Aufgabe 1

1. - *Was ist ein Index über einem File of Records?*
   Ein Suchbaum, der auf einer Relation effizientes Aufsuchen von Records ermöglicht.

   - *Was ist der search key eines INdexes?*
   Er entspricht recordweisen Metadaten, welche zur eindeutigen Suche auf dem Suchbaum verwendet werden.

   - *Wozu benötigen wir Indizes?*
   Sie ermöglichen das effiziente Aufsuchen von Records einer bestimmten Range, da bei der Suche größere Schritte gemacht werden.
2. *Welche grundlegende Designalternativen gibt es bezogen auf die Einträge eines Indexes?*
   Alternativen im Umfang des Eintrags:

   - eines Records rid
   - mehrerer Records rid
   - einen Record komplett
3. - *Was ist der Unterschied zwischen einem clustered und einem unclustered Index?*

     clustered Zu Deutsch: zusammen gebunden. Die Daten zwischen Index und Datei befinden sich gebunden in der gleichen Reihenfolge angeordnet.

     unclustered Die Daten der Datei werden frei geordnet.

   - *Kann ein Index unclustered sein, wenn dessen Einträge die Daten-Records enthalten?*
   Ein solcher die Daten-Records enthaltender Index ist immer clustered.

Aufgabe 2

Aufgabe 3

1. • *Schätzen sie für jede der drei in der Vorlesung erläuterten, grundlegenden Datei-Organisationen die Kosten der Query für den Fall, dass die Bedienung auf kein Tupel der Relation zutrifft.*

   heap Zugriff auf jede Page und vergleichen aller Tupel

   $$\text{Delete}_{\text{heap}} = b \cdot (D + r \cdot (C + H))$$

   sorted Die binäre Suche findet Page und Tupel

   $$\text{Delete}_{\text{sorted}} = log_2\,(b) \cdot D + log_2\,(r) \cdot (C + H)$$

   hashed Konstanter Pagezugriff durch Hash, lineare Pagesuche

   $$\text{Delete}_{\text{hashed}} = H + D + r \cdot C$$

   • *WIe hoch sind die Kosten, wenn* `a` *kein Schlüssel von* `R` *ist?*

   heap, sorted Zugriff auf jede Page und vergleichen aller Tupel

   $$\text{Delete}_{\text{sorted}} = \text{Delete}_{\text{heap}} = b \cdot (D + r \cdot (C + H))$$

   hashed Zugriff auf jede Page und vergleichen aller Tupel

   $$\text{Delete}_{\text{hashed}} = \underbrace{(100/80)}_{1.25} \cdot (D + r \cdot (C + H)) = \underbrace{(100/80)}_{1.25} \cdot \text{Delete}_{\text{heap}}$$

2. *[..] Ist hier das Benutzen des Indexes immer eine gute Idee?* Nein, da die Heap-Datei unsortiert ist, gilt es folgendes abzuwägen:
   Potenziell ist bei einer Range-Query das Auswerten mit einem Index aufgrund der Suche im Baum schneller, da jedoch in der Datei, unclustered, zwischen Pages gesprungen wird verzögert sich der Zugriff, sodass ein Sequential Scan mit dem Vermeiden von Seeks deutlich schneller ist.

Aufgabe 4

1. *Gegeben Sei ein unclustered B+-Index. Berechnen Sie die Ordnung d der inneren Knoten des Baumes unter der Annahme folgender Größen: Pagesize* $p = 4096\,\text{Bytes}$*, Schlüsselgröße* $s = 4\,\text{Bytes}$*, Pointerlänge* $v = 6\,\text{Bytes}$*.*

$$\underbrace{(n+1)}_{\#\text{ Pointer}} \cdot 6\,\text{Bytes} + \underbrace{n}_{\#\text{ Schlüssel}} \cdot 4\,\text{Bytes} = 4096\,\text{Bytes} \Rightarrow (n+1)) = \underbrace{410}_{\text{maximale }\#\text{ Pointer á Page}}$$

   Mit Ordung d hat jede Page des B+-Baumes mit $d \leq k \leq 2d$ einen Füllgrad größer $\frac{1}{2}$:

$$d = \left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{409}{2} \right\rceil = 205$$

2. *Wie groß ist in diesem Fall der fanout des Baumes?*
   Mit der maximalen Pointeranzahl einer Page $\frac{n+1}{2} \leq F \leq n$ ist der fanout:

$$205 \leq F \leq 410 \Rightarrow F_{\max} = 410$$

3. *Angenommen die indexierte Relation habe* 10^8 *Tupel. Wie tief ist der B+-Baum?*

$$\left\lfloor \log_{205}\left(10^8 + 1\right) \right\rfloor = 3$$

   Bei einem fanout von 205 hat der Baum mindestens eine Tiefe von 3. *Wie tief wäre stattdessen ein balancierter binärer Suchbaum für diese Relation?*

$$\left\lfloor \log_{2}\left(10^8 + 1\right) \right\rfloor = 26$$

   Der Binäre Baum hat eine Mindesttiefe von 26.