《大型系统/应用软件课程设计》报告

项目题目：图书管理信息系统

班　　号：**F0503602**

项目组名称：**www.65536.cn Developer**

项目组成员：　　姓名　　学号

石君霄　**5050369043**

指导教师：刘海涛

开始日期：**2008-03-16**

完成日期：**2008-05-07**

提交日期：**2008-05-08**

上海交通大学信息安全工程学院

图书管理信息系统 软件需求说明书

版本历史

2008-03-30 文档创建

2008-04-06 增加文献入库工作台、添加部署图

2008-04-09 更新用例图

1 引言

1.1 编写目的

本报告对传统图书馆的微观工作进行全面、详细的调查了解，运用结构化的方法进行系统的需求分析，得出系统的信息需求、功能需求、信息和功能的关系等。整个系统涵盖了传统图书馆微观工作体系中采编和外借阅览部门的工作，使图书馆的工作效率得到提高，走向图书馆自动化。

1.2 背景

图书馆自动化，主要是指以计算机为主体，利用通讯技术和高密度存贮技术，对图书馆工作的各个环节（采访、编目、流通阅览、信息检索、图书馆管理等）实行程序控制下的自动管理，从而提高图书馆的工作效率，减轻工作人员的劳动量，加速文献流通速度，向用户提供更多信息。

- 系统名称：图书管理信息系统
- 任务提出者：刘海涛
- 开发者：F0503602 石君霄 5050369043
- 用户：传统图书馆

本系统与其他系统的关系：

- 与会员、员工、学生等信息管理系统连接，以便将会员卡、考勤卡、学生证等用作读者证
- 向图书馆公共查询系统提供接口，以便读者通过 Web 远程查询书目
- 与《中国图书分类法》数据库连接，以提供采编工作效率、实现采编工作标准化

1.3 定义

图书管理信息系统：图书管理信息系统是指传统图书馆中，进行书目数据自动化和图书馆事务处理的图书馆自动化集成管理系统。

传统图书馆：收藏资讯、原始资料、资料库并提供相关服务的地方，其收藏物是看得见、摸得着的图书、光盘、连续性出版物、地图、录像带等。

1.4 参考资料

a) 《图书馆微观工作体系》，北京大学，http://www.im.pku.edu.cn/jpk/

b) 《图书馆》，维基百科，http://zh.wikipedia.org/wiki/图书馆

2 任务概述

2.1 目标

图书管理信息系统是基于先进的网络软件技术和大容量高速硬盘系统基础上的图书馆自动化解决方案，通过集中式书目数据库、中图法标准采编系统、带有二维码的读者证等，达到降低成本、提高效率、改进读者服务的目的。它通过与图书馆所在单位现有的信息管理系统相连接，保证图书馆微观管理与所在单位同步发展。

2.2 用户的特点

- 图书采编馆员：具有足够的计算机操作水平和快速汉字录入能力，习惯使用键盘。软件界面应尽量方便操作，应支持全键盘操作，以提高录入速度。
- 借还书、读者服务馆员：具有足够的计算机操作水平。软件界面应尽量简单，以避免差错、提高工作效率。
- 文献典藏馆员：具有初步的计算机或手持终端操作水平。手持终端应使用较大的字体，界面要支持单手键盘操作。
- 读者：具有或不具有计算机操作水平，可能视力不佳。读者自助查询界面应尽量简单直观，使用较大的字号，提供一定的向导。

- 自动化部人员：熟悉软硬件操作，能够进行系统升级、故障排除、日常备份等。能够与开发者进行交流，能够完成对图书采编人员、借还书服务人员的培训。

2.3 假定和约束

假设：以下假设是必要非充分条件

- 图书采编人员、借还书服务人员经过了足够的培训并且满足要求
- 自动化部人员能对硬件、软件、数据库进行日常维护、备份等操作
- 图书馆与开发者进行沟通、配合
- 如果需要与会员、员工、学生等信息管理系统连接，假设连接能够顺利实施

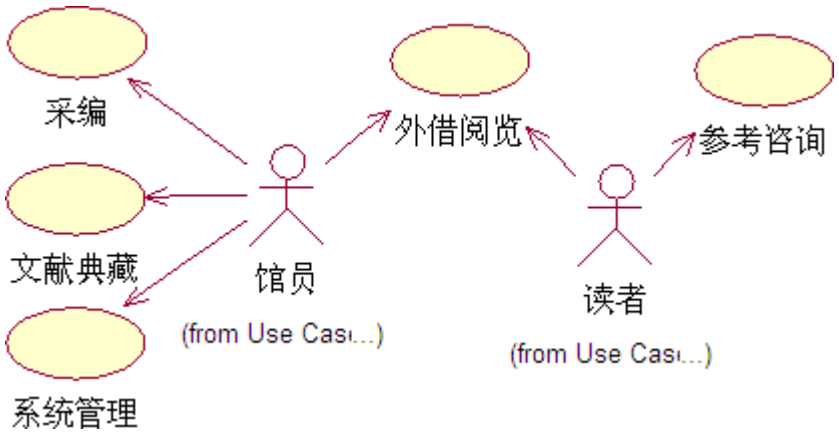
约束：

- 开发时限：2008 年 3 月 14 日～2008 年 4 月 30 日
- 开发经费：不多于人民币 15 万元 （含数据库服务器、资料采购）

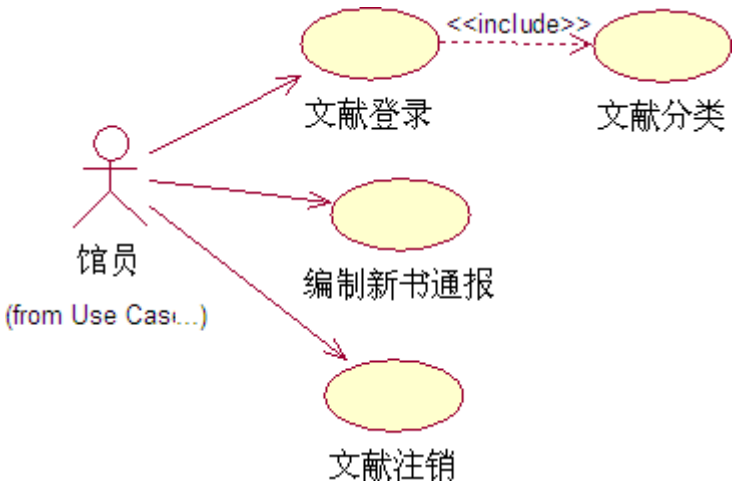
3 需求规定

3.1 对功能的规定

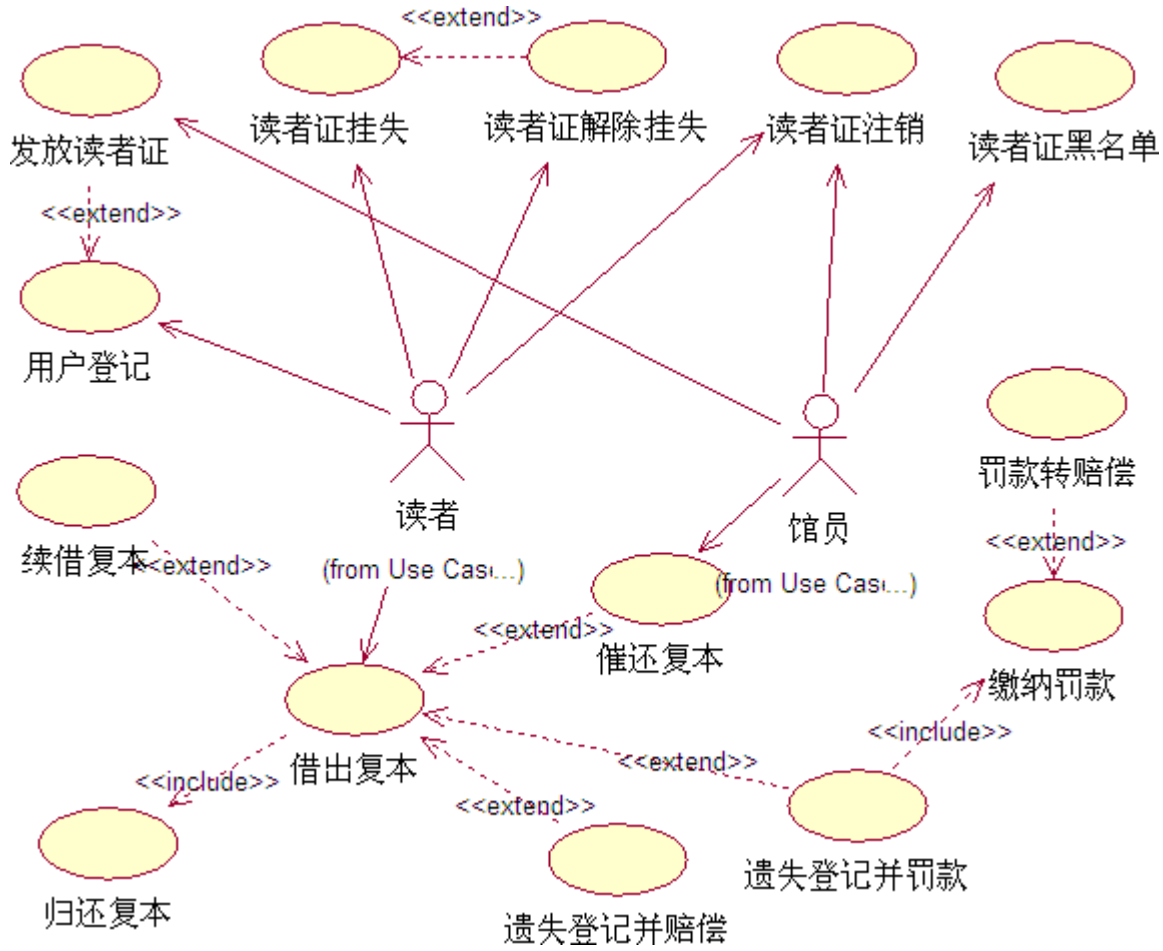
总用例图：



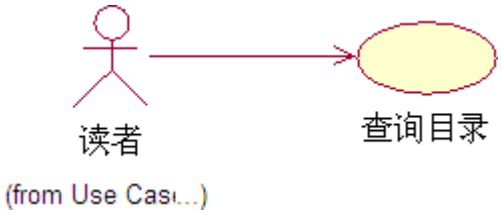
采编功能：



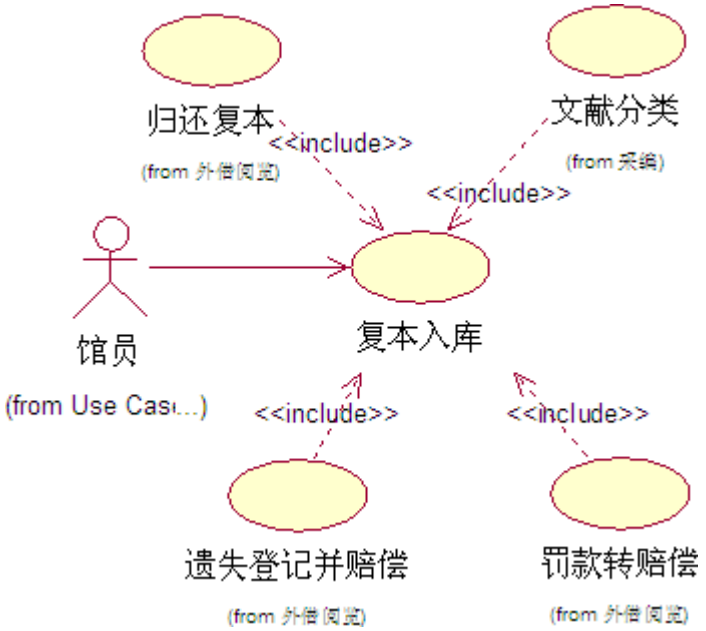
外借阅览功能



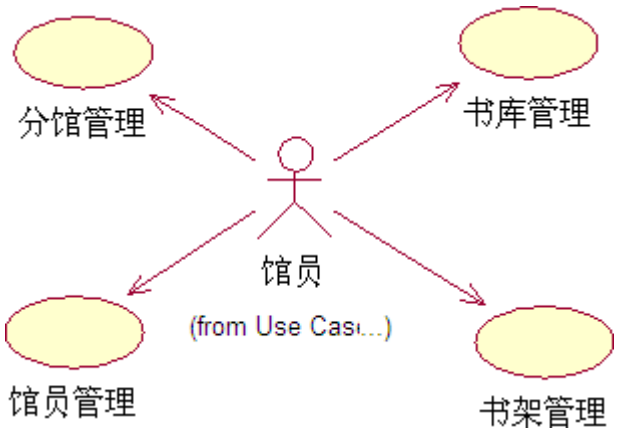
参考咨询功能：



文献典藏功能：

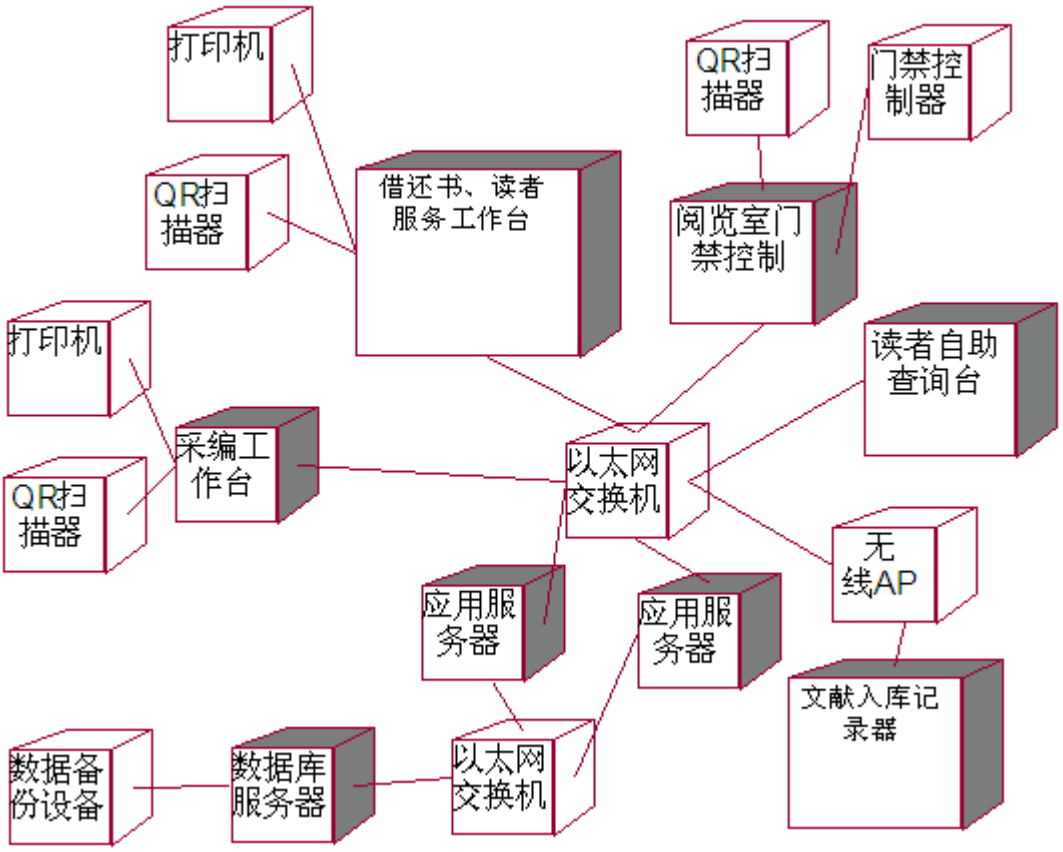


系统管理功能：



4 运行环境规定

4.1 设备



- 数据库服务器
  型号：Dell PowerEdge 2950 III
  尺寸：机架式 2U
  处理器：1 颗主频为 3.0GHz 的英特尔至强 5200 系列四核处理器
  内存：4 个 1GB
  存储：3.5 英寸 250GB SATA 硬盘×2 块，RAID0 镜像备份
- 应用服务器
  型号：Dell PowerEdge R300
  尺寸：机架式 1U
  处理器：1 颗主频为 2.83GHz 的英特尔至强 3300 系列四核处理器
  内存：2 个 1GB

