西安交通大學

XI'AN JIAOTONG UNIVERSITY

DFD 模型转化为 UML 模型

第五次作业

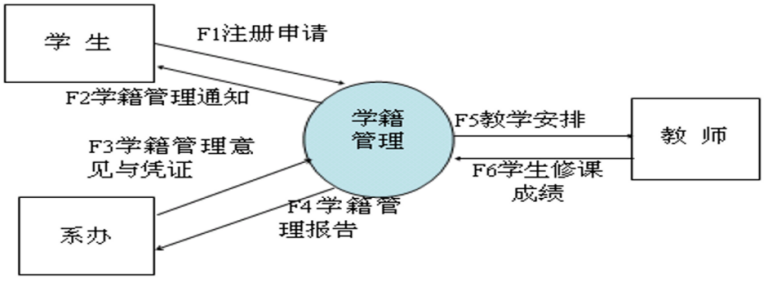
课程名称：	软件系统分析与设计
姓名：	凌晨
学院：	软件学院
专业：	软件工程
学号：	**2214414320**

2023 年 11 月 5 日

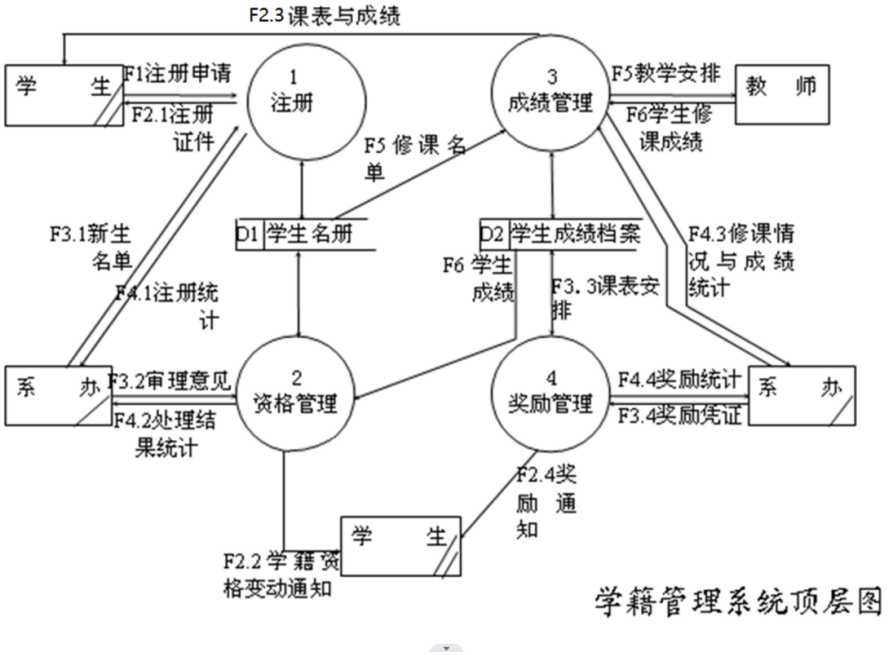
目录

一、 目的和要求

- 系统需求分析与设计： 你作为一名软件系统分析员，在某一个高校的学生管理系统中负责系统的分析与设计工作，为了更快地将客户的需求进行建模，你采用了DFD的方法建立了两层数据流模型如下：



学籍管理系统关联图



学籍管理系统顶层图

问题要求：

- 但是在与客户经理以及开发人员进行沟通交流时，大家认为这种描述方法已过时，希望能够采用面向对象的方法来进行业务需求的建模与分析，迫于用户和开发人员的要求，你准备对现有的建模方法进行调整。

- 请利用UML的建模方法将该模型转换成等效的功能模型（USE CASE图，并简要描述事件流）、动态模型（活动图与分析时序图）、以及静态模型（分析类图），数据库ER模型，注意说明并解释模型之间存在的关系，且可以根据需要进行扩展，尽量完整和细化。

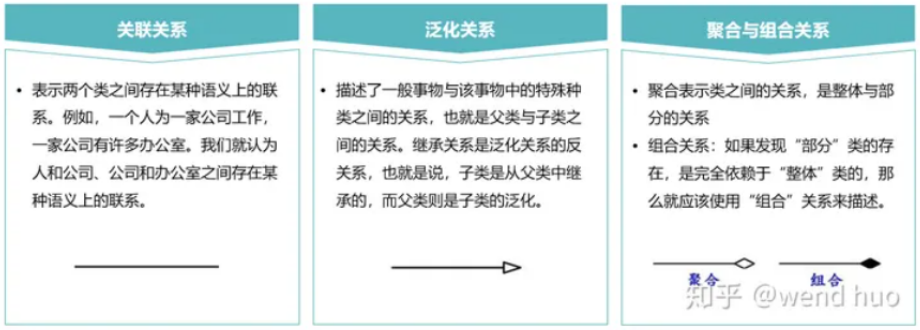
二、　面向对象

1. 类的表示法

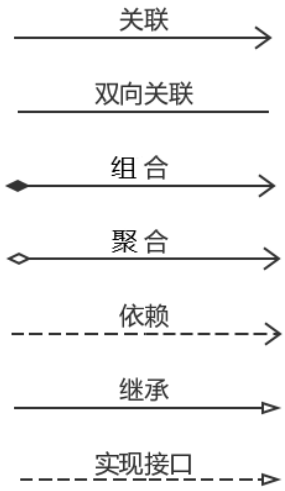
类是对一组具有相同属性、操作、关系和语义的对象的描述。关系是类之间的、语义是蕴藏的，对于一个类而言，其关键的特性是属性（成员变量）和操作（成员方法）。类用一个矩形表示的，包含三个分栏，每个分栏分别写入类的名称、类的属性和类的操作。

2. 类之间的关系

类之间通常有关联关系、泛化关系、聚合与组合关系。



除此之外，在 UML 类图中，定义了六种类之间的关系，他们分别是：实现（Realization），依赖 (Dependency) 关系。



3. 类与对象

对象是由数据（描述事物的属性）和作用于数据的操作（体现事物的行为）组成的封装体，描述客观事物的一个实体，是构成系统的基本单元。

类是对一组有相同数据和相同操作的对象的定义，是对象的模板，其包含的方法和数据描述一组对象的共同行为和属性。类是在对象之上的抽象，对象则是类的具体化，是类的实例。类可有其子类，也可有其他类，形成类层次结构。

三、 面向对象的需求分析建模

在面向对象的需求分析模型中一般而言存在两种模型：领域模型和用例模型，下面将分别讲解这两个模型的概念，已经建立步骤。

1. 领域模型

领域模型是对领域内的概念类或现实世界中对象的可视化表示。又称概念模型、领域对象模型、分析对象模型。它专注于分析问题领域本身，发掘重要的业务领域概念，并建立业务领域概念之间的关

系。概念比较深奥，其实说白了就是把基于对业务的理解画成一个类图，并画出这些类之间的关系（面向对象）。

领域模型可以整理业务中的概念以及关系，帮助团队中的成员对业务的理解保持一致，往后可以指导数据库设计、系统功能设计、指导开发。在整个系统建设周期能起到上接需求，下承开发的作用。

