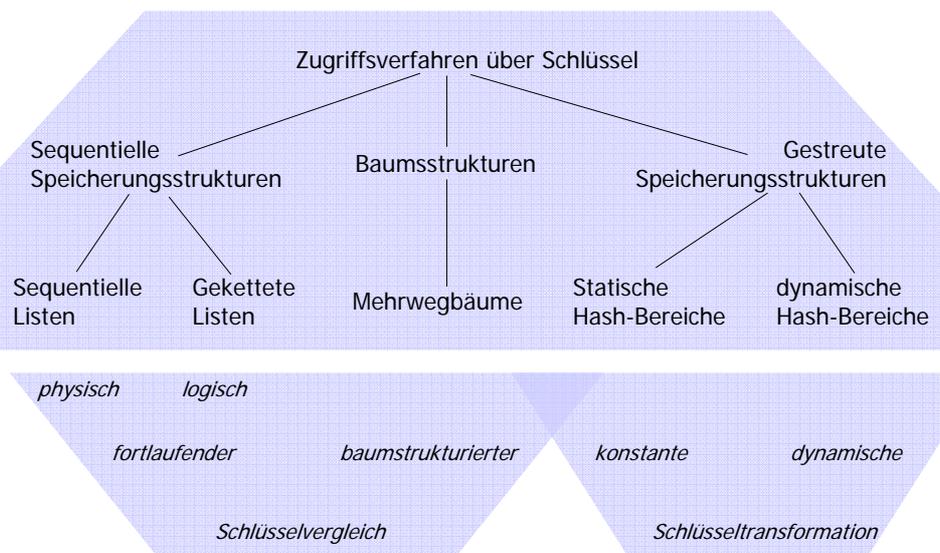




DB-Zugriffsverfahren:

- Übersicht
- B-Bäume
- B*-Bäume

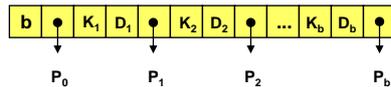
Übersicht



m-Wege-Suchbäume

Definition: Ein m-Wege-Suchbaum oder ein m-ärer Suchbaum B ist ein Baum, in dem alle Knoten einen Grad $\leq m$ besitzen. Entweder ist B leer oder er hat folgende Eigenschaften:

1. Jeder Knoten des Baums hat folgende Struktur:



Die P_i , $0 \leq i \leq b$, sind Zeiger auf die Unterbäume des Knotens und die K_i und D_i , $1 \leq i \leq b$, sind Schlüsselwerte und Daten.

2. Die Schlüsselwerte im Knoten sind aufsteigend geordnet:
 $K_i \leq K_{i+1}$, $1 \leq i < b$.
3. Alle Schlüsselwerte im Unterbaum von P_i sind kleiner als der Schlüsselwert K_{i+1} , $0 \leq i < b$.
4. Alle Schlüsselwerte im Unterbaum von P_i sind größer als der Schlüsselwert K_i , $1 \leq i \leq b$.
5. Die Unterbäume von P_i , $0 \leq i \leq b$, sind auch m-Wege-Suchbäume.

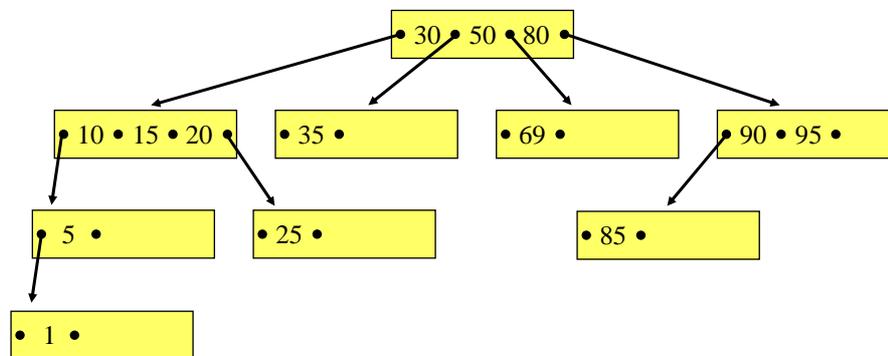
N. Ritter, HMS, 13.09.2007

3

Aufbaubeispiel m-Wege-Suchbäume

$m = 4$

Einfügereihenfolge: 30, 50, 80, 10, 15, 69, 90, 20, 35, 5, 95, 1, 25, 85



N. Ritter, HMS, 13.09.2007

4

Beobachtung zu m-Wege-Suchbäume

Beobachtung:

Die Schlüssel in den inneren Knoten besitzen zwei Funktionen. Sie identifizieren Daten(sätze) und sie dienen als Wegweiser in der Baumstruktur.

Der m-Wege-Suchbaum ist im allgemeinen nicht ausgeglichen. Es ist für Aktualisierungsoperationen kein Balancierungsmechanismus vorgesehen.
(schlechte Platzausnutzung, Entartung)

Wichtige Eigenschaften von m-Wege-Suchbäumen

- Die D_i können Daten oder Zeiger auf die Daten repräsentieren. Zur Vereinfachung werden die D_i (in den Illustrationen) weggelassen.
- $S(P_i)$ sei die Seite, auf die P_i zeigt, und $K(P_i)$ sei die Menge aller Schlüssel, die im Unterbaum mit Wurzel $S(P_i)$ gespeichert werden können.

Dann gelten folgende Ungleichungen:

$$i. \quad x \in K(P_0): (x < K_1)$$

$$ii. \quad x \in K(P_i): (K_i < x < K_{i+1}), \text{ für } i = 1, 2, \dots, b-1$$

$$iii. \quad x \in K(P_b): (K_b < x)$$

Sie gelten für alle Mehrwegbäume!

