CSE 444 - Lab 1

In this Lab, we implemented several different components of SimpleDB in order to get one of the operators - SeqScan to properly return the tuples it should. SimpleDB comprises of several different components:

1. TupleDesc
2. Tuple
3. BufferPool
4. HeapPage
5. HeapFile
6. SeqScan
7. IDs for the different components of SimpleDB like Records and Pages.

In SimpleDB, a single table is saved on a single file and the single file has a single HeapFile associated with it. Essentially, every single table has exactly one heap file associated with it. A Heap File can consist of several different pages filled with data. Each of these pages is a HeapPage. HeapPage acts as a single page of tuples that stores a certain number of bytes of data in it. Each of the HeapPages store a collection of tuples stored in the order that they're stored in disk. This Data is basically in the form of Tuples (which is a single row of data in the table) . Each of the tuples have an associated TupleDesc which determines the format of the tuples (the schema of the table). The BufferPool is basically a cache that stores pages that we've already seen in order to get quicker access to tuples. The BufferPool however cannot read data directly from disk. The BufferPool depends on the HeapFile to read data from disk and then stores each page for quicker access. Presently the BufferPool can only hold a specific number of pages (and will throw an exception if the number of pages is exceeded).

When the SeqScan operator is called, it is passed a tableId. The SeqScan can hence pull out the associated file for that table and call it's iterator. As the SeqScan has to return another tuple, it simply asks the associated HeapFile to return another tuple. The HeapFile internally has an iterator that can keep track of the different HeapPages that it needs to read consecutively to get all the tuples in the table. The HeapFile stores an instance of the iterator on the heapPage it is looking at right now. When the HeapPage doesn't have any more tuples (hasNext()) returns false, the HeapFile moves onto the next page. Internally in the HeapPage, the heap page keeps track of the valid tuples in the page (with it's header). The heap page keeps returning tuples until it runs out of tuples in the page. Hence, the SeqScan operator gets all the tuples for a specific file as the heap file associated with the table keeps returning tuples one by one.

Design Decisions - I took the liberty of returning null if there is any exception thrown while reading data from disk using readPage()

Unit Test addition - One example of a unit test that can be added is to check what happens when there is an error while reading Pages in HeapFile (readPage).

Changes to API - No Changes

No missing/incomplete elements

Feedback - Preferably more comprehensive test set to make sure the code we've written so far works properly. There are some cases where the tests pass but the code is incorrect. Hence, a more comprehensive test set for the different cases might be really helpful.