**逐梦校友圈**

**数**

**据**

**库**

**设**

**计**

**说**

**明**

**书**

组员：郑茂基，许渊博，谢语涵，蔡建斌，陈亮亮，翁炜华，

邹普凡，黄海翔，党朝媚

逐梦校友圈组 编写

2021.4.18

目录

[1. 引言 2](#_Toc30111)

[1.1 编写目的 2](#_Toc23061)

[1.2 项目背景 3](#_Toc13475)

[1.3 文档规约 3](#_Toc25755)

[1.4 预期读者 3](#_Toc20348)

[1.5 定义 4](#_Toc12657)

[1.6 参考资料 4](#_Toc11000)

[2. 数据流图 4](#_Toc2716)

[2.1 模块1数据流图 4](#_Toc18360)

[3. 结构设计 4](#_Toc31295)

[3.1 E-R图 5](#_Toc27463)

[3.2 实体属性图 6](#_Toc27593)

[3.3 数据库逻辑结构图 9](#_Toc26403)

[3.4 数据库表结构 11](#_Toc22368)

[4. 运用设计 16](#_Toc5366)

[4.1 表结构设计 16](#_Toc20924)

[4.2 安全保密设计 19](#_Toc6531)

# 引言

## 编写目的

本文档为筑梦校友圈团队开发人员关于系统数据库模型设计及相应的权限设计的详细文档，用于指导系统的开发。

具体使用人员有以下几类：系统设计人员、系统开发人员、系统测试人员、系统维护人员。

本数据库设计是根据前期《需求规格说明书》基础之上制定，作为系统开发的根基，数据库设计的好坏影响到整个系统开发的效率，以及系统的灵活性。好的数据库设计能够让方便我们的项目开发，加快我们的开发的进度，严格规定的格式也避免了数据交互时一些不必要的错误。数据库模型必须是能够满足用户的需求，支持所有的用户功能，并且拥有较高的范式，能够支持二次开发。

逐梦校友圈小程序的开发者应该最大程度地围绕这个数据库设计进行开发，依据文档进行项目设计，将它作为整个系统编码的依据。

## 项目背景

本次项目是完成的是一个微信小程序，名为“逐梦校友圈”。

随着我们迈入大学校园后，会见到许许多多校园的墙，的确给我们提供了很多我们需要的功能，例如拼车、拼单甚至可以找对象，关注人数较多的墙同时浏览数也都在5000+，具有很强的信息传播能力。但是同时我们也不得不承认，在琳琅满目的墙面前，我们使用者对于墙的具体功能很不明确。墙的机制是人工审核，每天定点发，就导致了时效性很差。qq 又有着同类型设备只支持一台设备登录，也就需要大量的人力，成为墙的门槛低，墙的种类也就因此繁多起来，同种类型的墙甚至有多个，例如福大表白墙我已知的就有4个。根据我们之前大家提供的需求分析就可以看到，我们的产品可以提供-一个很好的平台，组局功能可以消除人与人间的距离感。最后一点就是可以增强我们对校园的归属感。

开发该应用是致力于解决大学生信息搜集，拼团组局，寻求帮助存在困难等问题，节约大学生在各个平台寻找需求信息的时间，也希望可以通过互联网这个窗口，让大家看到校园的生活，可以让大家感受到原来校园中还有其他的什么样的生活的存在，希望可以借此让大家拉近人与人间的距离，感受一下大学生活更多的温暖。

主要功能：

1. 帖文展示，收藏，评论，筛选，发布
2. 组局展示，筛选，创建
3. 对话消息，用户回复评论

## 文档规约

按照以下要求书写文档

1. 标题最多三级标题

　　(1)第一级标题，宋体加粗，二号字体

　　(2)第二级标题，黑体加粗，三号字体

　　(3)第三级标题，宋体加粗，三号字体

1. 正文字体为宋体小四字体，单倍行距
2. 图标注释为宋体五号字体
3. 正文当中有小点，直接分成(1)(2)，英文括号
4. 段落首行缩进两个字符

## 预期读者

项目经理：根据此文档，制定后续开发进度

系统开发人员：根据此文档对后续开发进行指导

## 定义

1、E-R 图：实体关系图

2、MySQL：一种关系型数据库管理系统

3、数据流图：简称DFD，它从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具及用于表示软件模型的一种图示方法。

## 参考资料

1、[阿里巴巴Mysql数据库规约](https://learnku.com/laravel/t/45456)

2、[数据库设计说明书编写规范(国标)](https://wenku.baidu.com/view/fcccbc33168884868662d625.html)

