

**系统能力综合培养实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学 院： | 计算机科学与技术 | | |
| 专 业： | 计算机科学与技术 | | |
| 班 级： |  | | |
| 学 号： |  |  |  |
| 姓 名： |  |  |  |
| 指导教师： |  | | |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

年 月 日

目 录

[1 实验任务概述 1](#_Toc493799738)

[2 记录管理模块 4](#_Toc493799739)

[2.1 模块概述 4](#_Toc493799740)

[2.2 模块设计 4](#_Toc493799741)

[2.3 模块实现 6](#_Toc493799742)

[2.4 实验过程 7](#_Toc493799743)

[3 查询处理模块 14](#_Toc493799749)

[4.1 模块概述 14](#_Toc493799750)

[4.2 模块设计 14](#_Toc493799751)

[4.3 模块实现 14](#_Toc493799752)

[4.4 实验过程 17](#_Toc493799753)

[4 实验总结 18](#_Toc493799759)

[6.1 工作总结 18](#_Toc493799760)

[6.2 改进方案 18](#_Toc493799761)

[附录 19](#_Toc493799762)

# 实验任务概述

## 实验目的

1. 了解数据库、操作系统、编译、数据结构等专业知识在关系数据库管理系统实现中的综合运用；
2. 了解关系数据库管理软件的体系结构和设计方法；
3. 了解关系数据库中数据和元数据的组织方法；
4. 掌握关系数据的底层存储实现技术；
5. 掌握基本关系操作在关系数据上的实现算法；
6. 掌握基本SQL语句的实现方法；
7. 掌握基本的查询优化技术。

## 实验任务要求

基于HustBase系统的总体设计架构，根据预定给定的系统框架、部分模块和接口要求，设计并实现系统中的其余模块功能，完成一个具有基本数据定义、数据操纵和数据查询功能的单用户关系数据库管理系统。

## HustBase系统

HustBase是一个用于实践教学的简化版数据库管理系统，其设计目标是支持简单的SQL语句，提供基本的数据定义、数据操纵、数据查询和查询优化功能。为了便于学生在为期四周的实验周期内掌握系统框架并完成实验内容，与实际的商用DBMS相比，HustBase做了大量的功能简化，舍弃了如多用户、事务、故障恢复等常见特性，支持的SQL语法也非常有限。

HustBase系统是基于多模块开发的单用户简单关系数据库管理系统，其整体的体系架构如图 1.1所示。整个系统由页面管理、记录管理、索引管理、系统管理、查询处理以及语法分析主要模块构成。主要模块的功能简述如下：

1. 页面管理：数据库中所有的数据、元数据和索引数据均以文件的形式存储在磁盘上。为了实现对磁盘文件的高效I/O，需要将流文件转换为记录文件。本模块提供将流文件抽象为分页文件的功能，提供以页面为单位的文件读写接口，具体包括：创建、销毁、打开和关闭分页文件；遍历指定文件中的所有页面；从指定文件中读取一个特定页面；在指定文件中添加、删除及修改页面等。
2. 记录管理：系统表和数据表中的数据均以记录为单位进行存取。本模块在页面管理模块的基础上，将文件中的每个页面又划分为若干个固定大小的记录插槽，实现记录的插入、删除、修改和查找，为上层的数据操纵和查询提供支持。具体包括：创建、销毁、打开和关闭记录文件；插入、删除、修改记录文件；查找符合条件的记录等。
3. 索引管理：利用索引，系统可以为查询提供快速访问路径。本模块在页面管理的基础上，将文件中每个页面又划分为若干个固定大小的索引项插槽，提供对索引项的管理。索引文件与记录文件的不同之处在于，索引采用B+树结构组织索引项，以便提高查找效率。具体包括：创建、销毁、打开和关闭索引文件；索引项的插入、删除、修改以及查找等。
4. 系统管理：提供对数据定义（CREATE/DROP TABLE，CREATE/DROP INDEX）和数据操纵功能（INSERT、DELETE、UPDATE）的支持。本模块的实现依赖于记录管理模块和索引管理模块。
5. 语法分析：提供对SQL命令的语法分析，并将结果以语法树的形式提供给系统管理模块和查询处理模块，以便其做进一步的分析和处理。
6. 用户界面：本模块为用户提供操作界面，接收用户输入的SQL命令，并向用户输出SQL命令的执行结果。