虽然领域模型如此重要，但并不是要在类图中尽可能的展示对象的属性和方法，在建模的时候不要将注意力集中在属性或行为上，应该摆脱这些细枝末节，抓住领域对象定义的最基本特征，只需要体现对象模型的重要概念。如果细节过多很容易产生”只见树木，不见森林“的现象。

2. 领域模型的建立

完成一个领域模型建模，主要需要做两件事：

(1) 定义类的关键属性和关键行为；

(2) 定义类与类之间的关联关系。

下面为建立步骤：

(1) 识别概念类：找出当前需求中的候选概念类：通过对用例描述中的名词或名词短语寻找和识别概念类；

(2) 描述概念类：在领域模型中描述这些概念类。用问题域中的词汇对概念类进行命名，将与当前需求无关的概念类排除在外。

(3) 添加关联：在概念类之间添加必要的关联来记录那些需要保存记忆的关系，概念之间的关系用关联、继承、组合/聚合等等来表示。

概念类和属性的辨析:

- 属性一般是可以赋值的，比如数字或者文本。如果该名词不能被赋值，那么就“有可能”是一个概念类。
- 如果对一个名词是概念类还是属性举棋不定的时候，最好将其作为概念类处理。

3. 用例模型

定义了“目标系统”做什么的需求。由以下四个部分组成：

(1) 用例图（基础）：角色、用例、联系

(2) 用例说明（基础）：成功场景和分支场景

(3) 系统顺序图（附加说明）

(4) 操作契约（附加说明）

4. 用例模型的建立

以用例为核心从使用者的角度描述和解释待构建系统的功能需求。

(1) 确定问题域的分析范围：问题域指的是在一次用例建模过程中，需要思考的问题边界，和场景相关（场景包括环节，环节中体现问题）

(2) 确定该范围内可能出现的角色：可以是使用系统的使用者、和用户使用场景关联的人、

(3) 根据业务背景或者领域模型：确定每个角色需要的用例，并形成用例图，每个角色用例不同。

(4) 基于确定的用例：整理成规范的用例描述文本，也就是需要生成用例说明

(5) 用例图整合：在可能的情况下，将多个角色的用例图整合成一个相对完整的用例图；

(6) 绘制系统顺序图：针对每个用例，结合相应的用例描述，确定系统顺序图中角色与系统之间的交互，绘制基于用例的系统顺序图，系统顺序图描述角色和系统的交互过程

(7) 确定操作契约：基于每个系统顺序图，确定每个事件交互经过系统处理后应该返回给角色的声明性结果，即操作契约；

四、 UML

UML（统一建模语言）是一种用于软件系统建模的标准化语言。它提供了一组图形符号和规则，用于描述和可视化软件系统的结构、行为和交互。

UML 包含多种图形符号，用于表示系统中的不同方面和概念。以下是 UML 中常用的几种图形符号：

1. 类图（*Class Diagram*）：用于描述系统的静态结构，包括类、对象、属性、方法和类之间的关系。类图是 UML 中最常用的图形符号之一。

2. 时序图（*Sequence Diagram*）：用于描述对象之间的交互和消息传递顺序。时序图显示对象之间的时间顺序和消息流，有助于表示系统的动态行为。

3. 活动图（*Activity Diagram*）：用于描述系统中的业务流程、操作流程或算法。活动图显示了各个活动、决策点、并行处理和条件分支等，有助于描述系统的行为流程。

4. 用例图（*Use Case Diagram*）：用于描述系统的功能需求和参与者之间的关系。用例图显示了系统的各个功能模块（用例）以及与之交互的参与者（用户、外部系统等）。

5. 组件图（*Component Diagram*）：用于描述系统的组件和组件之间的关系。组件图显示了系统的物理组件、库、框架等，以及它们之间的依赖和接口。

6. 部署图（*Deployment Diagram*）：用于描述系统的物理部署结构，包括硬件设备、软件组件和网络连接。部署图显示了系统的分布和部署方式。

除了上述常用的图形符号，UML 还包括其他一些图形符号，如状态图（State Diagram）、通信图（Communication Diagram）、包图（Package Diagram）等，用于描述系统的不同方面和视角。

UML 提供了一种标准化的语言和图形符号，使得软件开发团队可以更好地理解和沟通系统的设计和需求。它可以用于不同的开发阶段，从需求分析到系统设计和实现，以及系统的文档编写和维护。UML 的使用有助于提高软件开发的可靠性、可维护性和可扩展性。

五、 领域模型的建立

1. 领域词汇分析

首先，需要将题目所给的学籍管理系统的关联图和学籍管理系统顶层图进行剖析，得到了以下名词和动词。

名称	类型	属性
学生	名词	外部实体
系办	名词	外部实体
教师	名词	外部实体
1 注册	动词	数据处理
2 资格管理	动词	数据处理
3 成绩管理	动词	数据处理
4 奖励管理	动词	数据处理
D1 学生名册	动词	数据存储
D2 学生成绩档案	动词	数据存储

其中，对于数据处理部分，我们需要更详细的剖析，如下表：

表 1: 数据流向

功能	数据流向
F1 注册申请	由学生流向注册
F2.1 注册证件	由注册流向学生
F2.2 学籍资格变动通知	由资格管理流向学生
F2.3 课表与成绩	由成绩管理流向学生
F2.4 奖励通知	由奖励管理流向学生
F3.1 新生名单	由系办流向注册
F3.2 审理意见	由系办流向资格管理
F3.3 课表安排	由学生成绩档案流向奖励管理
F3.4 奖励凭证	由奖励管理流向系办
F4.1 注册统计	由注册流向系办
F4.2 处理结果统计	由资格管理流向系办
F4.3 修课情况与成绩统计	由成绩管理流向系办
F4.4 奖励统计	由奖励管理流向系办
F5 教学安排	由成绩管理流向教师
F6 学生修课成绩	由教师流向成绩管理

2. 词汇分析

下表为基本分析步骤：

表 2: 基本分析步骤

属性	分析
外部实体	根据学校的环境，定义符合其外部实体的属性即可，同时看实体的有关动词定义行为
数据存储	在保证数据存储功能完整的情况下，可以进行拆分减少冗余，建立数据字典
数据处理	找到需要处理的数据，使用已有或建立类/属性，使用类之间的关系抽象处理过程

3. 类/接口的建立

下面为分析出来的类/接口：

表 3: 学生类

属性名	属性类型
姓名	*string*
性别	*byte*
年龄	*int*
学号	*string*
专业	*string*
班级	*string*
学院	*string*
行为	
注册	
查看通知	
查看成绩	
查看课表	

表 4: 教师类

属性名	属性类型
姓名	*string*
性别	*byte*
年龄	*int*
任职	*string*
任职时长	*string*
行为	
登记学生修课成绩	
上传教学安排	

