数据库 PJ1 报告

姓名：马逸君

学号：17300180070

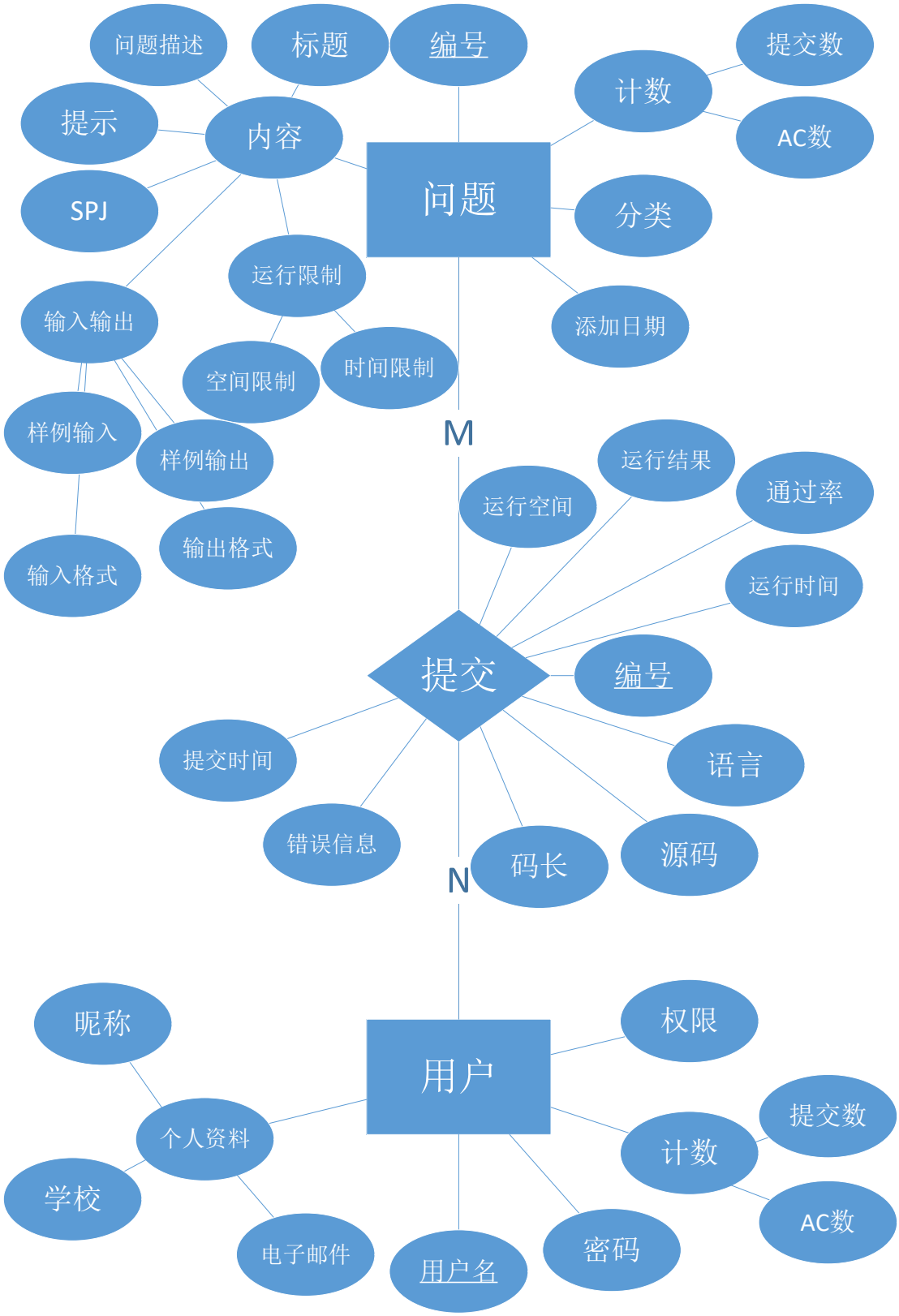
项目简介及技术架构

本次 PJ 的内容是一个 OJ（Online Judge），OJ 是一个刷编程题的网站，亦即，提供若干编程题目，注册用户可以提交自己的代码，评测核心会将设定好的测试数据输入给提交的代码，检查其输出是否与正确输出相符，也会检查时间、空间是否超限，代码是否有危险行为，等等，若答案正确且这些方面均无问题，则判断为正确。

