西安电子科技大学

编译原理

(2018 年度)

上机报告

实验名称：DBMS 的设计与实现

班　　级：1503041

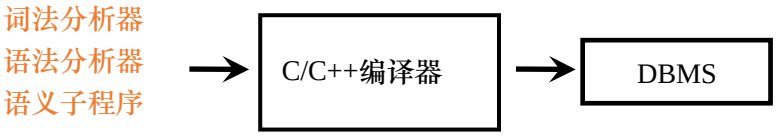
姓　　名：卢亮

学　　号：15030410015

一、实验内容

设计并实现一个DBMS原型系统，可以接受基本的SQL语句，对其进行词法分析、语法分析，然后解释执行SQL语句，完成对数据库文件的相应操作，实现DBMS的基本功能。

二、分析及设计



按照编译原理的一般思路，DBMS系统可以分解为词法部分，语法部分，语义部分，其中语义部分不仅包含对数据库操作的执行，还包含数据库底层的数据结构设计。词法部分和语法部分分别使用lex和yacc进行识别，识别的正确的SQL语句交由语义部分进行操作。

三、详细实现

1、词法分析

识别关键字

```
"use"|"USE"                 return USE;
"select"|"SELECT"           return SELECT;
"create"|"CREATE"           return CREATE;
"database"|"DATABASE"       return DATABASE;
"into"|"INTO"               return INTO;
"values"|"VALUES"           return VALUES;
"insert"|"INSERT"           return INSERT;
"from"|"FROM"               return FROM;
"table"|"TABLE"             return TABLE;
"tables"|"TABLES"           return TABLES;
"int"|"INT"|"integer"|"INTEGER"     return INTEGER;
"char"|"CHAR"|"varchar"|"VARCHAR"       return VARCHAR;
"double"|"DOUBLE"           return DOUBLE;
              //return CHAR;
"primary"|"PRIMARY"         return PRIMARY;
"key"|"KEY"                 return KEY;
"drop"|"DROP"               return DROP;
"value"|"VALUE"             return VALUE;
"exit"|"EXIT"               return QUIT;
"logico"|"LOGICO"           return LOGICO;
"show"|"SHOW"               return SHOW;
"databases"|"DATABASES"     return DATABASES;
```

识别数字、字符串和其它符号

```
[;]          return END; /*return *yytext;*/
([a-zA-Z_]+[0-9_]*)+          {yylval.strval = yytext; return OBJECT;}
[0]|([1-9][0-9]*)             {yylval.strval = yytext; return NUMBER;}
[0-9][0-9]*[\.][0-9]*         {yylval.strval = yytext; return DOUBLE;}
([\'][^']|'')*[\'])           {yylval.strval = yytext; return STRING;}
[ \t]+ {}
[,]                               return *yytext;
[)]                       {yylval.strval = yytext; return FECHA_P;}
[(]                       {yylval.strval = yytext; return ABRE_P;}
[*]                       {yylval.strval = yytext; return ALL;}
"\n" { return 0; }
```

2、语法分析

总体的语法识别

```
start    : create_database
         | QUIT END{printf("bye\n");exit(0);}
         | drop_database
         | select_database
         | show_database
         | create_table
         | show_tables
         | drop_table
         | insert
         | select
```

创建数据库

```
create_database: CREATE DATABASE {/*setMode(OP_CREATE_DATABASE);*/mode=OP_CREATE_DATABASE;}
    object END {add_database(name); return 0;}/*semicolon {return 0;}*/;
```

删除数据库

```
drop_database: DROP DATABASE {mode = OP_DROP_DATABASE;} object END {del_database(name); return 0;};
```

选择数据库

```
select_database: USE {mode = OP_SELECT;} object END {select_database(name); return 0;};
object: OBJECT {free(name);name=(char*)malloc((strlen(yytext)+1)*sizeof(char));strcpy(name, yytext);};
```