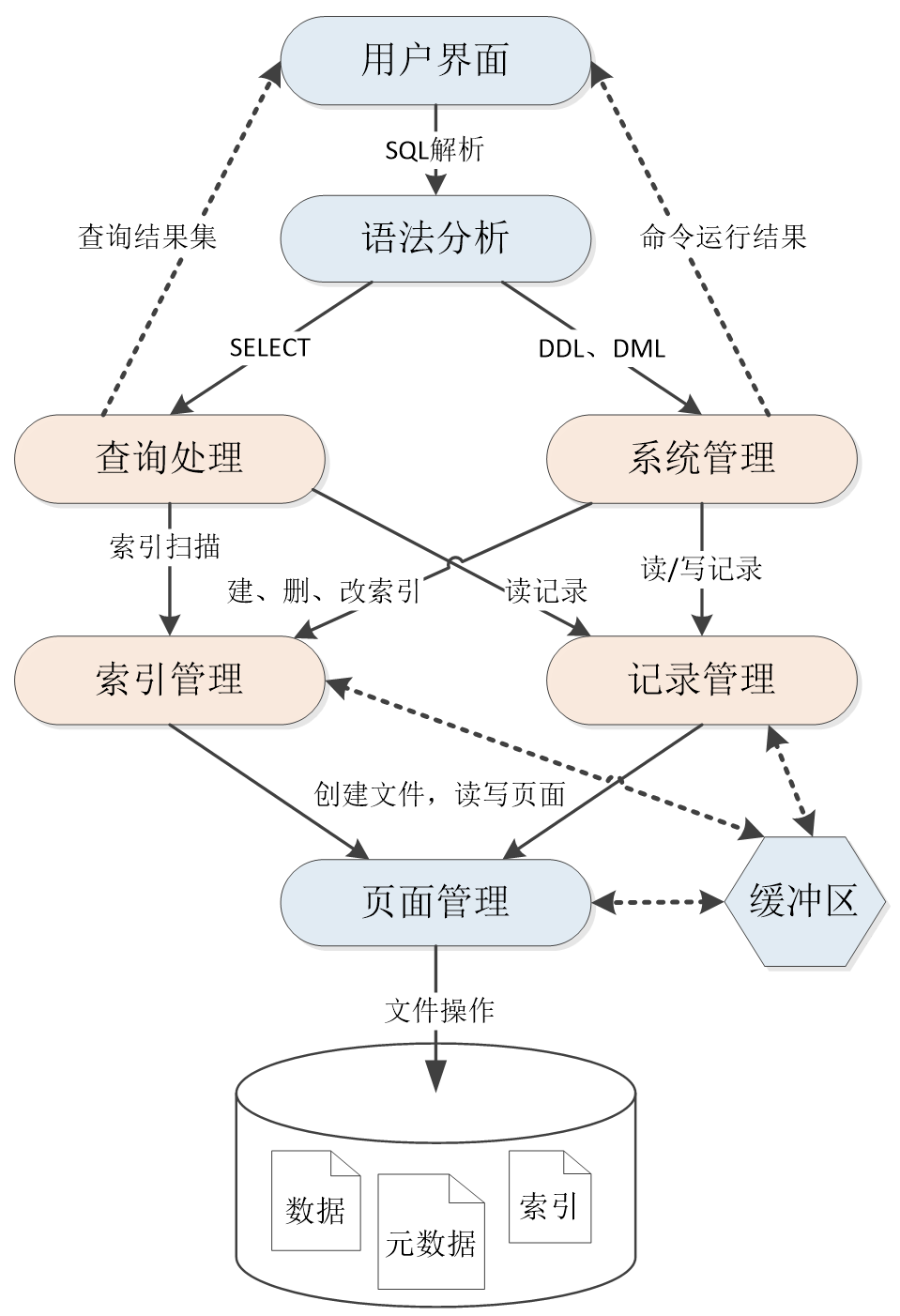


图 1.1 HustBase体系架构图

## 小组成员分工

实验课时安排为四周，按周次划分为四个阶段。除去已预先提供的用户界面、语法分析和页面管理模块，要求学生自己设计实现记录管理、系统管理、查询处理和索引管理模块以及所有模块的联调，完成整个HustBase系统的开发。具体课程内容安排如表1.1所示。

