由.NET和SQL的银行账户管理系统开发

一、系统开发设计思想

（1）系统应符合银行账户管理的规定，满足银行相关人员的日常工作所需，并在操作过程中达到直观、方便、实用且安全等要求。

（2）系统采用模块化程序设计方法，既能便于系统功能的各种组合和修改，又便于后期系统的维护与补充。

（3）系统应具有数据库维护功能，及时根据用户需求进行数据的添加、删除和修改等操作。

### 系统开发和运行环境

开发工具：Visual Studio 2010 旗舰版 、SQL Server 2012

运行环境：Windows 10 家庭版

### 二、 系统功能分析

系统功能分析是在系统开发的总体任务的基础上完成的。本系统需要完成的功能主要有以下方面：

\* 管理员信息管理

\* 用户信息修改、开户、销户

\* 办卡、挂失卡

\* 存款、取款

\* 取款机信息管理

\* 管理员与用户对信息的查询

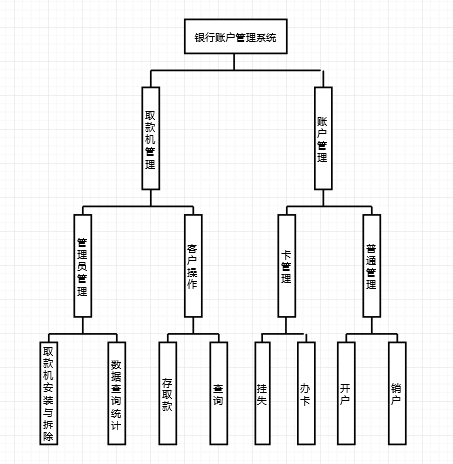
系统功能模块图：如

图 系统功能模块图

系统可以根据需要选择身份进行操作。若选择管理员身份，则经过登陆，确认为管理员身份后，能够对业务进行办理，并且能进行数据查询统计与修改。若选择客户身份，则通过登陆，可进行存取款和查询操作，并且可对管理员申请开户、销户、挂失和办卡操作。

