# 程序设计与数据结构2018年暑期课程设计

# 简易数据存储系统

发布：2018年7月2日   
 提交：2018年7月28日 (周六)

答辩：2018年7月29日（周日）

## 概述

“程序设计与数据结构”暑期课程设计，强调数据结构的选择和对性能的分析。涉及查找、树、散列等操作。目的在于让同学们利用所学的知识，设计并构建一个简易的键值对数据存储系统，进一步理解和掌握基本的数据结构和算法。

## 功能要求

你需要完成对数据的增删改查。每一组数据包含一个键(key)和其值(value)，对数据的增删改查的接口可以参考哈希表/树的实现。以下给出了一个例子，你可以不按照例子来设计接口：

class database {

public:

// For query

string get(const int key);

string[] getRange(const int key1, const int key2);

// For insertion and modification

string put(const int key, const string& value);

// for remove

string remove(const int key);

};

你实现的存储系统需要将每组数据的内容保存在磁盘中，并且能够根据名字查找对应的内容，进行查找（支持**单键查找**和**范围查找**）、删除、修改操作。范围查找指的是查找在[key1， key2]之间的数据（闭区间）。数据存储、索引的形式有较大的灵活性，你可以选择合适的数据结构进行实现。所以你需要在最后提交的文档中描述你使用了怎样的数据结构，效果如何；以及磁盘中数据和索引的组织格式。

**注： 你需要采用C++类及面向对象设计。不能采用传统的C的结构化设计风格来组织代码。**

### 数据文件

数据文件存储所保存数据的内容，保证下一次运行时能够从磁盘文件中读取。最简单的方式就是把每组键值对依次排列在一起。数据文件的格式可以自行设计，包括文件头、数据描述信息等。一般而言这类数据使用二进制格式进行存储。

### 索引文件

除了数据以外，还需要考虑对数据进行索引，以便根据键(key)能快速找到对应的数据内容。通过索引可以快速定位数据在文件中的位置并获取相关数据。

你可以在磁盘上采用树、散列等数据结构保存索引信息，具体实现自行设计，自由度较大。

你可以将全部索引保存在一个文件中，或者保存在多个文件中，甚至采用二级索引以便减小单个索引文件的大小。数据文件你也可以将其全部保存在一个文件中，或者保存在多个文件中。但不允许以每组数据（名字和其内容）为单位单独保存在一个文件中，索引和数据文件需要分开。

### 缓存与一致性刷新方法设计

相比于内存读写，磁盘读写的速度是很慢的。所以系统应该尽可能多的访问内存。这时系统把内存当作一个缓存。例如当索引文件被读取到内存后，接下来对索引文件的访问就变成了内存访问。对于数据文件也有相同的道理，当读取一条数据时，系统把它放在内存中，接下来的读写操作都变成了内存操作。当然系统不可能把所有的数据都读到内存中来（那样就变成了内存型数据库），当内存中缓存的数据变过多时，系统应该把一些数据写回到磁盘以释放内存空间。

缓存的问题在于系统运行时不能够将所有的数据存放在内存中，因此一份数据可能同时出现在磁盘和内存中。你需要考虑何时对文件进行读写以便保持文件的一致性，且保证运行效率。

常见的方法有两种，一种是在每次（写）操作后立刻把数据写入磁盘，另一种是新增一个flush接口。只有当用户显式调用flush操作，flush之前的写操作才会被写入磁盘。前者提供较强的一致性，但性能较差，而后者的一致性保证较弱，但性能较好。当然，当缓存空间不足时，也需要把一些数据写回到磁盘。

### 初步的软件工程考察

本次课程设计要求体现基本的软件工程方法，能够对关键代码进行测试（如单元测试或系统测试），在类的设计和接口的设计上考虑代码的可读性、可维护性。

## 系统测试

关于测试的程序，你的测试应该独立于实现的数据存储系统，并且应该直接针对该系统的API进行编写，不需要通过读取测试脚本等外部IO方式进行测试。

具体的测试内容可以自行设计，至少包括增删改查操作。可参考附录[1]中的测试。

你需要为你的系统编写一个简易的UI（命令行即可），以便在答辩的时候给助教演示。

### 正确性测试

正确性测试既包括结果正确没有引发程序的异常退出或内存泄露，也包括符合预期的性能要求。你需要自行书写测试内容，测试数据可以用使用一些随机数据生成工具。

测试应涵盖程序的各个操作，并且应当保证数据量不少于**100万**条。

### 性能测试

性能测试与正确性测试要求类似，但注重考查程序的性能。同样需要进行不少于100万数据量的测试，可以记录不同数据量的操作所耗费的时间，不同数据规模时进行同样的操作所耗费的时间，相同数据规模时进行不同的操作所耗费的时间。并使用相关绘图软件绘制时间-数据量的散点图或者拟合图，并对结果图进行分析解释。

