附件 1

《数据库系统原理》大作业

系统设计报告

题目名称：　　NBA 比赛数据管理系统

学号及姓名：23373105 刘锦涛

23373502 王晨希

年　　月　　日

组内同学承担任务说明

学生姓名	工作内容		工作量占比（组内同学总和为 1）
	子任务 1：系统功能设计与 XXX 功能数据库操作实现	子任务 2：系统数据库设计与 XXX 功能数据库操作实现	
刘锦涛	系统功能规划（需求分析、模块划分、交互 / 跳转逻辑、接口规范）、权限体系设计、Vue3 前端架构搭建、12 个核心页面开发、数据可视化实现、46 个 API 对接测试		0.5
王晨希		数据库设计（18 张数据表、5 个触发器、56 个存储过程、索引优化）、Flask 后端架构搭建、10 个路由模块开发、权限控制与数据交互实现、API 文档输出	0.5

一．需求分析

1．需求描述

本项目实现了一个 NBA 比赛数据管理系统，用于对 NBA 相关的基础信息与比赛数据进行统一管理和查询。系统以数据库为核心，围绕球队、球员、赛季、比赛以及技术统计等核心数据展开。

系统支持对 NBA 各支球队的基本信息进行管理，包括球队名称、所属城市等；支持对球员信息进行管理，包括球员姓名、位置、身高体重以及所属球队；支持按赛季组织比赛数据，记录每一场比赛的主客队、比赛时间以及比赛结果。

此外，系统还支持对球员在具体比赛中的技术统计数据进行管理，如得分、篮板、助攻等，为后续的数据分析和统计查询提供数据基础。普通用户可以通过系统查询球队、球员及比赛数据，而管理员可以对相关数据进行维护与更新，从而保证系统数据的准确性和完整性。

2. 功能设计

2.1 概述

- 目标: 提供一个面向普通用户、管理员和数据分析师的 NBA 比赛数据管理与社区系统，支持球队/球员/比赛数据管理、社区互动、用户评分与竞猜、以及数据分析接口。
- 技术栈: 后端 Flask（backend/app.py / 路由见 backend/routes），MySQL（存储过程驱动），前端 Vue 3（frontend/src/views）。

2.2 模块与主要接口

- 认证与用户管理（后端）： `backend/routes/auth.py`
    - POST /api/auth/register: 注册（支持 role + 密钥校验）

    - POST /api/auth/login: 登录，返回 JWT
    - POST /api/auth/logout: 登出（将 jti 加入黑名单）
    - GET/PUT /api/auth/me: 获取/更新用户信息
    - GET /api/auth/me/posts, /me/ratings, /me/points-history: 用户相关历史数据
    - POST /api/auth/verify-password, DELETE /api/auth/delete-account
- 帖子（社区）： `backend/routes/posts.py`
    - GET /api/posts: 列表（支持 game_id 过滤）
    - POST /api/posts: 创建（需登录，支持图片关联）
    - POST /api/posts/{id}/view: 浏览计数
    - POST/DELETE /api/posts/{id}/like: 点赞/取消
    - GET /api/posts/{id}/like-status: 当前用户点赞状态
    - GET/POST /api/posts/{id}/comments: 评论列表/创建
    - DELETE /api/posts/{id}: 删除（作者或管理员）
- 评论点赞： `backend/routes/comments.py`
    - POST/DELETE /api/comments/{id}/like
- 球员管理： `backend/routes/players.py`
    - GET /api/players: 列表（支持 team_id）
    - GET /api/players/{id}: 详情（含生涯/统计）
    - GET /api/players/{id}/stats, /details: 雷达/详细比赛数据
    - POST/PUT/DELETE /api/players (仅管理员)
    - POST /api/players/{id}/photo: 上传球员照片（管理员）
- 比赛管理： `backend/routes/games.py`
    - GET /api/games: 列表（日期/球队过滤）

    - GET /api/games/{id}: 详情（竞猜统计、已结束时球员数据）
    - POST /api/games: 创建（管理员，可包含球员比赛数据）
    - PUT/DELETE /api/games/{id}: 更新/删除（管理员）
    - POST /api/games/{id}/predict: 投票竞猜（需登录）
    - POST /api/games/{id}/claim: 领取竞猜奖励
    - GET/POST /api/games/{id}/ratings: 获取/提交球员评分
    - GET /api/games/{id}/players/{pid}: 单场球员详情与评论
- 球队管理: `backend/routes/teams.py`
    - GET /api/teams、POST/PUT/DELETE /api/teams/{id}（管理员）
    - POST /api/teams/{id}/logo: 上传球队 logo（管理员）
- 图片管理: `backend/routes/images.py`
    - POST /api/images/upload: 上传图片（返回 id + url）
    - POST /api/images/avatar: 上传用户头像并关联
    - GET /api/images/{id}: 返回图片二进制
- 榜单与统计: `backend/routes/rankings.py`
    - GET /api/rankings/teams: 球队战绩榜
    - GET /api/rankings/players: 球员数据榜（stat 参数）
    - GET /api/rankings/players/leaders: 各项领跑者
- SQL 分析: `backend/routes/query.py`（受限，仅 analyst）
    - POST /api/query: 执行自定义 SQL（需权限和安全过滤）

2.3 前端页面与交互流程（主视图）

- 登录/注册: `frontend/src/views/Login.vue`
    - 支持选择角色、密钥输入（管理员/分析师），本地表单校验 -> 调用 /api/auth/register/login。