## 三、数据库结构设计

### 3.1数据库概念结构设计

经过实地调研，进行需求分析后，可设计出能够满足用户需求的各种实体，以及他们之间的关系，为后面的逻辑结构设计打下基础。

根据设计规划，可得到本实例的实体有：管理员实体、用户实体、取款机实体和银行卡实体。

得到银行账户管理系统的E-R图，如图3-1所示。

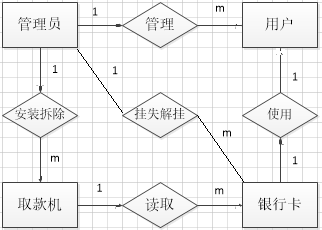


图3-1 银行账户管理系统的E-R图

对于一个实体型——管理员（管理员id,口令），其E-R图如图3-2所示。

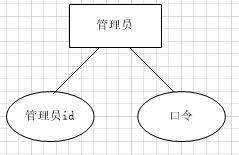


图3-2 管理员实体及其属性的E-R图

对于一个实体型——用户（开户时间，用户姓名，地址，身份证号，账号），其E-R图如图3-3所示。

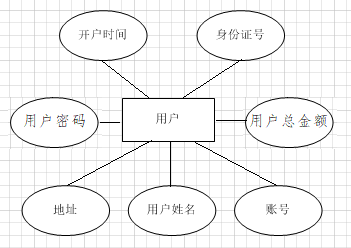


图3-3 用户实体及其属性的E-R图

对于一个实体型——银行卡（卡号，卡密码，金额，状态），其E-R图如图3-4所示。

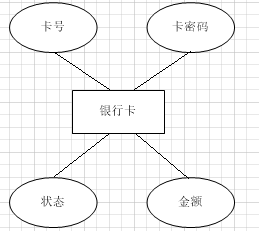


图3-4 银行卡实体及其属性的E-R图

对于一个实体型——取款机（取款机id,取款金额，取款时间，取款摘要;取出），其E-R图如图3-5所示。

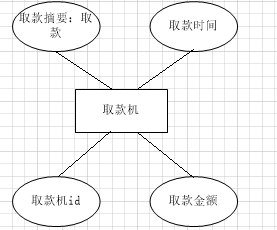


图3-5 取款机实体及其属性的E-R图

### 3.2数据库逻辑结构设计

#### 3.2.1关系模型

**一、实体和属性的转变**

由E-R图可知，本系统有4个实体，分别为管理员实体、用户实体、取款机实体和银行卡实体。转化为关系模式为：

管理员（管理员号，口令，）

用户（账号，用户身份证，用户姓名，用户密码，地址，开户时间，用户总金额，管理员号）

取款机（取款机ID号，卡号，取款时间，取款摘要：取款，取款金额，安装地址，安装时间，拆除时间，管理员号）

银行卡（卡号，卡密码，总金额，状态，管理员号）

以上关系范式都符合第一范式，即属性集中的每个属性的值域只包括原子项。

**三、关系模式的优化**

将上式第一范式优化为第二范式，第二范式即关系模式在属于第一范式的基础上，每个非主属性都完全函数依赖与任一候选码。则5个关系模式转化为：

管理员（管理员号，口令）

用户（账号，用户身份证，用户姓名，地址，管理员号） “管理员号”是用户关于管理员的外键

账户（账号，用户密码，开户时间，用户总金额） “账号”是用户关于账户的外键

取款机（取款机ID号，卡号，取款时间，取款摘要：取款，取款金额，安装地址，安装时间，拆除时间，管理员号） “管理员号”是取款机关于管理员的外键

银行卡（卡号，卡密码，总金额，状态，管理员号） “管理员号”是银行卡关于管理员的外键，“卡号”是银行卡关于取款机的外键

将第二范式优化为第三范式，第三范式即关系模式在属于第二范式的基础上，每个非主属性都不传递函数依赖于任一候选码。则5个关系模式转化为：

管理员（管理员号，口令）

用户（账号，用户姓名，用户身份证，地址，管理员号） “管理员号”是用户关于管理员的外键

账户（账号，用户密码，开户时间，用户总金额） “账号”是用户关于用户的外码

取款机（取款机ID号，卡号，取款时间，取款摘要：取款，取款金额，安装地址，安装时间，拆除时间，管理员号） “管理员号”是取款机关于管理员的外键

银行卡（卡号，卡密码，总金额，状态，管理员号） “管理员号”是银行卡关于管理员的外键，“卡号”是银行卡关于取款机的外键

由上式范式优化可得最终的关系模式为：

管理员（管理员号，口令）

用户（账号，用户姓名，用户身份证，地址，管理员号） “管理员号”是用户关于管理员的外键

账户（账号，用户密码，开户时间，用户总金额） “账号”是用户关于用户的外码

取款机（取款机ID号，卡号，取款时间，取款摘要：取款，取款金额，安装地址，安装时间，拆除时间，管理员号） “管理员号”是取款机关于管理员的外键

银行卡（卡号，卡密码，总金额，状态，管理员号） “管理员号”是银行卡关于管理员的外键，“卡号”是银行卡关于取款机的外键

一个好的关系模式要满足一些既定的标准，通过从第一范式到第二范式再到第三范式的优化，能消除一定的传递函数依赖，减少一定程度的数据冗余，以及插入异常和删除异常等问题。这些问题的存在也会影响系统的性能，容易造成系统效率低下。