3、数据库系统概论（第四版）  王珊，萨师煊编著 高等教育出版社

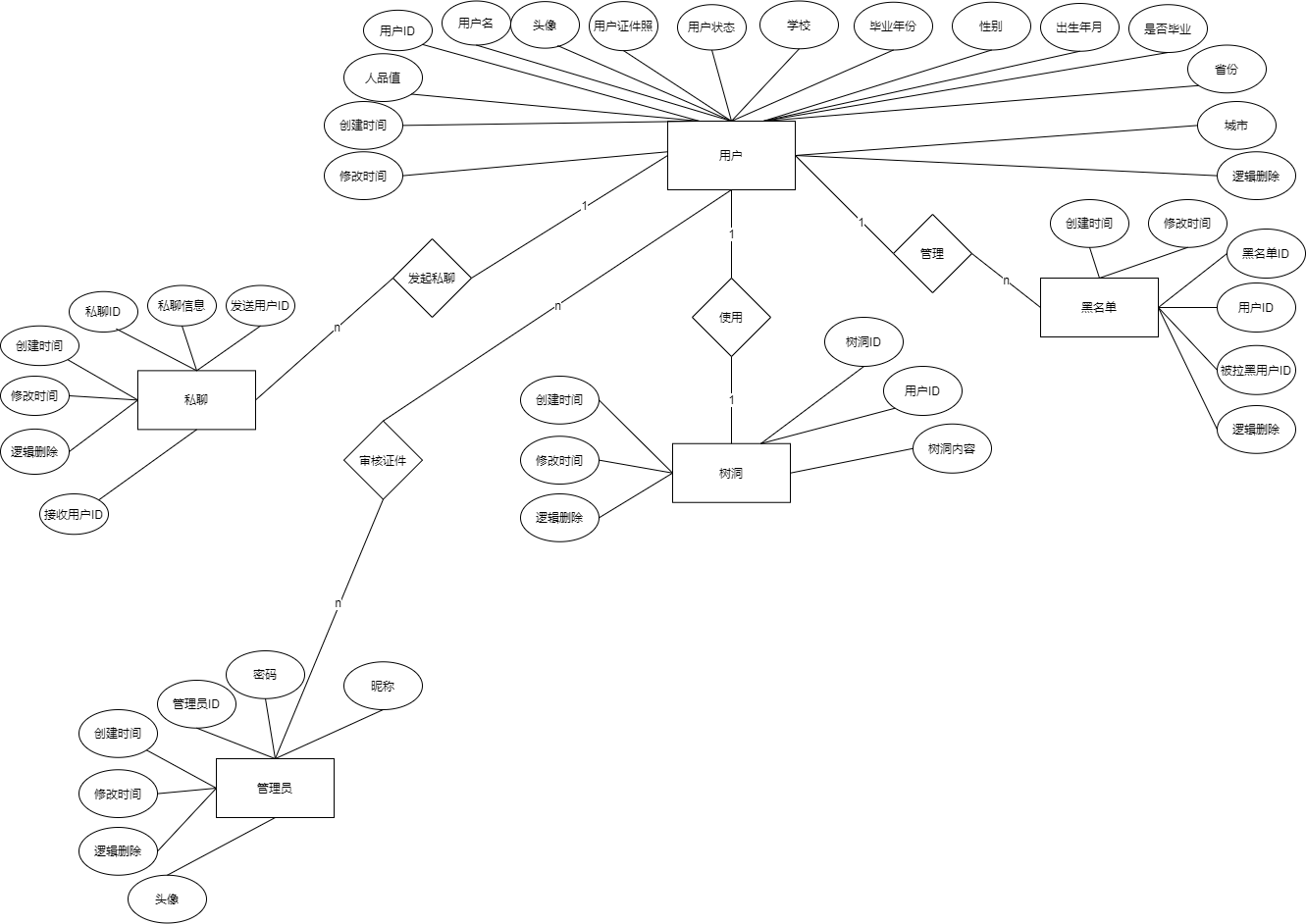
4、构建之法（第三版） 邹欣编著 人民邮电出版社

# 数据流图

## 2.1 模块1数据流图

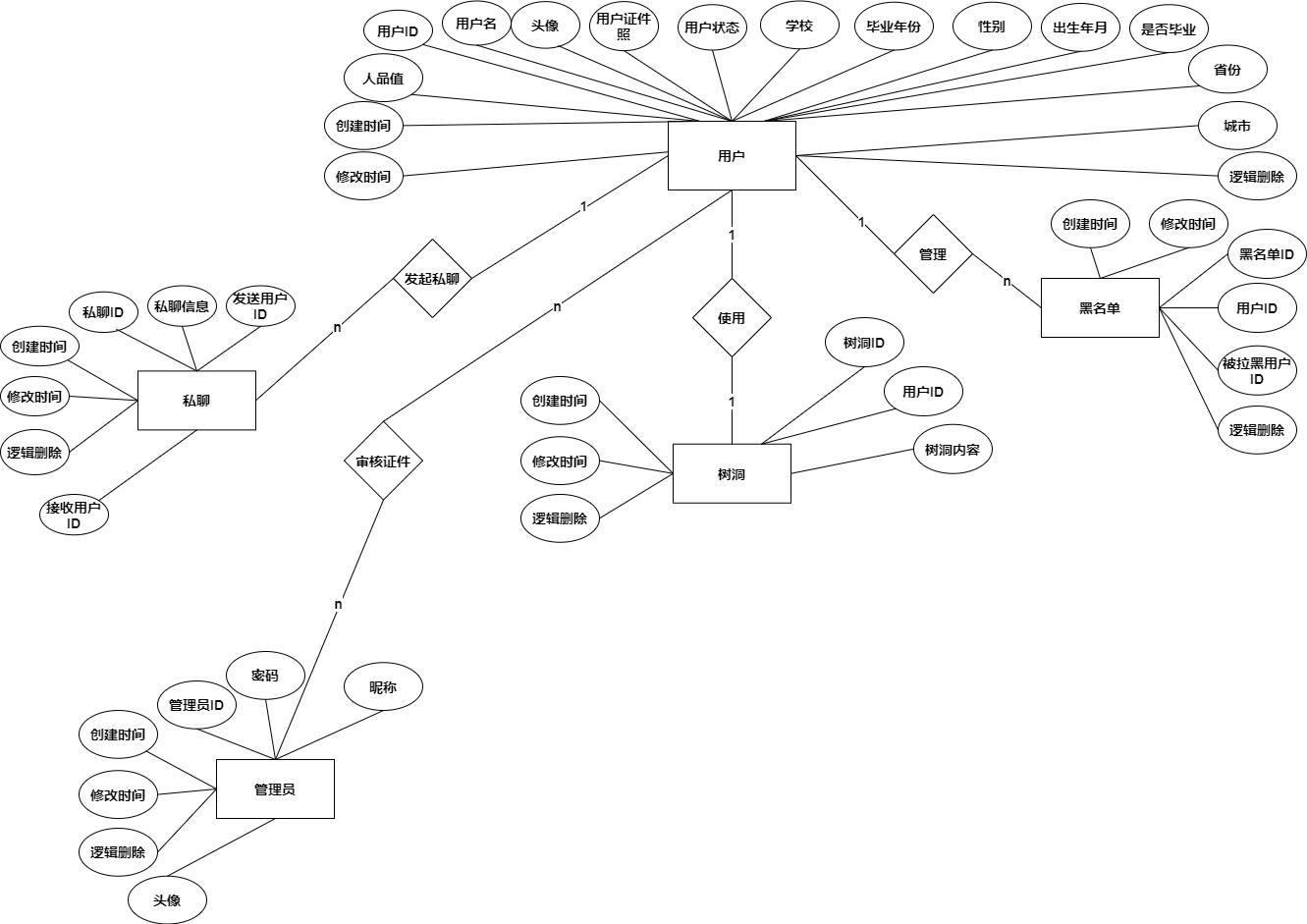
# 结构设计

## 3.1 E-R图

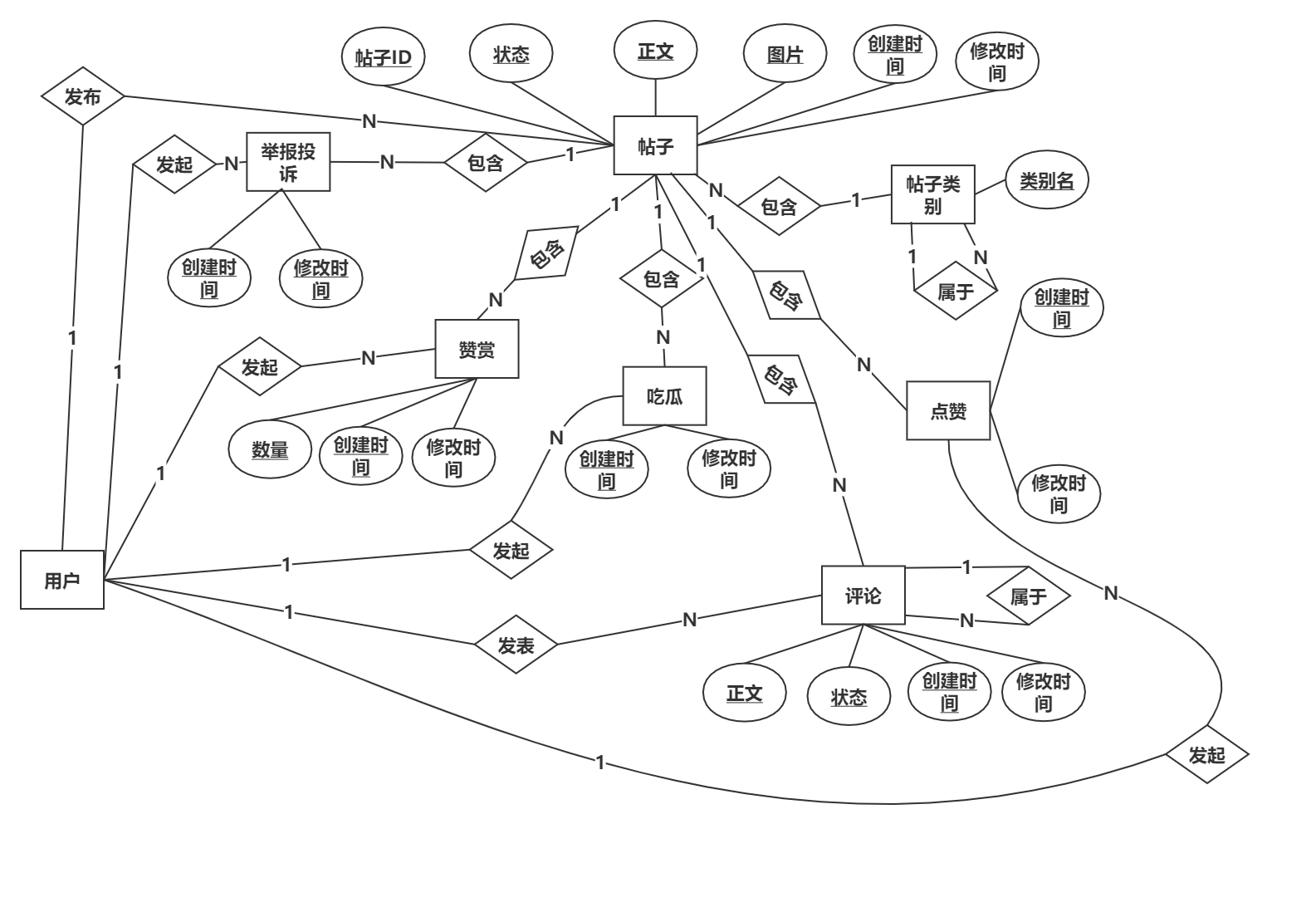


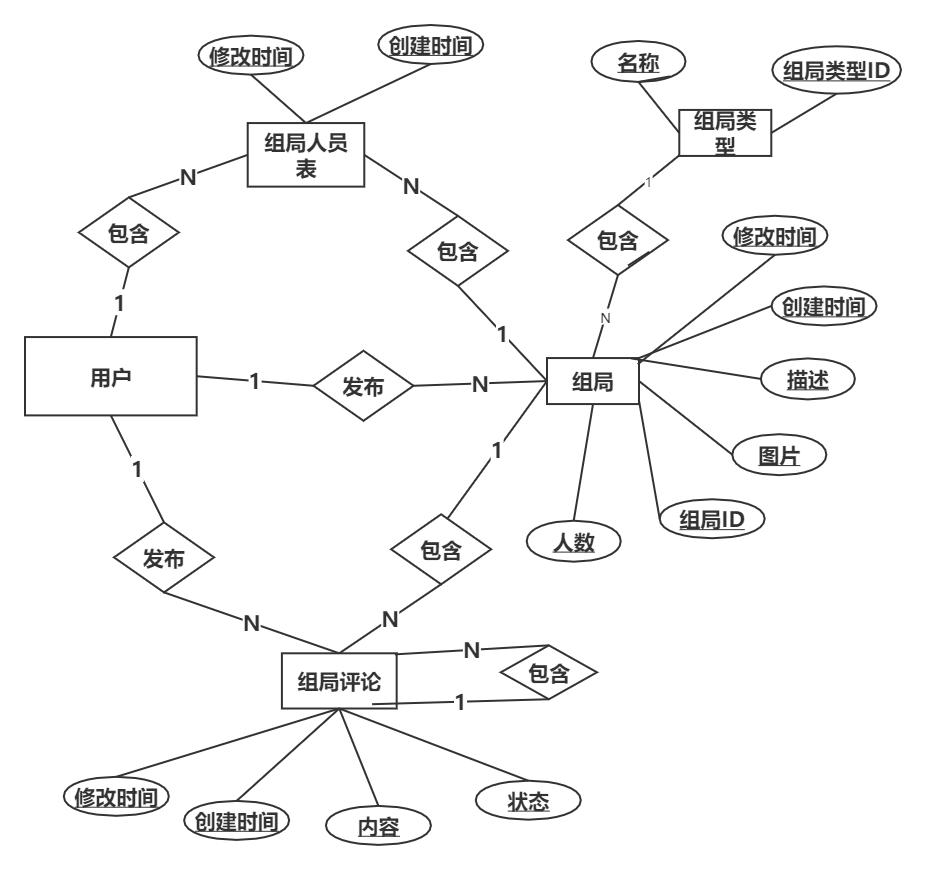
## 3.2 实体属性图

### 3.2.1 用户(user)和管理员(admin)部分



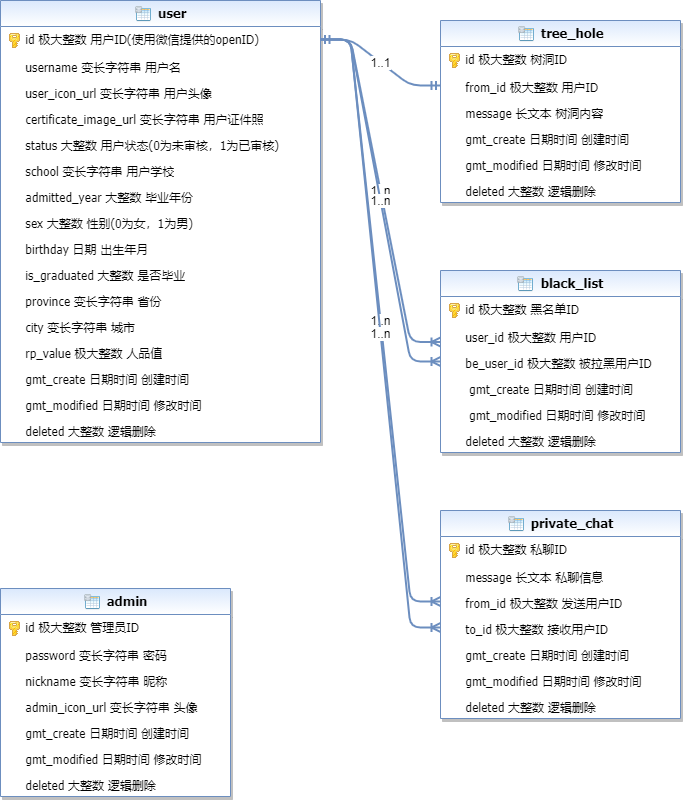
### 3.2.2 帖文(post)部分



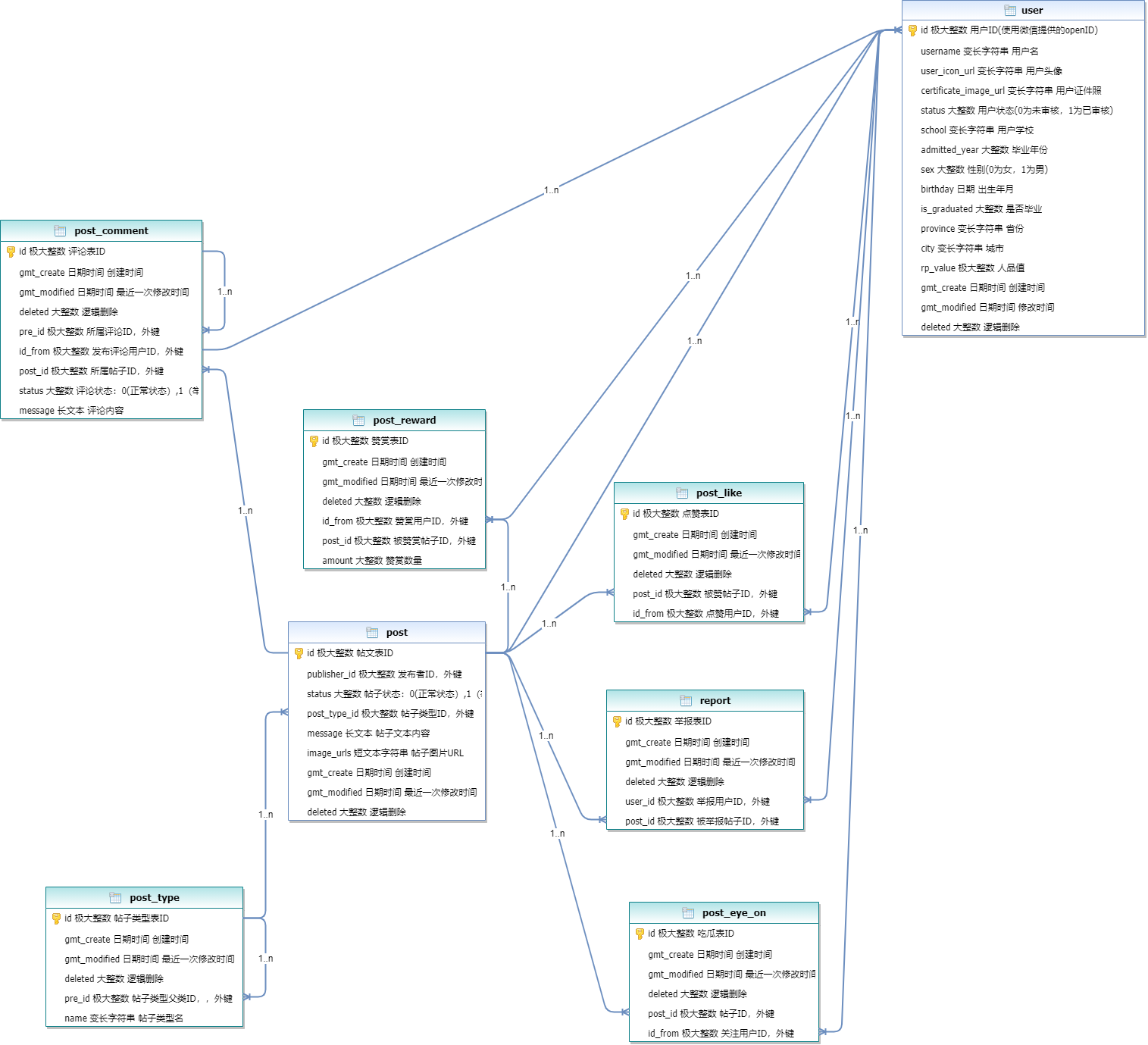


## 3.3 数据库逻辑结构图

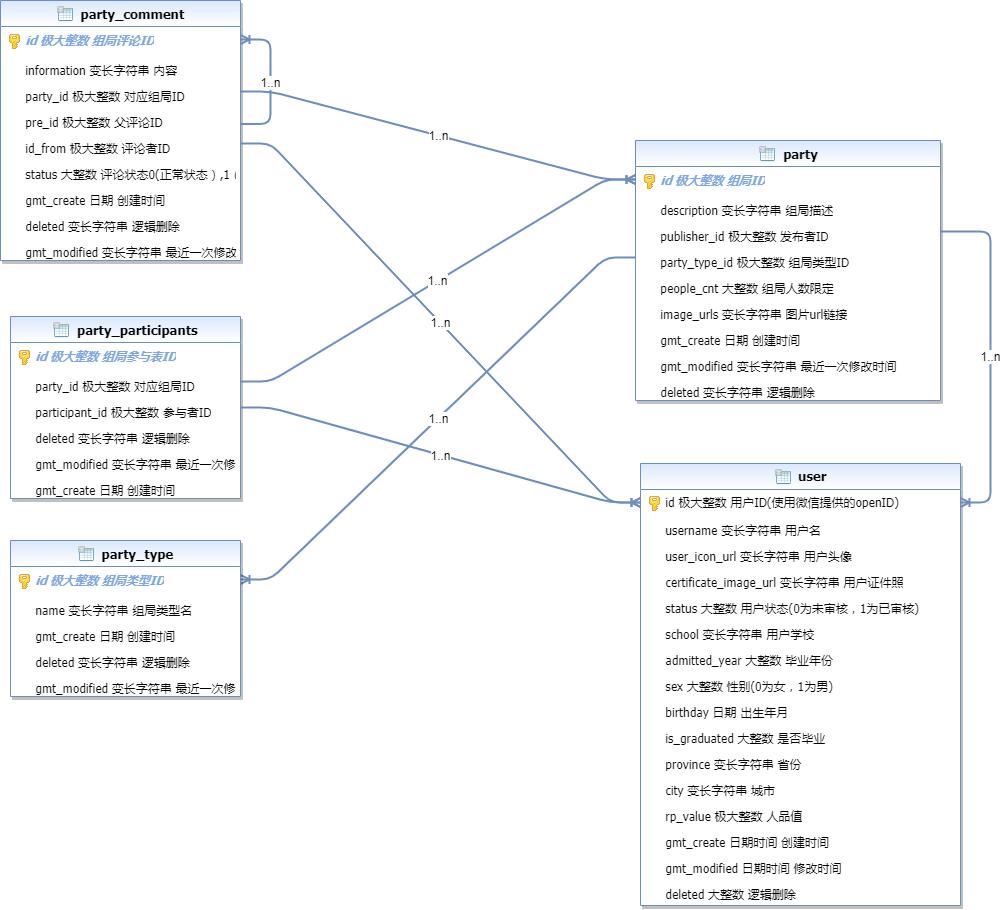
