Design and Implementation of SQL Online Practice Platform

Wenhui Zhang
Western Kentucky University, zhangwenhui2009@gmail.com

Follow this and additional works at: https://digitalcommons.wku.edu/theses
Part of the Computer Engineering Commons

Recommended Citation
https://digitalcommons.wku.edu/theses/3091

This Thesis is brought to you for free and open access by TopSCHOLAR®. It has been accepted for inclusion in Masters Theses & Specialist Projects by an authorized administrator of TopSCHOLAR®. For more information, please contact topscholar@wku.edu.
DESIGN AND IMPLEMENTATION OF SQL ONLINE PRACTICE PLATFORM

A Specialist Project
Presented to
The Faculty of the School of Engineering and Applied Sciences
Western Kentucky University
Bowling Green, Kentucky

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Wenhui Zhang

May 2019
I dedicate this thesis to my American parents, Mark Jessen and Lauren Jessen, who are a great inspiration to me.
ACKNOWLEDGMENTS

I want to convey my deepest gratitude and respect for my thesis and research advisor, Dr. Guangming Xing, for his support, guidance, and patience during this research. His intelligence and encourage have made this research enjoyable and fruitful. I am very fortunate knowing and working with him. I would like to thank him for his patience and for trusting me with this research. Dr. Guangming Xing’s valuable comments helped me to develop a broader perspective of my thesis topic, and in learning beyond the textbooks. I have learned a lot from him, and his teachings have been truly inspiring. I would like to sincerely thank Dr. Qi Li and Dr. Zhonghang Xia for serving on my committee, provided valuable advice, reviewing my work, and all other efforts I have been given. I would also like to thank the faculty and staff of the Department of Computer Science.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>RELEVANT COMPUTER TECHNOLOGY AND TOOLS</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>SYSTEM ANALYSIS</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>SYSTEM DESIGN</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>SYSTEM IMPLEMENTATION</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>SYSTEM TESTING</td>
<td>47</td>
</tr>
<tr>
<td>7</td>
<td>CONCLUSION</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>BIBLIOGRAPHY</td>
<td>51</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Composition of SQL statement</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>B/S architecture</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>MVC</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Teacher use case</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>Student use case</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>System design of teacher</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>System design of student</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Function module</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>Database design</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Project explore</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>Teacher login page</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>Main page of teacher</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>Upload SQL file</td>
<td>29</td>
</tr>
<tr>
<td>14</td>
<td>Main page after upload SQL file</td>
<td>31</td>
</tr>
<tr>
<td>15</td>
<td>Add question file</td>
<td>33</td>
</tr>
<tr>
<td>16</td>
<td>Question list</td>
<td>33</td>
</tr>
<tr>
<td>17</td>
<td>Student login page</td>
<td>35</td>
</tr>
<tr>
<td>18</td>
<td>Student register page</td>
<td>35</td>
</tr>
<tr>
<td>19</td>
<td>Student main page</td>
<td>37</td>
</tr>
<tr>
<td>20</td>
<td>Loading page</td>
<td>38</td>
</tr>
<tr>
<td>21</td>
<td>Docker container in the shell</td>
<td>38</td>
</tr>
</tbody>
</table>
Figure 22 Databases information inside of the container .................................................. 39
Figure 23 Question display ................................................................................................ 42
Figure 24 Result display .................................................................................................... 43
Figure 25 Answer display .................................................................................................. 43
Figure 26 Error display ..................................................................................................... 44
LIST OF TABLES

Table 1 ER diagram of teacher ................................................................. 23
Table 2 ER diagram of student ................................................................. 23
Table 3 ER diagram of teacher_database.................................................. 23
Table 4 Testing examples - teacher ......................................................... 47
Table 5 Testing examples – student.......................................................... 48
Table 6 Testing examples - multiple users ............................................... 48
Database management is a very important subject in computer science education. The study of SQL syntax is the most important part of this course. The traditional teaching method only lets the students practice SQL syntax through the textbook exercise. Students cannot get the query result directly. Moreover, students do not know intuitively what is wrong with the query sentence. In this study, we want to find a better way for the teachers and the students to learn this part of database management. Teacher uploads a SQL file through a web page, it will create a database with several tables and some data records on the database server. The student logs in to the platform and chooses the database that he wants to practice. The student can do all regular database management operations such as query data, insert data, delete data, etc. If the query sentence is correct, the query result will be displayed on the webpage. If not, the detailed error information will show up. Since there is more than one student using the platform, so these operations cannot change the database created by the teacher. Therefore, the student’s process will run in a Docker container. In the container, there is a duplicated database of the teacher’s database. Meanwhile, the platform supports multiple users to do exercise with one database at the same time,
CHAPTER 1 INTRODUCTION

1.1. Background

Structured Query Language (SQL) is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system. It is particularly useful in handling structured data where there are relations between different entities/variables of the data. SQL is based initially upon relational algebra and tuple relational calculus, it consists many types of statements, which may be informally classed as sublanguages, commonly: a data query language (DQL), a data definition language (DDL), a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control.

The SQL language is subdivided into several language elements, including:

- Clauses, which are constituent components of statements and queries.
- Expressions, which can produce either scalar values, or tables consisting of columns and rows of data.
- Predicates, which specify conditions that can be evaluated to SQL three-valued logic (true/false/unknown) or Boolean truth values and are used to limit the effects of statements and queries, or to change program flow.
- Queries, which retrieve the data based on specific criteria.
- Statements, which may have a persistent effect on schemata and data, or may control transactions, program flow, connections, session, or diagnostics.
SQL statements also include semicolon ";" as a statement terminator, though it is not required on every platform.

