**Concordia University**

**Department of Computer Science and Software Engineering**

**COMP 6521 Advanced Database Technology and Application**

**Winter 2013**

**Project 3**

***Prepared by Group No.8:***

***Dan Zhan 6486592***

***Lin Wang 6388663***

***Date: April 4, 2013***

# Introduction

Project 3 is to use implementation in project 1 and develop an efficient sort-based procedure to difference tables in the last step of phase 2 of the 2PMMS technique.

# Implementation 1.create sorted sublists of file table1 on table2 respectively, the size of sublists is constrained by the available memory size. Since the size of each tuple in each file is almost equal to 100bytes, in situation of memory 5M, each time 20,000 tuples can be read into memory, so the size of sublist is 2M with considering the remaining memory is used by system variables and computing. Similarly in situation of memory 10M, each time 25,000 tuples can be read into memory, and the size of sublist is 2.5M; in situation of memory 20M, each time 50,000 tuples can be read into memory, and the size of sublist is 5M.

In this phase, we use quicksort algorithm to sort tuples of each sublist on EmpID since the quicksort is executed in place and won’t occupy extra memory space, the complexity of quicksort is O(nlog n), we can trade off between memory space and execution time.

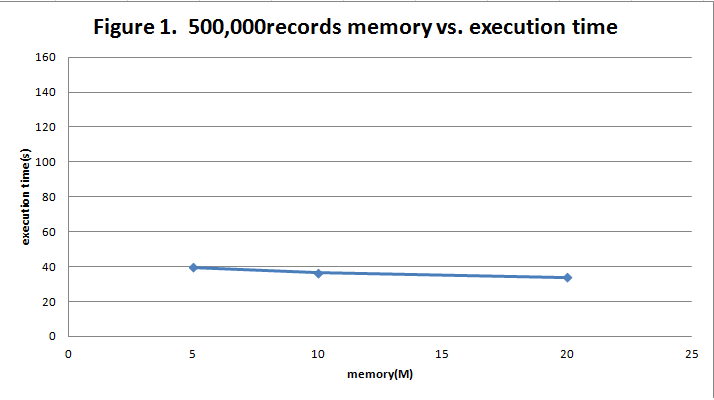
2.perform sort based bag difference of file Employee and file Table1 on common attribute Table2.

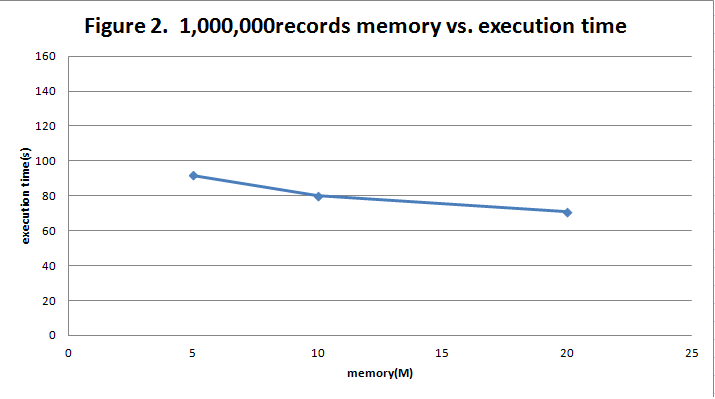
Bring the first tuple of each sublist of file Table1 into one memory buffer; Bring the first tuple of each sublist of file Table2 into the other memory buffer. And make each tulpe a count to indicate the duplicate count; Each time reading some new tuple, compare with the exist tuple, if found, count plus, if not create a new record until read M distinct tuple in record.

3.Repeatedly find the least value of Table1 among the first available tuples of all sublists and decrease the count by 1. Identify the tuple of both files , if the value is the same, skip by decrease there counter by 1 and choose the small value to output by times its counter indicated. When the count equals to zero read a new tuple in following the previous rule.

# Report the total execution time(sort+bag difference)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5MB(each time read 20,000tuples) | 10MB(each time read 25,000tuples) | 20MB (each time read 50,000tuples) |
| 500,000tuples | 39.456 s | 36.291 s | 33.8 s |
| 1,000,000tuples | 91.772 s | 79.989 s | 70.782 s |





number of disk I/Os

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5MB | 10MB | 20MB |
| 500,000tuples | 1,500,100 times | 1,500,080times | 1,500,040times |
| 1,000,000tuples | 3,000,200times | 3,000,160times | 3,000,080times |