CSE 510 - Database Management System Implementation
Spring 2018
Phase III
Due Date: Midnight, April 29th

1 Goal

The version of the MiniBase I have distributed to you implements various modules of a relational database management system. Our goal this semester is to use these modules of MiniBase as building blocks for implementing a *column-oriented DBMS*.

2 Project Description

In this phase of the project, we will focus on the `Iterator` class, which provides methods for duplicate elimination, file scan, index scan, nested loop joins, sort merge join, and sorting. In the previous phase, you have already worked on file scan and index scan on columnar files. In this phase, we will generalize these. The following is a list of tasks that you need to perform for this final phase of the project (note that getting these working may involve other changes to various modules not described below):

- You have already implemented a `ColumnarFileScan` class; no change to that class is necessary.
- You have already implemented a `ColumnIndexScan` class that help access to a *single* column using an index. Generalize that class to a `ColumnarIndexScan` class, with constructor

  ```
  ColumnarIndexScan(
       java.lang.String relName,
       int[] fldNum,
       IndexType[] index,
       java.lang.String[] indName,
       AttrType[] types,
       short[] str_sizes,
       int noInFlds,
       int noOutFlds,
       FldSpec[] outFlds,
       CondExpr[] selects,
       boolean indexOnly)
  ```

  which scans the given columnar file using the given one or more index files and returns matching tuples.

Note that, in the previous phase, when indexType is a bitmap, it was sufficient to consider only equality searches. In this phase, generalize this to also inequality searches.

- Implement a `ColumnarNestedLoopJoins` class, with constructor

```
ColumnarNestedLoopsJoins(
     AttrType[] in1,
     int len_in1,
     short[] t1_str_sizes,
     AttrType[] in2,
     int len_in2,
     short[] t2_str_sizes,
     int amt_of_mem,
     Iterator am1,
     java.lang.String columnarFileName,
     CondExpr[] outFilter,
     CondExpr[] rightFilter,
     FldSpec[] proj_list,
     int n_out_flds)
```

  which joins two columnar files.

- Implement a `ColumnarBitmapEquiJoins` class, with constructor

```
ColumnarBitmapEquiJoins(
     AttrType[] in1,
     int len_in1,
     short[] t1_str_sizes,
     AttrType[] in2,
     int len_in2,
     short[] t2_str_sizes,
     int amt_of_mem,
     java.lang.String leftColumnarFileName,
     int leftJoinField;
     java.lang.String rightColumnarFileName,
     int rightJoinField;
     FldSpec[] proj_list,
     int n_out_flds)
```

  which joins two columnar files using bitmap indexes on the given fields.

- Implement a `ColumnarSort` class, with constructor

```
ColumnarSort(AttrType[] in,
     short len_in,
     short[] str_sizes,
     java.lang.String ColumnarFileName,
     int sort_fld,
     TupleOrder sort_order,
     int sort_fld_len,
     int n_pages)
```

which sorts the given columnar file based on the given conditions.

- **If you are a large group:** Implement a `ColumnarDuplElim` class, with constructor

```
ColumnarDupleElim(AttrType[] in,
     short len_in,
     short[] str_sizes,
     java.lang.String ColumnarFileName,
     int amt_of_mem,
     boolean inp_sorted)
```

which eliminates duplicate tuples from the input columnar file.

- Implement a test-driver that lets the user (a) create a new columnar database and populate it with columnar files or (b) open an existing columnar database, and (c) perform any of the above operations.

  Note that in the precious phase, the value constraint was of the form {`COLUMNNAME OPERATOR VALUE`}. In this case, you will need to generalize the search and join condition to take in multiple columns and more complex logical operations.

  For each case, the test driver should present the user the output of the operation and also print out the number of read/write page accesses.

3 Deliverables

You will be provided with a sample data set. You have to return the following before the deadline:

- Your source code properly **commented**, `tar`ed and `zip`ed.
- The output of your program with the provided test data and the driver.
- A report. The report should contain a short description of the operators.
- The report should clearly state *who did what*. This will be taken very seriously! So, be honest. Be prepared to explain on demand (not only your part) but the entire set of modifications. See the report specifications.
- A confidential document (individually submitted by each group member) which rates group members' contributions out of 100 (100 best; 0 worst). Please provide a brief explanation for each group member.