Kostenanalyse für m-Wege-Suchbäume

- Die Anzahl der Knoten N in einem vollständigen Baum der Höhe h ist

$$N = \sum_{i=0}^{h-1} m^i = \frac{m^h - 1}{m - 1}$$

- bei voller Belegung ergibt sich als Anzahl der Einträge

$$n = N \cdot (m - 1)$$

- im ungünstigsten Fall ist der Baum völlig entartet

$$n = N = h$$

- Schranken für die Höhe eines m-Wege-Suchbaums

$$\log_m(n+1) \leq h \leq n$$

Wartungsaufwand: $O(n)$

N. Ritter, HMS, 13.09.2007

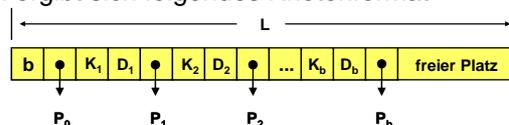
7

B-Bäume

Definition: Seien k, h ganze Zahlen, $h \geq 0, k > 0$. Ein B-Baum B der Klasse $\tau(k, h)$ ist entweder ein leerer Baum oder ein geordneter Suchbaum mit folgenden Eigenschaften:

- Jeder Pfad von der Wurzel zu einem Blatt hat die gleiche Länge $h-1$.
- Jeder Knoten außer der Wurzel und den Blättern hat mindestens $k+1$ Söhne. Die Wurzel ist ein Blatt oder hat mindestens 2 Söhne.
- Jeder Knoten hat höchstens $2k+1$ Söhne.
- Jedes Blatt mit der Ausnahme der Wurzel als Blatt hat mindestens k und höchstens $2k$ Einträge.

Für einen B-Baum ergibt sich folgendes Knotenformat



N. Ritter, HMS, 13.09.2007

8

B-Bäume

Einträge

- Die Einträge für Schlüssel, Daten und Zeiger haben die festen Längen l_b , l_K , l_D und l_p .
- Die Knoten- oder Seitengröße sei L .
- Maximale Anzahl von Einträgen pro Knoten $b_{\max} = \left\lfloor \frac{L - l_b - l_p}{l_K + l_D + l_p} \right\rfloor = 2k$

Reformulierung der Def.:

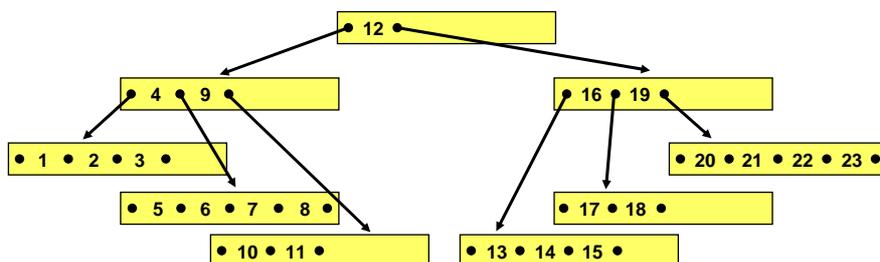
- (4) und (3) ‚Eine Seite darf höchstens voll sein.‘
- (4) und (2) Jede Seite (außer der Wurzel) muss mindestens halb voll sein. Die Wurzel enthält mindestens einen Schlüssel.
- (1) Der Baum ist, was die Knotenstruktur angeht, vollständig ausgeglichen.

N. Ritter, HMS, 13.09.2007

9

B-Bäume

Beispiel: B-Baum der Klasse $\tau(2,3)$



- In jedem Knoten stehen die Schlüssel in aufsteigender Ordnung mit $K_1 < K_2 < \dots < K_b$.
- Jeder Schlüssel hat eine Doppelrolle als Identifikator eines Datensatzes und als Wegweiser im Baum.
- Die Klassen $\tau(k,h)$ sind nicht alle disjunkt. Beispielsweise ist ein maximaler Baum aus $\tau(2,3)$ ebenso in $\tau(3,3)$ und $\tau(4,3)$.

N. Ritter, HMS, 13.09.2007

10

B-Bäume

Bei einem B-Baum der Klasse $\tau(k,h)$ mit n Schlüsseln gilt für seine Höhe: $\log_{2k+1}(n+1) \leq h \leq \log_{k+1}((n+1)/2) + 1$ für $n \geq 1$
 $h = 0$ für $n = 0$

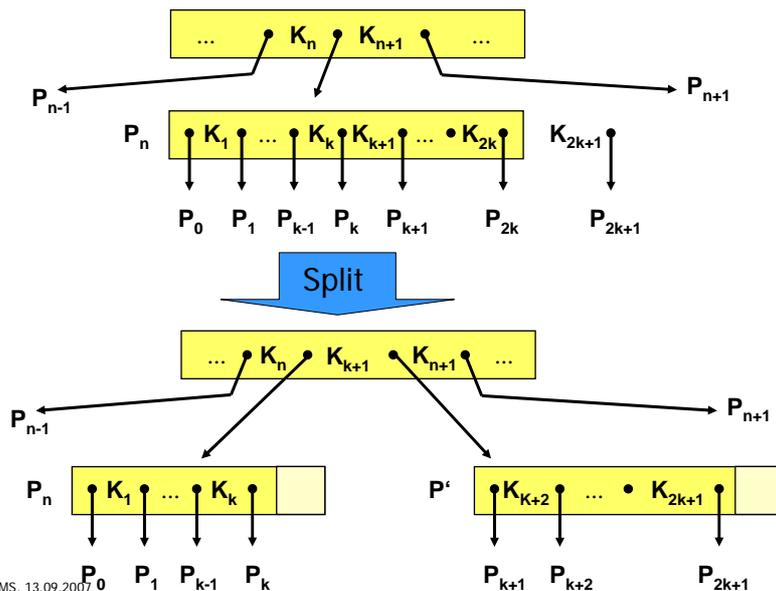
Balancierte Struktur

- unabhängig von Schlüsselmenge
- unabhängig von ihrer Einfügereihenfolge

Einfügealgorithmus (ggf. rekursiv)

