

数据库设计文档 同济绿茵

———基于五大联赛与中超的足球交流平台

数据库设计文档 同济绿茵

———基于五大联赛与中超的足球交流平台

1.1) 概念设计

数据库总体E-R图

实体及其属性的介绍

实体间联系的介绍

模块E-R图

用户相关关系模块ER图

论坛帖子关系ER图

新闻ER图

球队球员赛事关系ER图

1.2) 逻辑设计

表设计

通知表

球员参赛表

球员表

帖子图片表

帖子表

发布帖子表

举报表

帖子标签表

球队表

球队签约球员表

主题表

管理员发布公告

管理员

用户登录

用户收藏帖子

用户评论帖子

用户关注用户

赛事

赛事队伍

高光时刻

用户赞同帖子

新闻

新闻拥有图片

视频

用户

用户收藏球队

1.3) 部署策略

部署选择

部署流程

1.4) 库设计

表设计

1.5) 配置管理

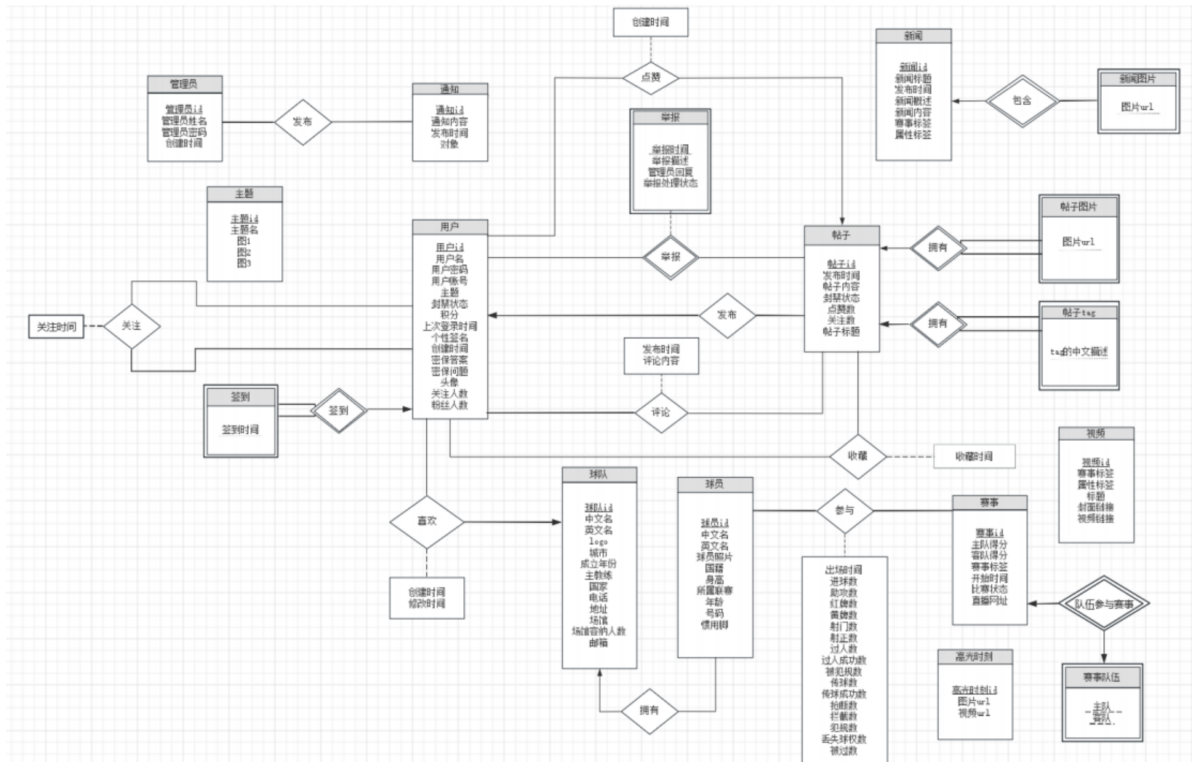
数据持久化

参数文件

日志信息

1.1) 概念设计

数据库总体E-R图



实体及其属性的介绍

- **用户账号**

用来记录用户的账号以及个人信息，用户在注册账号时新增一条记录，同时记录用户在使用期间所产生的动态信息。属性包括：用户id，用户昵称，密码，账号，是否封禁，用户积分，主题，个性签名，创建时间，密保问题，密保答案，头像，粉丝数，关注数，上次登录时间。主码为：用户id。

- **管理员账号**

管理员与用户类似，其拥有用户拥有的所有属性与权限，但又独立于用户。管理员拥有整个系统的最高权限，以达到对系统进行管理的目的。属性包括：管理员id，管理员姓名，密码，创建时间。主码为：管理员id。

- **通知**

通知是管理员发布给用户的站务通知，根据对象的值决定是私信还是广播。属性包括：通知id，通知内容，发布时间，对象。主码为：通知id。

- **帖子**

帖子是由用户发布在论坛的内容，用户在发帖时新增一条帖子记录。属性包括：帖子id，发布时间，帖子内容，封禁状态，点赞，关注数，帖子标题。主码为：帖子id。

- **帖子tag (标签)**

用来记录帖子的tag标签，用户发帖时可选择或自定义tag。属性包括：帖子id，tag内容。主码为：帖子id，tag内容

- **帖子图片**

用来记录帖子的图片，用户在发帖时上传配套图片并存储在服务器中。属性包括：帖子id，图片url。主码为：帖子id，图片url。

- **举报**

举报用来存储用户举报帖子的相关信息。当用户举报一条帖子时，新增一条举报记录。属性包括：举报时间，举报描述，管理员回复，举报处理状态，举报人id，帖子id，举报时间。

- **签到**

用来记录用户的签到记录，用户每进行一次签到便新增一条记录。属性包括：用户id，签到时间。主码为：用户id，签到时间。

- **主题**

用来存储网站可用的主题。属性包括：主题id，主题名，图1，图2，图3。主码为：主题id。

- **** 新闻**

新闻用来存储足球相关新闻。属性包括：新闻id，新闻标题，发布时间，新闻概述，新闻内容，赛事标签，属性标签。主码是：新闻id。

- **新闻图片**

用来记录新闻对应的图片。有的新闻存在图片，记录新闻对应的图片url。属性包括：新闻id，图片url，主码是：新闻id，图片url。

- **高光时刻**

高光时刻类似与新闻，用来记录足球场上的精彩瞬间。属性包括：高光时刻id，图片url，视频url。主码为：高光时刻id。

- **视频**

用于存储足球相关视频。属性为：视频id，赛事标签，属性标签，标题，封面连接，视频链接。主码为：视频id。

- **球队**

用来记录球队的相关信息。属性包括：球队id，中文名，英文名，logo，城市，成立年份，主教练，国家，电话，地址，场馆，场馆容纳人数，邮箱。主码是：球队id。

- **球员**

用来记录球员的相关信息。属性包括：球员id，中文名，英文名，球员照片，国籍，身高，所属联赛，年龄，号码，惯用脚。主码是：球员id。

- **赛事**

用来记录单场的足球赛事信息。属性包括：赛事id，主队得分，客队得分，赛事标签，开始时间，比赛状态，直播网址。主码为：赛事id。

- **** 赛事所属球队**

用来记录单场比赛包含的主客队id。本身没有可以唯一标识的主码，必须依赖于赛事才可以确认主码。

实体间联系的介绍

从E-R图中可以看出，各个实体集之间存在着极为密切的联系，接下来将——说明：

- **用户与用户的关注关系**

用户可关注另一个用户，此时需要记录或更新用户与用户之间的关系，一个用户可以关注多个用户，也可以被多个用户关注，因此该联系集是多对多的。

- **用户与签到的关联关系**

用户可以在不同日期进行签到，一个用户对应多条签到记录，一个签到记录只对应一个用户，因此该联系集是一对多的。

- **帖子与Tag的关联关系**

一个帖子在发布时可由发布者添加多个Tag于帖子上，此时需要记录帖子与Tag之间的关系，一条帖子可以拥有多个Tag，Tag只属于创建时属于的帖子，因此该联系集是一对多的。

- **帖子与帖子图片的关联关系**

一个帖子在发布时可由发布者添加多个帖子图片于帖子上，此时需要记录帖子与帖子图片之间的关系，一条帖子可以拥有多个帖子图片，帖子图片只属于创建时属于的帖子，因此该联系集是一对多的。

- **用户与帖子的发布关系**

用户可以向网站发布帖子，此时需要记录帖子与发布者之间的关系。一个用户可发布多个帖子，每个帖子只能被一个用户发布，因此该联系集是一对多的，且帖子全部参与该联系集。

- **用户与帖子的点赞关系**

用户在对帖子进行点赞时，需要更新帖子的点赞量，同时记录用户和帖子的点赞关系，一个用户可对多个视频进行帖子，同时一个帖子可被多个用户点赞，因此该联系集是多对多的。

- **用户与帖子的收藏关系****

用户在对帖子进行收藏时，需要更新帖子的收藏量，同时记录用户和帖子的收藏关系，一个用户可对多个帖子进行收藏，同时一个帖子可被多个用户收藏，因此该联系集是多对多的。

- **用户、帖子的举报关系**

用户在对帖子进行举报时，需要记录用户和帖子的举报关系，一个用户可对多个帖子进行举报，同时一个帖子可被多个用户举报，因此该联系集是多对多的。

- **用户与帖子评论的发表关系**

用户发表一条对帖子内容的评论时，需要记录用户和评论的发表关系。一个用户可对多个帖子进行评论，然而一条评论只能被一个用户发表，因此该联系集是一对多的，而且评论全部参与到该关系中。

