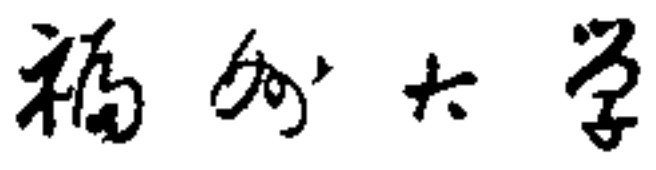
******

**《数据库应用实践》课程报告**

**学号： 102301116**

****

**姓名： 俞烨**

****

**年级： 2023级**

****

**学院： 计算机与大数据学院**

****

**专业： 计算机科学与技术**

****

**实验时间：2025－2026 学年第一学期**

目录

[实验一：数据库管理系统及其应用开发环境的创建使用 3](#_Toc217120836)

[1、数据库及表的情况简介。 4](#_Toc217120837)

[1.1 应用场景 4](#_Toc217120838)

[1.2 数据库表结构设计 4](#_Toc217120839)

[2、SQL语句实现结果 7](#_Toc217120840)

[2.1创建学生选课系统数据库 7](#_Toc217120841)

[2.2 创建基本表 7](#_Toc217120842)

[2.3表结构修改操作 9](#_Toc217120843)

[2.4索引创建与删除 10](#_Toc217120844)

[2.5数据操作（DML） 10](#_Toc217120845)

[2.6数据库查询操作 12](#_Toc217120846)

[2.7视图创建与使用 14](#_Toc217120847)

[实验总结 16](#_Toc217120848)

[实验二：数据库管理系统的维护与管理 17](#_Toc217120849)

[1、 数据库安全性实现流程 17](#_Toc217120850)

[1.1 数据库账户的添加与删除 17](#_Toc217120851)

[1.2对账户授予权限与收回权限 18](#_Toc217120852)

[2、触发器与存储过程的使用实现流程 19](#_Toc217120853)

[2.1创建存储过程并执行 19](#_Toc217120854)

[2.2 创建触发器并测试效果 20](#_Toc217120855)

[3、数据库备份与恢复实现流程 23](#_Toc217120856)

[3.1对所创建的数据库进行备份 23](#_Toc217120857)

[3.2利用备份进行数据库恢复 24](#_Toc217120858)

[实验总结： 25](#_Toc217120859)

[实验三：数据库应用系统的开发 27](#_Toc217120860)

[一、系统选题与工作情况说明 27](#_Toc217120861)

[二、概念数据模型（E-R 图） 28](#_Toc217120862)

[三、系统中每张表的说明 29](#_Toc217120863)

[四、系统运行环境配置与关键语句 33](#_Toc217120864)

[五、系统主要功能界面 35](#_Toc217120865)

[七、编程工作总结 41](#_Toc217120866)

[附：Excel 示例与导出 43](#_Toc217120867)

学号：102301116 姓名：俞烨 专业：计算机科学与技术 班级：计算机类01班 《数据库应用实践》

# 实验一：数据库管理系统及其应用开发环境的创建使用

实验环境：openEuler+openGauss数据库

实验内容：

## 1、数据库及表的情况简介。

1.1 应用场景

本实验设计并实现了一个“学生选课管理系统”数据库。该系统旨在模拟大学中的学生选课流程，主要业务需求包括：

管理学生基本信息（学号、姓名、性别、出生日期等）和所在院系。

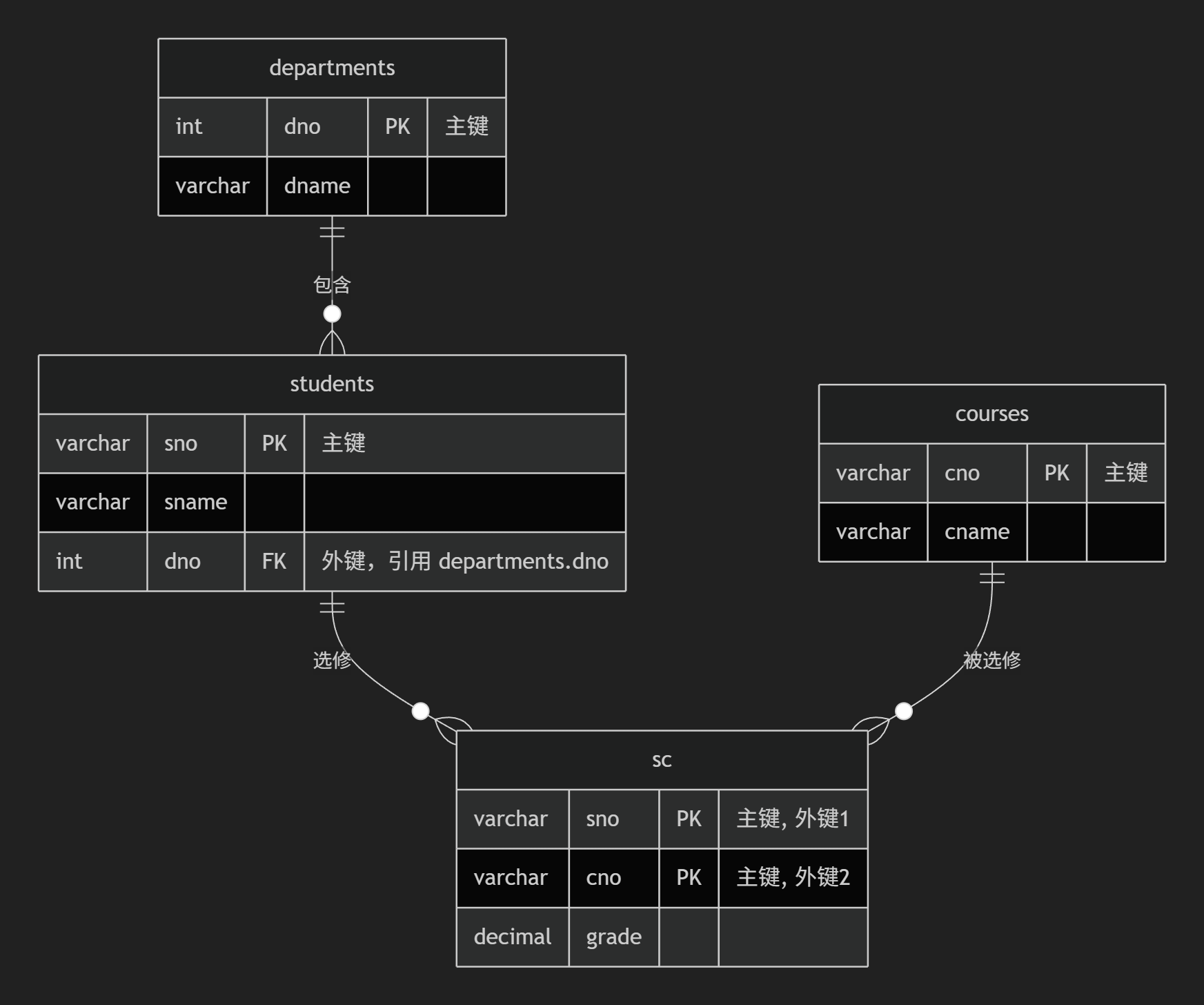
管理课程信息（课程号、课程名、学分、授课教师等）。

记录学生的选课情况以及对应的成绩。

支持按多种条件（如按院系、按成绩、按课程）进行灵活的查询统计。

1.2 数据库表结构设计

根据上述场景，我们在名为 db\_yuye的数据库中创建了四个核心表，它们之间的关联关系如下图所示：



各表详细说明：

院系表 (departments)

功能：存储所有院系的信息。

结构：

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **约束** | **说明** |
| dno | CHAR(4) | 主键 | 院系编号 |
| dname | VARCHAR(40) | 非空，唯一 | 院系名称 |
| dean | VARCHAR(20) |  | 系主任姓名 |
| establish\_date | DATE | 默认值为当前日期 | 成立日期 |

学生表 (students)

功能：存储所有学生的基本信息。

结构：

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **约束** | **说明** |
| sno | CHAR(10) | 主键 | 学号 |
| sname | VARCHAR(20) | 非空 | 学生姓名 |
| sex | CHAR(1) | 非空，Check约束 (性别 IN ('男', '女')) | 性别 |
| birthdate | DATE | 非空 | 出生日期 |
| height | DECIMAL(3,2) | Check约束 (身高 BETWEEN 1.00 AND 2.50) | 身高 |
| dorm | VARCHAR(20) | 默认值 '未分配' | 宿舍 |
| dno | CHAR(4) | 外键，引用 departments(dno) | 所属院系 |

课程表 (courses)

功能：存储所有课程的信息。

结构：

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **约束** | **说明** |
| cno | VARCHAR(10) | 主键 | 课程编号 |
| cname | VARCHAR(40) | 非空，唯一 | 课程名称 |
| period | SMALLINT | Check约束 (学时 BETWEEN 16 AND 128) | 学时 |
| credit | TINYINT | 非空 | 学分 |
| teacher | VARCHAR(20) | 非空 | 授课教师 |

选课表 (sc)

功能：记录学生选课及成绩信息，是连接学生和课程的纽带。

结构：

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **约束** | **说明** |
| sno | CHAR(10) | 联合主键，外键引用 students(sno) | 学号 |
| cno | VARCHAR(10) | 联合主键，外键引用 courses(cno) | 课程号 |
| grade | DECIMAL(4,1) | Check约束 (成绩 IS NULL OR 成绩 BETWEEN 0 AND 100) | 成绩 |
| selection\_date | TIMESTAMP | 默认值为当前时间戳 | 选课时间 |