- 仪表盘: `frontend/src/views/Dashboard.vue`
  - 展示近期比赛、即将比赛、社区热门帖子，快速导航到赛程/榜单/球队。
- 赛程列表与详情: `frontend/src/views/Games.vue` / `frontend/src/views/GameDetails.vue`
  - 列表支持按日期/球队筛选；详情页展示比赛比分、球员数据、竞猜统计、用户可投票和评分、评论。
- 球员库与详情: `frontend/src/views/Players.vue` / `frontend/src/views/PlayerDetails.vue`
  - 列表、筛选、球员详情含雷达图、比赛记录、平均数据、可评分/评论。
- 球员对比: `frontend/src/views/PlayerComparison.vue`
  - 支持两个球员统计对比。
- 球队页面: `frontend/src/views/Teams.vue`
  - 列表+logo，管理员可以上传 logo。
- 榜单视图: `frontend/src/views/Rankings.vue`
  - 支持多种统计类型（得分/篮板/助攻等）。
- 社区帖子: `frontend/src/views/Posts.vue`
  - 帖子列表、查看详情、创建帖子、上传图片、点赞、评论、删除（权限检查）。
- 资料页: `frontend/src/views/Profile.vue`
  - 查看/编辑个人资料、头像上传、积分历史、我的帖子/评分历史。
- SQL 查询工具: `frontend/src/views/Query.vue`（仅 analyst）
  - 前端表单提交 SQL，展示结果（需严控权限与白名单）。

2.4 数据模型与存储过程

- 核心表: User, Team, Player, Game, Player_Game, Post, Comment, Rating, Image, Team_Logo, Player_Image, User_Avatar, Prediction 等（在 backend/database 目录和 SQL 脚本中定义）。
- 存储过程驱动业务: 大量业务通过存储过程（sp_*）完成（如 sp_create_game、sp_get_player_career_stats、sp_upsert_rating、sp_get_posts 等），后端负责事务控制与结果格式化。
- 图片存储: 图片以二进制存入 Image 表（MIME + 数据），通过 /api/images/{id} 提取并作为 URL 使用（前端展示使用该 URL）。

2.5 安全与权限设计

- 认证: JWT（flask_jwt_extended）管理会话，token 可加入黑名单实现登出/撤销。
- 角色控制:
  - 普通用户: 浏览、发帖、评论、评分、竞猜。
  - 管理员: CRUD 球员/球队/比赛、删除任意帖子、上传资源。
  - 数据分析师: 额外的 SQL 查询能力（接口需密钥并进行白名单/语法校验）。
- 注册密钥: 管理员/分析师注册需提供密钥（见 `backend/routes/auth.py` 的静态示例）。
- 输入校验与注入防护: 后端使用参数化查询与存储过程；查询执行接口需额外过滤（目前在 `backend/routes/query.py` 实现）。
- 文件上传防护: 仅允许指定扩展名（png/jpg/jpeg/gif），并在后端检查 MIME。

2.6 事务与错误处理

- 关键写操作（创建比赛并插入球员数据、用户注销、评分/竞猜领奖等）使用事务（conn.autocommit 控制、异常回滚）。
- 所有路由对数据库连接失败、参数缺失、权限不足返回明确的 HTTP 状态码与 JSON 错误信息（400/401/403/404/500）。
- 图片/二进制数据读取使用 send_file 返回，防止直接暴露文件系统路径。

3. 数据流图

3.1 顶层数据流图

在顶层视角下，用户可以向 NBA 比赛数据管理系统 发起各类查询请求，包括：球队信息、球员信息、比赛数据、统计分析数据以及比赛竞猜信息。系统在接收到请求后，将访问对应的数据存储，处理查询并将结果返回给用户。同时，用户还可以在社区模块中发帖、评论和点赞，实现互动交流。

管理员可以向系统提交数据维护请求，包括对球队、球员、比赛、统计数据及竞猜信息的增删改操作。系统在执行相应操作后，将操作结果或反馈信息返回给管理员，确保数据的完整性与准确性。

此外，系统在统计数据模块中提供查看榜单和球员对比功能，为用户提供更深入的数据分析和对比支持。

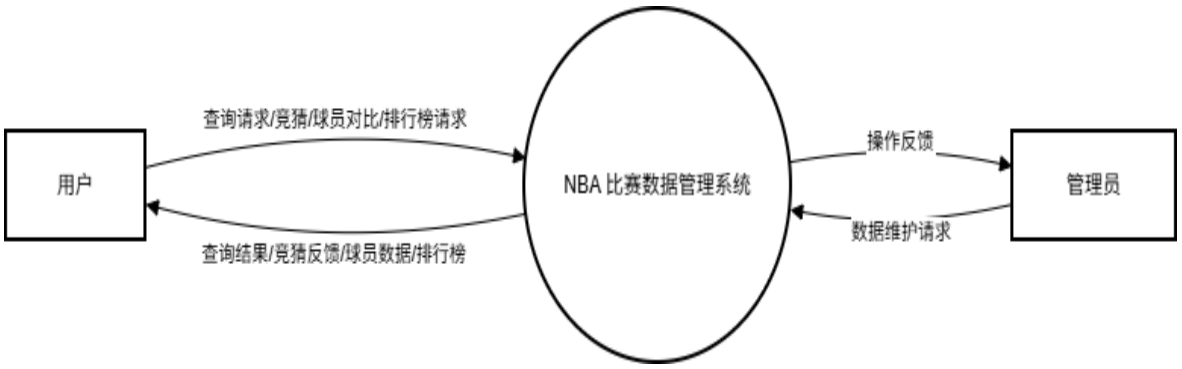


图 1 顶层数据流图

3.2 0 层数据流图

NBA 比赛数据管理系统在 0 层数据流图中被划分为比赛管理子系统和统计数据管理子系统，社区互动子系统。

用户主要向各子系统发起查询请求，系统从相应的数据存储中读取数据并将结果返回给用户；管理员可以对各类数据发起维护请求，子系统在对数据库进行更新后向管理员返回操作反馈。各子系统通过对数据库表的读写操作实现数据的统一管理。

