Interpreter 模块设计

计算机学院 软件工程 1201 尹依婷 3120102057

一、模块概述

Interpreter 模块直接与用户交互，主要实现以下功能：程序流程控制，即“启动并初始化 → ‘接收命令、处理命令、显示命令结果’循环 → 退出”流程。接收并解释用户输入的命令，生成命令的内部数据结构表示，同时检查命令的语法正确性和语义正确性，对正确的命令调用 API 层提供的函数执行并显示执行结果，对不正确的命令显示错误信息。

二、主要功能

1．启动并初始化：启动 minisql 程序，初始化 categoryManager;，recordManager 和 bufferManager。

2．接收用户输入的命令：当用户输入“;”时，视为一条命令的结束，interpreter 模块接收此条命令并保存在字符串数组中。

3．检查命令语法的正确性：如果正确，进行命令的语义正确性检查；如果不正确，告诉用户命令中存在的语法的错误。

4．检查命令语义的正确性：如果正确，进行命令的解释相关函数的调用；如果不正确，告诉用户命令中存在的语义的错误。

5．显示命令执行结果：如果命令执行正确，显示执行结果；如果命令执行错误，告诉用户错误原因。

三、对外提供的接口

1．接收用户输入的命令，并检查语法正确性

```
public bool InitCommand(CString &cmd);
```

2．解释用户输入的命令，并检查语义正确性

```
public bool ExecuteCommand(CString cmd);
```

3．创建表操作

```
public bool CreateTable(CString TableName,CString TableInfo);
```

4．删除表操作

```
public bool DropTable(CString table_name);
```

5．创建索引操作

```
public bool CreateIndex(CString IndexName, CString TableName, CString
AttName,   bool unique);
```

6．删除索引操作

```
public bool DropIndex(CString IndexName);
```

7．选择操作

```
public bool Select(CString Selection, CString TableName, CString
```

```
Condition);
```

8. 插入操作

```
public bool insert(CString table_name, CString info);
```

9. 删除操作

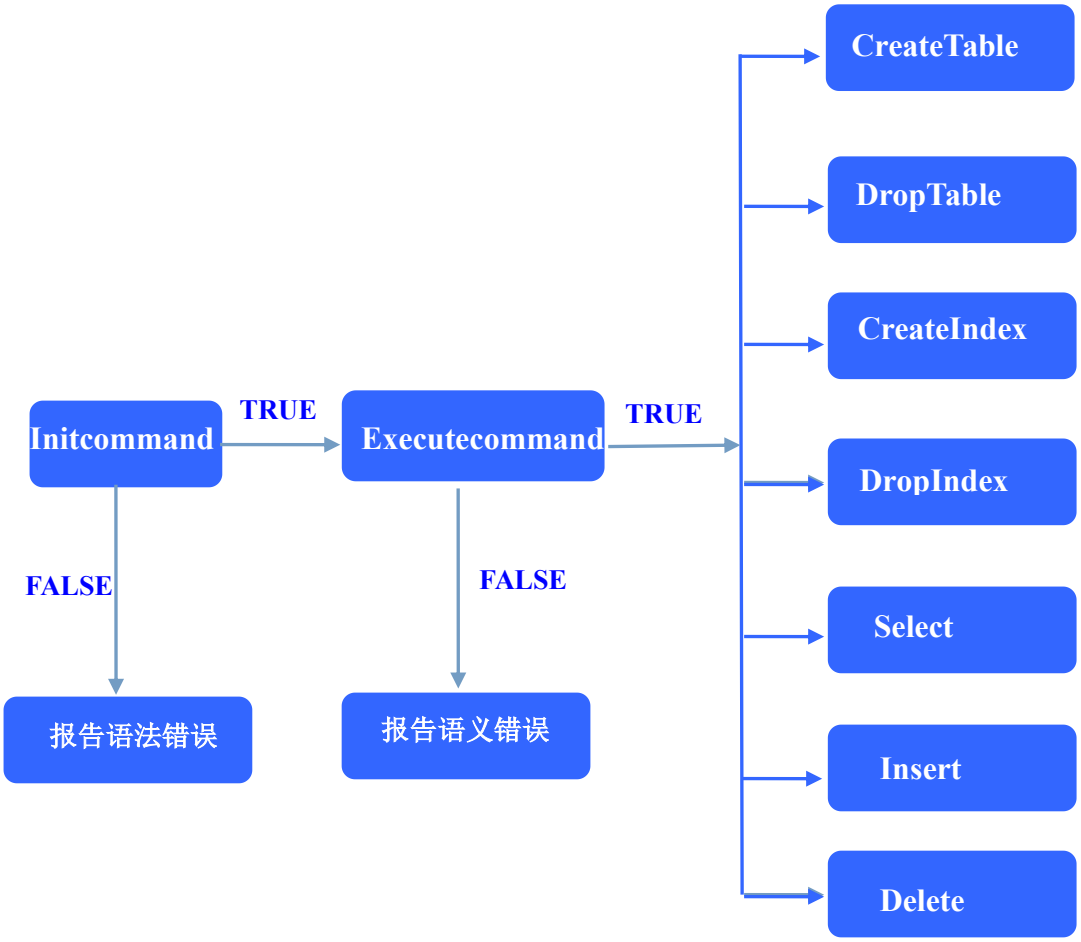
```
public bool Delete(CString Table , CString info);
```

