



浙江大学  
ZHEJIANG UNIVERSITY

## 实验 2 SQL 数据定义和操作

2024-2025 春夏学期 数据库系统  
课程实验报告

姓名 王浩雄

学号 3230106032

年级 2023 级

专业 混合班（计算机科学与技术）

班级 混合 2303 班

2025 年 3 月 5 日

# 目录

<b>1</b>	<b>实验综述</b>	<b>3</b>
1.1	实验目的	3
1.2	实验内容	3
<b>2</b>	<b>实验环境</b>	<b>3</b>
<b>3</b>	<b>数据定义</b>	<b>4</b>
3.1	表的建立与删除	4
3.1.1	表的建立	4
3.1.2	表的删除	7
3.2	索引的建立与删除	7
3.2.1	索引的建立	7
3.2.2	索引的删除	7
<b>4</b>	<b>数据查询</b>	<b>8</b>
4.1	单表查询	8
4.2	多表查询	8
4.3	嵌套子查询	9
<b>5</b>	<b>数据更新</b>	<b>10</b>
5.1	插入表数据——INSERT 语句	10
5.2	删除表数据——DELETE 语句	11
5.3	更新表数据——UPDATE 语句	12
<b>6</b>	<b>视图操作</b>	<b>12</b>
6.1	视图的建立与删除	12
6.1.1	视图的建立	12
6.1.2	视图的删除	14
6.2	通过视图进行数据查询	14
6.3	通过视图进行数据更新	14
<b>7</b>	<b>实验总结与感想</b>	<b>15</b>

# 实验 2 SQL 数据定义和操作

## 1 实验综述

### 1.1 实验目的

1. 掌握关系数据库语言 SQL 的使用。
2. 面向某个应用场景，定义数据模式和操作数据。

### 1.2 实验内容

1. 以某个应用场景（如 Banking）为例，建立数据库。
2. 数据定义：表的建立、删除；索引的建立、删除；视图的建立、删除。
3. 数据查询：单表查询、多表查询、嵌套子查询等。
4. 数据更新：用 INSERT/DELETE/UPDATE 语句插入/删除/更新表数据。
5. 视图操作：通过视图进行数据查询和数据更新。
6. 完成实验报告。

## 2 实验环境

- 操作系统：  
Windows 11 Pro 24H2（64 位操作系统，基于 x64 的处理器）
- DBMS 版本：  
SQL Server Developer（64-bit）v16.0.1135.2  
SQL Server Management Studio v20.2.30.0

## 3 数据定义

### 3.1 表的建立与删除

#### 3.1.1 表的建立

使用下述 SQL 代码，建立大学（University）数据库所涉及的多种表。

##### 1. 部门表——departments

```
1 CREATE TABLE departments (  
2 department_id INT IDENTITY(1, 1) NOT NULL,  
3 department_name NVARCHAR(50) NOT NULL,  
4 PRIMARY KEY (department_id),  
5 );
```

##### 2. 学生表——students

```
1 CREATE TABLE students (  
2 student_id INT IDENTITY(1, 1) NOT NULL,  
3 student_name NVARCHAR(50) NOT NULL,  
4 gender NVARCHAR(10) CHECK (gender IN ('Male', 'Female', 'Other'))  
5 ),  
6 birth_date DATE,  
7 phone_number NVARCHAR(15),  
8 department_id INT,  
9 PRIMARY KEY (student_id),  
10 FOREIGN KEY (department_id) REFERENCES departments(department_id)  
11 ) ON DELETE SET NULL  
12 );
```

##### 3. 教师表——teachers

```
1 CREATE TABLE teachers (  
2 teacher_id INT IDENTITY(1, 1) NOT NULL,  
3 teacher_name NVARCHAR(50) NOT NULL,  
4 gender NVARCHAR(10) CHECK (gender IN ('Male', 'Female', 'Other'))  
5 ),  
6 birth_date DATE,  
7 phone_number NVARCHAR(15),  
8 department_id INT,  
9 PRIMARY KEY (teacher_id),
```

```
9 FOREIGN KEY (department_id) REFERENCES departments(department_id
    ) ON DELETE SET NULL
10 );
```

