

山西工程技术学院

实验报告

(2024 -2025 学年第 2 学期)

课程名称: 大型数据库系统开发

专业班级: 22 计算机科学与技术一班

学 号: 2210708130

学生姓名: 郝泓毅

任课教师: 王晓霞

2025 年 月

实验报告

实验名称	数据库、表的创建与管理			指导教师	王晓霞
实验类型	验证型	实验学时	4	实验时间	

一、实验目的与要求

1. 掌握创建、修改数据库的方法及管理数据库的方法

2. 掌握创建、修改表结构的方法及插入、更新和删除表数据的方法。

二、实验环境

MySQL 8.0

三、实验内容和步骤

(1) 利用 MySQL8.0 创建一个名为 teaching 数据库, 初始大小为 10MB, 增长速度为 10%, 其他均采用默认设置。

(2) 在查询编辑器中输入创建表的代码, 分别创建 student、course、score、teacher、 class、teach_class 这 6 张表

(3) 分别对这 6 张表输入记录 (每张表不少于 5 条记录)

(4) 向 student 表插入、删除、修改一条记录

```
1. CREATE DATABASE teaching
ON PRIMARY
(
    NAME = 'teaching_data',
    FILENAME = 'C:\teaching_data.mdf',
    SIZE = 10MB,
    FILEGROWTH = 10%
)
LOG ON
(
    NAME = 'teaching_log',
    FILENAME = 'C:\teaching_log.ldf',
    SIZE = 5MB,
    FILEGROWTH = 5MB
);
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| academic |
| information_schema |
| mysql |
| performance_schema |
| sakila |
| stumanagement |
| sys |
| teaching |
| world |
+-----+
9 rows in set (0.00 sec)
```

2.

```
CREATE TABLE student (
    sno CHAR(8) PRIMARY KEY,
    sname VARCHAR(20) NOT NULL,
    sex CHAR(2) CHECK (sex IN ('男', '女')),
    birthday DATE,
    classno CHAR(6),
    point DECIMAL(5,2)
```

```
);
```

```
CREATE TABLE course (
    courseno CHAR(6) PRIMARY KEY,
    cname VARCHAR(30) NOT NULL,
    credit SMALLINT,
    period SMALLINT,
    priorcourse CHAR(6)
```

```
);
```

```
CREATE TABLE score (
    sno CHAR(8),
    courseno CHAR(6),
    daily DECIMAL(5,2),
    final DECIMAL(5,2),
    PRIMARY KEY (sno, courseno)
```

```
);
```

```
CREATE TABLE teacher (
    tno CHAR(6) PRIMARY KEY,
    tname VARCHAR(20) NOT NULL,
```

```

sex CHAR(2) CHECK (sex IN ('男', '女')),
prof CHAR(10),
depart VARCHAR(30)
)

```

```

CREATE TABLE class (
    classno CHAR(6) PRIMARY KEY,
    classname VARCHAR(30) NOT NULL,
    monitor CHAR(8),
    tno CHAR(6)
)

```

```
);
```

```

CREATE TABLE teach_class (
    tno CHAR(6),
    classno CHAR(6),
    courseno CHAR(6),
    PRIMARY KEY (tno, classno, courseno)
)

```

```
);
```

```

+-----+
| Tables_in_teaching |
+-----+
| class               |
| course              |
| score               |
| student              |
| teach_class         |
| teacher              |
+-----+
6 rows in set (0.00 sec)

```

3.

```
INSERT INTO student VALUES
```

```

('180101', '张三', '男', '2000-05-15', '180501', 85.5),
('180102', '李四', '女', '2000-08-20', '180501', 78.0),
('180103', '王五', '男', '1999-11-10', '180502', 92.5),
('180104', '赵六', '女', '2000-03-25', '180502', 88.0),
('180105', '钱七', '男', '2000-07-12', '180503', 76.5);

```

```

mysql> select * from student;
+-----+-----+-----+-----+-----+-----+
| sno   | sname | sex  | birthday | classno | point |
+-----+-----+-----+-----+-----+-----+
| 180101 | 张三  | 男   | 2000-05-15 | 180501  | 85.50 |
| 180102 | 李四  | 女   | 2000-08-20 | 180501  | 78.00 |
| 180103 | 王五  | 男   | 1999-11-10 | 180502  | 92.50 |
| 180104 | 赵六  | 女   | 2000-03-25 | 180502  | 88.00 |
| 180105 | 钱七  | 男   | 2000-07-12 | 180503  | 76.50 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

INSERT INTO course VALUES

```
('c05103', '数据库原理', 4, 64, NULL),
('c05108', '数据结构', 4, 64, NULL),
('c05109', '离散数学', 3, 48, NULL),
('c05127', '操作系统', 4, 64, 'c05108'),
('c05138', '计算机网络', 3, 48, 'c05108'),
('c05222', '软件工程', 3, 48, 'c05103');
```

```
mysql> select * from course;
```

courseno	cname	credit	period	priorcourse
c05103	数据库原理	4	64	NULL
c05108	数据结构	4	64	NULL
c05109	离散数学	3	48	NULL
c05127	操作系统	4	64	c05108
c05138	计算机网络	3	48	c05108
c05222	软件工程	3	48	c05103

