SQL交互式在线学习系统设计

项目介绍

项目目的

本项目的目的是设计一个Web端交互式SQL学习系统，可供数据库课程实验及上课练习使用。项目包括完整的账户机制，SQL学习与评测系统以及课堂练习功能。

项目特点

项目的核心功能基于JS数据库AlaSQL实现。利用其高效快速的内存内SQL数据处理能力，我们实现了实时响应式的SQL语句评测与错误提示功能，用户在学习过程中能及时地得到反馈。因为所有SQL执行和评测均在前端执行，天然实现了用户间隔离，且无需与后台数据库进行大规模数据交互，大大减轻了后端的存储与处理压力。

> AlaSQL是一个专为浏览器及Node.js设计的轻量级客户端内存数据库，由纯JavaScript实现，无需后端数据库或WebSQL支持，数据库的存储及操作均在内存完成，注重查询速度和数据源灵活性。

项目分工

我们的小组成员有齐划一，杜洪超，王文嵩及张云飞。其中齐划一负责后端实现，主要包括账户机制和题库数据库设计；杜洪超和王文嵩负责前端实现，杜洪超负责SQL学习和评测系统的逻辑实现和部分页面设计，王文嵩负责页面设计和题库数据导入；张云飞提供了前端数据库的实现思路和调研。

功能展示

账户系统

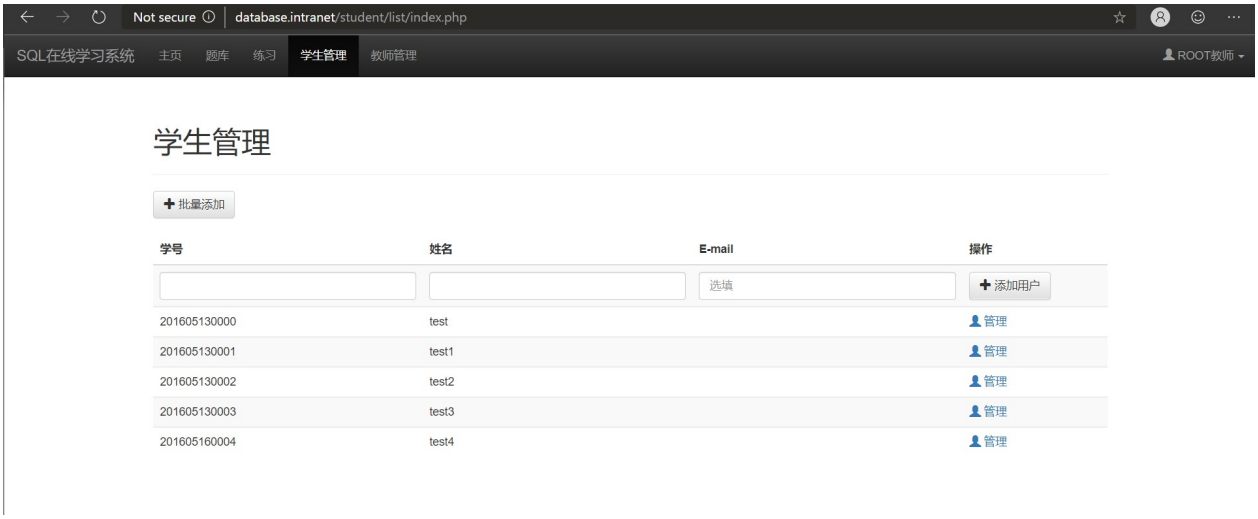
- 登陆界面

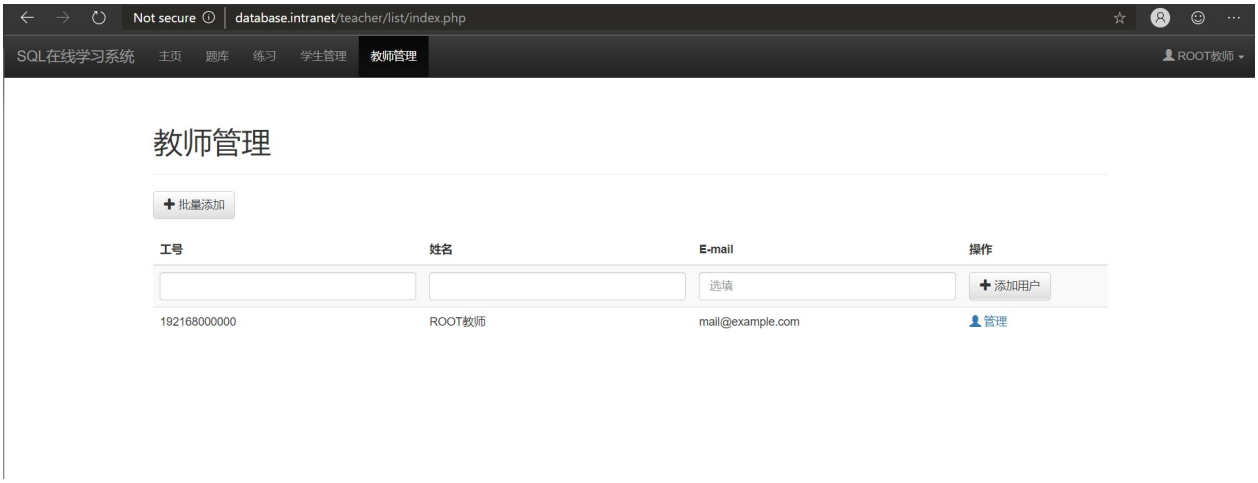
- 账户类型包括游客，学生，教师等不同类型，依据权限的高低，可访问的内容不同。具有管理员权限的用户界面如下：

