目 录

一. 系统需求分析

1. 系统描述

教务管理是大学的主要日常管理工作之一，涉及到校、系、师、生的诸多方面，随着教学体制的不断改革,尤其是学分制、选课制的展开和深入，教务日常管理工作日趋繁重、复杂。如何把教务工作信息化、模块化、便捷化是现代高校发展的重点，因此如今一个完整统一、技术先进、高效稳定、安全可靠的教学信息管理系统已经成为大学教务不可或缺的一部分。

本教务系统支持三类用户的使用：学生、教师及教务人员。在教务系统中，所有用户都可以进行对院系、空闲教室、开课信息等基本信息进行查询。在此之上，学生可以进行选课、查询成绩与课表等学生操作功能，教师可以查看自己开设的课程、为自己班上学生评定成绩，管理员则可以在数据库执行添加、删除、修改等后台管理功能。

教务管理系统跨越了时间和空间的限制，能够使教务人员与高校师生在线进行教务操作，减少相关环节工作量，提高工作效率。

2. 数据存储需求

教务系统数据库需要存储如下信息。有关注册用户的信息涉及用户 ID、用户名、密码与用户角色信息。系统中同时还存储诸如当前学期、是否在选课时间内等状态信息。

教师与学生从属于一个院系，一个院系下有多名教师与学生，并可开设多门课程。同时院系可以按年级建立多个学生培养方案。每个培养方案与多门课程相关联，课程之间有着先修课程的联系，每门课程从属于一个院系。课程可以实例化为多门开课，每一门开课可看作是一个课程的实例，一门开课可以由多名教师执教，多名学生注册选课，同时一门开课可以有多个开课安排。开课安排包含时间槽与教室信息，时间槽是一段连续的课时，包含星期，起始小节与结束小节。每个小节对应于一个具体的时间安排。

3. 系统常做的查询与更新

经常做的查询，或许对创建索引有影响的：

- 查询全校所有院系信息；
- 查询当前学期空闲教室；
- 查询各院系培养方案；
- 学生查询自己所选课程与成绩；
- 教师查询自己教授课程的学生信息。

根据经常做的查询，需要创建有关视图的：

- 查询开课信息，包含课程信息，开课信息，授课教师与开课的时间地点安排。

关于更新

- 院系，时间槽，时间表较少发生更新；
- 培养方案，开课，开课安排，学生，选课，系统状态会以学期为周期进行更新；
- 教师，课程信息会不定期发生更新。

4. 应用程序功能

前台的主要功能如下：

1) 对所有用户开放：
    a. 课程查询：根据指定条件查询开设的课程；
    b. 教室查询：根据指定条件查询当前学期空闲的教室，并可针对某教室查询在此处所开设课程；
    c. 院系查询：查询全校院系信息；
    d. 个人信息：浏览或更新个人信息；
    e. 培养方案查询：查询全校的培养方案，对学生会默认显示其自身的培养方案。
2) 对学生开放：
    a. 我的成绩：查询该学生不同学期的成绩与统计信息；

b. 我的课表：查询该学生不同学期所选课程，并形成课表；

c. 学生选课：在指定时段从当学期开设的课程中进行选课或退课。

3) 对教师开放：

a. 学生查询：查询自己教授班级的学生信息；

b. 学生评分：维护选修了自己课程的学生得分信息；

后台的主要功能如下：

（下述的管理，均指的是对相关信息进行查询，修改，增添，删除操作）

1) 系统设置：

a. 系统状态设置：设置当前学期、选课学期、是否在选课时段信息，并可在该界面对当前学期学生成绩进行结算并给出绩点；

b. 账户管理：管理学生、教师与管理员的系统登陆账号。

2) 人员管理

a. 教师信息管理：管理教师信息；

b. 学生信息管理：管理学生信息；

c. 院系信息管理：管理院系信息。

3) 课程管理：

a. 课程信息管理：管理课程信息；

b. 开课信息管理：管理开课信息；

c. 开课安排管理：管理开课的时间、地点安排；

d. 教室信息管理：管理教室信息；

e. 选课信息管理：管理学生选课信息；

f. 授课信息管理：管理教师授课信息；

g. 先修课程管理：管理课程的先修课程条件，不满足先修条件将无法选课；

h. 培养方案管理：管理各院系各级学生的培养方案。

二. 数据库概念设计

1. 确定实体和属性

分析教务管理系统的系统需求，将系统中设计的人、物进行抽象，得到了系统的实体如下：

1) 院系信息实体集。属性包括：院系名称、办公室地址、办公室联系电话；
2) 教室信息实体集。属性包括：教学楼名称、房间号、教室容量；
3) 课程信息实体集。属性包括：课程编号、课程标题、课程类型、开课院系、学分；
4) 教师信息实体集。属性包括：教师工号、教师姓名、所属院系、性别、雇用年份、月薪、身份证号码、出生日期、联系电话、地址、电子邮件；
5) 学生信息实体集。属性包括：学生学号、学生姓名、所属院系、性别、入学年份、身份证号码、出生日期、联系电话、地址、电子邮件；
6) 开课信息实体集。属性包括：开课编号、课程编号、学期、学年、课程容量；
7) 小节时间实体集。属性包括：小节编号、开始时间、结束时间；
8) 上课时间实体集。属性包括：上课时间编号、星期、起始小节编号、结束小节编号；
9) 用户信息实体集。属性包括：用户 ID、用户名、密码哈希、SALT、用户类型；
10) 系统状态实体集。属性包括：状态字段、状态值。

系统的联系集如下：

1) 选课信息联系集。属性包括：学生学号、开课编号、平时成绩、期末成绩、总成绩、绩点、成绩等级、状态；
2) 授课信息联系集。属性包括：教师工号、开课编号；
3) 开课安排联系集。属性包括：开课编号、上课时间编号、教学楼名称、

