# 浙江大学


题　　目： MINISQL 设计总报告

组　　员：3120101973 陆洲 计科 1204

组　　员：3120102057 尹依婷 软工 1201

课程名称：数据库系统设计

授课教师：孙建伶

年　　级：2012 级

# ———————————— 目 录 ————————————

# 第 1 章 MINISQL 总体框架

设计并实现一个精简型单用户 SQL 引擎(DBMS)MiniSQL，允许用户通过字符界面输入 SQL 语句实现表的建立/删除；索引的建立/删除以及表记录的插入/删除/查找。

## 第 1.1 节 MiniSQL 实现功能分析

### 1.1.1 数据类型

只要求支持三种基本数据类型：int，char(n)，float，其中 char(n)满足 1 <= n <= 255 。

### 1.1.2 表定义

一个表最多可以定义 32 个属性，各属性可以指定是否为 unique；支持单属性的主键定义。

### 1.1.3 索引的建立和删除

对于表的主属性自动建立 B+树索引，对于声明为 unique 的属性可以通过 SQL 语句由用户指定建立/删除 B+树索引（因此，所有的 B+树索引都是单属性单值的）。

### 1.1.4 查找记录

可以通过指定用 and 连接的多个条件进行查询，支持等值查询和区间查询。

### 1.1.5 插入和删除记录

支持每次一条记录的插入操作；支持每次一条或多条记录的删除操作。

## 第 1.2 节 MiniSQL 系统体系结构

本小组设计的 MiniSQL 系统，除了实验指导书中规定的模块之外，还添加了一个 Basic 基类来支持所有的其他类。其余的设计和实验指导书中一致。

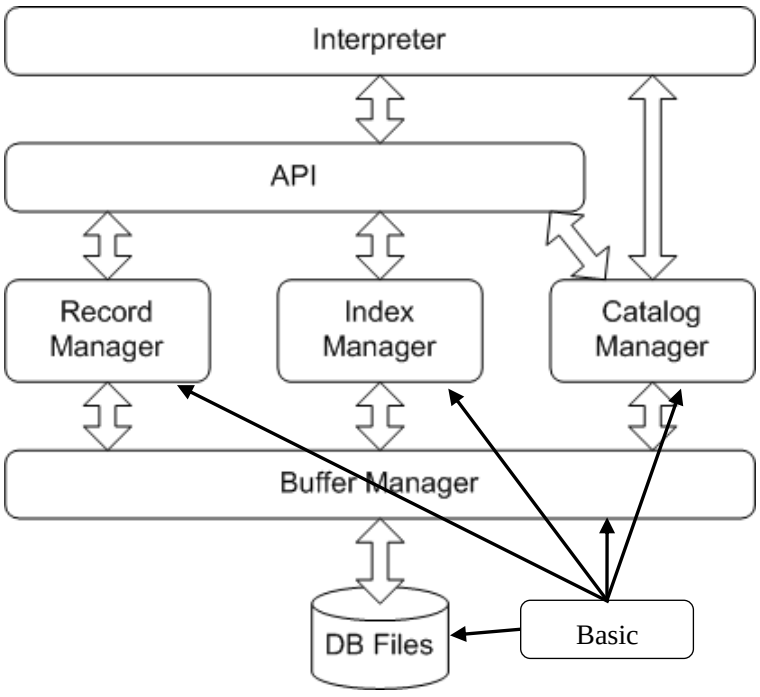



图 1 MiniSQL 体系结构

## 第 1.3 节 设计语言与运行环境

设计语言：C++
开发环境：Visual Studio 2013
操作系统：Windows 7

# 第 2 章 MINISQL 各模块实现功能

## 第 2.1 节 Interpreter 实现功能

Interpreter 模块直接与用户交互，主要实现以下功能：

1.程序流程控制，即“启动并初始化->‘接收命令、处理命令、显示命令结果’->循环->退出”流程。

2.接收并解释用户输入的命令，生成命令的内部数据结构表示，同时检查命令的语法正确性和语义正确性，对正确的命令调用 API 层提供的函数执行并显示执行结果，对不正确的命令显示错误信息。