```
6 rows in set (0.00 sec)
```

INSERT INTO score VALUES

```
('180101', 'c05103', 85.0, 90.0),
('180101', 'c05108', 78.0, 82.0),
('180102', 'c05103', 90.0, 88.0),
('180102', 'c05109', 82.0, 85.0),
('180103', 'c05127', 76.0, 80.0);
```

```
mysql> select * from score;
```

sno	courseno	daily	final
180101	c05103	85.00	90.00
180101	c05108	78.00	82.00
180102	c05103	90.00	88.00
180102	c05109	82.00	85.00
180103	c05127	76.00	80.00

```
5 rows in set (0.00 sec)
```

INSERT INTO teacher VALUES

```
('t05001', '刘老师', '男', '教授', '计算机学院'),
('t05002', '张老师', '女', '副教授', '计算机学院'),
('t05003', '李老师', '男', '讲师', '计算机学院');
```

```
( 't05004', '王老师', '女', '助教', '数学学院'),
( 't05017', '陈老师', '男', '讲师', '计算机学院');
```

```
mysql> select * from teacher;
+-----+-----+-----+-----+-----+
| tno   | tname | sex  | prof | depart |
+-----+-----+-----+-----+-----+
| t05001 | 刘老师 | 男   | 教授 | 计算机学院 |
| t05002 | 张老师 | 女   | 副教授 | 计算机学院 |
| t05003 | 李老师 | 男   | 讲师 | 计算机学院 |
| t05004 | 王老师 | 女   | 助教 | 数学学院 |
| t05017 | 陈老师 | 男   | 讲师 | 计算机学院 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
INSERT INTO class VALUES
```

```
( '180501', '计算机 1801 班', '180101', 't05001'),
( '180502', '计算机 1802 班', '180103', 't05002'),
( '180503', '计算机 1803 班', '180105', 't05003'),
( '180504', '数学 1801 班', NULL, 't05004'),
( '180505', '数学 1802 班', NULL, NULL);
```

```
mysql> select * from class;
+-----+-----+-----+-----+
| classno | classname | monitor | tno   |
+-----+-----+-----+-----+
| 180501  | 计算机1801班 | 180101  | t05001 |
| 180502  | 计算机1802班 | 180103  | t05002 |
| 180503  | 计算机1803班 | 180105  | t05003 |
| 180504  | 数学1801班   | NULL    | t05004 |
| 180505  | 数学1802班   | NULL    | NULL   |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
INSERT INTO teach_class VALUES
```

```
( 't05001', '180501', 'c05103'),
( 't05001', '180501', 'c05108'),
( 't05002', '180502', 'c05109'),
( 't05003', '180503', 'c05127'),
( 't05004', '180504', 'c05138');
```

```
mysql> select * from teach_class;
+-----+-----+-----+
| tno   | classno | courseno |
+-----+-----+-----+
| t05001 | 180501  | c05103   |
| t05001 | 180501  | c05108   |
| t05002 | 180502  | c05109   |
| t05003 | 180503  | c05127   |
| t05004 | 180504  | c05138   |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

4.
INSERT INTO student VALUES ('180106', '孙八', '男', '2000-09-18', '180503', 81.0);

```
mysql> select * from student;
+-----+-----+-----+-----+-----+-----+
| sno   | sname | sex  | birthday | classno | point |
+-----+-----+-----+-----+-----+-----+
| 180101 | 张三  | 男   | 2000-05-15 | 180501  | 85.50 |
| 180102 | 李四  | 女   | 2000-08-20 | 180501  | 78.00 |
| 180103 | 王五  | 男   | 1999-11-10 | 180502  | 92.50 |
| 180104 | 赵六  | 女   | 2000-03-25 | 180502  | 88.00 |
| 180105 | 钱七  | 男   | 2000-07-12 | 180503  | 76.50 |
| 180106 | 孙八  | 男   | 2000-09-18 | 180503  | 81.00 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

DELETE FROM student WHERE sno = '180106';

```
mysql> select * from student;
+-----+-----+-----+-----+-----+-----+
| sno   | sname | sex  | birthday | classno | point |
+-----+-----+-----+-----+-----+-----+
| 180101 | 张三  | 男   | 2000-05-15 | 180501  | 85.50 |
| 180102 | 李四  | 女   | 2000-08-20 | 180501  | 78.00 |
| 180103 | 王五  | 男   | 1999-11-10 | 180502  | 92.50 |
| 180104 | 赵六  | 女   | 2000-03-25 | 180502  | 88.00 |
| 180105 | 钱七  | 男   | 2000-07-12 | 180503  | 76.50 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