表 1.1 实验内容安排

|  |  |
| --- | --- |
| 周次 | 实验内容 |
| 第一周 | 学习HustBase的系统结构和工作原理，在已提供模块的基础上，开始记录管理模块、系统管理模块和索引管理模块的编码。 |
| 第二周 | 完成建库、建表功能的编码和联调；开始数据查询模块的编码。 |
| 第三周 | 完成数据操纵和数据查询功能的编码和联调。 |
| 第四周 | 完成索引管理功能，与其他相关模块结合，完成系统整体联调。 |

小组分工：

肖奇祥：承担组员A的任务，主要负责记录管理模块和查询模块的任务。

张宗琨：承担组员B的任务，主要负责系统管理模块的任务。

李昕宇：承担组员C的任务，主要负责索引管理模块的任务。

# 记录管理模块

## 模块概述

数据表中的数据由若干条记录构成，最终存储在磁盘文件中。本模块在分页文件的基础上，进一步将页面文件改造为记录文件，每个页面的数据区被划分为若干个记录插槽，每个记录插槽用于存放一条记录，并对外提供以记录为单位的记录文件读写接口。记录管理模块依赖于下层的页面管理模块，页面管理模块以页面为单位进行文件读写，而记录管理模块则以记录为单位进行文件读写，每条记录以页面号+插槽号为唯一标识。为简化实现，规定如下：1. 记录文件中存储的记录为定长记录；2. 同一个文件中的记录格式相同，即一个数据表对应一个文件；3. 记录不能跨页，且单条记录的长度不超过页面数据区大小。

## 模块设计

本模块的设计主要遵循指导书的思路进行设计，设计主要从记录文件的结构和对外提供的功能接口进行介绍。

### 2.2.1 记录文件的结构设计

记录文件的底层是页面文件，通过对页面文件进行改造以便实现以一条记录为单位进行操作，从而很好的支持数据操纵和数据查询的功能。记录文件的结构如图 2.1所示。

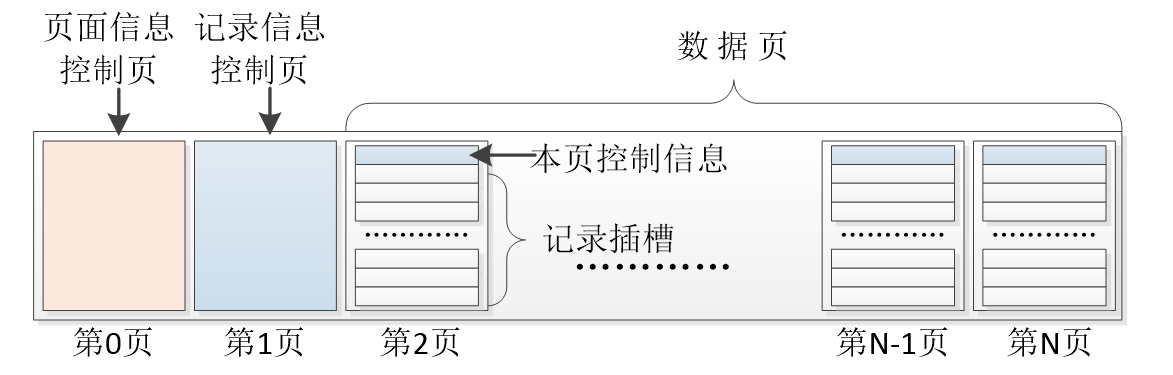


图 2.1 记录文件结构示意图

记录文件在页面文件的基础上进行了处理和优化，对部分页面进行了进一步设计。第0页保持不变，仍然是该记录文件对应的页面信息控制页，存放页面的使用情况。但是将第1页作为本记录文件的信息控制页，存放改记录文件记录的整体信息以及后面页面是否存储记录已满。数据页在普通的页的基础上添加了该页存储记录的位图，表示该页在相应的位置上（记录插槽）是否存储了有效的记录，每一位对应着一个记录插槽。由于创建记录文件时已经知道记录的大小，可以确定每一个数据页可以存放的记录的条数，并且每一数据页存放的记录个数是确定而且相等的。

