目录

一、需求分析

1. 功能需求

- 顾客：登录/注册；查看所有餐厅及菜单；下单、取消订单、标记完成订单、支付订单；查看未完成订单和已完成订单；生成订单记录；修改并保存个人信息
- 餐厅：登录/注册；查看待出餐订单；接单、拒单、出餐；修改或添加菜单；查看并删除订单记录；修改并保存餐厅信息
- 骑手：登录/注册；查看并接单；查看未完成订单并完成订单；查看已完成订单并删除记录；修改并保存个人信息
- 管理员：登录/注册；查看所有表；删除记录

2. 非功能需求

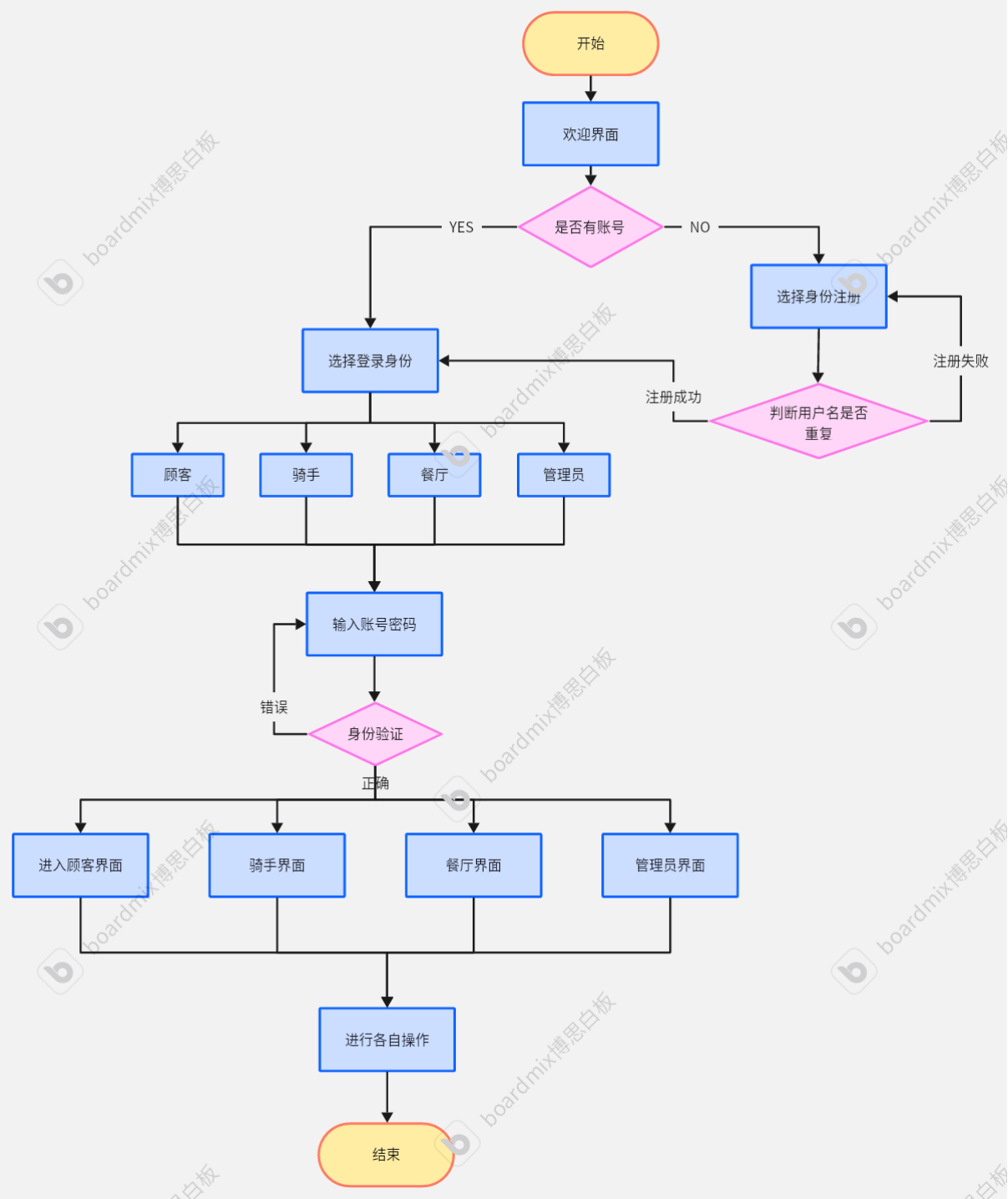
- 系统应具有高可靠性和可用性
- 系统应具备良好的安全性，保护用户数据
- 界面应友好，操作简便

二、系统设计

1. 登录注册模块

用户可以选择自己的身份进行登陆注册，登陆成功后可以进入各自的身份界面。
概览图

流程图

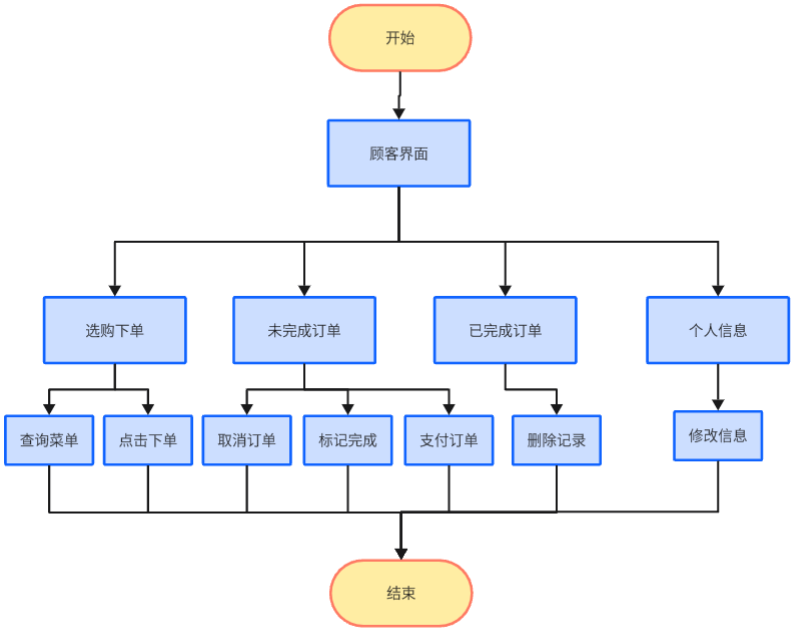


2. 顾客界面模块

顾客界面共有四个小界面（选购下单、未完成订单、已完成订单、个人信息），选购下单界面可以根据餐厅序号，自动在下拉框中填充菜单，顾客选择单品数量自动计算价格，完成下单；未完成订单界面可以查看订单并选择取消订单、标记完成、支付订单；已完成订单界面可以查看订单记录并选择删除；个人信息界面可以查看并修改个人信息。

概览图

流程图

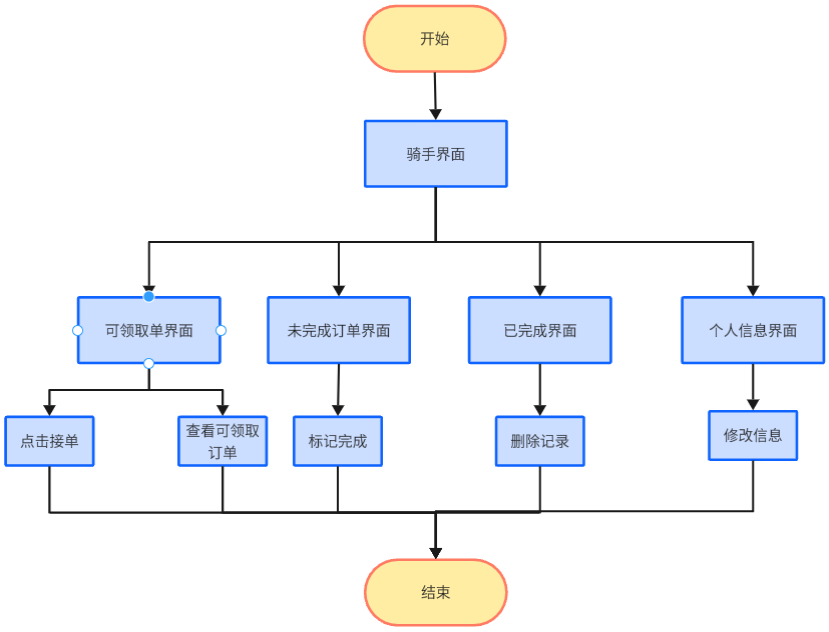


3. 骑手界面模块

骑手界面共有四个分界面（可领取订单界面、未完成订单界面、已完成订单界面、个人信息界面），其中可领取订单界面可以查看可以领取的订单并接单；未完成订单界面可以查看自己已领取但未完成的订单并标记完成；已完成订单界面可以查看已经派送完成的订单记录并删除；个人信息界面可以查看并修改个人信息。

