编译原理教学实习报告

张俊华 16030199025

题目

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。

实习目标

1. 加深编译原理基础知识的理解：词法分析、语法分析、语法制导翻译等；
2. 加深相关基础知识的理解：数据库系统、数据结构、操作系统等。

实验环境

- 操作系统：windows10
- 集成开发环境：Jetbrains Clion
- 词法分析工具：flex
- 语法分析工具：bison

项目概况

1.2 完成情况

本项目基于 C++ 语言，采用面向对象的编程思路，设计实现了

`Core` ：数据库核心类，负责 SQL 语句的执行，磁盘中数据库文件的更新

`dbShell`: 数据库界面类， 是用户输入、语法分析与数据库核心的桥梁，用户的输入经过语法分析后到达 dbShell 类，通过 dbShell 访问数据库内核的服务，并将执行结果格式化输出。

语法分析模块：基于 Yacc 在 Linux 上的开源实现：bison 生成，编写了 `sql.y` 文件，调用 `yyparse()` 函数，在语法分析构建语法树的过程中，对语义进行处理，调用 dbShell 提供的数据库访问接口。

词法分析模块：基于 lex 在 Linux 上的开源实现：flex 生成，编写 `sql.l` 文件，为语法分析过程提供输入记号流。

- **实现的功能**
    1. 实现了基本 SQL 语句的词法分析和语法分析。
    2. 实现了 SQL 语法制导翻译
    3. 实现了数据库在本地磁盘上的分页式持久化存储
    4. 实现了数据库前后端分离，查询结果表格化输出
- **完成适配的 SQL 语句：**
    - 数据库的创建、显示和删除

> CREATE DATABASE、 SHOW DATABASES、 DROP DATABASE、 USE DATABASE

- 数据表的创建、显示和删除

  > REATE TABLE SHOW TABLES DROP TABLE

- SQL 插入指令（指定列或按顺序）

  > INSERT INTO VALUES

- 数据库单表查询 （WHERE 复合条件判断、选定指定列）

  > SELECT SNAME,SAGE FROM STUDENT WHERE SAGE=21;