项目采用的技术栈是 WAMP。本 OJ 分为三个部分：网页前端、数据库后端、评测核心。网页前端使用 php 语言；数据库后端由我自己独立设计实现，使用 MySQL；部署使用 Apache2；评测核心是用 C++写成的，它是一套相对独立的系统，功能是完成对用户提交的代码的评测工作（具体工作原理在数据库设计部分说明）。因为前端和评测核心超出了本课程的内容，因此本 OJ 的前端套用了 HUSTOJ 中的网页模板，评测核心来自 github 上的公开项目。

数据库设计

ER 图:



注：图中的复合属性仅为体现设计思路，实际编程时只用到了最底层的简单属性。

表结构:

problem		问题表	
字段名	类型	是否允许为空	备注
problem_id	int	N	主键；AUTO_INCREMENT
title	varchar(100)	N	
description	text	Y	
input	mediumtext	Y	输入格式
output	mediumtext	Y	输出格式
sample_input	mediumtext	Y	
sample_output	mediumtext	Y	
spj	Boolean	N	是否使用 Special Judge
hint	mediumtext	Y	
category	varchar(30)	Y	DEFAULT NULL
in_date	datetime	Y	问题添加日期；DEFAULT NULL
time_limit	int	N	限时（秒）
memory_limit	int	N	空间限制（MB）
ac_count	int	Y	DEFAULT 0
submit_count	int	Y	DEFAULT 0

submission		提交记录、提交源码、程序运行结果记录	
字段名	类型	是否允许为空	备注
submission_id	int	N	主键
problem_id	int	N	
username	char(20)	N	
language	tinyint	N	
code_len	int	N	
run_time	int	N	运行用时（ms）；DEFAULT 0
run_memory	int	N	所用空间（KB）；DEFAULT 0
result	tinyint	N	
pass_rate	float	N	通过率（通过的测试数据占测试数据总数的比率）；DEFAULT 0
submit_time	datetime	N	
source	text	N	源代码
err_info	text	Y	错误信息（包括编译错误、运行错误）

user		用户表	
字段名	类型	是否允许为空	备注
username	char(20)	N	主键
password	varchar(32)	Y	密码（加密）；DEFAULT NULL
privilege	boolean	Y	DEFAULT 0
submit_count	int	Y	DEFAULT 0
ac_count	int	Y	DEFAULT 0
nick	varchar(100)	Y	昵称；DEFAULT NULL
school	varchar(50)	Y	DEFAULT NULL
email	varchar(50)	Y	DEFAULT NULL

建表 SQL 代码：

```
CREATE TABLE problem (
  problem_id int NOT NULL AUTO_INCREMENT,
  title varchar(100) NOT NULL,
  description text,
  input mediumtext,
  output mediumtext,
  sample_input mediumtext,
  sample_output mediumtext,
  spj boolean NOT NULL DEFAULT 0,
  hint mediumtext,
  category varchar(30) DEFAULT NULL,
  in_date datetime DEFAULT NULL,
  time_limit int NOT NULL,
  memory_limit int NOT NULL,
  ac_count int DEFAULT NULL,
  submit_count int DEFAULT NULL,
  PRIMARY KEY (problem_id)
);
CREATE TABLE user (
  username char(20) NOT NULL,
  password varchar(32) DEFAULT NULL,
  privilege boolean DEFAULT 0,
  submit_count int DEFAULT 0,
  ac_count int DEFAULT 0,
  nick varchar(100) DEFAULT NULL,
  school varchar(50) DEFAULT NULL,
```

```
  email varchar(50) DEFAULT NULL,
  PRIMARY KEY (username)
);
CREATE TABLE submission (
  submission_id int NOT NULL,
  problem_id int NOT NULL,
  username char(20) NOT NULL,
  language tinyint NOT NULL,
  code_len int NOT NULL,
  run_time int NOT NULL DEFAULT 0,
  run_memory int NOT NULL DEFAULT 0,
  result tinyint NOT NULL,
  pass_rate float NOT NULL DEFAULT 0,
  submit_time datetime NOT NULL,
  source text NOT NULL,
  err_info text,
  PRIMARY KEY (submission_id),
  FOREIGN KEY (problem_id) REFERENCES problem(problem_id),
  FOREIGN KEY (username) REFERENCES user(username)
);
```

