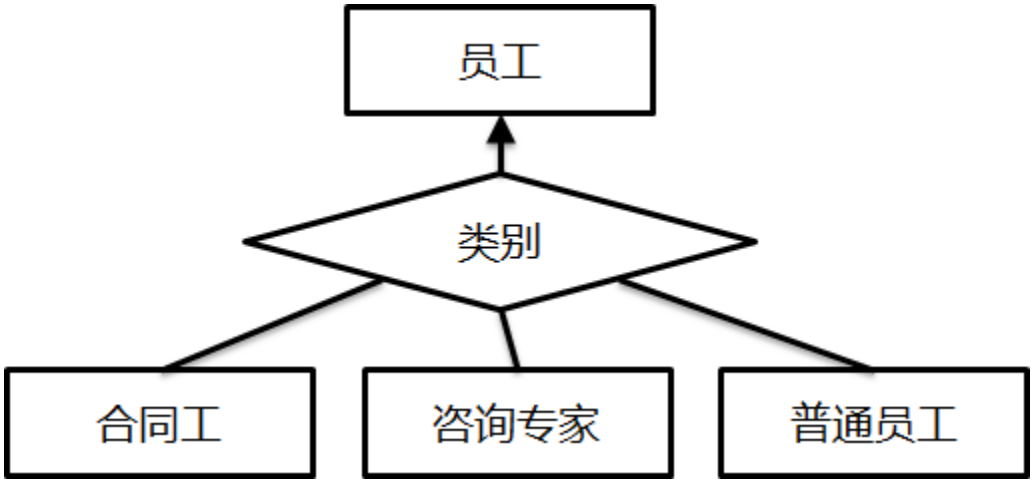
1. 下图所示的 ER 图刻画了四类实体，对“员工”而言，另三类实体（合同工、咨询专家和普通员工）都是它的子类。也就是说，员工拥有的属性，另三类实体都拥有；而子类拥有的属性父类未必拥有。请尝试设计这四类实体对应的关系表。



在设计实体-关系（ER）模型时，通常会为每个实体类创建一个关系表。在这个情况下，我们有“员工”这个父类，以及三个子类：“合同工”、“咨询专家”和“普通员工”。子类继承父类的属性，并可能有它们自己独特的属性。
下面是如何为这四类实体设计关系表：

员工（Employees）表 - 存储所有员工共有的属性，如：

员工 ID（EmployeeID） - 主键
姓名（Name）
地址（Address）
电话（Phone）
雇佣日期（HireDate）
…（任何其他员工共有的属性）

合同工（Contractors）表 - 存储合同工特有的属性，同时包括员工 ID 作为外键：

员工 ID（EmployeeID） - 外键，关联到员工表
合同结束日期（ContractEndDate）
日薪（DailyRate）
…（任何其他合同工特有的属性）

咨询专家（Consultants）表 - 存储咨询专家特有的属性，同样包括员工 ID 作为外键：

员工 ID（EmployeeID） - 外键，关联到员工表
专业领域（FieldOfExpertise）
咨询费率（ConsultancyRate）
…（任何其他咨询专家特有的属性）

普通员工（RegularEmployees）表 - 存储普通员工特有的属性，也包括员工 ID 作为外键：
员工 ID（EmployeeID）- 外键，关联到员工表
部门（Department）
职位（Position）
年薪（AnnualSalary）
…（任何其他普通员工特有的属性）
在这种设计中，**员工 ID 在所有子类表中作为外键**，确保了数据的**一致性和完整性**。每个子类表都只包含与之特别相关的属性，**父类表则包含所有子类共有的属性**。在实际应用中，可能还需要考虑其他的设计要素，比如继承策略（单表继承、具体表继承、类表继承等），以及如何高效地处理多态查询和更新操作。

