Write Up for acmdb-lab2

在 lab2 中，我们实现了数据库中页的置换算法，以及通过 B+树实现的 index 功能。

Exercise 1 (Page eviction)

lab 并未给出对具体的 eviction policy 的要求，此处实现的是 LRU 页面置换算法。因此在 LRUCache.java 中定义了该类，其主体为一个双向链表。

Exercise 2 (Search in B+ Tree)

这个 exercise 实现的功能是在 B+树中搜索，其逻辑为不断向左或向右寻找，直到找到索引符合的节点。

在 BTreeFile 中，主要有以下几个结点类：BTreeRootPtrPage 为 B+树的根节点，BTreeInternalPage 为 B+树的内部节点，BTreeLeafPage 为 B+树的叶子节点，BTreeHeaderPage 用于记录整个 B+树中的一个页面的使用情况。

Exercise 3 (Insert in B+ Tree)

B+树的插入流程主要如下：如果为空树则直接插入，否则找到插入结点的位置，并在叶子结点中插入值。如果当前结点的 key 过大，则均分成两个叶子结点，并添加中间结点的 key 加入父结点，这一操作需要递归执行。

Exercise 4 (Delete in B+ Tree)

B+树的删除逻辑主要如下：删除叶子结点，如果能从其兄弟结点那里偷窃，则偷窃过来并将兄弟结点的第一个数据作为父结点，如果此时结点能够与其兄弟结点合并，那么合并并删除兄弟结点，并删除其父结点指向该兄弟结点的部分，如果不能则检查父结点能否与其兄弟结点合并，递归执行。

Exercise 5 (Merging Pages)

合并的逻辑相对来说较为简单，在此不做赘述。

总结

lab2 实现的东西都是相对技术性较强的内容，需要对于其中的算法逻辑有一定的理解才能完成，在明白了其逻辑之后，编程相对来说会简单一些。此次 lab2 也让我对于这些数据结构有了更加深刻的体会。