- **帖子与帖子评论的从属关系**

用户在一个帖子的评论区发表一条对帖子内容的评论时，需要记录该评论与帖子的从属关系。一个帖子可以有很多条评论，然而一条评论只能从属于一个帖子，因此该联系集是一对多的，而且评论全部参与到该关系中。

- **新闻与新闻图片的关联关系**

- 一个新闻可添加多个图片于新闻上，此时需要记录新闻与帖子图片之间的关系，一条新闻可以拥有多个新闻图片，新闻图片只属于创建时属于的新闻，因此该联系集是一对多的。

- **用户与球队的关联关系**

用户最多只能将一个球队设置为主队，而球队可以被多个用户设置为主队，因此该联系集是多对一的。

- **球员与球队的关联关系**

球员与球队是拥有关系，一个球员只能属于一支球队，而一支球队可以拥有多名球员，因此该联系集是多对一的。

- **球员与赛事的关联关系**

球员与赛事的关联关系记录球员参与的每一场比赛的数据。球员可以参与多场赛事，一场赛事有多名球员参与，因此该联系集是多对多的。

- 球队参与赛事

赛事与球队需要此联系集进行标识。在转换为关系模式时候，由于每个赛事只能对应一个球队参与的信息，只需要转换为一个包含了赛事id、主客队id的表即可。

- 管理员与通知的关联关系

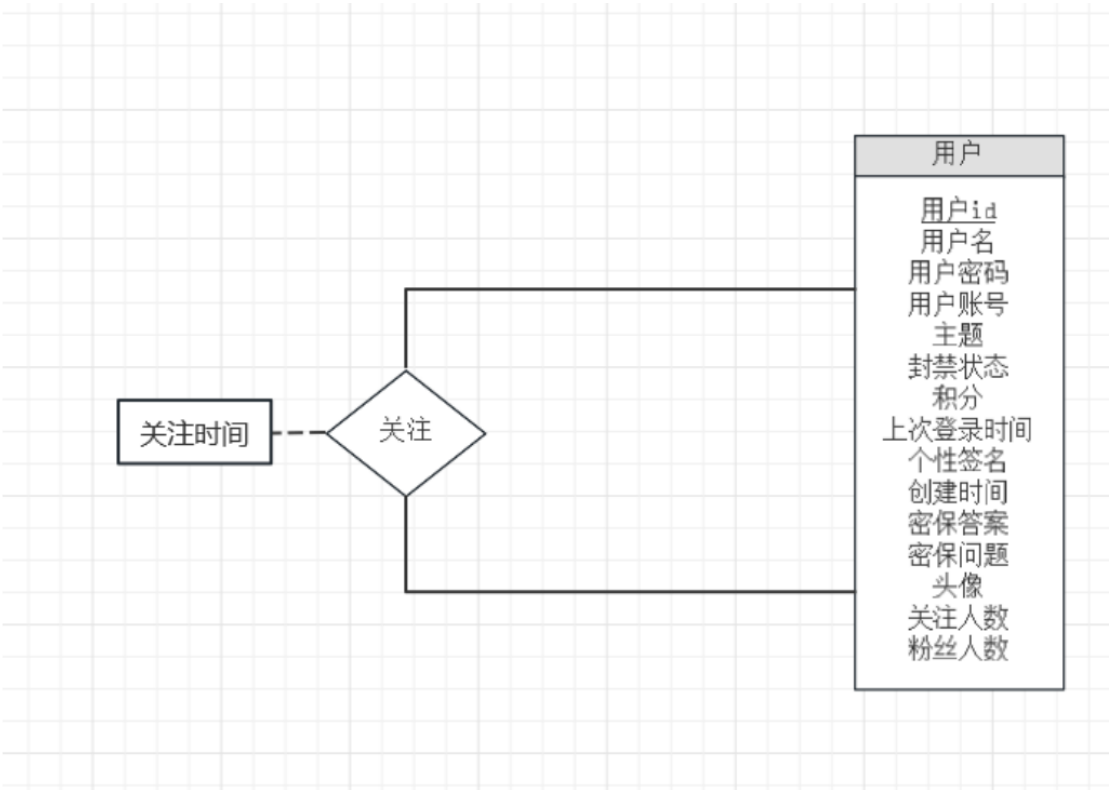
管理员在发布站务通知时，需要记录管理员与通知的关系。管理员可以发布多条通知，而一条通知只能由一名管理员发布，因此该联系集是一对多的。

