DataBase MiniSQL 设计报告

秋冬学期数据库系统Project

作者：范源颢 3180103574 课程：秋冬学期数据库系统（周一一、二节）

宋天泽 3180105221 指导老师：庄越挺 慕宗燊

李国耀 3180103687

江雨辰 3180106317

用于回应课题：

“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。”

“报告应该包括1.整体框架 2.各模块实现功能 3.分工说明 4.各模块提供接口和内部实现 5.界面说明 6.系统测试这几部分”

# 概论

本实验工程是秋冬学期数据库系统的期末作业，功能是实现一个精简单用户SQL引擎MiniSQL。实验是单用户的，因此并不涉及用户登录的问题，在运行程序之后即可开始对于数据库的操作，支持的操作包括建立/删除表，索引的建立/删除以及表记录的插入/删除/查找。

## 1.1开发环境和使用方法

本工程通过系统兼容性较强的python语言实现，使用的语言标准可以在python3.7-3.8的版本中正常运行（尚未测试更加古老的版本）。实验过程中的开发环境，操作系统通常是win10，工程软件包括pycharm、vscode、visual studio等等，均可以在装有上述软件的win10电脑中右键点击文件夹-点击“通过pycharm打开”/“通过vscode打开”等打开文件夹查看代码，实验的主程序是Interpreter.py，在终端中输入python Interpreter.py即可开始运行程序。运行过程中可以使用quit命令退出程序。

## 1.2支持的数据类型

按照本实验的期末作业要求，我们支持三种基本数据类型：int【整型变量】，char(n)【带长度限定的字符型变量，1<= n <= 255】，float【浮点型小数变量】。

## 1.3支持的表结构

实验中创建的表最多包含32个属性，各个属性可以指定是否为unique，支持单属性主键定义，不过要求表在创建时必须包含主键的声明（否则程序报错primary\_key 'None' does not exist. 【主键不存在】，拒绝创建表）。

创建表的语句格式如下：

create table 表名 (

列名 类型 ,

列名 类型 ,

列名 类型 ,

primary key ( 列名 )

);

删除表使用drop命令：

drop table 表名 ;

## 1.4支持的索引结构

实验中涉及到的索引都是单属性单值的，对于表的主属性，程序将自动建立B+树的索引， 对于用户声明为unique的属性，用户可以通过SQL语句制定建立/删除B+树索引。

相关命令如下：

创建索引：

create index 索引名 on 表名 ( 列名 );

删除索引：

drop index 索引名;

## 1.5支持的查询语言

实验中设计的数据库访问语言，支持的包括查找（select）、删除（delete）、插入（insert）。Delete和insert查询语句都支持使用where条件句，条件允许使用and、or等逻辑词并列，条件中也可以出现 = <> < > <= >=等比较运算符（但暂时不支持算数运算）。

相关语句的格式如下：

选择：（示例语句， \* 表示全部）

select \* from student;

select \* from student where sno = ‘88888888’;

select \* from student where sage > 20 and sgender = ‘F’;

插入：

insert into 表名 values ( 值1 , 值2 , … , 值n );

删除：（示例语句）

delete from student;

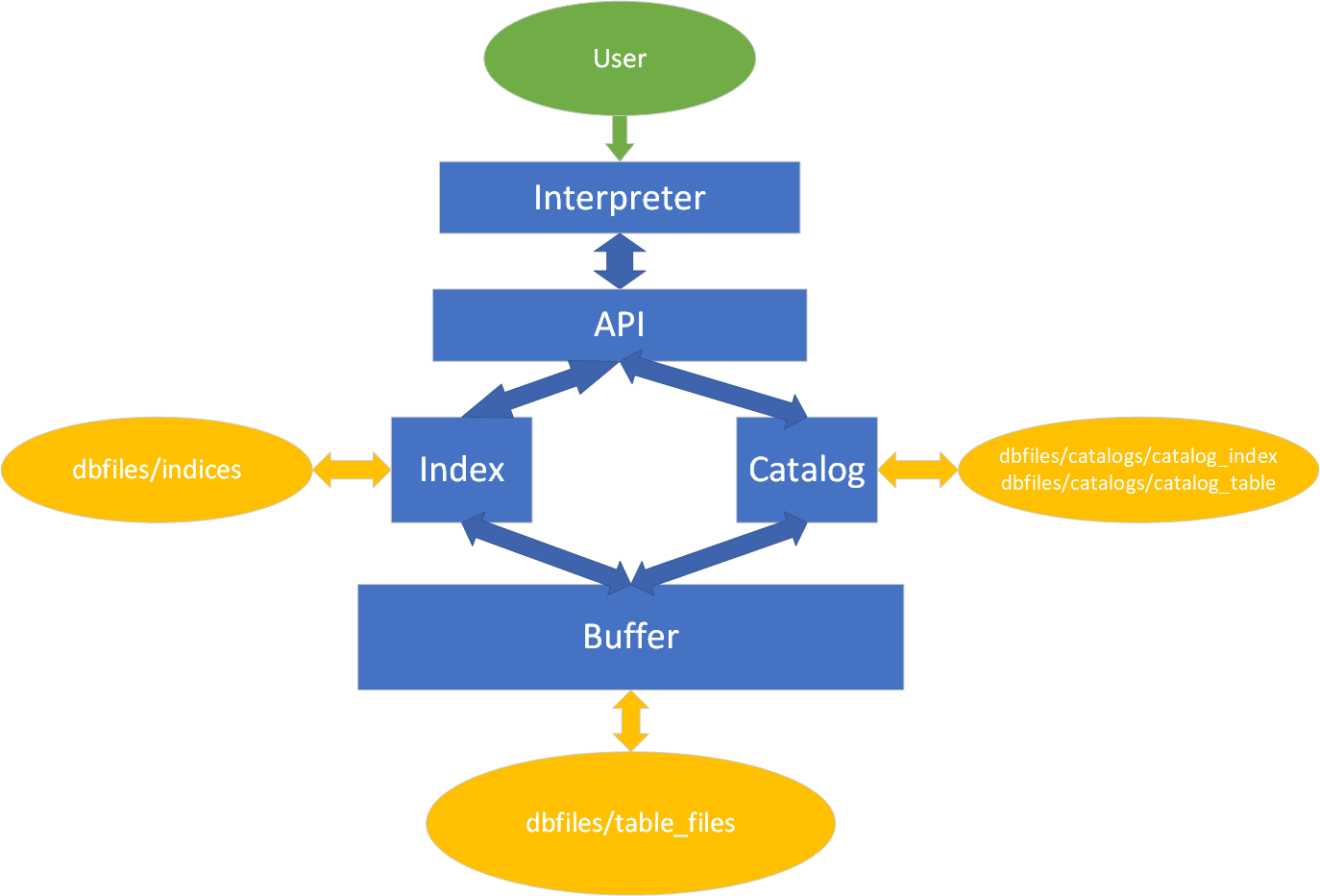
delete from student where sno = ‘88888888’;

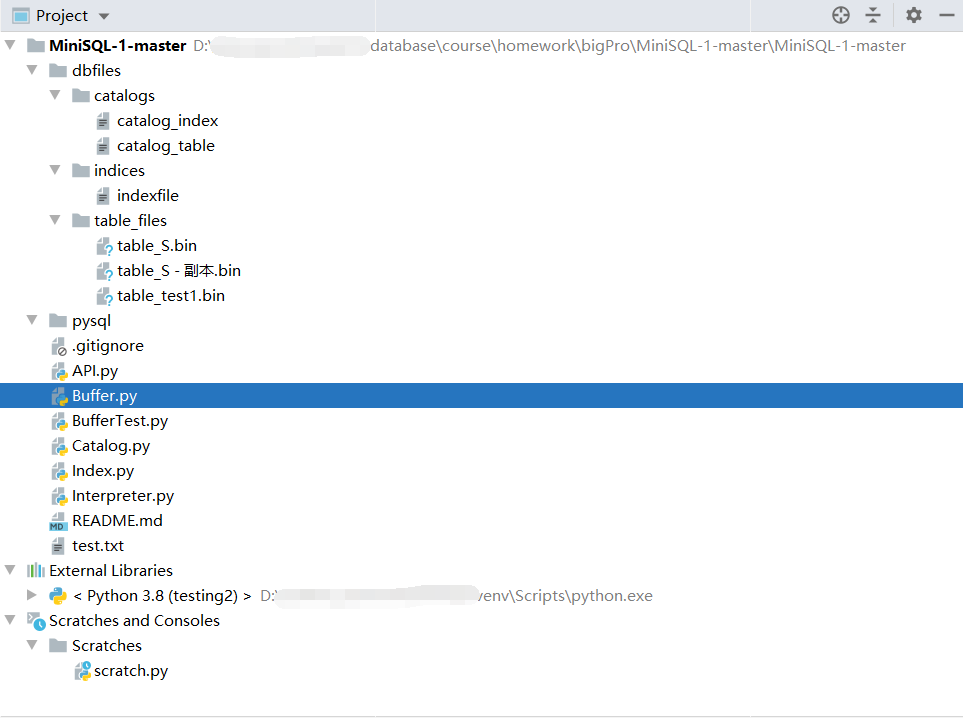
相关的查询语言可以编写为execfile脚本，其中可以包含任意多条上述的SQL语言，程序读入之后按照顺序执行脚本中的指令。

# 整体框架

实验按照题目给出的建议，我们设计了Interpreter.py和API.py作为交互接口，之后，我们插入的所有record都在Buffer.py中暂存（这里注意和实验课题的建议有所不同，我们没有recordManager或者与之相似的模块，实验建议中，应该由recordManager实现的内容都被Buffer.py实现了，亦即，在本程序中。RecordManager 和BufferManager 合并了，合并后的名字在这里是Buffer.py）之后，我们的Index.py和Catalog.py分别处理用户输入的索引信息和表头信息，然后，表头信息、索引信息，连同Buffer处理过的记录信息全部被存储到了dbfiles文件夹下。表头信息，也就是各个表的名称和行、列属性储存在catalogs文件夹下，由catalog\_index和catalog\_table分别储存。索引信息由indices下的indexfile储存，各个表格的记录则由table\_files下的各个table\_***tableName***储存。

我们总体的模块构成可以用下图表示：



 存储目录如下：

由上图可知，已经创建的表格包括‘S’和‘test1’。

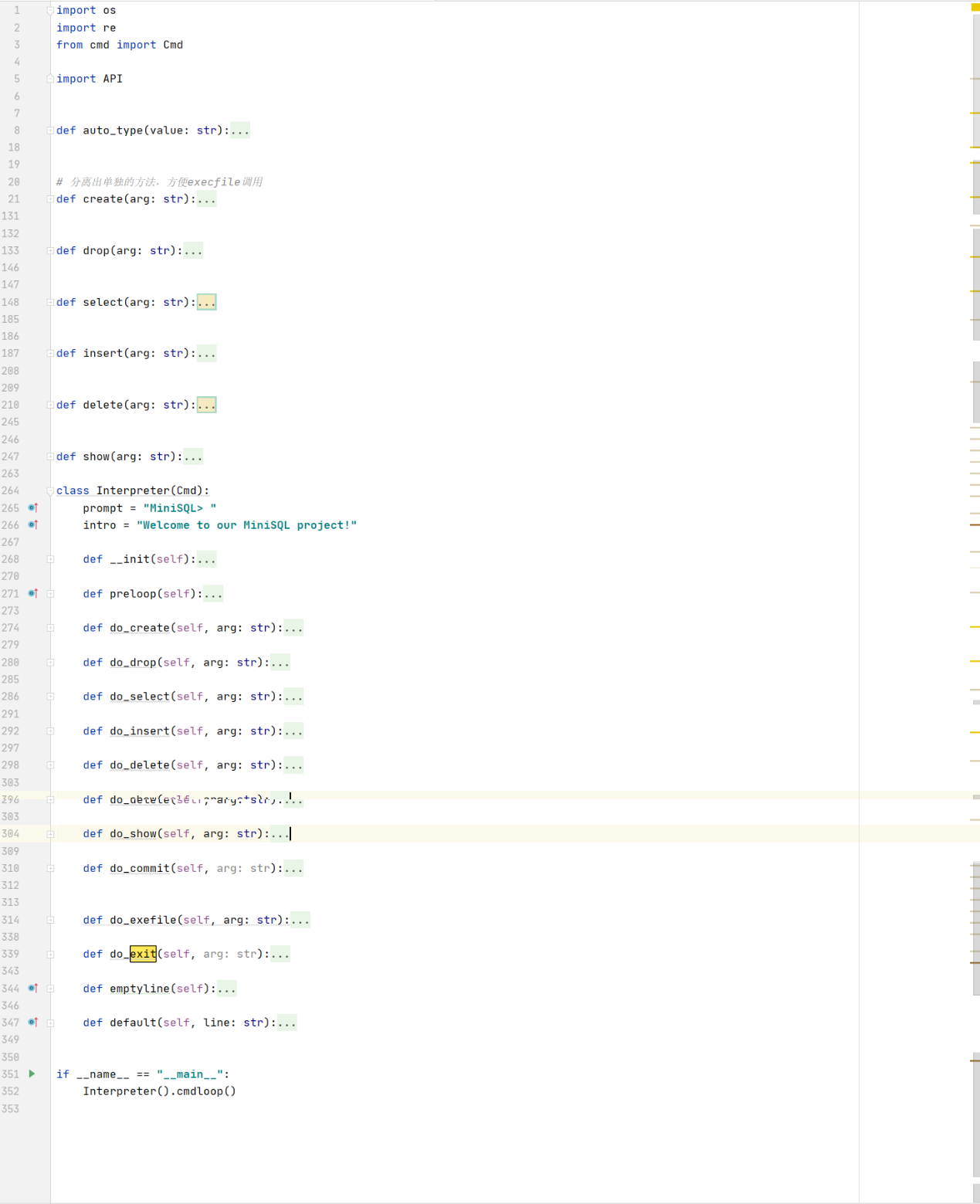
# 各模块实现功能

## 3.1 Interpreter

Interpreter （义即，解释器）模块直接与用户交互，是程序的前段，主要实现以下功能:

1. 程序流程控制，即启动并且初始化、接受命令、处理命令、显示命令结果、循环、退出等等流程
2. 处理用户输入的命令，主要通过python的re库中的正则表达式方法，将用户在终端输入的指令转化为可释读的结构，将语句做剖析（parse），同时先进行简单的语法检查，确认是支持的语句之后，通过调用API来实现相关的命令，如果不是合法的语句，则会通过错误捕捉机制打印相关的错误。
3. SQL脚本文件执行的部分也在这个模块之内完成。

文本结构：



## 3.2 API

API（application program interface义即应用程序编程接口）是程序前段和后端的接口，是系统的核心部件，主要功能就是将Interpreter层解析出的函数作为模块输入，之后根据Catalog提供的信息作为执行规则，调用Index、 Catalog、Buffer提供的相应接口执行，最后返回的执行结果给Interpreter。

主要实现的函数包括:

create\_table, 创建表

create\_index,创建目录

drop\_table,删除表

drop\_index,删除目录

select,选择记录

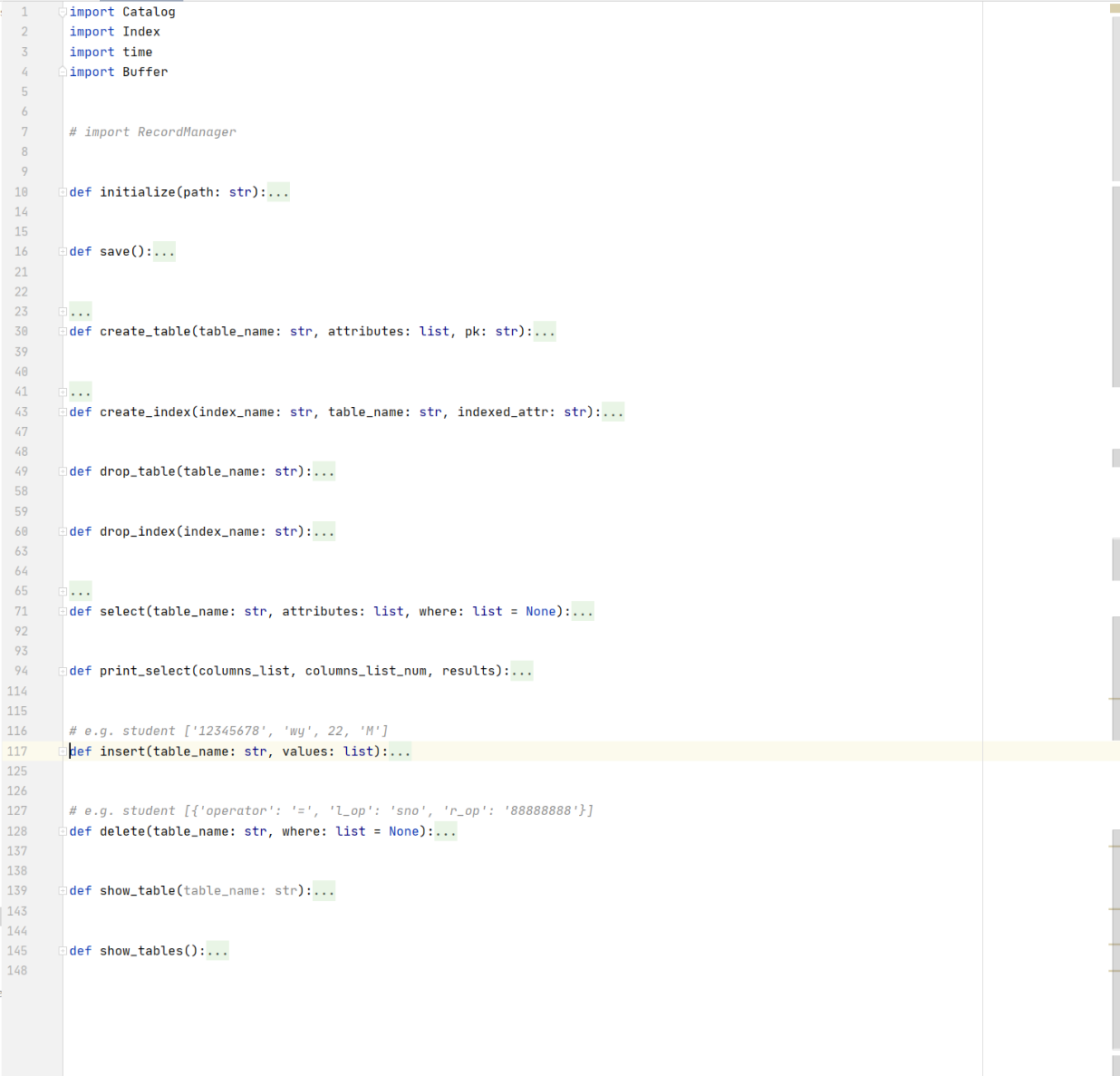
print\_select 用于进行选择结果的打印

insert，插入记录

delete，删除记录

show\_table，展示数据库某个特定表的表头信息

show\_tables, 展示数据库中所有的表

文件总览：

## 3.3 Catalog

这里的catalog（编目）是catalog Manager的意思，他主要负责实现数据库的所有模式信息，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、定义在该表上的索引。
2. 表中每个字段的定义信息，包括字段类型、是否唯一等。
3. 数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。

