Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

Datenbanken

B-Baum, B^*-Baum, B^+-Baum

Dipl.-Inf. Robert Hartmann

Fachbereich Informatik (Sankt Augustin)

Juni 2024, KW 26

1. B-Baum (BAYER-Baum)
2. B^{+}–Baum und B^{*}–Baum
3. Darstellung B–Baum, B^{+}–Baum und B^{*}–Baum
4. Algorithmus zum Einfügen in einen B–Baum
5. Algorithmus zum Löschen aus einem B–Baum
6. Beispiel B–Baum, B^{+}–Baum und B^{*}–Baum

B-Baum (Bayer-Baum)

B für **B**ayer, **b**alanced (=ausgeglichen, ausgewogen), **B**oeing, ...

Fakt (Bayer-Baum)

Ein B-Baum (1972 von Rudolf Bayer *und* Ed McCreight, *im Boeing Scientific Research Lab) ist ein höhenbalancierter Baum, bei dem alle Blätter auf gleichem Niveau liegen.*

Fakt

Der B-Baum erwies sich als ideale Datenstruktur zur Verwaltung von Indizes für das relationale Datenbankmodell, das 1970 von Edgar F. Codd *entwickelt wurde. Diese Kombination führte zur Entwicklung des ersten SQL-Datenbanksystems* `System R` *bei IBM.*

Fakt

Jeder innere Knoten eines B-Baum enthält Suchschlüssel und zugehörige Datenwerte, sowie Verweise auf nachfolgende Knoten.

Ursprüngliche Definition (B-Baum)

Ein B-Baum heißt von der **Klasse** m, wenn gilt

- Jeder Knoten enthält höchstens $2m$ Schlüssel.
- Jeder Knoten mit Ausnahme des Wurzelknoten enthält mindestens m Schlüssel.
- Ein Knoten mit k Schlüsseln hat genau $k+1$ Nachfolger oder keinen Nachfolger.
- Alle Knoten, die keine Nachfolger haben, sind Blätter auf gleichem Niveau.
- (Suchbaumeigenschaft): Sind $s_1, s_2, \ldots, s_k$ mit $m \leq k \leq 2m$ die Schlüssel eines Knotens x, dann sind alle Schlüssel des ersten Nachfolgers von x kleiner als s_1, alle Schlüssel des $(k+1)$-tem Nachfolgers größer als s_k und alle Schlüssel des i-ten Nachfolgers, $1 < i < k+1$, größer als s_{i-1} und kleiner als s_i.

Suchen im B-Baum der Klasse m mit n Knoten liegt in der Komplexitätsklasse $\mathcal{O}(m \cdot log_{2m+1}(n))$.

B-Baum Erweiterung

Definition (B-Baum)

Ein B-Baum heißt von der **Ordnung** n, wenn gilt

- Alle Blätter besitzen gleiches Niveau.
- Jeder Knoten mit Ausnahme der Wurzel besitzt mindestens $\left[\frac{n}{2}\right]$ Nachfolger und höchstens n Nachfolger.
- Die Wurzel besitzt mindestens 2 und höchstens n Nachfolger.
- Jeder Knoten mit k Nachfolgern enthält $k - 1$ Schlüssel.
- (Suchbaumeigenschaft) wie zuvor.

Dadurch sind mehr Bäume erfasst; so lassen sich die B-Bäume der Ordnung 3, die so genannten 2-3–Bäume (Hopcroft, 1970) definieren, was mit der ursprünglichen Definition nicht möglich ist, da hier – beim 2-3–Baum – die maximale Anzahl von Schlüsseln in einem Knoten ungerade ist.
Der B-Baum der Ordnung 4, auch bekannt als 2-3-4–Baum, lässt sich sehr leicht durch binäre Bäume, die *Rot-Schwarz-Bäume*, implementieren.