## 4. 开课信息表——courses

```
1 CREATE TABLE courses (
2   course_id INT IDENTITY(1, 1),
3   course_name NVARCHAR(100) NOT NULL,
4   credits FLOAT(1) NOT NULL,
5   department_id INT,
6   teacher_id INT,
7   PRIMARY KEY (course_id),
8   FOREIGN KEY (department_id) REFERENCES departments(department_id
    ) ON DELETE SET NULL,
9   FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id) ON
    DELETE SET NULL
10 );
```

## 5. 学生选课表——student\_courses

```
1 CREATE TABLE student_courses (
2   student_id INT NOT NULL,
3   course_id INT NOT NULL,
4   FOREIGN KEY (student_id) REFERENCES students(student_id) ON
    DELETE CASCADE,
5   FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE
    CASCADE
6 );
```

## 6. 学生成绩表——grades

```
1 CREATE TABLE grades (
2   grade_id INT IDENTITY(1, 1),
3   student_id INT NOT NULL,
4   course_id INT NOT NULL,
5   grade INT CHECK (grade BETWEEN 0 AND 100),
6   PRIMARY KEY (grade_id),
7   FOREIGN KEY (student_id) REFERENCES students(student_id) ON
    DELETE CASCADE,
8   FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE
    CASCADE
```

9 );

### 7. 导学关系表——student\_advisor

```

1 CREATE TABLE student_advisor (
2 relationship_id INT IDENTITY(1, 1),
3 student_id INT NOT NULL,
4 advisor_id INT NOT NULL,
5 PRIMARY KEY (relationship_id),
6 FOREIGN KEY (student_id) REFERENCES students(student_id) ON
  DELETE CASCADE,
7 FOREIGN KEY (advisor_id) REFERENCES teachers(teacher_id) ON
  DELETE CASCADE
8 );
    
```

建立的数据库关系如下图所示：

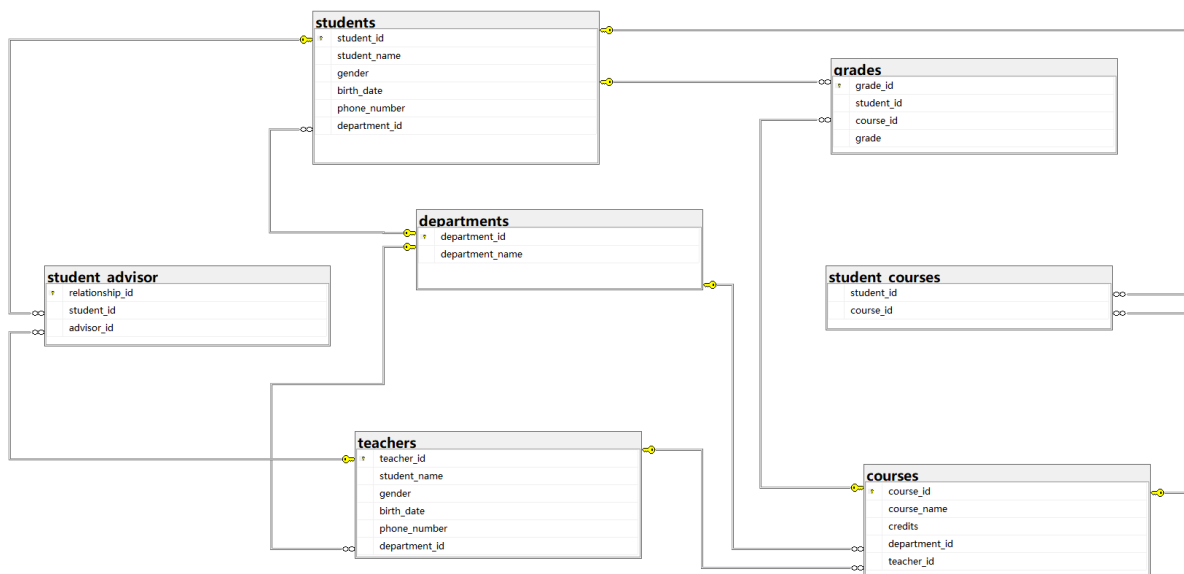


图 1: 数据库关系图

在上述表的建立语句中，我使用了如下特性，以确保数据库的信息完整与维护便利：

1. 指定主键的自增规则，避免人工设置主键可能带来的重复性、繁琐性问题。

```
1 ${PRIMARY KEY NAME} INT IDENTITY(1, 1),
```