Figure 1 is a chart showing several of the SQL language elements that compose a single statement.

```
UPDATE clause  {UPDATE country
    expression
}

SET clause    {SET population = population + 1
    expression
}

WHERE clause  {WHERE name = 'USA'
    predicate
}
```

Figure 1 Composition of SQL statement

Database Management is an essential course in computer science education, in which SQL syntax may be the most challenging part. A correct SQL statement is needed to get the information we want from thousands of records. When the query is related to multiple data tables, the syntax of the SQL statement becomes more complicated. The traditional teaching method focus on the teacher analyze of SQL syntax. Then students practice SQL syntax through the textbook exercise. However, the textbook is not able to display the query result to students and is not able to tell students where his query sentence is wrong. There is no feedback during the practice, so it is boring and ineffective.

1.2. **Aim and Objectives**

The paper aims to provide a highly interactive, convenient online platform for SQL practice. The teacher creates a database with exercise questions; the student can do SQL exercise with this database. The platform will display the result of a student's query;
it is easy to check whether the SQL statement is correct. Because the student can get feedback about his query, the practice process becomes more exciting and useful.

Teacher uploads a SQL file includes statements to create a database with several tables and some data records through the platform. The file generates a database on the server. When the student logs in to the platform and clicks the database name that he wants to practice, a Docker container is used for isolation as well as security. All operations the student does during exercise won’t affect his teacher’s database. Instead, an instance of the teacher-created database is created in the Docker container. Students can do all regular database management operations according to the questions; these operations will only affect the duplicated database in the Docker container. The platform supports multiple users to do exercise with one database at the same time.

1.3. Thesis Structure

The remainder of the thesis is organized as follows. Chapter 2 presents the relevant technology and tools that will be used in the project. Chapter 3 and Chapter 4 analysis the system requirement and system design. Chapter 5 is the implement of the system.
CHAPTER 2 RELEVANT COMPUTER TECHNOLOGY AND TOOLS

2.1. B/S Architecture

B/S architecture, B refers to browser and S refers to the server. It is a sufficient improvement of the C/S structure with the development of Internet technology, which is realized in the user interface through a web browser. And the vast majority of logical transactions are achieved in the server. The design reduces the load on the client computers and increases the efficiency of system maintenance and upgrades. To access a B/S based system, a user only needs a web browser on his computer, laptop, or mobile devices. Web browser exchanges data through the web server and database server.

B/S architecture software has the advantages of simple employ. With the Internet and web browser, the user can visit operate the system. No special software is needed, so there is no maintenance work for the client. And it also makes the system easy to extend. Thus, B/S architecture becomes more popular now. Moreover, with the development of AJAX, the client computer can also perform some processing, which helps to reduce the running pressure of the web server and enhance the interactive ability.
The graph shows the system framework of B/S architecture. Users access the system via an HTML webpage on the web browser. Web server, such as Apache or Tomcat, compiles the request. The web server will send all requests related to data transactions to the database server. The requested data will be returned to the web server and displayed on the web page.

### 2.2. MVC Framework

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects.
The user sends a request through a web interface such as HTML page. The controller acts as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. Controller finds the data query in the Model. When the specific data are got, returns the data to the View component. The Model component corresponds to all the data-related logic that the user works with. The data can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. The View component is used for all the UI logic of the application. It will process the data sent by the Controller and display them on the user interface.

MVC makes it easy to manipulate complicated projects and applications, reduce the development complexity, and improve development efficiency. Because of the separation of Model, view, controller, MVC achieves the separation of the business logic layer and the persistence layer. Any change of the front-end, such as display style, has little effect on other layers. Meanwhile, the changes in the underlying architecture, such as the database, do not affect the front-end. These characters significantly improve the system's usability.
2.3. ThinkPHP

ThinkPHP is a lightweight, easy-to-use and compatible PHP development framework, developed and maintained by a Chinese team. The framework is completely free and open source, using the MVC pattern and object-oriented development structure. ThinkPHP is transplanted from Structs with lots of improvement. ThinkPHP can run under servers such as Linux/Unix/Windows and supports multiple types of databases. In a system development scenario, most of the requirements in system development can be solved with ThinkPHP. ThinkPHP itself contains many common system elements and is very convenient for cross-platform, cross-version and cross-database transplant, so in the development, we can only focus on the business logic. The use of the ThinkPHP framework maximizes the development efficiency of the system, as well as enhances the maintainability and scalability of the system, and reduces the difficulty of system maintenance.

2.4. Apache Web Server

Apache has been the most popular web server on the internet since 1996, with more than half the sites on the web running on it. It played a crucial role in shaping and making the World Wide Web what it is today.

Apache is a free and open source web server. The source code is visible to anyone and everyone, which enables anyone to adjust the system, optimize it and fix errors and security holes. People can add new features and write new modules. Apache can be used not only for small websites of one or two pages but also for large websites of hundreds and thousands of pages, serving millions of regular visitors each month. It can be applied
to different platforms, serves both static and dynamic content, and has stable performance, high security and fast speed.

2.5. MySQL

MySQL is an open source relational database management system (RDMS); it is free and open-source under the terms of the GNU General Public License. MySQL was owned and sponsored by the Swedish company MySQL AB. MySQL is a component of the LAMP web application software stack, which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python.

MySQL is based on a client-server model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). MySQL server is available as a separate program for use in a client-server networked environment and as a library that can be embedded (or linked) into separate applications. MySQL operates along with several utility programs which support the administration of MySQL databases.

