

# 信息与软件工程学院上机实验报告

## (第 2 次)

### 一、实验名称

实验二：数据的 DML 及数据查询实验

### 二、实验目的及要求

#### 2.1 知识能力目标

1. 掌握简单查询、条件查询、排序、分组统计等基本 SQL 查询方法，能够对关系表中的数据进行检索与分析。
2. 掌握多表连接、子查询、聚合统计与条件表达式等关系查询方法，能够综合利用多种 SQL 语句完成结果分析。
3. 掌握更新、删除、触发器、存储过程以及宿主语言访问数据库的基本方法，理解 DML 操作与数据库程序设计的结合方式。

#### 2.2 素养目标

1. 按照工程认证“研究”要求，围绕不同查询和更新任务设计验证步骤，并根据结果对 SQL 语句执行效果进行分析。
2. 按照工程认证“使用现代工具”要求，能够综合使用 Docker、MySQL、Data-Grip 及宿主语言程序完成数据库访问、结果验证和问题定位。
3. 按照工程认证“工程与可持续发展”要求，理解对数据库进行更新、删除和程序化访问时应保持数据一致性、可恢复性与后续实验可复用性。

### 三、实验环境

1. Docker 28.5.2, build ecc6942
2. MySQL 8.0
3. DataGrip 2026.1.1

### 四、实验设计

#### 4.1 实验内容与步骤

##### 4.1.1 StudentCourse 简单查询

目的：掌握 StudentCourse 数据库中全表查询、条件查询、排序查询、限制结果数量以及分组统计等基本查询方法。

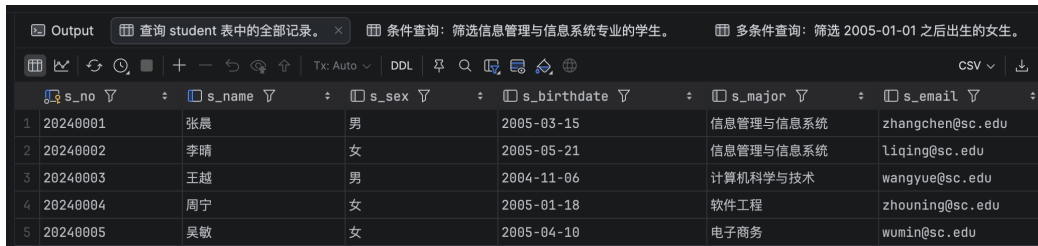
数据恢复后，依次执行以下查询并核对结果。

##### (1) 全表查询

---

```
USE StudentCourse;
```

```
SELECT *
FROM student
ORDER BY s_no;
```



Output 查询 student 表中的全部记录。 条件查询: 筛选信息管理与信息系统专业的学生。 多条件查询: 筛选 2005-01-01 之后出生的女生。

|   | s_no     | s_name | s_sex | s_birthdate | s_major   | s_email          |
|---|----------|--------|-------|-------------|-----------|------------------|
| 1 | 20240001 | 张晨     | 男     | 2005-03-15  | 信息管理与信息系统 | zhangchen@sc.edu |
| 2 | 20240002 | 李晴     | 女     | 2005-05-21  | 信息管理与信息系统 | liqing@sc.edu    |
| 3 | 20240003 | 王越     | 男     | 2004-11-06  | 计算机科学与技术  | wangyue@sc.edu   |
| 4 | 20240004 | 周宁     | 女     | 2005-01-18  | 软件工程      | zhouning@sc.edu  |
| 5 | 20240005 | 吴敏     | 女     | 2005-04-10  | 电子商务      | wumin@sc.edu     |

## (2) 按专业条件查询

```
SELECT s_no, s_name, s_major
FROM student
WHERE s_major = '信息管理与信息系统';
```

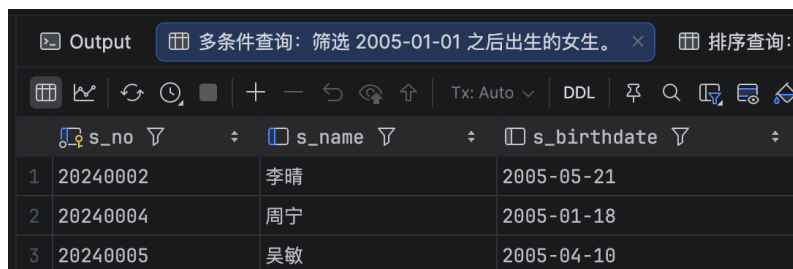


Output 查询 student 表中的全部记录。 条件查询: 筛选信息管理与信息系统专业的学生。 ×

|   | s_no     | s_name | s_major   |
|---|----------|--------|-----------|
| 1 | 20240001 | 张晨     | 信息管理与信息系统 |
| 2 | 20240002 | 李晴     | 信息管理与信息系统 |