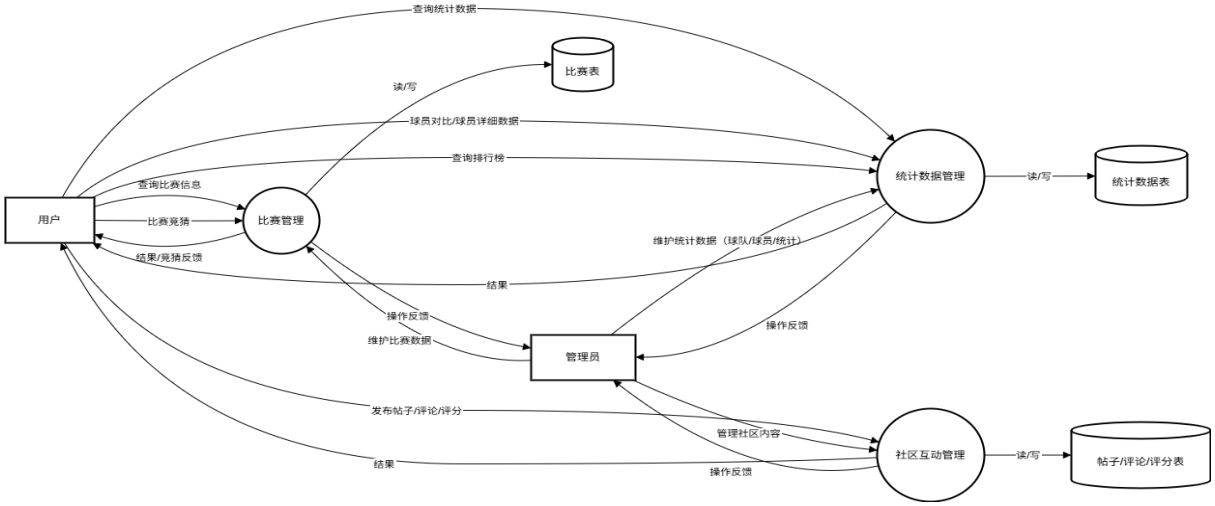


图 20 层数据流图

3.3 1 层数据流图

3.3.1.1 比赛管理子系统

在 1 层数据流图中，对比赛管理子系统进行了进一步细化。该子系统主要包含比赛信息查询、比赛竞猜管理、比赛信息维护和比赛结果处理四个处理过程。

用户可以向比赛信息查询模块提交比赛查询请求，系统从比赛表、球队表及赛季表中读取相关数据并返回查询结果，并通过比赛竞猜管理管理模块参与比赛竞猜活动；管理员可以通过比赛信息维护模块对比赛基本信息进行管理，并通过比赛结果处理模块录入和更新比赛结果，系统在完成数据库更新后向管理员返回操作反馈。

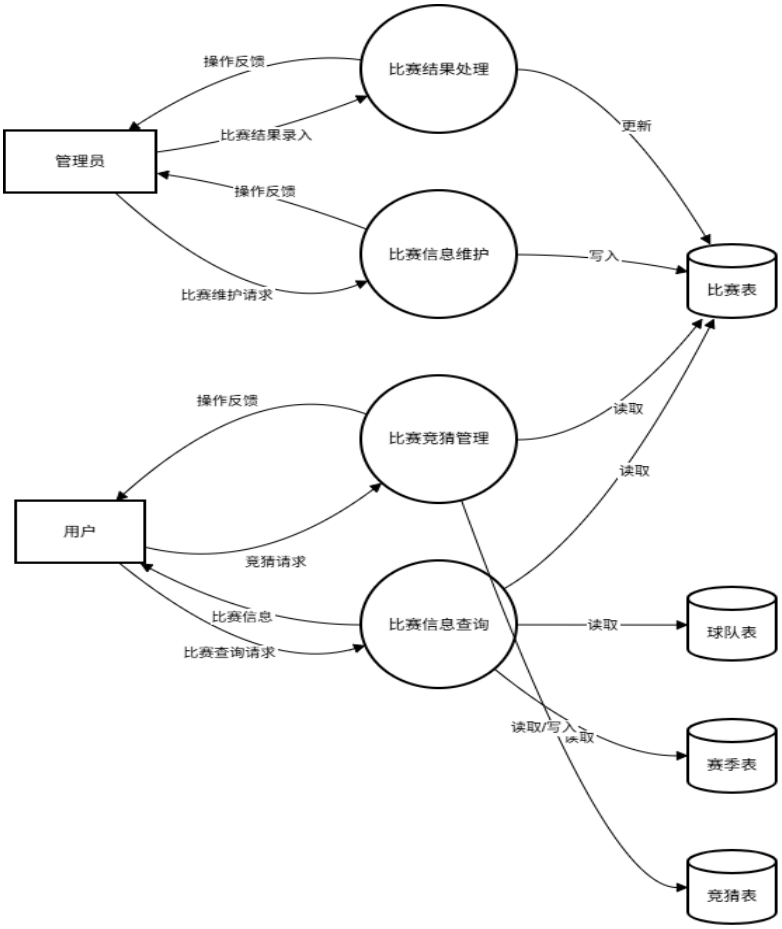


图 3 比赛管理系统数据流图

3.3.1.2 统计数据管理子系统

在统计数据管理子系统的 1 层数据流图中，系统被细化为 统计数据查询、球员统计数据维护、球队统计数据维护、球员对比分析和排行榜统计 五个处理过程。

用户可以向 统计数据查询模块 提交统计查询请求，系统从统计数据表、比赛表、球员表和球队表中读取相关数据，并返回统计结果。同时，用户可以通过 球员对比分析模块 对多个球员的关键数据进行比较，或通过 排行榜统计模块 查看球员和球队在各类指标上的排名情况，为用户提供更直观和深入的统计分析。管理员可以分别通过 球员统计数据维护模块 和 球队统计数据维护模块 对统计数据进行录入和修改，系统在完成数据更新后向管理员返回操作反馈。

