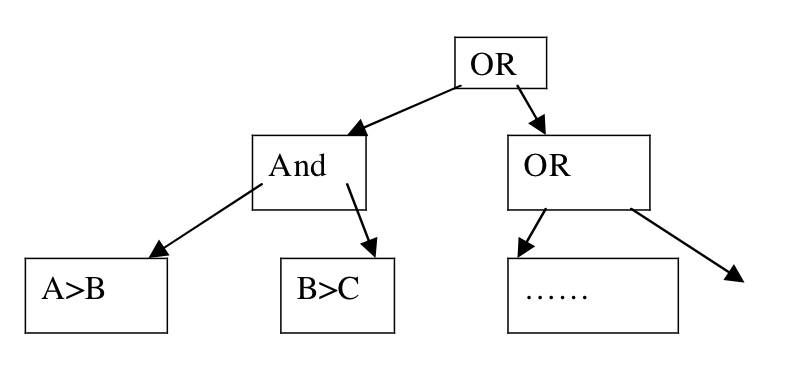
功能及实现概述：

将用户的输入命令进行语法分析和语义解析并得到需要的命令参数，最后将该命令参数封装成对应命令的参数类对象（即内部表达形式），允许 API 模块调用，供其他模块读取使用。所有操作由一个类的成员函数完成；若检测到错误输入，即刻输出提示信息并不再做Parsing和封装。

因为该模块设计中只包含一个主要类Interpreter，且其成员函数和数据对象的名称都较清晰地代表了它们的功能，实现的功能也是明确的，因此不基于每一项讲述功能，而重在图解其翻译过程；在最后附上头文件的实现，必要处给予注释说明。

1. 解析命令过程:

将命令读入，截取token，保留有用的信息。然后解析该命令，判断出命令类型，交由相关函数处理；提取语句中与该类型命令匹配的字符串作为命令参数，交给 API模块调用其余模块相关函数执行操作。如果操作是包含条件的语句，将条件放入大致结构如下的条件树中：

第一步：根据语句中的第一个关键字进行分类，可分为：

1.CREATE; 2. DROP; 3.SELECT; 4.DELETE; 5.INSERT; 6.QUIT；

第二步：可将 CREATE 类型进行进一步的分类，即CREATE TABLE和CREATE INDEX；类似地，可将 DROP 类型分类为DROP TABLE和DROP INDEX。

第三步：对各语句进行分别处理，封装后返回相应的数据结构并交给 API 进行相关判断操作；同时在CatalogManager中写入元数据，创建表、索引等的信息。

SQL语句子集的解析逻辑图大致如下：

|  |
| --- |
| class Interpreter{  public:  int m\_operation;  string m\_tabname;  string m\_indname;  string m\_colname;  vector<Attribute> column;  vector<Condition> condition; //条件列表  Row row;  Table getTableInfo;  Index getIndexInfo;  int PrimaryKeyPosition; //若建表时未定义则初始设为-1  int UniquePostion;  vector<Condition> UniqueCondition;  friend void Execute(); //API中的函数，允许其获取Interpreter中的数据  Interpreter();  void Parse(char\* command); //核心函数，实现了上图SQL语句到内部表达的转换流程  void initcol();  void initcond();  void initvalue();  void inittable();  void initindex();  bool GetWord(string & src, string & des); //截取token中的有用信息，返回值表示是否成功，用以判断格式是否正确  protected:  void MakeInitilate();  bool GetStr(string & src, string & des);  short int IsInt(const char \*);  short int IsFloat(char \*input); //判断参数类型  }; |

