Interpreter

一. 模块概述

Interpreter 模块的基本功能是将用户输入的命令进行语法分析和语义解析，得到需要的命令参数，将命令参数封装成对应命令的参数类对象传给 API 执行相应的操作。同时对于不能识别的命令向用户提供错误信息。

二. 主要功能和对外提供的接口

1. select
   - 需要查询的表的名字 table_name
   - 需要查询的属性名字 attribute_name
   - 需要查询的属性条件 condition
   - 查询的属性条件的“=”等操作符 operand

2. insert
   - 插入的表的名字 table_name
   - 插入条目的信息 vector data
   - 插入条目的信息对应的 type datatype

3. delete
   - 要删除的表名 table_name
   - 需要删除的属性名字 attribute_name
   - 需要删除的属性条件 condition
   - 删除的属性条件的“=”等操作符 operand

4. drop table
   - 要删除的表的名字 table_name

5. drop index
   - 要删除的 index 的名字 index_name
   - 要删除的 index 所在的表名 table_name

6. create table

- 要创建的表的名字 table_name
- 要创建的表包含的属性的 vector a
- 要创建的表的主键 primary

7. create index

- 需要创建索引的表的名字 table_name
- 需要创建索引的属性的名字 attribute_name
- 要创建的索引的名字 index_name

8. execute file

不对外提供接口

读取用户提供的文件并执行其中的语句

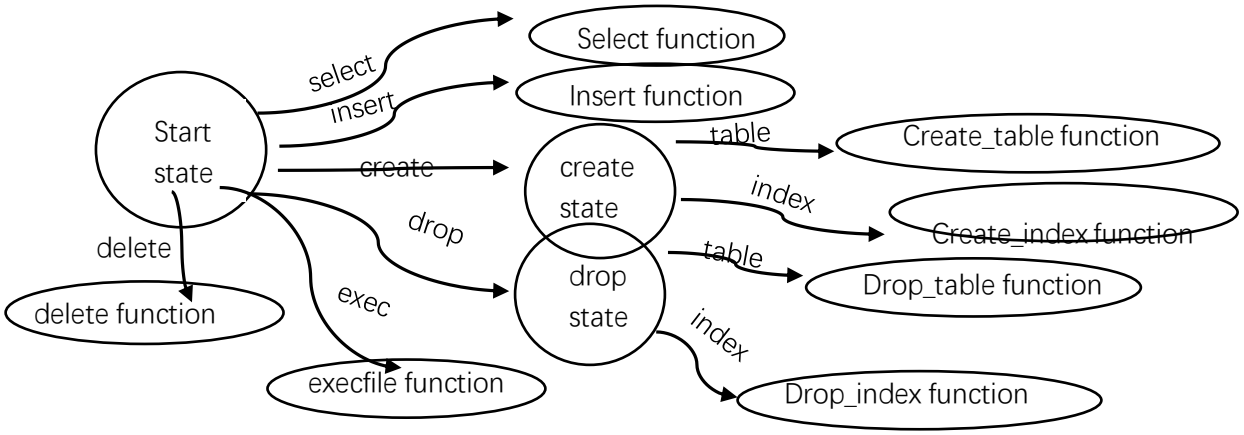
三. 设计思路

1.读取用户输入的一条语句，以“;”字符作为一条语句结束的标记。将语句解 析成为一个有意字串的 vector（具体实现名称为 tokens)。例如下列语句

```
create table tablename
  (
      variblename1 type1,
      variblename2 type2
  );
```

“create ”,“table”,“tablename”,“(“,“variblename1”,“type1”等都是一个有意字串.