1. 记录文件信息控制页

记录文件的第1页作为记录信息控制页，记录的信息主要是跟该文件实际存储的记录有关。主要使用如下的数据结构，从该页面Page的数据区pData[0]开始存放该数据结构：

typedef struct {

int nRecords; //当前文件中包含的记录数

int recordSize; //每个记录的大小

int recordsPerPage; //每个页面可以装载的记录数量

int firstRecordOffset; //每页第一个记录在数据区中的开始位置

} RM\_FileSubHeader;

该结构存储了该记录文件处理的记录的基本信息以及每页可以存储的记录的数量以及第一条记录在每页的数据区的偏移位置。对记录的操作提供了一些基本信息，更方便地进行记录地处理。在第一页存放该数据结构后，即从pData[sizeof(RM\_FileSubHeader)]开始存放一个位图，每一位对应一页，用来标记记录文件中的哪些页面为满页，哪些为非满页。可以将第0，1号位置标记为满页。非满页是指页面中还存在空闲记录插槽的页面，位图中0表示非满页，1表示为满页。此位图与页面0中的位图类似，只是含义不同。

1. 记录文件的数据页

从分页文件的第2页开始为数据页面，每个数据页面被划分为若干个记录插槽，记录在数据页面中采用顺序存储方式。此外，为了区分当前页面中哪些插槽已经存放了记录数据，哪些插槽为空，还需要记录相关的控制信息。在每个数据页面中，从pData[0]开始存放一个位图，用于管理该页面中记录插槽的使用情况。如果一个数据页面能够存放n个记录，则需要一个大小为 Sup(n/8) 字节的位图。若某个插槽含有有效记录，位图中相应的位为1；若某个插槽为空插槽，相应的位为0。位图后的空间按记录长度划分为若干个插槽。注意，位图的大小取决于一个页面能够存放的记录数，而一个页面能够存放的记录数又是由记录的大小决定。因此，每个记录文件在创建时都需要根据其存储的记录长度来计算位图所需的字节数。

### 2.2.2 记录文件的接口设计

记录文件的操作主要分为对文件的操作和对记录的操作，主要接口细分如下：

1. 创建记录文件：根据记录的大小创建一个记录文件。
2. 打开记录文件：根据文件名，获取一个记录文件的句柄。
3. 关闭记录文件：关闭文件，关闭该记录文件占用的资源。
4. 插入记录：插入一条新的记录到指定的记录文件中，并且返回记录位置。

插入记录时，首先在文件中找到一个非满页（若没有则需要申请一个新的页面），在该页中找到一个空插槽并插入记录。记录插入后，要更新控制页，即将文件包含的记录数加1，同时将插槽位图中的相应位置1。如果记录插入后，该页面已经没有空插槽，则将该页面标记为满页，即在记录文件的控制页的位图上将该页面所对应的位置1。

1. 获取一条记录：从指定的记录文件中获取指定位置的记录。
2. 删除一条记录：删除指定记录文件中指定位置的记录。删除记录时，对

于一个给定的记录ID，先检查该记录ID的有效性，然后找到相应的记录。记录删除并不需要将页面中记录所占用的区域清零，只需要将插槽位图中将相应的位置0，并把该页面标记为非满页。同时将控制页中的记录个数减1。

