**数据库课程设计报告**

**题目 二手书交易系统**

**姓 名 \_\_\_\_\_ 张俊帆\_\_\_\_**

**学 号 \_\_\_\_ 201610311104\_\_**

**指导教师 栾翠菊**

**学科(专业) 计算机科学与技术**

**所在学院 信息工程学院**

**提交日期 2018年12月20日**

**目 录**

[1. 概述 1](#_Toc20202)

[1.1. 研究背景 1](#_Toc21311)

[1.2. 需求 1](#_Toc7058)

[1.3. 现状 1](#_Toc21458)

[2. 需求分析 2](#_Toc985)

[2.1. 数据流图 2](#_Toc1054)

[2.1.1. 顶层数据流图 2](#_Toc16942)

[2.1.2. 第二层数据流图 2](#_Toc15251)

[2.2. 数据字典 3](#_Toc9239)

[3. 概念模型设计 4](#_Toc24105)

[3.1. 局部E-R图 4](#_Toc16392)

[3.2. 全局E-R图 6](#_Toc26967)

[3.3. 优化E-R图 6](#_Toc17170)

[4. 逻辑模型设计 7](#_Toc11678)

[4.1. 关系模型 7](#_Toc23732)

[4.2. 数据类型定义 8](#_Toc16756)

[5. 数据库设计 11](#_Toc17051)

[5.1. 基本表 11](#_Toc24735)

[5.2. 视图 13](#_Toc21277)

[5.3. 索引 14](#_Toc17205)

[6. 应用系统设计 15](#_Toc21669)

[6.1. SqlHelper类 15](#_Toc4578)

[6.1.1. 用于数据库连接的函数GetConn 15](#_Toc367)

[6.1.2. 执行任意SQL语句的函数ExecuteNonQuery 15](#_Toc2345)

[6.1.3. 查询是否有对应信息的函数 15](#_Toc19017)

[6.1.4. 执行插入操作的函数InsertItem 16](#_Toc11179)

[6.1.5. 执行删除操作的函数DeleteItem 16](#_Toc20532)

[6.1.6. 执行更新操作的函数UpdatetItem 17](#_Toc9153)

[6.1.7. 其他重要的函数 17](#_Toc21518)

[6.2. 命名规范的设计 19](#_Toc14071)

[6.2.1. 窗体的命名规范 19](#_Toc5309)

[6.2.2. 控件的命名规范 20](#_Toc538)

[6.3. 复杂功能的设计 20](#_Toc5992)

[6.3.1. 禁止销售功能的设计 20](#_Toc11002)

[6.3.2. 取消订单功能的设计 22](#_Toc29379)

[7. 应用系统界面展示 23](#_Toc23832)

[7.1. 登录界面 23](#_Toc15742)

[7.2. 注册界面 23](#_Toc20684)

[7.3. 管理员界面 24](#_Toc334)

[7.3.1. 信息设置 24](#_Toc22708)

[7.3.2. 书目设置 25](#_Toc24368)

[7.3.3. 用户管理 25](#_Toc3936)

[7.3.4. 现存订单查询 26](#_Toc14038)

[7.3.5. 历史订单查询 27](#_Toc20375)

[7.4. 用户界面 27](#_Toc29273)

[7.4.1. 信息设置 28](#_Toc13237)

[7.4.2. 购物车 28](#_Toc24744)

[7.4.3. 商品查询 29](#_Toc31975)

[7.4.4. 发布商品 29](#_Toc7249)

[7.4.5. 管理商品 30](#_Toc2384)

[7.4.6. 现存订单查询(买家) 31](#_Toc10636)

[7.4.7. 现存订单查询(卖家) 31](#_Toc30817)

[7.4.8. 历史订单查询 32](#_Toc3511)

[8. 总结与展望 33](#_Toc22249)

[参考文献 34](#_Toc29605)

# 概述

## 研究背景

大学生群体规模庞大，但一直以来由于诸如课程安排或是教学时长所限等问题，不少教科书在当前学期使用过后，便长期搁置，造成了极大的资源浪费。现如今，不少商家看到如此商机，纷纷在校园建立二手书回收买卖的站点，但这些仅仅停留在线下。如果能将二手书交易搬到线上，这必将为同学们提供极大的便利。

## 需求

一个便捷的二手书交易系统，提供如下基本功能：

1. 提供管理员身份与用户身份
2. 允许管理员：修改个人信息、管理书目、管理用户、管理订单
3. 为用户添加买书功能：查询商品、添加至购物车、下单、付款、取消订单、查询购买生成的现存订单和历史订单
4. 为用户添加卖书功能：发布商品、管理商品、确认发货、取消订单、查询售卖生成的现存订单和历史订单

## 现状

暂无有关二手书交易的系统，但可以将各大电商系统作为参照，结合二手书的有关特性，创造出一个专门用于二手书籍交易

# 需求分析

## 数据流图

### 顶层数据流图

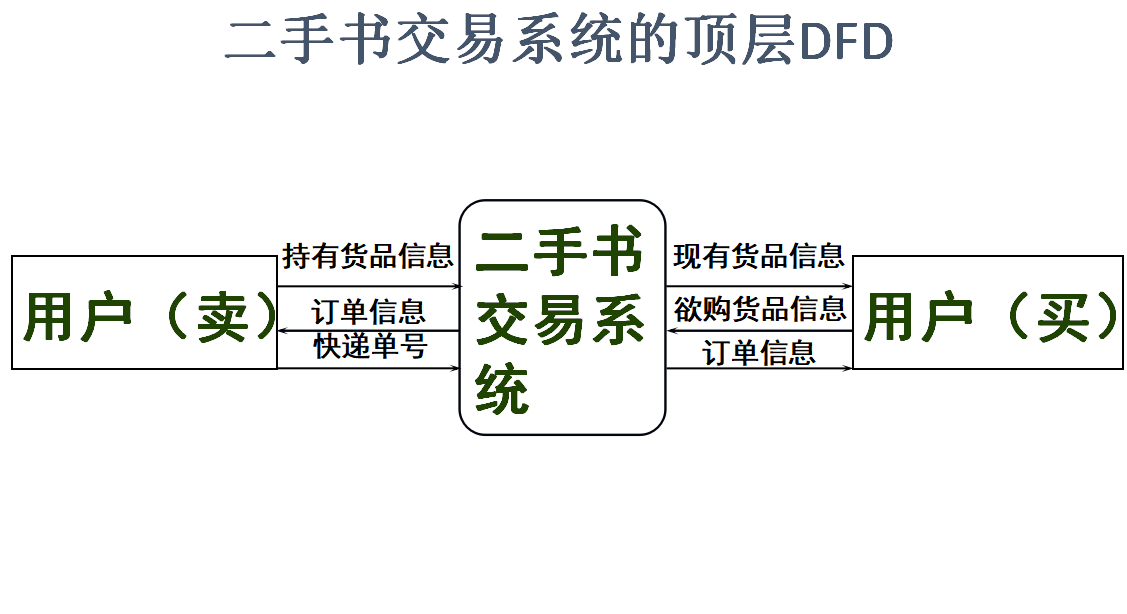


图 2.1 顶层数据流图

注：以下的“系统”均指代“二手书交易系统”

1. 用户(卖家)将持有货品信息发布到系统
2. 用户(买家)从系统获取现有货品信息
3. 用户(卖家)将已挑选的货品的信息输送到系统
4. 系统将生成订单，并发送给用户(卖家)
5. 用户卖家自行进行发货，并将快递单号发送给系统
6. 最终，系统将快递单号整合入订单信息中，并将订单信息发送给用户(买家)

