《数据库系统原理》大作业

系统设计报告

题目名称：航言·北航学生交流社区

学号及姓名：19373174 黄炜

19373225 汤裕

19373132 周子颖

2021年11月　22日

组内同学承担任务说明

	负责学生	备注
子任务 1：系统功能设计与数据库设计	黄炜	主要包括，数据库设计，前后端接口设计实现，Json文件传输设置，前端页面优化，前端业务逻辑，文档撰写
子任务 2：系统服务器端开发	周子颖	主要包括，后端代码实现，后端数据库建立，完成路由内容，后端框架结构设计，前端功能逻辑，数据库优化
子任务 3：系统客户端开发	汤裕	主要包括，前端页面设计实现，前端js页面内逻辑实现，前端框架结构设计，数据库优化

一. 需求分析

1. 需求描述

1.1 用户需求：

本系统的最终用户为北航在校大学生，根据我们在校生活的需求和经验，结合我们对身边同学关于社交和互动方面的需求的收集，得出用户的下列实际要求：

1.2 功能需求：

本项目是一个开源交流社区，为同学们提供了一个可以畅所欲言的平台，大家可以交流学习，分享身边的趣事，吐槽生活，交到志同道合的朋友。旨在拉近北航同学之间的距离，丰富大家的生活。为了更加满足用户的使用体验感，我们将提供以下功能。

用户个人信息页面：

包括用户自己设定的个性名称，系统随机为用户配置的个性头像；注册用户需要用户的手机号码；同时用户还可以为自己打上标签，我们会将用户常发帖子的的标签进行运算，选出最能表达用户特点的几个标签放到用户的个人主页，大家可以根据标签结识志同道合的朋友。

用户还可以多多发帖获得更多的粉丝，也可以关注别的用户成为别人的粉丝。

同时我们保证用户的个人信息除了用户名和标签都是隐藏的。

广场功能：

我们将会为社区设定一个“广场”，会将所有的最新动态都放在广场当中，根据时间实时更新，会显示每一条帖子的简要信息，包括标题、发布时间、以及帖子标签等。用户可在快乐的在广场中刷帖子。

发帖功能：

用户可以通过图文并茂的方式，将自己的心情、对生活的分享、需要的帮助发布到我们的社区广场中，广大用户将会在广场中看到大家的动态，帖子包含发布时间、标题、内容以及配图，同时用户在发布的时候可以为帖子打上标签，这样可以展示出帖子的内容类型。

帖子评论功能：

用户可以在广场随机刷帖，看到喜爱的标签或者感兴趣的标题就可以点击这条帖子查看详细信息，用户可以对喜欢的帖子作出评论，和广大网友一同进行交流这条帖子。

帖子搜索功能：

用户可以在广场的搜索栏中输入喜爱的标题，搜索相关内容的帖子，只阅读自己喜爱的话题。

帖子点赞功能：

用户如果在广场中刷到了喜欢的帖子，或者是和自己的观点相同的帖子，那么用户可以对该帖子进行点赞。

帖子踩功能：

用户如果在广场中刷到了不喜欢的帖子，或者是和自己的观点相悖的帖子，那么用户可以对该帖子进行踩。

用户之间交互功能（留言板）：

如果用户对其他用户很有兴趣，那么我们将提供留言板功能，该用户可以将想说的话留在对方主页，对方登陆后将会看到留言版中的新增内容。或者对喜欢的用户可以在对方主页点击关注，这样就可以每次看到对方最新的动态，成为对方的粉丝。

1.3 系统需求：

A 信息需求：

我们保证了用户的个人信息的绝对隐私，也是为了用户能够在我们的社区中畅所欲言；对其他用户来说，社区同学们能看到他或她的个性用户名以及常用标签，也能看到该用户已经发布的帖子；对该用户来说，他或她也能看到别人的相关信息，以及广场中的所有帖子。

B 处理需求：

用户可以发布表达此时此刻心情的帖子，内容完全由自己编辑，同时如果用户在某一天想要删除该帖子，用户个已在个人主页选择想要删除的帖子。用户忘记密码可以联系管理员进行密码找回。

C 安全性与完整性需求

安全性要求：

系统会设置访问用户的标识来鉴别该访问用户是否合法，合法用户将会被设置密码。保证用户身份不会被盗；

系统会对不同的数据设置不同的访问级别，限制访问的用户可以查询和处理数据的类别以及内容；

完整性要求

系统保证各种录入信息必须保证信息记录的完整性，关键信息内容不能为空；

各种数据之间相互联系正确性保证；

相同的数据在不同记录中的一致性保证。

D 数据标签需求

用户可以通过检索功能检索想要留言的帖子，实现了用户对数据的掌握和管理

E 可靠性需求

网络系统的稳定可靠是应用系统正常运行的关键，在网络设计中我们选取了优质的网络服务端口，本地数据库设备充分考虑冗余、容错能力；我们还需要合理设计网络架构，指定可靠的网络备份策略，保证网络具有故障后的自愈能力，需要最大限度地支持系统的正常运行。网络设备再出现故障时应便于诊断和排除。

F 性能需求

性能需求的实现依赖于两部分技术，第一点是服务器网络设备的良好运行，提高用户端的交互响应速度，保证远端数据的高质量传输；第二点是后端存取，以及数据库数据管理当中数据的物理结构分配，我们将使用Mysql数据库管理结构，对数据进行磁盘自动化管理，并且在设计后端逻辑时使用高效的算法，减少数据库调用查询次数，提高访存效率。

