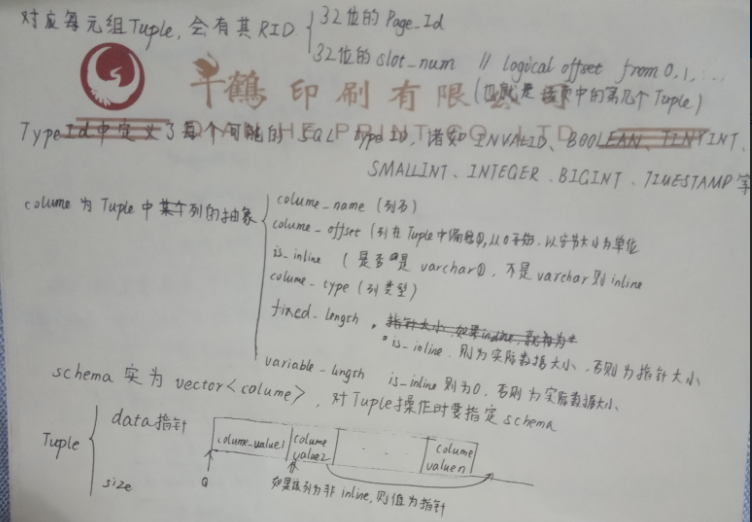
# 存储方式

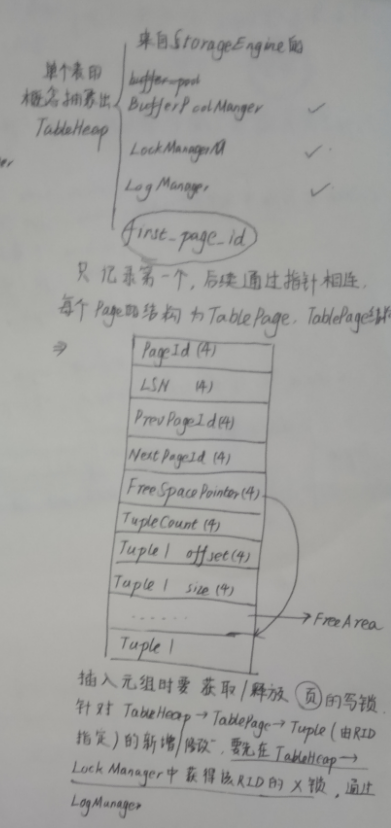
每元组Tuple的存储方式如下：

基本就是对着本表的schema来存，不过对于varchar，存的是4字节的指针，指向的位置紧挨着本元组

对于每个元组，其唯一标识为RID，实际就是存储的页id和在本页中的读几个元组

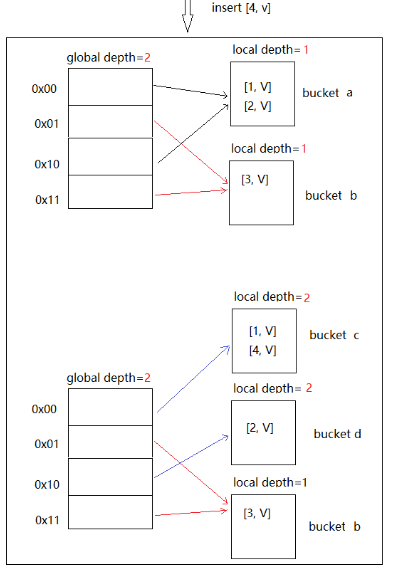


表在内存中的表示方式为TableHeap，他只要保存本表第一个页的页号，因为每个页都会有其相关前后页的页号。这里用于存储数据库记录的一个页，在内存中的表示方式为TablePage，其格式如下：

