

**大作业报告**

**课程名称 数据库应用技术（SQL）**

**作业名称 学生信息管理系统数据库的设计与实现**

**学 院 专业班级**

**姓 名 学 号**

**2024年12月26日**

目 录

[目 录 III](#_Toc184719869)

[第1章 绪论 1](#_Toc184719870)

[1.1 研究背景及意义 1](#_Toc184719871)

[1.2 研究内容 1](#_Toc184719872)

[1.3 开发技术 1](#_Toc184719873)

[第2章 总体设计 2](#_Toc184719874)

[2.1 总体结构设计 2](#_Toc184719875)

[2.2 数据库结构框架 2](#_Toc184719876)

[第3章 数据库设计 3](#_Toc184719877)

[3.1 需求分析 3](#_Toc184719878)

[3.2 数据库概念结构设计 3](#_Toc184719879)

[3.3 数据库逻辑结构设计 4](#_Toc184719880)

[3.4 数据库设计 4](#_Toc184719881)

[3.5 数据库关系图 4](#_Toc184719882)

[第4章 总结与展望 6](#_Toc184719883)

[4.1 总结 6](#_Toc184719884)

[4.2 展望 6](#_Toc184719885)

# 第1章 绪论

## 1.1 研究背景及意义

1、研究背景

随着信息技术的飞速发展，数据库管理系统已成为现代信息系统的核心工具。特别是在教育领域，学生管理涉及到学籍、成绩、课程、考勤等多方面的数据处理，传统的人工管理方式难以满足现今学校对高效性、准确性和安全性的需求。  
SQL Server 作为一个功能强大的关系型数据库管理系统，具备数据存储、查询优化、安全控制等功能，能够为学校提供一种高效、可靠、低成本的学生管理解决方案。因此，开发基于 SQL Server 的学生管理系统不仅符合时代需求，还能极大提升学校管理水平和工作效率。

2、研究意义

**(1) 提高管理效率：**  
传统学生管理方式依赖纸质档案或电子表格，存在冗余数据多、易出错、信息查询困难等问题。基于 SQL Server 的系统能够通过自动化管理提高效率。

**(2) 数据准确性和安全性：**  
SQL Server 提供了事务管理、数据一致性维护和用户权限控制等功能，能确保学生信息的准确性和安全性，避免数据泄露或丢失的风险。

**(3) 促进教育信息化建设：**  
构建学生管理系统是教育信息化的基础工作，能够为后续的智慧校园、学术数据分析等提供可靠的数据支持。

**(4) 为学校决策提供支持：**  
通过系统化的数据管理，管理者可以实时获取学生数据，进行科学的分析和预测，从而为教育管理决策提供依据。

## 1.2 研究内容

学生管理系统数据库的研究内容主要涵盖如下几个方面：

**(1) 系统需求分析：**

* 确定系统需要管理的主要模块，例如学生信息、课程管理、成绩管理、考勤管理等。
* 分析用户需求，包括管理员、教师和学生的权限划分。

**(2) 数据库设计：**

* 采用 SQL Server 设计系统数据库，包括表的结构设计、主键与外键的定义。
* 确定主要数据表：学生表、课程表、成绩表、考勤表等。
* 制定 ER 图，明确数据表之间的关系。

**(3) 功能模块开发：**  
基于 SQL Server 进行功能开发，包括：

* **学生信息管理模块：** 包括学生的新增、删除、修改和查询功能。
* **成绩管理模块：** 管理学生的考试成绩，支持成绩的统计与分析功能。
* **课程管理模块：** 课程的安排、查询及更新。
* **考勤管理模块：** 记录学生考勤情况并生成报告。

