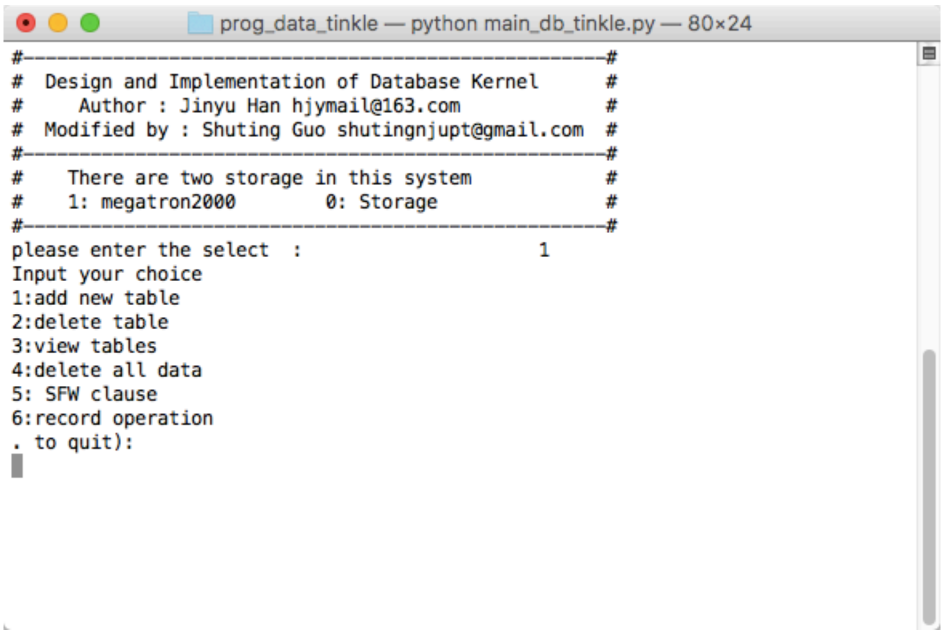
数据库系统内核的设计与实现

Design and Implementation of Database Kernel



```
#-----------------------------------------------------#
#  Design and Implementation of Database Kernel       #
#     Author : Jinyu Han hjymail@163.com              #
#  Modified by : Shuting Guo shutingnjupt@gmail.com  #
#-----------------------------------------------------#
#    There are two storage in this system             #
#    1: megatron2000        0: Storage                #
#-----------------------------------------------------#
please enter the select  :                       1
Input your choice
1:add new table
2:delete table
3:view tables
4:delete all data
5: SFW clause
6:record operation
. to quit):
```

[班级]:	10160411
[学号]:	1016041128
[姓名]:	郭舒婷
[专业]:	计算机应用技术
[指导教师]:	韩京宇

2017 年 1 月

目录

数据库系统内核的设计与实现

Design and Implementation of Database Kernel

第一部分

1.1 MegaStorage类

1.1.1 MegaStorage描述

mega_storage.py is to store table data in separate files.

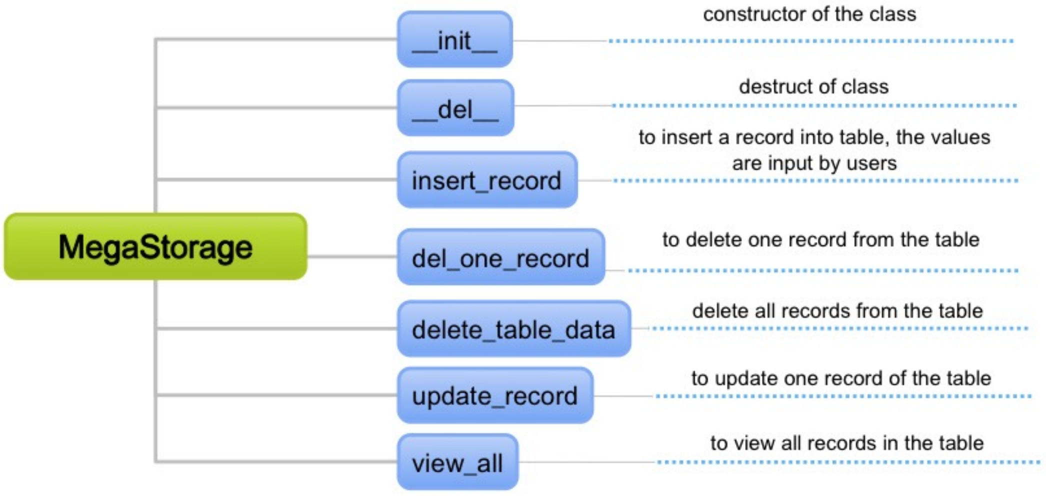
Each table is stored in a separate file with the suffix “.dat".

For example, table named moviestar is stored in file moviestar.dat.

As it is to demonstrate principles in chapter one, it is rather simple.

The file is in ASCII text format, not binary one.

Each line corresponds to one record and different field values are separated by |


MegaStorage
__init__
constructor of the class
__del__
destruct of class
insert_record
to insert a record into table, the values are input by users
del_one_record
to delete one record from the table
delete_table_data
delete all records from the table
update_record
to update one record of the table
view_all
to view all records in the table

MegaStorage类中函数及其功能

1.1.2 MegaStorage的主要参数和函数实现

MegaStorage的两个参数：

f_handle: 文件指针

record_list: .存储dat文件中的内容

（1）MegaStorage的初始化（__init__函数）

（2）MegaStorage的析构（__del__函数）

（3）插入数据（insert_record函数）

（4）删除一条数据（del_one_record函数）

传入参数：

要删除的field_name和对应的值value（用户输入），field_name_list（Schema提供）

返回参数：无

主要操作：

step1:找到field_name在field_name_list的位置deleteIndex

step2:在record_list的每条记录record上查看record[deleteIndex]==value是否成立，如果成立，则在record_list上删除该条记录

step3:将f_handle的offset设为0，清空文件，再写入record_list

step4:刷新缓冲区，将缓冲区中的数据立刻写入文件，同时清空缓冲区

```
# --------------------------------
# to delete one record from the table
# input
#  value_list: the list of field values of which each element is a tuple (field_name,new_field_value)
# -------------------------------
def del_one_record(self, value_list,field_name_list):
        if value_list[0] not in field_name_list:
                print 'No Field!'
                return
        updateIndex = field_name_list.index(value_list[0])
        tmp_List=[]
        for record in self.record_list:
                if record.split('|')[updateIndex] != value_list[1]:
                        tmp_List.append(record)
        self.record_list=tmp_List[:]

        self.f_handle.seek(0)       #back to the head of file,offset=0
        self.f_handle.truncate(0)    #cut all data after offset=0
        # print self.f_handle.tell()
        self.f_handle.write('\n'.join(self.record_list))  #write record_list
        self.f_handle.write('\n')
        self.f_handle.flush()
```

（5）删除全部数据（delete_table_data函数）

传入参数：无

传出参数：无

主要操作：

step1:将f_handle的offset设为0，清空文件

step2:刷新缓冲区，将缓冲区中的数据立刻写入文件，同时清空缓冲区

```
# ------------------------------
# delete all records from the table
# ---------------------------------
def delete_table_data(self):
      self.f_handle.truncate(0)
      self.f_handle.seek(0)
      self.f_handle.flush()
```

（6）更新记录（update_record函数）

传入参数：

要修改的field_name和原始值value，以及修改后的值value1（用户输入），field_name_list（Schema提供）

```
            # --------------------------------
            # to update one record of the table
            # input
            #       condition_list: the where conditon, of which each element is a tuple (field_name, 
field_value)
            #       new_value_list: new value list, of which each element is a tuple 
(field_name,new_field_value)
            # -------------------------------

            def update_record(self, condition_list, new_value_list,field_name_list):
                        if condition_list[0] not in field_name_list:
                                    print 'No Field!'
                                    return
                        updateIndex = field_name_list.index(condition_list[0])
                        for idx in range(len(self.record_list)):
                                    tmp = self.record_list[idx].split('|')
                                    if tmp[updateIndex] == condition_list[1]:
                                                tmp[updateIndex] = new_value_list[1]
                                                self.record_list[idx] = '|'.join(tmp)

                        self.f_handle.truncate(0)
                        #print self.f_handle.tell()
                        self.f_handle.seek(0)
                        self.f_handle.write('\n'.join(self.record_list))
                        self.f_handle.write('\n')
```

