编译原理上机报告

《DBMS 的设计与实现》

姓名：petite7

邮箱：petite7@vip.qq.com

完成时间：　2018　年　6 月　　20　日

目　　录

1. 项目概况

1.1 基本目标

使用 Lex 与 Yacc 编写一个识别简单 SQL 语法的 DBMS 系统，实现对 SQL 词法和语法的分析，并添加相应的语义动作和物理结构，以体现 SQL 的基本功能。将 ParserGenerator 产生的相应头文件和库文件添加到 Visual Studio 2017 项目中，编译产生可执行文件，测试语句执行的正确性。

项目文件依赖关系如下：

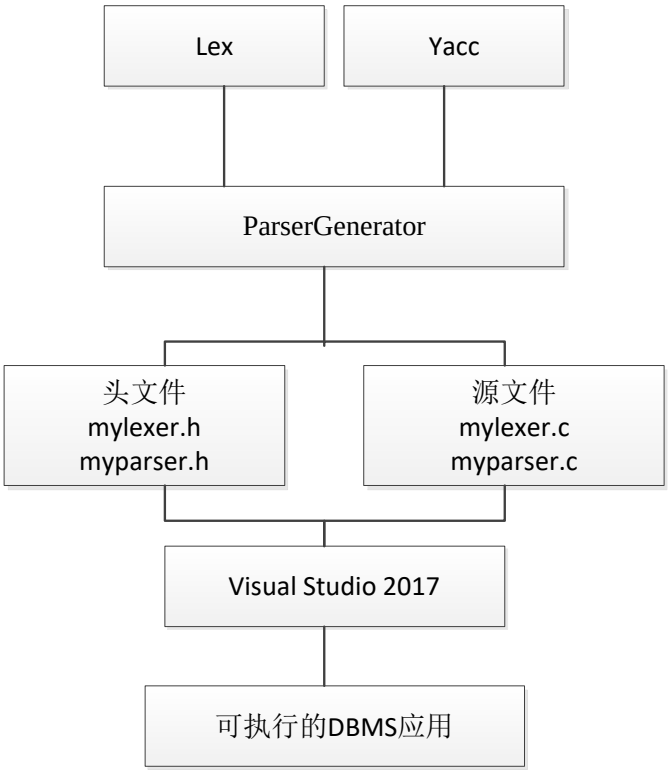


图 1：项目文件依赖关系

1.2 完成情况

>已实现语句：

操作对象	实现语句
DATABASE	CREATE DATABASE SHOW DATABASE DROP DATABASE USE DATABASE
TABLE	CREATE TABLE SHOW TABLE DROP TABLE
SELECT	SELECT * FROM TABLES SELECT * FROM TABLES WHERE CONDITIONS SELECT FIELDS FROM TABLES SELECT FIELDS FROM TABLES WHERE CONDITIONS
INSERT	INSERT INTO TABLE VALUES INSERT INTO TABLE(FIELDS) VALUES(VALUES)
UPDATE	UPDATE TABLE SET VALUES UPDATE TABLE SET VALUES WHERE CONDITIONS
DELETE	DELETE FROM TABLES WHERE CONDITIONS

2. 项目实现方案

2.1 逻辑结构与物理结构

建立在文件系统上的物理逻辑结构如下图 1 所示，主目录下包含一个 sys.dat 文件和若干文件夹。sys.dat 内每行数据代表一个数据库的名称；每个文件夹代表对应文件夹名的数据库。代表数据库的文件夹内同样包含一个 sys.dat 文件和若干个 txt 文本文档。sys.dat 内每行存储的表名以及字段信息如下表 1 所示；每个文本文档代表对应文件名的表数据，每条记录占用一行，数据之间以空格分隔。

