附件 1

《数据库系统原理》大作业

系统设计报告

题目名称：Ting2Django北航校园论坛

学号及姓名： 22373537 张馨月 _（组长）

22371351 王宇汀

22371032 张峻誉

2024 年 12 月 12 日

Ting2Django北航校园论坛___系统设计报告

组内同学承担任务说明

学生姓名	子任务1：系统功能设计与数据库设计	子任务2：系统服务器端开发	子任务3：系统客户端开发	工作量占比
王宇汀	讨论想法，功能、页面设计	前后端连接，敲定接口文档，测试后端接口	前端程序架构设计和实现；网页端测试；前后端代码测试；前端功能补充	33.3%
张峻誉	建立数据表、E-R图	数据库存储定义与实现，后端架构设计与实现	前后端代码测试	33.3%
张馨月	讨论想法，功能、页面设计	前后端连接，敲定接口文档，测试后端接口	前端程序架构实现；网页端测试；前后端代码测试；前端布局美化	33.3%

一、需求分析

1.需求描述

Ting2Django北航校园论坛是一款全匿名性质的交流平台。旨在为学生、老师及其他校园成员提供一个在线交流和信息分享的平台。用户可以发布帖子、评论、点赞和收藏感兴趣的内容，使用标签进行内容分类，并通过私信功能与其他用户进行交流。此外，管理员可以管理用户、帖子和评论，以维护论坛的良好秩序。

具体功能描述 得改

1. 用户功能

- **注册与登录**：用户可以注册新账户，使用用户名和密码进行登录。
- **用户登出**：用户可以退出登录。
- **个人资料管理**：用户可以查看和编辑个人资料，包括头像、简介等信息。
- **标签管理**：用户可以关注感兴趣的标签，以便获取相关内容。

2. 帖子功能

- **发帖**：用户可以发布帖子，包含标题、内容和标签。
- **查看帖子流**：用户可以浏览所有帖子，筛选并查看不同分区的帖子。
- **帖子分类**：根据标签对帖子进行分类，便于用户查找。
- **帖子热度**：用户可以在首页看到帖子热榜，了解不同类型的热榜话题。
- **删除帖子**：用户可以删除自己发布的贴子。

3. 评论功能

- **评论帖子**：用户可以对帖子进行评论，发表自己的看法。
- **查看评论**：用户可以查看帖子的所有评论，了解其他用户的观点。
- **删除评论**：用户可以删除自己发布的评论，帖子拥有者可以删除帖子的任意评论。

4. 点赞功能

- **点赞帖子**：用户可以对帖子进行点赞，表达对内容的认可。
- **查看点赞**：用户可以查看自己点赞的帖子列表。
- **查看点赞数**：用户可以查看每个帖子的点赞数量，了解其受欢迎程度。

5. 收藏功能

- **收藏帖子**：用户可以将感兴趣的帖子收藏，方便日后查阅。
- **查看收藏**：用户可以查看自己收藏的帖子列表。
- **查看收藏数**：用户可以查看每个帖子的收藏数量，了解其受欢迎程度。

6. 私信功能

- **发送私信**：用户可以向其他用户发送私信，进行一对一交流。
- **查看私信**：用户可以查看自己的私信记录，下拉可以刷新记录。
- **查看联系人**：用户可以查看与自己有消息往来的用户。

7. 管理员功能

- **用户管理**：管理员可以查看、编辑或删除用户信息。
- **帖子管理**：管理员可以审核、删除不当帖子，维护论坛秩序。
- **评论管理**：管理员可以管理评论，删除不当评论。

8. 标签分区功能

- **帖子分区**：用户可以根据分区对帖子进行筛选，查看自己感兴趣的分区。
- **标签管理**：用户可以查看所有标签、以及使用了该标签的所有帖子。
- **问题帖子**：用户可以发布问题和文章，作为帖子的两种表现形式。

9. 搜索通知功能

- **字符串搜索**：用户可以输入字符串，搜索包含该字符串的帖子、用户或者标签。
- **消息通知**：当用户收到点赞、收藏或者评论时，系统会以私信的形式通知用户。