四、设计思路

Interpreter 以流程图的操作模式，读入用户命令以后，对命令进行解析，包括语法上和语义上的分析，命令语法和语义都正确以后，从 API 中调用相应的函数。

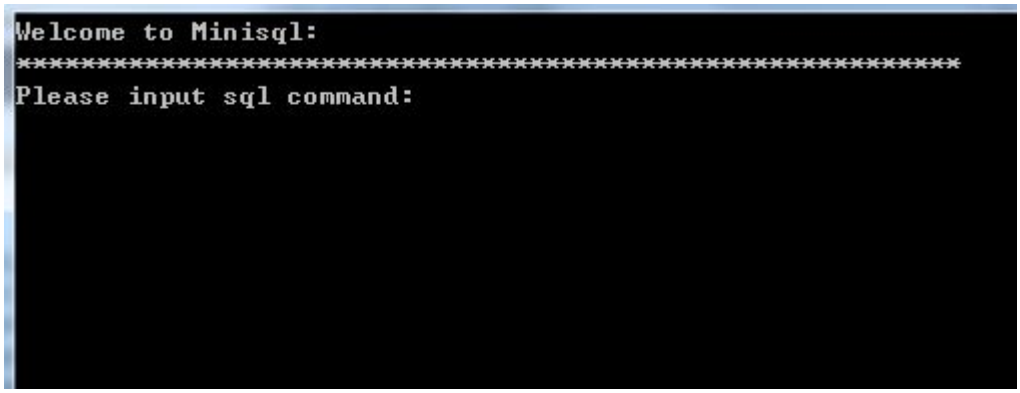
Interpreter 模块处理用户输入命令的流程如下图：

其中 “quit;”和 “execfile 文件名;”命令在 main()中实现。

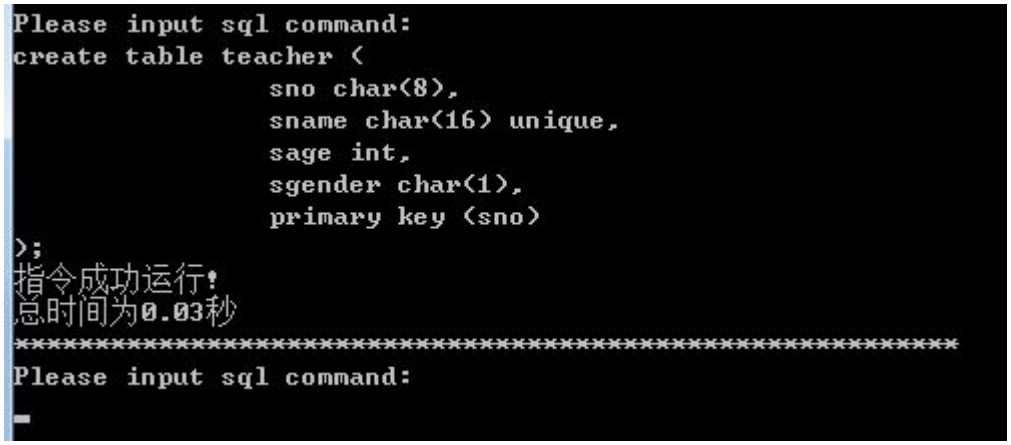


五、主窗口设计及报错机制

Interpreter 模块直接与用户交互，呈现的窗口如下：



用户在命令行输入语句，interpreter 会进行语法和语义的检查。
如果语句正确，会打印“指令运行成功!”和指令运行的总时间

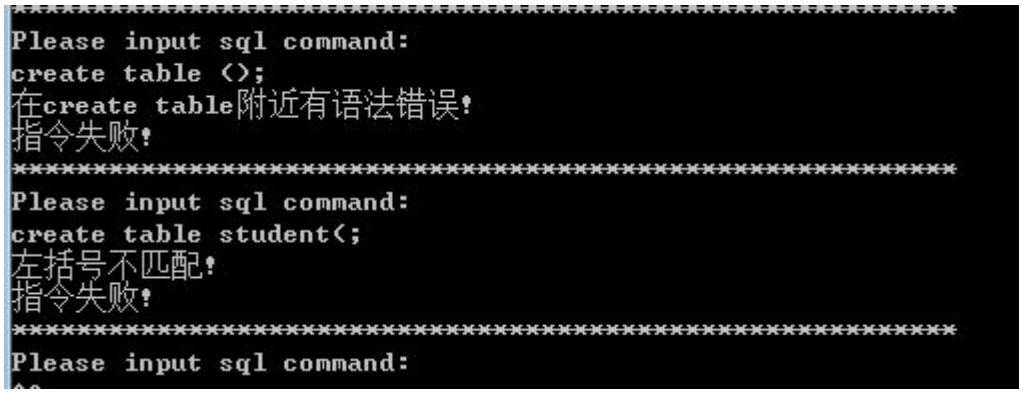


如果语句的语法或语义有错，会打印“指令失败!”和具体错误原因，本程序所涉及的错误有：

"右括号不匹配!"
"左括号不匹配!"
"单引号不匹配!"
"语句为空!"
"表达式应以分号结尾!"
"表达式错误!"
"在 create table 附近有语法错误!"
"在 drop table 附近有语法错误!"
"在 create index 附近有语法错误!"
"在 drop index 附近有语法错误!"
"在 select 附近有语法错误!"
"在 select 附近有语法错误!"
"insert into 附近有语法错误!"
"在 delete from 附近有语法错误!"
"不存在这个表"
"第" << << "个格式出错"
"超过字符数限制!"
"属性个数不符合."
"有重复值输入!"

"在 primary 附近有语法错误!"
"在 primary key 附近有语法错误!"
"在;附近有语法错误!"
"类型错误!"
"不存在该表"
"格式有误"
"没有找到记录!" 等等。

超过本程序所设定的错误范围的错误将直接打印“指令失败!”不再说明原因。



六、关键函数及代码（伪代码）

初始化用户输入的命令（检查语法）

```
bool interpreter::InitCommand(CString &cmd)
{
    stack.Empty();
    Read in user's command in CString cmd;
    If there is any grammar error, quit and print error;
    Replace all '\t','\n','\r' by '';
If (cmd) return TRUE;  //初始化完成返回 TRUE
}
```

执行用户输入的命令（同时检查语义）

```
bool interpreter::ExecuteCommand(CString cmd)
{
    if(InitCommand(cmd))
    {
        if(temp.Find("COMMAND") == 0)
        //COMMAND refers to specific commands including "create table","drop table","create index","drop index","insert" and so on
        {
            If command is correct, do "COMMAND";
            If command exists errors
            {   print error;
                return false;
```

```
                    }
                }
            }
            Else return false;
}
```

语法检查和语义检查都通过以后，interpreter 模块将执行以下具体操作：

Insert 操作

```
bool interpreter::insert(CString table_name, CString info)
{
    record Record;
    Check if the table exists first;
    if(table_name != currentTableName)
    {
        Clear related variables;
          Read in and save attributes;
        Check if the form is correct;
          Check if any reduplicate value input;
     }
    return 1;
}
```

Create table 操作

```
bool interpreter::CreateTable(CString TableName,CString TableInfo)
{
    for(k=0;;k++)
    {
        CString AttName,AttType,s,temp;
        int AttTypeCount;
        AttName = p;
        p += AttName.GetLength()+1;