- suche Einfügeposition;
- wenn Platz vorhanden ist, speichere Element, sonst schaffe Platz durch Split-Vorgang und füge ein.

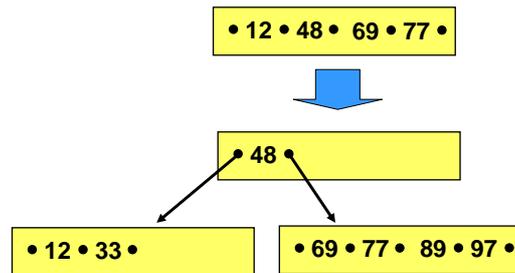
B-Bäume



B-Bäume

Beispiel: B-Baum der Klasse $\tau(2, h)$, Einfügen

Einfügereihenfolge: 77, 12, 48, 69, 33, 89, 97, 91, 37, 45, 83, 2, 5, 57, 90, 95, 99, 50



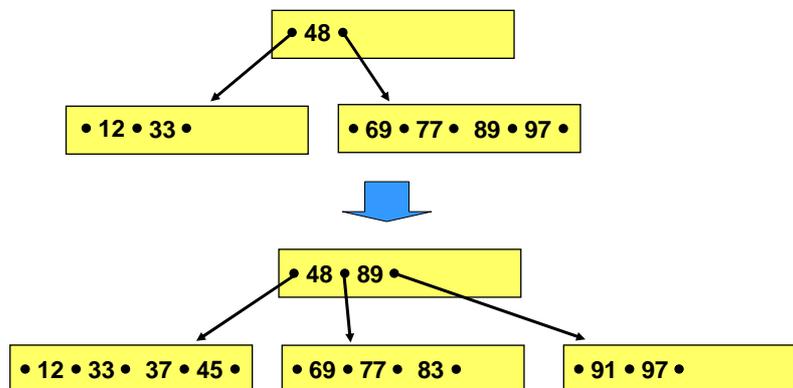
N. Ritter, HMS, 13.09.2007

13

B-Bäume

Beispiel: B-Baum der Klasse $\tau(2, h)$, Einfügen

Einfügereihenfolge: 77, 12, 48, 69, 33, 89, 97, 91, 37, 45, 83, 2, 5, 57, 90, 95, 99, 50



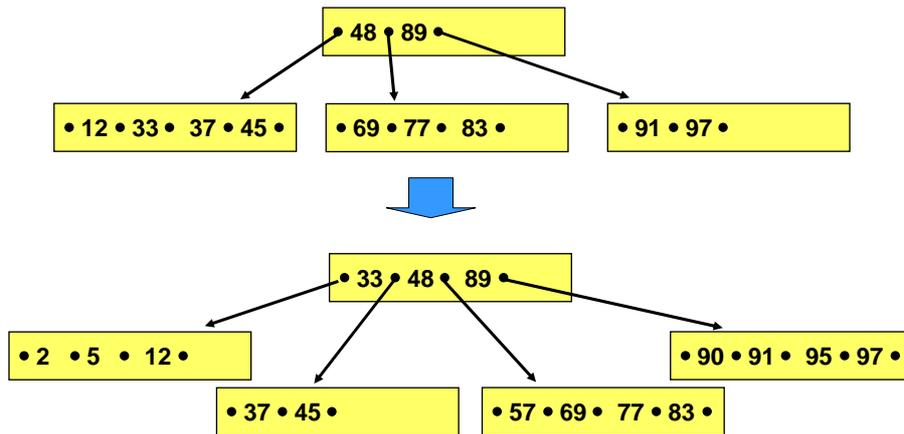
N. Ritter, HMS, 13.09.2007

14

B-Bäume

Beispiel: B-Baum der Klasse $\tau(2, h)$, Einfügen

Einfügereihenfolge: 77, 12, 48, 69, 33, 89, 97, 91, 37, 45, 83, 2, 5, 57, 90, 95, 99, 50



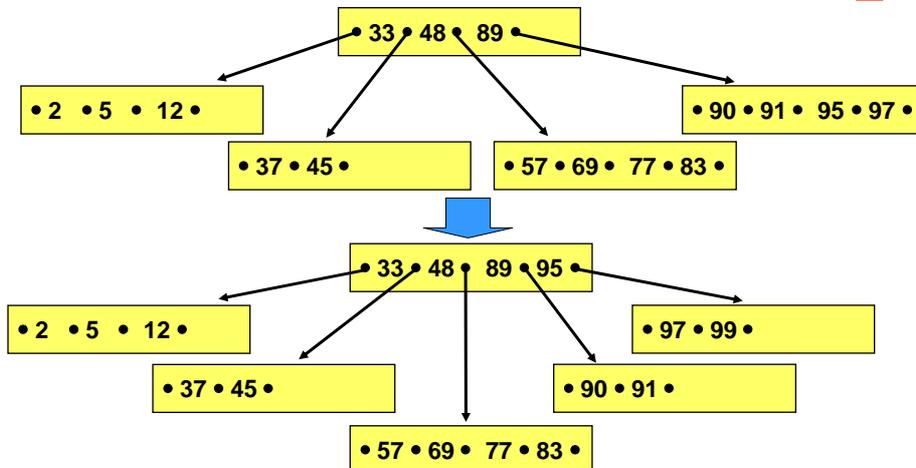
N. Ritter, HMS, 13.09.2007

15

B-Bäume

Beispiel: B-Baum der Klasse $\tau(2, h)$, Einfügen

Einfügereihenfolge: 77, 12, 48, 69, 33, 89, 97, 91, 37, 45, 83, 2, 5, 57, 90, 95, 99, 50