Catalog还提供了访问及操作的接口，供Interpreter和API模块使用。

Catalog定义了表头文件tables，目录文件indices，同时也指定了相关的文件路径，即catalogpath、tablecatalog、indexcatalog。

之后Catalog设定了两个比较重要的类，Table()：记录表的名字，他的主键，以及每列的名字；Column()：组成Table类的一个结构，表示的是Table的各个attribute，记录了这一列的数据类型，以及名称、是否限定不可重复，总长度等等信息。

Catalog随后实现了以下功能：

create table:将API反馈的table写入到tables字典储存（这一步没有通过buffer，直接写入）

drop\_table：将相关的表从tables 字典对象中删除

check\_types\_of\_table: 根据tables字典对象，检查table中各个属性的类型和用户插入的是否一致的函数，被API 模块调用，如果不满足则抛出一个异常。

exists\_table：根据tables字典对象，检查table是否已经存在的函数，若有则抛出异常，也是被API调用的函数；

not\_exist\_table：根据tables字典对象，检查table是否不存在的函数，若确实不存在则抛出异常，也是被API调用的函数，用于对于用户的行为作出检查。

exists\_index: 根据indices字典对象，检查index是否存在，若已经存在则抛出异常，被API调用；

exists\_index：根据indices字典对象，检查index是否不存在的函数，功用同上；

drop\_index：实现将索引从indices字典对象中删除的功能；

create\_index：实现创建索引并且储存到indices字典对象中的功能；

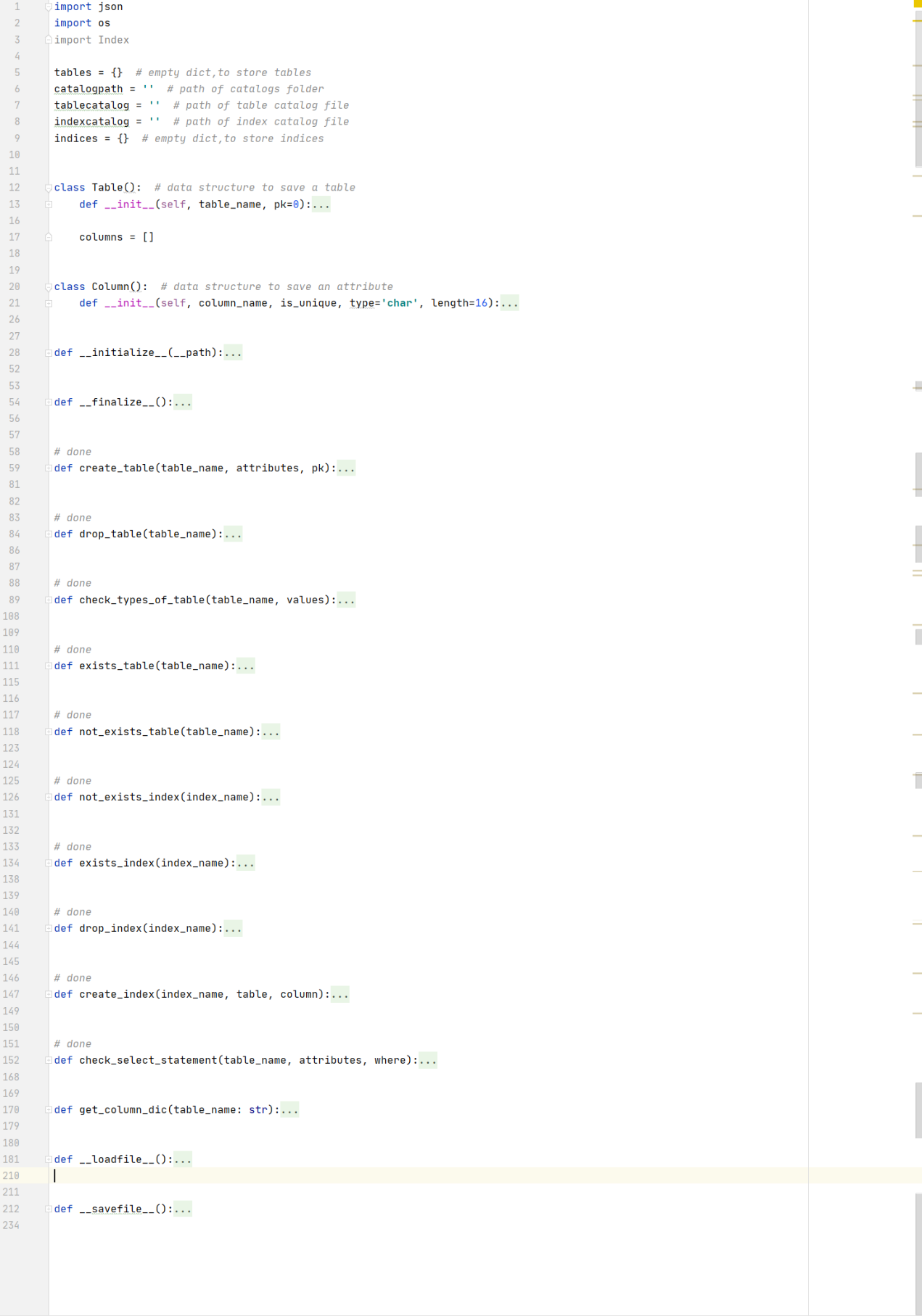
check\_select\_statement：根据tables字典检查select语法的函数。

get\_column\_dic：返回相关表的属性的函数，用在API中，方便select、delete的实现

\_\_loadfile\_\_：将table\_fies文件中读出，然后储存到tables字典中，同时将indices文件读出，将相关变量储存到indices字典中；

\_\_savefile\_\_: 将对于indices和tables字典的修改保存到相关的磁盘文件中。

文件总览：



## Index

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文件中定义了B+树的节点：node类型。

之后我们实现了以下函数：

\_\_load\_\_: 从indices文件夹下的indexfile文件中读入B+树的信息；

load\_nodes: 将indexfile中的B+树解析成一个个树节点，还原成可操作的B+树的pointer\_list和node\_list。

\_\_store\_\_:将操作之后的B+树存储回indexfile文件中, 调用recursive\_store\_node实现；

recursive\_store\_node: 递归法将各个node中的信息存储到相应文件中。

insert\_into\_table: 根据API解析后的insert命令修改B+树内容；

create\_table：根据API解析后的create命令增加一颗B+树；

delete\_from\_table：根据API解析后的delete命令删除B+树的记录；

check\_conditions：检查select语句中的操作符（==， >=等等）便于搜索；

maintain\_B\_plus\_tree\_after\_delete：在delete之后修正B+树的结构；

create\_index: 创建索引以及相关内容

print\_select: 将搜索的结果打印出来，被API的同名函数调用。

select\_from\_table: 实现API中的select函数

check\_unique: 检查用户的操作是否破坏了unique约束

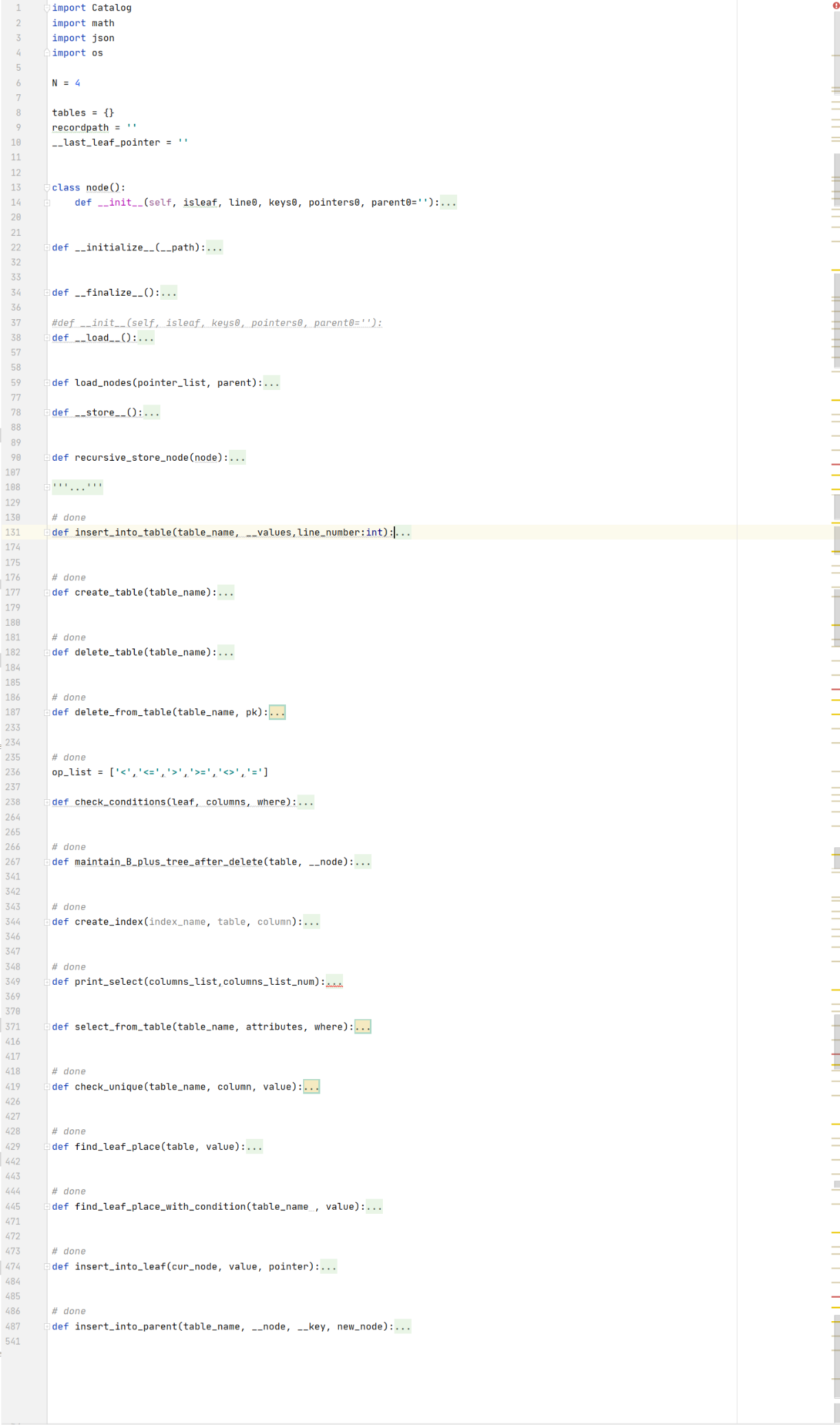
find\_leaf\_place：B+树操作，用于快速寻找特定的B+树节点；

find\_leaf\_place\_with\_condition：B+树操作，用于快速寻找B+树中一个用户指定的区间；

insert\_into\_leaf：B+树操作，用于在B+树的叶结点中插入一条记录；

insert\_into\_parent: B+树操作，递归的插入需要的parent节点，用于维护。

文件概览：



## Buffer

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。

Buffer中我们指定的缓冲大小比较小，用BUFFER\_SIZE来设置缓冲区可以包含的记录数。

Buffer文件中定义了Buffer类，用来描述buffer的大小信息，并且定义了判定是否满，是否可写入，以及保存等方法。

Buffer中实现的函数如下：

check: 检查某条记录是否符合where条件；

decode: 将字节串按照格式解码为实际的记录；

find\_attr\_pos: 找到当前的属性是名是第几列；

find\_line: 从缓冲区或文件取得对应行数的记录

find\_record: 从缓冲区或文件取得符合条件的记录

delete\_line: 从缓冲区或文件删除对应行数的记录

delete\_record: 从缓冲区或文件删除符合条件的记录

check\_unique: 检查待插入记录中unique属性的唯一性是否被破坏

insert\_record: 插入记录

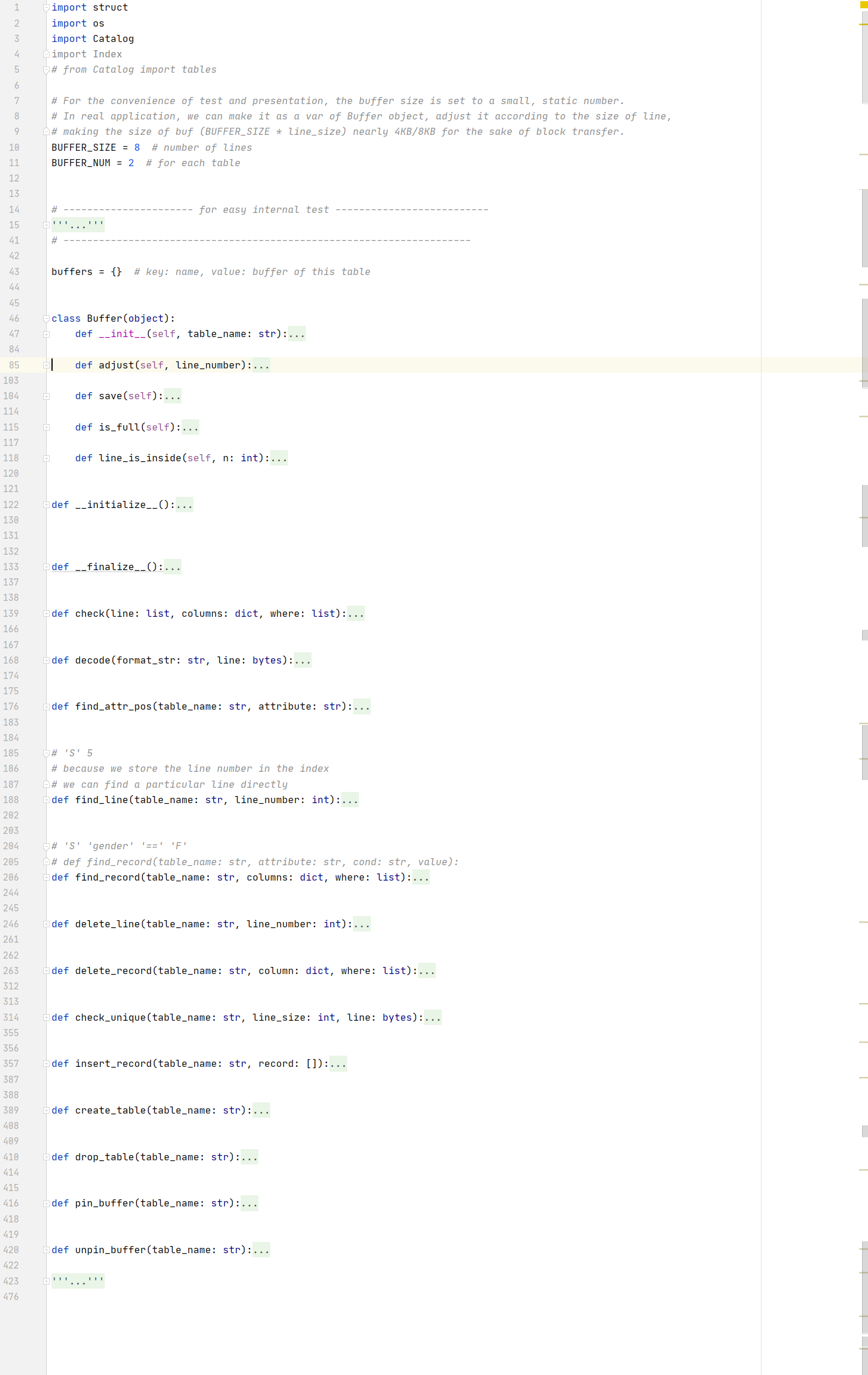
create\_table: 创建新的表文件

drop\_table: 删除表文件

pin\_buffer: 锁定缓冲区，不允许替换

unpin\_buffer: 解除缓冲区锁定

文件概览：



# 分工说明

宋天泽：interpreter、API、Buffer

李国耀：Index、Catalog、Buffer

江雨辰：Index、Catalog、Buffer

范源颢：interpreter、API、Index

# 各模块接口和内部实现

## 5.1 Interpreter

Interpreter的实现借助了python的包OS（operation system操作系统包），re（regular expression正则表达式包）以及cmd（command，命令行包）。

