**题 目： 学生学籍管理系统**

**姓 名：**

**学 号：**

**班 级：**

# 问题描述

学生学籍管理系统是一个用于管理学生信息、班级信息、课程信息和成绩信息的系统。该系统提供了学生信息的增加、删除、修改和查询功能，班级信息的增加、删除、修改和查询功能，课程信息的增加、删除、修改和查询功能，以及成绩信息的录入和查询功能。

学生信息管理模块是系统的核心模块之一。通过该模块，管理员可以录入学生的基本信息，包括学号、姓名、性别等，并将学生分配到相应的班级和课程中。管理员可以根据学生的学号或姓名进行查询，并可以对学生信息进行修改或删除操作。这样可以方便地管理学生的个人信息和学籍信息。

班级信息管理模块允许管理员录入班级的信息，包括班级编号、班级名称等。管理员可以根据班级编号或班级名称进行查询，并可以对班级信息进行修改或删除操作。这样可以有效地管理班级的基本信息和学生的分配情况。

课程信息管理模块允许管理员录入课程的信息，包括课程编号、课程名称等。管理员可以根据课程编号或课程名称进行查询，并可以对课程信息进行修改或删除操作。这样可以方便地管理课程的基本信息和学生的选课情况。

成绩信息管理模块允许管理员录入学生的成绩信息，包括学生的学号、课程的编号和成绩。管理员可以根据学生的学号或课程的编号进行查询，并可以对成绩信息进行修改。同时，系统还提供了成绩查询功能，学生和教师可以根据学号或课程编号查询学生成绩，并进行统计和分析。

通过以上功能，学生学籍管理系统可以实现对学生信息、班级信息、课程信息和成绩信息的全面管理。管理员可以方便地录入、修改、删除和查询相关信息，提高了学籍管理的效率和准确性。同时，学生和教师也可以通过系统方便地查询和管理自己的学籍和成绩信息，提高了信息的透明度和可访问性。这样的系统可以有效地支持学校的学籍管理工作，提升学校管理水平和服务质量。

# 需求分析

该需求分析描述了一个学生信息管理系统的基本功能和要求。该系统需要能够管理学生的基本信息，包括学号、姓名、性别、年龄、班级等。这将使学校能够轻松地记录和更新学生的个人信息，并提供一个集中管理的平台。

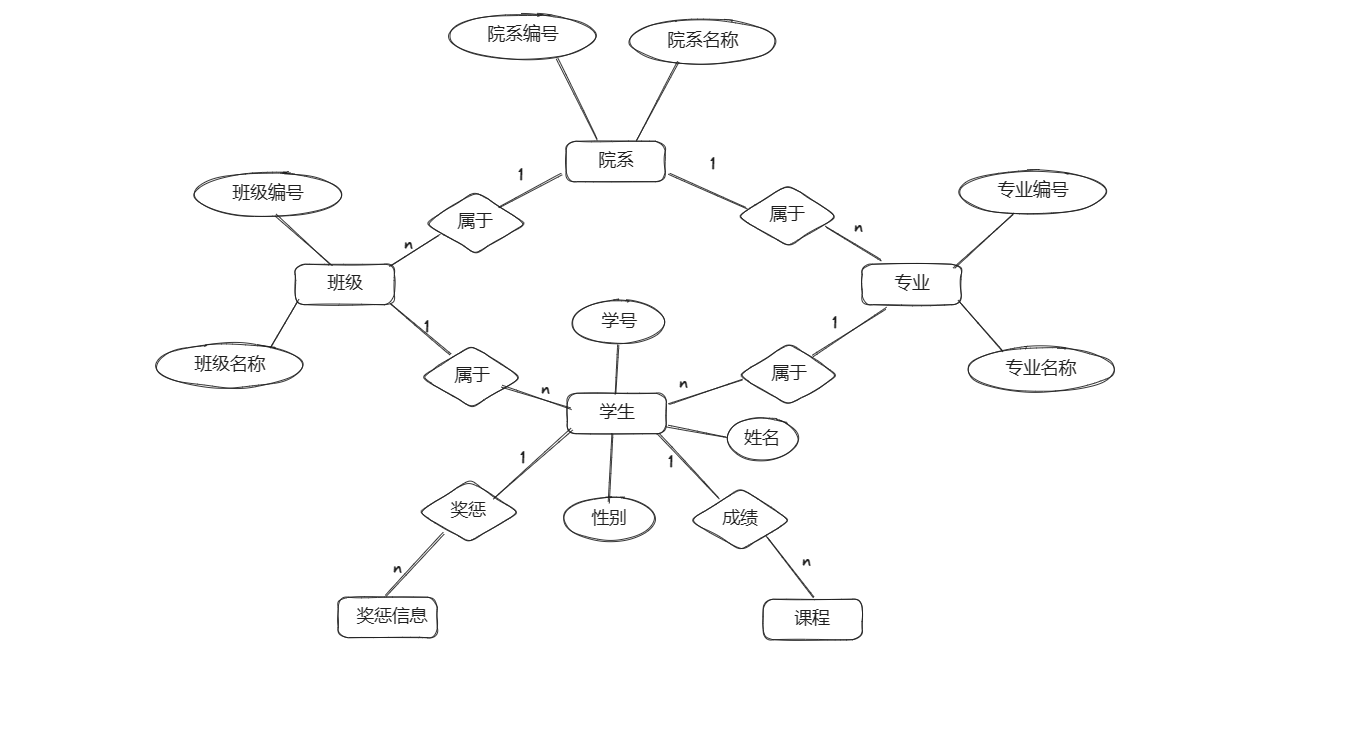
此外，系统还需要能够管理班级信息，包括班级编号、班级名称、所属院系等。这将有助于学校组织和管理不同班级的学生，并提供班级相关信息的查询和统计功能。

另外，系统还需要能够管理课程信息，包括课程编号、课程名称、学分等。这将使学校能够维护课程的基本信息，并方便学生和教师查询和选择课程。

最后，系统还需要能够录入和查询学生的成绩信息，包括学生学号、课程编号、成绩等。这将使学校能够记录和管理学生的学业成绩，并提供成绩查询和统计的功能。学生和教师可以方便地查询个人或班级的成绩情况，学校也可以通过这些数据进行绩效评估和统计分析。

综上所述，该学生信息管理系统将提供学生、教师和学校管理人员所需的基本功能，包括学生信息管理、班级信息管理、课程信息管理和成绩信息管理。通过这个系统，学校可以更高效地管理学生信息，提供更好的服务和支持，进一步提升教育质量和管理水平。

# ER图



# 关系模式

实体的属性描述：  
- 学生（students）实体的属性：student\_id（主键），name，gender，class\_id（外键），major\_id（外键）。  
- 班级（classes）实体的属性：class\_id（主键），class\_name，department\_id（外键）。  
- 院系（departments）实体的属性：department\_id（主键），department\_name。  
- 专业（majors）实体的属性：major\_id（主键），major\_name，department\_id（外键）。  
- 课程（courses）实体的属性：course\_id（主键），course\_name。  
- 成绩（scores）实体的属性：score\_id（主键），student\_id（外键），course\_id（外键），score。  
- 奖惩信息（rewards\_punishments）实体的属性：rp\_id（主键），student\_id（外键），description

# 数据字典

**Table: students**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| student\_id | INT | 学生ID (Primary Key) |
| name | VARCHAR(50) | 学生姓名 |
| gender | ENUM | 学生性别 ('男' 或 '女') |
| class\_id | INT | 班级ID |
| major\_id | INT | 专业ID |

**Table: classes**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| class\_id | INT | 班级ID (Primary Key) |
| class\_name | VARCHAR(50) | 班级名称 |
| department\_id | INT | 院系ID |

**Table: departments**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| department\_id | INT | 院系ID (Primary Key) |
| department\_name | VARCHAR(50) | 院系名称 |

**Table: majors**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| major\_id | INT | 专业ID (Primary Key) |
| major\_name | VARCHAR(50) | 专业名称 |
| department\_id | INT | 院系ID |

**Table: courses**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| course\_id | INT | 课程ID (Primary Key) |
| course\_name | VARCHAR(50) | 课程名称 |

**Table: scores**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| score\_id | INT | 成绩ID (Primary Key) |
| student\_id | INT | 学生ID |
| course\_id | INT | 课程ID |
| score | DECIMAL | 成绩 (保留两位小数) |

**Table: rewards\_punishments**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| rp\_id | INT | 奖惩信息ID (Primary Key) |
| student\_id | INT | 学生ID |
| description | VARCHAR(100) | 奖惩描述 |

# 系统功能结构

1. 学生管理模块：负责学生信息的录入、修改、删除和查询。包括学生表（students）和奖惩信息表（rewards\_punishments）的操作。
2. 班级管理模块：负责班级信息的录入、修改、删除和查询。包括班级表（classes）的操作。
3. 院系管理模块：负责院系信息的录入、修改、删除和查询。包括院系表（departments）的操作。
4. 专业管理模块：负责专业信息的录入、修改、删除和查询。包括专业表（majors）的操作。
5. 课程管理模块：负责课程信息的录入、修改、删除和查询。包括课程表（courses）的操作。
6. 成绩管理模块：负责学生成绩的录入、修改、删除和查询。包括成绩表（scores）的操作。
7. 学生信息查询模块：提供学生信息的查询功能，包括学生信息视图（student\_info）的查询。
8. 成绩单查询模块：提供查询指定学生的成绩单的功能，通过存储过程（get\_student\_scores）实现。
9. 触发器模块：包括三个触发器，用于在学生表（students）的插入、删除和更新操作后自动更新班级表（classes）中的学生人数。
10. 参照完整性约束：通过外键约束，保证表之间的关联关系和数据的完整性。

# MySql代码

CREATE DATABASE IF NOT EXISTS STU;

USE STU;

-- 创建院系表

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(50)

);

-- 创建专业表

CREATE TABLE majors (

major\_id INT PRIMARY KEY,

major\_name VARCHAR(50),

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- 创建班级表

CREATE TABLE classes (

class\_id INT PRIMARY KEY,

class\_name VARCHAR(50),

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- 创建课程表

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50)

);

-- 创建学生表

CREATE TABLE students (

student\_id INT PRIMARY KEY,

name VARCHAR(50),

gender ENUM('男', '女'),

class\_id INT,

major\_id INT,

FOREIGN KEY (class\_id) REFERENCES classes(class\_id),

FOREIGN KEY (major\_id) REFERENCES majors(major\_id)

);

-- 创建成绩表

CREATE TABLE scores (

score\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

score DECIMAL(5, 2),

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

);

-- 创建奖惩信息表

CREATE TABLE rewards\_punishments (

rp\_id INT PRIMARY KEY,

student\_id INT,

description VARCHAR(100),

FOREIGN KEY (student\_id) REFERENCES students(student\_id)

);

-- 创建视图，查询学生信息

CREATE VIEW student\_info AS

SELECT s.student\_id, s.name, c.class\_name, m.major\_name, d.department\_name

FROM students s

JOIN classes c ON s.class\_id = c.class\_id

JOIN majors m ON s.major\_id = m.major\_id

JOIN departments d ON m.department\_id = d.department\_id;

-- 创建存储过程，查询指定学生的成绩单

DELIMITER //

CREATE PROCEDURE get\_student\_scores(IN student\_id INT)

BEGIN

SELECT s.name, c.course\_name, sc.score

FROM students s

JOIN scores sc ON s.student\_id = sc.student\_id

JOIN courses c ON sc.course\_id = c.course\_id

WHERE s.student\_id = student\_id;

END //

DELIMITER ;

-- 创建触发器，自动修改班级学生人数

DELIMITER //

CREATE TRIGGER update\_class\_student\_count\_insert

AFTER INSERT ON students

FOR EACH ROW

BEGIN

UPDATE classes

SET student\_count = student\_count + 1

WHERE class\_id = NEW.class\_id;

END //

DELIMITER ;

-- 创建触发器，自动修改班级学生人数

DELIMITER //

CREATE TRIGGER update\_class\_student\_count\_delete

AFTER DELETE ON students

FOR EACH ROW

BEGIN

UPDATE classes

SET student\_count = student\_count - 1

WHERE class\_id = OLD.class\_id;

END //

DELIMITER ;

-- 创建触发器，自动修改班级学生人数

DELIMITER //

CREATE TRIGGER update\_class\_student\_count\_update

AFTER UPDATE ON students

FOR EACH ROW

BEGIN

IF OLD.class\_id != NEW.class\_id THEN

UPDATE classes

SET student\_count = student\_count - 1

WHERE class\_id = OLD.class\_id;

UPDATE classes

SET student\_count = student\_count + 1

WHERE class\_id = NEW.class\_id;

END IF;

END //

DELIMITER ;

-- 建立参照完整性约束

ALTER TABLE students

ADD CONSTRAINT fk\_class\_id

FOREIGN KEY (class\_id) REFERENCES classes(class\_id)

ON DELETE SET NULL

ON UPDATE CASCADE;

ALTER TABLE students

ADD CONSTRAINT fk\_major\_id

FOREIGN KEY (major\_id) REFERENCES majors(major\_id)

ON DELETE SET NULL

ON UPDATE CASCADE;

ALTER TABLE classes

ADD CONSTRAINT fk\_department\_id

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE majors

ADD CONSTRAINT fk\_department\_id

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE scores

ADD CONSTRAINT fk\_student\_id

FOREIGN KEY (student\_id) REFERENCES students(student\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE scores

ADD CONSTRAINT fk\_course\_id

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE rewards\_punishments

ADD CONSTRAINT fk\_student\_id

FOREIGN KEY (student\_id) REFERENCES students(student\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

# 安全性说明

1. 限制访问权限：

-- 创建新用户

CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

-- 授予用户对数据库的访问权限

GRANT SELECT, INSERT, UPDATE, DELETE ON database\_name.\* TO 'username'@'localhost';

-- 刷新权限

FLUSH PRIVILEGES;

2. 数据备份与恢复：

使用MySQL提供的工具进行数据库备份和恢复。例如，使用`mysqldump`命令进行备份：

mysqldump -u username -p database\_name > backup.sql

要恢复数据，可以使用以下命令：

mysql -u username -p database\_name < backup.sql

3. 数据加密：

可以使用MySQL提供的加密函数对敏感数据进行加密存储。例如，可以使用AES加密算法：

-- 创建表时定义加密字段

CREATE TABLE table\_name (

id INT PRIMARY KEY,

sensitive\_data VARBINARY(255)

);

-- 插入加密数据

INSERT INTO table\_name (id, sensitive\_data)

VALUES (1, AES\_ENCRYPT('sensitive information', 'encryption\_key'));

-- 查询并解密数据

SELECT id, AES\_DECRYPT(sensitive\_data, 'encryption\_key') AS sensitive\_data

FROM table\_name;

# 数据库维护计划

1. 定期进行数据库备份，确保数据的安全性和可恢复性。
2. 监控数据库性能，及时调整数据库配置和优化查询语句，以提高系统的响应速度。
3. 定期清理无用数据和日志文件，以减少数据库的存储空间占用。
4. 定期更新数据库软件，安装最新的安全更新和补丁，以确保系统的安全性。
5. 定期进行数据库的性能测试和压力测试，以发现潜在的性能问题并进行优化。