#### 3.2.2数据表

根据E-R图转换后，需要建立5张相关的数据表：

(1)用户基本信息表。该表用于存放所有用户的基本信息。对应关系模式为U用户基本信息。该表具体内容见表3-1。



表3-1

(2)用户卡信息表。该表用于存放卡的信息。对应关系模式为Ca用户卡信息表。该表具体内容见表3-2。



表3-2

（3）账户基本信息表。该表用于存放取款机的基本信息。对应关系模式为Zh基本信息表。该表具体内容见表3-3。



表3-3

(4)取款机取款信息表。该表用于存放用户在取款机取款的基本信息。对应关系模式为ATM取款机存取款信息表。该表具体内容见表3-4。



表3-4

(5)管理系统的用户口令表。该表用于存放管理员的基本信息。对于关系模式为Tms管理系统的用户口令表。该表具体内容见表3-5。



表3-5

## 四、数据库的实现

在需求分析、概念结构设计的继承上得到数据库的逻辑结构之后，就可以在SQL Server Management Studio 2012中实现该逻辑结构。

实现数据库的逻辑结构的方式是选择SSMS，在该DBMS系统中利用其提供的DDL建立数据库结构。实现过程如下：

### 4.1数据表的建立

打开SQL Server Management Studio 2012，连接服务器，单击新建查询。

（1）输入图4-1的SQL语句创建数据库。

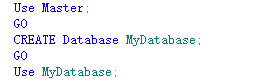


图4-1

（2）输入图4-2的SQL语句创建用户基本信息数据表。

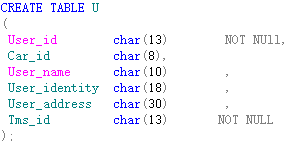


图4-2

主键：



外键：



（3）输入图4-3的SQL语句对U数据表进行数据插入。

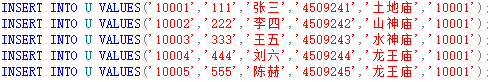


图4-3

（4）输入图4-4的SQL语句创建用户卡信息表。

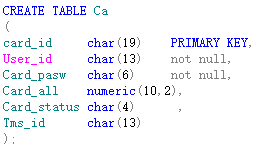


图4-4

外键：



（5）输入图4-5的SQL语句对Ca数据表进行数据插入。

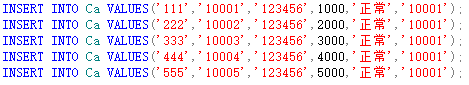


图4-5

（6）输入图4-6的SQL语句创建Zh基本信息表。

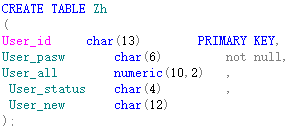


图4-6

外键：



（7）输入图4-7的SQL语句对Zh数据表进行数据插入。

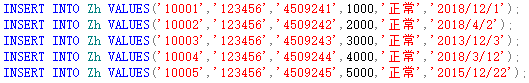


图4-7

（8）输入图4-8的SQL语句创建ATM取款机取款信息表。

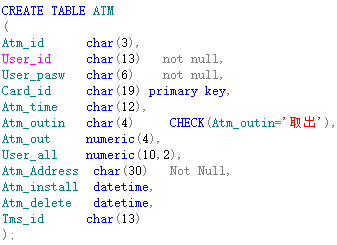


图4-8

外键：



（9）输入图4-9的SQL语句对ATM数据表进行数据插入。

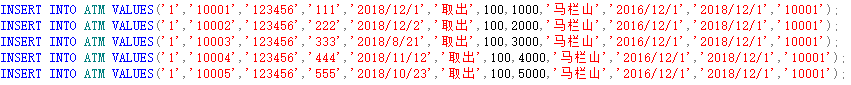


图4-9

（10）输入图4-10的SQL语句创建管理系统的用户口令表。

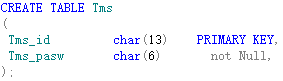


图4-10

（11）输入图4-11的SQL语句对Tms数据表进行数据插入。



图4-11

在数据库结构定义以后，通过DBMS提供的编译处理程序编译后即可形成实际可运行的数据库。

### 4.2索引的建立

为了提高对数据表的搜索效率，对U表进行创建组合索引和Ca表进行创键非聚集索引。

对于表U，按账号和用户名查询用户个人信息是通常使用的查询方式，因此对User\_id和User\_name的组合创建一个名为myIndex0的唯一组合索引。代码如下：



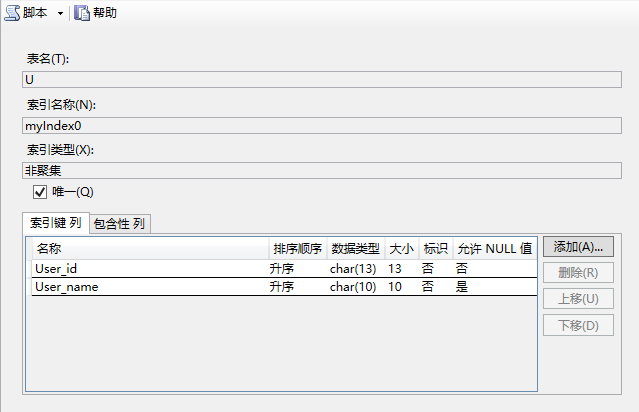


图4-12 实现组合索引myIndex0

对于表Ca,由于一个账户对应一个卡号，所以根据账号查询卡是查询方式之一，因此对User\_id创建一个名为myIndex1的非聚集索引。

实现代码如下：



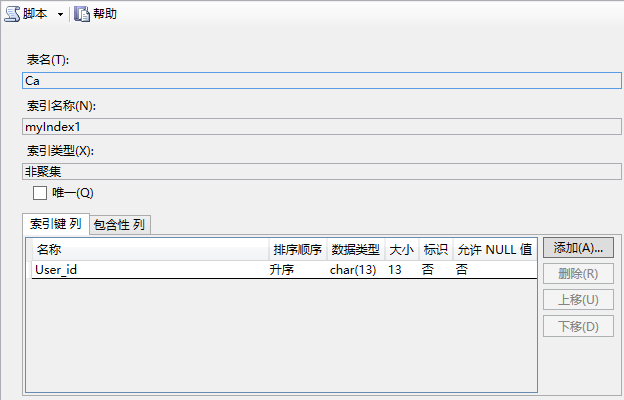
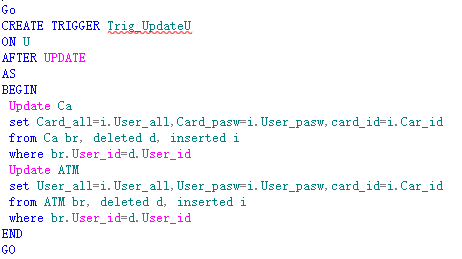


图4-13 实现非聚集索引myIndex1

### 4.3触发器的建立

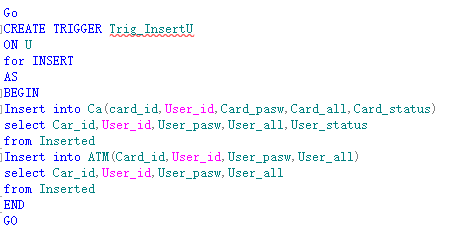
触发器是数据库服务器中发生事件时自动执行的一种特殊的存储过程。为了实现比较复杂的数据完整性规则，检查数据的有效性、实现对用户操作的数据状态的实时监控，对U表进行创建触发器操作，共创建了3个触发器。

1. 用于更新的AFTER触发器，当更新U表时，Ca表和ATM表也更新总金额，卡密码和卡号的信息，代码如下：



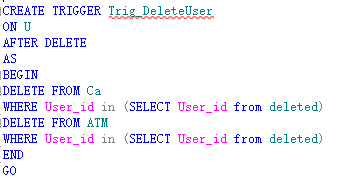
1. 用于插入的DML触发器，当U表插入一行新数据时，Ca表也插入卡号，账号，卡密码，总金额和状态，ATM表也插入卡号，账号，密码和总金额。

代码如下：



1. 用于删除的AFTER触发器，当U表的一条数据删除时，Ca表和ATM表中对应的相同账号行数据也删除。

代码如下：



## 五、各个功能模块的建立

建立功能模块的目的在于：一个是使用者能简单明了的掌握软件功能，另一个是未参加开发的人员也能轻松的进行维护。根据需求分析，将系统分为3个功能模块，分别为：账户管理、查询和ATM取款机模拟。

