Lab 2 – SimpleDB Report

Shubhika Garg
Qin Zheng

TupleDesc.java

- **Time Spent:** 5 hours
- **Tasks & Solutions**
  - **TupleDesc ()**
    - An ArrayList is used because it is a dynamic array and we do not have to specify the size while creating it, the size of the array automatically increases when we dynamically add and remove data items.
  - **numFields()**
    - Used the size() function to return the number of fields in the given TupleDesc
  - **getFieldType(), fieldNameToIndex()**
    - Used the get() method to retrieve value at a specific index.
  - **getSize()**
    - We did addition of the size of tuples and stored in a variable for size of tuple in bytes.
  - **TupleDesc merge()**
    - An iterator is used to retrieve elements one by one.
    - The hasNext() method returns true if the iteration has more elements.
  - **toString()**
    - A StringBuilder is used as it creates a mutable sequence of characters. The append function appends the specified string at the end of a string.
  - **equals()**
    - Usage of instanceof keyword for checking if a reference variable is containing a given type of object reference or not. Here, it checks if object o is not an instanceof TupleDesc.
- **Problems**
  - In the method merge(), initially we initialized two loop variables with value 0 and used it in td1 and td2. We then re-read the comments which stated to have last fields of td2. We then removed the second loop variable.
  - **'> ='** used in some places instead of '>=' and missed curly braces in some conditions. It gave error: illegal start of expression.

Tuple.java

- **Time Spent:** 1.5 hours
- **Tasks & Solutions**
  - **toString()**
    - A StringBuilder is used as it creates a mutable sequence of characters. The append function appends the specified string at the end of a string.
  - **Iterator<Field> fields()**
    - The asList() method is used to return a fixed-size list backed by the specified array.
  - **getTupleDesc(), getRecordId(), setRecordId(), setField(), getField(), resetTupleDesc()**
    - Used this keyword to refer to the current class objects.

Catalog.java

- **Time Spent:** 3 hours
- **Tasks & Solutions**
  - **Catalog()**
    - Initialized the ArrayList variables.
  - **addTable()**
    - Used set() method to replace element present at loop variable index with method parameter values.
  - **getTableId()**
    - If the method parameter name is not null and is equal to the specified name of the list, then return the DBFile of that specific index.
  - **getTupleDesc, getDatabaseFile, getPrimaryKey() and getTableName()**
    - Used the get() method to retrieve value at a specific index.
  - **Iterator()**
    - Used an iterator() to retrieve elements one by one.
  - **clear()**
    - Initialized the ArrayList variables again.
- **Problems**
  - In the method addTable(), initially used the same loop variable for both the while loops without re-initializing the variable with 0 before using it in the second while loop.

BufferPool.java

- **Time Spent:** 1 hour
- **Tasks & Solutions**
  - **BufferPool()**
    - Defined the LinkedList and the HashMap variables and also the numPages variable.
  - **getPage()**
    - Used the getCatalog() method of the Connection interface to return the name of the current catalog /database, of the current connection object.
    - Used the addFirst() method to insert a specific pid at the beginning of a LinkedList.
    - Used put() method to associate the specified value (p) with the specified key (pid) in the map.

HeapPageId.java

- **Time Spent:** 1 hour
- **Tasks & Solutions**
  - **hashCode()**
    - We use Objects.hash to implement it
  - **equals()**
    - Check if the type, page number and table id of current pageId are the same with those of another object
- **Problems**
  - For equals(), we didn't check the instance type of the comparing object at first and it failed the test case.

RecordId.java

- **Time Spent:** 40 minutes
- **Tasks & Solutions**
  - **equals()**
    - checking if the type, tuple numbers and page id of current RecordId are equal with those of another object.
  - **hashCode()**
    - We used Objects.hash.

HeapPage.java

- **Time Spent:** 2.5 hours
- **Tasks & Solutions**
  - **getNumTuples()**
    - The number of bits used by a tuple is this.td.getSize()*8+1, the extra one bit is for indicating whether the tuple is used.
    - The total number of bits in one page can be obtained using BufferPool.getPageSize() * 8.0
    - As the tuples in a page should all be complete, we used (int) Math.floor() to round it down.
  - **getHeaderSize()**
    - The size of header is linked with the number of tuples in the page, so the number of bytes used by the header is this.getNumTuples() * 1.0 / 8.0.
    - As each tuple takes a bit in header, we use (int) Math.ceil() to round it up.
  - **isSlotUsed()**
    - The corresponding byte for the i^{th} slot can be obtained using header [i / 8]
    - The offset in the byte for the slot can be obtained with 1 << (i % 8)
    - For example, when the slot_id is 26, then the bit in the header indicating its usage is the second bit from high to low, in the 3rd byte (start from 0)
  - **iterator()**
    - Add tuples used to the list
- **Problems**
  - For isSlotUsed(int i), at first we tried to use a for loop to check the bitmap and misunderstood the order from lower to higher, but it turned out using ">>" is much better.
  - For iterator(), we forgot to check if the slot is empty at first.

HeapFile.java

- **Time Spent:** 4 hours
- **Tasks & Solutions**
  - **getId()**
    - Using absolute file name for hashing

- **readPage(PageId pid)**
  - The method is only called by BufferPool.getPage()
  - Using RandomAccessFile.seek(), we are able to seek from the offset rather than from the beginning
  - with RandomAccessFile.read(), we can put data into buffer
  - For iterator(TransactionId tid), we create a new class named HeapFileIterator
- **readNext()**
  - First, check if it is the last tuple in the page, if yes, go to the next page
  - Then in the while loop, we check if the offset exceeds the range of current page, if yes, we go to the next page
  - If this is the last page, then we stop iterating

- **Problems**
  - For readPage(PageId pid), we misunderstood the offset in RandomAccessFile.seek() and RandomAccessFile.read() for the same thing at first and it failed.
  - For readNext(), we forgot a lot of things, yet still pass the HeapFileReadTest. Till doing ScanTest we knew how wrong we were.

SeqScan.java

- **Time Spent:** 4 hours
- **Tasks & Solutions**
  - **reset(int tableid, String tableAlias)**
    - Sets a new TupleDesc object.
- **Problems**
  - We forgot to add one when turning the page in HeapFile.readNext()
  - We forgot to check if the page number is within current page and if it is null.