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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS13/14

Henrik Mühe (muehe@in.tum.de)
http://www-db.in.tum.de/teaching/ws1314/dbsys/exercises/

Blatt Nr. 11

Hausaufgabe 1

Bringen Sie die folgende Relation verlustlos und abhängigkeitsbewahrend in die 3. NF.

$$R(A, B, C, D, E, F)$$

FDs:

- $AB \to CD$
- $ABC \to D$
- $E \to C$
- $D \to C$
- $CDE \to AB$

Beachten Sie, dass es für die Lösung notwendig ist, einen Kandidatenschlüssel zu ermitteln, jedoch nicht alle Kandidatenschlüssel. Beachten Sie außerdem, dass die Relation das Attribut F enthält, welches bei der Zerlegung nicht wegfallen darf.

Der Kandidatenschlüssel **muss** bestimmt werden, anders verliert man im dritten Schritt des Synthesealgorithmus potentiell Semantik und in diesem Fall insbesondere das Attribut F. Es müssen (da nicht explizit gefordert) jedoch nicht alle Kandidatenschlüssel sondern lediglich einer bestimmt werden. Ideales Verfahren hierzu ist, dass wir betrachten, welche Attribute nicht auf der rechten Seite von mindestens einer FD stehen. Diese **müssen** im Kandidatenschlüssel enthalten sein. Dies ist hier für E und F der Fall. Nun untersuchen wir, ob EF bereits ein Schlüssel für die Relation ist, womit wir dann bereits fertig wären. Wegen $AttrHuelle(FD, \{E, F\}) = \{E, F, C\} \neq R$ ist dies nicht der Fall. Wir versuchen nun möglichst intelligent ein Attribut hinzuzufügen, um Schlüsseleigenschaft zu erreichen. Da aus DE sofort AB folgt und E bereits definitiv im Schlüssel enthalten sein muss, fügen wir D hinzu. Damit gilt $AttrHuelle(FD, \{E, F, D\}) = \{E, F, D, C, A, B\} = R$, daher EDF ist Schlüssel. Insbesondere ist EFD auch Kandidatenschlüssel, da EF wie zuvor erklärt in jedem Schlüssel enthalten sein müssen, jedoch alleine kein Schlüssel sind, Reduktion nicht möglich, Schlüssel ist also Kandidatenschlüssel.

Ein weiterer Kandidatenschlüssel ist $ABEF$.

Nach dieser kurzen Überlegung muss lediglich der Synthesealgorithmus ausgeführt werden. Der Algorithmus ist in den Folien, im Buch und auf Wikipedia im Detail beschrieben. Sie werden massiv Punkte verlieren, wenn Sie ihn nicht beherrschen. Ein ausführliches Beispiel liegt im vorherigen Übungsblatt vor, an dieser Stelle sei lediglich zum Abgleich das Ergebnis angegeben:

Kanonische Überdeckung:

- $AB \to D$

- $E \to C$
- $D \to C$
- $DE \to AB$

Entstehende Relationen also:

- $R_1(A, B, D, E)$ mit zugeordneter FD $AB \to D$ sowie $DE \to AB$ (Zusammenfassung der Relationen, da eine in der anderen enthalten.)
- $R_2(E, C)$ mit zugeordneter FD $E \to C$
- $R_3(D, C)$ mit zugeordneter FD $D \to C$
- $R_4(E, F, D)$ ohne FD, für gewählten Kandidatenschlüssel eingefügt, da Kandidatenschlüssel nirgendwo enthalten.

Hausaufgabe 2

Geben Sie für jede der Normalformen 1NF, 2NF, und 3NF jeweils eine Relation mit FDs an, so dass die Relation in der gewünschten Normalform ist (und in keiner höheren). Geben Sie außerdem ein Beispiel für ein Relationenschema an, welches mindestens in BCNF ist. Ihres Vorgehens, so dass man die Methodik erkennen kann.

Für alle Normalformen betrachten wie die Relation $R(A, B, C, D)$.

- 1.NF:

  FDs:

  - $AB \to C$
  - $B \to D$

  Die Relation ist nicht mengenwertig, daher 1. NF. D ist lediglich von B abhängig, der Kandidatschlüssel ist aber AB, weswegen D nicht voll funktional vom Kandidatschlüssel abhängig ist, daher keine 2. NF.

