Writeup for lab4

Yuwei Wu
ACM Class, Zhiyuan College, SJTU

CS392 Database Management System

Prof. **Feifei Li**

Due Date: June 2, 2019

Submit Date: June 1, 2019

Design decisions about lab4

- When implementing the *IntHistogram* class, I just keep a int histogram array to store the record number of added value. The histogram is simply the equi-width histogram. And every added value is store under the index according to the min and max value. Selectivity is estimated according to the given formula.
- When implementing the *TableStats* class, I just keep two histogram array: one for int and one for string. And two scans through all the tuples give the final histogram. The first scan is to determine the min and max value for the histogram under specific field and the second scan is to add field value to the histogram.
- For the *JoinOptimizer* part, I just use the formula $scancost(t1) + ntups(t1) \times scancost(t2) + ntups(t1) \times ntups(t2)$ to estimate JoinCost as I just implemented a nested-loops join and I just follow the pseudocode code to implement the orderJoin method.
- The other parts are quite straight-forward.

API changes

- No API changes.

Missing or incomplete elements

- No missing parts.

Time spent and difficulties

- I spent about three days working on lab4.
- I found meet problems in passing queryTest as the time limit was exceeded at first. Finally, I found it was due to the inefficient iterating method I done in HeapFile iterator method. I then replaced the while loop with a single line that do not need any iterations.

- I found when the query statements have no "And", i.e. $this.joins.size() = 0$ in this case. This will cause troubles in orderJoin as no best plan will be found, resulting in a null return value. To solve this problem, I just return the *this.joins* in orderJoin method when $this.joins.size() = 0$.