模块E-R图

用户相关关系模块ER图

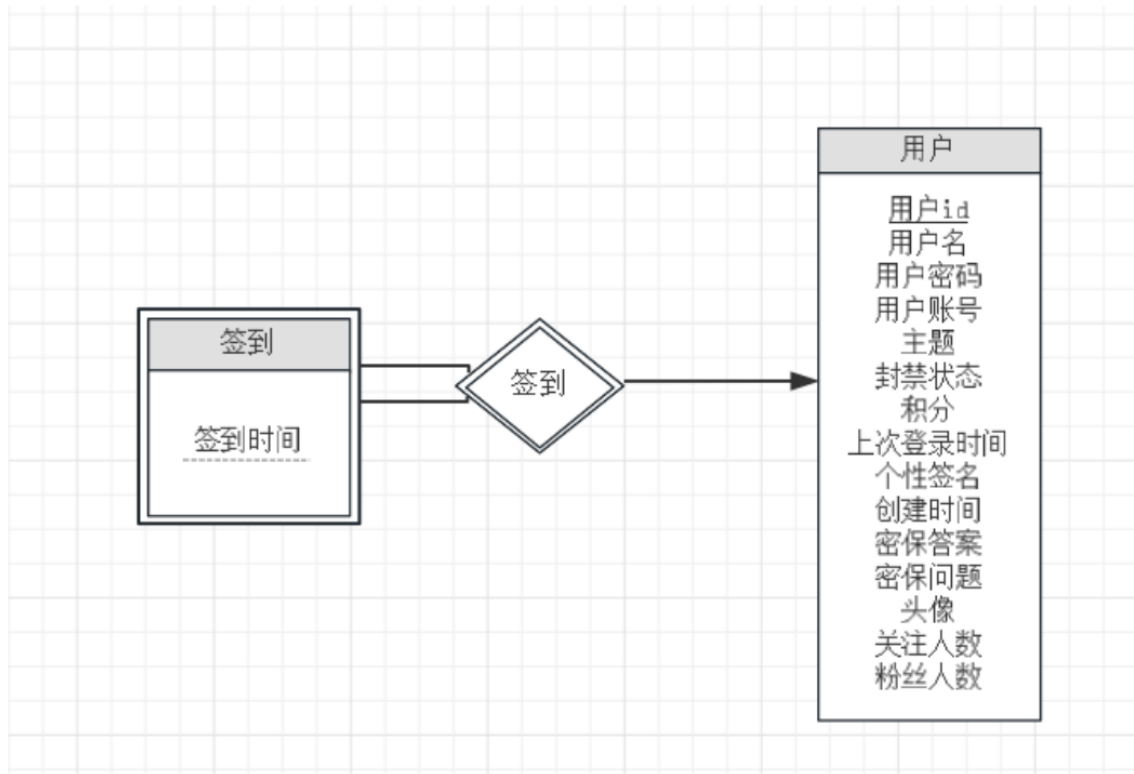
- 用户关注用户

附加属性为关注时间

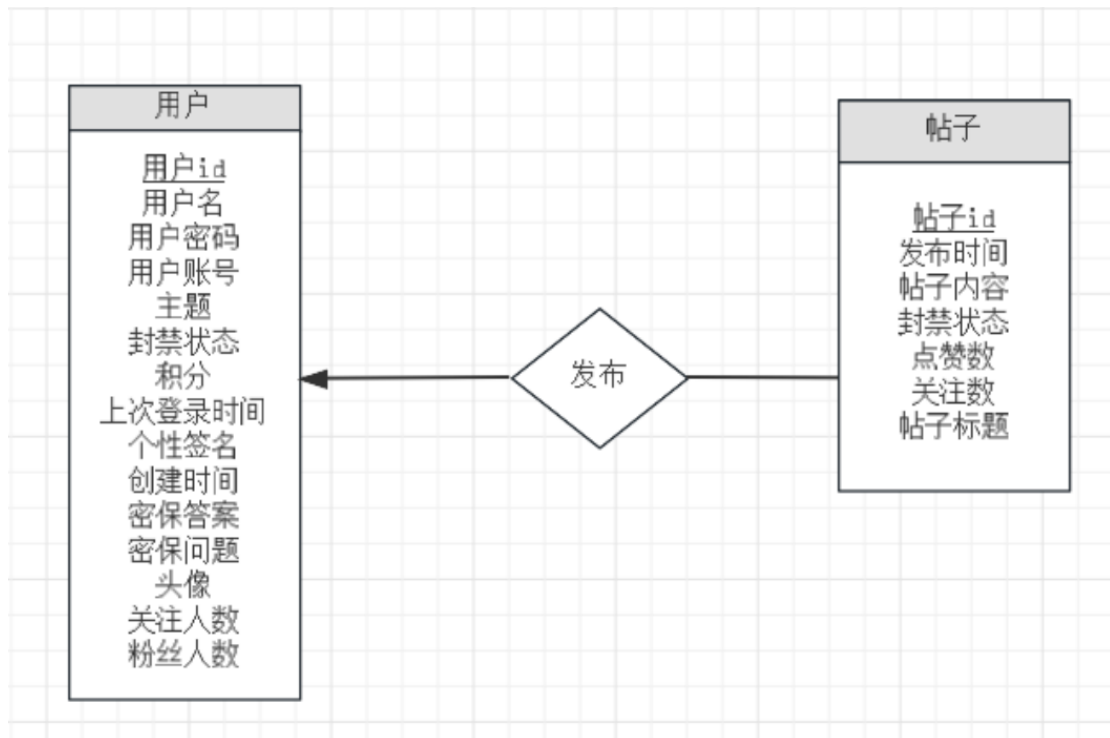


- 用户签到

签到集合为弱实体集，必须依靠用户的id进行唯一标识

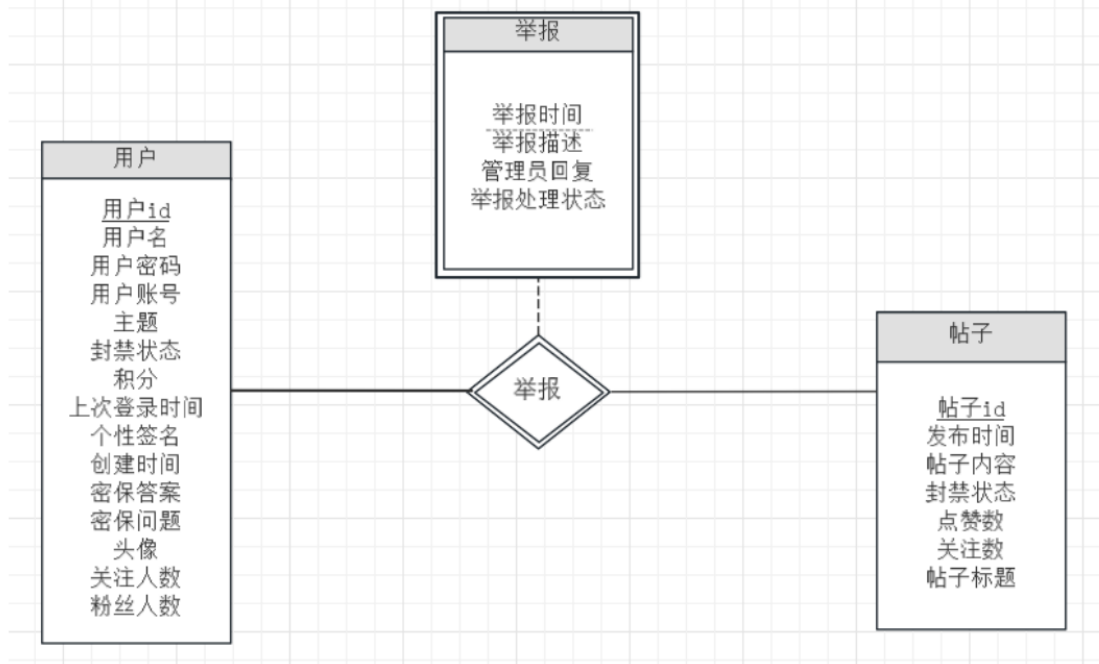


- 用户发布帖子



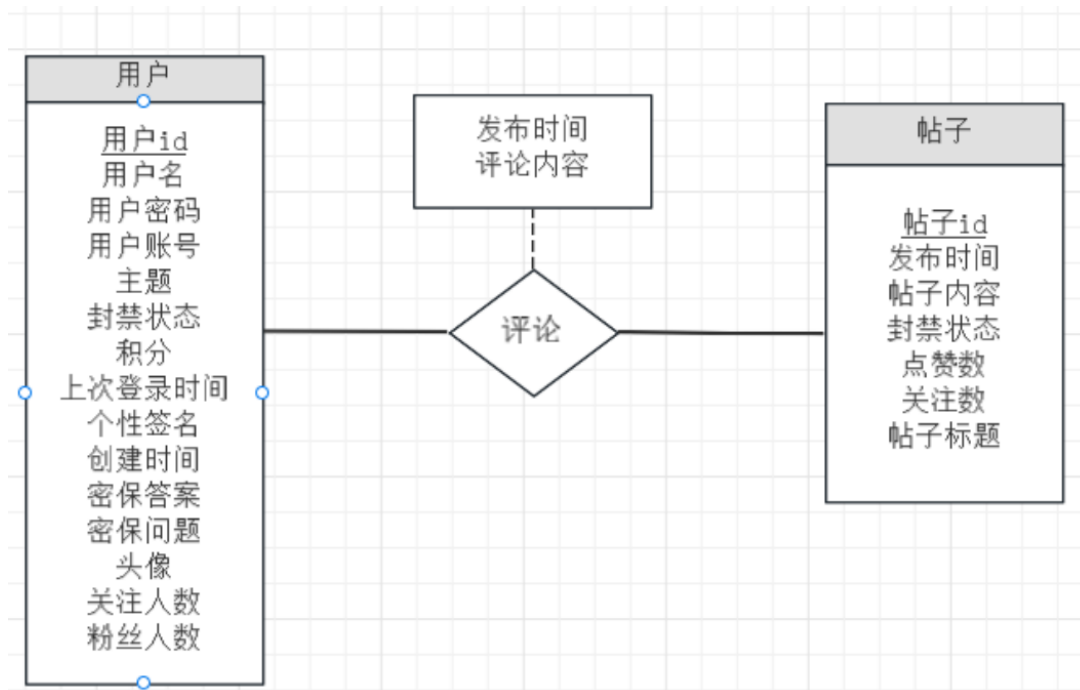
- 用户举报帖子

举报同样也是弱实体集，需要自身的举报时间属性以及举报者、帖子的主码进行标识用户评论帖子



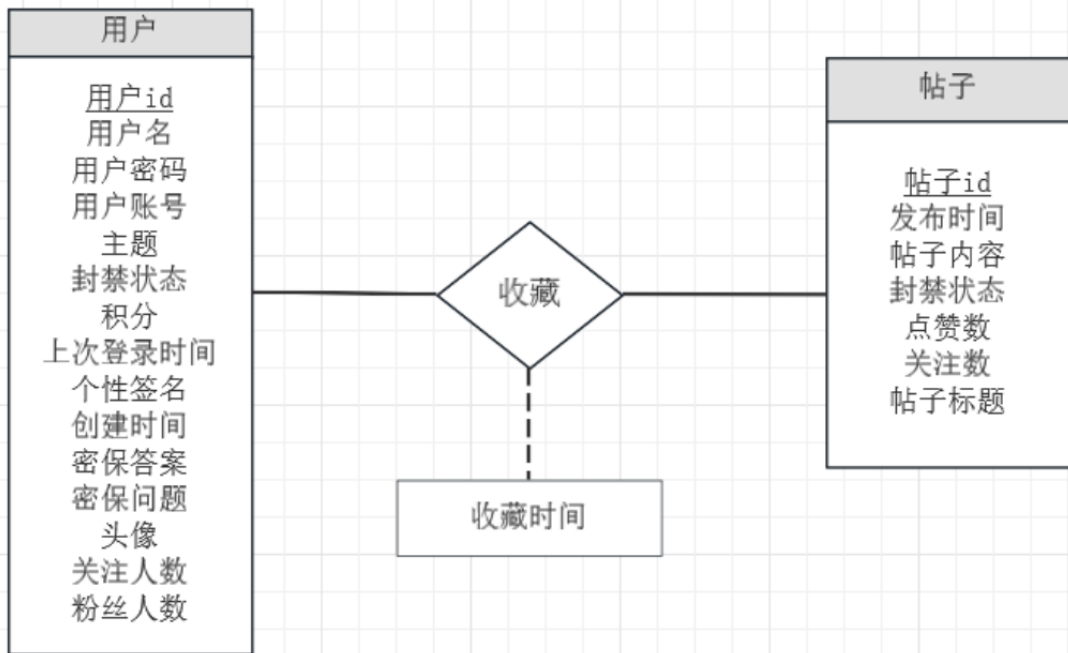
- 用户评论帖子

附加属性为发布时间以及评论内容

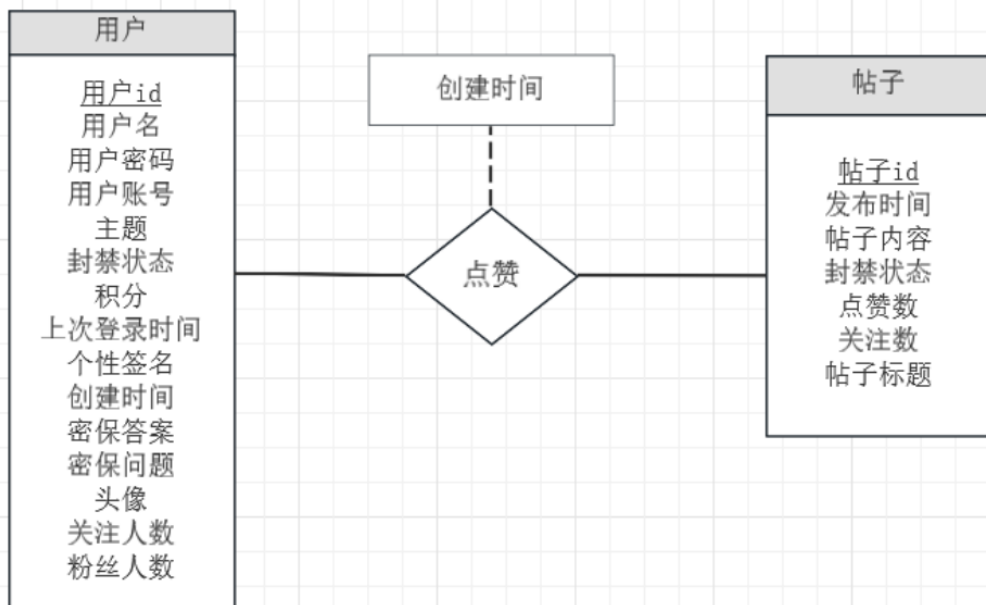


- 用户收藏帖子

附加属性为收藏时间用户点赞帖子

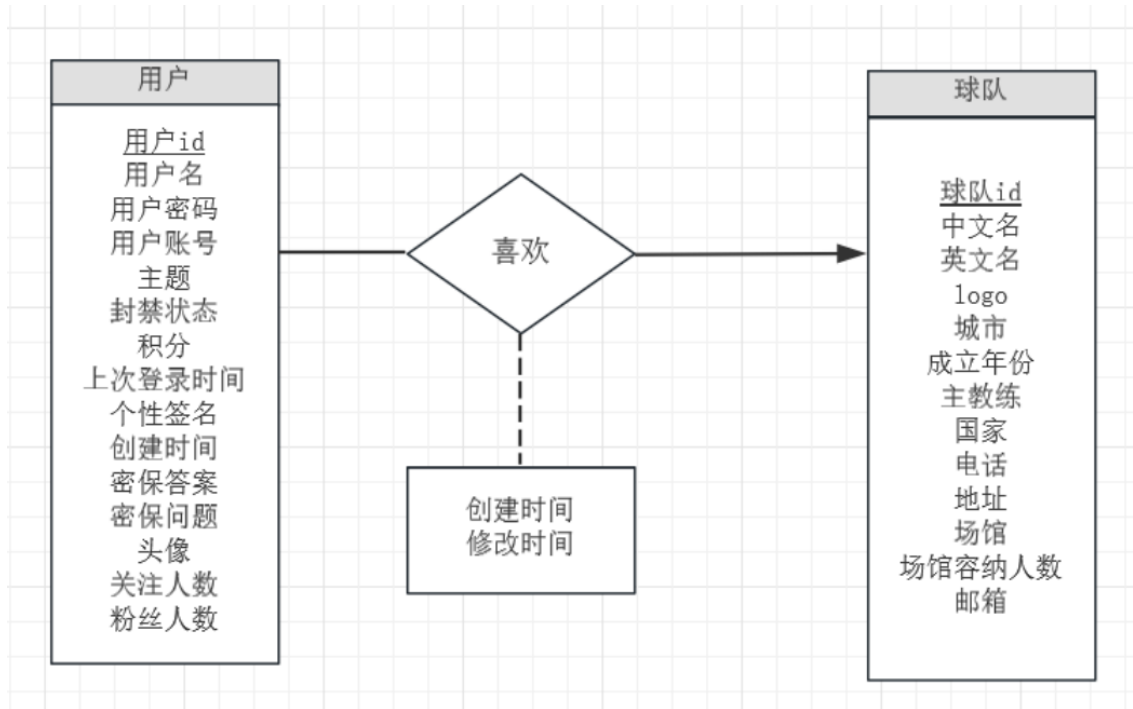


- 用户点赞帖子