房间号、类型；

4) 先修课程联系集。属性包括：课程编号、先修课程编号；

5) 培养方案联系集。属性包括：院系名称、课程编号、学级、推荐学期。

2. E-R 图

系统 E-R 图如图 2-1 所示：

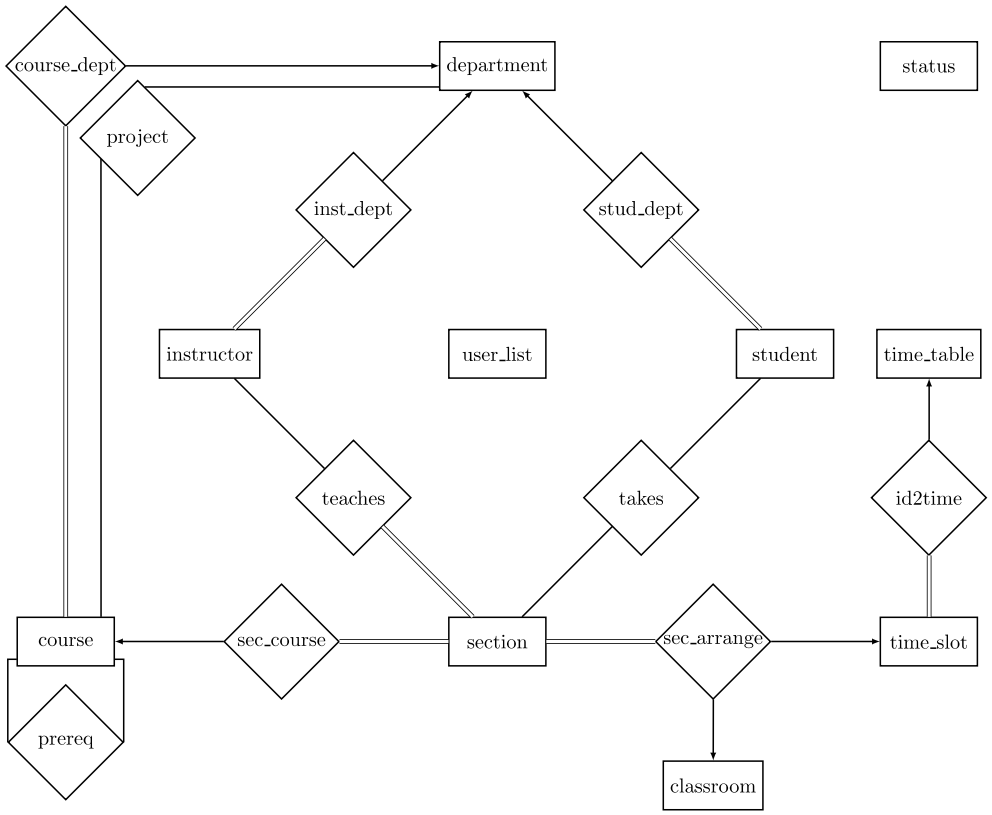


图 2-1 E-R 图

三. 数据库逻辑结构设计

1. 关系模式设计

根据概念结构设计得到的 E-R 图和转换规则，得到如下关系模式（主外键此处未标，在基本表设计处体现）：

院系信息表(院系名称，办公室地址，办公室联系电话)

教室信息表(教学楼名称，房间号，教室容量)

课程信息表(课程编号，课程标题，课程类型，开课院系，学分)

教师信息表(教师工号，教师姓名，所属院系，性别，雇用年份，月薪，身份证号码，出生日期，联系电话，地址，电子邮件)

学生信息表(学生学号，学生姓名，所属院系，性别，入学年份，身份证号码，出生日期，联系电话，地址，电子邮件)

开课信息表(开课编号，课程编号，学期，学年，课程容量)

小节时间表(小节编号，开始时间，结束时间)

上课时间表(上课时间编号，星期，起始小节编号，结束小节编号)

用户信息表(用户 ID，用户名，密码哈希，SALT，用户类型)

系统状态表(状态字段，状态值)

选课信息表(学生学号，开课编号，平时成绩，期末成绩，总成绩，绩点，成绩等级，状态)

授课信息表(教师工号，开课编号)

开课安排表(开课编号，上课时间编号，教学楼名称，房间号，类型)

先修课程表(课程编号，先修课程编号)

培养方案表(院系名称，课程编号，学级，推荐学期)

2. 基本表设计

基本表设计如表 3-1~3-15 所示。

表 3-1：院系信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
dept_name	VARCHAR(50)	否			主键	院系名称
office	VARCHAR(50)	是				办公室地址
office_phone	VARCHAR(20)	是				办公室电话

表 3-2：教室信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
building	VARCHAR(30)	否			主键	教学楼名称
room_number	VARCHAR(10)	否			主键	房间号
capacity	NUMERIC(4, 0)	否				教室容量

表 3-3：课程信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
course_id	VARCHAR(20)	否			主键	课程编号
title	VARCHAR(100)	否				课程标题
type	VARCHAR(20)	否	见 DDL			课程类型
dept_name	VARCHAR(50)	否			外键	开课院系
credits	NUMERIC(2, 1)	否	见 DDL			学分

表 3-4：教师信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
inst_id	VARCHAR(20)	否			主键	教师工号

name	VARCHAR(50)	否				教师姓名
dept_name	VARCHAR(50)	否			外键	所属院系
sexual	VARCHAR(1)	否	见 DDL			性别
year	INTEGER	否	见 DDL			雇用年份
salary	NUMERIC(8, 2)	是	见 DDL			月薪
identity_number	VARCHAR(20)	是				身份证号码
birthday	DATE	是				出生日期
phone_number	VARCHAR(20)	是				联系电话
address	VARCHAR(100)	是				地址
email	VARCHAR(50)	是				电子邮件

表 3-5：学生信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
stu_id	VARCHAR(20)	否			主键	学生学号
name	VARCHAR(50)	否				教师姓名
dept_name	VARCHAR(50)	否			外键	所属院系
sexual	VARCHAR(1)	否	见 DDL			性别
year	INTEGER	否	见 DDL			入学年份
identity_number	VARCHAR(20)	是				身份证号码
birthday	DATE	是				出生日期
phone_number	VARCHAR(20)	是				联系电话
address	VARCHAR(100)	是				地址
email	VARCHAR(50)	是				电子邮件

表 3-6：开课信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
sec_id	INTEGER	否			主键	开课编号
course_id	VARCHAR(20)	否			外键	课程编号
semester	VARCHAR(10)	否	见 DDL			学期
year	NUMERIC(4, 0)	否	见 DDL			学年
capacity	INTEGER	否				课程容量

表 3-7：小节时间表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
time_id	INTEGER	否			主键	小节编号
start_time	TIME	否				开始时间
end_time	TIME	否				结束时间

表 3-8：上课时间表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
time_slot_id	INTEGER	否			主键	上课时间编号
day	VARCHAR(3)	否	见 DDL			星期
start_time_id	INTEGER	否			外键	起始小节编号
end_time_id	INTEGER	否			外键	结束小节编号

表 3-9：用户信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
user_id	VARCHAR(20)	否			主键	用户 ID

user_name	VARCHAR(50)	否				用户名
pwd_hash	BINARY(64)	否				密码哈希
salt	UNIQUEIDENTIFIER	否				SALT
role	VARCHAR(20)	否	见 DDL			用户类型

表 3-10：系统状态表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
keyword	VARCHAR(50)	否			主键	状态字段
value	VARCHAR(50)	否				状态值

表 3-11：选课信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
stu_id	VARCHAR(20)	否			主键/外键	学生学号
sec_id	INTEGER	否			主键/外键	开课编号
regular_grade	NUMERIC(3, 0)	是	见 DDL			平时成绩
final_grade	NUMERIC(3, 0)	是	见 DDL			期末成绩
total_grade	NUMERIC(4, 1)	是	见 DDL			总成绩
grade_point	NUMERIC(2, 1)	是	见 DDL			绩点
mark	VARCHAR(2)	是	见 DDL			成绩等级
status	VARCHAR(3)	否	见 DDL			状态

表 3-12：授课信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释

inst_id	VARCHAR(20)	否			主键/外键	教师工号
sec_id	INTEGER	否			主键/外键	开课编号

表 3-13：开课安排表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
sec_id	INTEGER	否			主键/外键	开课编号
time_slot_id	INTEGER	否			主键/外键	上课时间编号
building	VARCHAR(30)	否			主键/外键	教学楼名称
room_number	VARCHAR(10)	否			主键/外键	房间号
slot_type	VARCHAR(20)	否	见 DDL	见 DDL		类型