**(4) 数据操作实现：**

* 数据的插入、更新、删除与查询操作。
* 复杂查询：如通过联表查询学生的成绩及平均分、排名等信息。
* 存储过程与触发器：提高系统操作的效率和安全性。

## 1.3 开发技术

1、开发工具

Navicat

2、开发语言

Sql server

# 第2章 总体设计

## 2.1 总体结构设计

总体结构图如图2-1所示。



图2-1 总体结构图

## 2.2 数据库结构框架

数据库框架结构如图2-2所示。

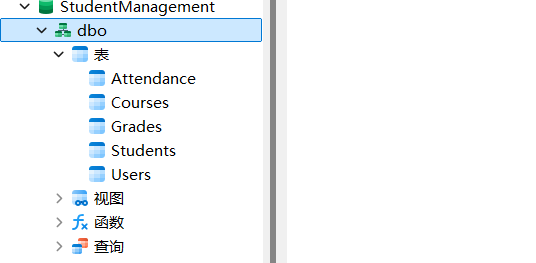


图2-2 数据库框架结构图

# 第3章 数据库设计

## 3.1 需求分析

1. 功能需求分析

功能需求主要包括系统的核心模块及其具体功能，覆盖学生管理的方方面面。

(1) 学生信息管理模块

新增学生信息：录入新生的基本信息，包括姓名、学号、性别、年龄、专业、入学日期等。

删除学生信息：针对毕业、退学或错误录入的学生信息进行删除。

修改学生信息：对学生的基本信息进行更新，如联系方式、班级等。

查询学生信息：根据学号、姓名、班级等条件快速查询学生的详细信息。

(2) 课程管理模块

新增课程：录入课程信息，包括课程编号、名称、授课教师、学分、时间安排等。

删除课程：删除已停开或不再需要的课程。

更新课程信息：修改课程的相关信息，如授课教师、课程时间等。

查询课程信息：根据课程编号、名称或教师姓名查找课程信息。

(3) 成绩管理模块

录入成绩：支持管理员或教师录入学生的考试成绩。

查询成绩：根据学生姓名、学号或课程查看考试成绩。

统计与分析成绩：生成成绩分布、平均分、最高分、最低分等统计结果，用于分析学生的学习情况。

(4) 考勤管理模块

记录考勤：记录学生的日常考勤情况，包括迟到、缺席等状态。

查询考勤：按时间段或学生查询考勤记录。

生成考勤报告：为班主任或学校管理者生成特定学生或班级的考勤分析报告。

(5) 报表与统计模块

生成学生综合报表：结合学生的基本信息、成绩、考勤等生成综合报告。

数据分析功能：对学生的成绩、课程、考勤数据进行图形化展示和分析。

导出功能：支持将数据导出为 Excel、PDF 或其他格式。

(6) 用户管理模块

用户权限管理：

管理员：管理所有功能模块。

教师：管理与其相关的课程、成绩录入和考勤记录。

学生：仅能查看自己的信息、成绩和考勤情况。

## 3.2 数据库概念结构设计

本系统的整体E-R如图



图3-1系统整体E-R图

## 3.3 数据库逻辑结构设计

(1) 学生信息表（Students）

学生信息表用于存储学生的基本信息，表结构如表 4-1 所示。

表 4-1 学生信息表（Students）

| 字段名称 | 类型 | 字符长度 | 主键 | 备注 | 是否为空 |
| --- | --- | --- | --- | --- | --- |
| StudentID | Int | 4 | 是 | 学生编号 | 否 |
| Name | Nvarchar | 50 |  | 学生姓名 | 否 |
| Gender | Char | 1 |  | 性别（M/F） | 否 |
| DateOfBirth | Date |  |  | 出生日期 | 否 |
| Major | Nvarchar | 50 |  | 专业 | 否 |
| EnrollmentDate | Date |  |  | 入学日期 | 否 |
| Contact | Nvarchar | 20 |  | 联系方式 | 是 |

(2) 课程信息表（Courses）

课程信息表存储课程的详细信息，表结构如表 4-2 所示。

表 4-2 课程信息表（Courses）

| 字段名称 | 类型 | 字符长度 | 主键 | 备注 | 是否为空 |
| --- | --- | --- | --- | --- | --- |
| CourseID | Int | 4 | 是 | 课程编号 | 否 |
| CourseName | Nvarchar | 100 |  | 课程名称 | 否 |
| Credits | Int |  |  | 学分 | 否 |
| Instructor | Nvarchar | 50 |  | 授课教师 | 否 |

