B561 Assignment 7

1 Key-value Stores (MapReduce and Spark)

1. Consider the document "MapReduce and the New Software Stack" available in the module on MapReduce.[1] In that document, you can in Sections 2.3.3-2.3.7 find descriptions of algorithms to implement relational algebra operations in MapReduce. (In particular, look at the mapper and reducer functions for various RA operators.)

   In the following problems, you are asked to write basic MapReduce programs that implement these operators in PostgreSQL. In addition, you should add the code which permits the PostgreSQL simulation for the basic MapReduce programs. (Look in particular at the "Additional Lecture Notes: Key-Value-MapReduce.pdf" file for details on how to do this.)

   (a) Write, in PostgreSQL, a basic MapReduce program, i.e., a mapper function and a reducer function, as well as a 3-phases simulation that implements the projection $\pi_A(R)$ where $R(A, B)$ is a relation. You can assume that the domains of A and B are integer. (Recall that in a projection, duplicates are eliminated.)

   (b) Write, in PostgreSQL, a basic MapReduce program, i.e., a mapper function and a reducer function, as well as a 3-phases simulation that implements the set difference of two unary relations $R(A)$ and $S(A)$, i.e., the relation $R(A) - S(A)$. You can assume that the domain of A is integer.

   (c) Write, in PostgreSQL, a basic MapReduce program, i.e., a mapper function and a reducer function, as well as a 3-phases simulation that implements the natural join $R \bowtie S$ of two relations $R(A, B)$ and $S(B, C)$. You can assume that the domains of A, B, and C are integer.

   (d) Let $R(A)$, $S(A)$, and $T(A)$ be unary relations that store integers. Write, in PostgreSQL, a MapReduce program that implements the RA expression $(A - B) \cup (C - A)$. Also provide a simulation that evaluates this program.

[1] This is Chapter 2 in *Mining of Massive Datasets* by Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman.

2. Let $R(K, V)$ and $S(K, W)$ be two binary relations. Consider the cogroup operator `R.cogroup(S)` introduced in the lecture on Spark. You can assume that the domains of K, V, and W are integers.

   (a) Define a PostgreSQL view that represents `R.cogroup(S)`. Show that this view works.

   (b) Write a PostgreSQL function that uses this `cogroup` view to compute $R \bowtie S$.

   (c) Write a PostgreSQL function that use this `cogroup` view to compute $R \overline{\ltimes} S$. Recall that $R \overline{\ltimes} S = R - R \bowtie \pi_K(S)$.

3. Let A and B be two unary relations of integers. Consider the `cogroup` operator introduced in the lecture on Spark.

   (a) Write a PostgreSQL function that uses the cogroup operator to compute $A \cap B$.

   (b) Write PostgreSQL functions that uses the cogroup operator to compute $A - B$ and $B - A$.

2 Nested Relations and Semi-structured databases

4. Consider the lecture on Nested and Semi-structured Data models. In that lecture, we considered the `studentGrades` nested relation and we constructed it using a PostgreSQL query starting from the Enroll relation.

   (a) Write a PostgreSQL query that creates the nested relation `courseGrades`. The type of this relation is

   $$(\texttt{cno}, \texttt{gradeInfo}\{(\texttt{grade}, \texttt{courses}\{(\texttt{sid})\})\})$$

   This relation stores for each course, the grade information of the students enrolled in this course. In particular, for each course and for each grade, this relation stores in a set the students who obtained that grade in that course.

   (b) Starting from this nested relation `courseGrades`, write a PostgreSQL that creates the nested relation `studentGrades` which is as described in the lecture.

   (c) In the lecture, we defined the `jstudentGrades` semi-structured relation. Write a PostgreSQL query that creates a `jcourseGrades` semi-structured relation which stores JSON objects whose structure conforms with the structure of tuples as described for the `courseGrades` nested relation in question 4a.

(d) Repeat question 4b but now for the semi-structured relations `jcourseGrades` and the `jstudentGrades`. In other words, starting from this semi-structured relation `courseGrades`, write a PostgreSQL that creates the semi-structured relation `studentGrades`.

(e) In the lecture on Nested and Semi-structured data models, we considered the query "For each student who major in 'CS', list his or her sid and sname, along with the courses she is enrolled in. Furthermore, these courses should be grouped by the department in which they are enrolled." We formulated this query in the context of nested relations.

Write a PostgreSQL to solve this query, but this time for semi-structured relations that store only JSON objects.