MiniSQL总体设计报告

刘轩铭，蔡灿宇，胡洋凡

设计分工

- 刘轩铭 Buffer Manager & Index Manager
- 蔡灿宇 Interpret & Catalog Manager
- 胡洋凡 Record Manager & 整体整合

总体概述

- 编写目的
  - 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、删除以及表的插入、删除、查找。
  - 通过对MiniSQL的设计与实现，提高系统编程能力，加深对数据库系统原理的理解。
- 项目背景
  - 本项目是由刘轩铭，蔡灿宇，胡洋凡组成小组，以课堂大程的形式独立完成的MiniSQL的小型系统。
  - 本组开发人员具备数据库应用与实现的知识，且有较好的C++基础和良好的团队精神。

任务概述

- 功能描述
  - 数据类型
    - 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
    - 支持单属性的主键定义。
  - 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）。
  - 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。
    - 如果能过提供 or 连接为加分项
  - 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。
- 运行环境
  - 实现语言：C++
  - 操作系统：Windows

语句规定

create table

```
create table student (
        sno char(8),
        sname char(16) unique,
        sage int,
        sgender char(1),
        primary key (sno)
);
```

- 注意：
    - 括号内不能有空格
    - primary key结尾不能有逗号
    - 所有符号皆为英文格式

create index

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

- 注意：
    - 括号内不能有空格
    - 每个字段都由空格隔开，上述句子共有六个字段
    - 所有符号皆为英文格式

drop table

```
drop table student;
```

drop index

```
drop index stunameidx;
```

select * from

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

- 注意：
    - 每个字段都要以空格隔开，包括等于号=
    - 单引号内不能有空格
    - 所有符号皆为英文格式

insert into

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

- 注意：
    - 每个字段都要以空格隔开
    - 括号、单引号内不允许有空格

- 所有符号皆为英文格式

delete from

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

- 注意：
  - 每个字段都要以空格隔开
  - 单引号内不允许有空格
  - 所有符号皆为英文格式

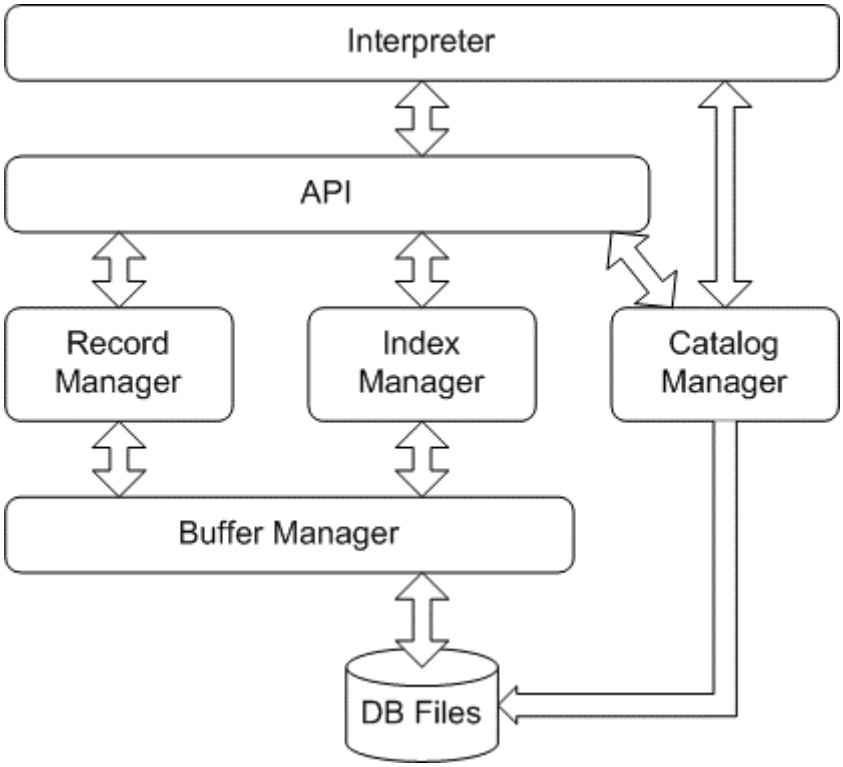
quit与execfile filename

```
quit;
execfile filename;
```

- 注意：
  - 记得末尾的分号

系统体系结构设计

- 系统结构设计中，我们主要是按照实验设计指导书中给定的进行设计，各个模块之间的接口会在下面的模块接口设计中进一步阐述，设计结构图完全参照实验指导上的设计结构，如下：



- **Interpreter**的主要功能是接受用户输入的SQL语句及其他命令语句，并检验用户输入的SQL语句及其他命令语句的格式，语法和语义的正确性。同时将符合要求的语句转化为内部形式，供API及Catalog模块使用；而对不符合要求的语句，显示其出错信息，供用户参考。
- **API**是整个系统的核心，其主要功能是根据Interpreter层解释生成的命令内部形式，调用Catalog Manager提供的信息进行进一步的验证及确定执行规则，并调用Record Manager、Index Manager和Catalog Manager提供的相应接口执行各SQL语句及命令语句。
- **Catalog Manage**r负责管理数据库的所有模式信息。

    - 数据库中所有表的定义信息，包括表的名称、表中字段数、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。
    - 提供访问及操作上述信息的接口，供Interpreter和API模块使用。
- **Record Manager**负责管理记录表中数据的数据文件。
    - 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。
    - 外提供相应的接口。
    - 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。
- **Index manager**是程序的索引部分,直接对buffer manager提供的内存索引块操作。
    - 构建B + 树。
    - 负责实现record manager这一模块（提供函数接口）所需要的函数。
- **Buffer Manager**负责缓冲区的管理。
    - 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。
    - 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
    - 记录缓冲区中各页的状态，如是否被修改过等
    - 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
- 设计思路：

  要实现这样一个小型SQL引擎，几个模块需要各司其职，并且紧密合作，完成每一条命令。经过需求分析，我们首先设计了各个模块的功能以及模块之间的联系，并进行了分工，每人负责不同的模块。完成自己的模块后，进行单元测试，避免在拼接后出现问题。最后进行拼接并设计了一些测试方法来对拼接之后的完整系统进行测试。最终完成的MiniSQL能实现各种简单的命令，能针对各种用户操作的不同情况进行处理，并能将处理结果与处理时间反馈给用户。

接口设计

- 外部接口

  本系统的外部接口为控制台。

  使用者通过控制台的字符界面输入查询语句使用MiniSQL。输入方式为键盘输入。

- 内部接口

  本系统内部模块间采用类创建与函数调用的方式。

出错处理设计

- Interpret的语法分析

  在Interpret模块中采取了相应的方式判断输入是否符合语法，若不符合会提示语法具体错误并重新输入。

- 查询结果异常

  若出现表格、索引已存在或不存在的问题，则会返回出错的具体信息，由Catalog模块抛出，并输出具体错误信息，提供重新输入。