



# MiniSQL 设计总报告

组员：沈栋 杨健 吴陈奥

课程：数据库系统原理

专业：计算机科学与技术

# 目录

## 第一章 MiniSQL 总体框架

---

### 第 1.1 节 MiniSQL 实现功能分析

### 第 1.2 节 MiniSQL 系统体系结构

### 第 1.3 节 设计语言及运行环境

## 第二章 MiniSQL 各模块实现功能

---

### 第 2.1 节 Interpreter 实现功能

### 第 2.2 节 API 实现功能

### 第 2.3 节 Catalog Manager 实现功能

### 第 2.4 节 Recorder Manager 实现功能

### 第 2.5 节 Index Manager 实现功能

### 第 2.6 节 Buffer Manager 实现功能

### 第 2.7 节 DBFile 实现功能

## 第三章 内部数据形式及各模块提供的窗口

---

### 第 3.1 节 内部数据形式

### 第 3.2 节 主窗口及函数设计

### 第 3.3 节 Catalog Manager 接口

### 第 3.4 节 Recorder Manager 接口

### 第 3.5 节 Index Manager 接口

### 第 3.6 节 Buffer Manager 接口

## 第四章 MiniSQL 系统测试

---

## 第五章 分工说明

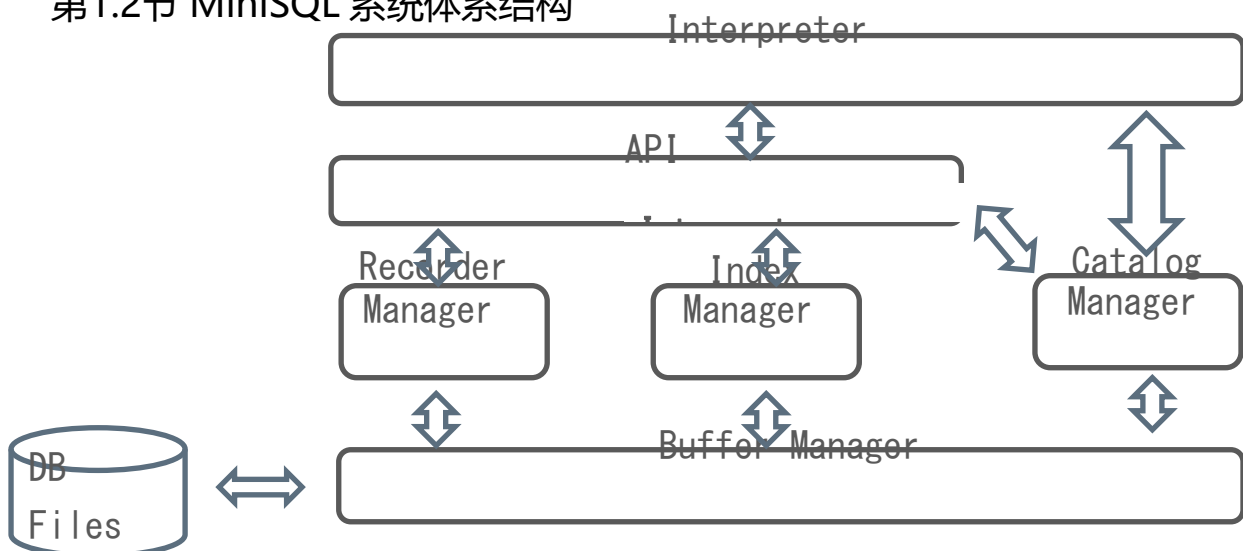
---

# 第 1 章 MINISQL 总体框架

## 第1.1节 MiniSQL 实现功能分析

- 1) 总功能：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。
- 2) 数据类型：支持三种基本数据类型：int，char(n)，float，其中 char(n)满足  $1 \leq n \leq 255$
- 3) 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。
- 4) 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，所有的 B+树索引都是单属性单值。
- 5) 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、不等值和区间查询。
- 6) 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



第 1.3 节设计语言及运行环境 工具：Visual Studio 2013 环境：WIN10

## 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，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，负责各个模块间的转换。该接口以 Interpreter 层解释生成的命令内部表示为输入，根据命令的类型制定执行规则，并调用 Record Manager、Index Manager 和 Catalog Manager 提供的相应接口进行执行，最后将执行结果输出。