## 第 2.2 节 API 实现功能

API 模块是整个系统的核心，其主要功能为提供执行 SQL 语句的接口，供 Interpreter 层调用。该接口以 Interpreter 层解释生成的命令内部表示为输入，根据 Catalog Manager 提供的信息确定执行规则，并调用 Record Manager、Index Manager 和 Catalog Manager 提供的相应接口进行执行，最后返回执行结果给 Interpreter 模块。

## 第 2.3 节 Catalog Manager 实现功能

Catalog Manager 负责管理数据库的所有模式信息，包括：

3.数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。

4.表中每个字段的定义信息，包括字段类型、是否唯一等。

5.数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。

Catalog Manager 还必需提供访问及操作上述信息的接口，供 Interpreter 和 API 模块使用。

## 第 2.4 节 Record Manager 实现功能

Record Manager 负责管理记录表中数据的数据文件。主要功能为实现数据文件的创建与删除（由表的定义与删除引起）、记录的插入、删除与查找操作，并对外提供相应的接口。其中记录的查找操作要求能够支持不带条件的查找和带一个条件的查找（包括等值查找、不

等值查找和区间查找）。

数据文件由一个或多个数据块组成，块大小应与缓冲区块大小相同。一个块中包含一条至多条记录，为简单起见，只要求支持定长记录的存储，且不要求支持记录的跨块存储。

## 第 2.5 节 Index Manager 实现功能

Index Manager 负责 B+树索引的实现，实现 B+树的创建和删除（由索引的定义与删除引起）、等值查找、插入键值、删除键值等操作，并对外提供相应的接口。

B+树中节点大小应与缓冲区的块大小相同，B+树的叉数由节点大小与索引键大小计算得到。

## 第 2.6 节 Buffer Manager 实现功能

Buffer Manager 负责缓冲区的管理，主要功能有：

6.根据需要，读取指定的数据到系统缓冲区或将缓冲区中的数据写出到文件

7.实现缓冲区的替换算法，当缓冲区满时选择合适的页进行替换

8.记录缓冲区中各页的状态，如是否被修改过等

9.提供缓冲区页的 pin 功能，及锁定缓冲区的页，不允许替换出去

为提高磁盘 I/O 操作的效率，缓冲区与文件系统交互的单位是块，块的大小应为文件系统与磁盘交互单位的整数倍，一般可定为 4KB 或 8KB。

## 第 2.7 节 DB Files 实现功能

DB Files 指构成数据库的所有数据文件，主要由记录数据文件、索引数据文件和 Catalog 数据文件组成。

## 第 2.8 节 Basic 基类实现功能

在 MiniSQL 中，其他模块均有一部分功能要实现，而他们每个功能使用的数据单元在 C++中并没有可直接使用的定义。这可能造成混淆和麻烦。我们小组在开发过程中，感到首先需要对数据库常见的概念给予一个统一的定义，因此有了基类模块。