G 技术先进性与实用性

采用先进成熟的系统体系，使用成熟的概念、技术和方法，能够支撑各种现在与未来一段时期的主流应用网络，而且使我们的平台具有为未来发展潜力；可以按照模块化、层次化的原则设计网络，网络具有良好的伸缩性，根据雾浮起端网络配置进行灵活扩展，当前采取最新的Django后台管理框架，前端采取最新的Vue框架，保证了平台设计体系的先进，同时我们需要保证平台平稳运行，并且在适当的时间根据用户的体验需求的增加，在原有平台的基础上进行增量开发。

1.4 系统功能设计：

系统架构设计：

我们的项目业主要采用了三层架构(3-tier architecture)的系统结构，也可以理解为传统的MVC设计模式，就是将整体的业务应用划分为了三个层次：界面层（User Interface layer）、业务逻辑层（Business Logic Layer）、数据访问层（Data access layer），区分层析的目的就是为了在大型系统当中实现“高内聚低耦合”，保证了系统的结构优良，设计维护的便利以及迭代开发的扩展性。三个层次当中，系统主要功能和业务逻辑都在业务逻辑层进行处理。

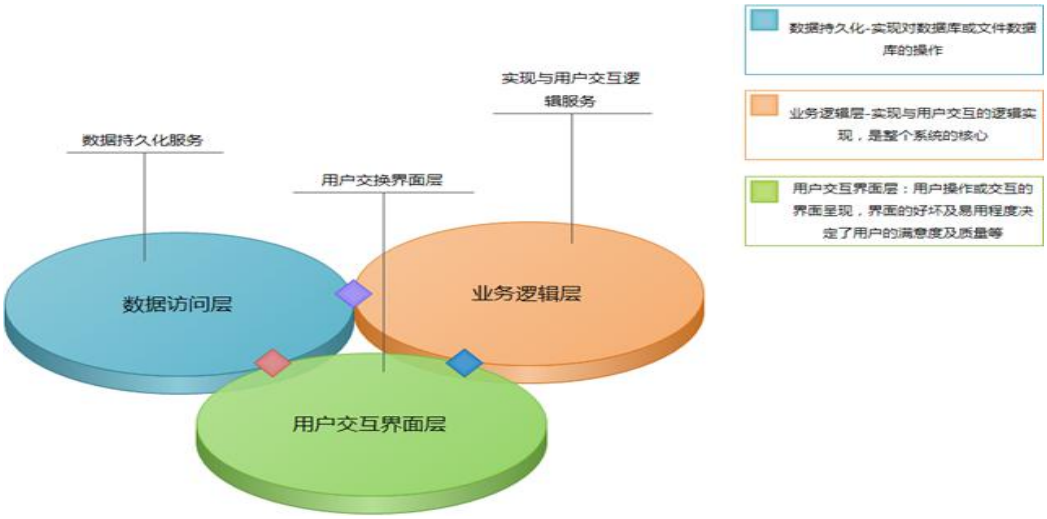


图1-1 三层架构图

分层设计的原则：

（1）层意味着组建的逻辑分组。例如，对用户界面，业务逻辑和数据访问组建应该使用不同的不同的层。

（2）在一个层内组建应该聚合的。如业务层组建仅应提供与业务逻辑相关的操作，而不是提供其他操作。

（3）对于Web应用程序，在表示层和业务逻辑层之间实现基于消息的接口是一个好主意，即使这两层没有跨越物理边界。基于消息的接口更适合于无状态的Web操作。

数据访问层：

有时候也称为是持久层，其功能主要是负责数据库的访问，可以访问数据库系统、二进制文件、文本文档或是XML文档。简单的说法就是实现对数据表的Select，Insert，Update，Delete的操作。如果要加入ORM的元素，那么就会包括对象和数据表之间的mapping，以及对象实体的持久化。

业务逻辑层：

主要是针对具体的问题的操作，也可以理解成对数据层的操作,对数据业务逻辑处理，如果说数据层是积木，那逻辑层就是对这些积木的搭建。业务逻辑层（Business Logic Layer）无疑是系统架构中体现核心价值的部分。它的关注点主要集中在业务规则的制定、业务流程的实现等与业务需求有关的系统设计，也即是说它是与系统所应对的领域（Domain）逻辑有关，很多时候，也将

业务逻辑层称为领域层。

视图层：

位于最外层（最上层），离用户最近。用于显示数据和接收用户输入的数据，为用户提供一种交互式操作的界面。主要表示WEB方式,也可以表示成WINFORM方式,WEB方式也可以表现成:aspx, 如果逻辑层相当强大和完善,无论表现层如何定义和更改,逻辑层都能完善地提供服务。