### 第二层数据流图

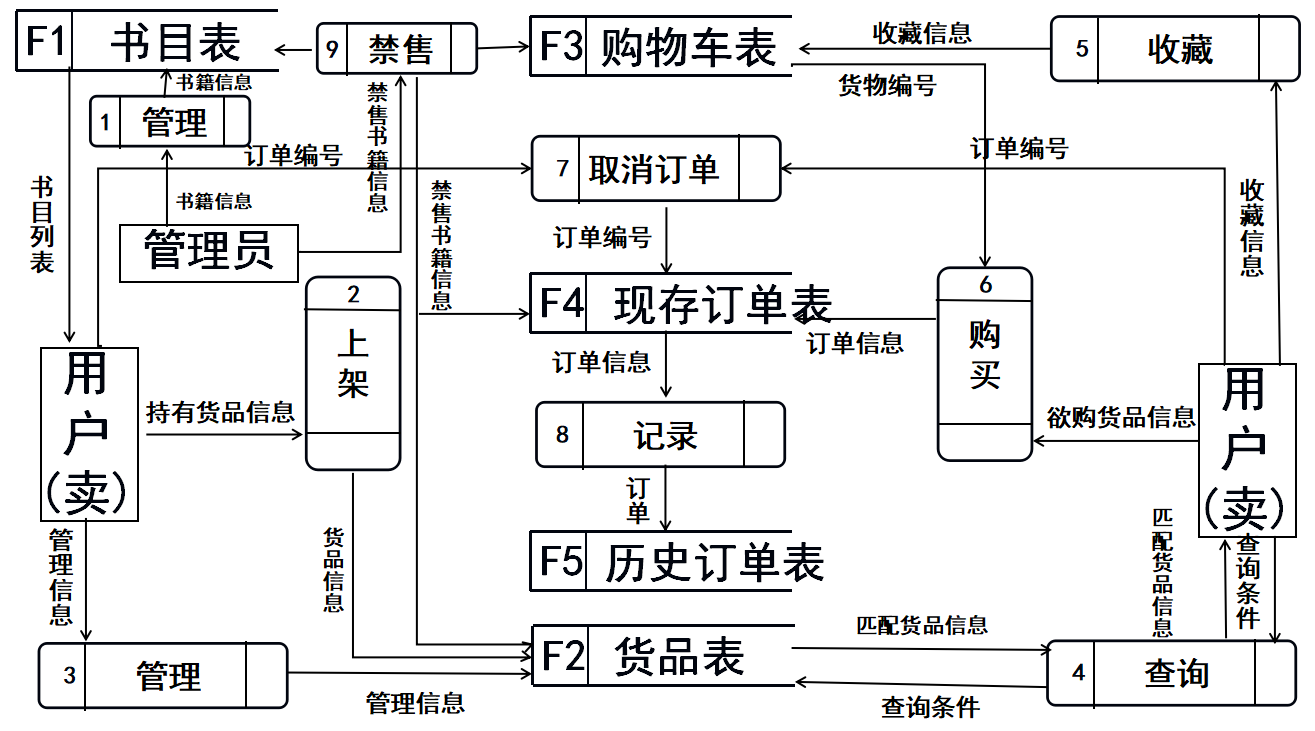


图 2.2 第二层数据流图

1. 管理员将可以通过”管理”操作，增删改书目表中的书籍信息
2. 用户(卖家)将持有货品信息发布，由”上架”操作将货品信息记录到”货品表”中
3. 用户(卖家)通过”管理”操作管理发布的货品，此记录将删改货品表中的信息
4. 用户(买家)通过”查询”操作，按选定的查询条件，查询到匹配的货品信息
5. 用户(买家)可通过”收藏”操作，将选定的货品收藏，此操作将其记录到购物车表中
6. 用户(买家)将购物车中的货物编号或刚刚查询到的货物编号，结合自身的用户编号输送到”购买操作”中，此操作将生成订单，并将其记录到”现存订单表”
7. 用户(买家/卖家)可将订单编号传送到”取消订单”操作，以取消现存订单，此操作将删除现存订单表中的信息
8. ”记录”操作将已完成的订单或被用户(买家/卖家)取消的订单记录到历史订单表中
9. 管理员可以通过”禁售”操作将书目表中的指定书籍设置成禁售；同时将使所有购物车表中对应的条目设置为失效状态；删除现存订单表中对应的状态处于”未付款”或”已付款”的订单(因为已发货的货品无法追回，所以不可取消”已发货”状态的订单)；再将删改货品表，若对应的货品的已订数量不为0(即还存在”已发货”状态的订单)，则保留此货品，并将其库存置零(注：对于已经禁售的货品，在”管理”操作中只能下架，不可更改货品信息)，这实现了对此禁售货品的屏蔽(注：用户查询不到库存为0的货品)；若对应的货品的已订数量为0，则直接删除此条目

## 数据字典

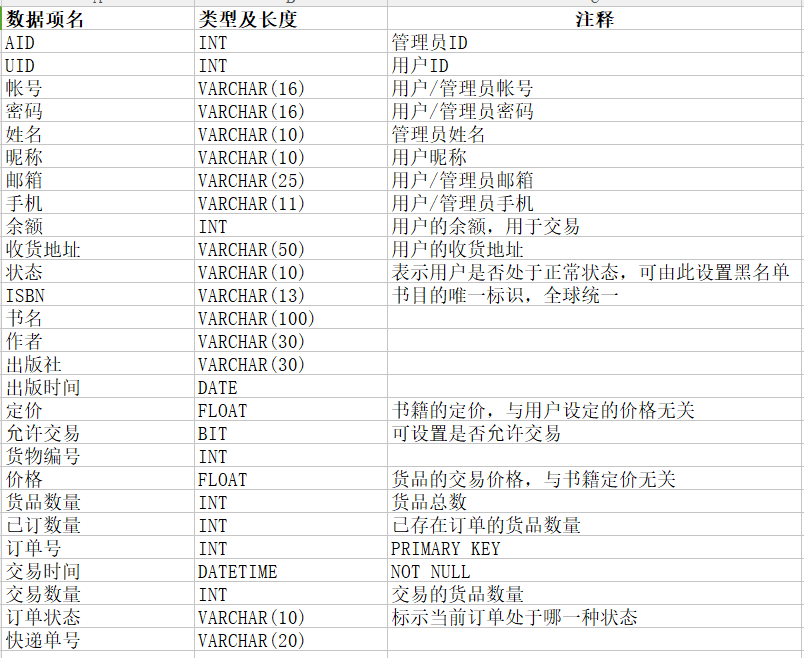


图 2.3 数据字典

# 概念模型设计

## 局部E-R图

1. 管理员实体及其属性

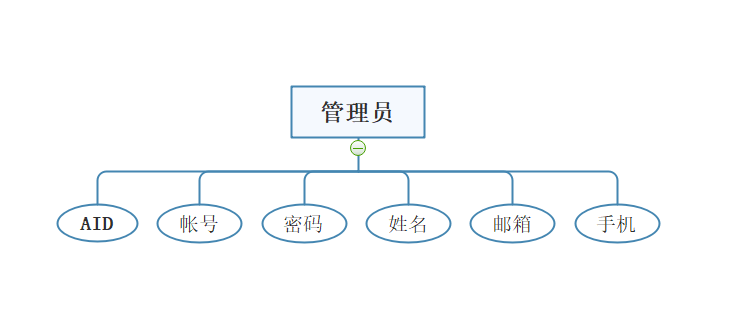


图 3.1 管理员实体E-R图

管理员身份主要用于管理书籍和用户。

管理员实体除了唯一标识AID、帐号和密码，只记录其他一些必要信息。

1. 用户实体及其属性

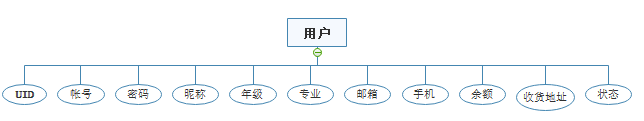


图 3.2 用户实体E-R图

用户实体具有其唯一标志UID、帐号和密码，以及其他一些必要信息。其中，”余额”用于买卖货品，”状态”用于用户管理(若为”黑名单”则禁止其登录)，”收货地址”可以为空，但在付款操作前必须填写