可以使用题库、练习、学生管理和教师管理功能。不具有管理员权限的账户仅有题库和练习或更少的功能。

- 学生和教师管理

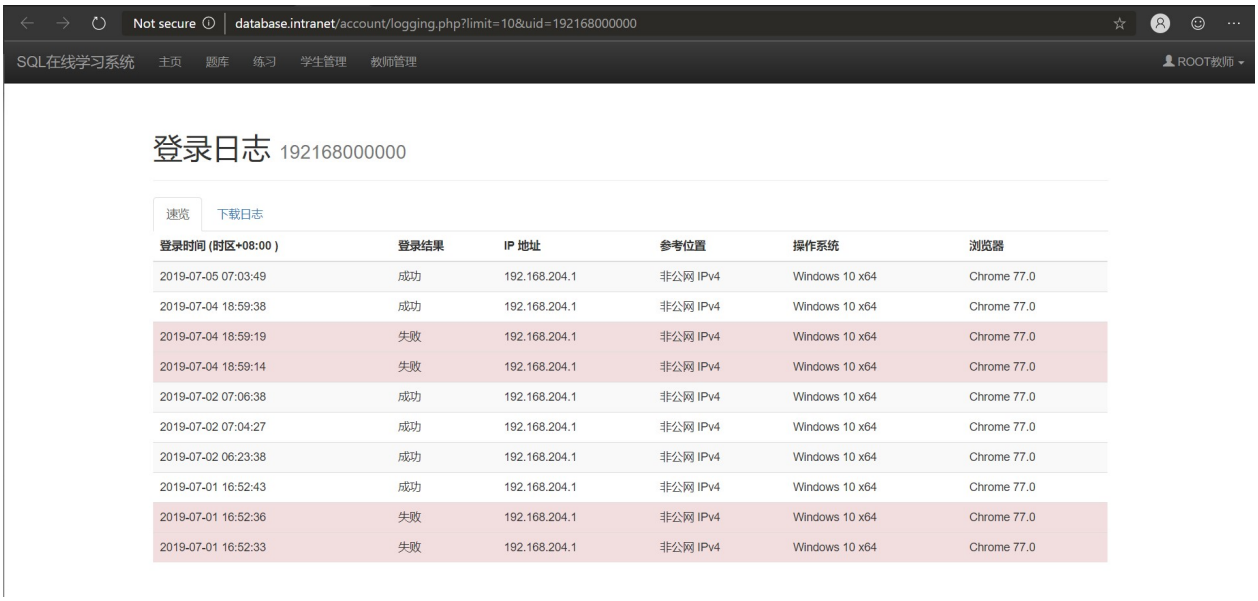
学生和教师管理包括添加单个账户、账户管理和批量添加功能。

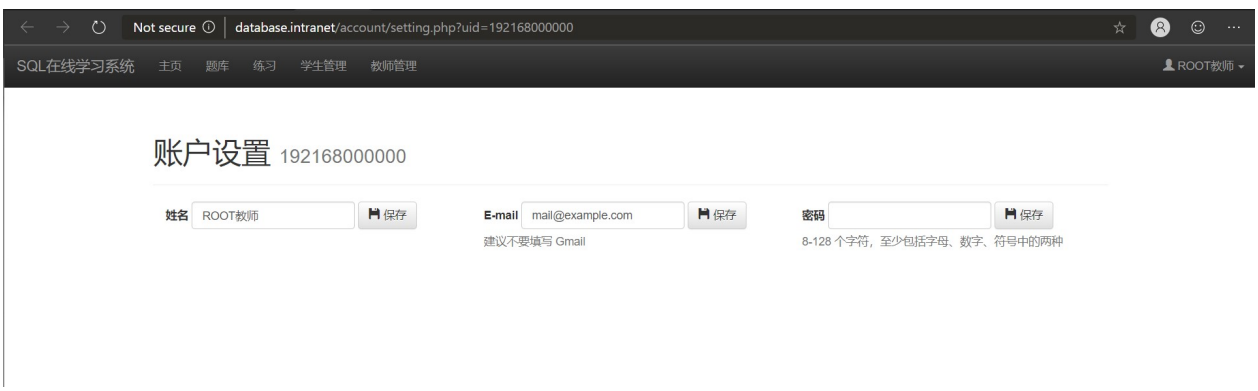


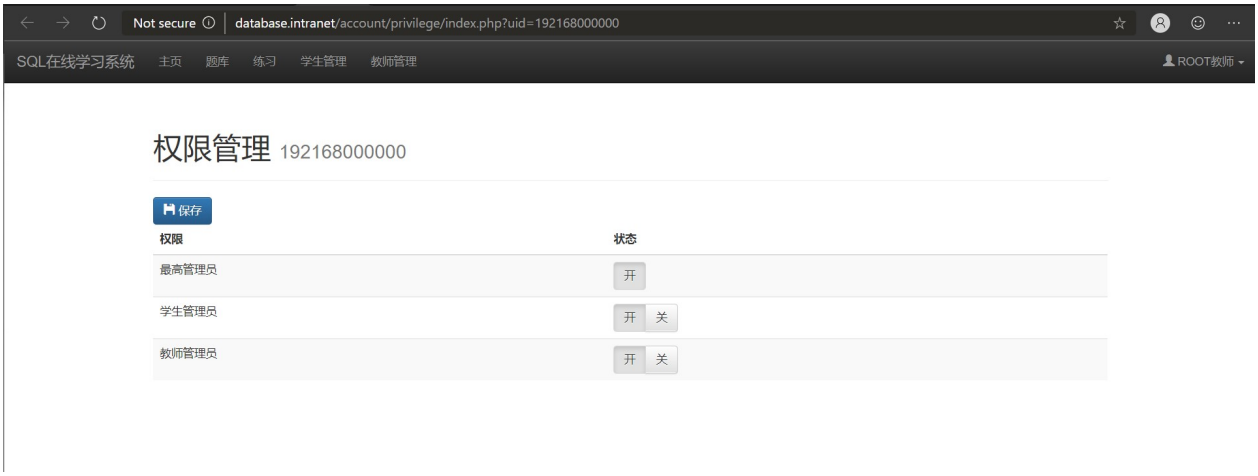


- 账户管理

账户管理支持查看登录日志、重置密码、修改姓名或邮箱、封禁用户和权限管理功能。

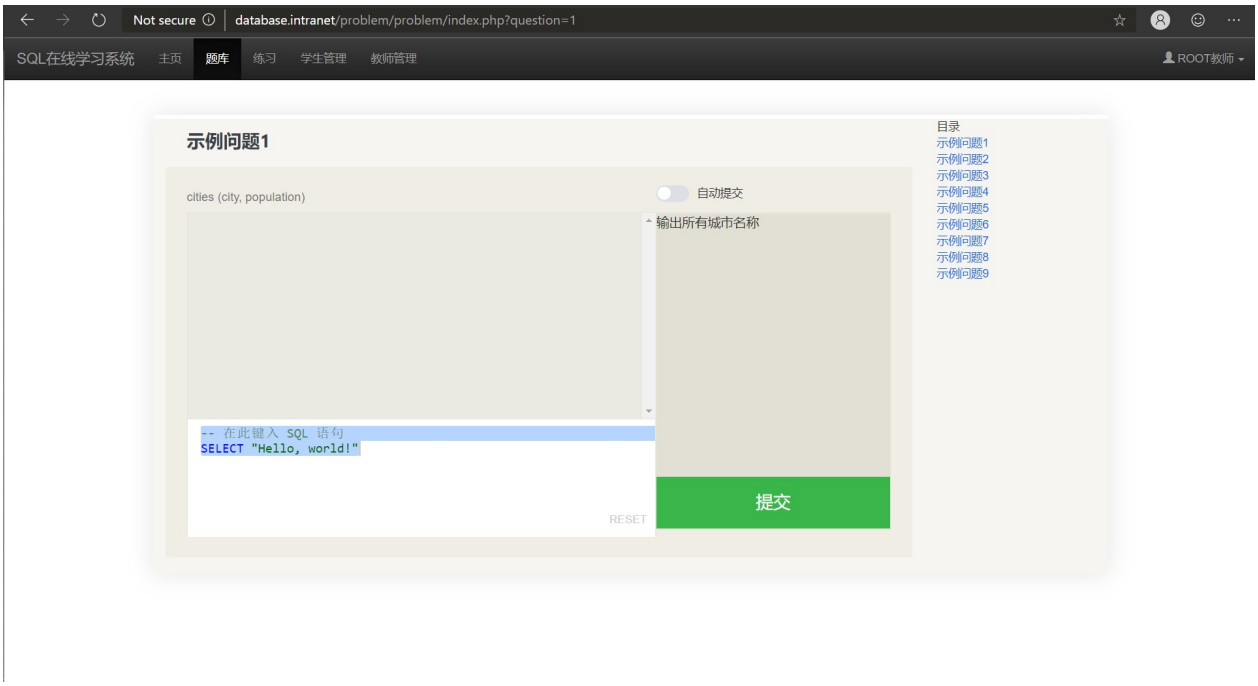




