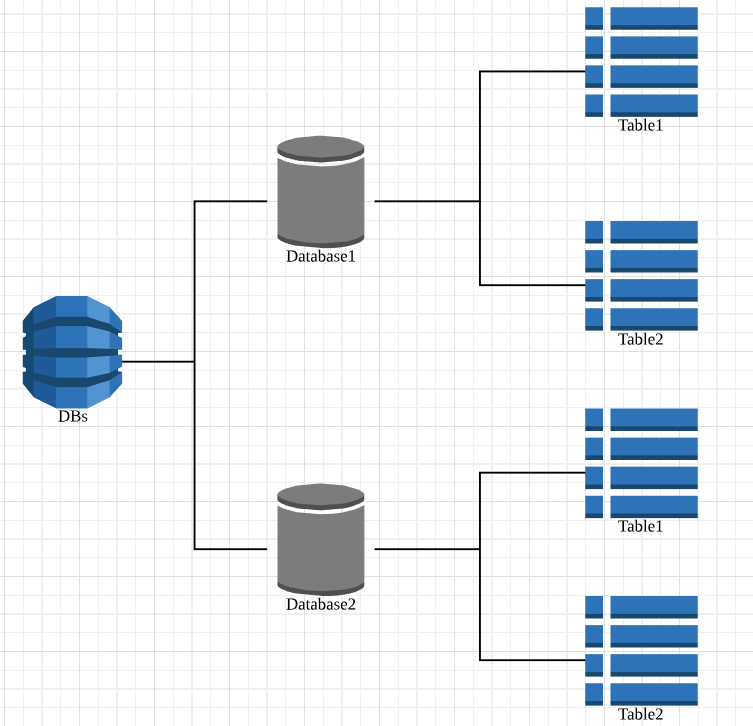
**设计文档**

1数据库系统数据存储结构

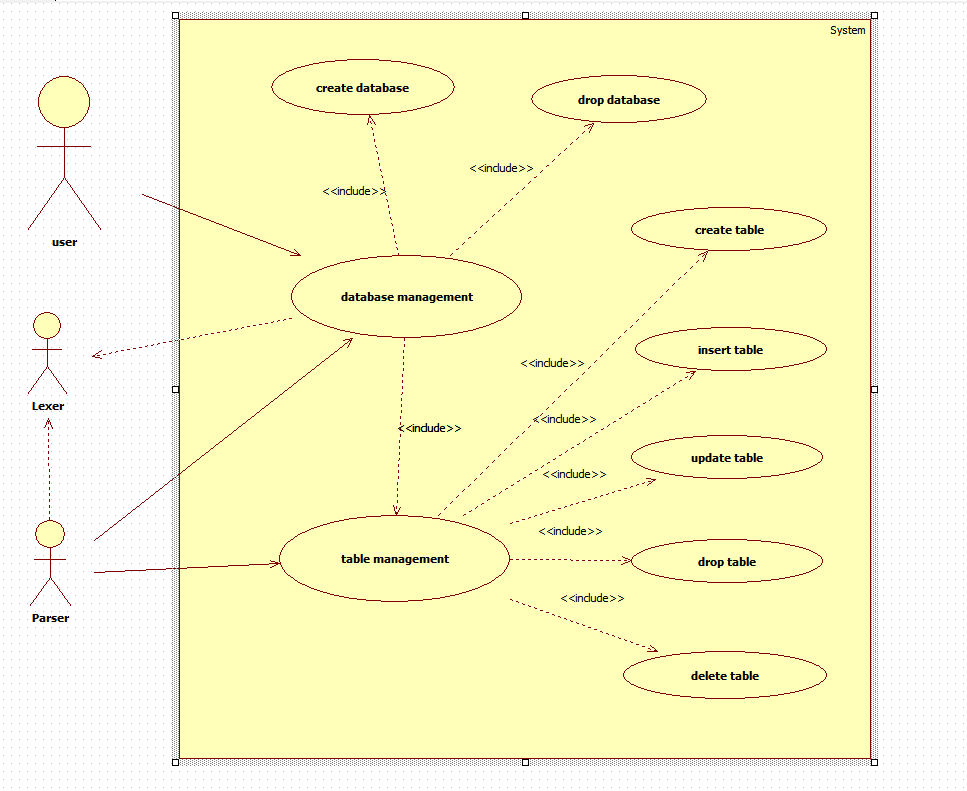
我们小组设计的内存数据库的数据存储结构如下图所示。整个数据库为一个DBs的对象，该对象里面存储由create database databaseName语句创建的database，每一个database里面存储数个由create table tableName 语句创建的table对象。每一个table对象中包含一个行对象数组和一个列对象数组，分别为vector<Row> row和vectoe<Column> column。列对象数组中的每一个列对象储存了该列的属性。行对象数组中的每个行对象中包含了一个string数组来存储由insert语句插入的数据。



