目录

1 需求分析 3
1.1 信息要求 3
1.2 处理要求 3
1.3 安全性与完整性要求 3
2 概念结构设计 4
2.1 六个实体 4
2.2 实体之间的联系 4
2.3 完整 E-R 图 4
3 逻辑结构设计 5
3.1 实体对应的关系模式 5
3.2 联系对应的关系模式 5
4 数据库实施 5
4.1 数据库的创建 5
4.1.1 表的创建： 5
4.1.2 用户定义完整性（CHECK）： 6
4.1.3 参照完整性 8
4.2 数据库的操作 8
4.2.1 增（插入，insert） 8
4.2.2 删（delete） 9
4.2.3 查（select） 9
4.2.4 改（update） 10
5 数据库运行和维护 11
5.1 数据库与 C#的连接 11
5.2 高级语言 C#对数据库的操作 11
5.3 防止非法数据插入数据库的提示 12
5.4 意外退出系统后重起系统的初始化操作 13
5.5 正常退出系统的数据库更新操作 14
6 系统演示、使用手册 16
6.1 开始界面 16
6.2 顾客界面 17
6.2.1 添加顾客信息 17
6.2.2 修改顾客信息 17
6.2.3 预定功能 17
6.2.4 就餐（选座，就餐） 18
6.3 管理员界面 20
6.4 房间和餐桌管理 21
6.4.1 增加和修改房间信息 21
6.4.2 增加餐桌 22
6.4.3 删除房间和餐桌 23
6.5 职工管理 23
6.5.1 界面介绍 23
6.5.2 对职工信息的操作（增、删、改） 24

6.6 订单管理 24
6.6.1 界面介绍 24
6.6.2 查看历史订单 25
6.6.3 点菜功能 26
6.6.4 结账功能 26
6.6.5 点评功能 27
6.7 顾客管理 28
6.7.1 界面介绍 28
6.7.2 对顾客信息的操作（增、删、改） 28
6.7.3 光顾功能 29
6.7.4 光顾历史记录 29
6.8 菜谱管理 30
6.8.1 界面介绍 30
6.8.2 对菜谱的基本操作（增、删、改） 31
6.8.3 快速查找和模糊查找 31
6.8.4 快速下单 32
7 体会 32

1 需求分析

1.1 信息要求

目前大多数酒店由于规模的限制，忽略了点菜系统的重要性。点菜系统专为具有一定规模和经济条件的大型酒店设计，通过集成从顾客定桌、点菜、上菜到结账等一系列功能，为每个环节明确分工，并通过可视化的软件支持，有效减小人为差错的概率，代之以高效、便捷、准确的数字化服务系统，使酒店的管理更加规范化。

因此该系统应当可以对餐桌信息、菜谱信息、职工信息以及顾客信息进行管理，同时点菜系统少不了订单，因此也应该能对订单包括过去、现在、未来订单进行管理。

1.2 处理要求

系统应当允许对服务员信息、菜单信息、厨师信息、房间信息、餐桌信息的管理：查询、增、删、改。预定服务：顾客可以根据自己的需求，预定不同型号的房间或大厅，餐桌和时间就餐。

点菜功能：顾客可以自行点菜，也可以让管理员代为服务，点菜后能生成订单为对其进行后序的做菜上菜结账评价等服务。

厨师做菜服务：厨师会根据订单状态主动工作。厨师和菜分别分组，每组厨师和一组菜一一对应，该组每位厨师会做该组所有的菜。厨师做菜管理：厨师可以获得自己的待做菜单，并对已做的菜进行标记。

结账、评价服务：审核菜单，协助顾客结账。结账完成后可以对订单进行评价，选出自己喜欢的菜。在查看自己个人信息时可以看到自己常点的菜。评价除了是对菜的评价，也是对做这道菜的厨师进行评价，方便日后拓展对厨师业绩进行考核的功能。

1.3 安全性与完整性要求

顾客能对自己的信息进行编辑，不能对其他信息进行任何编辑。对于房间和餐桌信息也只能看到可用的房间和餐桌。而且无法获取职工的信息包括其工作状态。同时顾客没有任何删除数据的权限，即使是自己的数据也无法删除。

管理员用于几乎对所有实体信息进行编辑，包括增、删、查、改。当然在操作前应当检查操作的数据是否合法，例如增加顾客时，顾客联系方式应当是数字，且固话应当是 8 位或 12 位，手机应当是 11 位等。这些理应属于数据库的用户定义完整性约束，但在此交由高级语言实现。

但是对于部分信息只能由系统产生和修改，例如订单号的生成是自动完成的，管理员也无法干预。顾客评价后系统会对相应的菜谱和厨师进行业绩属性的修改，这也是系统自动完成的。与订单相关的信息目前只能由系统增加和更改其状态（是否上菜或结账），这部分信息为了保证完整新和安全性，是无法删除的。再例如房间和餐桌的信息，房间中的餐桌数和使用情况也是有系统自动更新完成，为防止用户不当操作，不允许管理员直接对这部分信息进行修改。

数据库设计时应当考虑实体有实体完整性约束，实体之间联系产生的表具有参照完整新约束，对于部分属性有根据语义定义的用户定义完整性约束。具体举例见 4.1。

2 概念结构设计

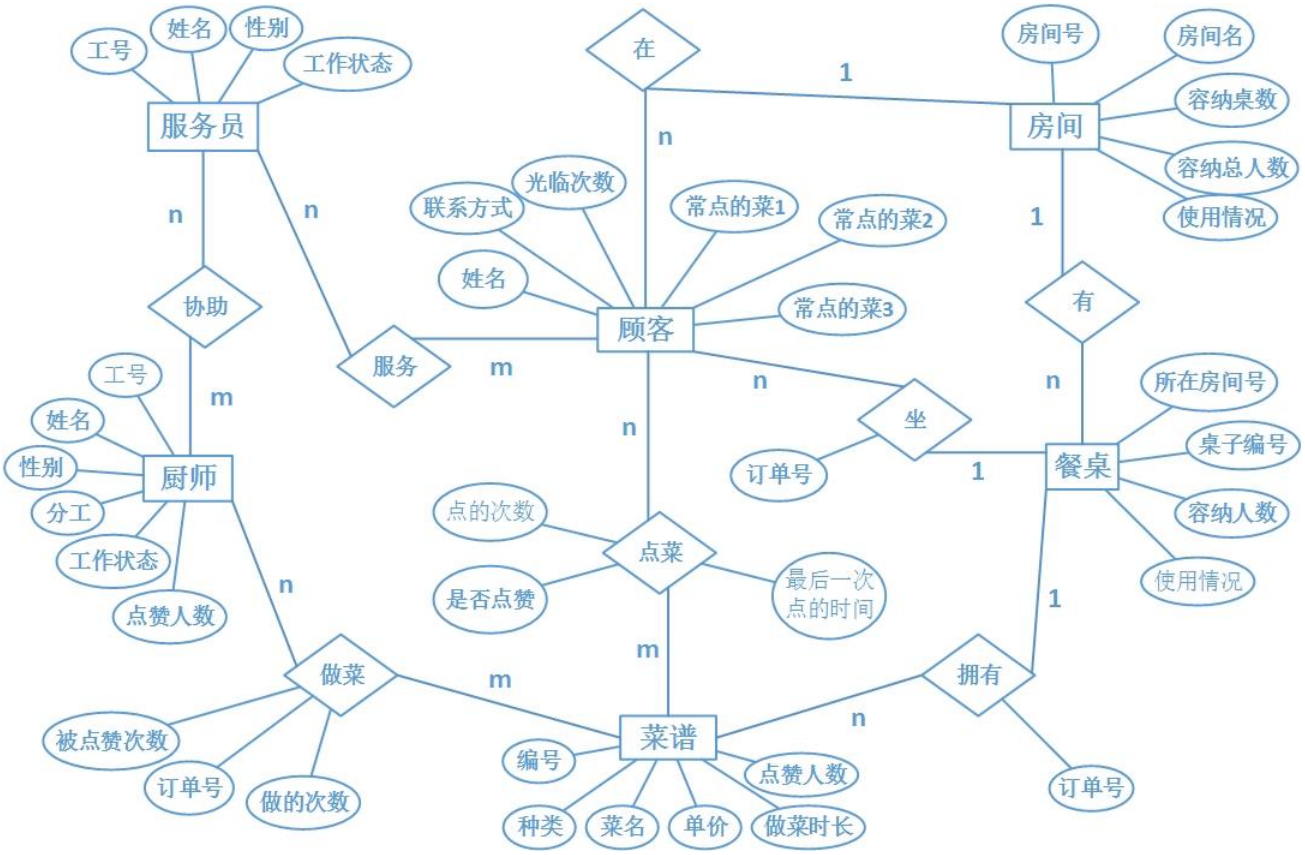
2.1 六个实体

根据需求分析，本系统共有六个实体，即服务员，厨师，顾客，房间，餐桌，菜谱，他们的属性如 2.3 中的 E-R 图所示。

2.2 实体之间的联系

大部分联系是多对多的联系，如服务员服务顾客等；也有一对一联系，如一个顾客只能坐一张桌子；一对多的联系主要体现在一个房间可以有多张餐桌。大部分联系是二元联系，而且会产生其他属性，如顾客和餐桌的联系会产生一个订单号。

2.3 完整 E-R 图



3 逻辑结构设计

3.1 实体对应的关系模式

Chref (COid, COname, COsex, COjob, COifwork, CObethumbup)
Waiter (Wid, Wname, Wsex, Wifwork)
Desk (Did, Droomid, Dpersoncount, Difuse)
Room (Rid, Rname, Rdeskcount, Rpersoncount, Rusedesks)
Customer (Ctellphone, Cname, Csex, Ccometimes, Cregularfood1, Cregularfood2, Cregularfood3)
Menu (Fid, Ftype, Fname, Fprice, Fcooktime, Fthumbup)

3.2 联系对应的关系模式

厨师工作记录：ChrefMenu (COid, Fid, Ordernum, Ifthumbup)
订单记录：CustomerDesk (Ctellphone, Did, Ordernum, Orderdate, Ifcheckout, Ordermoney)
顾客点菜历史：CustomerMenu (Ctellphone, Fid, Lastordertime, Ordercount, Ifthumbup)
上菜状态记录：DeskMenu (Did, Fid, Ordernum, Ifservedish)

