Introduction to Databases, Spring 2019

Homework #4 (50 Pts, June 19, 2019)

Student ID _____

Name _____

(1) [20 pts] Check whether the following schedule is serializable. For a serializable schedule, determine the equivalent serial schedule(s). Note that $r_1(X)$ indicates *read* operation on X for T_1. Explain why (**5pts each**).

(a) $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X);$

Answer:

(b) $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X);$

Answer:

(c) $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X);$

Answer:

(d) $r_3(X); r_2(X); r_1(X); w_3(X); w_1(X);$

Answer:

(2) [10 pts] Consider the three transactions T1, T2, and T3. Draw the serializibility (precedence) graphs for two schedules S1 and S2, and check whether each schedule is serializable or not. If the schedule is serializable, write down the equivalent serial schedule(s) (**5pts each**).

$T1: r_1(x); r_1(z); w_1(x);$

$T2: r_2(z); r_2(y); w_2(z); w_2(y);$

$T3: r_3(x); r_3(y); w_3(y);$

(a) $S1: r_1(x); r_2(z); r_1(z); r_3(x); r_3(y); w_1(x); w_3(y); r_2(y); w_2(z); w_2(y);$

Answer:

(b) $S2: r_1(x); r_2(z); r_3(x); r_1(z); r_2(y); r_3(y); w_1(x); w_2(z); w_3(y); w_2(y);$

Answer:

(3) [20 pts] Implement the following SQL statement using MapReduce framework. Assume that there are two tables: emp(empno, ename, deptno) and dept(deptno, dname, location) as follows.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	1987-04-19	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987-05-23	1100.00	NULL	20
7900	JAMES	CLERK	7698	1981-12-03	950.00	NULL	30
7902	FORD	ANALYST	7566	1981-12-03	3000.00	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Implement and explain your map and reduce function with execution snapshots.

```
SELECT d.dname, count(*)
FROM emp e, dept d
WHERE e.deptno = d.deptno
GROUP BY d.dname
```

```
pengsasm@pengsasm-VirtualBox:~/join$ cat output/*
ACCOUNTING      3
RESEARCH        5
SALES   6
```

Map function 1(emp table):

각 레코드의 값을 읽어서 다음과 같은 key-value 값으로 mapping했다.

(레코드의 deptno, “emp,” + “레코드의 empno”)

Map function 2(dept table):

마찬가지로 각 레코드의 값을 읽어서 다음과 같은 key-value 값으로 mapping했다.

(레코드의 deptno, “dept,” + “레코드의 dname”)

Reduce function:

Key에 해당하는 모든 value들에 대해, 만약 그 value가 emp라는 단어를 포함하고 있으면, empno를 저장하는 array에 그 값을 넣는다. 반면 그 value가 dept라는 단어를 포함하고 있으면 dname을 저장하는 string에 그 값을 넣는다. (deptno 별로 dname은 하나만 정해지기 때문에 가능)

위 과정이 끝나면 empno에 얼마나 많은 원소가 있는지, 즉 employee가 몇 명인지 센다. 만약 empno에 원소가 없거나 dname의 값이 비어있다면, join된 레코드가 없다는 뜻이므로 아무것도 쓰지 않는다. 둘 다 값이 존재하는 경우에만 context에 다음 값을 썼다.

(dname, employee의 수)

Code

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
```

```
public class join{

	public static class EmpMapper extends Mapper <Object, Text, Text, Text> {
		public void map(Object key, Text value, Context context)
				throws IOException, InterruptedException {
			String record = value.toString();	// Reads the record
			String[] parts = record.split(","); // Split the input.
			context.write(new Text(parts[7]), new Text("emp," + parts[0]));
			// key : deptno, value: emp(sign that it is emp) + empno
		}
	}

	public static class DeptMapper extends Mapper <Object, Text, Text, Text> {
		public void map(Object key, Text value, Context context)
				throws IOException, InterruptedException {
			String record = value.toString();	// Reads the record
			String[] parts = record.split(","); // Split the input.
			context.write(new Text(parts[0]), new Text("dept," + parts[1]));
			// key : deptno, value: dept(sign that it is dept) + dept_name
		}
	}

	public static class ReduceJoinReducer extends Reducer <Text, Text, Text, IntWritable> {
		private IntWritable val = new IntWritable();
		public void reduce(Text key, Iterable<Text> values, Context context)
			throws IOException, InterruptedException {
```

```
                ArrayList<String> empno = new ArrayList<String>();
                // String that stores the department name of each department number
                String dept_name = "";

                // Records the emp, dept value that corresponds to dept_no
                for (Text t : values) {
                        String parts[] = t.toString().split(",");
                        if (parts[0].equals("emp")) empno.add(parts[1]);

                        // Note that there is one department name for each department number
                        else if (parts[0].equals("dept"))       dept_name = parts[1];
                }

                // Counts the number of employee
                int sum = 0;
                for (String s : empno){
                        sum += 1;
                }
                val.set(sum);
                // If there is no department name or employee, that it is not recorded
                if(sum != 0) context.write(new Text(dept_name), val);
        }
}

public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "Reduce-side join");
        job.setJarByClass(join.class);
```

```
        job.setReducerClass(ReduceJoinReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        MultipleInputs.addInputPath(job, new Path(args[0]),
                TextInputFormat.class, EmpMapper.class);
        MultipleInputs.addInputPath(job, new Path(args[1]),
                TextInputFormat.class, DeptMapper.class);

        Path outputPath = new Path(args[2]);
        FileOutputFormat.setOutputPath(job, outputPath);
        outputPath.getFileSystem(conf).delete(outputPath);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```