2. 为外键设置级联操作，允许在被引用对象被删除时，自动为引用对象进行删除/置空相关项的操作。

```
1 FOREIGN KEY (student_id) REFERENCES students(student_id) ON
  DELETE CASCADE,
2 FOREIGN KEY (student_id) REFERENCES students(student_id) ON
  DELETE NULL
```

3. 规则检查，限制 gender 字段、grade 的填写内容。

```
1 gender NVARCHAR(10) CHECK (gender IN ('Male', 'Female', 'Other')
  ),
2 grade INT CHECK (grade BETWEEN 0 AND 100)
```

### 3.1.2 表的删除

使用下述 SQL 代码，删除 student\_advisor 表。

```
1 DROP TABLE student_advisor
```

## 3.2 索引的建立与删除

### 3.2.1 索引的建立

对于大学数据库，学生和教师的姓名和工号是常见的搜索字段。由于工号作为主键，已由数据库自动建立索引，因此使用下述 SQL 代码，为学生和教师表的姓名字段建立索引。

```
1 CREATE INDEX idx_students_name ON students(student_name);
2 CREATE INDEX idx_teachers_name ON teachers(teacher_name);
```

### 3.2.2 索引的删除

使用下述 SQL 代码，删除刚刚建立的索引。与课堂所讲不同的是，在 SQL Server 中，删除索引时必须指定索引所在的表名。

```
1 DROP INDEX idx_students_name ON students;
2 DROP INDEX idx_teachers_name ON teachers;
```

## 4 数据查询

### 4.1 单表查询

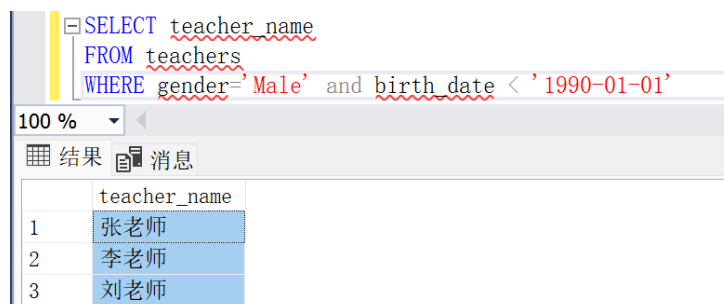
#### 操作 1

给定表 teacher，查询 gender=Male、birth\_date 早于 1990-01-01 的所有教师姓名。

SQL 代码：

```
1 SELECT teacher_name
2 FROM teachers
3 WHERE gender = 'Male' and birth_date < '1990-01-01'
```

查询结果：



The screenshot shows a SQL query editor with the following query:

```
SELECT teacher_name
FROM teachers
WHERE gender='Male' and birth_date < '1990-01-01'
```

Below the query, there is a section labeled "结果" (Results) showing the output of the query. The results are displayed in a table with the following data:

	teacher_name
1	张老师
2	李老师
3	刘老师

### 4.2 多表查询

#### 操作 2

查询学校开设的所有课程，并统计每门课程的选课人数。

SQL 代码：

```
1 SELECT
2     courses.course_id,
3     courses.course_name,
4     COUNT(student_courses.student_id) AS student_count
5 FROM courses, student_courses
6 WHERE courses.course_id = student_courses.course_id
7 GROUP BY courses.course_id, courses.course_name;
```

查询结果：



```

SELECT
    courses.course_id,
    courses.course_name,
    COUNT(student_courses.student_id) AS student_count
FROM courses, student_courses
WHERE courses.course_id = student_courses.course_id
GROUP BY courses.course_id, courses.course_name;

```

100 %

结果 消息

	course_id	course_name	student_count
1	1	离散数学	3
2	2	游泳（初级班）	2
3	3	微积分（甲）I	1
4	4	线性代数（甲）	2
5	5	计算机组成	2
6	6	数据库系统	2

### 4.3 嵌套子查询

#### 操作 3

查询学校开设的所有课程，并统计每门课程的选课人数。

SQL 代码：

```

1 SELECT
2     c.course_id,
3     c.course_name,
4     (
5         SELECT COUNT(*)
6         FROM student_courses sc
7         WHERE sc.course_id = c.course_id) AS student_count
8 FROM courses c