系统架构体系：

综合上述设计思想以及逻辑架构设计思路，我们可以给出整体的系统架构图。

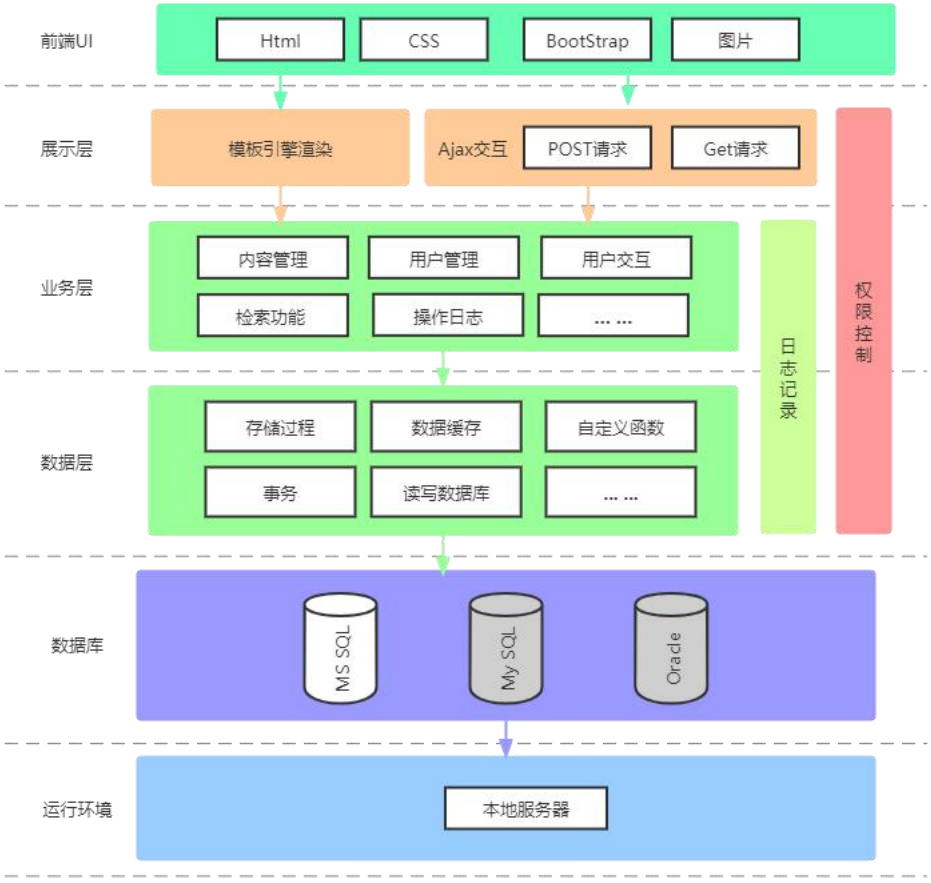


图1-2 系统架构图

根据上述的用户需求以及系统架构分析，我们将本系统按照功能进行了详细的分类。我们对我们的社区当中系统功能进行了大体的划分，主要划

分为了7个板块，集中精力提升用户体验，所有功能都是为用户服务的。而在每个板块之下又包含不同的操作，从两个层级进行了系统功能的划分，主要包括：板块功能、详细操作。

详见下方功能划分图。

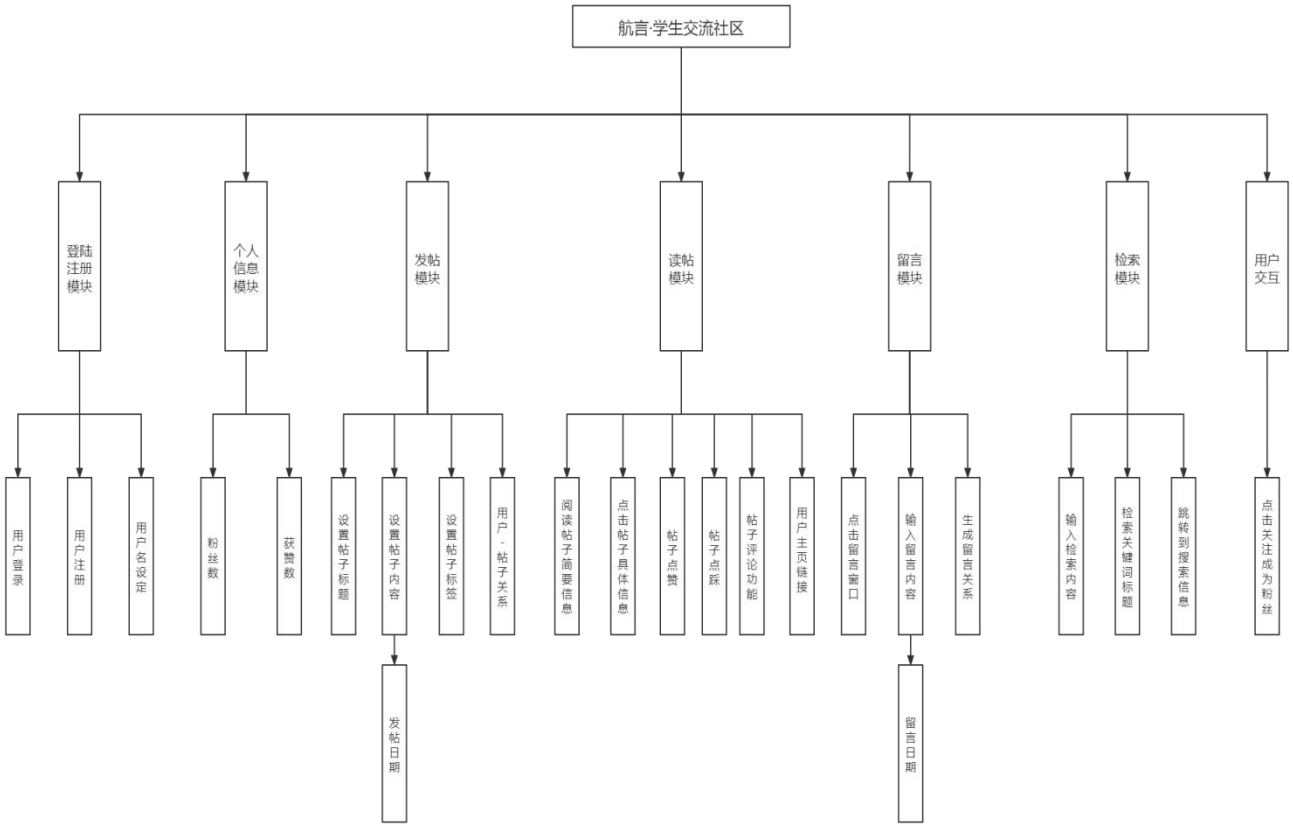


图1-3 系统功能模块划分图

用户功能：

- 处理用户登录
- 用户名的个性设定
- 用户注册
- 用户发帖功能
- 用户发帖设置帖子标题
- 用户发帖设置帖子内容
- 用户发帖设置帖子展示图片
- 用户设置帖子标签
- 用户删帖功能
- 用户阅读帖子