## (3) 多条件查询

```
SELECT s_no, s_name, s_birthdate
FROM student
WHERE s_sex = '女' AND s_birthdate > '2005-01-01';
```



Output 多条件查询: 筛选 2005-01-01 之后出生的女生。 × 排序查询: ×

|   | s_no     | s_name | s_birthdate |
|---|----------|--------|-------------|
| 1 | 20240002 | 李晴     | 2005-05-21  |
| 2 | 20240004 | 周宁     | 2005-01-18  |
| 3 | 20240005 | 吴敏     | 2005-04-10  |

## (4) 排序查询

```
SELECT s_name, s_birthdate
FROM student
ORDER BY s_birthdate DESC;
```

|   | s_name | s_birthdate |
|---|--------|-------------|
| 1 | 李晴     | 2005-05-21  |
| 2 | 吴敏     | 2005-04-10  |
| 3 | 张晨     | 2005-03-15  |
| 4 | 周宁     | 2005-01-18  |
| 5 | 王越     | 2004-11-06  |

#### (5) 限制返回记录数

```
SELECT s_name, s_birthdate
FROM student
ORDER BY s_birthdate
LIMIT 2;
```

|   | s_name | s_birthdate |
|---|--------|-------------|
| 1 | 王越     | 2004-11-06  |
| 2 | 周宁     | 2005-01-18  |

#### (6) 分组统计

```
SELECT s_major, COUNT(*) AS student_count
FROM student
GROUP BY s_major;
```

|   | s_major   | student_count |
|---|-----------|---------------|
| 1 | 信息管理与信息系统 | 2             |
| 2 | 计算机科学与技术  | 1             |
| 3 | 软件工程      | 1             |
| 4 | 电子商务      | 1             |

### 4.1.2 StudentCourse 关系查询与连接

目的：掌握多表连接、子查询、聚合统计和 CASE 成绩分级等关系查询方法，理解不同关系操作的适用场景。

StudentCourse 中的关系查询结果如下。

#### (1) 三表连接查询

```
SELECT s.s_name, c.c_name, sc.grade
FROM student AS s
JOIN sc ON s.s_no = sc.s_no
JOIN course AS c ON sc.c_no = c.c_no;
```

Output 三表连接: 查看学生、课程和成绩的对应关系。 子查询:

|   | s_name | c_name  | grade |
|---|--------|---------|-------|
| 1 | 张晨     | 数据库原理   | 86    |
| 2 | 张晨     | 数据库应用实验 | 90    |
| 3 | 李晴     | 数据库原理   | 93    |
| 4 | 王越     | 程序设计基础  | 88    |
| 5 | 王越     | 数据结构    | 84    |
| 6 | 周宁     | 程序设计基础  | 91    |

## (2) 查询高于平均分的成绩

```
SELECT s.s_name, c.c_name, sc.grade
FROM student AS s
JOIN sc ON s.s_no = sc.s_no
JOIN course AS c ON sc.c_no = c.c_no
WHERE sc.grade > (
    SELECT AVG(grade)
    FROM sc
    WHERE grade IS NOT NULL
);
```

Output 子查询: 查询高于平均分的成绩。 按课程统计最高分、最低分

|   | s_name | c_name  | grade |
|---|--------|---------|-------|
| 1 | 张晨     | 数据库应用实验 | 90    |
| 2 | 李晴     | 数据库原理   | 93    |
| 3 | 周宁     | 程序设计基础  | 91    |

## (3) 按课程统计成绩情况

```
SELECT c.c_name,
       MAX(sc.grade) AS max_grade,
       MIN(sc.grade) AS min_grade,
       AVG(sc.grade) AS avg_grade,
       COUNT(sc.s_no) AS student_count
FROM course AS c
LEFT JOIN sc ON c.c_no = sc.c_no
GROUP BY c.c_no, c.c_name;
```

|   | c_name  | max_grade | min_grade | avg_grade | student_count |
|---|---------|-----------|-----------|-----------|---------------|
| 1 | 数据库原理   | 93        | 86        | 89.5000   | 2             |
| 2 | 数据库应用实验 | 90        | 90        | 90.0000   | 1             |
| 3 | 数据结构    | 84        | 84        | 84.0000   | 1             |
| 4 | 程序设计基础  | 91        | 88        | 89.5000   | 2             |