1. 书目实体及其属性

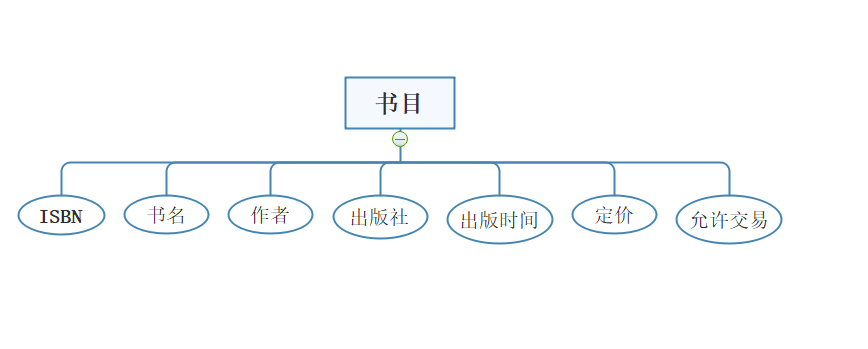


图 3.3 书目实体E-R图

”ISBN”为书目实体的唯一标识，”允许交易”用于禁止/允许销售操作。

1. 货品实体及其属性

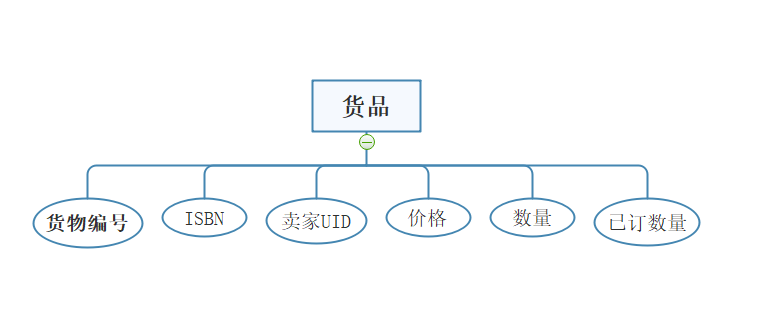


图 3.4 货品实体E-R图

“货物编号”为货品实体的唯一标识，”数量”为货品总数，”已订数量”为已产生订单的货品总数

1. 现存订单实体及其属性

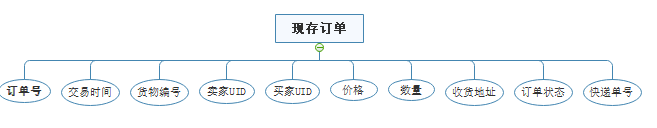


图 3.5 现存订单实体E-R图

“订单号”为现存订单实体的唯一标识，”订单状态”有”未付款”、”已付款”和”已发货”这三项，”快递单号”由卖方用户填写，在确认发货之前均为空。

现存订单将在确认收货操作、取消订单操作以及禁止销售操作之后被删除(除非状态为”已发货”)，并生成新的历史订单实体(如图3.6)

1. 历史订单实体及其属性

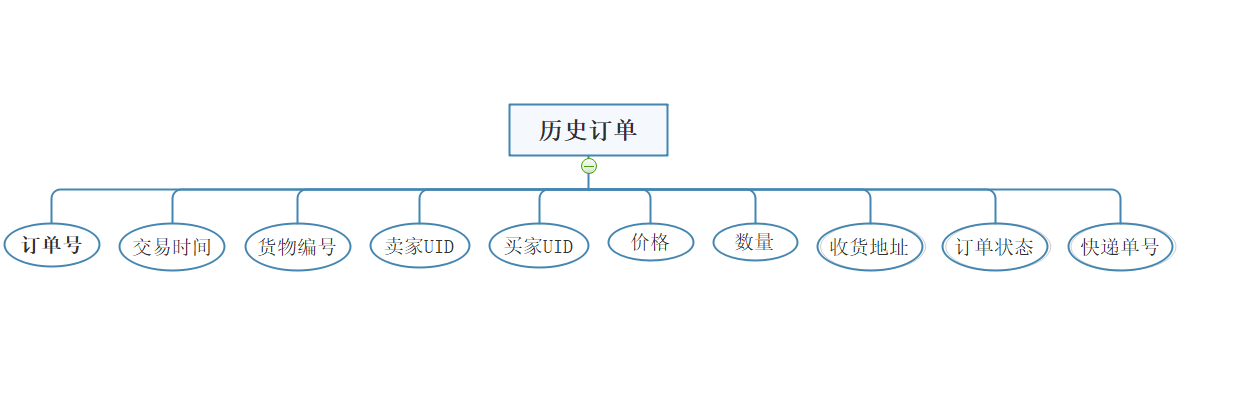


图 3.6 历史订单实体E-R图

“订单号”为现存订单实体的唯一标识，”订单状态”有”完成”、”买方取消”、”卖方取消”和”禁止销售”这四项。除非”订单状态”为”完成”，否则”快递单号”和”收货地址”必须为空。

## 全局E-R图

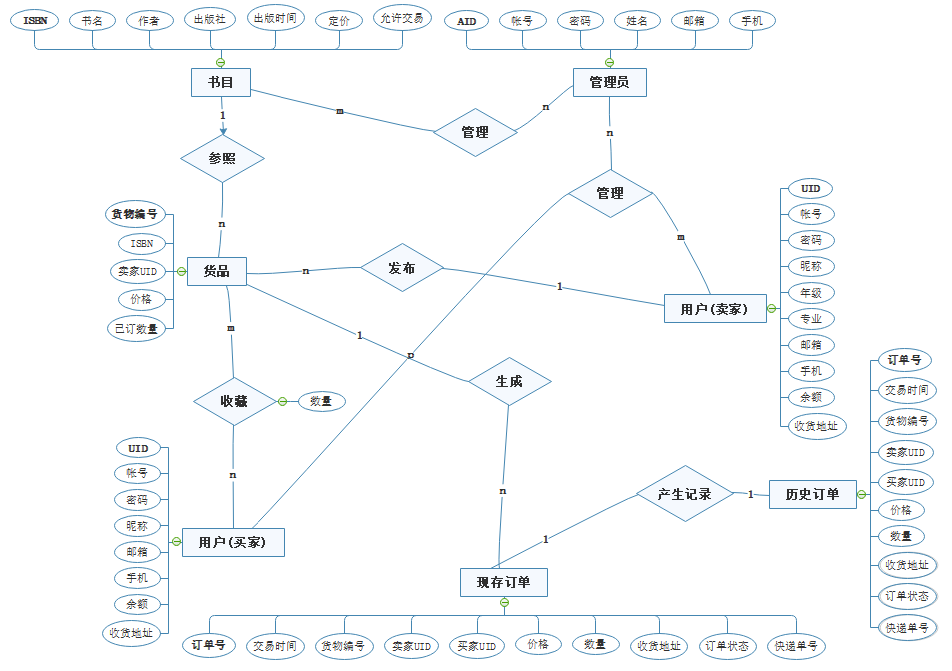


图 3.7 全局E-R图

## 优化E-R图

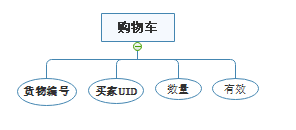


图 3.8 购物车实体E-R图

由”收藏”联系，新生成：购物车(**货物编号，买家UID**，数量，有效

# 逻辑模型设计

## 关系模型

注：加粗代表主键，斜体代表外键 

基本表：

管理员表(**AID**，帐号，密码，姓名，邮箱，手机)

用户表(**UID**，帐号，密码，昵称，年级，专业，邮箱，手机，余额，收货地址，状态)

书目表(**ISBN**，书名，作者，出版社，出版时间，定价，允许交易)

货品表(**货物编号**，*ISBN*，*卖家UID*，价格，数量，已订数量)

购物车表(**货物编号,UID**,数量)

现存订单表(**订单号**，交易时间，*货物编号，卖家UID，买家UID*，价格，数量，收货地址，订单状态，快递单号)

历史订单表(**订单号**，交易时间，货物编号，卖家UID，买家UID，价格，数量，收货地址，订单状态，快递单号)

权限：

管理员(帐号、密码)

用户表(帐号、密码)

参照关系：

货品表：

FOREIGN KEY (ISBN) REFERENCES 书目表(ISBN),

FOREIGN KEY (卖家UID) REFERENCES 用户表(UID)

