《数据库系统原理》系统设计报告

需求分析

1. 现状与思考

走在校园里，我们时常会看到形形色色的猫猫出现在各种角落中，这时难免会萌生这样的想法：这些猫猫在冬天，**是否有足够的食物和取暖的空间**？在生病时，是否有得到**及时的救助**？它们的日常生活**是否有人关心，有人呵护**？

正是带着这些疑问，我们意识到，这些看似自由自在的猫猫，其实也需要我们的关怀和帮助。于是，我们希望通过一个校园猫猫管理平台，连接起所有爱心，为这些猫猫提供一个更有保障的生活环境。

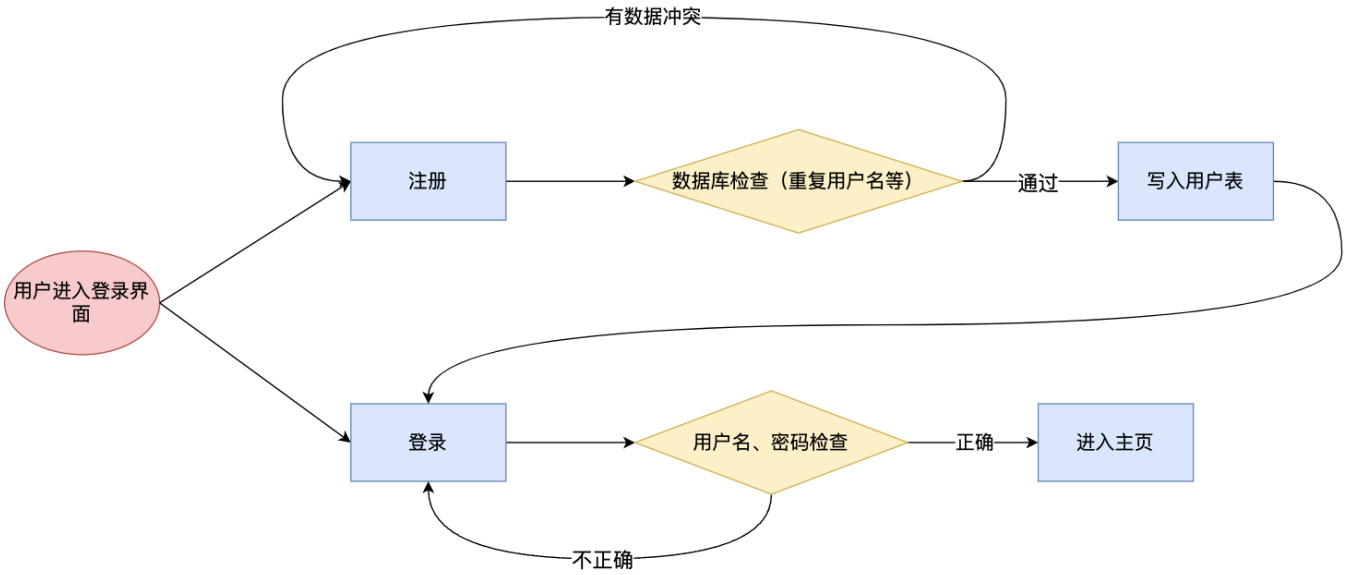
2. 设计思路

综合考虑用户群体的身份与时间分配，我们决定以志愿者与志愿活动为切入点展开对于校园内猫猫的活动。通过设置三种类型的用户层次：**普通用户**、**志愿者**、**管理员**来更好的实现用户管理与授权问题。我们综合考虑了以下的用户与活动需求：

