数据库系统概论期中报告

学号：2017011462　　姓名：方言

当前进展

此项目单人完成。

1. 文件系统（file system）

这部分修改了hash策略（原本的fileID + pageID在同时管理多文件时太容易发生冲突），使得可以比较好的同时管理多个文件。

2. 记录管理模块（Record Manager）

实现了完整的记录管理模块的功能。

这部分设计如下：

- 使用一个单独的文件作为记录文件，文件的第一页用于储存一些额外的信息，如空闲页面，页内偏移量等等。
- 每一页的开头储存当页的一些必要信息，使用bitmap记录当前页面内空闲的槽位。其余空间用于储存Record内容。
- 每个Record对应唯一标识RID，由文件页号pageID和页内的偏移量slotID组成。

每次插入时，通过第一页的信息即可找到一个有空闲槽位的页面，再通过页头的bitmap可快速确定偏移量，之后将对应记录写入。

为了保证寻找空闲页面的效率，将所有空闲页面组织成链表管理。每次插入记录占满一页时，将其移除链表，并将构造新的页面加入链表。而删除记录导致页面空闲时，也将其重新加入链表。

3. 索引管理模块（Index Manager）

基于B+树实现了完整的索引管理模块的功能。

这部分设计如下：

- 使用一个单独的文件作为索引文件，文件的第一页用于储存一些额外的信息，如字段的类型和长度等等。
- 以B+树的形式组织文件的页面，每一个页面作为一个节点。页头记录一些必要信息。
- 以非簇集索引方式构建，这样使得索引模块与记录模块是分离的。

- 对于重复key值，redbase上的做法是采用bucket页作为溢出页，储存某一个key对应的所有RID。这样保证叶子节点中的key是不重复的，但是空间效率低下（bucket页很可能只有少部分空间被使用，造成大量浪费），因此我允许重复key值在节点中出现，并且稍微修改了delete和search的逻辑。在叶子节点记录了指向兄弟节点的指针，因此如果要delete（或search）某个key-RID对应的entry，我的做法会找寻该key值出现的第一个叶子节点，之后通过兄弟节点指针依次遍历，这样可以找到这个key对应的所有RID。而redbase的做法会跳转到该key值对应的bucket页，之后在bucket页内进行遍历查找RID。因此我的做法时间效率和redbase相当，但空间效率会高很多。
- 对于联合索引，只需要在计算长度时将每个字段长度求和，比较两个key时，根据多个字段的优先级比较即可实现。

4. 测试

对记录管理模块和索引管理模块都有对应的测试用例，对于B+树，使用stl的map进行对拍测试，具体使用详见README

下一步计划

- 尽快完成系统管理模块和查询解析模块，并且着手实现扩展功能以及GUI界面。
- 更好地组织单元测试

参考资料

- Stanford RedBase Project: https://web.stanford.edu/class/cs346/2015/
- DBNoC Project: https://github.com/RecursionSheep/DBNoC
- B Plus Tree: https://github.com/begeekmyfriend/bplustree