其实现的主要逻辑如下：

### 5.1.1 类型转换

我们首先根据正则表达式，设定auto\_type函数将语句中的整形数字、浮点型数字、以及引号内的字符串转化为相关的变量类型，语句的这些部分不再理解为关键字。

**def** auto\_type**(**value**:** **str):**

**if** value**[**0**]** **==** "'" **and** value**[-**1**]** **==** "'"**:**

value **=** value**[**1**:-**1**]**

**elif** re**.**match**(**r'^-?[0-9]+\.+[0-9]+$'**,** value**):**

value **=** **float(**value**)**

**elif** re**.**match**(**r'^-?[0-9]+'**,** value**):**

value **=** **int(**value**)**

**else:**

**raise** **Exception(**f"Unsupported format: {value}"**)**

**return** value

### 5.1.2 句法分析

之后我们进行句法分析，根据cmd中读入的字符，我们将用户输入的合法字符串分类为create、drop、select、insert、delete、show这几类，每一种按照相关的语法拆分这些字符串，然后将相关的关键字下的内容传递给API，使得API完成用户根据完整句子给出的需求，

这里谨以最复杂的create为例，展示句法分析如何完成：

create**(**arg**:** **str):**

arg **=** arg**.**strip**()**

**if** arg**[-**1**]** **==** '('**:** # 支持create的多行输入

**while** 1**:**

line **=** **input().**rstrip**()**

arg **=** arg **+** line

**if** line**[-**1**]** **!=** ',' **and** line**[-**1**]** **!=** ')'**:**

**break**

arg **=** arg**.**rstrip**(**';'**).**strip**()** # 去尾部分号

arg **=** re**.**sub**(**' +'**,** ' '**,** arg**)** # 将多空格换为单空格（两边是没有空格的）

**if** arg**[:**5**]** **==** 'table'**:**

arg **=** arg**[**5**:]**

arg **=** arg**.**lstrip**()** # 去table后空格

table\_name **=** arg**[:**arg**.**find**(**'('**)].**strip**()** # 通过定位'('获取表名

**if** table\_name **==** ''**:**

**raise** **Exception(**"No table name found."**)**

# 去除定义表的括号

arg **=** arg**[**arg**.**find**(**'('**):]**

arg **=** arg**.**lstrip**(**'('**).**strip**()**

**if** arg**[-**1**]** **==** ')'**:**

arg **=** arg**[:-**1**]**

arg **=** arg**.**strip**()**

**if** arg **==** ''**:**

**raise** **Exception(**"No table specification found."**)**

# 获取属性定义、pk等

attribute\_specifications **=** arg**.**split**(**','**)**

attribute\_specifications **=** **list(map(str.**strip**,** attribute\_specifications**))**

**if** attribute\_specifications **==** **[]:**

**raise** **Exception(**"No table attribute found."**)**

# 先处理pk

pk **=** **None**

**if** attribute\_specifications**[-**1**].**startswith**(**'primary key'**):**

pk **=** attribute\_specifications**[-**1**]**

**if** ',' **in** pk**:**

**raise** **Exception(**"Only single primary key is supported."**)**

pk **=** pk**[**11**:].**strip**().**lstrip**(**'('**).**rstrip**(**')'**).**strip**()**

attribute\_specifications **=** attribute\_specifications**[:-**1**]**

# 依序处理属性定义

attributes **=** **[]**

attribute\_names **=** **[]**

**for** attribute\_specification **in** attribute\_specifications**:**

# item: attribute name, type, and optional unique

unique **=** **False**

type\_len **=** 0

item **=** attribute\_specification**.**split**(**' '**)**

**if** item**[**1**]** **not** **in** **[**'int'**,** 'float'**]:**

**if** item**[**1**].**startswith**(**'char'**):**

type\_len **=** **int(**item**[**1**][**4**:].**strip**().**lstrip**(**'('**).**rstrip**(**')'**))**

**if** type\_len **<=** 0**:**

**raise** **Exception(**f"The size of the type is negative."**)**

item**[**1**]** **=** 'char'

**else:**

**raise** **Exception(**f"The type of attribute {item**[**0**]**} is {item**[**1**]**}, which is not supported."**)**

**if** **len(**item**)** **==** 3**:**

**if** item**[**2**]** **==** "unique"**:**

unique **=** **True**

**else:**

**raise** **Exception(**f"The command behind {item**[**0**]**} {item**[**1**]**} is not supported."**)**

attribute\_names**.**append**(**item**[**0**])**

attributes**.**append**({**

'attribute\_name'**:** item**[**0**],**

'type'**:** item**[**1**],**

'type\_len'**:** type\_len**,**

'unique'**:** unique

**})**

**if** pk**:**

**if** pk **not** **in** attribute\_names**:**

**raise** **Exception(**f"The primary key {pk} you want is not in the attribute list."**)**

**else:**

attributes**[**attribute\_names**.**index**(**pk**)][**'unique'**]** **=** **True**

**print(**table\_name**,** attributes**,** pk**)**

API**.**create\_table**(**table\_name**,** attributes**,** pk**)**

**elif** arg**[:**5**]** **==** 'index'**:**

arg **=** arg**[**5**:]**

arg **=** arg**.**lstrip**()** # 去index后空格

location\_on **=** arg**.**find**(**'on'**)**

**if** location\_on **==** **-**1**:**

**raise** **Exception(**f"'on' is missing when creating index."**)**

index\_name **=** arg**[:**location\_on**].**strip**()**

location\_lbracket **=** arg**.**find**(**'('**)**

**if** location\_lbracket **==** **-**1**:**

**raise** **Exception(**f"Indexed attribute format is wrong."**)**

table\_name **=** arg**[**location\_on **+** **len(**'on'**):** location\_lbracket**].**strip**()**

location\_rbracket **=** arg**.**find**(**')'**)**

**if** location\_rbracket **==** **-**1**:**

**raise** **Exception(**f"Indexed attribute format is wrong."**)**

indexed\_attr **=** arg**[**location\_lbracket**+**1**:** location\_rbracket**].**strip**()**

**if** ',' **in** indexed\_attr**:**

**raise** **Exception(**'Only single attribute index is supported.'**)**

**print(**index\_name**,** table\_name**,** indexed\_attr**)**

API**.**create\_index**(**index\_name**,** table\_name**,** indexed\_attr**)**

**else:**

**raise** **Exception(**"The item you want to create is not supported."**)**

可以看到，在处理完了所有报错之后，我们将创建表或者创建索引的任务交给了API

### 5.1.3 用户的命令行交互

在实现了用户的句法分析之后，我们还需要通过终端捕捉用户的行为，为此我们使用了python含有的cmd类来实现程序和用户的交互

Cmd类主要给出了对用户的提示和导引，对于用户的操作解析全部放到了5.1.2的那些句法分析函数之中了，此类的主要责任是使用try：except语法给出必要的报错

**class** **Interpreter(**Cmd**):**

prompt **=** "MiniSQL> "

intro **=** "Welcome to our MiniSQL project!"

**def** \_\_init**(**self**):**

Cmd**.**\_\_init\_\_**(**self**)**

**def** preloop**(**self**):**

API**.**initialize**(**os**.**getcwd**())**

**def** do\_create**(**self**,** arg**:** **str):**

**try:**

create**(**arg**)**

**except** **Exception** **as** e**:**

**print(**e**)**

**def** do\_drop**(**self**,** arg**:** **str):**

**try:**

drop**(**arg**)**

**except** **Exception** **as** e**:**

**print(**e**)**

**def** do\_select**(**self**,** arg**:** **str):**

**try:**

select**(**arg**)**

**except** **Exception** **as** e**:**

**print(**e**)**

**def** do\_insert**(**self**,** arg**:** **str):**

**try:**

insert**(**arg**)**

**except** **Exception** **as** e**:**

**print(**e**)**

**def** do\_delete**(**self**,** arg**:** **str):**

**try:**

delete**(**arg**)**

**except** **Exception** **as** e**:**

**print(**e**)**

**def** do\_show**(**self**,** arg**:** **str):**

**try:**

show**(**arg**)**

**except** **Exception** **as** e**:**

**print(**e**)**

**def** do\_commit**(**self**,** arg**:** **str):**

API**.**save**()**

**def** do\_exefile**(**self**,** arg**:** **str):**

switch **=** **{**

'create'**:** create**,**

'drop'**:** drop**,**

'select'**:** select**,**

'insert'**:** insert**,**

'delete'**:** delete**,**

'show'**:** show

**}**

i **=** 1

**try:**

f **=** **open(**arg**.**strip**(**';'**).**strip**(),** 'r'**)**

**while** 1**:**

line **=** f**.**readline**().**strip**()**

**if** line **==** ''**:**

**break**

command **=** line**[:**line**.**find**(**' '**)]**

arg **=** line**[**line**.**find**(**' '**):]**

switch**[**command**](**arg**)**

i **+=** 1

**except** **Exception** **as** e**:**

**print(**f"An exception occurred at line {i}:"**)**

**print(**e**)**

**pass**

**def** do\_exit**(**self**,** arg**:** **str):**

API**.**save**()**

**print(**'Bye~'**)**

**return** **True**

**def** emptyline**(**self**):**

**pass**

**def** default**(**self**,** line**:** **str):**

**print(**f"Unknown command: {line**.**split**(**' '**)[**0**]**}"**)**

### 5.1.4 主函数

在定义cmd类之后，我们所需要的的就是一个轮询的命令提示符的函数，就可以完成设计：

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

Interpreter**().**cmdloop**()**

## 5.2 API

API主要是程序的接口文件，负责将句法分析的结果转交给Catalog和Index来实现，然后根据这两个模块的返回结果反馈用户信息。然而，其实，真正用户需要即时反馈信息的命令只有select一个，其他的函数我们仅仅根据Catalog和Index的结果告诉用户是否有报错，如若成功，再依靠time()函数给出操作总共的用时。API作为一个接口，主要的任务是转交，因此它本身的代码量反倒不多，下文以期最复杂的select命令为例展示其实现过程。

**def** select**(**table\_name**:** **str,** attributes**:** **list,** where**:** **list** **=** **None):**

time\_start **=** time**.**time**()**

Catalog**.**not\_exists\_table**(**table\_name**)**

Catalog**.**check\_select\_statement**(**table\_name**,** attributes**,** where**)**

#Index.select\_from\_table(table\_name, attributes, where)

col\_dic **=** Catalog**.**get\_column\_dic**(**table\_name**)**

**print(**col\_dic**)**

results **=** Buffer**.**find\_record**(**table\_name**,** col\_dic**,** where**)**

**print(**results**)**

numlist **=** **[]**

**if** attributes **==** **[**'\*'**]:**

attributes **=** **list(**col\_dic**.**keys**())**

numlist **=** **list(**col\_dic**.**values**())**

**else:**

**for** att **in** attributes**:**

**print(**att**)**

numlist**.**append**(**col\_dic**[**att**])**

print\_select**(**attributes**,** numlist**,** results**)**

time\_end **=** time**.**time**()**

**print(**" time elapsed : %fs." **%** **(**time\_end **-** time\_start**))**

**def** print\_select**(**columns\_list**,** columns\_list\_num**,** results**):**

**print(**'-' **\*** **(**17 **\*** **len(**columns\_list\_num**)** **+** 1**))**

**for** i **in** columns\_list**:**

**if** **len(str(**i**))** **>** 14**:**

output **=** **str(**i**)[**0**:**14**]**

**else:**

output **=** **str(**i**)**

**print(**'|'**,** output**.**center**(**15**),** end**=**''**)**

**print(**'|'**)**

**print(**'-' **\*** **(**17 **\*** **len(**columns\_list\_num**)** **+** 1**))**

**for** i **in** results**:**

**for** j **in** columns\_list\_num**:**

**if** **len(str(**i**[**j**]))** **>** 14**:**

output **=** **str(**i**[**j**])[**0**:**14**]**

**else:**

output **=** **str(**i**[**j**])**

**print(**'|'**,** output**.**center**(**15**),** end**=**''**)**

**print(**'|'**)**

**print(**'-' **\*** **(**17 **\*** **len(**columns\_list\_num**)** **+** 1**))**

**print(**"Returned %d entries," **%** **len(**results**),** end**=**''**)**

可以看到，在正式插入之前，我们首先用Catalog模块的几个函数检查了语法是否和table文件夹下的表头定义是否相合，然后我们有调用Buffer模块的内容将相应的记录真正搜寻出来，之后我们用numlist暂存需要打印的属性（attribute，attr），最后通过print\_select函数将这些内容全部打印出来。

其他的功能也类似，在具体的某个函数中，如何和Catalog以及 Index联系是这个函数的功能所在，例如，相对简单的一个create\_table 的函数实现过程如下：

**def** create\_table**(**table\_name**:** **str,** attributes**:** **list,** pk**:** **str):**

time\_start **=** time**.**time**()**

Catalog**.**exists\_table**(**table\_name**)**

Index**.**create\_table**(**table\_name**)**

Catalog**.**create\_table**(**table\_name**,** attributes**,** pk**)**

Buffer**.**create\_table**(**table\_name**)**

time\_end **=** time**.**time**()**

**print(**"Successfully create table '%s', time elapsed : %fs." **%**

**(**table\_name**,** time\_end **-** time\_start**))**

## 5.3 Catalog

Catalog是程序和硬盘的接口，是将表头信息存储到相应的文件中的模块，和硬盘的交互主要是借助了python系统中的os模块（用于读取和存储硬盘的文件），以及json模块（用于高效率的将存储的表头信息读取出来）。

### 5.3.1 数据类型

在处理表的时候，我们定义了表的类型（class Table），表中的各个属性也有自己的信息，储存在class Column（）中。

**class** **Table():** # data structure to save a table

**def** \_\_init\_\_**(**self**,** table\_name**,** pk**=**0**):**

self**.**table\_name **=** table\_name

self**.**primary\_key **=** pk

columns **=** **[]**

**class** **Column():** # data structure to save an attribute

**def** \_\_init\_\_**(**self**,** column\_name**,** is\_unique**,** **type=**'char'**,** length**=**16**):**

self**.**column\_name **=** column\_name

self**.**is\_unique **=** is\_unique

self**.type** **=** **type**

self**.**length **=** length

### 5.3.2 文件存储和读取

我们通过\_\_loadfile\_\_(),以及\_\_savefile\_\_()两个函数和将硬盘中的表文件catalog\_index，catalog\_table读入，解析其语法结构，然后写入本地的Table和Column的变量之中，便于操作。

catalog\_table的文件样例如下：

**{**"S"**:** **{**"columns"**:** **{**"ID"**:** **[**true**,** "int"**,** 0**],** "name"**:** **[**true**,** "char"**,** 12**],** "age"**:** **[**false**,** "int"**,** 0**],** "gender"**:** **[**false**,** "char"**,** 1**]},** "pk"**:** 0**},** "test1"**:** **{**"columns"**:** **{**"ID"**:** **[**true**,** "int"**,** 0**],** "name"**:** **[**true**,** "char"**,** 12**],** "gender"**:** **[**false**,** "char"**,** 1**]},** "pk"**:** 0**}}**

读取这样的文件我们借助了python库中的json.loads函数，可以将上述的字符串Table类型的字典变量tables[]之中，相关的函数时\_\_loadfile\_\_()，实现如下：

**def** \_\_loadfile\_\_**():** # from file to memory

f **=** **open(**tablecatalog**)**

json\_tables **=** json**.**loads**(**f**.**read**())**

**for** table **in** json\_tables**.**items**():**

temp\_name **=** table**[**0**]**

temp\_pk **=** table**[**1**][**'pk'**]**

temp\_columns **=** **[]**

\_\_table **=** Table**(**temp\_name**,** temp\_pk**)** # table\_name&primary key

**for** \_\_column **in** table**[**1**][**'columns'**].**items**():**

temp\_attname **=** \_\_column**[**0**]**

temp\_isunique **=** \_\_column**[**1**][**0**]**

temp\_type **=** \_\_column**[**1**][**1**]**

temp\_len **=** \_\_column**[**1**][**2**]**

temp\_columns**.**append**(**Column**(**temp\_attname**,** temp\_isunique**,** temp\_type**,** temp\_len**))**

