信息与软件工程学院

上机实验报告册

专　　业_　　-　　_

　　级_____　　_____

学　　号___　　-　　___

姓　　名_____　　_____

课程名称 数据库原理及应用

教　　师　　喻佳

学　　期 2025—2026 学年第 2 学期

年　　月　　日

信息与软件工程学院上机实验报告

（第 1 次）

一、实验名称

实验一：数据库 DDL 实验

二、实验目的及要求

2.1 知识能力目标

1. 掌握数据库系统的基本概念和基本知识，能够使用 SQL 完成数据库和基本表的定义、操纵与控制。
2. 理解数据库完整性机制，能够运用主键、唯一约束、外键和检查约束实现实体完整性、参照完整性和用户自定义完整性。
3. 围绕 StudentCourse 与 SPJ 两个关系数据库，完成建库、建表、结构修改、索引处理和结果验证，形成数据库初步建立与设计能力。

2.2 素养目标

1. 按照工程认证“研究”要求，围绕实验目标设计执行步骤，分析运行结果，并据此形成清晰结论。
2. 按照工程认证“使用现代工具”要求，能够选用 Docker、MySQL 和 DataGrip 等工具完成数据库环境搭建、脚本执行与结果验证。
3. 按照工程认证“工程与可持续发展”要求，理解数据库对象设计应兼顾规范性、可维护性和后续复用需求，为后续实验继续使用同一数据库奠定基础。

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 建立 StudentCourse 数据库和基本表

目的：完成 StudentCourse 数据库及其三张基本表的定义，为后续完整性约束、索引处理和数据验证提供结构基础。

核心代码如下：

```
CREATE DATABASE IF NOT EXISTS StudentCourse
DEFAULT CHARACTER SET utf8mb4
```

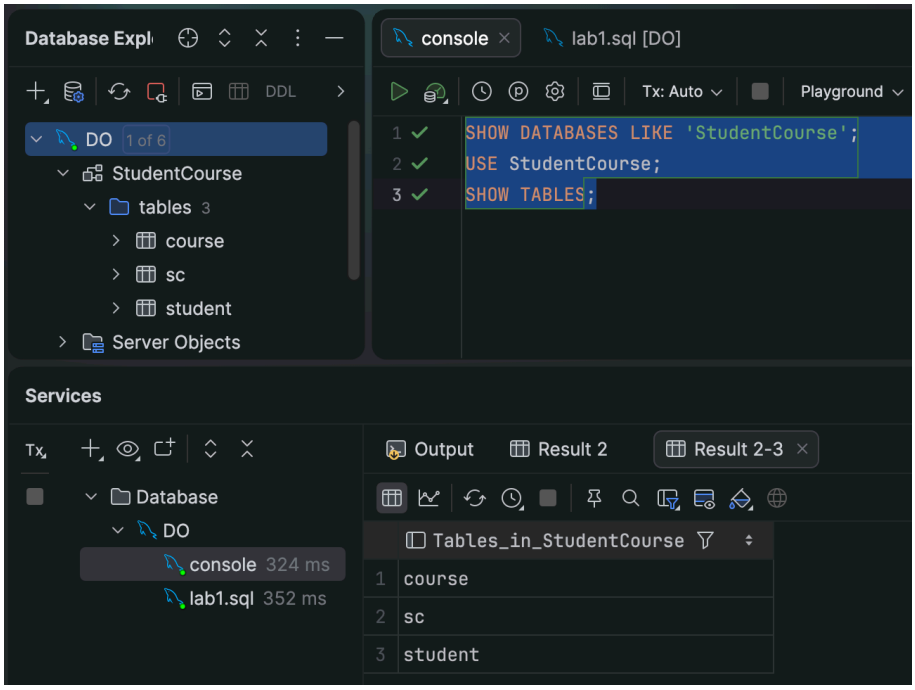
```
COLLATE utf8mb4_unicode_ci;

USE StudentCourse;

CREATE TABLE student (
  s_no CHAR(8),
  s_name VARCHAR(20),
  s_sex CHAR(6),
  s_birthdate DATE,
  s_major VARCHAR(40),
  CONSTRAINT pk_student PRIMARY KEY (s_no),
  CONSTRAINT uk_student_s_name UNIQUE (s_name),
  CONSTRAINT chk_student_s_sex CHECK (s_sex IN ('男', '女'))
);

CREATE TABLE course (
  c_no CHAR(5),
  c_name VARCHAR(40) NOT NULL,
  c_credit SMALLINT,
  c_pno CHAR(5),
  CONSTRAINT pk_course PRIMARY KEY (c_no),
  CONSTRAINT chk_course_c_credit CHECK (c_credit > 0),
  CONSTRAINT fk_course_c_pno FOREIGN KEY (c_pno) REFERENCES course(c_no)
);

CREATE TABLE sc (
  s_no CHAR(8),
  c_no CHAR(5),
  grade SMALLINT,
  semester CHAR(5),
  teaching_class CHAR(8),
  CONSTRAINT pk_sc PRIMARY KEY (s_no, c_no),
  CONSTRAINT chk_sc_grade CHECK (grade IS NULL OR (grade >= 0 AND grade <=
     100)),
  CONSTRAINT fk_sc_s_no FOREIGN KEY (s_no) REFERENCES student(s_no),
  CONSTRAINT fk_sc_c_no FOREIGN KEY (c_no) REFERENCES course(c_no)
);
```



4.1.2 修改 StudentCourse 表结构并处理索引

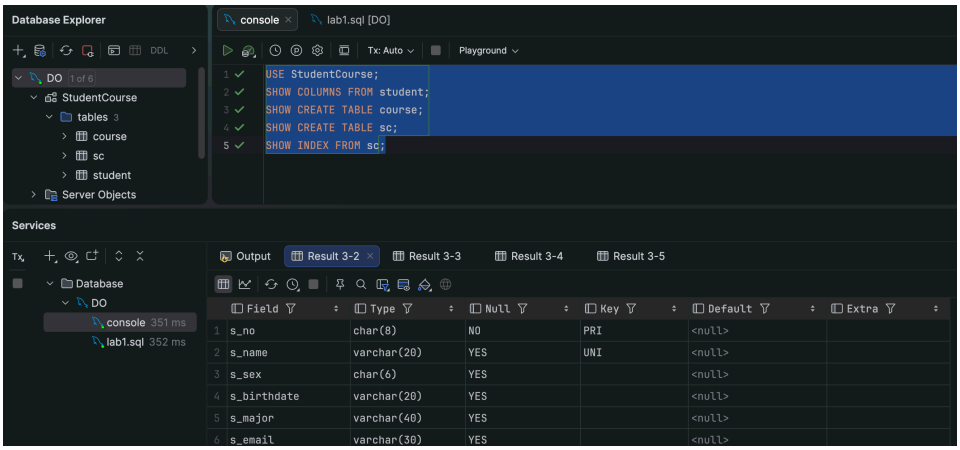
目的：通过 ALTER TABLE 与索引语句完成字段调整、补充约束和索引管理，体现数据库对象演化与性能辅助设计过程。

