实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

CS33503 数据库系统实验

实验检查记录

实验结果的正确性(60%)		表达能力(10%)	
实验过程的规范性(10%)		实验报告(20%)	
加分(5%)		总成绩(100%)	

实验报告

一、实验目的（介绍实验目的）

1、掌握一种关系数据库管理系统（RDBMS）的使用方法。
2、学会使用 SQL 创建、修改、查询和控制关系数据库。

二、实验环境（介绍实验使用的硬件设备、软件系统、开发工具等）

1、硬件设备：
i7-9750H CPU@2.60GHz 2.59GHz; 1.8GHz; 16G RAM;
2、软件系统
Ubuntu 20.04.4 LTS (GNU/Linux 5.10.102.1-microsoft-standard-WSL2 x86_64)
3、开发工具
MySQL

三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

1、连接数据库

输入命令“mysql -u root -p”并按照提示输入密码

```
root@Youngsc-Desktop:/mnt/c/Users/Youngsc# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.28-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

2、MySql 常用命令

2.1 显示数据库列表
输入命令“show databases;”查看当前服务器上的数据库。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

2.2 切换数据库

输入命令“use 数据库名;”切换当前使用的数据库。

```
mysql> use college;
Database changed
```

2.3 查看表

输入命令“show tables;”可以显示当前数据库的所有表名。

```
mysql> show tables;
+------------------+
| Tables_in_college |
+------------------+
| Course           |
| SC               |
| Student          |
+------------------+
3 rows in set (0.01 sec)
```

2.4 查看关系

输入命令“desc 表名;”可以查看对应关系

```
mysql> desc SC;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| Sno   | char(6) | NO   | PRI | NULL    |       |
| Cno   | char(4) | NO   | PRI | NULL    |       |
| Grade | int     | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

2.5 导入数据库

命令: source sql 文件路径

```
mysql> source /mnt/d/github/School_Pro/数据库系统/Lab1/college.sql;
Query OK, 0 rows affected (0.07 sec)

Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.03 sec)

Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.05 sec)

Query OK, 7 rows affected (0.00 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

2.6 查看参数值

命令: show variables like "参数名";

```
mysql> show variables like "%char%";
+-------------------------------------+----------------------------+
| Variable_name                       | Value                      |
+-------------------------------------+----------------------------+
| character_set_client                | utf8mb4                    |
| character_set_connection            | utf8mb4                    |
| character_set_database              | utf8mb4                    |
| character_set_filesystem            | binary                     |
| character_set_results               | utf8mb4                    |
| character_set_server                | utf8mb4                    |
| character_set_system                | utf8mb3                    |
| character_sets_dir                  | /usr/share/mysql/charsets/ |
| validate_password.special_char_count | 1                          |
+-------------------------------------+----------------------------+
9 rows in set (0.00 sec)
```

2.7 退出 MySql

命令: exit;

```
mysql> exit;
Bye
```

3、实用 SQL 命令

3.1 创建数据库

输入命令“create database 数据库名;”来创建数据库。

```
mysql> create database college;
Query OK, 1 row affected (0.03 sec)
```

3.2 删除数据库

输入命令“drop database 数据库名;”删除对应数据库。

```
mysql> drop database college;
Query OK, 0 rows affected (0.03 sec)
```

3.3 修改表名称

```
mysql> rename table SC to SCC;
Query OK, 0 rows affected (0.03 sec)
```

4、创建数据库

4.1 新建数据库并使用该数据库

```
mysql> create database college;
Query OK, 1 row affected (0.02 sec)

mysql> use college;
Database changed
```

4.2 创建关系

```
mysql> CREATE TABLE Student (
    -> Sno CHAR(6) PRIMARY KEY,
    -> Sname VARCHAR(10) NOT NULL,
    -> Ssex CHAR CHECK (Ssex IN ('M', 'F')),
    -> e INTSage INT CHECK (Sage > 0),
    -> Sdept VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> CREATE TABLE Course (Cno CHAR(4) PRIMARY KEY);
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE SC (
    -> Sno CHAR(6),
    -> Cno CHAR(4),
    -> Grade INT,
    -> PRIMARY KEY (Sno, Cno),
    -> FOREIGN KEY (Sno) REFERENCES Student(Sno),
    -> FOREIGN KEY (Cno) REFERENCES Course(Cno)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> desc Student;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| Sno   | char(6)     | NO   | PRI | NULL    |       |
| Sname | varchar(10) | NO   |     | NULL    |       |
| Ssex  | char(1)     | YES  |     | NULL    |       |
| Sage  | int         | YES  |     | NULL    |       |
| Sdept | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> desc SC;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| Sno   | char(6) | NO   | PRI | NULL    |       |
| Cno   | char(4) | NO   | PRI | NULL    |       |
| Grade | int     | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> desc Course;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| Cno   | char(4) | NO   | PRI | NULL    |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

