MiniSQL 实验指导书

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。

2 实验需求

2.1 需求概述

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 。

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 unique 属性的主键定义。

索引的建立和删除

对于表的主键自动建立 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）。

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

支持每次一条记录的插入操作；

支持每次一条或多条记录的删除操作。(where 条件是范围时删除多条)

2.2 语法说明

MiniSQL 支持标准的 SQL 语句格式，每一条 SQL 语句以分号结尾，一条 SQL 语句可写在一行或多行。为简化编程，要求**所有的关键字都为小写**。在以下语句的语法说明中，用黑体显示的部分表示语句中的原始字符串，如 **create** 就严格的表示字符串“create”，其他非黑体显示的有其他的含义，如 表名 并不是表示字符串 “表名”，而是表示表的名称。

创建表语句

该语句的语法如下：

```
create table 表名 (
    列名 类型 ,
    列名 类型 ,

    列名 类型 ,
    primary key ( 列名 )
);
```

其中类型的说明见第二节“功能需求”。

若该语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。

示例语句：

```
create table student (
       sno char(8),
```

```
        sname char(16) unique,
        sage int,
        sgender char (1),
        primary key ( sno )
);
```

删除表语句

该语句的语法如下：

```
drop table 表名 ;
```

若该语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。

示例语句：

```
drop table student;
```

创建索引语句

该语句的语法如下：

```
create index 索引名 on 表名 ( 列名 );
```

若该语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。

示例语句：

```
create index stunameidx on student ( sname );
```

删除索引语句

该语句的语法如下：

drop index 索引名 ;

若该语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。

示例语句：

```
drop index stunameidx;
```

选择语句

该语句的语法如下：

select * from 表名 ;

或：

select * from 表名 **where** 条件 ;

其中“条件”具有以下格式：列 op 值 and 列 op 值 … and 列 op 值。

op 是算术比较符：= <> < > <= >=

若该语句执行成功且查询结果不为空，则按行输出查询结果，第一行为属性名，其余每一行表示一条记录；若查询结果为空，则输出信息告诉用户查询结果为空；若失败，必须告诉用户失败的原因。

示例语句：

```
select * from student;
select * from student where sno = '88888888';
select * from student where sage > 20 and sgender = 'F';
```

插入记录语句

该语句的语法如下：

```
insert into 表名 values ( 值 1 , 值 2 , … , 值 n );
```

若该语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。

示例语句：

```
insert into student values ('12345678','wy',22,'M');
```

删除记录语句

该语句的语法如下：

delete from 表名 ;

或：

delete from 表名 **where** 条件 **;**

若该语句执行成功，则输出执行成功信息，其中包括删除的记录数；若失败，必须告诉用户失败的原因。

示例语句：

```
delete from student;
delete from student where sno = '88888888';
```

退出 MiniSQL 系统语句

该语句的语法如下：

quit;

执行 SQL 脚本文件语句

该语句的语法如下：

execfile 文件名 **;**

SQL 脚本文件中可以包含任意多条上述 8 种 SQL 语句，MiniSQL 系统读入该文件，然后按序依次逐条执行脚本中的 SQL 语句。

关于返回信息

返回信息的内容参考 mysql，语言限定为英文，主要包括执行 sql 返回的行数(n row in set), 或者影响的行数(n row affects)，必须给出执行 SQL 所花费的时间 Duration。

执行出错需要指出错误的类型，具体位置不做要求。

错误类型包括(SYNTAX ERROR， INVALID INDENTIFIER， INVALID VALUE , INVALID ATTR FOR INDEX)，即语法错误(语法参考 mysql，miniSQL 不支持的数据类型和 SQL 等也报语法错误), 标识符错误(不存在的表名，字段名等) , 值错误(变量长度不足以容纳实际数据), 索引定义在非 unique 属性上。

可以参考 mysql workbench 和 mysql terminal client 的返回信息。

	#	Time	Action	Message	Duration / Fetch
✔	67	17:16:51	drop database rebuild	64 row(s) affected	2.578 sec

```
+----------+
| count(*) |
+----------+
|    82473 |
+----------+
1 row in set (7.71 sec)

mysql> create table test (char a);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
 for the right syntax to use near 'char a)' at line 1
```

