票务管理系统项目介绍

1. 在 My SQL 8.0 中创建一个存储航班信息的数据库；

首先创建的数据库，名为 MIS_DB

```
CREATE DATABASE MIS_DB;
```

2. 在刚创建的数据库中，设计和创建实验所用到的数据库结构；

（1）首先创建 USER 表

```
CREATE TABLE `USER` (
  `u_id` varchar(4) NOT NULL,
  `u_name` varchar(20) DEFAULT NULL,
  `u_password` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`u_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（2）然后向 USER 表插入数据

```
INSERT INTO `user` VALUES ('0001', 'HHR_B19021418', '123456');
INSERT INTO `user` VALUES ('0002', 'XB_Teacher', '123456');
```

（3）再创建 Flight 表

```
CREATE TABLE `flight` (
  `f_id` char(8) NOT NULL,
  `f_src` varchar(15) DEFAULT NULL,
  `f_des` varchar(15) DEFAULT NULL,
  `f_date` date NOT NULL,
  `f_start_time` char(6) DEFAULT NULL,
  `f_end_time` char(6) DEFAULT NULL,
  `f_remain_seats` int(4) DEFAULT NULL,
  `f_fares` float(8,0) DEFAULT NULL,
  `f_discount_nums` float(8,0) DEFAULT NULL,
  `f_discount` float(8,0) DEFAULT NULL,
  `f_subordinate_company` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`f_id`,`f_date`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（4）然后向 Flight 表插入数据数据

INSERT INTO `flight` VALUES ('F001', '南京', '北京', '2022-05-12', '13:30', '15:20', '5', '999', '5', '1', '国航');

INSERT INTO `flight` VALUES ('F002', '南京', '上海', '2022-05-13', '12:20', '13:50', '20', '599', '5', '1', '国航');

INSERT INTO `flight` VALUES ('F003', '南京', '广州', '2022-05-13', '9:20', '11:50', '4', '899', '5', '1', '南航');

INSERT INTO `flight` VALUES ('F004', '南京', '深圳', '2022-05-13', '7:20', '9:50', '2', '799', '5', '1', '东航');

INSERT INTO `flight` VALUES ('F005', '南京', '北京', '2022-05-14', '12:20', '14:50', '40', '999', '5', '1', '川航');

实现过程如下：

```
mysql> USE MIS_DB
Database changed
mysql> SET FOREIGN_KEY_CHECKS=0;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> CREATE TABLE `USER` (
    ->   `u_id` varchar(4) NOT NULL,
    ->   `u_name` varchar(20) DEFAULT NULL,
    ->   `u_password` varchar(10) DEFAULT NULL,
    ->   PRIMARY KEY (`u_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (0.04 sec)

mysql> INSERT INTO `user` VALUES ('0001', 'HHR_B19021418', '123456');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `user` VALUES ('0002', 'XB_Teacher', '123456');
Query OK, 1 row affected (0.01 sec)

mysql> CREATE TABLE `flight` (
    ->   `f_id` char(8) NOT NULL,
    ->   `f_src` varchar(15) DEFAULT NULL,
    ->   `f_des` varchar(15) DEFAULT NULL,
    ->   `f_date` date NOT NULL,
    ->   `f_start_time` char(6) DEFAULT NULL,
    ->   `f_end_time` char(6) DEFAULT NULL,
    ->   `f_remain_seats` int(4) DEFAULT NULL,
    ->   `f_fares` float(8,0) DEFAULT NULL,
    ->   `f_discount_nums` float(8,0) DEFAULT NULL,
    ->   `f_discount` float(8,0) DEFAULT NULL,
    ->   `f_subordinate_company` varchar(20) DEFAULT NULL,
    ->   PRIMARY KEY (`f_id`,`f_date`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 5 warnings (0.03 sec)

mysql>
mysql> INSERT INTO `flight` VALUES ('F001', '南京', '北京', '2022-05-12', '13:30', '15:20', '5', '999', '5', '0.5', '国航');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `flight` VALUES ('F002', '南京', '上海', '2022-05-13', '12:20', '13:50', '20', '599', '5', '0.8', '国航');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `flight` VALUES ('F003', '南京', '广州', '2022-05-13', '9:20', '11:50', '4', '899', '5', '0.7', '南航');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `flight` VALUES ('F004', '南京', '深圳', '2022-05-13', '7:20', '9:50', '2', '799', '5', '1', '东航');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `flight` VALUES ('F005', '南京', '北京', '2022-05-14', '12:20', '14:50', '40', '999', '5', '1', '川航');
Query OK, 1 row affected (0.01 sec)
```

