



Übung zur Vorlesung *Grundlagen: Datenbanken* im WS16/17

Harald Lang, Linnea Passing (gdb@in.tum.de)

<http://www-db.in.tum.de/teaching/ws1617/grundlagen/>

Blatt Nr. 12

Hausaufgabe 1

- (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.“

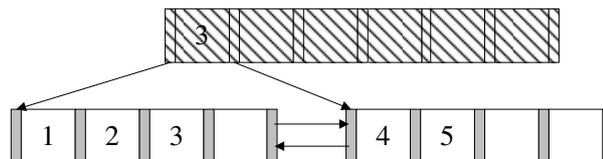
Lösung:

- (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.

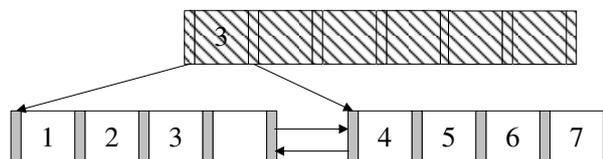
Nachdem man die Zahlen 1 bis 4 eingefügt hat, liegt folgender B-Baum vor (da die Wurzel in diesem Fall ein Blatt ist können höchstens 4 Einträge eingefügt werden):



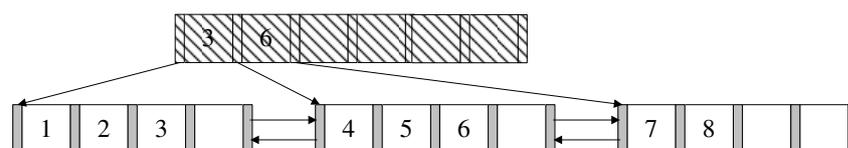
Beim Einfügen von 5 wird der Knoten gespalten. Der Referenzschlüssel 3 wandert in die neue Wurzel, deren Kapazität 6 ist. Die neuen Blattknoten haben eine Kapazität von 4 und sind untereinander verlinkt.



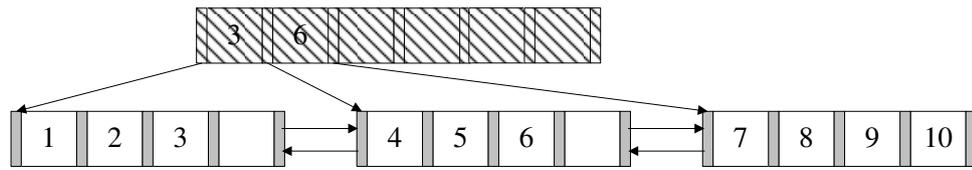
Die nächsten beiden Einträge lassen sich wieder ohne Probleme einfügen.



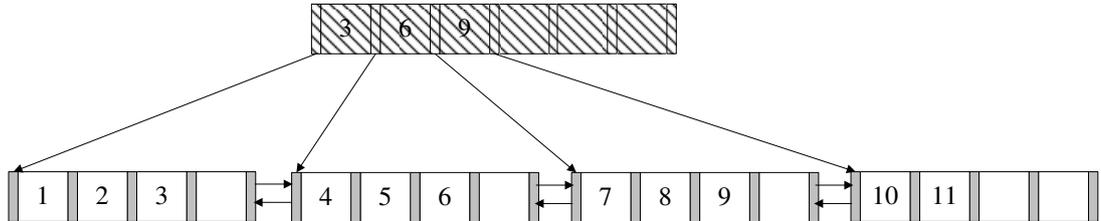
Beim Einfügen der 8 kommt es erneut zum Überlauf. Die 6 wandert in die Wurzel.