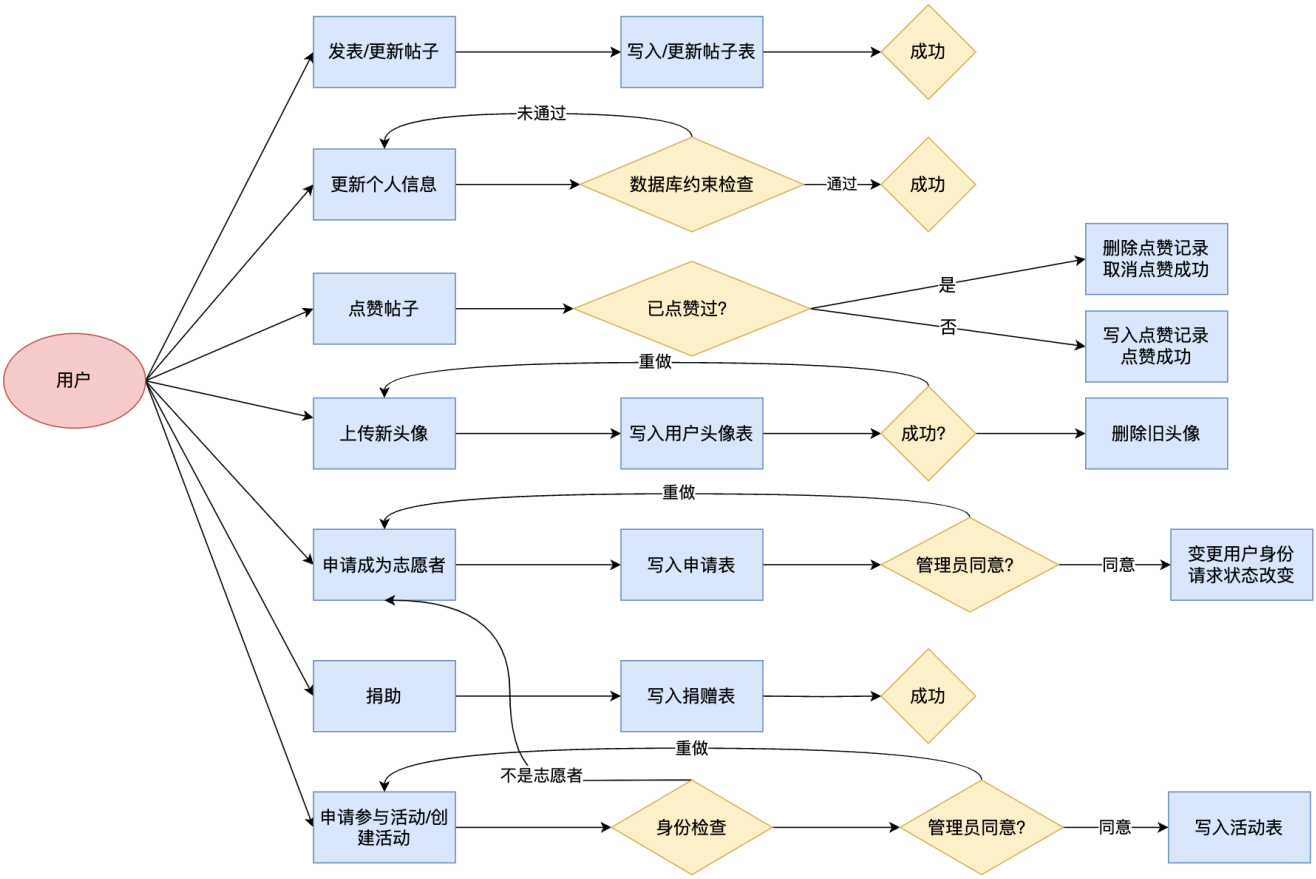
- 用户需要注册、登录、身份信息、鉴权
- 猫猫需要展示相关信息与图片、创建/更新状态、位置
- 需要有共同的论坛交流关于猫猫的信息，分享自己的见闻
- 需要有志愿者开展活动与志愿者申请
- 需要支持捐助
- 管理员需要对用户数据与站点信息进行审核与统计
- 管理员需要审核用户的申请、对敏感数据的修改