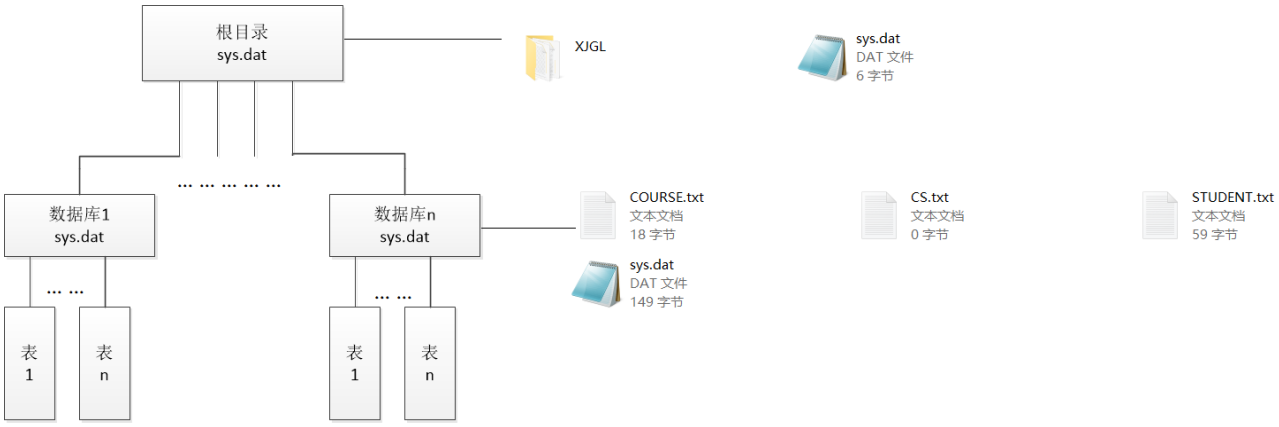


图 1：文件系统上的物理结构和逻辑结构

表名	列号	字段名	字段类型	长度
STUDENT	1	Sno	CHAR	20
STUDENT	2	Sname	CHAR	20
…				

表 1：数据库表内记录的表文件信息

使用这种物理结构的**优点**是能够简单方便地访问到对应的数据库和表，不论是对程序还是用户来说，访问都是简单清晰的。但是这种物理结构的**缺点**也是显而易见的，首先数据库和表数据都以明文和可见数据的形式暴露，容易被

破坏和修改，其次文件结构是简单的线性结构，在数据量庞大的情况下，程序执行的效率将十分低。

2.2 语法结构与数据结构

①CREATE 语句的产生式语法结构：

```
createsql:CREATE TABLE table '(' fieldsdefinition ')' ';'
      |CREATE DATABASE ID ';'
       table:ID;
       fieldsdefinition:field_type
      |field_type ',' fieldsdefinition;
       field_type:field type;
       field:ID;
       type:CHAR '(' NUMBER ')'
      |INT;
```

CREATE 语句的非终结符结构体：

其中：Createfieldsdef 为 CREATE 语句中字段的定义；

Createstruct 为 CREATE 语法树的根节点；

fieldType 为字段类型的定义。

```
struct Createfieldsdef{
    char        *field;
    enum TYPE   type;
    int         length;
    struct  Createfieldsdef *next_fdef;
};
struct fieldType{
    enum TYPE   type;
    int     length;
};

struct Createstruct{
    char *table;
    struct  Createfieldsdef *fdef;
};
```

实例：

```
CREATE TABLE Student( Sno CHAR(9), Sname CHAR(20), Ssex CHAR(2), Sage INT );
```

Student 将以根节点结构体表示，随后的 Sno，Sname， Ssex，Sage 都以表字段结构体表示。

对应的数据结构如下图 2 所示：

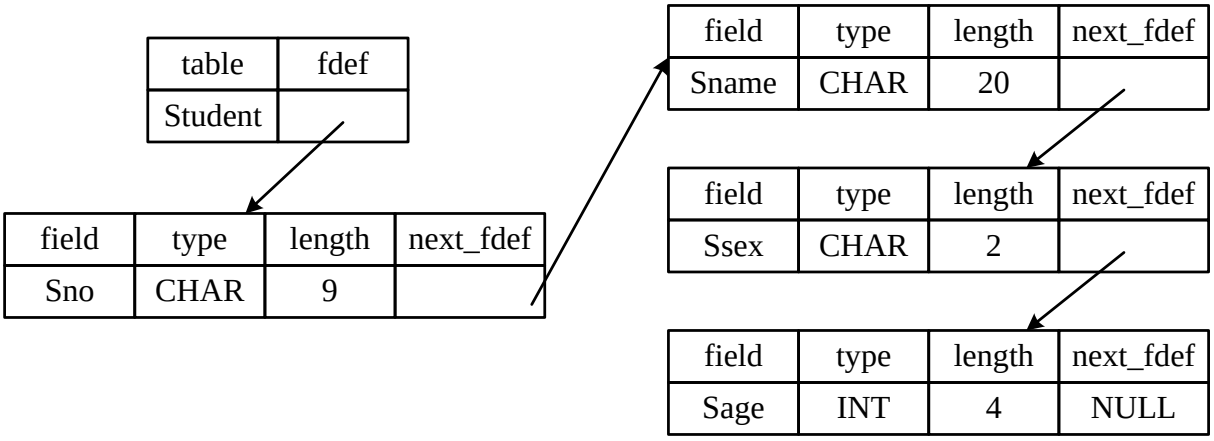


图 2 数据结构图

②SELECT 语句的产生式语法结构：

```
selectsql:SELECT fields_star FROM tables ';'
      |SELECT fields_star FROM tables WHERE conditions ';';
       fields_star:table_fields
      |'*';
       table_fields:table_field
      |table_field ',' table_fields;
       table_field:field
      |table '.' field;
       tables:table
      |table ',' tables;
       conditions:condition
      |'('conditions')'
      |conditions AND conditions
      |conditions OR conditions;
      condition:comp_unit comp_op comp_unit;
      comp_unit:table_field
      |NUMBER
      |QUOTE ID QUOTE
      |QUOTE NUMBER QUOTE
      comp_op:'<' |'>' |'='    |'!''=';
```