2.数据流图

顶层数据流图

3.数据元素表

1. 用户表 (Users)

功能: 存储论坛用户的信息，包括用户名、密码和其他个人资料。
字段:

- id: 唯一标识每个用户的主键。
- username: 用户的昵称，不同用户的用户名必须唯一。
- email: 用户的电子邮件地址，在系统中也需唯一，通常用于密码重置等功能。
- password: 存储用户密码，以保护用户的隐私安全，建议使用加密存储。
- created_at: 记录用户账户的创建时间，用于审计和用户管理。
- last_login: 记录用户最后一次登录的时间，方便追踪账户活动。
- profile: 用户个人简介，以展示在用户个人资料或帖子中。
- avatar: 用户头像的 URL，用于在论坛中显示用户的个人形象。

字段名	类型	含义
id	int	用户唯一标识
username	varchar(50)	用户名
email	varchar(100)	用户邮箱
password	varchar(255)	用户密码
created_at	datetime	用户注册时间
last_login	datetime	最后登录时间
profile	varchar(255)	用户个人简介
avatar	varchar(255)	用户头像

2. 帖子表 (Posts)

功能: 存储用户发布的帖子信息，包括帖子标题、内容和相关属性。
字段:

- post_id: 唯一标识每个帖子的主键。
- user_id: 发布帖子的用户ID，关联到用户表，表示该帖子由哪个用户发布。
- post_area: 帖子所属的区域或分类，用于帖子管理和展示。
- post_type: 帖子的类型（如问答、分享等），用于内容分类和过滤。
- post_title: 帖子的标题，简要描述帖子内容。
- created_at: 记录帖子创建的时间，用于排序和管理。

字段名	类型	含义
post_id	int	帖子唯一标识
user_id	int	发布用户的ID
post_area	int	帖子区域
post_type	varchar(10)	帖子类型
post_title	varchar(255)	帖子标题
created_at	datetime	帖子创建时间

3. 用户-标签表 (UserTags)

功能: 存储用户关注的标签信息，以便用户获取相关内容。
字段:

- id: 唯一标识每个记录的主键。
- user_id: 关注标签的用户ID，关联到用户表。
- tag: 用户关注的标签名称，帮助用户定制内容展示。

字段名	类型	含义
id	int	唯一标识
user_id	int	用户ID
tag	varchar(255)	用户关注的标签

4. 帖子-标签表 (PostTags)

功能: 存储帖子使用的标签，以便对帖子进行分类和搜索。
字段:

- id: 唯一标识每个记录的主键。
- post_id: 关联的帖子ID，表示该标签属于哪个帖子。
- tag: 帖子使用的标签名称，便于用户通过标签查找相关帖子。

字段名	类型	含义
id	int	唯一标识
post_id	int	帖子ID
tag	varchar(255)	帖子使用的标签

5. 评论表 (Comments)

功能: 存储用户对帖子发表的评论，促进用户之间的互动。
字段:

- comment_id: 唯一标识每条评论的主键。
- post_id: 评论所属的帖子ID，关联到帖子表。
- user_id: 发表评论的用户ID，关联到用户表。
- content: 评论的内容，记录用户对帖子的看法和反馈。
- created_at: 记录评论创建的时间，用于排序和管理。

字段名	类型	含义
comment_id	int	评论唯一标识
post_id	int	评论所属帖子ID
user_id	int	评论用户ID
content	TEXT	评论内容
created_at	datetime	评论创建时间

6. 帖子-内容表 (PostContents)

功能: 存储帖子的详细内容，可以用于多种内容类型。
字段:

- id: 唯一标识每条内容的主键。
- post_id: 关联的帖子ID，表示该内容属于哪个帖子。
- type: 内容的类型（如文本、图片、视频等），用于内容展示。
- data: 内容数据，存储具体的内容信息。