3. 数据流分析与数据流图

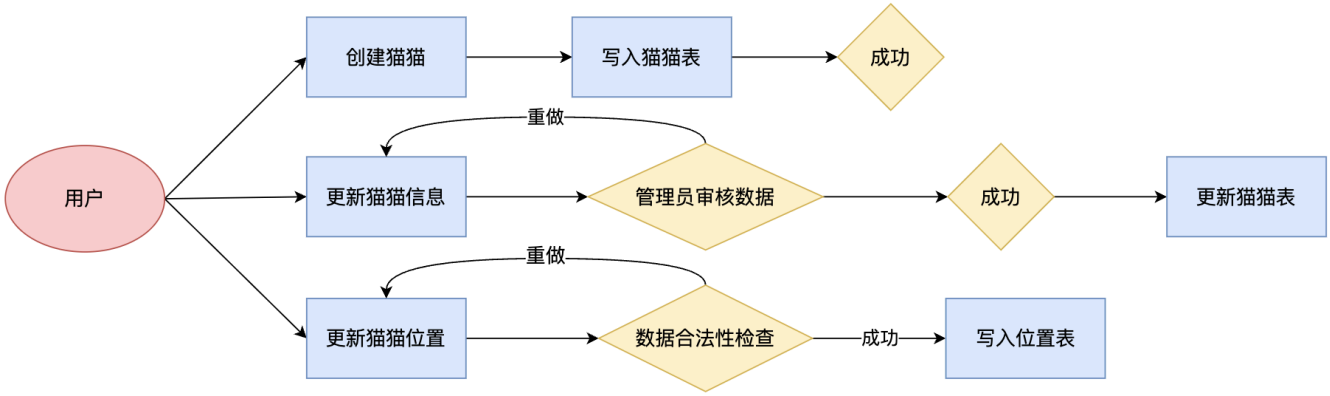
3.1 用户登录、注册数据流



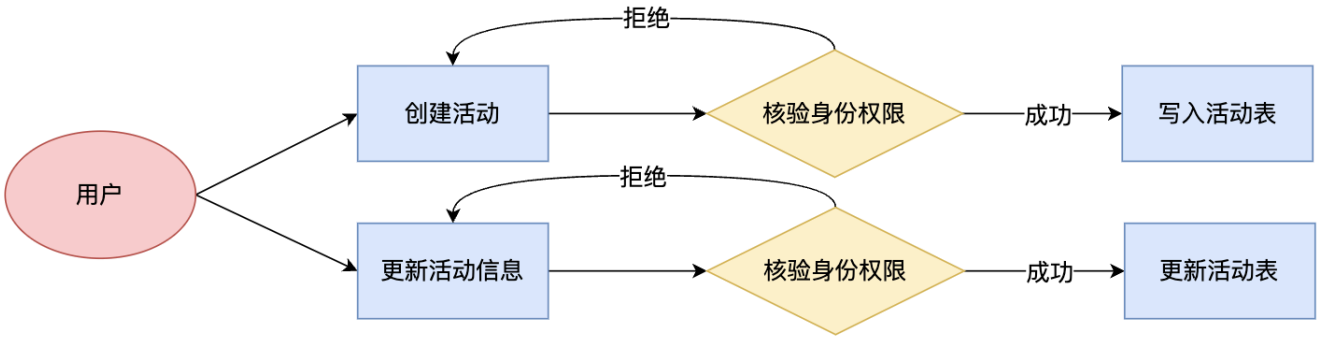
3.2 用户行为数据流



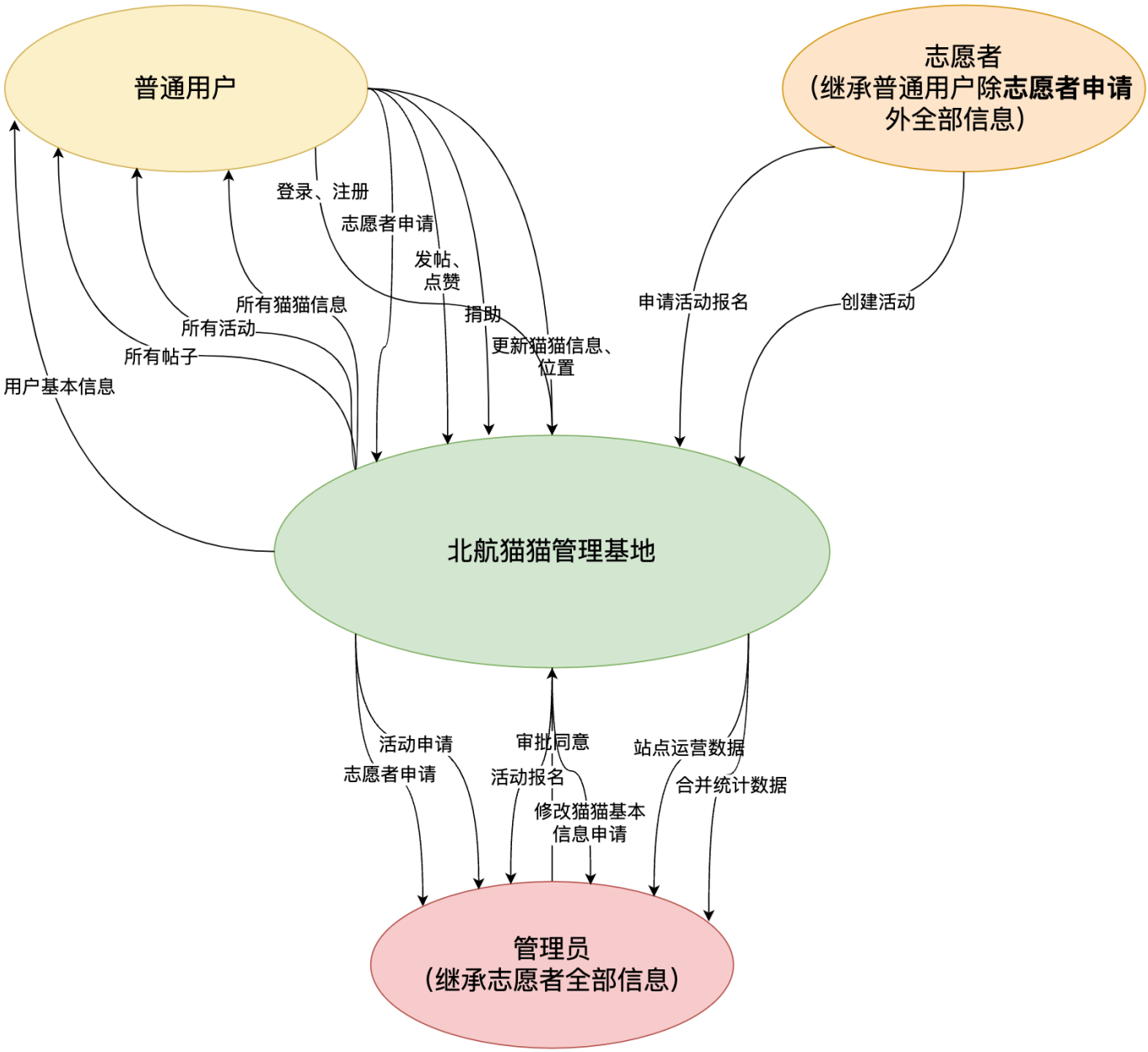
3.3 猫猫数据流



3.4 志愿活动数据流



3.5 项目主要数据流一览



4. 数据元素表

我们总共建立了13张数据表，现在请让我们逐张的描述这些表的结构。为了方便快速的查询与维护表之间的依赖关系，我们选择结合使用**建立索引**与**级联删除**的机制来建立数据表。

4.1 用户数据表 `User`

字段名称	数据类型	是否为空	默认值	说明
email	varchar(100)	否	无	用户的邮箱地址，唯一且**有索引**
nickname	varchar(50)	是	"User_" + 随机12字符	用户昵称，可为空
is_superuser	tinyint(1)	否	0 (False)	是否为超级用户
is_volunteer	tinyint(1)	否	0 (False)	是否为志愿者
avatar_url	varchar(255)	否	系统默认头像 URL	用户头像链接地址
id	**char(32)**	**否**	**UUID 自动生成**	**主键，唯一标识用户**
hashed_password	varchar(255)	否	无	用户密码的哈希值

4.2 帖子数据表`Post`

字段名称	数据类型	是否为空	默认值	说明
id	**char(32)**	**否**	**UUID 自动生成**	**主键，唯一标识每个帖子**
user_id	char(32)	否	无	**外键**，关联用户表 `user.id`，有索引
cat_id	char(32)	是	无	**外键**，可选，关联猫猫表 `cat.id`，有索引
title	varchar(255)	否	无	帖子的标题
content	varchar(4096)	是	无	帖子的内容
created_at	datetime	否	当前时间 (UTC+8)	帖子创建时间戳