它主要规范基本数据单元的使用，包含属性、记录、表格属性等这些每个其他模块都会使用到的类。

# 第 3 章 各模块总体架构及提供的接口

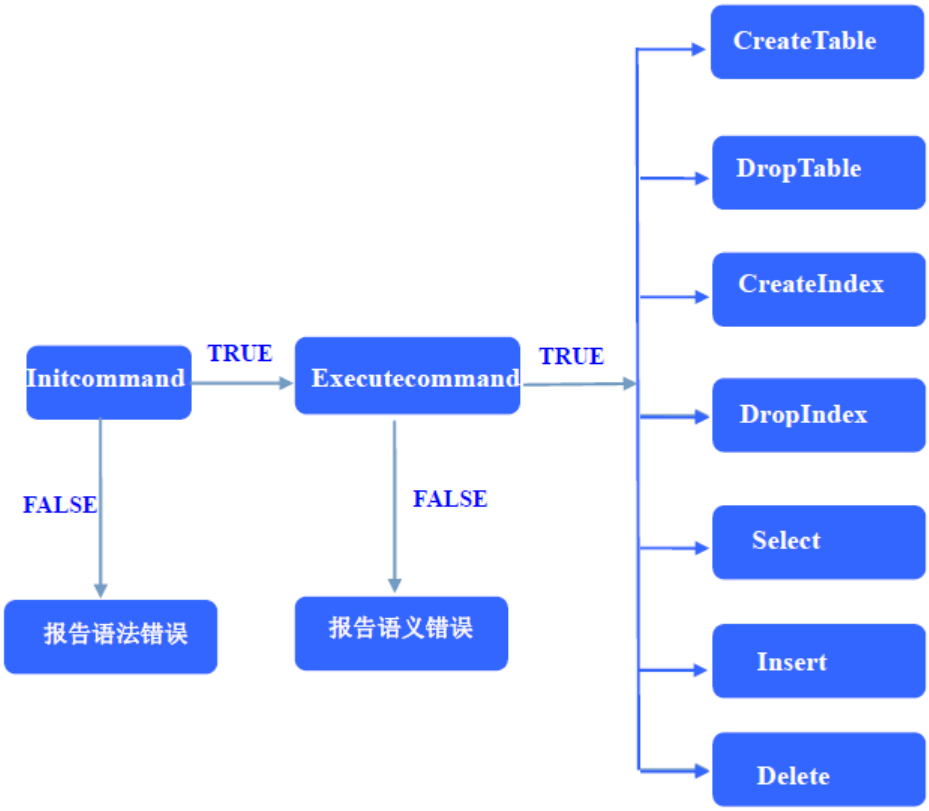
## 第 3.1 节 模块总体架构

在 MiniSQL 内部，底层主要用数组来储存数据，在上层数据主要是以 Record 的形式

存在。在进行数据交换时的单位是块，大小为 4KB。

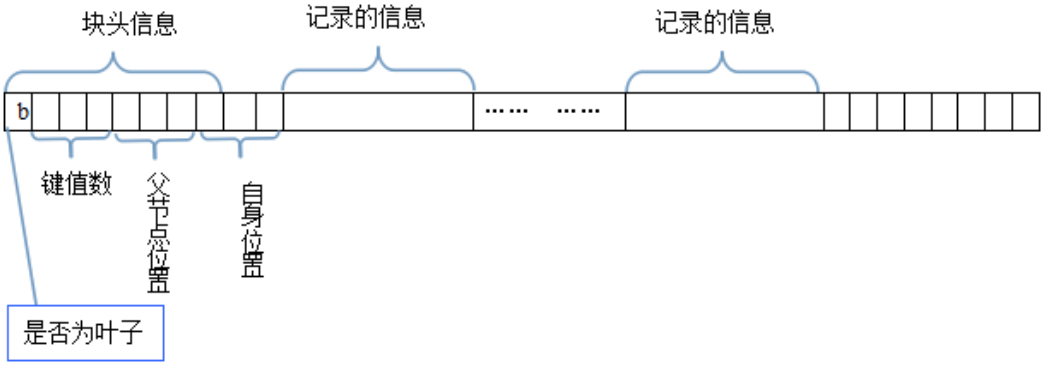
Interpreter 类

Interpreter 类以流程图的操作模式，读入用户命令以后，对命令进行解析，包括语法上和语义上的分析，命令语法和语义都正确以后，从 API 中调用相应的函数。

