

# 数据库设计文档

选    题	<u>宠悦：为宠物爱好者提供的一站式信息与资源交流天地</u>
学    院	<u>软件学院</u>
专    业	<u>软件工程</u>
组    长	<u>林继申</u>
学    号	<u>2250758</u>
指导教师	<u>袁时金</u>
日    期	<u>2024 年 9 月 7 日</u>

## 宠悦 | PetJoy 项目开发组

姓名	学号	GitHub 主页	电子邮箱
林继申 (组长)	2250758	MinmusLin	2250758@tongji.edu.cn
钟承哲	2153495	FDSAIUOGH	2153495@tongji.edu.cn
李明哲	2250931	jaychou729	2250931@tongji.edu.cn
杜天乐	2251310	lleell04	2251310@tongji.edu.cn
刘淑仪	2251730	bunnyoi	2251730@tongji.edu.cn
梁哲旭	2252432	wangzai200	2252432@tongji.edu.cn
赵思源	2252444	aaa705	2252444@tongji.edu.cn
杨宇琨	2252843	YangRuakaka	2252843@tongji.edu.cn
刘垚	2253215	yaoyaolove	2253215@tongji.edu.cn
王沫涵	2253333	WCSCHM	2253333@tongji.edu.cn

GitHub 仓库地址：<https://github.com/MinmusLin/PetJoy>

## 目 录

1	数据库设计概述 .....	1
1.1	系统数据需求分析 .....	1
1.2	数据库概念设计 .....	1
1.3	数据库逻辑设计 .....	1
2	系统数据需求分析 .....	2
2.1	有关用户表的功能数据需求 .....	2
2.2	有关用户设置表的功能数据需求 .....	2
2.3	有关用户打卡表的功能数据需求 .....	3
2.4	有关用户关注表的功能数据需求 .....	3
2.5	有关用户留言表的功能数据需求 .....	4
2.6	有关用户反馈表的功能数据需求 .....	4
2.7	有关帖子表和帖子分类表的功能数据需求 .....	5
2.8	有关帖子评论表的功能数据需求 .....	5
2.9	有关帖子/帖子评论点赞/点踩表的功能数据需求 .....	6
2.10	有关帖子收藏表的功能数据需求 .....	7
2.11	有关帖子/帖子评论举报表的功能数据需求 .....	7
2.12	有关新闻表和新闻标签表的功能数据需求 .....	8
2.13	有关新闻评论表的功能数据需求 .....	9
2.14	有关新闻/新闻评论点赞/点踩表的功能数据需求 .....	9
2.15	有关新闻收藏表的功能数据需求 .....	10
2.16	有关新闻评论举报表的功能数据需求 .....	11
2.17	有关宠物分类/子类表的功能数据需求 .....	11
2.18	有关宠物护理指导表的功能数据需求 .....	12
2.19	有关宠物领养表的功能数据需求 .....	13
2.20	有关开发团队表的功能数据需求 .....	13
3	数据库概念设计 .....	14
3.1	数据库实体及相关属性 .....	14
3.1.1	用户模块 .....	14
3.1.2	宠物社区模块 .....	14
3.1.3	宠物新闻模块 .....	15
3.1.4	宠物百科与领养模块 .....	15
3.1.5	开发团队模块 .....	16
3.2	数据库总体 E-R 图 .....	16

装  
订  
线

3.3 数据库模块 E-R 图 .....	16
3.3.1 用户模块 E-R 图 .....	16
3.3.2 宠物社区模块 E-R 图 .....	16
3.3.3 宠物新闻模块 E-R 图 .....	18
3.3.4 宠物百科与领养模块 E-R 图 .....	18
3.3.5 开发团队模块 E-R 图 .....	21
4 数据库逻辑设计 .....	23
4.1 数据库关系模式图 .....	23
4.1.1 模块化设计 .....	23
4.1.2 用户中心设计 .....	23
4.1.3 内容管理设计 .....	24
4.1.4 举报系统设计 .....	24
4.1.5 宠物领养模块设计 .....	24
4.1.6 开发团队信息设计 .....	24
4.2 表设计 .....	24
4.2.1 用户表 USER .....	27
4.2.2 用户设置表 USER_SETTING .....	27
4.2.3 用户打卡表 USER_CHECK_IN .....	28
4.2.4 用户关注表 USER_FOLLOW .....	29
4.2.5 用户留言表 USER_MESSAGE .....	29
4.2.6 用户反馈表 USER_FEEDBACK .....	29
4.2.7 帖子分类表 POST_CATEGORY .....	30
4.2.8 帖子表 POST .....	30
4.2.9 帖子评论表 POST_COMMENT .....	31
4.2.10 帖子点赞表 POST_LIKE .....	32
4.2.11 帖子点踩表 POST_DISLIKE .....	32
4.2.12 帖子评论点赞表 POST_COMMENT_LIKE .....	32
4.2.13 帖子评论点踩表 POST_COMMENT_DISLIKE .....	33
4.2.14 帖子收藏表 POST_FAVORITE .....	33
4.2.15 帖子举报表 POST_REPORT .....	34
4.2.16 帖子评论举报表 POST_COMMENT_REPORT .....	34
4.2.17 新闻标签表 NEWS_TAG .....	35
4.2.18 新闻表 NEWS .....	35
4.2.19 新闻评论表 NEWS_COMMENT .....	36

装  
订  
线

4.2.20 新闻点赞表 NEWS_LIKE .....	37
4.2.21 新闻点踩表 NEWS_DISLIKE .....	37
4.2.22 新闻评论点赞表 NEWS_COMMENT_LIKE .....	37
4.2.23 新闻评论点踩表 NEWS_COMMENT_DISLIKE .....	38
4.2.24 新闻收藏表 NEWS_FAVORITE .....	38
4.2.25 新闻评论举报表 NEWS_COMMENT_REPORT .....	38
4.2.26 宠物分类表 PET_CATEGORY .....	39
4.2.27 宠物子类表 PET_SUBCATEGORY .....	40
4.2.28 宠物护理指导表 PET_CARE_GUIDE .....	43
4.2.29 宠物领养表 PET_ADOPTION .....	44
4.2.30 开发团队表 DEVELOPMENT_TEAM .....	45
4.3 数据库设计 .....	45
4.3.1 规范化 .....	45
4.3.2 主键和索引 .....	46
4.3.3 外键 .....	47
4.3.4 数据类型和约束 .....	47
4.3.5 安全性和隐私 .....	48
4.3.6 性能优化 .....	48
4.3.7 数据持久化 .....	49
4.3.8 扩展性和可维护性 .....	56
4.4 数据定义语言 (DDL) 设计 .....	56
4.4.1 用户表 USER .....	56
4.4.2 用户设置表 USER_SETTING .....	57
4.4.3 用户打卡表 USER_CHECK_IN .....	58
4.4.4 用户关注表 USER_FOLLOW .....	58
4.4.5 用户留言表 USER_MESSAGE .....	59
4.4.6 用户反馈表 USER_FEEDBACK .....	59
4.4.7 帖子分类表 POST_CATEGORY .....	60
4.4.8 帖子表 POST .....	60
4.4.9 帖子评论表 POST_COMMENT .....	61
4.4.10 帖子点赞表 POST_LIKE .....	62
4.4.11 帖子点踩表 POST_DISLIKE .....	62
4.4.12 帖子评论点赞表 POST_COMMENT_LIKE .....	62
4.4.13 帖子评论点踩表 POST_COMMENT_DISLIKE .....	63

装 订 线

4.5.17 向帖子收藏表插入记录: 增加收藏数和被收藏数 .....	84
4.5.18 从帖子收藏表删除记录: 减少收藏数和被收藏数 .....	85
4.5.19 向新闻评论表插入记录: 增加评论数 .....	85
4.5.20 从新闻评论表删除记录: 减少评论数 .....	86
4.5.21 向新闻点赞表插入记录: 增加点赞数和被点赞数 .....	86
4.5.22 从新闻点赞表删除记录: 减少点赞数和被点赞数 .....	87
4.5.23 向新闻点踩表插入记录: 增加点踩数 .....	87
4.5.24 从新闻点踩表删除记录: 减少点踩数 .....	87
4.5.25 向新闻评论点赞表插入记录: 增加点赞数和被点赞数 .....	88
4.5.26 从新闻评论点赞表删除记录: 减少点赞数和被点赞数 .....	88
4.5.27 向新闻评论点踩表插入记录: 增加点踩数 .....	89
4.5.28 从新闻评论点踩表删除记录: 减少点踩数 .....	89
4.5.29 向新闻收藏表插入记录: 增加收藏数和被收藏数 .....	90
4.5.30 从新闻收藏表删除记录: 减少收藏数和被收藏数 .....	90
4.5.31 用户注册时向用户设置表插入记录 .....	91

## 1 数据库设计概述

在本节（章节 1）中，我们将对数据库设计进行概述。

### 1.1 系统数据需求分析

在章节 2 中，我们将对系统数据需求进行分析。

在需求分析阶段，我们将详细讨论如何收集和分析用户需求以及业务规则。这是设计数据库前的首要步骤，目的是确定所需数据的类型、范围及其相互关系。这一阶段的输出是一个详尽的需求说明书，为后续的概念设计奠定基础。

### 1.2 数据库概念设计

在章节 3 中，我们将介绍数据库实体及相关属性、数据库总体 E-R 图和数据库模块 E-R 图。

概念设计阶段关注于根据需求分析结果建立数据库的高层结构。这一阶段的主要任务是定义数据模型，通常使用实体-关系模型（E-R 图）来表示。我们将详细描述数据库中的实体、属性、关系及其约束，形成一个完整的 E-R 图，覆盖数据库的整体结构及各个模块。

### 1.3 数据库逻辑设计

在章节 4 中，我们将介绍数据库逻辑设计，包括数据库关系模式图、表设计、数据库设计、数据定义语言（DDL）设计和触发器设计。

在逻辑设计阶段，将概念设计转化为逻辑模型。这一阶段主要包括定义关系模式、正规化过程以避免数据冗余、以及设计逻辑表结构。逻辑设计的结果是一系列详细的关系模式图和数据表设计，以及相应的数据定义语言（DDL）代码，用于实际建立数据库。

在数据库逻辑设计中，触发器设计也是一个关键部分，它用于在特定事件发生时自动执行预定义的操作。触发器可以用于维护数据完整性、实现复杂的业务规则、自动生成审计日志等。我们在插入、更新或删除操作上定义触发器，以确保数据库状态的一致性和可靠性。



## 2 系统数据需求分析

在本节（章节 2）中，我们将对系统数据需求进行分析。

### 2.1 有关用户表的功能数据需求

用户表是管理用户账户的核心数据结构，承担着重要的身份识别和账户管理功能。通过存储用户的唯一标识符、用户名、密码哈希等基本信息，用户表可以有效地进行身份验证和账户管理。为了保护账户安全，密码存储采用了加密哈希技术，防止密码被直接读取或破解。此外，用户表还包含了手机号码等联系方式，这些信息不仅用于账户恢复，还为多因素验证提供了基础。

用户表中的注册日期和上次登录日期字段记录了用户账户的创建和使用情况，有助于跟踪用户的活跃度和账户历史。用户角色字段通过数字标识不同的权限级别，如管理员或普通用户，从而实现平台的权限管理，确保不同角色的用户只能访问和操作其权限范围内的功能和数据。用户状态字段用于管理账户的访问权限，如正常、封禁等状态，通过这种方式可以有效地控制用户的行为，维护平台的安全和秩序。

除了基本的账户信息，用户表还包含了一些用户的个人资料信息，如头像链接、个人简介、性别和出生日期等。这些信息不仅增强了用户账户的个性化，也为平台的社交功能提供了支持。例如，用户的经验点数、关注数、被关注数、收藏数、被收藏数、点赞数、被点赞数和留言数等字段反映了用户在平台上的互动和活跃度，这些数据可以用于用户奖励机制、排行榜以及推荐系统等功能，激励用户积极参与平台活动。

### 2.2 有关用户设置表的功能数据需求

用户设置表通过提供个性化界面和隐私控制选项，赋予用户对其账户体验的更高控制权。

隐私控制是用户设置表的核心功能之一。用户可以设定哪些个人信息对外公开，哪些保持私密，包括手机号码、注册日期、个人简介、性别和出生日期等。这种细致的隐私控制增强了用户对其个人数据的掌控感，避免了不必要的隐私泄露。

除了基本的隐私设置，用户还可以选择是否公开其社交数据，如等级、关注数、被关注数、收藏数、被收藏数、点赞数、被点赞数和留言数等。这些设置允许用户根据个人需求决定是否展示其在平台上的活跃度和互动情况，保护用户的隐私，同时仍保留了一定的社交互动功能。

用户设置表还包括一些互动设置，如是否允许他人关注。通过这个选项，用户可以控制自己的社交关系，决定是否开放关注功能，从而保护自己的隐私和社交圈。

通过用户设置表，平台能够提供更加个性化和隐私友好的用户体验。用户不仅可以定制界面和隐私设置，还能根据个人需求调整通知接收和互动方式。这些功能共同作用，确保用户在平台上享有安全、舒适和受控的使用体验。

## 2.3 有关用户打卡表的功能数据需求

用户打卡表旨在通过记录用户的每日登录活动，激励用户更频繁地使用平台，从而提升整体参与度和用户粘性。每次用户登录平台时，系统都会在用户打卡表中创建一条记录，包含唯一的打卡记录 ID、用户 ID 和打卡时间。这些数据不仅能有效反映用户的活跃度和忠诚度，还为平台运营提供了有价值的用户行为分析基础。

记录用户的每日打卡数据可以帮助平台了解用户的使用习惯和活跃时间段。通过分析这些数据，平台可以优化功能和内容推荐，提升用户体验。例如，平台可以在用户最常登录的时间段内推出特别活动或内容，以吸引更多的用户参与和互动。

此外，用户打卡表的数据还可以用于制定和执行奖励机制。通过对连续打卡的用户给予积分奖励或其他形式的激励，平台可以鼓励用户保持日常活跃度，增加用户对平台的依赖性和忠诚度。这样的奖励机制不仅能提升用户的参与感，还能促进用户形成良好的使用习惯，从而长期保持高活跃度。

长期积累的打卡数据对于用户行为分析也是至关重要的。通过分析用户的打卡频率、打卡时间和持续时间，平台可以更好地了解用户的行为模式和偏好。这些分析结果可以为平台的改进提供数据支持，帮助平台更精准地满足用户需求，提升整体服务质量。

用户打卡表中的数据还可以用于检测和防止异常行为。比如，通过监控异常频繁的打卡记录，平台可以及时发现并防范潜在的恶意行为或账号滥用，确保平台的公平性和安全性。

## 2.4 有关用户关注表的功能数据需求

用户关注表用于记录和管理用户之间的关注关系，旨在提升用户的社交互动和平台的用户粘性。每次用户在平台上关注或取消关注其他用户时，系统会在用户关注表中创建或更新一条记录，其中包含被关注者 ID、关注者 ID 以及关注时间。通过记录这些数据，平台能够有效追踪用户的社交网络发展情况，并为用户推荐更符合其兴趣的内容和社交连接。

记录用户的关注关系数据有助于平台深入了解用户的社交行为和兴趣爱好，从而在内容推荐和社交互动方面进行优化。例如，平台可以根据用户的关注关系推荐更符合其兴趣的内容或活动，进一步提升用户的参与感。此外，通过分析关注关系，平台可以识别出活跃的用户群体，并有针对性地提供个性化服务或推送。

用户关注表的数据还可以用于提升用户对平台的粘性。通过分析用户的关注行为，平台可以设计并实施激励机制，鼓励用户进行更多的关注操作。例如，平台可以通过积分奖励系统，激励用户在一定时间内关注一定数量的其他用户，从而形成良好的社交互动习惯，提升用户活跃度。

长期积累的用户关注数据对用户行为的深度分析非常关键。平台可以通过分析用户的关注模式，了解不同用户的社交行为特点，并据此进行更精确的市场细分和用户画像。与此同时，平台可以利用这些数据检测异常行为，例如异常频繁的关注或取消关注操作，以防止潜在的恶意行为或账号滥用，确保平台的公平性和安全性。

## 2.5 有关用户留言表的功能数据需求

用户留言表提供了一个平台内部的直接消息交换方式，允许用户之间进行直接沟通，从而支持社区内的交流和信息分享。这种直接交流方式不仅可以促进用户之间的互动，还可以增强社区的合作氛围和支持感。通过用户留言表，用户可以进行私人交流，提出问题或参与讨论，从而构建一个更为活跃和紧密的社区环境。

每条留言记录都包含唯一的留言 ID、用户 ID、留言者 ID、留言内容和留言时间等信息。这些字段确保了每条留言的唯一性和可追溯性，同时也提供了必要的信息来进行数据分析和社区管理。通过记录留言时间，平台可以分析用户活跃时段，优化系统资源分配和社区管理策略。

用户留言表的设计旨在鼓励用户之间的互动。用户可以在平台上给其他用户留言，无论是提出问题、提供建议还是进行讨论，都可以通过这种方式进行。这种互动不仅有助于用户之间建立联系，也有助于构建一个支持性和合作性的社区氛围。用户留言表为用户提供了一个表达观点和分享信息的渠道，使社区变得更加动态和多元化。

此外，用户留言表的数据还可以用于社区管理和内容审核。平台管理者可以通过分析留言内容和互动频率，了解用户关注的话题和热点问题，从而更好地引导社区讨论和维护社区秩序。同时，留言内容也可以作为用户行为和偏好分析的重要数据来源，帮助平台优化内容推荐和用户体验。

## 2.6 有关用户反馈表的功能数据需求

用户反馈表为用户提供了一个有效的渠道，以提交对平台功能、内容和用户体验的意见和建议。通过这种反馈机制，用户可以报告遇到的问题、提出改进建议或表达对某些功能的满意度。这些反馈对于平台的持续改进和优化至关重要，有助于开发团队及时识别和解决问题，并了解用户的需求和期望。

用户反馈表包含多个关键字段，包括唯一的反馈意见 ID、反馈类型、反馈内容、反馈时间、电子邮箱和手机号码等。反馈类型字段允许用户分类其反馈内容，如功能问题、内容建议或用户体验评价，从而帮助平台更有针对性地处理不同类型的反馈。反馈内容字段提供了用户详细描述其意见和建议的空间，确保反馈信息的完整性和具体性。

通过记录反馈时间，平台可以追踪用户提交反馈的时间点，分析反馈的时效性和频率。这有助于平台识别出高峰期的反馈提交时间段，从而更好地分配资源，及时响应用户的需求。此外，电子邮箱和手机号码字段为平台与用户之间的进一步沟通提供了联系方式，使平台能够在需要时向用户寻求更多详细信息或反馈处理进度。

系统化地收集和分析用户反馈数据，平台可以识别出共性问题 and 用户普遍关心的功能或内容。这些数据不仅可以用于修复现有问题，还可以为平台的未来开发方向提供重要的参考依据。例如，如果大量用户反馈某一功能存在使用困难，开发团队可以优先考虑对该功能进行改进。同时，正面的反馈也能帮助平台确认哪些功能受用户欢迎，从而继续优化和推广这些功能。

通过用户反馈表，平台能够更好地理解用户的需求和期望，增强用户参与感和满意度。用户看

到其反馈得到重视并采取行动，会增强对平台的信任和忠诚度。定期分析和总结反馈数据，平台可以不断优化用户体验，提升整体服务质量，最终实现平台与用户的双赢局面。

## 2.7 有关帖子表和帖子分类表的功能数据需求

帖子表在社区交流中扮演着核心角色，不仅存储用户发表的内容，还记录了与帖子相关的互动数据，如点赞数、评论数和收藏数。这些数据是衡量内容受欢迎程度的重要指标，同时也是分析用户行为、优化内容推荐和搜索功能的基础。通过这些互动数据，平台能够更好地了解哪些内容更受用户欢迎，从而调整内容生成和推荐策略，提高用户满意度和参与度。

帖子表包含多个关键字段，包括唯一的帖子 ID、发帖用户 ID、标题、内容、创建时间、更新时间、是否置顶、点赞数、点踩数、收藏数和评论数等。这些字段确保了每个帖子的唯一性和完整性，并提供了丰富的信息来进行数据分析和管理工作。

记录帖子的创建时间和更新时间，有助于平台追踪内容的发布和修改历史。这不仅可以用于内容管理，还能帮助用户了解帖子的时效性和更新频率。置顶字段则用于标识重要或优质的帖子，使其在社区页面上得到优先展示，增加曝光度和阅读量。

点赞数、点踩数、收藏数和评论数等互动数据，是平台了解用户偏好和内容受欢迎程度的重要参考。这些数据不仅可以用于制定用户奖励机制，还能帮助平台识别出高质量或有争议的内容，从而进行更精准的内容推荐和搜索优化。例如，高点赞数和收藏数的帖子可以优先推荐给其他用户，而高评论数的帖子可能代表着用户讨论的热点话题。

通过对这些互动数据的分析，平台可以不断优化用户体验。了解用户偏好后，平台可以调整内容推荐策略，使用户更容易找到自己感兴趣的内容，增加停留时间和互动频率。这不仅提高了用户的满意度，还能增强社区的活跃度和黏性。

帖子表中的数据还可以用于用户行为分析，帮助平台了解用户的发帖习惯和互动模式。例如，分析哪些用户更活跃，哪些主题更受关注，可以为平台的内容策略和用户管理提供数据支持。这些分析结果可以帮助平台更好地引导用户参与，提升社区的整体质量和氛围。

帖子分类表是社区交流平台中对帖子进行有效管理和组织的重要工具。通过对帖子进行分类，用户能够更快地找到与自己兴趣相关的内容，同时也便于平台对帖子内容进行更精细的推荐和管理。

## 2.8 有关帖子评论表的功能数据需求

帖子评论表为用户提供了一个直接反馈和互动的平台，让用户可以对帖子内容进行评价和讨论，从而增加用户的参与度和帖子的可视性。评论作为社区互动的重要组成部分，不仅促进了内容的深入讨论和共享，还为评估内容质量提供了重要指标。

通过帖子评论表，用户可以针对某个帖子发表自己的看法，提出问题或补充信息，从而形成更丰富和多样化的讨论环境。每条评论记录包含唯一的评论 ID、关联的帖子 ID、评论用户 ID、评论

内容、评论时间、点赞数和点踩数等信息。这些字段确保了评论的唯一性和完整性，并为数据分析提供了必要的基础。

评论时间字段记录了每条评论的发表时间，有助于平台追踪用户的互动频率和高峰时段。这些数据可以用于优化平台的资源分配和活动安排，提高用户的使用体验和平台的运行效率。通过分析评论的时间分布，平台可以识别出用户活跃的时间段，针对性地推送内容或开展活动，增加用户的参与度。

评论的点赞数和点踩数字段是衡量评论质量和用户反馈的重要指标。高点赞数的评论通常代表着用户认为有价值的内容，而高点踩数的评论则可能反映了用户的负面反馈。通过分析这些互动数据，平台可以进一步优化内容推荐算法，将优质评论和内容优先展示给用户，提高内容的互动率和社区的活跃度。

此外，帖子评论表还支持嵌套评论功能，即评论可以有父评论。通过父评论 ID 字段，用户可以对其他用户的评论进行回复，形成多层次的讨论结构。这种结构不仅增加了讨论的深度和广度，还增强了用户之间的互动和联系，促进了社区的繁荣发展。

用户对帖子和评论的互动数据还可以用于平台的用户行为分析，帮助平台了解用户的兴趣和偏好。通过对评论内容和互动数据的系统分析，平台可以识别出用户关注的热点话题和优质内容，从而优化内容生成和推荐策略，提升用户满意度和平台的整体质量。

## 2.9 有关帖子/帖子评论点赞/点踩表的功能数据需求

点赞和点踩表记录了用户对帖子和评论的正面或负面反馈，提供了一种简单而直接的互动方式，使用户能快速表达对内容的看法。这些反馈不仅是内容创建者了解其内容受欢迎程度的重要数据来源，也是平台运营者优化内容质量和用户体验的关键依据。

通过点赞和点踩表，平台能够收集到用户对特定内容的即时反应。每条记录包含用户 ID、帖子或评论 ID 以及反馈类型（点赞或点踩）。这些数据为平台提供了具体的用户行为数据，使平台能够精确地分析哪些内容受到了用户的喜爱，哪些内容可能引起了用户的不满。这种分析有助于平台不断改进内容推荐算法，确保用户看到更多他们感兴趣的内容。

内容创建者可以通过查看其帖子或评论的点赞和点踩数量，了解用户对其内容的反馈，从而调整自己的内容创作策略。高点赞数表明内容受到了广泛的认可和喜爱，而高点踩数则提示创作者需要考虑改进内容质量或调整内容方向。这种直接的反馈机制不仅提高了内容创作者的创作动力，也促进了优质内容的产生。

对于平台运营者来说，点赞和点踩数据是优化内容质量的重要参考依据。通过系统化地收集和分析这些数据，平台可以识别出优质内容和问题内容。优质内容可以获得更多的推荐和曝光机会，而问题内容则可能需要进一步审核或改进，甚至在某些情况下，可能会被删除以确保平台内容的整体质量。

点赞和点踩数据还可以用于用户行为分析，帮助平台了解用户的兴趣和偏好。通过分析用户对

不同类型内容的反馈，平台可以更好地为用户推荐个性化内容，提升用户满意度和粘性。这种个性化推荐不仅增强了用户体验，还能增加用户的互动和参与，促进平台的整体活跃度。

## 2.10 有关帖子收藏表的功能数据需求

帖子收藏表为用户提供了一个便捷的方式来标记并保存他们感兴趣的内容，以便未来查看。这种收藏功能不仅提高了内容的再访问率，还通过分析这些数据，帮助平台获得关于用户兴趣的宝贵信息，从而推动个性化推荐和内容策略的调整。

用户在浏览平台时，可以通过收藏功能将感兴趣的帖子保存起来，形成自己的内容库。这种个人化的内容管理方式增强了用户的参与感，使用户能够更方便地找到之前感兴趣的内容，提升了用户体验和平台的使用粘性。

收藏表中的数据对于平台运营者来说，是分析用户兴趣的重要数据来源。每条收藏记录包含用户 ID 和帖子 ID 等信息，通过分析这些数据，平台可以了解哪些内容更受用户欢迎，从而优化内容推荐系统。例如，如果某些帖子被大量用户收藏，这些内容可能在质量、实用性或趣味性方面有突出的表现，平台可以优先推荐类似内容给其他用户。

此外，收藏功能还可以帮助平台识别用户的长期兴趣偏好。通过对用户收藏内容的分析，平台可以更精准地了解用户的兴趣变化和趋势，为用户提供更加个性化和有针对性的内容推荐。这种个性化的推荐不仅提高了用户的满意度，也增加了用户在平台上的停留时间和互动频率，促进了平台的整体活跃度。

收藏数据还可以用于内容创作者的反馈机制。内容创作者可以通过查看其帖子被收藏的次数，了解其内容在用户中的受欢迎程度，从而调整自己的创作策略。高收藏数表明内容受到了用户的高度认可和重视，创作者可以继续探索和发展类似的主题或风格，提升内容质量和吸引力。

平台还可以利用收藏数据来优化广告投放策略。通过了解用户收藏的内容类型，平台可以更精准地投放相关广告，提高广告的点击率和转化率。这不仅增加了平台的广告收益，也提升了用户对广告的接受度，形成良性循环。

## 2.11 有关帖子/帖子评论举报表的功能数据需求

帖子和帖子评论举报表为用户提供了一种机制，可以标记不当或不适宜的内容，维护社区的健康和安全。这个功能对于快速响应社区规范的违反情况至关重要，确保平台内容的质量和合规性。通过处理举报信息，运营团队能够及时采取措施，如删除不当内容或对违规用户进行处罚，从而保护社区用户不受有害内容的影响。

举报表记录了用户举报的详细信息，包括举报的帖子或评论 ID、举报用户 ID、被举报用户 ID、举报理由和举报时间等。这些数据使得平台能够系统化地管理和跟踪每个举报事件，确保所有举报都能得到及时和公正的处理。举报理由字段让用户可以详细说明举报的原因，有助于运营团队快速准确地判断内容是否违反了社区规范。

每条举报记录都包括一个唯一的举报 ID，用于识别和管理举报事件。举报的帖子或评论 ID 字段则明确指出了被举报的具体内容，帮助运营团队迅速定位问题所在。通过关联用户 ID，平台可以跟踪到举报者和被举报者的账户信息，从而进行进一步的调查和处理。

举报时间字段记录了每次举报的具体时间点，这对于追踪和管理举报事件的处理进度非常重要。运营团队可以根据举报时间的先后顺序处理事件，确保举报能够按照提交的时间顺序得到公平和及时的处理。

举报表中的数据还可以用于分析和优化平台的管理策略。例如，通过分析举报的频率和类型，平台可以识别出哪些类型的内容最容易引发争议或违反社区规范，从而采取预防措施，改进内容审核机制。对于频繁被举报的用户，平台可以进行重点监控或采取进一步的限制措施，防止其再次发布不当内容。

