

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА—Российский технологический университет»**

**РТУ МИРЭА**

**Институт комплексной безопасности и специального приборостроения**

(наименование института, филиала)

Кафедра КБ-2 «Прикладные информационные технологии»

(наименование кафедры)

**Курсовая работа**

по дисциплине \_\_\_\_\_\_\_\_\_ Клиент-серверные базы данных \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

**Тема курсовой работы** Создание клиент-серверного приложения для работы с базой данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Студент группы**  БСБО 03-18 Стаценко В.Е.

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

**Руководитель курсовой работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(должность, звание, ученая степень) (подпись руководителя)

**Рецензент** (при наличии)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, звание, ученая степень) (подпись рецензента)

Работа представлена к защите «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Допущен к защите «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА—Российский технологический университет»**

**РТУ МИРЭА**

**Институт комплексной безопасности и специального приборостроения**

(наименование института, филиала)

Кафедра КБ-2 «Прикладные информационные технологии»

(наименование кафедры)

Утверждаю

Заведующий кафедрой:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_\_Клиент-серверные базы данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

**Тема курсовой работы** Создание клиент-серверного приложения для работы с базой данных **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Студент группы**  БСБО-03-18 Стаценко В.Е.  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

**Исходные данные**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Перечень вопросов, подлежащих обработке, и обязательного графического материала:**

Создание клиента для работы с БД\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Реализация прав доступа в клиентской части\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Создание БД, таблиц, функций, процедур, триггеров индексов, транзакций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Реализация в клиенте запросов согласно заданию\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Добавление индексов для ускорения запросов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Срок предоставление к защите курсовой работы до** «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Задание на курсовую работу выдал \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Ф.И.О. руководителя) (подпись руководителя)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Задание на курсовую работу получил** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О. исполнителя) (подпись исполнителя)

**Задание на курсовую работу по дисциплине «Базы данных»**

Разработать клиент-серверное приложение, серверная часть которой реализована на Microsoft SQL Server или PostgreSQL, представляющая собой модель предметной области в соответствии с вариантом задания. В рамках заданной предметной области реализовать заданную (по варианту) схему отношений, т.е. выделить сущности и их атрибуты, так чтобы связи между сущностями соответствовали представленной схеме. В рамках курсовой работы необходимо на стороне сервера реализовать и использовать при демонстрации приложения следующие компоненты:

1. Постоянные таблицы и связи между ними, количество таблиц и наличие связей должно соответствовать заданию, допускается увеличение числа таблиц и их полей для более адекватного представления предметной области;
2. В приложении реализовать не менее пяти запросов, включая (для демонстрации навыков работы)
   1. Составной многотабличный запрос с параметром, включающий соединение таблиц и CASE-выражение;
   2. На основе обновляющего представления (многотабличного VIEW), в котором критерий упорядоченности задает пользователь при выполнении;
   3. Запрос, содержащий коррелированные и некоррелированные подзапросы в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);
   4. Многотабличный запрос, содержащий группировку записей, агрегативные функции и параметр, используемый в разделе HAVING;
   5. Запрос, содержащий предикат ANY(SOME) или ALL;
3. Создать индексы для увеличения скорости выполнения запросов;
4. В таблице (в соответствии с вариантом) предусмотреть поле, которое заполняется автоматически по срабатыванию триггера при добавлении, обновлении и удалении данных, иметь возможность продемонстрировать работу триггера при работе приложения. Триггеры должны обрабатывать только те записи, которые были добавлены, изменены или удалены в ходе текущей операции (транзакции).
5. Операции добавления, удаления и обновления реализовать в виде хранимых процедур (функций) с параметрами для всех таблиц;
6. Реализовать отдельную хранимую процедуру(функцию), состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана;
7. В триггере или хранимой процедуре реализовать курсор на обновления отдельных данных;
8. В запросе (из пункта 2 или в дополнительном к тому перечню) использовать собственную скалярную функцию, а в хранимой процедуре – векторную (или табличную) функцию. Функции сохранить в базе данных
9. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

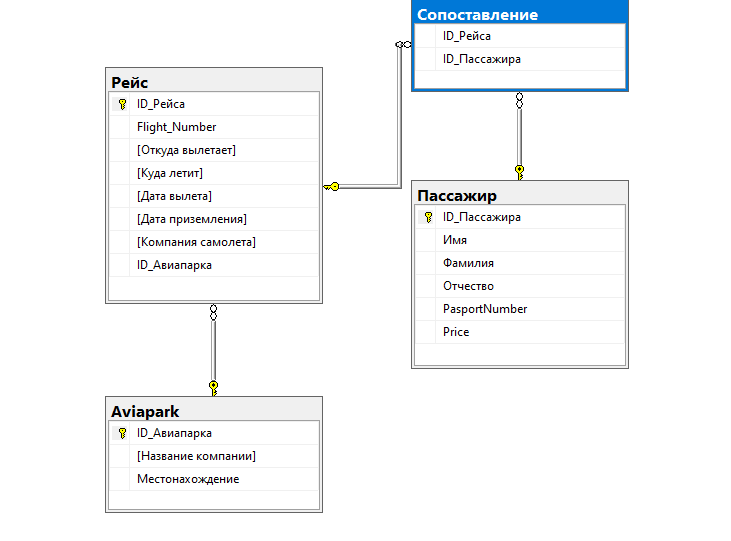
Клиент должен обеспечивать добавление, модификацию и удаление данных по всей предметной области. Добавление, редактирование данных в таблице производить в отдельном окне.