图 1 My SQL mis_db 数据库创建过程

3. UI 简介

在本项目中一共设计了四个页面，

(1) 开始菜单页面：menu.py

图 2 开始菜单 ui 及其控件功能介绍

开始菜单页面共包含两个主要控件，功能如图 2 所示。

(2) 登录页面：login.py

图 3 登录页面 ui 及其控件功能介绍

登录菜单页面共包含四个主要控件，功能如图 3 所示。

(3) 操作页面：operater.py

图 4 操作页面 ui 及其控件功能介绍

操作页面共包含五个主要控件，功能如图 4 所示。

(4) 增加数据对话框页面：add_dialog.py

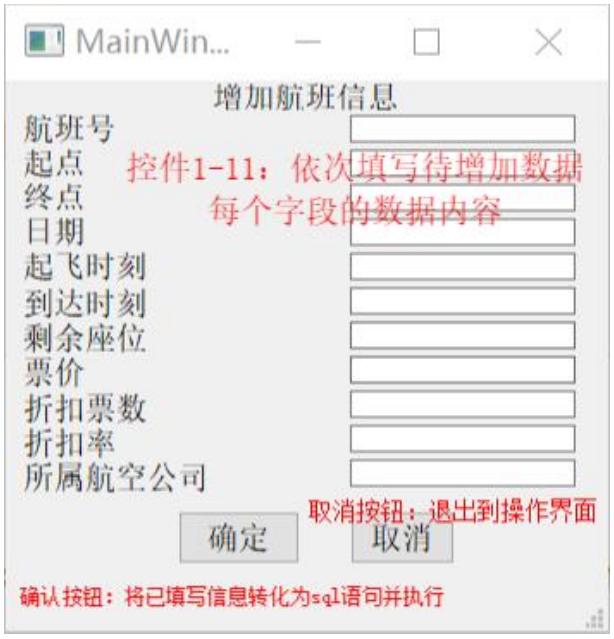

图 5 增加数据对话框页面 ui 及其控件功能介绍

增加数据对话框页面共包含十三个主要控件，功能如图 5 所示。

(5) 前端代码生成过程（以增加数据对话框页面为例）：

① 通过 Qt designer，进行界面设计，拖拽制作的 .ui 文件

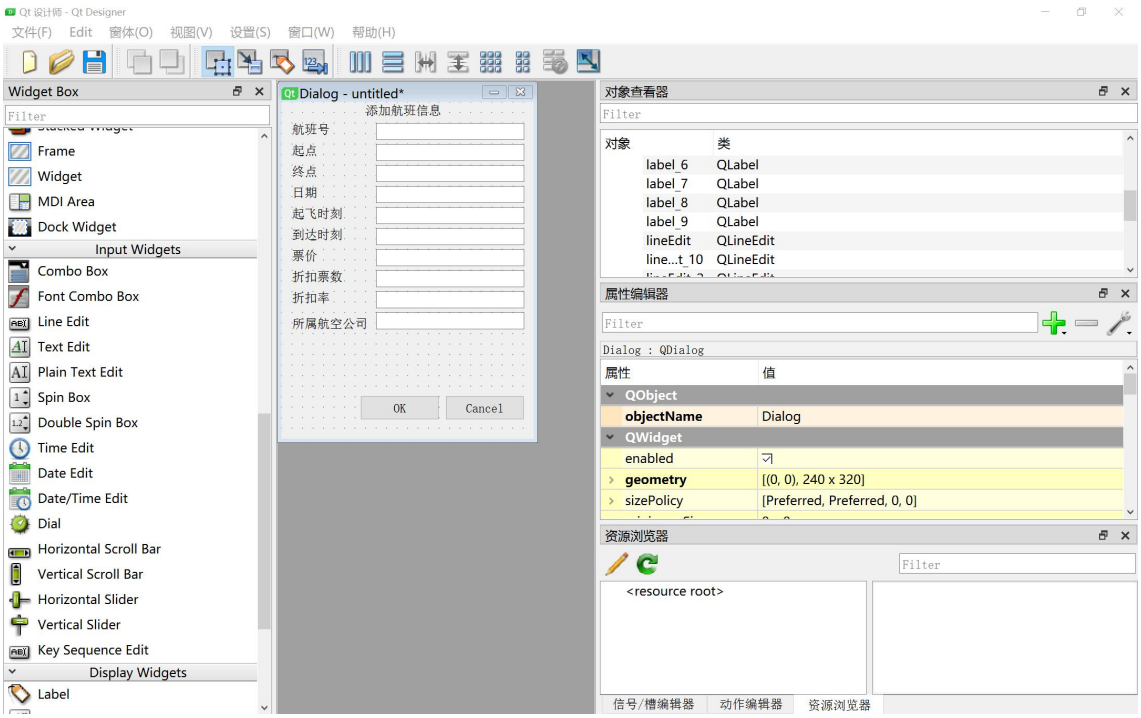

图 6 Qt designer 上对增加数据对话框设计

② **再通过 pyuic5 工具转为 .py 文件.**

图 7 通过 pyui 命令行指令转换 ui 文件为 py 文件

③ **将.py 文件加入 Pycharm 项目.**

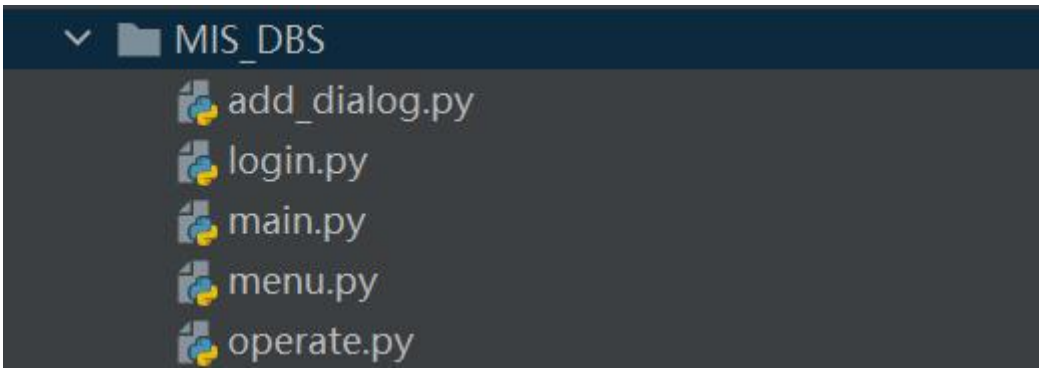

图 8 Pycharm 项目结构

图 8 中 login.py add_dialog.py menu.py operate.py 为所生成的 4 个 UI 界面前端代码。

4. UI 界面间跳转的实现：

本项目中我通过定义 Controller 类实现页面间跳转:

```
# 页面的跳转
class Controller:
    def __init__(self):
        self.menu = MenuWindow()#菜单窗口
        self.login = LoginWindow()#登录窗口
        self.operate = OperateWindow()#操作窗口
        self.add = AddWindow()#增加数据对话框窗口
    def show_menu(self):
        self.menu.switch_window1.connect(self.show_login) #跳转登陆界面
        self.menu.switch_window2.connect(self.show_operate) #跳转操作界面
        self.menu.show()
        self.login.close()
        self.operate.close()
    def show_login(self):
        self.login.switch_window1.connect(self.show_operate)#跳转操作界面
        self.login.switch_window2.connect(self.show_menu)#跳转菜单界面
        self.login.show()
        self.menu.close()
        self.operate.close()
    def show_operate(self):
        self.operate.switch_window1.connect(self.show_add)#跳转数据对话框界面
        self.menu.close()
        self.login.close()
        self.add.close()
        self.operate.show()
    def show_add(self):
        self.add.switch_window.connect(self.show_operate)#跳转操作界面
        self.operate.close()
        self.add.show()
```