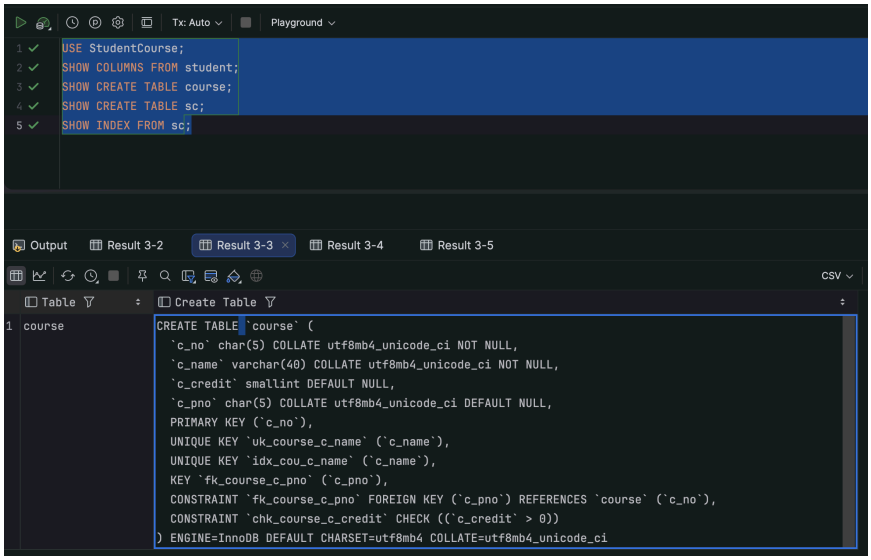
核心代码如下：

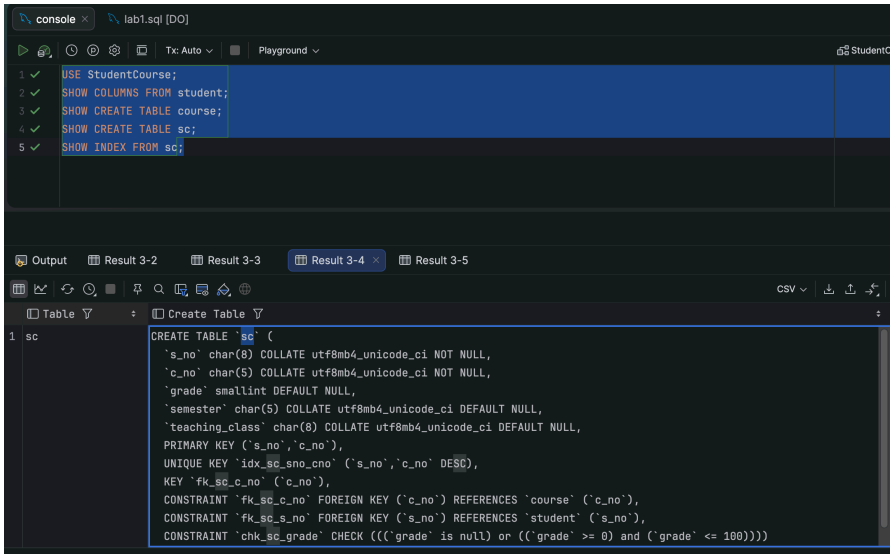
```
ALTER TABLE student ADD s_email VARCHAR(30);
ALTER TABLE student MODIFY s_birthdate VARCHAR(20);
ALTER TABLE course ADD CONSTRAINT uk_course_c_name UNIQUE (c_name);

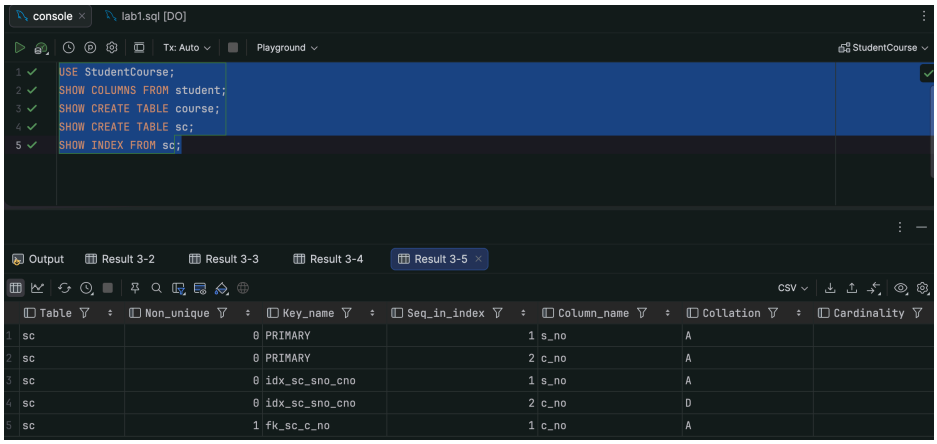
CREATE UNIQUE INDEX idx_stu_s_name ON student(s_name);
CREATE UNIQUE INDEX idx_cou_c_name ON course(c_name);
CREATE UNIQUE INDEX idx_sc_c_co ON sc(s_no ASC, c_no DESC);

ALTER TABLE sc RENAME INDEX idx_sc_c_co TO idx_sc_sno_cno;
ALTER TABLE student DROP INDEX idx_stu_s_name;
```









4.1.3 装入 StudentCourse 基础数据并验证选课关系

目的：向三张基本表装入实验数据，并通过连接查询验证选课记录、课程信息和学生信息之间的参照关系。

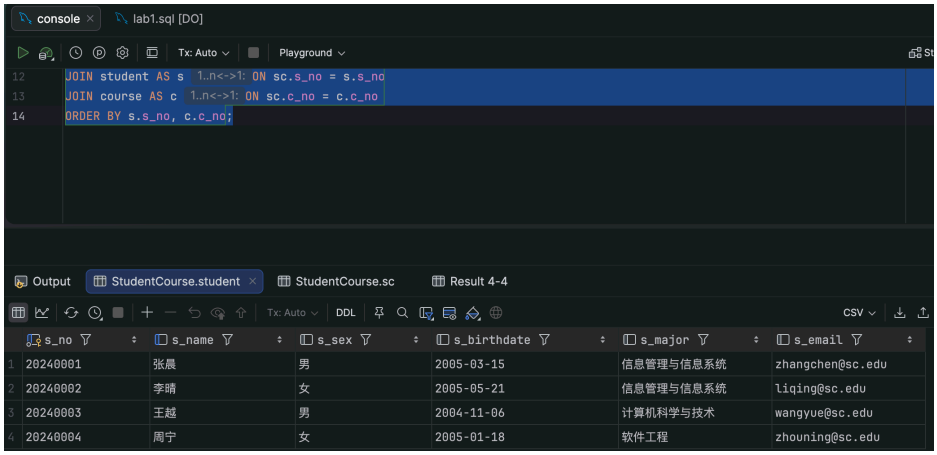
核心代码如下：

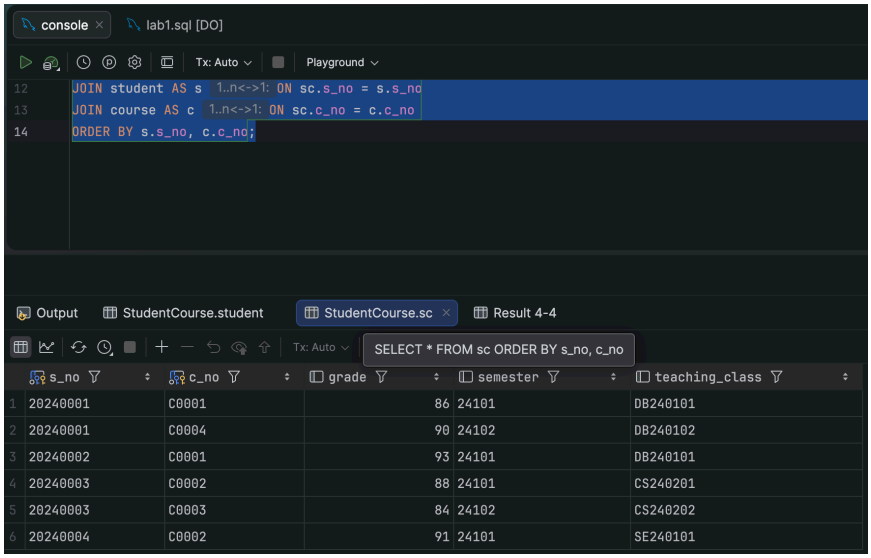
```
INSERT INTO student (s_no, s_name, s_sex, s_birthdate, s_major, s_email)
    VALUES (...);
```

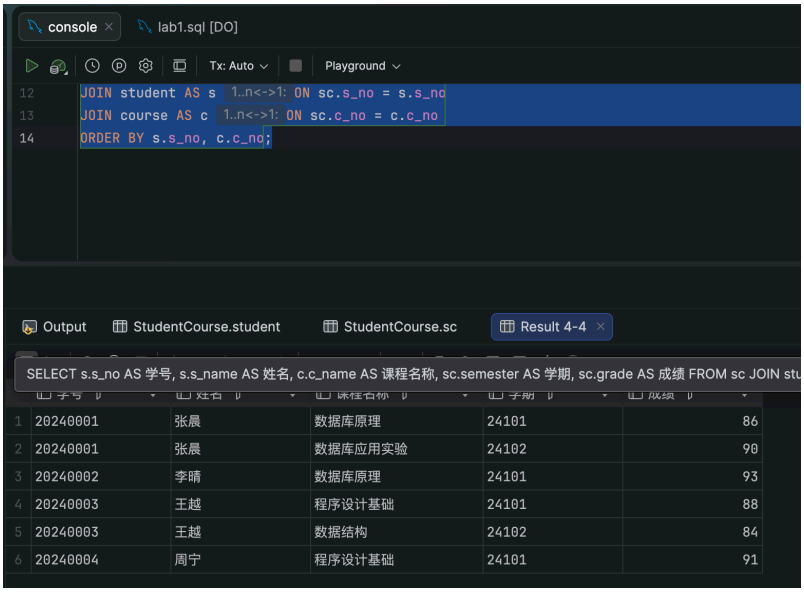
```sql
INSERT INTO course (c_no, c_name, c_credit, c_pno) VALUES (...);
INSERT INTO sc (s_no, c_no, grade, semester, teaching_class) VALUES (...);

SELECT * FROM student ORDER BY s_no;
SELECT * FROM sc ORDER BY s_no, c_no;

SELECT
  s.s_no AS student_no,
  s.s_name AS student_name,
  c.c_name AS course_name,
  sc.semester,
  sc.grade
FROM sc
JOIN student AS s ON sc.s_no = s.s_no
JOIN course AS c ON sc.c_no = c.c_no
ORDER BY s.s_no, c.c_no;
```







