DataBase Lab5 Report

-521120910259
-郭培宇

1 Designing Part

在本实验中，我将在 SimpleDB 中实现一个简单的基于锁定的事务系统。 我需要在代码的适当位置添加锁定和解锁调用，还需要在代码的适当位置添加代码，以跟踪每个事务持有的锁，并在需要时向事务授予锁的代码。

1.1 Exercise1

以下是针对 BufferPool 中获取和释放锁的方法实现。假设我们使用页级锁定，我完成了以下内容：

修改 getPage() 方法，在返回页面之前阻塞并获取所需的锁。

实现 releasePage() 方法。该方法主要用于测试和事务结束时释放页面。

实现 holdsLock() 方法，使得逻辑可以确定一个事务是否已经锁定某个页面。

同时我定义了一个负责维护事务和锁状态的类。

1.2 Exercise2

确保在整个 SimpleDB 中获取和释放锁。一些（但不一定全部）需要验证正确运行的操作包括：

在 SeqScan 期间从页面读取元组（如果在 BufferPool.getPage() 中实现了锁定，只要 HeapFile.iterator() 使用 BufferPool.getPage()，这应该能正确运行）。
通过 BufferPool 和 HeapFile 方法插入和删除元组（如果在 BufferPool.getPage() 中实现了锁定，只要 HeapFile.insertTuple() 和 HeapFile.deleteTuple() 使用 BufferPool.getPage()，这应该能正确运行）。

1.3 Exercise3

在 BufferPool 的 evictPage 方法中实现了必要的页面驱逐逻辑，同时未驱逐脏页面。

1.4 Exercise4

在此练习中我实现了 BufferPool 中的 transactionComplete() 方法。transactionComplete 有两个版本，一个接受一个额外的布尔参数 commit，另一个不接受。没有额外参数的版本应该始终提交事务，因此可以通过调用 transactionComplete(tid, true) 来实现。

当提交事务时，应将与该事务相关的脏页面刷新到磁盘。当中止事务时，应通过恢复页面到其磁盘状态来还原事务所做的任何更改。

无论事务是提交还是中止，还应该释放 BufferPool 保留的与该事务相关的任何状态，包括释放该事务持有的任何锁。

1.5 Exercise 5

在这里在 src/simpledb/BufferPool.java 中实现死锁检测和解决。当发生死锁时，应该确保代码能够正确中止事务，通过抛出 TransactionAbortedException 异常。这个异常会被执行事务的代码捕获（例如 TransactionTest.java），该代码应该调用 transactionComplete() 来清理事务。

我的具体实现方法是

死锁检测数据结构：使用了 DependencyGraph 类来管理事务间的依赖关系，以进行死锁检测和解决。该类内部维护了两个 ConcurrentHashMap，分别记录了信号量边和事务边的信息，用于构建事务之间的依赖图。

死锁检测：DependencyGraph 类中的 wait()方法用于在事务等待锁的过程中进行死锁检测。该方法首先将事务及其等待的信号量加入到依赖图中，然后使用 BFS 算法检测是否存在环路，若存在环路则表示发生了死锁。

wait()方法在获取锁时被调用，如果检测到死锁，将抛出 TransactionAbortedException 异常，终止当前事务的执行。

死锁解决：当检测到死锁时，wait()方法会立即解除之前等待的锁，并抛出异常。

DependencyGraph 类中的 acquire()方法用于在成功获取锁后更新依赖图，release()方法用于释放锁时从依赖图中移除相应的边。

死锁解决的具体实现包括了事务的回滚和资源的释放，以确保系统能够继续执行其他事务。

代码中的应用：BufferPool 类中的 getPage()方法和 ReadWriteSemaphore 类中的获取锁方法都调用了死锁检测与解决的相关操作，确保了在并发环境下的安全访问。

在这一点上，我已经有一个可恢复的数据库，这意味着如果数据库系统崩溃（除了 transactionComplete()）或用户明确中止一个事务，那么系统重启后（或事务中止后）任何正在运行的事务的效果将不会可见。

2 实验心得

此次 lab5 完善了事务处理及死锁检测等任务，本次 lab 的难点可能是需要尝试一些额外的方法，以及进一步熟悉 java 对于异常处理的代码规范及处理方法，同时还需要我们回顾前四次 lab 的代码尤其是第一次中对于 bufferpool 中的代码流程，因此在思路构建方面就需要大量的复习之前代码或者看新代码的时间。同时此次编写操作符的一系列文件 debug 难度和复杂度也相较于之前有增加。

但是整体做完感觉收获还是非常多，我认为我对这 simpleDB 的底层架构到流程实现的认知又进一步清晰了。