在进入各个功能模块前，应设计一个主界面。以下，将按系统开发顺序进行设计。

### 5.1系统主界面的设计

系统主界面的功能在于引导用户进入不同的功能模块。本系统为了简单起见，设计界面如图5-1所示。



图5-1 Form1

界面中包含3个按钮，分别为“账户管理”、“查询”和“取款机”，引导进入不同的功能模块。为了美观，在Form1中插入背景图。

实现步骤如下：

点击属性窗口，点击form1，在BackgroundImage属性处添加图片。在工具箱中添加3个button按键，在其属性窗口的Text属性处修改按键名。在“账户管理”按键中输入如图5-2代码，在“查询”按键中输入如图5-3代码，在“取款机”按键中输入如图5-4代码。主界面的设计到此结束。

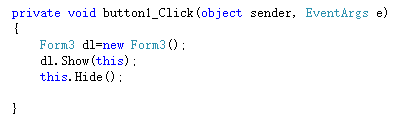


图5-2 “账户管理”按键

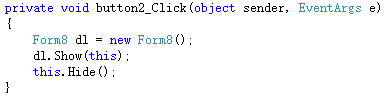


图5-3 “查询”按键

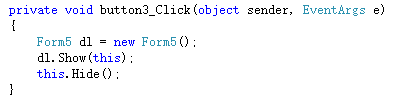
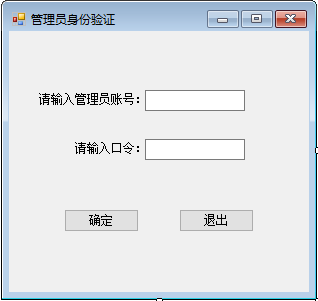


图5-4 “取款机”按键

### 5.2身份验证界面的设计

由于本系统的特殊性，在系统各处会涉及到身份验证。如管理员身份验证，用户身份验证等。

先设计进入“账户管理”的身份验证。单击“账户管理”，进入身份验证界面Form3。界面如图5-5所示。

 图5-5 Form 3

添加两个Label按键、两个textBox按键和两个button按键，在button按键中分别输入图5-6和5-7代码，完成对该界面的设计。

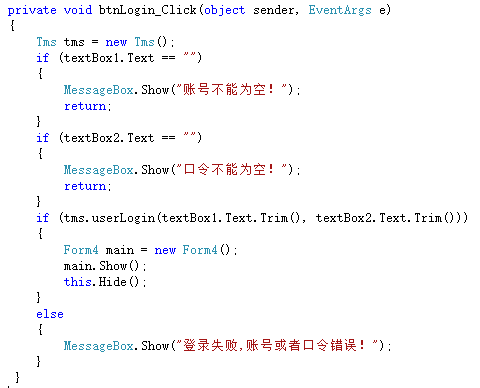


图5-6“确认”按键

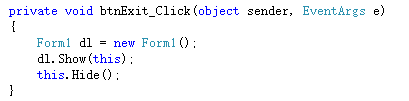


图5-7“退出”按键

再设计进入“查询”的身份验证界面。单击“查询”，进入图5-8界面，选择“管理员登陆”，进入身份验证界面Form2。界面如图5-5。界面设计如上管理员身份验证界面。

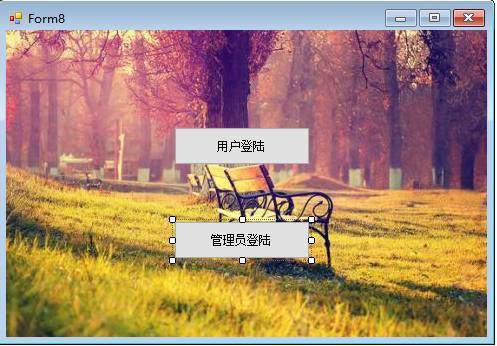


图5-8 Form2

用户身份验证界面：

如图5-8，单击“用户登陆”进入身份验证界面Form9，界面如图5-9所示。

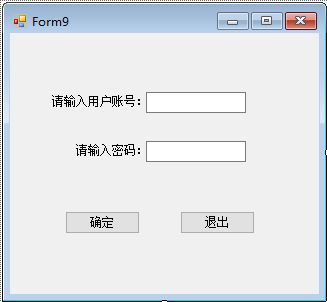


图5-9 Form9

添加两个Label按键、两个textBox按键和两个button按键，在button按键中分别输入图5-10和5-11代码，完成对该界面的设计。



图5-10 “确定”按键

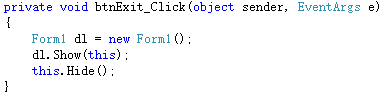
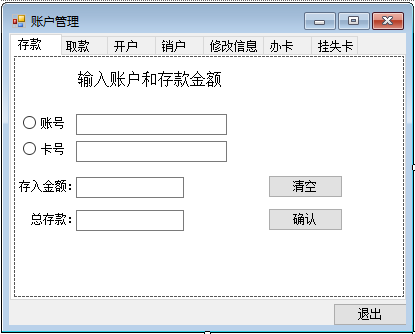


