aligned}$$

Überführen Sie die Relation verlustfrei und abhängigkeitsbewahrend in die dritte Normalform.

Lösung:

Kanonische Überdeckung

Linksreduktion:

$$\begin{aligned} A &\rightarrow BC \\ \cancel{D}E &\rightarrow B \\ F &\rightarrow A \\ E &\rightarrow BF \\ A &\rightarrow DE \\ C &\rightarrow A \end{aligned}$$

Rechtsreduktion:

$$\begin{aligned} A &\rightarrow \cancel{B}C \\ E &\rightarrow \emptyset \\ F &\rightarrow A \\ E &\rightarrow BF \\ A &\rightarrow DE \\ C &\rightarrow A \end{aligned}$$

Zusammenfassen der FDs mit gleichen linken Seiten:

$$\begin{aligned} A &\rightarrow CDE \\ E &\rightarrow \emptyset \\ F &\rightarrow A \\ E &\rightarrow BF \\ C &\rightarrow A \end{aligned}$$

Entfernen von FDs mit leerer Menge auf der rechten Seite:

$$\begin{aligned} A &\rightarrow CDE \\ F &\rightarrow A \\ E &\rightarrow BF \\ C &\rightarrow A \end{aligned}$$

Synthesealgorithmus

Zerlegung anhand der kanonischen Überdeckung:

$$\begin{aligned}\mathcal{R}_1 &= \{\underline{A}, C, D, E\}\\ \mathcal{R}_2 &= \{A, \underline{F}\}\\ \mathcal{R}_3 &= \{B, \underline{E}, F\}\\ \mathcal{R}_4 &= \{A, \underline{C}\}\end{aligned}$$

Kandidatenschlüssel bestimmen:

$$\kappa_1 = \{A, G\},\ \kappa_2 = \{F, G\},\ \kappa_3 = \{C, G\},\ \kappa_4 = \{E, G\}$$

Kandidatenschlüssel der Relationenzerlegung hinzufügen:

Da keiner der Kandidatenschlüssel in der Zerlegung enthalten ist,
wähle ein beliebiges κ_i und erstelle $\mathcal{R}_\kappa$, z. B. $\mathcal{R}_\kappa = \{\underline{A, G}\}$.

Redundanzen entfernen:

$\mathcal{R}_4$ verwerfen, da $\mathcal{R}_4 \subseteq \mathcal{R}_1$

Ergebnis:

$$\{\underline{A}, C, D, E\},\ \{A, \underline{F}\},\ \{B, \underline{E}, F\},\ \{\underline{A, G}\}$$