### 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：

- 1) 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。
- 2) 表中每个字段的定义信息，包括字段类型、是否唯一等。
- 3) 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 章 MINISQL 各模块实现功能

### 第 2.4 节 Recorder Manager 实现功能

Recorder Manager 负责管理记录表中数据的数据文件。主要功能为实现记录的插入、删除与查找操作，对 index 所需数据信息的查找返回，并对外提供相应的接口。其中记录的查找操作能够支持不带条件的查找和带一个或多个条件的查找（包括等值查找、不等值查找和区间查找）。

### 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）、多种条件（等值、区间）查找、插入键值、删除键值等操作，并对外提供相应的接口。

### 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：

- 1) 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
- 2) 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
- 3) 记录缓冲区中各页的状态，如是否被修改过等
- 4) 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
- 5) 向 Recorder Manager、Index Manager 提供供写入的块。

### 第 2.7 节 DBFile 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。同时还有写回文件和读取文件的功能。

## 第 3 章 内部数据形式及各模块提供的接口

### 第 3.1 节 内部数据形式

Interpreter 返回信息：

1. 如果输入有错，则 error\_info=相应错误信息(TABEXISTED, COLERR 等)。
2. 否则
  - 1) operation\_type=操作类型 ( SEL\_WHERE,DELETE,INSERT 等 )。
  - 2) tableInfo 存储操作中所用的有关于表的信息,结构如下

```
class Table{
public:
    string name; //表的名称
    int attriNum; //表中的属性总数
    int totalLength; //一条记录的总长度
    int primaryindex; //标记哪一个元素为Primary index
    int biggestnum; //这个表的最后一个的block的编号 ( 文件编号+block编号 ) 用于insert 操作
    vector<Attribute>attributes; //属性的信息
    Table() :name(""), attriNum(0), totalLength(0){}
    ~Table(){}
};
```

其中 Attribute 结构如下：

```
class Attribute{
public:
    string name; //名字
    int type; //类型
    int length; //长度
    bool isPk; //Primary key标记
    bool isUnique; //Unique标记
    Attribute(){
        isPk = false;
        isUnique = false;
    }
    Attribute(string n, int t, int l, bool p, bool u)
        :name(n), type(t), length(l), isPk(p), isUnique(u){}
    ~Attribute(){}
};
```

## 第 3 章 内部数据形式及各模块提供的接口

除了在 create table 操作中，table 信息完全填充，其余操作只使用 name，为方便接口，统一使用类作为参数。

3) indexInfo 用于存储和 index 有关信息，结构如下：

```
class Index
{
public:
    string index_name;           //索引名
    string table_name;           //对应的表名
    string attribute_name;       //对应的属性名字
    int BBN;                     //记录的是文件中最大block的编号
    int columnLength;            //记录的是index对应的value的总长度
    int blockoffset;             //block的偏移量
    int fileoffset;              //这个index是哪一个文件中的 记录他文件偏移
    int type;                    //属性对应类型
    Index() : columnLength(0){}
    ~Index(){}
};
```

除了在 create index 中类成员会被全部赋值，其余操作只使用 index\_name，为方便接口传输，统一使用类为参数

4) value 保存 insert 操作需要插入的记录，结构如下：

```
class Row{
public:
    vector<string> columns; //存储每一个值
    int isDelete; //标记懒惰删除
    int biggestnum; //记录对应文件中block编号
    int blockoffset; //在block中的偏移量
    Row() :isDelete(0), biggestnum(0), blockoffset(0){};
};
class Data{
public:
    vector<Row>rows;
};
```