4.3 修改关系

```
mysql> ALTER TABLE Student ADD attach Char(5);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc Student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Sno    | char(6)     | NO   | PRI | NULL    |       |
| Sname  | varchar(10) | NO   |     | NULL    |       |
| Ssex   | char(1)     | YES  |     | NULL    |       |
| Sage   | int         | YES  |     | NULL    |       |
| Sdept  | varchar(20) | YES  |     | NULL    |       |
| attach | char(5)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE Student DROP attach;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4.4 添加元组

```
mysql> INSERT INTO Student VALUES
    -> ('PH-001', 'Nick', 'M', 20, 'Physics'),
    -> ('CS-001', 'Elsa', 'F', 19, 'CS'),
    -> ('CS-002', 'Ed', 'M', 19, 'CS'),
    -> ('MA-001', 'Abby', 'F', 18, 'Math'),
    -> ('MA-002', 'Cindy', 'F', 19, 'Math')
    -> ;
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Course VALUES
    -> ('1002'),
    -> ('1001'),
    -> ('2003'),
    -> ('3006')
    -> ;
Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> INSERT INTO SC VALUES
    -> ('PH-001', '1002', 92),
    -> ('PH-001', '2003', 85),
    -> ('PH-001', '3006', 88),
    -> ('CS-001', '1002', 95),
    -> ('CS-001', '3006', 90),
    -> ('CS-002', '3006', 80),
    -> ('MA-001', '1002', NULL)
    -> ;
Query OK, 7 rows affected (0.02 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

4.5 删除元组

```
mysql> DELETE FROM Course WHERE Cno='1001';
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM Course;
+------+
| Cno  |
+------+
| 1002 |
| 2003 |
| 3006 |
+------+
3 rows in set (0.00 sec)
```

4.6 修改元组

```
mysql> UPDATE Student SET Sage=21 WHERE Sname='Cindy';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM Student;
+--------+-------+------+------+---------+
| Sno    | Sname | Ssex | Sage | Sdept   |
+--------+-------+------+------+---------+
| CS-001 | Elsa  | F    |   19 | CS      |
| CS-002 | Ed    | M    |   19 | CS      |
| MA-001 | Abby  | F    |   18 | Math    |
| MA-002 | Cindy | F    |   21 | Math    |
| PH-001 | Nick  | M    |   20 | Physics |
+--------+-------+------+------+---------+
5 rows in set (0.00 sec)
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

五、在 mysql 中验证本课程第 3 章例子中给出的 College 数据库上的 SQL 语句的正确性。

（SQL 数据定义和 SQL 数据更新的例子在上述实验过程中均有验证。）

5.1 单关系查询

投影查询：

查询学生的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student;
+---------+-------+
| Sno     | Sname |
+---------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
| MA-001 | Abby  |
| MA-002 | Cindy |
| PH-001 | Nick  |
+---------+-------+
5 rows in set (0.00 sec)
```

查询所有的系名

```
mysql> SELECT DISTINCT Sdept FROM Student;
+---------+
| Sdept   |
+---------+
| CS      |
| Math    |
| Physics |
+---------+
3 rows in set (0.00 sec)
```

查询全部学生信息

```
mysql> SELECT * FROM Student;
+---------+-------+------+------+---------+
| Sno     | Sname | Ssex | Sage | Sdept   |
+---------+-------+------+------+---------+
| CS-001 | Elsa  | F    |   19 | CS      |
| CS-002 | Ed    | M    |   19 | CS      |
| MA-001 | Abby  | F    |   18 | Math    |
| MA-002 | Cindy | F    |   19 | Math    |
| PH-001 | Nick  | M    |   20 | Physics |
+---------+-------+------+------+---------+
5 rows in set (0.00 sec)
```

扩展的投影查询：

查询学生的学号和姓名（姓名全大写）

```
mysql> SELECT Sno, UPPER(Sname) FROM Student;
+---------+--------------+
| Sno     | UPPER(Sname) |
+---------+--------------+
| CS-001 | ELSA         |
| CS-002 | ED           |
| MA-001 | ABBY         |
| MA-002 | CINDY        |
| PH-001 | NICK         |
+---------+--------------+
5 rows in set (0.01 sec)
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

查询学生的姓名和出生年份