Запрещено в качестве вводимых данных, в том числе для связи таблиц, указывать значения первичных и внешних ключей – для обеспечения ссылочной целостности пользователь должен выбирать значения из справочника, а соответствующие значения должны подставляться программно (тем или иным способом – автоматически).

*Клиент должен быть реализован на средствах (языках программирования) изученных (изучаемых) в рамках учебной программы*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 27 | Организация движения самолетов в аэропорту | Ведение базы расписаний вылетов, прилетов, пассажиропотока рейсов, используемый авиапарк | 7 | 3 |

Задание 1



CREATE TABLE Aviapark(

[ID\_Авиапарка] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

[Название компании] [nvarchar](50) NOT NULL,

[Местонахождение] nvarchar(50) not null

)

CREATE TABLE Рейс(

ID\_Рейса int PRIMARY KEY IDENTITY(1,1) NOT NULL,

Flight\_Number int not null,

[Откуда вылетает] nvarchar(50) not null,

[Куда летит] nvarchar(50) not null,

[Дата вылета] date not null,

[Дата приземления] date not null,

[Компания самолета] [nvarchar](50) NOT NULL,

[ID\_Авиапарка] [int] NOT NULL,

FOREIGN KEY (ID\_Авиапарка) REFERENCES dbo.Aviapark(ID\_Авиапарка)

)

CREATE TABLE Сопоставление(

ID\_Рейса int not null,

ID\_Пассажира int not null,

FOREIGN KEY (ID\_Рейса) REFERENCES dbo.Рейс(ID\_Рейса),

FOREIGN KEY (ID\_Пассажира) REFERENCES dbo.Пассажир(ID\_Пассажира)

)

CREATE TABLE Пассажир(

[ID\_Пассажира] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

[Имя] [nvarchar](50) NOT NULL,

[Фамилия] [nvarchar](50) NOT NULL,

[Отчество] [nvarchar](50) NULL,

PasportNumber int

)

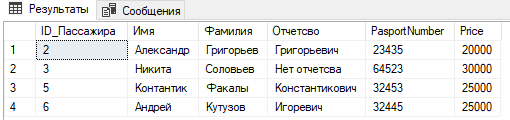
ALTER TABLE Пассажир ADD Price int

ALTER TABLE Пассажир alter column Price int not null

Задание2

a) use root;

SELECT ID\_Пассажира, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчетсва' ELSE [Отчество] END [Отчетсво], PasportNumber, Price FROM Пассажир

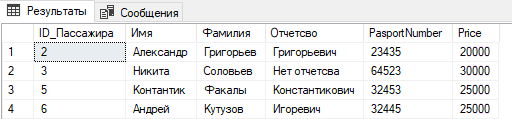


b)

use root;

CREATE VIEW Passenger AS SELECT ID\_Пассажира, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчетсва' ELSE [Отчество] END [Отчетсво], PasportNumber FROM Пассажир

SELECT \* FROM Passenger



c)

Запрос с коррелированным подзапросом в разделе WHERE

SELECT ID\_Пассажира, Фамилия FROM Пассажир AS t1 WHERE Price > (SELECT AVG(Price) FROM Пассажир AS

t2 WHERE t1.ID\_Пассажира = t2.ID\_Пассажира);

Запрос с некоррелированным подзапросом в разделе WHERE

SELECT ID\_Рейса FROM Рейс WHERE Flight\_Number < (SELECT Flight\_Number FROM Рейс WHERE ID\_Авиапарка = (SELECT ID\_Авиапарка FROM Aviapark WHERE [Название компании] = 'S7'))

Запрос c коррелированным и некоррелированным подзапросом в разделе

FROM

SELECT \* FROM (SELECT ID\_Пассажира FROM Пассажир WHERE Price >= 20000) AS Pas JOIN Пассажир ON Pas.ID\_Пассажира =

Пассажир.ID\_Пассажира;

SELECT \* FROM (SELECT Фамилия, Price AS Pr1 FROM Пассажир) AS Pr2 WHERE Pr1 > 15000;

Запрос с коррелированным и некоррелированным подзапросом в разделе

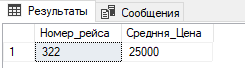
SELECT

SELECT Av.\*, (SELECT Flight\_Number FROM Рейс AS R WHERE R.ID\_Авиапарка = Av.ID\_Авиапарка) AS 'Номер рейса' FROM Aviapark AS