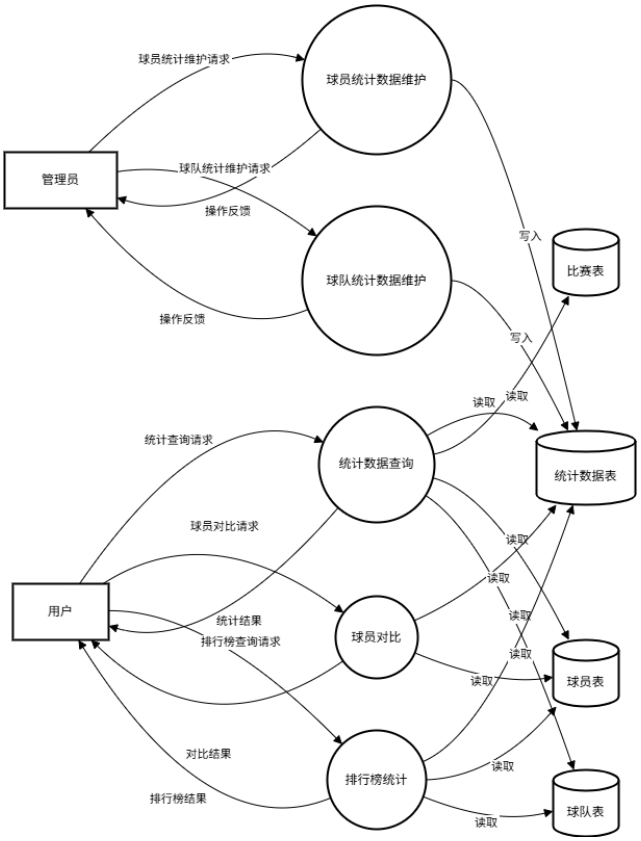
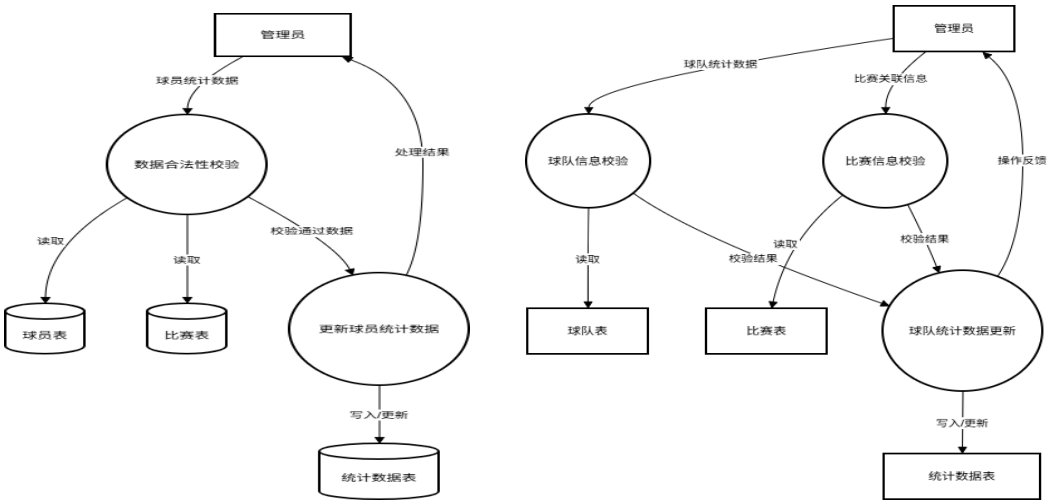


图 4 统计数据管理系统数据流图



(a)球员数据统计　　(b)球队数据统计

图 5 统计数据管理系统主要功能模块数据流图

3.3.1.3 社区互动管理子系统

在社区互动子系统的 1 层数据流图中，系统被细化为浏览帖子、发帖、评论帖子、点赞/收藏、帖子管理、球员卡片和用户管理七个处理过程。

用户可以通过浏览帖子模块查看帖子内容及评论，通过发帖模块发布新帖子，通过评论模块对帖子进行评论，通过点赞/收藏模块对帖子或评论进行点赞和收藏，用户收集的球员卡片，可用于社区互动或展示。系统在完成相应操作后会向用户返回操作反馈。