        if(AttName == "KEY")
        // KEY refers some keyword about attribute such as “primay”,”unique” and so on
          Deal with "KEY";
        Check the consistency of type and input;
    return true;
}
```

Drop table 操作

```
bool interpreter::DropTable(CString tableName)
{
    rm->rmRecord(tableName);
    cm->dropTable(tableName);
```

```
    return 1;
}
```

Delete 操作

```
bool interpreter::Delete(CString table_name, CString info)
{
    Find the table first;
    Find the attribute then;
    Find the type of the attribute;
    rm->rmRecord(table_name, &Table_info, rp);
    delete rp; //free the pointer
    return 1;
}
```

Create index 操作

```
bool interpreter::CreateIndex(CString IndexName, CString TableName, CString AttName, bool unique)
{
    if(cm->readTable(TableName, a))
    {
        Get the attribute name that required;
        Check if the attribute exists;
          if(a.info[i].unique == false && unique == false)
          {
                Print "该属性不唯一，无法创建 index!";
                return 0;
          }
    }
    else
    {
          print "不存在此表" ;
          return 0;
    }
    return 1;
}
```

Drop index 操作

```
bool interpreter::DropIndex(CString indexName)
{
    return 1;
}
```

Select 操作（分成两种情况）

```
bool interpreter::Select(CString Selection, CString TableName, CString Condition)
{
	bool flag;
	if(Condition.GetLength() == 0)
		flag = print1(TableName, Selection);
	else
		flag = print2(TableName, Selection, Condition);
	return flag;
}
bool interpreter::print1(CString Table_name , CString attr)
{
	rp = rm->readRecords(Table_name, Table_info);
	cout.setf(ios::right);
	if(attr == "*;")
	   Print all the attributes that satisfy the condition;
	else
	   Print required attributes that satisfy the condition;
	cout.unsetf(ios::right);
	delete rp;
	return 1;
}
bool interpreter::print2(CString Table_name , CString attr , CString info)
{
	 flag = cm->readTable(Table_name , Table_info);
	 Check if the table exists;
	Count the size of table_info;
	if(size%4 != 0)
	{
		print "格式有误" ;
		return 0;
	}
	Check if the attribute exists;
	Configure the operation that in the condition;
	Check the type of table_info;
		rp = rm->readRecords(Table_name, Table_info , ccp);
	Find the record;
	Print the result of selection;
	cout.unsetf(ios::right);
	Free the pointer rp;
	return 1;
}
```