#### (4) 按学生统计平均成绩与选课门数

```
SELECT s.s_no, s.s_name, s.s_major,
       AVG(sc.grade) AS avg_grade,
       COUNT(sc.c_no) AS course_count
FROM student AS s
LEFT JOIN sc ON s.s_no = sc.s_no
GROUP BY s.s_no, s.s_name, s.s_major
ORDER BY avg_grade DESC;
```

|   | s_no     | s_name | s_major   | avg_grade | course_count |
|---|----------|--------|-----------|-----------|--------------|
| 1 | 20240002 | 李晴     | 信息管理与信息系统 | 93.0000   | 1            |
| 2 | 20240004 | 周宁     | 软件工程      | 91.0000   | 1            |
| 3 | 20240001 | 张晨     | 信息管理与信息系统 | 88.0000   | 2            |
| 4 | 20240003 | 王越     | 计算机科学与技术  | 86.0000   | 2            |
| 5 | 20240005 | 吴敏     | 电子商务      | <null>    | 0            |

#### (5) 按 CASE 语句划分成绩等级

```
SELECT s.s_name, c.c_name, sc.grade,
       CASE
         WHEN sc.grade >= 90 THEN 'A'
         WHEN sc.grade >= 80 THEN 'B'
         WHEN sc.grade >= 70 THEN 'C'
         WHEN sc.grade >= 60 THEN 'D'
         ELSE 'F'
       END AS grade_level
FROM student AS s
JOIN sc ON s.s_no = sc.s_no
JOIN course AS c ON sc.c_no = c.c_no;
```

|   | s_name | c_name  | grade | grade_level |
|---|--------|---------|-------|-------------|
| 1 | 张晨     | 数据库原理   | 86    | B           |
| 2 | 张晨     | 数据库应用实验 | 90    | A           |
| 3 | 李晴     | 数据库原理   | 93    | A           |
| 4 | 王越     | 程序设计基础  | 88    | B           |
| 5 | 王越     | 数据结构    | 84    | B           |
| 6 | 周宁     | 程序设计基础  | 91    | A           |

#### 4.1.3 StudentCourse 的更新与删除

目的：掌握 StudentCourse 数据库中的更新与删除操作，理解条件更新、子查询

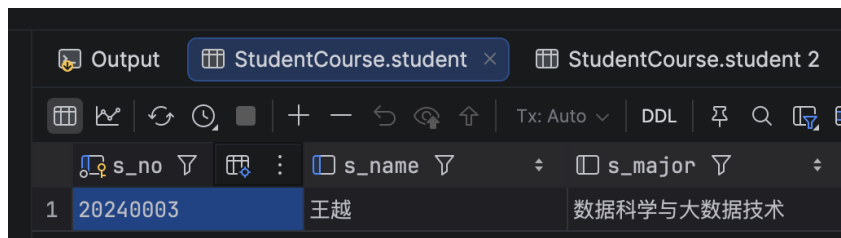
更新及按条件删除记录的实现方法。

更新与删除操作及其结果如下。

### (1) 更新学生专业

```
UPDATE student
SET s_major = '数据科学与大数据技术'
WHERE s_no = '20240003';

SELECT s_no, s_name, s_major
FROM student
WHERE s_no = '20240003';
```



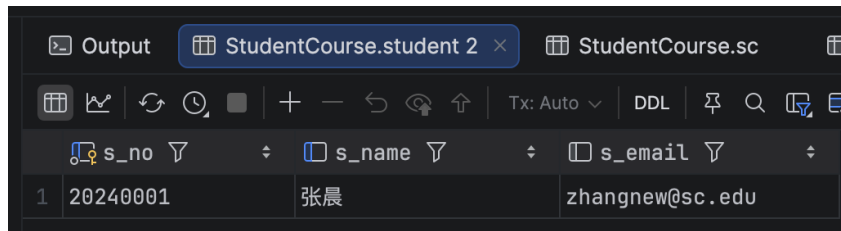
The screenshot shows a database query result in a table named 'StudentCourse.student'. The table has three columns: s\_no, s\_name, and s\_major. The result shows one row with s\_no '20240003', s\_name '王越', and s\_major '数据科学与大数据技术'.

