大连理工大学城市学院

本科生毕业设计（论文）

学　　院：计算机工程学院

专　　业：软件工程

学　　生：高砚策

指导教师：张应博

完成日期：2020 年 6 月 3 日

大连理工大学城市学院本科生毕业设计（论文）

基于 SpringBoot 毕业设计过程管理系统分析设计与实现

总计　毕业设计（论文）　71　页
表格　9　个
插图　38　幅

摘 要

进入 21 世纪以来，在现代计算机产业的发展中，计算机软硬件成本不断下降，因此，电子化办公系统开始在全世界开始流行，各类信息化管理系统层出不穷，也得到了广泛的应用，各行各业领域都逐步进入了信息化时代，建设起了信息化的现代管理办公系统。在高校的校园环境和条件下，应用信息化教学管理系统等能够极大地提升高校教学的效率和办公管理效率，也能拉近师生之间的距离，使师生之间的交流和学习更加便利，学校管理部门也能更快更方便的掌握师生的思想、学习、生活方面的动向和进度，更加便于针对学生的个性化培养。目前我国高校的信息化建设过程中，各个高校优先开发建设的信息化系统是毕业设计过程管理系统，这类系统可以帮助高校对毕业年级学生的学习进度和毕业设计（论文）完成情况进行很好的统计管理工作。

本毕业设计过程管理系统就是在这一背景下研究出来的课题，本系统主要使用的技术是基于 Java 语言的 Spring Boot 框架结合 MyBatis 框架进行的开发，数据库使用的是 MySQL5.7 版本，前端使用的技术是 Thymeleaf 渲染模板结合 HTML5、CSS3、JavaScript、jQuery、Bootstrap 等主流前端技术开发。本系统实现了对毕业年级学生、毕业设计（论文）指导教师、学校管理领导三类用户的管理，实现了对高校毕业设计（论文）过程的信息化、网络化、电子化管理工作。

本系统既能减轻学校教学管理部门的工作难度，也能使毕业年级学生对自己的毕业论文过程进度能够更好的了解和把握，也可以帮助毕业论文指导教师减轻自己的工作压力，更方便的了解自己指导的学生的进度。

关键词：Spring Boot；MyBatis；毕业设计过程管理

Abstract

Since the beginning of the 21st century, in the development of the modern computer industry, the cost of computer hardware and software has been declining. Therefore, electronic office systems have become popular all over the world. Various information management systems have emerged in an endless stream and have been widely used. The industry has gradually entered the information age, and built a modern information management office system. In the campus environment and conditions of colleges and universities, the application of information-based teaching management systems can greatly improve the efficiency of teaching and office management in colleges and universities, and can also shorten the distance between teachers and students, making communication and learning between teachers and students more Convenience, the school management department can also grasp the trends and progress of teachers and students' thoughts, learning and life faster and more conveniently, which is more convenient for personalized training for students. At present, in the process of informatization construction in colleges and universities in our country, the information system that every university preferentially develops and constructs is the graduation design process management system. This type of system can help colleges and universities to do a good job in the progress of graduation grades and the completion of graduation design (thesis) Statistics management.

This graduation design process management system is a subject researched under this background. The main technology used in this system is the development based on the Java language Spring Boot framework combined with the MyBatis framework. The database uses the MySQL 5.7 version and the front end The technology is Thymeleaf rendering template combined with HTML5, CSS3, JavaScript, jQuery, Bootstrap and other

mainstream front-end technology development. This system realizes the management of three types of users of graduation grade students, graduation design (thesis) instructors, and school management leaders, and realizes the information, network and electronic management of the college graduation design (thesis) process.

This system can not only reduce the difficulty of the work of the school's teaching management department, but also enable the graduate students to better understand and grasp the progress of their graduation thesis process, and also help the graduation thesis instructors to reduce their work pressure Understand the progress of the students you guide.

Key words: Spring Boot; MyBatis; Graduation design process management

目 录

第一章 引言

1.1 项目的来源及背景

每年的大学应届毕业生数量都十分庞大，并且在毕业年级有很多任务和事情需要处理，例如毕业生需要完成毕业设计（论文）的撰写任务，毕业论文的指导教师要完成对自己的学生的指导工作和评审把关工作，学校教学管理部门老师要完成对毕业年级学生的毕业资质审核、毕业论文进度控制等管理工作，各个院系专业的教学主任领导需要完成对各自院系的师生的进度管理。在没有一个成熟的毕业设计过程管理系统之前，上述工作是十分繁琐和复杂的，而且容易在工作中出现错误，给毕业生和学校工作生活造成困扰。随着工程化培养工作的推进及教学多元化的发展，新形势下传统的管理方式已不能满足毕业设计的管理要求，因此急需一个基于信息化管理的毕业设计过程管理系统。[2]

近年来，随着信息化在我国教育行业的不断普及，各大高校纷纷基于自己学校的实际业务，开发建设了一系列的信息化管理系统，毕业设计过程管理系统就是其中的一类高校信息化管理系统。合理的使用毕业设计过程管理系统，可以大大减少毕业年级学生和教师的工作压力，提升师生的效率，节约了大量的时间，也降低了工作出错的概率。

基于此，针对我国各大高校的信息化建设现状和特点，提出了基于 Spring Boot 的毕业设计过程管理系统。

1.2 行业的现状

教育信息化这个概念，是伴随着计算机行业的发展而提出的，最早是在 20 世纪 80 年代末美国的“兴建信息高速公路”计划中提出的，旨在通过现代化的高科技来改变传统教育的方式，实现 21 世纪教育改革。在国外，几乎所有的高校已经在 21 世纪初期完成了信息化管理，基本做到了教育信息化、电子化，助力学校发展。

在我国，教育信息化的概念是在 20 世纪末 21 世纪初提出来的，此

时我国的计算机网络技术已经进入了高速发展，在社会上已经开始普及，信息技术的发展也会开始影响社会的发展。教育信息化的概念自此横空出世，作为 21 世纪中国教育界的重要发展理念。

在教育信息化的部署发展中，高校的信息化建设是尤为重要的，并且也是十分迫切的。在大学的学习和生活中，学生和老师离不开科学技术和互联网，大量的工作也需要进行改革，化繁为简，减轻师生的负担和相关部门的工作压力。至此，高校信息化建设的呼声越来越高，也是我国高校面向现代化和进军世界一流高校的必要条件。高校信息化是教育信息化的具体方案之一，目的是采用当今世界上主流的信息技术手段和方法构建一个数字化的信息化校园系统，完成对学校各个方面的资源整合，从而提高管理的效率，提高高校教学质量，提升高校综合实力水平。

在高校信息化建设过程中，首当其冲需要建设的，就是教务类综合管理系统平台相关软件。这类软件既为全面了解学生的毕业设计综合信息提供了方便，使得查询统计更为准确便捷，又可以用电子档案逐步代替传统的纸质档案，实现管理手段现代化。[9]而对于毕业年级的学生来说，由于面临着毕业求职或升学压力，时间和效率变得更加重要，因此，毕业年级师生对于一个信息化网络化的管理平台系统的需求就显得更加急迫了。本毕业设计过程管理系统就是在此背景下提出的。

1.3 系统实施方案和技术手段

1.3.1 系统实施方案

本系统主体是利用 Java 语言进行构建开发的，使用 MVC 模式进行设计开发，后端部分使用了当前最流行的 Spring Boot 框架技术进行开发，持久层使用 MyBatis 框架；前端使用的技术是 Thymeleaf 技术结合 HTML5 技术、CSS3 技术、jQuery 技术、Bootstrap 技术进行构建开发的。

数据库方面，根据前期的调研，综合了性能和成本的考虑，本系统选择了免费的 MySQL 数据库，版本选择的是当前市面上主流的

MySQL5.7 版，该版本数据库性能优异，功能稳定，兼容性好。在数据库的表结构设计方面，遵循了规范化设计，符合数据库三范式的设计模式，避免在对数据库操作时出现异常，影响系统运行和系统功能稳定性。

1.3.2 技术实现手段

在需求方面，通过在前期搜集大量的资料和数据进行分析，通过互联网和图书馆大量相关资料和书籍等数据对需求进行详尽的分析，确定问题关键所在，咨询相关人士或老师了解需求详情。根据了解和分析的结果，确定实体和实体之间的关系，每个实体的具体属性，制作逻辑模型和物理模型，并以此来制作数据库原型，创建出一个符合业务规范要求的数据库。

在设计方面，本系统是基于 B/S 模式进行设计开发的，在用户客户端只需安装使用主流浏览器来访问使用系统，开发时只需负责开发服务端代码即可，前端页面做好兼容适配工作。MVC 为模型-视图-控制器的缩写，特点是开发时可以分为 Model-View-Controller 三层模块进行开发，有利于分块开发，后期系统升级时只需更改需要升级的代码部分（前端界面或后端），无需全部重写，降低开发人员的工作量，提升开发效率。

在技术方面，整个项目使用 Apache Maven 进行管理和构建，本系统服务端主要使用 Spring Boot 框架进行开发；持久层框架部分，选用当前主流的 MyBatis 框架进行开发，简化与数据库交互的代码。前端界面部分，使用的是 Thymeleaf 模板引擎结合 HTML5、CSS3、jQuery、Bootstrap 等相关前端主流开发设计技术进行开发。Thymeleaf 相较于传统的 jsp 技术，主要是大大提升了渲染性能，并且支持静态建模，有利于当前市面主流的前后端分离开发技术方法，向后兼容性好。Web 服务器部分使用的是 Spring Boot 自带的 Web 容器进行发布，减少代码开发量和后期调试工作。数据库服务器选用的是市面上主流的 MySQL5.7 版本。

第二章 毕业设计过程管理系统需求分析

2.1 系统功能分析

毕业设计过程管理系统的主要功能有：毕业生信息管理、指导教师信息管理、管理员信息管理、毕业设计（论文）过程管理四大基本功能。不同用户拥有的功能访问权限不同，通过登陆时进行判断。当使用毕业生账号登陆后，毕业生可以访问的功能主要有：查看公告、选题、上传开题报告、中期检查、论文查重检测、论文终稿上传等，其中，未选题的学生无法完成后续功能操作。指导教师可以访问的功能主要有：查看学校发布公告、上报选题、查看开题报告、查看中期检查报告、查重报告管理、查看论文终稿、答辩安排等功能。管理员可访问的功能主要有：公告管理、毕业生信息维护、指导教师信息维护、专业主任维护、毕业论文评审等功能。系统整体功能图如图 2-1 所示。

图 2-1 系统整体功能图

2.2 用户分析

毕业设计过程管理系统的主要用户有三类：第一类是毕业年级学生、第二类是毕业设计（论文）指导教师、第三类是具有相关管理权限的老师。其中具有管理权限的用户有全局超级管理员和院系（专业）负责主任等。这三类用户登陆后可以访问的功能和权限不尽相同，通过代码实

现权限控制，避免越权访问破坏系统。毕业设计过程管理系统的用户类型如图 2-2 所示。

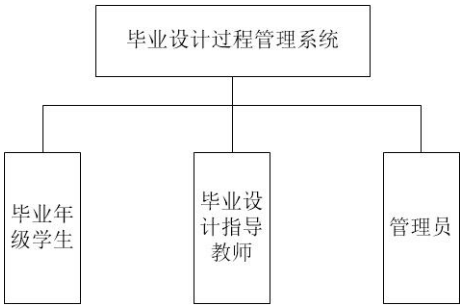

图 2-2 用户类型图

本毕业设计过程管理系统的用户在访问系统首页时，如果检测到当前未登录则会自动跳转到统一登录页面，在页面上，要登录的用户选择自己的用户角色，然后输入用户名和密码，点击登录按钮，如果用户名和密码正确，将会跳转到用户首页公告栏。用户在登陆后，没有权限访问的功能和页面将不会展示在导航栏，无权限情况下也无法访问。

登录成功的三类用户，会根据不同的角色（role）跳转到不同的首页展示不同的功能。

2.3 用例分析

根据对毕业设计过程管理系统的用户分析后，为了将系统各个模块的功能进行分析设计，需要进行详细的系统用例分析。

2.3.1 毕业生用例分析

毕业生使用自己的账号（用户名）和密码，在登陆页面选择登录角色为“毕业生”，输入用户名和密码后，页面会给出提示，用户名密码都正确即可登录成功，跳转到毕业生公告栏，即为毕业生登录后首页，可以在此页面查看学校或指导教师给毕业生发布的通知公告。

在导航栏菜单处，学生可以选择“选题”按钮，点击即可进入选题管理页面。在页面上会显示毕业生的基本信息和选题情况，如果该生目前尚未选题，页面会给出明显提示并引导点击进入选题系统界面。进入

选题界面后，毕业生可以选择自己所在专业内有空闲课题的导师进行选题，选择完成后进行提交操作，系统会给出提示是否选题成功。如果学生无特殊情况原因下错过规定的选题时间，按学校相关规定，将自动视为该学生放弃毕业论文这一环节，自动申请延长毕业时间。[3]

选题完成后，学生可以进入开题报告上传功能模块，在这里，毕业生可以按照指导教师的要求和安排，在规定的时间内按照要求上传自己完成的开题报告等相关资料，等待指导教师进行审阅。

在开题阶段结束后，按照学校院系和指导教师的安排要求，学生可以进入导航菜单中的“中期检查”模块，上传自己的中期检查报告和相关资料，等待导师审阅。

在中期检查阶段结束后，按照学校要求，学生可以在规定时间内，将自己的毕业论文进行查重检测，在规定截止日期前通过导航菜单访问“查重报告管理”模块，将最终稿的查重报告上传到毕业设计过程管理系统中。

