

## Aufgabe 1 : Wiederholung von Begriffen

- a) Wie viele Primärindexte und wie viele Sekundärindexte kann eine Relation haben?  
(siehe Erklärung Folie 212)

Ein Primärindex bestimmt die Organisationsform der Datensätze, der Sekundärindex ist ein zusätzlicher Zugriffspfad ohne Abhängigkeit zur Organisationsform. Eine Relation enthält immer einen Primärindex.

Bei der Anzahl der Sekundärindexte gibt es innerhalb einer Relation keine Begrenzungen.

- b) Ein dichter oder dichtbesetzter Index sieht für jeden Datensatz der internen Relation einen Indexeintrag vor, ein dünner oder dünnbesetzter Index dagegen speichert nicht für jeden Zugriffsattributwert einen Eintrag in der Indexdatei.

Welcher Indextyp (dicht oder dünn) eignet sich für einen Primär-, welcher für einen Sekundärindex?

Bitte begründen Sie Ihre Aussagen!

Für einen Primärindex reicht ein dünner Index, für einen Sekundärindex benötigt man einen dichten Index.

Der Primärindex nutzt die Reihenfolge der Relation aus und findet daher jeden einzelnen Datensatz, der Sekundärindex muss mittels des dichten Indexes zum gesuchten Datensatz führen.

## Aufgabe 2 : Pflichtaufgabe: Sequentielle und Indexsequentielle Organisation

Gegeben sei eine sequentielle (sortierte) Datei mit 1 000 000 Datensätzen der festen Länge 100 Bytes, die auf einer Festplatte mit einer Blockgröße von 1 KB als Nicht-Spannsätze gespeichert sind. Ein Suchschlüselfeld besitze die Länge 9 Bytes und ein Zeiger vom Index auf einen Block der Datei die Länge 6 Bytes.

- a) Wie viele Seitenzugriffe werden bei einer linearen Suche auf dieser Datei im Durchschnitt benötigt?

Zunächst errechnen wir das Verhältnis zwischen Blöcken und Datensätzen:

Formel:  $\frac{1024 \text{ Byte (=1 KB Blockgröße)}}{100 \text{ Bytes (feste Länge e. DS)}} = 10$  ganze Datensätze pro Block

Da "Block" mit "Seite" gleichzusetzen ist, nehmen wir nun die Datensätze hinzu und ermitteln den Durchschnitt (also durch 2 teilen):

$$\text{Anzahl der benötigten Blöcke} = \frac{1.000.000}{10} = 100.000$$

$$\text{Durchschnittliche Seitenzugriffe} = \frac{100.000}{2} = 50.000 \text{ Seitenzugriffe bei linearer Suche}$$

- b) Wie viele Seitenzugriffe werden zum Auffinden eines Datensatzes benötigt, wenn für die Datei ein dichtbesetzter bzw. dünnbesetzter Primärindex erzeugt wird?

Ein Suchindex besteht aus einem Suchschlüselfeld und einem Zeiger. Wir rechnen also:  
Suchindex = 9 Byte + 6 Byte = 15 Byte

Ein Block auf der Festplatte hat eine Kapazität von 1 KB, das entspricht 1.024 Bytes.

Jetzt wird errechnet, wie viele Suchindizes in einen Block passen:  $\frac{1.024 \text{ Byte}}{15 \text{ Byte}} = 68$

$\frac{1.000.000}{68}$  ergibt ungefähr 14.705 Indexseiten,

$1 + \log_2 14.705 = 15$  Seitenzugriffe (binäre Suche, dichtbesetzt)

$\frac{100.000}{68}$  ergibt ungefähr 1.471 Indexseiten,  $1 + \log_2 1.471 = 12$  Seitenzugriffe (dünnbes.)

c) Wie viele Seitenzugriffe werden zum Auffinden eines Datensatzes benötigt, wenn für die Datei ein Sekundärindex erzeugt wird?

Die Lösung ist in b): 15

d) Wie viele Seitenzugriffe werden zum Auffinden eines Datensatzes benötigt, wenn für die Datei ein mehrstufiger Index erzeugt wird?

Anmerkung: Bitte beantworten Sie diese Fragen nicht (nur) mit Ergebniszahlen, sondern jeweils mit einem schlüssig und verständlich aufgeschriebenen Lösungsweg.

Dichtbesetzter Index:  $68^n \geq 10^6 \Leftrightarrow n \geq \log_{68} 10^6$  4stufig

Dünnbesetzter Index:  $68^n \geq 10^5 \Leftrightarrow n \geq \log_{68} 10^5$  3stufig

### Aufgabe 3 : Suchen mit mehrstufigen Indexen

Gegeben sei ein dünnbesetzter n - stufiger Primärindex über einer Datei. Skizzieren Sie einen effizienten Algorithmus, der anhand eines Primärschlüssels einen Datensatz in dieser Datei findet.

Binärsuche: Zunächst wird das Element in der Mitte gewählt. Wenn das gesuchte Element größer ist, wird die Mitte der oberen Hälfte ermittelt und je danach, in welcher Hälfte sich das Element befindet, wieder die jeweilige Mitte ermittelt. Das geht so lange, bis das gesuchte Element gefunden ist.

### Aufgabe 5 : Pflichtaufgabe B\* - Bäume

Berechnen Sie die größtmögliche Ordnung k eines B+ -Baums unter den Bedingungen wie in Aufgabe 2. Je ein Knoten des B+ - Baums werde in einem Block gespeichert.

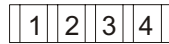
$$2k * 9 + (2k + 1) * 6 \leq 1.024$$
$$k \leq 33$$

#### Aufgabe 4 : Einfügen in einen B-Baum

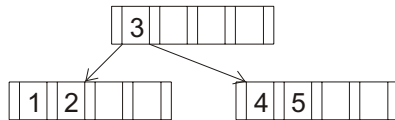
Fügen Sie in einen zunächst leeren B - Baum der Ordnung 2 die Zahlen von 1 bis 20 in aufsteigender Reihenfolge ein.

"Ordnung 2" besagt, daß 2\*2 Felder je Ast und Blatt zu besetzen sind. Die ersten vier Blätter lassen sich daher ganz locker besetzen:

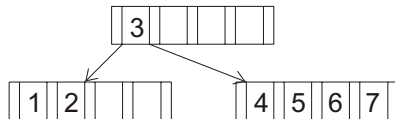
1, 2, 3 und 4 werden eingefügt:



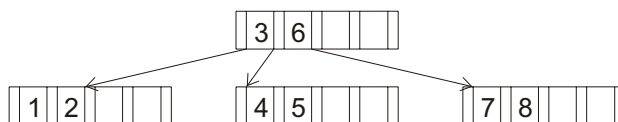
5 kommt hinzu:



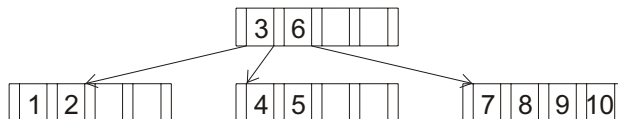
6 und 7 kommen hinzu:



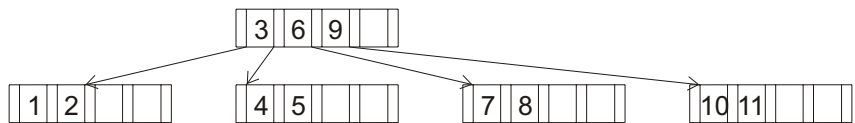
8 kommt hinzu:



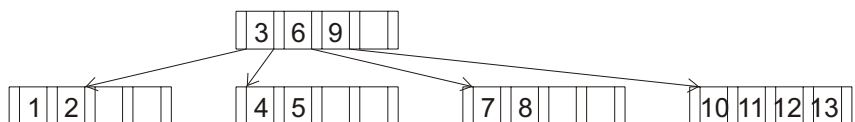
9 und 10 kommen hinzu:



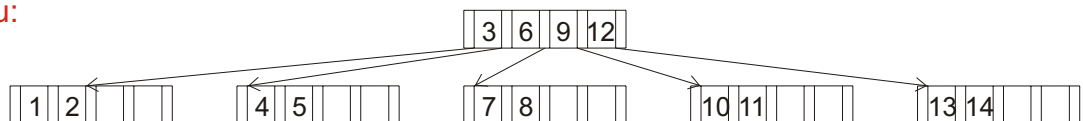
11 kommt hinzu:



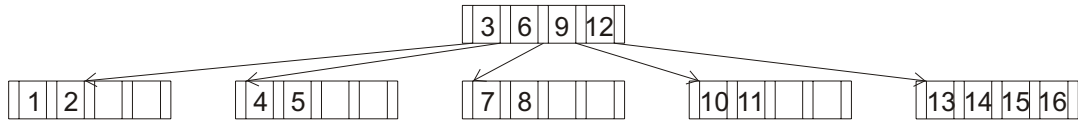
12 und 13 kommen hinzu:



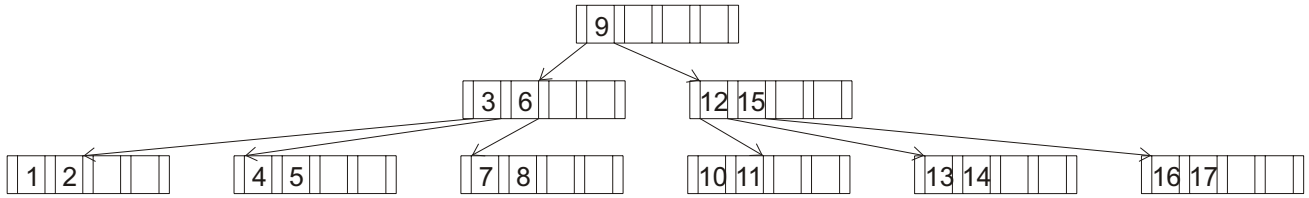
14 kommt hinzu:



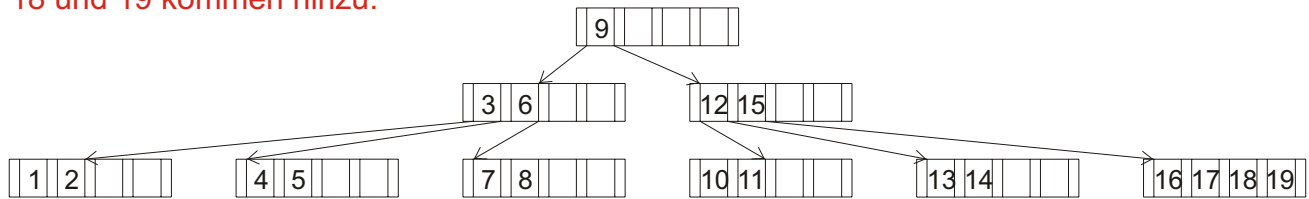
15 und 16 kommen hinzu:



17 kommt hinzu:



18 und 19 kommen hinzu:



20 kommt hinzu:

### Aufgabe 5 : Pflichtaufgabe B+ - Bäume

Berechnen Sie die größtmögliche Ordnung  $k$  eines B+ - Baums unter den Bedingungen wie in Aufgabe 2. Je ein Knoten des B+ - Baums werde in einem Block gespeichert.