表 3-14：先修课程表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
course_id	VARCHAR(20)	否			主键/外键	课程编号
prereq_id	VARCHAR(20)	否			主键/外键	先修课程编号

表 3-15：培养方案表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
dept_name	VARCHAR(50)	否			主键/外键	院系名称
course_id	VARCHAR(20)	否			主键/外键	课程编号
year	INTEGER	否			主键	学级
term	INTEGER	否				推荐学期

四. 数据库物理设计和实施

1. 数据库的创建

使用 MS SQL Server 建立教务管理系统的数据库，数据库基本信息如下：

表 4-1：教务管理系统的数据库参数表

选项		参数
数据库名称		University
数据文件	逻辑文件名	University
	物理文件名	D:\Program\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\University.mdf
	初始容量	8MB
	最大容量	Unlimited
	增长量	64MB
日志文件	逻辑文件名	University_log
	物理文件名	D:\Program\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\University_log.ldf
	初始容量	8MB
	最大容量	Unlimited
	增长量	64MB

2. 创建基本表

```
CREATE TABLE department (
   dept_name VARCHAR(50) NOT NULL,
```

```
   office VARCHAR(50),
   office_phone VARCHAR(20),
   PRIMARY KEY (dept_name)
);

CREATE TABLE classroom (
   building VARCHAR(30) NOT NULL,
   room_number VARCHAR(10) NOT NULL,
   capacity NUMERIC(4, 0) NOT NULL,
   PRIMARY KEY (building, room_number)
);

CREATE TABLE course (
   course_id VARCHAR(20) NOT NULL,
   title VARCHAR(100) NOT NULL,
   type VARCHAR(20) NOT NULL
      CHECK(type IN ('Basic', 'Optional Core', 'Obligatory
Core', 'Obligatory Public', 'Optional Public')),
   dept_name VARCHAR(50) NOT NULL,
   credits NUMERIC(2, 1) NOT NULL
      CHECK(credits > 0),
   PRIMARY KEY (course_id),
   FOREIGN KEY (dept_name) REFERENCES department
      ON DELETE CASCADE
);

CREATE TABLE instructor (
   inst_id VARCHAR(20) NOT NULL,
   name VARCHAR(50) NOT NULL,
```

```
   dept_name VARCHAR(50) NOT NULL,
   sexual VARCHAR(1) NOT NULL
      CHECK (sexual in ('M', 'F')),
   year INTEGER NOT NULL
      CHECK (year BETWEEN 1700 AND 2100),
   salary NUMERIC(8, 2)
      CHECK (salary > 0),
   identity_number VARCHAR(20),
   birthday DATE,
   phone_number VARCHAR(20),
   address VARCHAR(100),
   email VARCHAR(50),
   PRIMARY KEY (inst_id),
   FOREIGN KEY (dept_name) REFERENCES department
);

CREATE TABLE time_table (
   time_id INTEGER NOT NULL,
   start_time TIME NOT NULL,
   end_time TIME NOT NULL,
   PRIMARY KEY (time_id)
)

CREATE TABLE timeslot (
   time_slot_id INTEGER NOT NULL IDENTITY(1, 1),
   day VARCHAR(3) NOT NULL
      CHECK(day IN ('MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT',
'SUN')),
   start_time_id INTEGER NOT NULL,
```

```
   end_time_id INTEGER NOT NULL,
   PRIMARY KEY (time_slot_id),
   FOREIGN KEY (start_time_id) REFERENCES time_table,
   FOREIGN KEY (end_time_id) REFERENCES time_table
);

CREATE TABLE section (
   sec_id INTEGER NOT NULL IDENTITY(1, 1),
   course_id VARCHAR(20) NOT NULL,
   semester VARCHAR(10) NOT NULL
      check (semester IN ('Spring', 'Summer', 'Autumn',
'Winter')),
   year NUMERIC(4, 0) NOT NULL
      CHECK (year BETWEEN 1970 AND 2100),
   capacity INTEGER NOT NULL,
   PRIMARY KEY (sec_id),
   FOREIGN KEY (course_id) REFERENCES course
      ON DELETE CASCADE
);

CREATE TABLE sec_arrange (
   sec_id INTEGER NOT NULL,
   time_slot_id INTEGER NOT NULL,
   building VARCHAR(30) NOT NULL,
   room_number VARCHAR(10) NOT NULL,
   slot_type VARCHAR(20) DEFAULT 'All weeks' NOT NULL
      CHECK (slot_type IN ('Single week', 'Double week', 'All
weeks', 'Former term', 'Latter term')),
   PRIMARY KEY (sec_id, time_slot_id, building,
```

```
room_number),
   FOREIGN KEY (sec_id) REFERENCES section
      ON DELETE CASCADE,
   FOREIGN KEY (time_slot_id) REFERENCES timeslot,
   FOREIGN KEY (building, room_number) REFERENCES classroom
);

CREATE TABLE teaches (
   inst_id VARCHAR(20) NOT NULL,
   sec_id INTEGER NOT NULL,
   PRIMARY KEY (inst_id, sec_id),
   FOREIGN KEY (sec_id) REFERENCES section
      ON DELETE CASCADE,
   FOREIGN KEY (inst_id) REFERENCES instructor
      ON DELETE CASCADE
);

CREATE TABLE student (
   stu_id VARCHAR(20) NOT NULL,
   name VARCHAR(50) NOT NULL,
   dept_name VARCHAR(50) NOT NULL,
   sexual VARCHAR(1) NOT NULL
      CHECK (sexual in ('M', 'F')),
   year INTEGER NOT NULL
      CHECK (year BETWEEN 1700 AND 2100),
   identity_number VARCHAR(20),
   birthday DATE,
   phone_number VARCHAR(20),
   address VARCHAR(100),
```

```
   email VARCHAR(50),
   PRIMARY KEY (stu_id),
   FOREIGN KEY (dept_name) REFERENCES department
      ON DELETE CASCADE
);

CREATE TABLE takes (
   stu_id VARCHAR(20) NOT NULL,
   sec_id INTEGER NOT NULL,
   regular_grade NUMERIC(3, 0)
      CHECK (regular_grade >= 0 AND regular_grade <= 100),
   final_grade NUMERIC(3, 0)
      CHECK (final_grade >= 0 AND final_grade <= 100),
   total_grade NUMERIC(4, 1)
      CHECK (total_grade >= 0.0 AND total_grade <= 100.0),
   grade_point NUMERIC(2, 1)
      CHECK (grade_point >= 0.0 AND grade_point <= 4.0),
   mark VARCHAR(2)
      CHECK (mark IN ('A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+',
'C', 'F')),
   status VARCHAR(3) DEFAULT 'INP' NOT NULL
      CHECK (status IN ('FIN', 'INP')),
   PRIMARY KEY (stu_id, sec_id),
   FOREIGN KEY (sec_id) REFERENCES section
      ON DELETE CASCADE,
   FOREIGN KEY (stu_id) REFERENCES student
);

CREATE TABLE prereq (
```

```
   course_id VARCHAR(20) NOT NULL,
   prereq_id VARCHAR(20) NOT NULL,
   PRIMARY KEY (course_id, prereq_id),
   FOREIGN KEY (course_id) REFERENCES course
      ON DELETE CASCADE,
   FOREIGN KEY (prereq_id) REFERENCES course
);

CREATE TABLE project (
   dept_name VARCHAR(50) NOT NULL,
   course_id VARCHAR(20) NOT NULL,
   year INTEGER NOT NULL,
   term INTEGER NOT NULL
      CHECK (term >= 1 AND term <= 16),
   PRIMARY KEY (dept_name, course_id, year),
   FOREIGN KEY (dept_name) REFERENCES department
      ON DELETE CASCADE,
   FOREIGN KEY (course_id) REFERENCES course
);

CREATE TABLE user_list (
   user_id VARCHAR(20) NOT NULL,
   user_name VARCHAR(50) NOT NULL,
   pwd_hash BINARY(64) NOT NULL,
   salt UNIQUEIDENTIFIER NOT NULL,
   role VARCHAR(20) NOT NULL
      CHECK    (role    IN    ('student',    'instructor',
'administrator')),
   PRIMARY KEY (user_id)
```

```
);

CREATE TABLE status (
   keyword VARCHAR(50) NOT NULL,
   value VARCHAR(50) NOT NULL,
   PRIMARY KEY (keyword)
);
```