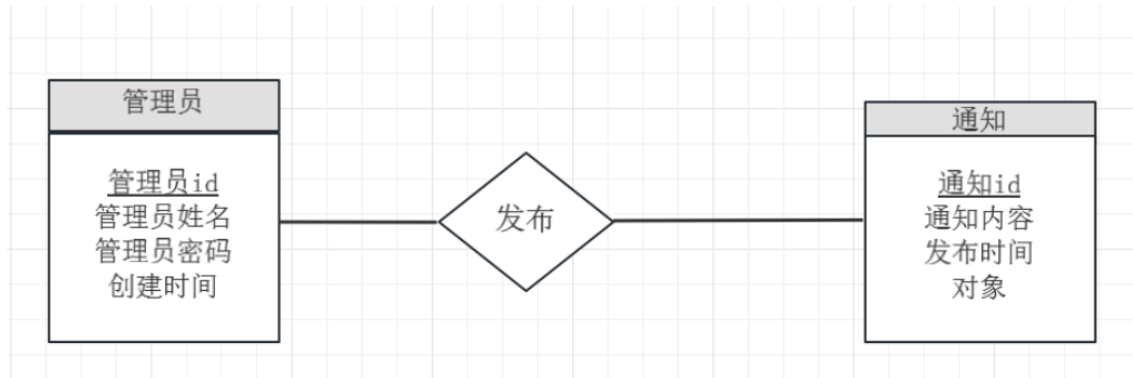


- 用户喜欢球队

附加属性为创建时间与改进时间

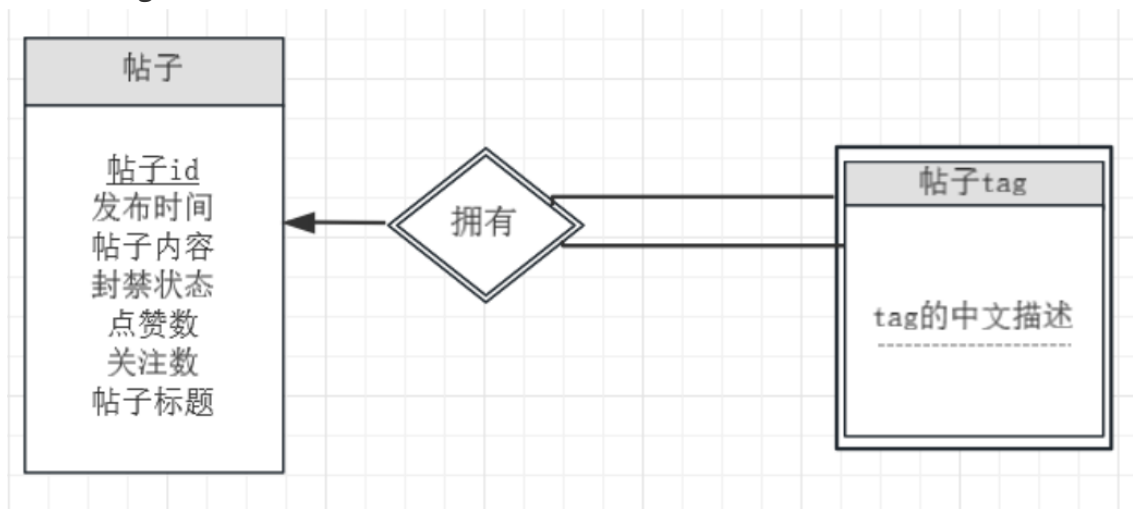


- 管理员发布通知

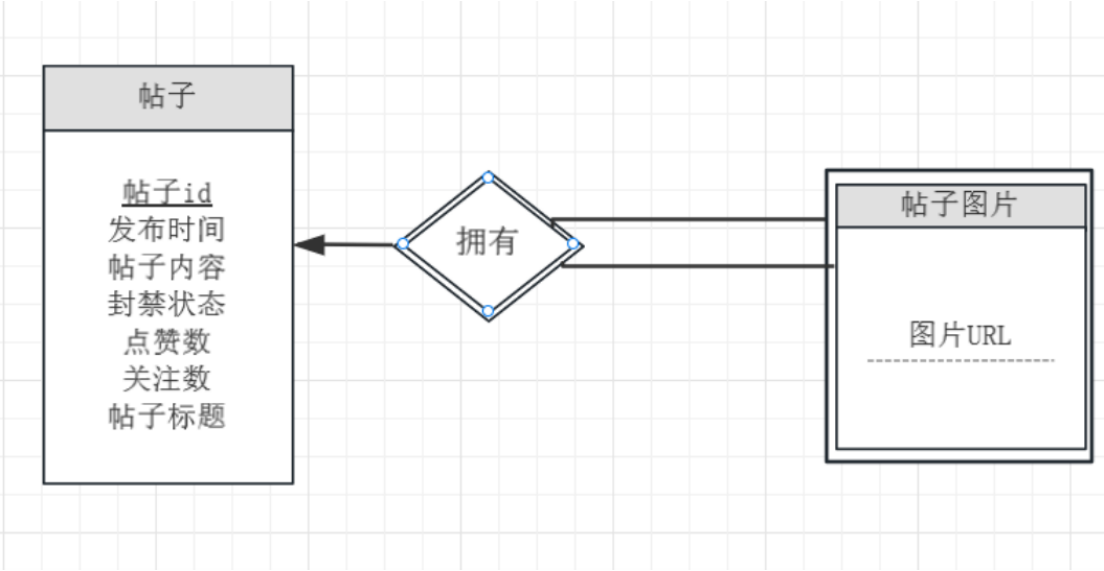


论坛帖子关系ER图

- 帖子拥有tag

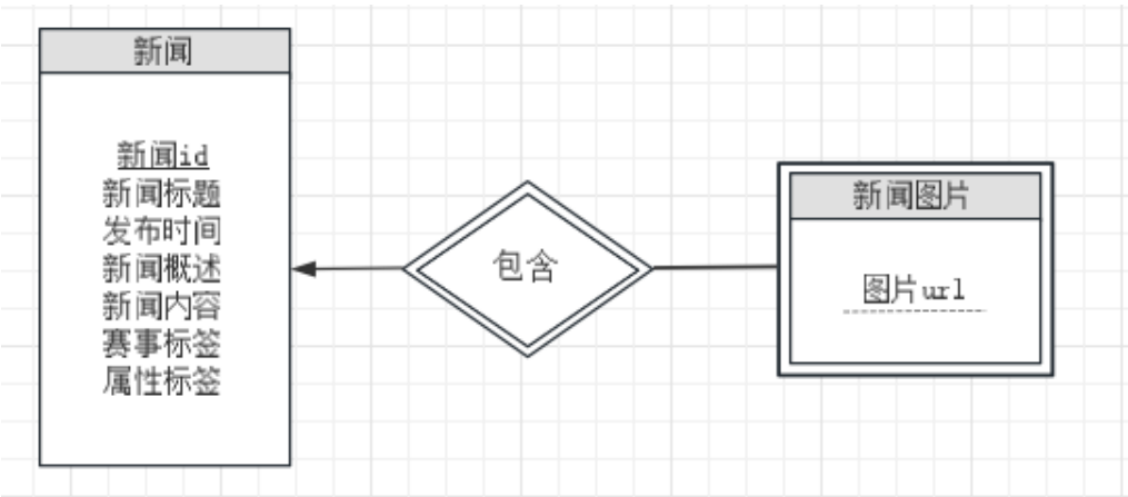


- 帖子拥有图片



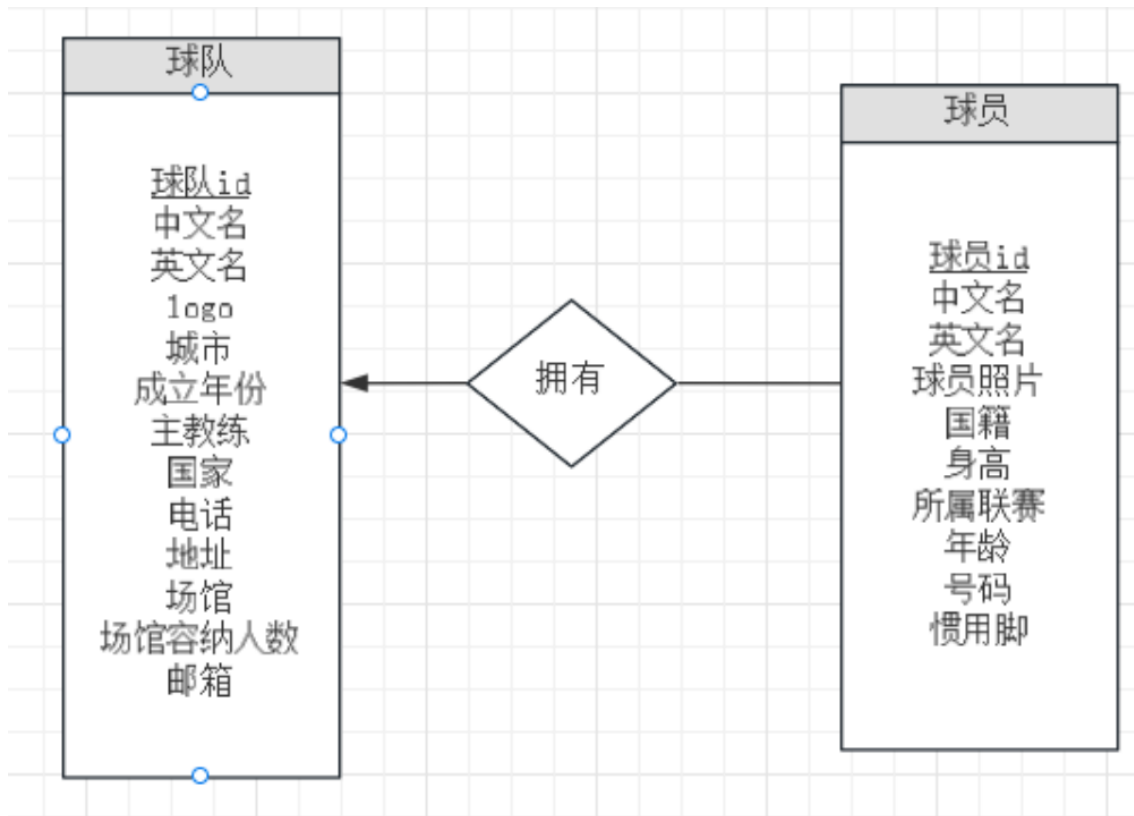
新闻ER图

- 新闻包含图片



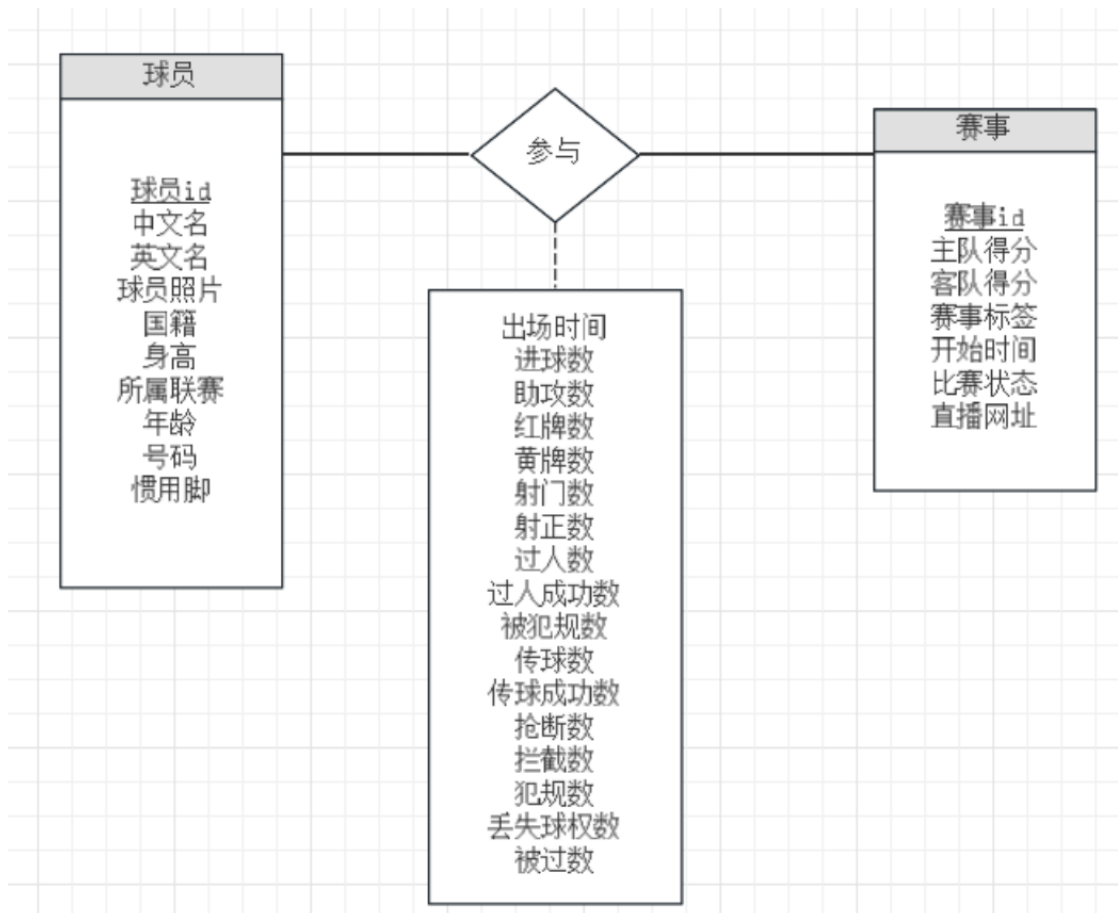
球队球员赛事关系ER图

- 球队拥有球员

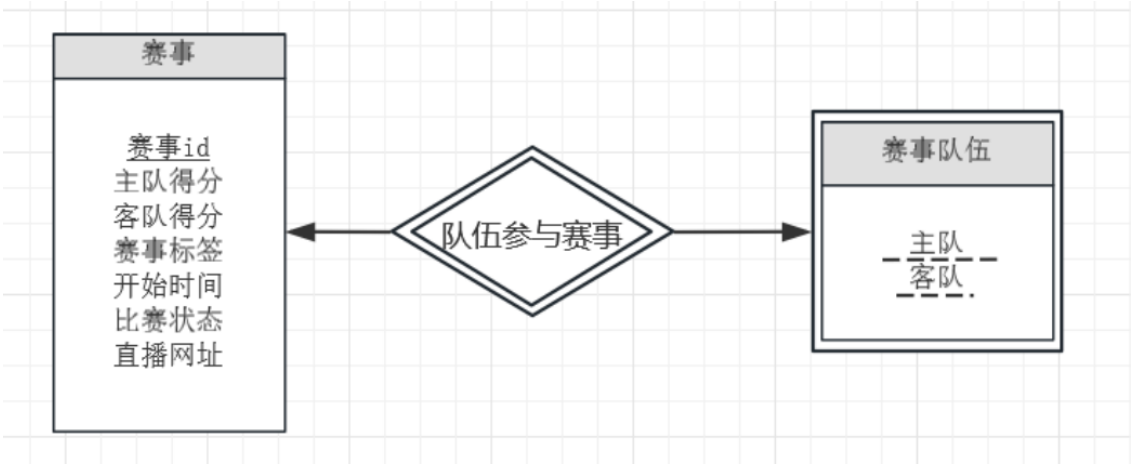


- 球员参加比赛

附加属性为球员的表现数据球队参与比赛



• 球队参与比赛



1.2) 逻辑设计

表设计

- 1. 通知：记录所有站务通知内容
- 2. 球员参赛记录：记录球员参赛的记录信息