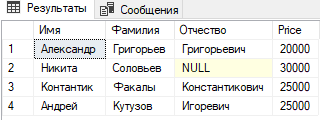
Av;

SELECT R.Flight\_Number, (SELECT Count(\*) FROM Aviapark WHERE ID\_Авиапарка = R.ID\_Авиапарка) AS 'Количество' FROM Рейс AS R;

e) SELECT Рейс.Flight\_Number as Номер\_рейса, avg(Пассажир.Price) as Средння\_Цена FROM Пассажир JOIN Сопоставление ON Пассажир.ID\_Пассажира = Сопоставление.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса GROUP BY Flight\_Number HAVING AVG(Price) > 20000



d) SELECT Имя, Фамилия, Отчество, Price FROM Пассажир WHERE ID\_Пассажира = ALL(SELECT ID\_Пассажира FROM Сопоставление WHERE ID\_Рейса = ALL(SELECT ID\_Рейса FROM Рейс))



Задание3

create index pass on Пассажир(ID\_Пассажира, Фамилия)

create index Flight on Рейс(ID\_Рейса, FlightNumber)

create index Avia on Aviapark(ID\_Авиапарка)

create index rav on Сопоставление(ID\_Рейса, ID\_Пассажира)

Задание4

REATE TRIGGER delete\_passenger ON Пассажир INSTEAD OF DELETE AS

BEGIN

DELETE FROM Сопоставление WHERE ID\_Пассажира=(

SELECT ID\_Пассажира FROM deleted)

DELETE FROM Пассажир WHERE ID\_Пассажира = (

SELECT ID\_Пассажира FROM deleted)

END;

CREATE TRIGGER delete\_flight ON Рейс INSTEAD OF DELETE AS

BEGIN

DELETE FROM Сопоставление WHERE ID\_Рейса = (

SELECT ID\_Рейса FROM deleted)

DELETE FROM Рейс WHERE ID\_Рейса = (

SELECT ID\_Рейса FROM deleted)

END;

Задание5

CREATE PROCEDURE Select\_Flight (@FlightNumber int)

AS

SELECT \* FROM Рейс WHERE Flight\_Number = @FlightNumber

CREATE PROCEDURE Select\_Passenger (@Last\_Name nvarchar(50))

AS

SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Фамилия = @Last\_Name;

CREATE PROCEDURE Insert\_Aviapark (@Name nvarchar(50), @Place [nvarchar](50))

AS

BEGIN

INSERT INTO Aviapark ([Название компании], [Местонахождение]) VALUES (@Name, @Place) END;

CREATE PROCEDURE Delete\_Aviapark (@id int)

AS

BEGIN

DELETE FROM Aviapark WHERE ID\_Авиапарка = @id END;

CREATE PROCEDURE Update\_Aviapark (@id int, @Name nvarchar(50), @Place nvarchar(50))

AS

BEGIN

UPDATE Aviapark SET [Название компании] = @Name, [Местонахождение] = @Place WHERE ID\_Авиапарка = @id END;

Задание 6

CREATE PROCEDURE Trans(@id int, @Name INT, @Place nvarchar(50))

AS

BEGIN

BEGIN TRANSACTION

INSERT INTO Aviapark([Название компании], [Местонахождение] VALUES (@Name, @Place);

IF @Name = @Place

ROLLBACK

COMMIT TRANSACTION

END

Задание 7

CREATE PROCEDURE Cur\_Update\_Aviapark(@update\_id INT, @update\_name nvarchar(50), @update\_place nvarchar (50)) AS

BEGIN

declare cur CURSOR

FOR SELECT ID\_Авиапарка FROM Aviapark;

OPEN cur

DECLARE @id int;

FETCH NEXT FROM cur INTO @id;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @update\_id = @id

UPDATE Aviapark

SET [Название компании] = '', Местонахождение = ''

WHERE ID\_Авиапарка = @update\_id

FETCH NEXT FROM cur INTO @id

END

CLOSE cur

DEALLOCATE cur;

END;

Задание 8(1)

CREATE FUNCTION Select\_from\_Fligt(@FlightNumber int)

RETURNS

TABLE

AS

RETURN

SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = @FlightNumber;

Задание 8(2)

CREATE PROCEDURE Select\_from\_Flight(@Last\_Name1 nvarchar (50))

AS

BEGIN

EXEC Select\_Passenger @Last\_Name1

END

Задание9

CREATE ROLE test;

GRANT SELECT TO test;

CREATE LOGIN tester WITH PASSWORD = '12345';

CREATE USER tester FOR LOGIN tester

EXEC sp\_addrolemember 'test','tester';

CREATE ROLE root;

GRANT SELECT, INSERT, DELETE, EXECUTE to root;

