CS 6530 Database Systems

Project 1: SimpleDB Basics (Tuple, Catalog, Buffer Pool, Heap File, Scan Operator)

Team 15 - Tanvi Gangadhar(u1205740), Greeshma Mahadeva Prasad (u1141804)

DECISION DESIGN

Fields and Tuples

- Implementing methods in *TupleDesc.java* was straightforward. We created the required class members and implemented the methods.
- In *Tuple.java* we maintained a fieldList of type ArrayList and we iterated over the same while overiding the toString() method. Using the arrayList iterator to iterate over the tuple fields.

Catalog

- While implementing the methods in *Catalog.java*, we realized that multiple HashMaps had to be maintained. To simplify this, we created a *Table class*. The Table class has members, DbFile, tableName, primaryKeyField, tupleDesc and tableId.
- Used ConcurrentHashMap in *Catalog.java* to maintain the list of tables because the operations had to be thread-safe, the unique Id from the Dbfile is the HashMap key and the value is a Table object.

BufferPool

- In *BufferPool.java* as well we used a ConcurrentHashMap to maintain Pages with key-value pair as PageId-Page.
- We've implemented getPage method such that if a page is already present in the BufferPool, we simply return it. Otherwise, we retrieve the page from the DBFile, add it to our BufferPool and then return it.

Heap File access method

- *HeapPageId and RecordId* had similar implementations, had to populate the hashCode and getter methods.
- In *HeapPage.java* we used the formulas provided for computing tuple size and header size.
  To compute the number of empty slots, we had to determine whether a particular slot was used or not, for this we had to find two values,
  1. Which byte in the header has the information about the given slot.
  2. Which bit in the header byte should be checked (0 - unused, 1- used).
- *TupleIterartor* - to iterate over all tuples. Wrote methods to override hasnext()- keeps iterating and checks for the next available slot, returns true if a slot is available and next()- returns the next available Tuple.
- *HeapFile.java* - When a page requested is not in BufferPool, it calls the readPage method.

To read a page from the dbFile, we used a RandomAccessFile pointer. The offset and length of bytes to be read was computed using pagesize and page number. The page data is read as a byte array and returned as a HeapPage to the BufferPool.

- For the dbFileIterator, we created a new class HeapFileIterator that inherits the abstract dbFileIterator class. Implementing close(), open() and rewind() were fairly simple, however for readNext() we had to check a few cases,
    1. If the existing page had more tuples.
    2. If the tuples in the current page were exhausted and we had to read tuples from the next page in the same heapFile by setting the tuple iterator.
    3. End of file.

Scan Operator

- Implemented required methods that are used to iterate over the tuples using the DBFile Iterator which was pretty intuitive.
- Prefixed tableAlias to all field names in the TupleDesc which becomes useful when joining tables having fields with the same name.

API CHANGES

We have not made any changes to the provided API. However, we have implemented two new classses

- Table.java
- HeapFileIterator.java

MISSING/INCOMPLETE ELEMENTS

All the unit and system test cases passed, we believe we have implemented everything needed as per project 1 requirements. When we looked at the getpage() method in buffer pool, we noticed that transaction id and permissions were being passed but were not needed to be used to fulfil the requirements for this lab.

TIME SPENT

We pair programmed most of part of the project and together we spent about 9-11 hours. The breakup is as shown below,

- Getting familiar with the environment and setting up eclipse/intelliJ, Git - 1 hour
- Tuple, Catalog - 3 hours
- BufferPool, HeapFile, Scan operator - 6 hours
- Checking for edge case, testing - 1 hour

DIFFICULTIES ENCOUNTERED

We spent most of our time trying to figure out the HeapFile iterator and HeapPage tuple iterator, we were only checking if the next tuple was available instead of the next earliest available tuple. Once we did that we could similarly figure out HeapFile iterator. Also, in BufferPool while adding a new page from dbFile, we were not returning the updated value from the BufferPool map.