数据库系统数据存储结构图

2数据库系统用例图

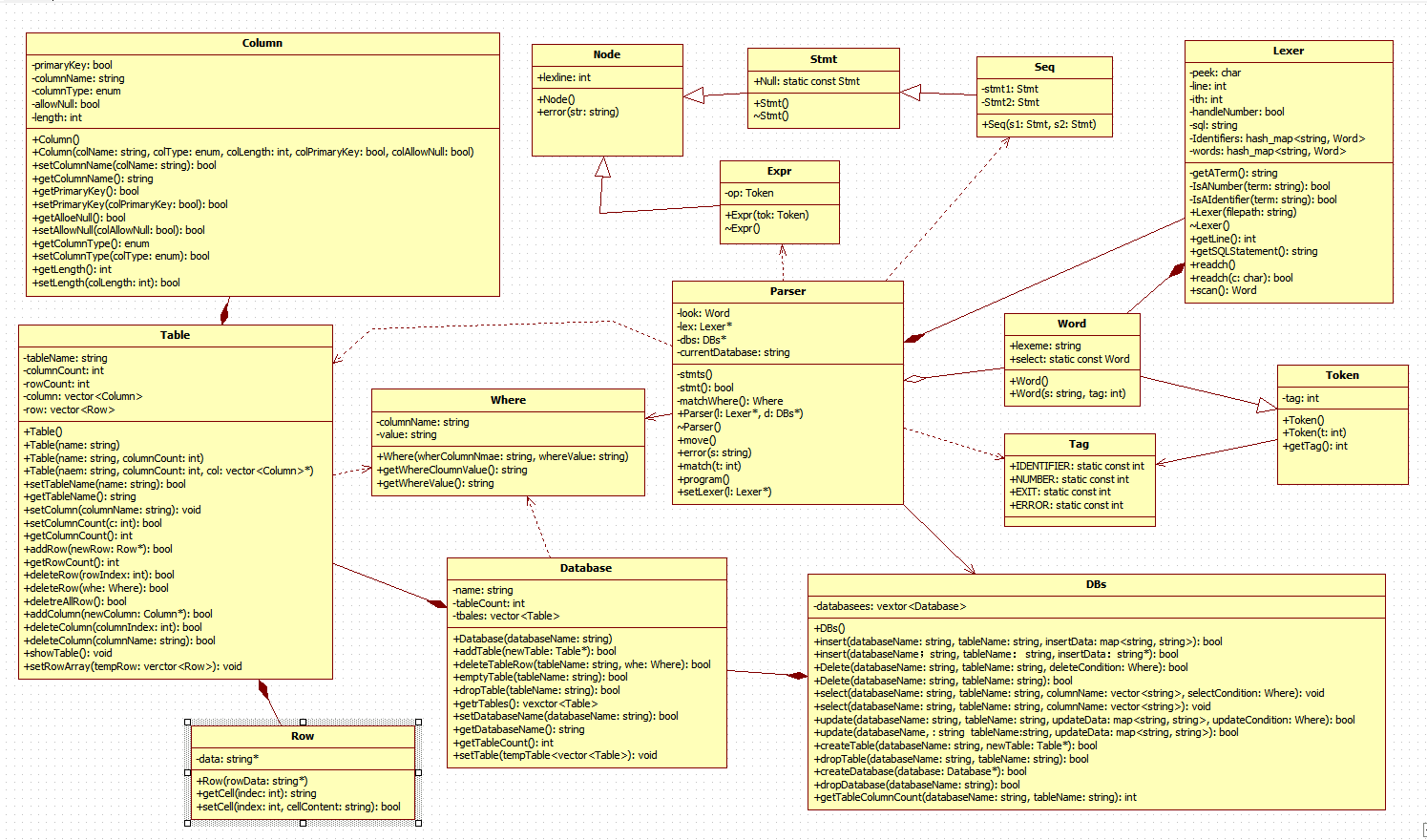
用例图展示了使用者和数据库系统的交互，用户进入系统，输入sql文件路径或者直接输入sql语句，词法分析器提取sql语句中的token并传递给语法分析器，语法分析器获取token并分析sql语句的语义，按照语义调用数据引擎的接口，实现sql语句的执行。