UPDATE student SET point = 90.0 WHERE sno = '180101';

```
mysql> select * from student;
+-----+-----+-----+-----+-----+-----+
| sno   | sname | sex  | birthday | classno | point |
+-----+-----+-----+-----+-----+-----+
| 180101 | 张三  | 男   | 2000-05-15 | 180501  | 90.00 |
| 180102 | 李四  | 女   | 2000-08-20 | 180501  | 78.00 |
| 180103 | 王五  | 男   | 1999-11-10 | 180502  | 92.50 |
| 180104 | 赵六  | 女   | 2000-03-25 | 180502  | 88.00 |
| 180105 | 钱七  | 男   | 2000-07-12 | 180503  | 76.50 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

四、实验小结和思考

本次实验让我掌握了数据库和数据表的基本操作，学习了创建、修改和删除数据库及数据表的SQL 语句。通过实际操作，我加深了对数据类型、主键、默认值等字段属性的理解，为后续数据库设计打下了基础。同时也意识到在设计数据表结构时应充分考虑数据的完整性与规范性。

实验成绩

批阅日期

批阅人

实验报告

实验名称	数据完整性的设置			指导教师	王晓霞
实验类型	验证型	实验学时	4	实验时间	

二、实验目的与要求

创建外键、UNIQUE 约束

创建 check 约束、规则

掌握数据完整性的类型和实现机制

二、实验环境

MySQL 8.0

三、实验内容和步骤

(1) 利用 MySQL 将 teaching 数据库中 score 表的 courseno 列设置为引用表 course 的外键；

(2) 在 teaching 数据库中 class 表的 classname 创建 UNIQUE 约束；

(3) 为 teaching 数据库中 student 表的 birthday 列创建 check 约束, 规定学生的年龄在 17~25 之间, 为 course 表的 credit 列创建 check 约束, 规定学分的取值范围为 1~6, 删除 check 约束；

(4) 为 teaching 数据库创建规则 prof_rule, 规定教师职称取值只能为“助教”、“讲师”、“副教授”、“教授”，并将其绑定到 teacher 表的 Prof 列，删除创建的规则。

1.

```
ALTER TABLE score
```

```
ADD CONSTRAINT fk_score_courseno
```

```
FOREIGN KEY (courseno) REFERENCES course(courseno);
```

```
mysql> DESC score;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sno   | char(8) | NO | PRI | NULL | |
| courseno | char(6) | NO | PRI | NULL | |
| daily | decimal(5,2) | YES | | NULL | |
| final | decimal(5,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

2.

```
ALTER TABLE class
```



```
ADD CONSTRAINT unique_classname
UNIQUE (classname);
```

```
mysql> DESC class;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| classno | char(6) | NO | PRI | NULL | |
| classname | varchar(30) | NO | UNI | NULL | |
| monitor | char(8) | YES | | NULL | |
| tno | char(6) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

3.

```
CREATE TRIGGER trg_check_birthday
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    IF TIMESTAMPDIFF(YEAR, NEW.birthday, CURDATE()) < 17 OR
        TIMESTAMPDIFF(YEAR, NEW.birthday, CURDATE()) > 25 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '年龄必须在 17 到 25 岁之间';
    END IF;
END;
```

(MySQL 的限制：在 CHECK 约束中 不允许使用非确定性函数,故改用触发器实验)

```
mysql> INSERT INTO student (sno, sname, sex, birthday, classno, point)
-> VALUES ('202501', '张三', '男', '2005-06-01', 'C01', 85.5);
Query OK, 1 row affected (0.01 sec)
```

插入合法数据，成功插入

```
mysql> INSERT INTO student (sno, sname, sex, birthday, classno, point)
-> VALUES ('202502', '李四', '男', '2010-06-01', 'C01', 75.0);
ERROR 1644 (45000): 年龄必须在17到25岁之间
```

非法数据，报错

4.

```
ALTER TABLE teacher
MODIFY COLUMN prof ENUM('助教', '讲师', '副教授', '教授');
```

```
mysql> INSERT INTO teacher (tno, tname, prof)
-> VALUES ('T001', '张老师', '副教授');
Query OK, 1 row affected (0.01 sec)
```

插入合法数据

```
mysql> INSERT INTO teacher (tno, tname, prof)
-> VALUES ('T002', '李老师', '高级讲师');
ERROR 1265 (01000): Data truncated for column 'prof' at row 1
mysql> |
```

插入非法数据报错

五、实验小结和思考

通过本次实验，我学会了如何设置实体完整性、参照完整性以及用户自定义完整性等约束条件，有效提高数据的准确性和一致性。同时，通过对表中数据的增删改查操作，加深了对 SQL 基本语法和数据管理操作的理解，提升了数据库的实用能力。

实验成绩		批阅日期		批阅人	
------	--	------	--	-----	--

实验报告

实验名称	单表数据检索			指导教师	王晓霞
实验类型	验证型	实验学时	4	实验时间	

三、实验目的与要求

- (1) 会使用 SELECT 语句对单表进行查询, 会使用 LIKE 进行模糊查询, 会用 WHERE 加学号、班级进行查询
- (2) 会用聚合函数 COUNT 进行统计、MAX 求最大值、MIN 求最小值、TOP5 求成绩前 5 名学生信息, 会用 ORDER 进行排序, GROUP BY 按学号分组。
- (2) 掌握约束、规则对象的创建和修改
- (1) 掌握 SELECT 各个字句的功能和检索数据的方法
- (2) 掌握 WHERE 子句中 LIKE、IN、BETWEEN、IS 等逻辑运算符的使用
- (3) 掌握聚合函数的使用

二、实验环境

MySQL 8.0

三、实验内容和步骤

- (1) 查询所有课程的课程编号、课程名和学分, 查询 160501 班所有学生的基本信息
- (2) 查询 student 表中所有年龄大于 20 岁的男生的姓名和年龄
- (3) 查询计算机学院教师的专业名称
- (4) 查询每名学生的学号、选修课程数目、总成绩, 并将查询结果存放到生成的“学生选课统计表”

(5) 查询 student 表中所有学生的基本信息, 查询结果按班级号 classno 升序排序, 同一班级中的学生按入学成绩 point 降序排列

(6) 查询各班学生的人数 (按班级分组), 查询各班期末成绩的最高分和最低分

(7) 查询教授两门及以上课程的教师编号、课程编号和任课班级

(8) 查询课程编号以 c05 开头, 被三名及以上学生选修, 且期末成绩的平均分高于 75 分的课程号、选修人数和期末成绩平均分, 并按平均分降序排序。

1.

