# Mini-db

# 概要设计说明书

# 目录

[**1软件基本结构 1**](#_Toc15526)

[1.1软件标识及运行环境 1](#_Toc32423)

[1.2 软件体系结构 1](#_Toc25847)

[**2系统模块设计和划分 2**](#_Toc22651)

[2.1系统模块划分 2](#_Toc18500)

[2.2 系统模块设计 2](#_Toc8412)

[**3接口设计 3**](#_Toc19056)

[3.1用户接口 3](#_Toc29196)

[3.2程序内部接口 3](#_Toc4829)

[3.3外部接口 3](#_Toc7024)

[**4数据结构 3**](#_Toc27352)

[4.1索引数据结构： 3](#_Toc683)

[4.2索引与数据： 4](#_Toc9407)

[**5算法设计 5**](#_Toc20169)

[5.1功能实现 5](#_Toc30988)

[**6技术难点及其解决方案 8**](#_Toc13709)

[6.1遇到的技术难点： 8](#_Toc28894)

[6.2难点对应的解决方案 8](#_Toc17499)

[**7变量和函数等的命名规范 9**](#_Toc29124)

[7.1通用规则： 9](#_Toc2397)

[7.2详细规则： 9](#_Toc23754)

# 1软件基本结构

## 1.1软件标识及运行环境

文件名称：概要设计说明书

项目名称：Mini-db

系统名称：本地迷你数据库操作系统

运行环境：windows7、8、10操作系统

## 1.2 软件体系结构

## data\_structure.h为系统提供数据结构模块

## syntax.h为系统提供语法解析模块

## func.h为系统提供结构设计和功能模块

## support.h为系统提供辅助功能函数

# 2系统模块设计和划分

## 2.1系统模块划分

系统主要分为两大模块：1.索引结构2.语法解析器3.内外存读写模块