- 3. 球员：记录球员基本信息
- 4. 帖子图片：记录帖子的图片
- 5. 帖子：记录所有帖子内容
- 6. 帖子发布记录：记录发帖人和帖子的相关信息
- 7. 举报：记录举报人和被举报帖子帖子的相关信息
- 8. 帖子标签：记录帖子与标签的关系
- 9. 球队：记录球队基本信息
- 10. 球队签约球员：记录球员与球队的归属关系
- 11. 主题：记录主题基本信息
- 12. 主队：记录用户最喜欢的球队信息
- 13. 用户：记录用户基本信息
- 14. 视频：记录视频基本信息
- 15. 管理员发布公告：记录管理员与公告的对应关系
- 16. 管理员：记录管理员账户信息
- 17. 登录：记录用户的登录信息
- 18. 收藏帖子：记录用户和帖子的收藏关系
- 19. 评论：记录用户对帖子的评论关系
- 20. 关注：记录用户间的关注信息
- 21. 赛事：记录比赛信息
- 22. 赛事球队：记录参与赛事的球队信息
- 23. 高光时刻：记录高光时刻
- 24. 喜欢帖子：记录用户与帖子的喜欢关系
- 25. 新闻：记录新闻信息
- 26. 新闻图片：记录帖子的所有图片