Commands are sent to MySQL server via the MySQL client, which is installed on a computer. MySQL enables data to be stored and accessed across multiple storage engines and is also capable of replicating data and partitioning tables for better performance and durability. MySQL users aren't required to learn new commands; they can access their data using standard SQL commands.

MySQL is written in C and C++ and accessible and available across over 20 platforms, including Mac, Windows, Linux, and Unix. The RDMS supports large databases with millions of records and supports many data types. For security, MySQL uses an access privilege and encrypted password system that enables host-based verification. MySQL clients can connect to MySQL Server using several protocols,
including TCP/IP sockets on any platform. MySQL also supports several client and utility programs, command-line programs and administration tools such as MySQL Workbench.

With so many advantages, MySQL is widely used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress, and used by many popular websites, including Google, Facebook, Twitter, Flickr, and YouTube.

2.6. Docker

Docker is developed by Docker Inc; it was first released in 2013. Docker is a computer program that can perform operating-system-level virtualization, also known as "containerization." Docker is used to running software packages called "containers." Containers are isolated from each other and bundle their application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and are thus more lightweight than virtual machines. Containers are created from "images" that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.

The Docker software is consisting of three components: software, container, and images.

Software: the Docker daemon, called dockerd, is a persistent process that manages Docker containers and handles container objects. The daemon listens for requests sent via the Docker Engine API. The Docker client program, called Docker, provides a command-line interface that allows users to interact with Docker daemons.
Objects: Docker objects are various entities used to assemble an application in Docker. The main classes of Docker objects are images, containers, and services. A Docker container is a standardized, encapsulated environment that runs applications. A container is managed using the Docker API or CLI. A Docker image is a read-only template used to build containers. Images are used to store and ship applications. A Docker service allows containers to be scaled across multiple Docker daemons. The result is known as a "swarm", a set of cooperating daemons that communicate through the Docker API.

Registries: A Docker registry is a repository for Docker images. Docker clients connect to registries to download ("pull") images for use or upload ("push") images that they have built. Registries can be public or private. Two main public registries are Docker Hub and Docker Cloud. Docker Hub is the default registry where Docker looks for images. Docker registries also allow the creation of notifications based on events.
CHAPTER 3 SYSTEM ANALYSIS

3.1. Feasibility Analysis

Feasibility analysis is necessary for system development; it measures the ability and likelihood to complete a project successfully including all relevant factors. Feasibility analysis accounts for factors that affect the development of a system, such as economic, technological, legal and scheduling factors. In the actual design and development process, the system will always encounter various unpredictable problems, such as technical problems, economic problems, and time problems. Any of these problems will lead to the termination of development; then the previous development must be abandoned, bringing unpredictable losses of money and resources.

3.1.1. Market Feasibility

SQL queries can be a big difficulty in the learning of database management. The syntax of the query is not easy to write correctly. Moreover, a query to find some specific data is the toughest, especially when a query is related to several different tables. Without the right statement, it is impossible to get the data we need from the database. Most of the practice is from textbooks; students look at database tables and figure out the query statement to get the aimed data. It is hard to check whether the query is correct. Meanwhile, the only way for a teacher to know the students' ability is to give them homework. A more efficient way is needed to help both teachers and students.

3.1.2. Technical Feasibility

Technical feasibility analysis refers to whether the technical methods used in the development of the system can meet the requirements of the system. Within the existing
technical level, the complete development of the system can be realized. Therefore, the technical feasibility analysis is also very important; it determines whether the system can be completed smoothly.

The operating system for developing the project is Linux, web server is Apache web server. It uses MySQL as the database and ThinkPHP as the development framework. Technically, it is possible to finish the project with these tools.

3.2. Functional Requirement Analysis

The system operation process is like that: teacher uploads SQL file and practice questions about SQL query, student logs in to the system and completes practice on the website with these questions. If a query sentence is input correctly, related data will be displayed on the website. Otherwise, the error information will be displayed. There are three different user roles in the system, administrator, teacher, student. Each role has its user requirements and system requirements.

1. The platform will run on a web interface. A regular web browser such as Google Chrome, Firefox, can access the platform.

2. The platform will allow students and teachers to register or log in. Students and teachers have two different login pages. On the teacher's login page, the teacher can log in to the system with the correct username and password. For students, a student can choose to login directly or to register. After register successfully, the student can log in to the system.

3. The platform will keep the username and password for each user. There is a database named platform on MySQL server, all accounts' information is stored in this database.
4. The teacher can upload a SQL file to generate a database. All statements in the file will be executed and create a new database with this teacher's name on the database server.

5. The teacher can delete the database created by him.

6. The teacher can add, query and delete questions and answers to the database by uploading a SQL file. This file includes commands to create a question table as well as questions descriptions and solutions. Data of questions will be stored in the new database.

7. The teacher can create more than one database for students to practice. By repeating #4 and #5, the teacher can create multiple databases.

8. Student gets one or more databases which assigned by the teacher to practice. When a student logs in, the databases created by his teacher are displayed on the web page.

9. The student can do practice on the web page. The student clicks the database name that he chooses to practice, a Docker container will be created on the web server. This container is based on MySQL image, all commands that used to create teacher's database and to create the question table are executed in the container. The description of the database such as the names of tables, detail of tables is displayed. Questions related to the database are displayed on the page.

10. The Student can submit his answer to the web server. After the student finishes answering one question, he can click the submit button.
11. The student will get feedback after he submits his answer. If the answer has no syntax error, the result can be displayed on the web page. If there are some syntax or another kind of errors, details are displayed.

