**Simple DB Lab1**

* **命名方式：**

对于大多数成员变量采用驼峰式命名法

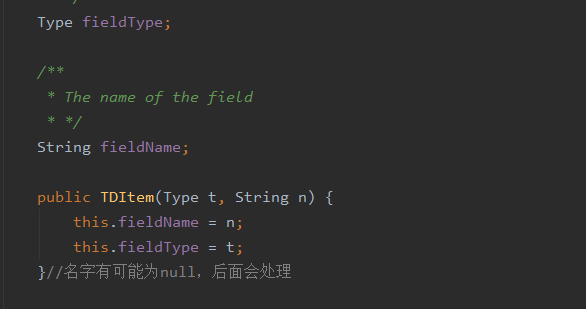
Hash型命名是 ‘key属性’To’val属性’ , ex: idToName

对于临时变量采用 tmp开头，ex: tmpFile

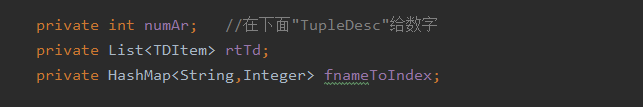
对于迭代器函数采用Itr开头

* **Exercise 1** （TupleDesc&Tuple）

**设计思路：**

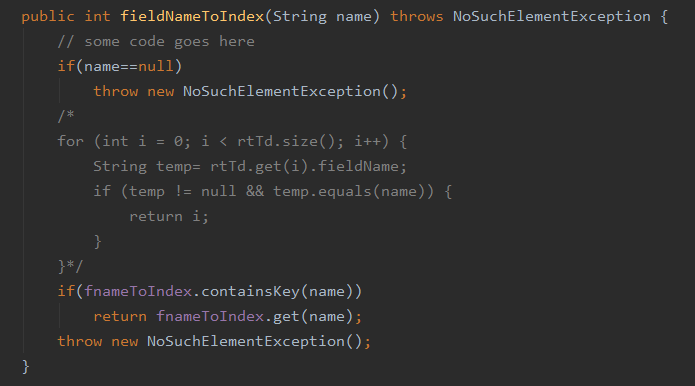
1. 因为内置给了·TDItem类，所以选择使用List<TDItem> 来储存fieldtype 和 fieldname。 List类在随机存储方面和数组一样并不复杂，但是如果以后需要删除某一元素并前移或者更改总的属性有多少个的话会方便很多。

同理，后续文件中能使用List的部分基本也是使用List而不是数组

1. HashMap

这个改动是我第二次完善的时候加上的。

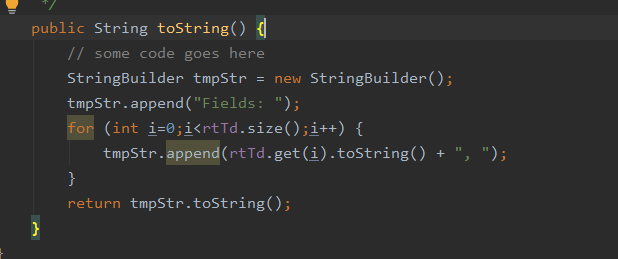
原因在于，在调用下面的 fieldnameToIndex（String name中，如果不使用hashMap，目前我能想到的办法只有遍历，然后逐个比较，显然效率是比较低下的。换成HashMap就可以利用键值直接取值了。（见下图）

