FernUniversität in Hagen

Lösungsvorschläge zur Klausur „1664 Implementierungskonzepte für DBS“

17.02.2018

Aufgabe 1

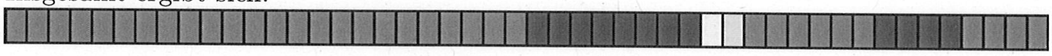
Hier erhalten Sie den Punkt, wenn Sie alle Klausurblätter korrekt und vollständig ausgefüllt haben, also Namen, Matrikelnummer und Adresse korrekt eingetragen und genau diejenigen Aufgaben markiert haben, die Sie auch bearbeitet haben.

Aufgabe 2

(a)

Das `Gehalt` muss an einer durch 8 teilbaren Adresse beginnen. Daher sind zwischen `Name` und dem `Gehalt` 3 zusätzliche, ungenutzte Bytes notwendig. Für die Positionierung von `Tag` und `Monat` sind keine zusätzliches Bytes notwendig, da diese eine Größe von 1 besitzen. Um das `Jahr` auf eine durch 4 teilbare Adresse zu bringen, sind zwei Füllbytes notwendig. Die `AbtNr` ist damit bereits an einer korrekten Position. Damit ist die Gesamtgröße bisher 44 Bytes. Der größte verwendete Typ besitzt 8 Bytes. Damit der nächste Datensatz an einer durch 8 teilbaren Adresse beginnt, sind weitere 4 Bytes zum Auffüllen notwendig, woraus sich eine Gesamtgröße von 48 Bytes ergibt.

Insgesamt ergibt sich:



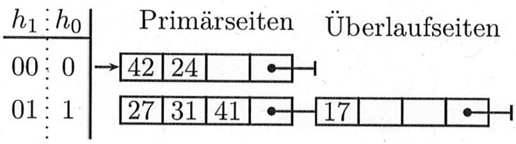
wobei die rot markierten Bereiche nicht verwendet werden.

(b)

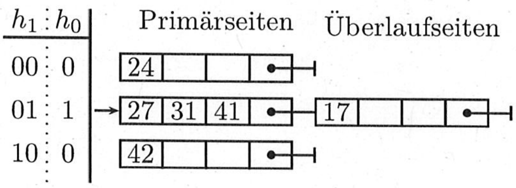
Die tatsächlich genutzte Anzahl an Bytes dieses Datensatzes ergibt sich aus: $21 + 8 + 1 + 1 + 4 + 4 = 39$ Bytes. Damit der Datensatz an einer durch 8 teilbaren Adresse beginnt, sind also 40 Bytes notwendig. Jede Anordnung, die nicht mehr als 40 Bytes benötigt, erfüllt demnach die Forderung nach Speicheroptimalität. Die Anordnung `Gehalt, AbtNr, Jahr, Monat, Tag, Name` erfüllt diese Anforderung. Tatsächlich gibt es hierbei keinen ungenutzten Platz zwischen den einzelnen Attributen.

Aufgabe 3

Überlauf beim Einfügen von 17:



Nach Überlaufbehandlung:



Überlauf beim Einfügen von 11:

$h_1 : h_0$		Primärseiten	Überlaufseiten
00 : 0		24 | 4 | |	
01 : 1	→	27 | 31 | 41	17 | 69 | 45 → 11 | |
10 : 0		42 | |	

Nach Überlaufbehandlung:

$h_1 : h_0$		Primärseiten	Überlaufseiten
000 : 00	→	24 | 4 |	
001 : 01		41 | 17 | 69	45 | |
010 : 10		42 | |	
011 : 11		27 | 31 | 11	

Überlauf beim Einfügen von 19:

$h_1 : h_0$		Primärseiten	Überlaufseiten
000 : 00	→	24 | 4 |	
001 : 01		41 | 17 | 69	45 | 61 |
010 : 10		42 | 2 | 30	
011 : 11		27 | 31 | 11	19 | |

Nach Überlaufbehandlung:

$h_1 : h_0$		Primärseiten	Überlaufseiten
000 : 00		24 | |	
001 : 01	→	41 | 17 | 69	45 | 61 |
010 : 10		42 | 2 | 30	
011 : 11		27 | 31 | 11	19 | |
100 : 00		4 | |	

Überlauf beim Einfügen von 66:

$h_1 : h_0$		Primärseiten	Überlaufseiten
000 : 00		24 | |	
001 : 01	→	41 | 17 | 69	45 | 61 |
010 : 10		42 | 2 | 30	66 | |
011 : 11		27 | 31 | 11	19 | |
100 : 00		4 | |	

Nach Überlaufbehandlung erhalten wir die finale Tabelle:

h_1 : h_0	Primärseiten	Überlaufseiten
000 : 00	24	
001 : 01	41, 17	
010 : 10	42, 2, 30	66
011 : 11	27, 31, 11	19
100 : 00	4	
101 : 01	69, 45, 61	

Aufgabe 4

(a)

Das direkte 2-Wege-Merge-Sort erzeugt Anfangsläufe der Länge 1, die in den verschiedenen Phasen verschmolzen werden. Wir geben jeweils den Inhalt der Dateien an. Die einzelnen Läufe innerhalb einer Datei werden durch das Zeichen „|“ voneinander getrennt dargestellt.