- 数据库多表查询

  > SELECT fields_star FROM tables

- 记录的删除

  > DELETE FROM table WHERE conditions

- 记录指定字段更新

  > UPDATE table SET updates WHERE conditions

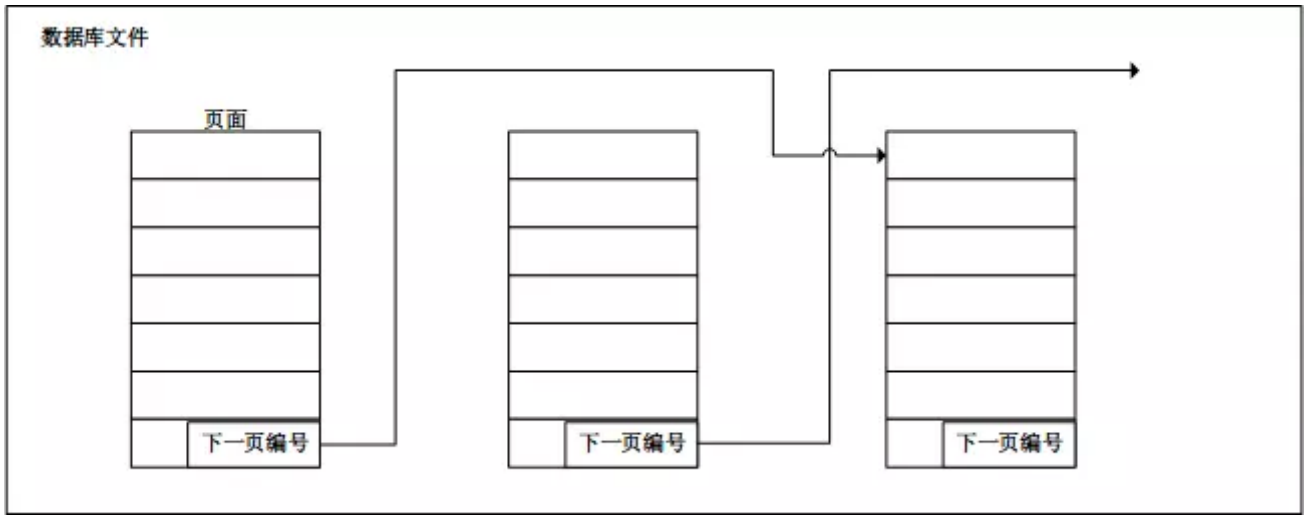
实验方案——需求分析与测试

2.1 逻辑结构与物理结构

数据库物理结构设计

数据的基本存储单元为 页，每一页存放格式（每一行的长度，每个字段的偏移量）完全相同的数据项，

每一页的大小为 2K，每一页的最后四个字节，用于标识下一页编号，通过编号*页大小，获取下一页的起始地址，相同类型的页面之间，可以不连续。



每个数据库采用两个文件进行持久化记录：

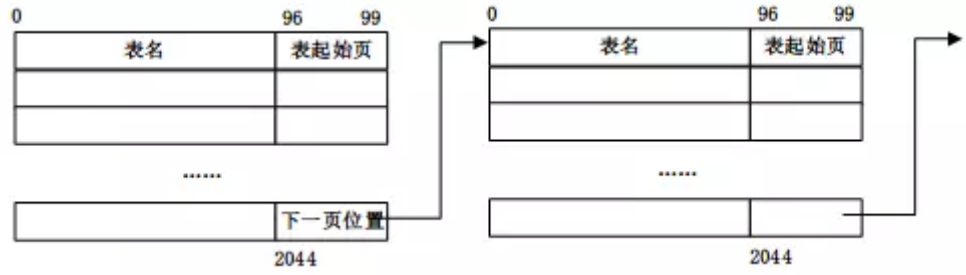
1. 元数据文件 `*.db`

元数据文件中有两种页面类型：表字典页、列字典列；

- 表字典页：初始编号为 0

  字典页中记录每个表在基本数据文件 `*.dat` 中的起始页

  每一条记录长度为 100 字节，其中，表名占 96 字节，表的起始页编号使用 int 型表示，占 4 字节。



- 列字典页：初始编号为 1

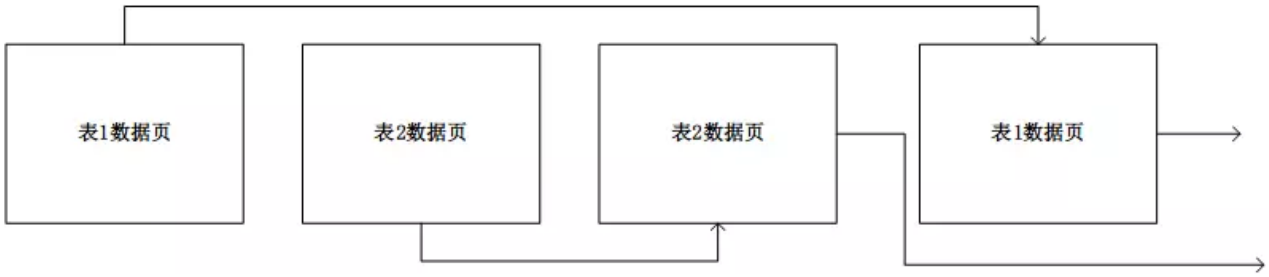
  列字典页记录了每个表的列属性，其中，每一条记录长度为 200 字节，具体结构如下图：

元数据文件在初始情况下只有两页，后期根据需求，自动向文件后部添加表字典页或列字典页，两个不同类型的页面在元数据文件中交替排布：

2. 基本数据文件 `*.dat`

   记录了每个表的具体内容，依据列字典，得到偏移量，进行文件读写。

   数据库中每一个表对应一种类型的页面，页面之间通过每一页的最后四个字节的下一页编号进行连接。



2.2 语法结构和数据结构

- CREATE 语句的产生式语法结构：

  `createsql: CREATE TABLE table '(' fieldsdefinition ')' ';'`

  - 非终结符 `fieldsdefinition` 的数据结构

```
struct table_field_node {
    char * field_name;
    enum type {
        INT, STRING
    }type;
    int len;
    int offset;
    table_field_node * next = nullptr;
};
```