\_\_table**.**columns **=** temp\_columns

tables**[**temp\_name**]** **=** \_\_table # add into the tables dict in memory

f**.**close**()**

f **=** **open(**indexcatalog**)**

json\_indices **=** f**.**read**()**

json\_indices **=** json**.**loads**(**json\_indices**)**

**for** index **in** json\_indices**.**items**():**

temp\_indexname **=** index**[**0**]** # name of this index

temp\_index **=** index**[**1**]** # the actual component of this index

indices**[**temp\_indexname**]** **=** temp\_index

f**.**close**()**

相关操作完成之后，将tables字典中的文件转存到文件的函数也可以类似实现，如下：

**def** \_\_savefile\_\_**():** # from memory to file

\_\_tables **=** **{}**

**for** items **in** tables**.**items**():**

definition **=** **{}**

temp\_name **=** items**[**0**]**

\_\_columns **=** **{}**

**for** i **in** items**[**1**].**columns**:**

\_\_columns**[**i**.**column\_name**]** **=** **[**i**.**is\_unique**,** i**.type,** i**.**length**]**

definition**[**'columns'**]** **=** \_\_columns

definition**[**'pk'**]** **=** items**[**1**].**primary\_key

\_\_tables**[**temp\_name**]** **=** definition

j\_tables **=** json**.**dumps**(**\_\_tables**)**

j\_indices **=** json**.**dumps**(**indices**)**

f **=** **open(**tablecatalog**,** 'w'**)**

f**.**write**(**j\_tables**)**

f**.**close**()**

f **=** **open(**indexcatalog**,** 'w'**)**

f**.**write**(**j\_indices**)**

f**.**close**()**

### 5.3.3 与硬盘的交互

catalog 处理两个文件，catalog\_index和catalog\_table，即表头信息和索引信息，这些内容存放在电脑的硬盘之中，5.3.2讲述了如何将文件读入为可用的信息，然后这里我们将实现从硬盘中找到文件的功能。

其原理是使用python的库os，根据os.path设定文件路径，然后用open、close函数打开文件读写。这些功能主要在\_\_initialize\_\_()和\_\_finalize\_\_()函数中完成。

内容如下：

**def** \_\_initialize\_\_**(**\_\_path**):** # initialize the file of catalog

**global** catalogpath

**global** tablecatalog

**global** indexcatalog

catalogpath **=** os**.**path**.**join**(**\_\_path**,** 'dbfiles/catalogs'**)**

tablecatalog **=** os**.**path**.**join**(**catalogpath**,** 'catalog\_table'**)**

indexcatalog **=** os**.**path**.**join**(**catalogpath**,** 'catalog\_index'**)**

**if** **not** os**.**path**.**exists**(**catalogpath**):**

os**.**makedirs**(**catalogpath**)**

f1 **=** **open(**tablecatalog**,** 'w'**)**

f2 **=** **open(**indexcatalog**,** 'w'**)**

f1**.**close**()**

f2**.**close**()**

\_\_savefile\_\_**()**

\_\_loadfile\_\_**()**

**def** \_\_finalize\_\_**():**

\_\_savefile\_\_**()**

### 5.3.4 数据操作

如前所述，这里主要处理和表格相关的信息，最基础的是catalog创建表的操作，如下：

**def** create\_table**(**table\_name**,** attributes**,** pk**):**

**global** tables

cur\_table **=** Table**(**table\_name**,** pk**)**

columns **=** **[]**

**for** attr **in** attributes**:**

columns**.**append**(**Column**(**attr**[**'attribute\_name'**],**

attr**[**'unique'**],**

attr**[**'type'**],**

attr**[**'type\_len'**]))**

cur\_table**.**columns **=** columns

seed **=** **False**

**for** index**,** \_\_column **in** **enumerate(**cur\_table**.**columns**):**

**if** \_\_column**.**column\_name **==** cur\_table**.**primary\_key**:**

cur\_table**.**primary\_key **=** index

seed **=** **True**

**break**

**if** seed **==** **False:**

**raise** **Exception(**"primary\_key '%s' does not exist."

**%** cur\_table**.**primary\_key**)**

tables**[**table\_name**]** **=** cur\_table

按照API的实现功能，Catalog还需要检查用户输入的table的名称是否存在或者不存在，以便后续操作，函数实现和上文类似，也是通过遍历tables[]字典实现的。

这一部分模块还负责和检查select的听到的属性是否和数据库的表头相合，相关的检查函数是check\_select\_statement，实现如下：

**def** check\_select\_statement**(**table\_name**,** attributes**,** where**):**

# raise an exception if something is wrong

columns **=** **[]**

**for** i **in** tables**[**table\_name**].**columns**:**

columns**.**append**(**i**.**column\_name**)**

**if** where **is** **not** **None:**

**for** i **in** where**:**

**if** i**[**'l\_op'**]** **not** **in** columns**:**

**raise** **Exception(**"No column"

" name '%s'." **%** i**[**'l\_op'**])**

**if** attributes **==** **[**'\*'**]:**

**return**

**for** i **in** attributes**:**

**if** i **not** **in** columns**:**

**raise** **Exception(**"No column name **(**"No column name '%s'." **%** i**)**

我们还设计了get\_column\_dic函数快速获取一个表的属性列表

**def** get\_column\_dic**(**table\_name**:** **str):**

result **=** **{}**

cnt **=** 0

**global** tables

**for** fullcol **in** tables**[**table\_name**].**columns**:**

colname **=** fullcol**.**column\_name

result**[**colname**]** **=** cnt

cnt **+=** 1

**return** result

## 5.4 Index

Index是通过B+树结构存储索引的模块，通过node变量存储建立B+树，B+树首先从硬盘中的文件indexfile读取，然后再根据API解析出来的命令，进行B+树的搜寻、查找、删除操作：

### 5.4.1数据类型

主要数据类型是node，存储B+树的结点，包含键值、指针、是否为叶结点的判定。

**class** **node():**

**def** \_\_init\_\_**(**self**,** isleaf**,** line0**,** keys0**,** pointers0**,** parent0**=**''**):**

self**.**is\_leaf **=** isleaf

self**.**line **=** line0

self**.**keys **=** keys0

self**.**pointers **=** pointers0

self**.**parent **=** parent0

### 5.4.2文件存储和读取

B+树在文件中的存储和上文中catalog的存储很类似，也是一段长条形的字符串，一层层顺序排列，如下所示。

**{**"S"**:** **{**"is\_leaf"**:** true**,** "line"**:** 0**,** "keys"**:** **[]},** "test1"**:** **{**"is\_leaf"**:** true**,** "line"**:** **[],** "keys"**:** **[]},** "test2"**:** **{**"is\_leaf"**:** true**,** "line"**:** **[],** "keys"**:** **[]}}**

读取时也按照上文所述的json库实现，但是这里根据文件的特性将函数分拆为两部分，首先调用\_\_load\_\_,之后在\_\_load\_\_中调用load\_node，具体如下：

**def** \_\_load\_\_**():**

**global** \_\_last\_leaf\_pointer

**print(**recordpath**)**

f **=** **open(**os**.**path**.**join**(**recordpath**,** 'indexfile'**))**

json\_tables **=** json**.**loads**(**f**.**read**())**

f**.**close**()**

**for** table **in** json\_tables**.**items**():**

temp\_name **=** table**[**0**]**

temp\_content **=** table**[**1**]**

**if** **len(**temp\_content**[**'keys'**])** **==** 0**:**

tables**[**temp\_name**]** **=** node**(True,** 0**,** **[],** **[])**

**continue**

tables**[**temp\_name**]** **=** \

node**(**temp\_content**[**'is\_leaf'**],**temp\_content**[**'line'**],**temp\_content**[**'keys'**],** temp\_content**[**'pointers'**],** ''**)**

**if** tables**[**temp\_name**].**is\_leaf**:**

**continue**

tables**[**temp\_name**].**pointers **=** \

load\_nodes**(**temp\_content**[**'pointers'**],** tables**[**temp\_name**])**

**def** load\_nodes**(**pointer\_list**,** parent**):**

**global** \_\_last\_leaf\_pointer

nodelist **=** **[]**

**for** pointer **in** pointer\_list**:**

**if** pointer**[**'is\_leaf'**]:**

new\_node **=** node**(**pointer**[**'is\_leaf'**],** pointer**[**'line'**],**pointer**[**'keys'**],** pointer**[**'pointers'**],** parent**)**

nodelist**.**append**(**new\_node**)**

**if** \_\_last\_leaf\_pointer **==** ''**:**

\_\_last\_leaf\_pointer **=** new\_node

**else:**

\_\_last\_leaf\_pointer**.**pointers**.**append**(**new\_node**)**

\_\_last\_leaf\_pointer **=** new\_node

**else:**

new\_node **=** node**(**pointer**[**'is\_leaf'**],** pointer**[**'line'**],**pointer**[**'keys'**],** pointer**[**'pointers'**],** parent**)**

nodelist**.**append**(**new\_node**)**

new\_node**.**pointers **=** load\_nodes**(**pointer**[**'pointers'**],** nodelist**[-**1**])**

**return** nodelist

至于存储，则和catalog基本一样：

**def** \_\_store\_\_**():**

**global** recordpath

\_\_tables **=** **{}**

**for** table **in** tables**.**items**():**

\_\_tables**[**table**[**0**]]** **=** recursive\_store\_node**(**table**[**1**])**

f **=** **open(**os**.**path**.**join**(**recordpath**,** 'indexfile'**),** 'w'**)**

json\_tables **=** json**.**dumps**(**\_\_tables**)**

f**.**write**(**json\_tables**)**

f**.**close**()**

### 5.4.3和硬盘的交互

和catalog中的思路一样，我们首先通过os库确定文件在硬盘中的存储路径（path），之后调用load，或者\_\_store\_\_

**def** \_\_initialize\_\_**(**\_\_path**):**

**global** recordpath

recordpath **=** os**.**path**.**join**(**\_\_path**,** 'dbfiles/indices'**)**

**if** **not** os**.**path**.**exists**(**recordpath**):**

os**.**makedirs**(**recordpath**)**

tables**[**'sys'**]** **=** node**(True,** **[],** **[**'key0'**],** **[[**'key0'**,** 'key1'**],** ''**])**

\_\_store\_\_**()**

\_\_load\_\_**()**

**def** \_\_finalize\_\_**():**

\_\_store\_\_**()**

### 5.4.4 B+树操作

B+树的操作大体和课本的描述一致，这里分别展示一下对于B+树insert、find、和delete的代码，注意这里我们构造的B+树是N=4的类型，也就是一颗2-3-4树

#### find方法

通过B+树内部的索引查找相关数值，首先实现find\_leaf\_node，寻找单个叶结点，之后实现find\_leaf\_node\_with\_condition，实现在查找区间，这样就可以实现select函数，根据select的条件获得返回值。

find\_leaf\_node：

**def** find\_leaf\_place**(**table**,** value**):**

# search on primary key

cur\_node **=** tables**[**table**]**

**while** **not** cur\_node**.**is\_leaf**:**

seed **=** **False**

**for** index**,** key **in** **enumerate(**cur\_node**.**keys**):**

**if** key **>** value**:**

cur\_node **=** cur\_node**.**pointers**[**index**]**

seed **=** **True**

**break**

**if** seed **==** **False:**

cur\_node **=** cur\_node**.**pointers**[-**1**]**

**return** cur\_node

find\_leaf\_place\_with\_condition:

**def** find\_leaf\_place\_with\_condition**(**table\_name **,** value**):**

# \_\_primary\_key = CatalogManager.catalog.tables[table].primary\_key

\_\_primary\_key **=** 0

head\_node **=** tables**[**table\_name**]**

first\_leaf\_node **=** head\_node

**while** first\_leaf\_node**.**is\_leaf **!=** **True:**

first\_leaf\_node **=** first\_leaf\_node**.**pointers**[**0**]**

lists **=** **[]**

#op\_list = ['<','<=','>','>=','<>','=']

#if \_\_primary\_key == column and condition != op\_list[4]:

**while** **not** head\_node**.**is\_leaf**:**

seed **=** **False**

**for** index**,** key **in** **enumerate(**head\_node**.**keys**):**

**if** key **>** value**:**

head\_node **=** head\_node**.**pointers**[**index**]**

seed **=** **True**

**break**

**if** seed **==** **False:**

head\_node **=** head\_node**.**pointers**[-**1**]**

**for** pointer **in** head\_node**.**pointers**[**0**:-**1**]:**

**if** pointer**[**0**]** **==** value**:**

lists**.**append**(**head\_node**)**

**return** lists

select\_from\_table

**def** select\_from\_table**(**table\_name**,** attributes**,** where**):**

results **=** **[]**

columns **=** **{}**

**for** i**,** col **in** **enumerate(**Catalog**.**tables**[**table\_name**].**columns**):**

columns**[**col**.**column\_name**]** **=** i

\_\_primary\_key **=** Catalog**.**tables**[**table\_name**].**primary\_key

# \_\_primary\_key = 0

# columns = {'num': 0, 'val': 1}

**if** **len(**tables**[**table\_name**].**keys**)** **==** 0**:**

**pass**

**else:**

**if** where **is** **not** **None:**

nodes **=** find\_leaf\_place\_with\_condition**(**table\_name**,** columns**[**where**[**0**][**'l\_op'**]],** where**[**0**][**'operator'**],** where**[**0**][**'r\_op'**])**

**for** cond **in** where**:**

**if** columns**[**cond**[**'l\_op'**]]** **==** \_\_primary\_key**:**

nodes **=** find\_leaf\_place\_with\_condition**(**table\_name**,** columns**[**cond**[**'l\_op'**]],** cond**[**'operator'**],** cond**[**'r\_op'**])**

**break**

**for** \_\_node **in** nodes**:**

**for** pointer **in** \_\_node**.**pointers**[**0**:-**1**]:**

**if** check\_conditions**(**pointer**,** columns**,** where**):**

results**.**append**(**pointer**)**

**else:**

first\_leaf\_node **=** tables**[**table\_name**]**

**while** first\_leaf\_node**.**is\_leaf **!=** **True:**

first\_leaf\_node **=** first\_leaf\_node**.**pointers**[**0**]**

**while** **True:**

**for** i **in** first\_leaf\_node**.**pointers**[**0**:-**1**]:**

results**.**append**(**i**)**

**if** first\_leaf\_node**.**pointers**[-**1**]** **!=** ''**:**

first\_leaf\_node **=** first\_leaf\_node**.**pointers**[-**1**]**

**else:**

**break**

**if** attributes**[**0**]** **==** '\*'**:**

\_\_columns\_list **=** **list(**columns**.**keys**())**

\_\_columns\_list\_num **=** **list(**columns**.**values**())**

**else:**

\_\_columns\_list **=** **[]**

\_\_columns\_list\_num **=** **[]**

**for** i **in** **range(**0**,len(**attributes**)):**

\_\_columns\_list**.**append**(**attributes**[**i**])**

\_\_columns\_list\_num**.**append**(**columns**(**attributes**[**i**]))**

print\_select**(**\_\_columns\_list**,**\_\_columns\_list\_num**)**

print\_select的实现和API中的类似，这里就不赘述了。

#### insert方法

主函数是insert\_into\_tables，按照B+树插入法，调用find\_leaf\_node，以及

find\_leaf\_place\_with\_condition实现。

主函数insert\_into\_tables:

**def** insert\_into\_table**(**table\_name**,** \_\_values**,**line\_number**:int):**

cur\_node **=** tables**[**table\_name**]**

\_\_primary\_key **=** Catalog**.**tables**[**table\_name**].**primary\_key

# \_\_primary\_key = 0

**if** **len(**cur\_node**.**keys**)** **==** 0**:**

# new tree

cur\_node**.**keys**.**append**(**\_\_values**[**\_\_primary\_key**])**

cur\_node**.**pointers**.**append**([**\_\_values**[**\_\_primary\_key**]]+[**line\_number**])**

cur\_node**.**pointers**.**append**(**''**)**

**print(**'Successfully insert into table %s,' **%** table\_name**,** end**=**''**)**

**return**

cur\_node **=** find\_leaf\_place**(**table\_name**,** \_\_values**[**\_\_primary\_key**])**

**if** **len(**cur\_node**.**keys**)** **<** N **-** 1**:**

insert\_into\_leaf**(**cur\_node**,** \_\_values**[**\_\_primary\_key**],** **[**\_\_values**[**\_\_primary\_key**]]+[**line\_number**])**

**else:**

insert\_into\_leaf**(**cur\_node**,** \_\_values**[**\_\_primary\_key**],** **[**\_\_values**[**\_\_primary\_key**]]+[**line\_number**])**

new\_node **=** node**(True,** **[],[],** **[])**

tmp\_keys **=** cur\_node**.**keys

tmp\_pointers **=** cur\_node**.**pointers

cur\_node**.**keys **=** **[]**

cur\_node**.**pointers **=** **[]**

**for** i **in** **range(**math**.**ceil**(**N **/** 2**)):**

cur\_node**.**keys**.**append**(**tmp\_keys**.**pop**(**0**))**

cur\_node**.**pointers**.**append**(**tmp\_pointers**.**pop**(**0**))**

**for** i **in** **range(**N **-** math**.**ceil**(**N **/** 2**)):**

new\_node**.**keys**.**append**(**tmp\_keys**.**pop**(**0**))**

new\_node**.**pointers**.**append**(**tmp\_pointers**.**pop**(**0**))**

cur\_node**.**pointers**.**append**(**new\_node**)**

new\_node**.**pointers**.**append**(**tmp\_pointers**.**pop**(**0**))**

insert\_into\_parent**(**table\_name**,** cur\_node**,** new\_node**.**keys**[**0**],** new\_node**)**