```
mysql> SELECT Sname, (2022-Sage) AS bd FROM Student;
+-------+------+
| Sname | bd   |
+-------+------+
| Elsa  | 2003 |
| Ed    | 2003 |
| Abby  | 2004 |
| Cindy | 2003 |
| Nick  | 2002 |
+-------+------+
5 rows in set (0.00 sec)
```

选择查询:

查询计算机系（CS）全体学生的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student WHERE Sdept = 'CS';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
+--------+-------+
2 rows in set (0.00 sec)
```

查询计算机系（CS）全体男同学的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student WHERE Sdept = 'CS' AND Ssex = 'M';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-002 | Ed    |
+--------+-------+
1 row in set (0.00 sec)
```

查询计算机系（CS）和数学系（Math）全体学生的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student WHERE Sdept = 'CS' OR Sdept = 'Math';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
| MA-001 | Abby  |
| MA-002 | Cindy |
+--------+-------+
4 rows in set (0.00 sec)
```

字符串匹配:

查询首字母为 E 的学生的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student WHERE Sname LIKE 'E%';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
+--------+-------+
2 rows in set (0.00 sec)
```

查询姓名为四个字母且首字母为 E 的学生的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student WHERE Sname LIKE 'E___';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
+--------+-------+
1 row in set (0.00 sec)
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

正则表达式：

查询姓名首字母为 E 或 F 的学生的学号和姓名

```
mysql> SELECT Sno, Sname FROM Student WHERE Sname REGEXP '^[EF].*';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
+--------+-------+
2 rows in set (0.03 sec)
```

空值判断：

查询选了课但还未取得成绩的同学

```
mysql> SELECT Sno FROM SC WHERE GRADE IS NULL;
+--------+
| Sno    |
+--------+
| MA-001 |
+--------+
1 row in set (0.01 sec)
```

集合操作：

查询选修了 1002 号或 3006 号课的学生的选课信息

```
mysql> SELECT * FROM SC WHERE Cno = '1002' UNION ALL
    -> SELECT * FROM SC WHERE Cno = '3006';
+--------+------+-------+
| Sno    | Cno  | Grade |
+--------+------+-------+
| CS-001 | 1002 |    95 |
| MA-001 | 1002 |  NULL |
| PH-001 | 1002 |    92 |
| CS-001 | 3006 |    90 |
| CS-002 | 3006 |    80 |
| PH-001 | 3006 |    88 |
+--------+------+-------+
6 rows in set (0.00 sec)
```

查询选修了 1002 号或 3006 号课的学生的学号

```
mysql> SELECT Sno FROM SC WHERE Cno = '1002' UNION
    -> SELECT Sno FROM SC WHERE Cno = '3006';
+--------+
| Sno    |
+--------+
| CS-001 |
| MA-001 |
| PH-001 |
| CS-002 |
+--------+
4 rows in set (0.00 sec)
```

结果排序

查询计算机系（CS）全体学生的学号和姓名，并按照学号升序排列

```
mysql> SELECT Sno, Sname FROM Student WHERE Sdept = 'CS' ORDER BY Sno;
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
+--------+-------+
2 rows in set (0.00 sec)
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

查询全体学生的信息，结果按所在系升序排列，同一个系得学生按年龄降序排列

```
mysql> SELECT * FROM Student ORDER BY Sdept ASC, Sage DESC;
+--------+-------+------+------+---------+
| Sno    | Sname | Ssex | Sage | Sdept   |
+--------+-------+------+------+---------+
| CS-001 | Elsa  | F    |   19 | CS      |
| CS-002 | Ed    | M    |   19 | CS      |
| MA-002 | Cindy | F    |   19 | Math    |
| MA-001 | Abby  | F    |   18 | Math    |
| PH-001 | Nick  | M    |   20 | Physics |
+--------+-------+------+------+---------+
5 rows in set (0.00 sec)
```

限制查询结果数量：

查询 3006 号课得分最高的前 2 名学生的学号和成绩

```
mysql> SELECT Sno, Grade FROM SC WHERE Cno = '3006'
    -> ORDER BY Grade DESC LIMIT 2;
+--------+-------+
| Sno    | Grade |
+--------+-------+
| CS-001 |    90 |
| PH-001 |    88 |
+--------+-------+
2 rows in set (0.00 sec)
```