1. 更新一条记录：对指定记录文件的指定一条记录进行更新。修改记录时，

对给定的记录ID，先检查该记录ID的有效性，然后找到相应的记录，用新的记录值替换原记录值。

1. 记录扫描：根据给定的条件，返回符合条件的所有记录。此操作分两步

完成，用户先初始化一个指定条件的文件扫描（打开扫描），然后逐个获取符合条件的记录。

## 模块实现

### 2.3.1 重要数据结构

1) 记录文件句柄RM\_FileHandle

typedef struct{

bool bOpen; //句柄是否打开（是否正在被使用）

char\* fileName; //该句柄关联的文件名

PF\_FileHandle\* pfileHandle; //关联的页面文件句柄

Frame \*pHdrFrame;//该记录文件的头帧（记录文件的控制页的）的指针

Page\* pHdrPage; //该记录文件的头页（记录文件的控制页）的指针

char\* pBitmap; //记录文件的控制页的位图的指针

RM\_FileSubHeader\* fileSubHeader;//记录文件的总体数量/结构信息

}RM\_FileHandle; //文件句柄

与记录相关的数据结构有很多，具体结构如下：

1. 记录ID

typedef int SlotNum;

typedef struct {

PageNum pageNum; //记录所在页的页号

SlotNum slotNum; //记录的插槽号

bool bValid; //true表示为一个有效记录的标识符

}RID; //记录ID

1. 记录RM\_Record

typedef struct{

bool bValid; // False表示还未被读入记录

RID rid; // 记录的标识符

char \*pData; //记录所存储的数据

}RM\_Record;

1. 记录扫描RM\_FileScan

typedef struct{

bool bOpen; //扫描是否打开

RM\_FileHandle \*pRMFileHandle; //扫描的记录文件句柄

int conNum; //扫描涉及的条件数量

Con \*conditions; //扫描涉及的条件数组指针

PF\_PageHandle PageHandle; //处理中的页面句柄

PageNum pn; //扫描即将处理的页面号

SlotNum sn; //扫描即将处理的插槽号

}RM\_FileScan;

1. 扫描条件Con

typedef struct

{

int bLhsIsAttr,bRhsIsAttr; //左、右是属性（1）还是值（0）

AttrType attrType; //该条件中的数据的类型

int LattrLength,RattrLength; //若是属性的话，表示属性的长度

int LattrOffset,RattrOffset; //若是属性的话，表示属性的偏移量

CompOp compOp; //比较操作符

void \*Lvalue,\*Rvalue; //若是值的话，指向对应的值

}Con; //条件结构体，保存单个谓词条件

其中attrType和LattrLength/RattrLength分别表示所要进行比较的属性的类型和长度。当属性的类型为整型或浮点型时，LattrLength/RattrLength固定为4。当属性的类型为字符串类型时，LattrLength/RattrLength表示的是字符串的长度。LattrOffset/RattrOffset表示所指定的属性在记录中的偏移量，通过它就能给在记录中找到所要比较的属性值。

1. 比较操作符CompOp

typedef enum {

EQual, LessT, GreatT, // 等于，小于，大于

NEqual, LEqual, GEqual, // 不等于，小于等于，大于等于

NO\_OP // 无比较

}CompOp;

### 2.3.2 记录文件的接口实现

（1）记录文件管理函数

1) RC RM\_CreateFile(char \*filename, int recordSize);

创建一个名为fileName的记录文件，该文件中每条记录的大小为recordSize。主要在创建页面文件的基础上申请第1页的空间，并且填写记录信息控制页的控制信息。主要流程如图 2.2所示。



图 2.2 RM\_OpenFile函数流程图

2) RC RM\_OpenFile (char \*fileName, RM\_FileHandle \*fileHandle);

根据文件名打开指定的记录文件，返回其文件句柄指针。本函数填充文件句柄中的值，返回的文件句柄可以持续操作该文件，流程如图 2.3所示。



图 2.3 RM\_OpenFile函数流程图

3) RC RM\_CloseFile (RM\_FileHandle \*fileHandle);

关闭给定句柄对应的记录文件，并且释放对应文件的资源。主要调用了页面文件的CloseFile函数关闭文件，会清除该文件的页面在缓冲区中的占用，同时将该句柄置为关闭。

（2）记录操作函数

1) RC GetRec (RM\_FileHandle \*fileHandle, RID \*rid, RM\_Record \*rec);