```

查询结果：

```

SELECT
    c.course_id,
    c.course_name,
    (
        SELECT COUNT(*)
        FROM student_courses sc
        WHERE sc.course_id = c.course_id) AS student_count
FROM courses c

```

100 %

结果 消息

	course_id	course_name	student_count
1	1	离散数学	3
2	2	游泳（初级班）	2
3	3	微积分（甲）I	1
4	4	线性代数（甲）	2
5	5	计算机组成	2
6	6	数据库系统	2

## 5 数据更新

### 5.1 插入表数据——INSERT 语句

#### 操作 4

新增一条学生数据。

SQL 代码:

```
1 INSERT INTO students(student_name,gender,department_id)
2 VALUES ('祝子为','Female',1)
```

运行结果 (students 表):

	student_id	student_name	gender	birth_date	phone_number	department_id
1	1	王浩雄	Male	2004-11-02	15703396384	1
2	2	周子为	Male	2005-09-09	NULL	1
3	3	黄逸轩	Female	2003-02-04	14853953856	2
4	4	范哲玮	Male	2006-07-27	18348926655	1

(运行前)

	student_id	student_name	gender	birth_date	phone_number	department_id
1	1	王浩雄	Male	2004-11-02	15703396384	1
2	2	周子为	Male	2005-09-09	NULL	1
3	3	黄逸轩	Female	2003-02-04	14853953856	2
4	4	范哲玮	Male	2006-07-27	18348926655	1
5	5	祝子为	Female	NULL	NULL	1

(运行后)

#### 操作 5

根据学生的姓名和课程的名称 (假设唯一) 新增一条学生选课数据。

SQL 代码:

```
1 INSERT INTO student_courses (student_id, course_id)
2 VALUES (
3 (SELECT TOP 1 student_id FROM students WHERE student_name = '周子为')
4 ,
5 (SELECT TOP 1 course_id FROM courses WHERE course_name = '游泳 (初级
   班)')
6 );
```

运行结果 (student\_courses 表):

	student_id	course_id		student_id	course_id
1	1	1	1	1	1
2	1	2	2	1	2
3	1	4	3	1	4
4	2	1	4	2	1
5	2	3	5	2	3
6	2	6	6	2	6
7	3	1	7	3	1
8	3	5	8	3	5
9	4	2	9	4	2
10	4	4	10	4	4
11	4	5	11	4	5
12	1	6	12	1	6
			13	2	2

(运行前)

(运行后)

## 5.2 删除表数据——DELETE 语句

### 操作 6

删除名为“周子为”的学生在 student\_courses 表中的所有选课记录。

SQL 代码:

```

1 DELETE FROM student_courses
2 WHERE student_id = (SELECT student_id FROM students WHERE
   student_name = '周子为');

```

运行结果 (student\_course 表):

	student_id	course_id		student_id	course_id
1	1	1	1	1	1
2	1	2	2	1	2
3	1	4	3	1	4
4	2	1	4	3	1
5	2	3	5	3	5
6	2	6	6	4	2
7	3	1	7	4	4
8	3	5	8	4	5
9	4	2	9	1	6
10	4	4			
11	4	5			
12	1	6			
13	2	2			

(运行前)

(运行后)

### 5.3 更新表数据——UPDATE 语句

#### 操作 7

将姓“黄”的学生的每科成绩乘以 0.8。

SQL 代码:

```
1 UPDATE grades
2 SET grade = grade * 0.8
3 WHERE student_id IN (
4     SELECT student_id FROM students WHERE student_name LIKE '黄%'
5 );
```

运行结果 (grade 表):

	grade_id	student_id	course_id	grade		grade_id	student_id	course_id	grade
1	1	1	1	89	1	1	1	1	89
2	2	1	2	99	2	2	1	2	99
3	3	1	4	92	3	3	1	4	92
4	4	2	1	77	4	4	2	1	77
5	5	2	3	88	5	5	2	3	88
6	6	2	6	91	6	6	2	6	91
7	7	3	1	96	7	7	3	1	76
8	8	3	5	89	8	8	3	5	71
9	9	4	2	59	9	9	4	2	59
10	10	4	4	68	10	10	4	4	68
11	11	4	5	100	11	11	4	5	100
12	12	1	6	91	12	12	1	6	91