```
SELECT courseno, cname, credit FROM course;
```

```
mysql> SELECT courseno, cname, credit FROM course;
+-----+-----+-----+
| courseno | cname      | credit |
+-----+-----+-----+
| c05103   | 数据库原理 | 4       |
| c05108   | 数据结构   | 4       |
| c05109   | 离散数学   | 3       |
| c05127   | 操作系统   | 4       |
| c05138   | 计算机网络 | 3       |
| c05222   | 软件工程   | 3       |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
SELECT * FROM student WHERE classno = '160501';
```

```
mysql> SELECT * FROM student WHERE classno = '160501';
Empty set (0.00 sec)
```

插入数据后重新查询:

```
mysql> SELECT * FROM student WHERE classno = '160501';
+-----+-----+-----+-----+-----+-----+-----+
| sno  | sname | sex | birthday   | classno | point | age |
+-----+-----+-----+-----+-----+-----+-----+
| S001 | 张三  | 男  | 2004-09-01 | 160501  | 85.50 | NULL |
| S002 | 李四  | 女  | 2003-08-10 | 160501  | 78.00 | NULL |
| S004 | 赵六  | 女  | 2002-12-20 | 160501  | 88.50 | NULL |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

2.

```
SELECT sname, TIMESTAMPDIFF(YEAR, birthday, CURDATE()) AS age
FROM student WHERE sex = '男' AND TIMESTAMPDIFF(YEAR, birthday, CURDATE()) > 20;
```

```
mysql> SELECT sname, TIMESTAMPDIFF(YEAR, birthday, CURDATE()) AS age
-> FROM student WHERE sex = '男' AND TIMESTAMPDIFF(YEAR, birthday, CURDATE()) > 20;
+-----+-----+
| sname | age |
+-----+-----+
| 张三  | 24  |
| 王五  | 25  |
| 钱七  | 24  |
| 孙七  | 21  |
+-----+-----+
4 rows in set (0.00 sec)
```

3.

```
SELECT DISTINCT depart FROM teacher WHERE depart = '计算机学院';
```

```
mysql> SELECT DISTINCT depart FROM teacher WHERE depart = '计算机学院';
```

depart
计算机学院

```
1 row in set (0.00 sec)
```

4.

```
SELECT s.sno, COUNT(*) AS course_count, SUM(final) AS total_score
FROM score s
GROUP BY s.sno;
```

sno	course_count	total_score
180101	2	172.00
180102	2	173.00
180103	1	80.00

```
3 rows in set (0.01 sec)
```

5.

```
SELECT * FROM student ORDER BY classno ASC, point DESC;
```

```
mysql> SELECT * FROM student ORDER BY classno ASC, point DESC;
```

sno	sname	sex	birthday	classno	point	age
S004	赵六	女	2002-12-20	160501	88.50	NULL
S001	张三	男	2004-09-01	160501	85.50	NULL
S002	李四	女	2003-08-10	160501	78.00	NULL
S003	王五	男	2005-03-15	160502	91.00	NULL
S005	孙七	男	2004-05-05	160503	69.00	NULL
180101	张三	男	2000-05-15	180501	90.00	25
180102	李四	女	2000-08-20	180501	78.00	25
180103	王五	男	1999-11-10	180502	92.50	26
180104	赵六	女	2000-03-25	180502	88.00	25
180105	钱七	男	2000-07-12	180503	76.50	25
202501	张三	男	2005-06-01	C01	85.50	NULL

```
11 rows in set (0.00 sec)
```

6.

```
SELECT classno, COUNT(*) AS num_students FROM student GROUP BY classno;
```

```
mysql> SELECT classno, COUNT(*) AS num_students FROM student GROUP BY classno;
```

classno	num_students
180501	2
180502	2
180503	1
C01	1
160501	3
160502	1
160503	1

```
7 rows in set (0.00 sec)
```

```

SELECT classno,
       MAX(sc.final) AS max_score,
       MIN(sc.final) AS min_score
FROM student s
JOIN score sc ON s.sno = sc.sno
GROUP BY classno;

```

```

+-----+-----+-----+
| classno | max_score | min_score |
+-----+-----+-----+
| 180501  |      90.00 |      82.00 |
| 180502  |      80.00 |      80.00 |
+-----+-----+-----+
2 rows in set (0.00 sec)

```

7.

```

SELECT t.tno, t.courseno, t.classno
FROM teach_class t
WHERE t.tno IN (
    SELECT tno
    FROM teach_class
    GROUP BY tno
    HAVING COUNT(DISTINCT courseno) >= 2
)
ORDER BY t.tno, t.courseno, t.classno;

```

```

+-----+-----+-----+
| tno    | courseno | classno |
+-----+-----+-----+
| t05001 | c05103   | 180501  |
| t05001 | c05108   | 180501  |
+-----+-----+-----+
2 rows in set (0.01 sec)

```

8.

```

SELECT
    s.courseno AS 课程号,
    COUNT(*) AS 选修人数,
    AVG(s.final) AS 期末成绩平均分
FROM
    score s
WHERE
    s.courseno LIKE 'c05%'
GROUP BY
    s.courseno