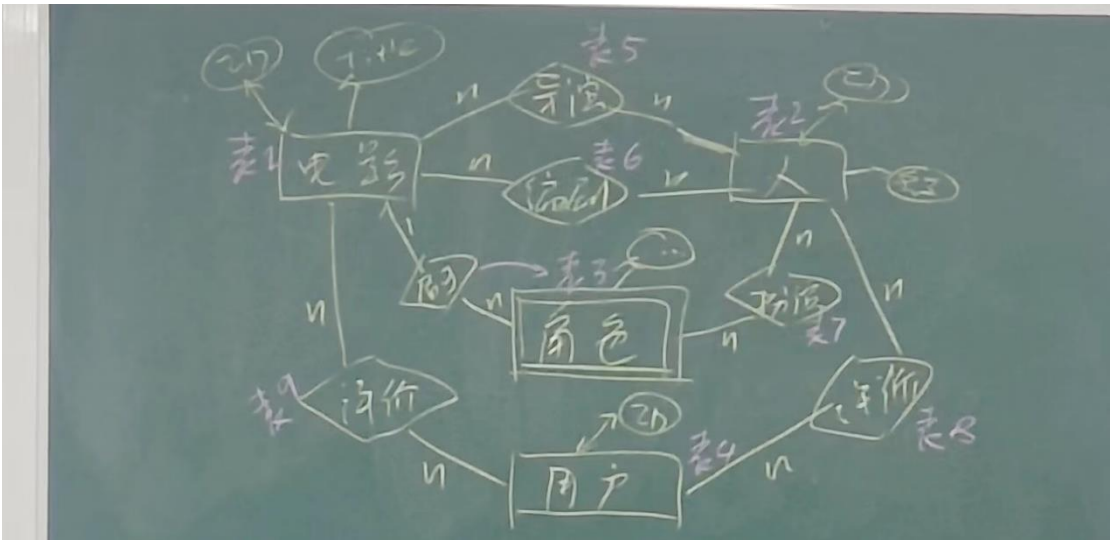
2. 请针对以下需求设计 ER 图，并构建相应的关系模式：

   一个关于电影、制作人员和演员的网站（类似一个简易的 IMDB 网站）。用户可以浏览每一部电影的简介和相关信息（出品年份、电影类型、时长、评级），以及其导演、编剧和演员的列表。用户还可以浏览每一位导演、编剧或演员的信息（姓名、性别、年龄、简介），以及他们参与过哪些电影作品。导演、编剧或演员只是职位，一个人可以身兼数职。
   每一位演员在其参演的电影中都扮演一定的角色。用户在浏览电影时，除了能看到演员信息，还能看到每位演员扮演了什么角色。用户在浏览演员时，除了能看到他（或她）参演的电影，还能看到他（或她）在每一部电影中扮演的角色。（注意：一位演员可以在一部电影中扮演多个角色。同一个角色也可能由多名演员扮演，比如，年少时由一位演员扮演，年老时由另一位演员扮演。）
   用户登录后还可以针对每一部电影、每一位导演、编剧或演员进行评价和打分，供别人参考。

在设计 ER 图和关系模式时，我们需要确定实体（Entities）、实体之间的关系（Relationships）以及属性（Attributes）。基于您提供的需求，以下是一个简化的 ER 图设计和关系模式的描述。
ER 图设计
实体：
电影（Movie）
属性：电影 ID（MovieID），电影名称（Title），出品年份（Year），电影类型（Genre），时长（Duration），评级（Rating）
职位（Position）
属性：职位 ID（PositionID），职位名称（PositionName）（如导演、编剧、演员）
人员（Person）