处理举报信息不仅是维护社区健康和安全的重要手段，也是提升用户信任和满意度的关键。用户看到他们的举报能够得到及时有效的处理，会增强对平台的信任感和归属感。同时，这也激励用户积极参与社区管理，共同维护一个友好和谐的社区环境。

## 2.12 有关新闻表和新闻标签表的功能数据需求

新闻表和新闻标签表共同构建了平台的新闻内容发布和管理系统，丰富了平台的内容类型，并为用户提供了教育、娱乐和信息资源。这些表的设计和函数需求不仅增强了内容的可发现性，还帮助平台吸引和保留用户，提高用户的参与度。

新闻标签表用于对新闻内容进行分类，提供了灵活的标签系统，使用户和系统能够根据主题或兴趣对新闻进行过滤和搜索。每个标签都有一个唯一的标签 ID 和标签名称，确保标签的唯一性和可辨识度。通过这种分类系统，平台可以更好地组织和展示新闻内容，提高新闻的可访问性和用户体验。

新闻表存储了关于新闻的详细信息，包括标题、摘要、内容、作者 ID、标签 ID、创建时间、更新时间、封面图片链接，以及用户互动数据如点赞数、点踩数、收藏数和评论数。新闻表的设计确保了每条新闻的唯一性和完整性，为新闻内容的发布和管理提供了坚实的基础。

每条新闻记录都包含一个唯一的新闻 ID，用于识别和管理新闻内容。用户 ID 字段关联到发帖的管理员，确保新闻内容的发布和审核由有权限的用户进行。标签 ID 字段则关联到新闻标签表，使得每条新闻都能通过标签进行分类和搜索，增强了新闻内容的组织和发现。

新闻的创建时间和更新时间字段记录了新闻的发布和修改历史，有助于平台跟踪新闻内容的时效性和变化。封面图片链接字段为新闻提供了视觉展示，提高了新闻内容的吸引力。置顶字段用于标识重要或优质的新闻，使其在新闻页面上得到优先展示，增加曝光度和阅读量。

用户互动数据如点赞数、点踩数、收藏数和评论数等，是衡量新闻内容受欢迎程度的重要指标。这些数据不仅可以用于评估新闻质量，还可以帮助平台优化内容推荐算法，确保用户看到更多他们感兴趣的新闻内容。高互动数据的新闻可以优先推荐给其他用户，而低互动数据的新闻则可能需要

进一步优化或调整。

## 2.13 有关新闻评论表的功能数据需求

新闻评论表为用户提供了一个平台，让他们可以对新闻内容进行反馈和讨论，从而增强了新闻内容的互动性和可视性。通过评论，用户可以分享观点、进行讨论或提供反馈，这不仅提高了用户的参与感，还增加了新闻内容的深度和广度。

新闻评论表包含多个关键字段，包括唯一的评论 ID、新闻 ID、用户 ID、父评论 ID、评论内容、评论时间、点赞数和点踩数等。这些字段确保了评论的唯一性和完整性，并为数据分析提供了必要的基础。

评论 ID 是每条评论的唯一标识符，用于区分和管理不同的评论。新闻 ID 字段则关联到特定的新闻条目，使得评论能够准确地与相应的新闻内容关联。用户 ID 字段记录了发表评论的用户信息，确保评论来源的可追溯性。

父评论 ID 字段允许评论形成嵌套结构，即用户可以对其他评论进行回复，形成多层次的讨论。这种嵌套评论功能不仅增加了讨论的深度和广度，还增强了用户之间的互动和联系，促进了社区的繁荣发展。

评论内容字段记录了用户的具体反馈和观点，评论时间字段记录了每条评论的发表时间，有助于平台追踪用户的互动频率和高峰时段。这些数据可以用于优化平台的资源分配和活动安排，提高用户的使用体验和平台的运行效率。

点赞数和点踩数字段是衡量评论质量和用户反馈的重要指标。高点赞数的评论通常代表着用户认为有价值的内容，而高点踩数的评论则可能反映了用户的负面反馈。通过分析这些互动数据，平台可以进一步优化内容推荐算法，将优质评论和内容优先展示给用户，提高内容的互动率和社区的活跃度。

新闻评论表中的数据还可以用于分析和优化平台的管理策略。例如，通过分析评论的频率和质量，平台可以识别出哪些新闻内容引发了更多的讨论和反馈，从而为内容编辑和策略调整提供指导。同时，评论数据也可以作为用户行为分析的重要数据来源，帮助平台了解用户的兴趣和偏好，优化内容推荐和用户体验。

## 2.14 有关新闻/新闻评论点赞/点踩表的功能数据需求

新闻及其评论的点赞和点踩表记录用户对新闻内容及其评论的正面或负面反馈，为用户提供了快速简便的方式来表达对内容的看法。这些互动数据不仅是评估内容受欢迎程度的重要指标，也是平台优化内容推荐和用户体验的关键数据源。

点赞和点踩表记录了每个点赞或点踩的具体信息，包括用户 ID、新闻或评论 ID、反馈类型（点赞或点踩）以及反馈时间。通过这些数据，平台可以了解用户对特定内容的即时反应，帮助内容创建者和平台运营者更好地评估内容质量。



对于内容创建者来说，点赞和点踩数据提供了直接的用户反馈。高点赞数表明内容受到用户的认可和喜爱，而高点踩数则可能提示内容存在不足之处，需要改进。这种反馈机制不仅激励内容创建者提升内容质量，还帮助他们调整创作方向，更好地满足用户需求。

平台运营者通过分析点赞和点踩数据，可以识别出哪些新闻和评论更受用户欢迎，从而优化内容推荐策略。受欢迎的内容可以得到更多的推荐和曝光机会，增加阅读量和互动率。而不受欢迎的内容则需要进一步审查和调整，甚至可能会被撤下，以确保平台内容的整体质量。

点赞和点踩数据还可以用于用户行为分析，帮助平台了解用户的兴趣和偏好。通过分析用户对不同类型内容的反馈，平台可以更精准地推荐用户感兴趣的内容，提升用户满意度和粘性。这种个性化推荐不仅增强了用户体验，还增加了用户在平台上的停留时间和互动频率，促进了平台的整体活跃度。

此外，点赞和点踩数据可以帮助平台进行内容审核和管理。对于频繁被点踩的内容，平台可以进行重点监控和审核，确保其符合社区规范和质量标准。对于频繁获得点赞的内容，平台可以加大推广力度，吸引更多用户关注和参与。

## 2.15 有关新闻收藏表的功能数据需求

新闻收藏表为用户提供了一种便捷的方式来保存他们感兴趣的新闻内容，以便日后查看。这种功能不仅增加了新闻内容的再访问概率，还通过分析用户的收藏行为，为平台提供了宝贵的数据，从而更好地了解用户的兴趣和偏好。

新闻收藏表记录了每次收藏的详细信息，包括用户 ID、新闻 ID 和收藏时间。通过这些数据，平台可以系统化地管理用户的收藏行为，并利用这些信息优化内容推荐和发布策略。用户 ID 字段关联到收藏行为的具体用户，新闻 ID 字段关联到被收藏的新闻内容，收藏时间字段则记录了用户进行收藏操作的具体时间点。

收藏功能增强了用户的参与感，使用户能够轻松地保存和管理自己感兴趣的新闻内容。这种人性化的内容管理方式提高了用户体验和平台的使用粘性，用户可以随时访问自己收藏的新闻，增加了内容的再访问率和用户的满意度。

平台通过分析用户的收藏数据，可以获得关于用户兴趣和偏好的宝贵信息。例如，某些新闻内容如果被大量用户收藏，说明该内容具有较高的吸引力和价值，平台可以据此调整内容发布策略，优先推送类似内容。通过了解用户的收藏行为，平台可以更精准地进行个性化内容推荐，确保用户看到更多他们感兴趣的新闻内容，提升用户的整体体验和满意度。

收藏数据还可以帮助平台识别出用户的长期兴趣偏好和行为模式。通过对收藏内容的分析，平台可以了解用户的兴趣变化和趋势，优化内容生成和发布策略。这种个性化的推荐和策略调整不仅提高了用户的满意度，还增强了用户在平台上的粘性和互动频率，促进了平台的整体活跃度。

此外，新闻收藏表的数据还可以用于广告投放策略的优化。通过了解用户收藏的新闻类型，平台可以更精准地投放相关广告，提高广告的点击率和转化率。这不仅增加了平台的广告收益，也提

升了用户对广告的接受度，形成良性循环。

## 2.16 有关新闻评论举报表的功能数据需求

新闻评论举报表为用户提供了一种机制，使他们能够标记不当或不适当的评论内容，从而维护社区的健康和安全。这一功能对于快速响应和处理社区规范的违反行为至关重要，有助于确保平台内容的质量和合规性。通过有效的举报处理，平台可以及时采取措施，保持社区环境的秩序和正面形象。

每条举报记录都包含一个唯一的新闻评论举报 ID，用于标识和管理不同的举报事件。举报者 ID 字段记录了提交举报的用户信息，被举报者 ID 字段则记录了被举报评论的作者信息。这些信息使得举报内容具有可追溯性，确保平台能够准确地识别和处理违规行为。

被举报评论所属新闻 ID 字段关联到被举报评论所属的具体新闻，有助于平台了解举报事件的背景和上下文。被举报评论 ID 字段直接关联到具体的评论，确保举报内容的精准定位和处理。举报原因字段让用户详细说明举报的原因，帮助平台快速判断内容是否违反了社区规范。

举报时间字段记录了每次举报的具体时间点，有助于平台追踪举报事件的处理进度。处理状态字段记录了每个举报事件的当前处理状态，如待处理、处理中或已处理等。这些信息使平台能够系统化地管理和跟踪每个举报事件，确保所有举报都能得到及时和公正的处理。

通过系统化地收集和分析举报数据，平台可以识别出常见的违规行为和高风险用户，从而采取预防措施，改进内容审核机制。这不仅有助于平台保持内容的高质量和合规性，还能保护社区用户不受有害内容的影响，提升用户的信任度和满意度。

处理举报信息也是提升用户参与感和社区健康的重要手段。用户看到他们的举报能够得到及时有效的处理，会增强对平台的信任感和归属感。同时，这也激励用户积极参与社区管理，共同维护一个友好和谐的社区环境。

举报数据还可以帮助平台进行用户行为分析和策略优化。例如，频繁被举报的用户可能需要进一步的监控或教育，确保他们遵守社区规范。对于频繁出现的举报原因，平台可以加强相关内容的审核力度，避免类似问题的再次发生。

## 2.17 有关宠物分类/子类表的功能数据需求

宠物分类和子类表是宠物百科部分的核心数据结构，提供了详尽的宠物品种信息和分类。宠物分类表定义了宠物的高级分类，如狗、猫等，通过这些高级分类，用户可以快速找到自己感兴趣的宠物种类。每个高级分类都有唯一的分类 ID 和名称，以及简要描述和图片链接，方便用户浏览和理解。

宠物子类表则进一步详细到各个具体品种的信息，包括起源、体型、毛色、寿命、性格和饮食习惯等。这些详细信息不仅帮助用户更好地了解不同的宠物品种，还为用户在选择和护理宠物时提供了重要的参考依据。例如，用户可以根据宠物的体型和寿命选择适合家庭环境和生活方式的宠物；

通过了解宠物的性格和饮食习惯，用户可以更好地准备和照顾新宠物。

此外，宠物子类表中的描述和图片链接提供了直观的信息展示，帮助用户更全面地了解每个品种的特点和外观。起源地信息让用户了解宠物的历史背景，增强对宠物的文化认知。体型、毛色、寿命等字段则为用户提供了具体的生理信息，便于用户做出符合自身条件和喜好的选择。

性情和饮食习惯字段提供了宠物行为和喂养方面的详细信息，这对于潜在宠物主来说尤为重要。了解宠物的性情有助于用户判断宠物是否适合家庭成员，特别是有小孩或其他宠物的家庭。饮食习惯信息则帮助用户提前准备合适的食物和喂养计划，确保新宠物能够健康成长。

通过这些详尽的分类和子类信息，平台不仅帮助用户在选择宠物时做出明智的决策，还通过教育用户增强他们的宠物知识和参与度。系统化的分类结构和丰富的子类信息，使得平台能够提供高质量的宠物百科内容，提升用户体验和满意度，增加平台的专业性和可信度。

## 2.18 有关宠物护理指导表的功能数据需求

宠物护理指导表提供了详尽的宠物护理和维护指南，是宠物主人非常有价值的资源。该表包含了针对不同宠物子类的具体护理建议，如日常护理、健康监测、饲养技巧等内容。通过这些专业的护理信息，平台帮助宠物主人提高照顾宠物的能力，确保宠物的健康和幸福，同时增强用户对平台的信任和依赖。

宠物护理指导表记录了每条护理建议的详细信息，包括唯一的指导 ID、相关的宠物子类 ID、标题和具体内容。这些字段确保了每条护理指导的唯一性和完整性，并提供了必要的细节来帮助宠物主人。

每条护理指导关联到特定的宠物子类，通过宠物子类 ID 字段，确保护理建议能够针对性地提供给相关的宠物主人。标题字段简明扼要地描述了护理建议的主题，方便用户快速浏览和查找所需信息。具体内容字段则提供了详细的护理步骤和建议，确保用户能够准确理解和执行。

日常护理建议包括了宠物的清洁、喂食和活动安排，帮助宠物主人建立科学的日常护理习惯。健康监测内容则涵盖了常见健康问题的识别和预防措施，如疫苗接种、定期体检等，确保宠物能够得到及时的医疗关注和照顾。饲养技巧则提供了关于宠物饮食、运动和行为训练的专业建议，帮助宠物主人培养健康和行为良好的宠物。

通过提供这些详细的护理指导，平台不仅提高了宠物主人的护理能力，也提升了宠物的生活质量和幸福感。同时，这些专业的指导内容也增强了用户对平台的依赖，提升了平台的用户黏性和满意度。用户在平台上找到的护理信息越多、越有用，他们就越可能长期使用平台，推荐给其他宠物爱好者，形成良性循环。

平台还可以通过分析用户对不同护理指导的使用情况，优化内容推荐和更新策略。例如，针对用户反馈和需求，平台可以定期更新护理指导内容，增加新的护理技巧和健康建议，确保信息的时效性和实用性。

## 2.19 有关宠物领养表的功能数据需求

宠物领养表是宠物领养服务的数据支持核心，记录了所有可供领养宠物的详细信息，包括宠物的名称、年龄、品种、健康状况、位置和领养状态等。该表不仅提供了一个平台让用户浏览和选择适合领养的宠物，还有助于宠物找到合适的家庭。

每条领养记录包含了详细的宠物信息，确保潜在领养者能够全面了解每只宠物的背景和需求。宠物名称和年龄字段提供了基本的身份信息，帮助用户识别和选择适合自己的宠物。品种信息通过关联宠物分类和子类表，提供了详细的品种特性描述，帮助用户了解宠物的性格、体型和寿命等重要信息。

健康状况字段记录了宠物的当前健康信息，包括任何已知的健康问题或特殊护理需求。疫苗接种记录则提供了宠物的免疫状况，确保潜在领养者能够了解宠物的防疫情况，做好相应的照顾准备。通过这些健康和护理信息，平台确保领养过程的透明和规范，增加领养成功率。

位置字段记录了宠物的当前所在地，方便潜在领养者了解宠物的地理位置，做出合理的领养安排。领养状态字段标明了宠物的当前领养进度，如待领养、已预订、已领养等，帮助用户了解宠物的可用情况。

此外，宠物领养表还记录了宠物的特殊护理需求、饮食习惯、性格行为特征等详细信息。这些信息帮助潜在领养者全面了解宠物的日常需求，确保他们能够提供适合的照顾和环境。通过这些详细记录，平台不仅提高了宠物的领养率，还确保宠物能够找到合适的家庭，过上幸福的生活。用户也可以通过上传附件资料来补充信息。

平台通过系统化地收集和分析领养数据，可以优化领养服务和用户体验。例如，通过分析用户的领养偏好和行为，平台可以提供个性化的领养推荐，增加领养成功率和用户满意度。同时，这些数据也可以用于改进宠物照顾指南和领养流程，提升平台的整体服务质量。

## 2.20 有关开发团队表的功能数据需求

开发团队表存储有关平台开发团队成员的详细信息，包括姓名、学校、电子邮箱、GitHub 用户名和个人资料链接。这个表的目的是增加平台的透明度和可信度，通过展示背后的团队，用户可以看到是谁在开发和维护他们正在使用的服务。

## 3 数据库概念设计

在本节（章节 3）中，我们将介绍数据库实体及相关属性、数据库总体 E-R 图和数据库模块 E-R 图。

### 3.1 数据库实体及相关属性

在数据库设计中，模块划分是根据业务功能和逻辑结构进行的。具体划分为用户模块、宠物社区模块、宠物新闻模块、以及宠物百科与领养模块。每个模块负责特定的功能，如用户模块处理用户信息；宠物社区模块涉及社交互动，如帖子、评论和用户互动；宠物新闻模块管理新闻发布和用户反馈；宠物百科与领养模块则涉及宠物的详细信息管理和领养过程。

#### 3.1.1 用户模块

用户模块相关的数据库实体及相关属性：

- **用户表**（用户 ID，用户名，密码，手机号码，注册日期，上次登录时间，用户角色，用户状态，头像链接，个人简介，性别，出生日期，经验点数，关注数，被关注数，收藏数，被收藏数，点赞数，被点赞数，留言数）
- **用户设置表**（用户 ID，是否公开手机号码，是否公开注册日期，是否公开个人简介，是否公开性别，是否公开出生日期，是否公开等级，是否公开关注数，是否公开被关注数，是否公开收藏数，是否公开被收藏数，是否公开点赞数，是否公开被点赞数，是否公开留言数，是否允许他人关注，是否接受管理员通知，是否接受用户通知）
- **用户打卡表**（打卡记录 ID，用户 ID，打卡时间）
- **用户关注表**（被关注者 ID，关注者 ID，关注时间）
- **用户留言表**（留言 ID，用户 ID，留言者 ID，留言内容，留言时间）
- **用户反馈表**（反馈意见 ID，反馈类型，反馈内容，反馈时间，电子邮箱，手机号码）

#### 3.1.2 宠物社区模块

宠物社区模块相关的数据库实体及相关属性：

- **帖子分类表**（帖子分类 ID，帖子分类）
- **帖子表**（帖子 ID，发帖用户 ID，帖子分类 ID，标题，内容，创建时间，更新时间，是否置顶，点赞数，点踩数，收藏数，评论数，图片链接）
- **帖子评论表**（帖子评论 ID，帖子 ID，用户 ID，父评论 ID，评论内容，评论时间，点赞数，点踩数）
- **帖子点赞表**（帖子 ID，用户 ID，点赞时间）
- **帖子点踩表**（帖子 ID，用户 ID，点踩时间）

- 帖子评论点赞表 (帖子评论 ID, 用户 ID, 点赞时间)
- 帖子评论点踩表 (帖子评论 ID, 用户 ID, 点踩时间)
- 帖子收藏表 (帖子 ID, 用户 ID, 收藏时间)
- 帖子举报表 (帖子举报 ID, 举报者 ID, 被举报者 ID, 被举报帖子 ID, 举报原因, 举报时间, 处理状态)
- 帖子评论举报表 (帖子评论举报 ID, 举报者 ID, 被举报者 ID, 被举报评论所属帖子 ID, 被举报评论 ID, 举报原因, 举报时间, 处理状态)

### 3.1.3 宠物新闻模块

宠物新闻模块相关的数据库实体及相关属性：

- 新闻标签表 (新闻标签 ID, 新闻标签)
- 新闻表 (新闻 ID, 管理员 ID, 新闻标签 ID, 标题, 新闻摘要, 内容, 创建时间, 更新时间, 封面图片链接, 是否置顶, 点赞数, 点踩数, 收藏数, 评论数)
- 新闻评论表 (新闻评论 ID, 新闻 ID, 用户 ID, 父评论 ID, 评论内容, 评论时间, 点赞数, 点踩数)
- 新闻点赞表 (新闻 ID, 用户 ID, 点赞时间)
- 新闻点踩表 (新闻 ID, 用户 ID, 点踩时间)
- 新闻评论点赞表 (新闻评论 ID, 用户 ID, 点赞时间)
- 新闻评论点踩表 (新闻评论 ID, 用户 ID, 点踩时间)
- 新闻收藏表 (新闻 ID, 用户 ID, 收藏时间)
- 新闻评论举报表 (新闻评论举报 ID, 举报者 ID, 被举报者 ID, 被举报评论所属新闻 ID, 被举报评论 ID, 举报原因, 举报时间, 处理状态)

### 3.1.4 宠物百科与领养模块

宠物百科模块相关的数据库实体及相关属性：

- 宠物分类表 (宠物分类 ID, 宠物分类名称 (十种语言), 描述 (十种语言), 图片链接)
- 宠物子类表 (宠物子类 ID, 宠物分类 ID, 宠物子类名称 (十种语言), 描述 (十种语言), 图片链接, 起源地 (十种语言), 体型 (十种语言), 毛色 (十种语言), 寿命 (十种语言), 性情 (十种语言), 饮食习惯 (十种语言), 经纬度参数)
- 宠物护理指导表 (宠物护理指导 ID, 宠物子类 ID, 标题 (十种语言), 内容 (十种语言))
- 宠物领养表 (宠物领养 ID, 用户 ID, 宠物分类 ID, 宠物子类 ID, 宠物名称, 图片链接, 宠物年龄, 地址, 转让原因, 健康情况, 最近一次健康检查日期, 疫苗接种情况, 特殊护理需求, 特殊饮食需求, 性格行为特征, 是否已绝育, 备注, 领养状态, 性别, 附件链接)

## 3.1.5 开发团队模块

开发团队模块相关的数据库实体及相关属性：

- 开发团队表（学号，姓名，学校，电子邮箱，Github 名称，Github 主页）

## 3.2 数据库总体 E-R 图

数据库总体 E-R 图见图 3.1。

数据库的总体 E-R 图是设计的宏观视图，展示了所有主要实体及其关系。这个图包括了用户、帖子、新闻、宠物分类、宠物子类等实体，以及它们之间的关系如一对多、多对多关系。例如，用户与帖子之间的多对一关系（一个用户可以发布多个帖子），帖子与评论之间的一对多关系（一个帖子可以有多个评论）。总体 E-R 图还明确展示了如用户设置、用户关注、用户留言、新闻评论等更细致的功能关系。通过这样的设计，可以清楚地理解数据库的结构和各部分之间的交互，为数据库的实际建设和未来的数据操作提供明确的指导。

## 3.3 数据库模块 E-R 图

### 3.3.1 用户模块 E-R 图

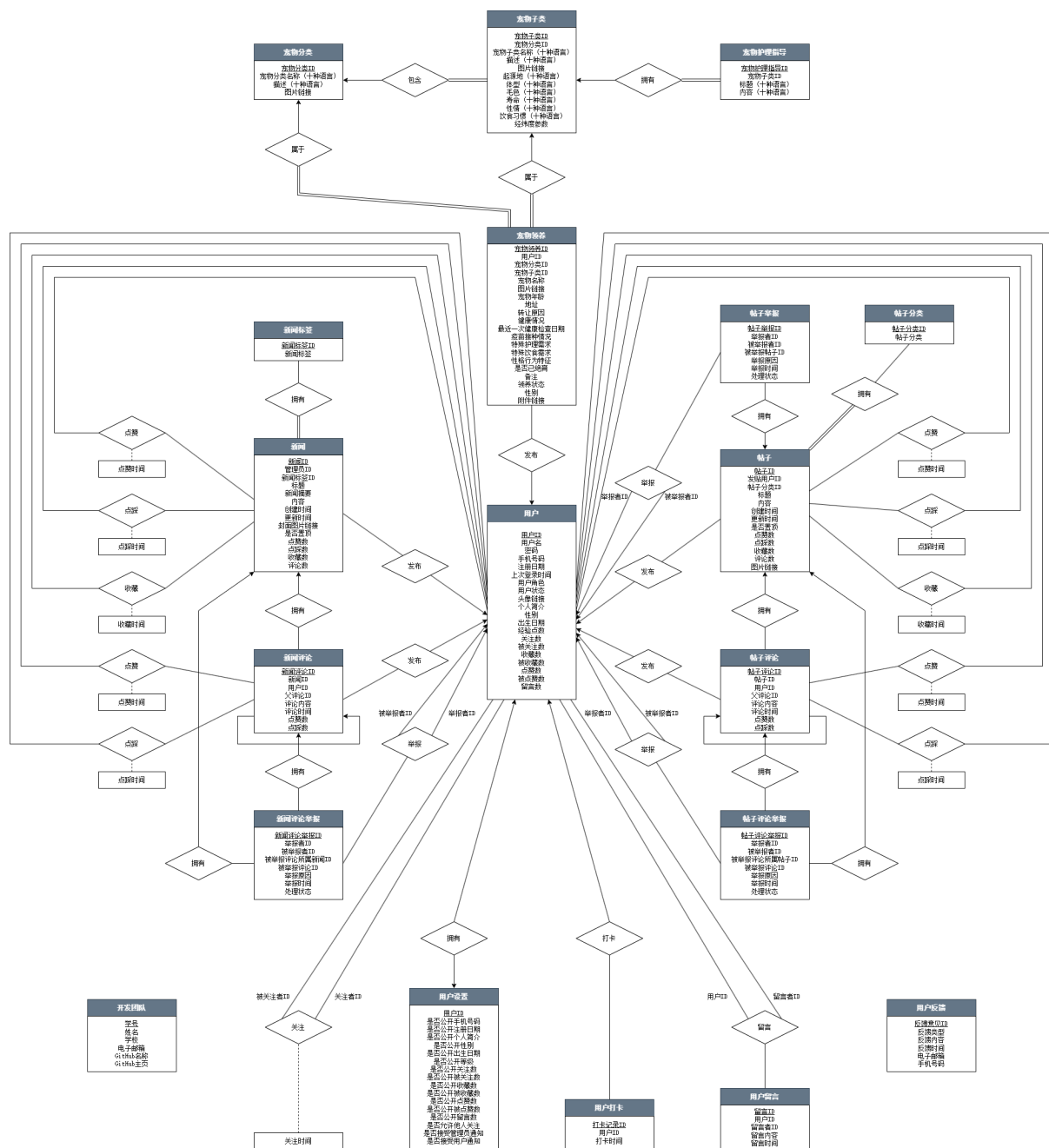
用户模块的 E-R 图设计专注于维护用户的基本信息和用户间的互动。核心实体包括用户表，用户设置表，用户打卡表，用户关注表，用户留言表。用户表存储所有注册用户的详细信息，如用户名、密码、联系方式等，同时提供唯一标识符来关联其他表中的用户活动和设置。用户设置表与用户表具有一对一关系，记录用户的个性化设置如隐私偏好和通知接收偏好。用户打卡表、用户关注表和用户留言表则描述用户的社交互动，展示了用户间如何互动和通信。

用户模块 E-R 图见图 3.2。

### 3.3.2 宠物社区模块 E-R 图

宠物社区模块的 E-R 图设计专注于处理与社区相关的用户互动，例如帖子的发布、评论、点赞、点踩、收藏以及举报等功能。此模块包括帖子表、帖子评论表、帖子点赞和点踩表、帖子收藏表以及帖子和评论的举报表。每个帖子可以有多个评论、多个点赞、多个点踩、多个收藏，并且可以被多次举报。用户可以发表帖子、对帖子进行评论和对评论进行回复。此外，E-R 图展示了用户与帖子及其交互之间的关系，如用户和帖子之间的发布关系、用户与点赞或点踩之间的关系。设计的重点在于明确和组织用户互动的数据结构，确保用户行为的追踪和管理，以便为社区成员提供丰富且互动的用户体验。