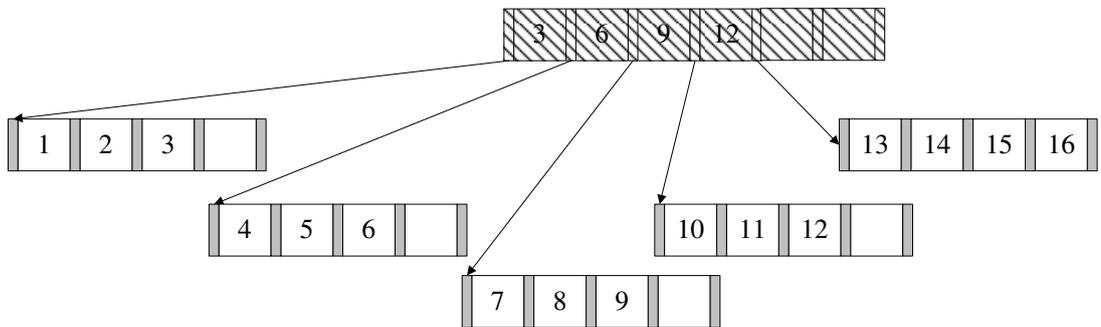
9 und 10 lassen sich wieder ohne Probleme einfügen. Bei 11 kommt es zum Überlauf.



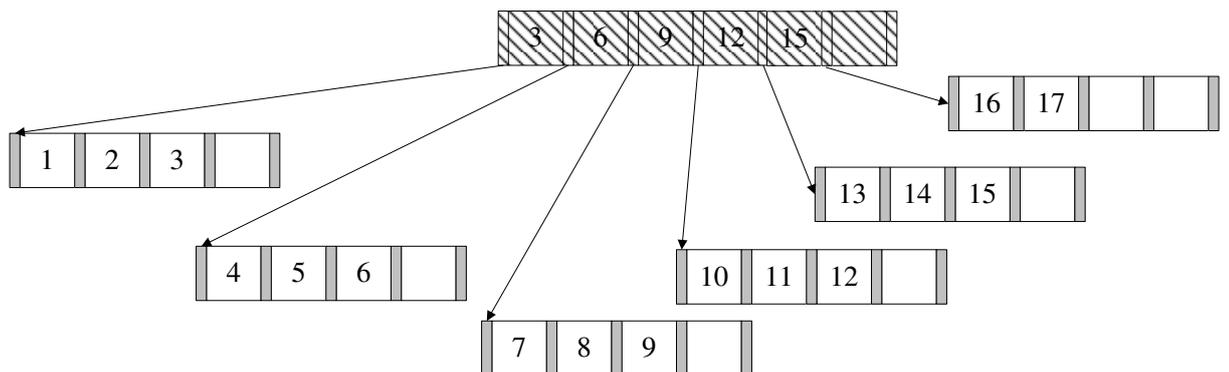
Nach dem Aufspalten erhält man dann:



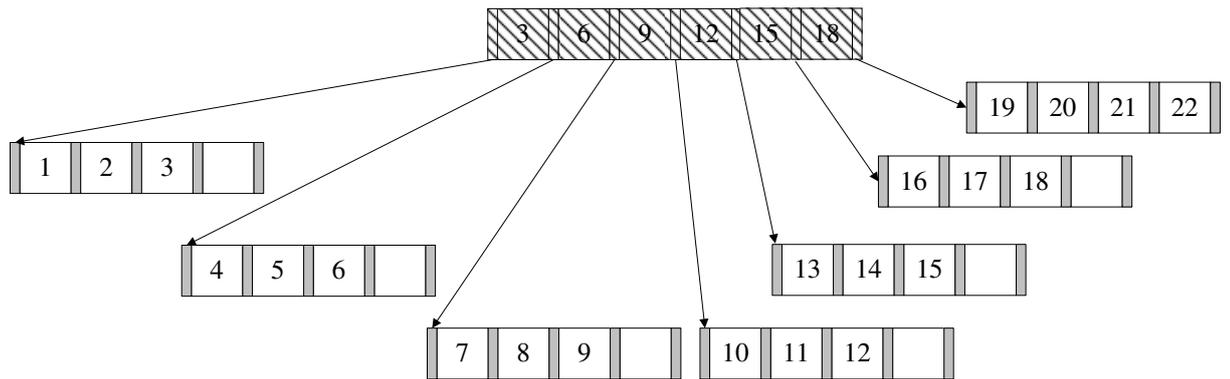
Es werden nun die nächsten Zahlen bis 16 analog eingefügt. (Die Pointer zwischen den Blattknoten existieren weiterhin, werden hier jedoch nicht mehr dargestellt.)



