
Aufgabe 1 - *Slotted Page*.1 Punkt

Gegeben sei eine Slotted Page mit folgenden Eigenschaften:

- Größe: $2^{11} = 2048$ Bytes
- Adressierungstyp: **Byte-Adressierung** (es kann jedes Byte adressiert werden)
- Header: **dense** (dicht gepackt)

In dieser Slotted Page werden die Tupel R , S , T gespeichert:

- $d_1: |R| = 9$ Bytes
- $d_2: |S| = 110$ Bytes
- $d_3: |T| = 245$ Bytes

Wie groß (in Bytes) ist der Free Space?

Aufgabe 2 - Erweiterbares Hashing.

1 Punkt

Die Hashfunktion h liefert die in der Tabelle angegebenen Binärwerte. **Ergänzen Sie die fehlenden Werte/Komponenten (globale/lokale Tiefen, Pointer).**

Hash-Tabelle:

x	$h(x)$
6	1100
13	1011
20	1000
21	1010
23	1110
28	1001
33	0010
34	0110

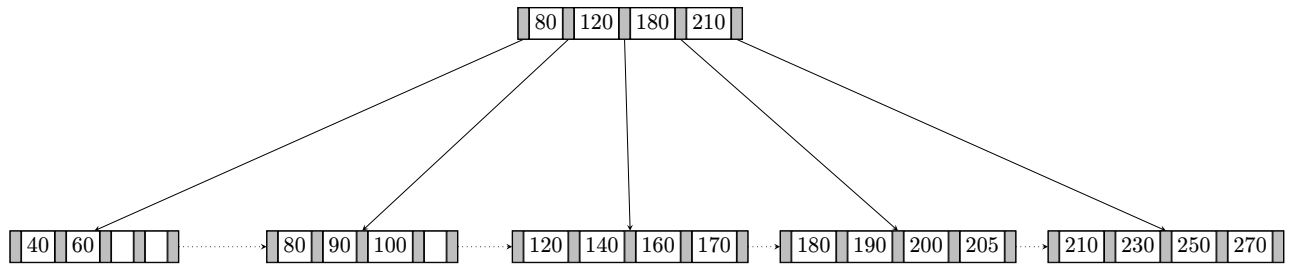
000
001
010
011
100
101
110
111

33
34
20
28
13
21
6
23

Aufgabe 3 - B^+ -Baum Löschen.

1 Punkt

Gegeben ist ein B^+ -Baum mit $m = 5$. Zeichnen Sie den B^+ -Baum, der nach dem Löschen von **60** entsteht.



Aufgabe 4 - Duplikate und Indexstrukturen.

1 Punkt

Betrachten Sie die folgende (unsortierte) Relation $R[name]$ mit Duplikaten.

- Zeichnen Sie einen **einstufigen, non-clustered Index** für Attribut *name* und lösen Sie Duplikate mittels **Buckets** auf. Hinweis: Obwohl die Relation Duplikate enthält, darf der erstellte Index keine Duplikate in den Suchschlüsseln enthalten.
- Duplikate können auch über **Tupel Identifier (TID)** aufgelöst werden. Erklären Sie, was ein **TID** ist, und **wie TIDs benutzt werden können** um in einem Index Duplikate in den Suchschlüsseln zu vermeiden.

 $R[name]:$

Noah
George
Noah
Ivy
Harry
Noah
George
Lily
Ivy
Florence
Noah
George

Aufgabe 5 - *Bitmap Index Scan*.1 Punkt

Gegeben sei eine Relation $R[A, B]$ mit folgenden Eigenschaften:

- $|R| = 10^6$ Tupel gespeichert auf $b_R = 10^3$ Blöcken,
- Non-clustering B⁺-Baum-Index auf Attribut B , $m = 1024$, maximale Höhe (d.h. die Knoten sind minimal befüllt),
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst,
- Attribut B hat insgesamt 10^3 mal den Wert 2500, verteilt auf 100 Datenblöcke.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{B=2500}(R)$$

Geben Sie die Anzahl an **Datenblockzugriffen (worst case)**

- a. **ohne Anwendung** eines Bitmap Index Scans (**0.5 Punkte**),
- b. **unter Anwendung** eines Bitmap Index Scans (**0.5 Punkte**)

an. Es darf vereinfachend angenommen werden, dass bereits gelesene Datenblöcke nicht gepuffert werden.

