**MiniSQL总体设计报告**

# MiniSQL系统概述

## 背景

### 项目名称

MiniSQL 数据库系统设计与实现。

### 项目背景

数据库系统设计与实现实验。

主要目的：

通过实验加深对DBMS及其内部实现技术的理解。

实践系统软件开发的工程化方法。

### 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。

### 相关资料、缩略语、定义

MiniSQL: Mini Structural Query Language Engine

API: application programming interface.

## 功能描述

1. 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 。

1. 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。

1. 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）。

1. 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。

1. 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。

1. 执行文件

支持将SQL存在.sql文件里，批量处理这些语句。

**特殊说明：**MiniSQL仅支持报告内语法语句，每一条语句以分号结尾，一条语句可以写在一行或者多行。此外，MiniSQL的所有关键词仅支持小写，而且关键词不可被当做表的名字使用。我们仅限使用a~z,A~Z,0-9,\_共计63个字符。

## 运行环境和配置

Windows XP/7/8/10，Microsoft Visual Studio 2015 pro。

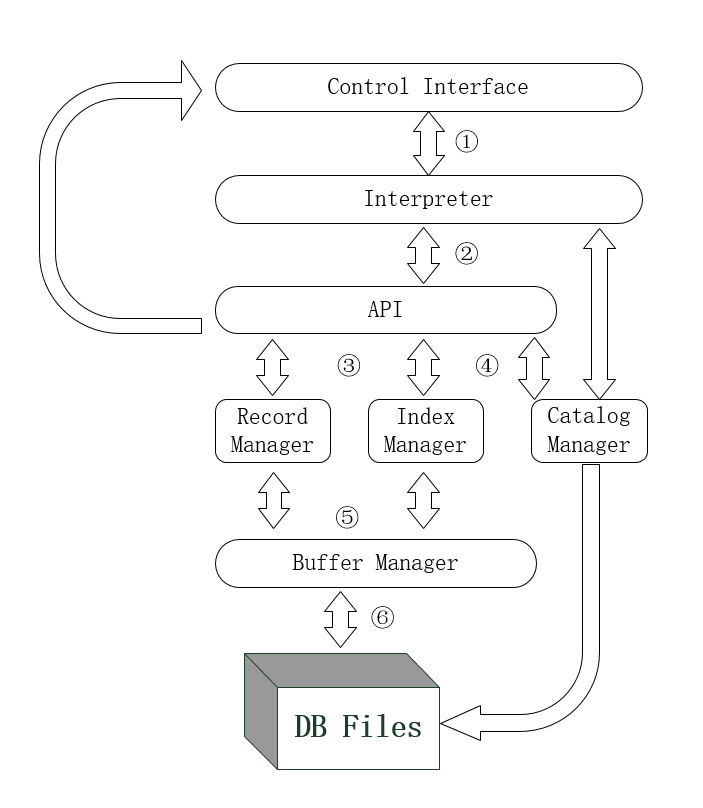
## 参考资料

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

# MiniSQL系统结构设计

## 总体设计：

系统体系结构



**Figure 1：MiniSQL 系统结构图**

图中各标号简明解释：

1. 判断并接受用户字符输入，作为解释器的输入
2. 解释器产生操作数
3. 执行选定的API，返回输出
4. 各模块注册至API，并整合成适合用户调用的API
5. 各模块调用Buffer类实现方法
6. Buffer类对数据库文件的操作

## 系统目录结构

如Figure2图所示，其中：

bin目录存放可执行文件

Dosc目录存放系统开发文档

record目录存放数据记录文件，每个文件的名称为:

（数据库名）\_（表名）.record

catalog目录存放元数据，关于表的数据的文件名称为：

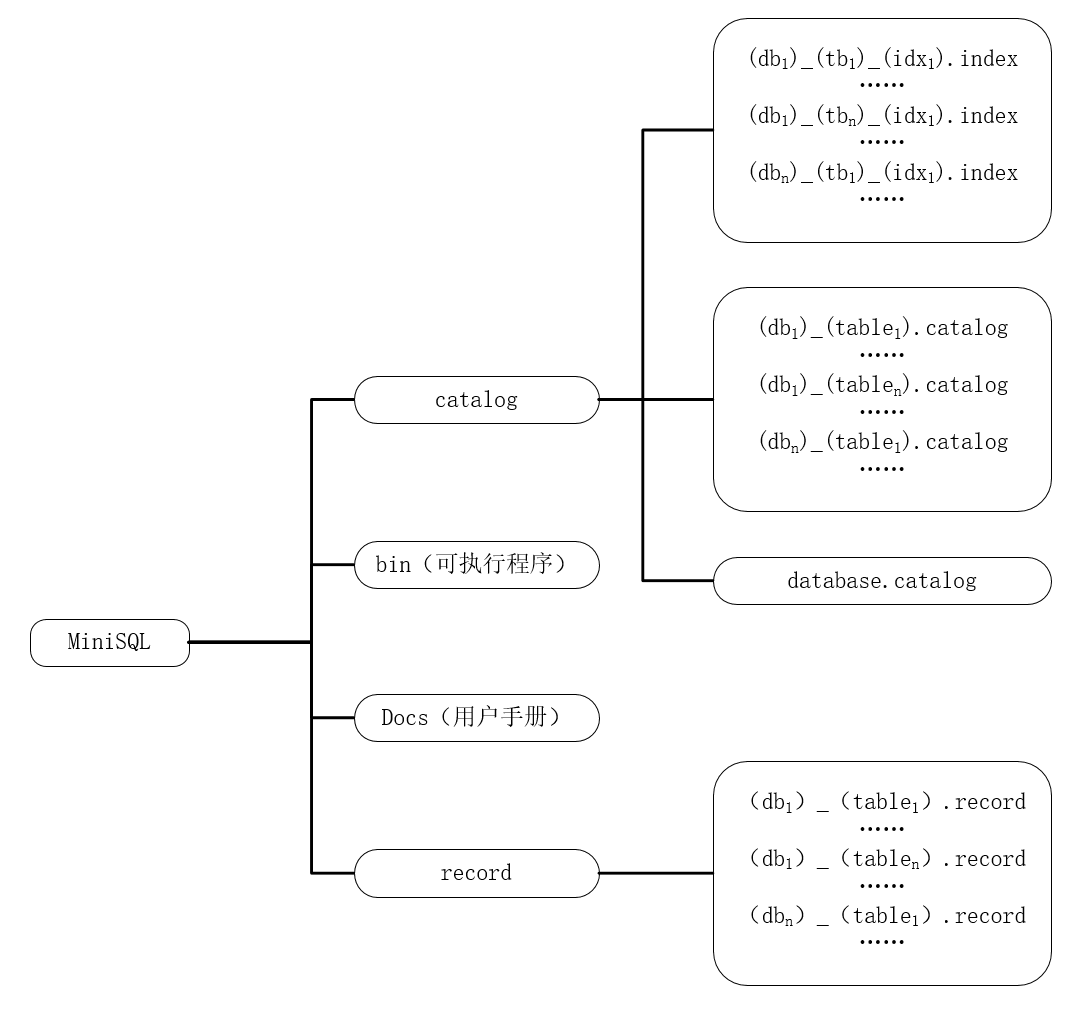
（数据库名）\_（表名）.catalog

关于索引的数据的表的名称为：

（数据库名）\_（表名）\_（索引名）.index

关于所有数据库元数据的文件名称为：

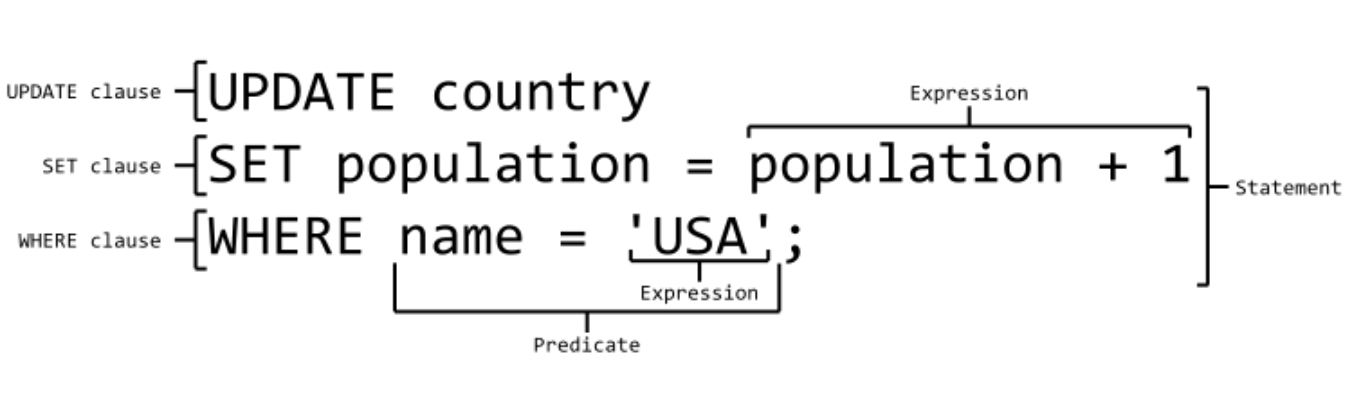
database.catalog



**Figure 2：MiniSQL 系统文件结构**

## Interpreter 模块：

**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。help指令帮助用户书写规范语句。



**Figure3：update语句分析**

Interpreter模块语句：

1. **Create语句**

创建数据库

**create database** database\_name;

创建表格

**create table** table\_name(

列名类型，(unique)

列名类型，

列名类型，

**primary key**(列名)

);

创建索引

**create index** index\_name **on** table\_name(列名);

1. **Drop语句**

删除数据库

**drop database** database\_name;

删除表格

**drop table** table\_name;

删除索引

**drop index** index\_name;

1. **选择语句**

语法：

**select \* from**表名;或**select \* from**表名**where**条件;

其中条件：列op 值and 列op 值…and 列op 值。

op: 0：=，1：<>，2：<，3：>，4：<=，5：>=

1. **插入语句**

语法：

**insert into** table\_name **values** (value1,value2,value3,…,valuen);

1. **删除语句**

语法：