传出参数：

无

主要操作：

step1:找到field_name在field_name_list的位置updateIndex

step2:在record_list的每条记录record上查看record[updateIndex]==value是否成立，如果成立，则将record[updateIndex]修改为value1

step3:将f_handle的offset设为0，清空文件，再写入record_list

step4:刷新缓冲区，将缓冲区中的数据立刻写入文件，同时清空缓冲区

（7）获取所有数据（view_all函数）

传入参数：无

传出参数：无

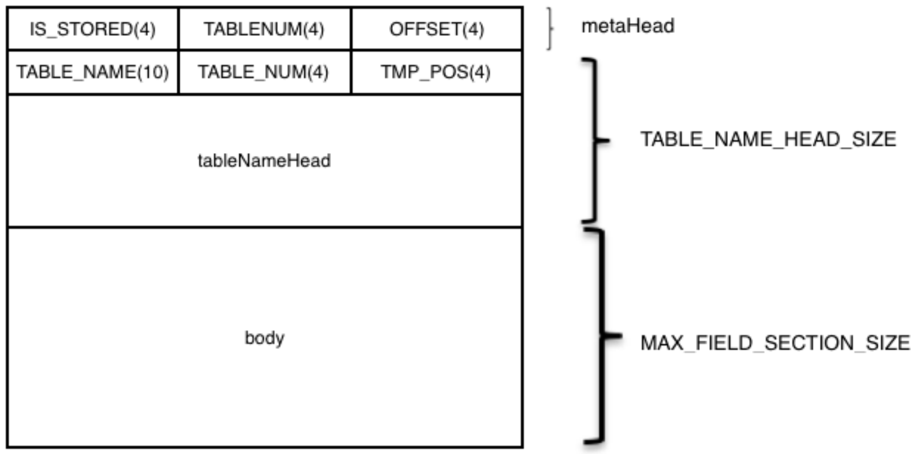
主要操作：将record_list的记录打印出来即可

备注：为了系统集成，我改成了获取全部数据

```
# --------------------------------
# to view all records in the table
# input
# -------------------------------
def get_record_list(self):
        tmp=[]
        if len(self.record_list) > 0:
                return list(map(lambda x:x.split('|'),self.record_list))
        else:
                return tmp
```

1.2 Schema类

1.2.1 Schema描述（all.sch文件描述）



all.sch 存储了本数据库第一部分的schema data，all.sch文件可以分为三个部分metaHead, tableNameHead 和 body.metaHead存储了IS_STORED(是否有数据存取，bool类型，占4 byte)，TABLENUM（表个数，int类型，占4 byte），OFFSET（body偏移量位置，int类型，占4 byte）。tableNameHead存储了表的信息，包括TABLE_NAME（表名，char类型，占10 byte），TABLE_NUM（表中FIELD个数，int类型，占4byte）和TMP_POS（该表第一个FIELD所在位置）。

1.2.2 schema_db的主要参数和函数实现

（1）Schema的初始化（__init__函数）

（2）Schema的析构（__del__函数）

（3）插入数据（appendTable函数）

（4）删除一条表数据（delete_table_schema函数）

```
        # ---------------------------------------------
        # to delete the schema of a table from the schema file
        # input
        #       table_name: the table to be deleted
        # output
        #       True or False
        # -----------------------------------------------
        def delete_table_schema(self, table_name):
                tmpIndex=-1
                for i in range(len(self.headObj.tableNames)):
                        if self.headObj.tableNames[i][0]==table_name:
                                tmpIndex=i
                if tmpIndex>=0:
                        self.headObj.tableNames.remove(self.headObj.tableNames[tmpIndex])
                        self.headObj.tableFields.remove(self.headObj.tableFields[tmpIndex])
                        self.headObj.lenOfTableNum-=1
                        if len(self.headObj.tableNames):
                                name_list = map(lambda x: x[0], self.headObj.tableNames)
                                table_num = map(lambda x: x[1], self.headObj.tableNames)
                                table_offset= map(lambda x: x[2], self.headObj.tableNames)
                                table_offset[0] = BODY_BEGIN_INDEX
                                for idx in range(1,len(table_offset)):
                                        table_offset[idx] = table_offset[idx-1] + table_num[idx-1]*10
                                self.headObj.tableNames=zip(name_list,table_num,table_offset)
                                self.headObj.offsetOfBody=self.headObj.tableNames[-1]
   [2]+self.headObj.tableNames[-1][1]*10
                                self.WriteBuff()
                        else:
                                self.headObj.offsetOfBody = BODY_BEGIN_INDEX
                                self.headObj.isStored = False
                        return True
                else:
                        print 'Cannot find the table!'
                        return False
```

传入参数：表名table_name

传出参数：删除成功或者失败（TRUE / FALSE）

主要操作：

step1:在tableNames中寻找是否有名为table_name的table，如果有，返回所在位置Index

step2:在tableNames和tableFields中删除下标为Index的数据

step3: 修改每个table的tmp_pos，修改offsetofBody

step4:将数据重新写入all.sch中

（5）获取表名(get_table_name_list函数)

```
# ---------------------------
# to return the list of all the table names
# input
# output
#       table_name_list: the returned list of table names
# -------------------------------
def get_table_name_list(self):
                return map(lambda x:x[0],self.headObj.tableNames)
```

（6）读取表的结构(viewTableStructure函数)

传入参数：table_name

传出参数：table_name的field结构

```
# -------------------------------
# to view the table schema given table name
# input
#   table_name:
# output
#   field list of the table
# ----------------------------------------
def viewTableStructure(self, table_name):
        #print 'viewTableStructure begins to execute'
        tmp=[]
        for i in range(len(self.headObj.tableNames)):
                if self.headObj.tableNames[i][0] == table_name:
                        tmp = [j.strip() for j in self.headObj.tableFields[i]]
```

（7）删除所有表（deleteAll函数）

```
# --------------------------
# delete all the contents in the schema file
# ----------------------------------------
def deleteAll(self):
        self.fileObj.seek(0)
        self.fileObj.truncate(0)
        self.headObj.isStored = False
        self.headObj.lenOfTableNum = 0
        self.headObj.offsetOfBody = self.body_begin_index
        self.fileObj.flush()
        print "all.sch file has been truncated"
```

（8）查找表（find_table函数，判断table_name是否在all.sch文件中）

```
# -------------------------------
# to determine whether the table named table_name exist
# input
#       table_name
# output
#       true or false
# -----------------------------------------------------
def find_table(self, table_name):
            Tables = map(lambda x: x[0], self.headObj.tableNames)
            if table_name in Tables:
                        return True
```