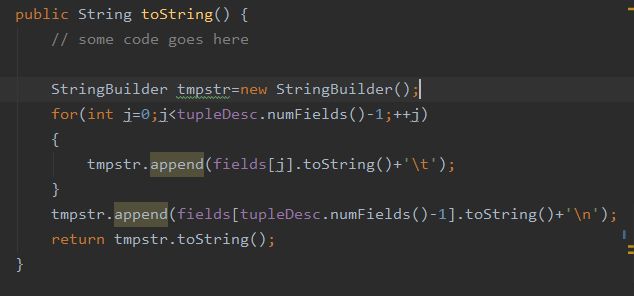


、、、

1. toString

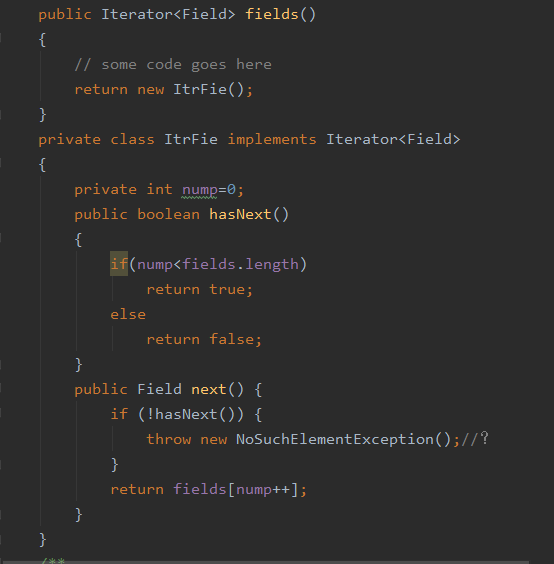
toString()采用的是StringBuilder, 一是因为相比于基础的String类型，StringBuilder 类的对象能够被多次的修改，并且不产生新的未使用对象；再者，相比于StringBuffer,它速度更快。后面的toString也是这样。





4. 迭代器

文件中的迭代器都是继承重写的方式实现的



5.其余部分跟着指示声明变量就可以了

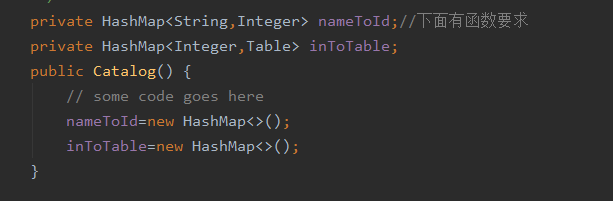
**难点：**

1. 不熟悉JAVA语法，开始对怎么“填空”不太习惯。经常有做到后面的函数才发现前面应该声名更多变量的情况。

* **Exercise 2 （Catalog）**

**设计思路：**

1.HashMap



对hamap的设置源自于下面的各种函数。

**利用inToTable：**getTupleDesc(),getDatabaseFile(),getPrimaryKey(),getTableName()都可以获得，遍历tableIdIterator()也可以调用hashmap自带的迭代器