**print(**'Successfully insert into table %s,' **%** table\_name**,** end**=**''**)**

insert\_into\_leaf，用于在叶结点上插入值

**def** insert\_into\_leaf**(**cur\_node**,** value**,** pointer**):**

**for** index**,** key **in** **enumerate(**cur\_node**.**keys**):**

**if** key **==** value**:**

**raise** **Exception(**"Index Module : primary\_key already exists."**)**

**if** key **>** value**:**

cur\_node**.**pointers**.**insert**(**index**,** pointer**)**

cur\_node**.**keys**.**insert**(**index**,** value**)**

**return**

cur\_node**.**pointers**.**insert**(len(**cur\_node**.**keys**),** pointer**)**

cur\_node**.**keys**.**insert**(len(**cur\_node**.**keys**),** value**)**

insert\_into\_parent,用于在父节点中增加必要的新的值，递归实现

**def** insert\_into\_parent**(**table\_name**,** \_\_node**,** \_\_key**,** new\_node**):**

**if** \_\_node**.**parent **==** ''**:**

cur\_node **=** node**(False,** **[],** **[],[],** ''**)**

cur\_node**.**pointers**.**append**(**\_\_node**)**

cur\_node**.**pointers**.**append**(**new\_node**)**

cur\_node**.**keys**.**append**(**\_\_key**)**

\_\_node**.**parent **=** cur\_node

new\_node**.**parent **=** cur\_node

tables**[**table\_name**]** **=** cur\_node

**else:**

p **=** \_\_node**.**parent

**if** **len(**p**.**pointers**)** **<** N**:**

seed **=** **False**

**for** index**,** key **in** **enumerate(**p**.**keys**):**

**if** \_\_key **<** key**:**

p**.**keys**.**insert**(**index**,** \_\_key**)**

p**.**pointers**.**insert**(**index **+** 1**,** new\_node**)**

seed **=** **True**

**break**

**if** seed **==** **False:**

p**.**keys**.**append**(**\_\_key**)**

p**.**pointers**.**append**(**new\_node**)**

new\_node**.**parent **=** p

**else:**

seed **=** **False**

**for** index**,** key **in** **enumerate(**p**.**keys**):**

**if** \_\_key **<** key**:**

p**.**keys**.**insert**(**index**,** \_\_key**)**

p**.**pointers**.**insert**(**index **+** 1**,** new\_node**)**

seed **=** **True**

**break**

**if** seed **==** **False:**

p**.**keys**.**append**(**\_\_key**)**

p**.**pointers**.**append**(**new\_node**)**

\_\_new\_node **=** node**(False,** **[],** **[],[])**

tmp\_keys **=** p**.**keys

tmp\_pointers **=** p**.**pointers

p**.**keys **=** **[]**

p**.**pointers **=** **[]**

**for** i **in** **range(**math**.**ceil**(**N **/** 2**)):**

p**.**keys**.**append**(**tmp\_keys**.**pop**(**0**))**

p**.**pointers**.**append**(**tmp\_pointers**.**pop**(**0**))**

p**.**pointers**.**append**(**tmp\_pointers**.**pop**(**0**))**

k\_\_ **=** tmp\_keys**.**pop**(**0**)**

**for** i **in** **range(**N **-** math**.**ceil**(**N **/** 2**)** **-** 1**):**

\_\_new\_node**.**keys**.**append**(**tmp\_keys**.**pop**(**0**))**

\_\_tmp **=** tmp\_pointers**.**pop**(**0**)**

\_\_tmp**.**parent **=** \_\_new\_node

\_\_new\_node**.**pointers**.**append**(**\_\_tmp**)**

\_\_tmp **=** tmp\_pointers**.**pop**(**0**)**

\_\_tmp**.**parent **=** \_\_new\_node

\_\_new\_node**.**pointers**.**append**(**\_\_tmp**)**

new\_node**.**parent **=** \_\_new\_node

insert\_into\_parent**(**table\_name**,** p**,** k\_\_**,** \_\_new\_node**)**

#### delete方法

主要分为两步，首先将用户要求delete的值从树中减去，然后通过一个专门的函数maintain\_B\_plus\_tree\_after\_delete来维持删除后B+树的性质。

删除的主函数：

**def** delete\_from\_table**(**table\_name**,** pk**):**

# delete rows from table according to the statement's condition

# usage : find\_leaf\_place\_with\_condition(table, column, condition, value)

#[{'operator': '=', 'l\_op': 'sno', 'r\_op': '88888888'}]

**print(**pk**)**

**print(**"pk printed"**)**

times **=** 0

**for** eachpk **in** pk**:**

#nodes = find\_leaf\_place\_with\_condition(table\_name, columns[where[0]['l\_op']], where[0]['operator'], where[0]['r\_op'])

nodes **=** find\_leaf\_place**(**table\_name**,**eachpk**)**

#if len(nodes) == 0:

# break

seed **=** **False**

#for \_\_node in nodes:

# if seed == True:

# break

\_\_node**=**nodes**;**

**for** index**,** leaf **in** **enumerate(**\_\_node**.**pointers**[**0**:-**1**]):**

#if check\_conditions(leaf, columns, where):

**if(**leaf**[**0**]==**eachpk**):**

\_\_node**.**pointers**.**pop**(**index**)**

\_\_node**.**keys**.**pop**(**index**)**

maintain\_B\_plus\_tree\_after\_delete**(**table\_name**,** \_\_node**)**

times **=** times **+** 1

# seed = True

# break

#if seed == False:

# break

**print(**"Index:Successfully deleted"**)**

删除之后维持B+树的性质

**def** maintain\_B\_plus\_tree\_after\_delete**(**table**,** \_\_node**):**

**global** N

**if** \_\_node**.**parent **==** '' **and** **len(**\_\_node**.**pointers**)** **==** 1**:**

\_\_node**.**pointers **=** **[]**

**elif** **((len(**\_\_node**.**pointers**)** **<** math**.**ceil**(**N **/** 2**)** **and** \_\_node**.**is\_leaf **==** **False)** **or**

**(len(**\_\_node**.**keys**)** **<** math**.**ceil**((**N **-** 1**)** **/** 2**)** **and** \_\_node**.**is\_leaf **==** **True))** \

**and** \_\_node**.**parent **!=** ''**:**

previous **=** **False**

other\_node **=** node**(True,[],** **[],** **[])**

K **=** ''

\_\_index **=** 0

**for** index**,** i **in** **enumerate(**\_\_node**.**parent**.**pointers**):**

**if** i **==** \_\_node**:**

**if** index **==** **len(**\_\_node**.**parent**.**pointers**)** **-** 1**:**

other\_node **=** \_\_node**.**parent**.**pointers**[-**2**]**

previous **=** **True**

K **=** \_\_node**.**parent**.**keys**[**index **-** 1**]**

**else:**

K **=** \_\_node**.**parent**.**keys**[**index**]**

other\_node **=** \_\_node**.**parent**.**pointers**[**index **+** 1**]**

\_\_index **=** index **+** 1

**if** **(**other\_node**.**is\_leaf **==** **True** **and** **len(**other\_node**.**keys**)** **+** **len(**\_\_node**.**keys**)** **<** N**)** **or** \

**(**other\_node**.**is\_leaf **==** **False** **and** **len(**other\_node**.**pointers**)** **+**

**len(**\_\_node**.**pointers**)** **<=** N**):**

**if** previous **==** **True:**

**if** other\_node**.**is\_leaf **==** **False:**

other\_node**.**pointers **=** other\_node**.**pointers **+** \_\_node**.**pointers

other\_node**.**keys **=** other\_node**.**keys **+** **[**K**]** **+** \_\_node**.**keys

**for** \_\_node\_\_ **in** \_\_node**.**pointers**:**

\_\_node\_\_**.**parent **=** other\_node

**else:**

other\_node**.**pointers **=** other\_node**.**pointers**[**0**:-**1**]**

other\_node**.**pointers **=** other\_node**.**pointers **+** \_\_node**.**pointers

other\_node**.**keys **=** other\_node**.**keys **+** \_\_node**.**keys

\_\_node**.**parent**.**pointers **=** \_\_node**.**parent**.**pointers**[**0**:-**1**]**

\_\_node**.**parent**.**keys **=** \_\_node**.**parent**.**keys**[**0**:-**1**]**

maintain\_B\_plus\_tree\_after\_delete**(**table**,** \_\_node**.**parent**)**

**else:**

**if** other\_node**.**is\_leaf **==** **False:**

\_\_node**.**pointers **=** \_\_node**.**pointers **+** other\_node**.**pointers

\_\_node**.**keys **=** \_\_node**.**keys **+** **[**K**]** **+** other\_node**.**keys

**for** \_\_node\_\_ **in** other\_node**.**pointers**:**

\_\_node\_\_**.**parent **=** \_\_node

**else:**

\_\_node**.**pointers **=** \_\_node**.**pointers**[**0**:-**1**]**

\_\_node**.**pointers **=** \_\_node**.**pointers **+** other\_node**.**pointers

\_\_node**.**keys **=** \_\_node**.**keys **+** other\_node**.**keys

\_\_node**.**parent**.**pointers**.**pop**(**\_\_index**)**

\_\_node**.**parent**.**keys**.**pop**(**\_\_index **-** 1**)**

maintain\_B\_plus\_tree\_after\_delete**(**table**,** \_\_node**.**parent**)**

**else:**

**if** previous **==** **True:**

**if** other\_node**.**is\_leaf **==** **True:**

\_\_node**.**keys**.**insert**(**0**,** other\_node**.**keys**.**pop**(-**1**))**

\_\_node**.**pointers**.**insert**(**0**,** other\_node**.**pointers**.**pop**(-**2**))**

\_\_node**.**parent**.**keys**[-**1**]** **=** \_\_node**.**keys**[**0**]**

**else:**

\_\_tmp **=** other\_node**.**pointers**.**pop**(-**1**)**

\_\_tmp**.**parent **=** \_\_node

\_\_node**.**pointers**.**insert**(**0**,** \_\_tmp**)**

\_\_node**.**keys**.**insert**(**0**,** \_\_node**.**parent**.**keys**[-**1**])**

\_\_node**.**parent**.**keys**[-**1**]** **=** other\_node**.**keys**.**pop**(-**1**)**

**else:**

**if** other\_node**.**is\_leaf **==** **True:**

\_\_node**.**keys**.**insert**(-**1**,** other\_node**.**keys**.**pop**(**0**))**

\_\_node**.**pointers**.**insert**(-**2**,** other\_node**.**pointers**.**pop**(**0**))**

\_\_node**.**parent**.**keys**[**\_\_index **-** 1**]** **=** other\_node**.**keys**[**0**]**

**else:**

\_\_tmp **=** other\_node**.**pointers**.**pop**(**0**)**

\_\_tmp**.**parent **=** \_\_node

\_\_node**.**pointers**.**insert**(-**1**,** \_\_tmp**)**

\_\_node**.**keys**.**insert**(-**1**,** \_\_node**.**parent**.**keys**[**\_\_index **-** 1**])**

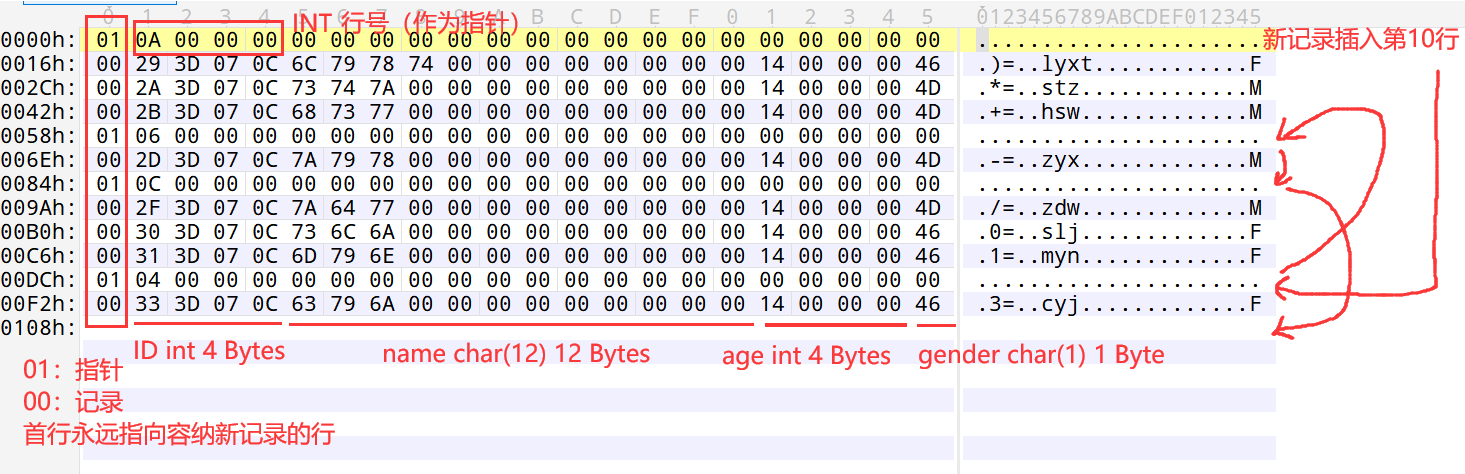
\_\_node**.**parent**.**keys**[**\_\_index **-** 1**]** **=** other\_node**.**keys**.**pop**(**0**)**

## 5.5 Buffer

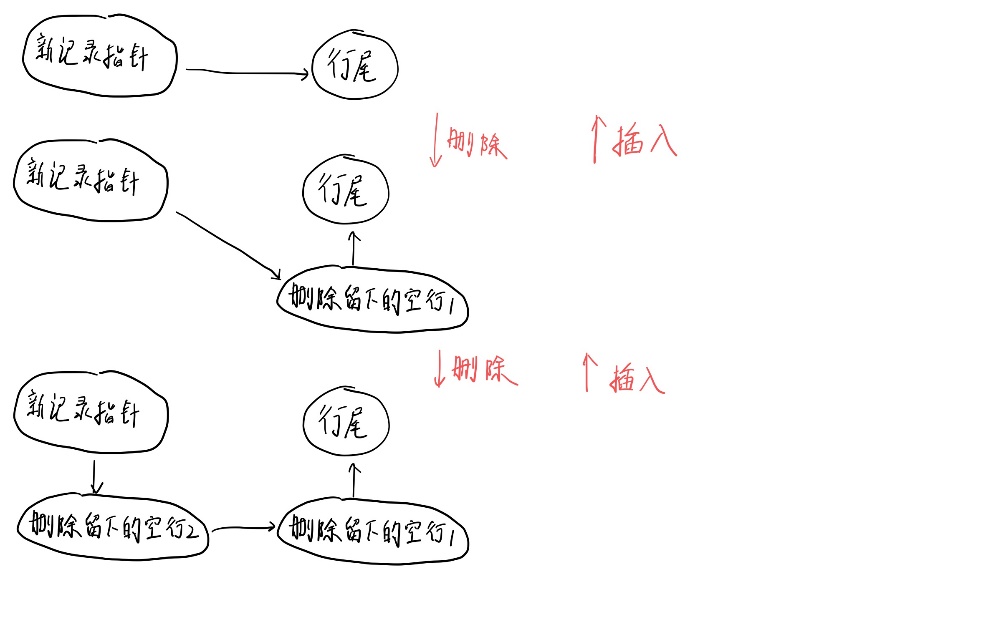
buffer模块通过python的struct包，将记录转换为二进制字节串，直接储存在二进制文件中。这种方法储存的文件密度高，且由于我们储存的是定长记录，通过行数\*每行的字节数可以很方便地找到某条记录的偏移位置，方便按照行数直接建立索引，快速查询记录。

### 5.5.1 记录储存方式

接下来以S(ID int, name char(12) unique, age int, gender char(1), primary key (ID)) 为例展示记录在文件中的储存形式。



首先，在创建表时，我们会根据表中各个属性的大小决定二进制文件中“一行”的字节数。（注意，由于采用二进制文件进行储存，这里的“一行”不是文本文件意义上的一行（如以CRLF作为行的结尾），而是人为划定字节个数的、储存一条记录的“一行”，后文若无特别说明，均为这种含义）。int、float为4字节（且使用小端存储），char按照创建表时规定的字节大小，再加上用来指示本行是记录还是指针的一个字节，即为一行的字节数。若属性所占字节数小于4，也会补成4，以在不储存记录的时候给指针提供4个字节的空间。图中一行的字节数为22。之后，文件的首行固定为指针行，储存容纳新记录的行的行号（为方便，以后称作新记录指针）。