12. The student has the option to know the solution to any practice question. By clicking the answer button, the correct answer is displayed on the page.

13. Multiple students can practice simultaneously. Each student's process is running in a different Docker container; the original database created by teacher keeps completely unaffected. When the student logs out of the system, his container will be destroyed.

3.2.1. Teacher

The teacher can upload a SQL file to generate a database. Teacher uploads a SQL file, which includes several tables and data. A database is generated by the file automatically. The database created by the file is a separated one from the admin database. The teacher can name the database. Meanwhile, a record with this database name will be added into the teacher’s account table in the admin database.

The teacher can query or delete the database created by him. The teacher can get the names of databases he has created from the admin database. When the teacher decides to delete a database, the database is deleted according to the name, and the record in the admin database will be deleted.

The teacher can add, query and delete questions and answers to the database. The teacher can upload a text file which has practice question and answers. Each pair of question and answer is considered as one record and separated by a semicolon. All
questions will be uploaded into the admin database, with the information of the database name.

The teacher can create more than one database for students to practice. Teacher uploads another SQL file; a new database is created by the file. Practice questions can be added to this database. It is possible for the teacher to check all databases he has created.

![Figure 4 Teacher use case](image)

### 3.2.2. Student

Student gets the database to practice. After student logs in to the platform, he can get a name list of databases that the teacher assigns to him. Student picks one database, a description of the database such as table names, table attributes and other details will be displayed. Questions that ask the students to query the database are also displayed. An input area is used for the student to write the SQL query as the answer to the question. After the student submits the answer, the query result is displayed. If the query is correct,
data from the database will be displayed; if not, the error information generated by MySQL will be displayed. The student has the option to get the solution of the question after several unsuccessful attempts.

Multiple students can practice simultaneously. Every student logs in with his account. The student can only see the databases assigned by his teacher. Student’s SQL operation won’t modify the database created by the teacher. What student get is an instance of the teacher’s database, so when another student is doing the practice, he can still get the unmodified database. After the student finishes practicing, the database instance will be destroyed.

3.3. Non-functional Requirement

3.3.1. System Configuration

The platform will be hosted on a web server. I will use my laptop as a web server. The technical specification of my laptop is as follows:
• 2.6GHz 6-core Intel Core i7, Turbo Boost up to 4.3GHz, with 9MB shared L3 cache
• 512GB SSD
• 16GB of 2400MHz DDR4 onboard memory
• Radeon Pro 560X with 4GB of GDDR5 memory and automatic graphics switching
• 802.11ac Wi-Fi wireless networking; IEEE 802.11a/b/g/n compatible

3.3.2. Database

There is a database server used for this platform too. On the database server, there are several independent databases. A database named admin will be used to store administrator, teacher and student accounts. To achieve the goal of database isolation and system security, only the information of administrator, teacher and student accounts is stored in the admin database, practice questions and keys are stored in the practice database. A student gets an instance of the database generated by the teacher's SQL file. All operations the student does during the practice will work on the copied database, rather than the database created by the teacher. When the student leaves the platform, his database instance will be dropped.

3.3.3. Docker

Student’s process will run in a Docker container. When a student is practicing on his database instance, there are several risks to the web server. For example, the student can insert huge data to his database to make the web server crashed down. A student may stay on the platform too long without any action which will slow down the server's performance. To make sure the system security, all students’ processes are run in a
Docker container. Whenever some danger is censored, the container can be killed by the server.

### 3.3.4. Multiple Users

The platform will support multiple simultaneous users without noticeable slowdown. When there is more than one student doing practice on the platform at the same time, there should not have any significantly increased wait times for any student.
CHAPTER 4 SYSTEM DESIGN

4.1. Framework

The system of the MySQL Online Practice Platform uses the B/S architecture. Under the B/S architecture, the project is implemented with three layers, presentation layer, application layer, and data layer. A web browser is used as an interface. The web server analyzes the information and get the SQL query required by the database server, then gives the query to the database server. The database server executes the query and provides the result to the web server. The web server then displays the information through the web browser.

There are two kinds of user roles in the project: the teacher and the student. The graph below is used to demonstrate the design of the system framework as a teacher.

![System design of teacher](image)

Figure 6 System design of teacher
On the presentation layer, when the teacher tries to log into the system, his request is sent to the web server including username and password. The web server will analyze the information and get the SQL query required by the database server, then gives the query to the database layer. In the database server, there is a database named platform. The database server executes the query, if the username and password can be found in the database platform, then the request to log in is allowed. The decision is sent to the web server, and login information is displayed on the browser.

The role of the teacher has another main function, which is to upload a SQL file to create a database. The SQL file is uploaded to the web server. The web server transfers all SQL commands in the file and sends them to the database server. The database server executes all commands to create a dynamic database which is named by the teacher. After this, the teacher can upload a SQL file with homework questions. The commands in the question file will be executed both in the platform database and the dynamic database. The information is added to the question table of platform database. Meanwhile, a question table is created in the dynamic database with the questions and solutions.

For the role of the student, the data layer is a little different from the system design of the teacher.
When the student inputs his username and password to log in to the system, his request will be sent to the database named platform in the database server. If the username matches the password in the database, then the student can log in. After he logs in, he can see all the databases created by his teacher. When he chooses one name of these databases, the web server sends the request to connect that dynamic database. Since we need to achieve the goal of isolation, the student won't be able to edit that database. A Docker container is created to host the request of this student. All information of the dynamic database including tables and data will be transferred into SQL commands. The Docker container is based on MySQL image of the Docker, so we can execute the SQL commands in the container to make a duplication of the dynamic database. When the student is answering the questions, all operations will be working on the database in the container.
4.2. Function Module

The overall function modules of the system are divided into two, the teacher module and the student module. The corresponding system functions are designed and analyzed according to requirement analysis, as shown in the figure.