在完成查重检测后，学生按规定和要求在截止日期前将自己课题的毕业论文终稿通过毕业设计过程管理系统的“论文管理”模块中，等待接受评审组老师审阅。

毕业生的用例图如图 2-3 所示。

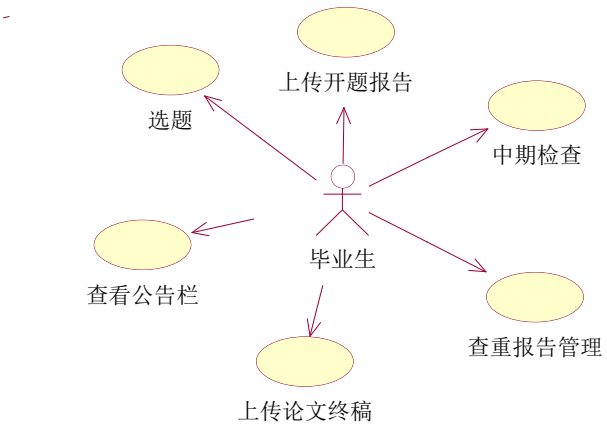

图 2-3 毕业生用例图

依据上图对毕业生用例的划分，撰写用例文档。毕业生的用例文档如下：

1、毕业生登录系统用例

用例名：登录。

参与者：毕业生。

前置条件：用户访问系统部署的网站地址登录入口。

主事件流：

（1）毕业生选择登录角色为“毕业生”，然后输入用户名和密码。

（2）如果用户名或密码错误，则弹出“用户名或密码错误”提示窗口，点击确定后重新核对用户名密码重新进行登录操作。

（3）如果用户名密码输入正确，则登录成功，自动跳转进入毕业生系统主界面。

后置条件：成功登录到系统。

2、毕业生查看公告用例

用例名：查看发布给毕业生的公告。

参与者：毕业生。

前置条件：用户成功登录系统。

主事件流：

（1）毕业生登录系统后访问首页。

（2）在登录成功后访问首页，上面的公告栏就是展示给毕业生的所有公告信息。

后置条件：无。

3、毕业生选题用例

用例名：毕业生选题。

参与者：毕业生。

前置条件：毕业生已经成功登录系统。

主事件流：

（1）毕业生选择菜单导航访问选题界面。

（2）毕业生查看界面展示是否选题，未选题则可以进入选题界面，

已选题则只能查看自己的选题结果。

（3）未选题的毕业生点击选题按钮进入选题界面，可以选择本院系的指导老师及课题。

（4）选择完成后点击提交，系统检查该课题是否可选，并返回成功与否结果提示。

后置条件：毕业生选题成功。

4、毕业生上传开题报告用例

用例名：毕业生上传开题报告。

参与者：毕业生。

前置条件：毕业生已经成功登录系统。

主事件流：

（1）毕业生选择菜单导航访问上交开题报告界面。

（2）系统校验该用户是否完成选题，未完成选题则弹出提示并跳转到选题界面，已完成选题的用户则进入下一步。

（3）根据系统提示上传自己的电子版开题报告到系统中。

（4）系统会给出提示是否完成上传。

后置条件：毕业生开题报告上传成功。

5、毕业生上传中期检查报告用例

用例名：毕业生上传中期检查报告。

参与者：毕业生。

前置条件：毕业生已经成功登录系统。

主事件流：

（1）毕业生选择菜单导航访问中期检查报告管理界面。

（2）系统校验该用户是否完成开题报告，未完成则弹出提示要求上交开题报告，已完成的用户则进入下一步。

（3）根据系统提示上传自己的电子版中期检查报告到系统中。

（4）系统将对是否完成上传给出提示。

后置条件：毕业生中期检查报告上传成功。

6、毕业生上传论文查重报告用例

用例名：毕业生上传论文查重报告。

参与者：毕业生。

前置条件：毕业生已经成功登录系统。

主事件流：

（1）毕业生选择菜单导航访问论文查重报告管理界面。

（2）系统校验该用户是否完成中期检查，未完成则弹出提示要求上交中期检查报告，已完成的用户则进入下一步。

（3）根据系统提示选择文件目录，上传自己查重结束后的报告单到系统中。

（4）系统给出提示是否上传成功查重报告。

后置条件：毕业生论文查重报告上传成功。

7、毕业生上交毕业论文终稿功能用例

用例名：毕业生上交毕业论文终稿功能。

参与者：毕业生。

前置条件：毕业生已经成功登录系统。

主事件流：

（1）毕业生选择菜单导航访问上交毕业论文终稿功能界面。

（2）系统校验该用户是否完成查重，未完成则弹出提示要求上交查重报告，已完成的用户则进入下一步。

（3）根据系统提示选择文件目录，上传自己查重结束后的毕业论文终稿到系统中。

（4）系统给出提示是否提交成功。

后置条件：毕业生毕业论文终稿提交成功。

2.3.2 指导教师用例分析

指导教师凭自己的用户名和密码进入系统登录页面，密码正确，则会自动跳转到指导教师公告栏首页，可以查看学校发布的公告。指导教师登陆后可以访问的功能有：

1、　　上报课题

2、 开题过程管理

3、 中期检查管理

4、 查重报告管理

5、 毕业过程管理

指导教师可以在登陆后通过导航菜单进入“上报课题”功能页面，首页是该教师目前名下的课题列表，教师可以点击页面上的上报新课题按钮添加新的课题供毕业年级学生选择。

指导教师可以进入“开题管理”模块，查看自己名下所带的毕业生的课题开题报告完成情况，审阅学生上传的开题报告，决定是否同意开题。

通过“中期检查管理”功能，指导教师可以在规定的时间内检查毕业生的毕业设计（论文）研究进度，检查提交的中期检查报告。

通过“查重报告管理”模块，指导教师可以查看毕业生所提交的毕业论文的查重率是否符合学校及国家有关规定要求，并能了解到毕业论文的质量。

通过“毕业过程管理”功能，指导教师可以查看下载所有选择自己名下课题的学生上传的毕业论文，并检查论文内容和质量，决定该生是否可以参加答辩等功能。

指导教师的用例图如 2-4 所示。

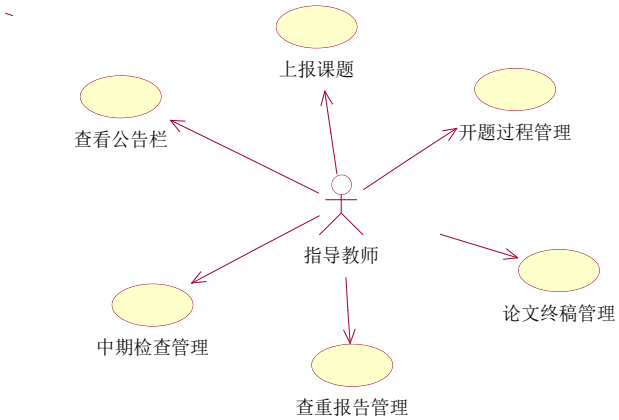

图 2-4 指导教师用例图

依据上图对指导教师用例的划分，撰写用例文档。指导教师的用例文档如下：

1、指导教师登录系统用例

用例名：登录。

参与者：指导教师。

前置条件：用户访问系统部署的网站地址登录入口。

主事件流：

（1）毕业生选择登录角色为“指导教师”，然后输入用户名和密码。

（2）如果用户名或密码错误，则弹出“用户名或密码错误”提示窗口，点击确定后重新核对用户名密码重新进行登录操作。

（3）如果用户名密码输入正确，则登录成功，自动跳转进入指导教师系统主界面。

后置条件：成功登录到系统。

2、指导教师查看公告用例

用例名：查看发布给指导教师的公告。

参与者：指导教师。

前置条件：用户成功登录系统。

主事件流：

（1）指导教师登录系统后访问首页。

（2）在登录成功后访问首页，上面的公告栏就是展示给指导教师的所有公告信息。

后置条件：无。

3、指导教师上报课题用例

用例名：指导教师上报课题。

参与者：指导教师。

前置条件：指导教师已经成功登录系统。

主事件流：

（1）指导教师选择菜单导航访问上报选题系统界面。

（2）指导教师查看界面展示自己名下的所有课题，也可以查看课题被学生选择情况。

（3）指导教师也可以点击上报新选题功能上报新的选题。

（4）填写课题内容完成后点击提交，返回成功与否结果提示。

后置条件：上报选题成功。

4、指导教师管理开题报告用例

用例名：指导教师管理开题报告。

参与者：指导教师。

前置条件：指导教师已经成功登录系统。

主事件流：

（1）指导教师选择菜单导航访问开题报告管理界面。

（2）指导教师可以查看自己名下负责的课题和学生对应的开题报告。

（3）指导教师可以选择自己想要查看的开题报告进行下载查阅操作。

后置条件：无。

5、指导教师管理中期检查报告用例

用例名：指导教师管理中期检查报告。

参与者：指导教师。

前置条件：指导教师已经成功登录系统。

主事件流：

（1）指导教师选择菜单导航访问中期检查报告管理界面。

（2）指导教师可以查看自己名下负责的课题和学生对应的中期检查报告。

（3）指导教师可以选择自己想要查看的中期检查报告进行下载查阅操作

后置条件：无。

6、指导教师管理论文查重报告用例

用例名：指导教师管理论文查重报告。

参与者：指导教师。

前置条件：指导教师已经成功登录系统。

主事件流：

（1）指导教师选择菜单导航访问论文查重报告管理界面。

（2）指导教师可以查看自己名下负责的课题和学生对应的论文查重报告。

（3）指导教师可以选择自己想要查看的论文查重报告进行下载查阅操作

后置条件：无。

7、指导教师毕业论文终稿管理功能用例

用例名：指导教师毕业论文终稿管理功能。

参与者：指导教师。

前置条件：指导教师已经成功登录系统。

主事件流：

（1）指导教师选择菜单导航访问查看毕业论文终稿管理功能界面。

（2）指导教师可以查看和管理自己名下负责的课题和学生对应的毕业论文终稿内容。

（3）指导教师可以选择自己想要查看的毕业论文终稿进行下载查阅操作

后置条件：无。

2.3.3 管理员用例分析

在登录界面，管理员用户选择使用“管理员”角色登录，在用户名和密码栏输入正确的用户名和密码，登录成功后进入系统首页。在本系统中，管理员角色包括院系教务管理人员和各院系毕业年级负责人，二者权限略有不同，教务管理员可以访问并查看所有数据，而院系负责人只能查看和评审本院系的学生毕业论文。系统首页包含公告栏模块，可以查看、发布公告，并可以管理自己发布的公告。

在导航栏，有四大功能模块，管理员用户可以根据需求进入不同的

功能模块进行操作。

1、 毕业生管理模块：管理人员可以在该模块下录入毕业生信息并创建账号、查看毕业生资料、修改毕业生资料、删除毕业生信息、修改毕业生登录密码等功能。

2、 指导教师管理模块：管理人员可以在该模块下添加指导教师信息并创建账号、查看指导教师信息、删除指导教师信息、修改登录密码等功能。

3、 毕业设计过程管理模块：管理人员可以在该模块下为某指导教师名下添加课题、查看课题选题情况、修改选题信息、管理开题报告、管理中期检查报告、管理查重报告、管理毕业论文终稿、审阅毕业论文等功能。

4、 管理员管理：本模块可以由学校教务负责管理人员使用，仿负责添加或删除院系的管理员或负责人权限。

5、 公告板管理：管理人员可以在此发布、查看、已经发布的公告，可以删除由登录账号发布的公告。

管理员的用例图如图 2-5 所示。

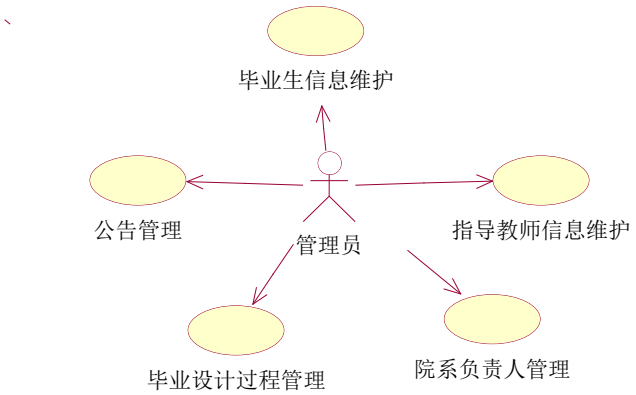

图 2-5 管理员用例图

依据上图对管理员用例的划分，撰写用例文档。管理员的用例文档如下：

1、管理员登录系统用例

用例名：登录。

参与者：管理员。

前置条件：用户访问系统部署的网站地址登录入口。

主事件流：

（1）毕业生选择登录角色为“管理员”，然后输入用户名和密码。

（2）如果用户名或密码错误，则弹出“用户名或密码错误”提示窗口，点击确定后重新核对用户名密码重新进行登录操作。

（3）如果用户名密码输入正确，则登录成功，自动跳转进入管理员系统主界面。

后置条件：管理员成功登录到系统。

2、管理员发布公告用例

用例名：管理员发布公告。

参与者：管理员。

前置条件：用户成功登录系统。

主事件流：

（1）管理员登录系统后访问首页。

（2）在登录成功后访问首页，进入公告管理模块，选择发布新公告。

（3）输入想要发布的公告信息详情。

后置条件：公告发布成功。

3、管理员添加毕业生信息用例

用例名：管理员添加毕业生信息。

参与者：管理员。

前置条件：用户成功登录系统。

主事件流：

（1）管理员登录系统后访问首页。

（2）在登录成功后访问首页，在左侧导航菜单中选择毕业生信息管理。

（3）点击添加毕业生信息按钮。

（4）录入毕业生的所有信息，设置毕业生登录系统密码。

后置条件：添加毕业生信息成功。

4、管理员添加指导教师信息用例

用例名：管理员添加指导教师信息。

参与者：管理员。

前置条件：用户成功登录系统。

主事件流：

（1）管理员登录系统后访问首页。

（2）在登录成功后访问首页，在左侧导航菜单中选择指导教师信息管理。

（3）点击添加指导教师信息按钮。

（4）录入指导教师的所有信息，设置指导教师登录系统密码。

后置条件：添加指导教师信息成功。

5、管理员添加课题详情用例

用例名：管理员添加课题详情。

参与者：管理员。

前置条件：用户成功登录系统。

主事件流：

（1）管理员登录系统后访问首页。

（2）在登录成功后访问首页，在左侧导航菜单中选择毕业设计过程管理。

（3）点击添加课题信息按钮。

（4）录入课题的所有相关信息。

后置条件：添加课题信息成功。

6、审阅毕业论文用例

用例名：审阅毕业论文。

参与者：管理员。

前置条件：用户成功登录系统。

主事件流：

（1）管理员登录系统后访问首页。

（2）在登录成功后访问首页，在左侧导航菜单中选择毕业设计过程管理。

（3）点击审阅毕业论文按钮。

（4）查看毕业生上交的论文终稿，并根据情况进行打分和评语。

后置条件：审阅论文成功。

2.4 可扩展性分析

在系统完成后，随着时间的推移，可能需要添加或者调整一些新的功能，本系统是采用 Spring Boot 框架技术开发的，并采用了 MVC 架构，利于后续的扩展升级，且不影响原有功能。

2.5 可行性分析

通过在学校进行调研以及查阅大量资料，本系统具备以下几点可行性。

2.5.1 技术可行性

本系统采用 Java 语言开发，使用了当前市场上流行的 Spring Boot 框架进行开发，每一个 Spring Boot App 都是独立存在的个体应用，只需要基本的 Java 运行环境即可满足部署要求，Spring Boot 内部嵌入了 Tomcat，从而无需部署 Web 容器，易于开发和部署。[4]ORM 框架部分，本系统选择使用 MyBatis 框架，易于上手学习，可以自定义 SQL 语句，便于优化查询性能，并且减少代码量，减轻开发工作压力。整个项目使用 Maven 构建，管理所有项目需要的 Jar 包，利于统一管理也便于打包发布项目。前端采用 Thymeleaf 和 HTML5 相关技术，便于开发测试，也相较于传统的 JSP 技术大大的提升了渲染速度。Thymeleaf 的最大优点主要是其能够与 SpringMVC 实现良好的融合，其次，该模板引擎相比其他引擎方式来，可实现在浏览器中直接显示，也是一种常见的模板系数。[6]数据库方面选用 MySQL 数据库，该数据库具有免费、速度快、性能优越、稳定性好、兼容性强等显著特点。由此看来，本系统在技术开发方

面完全可行。

2.5.2 法律可行性

本系统从建立课题开始，到设计、编码、开发、测试等环节，均为本人独立原创设计开发，设计开发期间所查阅的所有资料均为公开的资料，无任何违法违规侵权行为。因此，从法律方面，本系统完全可行。

2.6 系统非功能性需求分析

2.6.1 安全性

本毕业设计过程管理系统设计初衷就是发布在网络中供所有用户使用的，在互联网上开放的服务，就有被攻击的可能，要做好安全防护，确保万无一失。本系统在登陆时有用户角色权限校验功能，避免非法访问，导致非法账户权限提升，造成对系统的破坏。另外，项目连接数据库服务时要避免使用最高权限账户（root 用户）连接数据库服务器，防止被攻击破坏服务器中的数据信息。在防 SQL 注入攻击方面，由于本系统 DAO 层框架使用 MyBatis 框架进行开发，从源代码上避免了被 SQL 注入的风险。系统部署在网络中后，要注意对部署的服务器进行安全加固防护，关闭不必要的端口和服务，避免被非法攻击破坏。