现存订单表：

FOREIGN KEY (货物编号) REFERENCES 货品表(货物编号),

FOREIGN KEY (卖家UID) REFERENCES 用户表(UID),

FOREIGN KEY (买家UID) REFERENCES 用户表(UID)

约束：

用户表：

CHECK (年级 IN('大一','大二','大三','大四')),

CHECK (余额>=0),

CHECK (状态 IN ('正常','黑名单')),

货品表：

CHECK(已订数量<=数量),

现存订单表：

CHECK (卖家UID != 买家UID),

CHECK (数量 > 0),

CHECK (订单状态 IN('未付款','已付款','已发货')),

CHECK ((快递单号 IS NOT NULL AND 订单状态 = '已发货') OR (快递单号 IS NULL AND 订单状态 != '已发货')),

历史订单表：

CHECK (卖家UID != 买家UID),

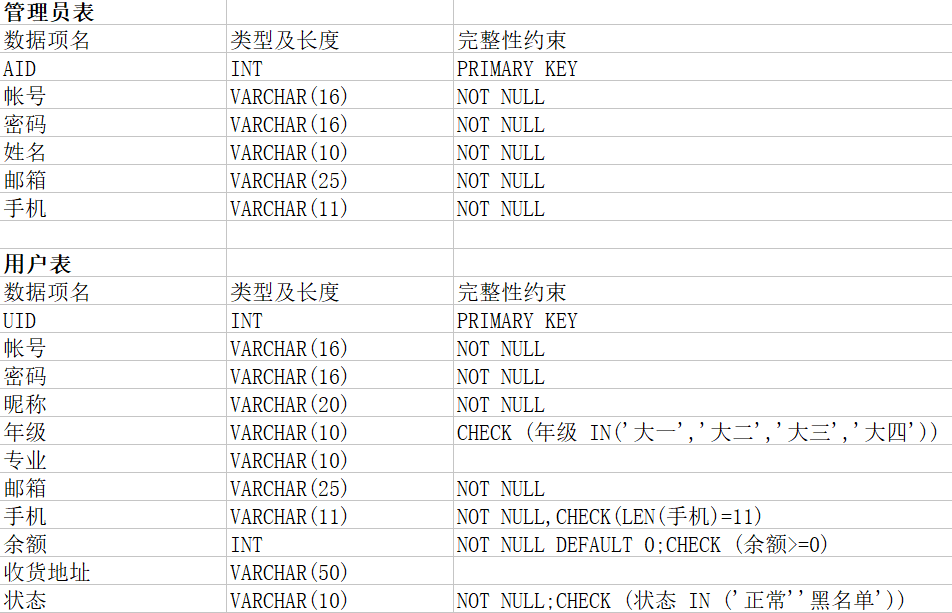
CHECK (数量 > 0),

CHECK (订单状态 IN('完成','买方取消','卖方取消','禁止销售')),

CHECK ((快递单号 IS NOT NULL AND 订单状态 = '完成') OR (快递单号 IS NULL AND 订单状态 != '完成')),

## 数据类型定义

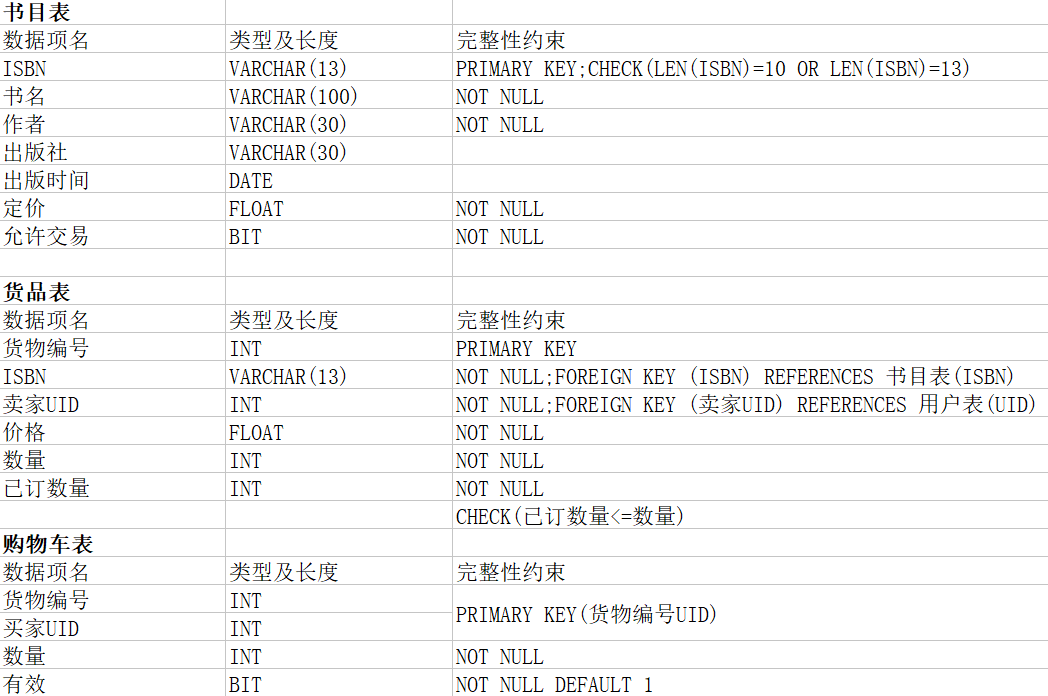
表 4.1 管理员表与用户表



“用户表”注：

1. 属性”年级”限定为”大一”、”大二”、”大三”、”大四”
2. ”手机”的长度限定为11
3. “余额”限定为大于等于0
4. 用户的”状态”属性若为”正常”则允许登录，否则禁止，注册时默认为”正常”，由管理员更改

表 4.2 书目表、货品表与购物车表



“书目表”注：

1. ”ISBN”的长度限定为10或13，是由于2007年1月1日前，ISBN由10位字组成;2007年1月1日起，新版ISBN由13位数字组成

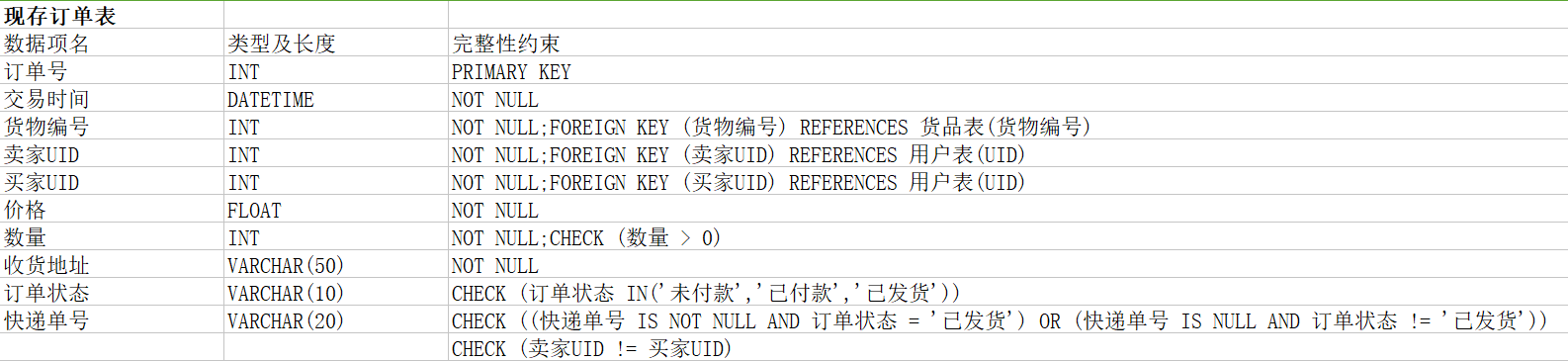
“货品表”注：

1. ”已订数量”为已被下单的货品总数，可由”数量”减去”已订数量”得到库存
2. ”已订数量”限定小于等于”数量”

“购物车”注：

1. ”货物编号”与”买家UID”为联合主键，表示了是哪一位用户收藏了哪一个货品
2. “有效”为表示这项收藏的有效情况，当对应的书籍被禁止销售或者该货品的下架时，”有效”将被切换为0(即：false)