存储：73GB SAS 硬盘
网络：双网卡，一块连接数据库服务器、另一块对外
（规模扩大时，可以增加应用服务器数量，并配置负载均衡）

- 采编工作台计算机，借还书、读者服务工作台计算机
  处理器：1.6GHz 或以上
  内存：1GB
  存储：150MB 空闲空间
  特殊设备：QR 二维码扫描器、打印机
- 读者自助查询台计算机
  处理器：1.6GHz 或以上
  内存：1GB
  存储：100MB 空闲空间
- 大门或阅览室门禁控制器
  处理器：ARM9
  内存：32MB
  存储：32MB Flash
- 文献入库记录器
  机型：Android 手持终端
  特殊设备：摄像头、WiFi 无线接入
- 百兆以太网交换机
- 无线 AP（WiFi 无线接入点）

4.2 支持软件

- 数据库服务器
  操作系统：RedHat Enterprise Linux 4
  数据库软件：MySQL Enterprise Server 5.0
- 应用服务器
  操作系统：Ubuntu Server 7.10
  其他软件：Python 2.5、Python-MySQL
- 采编工作台计算机，借还书、读者服务工作台计算机，读者自助查询台计算机
  操作系统：Windows Vista Home Basic
  中间件：.Net Framework 3.5
- 大门或阅览室门禁控制器
  操作系统：Linux
- 文献入库记录器
  软件平台：Linux、Java、Andriod、QR 二维码识别软件

4.3 接口

如果需要连接会员、员工、学生等信息管理系统，则使用 XMLRPC 方式调用。
如果增加 Web 查询系统，查询系统用 XMLRPC 调用应用服务器。

4.4 控制

服务器每天开馆时由自动化部人员负责开启，闭馆时执行每日备份并关机。软件在各工作台开机时自动启动，由用户操作，闭馆时手动关闭。

图书管理信息系统 数据库设计说明书

版本历史

2008-04-06 文档创建

2008-04-08 复本表、读者表等增加一些字段，调整字段次序

2008-04-15 增加读者分类、书库分类

2008-04-27 增加预约馆藏表、馆员权限表、中国图书馆分类法表

2008-04-29 增加著者生卒日期、国籍

1 引言

1.1 编写目的

图书管理信息系统有大量数据存储的要求，因此需要存储在关系型数据库里。本文档根据需求分析，定义了图书管理信息系统所需数据库的设计方案。

1.2 背景

- 数据库名称：图书管理信息系统数据库
- 使用此数据库的软件系统：图书管理信息系统应用服务端
- 任务提出者：刘海涛
- 开发者：F0503602 石君霄 5050369043
- 用户：传统图书馆

1.3 定义

- 文献：图书馆内一切可以出借或阅览的资料，包括图书、磁带、光盘、连续性出版物等；例如《操作系统教程》、《读者》2008 年第 8 期
- 连续性出版物：图书馆长期保留的报纸、杂志、期刊等
- 著者：写作图书的作者、表演磁带中节目的艺术家、制作光盘的作者、连续性出版物的编辑单位；例如“陆松年”、“《读者》编辑部”
- 丛书：汇辑多个文献而冠以总名；例如“清华大学计算机系列教材”
- 索书号：文献的编号；根据《中国图书馆分类法》编排，前半部分表示文献所属的分类、后半部分表示文献的著者、种类；每一文献在馆内有惟一的索书号；例如 TP316.81/6243
- ISBN：图书的国际标准书号，由 ISO 2108 标准定义；例如 7-121-02139-0
- ISSN：连续性出版物的国际标准序列号，由 ISO 3297 标准定义；例如 1005-7501
- 复本：馆藏的一份文献；例如图书馆内有《操作系统教程》10 册，每册称为 1 个复本

1.4 参考资料

a) 图书管理信息系统 软件需求说明书

b) International Standard Book Number，维基百科 http://en.wikipedia.org/wiki/ISBN

c) International Standard Serial Number，维基百科 http://en.wikipedia.org/wiki/ISSN

2 外部设计

2.1 标识符和状态

数据库标识符：library

2.2 使用它的程序

图书管理信息系统应用服务端

2.3 约定

数据库中的表名、列名均使用英文或英文简称表述

2.4 专门指导

MySQL 必须使用 utf8 字符集，以免造成中文乱码

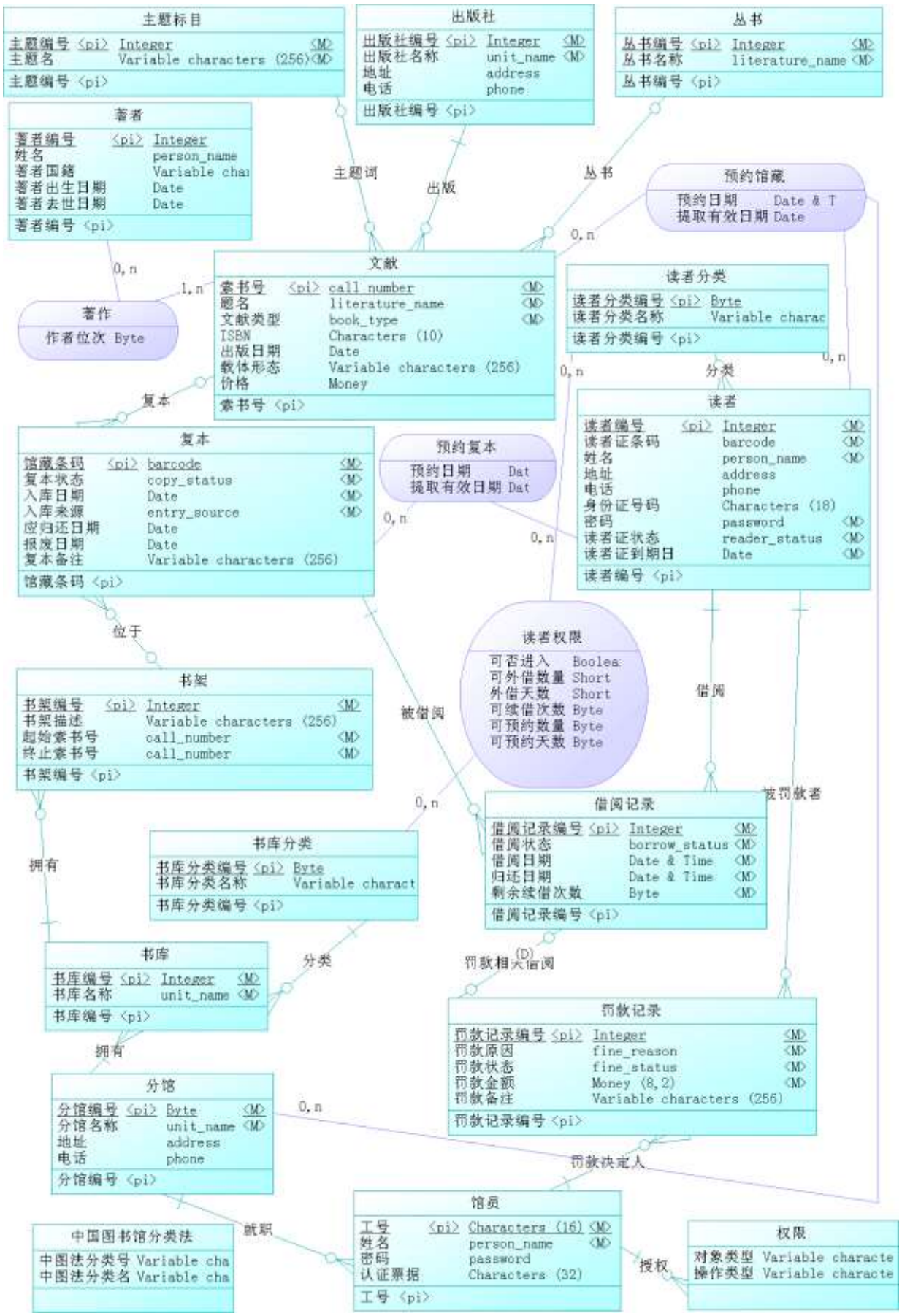
2.5 支持软件

数据库管理系统：MySQL Enterprise Server 5.0

管理工具：phpMyAdmin 2.11.5.1

3 结构设计

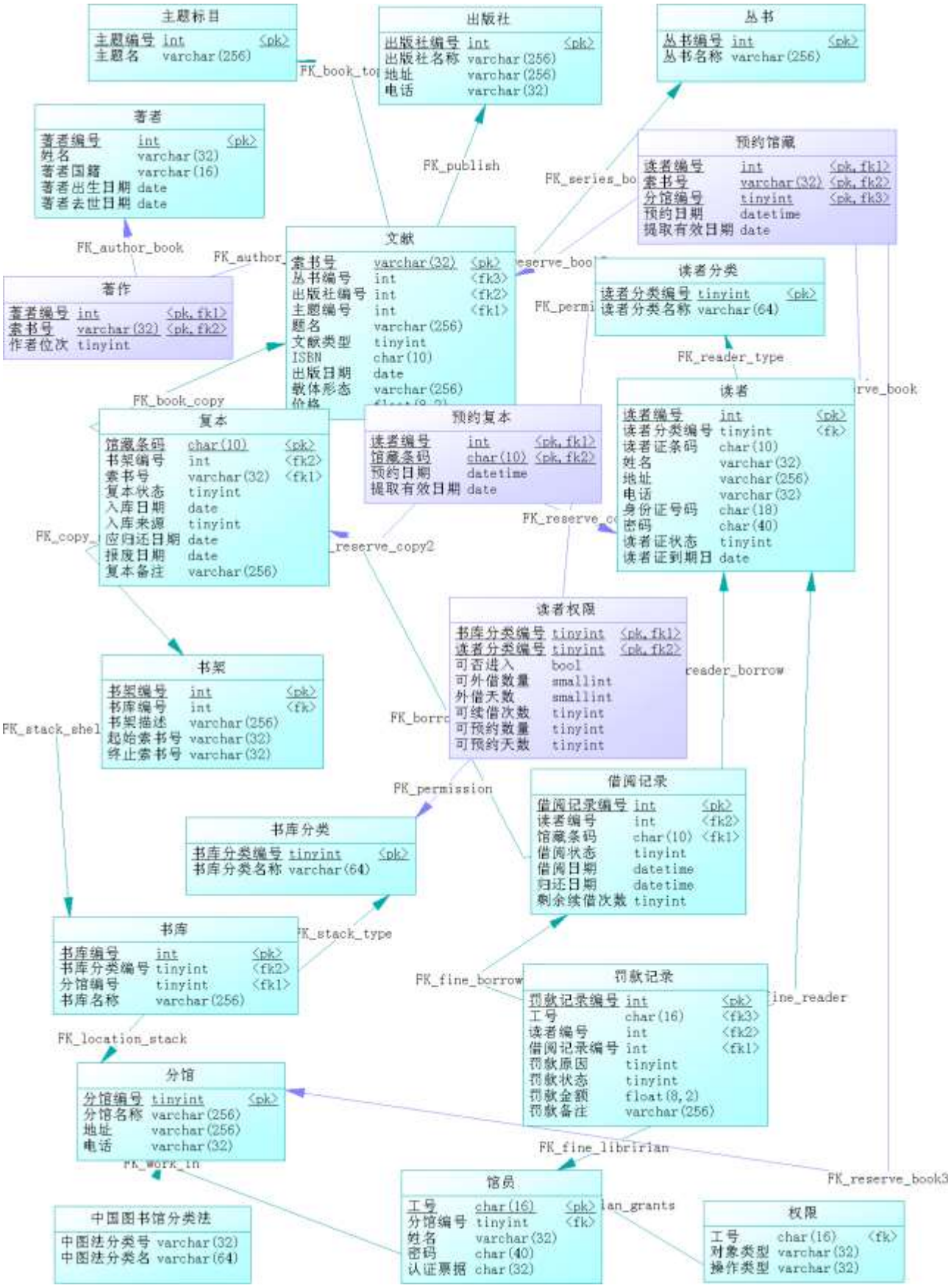
3.1 概念结构设计



3.2 逻辑结构设计

在设计概念模型时已经考虑了逻辑结构，因此概念结构设计就是逻辑结构设计

3.3 物理结构设计



注：所有表名、列名均有英文 code，程序中将使用英文 code，可以在附件 cdm、pdm 文件中查看

4 运用设计

4.1 数据字典设计

完整的数据字典在上述概念结构设计、物理结构设计图中可以看出，不再重复列出

部分数据项的解释：

- 文献-载体形态：文献的物理形态，例如“326 页，29cm”
- 文献-ISBN：图书存储 10 位 ISBN 号，连续性出版物存储 8 位 ISSN 号，其他文献置空
- 复本-应归还日期：对于被预约复本，存储预约保留截止日期；对于已借出复本，存储应归还日期，续借时予以更新
- 读者权限-可否进入：该类读者能否进入这类书库
- 读者权限-可外借数量：允许该类读者同时外借的这类书库文献数量；如果为 0，该类读者只能阅览这类书库
- 读者权限-外借天数：这类书库的文献每次外借或续借的天数；对于阅览室，置 0
- 读者权限-可续借次数：这类书库的文献借出后允许该类读者续借次数；如果为 0，不能续借
- 读者权限-可预约数量：允许该类读者同时预约的这类书库的文献数量；如果为 0，不能预约
- 读者权限-可预约天数：该类读者预约这类书库的文献后保留的天数；如果不能预约，此项无意义
- 书架-起始索书号、书架-终止索书号：安排在该书架上的文献的索书号范围；在上架时系统根据此信息给出建议，但不强制执行
- 借阅记录-剩余续借次数：这次借阅还允许续借几次，初值为书库-可续借次数，每次续借时减 1，减为 0 时不能再续借
- 权限-对象类型：指定这名馆员可以对哪种对象进行操作，例如“copy”
- 权限-操作类型：指定这名官员可以对这种对象进行什么操作，例如“create”
- 著者-国籍：以 ISO3166 规定的国家代码表示

