# simpleDB lab1 实验报告

### 唐文涛

#### March 20, 2023

## Contents

| 1 | Git Commit History | 1 |
|---|--------------------|---|
| 2 | 设计思路               | 1 |
| 3 | 重难点                | 3 |
|   | 3.1 HeapPage 类     | 3 |
|   | 3.2 HeapFile 类     | 3 |
|   |                    |   |

## 1 Git Commit History

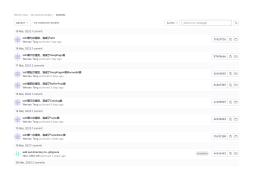


Figure 1: Git Commit History

# 2 设计思路

TupleDesc 是 TDItem 数组, TDItem 对象由 Type 对象 fieldType 和字符串对象 fieldName 组成。

Tuple 是 Field 数组, Field 接口可实现为 IntField 和 StringField。Tuple 类成员变量 TupleDesc 定义了 Tuple 的模式。如图 2(a) 所示

Catalog 是储存所有表信息的记录,用数据结构 ConcurrentHashMap 实现,便于查找。其中每个表的信息以 Dbfile 接口形式记录,一个 Dbfile 记录一张表,对应一个 TupleDesc. 如图 2(b) 所示

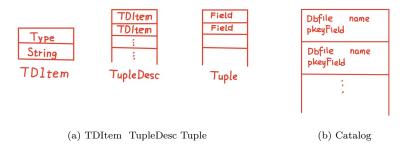


Figure 2: 概念示意图

BufferPool 基于 ConcurrentHashMap 储存了部分活跃的 page,可以通过 PageId 向 BufferPool 中请求 page。若目标在 BufferPool 中则直接返回; 否则需要从磁盘中读取。如果此时 BufferPool 已满,则抛出 DbException; 否则通过 PageId 的 TableId 向 Catalog 请求对应的 Dbfile,再从 Dbfile 中读出目标 page。如图 3 所示

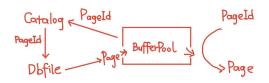


Figure 3: BufferPool

HeapPageId 记录了 HeapPage 对象所在的表 (tableId) 以及在表中的位置信息 (pgNo), RecordId 记录了一个 Tuple 所在 Page (pid) 以及在 Page 中的位置信息 (tupleNo)。

HeapPage 通过 byte 数组储存了一些 Tuple 以及相应的 header。

HeapFile 连续储存了一系列 HeapPage。读取时通过 RandomAccessFile 方法,在指定位置读出 Page。位置 OFFSET 通过 PageNumber\*PageSize 获得。如图 4 所示

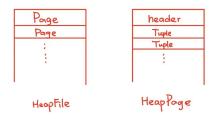


Figure 4: HeapPage HeapFile

SeqScan 对象实现了对给定 Dbfile 中 Tuple 的扫描, 其功能主要依靠成员变量 DbFileIt-

erator 实现。getTupleDesc() 方法需要获取类成员 DbFile 的 TupleDesc, 并据此构造一个新的 TupleDesc。

# 3 重难点

#### 3.1 HeapPage 类

计算 numSlots 时,利用 int 型除法向下取整的性质,可直接用一个 Page 可用空间除以每个 Tuple 所需空间 (加上 1bit 的 header);而计算 HeaderSize 时,应避免 int 型除法向下取整性质,采用 ceil(getNumTuples() / 8.0)。

实现 getNumEmptySlots() 时,考虑用 slots 总数减去 header 中 valid slots 的数量。通过对 header 数组中每个 byte 进行循环移位,再对 2 取模统计 valid slots 数量,其中特别关注最后一个 byte 未满的情况 (即最后一个 byte 未存满,部分数值无效)。因为一个 byte 中,低比特位记录了 page 中更靠前 slot 的信息,故此时只需进行 numSlots%8 次移位取模操作。在判断一个 slot 是否为空的函数 isSlotUsed 中,同样对相应 byte 进行循环移位取模操作,来判断相应位置是否为 1。

#### 3.2 HeapFile 类

实现 DbFileIterator 时,定义辅助类 HeapFileIterator 并加入成员变量 Iterator<Tuple> it,用于对 HeapPage 的遍历。

实现 HeapFileIterator 的 open 函数时,用当前 file 的首个 page (pageNo=0) 建立一个 PageId 对象 pid,再通过 pid 在 Database 的全局 BufferPool 中取出对应 page,通过该 page 初始化 it。

实现 HeapFileIterator 的 hasNext 函数时,需要分情况处理: 当 iterator 没有 open 时 (即 it 为空值),返回 false; 当 it.hasNext() 为 true 时,返回 true; 否则确定当前 page 已无有效 tuple,继续搜索接下来的 page,直到找到有效 tuple 或遍历所有 page。