管理员可以通过帖子管理模块对帖子进行审核、删除或修改，通过用户管理模块对用户账户进行管理。系统在完成操作后向管理员返回操作反馈。

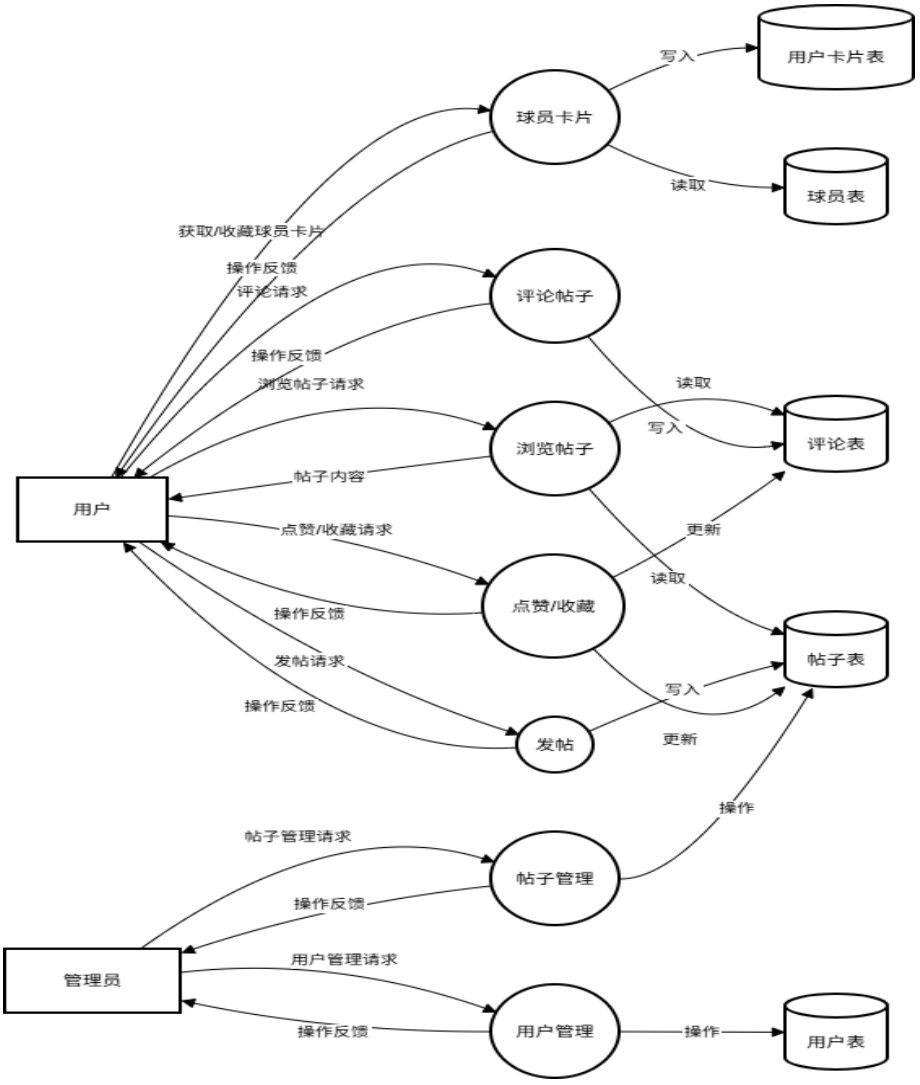


图 6 社区互动管理系统数据流图

4．数据元素表

（1）数据组名：用户信息

字段名	数据类型	值约束	说明
user_id	INT	主键、自增	用户 ID
用户名	VARCHAR(50)	唯一、非空	唯一登录账号
密码	VARCHAR(255)	非空	加密后的密码
角色	ENUM	默认'user'、非空	user/admin/analyst
注册时间	DATETIME	默认当前时间	账号创建时间
最后登录时间	DATETIME	/	最近登录时间
邮箱	VARCHAR(100)	/	邮箱地址
手机号	VARCHAR(20)	/	联系电话
points	INT	默认 0	用户积分

（2）数据组名：球队信息

字段名	数据类型	值约束	说明
team_id	INT	主键、自增	球队 ID
名称	VARCHAR(10)	唯一、非空	球队名称
城市	VARCHAR(30)	默认空串	所在城市

字段名	数据类型	值约束	说明
场馆	VARCHAR(30)	默认空串	主场场馆
分区	ENUM	非空	东部/西部
成立年份	INT	CHECK(>1900)	球队成立年份

（3）数据组名：球员信息

字段名	数据类型	值约束	说明
player_id	INT	主键、自增	球员 ID
姓名	VARCHAR(50)	非空	球员姓名
位置	ENUM	非空	控卫/得分后卫/小前锋/大前锋/中锋
球衣号	INT	/	球衣号码
身高	DECIMAL(3,2)	/	身高（米）
体重	DECIMAL(5,2)	/	体重（公斤）
出生日期	DATE	/	出生日期
国籍	VARCHAR(50)	/	国籍
当前球队 ID	INT	外键 →Team	所属球队 ID

字段名	数据类型	值约束	说明
合同到期	DATE	/	合同到期日期
薪资	DECIMAL(10,2)	/	年薪（万）

（4）数据组名：比赛信息

字段名	数据类型	值约束	说明
game_id	INT	主键、自增	比赛 ID
赛季	VARCHAR(20)	非空	如"2024-2025"
日期	DATETIME	非空	比赛日期时间
主队 ID	INT	非空、外键→Team	主队球队 ID
客队 ID	INT	非空、外键→Team	客队球队 ID
主队得分	INT	/	主队得分
客队得分	INT	/	客队得分
状态	ENUM	默认'未开始'	未开始/已结束
获胜球队 ID	INT	外键→Team	获胜球队 ID
场馆	VARCHAR(100)	/	比赛场馆

（5）数据组名：帖子

字段名	数据类型	值约束	说明
post_id	INT	主键、自增	帖子 ID
user_id	INT	非空、外键→User	发帖用户 ID
标题	VARCHAR(50)	非空	帖子标题
内容	TEXT	非空	帖子内容
game_id	INT	外键→Game	关联比赛 ID
创建时间	DATETIME	默认当前时间	发帖时间
浏览量	INT	默认 0	浏览次数
点赞数	INT	默认 0	点赞次数

（6）数据组名：评论

字段名	数据类型	值约束	说明
comment_id	INT	主键、自增	评论 ID
user_id	INT	非空、外键→User	评论用户 ID
player_id	INT	外键→Player	关联球员 ID
game_id	INT	外键→Game	关联比赛 ID
post_id	INT	外键→Post	关联帖子 ID

字段名	数据类型	值约束	说明
内容	TEXT	非空	评论内容
创建时间	DATETIME	默认当前时间	评论时间
点赞数	INT	默认 0	点赞次数

（7）数据组名：评分

字段名	数据类型	值约束	说明
user_id	INT	主键(联合)、外键→User	评分用户 ID
player_id	INT	主键(联合)、外键→Player	被评球员 ID
game_id	INT	主键(联合)、外键→Game	关联比赛 ID
分数	DECIMAL(2,0)	默认 0、CHECK(0-10)、非空	评分(0-10)
创建时间	DATETIME	默认当前时间	评分时间