聚集查询：

查询计算机系全体学生的数量

```
mysql> SELECT COUNT(*) FROM Student WHERE Sdept = 'CS';
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.01 sec)
```

查询计算机系学生的最大年龄

```
mysql> SELECT MAX(Sage) FROM Student WHERE Sdept = 'CS';
+-----------+
| MAX(Sage) |
+-----------+
|        19 |
+-----------+
1 row in set (0.00 sec)
```

分组查询：

统计每门课的选课人数和平均成绩

```
mysql> SELECT Cno, COUNT(*), AVG(Grade) FROM SC GROUP BY Cno;
+------+----------+------------+
| Cno  | COUNT(*) | AVG(Grade) |
+------+----------+------------+
| 1002 |        3 |    93.5000 |
| 2003 |        1 |    85.0000 |
| 3006 |        3 |    86.0000 |
+------+----------+------------+
3 rows in set (0.00 sec)
```

统计每个系的男生人数和女生人数

```
mysql> SELECT Sdept, Ssex, COUNT(*) FROM Student
    -> GROUP BY Sdept, Ssex;
+---------+------+----------+
| Sdept   | Ssex | COUNT(*) |
+---------+------+----------+
| CS      | F    |        1 |
| CS      | M    |        1 |
| Math    | F    |        2 |
| Physics | M    |        1 |
+---------+------+----------+
4 rows in set (0.00 sec)
```

查询选修了两门以上课程的学生的学号和选课数

```
mysql> SELECT Sno, COUNT(*) FROM SC GROUP BY Sno HAVING COUNT(*) >= 2;
+--------+----------+
| Sno    | COUNT(*) |
+--------+----------+
| CS-001 |        2 |
| PH-001 |        3 |
+--------+----------+
2 rows in set (0.00 sec)
```

查询两门以上课程得分超过 80 的学生的学号以及这些课程的平均分

```
mysql> SELECT Sno, AVG(Grade) FROM SC WHERE Grade >= 80 GROUP BY Sno HAVING COUNT(*) >= 2;
+--------+------------+
| Sno    | AVG(Grade) |
+--------+------------+
| CS-001 |    92.5000 |
| PH-001 |    88.3333 |
+--------+------------+
2 rows in set (0.00 sec)
```

5.2 连接查询

笛卡尔积：

查询学生及其选课情况，列出学号、姓名、课号、得分

```
mysql> SELECT Student.Sno, Sname, Cno, Grade
    -> FROM Student, SC
    -> WHERE Student.Sno = SC.Sno;
+--------+-------+------+-------+
| Sno    | Sname | Cno  | Grade |
+--------+-------+------+-------+
| CS-001 | Elsa  | 1002 |    95 |
| CS-001 | Elsa  | 3006 |    90 |
| CS-002 | Ed    | 3006 |    80 |
| MA-001 | Abby  | 1002 |  NULL |
| PH-001 | Nick  | 1002 |    92 |
| PH-001 | Nick  | 2003 |    85 |
| PH-001 | Nick  | 3006 |    88 |
+--------+-------+------+-------+
7 rows in set (0.01 sec)
```

内连接：

查询学生及其选课情况，列出学号、姓名、课号、得分

```
mysql> SELECT Student.Sno, Sname, Cno, Grade FROM Student JOIN SC ON (Student.Sno = SC.Sno);
+--------+-------+------+-------+
| Sno    | Sname | Cno  | Grade |
+--------+-------+------+-------+
| CS-001 | Elsa  | 1002 |    95 |
| CS-001 | Elsa  | 3006 |    90 |
| CS-002 | Ed    | 3006 |    80 |
| MA-001 | Abby  | 1002 |  NULL |
| PH-001 | Nick  | 1002 |    92 |
| PH-001 | Nick  | 2003 |    85 |
| PH-001 | Nick  | 3006 |    88 |
+--------+-------+------+-------+
7 rows in set (0.00 sec)
```

```
mysql> SELECT Student.Sno, Sname, Cno, Grade FROM Student JOIN SC USING (Sno);
+--------+-------+------+-------+
| Sno    | Sname | Cno  | Grade |
+--------+-------+------+-------+
| CS-001 | Elsa  | 1002 |    95 |
| CS-001 | Elsa  | 3006 |    90 |
| CS-002 | Ed    | 3006 |    80 |
| MA-001 | Abby  | 1002 |  NULL |
| PH-001 | Nick  | 1002 |    92 |
| PH-001 | Nick  | 2003 |    85 |
| PH-001 | Nick  | 3006 |    88 |
+--------+-------+------+-------+
7 rows in set (0.00 sec)
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

自然连接：

查询学生及其选课情况，列出学号、姓名、课号、得分