## 2.2 系统模块设计

1.索引结构设计：

索引结构选择 AVL索引树

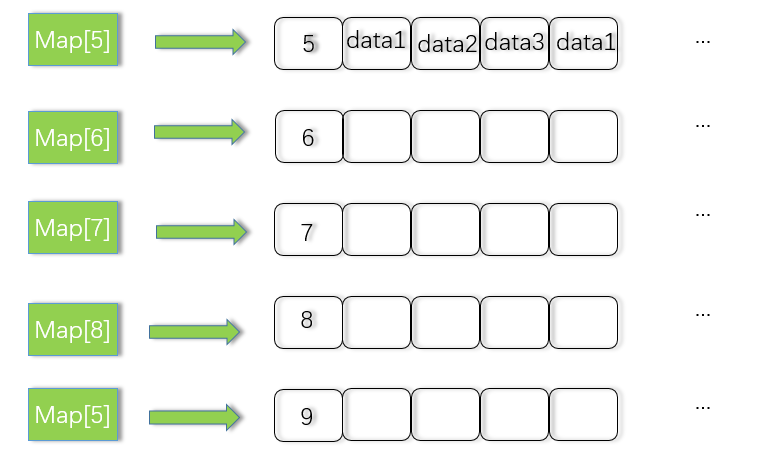
2.语法解析器结构：

用户通过所提示的语法规则在cmd中输入命令，通过语法解析器来调用对应的函数，函数会借助索引模块完成操作命令。

基本操作命令格式为：-x（x为选定的功能）

3.内外存读写模块：

该模块功能有



1. 负责在打开表进行单表操作时，将外存中的数据有组织的读入内存。
2. 在用户所有操作完毕时，将内存中的数据重新写入外存。

## 

# 3接口设计

## 3.1用户接口

提供给用户的接口就是CMD的窗口操作界面，用户通过基于系统的语法命令进行一系列的对数据的操作。

## 3.2程序内部接口

不同的模块进行分别的封装，类内、类与类之间通过函数作为借口进行系统的操作。

## 3.3外部接口

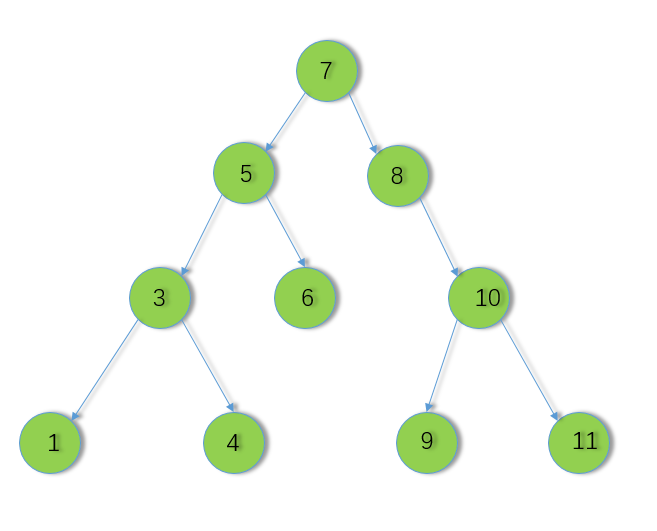
此系统为本地数据库系统，不需要外部接口。

# 4数据结构

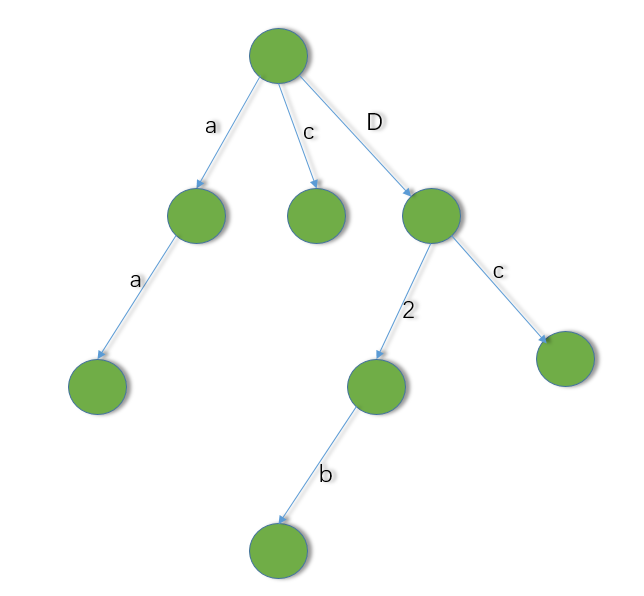
## 4.1索引数据结构：

该表的索引的数据结构为AVL树， 和字典树。

AVL树：



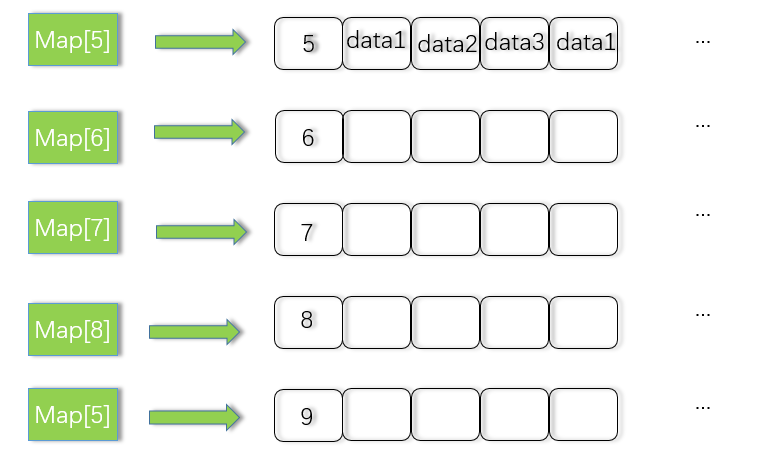
字典树：



## 4.2索引与数据：

首先读取文件的第一部分，即文件的第一行。它保存着文件的基本信息即表头信息和数据类型信息。现在支持的数据类型包括字符型，整型，浮点型，日期。再通过第二部分的数据信息在内存中建立AVL树。同时新建数据节点，并通过map将数据节点和数据编号联系起来。。