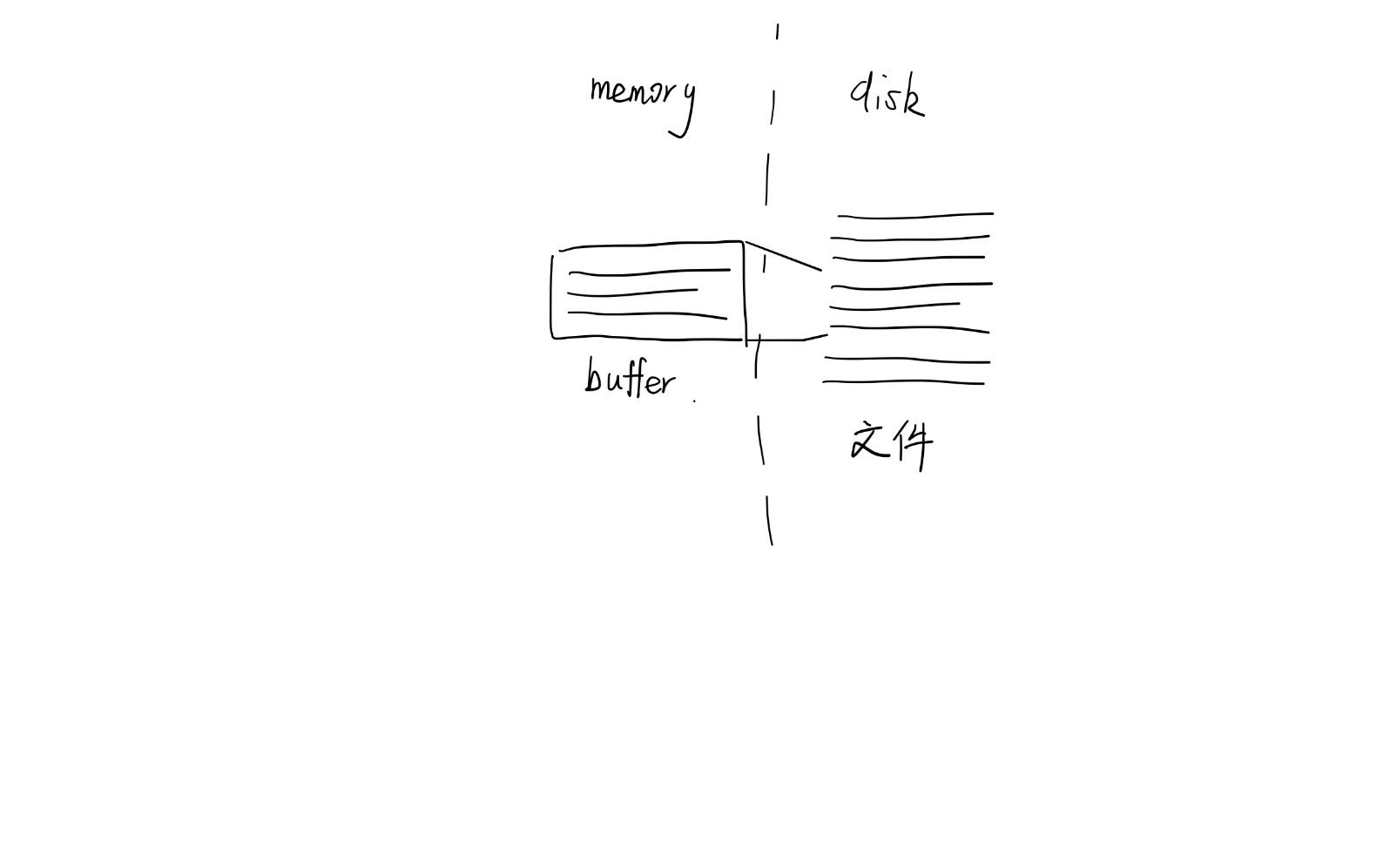


由于我们要按照行数建立索引，因此记录所在的行号是不方便改动的，因此我们采用指针将所有删除的行串起来，并且越晚删除的行越靠近链头。当我们删除记录时，我们将该行清空，首字节变01表示指针行，在2~5个字节中指向当前新记录指针指向的位置，再将新记录指针指向该行。这样当我们插入记录时，可以首先填补空行，而不是一味地在行尾插入，中间留下许多“空洞”。以上的删除也可以连续进行多次，新删除的记录总会变成指针指向老的空行，形成一个链表。当我们插入新记录时，首先按照新记录指针跳到该行，并检查其中内容。如果是文件末尾，那么直接插入即可，同时新记录指针+1。如果不是文件末尾，而是一个指针行，那么我们先将储存的行号转存到新记录指针里（这样下次插入就可以填充链表里的第下个空行了），在存入相应的记录。

简单来说，我们利用删除的记录留下的空间，维护一个所有空行的链表，使得对一些记录的操作不会影响其他记录的行号，同时又能填补穿插在记录之间的空行。而且只通过链头的指针进行操作，保证找到空行只需要常数的时间。

### 5.5.2 buffer与文件的关系

在程序运行的过程中，文件的一部分内容是放在程序内部的结构里的。这些内容存在于内存之中，因而存取较快。下面就将展示buffer是如何设计的。



buffer容纳了文件中连续的几行，因此我们有必要记录buffer从文件的哪行开始，记录了几行，其内容是什么。为了正确地解释从文件中读取的字节，我们还需要记录一行的大小以及一个格式码，用来编码、解码。同时，由于buffer不一定会容纳文件的首行，我们还要单独储存新记录指针，这样在插入新行时不需要将文件指针倒回文件开头查询。再加上指示buffer是否dirty、是否被锁的两个布尔变量，以及buffer所属的表名，就构成了buffer的所有成员。

除了成员，buffer也需要一些操作。构造函数负责按文件最开头几行的内容初始化buffer，按照catalog模块的信息构建格式码、行大小。save函数负责将buffer的改动保存到文件中。adjust负责调整buffer储存的内容（从文件的第几行开始）。同时还要is\_full，line\_is\_inside两个辅助函数，用来返回buffer是否已满、以及某行号对应的行是否储存在buffer之中。

以下是buffer的结构体

**class** **Buffer(object):**

**def** \_\_init\_\_**(**self**,** table\_name**:** **str):**

self**.**table\_name **=** table\_name

self**.**file\_line **=** 0 # the position of buffer's first line in actual file

self**.**is\_dirty **=** **False**

self**.**pin **=** **False**

self**.**format\_list **=** **[**'<c'**]**

self**.**line\_size **=** 1 # we have a flag of 1 byte indicating record (0x00) or pointer (0x01)

# get the format char to support pack and unpack

**for** column **in** Catalog**.**tables**[**table\_name**].**columns**:**

**if** column**.type** **==** 'int'**:**

self**.**format\_list **+=** **[**'i'**]**

self**.**line\_size **+=** 4

**elif** column**.type** **==** 'float'**:**

self**.**format\_list **+=** **[**'f'**]**

self**.**line\_size **+=** 4

**elif** column**.type** **==** 'char'**:**

self**.**format\_list **+=** **[**f'{column**.**length}s'**]**

self**.**line\_size **+=** column**.**length

**if** self**.**line\_size **<** 5**:**

self**.**line\_size **=** 5 # we need 4 extra bytes at least to store the empty line pointer

f **=** **open(**f'dbfiles/table\_files/table\_{self**.**table\_name}.bin'**,** 'rb'**)**

self**.**buf\_size **=** BUFFER\_SIZE # as I mentioned before, it can be adjusted according to the real application

# self.buf\_size = 4096 // self\_size # such as this

self**.**cur\_size **=** BUFFER\_SIZE

self**.**content **=** **[]**

# fill the buffer

**for** i **in** **range(**self**.**buf\_size**):**

line **=** f**.**read**(**self**.**line\_size**)**

**if** line **==** b''**:**

self**.**cur\_size **=** i

**break**

self**.**content**.**append**(**line**)**

f**.**close**()**

self**.**ins\_pos **=** struct**.**unpack**(**'<I'**,** self**.**content**[**0**][**1**:**5**])[**0**]**

**def** adjust**(**self**,** line\_number**):**

**if** self**.**pin**:**

**raise** **Exception(**"Locked buffer is not allowed to replace!"**)**

**if** self**.**is\_dirty**:**

self**.**save**()**

f **=** **open(**f'dbfiles/table\_files/table\_{self**.**table\_name}.bin'**,** 'rb'**)**

# set a different start point

f**.**seek**(**line\_number **\*** self**.**line\_size**)**

self**.**cur\_size **=** BUFFER\_SIZE

self**.**content **=** **[]**

**for** i **in** **range(**self**.**buf\_size**):**

line **=** f**.**read**(**self**.**line\_size**)**

**if** line **==** b''**:**

self**.**cur\_size **=** i

**break**

self**.**content**.**append**(**line**)**

self**.**file\_line **=** line\_number

f**.**close**()**

**def** save**(**self**):**

f **=** **open(**f'dbfiles/table\_files/table\_{self**.**table\_name}.bin'**,** 'rb+'**)**

f**.**seek**(**self**.**file\_line **\*** self**.**line\_size**)**

**for** line **in** self**.**content**:**

f**.**write**(**line**)**

# save the insert position as well

f**.**seek**(**0**)**

f**.**write**(**struct**.**pack**(**f'<cI{self**.**line\_size **-** 5}s'**,** b'\x01'**,** self**.**ins\_pos**,** b'\x00' **\*** **(**self**.**line\_size **-** 5**)))**

f**.**close**()**

self**.**is\_dirty **=** **False**

**def** is\_full**(**self**):**

**return** self**.**cur\_size **==** self**.**buf\_size

**def** line\_is\_inside**(**self**,** n**:** **int):**

**return** self**.**file\_line **<=** n **<** self**.**file\_line **+** self**.**cur\_size

### 5.5.3数据处理操作（insert、delete、find）

有了前面的基础操作，数据处理也就可以方便地实现。例如find\_line函数返回文件的某一行，可以先通过buffer的line\_is\_inside方法检查这一行是否在buffer中，如果在可以直接快速返回。如果不在，就将buffer通过adjust调整到这一行，再进行读取。insert\_record, delete\_line的方法也类似，只不过加入了对buffer记录指针的修改，插入的时候还要检查unique性质是否保持。find\_line, delete\_line功能是按照条件搜索，所以buffer和文件中的记录都要搜索，注意在搜索文件时到buffer已经存储的行号时，就进入buffer搜索，以免读了已被改动的数据，同时保持记录在文件里的存放顺序。其实数据处理操作的细节还有很多，比如dirty置位、编码解码、空buffer插入等等，这里不再赘述。以下是各种操作的代码。

我们先给出find\_line 的代码

**def** find\_line**(**table\_name**:** **str,** line\_number**:** **int):**

**global** buffers

buffer **=** buffers**[**table\_name**]**

# if the line is not in buffer, we need to fetch the page first

**if** **not** buffer**.**line\_is\_inside**(**line\_number**):**

buffer**.**adjust**(**line\_number**)**

**if** **not** buffer**.**line\_is\_inside**(**line\_number**):**

**raise** **Exception(**"The line you want to retrieve exceeds the file."**)**

line **=** buffer**.**content**[**line\_number **-** buffer**.**file\_line**]**

**if** line**[**0**]** **==** 1**:**

**raise** **Exception(**"The line you want to retrieve is not existed."**)**

line **=** decode**(**''**.**join**(**buffer**.**format\_list**),** line**)**

**return** line

注意其中对于line\_is\_inside，adjust的使用。

相关的delete\_line 如下：

**def** delete\_line**(**table\_name**:** **str,** line\_number**:** **int):**

**global** buffers

buffer **=** buffers**[**table\_name**]**

remain **=** struct**.**pack**(**f'<cI{buffer**.**line\_size **-** 5}s'**,** b'\x01'**,** buffer**.**ins\_pos**,** b'\x00' **\*** **(**buffer**.**line\_size **-** 5**))**

# if the line is not in buffer, we need to fetch the page first

**if** **not** buffer**.**line\_is\_inside**(**line\_number**):**

buffer**.**adjust**(**line\_number**)**

**if** **not** buffer**.**line\_is\_inside**(**line\_number**):**

**raise** **Exception(**"The line you want to retrieve exceeds the file."**)**

**if** buffer**.**content**[**line\_number **-** buffer**.**file\_line**][**0**]** **==** 1**:**

**raise** **Exception(**"The line you want to retrieve is not existed."**)**

buffer**.**content**[**line\_number **-** buffer**.**file\_line**]** **=** remain

buffer**.**is\_dirty **=** **True**

buffer**.**ins\_pos **=** line\_number

对于record的处理也就比较简单了，首先在Buffer中查找是否有这样的二进制串，如若没有，进入disk查找。

比如如下的find

**def** find\_record**(**table\_name**:** **str,** columns**:** **dict,** where**:** **list):**

# first find them in buffer

**global** buffers

buffer **=** buffers**[**table\_name**]**

results **=** **[]**

buffer\_range **=** **range(**buffer**.**file\_line**,** buffer**.**file\_line **+** buffer**.**cur\_size**)**

# then find them in file (no fetch)

f **=** **open(**f'dbfiles/table\_files/table\_{table\_name}.bin'**,** 'rb'**)**

f**.**seek**(**buffer**.**line\_size**)**

i **=** 0

**while** 1**:**

i **+=** 1

# skip the line already scanned in the buffer

**if** i **in** buffer\_range**:**

# buffer 里面的查找放在这里 保证记录整体顺序

**for** line **in** buffer**.**content**:**

**if** line**[**0**]** **==** 1**:**

**continue**

line **=** decode**(**''**.**join**(**buffer**.**format\_list**),** line**)**

**if** check**(**line**,** columns**,** where**):**

results **+=** **[**line**]**

i **=** buffer\_range**[-**1**]**

f**.**seek**(**buffer**.**line\_size **\*** **(**i **+** 1**))**

**continue**

line **=** f**.**read**(**buffer**.**line\_size**)**

# reach EOF

**if** line **==** b''**:**

f**.**close**()**

**break**

# a pointer line

**if** line**[**0**]** **==** 1**:**

**continue**

**else:**

line **=** decode**(**''**.**join**(**buffer**.**format\_list**),** line**)**

**if** check**(**line**,** columns**,** where**):**

results **+=** **[**line**]**

**return** results

还有delete 同理：

**def** delete\_record**(**table\_name**:** **str,** column**:** **dict,** where**:** **list):**

pk\_pos **=** Catalog**.**tables**[**table\_name**].**primary\_key

# first search in the buffer

**global** buffers

buffer **=** buffers**[**table\_name**]**

buffer\_range **=** **range(**buffer**.**file\_line**,** buffer**.**file\_line **+** buffer**.**cur\_size**)**

deleted\_pks **=** **[]**

**for** i**,** line **in** **enumerate(**buffer**.**content**):**

**if** line**[**0**]** **==** 1**:**

**continue**

line **=** decode**(**''**.**join**(**buffer**.**format\_list**),** line**)**

**if** check**(**line**,** column**,** where**):**

deleted\_pks **+=** **[**line**[**pk\_pos**]]**

buffer**.**content**[**i**]** **=** \

struct**.**pack**(**f'<cI{buffer**.**line\_size **-** 5}s'**,**

b'\x01'**,** buffer**.**ins\_pos**,**

b'\x00' **\*** **(**buffer**.**line\_size **-** 5**))**

buffer**.**ins\_pos **=** buffer**.**file\_line **+** i

buffer**.**is\_dirty **=** **True**

# then search in the file (no fetch)

f **=** **open(**f'dbfiles/table\_files/table\_{table\_name}.bin'**,** 'rb+'**)**