SELECT 语句的非终结符结构体：

其中：Selectedfields 为 SELECT 语句中选中字段的定义；

Seletedtables 为 SELECT 语句中选中表的定义；

Selectstruct 为 SELECT 语句中根节点的定义；

Conditions 为各类语句中出现的条件判断树结构体的定义。

```
struct  Selectedfields{
    char    *table;
    char    *field;
    struct  Selectedfields  *next_sf;
};
struct  Selectedtables{
    char      *table;
    struct  Selectedtables  *next_st;
};
struct  Selectstruct{
    struct Selectedfields   *sf;
    struct Selectedtables   *st;
    struct Conditions   *cons;
};
struct Conditions{
    struct Conditions *left;
    struct Conditions *right;
    char comp_op;
    int type;
    char *value;
    char *table;
};
```

实例：

```
SELECT Sno, Sname FROM student WHERE Ssex='男' AND Sage=20;
```

Sno 与 Sname 字段将存入代表选中字段的非终结符结构体中，student 将存入代表已选中表的非终结符结构体中，Ssex='男' AND Sage=20 将存入条件判断树中。

对应的数据结构如下图 3 所示：

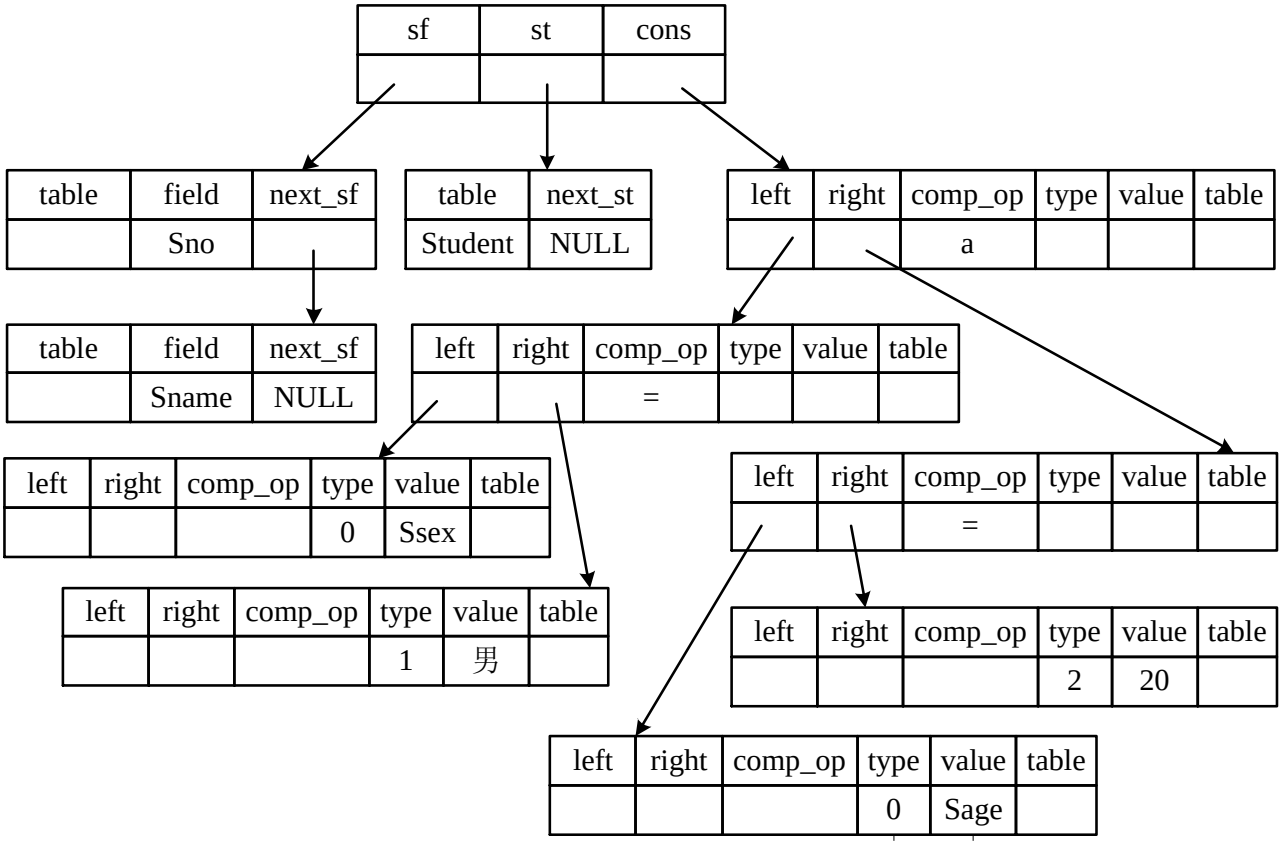


图 3 数据结构图

③INSERT 语句的产生式语法结构：

```
insertsql:INSERT INTO table VALUES '(' values ')' ';'
      |INSERT INTO table '(' values ')' VALUES '(' values ')' ';'  ;
       values:value
      |value ',' values  ;
       value:QUOTE ID QUOTE
      |QUOTE NUMBER QUOTE
      |NUMBER
      |ID;
```

