Interpreter 设计报告

计算机科学与技术学院 计算机科学与技术 1404 班 杨健 3140102451

一、 模块概述

MiniSQL 的整体设计要求与用户实现交互，可以做到程序流程控制，即并初始化->‘接收命令、处理命令、显示命令结果’循环->退出“流程，同时对于用户输入的命令进行语法正确性和语义正确性的检查并对不正确的命令显示错误信息。

Interpreter 负责把用户输入的命令转化成内部数据结构的表示，同时检查语法和语义的正确性，对正确命令调用 API 进行相应功能的实现，对错误命令向用户反馈错误信息。

二、 主要功能

检查语法正确性：正确则检查语义正确性，错误返回错误信息。

检查语义正确性：调用 catalog 模块，检查表、索引等是否存在等，正确则由 API 实现语句，错误则返回错误信息。

内部数据结构表示：解析语句，将语句信息保存至内部数据结构中，供 API 调用实现功能。

三、 对外提供的接口

1. 进行语句解析

   Parse(string command)//需要用户输入语句

2. 各成员变量

```
int operation_type;//操作类型
int error_info;//错误信息
Table tableInfo;//表的信息
Index indexInfo;//索引信息
Data value; //insert 的数据
Conditiondata  conditions;//where 后的情况
Conditiondata Uniqueconditions;//unique 属性对应值
```

四、 设计思路

从头开始对用户输入的语句进行解析，根据命令格式进行分离，将相应信息填入对应的成员变量中，供 API 调用。

首先需要一个用于分解命令的函数，每次读进命令进度的一个部分，然后需要对读入的单词进行判断，执行相应操作，再次读入下一个，直至读到结束标志，完成解析。

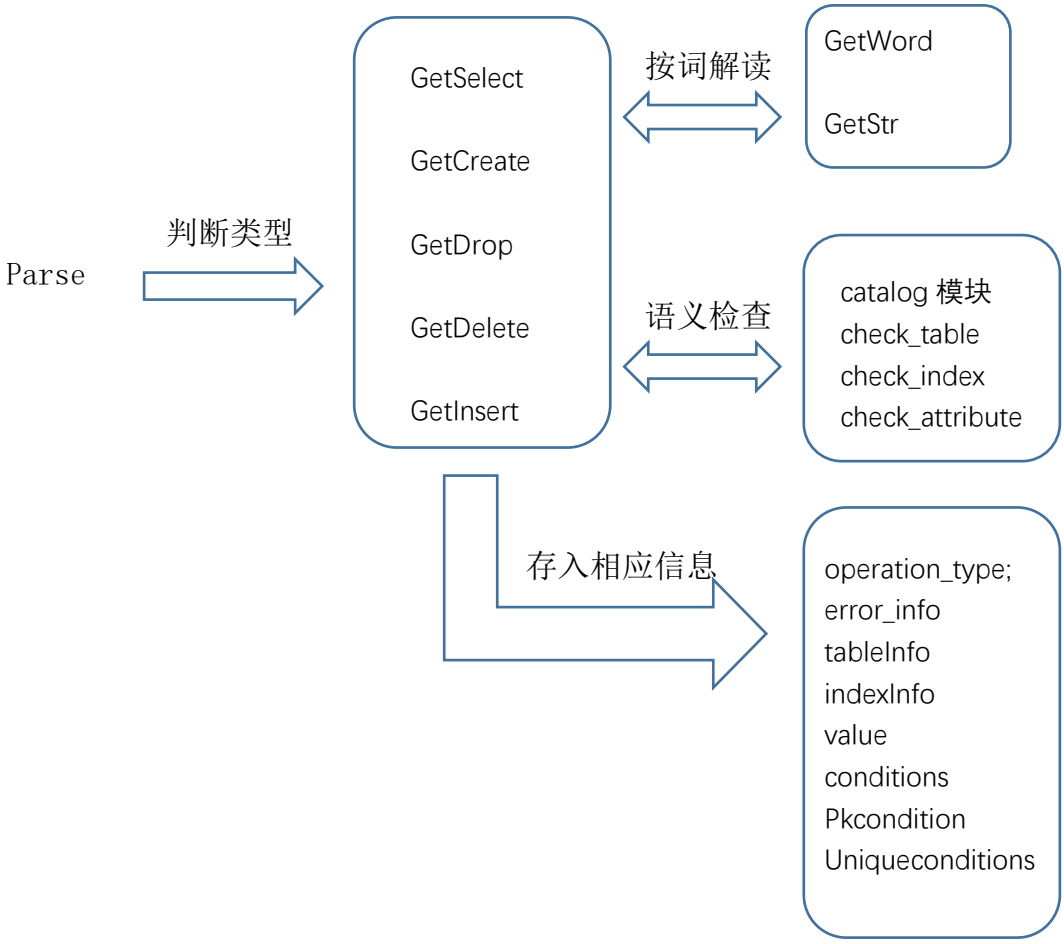
其中分解命令函数 bool GetWord(string &src, string &des)将输入的语句 src 读取一个单词，存入 des 中，并删除 src 中该单词，便于下次再调用。获得单词后，首先判断命令的类型，再根据命令调用相应进一步解析函数 GetSelect 等，并在解析中不断调用 catalog 函数判断语法和语义的正确性，错误则结束解析，返回错误信息。

其中 where 后属性采用最大最小值的表示方法，即保存比较类型，用

maxvalue，minvalue 和 eqvalue 保存对应值，如 A<4,则保存类型为小于，maxvalue=4，minvalue=设定最小值。

五、 整体架构

设计类 Interpreter，在其成员函数中存储语句内容所有相应信息，提供对外调用。在解析过程中，进行同步语法检查，同时调用 catalog 模块中的检测函数，实现语义正确的判断。



六、 关键函数和代码（伪代码）

获取命令中的各个词：

```
bool GetWord(string &src, string &des){
    des.clear();
    for(srcpos=0;scrpos<src.length();srcpos){
        if(src[srcpos]!=(' ',' \t',' \n',' \r')//去除无用字
            break;
    }
    switch（tempchar=src[srcpos]）
    {
        case ',': case '(': case ')': case '*': case '=':
```

```
        case '\'':
            des += tempchar;//读取完毕，存入 des
            src.erase(0,srcpos);//删除前方已被读取的字
            break;
        case '<' : //大于号，可能是<, <=, <>
            put into des and tempchar = next key
            if(tempchar == '=' ||tempchar == '>' )
                add to des and delete
            else
                delete
            break;
        case '>' //小于号，可能是>, >=
             if(tempchar == '=' )
                add to des and delete
            else
                delete
            break;
        default: //读取数值
             put into des until key is not a number or a letter
            break
        }
        return true;
    }
```

对于指令的解析函数思路相同，此处以 GetSelect 为例(基本用到所有成员变量)：

```
    bool GetSelect(string &query){
        try{
            word = get a new word from command
            test word is "*" and next "from"
            word = get a new word
            temptable.name=word;
            check if the table exists,
                if not, error_info=table not exist and return false
            tableinfo = get_table_info//从 catalog 中获得表的信息
            if there is no word in command
                operation_type=SEL_NOWHERE, return true
            else{
                 operation_type=SEL_WHERE
                 get conditons //获得 where 后的数据，并保存
                 get unique conditions//保存对应值，查重检验
            }
        }
```

```
    catch () {};
}
```