CREATE LOGIN root WITH PASSWORD = 'admin'

CREATE USER root FOR LOGIN root

EXEC sp\_addrolemember 'root','root';

**Пример кода клиента**

Для таблицы Авиапарк

self.ui.Show\_Aviapark.setRowCount(len(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()))  
for i in range (len(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall())):  
 self.ui.Show\_Aviapark.setItem(i, 0, QTableWidgetItem(str(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()[i][0])))  
 self.ui.Show\_Aviapark.setItem(i, 1, QTableWidgetItem(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()[i][1]))

Для таблицы пассажиров

LastName\_Passenger = str(self.ui.get\_ID\_Passenger.text())  
self.ui.Show\_Passenger.setColumnCount(len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()))  
for i in range(len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?)", (LastName\_Passenger)).fetchall())):  
 self.ui.Show\_Passenger.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][0])))  
 self.ui.Show\_Passenger.setItem(1, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][1]))  
 self.ui.Show\_Passenger.setItem(2, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][2]))  
 self.ui.Show\_Passenger.setItem(3, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][3]))  
 self.ui.Show\_Passenger.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][4])))  
 self.ui.Show\_Passenger.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][5])))

Для таблицы рейсов

Flight\_Number = int(self.ui.get\_ID\_Flight.text())  
Flight = self.cursor.execute("SELECT \* FROM [Рейс] WHERE Flight\_Number = (?);", (Flight\_Number)).fetchall()  
self.ui.Show\_Flight.setItem(0, 0, QTableWidgetItem(str(Flight[0][2])))  
self.ui.Show\_Flight.setItem(1, 0, QTableWidgetItem(str(Flight[0][3])))  
self.ui.Show\_Flight.setItem(2, 0, QTableWidgetItem(str(Flight[0][4])))  
self.ui.Show\_Flight.setItem(3, 0, QTableWidgetItem(str(Flight[0][5])))  
self.ui.Show\_Flight.setItem(4, 0, QTableWidgetItem(str(Flight[0][6])))  
self.ui.Show\_Flight.setItem(5, 0, QTableWidgetItem(str(Flight[0][7])))

Для вывода пассажиров на рейсе

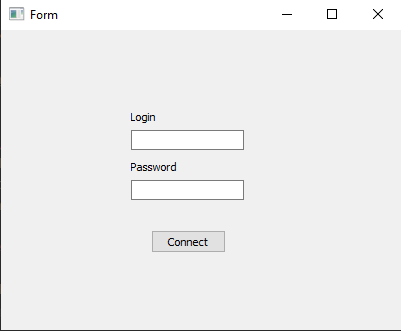
set\_Passanger = str(self.ui.Show\_Passenger\_Edit.text())  
self.ui.Show\_Passenger\_Board.setColumnCount(len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);", (set\_Passanger)).fetchall()))  
for i in range (len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall())):  
 self.ui.Show\_Passenger\_Board.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall()[i][0])))  
 self.ui.Show\_Passenger\_Board.setItem(1, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall()[i][1])))  
 self.ui.Show\_Passenger\_Board.setItem(2, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall()[i][2])))  
 self.ui.Show\_Passenger\_Board.setItem(3, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);", (set\_Passanger)).fetchall()[i][3])))  
 self.ui.Show\_Passenger\_Board.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);", (set\_Passanger)).fetchall()[i][4])))  
 self.ui.Show\_Passenger\_Board.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);" ,(set\_Passanger)).fetchall()[i][5])))

**Код программы**

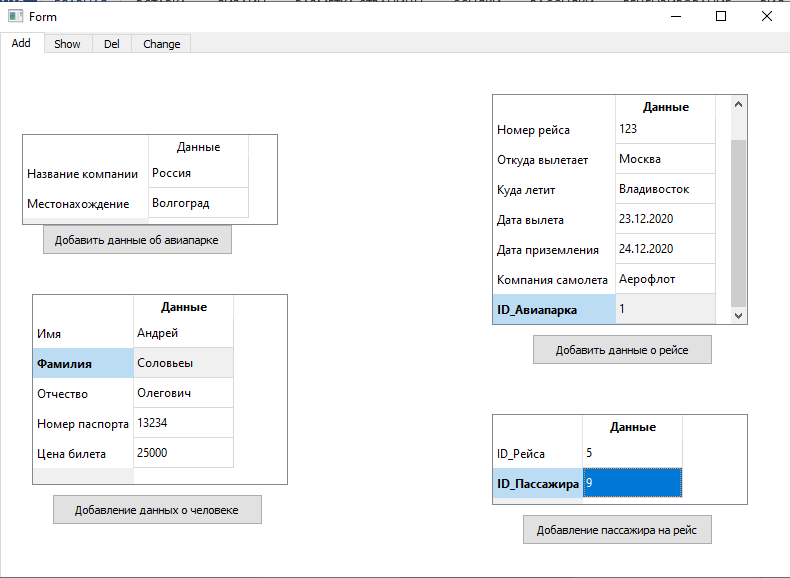
from PyQt5 import QtCore  
from PyQt5.QtWidgets import \*  
import sys  
from Main import Ui\_Form  
import pyodbc  
from admin import Ui\_Administration  
  