B^{+}–Baum und B^{*}–Baum

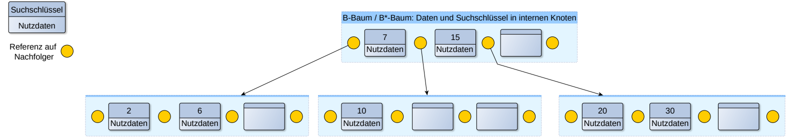
Unterschied zu B-Bäumen

- B^{*}-Baum (Donald Ervin Knuth,1973): innere Knoten zu $\frac{2}{3} = 66\%$ gefüllt bei B-Baum mindestens $\frac{1}{2}$. Im Reiser4 Dateisystem werden Blätter des B^{*}-Baums mit linearer Liste verbunden.
- B^{+}-Baum : Datenelemente nur in Blattknoten. Innere Knoten enthalten Suchschlüssel und Verweise auf Nachfolger. Im NTFS Dateisystem werden die Blätter des B^{+}-Baums mit linearer Liste verbunden.

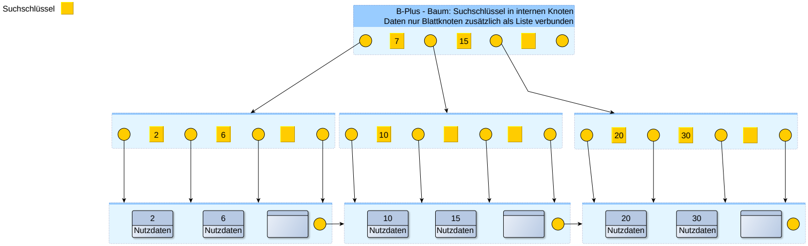
Keine allgemein gültige Benamung.

Der B^{+}–Baum wird aus historischen Gründen manchmal auch als B^{*}–Baum bezeichnet.

B–Baum, B*–Baum



B$^+$–Baum



Algorithmus zum Einfügen in einen B–Baum I

1. Füge anfangs in ein leeres Feld der Wurzel ein
2. Die ersten n Werte (d.h. Suchschlüssel mit Daten) werden sortiert als Separatoren in die Wurzel eingetragen
3. Der nächste Wert, das ist der $(n+1)$-te, passt nicht mehr in die Wurzel und erzeugt einen **Überlauf** (overflow).
4. Die Wurzel wird **geteilt** mit der *Split*-Operation:
   - Jeder der beiden Kinder bekommt $\left\lfloor \frac{n}{2} \right\rfloor$ Werte.
   - Die Wurzel bekommt den **Median** der $(n+1)$ Werte als Separator eingetragen.
5. Neue Werte ($=$ Schlüssel, Daten) werden in den Blättern sortiert gespeichert.

Algorithmus zum Einfügen in einen B–Baum II

6. Läuft ein Blatt über, d.h. der $(n+1)$-te Eintrag müsste gemacht werden, wird ein Splitt durchgeführt:
   - Das Blatt wird geteilt $\Rightarrow$ 2 neue Blätter mit je $\left\lfloor \frac{n}{2} \right\rfloor$ Einträgen.
   - Diese werden mit dem Elternknoten anstelle des ursprünglichen Blattes verbunden.
   - Der **Median** der $(n+1)$ Werte wird als Separator in den Elternknoten eingetragen.
7. Ist auch dieser Elternknoten voll, wird das Splitten fortgesetzt, im schlimmsten Fall bis zur Wurzel.

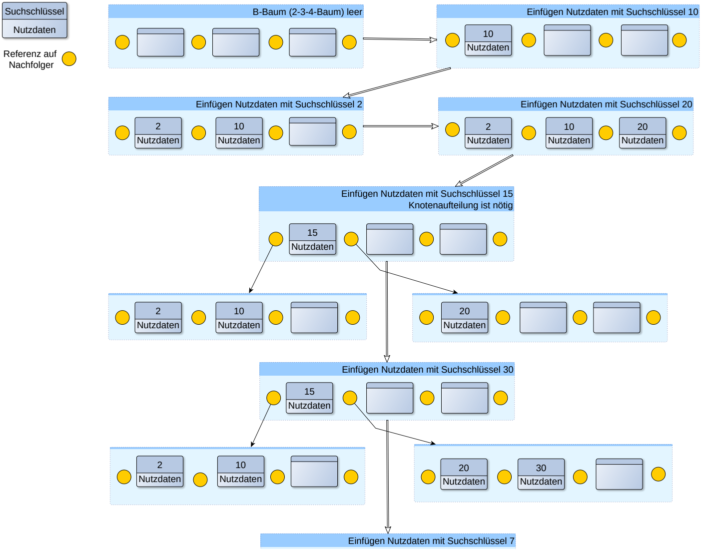
Algorithmus zum Löschen aus einem B–Baum I