获取指定文件中标识符为rid的记录内容到rec指向的记录结构中。函数的功能如图 2.4所示。



图 2.4 GetRec函数流程图

2) RC InsertRec (RM\_FileHandle \*fileHandle, char \*pData, RID \*rid);

插入一个新的记录到指定文件中，pData为指向新纪录内容的指针，返回该记录的标识符rid。找到空槽位插入后，需要将该页面该槽位置为1，记录总数加1，还要判断该页面是否已满，若页面变满页，需要修改控制页的位图。总体的功能如图 2.5所示。

3) RC DeleteRec (RM\_FileHandle \*fileHandle, RID \*rid);

从指定文件中删除标识符为rid的记录，rid中包含该记录的页面号和记录槽位号。需要将该槽位置为空，总记录数减1。判断页面是否变成不满，变为不满需要修改控制页的位图。函数功能如图 2.6所示。



图 2.5 InsertRec函数流程图

4) RC UpdateRec (RM\_FileHandle \*fileHandle, RM\_Record \*rec);

更新指定文件中的记录，rec指向的记录结构中的rid字段为要更新的记录的标识符，pData字段指向新的记录内容。先检查句柄是否已经打开和rid的合法性，然后只需要找到rid中只是的记录的页面的槽位，复制新的记录即可。函数功能如图 2.7所示。



图 2.6 DeleteRec函数流程图



图 2.7 UpdateRec函数流程图

（3）文件扫描函数

1) RC OpenScan(RM\_FileScan \*rmFileScan, RM\_FileHandle \*fileHandle, int conNum, Con \*conditions);

打开一个文件扫描。本函数利用从第二个参数开始的所有输入参数初始化一个由参数rmFileScan指向的文件扫描结构，在使用中，用户应先调用此函数初始化文件扫描结构，然后再调用GetNextRec函数来逐个返回文件中满足条件的记录。如果条件数量conNum为0，则意味着检索文件中的所有记录。如果条件不为空，则要对每条记录进行条件比较，只有满足所有条件的记录才被返回。

该函数主要对rmFileScan结构体进行了赋值，该结构体存储了处理的记录文件句柄，要求查询扫描的条件个数以及条件的指针。更重要的是存储接下来要扫描到（即将要处理）的记录的位置，通过pn页面号和sn该页面的槽位号进行标志，知道处理完该文件的所有记录，此次扫描才结束。初次调用时，将pn设为2，sn设为-1，通过辅助函数nextRec来将该位置在逻辑上加1。

2) RC GetNextRec (RM\_FileScan \*rmFileScan, RM\_Record \*rec);

获取一个符合扫描条件的记录。如果该方法成功，返回值rec应包含记录副本及记录标识符。如果没有发现满足所有扫描条件的记录，则返回RM\_EOF。为完成该函数定义了2个辅助函数，定义如下。通过辅助函数的运用后，GetNextRec实现起来很容易，具体实现如图 2.8所示。

* RC nextRec(RM\_FileScan \*rmFileScan);

该函数在GetNextRec函数中使用，用来将rmFileScan中的pn和sn指示的

槽位在逻辑上加1，即指向下一个有效的记录的槽位置。实现起来较为容易，从pn页sn槽位开始遍历，若是遇到一个有效的记录，就将pn和sn置为该记录的位置，若是没有则返回RM\_EOF表示没有有效的记录了，并且将pn和sn置为-1表示不存在记录了。

* bool selected(Con \*pCon, char \*pData);

该函数测试该条记录pData是否符合pCon表示的条件，如果符合则返回true，

不符合返回false。函数的实现较为简单，通过pCon中存储的类型和比较运算符，然后通过左值和右值是属性则在pData记录中取出否则在pCon中取出，如果左值和右值符合比较运算符返回true，否则返回false。

3) RC CloseScan(RM\_FileScan \*rmFileScan);