概览图

流程图

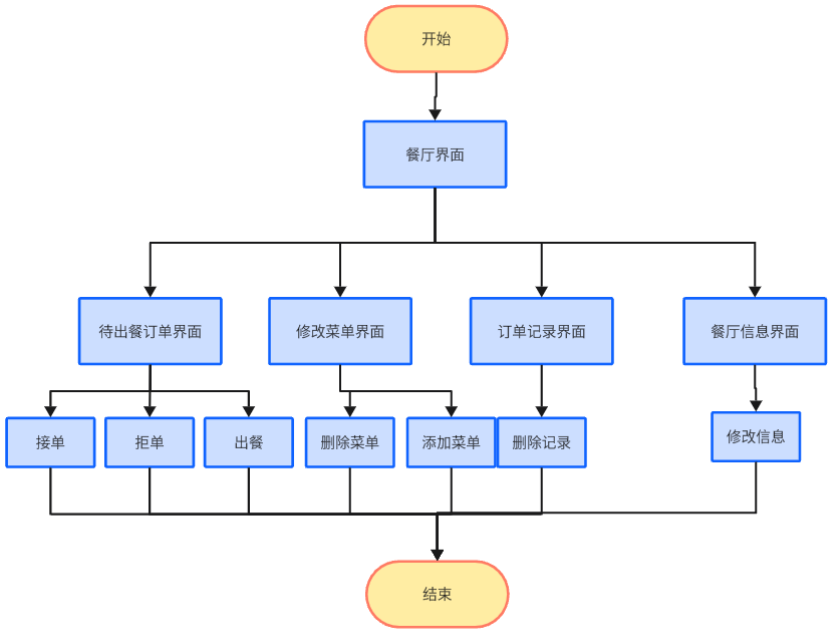


4. 餐厅界面模块

餐厅界面共有四个小界面（待出餐订单、修改订单、订单记录、餐厅信息），其中待出餐界面可以查看接收到的订单并选择接单或拒单或出餐；修改菜单界面可以查看自己的菜单并选择删除菜单或添加菜单；订单记录界面可以查看本餐厅的订单记录并删除记录；餐厅信息界面可以查看并修改餐厅的信息。

概览图

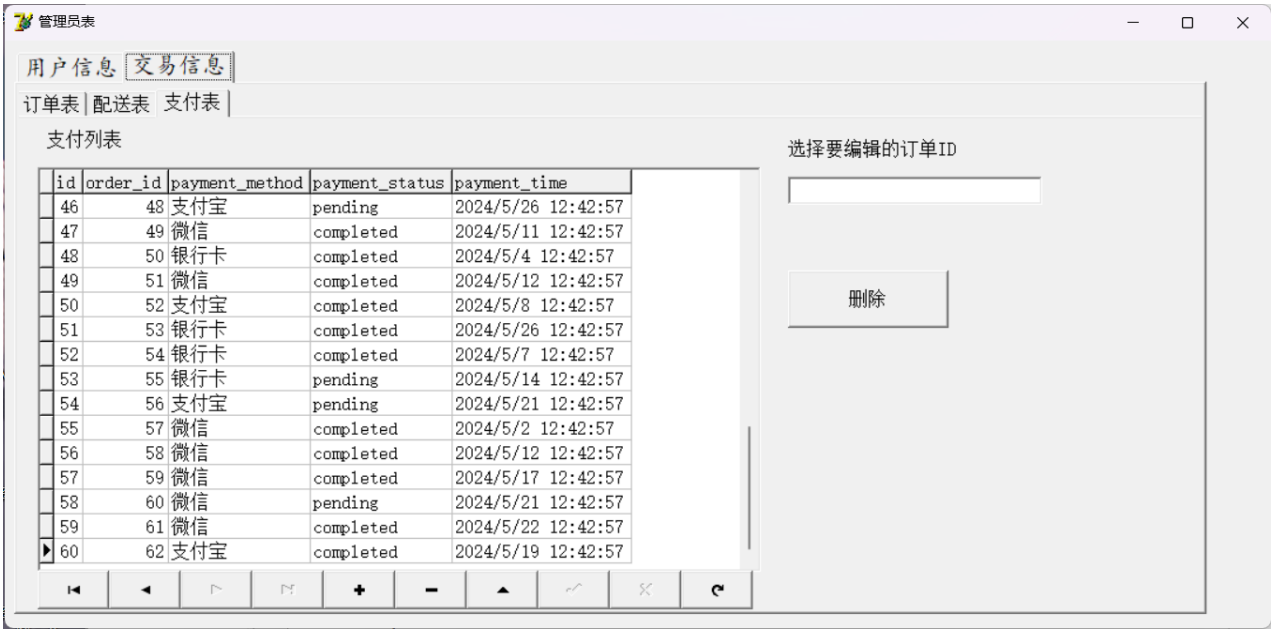
流程图



5. 管理员界面模块

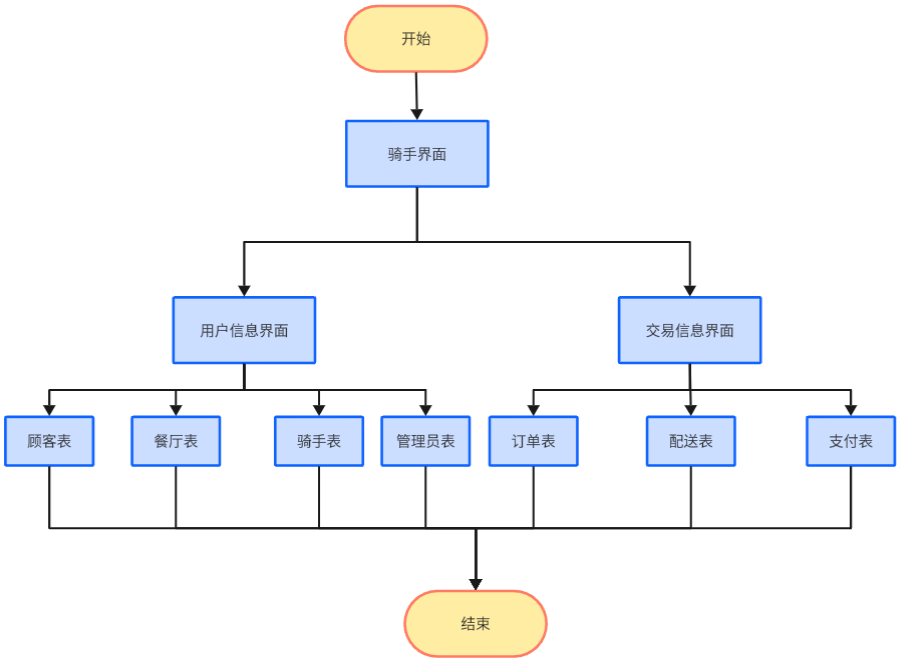
管理员界面可以查看所有的表信息，并且可以删除任意表的信息。界面主要分为用户信息和交易信息，其中用户信息分为顾客表、餐厅表、骑手表、管理员表；交易信息分为订单表、配送表、支付表。

