Datenbanken II

Übungsblatt 1 – WiSe 2018/19

1. Gegeben sind zwei Relationen $R(A)$ und $S(A)$. Die Werte in R sind nicht sortiert, S ist nach dem Attribut A sortiert. R und S speichern dieselben numerischen Werte, die zwischen 5.000.000 und 10.000.000 gleichverteilt sind; ein bestimmter Wert kann auch mehrfach vorkommen.

   Blockgröße $B = 8192$B. Tupelgröße $t = 120$B. $n = |R| = |S| = 1.000.000$ Tupel. Die Zeit für 1 Lesezugriff auf einen Block ist 0.02s.

   Ermittle die Ausführzeit für folgende Anfragen:

   - $\sigma_{A \neq 7.000.000}(R)$
   - $\sigma_{A \neq 7.000.000}(S)$
   - $\sigma_{A < 7.000.007}(R)$
   - $\sigma_{A < 7.000.007}(S)$

2. Eine Slotted Page der Größe 2^{13}B soll dimensioniert werden, d.h., die Größe der Felder im Kopfteil der Seite und die Adressierungsart sollen bestimmt werden. Der Kopfteil hat die Form $(a, f, g_1, p_1, g_2, p_2, \ldots, p_n, g_n)$, wobei a die Anzahl der Datensätze im Block $(= n)$ speichert, f den Free Space Pointer, g_i die Größe des i-ten Datensatzes und p_i den Pointer zum i-ten Datensatz, $1 \leq i \leq n$.

   Byte-Adressierung: Es kann jedes Byte adressiert werden. Die kleinste Adresse ist 0 und die größte Adresse ist 8191.

   Für die Byte-Adressierung werden für a, f, g_i und p_i jeweils 13 Bit benötigt (maximaler Wert 8191). Um Platz zu sparen, werden Byte-Grenzen im Kopfteil ignoriert und die Bits werden dicht gepackt, d.h., für jedes Feld werden nur 13 Bit gebraucht. Wieviele Datensätze der Größe 1 Byte, 2^5 Byte, oder 2^7 Byte können so auf eine Slotted Page gespeichert werden?

3. Betrachten Sie die folgende Tabelle:

```
CREATE TABLE boats (
  bid int,            -- 4 Bytes
  bname varchar(20)   -- 1 Byte pro Character
);
```

Die Tupel sind auf Slotted Pages der Größe 8KB gespeichert. Die Struktur der Slotted Pages ist gleich wie in Übung 5 mit der Ausnahme, dass Word-Adressierung verwendet wird.

Word-Adressierung: Es kann nur jedes zweite Byte adressiert werden. Die kleinste Adresse ist 0 und die größte Adresse ist 4095 (und adressiert das 8191. Byte)

Visualisieren Sie den Inhalt der Slotted Page (Felder **und** Werte) nach den folgenden Operationen:

```
INSERT INTO boats VALUES (1, 'Alpha');
INSERT INTO boats VALUES (2, 'Pi');
INSERT INTO boats VALUES (3, 'Epsilon');
```

Datenbanken II

Übungsblatt 2 – WiSe 2018/19

4. Erstellen und zeichnen Sie für die folgende Tabelle einen 2-stufigen Sekundärindex auf dem Attribut `Dept`. D.h. der Index hat 2 Index-Stufen, wobei die erste Stufe dense und die zweite Stufe sparse sein soll. Pro Indexblock können 10 Einträge gespeichert werden. (Da es ein Sekundärindex ist, wird angenommen, dass ein Index-Eintrag auf ein Tupel verweist).