3. 触发器设计

在 student 和 instructor 中添加新的学生或教师时，自动在 user_list 表中以其工号或学号为账号，123456 为默认密码创建登陆账户。

```
CREATE TRIGGER generateStuUser
ON student AFTER INSERT AS
BEGIN
   DECLARE @stu_id VARCHAR(20), @name VARCHAR(50);
   SELECT @stu_id = stu_id, @name = name
   FROM inserted
   EXEC add_user @stu_id, @p_user_name = @name, @p_pwd =
'123456', @role = 'student';
END

CREATE TRIGGER generateInstUser
ON instructor AFTER INSERT AS
BEGIN
   DECLARE @inst_id VARCHAR(20), @name VARCHAR(50);
   SELECT @inst_id = inst_id, @name = name
   FROM inserted
   EXEC add_user @inst_id, @p_user_name = @name, @p_pwd =
```

```
'123456', @role = 'instructor';
END
```

在 instructor 或 student 表中姓名发生更改时，自动更新 user_list 表对应账户的用户名。

```
CREATE TRIGGER changeStuName
ON student AFTER UPDATE AS
BEGIN
   IF UPDATE(name)
      UPDATE user_list
      SET user_name = (SELECT name FROM inserted)
      WHERE user_id = (SELECT stu_id FROM inserted)
END

CREATE TRIGGER changeInstName
ON instructor AFTER UPDATE AS
BEGIN
   IF UPDATE(name)
      UPDATE user_list
      SET user_name = (SELECT name FROM inserted)
      WHERE user_id = (SELECT inst_id FROM inserted)
END
```

当学生注册一门课时，如该课人数已满，则拒绝此次操作。

```
CREATE TRIGGER sectionCapacityFull
ON takes AFTER INSERT AS
BEGIN
   DECLARE @currentCount INT
   SELECT @currentCount = COUNT(stu_id)
```

```
   FROM takes
   WHERE sec_id = (SELECT sec_id FROM inserted)
   IF @currentCount > (SELECT capacity FROM section WHERE
sec_id = (SELECT sec_id FROM inserted))
   BEGIN
      ROLLBACK TRANSACTION
      RAISERROR('Section capacity full!', -1, -1);
   END
END
```

当学生试图注册/取消注册一门课程时，如不在选课时间内，则拒绝此次操作。

```
CREATE TRIGGER checkRegisterTime
ON takes AFTER INSERT, DELETE AS
BEGIN
   IF (SELECT value FROM status WHERE keyword = 'Open
registration') = 'False'
   BEGIN
      ROLLBACK TRANSACTION;
      RAISERROR('Not in registration time!', -1, -1);
   END
END
```

当更新 takes 表中状态字段为完结的项的除状态以外的域时，拒绝此次更新。该触发器拒绝修改一项已经完结的选课记录的成绩信息。

```
CREATE TRIGGER protectScore
ON takes AFTER UPDATE AS
BEGIN
   IF (NOT UPDATE(status)) AND ((SELECT status FROM inserted)
= 'FIN')
```

```
    BEGIN
        ROLLBACK TRANSACTION
        RAISERROR('Cannot modify ended section!', -1, -1)
    END
END
```

当学生注册一门课时，如不满足该课的先修条件，则拒绝此次操作。

```
CREATE TRIGGER sectionPrerequisite
ON TAKES AFTER INSERT AS
BEGIN
    DECLARE @sec_id INT, @stu_id VARCHAR(20)
    SELECT @sec_id = sec_id, @stu_id = stu_id FROM inserted
    IF EXISTS (
        (SELECT course_id FROM prereq WHERE course_id = (SELECT
course_id FROM section WHERE sec_id = @sec_id))
        EXCEPT
        (SELECT course_id FROM section WHERE sec_id IN (SELECT
sec_id FROM takes WHERE stu_id = @stu_id)))
    BEGIN
        ROLLBACK TRANSACTION;
        RAISERROR('Prerequisite not satisfied!', -1, -1);
    END
END
```

4. 存储过程设计

新增用户，四个参数依次为账号、用户名、密码、角色，返回 0 代表执行成功，否则为执行失败。

```
CREATE PROCEDURE add_user
    @p_user_id VARCHAR(20),
```

```
    @p_user_name VARCHAR(50),
    @p_pwd VARCHAR(40),
    @role VARCHAR(20)
AS
BEGIN
    DECLARE @salt UNIQUEIDENTIFIER = NEWID();
    DECLARE @ret INTEGER;
    BEGIN TRY
        INSERT INTO user_list (user_id, user_name, pwd_hash,
salt, role)
        VALUES          (@p_user_id,          @p_user_name,
HASHBYTES('SHA2_512', @p_pwd + CAST(@salt AS VARCHAR(40))),
@salt, @role);
        SET @ret = 0
    END TRY
    BEGIN CATCH
        SET @ret = ERROR_NUMBER()
    END CATCH

    SELECT @ret
END
```

用户登陆。接受账号和密码作为参数，返回 1 代表是管理员，2 代表学生，3 代表教师，负值代表验证失败。

```
CREATE PROCEDURE user_login
    @p_user_id VARCHAR(20),
    @p_pwd VARCHAR(40)
AS
BEGIN
```

```
    DECLARE @role VARCHAR(20)
    DECLARE @ret INTEGER
    SET @role = (SELECT role FROM user_list
        WHERE user_id = @p_user_id AND pwd_hash =
HASHBYTES('SHA2_512', @p_pwd + CAST(salt AS VARCHAR(40))))
    IF @role IS NULL /*wrong user id or password*/
        SET @ret = -1
    ELSE IF @role = 'administrator'
        SET @ret = 1
    ELSE IF @role = 'student'
        SET @ret = 2
    ELSE IF @role = 'instructor'
        SET @ret = 3
    ELSE /*unexpected error*/
        SET @ret = -2
    SELECT @ret
END
```