5) conditions 保存查找和删除时的条件，结构如下：

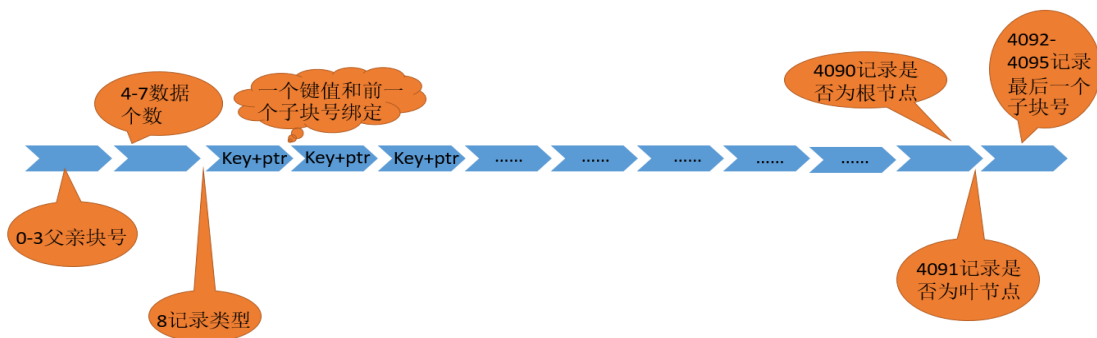


## 第 3 章 内部数据形式及各模块提供的接口

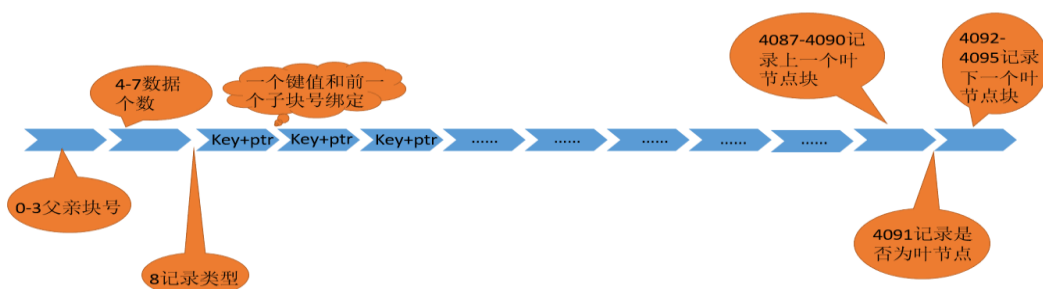
```
enum Comparision{ Lt, Le, Gt, Ge, Eq, Ne, LtLt, LtLe, LeLt, LeLe, Un };  
class Condition{  
public:  
    Comparision op; //操作类型, >, <, >=, <=, <>等  
    string column_name; //对应的属性名字  
    string minValue; //最小值, >, >=或者左侧为<, <=时使用  
    string maxValue; //最大值, <, <=使用  
    string eqValue; //等值查询  
    string eqValue2; //第二等值, 左侧为<=时使用  
    Condition(){  
        op = Un;  
        column_name = "";  
        minValue = MIN;  
        maxValue = MAX;  
        eqValue = "";  
    }  
};
```

6) Pkconditions 和 uniqueconditions 用于存储主键和 unique 属性对应值的信息, 结构和 conditions 相同。

B+树中间块结构：



叶块 Leaf 的结构设计：





## 第 3 章 内部数据形式及各模块提供的接口

Catalog 数据：

BufferBlock 数据结构：

### 第 3.2 节 主窗口及主函数设计

运行时系统界面如下：

```
*****
Welcome to MiniSQL!
2016-6-18 23:39:0
*****
->
```

SQL 命令在 “->” 提示符后输入

### 第 3.3 节 Catalog 接口

1) 创建关于表和 index 的记录

```
bool CreateTable_catalog(const Table &);
```

```
bool CreateIndex_catalog(const Index&);
```

2) 删除表和 index

```
bool DropTable_catalog(const Table &);
```

```
bool DropIndex_catalog(const Index &);
```

3) 得到这个表的最后面的一个 block 的编号

```
int get_Lblockindex(const Table &);
```

```
int get_Lblockindex(const Index &);
```

## 第 3 章 内部数据形式及各模块提供的接口

4) 对表这个最后面的 block 编号更新

```
bool add_Lblockindex(Table &);
```

```
bool add_Lblockindex(Index &);
```

5) 检查是否是 pk, 表是否存在, 是否该元素, 那个元素是 index

```
bool check_PK(const Table&);
```

```
bool check_Table(const Table &);
```

```
bool check_attribute(const Table&);
```

```
bool check_index(const Index &);
```

```
vector<Index> check_index(Table &);
```

6) 返回 table 和 index 的 information

```
Table gettable(const Index&);
```

```
Table get_table_info(const Table&);
```

### 第 3.4 节 Recorder Manager 接口

1) 插入记录 :

```
Data InsertValue(Table &table, Data &data)
```

2) 无索引记录 :

```
Data SelectNoIndex(Table &table, Conditiondata conditions)
```

3) 有索引查找

## 第 3 章 内部数据形式及各模块提供的接口

Data SelectIndex(Table &table, Indexcolumn &indexdata, Conditiondata conditions)

