Interpreter 模块设计报告

王哲峰（3110000026）

一、模块功能

Interpreter 模块直接与用户交互，主要实现以下功能：

1. 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。
2. 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、设计方法

本次设计的是一个简易的数据库，因此这个解释器也没有用到正则表达式等高级工具，仅仅是字符串的分析及处理。根据空格进行分割，每读入一个词都进行一次解析，并根据解析出的内容进行相应的操作，如果不符合语法则给出相应的错误信息然后重新开始。如果语法正确，则在解析出全部参数之后，调用 catalogman 模块函数进行语义检查，如果出现错误则抛出异常，interpreter 接收异常然后显示错误。最后调用相应 API 进行查询，如果查询过程中出现错误，则也会以异常的形式抛给 interpreter 进行输出显示。

三、过程说明

在 main.cpp 中，设置一个循环不断读入命令到字符串 query 中，每次读取一行。如果该行命令中没有出现分号，将命令加到 query 中继续进行读取，直到出现分号，则将当前 query 加入到 interpreter 中进行解析，然后将 query 清空后继续循环下去。

在 interpreter 中，具体解析步骤如下：

（1）添加相应的空格

因为本解析器主要以空格进行分割，所以当某些地方缺少空格的时候则会出现解析失败的情况，如“create table stu(”，由于没有正则表达式匹配，解析器难以将 stu 与（分开，因此在解析之前，应给对原始命令中添加一些空格。由于缺少空格的情况主要在括号和逗号等处，所以我们只需这些符号找出来然后在符号的两边添加空格即可。

（2）如果 query 为 quit 则直接退出程序，否则将 query 导入 strstream 字符串流 str 中，该字符串流是以空格进行分割的，所以每次读取只需像 cin 一样读入即可。

（3）读入第一个词，如果这个词是 create,drop,insert,select,delete 或 execfile，则分别进入相应的函数中，否则提示语法错误。

（4）各函数说明：

void Interpreter::onCreateOp(strstream& str)：

此为创建table或index的函数。首先读取表名，然后分别读取每一行的属性名和类型，如果是char则还需要读取字符长度，最后以一个primary key()命令进行结尾，全部命令解析完毕。调用 CatalogMan::createTableCheck(tablename, newTable);函数进行语义检查，如果没有抛出任何异常则调用

APII::createTable(tablename,newTable);进行表的创建，如果也未返回任何异常则显示表创建成功。

void Interpreter::onDropOp(strstream& str):

形式与create类似，先判断是drop table还是drop index，然后分别调用API:: dropTable(tab leName);和API:: dropIndex(indexName);即可。

void Interpreter::onInsertOp(strstream& str):

解析出名字和各属性需要填写的值后调用CatalogMan:: insertTupleCheck(tableName, newValues);函数进行检查，如果出现该表不存在，字符串超出限定长度和其他不符合规范的问题，则会报错，否则调用API:: insertTuple(tableName , newValues);进行Tuple的添加。tableName是所插入表格的名字，newValues是一个vector<string>，分别是所属行所需要填写的值。

void Interpreter::onSelectOp(strstream& str):

调用API:: selectTuple(newQuery);函数即可，newQuery是一个在global.h中定义的类Query，里面包含一个string类型的m_tableName和一个vector<Condition> 类型的m_condition。Condition也是在global.h中定义的一个类里面包含一个string类型的属性名，一个OP_TYPE类型的操作符和一个string类型的操作数，三者合起来可以组成一个完整的条件语句，如sage > 22。Interpreter只需将这些条件全部解析出来然后传入到newQuery中，调用API即可完成查询。

void Interpreter::onDeleteOp(strstream& str):

调用API:: deleteTuple(query);即可，query是Query类型，前面已经提到，包含一个表名和若干条件，Interpreter只需将这些条件全部解析出来然后传入到newQuery中，调用API即可完成查询。

void Interpreter::execfile(string filename):

读取此文件中的内容，然后依次加入到query中即可，每一条命令的调用流程与CLI中输入的命令效果完全相同。

四、感想与心得

一开始认为 Interpreter 是几个模块中相对来说比较简单的一个模块，只是一大堆字符串操作的堆积，但是在实际的编码中则碰到了很多问题。由于本身编程基础比较差，对 C++的 STL 和一些字符串处理函数不是很熟悉，导致了一开始根本无从下手，知道发现了字符串流 strstream，通过这个类我可以很容易地对每一个词进行提取和分析，大大缩小了了代码量。

本解析器没有用到正则表达式，所以只能依次对每一个错误进行判断，因此有些输入的错误可能不能及时给出相应的错误提示，这是本解析器需要改进的地方。但是如果输入的查询语句符合语法规范，解析器还是可以能够做到正确将各参数提取出来然后调用相应的 API 进行查询的。

由于之前对于一些接口规范的没有进行充分的讨论，导致后来整合的时候出现了很多意想不到的问题，程序编写一度进行不下去，小组中的一名成员的离去更是使得整个项目几乎烂尾掉，不过最后我们通过整整三天辛苦的编程还是基本实现了所要求的全部功能。通过了这门课程和最后大程的编写我学到了很多东西，不知是数据库方面的，更是学会了如何与队员之间进行有效的沟通和分工，以达到最大的编程效率，这些将在我以后的工作和学习中起到非常重要的作用。