4 数据库实施

4.1 数据库的创建

4.1.1 表的创建：

其中包含了实体完整性，即主码必须唯一且不为空。默认违约处理是拒绝插入和修改。详细表的建立见如下截图。

厨师表

DESKTOP-ACD5BL...系统 - dbo.Chref*

列名	数据类型	允许 Null 值
COid	char(4)	☐
COname	varchar(20)	☐
COsex	char(2)	☐
COjob	varchar(10)	☐
COifwork	char(4)	☐
CObethumbup	smallint	☐
		☐

菜谱表

DESKTOP-ACD5B...系统 - dbo.Menu

列名	数据类型	允许 Null 值
Fid	char(6)	☐
Ftype	varchar(10)	☐
Fname	varchar(20)	☐
Fprice	real	☐
Fcooktime	smallint	☐
Fthumbup	smallint	☐

顾客信息表

订单表

DESKTOP-ACD5BL...- dbo.Customer

列名	数据类型	允许 Null 值
Ctellphone	varchar(12)	☐
Cname	varchar(20)	☐
Csex	char(2)	☐
Ccometimes	smallint	☐
Cregularfood1	char(6)	☑
Cregularfood2	char(6)	☑
Cregularfood3	char(6)	☑

DESKTOP-ACD5BL...o.CustomerDesk*

列名	数据类型	允许 Null 值
Ordernum	char(12)	☐
Ctellphone	varchar(12)	☐
Did	char(7)	☐
Orderdate	date	☐
Ifcheckout	char(2)	☐
Ordermoney	real	☐

顾客点菜历史表

DESKTOP-ACD5B...o.CustomerMenu

列名	数据类型	允许 Null 值
Ctellphone	varchar(12)	☐
Fid	char(6)	☐
Lastordertime	date	☐
Ordercount	smallint	☐
Ifthumbup	char(2)	☑

餐桌表

DESKTOP-ACD5B...菜系统 - dbo.Desk

列名	数据类型	允许 Null 值
Did	char(7)	☐
Droomid	char(4)	☐
Dpersoncount	smallint	☐
Difuse	char(2)	☐

上菜状态记录表

DESKTOP-ACD5BL...- dbo.DeskMenu

列名	数据类型	允许 Null 值
Did	char(7)	☐
Fid	char(6)	☐
Ordernum	char(12)	☐
Ifservedish	char(2)	☐

厨师工作记录表

DESKTOP-ACD5BL...dbo.ChrefMenu

列名	数据类型	允许 Null 值
COid	char(4)	☐
Fid	char(6)	☐
Ordernum	char(12)	☐
Ifthumbup	char(2)	☐

房间表

DESKTOP-ACD5B...系统 - dbo.Room

列名	数据类型	允许 Null 值
Rid	char(4)	☐
Rname	varchar(10)	☐
Rdeskcount	smallint	☐
Rpersoncount	smallint	☐
Rusedesks	smallint	☐

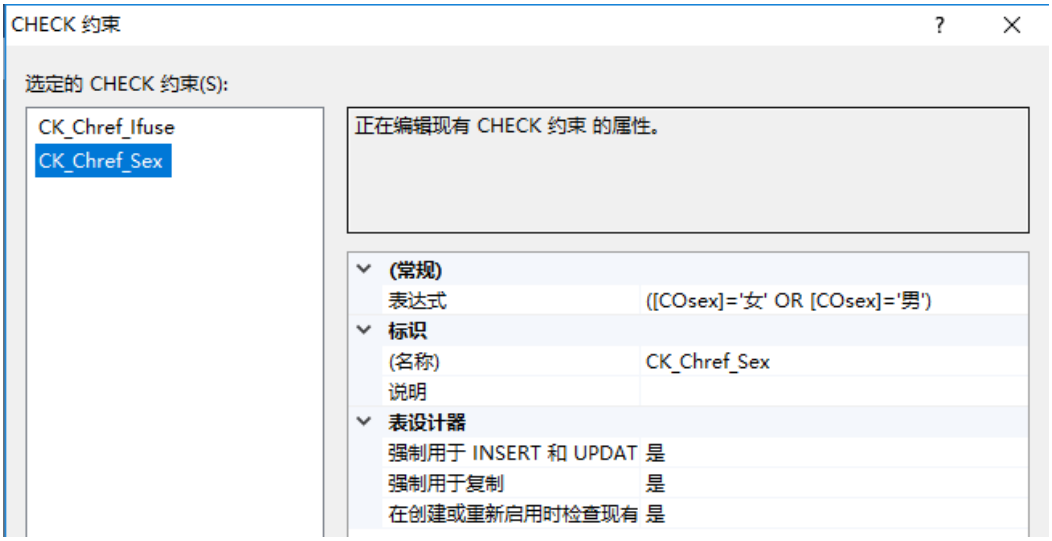
服务员表

DESKTOP-ACD5BL...统 - dbo.Waiter

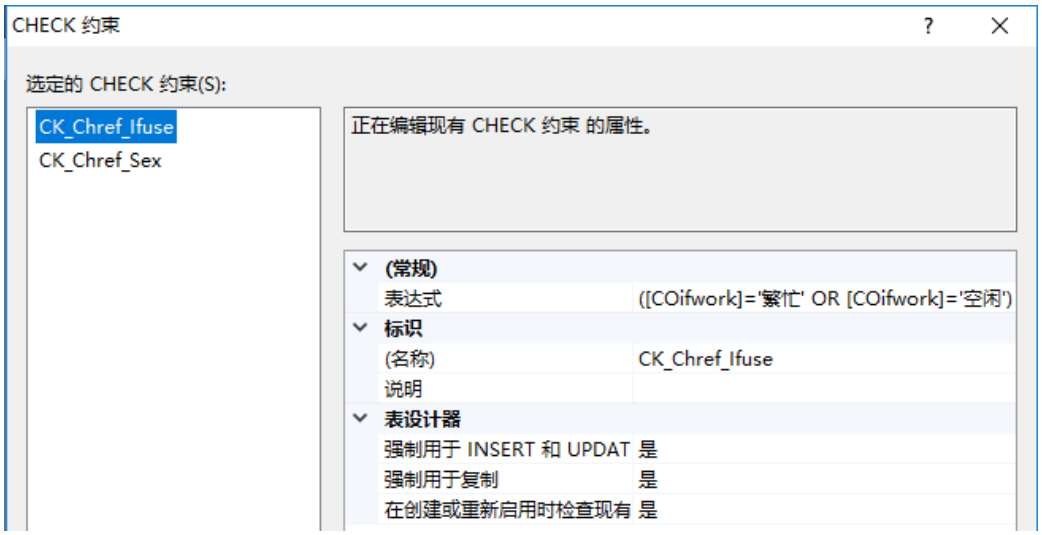
列名	数据类型	允许 Null 值
Wid	char(4)	☐
Wname	varchar(20)	☐
Wsex	char(2)	☐
Wifwork	char(4)	☐

4.1.2 用户定义完整性（CHECK）：

对于部分属性，其值有语义要求，因而需要定义 CHECK 约束，如顾客、职工的性别（只能是男或女），职工的工作状态（只能是空闲或繁忙），上菜状态和订单结账（只能是是或否），部分截图如下：



（厨师表中对性别的约束）

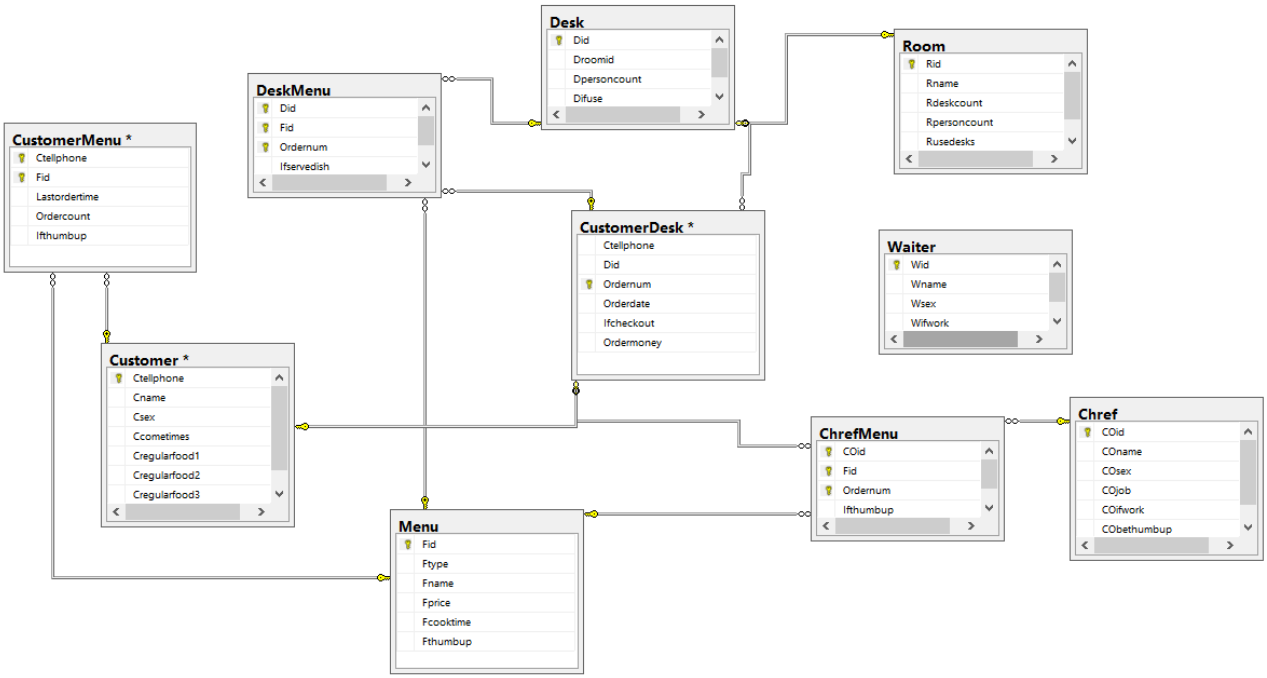


（厨师表中对工作状态的约束）

（厨师工作记录表中是否被点赞的约束）

4.1.3 参照完整性

参照完整性约束主要是外键的定义。在本系统中，除顾客表外其他表的顾客联系方式都是外键，被参照关系是顾客表。除订单表外的订单号都是外键，被参照关系是订单表，厨师编号、菜编号、房间号同理，它们对应的被参照关系是厨师表，菜谱表，房间表。下图是 sql server 中自动生成的数据库关系图，比较直观地看到参照完整性情况。