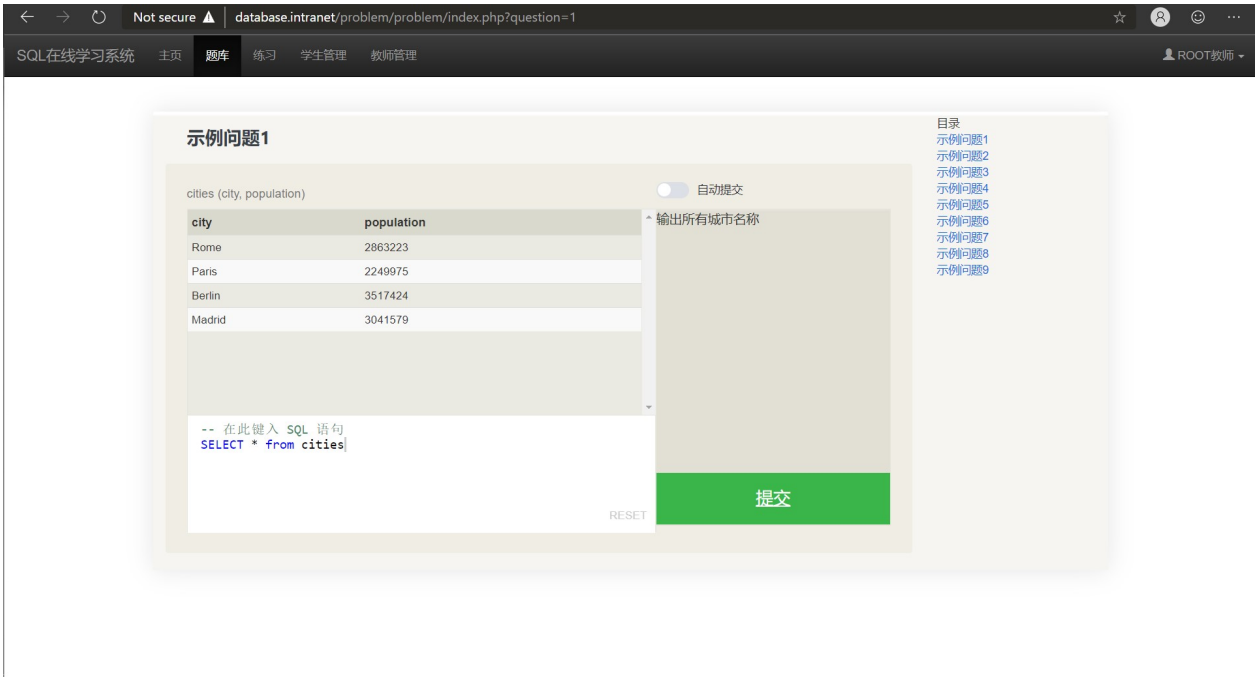


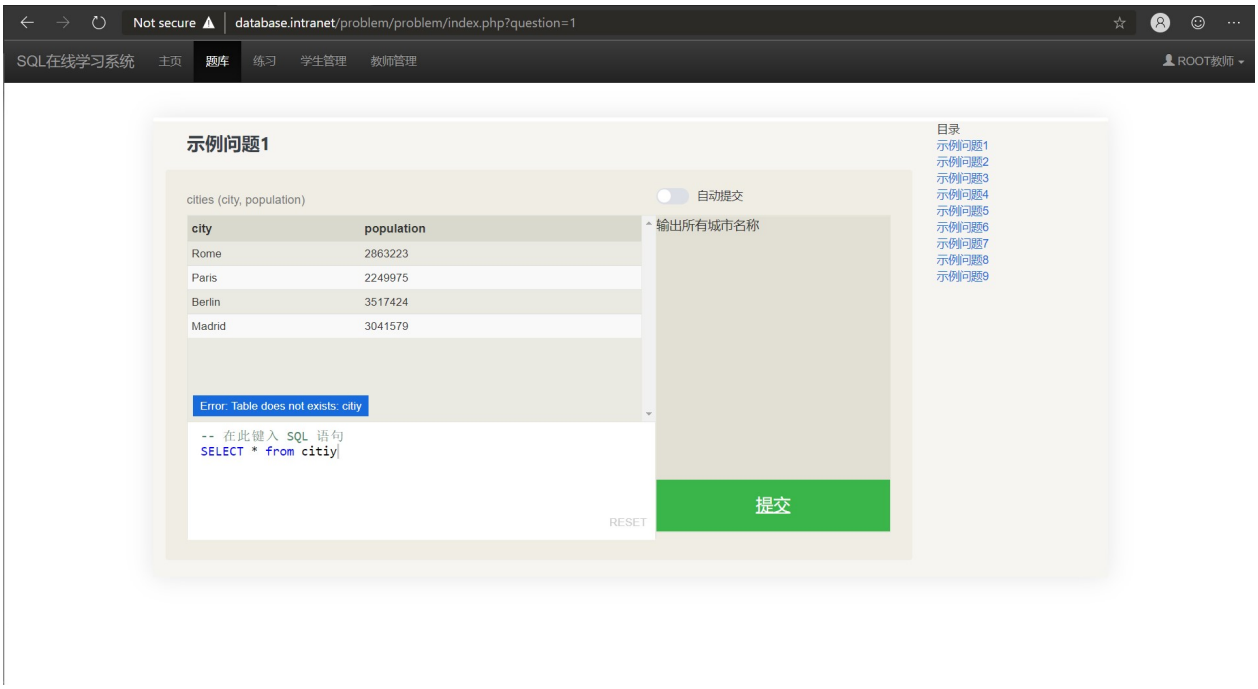
题库展示

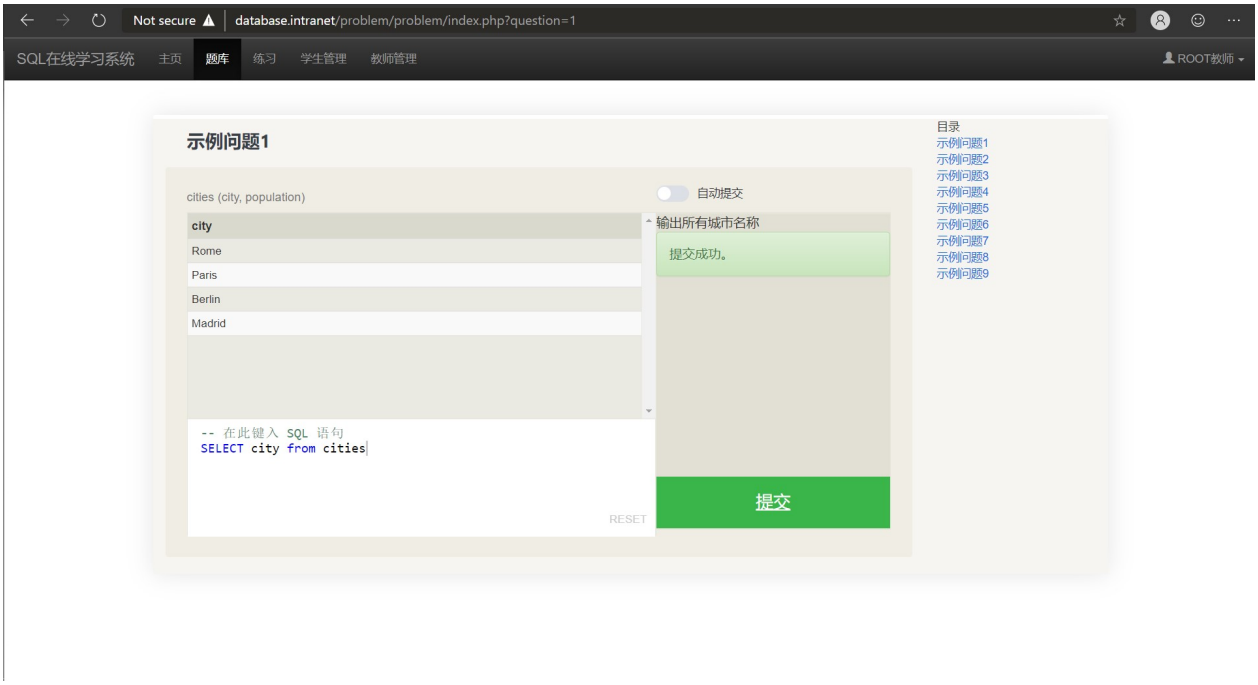
- 进入题库后，右侧为目录栏，左侧包括表的描述，结果显示，输入框，题目描述及提交、重置和自动提交按钮：



- 在输入框中输入SQL语句，点击提交按钮，显示执行结果，如果SQL执行出错提示错误信息，如果执行结果与答案相符自动将结果提交到后台，重置按钮可重新做题：

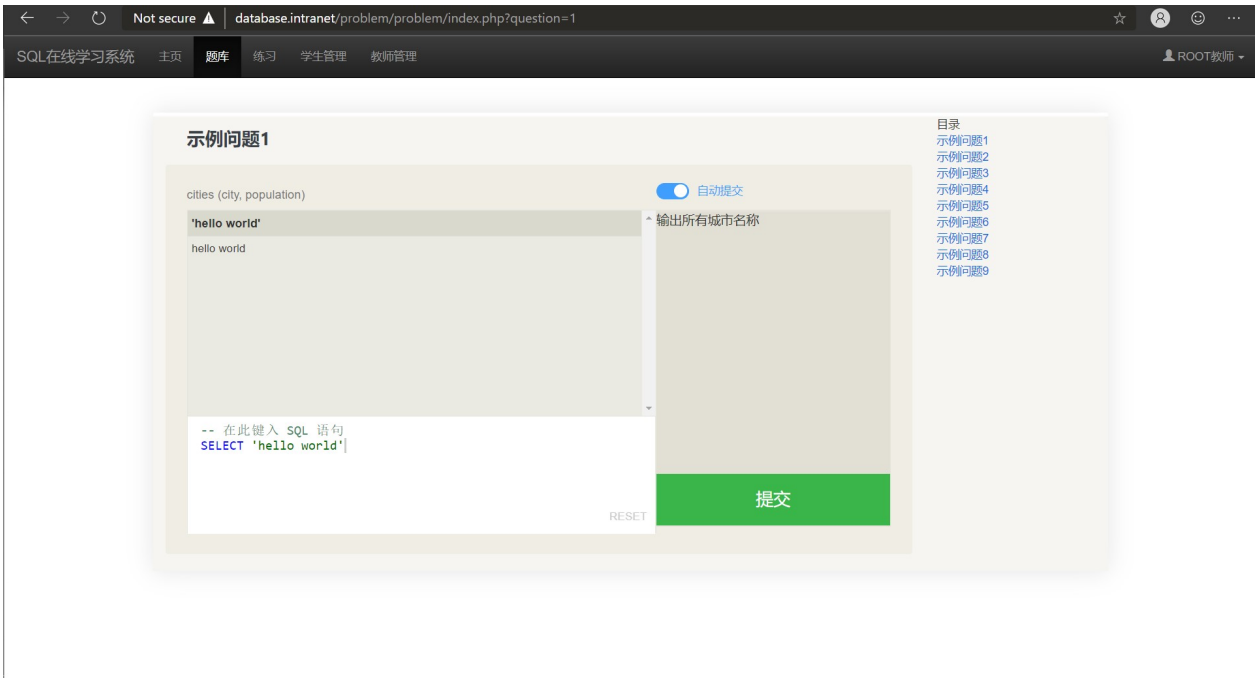

Not secure
database.intranet/problem/problem/index.php?question=1
SQL在线学习系统
主页
题库
练习
学生管理
教师管理
ROOT教师
示例问题1
cities (city, population)
自动提交
city
population
Rome
2863223
Paris
2249975
Berlin
3517424
Madrid
3041579
输出所有城市名称
-- 在此键入 SQL 语句
SELECT * from cities
RESET
提交
目录
示例问题1
示例问题2
示例问题3
示例问题4
示例问题5
示例问题6
示例问题7
示例问题8
示例问题9