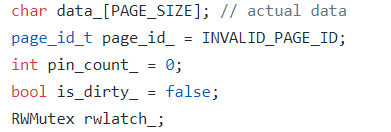


# 缓冲区管理

哈希表+LRU。这里的哈希表实现为可扩展哈希



**缓冲区管理的单位是页**，Page，包括读写锁、脏写位、引用位、页ID等，其他的包括用于存储数据库记录的页、用于索引的页（b+ tree相关页）都继承自该类。

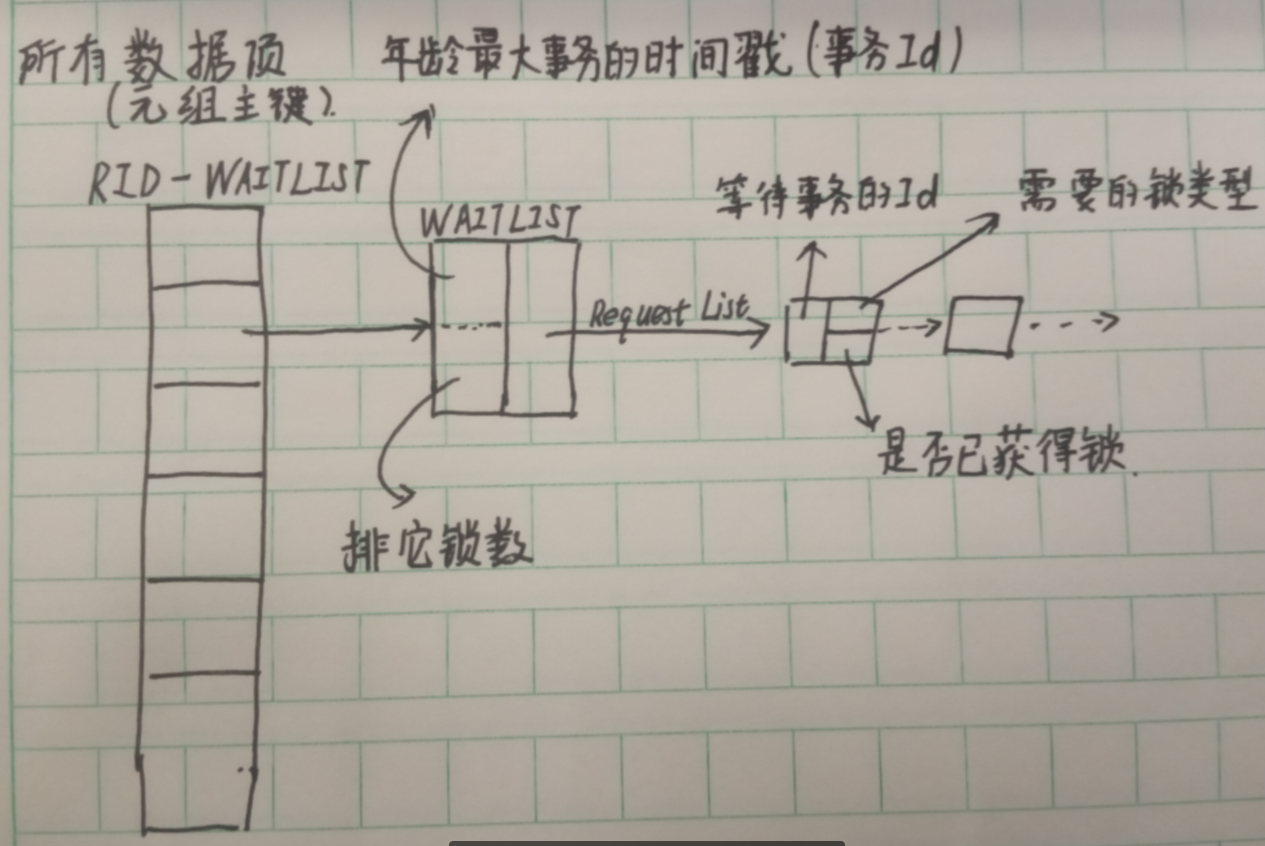


其他的页在实际使用时只用到data\_部分。

# 索引

实现为B+树，具体代码在Page下，前缀为B\_Plus\_Tree的几个文件，区分开根页、中间页、叶子页

# 2PL



# 日志

## 日志管理

日志管理是通过log\_manager，通过disk\_manager读写日志。log\_manager在构造时创建　日志刷新线程，每当有日志写入或超时或强制唤醒时　会刷新缓冲区日志到持久化存储中。

日志管理会持有一个当前日志计数，每次新增日志，计数++并作为日志id

## 写日志

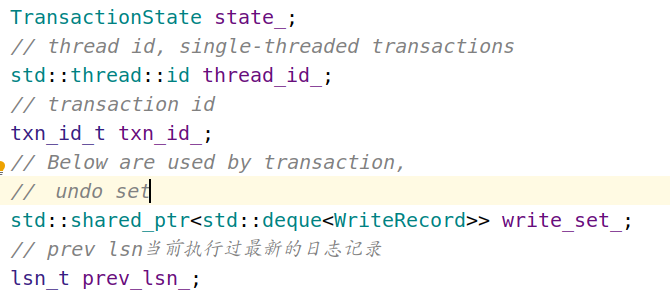
具体就是在每次table\_page中更新元组时，新建并插入相关日志，格式如下

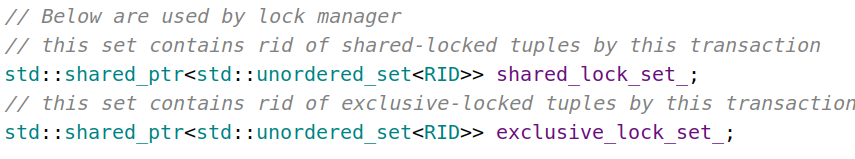


每个日志会保存在本事务中的上一条日志的id，便于undo时操作

## 事务及事务管理

事务主要包括





事务管理器负责管理事务计数，事务状态改变（begin, commit, abort）

IMG_259

加锁阶段和解锁阶段和锁管理交互

## 日志恢复

日志记录读取单位还是一页，从日志第一页/检查点开始读，第一遍redo日志，并记录未完成的事务(刚开始所有事务都是未完成，commit/abort后才会从里面删掉)。

对于日志记录重做，如果要更新的页(其lsn大于等于本日志记录lsn，则不用更新)

undo从每个未完成事务的最后一个日志开始，为了加快索引，引入