(3) 成绩信息表（Grades）

成绩信息表用于记录学生的考试成绩，表结构如表 4-3 所示。

表 4-3 成绩信息表（Grades）

| 字段名称 | 类型 | 字符长度 | 主键 | 备注 | 是否为空 |
| --- | --- | --- | --- | --- | --- |
| GradeID | Int | 4 | 是 | 成绩编号 | 否 |
| StudentID | Int | 4 |  | 学生编号 | 否 |
| CourseID | Int | 4 |  | 课程编号 | 否 |
| Score | Decimal | 5,2 |  | 分数 | 否 |
| GradeDate | Date |  |  | 成绩录入日期 | 否 |

(4) 考勤信息表（Attendance）

考勤信息表用于存储学生的考勤记录，表结构如表 4-4 所示。

表 4-4 考勤信息表（Attendance）

| 字段名称 | 类型 | 字符长度 | 主键 | 备注 | 是否为空 |
| --- | --- | --- | --- | --- | --- |
| AttendanceID | Int | 4 | 是 | 考勤编号 | 否 |
| StudentID | Int | 4 |  | 学生编号 | 否 |
| AttendanceDate | Date |  |  | 考勤日期 | 否 |
| Status | Nvarchar | 20 |  | 考勤状态 | 否 |
| Remarks | Nvarchar | 200 |  | 备注 | 是 |

(5) 用户信息表（Users）

用户信息表用于存储系统用户的登录信息，表结构如表 4-5 所示。

表 4-5 用户信息表（Users）

| 字段名称 | 类型 | 字符长度 | 主键 | 备注 | 是否为空 |
| --- | --- | --- | --- | --- | --- |
| UserID | Int | 4 | 是 | 用户编号 | 否 |
| Username | Nvarchar | 50 |  | 用户名 | 否 |
| PasswordHash | Nvarchar | 255 |  | 密码（加密） | 否 |
| Role | Nvarchar | 20 |  | 用户角色 | 否 |

## 3.4 数据库设计

CREATE DATABASE StudentManagement;

GO

USE StudentManagement;

GO

CREATE TABLE Students (

StudentID INT PRIMARY KEY IDENTITY(1,1), -- 学号（自动增长）

Name NVARCHAR(50) NOT NULL, -- 学生姓名

Gender CHAR(1) CHECK (Gender IN ('M', 'F')), -- 性别（M: 男, F: 女）

DateOfBirth DATE NOT NULL, -- 出生日期

Major NVARCHAR(50) NOT NULL, -- 专业

EnrollmentDate DATE NOT NULL, -- 入学日期

Contact NVARCHAR(20) -- 联系方式

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY IDENTITY(1,1), -- 课程编号（自动增长）

CourseName NVARCHAR(100) NOT NULL, -- 课程名称

Credits INT CHECK (Credits > 0), -- 学分

Instructor NVARCHAR(50) NOT NULL -- 授课教师

);

CREATE TABLE Grades (

GradeID INT PRIMARY KEY IDENTITY(1,1), -- 成绩编号

StudentID INT NOT NULL, -- 学生编号（外键）

CourseID INT NOT NULL, -- 课程编号（外键）

Score DECIMAL(5,2) CHECK (Score >= 0 AND Score <= 100), -- 分数

GradeDate DATE NOT NULL, -- 成绩录入日期

CONSTRAINT FK\_Grades\_Student FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

CONSTRAINT FK\_Grades\_Course FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

);

CREATE TABLE Attendance (

AttendanceID INT PRIMARY KEY IDENTITY(1,1), -- 考勤编号

StudentID INT NOT NULL, -- 学生编号（外键）

AttendanceDate DATE NOT NULL, -- 考勤日期

Status NVARCHAR(20) CHECK (Status IN ('Present', 'Absent', 'Late')), -- 考勤状态

Remarks NVARCHAR(200), -- 备注

CONSTRAINT FK\_Attendance\_Student FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

);

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY(1,1), -- 用户编号

Username NVARCHAR(50) NOT NULL UNIQUE, -- 用户名

PasswordHash NVARCHAR(255) NOT NULL, -- 加密的密码

Role NVARCHAR(20) CHECK (Role IN ('Admin', 'Teacher', 'Student')) -- 用户角色

);

INSERT INTO Students (Name, Gender, DateOfBirth, Major, EnrollmentDate, Contact)

VALUES

('张三', 'M', '2002-05-12', '计算机科学', '2020-09-01', '123456789'),

('李四', 'F', '2003-08-21', '信息管理', '2021-09-01', '987654321');

