Lab 1

1.Designs of Different Java Files (Only List Import Parts)

1.1 Exercise 1:

Exercise 1 要求我们实现 Tuple.java 和 TupleDesc.java 文件。Tuple 类作为 Table 的重要组成部分，由三部分组成——td(TupleDesc,存储 Table 的 Schama 信息)、fields(每列的值)、rid(tuple 在 disk 中位置)。TupleDesc 类存储了 Table 的 Schama 信息，由一系列元组(TDitem<type,fieldAr>)组成。

TupleDesc.java

- 成员变量 tdItems(记录每一列的类型和列名)；类中实现了 TDitem 类，由两个成员变量组成——fieldType 和 fieldName。
- 成员函数 getFieldName(int i)和 getFieldType(int i)用来获得第 i 个列的列名和类型，注意要判断是否溢出。
- 成员函数 fieldNameToIndex(String name)将列名转换为 index。在完成 Exercise 6 之前，我未曾考虑到“alias.fieldName”的情况（但是测试还是过了?）。因此，我们首先需要通过 String.contain('.')判断 name 是那种形式，**然后分离出 fieldName 部分**。
- 成员函数 merge(TupleDesc td1,TupleDesc td2)用来合并两个表格，先判断两个 table 是否为空来提高鲁棒性；成员函数 equals(Object o)用来比较俩个 schama 信息，首先判断类型是否为 schama，然后判断列数是否一致，最后遍历比较每一个 TDitem 是否相等。

Tuple.java

- 成员函数 setField(int i,Field f)用来设置列值，注意当 i 等于 field.size()时，需要将值添加到 fields 的末尾,若溢出则不再添加；成员函数 getField(int i)需要条件判断，当 i<0 或 fields 未初始化或溢出时，需要返回空值
- 成员函数 toString()，遍历 TDitems，返回形如“co11 (value1)\t col2 (value2)\t…"字符串。

1.2 Exercise 2：

Exercise 2 要求我们实现 Catalog 类，用以追踪所有可用的表，因此设置两个成员变量 Map_id_table,Map_name_id(考虑到一个 getTableId(String name)待实现，所以建立该映射而不是建立 Map_name_table 映射)来建立 id 到 table，name 到 id 的映射。由于 Table 在 lab 未实现，因此我简单实现了 Table 类，受 addTable 函数的提示，我设置了三个成员变量。

Table.java

- 三个成员变量——file(DbFile),Table_Name(String),pkey_field(主键,String)，并构建 getFile()、getPrimaryKey()、getName() 成员函数，方便后续调用。

Catelog.java

- 成员函数 getDatabaseFile(int id)、getPrimary(int id)和 getTableName(int id)等都是通过 tableid、name 进行索引返值，需要先判断 Table 是否可用（即 CateLog 是否记录了）。

1.3 Exercise 3：

Exercise 3 要求我们实现 BufferPool 类。BufferPool 是缓存区，用以暂存之前访问过的 Table，以提高访存速度。缓存区有多个页，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。

BufferPool.java

- 成员函数 getPage(tid,pid,perm)模拟了如何读取数据，当 pid 不在映射表中，说明缓冲区中未存有该表的特定页(pid.pgNo)，需要从 disk 中转移过来，并添加到映射表中。然后根据映射表返回所需要的 Page。

1.4 Exercise 4:

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）。

HeapPageId.java

- 两个成员变量 tableid 和 pgNo，用来标识页。成员函数 hashCode()用来存储堆页的 hash 值便于快速索引，我首先拼接 tableid 和 pgNo 为 tableid×521+pgNo×1314，然后用 Integer.toString(xx).hashCode()。equals(Object o)沿用 Exercise 1 中思

路即可。

HeapPage.java

- 每页的 Tuples 数量 = $\frac{\text{page}_{\text{size(byte)}}\times 8}{Tuple_{szie(byte)}\times 8+1bit(valid\ bit)}$,Header 是由标识符组成，HeaderSize(byte) = $\left\lceil\frac{Num\ Tuple}{8}\right\rceil$。需要注意 byte 和 bit 直接的转化，以及 Tuple.getSize()函数和 Math.ceil()取整函数调用。
- 成员函数 isSlotUsed(int i)用来判断该 slot 是否被占用。Header 数组的元素是 byte，即 8bit，我们先通过 i/8，定位到标识符所在的位置；然后 i%8 计算标识符在该 byte 元素里的偏移量,并右移>>offset（**我采用 Big Edian**）,取最后 1 个 bit 即所求标识符。注意溢出判断。getNumEmptySlots()函数遍历 numSlots，调用 isSlotUsed(i)函数，判断是否占用并计数。
- Iterator<Tuple>ierator()，遍历 numSlots，将 tuple[i]加入到数组中，然后返回迭代器。

RecordId.java

- hashCode()函数对 tuple 进行标识，确定一个 tuple 需要<tableid,pgNo,tupleno>。首先拼接 tableid×521+pgNo×1314+tupleno ×15，然后用 Integer.toString(xx).hashCode()压缩。

1.5 Exercise 5(The Most Difficult):

Exercise 5 要求我们实现 HeapFile 类。一个 HeapFile 关联一个一个 Table。我们需要实现一个 HeapFile 迭代器(DbFileIterator 的子类)。**尤其注意 BufferPool 写入时是整页写入，防止内存泄漏（*)！**

HeapIterator(存有一个 tupleIterator 迭代器)

- 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_ONlY)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()。

HeapFile 的其他成员函数

- 成员函数 readPage(pid)需要先判断 pid.pgNo 是否溢出了 HealFile 的边界**（第一层防止内存泄漏）**；其次判断 read(读取到的实际 bytes)是否等于缓存区页大小，否则（*）条件违背，说明缓存区写入时，将未满一页的内容写入了，造成内存泄漏**（第二层检漏机制）**。无论读取是否顺利，都要将 f 文件关闭。
- 成员函数 numPages(),由于要整页写入，所以向下取整。（有点疑惑）。

1.6 Exercise 6

Exercise 6 要求我们实现一个 operator，比较有意思的是要拼接 tableAlias 和 table 的列名（也为 Exercise 1 提供修正）。其他操作偏简单故不赘述。

2. 实验心得

本人一共花了两天时间完成了这部分 lab。在学习过程中，我认为比较困难的部分是前期对 java 的学习，由于我只学过 C++和 python，因此在前期需要借助 ChatGpt 帮助我对 java 语言有一个大致的了解。当然还有环境配置，我比较呆，瞎捣鼓，浪费了了 3 个小时┬┴┬﹏┬┴┬。

在构建代码思路的过程中，我认为最难理解的部分是文件迭代器的构建和 HeapPage 的读取。本次实验也给我留下了一个困惑的点，亟待助教老师帮忙指导！**HeapFile 的 numPages 部分采用的是向下取整，说明 File f 有一部分是没被记录到 HeapFile 中的，那如果要访问该部分，需要怎么访问呢？还是说有将 File f 填充的机制被我遗漏了？**提前感谢助教老师解惑，祝您阖家幸福。