4.1.4 建立 SPJ 数据库和基本表

目的：完成 SPJ 经典供应商-零件-工程数据库的结构定义，突出联合主键、外键和检查约束在多表场景中的应用。

核心代码如下：

```
CREATE DATABASE IF NOT EXISTS SPJ
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE SPJ;

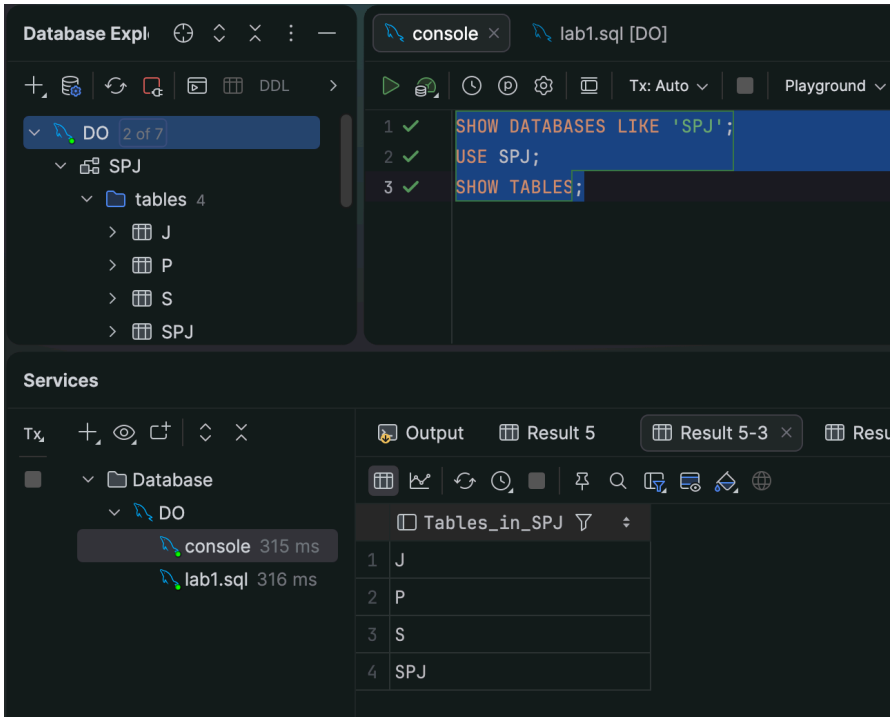
CREATE TABLE S (
  SNO VARCHAR(10),
  SNAME VARCHAR(50) NOT NULL,
  STATUS INT NOT NULL,
  CITY VARCHAR(50) NOT NULL,
  CONSTRAINT pk_s PRIMARY KEY (SNO),
  CONSTRAINT chk_s_status CHECK (STATUS > 0)
);

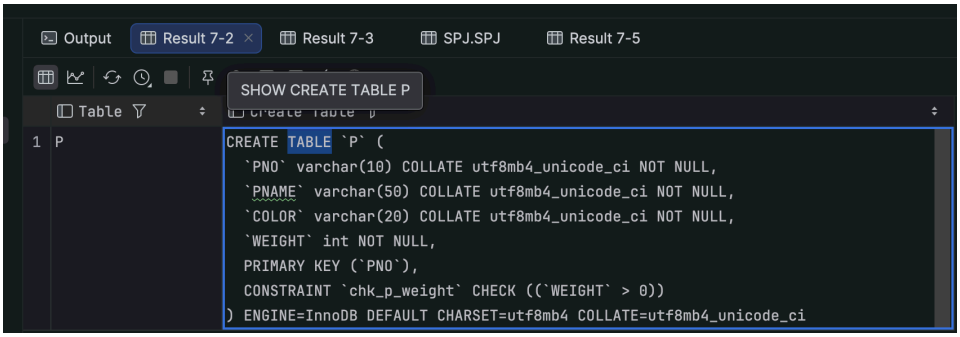
CREATE TABLE P (
  PNO VARCHAR(10),
  PNAME VARCHAR(50) NOT NULL,
  COLOR VARCHAR(20) NOT NULL,
  WEIGHT INT NOT NULL,
  CONSTRAINT pk_p PRIMARY KEY (PNO),
  CONSTRAINT chk_p_weight CHECK (WEIGHT > 0)
);

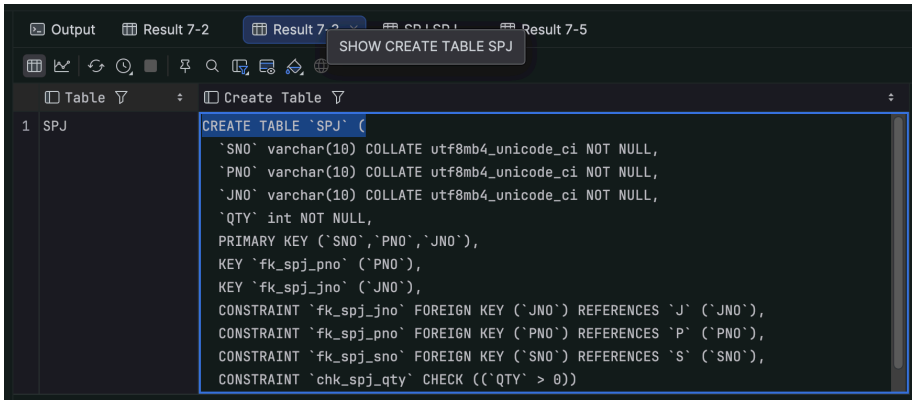
CREATE TABLE J (
  JNO VARCHAR(10),
  JNAME VARCHAR(50) NOT NULL,
  CITY VARCHAR(50) NOT NULL,
```

```
  CONSTRAINT pk_j PRIMARY KEY (JNO)
);

CREATE TABLE SPJ (
  SNO VARCHAR(10) NOT NULL,
  PNO VARCHAR(10) NOT NULL,
  JNO VARCHAR(10) NOT NULL,
  QTY INT NOT NULL,
  CONSTRAINT pk_spj PRIMARY KEY (SNO, PNO, JNO),
  CONSTRAINT chk_spj_qty CHECK (QTY > 0),
  CONSTRAINT fk_spj_sno FOREIGN KEY (SNO) REFERENCES S(SNO),
  CONSTRAINT fk_spj_pno FOREIGN KEY (PNO) REFERENCES P(PNO),
  CONSTRAINT fk_spj_jno FOREIGN KEY (JNO) REFERENCES J(JNO)
);
```







4.1.5 装入 SPJ 基础数据并验证供应关系

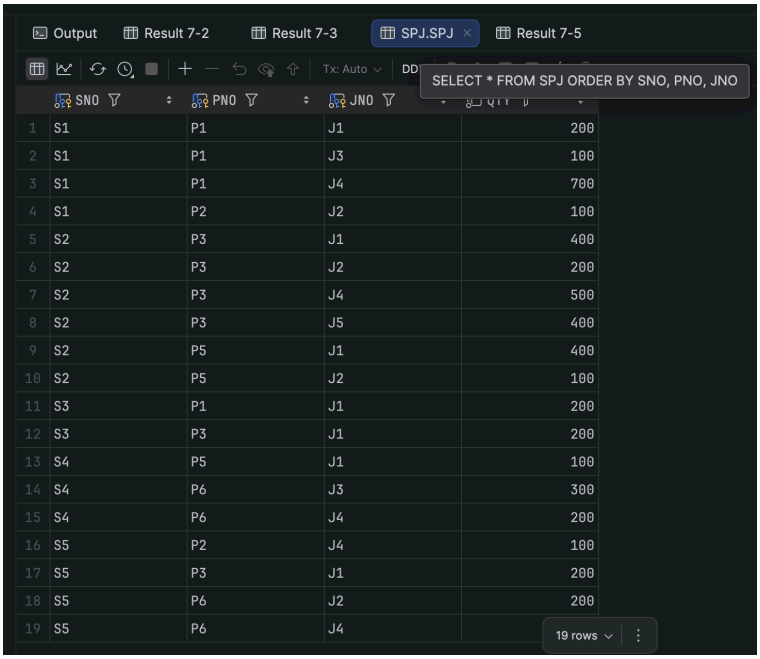
目的：向 SPJ 数据库装入基础数据，通过多表连接验证供应商、零件和工程项目之间的供应关系是否建立正确。