用例图

3数据库系统设计类图

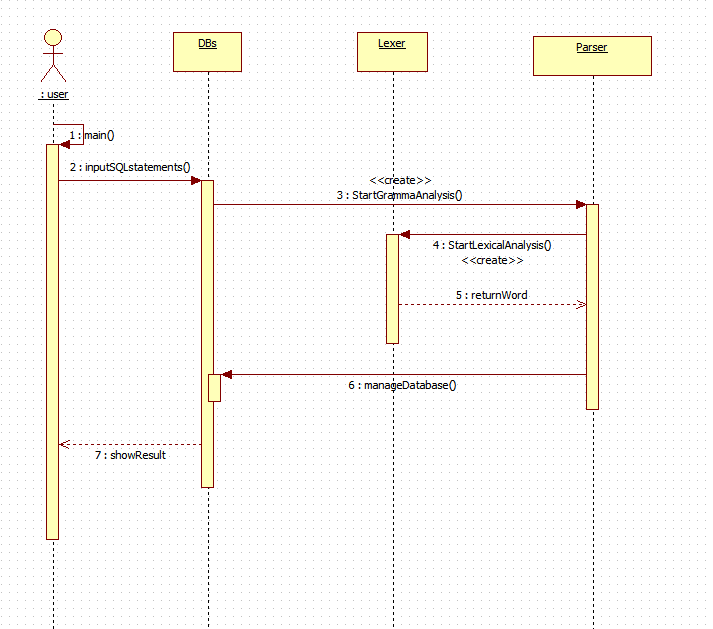
设计类图展示了数据库系统个各类的功能和类之间的相互关系，图中左下为数据引擎的类，右上为词法分析和语法分析的类。



设计类图

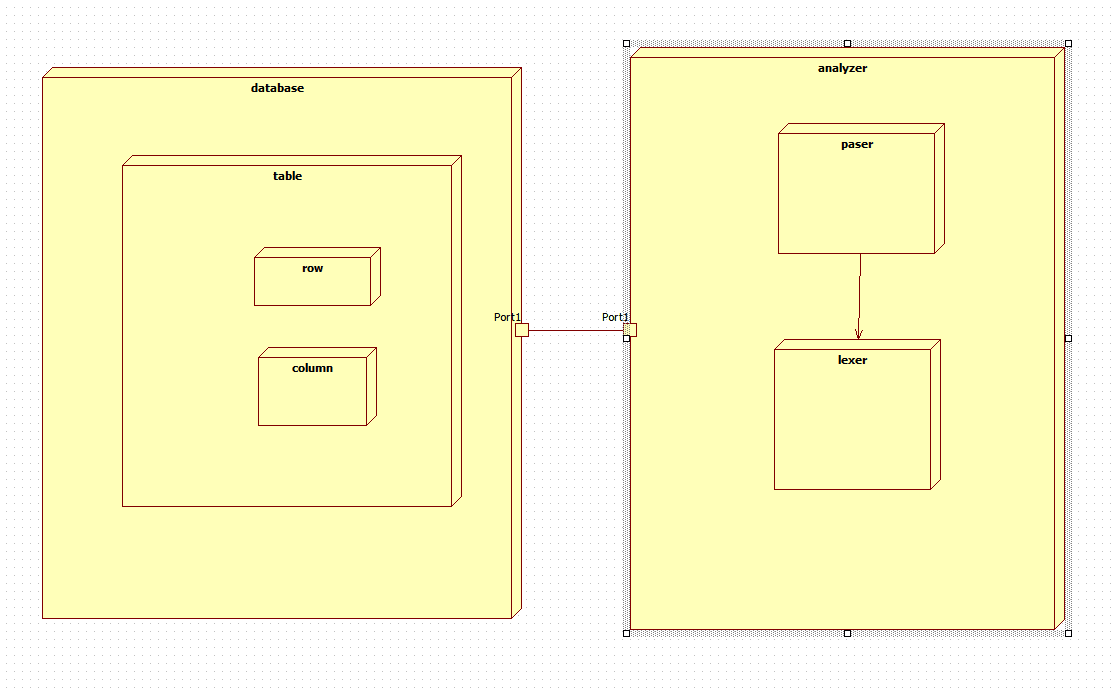
4数据库系统时序图

用户首先进入main() 函数，并输入sql语句，调用lexer和Parser，进行词法分析和语法分析，语法分析结束后，调用DBs的数据操作函数进行数据操作并肩操作结果返回给用户。