1.3 SQL语句解析

SQL SELECT 语句用于从表中选取数据，结果被存储在一个结果表中（称为结果集），其语法为：

```
SELECT 列名称 FROM 表名称
```

或者

```
SELECT * FROM 表名称
```

通常WHERE语句可以与SELECT搭配使用

```
SELECT 列名称 FROM 表名称 WHERE 列 运算符 值
```

在mega_sfw.py函数中，实现了对形如

select */列名 from 表名 where 列名 = 值

select */列名 from 表名

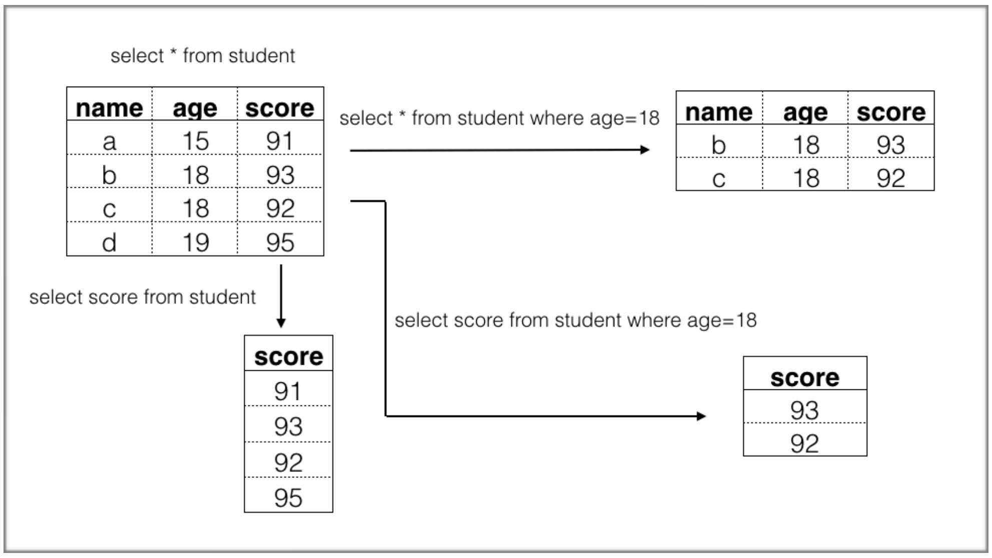
的解析。

主要操作如下：

step1：（parse_sfw函数）将sql语句解析为

select sel_list from from_list where where_list = where_list_condition格式，并返回参数

该函数主要通过正则表达式对sfw语句进行匹配，如果匹配失败，则返回失败信号（ISFlase=True），如果匹配成功，则进入step2 step2：（process_sfw函数）从schema_obj对象中获得from_list中表的结构，并用mega_storage类对from_list.dat进行操作，获取所有的record，根据 where_list 和 where_list_condition对record进行操作，最后将sel_list列的数据输出。



```
#----------------------------------------
# parse the sfw clause into select list, from list and where list
# input
#       sql_str
# output
#       sel_list
#       from_list
#       where_list
#------------------------------------------------------
def parse_sfw(self,sql_str):
    sql_str.strip()
    sel_list='';from_list='';where_list='';where_list_condition=''
    ISFlase=0
    try:
        s = re.match(r'select (.*) from (.*) where (.*)', sql_str).groups()
        sel_list = s[0]
        from_list = s[1]
        where_list = s[2].split('=')[0]
        where_list_condition = s[2].split('=')[1]
    except:
        try:
            s = re.match(r'select (.*) from (.*)', sql_str).groups()
            sel_list = s[0]
            from_list = s[1]
            where_list = ''
            where_list_condition = ''
        except:
            ISFlase=1
    return sel_list,from_list,where_list,where_list_condition,ISFlase
```

```
#---------------------------------
# to get the "select from where" result
# input
#   sql_str: select from where clause
# note: we needs to create a tempory mega_storage object to get data from table
#---------------------------------------
def process_sfw(self,sql_str):
    sql_str.strip()
    sel_list, from_list, where_list, where_list_condition, isFalse=self.parse_sfw(sql_str.strip())
    if isFalse:
        print 'WRONG SQL QUERY!'
        return

    #print self.schema_ptr.get_table_name_list()

    if from_list not in self.schema_ptr.get_table_name_list():
        print 'Cannot Find Table '+from_list+'!\n'
        return False
    else:
        dataObj = mega_storage.MegaStorage(from_list)
        Data_List=dataObj.get_record_list()
        Field_List=self.schema_ptr.viewTableStructure(from_list)
        if where_list=='':
            if sel_list=='*':
                print '|'.join(Field_List)
                for record in Data_List:
                    print '|'.join(record)
            else:
                if sel_list not in Field_List:
                    print 'Cannot Find Field '+sel_list
                    return False
                else:
                    Field_Index = Field_List.index(sel_list)
                    Output= list(map(lambda x:x[Field_Index],Data_List))
                    print '|'+sel_list+'|'
                    for record in Output:
                        print '|'+record+'|'
        else:
            if where_list not in Field_List:
                print 'Cannot Find Field ' + where_list
                return False
            else:
                Field_Condition_Index = Field_List.index(where_list)
                if sel_list=='*':
                    for record in Data_List:
                        if record[Field_Condition_Index] == where_list_condition:
                            print '|'+'|'.join(record)+'|'
                else:
                    if sel_list not in Field_List:
                        print 'Cannot Find Field ' + sel_list
                        return False
                    else:
                        Field_Index = Field_List.index(sel_list)
                        #Output = list(map(lambda x: x[Field_Index], Data_List))
                        print '|' + sel_list + '|'
                        for record in Data_List:
                            if record[Field_Condition_Index] == where_list_condition:
                                print '|' + record[Field_Index] + '|'
```

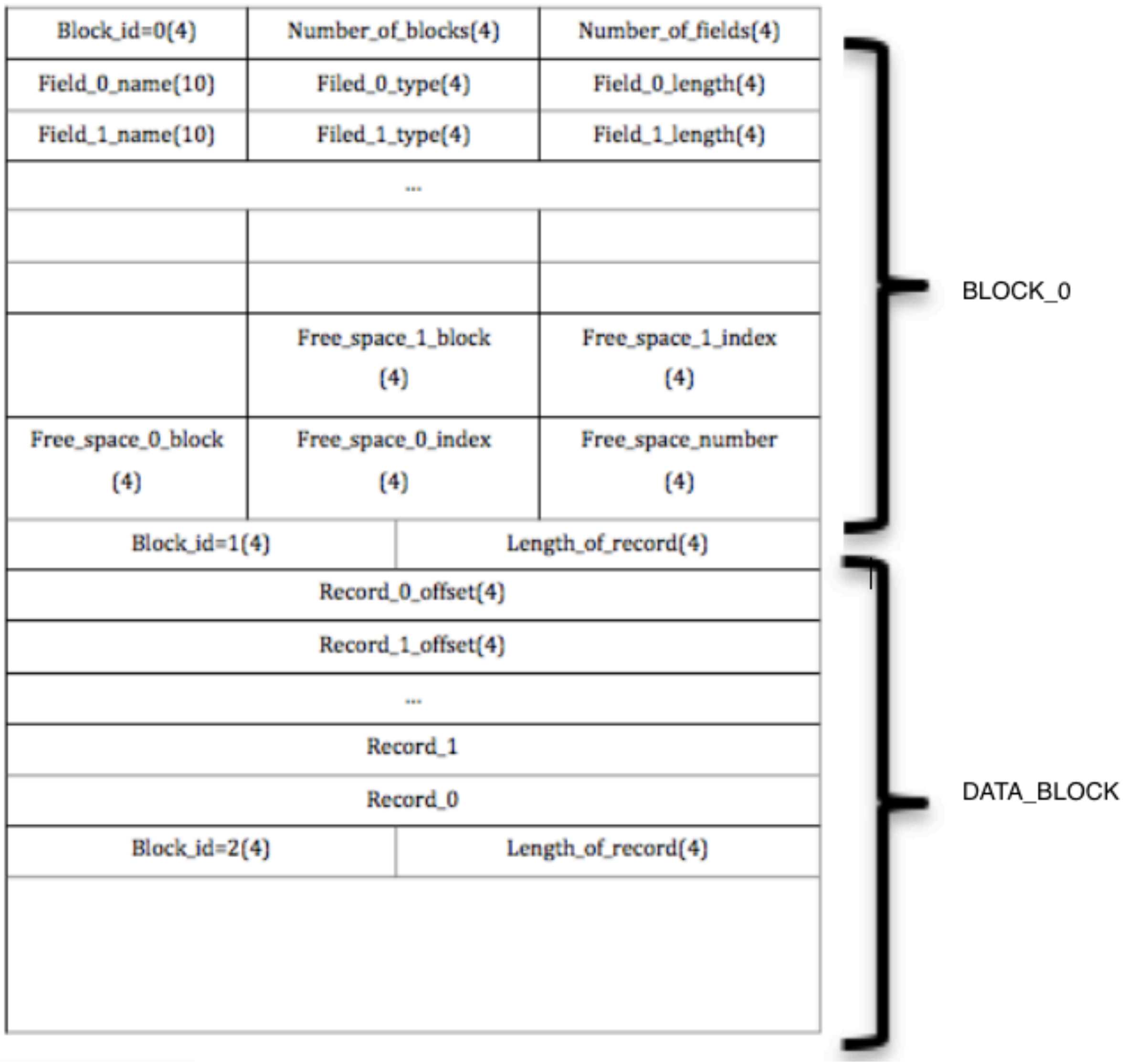
第二部分

2.1 Storage类

2.1.1 Storage描述

与第一部分的mega_storage类不同的是，第二部分的storage类对tablename.dat的操作是二进制的。在tablename.dat中，数据是分块存储的。block_0部分中存储了表field个数以及每个field的属性（name,type,length），而数据的存取是根据类型来存储的，不再是第一部分纯str形式的存储方式。tablename..dat中的结构如下图所示。