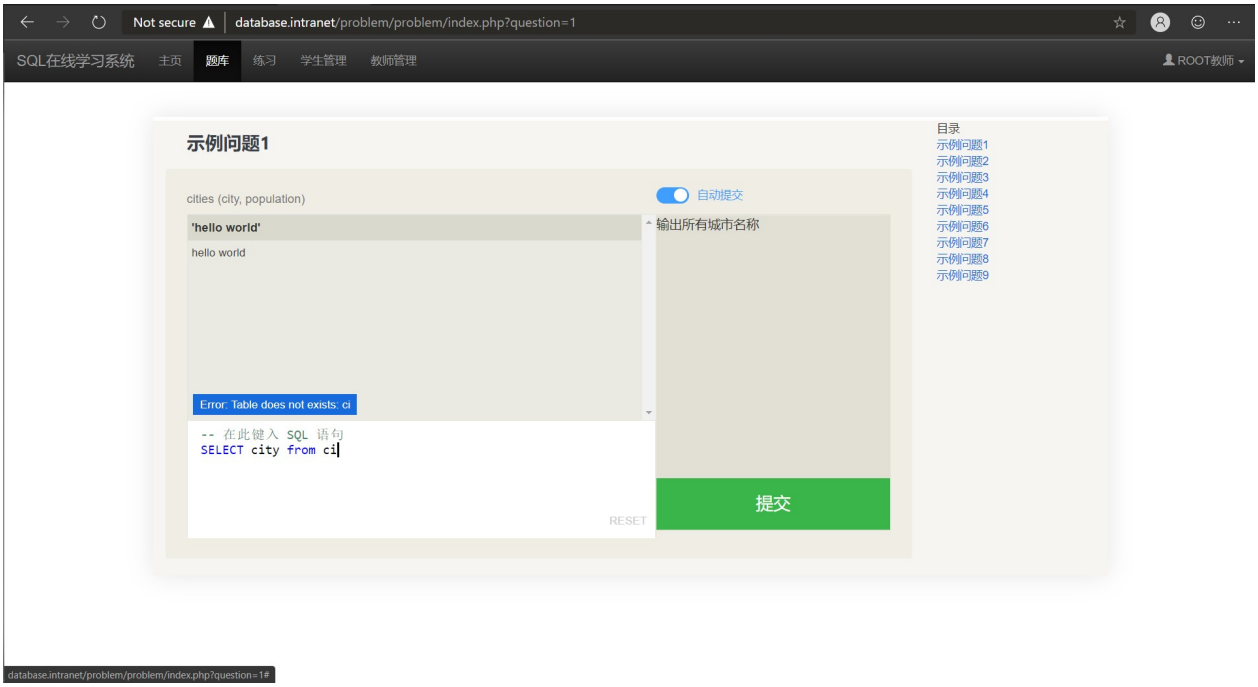

Not secure
database.intranet/problem/problem/index.php?question=1
SQL在线学习系统
主页
题库
练习
学生管理
教师管理
ROOT教师
示例问题1
cities (city, population)
自动提交
city
population
Rome
2863223
Paris
2249975
Berlin
3517424
Madrid
3041579
输出所有城市名称
Error: Table does not exists: citiy
-- 在此键入 SQL 语句
SELECT * from citiy
RESET
提交
目录
示例问题1
示例问题2
示例问题3
示例问题4
示例问题5
示例问题6
示例问题7
示例问题8
示例问题9



- 自动提交

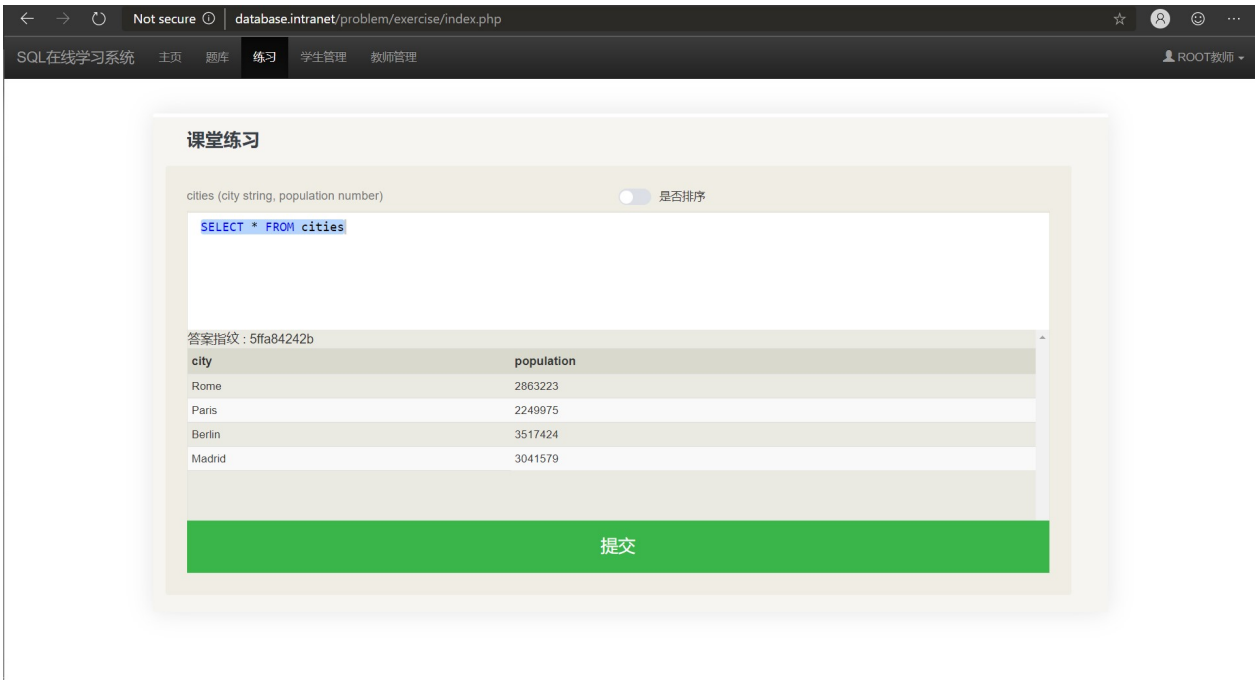
开启自动提交后，随着输入语句的变化后进行实时结果显示和评测，并具有实时监测语句错误功能，可提高SQL语句书写效率：



