**哈尔滨工业大学计算机学院**

**软件设计与开发实践II**

**DBMS原型系统设计与开发结题报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **班级** | | **1103104** | | **结题时间** | **2013.10.25** | | |
| **指导教师** | | **徐汉川&高宏** | |  |  | | |
| **小组成员** | **姓名** | **学号** | **电子邮件** | | | **分工** | **工作比例** |
| **柴成亮** | **1110310407** | 1064119346@qq.com | | | **组长** | **38%** |
| **万佳林** | **1110310419** | 1010541594@qq.com | | | **组员** | **32%** |
| **马良** | **1110310410** | 403893417@qq.com | | | **组员** | **30%** |

**一、系统设计**

**1.1系统功能**

本小组开发的简易DBMS原型系统仿照MYSQL，完成了关系数据库的一些基本操作，主要提供了建表、查询、投影、插入、删除、更新、连接等操作。

用户可按照附录中的使用说明书进行相关操作。

**1.2系统结构**

本DBMS原型系统将待操作的所有信息存放在磁盘上的文件中，用户点击运行DBMS后，在命令行按照提示输入基本操作信息，系统根据操作类型和操作条件，从磁盘相应位置读取所需信息至内存，进行相应操作后，将信息写到磁盘相应位置，并在命令行上显示返回信息，包括错误提示、操作成功提示等。

**1.3数据库引擎**

**1.3.1 数据存储结构**

1）采用文件存储。对于整体的数据库，建立一个文件，记录数据库中所有的关系表,每个关系表之间用分号隔开。对于每个关系表，建立一个文件，记录其元组，每个元组之间用分号隔开，方便读写，每条元组记录前加一个标志位，代表其是否被删除。每个关系表前记录一个数字，含义是该表在b+树数组的数组下标。

2）采用b+树结构进行索引排序。对于每个关系表，按照其主键建立b+树，这样的索引结构使得查询快速。连接操作中利用b+树按照属性排序，采用归并的方法减少IO次数提高查询效率。

**1.3.2数据存取方法**

1）读取时每次从文件中读取一条元组（字符串）到内存，因为每条元组之间有分号分隔，所以较为容易。

2）存入时同理每次存入一条元组（字符串）到文件，添加分号。

3）系统每次启动时，需要将每个关系表根据表头的数字，生成一个b+树树组。需要导入文件，提取其主键并根据键值大小插入。

**1.3.3数据操作方法**

系统根据主键值在b+树中查找元组在表文件中的偏移位置，将对应位置的一条元组读取至内存，然后扫描这条元组，根据逗号的位置和输入的信息判断需要的属性值。

**1.4系统界面**

本DBMS原型系统采用简易的命令行窗口界面，用户根据系统提示和系统使用说明书来进行所需操作。本小组主要针对数据处理的速度做了一定的优化，例如加入了b+树索引来提升查询速度等，而系统界面方面还可以做得更好。

**二、系统开发**

**2.1关键技术**

1. model文件存储所有关系表的表名：

当命令行操作中出现from语句时，根据其后的字符串查找model中的表名，若存在，则进行后续操作；否则，提示出错。

1. 每个关系表文件头需要有一个整数作该表对应b+树的标示符，按关系表被新建的次序，依次为1,2,3……

3）每个元组前有标示是否被删除的标识符，’~’表示未删除，’!’表示删除。

4）采用了b+树索引结构：

每个关系表对应一个b+树，由一个b+树数组进行存储。当系统开始运行时，所有b+树根据对应关系表进行初始化。对于每个关系表，按照其主键建立b+树，这样的索引结构使得查询快速。连接操作中利用b+树按照属性排序，采用归并的方法减少IO次数提高查询效率。

**2.2 主要模块及接口**

1）模块：

①外部存储器模块

通过文件的方式存储在计算机的硬件设备上，即DBMS的外部存储器模块与机器的硬件存储系统相连接。

②b+树索引模块

打开系统时，根据文件信息自动载入b+树

2）基本数据结构定义：

①b+树基本结构

struct Btree

{

bool flag;

int count;

char key[max+1][100];

int offset[max+1];