2.6.2 稳定性

本系统主要面向高校毕业年级全体师生使用，用户量较为庞大，尤其在选题开题期间或其他学校规定的时间节点，一定会有大批量的用户访问系统进行操作。在开发时，要做到一定的并发量，避免在关键时刻出现系统异常或崩溃的情况。

2.6.3 运行环境需求

本系统是使用当前市面上主流的 Java 企业级开发技术进行开发的，对于开发及运行环境有一定的要求，保障系统的流畅运行和功能稳定。

1、 开发环境：

IntelliJ IDEA x64 版

2、 硬件需求：

CPU：Intel Core i5-4590

内存：8GB

显卡：GTX 750Ti

3、 软件需求：

操作系统：Windows 7/8/8.1/10 或 Windows Server 2008 R2/2012/2016

系统运行依赖环境：JDK 1.8、MySQL 5.7

第三章 系统概要设计

3.1 系统概要

本毕业设计过程管理系统是使用 Java 语言进行开发的，后端服务端使用了主流的 Spring Boot 框架结合市面上应用广泛的 DAO 层框架 MyBatis 框架，数据库层面使用了 MySQL5.7 版本。前端使用了 Thymeleaf 模板引擎结合 HTML5、CSS3、jQuery、Bootstrap 等优秀的前端开发技术进行构建开发。

本毕业设计选题管理系统主要包含五大功能模块：公告管理、毕业生信息管理、指导教师信息管理、管理员信息管理、毕业设计过程管理。其中，公告管理功能仅限管理员角色用户登录后进行编辑或发布操作，其他角色用户只能查看已发布的公告。在用户登录角色上，本系统分为以下三种角色：管理员（院系负责人）、毕业生、毕业设计指导教师。三种角色的用户在登录界面输入正确的用户名密码成功登录后，系统会根据不同用户的角色来展现不同的功能或内容，不具备某功能访问权限的用户非法访问会被拒绝，避免系统被攻击或破坏。

当毕业生登录系统后，会跳转到学生公告栏首页，查看学校发布的最新动态公告，也可以在导航栏进行修改密码或退出登录操作。在导航菜单栏中，可以访问的功能有选题、上交开题报告、上交中期检查报告、上交查重报告、上交毕业论文终稿等几大功能。在未选题前，其他上交功能均不可用，会被引导到选题系统页面。在选题系统页面，可以查看到自己当前的选题情况。在选题成功后可以在选题系统首页看到选题结果和指导教师姓名。选题完成后可以在学校或老师规定的时间内访问菜单中的上交开题报告、上交中期检查报告、上交查重结果报告、上交毕业论文终稿等功能，部分功能根据规定只能提交一次，如需更改需要联系指导教师或者教务管理员老师。

指导教师在登录界面输入用户名密码登录毕业设计过程管理系统后，会跳转到系统首页，在首页可以查看发布给指导教师的公告信息。

在导航栏菜单中，指导教师可以访问以下功能：上报课题、开题过程管理、中期检查管理、查重报告管理、毕业设计过程管理。其中，指导教师点击进入上报课题功能，进入课题管理模块首页，可以查看自己名下的所有课题和课题被毕业生选择的情况，指导教师可以点击上报按钮上报自己名下的新的课题内容供毕业生选题。在开题过程管理模块中，指导教师可以查看选择自己名下课题的学生上交的开题报告，并进行审阅。在中期检查管理模块中，指导教师可以查阅自己名下的学生提交的中期检查报告，并核查学生的论文进度情况。在查重报告管理模块，指导教师可以查看名下学生论文的查重报告，检查论文重复率是否符合学校以及相关规定的要求。在毕业设计过程管理模块，指导教师可以查看学生上传的毕业论文终稿，并进行查阅，检查论文质量，并决定是否同意该学生的毕业论文进入毕业论文评阅组由院系评审组负责人老师进行审阅打分。

管理员用户可以凭借自己的用户名和密码登录系统，进入系统后展示系统首页，即公告管理模块，可以发布公告，也可以查看所有已发布的公告，并且可以删除自己发布的公告。管理员可以访问的功能模块有：毕业生管理、指导教师管理、管理员信息管理、毕业设计过程管理、毕业论文评审。毕业生管理模块，管理员可以添加毕业生信息、修改毕业生信息、删除毕业生、重置毕业生登录密码、查询毕业生资料等功能。指导教师管理模块，管理员可以查询指导教师信息、添加指导教师信息、修改指导教师信息、删除指导教师信息、重置指导教师登录密码等功能。在管理员管理模块，可以增加、查看、修改、删除管理员信息。管理员可以访问毕业设计过程管理模块，查看毕业论文完成进度，以及进入毕业论文评审功能，为进入答辩的同学的毕业论文打分写评语，实现对毕业过程的管理功能。

3.2 系统体系结构设计

本系统是基于 MVC 模式采用面向对象进行开发的基于 B/S 架构进

行设计实现的，MVC 即 Model、View、Controller 的缩写，代表着这个模式的最大特点即分为模型层、视图层、控制层进行设计开发。使用 MVC 模式开发的最终目的是达成一种动态程序的设计开发，简化后期对程序的改动和拓展，因此可以实现模块式的开发系统，有利于开发过程降低工作量减少错误。

本系统是基于 MVC 模式进行设计设计实现的，系统整体分为四大模块部分，分别为毕业生模块、指导教师模块、管理员模块、毕业设计过程管理模块，根据不同的用户权限可以控制访问不同的功能。本系统视图层部分使用 Thymeleaf 渲染模板结合 HTML5、CSS3、jQuery、Bootstrap 等前端优秀技术框架进行开发。系统的控制层基于 Spring 框架特点进行开发，代码实现了 RESTful 接口规范。

本系统基于此使用了 MVC 模式分层进行了开发实现，做到了降低代码耦合度，使程序结构更加直观化。

3.3 系统总体交互流程

本系统的开发采用了当前市场上主流的 Spring Boot 框架，继承了 Spring 框架的所有优点，而 Spring Boot 的最大特点之一是约定优于配置。约定优于配置的策略减少软件开发人员需要做决定的数量，获得简单的好处的同时，而又不失灵活性。[7]DAO 层框架使用了国际主流的 ORM 框架 MyBatis 框架，前端使用 Thymeleaf 模板引擎结合 HTML5 等流行前端技术进行开发。系统前台与后台的交互是通过前后端 ajax 交互的，前端通过 ajax 技术向后台发送 json 格式的数据，后台接收数据并进行校验，返回操作结果。页面请求地址通过 SpringBoot 框架内置的 DispatcherServlet()方法获取请求的资源地址，并进行分发数据。系统的资源地址 URL 设计采用了国际通用的 RESTful 规范，便于识别区分，也便于后续更新升级系统。

3.4 系统数据库表需求分析设计

本毕业设计过程管理系统是面向高校毕业年级师生和学校教务管理负责教师使用的信息化平台，因此需要在设计数据库时考虑到性能是否能够满足业务并发请求。在设计数据库表时，要尽力满足数据库三范式，达到设计的规范化。数据库表设计要合理，符合业务逻辑需求，满足相关属性的存储要求。本系统的数据库表如表 3-1 所示。

表 3-1 数据库表

数据库表名	表用途
dashboard	存储公告栏数据
dept	存储院系数据
major	存储专业数据
student	存储毕业生相关数据
teacher	存储指导教师相关数据
leader	存储管理员相关数据
gradus	存储毕业设计过程相关数据

3.5 系统概念模型图设计

实体-属性图，又简称为 E-R 图，表达了实体与属性之间的关系以及联系方法，生动形象地描绘表达了在现实中实体之间的关系属性，便于完成系统的设计工作。

毕业生的常用属性为学号、姓名、登录密码、院系、专业、班级等，这几类属性与其他表中公用数据进行对应映射，建立联系关系。

毕业生的概念模型图如图 3-1 所示。

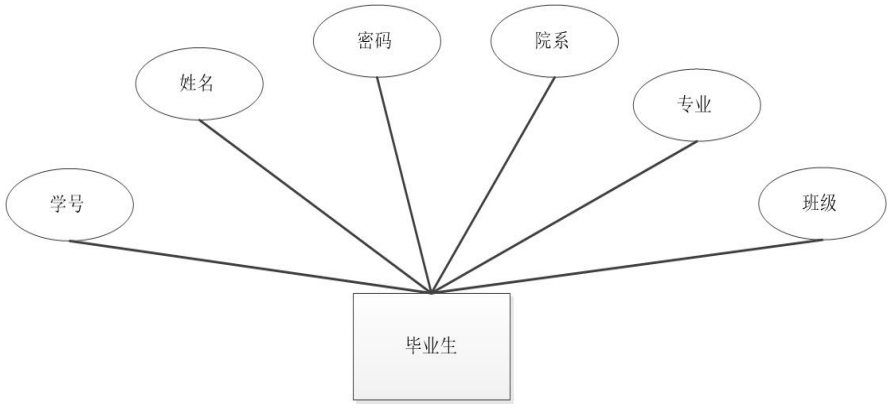

图 3-1 毕业生概念模型图

指导教师的常用属性为工号、姓名、密码、职称、院系、专业等，通过以上常用属性确定实体属性对应关系。

指导教师的概念模型图如图 3-2 所示。

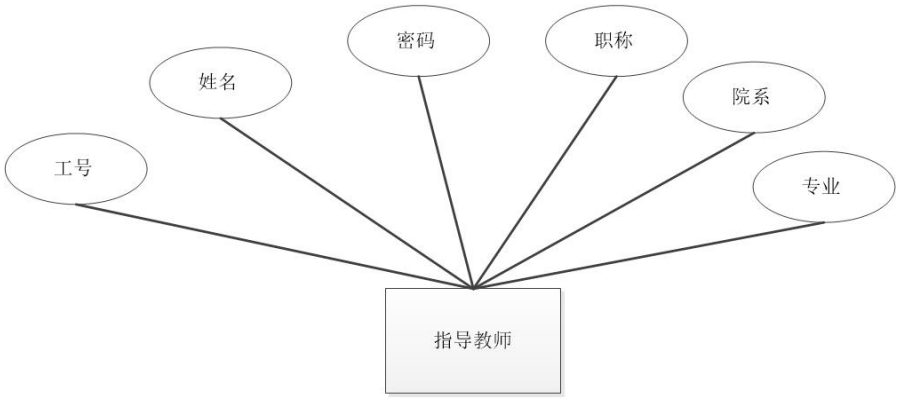

图 3-2 指导教师概念模型图

管理员实体的常用属性为管理员编号、姓名、登录密码、职务、所属院系、所属专业等。

管理员的概念模型图如图 3-3 所示。

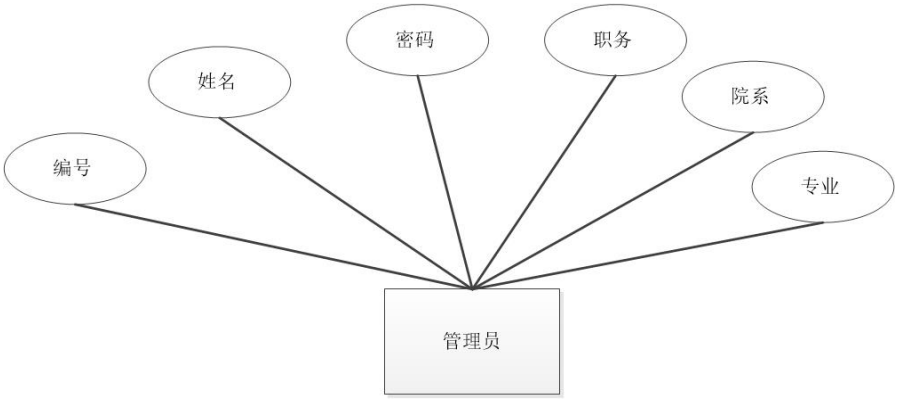

图 3-3 管理员概念模型图

毕业设计的实体主要属性有毕业设计课题编号、课题题目、课题内容、所属教师、选题学生、开题报告、中期检查报告、查重报告、毕业论文终稿等属性。

毕业设计的概念模型图如图 3-3 所示。

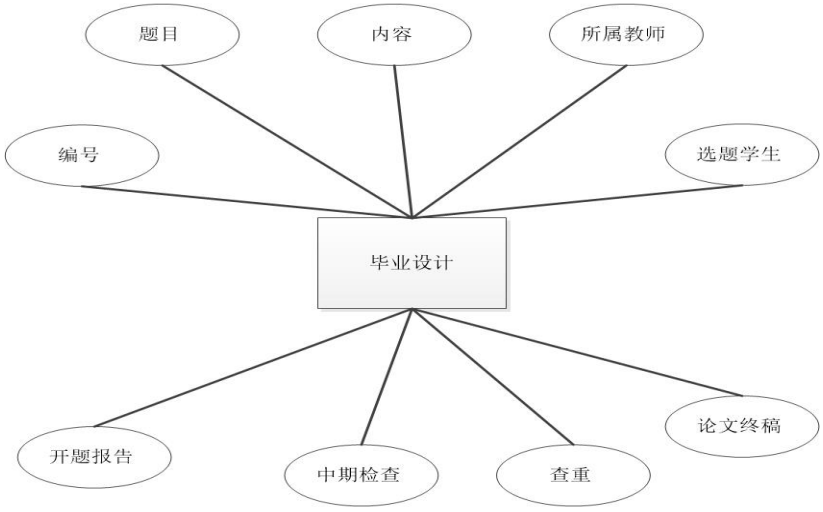

图 3-4 毕业设计概念模型图

3.6 系统逻辑模型图设计

在数据库的设计基础上，进行逻辑模型图的制作，同时对之前的设想进行复盘整理，改正不合逻辑或规范的设计，并对数据库表中各个字段的约束进行设计和配置。经过整理分析，本系统的数据库需要设计 7

