中国科学技术大学

University of Science and Technology of China

图书馆管理系统课程设计报告

姓名：高茂航、李宇湘

日期：2024.7.20

1

2

3 数据库概念结构设计

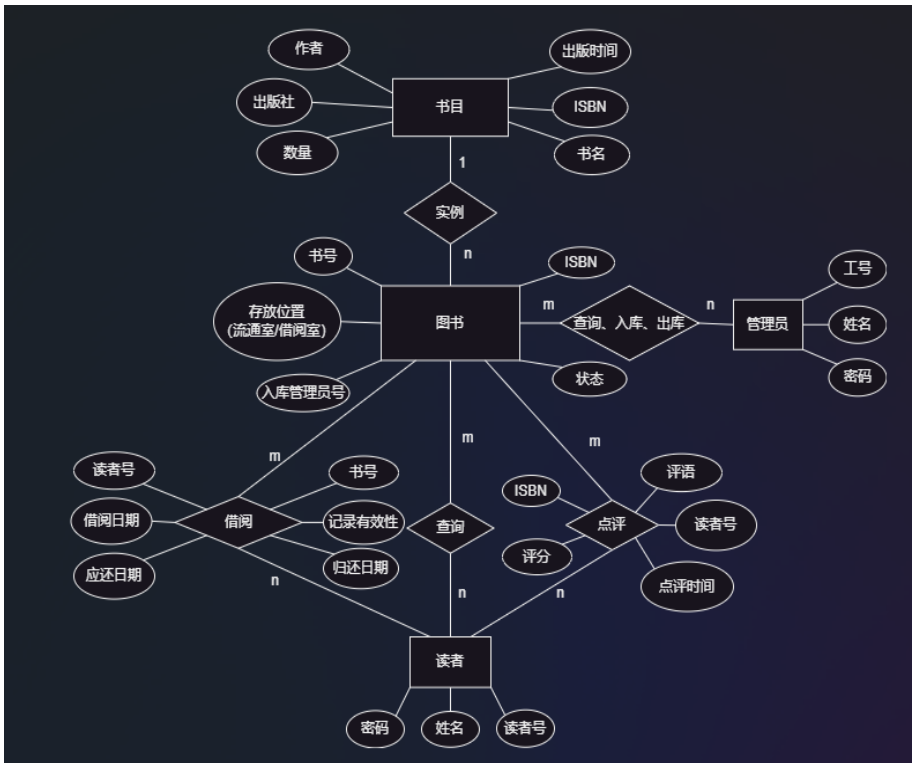

图 1: ER 图

4 数据库逻辑结构设计 (models.py)

4.1 数据库表结构

表名 (表名含义)	列名 (注释)	数据类型
dzTable (读者表)	dzid (读者 ID)	AutoField
	psw (密码)	CharField(max_length=256)
	xm (姓名)	CharField(max_length=10)
tsglyTable (图书管理员表)	glyid (管理员 ID)	CharField(max_length=10)
	psw (密码)	CharField(max_length=256)
	xm (姓名)	CharField(max_length=10)
smTable (书目表)	isbn	CharField(max_length=50)
	sm (书名)	CharField(max_length=50)
	zz (作者)	CharField(max_length=50)

表名 (表名含义)	列名 (注释)	数据类型
	cbs (出版社)	CharField(max_length=50)
	cbny (出版时间)	DateTimeField
	count (数量)	IntegerField(default=0)
tsTable (图书实例表)	tsid (图书 ID)	AutoField
	isbn (对应书目)	ForeignKey(smTable)
	cfwz (存放位置)	CharField(max_length=20)
	zt (状态)	CharField(max_length=20)
	jbr (入库该书管理员工号)	ForeignKey(tsglyTable)
BookReview (书评表)	dzid (读者 ID)	ForeignKey(dzTable)
	isbn	ForeignKey(smTable)
	score (评分)	IntegerField(validators=[0,10])
	comment (评论)	TextField(max_length=300, null=True)
	comment_time (评论时间)	DateTimeField(auto_now_add=True)
jsTable (借书表)	dzid (读者 ID)	ForeignKey(dzTable)
	tsid (图书 ID)	ForeignKey(tsTable, null=True)
	jysj (借阅时间)	DateTimeField
	yhsj (应还时间)	DateTimeField
	ghsj (归还时间)	DateTimeField(null=True)
	is_valid (记录是否有效)	BooleanField(default=True)