通知表

属性	类型	特殊属性	备注
NOTICE_ID	NUMBER	主键	通知id
TEXT	VARCHAR2	非空	通知内容
NOTICE_CREATE_TIME	DATE	非空	发布时间

PUBLISHDATE TIME	DATE	非空	发布时间
属性	类型	特殊属性	备注
RECEIVER	NUMBER	非空	发布对象，缺省为0，表示广播

球员参赛表

属性	类型	特殊属性	备注
GAME_ID	NUMBER	主键，外键	赛事id
PLAYER_ID	NUMBER	主键，外键	球员id
MINUTES	NUMBER	非空	上场时间
GOAL	NUMBER	非空	进球数
ASSIST	NUMBER	非空	助攻数
RED	NUMBER	非空	红牌数
YELLOW	NUMBER	非空	黄牌数
SHOOT	NUMBER	非空	射门数
TARGET	NUMBER	非空	射正数
SURPASS	NUMBER	非空	过人数
SURPASS_SUCCESS	NUMBER	非空	过人成功数
FOULED	NUMBER	非空	被犯规数
PASS	NUMBER	非空	传球数
PASS_RATE	NUMBER	非空	传球成功数
TACKLE	NUMBER	非空	抢断数
INTERCEPT	NUMBER	非空	拦截数
FOUL	NUMBER	非空	犯规数
LOST	NUMBER	非空	丢失球权数
SURPASSED	NUMBER	非空	被过数

球员表

属性	类型	特殊属性	备注
PLAYER_ID	NUMBER	主键	球员id
CHINESENAME	VARCHAR2	非空	中文名
ENNAME	VARCHAR2	非空	英文名
PHOTO	VARCHAR2	非空	照片
COUNTRY	VARCHAR2	非空	国籍

属性	类型	特殊属性	备注
HEIGHT	VARCHAR2	非空	身高
TYPE	VARCHAR2	非空	所属联赛
AGE	VARCHAR2	非空	年龄
PLAYERNUMBER	VARCHAR2	非空	号码
FOOT	VARCHAR2	非空	惯用脚

帖子图片表

属性	类型	特殊属性	备注
POST_ID	NUMBER	主键	帖子id
PIC	VARCHAR2	主键	图片URL地址

帖子表

属性	类型	特殊属性	备注
POST_ID	NUMBER	主键	帖子id
PUBLISHDATETIME	DATE	非空	发布时间
CONTAINS	VARCHAR2	非空	帖子内容
ISBANNED	NUMBER(1,0)	非空	是否封禁
APPROVALNUM	NUMBER	非空	点赞数
FAVOURITENUM	NUMBER	非空	收藏数
TITLE	VARCHAR2	非空	标题

发布帖子表

属性	类型	特殊属性	备注
POST_ID	NUMBER	主键，外键	帖子id
USER_ID	NUMBER	主键，外键	发帖人id

举报表

属性	类型	特殊属性	备注
POST_ID	NUMBER	主键，外键	帖子id
REPORTER_ID	NUMBER	主键，外键	举报人id
DESCRIPTIONS	VARCHAR2		举报描述
REPLY	VARCHAR2		管理员回复
REPORT_TIME	DATE	非空	举报时间

属性	类型	特殊属性	备注
POST_ID	NUMBER	主键，外键	帖子id
TAGNAME	VARCHAR2	主键	标签名称

帖子标签表

属性	类型	特殊属性	备注
POST_ID	NUMBER	主键，外键	帖子id
TAGNAME	VARCHAR2	主键	标签名称

球队表

属性	类型	特殊属性	备注
TEAM_ID	NUMBER	主键	球队id
CHINESENAME	VARCHAR2	非空	中文名
ENNAME	VARCHAR2	非空	英文名
LOGO	VARCHAR2	非空	球队LOGO
CITY	VARCHAR2	非空	城市
FOUNDEDYEAR	NUMBER		成立年份
COACH	VARCHAR2	非空	主教练
COUNTRY	VARCHAR2	非空	国家
TELEPHONE	VARCHAR2	非空	电话
ADDRESS	VARCHAR2	非空	地址
VENUE_NAME	VARCHAR2	非空	场馆
VENUE_CAPACITY	NUMBER		场馆容纳人数
EMAIL	VARCHAR2	非空	邮箱

球队签约球员表

属性	类型	特殊属性	备注
TEAM_ID	NUMBER	主键，外键	球队id
PLAYER_ID	NUMBER	主键，外键	球员表

主题表

属性	类型	特殊属性	备注
ID	NUMBER	主键	主题id
NAME	VARCHAR2	非空	主题名称
IMAGE1	VARCHAR2	非空	图1
IMAGE2	VARCHAR2	非空	图2
IMAGE3	VARCHAR2	非空	图3

管理员发布公告

属性	类型	特殊属性	备注
admin_id	number	主码、非空、外码	管理员id
notice_id	number	主码、非空、外码	公告id

管理员

属性	类型	特殊属性	备注
admin_id	number	主码、非空	管理员id
adminName	varchar2	非空	姓名
adminPassword	varchar2	非空	密码
createDatetime	date	非空	创建时间

用户登录

属性	类型	特殊属性	备注
user_id	number	主码、非空、外码	用户id
sign_in_date	date	主码、非空	签到时间

用户收藏帖子

属性	类型	特殊属性	备注
user_id	number	主码、非空、外码	用户id
post_id	number	主码、非空、外码	帖子id
createDatetime	date	非空	创建时间

用户评论帖子

属性	类型	特殊属性	备注
publishDatetime	date	主码、非空	发布时间
contains	varchar2	非空	内容
post_id	number	主码、非空、外码	帖子id
user_id	number	主码、非空、外码	用户id

用户关注用户

属性	类型	特殊属性	备注
follower_id	number	主码、非空、外码	关注者id
follow_id	number	主码、非空、外码	被关注者id
createDatetime	date	非空	创建时间

赛事

属性	类型	特殊属性	备注
game_id	number	主码、非空	赛事id
type	varchar2	非空	类型
startTime	date	非空	开始时间
status	varchar2	非空	状态
liveUrl	varchar2	非空	直播链接
homeScore	number	非空	主队分数
guestScore	number	非空	客队分数

赛事队伍

属性	类型	特殊属性	备注
game_id	number	主码、非空、外码	比赛id
homeTeam	number	非空、外码	主队id
guestTeam	number	非空、外码	客队id

guestTeam 属性	number 类型	非空、外码 特殊属性	客队id 备注
-----------------	--------------	---------------	------------

高光时刻

属性	类型	特殊属性	备注
highlight_id	number	主码、非空	高光id
photo	varchar2		图片
videoUrl	varchar2		视频链接

用户赞同帖子

属性	类型	特殊属性	备注
user_id	number	主码、非空、外码	用户id
post_id	number	主码、非空、外码	帖子id
createDatetime	date	非空	创建时间

新闻

属性	类型	特殊属性	备注
news_id	number	主码、非空	新闻id
title	varchar2	非空	标题
publishDatetime	date	非空	发布时间
summary	varchar2	非空	简略信息
contains	varchar2	非空	内容
matchTag	varchar2	非空	赛事tag
propertyTag	varchar2	非空	类型tag

新闻拥有图片

属性	类型	特殊属性	备注
news_id	number	主码、非空、外码	新闻id
pictureRoute	varchar2	非空	图片路径

视频

属性	类型	特殊属性	备注
video_id	number	主码、非空	视频id
matcgTag	varchar2	非空	赛事tag
propertyTag	varchar2	非空	类型tag
title	varchar2	非空	标题
cover	varchar2	非空	封面图片
urlLink	varchar2	非空	视频链接

用户

属性	类型	特殊属性	备注
user_id	number	主码、非空	用户id
userName	varchar2	非空	用户昵称
userPassword	varchar2	非空	用户密码
userAccount	varchar2	非空	用户账户
isbanned	number	非空	是否被封禁
userPoint	number	非空	积分
themeType	number	非空	风格
signature	varchar2		签名

属性	类型	特殊属性	备注
createDatetime	date	非空	创建时间
userSecAns	varchar2	非空	安全问题
userSecQue	varchar2	非空	安全密码
avatar	varchar2		头像
followedNumber	number	非空	被关注数
followNumber	number	非空	关注数
signDate	date		签到时间

用户收藏球队

属性	类型	特殊属性	备注
user_id	number	主码、非空、外码	用户id
team_id	number	主码、非空、外码	队伍id
createDatetime	date	非空	创建时间
modifyDatetime	date		修改时间