Name	Dept	CourseNo
Donetta	CS	457
Annabelle	Arch.	45
Roosevelt	CS	27
Allyson	Socio.	470
Debra	Psych.	457
Bobbie	Psych.	27
Bradly	CS	11
Marcell	CS	470
Amanda	Psych.	470
Danelle	Socio.	470
Michael	Arch.	125
Ann	Path.	350
Tom	Gen.	291
Camilla	Pol.	11
Abdul	CS	27
Carrie	CS	27
Ewa	Psych.	125
Conrad	CS	350
Lucille	Socio.	350
Roberto	Arch.	125

5. Ein Block kann 500 Index-Einträge oder 80 Datensätze der Relation R speichern. R enthält 10.000.000 Datensätze.
   a) Wieviele Blöcke werden für einen (flachen) *dense* Index auf R benötigt.
   b) Wieviele Blöcke werden für einen (flachen) *sparse* Index auf R benötigt, der einen Eintrag pro Block der Daten Datei enthält.
6. Gegeben ist eine Relation $R[A, B, \ldots]$ mit den folgenden Eigenschaften:
   - $|R| = 1.000.000$ Tupel,
   - die Werte von Attribut A sind gleichverteilt im Intervall $[1, 100.000.000]$,
   - die Werte von Attribut B sind gleichverteilt im Intervall $[1, 1.000]$,
   - es existiert ein sparse Primärindex auf A (d.h. jeder Index-Eintrag verweist auf einen Datenblock von R),
   - es existiert ein dense Sekundärindex auf B (d.h. jeder Index-Eintrag verweist auf ein Tupel von R),
   - ein Block speichert 500 Index Einträge oder 50 Datensätze.

   Es werden folgende Anfragen auf R gestellt:

   $$\text{Q1: } \sigma_{A>60.000.000}(R), \quad \text{Q2: } \sigma_{B>600}(R)$$

   Wie viele Blöcke müssen gelesen werden, wenn die Indizes zur Beantwortung von Q1 bzw. Q2 **nicht** verwendet werden.

7. Angabe wie in Aufgabe 6.
   Wie viele Blöcke müssen gelesen werden, wenn die Indizes zur Beantwortung von Q1 bzw. Q2 verwendet werden.
8. Gegeben ist eine Relation $R[A, B, C, E]$ mit den folgenden Eigenschaften:
   - $|R| = 10.000.000$ Tupel
   - die Werte von Attribut A sind gleichverteilt im Intervall $[1, 100.000.000]$ und eindeutig,
   - die Werte von Attribut B sind gleichverteilt im Intervall $[1, 5.000]$,
   - die Werte von Attribut C sind gleichverteilt im Intervall $[1, 25.000]$,
   - die Werte von Attribut E sind gleichverteilt im Intervall $[1, 500.000]$,
   - es existiert ein flacher sparse Index auf Attribut A,
   - auf den Attributen B, C und E existiert jeweils ein flacher dense Index,
   - ein Datenblock kann 25 Einträge speichern,
   - ein Indexblock kann 100 Einträge speichern.

   Es werden die folgenden Anfragen gestellt:

   $$\text{Q1: } \sigma_{C=20.000 \wedge A<50.000.001}(R) \quad \text{Q2: } \sigma_{C=20.000 \vee A<50.000.001}(R)$$

   Beschreiben Sie die nötigen Schritte um Q1 bzw. Q2 möglichst effizient zu beantworten und geben Sie die notwendige Zahl an Blockzugriffe für die beschriebenen Schritte an.

Datenbanken II

Übungsblatt 3 – WiSe 2018/19

9. Die folgenden Werte sollen in einen B^+ Baum eingefügt werden:

$$3, 4, 6, 8, 12, 18, 20, 24, 30, 32$$

Wie sieht der B^+ Baum mit $m = 4$ bzw. $m = 5$ Zeigern pro Knoten aus, wenn diese Werte in der angegebenen (aufsteigenden) Reihenfolge eingefügt werden. Zeichnen Sie den B$^+$ Baum nach jedem relevanten Schritt, d.h. zumindest nach den Schritten, in denen sich die Anzahl der Knoten im B$^+$ Baum ändert.