4.2 具体分析

4.2.1 dzTable (读者表)

- **结构**: 包含读者 ID(主码)、密码、姓名。
- **解释**: 此表用于存储图书馆读者的基本信息。
- **3NF 分析**: 满足 3NF，因为每个非主属性完全函数依赖于主码(读者 ID)，不存在传递依赖或部分依赖。
- **BCNF 分析**: 满足 BCNF，由于不存在非主属性对主码的部分或传递依赖，同时每个候选码也是超码。
- **4NF 分析**: 满足 4NF，因为表中不存在多值依赖。

4.2.2 tsglyTable (图书管理员表)

- **结构**: 包含管理员 ID(主码)、密码、姓名。

- **解释**: 此表用于存储图书管理员的基本信息。
- **3NF 分析**: 满足 3NF，每个非主属性完全函数依赖于主码（管理员 ID），不存在传递依赖或部分依赖。
- **BCNF 分析**: 满足 BCNF，因为不存在非主属性对主码的部分或传递依赖，且每个候选码也是超码。
- **4NF 分析**: 满足 4NF，因为表中不存在多值依赖。

4.2.3 smTable (书目表)

- **结构**: 包含 ISBN（主码）、书名、作者、出版社、出版时间、数量。
- **解释**: 此表记录了馆藏每个 ISBN 的书目的详细信息。
- **3NF 分析**: 满足 3NF，因为每个非主属性完全函数依赖于主码（ISBN），不存在传递依赖或部分依赖。
- **BCNF 分析**: 满足 BCNF，因为不存在非主属性对主码的部分或传递依赖，且每个候选码也是超码。
- **4NF 分析**: 满足 4NF，因为表中不存在多值依赖。

4.2.4 tsTable (图书实例表)

- **结构**: 包含图书 ID（主码）、该书的 ISBN、存放位置（流通室/阅览室）、状态（不外借/未借出/已借出）、经办人。
- **解释**: 此表记录了图书馆中每本图书的具体信息。
- **3NF 分析**: 满足 3NF，因为每个非主属性完全函数依赖于主码（图书 ID），不存在传递依赖或部分依赖。
- **BCNF 分析**: 满足 BCNF，因为不存在非主属性对主码的部分或传递依赖，且每个候选码也是超码。
- **4NF 分析**: 满足 4NF，因为表中不存在多值依赖。

4.2.5 BookReview (书评表)

- **结构**: 包含读者 ID、ISBN、评分、评论、评论时间，其中读者 ID 和 ISBN 共同构成复合主码。
- **解释**: 此表用于存储读者对图书的评分和评论。
- **3NF 分析**: 满足 3NF，因为每个非主属性完全函数依赖于复合主码（读者 ID 和 ISBN），不存在传递依赖或部分依赖。
- **BCNF 分析**: 满足 BCNF，因为不存在非主属性对复合主码的部分或传递依赖，且每个候选码也是超码。
- **4NF 分析**: 满足 4NF，因为表中不存在多值依赖。

4.2.6 jsTable (借书表)

- **结构**: 包含读者 ID、图书 ID、借阅时间、应还时间、归还时间、记录是否有效，其中读者 ID、图书 ID 和借阅时间共同构成复合主码。
- **解释**: 此表记录了读者借阅图书的详细信息，其中记录是否有效表示借过的书是否仍在库，这样可以避免将借过的书出库后借书记录丢失。
- **3NF 分析**: 满足 3NF，因为每个非主属性完全函数依赖于复合主码（读者 ID、图书 ID 和借阅时间），不存在传递依赖或部分依赖。
- **BCNF 分析**: 满足 BCNF，因为不存在非主属性对复合主码的部分或传递依赖，且每个候选码也是超码。
- **4NF 分析**: 满足 4NF，因为表中不存在多值依赖。