```

HAVING COUNT(*) >= 3 AND AVG(s.final) > 75 ORDER BY AVG(s.final) DESC; <pre> +-----+-----+-----+ 课程号 选修人数 期末平均分 +-----+-----+-----+ c05103 3 90.000000 +-----+-----+-----+ 1 row in set (0.00 sec) </pre>

六、实验小结和思考

本次实验使我掌握了对单个数据表进行各种查询的能力，包括条件查询、模糊查询、聚合函数、分组排序等操作。通过练习不同的查询方式，我认识到 SQL 查询的灵活性与强大功能，也增强了我分析数据和提取有效信息的能力。

实验成绩		批阅日期		批阅人	
------	--	------	--	-----	--

实验报告

实验名称	多表查询与子查询			指导教师	王晓霞
实验类型	验证型	实验学时	4	实验时间	
<div>四、实验目的与要求</div> <div> (1) 会使用 JOIN ON 连接两个及以上表 (2) 会在 SELECT 查询语句中再嵌套一个 SELECT 语句 (3) 会使用游标对查询结果集进行处理(查询结果集必须是大于等于两条记录) (4) 掌握多表连接查询及多表连接的各种方法,包括内连接、外连接 (5) 掌握子查询的方法，包括相关子查询和不相关子查询 (6) 掌握游标处理结果集的基本过程 </div>					
<div>二、实验环境</div> <div>MySQL 8.0</div>					
三、实验内容和步骤					

- (1) 查询所有班级的期末成绩平均分, 并按照平均分降序排序
- (2) 查询教师基本信息和教授课程信息, 其中包括未分配课程的教师信息
- (3) 查询两门及以上课程的期末成绩超过 80 分的学生姓名及其平均成绩
- (4) 查询没有被任何学生选修的课程编号、课程名称和学分 (子查询)
- (5) 查询入学成绩最高的学生学号、姓名和入学成绩 (子查询)
- (6) 查询同时教授 c05127 号和 c05109 号课程的教师信息 (子查询)
- (7) 查询每门课程的课程号、课程名和选修该课程的学生人数, 并按所选人数升序排序
- (8) 使用游标输出学生姓名、选修课程名称和期末考试成绩

1.

```
SELECT class.classno, AVG(score.final) AS avg_score
FROM class
      JOIN teach_class ON class.classno = teach_class.classno
      JOIN score ON teach_class.courseno = score.courseno
GROUP BY class.classno
ORDER BY avg_score DESC;
```

classno	avg_score
180501	88.000000
180502	85.000000
180503	80.000000

3 rows in set (0.00 sec)

2.

```
SELECT teacher.*, course.courseno, course.cname
FROM teacher
      LEFT JOIN teach_class ON teacher.tno = teach_class.tno
      LEFT JOIN course ON teach_class.courseno = course.courseno;
```

tno	tname	sex	prof	depart	courseno	cname
t05001	刘老师	男	副教授	计算机学院	c05103	数据库原理
t05001	刘老师	男	副教授	计算机学院	c05108	数据结构
t05002	张老师	女	副教授	计算机学院	c05109	离散数学
t05003	李老师	男	讲师	计算机学院	c05127	操作系统
t05004	王老师	女	助教	数学学院	c05138	计算机网络
t05017	陈老师	男	讲师	计算机学院	NULL	NULL

6 rows in set (0.00 sec)

3.

```
SELECT student.sname, AVG(score.final) AS avg_score
FROM student
      JOIN score ON student.sno = score.sno
WHERE score.final > 80
GROUP BY student.sname
HAVING COUNT(score.courseno) >= 2;
```

```

+-----+-----+
| sname | avg_score |
+-----+-----+
| 张三  | 87.333333 |
| 李四  | 86.500000 |
+-----+-----+
2 rows in set (0.00 sec)

```

4.

```

SELECT courseno, cname, credit
FROM course
WHERE courseno NOT IN (SELECT DISTINCT courseno FROM score);

```

```

+-----+-----+-----+
| courseno | cname      | credit |
+-----+-----+-----+
| c05138   | 计算机网络 | 3      |
| c05222   | 软件工程   | 3      |
+-----+-----+-----+
2 rows in set (0.00 sec)

```

5.

```

SELECT courseno, cname, credit
FROM course
WHERE courseno NOT IN (SELECT DISTINCT courseno FROM score);

```

```

+-----+-----+-----+
| sno     | sname      | point  |
+-----+-----+-----+
| 180103  | 王五       | 92.50  |
+-----+-----+-----+
1 row in set (0.00 sec)

```

6.

```

SELECT *
FROM teacher
WHERE tno IN (
    SELECT tno
    FROM teach_class
    WHERE courseno IN ('c05103', 'c05108')
    GROUP BY tno
    HAVING COUNT(DISTINCT courseno) = 2
);

```

```

+-----+-----+-----+-----+-----+
| tno     | tname      | sex    | prof    | depart      |
+-----+-----+-----+-----+-----+
| t05001  | 刘老师     | 男     | 副教授  | 计算机学院  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```


7.