部分数据项的代码定义：

- 文献类型

  0=其他，10=普通图书，11=外文图书，12=少数民族文字图书，13=盲文图书，30=地图，50=报纸，51=杂志期刊，110=录音磁带，111=音频 CD，130=视频磁带，131=视频 VCD，132=视频 DVD，133=视频蓝光，150=数据 CD，151=数据 DVD
- 入库来源

  0=其他，1=采购，2=捐赠，3=赔偿
- 复本状态

  0=未知，1=归还,待上架，2=上架,可借出，3=已借出，4=阅览室,不可借出，5=新书,待上架，6=损坏或遗失报废，7=其他原因报废，8=被预约
- 借阅状态

  0=借出，1=归还，2=罚款，3=预约,未借出
- 罚款原因

  0=其他，1=逾期，2=损坏，3=遗失
- 罚款状态

  0=未缴，1=已缴，2=撤销
- 读者证状态

  0=正常，1=挂失，2=过期，3=黑名单，4=注销

代码表是一种实用的设计，但不是规范的设计，也不是效率最高的设计。在这个软件中，可能需要修改的代码将存储在配置文件里、而不是数据库。

4.2 安全保密设计

馆员对于数据库有 SELECT INSERT UPDATE 权限，读者对于数据库有 SELECT 权限

图书管理信息系统 概要设计说明书

版本历史

2008-04-23 文档创建

1 引言

1.1 编写目的

本文档根据需求分析，定义了图书管理信息系统的模块划分和各模块的功能。

1.2 背景

- 软件系统名称：图书管理信息系统
- 任务提出者：刘海涛
- 开发者：F0503602 石君霄 5050369043
- 用户：传统图书馆

1.3 定义

- JSON-RPC：JSON Remote Procedure Call，一种轻量级远程过程调用协议。客户端可以调用服务器上的一个函数，并获取其返回值。
- JSON：JavaScript Object Notation，用 JavaScript 的语法对对象进行编码。
- JavaScript：ECMA262 标准定义的 ECMAScript 脚本语言的俗称。

1.4 参考资料

a) 图书管理信息系统 软件需求说明书

2 总体设计

2.1 需求规定

2.1.1 采编功能

- 文献登录：在获得文献时，馆员将文献的索书号、题名、类型、ISBN、著者、出版日期、出版社、丛书名称、载体形态、价格，以及复本的入库日期、入库来源录入系统
- 文献分类：文献登录以后，馆员对文献指定一个或多个主题标目
- 编制新书通报：馆员打印一定日期范围内的新书通报表格
- 文献注销：在复本因陈旧等原因报废时，馆员修改复本状态并记录报废日期

2.1.2 外借阅览功能

- 用户登记、发放读者证：读者凭身份证申请读者证，馆员予以审查，条件符合的（身份证号无对应读者证、或原读者证已经注销）当场制作并发放读者证，读者证归入一个分类；读者证条码号必须惟一，不能与正在使用或已注销的任何条码号重复
- 读者证挂失：读者遗失读者证后，凭身份证挂失读者证，馆员审查身份证有效性，有效的将对应读者证置为挂失状态
- 读者证解除挂失：读者找到读者证后，凭身份证和读者证解除挂失读者证，馆员审查证件有效性，有效的将读者证撤销挂失状态
- 读者证注销：读者凭身份证主动向馆员提出、读者证过期且无未归还文献和未缴罚款、读者证挂失后需要重新办理，这三种情况可将读者证注销，馆员将读者证置为注销状态
- 借出复本：读者凭读者证、已取出的复本向馆员提出外借，馆员审查读者证处于正常状态、没有未缴罚款、权限允许外借数量未满、复本允许外借，办理外借、修改复本状态、作借阅记录，如果曾经预约则删除相应预约记录
- 续借复本：读者凭读者证向馆员提出续借，馆员审查读者证处于正常状态、没有未缴罚款、借阅记录允许的续借次数未满，办理续借、更新借阅记录和复本应归还日期
- 催还复本：馆员发现借阅记录归还日期已到但是未归还，电话催促读者归还
- 归还复本：读者将复本交还馆员，馆员检查损坏情况，如损坏则登记罚款记录（可当场或事后缴纳），更新借阅状态、复本状态，如罚款当场未缴更新读者证状态为罚款
- 遗失登记并罚款：读者告知馆员复本遗失，馆员登记罚款记录（可当场或事后缴纳），更新借

阅状态、复本状态、报废日期，如罚款当场未缴更新读者证状态为罚款

- 遗失登记并赔偿：读者告知馆员复本遗失并提交同种复本，馆员审查新复本的品种是否一致，一致的予以接受，更新借阅状态、复本状态，新复本转交采编部门
- 缴纳罚款：读者将罚款款项交给馆员，馆员根据读者证查询待缴罚款、逐项收费，更新罚款状态，如全部罚款缴清且读者证未过期则将读者证置为正常状态
- 罚款转赔偿：读者向馆员提交曾遗失文献的同种复本，馆员根据读者证查询相应的待缴罚款，审查新副本的品种是否一致，一致的予以接受，更新罚款状态，新副本转交采编部门，如全部罚款缴清且读者证未过期则将读者证置为正常状态
- 读者证黑名单：长期不还书、长期不缴纳罚款，馆员将读者证列入黑名单，同一身份证不能再次进行用户登记
- 预约馆藏：读者自助或由馆员协助预约某分馆的某种文献，该文献一旦有任一复本被归还到该分馆，即为该读者保留一段时间
- 预约复本：读者自助或由馆员协助预约某个复本，该复本一旦被归还，即为该读者保留一段时间
- 预约清理：定期删除过期未提取的预约记录
- 门禁控制：读取读者证二维码，根据读者证有效性和读者权限确定是否放行读者进入书库

2.1.3 参考咨询功能

- 查询目录：读者自助根据索书号、题名、文献类型、ISBN 或 ISSN、主题标目、著者、出版社、出版日期、丛书、复本所在分馆、复本可否借出等条件查询文献目录，系统返回文献查询结果、该文献的所有复本状态

2.1.4 文献典藏功能

- 复本入库：如果是新文献——馆员选择复本所在书库，系统建议复本索书号所在书架，馆员接受建议或选定另一书架、将复本上架。如果是归还的文献——系统提示复本条码所在书架，馆员根据提示将复本归架，必要时也可以更换书架位置或迁移到另一分馆或书库。上架后，系统更新复本状态，如果该复本已被预约则状态为预约，如果满足预约馆藏要求则预约馆藏转化为预约复本。

2.1.5 系统管理功能

- 分馆管理：添加分馆，修改分馆，删除无书库也无馆员的分馆
- 书库管理：添加书库，修改书库，删除无书架的书库，设置书库所属分类
- 书架管理：添加书架，修改书架，删除无复本的书架
- 馆员管理：添加馆员，修改馆员，注销馆员
- 读者分类管理：添加读者分类，修改读者分类名称，删除无读者的读者分类（同时删除关联的权限项），设置读者分类对于每一类书库的进入和借阅权限
- 统计分析：获取各类统计信息

2.2 运行环境

见需求分析说明书

2.3 基本设计概念和处理流程

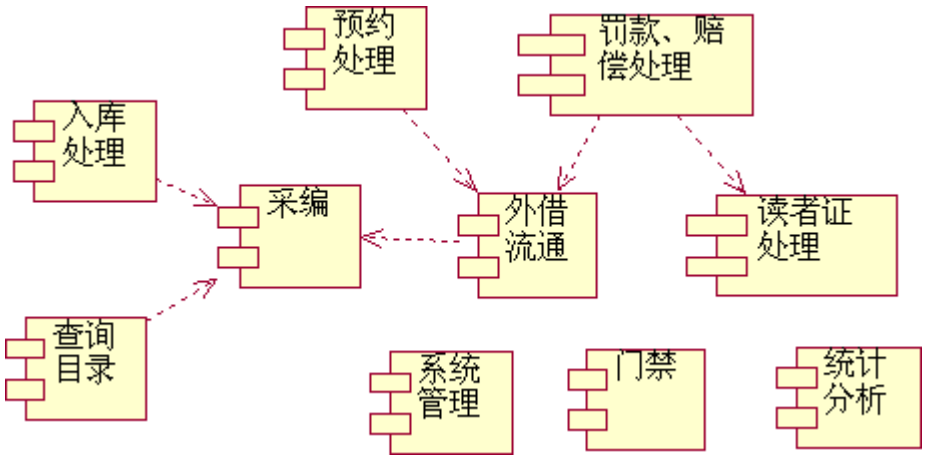
整个系统的设计围绕本系统的核心要素——复本、读者。

2.4 结构

采用企业应用三层结构设计：

- 第一层：工作台桌面程序，负责与用户交互。可以缓冲数据，但不持久保留数据。
- 第二层：应用服务器程序，负责处理事务逻辑。根据请求运行，无状态的 API。
- 第三层：数据库服务器，负责持久存储数据。带有基本的完整性约束，没有事务逻辑功能。

工作台以 JSON-RPC 协议调用应用服务器，这种调用经封装后可以相当于本地函数的调用、（网速很快时）对工作台是透明的。



2.5 功能需求与程序的关系

	系统管理	采编	外借流通	入库处理	查询目录	读者证处理	预约处理	罚款、赔偿处理	统计分析	门禁
文献登录		√								
文献分类		√								
编制新书通报		√								
文献注销		√								
用户登记、发放读者证						√				
读者证挂失						√				
读者证解除挂失						√				
读者证注销						√				
借出复本			√							
续借复本			√							
催还复本			√							
归还复本			√							
遗失登记并罚款								√		
遗失登记并赔偿								√		
缴纳罚款								√		
罚款转赔偿								√		
读者证黑名单						√				
预约馆藏			√							
预约副本			√							
预约清理			√							
门禁控制										√
查询目录					√					
复本入库				√						
分馆管理	√									
书库管理	√									
书架管理	√									
馆员管理	√									
读者分类管理	√									
统计分析									√	

2.6 人工处理过程

备份数据库必须人工执行。复本的电话催还需要人工打电话。

2.7 尚未问决的问题

3 接口设计

3.1 用户接口

各工作台程序均使用图形界面、不支持命令行操作。

3.2 外部接口

本软件使用 Python 的 MySQLdb 库调用 MySQL 数据库服务。

本软件的升级版本可以增加对企业员工系统、馆际互借系统的外部接口。

3.3 内部接口

本软件的应用服务器使用 JSON-RPC 向工作台提供无状态的 API。API 的详情将在应用服务器的概要设计中定义。

4 运行设计

4.1 运行模块组合

每种运行都要经历相应的工作台、应用服务器、数据库。

4.2 运行控制

工作台的每种运行均由图形界面的工具按钮或菜单控制执行。门禁系统的运行由 QR 二维码扫描器控制。

4.3 运行时间

各种运行模块将尽可能快的返回，以提高系统效率。

5 系统数据结构设计

5.1 逻辑结构设计要点

本软件需要的数据结构由应用服务器的无状态 API 确定。API 的详情将在应用服务器的概要设计中定义。

5.2 物理结构设计要点

数据项存储于关系数据库，见数据库设计说明书。

5.3 数据结构与程序的关系

数据结构由应用服务器与相应的工作台共享。

6 系统出错处理设计

6.1 出错信息

应用服务器通过 JSON-RPC 响应的 error 对象发送错误信息，工作台用友好的形式将错误信息呈现给用户。

6.2 补救措施

请用户更正错误后重新提交。

6.3 系统维护设计

使用 Python 语言编写应用服务器，Python 本身的特点使应用服务器容易调试，只需在适当的地方插入 logging 即可检测系统状态。

图书管理信息系统 应用服务器 概要设计说明书

版本历史

2008-04-29 文档创建

2008-05-04 修改补充了 Book.reserve 和 ReaderType

1 引言

1.1 编写目的

本文档详细描述了应用服务器对工作台等终端提供的 JSON API 接口及其设计细节。

1.2 背景

- 软件系统名称：图书管理信息系统 应用服务器
- 任务提出者：刘海涛
- 开发者：F0503602 石君霄 5050369043
- 用户：传统图书馆

1.3 定义

- KISS 原则：“Keep It Simple, Stupid”，指出了设计的简洁性是一个主要目标、应当避免不必要的繁杂。常见的变体有“Keep It Sweet & Simple”、“Keep It Short & Simple”。
- API：Application Programming Interface，应用编程接口是一种源代码形式的接口，操作系统、库、服务通过提供 API 来支持计算机程序发出的请求。
- JSON-RPC：一种用 JSON 编码的远程过程调用协议。它是一种非常简单的协议（并且非常接近 XML-RPC），只定义了少量的数据类型和命令。

1.4 参考资料

a) 图书管理信息系统 需求分析说明书
b) 图书管理信息系统 概要设计说明书
c) KISS principle，http://en.wikipedia.org/wiki/KISS_principle
d) JSON-RPC Specification，http://json-rpc.org/wiki/specification
e) 索书号的构成，http://www.jdsy.com.cn/Article.asp?id=426

2 总体设计

2.1 需求规定

图书管理信息系统是根据三层结构设计的，工作台等终端负责显示界面并与应用服务器交互，应用服务器处理事务逻辑并操作数据库。具体的事务逻辑内容参见整个系统的概要设计说明书