4.2 数据库的操作

根据需求分析知道具体要对具体表的操作，为了方便展示，下面情况各选取一个例子展示。由于是通过嵌入式 sql，所以展示图例是本次实验使用的高级语言 C#的代码截图。

4.2.1 增（插入，insert）

顾客预订或点菜后，实际上是只要顾客选择好座位后会生成订单并插入到订单表中，对应的的插入语句是：为了方便日后维护，参数化 sql 的语句的建立不用 string 类而是用 StringBuilder 类，该类在处理大量字符串时效率低，但是应对 sql 语句这种不长的字符串不会对效率产生副作用，而且建立过程十分直观，方便日后维护。

```
StringBuilder strSQLinsert = new StringBuilder();
strSQLinsert.Append("insert ");
strSQLinsert.Append("into CustomerDesk(Ctellphone, Did, Ordernum, Orderdate, Ifcheckout, Ordermoney) ");
strSQLinsert.Append(
    string.Format
    ("values('{0}','{1}','{2}','{3}','否',{4});", nowOrderDishCustomerID, deskNum, orderNum, sqlDate, orderMoney));
cmd.CommandText = strSQLinsert.ToString();
cmd.ExecuteNonQuery();
```

4.2.2 删（delete）

删除操作顾客是没有的，只有管理员，而且仅仅限于对顾客信息，职工信息，餐桌信息，房间信息。其他信息是无法删除的。由于考虑到参照完整性约束，这些删除操作都不是独立的，应当考虑到参照表。下面的举例是删除某房间时，应当删除该房间下所有的餐桌：

```
if (result == DialogResult.OK)
{
    SqlConnection conn = new SqlConnection(Properties.Settings.Default.connDateString);
    conn.Open();
    //删除该房间下的所有餐桌
    string sqlDelDesk = string.Format("delete from Desk where Droomid = '{0}';", Rid);
    SqlCommand cmd = new SqlCommand(sqlDelDesk, conn);
    cmd.ExecuteNonQuery();

    //删除该房间
    cmd.CommandText = string.Format("delete from Room where Rid = '{0}';", Rid);
    if (cmd.ExecuteNonQuery() > 0)
    {
        MessageBox.Show("删除成功！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
        roomDataView.Rows.Remove(roomDataView.SelectedRows[0]);
    }
}
```

4.2.3 查（select）

查询是最常见的数据库操作，本系统允许用户根据需求查询，例如点击某类菜的种类，就能看到该种类下的菜谱，再例如点击对应的厨师，应当能显示该厨师的工作记录等。同时点菜时支持对菜名的模糊查询。下面的举例是带排名（按照点赞次数的降序排序）的菜谱查询以及对菜名的模糊查询。

```
/// <summary>
/// 选择菜的种类后，会自动显示该种类下有什么具体的菜
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 个引用
private void SearchFoodInThisType(object sender, EventArgs e)
{
    menuDataView.ClearSelection();
    string strSelectFtype = typeList.SelectedItem.ToString();
    string strSqlFoodInFtype;
    if (strSelectFtype == "全部")
    {
        strSqlFoodInFtype = "select Fid, Ftype, Fname, Fprice, Fcooktime, Fthumbup from Menu order by Fthumbup desc;";
    }
    else
    {
        strSqlFoodInFtype =
            string.Format("select * from Menu where Ftype = '{0}' order by Fthumbup desc;", strSelectFtype);
    }

    SqlConnection conn = new SqlConnection(Properties.Settings.Default.connDateString);
    conn.Open();
    SqlDataAdapter adt = new SqlDataAdapter(strSqlFoodInFtype, conn);
    DataTable dt = new DataTable();
    adt.Fill(dt);
    menuDataView.DataSource = dt;

    adt.Dispose();
    dt.Dispose();
    conn.Close();
}
```

（点击菜的种类就能看到该种类下的菜谱，并根据点赞次数降序呈现结果）

```
private void btn模糊查询_Click(object sender, EventArgs e)
{
    string tempFname = textBox模糊搜索.Text.ToString().Trim();
    if (tempFname == "")
    {
        MessageBox.Show("找不到对象！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    string sqlFname = "%" + tempFname + "%";
    menuDataView.ClearSelection();

    string sqlSelect =
        string.Format("select * from Menu where Fname like '{0}' order by Fthumbup desc;", sqlFname);

    SqlConnection conn = new SqlConnection(Properties.Settings.Default.connDateString);
    conn.Open();
    SqlDataAdapter adt = new SqlDataAdapter(sqlSelect, conn);
    DataTable dt = new DataTable();
    adt.Fill(dt);
    menuDataView.DataSource = dt;

    adt.Dispose();
    dt.Dispose();
    conn.Close();
}
```

（模糊查询，把输入信息“包”在两个%中间即可，百分号可以代替任意长度的字符串，包括长度为 0）

4.2.4 改（update）

管理员对数据的修改和删除一样限制在几个表中，顾客其实也可以间接修改数据，例如点赞某道菜的时候实际上是在修改点赞情况。其他的修改操作均有系统自动完成，例如上菜状态更新，账单状态更新等，还有对顾客常点的菜的更新和厨师被点赞的次数都是系统完成。下面展示是对厨师被点赞次数的更新：

```
//更新厨师被点赞次数
SqlCommand cmd = new SqlCommand("select COid from Chref", conn);
SqlDataReader sdr = cmd.ExecuteReader();
while (sdr.Read())
{
    string COid = sdr["COid"].ToString();
    SqlCommand cmd1 = new SqlCommand();
    cmd1.Connection = conn;
    cmd1.CommandText = string.Format
        ("select count(*) from ChrefMenu where COid = '{0}' and Ifthumbup = '是'", COid);
    int CObethumbup = int.Parse(cmd1.ExecuteScalar().ToString());
    cmd1.CommandText = string.Format
        ("update Chref set CObethumbup = {0} where COid = '{1}';", CObethumbup, COid);
    cmd1.ExecuteNonQuery();
}
cmd.Dispose();
sdr.Dispose();
```

5 数据库运行和维护

5.1 数据库与 C#的连接

将连接字符串保存在配置文件中，调用时直接调用配置文件中的信息这样做的好处是程序移植时只需要修改配置文件中的连接字符串即可，不需要修改任何程序，同时也方便程序编写时的调用。连接字符串的修改也非常简单，可以有系统自动生成。下图分别是配置文件中的信息以及配置配置文件的截图之一。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
    </configSections>
    <connectionStrings>
        <add name="version0._1.Properties.Settings.connDateString" connectionString="Data Source=DESKTOP-5CJHD10;
            providerName="System.Data.SqlClient" />
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
</configuration>
```

（留意到 connDateString，这个就是连接字符串的名字）

	名称	类型	范围	值
▶	connDateString	(连接字符串)	应用程序	Data Source=DESKTOP-5CJHD10;Initial Catalog=点菜系统;Integrated Security=True;MultipleActiveResultSets=true;
*				

（连接字符串的值是选择好服务器和数据库后，系统自动生成的）

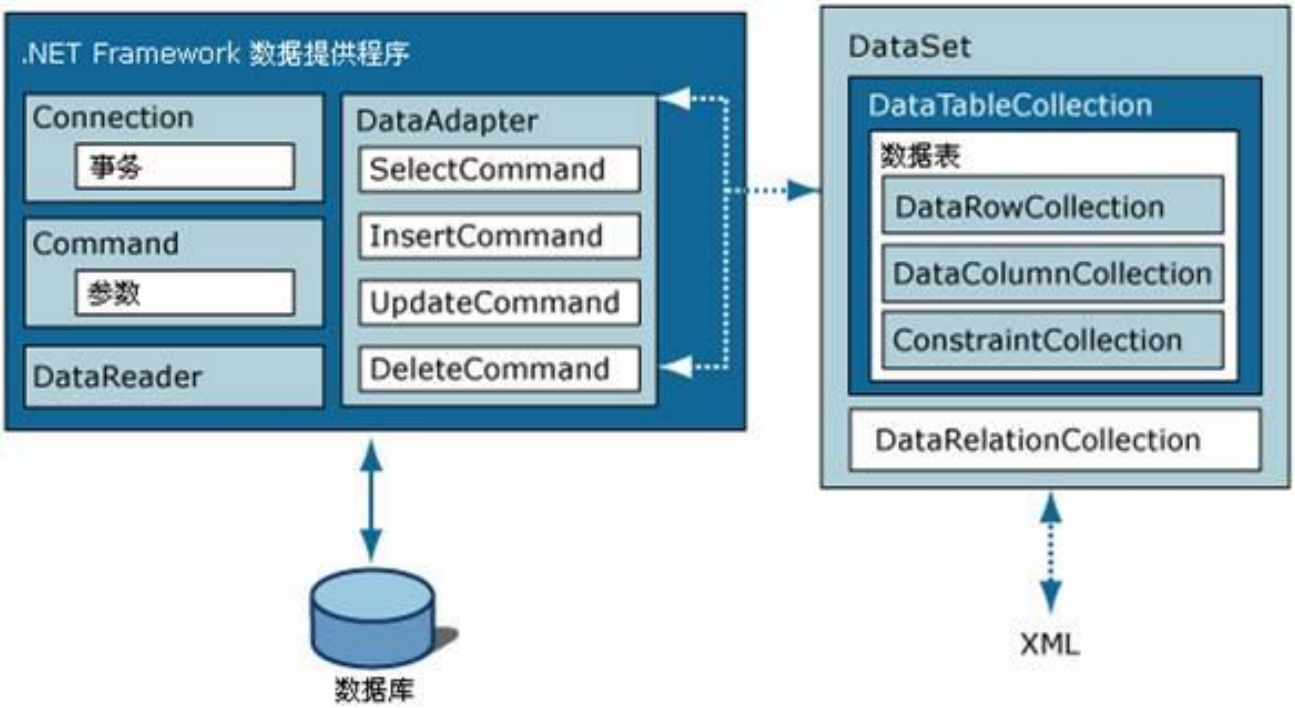
连接字符串使用 C#提供的 SqlConnection 类，包含在命名空间 system.Data.SqlClient 中，调用构造函数时需要传参，参数是连接字符串。详细见下图，连接打开后，操作完数据库后需要及时释放资源和关闭连接。