(运行前)

(运行后)

## 6 视图操作

### 6.1 视图的建立与删除

#### 6.1.1 视图的建立

#### 操作 8

建立一个名为 StudentPhoneView 的视图, 用于在同一张表中查看学生的学号、姓名和手机号码。

SQL 代码:

```
1 CREATE VIEW StudentPhoneView AS
2 SELECT student_id, student_name, phone_number
3 FROM students;
```

## 操作 9

建立一个名为 Student\_Avg\_Grade\_Rank 的视图，用于在同一张表中输出学生姓名、学生所在学院名称、学生平均成绩和学生在学院内部的排名。

SQL 代码：

```
1 CREATE VIEW Student_Avg_Grade_Rank AS
2 WITH StudentAvg AS (
3 -- 计算每个学生的平均成绩
4 SELECT
5 student_id,
6 AVG(grade) AS avg_grade
7 FROM grades
8 GROUP BY student_id
9 ),
10 DepartmentAvg AS (
11 -- 计算每个学生的院系内平均成绩排名
12 SELECT
13 student_id,
14 (SELECT department_name FROM departments
15 WHERE department_id = (SELECT department_id FROM students WHERE
16     students.student_id = StudentAvg.student_id)
17 ) AS department_name,
18 avg_grade,
19 (SELECT COUNT(*)
20 FROM StudentAvg sa_inner
21 WHERE sa_inner.student_id IN (
22 SELECT student_id FROM students
23 WHERE department_id = (SELECT department_id FROM students WHERE
24     students.student_id = StudentAvg.student_id)
25 )
26 AND sa_inner.avg_grade > StudentAvg.avg_grade) + 1 AS department_rank
27 FROM StudentAvg
28 )
29 SELECT
30 (SELECT student_name FROM students WHERE students.student_id =
31     DepartmentAvg.student_id) AS student_name,
32 department_name, avg_grade, department_rank
33 FROM DepartmentAvg;
```

### 6.1.2 视图的删除

使用下述 SQL 代码，删除 StudentPhoneView 视图。

```
1 DROP VIEW StudentPhoneView
```

## 6.2 通过视图进行数据查询

### 操作 10

通过 Student\_Avg\_Grade\_Rank 视图，在同一张表中输出学生姓名、学生所在学院名称、学生平均成绩和学生在学院内部的排名。

SQL 代码：

```
1 SELECT * FROM Student_Avg_Grade_Rank
```

运行结果：

	student_name	department_name	avg_grade	department_rank
1	王浩雄	计算机科学与技术学院	92	1
2	周子为	计算机科学与技术学院	85	2
3	黄逸轩	数学科学学院	73	1
4	范哲玮	计算机科学与技术学院	75	3

## 6.3 通过视图进行数据更新

### 操作 11

通过 StudentPhoneView 视图，更新学生的电话号码。

SQL 代码：

```
1 UPDATE StudentPhoneView
2 SET phone_number = '13812345678'
3 WHERE student_name = '王浩雄';
```

运行结果（students 表）：

	student_id	student_name	gender	birth_date	phone_number	department_id
1	1	王浩雄	Male	2004-11-02	15703396384	1
2	2	周子为	Male	2005-09-09	NULL	1
3	3	黄逸轩	Female	2003-02-04	14853953856	2
4	4	范哲玮	Male	2006-07-27	18348926655	1
5	5	祝子为	Female	NULL	NULL	1

（运行前）

	student_id	student_name	gender	birth_date	phone_number	department_id
1	1	王浩雄	Male	2004-11-02	13812345678	1
2	2	周子为	Male	2005-09-09	NULL	1
3	3	黄逸轩	Female	2003-02-04	14853953856	2
4	4	范哲玮	Male	2006-07-27	18348926655	1
5	5	祝子为	Female	NULL	NULL	1

(运行后)

## 7 实验总结与感想

在本次实验中，我初步尝试使用 SQL 语句进行数据的定义与操作。通过实践操作，我进一步领略到 SQL 语句的统一格式与强大功能。同时，在实验过程中，我尝试使用了一些课堂上没有提到的 SQL 特性，并进一步体会到 SQL 不同发行版之间存在的区别。