10. Gegeben ist der B$^+$ Baum in Abbildung 1. Löschen Sie aus diesem B$^+$ Baum die Werte 7, 10, 6, 3, 5, 9, 1 (in dieser Reihenfolge) und zeigen Sie den B$^+$ Baum nach jedem relevanten Schritt (wie in Bsp. 9).

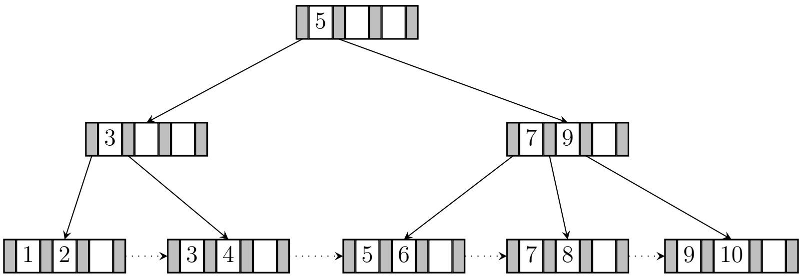

Abbildung 1: B$^+$ Baum für Aufgabe 10.

11. Gegeben ist die Relation R (siehe Tabelle 1). Zeichnen Sie einen *gültigen* B$^+$ Baum für das Attribut `RegNo` dieser Relation. Dieser B$^+$ Baum soll die *minimale* Tiefe für diese Anzahl an Schlüsseln aufweisen. Als Schlüssel werden jeweils die letzten 2 Ziffern der RegNo (**fett** gedruckt) verwendet. Bspw. ist **17** der Schlüssel für '*Jesse*'. Ein Knoten im B$^+$ Baum kann bis zu 4 Schlüssel halten, d.h. $m = 5$.

Die Tiefe eine B$^+$ Baumes ist hierbei die Anzahl der Kanten, die verfolgt werden müssen, um einen Blattknoten zu erreichen. Der Baum in Abbildung 1 hat also eine Tiefe von 2.

Die Schlüssel sind aufsteigend zu ordnen/sortieren. Die Einträge in Tabelle 1 sind bereits entsprechend sortiert.

Hinweis: Da alle Schlüssel im Voraus bekannt sind, ist es nicht zielführend die Schlüssel einzeln in den B$^+$ Baum einzufügen. Vielmehr kann der Baum *bottom-up* konstruiert werden. D.h. Sie fangen mit der Ebene der Blattknoten an und die darüberliegenden Ebenen können dann so befüllt werden, dass ein *gültiger* B$^+$ Baum mit *minimaler* Tiefe entsteht.

RegNo	Name	Dept
017000**00**	Dexter	EE
017000**01**	Arthur	CS
017000**02**	Charlie	BA
017000**03**	Ryan	DS
017000**04**	Claire	DS
017000**05**	Doug	CS
017000**06**	Alan	EE
017000**07**	Joe	CS
017000**08**	Spencer	CS
017000**09**	Debra	EE
017000**10**	Rita	CS
017000**11**	Ephraim	BA
017000**12**	Abraham	BA
017000**13**	Rick	BA
017000**14**	Lucille	CS
017000**15**	Daryl	CS
017000**16**	Walter	EE
017000**17**	Jesse	CS
017000**18**	Gustavo	DS
017000**19**	Berta	DS
017000**20**	Jake	DS

Tabelle 1: Relation R für Aufgabe 11.

12. Gegeben sei eine (unsortierte) Relation $R[A]$ und ein dense B$^+$ Baum auf dem Attribut A. Der B$^+$ Baum hat folgende Eigenschaften:

    - 100.000 **vollständig befüllte** Blattknoten
    - $m = 2^8 = 256$

    Die Werte von Attribut A sind eindeutig und fortlaufend im Bereich $[1; 25.500.000]$, beginnend mit Wert 1.