替换密码。接受用户账号和新密码作为参数，只能由管理员使用，直接将该用户密码更新为新密码，返回 0 代表执行成功。

```
CREATE PROCEDURE renew_pwd
    @p_user_id VARCHAR(20),
    @p_pwd VARCHAR(40)
AS
BEGIN
    DECLARE @salt UNIQUEIDENTIFIER = NEWID();
    DECLARE @ret INTEGER;
    BEGIN TRY
        UPDATE user_list SET pwd_hash = HASHBYTES('SHA2_512',
```

```
@p_pwd + CAST(@salt AS VARCHAR(40))), salt = @salt
      WHERE user_id = @p_user_id
      SET @ret = 0
   END TRY
   BEGIN CATCH
      SET @ret = ERROR_NUMBER()
   END CATCH

   SELECT @ret
END
```

更改密码。接受用户账号、原密码、新密码作为参数，如原密码正确则更改密码为新密码并返回 0，否则返回错误代码。

```
CREATE PROCEDURE change_pwd
   @p_user_id VARCHAR(20),
   @p_origin_pwd VARCHAR(40),
   @p_new_pwd VARCHAR(40)
AS
BEGIN
   DECLARE @salt UNIQUEIDENTIFIER = NEWID();
   DECLARE @role VARCHAR(20);
   DECLARE @ret INTEGER;
   BEGIN TRY
      SET @role = (SELECT role FROM user_list
         WHERE user_id = @p_user_id AND pwd_hash =
HASHBYTES('SHA2_512', @p_origin_pwd + CAST(salt AS
VARCHAR(40))))
      IF @role IS NULL
         SET @ret = -1
```

```
      ELSE
      BEGIN
         UPDATE user_list SET pwd_hash =
HASHBYTES('SHA2_512', @p_new_pwd + CAST(@salt AS
VARCHAR(40))), salt = @salt
         WHERE user_id = @p_user_id
         SET @ret = 0
      END
   END TRY
   BEGIN CATCH
      SET @ret = ERROR_NUMBER()
   END CATCH

   SELECT @ret
END
```

查询空闲教室。接受参数指定星期，起始小节与结束小节，查询在该时段内的空闲教室信息，如某参数为空则代表不指定该项条件。

```
CREATE PROC queryRoom
@weekday VARCHAR(3),
@start_time INT,
@end_time INT
AS
SET NOCOUNT ON
BEGIN
   IF @start_time = 0
      SET @start_time = 1
   IF @end_time = 0
      SET @end_time = 11
```

```
   DECLARE @year INT, @semester VARCHAR(10)
   SET @year = CONVERT(INT, (SELECT value FROM status WHERE
keyword = 'Current year'))
   SET @semester = CONVERT(VARCHAR(10), (SELECT value FROM
status WHERE keyword = 'Current semester'))

   (   SELECT building, room_number
      FROM classroom
   ) EXCEPT (
      SELECT building, room_number
      FROM sec_arrange, timeslot, section
      WHERE           sec_arrange.time_slot_id           =
timeslot.time_slot_id
         AND sec_arrange.sec_id = section.sec_id
         AND year = @year AND semester = @semester
         AND day = @weekday AND end_time_id >= @start_time
AND start_time_id <= @end_time
   )
END
```

查询学生/教师信息。此过程抽象了教师与学生的信息查询过程，使用同一个查询语句即可兼容学生与教师，方便了程序的设计。

```
CREATE PROC queryInfo
@id VARCHAR(20), @role INT
AS
BEGIN
   IF @role = 2
      SELECT name, stu_id, dept_name, sexual, year, birthday,
phone_number, address, email, identity_number
```

```
      FROM student
      WHERE stu_id = @id
   ELSE IF @role = 3
      SELECT name, inst_id, dept_name, sexual, year,
birthday, phone_number, address, email, identity_number
      FROM instructor
      WHERE inst_id = @id
END
```

更新学生/教师信息。此过程抽象了教师与学生的信息更新过程，使用同一个查询语句即可兼容学生与教师，方便了程序的设计。

```
CREATE PROC updateInfo
@id VARCHAR(20), @role INT,
@birthday DATE,
@phone VARCHAR(20),
@address VARCHAR(100),
@email VARCHAR(50),
@identity VARCHAR(20)
AS
BEGIN
   IF @role = 2
      UPDATE student
      SET birthday = @birthday, phone_number = @phone,
address = @address, email = @email, identity_number =
@identity
      WHERE stu_id = @id
   ELSE IF @role = 3
      UPDATE instructor
      SET birthday = @birthday, phone_number = @phone,
```

```
address = @address, email = @email, identity_number =
@identity
      WHERE inst_id = @id
END
```

统计绩点。此过程执行后会进行当前学期的绩点统计，按照平时成绩 40%+期末成绩 60%的方式计算出总成绩，并在每个授课班级内按总成绩自高到低评定绩点，绩点评定方式与华东师大 2015 级本科生手册中记载的评定方式一致。但出于简化，没有实现人数不足 5 人时的对应方法，对此情形需要管理员后台修改。

```
CREATE PROC settle_mark AS
BEGIN
   DECLARE @semester VARCHAR(50), @year INT;
   SET @semester = (SELECT value FROM status WHERE keyword
= 'Current semester')
   SET @year = (SELECT value FROM status WHERE keyword =
'Current year')

   ALTER TABLE takes DISABLE TRIGGER protectScore
   UPDATE takes SET total_grade =
0.4*regular_grade+0.6*final_grade
   WHERE sec_id IN (SELECT sec_id FROM section WHERE semester
= @semester AND year = @year)

   UPDATE takes SET mark = 'F', grade_point = 0
   WHERE total_grade < 60.0 AND status = 'INP'

   UPDATE takes
   SET mark = CASE
   WHEN ranking <= 0.1 THEN 'A'
```

```
   WHEN ranking > 0.1 AND ranking <= 0.3 THEN 'A-'
   WHEN ranking > 0.3 AND ranking <= 0.45 THEN 'B+'
   WHEN ranking > 0.45 AND ranking <= 0.6 THEN 'B'
   WHEN ranking > 0.6 AND ranking <= 0.8 THEN 'B-'
   WHEN ranking > 0.8 AND ranking <= 0.9 THEN 'C+'
   WHEN ranking > 0.9 AND ranking <= 1 THEN 'C'
   END,
   grade_point = CASE
   WHEN ranking <= 0.1 THEN 4.0
   WHEN ranking > 0.1 AND ranking <= 0.3 THEN 3.7
   WHEN ranking > 0.3 AND ranking <= 0.45 THEN 3.3
   WHEN ranking > 0.45 AND ranking <= 0.6 THEN 3.0
   WHEN ranking > 0.6 AND ranking <= 0.8 THEN 2.5
   WHEN ranking > 0.8 AND ranking <= 0.9 THEN 2
   WHEN ranking > 0.9 AND ranking <= 1 THEN 1.5
   END
   FROM (SELECT sec_id, stu_id, PERCENT_RANK() OVER
(PARTITION BY sec_id ORDER BY total_grade DESC) AS ranking
      FROM takes
      WHERE total_grade IS NOT NULL AND total_grade >= 60)
AS tmp
   WHERE takes.sec_id = tmp.sec_id AND takes.stu_id =
tmp.stu_id
   AND takes.sec_id IN (SELECT sec_id FROM section WHERE
semester = @semester AND year = @year)

   UPDATE takes SET status = 'FIN'
   WHERE sec_id IN (SELECT sec_id FROM section WHERE semester
= @semester AND year = @year)
```

```
   ALTER TABLE takes ENABLE TRIGGER protectScore
END
```