4.3 帖子图片表`PostMedia`

字段名称	数据类型	是否为空	默认值	说明
id	**char(32)**	**否**	**UUID 自动生成**	**主键**，唯一标识每个媒体文件
post_id	char(32)	否	无	**外键**，关联帖子表 `post.id`，**有索引，级联删除**
image_url	varchar(512)	否	无	媒体文件的 URL，最长 512 个字符

4.4 帖子标签表`PostTag`

字段名称	数据类型	是否为空	默认值	说明
id	**char(32)**	**否**	**UUID 自动生成**	**主键**，唯一标识每个标签
user_id	char(32)	否	无	**外键**，关联用户表 `user.id`，**有索引**
name	varchar(255)	否	无	标签的名称，**有索引**

4.5 帖子点赞表`Like`

字段名称	数据类型	是否为空	默认值	说明
user_id	char(32)	否	无	**主键**，**外键**，引用`user.id`，**级联删除**
post_id	char(32)	否	无	**主键**，**外键**，引用`post.id`，**级联删除**，**有索引**

4.6 猫猫数据表`Cat`

字段名称	数据类型	是否为空	默认值	说明
id	**char(32)**	**否**	**无**	**主键**，自动生成UUID
name	varchar(256)	否	无	索引：`ix_cat_name`，猫的名字
is_male	tinyint(1)	否	true	猫的性别（默认为雄性）
age	int	是	无	猫的年龄（0-30岁）
health_condition	int	否	1	猫的健康状况（1-4，默认为1）
description	varchar(1024)	是	无	猫的描述（可选）
created_at	datetime	否	当前时间	猫记录创建时间（UTC-8）

4.7 猫猫图片表`CatMedia`

字段名称	数据类型	是否为空	默认值	说明
id	char(32)	否	无	**主键**，自动生成UUID
cat_id	char(32)	否	无	**外键**，引用`cat.id`，**级联删除**，**索引**：`ix_catmedia_cat_id`
image_url	varchar(255)	否	无	图片URL

4.8 猫猫位置表`CatLocation`

字段名称	数据类型	是否为空	默认值	说明
id	char(32)	否	无	**主键**，自动生成UUID
cat_id	char(32)	否	无	**外键**，引用`cat.id`，**级联删除**，索引：`ix_catlocation_cat_id`
user_id	char(32)	否	无	**外键**，引用`user.id`，索引：`ix_catlocation_user_id`
longitude	decimal(9, 6)	是	无	经度，范围为-180到180
latitude	decimal(9, 6)	是	无	纬度，范围为-90到90
created_at	datetime	否	当前时间	记录位置时间

4.9 志愿者申请表`VolunteerApplication`

字段名称	数据类型	是否为空	默认值	说明
id	char(32)	否	无	**主键**，自动生成UUID
user_id	char(32)	否	无	**外键**，引用`user.id`，**级联删除**，**索引**：`ix_volunteerapplication_user_id`
reason	varchar(1024)	是	无	申请理由
status	enum(`'PENDING'`, `'APPROVED'`, `'REJECTED'`)	否	PENDING	申请状态（默认PENDING）
created_at	datetime	否	当前时间	申请创建时间
updated_at	datetime	否	当前时间	申请最后更新时间

4.10 志愿活动数据表`Activity`

字段名称	数据类型	是否为空	默认值	说明
id	char(32)	否	无	**主键**，自动生成UUID
title	varchar(100)	否	无	活动标题
description	varchar(1024)	否	无	活动描述
location	varchar(100)	否	无	活动地点
starts_at	datetime	否	无	活动开始时间
ends_at	datetime	否	无	活动结束时间
signup_starts_at	datetime	是	无	报名开始时间
signup_ends_at	datetime	是	无	报名结束时间
created_at	datetime	否	当前时间	活动创建时间
creator_id	char(32)	否	无	**外键**，引用`user.id`，**索引**：`creator_id`
max_participants	int	否	无	最大参与人数

4.11 志愿活动报名表`ActivityRegistration`

字段名称	数据类型	是否为空	默认值	说明
user_id	char(32)	否	无	**主键**，**外键**，引用`user.id`，**级联删除**，**索引**：`ix_volunteerapplication_user_id`
activity_id	char(32)	否	无	**主键**，**外键**，引用`activity.id`，**级联删除**，**索引**：`ix_volunteerapplication_activity_id`
status	enum(`'PENDING'`, `'APPROVED'`, `'REJECTED'`)	否	PENDING	申请状态（默认PENDING）
created_at	datetime	否	当前时间	申请创建时间
updated_at	datetime	否	当前时间	申请最后更新时间

4.12 捐助记录表`Donation`

字段名称	数据类型	是否为空	默认值	说明
id	char(32)	否	无	**主键**，自动生成UUID
user_id	char(32)	否	无	**外键**，引用`user.id`，**级联删除**，**索引**：`ix_donation_user_id`
amount	float	否	无	捐赠金额，**索引**：`ix_donation_amount`
message	varchar(255)	是	无	捐赠留言
donated_at	datetime	否	当前时间	捐赠时间，**索引**：`ix_donation_donated_at`
is_anonymous	tinyint(1)	否	false	是否匿名捐赠

