MiniSQL设计报告

詹奇 3190103512

课程名称：数据库系统

指导教师：黄忠东

2021 年 6 月 3 日

一、实验概览

1.1 实验目的

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。

通过对MiniSQL的设计与实现，提高学生的系统编程能力，加深对数据库系统原理的理解。

1.2 实验需求

数据类型

要求支持3种数据类型：`int`, `float`, `char(n)`；其中 $1 \leq n \leq 255$

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`；必须定义单属性的主键

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的）

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。

插入和删除记录

支持每次一条的插入操作；支持每次一条或多条记录的删除操作。

1.3 整体架构

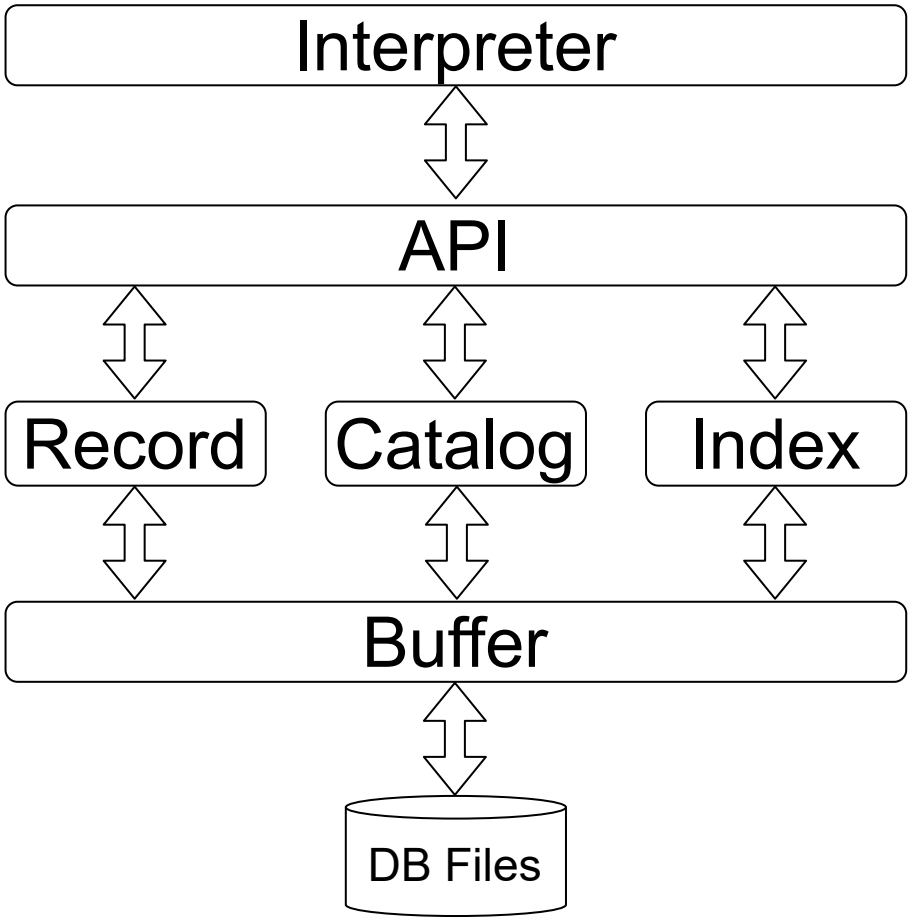
MiniSQL主要包含6个模块：

- Interpreter
- API
- Catalog Manager
- Record Manager
- Index Manager
- Buffer Manager

此外，数据库在磁盘上保存至相应的文件中：

- "./dbfiles/catalog" 文件保存数据库元数据信息
- "./dbfiles/record/" 目录下保存各表的记录
- "./dbfiles/index/" 目录下保存各表的索引

MiniSQL的各个模块的层级图如下：



该程序使用C++完成，将以上各个模块抽象成类，并由API直接调用下层模块类的方法来完成指令。

二、各模块实现功能

2.1 Interpreter

Interpreter模块直接与用户交互，主要实现以下功能：

- 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。
- 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。

2.2 API

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。

2.3 Catalog Manager

Catalog Manager负责管理数据库的所有模式信息，包括：

- 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。
- 表中每个字段的定义信息，包括字段类型、是否唯一等。
- 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API模块使用。

2.4 Record Manager

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条件的查找（包括等值查找、不等值查找和区间查找）。

2.5 Index Manager

Index Manager负责驱动BPlusTree，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。而BPlusTree模块则真正管理索引的增删查，通过4KB的结点在磁盘上构建B+树。

2.6 Buffer Manager

具体来说，Buffer Manager模块涉及`Block`和`BufferManager`对象，功能如下:

- 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。
- 记录缓冲区中各页的状态，如是否被修改过。
- 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。
- 使用LRU算法实现页面替换。