2.2 运行环境

见需求分析说明书

2.3 基本设计概念和处理流程

设计在 KISS 原则与系统效率之间妥协。

应用服务器的 JSON-RPC API 接口要尽可能简单明了。

2.4 结构

应用服务器分为 4 个部分进行设计：

- 登录认证、馆员用户部分
- 分馆、书库、书架部分
- 图书、编目部分
- 复本、读者、借阅、罚款部分

2.5 功能需求与程序的关系

程序的 4 个部分要作为一个整体工作，共同完成功能需求，互相之间有明显的耦合关系。

2.6 人工处理过程

应用服务器的所有处理都是自动完成的。

2.7 尚未问决的问题

3 接口设计

3.1 用户接口

应用服务器没有直接的用户接口。

3.2 外部接口

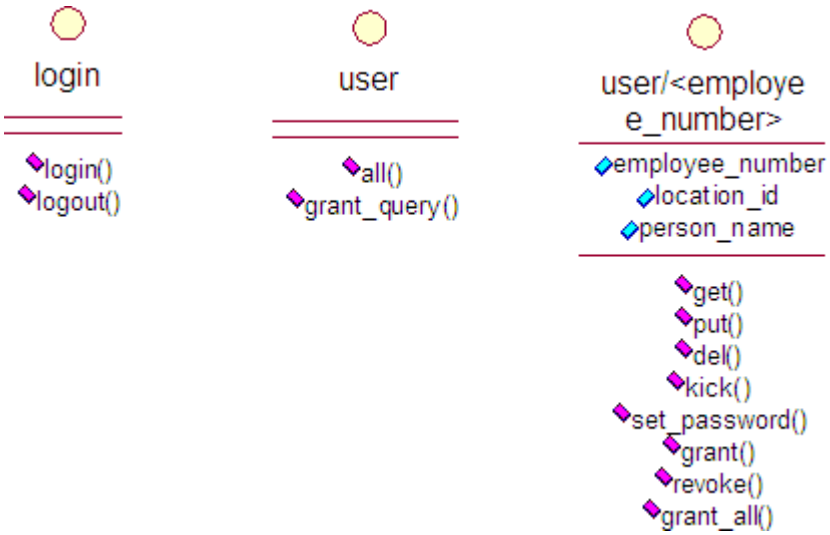
应用服务器使用 Python 的 MySQLdb 库调用 MySQL 数据库服务。

应用服务器通过 JSON-RPC 协议向工作台等终端提供无状态的 API 服务，API 的 URL、函数和返回值定义如下。

几个约定：

- “~/”表示 API 安装路径的起始目录，例如 http://api.my-library.org/api/。
- 变量名和函数名前写出了参数和返回值的类型(用 Python 类型表述)，但 Python 和 JavaScript 都是弱类型的，这里写上只为清晰定义所有接口、也方便转换成其他强类型语言。
- 以下含有参数的 URL，对应对象不存在的，都抛出 NotFoundError 异常。
- 除登录认证调用以外，所有请求都要在 Cookie 携带服务票据（Cookie 名称是 library_api_ticket），无服务票据或服务票据无效的，抛出 NotLoginError 异常。
- 需要权限的调用，当前用户权限不足的，抛出 NoGrantError 异常。
- 要求……没有……（例如要求分馆中没有书库），不满足条件时抛出 NotEmptyError 异常。
- 读者进行预约、借阅、续借等操作时：如果读者证非正常状态，抛出 ReaderStatusError 异常；如果没有相关权限，抛出 NoPermissionError 异常；如果有罚款未缴，抛出 HasFineError 异常；如果超出许可数量，抛出 OverQuotaError 异常。

3.2.1 登录认证、馆员用户部分



URL：~/login

登录认证服务

- str login(str employee_number, str password)
  执行登录认证。如果登录成功就创建一个服务票据作为返回值；登录失败就抛出 LoginFailedError 异常。此操作无需登录。
- None logout()
  注销当前用户，即将服务票据作废。

URL：~/user

表示所有馆员（用户）

- list<User> all()
  获取所有用户对象。

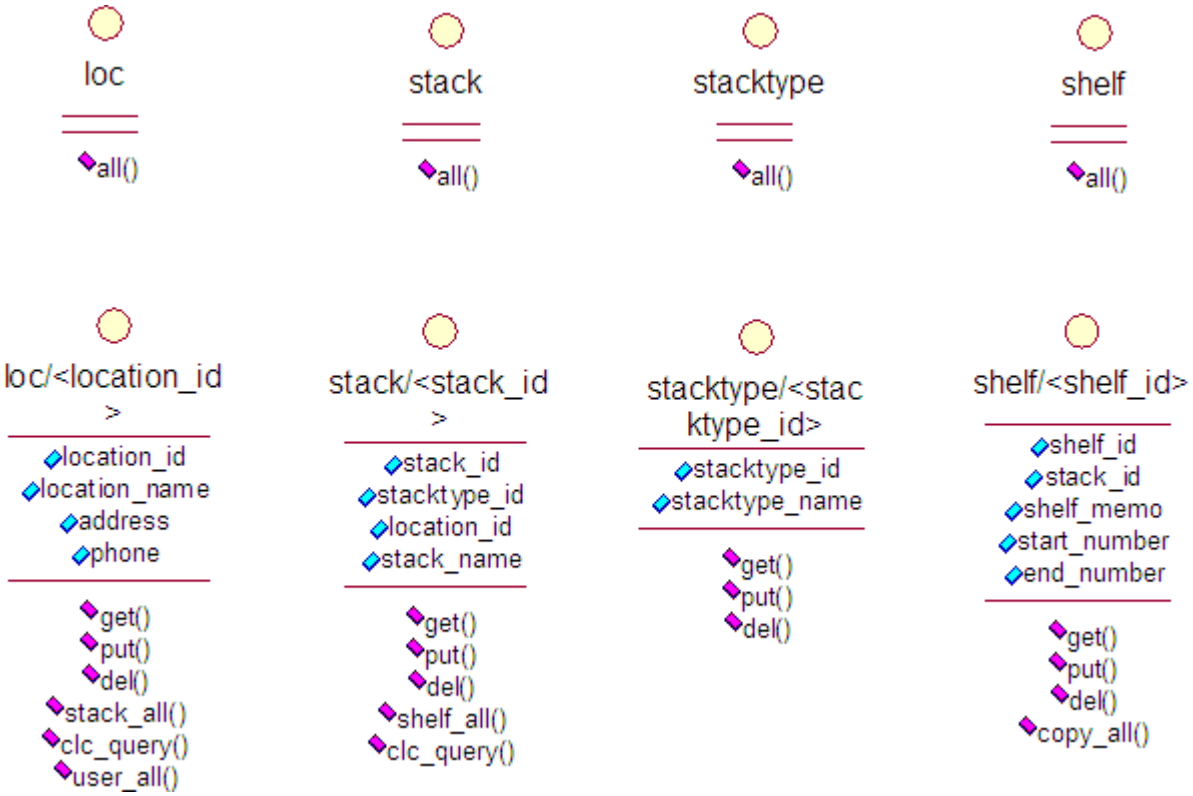
- list<User> grant_query(str grant_type, str grant_action)
  查询所有拥有某权限的用户对象，grant_type 与 grant_action 也可以是 list<str>类型。此操作需要权限(user,grant)。

URL：~/user/<工号> 或 ~/user/me 表示当前用户

表示一位馆员

- User get()
  获取该用户对象。
- None put(User user)
  保存该用户对象。此操作需要权限(user,write)。
- None del()
  删除该用户。此操作需要权限(user,write)。
- None kick()
  强制注销此用户的登录，用于管理员作废用户登录的服务票据。此操作需要权限(user,kick)。
- None set_password(str password)
  设置密码。此操作需要由登录用户本人进行，或者需要权限(user,password)。
- None grant(str grant_type, str grant_action)
  给用户授予权限；如果用户原有此权限，什么也不做。此操作需要权限(user,grant)。
- None revoke(str grant_type, str grant_action)
  剥夺用户的权限；如果用户原没有此权限，什么也不做。此操作需要权限(user,grant)。拥有(user.root)权限的超级管理员不能剥夺自己的(user,root)和(user.grant)权限，以免造成系统无法管理。
- list<tuple<str,str>> grant_all()
  查询用户拥有的所有权限。此操作需要本人进行，或者需要权限(user_grant)。

3.2.2 分馆、书库、书架部分



URL：~/loc

表示所有分馆

- list<Location> all()

获取所有分馆对象。

URL：~/loc/<分馆编号>

表示一个分馆

- Location get()
  获取该分馆对象。
- None put(Location location)
  保存该分馆对象。此操作需要权限(loc,write)。
- None del()
  删除该分馆，要求分馆中没有馆员、书库。此操作需要权限(loc,write)。
- list<Stack> stack_all()
  查询该分馆内所有书库对象。
- list<Shelf> clc_query(str call_number)
  查询该分馆内可以摆放这个索书号的所有书架。
- list<User> user_all()
  查询该分馆的所有馆员。

URL：~/stack

表示所有书库

- list<Stack> all()
  获取所有书库对象。

URL：~/stack/<书库编号>

- Stack get()
  获取该书库对象。
- None put(Stack stack)
  保存该书库对象。此操作需要权限(loc,write)。
- None del()
  删除该书库，要求书库中没有书架。此操作需要权限(loc,write)。
- list<Shelf> shelf_all()
  查询该书库内所有书架对象。
- list<Shelf> clc_query(str call_number)
  查询该书库内可以摆放这个索书号的所有书架。

URL：~/stacktype

表示所有书库分类

- list<StackType> all()
  获取所有书库分类。

URL：~/stacktype/<书库分类编号>

- StackType get()
  获取该书库分类对象。
- None put(StackType stacktype)
  保存该书库分类对象。此操作需要权限(loc,write)。
- None del()
  删除该书库分类，要求书库分类中没有书库。此操作需要权限(loc,write)。

URL：~/shelf

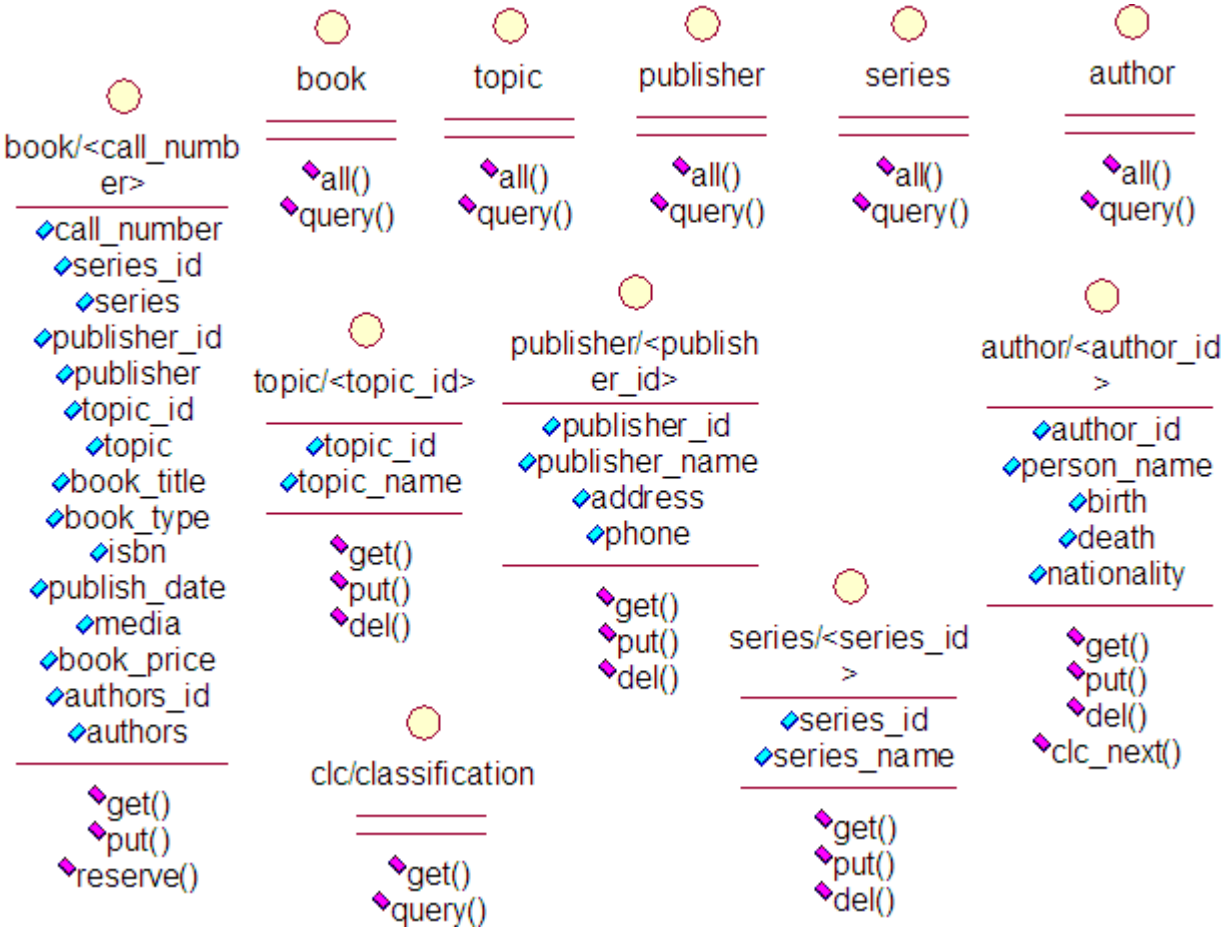
表示所有书架

- list<Shelf> all()
  获取所有书架对象。

URL：~/shelf/<书架编号>

- Shelf get()
  获取该书架对象。
- None put(Shelf shelf)
  保存该书架对象。此操作需要权限(loc,write)。
- None del()
  删除该书架，要求书架上没有复本。此操作需要权限(loc,write)。
- list<Copy> copy_all()
  查询该书架上所有复本对象。

3.2.3 图书、编目部分



URL：~/book
表示所有文献

- list<Book> all()
  获取所有文献对象。警告：数据量很大。
- list<Book> query(list<tuple<str,str,str>> filter, list<tuple<str,Boolean>> sort)
  查询文献。
  filter 参数是查询条件列表；
  格式举例一：[(‘author’,’=’,’陆松年’),(‘isbn’,’=’,’7121021390’)]，这个条件可以命中《操作系统教程》这本书；
  格式举例二：[(‘publisher’,’=’,’电子工业出版社’),(‘price’,’>’,’34.99’),(‘price’,’<’,’35.01’),(‘topic’,’=’,’操作系统—高等学校—教材’) ,(‘publish_date’,’>’,’20050101’)]，这个条件也可以命中《操作系统教程》这本书、以及其他若干文献；
  每个条件 tuple 中第一项为字段名，允许的值有 call_number 索书号、series 丛书名、publisher