```
SqlConnection conn = new SqlConnection(Properties.Settings.Default.connDateString);
conn.Open();//打开连接

sql操作

conn.Dispose();//释放该连接占用的资源
conn.Close();//关闭该连接
```

5.2 高级语言 C#对数据库的操作

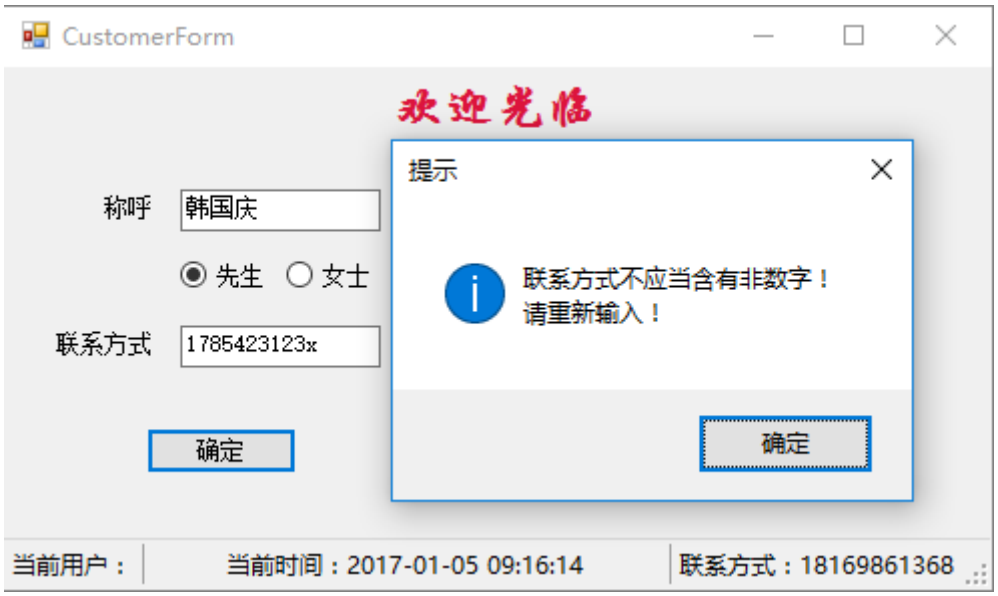
引用较为典型的 ADO.NET 结构图，其中为了能尽可能的练习 sql 语句，本次实验没有用到 DataAdapter 下的 4 个 Command 对象，这些方法是系统根据用户直接在 DataAdapter 上的修改操作自动生成对应的 sql 语句去更新数据库，但仅限于当个表的数据。同时也没有使用右边的 DataSet 和 XML，因为 DataSet 是针对需要连接外服务器数据库而设定的本地高速缓冲区，但实验所用的数据库是本地的，而且操作量不大，DataSet 本身也比较占用资源。

具体的用法在上述的 4.2 的截图中有所涉及。

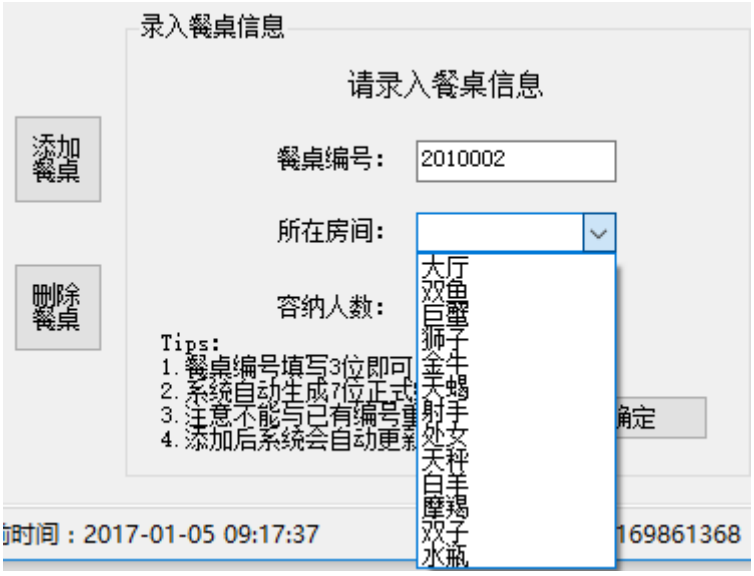


5.3 防止非法数据插入数据库的提示

这部分主要是通过 C#提供的 MessageBox 控件和下拉框完成。非法数据的来源主要是用户的不正当操作，例如数据长度不正确，格式不正确等。下面展示在新增顾客时，顾客输入联系方式时意外输入非数字字符的情况，这是通过 MessageBox 完成提示。以及新增餐桌时需要输入餐桌所在的房间，这是通过下拉框实现。



（联系方式不得有非数字字符）



（录入餐桌信息时只能选择已有房间）

5.4 意外退出系统后重起系统的初始化操作

由于厨师做菜等功能并非由数据库完成，而是存放在内存中，系统一旦退出数据无法存入到数据库（例如做菜剩余的时间）。如果再次进入系统时不对还没完成的订单进行检索，那么这些订单的没上的菜在系统中将一直不会被厨师烹饪。因此进入系统后会先检索是否还有没上的菜，有的话会加入到做菜队列，并触发厨师做菜的事件。核心代码如下：

```
public static void SearchNoServerdish()
{
    //检查DeskMenu表
    SqlConnection conn = new SqlConnection(Properties.Settings.Default.connDateString);
    conn.Open();
    StringBuilder sqlSearchNoServerdish = new StringBuilder();
    sqlSearchNoServerdish.Append("select Did, Fid, Ordernum ");
    sqlSearchNoServerdish.Append("from DeskMenu ");
    sqlSearchNoServerdish.Append("where Ifservedish = '否';");
    SqlCommand cmd = new SqlCommand(sqlSearchNoServerdish.ToString(), conn);
    SqlDataReader sdr = cmd.ExecuteReader();
    //联合Menu表构成待做的菜的结构体
    while (sdr.Read())
    {
        string Did = sdr["Did"].ToString();
        string Fid = sdr["Fid"].ToString();
        string Ordernum = sdr["Ordernum"].ToString();
        string sqlFoodType = string.Format("select Ftype from Menu where Fid = '{0}';", Fid);
        SqlCommand cmd1 = new SqlCommand(sqlFoodType, conn);
        string Ftype = cmd1.ExecuteScalar().ToString();
        string sqlFoodCooktime = string.Format("select Fcooktime from Menu where Fid = '{0}';", Fid);
        cmd1.CommandText = sqlFoodCooktime;
        int Fcooktime = int.Parse(cmd1.ExecuteScalar().ToString());
        Dish temp = new Dish { };
        temp.foodID = Fid;
        temp.deskID = Did;
        temp.orderNum = Ordernum;
        temp.foodType = Ftype;
        temp.cookTime = Fcooktime;
        //调用GetDish，该函数会定时触发厨师做菜的事件。
        ControlOrderClass.GetDish(temp);
        cmd1.Dispose();
    }
    sdr.Dispose();
    cmd.Dispose();
    conn.Dispose();
    conn.Close();
}
```

5.5 正常退出系统的数据库更新操作

正常注销退出系统时，会对已完成订单和菜进行归总，更新厨师被点赞次数以及更新顾客常点的菜，保证数据的完整性。核心代码如下：

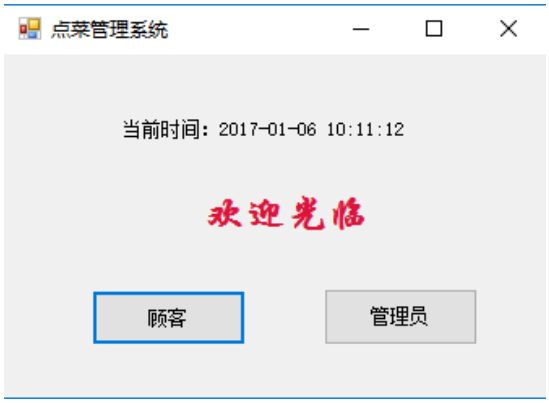
```
public static void UpdateCusAndChrefThumbUp()
{
    SqlConnection conn = new SqlConnection(Properties.Settings.Default.connDateString);
    conn.Open();

    //更新厨师被点赞次数
    SqlCommand cmd = new SqlCommand("select COid from Chref", conn);
    SqlDataReader sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        string COid = sdr["COid"].ToString();
        SqlCommand cmd1 = new SqlCommand();
        cmd1.Connection = conn;
        cmd1.CommandText = string.Format
            ("select count(*) from ChrefMenu where COid = '{0}' and Ifthumbup = '是'", COid);
        int CObethumbup = int.Parse(cmd1.ExecuteScalar().ToString());
        cmd1.CommandText = string.Format
            ("update Chref set CObethumbup = {0} where COid = '{1}';", CObethumbup, COid);
        cmd1.ExecuteNonQuery();
    }
    cmd.Dispose();
    sdr.Dispose();
    //更新顾客常点的菜
    cmd.CommandText = "select Ctellphone from Customer;";
    sdr = cmd.ExecuteReader();
    List<string> Ctellphone = new List<string>();
    while (sdr.Read())
    {
        Ctellphone.Add(sdr["Ctellphone"].ToString());
    }
    cmd.Dispose();
    sdr.Dispose();
```

```
    //对每个顾客更新
    foreach (string ct in Ctellphone)
    {
        StringBuilder sqlSelectTop3Dish = new StringBuilder();
        sqlSelectTop3Dish.Append("select top 3 Fid ");
        sqlSelectTop3Dish.Append("from CustomerMenu ");
        sqlSelectTop3Dish.Append("where Ordercount > 1 ");
        sqlSelectTop3Dish.Append(string.Format("and Ctellphone = '{0}' ",ct));
        sqlSelectTop3Dish.Append("order by Ordercount desc;");
        cmd.CommandText = sqlSelectTop3Dish.ToString();
        sdr = cmd.ExecuteReader();
        int i = 1;
        //写入最多三道菜
        while (sdr.Read())
        {
            string updateRegularfood = "Cregularfood" + i.ToString();
            string sqlUpdateRefood = string.Format
                ("update Customer set {0} = '{1}' where Ctellphone = '{2}';",
                updateRegularfood, sdr["Fid"].ToString(), ct);
            SqlCommand cmd1 = new SqlCommand(sqlUpdateRefood,conn);
            cmd1.ExecuteNonQuery();
            cmd1.Dispose();
            i++;
        }
        sdr.Dispose();
    }

    cmd.CommandText = "update Waiter set Wifwork = '空闲';";
    cmd.ExecuteNonQuery();

