CSC 244 Programming Assignment-2

NATURAL JOIN

RAMYA VELLORE RAMESH

SAC ID:219123243

**a)1.The description of your algorithm**.

We have Relation R(A,B,Y) & S(Y,Z,X) where secondary index is assigned based on the Y value.Relation S is sorted based on Y value.

Load the blocks of R and S into the Memory.

Load the index of R as key in the hash map i.e the Memory.

For each tuples in the block of Relation S

If secondary index of R matches with Y value of the tuples of Relation S

Then we join the tuples of R and S

We output the joined tuples into the output buffer

WriteToOutputBuffer

If size(outputbuffer )< 3

put the result to the outputbuffer

Else

Move the current buffer value to disk

put the result to outputbuffer

**2.The description of the simulation of the memory and disk storage**

Assuming Relation R and S are in disk,S has 6 tuples which is more than the memory(size=5),R has 8 tuples which is more than the size of the memory(size=5) .Relation S and R are assumed to be in two-dimensional array.Relation R and S contains 3 Attributes for each tuple.(B(R) > M and B(S)>M)

* 2-Dimensional arrays are used for both the relations R and S for disk storage.
* ArrayList is used as memory to store the block values of Relation R and S from the disk.
* HashMap is used to store the tuples of Relation R as a key-value pair.The secondary index is stored as key and the tuple of the block is stored as the value.The secondary index size of R which is stored in hash map as key is less than M.
* If the primary key of S i.e the Y value of each tuple in Relation S is equal to secondary index of R which is the key in hashmap ,then we join the tuples of the two relations R and S
* An ArrayList is used to store the Natural Join of tuples of Relation R and S to the output buffer.
* When the output buffer is full, the buffer values are copied to the diskoutput .ArrayList is used as disk output.

**3.Example used to test the program**

R={{2,4,1},{3,9,5},{6,2,5},{2,5,9},{9,1,2},{13,2,4},{7,5,8},{1,5,6}};

S= {{1,3,2},{2,4,6},{2,6,0},{3,2,4},{4,8,6},{5,2,3}};

**4. The input and output of your program using this example. Input can be either hardcoded or from the command line.**

**Input is hardcoded.**

**Output:**

Buffer contents after writing to Outputbuffer:

[2, 4, 1, 3, 2]

Buffer contents after writing to Outputbuffer:

[2, 4, 1, 3, 2]

[9, 1, 2, 4, 6]

Buffer contents after writing to Outputbuffer:

[2, 4, 1, 3, 2]

[9, 1, 2, 4, 6]

[9, 1, 2, 6, 0]

Buffer is full writing the buffer contents to disk:

[2, 4, 1, 3, 2]

[9, 1, 2, 4, 6]

[9, 1, 2, 6, 0]

Buffer contents after writing to Outputbuffer:

[13, 2, 4, 8, 6]

[6, 2, 5, 2, 3]

The Natural Join of R and S

[[2, 4, 1, 3, 2], [9, 1, 2, 4, 6], [9, 1, 2, 6, 0], [13, 2, 4, 8, 6], [6, 2, 5, 2,3]]

**b)How to Run:**

1. Running In IDE

a> Create a Empty Java Project

b> Add a new class with the name NaturalJoin

c> Copy the contents of the NaturalJoin.java from uploaded file to the class

d> Run as Java Application

2. How to run java source file on Athena or Command Prompt

a.Using javac complier by running the command ,compile the java file(NaturalJoin.java)

javac NaturalJoin.java

b.Execute the compiled java code by executing the command

java NaturalJoin

c.Output is displayed on the output screen.

**C) Source code. Proper comments/description should be provided**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

public class NaturalJoin {

static int[] append(int[] arr, int element) {

final int N = arr.length;

arr = Arrays.copyOf(arr, N + 1);

arr[N] = element;

return arr;

}

public static ArrayList<int[]> JoinIndex(ArrayList<int[]> newS, ArrayList<int[]> newR)

{

HashMap<Integer, int[]> map1= new HashMap<Integer, int[]>();

ArrayList<int []> bufferOutput = new ArrayList<int[]>(3); //Output buffer of size 3

ArrayList<int[]> diskOutput = new ArrayList<int[]>(15); //Output disk of size 15

// Storing the tuples of Relation R into hashmap for finding the Natural Join

for(int[] i : newR)

{ /\*assigning the secondary index on the Y attribute and putting

secondary index as the key in the hashmap for each of the tuples of the block \*/

map1.put(i[2],i);

}

for(int [] k : newS)

{

/\*checking if the y attribute value(Primary Key) of S matches with secondary index of R which is in the map as key ,

if yes then we do a natural join\*/

if(map1.containsKey(k[0]))

{

int [] Join = new int[10];

Join = map1.get(k[0]);

Join = append(Join,k[1]);

Join = append(Join,k[2]);

if(bufferOutput.size()<3)

{

bufferOutput.add(Join);

System.out.println("Buffer contents after writing to Outputbuffer:");

for(int[] m:bufferOutput)

{

System.out.println(Arrays.toString(m));

}

}

else

{

System.out.println("Buffer is full writing the buffer contents to disk:");

diskOutput.addAll(bufferOutput);

for(int[] n:diskOutput)

{

System.out.println(Arrays.toString(n));

}

bufferOutput = new ArrayList<int[]>(3); //clear the buffer after writing it to the disk by reintializing it

bufferOutput.add(Join);//after clearing the full buffer we add the next common elements in queue to be written into the buffer

}

}

}

diskOutput.addAll(bufferOutput);

return diskOutput;

}

public static void main(String[] args)

{ //No of Blocks=1

// Block values of Relation R in disk

int[][] R={{2,4,1},{3,9,5},{6,2,5},{2,5,9},{9,1,2},{13,2,4},{7,5,8},{1,5,6}};

// Block values of Relation S in disk

//Y values of the tuples are sorted in the Relation S

int[][] S= {{1,3,2},{2,4,6},{2,6,0},{3,2,4},{4,8,6},{5,2,3}};

ArrayList<int[]> newS= new ArrayList<int[]>(5);

ArrayList<int[]> newR= new ArrayList<int[]>(5);

//Loading block values of Relation R and S into Memory

for(int i=0;i<S.length;i++)

{

newS.add(S[i]);

}

for(int i=0;i<R.length;i++)

{

newR.add(R[i]);

}

//Function call to find the Natural Join

ArrayList<int[]> result= JoinIndex(newS, newR);

int[][] NaturalJoin = new int[result.size()][];

for (int i = 0; i < result.size(); i++) {

int[] row =result.get(i);

NaturalJoin[i] = row;

}

System.out.println("The Natural Join of R and S");

System.out.println(Arrays.deepToString(NaturalJoin));

}

}