Btree\* ptr[max+1];

Btree\* parent,\*next,\*prior;

};

Btree\* b[10],\*b\_join[5];

②存储对应元组在文件中的偏移量

struct Triple

{

Btree\* r;

int locate;

};

③存储where语句中的属性为对应关系表的第几个属性、属性值、比较符及属性类型（适用于where语句和update中的set语句）

struct sel\_where

{

bool flag; //int or char[]

int location;

char ch;

int num;

char buffer[20];

};

④存储查询、插入等操作中用户输入的信息

create操作：table[N]

select操作：select[N] , from[N] ,where[N]

insert操作：sentence[N]

delete操作：table[N], where[N]

update操作：from[N], set[N], where[N]

join操作：table\_1[N], table\_2[N], where[N]

describe操作: from[N]

display操作：from[N]

函数接口：

①int comma(char project\_tmp[],char from[],bool f)

输入：

属性名project\_tmp[]，关系表名from[]，bool型f

输出:

属性名project\_tmp[]在关系表from[]中的位置

②int analysis(char from [],char where [],sel\_where s\_where[],bool f)

输入：

关系表名from[]，条件语句where[]，sel\_where型的结构体数组s\_where[]，bool型f

输出：

s\_where[]数组中存储的元素个数（where语句分析后存入s\_where[]数组中）

③bool pkey\_check(char primary[],int num,char table[])

输入：

主键primary[]，b+树下标num，关系表名table[]

输出：

输入主键primary[]是否已经存在

④void creat()

输入：

系统提示输入建表语句

功能：

在model.txt中新加一条表示关系模型的元组，如student(sid char[20],sname char[10],sage int);并建立相应关系表文件

⑤void insert()

输入：

系统提示输入待插入表名和各属性值

功能：

在相应表文件中新加一条元组，如student(1110310401,zxm,20);

⑥void del()

输入：

系统提示输入待删除表名和条件语句where

功能：

根据相应表文件中每条元组的第一个字符（标示符），若为’!’，则表示已删除；若为’~’，则将其改为’!’

⑦void update()

输入：

系统提示输入待更新表名、设置各属性的新值及条件语句where

功能：

先根据where语句在关系表中查找符合条件的元组，对每条这样的元组，检查其第一个字符（标示符），若为’!’，则表示已删除，提示出错信息；若为’~’，则将其改为’!’，并在文件末尾加入以’~’开头的更新过的元组

⑧void select()

输入：

系统提示输入待查找表名、选择的属性列及条件语句where

功能：

若只有一个表名，则为单表查询，根据where语句扫描相应关系表中的每条元组，输出符合条件的对应属性列；

若有两个表名，则为多表等值连接，根据project()函数和where语句中的等值信息，将需要投影出的各个属性连接成一个新的关系模型，并新建一个关系表，然后写入符合条件的元组

**2.3开发平台、语言及工具**

Win7，C++，codeblocks和VC++

**2.4人员分工与开发进度**

1）人员分工

柴成亮作为组长，负责整个工程的统筹工作，完成了comma（）、analysis（）

等一系列主要函数的实现工作，并提供了b+树，编写了create（）、insert（）、join（）等函数；

万佳林完成了insert（）、delete（）、update（）、select（）的一部分以及错误处理等工作；

马良完成了代码的优化和测试工作，并编写了工程报告。

2）工程进度

4周时间基本完成，然后花2周进行了更详细的测试和代码优化，完成效果不错。

**三、安装与配置**

**3.1系统安装**

无需安装

**3.2 系统配置**

无需配置

**3.3 系统故障处理**

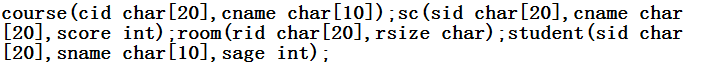
本DBMS原型系统支持主键查重处理，若插入或更新使主键值冲突时，会提示出错信息。