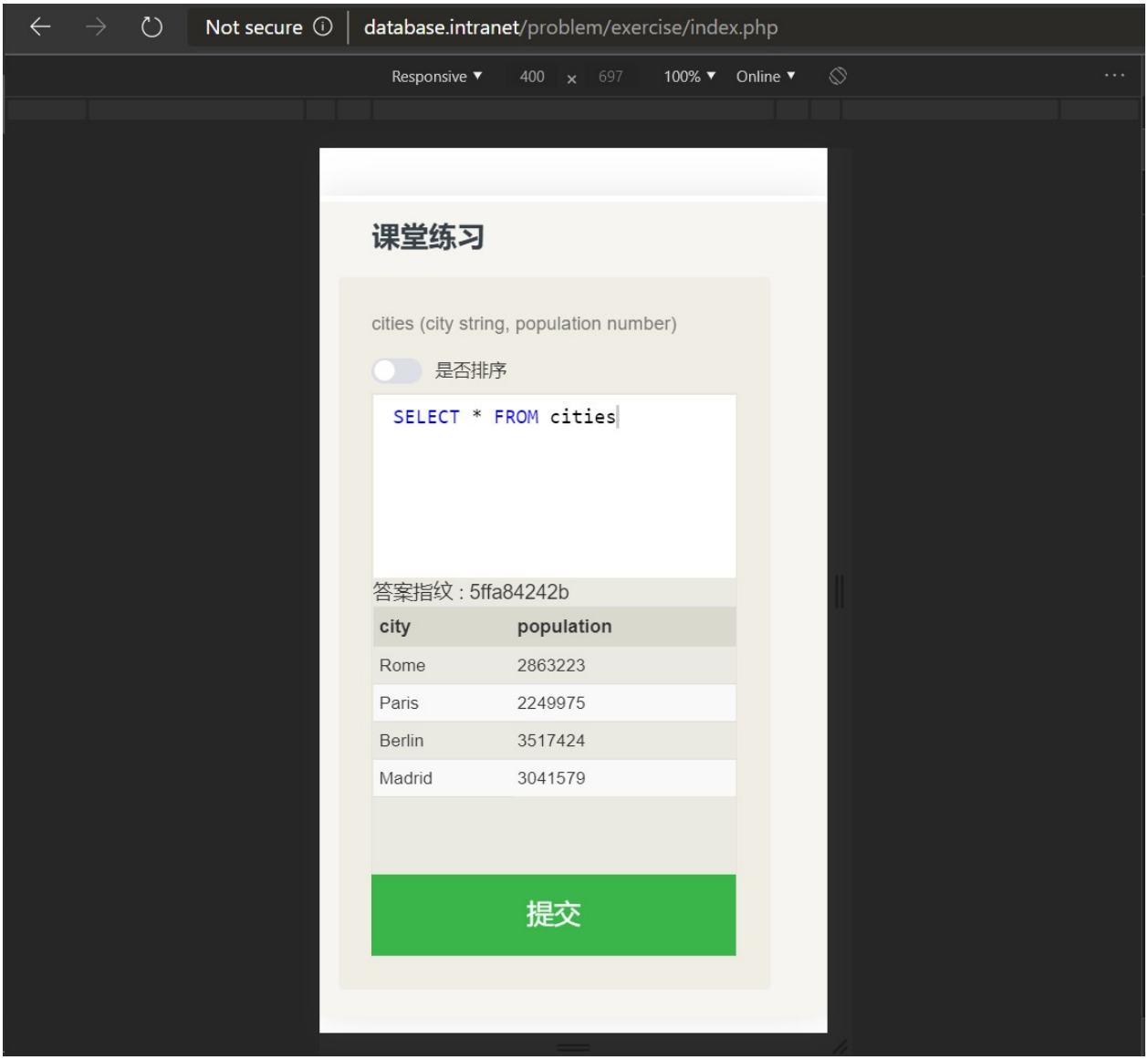


练习展示

- 练习界面主要针对课堂练习设计，相比题库减少了许多组件，添加了答案指纹和排序按钮；答案指纹随结果一同生成，可用于监测结果是否正确；排序选项用于区分需要有序和无需有序的题目，主要用于改变生成答案指纹的方式：