个不同的表，存储不同的字段内容。

本系统的 7 个表分别为院系表、专业表、公告栏表、毕业生信息表、指导教师信息表、管理员表、毕业设计过程表。

本系统的逻辑模型图如图 3-5 所示。

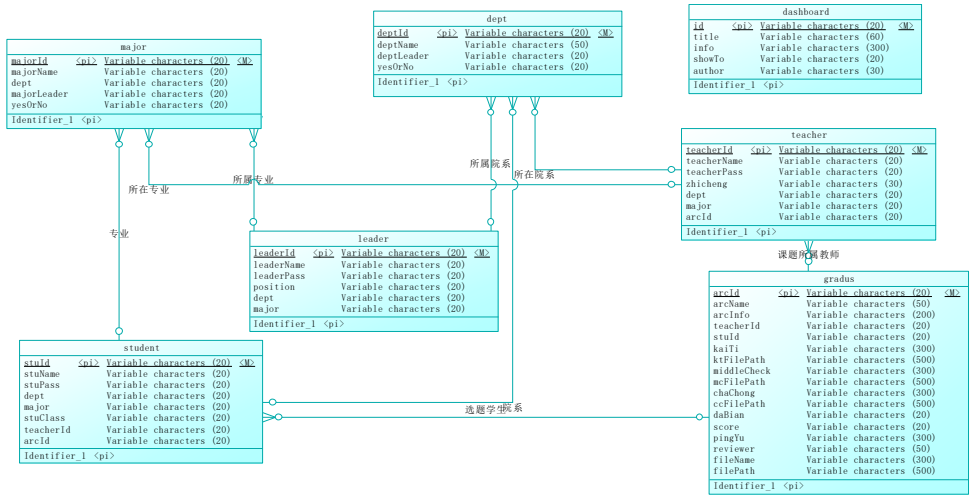

图 3-5 逻辑模型图

3.7 系统物理模型图设计

根据上一节的概念模型图的分析设计和制作，结合物理模型的特点和 MySQL 数据库的特性制作了本系统的物理模型图。

本系统的物理模型图如图 3-6 所示。

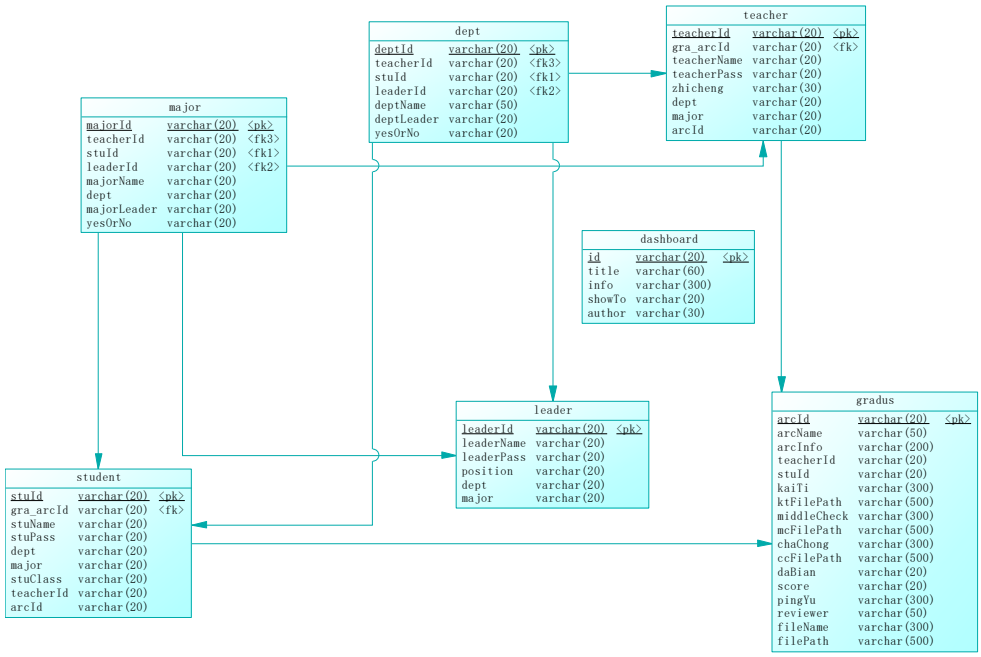

图 3-6 物理模型图

3.8 系统数据库表结构设计

根据对本系统的数据库表需求进行分析设计，可以了解到本系统需要设计 7 个不同的数据库表用以保存不同的数据内容。每个所需的表的结构、字段、内容、数据类型、字段含义等如下表 3-2~表 3-8 所示。

公告栏表（dashboard）结构内容如表 3-2 所示。

表 3-2 公告栏表结构

字段名	说明	数据类型	键类型	约束类型	备注
id	公告编号	varchar(50)	PK	非空	
title	公告标题	varchar(60)		非空	
info	公告内容详情	varchar(300)			
showTo	展示人群	varchar(20)		非空	
author	发布者	varchar(30)		非空	

院系表（dept）结构内容如表 3-3 所示。

表 3-3 院系表结构

字段名	说明	数据类型	键类型	约束类型	备注
deptId	院系编号	varchar(20)	PK	非空	
deptName	院系名称	varchar(50)		非空	
deptLeader	负责领导	varchar(20)			管理员
yesOrNo	是否启用	varchar(20)		非空	0 停用，1 启用

专业表（major）结构内容如表 3-4 所示。

表 3-4 专业表结构

字段名	说明	数据类型	键类型	约束类型	备注
majorId	专业编号	varchar(20)	PK	非空	
majorName	专业名称	varchar(50)		非空	
dept	所属院系	varchar(20)			
majorLeader	负责领导	varchar(20)		非空	管理员
yesOrNo	是否启用	varchar(20)		非空	0 停用，1 启用

毕业生信息表（student）结构内容如表 3-5 所示。

表 3-5 毕业生信息表结构

字段名	说明	数据类型	键类型	约束类型	备注
stuId	学号	varchar(20)	PK	非空	登录名
stuName	毕业生姓名	varchar(20)		非空	
stuPass	登录密码	varchar(300)		非空	登录密码
dept	所属院系	varchar(20)		非空	
major	所属专业	varchar(20)		非空	
stuClass	所在班级	varchar(20)		非空	
teacherId	指导教师	varchar(20)			
arcId	课题编号	varchar(20)			

指导教师信息表（teacher）结构内容如表 3-6 所示。

表 3-6 指导教师信息表结构

字段名	说明	数据类型	键类型	约束类型	备注
teacherId	指导教师编号	varchar(20)	PK	非空	登录名
teacherName	指导教师姓名	varchar(20)		非空	
teacherPass	登录密码	varchar(300)		非空	登录密码
zhicheng	职称	varchar(20)		非空	
dept	所属院系	varchar(20)		非空	
major	所属专业	varchar(20)		非空	
arcId	负责课题	varchar(20)			

管理员信息表（leader）结构内容如表 3-7 所示。

表 3-7 管理员信息表结构

字段名	说明	数据类型	键类型	约束类型	备注
leaderId	管理人员编号	varchar(20)	PK	非空	毕业生登录名
leaderName	姓名	varchar(20)		非空	
leaderPass	密码	varchar(20)		非空	登录密码
position	职务	varchar(20)		非空	
dept	所属院系	varchar(20)		非空	
major	所属专业	varchar(20)		非空	

毕业设计过程信息表（gradus）结构内容如表 3-8 所示。

表 3-8 毕业设计过程信息表结构

字段名	说明	数据类型	键类型	约束类型	备注
arcId	课题编号	varchar(20)	PK	非空	
arcName	课题题目	varchar(50)		非空	
arcInfo	课题内容	varchar(200)		非空	

teacherId	负责教师	varchar(20)		非空	
stuId	选题学生	varchar(20)			
kaiTi	开题标记	varchar(20)			
ktFilePath	开题报告文件保存路径	varchar(500)			
middleCheck	中期检查标记	varchar(20)			
mcFilePath	中期检查文件保存路径	varchar(500)			
chaChong	查重标记	varchar(20)			
ccFilePath	查重报告文件保存路径	varchar(500)			
daBian	是否同意进入答辩	varchar(20)			0 不同意，1 同意
score	毕业论文成绩	varchar(20)			
pingYu	论文评语	varchar(300)			
reviewer	评阅人	varchar(50)			评阅人用户名
fileName	毕业论文文件标记	varchar(20)			
filePath	毕业论文文件保存路径	varchar(500)			

3.9 系统界面设计

本系统采用 B/S 架构进行开发，因此，前端用户交互界面展示部分主要使用 HTML5 技术进行开发，用户通过浏览器访问系统进行操作使用。为了提高使用者的效率，本系统的前端界面力求简洁高效好用，便于学习和使用。

3.9.1 登录界面原型设计

登陆界面设计十分简洁，主要功能区内有角色选择按钮、用户名输入框、密码输入框、登录按钮等。首次访问使用本系统的用户可以直观地了解到登录所需的凭据信息等，友好便于上手。

登录界面原型如图 3-7 所示。

图 3-7 登录界面原型图

3.9.2 系统首页界面原型设计

系统首页为公告栏，展示系统中发布的公告，方便用户登录后立即查看到。在使用管理员身份登录系统后，首页公告处可以进行发布、修改、删除公告等管理操作。

系统首页界面原型如图 3-8 所示。

图 3-8 系统首页界面原型图

3.9.3 毕业生信息管理界面原型设计

毕业生信息管理主界面展示了当前管理员权限下的所属院系的所有学生列表信息，支持分页功能和模糊查询功能。可以对系统中已有的学生信息进行查看、修改、删除等操作，也可以新增毕业生信息。

毕业生信息管理界面原型如图 3-9 所示。

图 3-9 毕业生信息管理界面原型图

3.9.4 指导教师管理界面原型设计

指导教师信息管理主界面展示了当前管理员权限下的所属院系的所有指导教师列表详细信息，支持分页功能和模糊查询功能。

指导教师管理界面原型如图 3-10 所示。

图 3-10 指导教师管理界面原型图

3.9.5 毕业论文审阅界面原型设计

毕业论文审阅管理主界面展示了当前评阅组管理教师登录后可以进行评审的学生和课题名单列表，支持分页查询功能，可以直观的看到课题对应学生相关信息，点击审阅按钮进入审阅状态。

毕业论文审阅界面原型如图 3-11 所示。

图 3-11 毕业论文审阅界面原型图

3.9.6 学生选题界面原型设计

学生选题界面展示了当前登录的学生的选题结果，如果未选题界面

会显示当前未选题，同时界面上会显示选题按钮。如果已经选题的学生进入后界面会显示学生的选题结果。

学生选题界面原型如图 3-12 所示。

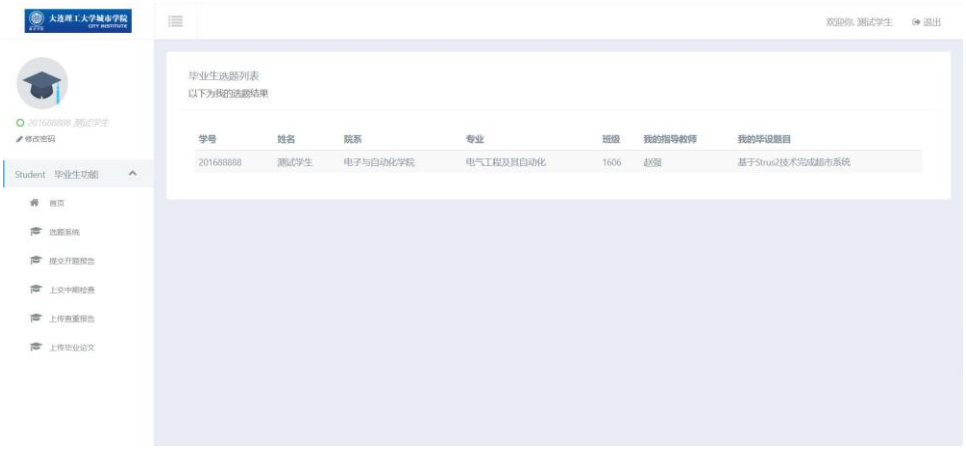

图 3-12 学生选题界面原型图

3.9.7 学生上交毕业论文终稿界面原型设计

学生上交毕业论文终稿界面展示了上交毕业论文终稿界面的功能按钮和所需的操作步骤，便于操作。

学生上交毕业论文终稿界面原型如图 3-13 所示。

图 3-13 学生上交毕业论文终稿界面原型图

第四章 系统详细设计

4.1 系统详细设计方案

毕业设计过程管理系统主要分为前端和后端两部分，后端使用 Spring Boot 框架结合 MyBatis 框架进行开发，实现系统后端的数据处理和分发，前端使用 Thymeleaf 结合 HTML5、jQuery、Bootstrap 等技术渲染构建，前端和后端之间的交互通过 ajax 进行。

本系统的目标用户为毕业年级学生、指导教师和管理员教师，这三类用户角色可以凭借自己的用户名和密码登录访问系统，不同的用户具备不同的功能访问权限。

4.2 详细设计原则

本系统的用户量较为庞大，每年毕业年级的师生都需要依靠使用本系统来完成毕业设计的过程管理，因此系统开发时要注意适配支持高并发访问，避免服务器宕机造成损失。本系统采用 B/S 架构设计开发，对于服务的部署以及后续的维护工作带来极大的便利性，减少工作量。系统整体开发层面使用的是 MVC 模式，分模块分层次进行开发，再将开发好的模块结合需求分析的结果进行“组装”，完成最终开发工作。