![Function Module Diagram](image)

Figure 8 Function module

4.3. Database Design

A basic database named admin is created for the system to store user accounts information and practice questions. Whenever the teacher uploads a SQL file, a database is generated by the file dynamically. When a student logs in to practice, an instance of the dynamic database is created in a Docker container.
The ER diagrams for the platform are as below:

Table 1 ER diagram of teacher

<table>
<thead>
<tr>
<th>teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>password</td>
</tr>
</tbody>
</table>

Table 2 ER diagram of student

<table>
<thead>
<tr>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>password</td>
</tr>
<tr>
<td>teacher</td>
</tr>
</tbody>
</table>

Table 3 ER diagram of teacher_database

<table>
<thead>
<tr>
<th>teacher_database</th>
</tr>
</thead>
<tbody>
<tr>
<td>teacher_database_id</td>
</tr>
<tr>
<td>database_name</td>
</tr>
<tr>
<td>teacher_id</td>
</tr>
<tr>
<td>------------</td>
</tr>
</tbody>
</table>

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1. ThinkPHP Deployment

To deploy the ThinkPHP framework, we need to go to the official website of ThinkPHP to download the framework. The GitHub address for ThinkPHP is https://github.com/top-think/thinkphp. For the project, I choose edition 3.2. The latest updated edition is ThinkPHP 5, which has been made so many changes that it cannot be compatible with edition 3.2. When we open the ThinkPHP folder from the browser, the framework will generate front-end and back-end folders automatically.

This is the structure of the whole system in the project explorer.

![Project explorer](image)

Figure 10 Project explore

All resource code is in folder Application. PLATFORM/Application/Admin is the back-end of the project, which can be accessed by the teacher only in the system. The IP address for the teacher to log into the system is http://localhost/platform/index.php/Admin. PLATFORM/Application/Home is the front-
end of the project; students can access the system and do practice. The IP address is http://localhost/platform.

### 5.2. Shell Environment Configuration

Because the process of the student is running in Docker container, we need to install Docker firstly. The Docker company provides very detailed instructions to install docker on Linux. After docker installed successfully, to run a Docker command through PHP code, the file sudoers has to be edited. Run the following code in PHP to get web service's user name:

```php
<?php
    echo exec('whoami')
?>

> vim /etc/sudoers
```

Assume the result is daemon. Edit /etc/sudoers and insert "daemon ALL=(ALL) NOPASSWD:ALL" after "root ALL=(ALL) ALL". Use the following command to open sudoers.

```
vim /etc/sudoers
```

The command allows root to run any commands anywhere.

```
## Allow root to run any commands anywhere
root      ALL=(ALL)     ALL
daemon    ALL=(ALL)      NOPASSWD:ALL
```

Run "whoami" locally, get the current user.

```
Whoami
```

In my computer, the result is stella. Create new user group docker, add current user to the group.

```
sudo groupadd docker
sudo gpasswd -a stella docker
newgrp docker
```
Let’s test Docker in php,

```php
<?php
echo exec ('docker images');
?>
```

The result shows all Docker images on the computer. Now, all prepared to run the project has finished.

### 5.3. Back-end Module Design

#### 5.3.1. Teacher Login

Every teacher has to login to use the system. There is no register function for teachers. The teachers can only get username and password from the system manager. Below is the login page for teachers.

![Teacher login page](image)

> Figure 11 Teacher login page

After input username and password, click the login button, it will submit the form to Controller; the function is login(). The database server will check whether there is such
user with this password. If the username matches the password, a session is created and sent to the web server, if not, it will give the error information.

This is function login().

```php
//teacher login
public function login(){
    $data['account']=I('account');
    $res=M('admin')->where($data)->find();
    if($res['password']==I('password')){
        session('username',$res['account']);
        session('userid',$res['id']);
        $this->success('successfully login','./mainPage');
    }else{
        $this->error('wrong password');
    }
}
```

5.3.2. Create Database

After the teacher logs in successfully, it will jump to the main page. Below is the main page.

![Database Management](image)

Figure 12 Main page of teacher
To create a new database for practice, the teacher needs to click the green button Add Database. It will jump to a new page as below. Firstly, give a name to the database that we want to create, then click the browse button to find the SQL file that we need, and click the submit button. In the following screenshot, we name the database test and upload a SQL file named test.sql. In the database server, a database named by the teacher’s name and the name we input will be created.

![Figure 13 Upload SQL file](image)

The function for uploading SQL file is called uploadSQLFile(). The web server will analyze the file, get the content of the file, which is SQL commands. Then, the web server sends commands to the database server. The database server will execute all SQL commands to create a new database.

```php
public function uploadSQLFile(){
    //get the current userid
    $userid = session('userid');
    //get the post data database_name
    $databaseName = I('post.database_name');
```
When we click Submit button, it will jump back to main page, and display a list of all databases created by the teacher.
The SQL file that we are using is a regular file; it contains commands to create tables and insert data. This is the content of sample SQL file, test.sql.

```sql
DROP TABLE IF EXISTS `payment`;  
CREATE TABLE `payment` (  
    `id` mediumint(8) unsigned NOT NULL auto_increment,  
    `account_number` varchar(255),  
    `customer_id` mediumint default NULL,  
    PRIMARY KEY (`id`)  
);  
INSERT INTO `payment` (`id`, `account_number`, `customer_id`) VALUES ();  
DROP TABLE IF EXISTS `customer`;  
CREATE TABLE `customer` (  
    `id` mediumint(8) unsigned NOT NULL auto_increment,  
    `name` varchar(255) default NULL,  
    `city` varchar(255),  
    `state` varchar(100) default NULL,  
    PRIMARY KEY (`id`)  
);  
INSERT INTO `customer` (`id`, `name`, `city`, `state`) VALUES ();
```