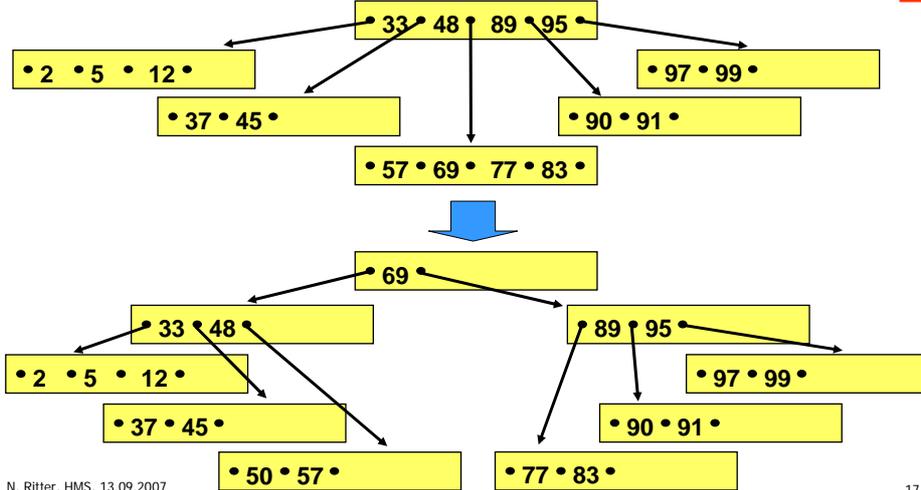
N. Ritter, HMS, 13.09.2007

16

B-Bäume

Beispiel: B-Baum der Klasse $\tau(2, h)$, Einfügen

Einfügereihenfolge: 77, 12, 48, 69, 33, 89, 97, 91, 37, 45, 83, 2, 5, 57, 90, 95, 99, 50



N. Ritter, HMS, 13.09.2007

17

B-Bäume

Kostenanalyse für Einfügen und Suchen

Anzahl der zu holenden Seiten : f (fetch)

Anzahl der zu schreibenden Seiten : w (write)

Direkte Suche

$f_{\min} = 1$: der Schlüssel befindet sich in der Wurzel

$f_{\max} = h$: der Schlüssel ist in einem Blatt

$f_{\text{avg}} = ?$: bei maximaler/minimaler Belegung des B-Baumes

N. Ritter, HMS, 13.09.2007

18

B-Bäume

Kostenanalyse Direkte Suche

gesamte Zugriffskosten bei maximaler Belegung

$$Z_{\max} = 2k \cdot \sum_{i=0}^{h-1} (i+1) \cdot (2k+1)^i = h \cdot m^h - 1 - \frac{m^h - m}{m-1} \text{ mit } m = 2k+1$$

mittlere Zugriffskosten bei maximaler Belegung

$$f_{\text{avg}}(\max) = \frac{Z_{\max}}{n_{\max}} = h - \frac{1}{2k} + \frac{h}{(2k+1)^h - 1}$$

gesamte Zugriffskosten bei minimaler Belegung

$$Z_{\min} = 1 + 2k \cdot \sum_{i=0}^{h-2} (i+2) \cdot (k+1)^i = 1 + 2h \cdot m^{h-1} - 4 - 2 \frac{m^{h-1} - m}{m-1} \text{ mit } m = k+1$$

mittlere Zugriffskosten bei minimaler Belegung

$$f_{\text{avg}}(\min) = \frac{Z_{\min}}{n_{\min}} = h - \frac{1}{k} + \frac{h-1}{2(k+1)^{h-1} - 1} + \frac{1}{k(2(k+1)^{h-1} - 1)}$$

B-Bäume

Kostenanalyse Direkte Suche (Forts.)

Abschätzung möglich, da k meistens im Bereich ,100 bis 200':

$$f_{\text{avg}} = h \quad \text{für } h=1$$

$$h - \frac{1}{k} \leq f_{\text{avg}} \leq h - \frac{1}{2k} \quad \text{für } h>1$$

Beim B-Baum sind die maximalen Zugriffskosten eine gute Abschätzung der mittleren Zugriffskosten:

$$\text{bei } h=3 \text{ und } k=100 \text{ ergibt sich } 2.99 \leq f_{\text{avg}} \leq 2.995$$

B-Bäume

Kostenanalyse (Forts.)

Sequentielle Suche

Durchlauf in symmetrischer Ordnung: $f_{\text{seq}} = N$;
Pufferung der Zwischenknoten im HSP wichtig!

