B+tree

这个项目的主要目的是在你将要实现的数据库系统中增加索引，这样便可以在某列数据上建立索引并且实现快速的 index 操作。

在这次作业中，你需要实现磁盘上的 B+tree , 新的代码中已经重写了一个文件系统可以直接使用（src/file）。

提供参考文章：

http://web.archive.org/web/20161221112438/http://www.toadworld.com/platforms/oracle/w/wiki/11001.oracle-b-tree-index-from-the-concept-to-internals

具体要求

这里我们会关于实现的一些步骤和提示供你参考

逻辑上：

- 设计索引节点 B+tree pages : 你可以参考已有的 B+tree 框架，设计节点在文件中的存储格式。你需要设计 index pages 来存放 B+tree 的中间节点和子节点(建议从一个基类继承)。你需要根据节点大小计算一个 index page 可以存放多少数据。
    - B+tree pages 需要指明 page 类型(是否为叶子节点)，一个唯一的 page id 和父节点的 page id ，目前在节点中的 K-V pair 个数以及最大个数限制(与 page size 有关)。每次读写一个 B+tree page 的时候需要通过 page id 将它从磁盘上读上来并且
        - 对于中间节点，需要指向子节点的信息，并在任一时刻保持至少 half-full
        - 叶子节点需要存储每个 record 的 rid，以及指向兄弟节点的指针
- 与存储系统的交互，合适的序列化/反序列化格式：给定一个 Page 大小，你需要设计序列化/反序列化格式对 B+tree 节点进行存储。一个磁盘页可以存一个或者多个 B+tree 节点。举一个非常简单的例子，假如一个 10G 的文件中有 1000 个 disk page，第一个 page 用来存储 B+tree 的root 节点，其中包含很多关于子节点的地址信息，后面的空页用来存放子节点。随着 record 的插入，需要更多的子节点，开新的页面来存子节点。假如此时需要分叉，则下一个空页可以用来存放中间节点信息，关于子节点的信息从 root 那里拷贝到中间节点，root 转而维护中间节点地址信息。在读取的过程中：首先从 root 进入索引得到找到下一个要访问节点的地址信息（如 page, pageoffset 等）随后一直重复操作直到进入子节点。
    - 我们建议这一步进行认真的调试，会大幅提升后续代码的开发效率并减少杀虫机会
- B+tree structure：在暑假小学期中你们已经完成了包含范围查找的 in memory b+tree 的各种操作，可以参考它的实现诸如 insert，update，delete，并进行 split node 或者 merge node 的维护。请基于你的文件系统和你定义的节点格式对 on-disk B+tree 的功能进行维护，我们会在测试中进行正确性检查。
- 最后，需要你将 B+tree 按照 index.h 中给定的接口封装起来，我们会调用这些标准接口进行测试。（main.cc）

代码说明：

1. 文件系统：src/file，其中为你们实现了不同file类型，可以用来存 meta data, data record或者index。并且 file.cc 中实现了文件流的读写（目前没有 buffer manager）。不同种类的文件与对应的page需要进行对

接，如在 data_file.cc 的 api 的部分实现了对接，调用 data_page 中的 set, unset, has, get 等函数实现了增删改查功能。因此在写 B+tree 时也需要将 index_file 与 index_page 链接起来。

2. 文件的组织形式Page: src/page 分为几种，metadata page, data page 或者 index page。其中 metadata page 你们已经在 **lab2** 中实现并写到记录metadata的文件中。data page 已经写好，根据 header 计算 record 大小，并且计算一个 page 中可以存储的 record 数量，用一个 bitmap 来记录某个 slot 是否为空。我们建议在实现 B+tree 时，实现 index page 用来存储 B+tree node，并合理分配页内空间。
3. 索引结构：src/index, 这部分的 in-memory 功能已经实现，但需要你将它正确将其写到 B+tree node 中，存在 index page 中并写到 index file 中。(就像 table 中实现的增删改查功能被写入 data page 和 data file)

测试

请自行构造一些命令测试 B+tree 正确性。我们的测试包含在某个 attribute 上调用 build 函数建立索引(保证 key 唯一)并且在上面应用索引进行相关增删改查操作。正确性和性能都会作为评分标准纳入考虑，请在代码中加入相应注释，若有超出基本要求的优化(如支持 duplicate keys)请单独交一个报告写明。

Bonus: 根据 metadata 判断某一列是否加了索引，并且在先前 table 实现的 insert, update 基础上实现 insert_with_index , update_with_index