出版社名称、topic 主题标目、title 题名、type 文献类型、isbn 图书 ISBN 或杂志 ISSN 编号、publish_date 出版日期、price 价格、author 著者姓名(不论第几著者)；

第二项为比较运算符，数字型参数支持=、!=、>、<、>=、<=六种，但是字符型参数支持=、!=、like(近似)三种；

第三项为比较参数，使用 like 运算符时可以用%作通配符；

filter 参数也可以是 str 类型，这种情况下在文献信息的任一处模糊查询这个关键字。

sort 参数是排序列表；

格式举例：[('author',False),('publish_date',True)]，表示先按第一著者姓名升序，再按出版日期降序。

URL：~/book/<索书号>

表示一种文献

series、publisher、topic 均为 id 对应的对象，authors 类型为 list<tuple<int,Author>>其 tuple 内为作者位次、著者对象，这样可以方便查询和修改。

authors_id 为 list<tuple<int,int>>类型，tuple 中第一项为作者位次、第二项为著者编号。

- Book get()
  获取该文献对象。
- None put(Book book)
  保存该文献对象，对于 series、publisher、topic、authors 的任何修改都不会保存、只保存 series_id、publisher_id、topic_id、authors_id。此操作需要权限(book,write)。
- None reserve(int reader_id, int location_id)
  预约馆藏。此操作需要权限(book,borrow) ，注意是指登录的馆员要有此权限。

URL：~/topic

表示所有主题标目

- list<Topic> all()
  获取所有主题标目对象。
- list<Topic> query(str topic_name_query)
  查询所有包含关键字的主题标目，此查询为模糊查询。

URL：~/topic/<主题编号>

表示一个主题标目

- Topic get()
  获取该主题标目对象。
- None put(Topic topic)
  保存该主题标目对象。此操作需要权限(book,write)。
- None del()
  删除该主题标目，要求主题标目没有关联到文献。此操作需要权限(book,write)。

URL：~/publisher

表示所有出版社

- list<Publisher> all()
  获取所有出版社对象。
- list<Publisher> query(str publisher_query)
  查询所有包含关键字的出版社（关键字可以出现于名称、地址、电话任一处），此查询为模糊查询。

URL：~/publisher/<出版社编号>

表示一个出版社

- Publisher get()
  获取该出版社对象。

- None put(Publisher publisher)
  保存该出版社对象。此操作需要权限(book,write)。
- None del()
  删除该出版社，要求出版社没有关联到文献。此操作需要权限(book,write)。

URL：~/series

表示所有丛书

名称相同的丛书被视为同一丛书

- list<Topic> all()
  获取所有丛书对象。
- list<Topic> query(str series_name_query)
  查询所有包含关键字的丛书，此查询为模糊查询。

URL：~/series/<丛书编号>

表示一套丛书

- Series get()
  获取该丛书对象。
- None put(Series series)
  保存该丛书对象。此操作需要权限(book,write)。
- None del()
  删除该丛书，要求丛书没有关联到文献。此操作需要权限(book,write)。

URL：~/author

表示所有著者

- list<Author> all()
  获取所有著者对象。
- list<Author> query(list<tuple<str,str,str>> filter, list<tuple<str, Boolean>> sort)
  查询著者。
  过滤条件的写法与文献查询类似；
  格式举例：[('name','=','陆松年'),('nationality','=','CN'),('birth','>','1960-01-01')]

URL：~/author/<著者编号>

表示一位著者

- Author get()
  获取该著者对象。
- None put(Shelf shelf)
  保存该著者对象。此操作需要权限(book,write)。
- None del()
  删除该著者，要求著者没有关联到文献。此操作需要权限(book,write)。
- str clc_next(str clc_prefix)
  给出该第一著者在某分类中下一个可用的索书号。
  通常索书号是根据“中图法分类号/著者流水号-种次号”格式排列的，例如 TP316/8-2，表示“操作系统”分类第 8 位作者的第 2 种文献。调用 clc_next('TP316')就会判断馆内是否已经有该著者的文献；如果该著者已有 TP316/8-2 就会返回 TP316/8-3；如果没有，就会分配一个新的著者流水号及种次号 1，例如返回 TP316/9-1。

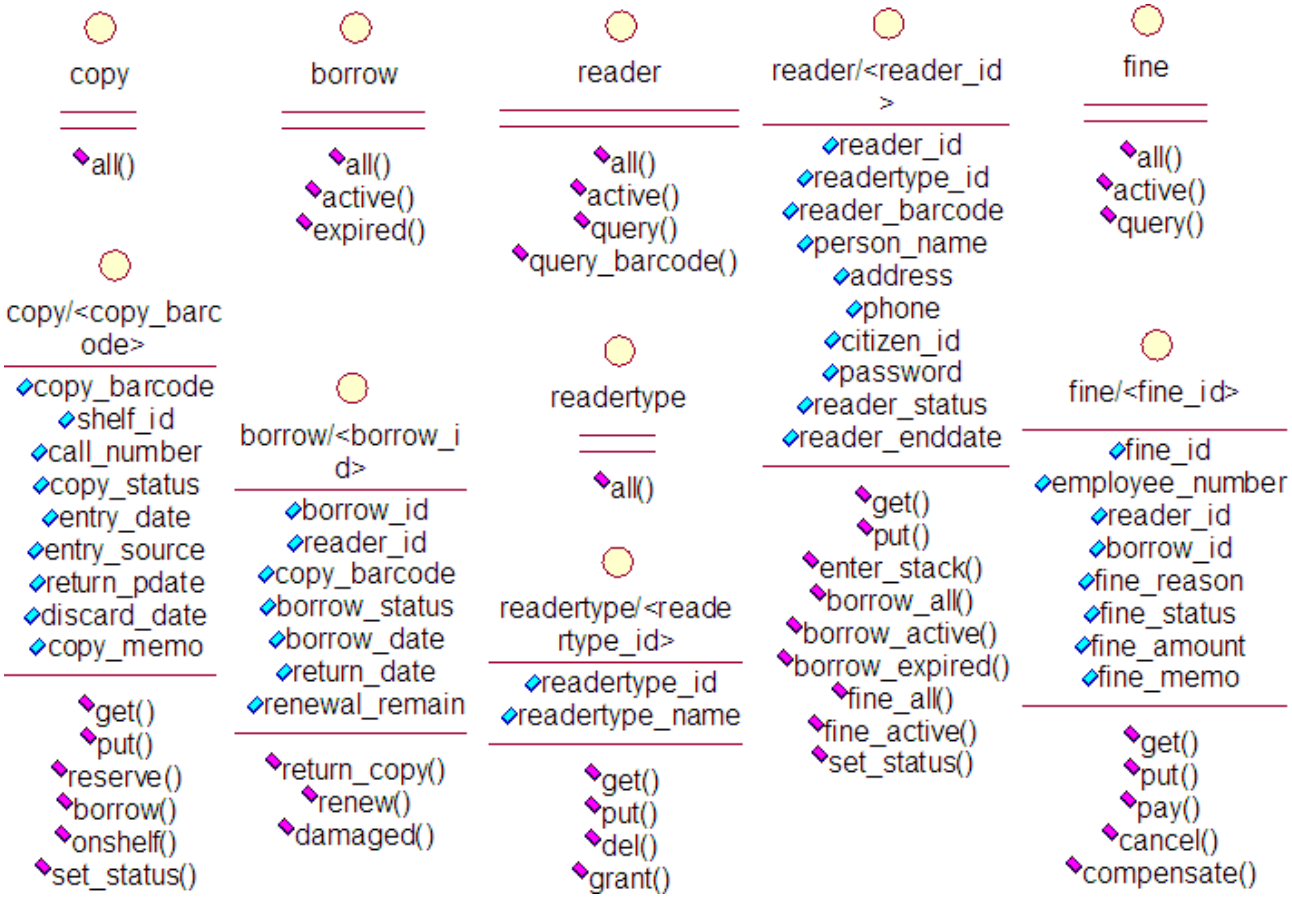
URL：~/clc/classification

中国图书馆分类法分类查询服务

- str get(str call_number)
  返回索书号对应的分类名，根据最详细层次
- str query(str query, bool like)

查询索书号，query 参数是类目名，like 参数取 True 时执行模糊查询。

3.2.4 复本、读者、借阅、罚款部分



URL：~/copy

表示所有复本

- list<Copy> all()
  获取所有复本对象。警告：数据量很大。

URL：~/copy/<馆藏条码>

表示一个复本

- Copy get()
  获取该复本对象。
- None put(Copy copy)
  保存该复本对象。此操作需要权限(book,write)。
- None reserve(int reader_id)
  预约复本。此操作需要权限(book,borrow)，注意是指登录的馆员要有此权限。
- None borrow(int reader_id)
  外借复本。此操作需要权限(book,borrow)。
- None onshelf()
  上架。此操作需要权限(book,onshelf)。
- None set_status(int copy_status)
  设置复本状态。此操作需要权限(book,write)。

URL：~/borrow

表示所有借阅记录

- list<Borrow> all()
  获取所有借阅记录。警告：数据量很大。此操作需要权限(book,borrow)。

- list<Borrow> active()
  查询当前借出的借阅记录。此操作需要权限(book,borrow)。
- list<Borrow> expire()
  查询当前逾期的借阅记录。此操作需要权限(book,borrow)。

URL：~/borrow/<借阅记录编号>

表示一条借阅记录，一位读者借阅一个复本一次就登记一条借阅记录

- Borrow get()
  获取该读者对象。此操作需要权限(book,borrow)。
- int return_copy()
  归还复本。正常情况返回 0；如果逾期，自动登记罚款记录并返回罚款记录编号。此操作需要权限(book,borrow)。
- None renew()
  续借复本。此操作需要权限(book,borrow)。
- int damaged(int reason, float amount, str memo)
  损坏或遗失，登记罚款记录，返回罚款记录编号。此操作需要权限(book,borrow)。

URL：~/reader

表示所有读者

- list<Reader> all()
  获取所有读者对象。警告：数据量很大。此操作需要权限(reader,read)。
- list<Reader> active()
  获取未过期的读者对象。此操作需要权限(reader,read)。
- list<Reader> query(list<tuple<str,str,str>> filter, list<tuple<str,Boolean>> sort)
  根据条件查询读者。此操作需要权限(reader,read)。
  条件格式举例：[('citizen','like','310112%'),('address','=','上海交通大学闵行校区 D11-112')]；
  支持的查询字段名有 type,barcode,name,address,phone,citizen,status,enddate
- Reader query_barcode(str barcode)
  查询条码对应的读者。

URL：~/reader/<读者编号>

表示一名读者

- Reader get()
  获取该读者对象。此操作需要权限(reader,read)。
- None put(Reader reader)
  保存该读者对象。此操作需要权限(reader,write)。
- None enter_stack(int stack_id)
  进入书库。无返回值，但是读者无权进入书库时会抛出 NoPermissionError 异常。此操作无需登录。
- list<Borrow> borrow_all()
  查询该读者所有借阅记录。此操作需要权限(reader,read)。
- list<Borrow> borrow_active()
  查询该读者未归还借阅记录。此操作需要权限(reader,read)。
- list<Borrow> borrow_expired()
  查询该读者逾期借阅记录。此操作需要权限(reader,read)。
- list<Fine> fine_all()
  查询该读者所有罚款记录。此操作需要权限(reader,fine)。
- list<Fine> fine_active()
  查询该读者未缴罚款记录。此操作需要权限(reader,fine)。

- None set_status(int reader_status)
  设置读者状态。此操作需要权限(reader,write)。

URL：~/readertype

表示所有读者类型

- list<ReaderType> all()
  获取所有读者类型对象。

URL：~/readertype/<读者类型编号>

表示一种读者类型

- ReaderType get()
  获取该读者类型。
- None put(ReaderType readertype)
  保存该读者类型。此操作需要权限(reader,type)。
- None del()
  删除该读者类型，要求没有该类型的读者，自动删除授权项。此操作需要权限(reader,type)。
- None grant(int stacktype_id, Boolean can_enter, int borrow_limit, int borrow_days, int borrow_renews, int reserve_limit, int reserve_period)
  设置这类读者对某类书库的访问权限。此操作需要权限(reader,type)。

URL：~/fine

表示所有罚款记录

- list<Fine> all()
  获取所有罚款记录。此操作需要权限(reader,fine)。警告：数据量很大。
- list<Fine> active()
  查询未缴罚款记录。此操作需要权限(reader,fine)。
- list<Fine> query(list<tuple<str,str,str>> filter, list<tuple<str,Boolean>> sort)
  根据条件查询罚款记录。此操作需要权限(reader,fine)。
  条件格式举例：[('amount','>=','20.00')]；
  支持的查询字段名有 employee,reader,borrow,reason,status,amount,memo

URL：~/fine/<罚款记录编号>

表示一条罚款记录

- Fine get()
  获取该罚款记录。此操作需要权限(reader,fine)。
- None put(Fine fine)
  保存该罚款记录。此操作需要权限(reader,fine)。
- None pay()
  缴纳罚款。自动将收款人添加到原备注末尾。此操作需要权限(reader,fine)。
- None cancel(str memo)
  撤销罚款。自动将撤销人、新备注添加到原备注末尾。此操作需要权限(reader,fine)。
- None compensate()
  赔偿复本。自动将接受人添加到原备注末尾。此操作需要权限(reader,fine)。

3.3 内部接口

内部接口与外部接口基本一致，在此不作规定。

4 运行设计

4.1 运行模块组合

应用服务器本身就是一个模块。

4.2 运行控制

每种运行均由 JSON-RPC 请求触发，在有限的时间内返回，函数返回后不再继续执行。

4.3 运行时间

各函数将尽可能快的返回，以提高系统效率。

5 系统数据结构设计

5.1 逻辑结构设计要点

逻辑结构见外部接口部分图片。

5.2 物理结构设计要点

数据项存储于关系数据库，见数据库设计说明书。

注：演示系统采用 Django 制作，使用的物理结构不同于数据库设计说明书定义的结构；Django 可以在 SQLite 或其他数据库中存储模型定义的 Python 对象。