class Administration(QMainWindow, Ui\_Administration):  
  
  
 def \_\_init\_\_(self):  
 super(Administration, self).\_\_init\_\_()  
 self.setupUi(self)  
 self.pushButton.clicked.connect(self.login)  
  
 def login(self):  
 login = str(self.lineEdit.text())  
 password = str(self.lineEdit\_2.text())  
 try:  
 parameters = "DRIVER=ODBC Driver 17 for SQL Server;\  
 Server=DESKTOP-8NRUDO3;\  
 Database=Kursach;\  
 UID={};\  
 PWD={};\  
 Trusted\_connection=no;".format(login, password)  
 connection = pyodbc.connect(parameters)  
  
 cursor = connection.cursor()  
  
 self.window = Ui(cursor)  
 self.window.show()  
 self.close()  
 except:  
 QMessageBox(QMessageBox.Critical, "Пашол нахуй", "Пашол нахуй").exec()  
  
  
  
  
  
  
  
  
class Ui(QMainWindow, Ui\_Form):  
  
 editItems = False  
 currentValue = 0  
 cursor = 0  
  
 def \_\_init\_\_(self, cursor):  
 super(Ui, self).\_\_init\_\_()  
 self.cursor = cursor  
 self.ui = Ui\_Form()  
 self.ui.setupUi(self)  
  
 # if self.connection:  
 # QMessageBox(QMessageBox.Information, "Информация", 'Подключенно к БД!').exec()  
 self.errorbox = QMessageBox()  
 self.errorbox.setWindowTitle("Critical error")  
 self.errorbox.setText("Данные не были введены!")  
 self.errorbox.setIcon(QMessageBox.Critical)  
 self.complete = QMessageBox()  
 self.complete.setWindowTitle("Information")  
 self.complete.setText("Данные успешно введены!")  
 self.complete.setIcon(QMessageBox.Information)  
 self.ui.AddAviapark.clicked.connect(self.get\_Aviapark)  
 self.ui.AddFlight.clicked.connect(self.get\_Flight)  
 self.ui.Add\_Passenger\_Button.clicked.connect(self.Add\_Passenger)  
 self.ui.Add\_Passenger\_on\_board\_Button.clicked.connect(self.Passenger\_on\_board)  
 self.ui.Show\_Aviapark\_Button.clicked.connect(self.set\_Aviapark)  
 self.ui.Show\_Passenger\_Button.clicked.connect(self.set\_Passenger)  
 self.ui.Show\_Flight\_Button.clicked.connect(self.set\_Flight)  
 self.ui.Show\_Passenger\_Board\_Button.clicked.connect(self.set\_Passenger\_on\_Board)  
 self.ui.Show\_Del\_Passenger.clicked.connect(self.Show\_to\_Delete\_Passenger)  
 self.ui.Delete\_Passenger\_Button.clicked.connect(self.Delete\_Passenger)  
 self.ui.Show\_Del\_Flight.clicked.connect(self.Show\_to\_Delete\_Flight)  
 self.ui.Delete\_Flight\_Button.clicked.connect(self.Delete\_Flight)  
 self.ui.Switch\_Button.clicked.connect(self.Print\_Table)  
 self.ui.Change\_Table.itemDoubleClicked.connect(self.Changed)  
  
 # /-------------------------------------------------------------------------------------------------/  
 def get\_Aviapark(self): #Добавление данных  
 column\_conteiner=[]  
 try:  
 for x in range(0, self.ui.AviaparkAdd.rowCount()):  
 (column\_conteiner.append(self.ui.AviaparkAdd.item(x,0).text()))  
  
 self.cursor.execute("INSERT INTO Aviapark ([Название компании], [Местонахождение]) VALUES (?,?);", (column\_conteiner[0], column\_conteiner[1]))  
 self.complete.exec()  
 self.ui.AviaparkAdd.setItem(1, 0, QTableWidgetItem(""))  
 self.ui.AviaparkAdd.setItem(0, 0, QTableWidgetItem(""))  
 self.cursor.commit()  
 except:  
 self.errorbox.exec()  
  
 #/-------------------------------------------------------------------------------------------------/  
 def get\_Flight(self): #Добавление данных  
 column\_conteiner = []  
 try:  
 for x in range(0,self.ui.FlightAdd.rowCount()):  
 column\_conteiner.append(self.ui.FlightAdd.item(x,0).text())  
 self.cursor.execute(  
 "INSERT INTO Рейс (Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка]) VALUES (?, ?, ?, ?, ?, ?, ?);",  
 (column\_conteiner[0], column\_conteiner[1], column\_conteiner[2], column\_conteiner[3],  
 column\_conteiner[4], column\_conteiner[5], column\_conteiner[6]))  
 self.complete.exec()  
 for i in range (self.ui.FlightAdd.rowCount()):  
 self.ui.FlightAdd.setItem(i, 0, QTableWidgetItem(""))  
 self.cursor.commit()  
  
 except:  
 self.errorbox.exec()  
  