表 5: 系办类

属性名	属性类型
系名	*string*
联系方式	*string*
行为	
提供新生名单	
提供审理意见	
课表安排	
提供奖励凭证	
查看各种统计	

表 6: 通知类

属性名	属性类型
通知时间	*Date*
通知内容	*string*
行为	
推送给学生	

表 7: 课程类

属性名	属性类型
课程名	*string*
课程地点	*string*
课程学时	*int*
课程人数	*int*
行为	
发布到选课系统	
发布到授课系统	

表 8: 奖励类

属性名	属性类型
奖项名称	*string*
奖金	*string*
奖项立项时间	*Date*
行为	
立项（通知系办安排）	
发布通知	

表 9: 审理信息类

属性名	**属性类型**
审理内容 **审理时间**	*string* *Date*
行为	
立项（通知系办审理） **发布通知**	

表 10: 数据事务接口

属性名	**属性类型**
权限	**未知**
行为	
增加数据 **删除数据** **查找数据** **改变数据** **统计数据**	

4. 分析类图的建立

通过上面的分析，类图中的属性和行为已经基本可以确认。下面建立分析类图。

在建立分析类图前，我们可以采用更优雅的设计模式，而非简单的堆砌。主要采用以下几种设计模式：

- 发布-订阅模式
- 事件监听模式

这两个模式在之前的作业中已经说明过了，不再赘述。

下面为分析类图：

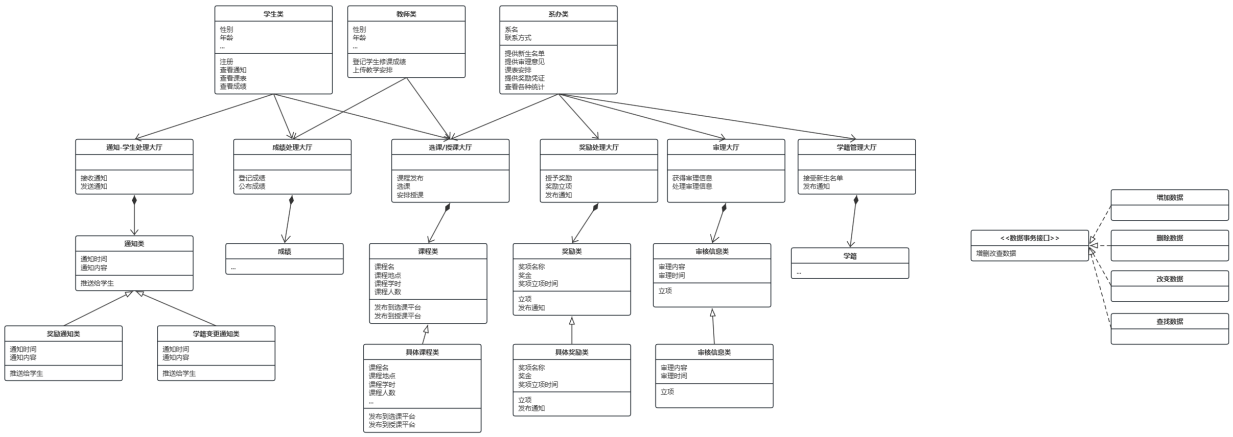


图 1: 分析类图

六、 用例模型的建立

1. 已完成工作

由于已经有 DFD 图和上面的领域模型分析，可以快速的确定问题的分析背景，场景和系统边界。可以得到角色，使用者，用例等。

比如：场景为学校，背景为学校学籍管理系统，系统边界如 DFD 图所示。

2. 绘制用例图

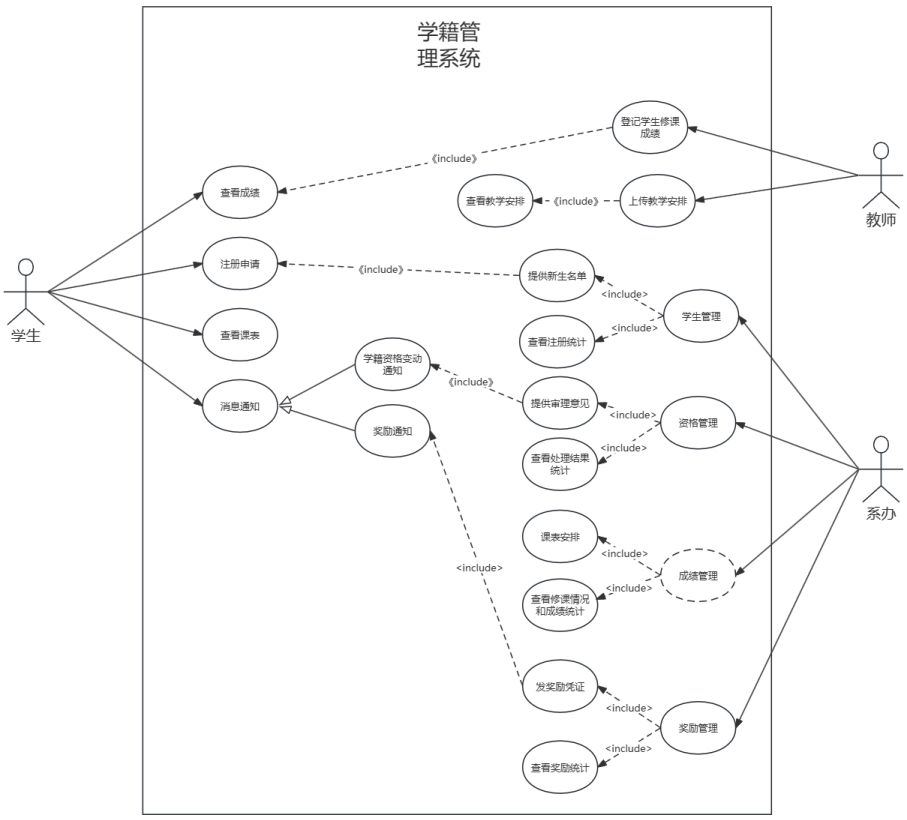


图 2: 用例图

下面详细说明绘制步骤：

(1) 根据确定的系统边界和角色的所需的用例，画出各角色的用例图。（该图为粗粒度）

(2) 根据顶层图，找出每个角色所需用例之间的关系，进行连线关联。（改图为细粒度）

(3) 修改细节，保证前后一致，完成用例图绘制。

3. 用例描述

为了描述用例，建立用例描述表。

表 11: 注册用例

用例名称	注册	
用例参与者	学生	
概述	学生进入系统，提出注册申请	
前置条件	系统运行正常	
后置条件	完成新生注册	
基本事件流	**参与者动作**	**系统响应**
	1）学生在系统界面点击注册	**2）系统跳转至注册页面**
	3）学生按照规定填写基本信息	**4）系统记录该学生信息并返回注册信息**
	5) 学生获得系统发送的注册证件	**6）系统跳转至学生应用界面**
拓展事件流	3a) 若学生信息填写不正确，系统提示：请重新填写正确信息!	
	4a) 若无法记录信息，则注册失败，日记记录失败原因	

表 12: 学生管理用例

用例名称	学生管理	
用例参与者	系办	
概述	系办提供新生名单、查看注册统计	
前置条件	系统运行正常	
后置条件	数据库中添加了新生	
基本事件流	**参与者动作**	**系统响应**
	1）系办上交新生名单	**2）系统检查合法性**
		3）系统记录名单
	4) 查看注册统计	**5）系统提供注册统计名单**
拓展事件流	2a) 若不合法，则提交失败	
	3a) 若无法记录信息，则失败，日记记录失败原因	
	5a) 若无法提供名单，可能数据库服务出错	

表 13: 资格管理用例

用例名称	资格管理	
用例参与者	系办	
概述	系办提供审理意见、查看处理结果统计	
前置条件	系统运行正常	
后置条件	完成资格变动通知	
基本事件流	**参与者动作**	**系统响应**
	1）系办提供审理意见	**2）系统检查合法性**
		3）系统记录提供审理意见
	4) 查看处理结果统计	**5）系统提供处理结果统计**
		6）系统进行学籍资格变动通知
拓展事件流	2a) 若不合法，则提交失败	
	3a) 若无法记录信息，则失败，日记记录失败原因	
	5a) 若无法提供统计，可能数据库服务出错	