（8）数据组名：帖子-点赞表

字段名	数据类型	值约束	说明
like_id	INT	主键、自增	点赞 ID
user_id	INT	非空、外键→User、级联删除	点赞用户 ID
post_id	INT	非空、外键→Post、级联删除	被赞帖子 ID

字段名	数据类型	值约束	说明
创建时间	DATETIME	默认当前时间	点赞时间

（9）数据组名：评论-点赞表

字段名	数据类型	值约束	说明
like_id	INT	主键、自增	点赞 ID
user_id	INT	非空、外键→User、级联删除	点赞用户 ID
comment_id	INT	非空、外键→Comment、级联删除	被赞评论 ID
创建时间	DATETIME	默认当前时间	点赞时间

（10）数据组名：图片

字段名	数据类型	值约束	说明
image_id	INT	主键、自增	图片 ID
名称	VARCHAR(50)	非空	图片名称
数据	LONGBLOB	非空	图片二进制数据
MIME 类型	VARCHAR(50)	非空	如 image/jpeg
上传时间	DATETIME	默认当前时间	上传时间

（11）数据组名：竞猜

字段名	数据类型	值约束	说明
prediction_id	INT	主键、自增	竞猜 ID
user_id	INT	非空、外键→User	竞猜用户 ID
game_id	INT	非空、外键→Game	竞猜比赛 ID
predicted_team_id	INT	非空、外键→Team	预测获胜球队
is_claimed	BOOLEAN	默认 FALSE	是否已领奖
create_time	DATETIME	默认当前时间	竞猜创建时间
update_time	DATETIME	默认当前时间、自动更新	更新时间

（12）数据组名：球员卡

字段名	数据类型	值约束	说明
id	INT	主键、自增	记录 ID
user_id	INT	非空、外键→User	持有用户 ID
player_id	INT	非空、外键→Player	球星卡球员
get_time	DATETIME	默认当前时间	获取时间

（13）数据组名：球队-比赛关系表

字段名	数据类型	值约束	说明
team_id	INT	主键(联合)、外键→Team	球队 ID
game_id	INT	主键(联合)、外键→Game	比赛 ID
主客类型	ENUM	非空	主场/客场

（14）数据组名：球员-比赛数据表

字段名	数据类型	值约束	说明
player_id	INT	主键(联合)、外键→Player	球员 ID
game_id	INT	主键(联合)、外键→Game	比赛 ID
上场时间	DECIMAL(4,1)	默认 0、CHECK(>=0)、非空	上场分钟数
得分	INT	默认 0、CHECK(>=0)、非空	得分
篮板	INT	默认 0、CHECK(>=0)、非空	篮板数
助攻	INT	默认 0、CHECK(>=0)、非空	助攻数
抢断	INT	默认 0、CHECK(>=0)、非空	抢断数
盖帽	INT	默认 0、CHECK(>=0)、非空	盖帽数
失误	INT	默认 0、CHECK(>=0)、非空	失误数
犯规	INT	默认 0、CHECK(>=0)、非空	犯规数

字段名	数据类型	值约束	说明
正负值	INT	/	正负值统计

（15）用户-头像关系表

字段名	数据类型	值约束	说明
user_id	INT	主键、外键→User	用户 ID
image_id	INT	非空、外键→Image	头像图片 ID

（16）帖子-图片关系表

字段名	数据类型	值约束	说明
post_id	INT	非空、外键→Post	帖子 ID
image_id	INT	主键、外键→Image	图片 ID

（17）球员-图片关系表

字段名	数据类型	值约束	说明
player_id	INT	主键、外键→Player	球员 ID
image_id	INT	非空、唯一、外键→Image、级联删除	球员照片 ID

（18）球队-队标关系表

字段名	数据类型	值约束	说明

字段名	数据类型	值约束	说明
team_id	INT	主键、外键→Team	球队 ID
image_id	INT	非空、唯一、外键→Image、级联删除	队标图片 ID

二．数据库概念模式设计

1．系统初步E-R 图

该初步 E-R 图仍然包含球队、球员、比赛和统计数据等核心实体，但仅保留实体的主键属性，用以表示实体的唯一性。通过这种方式，可以更加直观地展示系统中实体之间的联系结构，而不受具体属性数量的影响。