2.对有意字串进行分析处理,状态图如下



四. 关键函数和代码

由于函数大多都是用状态机的思想实现的，本节解释主要由状态图体现。

1. `int getTokens(const char* sql, vector<string>& tokens, vector<TOKEN>& type)`

负责将 sql 语句解析成有意字串 tokens 和它们对应的 type。

type 主要类型有：

INVALID: 无效字符；

IDLE: 空闲状态，等待读取下一个字串；

END: 语句读取结束状态，以“;”为跳转标志；

IDENTIFIER: 有实际意义的名字或单词，如“create”，“table”等；

NUMBER: 数字类型，包括整数和浮点数；

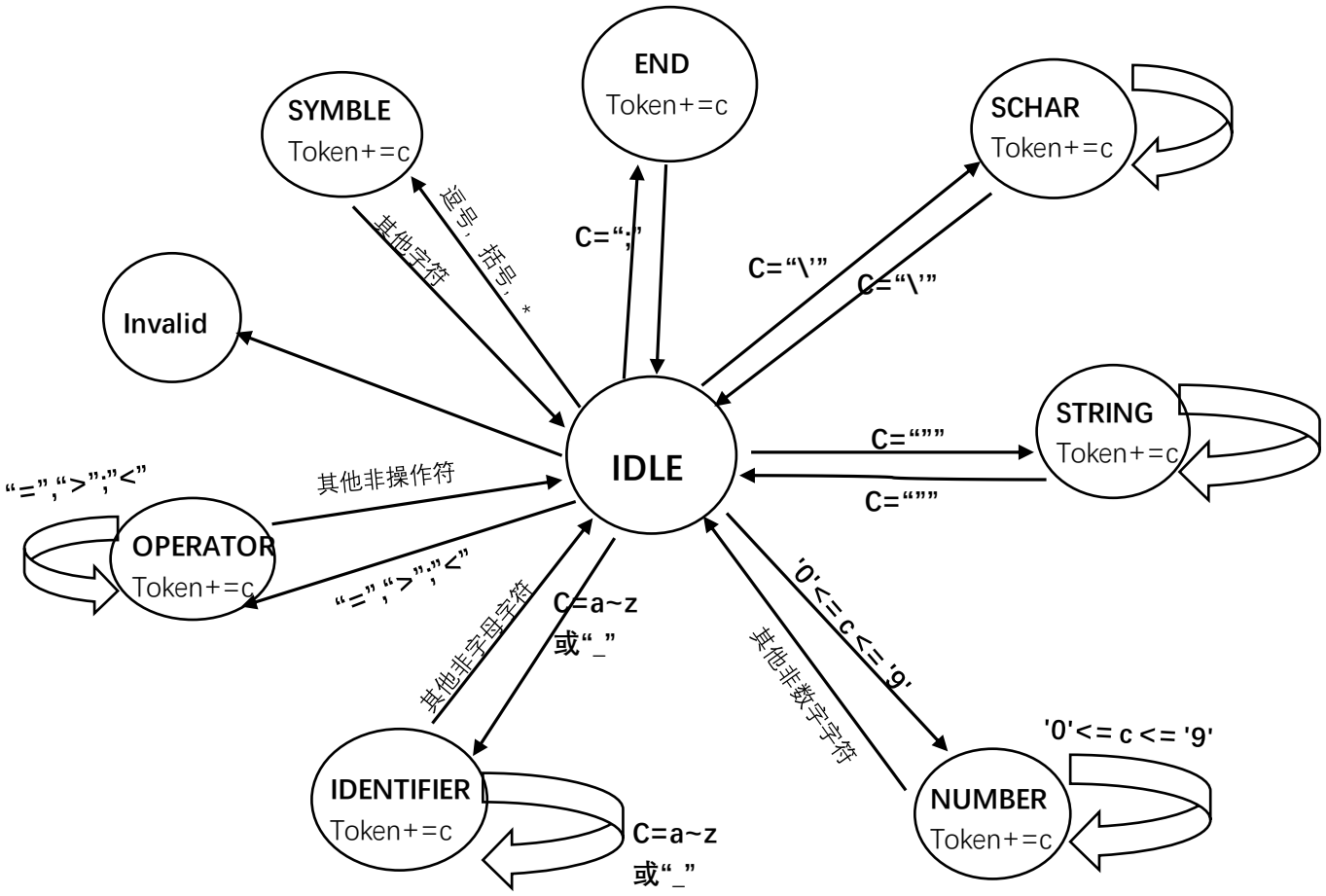
SCHAR: char 类型；

STRING: string 类型；

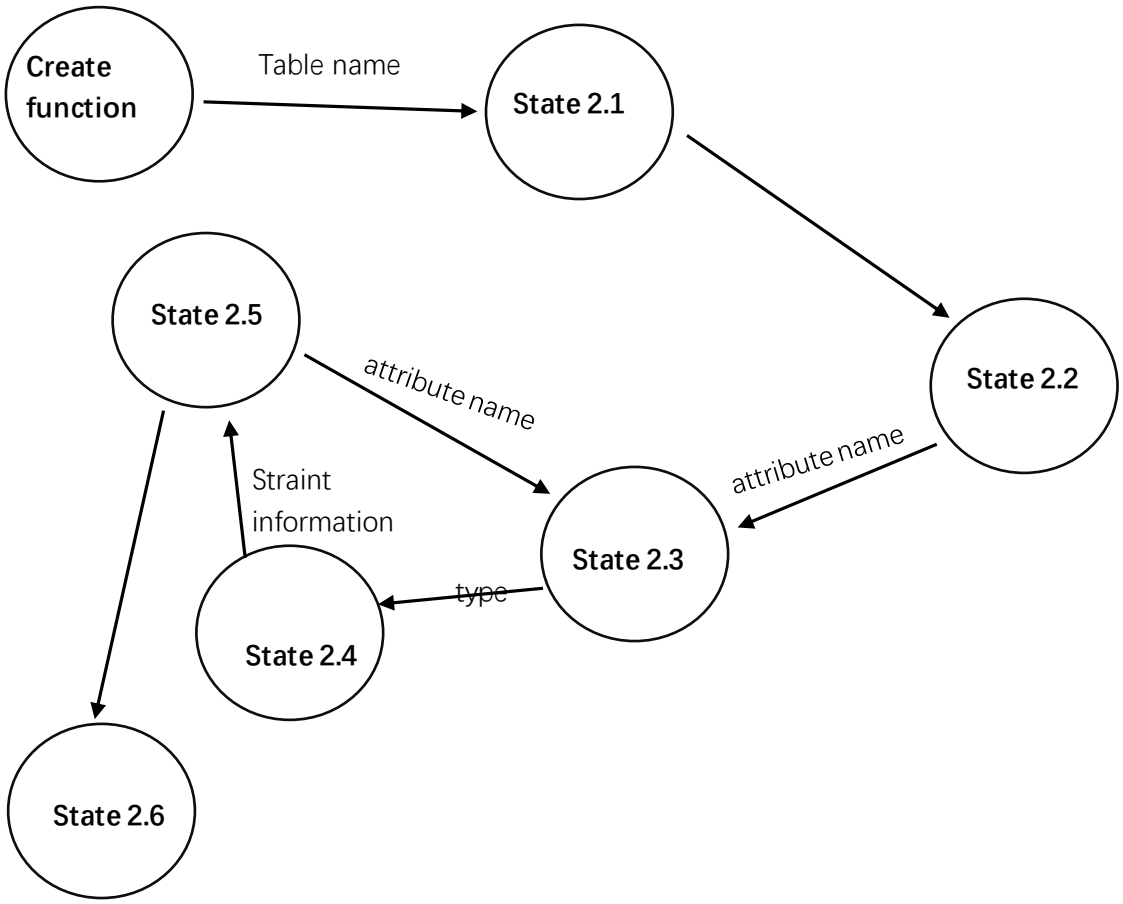
SYMBLE: “,”,“(”,“*”等 symble；

OPERATOR: “=”,“>”等操作符。

次函数主要状态图如下,c 是每次从 sql 里读取的字符，token 是解析出来的单个有意字串。每次回到 IDLE 状态时，token 和对应的 type 都加到相应的 vector 里。若出现不能识别的则给出错误信息。