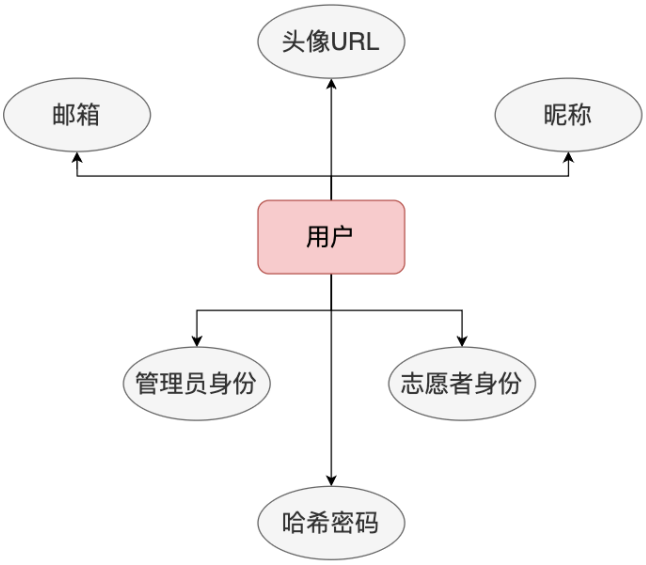
4.13 帖子、标签关系表`PostTagRelation`

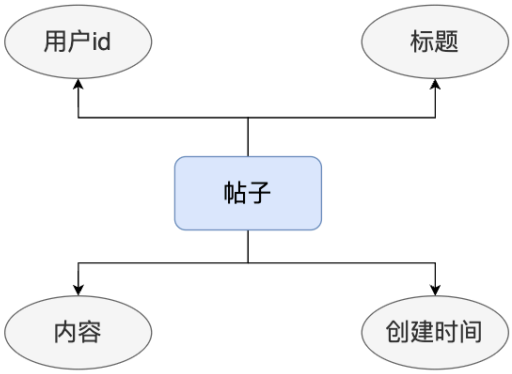
字段名称	数据类型	是否为空	默认值	说明
post_id	char(32)	否	无	**外键**，引用`post.id`，**主键**，**索引**：`ix_posttagrelation_post_id`
tag_id	char(32)	否	无	**外键**，引用`posttag.id`，**主键**，**索引**：`ix_posttagrelation_tag_id`

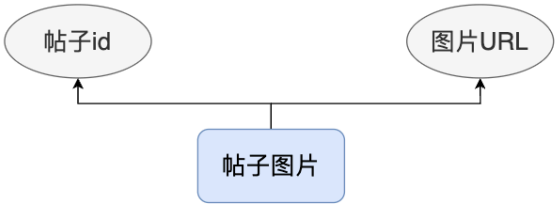
E-R图

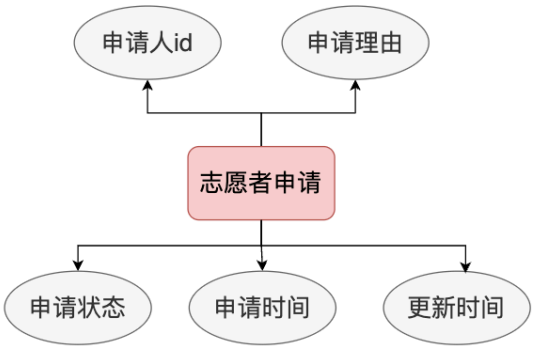
1. 各实体部分E-R图

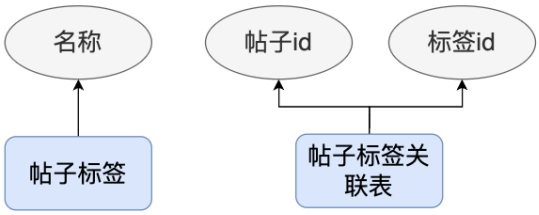
我们总共规划了如下的实体。为了美观起见，我们省略了每个实体的ID（在我们的项目中，所有数据库表都使用`UUID`作为主键来标识记录）。