图5-11 “退出”按键

### 5.3账户管理界面的设计

本模块的功能在于管理员通过简单的操作，可以实现银行对账户管理的一般功能。如存款、取款、开户、销户、修改密码、办卡和挂失卡等等，内容丰富，较为繁琐。

当管理员通过身份验证后，可到达账户管理界面Form4。如图5-12。界面中由一个tabControl控件、三十五个textBox控件、两个radioButton按键、十九个button按键、42个Label控件和一个richTextBox控件组成。

图5-12 Form4

1. 存款

存款功能是账户管理中必不可少的一项功能，其模拟过程与现实生活一样，由管理员输入账户的账户或卡号和存款金额，点击“确认”按键，即可存入，并显示总金额。如图5-13所示。

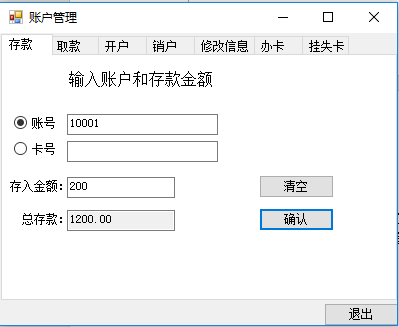


图5-13 存款

实现代码如下：

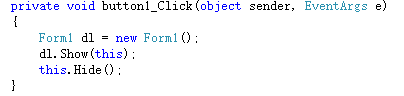


图5-14 “退出”按键

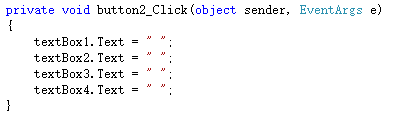


图5-15 “清空”按键



图5-16 “确认”按键

1. 取款

取款与存款操作类似，由管理员输入账户的账户或卡号和取款金额，点击“确认”按键，即可，并显示总金额。如图5-17。

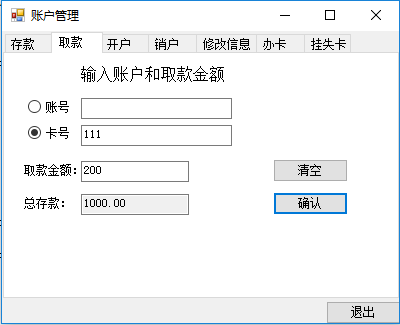


图5-17 取款

实现代码如下：

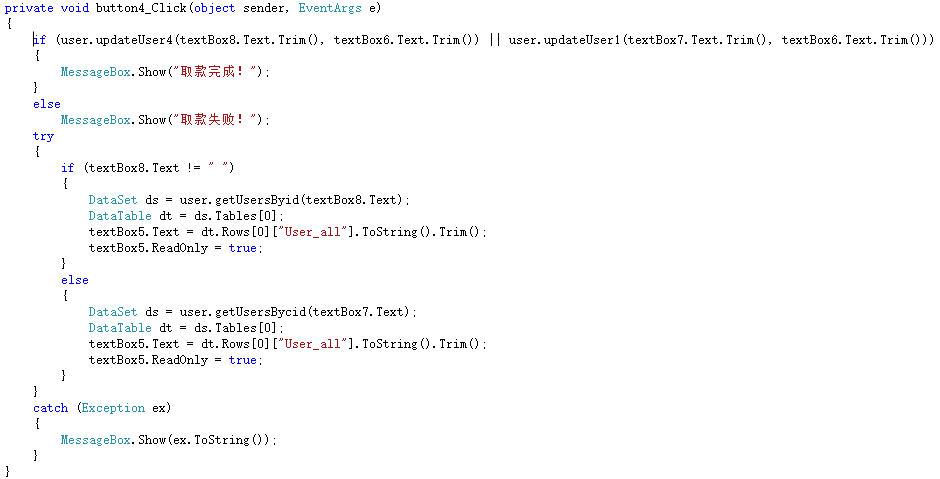


图5-18 “确认”按键

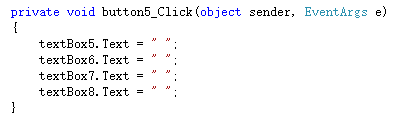
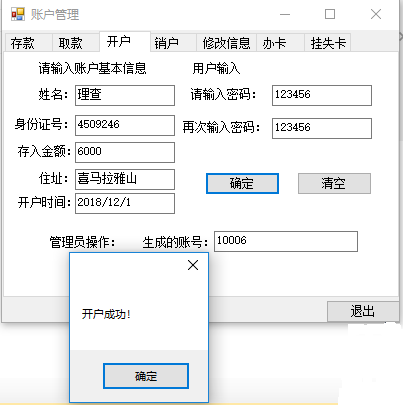