|   | s_no     | s_name | s_major    |
|---|----------|--------|------------|
| 1 | 20240003 | 王越     | 数据科学与大数据技术 |

### (2) 更新学生邮箱

```
UPDATE student
SET s_email = 'zhangnew@sc.edu'
WHERE s_no = '20240001';

SELECT s_no, s_name, s_email
FROM student
WHERE s_no = '20240001';
```



The screenshot shows a database query result in a table named 'StudentCourse.sc'. The table has three columns: s\_no, s\_name, and s\_email. The result shows one row with s\_no '20240001', s\_name '张晨', and s\_email 'zhangnew@sc.edu'.

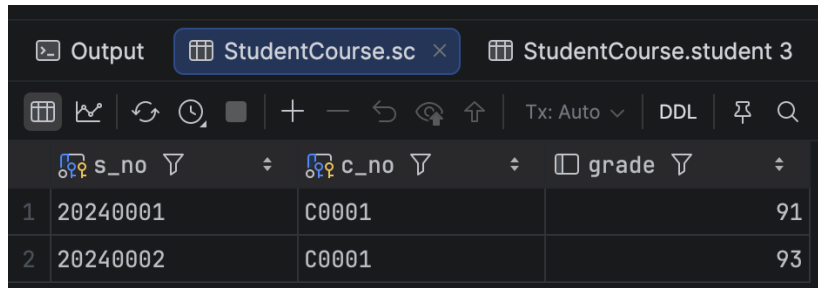
|   | s_no     | s_name | s_email         |
|---|----------|--------|-----------------|
| 1 | 20240001 | 张晨     | zhangnew@sc.edu |

### (3) 提高低于课程平均分的成绩

```
UPDATE sc
SET grade = grade + 5
WHERE c_no = 'C0001'
AND grade < (
    SELECT avg_grade
    FROM (
        SELECT AVG(grade) AS avg_grade
        FROM sc
        WHERE c_no = 'C0001'
    ) AS t
);

SELECT s_no, c_no, grade
```

```
FROM sc
WHERE c_no = 'C0001';
```

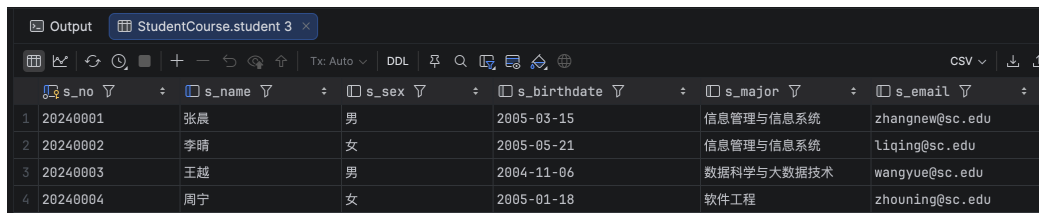


|   | s_no     | c_no  | grade |
|---|----------|-------|-------|
| 1 | 20240001 | C0001 | 91    |
| 2 | 20240002 | C0001 | 93    |

#### (4) 删除无选课记录的学生

```
DELETE FROM student
WHERE s_no = '20240005'
AND s_no NOT IN (SELECT s_no FROM sc);

SELECT *
FROM student
ORDER BY s_no;
```



|   | s_no     | s_name | s_sex | s_birthdate | s_major    | s_email         |
|---|----------|--------|-------|-------------|------------|-----------------|
| 1 | 20240001 | 张晨     | 男     | 2005-03-15  | 信息管理与信息系统  | zhangnew@sc.edu |
| 2 | 20240002 | 李晴     | 女     | 2005-05-21  | 信息管理与信息系统  | liqing@sc.edu   |
| 3 | 20240003 | 王越     | 男     | 2004-11-06  | 数据科学与大数据技术 | wangyue@sc.edu  |
| 4 | 20240004 | 周宁     | 女     | 2005-01-18  | 软件工程       | zhouning@sc.edu |