4.3 各个功能模块详细设计

4.3.1 登录功能设计

本系统的用户角色有三类，因此在登录界面上要设计有切换选择登录用户角色的功能区，用户选择自己的角色后再输入自己的用户名以及密码，系统校验通过后成功进入系统主界面。

由于本系统的用户有三类，因此本系统的登录部分设计需要分为三类分别进行设计开发实现。

第一类用户是管理员用户，该类用户一般具有系统的最高功能访问

权限，管理员用户在登录后将跳转到管理员的首页，即公告管理模块。管理员用户登录的顺序图如图 4-1 所示。

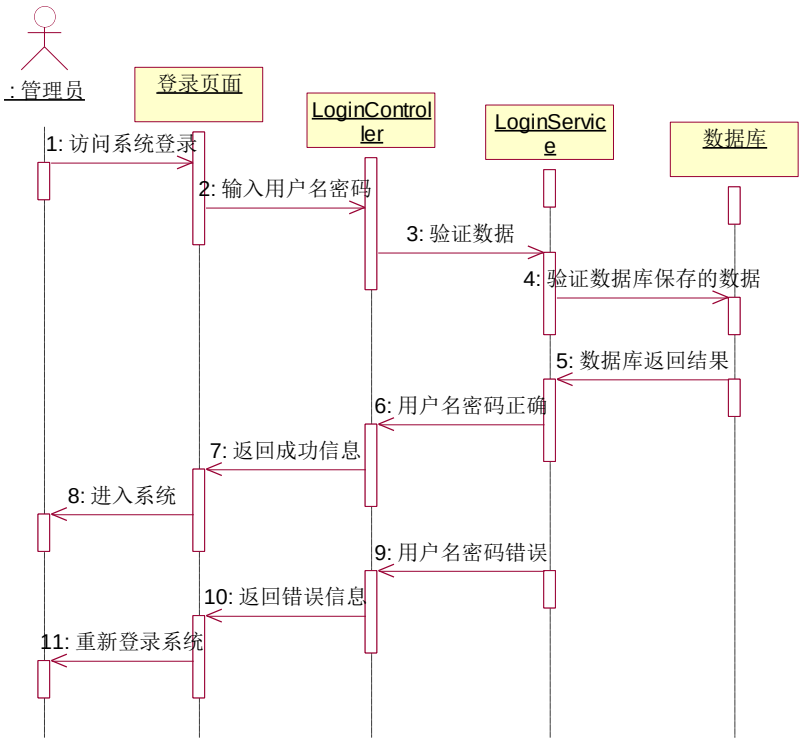

图 4-1 管理员登录顺序图

本系统的第二类用户是毕业生，该类用户登录后需要跳转到毕业生公告板首页，同时展示毕业生可用的模块功能资源。毕业生登录的顺序图如图 4-2 所示。

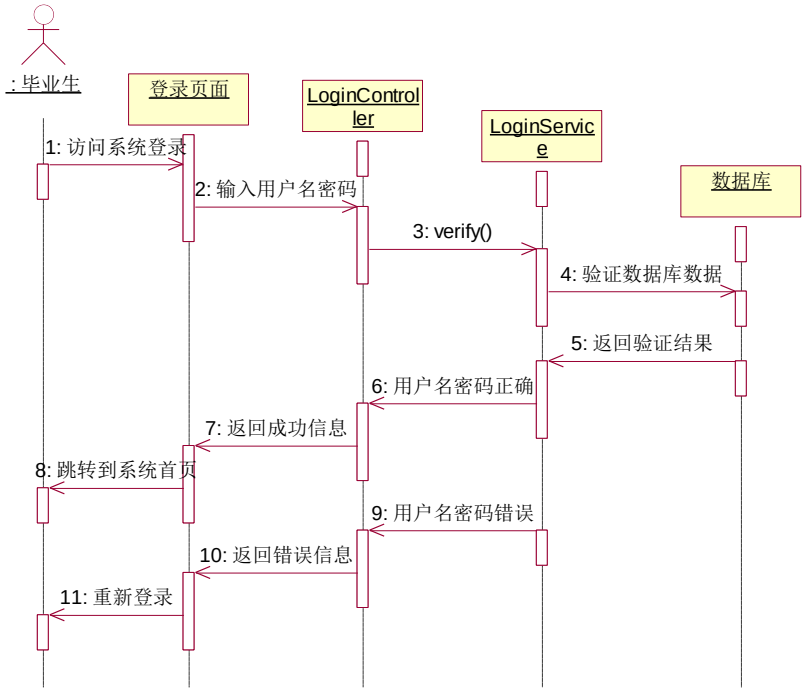

图 4-2 毕业生登录顺序图

本系统的第三类用户为毕业设计指导教师，该类用户在登录成功后会跳转到指导教师公告栏首页。指导教师的登录过程顺序图如图 4-3 所示。

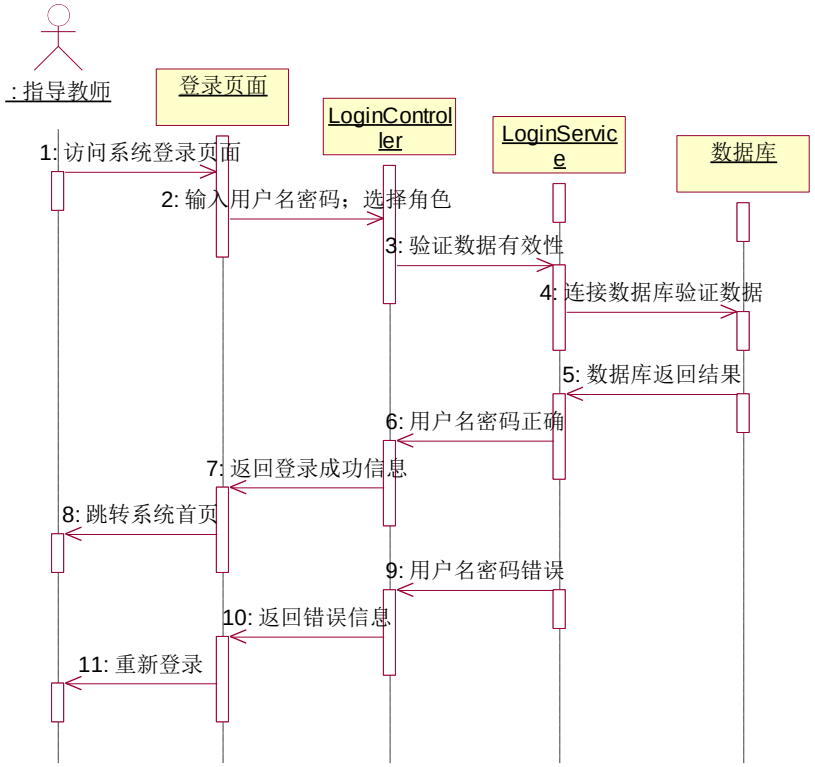

图 4-3 指导教师登录顺序图

登录界面如图 4-4 所示。

图 4-4 登录界面

如果输入的用户名或者密码错误，系统会给出提示，拒绝登录请求，

这时需要检查自己的输入是否正确。由于本系统的特殊性，如果忘记密码需要联系管理员来重置密码。

4.3.2 毕业生选题功能设计

毕业生在登录时选择“毕业生”角色，输入用户名密码登录后会进入系统首页，首页可以查看学校发布的给毕业生的通知公告，学生登录后即可查看信息。

在选题系统中，毕业生可以查看自己的选题情况。如果未选题页面上会给出尚未选题的提示，此时学生可以点击“开始选题”按钮进入选题功能界面，选择自己心仪的导师下的课题，选择完成后可以回到选题系统首页查看选题结果。选题公告顺序图如图 4-5 所示，选题系统界面如图 4-6、图 4-7 所示。

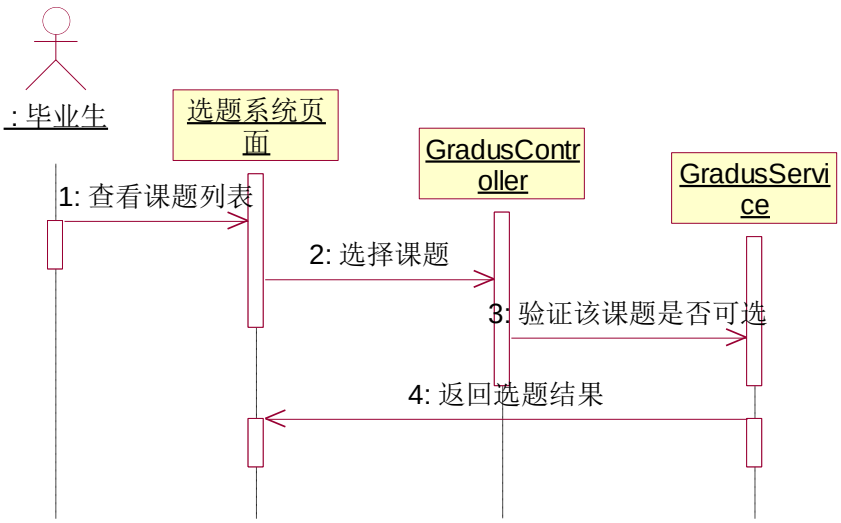

图 4-5 选题功能顺序图

图 4-6 毕业生选题系统首页

图 4-7 毕业生选题功能界面

4.3.3 毕业生上交开题报告功能设计

毕业生在选题后可以进入开题报告上传系统，如果毕业生未选题的话，进入开题报告上传系统时会提醒未选题，同时跳转回选题系统页面。开题报告未选题状态提示如图 4-9 所示。如果已经选题后进入系统，则可以在规定时间内上传自己的开题报告文件，上传开题报告界面如图 4-10 所示。上交开题报告模块功能设计的顺序图如图 4-8 所示。

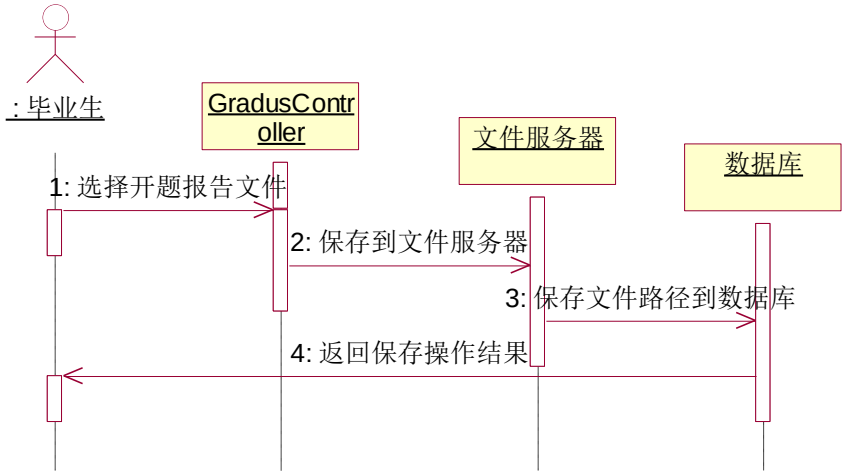

图 4-8 上交开题报告顺序图

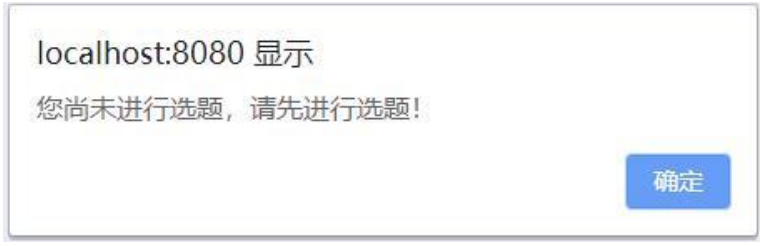

图 4-9 未选题进入开题报告系统提示

图 4-10 上传开题报告界面

4.3.4 毕业生上交查重报告功能设计

毕业生在完成论文撰写后，按照学校院系的规定时间去指定网站进行学术不端检查，检查完成后下载查重报告，在本系统内按照规定时间进行上传查重报告，供学校学术评审组老师和指导教师进行查看评审。

上交查重报告功能的顺序图如图 4-11 所示，上传查重报告系统界面如图 4-12 所示。

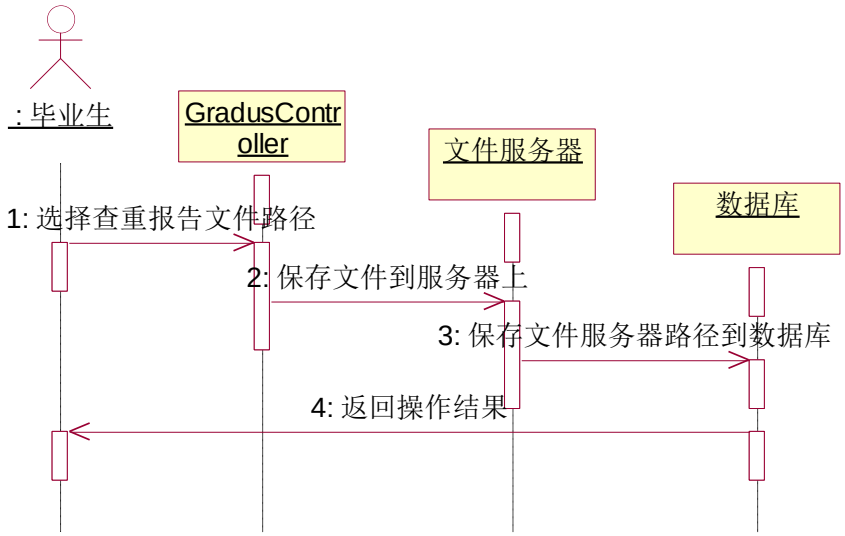

图 4-11 上传查重报告顺序图

图 4-12 上传查重报告系统界面

4.3.5 毕业生上交论文终稿功能设计

在毕业论文终稿上交截止日期前，毕业生需要登录本系统，访问上交论文终稿页面，进行毕业论文终稿的上传，供指导教师检查，决定是否同意进入答辩环节。

上交毕业论文终稿功能的顺序图如图 4-13 所示。上交毕业论文终稿界面如图 4-14 所示。

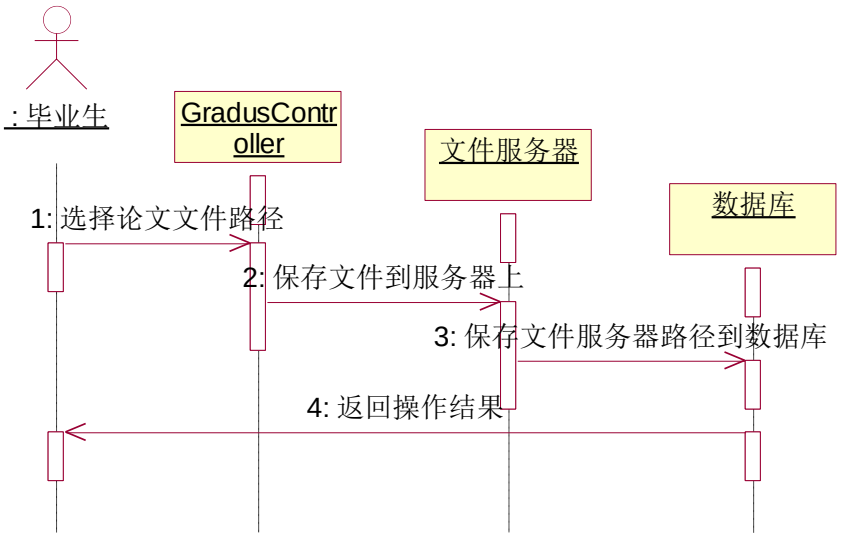

图 4-13 上交论文终稿顺序图

图 4-14 上传毕业论文终稿系统界面

4.3.6 毕业论文审阅功能设计

毕业论文审阅功能允许评审组老师凭账户密码登录进行评审，评审组老师只能评审自己所负责的专业或院系的毕业生论文，已经完成评审的不能再次进行评审。

毕业论文审阅功能的顺序图如图 4-15 所示。

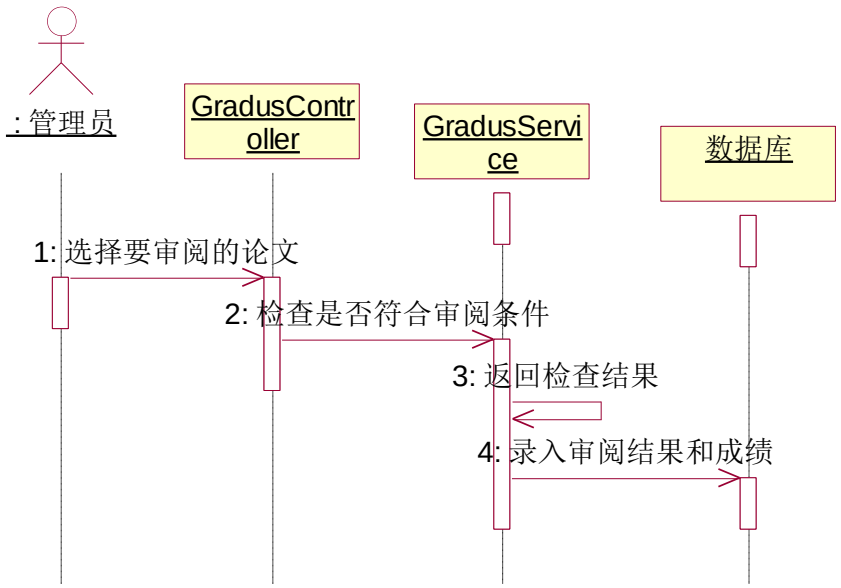

图 4-15 论文审阅顺序图

毕业论文审阅系统界面如图 4-16 所示。

图 4-16 毕业论文审阅功能界面

第五章 系统编码实现

5.1 系统编码规范

本系统使用 Java 语言进行开发实现，为了使系统开发过程符合业界主流规范，降低代码耦合性，减少后期修改或更新的工作压力，开发时遵循了当前市面上各大公司主流推荐的阿里巴巴 Java 开发规范进行编码约束。

阿里巴巴推行的 Java 开发规范中，对变量/常量命名、方法命名规范、编码格式规范、接口开发规范、数据处理判断、流程控制规范、注释规范、异常处理规范等各方面开发细节均做出了规范示例。本系统开发过程中参考了该规范，使系统代码更加整洁规范。

系统使用了国内市场流行使用的阿里巴巴规范进行约束设计实现，如变量命名尽量使用有意义的英文单词或字母，变量采用驼峰命名原则，使用自动生成的 get/set 方法避免错误，常量不允许出现魔法值等原则，并在开发时尽力遵循。