### 3.3.1 用户（user)和管理员（admin）部分



### 3.3.2 贴文（post）部分



### 3.3.3 组局（party）部分



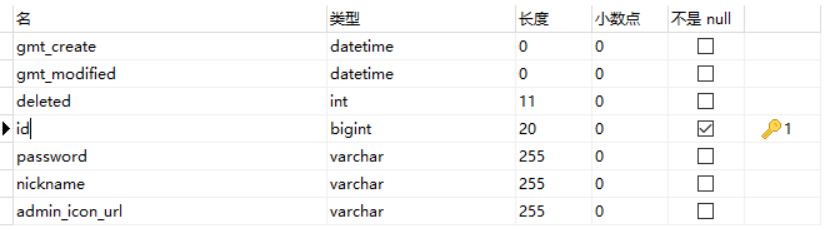
## 3.4 数据库表结构

根据阿里巴巴java开发手册-数据库部分，每个表都有id、gmt\_create、gmt\_modified字段，以及增加了用于逻辑删除的deleted字段。

#### 3.4.1 用户表 User



#### 3.4.2 管理员表Admin



#### 3.4.3 黑名单表Black\_list



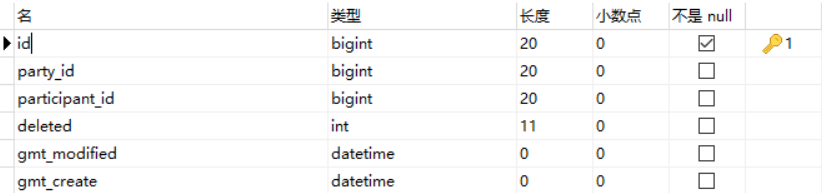
#### 3.4.4 组局表Party



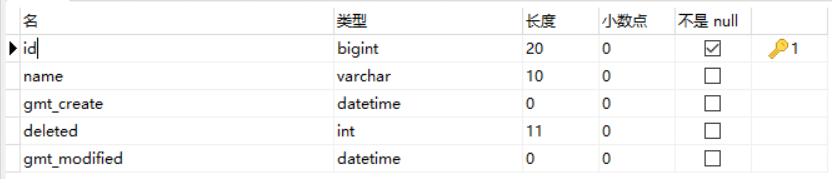
### 3.4.5 组局评论表Party\_comment



### 3.4.6 组局成员表Party\_participants



### 3.4.7 