Aufgabe 6 - *Join-Algorithmen*.1 Punkt

Gegeben seien **zwei Relationen** R und S mit folgenden Eigenschaften:

- $R[A]$ mit Werten 1, 2, 3, 4, 5, 6, 7, 8, 9,
- $S[A]$ mit Werten 2, 5, 6, 8, 9, 11, 13, 15, 16, 18, 19, 20,
- die Größe des Puffers beträgt $M = 4$ Blöcke und
- jeder Block fasst 1 Tupel.

Zeichnen Sie die Partitionen für einen Hash Join für beide Relationen. Erklären Sie Schritt für Schritt wie die Partitionen entstehen. Die Hashfunktion darf frei gewählt werden.

Aufgabe 7 - Effiziente Anfragebearbeitung.1 Punkt

Gegeben sei eine Relation $R[A, B, C]$ mit folgenden Eigenschaften:

- $|R| = 10 \cdot 10^{12}$ Tupel.
- Pro Datenblock werden $25 \cdot 10^3$ Tupel gespeichert.
- Attribut A hat ganzzahlige Werte gleichverteilt im Bereich $[1; 20 \cdot 10^6]$.
- Attribut B hat ganzzahlige Werte gleichverteilt im Bereich $[1; 20 \cdot 10^9]$.
- Attribut C hat ganzzahlige Werte gleichverteilt im Bereich $[1; 20 \cdot 10^3]$.
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst.
- Folgende Indizes existieren:
 - Einstufiger sequentieller sparse Index auf Attribut A mit $5 \cdot 10^6$ Einträgen pro Indexblock,
 - dense B^+ -Baum-Index auf Attribut B , $m = 1.024$, minimale Höhe.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{A=7 \cdot 10^3 \vee B=7 \cdot 10^3}(R)$$

Geben Sie die **Strategie (0.5 Punkte)** an und berechnen Sie die **Anzahl der Blockzugriffe (0.5 Punkte)** um die Anfrage **möglichst effizient** zu beantworten (1 Knotenzugriff im B^+ -Baum entspricht 1 Blockzugriff).

Aufgabe 8 - *Schätzung der Join-Kardinalität.*1 Punkt

Gegeben seien 3 Relationen $R[A, B, C]$, $S[A, D, E]$, $T[D, E, F]$ mit folgenden Eigenschaften:

- $|R[A, B, C]| = 1000$ Tupel, $V(R, A) = 100$, $V(R, B) = 200$, $V(R, C) = 300$
- $|S[A, D, E]| = 4000$ Tupel, $V(S, A) = 50$, $V(S, D) = 200$, $V(S, E) = 300$
- $|T[D, E, F]| = 2000$ Tupel, $V(T, D) = 200$, $V(T, E) = 400$, $V(T, F) = 600$

Die Werte in den Tupeln sind gleichverteilt und unabhängig. Schätzen Sie die Kardinalität der folgenden Anfrage ab. ($\sigma_{A=100}(R) \neq \emptyset$).

$$(\sigma_{A=100}(R)) \bowtie S \bowtie T$$

Aufgabe 9 - Recoverability.1 Punkt

Gegeben sei die folgende Historie (Schedule) bestehend aus drei Transaktionen T1-T3.

- Ist die gegebene **Historie recoverable**? **Begründen Sie Ihre Antwort. (0.5 Punkte)**
- Wenn **kaskadierende Rollbacks** nötig sind, dann identifizieren Sie alle Transaktionen für die ein solches Rollback durchgeführt werden muss. Wenn kein kaskadierendes Rollback nötig ist, dann **begründen Sie Ihre Antwort. (0.5 Punkte)**

T1:	T2:	T3:
read(X)		
write(X)		
	read(X)	
	write(Y)	
		read(Z)
		write(Z)
		commit
abort		

Aufgabe 101 Punkt

Betrachten Sie den folgenden Schedule und den **Two-Phase Locking** Scheduler.

- (a) Nehmen Sie an, dass alle Transaktionen eine **write** Operation auf Item **A** in der folgenden Ordnung durchführen wollen: **T3**, **T2**, **T1**. Komplettieren Sie den Schedule, indem Sie Lock Requests (e.g., **R:lock-S(A)**) und Granted Locks (e.g., **G:lock-S(A)**) für alle verbleibenden **read** und **write** Operationen einfügen. Markieren Sie im Schedule, ob eine Transaktion aufgrund eines inkompatiblen Locks einer anderen Transaktion warten muss.
- (b) Zeichnen Sie den Wait-For Graphen nach Bearbeitung aller angegebenen Lock Requests. Resultiert dieser Schedule in einem Deadlock? Begründen Sie Ihre Antwort.

T1: **T2:** **T3:**

R:lock-S(A)

G:lock-S(A)

read(A)

read(A)