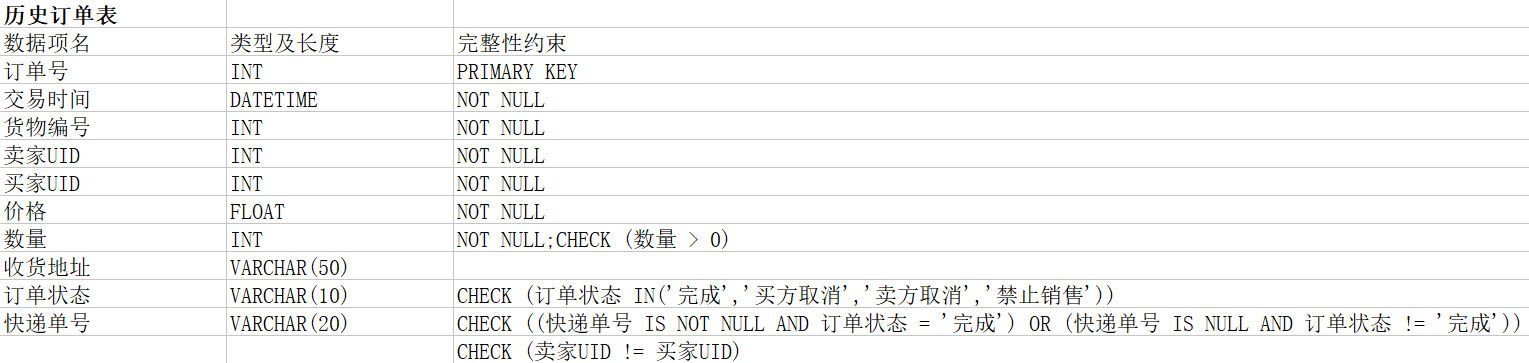
表 4.3 现存订单表



“现存订单表”注：

1. ”订单状态”限定为”未付款”、”已付款”和”已发货”
2. “卖家UID”与”买家UID”必须不相同
3. “订单状态”为”已发货”时，”快递单号”必须存在；“订单状态”不为”已发货”时，”快递单号”必须为空(发货前不可能有快递单号，要确认发货必须有快递单号)

表 4.4 历史订单表



注：

1. ”订单状态”限定为”完成”、”买方取消”、”卖方取消”、”禁止销售”
2. “完成”状态对应：订单由”未付款”→”已付款”→”已发货”直至被买方确认收货的正常完成的情况
3. “买方取消”和“卖方取消”状态分别对应：由于买方/卖方取消造成的订单撤销的情况
4. “禁止销售”状态对应：由于管理员禁售某些书籍，而造成的尚未发货的订单取消的情况(已发货的订单由于无法追回，故不可直接取消)

# 数据库设计

## 基本表

CREATE TABLE 书目表

(

ISBN VARCHAR(13) PRIMARY KEY,

书名 VARCHAR(100) NOT NULL,

作者 VARCHAR(30) NOT NULL,

出版社 VARCHAR(30),

出版时间 DATE,

定价 FLOAT NOT NULL,

允许交易 BIT NOT NULL,

CHECK(LEN(ISBN)=10 OR LEN(ISBN)=13)

--2007年1月1日前，ISBN由10位字组成;2007年1月1日起，新版ISBN由13位数字组成

)

CREATE TABLE 用户表

(

UID INT PRIMARY KEY,

帐号 VARCHAR(16) NOT NULL,

密码 VARCHAR(16) NOT NULL,

昵称 VARCHAR(20) NOT NULL,

年级 VARCHAR(10),

专业 VARCHAR(10),

邮箱 VARCHAR(25) NOT NULL,

手机 VARCHAR(11) NOT NULL,

余额 INT NOT NULL DEFAULT 0,

收货地址 VARCHAR(50),

状态 VARCHAR(10) NOT NULL,

CHECK (年级 IN('大一','大二','大三','大四')),

CHECK (余额>=0),

CHECK (状态 IN ('正常','黑名单')),

)

CREATE TABLE 管理员表

(

AID INT PRIMARY KEY,

帐号 VARCHAR(16) NOT NULL,

密码 VARCHAR(16) NOT NULL,

姓名 VARCHAR(10) NOT NULL,

邮箱 VARCHAR(25) NOT NULL,

手机 VARCHAR(11) NOT NULL,

)

CREATE TABLE 货品表

(

货物编号 INT PRIMARY KEY,

ISBN VARCHAR(13),

卖家UID INT,

价格 FLOAT NOT NULL,

数量 INT NOT NULL,

已订数量 INT NOT NULL,

CHECK(已订数量<=数量),

FOREIGN KEY (ISBN) REFERENCES 书目表(ISBN),

FOREIGN KEY (卖家UID) REFERENCES 用户表(UID)

)

CREATE TABLE 现存订单表

(

订单号 INT PRIMARY KEY,

交易时间 DATETIME NOT NULL,

货物编号 INT,

卖家UID INT,

买家UID INT,

价格 FLOAT NOT NULL,

数量 INT NOT NULL,

收货地址 VARCHAR(50) NOT NULL,

订单状态 VARCHAR(10),

快递单号 VARCHAR(20),

CHECK (卖家UID != 买家UID),

CHECK (数量 > 0),

CHECK (订单状态 IN('未付款','已付款','已发货')),

CHECK ((快递单号 IS NOT NULL AND 订单状态 = '已发货') OR (快递单号 IS NULL AND 订单状态 != '已发货')),

FOREIGN KEY (货物编号) REFERENCES 货品表(货物编号),

FOREIGN KEY (卖家UID) REFERENCES 用户表(UID),

FOREIGN KEY (买家UID) REFERENCES 用户表(UID)

)

CREATE TABLE 历史订单表

(

订单号 INT PRIMARY KEY,

交易时间 DATETIME NOT NULL,

货物编号 INT,

卖家UID INT,

买家UID INT,

价格 FLOAT NOT NULL,

数量 INT NOT NULL,

收货地址 VARCHAR(50),

订单状态 VARCHAR(10),

快递单号 VARCHAR(20),

CHECK (卖家UID != 买家UID),

CHECK (数量 > 0),

CHECK (订单状态 IN('完成','买方取消','卖方取消','禁止销售')),

CHECK ((快递单号 IS NOT NULL AND 订单状态 = '完成') OR (快递单号 IS NULL AND 订单状态 != '完成')),

)

CREATE TABLE 购物车表

(

货物编号 INT,

UID INT,

数量 INT NOT NULL,

有效 BIT NOT NULL,

PRIMARY KEY(货物编号,UID),

)

## 视图

CREATE VIEW 黑名单视图

AS

SELECT \*

FROM 用户表

WHERE 状态='黑名单';

CREATE VIEW 购物车视图

AS

SELECT 购物车表.货物编号,货品表.ISBN,书名,作者,出版社,出版时间,昵称 卖家昵称,卖家UID,CAST (定价 AS DECIMAL(18,2)) 原价,CAST (价格 AS DECIMAL(18,2)) 价格,购物车表.数量,货品表.数量-已订数量 库存,购物车表.有效,购物车表.UID

FROM 购物车表,货品表,书目表,用户表

WHERE 购物车表.货物编号 = 货品表.货物编号 AND 货品表.ISBN = 书目表.ISBN AND 卖家UID = 用户表.UID

CREATE VIEW 货品视图

AS

SELECT 货物编号,书名,作者,货品表.ISBN,出版社,出版时间,昵称 卖家昵称,UID 卖家ID,CAST (定价 AS DECIMAL(18,2)) 原价,CAST (价格 AS DECIMAL(18,2)) 价格,数量-已订数量 库存,允许交易

FROM 货品表,书目表,用户表

WHERE 货品表.ISBN = 书目表.ISBN AND 货品表.卖家UID = 用户表.UID;