系统时序图

**5.数据库系统部署图**



系统部署图

6数据库系统功能设计

**类（结构体）**

* Database 数据库
* Table 表
* Column 列
* Row 行
* Where 语句结构体

**需要实现的SQL语句与对应接口**

1. SQL: 插入 insert into tablename(column1,column2) values (value1,value2)
   * function:
2. bool insert(String tablename, String databasename, array[])

//array[] 为关联数组 例：array[column1]=value1

1. SQL: 删除 delete from tablename where column=value
   * function:

bool delete(String tablename, String databasename, Where 结构体)

1. SQl: 选择 select column1 from tablename where column2=value
   * function:
2. Table select(String tablename, String databasename, String[], Where 结构体)

//Sring[] 数组用来储存 select 语句的参数

1. SQL: 更新 update tablename set column1=value1, column2=value2 where column3=value3
   * function:
2. bool update(String tablename, String databasename, array[], Where 结构体)

//array[] 为关联数组 例：array[column1]=value1

1. SQL: 创建表
2. create table tablename

(

colume1 type,

column2 type,

primary key(column1)

)

* + function:

bool createTable(Table 结构体)

1. SQL: 删除表 drop table tablename
   * function:

bool dropTable(String tablename, String databasename)

1. SQL: 创建数据库 create database databasename
   * function

bool createDatabase(Database 结构体)

1. SQL: 删除数据库 drop database databasename
   * function:

bool dropDatabase(String databasename)

7数据库系统词法分析详细设计

词法分析器最主要的函数就是一个scan（扫描输入字符串）函数，其他函数或数据都是为它服务的。首先，一个词法分析器应该存储有自己能识别的词类型，这存储在Identifiers和words两个hash\_map<string,word>中，并在其构造函数中就将Word类里的常量放入words中，同时判断用户的输入，在文件中读取所有sql语句，或者用户的输入就是sql语句，直接赋值给词法分析器的sql变量。构造函数结束后即可调用scan函数获取词素，每运行一次scan函数将得到一个Word类的词（包括标点、标识符、关键字等）。

scan函数是这样设计的：

1.过滤空格、回车、制表符，调用readch函数一次从sql字符串里读取一个字符存于peek（char类型），通过整数ith变量记录当前读到的sql语句的位置，读到sql结尾即把peek赋为感叹号，待后面使用。

2.经过第一步多次调用readch过滤空白字符，得到单个字符peek，判断peek是否为标点符号，一旦是标点符号即可返回响应的Word常量类，特别注意peek应该及时重置为空格！此外，lexer有一个bool变量handleNumber用于判断当前是否接受数字，在某些符号（如左括号、小于号等等）后面可以接受数字，即把handleNumber赋为true，待后面使用。

3.接下来调用getATerm函数尝试读取一个词，此函数可以读取两类此，一类是只包括数字、字母和下划线（或者只包含其中一个）的词，一类是以双引号开头和结尾的词（即表项的值），如果字符peek是感叹号，读取到的term设置为空，即sql语句读取完毕。

4.读到一个term后根据handleNumber和判断term是否为数字的函数IsANumber来判断是否应该返回一个Word(term, Tag::NUMBER)，这个Word不需要存到map数据结构中。

5.然后该判断term是否为words里已有的词了，用迭代器遍历words，如果找到了term就返回相应的words[term]，否则继续往下执行。判断term是否为合理的标识符，如果是，在Identifiers（已有关键词的列表）里查找是否有term，如有则返回，否则新建一个Word类存于Identifiers，然后将之返回。

6.最后三步，判断term是否为双引号中的内容、是否为空（返回表示退出的Word），如果都不是，则报错，将行号和term输出。扫描完毕！