4.3 总结

所有表均满足第三范式 (3NF) 和 BCNF 的要求，因为它们的每个非主属性都直接依赖于主码（或复合主码），并且不存在任何传递依赖或部分依赖。此外，所有表也满足第四范式 (4NF)，因为它们的表结构比较简单，不存在多值依赖。这样的设计确保了数据的一致性、减少了数据冗余，并且提高了查询效率。

5 数据库物理结构设计

数据库的物理设计一般依赖于相应的数据库管理系统，由系统自主完成，不用程序员过多考虑，但有时为了效率的问题或其他的要求，必须进行相应的物理设计（包括建立索引），以达到相应的要求。由于本图书馆管理系统的数据库表结构较为简单，且目前数据不多，因此暂未建立索引或进行其他物理结构设计，仅使用默认操作。

6 后端功能设计与实现 (views.py)

6.1 home 函数：主页

1. **功能**：home 函数的主要功能是渲染并返回主页模板，处理对网站主页的请求；
2. **实现逻辑**：
   (a) 接收一个请求对象作为参数；
   (b) 使用 Django 的 `render` 函数，将请求对象和指定的模板（'home.html'）作为参数传递；
   (c) 返回渲染后的 HTML 页面给用户。

6.2 login_view 函数：登录

1. **功能**：login_view 函数处理用户登录逻辑，支持两种用户类型：读者和管理员。根据用户类型和提供的凭证（用户名和密码），进行相应的身份验证，并根据验证结果重定向到不同的页面或返回错误信息；

2. **实现逻辑：**

   (a) 初始化一个空字典 `context` 用于存储模板变量；

   (b) 检查请求方法是否为 POST。如果不是，直接渲染并返回主页模板；

   (c) 从 POST 请求中获取用户名、密码和用户类型；

   (d) 验证用户名和密码是否提供。如果任一未提供，设置错误消息并重新渲染主页模板，显示错误消息；

   (e) 根据用户类型（管理员或读者）执行不同的逻辑：

      i. 管理员：使用用户名查询 `tsglyTable` 表，检查密码是否匹配。如果登录成功，设置会话变量并重定向到管理员首页；

      ii. 读者：使用用户名查询 `dzTable` 表，检查密码是否匹配。如果登录成功，设置会话变量并重定向到读者首页；

   (f) 如果用户名或密码不匹配，设置错误消息并重新渲染主页模板，显示错误消息。

在这个过程中，`context` 字典被用来传递模板变量（如错误消息），而 Django 的 session 机制被用来在登录成功后存储用户信息，以便于跨请求保持用户状态。