因这样的存储方式，我对init函数进行了一番修改。



```
#------------------------------
#constructor of the class
# input:
#       tablename
#-------------------------------------
def __init__(self,tablename): #there must be a self argument for python even if
    #print "__init__ of ",Storage.__name__,"begins to execute"
    tablename.strip()

    self.Free_Space_Number = 0
    self.Free_Space = []
    self.record_list = []
    self.record_Position =[]
    if  not os.path.exists('database/'+tablename+'.dat'):# the file corresponding to the table does
not exist
        print 'table file '+tablename+'.dat does not exists'
        self.f_handle=open('database/'+tablename+'.dat','wb+')
        self.f_handle.close()
        print tablename+'.dat has been created'

    self.f_handle = open('database/'+tablename + '.dat', 'rb+')
    print 'table file ' + tablename + '.dat has been opened'
    self.open = True

    self.dir_buf = ctypes.create_string_buffer(BLOCK_SIZE)
    self.f_handle.seek(0)
    self.dir_buf = self.f_handle.read(BLOCK_SIZE)

    self.dir_buf.strip()
    my_len = len(self.dir_buf)
    self.field_name_list = []
    beginIndex = 0

    if my_len == 0:  # there is no data in the block 0, we should write meta data into the block
0
        self.num_of_fields = raw_input("please input the number of feilds in table " + tablename
+ ":")
        if int(self.num_of_fields) > 0:

            self.dir_buf = ctypes.create_string_buffer(BLOCK_SIZE)
            self.block_id = 0
            self.data_block_num =0
            struct.pack_into('!iii', self.dir_buf, beginIndex, 0, 0,
                             int(self.num_of_fields))  #
block_id,number_of_data_blocks,number_of_fields

            beginIndex = beginIndex + struct.calcsize('!iii')

            # the following is to write the field name,field type and field length into the buffer in
turn
            for i in range(int(self.num_of_fields)):
                field_name = raw_input("please input the name of field " + str(i) + " :")
                if len(field_name) < 10:
                    field_name = ' ' * (10 - len(field_name.strip())) + field_name
                flag=1
                while(flag):
                    field_type = raw_input(
                        "please input the type of field(0, str; 1, varstr; 2, int; 3, boolean) " + str(i) +
" :")
                    if int(field_type) in [0,1,2,3]:
                        flag=0

                field_length = raw_input("please input the length of field " + str(i) + " :")
                temp_tuple = (field_name, int(field_type), int(field_length))
                self.field_name_list.append(temp_tuple)
                struct.pack_into('!10sii', self.dir_buf, beginIndex, field_name, int(field_type),
                                 int(field_length))
                beginIndex = beginIndex + struct.calcsize('!10sii')
                struct.pack_into('!i',self.dir_buf,BLOCK_SIZE-struct.calcsize('!i'),int(0))
            self.f_handle.seek(0)
            self.f_handle.write(self.dir_buf)
            self.f_handle.flush()
else:  # there is something in the file
    self.block_id, self.data_block_num, self.num_of_fields = struct.unpack_from('!iii',
self.dir_buf, 0)
    self.Free_Space_Number = struct.unpack_from('!i',self.dir_buf,BLOCK_SIZE-
struct.calcsize('!i'))[0]
    if self.Free_Space_Number>0:
        for i in range(self.Free_Space_Number):
            self.Free_Space.append(struct.unpack_from('!ii',self.dir_buf,BLOCK_SIZE-
                                              struct.calcsize('!i')-struct.calcsize('!ii')*
                                              (i+1)))

    beginIndex = struct.calcsize('!iii')
    # the followins is to read field name, field type and field length into main memory structures
    for i in range(self.num_of_fields):
        field_name, field_type, field_length = struct.unpack_from('!10sii', self.dir_buf,
                                                     beginIndex + i * struct.calcsize(
                                                         '!10sii'))  # i means no memory alignment

        temp_tuple = (field_name, field_type, field_length)
        self.field_name_list.append(temp_tuple)
        #print "the " + str(i) + "th field information is ", temp_tuple

Dict_ = {0: 's', 1: 's', 2: 'i', 3: '?'}
'''0, str;1, varstr;2, int;3, boolean'''
self.EncodeMethod = []
for i in range(int(self.num_of_fields)):  # Output: field information
    if self.field_name_list[i][1] == 0 or self.field_name_list[i][1] == 1:
        self.EncodeMethod.append(str(self.field_name_list[i][2]) + Dict_[self.field_name_list[i]
[1]])
    else:
        self.EncodeMethod.append(Dict_[self.field_name_list[i][1]])
self.EncodeFormat = '!' + ''.join(self.EncodeMethod)
self.record_list = []
self.record_Position = []
Flag = 1
while (Flag <= self.data_block_num):
    self.f_handle.seek(BLOCK_SIZE * Flag)
    self.active_data_buf = self.f_handle.read(BLOCK_SIZE)
    self.block_id, self.Number_of_Records = struct.unpack_from('!ii', self.active_data_buf, 0)
    self.offsetList = []
    if self.Number_of_Records > 0:
        for i in range(self.Number_of_Records):
            if (Flag, i) not in self.Free_Space:
                self.record_Position.append((Flag, i))
                self.offsetList.append(struct.unpack_from('!i', self.active_data_buf,
                                                   struct.calcsize('!ii') + i * struct.calcsize(
                                                       '!i'))[0])
    #print self.offsetList
    for idx in self.offsetList:
        self.record_list.append(
                struct.unpack_from(self.EncodeFormat, self.active_data_buf, idx + struct.calcsize('!
ii10s')))
    Flag += 1
    #----------------------
    # destruct of class
    #------------------------
    def __del__(self):  # write the metahead information in head object to file
        record_content_len = struct.calcsize(self.EncodeFormat)
        record_head_len = struct.calcsize('!ii10s')
        record_len = record_head_len + record_content_len

        self.data_buf = ctypes.create_string_buffer(BLOCK_SIZE*(self.data_block_num+1))

        struct.pack_into('!iii',self.data_buf,0,0,int(self.data_block_num),int(self.num_of_fields))

        for i in range(int(self.num_of_fields)):
            struct.pack_into('!10sii',self.data_buf,struct.calcsize('!iii')+struct.calcsize('!10sii')*i,
                         self.field_name_list[i][0],self.field_name_list[i][1]
                         ,self.field_name_list[i][2])

        Begin_Index=BLOCK_SIZE-struct.calcsize('!i')
        #free_space_number
        struct.pack_into('!i',self.data_buf,Begin_Index,self.Free_Space_Number)
        #free_space
        for i in range(len(self.Free_Space)):
            struct.pack_into('!ii',self.data_buf,Begin_Index-struct.calcsize('!ii')*(i+1),
                         self.Free_Space[i][0],self.Free_Space[i][1])

        Flag=1
        while(Flag<=self.data_block_num):
            count=-1
            for idx,idj in zip(self.record_Position,self.record_list):
                if idx[0]==Flag:
                    count = idx[1]
                    offset = BLOCK_SIZE*Flag+struct.calcsize('!ii')+idx[1]*struct.calcsize('!i')
                    beginindex = BLOCK_SIZE-record_len*(idx[1]+1)
                    #offset
                    struct.pack_into('!i',self.data_buf,offset,beginindex)
                    record_schema_address = struct.calcsize('!iii')  # offset in  the block 0
                    update_time = '2016-11-16'  # update time
                    #message
                    struct.pack_into('!
    ii10s',self.data_buf,BLOCK_SIZE*Flag+beginindex,record_schema_address,record_len,update_time)
                    for i in range(len(idj)):
                        struct.pack_into('!' + self.EncodeMethod[i],
    self.data_buf,BLOCK_SIZE*Flag+beginindex+record_head_len + struct.calcsize(self.EncodeFormat)
                                 - struct.calcsize('!' + ''.join(self.EncodeMethod[i:])), idj[i])
            #block_id,num_of_records
            struct.pack_into('!ii', self.data_buf, BLOCK_SIZE*Flag, Flag,count+1)
            Flag+=1
        self.f_handle.seek(0)
        self.f_handle.write(self.data_buf)
        self.f_handle.flush()
```

在init函数中，我们可以获得如下信息（下划线的是新增的内容）：

数据块的个数：self.data_block_num

field的个数：self.num_of_fields

field的属性列表：self.field_name_list

可用空间的个数：self.Free_Space_Number

可用空间列表：self.Free_Space

field对应的编码形式：self.EncodeMethod

一条记录的编码形式：self.EncodeFormat

文件中的全部record内容：self.record_list

文件中record在每个块中存放的位置：self.record_Position

同时，为了方便起见，修改了del函数，在storage对象被删除时候，再将数据一次性写入.dat文件中（del函数）。

2.1.2 storage的主要参数和函数实现

（1）storage的初始化（__init__函数）

（2）storage的析构（__del__函数）

（3）插入数据（insert_record函数）

传入参数：input_record

传出参数：无

step1：根据self.field_name_list，对input_record进行类型转换，如果转换成功，进step2

step2：确定inpurt_record在.dat文件中的位置

如果当前self.Free_Space_Number==0，则无之前删除的可用空间，那么看self.record_Position[-1]所在的块block_id和位置index，如果index小于MAX_RECORD_NUM，位置为(block_id,index+1)，否则为(block_id+1,0)，更新self.data_block_num

如果当前self.Free_Space_Number!=0，则有可用的空间，那么将inpurt_record赋值为self.Free_Space[-1]，同时更新self.Free_Space_Number和self.Free_Space

step3：将inpurt_record和inpurt_record的位置分别加入self.record_list和self.record_Position

```
# --------------------------------
# to insert a record into table
# -------------------------------
def insert_record(self,input_record):
    insert_record=[]
    for i in range(int(self.num_of_fields)):
        try:
            if self.field_name_list[i][1]==2:
                insert_record.append(int(input_record[i].strip()))
            elif self.field_name_list[i][1]==3:
                insert_record.append(bool(input_record[i].strip()))
            else:
                if len(input_record[i])<self.field_name_list[i][2]:
                   insert_record.append(' ' * (10 - len(input_record[i].strip())) + input_record[i].strip())
         except:
             print 'Wrong Input!"
             return

    record_content_len = struct.calcsize(self.EncodeFormat)
    record_head_len = struct.calcsize('!ii10s')
    record_len = record_head_len + record_content_len

    MAX_RECORD_NUM = (BLOCK_SIZE - struct.calcsize('!i') - struct.calcsize('!ii')) / (
    record_len + struct.calcsize('!i'))

    #print 'Each Block Contains at most %s records' %(MAX_RECORD_NUM)
    self.record_list.append(insert_record)
    if len(self.Free_Space):
        self.record_Position.append(self.Free_Space[-1])
        self.Free_Space.remove(self.Free_Space[-1])
        self.Free_Space_Number=self.Free_Space_Number-1
    else:
        if not len(self.record_Position):
            self.data_block_num+=1
            self.record_Position.append((1,0))
        else:
            last_Position = self.record_Position[-1]
            if last_Position[1] == MAX_RECORD_NUM-1:
                self.record_Position.append((last_Position[0]+1,0))
                self.data_block_num+=1
```