#### 4.1.4 触发器与存储过程

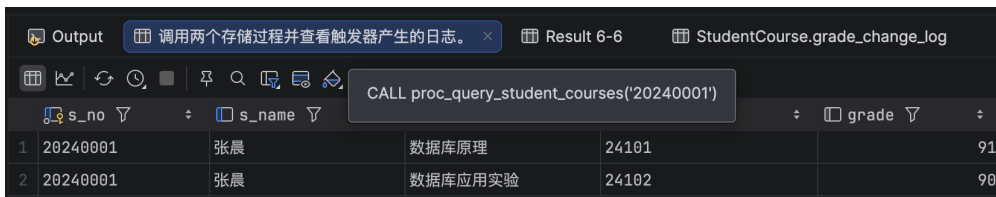
目的：掌握触发器和存储过程的建立与调用方法，实现成绩变更日志记录和参数化查询、批量更新等数据库程序设计功能。

先建立日志表、触发器和存储过程，再按下列方式调用并验证结果。

##### (1) 调用按学号查询选课情况的存储过程

```
CREATE PROCEDURE proc_query_student_courses(IN p_s_no CHAR(8))
BEGIN
    SELECT s.s_no, s.s_name, c.c_name, sc.semester, sc.grade
    FROM student AS s
    JOIN sc ON s.s_no = sc.s_no
    JOIN course AS c ON sc.c_no = c.c_no
    WHERE s.s_no = p_s_no;
END;

CALL proc_query_student_courses('20240001');
```

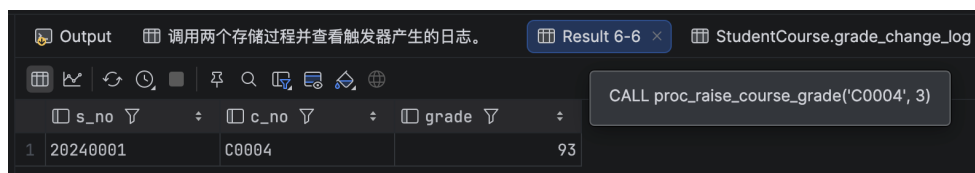


|   | s_no     | s_name | c_no    | c_name | grade |
|---|----------|--------|---------|--------|-------|
| 1 | 20240001 | 张晨     | 数据库原理   | 24101  | 91    |
| 2 | 20240001 | 张晨     | 数据库应用实验 | 24102  | 90    |

## (2) 调用统一加分存储过程并触发日志记录

```
CREATE TRIGGER trg_sc_after_update
AFTER UPDATE ON sc
FOR EACH ROW
BEGIN
    IF OLD.grade <> NEW.grade THEN
        INSERT INTO grade_change_log (s_no, c_no, old_grade, new_grade)
        VALUES (NEW.s_no, NEW.c_no, OLD.grade, NEW.grade);
    END IF;
END;

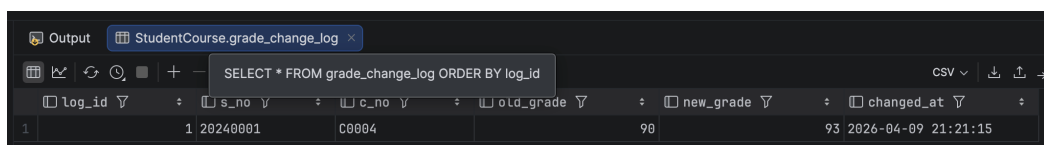
CALL proc_raise_course_grade('C0004', 3);
```



|   | s_no     | c_no  | grade |
|---|----------|-------|-------|
| 1 | 20240001 | C0004 | 93    |

## (3) 查看触发器写入的成绩变更日志

```
SELECT *
FROM grade_change_log
ORDER BY log_id;
```



|   | log_id | s_no     | c_no  | old_grade | new_grade | changed_at          |
|---|--------|----------|-------|-----------|-----------|---------------------|
| 1 | 1      | 20240001 | C0004 | 90        | 93        | 2026-04-09 21:21:15 |

### 4.1.5 SPJ 查询

目的：掌握 SPJ 数据库中的投影、条件查询、连接查询、分组统计、集合筛选和嵌套查询等方法。

SPJ 数据库中的典型查询如下。

#### (1) 查询供应商姓名和所在城市

```
USE SPJ;

SELECT SNAME AS supplier_name, CITY AS supplier_city
FROM S;
```

Output 查询供应商姓名和所在城市。 × 查询

供应商姓名 所在城市

|   |     |    |
|---|-----|----|
| 1 | 精益  | 天津 |
| 2 | 盛锡  | 北京 |
| 3 | 东方红 | 北京 |
| 4 | 丰泰盛 | 天津 |
| 5 | 为民  | 上海 |