f**.**seek**(**buffer**.**line\_size**)**

i **=** 0

**while** 1**:**

i **+=** 1

**if** i **in** buffer\_range**:**

i **=** buffer\_range**[-**1**]**

f**.**seek**(**buffer**.**line\_size **\*** **(**i **+** 1**))**

**continue**

line **=** f**.**read**(**buffer**.**line\_size**)**

**if** line **==** b''**:**

f**.**close**()**

**break**

**if** line**[**0**]** **==** 1**:**

**continue**

**else:**

line **=** decode**(**''**.**join**(**buffer**.**format\_list**),** line**)**

**if** check**(**line**,** column**,** where**):**

deleted\_pks **+=** **[**line**[**pk\_pos**]]**

f**.**seek**(**buffer**.**line\_size **\*** i**)**

f**.**write**(**

struct**.**pack**(**f'<cI{buffer**.**line\_size **-** 5}s'**,**

b'\x01'**,**

buffer**.**ins\_pos**,**

b'\x00' **\*** **(**buffer**.**line\_size **-** 5**)))**

buffer**.**ins\_pos **=** i

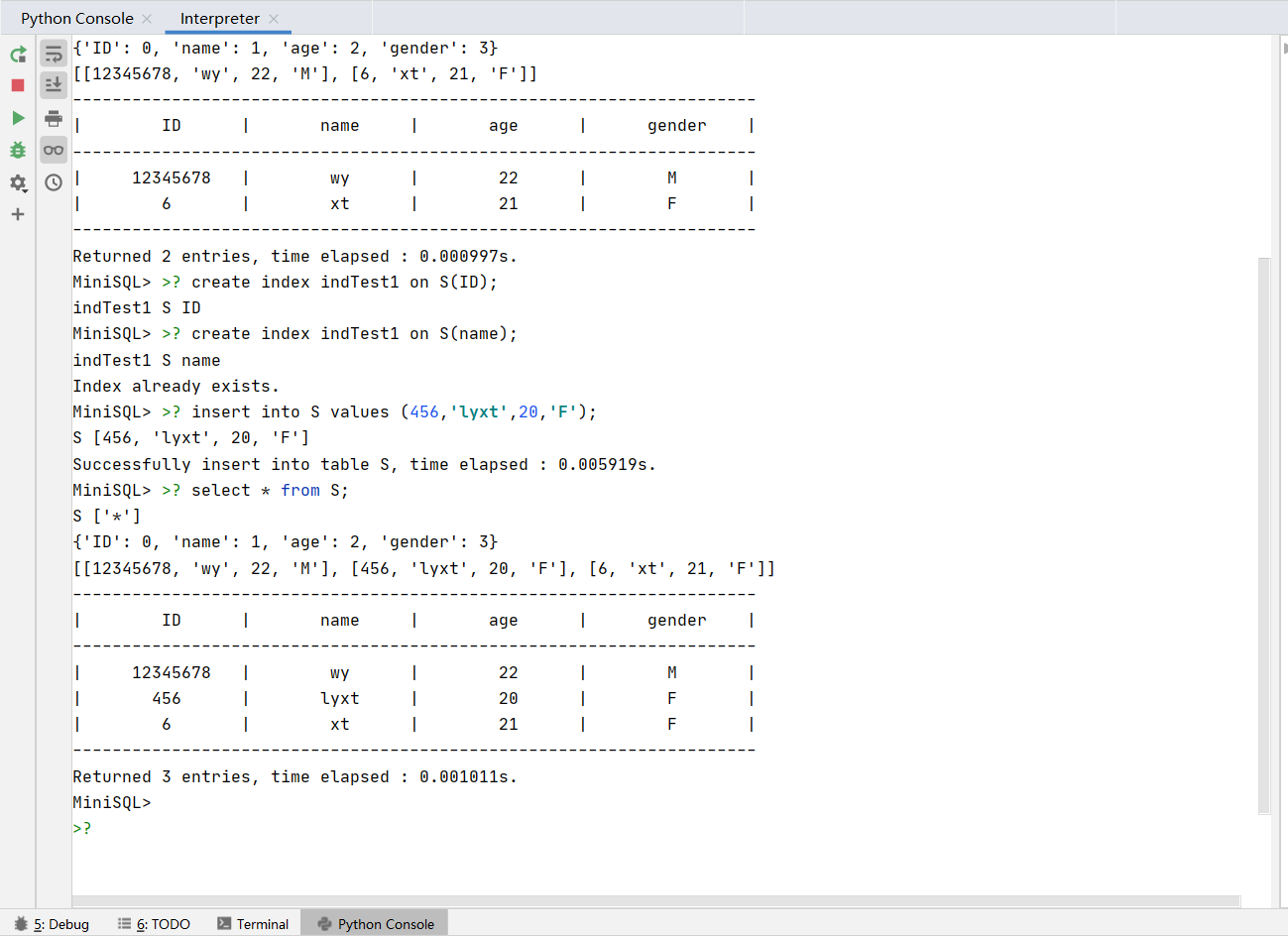
**return** deleted\_pks

# when you doing conditional delete, first call Buffer.delete\_record,

# it will tell you pks of the records which are deleted

# 界面说明

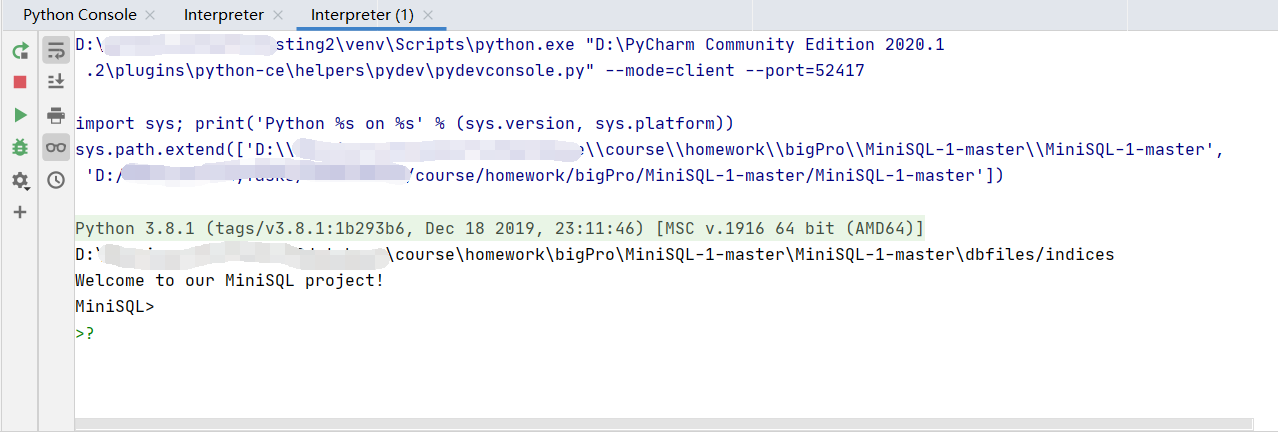
界面使用的是win10内置的Linux终端界面，Apple可以直接在vim中运行，使用vim的界面

如果是使用pycharm，则可以调用pycharm内部的python Console终端，显示效果如下：

# 系统测试

首先进行完备性测试，我们以一个test1的表作为实验体。

### 开始运行

 在pycharm中的interpreter文件中点击运行即可开始

### 表格创建 create

创建成功，注意我们将其catalog\_table 的内容和创建用时反馈了出来。

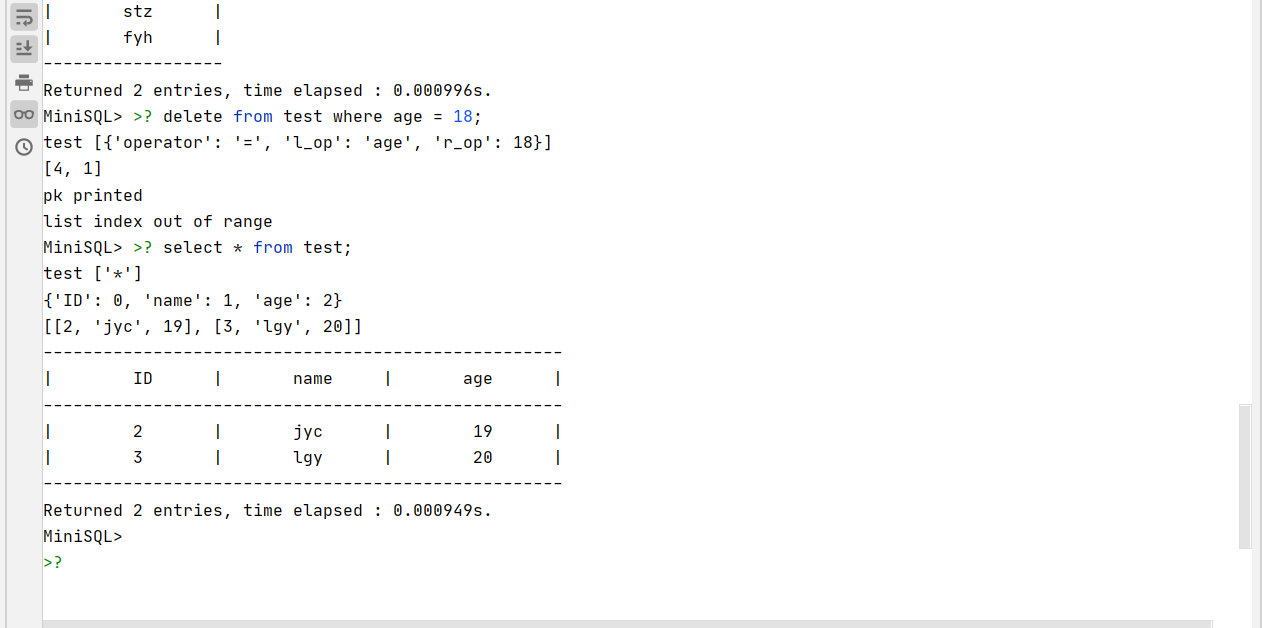
### 插入记录insert

如上所述，这里总共成功插入了四条记录，最后三条是错误示范，分别违背了unique唯一性原则，和插入数值和表头对应的原则。

### 搜索记录 select

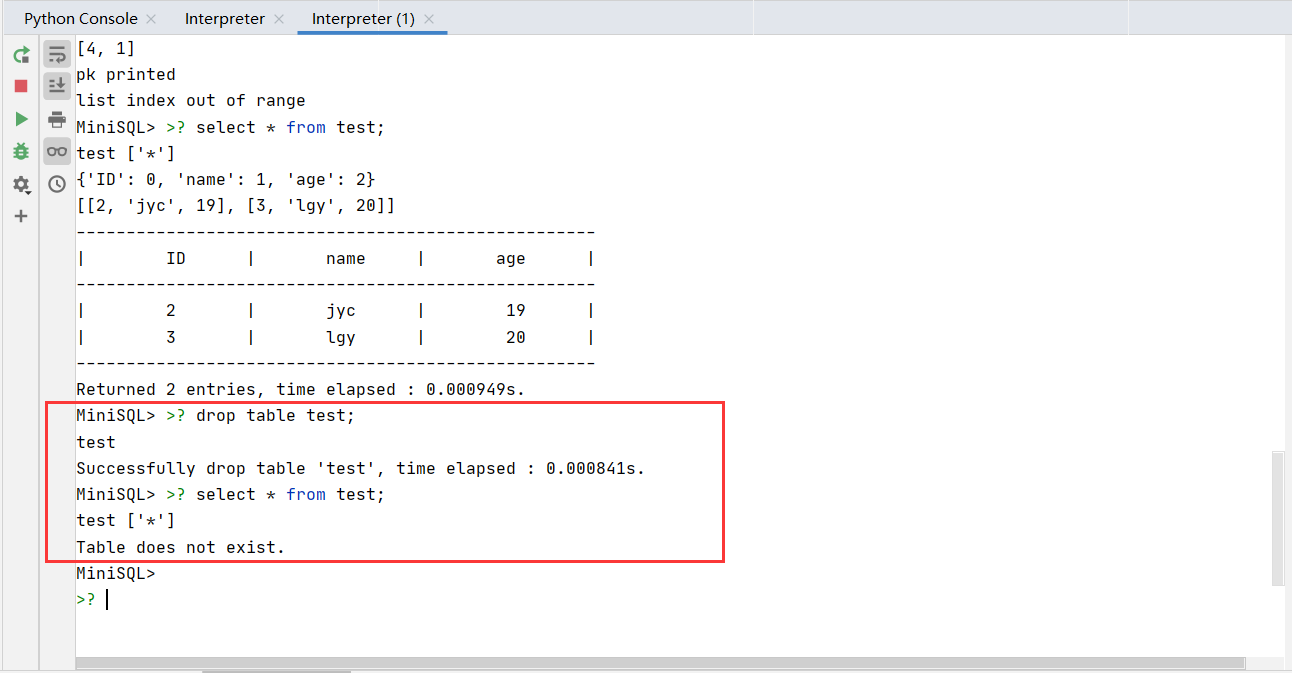
如图测试了select \* ，select 。。。where。。。的语法，均可以正常返回。

### 删除记录 delete

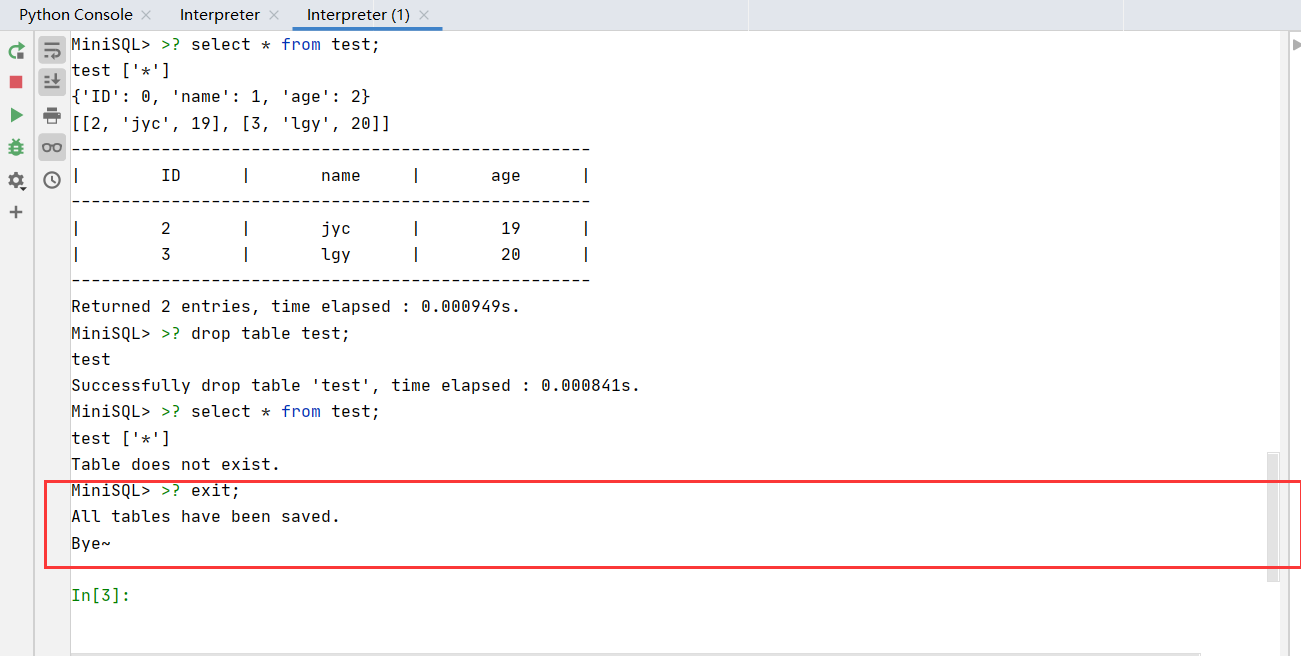


测试了delete 和select的结果，可以返回正常的结果。

### 删除表格和退出程序



关于drop的命令正常。



我们用exit或quit实现退出。

### 脚本文件

**drop** **table** S**;**

**create** **table** S**(**ID int**,** name char**(**12**)** **unique,** age int**,** gender char**(**1**),primary** **key** **(**ID**));**