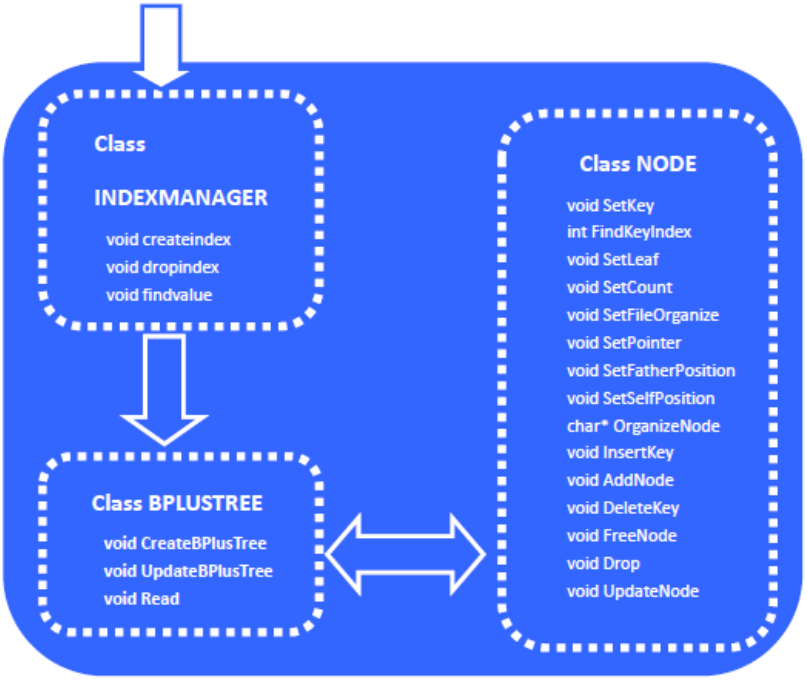



IndexManager 类

Index 的实现主要依赖与 B+树的实现，首先构造结点数据结构，也是最主要的数据结构，它包含了诸多信息：是否为叶节点、父节点位置、自身位置、指向所在树的指针。块的结构如下图所示：

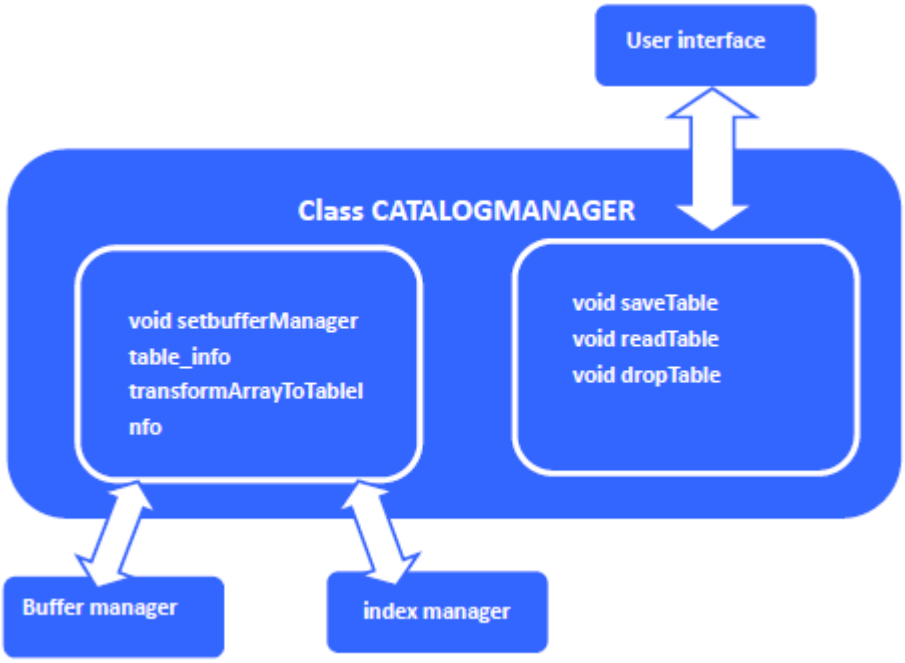



外部传入的对于 index 的操作具体由 B+树类中的操作承担，再细化到 NODE 类操作。BPLUSTREE 类主要三个函数（创建、更新、读取），而 NODE 类包括所有细节的操作，下图省略了 NODE 类读取相关位置信息的函数。




CatalogManager 类

CatalogManager 类为 API 调用 buffermanager 建立了桥梁，本模块中存储了数据库所有表的定义信息、表中所有字段的定义信息和所有索引的信息。Class CatalogManager 主要有两类函数，一类是与底层交互的转化表格信息的函数，一类是与用户界面交互的管理表格信息的函数（如下图所示）。




RecordManager 类

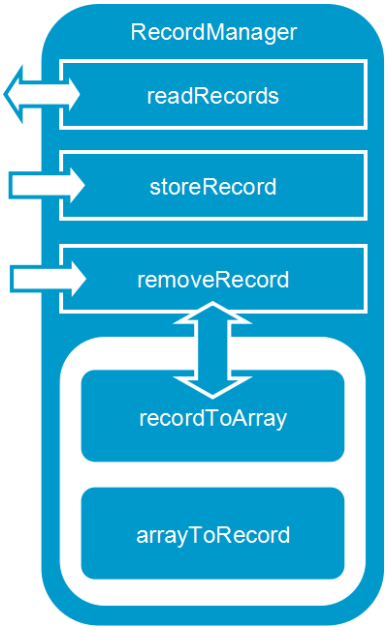
在模块中，将调用独立的基类模块 Basic，该模块中有 MiniSQL 使用的 Record 类、TableInformation 类的定义。