- INSERT 语句的产生式语法结构

```
insertsql: INSERT INTO table VALUES '(' values ')' ';'
         | INSERT INTO table '(' fields ')' VALUES '(' values ')' ';'
```

  - 非终结符 `insertsql` 的数据结构

```
struct insert_node {
    table_node * table = nullptr;
    table_field_node * field = nullptr;
    values_node * values = nullptr;
};
```

  - 非终结符 `fields` 的数据结构

```
struct table_field_node {
    char * field_name;
    enum type {
        INT, STRING
    }type;
    int len;
    int offset;
    table_field_node * next = nullptr;
};
```

  - 非终结符 `values` 的数据结构

```
struct values_node {
    enum type {
        INT, STRING
    }type;
    int intval;
    char * chval;
    values_node * next = nullptr;
};
```

- SELECT 语句的产生式语法结构

```
selectsql:  SELECT fields_star FROM tables ';'
         |  SELECT fields_star FROM tables WHERE conditions ';'
         ;
```

- 非终结符 `fields_star` 的产生式语法结构

```
fields_star: table_fields { $$ = $1;}
           | '*'
           ;
```

- 非终结符 fields_star 的数据结构

```
struct table_field_node {
    char * field_name;
    enum type {
        INT, STRING
    }type;
    int len;
    int offset;
    table_field_node * next = nullptr;
};
```

- 非终结符 `conditions` 的数据结构

```
struct condexp_node {
    condexp_node * left = nullptr;
    enum op {
        AND, OR, EQ, G, B, NOT
    }op;
    condexp_node * right = nullptr;
    enum type {
        INT, STRING, COLUM, LOGIC
    }type;
    int intval;
    char * chval;
};
```

- DELETE 语句产生式语法结构

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

- DELETE 语句数据结构同SELECT 语句

- UPDATE 语句产生式语法结构

```
updatesql: UPDATE table SET updates WHERE conditions  ';'
```

- UPDATE 语句数据结构同 SELECT 语句

2.3 词法分析

编写 `sql.l` 作为词法分析文件

在词法分析文件中，需要用到在语法分析中定义的终结符，因此，引入相关的头文件 `sql.tab.h`：

```
%{
    #include <stdio.h>
    #include <vector>
    #include "dbCore.h"
    #include "sql.tab.h"
    #include <cstring>
    int cur_line = 1;
%}
```

词法分析过程中，需要识别出相应的终结符， 例如

```
CREATE CREATE|create
SHOW SHOW|show
DROP DROP|drop
USE USE|use
DATABASE DATABASE|database
...
{CREATE} { return CREATE;}
{SHOW} {return SHOW; }
{DROP} {return DROP; }
{USE} {return USE;}
```

这样，用户输入的文本会被拆分成终结符流，并返回每一个终结符对应的类型。

对于特殊的终结符，例如 ID (标识符)、 NUMBER(数字)、STRING(字符串)，在进行词法分析的时候，还需要将其字面值存放到 `yylval` 变量中，以供语法分析文件调用。

```
{CHAR} {yylval.typeval = CHAR; return CHAR; }
{INT} {yylval.typeval = INT; return INT;}
```

yylval 为结构体，其在语法分析文件 `sql.y` 中定义。

2.4 语法分析和语义分析

语法分析文件为：`sql.y`，在语法分析文件中，进行了 yylval 结构体的定义，终结符以及终结符和非终结符的类型定义，以及构成每个非终结符的语法规则。

- **yylval 结构体定义：**

  由于该项目采用语法制导翻译的方式执行 sql 语句，因此，在语法分析的过程中，就需要对 sql 语句的关键信息进行提取，构建出树状结构。树状结构所需要的节点数据类型，在 %union 中定义：