展示数据库

```
show_database: SHOW DATABASES END {show_databases();return 0;};
```

创建数据表

```
create_table: CREATE TABLE {queue_c_num=0;queue_i_num=0;queue_d_num=0;} object ABRE_P table_column_attr FECHA_P END {add_table(); return 0;};
table_column_attr: column_create type /*attribute*/ | column_create type /*attribute*/ ',' table_column_attr;
type: INTEGER {/*setColumnTypeCreate('I');*/ queue_i[queue_i_num++] = 11;}
    | VARCHAR {/*setColumnTypeCreate('S');*/ queue_i[queue_i_num++] = 12;} ABRE_P NUMBER {/*setColumnSizeCreate(yylval.strval);*/} FECHA_P
    | DOUBLE {/*setColumnTypeCreate('D');*/ queue_i[queue_i_num++] = 13;};
    //| CHAR {setColumnTypeCreate('C');};
column_create: OBJECT {/*setColumnCreate(yytext);*/strcpy(queue_c[queue_c_num++], yytext);};  //列名和表名不能用同一个变量
```

展示数据表

```
show_tables: SHOW TABLES END {show_tables();return 0;};
```

删除数据表

```
drop_table: DROP TABLE object END {del_table(name); return 0;};
```

插入数据

```
insert:                    INSERT                    INTO
{/*setMode(OP_INSERT);*/queue_c_num=0;queue_i_num=0;queue_d_num=0;}
        OBJECT                                    {free(name);name=(char
*)malloc((strlen(yytext)+1)*sizeof(char));strcpy(name,    yytext);table_now    =
(database_now == NULL) ? NULL:find_table(name);}
        opt_column_list       VALUES       ABRE_P       value_list       FECHA_P
END{insert_to_table();return 0;};
opt_column_list:   /*optional*/{int   i;   if(table_now   !=NULL)   for   (i=0;
i<table_now->column_num; i++) queue_i[queue_i_num++]=i;}
               | ABRE_P column_list FECHA_P;
value_list: value | value ',' value_list;
value: DOUBLE {strcpy(queue_c[queue_c_num++], yytext);}
     | NUMBER {strcpy(queue_c[queue_c_num++], yytext);}
     | STRING {strcpy(queue_c[queue_c_num++], yytext);};
column_list: column | column ',' column_list;
column: OBJECT {queue_i[queue_i_num++] = get_column_position(yytext);};
```

查询数据

```
select: SELECT {queue_c_num=0;queue_i_num=0;queue_d_num=0;flag=0;} selection
        FROM table_select /*where*/ END {show_items(); return 0;};
table_select:                  OBJECT                    {free(name);name=(char
*)malloc((strlen(yytext)+1)*sizeof(char));strcpy(name,    yytext);table_now    =
(database_now == NULL) ? NULL:find_table(name);transis();};
selection: ALL {flag=1;}
         |  OBJECT {strcpy(queue_c[queue_c_num++], yytext);} selection2
         //|  ABRE_P OBJECT {adcProjSelect(yylval.strval);} selection2 FECHA_P
         ;
selection2: /* epsilon */
          | ',' OBJECT {strcpy(queue_c[queue_c_num++], yytext);} selection2;
```

3、语义制导

数据库的数据结构

```
typedef struct database
{
    char *name;
    table *table_first, *table_last;
    struct database *next;
}database;
```

数据表的数据结构

```
typedef struct table
{
    char *table_name;
    int column_num;
    char **column_name;
    int **column_type;
    item *item_first, *item_last;
    struct table *next;
}table;
```

数据项的数据结构

```
union content { int i; double d; char c[30]; }integer, fraction, string;

typedef struct item
{
    union content *it;
    struct item *next;
}item;
```

部分数据库操作

```
int database_init();

database *find_database (char *name);

void select_database(char *name);

void add_database(char *name);

void del_database(char *name);

void show_databases();

void add_table();

void init_table(table *table);

void show_tables();
```