宠物社区模块 E-R 图见图 3.3。





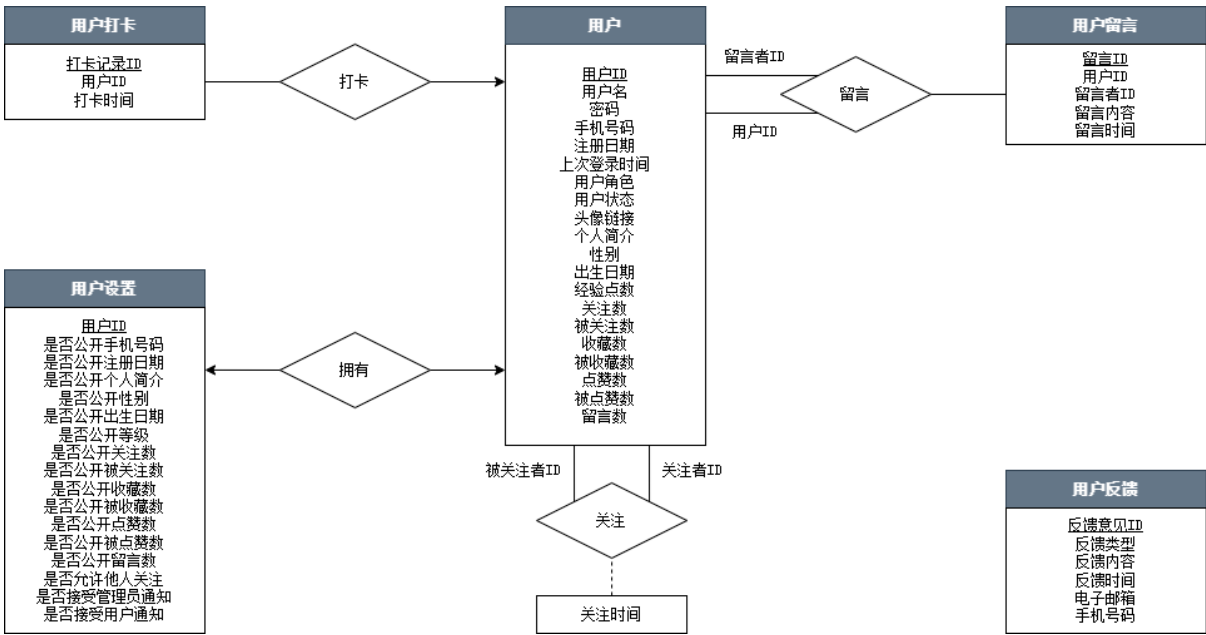


图 3.2 用户模块 E-R 图

3.3.3 宠物新闻模块 E-R 图

宠物新闻模块的 E-R 图设计关注于管理和展示新闻内容及其用户互动，包括新闻的发布、评论、点赞、点踩、收藏以及评论的举报。核心实体包括新闻表、新闻评论表、新闻点赞表、新闻点踩表、新闻收藏表和新闻评论举报表。新闻表与新闻标签表连接，表明每条新闻都归属于特定的新闻标签。新闻与用户的互动如评论、点赞、点踩和收藏通过一对多关系实现，每个新闻可以有多个评论和多次点赞或点踩，而每条评论也可以被多个用户点赞或点踩。此外，新闻和评论的举报功能允许用户报告不适当的内容，增强社区的内容管理效率。这个设计旨在确保新闻内容的有效管理和用户交互的流畅性，同时提供一个结构清晰、易于维护的数据库模型。

宠物新闻模块 E-R 图见图 3.4。

3.3.4 宠物百科与领养模块 E-R 图

宠物百科与领养模块的 E-R 图设计旨在详细管理宠物相关信息以及领养过程中的所有数据交互。这个模块包括宠物分类表、宠物子类表、宠物护理指导表以及宠物领养表等实体。宠物分类与子类之间的关系是一对多，表示一个宠物分类包含多个子类，例如犬类包括不同的狗种。每个宠物子类还关联到具体的护理指导，以提供针对性的护理信息。宠物领养表核心连接用户与宠物，每个领养记录关联一个用户和一个具体的宠物子类，确保领养过程中的信息一致性和完整性。此外，每个领养记录详细记录了宠物的健康状况、位置、年龄等关键信息。这个设计不仅有助于用户轻松获取宠物的详细信息和护理指南，还促进了透明和规范的宠物领养流程。

宠物百科与领养模块 E-R 图见图 3.5。

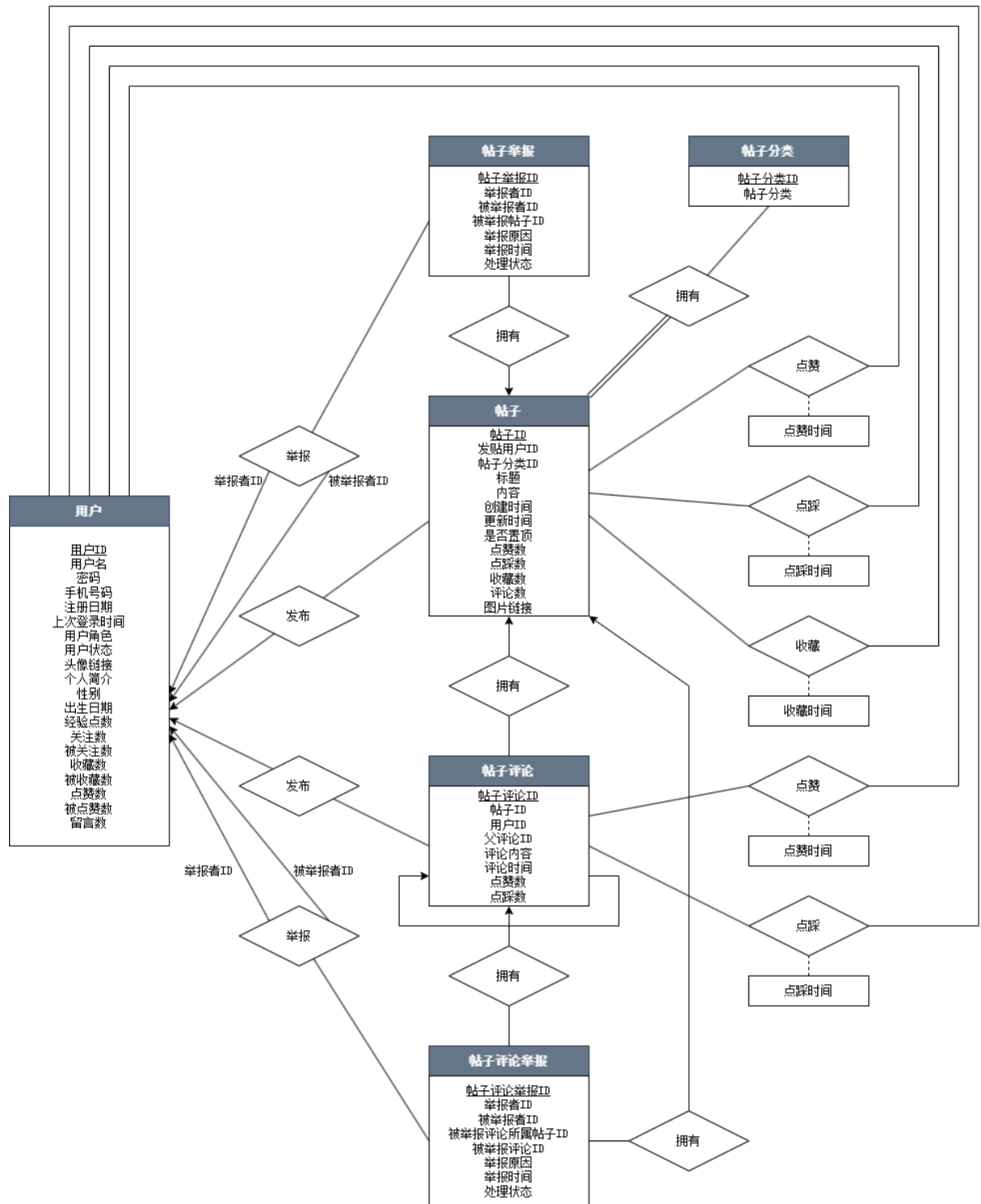


图 3.3 宠物社区模块 E-R 图

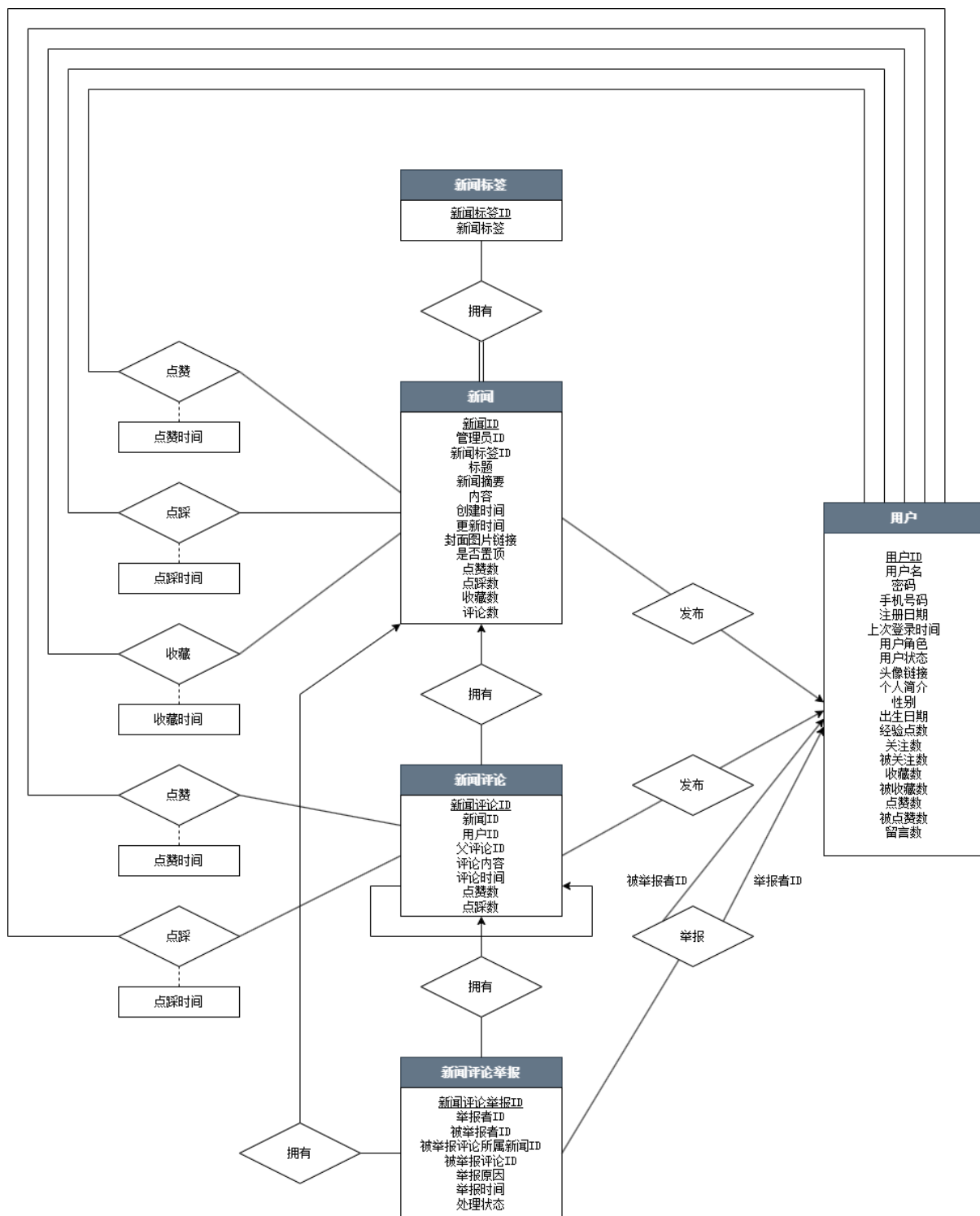


图 3.4 宠物新闻模块 E-R 图

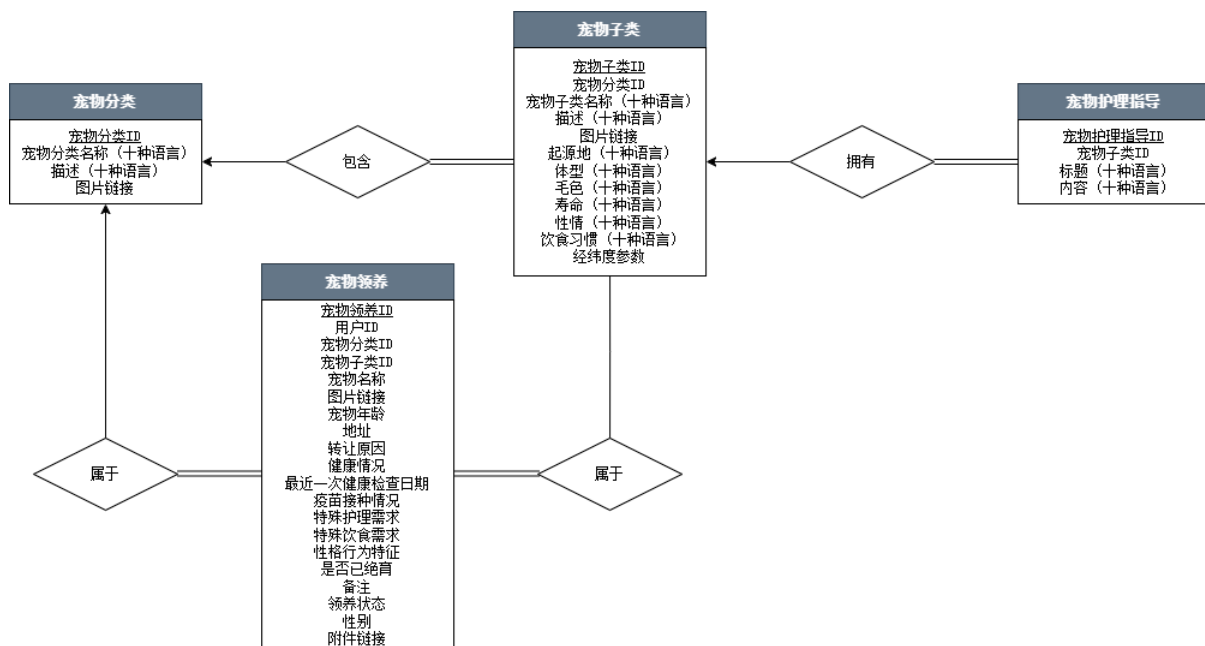


图 3.5 宠物百科与领养模块 E-R 图

### 3.3.5 开发团队模块 E-R 图

开发模块的 E-R 图设计相对简单，因为它主要涉及一个单独的表——开发团队表（DEVELOPMENT\_TEAM），该表用于存储开发团队成员的详细信息。这个表是独立的，不与数据库中的其他实体或表直接关联。其设计目的主要是为了在应用或网站的“关于宠悦”界面上展示开发团队的信息，如成员姓名、学号、学校、电子邮箱、GitHub 名称和 GitHub 主页等。

开发团队模块 E-R 图见图 3.6。

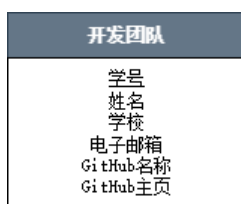


图 3.6 开发团队模块 E-R 图

在 E-R 图中，开发团队表被展示为一个单独的实体，具有自己的属性：

- **主键：**学号 (id)
- **其他属性：**姓名 (name)、学校 (school)、电子邮箱 (email)、GitHub 名称 (github\_name)、GitHub 主页 (github\_profile)

由于该表没有与其他表的关系，E-R 图中不会显示任何连接线或外键关联。这种设计强调了表的独立性和特定用途，即仅用于信息展示而不参与系统的其他功能或逻辑处理。这样的设计也有利

于简化数据库维护和数据管理，因为它不涉及复杂的关系处理，只需关注信息的准确性和更新。

|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
装  
|  
|  
|  
|  
|  
订  
|  
|  
|  
|  
|  
线  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|

## 4 数据库逻辑设计

在本节（章节 4）中，我们将介绍数据库逻辑设计，包括数据库关系模式图、表设计、数据库设计和数据定义语言（DDL）设计。

### 4.1 数据库关系模式图

数据库关系模式图（图 4.1）的设计反映了本项目的信息系统。

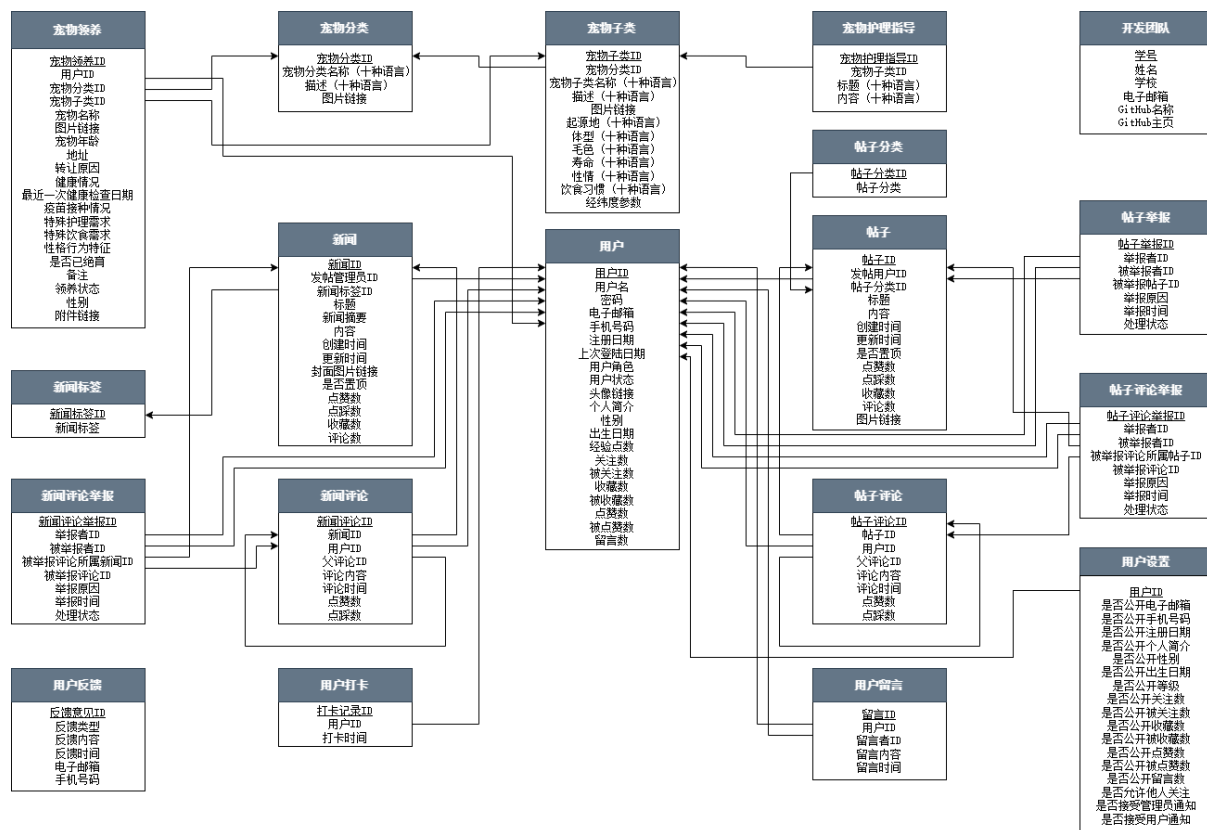


图 4.1 数据库关系模式图

#### 4.1.1 模块化设计

数据库的模块化设计通过将功能划分为独立的表来实现，例如用户信息、新闻、帖子和宠物等。每个模块作为一个单独的表存在，确保了系统内部的高度组织和结构化，同时显著降低了不同模块间的依赖性。

#### 4.1.2 用户中心设计

用户表作为数据库中的核心表，通过“用户 ID”与多个其他表如用户打卡表、用户留言表进行关联，这强调了用户在平台中的中心作用。该设计允许系统有效地集中处理和访问关于用户的所有

信息，包括他们的个人设置、活动记录以及与其他用户的互动。这样的集中管理有助于优化用户体验，并确保数据的一致性和准确性。

#### 4.1.3 内容管理设计

在内容管理方面，帖子表和新闻表是核心组件，与评论、点赞、点踩、收藏和举报等功能相关联。这种设计不仅体现了管理内容时的复杂性，还展示了如何有效追踪用户与内容的互动，如点赞和评论。此外，评论系统的设计支持对帖子和新闻的评论进行进一步的互动，如点赞或点踩，从而增强了平台的互动性和用户参与度。

#### 4.1.4 举报系统设计

通过帖子举报表、帖子评论举报表和新闻评论举报表，本系统为用户提供了一个举报不当内容的有效机制。这些表的设计保证了用户可以安全地报告问题内容，同时系统能够有效地处理这些举报，确保平台内容的健康和合规性，维护了一个积极和健康的社区环境。

#### 4.1.5 宠物领养模块设计

宠物领养模块通过宠物分类表、宠物子类表和宠物领养表的设计，实现了对宠物信息的详细分类和管理。这些表不仅提供了宠物的基本信息和分类，还详细记录了每只宠物的特定信息，如健康状况、领养需求等，确保潜在领养者能够找到与他们情况相匹配的宠物。

#### 4.1.6 开发团队信息设计

开发团队表聚焦于记录平台维护和开发团队成员的详细信息，如姓名、联系方式、所属学校等。这不仅有助于内部的团队管理和沟通，也提供了一个渠道供用户了解开发团队的背景，从而增强用户对平台的信任 and 安全感。

### 4.2 表设计

在本项目中，我们设计了多种表来存储用户信息、社区互动、内容发布、新闻系统、宠物领养等信息，这些表设计体现了一个为宠物爱好者提供的一站式信息与资源交流天地的数据需求。

在表设计过程中，我们综合考虑了规范化、主键和索引、外键、数据类型和约束、安全性和隐私、性能优化、数据持久化、扩展性和可维护性等方面。这样的表设计涵盖了本平台的核心功能，但在实践中，可能需要根据实际应用场景和性能测试结果进行调整和优化，设计时也充分考虑了将来可能的功能扩展和数据迁移的便利性。

本项目设计的表及其属性如下：

- 1. 用户表** (用户 ID, 用户名, 密码, 手机号码, 注册日期, 上次登录时间, 用户角色, 用户状态, 头像链接, 个人简介, 性别, 出生日期, 经验点数, 关注数, 被关注数, 收藏数, 被收

藏数，点赞数，被点赞数，留言数)

2. 用户设置表 (用户 ID, 是否公开手机号码, 是否公开注册日期, 是否公开个人简介, 是否公开性别, 是否公开出生日期, 是否公开等级, 是否公开关注数, 是否公开被关注数, 是否公开收藏数, 是否公开被收藏数, 是否公开点赞数, 是否公开被点赞数, 是否公开留言数, 是否允许他人关注, 是否接受管理员通知, 是否接受用户通知)
3. 用户打卡表 (打卡记录 ID, 用户 ID, 打卡时间)
4. 用户关注表 (被关注者 ID, 关注者 ID, 关注时间)
5. 用户留言表 (留言 ID, 用户 ID, 留言者 ID, 留言内容, 留言时间)
6. 用户反馈表 (反馈意见 ID, 反馈类型, 反馈内容, 反馈时间, 电子邮箱, 手机号码)
7. 帖子分类表 (帖子分类 ID, 帖子分类)
8. 帖子表 (帖子 ID, 发帖用户 ID, 帖子分类 ID, 标题, 内容, 创建时间, 更新时间, 是否置顶, 点赞数, 点踩数, 收藏数, 评论数, 图片链接)
9. 帖子评论表 (帖子评论 ID, 帖子 ID, 用户 ID, 父评论 ID, 评论内容, 评论时间, 点赞数, 点踩数)
10. 帖子点赞表 (帖子 ID, 用户 ID, 点赞时间)
11. 帖子点踩表 (帖子 ID, 用户 ID, 点踩时间)
12. 帖子评论点赞表 (帖子评论 ID, 用户 ID, 点赞时间)
13. 帖子评论点踩表 (帖子评论 ID, 用户 ID, 点踩时间)
14. 帖子收藏表 (帖子 ID, 用户 ID, 收藏时间)
15. 帖子举报表 (帖子举报 ID, 举报者 ID, 被举报者 ID, 被举报帖子 ID, 举报原因, 举报时间, 处理状态)
16. 帖子评论举报表 (帖子评论举报 ID, 举报者 ID, 被举报者 ID, 被举报评论所属帖子 ID, 被举报评论 ID, 举报原因, 举报时间, 处理状态)
17. 新闻标签表 (新闻标签 ID, 新闻标签)
18. 新闻表 (新闻 ID, 管理员 ID, 新闻标签 ID, 标题, 新闻摘要, 内容, 创建时间, 更新时间, 封面图片链接, 是否置顶, 点赞数, 点踩数, 收藏数, 评论数)
19. 新闻评论表 (新闻评论 ID, 新闻 ID, 用户 ID, 父评论 ID, 评论内容, 评论时间, 点赞数, 点踩数)
20. 新闻点赞表 (新闻 ID, 用户 ID, 点赞时间)
21. 新闻点踩表 (新闻 ID, 用户 ID, 点踩时间)
22. 新闻评论点赞表 (新闻评论 ID, 用户 ID, 点赞时间)
23. 新闻评论点踩表 (新闻评论 ID, 用户 ID, 点踩时间)
24. 新闻收藏表 (新闻 ID, 用户 ID, 收藏时间)
25. 新闻评论举报表 (新闻评论举报 ID, 举报者 ID, 被举报者 ID, 被举报评论所属新闻 ID, 被



举报评论 ID, 举报原因, 举报时间, 处理状态)

26. **宠物分类表** (宠物分类 ID, 宠物分类名称 (汉语), 宠物分类名称 (德语), 宠物分类名称 (英语), 宠物分类名称 (西班牙语), 宠物分类名称 (法语), 宠物分类名称 (意大利语), 宠物分类名称 (日语), 宠物分类名称 (韩语), 宠物分类名称 (葡萄牙语), 宠物分类名称 (俄语), 描述 (汉语), 描述 (德语), 描述 (英语), 描述 (西班牙语), 描述 (法语), 描述 (意大利语), 描述 (日语), 描述 (韩语), 描述 (葡萄牙语), 描述 (俄语), 图片链接)
27. **宠物子类表** (宠物子类 ID, 宠物分类 ID, 宠物子类名称 (汉语), 宠物子类名称 (德语), 宠物子类名称 (英语), 宠物子类名称 (西班牙语), 宠物子类名称 (法语), 宠物子类名称 (意大利语), 宠物子类名称 (日语), 宠物子类名称 (韩语), 宠物子类名称 (葡萄牙语), 宠物子类名称 (俄语), 描述 (汉语), 描述 (德语), 描述 (英语), 描述 (西班牙语), 描述 (法语), 描述 (意大利语), 描述 (日语), 描述 (韩语), 描述 (葡萄牙语), 描述 (俄语), 图片链接, 起源地 (汉语), 起源地 (德语), 起源地 (英语), 起源地 (西班牙语), 起源地 (法语), 起源地 (意大利语), 起源地 (日语), 起源地 (韩语), 起源地 (葡萄牙语), 起源地 (俄语), 体型 (汉语), 体型 (德语), 体型 (英语), 体型 (西班牙语), 体型 (法语), 体型 (意大利语), 体型 (日语), 体型 (韩语), 体型 (葡萄牙语), 体型 (俄语), 毛色 (汉语), 毛色 (德语), 毛色 (英语), 毛色 (西班牙语), 毛色 (法语), 毛色 (意大利语), 毛色 (日语), 毛色 (韩语), 毛色 (葡萄牙语), 毛色 (俄语), 寿命 (汉语), 寿命 (德语), 寿命 (英语), 寿命 (西班牙语), 寿命 (法语), 寿命 (意大利语), 寿命 (日语), 寿命 (韩语), 寿命 (葡萄牙语), 寿命 (俄语), 性情 (汉语), 性情 (德语), 性情 (英语), 性情 (西班牙语), 性情 (法语), 性情 (意大利语), 性情 (日语), 性情 (韩语), 性情 (葡萄牙语), 性情 (俄语), 饮食习惯 (汉语), 饮食习惯 (德语), 饮食习惯 (英语), 饮食习惯 (西班牙语), 饮食习惯 (法语), 饮食习惯 (意大利语), 饮食习惯 (日语), 饮食习惯 (韩语), 饮食习惯 (葡萄牙语), 饮食习惯 (俄语), 经纬度参数)
28. **宠物护理指导表** (宠物护理指导 ID, 宠物子类 ID, 标题 (汉语), 标题 (德语), 标题 (英语), 标题 (西班牙语), 标题 (法语), 标题 (意大利语), 标题 (日语), 标题 (韩语), 标题 (葡萄牙语), 标题 (俄语), 内容 (汉语), 内容 (德语), 内容 (英语), 内容 (西班牙语), 内容 (法语), 内容 (意大利语), 内容 (日语), 内容 (韩语), 内容 (葡萄牙语), 内容 (俄语))
29. **宠物领养表** (宠物领养 ID, 用户 ID, 宠物分类 ID, 宠物子类 ID, 宠物名称, 图片链接, 宠物年龄, 地址, 转让原因, 健康情况, 最近一次健康检查日期, 疫苗接种情况, 特殊护理需求, 特殊饮食需求, 性格行为特征, 是否已绝育, 备注, 领养状态, 性别, 附件链接)
30. **开发团队表** (学号, 姓名, 学校, 电子邮箱, Github 名称, Github 主页)