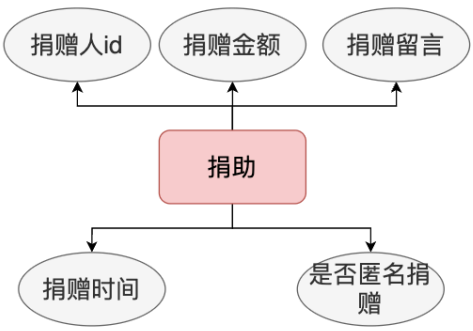

头像URL
邮箱
昵称
用户
管理员身份
志愿者身份
哈希密码

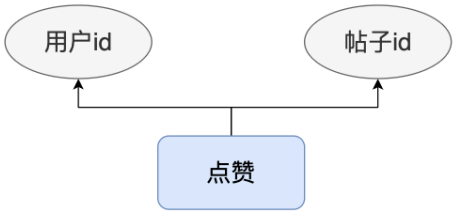

用户id
标题
帖子
内容
创建时间

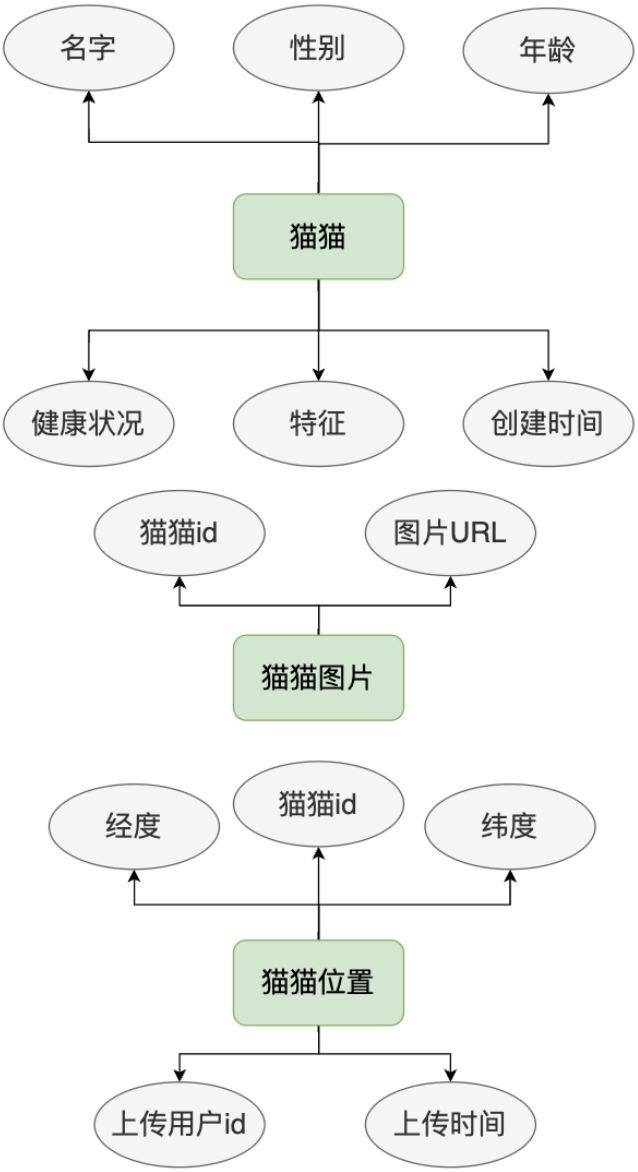

帖子id
图片URL
帖子图片

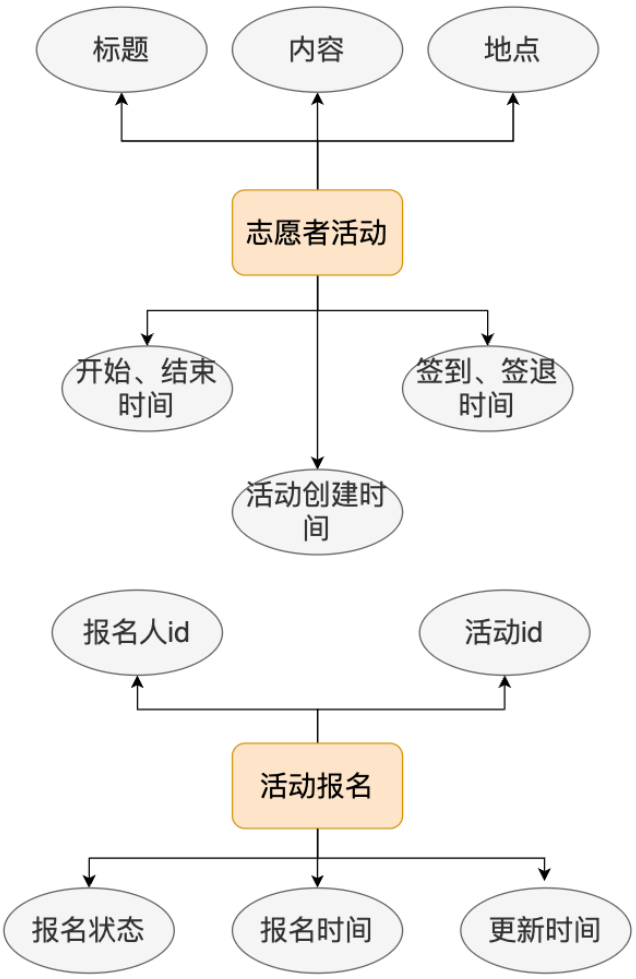

申请人id
申请理由
志愿者申请
申请状态
申请时间
更新时间


名称
帖子id
标签id
帖子标签
帖子标签关联表


捐赠人id
捐赠金额
捐赠留言
捐助
捐赠时间
是否匿名捐赠

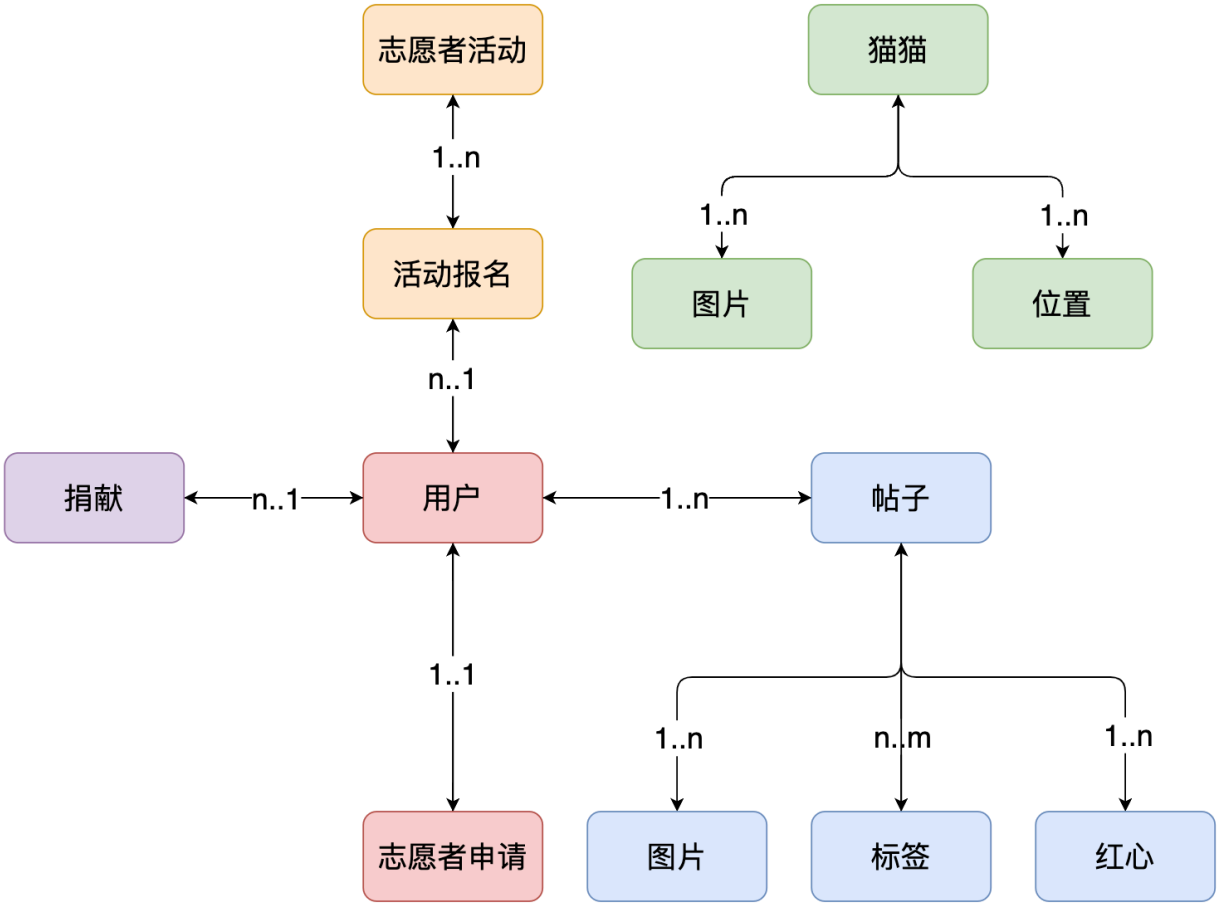

用户id
帖子id
点赞



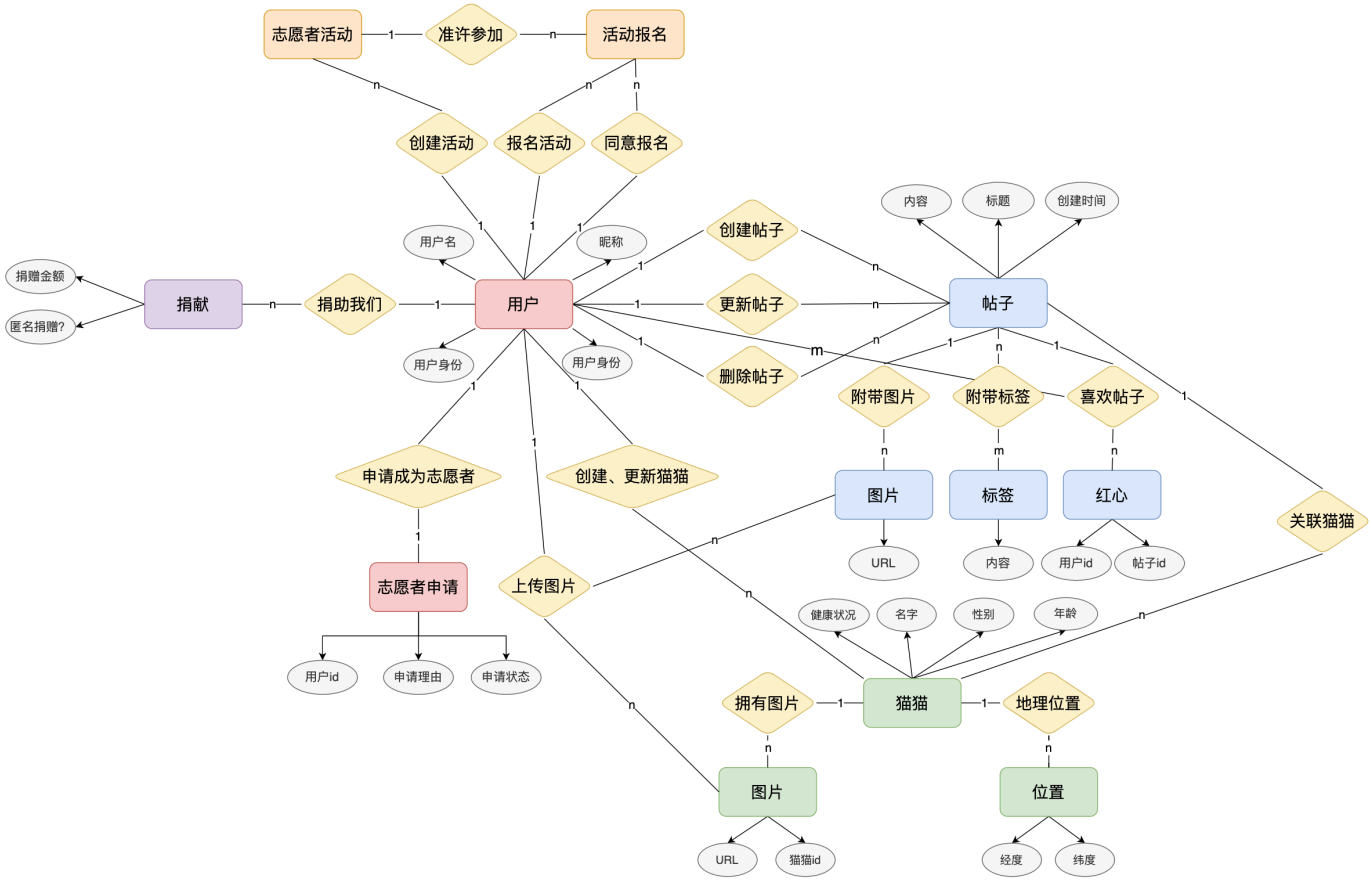


2. 系统整体E-R图

由于整个系统过于庞大，我们绘制了一个*简化的*E-R图来快速的表征系统各实体之间的关系：



完整的E-R图展示如下（为了方便展示，省略了部分属性）：



逻辑模式

1. 实体关系模式及一对多、一对一关系模式

1.1 `User` 实体

$F = \{id \to \text{其他属性}，email \to \text{其他属性}\}$

外码：ϕ

候选码：`id, email`

1.2 `Post` 实体

$F = \{id \to \text{其他属性}\}$

外码：`user_id, cat_id`

候选码：`id`

1.3 `PostMedia`

$F = \{id \to image\_url，id \to post\_id\}$

外码：`post_id`

候选码：`id`

1.4 `PostTag`

$F = \{id \to user\_id，id \to name\}$

外码：`user_id`

候选码：`id`

1.5 `Cat`

$F = \{id \to \text{其他属性}\}$

外码：ϕ

候选码：`id`

1.6 `CatMedia`

$F = \{id \to image\_url，id \to cat\_id\}$

外码：`cat_id`

候选码：`id`

1.7 `CatLocation`

$F = \{id \to \text{其他属性}\}$

外码：`cat_id, user_id`

候选码：`id`

1.8 `VolunteerApplication`

$F = \{id \rightarrow \text{其他属性}\}$

外码：`user_id`

候选码：`id`