# skip this line

**insert** **into** S **values(**1**,**'stz'**,**20**,**'M'**);**

**insert** **into** S **values(**2**,**'jyc'**,**19**,**'M'**);**

**insert** **into** S **values(**3**,**'lgy'**,**20**,**'M'**);**

**insert** **into** S **values(**4**,**'fyh'**,**19**,**'M'**);**

**insert** **into** S **values(**5**,**'homura'**,**500**,**'F'**);**

**insert** **into** S **values(**6**,**'Motoka'**,**600**,**'F'**);**

**insert** **into** S **values(**7**,**'AAB'**,**19**,**'F'**);**

**insert** **into** S **values(**8**,**'AAC'**,**17**,**'F'**);**

**insert** **into** S **values(**9**,**'AAD'**,**23**,**'F'**);**

**insert** **into** S **values(**10**,**'AAE'**,**13**,**'F'**);**

**insert** **into** S **values(**11**,**'AAF'**,**10**,**'M'**);**

**insert** **into** S **values(**12**,**'AAG'**,**10**,**'F'**);**

**insert** **into** S **values(**13**,**'AAH'**,**12**,**'M'**);**

**insert** **into** S **values(**14**,**'AAI'**,**13**,**'M'**);**

**insert** **into** S **values(**15**,**'AAJ'**,**25**,**'F'**);**

**insert** **into** S **values(**16**,**'AAK'**,**22**,**'M'**);**

**insert** **into** S **values(**17**,**'AAL'**,**18**,**'M'**);**

**insert** **into** S **values(**18**,**'AAM'**,**20**,**'M'**);**

**insert** **into** S **values(**19**,**'AAN'**,**24**,**'F'**);**

**insert** **into** S **values(**20**,**'AAO'**,**30**,**'M'**);**

**insert** **into** S **values(**21**,**'AAP'**,**25**,**'F'**);**

**insert** **into** S **values(**22**,**'AAQ'**,**13**,**'F'**);**

**insert** **into** S **values(**23**,**'AAR'**,**18**,**'M'**);**

**insert** **into** S **values(**24**,**'AAS'**,**30**,**'M'**);**

**insert** **into** S **values(**25**,**'AAT'**,**10**,**'M'**);**

**insert** **into** S **values(**26**,**'AAU'**,**15**,**'F'**);**

**insert** **into** S **values(**27**,**'AAV'**,**23**,**'M'**);**

**insert** **into** S **values(**28**,**'AAW'**,**10**,**'F'**);**

**insert** **into** S **values(**29**,**'AAX'**,**17**,**'M'**);**

**insert** **into** S **values(**30**,**'AAY'**,**11**,**'M'**);**

**insert** **into** S **values(**31**,**'AAZ'**,**12**,**'F'**);**

**insert** **into** S **values(**32**,**'ABA'**,**11**,**'F'**);**

**insert** **into** S **values(**33**,**'ABB'**,**24**,**'F'**);**

**insert** **into** S **values(**34**,**'ABC'**,**29**,**'M'**);**

**insert** **into** S **values(**35**,**'ABD'**,**25**,**'M'**);**

**insert** **into** S **values(**36**,**'ABE'**,**13**,**'F'**);**

**insert** **into** S **values(**37**,**'ABF'**,**12**,**'M'**);**

**insert** **into** S **values(**38**,**'ABG'**,**30**,**'F'**);**

**insert** **into** S **values(**39**,**'ABH'**,**10**,**'M'**);**

**insert** **into** S **values(**40**,**'ABI'**,**27**,**'M'**);**

**insert** **into** S **values(**41**,**'ABJ'**,**23**,**'F'**);**

**insert** **into** S **values(**42**,**'ABK'**,**14**,**'M'**);**

**insert** **into** S **values(**43**,**'ABL'**,**20**,**'F'**);**

**insert** **into** S **values(**44**,**'ABM'**,**23**,**'M'**);**

**insert** **into** S **values(**45**,**'ABN'**,**17**,**'M'**);**

**insert** **into** S **values(**46**,**'ABO'**,**12**,**'M'**);**

**insert** **into** S **values(**47**,**'ABP'**,**13**,**'M'**);**

**insert** **into** S **values(**48**,**'ABQ'**,**23**,**'M'**);**

**insert** **into** S **values(**49**,**'ABR'**,**10**,**'F'**);**

**insert** **into** S **values(**50**,**'ABS'**,**28**,**'F'**);**

**insert** **into** S **values(**51**,**'ABT'**,**16**,**'F'**);**

**insert** **into** S **values(**52**,**'ABU'**,**12**,**'M'**);**

**insert** **into** S **values(**53**,**'ABV'**,**29**,**'M'**);**

**insert** **into** S **values(**54**,**'ABW'**,**15**,**'M'**);**

**insert** **into** S **values(**55**,**'ABX'**,**25**,**'M'**);**

**insert** **into** S **values(**56**,**'ABY'**,**11**,**'M'**);**

**insert** **into** S **values(**57**,**'ABZ'**,**12**,**'F'**);**

**insert** **into** S **values(**58**,**'ACA'**,**28**,**'F'**);**

**insert** **into** S **values(**59**,**'ACB'**,**23**,**'M'**);**

**insert** **into** S **values(**60**,**'ACC'**,**17**,**'M'**);**

**insert** **into** S **values(**61**,**'ACD'**,**12**,**'F'**);**

**insert** **into** S **values(**62**,**'ACE'**,**20**,**'F'**);**

**insert** **into** S **values(**63**,**'ACF'**,**21**,**'F'**);**

**insert** **into** S **values(**64**,**'ACG'**,**24**,**'F'**);**

**insert** **into** S **values(**65**,**'ACH'**,**24**,**'M'**);**

**insert** **into** S **values(**66**,**'ACI'**,**15**,**'M'**);**

**insert** **into** S **values(**67**,**'ACJ'**,**13**,**'F'**);**

**insert** **into** S **values(**68**,**'ACK'**,**15**,**'M'**);**

**insert** **into** S **values(**69**,**'ACL'**,**15**,**'M'**);**

**insert** **into** S **values(**70**,**'ACM'**,**14**,**'M'**);**

**insert** **into** S **values(**71**,**'ACN'**,**25**,**'M'**);**

**insert** **into** S **values(**72**,**'ACO'**,**27**,**'M'**);**

**insert** **into** S **values(**73**,**'ACP'**,**15**,**'F'**);**

**insert** **into** S **values(**74**,**'ACQ'**,**23**,**'F'**);**

**insert** **into** S **values(**75**,**'ACR'**,**19**,**'M'**);**

**insert** **into** S **values(**76**,**'ACS'**,**17**,**'M'**);**

**insert** **into** S **values(**77**,**'ACT'**,**27**,**'M'**);**

**insert** **into** S **values(**78**,**'ACU'**,**29**,**'F'**);**

**insert** **into** S **values(**79**,**'ACV'**,**22**,**'F'**);**

**insert** **into** S **values(**80**,**'ACW'**,**11**,**'F'**);**

**insert** **into** S **values(**81**,**'ACX'**,**21**,**'F'**);**

**insert** **into** S **values(**82**,**'ACY'**,**20**,**'F'**);**

**insert** **into** S **values(**83**,**'ACZ'**,**18**,**'F'**);**

**insert** **into** S **values(**84**,**'ADA'**,**12**,**'F'**);**

**insert** **into** S **values(**85**,**'ADB'**,**18**,**'M'**);**

**insert** **into** S **values(**86**,**'ADC'**,**26**,**'F'**);**

**insert** **into** S **values(**87**,**'ADD'**,**15**,**'F'**);**

**insert** **into** S **values(**88**,**'ADE'**,**25**,**'F'**);**

**insert** **into** S **values(**89**,**'ADF'**,**17**,**'F'**);**

**insert** **into** S **values(**90**,**'ADG'**,**22**,**'M'**);**

**insert** **into** S **values(**91**,**'ADH'**,**16**,**'F'**);**

**insert** **into** S **values(**92**,**'ADI'**,**14**,**'M'**);**

**insert** **into** S **values(**93**,**'ADJ'**,**29**,**'M'**);**

**insert** **into** S **values(**94**,**'ADK'**,**27**,**'M'**);**

**insert** **into** S **values(**95**,**'ADL'**,**26**,**'F'**);**

**insert** **into** S **values(**96**,**'ADM'**,**15**,**'M'**);**

**insert** **into** S **values(**97**,**'ADN'**,**14**,**'F'**);**

**insert** **into** S **values(**98**,**'ADO'**,**13**,**'M'**);**

**insert** **into** S **values(**99**,**'ADP'**,**20**,**'M'**);**

**insert** **into** S **values(**100**,**'ADQ'**,**28**,**'M'**);**

**insert** **into** S **values(**101**,**'ADR'**,**11**,**'F'**);**

**insert** **into** S **values(**102**,**'ADS'**,**16**,**'F'**);**

**insert** **into** S **values(**103**,**'ADT'**,**24**,**'F'**);**

**insert** **into** S **values(**104**,**'ADU'**,**11**,**'M'**);**

**insert** **into** S **values(**105**,**'ADV'**,**16**,**'M'**);**

**insert** **into** S **values(**106**,**'ADW'**,**28**,**'F'**);**

**insert** **into** S **values(**107**,**'ADX'**,**13**,**'M'**);**

**insert** **into** S **values(**108**,**'ADY'**,**23**,**'F'**);**

**insert** **into** S **values(**109**,**'ADZ'**,**22**,**'F'**);**

**insert** **into** S **values(**110**,**'AEA'**,**24**,**'F'**);**

**insert** **into** S **values(**111**,**'AEB'**,**23**,**'M'**);**

**insert** **into** S **values(**112**,**'AEC'**,**28**,**'M'**);**

**insert** **into** S **values(**113**,**'AED'**,**25**,**'M'**);**

**insert** **into** S **values(**114**,**'AEE'**,**17**,**'M'**);**

**insert** **into** S **values(**115**,**'AEF'**,**12**,**'F'**);**

**insert** **into** S **values(**116**,**'AEG'**,**11**,**'F'**);**

**insert** **into** S **values(**117**,**'AEH'**,**13**,**'F'**);**

**insert** **into** S **values(**118**,**'AEI'**,**12**,**'M'**);**

**insert** **into** S **values(**119**,**'AEJ'**,**18**,**'F'**);**

**insert** **into** S **values(**120**,**'AEK'**,**19**,**'F'**);**

**insert** **into** S **values(**121**,**'AEL'**,**19**,**'F'**);**

**insert** **into** S **values(**122**,**'AEM'**,**14**,**'M'**);**

**insert** **into** S **values(**123**,**'AEN'**,**23**,**'F'**);**

**insert** **into** S **values(**124**,**'AEO'**,**22**,**'F'**);**

**insert** **into** S **values(**125**,**'AEP'**,**27**,**'F'**);**

**insert** **into** S **values(**126**,**'AEQ'**,**27**,**'F'**);**

**insert** **into** S **values(**127**,**'AER'**,**12**,**'M'**);**

**insert** **into** S **values(**128**,**'AES'**,**27**,**'F'**);**

**insert** **into** S **values(**129**,**'AET'**,**24**,**'M'**);**

**insert** **into** S **values(**130**,**'AEU'**,**30**,**'F'**);**

**insert** **into** S **values(**131**,**'AEV'**,**29**,**'F'**);**

**insert** **into** S **values(**132**,**'AEW'**,**18**,**'M'**);**

**insert** **into** S **values(**133**,**'AEX'**,**13**,**'F'**);**

**insert** **into** S **values(**134**,**'AEY'**,**16**,**'M'**);**

**insert** **into** S **values(**135**,**'AEZ'**,**18**,**'M'**);**

**insert** **into** S **values(**136**,**'AFA'**,**24**,**'M'**);**

**insert** **into** S **values(**137**,**'AFB'**,**10**,**'F'**);**

**insert** **into** S **values(**138**,**'AFC'**,**23**,**'F'**);**

**insert** **into** S **values(**139**,**'AFD'**,**28**,**'M'**);**

**insert** **into** S **values(**140**,**'AFE'**,**13**,**'M'**);**

**insert** **into** S **values(**141**,**'AFF'**,**20**,**'M'**);**

**insert** **into** S **values(**142**,**'AFG'**,**16**,**'M'**);**

**insert** **into** S **values(**143**,**'AFH'**,**19**,**'M'**);**

**insert** **into** S **values(**144**,**'AFI'**,**25**,**'F'**);**

**insert** **into** S **values(**145**,**'AFJ'**,**11**,**'M'**);**

**insert** **into** S **values(**146**,**'AFK'**,**20**,**'M'**);**

**insert** **into** S **values(**147**,**'AFL'**,**16**,**'M'**);**

**insert** **into** S **values(**148**,**'AFM'**,**26**,**'M'**);**

**insert** **into** S **values(**149**,**'AFN'**,**13**,**'F'**);**

**insert** **into** S **values(**150**,**'AFO'**,**25**,**'M'**);**

**insert** **into** S **values(**151**,**'AFP'**,**24**,**'M'**);**

**insert** **into** S **values(**152**,**'AFQ'**,**13**,**'M'**);**

**insert** **into** S **values(**153**,**'AFR'**,**19**,**'F'**);**

**insert** **into** S **values(**154**,**'AFS'**,**23**,**'M'**);**

**insert** **into** S **values(**155**,**'AFT'**,**29**,**'M'**);**

**insert** **into** S **values(**156**,**'AFU'**,**19**,**'F'**);**

**insert** **into** S **values(**157**,**'AFV'**,**15**,**'M'**);**

**insert** **into** S **values(**158**,**'AFW'**,**12**,**'F'**);**

**insert** **into** S **values(**159**,**'AFX'**,**15**,**'F'**);**

**insert** **into** S **values(**160**,**'AFY'**,**25**,**'M'**);**

**insert** **into** S **values(**161**,**'AFZ'**,**18**,**'M'**);**

**insert** **into** S **values(**162**,**'AGA'**,**25**,**'M'**);**

**insert** **into** S **values(**163**,**'AGB'**,**20**,**'F'**);**

**insert** **into** S **values(**164**,**'AGC'**,**24**,**'M'**);**

**insert** **into** S **values(**165**,**'AGD'**,**21**,**'M'**);**

**insert** **into** S **values(**166**,**'AGE'**,**20**,**'M'**);**

**insert** **into** S **values(**167**,**'AGF'**,**18**,**'M'**);**

**insert** **into** S **values(**168**,**'AGG'**,**24**,**'M'**);**

**insert** **into** S **values(**169**,**'AGH'**,**10**,**'M'**);**

**insert** **into** S **values(**170**,**'AGI'**,**23**,**'F'**);**

**insert** **into** S **values(**171**,**'AGJ'**,**14**,**'M'**);**

**insert** **into** S **values(**172**,**'AGK'**,**23**,**'M'**);**

**insert** **into** S **values(**173**,**'AGL'**,**18**,**'M'**);**

**insert** **into** S **values(**174**,**'AGM'**,**16**,**'M'**);**

**insert** **into** S **values(**175**,**'AGN'**,**22**,**'M'**);**

**insert** **into** S **values(**176**,**'AGO'**,**21**,**'F'**);**

**insert** **into** S **values(**177**,**'AGP'**,**15**,**'F'**);**

**insert** **into** S **values(**178**,**'AGQ'**,**21**,**'F'**);**

**insert** **into** S **values(**179**,**'AGR'**,**11**,**'M'**);**

**insert** **into** S **values(**180**,**'AGS'**,**21**,**'M'**);**

**insert** **into** S **values(**181**,**'AGT'**,**28**,**'F'**);**

**insert** **into** S **values(**182**,**'AGU'**,**14**,**'M'**);**

**insert** **into** S **values(**183**,**'AGV'**,**30**,**'M'**);**

**insert** **into** S **values(**184**,**'AGW'**,**21**,**'M'**);**

**insert** **into** S **values(**185**,**'AGX'**,**20**,**'F'**);**

**insert** **into** S **values(**186**,**'AGY'**,**28**,**'F'**);**

**insert** **into** S **values(**187**,**'AGZ'**,**22**,**'M'**);**

**insert** **into** S **values(**188**,**'AHA'**,**30**,**'M'**);**

**insert** **into** S **values(**189**,**'AHB'**,**30**,**'M'**);**

**insert** **into** S **values(**190**,**'AHC'**,**15**,**'F'**);**

**insert** **into** S **values(**191**,**'AHD'**,**18**,**'M'**);**

**insert** **into** S **values(**192**,**'AHE'**,**11**,**'M'**);**

**insert** **into** S **values(**193**,**'AHF'**,**11**,**'F'**);**

**insert** **into** S **values(**194**,**'AHG'**,**25**,**'M'**);**

**insert** **into** S **values(**195**,**'AHH'**,**29**,**'F'**);**

**insert** **into** S **values(**196**,**'AHI'**,**14**,**'F'**);**

**insert** **into** S **values(**197**,**'AHJ'**,**21**,**'F'**);**

**insert** **into** S **values(**198**,**'AHK'**,**19**,**'M'**);**

**insert** **into** S **values(**199**,**'AHL'**,**28**,**'F'**);**

**insert** **into** S **values(**200**,**'AHM'**,**23**,**'F'**);**

#

**create** **index** i\_name **on** S **(**name**);**

#

**select** **\*** **from** S**;**

**select** name **from** S **where** age **<** 20 **and** age **>** 10**;**

**select** ID **from** S **where** gender **=** 'F'**;**

#

**delete** **from** S **where** age **<** 18**;**

**delete** **from** S **where** ID **=** 13**;**

#

**select** **\*** **from** S**;**

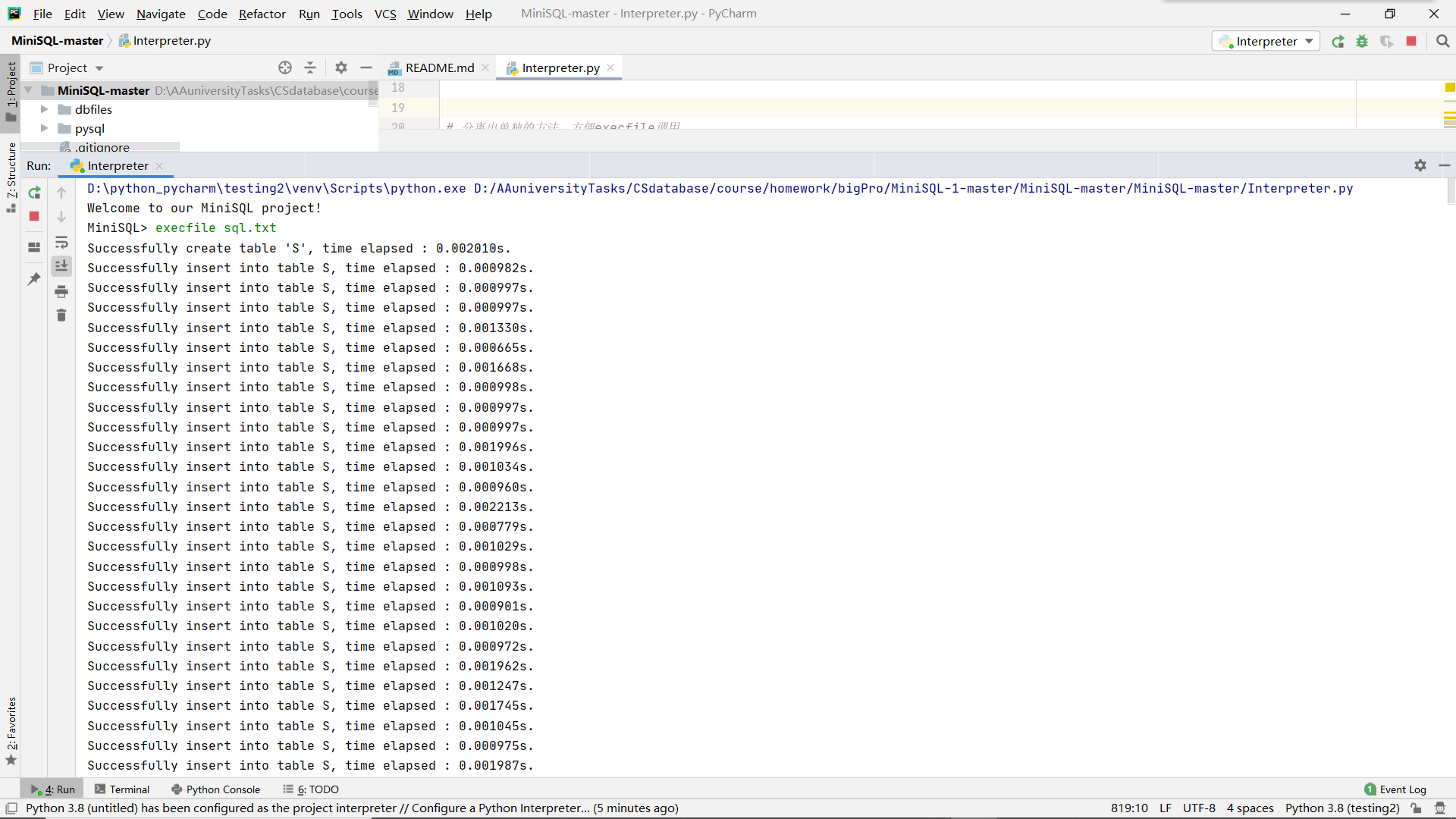
**select** name **from** S **where** age **<** 20 **and** age **>** 10**;**

**select** ID **from** S **where** gender **=** 'F'**;**

#

**drop** **table** S**;**

### 测试结果



如图所示，所有的记录都可以在千分之秒内完成

最后的完成用时不超过1秒钟。

对于学生的信息查找是正确的。

详见我们的视频。