## 2、SQL语句实现结果

2.1创建学生选课系统数据库

CREATE DATABASE db\_yuye OWNER yuye;



2.2 创建基本表

创建院系表

CREATE TABLE departments (

dno CHAR(4) PRIMARY KEY CHECK (dno ~ '^[A-Z]{2}\d{2}$'), -- 院系编号格式：2字母+2数字

dname VARCHAR(40) NOT NULL UNIQUE,

dean VARCHAR(20),

establish\_date DATE DEFAULT CURRENT\_DATE

);



创建学生表

CREATE TABLE students (

sno CHAR(10) PRIMARY KEY CHECK (sno ~ '^\d{10}$'), -- 学号必须为10位数字

sname VARCHAR(20) NOT NULL,

sex CHAR(2) NOT NULL CHECK (sex IN ('男', '女')),

birthdate DATE NOT NULL CHECK (birthdate > '1990-01-01'),

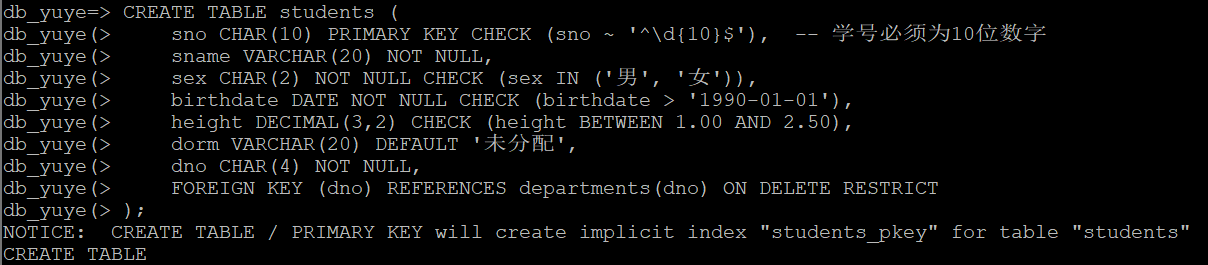
height DECIMAL(3,2) CHECK (height BETWEEN 1.00 AND 2.50),

dorm VARCHAR(20) DEFAULT '未分配',

dno CHAR(4) NOT NULL,

FOREIGN KEY (dno) REFERENCES departments(dno) ON DELETE RESTRICT

);



创建课程表

CREATE TABLE courses (

cno VARCHAR(10) PRIMARY KEY CHECK (cno ~ '^[A-Z]{2}-\d{3}$'), -- 课程编号格式：CS-001

cname VARCHAR(40) NOT NULL UNIQUE,

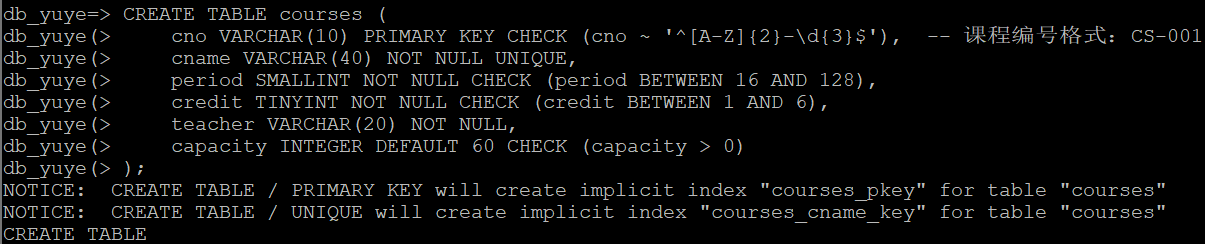
period SMALLINT NOT NULL CHECK (period BETWEEN 16 AND 128),

credit TINYINT NOT NULL CHECK (credit BETWEEN 1 AND 6),

teacher VARCHAR(20) NOT NULL,

capacity INTEGER DEFAULT 60 CHECK (capacity > 0)

);



创建选课表

CREATE TABLE sc (

sno CHAR(10) NOT NULL,

cno VARCHAR(10) NOT NULL,

grade DECIMAL(4,1) DEFAULT NULL CHECK (grade BETWEEN 0 AND 100),

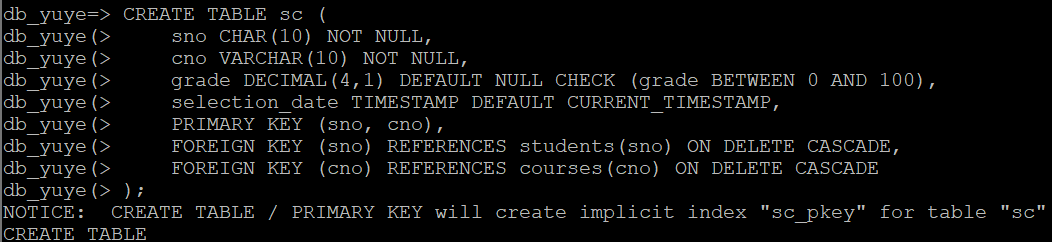
selection\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (sno, cno),

FOREIGN KEY (sno) REFERENCES students(sno) ON DELETE CASCADE,

FOREIGN KEY (cno) REFERENCES courses(cno) ON DELETE CASCADE

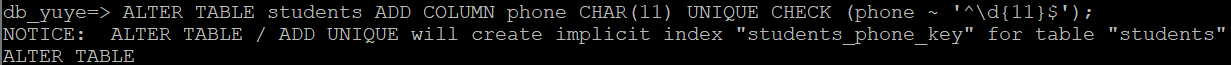
);



2.3表结构修改操作

为学生表添加联系方式字段

ALTER TABLE students ADD COLUMN phone CHAR(11) UNIQUE CHECK (phone ~ '^\d{11}$');



修改课程表字段类型

ALTER TABLE courses ALTER COLUMN teacher TYPE VARCHAR(30);



为院系表添加描述字段

ALTER TABLE departments ADD COLUMN description TEXT DEFAULT '暂无描述';



2.4索引创建与删除

在选课表的成绩字段上创建索引

CREATE INDEX idx\_sc\_grade ON sc(grade DESC);



在学生表的院系字段上创建索引

CREATE INDEX idx\_students\_dno ON students(dno);



删除成绩索引

DROP INDEX idx\_sc\_grade;



2.5数据操作（DML）

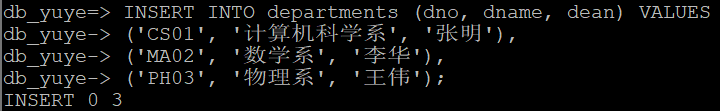
插入院系数据

INSERT INTO departments (dno, dname, dean) VALUES

('CS01', '计算机科学系', '张明'),

('MA02', '数学系', '李华'),

('PH03', '物理系', '王伟');



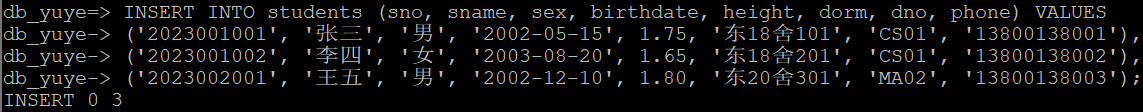
插入学生数据

INSERT INTO students (sno, sname, sex, birthdate, height, dorm, dno, phone) VALUES

('2023001001', '张三', '男', '2002-05-15', 1.75, '东18舍101', 'CS01', '13800138001'),

('2023001002', '李四', '女', '2003-08-20', 1.65, '东18舍201', 'CS01', '13800138002'),

('2023002001', '王五', '男', '2002-12-10', 1.80, '东20舍301', 'MA02', '13800138003');



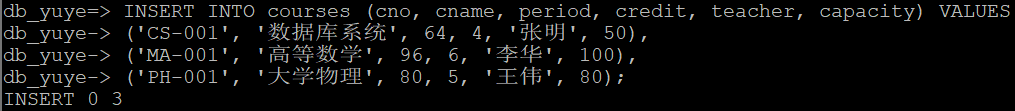
插入课程数据

INSERT INTO courses (cno, cname, period, credit, teacher, capacity) VALUES

('CS-001', '数据库系统', 64, 4, '张明', 50),

('MA-001', '高等数学', 96, 6, '李华', 100),

('PH-001', '大学物理', 80, 5, '王伟', 80);



插入选课数据

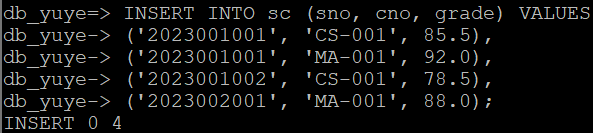
INSERT INTO sc (sno, cno, grade) VALUES

('2023001001', 'CS-001', 85.5),

('2023001001', 'MA-001', 92.0),

('2023001002', 'CS-001', 78.5),

('2023002001', 'MA-001', 88.0);



修改学生宿舍信息

UPDATE students SET dorm = '东18舍102' WHERE sno = '2023001001';



调整课程学分

UPDATE courses SET credit = credit + 1 WHERE cno = 'CS-001';



成绩调整（将数据库课程成绩低于80分的提高5分）

UPDATE sc SET grade = grade + 5 WHERE cno = 'CS-001' AND grade < 80;



删除选课成绩低于60分的记录

DELETE FROM sc WHERE grade < 60;



删除特定学生记录（级联删除选课记录）

DELETE FROM students WHERE sno = '2023002001';

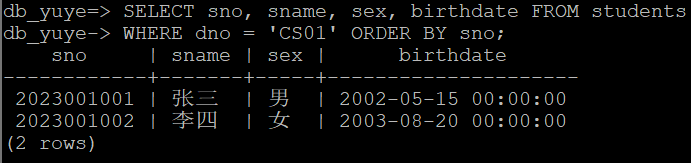


2.6数据库查询操作

查询计算机科学系所有学生信息，按学号排序

SELECT sno, sname, sex, birthdate FROM students

WHERE dno = 'CS01' ORDER BY sno;