以由开始菜单跳转到登录界面为例：

```
# 开始菜单窗口
class MenuWindow(QtWidgets.QMainWindow, Menu_Ui):
    switch_window1 = QtCore.pyqtSignal()
    switch_window2 = QtCore.pyqtSignal()
    def __init__(self):
        super(MenuWindow, self).__init__()
        self.setupUi(self)
        self.manageButton.clicked.connect(self.goLogin)
        self.queryButton.clicked.connect(self.goOperate)
    def goLogin(self):
        print("manageButton clicked!!")
        self.switch_window1.emit()
    def goOperate(self):
```

```
        print("queryButton clicked!!")
        self.switch_window2.emit()
```

在 MenuWindow（）类中定义跳转界面信号：switch_window1 = QtCore.pyqtSignal()，当票务管理按钮（manageButton）被点击时，跳转界面信号被激发。由于在 Controller 类中该信号被连接到了 show_login ( )函数，则其被激发时将跳转到登录界面。

5. 实现各功能的响应事件、事件的核心代码，如插入、更新或删除事件等

（1）连接到数据库（刚刚创建的 mis_db）

```
def  get_connection():
    conn = pymysql.connect(
        host='127.0.0.1',  # 连接主机，默认 127.0.0.1
        user='root',  # 用户名
        passwd='h20011126',  # 密码
        port=3306,  # 端口，默认为 3306
        db='mis_db',  # 数据库名称
        charset='utf8'  # 字符编码
    )
    return conn
```

（2）登录

图 9 登录功能过程展示

图 9 展示了登录过程，登录功能核心代码如下：

```
    def ok(self):
        username = self.userText.text()
```

```
        password = self.pwdText.text()
        # 创建数据库连接
        conn = get_connection()
        # 生成游标对象 cursor
        cursor = conn.cursor()
        if (cursor.execute("SELECT * FROM user WHERE u_name='%s' AND
u_password='%s'" % (username, password))):
            QMessageBox.information(self, username,"登陆成功",QMessageBox.Yes)
            cursor.close()
            conn.close()
            print("switch to operate")
            self.switch_window1.emit()
        else:
            QMessageBox.warning(self, "警告", "密码错误", QMessageBox.Cancel)
            cursor.close()
            conn.close()
```

（3）查询

图 10 查询功能过程展示

图 10 展示了查询过程，查询功能核心代码如下：

```
def query(self): #查询
        print("querying")
        self.tableWidget.clearContents() # 清空现有表格
        # 数据库连接对象
        conn = get_connection()
        # 游标对象
```

```
        cur = conn.cursor()
        # 查询的 sql 语句
        sql = "SELECT * FROM flight"
        cur.execute(sql)
        # 获取查询到的数据，是以二维元组的形式存储的，所以读取需要使用 data[i][j] 下
标定位
        data = cur.fetchall()
        # 遍历二维元组，将 id 和 name 显示到界面表格上
        x = 0
        for i in data:
            y = 0
            for j in i:
                self.tableWidget.setItem(x, y, QtWidgets.QTableWidgetItem(str(data[x][y])))
                y = y + 1
            x = x + 1
        cur.close()
        conn.close()
        print("querying done")
```

