浙江大学

课程名称：	数据库系统
报告题目：	MiniSQL 总体设计报告
指导老师：	陈岭
日期：	2020-6-20
作者：	吉庆雄、赵天辞、应江羽、陈文宇、丁贵州

1. 引言

1.1 编写目的

此总体设计报告是我们为我们要实现的简单数据库系统写的总体设计报告，目的是为了更好的实现我们的 MiniSQL，先对系统进行一个总体的分析与层次的设计，以便提高代码的效率与质量。

1.2 项目背景

本项目是数据库课程的课程大作业，要求我们小组来独立完成一个 MiniSQL 的小型系统。

本组开发人员具有数据库应用与实现的知识，并且有较好的 C++基础，而且有良好的团队精神。

1.3 定义

1．3．1 专门术语

C++：一种面向对象编程语言

SQL：一种访问数据库管理数据库的语言。

1．3．2 缩写

SQL: Structured Query Language（结构化查询语言）。

1.4 参考资料

《数据库系统概念》 Abraham Silberschatz
Henry F.Korth
S. Sudarshan

2.任务概述

2.1 目标

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。

2.2 运行环境

Windows 10

2.3 需求概述

数据类型

只要求支持三种基本数据类型：int 整数类型，char(n)字符串类型，float 浮点数类型。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，而且每个表要求必须有一个主键，以 primary key（属性名）的形式声明。

索引的建立和删除

对于表的主键属性自动建立 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。

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。

SQL 语句需求说明

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”，否则含有特殊的含义，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。