5.3 数据结构与程序的关系

面向对象设计，程序与数据结构定义在一起。

6 系统出错处理设计

6.1 出错信息

大部分情况下，应用服务器通过 JSON-RPC 响应的 error 对象发送错误信息，交给工作台处理。网络连接、服务器异常等严重错误，以 HTTP 状态码表示。

6.2 补救措施

只负责返回错误信息，没有补救措施。

6.3 系统维护设计

使用 Python 语言编写应用服务器，Python 本身的特点使应用服务器容易调试，只需在适当的地方插入 logging 即可检测系统状态。

图书管理信息系统详细设计说明书

版本历史

2008-05-05 文档创建

1 引言

1.1 编写目的

本文档详细描述了图书管理信息系统部分功能的设计细节。

1.2 背景

- 软件系统名称：图书管理信息系统
- 任务提出者：刘海涛
- 开发者：F0503602 石君霄 5050369043
- 用户：传统图书馆

1.3 定义

1.4 参考资料

a) Python 2.5 Documentation，http://docs.python.org/

b) Django Documentation 0.96，http://www.djangoproject.com/documentation/0.96/

2 程序系统的结构

图书管理信息系统根据企业应用的三层结构设计，工作台等终端负责显示界面并与应用服务器交互，应用服务器处理事务逻辑并操作数据库。

3 终端与应用服务器的通信

3.1 程序描述

终端与应用服务器采用 JSON-RPC 协议进行通信。调用是单向的，只能由终端调用应用服务器，应用服务器不能调用终端。JSON-RPC 报文使用 HTTP 协议承载。

3.2 功能

从终端看——输入[对象 URL]、[函数名]、[参数]，输出[函数执行结果]或[错误信息]。

3.3 性能

主要取决于网络速度。从终端看，这个程序在调用完毕前不会返回，因此当网络连接很慢时性能会显著降低。

3.4 输入项

- 对象 URI：字符串
- 函数名：字符串
- 参数：0 个~多个参数，一切 JSON 可编码的类型

3.5 输出项

- 函数执行结果：一切 JSON 可编码的类型
- 或 抛出异常：yoursunny.P2008.Library.API.Error 类

3.6 算法

使用 JSON-RPC 协议。

3.7 流程逻辑

终端程序调用 yoursunny.P2008.Library.API.JSONRPC.Call 函数，传入 URI、函数名、参数。生成随机字符串作为 id，构造请求数据，交由 Newtonsoft.Json.JavaScriptConvert.SerializeObject 函数编码成 JSON 数据，发送 HTTP POST 请求，并在 cookie 中附带用于验证身份的服务票据。

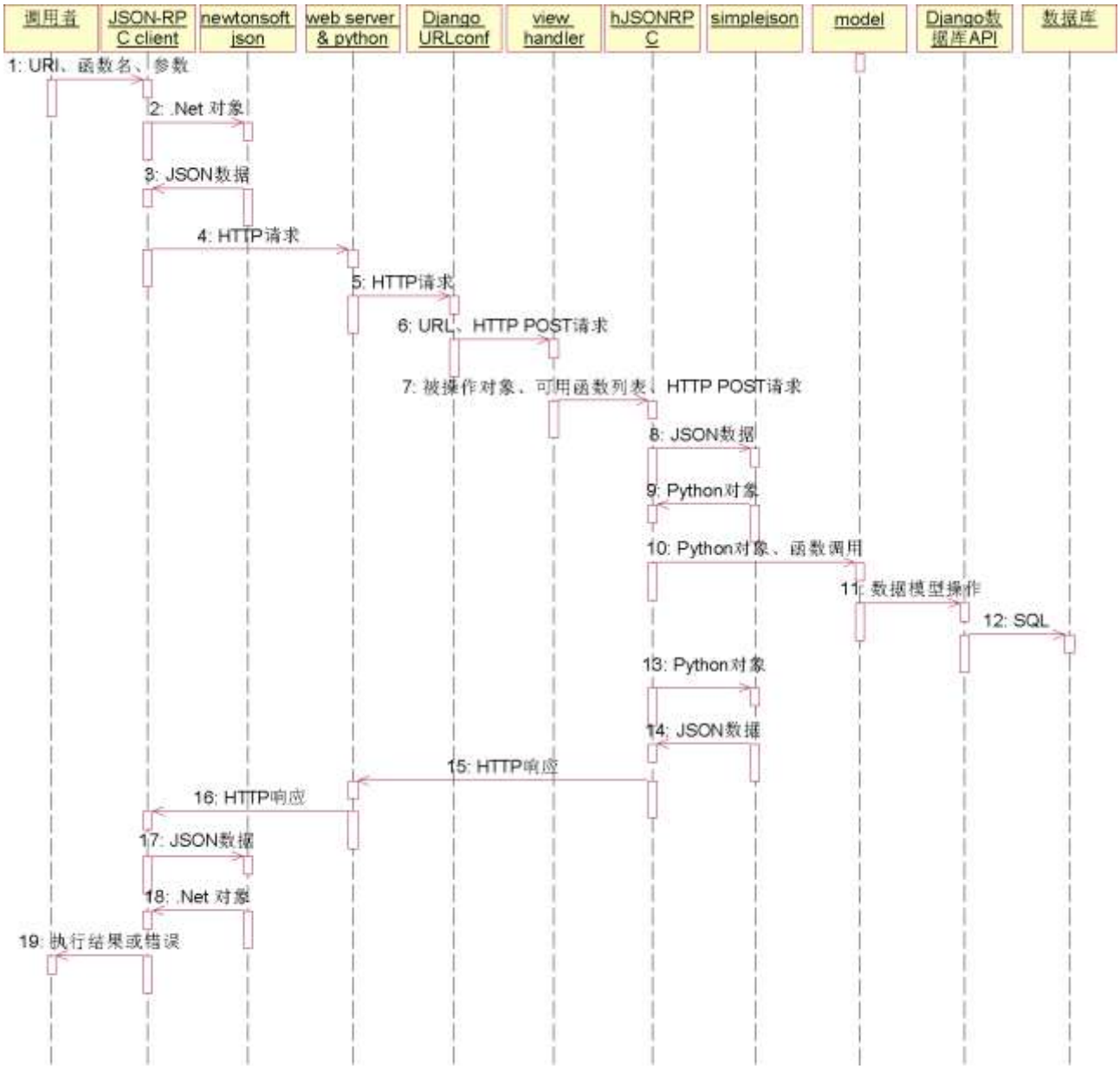
Web 服务器收到后由 Django 处理，Django 根据 urls.py 的定义调用应用服务器的 view（例如/api/loc/1/会调用 hLocation 函数）。尝试创建相应的 model 对象（例如 Location(1)对象），准备好允许调用的函数列表和需要的权限表，调用 hJSONRPC 函数。

用 simplejson.loads 函数对 JSON 数据进行解码，检查请求是否有效、当前用户是否有执行权限，如

果有，调用 model 对象的相应函数。model 对象可以根据需要调用 Django 数据库 API、通过它操作数据库。

hJSONRPC 接收 model 对象的函数的返回值，构造 JSON-RPC 响应，交由 simplejson.dumps 函数编码为 JSON 格式，返回给客户端。

JSONRPC.Call 函数收到 HTTP 响应后，用 Newtonsoft.Json.JavaScriptConvert.DeserializeObject 函数解码。如果发现 error 成员不为 null，抛出异常；否则返回 result 成员。



3.8 接口

```
public static object Call(string url, string method, params object[] p)
```

3.9 存储分配

3.10 注释设计

3.11 限制条件

这个函数执行前，必须将 JSONRPC.baseUrl 设置为应用服务器根目录 URI、或者在 AppSettings[“ServerUrl”]中设置好应用服务器根目录 URI。

必须保证能从网络到达应用服务器，且应用服务器软件正常工作。

3.12 测试计划

3.13 尚未解决的问题

4 应用服务器的 hJSONRPC 函数

4.1 程序描述

使用 Django 开发应用服务器时，每个 view 都是类似的流程——JSON 解码、检查登录、检查权限、获取模型对象、调用模型对象的相应函数、JSON 编码、返回结果。通过 hJSONRPC 函数，以统一的流程完成上述过程。

4.2 功能

hJSONRPC 函数负责解析 HTTP 承载的 JSON-RPC 请求，调用对象上的相应函数，然后返回结果或错误给客户端。

4.3 性能

4.4 输入项

- request：HTTP 请求
- o：模型对象，可以为 None
- cls：模型类，可以为 None
- methods：允许执行的函数、需要的权限

  这是一个 dict 对象，每个项目的 key 是 JSON-RPC 调用的函数名称，值为一个 tuple；

  tuple 有两个项目：

  第一个是对象上的实际函数名称，可以用前缀“-”表示允许自动创建新的模型对象；

  第二个是执行所需的权限，LoginService.always 常量表示允许未登录执行，True 表示允许任何登录用户执行，False 表示不允许执行，字符串“对象类型,操作类型”表示检查这个权限

一个举例的输入组合是：

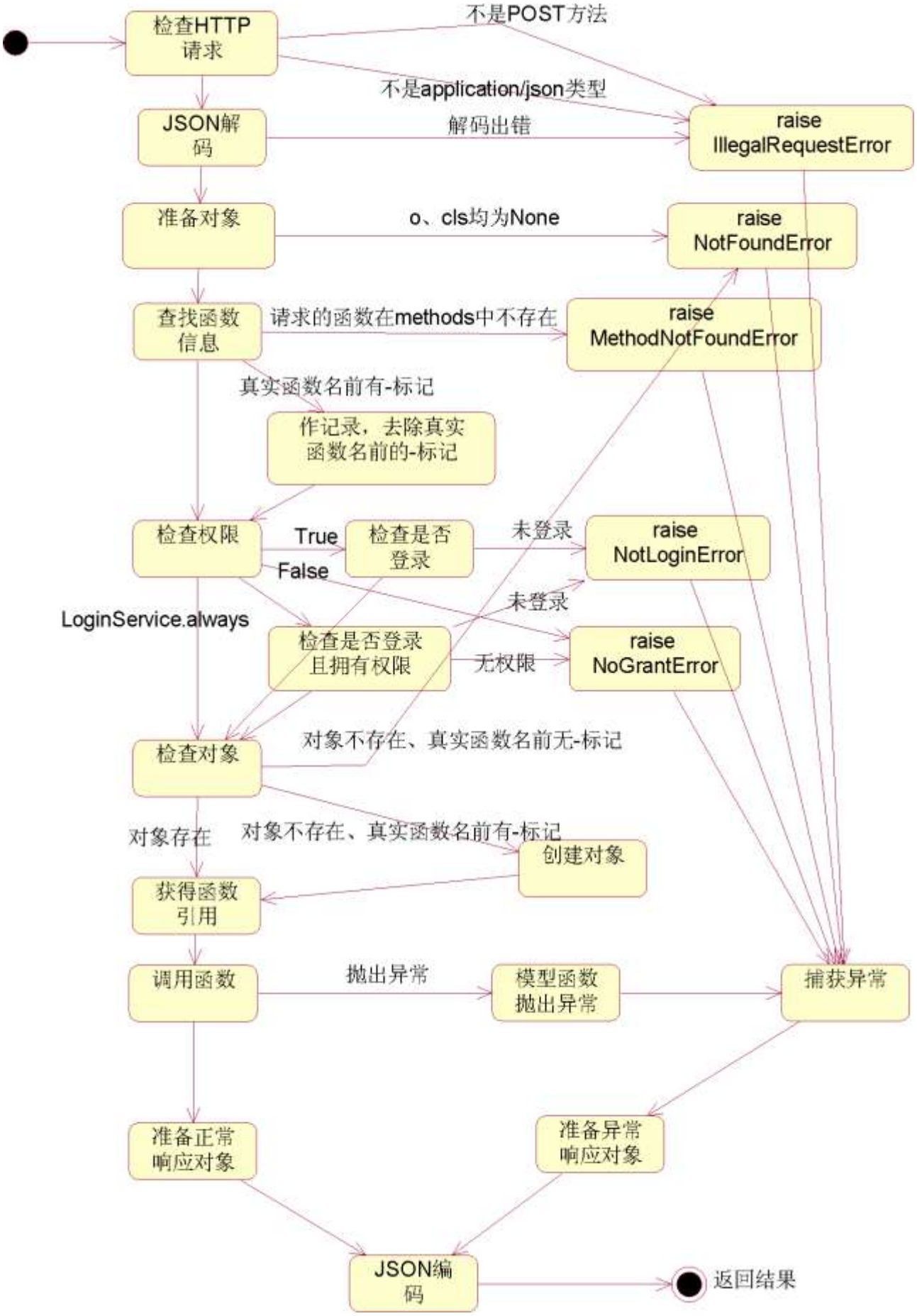
```
def hLocation(request,location_id):
    try:o=Location.objects.get(pk=int(location_id))
    except:o=None
    methods={
        'get':('get',LoginService.always),
        'put':('-put','loc,write'),
        'del':('del_','loc,write'),
        'stack_all':('stack_all',LoginService.always),
        'clc_query':('clc_query',LoginService.always),
        'user_all':('user_all',LoginService.always),
    }
    return JSONRPC(request,o,Location,methods)
```

4.5 输出项

- 一个 HttpResponse 对象

4.6 算法

4.7 流程逻辑


检查HTTP
请求
不是POST方法
不是application/json类型
JSON解
码
解码出错
raise
IllegalRequestError
准备对象
o、cls均为None
raise
NotFoundError
查找函数
信息
请求的函数在methods中不存在
raise
MethodNotFoundError
真实函数名前有-标记
作记录，去除真实
函数名前的-标记
检查权限
True
检查是否
登录
未登录
raise
NotLoginError
False
未登录
LoginService.always
检查是否登录
且拥有权限
无权限
raise
NoGrantError
检查对象
对象不存在、真实函数名前无-标记
对象存在
对象不存在、真实函数名前有-标记
创建对象
获得函数
引用
调用函数
抛出异常
模型函数
抛出异常
捕获异常
准备正常
响应对象
准备异常
响应对象
JSON编
码
返回结果

4.8 接口

```
def JSONRPC(request,o,cls,methods)
```

4.9 存储分配

4.10 注释设计

4.11 限制条件

该函数曾经只允许输入纯 ASCII 编码的 JSON 数据，而不允许 Unicode，因而不能使用汉字；经查是 simplejson 未支持 Unicode 多字节编码，经过修改 simplejson 模块，已经解决这个问题。