 # /-------------------------------------------------------------------------------------------------/  
 def Add\_Passenger(self):  
 column\_conteiner = []  
  
 try:  
 for x in range(0, self.ui.AddPassenger.rowCount()):  
 column\_conteiner.append(self.ui.AddPassenger.item(x, 0).text())  
 self.cursor.execute("INSERT INTO Пассажир ([Имя], [Фамилия], [Отчество], PasportNumber, Price) VALUES (?, ?, ?, ?, ?);",(column\_conteiner[0], column\_conteiner[1], column\_conteiner[2], column\_conteiner[3], column\_conteiner[4]))  
 self.complete.exec()  
 for i in range (self.ui.AddPassenger.rowCount()):  
 self.ui.AddPassenger.setItem(i, 0, QTableWidgetItem(""))  
 self.cursor.commit()  
 except:  
 self.errorbox.exec()  
  
  
 def Passenger\_on\_board(self):  
 try:  
  
 self.cursor.execute("INSERT INTO Сопоставление (ID\_Рейса, ID\_Пассажира) VALUES (?,?)",(self.ui.Add\_on\_Flight.item(0, 0).text(), self.ui.Add\_on\_Flight.item(1, 0).text()))  
 self.complete.exec()  
 for i in range(self.ui.Add\_on\_Flight.rowCount()):  
 self.ui.Add\_on\_Flight.setItem(i, 0, QTableWidgetItem(""))  
 self.cursor.commit()  
 except:  
 self.errorbox.exec()  
  
 def set\_Aviapark(self):  
 self.ui.Show\_Aviapark.setRowCount(len(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()))  
 for i in range (len(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall())):  
 self.ui.Show\_Aviapark.setItem(i, 0, QTableWidgetItem(str(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()[i][0])))  
 self.ui.Show\_Aviapark.setItem(i, 1, QTableWidgetItem(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()[i][1]))  
  
  
  
 def set\_Passenger(self):  
 LastName\_Passenger = str(self.ui.get\_ID\_Passenger.text())  
 self.ui.Show\_Passenger.setColumnCount(len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()))  
 for i in range(len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?)", (LastName\_Passenger)).fetchall())):  
 self.ui.Show\_Passenger.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][0])))  
 self.ui.Show\_Passenger.setItem(1, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][1]))  
 self.ui.Show\_Passenger.setItem(2, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][2]))  
 self.ui.Show\_Passenger.setItem(3, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][3]))  
 self.ui.Show\_Passenger.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][4])))  
 self.ui.Show\_Passenger.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет фамилии' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE [Фамилия] = (?);", (LastName\_Passenger)).fetchall()[i][5])))  
  
 def set\_Flight(self):  
 Flight\_Number = int(self.ui.get\_ID\_Flight.text())  
 Flight = self.cursor.execute("SELECT \* FROM [Рейс] WHERE Flight\_Number = (?);", (Flight\_Number)).fetchall()  
 self.ui.Show\_Flight.setItem(0, 0, QTableWidgetItem(str(Flight[0][2])))  
 self.ui.Show\_Flight.setItem(1, 0, QTableWidgetItem(str(Flight[0][3])))  
 self.ui.Show\_Flight.setItem(2, 0, QTableWidgetItem(str(Flight[0][4])))  
 self.ui.Show\_Flight.setItem(3, 0, QTableWidgetItem(str(Flight[0][5])))  
 self.ui.Show\_Flight.setItem(4, 0, QTableWidgetItem(str(Flight[0][6])))  
 self.ui.Show\_Flight.setItem(5, 0, QTableWidgetItem(str(Flight[0][7])))  
  
  
 def set\_Passenger\_on\_Board(self):  
 set\_Passanger = str(self.ui.Show\_Passenger\_Edit.text())  
 self.ui.Show\_Passenger\_Board.setColumnCount(len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);", (set\_Passanger)).fetchall()))  
 for i in range (len(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall())):  
 self.ui.Show\_Passenger\_Board.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall()[i][0])))  
 self.ui.Show\_Passenger\_Board.setItem(1, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall()[i][1])))  
 self.ui.Show\_Passenger\_Board.setItem(2, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?)", (set\_Passanger)).fetchall()[i][2])))  
 self.ui.Show\_Passenger\_Board.setItem(3, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);", (set\_Passanger)).fetchall()[i][3])))  
 self.ui.Show\_Passenger\_Board.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);", (set\_Passanger)).fetchall()[i][4])))  
 self.ui.Show\_Passenger\_Board.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number AS Номер\_рейса, [Имя], [Фамилия], CASE WHEN [Отчество] IS NULL THEN 'Нет отчества' ELSE Отчество END [Отчество], PasportNumber, Price FROM Пассажир JOIN Сопоставление ON Сопоставление.ID\_Пассажира = Пассажир.ID\_Пассажира JOIN Рейс ON Рейс.ID\_Рейса = Сопоставление.ID\_Рейса WHERE Flight\_Number = (?);" ,(set\_Passanger)).fetchall()[i][5])))  
  
