实验报告

一、实验环境

Ubuntu 22.04
Jdk 1.8
Ant 1.10.12

二、实验流程

Exercise 1: Transactions, Locking, and Concurrency Control

我定义了一个 LockManager 类来包装与锁相关的同步方法，避免了 BufferPool 类中其他方法的干扰。

Exercise 2：Lock Lifetime

获取锁的设计较简单，但实现起来很麻烦。我是按照指南的建议，在页面级别实现了锁定。注意的点是将脏页刷到磁盘上时，要关注是否是同步代码块。

Exercise 3：Implementing NO STEAL

事务的修改仅在提交后写入磁盘。这意味着可以通过丢弃脏页并从磁盘重新读取它们来中止事务。因此，我们不能驱逐脏页。除非所有页面都是脏页，则引发 DbException。所以对之前写的驱逐策略进行一定修改即可。

Exercise 4：Transactions

在每个查询的开头创建一个 TransactionId 对象。此对象将传递给查询中涉及的每个运算符。查询完成后，将调用 BufferPool 方法 transactionComplete（）。若事务成功完成，需要将磁盘中的脏页全部刷新到磁盘，若事务失败时则需要回滚：将磁盘中的反向刷新到 BufferPool。最后释放掉事务所拥有的所有锁。

Exercise 5：Deadlocks and Aborts

死锁就是多个并发进程因争夺系统资源而产生相互等待的现象。因为超时等待比较简单实现，而且也适合这种 lab，所以采取的是超时等待：如果等待没有获取到资源一段时间后，自动放弃所拥有的资源。

运行结果：

ant test：

```
BUILD SUCCESSFUL
Total time: 33 seconds
```

ant systemtest：

```
BUILD SUCCESSFUL
Total time: 48 seconds
```

三、困难

此次的 lab 因为涉及到锁与同步概念，因为操作系统中也对这个概念有一定涉及，所以理解起来较简单。只是实现上面有一定难度，在写第一个 lab 的时候，一开始是直接在 ButterPool 里面定义锁的，然后就有各种问题，后面单独在外面定义了锁的类，终于解决了问题。大概一共花了一两天时间。