**delete from** table\_name; 或**delete from** table\_name **where** 条件

1. **退出语句**

语法：**exit；**

1. **执行脚本文件格式**

语法：**source** file\_name;

1. **使用数据库的命令**

语法：**use** database\_name;

1. **帮助语句**

语法：**help；**

## API模块：

API模块是整个系统的核心模块。API调用Interpreter模块所生成的对执行语句的初步解析，根据Catalog Manager中存储的信息来对语句做进一步的解析验证及确定执行规则，并调用Catalog Manager、Record Manager和Index Manager提供的相应接口执行各SQL语句及命令相应的功能。API和三个模块进行交互和执行，保证Minisql功能的正常使用。

**具体实现：**API从Interpreter获得解析生成的命令内部形式的语句，并解析出Interpreter模块返回的命令编号，先根据需要调用Catalog Manager提供的接口进行语句的语义分析及验证，后根据结果调用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模块是这三个模块进行沟通的枢纽，三者之间的交互一般都需由API模块来进行功能调用。

## Catalog Manager模块：

Catalog Manager模块负责管理数据库的所有元数据即模式信息，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、定义在表上的索引
2. 数据库中每个字段的定义信息，包括字段类型、是否unique等
3. 数据库中所有索引的定义，包括所属表、建立索引在哪个字段上等
4. 数据库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总数目

Catalog Manager还必须提供访问及操作上述信息的接口，供Interpreter和API模块使用。为了减小模块之间的耦合，我们的Catalog Manager模块采用直接访问磁盘文件的形式，保存了数据库的相关信息，不通过Buffer Manager，Catalog中的数据也不需要块存储。