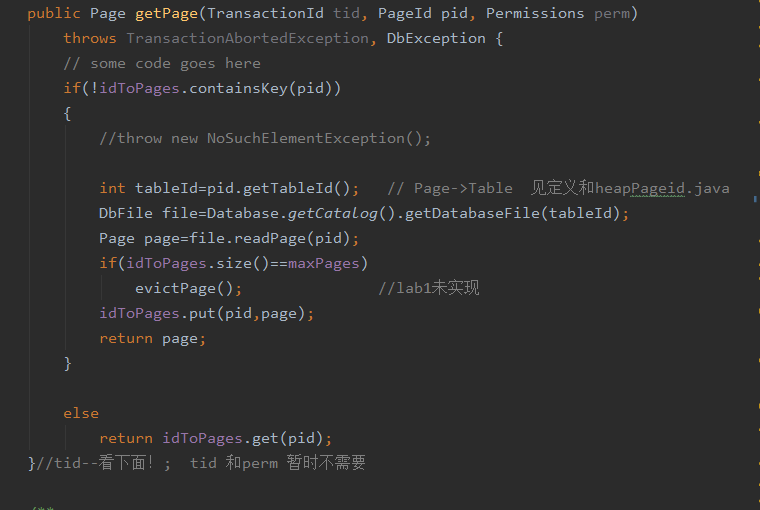
**利用 nameToId：**

与tupeldesc功能同理，在寻找表名对应的id时空间换时间，避免了遍历查找。

**难点：**

典型的看到了后面的函数才知道前面要写什么。一开始采用的是数组，后面发现这么做既不利于维护效率也不高，遂改成hash。

* **Exercise 3 （BufferPool）**



**难点：**

什么都不用写反而是最大的难点。

因为不知道tid和perm是什么，所以暂时也没有声名内部变量处理传进来的参数。通过后续的文件知道了BufferPool类内函数的调用方法

* **Exercise 4**

**设计思路**：

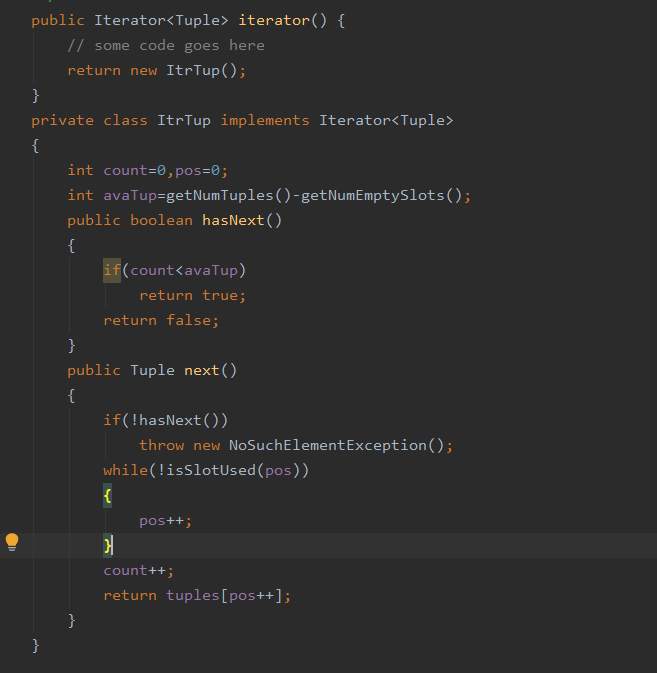
PageId RecordId 都是比较基础的填空，重点在HeapPage

首先明确是每八个tuples一个header，把一个字节的8bits分开来看使用情况，

然后根据根据注释里给的公式计算

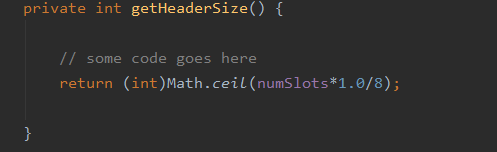
1.迭代器

因为内存不一定是连续占用的。有可能中间有空。所以要设置两个指针遍历

****

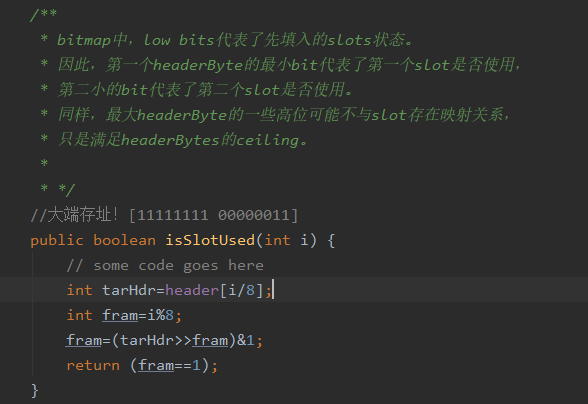
2.HeaderSize

整数除法要\*1.0才能实现ceil的结果



3. isSlotUsed()