 def Show\_to\_Delete\_Passenger(self):  
 self.ui.Delete\_Passenger.setColumnCount(len(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()))  
 for i in range (len(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall())):  
 self.ui.Delete\_Passenger.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()[i][0])))  
 self.ui.Delete\_Passenger.setItem(1, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()[i][1])))  
 self.ui.Delete\_Passenger.setItem(2, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()[i][2])))  
 self.ui.Delete\_Passenger.setItem(3, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()[i][3])))  
 self.ui.Delete\_Passenger.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()[i][4])))  
 self.ui.Delete\_Passenger.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall()[i][5])))  
 for i in range (5):  
 for j in range (len(self.cursor.execute("SELECT \* FROM Passenger ORDER BY Price").fetchall())):  
 self.ui.Delete\_Passenger.item(i, j).setFlags(QtCore.Qt.ItemIsEnabled)  
  
 def Delete\_Passenger(self):  
 Del\_Passenger = self.ui.Delete\_ID\_Passenger.text()  
 self.cursor.execute("DELETE FROM Сопоставление WHERE [ID\_Пассажира] = (?);", (Del\_Passenger))  
 self.cursor.execute("DELETE FROM Пассажир WHERE [ID\_Пассажира] = (?);", (Del\_Passenger))  
 self.cursor.commit()  
  
 def Show\_to\_Delete\_Flight(self):  
 self.ui.Delete\_Flight.setColumnCount(len(self.cursor.execute("SELECT \* FROM Рейс").fetchall()))  
 for i in range(len(self.cursor.execute("SELECT \* FROM Рейс").fetchall())):  
 self.ui.Delete\_Flight.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][0])))  
 self.ui.Delete\_Flight.setItem(1, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][1])))  
 self.ui.Delete\_Flight.setItem(2, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][2])))  
 self.ui.Delete\_Flight.setItem(3, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][3])))  
 self.ui.Delete\_Flight.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][4])))  
 self.ui.Delete\_Flight.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][5])))  
 self.ui.Delete\_Flight.setItem(6, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][6])))  
 self.ui.Delete\_Flight.setItem(7, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Рейс").fetchall()[i][7])))  
  
 def Delete\_Flight(self):  
 ID\_Flight = self.ui.Delete\_ID\_Flight.text()  
 self.cursor.execute("DELETE FROM Сопоставление WHERE ID\_РЕйса = (?);", (ID\_Flight))  
 self.cursor.execute("DELETE FROM Рейс WHERE ID\_Рейса = (?);", (ID\_Flight))  
 self.cursor.commit()  
  
 def Print\_Table(self):  
 Show\_table = int(self.ui.Switch\_Edit.text())  
 if (Show\_table == 1):  
 self.ui.Change\_Table.setRowCount(2)  
 self.ui.Change\_Table.setColumnCount(len(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark").fetchall()))  
 for i in range (len(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark;").fetchall())):  
 self.ui.Change\_Table.setItem(i, 0, QTableWidgetItem(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark;").fetchall()[i][0]))  
 self.ui.Change\_Table.setItem(i, 1, QTableWidgetItem(self.cursor.execute("SELECT [Название компании], [Местонахождение] FROM Aviapark;").fetchall()[i][1]))  
 elif (Show\_table == 2):  
 self.ui.Change\_Table.setRowCount(7)  
 self.ui.Change\_Table.setColumnCount(len(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()))  
 for i in range (len(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall())):  
 self.ui.Change\_Table.setItem(0, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][0])))  
 self.ui.Change\_Table.setItem(1, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][1]))  
 self.ui.Change\_Table.setItem(2, i, QTableWidgetItem(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][2]))  
 self.ui.Change\_Table.setItem(3, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][3])))  
 self.ui.Change\_Table.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][4])))  
 self.ui.Change\_Table.setItem(5, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][5])))  
 self.ui.Change\_Table.setItem(6, i, QTableWidgetItem(str(self.cursor.execute("SELECT Flight\_Number, [Откуда вылетает], [Куда летит], [Дата вылета], [Дата приземления], [Компания самолета], [ID\_Авиапарка] FROM Рейс").fetchall()[i][6])))  
 elif (Show\_table == 3):  
 self.ui.Change\_Table.setRowCount(5)  
 self.ui.Change\_Table.setColumnCount(len(self.cursor.execute("SELECT \* FROM Passenger").fetchall()))  
 for i in range (len(self.cursor.execute("SELECT \* FROM Passenger").fetchall())):  
 self.ui.Change\_Table.setItem(0, i, QTableWidgetItem(self.cursor.execute("SELECT \* FROM Passenger;").fetchall()[i][1]))  
 self.ui.Change\_Table.setItem(1, i, QTableWidgetItem(self.cursor.execute("SELECT \* FROM Passenger;").fetchall()[i][2]))  
 self.ui.Change\_Table.setItem(2, i, QTableWidgetItem(self.cursor.execute("SELECT \* FROM Passenger;").fetchall()[i][3]))  
 self.ui.Change\_Table.setItem(3, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger;").fetchall()[i][4])))  
 self.ui.Change\_Table.setItem(4, i, QTableWidgetItem(str(self.cursor.execute("SELECT \* FROM Passenger;").fetchall()[i][5])))  
 if (Show\_table != 1 and Show\_table != 2 and Show\_table != 3 ):  
 QMessageBox(QMessageBox.Critical, "Ошибка", 'Такой таблицы не существует').exec()  
  