    Über die darüberliegenden Ebenen des B$^+$ Baumes ist nichts bekannt. Sie müssen also davon ausgehen, dass jeder innere Knoten nur halbvoll ist, d.h. nur $\lceil \frac{m}{2} \rceil$ Zeiger auf Kinderknoten verweisen (die restlichen Knoten-Einträge sind leer).

    Der B$^+$ Baum soll benutzt werden um folgende Anfrage zu beantworten:

    $$\sigma_{A>20.000.000 \wedge A<20.002.551}(R)$$

    Geben Sie die *einzelnen Schritte* an, die nötig sind um die Ergebnismenge aus dem B$^+$ Baum zu erhalten, illustrieren Sie das Traversieren des B^+-Baumes und berechnen Sie außerdem die *Anzahl der Blockzugriffe*, die dafür nötig sind.

    Hinweis: Die Strategie und die Anzahl der Blockzugriffe beziehen sich nur auf den B$^+$ Baum, d.h. Sie müssen keine Blockzugriffe für den Zugriff auf die Relation $R[A]$ einrechnen.

23–10–2018_PS

I. Aufgabe 1

II. Aufgabe 2

III. Aufgabe 3

Aufgabe 1

Wir rechnen aus, wieviele Tupel in einem Block sind:

Blockgroesse $B = 8192$, Tupelgroesse $t = 120$. $\frac{B}{t} = 68$ Tupel pro Block (abgerundet).

Wir rechnen weiter aus, wieviele Bloecke die Relationen brauchen: $|R| = |S| = 1.000.000$ und $\frac{|R|}{68} = 14.706$

Fuer $\sigma_{A \neq 7.000.000}(R)$ und $\sigma_{A \neq 7.000.000}(S)$: Wir lesen einfach die ganze Relation udn schmeissen unpassende Tupel raus. Bei 0,02s fuer einen Lesezugriff erhalten wir $14.706 * 0.02 = 294s$. Es hilft nichts, dass S sortiert ist.

Fuer $\sigma_{A < 7.000.007}(R)$ muessen wir auch alle Bloecke lesen (und kommen wieder auf 294), das aendert sich aber bei S: Da S sortiert ist, koennen wir mit Binarysearch das Tupel 7.000.007 finden und dann alle Werte "darunter" ausgeben. Binarysearch ist logarithmisch und wir vernachlaessigen die Laufzeit davon: $\frac{2.000.007}{5.000.000} \approx 40\%$ und $14706 * 0.4 = 5882,4$ und $14706 * 0.4 = 5882,4$ und $14706 * 0.4 = 5882,4$. Wir koennen nur ganze Bloecke lesen und runden daher auf: $\lfloor 5882,4 \rfloor = 5883$.

Aufgabe 2

Unsere Slotted Page faengt an mit a, f als Pointer zum Ende des freien Platzes, q_n und p_n halten Details zu jedem Tupel fest. Die Datensatzgroesse ist d_n und wir rechnen zunaechst die Loesung fuer $|d_n| = 2^5$ aus.

Es gilt $|a| = |f| = |q_n| = |p_n| = 13$. Wir stellen eine Formel auf:

$$|a| + |f| + n(|q_n| + |p_n| + |d_n|) \leq 2^{13} * 8$$

Wir setzen ein und vereinfachen:

$$13 + 13 + n(13 + 13 + 2^5 * 8) \leq 2^{13} * 8$$

$$13 + 13 + n(13 + 13) \leq 2^{13} * 8$$

$$n \leq \frac{2^{13} * 8 - 2 * 13}{2 * 13 + 2^5 * 8} = 232$$

Man kann also **232** Datensaetze speichern.

Aufgabe 3

Wir haben $\frac{8192}{2} = 4096$ Datensaetze mit Adressen 0 bis 4095. Wir finden die Groesse unserer Datensaetze:

$|(1, \text{'Alpha'})| = 4 + 5 = 9$ und $|(2, \text{'Pi'})| = 4 + 2 = 6$ und $|(3, \text{'Epsilon'})| = 4 + 7 = 11$. Ungerade Groessen werden aufgerundet, da unsere Adressen durch 2 teilbar sein muessen: Wir erhalten **10, 6 und 12 Bytes** als Tupelgroessen, die Datensaetze nehmen also 5, 3 und 6 Woerter ein.