字段名	类型	含义
id	int	唯一标识
post_id	int	帖子ID
type	varchar(10)	内容类型
data	TEXT	内容数据

7. 点赞表 (Likes)

功能: 存储用户对帖子进行点赞的记录，反映帖子受欢迎程度。
字段:

- id: 唯一标识每条点赞记录的主键。
- post_id: 点赞的帖子ID，关联到帖子表。
- user_id: 点赞用户的ID，关联到用户表。
- create_time: 点赞时间，记录用户何时对帖子进行点赞。

字段名	类型	含义
id	int	唯一标识
post_id	int	点赞的帖子ID
user_id	int	点赞用户ID
create_time	datetime	点赞时间

8. 收藏表 (Bookmarks)

功能: 存储用户对帖子进行收藏的记录，方便用户日后查看。
字段:

- id: 唯一标识每条收藏记录的主键。
- post_id: 收藏的帖子ID，关联到帖子表。
- user_id: 收藏用户的ID，关联到用户表。
- create_time: 收藏时间，记录用户何时对帖子进行收藏。

字段名	类型	含义
id	int	唯一标识
post_id	int	收藏的帖子ID
user_id	int	收藏用户ID
create_time	datetime	收藏时间

9. 消息表 (Chats)

功能: 存储用户之间的私信记录，促进用户之间的交流。
字段:

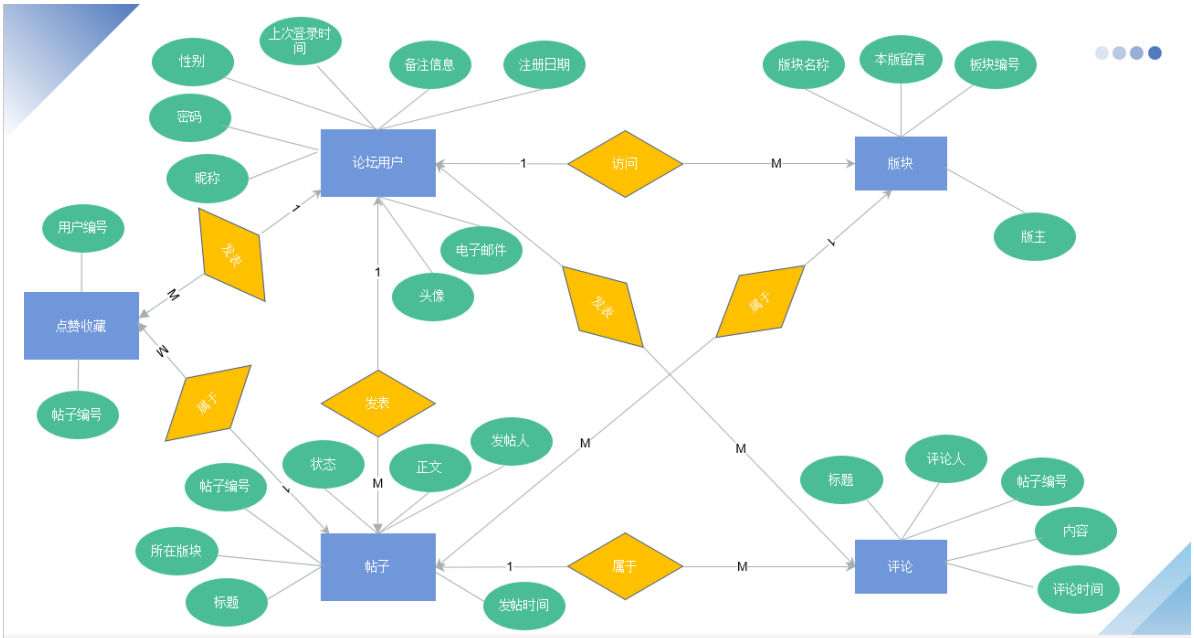
- chat_id: 唯一标识每条消息的主键。
- from_id: 发送者用户的ID，关联到用户表。
- to_id: 接收者用户的ID，关联到用户表。