6.3 register 函数：读者注册

1. **功能：**此函数用于新用户注册账户；

2. **实现逻辑：**

   (a) 初始化一个空字典 `context` 用于存储传递给模板的数据；

   (b) 如果请求方法为 GET，返回注册页面模板；

   (c) 如果请求方法为 POST，执行注册逻辑：

      i. 从 POST 数据中获取用户输入的姓名（xm）、密码（mm）和密码确认（mmqr）；

      ii. 验证姓名、密码和密码确认是否都已填写，如果有任何一个未填写，则在 `context` 中设置错误消息并重新渲染注册页面；

      iii. 验证两次输入的密码是否一致，如果不一致，则设置错误消息并重新渲染注册页面；

      iv. 验证密码长度是否至少为六位，如果不是，则设置错误消息并重新渲染注册页面；

      v. 检查用户名是否已被使用，如果是，则设置错误消息并重新渲染注册页面；

      vi. 如果用户名未被使用，计算新用户的 ID，创建新用户记录，并保存到数据库；

      vii. 注册成功后，重定向到登录页面。

   (d) 如果请求方法既不是 GET 也不是 POST，则返回注册页面模板。

6.4 logout_view 函数

1. **功能：**此函数用于处理读者和管理员的退出登录操作；

2. **实现逻辑：**

   (a) 检查 session 中是否存在 `login_type`，如果存在，则清空 session；

   (b) 重定向到网站主页。

6.5 dz_index 函数：读者首页

1. **功能**：显示读者的首页信息，包括当前借阅、历史借阅记录、用户排名等；
2. **实现逻辑**：
   (a) 验证用户登录类型是否为读者，如果不是，则重定向到主页；
   (b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；
   (c) 查询当前用户的所有借阅记录，并筛选出当前借阅记录（未归还的）；
   (d) 计算总用户数；
   (e) 基于借阅数量计算当前用户的排名；
   (f) 遍历所有借阅记录，提取书籍信息和借阅状态，存入列表 `grzt`；
   (g) 使用 `Paginator` 对 `grzt` 进行分页处理；
   (h) 从请求中获取页码，获取对应的页面对象 `page_obj`，并将其及其他信息添加到上下文；
   (i) 渲染并返回 `dz_index.html` 页面。

6.6 current_borrows_view 函数：当前借阅书籍

1. **功能**：显示读者当前借阅的书籍；
2. **实现逻辑**：
   (a) 验证用户登录类型是否为读者，如果不是，则重定向到主页；
   (b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；
   (c) 查询当前用户的所有未归还借阅记录；
   (d) 遍历当前借阅记录，提取书籍信息和借阅状态，存入列表 `grzt`；
   (e) 使用 `Paginator` 对 `grzt` 进行分页处理；
   (f) 从请求中获取页码，获取对应的页面对象 `page_obj`，并将其及其他信息添加到上下文；
   (g) 渲染并返回 `current_borrows.html` 页面。

6.7 book_details 函数：书籍详情

1. **功能**：显示一本书的详细信息，包括书籍信息和书评；
2. **实现逻辑**：
   (a) 使用 `isbn` 参数从 `tsTable` 表中过滤出书籍信息，存储在 `books_info` 变量中；
   (b) 使用 `isbn` 参数从 `BookReview` 表中过滤出评分，并计算平均值，如果没有评分则默认为 0，结果存储在 `average_score` 变量中；
   (c) 使用 `isbn` 参数从 `BookReview` 表中过滤出所有评论，存储在 `reviews` 变量中；
   (d) 创建一个字典 `context`，包含用户会话中的姓名和 ID，书籍信息，平均评分，和评论列表；
   (e) 使用 `render` 函数渲染 `book_details.html` 模板，并传递 `context` 字典.

6.8 dz_smztcx 函数:读者书目状态查询

1. **功能**:允许读者查询书目状态,包括书籍的详细信息和书评提交功能;
2. **实现逻辑**:
   (a) 验证用户登录类型是否为读者,如果不是,则重定向到主页;
   (b) 初始化上下文字典 `context`,并从会话中获取用户姓名和 ID,存入上下文;
   (c) 如果是 POST 请求且包含书评信息,则处理书评提交:
      i. 从 POST 请求中获取 ISBN 码、评分和评论内容;
      ii. 尝试根据 ISBN 码在 `smTable` 表中找到对应的书籍实例;
      iii. 如果找到,则在 `BookReview` 表中创建一条新的书评记录,并返回成功消息;
      iv. 如果未找到对应的书籍实例,则返回错误消息;
   (d) 如果是 GET 请求,则直接渲染并返回 `dz_smztcx.html` 页面;
   (e) 如果是 POST 请求但不是提交书评的请求,则处理书目查询:
      i. 从 POST 请求中获取书名、作者、ISBN 码和出版社信息,并存入上下文;
      ii. 如果书名为空,则返回错误消息并提示输入书名进行搜索;
      iii. 使用提供的信息对 `smTable` 表进行过滤查询,实现模糊搜索;
      iv. 遍历查询结果,提取每本书的详细信息和状态,存入列表 `smzt`;
      v. 将 `smzt` 列表和其他信息添加到上下文;
      vi. 渲染并返回 `dz_smztcx.html` 页面。