Einfügen

günstigster Fall (kein Split): $f_{\text{min}} = h$; $w_{\text{min}} = 1$;

ungünstigster Fall: $f_{\text{max}} = h$; $w_{\text{max}} = 2h + 1$;

durchschnittlicher Fall: $f_{\text{avg}} = h$; $w_{\text{avg}} < 1 + \frac{2}{k}$;

(wenn $k=100$ unterstellt wird, kostet eine Einfügung im Mittel
 $w_{\text{avg}} < 1 + 2/100$ Schreibvorgänge, d.h., es entsteht eine Belastung
von 2% für den Split)

B-Bäume

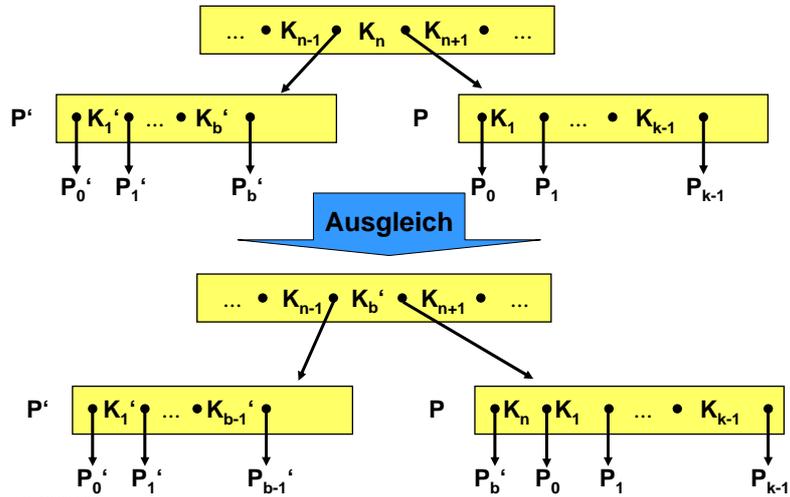
Löschen

Die B-Baum-Eigenschaft muss wiederhergestellt werden, wenn die Anzahl der Elemente in einem Knoten kleiner als k wird.

Durch **Ausgleich** mit Elementen aus einer Nachbarseite oder durch **Mischen** (Konkatenation) mit einer Nachbarseite wird dieses Problem gelöst. Beim Ausgleichsvorgang sind in der Seite P $k-1$ Elemente und in P' mehr als k Elemente.

B-Bäume

Löschmaßnahme: Ausgleich durch Verschieben von Schlüsseln

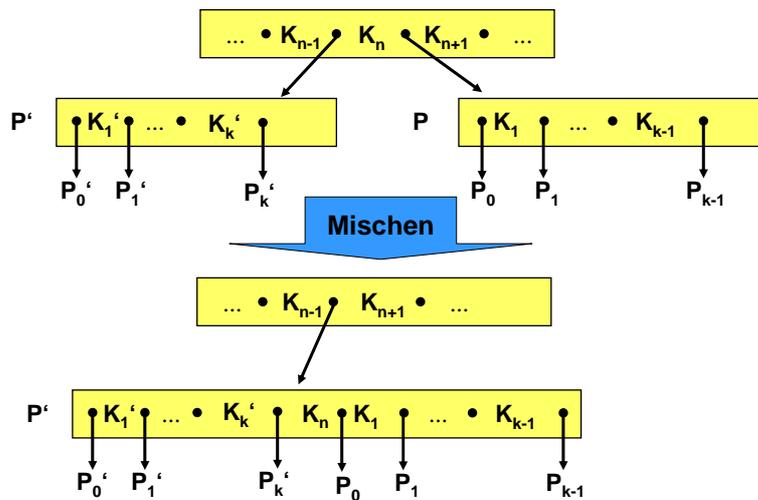


N. Ritter, HMS, 13.09.2007

23

B-Bäume

Löschmaßnahme: Mischen von Seiten



N. Ritter, HMS, 13.09.2007

24

B-Bäume

Löschalgorithmus

1. Löschen in Blattseite

Suche x in Seite P ;

Entferne x in P und wenn

$b \geq k$ in P : tue nichts,

$b = k-1$ in P und $b > k$ in P' : gleiche Unterlauf über P' aus,

$b = k-1$ in P und $b = k$ in P' : mische P und P' .

2. Löschen in innerer Seite

Suche x ;

Ersetze $x = K_i$ durch kleinsten Schlüssel y in $B(P_i)$ oder größten Schlüssel y in $B(P_{i-1})$ (nächst größerer oder nächst kleinerer Schlüssel im Baum);

Entferne y im Blatt P ;

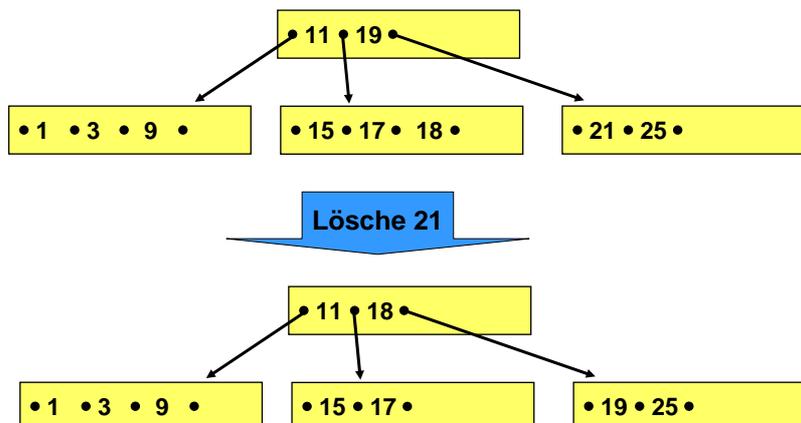
Behandle P wie unter 1.