- 2. NF:

  FDs:

  - $AB \to C$
  - $C \to D$

  Jedes Attribut der Relation ist voll funktional abhängig vom Kandidatschlüssel AB, daher 2.NF. Das Attribut D ist transitiv und nicht direkt vom Kandidatschlüssel abhängig, darum nicht 3. NF.

- 3. NF:

  FDs:

  - $AB \to CD$
  - $BC \to AD$
  - $D \to C$

  Für alle FDs gilt entweder, dass sie trivial ist, dass die linke Seite Superschlüssel ist oder dass die rechte Seite in einem Kandidatschlüssel enthalten ist, daher 3. NF. Bei der BCNF fällt die dritte erlaubte Art von FD weg, daher FDs müssen trivial sein oder ihre linke Seite Superschlüssel. Da die dritte FD des Beispiels dies verletzt ist die Relation nicht in BCNF und daher genau in 3. NF.

- BCNF:

  FDs:

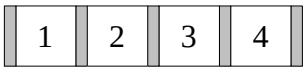
  - $AB \to CD$
  - $BC \to AD$
  - $D \to\!\!\to C$

  BCNF, da die BCNF verletzende FD aus dem Beispiel für 3. NF entfernt wurde.

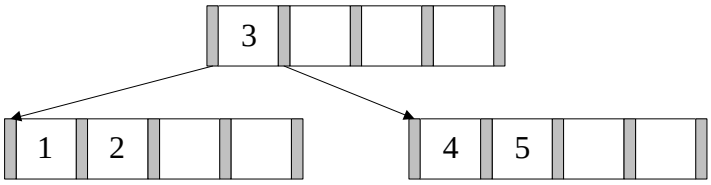
Hausaufgabe 3

Fügen Sie in einen anfänglich leeren B-Baum mit $k = 2$ die Zahlen eins bis zwanzig in aufsteigender Reihenfolge ein. Was fällt Ihnen dabei auf?

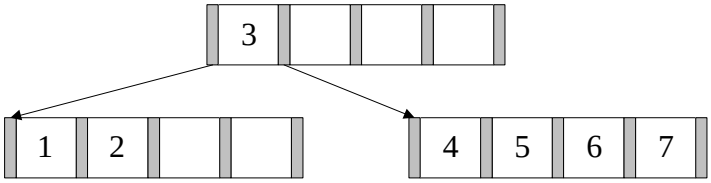
Nachdem man die Zahlen 1 bis 4 eingefügt hat, liegt folgender B-Baum vor:



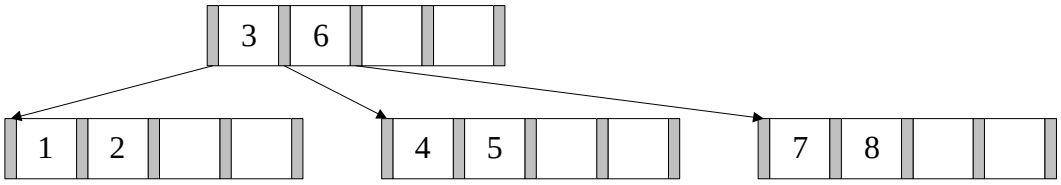
Beim Einfügen von 5 wird der Knoten gespalten und man erhält eine neue Wurzel.



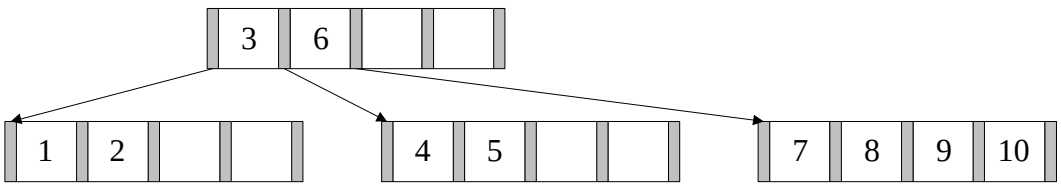
Die nächsten beiden Zahlen lassen sich wieder ohne Probleme einfügen.