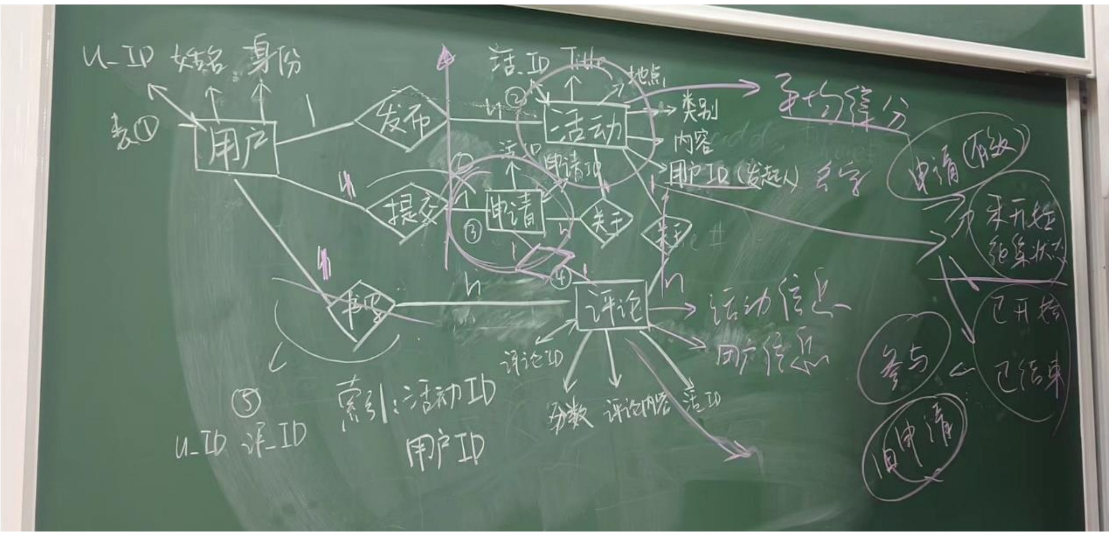
属性：人员 ID (PersonID)，姓名 (Name)，性别 (Gender)，年龄 (Age)，简介 (Bio)
角色 (Role)
属性：角色 ID (RoleID)，角色名称 (RoleName)
用户 (User)
属性：用户 ID (UserID)，用户名 (Username)
关系：
参与 (Participates) - 人员与电影之间的多对多关系，表示导演、编剧或演员参与了哪些电影。
属性：人员 ID (PersonID)，电影 ID (MovieID)，职位 ID (PositionID)
扮演 (Plays) - 演员与角色之间的多对多关系，表示演员在特定电影中扮演的角色。
属性：人员 ID (PersonID)，角色 ID (RoleID)，电影 ID (MovieID)
评价 (Reviews) - 用户与电影、人员之间的多对多关系，表示用户对电影或人员的评价。
属性：用户 ID (UserID)，电影 ID (MovieID) / 人员 ID (PersonID)，评分 (Score)，评论 (Comment)
关系模式设计
以下是基于上述 ER 图设计的关系模式：
电影 (Movie)
MovieID (PK)
Title
Year
Genre
Duration
Rating
职位 (Position)
PositionID (PK)
PositionName
人员 (Person)
PersonID (PK)
Name
Gender
Age
Bio
角色 (Role)
RoleID (PK)
RoleName
用户 (User)
UserID (PK)
Username
参与 (Participates)
PersonID (FK)
MovieID (FK)
PositionID (FK)
扮演 (Plays)
PersonID (FK)
RoleID (FK)

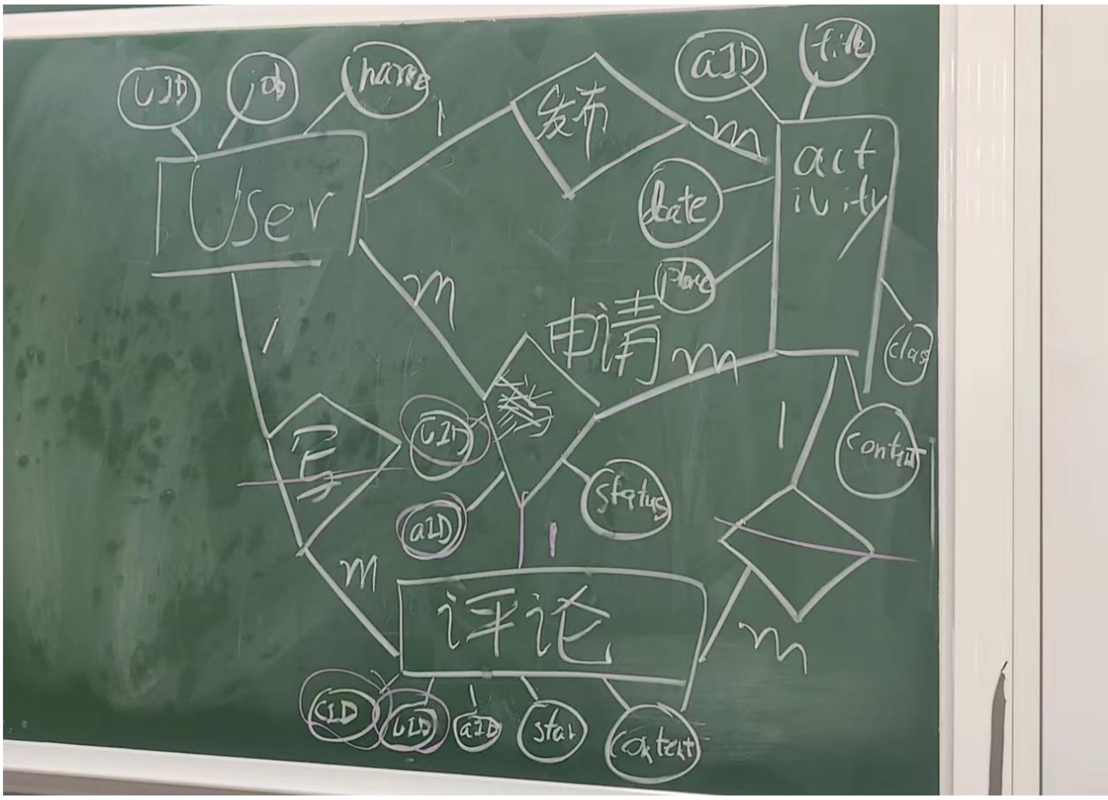
MovieID (FK)
评价 (Reviews)
UserID (FK)
MovieID (FK) / PersonID (FK)
Score
Comment
在实际数据库设计中，您可能需要考虑标准化规则，以减少数据冗余和保持数据一致性。您还可能需要创建联结表（junction tables）来有效地存储和查询多对多关系。在上述模型中，“参与”和“扮演”就是典型的联结表，它们允许存储和查询多对多关系数据。此外，设计中的每个关系都需要考虑一致性和完整性约束。