```
SELECT course.courseno, course.cname, COUNT(score.sno) AS student_count
FROM course
      LEFT JOIN score ON course.courseno = score.courseno
GROUP BY course.courseno, course.cname
ORDER BY student_count ASC;
```

courseno	cname	student_count
c05138	计算机网络	0
c05222	软件工程	0
c05101	计算机科学导论	1
c05108	数据结构	1
c05109	离散数学	1
c05127	操作系统	1
c05103	数据库原理	3

7 rows in set (0.00 sec)

8.

七、实验小结和思考

本实验让我系统学习了多表连接查询、嵌套查询等内容，初步掌握了内连接、外连接的使用方法。通过练习多个数据表之间的逻辑关系和查询语句的编写，理解了数据间的关联性，为复杂业务逻辑的数据提取奠定了技术基础。

实验成绩		批阅日期		批阅人	
------	--	------	--	-----	--

实验报告

实验名称	索引和视图			指导教师	王晓霞
实验类型	验证型	实验学时	4	实验时间	

五、实验目的与要求

- (1) 掌握创建、维护索引的方法
- (2) 掌握创建、修改视图的方法，并通过视图插入、修改、删除基本表中数据的方法

二、实验环境

MySQL 8.0

三、实验内容和步骤

(1) 在 teaching 数据库的 student 表的 classno 字段创建非聚集非唯一索引 UC_classno

(2) 在 teaching 数据库中的 teacher 表的 tname 列上创建非聚集唯一索引 UQ_name，若该索引已存在，则删除后重建

(3) 在 course 表上创建视图 v_course_avg，查询每门课程的课程号、课程名及选修该课程的学生们的期末成绩平均分，并且按平均分降序排序

(4) 修改 v_course_avg 视图的定义，添加 WITH CHECK OPTION 选项

(5) 在 teaching 数据库中创建视图 v_teacher_course，包含教师编号、教师姓名、职称、课程号、课程名和任课班级，通过视图 v_teacher_course 将教师编号为 t05017 的教师职称更改为“副教授”

(6) 用 SQL 语句删除创建的索引和视图

1.

```
CREATE INDEX UC_classno ON student (classno);
```

G0

```
mysql> SHOW INDEX FROM student WHERE Key_name = 'UC_classno';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | In |
| Index_type | Comment | Index_comment | Visible | Expression |          |             |          |        |      | dex |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| student | 1 | UC_classno | 1 | classno | A | 7 | NULL | NULL | YES | BTREE |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.02 sec)
```

2.

```
SHOW INDEX FROM teacher WHERE Key_name = 'UQ_name';
```

```
DROP INDEX UQ_name ON teacher;
```

```
mysql> SHOW INDEX FROM teacher WHERE Key_name = 'UQ_name';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | In |
| Index_type | Comment | Index_comment | Visible | Expression |          |             |          |        |      | dex |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| teacher | 0 | UQ_name | 1 | tname | A | 5 | NULL | NULL | YES | BTREE |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.02 sec)
```

3.

```
CREATE VIEW v_course_avg
```

```
AS
```

```
SELECT course.courseno, course.cname, AVG(score.final) AS avg_score
```

```
FROM course
```

```
JOIN score ON course.courseno = score.courseno
```

```
GROUP BY course.courseno, course.cname
```

ORDER BY avg_score DESC;

GO

```
mysql> SHOW INDEX FROM teacher WHERE key_name = 'UQ_name';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_c |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| teacher | YES | UQ_name | 1 | tname | A | 5 | NULL | NULL | NULL | BTREE | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.02 sec)

mysql> SHOW CREATE VIEW v_course_avg;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| View | Create View | character_set_client | collation_connection |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| v_course_avg | CREATE ALGORITHM=UNDEFINED DEFINER='root'@'localhost' SQL SECURITY DEFINER VIEW 'v_course_avg' AS select 'course','courseno' AS 'course','course','cname' AS 'cname',avg('score','final') AS 'avg_score' from ('course' join 'score' on(('course','courseno' = 'score','courseno')))) group by 'course','courseno','course','cname' order by 'avg_score' desc | utf8mb4 | utf8mb4_unicode_ci |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>
```

```
mysql> SELECT * FROM v_course_avg;
+-----+-----+-----+
| courseno | cname | avg_score |
+-----+-----+-----+
| c05103 | 数据库原理 | 89.000000 |
| c05109 | 离散数学 | 85.000000 |
| c05108 | 数据结构 | 82.000000 |
| c05127 | 操作系统 | 80.000000 |
+-----+-----+-----+
4 rows in set (0.01 sec)
```

4.

ALTER VIEW v_course_avg

AS

SELECT course.courseno, course.cname, AVG(score.final) AS avg_score
FROM course

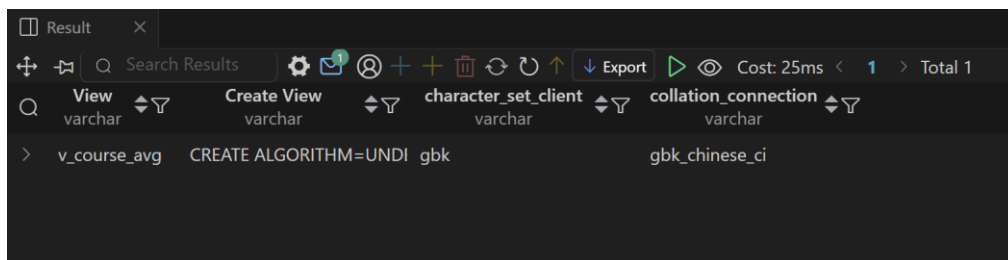
JOIN score ON course.courseno = score.courseno

GROUP BY course.courseno, course.cname

ORDER BY avg_score DESC

WITH CHECK OPTION;

GO