（4）增加

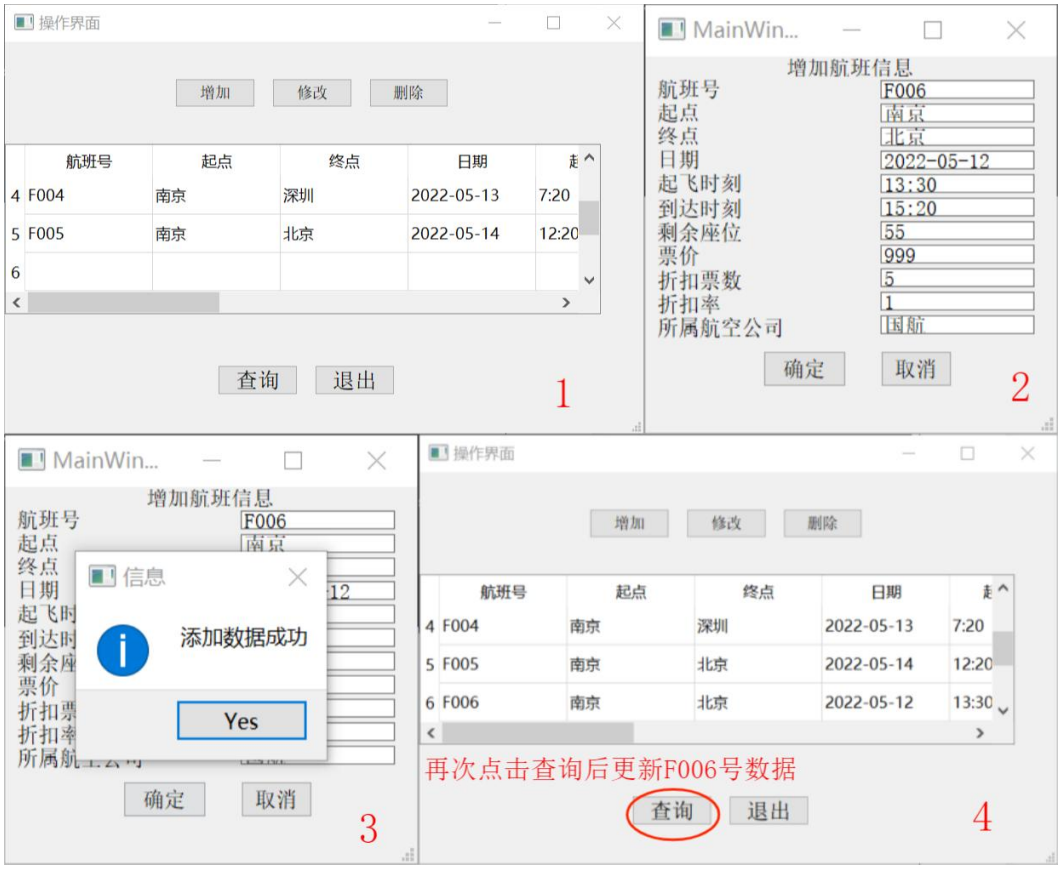

图 11 增加数据功能过程展示

图 11 展示了增加数据过程，增加数据功能核心代码如下：

```
    def add(self):
```

```
        v1 = self.lineEdit_1.text()
        v2 = self.lineEdit_2.text()
        v3 = self.lineEdit_3.text()
        v4 = self.lineEdit_4.text()
        v5 = self.lineEdit_5.text()
        v6 = self.lineEdit_6.text()
        v7 = self.lineEdit_7.text()
        v8 = self.lineEdit_8.text()
        v9 = self.lineEdit_9.text()
        v10 = self.lineEdit_10.text()
        v11 = self.lineEdit_11.text()
        values=(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11)
        conn = get_connection()
        #  游标对象
        cur = conn.cursor()
        #  增加的 sql 语句
        sql = "insert into " \
              "flight(f_id,f_src,f_des,f_date,f_start_time,f_end_time," \
             "f_remain_seats,f_fares,f_discount_nums,f_discount,f_subordinate_company) " \
              "values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        if(cur.execute(sql,values)):
            conn.commit()
            print("success")
            QMessageBox.information(self, "信息", "添加数据成功", QMessageBox.Yes)
            cur.close()
            conn.close()
            self.switch_window.emit()
```

（5）更新

图 12 更新数据功能过程展示

图 12 展示了更新数据过程，更新数据功能核心代码如下：

```
def goCha(self):
    print("change BUTTON CLICKED")
    conn = get_connection()
    print("connection success")
    # 游标对象
    fid, ok = QInputDialog.getText(None, "修改航班信息", "请输入待修改航班的航班号",)
    field, ok = QInputDialog.getText(None, "修改航班信息", "请输入待修改字段",)
    value, ok = QInputDialog.getText(None, "修改航班信息", "请输入修改值", )
    cur = conn.cursor()
    # 其他字段的修改类似实现，此处略
    if field == "起点":
        sql = "UPDATE flight SET f_src = '%s' WHERE f_id = '%s'" % (value,fid)
        if (cur.execute(sql)):
```

```
            conn.commit()
            print("success")
            QMessageBox.information(self, "信息", "修改数据成功", QMessageBox.Yes)
            cur.close()
            conn.close()
```

