**深 圳 大 学**

**实 验 报 告**

**课程名称： 数据库系统**

**实验序号： 实验3**

**实验名称： DBMS Application**

**学 号： 2021220003**

**姓名： 梁润宇**

**实验完成日期： 2023 年 11 月27 日**

**一、实验目的：**

1. 了解DBMS系统的功能、Web based数据库的软件组成及工具；

2、掌握数据库软件的使用方法、

3、掌握前台页面，后台数据库的基于WEB的数据库操作方法。

**二、实验要求：**

1、安装相关软件并浏览软件自带的帮助文件和功能菜单，

2、掌握PHP构建网页以及连接数据库的方法。

3、掌握phpmyadmin创建数据库与数据表的方法；

4、了解mysql的命令以及与php衔接的语句

5、了解网页构造的原理

**三、实验设备：**

计算机、数据库管理系统如DB2，mysql 等软件。

**四、实验内容**

**1. PHP+MySQL实现新闻系统**

本部分使用PHP+MySQL实现1个基础的新闻系统，其页面包括首页、查看新闻、搜索新闻、添加新闻。

* 1. **首页**

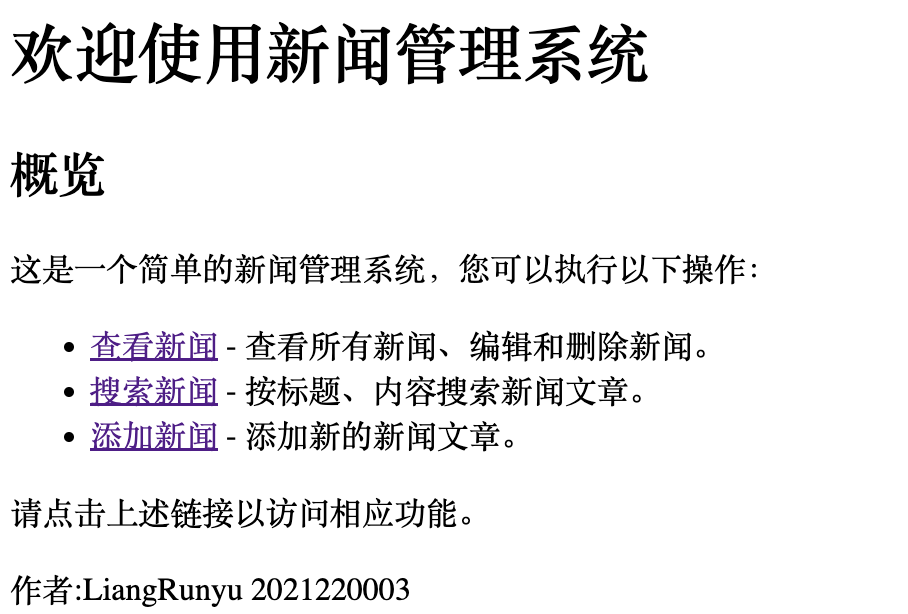


Fig.1 新闻系统首页



Fig.2 首页代码

首页提供一个导航的作用，可以进入到各个区域。

<ul>表示无序列表，<a>表示超链接，在本页面中用于跳转到其他页面。<p>表示段落，<h1>和<h2>表示一级标题和二级标题。

* 1. **数据库设计**

在phpMyAdmin内直接添加。



Fig.3 新闻表

第一个字段名为"id"，长度值为20，设置为非空，意思是在添加数据时这个字段为必填，如果没填则会出错。额外选择"auto\_increment"意思是该字段为自增长型，在添加一项数据时不用填这个字段它都会自动填入一数字.接着在旁边的主键那里选择上它。让这个id字段作为主键，一般数据表都会创建"id"字段作业主键以方便以后的操作。

接着设置第二个字段，字段名为"title"，作业新闻的标题，类型选择"VARCHAR"，长度设置为200； 接着设置第三个字段了，字段名为"add\_time"，用来存放新闻添加的时间，类型先"VARCHAR"吧，为什么不选"DATE"作业类型呢?因为选"DATE"类型号在以后的添加等操作中会很麻烦，而用VARCHAR类型可以实现"DATE"类型的功能而且操作起来很简单，长度100；第四个字段名为"content"，用来存放新闻的主要内容，类型选"TEXT"，就是文本内型，这种类型可以存放较多的字，这种类型的长度是不用填的；到这里所有字段都设置好了，点"保存"就可以了。

* 1. **数据库连接**

****

Fig.4 conn.php-数据库连接

保存为conn.php，那么在以后要用到这些代码时只要include这个页面就可以了.如add.php这个页面要要添加数据入数据库,那么在add.php里include "conn.php"; 就可以了.