Unsere Slotted Page besteht dann aus $a, f, q_{1...3}, p_{1...3}, d_{1...3}$. Die Adressen, an denen wir die Datensaetze ablegen, finden wir, indem wir sukzessive fuer jeden Datensatz von unserem Free-Space-Pointer f die Anzahl eingenommener Woerter des jeweiligen Datensatzes abziehen. f und a aendern wir nach jeder Insertion entsprechend.

Am Ende sieht unsere Slotted Page so aus:

a	f	q_1	p_1	q_2	p_2	q_3	p_3	**...freier Speicher...**	**Datensatz 3**	**Datensatz 2**	**Datensatz 1**
3	4018	9	4091	6	4038	12	4082	…	(3, ‘Epsilon)	(2, ‘Pi’)	(1, ‘Alpha’)

30–10–2018_PS

I. Aufgabe 4

II. Aufgabe 5

III. Aufgabe 6

IV. Aufgabe 8

Aufgabe 4

Zweistufiger Index. Erste Stufe dense, zweite sparse. 2 dense Stufen waeren sinnlos.

Dense Index	zeigt auf
—Block 1—	
Arch.	Annabelle
Arch.	Michael
Arch.	Roberto
CS	Donetta
CS	Roosevelt
CS	Bradly
CS	Marcell
CS	Abdul
CS	Carrie
CS	Conrad
—Block 2—	
Gen.	Tom
Path.	Ann
Pol.	Camilla
Psych.	Debra
Psych.	Bobbie
Psych.	Amanda
Psych.	Eva
Socio.	Allyson
Socio.	Danelle
Socio.	Lucille

Sparse Index	zeigt auf
Arch.	Arch. (erstes)
Gen.	Gen. (erstes)

Aufgabe 5

Sinn der Aufgabe: dense vs sparse Indices - Gefuehl kriegen.

Loesungen:
a) $\frac{10.000.000}{500} = 20.000$ Indexbloecke
b) $\frac{10.000.000}{80} = 125.000$ Datenbloecke und $\frac{125.000}{500} = 250$ Indexbloecke.

Unterschied Dense v Sparse: statt 1 Zeiger pro Tupel 1 Zeiger pro Datenblock. Wir haben Datenbloecke der Groesse 80… bemerke auch $\frac{20.000}{80} = 250$

Aufgabe 6

Gleichverteiltheit in den Intervallen ist zunaechst irrelevant

Wir haben $|R[A, B, \ldots]| = 1 * 10^6 T$ und $Q1 : \sigma_{A>60.000.000}(R)$ und $Q2 : \sigma_{B>600}(R)$

Wir finden zunaechst: Bloecke fuer den dense Sekundaerindex = $\frac{1.000.000}{500} = 2.000$, Datenblocke: $\frac{1.000.000}{50} = 20.000$, Bloecke fuer sparse Primaerindex: $\frac{20.000}{50} = 40$

Durch den sparse Primaerindex auf Attribut A ist die Relation nach A sortiert.

Nun zu $Q1$:

1. Binaere Suche auf Datenblocken… $\lceil log_2\left(\frac{10^6}{50}\right)\rceil = 15$ Datenbloecke
2. Lese restliche Datenbloecke: $40\% * 20.000 = 8.000$ Datenbloecke.

Wir haben also 8.015 Bloecke insgesamt.

Fuer den anderen Teil der Aufgabe:

1. binaere Suche auf sparse Index: $\lceil log_2\left(\frac{\lceil \frac{10^6}{50} \rceil}{500}\right)\rceil = 6$

Wir haben dann nur noch 8.006 Blockzugriffe.

