SimpleDB　Lab2（数据操作）实验报告

一、关键类的实现

1、BufferPool

BufferPool 是数据缓冲池，是完成 simpleDB 的一个关键类。

成员变量：

- private int numPages;　　//缓冲池中的页面数
- private int numValidPages; //缓冲池中有效的页面数
- private int [] accTime;　　//某个页面的访问次数
- private Page [] page;　　//存放的数据页
- private PageId [] pageId;　　//数据页 ID
- private PageLock pl;　　//页锁，用于 lab3

主要成员函数：

- public synchronized Page getPage(TransactionId tid, PageId pid, Permissions perm)，用于获取一个页。首先根据页的 id 查询 pageId 数组，若发现相同 id 直接返回对应页；否则从磁盘中读取一个页并缓存；若 BufferPool 已满，需要根据 accTime，选一个最近最少用的页剔除，得到空位后再进行缓存。
- public synchronized void insertTuple(TransactionId tid, int tableId, Tuple t)，在 tableId 对应的表中插入一个记录；
- public synchronized void deleteTuple(TransactionId tid, Tuple t)，删除记录 t；
- public synchronized void flushAllPages()，将缓冲池中所有页写入磁盘；
- private　synchronized void flushPage(PageId pid)，将 pid 页写入磁盘。

2、HeapFile

针对 Lab2，新增成员方法如下：

- public void writePage(Page page)，将 page 页写入文件；
- public void addEmptyPage(TransactionId tid)，新建一个空页，用于 addTuple 时所有页都满的情况；
- public ArrayList<Page> addTuple(TransactionId tid, Tuple t)，增加一个记录 t；
- public Page deleteTuple(TransactionId tid, Tuple t)，删除记录 t

3、Predicate

主要成员函数：

- public boolean filter(Tuple t)，该方法将 t 中的值与 Predicate 的操作数中的值相比较，并返回比较结果。

4、IntAggregator(StringAggregator 使用类似的方法)

主要成员函数：

- public void merge(Tuple tup)，将一个新的记录 tup 增加到类中，如果是 no-grouping 方式，直接插入，如果不是 no-grouping 方式，找到同 tup 中 gbfield 相同的第一个元素组,并把 tup 插入到该组的第一个位置。
- public void aggSum()，实现 sum 操作；
- public void aggMin()，实现 min 操作；
- public void aggMax()，实现 max 操作；
- public void aggAvg()，实现 avg 操作；
- public void aggCount()，实现 count 操作；

5、Insert、

成员变量：

- private TransactionId tid; //事务 ID
- private DbIterator dbIt; //待插入的记录，以迭代器的形式给出
- private int tableId;
- private boolean calledNext = false; //是否为第一次执行。

主要成员函数：

- getTupleDesc()，生成仅包括一个 INT_TYPE 域的 TupleDesc 后返回；
- readNext()，遍历 dbit 中的 Tuple，对于 dbit 中的每个 Tuple，调用 BufferPool 中的 insertTuple 方法，将该 Tuple 插入到对应于表 ID 的表中。操作发生在内存，不写入磁盘中。(这是为了实现 No-Steal)

6、Delete

与 Insert 类的实现类似。

7、Filter

Filter 类也是 AbstractDbIterator 的继承，主要用于筛选出给定的一个 Tuple 表中某些符合条件的 Tuple。构造函数中，遍历给定的 Tuple 表，并通过 Predicate.filter() 方法进行判断筛选，符合条件则加入到新的表中。在 open 中返回该表的迭代器。readNext()方法简单地判断表中是否有剩余的 Tuple，有则进行返回。

8、Join

类的实现包括两个 DbIterator，遍历两个表，将其中的 Tuple 逐一进行对比，符合条件的组合加入到新的表中。

二、遇到的困难及解决方法

1、BufferPool 中的 getPage(TransactionId tid, PageId pid, Permissions perm)函数

这是一个关键类中的关键函数，用于获取从 BufferPool 中获取一页。如果要获取

的页已经在缓冲池中，直接返回该页；如果要获取的页不在缓冲池中，并且缓冲池不满，将该页从磁盘读入缓冲池并返回，如果要获取的页不在缓冲池中，并且缓冲池已满，需要淘汰一页，再将该页读入磁盘并返回。

2、BufferPool 中的 evictPage()函数。

使用此函数淘汰缓冲池中的一页，淘汰策略为最近最少访问，通过设置 accTime 数组实现。

3、seqscan 类

实现对表的顺序扫描，这里要注意的是，将表文件读入内存时要按照需要，一页一页的读进来，而不是一次性将表文件全部读入内容，这样会导致内存占用过大。

4、BufferPool 中的 insertTuple(TransactionId tid, int tableId, Tuple t)函数。

这个方法实现了将记录 t 插入到指定表中，要注意的是，将记录插入到某一页时，是插入到缓冲池的页，而不直接写入磁盘，等到需要写磁盘时才写入，尽量将磁盘读写操作减少和延后，这样有利于数据库的访问速度。