INSERT 语句的非终结符结构体：

其中：insertValue 为将要插入记录的链表结构体。

```
struct insertValue {
    char *value;
    struct insertValue *next_Value;
};
```

实例：

```
1.  INSERT INTO STUDENT(SNAME,SAGE,SSEX) VALUES ('ZHANGSAN',22,1);
```

STUDENT 被加入代表 table 表的结构体中，被选中的字段 SNAME，SAGE，SSEX 和将要插入的记录’ZHANGSAN’， 21， 1 都将被加入键值链表中。

对应的数据结构如下图 4 所示：

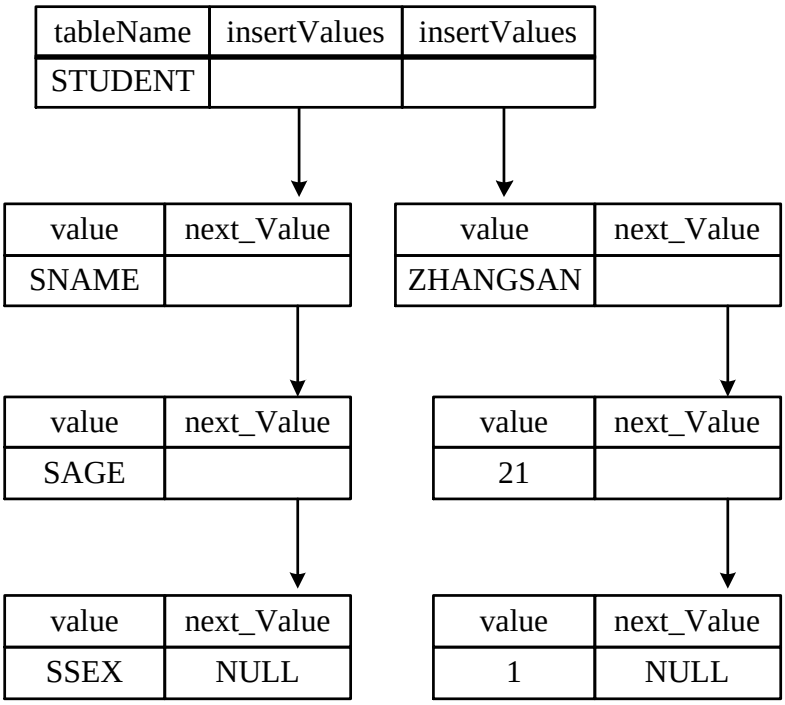


图 4 数据结构图

④UPDATE 语句的产生式语法结构：

```
1.  updatesql:UPDATE table SET sets ';'
2.        |UPDATE table SET sets WHERE conditions ';'  ;
3.         sets:set
4.        |set ',' sets  ;
5.         set:ID '=' NUMBER
6.        |ID '=' QUOTE ID QUOTE  ;
```

UPDATE 语句的非终结符结构体：

其中：Updatestruct 为 UPDATE 语句中更新字段和值的链表。

```
struct Updatestruct{
    char *field;
    char *value;
    struct Updatestruct *next_sf;
};
```

实例：

```
UPDATE STUDENT SET SAGE=27,SSEX=1 WHERE SNAME='ZHANGSAN';
```

STUDENT 将被加入更新表名字符串中，字段名 SAGE，SSEX 以及更新键值 27，1 将被加入更新链表中；SNAME='ZHANGSAN'将被加入 Condition 条件树中。

对应的数据结构如下图 5 所示：

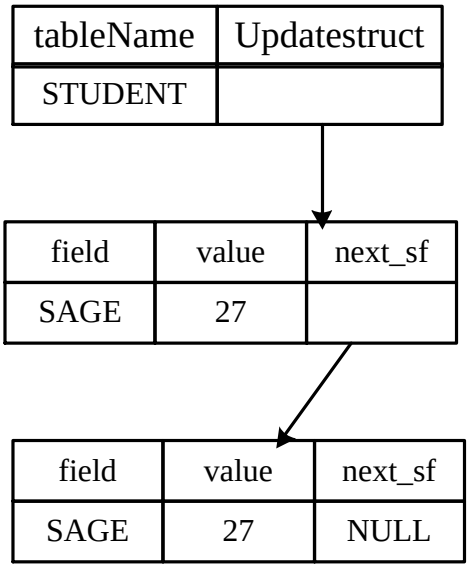


图 5 数据结构图

④DELETE 语句的产生式语法结构：

```
deletesql:DELETE FROM table ';'
    |DELETE FROM table WHERE conditions ';';
```

DELETE 语句本身没有相应的结构体支持，相关语义动作将要删除的表名和 Condition 条件加入相关函数并删除。

实例：

```
DELETE FROM STUDENT WHERE (SAGE>21) AND (SSEX=0);
```

表名 STUDENT 将作为字符串，条件(SAGE>21) AND (SSEX=0)将被加入条件树

⑤其余不需要相应数据结构支持的语句产生式

DROP

```
dropsql:DROP TABLE ID ';'
        |DROP DATABASE ID ';';