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

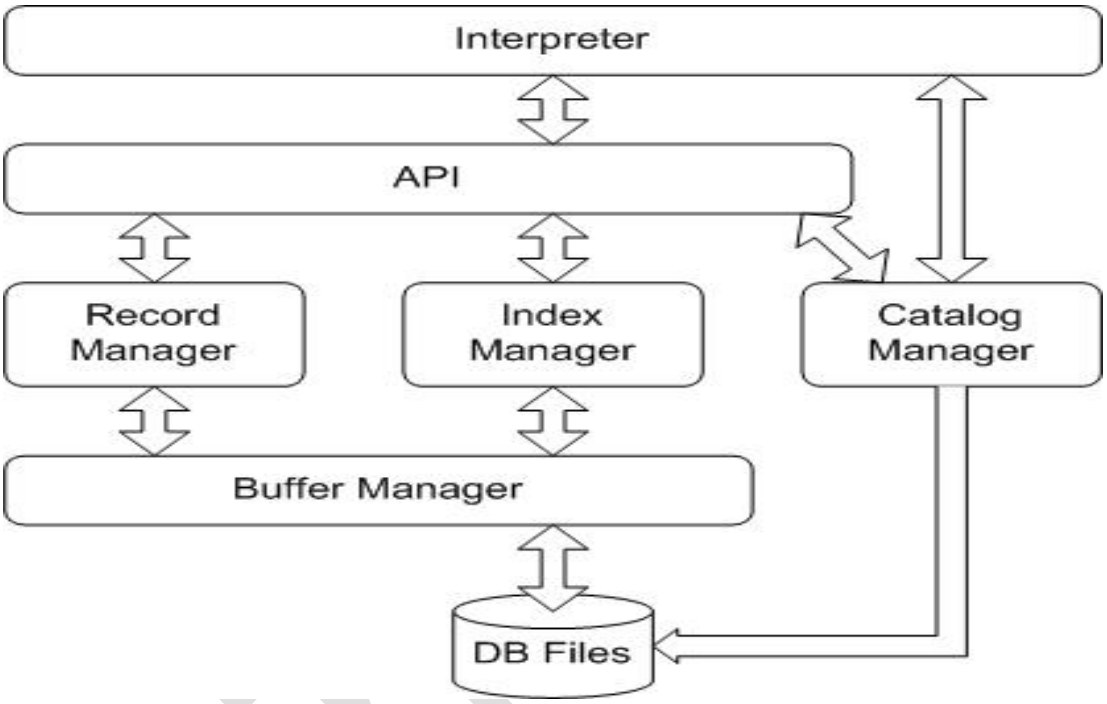
该语句的语法如下：

execfile ‘文件名’；

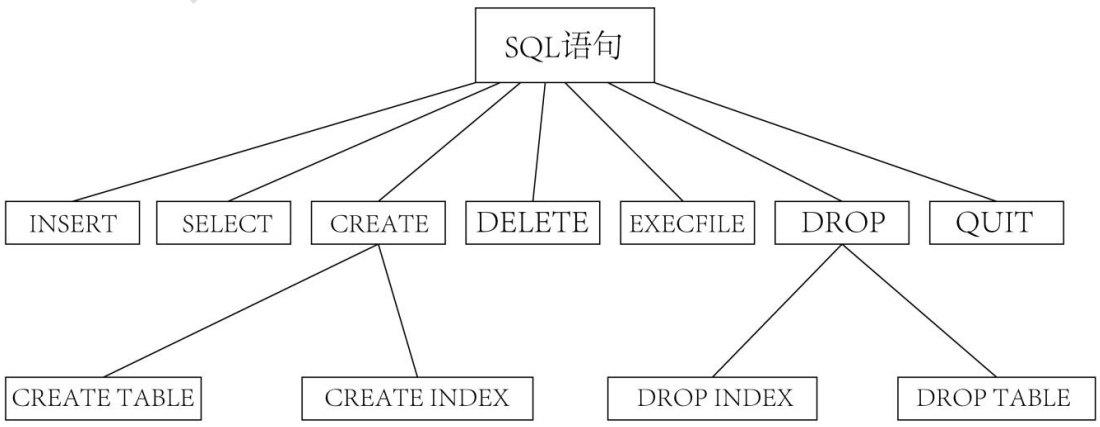
SQL 脚本文件中可以包含任意多条上述 8 种 SQL 语句，MiniSQL 系统读入该文件，然后按序依次逐条执行脚本中的 SQL 语句。

3. 系统体系结构设计：

在系统的结构设计中，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 模块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。SQL 语句的命令解析如下图所示：



API 模块是整个系统的核心，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。API 模块实现功能根据 Interpreter 层解释生成的命令内部形式的语句编号分别调用 Catalog Manager 提供的接口进行语句的语义验证，并根据结果调用 Record Manager,Index Manager 提供的接口执行各语句，同时也为 Record Manager 模块调用 Index Manager 模块的功能提供接口。概而言之，API 是 Interpreter 模块，Record Manager 模块及 Catalog Manager 模块进行沟通的枢纽，三者之间的交互一般都需由 API 模块来进行转接才能进行相互之间的功能调用。因此，虽然 API 模块所实现的功能十分有限，但是该模块却是整个系统的核心。

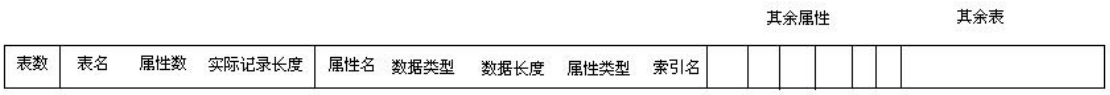
Catalog Manager 负责管理数据库的所有模式信息，包括：

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。
4. 数据表中的记录条数及空记录串的头记录号。
5. 数据库内已建的表的数目。

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。

为减小模块之间的耦合，Catalog 模块采用直接访问磁盘文件的形式，不通过 Buffer Manager，Catalog 中的数据也不要求分块存储。

数据字典文件格式：



Record 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，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）。

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（一个文件中只存储一个表或索引的内容），且不要求支持记录的跨块存储。

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，主要是构建 B＋树，负责实现 record manager 这一模块（提供函数接口）所需要的函数。功能有：存储\删除记录的索引，查找记录在 table 表中的相对位置等（由 buffer 负责写回磁盘）。