表 14: 成绩管理用例

用例名称	成绩管理管理	
用例参与者	系办	
概述	系办安排课表、查看修课情况和成绩统计	
前置条件	系统运行正常	
后置条件	课表安排成功	
基本事件流	**参与者动作**	**系统响应**
	1）系办安排课表	**2）系统检查合法性**
		3）系统记录课表
	4) 查看修课情况和成绩统计	**5）系统提供修课情况和成绩统计**
拓展事件流	2a) 若不合法，则提交失败	
	3a) 若无法记录信息，则失败，日记记录失败原因	
	5a) 若无法提供统计，可能数据库服务出错	

表 15: 奖励管理用例

用例名称	奖励管理	
用例参与者	系办	
概述	系办提供奖励凭证、查看奖励统计	
前置条件	系统运行正常	
后置条件	完成奖励通知	
基本事件流	**参与者动作**	**系统响应**
	1）系办提供奖励凭证	**2）系统检查合法性**
		3）系统记录奖励凭证
	4) 查看奖励统计	**5）系统提供处理结果统计**
		6）系统进行奖励通知
拓展事件流	2a) 若不合法，则提交失败	
	3a) 若无法记录信息，则失败，日记记录失败原因	
	5a) 若无法提供统计，可能数据库服务出错	

由于篇幅原因，不再给出用例描述了，可以仿照上述用例描述。

七、 动态模型

1. 活动图

下面只进行**学生注册和资格管理**的活动图的绘制，其他活动要么简单绘制，要么与这两个活动图类似，不再提供！

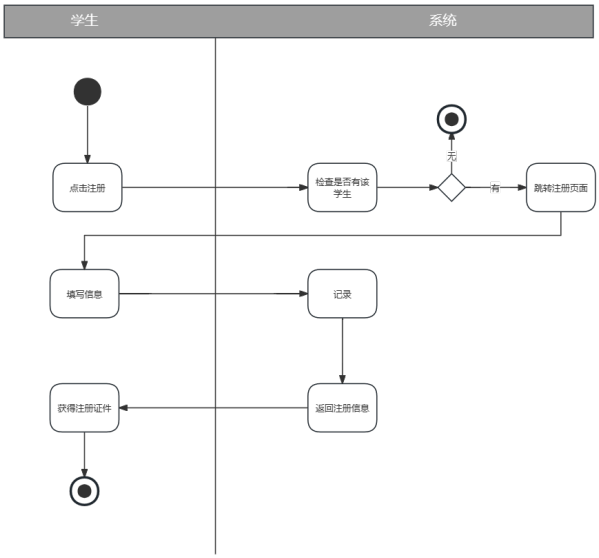


图 3: 学生注册活动