    conn.Dispose();
    conn.Close();
}
```

6 系统演示、使用手册

6.1 开始界面

开始界面有当前时间，欢迎词，还有使用者身份选择。进入管理员需要输入账号和密码。



6.2 顾客界面

6.2.1 添加顾客信息

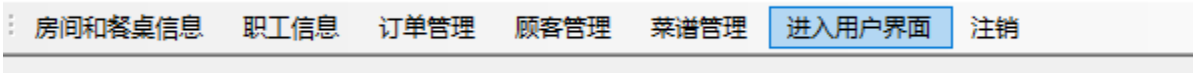
不需要注册，顾客只需要输入自己的信息，系统会自动检索数据库，如果没有则自动插入，有则把信息查找出来并显示。信息包括姓名、联系方式、常点的菜。

6.2.2 修改顾客信息

点击修改信息，左边的框会可以以输入，修改完信息后点击确定即可。

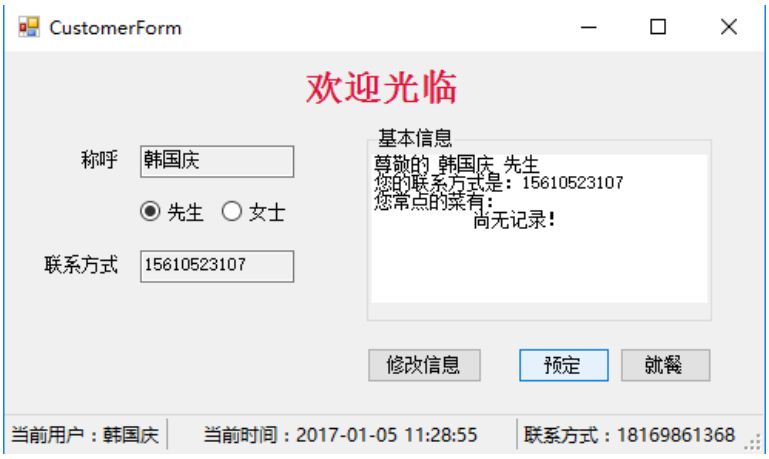
6.2.3 预定功能

1、在菜单栏中选择“进入用户界面”

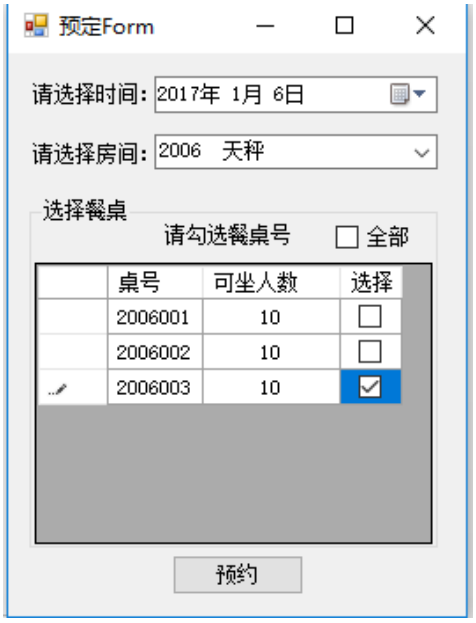


2、在弹出的窗口中输入信息，点击“确定”

3、选择“预定”



4、填写预定信息，然后点击“预约”，即可完成预约



6.2.4 就餐（选座，就餐）

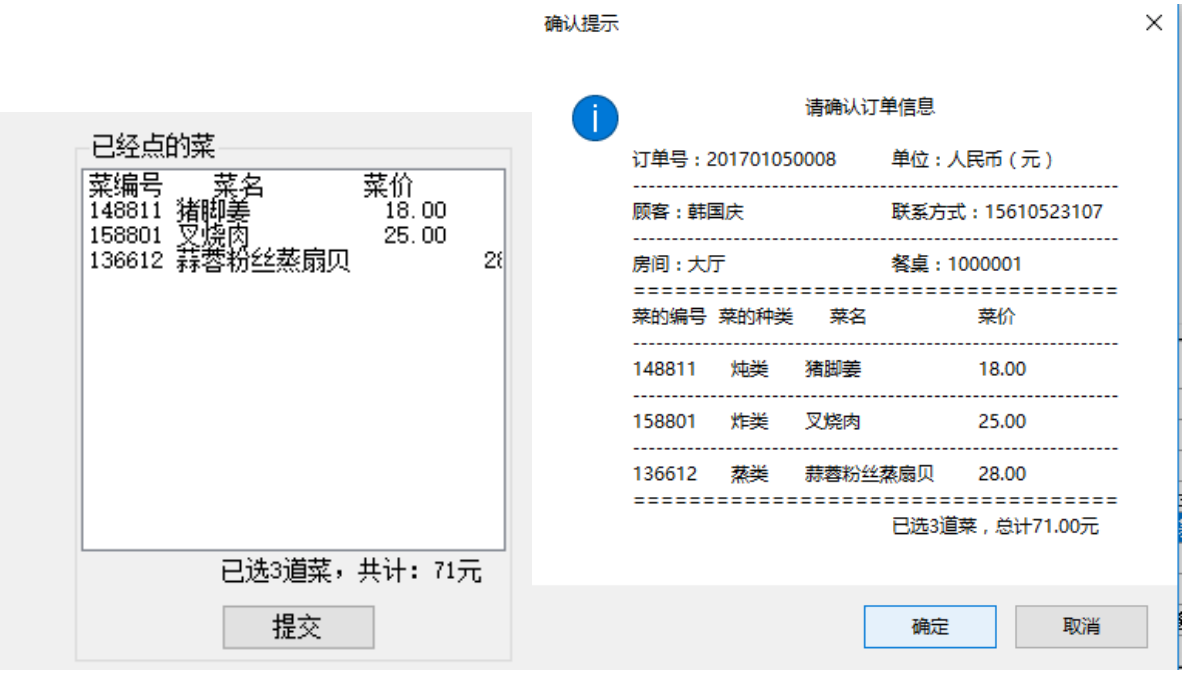
就餐步骤前两步同预约操作前两步，完成后进入如下页面

（选择“就餐”）

（点击下拉框选择房间）

（选定房间后会自动显示该房间里的餐桌，勾选餐桌。也可以输入桌号快速查找和模糊查找）

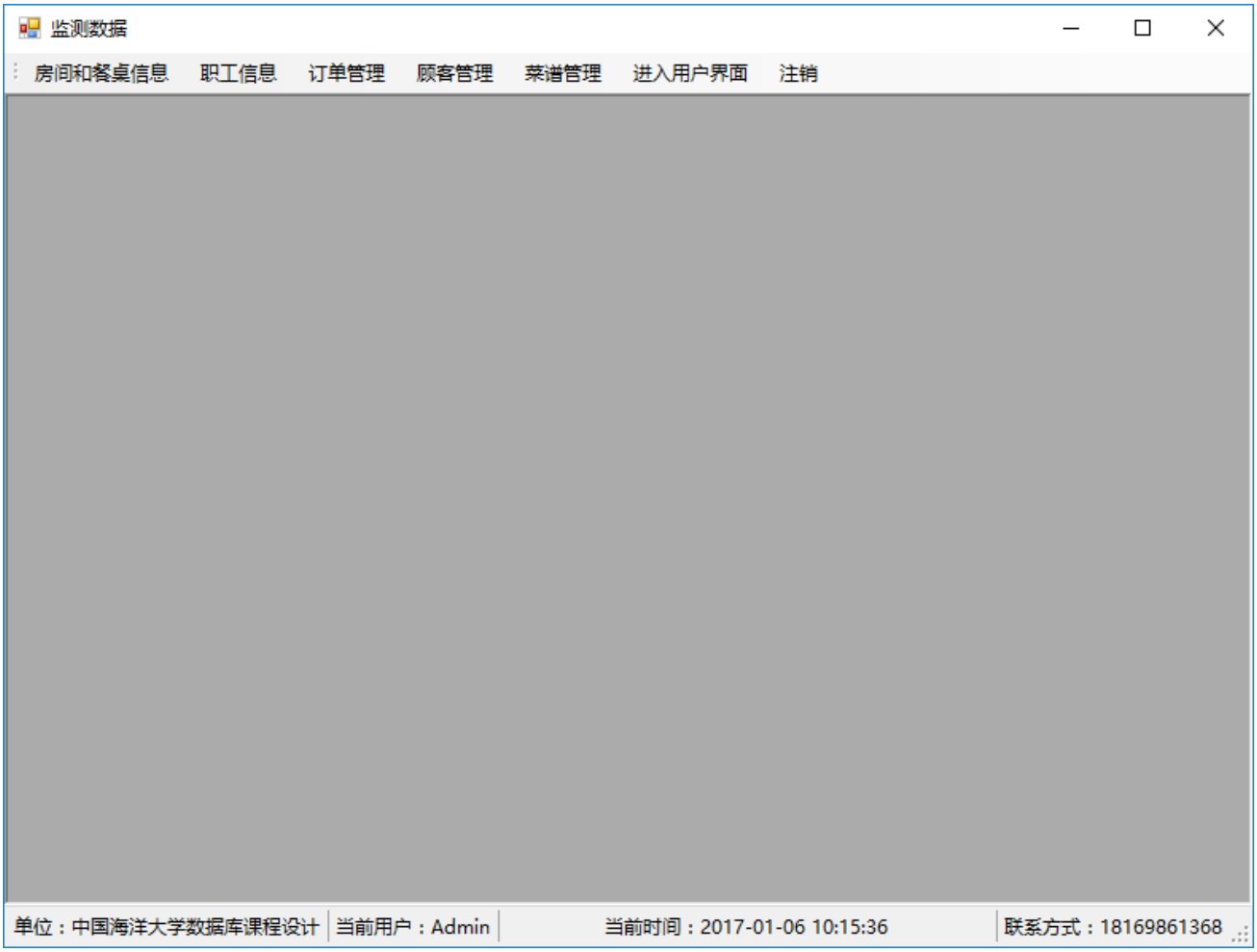
（点菜，可以选择菜的种类，双击要点的菜即可添加到菜单）



（点完菜后点击提交，确认订单信息后，即可完成就餐操作）

6.3 管理员界面

上面是菜单栏，点击对应的按钮会显示相应的界面，下面是状态栏，显示单位、时间以及作者。详细功能见下面描述。

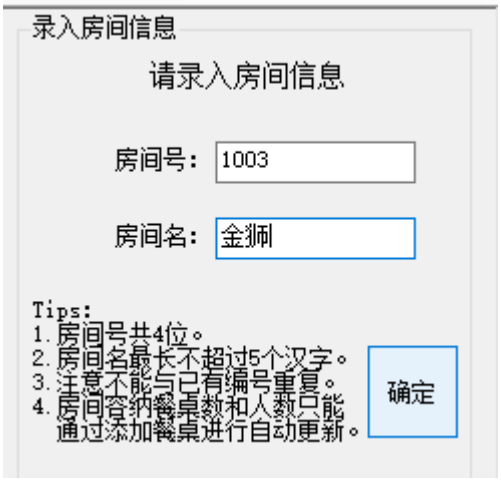


6.4 房间和餐桌管理

6.4.1 增加和修改房间信息

房间号	房间名	餐桌数	最大容纳人数	已使用餐桌数
1000	大厅	20	120	1
1001	双鱼	2	20	0
1002	巨蟹	2	20	0
2001	狮子	2	20	0
2002	金牛	1	10	0
2003	天蝎	1	6	0
2004	射手	1	6	0
2005	处女	1	6	0
2006	天秤	3	30	0

（进入“房间和餐桌信息”界面，在“房间概况”栏选择“添加房间”）



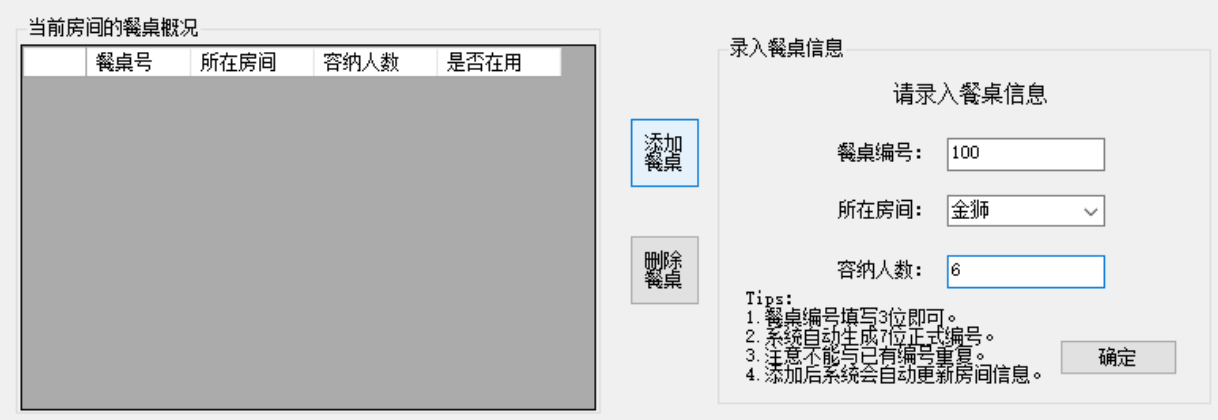
	1002	巨蟹	2	20	0
▶	1003	金狮	0	0	0
	2001	狮子	2	20	0

（录入房间信息后点击确定，即可添加房间）

若要修改房间信息，只需要选中要修改的房间，点击“修改房间信息”，然后修改房间名即可。

6.4.2 增加餐桌