概览图



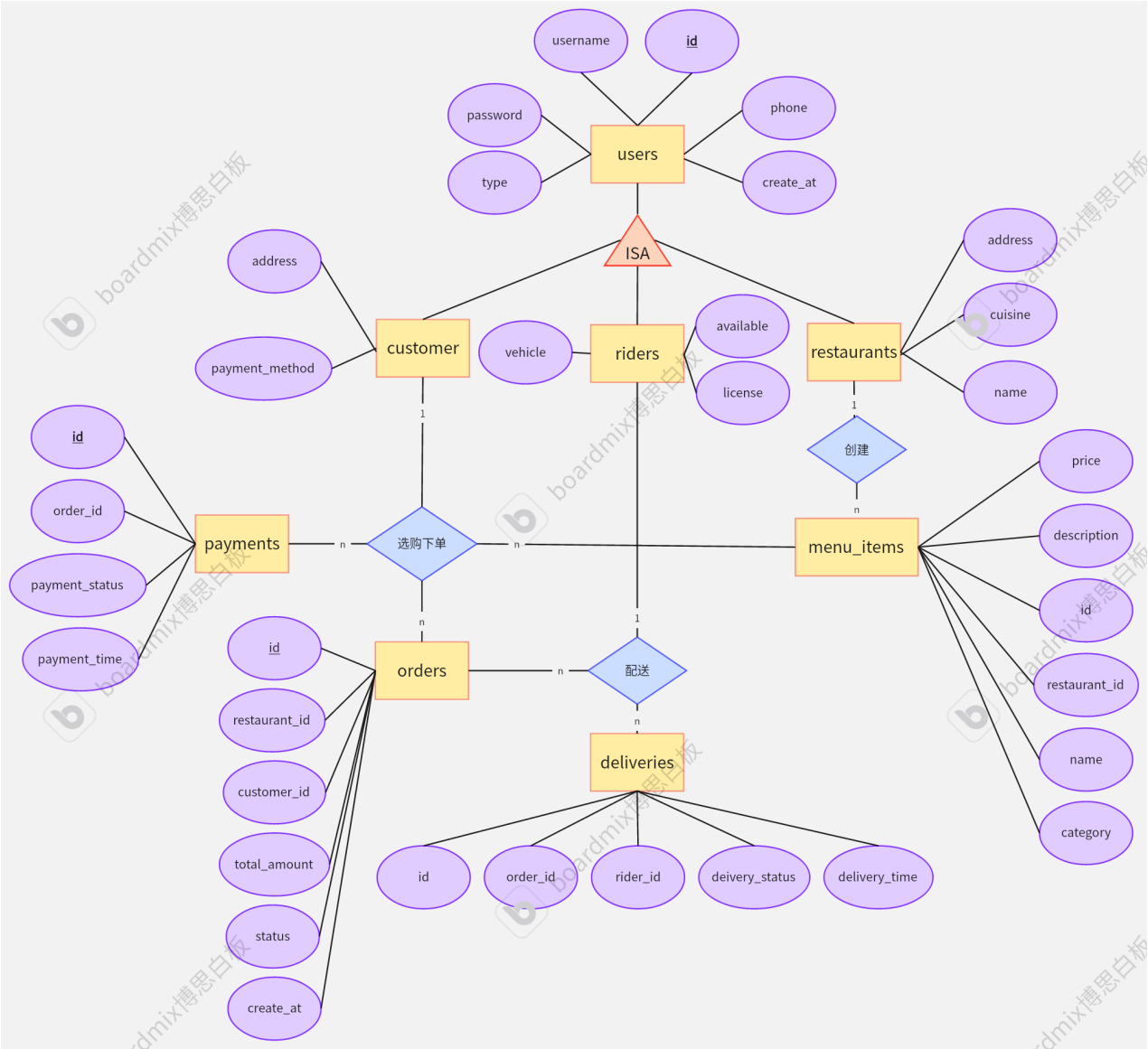
id	order_id	payment_method	payment_status	payment_time
46	48	支付宝	pending	2024/5/26 12:42:57
47	49	微信	completed	2024/5/11 12:42:57
48	50	银行卡	completed	2024/5/4 12:42:57
49	51	微信	completed	2024/5/12 12:42:57
50	52	支付宝	completed	2024/5/8 12:42:57
51	53	银行卡	completed	2024/5/26 12:42:57
52	54	银行卡	completed	2024/5/7 12:42:57
53	55	银行卡	pending	2024/5/14 12:42:57
54	56	支付宝	pending	2024/5/21 12:42:57
55	57	微信	completed	2024/5/2 12:42:57
56	58	微信	completed	2024/5/12 12:42:57
57	59	微信	completed	2024/5/17 12:42:57
58	60	微信	pending	2024/5/21 12:42:57
59	61	微信	completed	2024/5/22 12:42:57
60	62	支付宝	completed	2024/5/19 12:42:57

流程图



三、数据库设计

1. ER图



2. 数据库表结构设计

1. users 表

```sql
CREATE TABLE users (
    id          SERIAL PRIMARY KEY,                  -- id
    username    VARCHAR(50) NOT NULL,                -- 用户名
    password    VARCHAR(255) NOT NULL,               -- 密码
    phone       VARCHAR(20),                         -- 电话
    type        VARCHAR(10) CHECK (type IN ('customer', 'rider', 'restaurant', 'admin'))
NOT NULL,  -- 权限
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  -- 创建时间
);
```

字段名	字段含义	字段类型	字段长度	NULL	备注

字段名	字段含义	字段类型	字段长度	NULL	备注
id	ID	SERIAL			PK
username	用户名	字符	50		NOT NULL
password	密码	字符	255		NOT NULL
phone	电话	字符	20	√	
type	权限	字符	10		CHECK (type IN ('customer', 'rider', 'restaurant', 'admin')) NOT NULL
created_at	创建时间	时间戳			DEFAULT CURRENT_TIMESTAMP

2. customers 表

```
CREATE TABLE customers (
    id             INT PRIMARY KEY, -- id
    address        VARCHAR(255),    -- 地址
    payment_method VARCHAR(50),     -- 支付方式
    FOREIGN KEY (id) REFERENCES users (id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

字段名	字段含义	字段类型	字段长度	NULL	备注
id	ID	整数			PK, FK (REFERENCES users(id))
address	地址	字符	255	√	
payment_method	支付方式	字符	50	√	

3. riders 表

```sql
CREATE TABLE riders (
    id        INT PRIMARY KEY,       -- id
    vehicle   VARCHAR(50),           -- 车辆类型
    license   VARCHAR(20),           -- 车牌号
    available BOOLEAN DEFAULT TRUE, -- 工作状态
    FOREIGN KEY (id) REFERENCES users (id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

字段名	字段含义	字段类型	字段长度	**NULL**	备注
id	ID	整数			PK, FK (REFERENCES users(id))
vehicle	车辆类型	字符	50	√	
license	车牌号	字符	20	√	
available	工作状态	布尔			DEFAULT TRUE

4. restaurants 表

```sql
CREATE TABLE restaurants (
    id      INT PRIMARY KEY,         -- id
    name    VARCHAR(100) NOT NULL,  -- 名称
    address VARCHAR(255),           -- 地址
    phone   VARCHAR(20),            -- 电话
    cuisine VARCHAR(50),            -- 菜系
    FOREIGN KEY (id) REFERENCES users (id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

字段名	字段含义	字段类型	字段长度	**NULL**	备注
id	ID	整数			PK, FK (REFERENCES users(id))
name	名称	字符	100		NOT NULL
address	地址	字符	255	√	
phone	电话	字符	20	√	
cuisine	菜系	字符	50	√	

5. menu_items 表

```
CREATE TABLE menu_items (
    id            SERIAL PRIMARY KEY,          -- id
    restaurant_id INT,                         -- 餐厅id
    name          VARCHAR(100) NOT NULL,       -- 菜名
    description   VARCHAR(255),                -- 描述
    price         DECIMAL(10, 2) NOT NULL,     -- 价格
    category      VARCHAR(50),                 -- 品类
    FOREIGN KEY (restaurant_id) REFERENCES restaurants (id) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

字段名	字段含义	字段类型	字段长度	**NULL**	备注
id	ID	SERIAL			PK
restaurant_id	餐厅ID	整数		√	FK (REFERENCES restaurants(id))
name	菜名	字符	100		NOT NULL
description	描述	字符	255	√	
price	价格	小数	(10, 2)		NOT NULL
category	品类	字符	50	√	

6. orders 表

```
CREATE TABLE orders (
    id            SERIAL PRIMARY KEY,  -- id
    customer_id   INT,                 -- 顾客id
    restaurant_id INT,                 -- 餐厅id
    total_amount  DECIMAL(10, 2) NOT NULL, -- 总价
    status        VARCHAR(20) CHECK (status IN ('placed', 'preparing', 'delivering',
'completed', 'cancelled')) NOT NULL, -- 订单状态
    created_at    TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- 下单时间
    FOREIGN KEY (customer_id) REFERENCES customers (id) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (restaurant_id) REFERENCES restaurants (id) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

字段名	字段含义	字段类型	字段长度	**NULL**	备注

字段名	字段含义	字段类型	字段长度	NULL	备注
id	ID	SERIAL			PK
customer_id	顾客ID	整数		√	FK (REFERENCES customers(id))
restaurant_id	餐厅ID	整数		√	FK (REFERENCES restaurants(id))
total_amount	总价	小数	(10, 2)		NOT NULL
status	订单状态	字符	20		CHECK (status IN ('placed', 'preparing', 'delivering', 'completed', 'cancelled')) NOT NULL
created_at	下单时间	时间戳			DEFAULT CURRENT_TIMESTAMP

7. deliveries 表