6.9 check_review 函数:检查是否已经评论过

1. **功能**:检查用户是否已经对某本书进行了评论;
2. **实现逻辑**:
   (a) 从会话中获取用户 ID;
   (b) 使用用户 ID 和 ISBN 码查询 `BookReview` 表,检查是否存在评论记录;
   (c) 如果存在评论记录,返回提示信息,说明用户已经评论过该书;
   (d) 如果不存在评论记录,返回空消息,允许用户进行评论;
   (e) 返回包含检查结果和消息的 JSON 响应。

6.10 submit_review 函数:提交评论

1. **功能**:允许用户提交对书籍的评论;
2. **实现逻辑**:
   (a) 检查请求方法是否为 POST,如果不是,返回错误消息;
   (b) 从 POST 请求中获取 ISBN 码、评分和评论内容;
   (c) 从会话中获取用户 ID;
   (d) 如果评分为空,返回错误消息,说明评分是必填项;

(e) 尝试根据 ISBN 码在 `smTable` 表中找到对应的书籍实例；

(f) 如果找到书籍实例，创建一条新的书评记录，并返回成功消息；

(g) 如果未找到书籍实例，返回错误消息，说明提供的 ISBN 码无效；

6.11 dz_js 函数：读者借书

1. **功能**：实现读者借书的功能；

2. **实现逻辑**：

(a) 验证用户登录类型是否为读者，如果不是，则重定向到主页；

(b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；

(c) 如果是 GET 请求，直接渲染并返回 `dz_js.html` 页面；

(d) 如果是 POST 请求，处理借书逻辑：

i. 从 POST 请求中获取 ISBN 码，并存入上下文；

ii. 如果 ISBN 码为空，返回错误消息并提示填写完整的 ISBN 号；

iii. 尝试根据 ISBN 码在 `smTable` 表中找到对应的书籍实例；

iv. 如果未找到书籍实例，返回错误消息，说明 ISBN 号填写错误；

v. 检查当前用户的未归还借阅记录数是否已达上限；

vi. 如果已达上限，返回错误消息，说明借阅书籍数已达上限；

vii. 检查所选图书的状态是否为“未借出”；

viii. 如果所有图书已被借出，返回错误消息，说明无法借阅；

ix. 更新图书状态为“已借出”，并创建借书记录；

x. 返回成功消息，提示借阅成功，并显示图书 ID，借阅时间为 60 天，不支持续借。

6.12 dz_hs 函数：读者还书

1. **功能**：实现读者还书的功能；

2. **实现逻辑**：

(a) 验证用户登录类型是否为读者，如果不是，则重定向到主页；

(b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；

(c) 如果是 GET 请求，直接渲染并返回 `dz_hs.html` 页面；

(d) 如果是 POST 请求，处理还书逻辑：

i. 从 POST 请求中获取图书 ID，并存入上下文；

ii. 如果图书 ID 为空或不是数字，返回错误消息；

iii. 查询图书 ID 是否存在，如果不存在，返回错误消息；

iv. 查询该读者是否有未归还的借书记录，如果没有，返回错误消息；

v. 检查图书是否逾期未还，如果逾期，计算并提示逾期费用；

vi. 更新图书状态为“未借出”，并记录归还时间；

vii. 返回成功消息，提示图书已归还。

6.13 my_reviews 函数:读者评书记录

1. **功能:**显示读者的评书记录;
2. **实现逻辑:**
   (a) 从会话中获取读者 ID,并初始化上下文字典 `context`;
   (b) 查询当前读者的所有评书记录,并按评论时间排序;
   (c) 使用 `Paginator` 对评书记录进行分页处理;
   (d) 从请求中获取页码,获取对应的页面对象 `page_obj`,并将其及其他信息添加到上下文;
   (e) 遍历分页后的评书记录,提取书籍信息、评分、评论内容等,存入列表;
   (f) 渲染并返回 `my_reviews.html` 页面。