虽然总体是大数在低位存储，但对单个的8bits仍然是从右到左存放的，这个影响了对于判断某个具体slot有没有被占用时，应该往哪边移位。



**难点：**

1. Header内部的bitmap存储方式以及获取方式比较陌生

%2 和 &1 的区别，因为之前忘记了调试可以看输出导致困扰了我很久

1. Slot内部有可能会有空位置，要特殊考虑

* Exercise 5

设计思路：

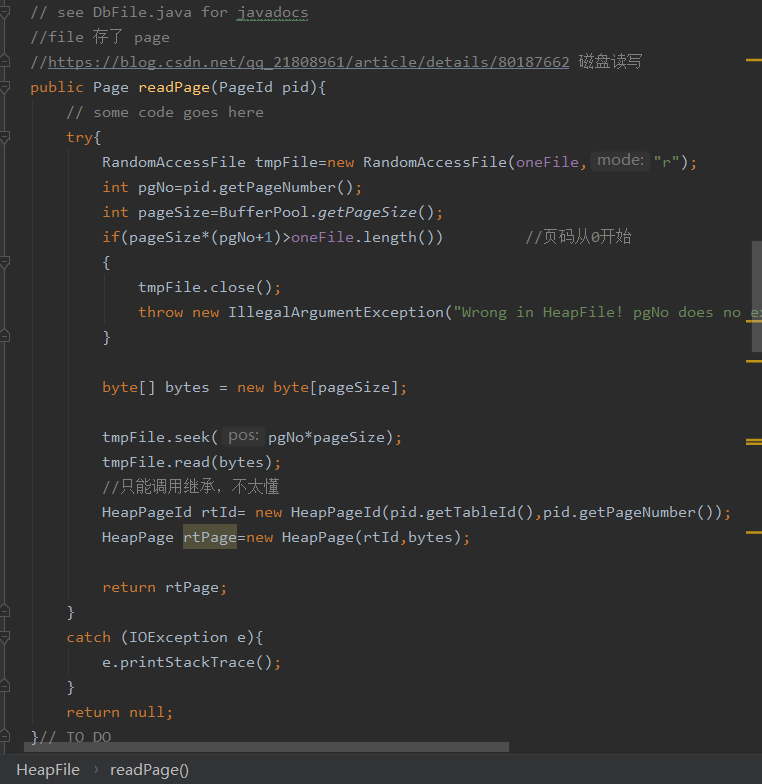
感觉这个部分像是对前面理解的一个测试，因为需要频繁的调用前面写过的函数。 然后一个table会有很多pages，对于这些pages，tid都是同一个值，明白了这个才好写后面的迭代器。

