Interpreter模块设计报告

计科 1404 卢涛 3140102441

**一 、 模块概述**

Interpret 模块是MiniSQL 系统中负责与外部输入相连接的模块。该模块负责接收外部输入的SQL语句，并对其进行语法分析，并传给API 执行相应的操作，最后打印出查询结果。

1. **主要功能**

读入SQL语句：每次读入一行，判断最后一个字符是不是‘；’，如果不是的话，就把它加入到输入的字符串中，继续读入下一行，直到读入的最后一个字符为‘；’则停止输入，最终得到的字符串即为输入的SQL语句。

解析SQL语句：将得到的输出分割成单个的单词或者是符号，如果发现语法上存在错误，就打印错误信息，如果语法没有错误，根据catalog的信号，判断是否能够执行该查询，如果不能，打印错误信息，如果可以，就调用API进行执行。

1. **具体解析过程**

一个单词一个单词地读入，首先读入operation，比如create table，select等，再读入要进行操作的table name，index name等，接着读入value值，最后读入操作的条件。SQL结构大致如下：

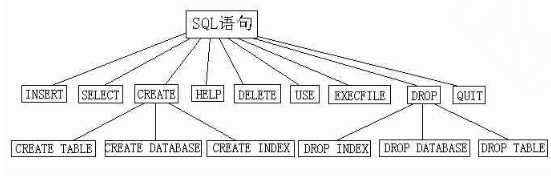


图1：SQL语句结构

具体实现方式：

先读入第一个关键，可分为：1、create 2、drop 3、delete 4、select 5、insert

6、quit 7、execfile

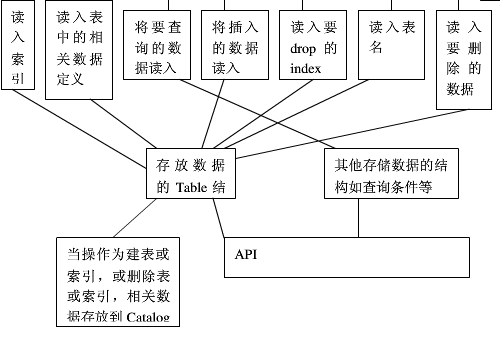
再对某些查询语句读入第二个关键词，比如：对create 有 create table，create index，对drop 有 drop index，drop table 。

最后对每条查询语句剩下的内容进行解析，是interpret的内部成员分别获得相应的值，传给API进行下一步地执行。

如果中途出现语法错误，则break，打印错误信息，或是从catalog里面读取信息后，发现该查询无法执行，也打印出相应的错误信息。

interpret 类的定义

|  |
| --- |
| class interpreter{  public:  int operation; //操作  string tablename; //要进行操作的table name  string indexname; //要进行操作的index name  string colname; //要进行操作的列  vector <Condition> conditions; //条件向量  vector <Attribute> column;  Table tableinfo; //table 信息  Index indexinfo; //index 信息  Row row;  int primarykeypos; //primary KEY的位置  int uniquekeypos; //unique KEY的位置  int wrongtype;  vector <Condition> uniqueconditions;  string filename; //要操作的文件名  interpreter(){ //构造函数  operation = UNKNOW;  tablename = indexname = colname = "";  colname = "";  primarykeypos = uniquekeypos = -1;  wrongtype = -1;  }  void initcol(); //初始化  void initcond();  void initvalue();  void inittable(){  tableinfo.name = "";  tableinfo.attriNum= tableinfo.blockNum= tableinfo.totalLength = 0;  tableinfo.attributes.clear();  }  void initindex(){  indexinfo.blockNum = indexinfo.columnNum= -1;  indexinfo.index\_name = filename = "";  } //成员函数  void analysis(string input); //解析SQL语句  bool cutword(string &input, string &word); //分割字符串  bool cutstr(string &input, string &word);  void MakeInitilate(){  operation = UNKNOW;  tablename = "";  indexname = "";  initcol();  initcond();  initvalue();  inittable();  initindex();  }  }; |



图二：interpreter实现流程

1. **部分代码：**

以drop为例

|  |
| --- |
| else if (word == "drop"){  flag = cutword(input, word); //分割单词  if (!flag)  return;  if (word == "table"){ //drop table  flag = cutword(input, word);  if (!flag)  return;  tablename = word; //赋值tablename  if (!catalog.isexisttable(word)){ //调用catalog模块  operation = NOTABLE;  return;  }  tableinfo = catalog.gettable(word); //调用catalog模块  flag = cutword(input, word);  if (!flag){  operation = DROPTABLE;  return;  }  }  else if (word == "index"){ //drop index  flag = cutword(input, word);  if (!flag)  return;  indexname = word;  if (!catalog.existindex(word)){ //调用catalog模块  operation = NOINDEX;  return;  }  flag = cutword(input, word);  if (!flag){  operation = DROPINDEX;  return;  }  }  } |

1. **测试要点**

一是输入错误的SQL语句，看是否输出报错信息，二是输入正确的SQL语句，看解析后的命令是否正确。

Interpreter API Catalog manage三个模块的执行关系图：

用户命令 interpreter 存储输入的数据结构

创建、删除表或索引

Catalog的读写 API

查询语句

Table 和 index 类的定义

|  |
| --- |
| class Table  {  public:  string name; //all the datas is store in file name.table  int blockNum; //number of block the datas of the table occupied in the file name.table  //int recordNum; //number of record in name.table  int attriNum; //the number of attributes in the tables  int totalLength; //total length of one record, should be equal to sum(attributes[i].length)  vector<Attribute> attributes;  Table(): blockNum(0), attriNum(0), totalLength(0){}  };  class Index  {  public:  string index\_name; //all the datas is store in file index\_name.index  string table\_name; //the name of the table on which the index is create  int columnNum; //on which column the index is created  int columnLength;  int blockNum; //number of block the datas of the index occupied in the file index\_name.table  int type;  Index(): columnNum(0), blockNum(0){}  }; |