```
mysql> SELECT Student.Sno, Sname, Cno, Grade FROM Student NATURAL JOIN SC;
+--------+-------+------+-------+
| Sno    | Sname | Cno  | Grade |
+--------+-------+------+-------+
| CS-001 | Elsa  | 1002 |    95 |
| CS-001 | Elsa  | 3006 |    90 |
| CS-002 | Ed    | 3006 |    80 |
| MA-001 | Abby  | 1002 |  NULL |
| PH-001 | Nick  | 1002 |    92 |
| PH-001 | Nick  | 2003 |    85 |
| PH-001 | Nick  | 3006 |    88 |
+--------+-------+------+-------+
7 rows in set (0.00 sec)
```

自连接：

查询和 Elsa 在同一个系学习的学生的学号和姓名

```
mysql> SELECT S2.Sno, S2.Sname
    -> FROM Student AS S1 JOIN Student AS S2
    -> ON S1.Sdept = S2.Sdept AND S1.Sno != S2.Sno
    -> WHERE S1.Sname = 'Elsa';
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-002 | Ed    |
+--------+-------+
1 row in set (0.01 sec)
```

外连接：

查询没有选课的学生的姓名和学号：

```
mysql> SELECT Student.Sno, Sname
    -> FROM Student LEFT JOIN SC ON (Student.Sno = SC.Sno)
    -> WHERE Cno IS NULL;
+--------+-------+
| Sno    | Sname |
+--------+-------+
| MA-002 | Cindy |
+--------+-------+
1 row in set (0.00 sec)
```

5.3 嵌套查询

子查询的类型：

查询和 Elsa 在同一个系学习的学生得学号和姓名（含 Elsa）

不相关子查询

```
mysql> SELECT Sno, Sname FROM Student WHERE Sdept =
    -> (SELECT Sdept FROM Student WHERE Sname = 'Elsa');
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
+--------+-------+
2 rows in set (0.00 sec)
```

相关子查询

```
mysql> SELECT Sno, Sname FROM Student AS S WHERE EXISTS
    -> (SELECT * FROM Student AS T
    -> WHERE T.Sname = 'Elsa' AND T.Sdept = S.Sdept);
+--------+-------+
| Sno    | Sname |
+--------+-------+
| CS-001 | Elsa  |
| CS-002 | Ed    |
+--------+-------+
2 rows in set (0.00 sec)
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

六、 将 Product.sql 文件下载到当前工作目录，创建 Product 数据库，然后用 SQL 编写本课程第 3 章习题 11 中的全部数据库查询和更新语句。

(a) Find the manufacturers that sell laptops but not PC's. (使用集合差运算)

```
SELECT DISTINCT maker FROM Product WHERE Product.type = 'laptop' AND Product.maker != ALL (SELECT maker FROM Product WHERE type = 'PC');
```

```
mysql> SELECT DISTINCT maker FROM Product WHERE Product.type = 'laptop' AND Product.maker != ALL (SELECT maker FROM Product WHERE type = 'PC');
+-------+
| maker |
+-------+
| F     |
| G     |
+-------+
2 rows in set (0.00 sec)
```

(b) Find the manufacturers that sell laptops but not PC's. (使用含有 IN 的嵌套查询)

```
SELECT DISTINCT maker FROM Product WHERE type = 'laptop' AND maker NOT IN (SELECT maker FROM Product WHERE type = 'pc');
```

```
mysql> SELECT DISTINCT maker FROM Product WHERE type = 'laptop' AND maker NOT IN (SELECT maker FROM Product WHERE type = 'pc');
+--------+
| maker |
+--------+
| F     |
| G     |
+--------+
2 rows in set (0.01 sec)
```

(c) Find the manufacturers that sell laptops but not PC's.(使用含有 EXISTS 的嵌套查询)

```
SELECT DISTINCT maker FROM Product as P1 WHERE P1.type = 'laptop' AND NOT EXISTS(SELECT * FROM Product AS P2 WHERE P1.type = 'laptop' AND P2.type = 'pc' AND P1.maker = P2.maker);
```

```
mysql> SELECT DISTINCT maker FROM Product as P1 WHERE P1.type = 'laptop' AND NOT EXISTS(SELECT * FROM Product AS P2 WHERE P1.type = 'laptop' AND P2.type = '
pc' AND P1.maker = P2.maker);
+-------+
| maker |
+-------+
| F     |
| G     |
+-------+
2 rows in set (0.00 sec)
```