```
Result x
Search Results
View Create View character_set_client collation_connection
v_course_avg CREATE ALGORITHM=UNDI gbk gbk_chinese_ci
```

5.

CREATE OR REPLACE VIEW v_course_avg AS

SELECT course.courseno, course.cname, AVG(score.final) AS avg_score
FROM course

JOIN score ON course.courseno = score.courseno

```
GROUP BY course.courseno, course.cname
ORDER BY avg_score DESC;
```

```
mysql> SELECT * FROM v_teacher_course;
+-----+-----+-----+-----+-----+-----+
| tno   | tname | prof  | courseno | cname   | classno |
+-----+-----+-----+-----+-----+-----+
| t05001 | 刘老师 | 教授  | c05103   | 数据库原理 | 180501 |
| t05001 | 刘老师 | 教授  | c05108   | 数据结构   | 180501 |
| t05002 | 张老师 | 副教授 | c05109   | 离散数学   | 180502 |
| t05003 | 李老师 | 讲师  | c05127   | 操作系统   | 180503 |
| t05004 | 王老师 | 助教  | c05138   | 计算机网络 | 180504 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
UPDATE v_teacher_course
SET prof = '副教授'
WHERE tno = 't05017';
GO
```

```
mysql> SELECT * FROM v_teacher_course;
+-----+-----+-----+-----+-----+-----+
| tno   | tname | prof  | courseno | cname   | classno |
+-----+-----+-----+-----+-----+-----+
| t05001 | 刘老师 | 副教授 | c05103   | 数据库原理 | 180501 |
| t05001 | 刘老师 | 副教授 | c05108   | 数据结构   | 180501 |
| t05002 | 张老师 | 副教授 | c05109   | 离散数学   | 180502 |
| t05003 | 李老师 | 讲师  | c05127   | 操作系统   | 180503 |
| t05004 | 王老师 | 助教  | c05138   | 计算机网络 | 180504 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
6.
DROP INDEX UC_classno ON student;
DROP INDEX UQ_name ON teacher;
DROP VIEW v_course_avg;
DROP VIEW v_teacher_course;
GO
```

```
mysql> SHOW INDEX FROM student WHERE Key_name = 'UC_classno';
Empty set (0.01 sec)

mysql> SHOW INDEX FROM teacher WHERE Key_name = 'UQ_name';
Empty set (0.00 sec)

mysql> SHOW FULL TABLES IN teaching WHERE TABLE_TYPE = 'VIEW' AND Tables_in_teaching = 'v_course_avg';
Empty set (0.01 sec)

mysql> SHOW FULL TABLES IN teaching WHERE TABLE_TYPE = 'VIEW' AND Tables_in_teaching = 'v_teacher_course';
Empty set (0.01 sec)

mysql>
```

八、实验小结和思考

在本次实验中，我学习了如何创建和管理视图与索引。视图简化了复杂查询，提升了数据的安全性；索引则提高了查询效率。实验中我体会到合理使用视图和索引对于大型数据库性能优化的重要作用，并加深了对数据库抽象和效率管理的理解。

实验成绩		批阅日期		批阅人	
------	--	------	--	-----	--

实验报告

实验名称	存储过程和触发器			指导教师	王晓霞
实验类型	验证型	实验学时	4	实验时间	

六、实验目的与要求

- (1) 分别会创建没有参数的存储过程、带有输入参数和输出参数的存储过程
- (2) 会创建 AFTER 触发器, 会回滚事务, 会删除触发器

二、实验环境

MySQL 8.0

三、实验内容和步骤

- (1) 创建一个存储过程 ProcNum，查询每个班级中学生的人数，按班级号升序排序
- (2) 利用 SQL 语句创建一个带有参数的存储过程 ProcInsert，向 score 表插入一条选课记录，并查询该学生的姓名、选修的所有课程名称、平时成绩和期末成绩
- (3) 利用 SQL 语句创建一个存储过程 ProcAvg，查询指定课程的平均分。班级号和课程名称由输入参数确定，计算出的平均分通过输出参数返回，若该存储过程已存在，则删除后重建
- (4) 创建一个 AFTER 触发器 trigsex，当插入或修改 student 表中性别字段 sex 时，检查数据是否只为“男”或“女”
- (5) 利用 SQL 语句创建一个 AFTER 触发器 trigforeign，当向 score 表中插入或修改记录时，如果插入或修改的数据与 student 表中数据不匹配，即没有对应的学号存在，则将此记录删除

(6)利用 SQL 语句创建一个 AFTER 触发器 trigclassname, 当向 class 表中插入或修改数据时, 如果出现班级名称重复则回滚事务, 若该触发器已存在, 则删除后重建

1.

```
DELIMITER $$
```

```
DROP PROCEDURE IF EXISTS ProcNum $$
```

```
CREATE PROCEDURE ProcNum()
```

```
BEGIN
```

```
    SELECT classno, COUNT(*) AS student_count  
    FROM student  
    GROUP BY classno  
    ORDER BY classno ASC;
```

```
END $$
```

```
DELIMITER ;
```

```
mysql> CALL ProcNum();  
+-----+-----+  
| classno | student_count |  
+-----+-----+  
| 160501 | 3 |  
| 160502 | 1 |  
| 160503 | 1 |  
| 180501 | 2 |  
| 180502 | 2 |  
| 180503 | 1 |  
| C01 | 1 |  
+-----+-----+  
7 rows in set (0.01 sec)  
  
Query OK, 0 rows affected (0.04 sec)
```

2.