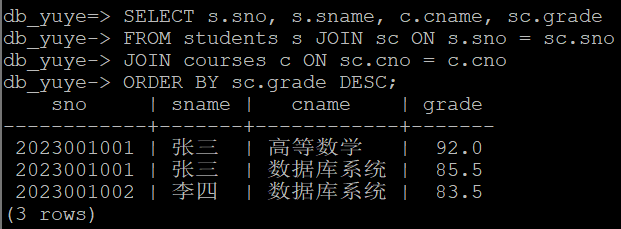
查询学生选课详情（学号、姓名、课程名、成绩），按成绩降序排列

SELECT s.sno, s.sname, c.cname, sc.grade

FROM students s JOIN sc ON s.sno = sc.sno

JOIN courses c ON sc.cno = c.cno

ORDER BY sc.grade DESC;



统计每门课程的平均分、最高分、最低分和选课人数

SELECT c.cno, c.cname,

AVG(sc.grade) AS avg\_grade,

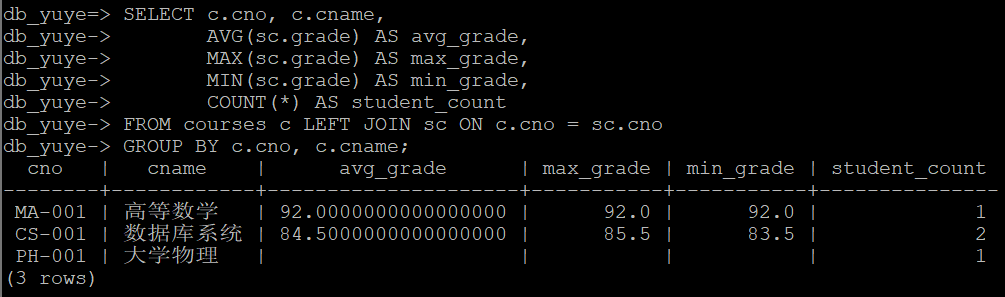
MAX(sc.grade) AS max\_grade,

MIN(sc.grade) AS min\_grade,

COUNT(\*) AS student\_count

FROM courses c LEFT JOIN sc ON c.cno = sc.cno

GROUP BY c.cno, c.cname;



按院系统计学生人数和平均身高

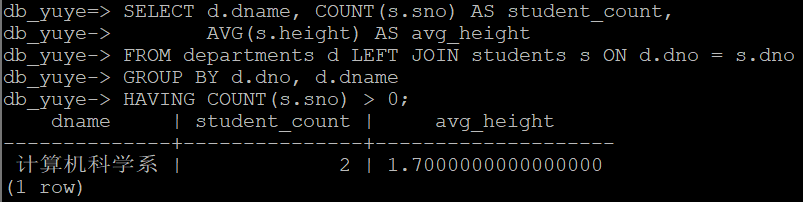
SELECT d.dname, COUNT(s.sno) AS student\_count,

AVG(s.height) AS avg\_height

FROM departments d LEFT JOIN students s ON d.dno = s.dno

GROUP BY d.dno, d.dname

HAVING COUNT(s.sno) > 0;



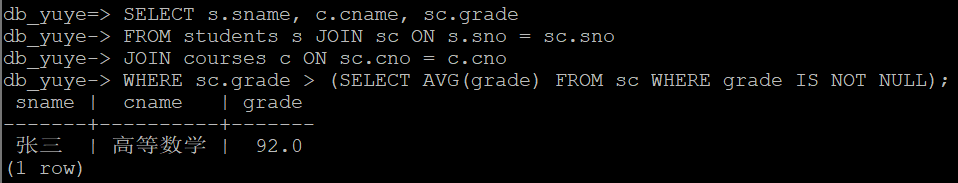
查询成绩高于平均分的学生选课信息

SELECT s.sname, c.cname, sc.grade

FROM students s JOIN sc ON s.sno = sc.sno

JOIN courses c ON sc.cno = c.cno

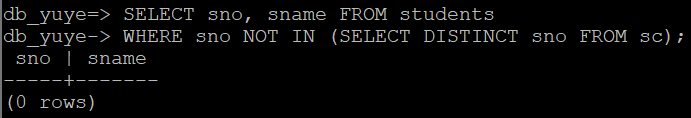
WHERE sc.grade > (SELECT AVG(grade) FROM sc WHERE grade IS NOT NULL);



查询没有选课的学生信息

SELECT sno, sname FROM students

WHERE sno NOT IN (SELECT DISTINCT sno FROM sc);



2.7视图创建与使用

创建学生选课详情视图

CREATE VIEW student\_course\_detail AS

SELECT s.sno, s.sname, s.sex, d.dname,

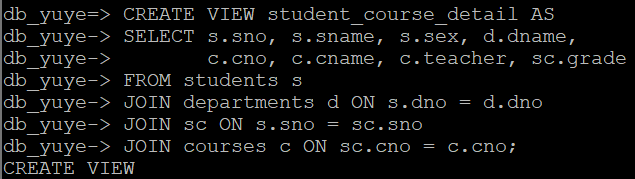
c.cno, c.cname, c.teacher, sc.grade

FROM students s

JOIN departments d ON s.dno = d.dno

JOIN sc ON s.sno = sc.sno

JOIN courses c ON sc.cno = c.cno;



创建院系学生统计视图

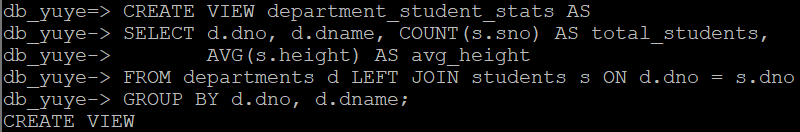
CREATE VIEW department\_student\_stats AS

SELECT d.dno, d.dname, COUNT(s.sno) AS total\_students,

AVG(s.height) AS avg\_height

FROM departments d LEFT JOIN students s ON d.dno = s.dno

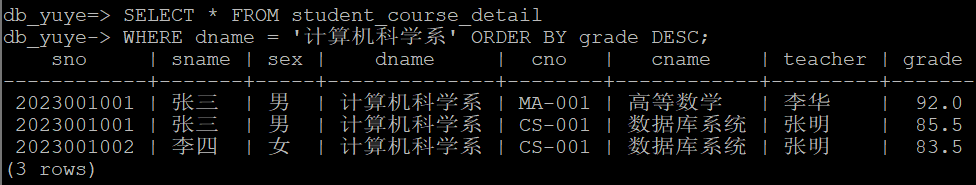
GROUP BY d.dno, d.dname;



查询计算机科学系学生选课情况

SELECT \* FROM student\_course\_detail

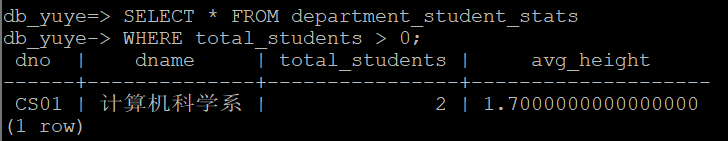
WHERE dname = '计算机科学系' ORDER BY grade DESC;



查询各院系学生统计信息

SELECT \* FROM department\_student\_stats

WHERE total\_students > 0;



## 实验总结

本实验通过设计并实现“学生选课管理系统”数据库，全面应用了MySQL数据库的相关知识，包括数据库设计与建模、SQL数据定义语言、SQL数据操作语言、数据查询与优化，以及高级特性（如索引管理和视图创建）。在实践过程中，遇到了多个问题：例如约束验证失败时，通过预先数据验证和BETWEEN条件确保合规性；外键引用冲突通过严格遵循插入顺序（院系→学生→课程→选课）解决；查询性能问题通过创建索引和使用EXPLAIN分析优化；数据更新逻辑错误通过添加NULL值检查避免误操作；级联删除导致数据丢失风险通过软删除或备份策略缓解；视图权限问题通过明确字段别名和基表权限验证处理。这些经验深化了我对数据库完整性、性能平衡和开发流程的理解，掌握了从需求分析到设计实现的完整技能，为后续复杂数据库应用奠定了坚实基础。

学号：102301116 姓名：俞烨 专业：计算机科学与技术 班级：计算机类01班 《数据库应用实践》

# 实验二：数据库管理系统的维护与管理

实验环境：openEuler+openGauss数据库

实验内容：

## 1、 数据库安全性实现流程

1.1 数据库账户的添加与删除

添加账户：使用CREATE USER语句添加用户。

CREATE USER joe WITH PASSWORD 'joe@123';



删除账户：使用DROP USER语句移除用户。

DROP USER IF EXISTS old\_joe;

IF EXISTS确保用户不存在时不报错。



查看是否删除：  
SELECT usename FROM pg\_user; 

1.2对账户授予权限与收回权限

首先创建一个employees表并插入数据用于测试：

CREATE TABLE employees (

employee\_id INTEGER PRIMARY KEY,

employee\_name VARCHAR(100) NOT NULL,

salary NUMERIC(10, 2),

department VARCHAR(50),

hire\_date DATE,

last\_updated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);  
INSERT INTO employees (employee\_id, employee\_name, salary, department, hire\_date)