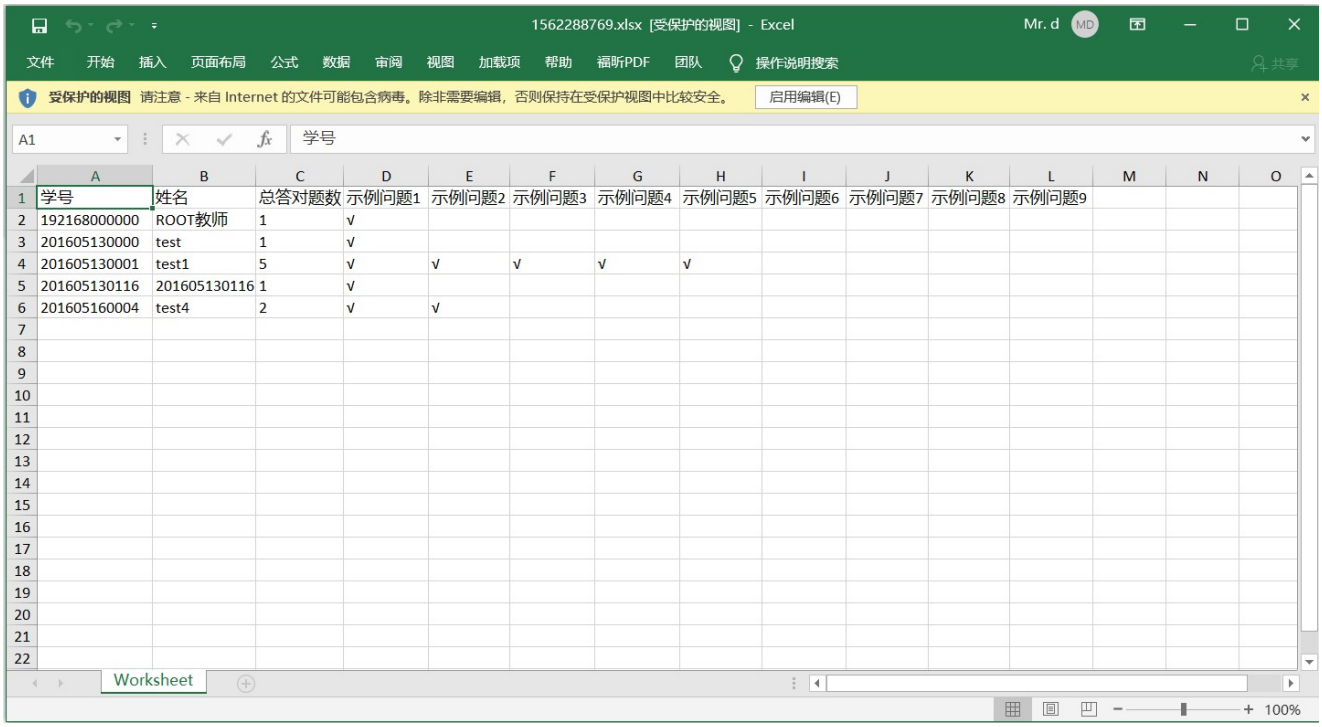


- 考虑到课堂上移动端的使用，页面针对移动端做了专门设计：



学生排名

学生可以点击主页的学生排名按钮，获得所有学生账户题库的进度信息：



1562288769.xlsx [受保护的视图] - Excel

受保护的视图 请注意 - 来自 Internet 的文件可能包含病毒。除非需要编辑，否则保持在受保护视图中比较安全。 启用编辑(E)

A1 学号

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	学号	姓名	总答对题数	示例问题1	示例问题2	示例问题3	示例问题4	示例问题5	示例问题6	示例问题7	示例问题8	示例问题9			
2	192168000000	ROOT教师	1	√											
3	201605130000	test	1	√											
4	201605130001	test1	5	√	√	√	√	√							
5	201605130116	201605130116	1	√											
6	201605160004	test4	2	√	√										
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															

Worksheet

100%

前端实现

前端整体基于Bootstrap和Vue实现，整体界面如上一节所示。因为我们使用的基于内存的前端数据库AlaSQL,因此前端还包括SQL学习系统的逻辑实现，主要包括页面数据的获取与展示，SQL语句的执行，评测与报错机制。

数据及含义

实现中用到的主要数据如下：

数据	含义	初始值
sql	要执行的sql语句	''
sql_init	初始sql语句	''
table_init	初始化表环境的sql语句	''
is_error	是否出错	false
error	错误信息	''
auto	是否自动提交	false
contents	目录	''
problem	题目类型	''
task	题目描述	''
table	初始环境中表的描述	[]
answer_hash	执行结果的hash值	''
columnList	执行结果的列名信息	[]
inforList	执行结果的行信息	[]
editor	编辑器实例	null

在页面加载时从后台获取这些数据的值，利用Vue的绑定特性，把这些数据和前端页面的元素绑定，就能方便地实现页面内容渲染。

函数及含义

- result; Vue中的计算属性，用于实现自动提交功能。和页面中编辑器内容绑定，每当编辑器内容发生变化就自动执行此函数，函数内部调用提交函数；
- init_table(); 初始化表格环境；先格式化旧环境，再执行table_init创建新环境；
- init_editor(); 初始化编辑器；使用ace实现的代码编辑器，设置为SQL语言模式，支持代码高亮；
- init(); 页面加载时执行此函数，调用init_table()和init_editor(),重置页面信息；
- Submit(); 提交函数，利用sql变量绑定编辑器内容，执行sql语句，捕获错误信息，如果出错设置error和is_error,error内容与页面错误提示绑定，实现实时显示错误信息；如果未出错则利用返回内容更新inforList和columnList,渲染与之绑定的结果元素；如果结果hash值与从后端获得的答案hash值相同，向后台提交成绩；

- get_hash(); 根据返回结果计算hash值；首先调用get_string()将object转化为字符串，再对字符串进行字符串hash；
- get_string(); 对对象进行递归处理，序列化为字符串。

数据导入

后端实现

本项目使用了 PHP 7.3 和 MariaDB 10.0 作为后端。

PHP 在开发小型项目上的效率较高，入手容易，支持面向对象的开发模式；PHP 7 相比 5 的性能提升有一倍之余，且在保证兼容性的同时，逐步抛弃老版本的过时设计和不安全的组件。

MariaDB 10.0 是 MySQL 5.6 的等价替代品。由于 Oracle 的许可证限制和潜在的风险，许多 Linux 发行版已将 MariaDB 作为 MySQL 的默认替代品。之所以不选择更高版本的 MySQL/MariaDB，是因为高版本的数据库软件对内存的要求更高，动辄 2GB（MySQL 5.7）、5.5GB（MySQL 8）起步的内存占用虽然可以显著提升数据库效率，但对于对数据库性能要求不高的小型项目来说并无必要。

MariaDB/MySQL 对操作系统无要求，但 PHP 在 64 位 Linux 上表现最佳，因此推荐在 Debian、CentOS 等发行版上部署后端。

数据表的设计

用户表（用户ID、密码哈希、邮箱地址、姓名、是否是学生、是否被禁止登录）

用户权限表（用户ID、权限名、权限值）

用户登录日志表（用户ID、登录时间、IP 地址、User Agent、是否登录成功）

数据库问题表（问题ID、问题标题、问题描述、预加载 SQL 语句、答案指纹）

数据库问题作答表（问题ID、用户ID、作答 SQL 语句、答案指纹、是否正确、提交时间）

PHP 与 MariaDB 的交互

SQL 注入的防范

使用了 PHP 的 mysqli 组件操作数据库。该组件自带 bind 功能，用于防止 SQL 注入。例如，要想查询姓名为`李狗蛋`的用户，传统的写法如下：

```
SELECT * FROM user WHERE name = "李狗蛋"
```

如果用户输入的姓名不是`李狗蛋`，而是`";DROP TABLE user;--`，则执行的 SQL 语句变成了：

```
SELECT * FROM user WHERE name = "";DROP TABLE user;--"
```

原本的一条语句变成了两条语句和一个注释，于是 user 表被清空。

mysqli 提供的 bind 功能则不需要开发者手动进行字符串的拼接、字符转义等。同样的查询功能，在开发时写作：

```
SELECT * FROM user WHERE name = ?
```