4) 条件有索引删除记录

Data DeleteIndex(Table &table, Indexcolumn &indexdata, Conditiondata conditions)

5) 条件无索引删除记录

Data DeleteNoIndex(Table &table, Conditiondata conditions)

6) 建立索引返回所有记录对应索引信息

Indexcolumn CreateIndexData(Table &table, Index &index)

7) 插入、删除返回索引信息

Indexcolumn GetIndexData(Table &table, Index &index , Data &data)

### 第 3.5 节 Index Manager 接口

1) 创建索引

```
void createIndex(Index& indexinfor, IndexColumn DataofIndex);
```

2) 单值查找

```
int selectEqual(Index indexinfor, string key);
```

3) 多值查找 ( 条件查找 )

```
list<int>::iterator selectBetween(Index indexinfor, string keyFrom, string  
keyTo, int& blockoffset);
```

## 第 3 章 内部数据形式及各模块提供的接口

### 4) 单值插入

```
void insertValue(Branch& Searchnode, IndexRow dataRecord, Index&
indexinfor);
```

### 5) 多值删除 ( 包含单值删除 )

```
void deleteValue(Index indexinfor, IndexColumn dc, int&blockoffset);
```

## 第 3.6 节 Buffer Manager 接口

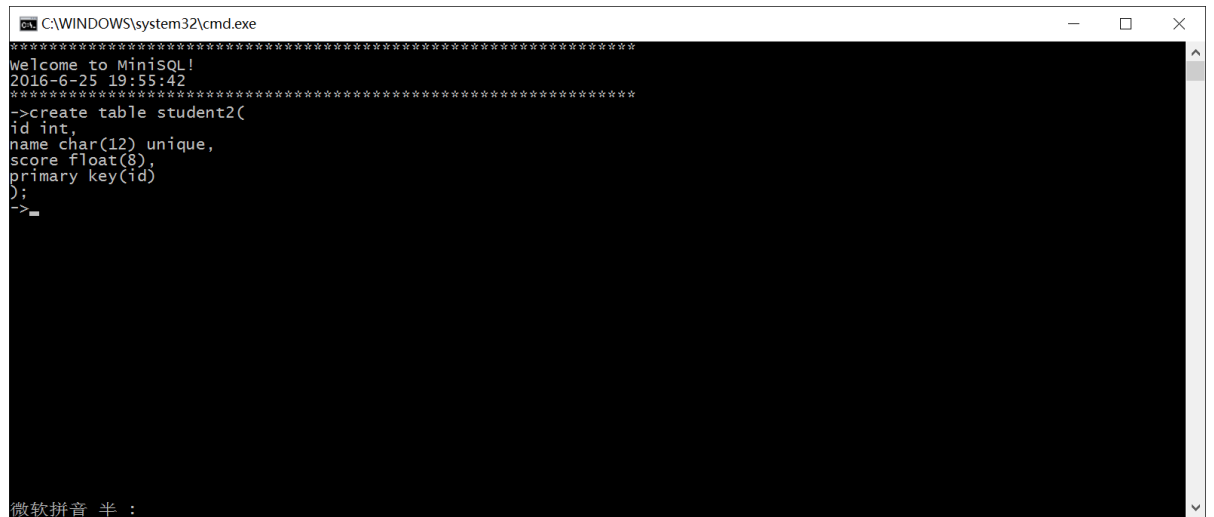
## 第 3.6 节 DBFile 接口

DBFile 完全有文件 ( .data, .xml ) 组成 , 在工程目录中有 3 个文件由 buffer 负责调用。

## 第 3 章 内部数据形式及各模块提供的接口

## 第 4 章 MINISQL 系统测试

### 1. 建表成功



```
C:\WINDOWS\system32\cmd.exe
Welcome to MiniSQL!
2016-6-25 19:55:42
-->create table student2(
id int,
name char(12) unique,
score float(8),
primary key(id)
);
-->
```

### 2. 查找

`select * from student2 where id=1080100345; --考察 int 类型上的等值条件查询`

```
select * from student2 where id=1080100345;
id      name      score
1080100345  name345  87.5
0.039
-->
```