```
CREATE TABLE deliveries (
    id              SERIAL PRIMARY KEY, -- id
    order_id        INT, -- 订单号
    rider_id        INT, -- 骑手号
    delivery_status VARCHAR(20) CHECK (delivery_status IN ('pending', 'in_transit',
'delivered')) NOT NULL, -- 配送状态
    delivery_time   TIMESTAMP, -- 配送时间
    menu_item_name  VARCHAR(255) NOT NULL, -- 菜品名称
    quantity        INT NOT NULL, -- 数

量
    price           DECIMAL(10, 2) NOT NULL, -- 价格
    FOREIGN KEY (order_id) REFERENCES orders (id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (rider_id) REFERENCES riders (id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

字段名	字段含义	字段类型	字段长度	NULL	备注
id	ID	SERIAL			PK
order_id	订单号	整数		√	FK (REFERENCES orders(id))
rider_id	骑手号	整数		√	FK (REFERENCES riders(id))
delivery_status	配送状态	字符	20		CHECK (delivery_status IN ('pending', 'in_transit', 'delivered')) NOT NULL
delivery_time	配送时间	时间戳		√	
menu_item_name	菜品名称	字符	255		NOT NULL
quantity	数量	整数			NOT NULL
price	价格	小数	(10, 2)		NOT NULL

8. payments 表

```
CREATE TABLE payments (
    id             SERIAL PRIMARY KEY, -- id
    order_id       INT,                -- 订单号
    payment_method VARCHAR(50),        -- 支付方式
    payment_status VARCHAR(20) CHECK (payment_status IN ('pending', 'completed',
'failed')) NOT NULL, -- 支付状态
    payment_time   TIMESTAMP,          -- 支付时间
    FOREIGN KEY (order_id) REFERENCES orders (id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

字段名	字段含义	字段类型	字段长度	NULL	备注
id	ID	SERIAL			PK
order_id	订单号	整数		√	FK (REFERENCES orders(id))

字段名	字段含义	字段类型	字段长度	NULL	备注
payment_method	支付方式	字符	50	√	
payment_status	支付状态	字符	20		CHECK (payment_status IN ('pending', 'completed', 'failed')) NOT NULL
payment_time	支付时间	时间戳		√	

3. 范式证明

1. 第一范式（1NF）:第一范式要求表中的每个列都是不可分割的基本数据项，即每个字段都是原子性的。在这个设计中：

   - 所有的 `id` 字段都是不可分的基本数据类型，如 `SERIAL` 、 `INT` 。
   - `username` 、 `password` 、 `phone` 、 `type` 、 `address` 、 `payment_method` 等字段都是直接的字符串或布尔值，没有复合信息。
   - `created_at` 、 `delivery_time` 、 `payment_time` 等时间戳也是不可分的。

   因此，所有表的设计都符合第一范式的要求。

2. 第二范式（2NF）:第二范式要求表中的非主键字段完全依赖于主键，而不能只依赖于主键的一部分。在这个设计中：

   - `users` 表的非主键字段（ `username` 、 `password` 、 `phone` 、 `type` 、 `created_at` ）都直接依赖于主键 `id` 。
   - `customers` 、 `riders` 、 `restaurants` 、 `admins` 表的非主键字段也都完全依赖于外键 `id` ，而这些外键作为主键是唯一的。
   - 其他关联表（如 `menu_items` 、 `orders` 、 `deliveries` 、 `payments` ）的非主键字段也直接依赖于各自的主键或外键。

   因此，所有的表设计均满足第二范式的要求。

3. 第三范式（3NF）:第三范式要求表中的非主键字段之间不存在传递依赖。即如果非主键字段A依赖于主键，非主键字段B依赖于A，则违反了3NF，除非这种依赖是函数依赖且是必要的。

- 在这个设计中，我们没有看到明显的传递依赖。例如，在 `menu_items` 表中，虽然 `price` 可能间接通过 `restaurant_id` 依赖于 `restaurants` 表中的信息，但这是合理的关联，而不是不必要的传递依赖。
- 类似地， `orders` 表中的 `total_amount` 直接依赖于订单本身的明细（虽然明细未直接体现在表中，但计算逻辑不构成传递依赖）。
- `deliveries` 表中的 `delivery_status` 和 `delivery_time` 直接依赖于配送记录本身，而非其他非主键字段。
- `payments` 表的字段依赖关系同样直接且合理。

所以，数据库设计遵循了第三范式的要求，没有发现非主键字段间的传递依赖。

四、关键源码说明

1. 开发环境

开发工具：Delphi7
数据库：GaussDB(华为云)
操作系统：Windows 11

2. 数据库连接

在用户通过sql脚本在本地实现数据库备份并在 `connection.ini` 文件中修改好配置信息后，程序将会读取 `connection.ini` 文件的配置信息并自动生成 `connectionString` 并赋值给ADOConnection的ConnectionString属性并与数据库建立相应的连接，具体代码如下：

```
procedure TmainForm.FormCreate(Sender: TObject);
var
  IniFile: TIniFile;
  CString: string;
  IniFilePath: string;
begin
    IniFilePath := ExtractFilePath(ParamStr(0)) + 'connection.ini';
    showMessage(IniFilePath);
    IniFile:= TIniFile.Create(IniFilePath);
    CString := 'Provider=' + IniFile.ReadString('Database', 'Provider', '') + ';' +
                           'Password=' + IniFile.ReadString('Database', 'Password', '') +
';' +
                           'Persist Security Info=' + IniFile.ReadString('Database',
'Persist Security Info', '') + ';'+
                           'User ID=' + IniFile.ReadString('Database', 'User ID', '') +
';' +
                           'Data Source=' + IniFile.ReadString('Database', 'Data Source',
'') + ';' +
```

```
                        'Initial Catalog=' + IniFile.ReadString('Database', 'Initial
Catalog', '');
    showMessage(CString);
    ADOConnection1.ConnectionString := CString;
    ADOConnection1.Connected := True; // 建立连接
end;
```

connection.ini

C: > Users > xuwen > Desktop > ASS3-2024-1120221397-徐文彬 > 源代码及可执行文件 > ASS3 > connection.ini

```
[DataBase]

; Provider指定了数据提供者
Provider=MSDASQL.1;
; 数据库登录密码
Password=dboper@123;
; 是否保持安全信息（如密码）的持久性
Persist Security Info=True;
; 数据库登录用户名
User ID=dboper;
; 数据源名称，指向特定的数据库服务器实例
Data Source=PostgreSQL35W;
; 初始目录或数据库名称
Initial Catalog=ass3;
```

3. 登录验证

当用户点击自己的身份开始登陆后，unit1.userClass会根据身份被赋予不同的值，接着根据userclass的值进行不同表信息的检验(case unit1.userClass of)，如果登陆成功，会根据用户的身份进入到不同的界面(顾客界面、餐厅界面、骑手界面、管理员界面)，登陆失败则会显示`用户名或密码错误`，具体代码如下：

```
procedure TloginForm.Button1Click(Sender: TObject);
begin
  case unit1.userClass of
    1:
      begin
      query:=Format('SELECT id FROM users WHERE username = ''%s'' AND password = ''%s''
and type=''customer''',[Edit1.Text,Edit2.Text]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      end;
    2:
      begin
      query:=Format('SELECT id FROM users WHERE username = ''%s'' AND password = ''%s''
and type=''restaurant''',[Edit1.Text,Edit2.Text]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      end;
    3:
      begin
```

```
      query:=Format('SELECT id FROM users WHERE username = ''%s'' AND password = ''%s''
and type=''rider''',[Edit1.Text,Edit2.Text]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      end;
    4:
      begin
      query:=Format('SELECT id FROM users WHERE username = ''%s'' AND password = ''%s''
and type=''admin''',[Edit1.Text,Edit2.Text]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      end;
  end;

  // 获取查询结果
  if not ADOQuery1.Eof then
      begin
        user_id:=ADOQuery1.FieldByName('id').AsInteger;
        username:=Edit1.Text;
        showMessage('登陆成功');
        loginForm.Hide;
        case unit1.userClass of
          1:customerForm.showModal;
          2:cookForm.showModal;
          3:riderForm.showModal;
          4:adminForm.showModal;
        end;
      end
  else
      showMessage('用户名或密码错误');
end;
```

4. 分页界面**(**以顾客界面为例**)**

使用PageControl组件，首先创建需要的界面并修改Caption，接着在PageControlChange事件中根据ActivePageIndex的不同再触发不同的事件，例如在顾客界面中当ActivePage为选购下单页面时就会查询所有的餐厅并显示在DBGrid中，具体代码如下：