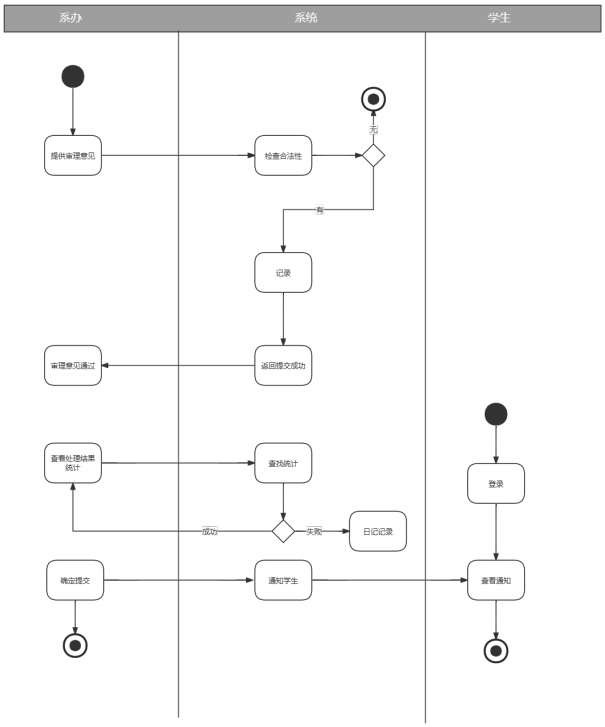


图 4: 资格管理活动图

2. 分析时序图

下面只进行**学生注册和学生管理**的分析时序图的绘制，其他时序图要么简单绘制，要么与这两个分析时序图类似，不再提供！

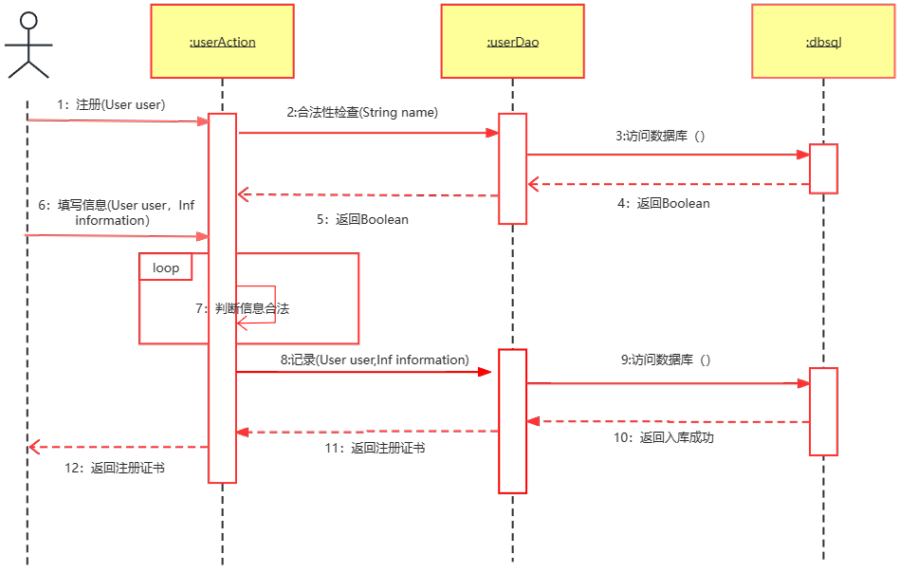


图 5: 学生注册时序图

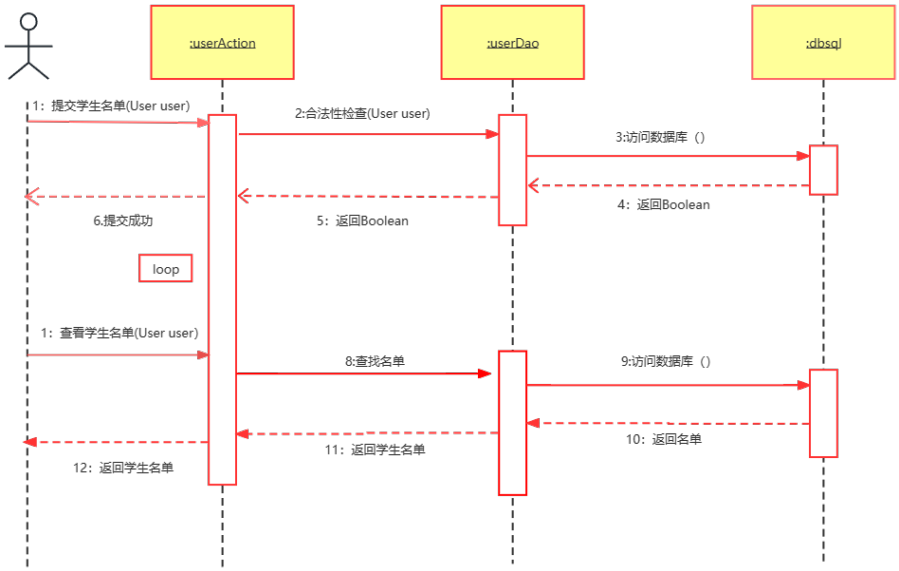


图 6: 学生管理时序图

八、 数据库设计

整体采用 RBAC3 实现数据库的设计，实现题目要求的功能！

1. 逻辑设计

RBAC： RBAC 采用用户-角色-权限分离的经典设计。

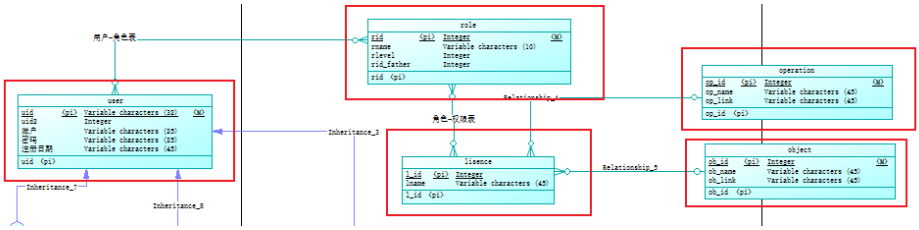


图 7: RBAC 部分

外部实体：一共有三个外部实体，通过继承用户建立实体。

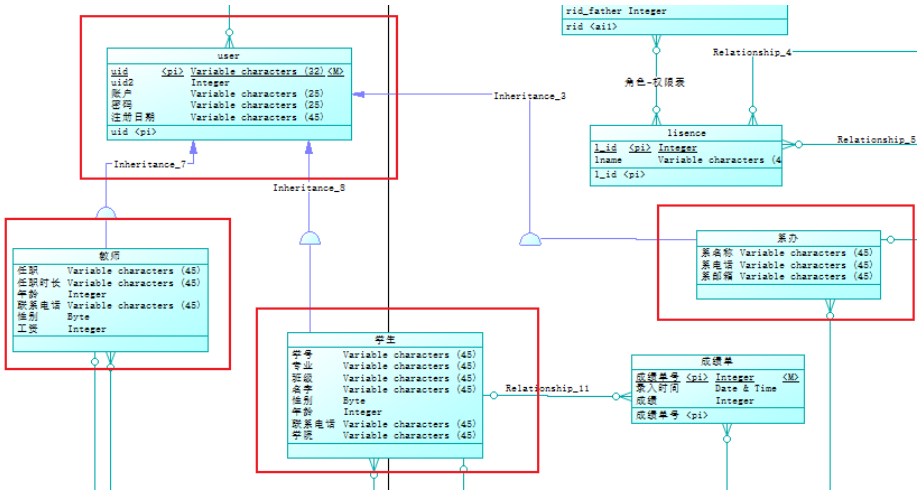


图 8: 外部实体部分

课程相关：通过图 9可以看出，有选课，课程安排，成绩等功能。

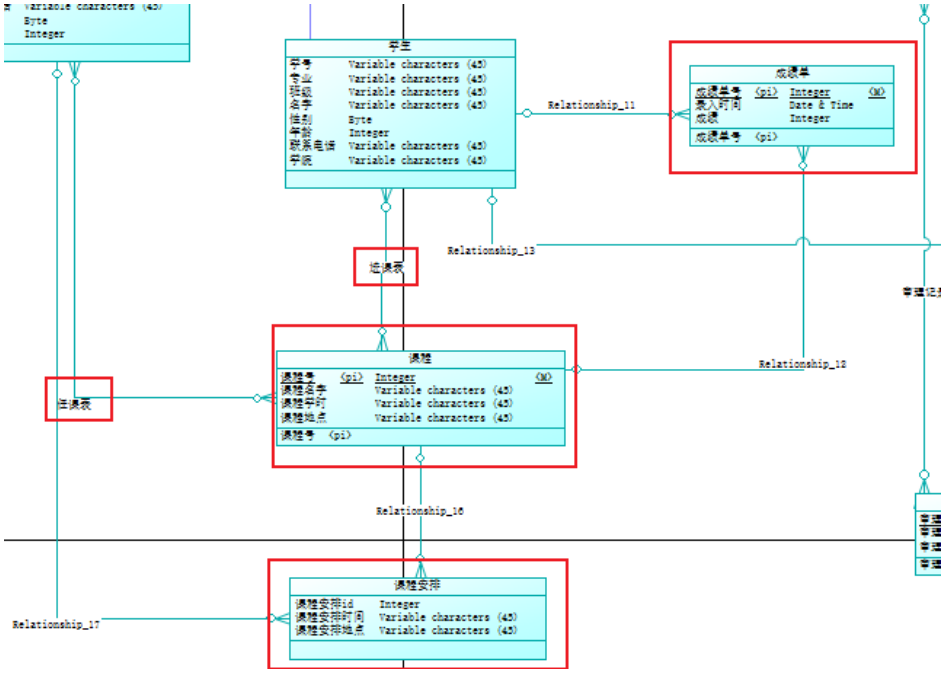


图 9: 课程相关功能部分

通知功能：通过图 10可以看出，有奖励通知和学籍变动通知。通知通过继承简化 ER 图。

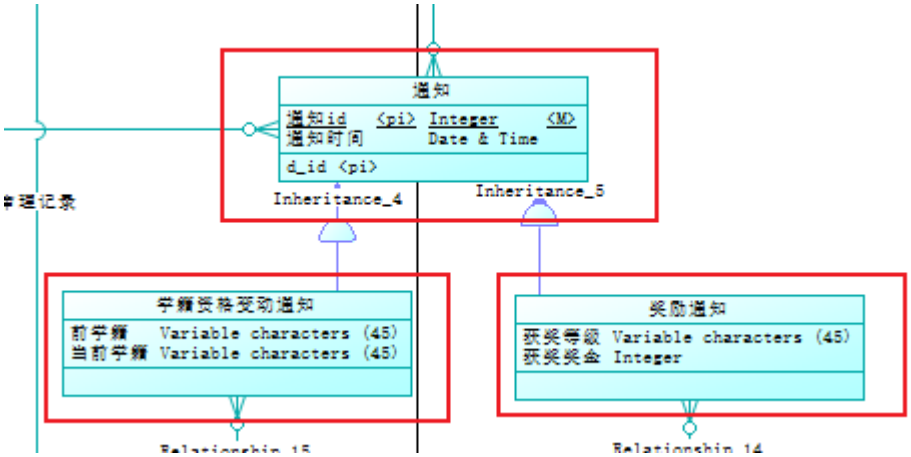


图 10: 通知功能部分

其他功能：通过图 11可以看出，有安排奖励和变动学籍等功能。