## (2) 查询零件名称、颜色和重量

```
SELECT PNAME AS part_name, COLOR, WEIGHT
FROM P;
```

Output 查询零件名称、颜色和重量。 × 查询供应商 S1 所参

零件名称 颜色 重量

|   |     |   |    |
|---|-----|---|----|
| 1 | 螺母  | 红 | 12 |
| 2 | 螺栓  | 绿 | 17 |
| 3 | 螺丝刀 | 蓝 | 14 |
| 4 | 螺丝刀 | 红 | 14 |
| 5 | 凸轮  | 蓝 | 40 |
| 6 | 齿轮  | 红 | 30 |

## (3) 查询供应商 S1 参与的工程代码

```
SELECT DISTINCT JN0 AS project_no
FROM SPJ
WHERE SN0 = 'S1';
```

Output 查询供应商 S1 所参与的工程代码。 ×

工程代码

|   |    |
|---|----|
| 1 | J1 |
| 2 | J3 |
| 3 | J4 |
| 4 | J2 |

工程代码: varchar(10)

## (4) 统计工程 J2 的零件数量

```
SELECT P.PNAME AS part_name, SUM(SPJ.QTY) AS total_qty
```

```
FROM SPJ
JOIN P ON SPJ.PNO = P.PNO
WHERE SPJ.JNO = 'J2'
GROUP BY SPJ.PNO, P.PNAME;
```

Output 统计工程 J2 使用的零件总数量。 ×

SELECT P.PNAME

|   | 零件名称 | 总数量 |
|---|------|-----|
| 1 | 螺栓   | 100 |
| 2 | 螺丝刀  | 200 |
| 3 | 凸轮   | 100 |
| 4 | 齿轮   | 200 |

(5) 查询上海供应商参与的工程名称

```
SELECT DISTINCT J.JNAME AS project_name
FROM SPJ
JOIN S ON SPJ.SNO = S.SNO
JOIN J ON SPJ.JNO = J.JNO
WHERE S.CITY = '上海';
```

Output 查询由上海供应商供应的零件代码。 ×

零件代码

|   |    |
|---|----|
| 1 | P2 |
| 2 | P3 |
| 3 | P6 |

(6) 查询上海供应商供应的零件代码

```
SELECT DISTINCT PNO AS part_no
FROM SPJ
WHERE SNO IN (SELECT SNO FROM S WHERE CITY = '上海');
```

Output 查询由上海供应商参与的工程名称。 ×

工程名称

|   |     |
|---|-----|
| 1 | 造船厂 |
| 2 | 三建  |
| 3 | 一汽  |

(7) 查询不使用天津供应商零件的工程代码

```
SELECT JNO AS project_no
FROM J
WHERE JNO NOT IN (
    SELECT DISTINCT SPJ.JNO
    FROM SPJ
    JOIN S ON SPJ.SNO = S.SNO
    WHERE S.CITY = '天津'
);
```

Output 查询没有使用天津供应商零件的工程代码。 ×

工程代码

|   |    |
|---|----|
| 1 | J5 |
| 2 | J6 |
| 3 | J7 |

#### 4.1.6 SPJ 的 DML 操作

目的：掌握 SPJ 数据库中的更新、修改供应关系、删除并恢复供应商及新增供应记录等操作方法。

SPJ 中的更新与删除操作如下。

##### (1) 把红色零件统一修改为蓝色

```
UPDATE P
SET COLOR = '蓝'
WHERE COLOR = '红';

SELECT * FROM P ORDER BY PNO;
```

Output SPJ.P × 先查看记录，再把 (S5, P6, J2) 这条供应关系改成 S3 供应。

|   | PNO | PNAME | COLOR | WEIGHT |
|---|-----|-------|-------|--------|
| 1 | P1  | 螺母    | 蓝     | 12     |
| 2 | P2  | 螺栓    | 绿     | 17     |
| 3 | P3  | 螺丝刀   | 蓝     | 14     |
| 4 | P4  | 螺丝刀   | 蓝     | 14     |
| 5 | P5  | 凸轮    | 蓝     | 40     |
| 6 | P6  | 齿轮    | 蓝     | 30     |

##### (2) 修改供应关系中的供应商编号

```
UPDATE SPJ
SET SNO = 'S3'
WHERE SNO = 'S5' AND PNO = 'P6' AND JNO = 'J2';
```

```
SELECT * FROM SPJ
WHERE PNO = 'P6' AND JNO = 'J2';
```