```
procedure TcustomerForm.PageControl1Change(Sender: TObject);
begin
  case PageControl1.ActivePageIndex of
    0:
      begin
      query:=Format('Select * From restaurants',[]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      AutoFitColumns(DBGrid1, DataSource1);
      end;
    1:
```

```
      begin
      query:=Format('Select * From user_unpaid_orders where customer_id = %d and
order_status in(''placed'',''preparing'',''delivering'') or payment_status=''pending''',
[unit2.user_id]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      AutoFitColumns(DBGrid2, DataSource1);
      end;
    2:
      begin
      query:=Format('Select * From orders where customer_id = %d and
status=''completed''',[unit2.user_id]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      AutoFitColumns(DBGrid3, DataSource1);
      end;
    3:
      begin
      infoNameEdit.Text:=unit2.username;
      query:=Format('Select * From customers where id=%d',[unit2.user_id]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      infoAddressEdit.Text:=ADOQuery1.FieldByName('address').AsString;
      infoPaymentEdit.Text:=ADOQuery1.FieldByName('payment_method').AsString;
      end;
  end;
end;
```

5. 顾客页面菜单自动填充

Edit1用来填要查询的餐厅名称，每当Edit1改变时，都会触发向菜单下拉框中填充菜单的事件，这样就能做到实时更新菜单，实现级联操作，代码如下：

```
procedure TcustomerForm.Edit1Change(Sender: TObject);
var
  resId:Integer;
begin
  if(Edit1.Text='') then
  begin
    resId:=1029;
  end
  else
  begin
    resId:=StrToInt(Edit1.Text);
  end;
  query:=Format('Select * From menu_items where restaurant_id= %d',[resId]);
  ADOQuery2.SQL.Text:=query;
  ADOQuery2.Open;
  // 清空ComboBox
```

```
  ComboBox1.Clear;

  // 将查询结果添加到ComboBox中
  while not ADOQuery2.Eof do
  begin
    ComboBox1.Items.Add(ADOQuery2.FieldByName('name').AsString);
    ADOQuery2.Next;
  end;
  // 关闭查询
  ADOQuery2.Close;
end;
```

6. 顾客页面价格自动计算

首先，当Edit2(填写单品数量)发生改变时，就会触发事件，查询Edit1对应餐厅中Combox1对应菜单的价格并于Edit2

中填写的单品数量做乘法，并将结果显示在Edit3中，这样就可以实时看到订单的总价格，实现级联操作，代码如下：

```
procedure TcustomerForm.Edit2Change(Sender: TObject);
begin
  query:=Format('Select * From menu_items where restaurant_id=%d and name=''%s''',
[StrToInt(Edit1.Text),ComboBox1.Text]);
  ADOQuery2.SQL.Text:=query;
  // 打开查询
  ADOQuery2.Open;

  // 检查是否找到记录
  if not ADOQuery2.Eof then
  begin
    // 显示找到的记录的其他字段值
    price := ADOQuery2.FieldByName('price').AsFloat;
    if edit2.Text ='' then
      num:=1
    else
      num:=StrToInt(Edit2.Text);
    prices:=price*num;
    Edit3.Text:=FloatToStrF(prices, ffFixed, 15, 2);;
  end
  else
  begin
    // 未找到记录，清空显示字段或显示提示
    Edit3.Text := '未找到记录';
  end;

  // 关闭查询
  ADOQuery2.Close;
end;
```

7.所有**DBGrid**组件的宽度自适应

定义procedure，每次当ADOQuery组件执行完SQL并且使用到DBGrid进行显示时执行该procedure。在procedure中首先会清除现有列，然后遍历每一个字段动态添加列并调整宽度（按照字段最大值的宽度+8），执行完后的DBGrid便可以最大限度的利用可视空间展示列表信息。

```
procedure AutoFitColumns(DBGrid: TDBGrid; DataSource: TDataSource);
var
  i, ColWidth, TitleWidth, DataWidth: Integer;
  Field: TField;
  DataSet: TDataSet;
begin
  DataSet := DataSource.DataSet;

  // 检查数据集是否为空或没有字段
  if (DataSet = nil) or (DataSet.FieldCount = 0) then
    Exit;

  // 清除现有列（防止索引超出范围）
  DBGrid.Columns.Clear;

  // 遍历每一个字段，动态添加列并调整宽度
  for i := 0 to DataSet.FieldCount - 1 do
  begin
    // 动态添加列
    with DBGrid.Columns.Add do
    begin
      FieldName := DataSet.Fields[i].FieldName;

      // 设置初始列宽为标题的宽度
      TitleWidth := DBGrid.Canvas.TextWidth(DataSet.Fields[i].DisplayName) + 8;
      ColWidth := TitleWidth;

      // 遍历每一条记录，计算字段值的最大宽度
      DataSet.DisableControls;
      try
        DataSet.First;
        while not DataSet.Eof do
        begin
          DataWidth := DBGrid.Canvas.TextWidth(DataSet.Fields[i].AsString) + 8;
          ColWidth := Max(ColWidth, DataWidth);
          DataSet.Next;
        end;
      finally
        DataSet.EnableControls;
      end;
```

```
      // 设置列宽
      Width := ColWidth;
    end;
  end;
end;
```

8.各用户个人信息修改**(**以餐厅界面为例**)**

首先从数据库中查询餐厅的个人信息，然后分别赋值显示到对应的组件上，然后用户可以在对应Edit组件直接进行编辑，编辑完成后点击保存按钮即可更新个人信息。

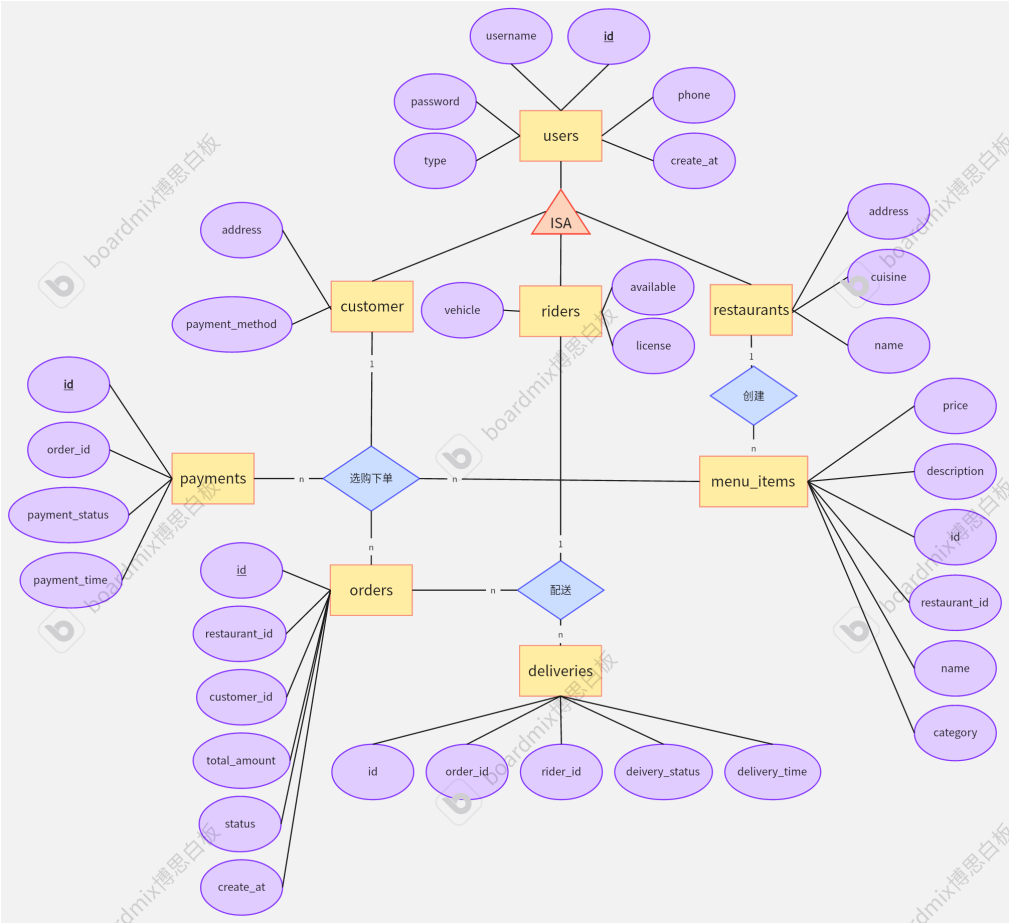
```
      query:=Format('Select * From restaurants where id=%d',[unit2.user_id]);
      ADOQuery1.SQL.Text:=Query;
      ADOQuery1.Open;
      infoNameEdit.Text:=ADOQuery1.FieldByName('name').AsString;
      infoAddressEdit.Text:=ADOQuery1.FieldByName('address').AsString;
      infoPhoneEdit.Text:=ADOQuery1.FieldByName('phone').AsString;
      infoCuisineEdit.Text:=ADOQuery1.FieldByName('cuisine').AsString;