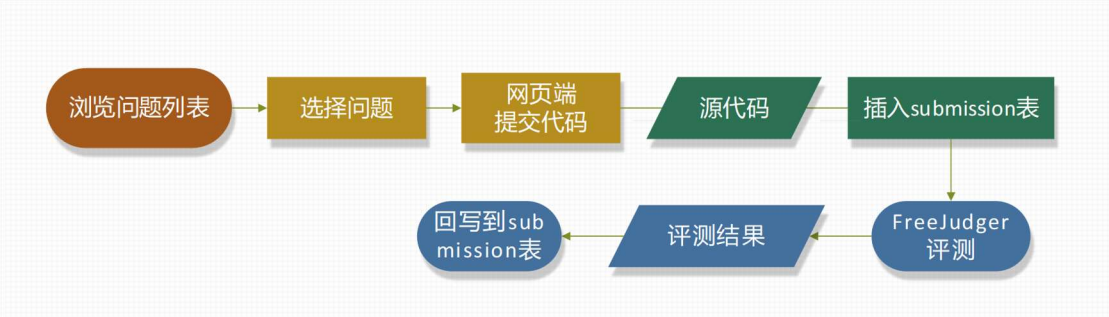
评测核心(FreeJudger)与后端的连接：

FreeJudger 轮询 submission 表，若发现待评测的提交记录，则评测之，然后更新 submission 表的 result、run_time、run_memory、err_info 等字段；若没有待评测的提交，则等待数秒后再次查询。

使用场景及实现细节

接下来，我们通过几个有代表性的使用场景，来展现实现细节（SQL）。

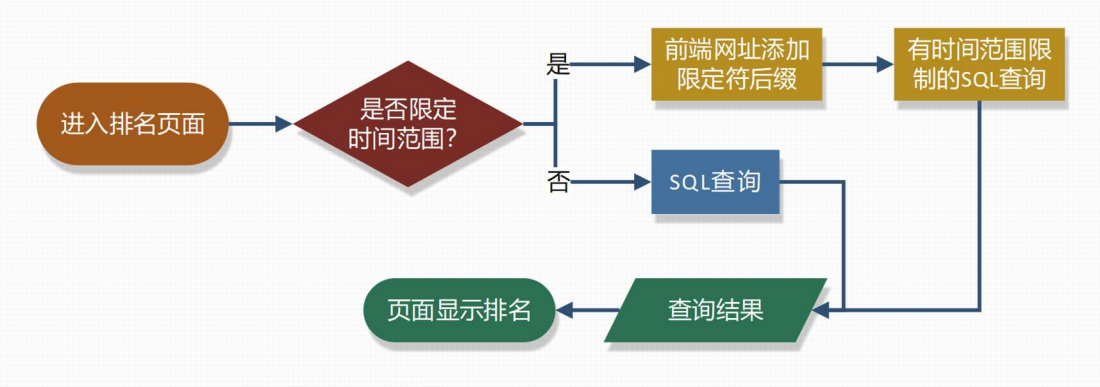
提交代码：



提交代码是 OJ 最基本的使用场景。其中最重要的一步是源代码插入后端数据库。

```
insert into submission(submission_id, problem_id, username, submit_time, language, source, code_len, result) values($sid, $id, $user_id, NOW(), $language, $source, $len, 0)
```

