simpleDB实验三实验报告

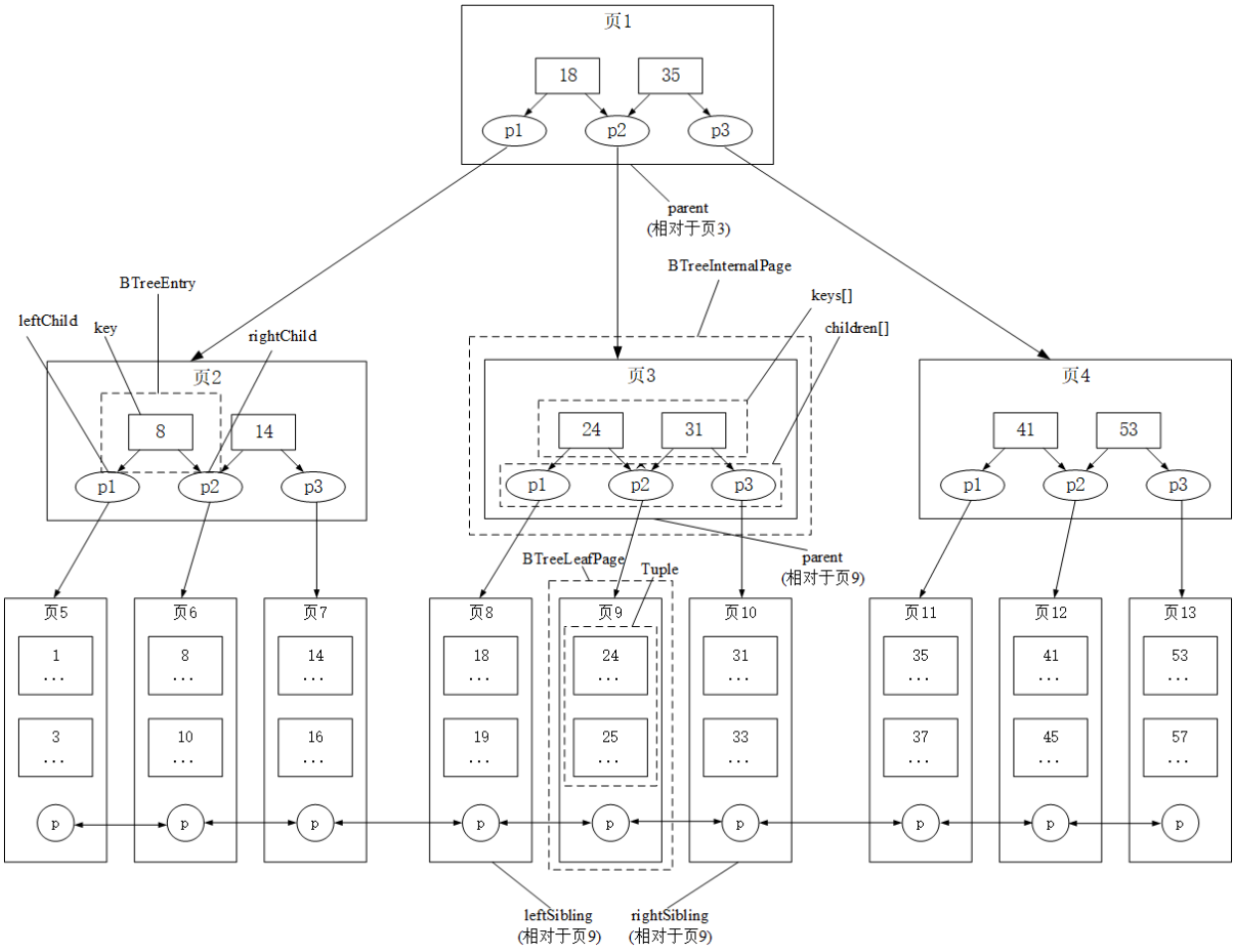
1. exercise1

1.1 设计思路

- `findLeafPage`方法：递归查找页面，递归结束的条件是查询到叶子节点。如果不是叶子节点，就查找符合条件的孩子节点，如果要比较的字段为空，则查找最左边的节点，如果不为空，则查找大于等于该字段的孩子节点。

1.2 重难点

- 判断是否为叶子节点，需要调用`pgcateg()`方法
- 获取最左边的孩子节点，需要调用`getLeftChild()`方法
- 比较字段大小，需要调用`compare(Op.GREATER_THAN_OR_EQ, f)`方法
- B+树的结构理解如下