procedure TcookForm.Button5Click(Sender: TObject);
begin
   query:=Format('Update restaurants set
name=''%s'',address=''%s'',phone=''%s'',cuisine=''%s'' where id=%d',
[infoNameEdit.Text,infoAddressEdit.Text,infoPhoneEdit.Text,infoCuisineEdit.Text,unit2.use
r_id]);
   ADOQuery1.SQL.Text:=query;
   ADOQuery1.ExecSQL;
end;
```

五、评分项实现

6张表

一共有用户表、顾客表、骑手表、餐厅表、菜单表、订单表、配送表、支付表8张表，符合。

ER图：



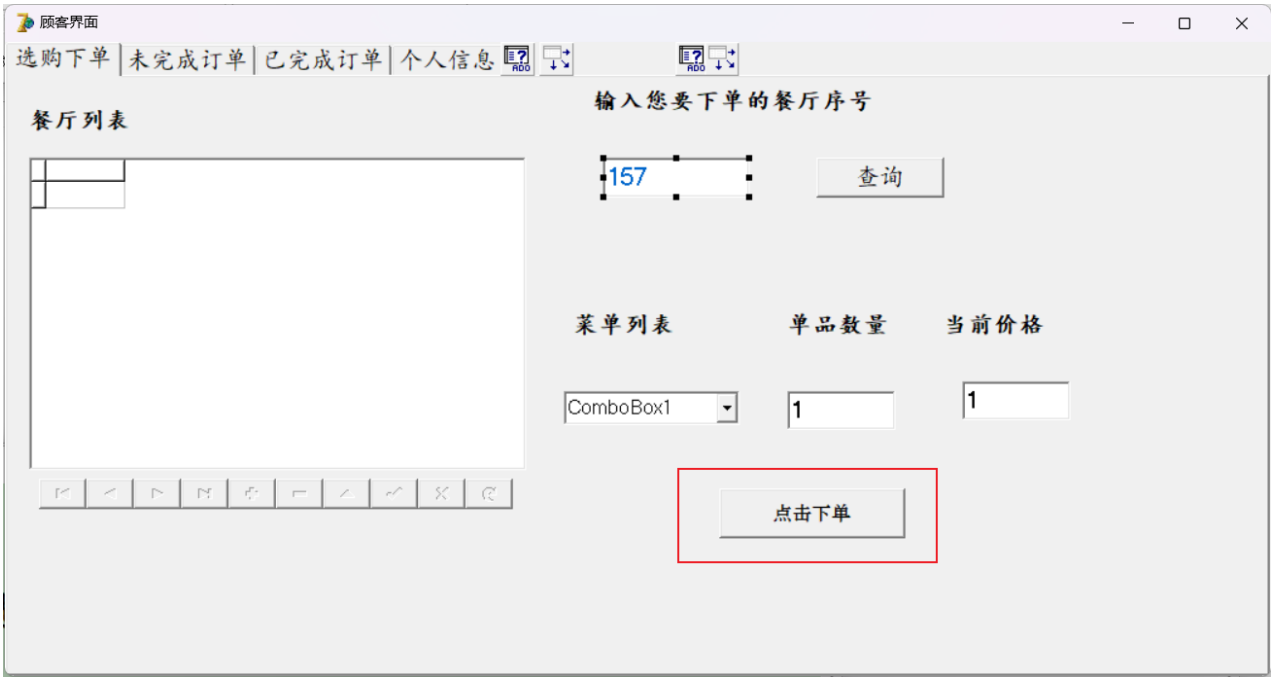
范式证明

上文已给出

表的增删改查

下面以订单表为例

下单即可增加新的订单

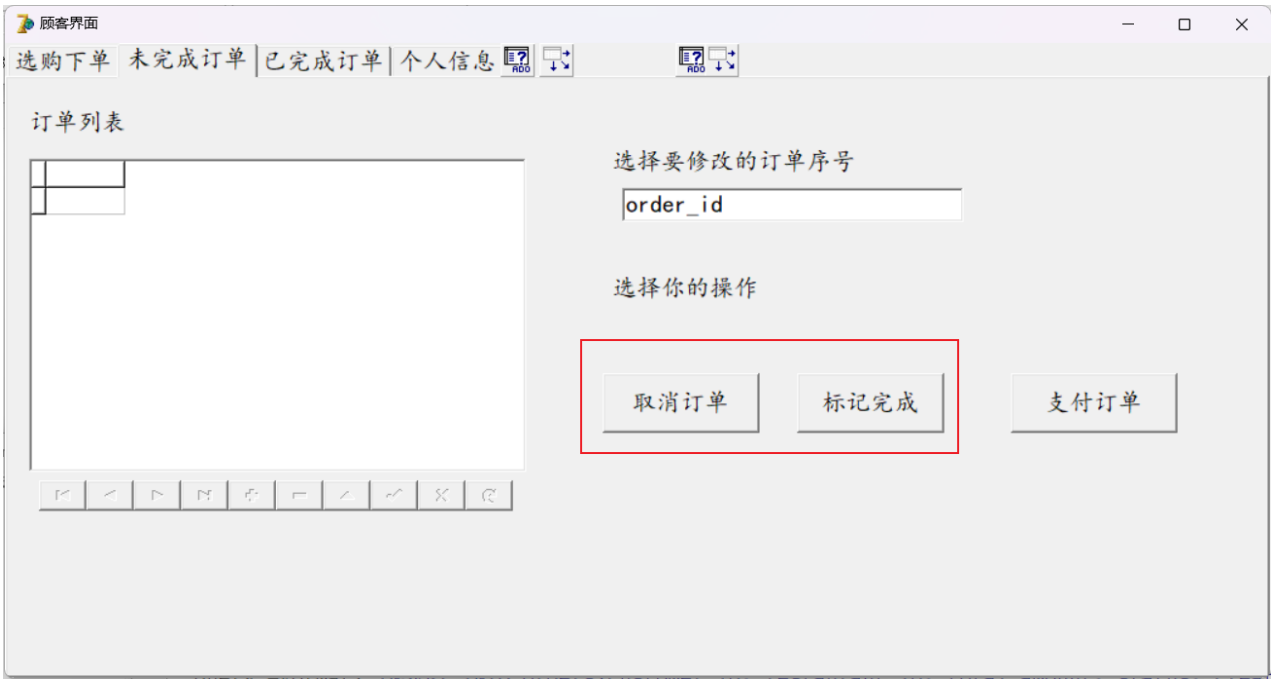



```
procedure TcustomerForm.Button1Click(Sender: TObject);
var
 orderId:Integer;
begin
  query:=Format('INSERT INTO orders(customer_id, restaurant_id, total_amount, status,
created_at) values (%d, %d, %f, ''%s'', ''%s'')',
[unit2.user_id,StrToInt(edit1.Text),StrToFloat(edit3.Text),'placed',DateTimeToStr(now)]);
  ADOQuery2.SQL.Text:=query;
  ADOQuery2.ExecSQL;
  ADOQuery2.SQL.Text:='Select Max(id) AS last_order_id From orders';
  ADOQuery2.open;
  orderId:=ADOQuery2.FieldByName('last_order_id').AsInteger;
  showMessage(IntToStr(orderId));
  query:=Format('insert into deliveries (order_id, delivery_status, menu_item_name,
quantity, price) values (%d,''pending'',''%s'',%d,%f)',
[orderId,combobox1.Text,num,prices]);
  ADOquery2.SQL.Text:=query;
  ADOQuery2.ExecSQL;
  showMessage('订单创建成功');
end;
```

删除订单记录实现删除操作