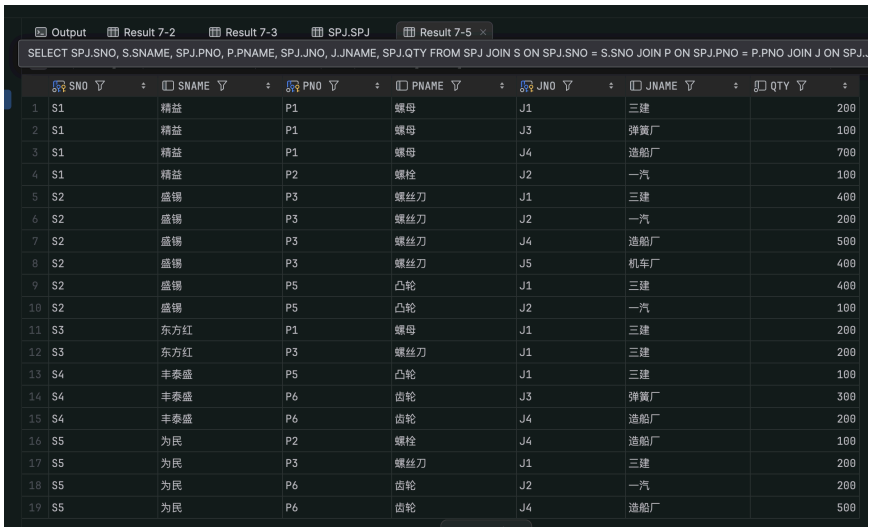
核心代码如下：

```sql
INSERT INTO S (SNO, SNAME, STATUS, CITY) VALUES (...);
INSERT INTO P (PNO, PNAME, COLOR, WEIGHT) VALUES (...);
INSERT INTO J (JNO, JNAME, CITY) VALUES (...);
INSERT INTO SPJ (SNO, PNO, JNO, QTY) VALUES (...);

SELECT * FROM SPJ ORDER BY SNO, PNO, JNO;

SELECT
  SPJ.SNO,
  S.SNAME,
  SPJ.PNO,
  P.PNAME,
  SPJ.JNO,
  J.JNAME,
  SPJ.QTY
FROM SPJ
JOIN S ON SPJ.SNO = S.SNO
JOIN P ON SPJ.PNO = P.PNO
JOIN J ON SPJ.JNO = J.JNO
ORDER BY SPJ.SNO, SPJ.PNO, SPJ.JNO;
```





4.2 实验总结与心得体会

本次实验围绕数据库 DDL 操作完成了两个典型数据库的建立与验证。通过 StudentCourse 数据库的设计与调整，进一步掌握了数据库、基本表、主键、唯一约束、外键、检查约束以及索引的创建与修改方法；通过 SPJ 数据库的建立与关联验证，进一步理解了多表结构下联合主键和参照完整性的实现方式。

实验过程中可以看出，数据库设计不仅要保证对象能够成功创建，还要关注约束是否完整、结构修改是否规范、查询结果是否与预期一致。只有把建库、建表、结构调整、数据装入和结果验证作为一个完整过程来处理，才能为后续 DML、视图、授权和备份恢复等实验提供稳定基础。

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、DataGrip 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

```sql
USE StudentCourse;

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	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

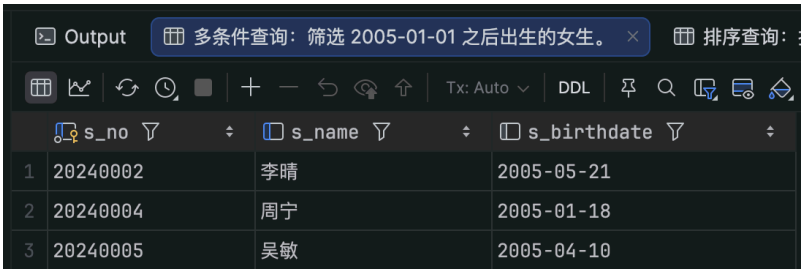
```sql
SELECT s_no, s_name, s_major
FROM student
WHERE s_major = '信息管理与信息系统';
```



	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

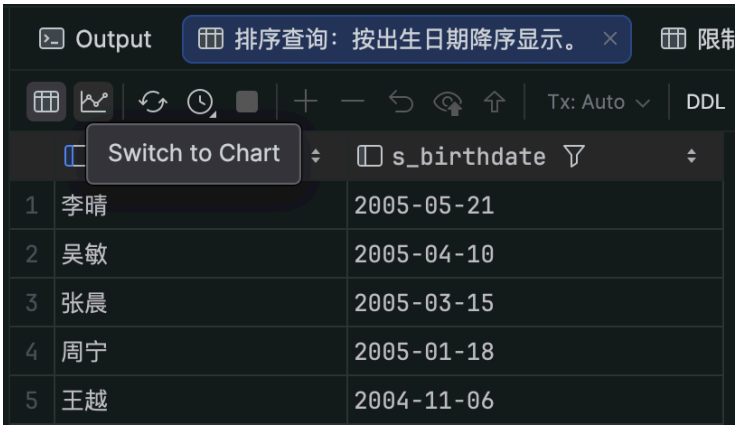
```sql
SELECT s_no, s_name, s_birthdate
FROM student
WHERE s_sex = '女' AND s_birthdate > '2005-01-01';
```



	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

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



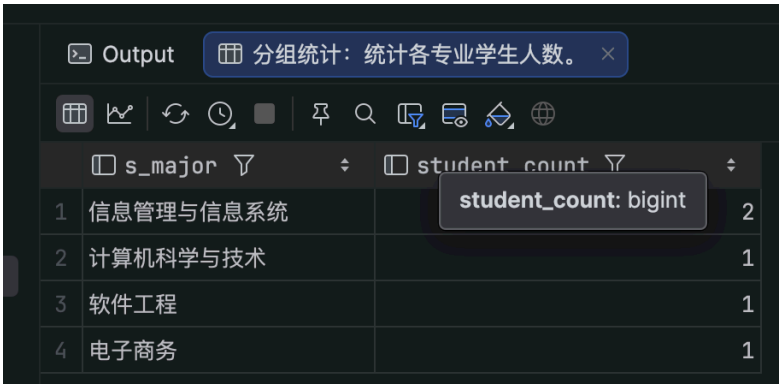
(5) 限制返回记录数

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



(6) 分组统计

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



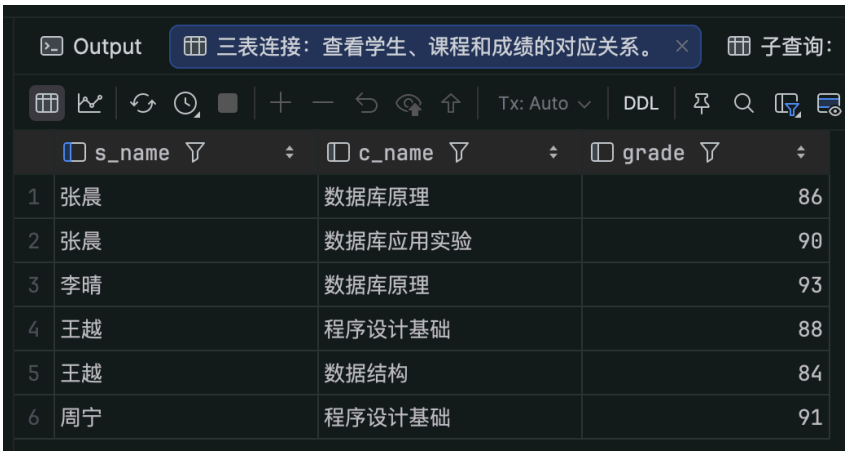
4.1.2 StudentCourse 关系查询与连接

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

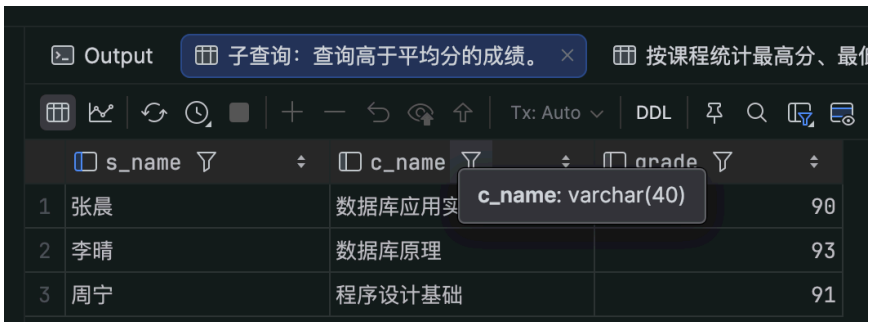
```
SELECT s.s_name, c.c_name, sc.grade
FROM student AS s
JOIN sc ON s.s_no = sc.s_no
JOIN course AS c ON sc.c_no = c.c_no;
```



	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
);
```



	s_name	c_name	grade
1	张晨	数据库应用实	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