整个模块包括 5 个函数，两个内部格式转换函数，三个分别负责对 Record 进行读取、

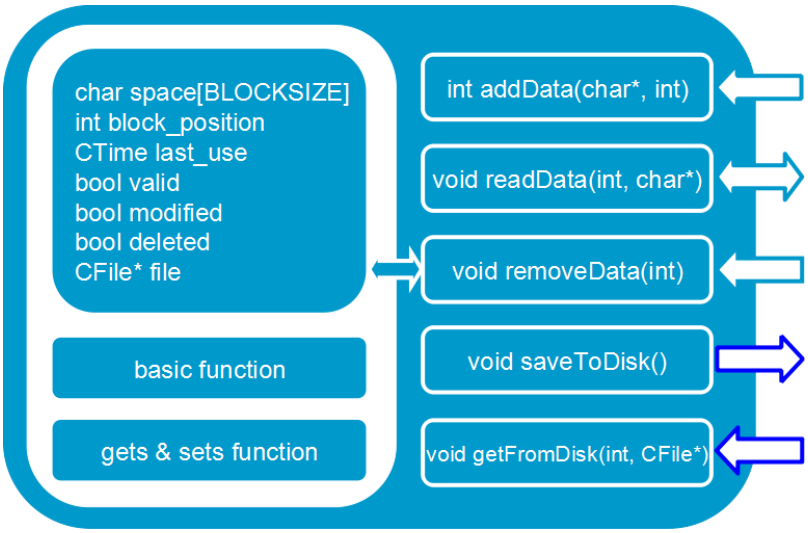
存储、删除的对外接口。在进行 Record 读取的过程中，表格所含的信息也会发生变化（即表格的 metadata），因此被操作的表格对应的 TableInformation 等也会被更改。