进入“房间和餐桌信息”页面，在“当前房间的餐桌概况”中选择“添加餐桌”，录入餐桌信息后，点击“确定”，即可完成操作



6.4.3 删除房间和餐桌

（选中要删除的房间，点击“删除房间”，即可完成操作）

删除餐桌和删除房间操作相同

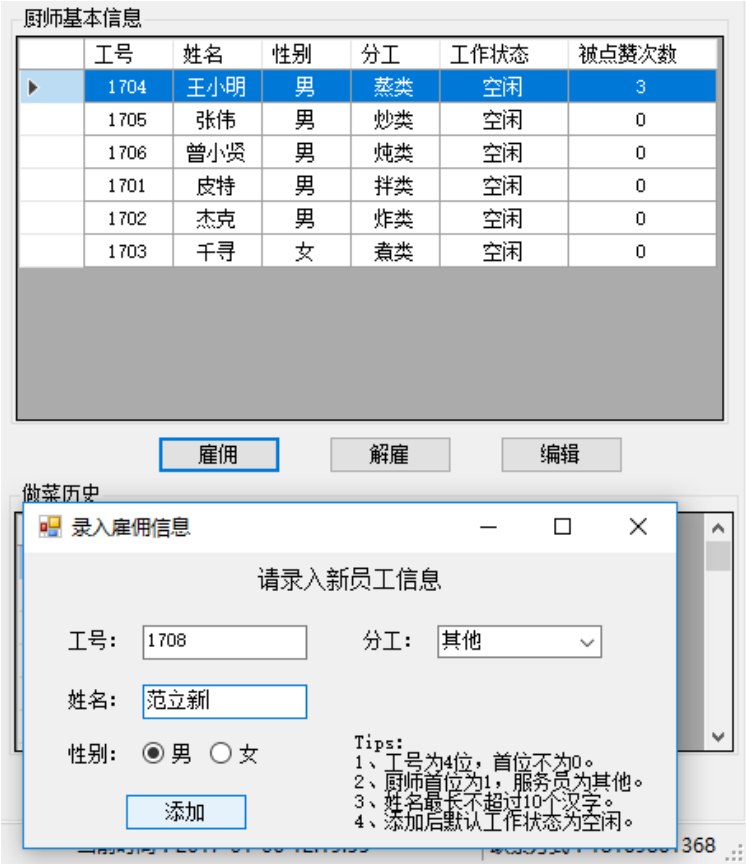
6.5 职工管理

6.5.1 界面介绍

职工管理界面分为三部分：服务员信息，厨师基本信息，做菜历史。“服务员信息”栏和“厨师基本信息”栏都提供三种功能：雇佣、解雇、编辑功能。管理员可以雇佣和解雇员工，或者修改员工的信息。“做菜历史”栏可以查看厨师的做菜信息，查看该厨师做的菜是否被点赞。

6.5.2 对职工信息的操作（增、删、改）

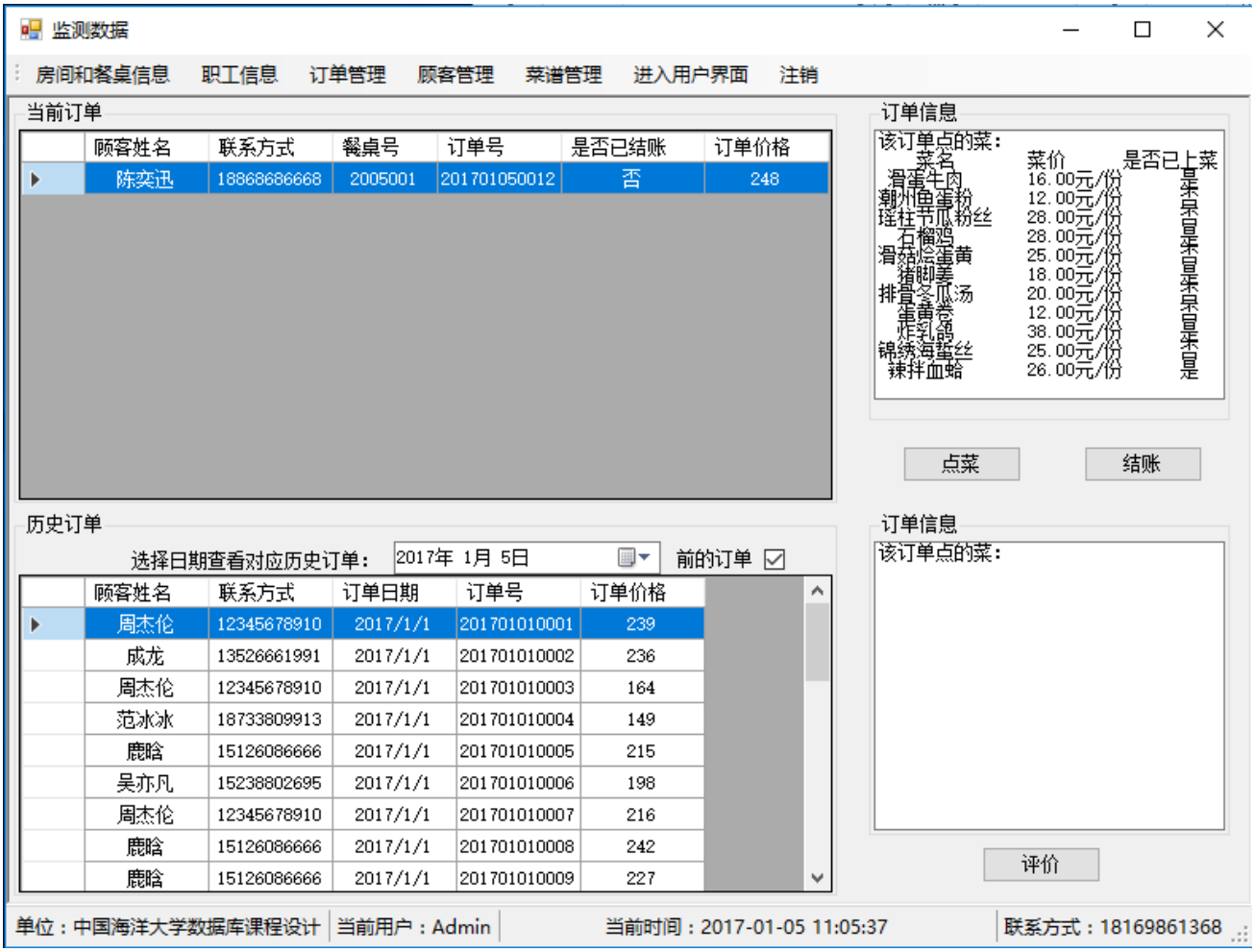
以添加厨师信息为例：点击“雇佣”，在弹窗输入员工信息，点击添加，即可完成操作。删除操作只需要选中要删除的员工，点击解雇，即可删除。编辑操作只需选中要修改信息的员工，点击修改，填写新的信息，即可完成



6.6 订单管理

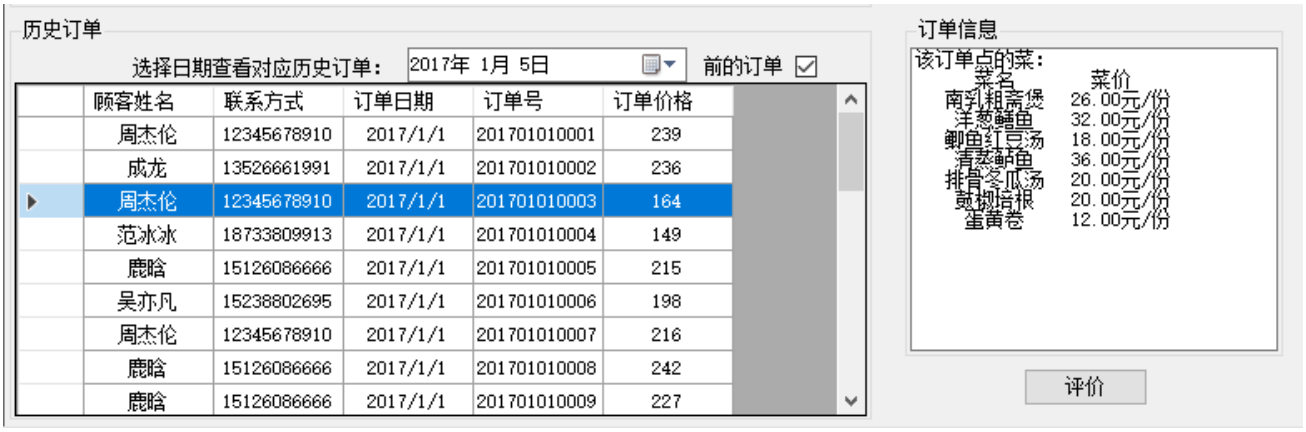
6.6.1 界面介绍

当在用户界面内，完成前面的操做提交订单之后，即可点击“订单管理”进入订单管理页面对订单进行管理。如下图所示：在“当前订单”页面我们可以看到当前正在处理的订单，点击正在处理的订单，我们可以在右侧查看该订单内所点的菜是否已经全部上菜。若菜已经全部显示“是”则表示菜已全部上齐，则顾客结账时可以点击结账按钮，进行结账。除此之外我们同样可以在订单管理页面对历史订单进行查看（该部分在下面会单独介绍）。



6.6.2 查看历史订单

在订单管理中我们同样可以查看历史订单的情况，方便餐厅管理人员对老顾客所喜爱的菜色进行统计，以达到可以为用户推荐菜色的功能。在“历史订单”窗口中我们可以对历史订单进行点击查看历史订单中所包含的菜的种类以及菜的价格。在历史订单中我们可以对日期进行选择查看具体某一天之前的订单，当取消对该项的选择时，则表示只查看当天的订单情况。



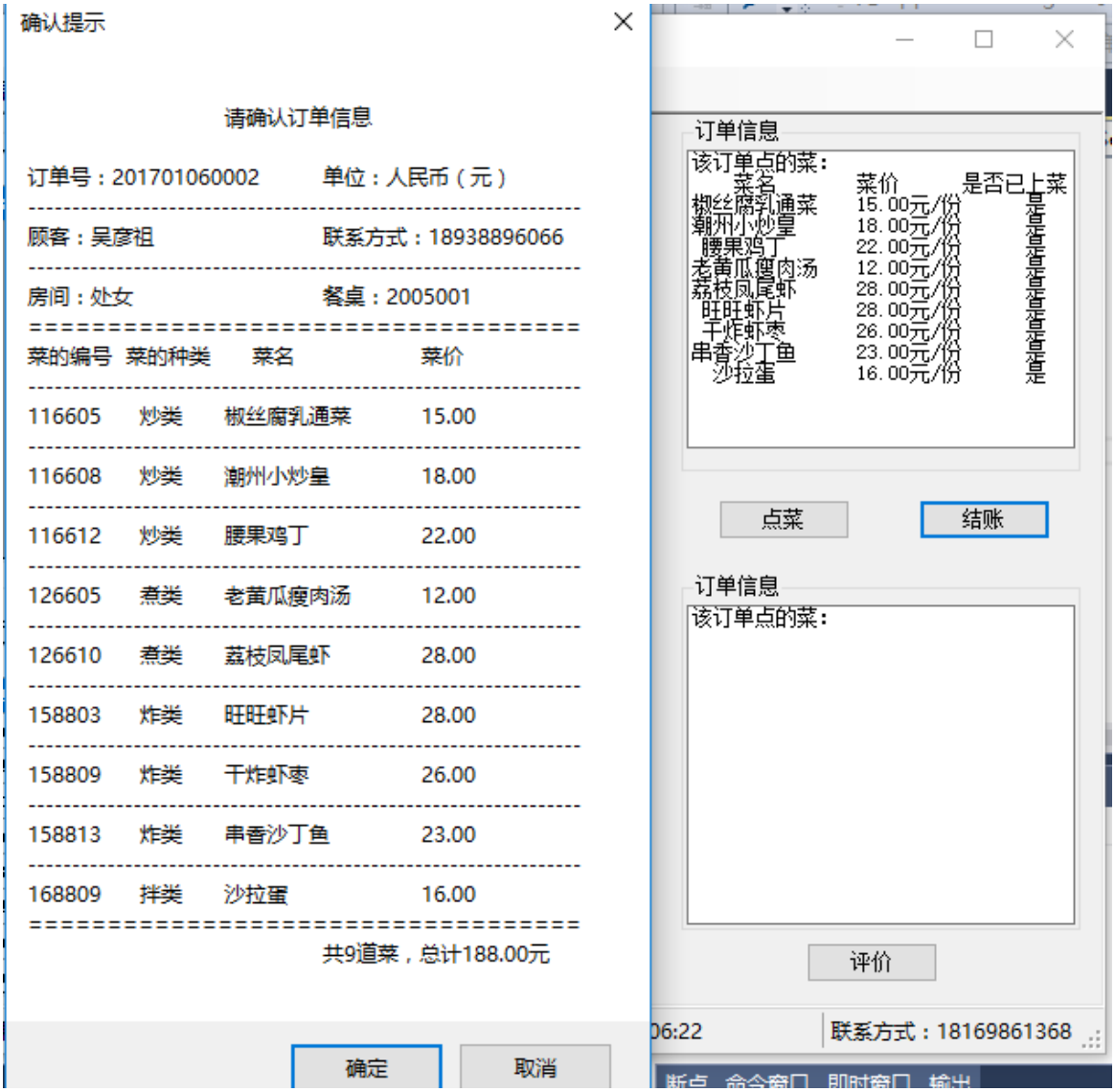
6.6.3 点菜功能

订单管理页面中同样存在点菜的选项，该选项是为了给使用预约功能的顾客点菜所使用的。当根据订单号查询到该预约信息时，我们可以点击点菜，为该订单进行点菜。

（在菜单窗口双击菜的元组即可将菜添加到订单中，同样取消时在订单中双击已点的菜既可以取消点菜）

6.6.4 结账功能

当订单中的菜，全部上完后。点击结账（如下图：）会弹出下图左侧的页面，方便用户对订单中的菜进行核对，以及对价格存在问题的菜进行查看，确定所应付款金额。



6.6.5 点评功能

（点击结账后会，弹出一个评价窗口。方便用户对订单内点的菜进行点赞）

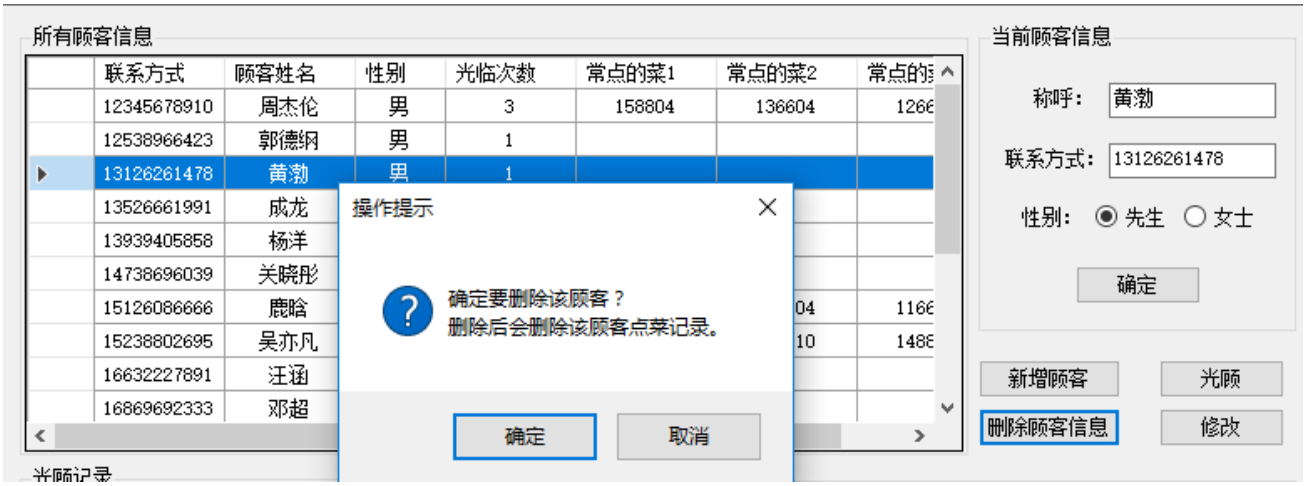