2. `void create_table()`

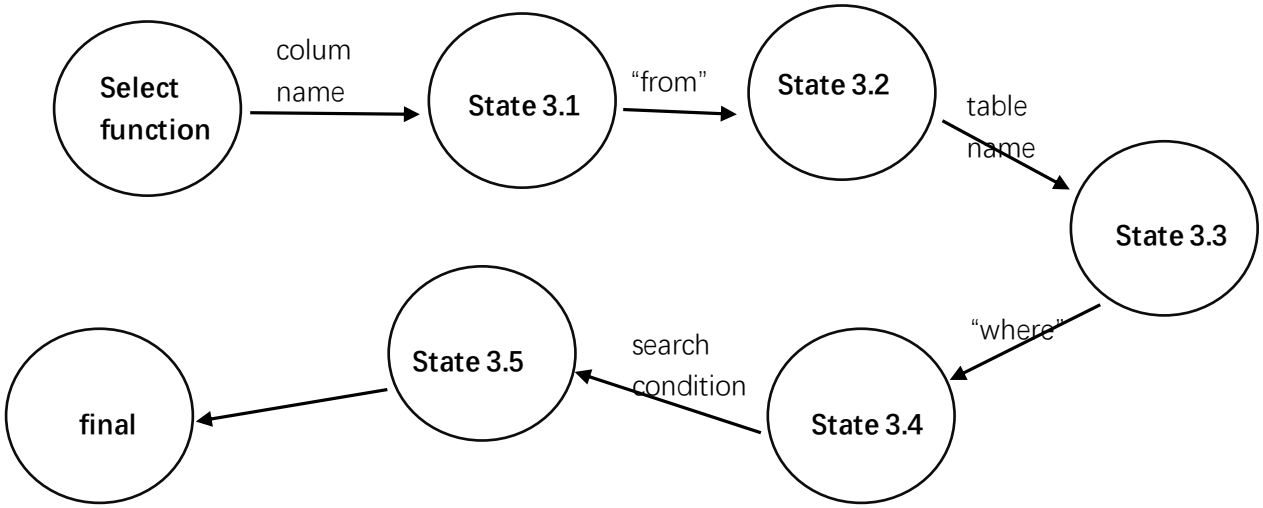


函数按照状态图所示顺序读取 tokens 中的有意字串信息，每一步都会对信息进行合法性检验，若不合法，则抛出错误信息，执行下一条语句。

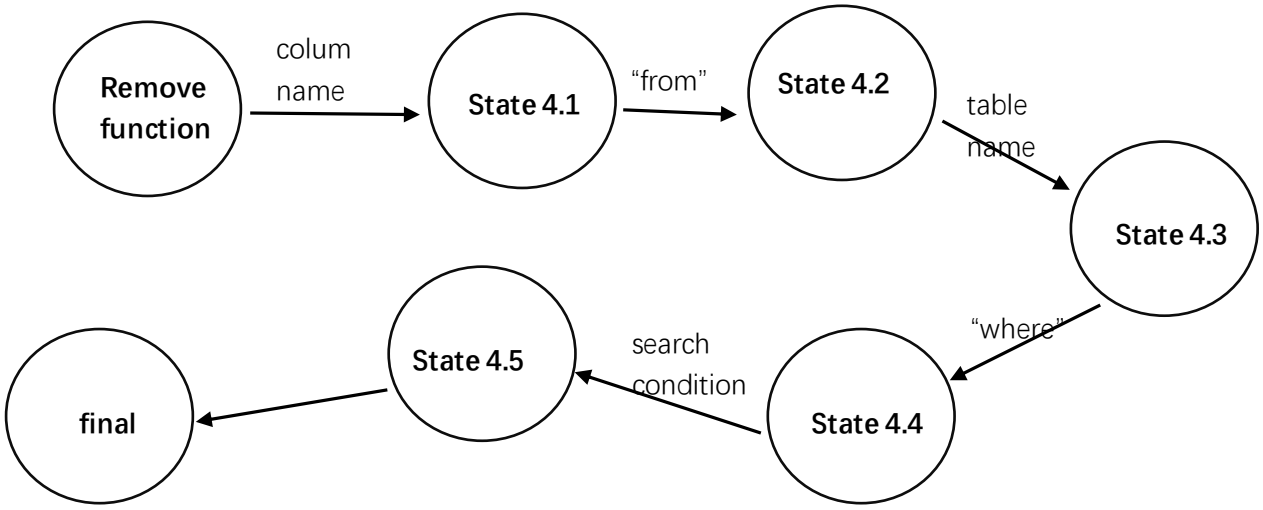
每一步得到信息时，都会对信息进行分析处理，得到例如表名，类型，约束条件等，将所得信息封装成对应的类，当顺利读取到“;“时，将所得信息传递给 API 里相应的函数执行相应操作。