6.14 revoke_review 函数:撤销评论

1. **功能:**允许读者撤销自己的评论;
2. **实现逻辑:**
   (a) 从 POST 请求的 body 中解析 JSON 数据,获取评论 ID;
   (b) 尝试根据评论 ID 找到对应的评论记录;
   (c) 如果找到,删除该评论记录,并返回成功消息;
   (d) 如果未找到或发生其他错误,返回错误消息。

6.15 ranking 函数:展示排名信息

1. **功能:**展示评分最高的十本书及其评分,以及被借阅次数最多的十本书及其被借次数。
2. **实现逻辑:**
   (a) 验证用户登录类型是否为读者,如果不是,则重定向到主页;
   (b) 初始化上下文字典 `context`,并从会话中获取用户姓名和 ID,存入上下文;
   (c) 获取评分最高的十本书及其评分,确保书籍有评分,存入上下文;
   (d) 获取被借阅次数最多的十本书及其被借次数,确保书籍被借阅过,存入上下文;
   (e) 渲染并返回 `ranking.html` 页面,展示排名信息。

6.16 gly_index 函数:管理员首页

1. **功能:**显示管理员首页。
2. **实现逻辑:**
   (a) 验证用户登录类型是否为管理员,如果不是,则重定向到主页;
   (b) 初始化上下文字典 `context`,并从会话中获取管理员姓名和 ID,存入上下文;
   (c) 渲染并返回 `gly_index.html` 页面,显示管理员首页。

6.17 book_details2 函数:管理员页面书籍详情

1. **功能:**在管理员页面显示书籍的详细信息。
2. **实现逻辑:**
   (a) 根据传入的 ISBN 查询书籍信息;
   (b) 查询并计算书籍的平均评分;
   (c) 查询书籍的所有评论;
   (d) 初始化上下文字典 `context`,并将查询到的信息存入上下文;
   (e) 渲染并返回 `book_details2.html` 页面,展示书籍详情。

6.18 gly_smztcx 函数:管理员书目状态查询

1. **功能:**允许管理员根据书名、作者、ISBN、出版社等信息查询书目状态。
2. **实现逻辑:**
   (a) 验证用户登录类型是否为管理员,如果不是,则重定向到主页;
   (b) 初始化上下文字典 `context`,并从会话中获取管理员姓名和 ID,存入上下文;
   (c) 如果是 GET 请求,直接渲染并返回 `gly_smztcx.html` 页面;
   (d) 如果是 POST 请求,处理书目状态查询逻辑:
      i. 从 POST 请求中获取书名、作者、ISBN、出版社等信息;
      ii. 如果书名为空,返回错误消息;
      iii. 使用模糊搜索查询满足条件的书目,并计算每本书的在库册数、不外借册数、未借出册数、已借出册数;
      iv. 将查询结果和其他信息存入上下文;
      v. 渲染并返回 `gly_smztcx.html` 页面,展示查询结果。

6.19 smzt_all 函数:所有书目状态查询

1. **功能:**显示所有书目的状态信息;
2. **实现逻辑:**
   (a) 初始化上下文字典 `context`,并从会话中获取管理员 ID,存入上下文;
   (b) 查询所有书目的信息;
   (c) 遍历查询结果,对每本书进行以下操作:
      i. 提取书目的基本信息(ISBN、书名、作者、出版社、出版年月);
      ii. 查询并计算该书目的库存总数、不外借数量、未借出数量、已借出数量;
      iii. 将上述信息存入列表 `smzt` 中;
   (d) 使用 `Paginator` 对 `smzt` 进行分页处理,每页显示 10 个书目;
   (e) 从请求中获取页码,获取对应的页面对象 `page_obj`,并将其及其他信息添加到上下文;
   (f) 渲染并返回 `smzt_all.html` 页面。

