CSE444

Lab1 – Write up

Yuanfeng Li

2021/01/21

What Lab1 was about:

In general:

In the holistic view, lab1 was a process to finish the whole process of reading Files from disk to main memory, and then using HeapFile objects to process files in disk, and tuples in the pages so that it can give operators in SQL the information such as table name, field names, and tuples that they need.

In details:

TupleDesc.java and Tuple.java :

Two classes that have almost all the information about tuples. In my TupleDesc.java class, I have arrays of Type, field names, and number of fields. So basically, TupleDesc can give you all the information about the tuple such as schema, field name, titles etc. And Tuple are mainly storing the information about the content of Tuple, and the location of the tuple in pages.

Catalog.java:

Catalog is a class that stored all the information about tables in the database. It stored their schemas as well, the difference between these schema and TupleDesc's typAr and FieldAr, is that catalog stores primary key, tableIds, and names, which TupleDesc doesn't have.

Buffer Pool.java:

Buffer Pool class has a lot of functions and it is very important during the whole process. In lab1, I only implement the getPage() function, in which it asks Heap File to readPage, and save it into cache. There will be more functions needs to be implemented in the future labs.

Heap File & related java classes:

Heap File and its related java classes such as HeapPageId, RecordId, Heap Page. All of these classes are aiming to read/write the File from/to disk, find out the page in File using tableId, pageId, FileId, and get the tuple using RecordId as well. I have done a lot of hashcode, iterator, and calculation in these files. And these classes are working really close with Buffer pool class and sequential Scan method.

Sequential Scan.java:

This class worked closely with Heap File classes as well, the major goal for this is to open() the HeapFile iterator and let it start to read the files from the disk. It is like a representative of Operators, which could tells HeapFile classes to start work and find out the tables, and tupleDesc it needs.

Design Decision:

Not that much, but I decide to put the HeapFile Iterator inside the HeapFile class as a subclass iterator, because I think it is easy to modify the code.

Example of unit potential unit test: We can add a unit test to check if there is an error while reading Pages using HeapFile class' readPage() function.

API: No Changes

Missing or incomplete codes: No Missing