1.9 `Activity`

$F = \{id \rightarrow \text{其他属性}\}$

外码：`creator_id`

候选码：`id`

1.10 `Donation`

$F = \{id \rightarrow \text{其他属性}\}$

外码：`user_id`

候选码：`id`

2. 多对多关系模式

2.1 `Like`

$F = \{(user\_id, post\_id) \rightarrow \text{其他属性}\}$

外码：`user_id, post_id`

候选码：`(user_id, post_id)`

2.2 `PostTagRelation`

$F = \{(post\_id, tag\_id) \rightarrow \text{其他属性}\}$

外码：`post_id, tag_id`

候选码：`(post_id, tag_id)`

2.3 `ActivityRegistration`

$F = \{(user\_id, activity\_id) \rightarrow \text{其他属性}\}$

外码：`user_id, activity_id`

候选码：`(user_id, activity_id)`

3. 关系模式范式等级的判定

在这部分判定过程中，很显然所有的关系模式都满足$2NF$，因此不再赘述判断过程。

实体关系的判定

- `User`：很显然不存在任何的非主属性传递依赖于其他的非主属性，且所有的非主属性都直接函数依赖于id，因此是$3NF$
- `Post`：所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖，因此属于$3NF$
- `PostMedia`：显然所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖（`post_id`与`image_url`之间没有函数依赖关系），因此属于$3NF$
- `PostTag`：显然所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖（`user_id`与`name`之间没有函数依赖关系），因此属于$3NF$
- `Cat`：所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖，因此属于$3NF$
- `CatMedia`与`CatLocation`：两者比较相似，因此放在一起来说。`cat_id`与`image_url`之间没有函数依赖关系，同样的，`cat_id, user_id`与剩下的任何非主属性之间不存在任何函数依赖，因此属于$3NF$
