DataBase-Lab1-Report

1. Designing part

1.1 Exercise 1:

Exercise 1主要是让我们去实现数据库的Tuples和TupleDesc，其中Tuples由一组 Field 对象组成，Field 是不同数据类型（例如，整数、字符串）实现的接口。Tuple 对象由底层访问方法（例如，堆文件或 B 树）创建。

Id(int)	Name(string)
001	Jack
007	Russ

那么(001, Jack)就是一个Tuple，其中的值就是Field，然后TupleDesc 是(id(int) , name(string))。

在exercise中具体实现这三个骨架:

src/simpledb/TupleDesc.java

src/simpledb/Tuple.java

src/simpledb/RecordId.java

Tuple就是filed objects的集合，然后filed类型可以不同。TupleDesc则是Tuple的schema，也就是其元信息。RecordId是Tuple的ID，通过页号PageId和元组在页内的偏移量TupleNum生成，页号PageId根据不同的存储结构(Heap或B+tree)会有不同的实现。

1.2 Exercise 2：

Exercise 2是去实现数据库里的Catalog，顾名思义就是为数据库添加一个目录，可以去访问所有表的集合。 Catalog 全局目录是为整个 SimpleDB 进程分配的单个实例。 全局目录可以通过方法 Database.getCatalog() 检索。 为追踪所有可用的数据表（ data table)， 设置了两个成员变量来建立 id 到 table，name 到 id 的映射，因此应对需求我另外设计了一个Table类在代码中。

1.3 Exercise 3：

虽然数据存储在磁盘中，但是也不能每次都从磁盘中读数据，这样性能是极差的。也因此为了提升查询性能，后续数据从磁盘中取出后，缓存在内存中，下次查询同样的数据的时候，直接从内存中读取、处理。Exercise 3是去实现数据库里的BufferPool，BufferPool （缓冲池）负责在内存中缓存最近从磁盘读取的页面（lab1中不需要考虑驱逐脏页），用以暂存之前访问过的 Table，以提高访存速度。而这读取的数据，则被分为若干的页，以页作为磁盘和内存之间的交互的基本单位。也就是下一个Exercise要实现的内容： 堆页。在此， 我们也需要建立 PageID 到 Page 的映射来记录缓存区占用情况。PageID 标识了 <tableid,pgNo>，表明缓存区存有 tableid 对应的表的第pgNo 页。

1.4 Exercise 4:

SimpleDB中的数据有两种组织方式，其中一种是堆文件Heap File，即无序存放。 对于实现Heap File的存放方式，在SimpleDB中，一个Heap File包括了一系列页，称为Heap Page，而每个页能存储的数据量是固定的(默认设定为4096KB)，在磁盘中这些Heap File就以无顺序的方式存放，需要用的时候以页为单位读入内存中就行。

Exercise 4 要求我们实现 HeapPage 类、HeapPageId 类和 RecordId 类。一个 HeapFile 与一个 Table 相联系，堆页概念的引入帮我们对页内结构进一步细分，将页分为多个槽 slot，每个槽存储一个 Tuple 和 1 bit 的标识符（判断该槽是否被赋值）。

在exercise1中， 每一个Tuple都会记录一个RecordId， 而这个RecordId = PageId + tupleno， 这个PageId在Heap存储结构中的具体实现就是HeapPageID，这个实现的成员变量是:

private int tableId;

private int pageNum;

需要注意的是：

page不仅记录了HeapPageId，TupleDesc，所有的tuple，还记录了所有tuple的标记，也就是用一些位置来标注每一条记录是不是有效的，这种方式称为Bit Map，每个元组对应1bit的位置，因此1B的位置可以标注8个元组，因此每个Heap Page中能存放的页的数量需要通过下面的方式计算：

//page_size*8 / (TupleDesc元信息计算得来的每一个tuple的大小+1bit标记位)

tuples_num= (int)Math.floor((BufferPool.getPageSize()*8.0)/(td.getSize()*8.0+1.0));

之所以需要+1，是因为每个page都会有一个header，这个header里面有个bitmap来表示对应Tuple是否已经in use了。

在这个page初始化的时候，就会初始化成员变量，会根据读取(上一层是HeapFile，从disk读取完毕会传到page中进行初始化)到的当前page的数据流进行初始化 所有tuple的标记+所有tuple数据；

需要设置一个dirty标记当前这个page是否成为了脏页；

1.5 Exercise 5**:

Exercise 5 要求我们实现 HeapFile 类。一个 HeapFile 关联一个一个 Table。HeapFile是DbFile interface的实现，一个HeapFile就是一张表/一个文件。HeapFile.readPage会根据传入的pageId通过RandomAccessFile.seek定位到该file的位置，然后通过randomAccessFile.read读取一页的数据。

HeapIterator:

- 成员函数 get_TupleIterator(int PageNumber)先要判断 PageNumber 是否超过 heapfile 对应的 table 的总页数，则调用getBufferPool().getPage(tid,pid,READ_ONIY)写入缓存，否则报错。
- 成员函数 hasNext()是判断是否还有下一个 tuple 可读。由于 tupleIterator 的单位是“页”，所以需要通过 index 不断“翻页”，当该页无 tuple 可读，index++（读指针）并调用 get_TupleIterator(index)查看下一页，若没有下一页，则返回 false（并将index 重设为 NumPage）.成员函数next()就是先调用hasNext()判断有没有下一个元素，然后返回tupleIterator.next()。

1.6 Exercise 6：

要求我们实现一个 operator，较为简单因此不多赘述。

2. 实验心得

本人前后花费了2-3天时间完成此次lab1。

在学习及lab编写过程中，此前本人接触 C++和 python等语言较多，对java不是很熟悉，所以耗费时间最长的是java语言的认识与了解，环境配置等问题也占据了较长时间，因此最后索性选择jetbrains的IDEA平台来写project。

在代码思路构建的过程中，我认为最难理解的部分是iterator和 HeapPage等，后续结合书本内容及在线资料等学习后对概念有逐渐清晰的认知。