VALUES

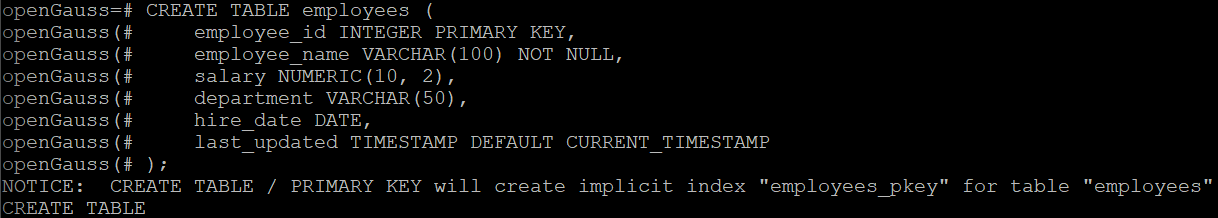
(1, '张三', 50000.00, '销售部', '2020-03-15'),

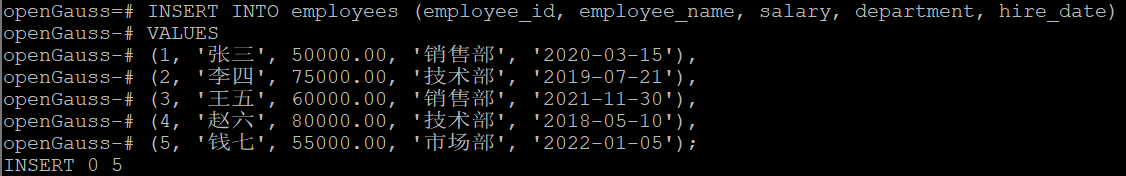
(2, '李四', 75000.00, '技术部', '2019-07-21'),

(3, '王五', 60000.00, '销售部', '2021-11-30'),

(4, '赵六', 80000.00, '技术部', '2018-05-10'),

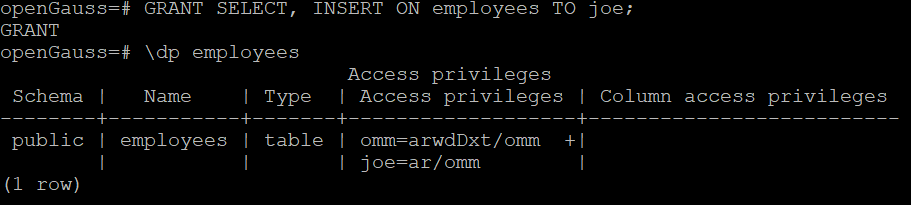
(5, '钱七', 55000.00, '市场部', '2022-01-05');





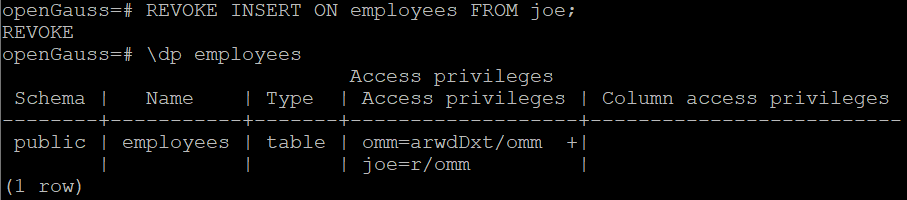
授予权限：使用GRANT语句分配权限。

GRANT SELECT, INSERT ON employees TO joe;



收回权限：使用REVOKE语句撤销权限。

REVOKE INSERT ON employees FROM joe;



## 2、触发器与存储过程的使用实现流程

2.1创建存储过程并执行

创建存储过程：使用CREATE PROCEDURE语句定义过程。

CREATE OR REPLACE PROCEDURE update\_salary(

IN emp\_id INT,

IN new\_salary DECIMAL

)

AS

BEGIN

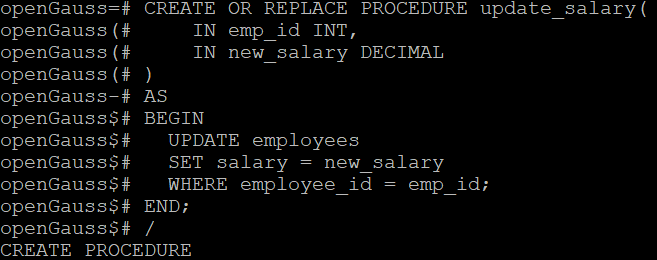
UPDATE employees

SET salary = new\_salary

WHERE employee\_id = emp\_id;

END;

/



此过程接受员工ID和新薪水参数，执行更新操作。

执行存储过程：

CALL update\_salary(1, 5000);



2.2 创建触发器并测试效果

创建触发器：需先创建触发器函数，再用CREATE TRIGGER绑定。：

先创建审计日志表用于存储日志：

CREATE TABLE audit\_log (

log\_id BIGSERIAL PRIMARY KEY, -- 自增主键

action VARCHAR(10), -- 操作类型: INSERT, UPDATE, DELETE

table\_name VARCHAR(50), -- 发生操作的表名

record\_id INT, -- 被操作的记录ID

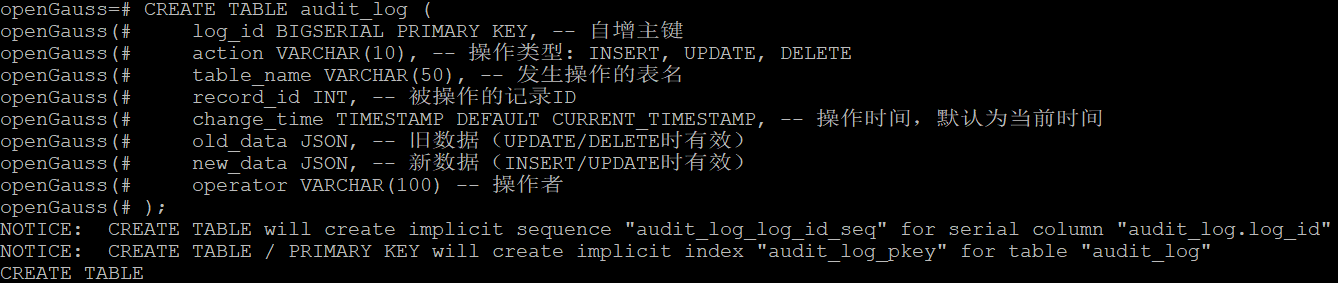
change\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- 操作时间，默认为当前时间

old\_data JSON, -- 旧数据（UPDATE/DELETE时有效）

new\_data JSON, -- 新数据（INSERT/UPDATE时有效）

operator VARCHAR(100) -- 操作者

);



创建触发器函数：

CREATE OR REPLACE FUNCTION log\_employee\_change()

RETURNS TRIGGER AS

$$

BEGIN

-- 使用条件判断处理不同的操作类型 (INSERT/UPDATE/DELETE)

IF (TG\_OP = 'DELETE') THEN

INSERT INTO audit\_log (action, table\_name, record\_id, change\_time)

VALUES (TG\_OP, 'employees', OLD.employee\_id, CURRENT\_TIMESTAMP);

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE' OR TG\_OP = 'INSERT') THEN

INSERT INTO audit\_log (action, table\_name, record\_id, change\_time)

VALUES (TG\_OP, 'employees', NEW.employee\_id, CURRENT\_TIMESTAMP);

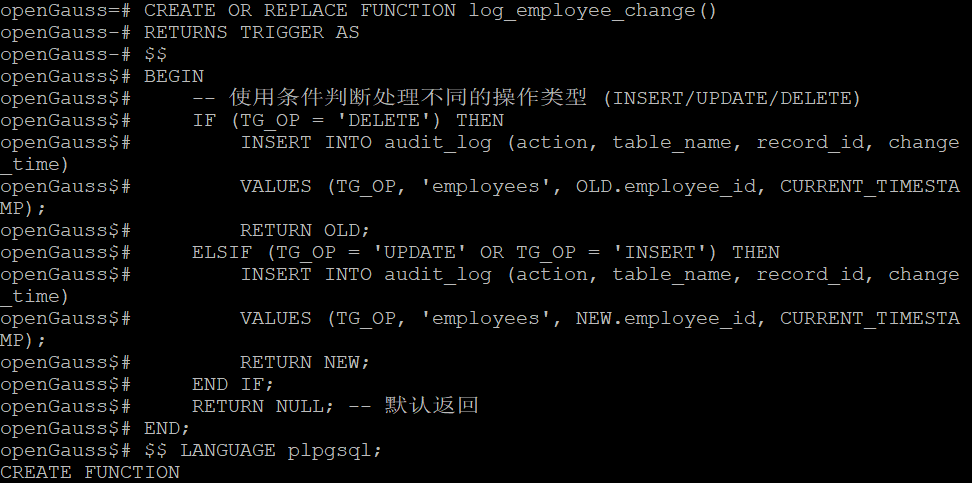
RETURN NEW;

END IF;

RETURN NULL; -- 默认返回

END;

$$ LANGUAGE plpgsql;



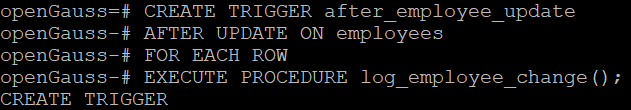
再创建触发器绑定到employees表：

CREATE TRIGGER after\_employee\_update

AFTER UPDATE ON employees

FOR EACH ROW

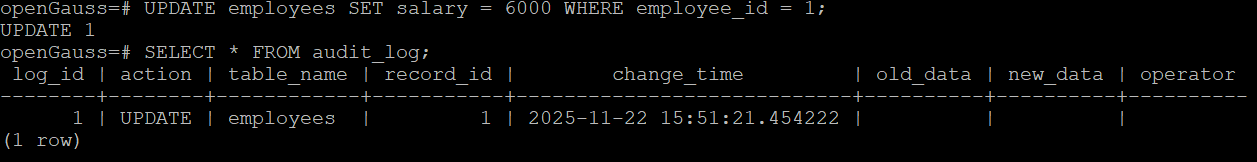
EXECUTE PROCEDURE log\_employee\_change();