查看排名：

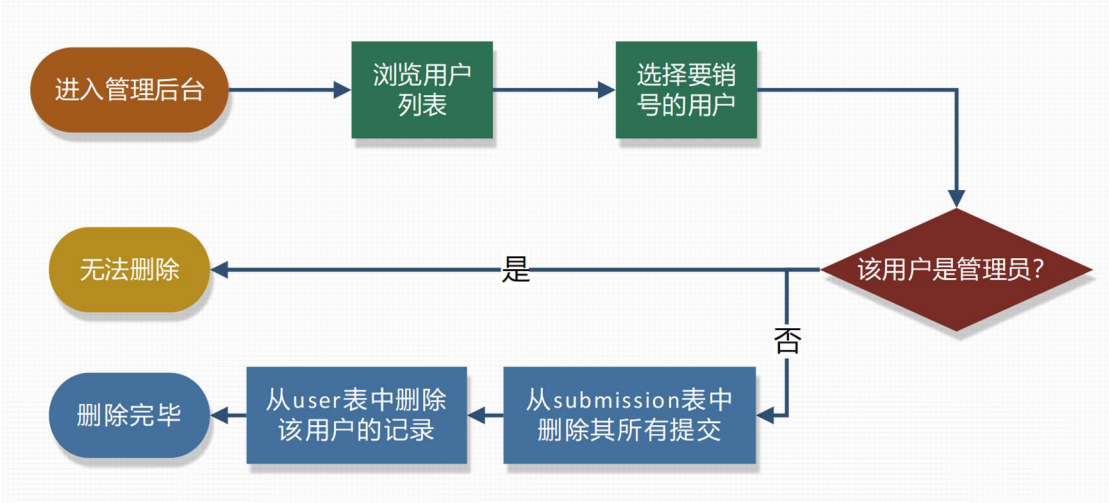


OJ 的“排名”是按照所有的或指定时间范围（如有）内各个用户的 AC 题数给所有用户排序获得的，若有两个用户 AC 题目数相同，则提交次数较少（即 AC 率较高）的用户靠前。统计排名的工作可以直接借助 SQL 语句的 order by 来实现。因为指定时间范围内的 AC 数和提交数必须在线统计，所以我们需要通过连接来实现这一查询。

```
select user.username, nick, s.ac_count, t.submit_count
    from user inner join
        (select username, count(distinct problem_id) as ac_count
        from submission
        where submit_time > str_to_date('$s', '%Y-%m-%d') and result = 4
        group by username
        order by ac_count desc limit " . strval ( $rank ) . ", $page_size) as s
        on user.username = s.username
    inner join
        (select username, count(problem_id) as submit_count
```

```
    from submission
    where submit_time > str_to_date('$s', '%Y-%m-%d')
    group by username
    order by submit_count desc) as t
    on user.username = t.username
order by s.ac_count desc, t.submit_count
limit 0, 50;
```

（管理员用户）销号：



这个功能是我在助教的要求下新增的。因为这个操作会删除该用户的所有提交代码且不可逆，所以我设计了一个确认框：

```
<a href=# onclick='javascript:if(confirm("警告：销号同时也会删除用户的所有提交且不可逆。确定要销号吗？")) location.href="user_df_change.php?cid=<?php echo $row['username']?>&getkey=
<?php echo $_SESSION[$OJ_NAME.'_'.'getkey']?>"'><span class=red>销号</span></a>
```

之后的删除操作相对简单，通过两条 SQL 语句实现：

```
delete from submission where username = $cid;
delete from user where username = $cid;
```

技术难点及开发过程花絮

当我第一次面对一屏 php 文件的时候，我的内心是崩溃的。最开始因为我不明白这个工程的 php 代码结构，我不知从何下手，在 Dreamweaver 中打开的页面或是提示“包含动态内容”，或是提示“未声明的变量”。后来我在 bilibili 上看了一点简单的 php 教程，知道了 php 中以$开头的是变量名、require_once 的作用类似 include 等，这样总算是对 php 没那么恐惧了。这时候我突然明白，我一直没弄清楚一个问题：**一个完整的 php 工程，代码应该从哪里开始阅读？**显而易见，应该从最先访问到的页面开始，就像一个 C++项目应

该从 main.cpp 开始一样。于是我点开了 index.php，自此总算是慢慢理清了 php 代码的结构。

在进行 php 开发的过程中，碰到过的主要难点有两处：

第一个是，当我改掉了原来前端里的数据库名称、数据库登录密码，看似配置好了 php 与 MySQL 的连接之后，打开页面却显示 **HTTP 500**。当时我是有点害怕的，毕竟网页开发是一个我以前从未涉足的领域。后来我开始 **“面向百度编程”**，先搜索怎样让 php 服务器显示出详细错误信息，然后再搜索这样的一个错误怎么解决。最后终于定位了问题：新版 MySQL Server 采用的密码验证方式与老版本不一样了，如需使用传统验证方式登录必须在 MySQL 命令行界面手动设定。至此问题解决。