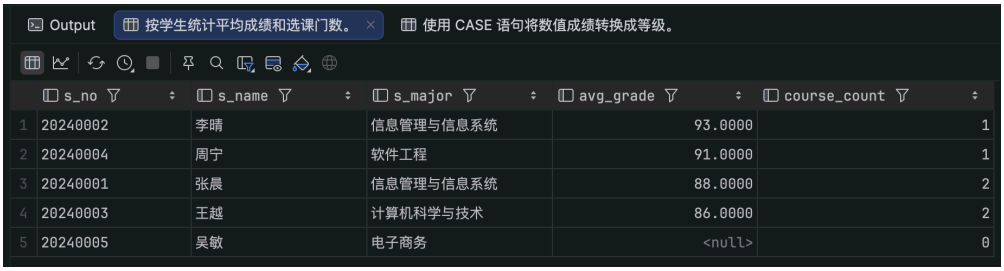
```
SELECT c.c_name,
       MAX(sc.grade) AS max_grade,
       MIN(sc.grade) AS min_grade,
       AVG(sc.grade) AS avg_grade,
       COUNT(sc.s_no) AS student_count
FROM course AS c
LEFT JOIN sc ON c.c_no = sc.c_no
GROUP BY c.c_no, c.c_name;
```



	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

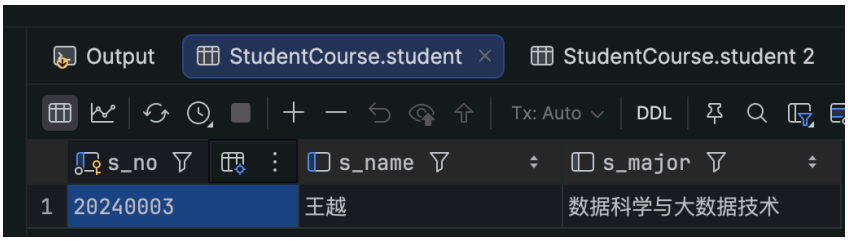
更新及按条件删除记录的实现方法。

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';

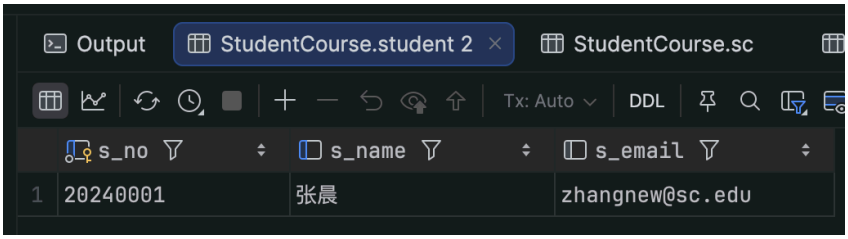
SELECT s_no, s_name, s_major
FROM student
WHERE s_no = '20240003';
```



(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';
```

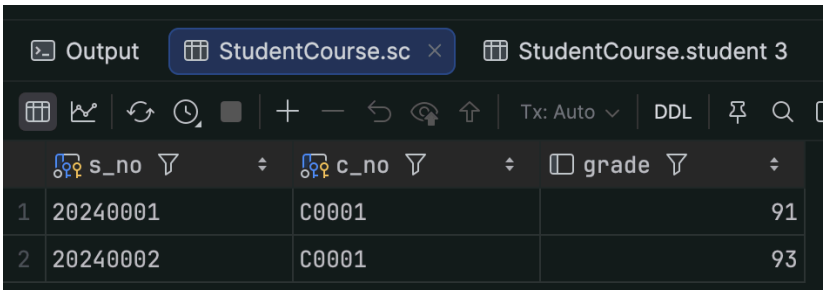


(3) 提高低于课程平均分的成绩

```
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
```

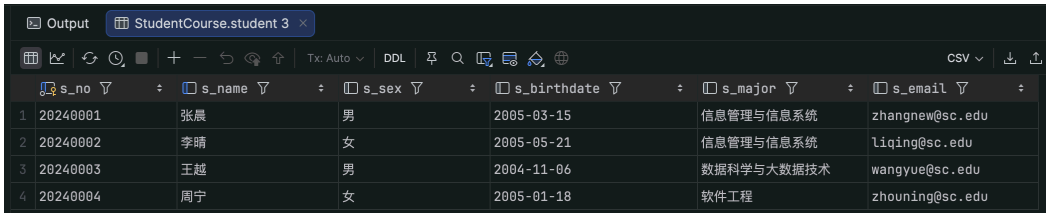
```
FROM sc
WHERE c_no = 'C0001';
```



(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;
```



4.1.4 触发器与存储过程

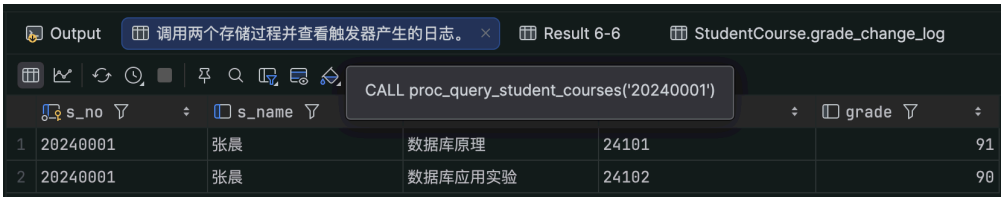
目的：掌握触发器和存储过程的建立与调用方法，实现成绩变更日志记录和参数化查询、批量更新等数据库程序设计功能。

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');
```



(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);
```



(3) 查看触发器写入的成绩变更日志

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



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;

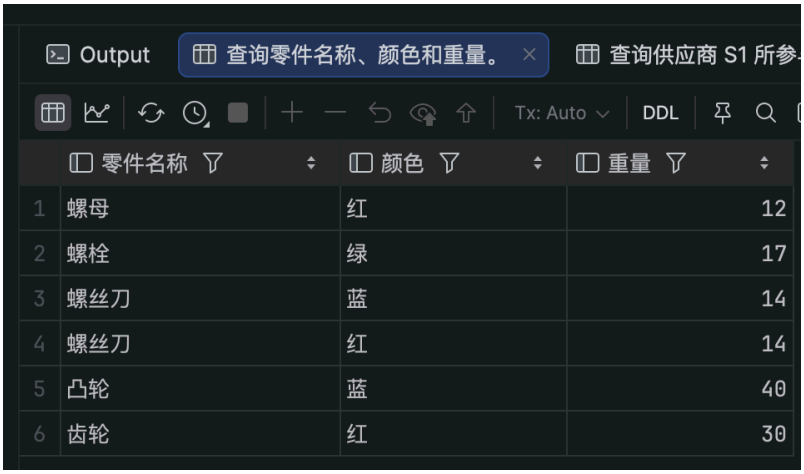
SELECT SNAME AS supplier_name, CITY AS supplier_city
FROM S;
```



	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

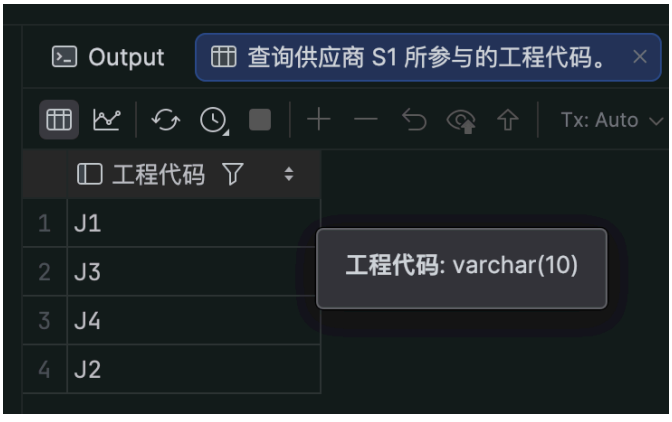
```
SELECT PNAME AS part_name, COLOR, WEIGHT
FROM P;
```



	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 JNO AS project_no
FROM SPJ
WHERE SNO = 'S1';
```

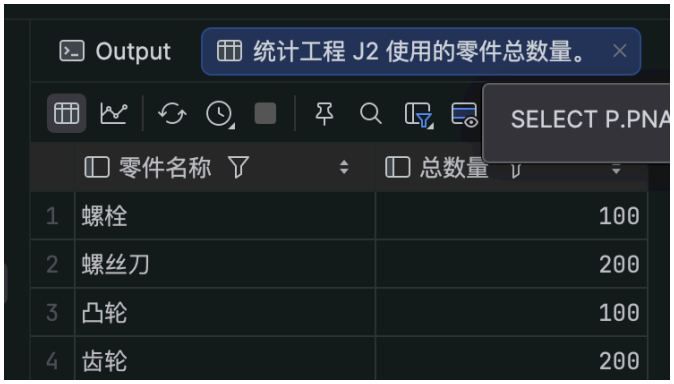


	工程代码
1	J1
2	J3
3	J4
4	J2

(4) 统计工程 J2 的零件数量

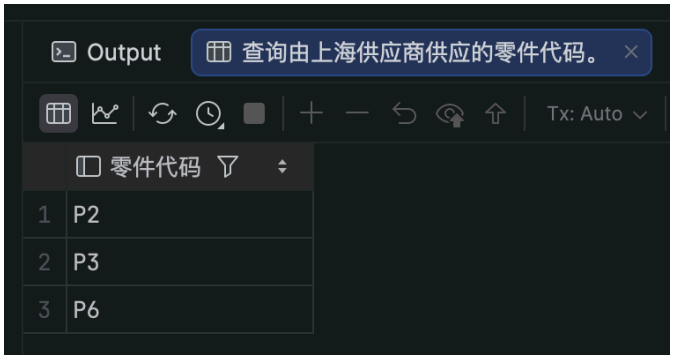
```
SELECT P.PNAME AS part_name, SUM(SPJ.QTY) AS total_qty
```

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



(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';
```