组局类型表Party\_type



### 3.4.8 帖文表Post



### 3.4.9 帖文评论表Post\_comment



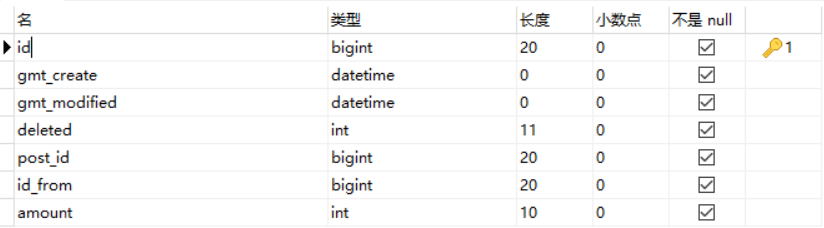
### 3.4.10 帖文审核表Post\_eye\_on



### 3.4.11 帖文收藏表Post\_like



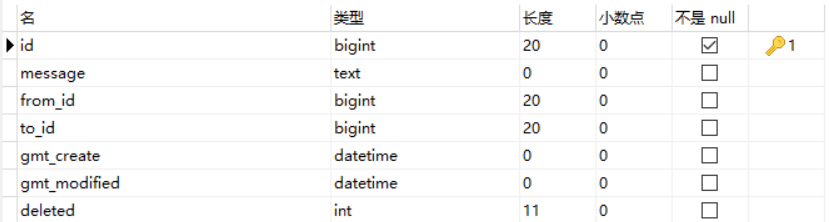
### 3.4.12 帖文赞赏表Post\_reward



### 3.4.13 帖文类型表Post\_type



### 3.4.14 私聊表Private\_chat



### 3.4.15 举报表Report



### 3.4.16 树洞表Tree\_hole



# 运用设计

# 4.1 表结构设计

### 4.1.1 约定

1、表名、字段名必须使用小写字母或数字；禁止出现数字开头，禁止两个下划线中间只出现数字。

2、禁用保留字，如 desc、range、match、delayed 等，请参考 MySQL 官方保留字。