5.3.3. Add Question
On the main page, there are four options that we can do to the database, add question, show all questions, delete all questions, and delete the database. When clicking the add question button, it will ask the teacher to upload a SQL file. The file contents question description and solutions. This is the content of the file.

```sql
CREATE TABLE IF NOT EXISTS `question`(
    `question_id` int(100) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    `content` varchar(100) NOT NULL,
    `solution` varchar(100) NOT NULL,
    `database_id` int(100)
);
INSERT INTO `question` (`content`,`solution`) VALUES
('please find the information of customer whose name is Sybil Wallace',
'select * from customer where name = 'Sybil Wallace')',
('create new table name which is item with two attributes, name as varchar(20), price as double(20)'),
'create table if not exists item (name varchar(20), price double(20))',
('please find all information from table payment, customer, where customer id = 14',
'select * from customer, payment where customer_id = 14 and customer.id = payment.customer_id',
'delete record from payment which id is 2',
'delete from payment where id = 2');
```

Click add question button, it jumps to the webpage to upload the fill. Here, we upload a file named quesiton.sql.
After submitting the file, it will jump to the main page again. Now we can click the question list button to check the question list in this database.
The function to query all questions is `questionList()`. When the web server transfers the request to the database server, the feature gives the database server a new database configuration. The configuration converts the working database from platform which is the admin database to the new database created by the teacher. Then database server queries from this database and sends data to the web server.

```php
// show question list
public function questionList()
{
    $id = I('id');
    $TeacherDatabase = new \Admin\Model\TeacherDatabaseModel();
    $condition['teacher_database_id'] = $id;
    $database = $TeacherDatabase->where($condition)->find();
    $databaseName = $database['database_name'];
    $databaseRealName = session('username')->$databaseName;
    // convert to the new database created by teacher from the platform database
    $options = array(
        // database config
        'DB_TYPE' => 'mysql', // database type
        'DB_HOST' => 'localhost', // host address
        'DB_NAME' => $databaseRealName, // database name
        'DB_USER' => 'root', // username
        'DB_PWD' => '', // password
        'DB_PORT' => '3306' // port
    );
    // convert to the database created by teacher, query question table
    $model = M('question', '', $options);
    $list = $model->select();
    $this->assign('list', $list);
    $this->display();
}
```

### 5.4. Front-end Module Design

#### 5.4.1. Student Register and Login

To use the system, students have to log in too. This is the login page below.
The student needs to register as a user before he can use. Click the Register button; it will jump to the register page.

Since every student belongs to a specific teacher, when students are registering, they need to provide their teacher’s name. The function for students to register is
registerUser(). The information in the form will be posted to the web server, web server
then sends the request to the database server to create a new user.

```php
public function registerUser(){
    $data = $_POST;
    if (empty($data['name']) || empty($data['password']) ||
        empty($data['repassword'])) {
        $this->error('user name or password cannot be empty');
    }
    //find the student
    $map['name'] = $data['name'];
    $res = M('student','','platform')->where($map)->find();
    //if can find the student, then need to use a different name
    if($res){
        $this->error('User name already exit');
    }
    //find the teacher
    $teacherName = $data['teacher_name'];
    $condition['account'] = $teacherName;
    $teacher = M('admin','','platform')->where($condition)->find();
    //if teacher with the name doesnot exit, means student cannot register
    if(!$teacher){
        $this->error('Teacher name not exit');
    }
    if($data['password'] != $data['repassword']){
        $this->error('Password does not match');
    }
    //find the teacher id
    $teacherId = $teacher['id'];
    $map['teacher_id'] = $teacherId;
    $map['password'] = $data['password'];
    //add the student to database
    M('student','','platform')->add( $map );
    $this->success('register successfully', 'index');
}
```

When student logs in to the platform, he can see a list of databases that created by
his teacher. To do practice, the student can click Practice button as the screenshot below.
According to the functional requirement of the system, student's SQL operation won't modify the database created by the teacher. We use Docker to achieve the goal. When a student chooses a database to practice, a container which has a name with student name, teacher name and the teacher’s dynamic database name will be created by Docker. Since it takes some time to create a container, a loading page is used to avoid NO-HOST error. When the student clicks the button to do the practice, it jumps to the loading page.
When the student clicks the start button, a Docker container `student1teacher1test` is created.

Figure 20 Loading page

Figure 21 Docker container in the shell
The container is created based on MySQL Docker image. It can be used as a database server. A copy of the teacher’s database is generated in this container. We can enter container to check the information in the database.