2. exercise2

2.1 设计思路

- `splitLeafPage`方法：先将一半的数组存到新页中，然后判断原来的页是否有左邻居，如果有左邻居，则更新它的指针。接着更新分裂出的两页的左右邻居指针，并将中间的键值加入到父结点中，更新父节点指针，最后返回插入元组所在的页面。
- `splitInternalPage`方法：先将一半的条目（entry）存到新页中，然后更新分裂出的两页的左右孩子指针，并将中间的条目加入到父结点中，更新父节点指针，最后返回可以找到插入元组所在的页面。

2.2 重难点

`splitLeafPage`方法

- 通过`getLeftSiblingId`方法获得左邻居指针，再调用`setRightSiblingId`方法设置他的右邻居。
- 通过调用`getParentWithEmptySlots`方法获得还存在空槽位的父节点，如果他原本的父节点已经没有空槽位了那就会分裂他原本的父节点。
- 通过调用`updateParentPointers`更新分类后的这两个页的父结点指针。

`splitInternalPage`方法

- 分裂时要先在原来的页删除该条目，然后再在新的页中添加。
- 通过调用`getParentWithEmptySlots`方法获得还存在空槽位的父节点，如果他原本的父节点已经没有空槽位了那就会分裂他原本的父节点。
- 通过调用`updateParentPointers`方法更新分裂出的新页的孩子节点的父节点，和分裂出的新页的父节点。

3. exercise3

3.1 设计思路

- `stealFromLeafPage`方法：首先判断`isRightSibling`的值，如果为真，就调用`sibling`的正向迭代器，如果为假，就调用`sibling`的反向迭代器，移动元组数量之差的一半元组到该页中，重新设置父结点中对应条目的键值并更新叶子节点的父节点。
- `stealFromLeftInternalPage`方法：从左孩子节点中取出适当个数的数使得两个节点中的条目数量各占一半，所以需要移动的条目数量为两页中条目数量之差的一半。然后将父结点中的条目移动到右孩子节点中，再反向迭代左孩子节点，取出一定数量的条目移动到右孩子节点中。父结点中的对应条目应为最后一个移动的条目的键值。最后更新父节点指针。
- `stealFromRightInternalPage`方法：从右孩子节点中取出适当个数的数使得两个节点中的条目数量各占一半，所以需要移动的条目数量为两页中条目数量之差的一半。然后将父结点中的条目移动到左孩子节点中，再正向迭代右孩子节点，取出一定数量的条目移动到左孩子节点中。父结点中的对应条目应为最后一个移动的条目的键值。最后更新父节点指针。
- `mergeLeafPages`方法：正向迭代右孩子页面的元组，将元组依次删除并放入左孩子页面中。更新左右邻居指针，并删除父结点中对应的条目。
- `mergeInternalPages`方法：先将父结点中对应的条目放入左孩子页面中，然后正向迭代右孩子页面的条目，依次删除并放入左孩子页面中，将右孩子页面标记为空页面，更新这两个页面的父节点指针并删除父结点中对应的条目。

3.2 重难点

三种`steal`方法

- 都需要调用`updateEntry`更新父结点中原本的节点中的`children`数组。

两种`merge`方法

- 都需要调用`setEmptyPage`将其中删除完毕所有内容的页面标记为空，以便重复利用。

4. 思考题

4.1 设计思路

- 按照调用结构，修改每部分所调用的正向迭代器，反转他的逻辑，使其适合反向迭代。

4.2 重难点

- 新建一个`BTreeReverseScan`类，将`BTreeScan`类中的代码复制进去，将所有`iterator`改为`reverseIterator`。
- 在`BTreeFile`类里新增方法`findLastLeafPage`，将`findLeafPage`类中的代码复制进去，将`left`改为`right`，将`GREATER`改为`LESS`。
- 在`BTreeFile`类里新增两个方法`indexReverseIterator`和`reverseIterator`，分别返回`new BTreeSearchReverseIterator(this, tid, ipred)`和`new BTreeFileReverseIterator(this, tid)`。
- 在`BtreeFile`类里新增一个`BTreeFileReverseIterator`类，复制`BTreeFileIterator`的代码，将`getRightSiblingId`改为`getLeftSiblingId`，将`nextp`改为`prevp`。
- 在`BtreeFile`类里新增一个`BTreeSearchReverseIterator`类，复制`BTreeSearchIterator`的代码，将`getRightSiblingId`改为`getLeftSiblingId`，将`nextp`改为`prevp`，将`iterator`改为`reverseIterator`，将`GREATER`改为`LESS`，将`LESS`改为`GREATER`。
- 在`BTreeReverseScanTest`类中，将`BTreeScan`改为`BTreeReverseScan`。为了使元组反向排列，修改`compare`方法中的判断条件如下

```
if(t1.get(keyField) >= t2.get(keyField)) {
    cmp = -1;
}
else if(t1.get(keyField) < t2.get(keyField)) {
    cmp = 1;
}
```

5.提交记录