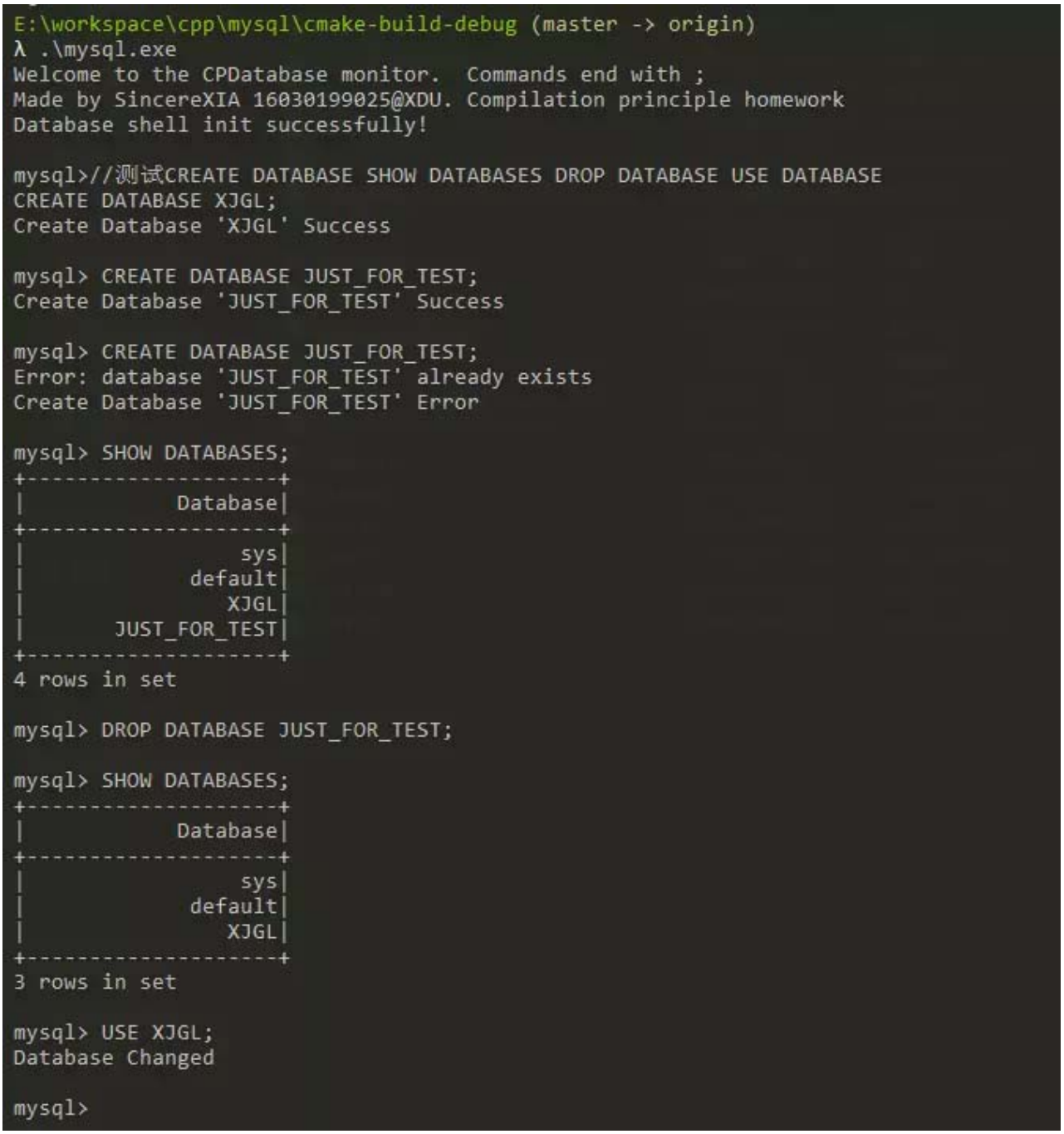
```
%union {
  int intval;
  char * chval;
  int typeval;
  table_field_node * tableField;
  table_node * tableNode;
  insert_node * insertNode;
  values_node * valuesNode;
  select_node * selectNode;
  condexp_node * condNode;
  update_node * updateNode;
  set_value * setNode;
}
```

