Datenbanksysteme WS 2017/18
Prof. Dr.-Ing. Sebastian Michel
Manuel Hoffmann, Marc Dörr
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 6: Ausgabe 05.12.2017, Abgabe 11.12.2017 16:00 MEZ

Aufgabe 1: Min-Hashing (1 P.)

In dieser Aufgabe geht es darum, Kunden auf Grund ihrer bestellten Teile zu vergleichen. Kunden werden im TPC-H-Schema etwa durch die Spalte `c_custkey` in der Tablle `customer` identifiziert, Teile durch die Spalte `l_partkey` in der Tabelle `lineitem`. Die Information "Welcher Kunde hat was bestellt?" kann über einen Join mit der Tabelle `orders` herausgefunden werden. Die Ähnlichkeit selbst wird mittels Jaccard-Koeffizienten beziehungsweise Min-Hashing gemessen.

a) Sichten erstellen

   i) Erstellen Sie eine materialisierte Sicht[1] im TPC-H Schema `cust_parts(custkey, partkey)`. Diese Sicht stellt dar, welche Kunden welche Teile gekauft haben.

   ii) Erstellen Sie eine materialisierte Sicht `cust_part_hashes(custkey, h1, h2, h3, h4)`, die auf der vorherigen aufbaut. Diese Sicht soll zu jedem custkey die Minima der Hashfunktionen h_1 bis h_4 angewendet auf die jeweiligen partkeys beinhalten. Die Hashfunktionen sind $h_1(x) = x \bmod 5531$, $h_2(x) = x \bmod 6173$, $h_3(x) = x \bmod 6803$, $h_4(x) = x \bmod 7919$.

b) Anfragen auf die Sichten stellen

   i) Ein neuer Kunde K interessiert sich für die Teile mit partkey = 91103, 24814, 32395. Als Ähnlichkeitsmaß zwischen Kunden wählen Sie die Anzahl der bestellten Teile. Schreiben Sie eine SQL-Abfrage, die alle Kunden aus der Datenbank liefert, die eine geschätzte Ähnlichkeit echt größer als 0 mit K haben.

   ii) Der Kunde mit custkey = 494 will beraten werden. Mit welchen anderen Kunden hat er eine geschätzte Ähnlichkeit ≥ 0.5?

   iii) Welche Anfrage liefert alle Kundenpaare, die mindestens eine geschätzte Ähnlichkeit von 0.5 haben? Und welche Indexe können Sie anlegen, um diese Anfrage zu unterstützen? Vergleichen Sie Ihre Annahmen mit der Ausgabe von PostgreSQLs EXPLAIN.

c) Wir wollen nun den Fehler unserer Schätzung für den Kunden mit custkey = 494 bewerten. Legen Sie dazu eine neue Tabelle an, in der Sie die custkey-Nummern aller anderen Kunden zusammen mit den geschätzten Ähnlichkeitswerten basierend auf den vier in (a.ii) genannten Hashfunktionen ablegen. In weiteren Spalten sollen nun der korrekte Jaccard-Koeffizient basierend auf den jeweils bestellten Teilenummern sowie der Fehler gespeichert werden. Wie groß ist der durchschnittliche Fehler?

Hinweise: Die Berechnungen der letzten Teilaufgabe können schnell sehr groß werden und damit auch sehr lange dauern. Machen Sie sich daher vorher Gedanken, wie ihre Anfrage aussehen könnte. Wenn Sie wissen wollen, ob die Anfrage syntaktisch richtig ist, reicht es auch, sich den Ausführungsplan ausrechnen zu lassen. Falls die Anfragen auf Ihrem System zu lange dauern, schränken Sie die materialisierten Sichten darauf ein, nur einen Teil der Kundendaten zu benutzen.

[1] `create materialized view cust_part_hashes(custkey, h1, h2, h3, h4) AS ....` Siehe auch `https://www.postgresql.org/docs/9.6/static/sql-creatematerializedview.html`

Aufgabe 2: kd-Baum (1 P.)

a) Fügen Sie folgende Werte in der angegebenen Reihenfolge in einen zunächst leeren kd-Baum mit $k = 3$ ein:

1. (8, 2, 77)
2. (10, 15, 15)
3. (95, 60, 56)
4. (16, 54, 84)
5. (62, 36, 4)
6. (28, 33, 88)
7. (50, 44, 37)
8. (39, 21, 76)
9. (86, 12, 82)
10. (2, 39, 68)
11. (54, 28, 15)
12. (48, 72, 44)