4.2.1 用户表 USER

用户表 (USER) 是社交平台数据库中核心的数据表之一，负责存储所有用户的基本信息和账户细节。该表包括了用户 ID、用户名、密码、手机号码等标识性信息，以及用户的注册日期和最后登录时间等活动记录。此外，用户表还详细记录了用户的角色和状态，支持安全特性如密码保护。个性化数据如头像链接、个人简介、性别和出生日期也被包括在内，为用户提供个性化的服务。此表还记录了与用户活动相关的统计数据，如关注数、被关注数、收藏数、被收藏数、点赞数、被点赞数和留言数等，这些数据对于用户互动和平台运营分析尤为重要。整体上，用户表的设计旨在确保用户数据的完整性、安全性和高效的数据访问，以支持社交平台的日常运营和用户管理。(表 4.1)

表 4.1 用户表 USER

字段名	数据类型	长度	说明	备注
user_id	INT		用户 ID	PK、非空
user_name	VARCHAR	64	用户名	非空、唯一、最长存储 16 个中文字符
password	VARCHAR	64	密码	非空、存储密码明文的哈希值
telephone	VARCHAR	16	手机号码	非空、唯一
registration_date	DATE		注册日期	非空
last_login_time	DATE		上次登录时间	非空
role	NUMBER(1)		用户角色	非空、Boolean: 0 - 普通用户; 1 - 管理
status	NUMBER(1)		用户状态	非空、Boolean: 0 - 正常; 1 - 封禁
avatar_url	VARCHAR	2048	头像链接	
profile	VARCHAR	512	个人简介	最长存储 128 个中文字符
gender	NUMBER(1)		性别	非空、Boolean: 0 - 男性; 1 - 女性
birthdate	DATE		出生日期	非空
experience_points	INT		经验点数	非空
follows_count	INT		关注数	非空
followed_count	INT		被关注数	非空
favorites_count	INT		收藏数	非空
favorited_count	INT		被收藏数	非空
likes_count	INT		点赞数	非空
liked_count	INT		被点赞数	非空
message_count	INT		留言数	非空

4.2.2 用户设置表 USER\_SETTING

用户设置表 (USER\_SETTING) 是一个关键的数据表，用于存储和管理用户的个人偏好设置。这包括对隐私设置（例如是否公开邮箱、电话号码等个人信息）以及通知偏好（是否接收来自管理员或其他用户的通知）的选择。每个设置项都通过一个布尔字段来表示，确保用户可以个性化其在

平台上的体验。表中的每一条记录都与用户表中的一个特定用户 ID 相关联，通过外键关系维护数据的一致性和完整性。(表 4.2)

表 4.2 用户设置表 USER\_SETTING

字段名	数据类型	长度	说明	备注
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
is_telephone_public	NUMBER(1)		是否公开手机号码	非空、Boolean: 0 - 不公开; 1 - 公开
is_registration_date_public	NUMBER(1)		是否公开注册日期	非空、Boolean: 0 - 不公开; 1 - 公开
is_profile_public	NUMBER(1)		是否公开个人简介	非空、Boolean: 0 - 不公开; 1 - 公开
is_gender_public	NUMBER(1)		是否公开性别	非空、Boolean: 0 - 不公开; 1 - 公开
is_birthdate_public	NUMBER(1)		是否公开出生日期	非空、Boolean: 0 - 不公开; 1 - 公开
is_level_public	NUMBER(1)		是否公开等级	非空、Boolean: 0 - 不公开; 1 - 公开
is_following_count_public	NUMBER(1)		是否公开关注数	非空、Boolean: 0 - 不公开; 1 - 公开
is_followers_count_public	NUMBER(1)		是否公开被关注数	非空、Boolean: 0 - 不公开; 1 - 公开
is_favorites_count_public	NUMBER(1)		是否公开收藏数	非空、Boolean: 0 - 不公开; 1 - 公开
is_favored_count_public	NUMBER(1)		是否公开被收藏数	非空、Boolean: 0 - 不公开; 1 - 公开
is_likes_count_public	NUMBER(1)		是否公开点赞数	非空、Boolean: 0 - 不公开; 1 - 公开
is_liked_count_public	NUMBER(1)		是否公开被点赞数	非空、Boolean: 0 - 不公开; 1 - 公开
is_message_count_public	NUMBER(1)		是否公开留言数	非空、Boolean: 0 - 不公开; 1 - 公开
allow_following	NUMBER(1)		是否允许他人关注	非空、Boolean: 0 - 不允许; 1 - 允许
receive_admin_notifications	NUMBER(1)		是否接受管理员通知	非空、Boolean: 0 - 屏蔽; 1 - 接受
receive_user_notifications	NUMBER(1)		是否接受用户通知	非空、Boolean: 0 - 屏蔽; 1 - 接受

4.2.3 用户打卡表 USER\_CHECK\_IN

用户打卡表 (USER\_CHECK\_IN) 设计用来记录用户的每日打卡活动，支持用户习惯性的日常签到或特定任务的完成记录。该表包含三个主要字段：打卡记录 ID (check\_in\_id)，作为主键确保每条记录的唯一性；用户 ID (user\_id)，作为外键与用户表 (USER) 连接，表示哪个用户完成了打卡；以及打卡时间 (check\_in\_time)，记录用户具体的打卡时刻。此表的存在有助于促进用户的活跃参与和社交平台的用户粘性，同时为平台提供了一个简便的途径来跟踪和激励用户的日常活动。通过分析打卡数据，平台可以更好地了解用户活跃度，并根据这些数据优化用户体验和提供个性化的内容。(表 4.3)

表 4.3 用户打卡表 USER\_CHECK\_IN

字段名	数据类型	长度	说明	备注
check_in_id	INT		打卡记录 ID	PK、非空

续下页

续表 4.3

字段名	数据类型	长度	说明	备注
user_id	INT		用户 ID	FK to USER(user_id)、非空
check_in_time	DATE		打卡时间	非空

#### 4.2.4 用户关注表 USER\_FOLLOW

用户关注表 (USER\_FOLLOW) 是用于记录和管理用户之间关注关系的数据表。该表主要包含三个字段：被关注者 ID (user\_id)、关注者 ID (follower\_id) 以及关注时间 (follow\_time)。这两个 ID 字段均设为主键并与用户表 (USER) 的用户 ID 字段通过外键关联。(表 4.4)

表 4.4 用户关注表 USER\_FOLLOW

字段名	数据类型	长度	说明	备注
user_id	INT		被关注者 ID	PK、FK to USER(user_id)、非空
follower_id	INT		关注者 ID	PK、FK to USER(user_id)、非空
follow_time	DATE		关注时间	非空

#### 4.2.5 用户留言表 USER\_MESSAGE

用户留言表 (USER\_MESSAGE) 是设计用来记录用户间交流的留言信息的数据表。该表包含留言 ID (message\_id)，作为主键确保每条留言的唯一性；用户 ID (user\_id) 指被留言者，与用户表的用户 ID 通过外键关联；留言者 ID (commenter\_id)，也通过外键与用户表关联，表示留言的发送者；留言内容 (message)，存储用户之间传递的具体文本信息；以及留言时间 (comment\_time)，记录留言创建的具体时刻。(表 4.5)

表 4.5 用户留言表 USER\_MESSAGE

字段名	数据类型	长度	说明	备注
message_id	INT		留言 ID	PK、非空
user_id	INT		用户 ID	FK to USER(user_id)、非空
commenter_id	INT		留言者 ID	FK to USER(user_id)、非空
message	VARCHAR	512	留言内容	非空、最长存储 128 个中文字符
comment_time	DATE		留言时间	非空

#### 4.2.6 用户反馈表 USER\_FEEDBACK

用户反馈表 (USER\_FEEDBACK) 用于记录和管理用户提供的反馈信息。该表包括多个字段，如反馈意见 ID (feedback\_id)，用作主键并且不能为空；反馈类型 (feedback\_category) 和反馈内容

(`feedback_content`)，用以详细说明用户的反馈点，这两者均为必填项；反馈时间 (`feedback_time`)，记录用户反馈的具体时间，也是必填项。此外，表中还包含用户的联系方式，如电子邮箱 (`email`) 和手机号码 (`telephone`)，这些字段可以为空，以便在需要时联系用户。整个表设计用于高效管理用户反馈，便于后续的分析和响应。(表 4.6)

表 4.6 用户反馈表 `USER_FEEDBACK`

字段名	数据类型	长度	说明	备注
<code>feedback_id</code>	INT		反馈意见 ID	PK、非空
<code>feedback_category</code>	VARCHAR	32	反馈类型	非空
<code>feedback_content</code>	VARCHAR	2048	反馈内容	非空、最长存储 512 个中文字符
<code>feedback_time</code>	DATE		反馈时间	非空
<code>email</code>	VARCHAR	2048	电子邮箱	
<code>telephone</code>	VARCHAR	16	手机号码	

## 4.2.7 帖子分类表 `POST_CATEGORY`

帖子分类表 (`POST_CATEGORY`) 是是用来管理和组织社区中帖子的分类信息的重要数据结构。这个表的存在使得平台能够更有效地组织内容，用户可以根据分类快速找到自己感兴趣的帖子。同时，这种分类机制也便于平台进行内容的推荐和管理，通过分析不同分类下的用户行为和反馈，平台可以调整内容的发布和推广策略，以满足用户的不同需求。(表 4.7)

表 4.7 帖子分类表 `POST_CATEGORY`

字段名	数据类型	长度	说明	备注
<code>category_id</code>	INT		帖子分类 ID	PK、非空
<code>category</code>	VARCHAR	64	帖子分类	非空、唯一、最长存储 16 个中文字符

## 4.2.8 帖子表 `POST`

帖子表 (`POST`) 是社交平台核心的数据结构，用于记录用户的发布内容。每条帖子包含唯一的帖子 ID (`post_id`) 作为主键，关联的用户 ID (`user_id`) 以链接到用户表，以及帖子的分类 ID (`category_id`)。帖子的主要信息包括标题 (`title`) 和内容 (`content`)，两者都有明确的字符长度限制。此外，帖子的生命周期通过创建时间 (`creation_date`) 和更新时间 (`update_date`) 进行跟踪。表中还包括一些用于社交互动的字段，如是否置顶 (`is_sticky`)、点赞数 (`like_count`)、点踩数 (`dislike_count`)、收藏数 (`favorite_count`) 和评论数 (`comment_count`)，以及可选的图片链接 (`image_url`)。这样的设计不仅优化了内容管理和用户互动，还提供了数据分析和内容推荐的基础。(表 4.8)

表 4.8 帖子表 POST

字段名	数据类型	长度	说明	备注
post_id	INT		帖子 ID	PK、非空
user_id	INT		发帖用户 ID	FK to USER(user_id)、非空
category_id	INT		帖子分类 ID	FK to POST_CATEGORY(category_id)、非空
title	VARCHAR	256	标题	非空、最长存储 64 个中文字符
content	VARCHAR	2048	内容	非空、最长存储 512 个中文字符
creation_date	DATE		创建时间	非空
update_date	DATE		更新时间	非空
is_sticky	NUMBER(1)		是否置顶	非空、Boolean: 0 - 否; 1 - 是
like_count	INT		点赞数	非空
dislike_count	INT		点踩数	非空
favorite_count	INT		收藏数	非空
comment_count	INT		评论数	非空
image_url	VARCHAR	2048	图片链接	

#### 4.2.9 帖子评论表 POST\_COMMENT

帖子评论表 (POST\_COMMENT) 是设计用于存储用户对帖子的评论及其相互作用的数据表。此表包括多个字段，如评论 ID (comment\_id)，作为主键提供唯一标识；帖子 ID (post\_id) 和用户 ID (user\_id)，分别通过外键关联到帖子表 (POST) 和用户表 (USER)，确保评论与特定帖子和用户相关联；父评论 ID (parent\_comment\_id)，允许评论形成层级结构，增加对话的深度和复杂性。此外，该表还跟踪每条评论的点赞数 (like\_count) 和点踩数 (dislike\_count)，这些数据对于评估社区对评论的接受程度和互动活跃性非常重要。(表 4.9)

表 4.9 帖子评论表 POST\_COMMENT

字段名	数据类型	长度	说明	备注
comment_id	INT		帖子评论 ID	PK、非空
post_id	INT		帖子 ID	FK to POST(post_id)、非空
user_id	INT		用户 ID	FK to USER(user_id)、非空
parent_comment_id	INT		父评论 ID	FK to POST_COMMENT(comment_id)
content	VARCHAR	512	评论内容	非空、最长存储 128 个中文字符
comment_time	DATE		评论时间	非空
like_count	INT		点赞数	非空
dislike_count	INT		点踩数	非空

#### 4.2.10 帖子点赞表 POST\_LIKE

帖子点赞表 (POST\_LIKE) 是用于记录用户对帖子的点赞行为的数据表。该表主要包含三个字段：帖子 ID (post\_id) 和用户 ID (user\_id)，这两个字段都设置为主键，并通过外键分别与帖子表 (POST) 和用户表 (USER) 相关联，确保点赞的行为明确指向特定的帖子和用户；点赞时间 (like\_time)，记录用户进行点赞的具体时间。这种设计使得每个点赞都被独立记录，支持了对用户互动行为的详尽追踪，有助于增强内容的可见性和用户的参与感。此外，它也为社交平台提供了重要的数据，用于分析用户行为、优化内容推荐算法以及提升用户体验。(表 4.10)

表 4.10 帖子点赞表 POST\_LIKE

字段名	数据类型	长度	说明	备注
post_id	INT		帖子 ID	PK、FK to POST(post_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
like_time	DATE		点赞时间	非空

#### 4.2.11 帖子点踩表 POST\_DISLIKE

帖子点踩表 (POST\_DISLIKE) 是用于记录用户对帖子的不赞同或反对行为的数据表。该表主要包含三个字段：帖子 ID (post\_id) 和用户 ID (user\_id)，这两个字段均作为主键，并且分别通过外键与帖子表 (POST) 和用户表 (USER) 连接，确保点踩行为明确针对特定的帖子和特定的用户；点踩时间 (dislike\_time)，记录用户进行点踩的具体时刻。这种数据记录允许平台跟踪和分析不受欢迎或有争议的内容，为内容管理和质量控制提供参考。(表 4.11)

表 4.11 帖子点踩表 POST\_DISLIKE

字段名	数据类型	长度	说明	备注
post_id	INT		帖子 ID	PK、FK to POST(post_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
dislike_time	DATE		点踩时间	非空

#### 4.2.12 帖子评论点赞表 POST\_COMMENT\_LIKE

帖子评论点赞表 (POST\_COMMENT\_LIKE) 是用于记录用户对帖子评论的赞同行为的数据表。该表包括三个主要字段：评论 ID (comment\_id)，用户 ID (user\_id) 和点赞时间 (like\_time)。评论 ID 和用户 ID 共同作为复合主键，并通过外键分别与帖子评论表 (POST\_COMMENT) 和用户表 (USER) 关联，确保每个点赞行为都可以准确地链接到特定的评论和特定的用户。(表 4.12)

表 4.12 帖子评论点赞表 POST\_COMMENT\_LIKE

字段名	数据类型	长度	说明	备注
comment_id	INT		帖子评论 ID	PK、FK to POST_COMMENT(comment_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
like_time	DATE		点赞时间	非空

#### 4.2.13 帖子评论点踩表 POST\_COMMENT\_DISLIKE

帖子评论点踩表 (POST\_COMMENT\_DISLIKE) 是用于记录用户对帖子评论的不赞同或反对行为的数据表。该表的设计包括三个关键字段：评论 ID (comment\_id)、用户 ID (user\_id) 和点踩时间 (dislike\_time)。评论 ID 和用户 ID 共同构成复合主键，确保每条点踩记录都是独一无二的，并通过外键与帖子评论表 (POST\_COMMENT) 及用户表 (USER) 相连接，从而确保每个点踩行为都能准确指向特定的评论和特定的用户。点踩时间字段记录了点踩的具体时刻。这种记录机制不仅帮助平台监控社区内的负面互动，也提供了反馈机制，用于评估内容的争议性，支持平台在必要时采取相应的内容管理措施以维护健康的讨论环境。(表 4.13)

表 4.13 帖子评论点踩表 POST\_COMMENT\_DISLIKE

字段名	数据类型	长度	说明	备注
comment_id	INT		帖子评论 ID	PK、FK to POST_COMMENT(comment_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
dislike_time	DATE		点踩时间	非空

#### 4.2.14 帖子收藏表 POST\_FAVORITE

帖子收藏表 (POST\_FAVORITE) 是设计用来记录用户对帖子的收藏行为的数据表，使用户能够保存他们喜欢或希望稍后查看的内容。此表包含三个主要字段：帖子 ID (post\_id)、用户 ID (user\_id) 和收藏时间 (favorite\_time)。帖子 ID 和用户 ID 共同作为复合主键，并通过外键分别与帖子表 (POST) 和用户表 (USER) 关联，确保每次收藏都能准确地指向特定的帖子和用户。收藏时间字段记录了用户收藏帖子的具体时间点。此表的存在不仅方便用户管理他们感兴趣的内容，也为平台提供了关于用户喜好和内容受欢迎程度的重要数据，有助于优化内容推荐算法和提高用户满意度。(表 4.14)

表 4.14 帖子收藏表 POST\_FAVORITE

字段名	数据类型	长度	说明	备注
post_id	INT		帖子 ID	PK、FK to POST(post_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空

续下页



续表 4.14

字段名	数据类型	长度	说明	备注
favorite_time	DATE		收藏时间	非空

#### 4.2.15 帖子举报表 POST\_REPORT

帖子举报表 (POST\_REPORT) 是用于记录用户对不适当或违反社区规定的帖子的举报行为的数据表。该表具备举报的详细信息，包括举报 ID (post\_report\_id)，作为主键确保每条举报记录的唯一性；举报者 ID (reporter\_id) 和被举报者 ID (reported\_user\_id)，通过外键与用户表 (USER) 关联，明确举报与被举报的用户；被举报帖子 ID (reported\_post\_id)，通过外键与帖子表 (POST) 关联，指向具体的被举报内容。此外，举报原因 (report\_reason) 详细记录用户举报的具体理由，举报时间 (report\_time) 记录了举报发生的具体时刻，处理状态 (status) 跟踪举报的审查进度。(表 4.15)

表 4.15 帖子举报表 POST\_REPORT

字段名	数据类型	长度	说明	备注
post_report_id	INT		帖子举报 ID	PK、非空
reporter_id	INT		举报者 ID	FK to USER(user_id)、非空
reported_user_id	INT		被举报者 ID	FK to USER(user_id)、非空
reported_post_id	INT		被举报帖子 ID	FK to POST(post_id)、非空
report_reason	VARCHAR	512	举报原因	非空、最长存储 128 个中文字符
report_time	DATE		举报时间	非空
status	NUMBER(1)		处理状态	非空、Boolean: 0 - 待处理; 1 - 已处理

#### 4.2.16 帖子评论举报表 POST\_COMMENT\_REPORT

帖子评论举报表 (POST\_COMMENT\_REPORT) 旨在记录用户对帖子评论的举报情况，特别是针对违规或不当评论的反馈。该表包括举报 ID (post\_comment\_report\_id) 作为主键，以确保每条举报记录的唯一性；举报者 ID (reporter\_id) 和被举报者 ID (reported\_user\_id)，通过外键与用户表 (USER) 关联，明确指明举报和被举报的用户身份；被举报评论所属帖子 ID (reported\_post\_id) 和被举报评论 ID (reported\_comment\_id)，通过外键与帖子表 (POST) 和帖子评论表 (POST\_COMMENT) 关联，精确指向被举报的具体评论。(表 4.16)

表 4.16 帖子评论举报表 POST\_COMMENT\_REPORT

字段名	数据类型	长度	说明	备注
post_comment_report_id	INT		帖子评论举报 ID	PK、非空

续下页

续表 4.16

字段名	数据类型	长度	说明	备注
reporter_id	INT		举报者 ID	FK to USER(user_id)、非空
reported_user_id	INT		被举报者 ID	FK to USER(user_id)、非空
reported_post_id	INT		被举报评论所属帖子 ID	FK to POST(post_id)、非空
reported_comment_id	INT		被举报评论 ID	FK to POST_COMMENT(comment_id)、非空
report_reason	VARCHAR	512	举报原因	非空、最长存储 128 个中文字符
report_time	DATE		举报时间	非空
status	NUMBER(1)		处理状态	非空、Boolean: 0 - 待处理; 1 - 已处理

#### 4.2.17 新闻标签表 NEWS\_TAG

新闻标签表 (NEWS\_TAG) 是用来管理和分类新闻内容的关键数据表，它通过标签系统来组织新闻文章，便于用户浏览和搜索特定主题的新闻。该表包含两个主要字段：新闻标签 ID (tag\_id)，作为主键确保每个标签的唯一性；以及新闻标签 (tag)，存储标签的文本描述，这是一个简短的字符字段，用于标记新闻内容的主题或类别。此字段设置为唯一，确保不会有重复的标签存在。通过这种方式，新闻标签表支持新闻管理系统的高效运作，使得新闻内容能够按照特定的主题或兴趣点被有效地分类和检索，增强了用户体验并提高了内容的可访问性。(表 4.17)

表 4.17 新闻标签表 NEWS\_TAG

字段名	数据类型	长度	说明	备注
tag_id	INT		新闻标签 ID	PK、非空
tag	VARCHAR	64	新闻标签	非空、唯一、最长存储 16 个中文字符

#### 4.2.18 新闻表 NEWS

新闻表 (NEWS) 是设计用来存储和管理社交平台上发布的新闻内容的数据表。此表包括新闻 ID (news\_id)，作为主键确保每篇新闻的唯一性；管理员 ID (user\_id)，通过外键与用户表 (USER) 连接，标识新闻的发布者；新闻标签 ID (tag\_id)，通过外键与新闻标签表 (NEWS\_TAG) 连接，归类新闻主题。新闻的基本属性如标题 (title)、摘要 (summary)、详细内容 (content)、创建和更新时间，以及封面图片链接 (cover\_url) 都被记录在表中。(表 4.18)

表 4.18 新闻表 NEWS

字段名	数据类型	长度	说明	备注
news_id	INT		新闻 ID	PK、非空

续下页

续表 4.18

字段名	数据类型	长度	说明	备注
user_id	INT		管理员 ID	FK to USER(user_id)、非空
tag_id	INT		新闻标签 ID	FK to NEWS_TAG(tag_id)、非空
title	VARCHAR	256	标题	非空、最长存储 64 个中文字符
summary	VARCHAR	512	新闻摘要	非空、最长存储 128 个中文字符
content	CLOB		内容	非空
creation_date	DATE		创建时间	非空
update_date	DATE		更新时间	非空
cover_url	VARCHAR	2048	封面图片链接	非空
is_sticky	NUMBER(1)		是否置顶	非空、Boolean: 0 - 否; 1 - 是
like_count	INT		点赞数	非空
dislike_count	INT		点踩数	非空
favorite_count	INT		收藏数	非空
comment_count	INT		评论数	非空

#### 4.2.19 新闻评论表 NEWS\_COMMENT

新闻评论表 (NEWS\_COMMENT) 是设计用来存储用户对新闻文章的评论及其相关互动的数据表。此表包括多个字段，其中新闻评论 ID (comment\_id) 作为主键确保每条评论的唯一性；新闻 ID (news\_id) 和用户 ID (user\_id) 分别通过外键与新闻表 (NEWS) 和用户表 (USER) 关联，指明评论针对的具体新闻和发表评论的用户；父评论 ID (parent\_comment\_id) 允许评论具有层级结构，即回复其他评论。(表 4.19)

表 4.19 新闻评论表 NEWS\_COMMENT

字段名	数据类型	长度	说明	备注
comment_id	INT		新闻评论 ID	PK、非空
news_id	INT		新闻 ID	FK to NEWS(news_id)、非空
user_id	INT		用户 ID	FK to USER(user_id)、非空
parent_comment_id	INT		父评论 ID	FK to NEWS_COMMENT(comment_id)
content	VARCHAR	512	评论内容	非空、最长存储 128 个中文字符
comment_time	DATE		评论时间	非空
like_count	INT		点赞数	非空
dislike_count	INT		点踩数	非空

#### 4.2.20 新闻点赞表 NEWS\_LIKE

新闻点赞表 (NEWS\_LIKE) 是用来记录用户对新闻内容表达赞同或支持的行为的数据表。该表包含三个主要字段：新闻 ID (news\_id)、用户 ID (user\_id) 和点赞时间 (like\_time)。新闻 ID 和用户 ID 共同作为复合主键，并且通过外键分别与新闻表 (NEWS) 和用户表 (USER) 关联，这确保了每个点赞行为都能精确地关联到特定的新闻和特定的用户。点赞时间字段记录了用户进行点赞的具体时刻。这种记录机制有助于增强新闻互动性，同时帮助平台优化内容推送。(表 4.20)

表 4.20 新闻点赞表 NEWS\_LIKE

字段名	数据类型	长度	说明	备注
news_id	INT		新闻 ID	PK、FK to NEWS(news_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
like_time	DATE		点赞时间	非空

#### 4.2.21 新闻点踩表 NEWS\_DISLIKE

新闻点踩表 (NEWS\_DISLIKE) 用于记录用户对新闻内容的不赞同或反对的行为。这张表包含三个主要字段：新闻 ID (news\_id)、用户 ID (user\_id) 和点踩时间 (dislike\_time)。新闻 ID 和用户 ID 作为复合主键，并通过外键与新闻表 (NEWS) 和用户表 (USER) 关联，确保每个点踩行为都与特定的新闻和用户直接关联。点踩时间字段记录了用户进行点踩的具体时刻。(表 4.21)

表 4.21 新闻点踩表 NEWS\_DISLIKE

字段名	数据类型	长度	说明	备注
news_id	INT		新闻 ID	PK、FK to NEWS(news_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
dislike_time	DATE		点踩时间	非空

#### 4.2.22 新闻评论点赞表 NEWS\_COMMENT\_LIKE

新闻评论点赞表 (NEWS\_COMMENT\_LIKE) 是专门用于记录用户对新闻评论的赞同行为的数据表。此表包含三个核心字段：评论 ID (comment\_id)、用户 ID (user\_id)、和点赞时间 (like\_time)。评论 ID 和用户 ID 共同作为复合主键，与新闻评论表 (NEWS\_COMMENT) 和用户表 (USER) 关联。(表 4.22)

表 4.22 新闻评论点赞表 NEWS\_COMMENT\_LIKE

字段名	数据类型	长度	说明	备注
comment_id	INT		新闻评论 ID	PK、FK to NEWS_COMMENT(comment_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
like_time	DATE		点赞时间	非空

#### 4.2.23 新闻评论点踩表 NEWS\_COMMENT\_DISLIKE

新闻评论点踩表 (NEWS\_COMMENT\_DISLIKE) 旨在记录用户对新闻评论的不赞同或反对行为的详细数据。该表包含三个主要字段：评论 ID (comment\_id)、用户 ID (user\_id) 和点踩时间 (dislike\_time)。评论 ID 和用户 ID 作为复合主键，并且通过外键分别与新闻评论表 (NEWS\_COMMENT) 和用户表 (USER) 关联，确保每条点踩记录准确关联到特定的评论和用户。(表 4.23)

表 4.23 新闻评论点踩表 NEWS\_COMMENT\_DISLIKE

字段名	数据类型	长度	说明	备注
comment_id	INT		新闻评论 ID	PK、FK to NEWS_COMMENT(comment_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
dislike_time	DATE		点踩时间	非空

#### 4.2.24 新闻收藏表 NEWS\_FAVORITE

新闻收藏表 (NEWS\_FAVORITE) 是设计用来记录用户对新闻文章的收藏行为的数据表。该表使用户能够保存他们感兴趣的新闻内容以便未来查阅。包含三个主要字段：新闻 ID (news\_id) 和用户 ID (user\_id)，这两者共同构成复合主键并通过外键分别与新闻表 (NEWS) 和用户表 (USER) 连接；收藏时间 (favorite\_time) 记录了用户收藏新闻的具体时刻。(表 4.24)

表 4.24 新闻收藏表 NEWS\_FAVORITE

字段名	数据类型	长度	说明	备注
news_id	INT		新闻 ID	PK、FK to NEWS(news_id)、非空
user_id	INT		用户 ID	PK、FK to USER(user_id)、非空
favorite_time	DATE		收藏时间	非空

#### 4.2.25 新闻评论举报表 NEWS\_COMMENT\_REPORT

新闻评论举报表 (NEWS\_COMMENT\_REPORT) 用于记录用户对不当或不适宜的新闻评论的举报行为。该表的设计包含多个字段，主要包括新闻评论举报 ID (news\_comment\_report\_id) 作为主键，

确保每条举报记录的唯一性。举报者 ID (reporter\_id) 和被举报者 ID (reported\_user\_id), 这两者通过外键与用户表 (USER) 连接, 明确举报和被举报的用户; 被举报评论所属的新闻 ID (reported\_news\_id) 和具体的被举报评论 ID (reported\_comment\_id), 通过外键分别与新闻表 (NEWS) 和新闻评论表 (NEWS\_COMMENT) 连接, 精确标识被举报的评论。此外, 举报原因 (report\_reason)、举报时间 (report\_time) 和处理状态 (status) 字段用于详细记录举报的理由、时间和处理进度。(表 4.25)

