Yinqi Tang
21758061
tangyinqi@gmail.com
October 15, 2014

ICS 222 Project 1

Design:

Key idea and how I design: I regard the following two points as the key points of the design of project 1: 1. free space management of each page, 2. slot directory management of each page. With the help first one, the page file manager is aware of which page has left space for the incoming record to be inserted. With the second part, the record based manager is able to know which slot in the page has enough space to hold that record, and which specific slot the record is when the record based manager needs to read that record.

For each file, the information of free space size of each page and the slot directory of each page will be saved in a file named as "filename.info" in the same directory with "filename", when the "filehandle" tried to close the file. Next time the file is opened, those page information will be read, and used as a reference for the future records to be inserted, and records to be read. Both save and read operation is managed by the Class FileHandle.

Why I design in such way: According to the textbook, free-space management is necessity for both records reading and inserting. So I added two class "class freebytes" and "class slotdirectory" to manage, which was designed as private members of the class fileHandle. But when I run the test case 10 for the first time, reading records operation failed, because those the information of fileHandle was in the memory, which was deleted after the class was destructed when it was closed after the inserting records operation finished. So I realized I need to save them in a file in disk so that the record based manager is able to read them for further use.

Implementation:

I added the following classes:

1. class SlotDirectoryNode:
it records the information of each slot 1. slot number, 2.offset, 3.length, and some getter setter method.

2. class SlotDirectory
it records the information of slot directory of each page: 1.page number, 2.lastSlotNum of the page 3. a slot list: vector<SlotDirectoryNode *> for all the slot information of that page.

3. class FreeBytes
it records the information of the free space of each page: 1. page number, 2. free bytes

4. in class FileHandle: I added the following private member and functions:
   a. FILE *pFile: the file handle for file of data.
   b. FILE *pageInfo: the file handle of file of file information such as free space and slot directory
   c. int pageMaxNum: the current max page number
   d. vector<FreeBytes *> fbList: fbList[i] is the free space information for page no.i.
   e. vector<SlotDirectory *> pageSlotDirectory: pageSlotDirectory[i] is the SlotDirectory information for page no.i
   f. savePageInfo(): save the freespace and slot directory information to file “filename.info”
   g. readPageInfo(): read the freespace and slot directory information from the file “filename.info”
   h. besides, class FileHandle has other methods responsible for the operations such like: appendToPageFreeSpace() for appending to the list fbList and appendToSlotDirectory() for appending to the list pageSlotDirectory when a new page is appended. And lookforPage(), lookforSlot(), getOffset() of slot and getLength() of that slot, and other operations supporting free space management and slot directory management when reading and inserting records.

5.readRecords:
   a.get the address of that record: 1.page number from parameter rid.pageNum, 2.slot offset from start of that page, based on information from parameter rid.slotNum, and rid.pageNum.
   b. get the bytes length of that specific slot which is read from the file “filename.info” when opening the file.
   c. read that number of bytes starting from the address.

6.insertRecords

a. No page with free space available found: append a new page, insert that record from the starting.

b. There is such page with enough space in the end of page: append a new slot with size of that record to the last slot.

c. There is a page with enough bytes left, but not enough in the end of page: this situation happens when some records in the middle of a page is deleted. Not implemented yet: we are supposed to re-organize the file so that all available space is moved to the end of the page, so that the record can be appended to the last slot. Luck there is no such test case.

TestCases: All passed. on mac, with gcc version: (though I didn't use c++11 features)

```
gcc -v
Configured with: --prefix=/Library/Developer/CommandLineTools/usr --
with-gxx-include-dir=/usr/include/c++/4.2.1
Apple LLVM version 6.0 (clang-600.0.51) (based on LLVM 3.5svn)
Target: x86_64-apple-darwin13.4.0
Thread model: posix
```