四、系统运行实例

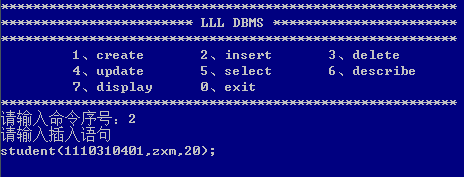
1.create table



新建4个关系表之后，model.txt中为：



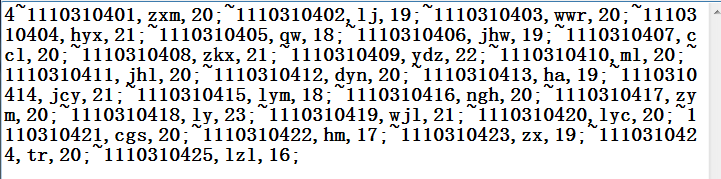
Insert



此时student.txt中为：



插入多次之后,student.txt中为:



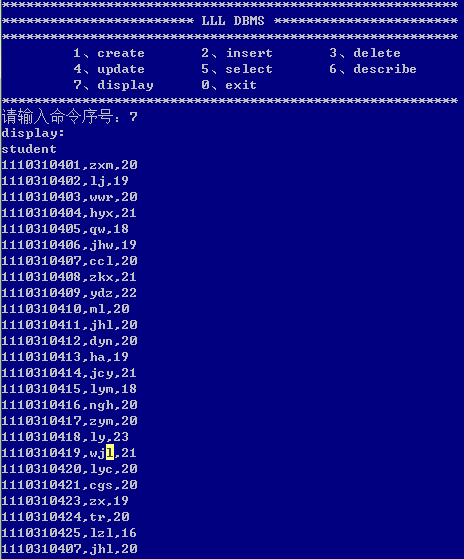
此时再插入主键（第一个属性）相同的元组，则显示出错信息：

Delete

删除姓名为“hm”的学生：



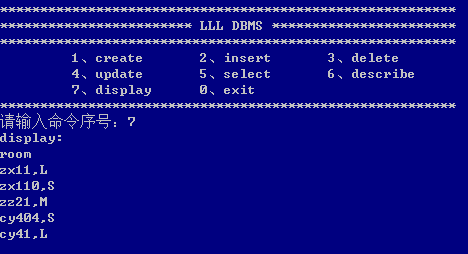
然后显示student的信息（hm已经删除啦）：



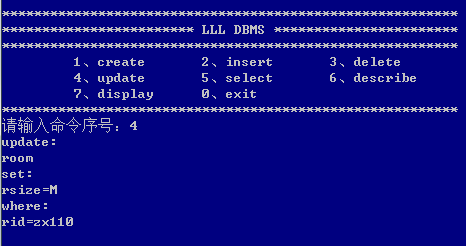
Update

更新教室信息，将正心110（zx110）由小（S）变为中（M）

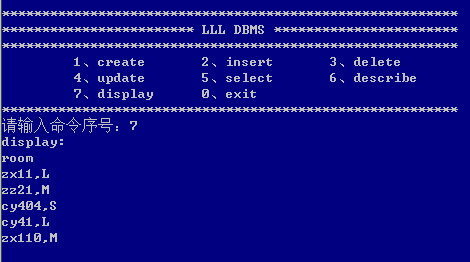
先显示原来信息：



更新操作：

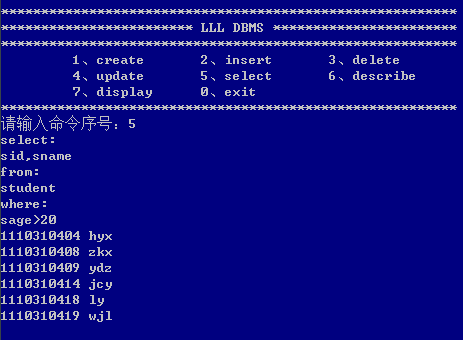


显示更新后的教室信息：

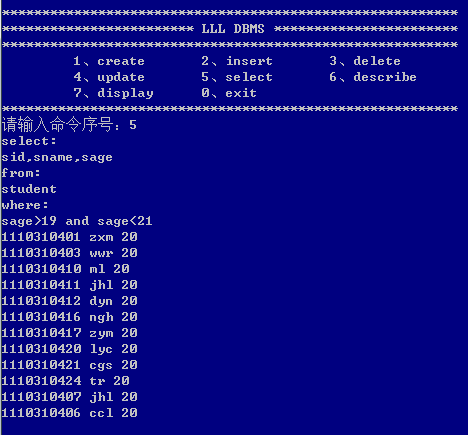


Select

单表单条件：



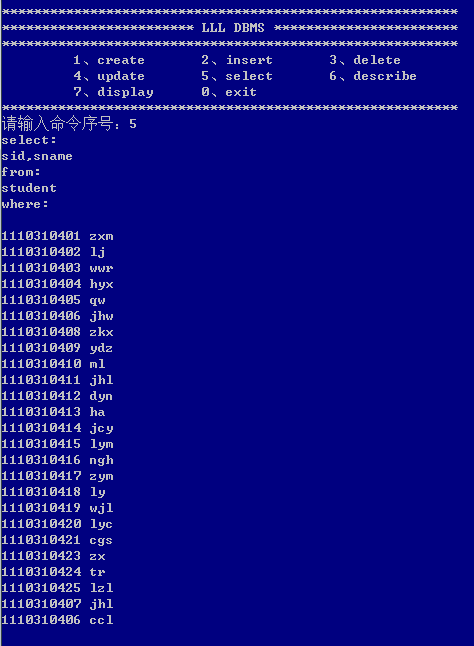
单表多条件：



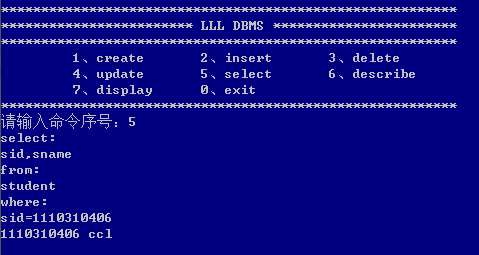
单表\*查询：



