Name: Aaron Chan

SSID: antzoom1

<u>Summary of SimpleDB:</u>

Lab 1 is a program that implements the core components of the SimpleDB architecture. Lab1 allowed SimpleDB to pass datafiles through the SQL server on the simplest scale. The process of these components can be ordered in the following: The query is initialized. Then the OpIterator is called which then calls the HeapPage for the desired pages. Then, the HeaPage will call the HeapFile to be loaded in the desired files. Then the HeapFile will reference into the BufferPool to find the necessary data. If the query data isn't found in the BufferPool then the BufferPool reads from the disk. In the current lab, I simply called an DbException error if the BufferPool doesn't have enough space. However, normally the BufferPool will incorporate some sort of eviction policy to open up space. One type of eviction that I recall from lecture is evicting the least used page. Then, the BufferPool will store the data obtained from the disk after evicting and return the query. The process keeps repeating until all operators are completed and closed.

<u>Components of the Implementation:</u>

- Tuple and TupleDesc: These classes allowed The DatabasePage and DatabaseFile classes to correctly know the field of each given schema as well as tuple. This uses the field for figuring out the page headers and page sizes.
- Catalog: This class retains the mapping to the databaseTable classes and their information.
- BufferPool: This class is a memory manager that gathers pages from disks to be used for queries.
- HeapPageId: This class keeps track of the page ids in the pages.

- RecordId: This class keeps track of tuple information in the pages.
- HeapPage: This class handles the information about files.
- HeapFile: This class manages the HeapPages and filters them into a single schema/table.
- SeqScan: This class performs an sequential/linear scan through the entire table/schema

Design Decisions:

I mainly used the specification to incorporate most of the designs that I used. As for the Catalog class, I noticed that the initial data incorporated a library for ConcurrentHashMaps. After I looked into it more in the javadocs, I decided to use that because I feel like it'll be easier to implement data and future classes as I can change data through the computations. As for the HeapFile class, I used the DbFileIterator interface to help with my iterator for HeapFile. This was because the javadocs and spec said not to load entire tables and only to load slots with Data in them.

Unit Test: One Unit Test that could be added to improve the set of unit tests given in this lab would be to check if the BufferPool is working as intended. We could have the test to assert true if the behavior of the Page that is returned from the BufferPool.getPage() method is the intended page. That way, we would have more verification that the BufferPool class was implemented properly besides only knowing if it works based on the implementation of the HeapFile class.

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

Missing or Incomplete Elements of Code: I did not have any missing or incomplete code in my implementation.

Additional Feedback: I have no additional feedback currently.