表 4.25 新闻评论举报表 NEWS\_COMMENT\_REPORT

字段名	数据类型	长度	说明	备注
news_comment_report_id	INT		新闻评论举报 ID	PK、非空
reporter_id	INT		举报者 ID	FK to USER(user_id)、非空
reported_user_id	INT		被举报者 ID	FK to USER(user_id)、非空
reported_news_id	INT		被举报评论所属新闻 ID	FK to NEWS(news_id)、非空
reported_comment_id	INT		被举报评论 ID	FK to NEWS_COMMENT(comment_id)、非空
report_reason	VARCHAR	512	举报原因	非空、最长存储 128 个中文字符
report_time	DATE		举报时间	非空
status	NUMBER(1)		处理状态	非空、Boolean: 0 - 待处理; 1 - 已处理

4.2.26 宠物分类表 PET\_CATEGORY

宠物分类表 (PET\_CATEGORY) 是用来管理不同宠物类别信息的基础数据表。这张表包含四个字段: 宠物分类 ID (category\_id), 作为主键确保每个分类的唯一性; 宠物分类名称 (category\_name), 存储每个分类的名称, 设置为唯一以避免重复; 描述 (description), 提供关于宠物分类的详细信息; 图片链接 (image\_url), 展示相关分类的图像。这个表的设计旨在帮助用户更好地了解不同的宠物类别, 通过图文结合的方式提供直观的分类信息, 便于用户选择感兴趣的宠物类型。(表 4.26)

表 4.26 宠物分类表 PET\_CATEGORY

字段名	数据类型	长度	说明	备注
category_id	INT		宠物分类 ID	PK、非空
category_name_zh	VARCHAR	1024	宠物分类名称 (汉语)	非空、唯一、最长存储 16 个中文字符
category_name_de	VARCHAR	1024	宠物分类名称 (德语)	非空、唯一
category_name_en	VARCHAR	1024	宠物分类名称 (英语)	非空、唯一
category_name_es	VARCHAR	1024	宠物分类名称 (西班牙语)	非空、唯一
category_name_fr	VARCHAR	1024	宠物分类名称 (法语)	非空、唯一
category_name_it	VARCHAR	1024	宠物分类名称 (意大利语)	非空、唯一
category_name_ja	VARCHAR	1024	宠物分类名称 (日语)	非空、唯一
category_name_ko	VARCHAR	1024	宠物分类名称 (韩语)	非空、唯一

续下页

续表 4.26

字段名	数据类型	长度	说明	备注
category_name_pt	VARCHAR	1024	宠物分类名称（葡萄牙语）	非空、唯一
category_name_ru	VARCHAR	1024	宠物分类名称（俄语）	非空、唯一
description_zh	CLOB		描述（汉语）	非空、最长存储 512 个中文字符
description_de	CLOB		描述（德语）	非空
description_en	CLOB		描述（英语）	非空
description_es	CLOB		描述（西班牙语）	非空
description_fr	CLOB		描述（法语）	非空
description_it	CLOB		描述（意大利语）	非空
description_ja	CLOB		描述（日语）	非空
description_ko	CLOB		描述（韩语）	非空
description_pt	CLOB		描述（葡萄牙语）	非空
description_ru	CLOB		描述（俄语）	非空
image_url	VARCHAR	2048	图片链接	非空

4.2.27 宠物子类表 PET\_SUBCATEGORY

宠物子类表 (PET\_SUBCATEGORY) 用于详细划分宠物分类表中的每个主类别，提供更具体的宠物种类信息。该表包括多个字段：宠物子类 ID(subcategory\_id)作为主键；宠物分类 ID(category\_id)作为外键，连接到宠物分类表，指明每个子类属于哪个主分类；宠物子类名称 (subcategory\_name)，存储每个子类的名称，设置为唯一；以及描述 (description)，提供关于宠物子类的详细描述。这些信息帮助用户全面了解每种宠物的习性和养护需求，从而做出更合适的养宠选择。（表 4.27）

表 4.27 宠物子类表 PET\_SUBCATEGORY

字段名	数据类型	长度	说明	备注
subcategory_id	INT		宠物子类 ID	PK、非空
category_id	INT		宠物分类 ID	FK to PET_CATEGORY(category_id)、非空
subcategory_name_zh	VARCHAR	1024	宠物子类名称（汉语）	非空、唯一、最长存储 16 个中文字符
subcategory_name_de	VARCHAR	1024	宠物子类名称（德语）	非空、唯一
subcategory_name_en	VARCHAR	1024	宠物子类名称（英语）	非空、唯一
subcategory_name_es	VARCHAR	1024	宠物子类名称（西班牙语）	非空、唯一
subcategory_name_fr	VARCHAR	1024	宠物子类名称（法语）	非空、唯一
subcategory_name_it	VARCHAR	1024	宠物子类名称（意大利语）	非空、唯一
subcategory_name_ja	VARCHAR	1024	宠物子类名称（日语）	非空、唯一
subcategory_name_ko	VARCHAR	1024	宠物子类名称（韩语）	非空、唯一
subcategory_name_pt	VARCHAR	1024	宠物子类名称（葡萄牙语）	非空、唯一

续下页

续表 4.27

字段名	数据类型	长度	说明	备注
subcategory_name_ru	VARCHAR	1024	宠物子类名称（俄语）	非空、唯一
description_zh	CLOB		描述（汉语）	非空、最长存储 512 个中文字符
description_de	CLOB		描述（德语）	非空
description_en	CLOB		描述（英语）	非空
description_es	CLOB		描述（西班牙语）	非空
description_fr	CLOB		描述（法语）	非空
description_it	CLOB		描述（意大利语）	非空
description_ja	CLOB		描述（日语）	非空
description_ko	CLOB		描述（韩语）	非空
description_pt	CLOB		描述（葡萄牙语）	非空
description_ru	CLOB		描述（俄语）	非空
image_url	VARCHAR	2048	图片链接	非空
origin_zh	VARCHAR	512	起源地（汉语）	非空、最长存储 128 个中文字符
origin_de	VARCHAR	512	起源地（德语）	非空
origin_en	VARCHAR	512	起源地（英语）	非空
origin_es	VARCHAR	512	起源地（西班牙语）	非空
origin_fr	VARCHAR	512	起源地（法语）	非空
origin_it	VARCHAR	512	起源地（意大利语）	非空
origin_ja	VARCHAR	512	起源地（日语）	非空
origin_ko	VARCHAR	512	起源地（韩语）	非空
origin_pt	VARCHAR	512	起源地（葡萄牙语）	非空
origin_ru	VARCHAR	512	起源地（俄语）	非空
size_zh	VARCHAR	512	体型（汉语）	非空、最长存储 128 个中文字符
size_de	VARCHAR	512	体型（德语）	非空
size_en	VARCHAR	512	体型（英语）	非空
size_es	VARCHAR	512	体型（西班牙语）	非空
size_fr	VARCHAR	512	体型（法语）	非空
size_it	VARCHAR	512	体型（意大利语）	非空
size_ja	VARCHAR	512	体型（日语）	非空
size_ko	VARCHAR	512	体型（韩语）	非空
size_pt	VARCHAR	512	体型（葡萄牙语）	非空
size_ru	VARCHAR	512	体型（俄语）	非空
coat_zh	VARCHAR	512	毛色（汉语）	非空、最长存储 128 个中文字符
coat_de	VARCHAR	512	毛色（德语）	非空
coat_en	VARCHAR	512	毛色（英语）	非空
coat_es	VARCHAR	512	毛色（西班牙语）	非空

续下页



续表 4.27

字段名	数据类型	长度	说明	备注
coat_fr	VARCHAR	512	毛色（法语）	非空
coat_it	VARCHAR	512	毛色（意大利语）	非空
coat_ja	VARCHAR	512	毛色（日语）	非空
coat_ko	VARCHAR	512	毛色（韩语）	非空
coat_pt	VARCHAR	512	毛色（葡萄牙语）	非空
coat_ru	VARCHAR	512	毛色（俄语）	非空
lifespan_zh	VARCHAR	512	寿命（汉语）	非空、最长存储 128 个中文字符
lifespan_de	VARCHAR	512	寿命（德语）	非空
lifespan_en	VARCHAR	512	寿命（英语）	非空
lifespan_es	VARCHAR	512	寿命（西班牙语）	非空
lifespan_fr	VARCHAR	512	寿命（法语）	非空
lifespan_it	VARCHAR	512	寿命（意大利语）	非空
lifespan_ja	VARCHAR	512	寿命（日语）	非空
lifespan_ko	VARCHAR	512	寿命（韩语）	非空
lifespan_pt	VARCHAR	512	寿命（葡萄牙语）	非空
lifespan_ru	VARCHAR	512	寿命（俄语）	非空
temperament_zh	VARCHAR	512	性情（汉语）	非空、最长存储 128 个中文字符
temperament_de	VARCHAR	512	性情（德语）	非空
temperament_en	VARCHAR	512	性情（英语）	非空
temperament_es	VARCHAR	512	性情（西班牙语）	非空
temperament_fr	VARCHAR	512	性情（法语）	非空
temperament_it	VARCHAR	512	性情（意大利语）	非空
temperament_ja	VARCHAR	512	性情（日语）	非空
temperament_ko	VARCHAR	512	性情（韩语）	非空
temperament_pt	VARCHAR	512	性情（葡萄牙语）	非空
temperament_ru	VARCHAR	512	性情（俄语）	非空
diet_zh	VARCHAR	512	饮食习惯（汉语）	非空、最长存储 128 个中文字符
diet_de	VARCHAR	512	饮食习惯（德语）	非空
diet_en	VARCHAR	512	饮食习惯（英语）	非空
diet_es	VARCHAR	512	饮食习惯（西班牙语）	非空
diet_fr	VARCHAR	512	饮食习惯（法语）	非空
diet_it	VARCHAR	512	饮食习惯（意大利语）	非空
diet_ja	VARCHAR	512	饮食习惯（日语）	非空
diet_ko	VARCHAR	512	饮食习惯（韩语）	非空
diet_pt	VARCHAR	512	饮食习惯（葡萄牙语）	非空
diet_ru	VARCHAR	512	饮食习惯（俄语）	非空

续下页

续表 4.27

字段名	数据类型	长度	说明	备注
latitude_and_longitude	VARCHAR	512	经纬度参数	非空

4.2.28 宠物护理指导表 PET\_CARE\_GUIDE

宠物护理指导表 (PET\_CARE\_GUIDE) 是设计用来提供宠物护理和养护相关指导的数据表，目的是帮助宠物主人更好地了解和照顾他们的宠物。此表包括宠物护理指导 ID (guide\_id)，作为主键确保每条指导信息的唯一性；宠物子类 ID (subcategory\_id)，通过外键与宠物子类表 (PET\_SUBCATEGORY) 连接，指定指导信息适用于哪种具体的宠物子类；标题 (title)，提供指导信息的简短描述；以及内容 (content)，详细记录护理和养护的步骤或建议。这种信息布局旨在为宠物主人提供易于理解和执行的护理知识，从而改善宠物的生活质量和健康状况。(表 4.28)

表 4.28 宠物护理指导表 PET\_CARE\_GUIDE

字段名	数据类型	长度	说明	备注
guide_id	INT		宠物护理指导 ID	PK、非空
subcategory_id	INT		宠物子类 ID	FK to PET_SUBCATEGORY(subcategory_id)、非空
title_zh	VARCHAR	2048	标题 (汉语)	非空、最长存储 64 个中文字符
title_de	VARCHAR	2048	标题 (德语)	非空
title_en	VARCHAR	2048	标题 (英语)	非空
title_es	VARCHAR	2048	标题 (西班牙语)	非空
title_fr	VARCHAR	2048	标题 (法语)	非空
title_it	VARCHAR	2048	标题 (意大利语)	非空
title_ja	VARCHAR	2048	标题 (日语)	非空
title_ko	VARCHAR	2048	标题 (韩语)	非空
title_pt	VARCHAR	2048	标题 (葡萄牙语)	非空
title_ru	VARCHAR	2048	标题 (俄语)	非空
content_zh	CLOB		内容 (汉语)	非空、最长存储 512 个中文字符
content_de	CLOB		内容 (德语)	非空
content_en	CLOB		内容 (英语)	非空
content_es	CLOB		内容 (西班牙语)	非空
content_fr	CLOB		内容 (法语)	非空
content_it	CLOB		内容 (意大利语)	非空
content_ja	CLOB		内容 (日语)	非空
content_ko	CLOB		内容 (韩语)	非空
content_pt	CLOB		内容 (葡萄牙语)	非空

续下页

字段名	数据类型	长度	说明	备注
content_ru	CLOB		内容（俄语）	非空

宠物领养表 (PET\_ADOPTION) 旨在记录和管理关于宠物领养的详细信息，以支持用户对宠物的领养过程。此表包括多个字段，其中宠物领养 ID (adoption\_id) 作为主键确保记录的唯一性；用户 ID (user\_id)、宠物分类 ID (category\_id) 和宠物子类 ID (subcategory\_id) 通过外键关联到用户表 and 宠物分类及子类表，指明领养宠物的具体分类和用户信息。表中还包含宠物的基本信息如名称 (name)、年龄 (age)、图片链接 (image\_url) 以及所在地点 (location)。此外，详细记录了宠物的健康状况 (health)、最近一次健康检查日期 (latest\_health\_check)、疫苗接种情况 (vaccination)、特殊护理和饮食需求 (care\_needs, dietary\_needs)、性格行为特征 (behavior) 以及是否已绝育 (neutered\_or\_spayed)。领养状态 (status) 字段标识宠物是否已被领养，帮助潜在领养者了解宠物的当前状态。(表 4.29)

字段名	数据类型	长度	说明	备注
adoption_id	INT		宠物领养 ID	PK、非空
user_id	INT		用户 ID	FK to USER(user_id)、非空
category_id	INT		宠物分类 ID	FK to PET_CATEGORY(category_id)、非空
subcategory_id	INT		宠物子类 ID	FK to PET_SUBCATEGORY(subcategory_id)、非空
name	VARCHAR	64	宠物名称	最长存储 16 个中文字符
image_url	VARCHAR	2048	图片链接	非空
age	INT		宠物年龄	非空
location	VARCHAR	1024	地址	非空、最长存储 256 个中文字符
reason	VARCHAR	1024	转让原因	非空、最长存储 256 个中文字符
health	VARCHAR	1024	健康情况	非空、最长存储 256 个中文字符
latest_health_check	DATE		最近一次健康检查日期	非空
vaccination	VARCHAR	1024	疫苗接种情况	非空、最长存储 256 个中文字符
care_needs	VARCHAR	1024	特殊护理需求	最长存储 256 个中文字符
dietary_needs	VARCHAR	1024	特殊饮食需求	最长存储 256 个中文字符
behavior	VARCHAR	1024	性格行为特征	最长存储 256 个中文字符
neutered_or_spayed	NUMBER(1)		是否已绝育	非空、Boolean: 0 - 未绝育; 1 - 已绝育
notes	VARCHAR	1024	备注	最长存储 256 个中文字符
status	NUMBER(1)		领养状态	非空、Boolean: 0 - 待领养; 1 - 已领养
gender	NUMBER(1)		性别	非空、Boolean: 0 - 雄性; 1 - 雌性

共 91 页 第 44 页

续表 4.29

字段名	数据类型	长度	说明	备注
appendix_url	VARCHAR	2048	附件链接	

### 4.2.30 开发团队表 DEVELOPMENT\_TEAM

开发团队表（DEVELOPMENT\_TEAM）是用于记录和管理软件开发团队成员详细信息的数据表，旨在提供一个系统化的方式来存储团队成员的关键信息。该表包括多个字段，如学号（id）作为主键，确保每位成员的唯一标识；姓名（name）、学校（school）、电子邮箱（email）为基本联系信息；以及 GitHub 名称（github\_name）和 GitHub 主页链接（github\_profile）。（表 4.30）

表 4.30 开发团队表 DEVELOPMENT\_TEAM

字段名	数据类型	长度	说明	备注
id	INT		学号	PK、非空
name	VARCHAR	64	姓名	非空、最长存储 16 个中文字符
school	VARCHAR	64	学校	非空、最长存储 16 个中文字符
email	VARCHAR	2048	电子邮箱	非空
github_name	VARCHAR	64	GitHub 名称	非空、最长存储 16 个中文字符
github_profile	VARCHAR	2048	GitHub 主页	非空

## 4.3 数据库设计

### 4.3.1 规范化

数据库规范化是设计数据库模式的过程，旨在减少数据冗余和提高数据完整性。规范化通常分为几个范式，其中最常用的是第一范式（1NF）、第二范式（2NF）和第三范式（3NF）。

#### 4.3.1.1 第一范式（1NF）

第一范式（1NF）要求表中的每个字段都是不可分割的基本数据项，即字段不可再分成其他几个字段。此外，表中的所有记录都必须是唯一的，不能有重复的记录。每个表都应有一个唯一标识，即主键。

在本数据库设计中，每个表的字段如 user\_id, post\_id, news\_id 等都是单一属性，符合 1NF 要求。

#### 4.3.1.2 第二范式（2NF）

第二范式（2NF）基于第一范式，要求表必须只依赖于主键。也就是说，非主键字段必须完全依赖于整个主键，而不是依赖于主键的一部分（如果主键是由多个字段组成的）。

在本数据库设计中，多字段主键的表如 POST\_LIKE(post\_id,user\_id)、USER\_FOLLOW(user\_id,follower\_id) 中的每个属性都完全依赖于整个复合主键，满足 2NF。

### 4.3.1.3 第三范式 (3NF)

第三范式 (3NF) 进一步要求表中的每个非主键字段不仅完全依赖于主键，而且只直接依赖于主键 (消除传递依赖)。也就是说，所有的非主键字段必须直接依赖于主键，不能依赖于其他非主键字段。

本数据库设计符合 3NF 要求。例如：USER 表的每个字段如 email, password, registration\_date 等直接依赖于 user\_id。

### 4.3.2 主键和索引

主键 (Primary Key) 是表中的一个字段或多个字段的组合，其值唯一标识表中的每一行记录。主键具有以下特点：

- 每个表只能有一个主键
- 主键列的值不能为空
- 主键列的值必须唯一

在本数据库设计中，每个表都有一个定义明确的主键。例如：

- USER 表的主键是 user\_id：

```
1 user_id INT not null
2   constraint USER_pk
3   primary key
```

- 对于复合主键，如 USER\_FOLLOW 表，其主键是由 user\_id 和 follower\_id 组成：

```
1 constraint USER_FOLLOW_pk
2   primary key (user_id, follower_id)
```

索引 (Index) 是数据库中用于加速数据检索的结构。通过创建索引，可以提高查询性能，特别是在处理大量数据时。索引类型包括唯一索引、非唯一索引和全文索引等。

在本数据库设计中，以下是一些适合创建索引的字段和原因：

- **唯一索引 (Unique Index)**：用于字段必须唯一的情况。数据库在创建主键时，通常会自动创建唯一索引。例如 user\_name 在 USER 表中是唯一的，因此创建唯一索引：

```
1 constraint user_name_uk unique (user_name)
```

- **非唯一索引 (Non-Unique Index)**：用于提高查询性能的字段。例如 user\_id 在 POST 表中用于查询用户的帖子，可以创建索引：

```
1 create index POST_user_id_idx on POST(user_id);
```

- **组合索引 (Composite Index)**：用于组合多个字段提高查询性能。例如 `user_id` 和 `follower_id` 在 `USER_FOLLOW` 表中是组合索引，用于查询用户与其关注者之间的关系：

```
1 create index USER_FOLLOW_user_follower_idx on USER_FOLLOW(user_id, follower_id);
```

### 4.3.3 外键

外键 (Foreign Key) 是一种约束，用于维护数据库表之间的关系和数据的一致性。外键确保在一个表中的值必须存在于另一个表的主键或唯一键中，从而维持数据的引用完整性。

外键的作用：

- **数据一致性**：确保子表中的外键值在父表中有对应的记录
- **维护表之间的关系**：通过外键建立表与表之间的关联，使得数据库结构更加清晰和规范
- **保证数据完整性**：防止无效数据的插入和删除操作对相关数据造成不一致的影响

在本数据库设计中，多个表使用外键来建立与 `USER` 表、`POST` 表和其他表的关系。例如：`USER_SETTING` 表中的 `user_id` 是 `USER` 表的外键。

```
1 user_id INT not null
2 constraint USER_SETTING_USER_fk
3 references "USER"(user_id)
```

### 4.3.4 数据类型和约束

在数据库设计中，选择合适的数据类型和应用适当的约束是至关重要的。数据类型决定了字段存储的数据类型，而约束则用于限制数据输入，以确保数据的一致性和完整性。

数据类型定义了字段能够存储的数据类型和大小。在本数据库设计中，主要使用了以下数据类型：

- **INT**：整数类型，适用于用户 ID、帖子 ID 等唯一标识
- **VARCHAR(n)**：变长字符串类型，适用于存储文本数据，如用户名、电子邮件等
- **DATE**：日期类型，适用于存储日期和时间信息，如注册日期、最后登录时间等
- **NUMBER(n)**：数字类型，适用于存储状态、角色等枚举值
- **CLOB**：字符大型对象类型，适用于存储大文本数据，如新闻内容

约束用于限制数据输入，确保数据的一致性和完整性。在本数据库设计中，主要使用了以下约束：

- **非空约束 (Not Null)**：确保字段不能为空
- **唯一约束 (Unique)**：确保字段中的所有值是唯一的
- **主键约束 (Primary Key)**：唯一标识表中的每一行记录
- **外键约束 (Foreign Key)**：确保一个表中的值在另一个表中存在
- **检查约束 (Check)**：用于确保字段中的值满足特定的条件

注释用于描述表和字段的含义，便于开发和维护。在本数据库设计中，主要使用了表注释和字段注释。

#### 4.3.5 安全性和隐私

在数据库设计中，安全性和隐私保护是关键因素。保护用户的敏感信息和确保数据的完整性至关重要。在本数据库设计中，使用 **SHA-256** 加密算法对密码进行哈希处理，可以增强密码的安全性。

**SHA-256** (Secure Hash Algorithm 256) 是 **SHA-2** 家族中的一种哈希函数。它生成一个 256 位 (32 字节) 的哈希值，通常表示为 64 位十六进制字符串。**SHA-256** 的主要特点是：

- **不可逆**：无法从哈希值反推出原始数据
- **雪崩效应**：输入的微小变化会导致输出的哈希值大幅改变
- **抗碰撞**：难以找到两个不同的输入产生相同的哈希值

密码哈希在应用程序层完成，然后将哈希值存储在数据库中。在数据库表中存储经过 **SHA-256** 哈希处理后的密码。由于 **SHA-256** 哈希值是一个 64 位的十六进制字符串，因此密码字段的长度设置为 64。

在用户登录时，需要验证输入的密码是否正确。这是通过将输入的密码进行 **SHA-256** 哈希处理，并与数据库中存储的哈希值进行比较来实现的。通过上述方法，可以确保用户密码在数据库中的安全性和隐私保护。在数据库中仅存储经过 **SHA-256** 哈希处理后的密码，避免了明文存储密码的安全风险。

#### 4.3.6 性能优化

在数据库设计和管理中，性能优化是确保数据库高效运行的关键步骤。优化的目标是减少查询时间、提高数据处理速度和节省存储空间。下面是本数据库的一些性能优化方法：

- **创建索引**：为高频查询的字段创建索引，如主键、外键以及常用的查询条件字段
- **查询优化**：编写高效的 **SQL** 查询是数据库性能优化的另一个重要方面
- **数据库设计优化**：合理的数据库设计可以显著提高性能
- **缓存机制**：通过缓存机制，可以减少数据库的直接访问，提升系统整体性能
- **定期维护和监控**：定期对数据库进行维护和监控，可以预防性能问题

### 4.3.7 数据持久化

数据持久化是确保数据库在服务器因故障重启或数据库重新挂载时不会丢失数据的重要措施。本数据库在 Ubuntu 服务器上通过 Docker 实现了数据持久化，从而保证了数据的可靠性和持久性。

#### 4.3.7.1 部署阿里云轻量应用服务器

轻量应用服务器（Simple Application Server），是可快速搭建且易于管理的轻量级云服务器；提供基于单台服务器的应用部署，安全管理，运维监控等服务，一站式提升您的服务器使用体验和效率。服务器版本信息见图 4.2，服务器防火墙规则见 tab:FirewallRules。

服务器配置详情：

- 实例 ID：dfc3f736506748c3ba9ced56b94a81d7
- 实例名称：轻量应用服务器
- 地域：中国香港
- 配置信息：2 核 vCPU | 4GB 内存 | 3072GB 每月流量 | 80GB ESSD | 30Mbps 限峰值带宽
- 镜像信息：Ubuntu 22.04
- 私有 IP 地址：172.19.54.72
- 公有 IP 地址：47.238.195.140
- 到期时间：2024 年 10 月 8 日 00:00:00
- 配置费用：¥ 102.00（优惠金额 ¥ 300.00）

表 4.31 防火墙规则

协议	端口范围	限制 IP 来源	备注
TCP	80	0.0.0.0/0	HTTP 默认端口
TCP	443	0.0.0.0/0	HTTPS 默认端口
TCP	1521	0.0.0.0/0	PetJoy Oracle Database 端口
TCP	5101	0.0.0.0/0	PetJoy DatabaseWebAPI 端口
TCP	22	0.0.0.0/0	SSH 远程服务默认端口

#### 4.3.7.2 部署 Docker 环境

部署 Docker 环境的流程如下：

##### 1. 设置 Docker 的 apt 仓库

```
1 # Add Docker's official GPG key:
2 sudo apt-get update
3 sudo apt-get install ca-certificates curl
4 sudo install -m 0755 -d /etc/apt/keyrings
5 sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
  ↪ /etc/apt/keyrings/docker.asc
```



```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-86-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Mon Apr  8 09:55:30 PM CST 2024

System load:  0.2744140625      Processes:            121
Usage of /:   3.1% of 78.37GB   Users logged in:     0
Memory usage: 6%               IPv4 address for eth0: 172.19.54.72
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

图 4.2 服务器版本信息

```
6  sudo chmod a+r /etc/apt/keyrings/docker.asc
7
8  # Add the repository to Apt sources:
9  echo \
10     "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
    ↪ https://download.docker.com/linux/ubuntu \
11     $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
12     sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
13  sudo apt-get update
```

## 2. 安装 Docker

```
1  sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
    ↪ docker-compose-plugin
```

## 3. 查看 Docker 版本，验证 Docker 安装是否成功（图 4.3、图 4.4）

```
1  docker version
```

### 4.3.7.3 数据持久化方式部署 Oracle 12c 数据库

数据持久化方式部署 Oracle 12c 数据库的流程如下：

#### 1. 下载镜像文件

```
Client: Docker Engine - Community
Version:      26.0.0
API version:  1.45
Go version:   go1.21.8
Git commit:   2ae903e
Built:        Wed Mar 20 15:17:48 2024
OS/Arch:      linux/amd64
Context:      default
```

图 4.3 客户端 Docker 版本信息

```
Server: Docker Engine - Community
Engine:
Version:      26.0.0
API version:  1.45 (minimum version 1.24)
Go version:   go1.21.8
Git commit:   8b79278
Built:        Wed Mar 20 15:17:48 2024
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.6.28
GitCommit:    ae07eda36dd25f8a1b98dfbf587313b99c0190bb
runc:
Version:      1.1.12
GitCommit:    v1.1.12-0-g51d5e94
docker-init:
Version:      0.19.0
GitCommit:    de40ad0
```

图 4.4 服务端 Docker 版本信息

```
1 docker pull absolutapps/oracle-12c-ee:latest
```

## 2. 查看镜像文件

```
1 docker images
```

## 3. 创建主机内路径

```
1 mkdir -p /home/local/oracle-data
```

## 4. 在 Docker 容器内运行 Oracle 12c 数据库

```
1 docker run -d --name petjoy-oracle-database --privileged -v
↪ /home/local/oracle-data:/u01/app/oracle -p 8080:8080 -p 1521:1521
↪ absolutapps/oracle-12c-ee
```

## 5. 查看容器启动日志 (图 4.5)

```
1 docker logs -f petjoy-oracle-database
```

## 6. 查看容器状态

```
1 docker ps
```

```
Look at the log file "/u01/app/oracle/cfgtoollogs/dbca/orcl/orcl.log" for further details.

LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 08-APR-2024 14:42:28

Copyright (c) 1991, 2014, Oracle. All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=5393b6f1bb0a)(PORT=1521)))
The command completed successfully
Database has been created in /u01/app/oracle/oradata/orcl
SYS and SYSTEM passwords are set to [oracle]
Setting HTTP port to 8080

PL/SQL procedure successfully completed.