图5-19 “清空”按键

1. 开户

由管理员输入账户基本信息：姓名、身份证号、存入金额、住址和开户时间，用户输入密码，并确认密码，管理员生成新账号，即可开户。如图5-20。

图5-20 开户

代码如下：

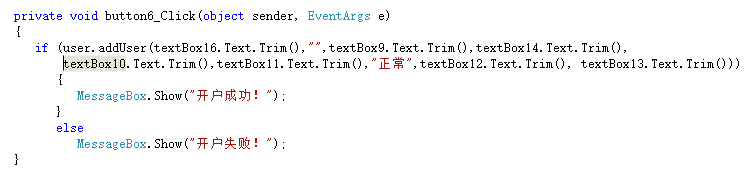


图5-21 “确认”按钮

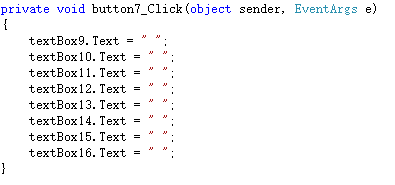


图5-22 “清空”按钮

1. 销户

由管理员输入要销户的账号，点击“确认”后，显示销户成功，并显示账户名和总金额。如图5-23。

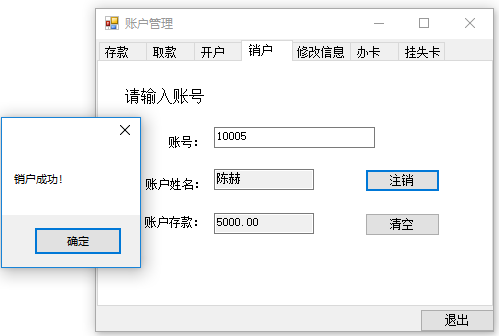


图5-23 销户

代码如下：



图5-24 “确认”按键

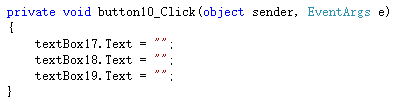


图5-25 “清空”按键

### 5.4查询界面的设计

点击主界面的查询按钮后，进入身份选择界面，如图5-8。选择用户登陆后，输入账号和用户名，即可查询到个人用户信息。如图5-35。



图5-35 用户查询

代码如下：

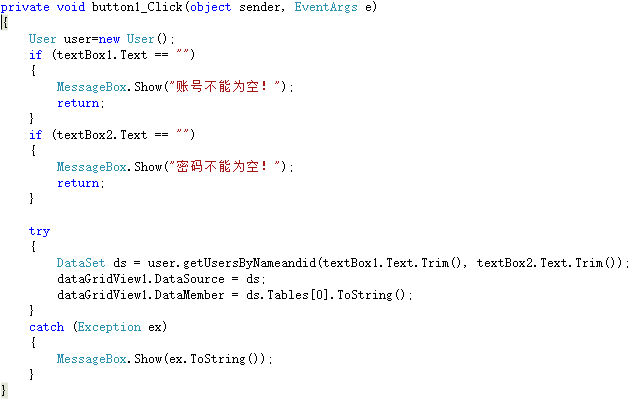


图5-36 “查询”按钮

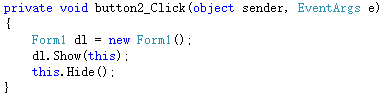


图5-37 “退出”按钮

选择管理员登陆后，可选择查询所需的表，点击“查询”后，即可查询。可在表中更改数据，更改后，点击“在表中更改数据”即可。如图5-38。

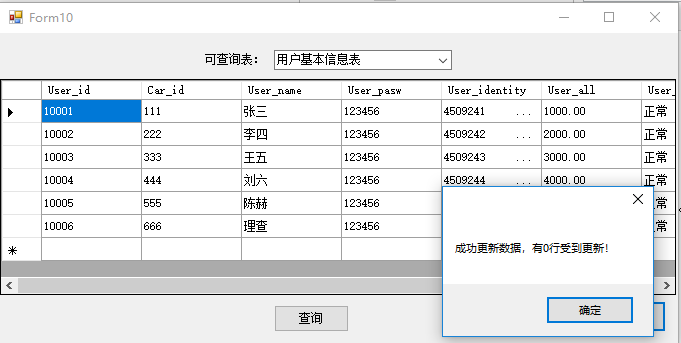


图5-38 管理员查询

代码如下：



图5-39 显示数据

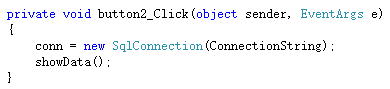


图5-40 “查询”按钮



图5-41 “在表中更新数据”按钮

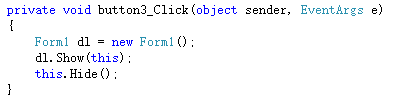


图5-42 “退出”按钮

### 5.5取款机界面的设计

在主界面中点击“取款机”，进入取款机界面，输入ATM机ID号，卡号和密码，点击“进入交易”按钮，显示现有存款，根据现有存款，可输入取款金额，点击“取出”按钮，取出现金。如图5-43。