3、表必备三字段：id, gmt\_create, gmt\_modified。

4、表的命名最好是加上 “业务名称\_表的作用”。

5、字符集采用 “UTF-8”   
6、不允许存储明文密码

7、如果修改字段含义或对字段表示的状态追加时，需要及时更新字段注释。

### 4.1.2 数据库建表

在数据库设计中，我们将数据库分为用户、贴文和组局三部分分开设计，之后再进行整合。这样设计更容易把握细节，且成员之间分工更加明确，效率更高。

|  |  |  |  |
| --- | --- | --- | --- |
| **表名** | **标识符** | **描述信息** | **状态** |
| 用户表 | user | 存放用户的基础信息 | 开发使用中 |
| 管理员表 | admin | 存放管理员的基础信息 | 开发使用中 |
| 树洞表 | tree\_hole | 存放树洞模块信息 | 开发使用中 |
| 私聊表 | private\_chat | 记录私聊信息 | 开发使用中 |
| 黑名单表 | black\_list | 记录黑名单信息 | 开发使用中 |
| 组局表 | party | 记录组局信息 | 开发使用中 |
| 组局类别表 | party\_type | 记录组局类别 | 开发使用中 |
| 组局评论表 | party\_comment | 记录组局的评论记录 | 开发使用中 |
| 组局参与人员表 | party\_participants | 记录组局的参与人员 | 开发使用中 |
| 贴文表 | post | 记录贴文信息 | 开发使用中 |
| 打赏表 | post\_reward | 打赏人品值记录表 | 开发使用中 |
| 贴文类别表 | post\_type | 记录贴文的类别 | 开发使用中 |
| 点赞表 | post\_like | 记录贴文的点赞情况 | 开发使用中 |
| 举报表 | report | 记录贴文的举报信息 | 开发使用中 |
| 贴文评论表 | post\_comment | 记录贴文的评论信息 | 开发使用中 |
| 贴文关注表 | post\_eye\_on | 记录贴文的关注情况 | 开发使用中 |