Please login to http://<ip_address>:8080/em to use enterprise manager
User: sys; Password oracle; Sysdba: true
Fixing permissions...
Running init scripts...
Init scripts in /oracle.init.d/: Ignoring /oracle.init.d/*
Done with scripts we are ready to go
```

图 4.5 容器启动日志

### 4.3.7.4 测试 Oracle 12c 数据库

测试 Oracle 12c 数据库的流程如下:

## 1. 进入容器

```
1 docker exec -it petjoy-oracle-database /bin/bash
```

## 2. 切换数据库用户

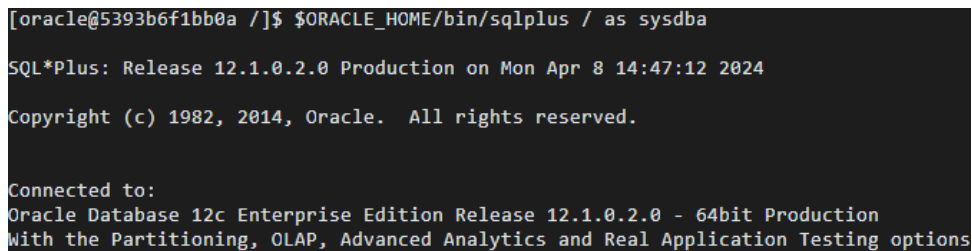
```
1 su oracle
```

## 3. 进入数据库（图 4.6）

```
1 $ORACLE_HOME/bin/sqlplus / as sysdba
```

## 4. 查询数据库状态

```
1 select status from v$instance;
```



```
[oracle@5393b6f1bb0a /]$ $ORACLE_HOME/bin/sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0 Production on Mon Apr 8 14:47:12 2024
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
```

图 4.6 进入数据库

### 4.3.7.5 添加 Oracle 12c 数据库用户

添加 Oracle 12c 数据库用户的流程如下：

#### 1. 连接到数据库作为 SYSDBA

```
1 conn system/oracle as sysdba;
```

#### 2. 修改 system 用户的密码为 oracle

```
1 alter user system identified by oracle;
```

#### 3. 修改 sys 用户的密码为 sys

```
1 alter user sys identified by sys;
```

#### 4. 修改默认的密码策略，将密码的有效期限设置为无限

```
1 ALTER PROFILE DEFAULT LIMIT PASSWORD_LIFE_TIME UNLIMITED;
```

5. 创建一个新用户 tjsse\_dbteam, 并且设置其密码为 tjsse2024

```
1 create user tjsse_dbteam identified by tjsse2024;
```

6. 为 tjsse\_dbteam 用户授权

```
1 grant connect, resource, dba to tjsse_dbteam;
```

7. 查询数据库账户

```
1 SELECT username FROM dba_users;
```

#### 4.3.7.6 以 tjsse\_dbteam 用户身份连接数据库

可以通过轻量应用服务器和 JetBrains DataGrip 以 tjsse\_dbteam 用户身份连接数据库。

##### 4.3.7.6.1 通过轻量应用服务器连接数据库

通过轻量应用服务器连接数据库的流程如下:

1. 进入容器

```
1 docker exec -it petjoy-oracle-database /bin/bash
```

2. 切换数据库用户

```
1 su oracle
```

3. 进入数据库

```
1 $ORACLE_HOME/bin/sqlplus
```

4. 输入用户名和密码

##### 4.3.7.6.2 通过 JetBrains DataGrip 连接数据库

通过 JetBrains DataGrip 连接数据库 (图 4.7):

- 主机: 47.238.195.140
- 端口: 1521
- SID: orcl
- 驱动程序: Thin

装  
订  
线



图 4.7 通过 JetBrains DataGrip 连接数据库

- 身份验证：用户与密码
- 用户：tjsse\_dbteam
- 密码：tjsse2024
- 保存：永久
- URL：jdbc:oracle:thin:@47.238.195.140:1521:orcl

## 4.3.8 扩展性和可维护性

在数据库设计中，扩展性和可维护性是确保数据库系统能够适应不断变化的业务需求和数据增长的重要因素。良好的设计可以使数据库在处理更大的数据量、更高的并发请求以及不断变化的业务需求时，仍然保持高效、稳定的运行。

本数据库使用了模块化设计。模块化设计是指将数据库结构划分为多个独立的模块，每个模块负责特定的功能或业务。这种设计有助于简化数据库管理和维护。通过将不同业务模块分开设计，以便独立开发和维护。

## 4.4 数据定义语言（DDL）设计

对于数据库设计，确保有效的数据定义语言（DDL）实现是至关重要的。基于数据库设计的规范化、主键和索引、外键、数据类型和约束、安全性和隐私、性能优化、数据持久化、扩展性和可维护性等方面，我们对各种表进行了数据定义语言（DDL）的设计。

### 4.4.1 用户表 USER

下面是用户表 USER 的数据定义语言：

```

1 create table "USER"
2 (
3     user_id            INT            GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY
4         ↪ 1) not null
5         constraint USER_pk
6             primary key,
7     user_name          VARCHAR(64)    not null
8         constraint user_name_uk
9             unique,
10    password           VARCHAR(64)    not null,
11    email               VARCHAR(2048)
12        constraint email_uk
13            unique,
14    registration_date   DATE           not null,
15    last_login_time     DATE           not null,
16    role                NUMBER(1)      not null,
17    status              NUMBER(1)      not null,
18    avatar_url          VARCHAR(2048),
19    profile             VARCHAR(512),
20    gender              NUMBER(1)      not null,
21    birthdate           DATE           not null,
22    experience_points   INT            not null,

```

```

22     follows_count      INT          not null,
23     followed_count     INT          not null,
24     favorites_count     INT          not null,
25     favorited_count    INT          not null,
26     likes_count        INT          not null,
27     liked_count        INT          not null,
28     message_count      INT          not null
29 );
30
31 comment on table "USER" is '用户';
32 comment on column "USER".user_id is '用户 ID';
33 comment on column "USER".user_name is '用户名';
34 comment on column "USER".password is '密码';
35 comment on column "USER".telephone is '手机号码';
36 comment on column "USER".registration_date is '注册日期';
37 comment on column "USER".last_login_time is '上次登录时间';
38 comment on column "USER".role is '用户角色';
39 comment on column "USER".status is '用户状态';
40 comment on column "USER".avatar_url is '头像链接';
41 comment on column "USER".profile is '个人简介';
42 comment on column "USER".gender is '性别';
43 comment on column "USER".birthdate is '出生日期';
44 comment on column "USER".experience_points is '经验点数';
45 comment on column "USER".follows_count is '关注数';
46 comment on column "USER".followed_count is '被关注数';
47 comment on column "USER".favorites_count is '收藏数';
48 comment on column "USER".favorited_count is '被收藏数';
49 comment on column "USER".likes_count is '点赞数';
50 comment on column "USER".liked_count is '被点赞数';
51 comment on column "USER".message_count is '留言数';

```

## 4.4.2 用户设置表 USER\_SETTING

下面是用户设置表 USER\_SETTING 的数据定义语言:

```

1 create table USER_SETTING
2 (
3     user_id                INT          not null
4         constraint USER_SETTING_pk
5             primary key
6     constraint USER_SETTING_USER_fk
7         references "USER" ON DELETE CASCADE,
8     is_telephone_public    NUMBER(1) not null,
9     is_registration_date_public NUMBER(1) not null,
10    is_profile_public       NUMBER(1) not null,
11    is_gender_public        NUMBER(1) not null,
12    is_birthdate_public     NUMBER(1) not null,
13    is_level_public         NUMBER(1) not null,
14    is_following_count_public NUMBER(1) not null,
15    is_followers_count_public NUMBER(1) not null,
16    is_favorites_count_public NUMBER(1) not null,
17    is_favored_count_public NUMBER(1) not null,
18    is_likes_count_public   NUMBER(1) not null,
19    is_liked_count_public   NUMBER(1) not null,
20    is_message_count_public NUMBER(1) not null,

```



```

21     allow_following          NUMBER(1) not null,
22     receive_admin_notifications NUMBER(1) not null,
23     receive_user_notifications NUMBER(1) not null
24 );
25
26 comment on table USER_SETTING is '用户设置';
27 comment on column USER_SETTING.user_id is '用户 ID';
28 comment on column USER_SETTING.is_telephone_public is '是否公开手机号码';
29 comment on column USER_SETTING.is_registration_date_public is '是否公开注册日期';
30 comment on column USER_SETTING.is_profile_public is '是否公开个人简历';
31 comment on column USER_SETTING.is_gender_public is '是否公开性别';
32 comment on column USER_SETTING.is_birthdate_public is '是否公开出生日期';
33 comment on column USER_SETTING.is_level_public is '是否公开等级';
34 comment on column USER_SETTING.is_following_count_public is '是否公开关注数';
35 comment on column USER_SETTING.is_followers_count_public is '是否公开被关注数';
36 comment on column USER_SETTING.is_favorites_count_public is '是否公开收藏数';
37 comment on column USER_SETTING.is_favored_count_public is '是否公开被收藏数';
38 comment on column USER_SETTING.is_likes_count_public is '是否公开点赞数';
39 comment on column USER_SETTING.is_liked_count_public is '是否公开被点赞数';
40 comment on column USER_SETTING.is_message_count_public is '是否公开留言数';
41 comment on column USER_SETTING.allow_following is '是否允许他人关注';
42 comment on column USER_SETTING.receive_admin_notifications is '是否接受管理员通知';
43 comment on column USER_SETTING.receive_user_notifications is '是否接受用户通知';

```

#### 4.4.3 用户打卡表 USER\_CHECK\_IN

下面是用户打卡表 USER\_CHECK\_IN 的数据定义语言：

```

1 create table USER_CHECK_IN
2 (
3     check_in_id    INT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1) not null
4     constraint USER_CHECK_IN_pk
5         primary key,
6     user_id        INT not null
7     constraint USER_CHECK_IN_USER_fk
8         references "USER" ON DELETE CASCADE,
9     check_in_time  DATE not null
10 );
11
12 comment on table USER_CHECK_IN is '用户打卡';
13 comment on column USER_CHECK_IN.check_in_id is '打卡记录 ID';
14 comment on column USER_CHECK_IN.user_id is '用户 ID';
15 comment on column USER_CHECK_IN.check_in_time is '打卡时间';

```

#### 4.4.4 用户关注表 USER\_FOLLOW

下面是用户关注表 USER\_FOLLOW 的数据定义语言：

```

1 create table USER_FOLLOW
2 (
3     user_id        INT not null
4     constraint USER_FOLLOW_USER_fk_1

```

```

5         references "USER" ON DELETE CASCADE,
6     follower_id INT not null
7         constraint USER_FOLLOW_USER_fk_2
8         references "USER" ON DELETE CASCADE,
9     follow_time DATE not null,
10    constraint USER_FOLLOW_pk
11        primary key (user_id, follower_id)
12 );
13
14 comment on table USER_FOLLOW is '用户关注';
15 comment on column USER_FOLLOW.user_id is '被关注者 ID';
16 comment on column USER_FOLLOW.follower_id is '关注者 ID';
17 comment on column USER_FOLLOW.follow_time is '关注时间';

```

#### 4.4.5 用户留言表 USER\_MESSAGE

下面是用户留言表 USER\_MESSAGE 的数据定义语言:

```

1 create table USER_MESSAGE
2 (
3     message_id INT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1) not
4     null
5     constraint USER_MESSAGE_pk
6     primary key,
7     user_id INT not null
8     constraint USER_MESSAGE_USER_fk_1
9     references "USER" ON DELETE CASCADE,
10    commenter_id INT not null
11    constraint USER_MESSAGE_USER_fk_2
12    references "USER" ON DELETE CASCADE,
13    message VARCHAR(512) not null,
14    comment_time DATE not null
15 );
16
17 comment on table USER_MESSAGE is '用户留言';
18 comment on column USER_MESSAGE.message_id is '留言 ID';
19 comment on column USER_MESSAGE.user_id is '用户 ID';
20 comment on column USER_MESSAGE.commenter_id is '留言者 ID';
21 comment on column USER_MESSAGE.message is '留言内容';
22 comment on column USER_MESSAGE.comment_time is '留言时间';

```

#### 4.4.6 用户反馈表 USER\_FEEDBACK

下面是用户反馈表 USER\_FEEDBACK 的数据定义语言:

```

1 create table USER_FEEDBACK
2 (
3     feedback_id INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4     not null
5     constraint USER_FEEDBACK_pk
6     primary key,
7     feedback_category VARCHAR(32) not null,

```

```

7      feedback_content VARCHAR(2048) not null,
8      feedback_time   DATE           not null,
9      email           VARCHAR(2048),
10     telephone       VARCHAR(16)
11 );
12
13 comment on table USER_FEEDBACK is '用户反馈';
14 comment on column USER_FEEDBACK.feedback_id is '反馈意见 ID';
15 comment on column USER_FEEDBACK.feedback_category is '反馈类型';
16 comment on column USER_FEEDBACK.feedback_content is '反馈内容';
17 comment on column USER_FEEDBACK.feedback_time is '反馈时间';
18 comment on column USER_FEEDBACK.email is '电子邮箱';
19 comment on column USER_FEEDBACK.telephone is '手机号码';

```

#### 4.4.7 帖子分类表 POST\_CATEGORY

下面是帖子分类表 POST\_CATEGORY 的数据定义语言:

```

1 create table POST_CATEGORY
2 (
3     category_id INT           not null
4         constraint POST_CATEGORY_pk
5             primary key,
6     category   VARCHAR(64) not null
7         constraint post_category_uk
8             unique
9 );
10
11 comment on table POST_CATEGORY is '帖子分类';
12 comment on column POST_CATEGORY.category_id is '帖子分类 ID';
13 comment on column POST_CATEGORY.category is '帖子分类';

```

#### 4.4.8 帖子表 POST

下面是帖子表 POST 的数据定义语言:

```

1 create table POST
2 (
3     post_id          INT           GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4         not null
5         constraint POST_pk
6             primary key,
7     user_id          INT           not null
8         constraint POST_USER_fk
9             references "USER" ON DELETE CASCADE,
10    category_id       INT           not null
11        constraint POST_POST_CATEGORY_fk
12            references POST_CATEGORY (category_id) ON DELETE CASCADE,
13    title              VARCHAR(256) not null,
14    content            VARCHAR(2048) not null,
15    creation_date      DATE          not null,
16    update_date        DATE          not null,

```

```

16     is_sticky      NUMBER(1)      not null,
17     like_count    INT             not null,
18     dislike_count INT             not null,
19     favorite_count INT            not null,
20     comment_count INT            not null,
21     image_url     VARCHAR(2048)
22 );
23
24 comment on table POST is '帖子';
25 comment on column POST.post_id is '帖子 ID';
26 comment on column POST.user_id is '发帖用户 ID';
27 comment on column POST.category_id is '帖子分类 ID';
28 comment on column POST.title is '标题';
29 comment on column POST.content is '内容';
30 comment on column POST.creation_date is '创建时间';
31 comment on column POST.update_date is '更新时间';
32 comment on column POST.is_sticky is '是否置顶';
33 comment on column POST.like_count is '点赞数';
34 comment on column POST.dislike_count is '点踩数';
35 comment on column POST.favorite_count is '收藏数';
36 comment on column POST.comment_count is '评论数';
37 comment on column POST.image_url is '图片链接';

```

## 4.4.9 帖子评论表 POST\_COMMENT

下面是帖子评论表 POST\_COMMENT 的数据定义语言:

```

1 create table POST_COMMENT
2 (
3     comment_id      INT             GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4     ↪ not null
5     constraint POST_COMMENT_pk
6         primary key,
7     post_id         INT             not null
8     constraint POST_COMMENT_POST_fk
9         references POST ON DELETE CASCADE,
10    user_id          INT             not null
11    constraint POST_COMMENT_USER_fk
12        references "USER" ON DELETE CASCADE,
13    parent_comment_id INT
14    constraint POST_COMMENT_POST_COMMENT_fk
15        references POST_COMMENT ON DELETE CASCADE,
16    content          VARCHAR(512) not null,
17    comment_time     DATE             not null,
18    like_count       INT             not null,
19    dislike_count    INT             not null
20 );
21
22 comment on table POST_COMMENT is '帖子评论';
23 comment on column POST_COMMENT.comment_id is '帖子评论 ID';
24 comment on column POST_COMMENT.post_id is '帖子 ID';
25 comment on column POST_COMMENT.user_id is '用户 ID';
26 comment on column POST_COMMENT.parent_comment_id is '父评论 ID';
27 comment on column POST_COMMENT.content is '评论内容';
28 comment on column POST_COMMENT.comment_time is '评论时间';

```

```
28 comment on column POST_COMMENT.like_count is '点赞数';
29 comment on column POST_COMMENT.dislike_count is '点踩数';
```

#### 4.4.10 帖子点赞表 POST\_LIKE

下面是帖子点赞表 POST\_LIKE 的数据定义语言：

```
1 create table POST_LIKE
2 (
3     post_id    INT    not null
4         constraint POST_LIKE_POST_fk
5             references POST ON DELETE CASCADE,
6     user_id    INT    not null
7         constraint POST_LIKE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     like_time  DATE not null,
10    constraint POST_LIKE_pk
11        primary key (post_id, user_id)
12 );
13
14 comment on table POST_LIKE is '帖子点赞';
15 comment on column POST_LIKE.post_id is '帖子 ID';
16 comment on column POST_LIKE.user_id is '用户 ID';
17 comment on column POST_LIKE.like_time is '点赞时间';
```

#### 4.4.11 帖子点踩表 POST\_DISLIKE

下面是帖子点踩表 POST\_DISLIKE 的数据定义语言：

```
1 create table POST_DISLIKE
2 (
3     post_id    INT    not null
4         constraint POST_DISLIKE_POST_fk
5             references POST ON DELETE CASCADE,
6     user_id    INT    not null
7         constraint POST_DISLIKE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     dislike_time DATE not null,
10    constraint POST_DISLIKE_pk
11        primary key (post_id, user_id)
12 );
13
14 comment on table POST_DISLIKE is '帖子点踩';
15 comment on column POST_DISLIKE.post_id is '帖子 ID';
16 comment on column POST_DISLIKE.user_id is '用户 ID';
17 comment on column POST_DISLIKE.dislike_time is '点踩时间';
```

#### 4.4.12 帖子评论点赞表 POST\_COMMENT\_LIKE

下面是帖子评论点赞表 POST\_COMMENT\_LIKE 的数据定义语言：

```

1 create table POST_COMMENT_LIKE
2 (
3     comment_id INT not null
4         constraint POST_COMMENT_LPC_fk
5             references POST_COMMENT ON DELETE CASCADE,
6     user_id INT not null
7         constraint POST_COMMENT_LIKE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     like_time DATE not null,
10    constraint POST_COMMENT_LIKE_pk
11        primary key (comment_id, user_id)
12 );
13
14 comment on table POST_COMMENT_LIKE is '帖子评论点赞';
15 comment on column POST_COMMENT_LIKE.comment_id is '帖子评论 ID';
16 comment on column POST_COMMENT_LIKE.user_id is '用户 ID';
17 comment on column POST_COMMENT_LIKE.like_time is '点赞时间';

```

#### 4.4.13 帖子评论点踩表 POST\_COMMENT\_DISLIKE

下面是帖子评论点踩表 POST\_COMMENT\_DISLIKE 的数据定义语言:

```

1 create table POST_COMMENT_DISLIKE
2 (
3     comment_id INT not null
4         constraint POST_COMMENT_DPC_fk
5             references POST_COMMENT ON DELETE CASCADE,
6     user_id INT not null
7         constraint POST_COMMENT_DISLIKE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     dislike_time DATE not null,
10    constraint POST_COMMENT_DISLIKE_pk
11        primary key (comment_id, user_id)
12 );
13
14 comment on table POST_COMMENT_DISLIKE is '帖子评论点踩';
15 comment on column POST_COMMENT_DISLIKE.comment_id is '帖子评论 ID';
16 comment on column POST_COMMENT_DISLIKE.user_id is '用户 ID';
17 comment on column POST_COMMENT_DISLIKE.dislike_time is '点踩时间';

```

#### 4.4.14 帖子收藏表 POST\_FAVORITE

下面是帖子收藏表 POST\_FAVORITE 的数据定义语言:

```

1 create table POST_FAVORITE
2 (
3     post_id INT not null
4         constraint POST_FAVORITE_POST_fk
5             references POST ON DELETE CASCADE,
6     user_id INT not null
7         constraint POST_FAVORITE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     favorite_time DATE not null,

```

```

10     constraint POST_FAVORITE_pk
11         primary key (post_id, user_id)
12 );
13
14 comment on table POST_FAVORITE is '帖子收藏';
15 comment on column POST_FAVORITE.post_id is '帖子 ID';
16 comment on column POST_FAVORITE.user_id is '用户 ID';
17 comment on column POST_FAVORITE.favorite_time is '收藏时间';

```

## 4.4.15 帖子举报表 POST\_REPORT

下面是帖子举报表 POST\_REPORT 的数据定义语言:

```

1  create table POST_REPORT
2  (
3      post_report_id    INT                GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4      ↪ not null
5      constraint POST_REPORT_pk
6          primary key,
7      reporter_id       INT                not null
8      constraint POST_REPORT_USER_fk_1
9          references "USER" ON DELETE CASCADE,
10     reported_user_id  INT                not null
11     constraint POST_REPORT_USER_fk_2
12         references "USER" ON DELETE CASCADE,
13     reported_post_id  INT                not null
14     constraint POST_REPORT_POST_fk
15         references POST ON DELETE CASCADE,
16     report_reason     VARCHAR(512) not null,
17     report_time       DATE                not null,
18     status            NUMBER(1)          not null
19 );
20
21 comment on table POST_REPORT is '帖子举报';
22 comment on column POST_REPORT.post_report_id is '帖子举报 ID';
23 comment on column POST_REPORT.reporter_id is '举报者 ID';
24 comment on column POST_REPORT.reported_user_id is '被举报者 ID';
25 comment on column POST_REPORT.reported_post_id is '被举报帖子 ID';
26 comment on column POST_REPORT.report_reason is '举报原因';
27 comment on column POST_REPORT.report_time is '举报时间';
28 comment on column POST_REPORT.status is '处理状态';

```

## 4.4.16 帖子评论举报表 POST\_COMMENT\_REPORT

下面是帖子评论举报表 POST\_COMMENT\_REPORT 的数据定义语言:

```

1  create table POST_COMMENT_REPORT
2  (
3      post_comment_report_id INT                GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT
4      ↪ BY 1) not null
5      constraint POST_COMMENT_REPORT_pk
6          primary key,

```

```

6      reporter_id          INT          not null
7          constraint POST_COMMENT_REPORT_USER_fk_1
8              references "USER" ON DELETE CASCADE,
9      reported_user_id     INT          not null
10         constraint POST_COMMENT_REPORT_USER_fk_2
11             references "USER" ON DELETE CASCADE,
12      reported_post_id     INT          not null
13         constraint POST_COMMENT_REPORT_POST_fk
14             references POST ON DELETE CASCADE,
15      reported_comment_id  INT          not null
16         constraint POST_COMMENT_REPORT_PC_fk
17             references POST_COMMENT ON DELETE CASCADE,
18      report_reason        VARCHAR(512) not null,
19      report_time          DATE          not null,
20      status               NUMBER(1)    not null
21 );
22
23 comment on table POST_COMMENT_REPORT is '帖子评论举报';
24 comment on column POST_COMMENT_REPORT.post_comment_report_id is '帖子评论举报 ID';
25 comment on column POST_COMMENT_REPORT.reporter_id is '举报者 ID';
26 comment on column POST_COMMENT_REPORT.reported_user_id is '被举报者 ID';
27 comment on column POST_COMMENT_REPORT.reported_post_id is '被举报评论所属帖子 ID';
28 comment on column POST_COMMENT_REPORT.reported_comment_id is '被举报评论 ID';
29 comment on column POST_COMMENT_REPORT.report_reason is '举报原因';
30 comment on column POST_COMMENT_REPORT.report_time is '举报时间';
31 comment on column POST_COMMENT_REPORT.status is '处理状态';

```

#### 4.4.17 新闻标签表 NEWS\_TAG

下面是新闻标签表 NEWS\_TAG 的数据定义语言：

```

1 create table NEWS_TAG
2 (
3     tag_id INT          not null
4         constraint NEWS_TAG_pk
5             primary key,
6     tag    VARCHAR(64) not null
7         constraint tag_uk
8             unique
9 );
10
11 comment on table NEWS_TAG is '新闻标签';
12 comment on column NEWS_TAG.tag_id is '新闻标签 ID';
13 comment on column NEWS_TAG.tag is '新闻标签';

```

#### 4.4.18 新闻表 NEWS

下面是新闻表 NEWS 的数据定义语言：

```

1 create table NEWS
2 (
3     news_id          INT          GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4         ↪ not null

```



```

4         constraint NEWS_pk
5             primary key,
6     user_id          INT                not null
7         constraint NEWS_USER_fk
8             references "USER" ON DELETE CASCADE,
9     tag_id           INT                not null
10        constraint NEWS_NEWS_TAG_fk
11            references NEWS_TAG (tag_id) ON DELETE CASCADE,
12    title             VARCHAR(256)      not null,
13    summary           VARCHAR(512)      not null,
14    content           CLOB               not null,
15    creation_date     DATE               not null,
16    update_date       DATE               not null,
17    cover_url         VARCHAR(2048)     not null,
18    is_sticky         NUMBER(1)         not null,
19    like_count        INT                not null,
20    dislike_count     INT                not null,
21    favorite_count    INT                not null,
22    comment_count     INT                not null
23 );
24
25 comment on table NEWS is '新闻';
26 comment on column NEWS.news_id is '新闻 ID';
27 comment on column NEWS.user_id is '管理员 ID';
28 comment on column NEWS.tag_id is '新闻标签 ID';
29 comment on column NEWS.title is '标题';
30 comment on column NEWS.summary is '新闻摘要';
31 comment on column NEWS.content is '内容';
32 comment on column NEWS.creation_date is '创建时间';
33 comment on column NEWS.update_date is '更新时间';
34 comment on column NEWS.cover_url is '封面图片链接';
35 comment on column NEWS.is_sticky is '是否置顶';
36 comment on column NEWS.like_count is '点赞数';
37 comment on column NEWS.dislike_count is '点踩数';
38 comment on column NEWS.favorite_count is '收藏数';
39 comment on column NEWS.comment_count is '评论数';

```

#### 4.4.19 新闻评论表 NEWS\_COMMENT

下面是新闻评论表 NEWS\_COMMENT 的数据定义语言：

```

1 create table NEWS_COMMENT
2 (
3     comment_id       INT                GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4     ↪ not null
5     constraint NEWS_COMMENT_pk
6         primary key,
7     news_id          INT                not null
8     constraint NEWS_COMMENT_NEWS_fk
9         references NEWS ON DELETE CASCADE,
10    user_id           INT                not null
11    constraint NEWS_COMMENT_USER_fk
12        references "USER" ON DELETE CASCADE,
13    parent_comment_id INT
14    constraint NEWS_COMMENT_NEWS_COMMENT_fk

```

```

14         references NEWS_COMMENT ON DELETE CASCADE,
15         content      VARCHAR(512) not null,
16         comment_time  DATE          not null,
17         like_count    INT           not null,
18         dislike_count INT           not null
19     );
20
21     comment on table NEWS_COMMENT is '新闻评论';
22     comment on column NEWS_COMMENT.comment_id is '新闻评论 ID';
23     comment on column NEWS_COMMENT.news_id is '新闻 ID';
24     comment on column NEWS_COMMENT.user_id is '用户 ID';
25     comment on column NEWS_COMMENT.parent_comment_id is '父评论 ID';
26     comment on column NEWS_COMMENT.content is '评论内容';
27     comment on column NEWS_COMMENT.comment_time is '评论时间';
28     comment on column NEWS_COMMENT.like_count is '点赞数';
29     comment on column NEWS_COMMENT.dislike_count is '点踩数';

```

#### 4.4.20 新闻点赞表 NEWS\_LIKE

下面是新闻点赞表 NEWS\_LIKE 的数据定义语言：

```

1  create table NEWS_LIKE
2  (
3      news_id    INT not null
4          constraint NEWS_LIKE_NEWS_fk
5              references NEWS ON DELETE CASCADE,
6      user_id    INT not null
7          constraint NEWS_LIKE_USER_fk
8              references "USER" ON DELETE CASCADE,
9      like_time  DATE not null,
10     constraint NEWS_LIKE_pk
11         primary key (news_id, user_id)
12 );
13
14     comment on table NEWS_LIKE is '新闻点赞';
15     comment on column NEWS_LIKE.news_id is '新闻 ID';
16     comment on column NEWS_LIKE.user_id is '用户 ID';
17     comment on column NEWS_LIKE.like_time is '点赞时间';

```

#### 4.4.21 新闻点踩表 NEWS\_DISLIKE

下面是新闻点踩表 NEWS\_DISLIKE 的数据定义语言：

