Datenbanken II

Wintersemester 2017/18

Checkliste Quiz 2

Daniel Kocher, Willi Mann

December 14, 2017

Diese Checkliste soll den Studierenden bei der Quiz-Vorbereitung helfen. Es werden Themengebiete/Fragestellungen aufgezählt, die die Studierenden auf jeden Fall beherrschen sollen. Wir weisen allerdings darauf hin, dass die Bearbeitung der Checkliste alleine keine Garantie ist, dass das Quiz positiv bestanden wird. Sie stellt lediglich ein weiteres Hilfsmittel zur Vorbereitung dar.

Die genannten Themengebiete sind aus praktischer Sicht zu verstehen, d.h. beim Quiz werden praktische Beispiele gelöst (ähnlich wie im Proseminar).

1 B^+ Baum

Kapitel 2 aus dem zweiten VO-Foliensatz *Indexstrukturen* [1].

- Grundlegende Eigenschaften (B^+ Baum, Blattknoten, innere Knoten)
- Vor- u. Nachteile (im Vergleich zu sequentiellen Indexstrukturen und Hashing)
- Unterschied gerader (bspw. $m = 6$) bzw. ungerader Knotengrad (bspw. $m = 9$)
- Suche im B^+ Baum; *vgl. Aufg. 15, UB6*
  - Algorithmus vom VO-Foliensatz (S. 31/90) anwenden
  - Unterschied Suche im B^+ Baum u. Suche in sequentiellem Index
  - Traversierung illustrieren
- Einfügen im B^+ Baum; *vgl. Aufg. 12, UB5*
  - Algorithmus vom VO-Foliensatz (S. 37/90, 38/90) anwenden
  - Blattknoten/inneren Knoten teilen
- Löschen im B^+ Baum; *vgl. Aufg. 13, UB5*
  - Algorithmus vom VO-Foliensatz (S. 48/90) anwenden
  - Vereinigen von Blattknoten/inneren Knoten
  - Verteilen von Einträgen von Blattknoten/inneren Knoten
- Bottom-up-Konstruktion eines B^+ Baumes mit minimaler Tiefe; *vgl. Aufg. 14, UB5*
- Bottom-up-Konstruktion eines B^+ Baumes mit maximaler Tiefe; *vgl. Aufg. 14, UB5*
- Anzahl der Knoten pro Ebene im Worst case berechnen (B^+ Baum hat maximale Tiefe); *vgl. Aufg. 15, UB6*
- Anzahl der Knoten pro Ebene im Best case berechnen (B^+ Baum hat minimale Tiefe); *vgl. Aufg. 15, UB6*

[1] `https://dbresearch.uni-salzburg.at/teaching/2017ws/db2/db2_02-handout-1x1.pdf`

2 Statisches Hashing

Kapitel 3 aus dem zweiten VO-Foliensatz *Indexstrukturen* [1].

- Vor- u. Nachteile (im Vergleich zu sequentiellen Indexstrukturen und B^+ Bäumen)
- Organisation einer Relation als Hash-Datei
  - Ist eine Hash-Datei ein clustering oder ein non-clustering Index? Warum?
  - Ist eine Hash-Datei ein Sekundär- oder ein Primärindex? Warum?
  - Gegebene Relation als Hash-Datei organisieren (Hash-Funktion gegeben). Anwendung anhand eines Beispiels; *vgl. Aufg. 16, UB6*
- Hash-Index einer Relation
  - Ist ein Hash-Index dense oder sparse? Warum?
  - Ist ein Hash-Index ein Sekundär- oder ein Primärindex? Warum?
  - Hash-Index für eine gegebene Relation zeichnen (Hash-Funktion gegeben). Anwendung anhand eines Beispiels; *vgl. Aufg. 16, UB6*
- Unterschied Hash-Datei und Hash-Index
- Suche in Hash-Datei bzw. Hash-Index (einzelne Schritte)
- Bucket Overflows
  - Wann tritt ein Bucket Overflow auf?
  - Welche Möglichkeiten gibt es mit Bucket Overflows umzugehen?
  - Was ist Skew und wie kommt Skew zustande?
  - Overflow Chaining (Closed Addressing) anwenden; *vgl. Aufg. 16, UB6*
- Hash-Funktionen
  - Eigenschaften einer guten (bzw. idealen) Hash-Funktion?
  - Was passiert wenn eine schlechte (bzw. die schlechteste) Hash-Funktion verwendet wird? Was sind die Auswirkungen auf die Performance?

3 Dynamisches Hashing

Kapitel 4 aus dem zweiten VO-Foliensatz *Indexstrukturen* [1].

- Warum ist dynamisches Hashing relevant (es gibt ja bereits statisches Hashing)?
- Erweiterbares Hashing
  - Was ist der Hash-Prefix und wie hängt er mit der Größe des Verzeichnisses zusammen?
  - Unterschied zwischen globaler Tiefe i und lokaler Tiefe i_j

  - Wieviele Einträge hat ein Verzeichnis mit einer gegebenen globalen Tiefe i? Anwendung anhand eines Beispiels
  - Suche: Algorithmus vom VO-Foliensatz (S. 73/90) anwenden
  - Einfügen: Algorithmus vom VO-Foliensatz (S. 74/90, 75/90) anwenden; *vgl. Aufg. 17, UB6*
  - Löschen; *vgl. Aufg. 18, UB6*
    * Algorithmus vom VO-Foliensatz (S. 77/90) anwenden (*mit Verkleinerung des Verzeichnisses*)
    * Wann kann *nicht* mit Nachbarbuckets verschmolzen werden? Anwendung anhand eines Beispiels
    * Was passiert, wenn ein Schlüssel gelöscht wird, der mehrfach in einem Bucket vorkommt? Anwendung anhand eines Beispiels
  - Auch beim erweiterbarem Hashing wird manchmal Overflow Chaining benötigt. Wann ist dies der Fall? Anwendung anhand eines Beispiels
  - Gegeben sei eine unvollständige Darstellung eines Hash Indexes, der per erweiterbaren Hashing erstellt wurde. Sie müssen diese Darstellung vervollständigen. Bspw. könnten fehlen:
    * Die globale bzw. eine/mehrerer lokale Tiefe/n
    * Einzelne Einträge in Buckets bzw. dem Verzeichnis
    * Ein oder mehrere Pointer
    * Ganze Buckets
- Vor- u. Nachteile
- Vergleich B^+ Baum und Hash-Index
  - Komplexität von Punktanfragen verstehen
  - Welche Indexstruktur ist für welche Anfragetypen besser geeignet und warum? Anwendung anhand eines Beispiels: Für gegebene Anfrage auf einer Relation sich für eine Indexstruktur entscheiden, die die Anfrage am effizientesten beantwortet.