关闭一个文件扫描，释放相应的资源。主要将bOpen置为false，表示该扫描已经关闭。



图 2.8 GetNextRec函数流程图

## 实验过程

本模块设计是遵循指导书的建议进行设计，思路比较清晰，在对记录扫描编码时觉得不好处理，于是通过辅助函数分步骤解决了获取下一条有效记录的问题。

在与系统模块进行联调的过程中发现了问题，发现插入记录，修改记录无效，最后调试发现是没有标记脏页造成的。其中页不乏有编码的错误，有些编码没有考虑清除造成了逻辑上的错误。通过设计阶段理清思路后很容易编写正确的代码。

最后通过与系统模块的联调，对数据表进行反复的插入，修改，删除记录后测试结果都是正确的，程序也没有出现其它的异常。

# 查询处理模块

## 模块概述

查询处理模块负责select语句的处理。HustBase只支持简单的select语句，支持语法详见指导书第9.1节。根据关系模型理论，最基本的查询处理方法为先计算关系的笛卡尔集，然后选择并投影出最终的结果集。但是在本模块的设计实现中，应尽可能考虑利用代数优化方法及索引对查询过程进行优化。

## 模块设计

本模块的设计主要是利用代数优化的方法进行查询优化，先对查询的表的记录进行单表选择，然后进行多表连接。设计主要从单表和多表查询进行考虑和设计，单表查询较为简单与普通的查询基本没什么差别，只是要将查询的结果的相关字段插入到结果集中，多表需要进行表记录的连接后将字段插入到结果集中。

## 模块实现

### 3.3.1 主要数据结构

1) 表属性RelAttr

struct RelAttr {

char \*relName; // 表名

char \*attrName; // 属性名

};

2) 条件Condition

typedef struct \_Condition Condition;

struct \_Condition{

int bLhsIsAttr; //操作符左边是属性为1，是值时为0

Value lhsValue; //当左边是值时，保存该值的信息

RelAttr lhsAttr; //当左边是属性时，保存属性信息

CompOp op; //比较运算符

int bRhsIsAttr; //操作符右边是属性为1，是值时为0

RelAttr rhsAttr; //当右边是属性时，保存属性信息

Value rhsValue; //当左边是值时，保存该值的信息

};

3) 查询结果SelResult

typedef struct SelResult{

int col\_num; //结果集的字段数，最多20个字段

int row\_num; //结果集的记录数

AttrType type[20]; //结果集各字段的数据类型

int length[20]; //结果集各字段值的长度

char fields[20][20]; //结果集各字段的字段名，字段名不超过20个字符

char \*\*res[100]; //结果集数据，每个链表节点最多记录100条记录

SelResult \*next\_res; //指向下一个节点的指针

}SelResult;

### 3.3.2 查询处理的接口实现

1) RC Select (int nSelAttrs, RelAttr \*selAttrs, int nRelations, char \*\* relations, int nConditions, Condition \*conditions, SelResult \* res);

前六个参数逻辑上可分为三组。每一组的第一个参数是一个整型n，表示本组中第二个参数的条目数量；第二个参数是一个包含实际条目数的为n的数组。例如，参数nSelAttrs包含所选属性的数量，而参数selAttrs是包含实际属性的长度为nSelAttrs的数组，其余类似。

* bool valueCompare(Value \*v1, Value \*v2, ComOp op);

比较同一种类型的Value是否符合运算符op，如果满足op运算符返回true，

否则返回false。

* int isInTableList(int nRelations, char \*\*relations, char \*relName);

测试relName是否在nRelations个表名的数组中，由于relations表示时查询

的表名的数组是逆序的，返回该表名位于查询表的第几个，不在其中返回-1。

* RC checkFields(int nSelAttrs, RelAttr \*\*selAttrs, int nRelations, char \*\*relations, int nConditions, Condition\* conditions, bool& isSelAll, int& col\_num, AttrType\* type, int\* offset, int\* length, char fields[][20]);

检查select语句中查询字段和条件中涉及的字段的合法性和有效性，若全部

合法，则用isSelAll返回是否为查询全部字段(\*)，并将查询的字段的信息存储在col\_num , type, offset, length, fields变量中。同时若是字段合法则将没有表名的字段附加上唯一的表名信息。处理过程如图 3.1所示。

