Report of PA3

Yuan Wei `U52001624`

Filter and Join

Use nested loops for both `Join` and `HashEquiJoin`. For `Join`, nested loop was implemented to go through all next child tuple pairs. For `HashEquiJoin`, it's nearly the same, but use a hash map to store all tuples in `child1` (under related fields).

it took approximately 4 hours.

Aggregates

For `IntegerAggregator`, I create a data structure to store `group<value, count>` info, and use a map to store `<field, group>` pairs. So if not grouping, the "group" will be indexed by a reserved field, which we can use it to generate result directly. For `StringAggregator`, just use the hash map to store group's count since there is only 1 COUNT operation.

It took about 3 hours. I was blocked at the beginning when trying to deal with different operations, but finally overcame by the design of the `group` data structure.

HeapFile Mutability

For `HeapPage.deleteTuple`, once a tuple is deleted, mark the slot as unused.

In `HeapPage`, i iterate the slot and find the first empty slot to insert

For `HeapFile`, just followed doc.

This section took me about numerous days around 14 hours ?

Insertion & Deletion

This part looks about 3 hours.

the tuple deletion and and the insertion must be excuted in bufferpoll, and the steps are first check if the page is in the table., and then delete the page using yhe delete function implemented in HeapPage.java. the insertion function works as the same logic

The implementations of `Insert` and `Delete` are quite straightforward. Most time that I spent was debugging `sys - Insert` test. It keeps saying tuple desc doesn't match.