5. 函数设计

由于本数据库中一节开课可以对应多个授课教师和多个开课时间地点，为方便开课相关查询，定义了如下函数，接受一个 sec_id 为参数，返回一个授课教师列表（格式为：教师 1;教师 2;…）或开课时间地点列表（格式为：星期 1 [第 i1 节-第 j1 节], 教室 1 房间号 1;星期 2 [第 i2 节-第 j2 节], 教室 2 房间号 2;…）的字符串。

```
CREATE FUNCTION dbo.ConcatSecArr(@sec_id INTEGER)
RETURNS VARCHAR(1000)
WITH SCHEMABINDING
AS
BEGIN
   DECLARE @s VARCHAR(1000);
   SELECT @s = COALESCE(@s + ';', '') + day + ' ' +
CONVERT(VARCHAR(2), start_time_id) + '-' +
CONVERT(VARCHAR(2), end_time_id) +
      ', ' + building + ' ' + room_number
   FROM dbo.sec_arrange JOIN dbo.timeslot ON
sec_arrange.time_slot_id = timeslot.time_slot_id
   WHERE sec_id = @sec_id
   RETURN (@s);
END
GO

CREATE FUNCTION dbo.ConcatTeachers(@sec_id INTEGER)
RETURNS VARCHAR(1000)
```

```
WITH SCHEMABINDING
AS
BEGIN
   DECLARE @s VARCHAR(1000);
   SELECT @s = COALESCE(@s + ';', '') + name
    FROM dbo.teaches JOIN dbo.instructor ON teaches.inst_id
= instructor.inst_id
   WHERE sec_id = @sec_id
   RETURN (@s);
END
GO
```

6. 视图设计

查询开课视图。本视图包含开课编号、对应课程编号、对应课程标题、开课年份、开课学期、课程类型、开课院系、授课教师列表、上课安排列表、课程容量和学分，方便开课相关查询的执行。

```
CREATE VIEW querySection(sec_id, course_id, title, year,
semester, type, dept_name, instructors, section_arranges,
capacity, credits) AS (
   SELECT sec_id, section.course_id, title, year, semester,
type,                                    course.dept_name,
dbo.ConcatTeachers(section.sec_id),
dbo.ConcatSecArr(section.sec_id), capacity, credits
   FROM course, section
   WHERE course.course_id = section.course_id
);
```

7. 安全机制设计

众所周知，在数据库中以明文存放密码是十分危险的行为，一旦遭到脱库，所有用户的信息都将毫无保留地呈现在攻击者的面前，因此我们选择在数据库中存放密码的哈希值。但只是存储密码的哈希还不够保证用户账户的安全，一旦攻击者实施字典攻击或者是 rainbow table 攻击，弱密码将可能被轻易攻破。有鉴于此，我们还对每个用户密码生成一串 SALT 值保存在数据库中，并将其连接在用户密码后面用作密码哈希验证，这样即使是弱密码也可以得到充分保护。

在本次实践中，我们使用 MSSQL 提供的 HASHBYTES 函数进行 SHA2-512 哈希加密，并使用 NEWID 函数生成一串长度为 36 的 UNIQUEIDENTIFIER 类型随机字符串作为 SALT 值，并在此基础上封装了用户注册、登录、修改密码相关的存储过程。存储过程的定义参见 4.3 节。

