数据库作业 记录管理模块 报告

计 65 赖金霖 2016011377

一、使用接口

使用记录管理模块需要包含 RecordManager.h 文件并定义 RecordManager 类的对象，如下所示：

```
RecordManager* rm=new RecordManager();
```

RecordManager 有许多函数，定义和说明如下：

```
void createFile(const char* name,int recordLength);
//用于创建文件，并在创建文件时指定记录长度（须是4的倍数）
bool deleteFile(const char* name);
int openFile(const char* name);
//返回fileIndex
void saveFile(int index);
bool closeFile(int index);
int insertRecord(int fileIndex,const char* record);
//记录以字符数组的形式传递
bool deleteRecord(int fileIndex,int recordIndex);
//以fileIndex和recordIndex为索引删除记录
bool renewRecord(int fileIndex,int recordIndex,char*
newrecord);
std::vector<char*> filterRecord(int fileIndex,int begin,int
end,const char* provided,RecordFilter* checker);
//意思是在fileIndex代表的文件中查找满足以下条件的记录：
//从第begin到第end-1个字符的子串与provided字符数组通过checker匹配
//checker可以为
EqualFilter,GreaterFilter,LowerFilter,RegexFilter之一
//使用方式如：std::vector<char*>
getter=rm->filterRecord(file1,0,4,"00..",new RegexFilter());
//返回值为一个内容为char*的vector，char*数组长度为recordLength+4，前
四位表示recordIndex的值，之后为记录数据，这样设计是因为比较方便
//可以传入返回的char*通过下面的parseIndex函数获得记录的recordIndex
uint parseIndex(const char* ch);
void outpmsgs();
//用于调试
```

二、具体实现

1、文件管理

程序在openFile时直接返回了文件系统提供的index，然而，程序不知道index的范围，却要对每个index存储一些文件信息，并根据index对文件进行操作。我通过一个hash表（NotThatSillyHashMap类）将index映射到fileInfo数组的下标，而fileInfo各位置分别是一个文件的信息，包括页数量、记录长度、记录数量、index、下一条记录将被分到的recordIndex。

2、插入记录

插入新记录的位置在当前所有记录之后，程序先计算出新纪录的坐标（页及

offset），然后新建或访问对应页，将记录写入对应位置，并更新文件信息。此时文件未保存，只有调用 saveFile 或 closeFile 函数程序才将文件写回。

3、删除记录

删除记录的实现比较 tricky，我的做法是用最后一条记录覆盖被删除的记录，并删去最后一条记录。这样做的好处是效率高（O(1)），且删除最后一条记录的代码相对简单。删除文件时对 fileInfo 数组的处理也类似。在删除记录时，为了方便，此 recordIndex 将永远不再被使用。

4、查找记录

查找记录时，程序会根据记录数量枚举记录的位置（页和 offset），对每条记录，把它的 begin、end 之间的子串和调用者提供的 provided 数组传给调用者提供的 checker，由 checker 的 check 函数判断记录是否满足条件，checker 的类的基类定义如下：

```
class RecordFilter{
public:
  virtual bool check(uint* tocheck,const char* provided,int
len1,int len2)=0;
};
```

三、扩展性

在以后的部分中，需要快速找到满足条件的 recordIndex，还需要通过 recordIndex 快速找到位置。这里的第一部分可以通过某些数据结构维护（内部顺序独立于文件写入顺序），第二部分也可以用一个 Hash 表来维护。

另外，如果需要其他的 RecordFilter，可以通过继承现有的抽象类来实现。