对于性能测试所进行的操作，一个直接的想法就是编写一段代码，一次遍历所有的数据，对每一条数据进行几次读写操作。但在现实中这样的数据访问模式并不常见，更多的情况是有少数数据被经常访问，而大多数数据很少被访问。以新浪微博为例，大多数用户的粉丝数量是0，而极少数用户（如鹿晗和关晓彤）的关注量非常高。这也是缓存的意义所在，让少数频繁访问的数据获得较好的性能。你应该编写一些具有这样特征的测试用例。

注意，由于每个人的机器配置不一样，和其他人比较性能是没有意义的，我们也不会根据你的性能进行评分，我们关注的是你通过测试可以证明程序符合设计的预期，并且通过测试可以发现在使用和不使用索引的性能差异，以及不同数据结构的性能差异。

提交作业时，需要包含测试文件和测试报告，测试报告应描述你的测试过程，测试结果图和对测试结果的分析。

## 其他限制要求

### 自行实现关键数据结构

程序中使用的关键数据结构，例如散列、B树需要自行实现，文件的一致性刷新也需要自行实现，不可使用任何已有的库（基本的C库等除外）。

### 函数库的使用

不涉及关键数据结构的部分，可以随意使用你喜欢的库进行开发。注意不要实现过多无用的功能而忽视了作业要求的部分，这对你的成绩不会有任何有益的影响。

### 开发和编译环境

程序需要用C++实现，答辩时需要大家演示自己的系统，程序为单机程序，不需要实现多线程、网络通讯等功能。不需要实现GUI界面（仅命令行界面）。

## 检查

你需要在小学期结束前完成了全部要求，到时通过答辩的方式进行检查。

如果你提前完成全部要求，请与助教联系，我们安排检查。

具体的检查安排，包括检查地点，会在检查前公布在课程网站上，请留意。

## 评分

满分100分中，数据库系统的实现占70分，测试和文档占30分。

如果你的索引实现了B树等较为复杂的数据结构，或者在某一部分使用了较为高级的算法和数据结构，数据库系统的实现部分的70分都可拿到满分。如果只使用了散列表作为索引，功能上的增删改查做到了，保证了正确性，这部分至少能拿60分。其他情况的得分视实现的难度和合理性而定。

测试和文档部分实现了正确性测试，性能测试描述中的部分，并且书写了详细的描述实现和解释测试结果的文档，能够得到25分以上。具体得分视测试和文档的覆盖面和完整性而定。

## 提示和建议

一般来说，测试和修复错误的时间会占到开发总时间的一半以上，尤其涉及到一些较为复杂的算法时测试所占的比重往往会更多。所以请尽早着手完成作业，注意合理安排好自己的时间。

考虑使用二进制文件读写而非文本文件读写，这样既可以提高程序效率，又可以降低开发难度。

虽然我们不允许你直接使用已有的库来实现系统中关键的数据结构，但建议你查阅相关资料，参考已有的数据结构而非从头开始自己的设计。例如你可以去了解数据库系统都是用了怎样的方式管理数据，虽然数据库系统可能有很多较为复杂而我们不需要关心的高级话题（如并发控制），但对于数据的存储和索引，这些设计对我们很具有参考价值。

建议你在开始书写程序之前就对你要使用的数据结构有所设想，以避免在开发过程中，因为要使用某种数据结构而出现对已经完成的代码大量的返工。

## 提交及Q&A

请在检查之前将你的代码和文档提交到课程网站上。

如果有其他疑问，请在讨论区上提出，或者联系助教。

## 参考及附录

1. 数据库函数库
2. 项目文档参考：HXHYDatabase开发文档
3. JPL Coding Standard
4. 华为编程规范
5. 答辩模板
6. SQLite: http://www.sqlite.org/
7. Oracle Berkely DB: http://www.oracle.com/technetwork/database/berkelydb/index.html
8. Google C++ Style Guide: http://www.google-styleguide.googlecode.com/svn/trunk/cppguide.xml