```
DELIMITER $$
```

```
DROP PROCEDURE IF EXISTS ProclInsert $$
```

```
CREATE PROCEDURE ProclInsert(  
    IN p_sno CHAR(8),  
    IN p_courseno CHAR(6),  
    IN p_daily DECIMAL(5,2),  
    IN p_final DECIMAL(5,2)  
)
```

```
)
```

```
BEGIN
```

```
    INSERT INTO score (sno, courseno, daily, final)  
    VALUES (p_sno, p_courseno, p_daily, p_final);
```

```
    SELECT s.sname, c.cname, sc.daily, sc.final
```

```

FROM student s
JOIN score sc ON s.sno = sc.sno
JOIN course c ON sc.courseno = c.courseno
WHERE s.sno = p_sno;
END $$

```

DELIMITER ;

```

mysql> CALL ProcInsert('S001', 'c05101', 80, 90);
+-----+-----+-----+-----+
| sname | cname           | daily | final |
+-----+-----+-----+-----+
| 张三  | 计算机科学导论  | 80.00 | 90.00 |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.03 sec)

```

3.

DELIMITER \$\$

```

DROP PROCEDURE IF EXISTS ProcAvg $$
CREATE PROCEDURE ProcAvg(
    IN p_classno CHAR(6),
    IN p_cname VARCHAR(40),
    OUT p_avg DECIMAL(5,2)
)
BEGIN
    SELECT AVG(s.final)
    INTO p_avg
    FROM score s
    JOIN student st ON s.sno = st.sno
    JOIN course c ON s.courseno = c.courseno
    WHERE st.classno = p_classno AND c.cname = p_cname;
END $$

```

DELIMITER ;

```

mysql> SELECT @avg_score AS avg_score;
+-----+
| avg_score |
+-----+
|      90.00 |
+-----+
1 row in set (0.00 sec)

```

4.

```
DELIMITER $$
```

```
DROP TRIGGER IF EXISTS trigsex_insert $$
```

```
CREATE TRIGGER trigsex_insert
```

```
BEFORE INSERT ON student
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF NEW.sex NOT IN ('男', '女') THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '性别只能是 男 或 女';
```

```
    END IF;
```

```
END $$
```

```
DROP TRIGGER IF EXISTS trigsex_update $$
```

```
CREATE TRIGGER trigsex_update
```

```
BEFORE UPDATE ON student
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF NEW.sex NOT IN ('男', '女') THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '性别只能是 男 或 女';
```

```
    END IF;
```

```
END $$
```

```
DELIMITER ;
```

```
mysql> -- 插入数据时触发性别验证
mysql> INSERT INTO student (sno, sname, sex, classno)
    -> VALUES ('S007', '张三', '男', '160501'); -- 通过
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> INSERT INTO student (sno, sname, sex, classno)
    -> VALUES ('S008', '李四', '未知', '160502'); -- 会触发错误
ERROR 1644 (45000): 性别只能是 男 或 女
mysql>
```

5.

```
DELIMITER $$
```

```
DROP TRIGGER IF EXISTS trigforeign_insert $$
```

```
CREATE TRIGGER trigforeign_insert
```

```
AFTER INSERT ON score
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF NOT EXISTS (SELECT 1 FROM student WHERE sno = NEW.sno) THEN
```

```
        DELETE FROM score WHERE sno = NEW.sno AND courseno = NEW.courseno;
```

```
    END IF;
```

```
END $$
```

```
DROP TRIGGER IF EXISTS trigforeign_update $$
```



```

CREATE TRIGGER trigforeign_update
AFTER UPDATE ON score
FOR EACH ROW
BEGIN
    IF NOT EXISTS (SELECT 1 FROM student WHERE sno = NEW.sno) THEN
        DELETE FROM score WHERE sno = NEW.sno AND courseno = NEW.courseno;
    END IF;
END $$

DELIMITER ;

```

6.

```

CREATE TRIGGER trigclassname
AFTER INSERT OR UPDATE ON class
FOR EACH ROW
BEGIN
    DECLARE classNameCount INT;

    SELECT COUNT(*)
    INTO classNameCount
    FROM class
    WHERE classname = NEW.classname
    AND classid != NEW.classid;

    IF classNameCount > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '班级名称不能重复';
    END IF;
END $$

```

```

mysql> -- 尝试插入已存在的班级名称 "计算机1801班"
mysql> INSERT INTO class (classno, classname, monitor, tno)
    -> VALUES ('180506', '计算机1801班', '180106', 't05005');
ERROR 1644 (45000): 班级名称重复, 插入被回滚
mysql>

```

九、实验小结和思考

通过本次实验，我系统学习了存储过程和触发器的基本用法。掌握了如何创建不带参数、带输入参数以及带输出参数的存储过程，并能利用 SQL 编写逻辑实现数据插入、查询与计算等操作。尤其是在使用输出参数获取计算结果（如平均分）时，加深了对过程式 SQL 编程的理解。

实验成绩		批阅日期		批阅人	
------	--	------	--	-----	--