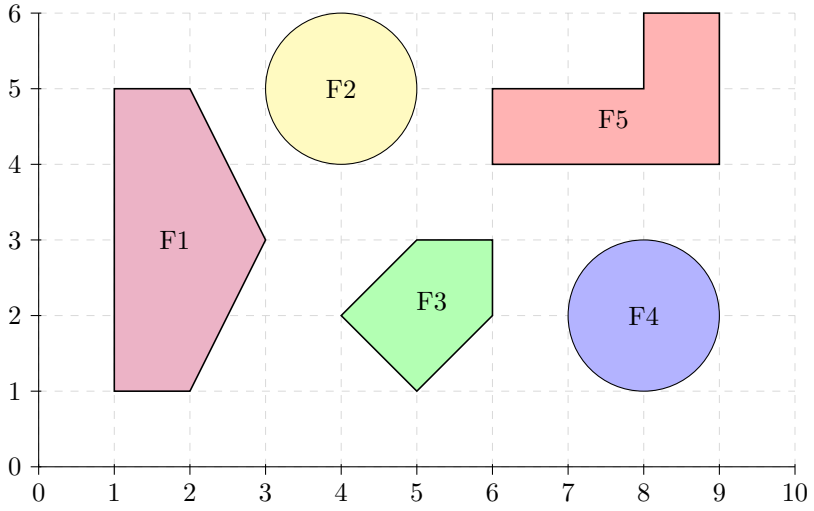
Zeichnen Sie das Ergebnis als Baumstruktur analog zur Darstellung im Skript. Machen Sie deutlich, welche Werte zum Splitten benutzt werden.

b) Adaptiver kd-Baum

Geben Sie einen Algorithmus an, der aus einer gegebenen Liste von Koordinaten in k Dimensionen einen kd-Baum konstruiert, in dessen Blättern höchstens zehn Datenpunkte sein dürfen. Die gewählte Dimension für jeden Split soll von der Varianz abhängen, die Splitkoordinate vom Durchschnitt der entsprechenden Werte der relevanten Punkte.

Aufgabe 3: R-Baum (1 P.)

Gegeben sind die wie folgt angeordneten zweidimensionalen Objekte.



Geben Sie für die folgenden R-Baum Operationen jeweils genau an, welche Schritte ausgeführt werden:

a) *Speichere die Objekte in der Reihenfolge F1, F2, F3, F4, F5 in einen zunächst leeren R-Baum.* In einen Knoten passen zwischen 1 und 2 Einträge.

b) *Finde alle Objekte, die den Punkt (6, 2) enthalten.*

c) *Finde alle Objekte, die sich vollständig im Rechteck Q, welches durch* $(2, 1)$ *und* $(9, 3)$ *definiert ist, befinden.*

d) *Finde alle Objekte, die sich mit dem Rechteck Q':* $(2, 2), (4, 4)$ *überlappen.*

e) *Suche alle nächsten Nachbarn des Punktes* $(4, 3)$. Geben Sie hierbei an, welche Pruning-Strategien Sie benutzt haben und erläutern Sie, wieso diese hier ein korrektes Ergebnis liefern.

Aufgabe 4: Raumfüllende Kurven für NN-Anfragen (1 P.)

Gegeben eine Tabelle namens `mydata`, die Datenpunkte enthält, und eine Tabelle namens `anfragen`, die Anfragen enthält. Die Tabellen sind wie folgt definiert:

```
CREATE TABLE mydata (id serial, d1 integer, d2 integer, d3 integer, zcurve_value integer);
CREATE TABLE anfragen (id serial, d1 integer, d2 integer, d3 integer, zcurve_value integer);
```

Auf der Vorlesungswebseite finden Sie einen Link zu dem Postgresql-Dump, der Beispieldaten zu beiden Tabellen enthält.

a) In den vorgegebenen Dumps fehlen die `zcurve_value` Einträge. Implementieren Sie eine User-Defined-Function in Postgresql:

```
CREATE or REPLACE FUNCTION zcurve(d1 int, d2 int, d3 int)
RETURNS int
AS $$
...
$$ LANGUAGE plpgsql;
```

welche für einen die Koordinaten eines Punktes in drei Dimensionen den entsprechenden Wert der Z-Kurve berechnet. Gehen Sie davon aus, dass jede Koordinate aus einem vorzeichenlosen Integer-Wert $< 2^{10}$ besteht. Füllen Sie mittels dieser Funktion die o.g. Tabellen.