五. 应用程序设计

1. 开发及运行环境介绍

系统使用的开发工具是 PyQt5 和 MSSQL Server 2017。在 Jetbrains PyCharm 与 SSMS 环境下开发。

2. 主要功能设计

本数据库采用 C/S 模式设计，客户端为 GUI 程序，程序的工作流程如图 5-2 所示。

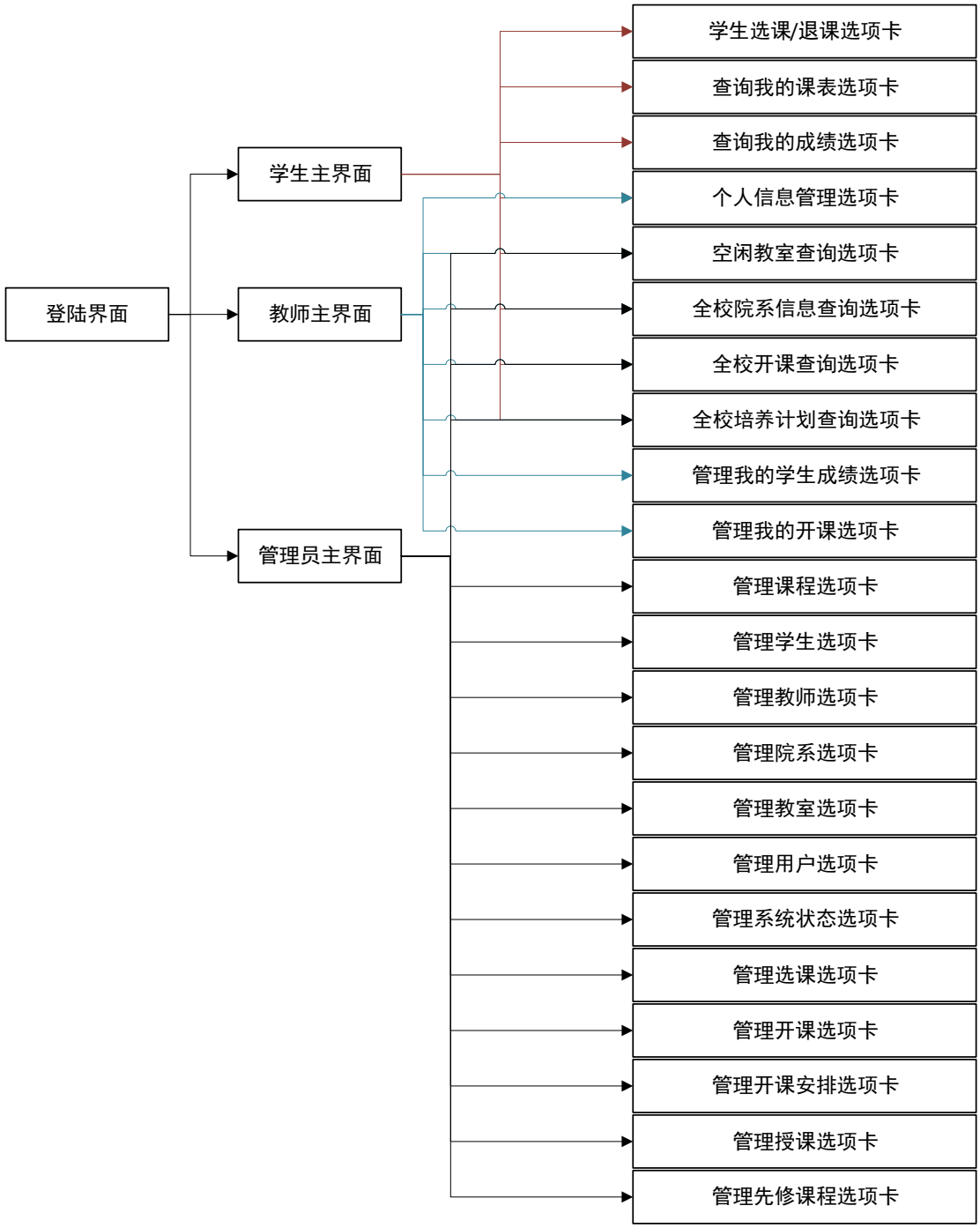


图 5-2 程序工作流程图

3. 主要界面

登录界面如图 5-3 所示。

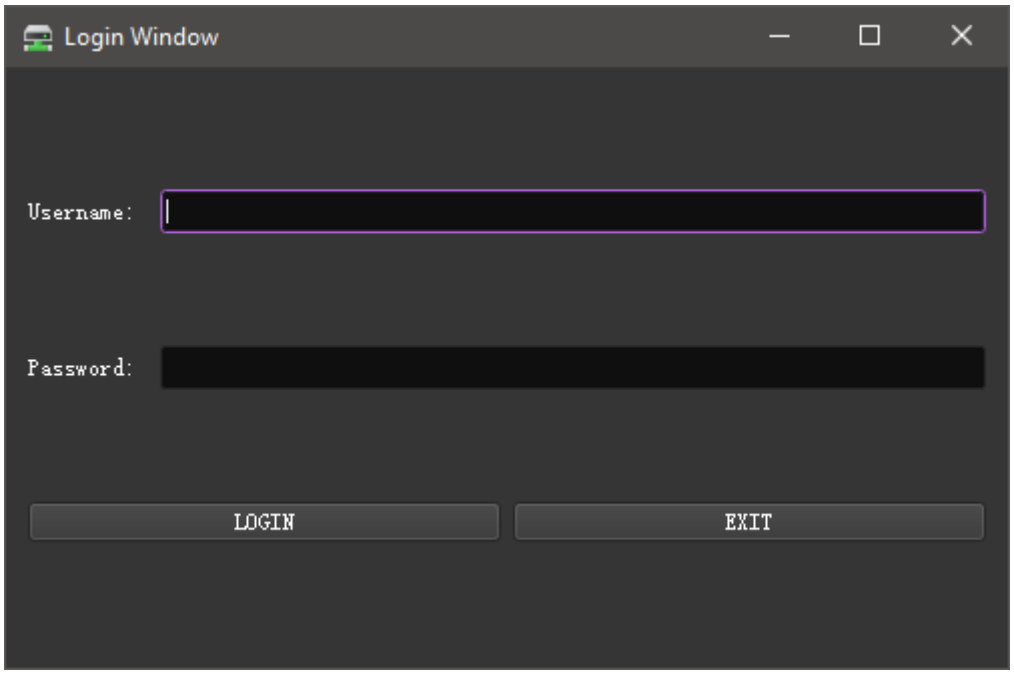
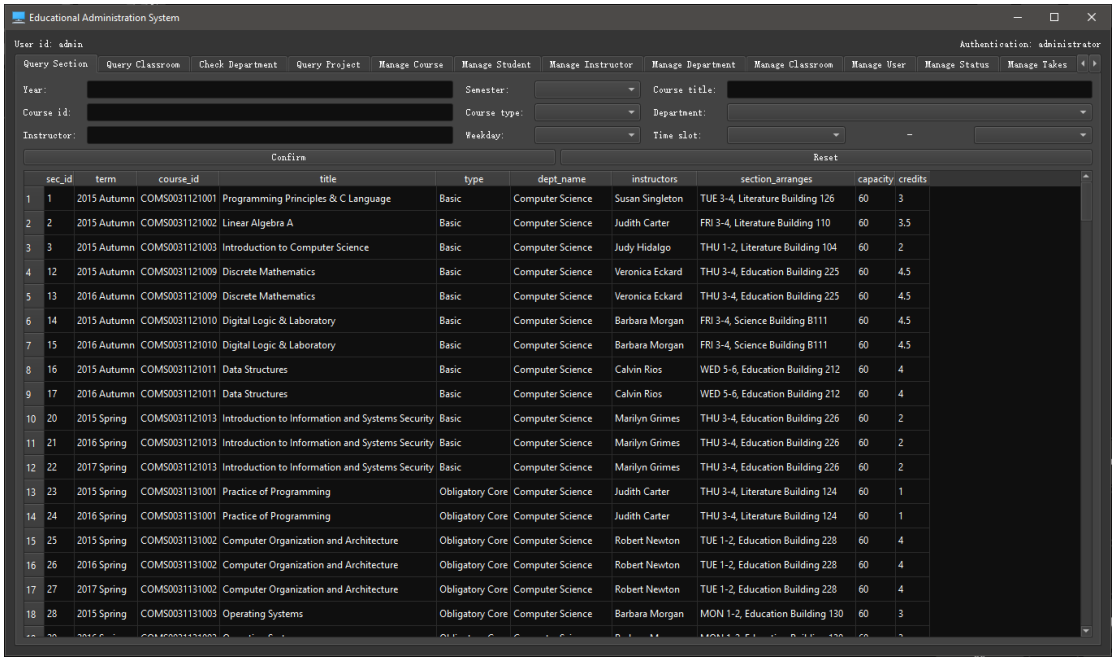