1. Sei x das zu löschende Element. Sei k die minimal nötige Knotenbelegung.
2. Suche x. (Graphtraversierung, vgl. *„Programmieren 2“*, *„Algorithmen, Datenstrukturen, Graphentheorie“*)
3. unterscheide 2 Fälle:
    - x ist in einem Blatt:
        - Verbleiben mindestens k Werte nach dem Löschen im Blatt $\Rightarrow$ okay
        - Verbleiben im Blatt $k - 1$ Werte (also zuwenig) und ein *direktes* Nachbarblatt besitzt mehr als k Werte $\Rightarrow$ Rotation (also Füllungsausgleich).
        - Sonst Blätter werden verschmolzen $\Rightarrow$ Merge

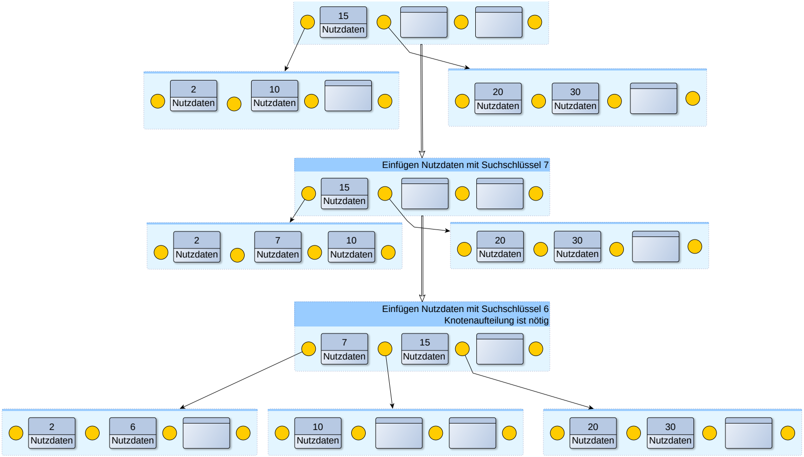
Algorithmus zum Löschen aus einem B–Baum II

- x ist in einem inneren Knoten:
  - Ersetze x durch den nächstgrößeren oder nächstkleineren Wert (bezogen auf den Suchschlüssel) aus dem linken bzw. rechten Nachfolgeknoten.
  - Lösche den gerade gewählten Wert aus dem entsprechenden Knoten.
    $\Rightarrow$ (Rekursion!)

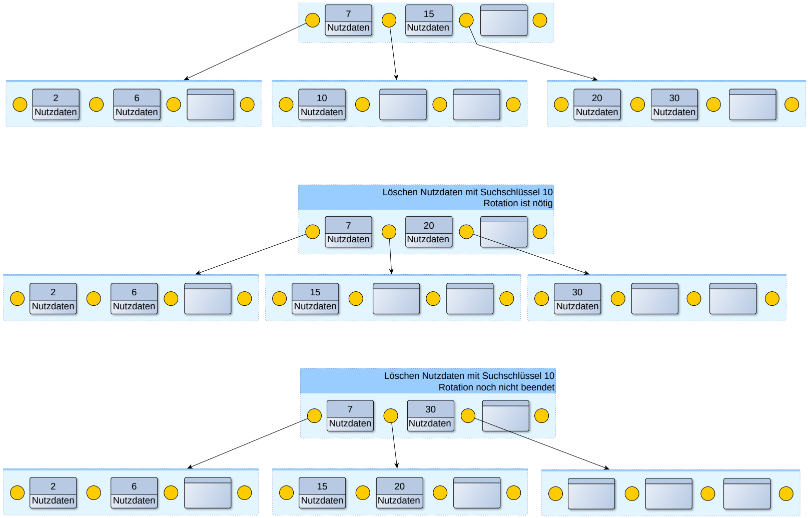
Beispiel B–Baum, B^+–Baum und B^*–Baum (I)



Beispiel B–Baum, B^+–Baum und B^*–Baum (II)



Beispiel B–Baum, B^{+}–Baum und B^{*}–Baum (III)



Beispiel B–Baum, B^+–Baum und B^*–Baum (IV)