### 4.1.3 建表语句

CREATE TABLE IF NOT EXISTS user ( id BIGINT PRIMARY KEY comment '用户ID(使用微信提供的openID)',

username VARCHAR(255) comment '用户名',

user\_icon\_url VARCHAR(255) comment '用户头像',

certificate\_image\_url VARCHAR(255) comment '用户证件照',

status INT default 0 comment '用户状态(0为未审核，1为已审核)',

school VARCHAR(255) comment '用户学校',

admitted\_year INT comment '毕业年份',

sex INT comment '性别(0为女，1为男)',

birthday DATE comment '出生年月',

is\_graduated INT comment '是否毕业',

province VARCHAR(255) comment '省份',

city VARCHAR(255) comment '城市',

rp\_value BIGINT default 0 comment '人品值',

gmt\_create DATETIME default CURRENT\_TIMESTAMP comment '创建时间',

gmt\_modified DATETIME default CURRENT\_TIMESTAMP comment '修改时间',

deleted INT default 0 comment '逻辑删除'

);

CREATE TABLE IF NOT EXISTS tree\_hole ( id BIGINT AUTO\_INCREMENT PRIMARY KEY comment '树洞ID',

from\_id BIGINT comment '用户ID',

message TEXT comment '树洞内容',

gmt\_create DATETIME default CURRENT\_TIMESTAMP comment '创建时间',

gmt\_modified DATETIME default CURRENT\_TIMESTAMP comment '修改时间',

deleted INT default 0 comment '逻辑删除'

);

alter table tree\_hole add constraint tree\_hole\_from\_id\_fk0 foreign key (from\_id) references user (id);

CREATE TABLE IF NOT EXISTS black\_list ( id BIGINT AUTO\_INCREMENT PRIMARY KEY comment '黑名单ID',

user\_id BIGINT comment '用户ID',

be\_user\_id BIGINT comment '被拉黑用户ID',

gmt\_create DATETIME default CURRENT\_TIMESTAMP comment '创建时间',

gmt\_modified DATETIME default CURRENT\_TIMESTAMP comment '修改时间',

deleted INT default 0 comment '逻辑删除'

);

alter table black\_list add constraint black\_list\_user\_id\_fk0 foreign key (user\_id) references user (id);

alter table black\_list add constraint black\_list\_be\_user\_id\_fk0 foreign key (be\_user\_id) references user (id);

CREATE TABLE IF NOT EXISTS private\_chat ( id BIGINT AUTO\_INCREMENT PRIMARY KEY comment '私聊ID',

message TEXT comment '私聊信息',

from\_id BIGINT comment '发送用户ID',

to\_id BIGINT comment '接收用户ID',

gmt\_create DATETIME default CURRENT\_TIMESTAMP comment '创建时间',

gmt\_modified DATETIME default CURRENT\_TIMESTAMP comment '修改时间',

deleted INT default 0 comment '逻辑删除'

);

alter table private\_chat add constraint private\_chat\_from\_id\_fk0 foreign key (from\_id) references user (id);

alter table private\_chat add constraint private\_chat\_to\_id\_fk0 foreign key (to\_id) references user (id);

CREATE TABLE IF NOT EXISTS admin ( id BIGINT AUTO\_INCREMENT PRIMARY KEY comment '管理员ID',

password VARCHAR(255) comment '密码',

nickname VARCHAR(255) comment '昵称',

admin\_icon\_url VARCHAR(255) comment '头像',

gmt\_create DATETIME default CURRENT\_TIMESTAMP comment '创建时间',

gmt\_modified DATETIME default CURRENT\_TIMESTAMP comment '修改时间',

deleted INT default 0 comment '逻辑删除'

);

DROP TABLE IF EXISTS `post`;

CREATE TABLE `post` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT '帖文ID',

`publisher\_id` bigint(0) NULL DEFAULT NULL COMMENT '发布者ID，外键',

`status` int(0) NOT NULL COMMENT '帖子状态：0(正常状态）,1（举报过多被挂起），2（已被删除）',

`post\_type\_id` bigint(0) NOT NULL COMMENT '帖子类型ID，外键',

`message` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL COMMENT '帖子文本内容',

`image\_urls` tinytext CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL COMMENT '帖子图片URL',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

PRIMARY KEY (`id`) USING BTREE,

INDEX `post\_publisher\_id\_fk0`(`publisher\_id`) USING BTREE,

INDEX `post\_post\_type\_id\_fk0`(`post\_type\_id`) USING BTREE,

CONSTRAINT `post\_post\_type\_id\_fk0` FOREIGN KEY (`post\_type\_id`) REFERENCES `post\_type` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `post\_publisher\_id\_fk0` FOREIGN KEY (`publisher\_id`) REFERENCES `user` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

DROP TABLE IF EXISTS `post\_comment`;

CREATE TABLE `post\_comment` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

