# 业务逻辑

**1. 主题/话题系统**

* 用户可在某一主题下发帖（如NBA、CBA、影视、数码）。
* 每个帖子可关联一个或多个话题（Tag系统）。
* 主题可以有层级（如 “运动” > “篮球” > “NBA”）。

**2. 帖子（Post）**

* 帖子类型多样：图文、视频、投票、评分帖等。
* 帖子可被用户评论、点赞、收藏。
* 帖子可关联评分对象（如球员、电影、手机等）。

**3. 用户系统**

* 用户拥有唯一身份，可发帖、评分、评论。
* 有用户等级、声望系统、勋章、关注等功能。
* 用户行为会被记录：浏览、发帖、打分、举报等。

**4. 评分系统**

* 用户可以对评分对象打分（0-10分或五分制等）。
* 可按不同维度打分（如球员：得分、助攻、防守）。
* 支持查看平均分、打分人数、分布图等统计信息。

**5. 评分对象**

* 可以是球员、球队、电影、数码产品等。
* 每种对象类型有自己的属性（如球员：位置、身高）。

**6. 评论系统**

* 评论可嵌套（楼中楼结构）。
* 评论可点赞、举报、引用。

**7. 社交功能**

* 用户可以关注其他用户，看到关注者动态。
* 私信系统（可选）。
* 用户间互动影响推荐算法。

**8. 管理与内容审核**

* 举报与违规管理。
* 管理员可以置顶、删除帖子。
* 内容标签如“优质”、“热帖”等。

**9. 推荐与热门机制**

* 基于点赞数、评论数、阅读量进行热度排行。
* 推荐系统基于用户兴趣画像。

| **实体** | **描述** | **主要属性** |
| --- | --- | --- |
| User | 用户 | id, 昵称, 等级, 注册时间, 积分, 粉丝数 |
| Post | 帖子 | id, user\_id, 内容, 发布时间, 类型, 浏览数, 点赞数 |
| Topic | 主题/分类 | id, 名称, 父主题id |
| Tag | 标签（话题） | id, 名称 |
| PostTag | 帖子-标签关联 | post\_id, tag\_id |
| Comment | 评论 | id, post\_id, user\_id, content, parent\_comment\_id |
| ScoreTarget | 被评分对象 | id, 名称, 类型（球员、电影、产品等） |
| ScoreMetric | 评分维度 | id, name（如“得分能力”、“颜值”） |
| ScoreRecord | 用户评分记录 | user\_id, target\_id, metric\_id, score, 时间戳 |
| UserFollow | 用户关注关系 | follower\_id, followee\_id |
| UserActionLog | 行为记录 | user\_id, 动作类型, 目标id, 时间戳 |
| FavoritePost | 收藏记录 | user\_id, post\_id |
| Like | 点赞 | user\_id, target\_type, target\_id |
| Report | 举报记录 | user\_id, target\_type, target\_id, 理由 |
| Admin | 管理员 | user\_id, 权限等级 |

# ER图？



# SQL建表

以下是为上面 ER 模型生成的 **SQL 建表语句（MySQL/PostgreSQL 通用风格）**，涵盖用户系统、帖子系统、评论系统、评分系统、标签与话题、内容管理等完整模块。字段类型和外键关系尽量详细且可运行。

**🧱 用户相关表**

CREATE TABLE User (

id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

nickname VARCHAR(50),

password\_hash VARCHAR(255) NOT NULL,

email VARCHAR(100) UNIQUE,

level INT DEFAULT 1,

reputation INT DEFAULT 0,

avatar\_url TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE UserFollow (

follower\_id INT NOT NULL,

followee\_id INT NOT NULL,

followed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (follower\_id, followee\_id),

FOREIGN KEY (follower\_id) REFERENCES User(id),

FOREIGN KEY (followee\_id) REFERENCES User(id)

);

**📝 帖子与评论系统**

CREATE TABLE Topic (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

parent\_id INT,

FOREIGN KEY (parent\_id) REFERENCES Topic(id)

);

CREATE TABLE Post (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

title VARCHAR(255),

content TEXT,

post\_type VARCHAR(20),

topic\_id INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

view\_count INT DEFAULT 0,

like\_count INT DEFAULT 0,

comment\_count INT DEFAULT 0,

FOREIGN KEY (user\_id) REFERENCES User(id),

FOREIGN KEY (topic\_id) REFERENCES Topic(id)

);