整体模块接收 Array 作为输入，Record 作为输出，若进行了不允许的操作，比如空表删除等，也会及时将错误返回给 Interpreter。

模块设计图如下：

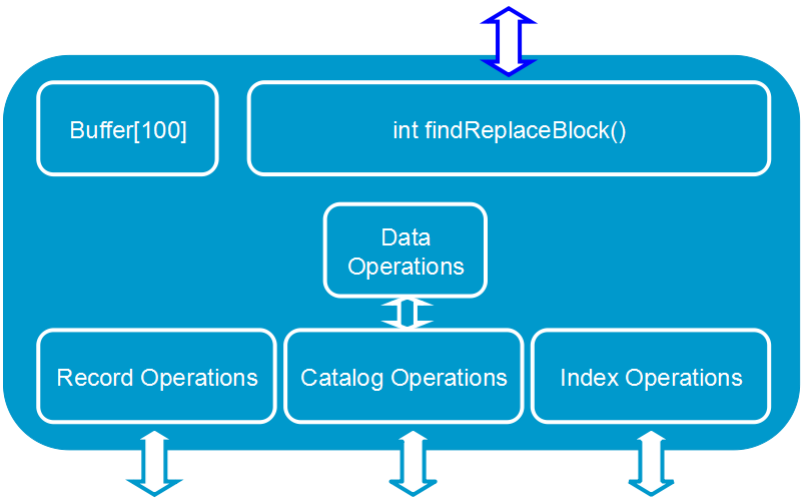



Block 类




该类拥有一些内部函数，主要是用于判断、修改标志值以及提供信息给接口函数使用。另外，在接口函数中，上三个函数用于对上层使用，下两个函数用于沟通下层。

2）BufferManager 类



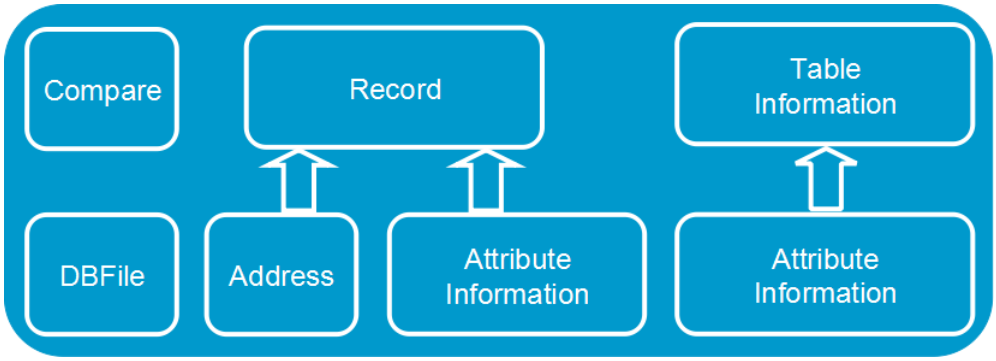

该类在主要实现对下层数据的管理和对上层 Manager 的支持。最核心的部分其实是对 Block 的调度。兼有一系列针对上层 Manager 的接口。

Basic 基类

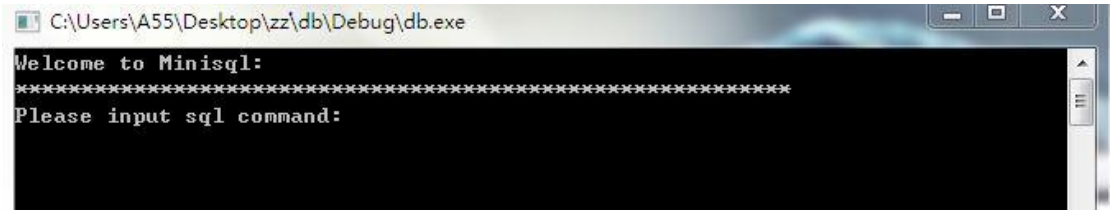
这个模块主要是被其他模块使用，因此类中属性重要性大于方法，大多的类仅需要实现一下拷贝构造、重载'='运算符即可。

其中 Address、AttributeValue 类是最基本的，Record、TableInformation 和 AttributeInformation 类则用于管理 AttributeValue 类，其中 TableInformation 类则是通过调用 AttributeInformation 类来管理。

整体设计图如下：




## 第 3.2 节 主窗口及主函数设计




主函数源码：

```
int main()
{
    ……
```

```
interpreter b;
char    in;
cout << "Welcome to Minisql: " << endl;
while(1)//循环读取指令
{
        cout << "************************************************** " << endl;
        cout << "Please input sql command: " << endl;
        a = "";
        //拆解指令
        do{
                gets_s(cmd);
                a += " ";
                a += cmd;
        }
        while(a.GetAt(a.GetLength() - 1) != ';');

        if(a == " quit;")
                break;
        CString temp = a;
        temp.MakeLower();
        if(temp.Find("create") != -1 || temp.Find("drop") != -1
                || temp.Find("select") != -1 || temp.Find("insert") != -1
                || temp.Find("delete") != -1)
        {
                if (b.ExecuteCommand(a)){
                        cout << "指令成功运行!" << endl;
                        cout << “总时间为” << b.Total_time << "秒" << endl;
                }
                else
                        cout << "指令失败!" << endl;
        }//读取脚本文件中的指令
        else if (temp.Find("execfile") != -1){
                for (int i = 10; i < temp.GetLength()-1; i++){
                        filename += temp.GetAt(i);
                }
                ifstream fin("e:\sql.txt");
                a = "";
                if (!fin) // check whether file opened successfully
                        cout << "File not exist!" << endl;
                else{
                        //逐行读取，具体过程与直接输入相同
                        ……
                        fin.close();
                }
```

```
            }
            else{
                cout << "指令格式错误!" << endl;
            }
        }
        return 0;
}
```

## 第 3.3 节 Catalog Manager 接口

3.3.1 与 buffer 交互

```
void setbufferManager(bufferManager* b)
```

3.3.2 管理数据库表中的信息

```
table_info transformArrayToTableInfo(char* b, int count);
```

3.3.3 保存数据库表

```
void saveTable(table_info a);
```

3.3.4 读取数据库表

```
bool readTable(CString table_name, table_info& tableInfo);
```

3.3.5 删除数据库表

```
void dropTable(CString table_name);
```

## 第 3.4 节 Record Manager 接口

3.4.1 读取记录，用于选择语句、索引查找

```
CArray<record, record&>* readRecords
(CString tableName, TableInformation tableInfo, CArray<Compare, Compare&>* cp );
```

3.4.2 存储记录，用于插入记录语句

```
void storeRecord(CString tableName, Record data);
```

3.4.3 删除记录，用于删除记录语句

```
void removeRecord
(CString tableName, TableInformation* tableInfo, CArray<Compare, Compare&>* cp )
```