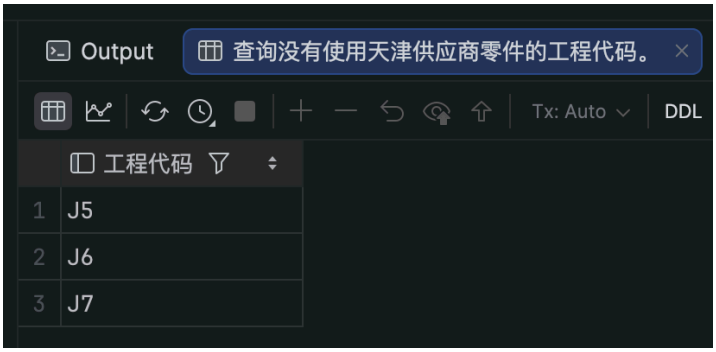
(6) 查询上海供应商供应的零件代码

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



(7) 查询不使用天津供应商零件的工程代码

```sql
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'
);
```



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

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

SELECT * FROM P ORDER BY PNO;
```



(2) 修改供应关系中的供应商编号

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

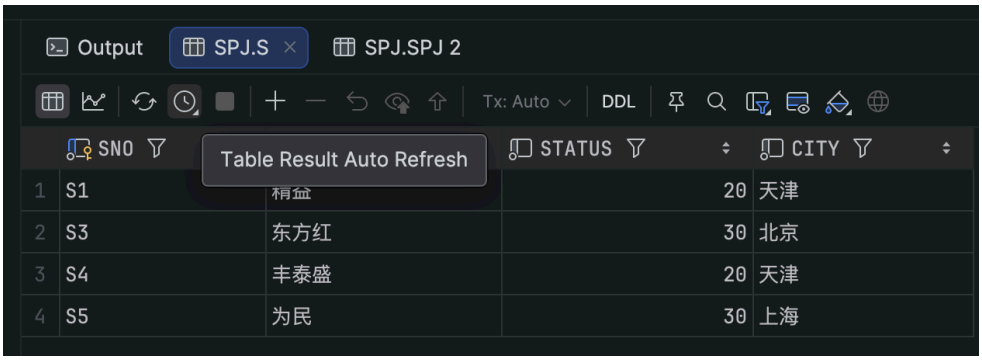
```
SELECT * FROM SPJ
WHERE PNO = 'P6' AND JNO = 'J2';
```



(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;
```

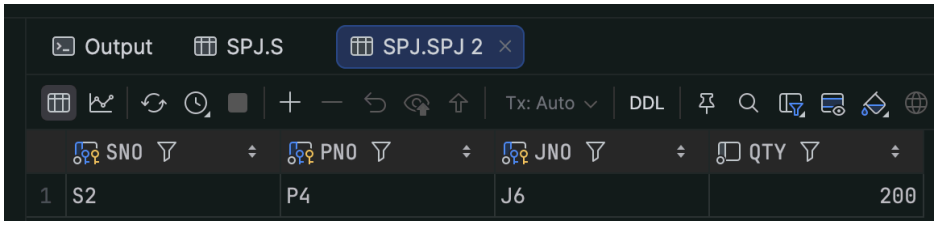


(4) 恢复供应商并新增供应记录

```
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';
```



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

```python
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。

信息与软件工程学院上机实验报告

（第 3 次）

一、实验名称

实验三：数据的 DCL 操作

二、实验目的及要求

2.1 知识能力目标

1. 掌握视图的创建与使用方法，能够利用视图对基表数据进行有目的的封装与展示。
2. 掌握数据库用户、角色、授权与回收权限等安全控制方法，理解数据库安全性的基本实现机制。
3. 掌握数据库备份与恢复的基本流程，能够结合脚本工具完成数据库导出、恢复与结果验证。

2.2 素养目标

1. 按照工程认证“研究”要求，围绕视图、安全控制和恢复验证设计实验步骤，并根据执行结果判断方案是否正确。
2. 按照工程认证“使用现代工具”要求，能够综合使用 Docker、MySQL、DataGrip 与 Shell 脚本完成权限管理、备份恢复与结果核验。
3. 按照工程认证“工程与可持续发展”要求，理解数据库安全配置与备份恢复对系统稳定运行、数据保护和持续维护的重要意义。

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 视图

目的：掌握在 StudentCourse 数据库上创建简单视图和连接视图的方法，并验证视图对象及查询结果是否正确。

(1) 创建 `is_student` 与 `student_grade_overview` 视图并查询结果

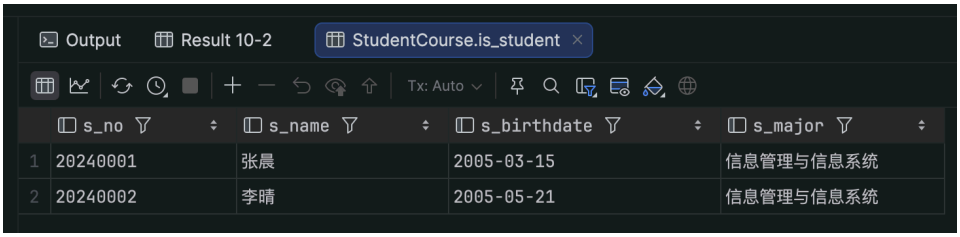
```
USE StudentCourse;
```

```sql
DROP VIEW IF EXISTS student_grade_overview;
DROP VIEW IF EXISTS is_student;

CREATE VIEW is_student AS
SELECT s_no, s_name, s_birthdate, s_major
FROM student
WHERE s_major = '信息管理与信息系统'
WITH CHECK OPTION;

CREATE VIEW student_grade_overview AS
SELECT s.s_no, s.s_name, s.s_major,
       c.c_no, c.c_name, c.c_credit,
       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;

SHOW FULL TABLES WHERE Table_type = 'VIEW';
SELECT * FROM is_student ORDER BY s_no;
```



	s_no	s_name	s_birthdate	s_major
1	20240001	张晨	2005-03-15	信息管理与信息系统
2	20240002	李晴	2005-05-21	信息管理与信息系统

4.1.2 SPJ 视图

目的：掌握在 SPJ 数据库上建立综合视图的方法，并验证视图是否正确整合供应商、零件、工程和数量信息。

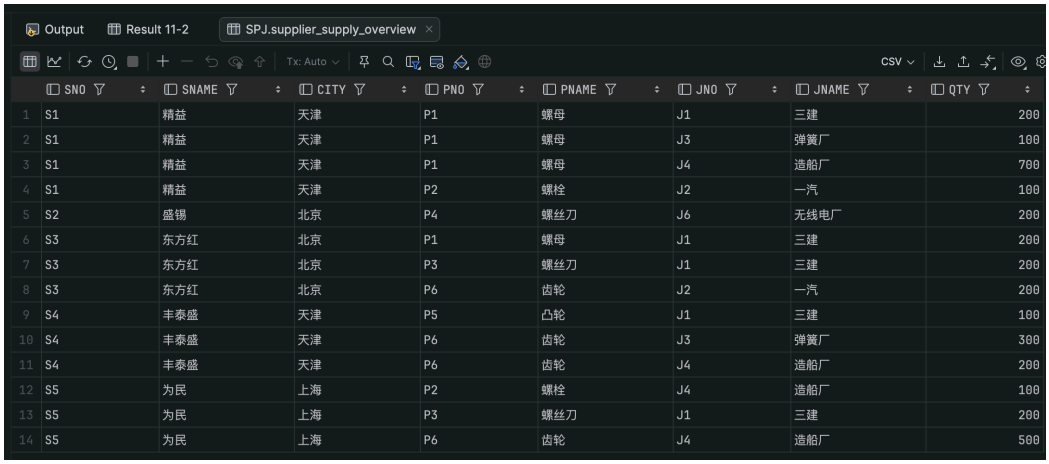
(1) 创建 supplier_supply_overview 视图并查询结果

```sql
USE SPJ;

DROP VIEW IF EXISTS supplier_supply_overview;

CREATE VIEW supplier_supply_overview AS
SELECT S.SNO, S.SNAME, S.CITY,
       P.PNO, P.PNAME,
       J.JNO, J.JNAME,
       SPJ.QTY
FROM SPJ
JOIN S ON SPJ.SNO = S.SNO
JOIN P ON SPJ.PNO = P.PNO
JOIN J ON SPJ.JNO = J.JNO;