* 1. **新闻查看**



Fig.5 新闻查看及管理

本功能在news.php中实现。

新闻查看的时候只显示ID和标题，便于查找。

本页面的底部设置“增加新闻”功能，点击后进入到增加新闻。每一条新闻的右侧都有修改和删除，点击后分别进入相应的操作。点击修改后可以看到全文，并决定是否进行修改。点击删除后直接删除。

新闻查看的思路很简单，从数据库中选择id和title，每条新闻之间用分割线<br>隔开，每条新闻右侧加上“修改”和“删除”按钮。修改和删除在另外的代码里定义实现。



Fig.6 显示新闻列表

* 1. **新闻添加**

使用add\_news.php和save\_news.php完成。前者用于创建表单并将表单传递给save\_news.php，后者用于连接数据库并发送sql指令。

****

Fig.7 add\_news.php

用户在填写表单后，点击提交按钮将触发POST请求，将数据发送给指定的服务器端脚本。



Fig.8 save\_news.php

当用户通过前端表单以POST方式提交新闻标题和内容时，save\_news.php会首先检查请求的合法性。如果是有效的POST请求，它会提取表单数据，检查标题和内容是否都已填写。若数据完整，它将准备SQL语句，将新闻标题、内容以及当前时间插入到数据库中。接着，它执行SQL语句，检查插入是否成功，并输出相应的成功或失败信息。如果请求方式不是POST，脚本会输出无效请求方式的错误信息。不论成功或失败，最后提供了一个链接，允许用户回到主页。最终，关闭了与数据库的连接，确保资源的释放。

* 1. **新闻查询**

本段代码实现了一个简单的新闻查询页面。用户可以通过在两个不同的表单中输入关键词（一个用于标题，另一个用于内容）来进行查询。当用户提交查询后，页面通过GET请求将查询关键词发送到"query\_news.php"进行处理。在处理阶段，页面检查用户输入的关键词类型，构建相应的SQL查询语句，并在数据库中执行模糊查询。如果有查询结果，页面会显示匹配的新闻标题和内容，否则将输出未找到匹配新闻的提示信息。同时，如果用户未输入查询关键词就进行了查询，页面会提醒用户请输入查询关键词。整个过程涉及到与数据库的连接，确保查询操作可以有效执行。





Fig.9 query\_news.php

**1.7 新闻编辑**

由于每个独立的编辑按钮放在每条新闻的右边，通过GET请求就可以获取要编辑新闻的ID。如果是有效的GET请求，页面会查询数据库，获取要编辑新闻的信息，并显示一个编辑表单，允许用户更新新闻的标题和内容。用户在编辑表单中提交修改后，页面通过POST请求将更新后的数据发送到"edit\_news.php"进行处理。处理阶段，代码会检查并获取用户输入的新标题和新内容，然后构建SQL语句执行更新操作。如果更新成功，页面会显示“新闻已成功更新”；否则，会输出相应的错误信息。



Fig.10 edit\_news.php

**1.8 新闻删除**

获取新闻的id，然后删除。



Fig.11 del\_news.php

**1.9 数据库连接**

这部分就是conn.php，需要连接数据库的时候引用即可。



Fig.12 conn.php

**2.思考题**

建立相关数据的数据库，并用一个页面显示所有的表名，即要求用户点击该表名显示出该表所有记录。并完成插入记录，删除记录以及修改记录的功能。

**2.1 一个页面显示所有的表名**

使用SHOW TABLES 查询，获取了数据库中的所有表名，并将这些表名存储在 $tables 数组中。每个表名都被添加到数组中，然后在 HTML 页面的部分使用 PHP 的 foreach 循环遍历这个数组，以生成包含表名的列表。这样，用户就可以点击表名链接来查看特定表的记录。

// php

// 获取所有表名

$tables = [];

$result = $conn->query("SHOW TABLES");

while ($row = $result->fetch\_row()) {

$tables[] = $row[0];

}

// html

<!-- 显示所有表名 -->

<ul>

<?php foreach ($tables as $table): ?>

<li><a href="?table=<?php echo $table; ?>">

<?php echo $table; ?>

</a></li>

<?php endforeach; ?>

</ul>

效果展示：

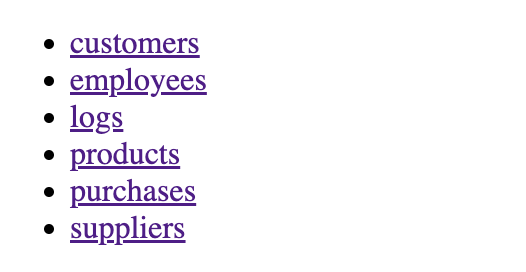


Fig13. 显示所有表名

**2.2 点击该表名显示出该表所有记录**

用户点击了某个表名链接，获取所选表的所有记录和表头信息，并存储在 $records 和 $tableHeaders 变量中。这些信息在 HTML 部分使用 PHP 的 foreach 循环来生成表格，展示了该表的所有记录。

