# Spring 2017 CSC 244 Homework assignment 2

Description

Karan Mitra
Sac ID : 219032620

## 1. Algorithm -

The algorithm used to join these two given tuples will be a **index-based simple sort join** using a sorted index. There are a few variations of the simple sort join that need to be applied here since the relation S and the index are sorted on Y. This algorithm is a two-pass algorithm which is required because of the given condition that neither the blocks of R or S can all completely fit into memory. In this algorithm,

- We first load the sorted secondary index of R whose size in much smaller than S and can thus fit into the memory without taking up significant space.
- We then load the number of blocks of S from disk that can fit into M-1 blocks of the remaining space in the memory.
- Once the blocks have been loaded, we loop through the Y attribute in the R index, comparing them to the Y attribute of the tuples of S.
- If a match is found, the corresponding block required to extract the tuple of R is brought in from disk using the pointer of that tuple in the index. This block is stored in the single empty block in the memory. At this point, the memory is completely utilized.
- Once this block has been brought in, the appropriate and required tuple of R is extracted from this block, we join the remaining tuples of R and S to form a joined tuple.
- This joined tuple is fed to the output buffer. If the output buffer is full, it is emptied out to the disk.
- Once all the tuples in memory of S are done being checked, the next required tuples of S are brought in till the time all are done.
- The output buffer is now flushed to output any joined tuples which may have remained in the buffer.
- The final output containing all the tuples is now present in the disk.

## 2. Description of Simulation of memory and disk storage -

Disk storage has been simulated by using the disk storage of the running machine. Text files, containing the required tuples of R and S are placed at the location of the code. These files are read and stored in an in-memory class, each object of which represents a tuple in the corresponding relation. There are also classes which represent the blocks of these tuples. They simply contain arrays of the above mentioned tuples to simulate blocked loading of tuples. I have assumed 3 tuples per block for both R and S.

Memory has been simulated by a java class. It has a single instance which simulates the required memory structure. The object of this memory structure is capable of holding the index (represented by a java util Map), one block of R and 2 blocks of S. This size restriction ensures the given memory condition of B(R) >M and B(S) >M since all block of R or S will not fit into memory.

Index has been simulated as a java util Map which is built and is present in the disk simulation, at the same level where the relations R and S lie after their respective files have been parsed. This index is later brought into memory before the tuple matching begins.

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

**R (A,B,Y)**

1 R1 456
2 R2 345
3 R3 234
4 R4 123
5 R5 789
6 R6 567
7 R7 901
8 R8 123
9 R9 910
10 R10 902
11 R11 091
12 R12 798

**S(Y,Z,X)**

123 S1 1.1
132 S5 5.5
234 S2 2.2
234 S4 10.1
234 S4 11.1
243 S6 6.6
345 S3 3.3
345 S7 12.1
456 S4 4.4
456 S8 8.8
798 S7 7.7

**Output**

| Y | A | B | Z | X |
|---|---|---|---|---|
| 123 | 8 | R8 | S1 | 1.1 |
| 234 | 3 | R3 | S2 | 2.2 |
| 234 | 3 | R3 | S4 | 10.1 |
| 234 | 3 | R3 | S4 | 11.1 |
| 345 | 2 | R2 | S3 | 3.3 |
| 345 | 2 | R2 | S7 | 12.1 |
| 456 | 1 | R1 | S4 | 4.4 |
| 456 | 1 | R1 | S8 | 8.8 |
| 798 | 12 | R12 | S7 | 7.7 |