6.7 顾客管理

6.7.1 界面介绍

在主页面中点击“顾客管理”页面可以对顾客的信息进行操作和分析。顾客管理页面由所有顾客信息，当前顾客信息，顾客的光顾记录，和点菜历史四部分组成。通过这四部分我们可以完成对所有顾客的信息和订单进行管理。各部分具体的功能在下面会有具体的介绍：

6.7.2 对顾客信息的操作（增、删、改）

在所有顾客信息部分我们可以对顾客的信息进行增加、删除、和修改的操作。下面，我们以删除顾客信息的操作进行详细讲解。在所有顾客信息窗口中点击要删除的用户，就可以在当前顾客信息中看到顾客的具体信息，此时，点击删除顾客信息，就会弹出操作确认的信息窗口，避免使用者错误操作，造成用户信息的丢失。若该操作无误，则点击确认即可删除当前所选的顾客的信息；点击顾客之后即可在当前顾客信息处完成对顾客信息的修改；点击新增顾客后输入当前顾客信息即可完成新增顾客的操作。



6.7.3 光顾功能

选中顾客信息后，点击光顾即可为顾客添加新的订单。在当前顾客信息下面点击光顾按键，则会弹出房间和餐桌信息界面，此界面下的操作和介绍我们在之前都有介绍，在此处就不再赘述

6.7.4 光顾历史记录

在所有顾客信息部分，点击某个顾客。在光顾记录部分会显示该顾客在本餐厅的所有的订单情况。以及该顾客的所有点菜历史，其中包括每种菜被点的次数和该顾客对这种菜的评价信息。该部分主要被用于数据库有用信息的挖掘，了解顾客对每种菜的喜爱程度，从而做到为顾客推荐本店所添加的新的菜色，以及对餐厅未来菜色的发展策略提供理论依据。

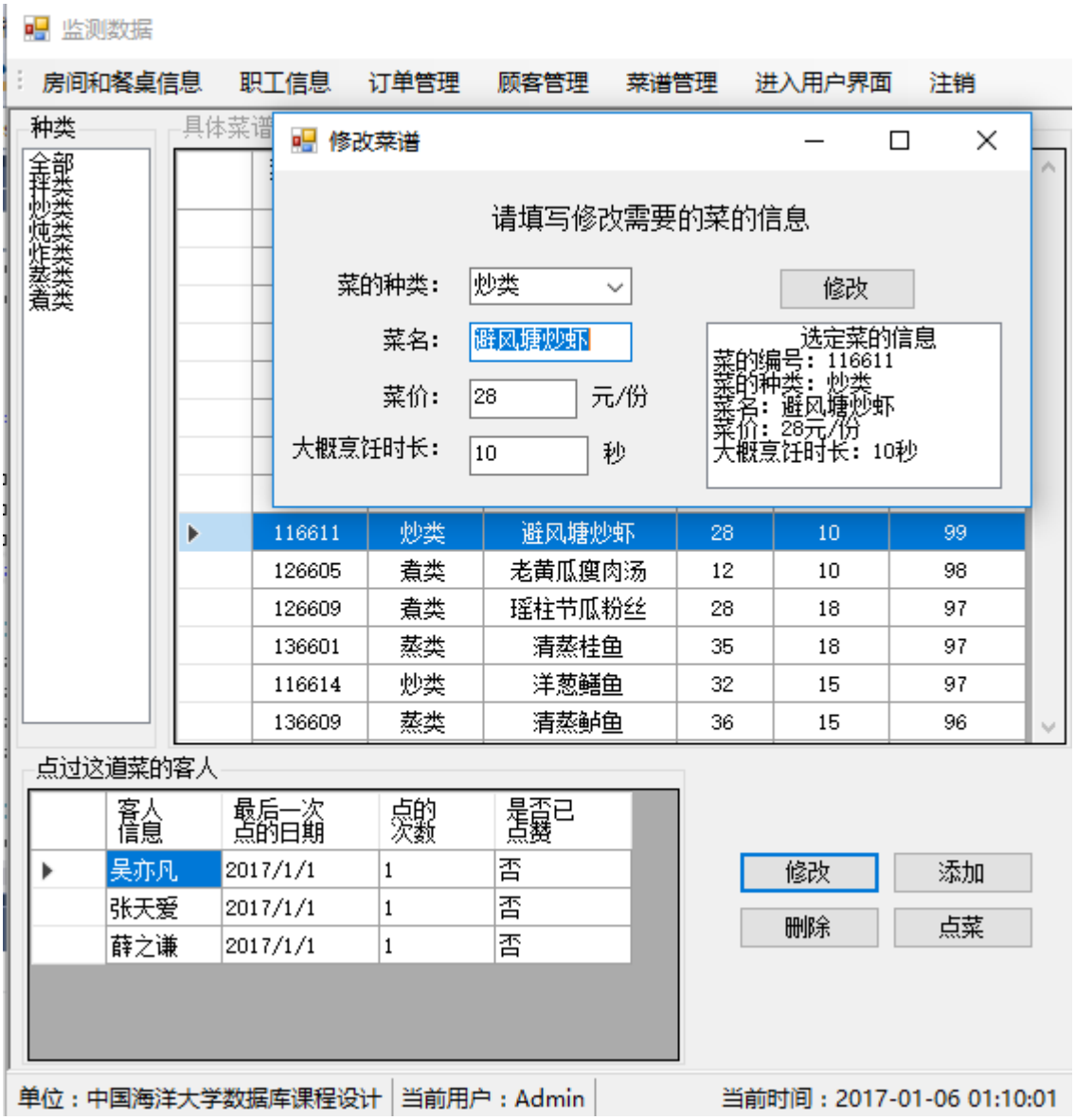
6.8 菜谱管理

6.8.1 界面介绍

菜谱管理主要是为了完成对餐厅菜谱的菜进行增加、删除、和修改而创建的，该页面除了完成最初设立的功能外，还可以根据不同菜的具体信息对菜进行查找。同时还可以查询点过这道菜的顾客都有哪些。最主要的是该也也可以调用已有的用户信息为用户点菜提供便利。

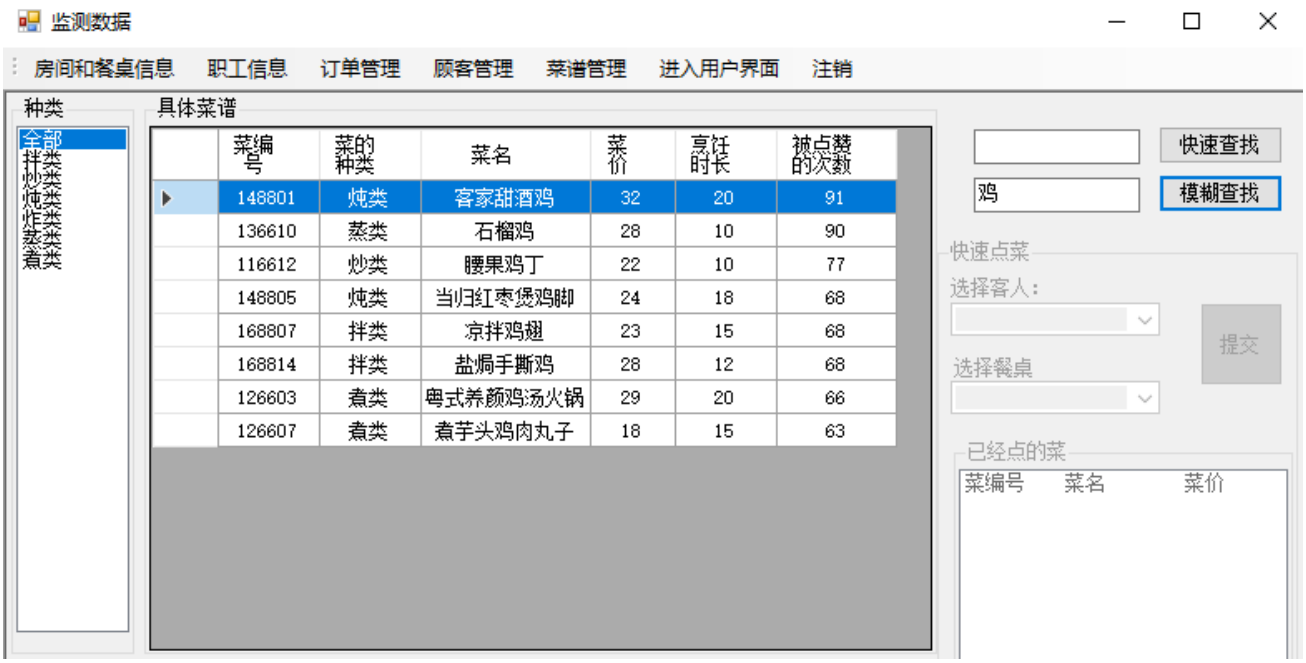
6.8.2 对菜谱的基本操作（增、删、改）

在菜谱中点击一种菜之后点击修改，会弹出图中所示的窗口，使用者只要输入所要修改的信息之后，再次点击修改菜谱窗口中的修改，即可确认对菜的信息进行修改，此时菜谱中就会显示修改后的信息。点击添加按键，录入菜的信息后再次确认即可完成对菜的添加；对菜的删除，选中一种菜后点击删除再次确认即可完成对菜的删除。



6.8.3 快速查找和模糊查找

快速查找和模糊查找，是为了避免客户点菜时总是要翻阅全部菜单浪费时间而设计的。因为快速查找只要输入菜的名字就可以对菜进行定位比较简单，所以这里我们对它不再做过多的介绍；模糊查找用户只要输入部分信息，系统既可以对数据库进行访问，然后返回能够匹配到用户输入的关键字的菜。



6.8.4 快速下单