- **终结符和非终结符定义：**

  非终结符使用 %type 描述其类型，类型确定后便可以用 $$ 符号对其值进行引用

实现与测试

- CREATE 测试



- 测试CREATE TABLE SHOW TABLES DROP TABLE

```
mysql> CREATE TABLE STUDENT(SNAME CHAR(20),SAGE INT,SSEX INT);CREATE TABLE COURSE(CNAME CHAR(20),CID INT);
Error: table 'STUDENT' already exists
Create Table 'STUDENT' Failed

mysql> Error: table 'COURSE' already exists
Create Table 'COURSE' Failed

mysql> CREATE TABLE CS(SNAME CHAR(20),CID INT);
Error: table 'CS' already exists
Create Table 'CS' Failed

mysql> CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22));
Create Table 'TEST_TABLE' Success

mysql> CREATE TABLE TEST_TABLE(COL1 CHAR(22),COL2 INT,COL3 CHAR(22));
Error: table 'TEST_TABLE' already exists
Create Table 'TEST_TABLE' Failed

mysql> SHOW TABLES;
+--------------------+
|      Tables_in_XJGL|
+--------------------+
|             STUDENT|
|              COURSE|
|                  CS|
|          TEST_TABLE|
+--------------------+
4 rows in set

mysql> DROP TABLE TEST_TABLE;
Drop Table Success

mysql> SHOW TABLES;
+--------------------+
|      Tables_in_XJGL|
+--------------------+
|             STUDENT|
|              COURSE|
|                  CS|
+--------------------+
3 rows in set

mysql>
```

- 测试INSERT INTO VALUES

```
mysql> INSERT INTO STUDENT(SNAME,SAGE,SSEX) VALUES ('ZHANGSAN',22,1);
STUDENT
Query OK, 1row affected

mysql> INSERT INTO STUDENT VALUES ('LISI',23,0);
STUDENT
Query OK, 1row affected

mysql> INSERT INTO STUDENT(SNAME,SAGE) VALUES ('WANGWU',21);
STUDENT
Query OK, 1row affected

mysql> INSERT INTO STUDENT VALUES ('ZHAOLIU',22,1);
STUDENT
Query OK, 1row affected

mysql> INSERT INTO STUDENT VALUES ('XIAOBAI',23,0);
STUDENT
Query OK, 1row affected

mysql> INSERT INTO STUDENT VALUES ('XIAOHEI',19,0);
STUDENT
Query OK, 1row affected

mysql> INSERT INTO COURSE(CNAME,CID) VALUES ('DB',1);
COURSE
Query OK, 1row affected

mysql> INSERT INTO COURSE (CNAME,CID) VALUES('COMPILER',2);
COURSE
Query OK, 1row affected

mysql> insert into course (CNAME,CID) VALUES('C',3);
COURSE
Query OK, 1row affected
```

- 测试单表查询

```
mysql> SELECT SNAME,SAGE,SSEX FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   22|    1|
|    LISI|   23|    0|
|  WANGWU|   21|    0|
| ZHAOLIU|   22|    1|
| XIAOBAI|   23|    0|
| XIAOHEI|   19|    0|
+--------------------+
6 rows in set

mysql> SELECT SNAME,SAGE FROM STUDENT;
+--------------+
|   SNAME| SAGE|
+--------------+
|ZHANGSAN|   22|
|    LISI|   23|
|  WANGWU|   21|
| ZHAOLIU|   22|
| XIAOBAI|   23|
| XIAOHEI|   19|
+--------------+
6 rows in set

mysql> SELECT * FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   22|    1|
|    LISI|   23|    0|
|  WANGWU|   21|    0|
| ZHAOLIU|   22|    1|
| XIAOBAI|   23|    0|
| XIAOHEI|   19|    0|
+--------------------+
6 rows in set

mysql> SELECT SNAME,SAGE FROM STUDENT WHERE SAGE=21;
+------------+
| SNAME| SAGE|
+------------+
|WANGWU|   21|
+------------+
1 rows in set
```