## 第 3.5 节 Index Manager 接口

3.5.1 创建索引

```
public void CreateBPlusTree(CString KeyType, int KeyTypeCount);
```

3.5.2 删除索引

```
public void Drop();
```

3.5.3 插入新的索引值

```
public void InsertValue(void* pValue, int Block, int Index);
```

3.5.4 删除索引值

```
public void DeleteValue(void* pValue);
```

3.5.5 等值查找

```
public bool indexmanager::FindValue(void* pValue, int& Block, int& Index)
```

## 第 3.6 节 Buffer Manager 接口

3.6.1 替换算法

```
int findReplaceBlock();
```

3.6.2 向下层的接口，请求磁盘数据

```
int readBlockFromDisk(int blockPosition);
```

3.6.3 面向 RecordManager 的接口

```
void addRecord(CString tableName, CString head, int size);
void removeRecord(Address add);
Address nextRecordPosition(Address currentAdd);
void findTableEntry(CString tableName, Address& returnAddress);
```

3.6.4 面向 CatalogManager 的接口

```
void addInfo(CString head, int size);
CString readInfo(CString tableName, int& size);
void removeInfo(CString tableName);
int addTableEntry(CString tableName, Address& returnAddress);
void removeTableEntry(CString tableName);
```

3.6.5 单个数据项的接口

```
int storeData(CString head, int size, Address& returnAddress);
CString readData(Address address, int& size);
void removeData(Address address);
```

## 第 3.7 节 DB Files 接口

3.7.1 创建新的 Block

```
int allocNewBlock();
```

3.7.2 查找是否有足够的空间存储该数据

```
int findEnoughSpace(int， int);
```

# 第 4 章 MINISQL 系统测试

## 第 4.1 节 基本测试

执行正确语句

```
Please input sql command:
create table teacher (
                sno char(8),
                sname char(16) unique,
                sage int,
                sgender char(1),
                primary key (sno)
);
指令成功运行!
总时间为0.03秒
**************************************************************
Please input sql command:
```

语句中存在语法错误

```
**************************************************************
Please input sql command:
create table ();
在create table附近有语法错误!
指令失败!
**************************************************************
Please input sql command:
create table student(;
左括号不匹配!
指令失败!
**************************************************************
Please input sql command:
```

# 第 4.2 节 Index 测试

创建 index 之前查找

```
select * from student where sname = 'wy';
              列名                 列名                 列名                 列名

               sno                sname                sage              sgender

           1001001                   wy                  22                    M

指令成功运行!
总时间为0.041秒
Welcome to Minisql:
Please input sql command:
```

创建 index 之后的查找

```
create index snameIndex on student ( sname );
指令成功运行!
总时间为0秒
Welcome to Minisql:
Please input sql command:

select * from student where sname = 'wy';
        列名          列名          列名          列名

         sno         sname         sage       sgender

     1001001            wy           22             M

指令成功运行!
总时间为0.011秒
Welcome to Minisql:
Please input sql command:
```

删除 index 之后的查找

```
drop index snameIndex;
指令成功运行!
总时间为0秒
Welcome to Minisql:
Please input sql command:

select * from student where sname = 'wy';
        列名          列名          列名          列名

         sno         sname         sage       sgender

     1001001            wy           22             M

指令成功运行!
总时间为0.041秒
Welcome to Minisql:
Please input sql command:
```

# 第 4.3 节 Buffer 测试

Buffer 大小调节为 100：

```
        9001001            yu           23             M

指令成功运行!
总时间为0.823秒
Welcome to Minisql:
Please input sql command:
```

Buffer 大小调节为 50：

```
        9001001            yu           23             M

指令成功运行!
总时间为0.859秒
Welcome to Minisql:
Please input sql command:
```

可以看到最终效果不是非常明显，可能是数据量还是不够大的关系。

# 第 5 章 分工说明

## 本系统的分工如下：

Interpreter 模块 ……尹依婷
API 模块 ……尹依婷
Catalog Manager 模块 ……尹依婷
Index Manager 模块 ……尹依婷
Record Manager 模块 ……陆洲
Buffer Manager 模块 ……陆洲
DB file 模块 ……陆洲
B asic 基类模块 ……陆洲
总体设计报告 ……全体
测试 ……全体