（3）删除数据（delete_table_data函数）

传入参数：record_tuple

传出参数：无

step1：判断选择的field(record_tuple[1])是否在self.field_name_list中，如果在，进行step2

step2：在self.record_list中删除内容record_tuple[0]，同时删除self.record_Position对应的位置，并将位置加入到self.Free_Space

step3：更新self.Free_Space_Number

```
#--------------------------------
# to delete one record from the  table
# input
#
#       record_tuple: the tuple of record to be deleted
# output
#       True or False
#-----------------------------------
def delete_table_data(self,record_tuple):
    try:
        Delete_Index=list(map(lambda x:x[0].strip(),self.field_name_list)).index(record_tuple[1])
    except:
        print 'Wrong Input!'
        return
    tmp_Record_Message=zip(self.record_list,self.record_Position)
    self.record_list=[]
    self.record_Position=[]
    self.Free_Space=[]
    for record in tmp_Record_Message:
        tmp_record = record[0][Delete_Index]
        try:
            tmp_record=tmp_record.split()[0]
        except:
            pass
        if tmp_record != record_tuple[0]:
            self.record_list.append(record[0])
            self.record_Position.append(record[1])
        else:
            self.Free_Space.append(record[1])
    self.Free_Space_Number=len(self.Free_Space)
```

（4）删除全部数据（delete_all_data函数）

```
# --------------------------------
# to delete all the data in the table
# input
#
# output
#       True or False
# ----------------------------------
def delete_all_data(self):
    if self.data_block_num==0:
        return False
    data = ctypes.create_string_buffer((self.data_block_num+1)*BLOCK_SIZE)
    self.f_handle.seek(0)
    self.f_handle.write(data)
    self.data_block_num=0
    self.Free_Space_Number=0
    self.Free_Space=[]
    return True
```

（5）显示全部数据（show_table_data函数）

```
#-------------------------------------------
# to show all the data in the table
#--------------------------------------
def show_table_data(self):
    #self.get_table_data()
    #print len(self.record_list)
    print '|'.join(map(lambda x:x[0].strip(),self.field_name_list))
    for idx in self.record_list:
        tmp=[]
        for j in idx:
            try:
                tmp.append(j.strip('\x00'))
            except:
                tmp.append(str(j))
        print '|'.join(tmp)
```

2.2 SQL语句解析（查询编辑器的生成）

2.2.1 查询处理器的三个步骤

查询处理器必须采取三个步骤：

（1）语法树生成：对使用诸如SQL的某种语言书写的查询进行语法分析，亦即将查询语句转换成按某种有用方式表示查询语句结构的语法树。

（2）逻辑查询计划：把语法分析树转换成关系代数表达式树。

（3）将逻辑查询计划转换成物理查询计划。

2.2.2 查询处理的具体处理过程

设输入sql语句为：

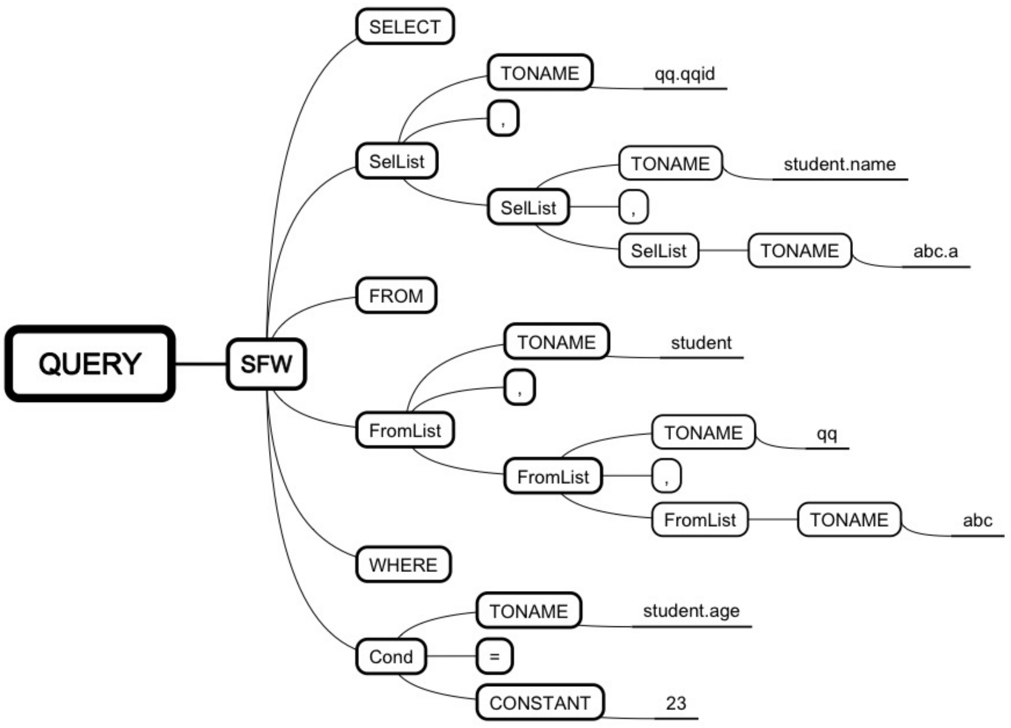
```
select qq.qqid,student.name,abc.a from student,qq,abc where student.age=23
```

step1：将sql 语句解析成语义树，树保存在common_db.global_syn_tree中

其树形如图所示，矩形框为节点类型，横线框为str类型。

涉及文件（lex_db.py, parser_db.py）

为了满足多表查询，对lex_db.py中的**t_TCNAME**函数和**t_CONSTANT**函数做了如下修改，可以支持 student.age 而不是单纯的 age。



```
def t_TCNAME(t):
    r'[A-Z_a-z]\w*(\.){0,1}[A-Z_a-z]\w*'
    #r'[A-Z_a-z]\w*'
    return t

def t_CONSTANT(t):
    #    r'\d+|\'\w+\''
    r'\d+(\.)*\d*|\'\w+\''
    return t
```

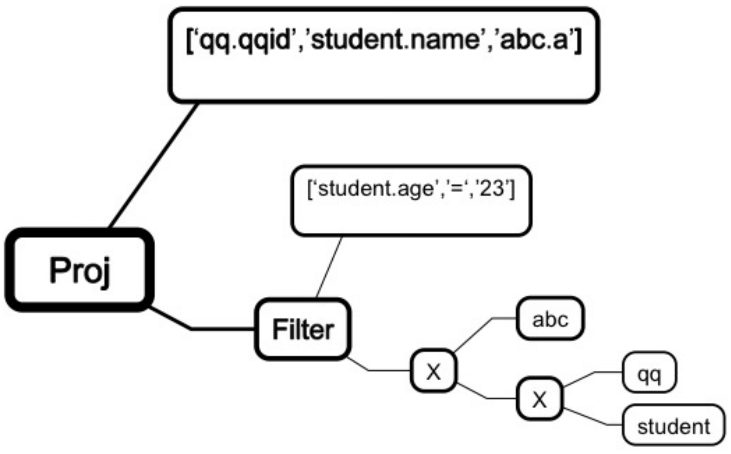
step2：将语义树逆向解析成，sel_list，from_list，where_list即

sel_list=[‘qq.qqid’,’student.name’,’abc.a’]

from_list=[‘student’,’qq’,’abc’]

where_list=[‘student.age’,’=‘,’23’]