- 用户阅读其他用户主页
- 用户给其他用户留言
- 用户对帖子进行评论
- 用户对贴子进行点赞
- 用户对帖子进行踩

2. 数据流图

登陆数据流图：

主要进行用户登陆以及注册过程中的数据流信息以及导向。

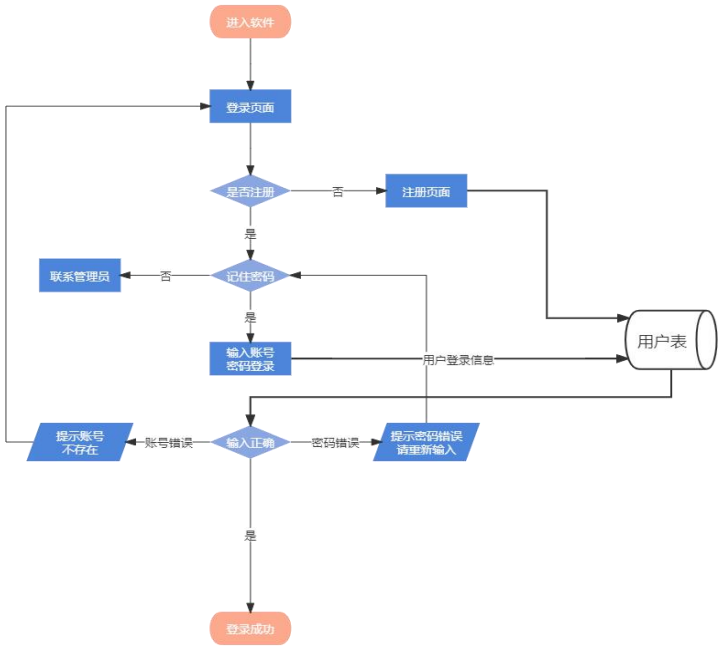


图2-1 登陆数据流图

用户个人信息流图：

主要进行用户登陆之后个人主页信息，以及查看其他用户个人主页信息的数据流i想你洗以及导向。

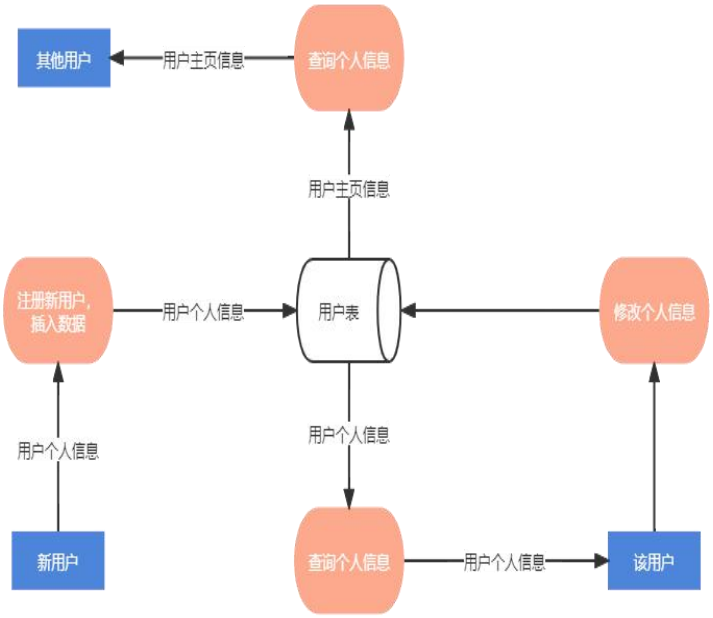


图2-2 用户个人信息数据流图

帖子信息流图：

主要进行用户发帖、删帖、修改内容等功能的数据流信息，也包括用户在广场刷帖子时观看的帖子信息。

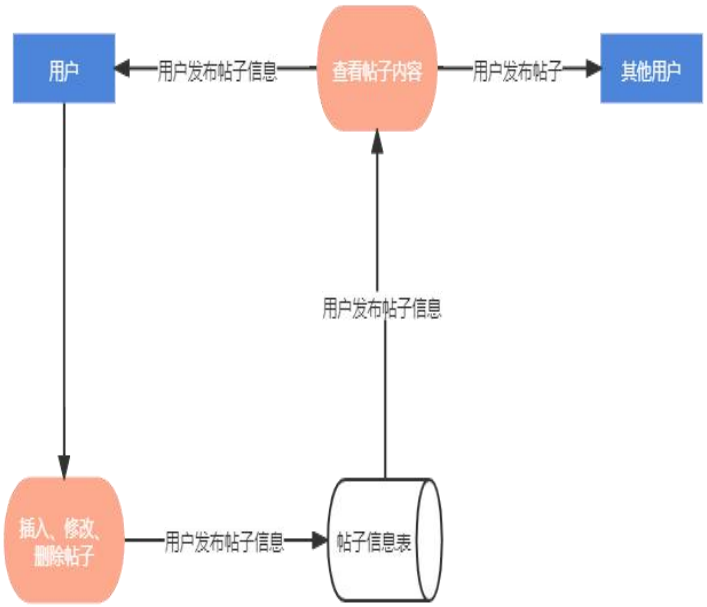


图2-3 帖子信息数据流图

评论信息流图：

当用户对相关帖子进行浏览时，可以发布评论，这一系统功能的数据信息流图如下。

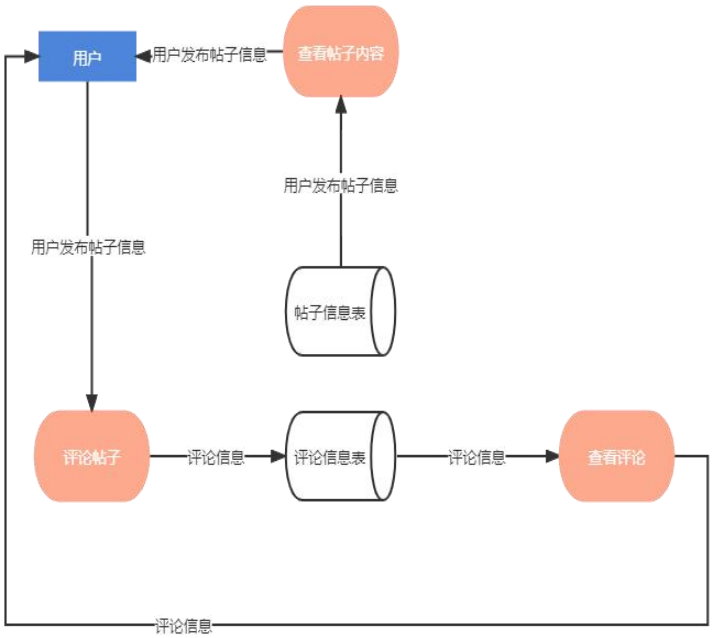


图2-4 评论信息数据流图

留言信息流图：

当用户对相关帖子进行浏览时，可以点击用户信息进入用户主页，之后可以点击留言板输入内容给用户留言，这一系统功能的数据信息流图如下。

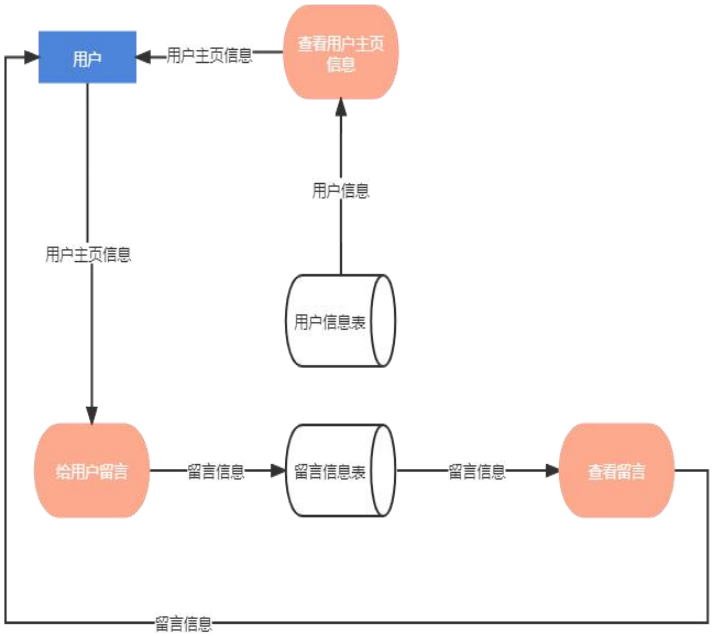


图2-5 留言信息数据流图

点赞信息流图：

当用户对相关帖子进行浏览时，可以对喜爱的帖子进行点赞，这一系统功能的数据信息流图如下。

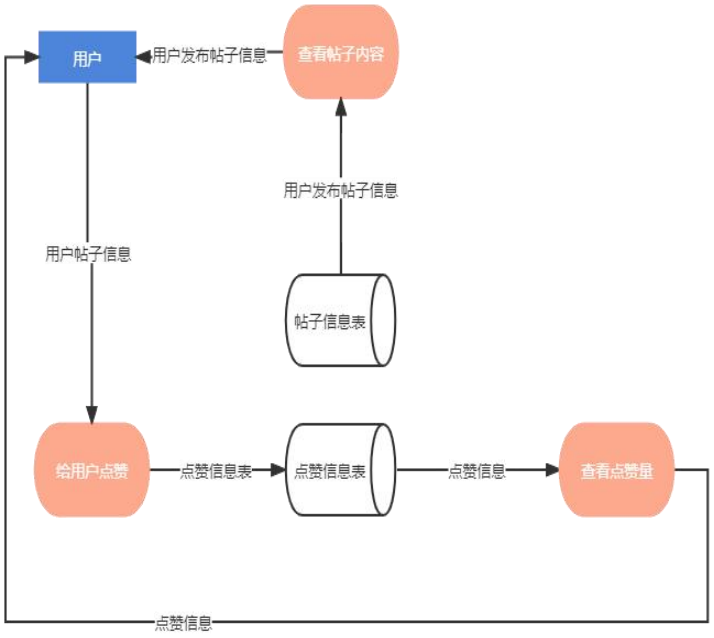


图2-6 点赞信息数据流图

关注信息流图：

当用户对相关帖子进行浏览时，可以点击用户信息进入用户主页，之后可以点击关注该用户，这一系统功能的数据信息流图如下。

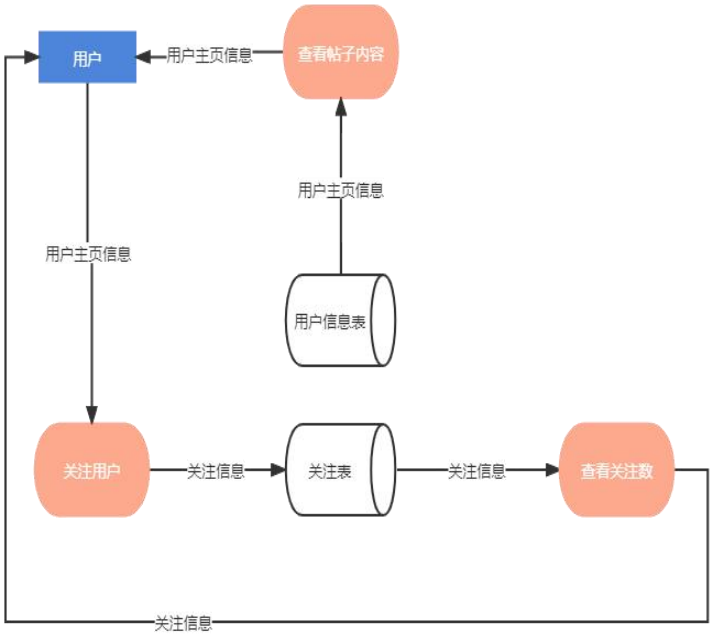


图2-7 关注信息数据流图