`pre\_id` bigint(0) NOT NULL COMMENT '所属评论ID，外键',

`post\_id` bigint(0) NOT NULL COMMENT '所属帖子ID，外键',

`id\_from` bigint(0) NOT NULL COMMENT '发布评论用户ID，外键',

`status` int(0) UNSIGNED NOT NULL COMMENT '评论状态：0(正常状态）,1（举报过多被挂起），2（已被删除）',

`message` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL COMMENT '评论正文',

PRIMARY KEY (`id`) USING BTREE,

INDEX `post\_comment\_pre\_id\_fk0`(`pre\_id`) USING BTREE,

INDEX `post\_comment\_post\_id\_fk0`(`post\_id`) USING BTREE,

INDEX `post\_comment\_id\_from\_fk0`(`id\_from`) USING BTREE,

CONSTRAINT `post\_comment\_id\_from\_fk0` FOREIGN KEY (`id\_from`) REFERENCES `user` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `post\_comment\_post\_id\_fk0` FOREIGN KEY (`post\_id`) REFERENCES `post` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `post\_comment\_pre\_id\_fk0` FOREIGN KEY (`pre\_id`) REFERENCES `post\_comment` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

DROP TABLE IF EXISTS `post\_eye\_on`;

CREATE TABLE `post\_eye\_on` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

`post\_id` bigint(0) NOT NULL COMMENT '帖子ID，外键',

`id\_from` bigint(0) NOT NULL COMMENT '关注用户ID，外键',

PRIMARY KEY (`id`) USING BTREE,

INDEX `post\_eye\_on\_post\_id\_fk0`(`post\_id`) USING BTREE,

INDEX `post\_eye\_on\_id\_from\_fk0`(`id\_from`) USING BTREE,

CONSTRAINT `post\_eye\_on\_id\_from\_fk0` FOREIGN KEY (`id\_from`) REFERENCES `user` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `post\_eye\_on\_post\_id\_fk0` FOREIGN KEY (`post\_id`) REFERENCES `post` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

DROP TABLE IF EXISTS `post\_like`;

CREATE TABLE `post\_like` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

`post\_id` bigint(0) NOT NULL COMMENT '被赞帖子ID，外键',

`id\_from` bigint(0) NOT NULL COMMENT '点赞用户ID，外键',

PRIMARY KEY (`id`) USING BTREE,

INDEX `post\_like\_post\_id\_fk0`(`post\_id`) USING BTREE,

INDEX `post\_like\_id\_from\_fk0`(`id\_from`) USING BTREE,

CONSTRAINT `post\_like\_id\_from\_fk0` FOREIGN KEY (`id\_from`) REFERENCES `user` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `post\_like\_post\_id\_fk0` FOREIGN KEY (`post\_id`) REFERENCES `post` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

DROP TABLE IF EXISTS `post\_reward`;

CREATE TABLE `post\_reward` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

`post\_id` bigint(0) NOT NULL COMMENT '被赞赏帖子ID，外键',

`id\_from` bigint(0) NOT NULL COMMENT '赞赏用户ID，外键',

`amount` int(0) UNSIGNED NOT NULL COMMENT '赞赏数量',

PRIMARY KEY (`id`) USING BTREE,

INDEX `post\_reward\_post\_id\_fk0`(`post\_id`) USING BTREE,

CONSTRAINT `post\_reward\_post\_id\_fk0` FOREIGN KEY (`post\_id`) REFERENCES `post` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

DROP TABLE IF EXISTS `post\_type`;

CREATE TABLE `post\_type` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

`pre\_id` bigint(0) NOT NULL COMMENT '帖子类型父类ID，外键',

`name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL COMMENT '帖子类型名',

PRIMARY KEY (`id`) USING BTREE,

INDEX `post\_type\_pre\_id\_fk0`(`pre\_id`) USING BTREE,

CONSTRAINT `post\_type\_pre\_id\_fk0` FOREIGN KEY (`pre\_id`) REFERENCES `post\_type` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

DROP TABLE IF EXISTS `report`;

CREATE TABLE `report` (

`id` bigint(0) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`gmt\_create` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '创建时间',

`gmt\_modified` datetime(0) NOT NULL DEFAULT CURRENT\_TIMESTAMP(0) COMMENT '最近一次修改时间',

`deleted` int(0) NOT NULL DEFAULT 0 COMMENT '逻辑删除',

`user\_id` bigint(0) NOT NULL COMMENT '举报用户ID，外键',

`post\_id` bigint(0) NOT NULL COMMENT '被举报帖子ID，外键',

PRIMARY KEY (`id`) USING BTREE,

INDEX `report\_user\_id\_fk0`(`user\_id`) USING BTREE,

INDEX `report\_post\_id\_fk0`(`post\_id`) USING BTREE,

CONSTRAINT `report\_post\_id\_fk0` FOREIGN KEY (`post\_id`) REFERENCES `post` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `report\_user\_id\_fk0` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci ROW\_FORMAT = Dynamic;

## 4.2 安全保密设计

后端设置过滤机制，使用过滤器对没有注册登录用户的请求进行拦截，不予放行，防

止非法用户恶意操作，只有经过常规途径注册并登录的用户才能使用系统进行数据操作。登陆注册时对电话号码，验证码和密码等信息进行严格验证，只有信息合法通过验证才能注册或登录成功。

后台设置拦截器防止同一 IP 在短时间内进行大量的恶意请求，造成服务器资源紧张，

瘫痪的现象。

后端对前端提交的数据进行校验，防止有用户通过非正规途径使用恶意的数据包对服

务器发送请求造成服务器端出现异常。

 数据库安全性，当数据库增删改操作异常时，对当前修改请求进行撤回，保证数据的安全性和完整性。

对登录用户，游客用户和管理员进行权限分配，不得越权操作，例如游客不能进行点

赞操作。

对数据库进行数据操作时需要注意以下几个安全问题：

访问安全 、网络安全 、传输安全 、备份安全 、数据安全 对数据库进行定时维护。