step3：根据sel_list，from_list，where_list生成查询树



step4：根据查询树进行解析

1. 表qq和表student做笛卡尔积得到表t1
2. 表t1和表abc做笛卡尔积得到表t2
3. 根据过滤条件['student.age','=','23']对表t2进行数据过滤得到表t3
4. 选择['qq.qqid','student.name','abc.a']得到最终表t4

```
class parseNode:
    def __init__(self):
        self.sel_list=[]
        self.from_list=[]
        self.where_list=[]
    def get_sel_list(self):
        return self.sel_list
    def get_from_list(self):
        return self.from_list
    def get_where_list(self):
        return self.where_list
    def update_sel_list(self,self_list):
        self.sel_list = self_list
    def update_from_list(self, from_list):
        self.from_list = from_list
    def update_where_list(self,where_list):
        self.where_list = where_list
#--------------------------------
# to extract data from gloal variable syn_tree
# output:
#       sel_list
#       from_list
#       where_list
#--------------------------------
def extract_sfw_data():
    print 'extract_sfw_data begins to execute'
    if syn_tree is None:
        print 'wrong'
    else:
        #common_db.show(syn_tree)
        PN = parseNode()
        destruct(syn_tree,PN)
        return PN.get_sel_list(),PN.get_from_list(),PN.get_where_list()

#---------------------------------
# Query  : SFW
#   SFW  : SELECT SelList FROM FromList WHERE Condition
# SelList: TCNAME COMMA SelList
# SelList: TCNAME
#
# FromList:TCNAME COMMA FromList
# FromList:TCNAME
# Condition: TCNAME EQX CONSTANT
#---------------------------------

def destruct(nodeobj,PN):
    if isinstance(nodeobj, common_db.Node):  # it is a Node object
        if nodeobj.children:
            if nodeobj.value == 'SelList':
                tmpList=[]
                show(nodeobj,tmpList)
                PN.update_sel_list(tmpList)
            elif nodeobj.value == 'FromList':
                tmpList = []
                show(nodeobj, tmpList)
                PN.update_from_list(tmpList)
            elif nodeobj.value == 'Cond':
                tmpList = []
                show(nodeobj, tmpList)
                PN.update_where_list(tmpList)
            else:
                for i in range(len(nodeobj.children)):
                    destruct(nodeobj.children[i],PN)
```

```
#----------------------------------
# to execute the query plan and return the result
# input
#       global logical tree
#--------------------------------------------

def execute_logical_tree():
    if common_db.global_logical_tree:
        def excute_tree():
            idx = 0
            dict_ = {}

            def show(node_obj, idx, dict_):
                if isinstance(node_obj, common_db.Node):  # it is a Node object
                    dict_.setdefault(idx, [])
                    dict_[idx].append(node_obj.value)
                    if node_obj.var:
                        dict_[idx][-1] = tuple((dict_[idx][-1], node_obj.var))
                    if node_obj.children:
                        for i in range(len(node_obj.children)):
                            show(node_obj.children[i], idx + 1, dict_)

            show(common_db.global_logical_tree, idx, dict_)
            idx = sorted(dict_.keys(), reverse=True)[0]

            def GetFilterParam(tableName_Order, current_field, param):
                # print tableName_Order,current_field
                if '.' in param:
                    tableName = param.split('.')[0]
                    FieldName = param.split('.')[1]
                    if tableName in tableName_Order:
                        TableIndex = tableName_Order.index(tableName)
                elif len(tableName_Order) == 1:
                    TableIndex = 0
                    FieldName = param
                else:
                    return 0, 0, 0, False
                tmp = list(map(lambda x: x[0].strip(), current_field[TableIndex]))
                if FieldName in tmp:
                    FieldIndex = tmp.index(FieldName)
                    FieldType = current_field[TableIndex][FieldIndex][1]
                    return TableIndex, FieldIndex, FieldType, True
                else:
                    return 0, 0, 0, False

            current_field = []
    while (idx >= 0):
        if idx == sorted(dict_.keys(), reverse=True)[0]:
            if len(dict_[idx]) > 1:
                a_1 = storage_db.Storage(dict_[idx][0])
                a_2 = storage_db.Storage(dict_[idx][1])
                current_list = []
                tableName_Order = [dict_[idx][0], dict_[idx][1]]
                current_field = [a_1.getfilenamelist(), a_2.getfilenamelist()]
                for x in itertools.product(a_1.getRecord(), a_2.getRecord()):
                    current_list.append(list(x))
            else:
                a_1 = storage_db.Storage(dict_[idx][0])
                current_list = a_1.getRecord()
                tableName_Order = [dict_[idx][0]]
                current_field = [a_1.getfilenamelist()]
        elif 'X' in dict_[idx] and len(dict_[idx]) > 1:
            a_2 = storage_db.Storage(dict_[idx][1])
            tableName_Order.append(dict_[idx][1])
            current_field.append(a_2.getfilenamelist())
            tmp_List = current_list[:]
            current_list = []
            for x in itertools.product(tmp_List, a_2.getRecord()):
                current_list.append(list((x[0][0], x[0][1], x[1])))
        elif 'X' not in dict_[idx]:
            if 'Filter' in dict_[idx][0]:
                FilterChoice = dict_[idx][0][1]
                TableIndex, FieldIndex, FieldType, isTrue =
    GetFilterParam(tableName_Order, current_field,
                                                    FilterChoice[0])
                if not isTrue:
                    return [], [], False
                else:
                    if FieldType == 2:
                        FilterParam = int(FilterChoice[2].strip())
                    elif FieldType == 3:
                        FilterParam = bool(FilterChoice[2].strip())
                    else:
                        FilterParam = FilterChoice[2].strip()
                    #print FilterParam
                tmp_List = current_list[:]
                current_list = []
                for tmpRecord in tmp_List:
                    if len(current_field) == 1:
                        ans = tmpRecord[FieldIndex]
                    else:
                        ans = tmpRecord[TableIndex][FieldIndex]
                    if FieldType == 0 or FieldType == 1:
                        ans = ans.strip()
                    if FilterParam == ans:
                        current_list.append(tmpRecord)
```

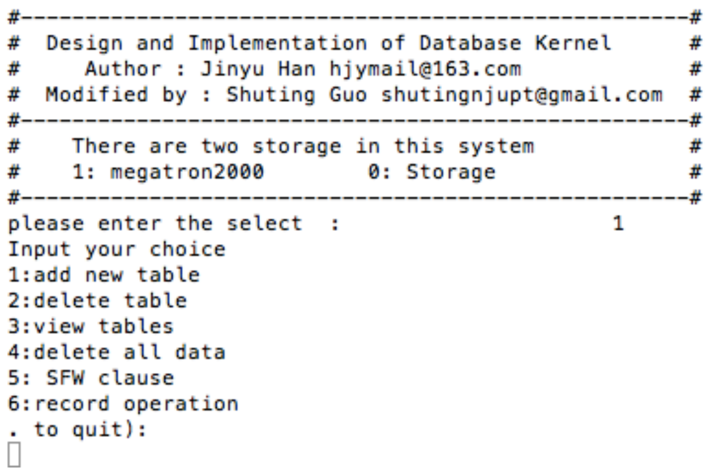
```
              if 'Proj' in dict_[idx][0]:
                SelIndexList = []
                for i in range(len(dict_[idx][0][1])):
                    TableIndex, FieldIndex, FieldType, isTrue = GetFilterParam(tableName_Order,
current_field,
                                                                 dict_[idx][0][1][i])
                    if not isTrue:
                        return [], [], False
                    SelIndexList.append((TableIndex, FieldIndex))
                tmp_List = current_list[:]
                current_list = []
                # print SelIndexList,current_field
                for tmpRecord in tmp_List:
                    # print tmpRecord
                    if len(current_field) == 1:
                        tmp = []
                        for x in list(map(lambda x: x[1], SelIndexList)):
                            tmp.append(tmpRecord[x])
                        current_list.append(tmp)
                    else:
                        tmp = []
                        for x in SelIndexList:
                            tmp.append(tmpRecord[x[0]][x[1]])
                        current_list.append(tmp)
                outPutField = []
                for xi in SelIndexList:
                    outPutField.append(
                        tableName_Order[xi[0]].strip() + '.' + current_field[xi[0]][xi[1]][0].strip())
                return outPutField, current_list, True
            idx -= 1

    outPutField, current_list, isRight = excute_tree()
    if isRight:
        print outPutField
        for record in current_list:
            print record
    else:
        print 'WRONG SQL INPUT!'
else:
    print 'there is no query plan tree for the execution'
```

实验结果

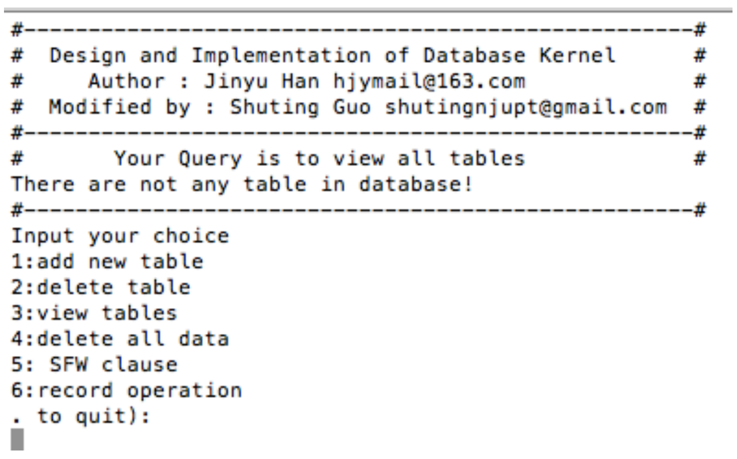
第一部分

（1）第一部分界面



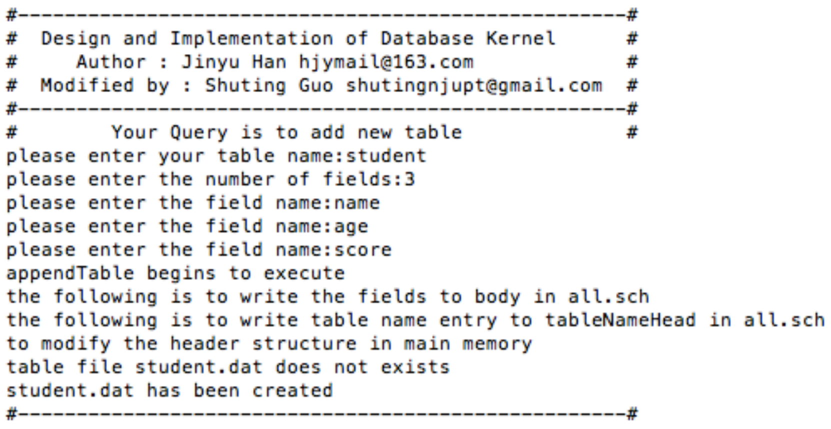
```
#--------------------------------------------------#
#  Design and Implementation of Database Kernel     #
#     Author : Jinyu Han hjymail@163.com            #
#  Modified by : Shuting Guo shutingnjupt@gmail.com #
#--------------------------------------------------#
#    There are two storage in this system            #
#    1: megatron2000        0: Storage               #
#--------------------------------------------------#
please enter the select  :                      1
Input your choice
1:add new table
2:delete table
3:view tables
4:delete all data
5: SFW clause
6:record operation
. to quit):
```