测试触发器效果：更新employees表数据，验证audit\_log表是否自动记录：

UPDATE employees SET salary = 6000 WHERE employee\_id = 1;

SELECT \* FROM audit\_log;



## 3、数据库备份与恢复实现流程

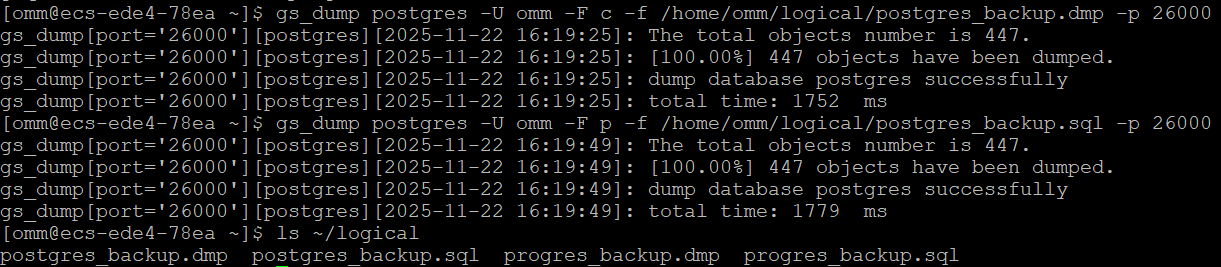
备份与恢复是数据库维护的核心，openGauss支持逻辑备份工具gs\_dump和恢复工具gs\_restore。

3.1对所创建的数据库进行备份

使用逻辑备份：在终端中，运行gs\_dump命令备份整个数据库或特定对象。例如，备份数据库postgres到文件postgres\_backup.sql：

gs\_dump postgres -U omm -F p -f /home/omm/logical/postgres\_backup.sql -p 26000

-U指定管理员账户，-f定义输出文件路径，-F p/c指定纯文本格式(dmp格式)。备份包含表结构、数据及权限信息。



3.2利用备份进行数据库恢复

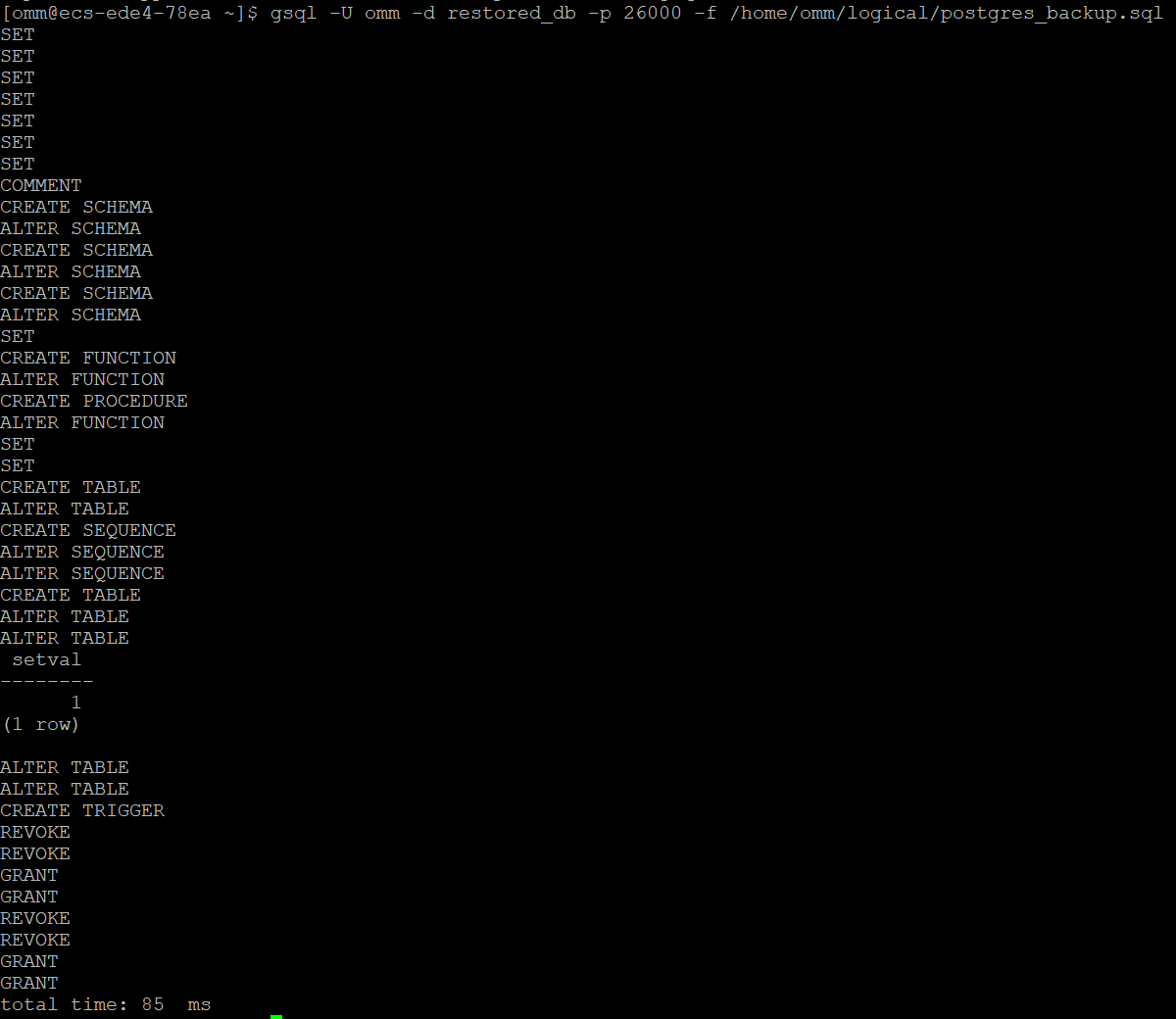
恢复数据库：使用gs\_restore工具从备份文件恢复。例如，恢复postgres\_backup.sql到恢复数据库restored\_db：

首先创建恢复数据库：

CREATE DATABASE restored\_db OWNER omm;



gsql -U omm -d restored\_db -p 26000 -f ~/logical/postgres\_backup.sql



## **实验总结**

本次实验围绕数据库管理系统的维护与管理展开，通过openGauss数据库环境实践了数据库安全性、触发器与存储过程的使用以及备份与恢复三大核心模块。在数据库安全性部分，实验涵盖了账户的添加与删除，使用CREATE USER和DROP USER语句管理用户，并通过GRANT和REVOKE语句实现对特定表（如employees表）的权限授予与收回，确保访问控制。触发器与存储过程部分演示了如何创建和执行存储过程（如update\_salary过程用于更新薪水），以及触发器的实现，通过创建audit\_log审计表和触发器函数log\_employee\_change，自动记录数据变更日志，测试了UPDATE操作触发效果。备份与恢复部分利用gs\_dump工具进行逻辑备份（如备份postgres数据库到文件）和gsql工具进行恢复，包括创建新数据库并导入备份数据，强调了备份策略的重要性。整体上，实验强化了数据库维护的关键技能，如权限管理、自动化审计和灾难恢复，虽然基于openGauss语法，但核心概念与MySQL类似，例如在MySQL中账户管理使用CREATE USER和GRANT语句，触发器通过CREATE TRIGGER定义，备份可使用mysqldump工具，本次实践为数据库管理提供了扎实基础。