```

1  create table NEWS_DISLIKE
2  (
3      news_id    INT not null
4          constraint NEWS_DISLIKE_NEWS_fk
5              references NEWS ON DELETE CASCADE,
6      user_id    INT not null
7          constraint NEWS_DISLIKE_USER_fk
8              references "USER" ON DELETE CASCADE,
9      dislike_time DATE not null,

```

```

10     constraint NEWS_DISLIKE_pk
11         primary key (news_id, user_id)
12 );
13
14 comment on table NEWS_DISLIKE is '新闻点踩';
15 comment on column NEWS_DISLIKE.news_id is '新闻 ID';
16 comment on column NEWS_DISLIKE.user_id is '用户 ID';
17 comment on column NEWS_DISLIKE.dislike_time is '点踩时间';

```

#### 4.4.22 新闻评论点赞表 NEWS\_COMMENT\_LIKE

下面是新闻评论点赞表 NEWS\_COMMENT\_LIKE 的数据定义语言:

```

1 create table NEWS_COMMENT_LIKE
2 (
3     comment_id INT not null
4         constraint NEWS_COMMENT_LIKE_NC_fk
5             references NEWS_COMMENT ON DELETE CASCADE,
6     user_id INT not null
7         constraint NEWS_COMMENT_LIKE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     like_time DATE not null,
10    constraint NEWS_COMMENT_LIKE_pk
11        primary key (comment_id, user_id)
12 );
13
14 comment on table NEWS_COMMENT_LIKE is '新闻评论点赞';
15 comment on column NEWS_COMMENT_LIKE.comment_id is '新闻评论 ID';
16 comment on column NEWS_COMMENT_LIKE.user_id is '用户 ID';
17 comment on column NEWS_COMMENT_LIKE.like_time is '点赞时间';

```

#### 4.4.23 新闻评论点踩表 NEWS\_COMMENT\_DISLIKE

下面是新闻评论点踩表 NEWS\_COMMENT\_DISLIKE 的数据定义语言:

```

1 create table NEWS_COMMENT_DISLIKE
2 (
3     comment_id INT not null
4         constraint NEWS_COMMENT_DISLIKE_NC_fk
5             references NEWS_COMMENT ON DELETE CASCADE,
6     user_id INT not null
7         constraint NEWS_COMMENT_DISLIKE_USER_fk
8             references "USER" ON DELETE CASCADE,
9     dislike_time DATE not null,
10    constraint NEWS_COMMENT_DISLIKE_pk
11        primary key (comment_id, user_id)
12 );
13
14 comment on table NEWS_COMMENT_DISLIKE is '新闻评论点踩';
15 comment on column NEWS_COMMENT_DISLIKE.comment_id is '新闻评论 ID';
16 comment on column NEWS_COMMENT_DISLIKE.user_id is '用户 ID';
17 comment on column NEWS_COMMENT_DISLIKE.dislike_time is '点踩时间';

```

## 4.4.24 新闻收藏表 NEWS\_FAVORITE

下面是新闻收藏表 NEWS\_FAVORITE 的数据定义语言：

```

1  create table NEWS_FAVORITE
2  (
3      news_id          INT      not null
4          constraint NEWS_FAVORITE_NEWS_fk
5              references NEWS ON DELETE CASCADE,
6      user_id          INT      not null
7          constraint NEWS_FAVORITE_USER_fk
8              references "USER" ON DELETE CASCADE,
9      favorite_time    DATE not null,
10     constraint NEWS_FAVORITE_pk
11         primary key (news_id, user_id)
12 );
13
14 comment on table NEWS_FAVORITE is '新闻收藏';
15 comment on column NEWS_FAVORITE.news_id is '新闻 ID';
16 comment on column NEWS_FAVORITE.user_id is '用户 ID';
17 comment on column NEWS_FAVORITE.favorite_time is '收藏时间';

```

## 4.4.25 新闻评论举报表 NEWS\_COMMENT\_REPORT

下面是新闻评论举报表 NEWS\_COMMENT\_REPORT 的数据定义语言：

```

1  create table NEWS_COMMENT_REPORT
2  (
3      news_comment_report_id INT          GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT
4      ↪ BY 1) not null
5          constraint NEWS_COMMENT_REPORT_pk
6              primary key,
7      reporter_id          INT      not null
8          constraint NEWS_COMMENT_REPORT_USER_fk_1
9              references "USER" ON DELETE CASCADE,
10     reported_user_id      INT      not null
11         constraint NEWS_COMMENT_REPORT_USER_fk_2
12             references "USER" ON DELETE CASCADE,
13     reported_news_id      INT      not null
14         constraint NEWS_COMMENT_REPORT_NEWS_fk
15             references NEWS ON DELETE CASCADE,
16     reported_comment_id   INT      not null
17         constraint NEWS_COMMENT_REPORT_NC_fk
18             references NEWS_COMMENT ON DELETE CASCADE,
19     report_reason          VARCHAR(512) not null,
20     report_time            DATE      not null,
21     status                 NUMBER(1)  not null
22 );
23
24 comment on table NEWS_COMMENT_REPORT is '新闻评论举报';
25 comment on column NEWS_COMMENT_REPORT.news_comment_report_id is '新闻评论举报 ID';
26 comment on column NEWS_COMMENT_REPORT.reporter_id is '举报者 ID';
27 comment on column NEWS_COMMENT_REPORT.reported_user_id is '被举报者 ID';
28 comment on column NEWS_COMMENT_REPORT.reported_news_id is '被举报评论所属新闻 ID';
29 comment on column NEWS_COMMENT_REPORT.reported_comment_id is '被举报评论 ID';

```

```

29 comment on column NEWS_COMMENT_REPORT.report_reason is ' 举报原因';
30 comment on column NEWS_COMMENT_REPORT.report_time is ' 举报时间';
31 comment on column NEWS_COMMENT_REPORT.status is ' 处理状态';

```

## 4.4.26 宠物分类表 PET\_CATEGORY

下面是宠物分类表 PET\_CATEGORY 的数据定义语言：

```

1 create table PET_CATEGORY
2 (
3     category_id      INT          not null
4         constraint PET_CATEGORY_pk
5             primary key,
6     category_name_zh VARCHAR(1024) not null
7         constraint category_name_zh_uk
8             unique,
9     category_name_de VARCHAR(1024) not null
10        constraint category_name_de_uk
11            unique,
12     category_name_en VARCHAR(1024) not null
13        constraint category_name_en_uk
14            unique,
15     category_name_es VARCHAR(1024) not null
16        constraint category_name_es_uk
17            unique,
18     category_name_fr VARCHAR(1024) not null
19        constraint category_name_fr_uk
20            unique,
21     category_name_it VARCHAR(1024) not null
22        constraint category_name_it_uk
23            unique,
24     category_name_ja VARCHAR(1024) not null
25        constraint category_name_ja_uk
26            unique,
27     category_name_ko VARCHAR(1024) not null
28        constraint category_name_ko_uk
29            unique,
30     category_name_pt VARCHAR(1024) not null
31        constraint category_name_pt_uk
32            unique,
33     category_name_ru VARCHAR(1024) not null
34        constraint category_name_ru_uk
35            unique,
36     description_zh   CLOB          not null,
37     description_de   CLOB          not null,
38     description_en   CLOB          not null,
39     description_es   CLOB          not null,
40     description_fr   CLOB          not null,
41     description_it   CLOB          not null,
42     description_ja   CLOB          not null,
43     description_ko   CLOB          not null,
44     description_pt   CLOB          not null,
45     description_ru   CLOB          not null,
46     image_url        VARCHAR(2048) not null
47 );
48

```

```

49  comment on table PET_CATEGORY is '宠物分类';
50  comment on column PET_CATEGORY.category_id is '宠物分类 ID';
51  comment on column PET_CATEGORY.category_name_zh is '宠物分类名称 (汉语)';
52  comment on column PET_CATEGORY.category_name_de is '宠物分类名称 (德语)';
53  comment on column PET_CATEGORY.category_name_en is '宠物分类名称 (英语)';
54  comment on column PET_CATEGORY.category_name_es is '宠物分类名称 (西班牙语)';
55  comment on column PET_CATEGORY.category_name_fr is '宠物分类名称 (法语)';
56  comment on column PET_CATEGORY.category_name_it is '宠物分类名称 (意大利语)';
57  comment on column PET_CATEGORY.category_name_ja is '宠物分类名称 (日语)';
58  comment on column PET_CATEGORY.category_name_ko is '宠物分类名称 (韩语)';
59  comment on column PET_CATEGORY.category_name_pt is '宠物分类名称 (葡萄牙语)';
60  comment on column PET_CATEGORY.category_name_ru is '宠物分类名称 (俄语)';
61  comment on column PET_CATEGORY.description_zh is '描述 (汉语)';
62  comment on column PET_CATEGORY.description_de is '描述 (德语)';
63  comment on column PET_CATEGORY.description_en is '描述 (英语)';
64  comment on column PET_CATEGORY.description_es is '描述 (西班牙语)';
65  comment on column PET_CATEGORY.description_fr is '描述 (法语)';
66  comment on column PET_CATEGORY.description_it is '描述 (意大利语)';
67  comment on column PET_CATEGORY.description_ja is '描述 (日语)';
68  comment on column PET_CATEGORY.description_ko is '描述 (韩语)';
69  comment on column PET_CATEGORY.description_pt is '描述 (葡萄牙语)';
70  comment on column PET_CATEGORY.description_ru is '描述 (俄语)';
71  comment on column PET_CATEGORY.image_url is '图片链接';

```

#### 4.4.27 宠物子类表 PET\_SUBCATEGORY

下面是宠物子类表 PET\_SUBCATEGORY 的数据定义语言：

```

1  create table PET_SUBCATEGORY
2  (
3      subcategory_id          INT          not null
4          constraint PET_SUBCATEGORY_pk
5              primary key,
6      category_id            INT          not null
7          constraint PET_SUBCATEGORY_PC_fk
8              references PET_CATEGORY ON DELETE CASCADE,
9      subcategory_name_zh     VARCHAR(1024) not null
10         constraint subcategory_name_zh_uk
11             unique,
12      subcategory_name_de     VARCHAR(1024) not null
13         constraint subcategory_name_de_uk
14             unique,
15      subcategory_name_en     VARCHAR(1024) not null
16         constraint subcategory_name_en_uk
17             unique,
18      subcategory_name_es     VARCHAR(1024) not null
19         constraint subcategory_name_es_uk
20             unique,
21      subcategory_name_fr     VARCHAR(1024) not null
22         constraint subcategory_name_fr_uk
23             unique,
24      subcategory_name_it     VARCHAR(1024) not null
25         constraint subcategory_name_it_uk
26             unique,
27      subcategory_name_ja     VARCHAR(1024) not null

```

```

28         constraint subcategory_name_ja_uk
29             unique,
30     subcategory_name_ko    VARCHAR(1024) not null
31         constraint subcategory_name_ko_uk
32             unique,
33     subcategory_name_pt    VARCHAR(1024) not null
34         constraint subcategory_name_pt_uk
35             unique,
36     subcategory_name_ru    VARCHAR(1024) not null
37         constraint subcategory_name_ru_uk
38             unique,
39     description_zh         CLOB          not null,
40     description_de         CLOB          not null,
41     description_en         CLOB          not null,
42     description_es         CLOB          not null,
43     description_fr         CLOB          not null,
44     description_it         CLOB          not null,
45     description_ja         CLOB          not null,
46     description_ko         CLOB          not null,
47     description_pt         CLOB          not null,
48     description_ru         CLOB          not null,
49     image_url              VARCHAR(2048) not null,
50     origin_zh              VARCHAR(512)  not null,
51     origin_de              VARCHAR(512)  not null,
52     origin_en              VARCHAR(512)  not null,
53     origin_es              VARCHAR(512)  not null,
54     origin_fr              VARCHAR(512)  not null,
55     origin_it              VARCHAR(512)  not null,
56     origin_ja              VARCHAR(512)  not null,
57     origin_ko              VARCHAR(512)  not null,
58     origin_pt              VARCHAR(512)  not null,
59     origin_ru              VARCHAR(512)  not null,
60     size_zh                VARCHAR(512)  not null,
61     size_de                VARCHAR(512)  not null,
62     size_en                VARCHAR(512)  not null,
63     size_es                VARCHAR(512)  not null,
64     size_fr                VARCHAR(512)  not null,
65     size_it                VARCHAR(512)  not null,
66     size_ja                VARCHAR(512)  not null,
67     size_ko                VARCHAR(512)  not null,
68     size_pt                VARCHAR(512)  not null,
69     size_ru                VARCHAR(512)  not null,
70     coat_zh                VARCHAR(512)  not null,
71     coat_de                VARCHAR(512)  not null,
72     coat_en                VARCHAR(512)  not null,
73     coat_es                VARCHAR(512)  not null,
74     coat_fr                VARCHAR(512)  not null,
75     coat_it                VARCHAR(512)  not null,
76     coat_ja                VARCHAR(512)  not null,
77     coat_ko                VARCHAR(512)  not null,
78     coat_pt                VARCHAR(512)  not null,
79     coat_ru                VARCHAR(512)  not null,
80     lifespan_zh            VARCHAR(512)  not null,
81     lifespan_de            VARCHAR(512)  not null,
82     lifespan_en            VARCHAR(512)  not null,
83     lifespan_es            VARCHAR(512)  not null,
84     lifespan_fr            VARCHAR(512)  not null,
85     lifespan_it            VARCHAR(512)  not null,
86     lifespan_ja            VARCHAR(512)  not null,

```



```

87     lifespan_ko          VARCHAR(512)  not null,
88     lifespan_pt          VARCHAR(512)  not null,
89     lifespan_ru          VARCHAR(512)  not null,
90     temperament_zh       VARCHAR(512)  not null,
91     temperament_de       VARCHAR(512)  not null,
92     temperament_en       VARCHAR(512)  not null,
93     temperament_es       VARCHAR(512)  not null,
94     temperament_fr       VARCHAR(512)  not null,
95     temperament_it       VARCHAR(512)  not null,
96     temperament_ja       VARCHAR(512)  not null,
97     temperament_ko       VARCHAR(512)  not null,
98     temperament_pt       VARCHAR(512)  not null,
99     temperament_ru       VARCHAR(512)  not null,
100    diet_zh              VARCHAR(512)  not null,
101    diet_de              VARCHAR(512)  not null,
102    diet_en              VARCHAR(512)  not null,
103    diet_es              VARCHAR(512)  not null,
104    diet_fr              VARCHAR(512)  not null,
105    diet_it              VARCHAR(512)  not null,
106    diet_ja              VARCHAR(512)  not null,
107    diet_ko              VARCHAR(512)  not null,
108    diet_pt              VARCHAR(512)  not null,
109    diet_ru              VARCHAR(512)  not null,
110    latitude_and_longitude VARCHAR(512)  not null
111 );
112
113 comment on table PET_SUBCATEGORY is '宠物子类';
114 comment on column PET_SUBCATEGORY.subcategory_id is '宠物子类 ID';
115 comment on column PET_SUBCATEGORY.category_id is '宠物分类 ID';
116 comment on column PET_SUBCATEGORY.subcategory_name_zh is '宠物子类名称 (汉语)';
117 comment on column PET_SUBCATEGORY.subcategory_name_de is '宠物子类名称 (德语)';
118 comment on column PET_SUBCATEGORY.subcategory_name_en is '宠物子类名称 (英语)';
119 comment on column PET_SUBCATEGORY.subcategory_name_es is '宠物子类名称 (西班牙语)';
120 comment on column PET_SUBCATEGORY.subcategory_name_fr is '宠物子类名称 (法语)';
121 comment on column PET_SUBCATEGORY.subcategory_name_it is '宠物子类名称 (意大利语)';
122 comment on column PET_SUBCATEGORY.subcategory_name_ja is '宠物子类名称 (日语)';
123 comment on column PET_SUBCATEGORY.subcategory_name_ko is '宠物子类名称 (韩语)';
124 comment on column PET_SUBCATEGORY.subcategory_name_pt is '宠物子类名称 (葡萄牙语)';
125 comment on column PET_SUBCATEGORY.subcategory_name_ru is '宠物子类名称 (俄语)';
126 comment on column PET_SUBCATEGORY.description_zh is '描述 (汉语)';
127 comment on column PET_SUBCATEGORY.description_de is '描述 (德语)';
128 comment on column PET_SUBCATEGORY.description_en is '描述 (英语)';
129 comment on column PET_SUBCATEGORY.description_es is '描述 (西班牙语)';
130 comment on column PET_SUBCATEGORY.description_fr is '描述 (法语)';
131 comment on column PET_SUBCATEGORY.description_it is '描述 (意大利语)';
132 comment on column PET_SUBCATEGORY.description_ja is '描述 (日语)';
133 comment on column PET_SUBCATEGORY.description_ko is '描述 (韩语)';
134 comment on column PET_SUBCATEGORY.description_pt is '描述 (葡萄牙语)';
135 comment on column PET_SUBCATEGORY.description_ru is '描述 (俄语)';
136 comment on column PET_SUBCATEGORY.image_url is '图片链接';
137 comment on column PET_SUBCATEGORY.origin_zh is '起源地 (汉语)';
138 comment on column PET_SUBCATEGORY.origin_de is '起源地 (德语)';
139 comment on column PET_SUBCATEGORY.origin_en is '起源地 (英语)';
140 comment on column PET_SUBCATEGORY.origin_es is '起源地 (西班牙语)';
141 comment on column PET_SUBCATEGORY.origin_fr is '起源地 (法语)';
142 comment on column PET_SUBCATEGORY.origin_it is '起源地 (意大利语)';
143 comment on column PET_SUBCATEGORY.origin_ja is '起源地 (日语)';
144 comment on column PET_SUBCATEGORY.origin_ko is '起源地 (韩语)';

```



装

订

线

```

145 comment on column PET_SUBCATEGORY.origin_pt is '起源地 (葡萄牙语)';
146 comment on column PET_SUBCATEGORY.origin_ru is '起源地 (俄语)';
147 comment on column PET_SUBCATEGORY.size_zh is '体型 (汉语)';
148 comment on column PET_SUBCATEGORY.size_de is '体型 (德语)';
149 comment on column PET_SUBCATEGORY.size_en is '体型 (英语)';
150 comment on column PET_SUBCATEGORY.size_es is '体型 (西班牙语)';
151 comment on column PET_SUBCATEGORY.size_fr is '体型 (法语)';
152 comment on column PET_SUBCATEGORY.size_it is '体型 (意大利语)';
153 comment on column PET_SUBCATEGORY.size_ja is '体型 (日语)';
154 comment on column PET_SUBCATEGORY.size_ko is '体型 (韩语)';
155 comment on column PET_SUBCATEGORY.size_pt is '体型 (葡萄牙语)';
156 comment on column PET_SUBCATEGORY.size_ru is '体型 (俄语)';
157 comment on column PET_SUBCATEGORY.coat_zh is '毛色 (汉语)';
158 comment on column PET_SUBCATEGORY.coat_de is '毛色 (德语)';
159 comment on column PET_SUBCATEGORY.coat_en is '毛色 (英语)';
160 comment on column PET_SUBCATEGORY.coat_es is '毛色 (西班牙语)';
161 comment on column PET_SUBCATEGORY.coat_fr is '毛色 (法语)';
162 comment on column PET_SUBCATEGORY.coat_it is '毛色 (意大利语)';
163 comment on column PET_SUBCATEGORY.coat_ja is '毛色 (日语)';
164 comment on column PET_SUBCATEGORY.coat_ko is '毛色 (韩语)';
165 comment on column PET_SUBCATEGORY.coat_pt is '毛色 (葡萄牙语)';
166 comment on column PET_SUBCATEGORY.coat_ru is '毛色 (俄语)';
167 comment on column PET_SUBCATEGORY.lifespan_zh is '寿命 (汉语)';
168 comment on column PET_SUBCATEGORY.lifespan_de is '寿命 (德语)';
169 comment on column PET_SUBCATEGORY.lifespan_en is '寿命 (英语)';
170 comment on column PET_SUBCATEGORY.lifespan_es is '寿命 (西班牙语)';
171 comment on column PET_SUBCATEGORY.lifespan_fr is '寿命 (法语)';
172 comment on column PET_SUBCATEGORY.lifespan_it is '寿命 (意大利语)';
173 comment on column PET_SUBCATEGORY.lifespan_ja is '寿命 (日语)';
174 comment on column PET_SUBCATEGORY.lifespan_ko is '寿命 (韩语)';
175 comment on column PET_SUBCATEGORY.lifespan_pt is '寿命 (葡萄牙语)';
176 comment on column PET_SUBCATEGORY.lifespan_ru is '寿命 (俄语)';
177 comment on column PET_SUBCATEGORY.temperament_zh is '性情 (汉语)';
178 comment on column PET_SUBCATEGORY.temperament_de is '性情 (德语)';
179 comment on column PET_SUBCATEGORY.temperament_en is '性情 (英语)';
180 comment on column PET_SUBCATEGORY.temperament_es is '性情 (西班牙语)';
181 comment on column PET_SUBCATEGORY.temperament_fr is '性情 (法语)';
182 comment on column PET_SUBCATEGORY.temperament_it is '性情 (意大利语)';
183 comment on column PET_SUBCATEGORY.temperament_ja is '性情 (日语)';
184 comment on column PET_SUBCATEGORY.temperament_ko is '性情 (韩语)';
185 comment on column PET_SUBCATEGORY.temperament_pt is '性情 (葡萄牙语)';
186 comment on column PET_SUBCATEGORY.temperament_ru is '性情 (俄语)';
187 comment on column PET_SUBCATEGORY.diet_zh is '饮食习惯 (汉语)';
188 comment on column PET_SUBCATEGORY.diet_de is '饮食习惯 (德语)';
189 comment on column PET_SUBCATEGORY.diet_en is '饮食习惯 (英语)';
190 comment on column PET_SUBCATEGORY.diet_es is '饮食习惯 (西班牙语)';
191 comment on column PET_SUBCATEGORY.diet_fr is '饮食习惯 (法语)';
192 comment on column PET_SUBCATEGORY.diet_it is '饮食习惯 (意大利语)';
193 comment on column PET_SUBCATEGORY.diet_ja is '饮食习惯 (日语)';
194 comment on column PET_SUBCATEGORY.diet_ko is '饮食习惯 (韩语)';
195 comment on column PET_SUBCATEGORY.diet_pt is '饮食习惯 (葡萄牙语)';
196 comment on column PET_SUBCATEGORY.diet_ru is '饮食习惯 (俄语)';
197 comment on column PET_SUBCATEGORY.latitude_and_longitude is '经纬度参数';

```

## 4.4.28 宠物护理指导表 PET\_CARE\_GUIDE

下面是宠物护理指导表 PET\_CARE\_GUIDE 的数据定义语言：

```

1  create table PET_CARE_GUIDE
2  (
3      guide_id          INT          not null
4          constraint PET_CARE_GUIDE_pk
5              primary key,
6      subcategory_id INT          not null
7          constraint PET_CARE_GUIDE_PSC_fk
8              references PET_SUBCATEGORY ON DELETE CASCADE,
9      title_zh          VARCHAR(2048) not null,
10     title_de          VARCHAR(2048) not null,
11     title_en          VARCHAR(2048) not null,
12     title_es          VARCHAR(2048) not null,
13     title_fr          VARCHAR(2048) not null,
14     title_it          VARCHAR(2048) not null,
15     title_ja          VARCHAR(2048) not null,
16     title_ko          VARCHAR(2048) not null,
17     title_pt          VARCHAR(2048) not null,
18     title_ru          VARCHAR(2048) not null,
19     content_zh         CLOB          not null,
20     content_de         CLOB          not null,
21     content_en         CLOB          not null,
22     content_es         CLOB          not null,
23     content_fr         CLOB          not null,
24     content_it         CLOB          not null,
25     content_ja         CLOB          not null,
26     content_ko         CLOB          not null,
27     content_pt         CLOB          not null,
28     content_ru         CLOB          not null
29 );
30
31 comment on table PET_CARE_GUIDE is '宠物护理指导';
32 comment on column PET_CARE_GUIDE.guide_id is '宠物护理指导 ID';
33 comment on column PET_CARE_GUIDE.subcategory_id is '宠物子类 ID';
34 comment on column PET_CARE_GUIDE.title_zh is '标题 (汉语)';
35 comment on column PET_CARE_GUIDE.title_de is '标题 (德语)';
36 comment on column PET_CARE_GUIDE.title_en is '标题 (英语)';
37 comment on column PET_CARE_GUIDE.title_es is '标题 (西班牙语)';
38 comment on column PET_CARE_GUIDE.title_fr is '标题 (法语)';
39 comment on column PET_CARE_GUIDE.title_it is '标题 (意大利语)';
40 comment on column PET_CARE_GUIDE.title_ja is '标题 (日语)';
41 comment on column PET_CARE_GUIDE.title_ko is '标题 (韩语)';
42 comment on column PET_CARE_GUIDE.title_pt is '标题 (葡萄牙语)';
43 comment on column PET_CARE_GUIDE.title_ru is '标题 (俄语)';
44 comment on column PET_CARE_GUIDE.content_zh is '内容 (汉语)';
45 comment on column PET_CARE_GUIDE.content_de is '内容 (德语)';
46 comment on column PET_CARE_GUIDE.content_en is '内容 (英语)';
47 comment on column PET_CARE_GUIDE.content_es is '内容 (西班牙语)';
48 comment on column PET_CARE_GUIDE.content_fr is '内容 (法语)';
49 comment on column PET_CARE_GUIDE.content_it is '内容 (意大利语)';
50 comment on column PET_CARE_GUIDE.content_ja is '内容 (日语)';
51 comment on column PET_CARE_GUIDE.content_ko is '内容 (韩语)';
52 comment on column PET_CARE_GUIDE.content_pt is '内容 (葡萄牙语)';
53 comment on column PET_CARE_GUIDE.content_ru is '内容 (俄语)';

```

## 4.4.29 宠物领养表 PET\_ADOPTION

下面是宠物领养表 PET\_ADOPTION 的数据定义语言：

```

1  create table PET_ADOPTION
2  (
3      adoption_id          INT          GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY
4      ↪ 1) not null
5          constraint PET_ADOPTION_pk
6              primary key,
7      user_id              INT          not null
8          constraint PET_ADOPTION_USER_fk
9              references "USER" ON DELETE CASCADE,
10     category_id          INT          not null
11         constraint PET_ADOPTION_PET_CATEGORY_fk
12             references PET_CATEGORY ON DELETE CASCADE,
13     subcategory_id       INT          not null
14         constraint PET_ADOPTION_PS_fk
15             references PET_SUBCATEGORY ON DELETE CASCADE,
16     name                  VARCHAR(64),
17     image_url             VARCHAR(2048) not null,
18     age                   INT          not null,
19     location              VARCHAR(1024) not null,
20     reason                VARCHAR(1024) not null,
21     health                VARCHAR(1024) not null,
22     latest_health_check   DATE         not null,
23     vaccination           VARCHAR(1024) not null,
24     care_needs            VARCHAR(1024),
25     dietary_needs         VARCHAR(1024),
26     behavior              VARCHAR(1024),
27     neutered_or_spayed    NUMBER(1)    not null,
28     notes                 VARCHAR(1024),
29     status                NUMBER(1)    not null,
30     gender                NUMBER(1)    not null,
31     appendix_url          VARCHAR(2048)
32 );
33
34 comment on table PET_ADOPTION is '宠物领养';
35 comment on column PET_ADOPTION.adoption_id is '宠物领养 ID';
36 comment on column PET_ADOPTION.user_id is '用户 ID';
37 comment on column PET_ADOPTION.category_id is '宠物分类 ID';
38 comment on column PET_ADOPTION.subcategory_id is '宠物子类 ID';
39 comment on column PET_ADOPTION.name is '宠物名称';
40 comment on column PET_ADOPTION.image_url is '图片链接';
41 comment on column PET_ADOPTION.age is '宠物年龄';
42 comment on column PET_ADOPTION.location is '地址';
43 comment on column PET_ADOPTION.reason is '转让原因';
44 comment on column PET_ADOPTION.health is '健康情况';
45 comment on column PET_ADOPTION.latest_health_check is '最近一次健康检查日期';
46 comment on column PET_ADOPTION.vaccination is '疫苗接种情况';
47 comment on column PET_ADOPTION.care_needs is '特殊护理需求';
48 comment on column PET_ADOPTION.dietary_needs is '特殊饮食需求';
49 comment on column PET_ADOPTION.behavior is '性格行为特征';
50 comment on column PET_ADOPTION.neutered_or_spayed is '是否已绝育';
51 comment on column PET_ADOPTION.notes is '备注';
52 comment on column PET_ADOPTION.status is '领养状态';
53 comment on column PET_ADOPTION.gender is '性别';

```

53 `comment on column PET_ADOPTION.appendix_url is ' 附件链接';`

## 4.4.30 开发团队表 DEVELOPMENT\_TEAM

下面是开发团队表 DEVELOPMENT\_TEAM 的数据定义语言：