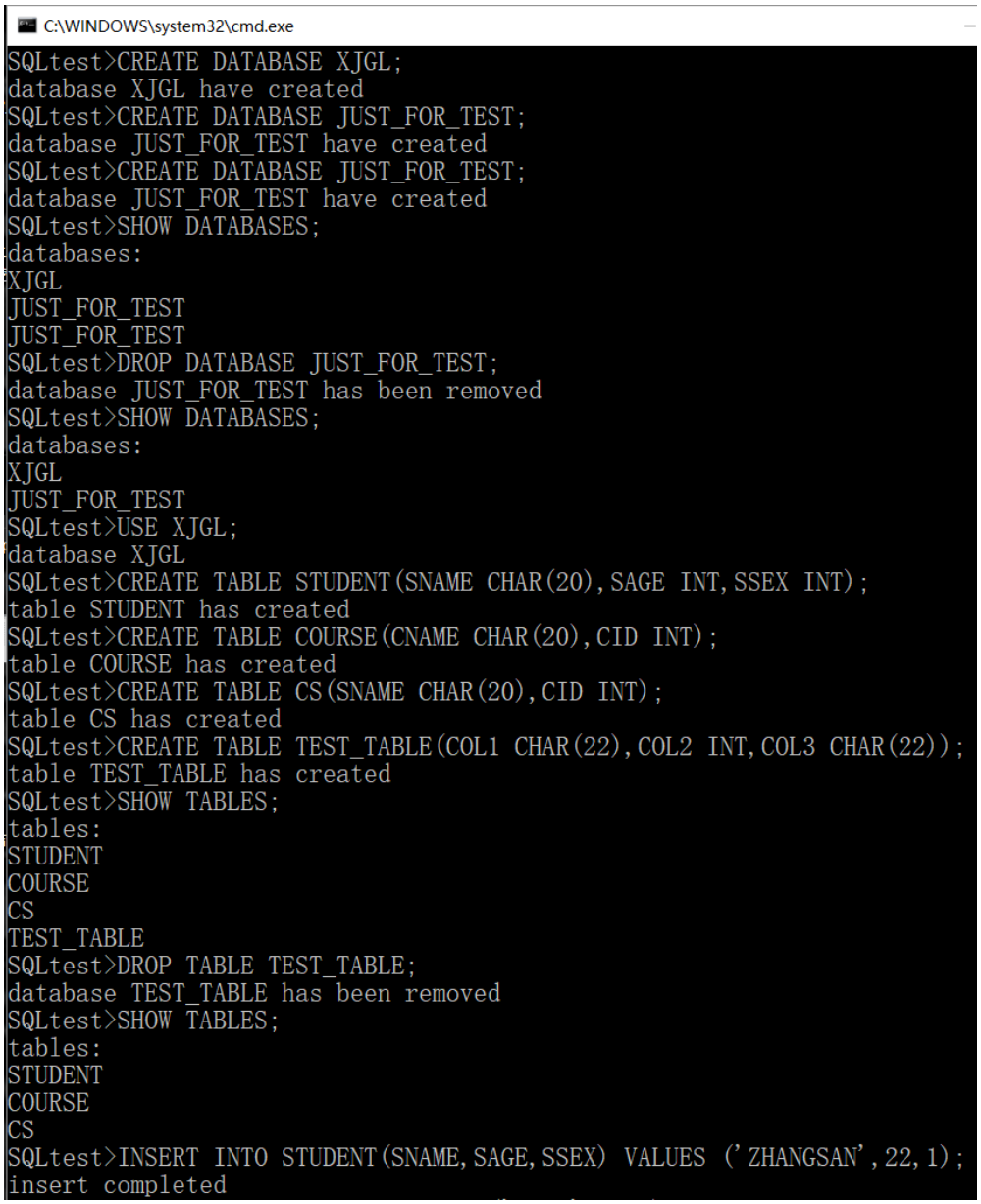
四、实验结果

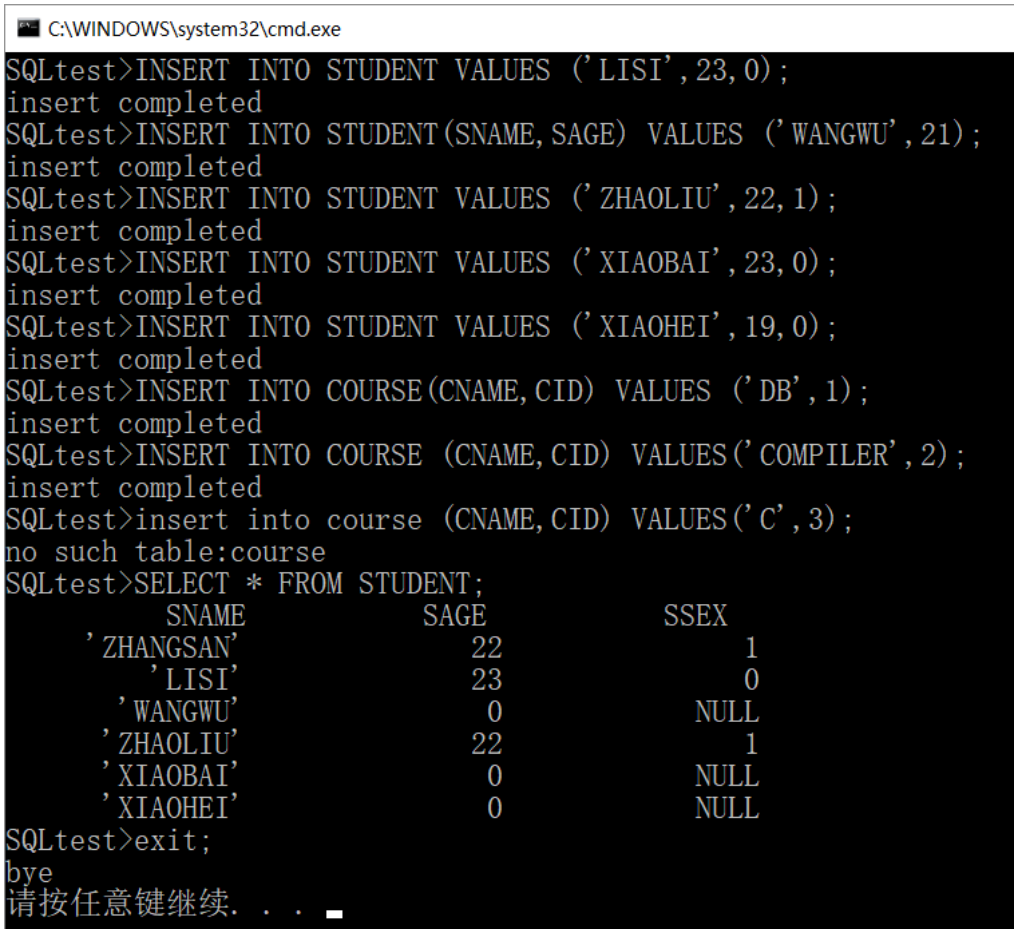
实现了词法分析和语法分析，以及部分的语义制导，其中包括数据库的添加、删除、选择、展示，表的添加、删除、展示，能向表中插入数据，并能进行简单查询，实现了数据库的正常退出，对于语法上正确而逻辑上不可行的语句能给出相应的提示。

由于时间不足，条件查询和数据更新两部分的语义没有完成，但是在底层数据结构里已经适配了不同类型的数据，能够满足条件查询和数据更新的要求。此外没有对错误输入进行反馈，而是选择让程序结束，后期有机会会处理这个问题。

另外，限制了数据库单个字段最大长度为30，单次查询、更新、插入等数据操作能处理的字段数也为30，这是由于实现的命令缓存队列只有这么长，这写没有显式在系统中说明。

部分语句测试


C:\WINDOWS\system32\cmd.exe
SQLtest>CREATE DATABASE XJGL;
database XJGL have created
SQLtest>CREATE DATABASE JUST_FOR_TEST;
database JUST_FOR_TEST have created
SQLtest>CREATE DATABASE JUST_FOR_TEST;
database JUST_FOR_TEST have created
SQLtest>SHOW DATABASES;
databases:
XJGL
JUST_FOR_TEST
JUST_FOR_TEST
SQLtest>DROP DATABASE JUST_FOR_TEST;
database JUST_FOR_TEST has been removed
SQLtest>SHOW DATABASES;
databases:
XJGL
JUST_FOR_TEST
SQLtest>USE XJGL;
database XJGL
SQLtest>CREATE TABLE STUDENT(SNAME CHAR(20),SAGE INT,SSEX INT);
table STUDENT has created
SQLtest>CREATE TABLE COURSE(CNAME CHAR(20),CID INT);
table COURSE has created
SQLtest>CREATE TABLE CS(SNAME CHAR(20),CID INT);
table CS has created
SQLtest>CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22));
table TEST_TABLE has created
SQLtest>SHOW TABLES;
tables:
STUDENT
COURSE
CS
TEST_TABLE
SQLtest>DROP TABLE TEST_TABLE;
database TEST_TABLE has been removed
SQLtest>SHOW TABLES;
tables:
STUDENT
COURSE
CS
SQLtest>INSERT INTO STUDENT(SNAME,SAGE,SSEX) VALUES ('ZHANGSAN',22,1);
insert completed



五、心得体会

通过这次实验的经历，我对编译的整个过程有了更深的了解，熟悉了正则表达式和产生式，C语言的编程能力也得到了进一步锻炼，尤其是在链表、共用体方面有了大的进步，实验过程中在网上查找相关资料也提高了我的信息检索能力，总的来说对自己能做到这个程度比较满意。