Collaborators: Calvin Chan and Simon Zou

TupleDesc: This was implemented using a vector of TDItems so as to have random access for the getFieldName and getFieldType functions. hashCode is implemented by hashing the toString.

Tuple: This is implemented with an array of Fields so as to have random access for setting and getting fields. The iterator is implemented by wrapping the array in a Java ArrayList and returning the iterator (had to look this up).

Catalog: A private Table class is written to store the name and file. Two HashMaps are used, one to map the table name to the table and the other to map the table id to the table. This allows for constant time access for the various getter methods (getTableId, getTupleDesc, etc.)

BufferPool: The cache is implemented as a HashMap that maps PageIds to Pages. This allows the getPage function to check in constant time if the Page is in the cache and add it if necessary.

HeapPageId: equals compares the two ids and hashCode concatenates the two ids into a string and uses Java's built in hashCode for strings. No complex data structures were necessary.

RecordId: Similar to HeapPageId, equals compares the PageId and tuple number to determine equality and hashCode concatenates the two ids into a string and uses Java's built in hashCode for strings. No complex data structures were necessary.

HeapPage: getNumTuples and getHeaderSize are implemented according to the spec. isSlotUsed finds the particular bit in a particular byte and then checks if that bit is on. getNumEmptySlots then iterates through the header using isSlotUsed to the get the count. A separate class was written for the HeapPage iterator, which calculates the number of tuples used and iterates through them.

HeapFile: readPage uses Java's RandomAccessFile and calculates the offset based on the pid and reads the page into main memory. A separate HeapFileIterator class was written for the HeapFile iterator. The iterator uses the HeapPage iterator if it reaches the end of it, checks for new pages, and if they exist, starts using the next page's iterator.

SeqScan: SeqScan calls the underlying DbFileIterator functions and otherwise has simple getter methods.

Changes to API: No changes to public interfaces were made although an additional public function availableTuples() was added to HeapPage so that the HeapPageIterator could access and use it.

Difficulties: We spent about 19 hours on the project. We got the idea for implementing hash codes by hashing string representations from asking the professor in office hours. Understanding how to do the bit manipulation in HeapPage was took some time. The HeapFile and HeapFileIterator were probably the most complex parts of the assignment and getting used to the Java syntax for working with File I/O, exceptions, and Random Access. It took us several hours to debug an error we were getting in systemtest/testsmall where we needed to cast getNumTuples() as a double before calling Math.ceil on it in getHeaderSize of HeapPage.