6.20 borrowed_books 函数：所有借阅信息

1. **功能：**显示所有当前借阅信息；
2. **实现逻辑：**
   (a) 验证用户登录类型是否为管理员，如果不是，则重定向到主页；
   (b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；
   (c) 查询所有当前未归还的借阅记录，并按读者 ID 排序；
   (d) 将查询结果存入上下文；
   (e) 渲染并返回 `borrowed_books.html` 页面。

6.21 gly_rk 函数：管理员入库

1. **功能：**管理员进行书籍入库操作；
2. **实现逻辑：**
   (a) 验证用户登录类型是否为管理员，如果不是，则重定向到主页；
   (b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；
   (c) 如果请求方法为 GET，直接渲染并返回 `gly_rk.html` 页面；
   (d) 如果请求方法为 POST，执行以下操作：
      i. 从请求中获取 ISBN、入库数量、入库后状态、书名、作者、出版社、出版年月等信息，并存入上下文；
      ii. 验证必要信息的完整性，不完整则返回错误信息；
      iii. 根据 ISBN 查询书目，判断是旧书入库还是新书入库；
      iv. 对于旧书入库，进一步验证书名、作者、出版社、出版年月的匹配性，不匹配则返回错误信息；
      v. 根据入库后状态（流通室、阅览室），创建相应数量的图书实例，并保存；
      vi. 更新书目的库存总数，并保存；
      vii. 设置成功信息，并可能刷新页面；
      viii. 渲染并返回 `gly_rk.html` 页面，显示操作结果。

6.22 gly_ck 函数：管理员出库

1. **功能：**管理员进行书籍出库操作；
2. **实现逻辑：**
   (a) 验证用户登录类型是否为管理员，如果不是，则重定向到主页；
   (b) 初始化上下文字典 `context`，并从会话中获取用户姓名和 ID，存入上下文；
   (c) 如果请求方法为 GET，直接渲染并返回 `gly_ck.html` 页面；
   (d) 如果请求方法为 POST，执行以下操作：
      i. 从请求中获取 ISBN、出库数量、出库优先位置，并存入上下文；

ii. 验证必要信息的完整性，不完整则返回错误信息；
iii. 验证出库优先位置是否合法（流通室或阅览室），不合法则返回错误信息；
iv. 根据 ISBN 查询书目，如果不存在则返回错误信息；
v. 计算未借出和不外借的图书数量，以及所有图书数量；
vi. 如果出库数量超过藏书总数或可出库数量，返回错误信息；
vii. 根据出库优先位置，优先出库未借出或不外借的图书；
viii. 更新图书状态为无效，并删除图书记录；
ix. 更新书目的库存总数，如果没有剩余的书，则删除书目记录；
x. 设置成功信息，并返回出库成功的图书 ID；
xi. 渲染并返回 `gly_ck.html` 页面，显示操作结果。

6.23 book__count__view 函数：书籍借阅次数统计

1. **功能：**统计并显示各书籍的借阅次数；
2. **实现逻辑：**
   (a) 从 `jsTable` 中查询每本书的 ISBN、书名和借阅次数；
   (b) 对查询结果按借阅次数降序排序；
   (c) 将查询结果存入上下文字典 `context`；
   (d) 渲染并返回 `book_count.html` 页面。

6.24 reader__count__view 函数：读者借阅次数统计

1. **功能：**统计并显示各读者的借阅次数；
2. **实现逻辑：**
   (a) 从 `jsTable` 中查询每位读者的 ID、姓名和借阅次数；
   (b) 对查询结果按借阅次数降序排序；
   (c) 将查询结果存入上下文字典 `context`；
   (d) 渲染并返回 `reader_count.html` 页面。

7 一些 bug 及修改思路

7.1 管理员注册

一开始管理员和读者注册的逻辑是一样的，但是经助教提醒，发现这样会导致信息的泄露。助教建议我们在新管理员注册后需要得到老管理员的批准，但我们发现这样实现起来比较困难，因此我们决定使用超级管理员总结在数据库中添加图书管理员的方式来实现管理员注册。