- message: 消息内容，记录用户发送的具体信息。
- created_at: 消息发送时间，便于用户查看聊天记录的时间顺序。

字段名	类型	含义
chat_id	int	消息唯一标识
from_id	int	发送者用户ID
to_id	int	接收者用户ID
message	varchar(255)	消息内容
created_at	datetime	消息发送时间

二、数据库概念模式设计

系统初步E-R图

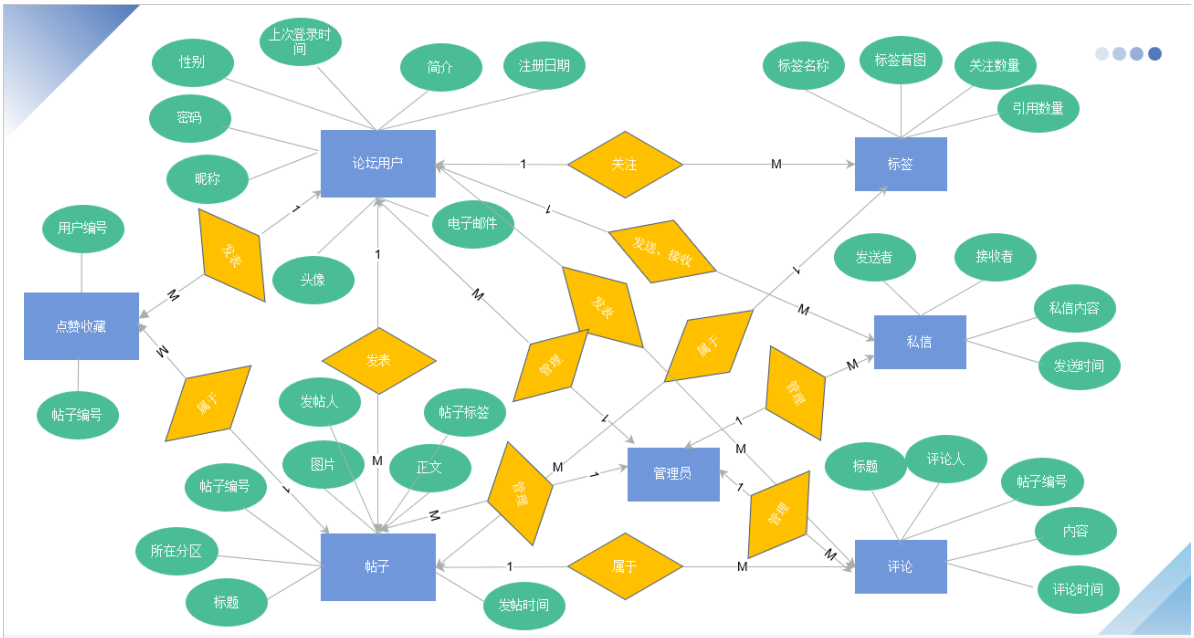


ER实体关系模型严格遵循 第三范式(3NF)

系统基本E-R图

系统后续的迭代中，实体设计主要有以下调整：

1. 用标签分区的思想取代了板块的功能，用户可以同时关注多个标签，帖子可以同时引用多个标签。
2. 新增了私信实体，用户之间可以发布多条私信，私信包含了发送、接收方信息、发送内容和发送时间。
3. 新增管理员，管理员隶属于用户实体，拥有管理用户、帖子、评论和私信的权限。



ER实体关系模型严格遵循 第三范式(3NF)

三、数据库逻辑模式设计与优化

1.数据库关系模式定义（由E-R图得到的关系模式）

1. 用户 (id, username, email, password, created_at, last_login, profile, avatar)
2. 帖子 (post_id, user_id, post_area, post_type, post_title, created_at)
3. 用户-标签 (id, user_id, tag)
4. 帖子-标签 (id, post_id, tag)
5. 评论(comment_id, post_id, user_id, content, created_at)
6. 帖子-内容 (id, post_id, type, data)
7. 点赞 (id, post_id, user_id, create_time)
8. 收藏 (id, post_id, user_id, create_time)
9. 消息 (chat_id, from_id, to_id, message, created_at)