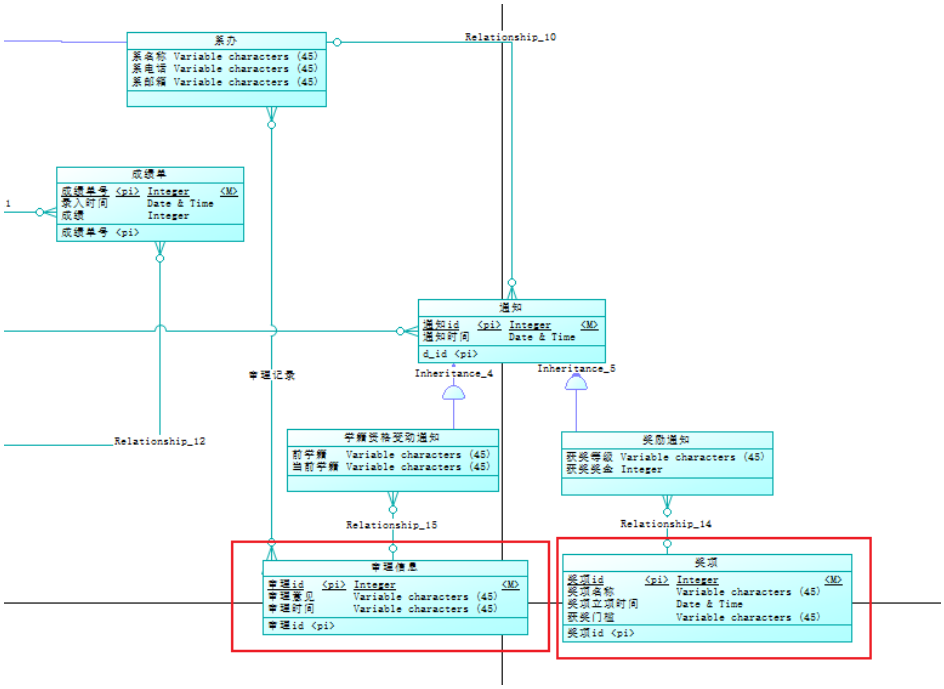


图 11: 其他功能部分

下面为数据库逻辑设计总体图：

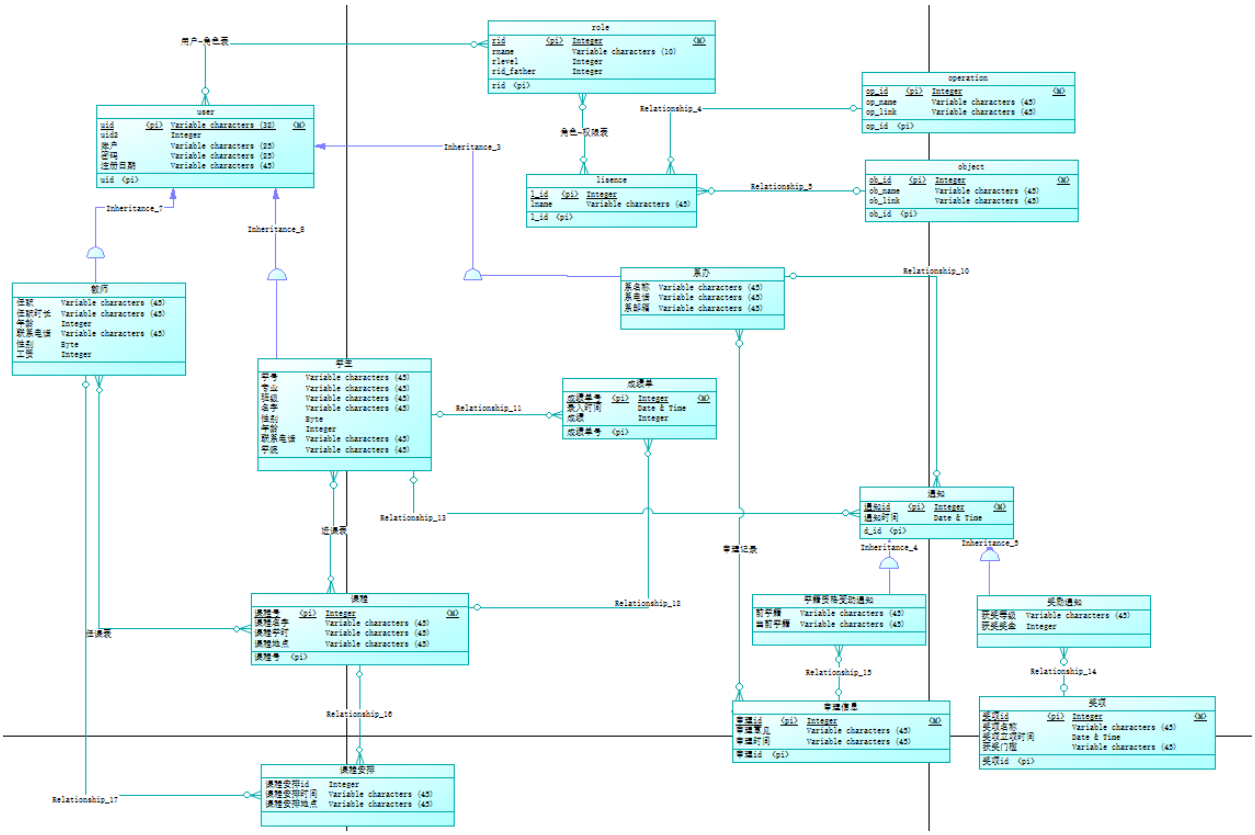


图 12: 数据库逻辑设计

2. 物理设计

下面为数据库物理设计总体图：

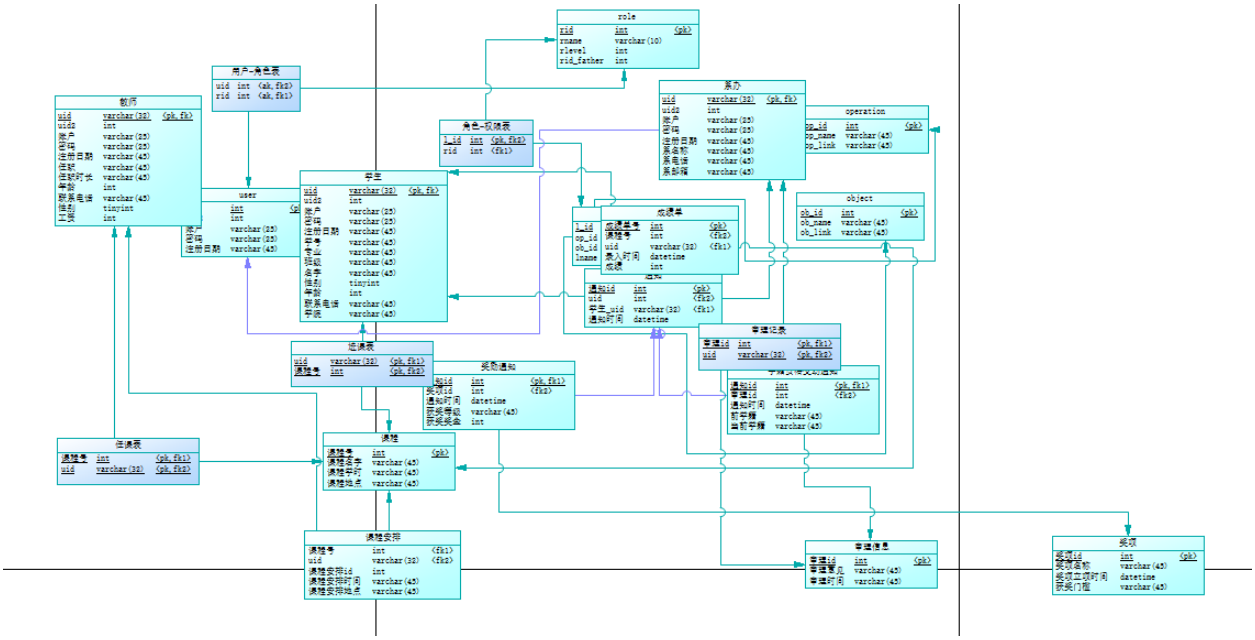


图 13: 数据库物理设计

九、 拓展部分

1. 软件体系架构图

软件体系架构图是一种图形化表示软件系统结构的工具。它用于展示软件系统中各个组件之间的关系、模块的划分以及数据流动的方式。软件体系架构图通常由不同的图形符号和线条组成，用于表示系统中的各个部分和它们之间的交互。

在软件体系架构图中，常见的表示元素包括：

模块/组件：表示系统中的功能模块或组件，通常使用矩形或圆角矩形表示。

接口：表示模块之间的通信接口，通常使用箭头或线条表示。

数据流/消息流：表示数据或消息在系统中的传递路径，通常使用箭头或线条表示。

层次结构：表示系统中的层次结构关系，通常使用水平分层或垂直分层的方式表示。

运行时组件：表示系统中的运行时组件，例如进程、线程、服务等，通常使用特定的符号表示。

软件体系架构图的目的是帮助软件开发人员和其他相关利益相关者更好地理解系统的结构和组成部分，以及它们之间的关系。通过使用体系架构图，团队成员可以更容易地进行系统设计、分析和交流，并确保系统的各个部分之间的一致性和协调性。

