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 MapReduce programs that implement some RA expressions in PostgreSQL. In addition, you need to add the code which permits the PostgreSQL simulations for these MapReduce programs. A crucial aspect of solving these problem is to develop an appropriate data representation for the input to these problems. Recall that, in MapReduce, the input is a **single** binary relation of (key,value) pairs. Take for example Problem 1c. To solve this problem, you need to find a representation that puts both the data from R and from S in the same binary key-value relation. And, for problem 1d, you need to have a representation that put all the data from R, S, and T in the same binary key-value relation.

   (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 $\pi_{A,B}(R)$ where $R(A, B, C)$ is a relation. You can assume that the domains of A, B, C 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 $R - S$. 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 semijoin $R \ltimes S$ 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 $R - (S \cup T)$. Also provide a simulation that evaluates this program.

2. 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 SQL query

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

```
SELECT r.A, array_agg(r.B), cardinality(array_agg(r.B))
FROM   R r
GROUP  BY (r.A)
HAVING COUNT(r.B) >= 2;
```

Here R is a relation with schema (A, B). You can assume that the domains of A and B are integers.

3. Let $R(K, V)$ and $S(K, W)$ be two binary key-value pair relations. Consider the cogroup transformation `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 represent the co-group transformation `R.cogroup(S)`. Show that this view works.

   (b) Write a PostgreSQL query that use this `cogroup` view to compute $R \ltimes S$.

   (c) Write a PostgreSQL query that uses this `cogroup` view to implement the SQL query

```
SELECT distinct r.K as rK, s.K as sK
FROM   R r, S s
WHERE  ARRAY(SELECT r1.V
             FROM   R r1
             WHERE  r1.K = r.K) <@ ARRAY(SELECT s1.W
                                         FROM   S s1
                                         WHERE  s1.K = s.K);
```

4. Let A and B be two unary relations of integers. Consider the `cogroup` transformation introduced in the lecture on Spark. Using an approach analogous to the one in Problem 3 solve the following problems:

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

   (b) Write a PostgreSQL query that uses the cogroup operator to compute the symmetric difference of A and B, i.e., the expression

$$(A - B) \cup (B - A).$$

2 Nested Relations and Semi-structured databases

5. 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

-The nested relational database: complex-object types
- the semi-structured database: JSON types

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

   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 5a.

   (d) Repeat question 5b 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 query to solve this query, but this time for semi-structured relations that store only JSON objects.

3 Graph Databases

6. Consider the database schema Student, Course, Buys, Major, and Cites that we have been using throughout the assignments.

   (a) Specify an Entity-Relationship Diagram that models this database schema.

   (b) Specify the node and relationship types of a Property Graph for this database schema. In addition, specify the properties, if any, associated with each such type.

7. Using the Property Graph model in Problem 6b, formulate the following queries in the Cypher query language:

   (a) Find the types of the relationships associated with Student nodes.

   (b) Find each student (node) whose name is 'John' and who bought a book whose price is at least $50.

   (c) Find each student (node) who bought a book that cites a book whose price is at least $50.

   (d) Find each book (node) that is cited directly or indirectly (i.e., recursively) by a book that cost more that $50.

   (e) Find for each book node, that node along with the number of students who major in both CS and in Math and who bought that book.