N. Ritter, HMS, 13.09.2007

25

B-Bäume

Beispiel: Löschen

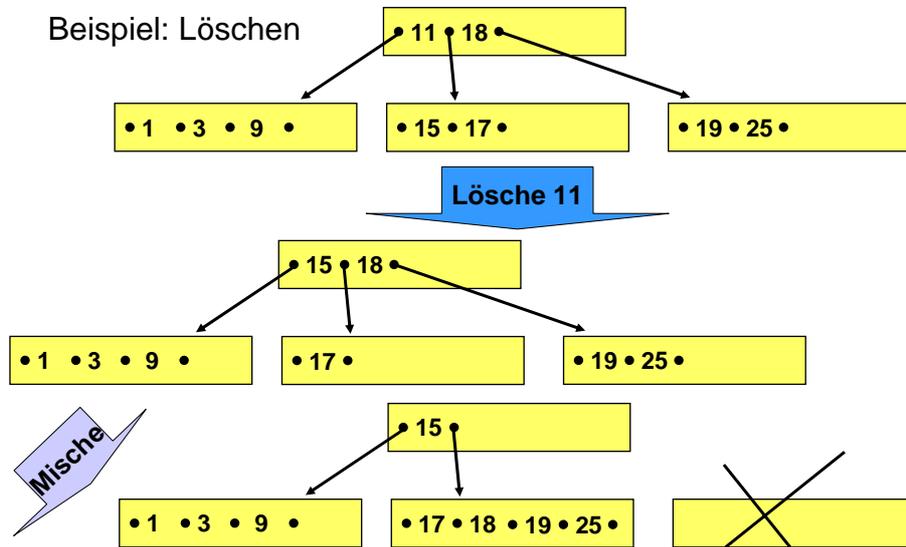


N. Ritter, HMS, 13.09.2007

26

B-Bäume

Beispiel: Löschen



N. Ritter, HMS, 13.09.2007

27

B-Bäume

Kostenanalyse Löschen

günstigster Fall: $f_{\min} = h$; $w_{\min} = 1$;

ungünstigster Fall (pathologisch): $f_{\max} = 2h - 1$; $w_{\max} = h + 1$;

obere Schranke für durchschnittliche Löschkosten

(drei Anteile: 1. Löschen, 2. Ausgleich, 3. anteilige Mischkosten):

$$f_{\text{avg}} \leq f_1 + f_2 + f_3 < h + 1 + \frac{1}{k}$$

$$w_{\text{avg}} \leq w_1 + w_2 + w_3 < 2 + 2 + \frac{1}{k} = 4 + \frac{1}{k}$$

N. Ritter, HMS, 13.09.2007

28

B*-Bäume

Definition: Seien k , k^* und h^* ganze Zahlen, $h^* \geq 0$, $k, k^* > 0$.

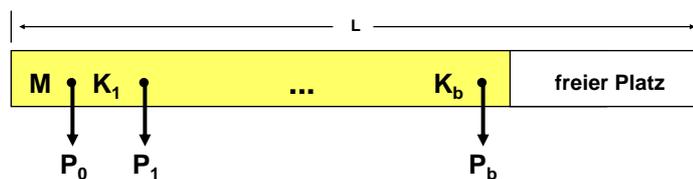
Ein B*-Baum B der Klasse $t(k, k^*, h^*)$ ist entweder ein leerer Baum oder ein geordneter Suchbaum, für den gilt:

1. Jeder Pfad von der Wurzel zu einem Blatt besitzt die gleiche Länge h^*-1 .
2. Jeder Knoten außer der Wurzel und den Blättern hat mindestens $k+1$ Söhne, die Wurzel mindestens 2 Söhne, außer wenn sie ein Blatt ist.
3. Jeder innere Knoten hat höchstens $2k+1$ Söhne.
4. Jeder Blattknoten mit Ausnahme der Wurzel als Blatt hat mindestens k^* und höchstens $2k^*$ Einträge.

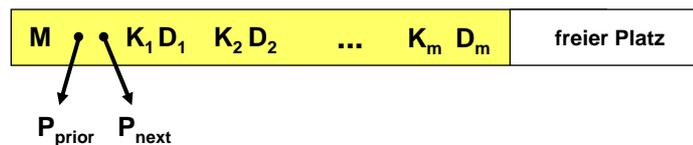
B*-Bäume

2 Knotenformate:

innerer Knoten



Blattknoten



M enthalte eine Kennung des Seitentyps sowie die Zahl der aktuellen Einträge

B*-Bäume

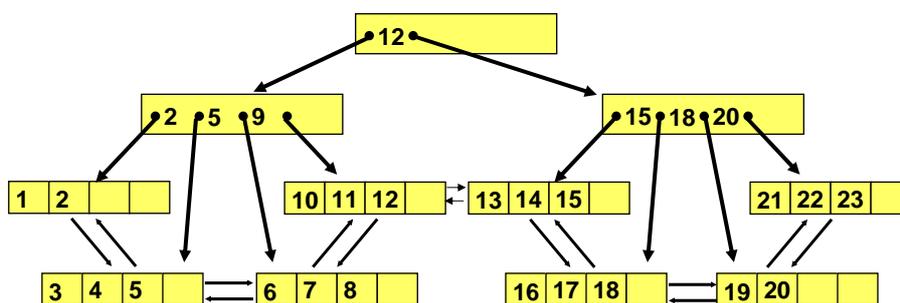
Es gilt:

$$L = l_M + l_P + 2 \cdot k(l_K + l_D); \quad k = \left\lfloor \frac{L - l_M - l_P}{2 \cdot (l_K + l_D)} \right\rfloor$$