对于之后的函数将不再赘述，只给出主要的状态图流程。

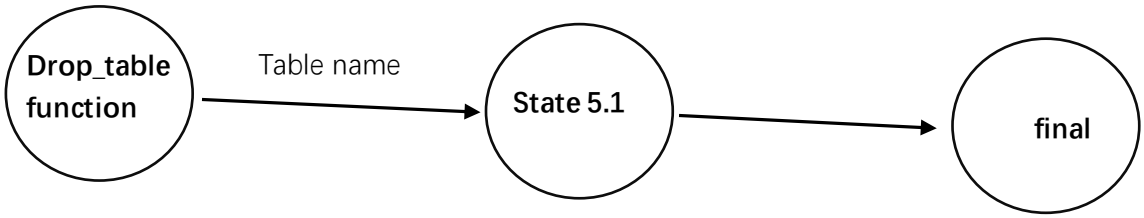
3. `void select()`



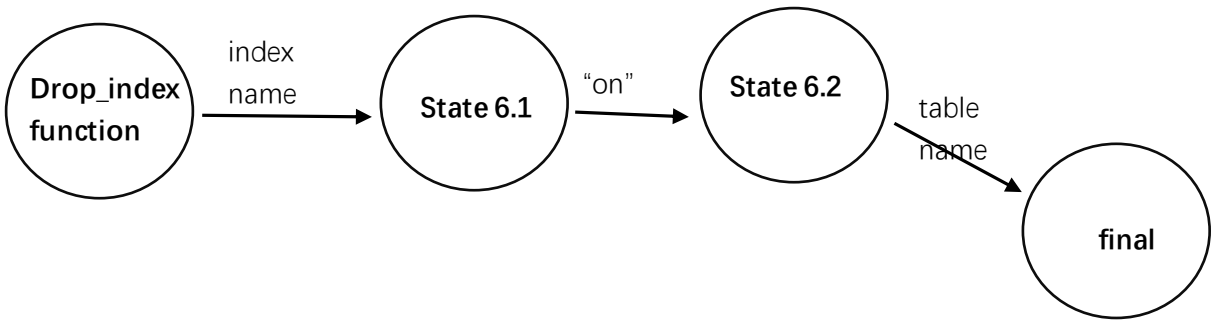
4. `void remove()`



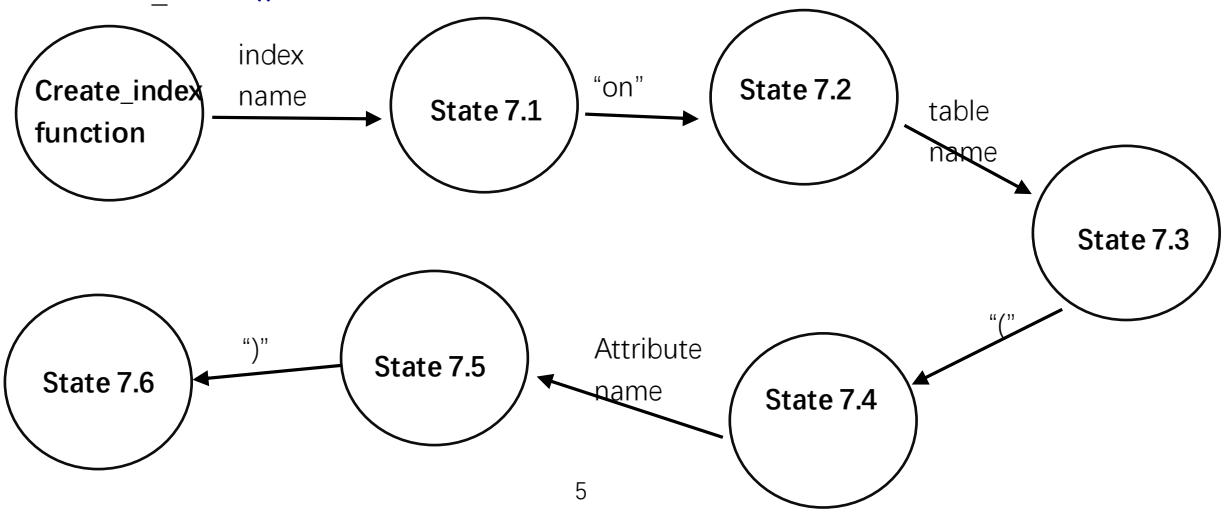
5. `void drop_table()`



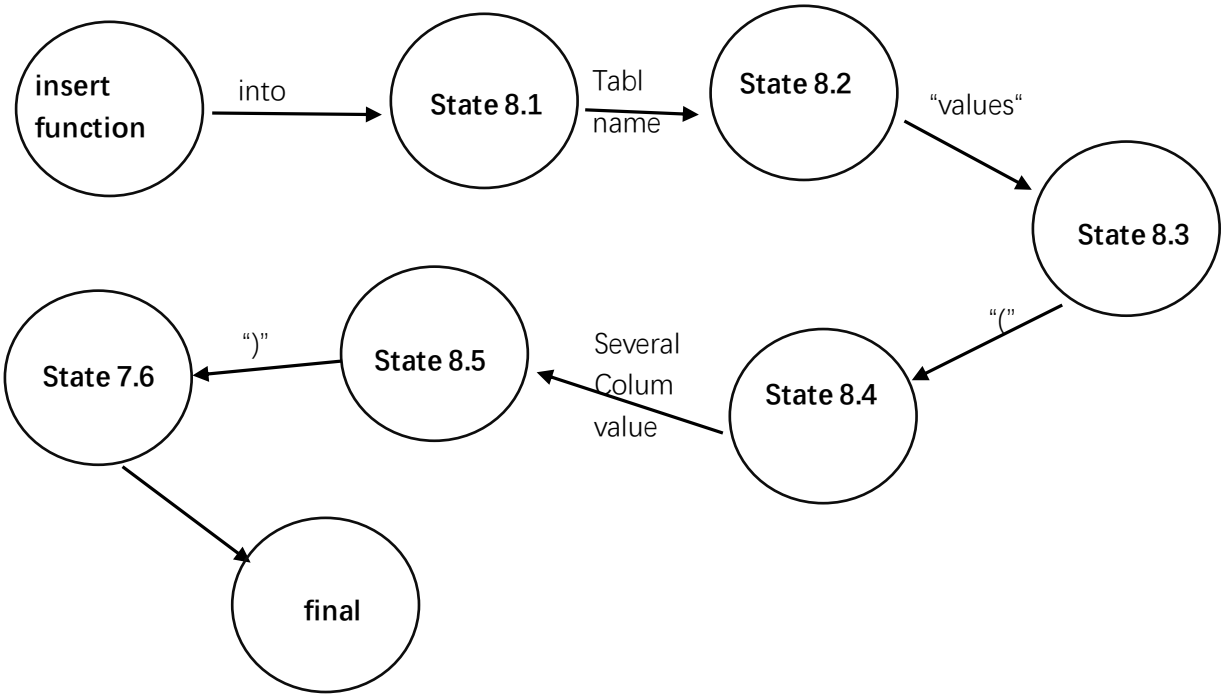
6. `void drop_index()`



7. `void create_index()`



8. void insert()



五. 测试样例

测试方法主要是输入各种类型的 sql 语句进行测试。

1. 输入语法正确的 sql 语句，打印出需要传给 API 的参数看是否符合要求。

如：

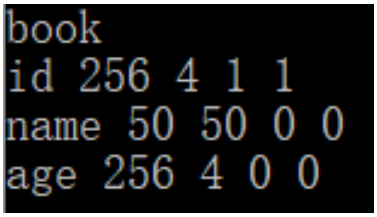
```
Create table book (
  id int primary key,
  name varchar(50),
  age INT,
  unique(id));
```

测试程序：

```
cout<<table_name<<endl;

for(vector<Attribute>::iterator iter=a.begin();iter!=a.end();++iter)
  {
   Attribute k = *iter;
   cout<<k.get_name()<<" "<<k.get_type()<<" "<<k.get_length()<<"
   "<<k.is_primary()<<" "<<k.is_unique()<<endl;
  }
```

得到：



信息正确

2. 输入语法有问题的 sql 语句，看是否提示出错信息。

如：

Drop table;(缺少 table name 信息)

得到：

正确给出错误信息。

经过测试，各方面表现均达到目标。