还需要注意页面必须从bufferpool里调取，也符合常识。

1. ReadPage

之前没学过java，磁盘存储的方法是从网上借鉴的。

1. 要注意页码标记都是从0开始，所以临界条件要做适配
2. 虽然返回值是Page，但Page是虚类，还是应该声名HeapPage进行构造
3. 异常处理



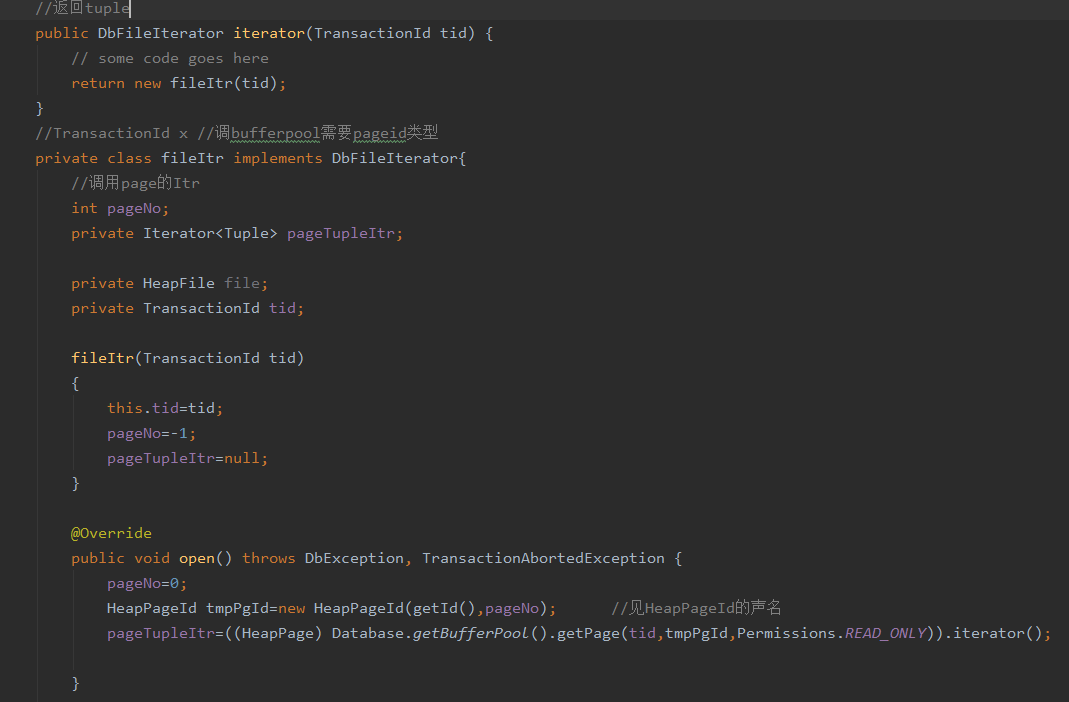
1. 迭代器

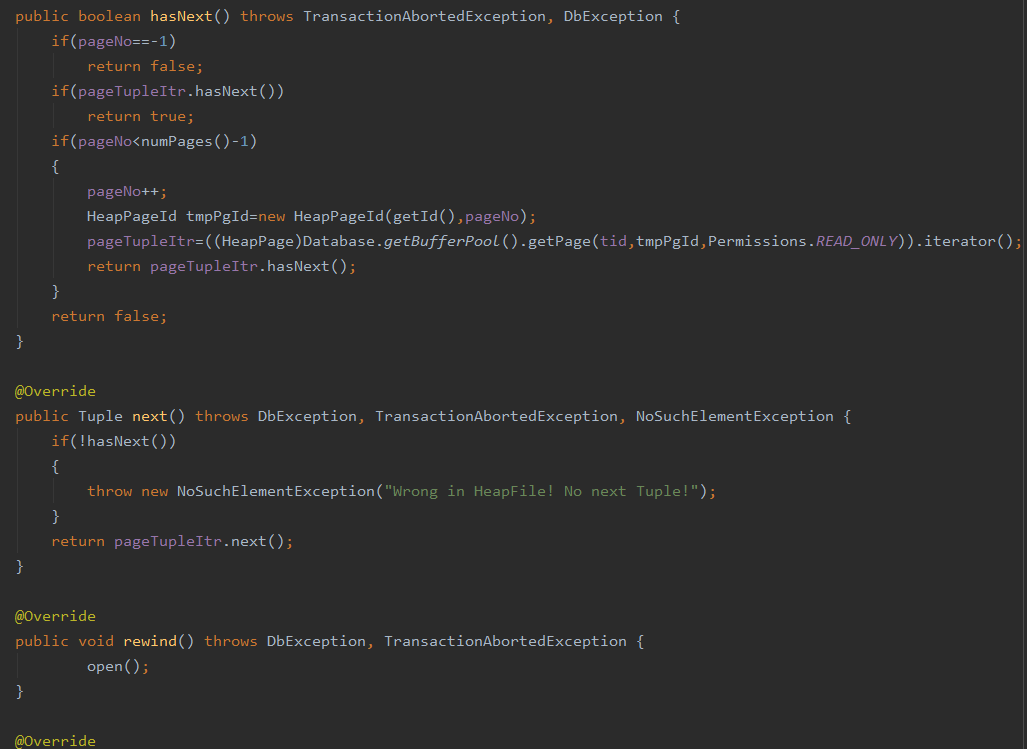
第一，明确归根结底是遍历Tuple，所以可以利用前面写的迭代器再改造。

第二，再一次注意页码从零开始

第三，一定从BufforPool调用

第四，!hasNext()有可能是没有开启迭代器或者到头了，其中后者单页的临界判定可以交给HeapPage的函数来做





* **Exercise6 Seqscan**

总体上来说就是填空，迭代器和前面思路差不多。

注意给每个fieldname改名时的“null”值处理，总的来说我用的是StringBuilder。