# 在线评测系统 - 数据库设计说明与E-R图

## 一、 数据库设计说明

本次数据库设计遵循了数据库范式，旨在减少数据冗余、保证数据一致性，并满足系统的功能和性能需求。核心设计思想如下：

1. **实体分离**: 将系统中的核心对象，如“用户”、“题目”、“赛事”等，抽象为独立的实体表。
2. **关系明确**: 使用外键和中间关联表来清晰地表达实体之间的关系（一对多、多对多）。
3. **软删除机制**: 对于核心数据，如用户注销、已结束的赛事删除等，采用“软删除”（通过一个标志位 is\_deleted 或 is\_active 来标记状态）而非物理删除。这能确保历史数据的完整性和可追溯性，例如，一个学生虽然注销了，但他在历史赛事中的提交记录和排名依然可以被查询到。

### 各表详细说明：

* **user (用户表)**
  + **用途**: 存储系统所有角色的基本信息，包括学生和教师。
  + **设计亮点**:
    - 通过 role 字段区分不同角色的用户，便于权限管理。
    - email 和 phone\_number 都设有唯一约束（UNIQUE），因为它们都可以作为登录凭据。
    - is\_active 字段用于实现用户的“软注销”，保留用户数据但使其无法登录。
* **problem (题库表)**
  + **用途**: 存储所有题目的详细信息。这是系统的核心资源之一。
  + **设计亮点**:
    - title 字段设为唯一，确保题目名称不重复。
    - test\_data\_url 字段存储的是测试数据文件的路径或链接，而不是文件本身，避免了数据库的臃肿。
    - is\_hidden 字段用于实现“软删除”。当一个题目被赛事使用后，不能物理删除，只能将其隐藏，这样不影响已存在的赛事，但在题库列表中不可见。
* **solution\_report (题解表)**
  + **用途**: 存储教师发布的官方题解。
  + **关系**: 与 problem 表是 **一对多** 关系（一个题目可以有多篇题解）。通过 problem\_id 关联。
* **contest (赛事表)**
  + **用途**: 定义一场赛事的基本信息，如时间、名称、类型等。
  + **设计亮点**:
    - password 字段用于区分公开赛和私密赛，NULL 值表示公开。
    - is\_deleted 字段用于软删除已结束的赛事。
* **contest\_problem (赛事-题目关联表)**
  + **用途**: 这是典型的 **多对多** 关系的中间表，用于连接 contest 和 problem。
  + **关系**: 一个赛事可以包含多道题目，一道题目也可以被用于多场赛事。
* **submission (提交记录表)**
  + **用途**: 记录用户的每一次代码提交，是系统中数据量最大、最核心的表。
  + **关系**: 它通过 user\_id, contest\_id, problem\_id 将用户、赛事、题目三大核心实体关联起来，形成了完整的评测记录。
  + **性能考量**: 该表的查询会非常频繁（如查询排名、个人提交历史），未来需要对 contest\_id 和 user\_id 等字段建立索引以优化性能。
* **training\_plan (训练计划表)**
  + **用途**: 定义一个训练计划的元信息。
* **training\_plan\_contest & training\_plan\_student (训练计划关联表)**
  + **用途**: 两个多对多关系的中间表。
  + training\_plan\_contest 用于描述一个训练计划包含哪些赛事。
  + training\_plan\_student 用于描述一个训练计划有哪些学生参与。

## 二、 实体关系图 (E-R Diagram)

下图使用“鸦爪表示法”（Crow's Foot Notation）展示了数据库中各个实体之间的关系。

erDiagram  
 user {  
 BIGINT user\_id PK  
 VARCHAR email UK  
 VARCHAR phone\_number UK  
 VARCHAR nickname  
 VARCHAR student\_id UK  
 VARCHAR password\_hash  
 ENUM role  
 TINYINT is\_active  
 }  
  
 problem {  
 BIGINT problem\_id PK  
 VARCHAR title UK  
 TEXT description  
 INT time\_limit  
 INT memory\_limit  
 VARCHAR test\_data\_url  
 BIGINT creator\_id FK  
 TINYINT is\_hidden  
 }  
  
 solution\_report {  
 BIGINT report\_id PK  
 BIGINT problem\_id FK  
 BIGINT creator\_id FK  
 VARCHAR title  
 TEXT content  
 }  
  
 contest {  
 BIGINT contest\_id PK  
 VARCHAR title UK  
 DATETIME start\_time  
 DATETIME end\_time  
 ENUM type  
 BIGINT creator\_id FK  
 TINYINT is\_deleted  
 }  
  
 submission {  
 BIGINT submission\_id PK  
 BIGINT contest\_id FK  
 BIGINT problem\_id FK  
 BIGINT user\_id FK  
 TEXT code  
 ENUM result  
 INT time\_used  
 INT memory\_used  
 }  
  
 training\_plan {  
 BIGINT plan\_id PK  
 VARCHAR title  
 BIGINT creator\_id FK  
 TINYINT is\_deleted  
 }  
  
 contest\_problem {  
 BIGINT contest\_id PK, FK  
 BIGINT problem\_id PK, FK  
 }  
  
 training\_plan\_contest {  
 BIGINT plan\_id PK, FK  
 BIGINT contest\_id PK, FK  
 }  
  
 training\_plan\_student {  
 BIGINT plan\_id PK, FK  
 BIGINT user\_id PK, FK  
 }  
  
 user ||--o{ problem : "创建"  
 user ||--o{ contest : "创建"  
 user ||--o{ training\_plan : "创建"  
 user ||--o{ solution\_report : "发布"  
 user ||--o{ submission : "提交"  
  
 problem ||--o{ solution\_report : "拥有"  
 problem ||--o{ submission : "属于"  
 contest ||--o{ submission : "拥有"  
  
 contest }o--|| contest\_problem : "包含"  
 problem }o--|| contest\_problem : "属于"  
  
 training\_plan }o--|| training\_plan\_contest : "安排"  
 contest }o--|| training\_plan\_contest : "属于"  
  
 training\_plan }o--|| training\_plan\_student : "包含"  
 user }o--|| training\_plan\_student : "参与"

### 图例说明：

* PK: 主键 (Primary Key)
* FK: 外键 (Foreign Key)
* UK: 唯一键 (Unique Key)
* | : 表示“一”
* o : 表示“零”
* } : 表示“多”
* ||--o{ : 表示“一对多”关系 (一边是一，多边是零或多)
* }o--o{ : 表示“多对多”关系 (通过中间表实现)