REPORT

Jiasen Li 李嘉森
517030910405
Report Github

Design Decision

1. **Tuple and Tuple Description**

   I used a vector of TD(standard helper class provided in the project) to store the tuple description.
   For the tuple, I used the Vector<Field>
   I do not need a hash code for tuples because a Record ID can identify a tuple. Besides, a tuple should not be an identifier of anything, unless you search for a tuple.
   Note: If findTuple() should be implemented, I should have the Tuple with HashCode.
   TD HashCode: Vector<Integer> with fieldName.hashCode() and fieldType.hashCode() in it.
   Return the Vector.hashCode(). (Why do I write it like this? Because Vector hashCode is well implemented and very fast optimized by Oracle).
   Tuple Desc hash code is the hash code of Vector<TD>.
   The rest part of lab 1 also follows the thoughts here.
   toString are utilizing standard toString Methods of Vector.

2. **RecordID and PageID**

   HashCode is of the same method
   Use Vector<Integer>.hashCode()
   Note: No To_String Method, Because it is just an index instead of data.

3. **Catalog**

   I do not need **class Table** at all.
   I use several Hashmaps to store the information.
   Int to Object HashMap stores information indexed by the table id.
   String to int HashMap deals with getID(String).
   I also use a Vector to store TableID.
   I check the duplication when it adds tables.
   Why do I do this? Because:

   a. Such standard containers are optimized by Oracle.
   b. The time complexity drops to constant time. No need to check tables one by one.

   TableIDIterator: I just return the Vector Iterator of TableID.
   Note: Table ID is the hashCode of the file, which is not changeable, so I do it this way. Also this way is more efficient if you just iterate the number.

4. **Buffer Pool**

   Buffer pool is actually a HashMap from PageID to Page. HashCode see PageID&RecordID. When the page is in the Map (I mean the Buffer Pool), the Page would not be released so it stays in the RAM. When getting a new Page, add it to the Map.

5. **Page**

   To construct a page in RAM with byte[PageSize] as its input.
   I do some simple calculations in it.
   I use bit count to count Empty Slots. This will make finding Empty Slots faster.
   I use standard Vector<Tuple> to form an Iterator<Tuple> because java is a language based on objects and using pointers. The cost of it is just a Vector of Pointers that point to Tuples. And the space cost is not higher than saving the Tuples in RAM.
   Its speed is not slow optimized by Oracle.

6. **DbFile**

   This part actually contacts the file system indirectly using File in Java.
   There is a trick that the offset of the page in a java File in the File is the PageNumber.
   For example, the 3rd Page has page number #3. Page Number is not like uuid. Page Number is not uniformly distributed.
   The doc of the homework is not very clear about it, but it can be inferred according to BufferPool.java.
   I use simple reading from files (random access). Number of Pages is inferred by the length of the file, which is a little bit tricky.
   The Iterator can utilize the iterator in Page.

   ```
               int PageIndex = -1;
               int TableId = -1;
               Iterator<Tuple> pageTupleIterator = null;
   ```

   After `if (!hasNext()) throw new NoSuchElementException();`, the next is in pageTupleIterator.
   Note: The DbFileIterator has open(), which is different from normal Iterators

7. **SeqScan**

   Just a simple task.

API Change? Incomplete Element

1. NoChange. I think that changing API may cause version problems, so I do not change it.
2. Any Required Element in Lab 1 is implemented.
3. Some functions like inserting tuple will be implemented in the following labs.

How long have I been doing this?

1. I forgot. I think it is between 1 to 3 days.
2. And I am not confused after asking the TAs. TAs are very nice and kind.