第二个是，**前后端变量名不一致**。出于先易后难的思想，我先设计好了后端数据库的结构，然后才开始向 php 模板写 SQL 语句。但是网页模板里已经给一些数据项指定好了字段名，和我设计的数据库结构的字段名不一样，这样就带来了诸多问题：最开始是我发现网页上的一些数据加载不出来，显示为空，后来我仔细读了源码，把涉及到 SQL 查询结果的语句逐个改掉，才算是解决问题。这个问题并不复杂，但却给开发过程平添了几分麻烦，我也算是在前后端连接这样一个问题上增长了一点经验吧。

调试评测核心是本工程最令我崩溃的一项内容了。因为 HUSTOJ 提供的原版评测核心 judged 只能在 Linux 上运行，所以我不得不另辟蹊径，先后尝试了三种办法：第一种是在 Win10 内置 Ubuntu 子系统上运行 judged，第二种是寻找其他能在 Windows 上运行的评测核心（后来找到了能用的 FreeJudger），第三种是装了个 Ubuntu 虚拟机并重新部署整个项目。这其中经历了太多辛酸。

第一种方法中，我尝试了对 judged 进行**输出调试、gdb 调试**等多种办法。

第二种方法中，我**没有弄清楚 FreeJudger 的文件结构**（当然，部分原因是这个项目的 readme 没写清楚），结果事倍功半。最开始的时候 FreeJudger 总是报错，说数据库连接失败，用户名或密码错误。我用 Visual Studio 在整个解决方案里找"db_name"这样一个关键词，最后才发现数据库名称和密码这些参数根本不是写在代码里而是写在 ini 文件里。这样的问题真是令我哭笑不得。

第三种方法，在 Ubuntu 系统上重新部署的时候，我再次面临 MySQL 密码验证方式的问题。但这次没有前一次那么简单，我弄了半天就是没法让它正常验证。中间有一次还因为操作失当，把 root 密码改错了（直接把密码数据库里的密文设成了想设的登录密码），完全无法登录，重置密码又花掉了我两个小时的时间。最后索性**设了空密码**，才算是暂时性地解决了问题。（虽然最后采用的并不是这套方案）

为了评测核心这样一个和本课程内容无关但对 OJ 而言又是不可或缺的部件，我熬了三个夜，每次都熬到两点；有一次凌晨看似终于运行正常了，睡了一觉起来又不行了。最后经过一通骚操作（我也想不起来我具体都做了些什么了），在第四个白天 FreeJudger 总算是被我调教好了。唉，不容易。

翻过了评测核心这座高山，接下来的路就是一马平川了。只出现了这样几个小问题：

出于显而易见的原因，我必须制造出几组符合各种评测结果的源代码来。在制造 TLE（Time Limit Exceeded，时间超限）的源代码时，我又遇到了障碍，**编译器的-O2 选项**会优化掉空的循环语句，导致我无法通过这种方法来制造 TLE 记录；而采用低效算法的程序，因我电脑的 CPU 与这些题目的原始评测机相比性能太好，导致慢的程序也能跑得很快，于是我不得不**降低题目时限**，才算是解决了问题。

另外还有这样一个细节问题，一开始我的 user 表是没有 privilege 这个字段的，在展示之前我临时决定增加这一项。因为我的 SQL 语句全部都是写成这种形式：

```
insert into user values($username, $password, 0, 0, $nick, $school, $email);
```

增加 privilege 字段以后，表的结构变了，这个语句开始插入失败，导致新用户无法注册等一系列问题。我不得不把 php 里每一处这样的插入语句都改掉，才解决问题。我想，如果我写成下面这种形式：

```
insert into user(username, password, nick, school, email) values($username, $password, $nick, $school, $email);
```

就不会出现这样的问题了。这样的法则也适用于其他的开发场景，我们需要通过类似于这样的技巧，**提高我们程序的可移植性。**