(d) Find the model numbers of all printers that are cheaper than the printer model 3002. (使用内连接查询)

```
SELECT P1.model FROM Printer AS P1 JOIN Printer AS P2 ON (P2.model = 3002 AND P1.price < p2.price);
```

```
mysql> SELECT P1.model FROM Printer AS P1 JOIN Printer AS P2 ON (P2.model = 3002 AND P1.price < P2.price);
+--------+
| model |
+--------+
|  3001 |
|  3004 |
|  3005 |
|  3006 |
|  3007 |
+--------+
5 rows in set (0.00 sec)
```

(e) Find the model numbers of all printers that are cheaper than the printer model 3002. (使用含有比较运算符的嵌套查询)

```
SELECT model FROM Printer WHERE price  < (SELECT price FROM Printer WHERE model = 3002);
```

```
mysql> SELECT model FROM Printer WHERE price  < (SELECT price FROM Printer WHERE model = 3002);
+--------+
| model |
+--------+
|  3001 |
|  3004 |
|  3005 |
|  3006 |
|  3007 |
+--------+
5 rows in set (0.00 sec)
```

(f) Find the model numbers of all printers that are cheaper than the printer model 3002. (使用含有 EXISTS 的嵌套查询)

```
SELECT model FROM Printer AS P1 WHERE EXISTS (SELECT * FROM Printer AS P2 WHERE P1.price < P2.price AND P2.model = 3002);
```

```
mysql> SELECT model FROM Printer AS P1 WHERE EXISTS (SELECT * FROM Printer AS P2 WHERE P1.price < P2.price AND P2.model = 3002);
+-------+
| model |
+-------+
|  3001 |
|  3004 |
|  3005 |
|  3006 |
|  3007 |
+-------+
5 rows in set (0.00 sec)
```

(g) Find the PC model with the highest available speed. (使用外连接查询)

```
SELECT DISTINCT P1.model FROM pc AS P1 LEFT JOIN pc AS P2 ON (P1.speed < P2.speed) WHERE P2.price IS NULL;
```

```
mysql> SELECT DISTINCT P1.model FROM PC AS P1 LEFT JOIN PC AS P2 ON (P1.speed < P2.speed) WHERE P2.price IS NULL;
+--------+
| model |
+--------+
|  1005 |
|  1006 |
+--------+
2 rows in set (0.00 sec)
```

(h) Find the PC model with the highest available speed. (使用含有 IN 的嵌套查询)

```
SELECT model FROM PC WHERE speed IN (SELECT MAX(speed) FROM PC);
```

```
mysql> SELECT model FROM PC WHERE speed IN (SELECT MAX(speed) FROM PC);
+--------+
| model |
+--------+
|  1005 |
|  1006 |
+--------+
2 rows in set (0.00 sec)
```

(i) Find the PC model with the highest available speed. (使用含有=的嵌套查询)

```
SELECT model FROM PC WHERE speed = (SELECT MAX(speed) FROM PC);
```

```
mysql> SELECT model FROM PC WHERE speed = (SELECT MAX(speed) FROM PC);
+--------+
| model |
+--------+
|  1005 |
|  1006 |
+--------+
2 rows in set (0.01 sec)
```

(j) Find the PC model with the highest available speed. (使用含有>=的嵌套查询)

```
SELECT model FROM PC WHERE speed >= ALL (SELECT speed FROM PC);
```

```
mysql> SELECT model FROM PC WHERE speed >= ALL (SELECT speed FROM PC);
+-------+
| model |
+-------+
|  1005 |
|  1006 |
+-------+
2 rows in set (0.00 sec)
```

(k) Find the PC model with the highest available speed. (使用含有 EXISTS 的嵌套查询)

```
SELECT model FROM PC AS P1 WHERE NOT EXISTS (SELECT * FROM PC AS P2 WHERE P1.speed < P2.speed);
```

```
mysql> SELECT model FROM PC AS P1 WHERE NOT EXISTS (SELECT * FROM PC AS P2 WHERE P1.speed < P2.speed);
+-------+
| model |
+-------+
|  1005 |
|  1006 |
+-------+
2 rows in set (0.00 sec)
```