5.2 代码演示

5.2.1 登录模块

模型（Model）层代码展示如下：

```
public class LoginModel {
    private String username;
    private String password;
    private String role;
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
```

```
            this.username = username;
        }
        public String getPassword() {
            return password;
        }
        public void setPassword(String password) {
            this.password = password;
        }
        public String getRole() {
            return role;
        }
        public void setRole(String role) {
            this.role = role;
        }
    }
```

视图（View）层代码展示如下：

```
    <form class="am-form tpl-form-line-form">
      <div class="am-form-group">
      <label class="am-radio-inline">
        <input type="radio" name="role" value="leader" data-am-ucheck
checked>
            管理员/领导
    </label>
    <label class="am-radio-inline">
        <input type="radio" name="role" value="teacher" data-am-ucheck>
            指导教师
    </label>
    <label class="am-radio-inline">
        <input type="radio" name="role" value="student" data-am-ucheck>
```

```
                毕业生
        </label>
      </div>

      <div class="am-form-group">
        <input type="text" class="tpl-form-input" name="username"
id="username" placeholder="请输入账号" required>
      </div>
      <div class="am-form-group">
        <input type="password" class="tpl-form-input" name="password"
id="password" placeholder="请输入密码" required>
      </div>
      <div class="am-form-group">
      <button type="button" id="submit" class="am-btn am-btn-primary
am-btn-block tpl-btn-bg-color-success    tpl-login-btn">登录</button>
      </div>
      </form>
```

控制（Controller）层代码展示如下：

```
      @RequestMapping(value = "/verify2")
            public ResultMsg loginMain(HttpServletRequest
request,@RequestBody LoginModel loginModel) throws Exception{
                  String userName = loginModel.getUsername();
                  String password = loginModel.getPassword();
                  String role = loginModel.getRole();
                  if (null==userName || userName.trim().length()==0 ||
"".equals(userName)){
                        return new ResultMsg(0,"用户名不能为空！");
                  } else if (null==password || password.trim().length()==0 ||
"".equals(password)){
```

```
                return new ResultMsg(0,"密码不能为空！");
            } else {
                if ("student".equals(role)){
                    if (studentService.verify(userName,password)){
                        StudentModel studentModel =
studentService.selectById(userName);
    HttpSession session = request.getSession();
   session.setAttribute("loginuser",studentModel.getStuId());
    session.setAttribute("username",studentModel.getStuName());
    session.setAttribute("role",role);
    return new ResultMsg(1,"登录成功！");
      }else {
          return new ResultMsg(0,"用户名或密码错误！");
      }
    } else if ("teacher".equals(role)){
        if (teacherService.verify(userName,password)){
            TeacherModel teacherModel =
teacherService.selectTeacherById(userName);
            HttpSession session = request.getSession();
        session.setAttribute("loginuser",teacherModel.getTeacherId());
        session.setAttribute("username",teacherModel.getTeacherName());
        session.setAttribute("role",role);
        return new ResultMsg(2,"登录成功！");
          }else {
              return new ResultMsg(0,"用户名或密码错误！");
          }
        }else if ("leader".equals(role)){
                    if (leaderService.verify(userName,password)){
                        LeaderModel leaderModel =
```

```
leaderService.selectLeaderById(userName);
                                    HttpSession session = request.getSession();
        session.setAttribute("loginuser",leaderModel.getLeaderId());

session.setAttribute("username",leaderModel.getLeaderName());
            session.setAttribute("role",role);
        return new ResultMsg(3,"登录成功！");
                                }else {
        return new ResultMsg(0,"用户名或密码错误！");
                                }
                            }else {
        return new ResultMsg(0,"请选择正确的登录角色!");
                            }
                        }
                    }
```

代码执行结果如图 5-1 所示。

图 5-1 登录模块代码执行结果

5.2.2 毕业生管理模块

模型（Model）层代码展示如下：

```
public class StudentModel {
    private String stuId;
    private String stuName;
    private String stuPass;
    private String dept;
    private String major;
    private String stuClass;
    private String arcId;
    public String getStuId() {
        return stuId;
    }
    public void setStuId(String stuId) {
        this.stuId = stuId;
    }
    public String getStuName() {
        return stuName;
    }
    public void setStuName(String stuName) {
        this.stuName = stuName;
    }
    public String getStuPass() {
        return stuPass;
    }
    public void setStuPass(String stuPass) {
        this.stuPass = stuPass;
    }
    public String getDept() {
        return dept;
    }
```

```
    public void setDept(String dept) {
        this.dept = dept;
    }
    public String getMajor() {
        return major;
    }
    public void setMajor(String major) {
        this.major = major;
    }
    public String getStuClass() {
        return stuClass;
    }
    public void setStuClass(String stuClass) {
        this.stuClass = stuClass;
    }
    public String getTeacherId() {
        return teacherId;
    }
    public void setTeacherId(String teacherId) {
        this.teacherId = teacherId;
    }
    public String getArcId() {
        return arcId;
    }
    public void setArcId(String arcId) {
        this.arcId = arcId;
    }
}
```

视图（View）层代码展示如下：

```
<table width="100%" class="am-table am-table-compact
am-table-striped tpl-table-black " id="example-r">
<thead>
  <tr>
    <th>学号</th>
    <th>姓名</th>
    <th>院系</th>
    <th>专业</th>
    <th>班级</th>
    <th>指导教师</th>
    <th>毕设题目</th>
    <th>操作</th>
  </tr>
</thead>
<tbody>
<tr class="gradeX"    th:each="student:${pageInfo.list}" >
<td th:text="${student.stuId}"></td>
<td th:text="${student.stuName}"></td>
<td th:text="${student.dept}"></td>
<td th:text="${student.major}"></td>
<td th:text="${student.stuClass}"></td>
<td th:text="${student.teacherId}"></td>
<td th:text="${student.arcId}"></td>
<td>
<div th:id="${student.stuId}" class="tpl-table-black-operation">
<a th:id="${student.stuId}" href="javascript:;"    name="edit"
id="edit">
<i th:id="${student.stuId}" class="am-icon-pencil"></i> 编辑
</a>
```

```
<a th:id="${student.stuId}" href="javascript:;" name="delete"
class="tpl-table-black-operation-del">
<i th:id="${student.stuId}" class="am-icon-trash"></i> 删除
</a>
</div>
</td>
</tr>
</tbody>
</table>
```

控制（Controller）层代码展示如下：

```
@RequestMapping(value = "/index")
public String listStu(Model model,@RequestParam(required =
false,defaultValue="1",value="pageNum") int
pageNum,@RequestParam(defaultValue="10",value="pageSize") int
pageSize) throws Exception{
PageHelper.startPage(pageNum,pageSize);
List<StudentModel> list = studentService.selectAll();
for (int i=0;i<list.size();i++){
                if (null != list.get(i).getDept()
&& !"".equals(list.get(i).getDept())){
                    String dept =
deptService.selectById(list.get(i).getDept()).getDeptName();
                    list.get(i).setDept(dept);
                }
                if (null != list.get(i).getMajor()
&& !"".equals(list.get(i).getMajor())){
                    String major =
majorService.selectById(list.get(i).getMajor()).getMajorName();
                    list.get(i).setMajor(major);
```

```
            }
        }
        PageInfo<StudentModel> pageInfo = new
PageInfo<StudentModel>(list,pageSize);
        model.addAttribute("pageInfo",pageInfo);
        return "/student/index";
    }
```

代码执行结果如图 5-2 所示。

图 5-2 毕业生管理模块代码执行结果

第六章 系统测试

6.1 测试目的

软件测试是为了验证开发的系统功能能否符合预期想象和设计进行运行。一般来说，在一个模块或功能完成开发后就要进行单元测试，在全部系统功能完成后，要进行集成测试对开发完成的系统进行验收检查，发现在开发时遗留的 Bug 或重大安全漏洞等，避免在系统上线部署运行后造成损失或影响。

6.2 测试计划

6.2.1 单元测试

单元测试一般是对一个系统的最小可测试模块单元进行功能验证测试，通常是对某一个函数或方法进行测试，验证输出的结果是否符合预期设想要求。

在本系统的单元测试过程中，使用了白盒测试作为单元测试的手段对系统各个方法功能接口等进行了功能验证测试。

6.2.2 集成测试

集成测试，又称组装测试，一般是在单元测试的基础上，将所有模块或函数按照业务逻辑要求进行组装再测试的过程。

本系统在各个模块开发完成后，使用黑盒测试的手段对各个业务模块划分进行了集成测试，可以测试出程序开发时的不足和缺陷。

6.2.3 性能测试

性能测试一般是采用了自动化的测试工具模拟各种正常或异常的负载条件，对系统的各类性能指标进行测试，从而检测系统的性能是否达到预期要求，能否胜任工作，也能从中了解到系统的最大负荷量，避免过载导致系统崩溃。

6.2.4 功能测试

本毕业设计过程管理系统在编码开发完成后选择使用黑盒测试测试了系统的关键模块功能是否正常，能否符合业务逻辑，运行是否报错等。

为验证系统功能，根据需求分析说明书，本系统采用黑盒测试设计测试用例，完成系统的功能测试。[8]黑盒测试不考虑代码编码方式和实现手段，只是依照需求规格说明来进行测试功能是否正常运行输出预期结果，黑盒测试可以大大提升系统的可靠性。

6.3 测试用例

根据毕业设计过程管理系统的需求，特此制定了部分测试用例用于系统测试，来测试系统关键功能是否可用，输出返回结果是否符合预期等。制定的测试用例如下表 6-1 所示。

表 6-1 测试用例

测试项目	测试步骤	预期结果	执行结果	是否达到预期
登录系统	使用错误的密码登录管理员角色	登录失败	登录失败	是
登录系统	使用毕业生账号，登录时选择管理员角色尝试登录	登录失败	登录失败	是
毕业生选题	在上交毕业论文终稿时尝试上传空文件	提示不能上传空文件	不能上传空文件	是
指导教师上报选题	尝试上报一个已经存在于系统的选题	提示重复	提示重复	是
审阅毕业	审阅一个已经	无法审阅	无法审阅	是

论文	被其他评审组教师审阅完成的论文			

其中，在进行登录系统的安全验证测试过程中，重点测试了能否正常进行登录已经登录安全校验的测试，如图 6-1 和图 6-2 所示。经过多次严谨的测试，本系统登录校验功能部分正常工作，符合设计的预期需求。

图 6-1 登录成功界面反馈

图 6-2 登录失败反馈界面

6.4 测试结论

通过多种科学有效的测试手段，对系统进行了一系列科学严谨的测试，设计了一系列高效的测试用例，经过以上部分的测试用例的测试，本系统功能符合预期需求，运行正常，不存在报错等干扰使用的情况，各项功能基本达到预期目标，实现预期功能，系统运行稳定无异常。

第七章 结论

本次毕业设计选择了基于 SpringBoot 毕业设计过程管理系统这个课题，本系统的意义在于使高校管理系统更加信息化，降低师生的压力，提高学校的工作效率，使高校的发展不再受牵制，从而提升高校的综合实力水平。

本系统在技术上，后端选用 Java 语言进行开发，使用了目前流行的 Spring Boot 框架进行开发，结合 MyBatis 框架完成 DAO 层代码开发实现。前端使用了 Thymeleaf 渲染引擎，结合 HTML5、CSS3、Bootstrap、jQuery 等前端技术实现前台界面的构建。服务器使用了 Spring Boot 集成的 Tomcat 服务器。

本系统在功能上实现了对毕业生、指导教师、管理员、毕业设计过程极大功能模块的管理。在用户上，分为毕业生、指导教师、管理员三类用户。实现本项目前，首先需要完成对本项目的需求分析，根据需求规格说明书完成对数据库实体的分析和设计，然后进行概要设计，详细分析系统的每一个功能，在此基础上进行详细的设计准备，再进入编码开发阶段，进行开发。开发编码完成后，对本系统进行测试，通过单元测试、黑盒测试、白盒测试、验收测试等，及时发现系统代码开发中存在的问题和漏洞，并及时进行修复，最后再进行回归测试，检查是否修正已发现的问题。

本项目还有一些美中不足的地方，系统的功能有些地方还不够完善。在未来有机会可以继续学习并改进这些问题。

致　谢

首先，我要在此感谢我的父母，是他们含辛茹苦地把我抚养成人，努力工作供我学习上大学，没有我的父母对我的付出和照顾就没有我的今天，我也不可能有机会接受到良好的高等教育。

其次，我要感谢我的母校，我的母校设置了很多专业的课程和基础课程，让我在大学中度过了忙碌而充实的四年生活。在大学期间，我凭借在学校学习到的专业知识和扎实的基本功，收获了不少奖励和经验，这些都为我的未来职业生涯做好了铺垫。

我还要感谢我在大学期间遇到的所有任课教师和辅导老师，他们每天不厌其烦地为我们学生答疑解惑，把我们当成他们自己的孩子一样，一视同仁地对待我们，帮助我们，照顾我们，不论是学习上还是生活上。在我的大学生涯中，我的老师们在我遇到挫折和困难时，总是会来安慰我、鼓励我，教育我人生的哲理，让我在四年中不断地成长，这对我的人生都有很大的帮助。

在毕业设计和论文的撰写过程中，我要感谢我的指导教师张应博教授，张老师在我遇到问题和困难时不厌其烦地耐心解答我的问题和疑惑，哪怕是周末节假日等休息时间，他还是能想着积极帮我答疑解惑。从开始的选题定题阶段，到需求分析阶段，数据库设计阶段，再到代码编写阶段，直到论文撰写阶段，张老师在每一阶段都悉心指导检查，帮助了我很多。并且我在张老师对我进行指导的过程中也学到了他多年来积累的一些经验，对我而言，这些宝贵的人生经历和经验无论是在毕业设计还是今后的职业生涯和人生道路上对我都将受益匪浅。

我也要感谢我的可爱的大学同学们，他们在我的日常学习生活中帮助了我很多，祝大家顺利毕业，前程似锦。

最后，我要再次感谢我的母校和所有我遇到的老师们，他们不仅教会了我们专业知识，也教会了我人生的哲理。

参考文献

[1] 陈光剑.Spring Boot 开发实战[M].北京：机械工业出版社.2018.

[2] 张志佳,彭新茗,朱天翔.毕业设计信息管理系统的研发与应用[J].黑龙江教育(高教研究与评估),2017(02):48-51.

[3] 李莉.“互联网+”背景下毕业设计管理系统的研究与设计[J].信息与电脑(理论版),2019,31(21):101-103.

[4] 张雷,王悦.基于 SpringBoot 微服务架构下的 MVC 模型研究[J].安徽电子信息职业技术学院学报,2018,17(04):1-9.

[5] 张峰.应用 SpringBoot 改变 web 应用开发模式[J].科技创新与应用,2017(23):193-194.

[6] 熊永平.基于 SpringBoot 框架应用开发技术的分析与研究[J].电脑知识与技术,2019,15(36):76-77.

[7] 吕宇琛.SpringBoot 框架在 web 应用开发中的探讨[J].科技创新导报,2018,15(08):168+173.

[8] 肖祥林.基于 SSM 的毕业设计管理系统设计与实现[J].电子科技,2016,29(10):115-117.

[9] 马明.毕业设计选题管理系统的设计与实现[J].电脑与电信,2015(10):61-63+69.

[10] 刘增辉.MyBatis 从入门到精通[M].北京：电子工业出版社.2017.

[11] Spring 企业级程序设计[M].武汉：中国地质大学出版社.2019.