2.关系模式范式等级的判定与规范化（已规范至3NF）

用户

字段: id, username, email, password, created_at, last_login, profile, avatar

- 每个用户都有一个唯一的 id，且每个字段都是对用户信息的直接描述。
- 该表满足第一范式（1NF），因为所有字段都具有原子性，并且没有重复的行。
- 该表不包含部分依赖，且每个字段完全依赖于主键，因此它是第二范式（2NF）。
- 该表不包含传递依赖，属于第三范式（3NF），是BCNF。

帖子

字段: post_id, user_id, post_area, post_type, post_title, created_at

- post_id 为主键，且所有其他字段都直接依赖于该主键。
- 该表满足1NF，无重复行和非原子值。
- 所有字段也都是完全依赖于主键，因此符合2NF。
- 无传递依赖，符合3NF，是BCNF。

用户-标签

字段: id, user_id, tag

- 主键 id 唯一标识每条记录，所有其他字段依赖于此主键。
- 满足1NF。
- 每个字段完全依赖于主键，符合2NF。
- 无传递依赖，符合3NF，是BCNF。

帖子-标签

字段: id, post_id, tag

- 主键 id 唯一标识每条记录。
- 满足1NF。
- 所有字段都依赖于主键，符合2NF。
- 无传递依赖，符合3NF，是BCNF。

评论

字段: comment_id, post_id, user_id, content, created_at

- 主键 comment_id 唯一标识评论。
- 该表满足1NF。
- 所有字段均完全依赖于主键，因此符合2NF。
- 无传递依赖，符合3NF，是BCNF。

帖子-内容

字段: id, post_id, type, data

- 主键 id 唯一标识每条内容。
- 满足1NF，无非原子值。
- 所有字段完全依赖于主键，符合2NF。
- 没有传递依赖，符合3NF，是BCNF。

点赞

字段: id, post_id, user_id, create_time

- 主键 id 唯一标识点赞记录。
- 满足1NF。
- 所有字段完全依赖于主键，符合2NF。
- 无传递依赖，符合3NF，是BCNF。

收藏

字段: id, post_id, user_id, create_time

- 主键 id 唯一标识收藏记录。
- 满足1NF。
- 所有字段完全依赖于主键，符合2NF。
- 无传递依赖，符合3NF，是BCNF。

消息

字段: chat_id, from_id, to_id, message, created_at

- 主键 chat_id 唯一标识每条消息。
- 满足1NF。
- 所有字段完全依赖于主键，符合2NF。
- 无传递依赖，符合3NF，是BCNF。

四、数据库关系模式优化

1. **索引优化**：
- 在经常查询的字段上建立索引，如 posts 表的 user_id、likes 表的 post_id 和 user_id，可以提高查询效率。
- 对 comments 表的 post_id 和 user_id 字段也可以建立索引，以加快评论查询速度。
2. **数据冗余**：
- 考虑在 posts 表中直接存储点赞数、评论数和收藏数，以减少查询时的计算开销，尤其是在访问量大的情况下。
3. **分表策略**：
- 对于高频访问的表（如 comments 和 likes），可以考虑使用分表策略，按时间或帖子进行分割，以提高性能。
4. **使用缓存**：
- 使用缓存技术（如 Redis）存储热门帖子和用户活动，减少数据库的直接访问频率，提高系统响应速度。
5. **数据归档**：
- 定期对旧数据进行归档，避免数据库表过于庞大造成性能下降，尤其是 comments 和 likes 表。
6. **外键约束**：
- 确保外键约束的合理性，避免在高并发情况下出现性能瓶颈。可以根据实际情况选择性地使用外键，确保数据完整性和性能之间的平衡。
7. **数据类型优化**：
- 根据实际需求选择合适的数据类型，避免使用过大的数据类型（如 TEXT），可以考虑使用 VARCHAR 或 JSON 存储结构化数据。