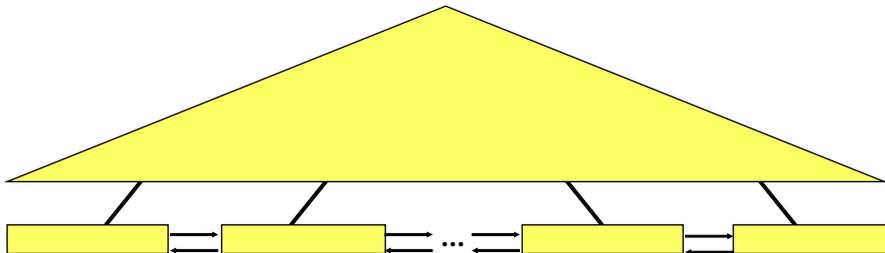
$$L = l_M + 2 \cdot l_P + 2 \cdot k^*(l_K + l_D); \quad k^* = \left\lfloor \frac{L - l_M - 2l_P}{2 \cdot (l_K + l_D)} \right\rfloor$$

B*-Bäume

Beispiel: B*-Baum der Klasse $\tau(3, 2, 3)$



B*-Bäume



Erklärungsmodell für den B*-Baum

Der B*-Baum lässt sich auffassen als eine gekettete sequentielle Datei von Blättern, die einen Indexteil besitzt, der selbst ein B-Baum ist. Im Indexteil werden insbes. beim Split-Vorgang die Operationen des B-Baums eingesetzt.

N. Ritter, HMS, 13.09.2007

33

B*-Bäume

Grundoperationen beim B*-Baum

Direkte Suche:

Da alle Schlüssel in den Blättern sind, kostet jede direkte Suche h^* Zugriffe. h^* ist jedoch im Mittel kleiner als h in B-Bäumen. Da f_{avg} beim B-Baum in guter Näherung mit h abgeschätzt werden kann, erhält man also durch den B*-Baum eine effizientere Unterstützung der direkten Suche.

Sequentielle Suche:

Sie erfolgt nach Aufsuchen des Linksaußen der Struktur unter Ausnutzung der Verkettung der Blattseiten. Es sind zwar ggf. mehr Blätter als beim B-Baum zu verarbeiten, doch da nur h^*-1 innere Knoten aufzusuchen sind, wird die sequentielle Suche ebenfalls effizienter ablaufen.

N. Ritter, HMS, 13.09.2007

34

B*-Bäume

Grundoperationen beim B*-Baum (Forts.)

Einfügen:

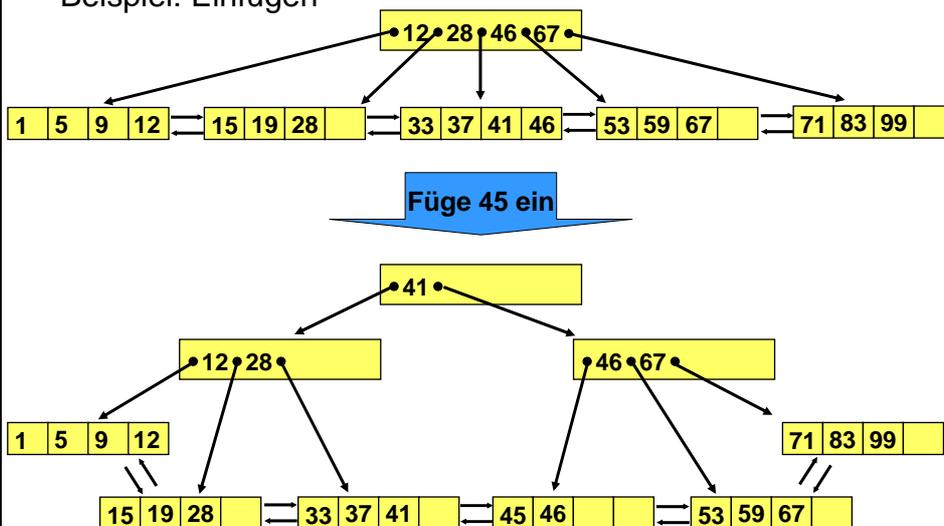
Es ist von der Durchführung und vom Leistungsverhalten her dem Einfügen in einen B-Baum sehr ähnlich. Bei inneren Knoten wird die Spaltung analog zum B-Baum durchgeführt. Beim Split-Vorgang einer Blattseite muss gewährleistet sein, dass jeweils die höchsten Schlüssel einer Seite als Wegweiser in den Vaterknoten kopiert werden. Die Verallgemeinerung des Split-Vorgangs lässt sich analog zum B-Baum einführen.

Löschen:

Datenelemente werden immer von einem Blatt entfernt (keine komplexe Fallunterscheidung wie beim B-Baum). Weiterhin muss beim Löschen eines Schlüssels aus einem Blatt dieser Schlüssel nicht notwendigerweise aus dem Indexteil entfernt werden; er kann seine Funktion als Wegweiser behalten.

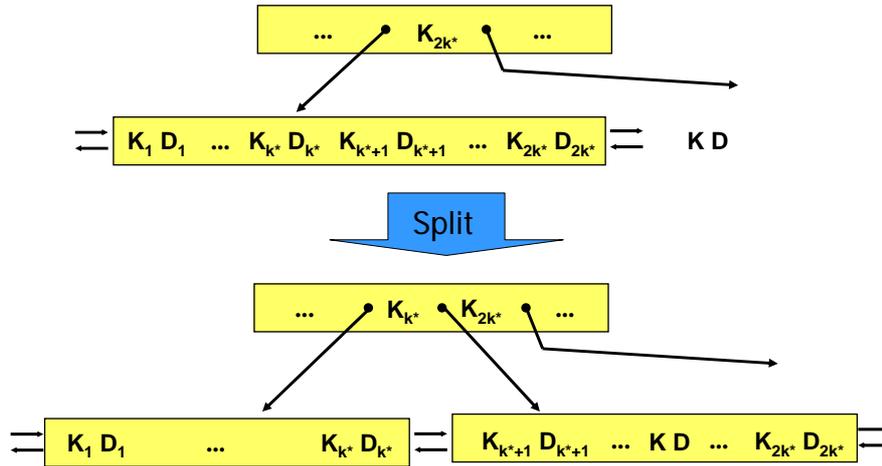
B*-Bäume

Beispiel: Einfügen



B*-Bäume

Schema für den **Split-Vorgang**:

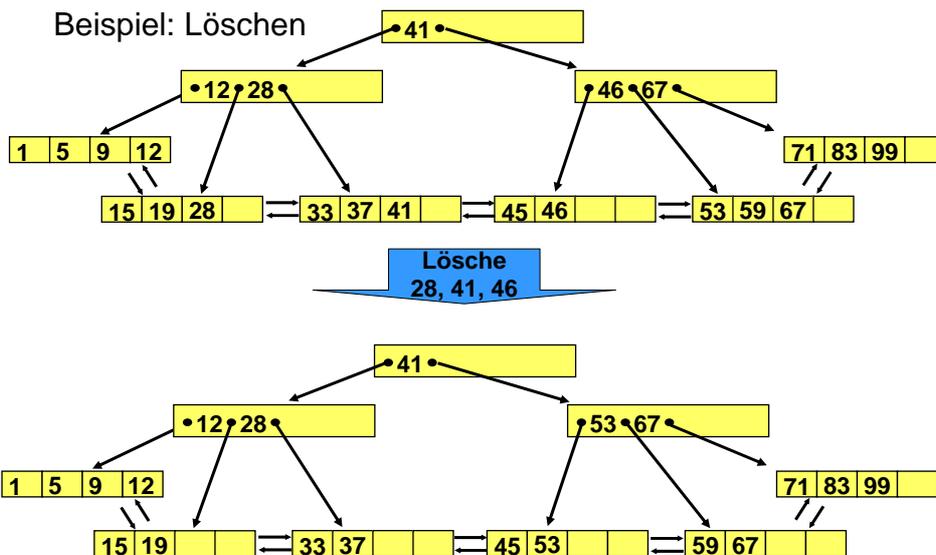


N. Ritter, HMS, 13.09.2007

37

B*-Bäume

Beispiel: Löschen



N. Ritter, HMS, 13.09.2007

38

B*-Bäume

Eigenschaften:

Anzahl der Blattknoten bei minimaler Belegung

$$B_{\min}(k, h^*) = 1 \quad \text{für } h^* = 1$$

$$B_{\min}(k, h^*) = 2(k+1)^{h^*-2} \quad \text{für } h^* \geq 2$$

Minimale Anzahl von Elementen

$$n_{\min}(k, k^*, h^*) = 1 \quad \text{für } h^* = 1$$

$$n_{\min}(k, k^*, h^*) = 2k^* \cdot (k+1)^{h^*-2} \quad \text{für } h^* \geq 2$$

B*-Bäume

Eigenschaften (Forts.):

Anzahl der Blattknoten bei maximaler Belegung

$$B_{\max}(k, h^*) = (2k+1)^{h^*-1} \quad \text{für } h^* \geq 1$$

Maximale Anzahl von Elementen

$$n_{\max}(k, k^*, h^*) = 2k^* (2k+1)^{h^*-1} \quad \text{für } h^* \geq 1$$

Es lässt sich zeigen, dass die Höhe h^* eines B*-Baumes mit n Datenelementen begrenzt ist durch

$$1 + \log_{2k+1}(n/2k^*) \leq h^* \leq 2 + \log_{k+1}(n/2k^*) \quad \text{für } h^* \geq 2$$

B- und B*-Baum: Quantitativer Vergleich

Seitengröße L: 2048 Bytes

Zeiger P_i , Schlüssel K_i , Hilfsinformation: 4 Bytes

Daten D_i eingebettet: 76 Bytes; separat: 4 Bytes (Zeiger)

Datensätze separat (k=85)

h	n_{\min}	n_{\max}
1	1	170
2	171	29.240
3	14.791	5.000.210
4	1.272.112	855.036.083

Datensätze eingebettet (k=12)

h	n_{\min}	n_{\max}
1	1	24
2	25	624
3	337	15.624
4	4.393	390.624

B-Baum

N. Ritter, HMS, 13.09.2007

41

B- und B*-Baum: Quantitativer Vergleich

Datensätze separat
(k=127, $k^*=127$)

h	n_{\min}	n_{\max}
1	1	254
2	254	64.770
3	32.512	16.516.350
4	4.161.536	4.211.669.268

Datensätze eingebettet
(k=12, $k^*=127$)

h	n_{\min}	n_{\max}
1	1	24
2	24	6.120
3	3.072	1.560.600
4	393.216	397.953.001

B*-Baum

Weiterführende Literatur B*-Bäume

Härder, T.: Mehrwegbäume

Härder, T., Rahm, E: Datenbanksysteme – Konzepte und Techniken der Implementierung, Springer, 1999.

N. Ritter, HMS, 13.09.2007

42