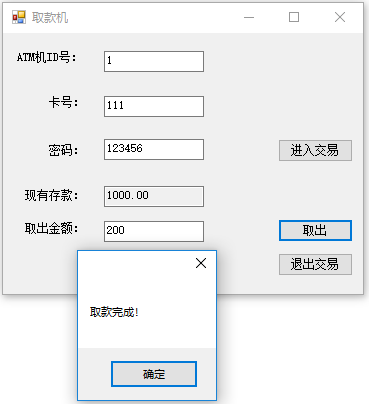


图5-43 取款机

代码如下：

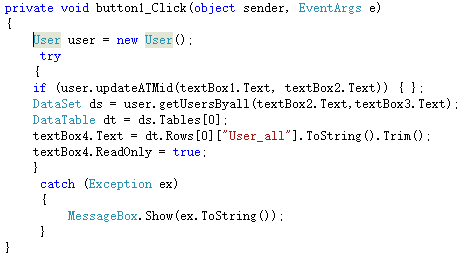


图5-44 “进入交易”按钮

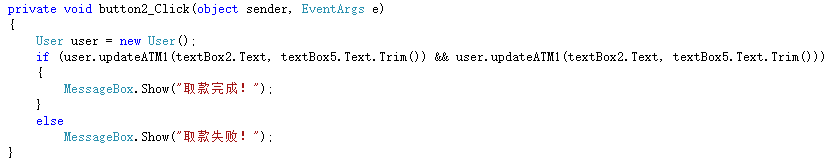


图5-45 “取出”按钮

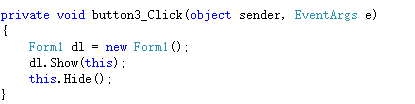


图5-46 “退出”按钮

## 参考文献

[1]《数据库原理与应用-基于SQL Server 2014》.蒙祖强、许嘉编著.清华大学出版社

[2]《SQL Server从入门到精通》.创客诚品 张保威、闰红岩编著.北京希望电子出版社

[3]《管理信息系统》.周山芙编著.中国人民大学出版社.2005年第二版

## 附录

SQLHelper类主要实现与数据库的连接，并提供了执行SQL语句的方法，而SQL语句的执行主要分为三类：

\*执行insert、delete和update语句，返回受影响的行数。

\*执行select语句，返回单值。

\*执行select语句，返回记录集。

SQLHelper类的关键代码如下：

class SQLHelper

{

private static string FLastError;

/// 最近一次异常信息

public static string LastError

{

get { return FLastError; }

}

/// 通用异常处理函数

private static void HandleException(Exception e)

{

if (e is SqlException)

{

FLastError = string.Format("在打开连接时出现连接级别的错误:{0}", e.Message);

}

else if (e is InvalidOperationException)

{

FLastError = e.Message;

}

else if (e is DBConcurrencyException)

{

FLastError = string.Format("尝试执行 INSERT、UPDATE 或 DELETE 语句，但没有记录受到影响:{0}", e.Message);

}

else

{

FLastError = string.Format("未预料的异常:", e.Message);

}

}

/// 获取数据库连接字符串、子类覆盖后自行修改

private static string GetConnectionString()

{

return "Data Source=.;Initial Catalog=MyDatabase;Integrated Security=True";

}

private static SqlConnection OpenConnection()

{

return new SqlConnection(GetConnectionString());

}

/// 无返回值的SQL语句执行

public static int ExecNonSQL(string ASql, params SqlParameter[] AParams)

{

using (SqlConnection Conn = new SqlConnection(GetConnectionString()))

{

try

{

//Open异常捕获

Conn.Open();

using (SqlCommand Cmd = Conn.CreateCommand())

{

Cmd.CommandText = ASql;

if (AParams != null)

{

foreach (SqlParameter param in AParams)

{

Cmd.Parameters.Add(param);

}

}

return Cmd.ExecuteNonQuery();

}

}

catch (Exception ex)

{

HandleException(ex);

}

}

return -1;

}

/// 无返回值的SQL语句执行

//执行select语句，返回单值

public static string ExecOneSQL(string ASql, params SqlParameter[] AParams)

{

using (SqlConnection Conn = new SqlConnection(GetConnectionString()))

{

try

{

//Open异常捕获

Conn.Open();

using (SqlCommand Cmd = Conn.CreateCommand())

{

Cmd.CommandText = ASql;

if (AParams != null)

{

foreach (SqlParameter param in AParams)

{

Cmd.Parameters.Add(param);

}

}

return Cmd.ExecuteScalar().ToString();

}

}

catch (Exception ex)

{

HandleException(ex);

}

}

return "1";

}

//执行select语句，返回记录集

public static DataSet ExecSQLByDataSet(string ASql, params SqlParameter[] AParams)

{

using (SqlConnection Conn = new SqlConnection(GetConnectionString()))

{

try

{

Conn.Open();

using (SqlCommand Cmd = Conn.CreateCommand())

{

Cmd.CommandText = ASql;

if (AParams != null)

{

foreach (SqlParameter param in AParams)

{

Cmd.Parameters.Add(param);

}

}

SqlDataAdapter Adapter = new SqlDataAdapter(Cmd);

DataSet Result = new DataSet();

Adapter.Fill(Result, "table");

return Result;

}

}

catch (Exception ex)

{

HandleException(ex);

}

}

return null;

}

}

}

User类主要用于对用户表、Ca表和ATM表进行增删查改功能，并提供用户的登录验证。

关键代码如下：

class User

{

//登录验证

public bool userLogin(string User\_id, string User\_pasw)

{

string cmd = "select count(\*) from U where User\_id=@User\_id and User\_pasw=@User\_pasw";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_id", User\_id),

new SqlParameter("User\_pasw", User\_pasw)

};

int ret = int.Parse(SQLHelper.ExecOneSQL(cmd, paras));

if (0 < ret)

{

return true;

}

return false;

}

//根据用户名和id进行数据库查询

public DataSet getUsersByNameandid(string User\_id,string User\_name)

