Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Übungen zur Vorlesung
"Datenbanksysteme II"
SS 2014
Benjamin Dietrich (b.dietrich@uni-tuebingen.de)

6. Übungsblatt

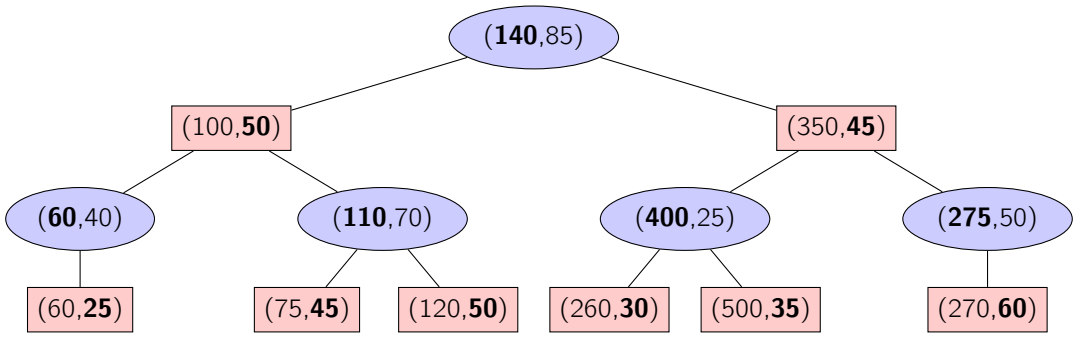
Ausgabe: 14. Mai 2014 · Abgabe: 20. Mai 2014

Aufgabe 1: B+ Baum IV **(4 Punkte)**

Stellen Sie sich vor, eine Datenbankseite kann höchstens vier Werte vom Typ `INTEGER` enthalten. Konstruieren Sie B^+-Bäume der Ordnung $d = 2$, so dass sich folgende Szenarien einstellen:

1. Einen B^+-Baum, in welchem das Löschen des Schlüssels 25 zu einer Umverteilung (*redistribution*) auf Blatteebene führt. Zeigen Sie die Struktur des Baumes vor und nach dem Entfernen des Schlüssels.
2. Einen B^+-Baum, in welchem das Löschen des Schlüssels 25 zu einer Verschmelzung (*merge*) von zwei Knoten auf Blattebene führt, aber ohne die Höhe des Baumes zu ändern.

Abbildung 1: k-d tree over attributes `salary` (○) and `age` (□)



Aufgabe 2: k-d Tree vs. B^+-Tree with Composite Key (8 Punkte)

1. Betrachten Sie für diese Aufgabe zunächst den k-d Baum in Abbildung 1. Auf welche Schlüsselwerte (Knoten und Blätter) des Baumes muss zugegriffen werden, um die folgenden Queries zu beantworten?

   (a) `salary = 120`

   (b) `age = 50`

   (c) `salary <= 300 AND age <= 40`

2. Eine alternative Strategie um mehrere Attribute in einem Schlüssel zusammenzufassen sind B^+-Bäume mit *composite search-keys*. Konstruieren Sie einen B^+-Baum mit einem solchen zusammengesetzten Schlüssel <`salary,age`> und $d = 2$ aus den Schlüsselwerten des k-d Baumes in Abbildung 1. Sortieren Sie dazu die Tupel entsprechend und bauen Sie den Baum mit Hilfe der *bulk loading* Methode auf.

3. Auf welche Schlüsselwerte des B^+-Baumes aus Nr. 2 muss nun zugegriffen werden, um die Queries aus Nr. 1 mit dessen Hilfe zu beantworten? (Hinweis: auch innerhalb der Blattknoten darf die Sortierung ausgenutzt werden. Sie dürfen hier von einer optimalen Suche ausgehen.)

Aufgabe 3: B^+-Bäume über Z-Codes (8 Punkte)

Eine weitere Möglichkeit, mehrdimensionale Daten mit Hilfe eines B^+-Baumes zu indexieren sind *Z-Codes*. In dieser Aufgabe wollen wir veranschaulichen, dass in einem solchen UB-Baum mit Z-Ordnung Nachbarschaft im mehrdimensionalen Raum zu physischer Lokalität führt: benachbarte Werte sind auf einer Blattseite des B^+-Baumes zu finden.

Um dies exemplarisch darzustellen gehen Sie wie folgt vor:

- Erzeugen Sie zunächst eine Tabelle `points(x INT, y INT)` welche Punkte in einem 2-dimensionalen Raum repräsentiert. Befüllen sie diese mit allen möglichen Punkten im Bereich $x = [0, 99]$, $y = [0, 99]$.[1]

- Erzeugen Sie nun einen Index über dem Z-Code dieser Punkte auf der Tabelle:

  ```
  CREATE INDEX zindex ON points USING btree (point_to_z(x,y));
  ```

  Die hierzu erforderliche PL/pgSQL-Funktion können Sie einfach aus Abbildung 2 herauskopieren.[2]

- Erzeugen Sie außerdem zum Vergleich einen zusammengesetzten Index `cindex` mit einem *composite seach-key* über den Spalten <`x,y`> der Tabelle `points`.

- Schreiben Sie nun eine Query welche Ihnen die `x`- und `y`-Koordinaten aller Punkte ausgibt, die sich gemeinsam auf einer beliebigen gegebenen Leaf-Page (z.B. 7) eines btree-Indexes befinden. (Hinweis: Nutzen Sie hierzu `CREATE EXTENSION pageinspect;`. Die Funktion `bt_page_items('zindex', 7)` lieftert ihnen dann unter anderem die `ctid` der Index-Einträge auf der Leaf-Page 7.)

- Veranschaulichen Sie nun wie weit die Werte der gewählten Leaf-Page im 2-dimensionalen Raum voneinander entfernt sind. Plotten Sie hierzu jeweils (für `zindex` und `cindex`) die Punkte in ein Koordinatensystem der Größe $100x100$. (Hinweis: dazu können Sie z.B. eine Online-Version von *gnuplot* verwenden: http://gnuplot.respawned.com.[3])

- Beschreiben und erklären sie *kurz* das beobachtete Ergebnis.

[1] Zur Erinnerung: Die Funktion `genereate_series(a,b)` erzeugt eine Sequenz der Zahlen von `a` bis `b`.

[2] Außerdem steht sie – falls kopieren Fehler erzeugt – unter http://db.inf.uni-tuebingen.de/staticfiles/teaching/ss14/db2/point_to_z.sql zum Download bereit.

[3] Beispieldateien für *Data* und *Plot script* finden Sie unter http://db.inf.uni-tuebingen.de/staticfiles/teaching/ss14/db2/gnuplot.zip.

Abbildung 2: Berechnet den Z-Code zu einem Punkt `(x,y)`

```
CREATE OR REPLACE FUNCTION point_to_z(x int, y int)
RETURNS int AS $$
DECLARE
  z bit varying := '';
BEGIN
  FOR i IN 0..6 LOOP
      z := (x >> i) :: bit(1) || (y >> i) :: bit(1) || z;
  END LOOP;
  RETURN z :: bit(14) :: int;
END
$$ LANGUAGE PLPGSQL IMMUTABLE;
```