4.12 测试计划

4.13 尚未解决的问题

5 查询可以摆放这个索书号的书架

5.1 程序描述

图书馆摆放文献通常都有一定的组织规律，将文献根据索书号排列，安排在不同的书架上。

5.2 功能

Location.clc_query、Stack.clc_query 函数，根据索书号或中图法分类号，从分馆或书库内的所有书架中，查询可以摆放这本书的书架列表。

5.3 性能

5.4 输入项

- call_number：索书号，形式为 TP393.1/2.4；可以是索书号的任意前缀，如 TP、TP3、TP393、TP393.1、TP393.1/2

5.5 输出项

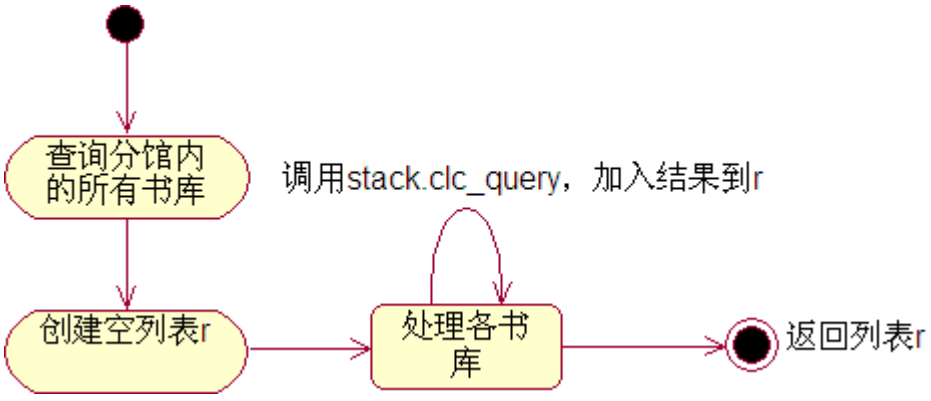
- list<Shelf>类型

5.6 算法

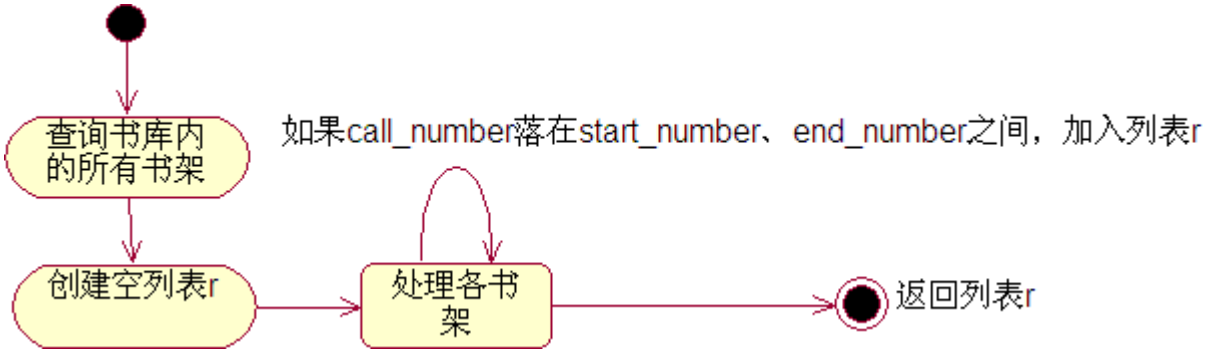
从数据模型中取出这个分馆或书库内的所有书架，遍历这个列表，查询每一个书架的摆放范围是否符合参数索书号。

5.7 流程逻辑

5.7.1 分馆的 clc_query 函数



5.7.2 书库的 clc_query 函数



5.8 接口

```
def clc_query(self,call_number)
```

5.9 存储分配

5.10 注释设计

5.11 限制条件

要求图书馆必须根据索书号排列文献，而不是采用其他排列方法。

5.12 测试计划

5.13 尚未解决的问题

6 文献查询

6.1 程序描述

BookManager.query 函数负责根据指定条件和排序方法，查询文献。

6.2 功能

分析输入的条件、排序方法列表，转换成数据模型查询参数，查询文献。

6.3 性能

文献数量巨大，必须进行必要的过滤，不允许遍历所有文献。

6.4 输入项

- filter：查询条件列表；
  格式举例一：[(‘author’,’=’,’陆松年’),(‘isbn’,’=’,’7121021390’)]，这个条件可以命中《操作系统教程》这本书；
  格式举例二：[(‘publisher’,’=’,’电子工业出版社’),(‘price’,’>’,’34.99’),(‘price’,’<‘,’35.01’),(‘topic’,’=‘,’操作系统—高等学校—教材’) ,(‘publish_date’,’>‘,’20050101’)]，这个条件也可以命中《操作系统教程》这本书、以及其他若干文献；
  每个条件 tuple 中第一项为字段名，允许的值有 call_number 索书号、series 丛书名、publisher 出版社名称、topic 主题标目、title 题名、type 文献类型、isbn 图书 ISBN 或杂志 ISSN 编号、publish_date 出版日期、price 价格、series 丛书名称、author 著者姓名(不论第几著者)；
  第二项为比较运算符，数字型参数支持=、!=、>、<、>=、<=六种，但是字符型参数支持=、!=、like(近似)三种；
  第三项为比较参数，使用 like 运算符时可以用%作通配符；
  filter 参数也可以是 str 类型，这种情况下在文献信息的任一处模糊查询这个关键字。
- sort：排序列表；
  格式举例：[(‘author’,False),(‘publish_date’,True)]，表示先按第一著者姓名升序，再按出版日期