快速下单是为了方便用户的点菜而设计的。只要在菜谱管理页面，点击“点菜”，即可选择数据库中已有的顾客，为他选择餐桌。在该页面直接就可以为顾客点菜。因为具体点菜操作在前面我们都已经详细介绍过了，在这里我们就不再对其进行赘述。
等。

7 体会

一开始走入了一个误区，那就是为了省空间而尽量少的建立表，其实这是一个极端。举个例子，当时我一致认为顾客的订单中的菜可以通过视图来查看，因为视图在数据库中属于外模式，不会占用磁盘存储空间，而且能用有效地限制用户的权限，而视图名就是订单号，这样通过订单号就能对订单进行操作。实际上这个想法是不好的，因为订单承载了很多信息，从系统可拓展性来说，这是不利的，而且实际上对视图的操作不仅仅会限制用户的操作，也会限制自身编程的操作。因此我们前后期大部分修改了我们的需求分析和概念结构设计。

对于数据库的建立，我深深地每个表建立一个唯一的 id 标识是很重要的，这个 id 标识是主属性。即使实体中原本的属性就有能唯一标识元组的属性，但也不应该直接设置为主码，而应该再设置一个 id，使用关系数据库自身提供的自增长类型即可，这样无论什么用户都不能对这个 id 修改，插入元组时数据库会自动生成。这样做的好处是因为主码的值是可能修改的，例如顾客的联系方式，没有 id 的话不利于参照完整性的实现，尤其是修改操作。

本次实验的核心工作是高级语言 C#编码与数据库的连接，之前没接触过 C#，纯属是现学现用，感觉学到了很多，也提高了自己的自学能力。同时让我基本入门了一门语言，C#真是一门有趣的高级语言。也就是在编程的时候和书上所学的知识结合起来，才进一步领略到数据库的重要性和数据库的应用，以及进一步理解书上的知识，例如游标的概念，其实和高级语言中的流是一个道理的，这种东西在使用时有一定的规则需要注意，例如不能两个同一对象流一起使用等。实验使我对高级语言的编写和调试以及一个项目的过程，

数据库的设计都有了进一步的掌握。实际应用中的数据库设计会比我们课程设计要复杂得多，这次实验也让我们对流程有所了解，方便日后的深入学习。

在实际编程时候，sql 语句往往是动态的，需要根据动态参数设定，这当中最容易出错的是字符串，由于 sql sever 查询语句中的字符串时用单引号的，但是高级语言一般都是双引号的，这方面容易出错。在者就是再编辑 sql 语句的时候容易把单引号漏掉，导致数据库操作异常。该问题本质上还是 sql 语句的练习不够多而造成的，因此平常要注意多加练习。

当然我也认识到我们的系统还是有很多不足，除了数据库设计上的不足外，还有系统本身功能的不足，例如顾客如果订很多张餐桌的话，每个单子的菜必须一致，且确定订单和结账都要操作多次，不利于用户体验，再者不能重复点菜。各个表的主码中，只有订单号是系统自动生成，其他均由顾客自己输入，不利于管理。许多过程不可逆，例如无法没有取消订单功能等。还有很多细节上的还有缺陷，日后应当多加学习。