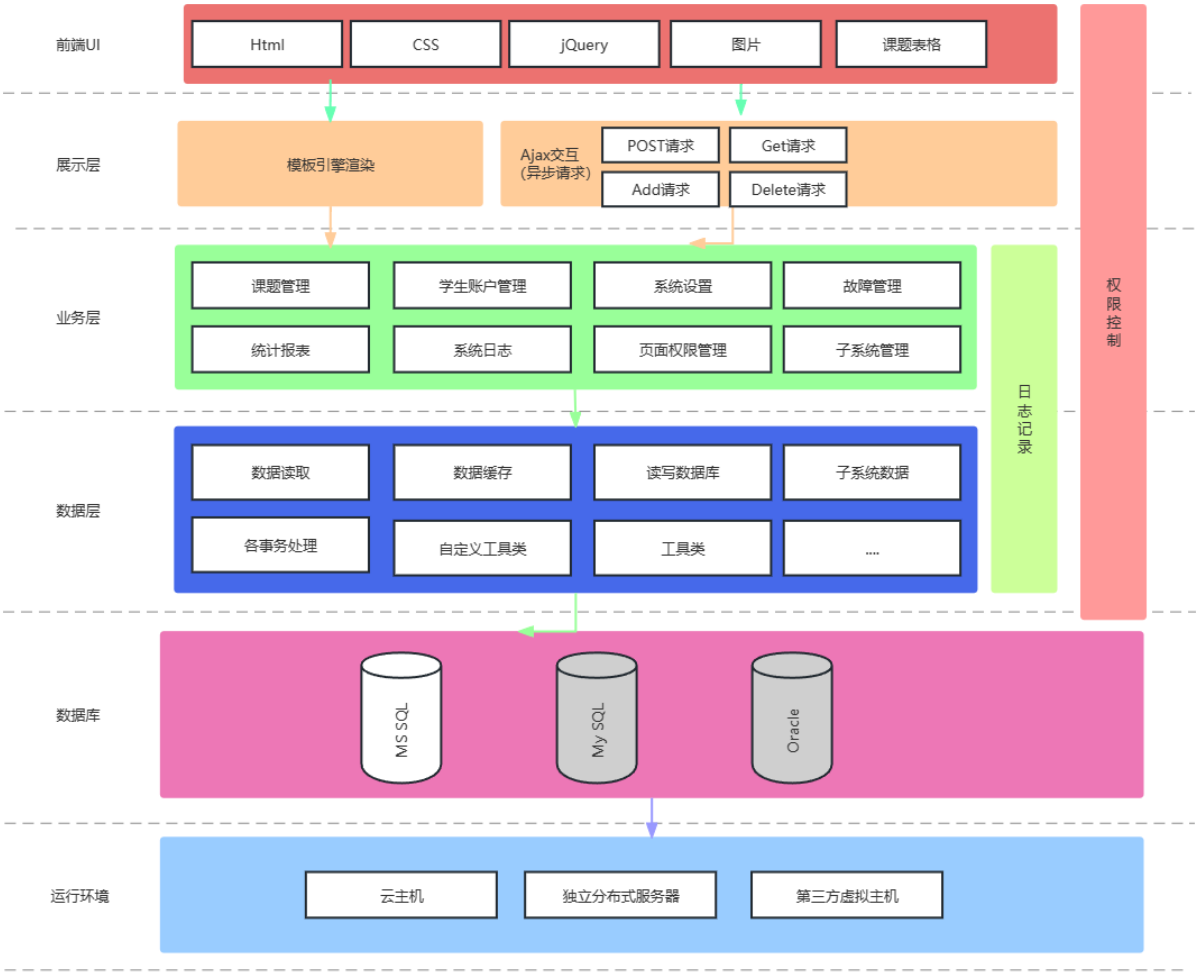


图 14: 系统体系结构设计

2. 流程图

下面提供部分功能的流程图。

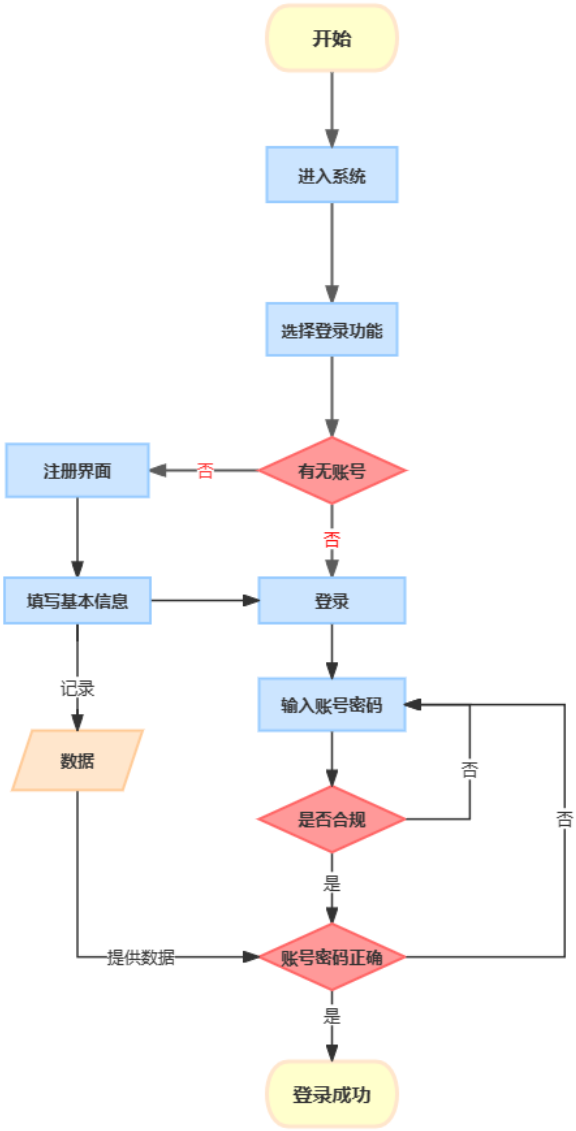


图 15: 登录流程图

我们假设各个成绩，资格，奖励统一抽象成数据，下面为管理数据流程图。

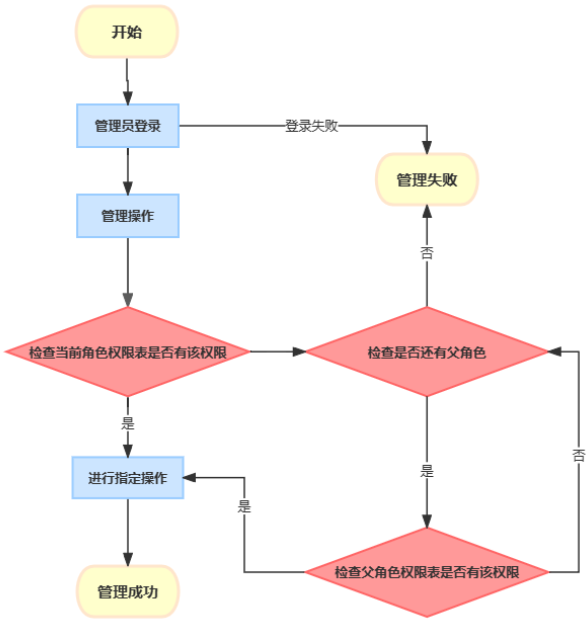


图 16: 管理数据流程图

3. 构件设计图

构件设计图是在软件开发过程中用于描述系统构件（Components）的图形化表示工具。构件是指在软件系统中具有独立功能和责任的模块或组件，它们可以是独立的软件模块、类、对象或其他系统实体。

下面给出部分功能的构件设计图：

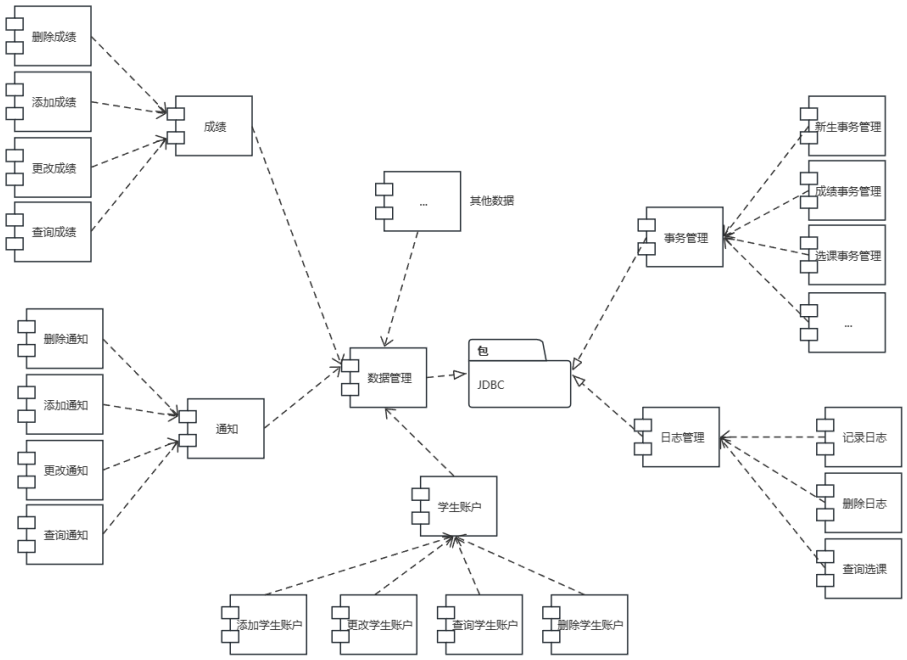


图 17: 构件设计图

十、 实验总结及心得

1. 实验总结

实验进行了如下工作：

(1) 学习了面向对象和需求分析

(2) 学习了 UML 各种图的概念

(3) 进行了领域模型的建立，绘制了分析类图

(4) 进行了用例模型的建立，绘制了各种用例图和编写了用例描述

(5) 进行了数据库设计，总体符合 RBAC3 设计规范，数据库做到了第二范式，存在部分冗余用来提升系统效率。

2. 实验心得

在本次实验中，我们使用了 UML 的建模方法将一个模型转换成等效的功能模型、动态模型、静态模型和数据库 ER 模型。通过这个过程，我对 UML 建模方法有了更深入的理解，并学会了如何将一个系统的需求和业务场景转化为可视化的模型。

首先，我开始了解系统的需求和业务场景，并根据这些信息开始创建功能模型，也就是 Use Case 图。在 Use Case 图中，我明确了系统的角色（Actor）和用例（Use Case），并展示了它们之间的交互关系。