(l) Find the manufacturers of PC's with at least three different speeds. (使用内连接查询)

```
SELECT maker FROM (SELECT DISTINCT maker, speed FROM PC JOIN Product ON (PC.model = Product.model)) AS P GROUP BY maker HAVING COUNT(*) >= 3;
```

```
mysql> SELECT maker FROM (SELECT DISTINCT maker, speed FROM PC JOIN Product ON (PC.model = Product.model)) AS P GROUP BY maker HAVING COUNT(*) >= 3;
+-------+
| maker |
+-------+
| A     |
| D     |
| E     |
+-------+
3 rows in set (0.00 sec)
```

(m) Find the manufacturers of PC's with at least three different speeds. (使用分组查询)

```
SELECT maker FROM (SELECT DISTINCT maker, speed FROM PC JOIN Product ON (PC.model = Product.model)) AS P group BY maker HAVING COUNT(*) >= 3;
```

```
mysql> SELECT maker FROM (SELECT DISTINCT maker, speed FROM PC JOIN Product ON (PC.model = Product.model)) AS P GROUP BY maker HAVING COUNT(*) >= 3;
+-------+
| maker |
+-------+
| A     |
| D     |
| E     |
+-------+
3 rows in set (0.00 sec)
```

(n) Find the manufacturers of PC's with at least three different speeds. (使用派生关系)

```
SELECT maker FROM (SELECT maker, COUNT(*) FROM (SELECT DISTINCT speed, maker FROM PC JOIN Product ON (PC.model = Product.model)) AS TEMP GROUP BY maker) AS P(maker, num) WHERE num >= 3;
```

```
mysql> SELECT maker FROM (SELECT maker, COUNT(*) FROM (SELECT DISTINCT speed, maker FROM PC JOIN Product ON (PC.model = Product.model)) AS TEMP GROUP BY mak
er) AS P(maker, num) WHERE num >= 3;
+-------+
| maker |
+-------+
| A     |
| D     |
| E     |
+-------+
3 rows in set (0.00 sec)
```

(o) Decrease the price of all PC's made by maker A by 10%. (使用含有=的更新条

件)

```
UPDATE PC SET price = price * 0.9 WHERE model = ANY (SELECT model FROM Product WHERE maker = 'A');
```

```
mysql> UPDATE PC SET price = price * 0.9 WHERE model = ANY (SELECT model FROM Product WHERE maker = 'A');
Query OK, 3 rows affected (0.08 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

```
mysql> SELECT * FROM PC JOIN Product ON (PC.model = Product.model);
+-------+-------+------+------+-------+-------+-------+------+
| model | speed | ram  | hd   | price | maker | model | type |
+-------+-------+------+------+-------+-------+-------+------+
|  1001 |  2.66 | 1024 |  250 |  1903 | A     |  1001 | pc   |
|  1002 |   2.1 |  512 |  250 |   896 | A     |  1002 | pc   |
|  1003 |  1.42 |  512 |   80 |   430 | A     |  1003 | pc   |
|  1004 |   2.8 | 1024 |  250 |   649 | B     |  1004 | pc   |
|  1005 |   3.2 |  512 |  250 |   630 | B     |  1005 | pc   |
|  1006 |   3.2 | 1024 |  320 |  1049 | B     |  1006 | pc   |
|  1007 |   2.2 | 1024 |  200 |   510 | C     |  1007 | pc   |
|  1008 |   2.2 | 2048 |  250 |   770 | D     |  1008 | pc   |
|  1009 |     2 | 1024 |  250 |   650 | D     |  1009 | pc   |
|  1010 |   2.8 | 2048 |  300 |   770 | D     |  1010 | pc   |
|  1011 |  1.86 | 2048 |  160 |   959 | E     |  1011 | pc   |
|  1012 |   2.8 | 1024 |  160 |   649 | E     |  1012 | pc   |
|  1013 |  3.06 |  512 |   80 |   529 | E     |  1013 | pc   |
+-------+-------+------+------+-------+-------+-------+------+
```

(p) Decrease the price of all PC's made by maker A by 10%. (使用含有 IN 的更新条件)

```
UPDATE PC SET price = price*0.9 WHERE model IN (SELECT model FROM Product WHERE maker = 'A');
```