检索信息流图：

当用户想要通过关键词搜索帖子的时候，可以通过搜索栏目搜索符合条件的帖子，该行为数据流图如下所示。

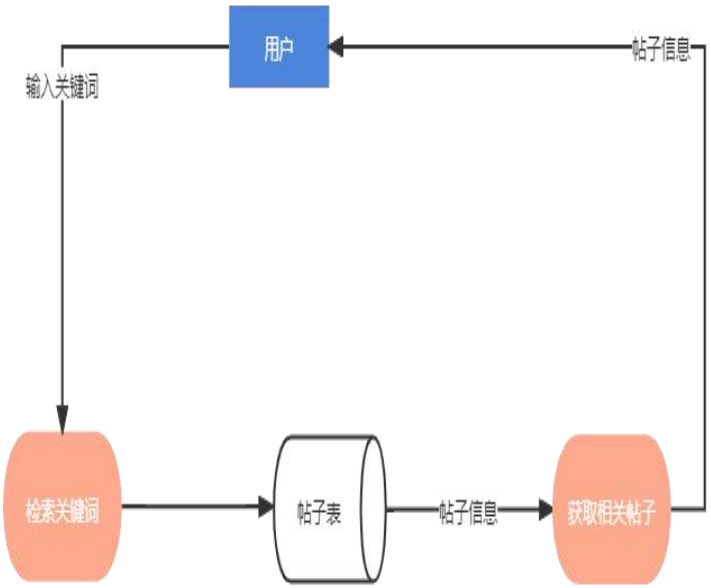


图2-7 检索信息数据流图

3. 数据元素表

用户信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
用户ID	id	int		每一个用户的独有ID	0-+∞
用户名	user_name	varchar	255	用户注册时设置的用户名	10个字以内的中文
用户手机号	user_phone	varchar	255	用户唯一的注册手机号	11位电话号码
用户密码	password	varchar	255	用户设定唯一密码	8位以上密码

粉丝数	follow_num	int		用户的粉丝个数	整数
关注数	star_num	int		用户自己关注的用户个数	整数
已发布帖子数	post_num	int		用户已发布的尚未删除的帖子	整数
头像	picture	varchar	255	用户上传的独有头像	字符串地址

帖子信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
帖子号	id	int		每条帖子生成后唯一的编号	0-+∞
标题	post_title	varchar	255	用户位帖子设定的标题	20字以内中文
内容	content	varchar	10000	用户帖子编辑的内容	500字以内中文
获赞	like_number	int		这条帖子获得的赞的个数	0-+∞
获踩	unlike_number	int		这条帖子被踩的个数	0-+∞
评论数	comment_num	int		这条帖子被评论的个数	0-+∞

	ber				
用户ID	user_id	int		发布者的ID	0-+∞
发布日期	post_date	datetime	6	帖子的发布日期	标准日期精确到分钟
标签词	keyword_name	varchar	255	描述帖子板块的标签词	0~128字符

点赞信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
点赞ID	id	int		一条点赞信息的ID	0-+∞
帖子ID	post_id	int		被点赞的帖子	0-+∞
用户ID	user_id	int		点赞的用户	0-+∞

踩帖信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
踩帖ID	id	int		一条点赞信息的ID	0-+∞
帖子ID	post_id	int		被点赞的帖子	0-+∞

用户ID	user_id	int		点赞的用户	0-+∞

评论信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
评论ID	id	int		一条评论的ID	0-+∞
帖子ID	post_id	int		被评论的帖子	0-+∞
用户ID	user_id	int		评论的用户	0-+∞
评论内容	content	varchar	10000	用户发布评论的内容	50字以内中英混合
评论时间	comment_date	datetime	6	发布评论的时间	标准日期精确到分钟

标签信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
标签ID	id	int		一个标签的ID	0-+∞
标签名	keyword_name	varchar	255	描述帖子板块的标签词	0~128字符

粉丝关注信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
关注ID	id	int		关注信息的ID	0-+∞
关注者D	follow_id	int		关注发起者的ID	0-+∞
被关注者ID	star_id	int		被关注用户的ID	0-+∞