学号：102301116 姓名：俞烨 专业：计算机科学与技术 班级：计算机类01班 《数据库应用实践》

# **实验三：数据库应用系统的开发**

实验环境：openEuler+openGauss数据库+windows/linux部署

实验内容：

## 一、系统选题与工作情况说明

选题：学生选课与成绩管理系统（B/S 架构）。

角色与功能：

学生：查看可选课程、选课/退选、查看所选课程成绩、修改密码。

教师：查看自己授课的课程、查看课程学生名单、录入/修改成绩、修改密码。

管理员：学生/教师/课程/选课全量 CRUD，Excel 批量导入课程与学生名单，导出课程成绩 Excel，用户与权限管理。

技术栈：

前端：Vue 3 + Vite，Fetch API 调用后端 REST。

后端：Flask + psycopg2 + openGauss；可选 SSH 隧道（sshtunnel/paramiko）。

数据库：openGauss（兼容 PostgreSQL 语法）。

主要工作概览：

设计与实现用户认证与基于角色的权限控制（Session）。

数据库概念与逻辑模型设计、DDL 与索引、约束实现。

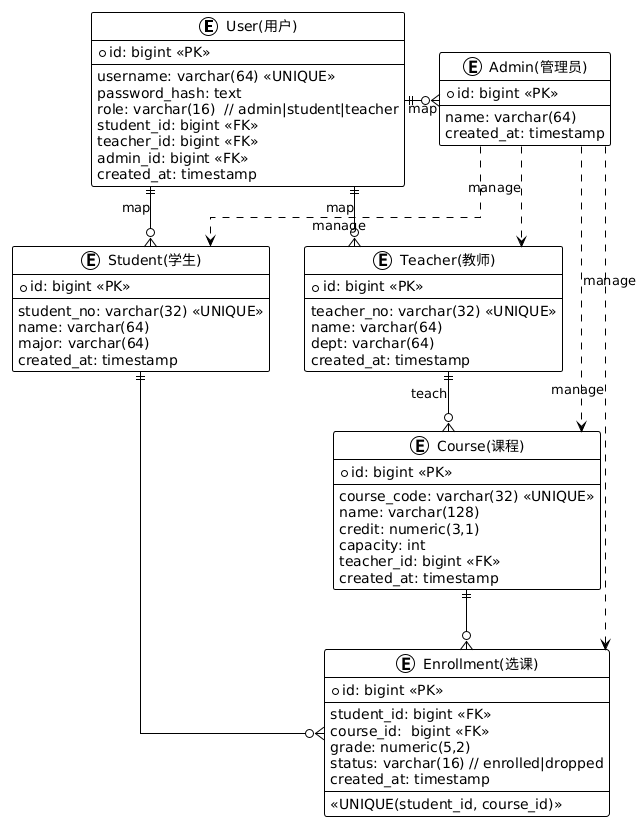
完整 REST API（学生/教师/管理员）。

Excel 导入课程/学生/选课、导出成绩（中文表头，文件名“课程-老师-时间.xlsx”）。

前端三大界面实现与美化（紧凑简约设计）。

## 二、概念数据模型（E-R 图）

实体：用户(User)、管理员(Admin)、学生(Student)、教师(Teacher)、课程(Course)、选课(Enrollment)。



## 三、系统中每张表的说明

**users**

* username：登录名，管理员为 admin；学生/教师默认规则支持“学号/工号”。
* password\_hash：口令哈希（后端统一校验）。
* role：student/teacher/admin。
* student\_id/teacher\_id：与真实身份关联（其一为空）。

**students**

* student\_no：学号，唯一。
* name/major：基本信息。

**teachers**

* teacher\_no：工号，唯一。
* name/dept：基本信息。

**courses**

* course\_code：课程号，唯一。
* name/credit/capacity：课程属性。
* teacher\_id：授课教师（FK）。

**enrollments**

* student\_id/course\_id：复合唯一约束防重复选课。
* grade：成绩，可空。
* status：enrolled/dropped，用于退选保留历史。

数据定义语言（DDL）（openGauss/PostgreSQL 语法）：

CREATE TABLE IF NOT EXISTS users (

id bigserial PRIMARY KEY,

username varchar(64) UNIQUE NOT NULL,

password\_hash text NOT NULL,

role varchar(16) NOT NULL CHECK (role IN ('admin','student','teacher')),

student\_id bigint REFERENCES students(id) ON DELETE SET NULL,

teacher\_id bigint REFERENCES teachers(id) ON DELETE SET NULL,

created\_at timestamp DEFAULT now()

);

CREATE TABLE IF NOT EXISTS students (

id bigserial PRIMARY KEY,

student\_no varchar(32) UNIQUE NOT NULL,

name varchar(64) NOT NULL,

major varchar(64),

created\_at timestamp DEFAULT now()

);

CREATE TABLE IF NOT EXISTS teachers (

id bigserial PRIMARY KEY,

teacher\_no varchar(32) UNIQUE NOT NULL,

name varchar(64) NOT NULL,

dept varchar(64),

created\_at timestamp DEFAULT now()

);

CREATE TABLE IF NOT EXISTS courses (

id bigserial PRIMARY KEY,

course\_code varchar(32) UNIQUE NOT NULL,

name varchar(128) NOT NULL,

credit numeric(3,1) DEFAULT 0,

capacity int DEFAULT 50,

teacher\_id bigint REFERENCES teachers(id) ON DELETE SET NULL,

created\_at timestamp DEFAULT now()

);

CREATE TABLE IF NOT EXISTS enrollments (

id bigserial PRIMARY KEY,

student\_id bigint NOT NULL REFERENCES students(id) ON DELETE CASCADE,

course\_id bigint NOT NULL REFERENCES courses(id) ON DELETE CASCADE,

grade numeric(5,2),

status varchar(16) DEFAULT 'enrolled' CHECK (status IN ('enrolled','dropped')),

created\_at timestamp DEFAULT now(),

CONSTRAINT uq\_enroll UNIQUE (student\_id, course\_id)

);

-- 常用索引

CREATE INDEX IF NOT EXISTS idx\_course\_teacher ON courses(teacher\_id);

CREATE INDEX IF NOT EXISTS idx\_enroll\_student ON enrollments(student\_id);

CREATE INDEX IF NOT EXISTS idx\_enroll\_course ON enrollments(course\_id);

## 四、系统运行环境配置与关键语句

环境

OS：Windows/Linux 开发机 + 云服务器 Linux（openEuler 等）

Python 3.10+，Node 18+，openGauss 5.x

后端依赖：Flask、psycopg2-binary、sshtunnel、paramiko、openpyxl、pandas

安装与运行（Windows\Linux）：

cd /path/to/repository

# 后端

python -m venv .venv

.venv/Scripts/Activate.ps1（source .venv/Scripts/activate）

pip install -r requirements.txt

cd backend

python main.py

# 前端

cd frontend/vue

npm install

npm run dev

数据库连接关键语句（后端简化片段）：

# 连接池 + 可选 SSH 隧道

from psycopg2 import pool

from sshtunnel import SSHTunnelForwarder

if os.getenv('OG\_SSH\_TUNNEL') == 'true':

tunnel = SSHTunnelForwarder(

(os.getenv('OG\_SSH\_HOST'), int(os.getenv('OG\_SSH\_PORT', 22))),

ssh\_username=os.getenv('OG\_SSH\_USER'),

ssh\_password=os.getenv('OG\_SSH\_PASSWORD'),

remote\_bind\_address=(os.getenv('OG\_TUNNEL\_REMOTE\_HOST','127.0.0.1'),

int(os.getenv('OG\_PORT', 5432)))

)

tunnel.start()

host, port = '127.0.0.1', tunnel.local\_bind\_port

else:

host, port = os.getenv('OG\_HOST'), int(os.getenv('OG\_PORT', 5432))

pgpool = pool.SimpleConnectionPool(

1, 10,

host=host, port=port,

dbname=os.getenv('OG\_DBNAME'),

user=os.getenv('OG\_USER'),

password=os.getenv('OG\_PASSWORD'),

)

前后端跨域与会话

Flask：CORS 指定 origin，开启 supports\_credentials；Session SameSite=Lax + HttpOnly。

前端 fetch：统一携带 credentials: 'include'。

Excel 导入/导出（关键说明）

导入工作簿须包含 Sheet“courses”，可选“students”“enrollments”。

导出命名：<课程名称>-<授课老师>-<导出时间>.xlsx，中文表头。

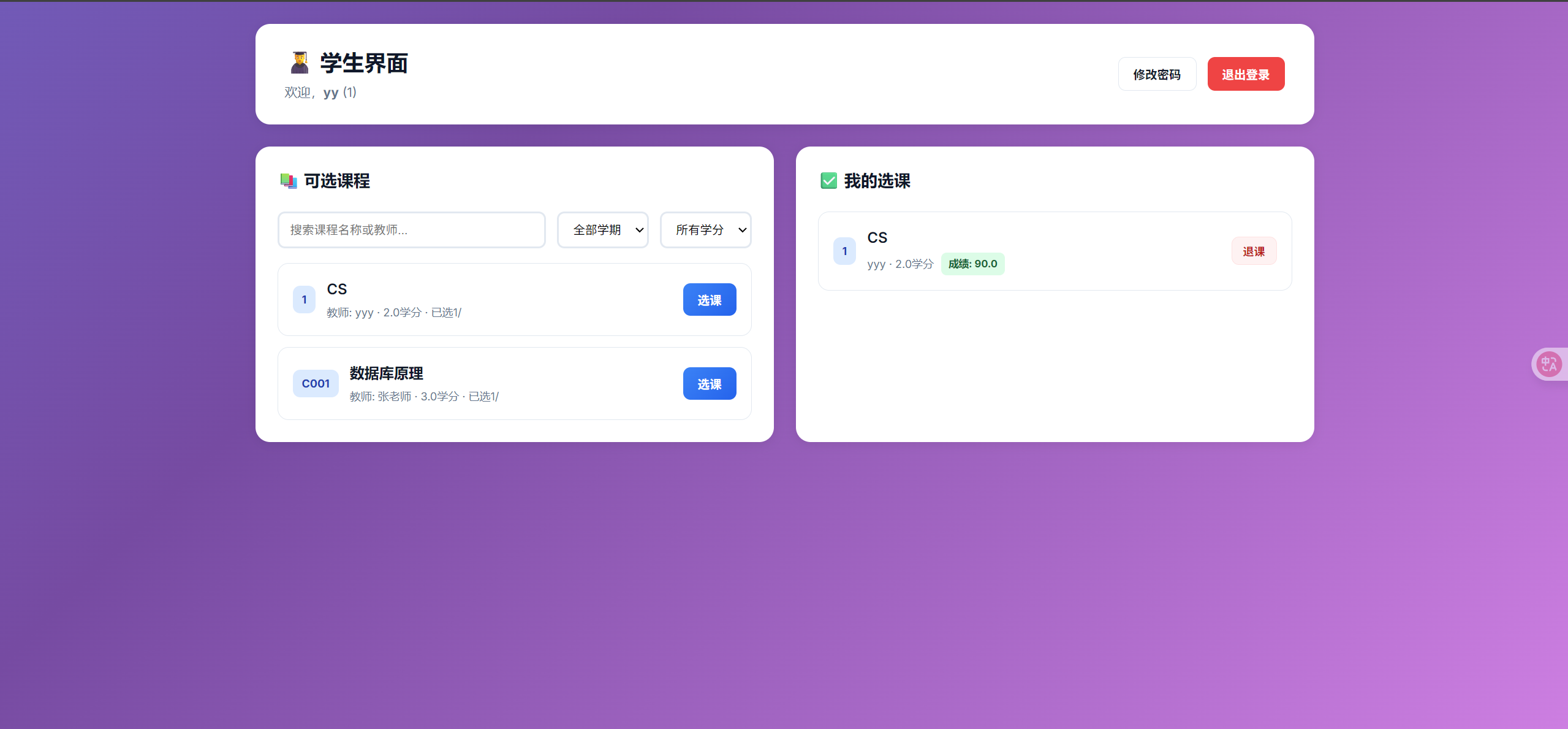
示例生成脚本：backend/scripts/generate\_sample\_excel.py（一键生成 sample\_import.xlsx）。