```
mysql> UPDATE PC SET price = price*0.9 WHERE model IN (SELECT model FROM Product WHERE maker = 'A');
Query OK, 3 rows affected (0.06 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT * FROM PC JOIN Product ON (PC.model = Product.model);
+-------+-------+------+------+-------+-------+-------+------+
| model | speed | ram  | hd   | price | maker | model | type |
+-------+-------+------+------+-------+-------+-------+------+
|  1001 |  2.66 | 1024 |  250 |  1903 | A     |  1001 | pc   |
|  1002 |   2.1 |  512 |  250 |   896 | A     |  1002 | pc   |
|  1003 |  1.42 |  512 |   80 |   430 | A     |  1003 | pc   |
|  1004 |   2.8 | 1024 |  250 |   649 | B     |  1004 | pc   |
|  1005 |   3.2 |  512 |  250 |   630 | B     |  1005 | pc   |
|  1006 |   3.2 | 1024 |  320 |  1049 | B     |  1006 | pc   |
|  1007 |   2.2 | 1024 |  200 |   510 | C     |  1007 | pc   |
|  1008 |   2.2 | 2048 |  250 |   770 | D     |  1008 | pc   |
|  1009 |     2 | 1024 |  250 |   650 | D     |  1009 | pc   |
|  1010 |   2.8 | 2048 |  300 |   770 | D     |  1010 | pc   |
|  1011 |  1.86 | 2048 |  160 |   959 | E     |  1011 | pc   |
|  1012 |   2.8 | 1024 |  160 |   649 | E     |  1012 | pc   |
|  1013 |  3.06 |  512 |   80 |   529 | E     |  1013 | pc   |
+-------+-------+------+------+-------+-------+-------+------+
13 rows in set (0.00 sec)
```

(q) Decrease the price of all PC's made by maker A by 10%. (使用含有 EXISTS 的更新条件)

```
UPDATE PC SET price = price * 0.9 WHERE EXISTS (SELECT model FROM Product WHERE maker = 'A' AND PC.model = Product.model);
```

```
mysql> UPDATE PC SET price = price * 0.9 WHERE EXISTS (SELECT model FROM Product WHERE maker = 'A' AND PC.model = Product.model);
Query OK, 3 rows affected (0.06 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT * FROM PC JOIN Product ON (PC.model = Product.model);
+-------+-------+------+------+-------+-------+-------+------+
| model | speed | ram  | hd   | price | maker | model | type |
+-------+-------+------+------+-------+-------+-------+------+
|  1001 |  2.66 | 1024 |  250 |  1903 | A     |  1001 | pc   |
|  1002 |   2.1 |  512 |  250 |   896 | A     |  1002 | pc   |
|  1003 |  1.42 |  512 |   80 |   430 | A     |  1003 | pc   |
|  1004 |   2.8 | 1024 |  250 |   649 | B     |  1004 | pc   |
|  1005 |   3.2 |  512 |  250 |   630 | B     |  1005 | pc   |
|  1006 |   3.2 | 1024 |  320 |  1049 | B     |  1006 | pc   |
|  1007 |   2.2 | 1024 |  200 |   510 | C     |  1007 | pc   |
|  1008 |   2.2 | 2048 |  250 |   770 | D     |  1008 | pc   |
|  1009 |     2 | 1024 |  250 |   650 | D     |  1009 | pc   |
|  1010 |   2.8 | 2048 |  300 |   770 | D     |  1010 | pc   |
|  1011 |  1.86 | 2048 |  160 |   959 | E     |  1011 | pc   |
|  1012 |   2.8 | 1024 |  160 |   649 | E     |  1012 | pc   |
|  1013 |  3.06 |  512 |   80 |   529 | E     |  1013 | pc   |
+-------+-------+------+------+-------+-------+-------+------+
```

实验题目	关系数据库管理系统与 SQL			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

(r) 题目(g) - (k)用不同方法编写相同的查询。请你从 SQL 语句的易读性和执行效率两方面对题目(g) - (k)的 SQL 语句进行分析和比较。在做效率分析时，我们假定每个关系上只有主索引，而没有其他索引(请自学第 6 章中索引的概念和功能)。

易读性：我认为（h）（i）（j）的可读性最高，因为其语句简短，并且使用了 in、比较符号这样的值观简单的数学关系符号，更加直观。而（g）语句是按照大小关系将所有的元素进行外连接，然后找到右侧为空的元素，（k）是采用了判断是否存在比当前元素更大的元素来寻找最大值，这两种方式都不是人们平时在寻找最大值时的常规思路。

执行效率：（g）仅需要执行一次外连接，效率较高，而（h）（i）（j）（k）需要执行两次连接，效率较低。

四、实验结论（总结实验发现及结论）

数据库管理系统是十分高效、便捷的数据管理方式，它可以通过连接、嵌套、单关系查询等的相互组合来实现各种各样的查询操作，而对于同一种操作的不同实现方式，其优缺点也是不尽相同，因此拥有清晰的逻辑对于使用 DBMS 是十分重要的。