```
1 create table DEVELOPMENT_TEAM
2 (
3     id          INT          not null
4     constraint DEVELOPMENT_TEAM_pk
5         primary key,
6     name        VARCHAR(64)  not null,
7     school      VARCHAR(64)  not null,
8     email       VARCHAR(2048) not null,
9     github_name VARCHAR(64)  not null,
10    github_profile VARCHAR(2048) not null
11 );
12
13 comment on table DEVELOPMENT_TEAM is ' 开发团队';
14 comment on column DEVELOPMENT_TEAM.id is ' 学号';
15 comment on column DEVELOPMENT_TEAM.name is ' 姓名';
16 comment on column DEVELOPMENT_TEAM.school is ' 学校';
17 comment on column DEVELOPMENT_TEAM.email is ' 电子邮箱';
18 comment on column DEVELOPMENT_TEAM.github_name is 'GitHub 名称';
19 comment on column DEVELOPMENT_TEAM.github_profile is 'GitHub 主页';
```

## 4.5 触发器设计

触发器 (Trigger) 是数据库管理系统中的一种特殊类型的存储过程，它可以自动执行（被触发）当对数据库进行指定的修改操作（如 INSERT、UPDATE、DELETE）时。触发器是对数据库事件的响应，它们在数据库表上定义，以保持数据的完整性和实现复杂的业务逻辑。

使用触发器的原因：

- **数据一致性和完整性：**触发器可以自动执行操作，确保数据遵循业务规则，从而保持数据的一致性和完整性。例如，在删除用户记录时，触发器可以自动删除该用户的所有相关数据，避免留下孤立记录。
- **自动化任务：**触发器可以用于自动更新数据库中的数据，如自动计算字段值或更新统计信息，减少应用层的复杂逻辑。
- **审计日志：**触发器可以用来自动记录数据更改历史，帮助构建审计日志，追踪数据的更改情况。
- **触发业务流程：**触发器可以启动其他业务流程，如在数据达到某个条件时发送通知或执行额外的业务规则。

在本项目中，我们设计了多个触发器来自动管理用户行为产生的数据变更，这包括对用户经验点、关注数、留言数、评论数、点赞数等的自动更新。设计触发器时，遵循的原则和步骤包括：

- **确定触发条件和时机**：每个触发器都关联一个明确的数据库事件（如 INSERT 或 DELETE），并确定是在事件之前还是之后触发（例如，AFTER 或 INSERT）。
- **编写业务逻辑**：触发器中包含必要的 SQL 语句，用于更新数据库中的相关表和字段。这些语句依据触发的数据操作来调整数值，如增加或减少点赞数。
- **错误处理**：设计时应考虑异常管理，确保触发器在处理失败时能够正确回滚或报告错误。
- **性能优化**：考虑触发器的执行效率，尽量减少对数据库性能的影响，特别是在高频更新的环境下。

使用触发器带来的主要好处包括：

- **维护数据的一致性和完整性**：自动处理相关数据的更新，确保数据库各处的数据同步。
- **减轻应用层负担**：将部分逻辑下放到数据库层，减轻应用层的计算压力。
- **自动化处理流程**：无需手动执行数据更新操作，触发器能够根据业务规则自动完成。
- **提高安全性**：通过数据库层的控制，可以更好地管理数据访问和修改权限。

通过上述设计和配置，触发器在保证数据操作效率的同时，也提高了系统的稳定性和可维护性。

#### 4.5.1 向用户打卡表插入记录：增加经验点数

下面是向用户打卡表插入记录时增加经验点数的触发器配置 SQL 代码：

```
1  -- 向用户打卡表插入记录：增加经验点数
2  CREATE OR REPLACE TRIGGER increment_experience_points
3  AFTER INSERT ON USER_CHECK_IN
4  FOR EACH ROW
5  BEGIN
6      -- 增加用户的经验点数
7      UPDATE "USER"
8      SET experience_points = experience_points + 100
9      WHERE user_id = :NEW.user_id;
10 END;
```

#### 4.5.2 从用户打卡表删除记录：减少经验点数

下面是从用户打卡表删除记录时减少经验点数的触发器配置 SQL 代码：

```
1  -- 从用户打卡表删除记录：减少经验点数
2  CREATE OR REPLACE TRIGGER decrement_experience_points
3  AFTER DELETE ON USER_CHECK_IN
4  FOR EACH ROW
5  BEGIN
6      -- 减少用户的经验点数
7      UPDATE "USER"
8      SET experience_points = experience_points - 100
9      WHERE user_id = :OLD.user_id;
10 END;
```

## 4.5.3 向用户关注表插入记录：增加关注数和被关注数

下面是向用户关注表插入记录时增加关注数和被关注数的触发器配置 SQL 代码：

```

1  -- 向用户关注表插入记录：增加关注数和被关注数
2  CREATE OR REPLACE TRIGGER increment_follow_counts
3  AFTER INSERT ON USER_FOLLOW
4  FOR EACH ROW
5  BEGIN
6      -- 增加关注者的关注数
7      UPDATE "USER"
8      SET follows_count = follows_count + 1
9      WHERE user_id = :NEW.follower_id;
10
11     -- 增加被关注者的被关注数
12     UPDATE "USER"
13     SET followed_count = followed_count + 1
14     WHERE user_id = :NEW.user_id;
15 END;
```

## 4.5.4 从用户关注表删除记录：减少关注数和被关注数

下面是从用户关注表删除记录时减少关注数和被关注数的触发器配置 SQL 代码：

```

1  -- 从用户关注表删除记录：减少关注数和被关注数
2  CREATE OR REPLACE TRIGGER decrement_follow_counts
3  AFTER DELETE ON USER_FOLLOW
4  FOR EACH ROW
5  BEGIN
6      -- 减少关注者的关注数
7      UPDATE "USER"
8      SET follows_count = follows_count - 1
9      WHERE user_id = :OLD.follower_id;
10
11     -- 减少被关注者的被关注数
12     UPDATE "USER"
13     SET followed_count = followed_count - 1
14     WHERE user_id = :OLD.user_id;
15 END;
```

## 4.5.5 向用户留言表插入记录：增加留言数

下面是向用户留言表插入记录时增加留言数的触发器配置 SQL 代码：

```

1  -- 向用户留言表插入记录：增加留言数
2  CREATE OR REPLACE TRIGGER increment_message_counts
3  AFTER INSERT ON USER_MESSAGE
4  FOR EACH ROW
5  BEGIN
6      -- 增加用户的留言数
7      UPDATE "USER"
8      SET message_count = message_count + 1
```



```

9      WHERE user_id = :NEW.user_id;
10 END;

```

## 4.5.6 从用户留言表删除记录：减少留言数

下面是从用户留言表删除记录时减少留言数的触发器配置 SQL 代码：

```

1  -- 从用户留言表删除记录：减少留言数
2  CREATE OR REPLACE TRIGGER decrement_message_counts
3  AFTER DELETE ON USER_MESSAGE
4  FOR EACH ROW
5  BEGIN
6      -- 减少用户的留言数
7      UPDATE "USER"
8      SET message_count = message_count - 1
9      WHERE user_id = :OLD.user_id;
10 END;

```

## 4.5.7 向帖子评论表插入记录：增加评论数

下面是向帖子评论表插入记录时增加评论数的触发器配置 SQL 代码：

```

1  -- 向帖子评论表插入记录：增加评论数
2  CREATE OR REPLACE TRIGGER increment_post_c_count
3  AFTER INSERT ON POST_COMMENT
4  FOR EACH ROW
5  BEGIN
6      -- 增加帖子的评论数
7      UPDATE POST
8      SET comment_count = comment_count + 1
9      WHERE post_id = :NEW.post_id;
10 END;

```

## 4.5.8 从帖子评论表删除记录：减少评论数

下面是从帖子评论表删除记录时减少评论数的触发器配置 SQL 代码：

```

1  -- 从帖子评论表删除记录：减少评论数
2  CREATE OR REPLACE TRIGGER decrement_post_c_count
3  AFTER DELETE ON POST_COMMENT
4  FOR EACH ROW
5  BEGIN
6      -- 减少帖子的评论数
7      UPDATE POST
8      SET comment_count = comment_count - 1
9      WHERE post_id = :OLD.post_id;
10 END;

```

## 4.5.9 向帖子点赞表插入记录：增加点赞数和被点赞数

下面是向帖子点赞表插入记录时增加点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 向帖子点赞表插入记录：增加点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER increment_post_like_counts
3  AFTER INSERT ON POST_LIKE
4  FOR EACH ROW
5  DECLARE
6      poster_user_id INT;
7  BEGIN
8      -- 增加帖子的点赞数
9      UPDATE POST
10     SET like_count = like_count + 1
11     WHERE post_id = :NEW.post_id;
12
13     -- 查找发帖用户 ID
14     SELECT user_id INTO poster_user_id
15     FROM POST
16     WHERE post_id = :NEW.post_id;
17
18     -- 增加发帖用户的被点赞数
19     UPDATE "USER"
20     SET liked_count = liked_count + 1
21     WHERE user_id = poster_user_id;
22
23     -- 增加用户的点赞数
24     UPDATE "USER"
25     SET likes_count = likes_count + 1
26     WHERE user_id = :NEW.user_id;
27 END;
```

## 4.5.10 从帖子点赞表删除记录：减少点赞数和被点赞数

下面是从帖子点赞表删除记录时减少点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 从帖子点赞表删除记录：减少点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER decrement_post_like_counts
3  AFTER DELETE ON POST_LIKE
4  FOR EACH ROW
5  DECLARE
6      poster_user_id INT;
7  BEGIN
8      -- 减少帖子的点赞数
9      UPDATE POST
10     SET like_count = like_count - 1
11     WHERE post_id = :OLD.post_id;
12
13     -- 查找发帖用户 ID
14     SELECT user_id INTO poster_user_id
15     FROM POST
16     WHERE post_id = :OLD.post_id;
17
18     -- 减少发帖用户的被点赞数
19     UPDATE "USER"
```



```

20     SET liked_count = liked_count - 1
21     WHERE user_id = poster_user_id;
22
23     -- 减少用户的点赞数
24     UPDATE "USER"
25     SET likes_count = likes_count - 1
26     WHERE user_id = :OLD.user_id;
27 END;

```

#### 4.5.11 向帖子点踩表插入记录：增加点踩数

下面是向帖子点踩表插入记录时增加点踩数的触发器配置 SQL 代码：

```

1  -- 向帖子点踩表插入记录：增加点踩数
2  CREATE OR REPLACE TRIGGER increment_post_dislike_count
3  AFTER INSERT ON POST_DISLIKE
4  FOR EACH ROW
5  BEGIN
6      -- 增加帖子的点踩数
7      UPDATE POST
8      SET dislike_count = dislike_count + 1
9      WHERE post_id = :NEW.post_id;
10 END;

```

#### 4.5.12 从帖子点踩表删除记录：减少点踩数

下面是从帖子点踩表删除记录时减少点踩数的触发器配置 SQL 代码：

```

1  -- 从帖子点踩表删除记录：减少点踩数
2  CREATE OR REPLACE TRIGGER decrement_post_dislike_count
3  AFTER DELETE ON POST_DISLIKE
4  FOR EACH ROW
5  BEGIN
6      -- 减少帖子的点踩数
7      UPDATE POST
8      SET dislike_count = dislike_count - 1
9      WHERE post_id = :OLD.post_id;
10 END;

```

#### 4.5.13 向帖子评论点赞表插入记录：增加点赞数和被点赞数

下面是向帖子评论点赞表插入记录时增加点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 向帖子评论点赞表插入记录：增加点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER increment_post_c_like_counts
3  AFTER INSERT ON POST_COMMENT_LIKE
4  FOR EACH ROW
5  DECLARE
6      commenter_user_id INT;
7  BEGIN

```

```

8      -- 增加帖子评论的点赞数
9      UPDATE POST_COMMENT
10     SET like_count = like_count + 1
11     WHERE comment_id = :NEW.comment_id;
12
13     -- 查找发布帖子评论用户 ID
14     SELECT user_id INTO commenter_user_id
15     FROM POST_COMMENT
16     WHERE comment_id = :NEW.comment_id;
17
18     -- 增加发布帖子评论用户的被点赞数
19     UPDATE "USER"
20     SET liked_count = liked_count + 1
21     WHERE user_id = commenter_user_id;
22
23     -- 增加用户的点赞数
24     UPDATE "USER"
25     SET likes_count = likes_count + 1
26     WHERE user_id = :NEW.user_id;
27 END;
```

#### 4.5.14 从帖子评论点赞表删除记录：减少点赞数和被点赞数

下面是从帖子评论点赞表删除记录时减少点赞数和被点赞数的触发器配置 SQL 代码：

```

1      -- 从帖子评论点赞表删除记录：减少点赞数和被点赞数
2      CREATE OR REPLACE TRIGGER decrement_post_c_like_counts
3      AFTER DELETE ON POST_COMMENT_LIKE
4      FOR EACH ROW
5      DECLARE
6          commenter_user_id INT;
7      BEGIN
8          -- 减少帖子评论的点赞数
9          UPDATE POST_COMMENT
10         SET like_count = like_count - 1
11         WHERE comment_id = :OLD.comment_id;
12
13         -- 查找发布帖子评论用户 ID
14         SELECT user_id INTO commenter_user_id
15         FROM POST_COMMENT
16         WHERE comment_id = :OLD.comment_id;
17
18         -- 减少发布帖子评论用户的被点赞数
19         UPDATE "USER"
20         SET liked_count = liked_count - 1
21         WHERE user_id = commenter_user_id;
22
23         -- 减少用户的点赞数
24         UPDATE "USER"
25         SET likes_count = likes_count - 1
26         WHERE user_id = :OLD.user_id;
27 END;
```

## 4.5.15 向帖子评论点踩表插入记录：增加点踩数

下面是向帖子评论点踩表插入记录时增加点踩数的触发器配置 SQL 代码：

```

1  -- 向帖子评论点踩表插入记录：增加点踩数
2  CREATE OR REPLACE TRIGGER increment_post_c_dislike_count
3  AFTER INSERT ON POST_COMMENT_DISLIKE
4  FOR EACH ROW
5  BEGIN
6      -- 增加帖子评论的点踩数
7      UPDATE POST_COMMENT
8      SET dislike_count = dislike_count + 1
9      WHERE comment_id = :NEW.comment_id;
10 END;
```

## 4.5.16 从帖子评论点踩表删除记录：减少点踩数

下面是从帖子评论点踩表删除记录时减少点踩数的触发器配置 SQL 代码：

```

1  -- 从帖子评论点踩表删除记录：减少点踩数
2  CREATE OR REPLACE TRIGGER decrement_post_c_dislike_count
3  AFTER DELETE ON POST_COMMENT_DISLIKE
4  FOR EACH ROW
5  BEGIN
6      -- 减少帖子评论的点踩数
7      UPDATE POST_COMMENT
8      SET dislike_count = dislike_count - 1
9      WHERE comment_id = :OLD.comment_id;
10 END;
```

## 4.5.17 向帖子收藏表插入记录：增加收藏数和被收藏数

下面是向帖子收藏表插入记录时增加收藏数和被收藏数的触发器配置 SQL 代码：

```

1  -- 向帖子收藏表插入记录：增加收藏数和被收藏数
2  CREATE OR REPLACE TRIGGER increment_post_favorite_counts
3  AFTER INSERT ON POST_FAVORITE
4  FOR EACH ROW
5  DECLARE
6      poster_user_id INT;
7  BEGIN
8      -- 增加帖子的收藏数
9      UPDATE POST
10     SET favorite_count = favorite_count + 1
11     WHERE post_id = :NEW.post_id;
12
13     -- 查找发帖用户 ID
14     SELECT user_id INTO poster_user_id
15     FROM POST
16     WHERE post_id = :NEW.post_id;
17
18     -- 增加发帖用户的被收藏数
```

```

19  UPDATE "USER"
20  SET favorited_count = favorited_count + 1
21  WHERE user_id = poster_user_id;
22
23  -- 增加用户的收藏数
24  UPDATE "USER"
25  SET favorites_count = favorites_count + 1
26  WHERE user_id = :NEW.user_id;
27  END;

```

#### 4.5.18 从帖子收藏表删除记录：减少收藏数和被收藏数

下面是从帖子收藏表删除记录时减少收藏数和被收藏数的触发器配置 SQL 代码：

```

1  -- 从帖子收藏表删除记录：减少收藏数和被收藏数
2  CREATE OR REPLACE TRIGGER decrement_post_favorite_counts
3  AFTER DELETE ON POST_FAVORITE
4  FOR EACH ROW
5  DECLARE
6      poster_user_id INT;
7  BEGIN
8      -- 减少帖子的收藏数
9      UPDATE POST
10     SET favorite_count = favorite_count - 1
11     WHERE post_id = :OLD.post_id;
12
13     -- 查找发帖用户 ID
14     SELECT user_id INTO poster_user_id
15     FROM POST
16     WHERE post_id = :OLD.post_id;
17
18     -- 减少发帖用户的被收藏数
19     UPDATE "USER"
20     SET favorited_count = favorited_count - 1
21     WHERE user_id = poster_user_id;
22
23     -- 减少用户的收藏数
24     UPDATE "USER"
25     SET favorites_count = favorites_count - 1
26     WHERE user_id = :OLD.user_id;
27  END;

```

#### 4.5.19 向新闻评论表插入记录：增加评论数

下面是向新闻评论表插入记录时增加评论数的触发器配置 SQL 代码：

```

1  -- 向新闻评论表插入记录：增加评论数
2  CREATE OR REPLACE TRIGGER increment_news_c_count
3  AFTER INSERT ON NEWS_COMMENT
4  FOR EACH ROW
5  BEGIN
6      -- 增加新闻的评论数
7      UPDATE NEWS

```

```

8      SET comment_count = comment_count + 1
9      WHERE news_id = :NEW.news_id;
10 END;

```

## 4.5.20 从新闻评论表删除记录：减少评论数

下面是从新闻评论表删除记录时减少评论数的触发器配置 SQL 代码：

```

1  -- 从新闻评论表删除记录：减少评论数
2  CREATE OR REPLACE TRIGGER decrement_news_c_count
3  AFTER DELETE ON NEWS_COMMENT
4  FOR EACH ROW
5  BEGIN
6      -- 减少新闻的评论数
7      UPDATE NEWS
8      SET comment_count = comment_count - 1
9      WHERE news_id = :OLD.news_id;
10 END;

```

## 4.5.21 向新闻点赞表插入记录：增加点赞数和被点赞数

下面是向新闻点赞表插入记录时增加点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 向新闻点赞表插入记录：增加点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER increment_news_like_counts
3  AFTER INSERT ON NEWS_LIKE
4  FOR EACH ROW
5  DECLARE
6      news_owner_id INT;
7  BEGIN
8      -- 增加新闻的点赞数
9      UPDATE NEWS
10     SET like_count = like_count + 1
11     WHERE news_id = :NEW.news_id;
12
13     -- 查找管理员 ID
14     SELECT user_id INTO news_owner_id
15     FROM NEWS
16     WHERE news_id = :NEW.news_id;
17
18     -- 增加管理员的被点赞数
19     UPDATE "USER"
20     SET liked_count = liked_count + 1
21     WHERE user_id = news_owner_id;
22
23     -- 增加用户的点赞数
24     UPDATE "USER"
25     SET likes_count = likes_count + 1
26     WHERE user_id = :NEW.user_id;
27 END;

```

## 4.5.22 从新闻点赞表删除记录：减少点赞数和被点赞数

下面是从新闻点赞表删除记录时减少点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 从新闻点赞表删除记录：减少点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER decrement_news_like_counts
3  AFTER DELETE ON NEWS_LIKE
4  FOR EACH ROW
5  DECLARE
6      news_owner_id INT;
7  BEGIN
8      -- 减少新闻的点赞数
9      UPDATE NEWS
10     SET like_count = like_count - 1
11     WHERE news_id = :OLD.news_id;
12
13     -- 查找管理员 ID
14     SELECT user_id INTO news_owner_id
15     FROM NEWS
16     WHERE news_id = :OLD.news_id;
17
18     -- 减少管理员的被点赞数
19     UPDATE "USER"
20     SET liked_count = liked_count - 1
21     WHERE user_id = news_owner_id;
22
23     -- 减少用户的点赞数
24     UPDATE "USER"
25     SET likes_count = likes_count - 1
26     WHERE user_id = :OLD.user_id;
27 END;
```

## 4.5.23 向新闻点踩表插入记录：增加点踩数

下面是向新闻点踩表插入记录时增加点踩数的触发器配置 SQL 代码：

```

1  -- 向新闻点踩表插入记录：增加点踩数
2  CREATE OR REPLACE TRIGGER increment_news_dislike_count
3  AFTER INSERT ON NEWS_DISLIKE
4  FOR EACH ROW
5  BEGIN
6      -- 增加新闻的点踩数
7      UPDATE NEWS
8      SET dislike_count = dislike_count + 1
9      WHERE news_id = :NEW.news_id;
10 END;
```

## 4.5.24 从新闻点踩表删除记录：减少点踩数

下面是从新闻点踩表删除记录时减少点踩数的触发器配置 SQL 代码：

```

1  -- 从新闻点踩表删除记录：减少点踩数
2  CREATE OR REPLACE TRIGGER decrement_news_dislike_count
3  AFTER DELETE ON NEWS_DISLIKE
4  FOR EACH ROW
5  BEGIN
6      -- 减少新闻的点踩数
7      UPDATE NEWS
8      SET dislike_count = dislike_count - 1
9      WHERE news_id = :OLD.news_id;
10 END;

```

#### 4.5.25 向新闻评论点赞表插入记录：增加点赞数和被点赞数

下面是向新闻评论点赞表插入记录时加点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 向新闻评论点赞表插入记录：增加点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER increment_news_c_like_counts
3  AFTER INSERT ON NEWS_COMMENT_LIKE
4  FOR EACH ROW
5  DECLARE
6      comment_user_id INT;
7  BEGIN
8      -- 增加新闻评论的点赞数
9      UPDATE NEWS_COMMENT
10     SET like_count = like_count + 1
11     WHERE comment_id = :NEW.comment_id;
12
13     -- 查找发布新闻评论用户 ID
14     SELECT user_id INTO comment_user_id
15     FROM NEWS_COMMENT
16     WHERE comment_id = :NEW.comment_id;
17
18     -- 增加发布新闻评论用户的被点赞数
19     UPDATE "USER"
20     SET liked_count = liked_count + 1
21     WHERE user_id = comment_user_id;
22
23     -- 增加用户的点赞数
24     UPDATE "USER"
25     SET likes_count = likes_count + 1
26     WHERE user_id = :NEW.user_id;
27 END;

```

#### 4.5.26 从新闻评论点赞表删除记录：减少点赞数和被点赞数

下面是从新闻评论点赞表删除记录时减少点赞数和被点赞数的触发器配置 SQL 代码：

```

1  -- 从新闻评论点赞表删除记录：减少点赞数和被点赞数
2  CREATE OR REPLACE TRIGGER decrement_news_c_like_counts
3  AFTER DELETE ON NEWS_COMMENT_LIKE
4  FOR EACH ROW
5  DECLARE
6      comment_user_id INT;

```

```

7 BEGIN
8     -- 减少新闻评论的点赞数
9     UPDATE NEWS_COMMENT
10    SET like_count = like_count - 1
11    WHERE comment_id = :OLD.comment_id;
12
13    -- 查找发布新闻评论用户 ID
14    SELECT user_id INTO comment_user_id
15    FROM NEWS_COMMENT
16    WHERE comment_id = :OLD.comment_id;
17
18    -- 减少发布新闻评论用户的被点赞数
19    UPDATE "USER"
20    SET liked_count = liked_count - 1
21    WHERE user_id = comment_user_id;
22
23    -- 减少用户的点赞数
24    UPDATE "USER"
25    SET likes_count = likes_count - 1
26    WHERE user_id = :OLD.user_id;
27 END;

```

#### 4.5.27 向新闻评论点踩表插入记录：增加点踩数

下面是向新闻评论点踩表插入记录时增加点踩数的触发器配置 SQL 代码：

```

1 -- 向新闻评论点踩表插入记录：增加点踩数
2 CREATE OR REPLACE TRIGGER increment_news_c_dislike_count
3 AFTER INSERT ON NEWS_COMMENT_DISLIKE
4 FOR EACH ROW
5 BEGIN
6     -- 增加新闻评论的点踩数
7     UPDATE NEWS_COMMENT
8     SET dislike_count = dislike_count + 1
9     WHERE comment_id = :NEW.comment_id;
10 END;

```

#### 4.5.28 从新闻评论点踩表删除记录：减少点踩数

下面是从新闻评论点踩表删除记录时减少点踩数的触发器配置 SQL 代码：

```

1 -- 从新闻评论点踩表删除记录：减少点踩数
2 CREATE OR REPLACE TRIGGER decrement_news_c_dislike_count
3 AFTER DELETE ON NEWS_COMMENT_DISLIKE
4 FOR EACH ROW
5 BEGIN
6     -- 减少新闻评论点踩数
7     UPDATE NEWS_COMMENT
8     SET dislike_count = dislike_count - 1
9     WHERE comment_id = :OLD.comment_id;
10 END;

```



## 4.5.29 向新闻收藏表插入记录：增加收藏数和被收藏数

下面是向新闻收藏表插入记录时增加收藏数和被收藏数的触发器配置 SQL 代码：

```

1  -- 向新闻收藏表插入记录：增加收藏数和被收藏数
2  CREATE OR REPLACE TRIGGER increment_news_favorite_counts
3  AFTER INSERT ON NEWS_FAVORITE
4  FOR EACH ROW
5  DECLARE
6      news_owner_id INT;
7  BEGIN
8      -- 增加新闻的收藏数
9      UPDATE NEWS
10     SET favorite_count = favorite_count + 1
11     WHERE news_id = :NEW.news_id;
12
13     -- 查找管理员 ID
14     SELECT user_id INTO news_owner_id
15     FROM NEWS
16     WHERE news_id = :NEW.news_id;
17
18     -- 增加管理员的被收藏数
19     UPDATE "USER"
20     SET favorited_count = favorited_count + 1
21     WHERE user_id = news_owner_id;
22
23     -- 增加用户的收藏数
24     UPDATE "USER"
25     SET favorites_count = favorites_count + 1
26     WHERE user_id = :NEW.user_id;
27 END;
```

## 4.5.30 从新闻收藏表删除记录：减少收藏数和被收藏数

下面是从新闻收藏表删除记录时减少收藏数和被收藏数的触发器配置 SQL 代码：

```

1  -- 从新闻收藏表删除记录：减少收藏数和被收藏数
2  CREATE OR REPLACE TRIGGER decrement_news_favorite_counts
3  AFTER DELETE ON NEWS_FAVORITE
4  FOR EACH ROW
5  DECLARE
6      news_owner_id INT;
7  BEGIN
8      -- 减少新闻的收藏数
9      UPDATE NEWS
10     SET favorite_count = favorite_count - 1
11     WHERE news_id = :OLD.news_id;
12
13     -- 查找管理员 ID
14     SELECT user_id INTO news_owner_id
15     FROM NEWS
16     WHERE news_id = :OLD.news_id;
17
18     -- 减少管理员的被收藏数
19     UPDATE "USER"
```

```

20 SET favorited_count = favorited_count - 1
21 WHERE user_id = news_owner_id;
22
23 -- 减少用户的收藏数
24 UPDATE "USER"
25 SET favorites_count = favorites_count - 1
26 WHERE user_id = :OLD.user_id;
27 END;

```

## 4.5.31 用户注册时向用户设置表插入记录

下面是用户注册时向用户设置表插入记录的触发器配置 SQL 代码:

```

1 -- 用户注册时向用户设置表插入记录
2 CREATE OR REPLACE TRIGGER add_user_setting_after_insert
3 AFTER INSERT ON "USER"
4 FOR EACH ROW
5 BEGIN
6     INSERT INTO USER_SETTING (
7         user_id,
8         is_telephone_public,
9         is_registration_date_public,
10        is_profile_public,
11        is_gender_public,
12        is_birthdate_public,
13        is_level_public,
14        is_following_count_public,
15        is_followers_count_public,
16        is_favorites_count_public,
17        is_favored_count_public,
18        is_likes_count_public,
19        is_liked_count_public,
20        is_message_count_public,
21        allow_following,
22        receive_admin_notifications,
23        receive_user_notifications
24    ) VALUES (
25        :NEW.user_id,
26        0, -- is_telephone_public
27        1, -- is_registration_date_public
28        1, -- is_profile_public
29        1, -- is_gender_public
30        0, -- is_birthdate_public
31        1, -- is_level_public
32        1, -- is_following_count_public
33        1, -- is_followers_count_public
34        1, -- is_favorites_count_public
35        1, -- is_favored_count_public
36        1, -- is_likes_count_public
37        1, -- is_liked_count_public
38        1, -- is_message_count_public
39        1, -- allow_following
40        1, -- receive_admin_notifications
41        1 -- receive_user_notifications
42    );
43 END;

```