* Con\* conditonToCon(int nConditions, Condition\* conditions);

将n个查询语句分析的Condition转化为n个Con结构的数组。

* bool insertSelRes(SelResult\* res, char\*\* pData);

向res结果集中插入由res->col\_num个记录中的部分字段组成的一条结果。

* RC multiTablesSelect(int nSelAttrs, RelAttr\*\* selAttrs, bool isSelAll, int nRelations, char\*\* relations, RM\_FileHandle\* nRelFileHandle, int \*consNum, Con\*\* cons, int nConditons, Condition\* conditions, char\*\* pData, int curRelation, SelResult\* res);

多表查询函数，递归调用，每次调用就表示连接一张表，等连接完nRelations

张表后表示记录符合要求，向结果集中插入此次连接表示的一条记录，直到扫描完所有的记录为止。

为了更好的实现Select查询，定义了上述6个辅助函数。通过上述函数Select

整体流程如图 3.2所示。

2) RC query(char \*sql, SelResult \*res);

执行一条SELECT语句，查询结果集通过res参数返回。此函数是提供给测试程序专用的接口，本身只是调用Select函数返回结果集。



图 3.1 checkFields函数流程图



图 3.2 Select函数流程图

## 实验过程

查询模块是整体上较为简单的，只是考虑单表和多表查询，单表查询用一次

遍历或者索引就可以完成。多表查询要考虑复杂的查询条件和连接变得有点复杂。

将单表查询和多变查询分离，在多表查询中将条件分离使得查询变得更加容易，不会出错。不过处理的过程中对res结果集的offset和res结果记录的表意不明确，开始是将一条结果的字段分开存放，也可以很好的进行结果记录的存取。

在与系统模块进行联调中，开始不能正确显示，因为最后要以字符串的形式打印，最后必须要进行转换。然而查询的结果的字段是逆序的，因为参数中的字段和表名等都是逆序的，在进行处理的时候就直接处理了。通过处理为正确的顺序后显示正确。与系统模块进行多次的查询测试，单表多表测试都没有出现其它错误。

# 实验总结

## 4.1 工作总结

主要工作：

（1）从记录管理模块开始编写，分析文件的底层的存储结构，实现了在页面文件基础上的记录文件模块。

（2）与系统模块联调，测试了记录模块的正确性。

（3）实现了单表和多表查询，完成了查询模块的编写。

（4）与系统模块联调，测试了查询模块的正确性。

（5）修改系统的版本，调试本系统通过了网站上的测试。

实验心得：

这次课设在HustBase这个框架下完成了单用户的简单的关系数据库。我主要完成了记录模块和查询模块部分，在与其它模块的联调过程中，也学习到了索引部分B+树的构造和索引记录的处理。虽然此次课设是以数据库为背景，但是也学习到了其它学科的基础知识，以及提高了基本编程能力。在页面到记录管理部分对页面缓存和利用下层抽象为上层提供记录管理的处理有了更加深刻的认识，对文件处理认识加深。数据库部分，理解了关系型数据库底层处理和检索数据的基本原理，对关系型数据库有了进一步的理解。

这次课设是对数据库等一些理论知识进行了一次实践，不仅学习到了编程层面的知识也加深了认识。在合作中分模块和联调也收益蛮多，总之本次实验是一次比较成功和收获很多。

## 4.2 改进方案

整体系统的功能是满足基本要求的，主要的改进就是代码的组织逻辑和查询模块的进一步优化。

代码中的接口函数有的并没有使用，成为冗余。但是有的代码例如转换Contitions条件会在多个模块中使用到，可以作为函数重复使用等等。

虽然本系统的索引模块编写完成，做了简单的测试，但是没有在查询模块中使用，一是实践不够充足二是需要修改已有的代码进行测试。整个系统能完成基本的SQL语句，但是复杂的不能完成，但是对了解和实践关系型数据库帮助很大。

# 附录

系统问题：

1. 查询模块中的结果集定义和测试不符合

结果集中的res是二级指针，但是测试程序是将一条记录视作整条结果记录，而不是将字段分开存储，与二级指针和销毁结果集代码冲突。