图 5-3 登陆界面

管理员主界面如图 5-4 所示，由于选项卡较多，只展示其一。



	sec_id	term	course_id	title	type	dept_name	instructors	section_arranges	capacity	credits
1	1	2015 Autumn	COMS0031121001	Programming Principles & C Language	Basic	Computer Science	Susan Singleton	TUE 3-4, Literature Building 126	60	3
2	2	2015 Autumn	COMS0031121002	Linear Algebra A	Basic	Computer Science	Judith Carter	FRI 3-4, Literature Building 110	60	3.5
3	3	2015 Autumn	COMS0031121003	Introduction to Computer Science	Basic	Computer Science	Judy Hidalgo	THU 1-2, Literature Building 104	60	2
4	12	2015 Autumn	COMS0031121009	Discrete Mathematics	Basic	Computer Science	Veronica Eckard	THU 3-4, Education Building 225	60	4.5
5	13	2016 Autumn	COMS0031121009	Discrete Mathematics	Basic	Computer Science	Veronica Eckard	THU 3-4, Education Building 225	60	4.5
6	14	2015 Autumn	COMS0031121010	Digital Logic & Laboratory	Basic	Computer Science	Barbara Morgan	FRI 3-4, Science Building B111	60	4.5
7	15	2016 Autumn	COMS0031121010	Digital Logic & Laboratory	Basic	Computer Science	Barbara Morgan	FRI 3-4, Science Building B111	60	4.5
8	16	2015 Autumn	COMS0031121011	Data Structures	Basic	Computer Science	Calvin Rios	WED 5-6, Education Building 212	60	4
9	17	2016 Autumn	COMS0031121011	Data Structures	Basic	Computer Science	Calvin Rios	WED 5-6, Education Building 212	60	4
10	20	2015 Spring	COMS0031121013	Introduction to Information and Systems Security	Basic	Computer Science	Marilyn Grimes	THU 3-4, Education Building 226	60	2
11	21	2016 Spring	COMS0031121013	Introduction to Information and Systems Security	Basic	Computer Science	Marilyn Grimes	THU 3-4, Education Building 226	60	2
12	22	2017 Spring	COMS0031121013	Introduction to Information and Systems Security	Basic	Computer Science	Marilyn Grimes	THU 3-4, Education Building 226	60	2
13	23	2015 Spring	COMS0031131001	Practice of Programming	Obligatory Core	Computer Science	Judith Carter	THU 3-4, Literature Building 124	60	1
14	24	2016 Spring	COMS0031131001	Practice of Programming	Obligatory Core	Computer Science	Judith Carter	THU 3-4, Literature Building 124	60	1
15	25	2015 Spring	COMS0031131002	Computer Organization and Architecture	Obligatory Core	Computer Science	Robert Newton	TUE 1-2, Education Building 228	60	4
16	26	2016 Spring	COMS0031131002	Computer Organization and Architecture	Obligatory Core	Computer Science	Robert Newton	TUE 1-2, Education Building 228	60	4
17	27	2017 Spring	COMS0031131002	Computer Organization and Architecture	Obligatory Core	Computer Science	Robert Newton	TUE 1-2, Education Building 228	60	4
18	28	2015 Spring	COMS0031131003	Operating Systems	Obligatory Core	Computer Science	Barbara Morgan	MON 1-2, Education Building 130	60	3

图 5-4 管理员主界面

学生主界面如图 5-5 所示，由于选项卡较多，只展示其一。

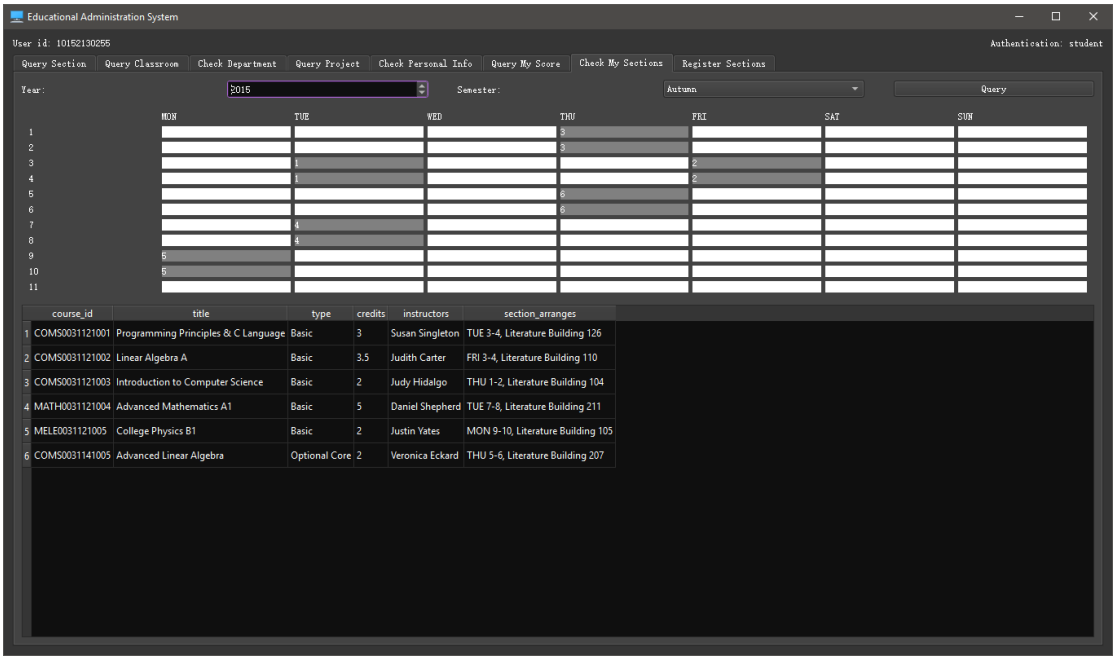


图 5-5 学生主界面

教师主界面如图 5-6 所示，由于选项卡较多，只展示其一。

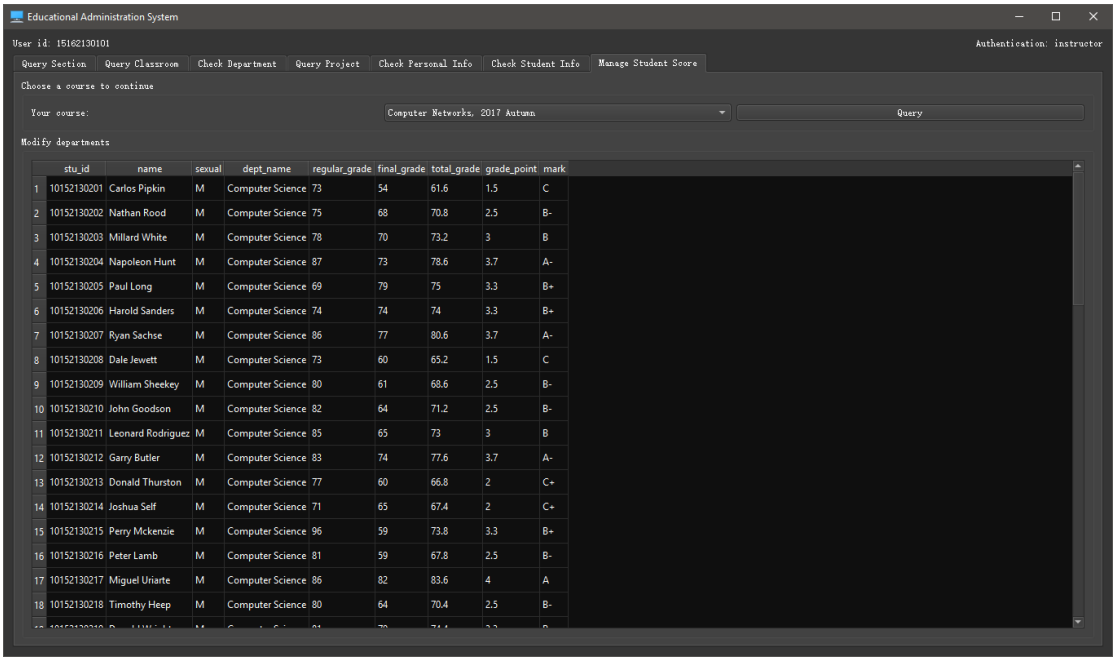


图 5-6 教师主界面

六. 心得体会

在本次数据库实践课程设计过程中，由于运用了良好的软件开发流程与技术成熟的模块，并具备了充足的前期储备知识与时间经验，设计顺利完成，没有遇到大的问题。

通过本次数据库实践的课程设计，加深了对数据库系统理论的理解，熟悉了 MSSQL Server 的使用，理解了存储过程、触发器等工具在数据库构建中的重要作用，熟练了使用 PyQt5 开发 GUI 程序的流程。