7.2 views 和 templates 的对应关系

有几次 views 传入的参数不足，导致渲染模板时出现错误，但报错信息并不明显，导致花了很多时间来找错误。

8 待完善的功能

- **图书图片展示**：由于时间原因，我们没有在书籍详情页面实现图书图片的展示；
- **图书分类和搜索**：在实际应用中，用户可能需要根据图书的分类或关键字进行搜索，这是一个可以增加系统实用性的功能；
- **图书借阅提醒**：用户借阅图书后，系统可以通过邮件或短信提醒用户还书日期，这是一个可以提高系统友好性的功能，但涉及到其他较为复杂的操作，暂时没有完成；
- **图书预约**：用户可以预约已被借出的图书，系统会在图书归还后通知该用户，且其他用户不能借阅被预约的书，这是一个可以提高系统服务质量的功能；
- **图书续借**：用户可以对已借阅的图书进行续借操作，延长借阅时间；
- **图书荐购**：用户可以向管理员推荐购买某本书，管理员可以根据用户的推荐进行购买决策；
- **图书推荐**：根据用户的借阅历史和评分记录，系统可以推荐用户可能感兴趣的图书，这是一个可以提高系统个性化的功能；
- **图书分类统计**：管理员可能需要查看每个分类的图书数量，以便更好地管理图书馆的藏书；
- **图书借阅统计**：管理员可能需要查看每本书的借阅次数，以便更好地了解读者的阅读喜好；
- **入库图书统计**：管理员可能需要查询自己入库的所有书籍。

9 总结

9.1 实验收获

在本次实验中，我们成功设计并实现了一个图书馆管理系统的基本功能。通过此次实验，我们收获了以下几点：

- **数据库设计与范式理论应用**：
  - 通过具体实践，我们学会了如何将业务需求转化为数据库表结构，并且掌握了表之间关系的设计，包括一对一、一对多、多对多等关系；
  - 我们也通过对每个表的 3NF、BCNF 和 4NF 的分析，更加深刻地理解了数据库设计的范式理论，以及如何通过范式理论减少数据冗余和提高数据一致性。
- **Django ORM(Object-Relational Mapping) 工具的使用**：
  - 学习了如何使用 Django 的 ORM 来定义数据库模型，并且通过这些模型与数据库进行交互，简化了数据操作的复杂度。
  - 掌握了如何在 Django 中进行数据迁移，维护数据库表结构的变更。
- **团队合作**：

  - 通过分工合作，培养了团队合作能力，提高了沟通和协调能力，确保了项目的顺利进行。

- **持续学习：**

  - 数据库技术在不断进步，新的设计理念和技术工具不断涌现。我们需要保持学习的态度，不断更新知识，以适应技术发展的需要。

9.2 一些问题和教训

- **需求分析不够全面：**

  - 在初期的需求分析阶段，由于考虑不够全面，导致后期需要对数据库表结构进行多次调整。这提醒我们在进行系统设计前，需要进行更为详细和全面的需求分析。

- **表结构调整的复杂性：**

  - 由于初始设计的一些缺陷，后期在对表结构进行调整时，涉及到的数据迁移和数据一致性问题较为复杂。这也提醒我们在设计数据库时，需要充分考虑未来可能的变化和扩展。

- **数据量和性能问题：**

  - 数据库设计中的每一个细节都可能影响到系统的性能和稳定性，虽然当前系统数据量较少，但随着数据量的增加，某些查询的效率可能会下降。这提示我们在实际应用中，需要考虑性能优化的问题，如建立索引、优化查询等。

- **测试不足：**

  - 在功能实现后，对系统的测试可能不够全面，不能保证所有功能在所有情况下都不会出现问题。

- **没写实验日志：**

  - 由于是初次完成这种较大的项目，因此没有记录完成每部分内容的时间点以及遇到的问题和解决方案，导致后期写实验报告时有些内容忘记了。