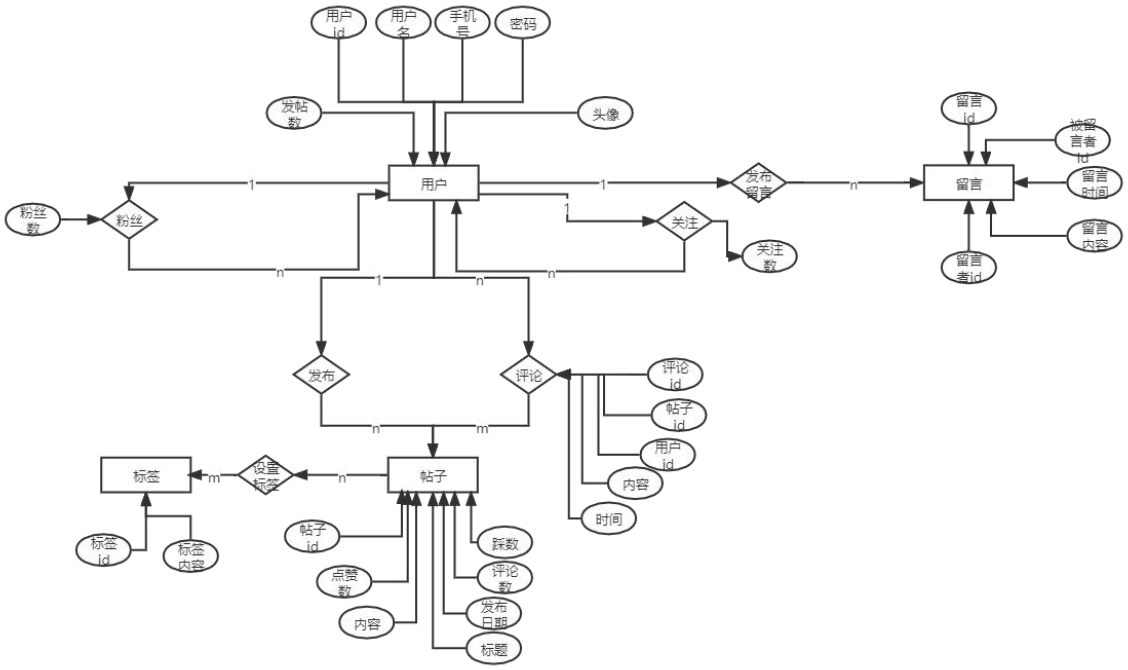
留言表信息

属性名称	字段代码	数据类型	长度	备注	取值和含义
留言ID	id	int		留言信息的ID	0-+∞
留言者ID	send_user_id	int		留言者的ID	0-+∞
被留言者ID	receive_user_id	int		被留言者的ID	0-+∞
留言内容	content	varchar	10000	留言信息当中的内容	50字以内中英混合
留言日期	message_date	datetime	6	发布留言的日期	标准日期精确

二．数据库概念模式设计

1. 系统初步E-R 图

根据上述需求分析、系统架构设计、系统功能分析以及元素表设计，我们设计了平台数据的初步ER图。



2. 系统基本 E-R 图

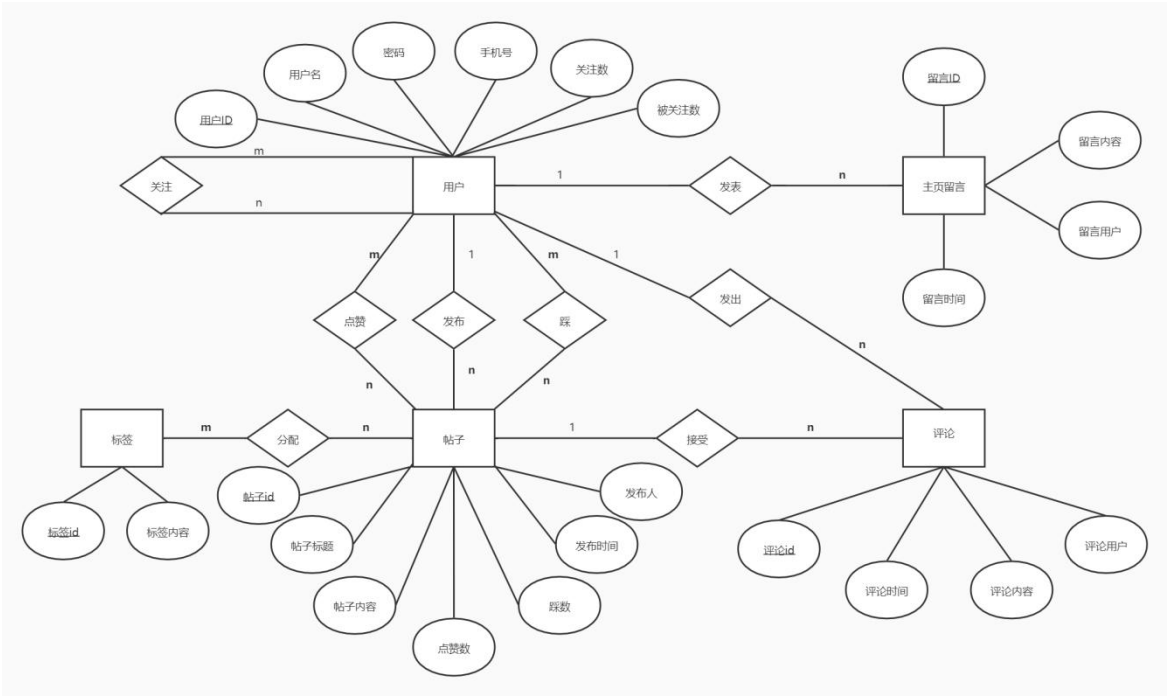
点赞、踩关系的属性（点赞数/踩数）归于帖子属性中，否则点赞/踩关系存在点赞数/踩数对帖子id的多值依赖。