// 处理表的点击事件

if (isset($\_GET['table'])) {

$tableName = $\_GET['table'];

// 获取表中所有记录

$records = [];

$result = $conn->query("SELECT \* FROM $tableName");

while ($row = $result->fetch\_assoc()) {

$records[] = $row;

}

// 获取表头信息

$tableHeaders = [];

$result = $conn->query("SHOW COLUMNS FROM $tableName");

while ($row = $result->fetch\_assoc()) {

$tableHeaders[] = $row['Field'];

}

// 获取主键列名

$primaryKey = $tableHeaders[0];

// ... 省略其他代码 ...

}

在html页面中，调用foreach 循环，遍历表头信息和表中所有记录，生成相应的HTML表格。每一条记录都有一个“删除”链接和一个“编辑”链接，用户可以点击这些链接执行删除和编辑操作。



Fig.14 查看数据表

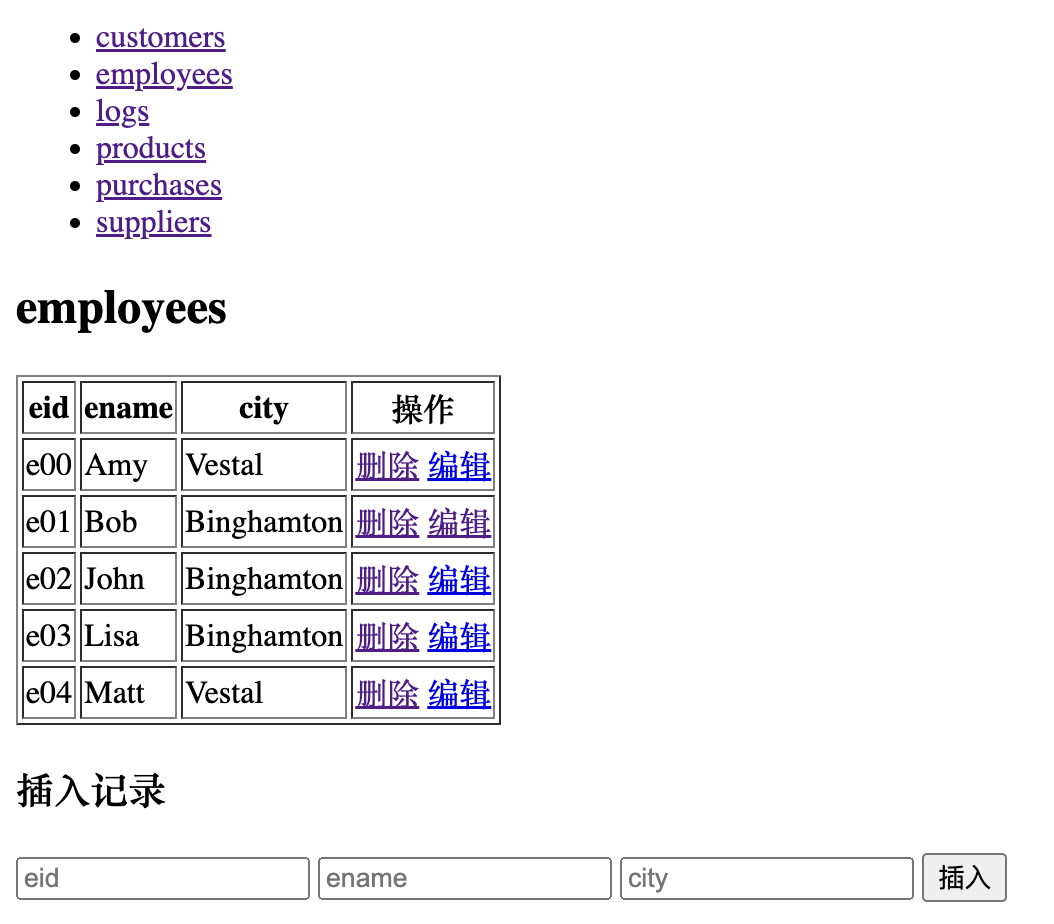


Fig.15 查看数据表效果

**2.3 插入记录**



Fig.16 插入记录对应代码

首先，代码通过检查 HTTP 请求的方法（$\_SERVER['REQUEST\_METHOD']）和是否存在插入记录的 POST 数据来确定是否触发了插入记录的操作。