（2）显示所有表



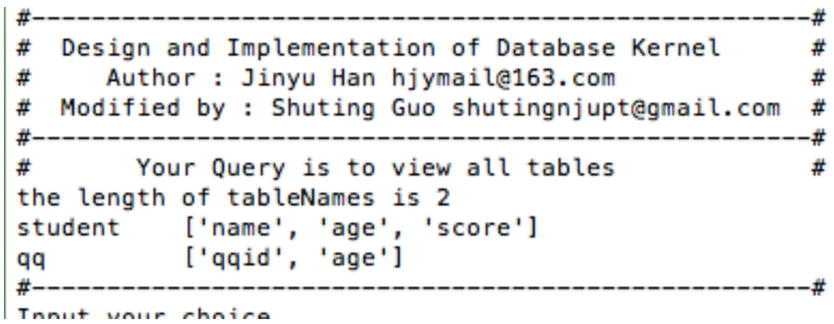
```
#--------------------------------------------------#
#  Design and Implementation of Database Kernel     #
#     Author : Jinyu Han hjymail@163.com            #
#  Modified by : Shuting Guo shutingnjupt@gmail.com #
#--------------------------------------------------#
#       Your Query is to view all tables             #
There are not any table in database!
#--------------------------------------------------#
Input your choice
1:add new table
2:delete table
3:view tables
4:delete all data
5: SFW clause
6:record operation
. to quit):
```

（3）插入表

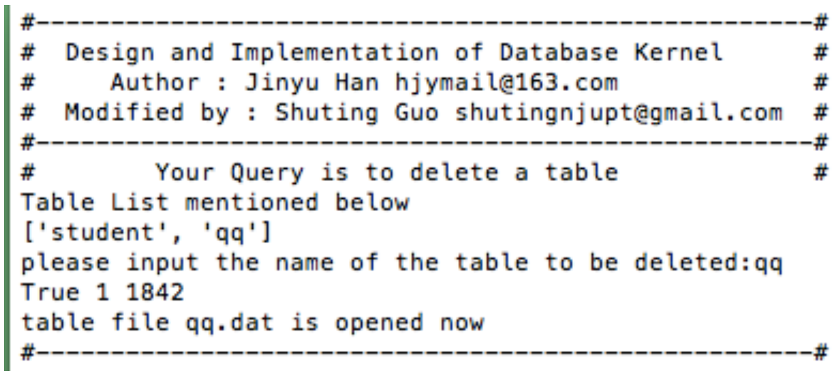


```
#--------------------------------------------------#
#  Design and Implementation of Database Kernel     #
#     Author : Jinyu Han hjymail@163.com            #
#  Modified by : Shuting Guo shutingnjupt@gmail.com #
#--------------------------------------------------#
#       Your Query is to add new table               #
please enter your table name:student
please enter the number of fields:3
please enter the field name:name
please enter the field name:age
please enter the field name:score
appendTable begins to execute
the following is to write the fields to body in all.sch
the following is to write table name entry to tableNameHead in all.sch
to modify the header structure in main memory
table file student.dat does not exists
student.dat has been created
#--------------------------------------------------#
```

（4）删除表

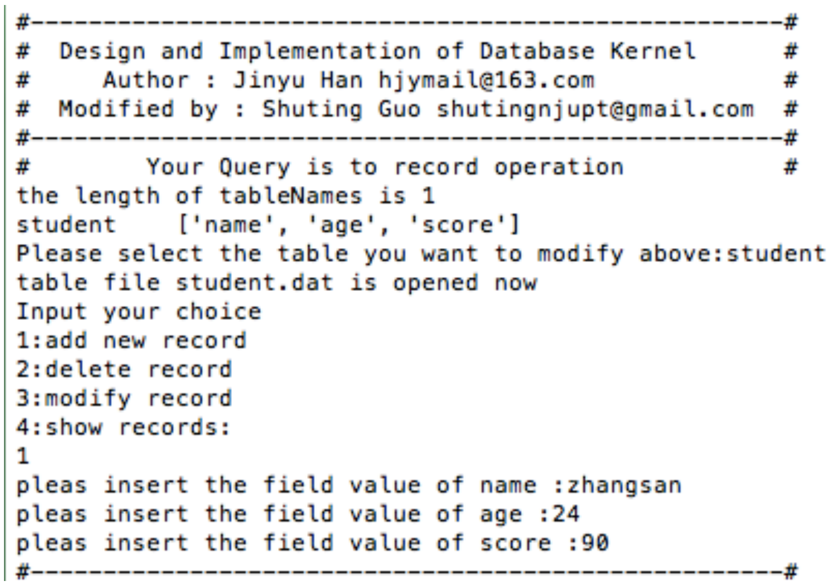


```
#-----------------------------------------------------#
#  Design and Implementation of Database Kernel       #
#     Author : Jinyu Han hjymail@163.com              #
#  Modified by : Shuting Guo shutingnjupt@gmail.com   #
#-----------------------------------------------------#
#       Your Query is to view all tables              #
the length of tableNames is 2
student     ['name', 'age', 'score']
qq          ['qqid', 'age']
#-----------------------------------------------------#
Input your choice
```

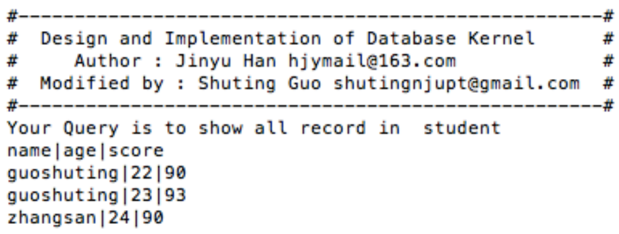


```
#-----------------------------------------------------#
#  Design and Implementation of Database Kernel       #
#     Author : Jinyu Han hjymail@163.com              #
#  Modified by : Shuting Guo shutingnjupt@gmail.com   #
#-----------------------------------------------------#
#        Your Query is to delete a table              #
Table List mentioned below
['student', 'qq']
please input the name of the table to be deleted:qq
True 1 1842
table file qq.dat is opened now
#-----------------------------------------------------#
```

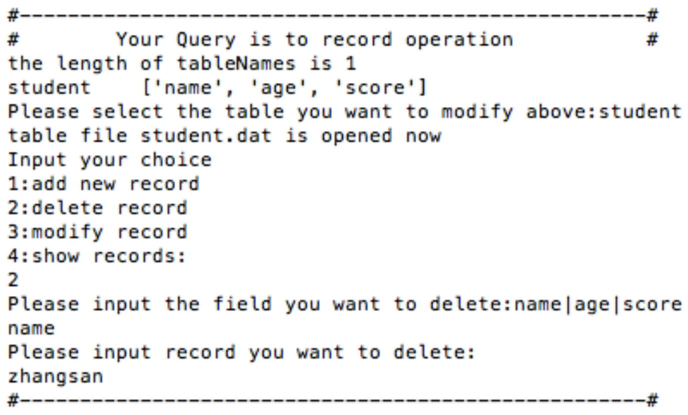
（5）插入记录

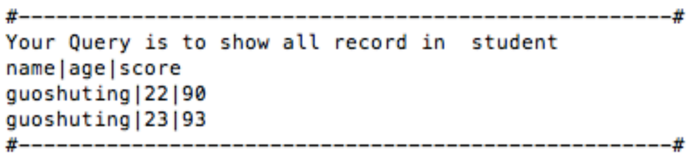


```
#-----------------------------------------------------#
#  Design and Implementation of Database Kernel       #
#     Author : Jinyu Han hjymail@163.com              #
#  Modified by : Shuting Guo shutingnjupt@gmail.com   #
#-----------------------------------------------------#
#       Your Query is to record operation             #
the length of tableNames is 1
student     ['name', 'age', 'score']
Please select the table you want to modify above:student
table file student.dat is opened now
Input your choice
1:add new record
2:delete record
3:modify record
4:show records:
1
pleas insert the field value of name :zhangsan
pleas insert the field value of age :24
pleas insert the field value of score :90
#-----------------------------------------------------#
```

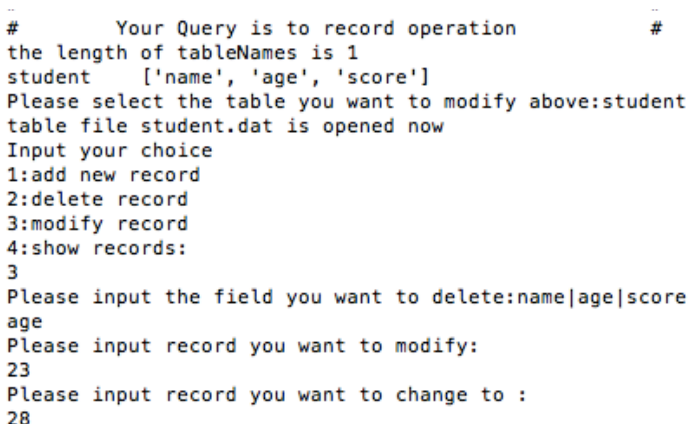

```
#----------------------------------------------------#
#  Design and Implementation of Database Kernel      #
#     Author : Jinyu Han hjymail@163.com             #
#  Modified by : Shuting Guo shutingnjupt@gmail.com  #
#----------------------------------------------------#
Your Query is to show all record in  student
name|age|score
guoshuting|22|90
guoshuting|23|93
zhangsan|24|90
```

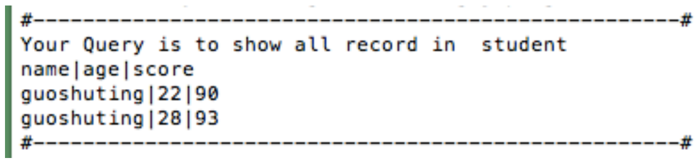
（6）删除记录


```
#----------------------------------------------------#
#        Your Query is to record operation           #
the length of tableNames is 1
student    ['name', 'age', 'score']
Please select the table you want to modify above:student
table file student.dat is opened now
Input your choice
1:add new record
2:delete record
3:modify record
4:show records:
2
Please input the field you want to delete:name|age|score
name
Please input record you want to delete:
zhangsan
#----------------------------------------------------#
```

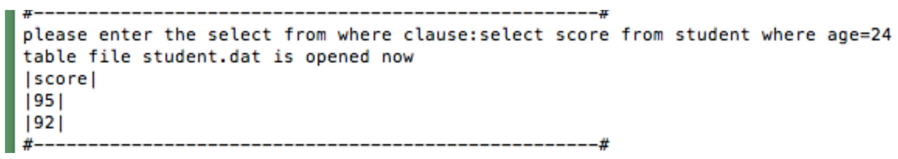

```
#----------------------------------------------------#
Your Query is to show all record in  student
name|age|score
guoshuting|22|90
guoshuting|23|93
#----------------------------------------------------#
```

（7）修改记录


```
#        Your Query is to record operation           #
the length of tableNames is 1
student    ['name', 'age', 'score']
Please select the table you want to modify above:student
table file student.dat is opened now
Input your choice
1:add new record
2:delete record
3:modify record
4:show records:
3
Please input the field you want to delete:name|age|score
age
Please input record you want to modify:
23
Please input record you want to change to :
28
```


```
#----------------------------------------------------#
Your Query is to show all record in  student
name|age|score
guoshuting|22|90
guoshuting|28|93
#----------------------------------------------------#
```

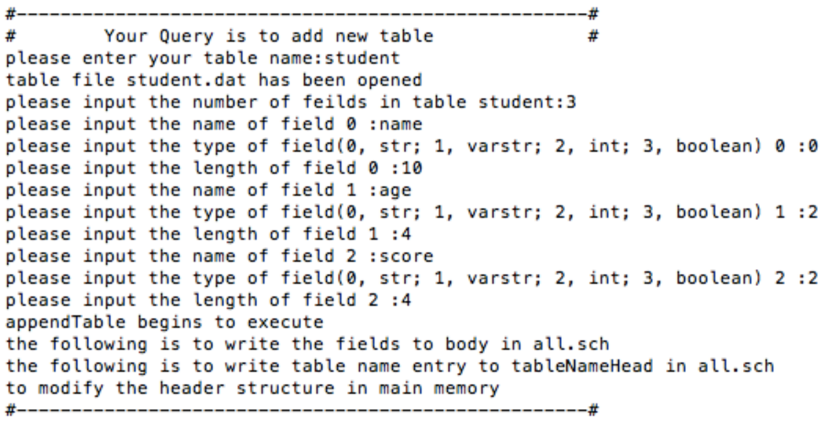
（8）SQL语句查询



```
#--------------------------------------------------------#
please enter the select from where clause:select score from student where age=24
table file student.dat is opened now
|score|
|95|
|92|
#--------------------------------------------------------#
```

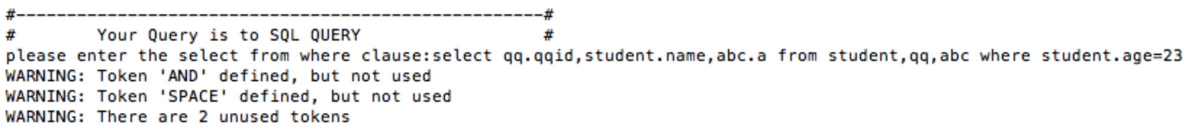
第二部分

（1）插入表

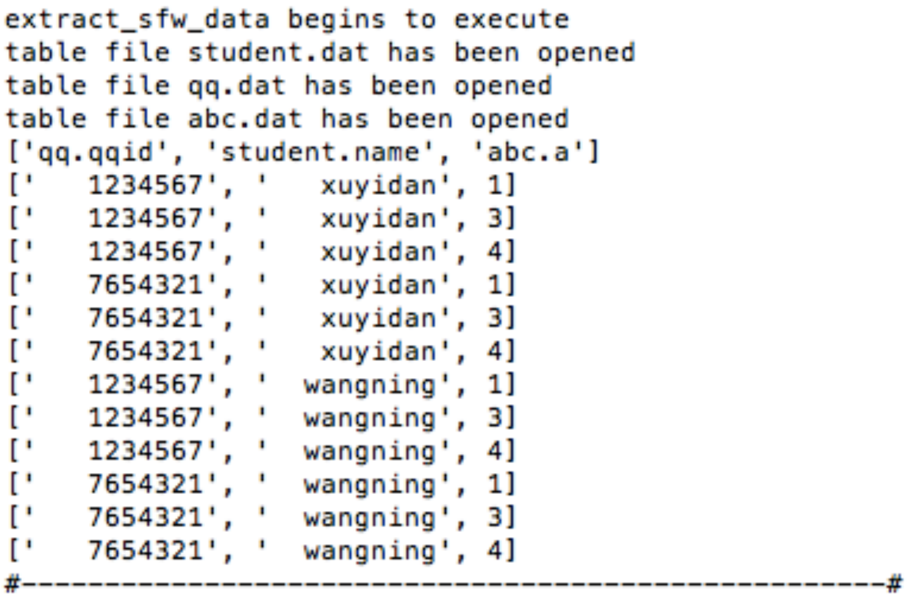


```
#--------------------------------------------------------#
#        Your Query is to add new table                  #
please enter your table name:student
table file student.dat has been opened
please input the number of feilds in table student:3
please input the name of field 0 :name
please input the type of field(0, str; 1, varstr; 2, int; 3, boolean) 0 :0
please input the length of field 0 :10
please input the name of field 1 :age
please input the type of field(0, str; 1, varstr; 2, int; 3, boolean) 1 :2
please input the length of field 1 :4
please input the name of field 2 :score
please input the type of field(0, str; 1, varstr; 2, int; 3, boolean) 2 :2
please input the length of field 2 :4
appendTable begins to execute
the following is to write the fields to body in all.sch
the following is to write table name entry to tableNameHead in all.sch
to modify the header structure in main memory
#--------------------------------------------------------#
```

（2）SQL查询



```
#--------------------------------------------------------#
#        Your Query is to SQL QUERY                      #
please enter the select from where clause:select qq.qqid,student.name,abc.a from student,qq,abc where student.age=23
WARNING: Token 'AND' defined, but not used
WARNING: Token 'SPACE' defined, but not used
WARNING: There are 2 unused tokens
```



```
extract_sfw_data begins to execute
table file student.dat has been opened
table file qq.dat has been opened
table file abc.dat has been opened
['qq.qqid', 'student.name', 'abc.a']
['   1234567', '    xuyidan', 1]
['   1234567', '    xuyidan', 3]
['   1234567', '    xuyidan', 4]
['   7654321', '    xuyidan', 1]
['   7654321', '    xuyidan', 3]
['   7654321', '    xuyidan', 4]
['   1234567', '   wangning', 1]
['   1234567', '   wangning', 3]
['   1234567', '   wangning', 4]
['   7654321', '   wangning', 1]
['   7654321', '   wangning', 3]
['   7654321', '   wangning', 4]
#--------------------------------------------------------#
```

实验小结

因为时间匆忙，有许多内容其实都可以再进一步改进，比如storage中的记录删除中回收空间的处理，还有SQL函数的进一步完善，根据代价函数对查询处理器进行优化，以及用户体验方面的改进等等。

这学期的数据库课程收获颇丰，让我一个从未接触过数据库的人体会了下数据库内核的设计和实现，加深了对数据库的理解。特别是语法树的内容，让我受益匪浅。

参考文献

[1] HectorGarcia-Molina, JeffreyD.Ullman, JenniferWidom,等. 数据库系统实现[M]. 机械工业出版社, 2010.

[2] 赫特兰, M. L.). Python基础教程: 第2版[M]. 人民邮电出版社, 2010.

[3] https://docs.python.org/2/library/struct.html

[4] http://www.w3school.com.cn/sql/