Mit folgender materialisierten Sicht werden nun für jede Anfrage die Distanzen bzgl. echten Werten und Werten der Z-Kurve berechnet.

```
CREATE MATERIALIZED VIEW all_distances AS (
SELECT a.id as aid, m.id as did, sqrt((m.d1-a.d1)^2+
       (m.d2-a.d2)^2+(m.d3-a.d3)^2) as rdistance,
 sqrt((m.zcurve_value-a.zcurve_value)^2) as zdistance
FROM mydata m, anfragen a)
```

b) Beantworten Sie anhand der oben erstellten Daten folgende Fragen:

- Betrachten Sie zu jeder Anfrage die 3 nächsten Punkte (3-NN) anhand der echten Werte und der Z-Kurven-Werte. Bei welchen Anfragen gibt es Übereinstimmungen bei den 3-NN? Geben Sie neben der Lösung auch die SQL-Anfrage(n) an, mit der die Lösung berechnet wurde.
- Berechnen Sie für jede Anfrage den Mean-Squared-Error (MSE) über die 3-NN bzgl. der Z-Kurven Werte im Vergleich zur echten Distanz. Geben Sie ferner den maximalen und minimalen MSE über alle Anfragen an. Geben Sie auch hier neben der Lösung die SQL-Anfrage(n) an, mit der die Lösung berechnet wurde.

Hinweise: Folgender Java-Code berechnet den Z-Wert für einen zweidimensionalen Punkt mit Koordinaten (d1,d2), für Koordinaten je in $[0, 2^5]$.

```
static int zcurve(int d1, int d2) {
  int ret = 0;
  for (int loop=0; loop<5; loop++) {
    if ((d1 & (1<<loop)) == (1<<loop))
      ret = ret | (1<<2*loop);
    if ((d2 & (1<<loop)) == (1<<loop))
      ret = ret | (1<<(2*loop+1));
  }
  return ret;
}
```

Um in Postgres Zahlen als Bitstring der Länge n angezeigt zu bekommen, können Sie die Zahl zu `bit(n)` casten. Zum Beispiel:

```
select 5::bit(8); -- liefert: 00000101
```

Aufgabe 5: Dies und Das (0 P.)

Diese Aufgaben dienen der Prüfungsvorbereitung. Einige davon sind das Ergebnis von *Aufgaben Erfinden* auf Blatt 2.

a) Gibt es Sequenzen von Seitenzugriffen, bei denen LRU und LFU gleich viele Misses haben?

b) Wie viele Seitenaufrufe braucht man mindestens, um bei einer Puffergröße von n einen Unterschied zwischen LRU und FIFO zu zeigen?

c) Erläutern Sie die Fünf-Minuten-Regel.

d) Angenommen, sequentielles und wahlfreies Lesen haben die selben Kosten. Wo liegt der Break Even Point?

e) Ermöglicht der B$^+$Baum eine sequentielle oder eine hierarchische Suche?

f) Wann muss ein Datenbanksystem Daten sortieren?

g) Welche Art von Abfragen profitieren in der Regel am meisten vom Clustering?

h) Ist die Reihenfolge der Attribute eines Composite Key ausschlaggebend für die Effizienz einer Anfragebeantwortung?

i) Beschreiben Sie, wie durch Hashing Duplikate entfernt werden können.

j) Wann könnte es Sinn machen ein Relationenschema in 3NF Relation einem Relationenschema in BCNF vorzuziehen?

k) Ist der Nested-Loop-Join immer langsamer als der Sort-Merge-Join?

l) Ist ein Hash-Join immer besser als ein Nested-Loop-Join?

m) Wie viele links-tiefe Bäume sind bei Stern-Anfragen mit n Relationen möglich, wenn keine Kreuzprodukte zugelassen werden?

n) Geben Sie ein Beispiel mit drei Relationen an, für welches der Join, der ein Kreuzprodukt enthält, am günstigsten ist.

o) Beurteilen Sie kurz die Sinnhaftigkeit eines Equi-Width-Diagramms (Width $= 4$) bei gegebenem $V(R, A) = 4$.

p) Was unterschiedet V-optimale Histogrammen von Histogrammen, die diskret gegebene Werte abbilden?

q) Beschreiben Sie das Histogramm, das mit folgender Funktion abgeschätzt wird: $f(x) = n$.

r) Gegeben eine Zufallsvariable X mit Verteilungsfunktion F. Was ist das p Quantil von X?

s) Welche ursprüngliche Werte werden durch die Wavelet-Transformierte $[4, 2, 1, 0]$ beschrieben?