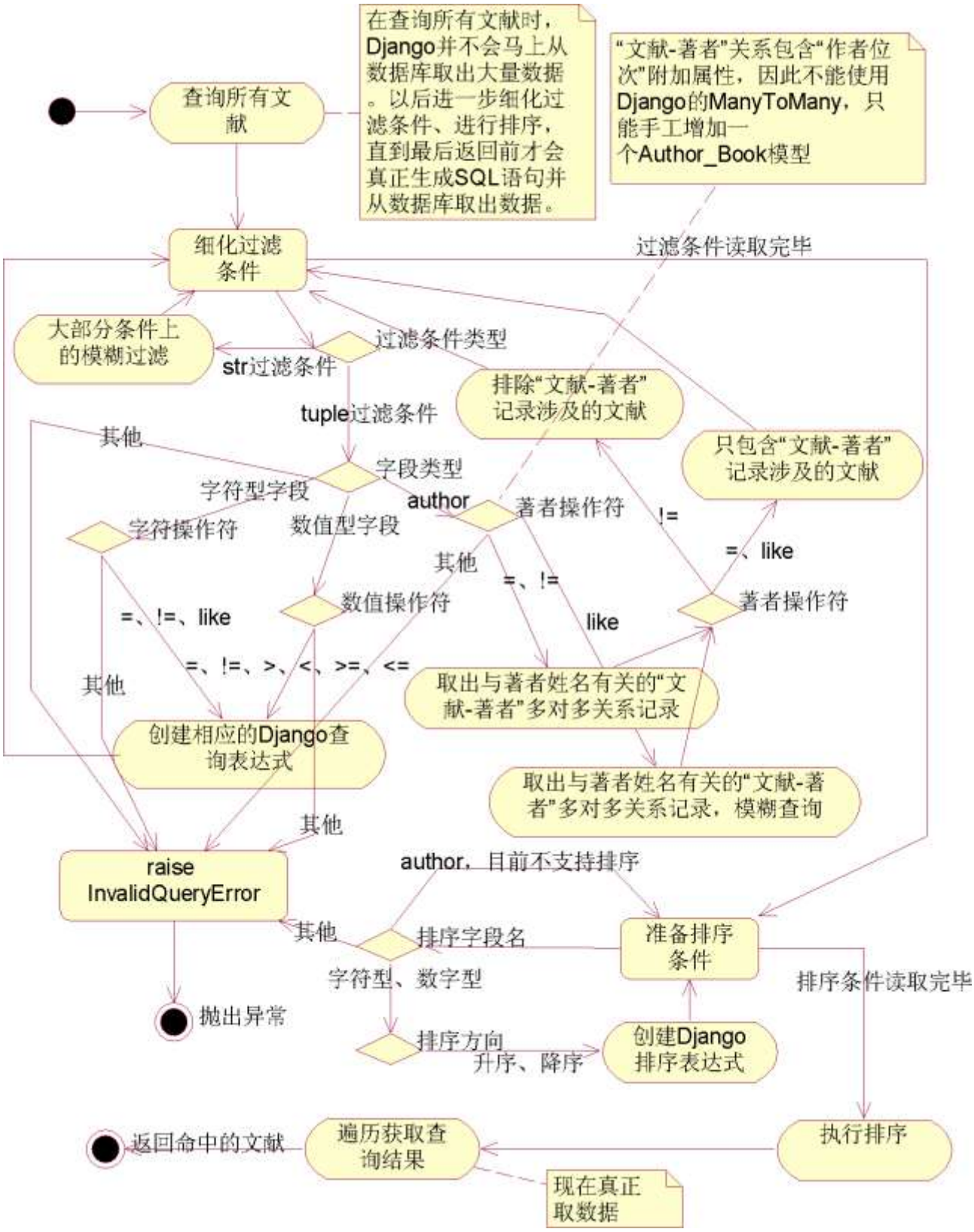
降序。

6.5 输出项

- list<Book>类型，命中的文献列表

6.6 算法

6.7 流程逻辑



6.8 接口

```
def query(self,filter,sort)
```

6.9 存储分配

6.10 注释设计

6.11 限制条件

6.12 测试计划

6.13 尚未解决的问题

由于 Django 多对多关系不支持附加属性，目前无法根据 author 排序。

7 预约馆藏、预约复本

7.1 程序描述

读者可以根据需要、在读者证权限允许范围内预约一定数量的文献。预约分两种：预约馆藏，即预约某文献在分馆内的任一复本；预约复本，预约特定的复本。

7.2 功能

预约馆藏 Book.reserve 函数，记录读者需要预约馆藏。

预约复本 Copy.reserve 函数，检查读者权限，如有权为该读者预约复本。

预约馆藏转为预约复本 Book.reserve_copy 函数，检查该文献的预约馆藏记录能否转成预约复本记录，当预约馆藏完成后、有复本归还时执行这个函数。

7.3 性能

7.4 输入项

7.4.1 预约馆藏

- reader_id：读者编号
- location_id：分馆编号

7.4.2 预约复本

- reader_id：读者编号

7.4.3 预约馆藏转为预约复本

- 无需参数

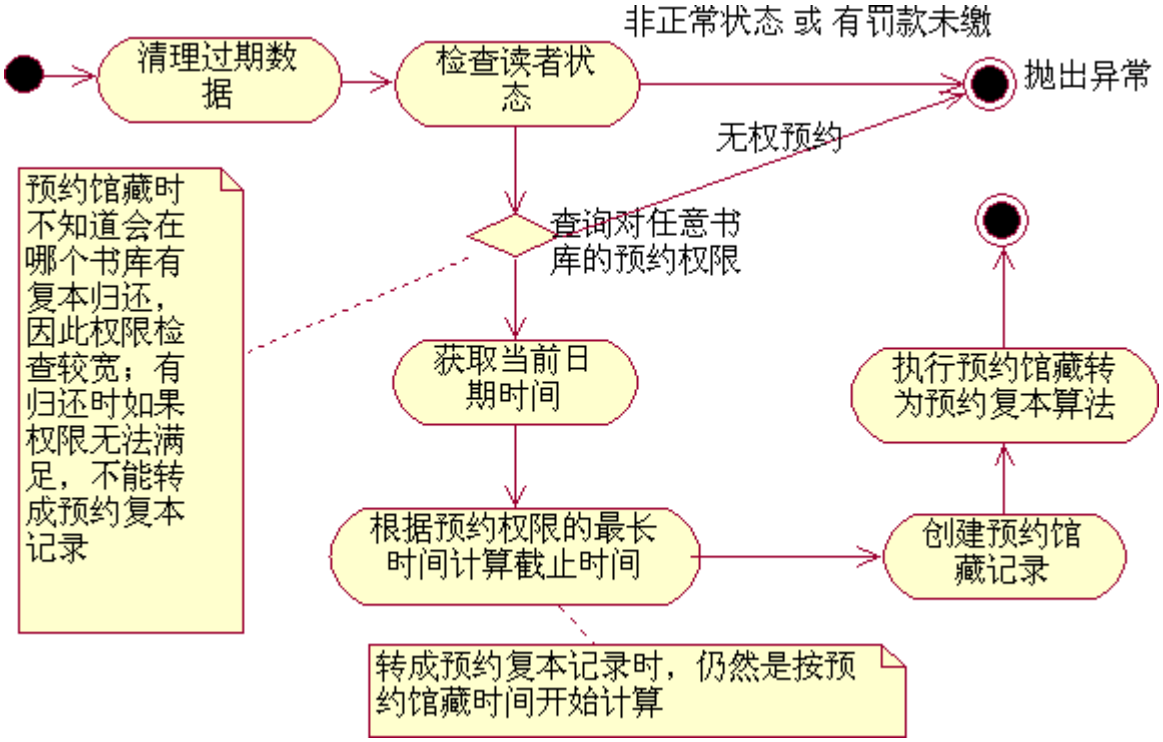
7.5 输出项

- 无返回值

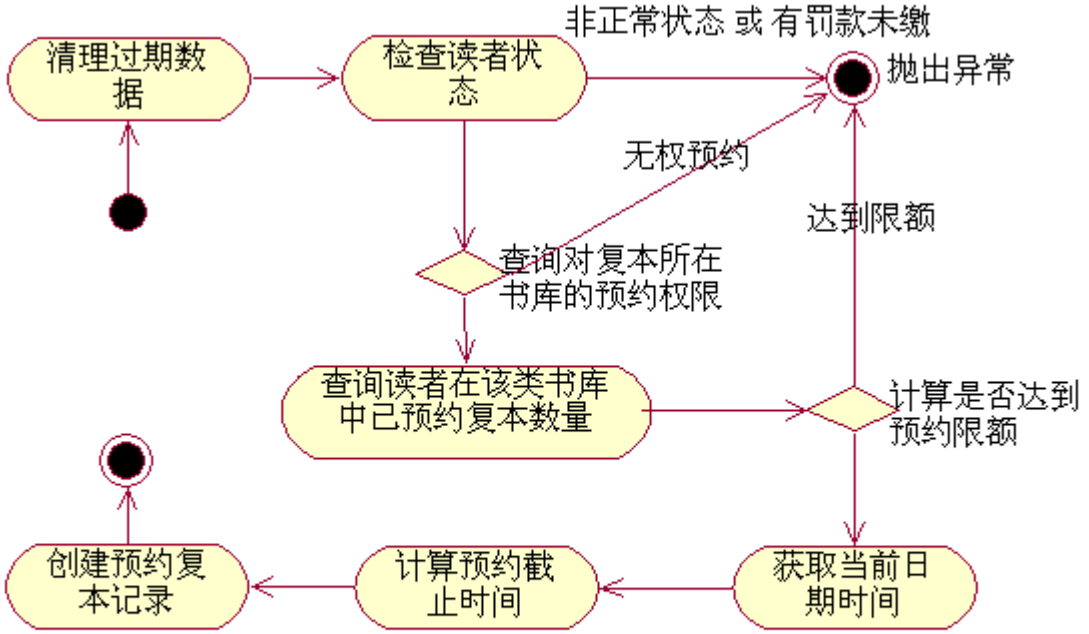
7.6 算法

7.7 流程逻辑

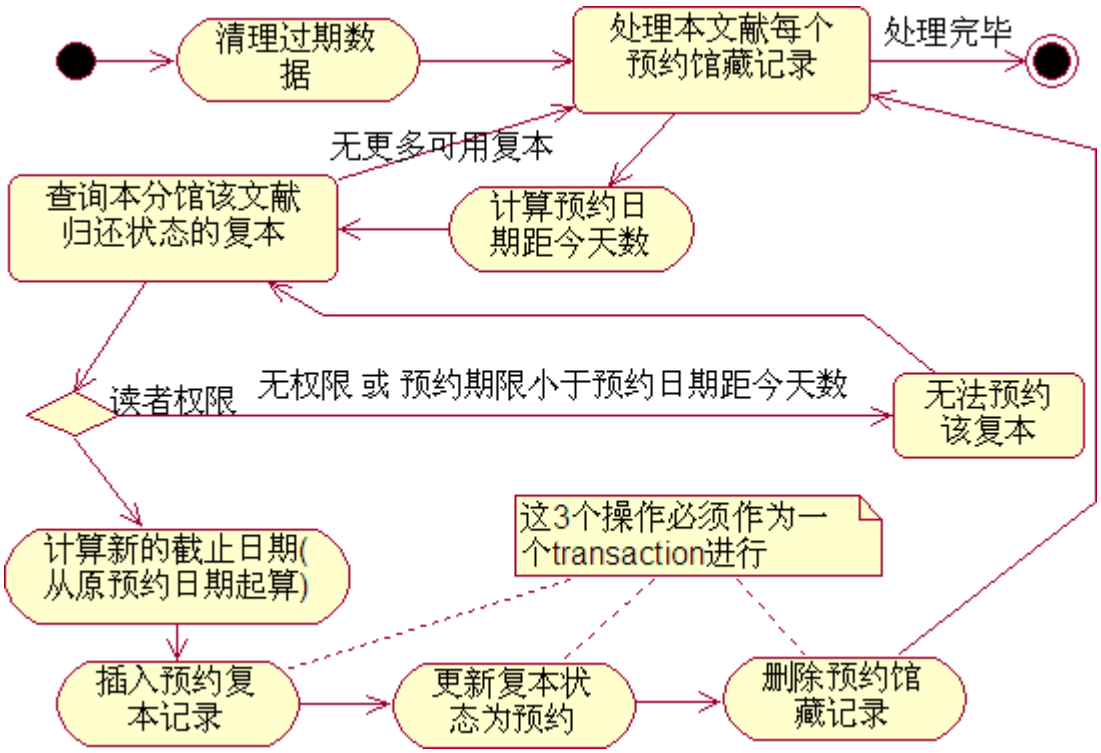
7.7.1 预约馆藏



7.7.2 预约复本



7.7.3 预约馆藏转为预约复本



7.8 接口

预约馆藏：`def reserve(self,reader_id,location_id)`

预约复本：`def reserve(self,reader_id)`

预约馆藏转为预约复本：`def reserve_copy(self)`

7.9 存储分配

7.10 注释设计

7.11 限制条件

7.12 测试计划

7.13 尚未解决的问题

8 复本的生命周期

8.1 程序描述

本节描述了一个复本从入库到报废的整个生命周期，以及程序对它的操作。

8.2 功能

复本采购入库、编目后，可以被安排到阅览室或可外借的书库。然后它的状态（Copy.copy_status）就会根据下面的规律变化。

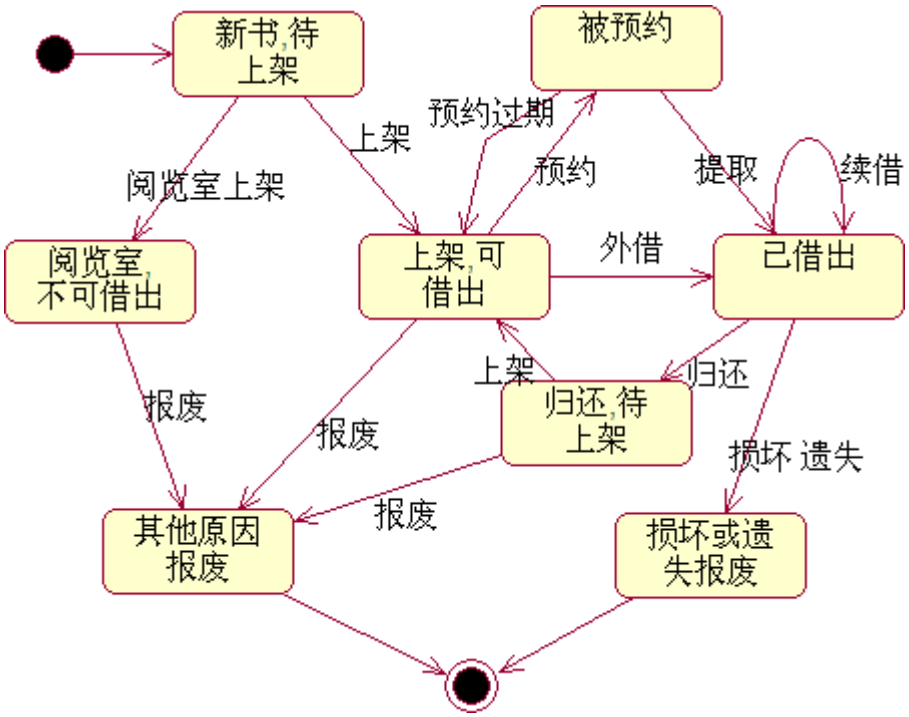
8.3 性能

8.4 输入项

8.5 输出项

8.6 算法

8.7 流程逻辑



8.8 接口

复本状态不是一个程序，而是一组常量定义：

```
copy_status_choices=(
    (0,u'未知'),
    (1,u'归还,待上架'),
    (2,u'上架,可借出'),
    (3,u'已借出'),
    (4,u'阅览室,不可借出'),
    (5,u'新书,待上架'),
    (6,u'损坏或遗失报废'),
    (7,u'其他原因报废'),
    (8,u'被预约'),
)
```

对复本状态进行操作的主要程序有：

- 直接设置复本状态为新的数值：Copy 类`def set_status(self,copy_status)`
- 上架：Copy 类`def onshelf(self)`
- 预约复本：Copy 类`def reserve(self,reader_id)`
- 预约馆藏转为预约复本：Book 类`def reserve_copy(self)`
- 预约过期：CopyManager 类`def clearOldReserves(self)`
- 外借、提取预约：Copy 类`def borrow(self,reader_id)`
- 续借：Borrow 类`def renew(self)`
- 归还：Borrow 类`def return_copy(self)`
- 损坏、遗失：Borrow 类`def damaged(self,reason,amount,memo)`

8.9 存储分配

8.10 注释设计

8.11 限制条件

8.12 测试计划

8.13 尚未解决的问题

在演示系统中，部分函数未实现

9

9.1 程序描述

9.2 功能

9.3 性能

9.4 输入项

9.5 输出项

9.6 算法

9.7 流程逻辑

9.8 接口

9.9 存储分配

9.10 注释设计

9.11 限制条件

9.12 测试计划

9.13 尚未解决的问题

10 应用服务器的 hJSONRPC 函数

10.1 程序描述

10.2 功能

10.3 性能

10.4 输入项

10.5 输出项

10.6 算法

10.7 流程逻辑

10.8 接口

10.9 存储分配

10.10 注释设计

10.11 限制条件

10.12 测试计划

10.13 尚未解决的问题

图书管理信息系统 演示系统开发结果

版本历史

2008-05-07 文档创建

1 演示系统平台

1.1 运行平台

- 数据库：SQLite 3，开源的小型关系型数据库系统
- 应用服务器：Django 0.96，Django Development Server
- 工作台：.Net Framework 3.5，Windows Presentation Foundation

1.2 系统要求

- 服务器：Ubuntu Server 8.04 hardy 或其他，安装 Python 2.5、Django 0.96
- 工作台：Windows Vista Home Basic 或其他，安装 Microsoft .Net Framework 3.5（不支持 Mono-Project）

1.3 启动方法说明

1.3.1 服务器的启动

安装好支持环境后，将 server.tar 上传到服务器，然后执行下列命令：

$ tar –xf server.tar

$ cd server

$ cd library

$ vi settings.py

将其中 DATABASE_NAME 改为数据库的绝对路径；数据库已经打包在 library/data.sqlite 文件中

$ python manage.py runserver 0.0.0.0:20809

1.3.2 客户端的启动

安装好支持环境后，将 win32bin.tar 解开（你可以使用 7-Zip 或 WinRAR 解开它）。然后修改每一个.config 文件，将 ServerUrl 改成服务器的网络地址，例如 http://192.168.2.1:20809/api/。修改完毕后，双击 LibraryLogin.exe 执行程序。

1.3.3 初始数据配置

- 工号 2222，密码 111111，拥有几乎所有权限
- 馆藏条码有 0000000001、0000000002、0000000003、0000000004、0000000005、0000000006
- 读者条码有 9999999999，该读者可以借 2 本书

可以修改 data.sqlite 文件以编辑这些功能

2 演示系统的功能

2.1 应用服务器的功能

2.1.1 完整实现的功能

- JSON 编解码，包括中文字符集支持；simplejson 修改而得
- JSON-RPC 调用接口，即 library.views.hJSONRPC 等函数
- 登录认证服务，修改密码，权限检查，即 library.views.LoginService 类
- 所有数据模型的字段、关系，即 library.models.Book 等类
- Django views，将 HTTP 请求转发给 model，即 library.views.hLocation 等函数
- 参数复制功能，即 library.models.paramCopy 等函数
- 馆员用户、权限管理
- 分馆、书库、书架管理
- 外借复本、预约复本、预约馆藏、预约馆藏转预约复本

2.1.2 部分实现的功能

- 文献查询
- 文献编目

- 读者管理

2.1.3 未实现的功能

- 归还复本、续借复本
- 罚款、黑名单
- 催还

2.2 客户端 API 的功能

客户端 API 位于 yoursunny.P2008.Library.API 命名空间，提供了相对于应用服务器接口的强类型.Net 类映射

2.2.1 完整实现的功能

- JSON 编解码，包括中文字符集支持；Newtonsoft.Json 修改而得
- JSON-RPC 调用接口，带服务票据 ticket 功能
- 文献
- 复本
- 分馆
- 登录认证
- 书架
- 书库
- 书库类型
- 馆员用户

2.2.2 不完整的功能

- 著者
- 借阅记录
- 罚款
- 出版社
- 读者
- 读者类型
- 丛书
- 主题标目

2.3 工作台的功能

2.3.1 完整实现的功能

- 登录、注销
- 修改密码
- 分馆添加、删除、修改
- 借阅复本
- 借还书、读者服务工作台的 RibbonBar 界面（2007 Office System 风格）

2.3.2 有界面、不能使用的功能

- 文献采编：支持 ISBN 号正确性验证
- 馆员管理

2.3.3 未实现的功能

- 权限管理
- 书库、书架管理
- 归还、预约
- 读者管理
- 罚款
- 催还
- 统计报表

2.4 其他

原计划的公共查询台、门禁控制器（ARM）、上架记录器（Andriod）未实现

3 技术说明

3.1 开发环境

- 计算机：DELL INSPIRON 600m，32 位 CPU，512MB 内存
- 操作系统：Windows Server 2003 标准版
- 中间件：Python 2.5，Microsoft .Net Framework 3.5
- 集成开发环境：PyDev，Visual C# 2008 Express Edition

3.2 采用的主要技术

3.2.1 Django

一套 Python 库，主要用于 Web 开发。

Django 根据 MVC 设计模式构造。它最强大的功能是根据定义的模型，自动生成数据库、自动操作数据库；这使开发者的精力集中在逻辑数据模型的设计上，而不必关心数据库的物理实现（表结构、SQL 语句都由 Django 自动生成）。

我把它用于应用服务器的开发，就是因为它强大的数据模型功能。

3.2.2 Windows Presentation Foundation

Windows Vista 中引入的界面技术。

WPF 用 XAML 语言制作界面，这是一种基于 XML 的标记语言，WPF 运行环境会根据这些标记、在终端界面上用适当的方法“绘制”界面。用 WPF 制作的界面可以不加修改的显示于 Windows Vista 客户端、IE7 浏览器窗口中、Windows Mobile 移动设备。

XAML 的主要艺术设计工具是 Microsoft Expression Blend，但是由于我的显示器分辨率有限，我仍然使用了 Visual C# 2008 Express Edition 以手工为主编写了 XAML。

XAML 中不能写入行为逻辑，控件的行为以类似 ASP.NET 的 code-behind 方式，用 C#语言写出。

我选择 WPF 进行工作台的开发，是因为我具有多年的 Web 开发经验，而 XAML 是一种标记语言，比较接近于 Web 开发的习惯。

3.2.3 RibbonBar

符合用户习惯的界面设计技术。

RibbonBar 是 Microsoft 在 2007 Office System 中首先推出的界面设计技术。取消了操作繁杂的菜单、将应用程序的所有功能用大大的按钮安排在窗口顶部。程序的功能一目了然，用户再也不需要层层深入查找所需功能。

借还书、读者服务工作台采用 RibbonBar 设计，可以提高工作效率。

本程序的 RibbonBar 采用 radRibbonBar 控件制作，注册 Visual C# 2008 Express Edition 后可以在 Microsoft Connect 免费获得这个控件。

3.3 API 和工作台编译方法

安装好 Visual C# 2008 Express Edition、或者 Visual Studio 2008 的任意版本。解开 vcs2008.tar，打开 library.sln，Build Solution 即可。注意要在 bin 目录运行 LibraryLogin.exe，Debug 目录的程序不保证能运行。

3.4 Credits

- 本程序使用了 famfamfam.com 的 Silk Icons