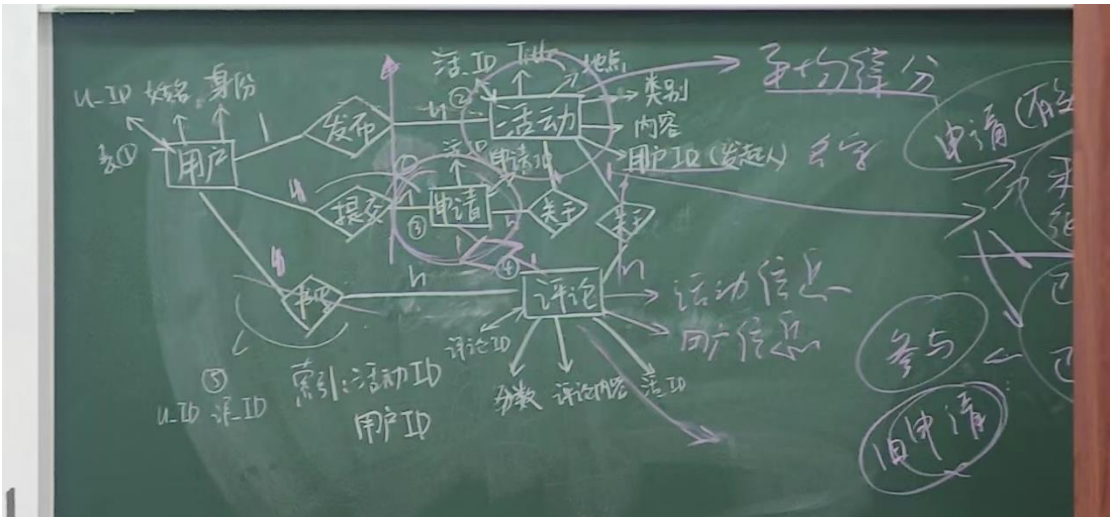


一个校园学术活动网站。所有学生和老师都可以成为网站的用户，将自己组织的学术活动在网站上发布，比如学术讲座、读书会等等。发布活动时需要指定活动的标题、时间、地点、类别和内容。一旦活动发布，其他用户就可以浏览活动的内容并报名。通常活动有人数限制，活动发起人可以选择部分报名者成为活动参与人，也可以开放给所有报名者参与。活动结束后，参与人可以给活动打分，并写评论记录活动内容。每位用户发起的所有活动将永久被网站记录，其他用户可以随时查看他发起活动的历史，以及平均得分。

请先用 ER 图为以上场景设计数据库模式，然后再根据应用访问数据库的方式对模式进行优化。（提示：需考虑索引的使用。）


用户
发布
提交
活动
申请
评论
类别
内容
活动信息
用户信息
参与


User
发布
申请
评论
aID
date
place
status
star


用户
发布
活动
申请
提交
评论
参与
申请