关注关系的属性（关注数）归于用户属性，否则关注关系存在关注人数对用户

id的多值依赖。

被关注关系和关注关系其实属于等价关系，删去一个。

评论可以作为用户和帖子的关系，但由于评论自身有若干属性，若作为关系，则存在评论属性对帖子id的多值依赖。

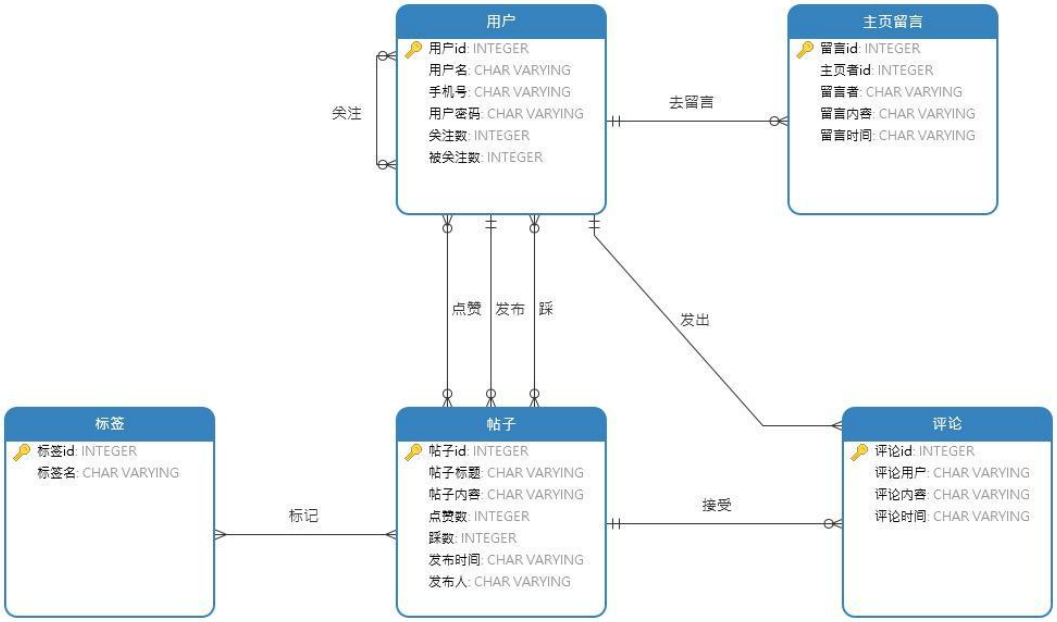


三、数据库逻辑模式设计

1. 数据库关系模式

注：由 E-R 图得到的关系模式

将实体转为关系模式：



将实体转为关系模式：

关注关系的属性归于用户属性，点赞和踩关系的属性归于帖子属性，发布和留言关系均没有属性。

因此关系的行均为二元组。

关注（用户id，用户id）。

去留言（用户id，留言的用户id）。

点赞（用户id，帖子id）。

踩（用户id，帖子id）。

发布帖子（用户id，帖子id）。

标记（标签id，帖子id）。

发出评论（评论id，用户id）。

接受评论（帖子id，评论id）。

2. 关系模式范式等级的判定与规范化

注：要规范到3NF

- 用户表中主码是用户id，其余属性直接函数依赖于用户id。满足3NF。
- 主页留言的主码是留言id，其余属性直接依赖于留言id。满足3NF。
- 标签的主码是标签id，其余属性直接函数依赖于标签id。满足3N。
- 评论的主码是评论id，其余属性直接函数依赖于评论id。满足3NF。
- 关系的关系模式中，任意模式的主码为全属性。满足3NF。

故该关系模型为3NF。

3. 数据库设计优化

系统功能主要是：用户登录、用户注册、文章列表显示、发布文章、个人文章显示、个人主页展示、关注用户、留言栏展示、留言选项、文章详情展示、评论及其展示、点赞和踩文章。

留言和留言栏的显示和操作需要的查询信息相同，故是同一实体，且和其他实体之间相对独立，无需优化。

数据库逻辑模式设计优化主要包括水平分解和垂直分解。

先分析垂直分解：

用户主页的显示包括用户名、关注数、被关注数、手机号、发布文章数，用户登录需要用到手机号和密码，用户注册需要用到用户名、手机号、密码和性别。下对用户表进行垂直分解：

登陆注册相关的表仅在登陆注册时查找，而进入主页后少有使用，因此可从原

用户表中抽离出来，单独成基本信息表。减小内存交换块的次数。

用户主页的显示则在进入他人主页时需要查询信息，将手机号、关注、文章数等信息抽离出来作为用户的主页信息、单独成主页信息表。这样虽然进行关注、发布文章等操作时会需要多修改一张表，但查找却不再需要从关注表、发布表中遍历求和，而只需要从主页信息表中查询合并即可。降低查询的时间复杂度。

分解之后两张表有公共的主键用户id，保有无损连接性；原用户表的函数依赖都是依赖于主键，而分解后都保有主键，故保有函数依赖。

文章列表是一个文章项的列表，每一个文章项包括标题、日期、作者、分类、摘要；文章详情包括文章的具体内容包括标题、日期、作者、内容、点赞数、踩数、评论。评论已经单独抽离出来成为实体，故不考虑优化。下对帖子表进行垂直分解：

文章列表的显示是一个查询操作，可将标题、日期、作者、分类、摘要从原帖子表中抽离出来、单独成文章概览表。这样做可将文章列表不需要查询的属性从表中去除，查询可直接从表中获得，减小内存交换块的次数。

文章详情的显示是一个查询操作可将点赞数、踩数、内容从原帖子表中抽离出来、单独成文章详情表。这样做虽然在点赞、踩时需要同时更新文章详情表，但在获得点赞数和踩数时不需要在点赞表、踩表中遍历求和，而可以直接从文章详情表中获得。另外考虑到文章详情显示之前一定会出现文章列表页面，故文章详情页面的标题、作者、时间等属性不需要再次查询数据库，前端进行参数传递即可。

分解之后两张表有公共的主键文章id，保有无损连接性；原帖子表的函数依赖都是依赖于主键，而分解后都保有主键，故保有函数依赖。

再分析水平分解：

查询个人发布帖子的事务操作的数据是个人发布的帖子，而文章列表的帖子显示的事务操作的数据是所有人发布的帖子。在文章列表整体数据很大时，查询个人发布的帖子会比较耗时，因此可将帖子表进行水平分解：个人发布的帖子和其他人发布的帖子。

由于一般情况下个人发布的帖子相比其他人发布的帖子体量很小，在对文章列表查询时多查询一次个人帖子表对性能影响不大，但是在对个人文章列表查询时查询个人帖子表效率会高很多，综合来看，提高了性能。