`select * from student2 where score<>99.5; --考察 float 类型上的不等条件查询`

## 第 4 章 MINISQL 系统测试

```
C:\WINDOWS\system32\cmd.exe
1080109973      name9973      76.5
1080109974      name9974      52.5
1080109975      name9975      97
1080109976      name9976      50
1080109977      name9977      56.5
1080109978      name9978      84.5
1080109980      name9980      88
1080109981      name9981      78.5
1080109982      name9982      53.5
1080109983      name9983      58.5
1080109984      name9984      64.5
1080109985      name9985      71
1080109986      name9986      61
1080109987      name9987      91.5
1080109988      name9988      75
1080109989      name9989      58
1080109990      name9990      75
1080109991      name9991      69
1080109992      name9992      50.5
1080109993      name9993      67.5
1080109994      name9994      97.5
1080109995      name9995      59.5
1080109996      name9996      65.5
1080109997      name9997      61
1080109998      name9998      84.5
1080109999      name9999      69.5
1080110000      name10000     80.5
126.76
->
微软拼音 半 :
```

select \* from student2 where score=99.5; --考察 float 类型上的等值  
条件查询

```
C:\WINDOWS\system32\cmd.exe
1080108107      name8107      99.5
1080108307      name8307      99.5
1080108394      name8394      99.5
1080108401      name8401      99.5
1080108403      name8403      99.5
1080108407      name8407      99.5
1080108532      name8532      99.5
1080108545      name8545      99.5
1080108643      name8643      99.5
1080108659      name8659      99.5
1080108753      name8753      99.5
1080108797      name8797      99.5
1080108848      name8848      99.5
1080109037      name9037      99.5
1080109095      name9095      99.5
1080109200      name9200      99.5
1080109277      name9277      99.5
1080109289      name9289      99.5
1080109298      name9298      99.5
1080109345      name9345      99.5
1080109348      name9348      99.5
1080109398      name9398      99.5
1080109574      name9574      99.5
1080109629      name9629      99.5
1080109634      name9634      99.5
1080109680      name9680      99.5
1080109979      name9979      99.5
1.864
->
微软拼音 半 :
```

select \* from student2 where name='name345'; --考察 char 类型上  
的等值条件查询，此处需观察执行时间 t1



## 第 4 章 MINISQL 系统测试

```
1.004  
->select * from student2 where name='name345';  
id      name      score  
1080100345    name345  87.5  
0.424  
->  
微软拼音 半 ;
```

select \* from student2 where name<>'name9999'; --考察 char 类型上的不等条件查询

```
-----  
1080109973    name9973    76.5  
1080109974    name9974    52.5  
1080109975    name9975     97  
1080109976    name9976     50  
1080109977    name9977    56.5  
1080109978    name9978    84.5  
1080109979    name9979    99.5  
1080109980    name9980     88  
1080109981    name9981    78.5  
1080109982    name9982    53.5  
1080109983    name9983    58.5  
1080109984    name9984    64.5  
1080109985    name9985     71  
1080109986    name9986     61  
1080109987    name9987    91.5  
1080109988    name9988     75  
1080109989    name9989     58  
1080109990    name9990     75  
1080109991    name9991     69  
1080109992    name9992    50.5  
1080109993    name9993    67.5  
1080109994    name9994    97.5  
1080109995    name9995    59.5  
1080109996    name9996    65.5  
1080109997    name9997     61  
1080109998    name9998    84.5  
1080110000    name10000   80.5  
121.85  
->  
微软拼音 半 ;
```

select \* from student2 where score>90 and score<95; --考察多条件 and 查询

## 第 4 章 MINISQL 系统测试

```
C:\WINDOWS\system32\cmd.exe
1080109776      name9776      93.5
1080109787      name9787      92
1080109796      name9796      90.5
1080109798      name9798      91.5
1080109802      name9802      90.5
1080109804      name9804      94.5
1080109814      name9814      94.5
1080109817      name9817      94.5
1080109830      name9830      92.5
1080109834      name9834      91
1080109843      name9843      91.5
1080109846      name9846      94
1080109855      name9855      93.5
1080109856      name9856      94
1080109863      name9863      91
1080109868      name9868      92
1080109876      name9876      94.5
1080109883      name9883      94.5
1080109889      name9889      92.5
1080109893      name9893      92.5
1080109900      name9900      91
1080109924      name9924      94.5
1080109926      name9926      93
1080109927      name9927      92.5
1080109947      name9947      91.5
1080109969      name9969      94
1080109987      name9987      91.5
12.289
-->
微软拼音 半 :
```