![Figure 22 Databases information inside of the container](image)

The function for practice is called test(). Firstly, we read the database created by the teacher, and then store all commands used to create the database. Then, create a Docker container in the shell. This container is based on MySQL image. Next, convert the database from teacher's database to database in the Docker container, and run the commands in the container to generate a copy of the teacher's database. In this way, we can realize the isolation and security requirement.

```php
public function test(){
    //find this database name as displayed in admin database table teacher_database
    $database = I('databaseName');
    //find teacher name in session('teacheraccount
    $databaseName = session('teacheraccount').$database;
    //$databaseRealName = session('name').$databaseName;
    $databaseRealName = $databaseName;
    session('databaseRealName', $databaseRealName);
}
```
//create docker container with this name
exec('docker run -p 3301:3301 --name '.$databaseRealName.' -e MYSQL_ROOT_PASSWORD=123456 -d mysql:5.7.25');
sleep(20);
$options = array(
    //database config
    'DB_TYPE' => 'mysql',
    'DB_HOST' => 'localhost',
    'DB_NAME' => $databaseName,
    'DB_USER' => 'root',
    'DB_PWD' => '',
    'DB_PORT' => '3306'
);
//get all information in teacher's database, and store in $text
$result = M('','',$options)->query('show tables');
$text .= "CREATE DATABASE IF NOT EXISTS `".$options['DB_NAME'].'` DEFAULT CHARACTER SET utf8 ;";
$text1 .= "CREATE DATABASE `".$database.'`;\n"
file_put_contents($file_name,$info,FILE_APPEND);
foreach ($result as $k=>$v) {
    //query table's structure
    $val = $v['tables_in_'].$options['DB_NAME'];
    $sql_table = "show create table ".$val;
    $res = M('','',$options)->query($sql_table);
    $text .= "DROP TABLE IF EXISTS `".$val.'`;\n"
    $text .= $res[0]['create table'].";\n"
    if (($res[0]['table']!='question')){
        $text1 .= $res[0]['create table'].";\n"
    }
    //query table's data
    $sql_data = "select * from ".$val;
    $data = M('','',$options)->query($sql_data);
    $count= count($data);
    if($count<1) continue;
    foreach ($data as $key => $value){
        $sqlStr = "INSERT INTO ".$val.' VALUES (";
        foreach($value as $v_d){
            $v_d = str_replace('"','"\"',$v_d);
            $sqlStr .= "$,".$v_d.'", ";
        }
        //remove comma and space
        $sqlStr = substr($sqlStr,0,strlen($sqlStr)-2);
        $sqlStr .= ";\n";
```php
$text .= $sqlStr;
}
$text .= "\n"
}

// the config to connect the database in container
$options1 = array(
    'DB_TYPE' => 'mysql',
    'DB_HOST' => '172.17.0.2', // ip address
    'DB_NAME' => 'mysql', // database name
    'DB_USER' => 'root', // username
    'DB_PWD' => '123456', // password
    'HOST_PORT' => '3301' // port
);

// create database with the command in $text
$result1 = M('', '', $options1)->query($text);
$text1 = strtolower($text1);
$str = 'engine=innodb default charset=latin1;
$str1 = 'engine=innodb auto_increment=101 default charset=latin1;
$text1 = str_replace($str, ';\r', $text1);
$text1 = str_replace($str1, ';\r', $text1);
$question = M('question', '', $options1)->select();
$count = count($question);
if (empty($count)) this->error('No question exist');

// display questions
$Page = new Think\Page($count, 1);
$Page->setConfig('first', 'First');
$Page->setConfig('prev', 'Prev');
$Page->setConfig('next', 'Next');
$Page->setConfig('last', 'Last');
$show = $Page->show();
$list = M('question', '', $options)->limit($Page->firstRow, $Page->listRows)->select();
$this->assign('page', $show);
$this->assign('text', $text1);
$this->assign('list', $list);
$this->display();
```

This is the web page for students to practice.
On the top is a brief description of this database, including the database name, tables in this database. The student can write his answer at the input box. After submitting the button, if the query sentence is correct, the page will display the query result.
The Answer button will show the correct answer.

