Datenbanksysteme WS 2016/17
Prof. Dr.-Ing. Sebastian Michel
BSc. Peter Brucker
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 12 (Das ist das Allerletzte): Ausgabe 31.01.2017, Abgabe 06.02.2017

Aufgabe 1: Joins in MapReduce 1 (1 P.)

(a) Gegeben folgende Map-Funktion:

```
map(egal, Tuple t) {
   //SB bezeichne Menge der distinct Attributwerte aus Spalte S.B
   if (t aus T) {
      if  (t.A in SB) {
         emit(t.A, t)
      }
   }
}
```

Beschreiben Sie, welche Art von Join hier berechnet wird (wenn überhaupt). Schreiben Sie die Map Funktion so um, dass ein regulärer Equi-Join $R \bowtie_{R.A=S.B} S$ berechnet wird. Geben Sie dafür die passende Reduce-Funktion an. Beschreiben Sie ferner was passiert, wenn die Menge SB durch eine komprimierte Struktur (z.B. einen sogenannten Bloomfilter) ersetzt wird, die genau weiß, wann ein Element in ihr enthalten ist, aber nicht 100% sicher sagen kann, ob ein Element nicht enthalten ist. Funktioniert die Join-Berechnung auch dann noch? Was sind die Konsequenzen?

(b) Folgende Tabellen sind gegeben: $R(A, B)$, $S(C, D)$ und $T(E, F)$. Wir möchten folgende Anfragen ausführen:

$$ans(x) \leftarrow R(z, w), S(u, v), T(y, x), w = u, u = x$$

Geben Sie Map- und Reduce-Funktionen für die Berechnung via Reduce-Side-Join an. Sie können dabei eine mögliche Joinreihenfolge wählen.

(c) Betrachten wir nun die Anfrage:

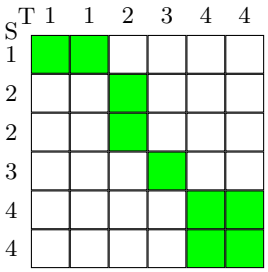
$$ans(x) \leftarrow R(z, w), S(u, v), T(y, x), w = u, v = y$$

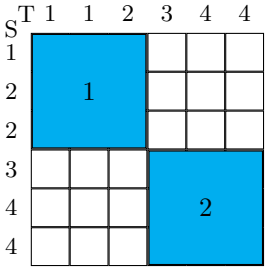
Können Sie diesen Join ebenfalls in nur einem Durchlauf von MapReduce berechnen? Falls ja, wie? Falls nicht, geben Sie eine Kaskade von MapReduce-Jobs an, die den Join berechnen. Mögliche Eingaben für Job Nr. i ist die Ausgabe vorheriger Jobs und die Basisrelationen.

Aufgabe 2: Joins in Map Reduce 2 (1 P.)

Gegeben folgende Join-Matrix (links). Berechnen Sie max-reducer-input für die “Standard”-Reduce-Side-Join Berechnung und für die rechts angegebene Zuordnung von Matrix-Zellen zu Reducern. Max-reducer-input beschreibt die Anzahl der Tupel, die beim am stärksten belasteten Reducer ankommen.

Gehen Sie bei erstgenannter Berechnung von zwei Reducern aus, die anhand key *mod* 2 zugewiesen werden.





Aufgabe 3: Data Layouts (1 P.)

Betrachten Sie folgende Anfragen aus dem TPC-H Benchmark.

```
select c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice, sum(
   l_quantity)
from customer, orders, lineitem
   where o_orderkey in (
    select l_orderkey
    from lineitem group by l_orderkey
    having sum(l_quantity) > [QUANTITY]
)
and c_custkey = o_custkey and o_orderkey = l_orderkey
group by c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
order by o_totalprice desc, o_orderdate;
```

o_orderkey: 4 bytes, o_custkey: 4 bytes, o_orderstatus: 4 bytes, o_totalprice: 4 bytes, o_orderdate: 8 bytes, o_orderpriority: 4 bytes, o_clerk: 4 bytes, o_shippingpriority: 4 bytes, o_comment: 200 bytes. Der explizite Schlüssel pro Column habe 4 bytes.

(i) Geben Sie jeweils die minimale Anzahl von Bytes an, die für Row- bzw. Column-Store für die orders-Relation von der Festplatte gelesen werden müssen.

(ii) Geben Sie eine geeignete vertikale Partitionierung (ohne Redundanz) an und vergleichen Sie sie bzgl. zu lesenden Bytes mit Row- und Column-Stores.