[12]Yuxiang Hou. The design and implementation of the framework for Spring+SpringMVC+MyBatis in the development of Web application[A]. Institute of Management Science and Industrial Engineering.Proceedings of 2019 4th International Industrial Informatics and Computer Engineering Conference(IIICEC 2019)[C].Institute of Management Science and Industrial Engineering:,2019:6.

英文翻译

学　　院 ：计算机工程学院

专　　业 ：软件工程

学　　生 ：高砚策

学　　号 ：201612012

指导教师 ：张应博

An Overview of World Wide Web Caching

With the continuing growth in popularity of Internet, coupled with the sudden dominance of the HTTP protocol, Web users are suffering from the network congestion and server overloading. So come the scalability demands on Web's infrastructure.

The World Wide Web (WWW) can be considered as a large distributed information system where users can access to shared data objects. Its usage is quite inexpensive, and accessing information is faster using the WWW than using any other means. Also, the WWW has documents that appeal to a wide range of interests, e.g. news, education, scientific research, sports, entertainment, stock market growth, travel, shopping, weather, maps, etc. Although the Internet backbone capacity increases as 60% per year, the demand for bandwidth is likely to outstrip supply for the foreseeable future as more and more information services are moved onto the Web. If some kind of solution is not undertaken for the problems caused by its rapidly increasing growth, the WWW would become too congested and its entire appeal would eventually be lost. Web proxy caching are the popular techniques to improve web's performance. Caching popular objects at locations close to the clients has been recognized as one of the effective solutions to alleviate Web service bottlenecks, reduce traffic over the Internet and improve the scalability of the WWW system.

WWW proxy caching system attempts to improve performance in three ways. Firstly, caching attempts to reduce the user-perceived latency associated with obtaining Web documents. Latency can be reduced because the proxy cache is typically much closer to the client than the content provider.

Secondly, caching attempts to lower the network traffic from the Web servers. Network load can be lowered because documents that are served

from the cache typically traverse less of the network than when they are served by the content provider. Finally, proxy caching can reduce the service demands on content providers since cache hits need not involve the content provider. It also may lower transit costs for access providers.

For the single cache, the main issues: it concerns should be prefeching, documents replacement strategy, caching performance index, coherence etc. However, for those multiple caching architecture, they should not only concerns the above things, but also should concern more complex ones, such as the organization of multiple caches, the routing problems for the cache requests, co-operative strategies between cache servers, cache server location in the: whole wide-area networks etc. Cooperative caching architectures can. be divided into three major categories, i.e., hierarchical, distributive and hybrid.

2.1 Hierarchical caching architecture

Hierarchical caching architecture was first proposed in the Harvest project . In this type of scheme, the caching system has a tree structure in which every node points to its parent (i.e. if a cache miss occurs at a cache, it forwards the request to its, parent. This goes on till we reach the root. The root then contacts the origin server if it is unable to satisfy the request.). When the document is found, either at a cache or at the original server, it travels in the opposite direction, leaving a copy at each of the intermediate caches along its path. This structure is. consistent with the present Internet with ISPs at each. level. It serves to diffuse the popular documents towards the demand. However, this architecture suffers from the following problems, such as' each hierarchy level will introduce additional delays in processing all request, there will be redundancy in the storage of documents as documents are replicated at each level, high level caches may become bottlenecks and have long queuing delays.

2.2 Distributed caching architecture

With distributed caching architecture no intermediate caches are set up and there are only institutional caches at the edge of the network that cooperate to serve each other's misses. Since there are no intermediate caches that store and centralize all documents requested by lower level caches, institutional caches need other mechanisms to share the documents the contain. Some of these mechanisms are:

- Institutional caches can query the other cooperating institutional caches for documents that resulted in local misses (this is usually done using the Inter Cache Protocol ICP). However, using a query-based approach may significantly increase the bandwidth consumption and the experienced latency by the client since a cache needs poll all cooperating caches and wait for the slowest one to answer.
- Institutional caches can keep a digest or summary of the content of the other cooperating caches, thus avoiding the need for queries/polls. Content digests/summaries are periodically exchanged among the institutional caches. To make the distribution of the digest/summaries more efficient and scalable, a hierarchical infrastructure of intermediate nodes can be used. However, this hierarchical infrastructure only distributes information about the location of the documents but does not store document copies.
- Institutional caches can cooperate using a hash function that maps a client request into a certain cache. With this approach there are no duplicated copies of the same document in different caches and there is no need for caches to know about each other's content. However, having only one single copy of a document among all cooperating caches, limits this approach to local environments with well- interconnected caches.

2.3. Hybrid caching architecture

In a hybrid scheme, caches cooperate with other caches at the same level or at a higher level using distributed caching. ICP is a typical example. The

document is fetched from a parent/neighbor cache that has the lowest RTT. Rabinovich proposed to limit the cooperation between neighbor caches to avoid obtaining documents from distant or slower caches, which would have been retrieved directly from the origin server at a lower cost.

Pablo and Christian have proposed a mathematical model to analysis some important performance parameters for all of the above three schemes. They find that hierarchical caching system have lower connection time while distributed caching system has lower transmission time. And hierarchical caching has lower bandwidth usage, while distributed caching will distributed the traffic better as it will use more bandwidth in lower network levels. Also the distributed caching will cost much smaller disk space, about only several Gbytes in an institutional cache; while in the hierarchical caching system, it will need hundreds of Gbytes at top-level cache. Moreover distributed caching system can share very well the total load of system and doesn't generate hot spots with high load. Their analysis shows that in a hybrid scheme the latency greatly varies depending on the number of caches that cooperate at every network level.

Fundamental Issues on Web Caching

3.1 Performance Index

Performance indexes on web caching system can be divided into two groups: one focuses only on single cache server, another addresses global network comprised of many cache servers and clients.

The measurements on sing cache address the flowing indexes :

- Traffic Patterns. It concerns on transfer size distribution, file size distribution, and proxy traffic intensity.
- Aggregate Performance. It will measure concurrent requests, request response time, etc.
- Hit Analysis. It concentrates on hit ratios and hit classification.

- Inbound Index. Client connect time and proxy reply time is its main topic.
- Disk Storage Subsystem. It concerns on disk traffic intensity, concurrent disk requests and disk response time.
- CPU and Memory utilization The global Index of cache servers will investigate the following measurements:
- Workload Analysis. It will analyze response time of different content types, content distribution based on type and content distribution based on time of day.
- Network Utilization. It includes average bandwidth consumption, latency and HOPs
- Outbound Index . It concerns on concurrent outgoing connections, proxy connect time and server reply time.
- Extra network traffic created by proxy .
- Useful traffic generated by proxy.
- Effect on proxy load. It measures extra load on the server and the fairness.
- Low-level metrics . It will investigate connection abort and caching connections.

3.2 Prefetching

All web caches can be divided into two categories from the prefetching point of view: passive caches and positive caches. A good prefetching algorithm can improve the web performance on the basis of passive caches by further reducing the web latency 10%-30%.

The prefetching task can be performed either by local caches using reference patterns to choose what to prefetch next, which is called local prefetch. Or by server-side caches using data accumulated by the server, such as information about data object. frequently requested by the clients, which is called server-hint prefetch.

There are several primary parameters for cache prefetching, such as prefetching models, popularity distributions, object size, object update pattern and lifetime, spare prefetching resources, and specified threshold. Different algorithms may take more weight on different parameters. Many prefetching strategies and algorithms are widely used trying to increase the performance.

The “Threshold algorithm” tries to balance object access frequency (which represent the benefit of keeping that object in system) and object update rate (which represent the cost of keeping an object). The “local prefetch predicate” algorithm introduced by uses another approach. It uses the missing page in the hinted reference stream that will be accessed after a given time, the predicted access time in the near future, and prefetching time for a data object as parameters. The “ Prefetching Hyperlinks” introduced by, using the server's knowledge about the hyperlink to help the individual clients for better prediction. The “Top-10” approach introduced by is based on the cooperation of clients and servers to make successful prefetch operations.

There is another similar prefetching technique used in the management of computer system or database applications to reduce the response time delays caused by I/O. In many systems and database applications, users spend a lot of time processing the pages, and the computer and I/O system are typically idle during that period. So some algorithm try to predict which pages the users will request next, and then fetch them into cache before the user ask for them.

Our study on previous work found 'that most current prefetching models are implemented on individual caches and based on the assumption that the storage is unlimited. The most considered factors are reference pattern, documents popularity; documents access probability, the time needed to prefetch the documents and some appropriate threshold value. If take the

storage capacity of individual caches into consideration, more constraints will be added to the prefetching algorithm. For example, the document size factor must be treated as one of the important parameters for the model.

In collaborative web cache architecture, more complicated algorithms should be considered, Individual cache prefetching strategy has to coordinate with the whole system. There must be an algorithm to determine which cache prefetch which document. The cooperation among different caches becomes very important, inefficient cooperation means waste of resource and would not generate a desired improvement result. The prefetching algorithms must be integrated among the caches in the system, each cache should keep the metadata of the contents of the others. Before making the prefetching decision according its own clients reference pattern or predicted future access probability, it should look at the metadata first and only prefetch the document that are not stored by any caches in the system. Or only duplicate the documents whose popularity exceeds a certain threshold.

3.3 Routing

Web caching systems tend to be composed of multiple distributed caches to improve system scalability and availability, or to leverage physical locality. The main challenge in such approaches is how to quickly locate a cache containing the desired document. The common traditional approach is to grow a caching distribution tree away from each popular server towards sources of high demand. There are two kinds of basic approaches: one is based cache routing table; another is based on hash functions. Traditional caching resolution approaches works well for very popular documents, while the performance will drop dramatically for less popular documents. To overcome such kinds of shorting comings, recently, some research groups proposed some extended algorithms based on those two traditional caching routing approaches. While other groups tried from different views, such as trying the routing resolution at network-layer level not at the

application-layer level. The main idea is to integrate cache routing into the network layer, which will result in shorter search paths and fewer application-level check.

The basic ideas of web caching routing based cache routing table is that: at first, web client request a page from one of the shared caches, let's say A; if the cache can't satisfy request, it will use ICP to queries all other sibling caches; if sibling satisfy, the shared cache A will use HTTP to request the object and store a copy in it and then sends the object to client; if no sibling satisfy, shared cache A fetches object from originating web server, store a copy in its cache, and then pass it to the client. Actually, these kinds of routing algorithm are based on the principles of ICP (Internet Caching Protocol).

Based on this principle, researchers have developed many evolved algorithms. For example, adaptive web caching, which build a distribution tree for each server and organize the caches into overlapping multicast group. To optimize the path of meshes to query, adaptive caching use CRP (Content Routing Protocol) to determine the likely direction of origin for the content.

The main idea of hashing functions based cache routing is that: all clients store a common hash function that maps URLs to a hash space. Hash space is partitioned and each set in the partition is associated with one of the sibling caches. When client desires an object, it first hashes object's URL, then requests the object from the sibling cache whose set contains the hash value. If cache can't satisfy, it retrieves object from originating server, place a cache in its cache, and forward the object to the client

They also have some extension: such as Cache Array Protocol (CARP) which use a hash function based upon the “array membership list” and URL to provide the exact cache location of the object. It was developed by Microsoft. Also the Summary Cache system: each cache keeps a summary of

the URLs of cached document at each participating cache and check these summaries before sending any queries.

3.4 Replacement Decision

Web caching is effective because a few resources are requested often by many users, or repeatedly by a specific user. This phenomenon is known as locality of reference.

LRU (least recently used) is the most widely used and important replacement algorithm ever developed for main memory and disk caching. LRU exploits temporal locality of reference, keeping the recently used while dropping the least recently used objects. LRU is simple to implement, robust and effective in paging scenarios. However, as LRU focuses on the recently used and equal sized objects, it is not suitable for web caching,

GreedyDual-Size is an interesting algorithm that combines recency of reference with object size and retrieval cost. Based on its size and cost, an object is given a initial value when it first gets in or gets a new reference. Then the value decreases little by little with time ifit gets no more reference. Object with least value is the candidate to be replaced. One problem with GreedyDual-Size is that the difference of popularity has not been used.

Least Relative Value (LRV) is another replacement algorithm for proxy cache. Although it has taken into account size, recency and frequency, however, it has been criticized as heavy parameterization and high overhead in implementation. There are several frequency-based extensions to LRU in research of traditional paging or buffering scenarios, including FBR (Frequency-based Replacement), LRU-K and 2Q. , and Segmented LRU. However, none of them take into account the variable sizes of web objects.

3.5 Cache coherence mechanisms

Every Web cache must update pages in its cache so that it can give users pages that are as fresh as possible. Caching and the problem of cache coherency on the WWW are similar to the problems of caching in distributed

file systems. Current coherence mechanisms may be divided into three types: Client initiation, Server initiation and Hybrid initiation.

For client initiation, there are several kinds of methods to maintain cache coherence. In PER (Poll Each Read) algorithm, a client asks the object's server if the object is valid before accessing a cached object. If so, the server responds affirmatively; otherwise, the server sends the current version. The Poll Periodically Algorithm is based on Poll Each Read, but it assumes that cached objects remain valid for at least a timeout period of t seconds after a client validates the data. It is difficult to choose the appropriate the timeout period. The Piggyback Cache Validation method capitalizes on requests sent from the proxy cache to the server to improve coherency. The TTL approach maintains cache consistency using the copy's time-to-live attribute. A cached copy is considered valid until its TTL expires. It is often hard to assign an appropriate time-to-live for a document. The adaptive TTL approach handles the problem by adjusting a document's time-to-live based on observations of its lifetime.

There are also various approaches in server initiation coherence mechanisms. In the Callback algorithm servers keep track of which clients are caching which objects. When an object will be modified, the server notifies the relative clients and then wait for the reply. When receiving the reply, the server finishes modifying the object. In the PSI mechanism, servers group resources into volumes. Each volume has a unique identifier and a current version. Whenever a resource changes within a volume, the server updates the volume version and records the resource(s) that changed between the previous and current version. Each proxy client maintains the current set of server volume identifiers and versions for resources in its cache.

Hybrid initiation mechanisms integrate the client initiation and server initiation. The Lease approach allows servers to make progress while maintaining strong consistency despite failure. If a client or network failure

prevents a server from invalidating a client's cache, the server need only wait until the lease expires before performing the write. To read an object, a client first checks if the lease has expired. The Volumes Leases algorithm uses a combination of object lease and volume lease. Object lease is associated with individual data objects, and volume lease is associated with a collection of related objects on the same server. The Adaptive Leases approach focuses on the leases approach that balances these tradeoffs and presents analytical models and policies for determining the optimal lease duration.

As the number of Internet users increased exponentially, the problems of Web traffic, increasing bandwidth demand and server overloading became more and more serious. Great efforts have been made to improve the Web performance, such as reducing web latency and alleviating server bottleneck. There are many different approaches existed try to address the above problems. Web cache is a well-accepted technique for reducing access time and saving bandwidth to compensate for the rapidly growing demand of the Web. This paper is the result of a survey of various papers and web sites. It provides a whole picture of the current Web caching techniques and trends. In this paper, we discussed the fundamental issues for the Web cache, such as the different schemas of the collaborative caching architecture, performance index of single proxy cache and global network of cache servers, prefetching algorithm, cache routing, replacement strategy, cache coherence mechanisms and a bird's view of the current proxy cache products. By surveying the previous work we found that there are still some open issues remained in Web caching. Among the technical issues are the proxy cache server placement, dynamic object caching, security issue, etc. These are challenges for both researchers and venders.