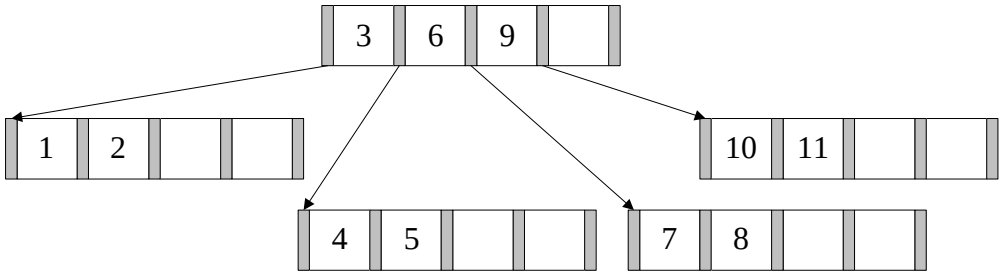
Beim Einfügen der 8 kommt es erneut zum Überlauf. Die 6 wandert in die Wurzel.



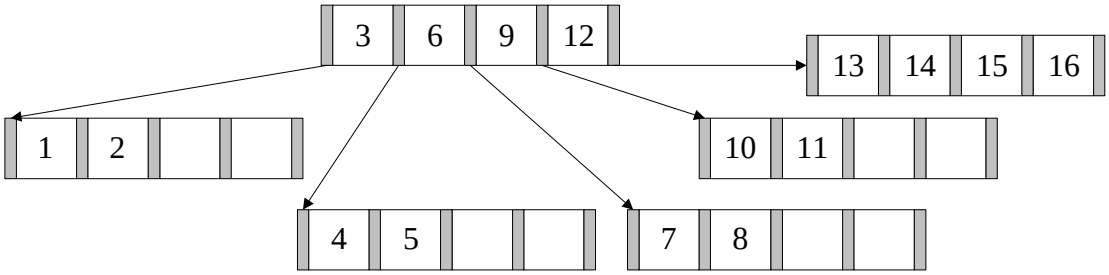
9 und 10 lassen sich wieder ohne Probleme einfügen. Bei 11 kommt es zum Überlauf.



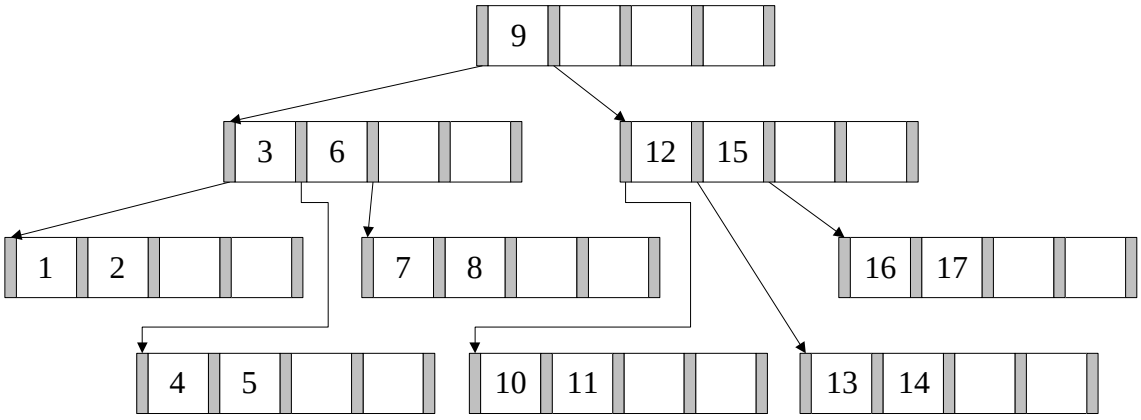
Nach dem Aufspalten erhält man dann:



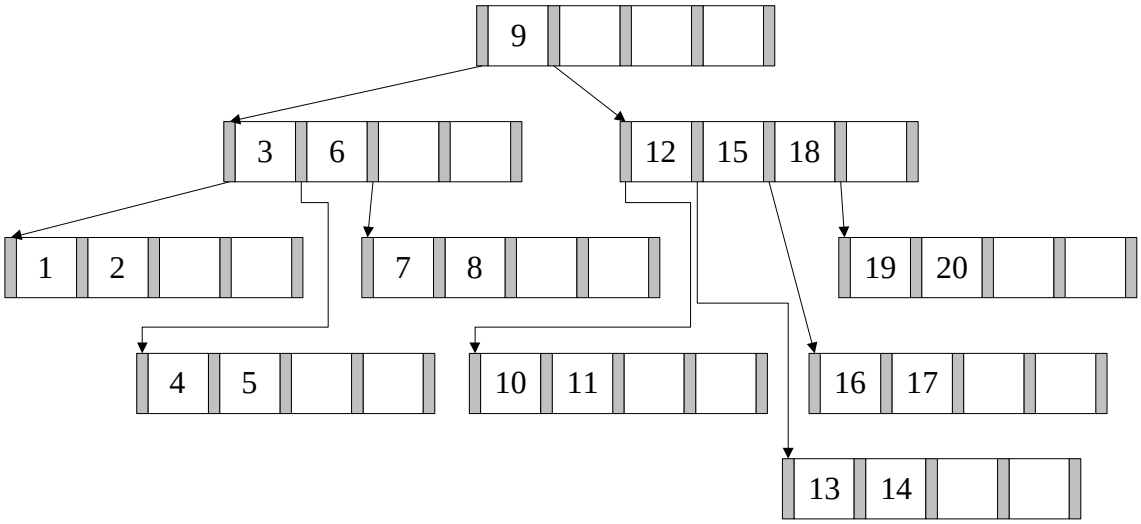
Es werden nun die nächsten Zahlen bis 16 analog eingefügt.



Bei 17 kommt es dann wieder zum Überlauf.



Fügt man nun noch die restlichen Zahlen ein, erhält man folgenden B-Baum:



Es fällt auf, dass der B-Baum nahezu minimale Auslastung aufweist. Dies liegt daran, dass eine aufsteigende Zahlenfolge sequentiell in den Baum eingefügt wird. Nach dem Aufspalten einer Seite in zwei Seiten werden dann in die Seite, die die kleineren Datensätze enthält, keine weiteren Werte mehr eingefügt. Allgemein ist das sortierte Einfügen der Schlüssel in einen B-Baum eine sehr schlechte Idee, da dies zu einer sehr geringen Auslastung führt.

Hausaufgabe 4

Bestimmen Sie k für einen B-Baum, der die folgenden Informationen aller Menschen auf der Erde (ca. 10 Milliarden) enthalten soll: Namen, Land, Stadt, PLZ, Straße und Hausnummer (insgesamt ca. 100 Byte). Dabei ist die Steuernummer eindeutig und 64 Bit lang und wird im B-Baum als Suchschlüssel verwendet. Gehen Sie bei der Berechnung davon aus, dass eine Speicherseite 16KiB groß ist und ein Knoten des B-Baums möglichst genau auf diese Seite passen sollte.

Zunächst gibt uns die Information über die zu erwartende Anzahl von Einträgen im B-Baum einen Hinweis auf die größe eines Verweises in den Speicher. Wir betrachten gängige Architekturen und sehen, dass eine Speicherung auf einer Maschine, deren Adressierbarer Speicher lediglich 2^{32} byte groß ist, nicht ohne weiteres möglich ist. Wir gehen im Verlauf der Aufgabe also von einer 64 bit Architektur aus, bei der ein Zeiger eine Größe von 8 byte hat.

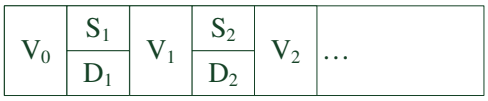


Abbildung 1: Struktur eines B-Baum Knotens

Um nun k zu bestimmen, muss die von k abhängige Größe eines Knotens maximiert werden, so dass dieser gerade noch auf eine Seite der Größe 16KiB passt. Die Struktur eines solchen Knotens ist in Abbildung 1 dargestellt. Zu beachten ist, dass ein komplett gefüllter Knoten genau $2k$ Schlüssel/Daten Paare, jedoch $2k + 1$ Verweise speichern muss.

Die Berechnung ergibt sich wie folgt:

$$
\begin{aligned}
2k * (\text{Schlüsselgröße} + \text{Datengröße}) + (2k+1) * \text{Zeigergröße} &\leq 16KiB \\
2k * (8\text{bytes} + 100\text{bytes}) + (2k+1) * 8\text{bytes} &\leq 16384\text{bytes} \\
2k * 116\text{bytes} + 8\text{bytes} &\leq 16384\text{bytes} \\
2k &\leq 16376\text{bytes}/116\text{bytes} \\
k \leq (16376\text{bytes}/116\text{bytes})/2 &\approx 70,59
\end{aligned}
$$

Da der Knoten „innerhalb“ einer Speicherseite liegen soll und daher nicht überlappen darf, sollte k idealerweise auf 70 gesetzt werden, keinesfalls auf 71.