select \* from student2 where score>90 and id<=1080100100; --考察多条  
件 and 查询

```
-->select * from student2 where id<=1080100100 and score >90;
id      name      score
1080100001      name1      99
1080100003      name3      98.5
1080100004      name4      91.5
1080100015      name15     93.5
1080100021      name21     93.5
1080100022      name22     93.5
1080100037      name37     92
1080100044      name44     91
1080100046      name46     98.5
1080100048      name48     94.5
1080100049      name49     97
1080100052      name52     97
1080100065      name65     93
1080100066      name66     99
1080100071      name71     94
1080100072      name72     93.5
1080100085      name85     98.5
1080100090      name90     96.5
1080100094      name94     99.5
0.104
```

insert into student2 values(1080100345,'name345',100); --报 primary key  
约束冲突 ( 或报 unique 约束冲突 )

## 第 4 章 MINISQL 系统测试

```
insert into student2 values(1080100345,'name345',100);
Primary Key exists!
->
微软拼音 半 :
```

create index stuidx on student2 ( name ); --在 name 这个 unique 属性上创建 index

```
create index stuidx on student2 ( name );
->
select * from student2 where name='name345';
id          name      score
1080100345  name345  87.5
0.055
->
```

select \* from student2 where name='name345'; --此处需观察执行时间 t2

```
->
select * from student2 where name='name345';
id          name      score
1080100345  name345  87.5
0.055
->
微软拼音 半 :
```

insert into student2 values(1080197998,'name97998',100); --考察在建立 b+树后再插入数据，b+树有没有做好 insert

```
->insert into student2 values(1080197998,'name97998',100);
->
微软拼音 半 :
```

select \* from student2 where name='name97998'; --此处需观察执行时间 t3

```
->select * from student2 where name='name97998';
id          name      score
1080197998  name97998  100
0.063
->
```

delete from student2 where name='name97998'; --考察 delete，同时考察 b+树的 delete

## 第4章 MINISQL 系统测试

```
0.063
->delete from student2 where name='name97998';
->
微软拼音 半 :
```

select \* from student2 where name='name97998';

```
>delete from student2 where name='name97998';
->select * from student2 where name='name97998';
Not Found!
0.049
->
微软拼音 半 :
```

insert into student2 values(1080197998,'name97998',100);

```
0.049
->insert into student2 values(1080197998,'name97998',100);
->
微软拼音 半 :
```

drop index stuidx; --考察 drop index

```
->
drop index stuidx;
->
微软拼音 半 :
```

select \* from student2 where name='name97998'; --需观察此处的执行时间

t4

```
->select * from student2 where name='name97998';
id      name      score
1080197998  name97998  100
0.415
->
微软拼音 半 :
```

select \* from student2 where name='name345'; --需观察此处的执行时间 t5

```
->select * from student2 where name='name345';
id      name      score
1080100345  name345  87.5
0.416
->
```

## 第 4 章 MINISQL 系统测试

delete from student2 where id=1080100345; --考察 delete

```
>insert into student2 values(1080100345, name);
->delete from student2 where id=1080100345;
->
```

select \* from student2 where id=1080100345;

```
->select * from student2 where id=1080100345;
Not Found!
0.02
->
```

delete from student2 where score=99.5; --考察 delete

```
>delete from student2 where score=99.5;
>
微软拼音 半 :
```

select \* from student2 where score=99.5;

```
>delete from student2 where score=99.5;
->select * from student2 where score=99.5;
Not Found!
0.347
->
微软拼音 半 :
```

delete from student2; --考察 delete

```
->delete from student2;
->
微软拼音 半 :
```

select \* from student2;

```
->select * from student2;
Not Found!
->
微软拼音 半 :
```

drop table student2; --考察 drop table

```
->drop table student2;
->
微软拼音 半 :
```

## 第 4 章 MINISQL 系统测试

select \* from student2;

```
->select * from student2;  
No such table exists!  
->
```

本系统的分工如下：

Interpreter 模块	杨健
API 模块	吴陈奥
Catalog Manager 模块	沈栋
Recorder Manager 模块	杨健
Index Manager 模块	吴陈奥
Buffer Manager 模块	沈栋
DBFile 模块	沈栋
总体设计报告	ALL
模块汇总	ALL
测试	ALL