SHOW FULL TABLES WHERE Table_type = 'VIEW';
SELECT * FROM supplier_supply_overview ORDER BY SNO, PNO, JNO;
```


Output Result 11-2 SPJ.supplier_supply_overview

Tx: Auto CSV

	SNO	SNAME	CITY	PNO	PNAME	JNO	JNAME	QTY
1	S1	精益	天津	P1	螺母	J1	三建	200
2	S1	精益	天津	P1	螺母	J3	弹簧厂	100
3	S1	精益	天津	P1	螺母	J4	造船厂	700
4	S1	精益	天津	P2	螺栓	J2	一汽	100
5	S2	盛锡	北京	P4	螺丝刀	J6	无线电厂	200
6	S3	东方红	北京	P1	螺母	J1	三建	200
7	S3	东方红	北京	P3	螺丝刀	J1	三建	200
8	S3	东方红	北京	P6	齿轮	J2	一汽	200
9	S4	丰泰盛	天津	P5	凸轮	J1	三建	100
10	S4	丰泰盛	天津	P6	齿轮	J3	弹簧厂	300
11	S4	丰泰盛	天津	P6	齿轮	J4	造船厂	200
12	S5	为民	上海	P2	螺栓	J4	造船厂	100
13	S5	为民	上海	P3	螺丝刀	J1	三建	200
14	S5	为民	上海	P6	齿轮	J4	造船厂	500

4.1.3 创建用户和角色

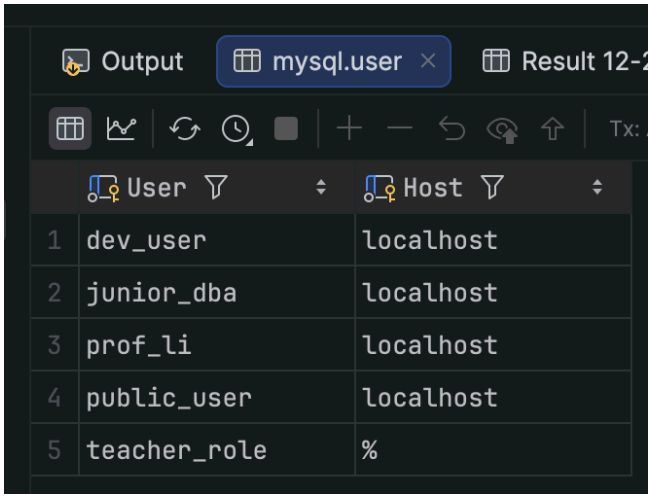
目的：掌握数据库用户和角色的建立方法，为后续授权与回收权限实验提供对象基础。

(1) 创建实验所需用户和角色并验证对象已建立

```
DROP USER IF EXISTS 'dev_user'@'localhost';
DROP USER IF EXISTS 'prof_li'@'localhost';
DROP USER IF EXISTS 'junior_dba'@'localhost';
DROP USER IF EXISTS 'public_user'@'localhost';
DROP ROLE IF EXISTS 'teacher_role';

CREATE USER 'dev_user'@'localhost' IDENTIFIED BY '123456';
CREATE USER 'prof_li'@'localhost' IDENTIFIED BY '123456';
CREATE USER 'junior_dba'@'localhost' IDENTIFIED BY '123456';
CREATE USER 'public_user'@'localhost' IDENTIFIED BY '123456';
CREATE ROLE 'teacher_role';

SELECT User, Host
FROM mysql.user
WHERE User IN ('dev_user', 'prof_li', 'junior_dba', 'public_user', '
    teacher_role')
ORDER BY User, Host;
```


Output mysql.user Result 12-2

Tx:

	User	Host
1	dev_user	localhost
2	junior_dba	localhost
3	prof_li	localhost
4	public_user	localhost
5	teacher_role	%

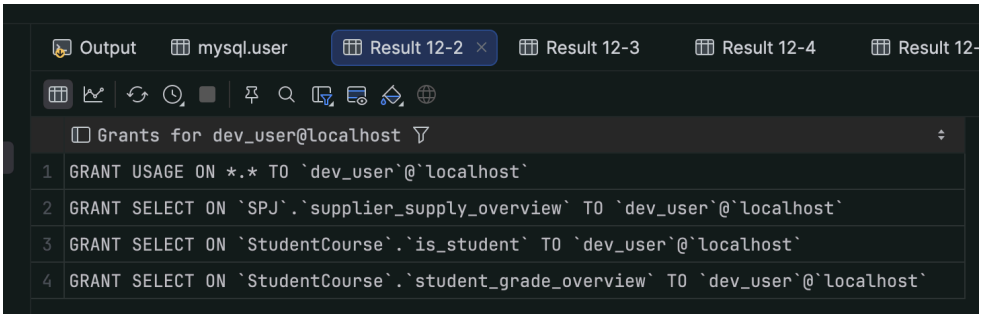
4.1.4 授权与权限验证

目的：掌握面向视图、角色和数据库对象的授权与回收方法，并验证不同账号的权限配置结果。

(1) 给普通查询用户授予视图访问权限

```
GRANT SELECT ON StudentCourse.is_student TO 'dev_user'@'localhost';
GRANT SELECT ON StudentCourse.student_grade_overview TO 'dev_user'@'
    localhost';
GRANT SELECT ON SPJ.supplier_supply_overview TO 'dev_user'@'localhost';

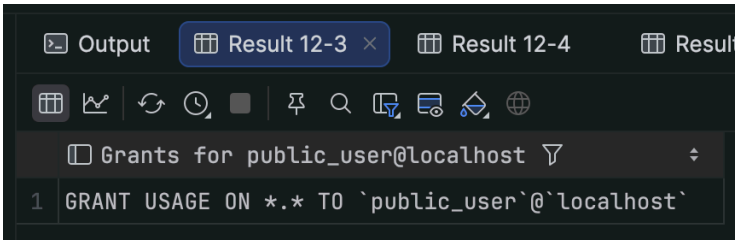
SHOW GRANTS FOR 'dev_user'@'localhost';
```



(2) 演示授权后回收权限

```
GRANT SELECT ON SPJ.P TO 'public_user'@'localhost';
REVOKE SELECT ON SPJ.P FROM 'public_user'@'localhost';

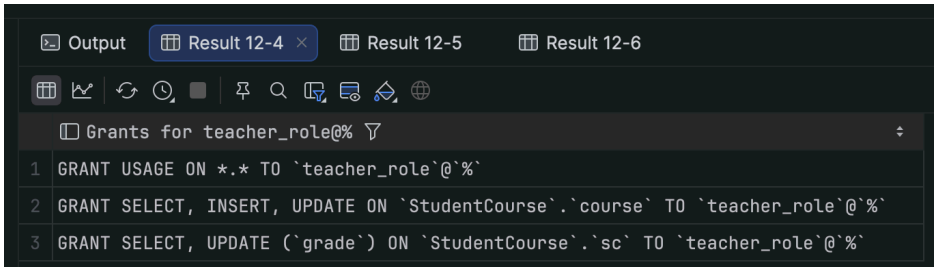
SHOW GRANTS FOR 'public_user'@'localhost';
```



(3) 给教师角色授予课程与成绩相关权限

```
GRANT SELECT, INSERT, UPDATE ON StudentCourse.course TO 'teacher_role';
GRANT SELECT, UPDATE(grade) ON StudentCourse.sc TO 'teacher_role';

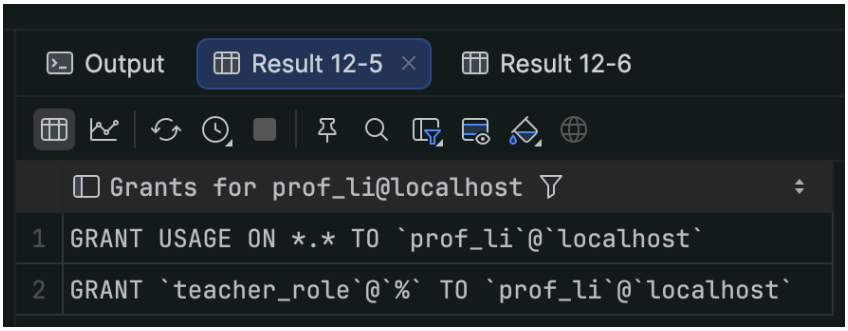
SHOW GRANTS FOR 'teacher_role';
```



(4) 把教师角色分配给教师用户并设置默认角色

```
GRANT 'teacher_role' TO 'prof_li'@'localhost';
SET DEFAULT ROLE 'teacher_role' TO 'prof_li'@'localhost';