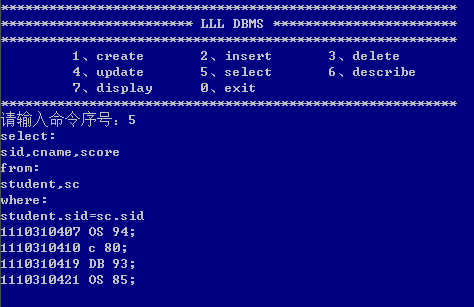
单表无限制条件：



单表主键查询（利用b+树以主键值作索引）：



双表查询：



**五、心得与体会**

**4.1 系统设计方案的不足**

1）系统功能较为简单，未实现所有代数操作和较为复杂的select语句。

2）错误处理不够详细。

3）只是利用b+树给主键加了索引，如果给所有属性都加上索引，可以提高查询速度。

4）可以采用外排序

**4.2 拟解决的问题**

拟解决问题：查询算法的优化

设计开发思路：对于不同的查询条件，为了提高系统的运行速度和查询效率，可采用启发式关系代数优化方法。

拟解决问题：排序优化

设计开发思路：外排序

**本小组全体成员郑重声明：**

**该DBMS原型系统的设计与开发工作由本小组独立完成。若发现任何抄袭之处，本小组愿意承担因抄袭而带来的一切处罚。**

**小组成员签字：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_年\_\_\_\_\_\_\_\_月\_\_\_\_\_\_\_\_日**

**附录**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*DBMS数据库管理系统使用说明书\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

一、使用环境：

Windows下使用codeblocks或VC++等编译运行

二、使用步骤：

打开工程，编译运行，按提示输入相应信息即可。