CREATE VIEW 现存订单视图

AS

SELECT 订单号,货物编号,卖家UID,Y.昵称 卖家昵称,买家UID,X.昵称 买家昵称,收货地址,现存订单表.价格,数量,交易时间,快递单号,订单状态

FROM 现存订单表,(SELECT UID,昵称 FROM 用户表) X,(SELECT UID,昵称 FROM 用户表) Y

WHERE 现存订单表.买家UID = X.UID AND 现存订单表.卖家UID = Y.UID

CREATE VIEW 历史订单视图

AS

SELECT 订单号,货物编号,卖家UID,Y.昵称 卖家昵称,买家UID,X.昵称 买家昵称,收货地址,历史订单表.价格,数量,交易时间,快递单号,订单状态

FROM 历史订单表,(SELECT UID,昵称 FROM 用户表) X,(SELECT UID,昵称 FROM 用户表) Y

WHERE 历史订单表.买家UID = X.UID AND 历史订单表.卖家UID = Y.UID

## 索引

CREATE UNIQUE INDEX 书目表\_ISBN ON 书目表(ISBN);

CREATE INDEX 用户表\_UID ON 用户表(UID);

CREATE INDEX 管理员表\_AID ON 管理员表(AID);

CREATE INDEX 货品表\_货物编号 ON 货品表(货物编号);

CREATE INDEX 现存订单表\_订单号 ON 现存订单表(订单号);

CREATE INDEX 历史订单表\_订单号 ON 历史订单表(订单号);

CREATE INDEX 购物车表\_货物编号 ON 购物车表(货物编号);

# 应用系统设计

## SqlHelper类

在此设计了一个名为SqlHelper的类，专门用于执行与数据库相关的操作。以下将展示类中包含的各个方法。

### 用于数据库连接的函数GetConn

**//获取数据库连接 返回连接对象**

public static SqlConnection GetConn()

{

return new SqlConnection("server=.;database=二手书交易系统;integrated security=SSPI");

}

### 执行任意SQL语句的函数ExecuteNonQuery

**// 执行sql语句，返回受影响行数**

public static int ExecuteNonQuery(string sql)

{

int rows = -1;

using (SqlConnection conn = GetConn())

{

SqlCommand cmd = new SqlCommand(sql, conn);

conn.Open();

rows = cmd.ExecuteNonQuery();

}

return rows;

}

### 查询是否有对应信息的函数

#### 用于比对密码的函数CheckPassword

**//传入账号和密码，查询“用户表”或“管理员表”中是否有对应的信息，用于登录模块**

public static bool CheckPassword(string Table, string Account, string Password)

{

using (SqlConnection conn = GetConn())

{

string sql = string.Format("SELECT COUNT(\*) FROM {0} WHERE 帐号 ='{1}' AND 密码 ='{2}';", Table, Account, Password);

SqlCommand cmd = new SqlCommand(sql, conn);

conn.Open();

int obj = Convert.ToInt32(cmd.ExecuteScalar());

if (obj > 0)

return true;

else

return false;

}

}

#### 查询表中是否有对应信息的函数CheckExist

**//查询指定表中是否有对应信息**

public static bool CheckExist(string Table, string Where)

{

using (SqlConnection conn = GetConn())

{

string sql = string.Format("SELECT COUNT(\*) FROM {0} WHERE {1};", Table, Where);

SqlCommand cmd = new SqlCommand(sql, conn);

conn.Open();

int obj = Convert.ToInt32(cmd.ExecuteScalar());

if (obj > 0)

return true;

else

return false;

}

}

### 执行插入操作的函数InsertItem

**//执行sql语句，对指定的Table进行数据的插入操作**

public static void InsertItem(string Table, string Values)

{

using (SqlConnection conn = GetConn())

{

string sql = String.Format("INSERT INTO {0} VALUES {1}", Table, Values);

SqlHelper.ExecuteNonQuery(sql);

}

}

### 执行删除操作的函数DeleteItem

**//执行sql语句，对指定的Table进行数据的删除操作**

public static bool DeleteItem(string Table, string Where)

{

using (SqlConnection conn = GetConn())

{

string sql = String.Format("DELETE FROM {0} WHERE {1};", Table, Where);

int j = SqlHelper.ExecuteNonQuery(sql); //得到受影响的行数

if (j == 1)

return true;

else

{

return false;

}

}

}

### 执行更新操作的函数UpdatetItem

**//执行sql语句，对指定的Table进行数据的更新操作**

public static bool UpdatetItem(string Table, string Set, string Where)

{

using (SqlConnection conn = GetConn())

{

string sql = String.Format(" UPDATE {0} SET {1} WHERE {2};", Table, Set, Where);

int i = SqlHelper.ExecuteNonQuery(sql);

if (i == 1)

return true;

else

return false;

}

}

### 其他重要的函数

#### 用于将查询结果投射到DataGridView的函数

**//执行sql语句，并把查询结果用传入的dataGridView显示出来**

public static void ShowDataWithDataGirdView(DataGridView dataGridView, string sql)//定义函数

{

using (SqlConnection conn = GetConn())

{

SqlDataAdapter da = new SqlDataAdapter(sql, conn);

DataSet ds = new DataSet(); //创建数据集

conn.Open();

da.Fill(ds);

conn.Close();

dataGridView.DataSource = ds.Tables[0].DefaultView;

}

}

#### 用于查询相应结果的函数InquireItem

**//执行sql语句，返回查询的第一个结果，若查询失败，返回字符串'Not Found!'**

public static string InquireItem(string Table, string Inquire\_name, string Where)

{

using (SqlConnection conn = GetConn())

{

string sql = String.Format("SELECT {0} FROM {1} WHERE {2}", Inquire\_name, Table, Where);

DataSet ds = new DataSet(); //创建数据集

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = new SqlCommand(sql, conn);

da.Fill(ds);

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

SqlDataReader dr = cmd.ExecuteReader();

string temp = "Not Found!";

if (dr.Read())

temp = dr.GetValue(0).ToString();

conn.Close();

return temp;

}

}

#### 用于自动生成编号的函数GetID

**//执行sql语句，返回查询结果中的最大项，主要用于新注册用户时对其自动编号(即取相应的Id的最大值再加上一）**

public static int GetID(string Table, string item)

{

using (SqlConnection conn = GetConn())

{

string sql = String.Format("SELECT MAX({0}) FROM {1};", item, Table);

DataSet ds = new DataSet(); //创建数据集

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = new SqlCommand(sql, conn);

da.Fill(ds);

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

int Number\_new = 1;

try

{

Number\_new = Convert.ToInt32(dr.GetValue(0)) + 1;

}

catch { }//如果查不到就会报错，最终就会返回1

conn.Close();

return Number\_new;

}

else

{

conn.Close();

return 1;

}

}

}

## 命名规范的设计

### 窗体的命名规范



图 6.1 窗体的命名规范

如图所示，所有窗体均按照”数字1\_数字2\_数字3\_名称”的规范进行命名。其中：

数字1的0、1、2分别表示登录之前、以管理员身份登录和以用户身份登录

数字2则与这个窗体在主界面菜单栏上的次序相对应(如：图7.3和图7.9)

数字3则与菜单栏的下拉列表中的次序相对应。

### 控件的命名规范



图 6.2 控件命名的规范

本系统中，所有控件如上图所示方式命名：前半段标示其功能，后半段则标示控件类型

注：上图与历史订单查询界面(图7.8)相对应。

## 复杂功能的设计

### 禁止销售功能的设计

**//1.修改对应的"书目表(允许交易)"为false**

SqlHelper.UpdatetItem("书目表", "允许交易=0", string.Format("ISBN={0}", ISBN码textBox.Text));

**//2.修改对应的"购物车(有效)"为false**

SqlHelper.UpdatetItem("购物车表", "有效=0", string.Format("货物编号 IN (SELECT 购物车表.货物编号 FROM 购物车表,货品表 WHERE 购物车表.货物编号 = 货品表.货物编号 AND ISBN={0})", ISBN码textBox.Text));

**//3.将已付的款项，返还给买方**

using (SqlConnection conn = SqlHelper.GetConn())

{

conn.Open();