## 五、系统主要功能界面

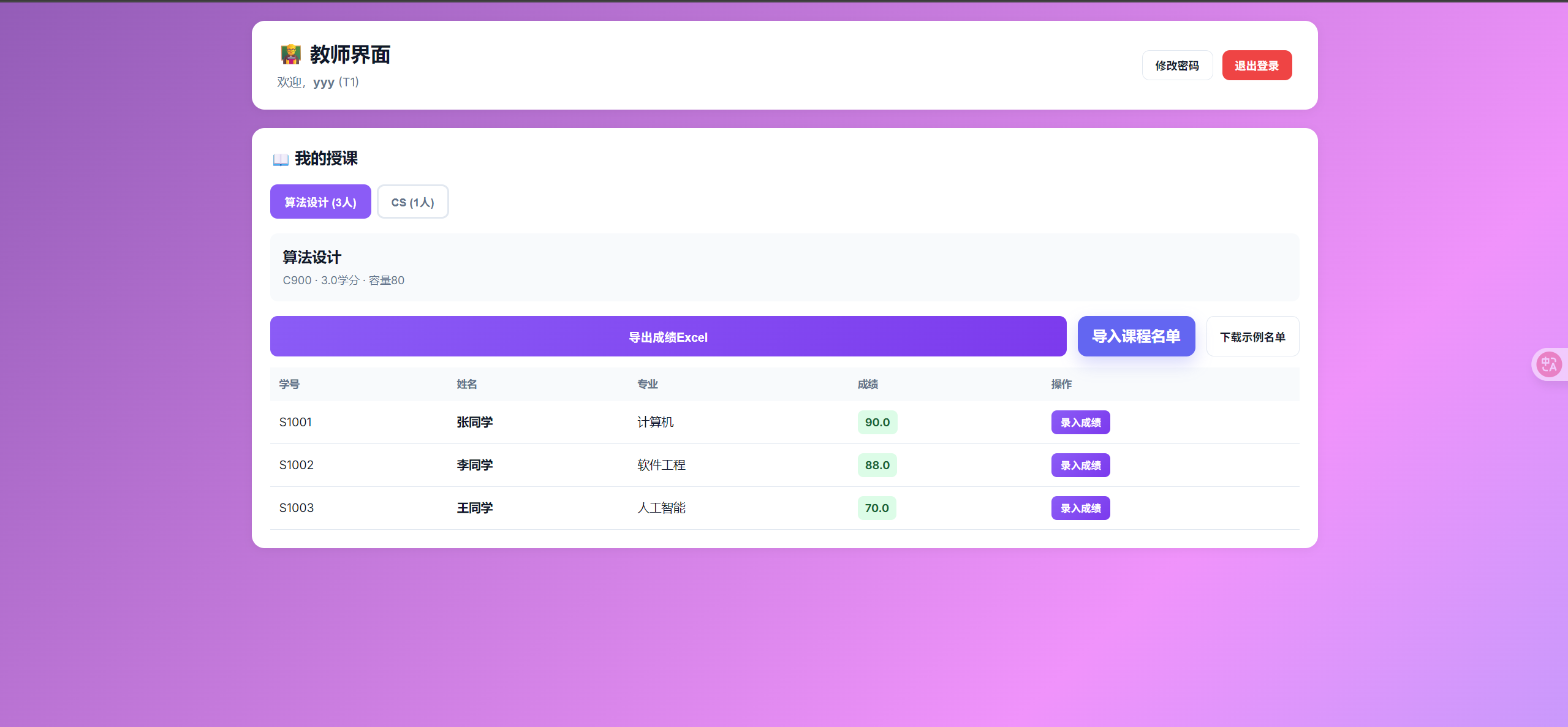
登录页：支持三角色登录与修改密码。



学生页：左侧“可选课程”，右侧“我的选课”；支持选课、退选、显示成绩。



教师页：展示“我的授课”卡片；进入课程可查看学生名单与录入/修改成绩。



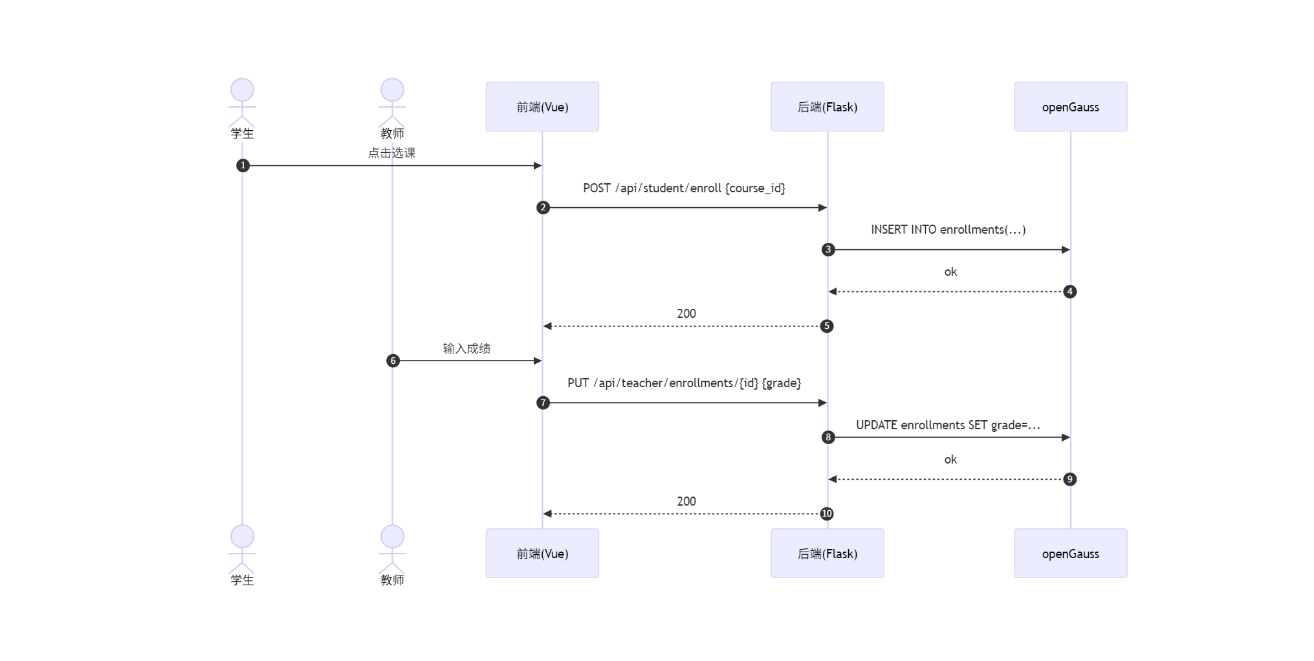
管理员页：学生/教师/课程/选课 CRUD；Excel 导入课程与学生名单；导出课程成绩。







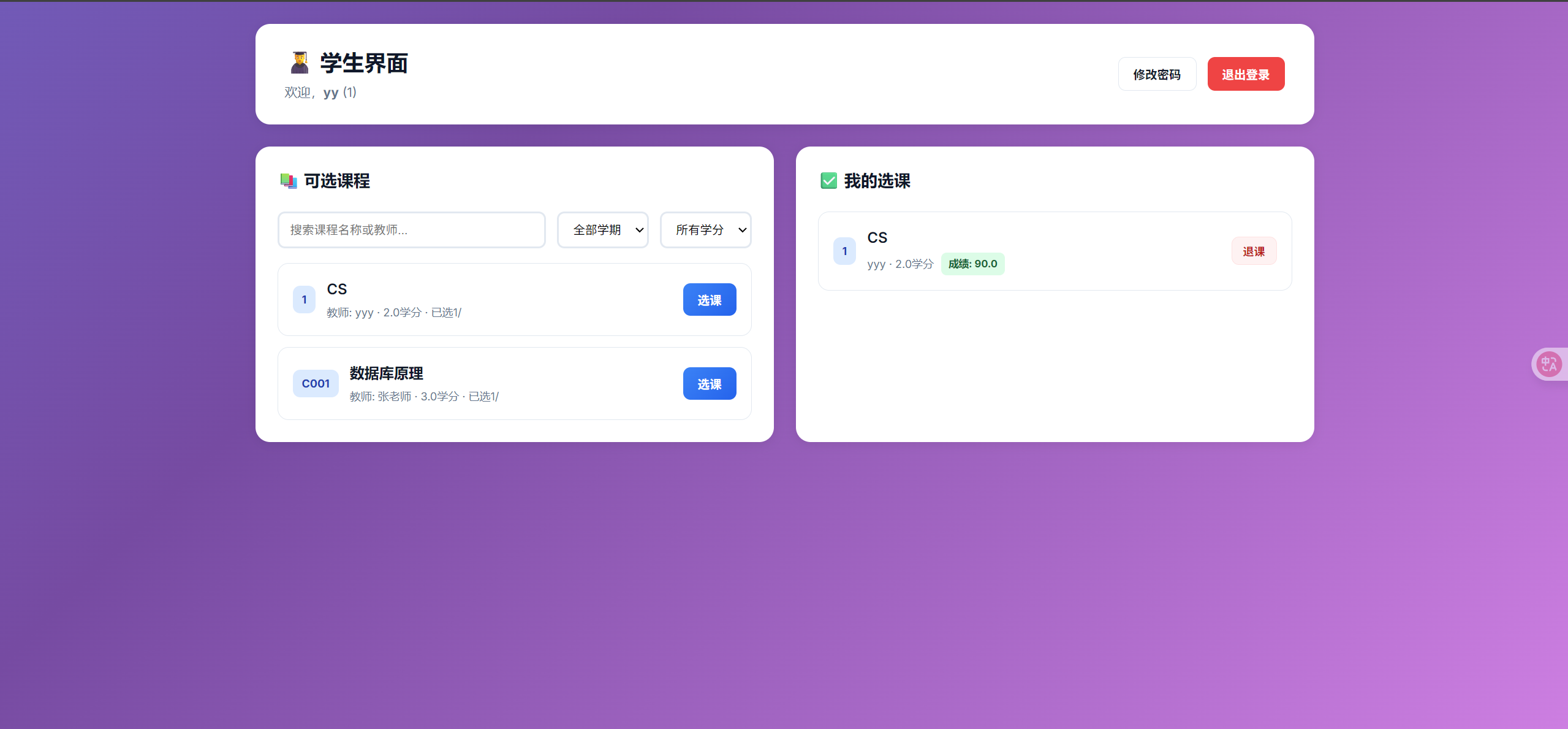
核心交互流程（选课/录分）：



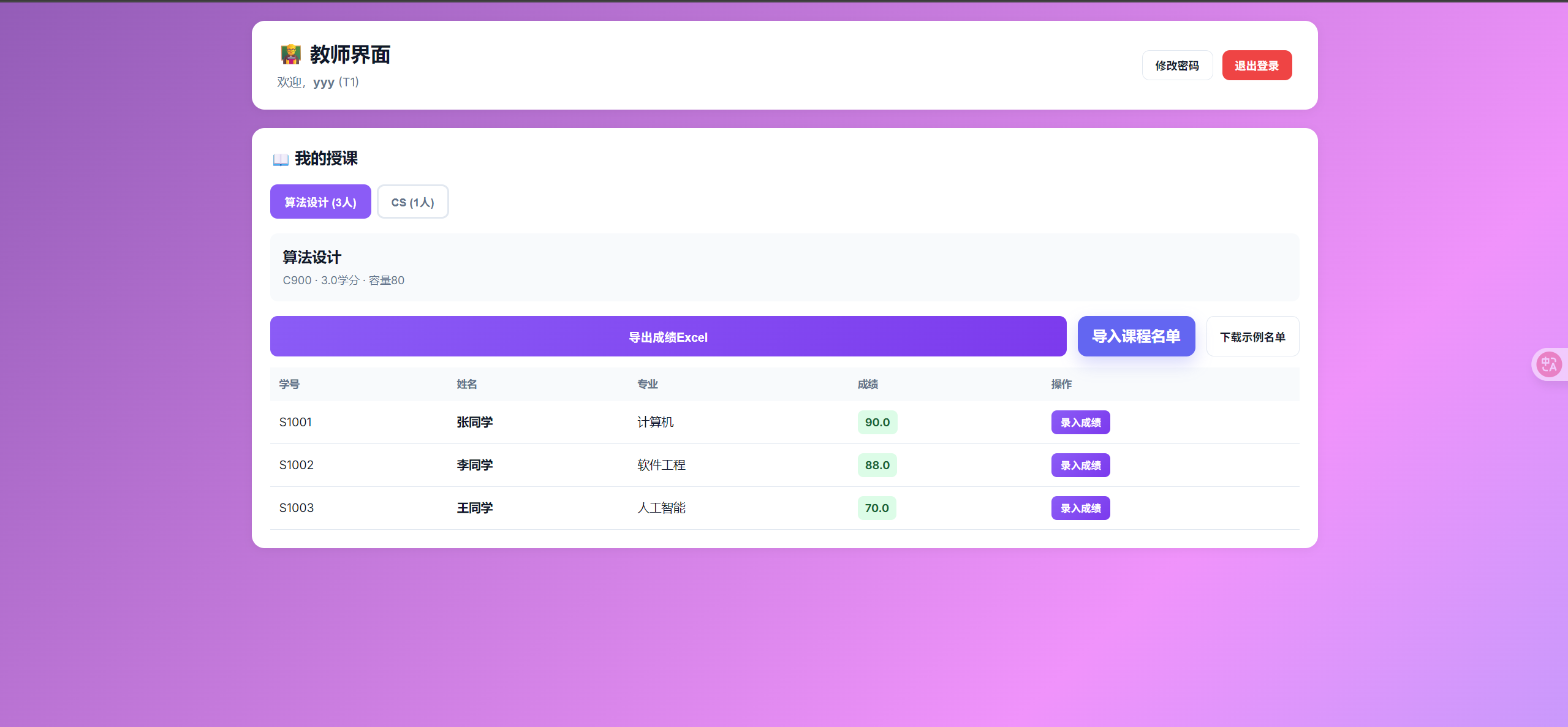
六、实验结果总结

运行效果：

学生端可正常选课/退选、查看成绩。



教师端能查看所授课程及名单，修改成绩即时生效。



管理端可通过 Excel 批量导入课程/学生/选课，导出成绩表，包含课程信息与成绩表，表头中文且命名按“课程-老师-时间”。





优点：

架构清晰，前后端分离；服务端分层（API/Service/DB）。

约束完善（唯一、外键、复合唯一），避免重复选课。

可选 SSH 隧道，适配云上私网数据库。

Excel 批处理提升数据运营效率。

不足与改进：

成绩审核与日志待完善（审计表）。

课程容量的并发控制可引入“扣减名额原子事务 + 队列”。

可以增加课程时间冲突检测与选课规则（先修课等）。

统一前端状态管理（Pinia）与接口错误提示更细致。

## 七、编程工作总结

**数据库设计经验**

先概念后逻辑：从 E-R（含 Admin/Student/Teacher/Course/Enrollment）到 DDL，逐步固化约束与索引。

约束前置：唯一键（username, student\_no, teacher\_no, course\_code）、复合唯一（student\_id, course\_id）防止重复选课，外键保证引用完整性，枚举/检查约束限定 status 与 role。

弹性字段：成绩允许为空以兼容“未评分”场景；教师关联可空以支持待分配课程；容量/学分设默认值减少数据录入成本。

索引实践：课程教师索引、选课学生/课程索引覆盖高频查询；复合唯一同时充当索引，兼顾一致性与性能。

数据导入健壮性：导入时先缓存现有键值映射（student\_no/teacher\_no/course\_code），幂等插入、跳过重复并汇总错误；缺失学生可按需创建以提高可用性。

**后端实现要点**

连接与安全：psycopg2 连接池 + 参数化查询防 SQL 注入；可选 SSH 隧道适配云上私网数据库。

分层解耦：蓝图按角色拆分（auth/student/teacher/admin），Service 层承载业务逻辑，DB 层只做数据访问。

会话与跨域：Flask-Session + SameSite=Lax Cookie，CORS 显式 origin + credentials，前端统一带凭证。

错误与权限：统一 json\_response/error\_response；@require\_auth 进行登录与角色校验；资源级校验（教师只能改自己课程的成绩，学生只能操作自己的选课）。

批处理与文件：pandas+openpyxl 读写 Excel；导出文件名包含课程名、授课老师与时间戳，表头中文；导入支持 courses/students/enrollments 多表。

**前端实现要点**

角色视图：根据用户角色呈现学生/教师/管理员界面；导航/登出统一。

数据拉取：封装 api() 统一 fetch，默认 credentials: 'include'；Promise.all 并行加载列表与统计。

UI 组织：列表/表单组件化，搜索与折叠可控；按钮态、错误提示、空态统一；统计卡片横向紧凑展示。

交互细节：删除/退课二次确认；导入/导出 loading/error 状态；成绩输入留空直接返回。

**工具与自动化**

Excel 工具链：pandas+openpyxl 处理导入/导出；示例生成脚本 generate\_sample\_excel.py 统一模板格式（courses/students/enrollments）。

启动与文档：PowerShell/命令行启动脚本，README 列出后端/前端启动步骤与 Excel 使用说明；.env 示例便于快速配置。

代码组织：前后端分离，Vite 开发热更新；后端模块化便于扩展（新增培养计划等功能时复用 Service/DB 层模式）。

**可选改进方向**

并发控制：选课容量扣减可加事务与行级锁或乐观并发控制，避免超额。

审计与日志：增加操作审计表，记录导入/导出、成绩修改、删除操作。

校验与规则：选课冲突检测、先修课程约束、成绩变更审批流。

前端状态：引入 Pinia/Redux 做全局状态与错误统一收敛，提升可维护性。

附：Excel 示例与导出

一键生成导入模板

cd /path/to/repository/backend/scripts

python generate\_sample\_excel.py   # 生成 sample\_import.xlsx

导入模板各表中文表头

课程(courses)：课程号, 课程名, 学分, 容量, 授课工号, 授课教师

学生(students)：学号, 姓名, 专业

选课(enrollments)：课程号, 学号, 成绩, 状态

成绩导出

管理员界面选择课程 → “导出成绩Excel”

文件命名：课程名称-授课老师-YYYYMMDD\_HHMMSS.xlsx

表头：课程号, 课程名, 学分, 授课教师, 学号, 姓名, 成绩, 选课状态实验