如果满足插入记录的条件，代码将通过 implode 函数将插入数据数组 $\_POST['insert'] 转换为用逗号分隔的字符串。然后，构建插入记录的 SQL 查询语句（$insertQuery），并使用数据库连接对象执行该查询。

如果插入操作成功，代码通过 header("Location: ?table=$tableName"); 进行重定向，以避免重新提交表单。这里使用了 PHP 的 header 函数来发送原始的 HTTP 头信息，实现页面的跳转。

如果插入操作失败，代码通过 echo "插入记录时出现问题: " . $conn->error; 输出错误信息。

实例：插入一个eid重复的记录



Fig.17 插入失败

可以在页面顶部看到插入失败的提示信息，即主键重复。

**2.4 删除记录**



Fig.18 删除记录对应代码

首先，代码通过检查是否存在删除记录的 GET 请求参数（$\_GET['delete']）来确定是否触发了删除记录的操作。

如果满足删除记录的条件，代码获取要删除的记录的主键值。然后，构建删除记录的 SQL 查询语句（$deleteQuery），并使用数据库连接对象执行该查询。

如果删除操作成功，代码通过 header("Location: ?table=$tableName"); 进行重定向，以避免重新提交表单。这里使用了 PHP 的 header 函数来发送原始的 HTTP 头信息，实现页面的跳转。