（6）删除

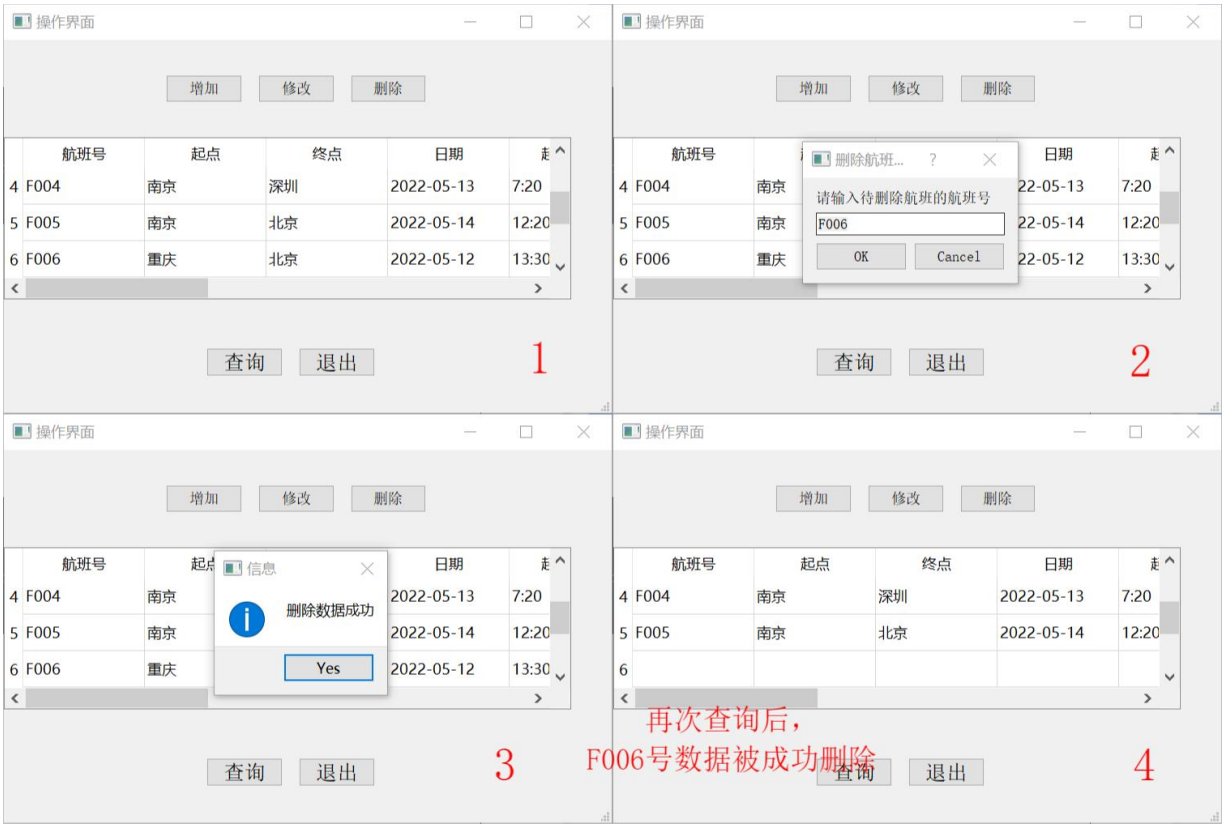

图 13 删除数据功能过程展示

图 13 展示了删除数据过程，删除数据功能核心代码如下：

```
def goDel(self):
    fid,ok = QInputDialog.getText(None, "删除航班信息", "请输入待删除航班的航班号",)
    conn = get_connection()
    # 游标对象
    cur = conn.cursor()
    # 查询的 sql 语句
    sql = "delete from flight where f_id='%s'" % (fid,)
    if(cur.execute(sql)):
        conn.commit()
        print("success")
        QMessageBox.information(self, "信息", "删除数据成功", QMessageBox.Yes)
        cur.close()
        conn.close()
```