```

SHOW

```
showsql :SHOW DATABASES ';'
     |SHOW TABLES ';';
```

USE

```
usesql:USE ID ';';
```

2.3 执行流程

SQL 语句执行需要调用的函数表一览：

CREATE	void createDataBase(); void createTable(struct Createstruct *createList);
SELECT	void selectPart(struct Selectedfields *field, struct Selectedtables *table); void selectPartCondition(struct Selectedfields *field, struct Selectedtables *table, struct Conditions *condition);
INSERT	void insertALL(char *tableName, struct insertValue *values); void insertPart(char *tableName, struct insertValue *valueName, struct insertValue *values);
UPDATE	void updateCondition(char *tableName, struct Updatestruct *valueList, struct Conditions *condition);
DELETE	void deleteCondition(char *tableName, struct Conditions *condition);
SHOW	void showDataBase(); void showTable();
DROP	void dropDataBase(); void dropTable(char *tableName);
USE	void useDataBase();

函数名称：createDataBase()

函数说明：根据全局变量 baseName 建立数据库

输入参数：无

输出参数：void

执行流程：函数首先从全局变量中读取要建立的数据库名，检查已有数据库中是否有同名数据库，若没有，则在 sys.dat 中写入数据库名，并创建同名文件夹。

函数名称：createTable(struct Createstruct *createList)

函数说明：在数据库目录下创建一个表

输入参数：CREATE 语句根节点指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，检查当前数据库下是否有头结点中包含的表名，如果没有，则将头结点下移，将CREATE链表中的字段逐一根据表定义写入sys.dat中，直到指针为NULL；然后在文件夹下创建同名文本文档。

函数名称：selectPart(struct Selectedfields *field, struct Selectedtables *table)

函数说明：根据输入字段和表查询记录，SELECT 语句中没有添加条件

输入参数：已选字段结构体指针，已选表结构体指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，然后将已选表中的表名保存到缓存，对于每一个表，扫描 sys.dat 中与已选结构体中相互匹配的字段，记录字段所在列，再根据列序号访问相应文本文档中对于列的数据，输出到控制台上，直到文件结尾。

函数名称：getCondition(struct Conditions *now, char fieldName[][maxn], char fieldValue[][maxn])

函数说明：判断输入的记录值是否和 condition 一致

输入参数：条件树头指针，要判断的表字段，当前记录

输出参数：BOOL，若条件为真，输出 true，否则输出 false

执行流程：对于当前条件树，如果指针指向的不是条件树的叶子节点，则将指针下移，进入下一个递归。当条件树指针到达叶子节点时，判断输入记录是否和条件树中当前键值匹配，是返回 true，否则返回 false；

函数名称：selectPartCondition(struct Selectedfields *field, struct Selectedtables *table, struct Conditions *condition)

函数说明：根据输入字段和表以及 SELECT 中的条件查询记录

输入参数：已选字段结构体指针，已选表结构体指针，条件树根节点指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，然后将已选表中的表名保存到缓存，对于每一个表，扫描 sys.dat 中与已选结构体中相互匹配的字段和所有字段集，然后读取对于文本文档内的每一列，对于每一行记录，将所有字段和这行记录以及条件树传入 getCondition 函数中，若函数返回 true，则将该行的对应列输出到控制台，直到文件结尾。

函数名称：insertALL(char *tableName, struct insertValue *values)

函数说明：执行 INSERT 语句插入所有键值

输入参数：表名字符串，插入值链表头指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，通过输入参数检查当前表的可达性之后，将输入链表中的参数逐一输入到对应的文本文档中，直到指针为 NULL。

函数名称：insertPart(char *tableName, struct insertValue *valueName, struct insertValue *values)

函数说明：执行 INSERT 语句插入对应键值

输入参数：表名字符串，插入字段链表头指针，插入值链表头指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，通过输入参数检查当前表的可达性之后，读取字段链表中的数据，并从 sys.dat 中查询对应字段所在列号，保存到缓存，然后将输入链表中与行号匹配的参数逐一输入到对应的文本文档中，不匹配的列号写入 NULL，直到指针为 NULL。

函数名称：updateCondition(char *tableName, struct Updatestruct *valueList, struct Conditions *condition)

函数说明：根据条件更新表数据

输入参数：表名字符串，更新字段链表头指针，条件树头指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，通过输入参数检查当前表的可达性之后，读取更新字段链表中的数据，从 sys.dat 中获得匹配的列号，然后在相应文本文档中逐一读取每行记录，并将数据缓存和条件树作为参数调用 getCondition 函数，若返回值为 true，则将链表键值更新到对应列中，直到文件结尾。

函数名称：deleteCondition(char *tableName, struct Conditions *condition)