CREATE TABLE Comment (

id SERIAL PRIMARY KEY,

post\_id INT NOT NULL,

user\_id INT NOT NULL,

parent\_comment\_id INT,

content TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

like\_count INT DEFAULT 0,

FOREIGN KEY (post\_id) REFERENCES Post(id),

FOREIGN KEY (user\_id) REFERENCES User(id),

FOREIGN KEY (parent\_comment\_id) REFERENCES Comment(id)

);

CREATE TABLE FavoritePost (

user\_id INT,

post\_id INT,

favorited\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (user\_id, post\_id),

FOREIGN KEY (user\_id) REFERENCES User(id),

FOREIGN KEY (post\_id) REFERENCES Post(id)

);

CREATE TABLE Like (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

target\_type VARCHAR(20), -- post, comment

target\_id INT,

liked\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES User(id)

);

**🏷 标签系统**

CREATE TABLE Tag (

id SERIAL PRIMARY KEY,

name VARCHAR(50) UNIQUE NOT NULL

);

CREATE TABLE PostTag (

post\_id INT NOT NULL,

tag\_id INT NOT NULL,

PRIMARY KEY (post\_id, tag\_id),

FOREIGN KEY (post\_id) REFERENCES Post(id),

FOREIGN KEY (tag\_id) REFERENCES Tag(id)

);

**⭐️ 评分系统**

CREATE TABLE ScoreTarget (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

type VARCHAR(50), -- player, movie, product, etc.

description TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE ScoreMetric (

id SERIAL PRIMARY KEY,

name VARCHAR(50) NOT NULL,

target\_type VARCHAR(50) NOT NULL -- 应用于哪类评分对象

);

CREATE TABLE ScoreRecord (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

target\_id INT NOT NULL,

metric\_id INT NOT NULL,

score FLOAT CHECK (score >= 0 AND score <= 10),

comment TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES User(id),

FOREIGN KEY (target\_id) REFERENCES ScoreTarget(id),

FOREIGN KEY (metric\_id) REFERENCES ScoreMetric(id)

);

**🚨 管理与举报系统**

CREATE TABLE Report (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

target\_type VARCHAR(20), -- post, comment

target\_id INT,

reason VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES User(id)

);

CREATE TABLE Admin (

user\_id INT PRIMARY KEY,

role VARCHAR(50), -- moderator, super\_admin

FOREIGN KEY (user\_id) REFERENCES User(id)

);

**📊 用户行为日志**

CREATE TABLE UserActionLog (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

action\_type VARCHAR(20), -- view, like, comment, score

target\_type VARCHAR(20),

target\_id INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES User(id)

);

# 简单API

以下是基于上述数据库模型设计的 **API 接口文档**，包含基本的功能模块，如用户认证、帖子管理、评分系统、评论和标签管理等。接口设计遵循 RESTful 风格，并提供了常见的请求方法（GET、POST、PUT、DELETE）和响应格式。

**📝 API 接口文档**

**🧑‍💻 用户模块**

**1. 用户注册**

* **URL**: /api/v1/users/register
* **方法**: POST
* **请求体**:
* {
* "username": "string",
* "password": "string",
* "email": "string",
* "nickname": "string"
* }
* **响应**:
  + 成功:
  + {
  + "message": "User registered successfully",
  + "user\_id": 123
  + }
  + 失败（如用户名已存在）:
  + {
  + "error": "Username already taken"
  + }

**2. 用户登录**

* **URL**: /api/v1/users/login
* **方法**: POST
* **请求体**:
* {
* "username": "string",
* "password": "string"
* }
* **响应**:
  + 成功:
  + {
  + "token": "JWT\_TOKEN",
  + "user\_id": 123
  + }
  + 失败（如密码错误）:
  + {
  + "error": "Invalid username or password"
  + }

**3. 获取用户信息**

* **URL**: /api/v1/users/{user\_id}
* **方法**: GET
* **响应**:
  + 成功:
  + {
  + "id": 123,
  + "username": "user1",
  + "nickname": "User One",
  + "email": "user@example.com",
  + "level": 1,
  + "reputation": 100
  + }

**4. 更新用户信息**

* **URL**: /api/v1/users/{user\_id}
* **方法**: PUT
* **请求体**:
* {
* "nickname": "New Nickname",
* "email": "newemail@example.com",
* "avatar\_url": "https://example.com/avatar.jpg"
* }
* **响应**:
  + 成功:
  + {
  + "message": "User information updated successfully"
  + }

**5. 关注用户**

* **URL**: /api/v1/users/{user\_id}/follow
* **方法**: POST
* **请求体**:
* {
* "followee\_id": 456
* }
* **响应**:
  + 成功:
  + {
  + "message": "Followed user successfully"
  + }

**6. 取消关注用户**

* **URL**: /api/v1/users/{user\_id}/unfollow
* **方法**: DELETE
* **请求体**:
* {
* "followee\_id": 456
* }
* **响应**:
  + 成功:
  + {
  + "message": "Unfollowed user successfully"
  + }

**📝 帖子模块**

**1. 创建帖子**

* **URL**: /api/v1/posts
* **方法**: POST
* **请求体**:
* {
* "user\_id": 123,
* "title": "Post Title",
* "content": "Post content here...",
* "post\_type": "text", // "text", "image", "video"
* "topic\_id": 1,
* "tags": ["basketball", "NBA"]
* }
* **响应**:
  + 成功:
  + {
  + "message": "Post created successfully",
  + "post\_id": 789
  + }

**2. 获取帖子信息**

* **URL**: /api/v1/posts/{post\_id}
* **方法**: GET
* **响应**:
  + 成功:
  + {
  + "id": 789,
  + "user\_id": 123,
  + "title": "Post Title",
  + "content": "Post content here...",
  + "post\_type": "text",
  + "topic": "Basketball",
  + "tags": ["basketball", "NBA"],
  + "created\_at": "2025-04-10T10:00:00Z",
  + "view\_count": 500,
  + "like\_count": 100,
  + "comment\_count": 20
  + }

**3. 更新帖子**

* **URL**: /api/v1/posts/{post\_id}
* **方法**: PUT
* **请求体**:
* {
* "title": "Updated Post Title",
* "content": "Updated content here...",
* "tags": ["NBA", "players"]
* }
* **响应**:
  + 成功:
  + {
  + "message": "Post updated successfully"
  + }

**4. 删除帖子**

* **URL**: /api/v1/posts/{post\_id}
* **方法**: DELETE
* **响应**:
  + 成功:
  + {
  + "message": "Post deleted successfully"
  + }

**5. 获取帖子评论**

* **URL**: /api/v1/posts/{post\_id}/comments
* **方法**: GET
* **响应**:
  + 成功:
  + [
  + {
  + "id": 1,
  + "user\_id": 123,
  + "content": "Great post!",
  + "created\_at": "2025-04-10T10:15:00Z",
  + "like\_count": 10
  + },
  + {
  + "id": 2,
  + "user\_id": 124,
  + "content": "I disagree with your point.",
  + "created\_at": "2025-04-10T10:20:00Z",
  + "like\_count": 2
  + }
  + ]

**6. 发表评论**

* **URL**: /api/v1/posts/{post\_id}/comments
* **方法**: POST
* **请求体**:
* {
* "user\_id": 125,
* "content": "Interesting perspective!"
* }
* **响应**:
  + 成功:
  + {
  + "message": "Comment posted successfully",
  + "comment\_id": 3
  + }

**7. 点赞帖子**

* **URL**: /api/v1/posts/{post\_id}/like
* **方法**: POST
* **响应**:
  + 成功:
  + {
  + "message": "Post liked successfully"
  + }

**📝 评分模块**

**1. 为评分对象打分**

* **URL**: /api/v1/scores
* **方法**: POST
* **请求体**:
* {
* "user\_id": 123,
* "target\_id": 456,
* "metric\_id": 1,
* "score": 8.5,
* "comment": "Great player!"
* }
* **响应**:
  + 成功:
  + {
  + "message": "Score submitted successfully"
  + }

**2. 获取评分对象的评分**

* **URL**: /api/v1/scores/{target\_id}
* **方法**: GET
* **响应**:
  + 成功:
  + {
  + "target\_id": 456,
  + "average\_score": 8.2,
  + "score\_count": 50,
  + "score\_details": [
  + {
  + "metric": "Defense",
  + "average\_score": 8.0
  + },
  + {
  + "metric": "Offense",
  + "average\_score": 8.4
  + }
  + ]
  + }

**📝 标签与话题模块**

**1. 获取标签**

* **URL**: /api/v1/tags
* **方法**: GET
* **响应**:
  + 成功:
  + [
  + {
  + "id": 1,
  + "name": "Basketball"
  + },
  + {
  + "id": 2,
  + "name": "NBA"
  + }
  + ]

**2. 获取话题列表**

* **URL**: /api/v1/topics
* **方法**: GET
* **响应**:
  + 成功:
  + [
  + {
  + "id": 1,
  + "name": "Basketball",
  + "parent\_id": null
  + },
  + {
  + "id": 2,
  + "name": "NBA",
  + "parent\_id": 1
  + }
  + ]

这个文档是 API 的一个基本设计，你可以根据需求调整字段、添加更多的功能，或者修改响应和请求的格式。是否有更多功能需要扩展？