Lab 2

Query Runtimes:

Query 1: The query runtime for Query 1 is 0.28 seconds.

Query 2: The Query resulted in an error of “java.io.FileNotFoundException: /Users/aaronchan/IdeaProjects/simple-db-antzoom1/authors.dat (Too many open files).” I think it might have to do something with my setup so I really don't know why exactly this is occurring. I went to office hours and talked on Edstem but cannot figure out this output correctly even though all tests passed except one.

Query 3: The same occurred in Query 3.

Summary of Lab: This lab was mainly about the basic components of a database used to operate a simple set of queries. We did things such as Filter, Join, String, and Integer Aggregates. We also were able to insert and delete tuples. The predicate and join predicates were mainly used to compare fields with the specified conditions. The Filter class was mainly to (as the name suggests) filter out tuples and iterate through them given a child iterator. Then we filter out the tuples based on the given predicate. The Join class merged two tuples together and iterate through tuples in child iterators based on their predicates. The Integer and String Aggregators added aggregate query functions over the specified integers and strings. It can iterate through the tuples and group by a common field key or index and store it on a concurrency Map. The Heap Page and File added methods for putting or removing tuples from the tables. The Insert and Delete class gave more operators for inserting and removing tuples from tables for HeapPage, HeapFile, and BufferPool. Lastly The Buffer Pool and its eviction policy. I used the LRU eviction policy and made and reversed an ArrayList of Pageids. Then I evicted the first page encountered in the iteration of the reversed ArrayList.

Design Choices: I mainly used Hash join. I also choose the LRU Eviction policy mainly because it seemed better performance-wise in general. As the specs and Edstem suggest, I kept using a Concurrency HashMap in my implementations. I was already familiar with doing Map stuff and sorting out id, keys, and values. As for the IntegerAggregator, I used 2 concurrency maps to check the values and counts for the implementation. I also solved cases using the operators to see which computation I needed to do.

Example of a Unit Test: One example of a unit test that would improve the set of unit tests provided for the lab would be a test that checks if the deletion of an existing tuple is done correctly in both the BufferPool and HeapFile. While there is a system test for deletion, it would be nice to also have a unit test for deletion.

Change to the API: I made no changes to the API.

Missing or Incomplete elements of my code: I found out I'm not supposed to return null on concurrency HashMaps in Integer and String Aggregators. Because of that, I didn't pass one test but other than that everything was completed.

Additional Feedback: I have no additional feedback.