函数说明：根据条件删除表记录

输入参数：表名字符串，条件树头指针

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库

中，通过输入参数检查当前表的可达性之后，读取更新字段链表中的数据，从 sys.dat 中获得该表所有字段，然后在相应文本文档中逐一读取每行记录，并将数据缓存和条件树作为参数调用 getCondition 函数，若返回值为 true，则将表中数据删除，直到文件结尾。

函数名称：showDataBase()

函数说明：打印所有数据库

输入参数：无

输出参数：void

执行流程：函数读取根目录下 sys.dat 中的数据库数据，并将每行数据输出，直到文件结束。

函数名称：showTable()

函数说明：打印数据库下的所有表字段

输入参数：无

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，读取数据库目录下 sys.dat 中的数据库数据，并将每行数据输出，直到文件结束。

函数名称：dropDataBase()

函数说明：删除全局变量 baseName 指定的数据库

输入参数：无

输出参数：void

执行流程：函数读取全局变量 baseName，检查根目录下是否存在同名的文件夹，若存在，则从 sys.dat 中删除匹配的数据库名，同时删除根目录下匹配的文件夹。

函数名称：dropTable(char *tableName)

函数说明：删除指定表

输入参数：无

输出参数：void

执行流程：函数首先根据全局变量 baseName 切换到当前已 USE 的数据库中，检查目录下是否存在与输入参数同名的文本文档，若存在，则从 sys.dat 中删除匹配的表名，同时删除目录下匹配的文本文档。

函数名称：useDataBase()

函数说明：根据语义动作输入修改全局变量 baseName

输入参数：无

输出参数：void

执行流程：函数根据语义动作修改的 baseName，检查 baseName 对应数据库的可达性。

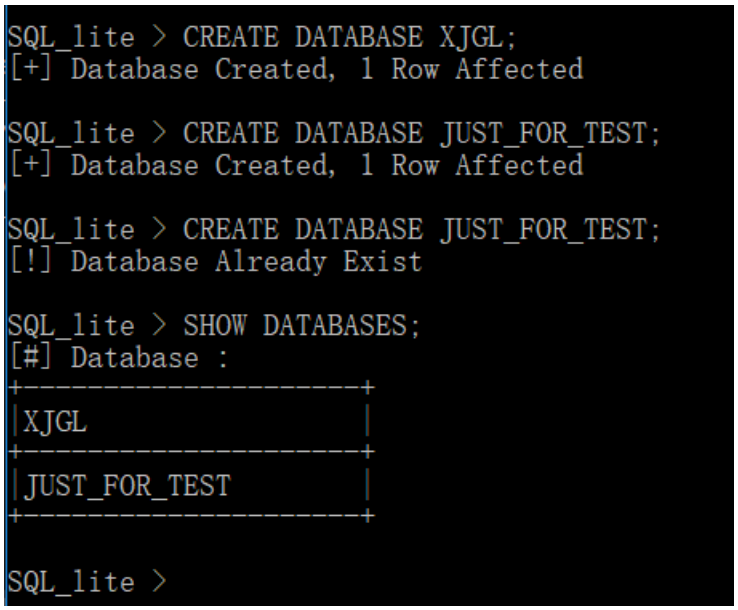
2.4 功能测试

测试 1

输入：

```
1. CREATE DATABASE XJGL;
2. CREATE DATABASE JUST_FOR_TEST;
3. CREATE DATABASE JUST_FOR_TEST;
4. SHOW DATABASES;
```

输出：成功创建数据库，并拒绝了重复创建



测试 2

输入：

```
1. CREATE TABLE STUDENT(SNAME CHAR(20),SAGE INT,SSEX INT);CREATE TABLE COURSE(CNAME CHAR(20),
   CID INT);
2. CREATE TABLE CS(SNAME CHAR(20),CID INT);
3. CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22);
4. CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22);
5. SHOW TABLES;
6. DROP TABLE TEST_TABLE;
7. SHOW TABLES;
```

输出 1：成功识别测试用例中的语法错误

```
SQL_lite > USE XJGL;
[#] Using Database : XJGL

SQL_lite > CREATE TABLE STUDENT(SNAME CHAR(20),SAGE INT,SSEX INT);CREATE TABLE COURSE(CNAME CHAR(20),CID INT);
[#] Table [STUDENT] Created With 3 Properties

SQL_lite > [#] Table [COURSE] Created With 2 Properties

SQL_lite > CREATE TABLE CS(SNAME CHAR(20),CID INT);
[#] Table [CS] Created With 2 Properties

SQL_lite > CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22);
syntax error at ->  ;

SQL_lite > syntax error at ->


SQL_lite > CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22);
syntax error at ->  ;

SQL_lite > syntax error at ->
```