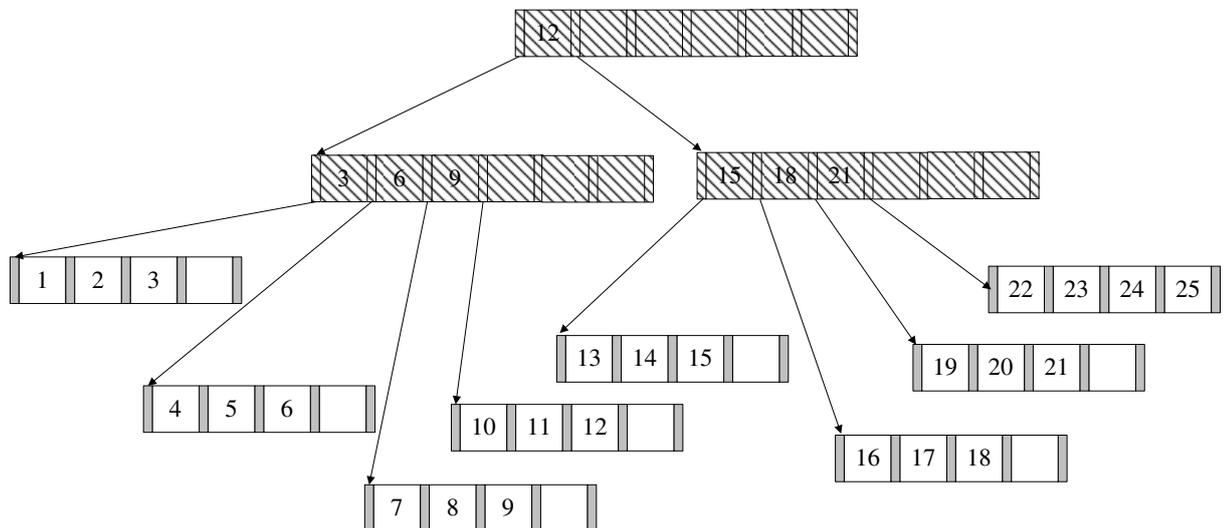
Bei 17 kommt es dann wieder zum Überlauf.



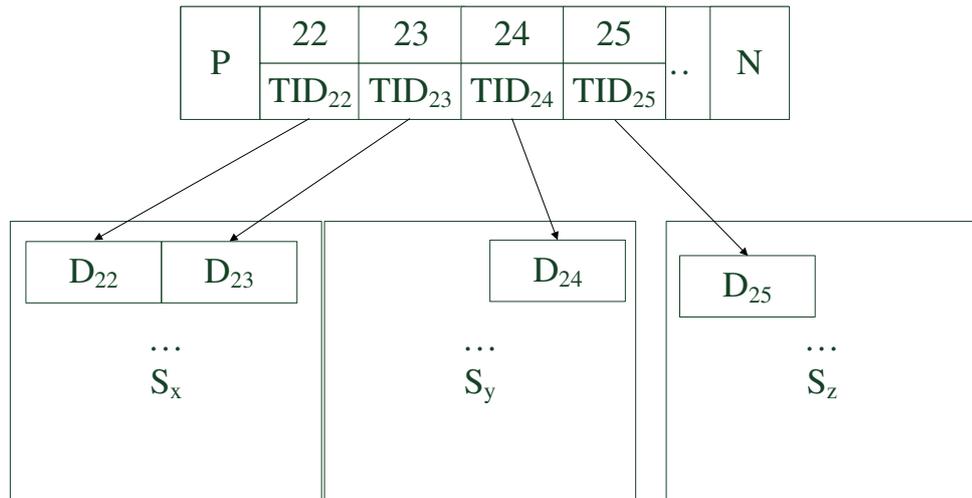
Die nächsten Zahlen werden wieder analog eingefügt. Bei 20 kommt es zum Überlauf.