1.3) 部署策略

部署选择

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的 Linux或Windows操作系统的机器上，也可以实现虚拟化。

```
modified
阿里云->腾讯云
```

本次课设中，我们采用腾讯云ubuntu服务器，使用docker部署oracle数据库，理由如下：

- 1. docker安装快速，效率高
- 2. docker隔离性好，可以安装多个oracle实例，互不干扰，只要管理好对主机端口的映射即可
- 3. 卸载管理更方便和干净，直接删除容器和镜像即可
- 4. 数据备份、迁移功能更加方便强大
- 5. 性能与直接安装接近，但是启动速度远快于直接安装

部署流程

docker的安装较为简单，直接apt安装即可，使用

```
$ docker search oracle
```

可以搜索所有支持的oracle镜像，在此选择使用oracle12c版本
使用

```
$ docker pull truevolly/oracle-12c
```

拉取镜像后，使用

```
$ docker run -d -p 1521:1521 truevolly/oracle-12c
```

即可将镜像实例化。通过truevolly/oracle-12c镜像实例化了一个容器，同时指定容器的1521端口（oracle的默认端口）映射到了宿主服务器的1521端口。实际上在此省略了一个容器卷的部署，在后面将提及

部署成功后，使用

```
$ docker exec -it ${镜像名} bash
```

即可进入容器，使用

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

可以进入数据库实例的sqlplus控制台

1.4) 库设计

表设计

• 保持规范性

- 为了保证数据库中数据的完整性，在表格设计时考虑了一些缺省、非空约束、check约束等例如在News表中

```
create table News (  
    news_id number not null,  
    title varchar2(128) not null,  
    publishDateTime date not null,  
    summary varchar2(512) not null,  
    contains varchar2(3999) not null,  
    matchTag varchar2(20) not null,  
    propertyTag varchar2(20) not null,  
    primary key (news_id),  
    check (matchTag in ('中超', '西甲', '意甲', '法甲', '德甲', '英超'))  
);
```

对于新闻的id、标题、发布时间、简略、内容、比赛tag、类型tag都进行了非空约束，对比赛tag还额外进行了check in 约束，保证数据的合法性。

而在Posts表中

modified

修改三行sql

```
create table Posts (
    post_id number not null,
    publishDateTime date not null,
    contains varchar2(1000) not null,
    isBanned number(1,0) default on null 0 not null,
    approvalNum number default on null 0 not null,
    favouriteNum number default on null 0 not null,
    title varchar2(50) not null,
    primary key (post_id)
);
```

除了一些非空约束以外，还设定了缺省值。例如封禁标志isBanned就缺省为0

- 为了数据库的可维护性，还需要对部分外码设置级联。如果发生进行删除用户等操作时发现由于完整性约束使得操作不可进行等事件，数据库的可维护性显然不足

例如在用户发表帖子的关系集PublishPost中

```
create table PublishPost (
    post_id number not null,
    user_id number not null,
    primary key (post_id, user_id),
    foreign key (post_id) references Posts on delete cascade,
    foreign key (user_id) references Usr on delete cascade
);
```

对Posts(post_id)和Usr(user_id)的外码引用设计了级联删除，这保证了在删除帖子或者删除用户时，对应的PublishPost关系也会被移除，数据库事务保持了一致性

而在球队球员联系集TeamOwnPlayer中

```
create table TeamOwnPlayer (
    team_id number,
    player_id number,
    primary key (team_id, player_id),
    foreign key (team_id) references team on delete cascade,
    foreign key (player_id) references players on delete cascade
);
```

对team(team_id)和players(player_id)的外码引用也设计了级联删除，在队伍或者球员信息被删除时，对应的TeamOwnPlayer关系也会被安全移除

- 在项目的设计中，有一些数据是经常同时需要，但又不能通过简单逻辑查询的，例如在获取Posts的数据时，倾向于同时获取其点赞数、收藏数，严格按照3NF设计的数据库中，点赞数、收藏数需要Posts表与likePost、collectPost表进行联表查询，这会造成性能的浪费以及后端逻辑的复杂性。

为了解决这个问题，我们尝试在数据库中增加了一些冗余字段，但造成数据库设计的不规范，破坏数据库的一致性问题，因此我们通过触发器对其进行维护。下面对触发器进行说明：

- incrFollowedNumber、decrFollowedNumber

```

create or replace TRIGGER decrFollowedNumber
BEFORE DELETE ON follow
FOR EACH ROW
BEGIN
    UPDATE usr
    SET followedNumber = followedNumber - 1
    WHERE USER_ID = :OLD.FOLLOW_ID;
END;

```

```

create or replace TRIGGER incrFollowedNumber
AFTER INSERT ON follow
FOR EACH ROW
BEGIN
    UPDATE usr
    SET followedNumber = followedNumber + 1
    WHERE USER_ID = :NEW.FOLLOW_ID;
END;

```

这两个触发器对用户的被关注数进行维护，当follow表单被进行插入或者删除操作时，对follow_id对应的用户的followedNumber字段进行自增、自减操作

- incrFollowNumber、decrFollowNumber

```

create or replace TRIGGER decrFollowNumber
BEFORE DELETE ON follow
FOR EACH ROW
BEGIN
    UPDATE usr
    SET followNumber = followNumber - 1
    WHERE USER_ID = :OLD.FOLLOWER_ID;
END;

```

```

create or replace TRIGGER incrFollowNumber
AFTER INSERT ON follow
FOR EACH ROW
BEGIN
    UPDATE usr
    SET followNumber = followNumber + 1
    WHERE USER_ID = :NEW.FOLLOWER_ID;
END;

```

这两个触发器对用户的关注数进行维护，当follow表单被进行插入或者删除操作时，对follower_id对应的用户的followNumber字段进行自增、自减操作

- incrPostApprovalNum、decrPostApprovalNum

```

create or replace TRIGGER decrPostApprovalNum
BEFORE DELETE ON LIKEPOST
FOR EACH ROW
BEGIN
    UPDATE posts
    SET approvalnum = approvalnum - 1
    WHERE post_id = :OLD.post_id;
END;

```

```

create or replace TRIGGER incrPostApprovalNum
AFTER INSERT ON LIKEPOST
FOR EACH ROW
BEGIN
    UPDATE posts
    SET approvalnum = approvalnum + 1
    WHERE post_id = :NEW.post_id;
END;

```

这两个触发器对帖子的赞同数进行维护，当LIKEPOST表单被进行插入或者删除操作时，对post_id对应的帖子的approvalnum字段进行自增、自减操作

modified

修改

- update_game_score_after_insert、update_game_score_after_delete

```

create or replace trigger update_game_score_after_insert
after insert on playerjoingame
for each row
declare
    this_team_id number;
    this_hometeam number;
    this_guestteam number;
begin

    -- 获取球员所属队伍的team_id
    select team_id into this_team_id
    from teamownplayer top
    where top.player_id=:new.player_id;

    -- 获取赛事的主客队id
    select hometeam into this_hometeam
    from gameteam gt
    where gt.game_id=:new.game_id;

    select guestteam into this_guestteam
    from gameteam gt
    where gt.game_id=:new.game_id;

    --更新主队得分
    update game g
    set g.homescore=g.homescore+:new.goal
    where g.game_id=:new.game_id and this_team_id=this_hometeam;

```



```

--更新客队得分
update game g
set g.guestscore=g.guestscore+:new.goal
where g.game_id=:new.game_id and this_team_id=this_guestteam;

end;

```

```

create or replace trigger update_game_score_after_delete
after delete on playerjoingame
for each row
declare
    this_team_id number;
    this_hometeam number;
    this_guestteam number;
begin

    -- 获取球员所属队伍的team_id
    select team_id into this_team_id
    from teamownplayer top
    where top.player_id=:old.player_id;

    -- 获取赛事的主客队id
    select hometeam into this_hometeam
    from gameteam gt
    where gt.game_id=:old.game_id;

    select guestteam into this_guestteam
    from gameteam gt
    where gt.game_id=:old.game_id;

    --更新主队得分
    update game g
    set g.homescore=g.homescore-:old.goal
    where g.game_id=:old.game_id and this_team_id=this_hometeam;

    --更新客队得分
    update game g
    set g.guestscore=g.guestscore-:old.goal
    where g.game_id=:old.game_id and this_team_id=this_guestteam;

end;

```

这两个触发器对比赛的得分进行维护，当playerJoinGame表单被进行插入或者删除操作时，对game中对应的帖子的homeScoe、guestScore字段进行自增、自减操作

- incrPostCollect、decrPostCollect

```

create or replace TRIGGER decrPostCollect
BEFORE DELETE ON COLLECTPOST
FOR EACH ROW
BEGIN
    UPDATE posts
    SET favouritenum = favouritenum - 1
    WHERE post_id = :OLD.post_id;
END;

```

```

create or replace TRIGGER incrPostCollect
BEFORE INSERT ON COLLECTPOST
FOR EACH ROW
BEGIN
    UPDATE posts
    SET favouritenum = favouritenum + 1
    WHERE post_id = :NEW.post_id;
END;

```

这两个触发器对帖子的收藏数进行维护，当COLLECTPOST表单被进行插入或者删除操作时，对post_id对应的帖子的favouritenum字段进行自增、自减操作