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。

记录管理模块（Record Manager）和索引管理模块（Index Manager）向缓冲区管理申请所要的数据，缓冲区管理器首先在缓冲区中查看数据是否存在，若存在，直接返回，否则，从磁盘中将数据读入缓冲区，然后返回。

最近最少使用(LRU)算法：用一个链表记录所有的缓冲块，每次访问到一个缓冲块就将它插入到链表的头部，这样链表尾的缓冲块就是最近最少使用的块，在需要的时候就可以替换出去。

改进：换出脏块时需要写出块，考虑优质换出非脏块。

4.接口设计

4.1 外部接口

本系统的外部接口为控制台界面。

用户可以在打开此系统后显示的控制台界面输入 SQL 查询语句使用此 MiniSQL 数据库系统，输入方式仅为键盘输入。

当开始输入一条 SQL 语句之前，控制台会输出一句“MiniSql->>”作为输入提示字符串，而且当一条 SQL 语句没有在一行内输入完时（即没有读入 SQL 语句结束符“；”——分号）就按回车键，此时会输出一句“>>>”作为继续输入的提示字符串，直到用户输入了一个“；”。

每次执行完一条 SQL 语句，就会在控制台输出执行结果，并且如果有查询结果也会输出来。执行结果会输出执行所需时间和受到更新的记录行数。查询结果会以一张表格的形式输出，第一行为属性名，下面多行为多条对应属性名的记录，并且每个元素以空格间隔。如果执行过程中有异常或错误，会输出出错信息，以供用户做出判断。

4.2 内部接口

本系统内部模块间采用函数调用的方式。

具体描述如：

API 和 Catalog Manager 为 Interpreter 提供各种与创建、删除、查询表有关的函数接口，Interpreter 调用相应的接口来实现用户的要求，并输出结果。调用时，Catalog Manager 利用 Record Manager 与 Index Manager 提供的接口来进行实际的对表的读写。其中，Index Manager 为 Catalog Manager 提供与索引创建、删除、插入相关的接口，Record Manager 为 Catalog Manager 提供读写文件的接口。Buffer Manager 为 Record Manager 提供查找文件的接口。

5. 出错处理设计

本系统出错主要在两方面，一个是 Interpreter 模块的语法分析（比如 SQL 语句不应该含有不相干的字符，输入的 SQL 语句不是规定的语法格式等等），另一个是 Interpreter 模块的语义分析（比如查询的表、索引等不存在，值序列与属性序列不对应等等）。

在 Interpreter 模块，先对输入的 SQL 语句进行语法错误的检查，若出现错误则输出错误信息并重新开始程序，并不关闭程序。如果没有语法错误，则继续调用 Catalog Manager 模块的相关函数对输入的 SQL 语句进行语义分析，检查此 SQL 语句是否与已存在数据库中的信息发生冲突，若出现错误则输出错误信息并重新开始程序，并不关闭程序。如果没有，则执行 SQL 语句。

我们还自定义了一些用于处理错误的异常类，比如 GrammarError、CatalogError、AttributeError 等等。它们专门存放对应的错误信息，并被抛出。

通常，各种错误都是采用异常的形式抛出，在 Interpreter 模块中被捕捉，此模块会根据异常类型以及异常对象内包含的错误信息，对异常对象进行相应的处理，并且输出错误信息。这里最后也是重新开始程序而不终止程序。

另外，此程序对退出语句（quit;）的处理也是通过对异常类（Quit）的抛出、捕捉和处理，来实现程序的退出的。首先，Interpreter 模块会分析出“quit;”语句，然后在执行此语句时创建并抛出一个 Quit 异常类，在主函数的大循环内被捕捉，从而跳出大循环，然后处理完一些后续工作，主函数就结束了，以此来实现程序的结束和退出。

6. 设计分工

此大程的小组分工如下：

API 模块……吉庆雄

Interpreter 模块……丁贵州

Catalog Manager 模块……应江羽

Record Manager 模块……赵天辞

Index Manager 模块……赵天辞

Buffer Manager 模块……陈文宇

DB File 模块……陈文宇

总体设计报告……丁贵州

测试及测试报告……应江羽、吉庆雄