INSERT INTO Courses (CourseName, Credits, Instructor)

VALUES

('数据库系统', 3, '王老师'),

('操作系统', 4, '李老师');

INSERT INTO Grades (StudentID, CourseID, Score, GradeDate)

VALUES

(1, 1, 85.5, '2023-06-15'),

(2, 2, 90.0, '2023-06-15');

INSERT INTO Attendance (StudentID, AttendanceDate, Status, Remarks)

VALUES

(1, '2023-12-01', 'Present', '按时到达'),

(2, '2023-12-01', 'Late', '迟到5分钟');

INSERT INTO Users (Username, PasswordHash, Role)

VALUES

('admin', 'hashed\_password\_1', 'Admin'),

('teacher1', 'hashed\_password\_2', 'Teacher'),

('student1', 'hashed\_password\_3', 'Student');

-- 用户视图

CREATE VIEW StudentGrades AS

SELECT

S.StudentID,

S.Name AS StudentName,

C.CourseName,

G.Score,

G.GradeDate

FROM

Grades G

INNER JOIN Students S ON G.StudentID = S.StudentID

INNER JOIN Courses C ON G.CourseID = C.CourseID;

-- 存储过程 添加学生信息

CREATE PROCEDURE AddStudent

@Name NVARCHAR(50),

@Gender CHAR(1),

@DateOfBirth DATE,

@Major NVARCHAR(50),

@EnrollmentDate DATE,

@Contact NVARCHAR(20)

AS

BEGIN

INSERT INTO Students (Name, Gender, DateOfBirth, Major, EnrollmentDate, Contact)

VALUES (@Name, @Gender, @DateOfBirth, @Major, @EnrollmentDate, @Contact);

END;