、

- 测试多表查询

```
mysql> select * from student,course;
+-----------------------------------+
|   SNAME| SAGE| SSEX|   CNAME|  CID|
+-----------------------------------+
|ZHANGSAN|   22|    1|      DB|    1|
|    LISI|   23|    0|      DB|    1|
|  WANGWU|   21|    0|      DB|    1|
| ZHAOLIU|   22|    1|      DB|    1|
| XIAOBAI|   23|    0|      DB|    1|
| XIAOHEI|   19|    0|      DB|    1|
|ZHANGSAN|   22|    1|COMPILER|    2|
|    LISI|   23|    0|COMPILER|    2|
|  WANGWU|   21|    0|COMPILER|    2|
| ZHAOLIU|   22|    1|COMPILER|    2|
| XIAOBAI|   23|    0|COMPILER|    2|
| XIAOHEI|   19|    0|COMPILER|    2|
|ZHANGSAN|   22|    1|       C|    3|
|    LISI|   23|    0|       C|    3|
|  WANGWU|   21|    0|       C|    3|
| ZHAOLIU|   22|    1|       C|    3|
| XIAOBAI|   23|    0|       C|    3|
| XIAOHEI|   19|    0|       C|    3|
+-----------------------------------+
18 rows in set

mysql> SELECT * FROM STUDENT,COURSE WHERE (SSEX=0) AND (CID=1);
+--------------------------------+
|  SNAME| SAGE| SSEX|CNAME|  CID|
+--------------------------------+
|   LISI|   23|    0|   DB|    1|
| WANGWU|   21|    0|   DB|    1|
|XIAOBAI|   23|    0|   DB|    1|
|XIAOHEI|   19|    0|   DB|    1|
+--------------------------------+
4 rows in set
```

- 测试DELETE语句

```
mysql> SELECT * FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   22|    1|
|    LISI|   23|    0|
|  WANGWU|   21|    0|
| ZHAOLIU|   22|    1|
| XIAOBAI|   23|    0|
| XIAOHEI|   19|    0|
+--------------------+
6 rows in set

mysql> DELETE FROM STUDENT WHERE (SAGE>21) AND (SSEX=0);
Query OK, 2rows affected

mysql> SELECT * FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   22|    1|
|  WANGWU|   21|    0|
| ZHAOLIU|   22|    1|
| XIAOHEI|   19|    0|
+--------------------+
4 rows in set
```

- 测试 UPDATE

```
mysql> SELECT * FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   21|    1|
|  WANGWU|   21|    0|
| ZHAOLIU|   21|    1|
| XIAOHEI|   19|    0|
+--------------------+
4 rows in set

mysql> UPDATE STUDENT SET SAGE=21 WHERE SSEX=1;
Query OK, 2rows affected

mysql> SELECT * FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   21|    1|
|  WANGWU|   21|    0|
| ZHAOLIU|   21|    1|
| XIAOHEI|   19|    0|
+--------------------+
4 rows in set

mysql> UPDATE STUDENT SET SAGE=27,SSEX=1 WHERE SNAME='ZHANGSAN';
Query OK, 1rows affected

mysql> SELECT * FROM STUDENT;
+--------------------+
|   SNAME| SAGE| SSEX|
+--------------------+
|ZHANGSAN|   27|    1|
|  WANGWU|   21|    0|
| ZHAOLIU|   21|    1|
| XIAOHEI|   19|    0|
+--------------------+
4 rows in set
```

总结

本次教学实习，我独立完成了DBMS原型系统，使我对编译原理的相关知识有了更深的了解和实际的体验，没有接触过计算机或者对计算机不是特别了解的人可能觉得计算机特别神秘而且不知道为什么它可以实现那么复杂的功能，而就我们而言越是深入学习越是渴望了解其工作原理。很幸运这学期我们开设了编译原理这一课程，也在编译原理课程中完成了教学实习的环节，体现了现在大学教学中对大学生动手能力的重视。本次教学实习，从词法分析，到语法分析，语法制导翻译，我对编译的具体过程有了更深刻的理解。但是由于课程及实验时间的限制，我想我们学到的东西还是太少了，不过没关系，这毕竟为我们以后的学习打下了基础。 总之，这次课程设计给我提供了动手实验的机会，使我对编译原理有了更深的印象和认识。编译原理是计算机专业的基础课。这门课使我们对高级语言有了更深的理解，对于我们后续课程的学习无疑也具有积极的意义。计算机专业是一个很渊博的专业，我们现在有很好的机会站在巨人的肩膀上学习，虽然通过这学期的课程设计学到了很多知识，但那只是计算机知识海洋中的一滴，我将继续努力进行深入的学习，为以后打下坚实的基础。