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 verwendet, 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 Tuple 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 2

Es sollen alle ca. 10 Milliarden Menschen in einer erweiterbaren Hashtabelle verwaltet werden. In jede Seite passen ca. 200 Einträge, durchschnittlich sind die Seiten halb voll. Je Verweis werden 4 Byte benötigt, da die Musterlösung aus einer Zeit stammt, in der es defakto nur Maschinen mit 32 bit CPU Architektur gab. Wie viel Speicherplatz verbraucht das Verzeichnis mindestens?

Lösung:

Das Verzeichnis enthält die Verweise auf alle Seiten (= Buckets), in dem die Einträge gehalten werden. Da pro Seite durchschnittlich 100 Einträge Platz haben, benötigen wir insgesamt $10^{10}/100 = 10^8$ Seiten. Um 10^8 Seiten zu referenzieren benötigen wir mindestens $\log_2 10^8$ Bits. Da dies eine positive ganze Zahl sein muss, ist die Anzahl der benötigten Bits $\lceil \log_2 10^8 \rceil$. Hiermit können $2^{\lceil \log_2 10^8 \rceil}$ Verweise im Verzeichnis abgelegt werden, da die Anzahl der Verweise in einem Verzeichnis immer einer 2er-Potenz entspricht. Pro Verweis werden 4 Byte benötigt, so dass das Verzeichnis eine Größe von $2^{\lceil \log_2 10^8 \rceil} \cdot 4$ Byte, also ungefähr 512 MB hat.

Hausaufgabe 3

Fügen Sie nacheinander die folgenden Einträge in eine anfangs leere erweiterbare Hashtabelle, welche 2 Einträge pro Bucket aufnehmen kann, ein. Es soll effizient nach der *KundenNr* gesucht werden können.

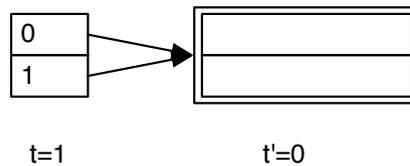
KundenNr	Name
10	Müller
25	Meier
30	Schmidt
18	Krause
40	Schulz
45	Kaufmann

Lösung:

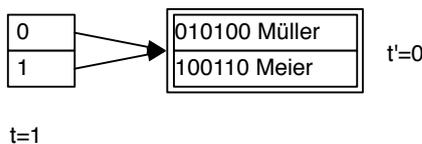
Werte mit binärer KundenNr sowie invers binärer Kundennummer, die für das Einfügen in den Hash genutzt wird:

KundenNr	Name	Binär	Umgekehrt Binär
10	Müller	001010	010100
25	Meier	011001	100110
30	Schmidt	011110	011110
18	Krause	010010	010010
40	Schulz	101000	000101
45	Kaufmann	101101	101101

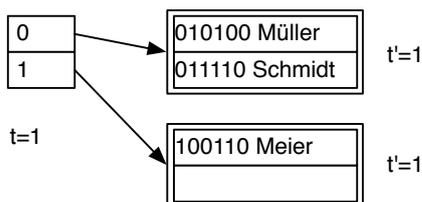
Zunächst eine leere erweiterbare Hashtabelle:



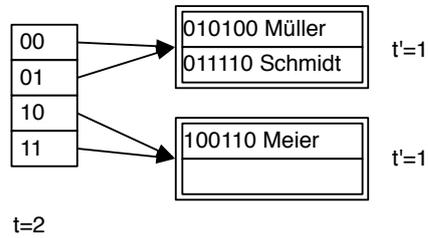
Wir fügen nun die ersten zwei Einträge ein, wonach die Hashtabelle wie folgt aussieht:



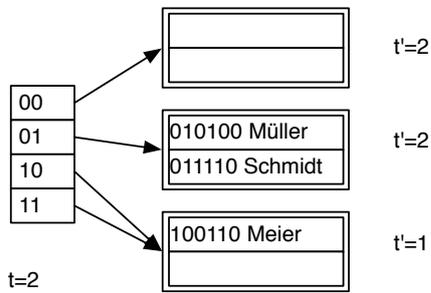
Der nächste Eintrag führt zu einem Überlauf. Da $t' < t$ können wir den Bucket teilen, dies führt zur folgenden Hashtabelle:



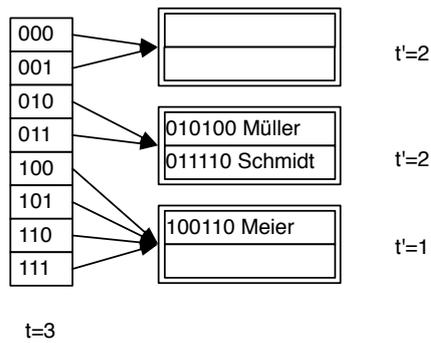
Das Einfügen von **Krause** führt erneut zu einem Überlauf, der Bucket kann aber nicht direkt geteilt werden, da $t' = t$ gilt. Das Verzeichnis wird verdoppelt:



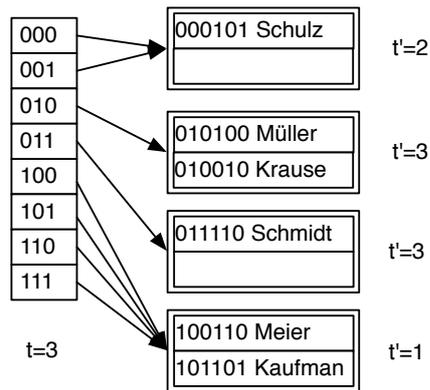
Nun kann der Bucket geteilt werden:



Das Einfügen ist leider immer noch nicht möglich und wieder hilt $t' = t$, weswegen das Verzeichnis erneut verdoppelt werden muss:



Nun kann der Bucket geteilt und alle Einträge eingefügt werden:



Hausaufgabe 4

Überführen Sie verlustlos in die 4. NF:

$$\mathcal{R} = \{A, B, C, D\} \text{ mit den FDs } F = \{C \rightarrow D, AB \rightarrow D, DA \rightarrow B, BA \rightarrow C\}.$$

Lösung:

Es kann entweder der BCNF oder der 4NF Algorithmus angewendet werden. Im ersten Fall erfolgt die Zerlegung wie zuvor geübt und das Ergebnis ist in BCNF. Da keine MVD gilt, die nicht aus FDs hergeleitet werden kann, ist die Relation damit automatisch auch in 4NF. Im anderen Fall gilt, dass alle MVDs auch FDs sind und man zieht diese für die Zerlegung heran.

Ein mögliches Ergebnis wäre:

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

Hausaufgabe 5 (optional)

Implementieren Sie in der Programmiersprache Ihrer Wahl einen B+ Baum *oder* eine Extendible Hashtable. Es sollten mindestens die Funktionen *insert* und *lookup* unterstützt werden. Zur Vereinfachung können Sie annehmen, dass lediglich Schlüssel-Wert-Paare bestehend aus Integern eingefügt werden.

Präsentieren Sie Ihre Implementierung in der Übung, diskutieren Sie Designentscheidungen die Sie getroffen haben und evtl. Herausforderungen die zu bewältigen waren.