{

string cmd = "select \* from U where User\_id like '%" + User\_id + "%'and User\_name like '%" + User\_name + "%'";

DataSet ds = SQLHelper.ExecSQLByDataSet(cmd,null);

return ds;

}

//删除指定用户

public bool deleteUser(string User\_id)

{

string cmd = "delete from U where User\_id=@User\_id";

SqlParameter[] paras = new SqlParameter[] { new SqlParameter("User\_id", User\_id) };

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//添加新用户

public bool addUser(string User\_id, string Car\_id, string User\_name, string User\_pasw, string User\_identity, string User\_all, string User\_status, string User\_address, string User\_new)

{

string cmd = "insert into U values(@User\_id,@Car\_id,@User\_name,@User\_pasw,@User\_identity,@User\_all,@User\_status,@User\_address,@User\_new)";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_id", User\_id),

new SqlParameter("Car\_id", Car\_id),

new SqlParameter("User\_name", User\_name),

new SqlParameter("User\_pasw", User\_pasw),

new SqlParameter("User\_identity", User\_identity),

new SqlParameter("User\_all", User\_all),

new SqlParameter("User\_status", User\_status),

new SqlParameter("User\_address", User\_address),

new SqlParameter("User\_new", User\_new)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//添加新卡

public bool addCard(string Car\_id, string User\_id, string Card\_pasw)

{

string cmd = "update U set Car\_id=@Car\_id where User\_id like '%" + User\_id + "%' and User\_pasw like '%" + Card\_pasw + "%' ";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("Car\_id", Car\_id),

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//挂失卡

public bool updateCard(string Card\_id, string Card\_pasw)

{

string cmd = "update Ca set Card\_status='挂失'where card\_id like '%" + Card\_id + "%' and Card\_pasw like '%" + Card\_pasw + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("Card\_pasw", Card\_pasw),

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//恢复卡

public bool updatereCard(string Card\_id, string Card\_pasw)

{

string cmd = "update Ca set Card\_status='正常'where card\_id like '%" + Card\_id + "%' and Card\_pasw like '%" + Card\_pasw + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("Card\_pasw", Card\_pasw),

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//更新用户密码

public bool updateUser(string User\_id,string User\_pasw)

{

string cmd = "update U set User\_pasw=@User\_pasw where User\_id like '%" + User\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_pasw", User\_pasw)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//更新ATM\_id

public bool updateATMid(string Atm\_id, string Card\_id)

{

string cmd = "update ATM set Atm\_id=@Atm\_id where Card\_id like '%" + Card\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("Card\_id", Card\_id)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//根据卡id和卡密码进行数据库查询

public DataSet getUsersByall(string Card\_id, string User\_pasw)

{

string cmd = "select \* from ATM where Card\_id like '%" + Card\_id + "%'and User\_pasw like '%" + User\_pasw + "%'";

DataSet ds = SQLHelper.ExecSQLByDataSet(cmd, null);

return ds;

}

//根据账号进行数据库查询

public DataSet getUsersByid(string User\_id)

{

string cmd = "select \* from U where User\_id like '%" + User\_id + "%'";

DataSet ds = SQLHelper.ExecSQLByDataSet(cmd, null);

return ds;

}

//根据卡号进行数据库查询

public DataSet getUsersBycid(string Card\_id)

{

string cmd = "select \* from U where Car\_id like '%" + Card\_id + "%'";

DataSet ds = SQLHelper.ExecSQLByDataSet(cmd, null);

return ds;

}

//更新ATM表信息

public bool updateATM1(string Card\_id, string Atm\_out)

{

string cmd = "update Atm set Atm\_out=@Atm\_out where Card\_id like '%" + Card\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("Atm\_out", Atm\_out)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//更新U表信息

public bool updateUser1(string Card\_id, string User\_all)

{

string cmd = "update U set User\_all=User\_all-@User\_all where Car\_id like '%" + Card\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_all", User\_all)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//更新U表信息

public bool updateUser2(string User\_id, string User\_all)

{

string cmd = "update U set User\_all=User\_all+@User\_all where User\_id like '%" + User\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_all", User\_all)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//更新U表信息

public bool updateUser3(string Card\_id, string User\_all)

{

string cmd = "update U set User\_all=User\_all+@User\_all where Car\_id like '%" + Card\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_all", User\_all)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

//更新U表信息

public bool updateUser4(string User\_id, string User\_all)

{

string cmd = "update U set User\_all=User\_all-@User\_all where User\_id like '%" + User\_id + "%'";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("User\_all", User\_all)

};

int ret = SQLHelper.ExecNonSQL(cmd, paras);

if (ret > 0) return true;

return false;

}

}

Tms类主要用于管理员登陆验证。

关键代码如下：

class Tms

{

//登录验证

public bool userLogin(string Tms\_id, string Tms\_pasw)

{

string cmd = "select count(\*) from Tms where Tms\_id=@Tms\_id and Tms\_pasw=@Tms\_pasw";

SqlParameter[] paras = new SqlParameter[] {

new SqlParameter("Tms\_id", Tms\_id),

new SqlParameter("Tms\_pasw", Tms\_pasw)

};

int ret = int.Parse(SQLHelper.ExecOneSQL(cmd, paras));

if (0 < ret)

{

return true;

}

return false;

}

}