//得到 买家UID,要退回款项

string sql = string.Format("SELECT UID 买家UID,现存订单视图.价格\*现存订单视图.数量 要退回款项 FROM 用户表,现存订单视图,货品表 WHERE 现存订单视图.货物编号 = 货品表.货物编号 AND UID = 买家UID AND 订单状态 = '已付款' AND ISBN = '{0}'", ISBN码textBox.Text);

using (SqlDataAdapter da = new SqlDataAdapter(sql, conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

foreach (DataRow dataRow in dt.Rows)

{

int buyer\_UID = (int)dataRow["买家UID"];

double money\_to\_return = (double)dataRow["要退回款项"];

SqlHelper.Updatet\_ChangeNum("用户表", "余额", string.Format("UID={0}", buyer\_UID), true, money\_to\_return);

}

}

conn.Close();

}

**//4.由于要删除订单，所以要回退货品表的"已订数量"属性，并使"数量"为"已订数量"。**

**//注：这是为了使被下架的货品库存为0，这样就不会被查询到**

using (SqlConnection conn = SqlHelper.GetConn())

{

conn.Open();

//得到 货物编号,数量

string sql = string.Format("SELECT 现存订单视图.货物编号, 现存订单视图.数量 FROM 现存订单视图, 货品表, 书目表 WHERE 现存订单视图.货物编号 = 货品表.货物编号 AND 货品表.ISBN = 书目表.ISBN AND 允许交易 = 0 AND 订单状态 != '已发货' AND 货品表.ISBN = '{0}';", ISBN码textBox.Text);

using (SqlDataAdapter da = new SqlDataAdapter(sql, conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

foreach (DataRow dataRow in dt.Rows)

{

int ID = (int)dataRow["货物编号"];

int count\_ordered = (int)dataRow["数量"];

//回退货品表的"已订数量"属性

SqlHelper.Updatet\_ChangeNum("货品表", "已订数量", string.Format("货物编号={0}", ID), false, count\_ordered);

}

//使"数量"为"已订数量"

SqlHelper.UpdatetItem("货品表", "数量=已订数量", "货物编号 IN (SELECT 货物编号 FROM 货品视图 WHERE 允许交易 = 0)");

}

conn.Close();

}

**//5.将即将从"订单表"删除的项，记录到"历史订单表"中，并将"订单状态"设置为"禁止交易"**

SqlHelper.ExecuteNonQuery(string.Format("INSERT INTO 历史订单表(订单号,交易时间,货物编号,卖家UID,买家UID,价格,数量,收货地址,订单状态) SELECT 订单号,交易时间,货物编号,卖家UID,买家UID,价格,数量,NULL 收货地址,'禁止销售' 订单状态 FROM 现存订单表 WHERE 订单状态 != '已发货' AND 货物编号 IN( SELECT 货物编号 FROM 货品表 WHERE ISBN = {0})", ISBN码textBox.Text));

**//6.从订单表删除对应的"订单属性"不为"已发货"的项**

SqlHelper.DeleteItem("现存订单表", string.Format("订单状态!='已发货' AND 货物编号 IN(SELECT 货物编号 FROM 货品表 WHERE ISBN = {0})", ISBN码textBox.Text));

**//7.库存为0的货品将直接被删除**

**//注：不为0的继续保留，是因为那个货品有已发货的订单**

SqlHelper.DeleteItem("货品表", string.Format("已订数量=0 AND ISBN='{0}'", ISBN码textBox.Text));

### 取消订单功能的设计

**//1.把将要取消订单，记录到历史订单表中**

SqlHelper.InsertItem("历史订单表(订单号ding,交易时间,货物编号,卖家UID,买家UID,价格,数量,收货地址,快递单号,订单状态)", string.Format("({0},'{1}','{2}',{3},{4},{5},{6},'{7}',NULL,'买方取消')", 订单号textBox.Text, System.DateTime.Now, 货物编号textBox.Text, 卖家IDtextBox.Text, this.U\_id, 价格textBox.Text, 数量textBox.Text, 收货地址textBox.Text));

int count\_ordered = Convert.ToInt32(SqlHelper.InquireItem("货品表", "已订数量", string.Format("货物编号={0}", 货物编号textBox.Text)));

int count\_dingdan = Convert.ToInt32(数量textBox.Text);

**//2.将货品的"已订数量"减回来**

SqlHelper.Updatet\_ChangeNum("货品表", "已订数量", string.Format("货物编号={0}", 货物编号textBox.Text), false, count\_dingdan);

if (订单状态textBox.Text == "已付款")//若已付款，则退回余额

{

double price = Convert.ToInt32(价格textBox.Text);

SqlHelper.Updatet\_ChangeNum("用户表", "余额", string.Format("帐号='{0}'", U\_account), true, price \* count\_dingdan);

账户余额textBox.Text = SqlHelper.InquireItem("用户表", "余额", string.Format("帐号='{0}'", U\_account));

}

**//3.从"现存订单表"中删除该订单**

SqlHelper.DeleteItem("现存订单表", string.Format("订单号={0}", 订单号textBox.Text));

查询button\_Click(sender, e);

清空textBox();

# 应用系统界面展示

## 登录界面

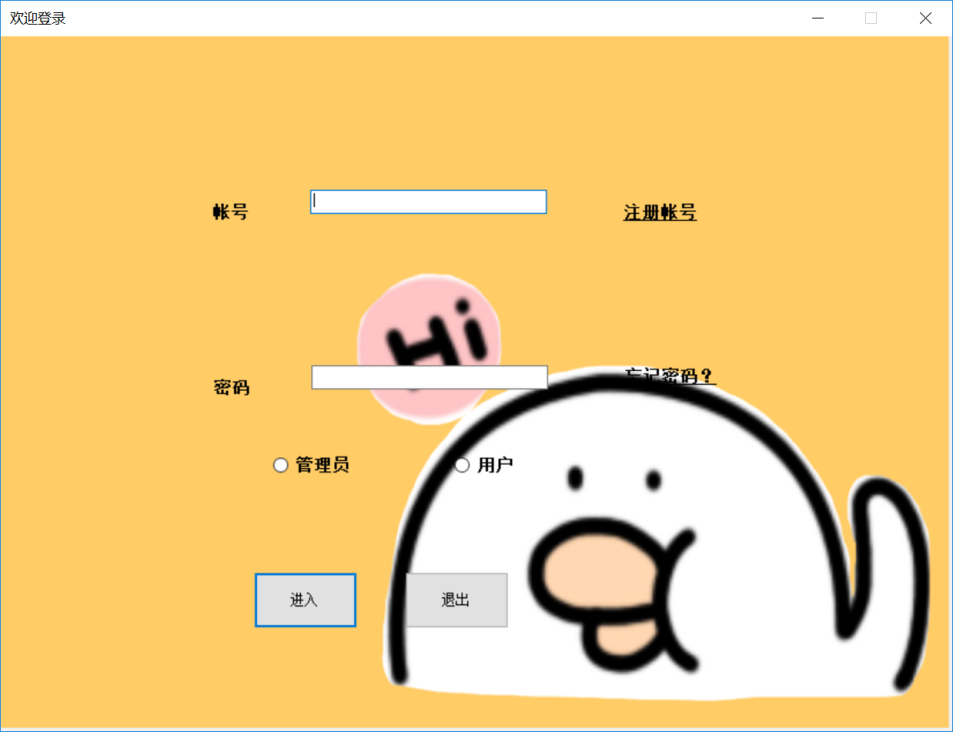


图 7.1 登录界面

点击”注册帐号”将跳转到如图6.2所示界面

## 注册界面



图 7.2 注册界面

注册成功后将取消后将回到登录界面(如图6.1)，注册之初用户余额为0。

## 管理员界面



图 7.3 管理员界面

在登录界面(如图6.1)以管理员身份登录时将进入此界面。在此界面可以跳转到各个主要操作界面。

### 信息设置



图 7.4 信息设置界面(管理员)

在此界面可以更改管理员密码、手机和邮箱，在此会对手机和邮箱格式做检验。

### 书目设置



图 7.5 书目设置界面

在此界面可以对书目进行增、改、查，也可以使本书禁止/允许交易

注：粉红色标注为已禁售书籍

### 用户管理



图 7.6 用户管理界面

在此界面可以对查询用户信息，也可以进行吊销和解禁操作

注：红色标注为黑名单状态的用户

### 现存订单查询



图 7.7 现存订单查询界面(管理员)