■ unique_report_trigger

```

create or replace TRIGGER unique_report_trigger
BEFORE INSERT ON reports
FOR EACH ROW
DECLARE
    existing_report_count NUMBER;
BEGIN
    IF :NEW.status = 'unhandled' THEN
        -- 检查是否存在其他相同的记录
        SELECT COUNT(*)
        INTO existing_report_count
        FROM reports
        WHERE reporter_id = :NEW.reporter_id
        AND post_id = :NEW.post_id
        AND status = 'unhandled'
        AND report_time != :NEW.report_time; -- 排除当前记录

        -- 如果存在其他相同的记录，阻止插入或更新操作
        IF existing_report_count > 0 THEN
            RAISE_APPLICATION_ERROR(-20001, '同一举报者对同一帖子的举报只能存在一个。');
        END IF;
    END IF;
END;

```

这个触发器是辅助完成业务逻辑设置的。

业务逻辑包含：同一个用户对同一个帖子的状态为'unhandled'的举报在同一时间最多只能存在一个，而使用触发器可以很好的在违反约束的事务进行前进行阻止。

当数据被插入reports时，先计数表格中同一个用户对同一个帖子的状态为'unhandled'的举报，当此种数据已经存在时，阻止插入事务并抛出报错信息。

优化尝试

- 对赛事通过时间进行排序的操作比较多，而在sql developer中进行该操作大概需要1.6s
尝试优化加速一下，因此在startTime上创建索引

```
create index game_time_index on game(starttime)
```

创建后对赛事通过时间进行排序大概需要1.4-1.5秒，优化不是非常显著，这是由于在数据的爬取时基本是按照时间进行的，导致数据处于有序状态。

1.5) 配置管理

数据持久化

容器通常是临时的、易于销毁和重新创建的。尤其是为了应对数据的不慎销毁问题，我们需要进行数据持久化处理来保证数据库的持久性和可靠性。

选择docker部署的另一个考虑就基于此。docker官方提供了一个叫做数据卷(Volume)的数据持久化方案。

数据卷是宿主机中的一个目录或者文件，当容器目录和数据卷目录绑定后，对方的修改会立即同步。

一个数据卷可以被多个容器同时挂载，一个容器也可以被挂载多个数据卷。

简单来说，数据卷本质其实是共享文件夹，是宿主机与容器之间数据共享的桥梁。

根据原理，数据卷将一个docker容器中的数据挂载到服务器中的文件上，在之后通过对该文件的操作可以轻松实现数据的备份、共享以及修改。

数据卷的创建并不复杂，创建数据卷并在实例化容器时指定数据卷挂载即可

```
$ docker volume create dbdata
$ docker run -d -p 1521:1521 -v dbdata:${path} truevol/oracle-12c
```

之后可以查看容器的具体信息

```
$ sudo docker volume inspect dbdata
```

可以看到下列输出：

```
ubuntu@VM-4-9-ubuntu:~$ sudo docker volume inspect dbdata
[
  {
    "CreatedAt": "2023-08-16T17:28:57+08:00",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/dbdata/_data",
    "Name": "dbdata",
    "Options": {},
    "Scope": "local"
  }
]
```

对服务器路径 `"/var/lib/docker/volumes/dbdata/_data"` 下的文件进行定时备份即可完成数据的持久化处理。当不慎删除了容器时，数据卷并不会被删除，只需要重新实例化容器并指定挂载的数据卷即可恢复数据

参数文件

Oracle中的参数文件是一个包含一系列参数以及参数对应值的操作系统文件。它们是在数据库实例启动第一个阶段时候加载的，决定了数据库的物理结构、内存、数据库的限制及系统大量的默认值、数据库的各种物理属性、指定数据库控制文件名和路径等信息，是进行数据库设计的重要文件。

更旧版本的oracle只采用了pfile一种参数文件格式，在本项目采用的12c版本中已经加入了spfile文件。

pfile可以通过文本编辑器直接修改参数值，并且需要手动重新启动数据库实例才能使更改生效。

spfile是二进制文件格式，也包含了Oracle数据库实例的配置参数，它不能直接通过文本编辑器进行修改，而是通过Oracle提供的ALTER SYSTEM语句或图形界面工具进行更改。

oracle实例默认是根据spfile文件进行启动的。由于希望修改一下会话时长的限制，使用了ALTER SYSTEM语句对spfile文件进行修改，并根据spfile文件与pfile文件进行了相互生成

```
create spfile from pfile
create pfile from spfile
```

由于疏忽并没有指定生成的文件的路径以及文件名，导致了文件的覆盖。当我关闭了数据库实例想要重启时候，无论是通过缺省方式还是指定pfile，都会给我报错参数错误

spfile无法查看，通过

```
vim /u01/app/oracle/product/12.1.0/xe/dbs/init.ora
```

发现内容为空，根据其生成的spfile也显然无法使用

根据官方文档，oracle有一个默认的配置文件init，理解为pfile文件的例子即可。尝试根据其启动数据库。首先在目录中找到该文件

```
/u01/app/oracle/admin/xe/pfile/init.ora.716202394632
```

对其进行备份后修改会话时长限制

尝试在sqlplus中启动数据库实例:

```
SQL> startup pfile="/u01/app/oracle/admin/xe/pfile/init.ora.716202394632"
...
Database mounted.
```

成功启动，之后根据其创建spfile、根据spfile生成pfile并进行备份，保证数据库配置文件的安全。

日志信息

日志信息是记录了数据库的变更，包括数据插入、更新、删除以及数据定义语言（DDL）操作等信息的文件，Oracle 数据库的日志是确保数据库可恢复性、高可用性和一致性的关键组成部分。对其的定期备份也是需要考虑的部分。

首先进入sqlplus停止监听服务

```
$ lsnrctl set log_status off;
```

进入监听文件的目录下,查看文件信息

```
$ cd /u01/app/oracle/diag/tnslsnr/604dc6ad6ec8/listener/trace
/u01/app/oracle/diag/tnslsnr/604dc6ad6ec8/listener/trace$ ls
listener.log
```

查看一下文件内容

```
$ vi listener.log
...
31-AUG-2023 03:40:32 * (CONNECT_DATA=(SERVICE_NAME=xe)(CID=
(PROGRAM=/home/ubuntu/DataBase/DBwebAPI/DBwebAPI/bin/Debug/net6.0/DBwebAPI.dll)
(HOST=VM-4-11-ubuntu)(USER=ubuntu))) * (ADDRESS=(PROTOCOL=tcp)
(HOST=xxx.xx.xxx.xxx)(PORT=xxxxx)) * establish * xe * 0
...
31-AUG-2023 03:41:32 * service_update * xe * 0
...
```

一切正常。将listener.log文件更名备份后, 创建一个空文件listener.log放在原目录下,最后进入sqlplus启动监听服务

```
$ lsnrctl set log_status on;
```

modified

增加

表空间设置

在oracle官方的图形化工具sql developer中查看一下表空间的使用情况

	TABLESPACE_NAME	PERCENT_USED	PCT_USED	ALLOCATED	USED	FREE	DATAFILES
1	SYSTEM	<div><div></div></div>	99.47	800	795.75	4.25	1
2	SYSAUX	<div><div></div></div>	93.85	820	769.56	50.44	1
3	USERS	<div><div></div></div>	85.45	13.75	11.75	2	1
4	UNDOTBS1	<div><div></div></div>	8.75	130	11.38	118.63	1
5	TESTSPACE	<div><div></div></div>	2	50	1	49	1
6	TEMP	<div><div></div></div>	0.51	197	1	196	1

其中项目中的用户文件是存放在USERS空间的。可以看到其空间占用较大, 而SYSTEM和SYSAUX也几乎占满。这是由于SYSTEM和SYSAUX空间本就是自动增长的, 在占满并自动增长之后数据占用就一直有着较高比例.

在本此设计中, 同一个表空间的所属数据文件都已有一个, 对表空间的修改可以简化为对数据文件的修改。

查看一下数据文件设置

```
SELECT * FROM dba_data_files;
```

发现其自动增长AUTOEXTENSIBLE都为YES，但是拓展大小NEXT都为空。

默认情况下，Oracle 数据库的 NEXT 大小通常会被设置为一个相对较小的固定值，通常是数兆字节。每次自动增长时，数据文件的大小只会增加一小部分。如果数据库经常执行大型操作或具有高数据增长率，这可能导致频繁的文件增长，从而导致额外的磁盘 I/O 和管理开销。数据文件也会快速达到最大限制，从而引发数据库性能问题和操作中斷。

因此对表空间进行管理

```
alter database datafile '/u01/app/oracle/oradata/xe/users01.dbf' resize 200M ;
```

利用此语句可以将数据文件大小设置为200M。根据上述信息所示，USERS表空间分配100M的空间是完全足够的。







```
alter database datafile '/u01/app/oracle/oradata/xe/users01.dbf' autoextend on  
next 20M ;
```

之后执行以上语句使数据文件自动增长步长为20M。

对系统级的文件操作希望更加慎重，因此进行RESIZE操作了。但是应该设置自动拓展步长。

```
alter database datafile '/u01/app/oracle/oradata/xe/system01.dbf' autoextend on  
next 100M ;  
alter database datafile '/u01/app/oracle/oradata/xe/sysaux01.dbf' autoextend on  
next 100M ;
```

现在再次查看表空间

	TABLESPACE_NAME	PERCENT_USED	PCT_USED	ALLOCATED	USED	FREE	DATAFILES
1	SYSTEM		99.47	800	795.75	4.25	1
2	SYSAUX		93.85	820	769.56	50.44	1
3	TESTSPACE		10	10	1	9	1
4	UNDOTBS1		8.75	130	11.38	118.63	1
5	USERS		5.87	200	11.75	188.25	1
6	TEMP		0.51	197	1	196	1

用户表空间十分充足