问号处该用什么填充，如何对字符进行转义等问题，由 mysqli 已经写好的函数代劳，开发者只需调用即可，无需关心细节。这在方便了开发者的同时杜绝了 SQL 注入攻击的可能。

面向对象

开发时，用户、问题、答案均作为类，这些类有静态方法：从数据库中读入、新建、调取全部对象；有属性：每个属性对应数据库的字段；有方法：写回数据库。因此，在编写其他页面时，无需再次关注底层的数据库细节，而是对对象进行操作。这在提升开发效率的同时降低了与数据库的耦合度，便于修改。虽然也会带来一定的性能损失，但总体而言，利大于弊。这类似于 ORM，不过是手动实现的。

后端与前端的交互

为了提高开发速度，本项目只在确实需要与前端交互时进行了前后分离，其他情况由 PHP 直接渲染前端的 HTML 页面。前后分离的代码放在 api 目录下，前端将 JSON 格式的参数以及 Cookie 一起 POST 到对应的 URL 下，后端在解析 JSON 后，判断用户权限，执行，返回结果，前端再根据结果显示对应的提示文字。

其他安全考虑

数据库的密码存储

注意到，用户表中并没有“密码”字段，而是“密码哈希”。简单地说，把密码明文存储（或可逆加密）到数据库中是一件愚蠢至极的行为，任何一个合格的开发人员都不应该这样做。倘若数据库泄露，所有用户的密码均被泄露，黑客便可利用这对密码去其他网站尝试登录，从而造成严重损失。可悲的是，无论是企业、学校乃至有些大型公司，这样的常识依然不是常识，以前两者尤甚。

合格的做法是“加盐哈希”。用户表中存在“密码哈希”和“盐”两个字段，满足 `Hash(原密码+盐)=密码哈希`，其中的 `Hash()` 表示哈希函数，即单向计算很容易，逆向计算异常困难的函数，常见的有 SHA256 等；加号表示字符串拼接。黑客攻破数据库后，破解单个用户密码的成本已经不可接受，更何况每个用户的密码都要单独破解，因此该方案是合理的。

本项目并未显式使用加盐哈希，而是使用了 PHP 提供的密码 API。该密码 API 实际上实现了上述加盐哈希的过程，只是将盐、算法和密码哈希组合成了一个字符串，便于保存在数据库

中。

HTTPS

然而，加盐哈希使得服务器失去了预测客户端正确输入的能力。Digest 认证可以在不安全的信道下完成用户身份的验证，同时不会造成密码泄露，但使用加盐哈希后，服务器不再有能力支持 Digest 认证。好在 HTTPS 可以为服务器和客户端之间提供安全的信道。权衡两者利弊，显然是加盐哈希方案最重要，而且 HTTPS 已经非常流行，只需在公网有域名即可免费部署。

后端部署

为简化部署，可选择安装 CentOS 7，并安装宝塔面板。地址是 https://www.bt.cn。

以下的部署过程仍不失一般性。

注记符

`<www>` 表示网站所在目录。

配置要求

1. 必须安装 Linux。
2. 必须安装 HTTP 服务器，推荐 Nginx；如果对安全性有要求，必须配置 HTTPS，推荐配置 HTTP 2.0 及 HSTS。
3. 必须安装 64 位的 PHP 7.1 或更高版本。推荐 PHP 7.2 或更高版本。
4. 必须安装 MySQL 5.5 或更高版本，推荐 MariaDB 10.0 或更高版本。数据库的编码必须设置为 `utf8mb4`。
5. 以下文件夹必须设置为对 HTTP 服务器可读写，且必须对浏览器返回 HTTP 403：
   - PHP 函数 `sys_temp_dir()` 返回的临时目录；
   - `<www>/private` 目录。
6. 推荐配置 `Zend Opcache` 以加速。

开始部署

1. 把工作目录切换到 `<www>` 目录

   ```
   cd <www>
   ```

2. 把 Composer 的 PHP 可执行文件安装到当前目录

   ```
   curl -sS https://getcomposer.org/installer | php
   ```

   完成后，当前目录中会得到 `composer.phar` 文件。

3. 让 Composer 安装依赖组件

```
php composer.phar install
```

4. 将 `database.sql` 文件导入数据库。

   可通过 mysql 客户端的 source 命令导入 sql 文件，或者利用 PHPMyAdmin 等图形化工具进行操作。

5. 将 `<www>/config-example` 文件夹移动到 `<www>/config`，并根据实际情况修改里面的 PHP 文件。

   涉及到数据库地址、用户名、密码；域名、网址等，如未正确配置则无法登录。

6. 把工作目录切换到 `<www>/private` 目录，下载 GeoLite2 数据库，并解压。可直接复制下面这条命令完成。

   ```
   (dir="/tmp" && name="$dir/$(wget
   http://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz -O - |
   gunzip | tar xvf - -C "$dir" | grep GeoLite2-City\\.mmdb )" && [ "$(echo
   "$name" | wc -l)" = "1" ] && [ ! "$dir/" = "$name" ] && mv "$name" . && rm -r
   "$(dirname "$name")")
   ```

   检查 `GeoLite2-City.mmdb` 文件是否出现。

   如果上面的命令没有执行成功，备用的命令如下。

```
   pushd /tmp
   wget http://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz -O
- | gunzip | tar xvf -
```

找到解压得到的 `GeoLite2-City.mmdb` 文件，将其移动至 `<www>/private/GeoLite2-City.mmdb`。

7. 必须将 `<www>/private` 目录设置为 HTTP 403。此项已在“配置要求”中给出，再强调一次。

   Nginx 配置文件示例：

   ```
   location /private {
       deny all;
   }
   ```

8. 将 `<www>/install` 目录删除，或设置为 HTTP 403。

9. 可将以下文件或目录设置为 HTTP 403：

    - `<www>/composer.lock`
    - `<www>/composer.json`
    - `<www>/vendor`
    - `<www>/.*`
    - `<www>/README.md`
    - `<www>/install`
10. 可为除 `.php` 和目录外的 URL 配置缓存。
11. 如果要配置 HTTPS，推荐按照 Mozilla 推荐的配置 进行配置。
12. 管理员默认用户名 `192168000000`，默认密码 `025f41c16c808eea`。登录后务必修改默认密码。

项目总结

本项目实现了Web端的交互式SQL学习系统，主要针对实验及课堂练习使用进行了优化，注重执行速度和使用体验。