InitialRuns:
f1: 13 | 9 | 11 | 42 | 5 | 21 | 91 | 77 |
f2: 2 | 1 | 32 | 6 | 44 | 66 | 52 | 88 |

Phase 1
g1: 2 13 | 11 32 | 5 44 | 52 91 |
g2: 1 9 | 6 42 | 21 66 | 77 88 |

Phase 2
f1: 1 2 9 13 | 5 21 44 66 |
f2: 6 11 32 42 | 52 77 88 91 |

Phase 3
g1: 1 2 6 9 11 13 32 42 |
g2: 5 21 44 52 66 77 88 91 |

Phase 4
f1: 1 2 5 6 9 11 13 21 32 42 44 52 66 77 88 91 |
f2:

(b)

Die Anfangsläufe beim natürlichen Mischen werden mit Hilfe eines Heaps erstellt. Jeder Lauf (mit Ausnahme des letzten) hat mindestens soviele Elemente, wie der Hauptspeicher Elemente aufnehmen kann. Die Läufe können jedoch auch länger sein.

1	2	3	4	1. Lauf
$\underline{1}$	2	9	13	1
$\underline{2}$	11	9	13	2
$\underline{9}$	11	13	32	9
$\underline{11}$	32	13	42	11
$\underline{13}$	32	42	$\overline{6}$	13
$\underline{32}$	42	$\overline{5}$	$\overline{6}$	32
$\underline{42}$	44	$\overline{5}$	$\overline{6}$	42
$\underline{44}$	$\overline{21}$	$\overline{5}$	$\overline{6}$	44
$\underline{66}$	$\overline{21}$	$\overline{5}$	$\overline{6}$	66
$\underline{91}$	$\overline{21}$	$\overline{5}$	$\overline{6}$	91
$\overline{52}$	$\overline{21}$	$\overline{5}$	$\overline{6}$	

1	2	3	4	2. Lauf
$\underline{5}$	6	52	21	5
$\underline{6}$	21	52	77	6
$\underline{21}$	77	52	88	21
$\underline{52}$	77	88		52
$\underline{77}$	88			77
$\underline{88}$				88

InitialRuns:
f1: 1 2 9 11 13 32 42 44 66 91 |
f2: 5 6 21 52 77 88 |

Phase 1
g1: 1 2 5 6 9 11 13 21 32 42 44 52 66 77 88 91 |
g2:

(c)

Es werden Anfangsläufe der Länge 5 erzeugt, die anschließend verschmolzen werden.

InitialRuns:
f1: 1 2 9 11 13 | 21 52 66 77 91 |
f2: 5 6 32 42 44 | 88 |

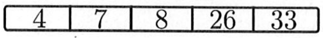
Phase 1
g1: 1 2 5 6 9 11 13 32 42 44 |
g2: 21 52 66 77 88 91 |

Phase 2
f1: 1 2 5 6 9 11 13 21 32 42 44 52 66 77 88 91 |
f2:

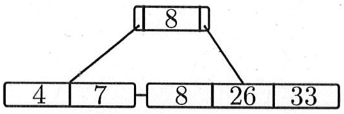
Aufgabe 5

(a)

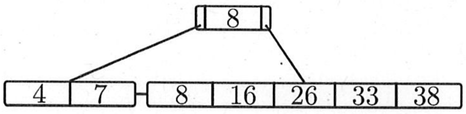
Nach dem Einfügen der 8 ist ein Split erforderlich:



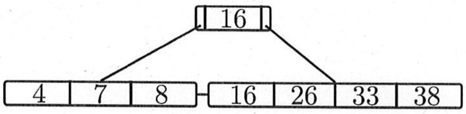
Ergebnis:



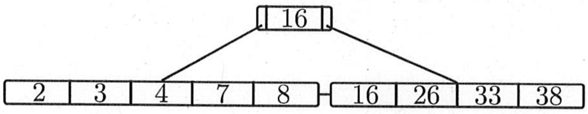
Nach dem Einfügen der 16 ist ein Redistribute erforderlich:



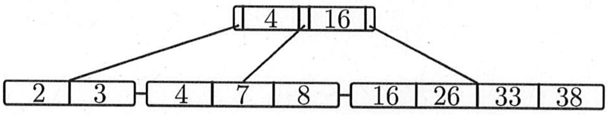
Ergebnis:



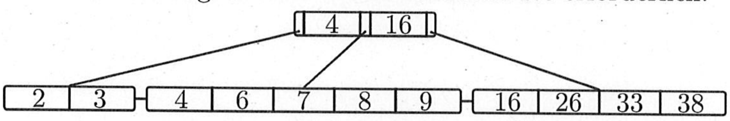
Nach dem Einfügen der 3 ist ein Split erforderlich:



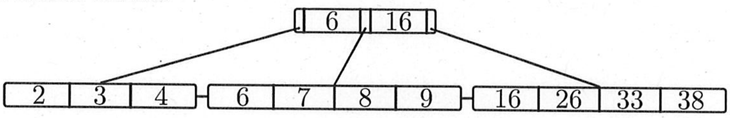
Ergebnis:



Nach dem Einfügen der 6 ist ein Redistribute erforderlich:

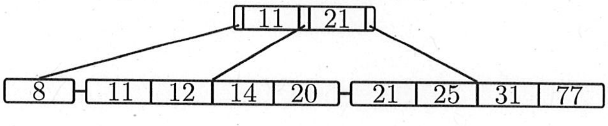


Finaler Baum:

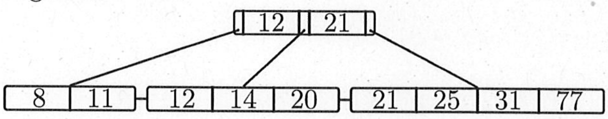


(b)

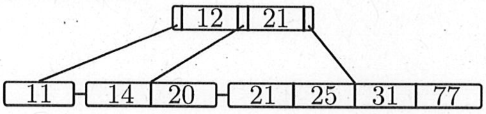
Das Löschen der 3 zieht keine Unterlaufbehandlung nach sich. Nach dem Löschen der 4 ist eine Balance-Operation erforderlich:



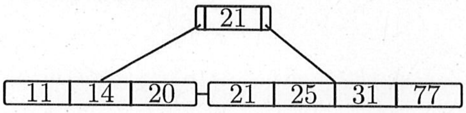
Ergebnis:



Das Löschen der 12 erfolgt ohne Unterlaufbehandlung. Nach dem Löschen der 8 ist ein Merge erforderlich.



Finaler Baum:



Aufgabe 6

Im Folgenden seien $S = Spieler$, $V = Verein$ und $T = Transfer$.

(a)

In Relationenalgebra lautet die Anfrage wie folgt:

$$\pi_{S.Name, V.Name, T.Abloese} \; \sigma_{S.Snr=T.Snr \,\wedge\, T.Kaeufernr=V.Vnr \,\wedge\, T.Datum>=01.01.2018 \,\wedge\, \neg(V.Land=\text{„Spanien“}) \,\wedge\, S.Vertragsende=2023 \,\wedge\, (S.Position=\text{„Stuermer“} \,\vee\, S.Position=\text{„Torwart“})}(S \times V \times T)$$

(b)

In disjunktiver Normalform hat die Anfrage die folgende Gestalt:

$$\pi_{S.Name, V.Name, T.Abloese} \; \sigma_{S.Snr=T.Snr \,\wedge\, T.Kaeufernr=V.Vnr \,\wedge\, T.Datum>=01.01.2018 \,\wedge\, \neg(V.Land=\text{„Spanien“}) \,\wedge\, S.Vertragsende=2023 \,\wedge\, S.Position=\text{„Stuermer“} \;\vee\; S.Snr=T.Snr \,\wedge\, T.Kaeufernr=V.Vnr \,\wedge\, T.Datum>=01.01.2018 \,\wedge\, \neg(V.Land=\text{„Spanien“}) \,\wedge\, S.Vertragsende=2023 \,\wedge\, S.Position=\text{„Torwart“}}(S \times V \times T)$$

(c)

Die entschachtelte SQL-Anfrage besitzt die folgende Form:

```
SELECT T.Datum, T.Abloese
FROM Transfer T, Spieler S
WHERE S.Snr = T.Snr
  AND S.Nation = 'Portugal'
```

(d)

Die Selektivität der ersten Anfrage kann folgendermaßen geschätzt werden:

$$s_\sigma(Jahresgehalt < 400000, S) = \frac{400.000 - \min Jahresgehalt}{\max Jahresgehalt - \min Jahresgehalt}$$
$$= \frac{200000}{49800000} \approx 0.004$$

Für die zweite Anfrage betrachten wir die Selektivität des Equijoins und der anschließenden Selektion getrennt. Es sei $R = S \bowtie_{Snr} T$. Da Snr in T ein Fremdschlüssel aus S ist, gibt es zu jedem Tupel von T genau einen Jointreffer. Somit ist $card(R) = card(T)$. Die Joinselektivität ergibt sich nun aus $\frac{card(R)}{card(S) \cdot card(T)} = \frac{1}{card(S)} = \frac{1}{100000}$.

Die anschließende Selektion lässt sich mit Hilfe der Formel aus dem Kurstext berechnen:

$$\begin{aligned}
&s_\sigma(Jahresgehalt > 5000000 \vee Abloese > 50000000, R)\\
&= s_\sigma(Jahresgehalt > 5000000, R) + s_\sigma(Abloese > 50000000, R)\\
&\quad - s_\sigma(Jahresgehalt > 5000000, R) \cdot s_\sigma(Abloese > 50000000, R)\\
&= \frac{45000000}{49800000} + \frac{170000000}{220000000} - \frac{45000000}{49800000} \cdot \frac{170000000}{220000000}\\
&\approx 0.9781
\end{aligned}$$

Insgesamt erhalten wir für die zweite Anfrage eine Selektivität von ca. $9.781 \cdot 10^{-6}$.