$Q2$:
Wir muessen die gesamte Relation lesen, da diese nicht nach dem Attribut B sortiert ist:

1. wieder binaere Suche auf dense Index: $\lceil log_2\left(\frac{10^6}{500}\right)\rceil = 11$
2. Scan der restlichen Indexeintrage und verfolge jeden Pointer zum Datenblock: $0,4 * 2.000 = 800$ und $800 * 500 = 400.000$

Somit haben wir 400.011 Blockzugriffe. Damit ist das viel langsamer, aber wenn wir eine Punktanfrage haetten dann waere der dense index viel schneller.

Aufgabe 8

Sehr aehnlich zu Aufgabe 6.

06–11–2018-PS

I. Aufgabe 11

II. Aufgabe 12

Quiz: Stoff bis exklusive B^+ Baum. 2. Quiz wird dann der ganze B^+ Stoff

Erste Programmieraufgabe ist live

Aufgabe 11

B^+ Tiefe ist Anzahl der Kanten von Wurzel bis zum am weitesten entfernten Leaf, Hoehe ist Anzahl Knoten.

Wir haben sortierte Schluessel, muessten sie sonst noch vorsortieren.

Ein Ast ist ein Pfad von Wurzel zu Blatt. Aeste sind gleich lang, da B^+ Baeume sortiert sind

Wir haben 5 Zeiger pro Blatt (?), 4 davon koennen wir fuer Daten verwenden. Wir haben 21 Datensaetze. Dann haben wir $\left\lceil \frac{21}{4} \right\rceil = 5$ Blaetter.

Wir zeichnen zunaechst eine Kette von Blaettern:

00|01|02|03 -> … -> 20| - | - | -

Wir muessen das Ende optimieren: Blaetter muessen halbvoll sein. Schieben 19 zur 20. Koennten die 18 auch rueberschieben, macht keinen Unterschied

… -> 16|17|18| - -> 19|20| - | -

Funktioniert genauso bei der Programmieraufgabe.

Fuer die naechste Ebene: Wir machen prinzipiell dasselbe. Wir fuellen die einzelnen Eintraege fuer die Blaetter mit dem linkesten Element des Blattes direkt rechts neben dem zum Eintrag korrespondierenden Blatt. Die naechste Ebene sieht dann so aus:

04|08|12|16 und 19| - | - | -

Letztes fehlt - 19 und 20 sind durch nichts nach oben beschraenkt. Muessen wieder halb voll sein, also schieben wir wieder:

04|08| - | - und 16|19| - | -

Wir sparen uns die 12. 8–11 sind die hoechsten Werte im linken Knoten, sind uncapped

Fuer die Wurzel: Muss nicht halbvoll sein. Besteht nur aus Eintrag 12 (hier taucht sie dann auf):

12| - | - | -

Programmieraufgabe wuerde wieder sehr aehnlich gehen.

Aufgabe 12

Bereichsanfrage: Anzahl Blockzugriffe ausrechnen. Diesmal haben wir Kontengrad 256. Wir haben 2550 Tupel im gegebenen Bereich. Wir muessen also $\left\lceil \frac{2550}{255} \right\rceil = 11$ Blaetter scannen

Wir machen eine vertikale Traversierung vom B$^+$ Baum, danach gehen wir einfach von Blatt zu Blatt. Wir haben sie aus genau diesem Grund "verkettet".

Die Traversierung kostet uns im worst case $\lceil log_{\lceil \frac{M}{2} \rceil}(L) \rceil + 1 = 4$ Blockzugriffe: M ist Knotengrad, wir halbieren, da wir nur garantiert haben, dass die Knoten halb voll sind. Danach brauchen wir noch 10 Blockzugriffe auf die konsekutiven Blaetter (nicht die vorherigen 11… das macht die $+1$ beim log.

Suche ist leicht im B$^+$ Baum, Einfuegen und Loeschen schwieriger. Slides haben Pseudocode, den im Zweifel einfach 1:1 befolgen.