**数据字典文件格式：**

database.catalog

数据库数量

数据库名1 表数量 索引数

数据库名2 表数量 索引数

…

(DBName)\_table.catalog

总表数

表名1 属性数 记录总数

属性名1 类型 属性（最大）长度 是否主键 是否唯一

属性名2 类型 属性（最大）长度 是否主键 是否唯一

…

表名2 属性数 记录总数

…

(DBName)\_(tableName).index

总索引数

索引名1 表名 属性名 长度 空块号（freenum） firstType

索引名2 表名 属性名 长度 空块号（freenum） firstType

…

设计思路：在数据库开始运行时将存储在磁盘里的数据字典信息读入内存，在其他模块需要时调用并提供信息。程序退出时调用析构函数将改动后的数据更新并存入磁盘。

## 2.5 Record Manager模块：

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）。

Record Manager通过调用Buffer Manager的接口实现对数据文件的操作。主要方法为读取指定文件指定快号的内容。数据文件一般由多个数据块组成，块大小和缓冲区块大小相同，即为4096B（4KB）。一个块中包含一条至多条记录，为了简单起见，数据文件只要求支持定长记录的存储，不要求支持记录的跨块存储。数据在文件块中的具体存储方式为attr1,attr2,attr3……

Record Manager通过调用Index Manager的接口来获得记录的索引。主要方法为读取指定文件指定值的块号和块内偏移量。有等值查询和区域查询下不同的索引获取方式。传入的索引结点包含索引名和属性值。

**选择（执行查询）语句：**

由API调用不带条件的查询或带一个条件的查询。前者输出指定表中的所有记录。后者先判断是否有可用的索引，调用无可用索引的查询或者有可用索引的查询。

若条件是不等于某值，则不通过Index Manager中的B+树结构。

无可用索引的查询中程序通过Buffer Manager模块依次读取每个非空块，对块中的每条记录进行条件判断，决定是否输出。有可用索引的查询中函数根据不同的比较条件分别调用等于索引的查询，大于或大于等于索引的查询或小于或小于等于索引的查询。

等于索引的查询中使用Index Manger中的函数来获取值相等的一个或多个记录的索引并返回，通过索引中包含的块号和块内偏移量来读取文件的指定块号指定偏移位置的记录。

大于或大于等于索引的查询中先使用Index Manager中的一个函数获取第一个索引，之后沿着底部链表把向右第一个及后面所有的偏移量都存入缓冲区内。

小于或小于等于索引查询与大于或大于等于索引查询类似。

**删除语句：**

由API调用不带条件删除或带条件删除。前者删除指定表中的所有记录。后者判断是否有可用的索引，调用无可用索引的删除或有可用索引的删除。

无可用索引的删除中程序通过Buffer Manager模块依次读取每个非空块，对块中的每条记录进行条件判断，决定是否删除。有可用索引的删除中函数根据不同的比较条件分别调用等于索引的删除，大于或大于等于索引的删除或小于或小于等于索引的删除。

等于索引的删除中使用Index Manager中的函数来获取值相等的一个或多个记录的索引并返回，通过索引中包含的块号和块内偏移量来到达文件的指定块号的指定偏移位置，并删除该条记录。

大于或大于等于索引的删除中先使用Index Manager中的一个函数获取第一个索引，之后向右沿着链表将所有键值及对应偏移量删除，直到此B+树叶节点无后续数据。

小于或小于等于索引删除与大于或大于等于索引删除类似。

**插入语句：**

根据表名打开对应B+树文件，调用Index Manager中定义的插入方法。

## Index Manager模块：

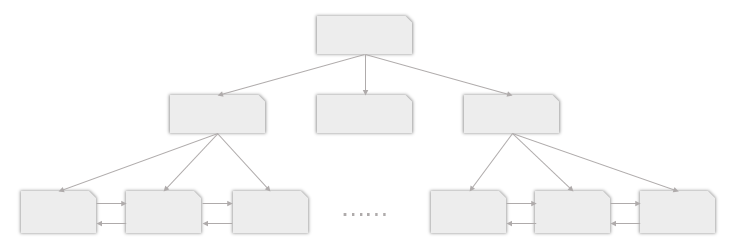
Index Manager负责B+树索引的实现，实现B+树的创建和删除（由索引的定义与删除引起）、等值查找、插入键值、删除键值等操作，并对外提供相应的接口。