输出 2：成功查询所有数据库

```
SQL_lite > SHOW TABLES;
[#] Tables In Database [XJGL]
+-----------------------------------------------------------------+
|STUDENT                          |1   |SNAME          |CHAR|20   |
+-----------------------------------------------------------------+
|STUDENT                          |2   |SAGE           |INT |4    |
+-----------------------------------------------------------------+
|STUDENT                          |3   |SSEX           |INT |4    |
+-----------------------------------------------------------------+
|COURSE                           |1   |CNAME          |CHAR|20   |
+-----------------------------------------------------------------+
|COURSE                           |2   |CID            |INT |4    |
+-----------------------------------------------------------------+
|CS                               |1   |SNAME          |CHAR|20   |
+-----------------------------------------------------------------+
|CS                               |2   |CID            |INT |4    |
+-----------------------------------------------------------------+
```

测试 3：

输入：

```
1.  INSERT INTO STUDENT(SNAME,SAGE,SSEX) VALUES ('ZHANGSAN',22,1);
2.  INSERT INTO STUDENT VALUES ('LISI',23,0);
3.  INSERT INTO STUDENT(SNAME,SAGE) VALUES ('WANGWU',21);
4.  INSERT INTO STUDENT VALUES ('ZHAOLIU',22,1);
5.  INSERT INTO STUDENT VALUES ('XIAOBAI',23,0);
6.  INSERT INTO STUDENT VALUES ('XIAOHEI',19,0);
7.  INSERT INTO COURSE(CNAME,CID) VALUES ('DB',1);
8.  INSERT INTO COURSE (CNAME,CID) VALUES('COMPILER',2);
9.  insert into course (CNAME,CID) VALUES('C',3);
```

输出：所有记录添加成功

```
SQL_lite > INSERT INTO STUDENT(SNAME,SAGE,SSEX) VALUES ('ZHANGSAN',22,1);
[#] Insert Into Table [STUDENT], 1 row Affected

SQL_lite > INSERT INTO STUDENT VALUES ('LISI',23,0);
[#] Insert Into Table [STUDENT], 1 row Affected

SQL_lite > INSERT INTO STUDENT(SNAME,SAGE) VALUES ('WANGWU',21);
[#] Insert Into Table [STUDENT], 1 row Affected

SQL_lite > INSERT INTO STUDENT VALUES ('ZHAOLIU',22,1);
[#] Insert Into Table [STUDENT], 1 row Affected

SQL_lite > INSERT INTO STUDENT VALUES ('XIAOBAI',23,0);
[#] Insert Into Table [STUDENT], 1 row Affected

SQL_lite > INSERT INTO STUDENT VALUES ('XIAOHEI',19,0);
[#] Insert Into Table [STUDENT], 1 row Affected

SQL_lite > INSERT INTO COURSE(CNAME,CID) VALUES ('DB',1);
[#] Insert Into Table [COURSE], 1 row Affected

SQL_lite > INSERT INTO COURSE (CNAME,CID) VALUES('COMPILER',2);
[#] Insert Into Table [COURSE], 1 row Affected
```

测试 4：

输入：

```
1.  SELECT SNAME,SAGE,SSEX FROM STUDENT;
2.  SELECT SNAME,SAGE FROM STUDENT;
3.  SELECT * FROM STUDENT;
4.  SELECT SNAME,SAGE FROM STUDENT WHERE SAGE=21;
5.  SELECT SNAME,SAGE FROM STUDENT WHERE (((SAGE=21)));
6.  SELECT SNAME,SAGE FROM STUDENT WHERE (SAGE>21) AND (SSEX=0);
7.  SELECT SNAME,SAGE FROM STUDENT WHERE (SAGE>21) OR (SSEX=0);
8.  SELECT * FROM STUDENT WHERE SSEX!=1;
```

输出：查询结果全部正确

```
SQL_lite > SELECT SNAME,SAGE,SSEX FROM STUDENT;
[#] Selected From Table [STUDENT]
+-----------------------------------------------------+
|ZHANGSAN   |22         |1          |
+-----------------------------------------------------+
|LISI       |23         |0          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|ZHAOLIU    |22         |1          |
+-----------------------------------------------------+
|XIAOBAI    |23         |0          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+
```

```
SQL_lite > SELECT SNAME,SAGE FROM STUDENT WHERE SAGE=21;
[#] In Table [STUDENT] Selected:
+---------------------------------------------+
|WANGWU     |21         |
+---------------------------------------------+

SQL_lite > SELECT SNAME,SAGE FROM STUDENT WHERE (((SAGE=21)));
[#] In Table [STUDENT] Selected:
+---------------------------------------------+
|WANGWU     |21         |
+---------------------------------------------+

SQL_lite > SELECT SNAME,SAGE FROM STUDENT WHERE (SAGE>21) AND (SSEX=0);
[#] In Table [STUDENT] Selected:
+---------------------------------------------+
|LISI       |23         |
+---------------------------------------------+
|XIAOBAI    |23         |
+---------------------------------------------+
```

```
SQL_lite > SELECT SNAME,SAGE FROM STUDENT WHERE (SAGE>21) OR (SSEX=0);
[#] In Table [STUDENT] Selected:
+---------------------------------------------+
|ZHANGSAN   |22         |
+---------------------------------------------+
|LISI       |23         |
+---------------------------------------------+
|ZHAOLIU    |22         |
+---------------------------------------------+
|XIAOBAI    |23         |
+---------------------------------------------+
|XIAOHEI    |19         |
+---------------------------------------------+

SQL_lite > SELECT * FROM STUDENT WHERE SSEX!=1;
[#] In Table [STUDENT] Selected:
+-----------------------------------------------------+
|LISI       |23         |0          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|XIAOBAI    |23         |0          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+
```