3 设计指导

3.1 系统体系结构

MiniSQL 体系结构如下：

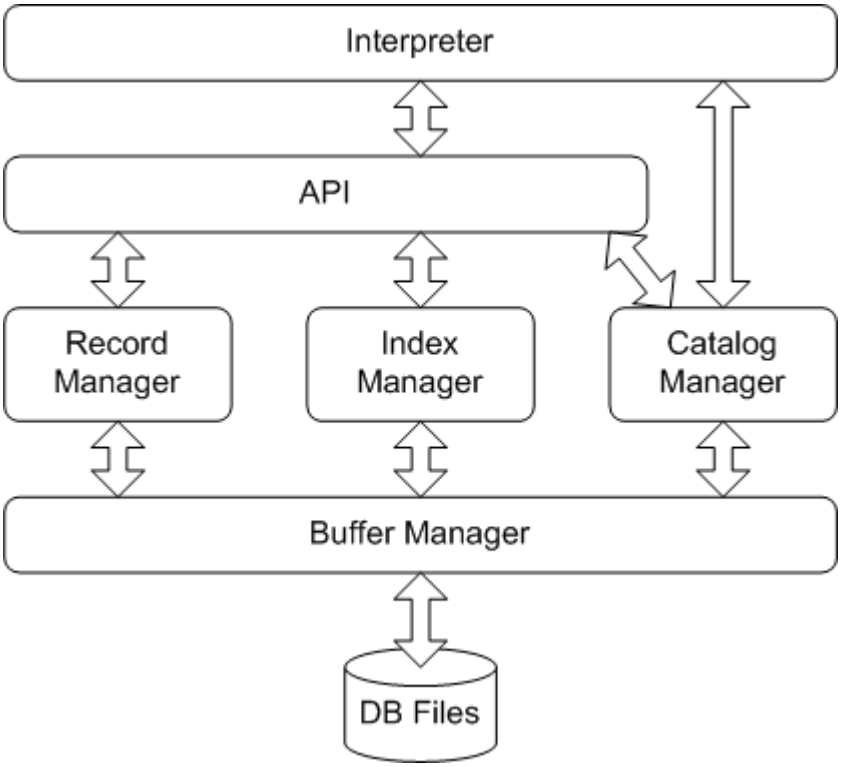

图 1 MiniSQL 体系结构

3.2 模块概述

3.2.1 Interpreter

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部分语义正确性，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。

3.2.2 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。

3.2.3 Catalog Manager

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。

3.2.4 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。

3.2.5 Index Manager

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。

3.2.6 Buffer Manager

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

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。

3.2.7 DB Files

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。

4 评分标准

本实验有一定的规模，实现上有一定的复杂度，推荐 1-3 人一组，完成一个完整的系统，不推荐一个人做其中的一个模块。实验采用“优、良、中、及格、不及格”五级评分制，具体的评分标准如下：

1. 实验最后得分由总体设计报告得分、详细设计报告得分和期末验收得分组成，其比例分别 20%、30%、50%；
2. 报告及时提交，则根据报告的质量给“优、良、中”中相应分级，未及时提交，则在报告质量分级基础上降一级，未提交报告或报告为抄袭，相应的报告得分为“不及格”；
3. 总体报告得分各小组成员相同，在总体报告中应给出各成员负责的模块，详细报告根据各人的任务单独给分。
4. 多人协作完成一个完整的系统，经验收功能完善且几乎没有错误，则组内各成员验收得分都为优；如某模块功能不完善或有较多错误，则对负责该模块的成员进行扣分；
5. 小组为 1 人，如果选择完成 1 个或 1 组模块，并能提供测试程序对该模块功能进行测试和演示，如果无法进行系统演示，根据测试结果给验收得分为“良、中、及格、不及格”，原则上不给“优”，不推荐只完成 Interpreter、API 模块，原因是 Interpreter 模块与数据库理论联系不紧密，而 API 层的实现依赖于 Catalog Manager、Index Manager、Record Manager，难以进行测试；
6. 多人协作完成一个系统，但最后系统无法联合运行，则按各人完成他负责的模块进行处理；
7. 若程序编写工作基本完成，但无法运行或无法进行测试，则根据程序质量给验收得分为“中、及格”
8. 若基本上未编写程序或程序纯属抄袭，验收得分为“不及格”