Hash存储引擎

设计目标

- 数据持久化到磁盘中
- 支持快速精确查找
  - 内存效率 O(1)
  - 访问磁盘1次
- 支持内存缓存
- 插入效率要高
  - 异步插入
  - 重做日志
- 支持断电恢复

概要设计

参照Bitcask设计，并进行简化

内存中使用HashMap存储索引

磁盘中存在一个顺序的数据文件。

当有更新或者删除时，不删除原来数据。也不标记旧数据为删除，只的创建一个新的数据并将版本号+1即可。

为了解决数据文件垃圾数据过多，当达到某些条件时，创建一个新的文件，写操作转移到该文件，旧文件有效数据写入新文件，最终删除旧文件

内存管理

采用固定式内存管理，主要内存由如下几个部分构成

- HashMap大小
- 缓存内存大小

缓存方案

参见《2-LRU缓存》，重用其实现。需要用户指定占用内存大小

HashMap

由于Key的尺寸可变所以需要自己实现一个内存索引结构，即HashMap

结构为 `<key，value>`

- key 为 主键
- value 为 在磁盘中的位置等信息
  - 版本号
  - 在磁盘中的位置

文件结构

数据文件

- 顺序文件
- 前4个字节为魔数（magic），值为 `0x960729ab`
- 随后的四个字节为版本号（version），目前值为 `0x1`
- 由元组构成 `<版本号，主键长度，值长度，键，值>`
- 当主键文件 `值长度` 为0，表示该数据被删除，等效于不存在
- 以上的出列 `键` 和 `值` 外，其他类型存储都为网络字节序存储（大端）

基本操作

提供增删改查操作

插入或更新

- 查找 读缓存
  - 若找到，从读缓存中移除；在 写缓存 中创建一个 (版本号+1，value=newValue) 的记录
- 查找 写缓存
  - 若找到， (版本号+1，value=newValue)
- 查找HashMap
  - 找到其位置，读取记录，在写缓存中创建一条 (版本号+1，value=newValue) 删除记录
- 将HashMap中这一项标记为 在内存中
- 记录重做日志（带版本号）
- 返回成功

删除

- 记录到重做日志
- 查找 读缓存
  - 若找到，从读缓存中移除；在 写缓存 中创建一个 (版本号+1，valueLen=0) 的删除记录
- 查找 写缓存
  - 若找到， (版本号+1，valueLen=0)
- 查找HashMap
  - 找到其位置，读取记录，在写缓存中创建一条 (版本号+1，valueLen=0) 的删除记录
- 返回成功

查找

- 从读缓存中找
  - 若找到，返回
- 从写缓存中找
  - 若找到，则返回（状态不为删除）
- 从HashMap中确定其在内存中的位置，从磁盘中读
  - 放入 读缓存
  - 返回

辅助操作

持久化操作

- 原子的从 `写缓存` 中拿数据，写入磁盘，确定写入完成后，原子的从 `写缓存中删除` 并创建或修改HashMap中的索引

正常停止

- 等待所有持久化线程结束
- 将HashMap写入磁盘，创建索引文件（可选）
- 程序退出

正常启动

- 读取索引文件，创建HashMap
- 完成

断电重启

- 遍历整个数据文件，创建HashMap，针对每一个项，插入或更新HashMap
- 执行重做日志
- 完成

垃圾回收

当 `无效项数/总项数` 达到一定阈值时执行垃圾回收

- 创建新的文件
- 新的写请求写入新文件
- 创建新的HashMap
- 旧的HashMap中的数据搬移到新的HashMap中，并放入写缓存