测试 5：

输入：

```
1. SELECT * FROM STUDENT;SELECT * FROM COURSE;
2. select * from student,course;
3. SELECT * FROM STUDENT,COURSE WHERE (SSEX=0) AND (CID=1);
```

输出：多表查询符合设计

```
SQL_lite > select * from student,course;
[#] Selected From Table [student] :
+-------------------------------------------------+
|ZHANGSAN   |22         |1          |
+-------------------------------------------------+
|LISI       |23         |0          |
+-------------------------------------------------+
|WANGWU     |21         |NULL       |
+-------------------------------------------------+
|ZHAOLIU    |22         |1          |
+-------------------------------------------------+
|XIAOBAI    |23         |0          |
+-------------------------------------------------+
|XIAOHEI    |19         |0          |
+-------------------------------------------------+
[#] Selected From Table [course] :
+-------------------------------------------------+
|DB         |1          |
+-------------------------------------------------+
|COMPILER   |2          |
+-------------------------------------------------+
```

```
SQL_lite > SELECT * FROM STUDENT,COURSE WHERE (SSEX=0) AND (CID=1);
[#] In Table [STUDENT] Selected:
+-------------------------------------------------+
|LISI       |23         |0          |
+-------------------------------------------------+
|XIAOBAI    |23         |0          |
+-------------------------------------------------+
|XIAOHEI    |19         |0          |
+-------------------------------------------------+
[#] In Table [COURSE] Selected:
+-------------------------------------------------+
|DB         |1          |
+-------------------------------------------------+
```

测试 6：

输入：

```
1.  SELECT * FROM STUDENT;
2.  DELETE FROM STUDENT WHERE (SAGE>21) AND (SSEX=0);
3.  SELECT * FROM STUDENT;
```

输出：成功删除记录

```
SQL_lite > SELECT * FROM STUDENT;
[#] Selected From Table [STUDENT] :
+-----------------------------------------------------+
|ZHANGSAN   |22         |1          |
+-----------------------------------------------------+
|LISI       |23         |0          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|ZHAOLIU    |22         |1          |
+-----------------------------------------------------+
|XIAOBAI    |23         |0          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+

SQL_lite > DELETE FROM STUDENT WHERE (SAGE>21) AND (SSEX=0);
[#] Data Deleted, 2 Rows Affected

SQL_lite > SELECT * FROM STUDENT;
[#] Selected From Table [STUDENT] :
+-----------------------------------------------------+
|ZHANGSAN   |22         |1          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|ZHAOLIU    |22         |1          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+
```

测试 7：

输入：

```
1. SELECT * FROM STUDENT;
2. UPDATE STUDENT SET SAGE=21 WHERE SSEX=1;
3. SELECT * FROM STUDENT;
4. UPDATE STUDENT SET SAGE=27,SSEX=1 WHERE SNAME='ZHANGSAN';
5. SELECT * FROM STUDENT;
```

输出：成功更新记录

```
SQL_lite > SELECT * FROM STUDENT;
[#] Selected From Table [STUDENT] :
+-----------------------------------------------------+
|ZHANGSAN   |22         |1          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|ZHAOLIU    |22         |1          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+

SQL_lite > UPDATE STUDENT SET SAGE=21 WHERE SSEX=1;
[#] Table Updated, 2 Rows Affected

SQL_lite > SELECT * FROM STUDENT;
[#] Selected From Table [STUDENT] :
+-----------------------------------------------------+
|ZHANGSAN   |21         |1          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|ZHAOLIU    |21         |1          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+
```

```
SQL_lite > UPDATE STUDENT SET SAGE=27,SSEX=1 WHERE SNAME='ZHANGSAN';
[#] Table Updated, 1 Rows Affected

SQL_lite > SELECT * FROM STUDENT;
[#] Selected From Table [STUDENT] :
+-----------------------------------------------------+
|ZHANGSAN   |27         |1          |
+-----------------------------------------------------+
|WANGWU     |21         |NULL       |
+-----------------------------------------------------+
|ZHAOLIU    |21         |1          |
+-----------------------------------------------------+
|XIAOHEI    |19         |0          |
+-----------------------------------------------------+
```

3. 总结与未来工作

3.1 未完成功能

多表连接的合理体现。

3.2 未来实现方案

建立主码与外码和约束的引索体系，在执行 CREATE 语句时支持对字段设置主码和外码引用功能，在执行 SELECT 多表查询语句时，如果几张表中有码的引用关系，则根据相应关系对字段进行结合与查重，再对检查后的表做笛卡尔积产生一张新的表输出。