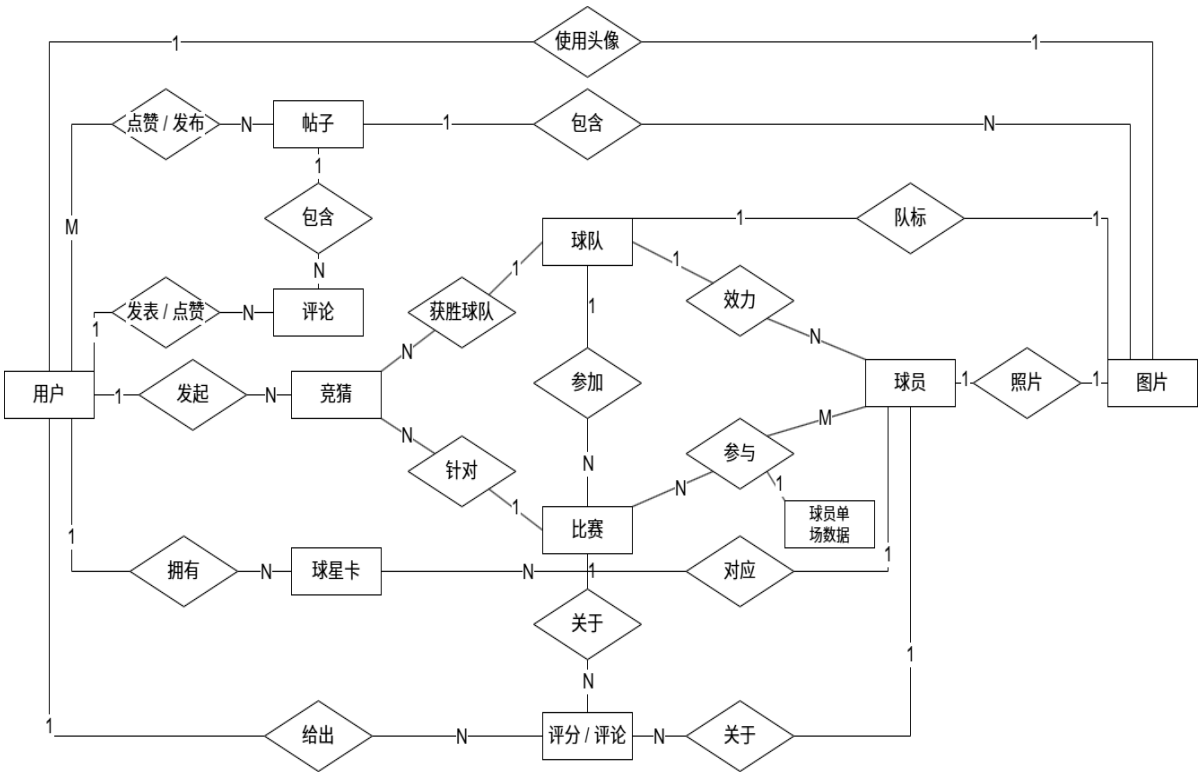


图 7 系统初步 E-R 图

2．系统基本 E-R 图

在系统基本 E-R 图中，对系统初步 E-R 图进行了进一步优化，重点解决多

对多关系和数据冗余问题，使概念结构更接近实际的数据库逻辑结构。

优化依据：球员，比赛，球员单场数据是 1:1:1 关系，因此将实体球员单场数据下降为球员参与比赛这一关系的属性

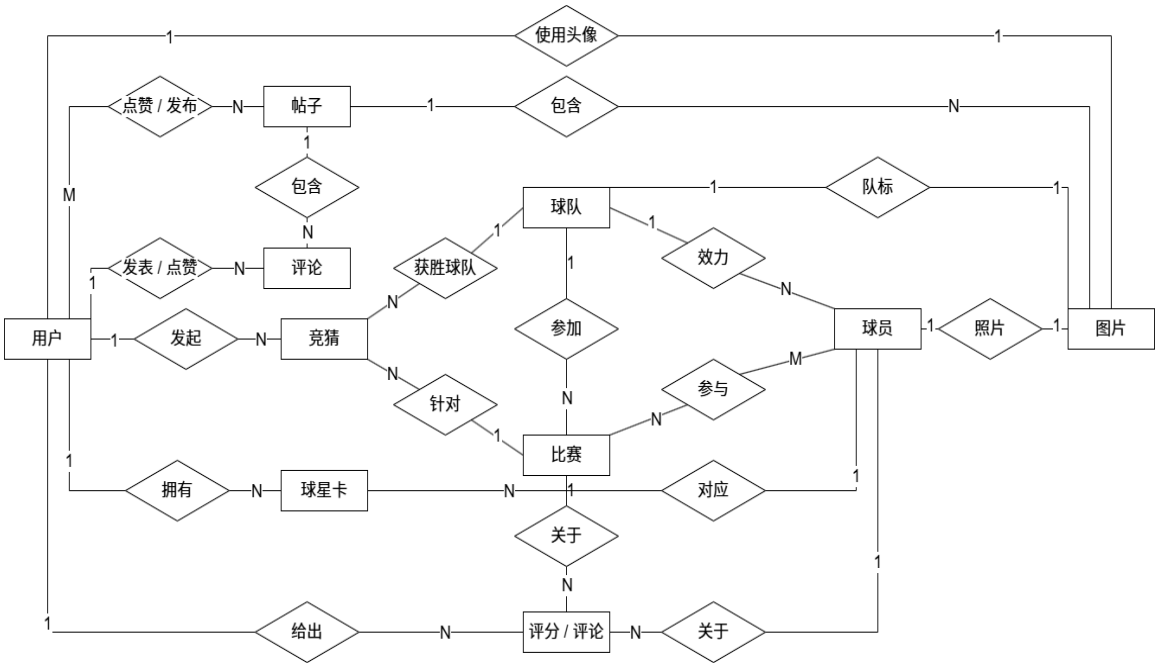


图 8 系统基本 E-R 图

三、数据库逻辑模式设计与优化

1．数据库关系模式定义

1.1 实体

用户(id, 用户名, 密码, 角色, 注册时间, 个人简介)

球队(id, 球队名, 所属城市, 分区, 主场, 成立年份, 创建时间)

球员(id, 姓名, 身高, 体重, 位置, 出生日期, 国籍, 球队 id)

比赛(id, 主队 id, 客队 id, 比赛日期, 比赛地点, 比赛状态)

帖子(id, 标题, 内容, 发布时间, 用户 id)

评论(id, 内容, 评论时间, 用户 id, 帖子 id)

评分(id, 分值, 评分时间, 用户 id, 比赛 id)

图片(id, 文件名, 文件类型, 创建时间, 创建者 id, 二进制流)

1.2 联系

用户-帖子(用户 id, 帖子 id)

帖子-评论(帖子 id, 评论 id)

用户-评论(用户 id, 评论 id)

用户-评分(用户 id, 评分 id)

比赛-评分(比赛 id, 评分 id)

球队-球员(球队 id, 球员 id)