Output 先查看记录, 再把 (S5, P6, J2) 这条供应关系改成 S3 供应。 SPJ.SP

|   | SNO | PNO | JNO | QTY |
|---|-----|-----|-----|-----|
| 1 | S5  | P6  | J2  | 200 |

### (3) 删除供应商 S2 及其供应记录

```
DELETE FROM SPJ WHERE SNO = 'S2';
DELETE FROM S WHERE SNO = 'S2';

SELECT * FROM S ORDER BY SNO;
```

Output SPJ.S SPJ.SPJ 2

|   | SNO | STATUS | CITY  |
|---|-----|--------|-------|
| 1 | S1  | 精益     | 20 天津 |
| 2 | S3  | 东方红    | 30 北京 |
| 3 | S4  | 丰泰盛    | 20 天津 |
| 4 | S5  | 为民     | 30 上海 |

### (4) 恢复供应商并新增供应记录

```
INSERT INTO S (SNO, SNAME, STATUS, CITY)
VALUES ('S2', '盛锡', 10, '北京');

INSERT INTO SPJ (SNO, PNO, JNO, QTY)
VALUES ('S2', 'P4', 'J6', 200);

SELECT *
FROM SPJ
WHERE SNO = 'S2';
```

Output SPJ.S SPJ.SPJ 2

|   | SNO | PNO | JNO | QTY |
|---|-----|-----|-----|-----|
| 1 | S2  | P4  | J6  | 200 |

#### 4.1.7 嵌入式 SQL 访问数据库

目的：掌握在宿主语言中连接数据库、执行参数化 SQL 并输出结果的方法，理解嵌入式 SQL 的基本实现思路。

在 Python 宿主语言中，通过参数化查询分别访问 StudentCourse 和 SPJ，程

序片段如下。

### (1) 宿主语言连接数据库并执行参数化查询

```
def connect(database: str):
    return pymysql.connect(
        host="127.0.0.1",
        port=3306,
        user="root",
        password="123456",
        database=database,
        charset="utf8mb4",
        cursorclass=pymysql.cursors.DictCursor,
    )

def query_student_course(student_no: str) -> None:
    sql = """
        SELECT s.s_no, s.s_name, c.c_name, sc.semester, sc.grade
        FROM student AS s
        JOIN sc ON s.s_no = sc.s_no
        JOIN course AS c ON sc.c_no = c.c_no
        WHERE s.s_no = %s
        ORDER BY c.c_no
    """

def query_supplier_by_city(city: str) -> None:
    sql = """
        SELECT SNO, SNAME, STATUS, CITY
        FROM S
        WHERE CITY = %s
        ORDER BY SNO
    """
```

### (2) 程序运行结果

```
Code/DB/Lab [🐧 orbstack][📦 v0.1.0][🐧 v3.13.12(Lab)]
) uv pip install pymysql
Resolved 1 package in 344ms
Prepared 1 package in 71ms
Installed 1 package in 5ms
+ pymysql=1.1.2

Code/DB/Lab [🐧 orbstack][📦 v0.1.0][🐧 v3.13.12(Lab)]
) uv run Lab02/manual/embedded_sql_demo.py

StudentCourse query result
-----
{'s_no': '20240001', 's_name': '张晨', 'c_name': '数据库原理', 'semester': '24101', 'grade': 91}
{'s_no': '20240001', 's_name': '张晨', 'c_name': '数据库应用实验', 'semester': '24102', 'grade': 93}

SPJ query result
-----
{'SNO': 'S5', 'SNAME': '为民', 'STATUS': 30, 'CITY': '上海'}
```

## 4.2 实验总结与心得体会

本次实验围绕数据查询与 DML 操作完成了 StudentCourse 与 SPJ 两个数据库上的多类实验任务。通过简单查询和关系查询，进一步掌握了条件筛选、排序、分组、连接、子查询和聚合统计等基本 SQL 方法；通过更新、删除、触发器和存储过程，进一步理解了数据库中数据修改与程序化处理的结合方式；通过宿主语言访问数据库，又加深了对嵌入式 SQL 实现流程的认识。

实验结果表明，数据库应用不仅要求能够写出正确的查询语句，还要求在更新和删除时兼顾数据一致性，在数据库程序设计中兼顾可维护性和可验证性。只有把查询分析、DML 操作、触发器与存储过程以及宿主语言接口结合起来，才能更完整地理解数据库系统在实际应用中的工作方式。