接下来，我转向动态模型，使用活动图和分析时序图来描述系统中的活动流程和事件顺序。活动图展示了系统中的活动和动作，并展示它们之间的关系和顺序。而分析时序图则更加详细地描述了各个对象之间的交互和消息传递。

然后，我创建了静态模型，也就是分析类图。在分析类图中，我明确了系统中的类、属性和方法，并展示了它们之间的关系，如关联、继承和依赖等。

最后，我设计了数据库 ER 模型，用于表示系统中的数据存储和关系。ER 模型使用实体-关系图的形式展示了实体、属性和关系之间的联系，并帮助我们理解系统中的数据结构。

通过这个实验，我深刻体会到了不同类型的 UML 模型之间的关系。功能模型、动态模型、静态模型和数据库 ER 模型相互补充和关联，形成了一个完整的系统视图。这些模型之间的关系帮助我们更好地理解和描述系统的不同方面，从而更好地设计和开发软件系统。

此外，我还意识到在实际应用中，这些模型可能需要根据实际需求进行扩展和细化。每个模型都可以进一步拆分和详细化，以满足系统设计和开发的需要。因此，在实际项目中，我们需要根据具体情况决定模型的深度和精确度。

总而言之，本次实验使我对 UML 建模方法有了更深入的认识和理解。通过将一个模型转换成等效的功能模型、动态模型、静态模型和数据库 ER 模型，我学会了如何使用 UML 建模方法来描述和设计软件系统，从而更好地理解和满足系统需求。这对我今后的软件开发和系统设计工作将起到积极的指导作用。

参考文献

[1] 郑智红. 基于 UML 的网络学习系统的分析和建模 [J]. 科学技术创新,2021(21):100-101.

[2] Kenneth E. Kendall,Julie E. Kendall. [M]. 系统分析与设计, 第九版. 机械工业出版社, 2019.

[3] 吴建, 郑潮, 汪杰. UML 基础与 Rose 建模案例 [M]. 人民邮电出版社:, 201207.324.