比赛-球队(比赛 id, 球队 id)

用户-图片(用户 id, 图片 id)

用户-头像(用户 id, 图片 id)

球队-图片(球队 id, 图片 id)

2．关系模式范式等级的判定与规范化

首先，系统中所有关系模式的属性均为不可再分的原子值，不存在在一个属性中嵌套表或重复组的情况，因此所有关系模式均满足第一范式（1NF）。

对于实体关系模式，包括用户、球队、球员、比赛、帖子、评论、评分和图片等实体，其主码均为 id，其中用户实体中用户名也可作为候选码。除由码引入的函数依赖外，各实体关系中不存在其他非平凡函数依赖关系，因此所有非主属性均完全函数依赖于主码，且不存在对主码的部分函数依赖和传递函数依赖。因此，上述所有实体关系模式均至少满足第三范式（3NF）。

对于二元联系关系模式，如用户-帖子、帖子-评论、用户-评论、用户-评分、比赛-评分、球队-球员、比赛-球队以及用户-图片等关系，由于关系中仅包含两个属性或由两个外键组成，不可能存在部分函数依赖和传递函数依赖，因此这些联系关系模式均满足第三范式（3NF）。

对于用户对比赛进行评分这一联系关系，其码为 *(用户 id, 比赛 id)*，唯一的函数依赖为

(用户 id, 比赛 id) → *评分值, 评分时间*,

非主属性完全函数依赖于该码，不存在部分函数依赖和传递函数依赖，因此该关系模式满足第三范式（3NF）。

对于比赛-球队这一联系关系，其码为 *(比赛 id, 球队 id)*，不存在非码属性，

也不存在非平凡函数依赖，因此该关系模式自然满足第三范式（3NF）。

综上所述，本系统中所有实体关系和联系关系均至少满足第三范式（3NF），有效避免了数据冗余和更新异常，满足数据库逻辑结构设计的规范化要求。

3．数据库关系模式优化

在本 NBA 比赛数据管理系统中，球队包含多个球员，比赛包含多条球员比赛数据和球队比赛数据，用户可以发布多个帖子，帖子包含多条评论，用户可以对多场比赛进行评分，均属于 1:n 关系。

对于上述 1:n 关系，可以通过在 n 端实体中加入 1 端实体的 id 作为外键 来实现关联，从而避免单独建立 1:n 联系表，简化数据库结构。

具体而言，在球员实体中加入球队 id，用以表示球员所属球队；在球员比赛数据和球队比赛数据实体中加入比赛 id，用以表示数据所属的比赛；在帖子实体中加入用户 id，用以表示帖子的发布者；在评论实体中加入帖子 id 和用户 id，用以表示评论的所属帖子和评论者；在评分实体中加入用户 id 和比赛 id，用以表示评分的发起用户和对应比赛。

用户与图片、球队与图片之间均为 1:1 关系，其中图片主要用于用户头像或球队标志。因此，可以在用户表中加入头像图片 id，在球队表中加入图片 id，从而避免单独建立用户-图片和球队-图片联系表，进一步减少冗余关系。

对于比赛-球队这一关系，由于一场比赛必然包含两支球队（主队和客队），且在比赛实体中已通过主队 id 和客队 id 明确表示参与比赛的球队，因此无需再单独建立比赛-球队联系表，该关系可由比赛实体自身属性直接体现。

最终，本系统中仅保留 用户-球队（关注） 这一 n:m 关系需要单独建立关系表，用于记录用户对球队的关注情况，其余关系均已通过在实体中引入外键或合并关系的方式进行了优化。

四、数据库物理设计

1. 存取方法选择

本系统使用 华为云 GaussDB(for MySQL) 作为数据库管理系统，所有表默认采用 InnoDB 存储引擎。

优化设计：

1) 对于 1:n 关系，如球队-球员、帖子-评论、比赛-球员比赛数据等，将 1 端实体的 id 放在 n 端实体中，避免额外建立 1:n 联系表；
2) 对于 1:1 关系，如用户头像、球队标志，将图片 id 放入用户表或球队表，避免单独建立联系表；
3) 对于 n:m 关系，如用户-球队关注，单独建立关系表，并通过联合主键保证唯一性。

2. 索引设计

为提高查询效率，系统对主键、外键及常用查询字段建立索引，具体如下：

1. 用户表
    - PRIMARY KEY(id)
    - UNIQUE(username)
2. 球队表
    - PRIMARY KEY(id)
    - INDEX(city)
3. 球员表
    - PRIMARY KEY(id)
    - INDEX(team_id)
4. 比赛表
    - PRIMARY KEY(id)
    - INDEX(home_team_id)
    - INDEX(away_team_id)
    - INDEX(match_date)
5. 帖子表
    - PRIMARY KEY(id)
    - INDEX(user_id)
6. 评论表
    - PRIMARY KEY(id)

- INDEX(post_id)
- INDEX(user_id)

7. 评分表
   - PRIMARY KEY(id)
   - UNIQUE(user_id, match_id)
   - INDEX(match_id)
8. 图片表
   - PRIMARY KEY(id)
   - INDEX(user_id)
   - INDEX(team_id)
9. 用户-球队关注关系表
   - PRIMARY KEY(user_id, team_id)
   - INDEX(user_id)
   - INDEX(team_id)

3. 设计说明

- 使用 InnoDB + 主键索引 + 外键约束，保证数据一致性、事务安全和高效检索；
- 辅助索引 提高系统查询效率，支持用户查看帖子、评论、评分和球员数据等操作；
- 联合唯一索引 保证 n:m 表中数据唯一性，避免重复关注或重复评分；
- BLOB 分块存储优化头像球队标志等大对象存取，降低内存压力；
- 数据库设计充分满足系统增删改查、统计查询及多用户并发访问的功能需求。