Buffer Manager掌管内存中以4KB为单位的一片缓冲池。更高层的模块利用Buffer Manager与磁盘文件交互，是一种能有效减少磁盘访问次数的读写数据库文件的方法。

三、各模块详细实现

各模块详细设计请参见模块详细设计报告，不再赘述。

四、系统测试

4.1 创建表语句

4.1.1 成功建表

```
Minisql>create table student (
sno char(8),
        sage int,
sname char(2) unique,
sgender char (1)  unique,
primary key (sage)
);
API::Query OK
(0.05 sec)
```

4.1.2 重复建表

```
Minisql>create table student (
sno char(8),
        sage int,
sname char(2) unique,
sgender char (1)  unique,
primary key (sage)
);
CatalogManager::Table 'student' already exists
(0.01 sec)
```

4.1.3 创建索引

```
Minisql>create index stunameidx on student ( sname );
API::Query OK
(0.01 sec)
```

4.1.4 重复创建索引

```
Minisql>create index stunameidx on student ( sname );
CatalogManager::Index already exists
(0.00 sec)
```

4.2 插入记录语句

4.2.1 成功插入

```
Minisql>insert into student values ('12345687',22,'wy','M');
API::Query OK
(0.00 sec)
```

4.2.2 插入重复的主键

```
Minisql>insert into student values ('12345687',22,'wy','M');
API::Query OK
(0.00 sec)
Minisql>insert into student values ('12345687',22,'wy','M');
Duplicate entry '22'
(0.01 sec)
```

4.2.3 插入数据type不匹配

```
Minisql>insert into student values ('12345687',22,'wy',12);
CatalogManager::Column type doesn't match value type
(0.00 sec)
```

4.2.4 插入数据count不匹配

```
CatalogManager::Column count doesn't match value count
valuecout=3tablecout=4
(0.00 sec)
```

4.3 查询语句

4.3.1 无条件查询

```
Minisql>select * from student;
sno          sage   sname   sgender
12345687     22     wy      M
12345689     23     xy      S
(0.01 sec)
```

4.3.2 单条件查询

```
Minisql>select * from student where sage = 22;
sno          sage   sname   sgender
12345687     22     wy      M
(0.00 sec)
```

4.3.3 多条件查询

```
Minisql>select * from student where sage = 22 and sname = 'wy';
sno          sage   sname   sgender
12345687     22     wy      M
(0.01 sec)
```

4.3.4 空查询

```
Minisql>select * from student where sage = 22 and sname = 'wdy';
empty set
(0.00 sec)
```

4.4 删除语句

4.4.1 全删除

```
Minisql>delete from student;
API::Query OK
(0.00 sec)
Minisql>select * from student;
empty set
(0.00 sec)
```

4.4.2 单条件删除

```
Minisql>select * from student;
sno          sage   sname   sgender
12345689     23     xy      S
12345687     22     wy      M
(0.02 sec)
Minisql>delete from student where sage = 22;
API::Query OK
(0.00 sec)
Minisql>select * from student;
sno          sage   sname   sgender
12345689     23     xy      S
(0.00 sec)
```

4.4.3 多条件删除

```
Minisql>delete from student where sage = 23 and sgender = 'S';
API::Query OK
(0.00 sec)
Minisql>select * from student;
empty set
(0.00 sec)
```

4.5 drop语句

4.5.1 成功删除索引

```
Minisql>drop index stunameidx;
API::Query OK
(0.00 sec)
```

4.5.2 删除不存在的索引

```
Minisql>drop index stunameidx;
IndexManager::No such index 'stunameidx'
(0.00 sec)
```

4.5.3 成功删除表

```
Minisql>drop table student;
API::Query OK
(0.00 sec)
```

4.5.4 删除不存在的表

```
Minisql>drop table student;
CatalogManager::Unknown table 'student'
(0.00 sec)
```

4.6 execfile语句

```
Minisql>execfile sqltest.txt;
drop table student
API::Query OK
create table student (
                sno char(8),
                        sage int,
                sname char(2) unique,
                sgender char (1)  unique,
                primary key (sage)
)
API::Query OK
create index aaaaaaaa on student ( sname )
API::Query OK
drop index stunameidx
IndexManager::No such index 'stunameidx'
insert into student values ('12345687',22,'wy','M')
API::Query OK
select * from student
sno          sage   sname   sgender
12345687     22     wy      M
(0.05 sec)
```

4.7 quit语句

```
PS D:\学习\大学\大二下\数据库系统\夏季大程> .\MiniSQL.exe
Minisql>quit;
PS D:\学习\大学\大二下\数据库系统\夏季大程>
```

4.8 语法错误

```
Minisql>create table student (;
Interpreter::SyntaxError
```

五、实验心得

这次数据库MiniSQL实验历时几周一人完成，因为复杂的模块调用，文件存储读取等遇到过很多问题，但最终都基本成功解决，收获良多。