

简易银行个人业务管理系统

背景简介：

目前而言，人们网络支付的频率日益增高，信息管理，人民币数字化趋势，人们逐渐重视网络货币的发展。这其中，利用数据库管理银行的个人业务无疑是最常见，使用频率最高的业务，虽然目前的银行数据库系统已经非常完善且复杂，但制作一个简易的银行业务数据库有利于我们学习数据库知识，并更好将其运用到实际中。

这次项目我主要采用了一体化的xmapp作为本地服务器代理（xmapp自带了Apache，mysql等工具）利用html和css，js制作简单的前端界面，利用php连接数据库并进行操作，其中php连接的数据库通过phpmyadmin管理，数据库采用innodb存储。

目录：

1. 背景简介

2. 需求分析

3. ER模型

4. 关系表和规范化

5. 数据库建立

6. 用例场景示范

7. 总结和后记

2. 需求分析：

一、有客户记录功能：即customer包括属性name，id，telephone

二、为客户提供存款功能即account实体，和depositor关系

三、为客户提供专业的银行员工服务（多对多的关系）

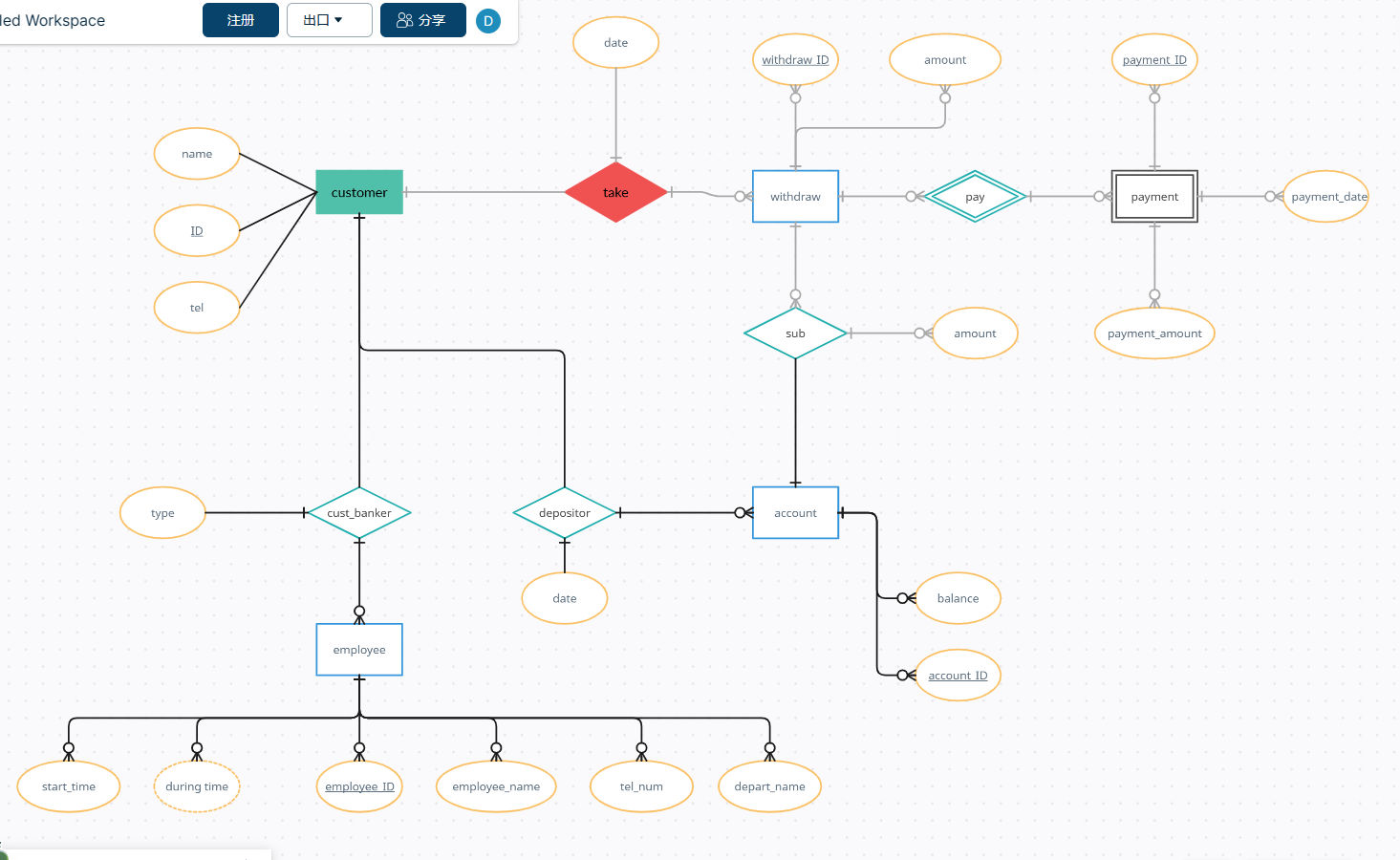
四、用户会消费，即payment实体，但每次消费都会从账户中取钱出来消费，所以payment实体是依赖于取钱（withdraw）的弱实体关系集。

五、用户需要从账户中取钱：customer需要通过 “take”关系与withdraw实体集联系起来。

六、每次用户取钱后，账户中应该减少相应的金额。（个人感觉自己为此设计的sub关系并不好用。）

3. ER模型

最后的er图模型示意如下：



4. 关系表和规范化

* 表 account 的列名：acc\_ID, balance,

主键：acc\_ID

* 表 cust\_banker 的列名：cus\_id, employee\_Id, type,

主键：cus\_id,employee\_id ；外键：cust\_banker (cus\_id)→customer（cus\_ID）；

* 表 customer 的列名：name, cus\_ID, cus\_tel,

主键：cus\_ID；

* 表 depositor 的列名：cus\_id, account\_id, deposit\_Date,

主键：cus\_id ；外键：depositor (cus\_id)→customer（cus\_ID），

depositor (account\_id)→account（acc\_ID）

* 表 employee1 的列名：tel, e\_ID, name,

主键：e\_ID

* 表 employee2 的列名：e\_ID, start\_time, depart,

主键：e\_ID

* 表 emplyoee3 的列名：start\_time, tel,
* 表 payment 的列名：pay\_ID, pay\_date, pay\_amount, withdraw\_ID,

主键：pay\_ID, pay\_date, pay\_amount, withdraw\_ID, ；

外键：payment (withdraw\_ID)→withdraw（ID）

* 表 sub 的列名：account\_ID, withdraw\_ID, amount,

主键：withdraw\_ID；外键：sub (account\_ID)→account（acc\_ID），

sub（withdraw\_ID）→withdraw (withdraw\_ID)

* 表 take 的列名：cus\_id, take\_date, withdraw\_id,

主键：withdraw\_id,， ；外键：take (cus\_id)→account（acc\_ID），

take（withdraw\_ID）→withdraw (withdraw\_ID)

* 表 withdraw 的列名：ID, amount,

主键：ID

函数依赖：

1. 表 account：balance→acc\_ID

2. 表 cust\_banker ：cus\_id, employee\_Id→type

3. 表 customer：cus\_ID→name, cus\_ID, cus\_tel；cus\_tel→name

4. 表 depositor：cus\_id→account\_id, deposit\_Date；

account\_id→deposit\_Date，cus\_id

5. 出于employee（id,name,tel,depart,start\_time）属性较多，为了起到展示根据函数依赖分解表以提高数据库效率的示范，我将employ分为三个表employee1，employee2，employee3。Employee中存在的函数依赖是：tel→id,name,start\_time; id,name→start\_time,depart; id→start\_time; id→tel;

通过以下步骤将employee实体集变化为满足3NF范式的关系模式：

一是求出employee的canonical cover，为tel→id,name; id→time,depart; time→tel；

二是将上述三个函数依赖左右两侧的属性合并为一个关系模式，即R1{tel,id,name}, R2{ id,time,depart }, R3{ time,tel },

三R1，R2中含有候选码id，且R1，R2，R3没有包含关系

四employee被分解为{ tel,id,name }，{ id,time,depart }，{ time,tel }。

6. 表payment：

7. 表sub：withdraw\_ID→amount, account\_ID

8. 表 take：withdraw\_id→cus\_id, take\_date

9. 表 withdraw：ID→amount。

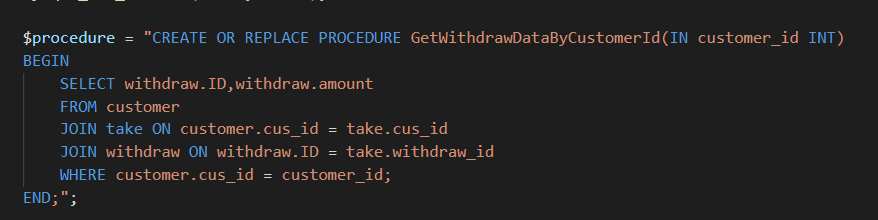
5. 数据库建立

通过在php数据库里面选定指定数据库（本项目数据库名字为bank\_dbs），在sql窗口中编写sql指令即可创建数据库。如下图所示。



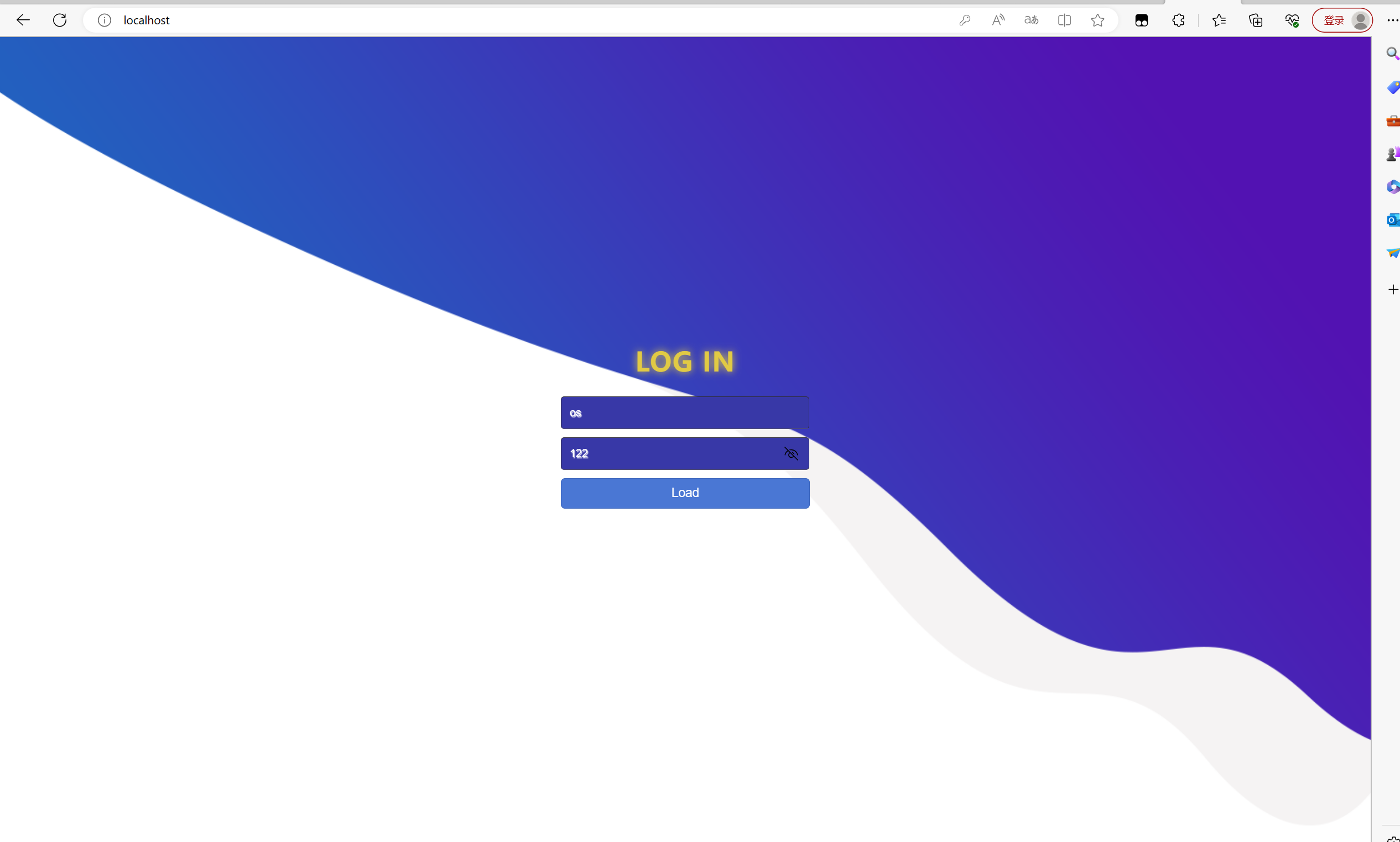
其他表以此类推。

设置预定义程序：为了便于用户查询指定客户（customer）的取钱记录（withdraw），我编写了预定义程序GetWithdrawDataByCustomerId（），并在前端设置了用户可以交互的按钮，方便了用户进行这类查询，以此做到可以使用预先定义的程序来执行我们经常执行的一些代码来提高工作效率。Sql代码如下：



6. 用例场景示范

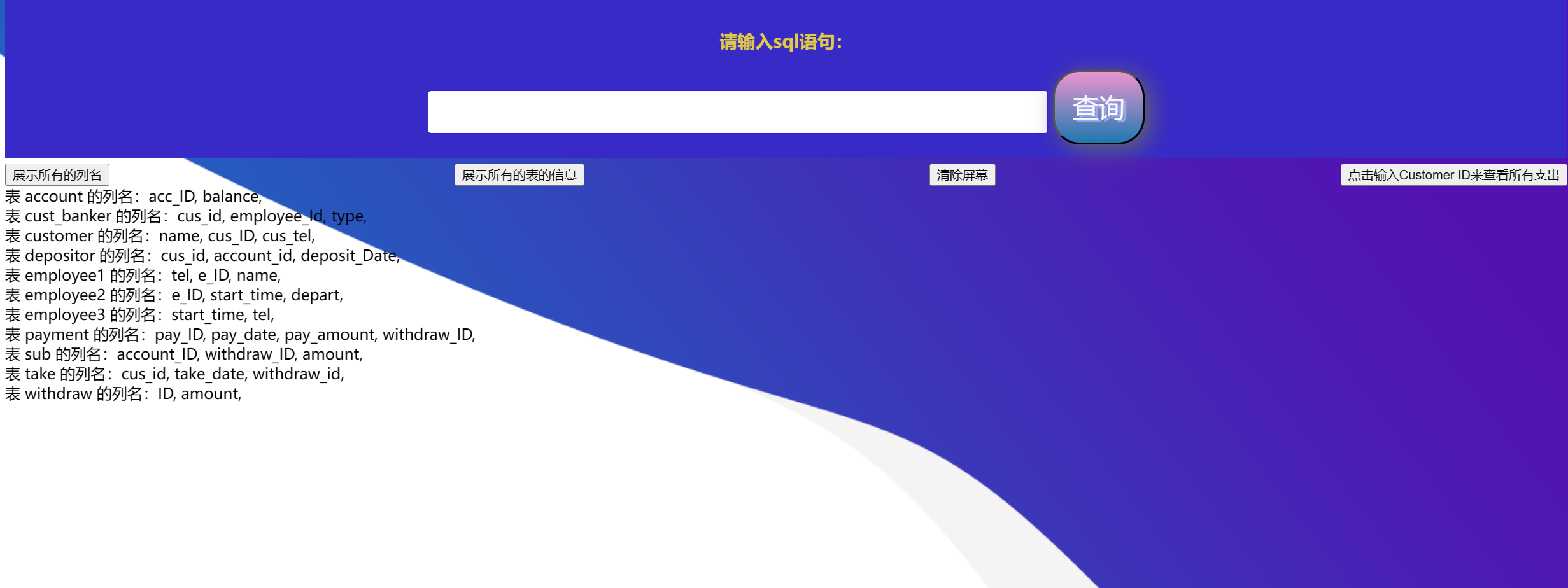
一用户登录：账号密码分别为os，122 ，暂未开放注册功能



二主界面如图所示

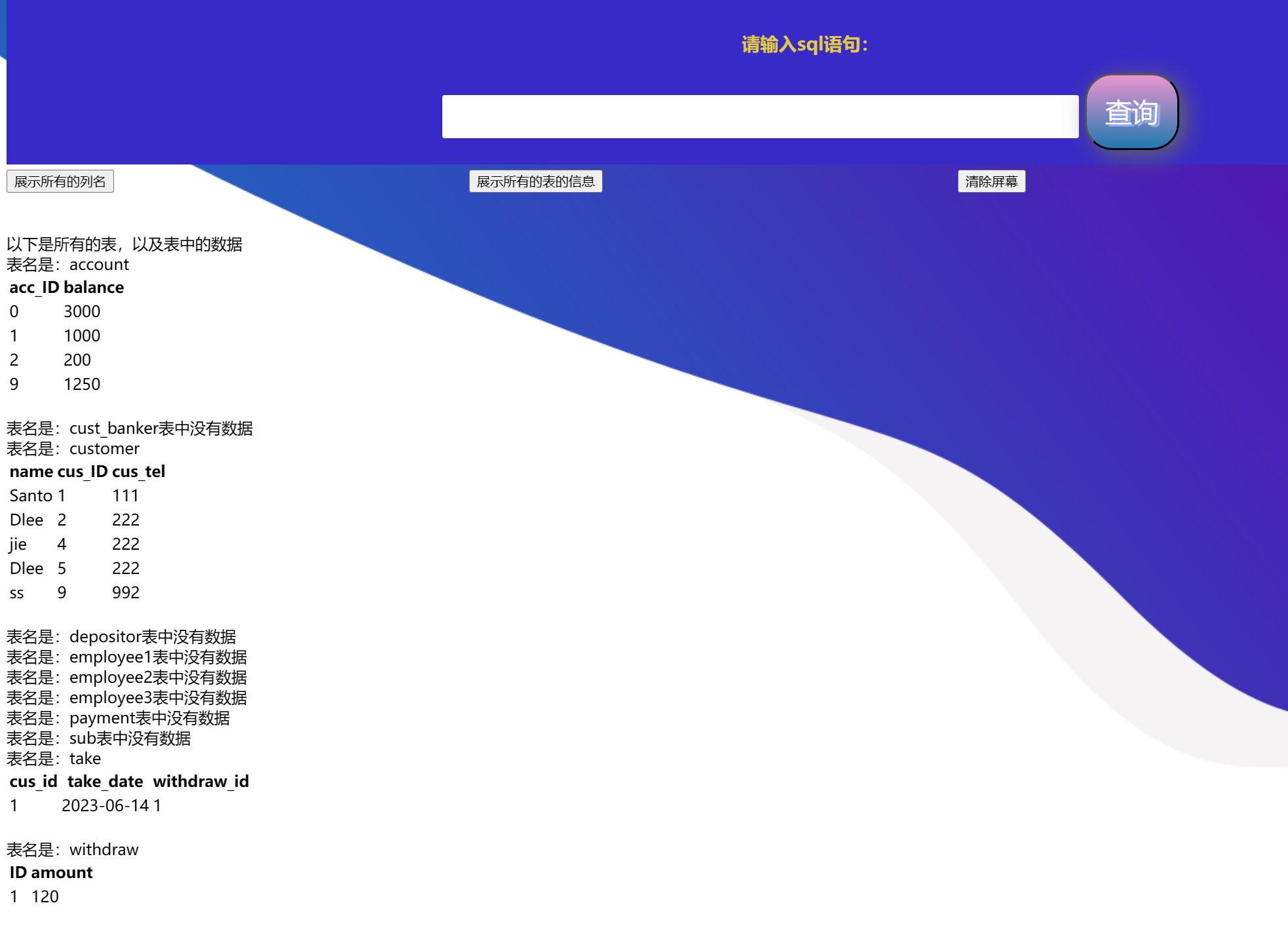


三点击“展示所有的列名”按钮（即第一个按钮）





四点击“展示所有的表的信息”按钮（即第二个按钮）





五清除屏幕（略）

六点击“输入Customer ID来查看所有支出”按钮

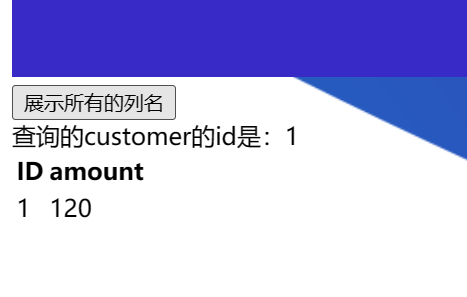












七用户在输入框中输入sql语句，则会展示修改的指定的表的全部内容（不含有表名的内容则不会展示）



7. 总结

这是一个十分简单的银行数据管理系统，实现了简单的前端界面，利用php连接前端和数据库，进行的一系列操作。虽然十分简单，但也增加了我对数据实际应用的体会，以及前端开发的经验。但值得注意的是，这个数据库虽然简单，但却也有很多值得深究的关键点，以及很多我没有完善的地方，例如大部分的表我都没有赋值，只对其中几个表进行了测试。这其中必然还会有很多我没有测试到的点。没有完善或者有bug的地方例如我假设customer和account是一对一的关系，但实际上并非如此，再例如我在尝试为withdraw这个表添加一个trigger触发器的时候发现我要联系acc\_id 和cus\_id 会遇到非常复杂的join问题。



后记：

由于xmapp提供了完整的本地服务器（如果是这样的话），并且占用了80端口，那么通过内网穿透工具Cporal 让80端口暴露在公网上，那么就可以在公网上访问这个网站了。效果如下：