## 3.5 数据库关系图

数据库关系图如图3-2所示。

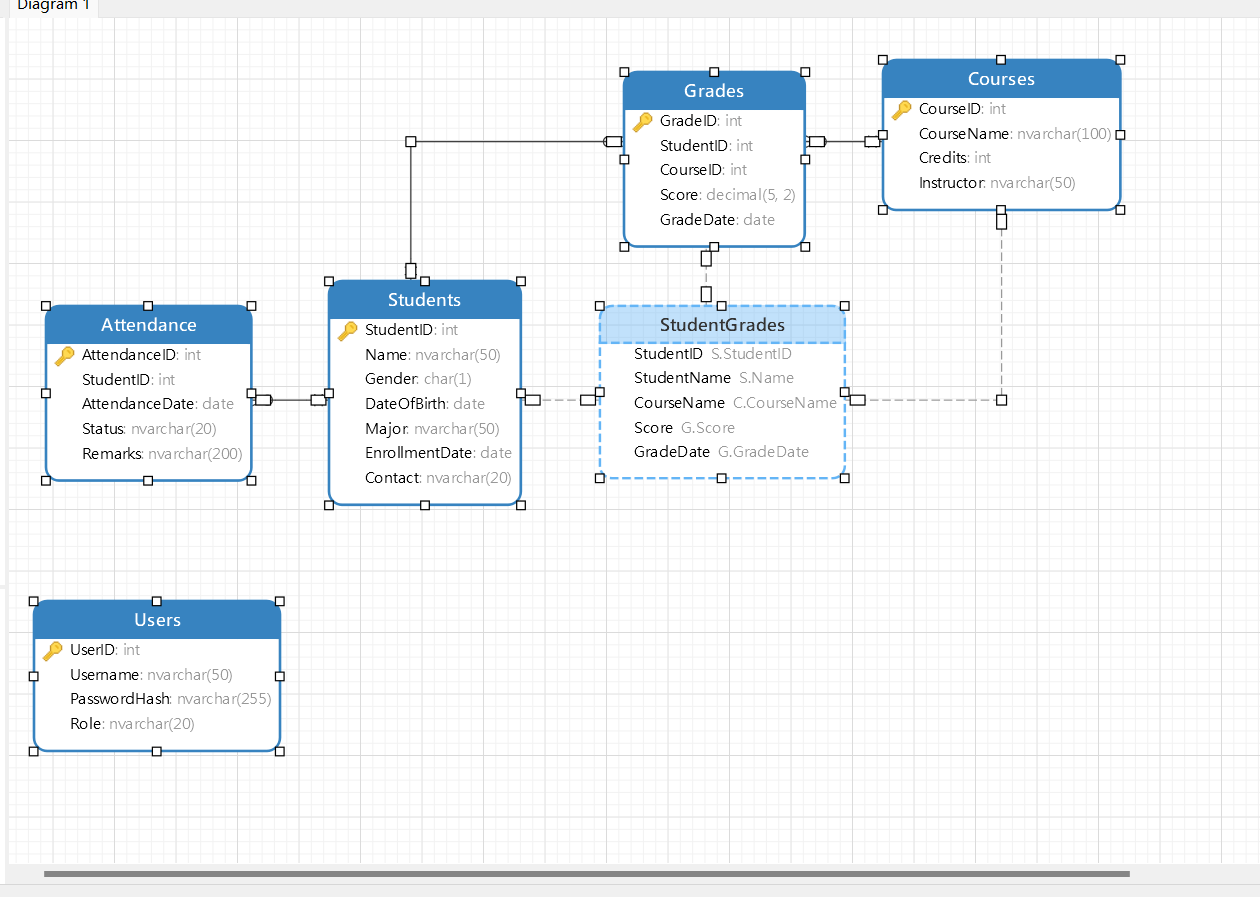


图3-2 系统数据库关系图

# 第4章 总结与展望

## 4.1 总结

本次学生管理系统数据库设计与实现项目，以 SQL Server 为基础，从系统需求分析到功能模块开发，完成了一个完整的数据库管理系统。以下是本项目的主要总结：

1. **系统设计全面**：  
   系统涵盖学生信息管理、课程管理、成绩管理、考勤管理及用户权限管理等多个模块，满足了学生管理的核心需求。通过详细的需求分析，我们明确了每个模块的功能，实现了从数据录入到查询分析的全流程覆盖。
2. **数据库结构合理**：  
   数据库设计遵循关系数据库设计原则，包括表的结构设计、主键与外键的定义，以及表间关系的明确。通过 E-R 图及数据库关系图，清晰展示了系统的逻辑结构和数据流动。表与表之间的主外键关联保证了数据的完整性和一致性。
3. **功能实现高效**：  
   系统开发过程中，应用了复杂查询、视图、存储过程等多种 SQL 技术。存储过程提升了数据操作的效率和安全性，而视图的引入简化了数据查询过程，方便用户获取所需信息。
4. **技术实践充分**：  
   本项目基于 SQL Server 的强大功能，结合 Navicat 工具及 SQL 开发语言，从数据表设计到最终实现了功能模块的开发，包括数据增删改查、统计分析等常用操作，具备较高的实际应用价值。

综上，本次学生管理系统数据库设计与实现，达到了预期目标，为教育信息化建设提供了一个高效、可靠的解决方案，同时也提升了我们在数据库设计与开发方面的实际应用能力。

## 4.2 展望

尽管本次项目取得了较好的成果，但仍存在改进和扩展的空间。以下是未来发展的展望：

1. **系统功能扩展**：  
   当前系统功能模块主要围绕学生管理展开，未来可以进一步扩展功能。例如：
   * 添加在线缴费管理模块，实现学费收缴及记录。
   * 引入选课系统模块，支持学生在线选课与退课。
   * 提供家长查看功能，使家长能够通过专属权限了解学生的学习与考勤情况。
2. **数据分析优化**：
   * 在现有基础上，增加更全面的数据分析功能，如学生成长趋势分析、课程受欢迎程度统计等，为学校管理者提供更有价值的决策支持。
   * 结合可视化工具（如 Power BI）展示分析结果，实现数据图表化，增强用户体验。
3. **用户体验优化**：
   * 提升用户界面设计，通过引入前端框架（如 Vue.js 或 React），实现更加直观和便捷的操作。
   * 开发移动端应用，使用户可以随时随地管理和查看学生信息。
4. **与其他系统的集成**：
   * 将学生管理系统与学校其他管理系统（如人事管理、教务系统）集成，实现数据共享和统一管理。
   * 支持与教育部门的接口对接，实现政策文件与数据的快速传递。

总之，未来的学生管理系统可以向着智能化、可扩展化和高效化的方向不断发展，为教育领域的信息化建设提供更全面的支持。