- `VolunteerApplication`：所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖（`user_id`与剩下的任何非主属性之间没有函数依赖），因此属于$3NF$（容易误解的一点是`user_id`与`reason`之间存在函数依赖，但实际上并没有：用户可以多次申请成为志愿者，每次申请的理由可能不同；且即便是同一用户申请志愿者，`reason`的填写也没有固定模式，可能随用户意愿而变化，完全有可能两个用户申请志愿者但是理由相同或不同）
- `Activity`：所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖，因此属于$3NF$
- `Donation`：所有的非主属性都直接函数依赖于主键id，且所有的非主属性之间不存在传递函数依赖，因此属于$3NF$，分析过程与`VolunteerApplication`相似

多对多关系的判定

- `Like`：除去联合主键外没有任何的其他属性。因此自然是$3NF$
- `PostTagRelation`：理由与上面的`Like`完全相同。
- `ActivityRegistration`：所有的非主属性都直接函数依赖于联合主键，且各非主属性之间不存在任何函数依赖，因此是$3NF$

因此，我们的所有关系表均满足$3NF$范式，最大程度上避免了插入异常与数据冗余。

附录：数据库优化设计

1. 建立索引

我们在设计关系表时，为经常用到的和需要关联进行查询的属性都加入了索引，提高了查询的速度。

例如，在`PostTag`表中，由于需要经常根据帖子查询对应的`tag_id`，或根据某一个标签查询所有对应的帖子，因此我们建立了下述索引：

```
CREATE INDEX ix_posttag_name
    ON posttag (name);

CREATE INDEX ix_posttag_user_id
    ON posttag (user_id);
```

2. 级联删除

为了减少删除表项时由于数据库约束造成的各种执行失败，也为了进一步降低操作难度，我们针对关联属性建立了级联删除。

比如在`Post`与`PostMedia`（或`Like`）中，如果`Post`表中的某一项被删除，那么对应的所有`PostMedia`也应该一并删除。于是：

```
CONSTRAINT postmedia_ibfk_1
    FOREIGN KEY (post_id) REFERENCES catstrack.post (id)
        ON DELETE CASCADE
```

3. 查询优化

- 在`SELECT`子句中只选择需要的列，避免`SELECT *`出现。
- 在进行连接操作时，尽量选取含有索引的列。