在建表操作时，首先对读入的数据进行编号，在建立数据节点并用map将数据节点与数据编号进行对应。



在做检索操作时先通过AVL树查找到相应的编号，在通过map找到数据节点，再对数据进行操作。

## 

# 5算法设计

## 5.1功能实现

-建库

1. 输入库名
2. 建立相应文件夹作为库
3. 返回数据

-删库

1. 确保库内已无表存在
2. 删除相应文件夹
3. 返回数据

-打开库

1. 对输入的库名进行检索
2. 返回内存地址

-关闭库

1. 退出已经打开的库
2. 返回数据

-建表

1.输入表头信息

2如果数据类型为字符型则建立字典树，否则建立AVL树

3.建立数据节点

4.建立map

5.返回建表信息

-删表

1.指定被删除的表

2.按索引树顺序依次释放链表空间

3.初始化索引树和map

4.返回数据

-更改表

1.确定修改的字段

2.根据需求对表头、索引树、数据进行修改

3.返回数据

-查找

1.对输入的表达式进行解析

2.查找索引树（AVL树及字典树， 且字典树支持模糊查询），同时验证索引树中的数据。

3.将查找到的数据编号通过逻辑运算保存在ans\_list的集合中

5.返回数据

-更新

1.确定数据索引序号

2.读出数据所在的数据节点，对指定记录进行修改

3.返回数据

-插入

1.更新索引树

2.新建一个数据节点

3.建立map对应

4.返回数据

-删除

1.确定要删除数据的索引序号

3.更新map

5.返回数据

-order by

1.根据信息查找到要求的数据

2.根据要求对ans\_list中的数据进行建树

3.树的头指针为ans\_tree

# 6技术难点及其解决方案

## 6.1遇到的技术难点：

1.结合自身情况，考虑到完全使用AVL树作为索引树会比较困难

2.语法的解析之前并没有尝试过，且SQL语句对于本系统可能过于复杂

3.一般的数据库由于数据仅仅将数据的关键字（索引）读入内存中，再通过索引指向的磁盘地址进行操作，这样在对非关键字的排序和查找会比较困难。

4.数据的存储格式假如一张表的所有记录都存储在同一个文件中，可能造成外存到内存的读取的算法实现困难比较大。

5.在之前的数据结构上是无法实现模糊查询的。

## 6.2难点对应的解决方案

1.经过计算，由于单表的数据容量较小，即使索引树选择使用较简单的AVL树也可以胜任，所以我们选择AVL树作为索引树。

2.选择放弃复杂的SQL语法，自己选择设计一套简洁的，不复杂的命令语法

3.系统的设计要求仅仅需要在单表中容纳10个字段名（即10列），那么我们决定在数据结构的设计上对每一个字段都建立一棵索引树，方便排序和查找。

4.由于仅仅只有10列，我们选择将每一列（即每一个字段单独成文件），读取就十分方便。

5.在AVL树的基础上添加字典树以实现模糊查询增加了代码的复杂度，要求两种数据结构完美匹配。

## 

# 7变量和函数等的命名规范

## 7.1通用规则：

所有函数、变量、类与文件命名不使用缩写

## 7.2详细规则：

1. 类的命名单词首字母使用大写且不能包含下划线。如：class MyClass

2. 所有变量命名都只能使用小写，单词与单词之间使用下划线间隔。类成员变量使用下划线结尾，函数的局部变量则不用下划线结尾。如：string table\_name;

3.常量使用全部大写。如：PI

4.命名空间应都是小写字母且基于项目名称以及存放路径。

5.通用的函数名使用每个单词首字母大写的方式，对类成员变量存取的函数与类成员变量名匹配。如：MyExcitingFunction()

6.文件名应该包含小写字母以及下划线(\_)。

7.类型和变量应使用名词，而函数应包含动词。如：my\_useful\_class.cc

8.宏命名要以全部大写以及下划线进行命名。如：#define PI\_ROUNDED 3.0