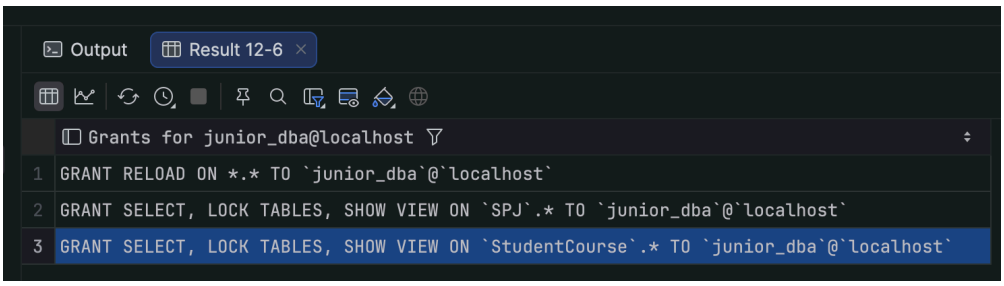
SHOW GRANTS FOR 'prof_li'@'localhost';
```



(5) 给受限管理员授予备份相关权限

```
GRANT SELECT, SHOW VIEW, LOCK TABLES ON StudentCourse.* TO 'junior_dba'@'
    localhost';
GRANT SELECT, SHOW VIEW, LOCK TABLES ON SPJ.* TO 'junior_dba'@'localhost';
GRANT RELOAD ON *.* TO 'junior_dba'@'localhost';

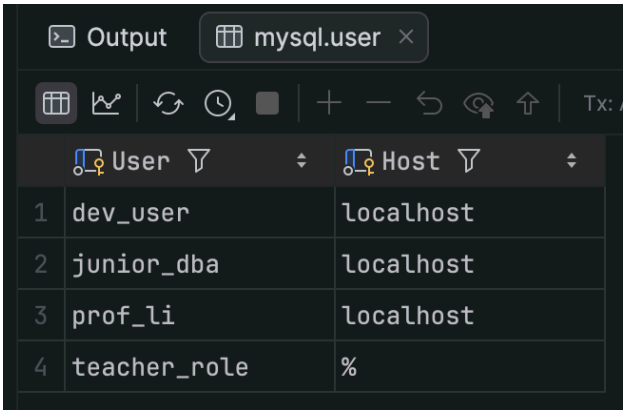
SHOW GRANTS FOR 'junior_dba'@'localhost';
```



(6) 删除临时用户并验证用户列表变化

```
DROP USER 'public_user'@'localhost';

SELECT User, Host
FROM mysql.user
WHERE User IN ('dev_user', 'prof_li', 'junior_dba', 'public_user', '
    teacher_role')
ORDER BY User, Host;
```



4.1.5 数据库备份

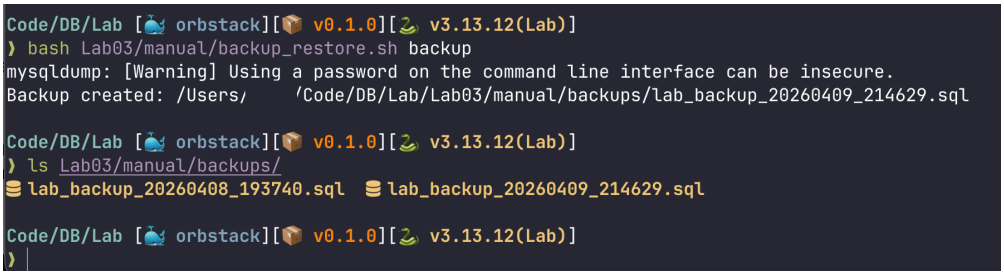
目的：掌握借助 Shell 脚本和 `mysqldump` 对 StudentCourse 与 SPJ 进行统一备份的方法。

(1) 备份脚本核心逻辑与执行命令

```
MODE="${1:-backup}"
BACKUP_DIR="${SCRIPT_DIR}/backups"

docker exec "${CONTAINER_NAME}" sh -c \
  "exec mysqldump -uroot -p${MYSQL_PASSWORD} --databases StudentCourse SPJ
     --routines --triggers --set-gtid-purged=OFF" \
  > "${OUTPUT_FILE}"

bash Lab03/manual/backup_restore.sh backup
```



4.1.6 恢复流程验证

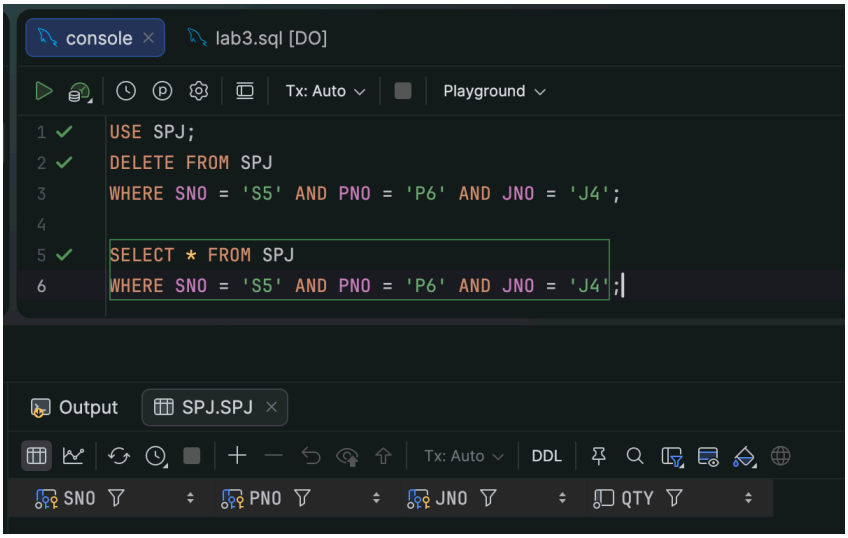
目的：掌握通过删除记录、执行恢复命令并再次查询的方式验证备份文件有效性的完整流程。

(1) 先删除一条供应记录并验证其已不存在

```
USE SPJ;

DELETE FROM SPJ
WHERE SNO = 'S5' AND PNO = 'P6' AND JNO = 'J4';

SELECT *
FROM SPJ
WHERE SNO = 'S5' AND PNO = 'P6' AND JNO = 'J4';
```



(2) 执行恢复命令重新导入备份文件

```
cat Lab03/manual/backups/lab_backup_20260409_214629.sql | \
docker exec -i db-lab-mysql8 sh -c "exec mysql -uroot -p123456"

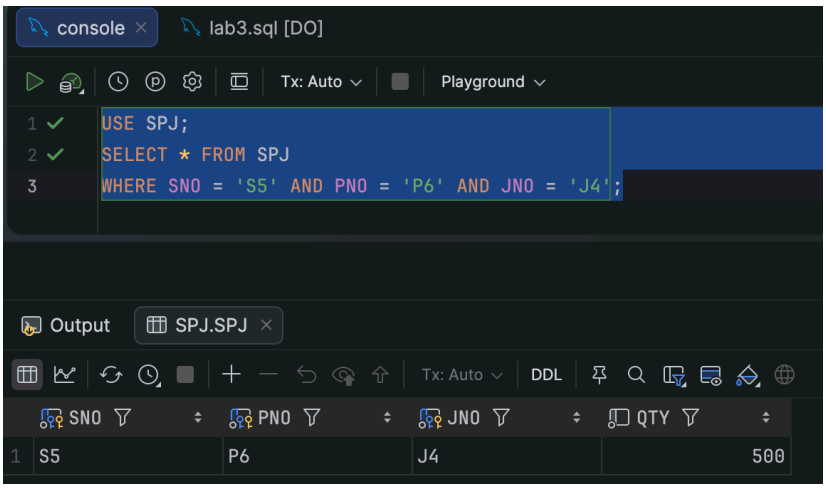
bash Lab03/manual/backup_restore.sh restore \
  Lab03/manual/backups/lab_backup_20260409_214629.sql
```

```
Code/DB/Lab [orbstack][v0.1.0][v3.13.12(Lab)]
) bash Lab03/manual/backup_restore.sh restore Lab03/manual/backups/lab_backup_20260409_214629.sq
l
mysql: [Warning] Using a password on the command line interface can be insecure.
Restore completed from: Lab03/manual/backups/lab_backup_20260409_214629.sql
```

(3) 恢复完成后再次查询原记录

```
USE SPJ;

SELECT *
FROM SPJ
WHERE SNO = 'S5' AND PNO = 'P6' AND JNO = 'J4';
```



4.2 实验总结与心得体会

本次实验围绕数据库安全控制、视图以及备份恢复完成了第三次上机内容。通过在 StudentCourse 和 SPJ 上建立视图，进一步理解了视图对基表数据的封装作用；通过创建用户、角色并实施授权与回收，进一步掌握了数据库安全性控制的基本方法；通过备份脚本与恢复验证，又进一步理解了数据库运维中数据保护与故障恢复的重要流程。

实验结果表明，数据库系统的管理不仅包括数据定义和数据查询，还包括权限边界控制、对象封装和异常恢复等环节。只有把视图、安全控制和备份恢复统一纳入实验过程，才能更完整地理解数据库系统在实际工程环境中的运行与维护方式。