```
//删除订单
procedure TcustomerForm.Button6Click(Sender: TObject);
begin
  query:=Format('delete From orders where id=%d',[StrToInt(edit5.Text)]);
  ADOQuery2.SQL.Text:=query;
  ADOQuery2.ExecSQL;
  query:=Format('Select * From orders where customer_id = %d',[unit2.user_id]);
  ADOQuery1.SQL.Text:=query;
  ADOQuery1.Open;
  AutoFitColumns(DBGrid3, DataSource1);
end;
```

取消订单和标记完成按钮可以实现对订单状态的修改



```
//取消订单
procedure TcustomerForm.Button3Click(Sender: TObject);
begin
  query:=Format('update orders set status =''cancelled'' where id=%d',
[StrToInt(Edit4.Text)]);
  ADOQuery2.SQL.Text:=query;
  ADOQuery2.ExecSQL;
  query:=Format('Select * From orders where customer_id = %d and status=''placed'' or
status=''preparing'' or status=''delivering''',[unit2.user_id]);
  ADOQuery1.SQL.Text:=query;
  ADOQuery1.Open;
  AutoFitColumns(DBGrid2, DataSource1);
end;
```

```
//标记完成
procedure TcustomerForm.Button7Click(Sender: TObject);
begin
  query:=Format('Update users set username=''%s'' where id=%d',
[infoNameEdit.Text,unit2.user_id]);
  ADOQuery1.SQL.Text:=query;
  ADOQuery1.ExecSQL;
  query:=Format('Update customers set address=''%s'',payment_method=''%s'' where id=%d',
[infoAddressEdit.Text,infoPaymentEdit.Text,unit2.user_id]);
  ADOQuery1.SQL.Text:=query;
  ADOQUery1.ExecSQL;
end;
```

查询订单


```
procedure TcustomerForm.PageControl1Change(Sender: TObject);
begin
  case PageControl1.ActivePageIndex of
    0:
```

```
      begin
      query:=Format('Select * From restaurants',[]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      AutoFitColumns(DBGrid1, DataSource1);
      end;
    1:
      begin
      query:=Format('Select * From user_unpaid_orders where customer_id = %d and
order_status in(''placed'',''preparing'',''delivering'') or payment_status=''pending''',
[unit2.user_id]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      AutoFitColumns(DBGrid2, DataSource1);
      end;
    2:
      begin
      query:=Format('Select * From orders where customer_id = %d and
status=''completed''',[unit2.user_id]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      AutoFitColumns(DBGrid3, DataSource1);
      end;
    3:
      begin
      infoNameEdit.Text:=unit2.username;
      query:=Format('Select * From customers where id=%d',[unit2.user_id]);
      ADOQuery1.SQL.Text:=query;
      ADOQuery1.Open;
      infoAddressEdit.Text:=ADOQuery1.FieldByName('address').AsString;
      infoPaymentEdit.Text:=ADOQuery1.FieldByName('payment_method').AsString;
      end;
  end;
end;
```

功能需求、设计说明

功能需求

- 顾客：登录/注册；查看所有餐厅及菜单；下单、取消订单、标记完成订单、支付订单；查看未完成订单和已完成订单；生成订单记录；修改并保存个人信息
- 餐厅：登录/注册；查看待出餐订单；接单、拒单、出餐；修改或添加菜单；查看并删除订单记录；修改并保存餐厅信息
- 骑手：登录/注册；查看并接单；查看未完成订单并完成订单；查看已完成订单并删除记录；修改并保存个人信息
- 管理员：登录/注册；查看所有表；删除记录

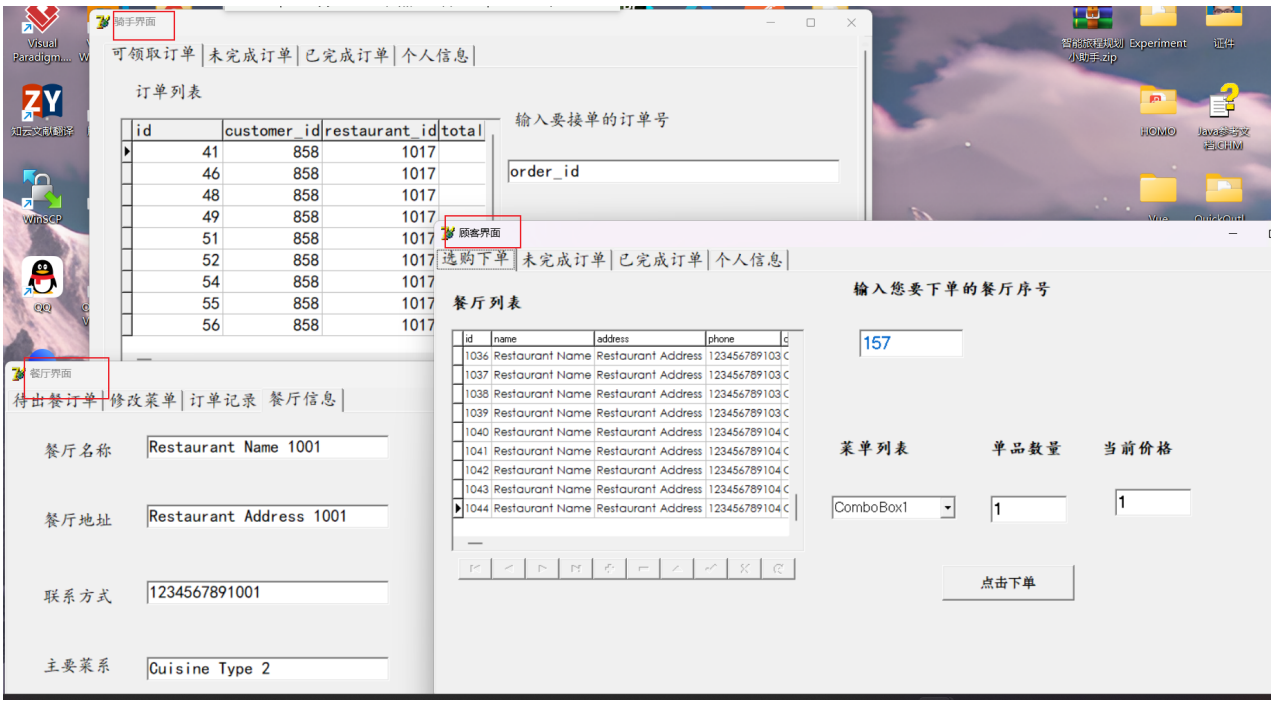
设计说明 ：见系统设计和数据库设计部分

有界面级联操作

当用户在餐厅序号中输入餐厅id时，会触发EditChange事件，程序会自动根据餐厅id进行相应的菜单查询并填充到菜单列表的下拉框中供用户选择，这是第一个级联操作；第二个级联操作为，当用户选好菜品，在单品数量中输入数字时，又会触发另一个EditChange事件，程序会自动查询该菜品的的单品价格并乘以数量并将结果显示在当前价格一栏中并且每次更改数量都会实时更新，这是第二个级联操作。

有数据库多个用户及连接：

如图所示，三个用户分别以顾客、餐厅、骑手身份同时连接登陆进行操作且正常工作，实现了多个用户同时连接。



有应用程序用户权限管理

本项目中共有四种不同的用户（顾客、餐厅、骑手、管理员），每种用户可以查看或操作的数据各不相同，实现了不同用户的权限管理。

视图、动态**SQL**

视图

```sql
--创建视图，显示用户未支付的订单
CREATE VIEW user_unpaid_orders AS
SELECT
    o.id AS order_id,
    o.customer_id,
    u.username AS customer_username,
    o.restaurant_id,
    r.name AS restaurant_name,
    o.total_amount,
    o.status AS order_status,
    p.payment_status,
    o.created_at
FROM
    orders o
JOIN
    customers c ON o.customer_id = c.id
JOIN
    users u ON c.id = u.id
JOIN
    restaurants r ON o.restaurant_id = r.id
LEFT JOIN
    payments p ON o.id = p.order_id
WHERE
    p.payment_status IS NULL OR p.payment_status != 'completed';
```

动态**SQL**

本项目使用了大量的动态sql执行查询操作，大大减少了ADOQuery组件的使用数量，使程序更加简便，下面以点击下单功能为例


```
procedure TcustomerForm.Button1Click(Sender: TObject);
var
  orderId: Integer;
  totalAmount: Double;
  restaurantId, customerId, quantity: Integer;
  menuItemName: string;
  status: string;
  createdAt: TDateTime;
begin
  // 获取输入的值
  customerId := unit2.user_id;
  restaurantId := StrToInt(Edit1.Text);
  totalAmount := StrToFloat(Edit3.Text);
  menuItemName := ComboBox1.Text;
  quantity := StrToInt(Edit2.Text); // 假设 Edit2 是数量输入框
  status := 'placed';
  createdAt := Now;