如果删除操作失败，代码通过 echo "删除记录时出现问题: " . $conn->error; 输出错误信息。

**2.5 修改记录**



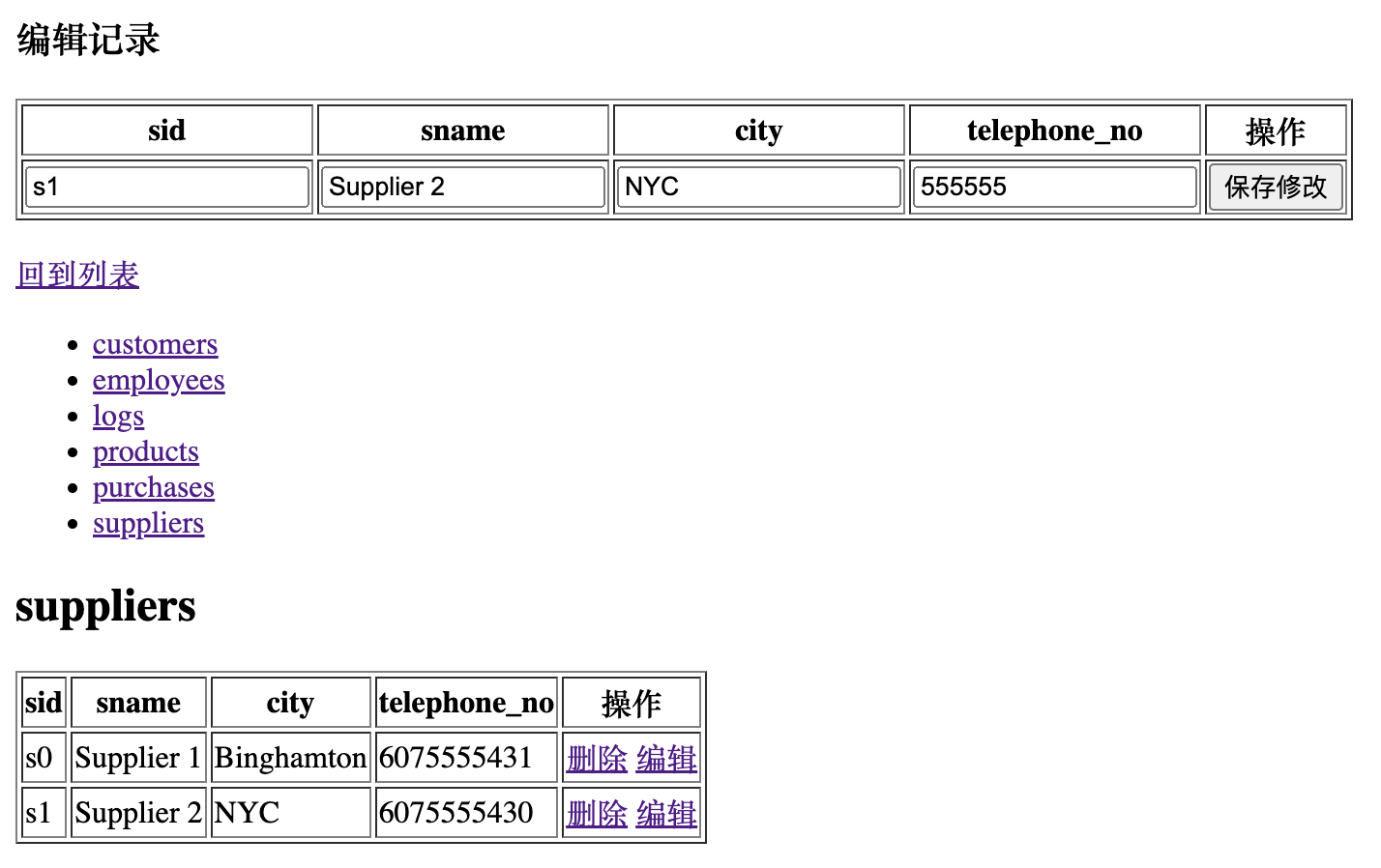


Fig.19 修改记录对应代码

首先，代码检查是否存在编辑记录的 GET 请求参数（$\_GET['edit']）以及请求方法是否为 GET。如果满足这些条件，表示用户正在编辑记录。然后获取要编辑的记录的主键值（$editId），查询数据库，获取要编辑记录的详细信息。如果找到要编辑的记录，代码显示一个包含表单的 HTML 页面，允许用户编辑记录的各个字段。

用户提交编辑后的表单时，通过 POST 请求处理。代码获取提交的编辑数据（$updateData），并构建更新记录的 SQL 语句（$updateQuery），执行 SQL 查询，更新数据库中的记录。再根据操作结果输出相应的信息，成功时提示"记录已成功更新"，失败时提示具体的错误信息。

最后，提供一个链接，让用户返回到表格列表。





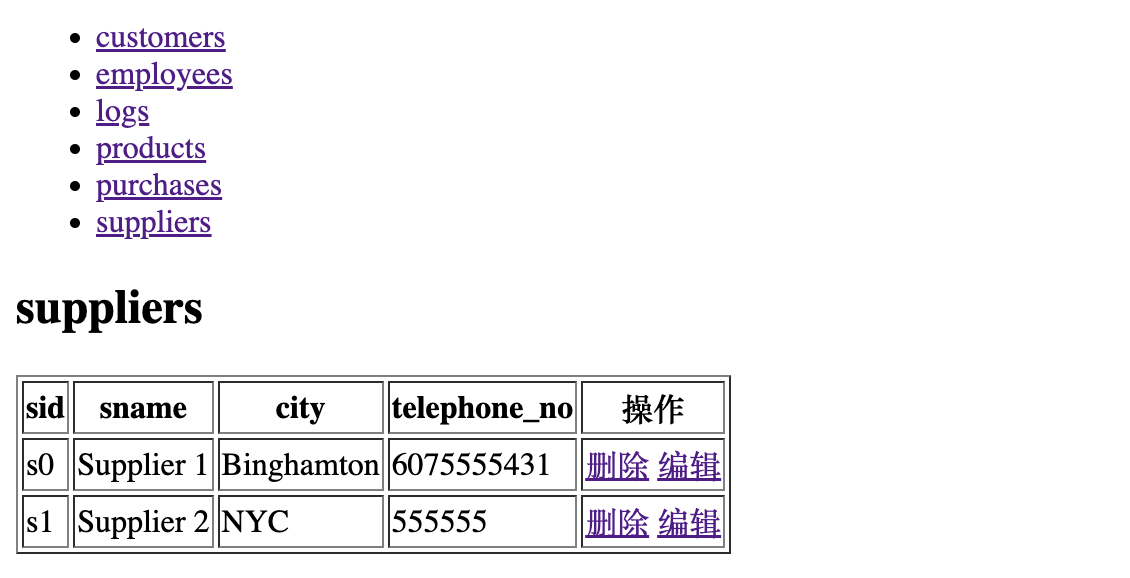


Fig.20 修改记录

**五.问题分析（碰到什么问题，如何解决）**

一开始遇到了无法删除的问题，在phpmyadmin里试了一下也无法删除，但给出了不能删除的原因-外键约束。于是在自己代码中加上输出提示信息的功能。

**六.实验心得**

本次实验练习了数据库的增删改查，同时也注意到在数据库设计的过程中约束的重要性。熟悉了php和html代码，对未来的网页开发积攒经验。

**七.诚信承诺**

**本组成员郑重承诺在项目实施的过程中不发生任何不诚信现象，一切不诚信所导致的后果均由本组成员承担。**

**签名**：

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |