MINISQL 总体设计报告

何浔劲 肖安 戴秉璋 丁治平

1 总体概述

1.1 项目介绍

设计并实现一个精简的单用户 SQL 引擎 MiniSQL，用户可以通过命令行界面输入 SQL 语句，从而实现数据库，数据表以及索引的建立和删除，还有单条数据的插入、检索还有删除。

这次课程大作业通过对 MiniSQL 的设计与实现，提高学生的系统编程能力以及工程项目的合作能力，加深对数据库系统原理的理解。

1.2 参考资料

《Database System Concept》（ Fifth Edition ）高等教育出版社

孙建伶老师《数据库系统设计》课程课件

1.3 开发环境

开发平台：Windows

开发工具：Visual Studio 2015

开发语言：C++

2 功能概述

实现数据类型 ：

int, char(n), float, 其中 int、 float 用 4 个 byte 来存储，char 的长度由用户输入决定。

表定义 ：

一个表可以定义个属性，各属性可以指定是否 unique。支持在单一属性作为 primary key。

索引建立和删除 ：

默认在 primary key 上建立 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。

查找记录 ：

支持 and 和 or 连接的两个或两个以内的条件进行查询，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。

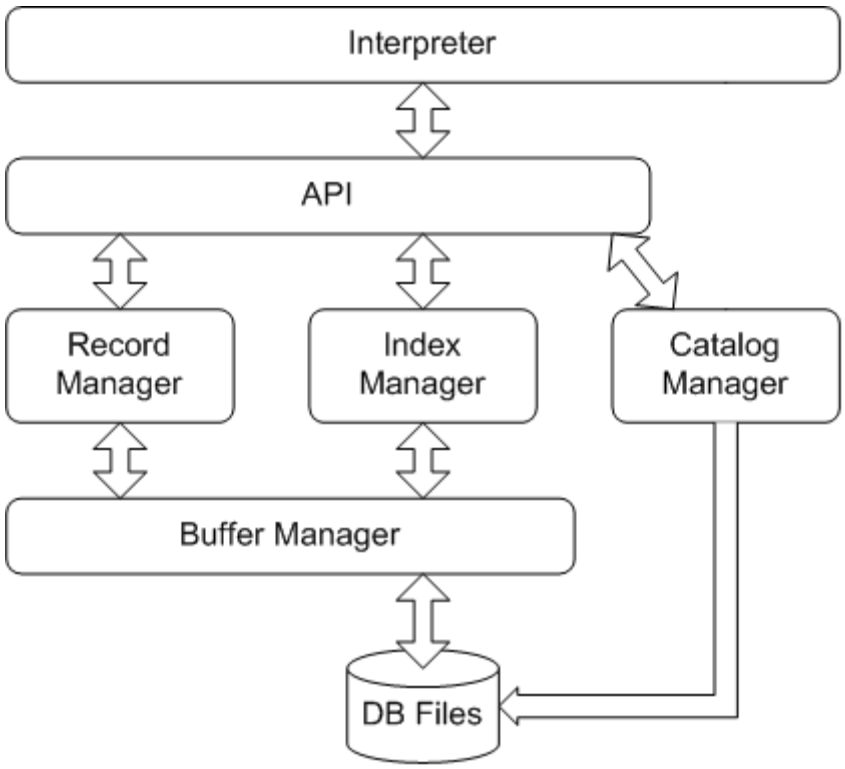
支持语句（ 忽略大小写 ）：

语句	语法	示例	注
使用数据库	use database 数据库名	use database db	使用指定的数据库
创建表	create table 表名 (列名 类型 , 列名 类型 , 列名 类型 , primary key (列名));	create table student (sno char(8), sname char(16) unique, sage int, sgender char (1), primary key (sno));	创建一个新表
删除表	drop table 表名 ;	drop table student;	删除指定的表
创建索引	create index 索引名 on 表名 (列名);	create index stunameidx on student (sname);	在指定的表的属性上建立索引
删除索引	drop index 索引名 ;	drop index stunameidx;	删除指定索引
选择	select * from 表名 ; 或： select * from 表名 where 条件 ;	select * from student; select * from student where sno = ‘88888888’; select * from student where sage > 20 and sgender = ‘F’;	执行查询语句，其中“条件”具有以下格式： 列 op 值 （ and/or 列 op 值 ）。 op 是算术比较符：= <> < > <= >=
插入	insert into 表名 values (值 1 , 值 2 , ... , 值 n);	insert into student values (‘12345678’,’wy’,22,’M’);	插入一条记录
删除	delete from 表名 ; 或： delete from 表名 where 条件 ;	delete from student; delete from student where sno =‘88888888’;	删除指定记录
执行脚本	execfile 文件名	execfile test.txt	批量执行脚本中的命

			令
输出所有表格名	show tables	Show tables	在控制台中输出所有存在的表格的名字
查询表格结构	describe 表格名	describe student	输出指定的表格，包含每个属性的名字、是否为 unique、是否建立 index 等信息
退出	exit	exit	退出数据库，关闭文件。

3 系统结构设计

系统结构设计主要参照给定的实验大纲进行设计，在后期用 MiniDB 类将所有组件包装起来，并且将 Interpret 的输入全部交由 API 层进行分类，然后分别交给 Record Manager， Index Manager 还有 Catalog Manager 进行操作。也就说将给定的设计中 Interpreter 到 catalog Manager 的链接删去了。



3.1 INTERPRETER 模块

Interpreter 负责接收用户输入并向 API 层传递截取好的字符串，主要功能有：

1. 解析用户输入的命令，构建 sql_node，提供命令类型，命令参数长度等信息；
2. 检测输入命令的语法正确性和语义正确性，对错误的输入提示用户发生错误的地方。

3.2 API 模块

API 模块是链接各个模块的核心，它的存在是为了减少各个主要模块间的耦合，在前期各个模块完成并通过内部测试后，在后期将各大模块联系起来完成它们之间的信息传递。主要功能如下：

1. 根据 Interpreter 传递的信息确定执行相应的函数，调用相应的模块；
2. 在执行 Create database 的时候，调用 Buffer Manager 创建同名文件夹；
3. 在执行 Create table 的时候，调用 Buffer Manager 创建数据文件，并调用 Catalog Manger 对 catalog 进行更新；
4. 在执行 Create Index 的时候，调用 Buffer Manager 创建 Index 文件，并调用 Catalog Manger 更新对应表格的信息，如果表中已经有若干数据，还要调用 Record Manager 遍历数据，利用获得的数据调用 Index Manager 对 Index 文件进行更新（B+树的插入）
5. 在执行 Insert 操作的时候，先调用 Catalog Manager 检查插入的内容的数据类型是否和 create table 时规定的相符，将 Interpreter 传入的字符串转换成 Record Manger 执行插入时需要的 Tuple 对象。如果表格中有某个属性为 unique/primary，还要检查要插入的值是否会破坏唯一性，如果要检查的属性都建立了 index，那么检查都通过调用 Index Manager 来完成，否则统一通过 Record Manger 来遍历检查。
6. 在执行 Select 操作时，将 Interpreter 传入的字符串转换成 Record Manager 中的 Selector 对象，从而获得 Tuple 对象进行输出。在输出的时候还要判断用户是否有指定要输出的属性（即投影操作）。
7. 在执行 Delete 操作时，类似 Select 操作，构造 Deleter 对象，从而获取每一条删除的数据，如果表中有某个属性建立的 Index，则根据获得的数据调用 Index Manager 执行 delete 操作（删除 B+树种的结点）。
8. 在执行 Drop index 操作时，调用 Catalog Manager 对 catalog 进行更新，并调用 Buffer Manager 中的函数直接删除 Index 文件。

9. 在执行 Drop table 操作时，先判断表格中建立了 Index 的属性，对这些属性执行 drop index 操作，然后调用 Buffer Manager 删除对应的数据文件。
10. 在执行 Show table 操作时，直接调用 Catalog Manager 获得当前所有表格的名字。
11. 在执行 describe table 操作时，调用 Catalog Manager 查询指定表格的信息。
12. 在执行 exit 操作时，调用 Buffer Manager 关闭文件，调用 exit 函数退出程序。

3.3 RECORD MANAGER 模块

Record Manager 负责 SQL 语句的具体执行，并对外提供相应的返回值，主要功能有：

1. 执行 Insert 语句向 block 中写入数据。
2. 执行 Select 语句从 block 中读出数据。
3. 执行 Delete 语句从 block 中删除数据。
4. 提供检查 unique/primary 属性唯一性是否被破坏的函数。
5. 修改 block 之后将 dirtybit 置为 true，为 Buffer Manager 的替换提供信息。

3.4 INDEX MANAGER 模块

Index Manager 负责索引相关的事务：

1. 索引的插入和删除操作。与 Record Manager 相似，需要从 Buffer Manager 获得对应的 block 进行读写，最后数据写回磁盘则由 Buffer Manager 负责。
2. 利用索引进行搜索。在选择条件包含建立了索引的属性时，能够通过 Index Manager 得到符合条件的数据在表格中的行号。
3. 在修改了对应的 block 之后要负责将 dirtybit 置为 true。

3.5 BUFFER MANAGER 模块

Buffer Manager 负责管理缓冲区和文件管理，主要功能有：

1. 负责文件夹以及文件的创建和删除；
2. 程序开始时初始化内存，在运行中提供给 Record Manager 和 Index Manager 所需的 block；

3. 记录各个缓冲区的状态，实现 Buffer Pool 的 LRU 替换策略。

3.6 CATALOG MANAGER 模块

Catalog Manager 负责管理数据库的"元数据"，以下是它的主要功能：

1. 记录数据库中所有表格的定义，包括表名，各个属性的名称，数据类型，占用空间，是否为 unique/primary key，是否创建 index。
2. 给其他模块提供接口，可以通过表格名称检索到表格的各种信息。
3. 提供对表格信息进行格式化输出的接口。

Catalog Manager 也进行文件的读写，但是不经过 Buffer Manager 的处理，因为所需空间较小，自己独立管理文件即可。

4 错误处理设计

对于整个 MiniSQL，出错有两方面，一方面是用户输入的语法错误，另一种是系统逻辑上的错误，包括用户要访问不存在的表格，输入的数据破坏了 unique 属性等等。

1. 对于用户的输入错误，统一交由 Interpreter 抛出异常提前结束；
2. 对于其它逻辑上的错误，各个模块编写相应的继承 c++标准库的异常类并重载 what 函数，在上层统一通过 catch 标准 exception 对象，然后调用对象的 what 函数输出错误信息，基于多态的特性，可以产生各种不同的输出来提示错误的原因。

在 main 函数中分开处理即可：

```
int main(int argc, char *argv[])
{
        MiniDB db;
        while (true) {
                try {
                        db.readinput();
                }
                catch (IllegalCommand e) {
                        e.DBerror();
                }
                catch (const exception& e)
                {
                        cout << e.what() << endl;
                }
        }
}
```

5 组员信息与分工

姓名	学号
何浔劲	3130100676
肖安	3130104006
戴秉璋	3120103469
丁治平	3120000098

模块	负责人
Interpreter	肖安
API	何浔劲、肖安
Catalog Manager	丁治平
Index Manager	戴秉璋
Buffer Manager	肖安
Record Manager	何浔劲、肖安
测试&脚本编写	全员
总体设计报告	何浔劲