Practicum 3: Sort-Merge Join

Deadline: 23:59 December 23 (Friday), 2005
This is a group assignment, and at most 2 students per group are allowed.

Cheating Policy: If you are caught cheating, your grade is 0.
Late Policy: You may hand in your late assignment before 23:59 on Saturday (12/24/2003) for 80% of original grade, or before 23:59 on Sunday (12/25/2003) for 70%. We will not accept any assignment submissions after Monday (12/26/2003).

Introduction

In this assignment, you will implement the sort-merge join algorithm.

Available Documentation

You should begin by reading the chapter on *Implementation of Relational Operations*, in particular, the section on *Sort-Merge Join*.

What You Have to Implement

```
class sortMerge
{
    public:
    sortMerge(
        char        *filename1,    // Name of heapfile for relation R.
        int         len_in1,       // # of columns in R.
        AttrType    in1[],         // Array containing field types of R.
        short       t1_str_sizes[], // Array containing size of columns in R.
        int         join_col_in1,  // The join column number of R.

        char        *filename2,    // Name of heapfile for relation S
        int         len_in2,       // # of columns in S.
        AttrType    in2[],         // Array containing field types of S.
```

```
        short           t2_str_sizes[],  // Array containing size of columns in S.
        int             join_col_in2,    // The join column number of S.

        char*           filename3,       // Name of heapfile for merged results
        int             amt_of_mem,      // Number of pages available for sorting
        TupleOrder      order,           // Sorting order: Ascending or Descending
        Status&         s                // Status of constructor
    );

    ~sortMerge();
}
```

The sortMerge constructor joins two relations R and S, represented by the heapfiles *filename1* and *filename2*, respectively, using the sort-merge join algorithms. Note that the columns for relation R (S) are numbered from 0 to *len_in1 - 1 (len_in2 - 1)*. You are to concatenate each matching pair of records and write it into the heapfile *filename3*. The error layer for the sortMerge class in JOINS, that is, you should use MINIBASE_CHAIN_ERROR(JOINS, status) to append an error information to the global error queue.

You will need to use the following classes which are given: Sort, HeapFile, and Scan. You will call the Sort constructor to sort the input heapfiles (which means your primary responsibility will be to implement the merging phase of the algorithm). To compare the join columns of two tuples, you will call the function **tupleCmp** (declared in sort.h). Once a scan is opened on a heapfile, the scan cursor can be positioned to any record within the heapfile calling the Scan method **position** with an RID argument. The next call to the Scan method *getNext* will proceed from the new cursor position.

Compiling Your Code and Running the Tests

Please copy all the files from web site into your own local directory. The files are:
- *Makefile*: A sample Makefile for you to compile your project. Set up any dependencies (as needed) by editing this file.
- *sortMerge.h*: Specifications for the class sortMerge. You have to implement these specifications as part of the assignment.
- *SMJTester.C*: sort-merge test driver program.

You will also find in the project directory the implementation of the external sort algorithm in the files *sort.C and sort.h* and the interface to the Scan and HeapFile classes in the directory *include*.

How to hand-in

Email the files "sortMerge.h" , "sortMerge.C" and report to the khchang@csie.org before the deadline with the title "DBMS Practicum 03", and submit two students' names and IDs in the your email body. The report should describe the detailed process of your sortMerge constructor.