在此界面可查询所有现存的订单

注：

粉红色标注为未付款订单

黄色标注为已付款订单

绿色标注为已发货订单

### 历史订单查询

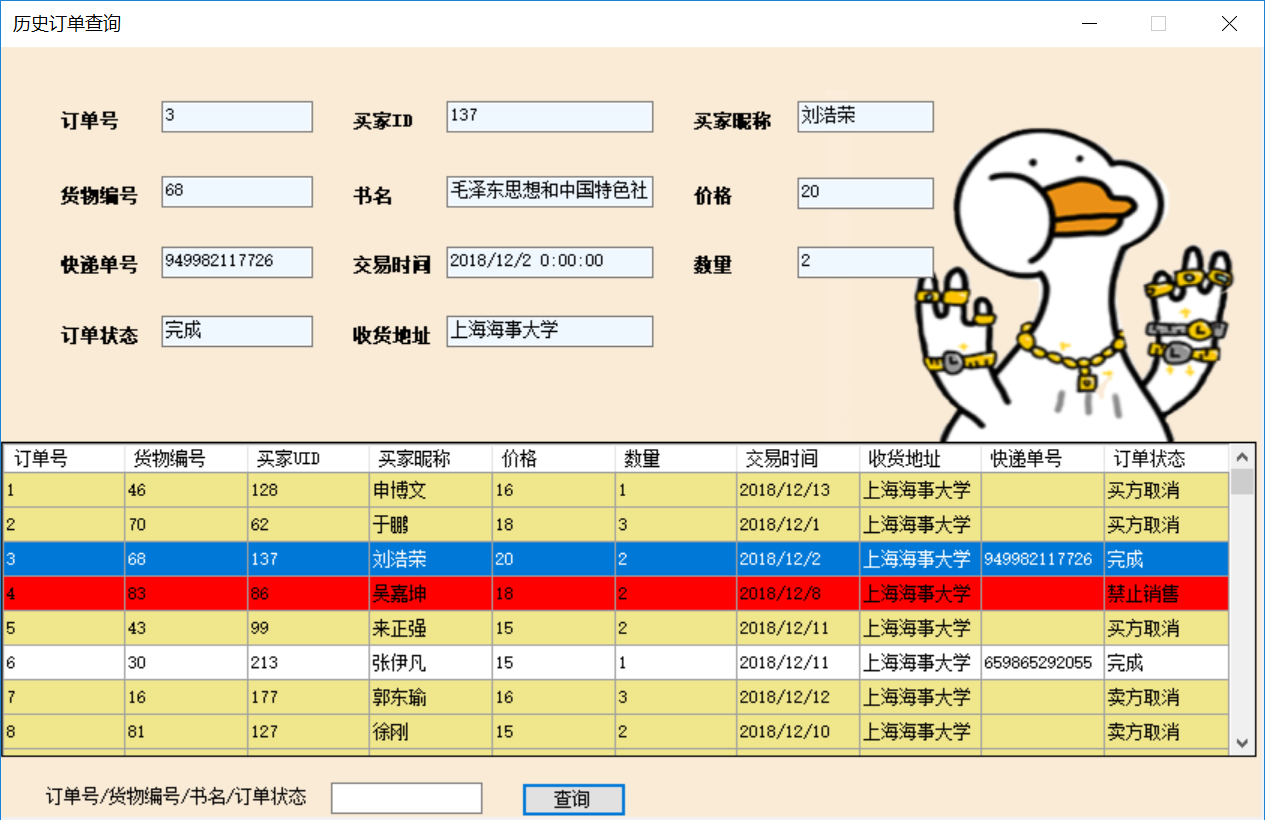


图 7.8 历史订单查询(管理员)

在此界面查询所有历史订单

注：

黄色标注为买方/卖方取消的订单

红色标注为禁售书籍而取消的订单

## 用户界面



图 7.9 用户界面

在登录界面(如图6.1)以管理员身份登录时将进入此界面。在此界面可以跳转到各个主要操作界面。

### 信息设置

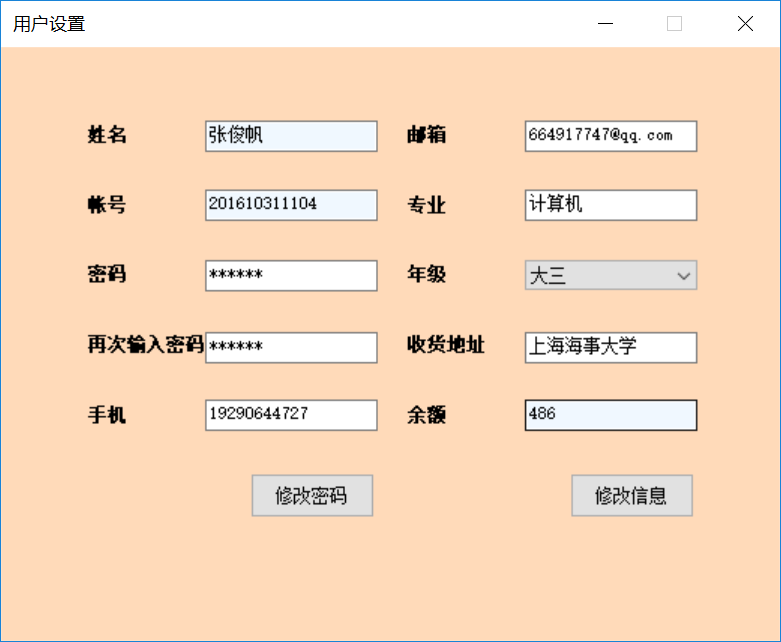


图 7.10 信息设置(用户)

在此界面可以对密码、手机、邮箱、专业、年级和收货地址进行设置，其中收货地址允许为空，但在付款前必须填写。

### 购物车



图 7.11 购物车界面

在此界面可以查看购物车并选择购买，也可将某项移出购物车，购物车内容可由商品查询界面(如图6.12)添加

### 商品查询



图 7.12 商品查询界面

在此界面可以直接购买或是将商品添加进购物车

### 发布商品



图 7.13 发布商品界面

在此界面可以挑选书籍种类并上架，种类由管理员在书目设置界面(如图6.5)添加

### 管理商品



图 7.14 管理商品界面

在此界面可以管理发布的商品，可修改其数量和库存，也可以将其下架(有订单的商品不允许直接下架，卖家应先将库存置零并处理完所有订单)

注：蓝色标注为存在订单的商品

### 现存订单查询(买家)



图7.15 现存订单查询界面(买家)

在此界面可以进行付款、确认收货和取消订单的操作。

### 现存订单查询(卖家)



图 7.16 现存订单查询界面(卖家)

在此界面可以进行确认发货和取消订单操作。

### 历史订单查询



图 7.17 历史订单界面

在此界面可以查询用户买或卖产生的所有历史订单。

# 总结与展望

由于在前期的设计环节没能完全考虑妥当，前期的设计曾被删改，导致开发上花费了不少时间，造成了不必要的时间和精力的浪费。由此吸取教训，应该重视前期的构思和设计环节，这样反而能大大地加快开发进度。

另外，虽已完善了整个流程，但由于时间问题，未能做出一些实际原定开发的板块(例如货品详情展示)，且虽提供了历史订单查询功能，但忽略了报表功能的设计(若以历史订单查询功能为基础上，即应该可以快速开发出报表功能)。

本次课程设计即是对实践能力的锻炼，亦是对上学期所学概念的复习，理论联系实际，巩固了对数据库的全方位理解。

总结起来，本次课程设计暴露出本人许多对于系统开发不太成熟的方面，由此吸取教训，若是能多注重前期设计，多参考商用的方案，也许日后将少走许多弯路。

# 参考文献

1. **[9787040406641]王珊,萨师煊,《数据库系统概论》,高等教育出版社，2014-09**
2. **[9787300210513]胡西川,《VisualC#.NET框架程序设计》，中国人民大学出版社，2015-05**