  // 插入订单记录
  ADOQuery2.SQL.Text := 'INSERT INTO orders (customer_id, restaurant_id, total_amount,
status, created_at) ' +
                        'VALUES (:customer_id, :restaurant_id, :total_amount, :status,
:created_at)';
  ADOQuery2.Parameters.ParamByName('customer_id').Value := customerId;
  ADOQuery2.Parameters.ParamByName('restaurant_id').Value := restaurantId;
  ADOQuery2.Parameters.ParamByName('total_amount').Value := totalAmount;
  ADOQuery2.Parameters.ParamByName('status').Value := status;
  ADOQuery2.Parameters.ParamByName('created_at').Value := createdAt;
  ADOQuery2.ExecSQL;

  // 获取刚插入的订单ID
  ADOQuery2.SQL.Text := 'SELECT MAX(id) AS last_order_id FROM orders';
```

```
  ADOQuery2.Open;
  orderId := ADOQuery2.FieldByName('last_order_id').AsInteger;
  ADOQuery2.Close;

  // 插入交付记录
  ADOQuery2.SQL.Text := 'INSERT INTO deliveries (order_id, delivery_status,
menu_item_name, quantity, price) ' +
                        'VALUES (:order_id, :delivery_status, :menu_item_name, :quantity,
:price)';
  ADOQuery2.Parameters.ParamByName('order_id').Value := orderId;
  ADOQuery2.Parameters.ParamByName('delivery_status').Value := 'pending';
  ADOQuery2.Parameters.ParamByName('menu_item_name').Value := menuItemName;
  ADOQuery2.Parameters.ParamByName('quantity').Value := quantity;
  ADOQuery2.Parameters.ParamByName('price').Value := totalAmount; // 假设价格与订单总金额相
同
  ADOQuery2.ExecSQL;

  // 显示订单创建成功的消息
  ShowMessage('订单创建成功');
end;
```

存储过程/函数、触发器

存储过程
该存储过程用于实现初始数据的批量导入

```
CREATE OR REPLACE PROCEDURE initialize_data()
LANGUAGE plpgsql
AS $$
BEGIN
  -- 为 users 表插入数据
  INSERT INTO users (username, password, phone, type, created_at)
  SELECT
      'user_' || s,
      'password_' || s,
      '123456789' || s,
      CASE
          WHEN s <= 100 THEN 'customer'
          WHEN s <= 150 THEN 'rider'
          WHEN s <= 200 THEN 'restaurant'
          ELSE 'admin'
      END,
      CURRENT_TIMESTAMP - INTERVAL '1' DAY * s
  FROM generate_series(1, 210) AS s;

  WITH numbered_users AS (
    SELECT id, row_number() OVER (ORDER BY id) AS rn
    FROM users
    WHERE password LIKE 'password_%'
```

```
)
UPDATE users
SET username = 'user_' || numbered_users.rn
FROM numbered_users
WHERE users.id = numbered_users.id;

-- 为 customers 表插入数据
DELETE FROM customers;
INSERT INTO customers (id, address, payment_method)
SELECT
    id,
    'Customer Address ' || id,
    CASE
        WHEN id % 3 = 0 THEN '支付宝'
        WHEN id % 3 = 1 THEN '微信'
        ELSE '银行卡'
    END
FROM users
WHERE type = 'customer';

-- 为 riders 表插入数据
DELETE FROM riders;
INSERT INTO riders (id, vehicle, license, available)
SELECT
    id,
    CASE
        WHEN id % 5 = 0 THEN '电动车'
        WHEN id % 5 = 1 THEN '自行车'
        WHEN id % 5 = 2 THEN '摩托车'
        WHEN id % 5 = 3 THEN '汽车'
        ELSE '步行'
    END,
    'License ' || id,
    TRUE
FROM users
WHERE type = 'rider';

-- 为 restaurants 表插入数据
DELETE FROM restaurants;
INSERT INTO restaurants (id, name, address, phone, cuisine)
SELECT
    id,
    'Restaurant Name ' || id,
    'Restaurant Address ' || id,
    '123456789' || id,
    'Cuisine Type ' || (id % 10 + 1)
FROM users
WHERE type = 'restaurant';

-- 向 admins 表插入数据
INSERT INTO admins (id, name)
```

```
  SELECT
      id,
      'Admin Name ' || id
  FROM users
  WHERE type = 'admin';

  -- 为 menu_items 表插入数据
  INSERT INTO menu_items (restaurant_id, name, description, price, category)
  SELECT
      r.id,
      'Menu Item ' || s,
      'Description for Menu Item ' || s,
      round(random() * 50, 2),
      CASE
          WHEN s % 4 = 0 THEN 'Appetizer'
          WHEN s % 4 = 1 THEN 'Main Course'
          WHEN s % 4 = 2 THEN 'Dessert'
          ELSE 'Beverage'
      END
  FROM generate_series(1, 30) AS s
  JOIN restaurants r ON r.id = (SELECT id FROM restaurants ORDER BY random() LIMIT 1);

  -- 为 orders 表插入数据
  INSERT INTO orders (customer_id, restaurant_id, total_amount, status, created_at)
  SELECT
      (SELECT id FROM customers ORDER BY random() LIMIT 1),
      (SELECT id FROM restaurants ORDER BY random() LIMIT 1),
      round(random() * 100 + 20, 2),
      CASE
          WHEN random() < 0.2 THEN 'placed'
          WHEN random() < 0.4 THEN 'preparing'
          WHEN random() < 0.6 THEN 'delivering'
          WHEN random() < 0.8 THEN 'completed'
          ELSE 'cancelled'
      END,
      CURRENT_TIMESTAMP - INTERVAL '1' DAY * (random() * 30)::int
  FROM generate_series(1, 30) AS s;

  -- 为 deliveries 表插入数据
  INSERT INTO deliveries (order_id, rider_id, delivery_status, delivery_time,
menu_item_name, quantity, price)
  SELECT
      o.id,
      r.id,
      CASE
          WHEN random() < 0.33 THEN 'pending'
          WHEN random() < 0.66 THEN 'in_transit'
          ELSE 'delivered'
      END,
      CURRENT_TIMESTAMP - INTERVAL '1' HOUR * (random() * 24)::int,
      mi.name,
```

```
        (random() * 5 + 1)::int,
        round(random() * 50, 2)
    FROM orders o
    JOIN riders r ON r.id = (SELECT id FROM riders ORDER BY random() LIMIT 1)
    JOIN menu_items mi ON mi.id = (SELECT id FROM menu_items ORDER BY random() LIMIT 1)
    ORDER BY o.id
    LIMIT 30;

    -- 为 payments 表插入数据
    INSERT INTO payments (order_id, payment_method, payment_status, payment_time)
    SELECT
        o.id,
        CASE
            WHEN random() < 0.33 THEN '支付宝'
            WHEN random() < 0.66 THEN '微信'
            ELSE '银行卡'
        END,
        CASE
            WHEN random() < 0.8 THEN 'completed'
            ELSE 'pending'
        END,
        CURRENT_TIMESTAMP - INTERVAL '1' DAY * (random() * 30)::int
    FROM orders o
    ORDER BY o.id
    LIMIT 30;

    -- 检查特定用户数量
    PERFORM COUNT(*) AS UserCount FROM users WHERE username = 'user_1' AND password =
'password_1';
END;
$$;
```

触发器

该触发器用于实现在用户注册时，在user总表更新后同时检查user各分表(顾客表等)，检查数据的完整性

```
-- 创建触发器函数
CREATE OR REPLACE FUNCTION after_insert_user_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.type = 'customer' THEN
        INSERT INTO customers (id) VALUES (NEW.id);
    ELSIF NEW.type = 'rider' THEN
        INSERT INTO riders (id) VALUES (NEW.id);
    ELSIF NEW.type = 'restaurant' THEN
        INSERT INTO restaurants (id) VALUES (NEW.id);
    ELSIF NEW.type = 'admin' THEN
        INSERT INTO admins (id) VALUES (NEW.id);
    END IF;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- 创建触发器
CREATE TRIGGER after_insert_user
AFTER INSERT ON users
FOR EACH ROW
EXECUTE PROCEDURE after_insert_user_trigger();
DROP TRIGGER after_insert_user ON users;
```

采用**Delphi**至少**5**种组件

本项目采用多种组件，如图所示，仅一个页面就有9中不同组件。