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。

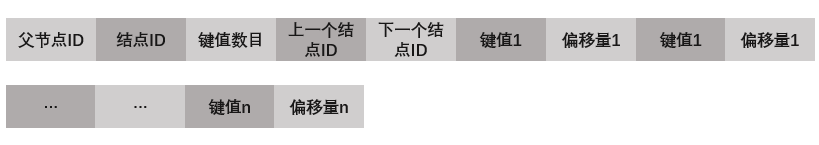
Index Manager通过构建B+树，完成Record Manager模块函数接口的实现，查找记录在table文件中的相对位置并由Buffer Manager负责写回磁盘。

其中B+树使用C++中的模板类以支持存储所有数据，并将叉数和类型均作为参数传入构造函数。B+树维护了结点类型、叉数等信息以及插入（索引）、删除（索引）、查找（使用索引）等方法。

Index Manager模块B+树的结构如下图所示：



**其中叶节点结构：**



其中偏移量为对应键值所在块的位置。

**内部结点结构：**



**创建索引操作/语句：**

创建表时，自动对主键和声明为unique的属性创建索引，其他情况根据create index语句所调用API调用函数接口，通过传入表名、类型名、类型大小计算叉数，创建B+树文件，存入叉数和类型名。

**删除索引语句：**

直接删除对应的索引文件并更新table文件内相关索引信息。

**判断是否违反约束：**

类似于等值查找，在叶子节点和非叶子节点中，一旦读到一个已存在的相同的值，立即返回false。

**实现索引结构的物理存储：**

按照格式化顺序，调用载入函数接口将B+树结构中信息进行相应的映射和格式转换，读到并且对每个字符串进行相应的映射和格式转换。第一个读到的结点为根节点。

**索引结构的物理载入：**

按照格式化顺序，把读到的文件划分为一个个字符串，并且对每个字符串进行相应的映射和格式转换。第一个读到的结点为根节点。

**声明：**Index Manager 中B+树的插入及删除实现算法参考了**MySQL**的innodb引擎中的B+树。

## Buffer Manager模块：

Buffer Manager负责缓冲区的管理，主要功能有：

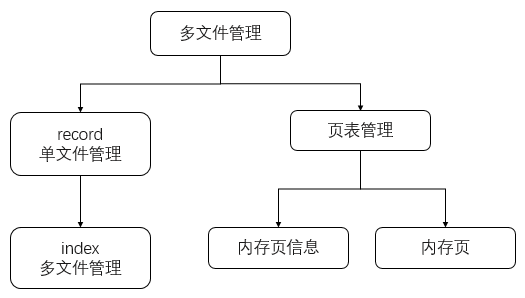
1. 根据需要，读取指定的数据到系统缓冲区或将缓冲区中的数据写出到文件
2. 实现缓冲区的替换算法，当缓冲区满时选择合适的页进行替换
3. 记录缓冲区中各页的状态，如是否被修改过等
4. 提供缓冲区页的pin功能，及锁定缓冲区的页，不允许替换出去

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），我们的MiniSQL将块大小定为4KB。

Record Manager和Index Manager向缓冲区管理申请所需要的数据，以一个内存块为基本单位，Buffer Manager首先在缓冲区中查看数据是否存在（根据文件名和块号定位），若不存在，直接返回。否则从磁盘中将数据读入缓冲区，然后返回。

缓冲区的替换算法使用LRU（最少最近使用）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其插入到链表头部，链表尾部的缓冲块就是最近最少使用的块，在必要时可替换出去。

