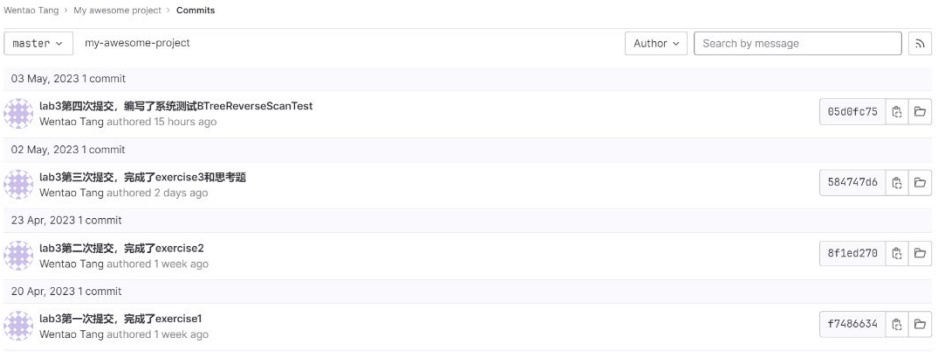
SimpleDB lab3 实验报告

一、Git Commit History



二、设计思路

Exercise1

findLeafPage() 函数递归地搜索内部节点，直到它到达对应于给定 key 的 leafPage。递归基为在传入的 BTreePageId 具有 pgcateg() 等于 BTreePageId.LEAF 的情况，表示它是一个 leafPage。这时，从 BufferPool 中获取该页并返回它。如果 key 为 null，则每次递归都在最左侧的子项上递归，以找到最左侧的叶子页。
每一步递归过程为，使用 BTreeInternalPage.iterator()迭代访问内部页中的 entry 并将值与给定的 key 进行比较。

Exercise2

为了保持 B+Tree 中 Tuple 的排序顺序并维护树的完整性，需要先调用 findLeafPage()函数找到应该插入 Tuple 的正确 leafPage。
根据 B+树的定义，尝试将 Tuple 插入到已满的 leafPage 中应该导致该页分裂，以便在两个新页之间平均分配 Tuple。每次 leafPage 分裂时，都需要向父节点添加一个新 entry，对应于第二页中的第一个 Tuple。有时内部节点可能已满并且无法接受新 entry。在这种情况下，父节点应该分裂并添加一个新 entry 到它的父节点。这可能会导致递归分裂，最终创建一个新的根节点。
实现 splitLeafPage()和 splitInternalPage()时，如果要分裂的页面是根页面，需要创建一个新的内部节点成为新的 root，并更新 BTreeRootPtrPage。否则，需要以 READ_WRITE 权限获取父页面，必要时递归分裂它，并添加一个新 entry。当内部节点被分裂时，需要使用 updateParentPointers()函数更新所有移动的子节点的父指针。此外，还需要更新任何已分裂的 leafPage 的兄弟指针。最后，根据提供的 key，返回应插入新 Tuple 或 entry 的页。

Exercise3

当从少于一半填充的 Page 中删除 Tuple 时，会导致该页从其 sibling 中获取 Tuple，或与其 sibling 合并。如果 page 的一个 sibling 有多余的 Tuple，则这些 Tuple 应在两个页面之间均匀分配，并相应更新父项的 entry。但是，如果 sibling 也处于最低占用状态，则应合并这两个页面并从父页中删除 entry。有时从 parent 中删除 entry 会导致 parent 的填充率少于一半。在这种情况下，parent 应从其 sibling 中获取 entry 或与 sibling 合并。这可能会导致递归合并或从根节点删除最后一个 entry。

实现从 sibling 中获取 entry 时，首先需要根据平均分配计算需要转移的 entry 数目，然后将这些 entry 从 sibling 中删去并插入当前 page，最后更新 parent 节点中对应的 entry。具体来说，转移叶节点中的 page 时，仅需在当前 page 与 sibling 中移动，最后将 parent 中对应 entry 的 key 更新为 entry 右孩子的首个 tuple 的 keyfield。当转移 internalPage 的 entry 时，需要将 sibling 的 entry 移至 parent，再将 parent 的 entry 移至当前 page。对于 stealFromLeftInternalPage()，stealFromRightInternalPage()，还需要更新已移动的 entry 的 parent 指针。
在 mergeLeafPages()和 mergeInternalPages()中，首先需要将右边 page 的 entry 删除并将其移至左边的 page，然后删去 parent 中对应的 entry，对于删去所导致的各种递归删除情况，均可用 deleteParentEntry()函数处理。最后调用 setEmptyPage()函数，使它们可以重新使用。

三、思考题

对于一个 BTreeScan 对象，他返回的 Tuple 又 IndexPredicate 决定。若 IndexPredicate 为空，则创建 BTreeFileIterator，通过 findLeafPaged 方法找到 BTree 中最左边的 Tuple，并通过叶节点的 sibling 指针向右迭代，逐个返回 Tuple。
若 IndexPredicate 非空，则创建 BTreeSearchIterator，根据其中的操作符 op 决定 scan 的 Tuple 范围。若 op 为>=、>、= 则从满足条件的最小 tuple 开始 scan，否则从最小的 tuple 开始，直到不满足 IndexPredicate 的条件

实现 BTreeReverseScan，就是对称地修改 BTreeScan 类。首先实现 findLeafPageReverse 以找到 scan 的起点，然后实现相应的 BTreeFileReverseIterator 和 BTreeReverseSearchIteratpr。最后在 BTreeReverseScan 中调用这两个 Iterator 即可。

四、重难点

1、Exercise1 中实现的 findLeafPage 方法为 peivate，不能直接调用。当实现 BtreeReverseScan 时，需要写一个私有的 findLeafPageReverse 方法以及公有的 findLeafPageReverse 方法，两方法参数不同，后者主要是为了创建一个新的 Hashmap 对象 dirtypage。
2、在从 sibling 获取 entry 时，需要先计算应该转移的 entry 数目，以达到平均分配。
3、dirtypage 参数用于记录操作中被修改的 page。在函数参数传递的过程中，Hashmap 对象为引用传递，函数内的修改对外部可见。