万维网缓存概述

随着Internet的持续普及，加上HTTP协议的突然出现，Web用户正遭受网络拥塞和服务器超载的困扰。因此，对Web基础结构的可扩展性提出了要求。

万维网（WWW）可以看作是一个大型的分布式信息系统，用户可以在万维网中访问共享的数据对象。它的使用成本是非常便宜的，并且使用WWW的信息访问比使用其他任何方式都更快。此外，WWW拥有吸引广泛兴趣的文档，例如新闻，教育，科学研究，体育，娱乐，股市增长，旅行，购物，天气，地图等。尽管Internet主干网容量每年增加60％，但随着越来越多的信息服务转移到Web上，对带宽的需求可能会超过供应。如果不为快速增长带来的问题采取某种解决方案，则WWW将会变得过于拥挤，最终将失去其整体吸引力。Web代理缓存是提高Web性能的流行技术。将流行的对象缓存在靠近客户端的位置已被认为是缓解Web服务瓶颈，减少Internet上的流量并提高WWW系统的可扩展性的有效解决方案之一。

WWW代理缓存系统尝试以三种方式提高性能。首先，缓存尝试减少与获取Web文档相关的用户感知的延迟。由于代理缓存通常比内容提供者更接近客户端，因此可以减少延迟。

其次，缓存将尝试降低来自Web服务器的网络流量。可以降低网络负载，因为与由内容提供者提供的文档相比，从缓存提供的文档通常遍历更少的网络。最后，代理缓存可以减少对内容提供者的服务需求，因为缓存命中不需要涉及内容提供者。它还可以降低访问提供商的运输成本。

对于单个缓存，主要问题是：应关注预缓存，文档替换策略，缓存性能指标，一致性等。但是，对于那些多重缓存体系结构，它们不仅应关注上述内容，还应关注更复杂的内容。例如多个缓存的组织，缓存请求的路由问题，缓存服务器之间的协作策略，缓存服务器在整个广域网中的位置等。协作缓存体系结构可以实现。分为三大类，即分层，分布式和混合。

2.1分层缓存架构

分层缓存架构最初是在Harvest项目中提出的。在这种类型的方案中，缓存系统具有树结构，其中每个节点都指向其父节点（即，如果在缓存中发生缓存未命中，它将请求转发到其父节点。这一直持续到我们到达根节点为止。然后，如果根服务器无法满足请求，则其与源服务器联系）当找到文档时，无论是在高速缓存中还是在原始服务器上，它都以相反的方向传播，并在其路径上的每个中间高速缓存中保留一个副本。这个结构是。符合当前Internet，每个Internet都带有ISP水平。它可以将受欢迎的文档传播到需求中。但是，该体系结构存在以下问题，例如每个层次结构级别都会在处理所有请求时引入额外的延迟，因为在每个级别上复制文档，文档的存储将存在冗余，高级缓存可能会成为瓶颈，并且长时间的排队延迟。

2.2分布式缓存架构

使用分布式缓存体系结构时，无需设置任何中间缓存，并且在网络边缘只有机构缓存可以合作为彼此的未命中服务。由于没有中间缓存来存储和集中较低级别缓存请求的所有文档，因此机构缓存需要其他机制来共享包含的文档。其中一些机制是：

- 机构缓存可以向其他合作机构缓存查询导致本地丢失的文档（通常使用“缓存间协议ICP [26]”来完成）。但是，使用基于查询的方法可能会显着增加带宽消耗和客户端所经历的等待时间，因为缓存需要轮询所有协作的缓存并等待最慢的缓存来回答。
- 机构缓存可以保留其他合作缓存的内容的摘要，从而避免了查询/轮询的需要。内容摘要/摘要在机构缓存之间定期交换。为了使摘要/摘要的分发更加有效和可扩展，可以使用中间节点的分层基础结构。但是，此分层基础结构仅分发有关文档位置的信息，而不存储文档副本。
- 机构缓存可以使用将客户请求映射到某个缓存的哈希函数进行协作。使用这种方法，就不会在不同的缓存中复制相同文档的副本，也不需要缓存来了解彼此的内容。但是，在所有协作的缓存中只有一个文档的单个副本，将这种方法限制为具有良好互连的缓存的本地环境。

2.3混合缓存架构

在混合方案中，缓存使用分布式缓存与相同级别或更高级别的其他缓存协作。ICP是一个典型的例子。从具有最低RTT的父/邻居缓存中获取文档。Rabinovich 建议限制相邻缓存之间的协作，以避免从遥远或较慢的缓存中获取文档，而这些文档本可以以较低的成本直接从原始服务器中检索到。

Pablo和Christian提出了一个数学模型来分析上述三种方案的一些重要性能参数。他们发现分层缓存系统的连接时间较短，而分布式缓存系统的传输时间较短。分层缓存具有较低的带宽使用率，而分布式缓存将在较低的网络级别使用更多带宽，从而更好地分配流量。而且，分布式缓存将占用更少的磁盘空间，在机构缓存中大约只有几GB。而在分层缓存系统中，顶级缓存将需要数百GB。而且，分布式缓存系统可以很好地共享系统的总负载，并且不会产生高负载的热点。

Web缓存的基本问题

3.1性能指标

Web缓存系统上的性能指标可以分为两类：一组仅关注单个缓存服务器，另一组针对由许多缓存服务器和客户端组成的全局网络。

对预缓存的测量可解决流动索引：

- 交通模式。它涉及传输大小分布，文件大小分布和代理流量强度。
- 综合绩效。它将测量并发请求，请求响应时间等。
- 命中分析。它着重于命中率和命中分类。
- 入站索引。客户端连接时间和代理回复时间是其主要主题。
- 磁盘存储子系统。它涉及磁盘流量强度，并发磁盘请求和磁盘响应时间。
- CPU和内存利用率缓存服务器的全局索引将调查以下度量：
- 工作量分析。它将分析不同内容类型的响应时间，基于类型的内容分发以及基于一天中时间的内容分发。
- 网络利用率。它包括平均带宽消耗，延迟和HOP
- 出站索引。它涉及并发传出连接，代理连接时间和服务器回复时

间。

- 代理创建的额外网络流量。
- 代理产生的有用流量。
- 对代理负载的影响。它测量服务器上的额外负载和公平性。
- 低级指标。它将调查连接中止和缓存连接。

3.2预缓存

从预抓取的角度来看，所有Web缓存都可以分为两类：被动缓存和肯定缓存。好的预取算法可以通过将Web延迟进一步降低10％-30％，从而在被动缓存的基础上提高Web性能。

可以由本地缓存使用参考模式选择下一步要预取的内容（称为本地预取）来执行预取任务。或通过服务器端缓存使用服务器积累的数据，例如有关数据对象的信息。客户端经常请求的内容，称为服务器提示预取。

缓存预取有几个主要参数，例如预取模型，流行度分布，对象大小，对象更新模式和生存期，备用预取资源以及指定的阈值。不同的算法可能对不同的参数施加更大的权重。为了提高性能，许多预取策略和算法被广泛使用。

“阈值算法”试图平衡对象访问频率（代表将对象保留在系统中的好处）和对象更新率（代表保留对象的成本）。引入的“本地预取谓词”算法使用了另一种方法。它使用提示参考流中的缺失页面（在给定时间之后将被访问），不久的将来的预计访问时间以及数据对象的预取时间作为参数。引入的“预取超链接”，利用服务器有关超链接的知识来帮助各个客户端进行更好的预测。引入的“Top-10”方法是基于客户端和服务器的协作来进行成功的预取操作。

在计算机系统或数据库应用程序的管理中还有另一种类似的预取技术，用于减少由I/O引起的响应时间延迟。在许多系统和数据库应用程序中，用户花费大量时间来处理页面，并且计算机和I/O系统通常在此期间处于空闲状态。因此，一些算法尝试预测用户接下来将请求哪些页面，然后在用户请求它们之前将其提取到缓存中。

我们对先前工作的研究发现，当前大多数预取模型是在单个缓存上实现的，并且基于存储不受限制的假设。最受考虑的因素是参考模式，文件受欢迎程度；文档访问概率，预取文档所需的时间和一些适当的阈值。如果考虑单个缓存的存储容量，则将更多约束添加到预取算法中。例如，必须将文档大小因子视为模型的重要参数之一。

在协作式Web缓存体系结构中，应考虑更复杂的算法。各个缓存的预取策略必须与整个系统协调。必须有一种算法来确定哪个缓存预取哪个文档。不同缓存之间的协作变得非常重要，效率低下的协作意味着资源浪费，并且不会产生预期的改进结果。预取算法必须集成在系统中的缓存之间，每个缓存应保留其他内容的元数据。在根据其自己的客户端参考模式或预测的未来访问概率做出预取决定之前，它应首先查看元数据，并且仅预取未由系统中的任何缓存存储的文档。

3.3路由

Web缓存系统往往由多个分布式缓存组成，以提高系统的可扩展性和可用性，或利用物理位置。这种方法的主要挑战是如何快速定位包含所需文档的缓存。常见的传统方法是从每个流行的服务器向高需求的源增长一个缓存分发树。基本方法有两种：一种是基于缓存的路由表；另一个基于哈希函数。传统的缓存解析方法对于非常受欢迎的文档非常有效，而对于不太受欢迎的文档，性能将急剧下降。为了克服这种不足，最近，一些研究小组基于这两种传统的缓存路由方法提出了一些扩展算法。主要思想是将缓存路由集成到网络层中，这将导致更短的搜索路径和更少的应用程序级检查。

基于Web缓存路由的缓存路由表的基本思想是：首先，Web客户端从一个共享缓存中请求一个页面，比如A；如果缓存不能满足请求，它将使用ICP查询所有其他同级缓存；如果兄弟姐妹满意，则共享缓存A将使用HTTP来请求对象并在其中存储副本，然后将该对象发送给客户端。如果没有兄弟姐妹满意，则共享缓存A从原始Web服务器获取对象，将副本存储在其缓存中，然后将其传递给客户端。实际上，这些路由算法都是基于ICP（Internet缓存协议）的原理。

基于此原理，研究人员开发了许多进化的算法。例如，自适应Web缓存，它为每个服务器建立一个分发树，并将缓存组织为重叠的多播组。为了优化要查询的网格的路径，自适应缓存使用CRP（内容路由协议）来确定内容的原始方向。

基于哈希函数的缓存路由的主要思想是：所有客户端都存储一个公共的哈希函数，该函数将URL映射到哈希空间。哈希空间已分区，分区中的每个集合都与同级缓存之一相关联。当客户端需要一个对象时，它首先对对象的URL进行哈希处理，然后从其集合中包含哈希值的同级缓存中请求该对象。如果缓存不能满足要求，它将从原始服务器检索对象，将缓存放入其缓存中，然后将对象转发给客户端

它们也有一些扩展：例如缓存阵列协议（CARP），它使用基于“数组成员列表”和URL的哈希函数提供对象的确切缓存位置。它是由微软公司开发的。还有摘要缓存系统：每个缓存都会在每个参与的缓存中保留缓存文档URL的摘要，并在发送任何查询之前检查这些摘要。

3.4替换决策

Web缓存之所以有效，是因为许多用户经常请求某些资源，或者特定用户反复请求一些资源。这种现象称为参考位置。

LRU（最近最少使用）是有史以来为主内存和磁盘缓存开发的使用最广泛，最重要的替换算法。LRU利用参考的时间局部性，在删除最近最少使用的对象的同时保留最近使用的对象。LRU易于实现，在寻呼场景中功能强大且有效。但是，由于LRU专注于最近使用且大小相等的对象，因此不适合用于Web缓存，

GreedyDual-Size 是一种有趣的算法，将引用的新近度与对象大小和检索成本相结合。根据对象的大小和成本，对象在首次进入或获得新引用时会被赋予初始值。然后，如果没有更多参考，该值将随着时间一点一点的减少。值最小的对象是要替换的候选对象。GreedyDual-Size的问题之一是尚未使用流行度差异。

最小相对值（LRV）是代理缓存的另一种替换算法。尽管它考虑了大小，新近度和频率，但是在实施中却被批评为参数设置繁重且开销高。

在传统寻呼或缓冲场景的研究中，LRU有几种基于频率的扩展，包括FBR（基于频率的替换），LRU-K，2Q和分段LRU。但是，它们都不考虑Web对象的可变大小。

3.5缓存一致性机制

每个Web缓存都必须更新其缓存中的页面，以便它可以为用户提供尽可能新鲜的页面。缓存和WWW上的缓存一致性问题类似于分布式文件系统中的缓存问题。当前的一致性机制可以分为三种类型：客户端启动，服务器启动和混合启动。

对于客户端启动，有几种方法可以保持高速缓存的一致性。在PER（轮询每次读取）算法中，客户端在访问缓存的对象之前会询问对象的服务器该对象是否有效。如果是，则服务器作出肯定响应；否则，服务器将响应。否则，服务器将发送当前版本。“定期轮询”算法基于“每次读取轮询”，但是它假定在客户端验证数据后，缓存的对象至少在t秒的超时时间内保持有效。很难选择合适的超时时间。搭载缓存验证方法利用从代理缓存发送到服务器的请求来提高一致性。TTL方法使用副本的生存时间属性来保持缓存一致性。缓存的副本在其TTL过期之前才被视为有效。通常很难为文档分配适当的生存时间。自适应TTL 方法通过根据对文档生命周期的观察来调整文档的生存时间来解决该问题。

服务器启动一致性机制中也有各种方法。在回调算法中，服务器跟踪哪些客户端正在缓存哪些对象。当对象将被修改时，服务器会通知相关的客户端，然后等待答复。收到答复后，服务器将完成对象的修改。在PSI机制中，服务器将资源分组为卷。每个卷都有唯一的标识符和当前版本。每当卷中的资源发生更改时，服务器都会更新该卷的版本并记录在先前版本和当前版本之间更改的资源。每个代理客户端在其缓存中维护服务器卷标识符和版本的当前集合。

混合启动机制将客户端启动和服务器启动集成在一起。租用方法允许服务器在出现故障的同时保持强大一致性的同时取得进展。如果客户端或网络故障阻止服务器使客户端的缓存无效，则服务器只需等到租约到期就可以执行写操作。要读取对象，客户端首先检查租约是否到期。

体租赁算法使用对象租赁和体租赁的组合。对象租约与单个数据对象相关联，而批量租约与同一服务器上相关对象的集合相关联。自适应租赁方法着重于平衡这些折衷的租赁方法，并提出用于确定最佳租赁期限的分析模型和策略。

随着Internet用户数量成倍增加，Web流量，带宽需求增加和服务器超载的问题变得越来越严重。为了提高Web性能，已经做出了巨大的努力，例如减少了Web延迟并减轻了服务器瓶颈。为了解决上述问题，存在许多不同的方法。Web缓存是一种广为接受的技术，可减少访问时间并节省带宽以补偿Web的快速增长的需求。本文是对各种论文和网站进行调查的结果。它概述了当前的Web缓存技术和趋势。在本文中，我们讨论了Web缓存的基本问题，例如协作缓存体系结构的不同架构，单个代理缓存和缓存服务器的全局网络的性能指标，预取算法，缓存路由，替换策略，缓存一致性机制以及当前代理缓存产品的鸟瞰图。通过调查以前的工作，我们发现Web缓存中仍然存在一些未解决的问题。其中的技术问题是代理缓存服务器的放置，动态对象缓存，安全性问题等。这对于研究人员和供应商都是挑战。