Buffer Manager部分的结构框图大致如下：



**Figure4：Buffer Manager结构**

说明：1） Buffer模块需要一个全局变量表示当前正在使用的数据库。

2） 支持单个数据库文件的多文件操作，如果需要切换数据库，则要通过函数实现参数的更改。

3） Buffer Manager可以实现文件物理页头信息、文件指针位置、内存页、内存页信息、页表、数据库当前操作信息的管理，缓冲区中替换算法使用LRU算法。

4） MiniSQL开始运行时，先将database.catalog中的数据读入缓冲区以便接下来对数据库的使用等操作快速进行，选择对应数据库时，不仅要将该数据库table.catalog读入缓冲区，还应将部分索引信息及记录信息读入缓冲区中，并记录页及数据的状态位置，在需要时进行页的替换进行缓冲区的管理。

5） 替换页时根据需要将更新后的数据写入磁盘并和磁盘进行交互。

# 测试方案和测试样例

**（1）创建数据库语句**

**正确：**

create database abc;

**错误：**

1.空字符串

create database ; //空字符串

2.以关键词作为数据库名字

create database create; //以关键词作为数据库名字

**（2）使用数据库语句**

**正确：**

use abc ;

**错误：**

1.不存在的数据库

use ??? ; //不存在的数据库

**（3）创建数据表**

**正确：**

create table stu

(

id int ,

name char(20) unique,

age int ,

GPA float,

primary key(id)

);

**错误：**

1.没有使用database 不能插入数据

2.缺少table名

create table //缺少table名

(

id int ,

name char(20) unique,

age int ,

GPA float,

primary key(id)

);

3.内部符号错误

create table stu

(

id int ,

name char(20) unique; //内部符号错误

age int ,

GPA float,

primary key(id)

);

4.缺少数据类型

create table stu

(

id int ,

name char(20) unique,

age , //缺少数据类型

GPA float,

primary key(id)

);

5.数据库名重复

create table stu //数据库名重复

(

id int ,

name char(20) unique,

age int ,

GPA float,

primary key(id)

);

6.数据类型不存在

create table stu

(

id int ,

name char(20) unique,

age long, //数据类型不存在

GPA float,

primary key(id)

);

**（4）执行文件**

**正确：**

source test.txt

**错误：**

1.不存在的文件

source tset.txt

2.文件内有语句不可插入

unique 重复

primary key 重复

**（5）建立索引语句**

**正确：**

create index name on stu;

**（6）选择语句**

**正确：**

select \* from stu;

select \* from stu where id > 2 and age >= 18 and GPA > 4.0;

select \* from stu

where id > 2

and age >= 18

and GPA > 4.0;

**错误：**

1.不知名表、不知名属性

select \* from stu2 where id > 2 and age >= 18 and num > 10;

**（7）删除语句**

**正确：**

delete from stu;

delete from stu where id = 3;

**错误：**

1.不知名表、不知名属性

delete from stu2 where id > 2;

**（8）删除索引语句**

**正确：**

drop index name on stu;

**（9）删除数据库语句**

**正确：**

drop database abc;

**（10）插入语句**

**正确：**

insert into stu values(1,"zhangsan",12,4.00);

insert into stu values(2,"lisi",18,3.96);

insert into stu values(3,"wangwu",20,4.50);

insert into stu values(4,"qinshihuang",198,2.0);

**错误：**

1.不知名表

insert into stu2 values(4,"qinshihuang",198,2.0);

2.重复属性

insert into stu values(2,"qinshihuang",198,2.0);

3.数据过多

insert into stu values(5,"qinshihuang",198,2.0，1000);

# 接口设计

## 与外系统的接口（include头文件）

由各个模块集成API模块，函数主要可分为以下五类功能：

1. 查询功能
2. 插入功能
3. 删除功能
4. 更新功能
5. 数据（索引、表等）定义功能

## 内部接口

用户接口子系统内部接口同4.1。

系统内核内部由Buffer Manager模块向上提供接口。

# 软件模块的物理分布

数据文件存储在磁盘上，index和table等元数据也存储在磁盘上，其他都在内存上操作。

# 用户交互界面设计

控制台模式。

# 分组与设计分工

本组成员：

蔡庆鹏 3150102196

刘纯一 3150103822

本系统的分工如下：

蔡庆鹏：Interpreter模块 Catalog Manager模块 API模块

刘纯一：Index Manager模块 Record Manager模块

合：Buffer Manager模块