If the student's query is not correct, the system will give the error information.
The function to check student’s SQL query is testsql().

```php
public function testsql(){
    $sql = I('sql');
    // get student's answer
    $sql = trim($sql);
    $databaseRealName = session('databaseRealName');
    $options = array(
        'DB_TYPE' => 'mysql',
        'DB_HOST' => '172.17.0.2',
        'DB_NAME' => $databaseRealName,
        'DB_USER' => 'root',
        'DB_PWD' => '123456',
        'HOST_PORT' => '3306'
    );
    $res = M('', '', $options)->query($sql);
    $error = M('', '', $options)->getDbError();
    $result_structure = array();
    $temp_array=$res;
    $element_index=1;
    // display query result
    if (count($temp_array) > 1){
        while (count($temp_array)!=1){
            $result_structure[$element_index]=count($temp_array);
            $element_index=$element_index+1;
        }
    }
}```
$temp_array = $temp_array[array_keys($temp_array)[0]]; 
}

$first_element = $res[array_keys($res)[0]]; 
$table = '<html><body><table width="80%" border="8">'; 
$table .= '<tr>'; 
foreach (array_keys($first_element) as $v) {
    $table .= '<th>"$v."</th>
}; 
$table .= '</tr>'; 
foreach ($res as $line_record) {
    $table .= '<tr>'; 
    foreach ($line_record as $v) {
        $table .= '<td>"$v."</td>
}; 
$table .= '</tr>'; 
}$table .= '</table></body></html>'; 
}

if (count($temp_array) == 1){
    $first_element = $res[array_keys($res)[0]]; 
    $table = '<html><body><table width="80%" border="8">'; 
    $table .= '<tr>'; 
    foreach (array_keys($first_element) as $v) {
        $table .= '<th>"$v."</th>
}; 
    $table .= '</tr>'; 
    foreach ($res as $line_record) {
        $table .= '<tr>'; 
        foreach ($line_record as $v) {
            $table .= '<td>"$v."</td>
}; 
        $table .= '</tr>'; 
    }
    $table .= '</table></body></html>'; 
}

if (empty($error)) {
    $this->show($table); 
} elseif ($error == "":
    echo M('', '', $options)->getLastError(); 
    echo '<br/>'; 
    echo ('Execute successfully'); 
} else {
    echo($error); 
} 
}
5.5. **Multiple Users**

The project allows various users using the platform simultaneously, which means more than one student belongs to the same teacher can practice on the same database in the same time, or more than one student belongs to different teachers can practice on different databases in the same time.

5.5.1. **Students with the Same Teacher**

Students with the same teacher will see the same list of databases created by the teacher. For example, student1 and student2 are both belong to teacher1. Teacher1 creates a database named test. When the two students log into the system, the main page of each student displays the database test. When clicking the practice button, two containers with the name pattern student name, teacher name, and the name that the teacher types in are created.

5.5.2. **Students with Different Teacher**

Students with different teachers will see the list of databases created by his teacher. For example, teacher1 creates a database test; teacher 2 creates a database exercise. Student1's teacher is teacher1, and student3's teacher is teacher2. When the students log in, they will only see the database created by their teacher. When clicking the practice button, two containers with the name pattern student name, teacher name, and the name that the teacher types in are created.
CHAPTER 6 SYSTEM TESTING

6.1. System Testing Environment

- 2.6GHz 6-core Intel Core i7, Turbo Boost up to 4.3GHz, with 9MB shared L3 cache
- 512GB SSD
- 16GB of 2400MHz DDR4 onboard memory
- Radeon Pro 560X with 4GB of GDDR5 memory and automatic graphics switching
- 802.11ac Wi-Fi wireless networking; IEEE 802.11a/b/g/n compatible

6.2. Testing Examples and Result

6.2.1. Teacher

<table>
<thead>
<tr>
<th>ID</th>
<th>Test Description</th>
<th>Step #</th>
<th>Test Step</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Verify that the teacher’s main page is NOT accessible to users without permission role</td>
<td>#1</td>
<td>User without admin role login using correct username and password</td>
<td>Users should log in successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#2</td>
<td>Navigate to teacher’s main page</td>
<td>The main page should display the options</td>
</tr>
<tr>
<td>2</td>
<td>Verify teacher can upload SQL file to create a database</td>
<td>#1</td>
<td>User clicks add database button</td>
<td>The upload SQL page shows up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>User clicks submit button</td>
<td>The SQL file is uploaded</td>
</tr>
<tr>
<td>3</td>
<td>Verify teacher can upload SQL file to add questions</td>
<td>#1</td>
<td>User clicks add question button</td>
<td>The upload SQL page shows up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#2</td>
<td>User clicks submit button</td>
<td>The SQL file is uploaded</td>
</tr>
</tbody>
</table>
### 6.2.2. Student

Table 5 Testing examples – student

<table>
<thead>
<tr>
<th>ID</th>
<th>Test Description</th>
<th>Step #</th>
<th>Test Step</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Verify that the platform is accessible to students</td>
<td>#1</td>
<td>Student with legal username and password login the webpage</td>
<td>Student logs in successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#2</td>
<td>Student clicks practice button</td>
<td>The practice page shows up</td>
</tr>
<tr>
<td>2</td>
<td>Verify student can use the platform to practice SQL syntax</td>
<td>#1</td>
<td>Student clicks submit button</td>
<td>The query result is displayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#2</td>
<td>Student clicks the answer button</td>
<td>The correct answer is displayed</td>
</tr>
</tbody>
</table>

### 6.2.3. Multiple Users

Table 6 Testing examples - multiple users

<table>
<thead>
<tr>
<th>ID</th>
<th>Test Description</th>
<th>Step #</th>
<th>Test Step</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Verify students with the same teacher can use the platform to practice SQL syntax</td>
<td>#1</td>
<td>Student1 start practicing</td>
<td>The practice pas shows up. After submits query sentence, the query result is displayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#2</td>
<td>Student2 start practicing</td>
<td>The practice pas shows up. After submits query sentence, the query result is displayed</td>
</tr>
<tr>
<td>2</td>
<td>Verify students with different</td>
<td>#1</td>
<td>Student1 start practicing</td>
<td>The practice pas shows up. After</td>
</tr>
</tbody>
</table>

48
teachers can use the platform to practice SQL syntax

| #2       | Student3 start practicing | The practice passes up. After submits query sentence, the query result is displayed. |
CHAPTER 7 CONCLUSION

The project provides a new way for computer students to learn database query. By using today's latest Internet technology, the design and implementation of the database practice platform system achieve most of our goals. Due to the lack of time and personal programming ability, the system still has some functions to be improved.
BIBLIOGRAPHY

Liu Yuan, Huang Haoguang, Hu Xiaobao, Design and implementation of a campus second-hand trading platform based on ThinkPhp. *Computer Era*, Editorial E-mail, 2018(10).

WANG Junfang, LI Yinfeng, WANG Chi, Research on ThinkPHP Framework Based on the Mode of MVC. Electronic Science and Technology, Editorial E-mail, 2014(04).

LIN Yu-man, QIN Hao-huan, GU Yong-lun, RAO Hao, Teaching Resource Management Platform Based on ThinkPHP. *Computer Knowledge and Technology*, Editorial E-mail, 2018(07).

LI Yan, WANG Siyan, Design and Implementation of the College Book Review System Based on ThinkPHP Framework. *Software Engineering*, Editorial E-mail, 2016(07).