 def Changed(self):  
 self.currentValue = self.ui.Change\_Table.currentItem().text()  
 if not self.editItems:  
 self.ui.Change\_Table.itemChanged.connect(self.Changed1)  
 self.editItems = True  
  
 def Changed1(self, item):  
 Show\_table = int(self.ui.Switch\_Edit.text())  
 value = item.text()  
 count\_column = item.column()  
 count\_row = item.row()  
 mass\_Id = []  
 if Show\_table == 1:  
 mass\_Id = self.cursor.execute("SELECT \* FROM Aviapark").fetchall()  
 if (count\_column == 0):  
 self.cursor.execute("UPDATE Aviapark SET [Название компании] = (?) WHERE [ID\_Авиапарка] = (?);", (value, mass\_Id[count\_row][0]))  
 elif (count\_column == 1):  
 self.cursor.execute("UPDATE Aviapark SET [Местонахождение] = (?) WHERE [ID\_Авиапарка] = (?);", (value, mass\_Id[count\_row][0]))  
 self.cursor.commit()  
 elif Show\_table == 2:  
 mass\_Id = self.cursor.execute("SELECT ID\_Рейса FROM Рейс").fetchall()  
 if count\_row == 0:  
 self.cursor.execute("UPDATE Рейс SET Flight\_Number = (?) WHERE [ID\_Рейса] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 1:  
 self.cursor.execute("UPDATE Рейс SET [Откуда вылетает] = (?) WHERE [ID\_Рейса] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 2:  
 self.cursor.execute("UPDATE Рейс SET [Куда летит] = (?) WHERE [ID\_Рейса] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 3:  
 self.cursor.execute("UPDATE Рейс SET [Дата вылета] = (?) WHERE [ID\_Рейса] = (?);", (str(value), mass\_Id[count\_column][0]))  
 elif count\_row == 4:  
 self.cursor.execute("UPDATE Рейс SET [Дата приземления] = (?) WHERE [ID\_РЕйса] = (?);", (str(value), mass\_Id[count\_column][0]))  
 elif count\_row == 5:  
 self.cursor.execute("UPDATE Рейса SET [Компания самолета] = (?) WHERE [ID\_Рейса] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 6:  
 self.cursor.execute("UPDATE Рейса SET [ID\_Авиапарка] = (?) WHERE [ID\_Рейса] = (?);", (int(value), mass\_Id[count\_column][0]))  
 self.cursor.commit()  
 elif Show\_table == 3:  
 mass\_Id = self.cursor.execute("SELECT [ID\_Пассажира] FROM Пассажир").fetchall()  
 if count\_row == 0:  
 self.cursor.execute("UPDATE Пассажир SET [Имя] = (?) WHERE [ID\_Пассажира] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 1:  
 self.cursor.execute("UPDATE Пассажир SET [Фамилия] = (?) WHERE [ID\_Пассажира] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 2:  
 self.cursor.execute("UPDATE Пассажир SET [Отчество] = (?) WHERE [ID\_Пассажира] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 3:  
 self.cursor.execute("UPDATE Пассажир SET PasportNumber = (?) WHERE [ID\_Пассажира] = (?);", (value, mass\_Id[count\_column][0]))  
 elif count\_row == 4:  
 self.cursor.execute("UPDATE Пассажир SET Price = (?) WHERE [ID\_Пассажира] = (?);", value, mass\_Id[count\_column][0])  
 self.cursor.commit()  
  
 self.ui.Change\_Table.itemChanged.disconnect(self.Changed1)  
 self.editItems = False  
 self.cursor.commit()  
  
def main():  
 app = QApplication(sys.argv)  
 application1 = Administration()  
 application1.show()  
 app.exec\_()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Демонстрация работы приложения**

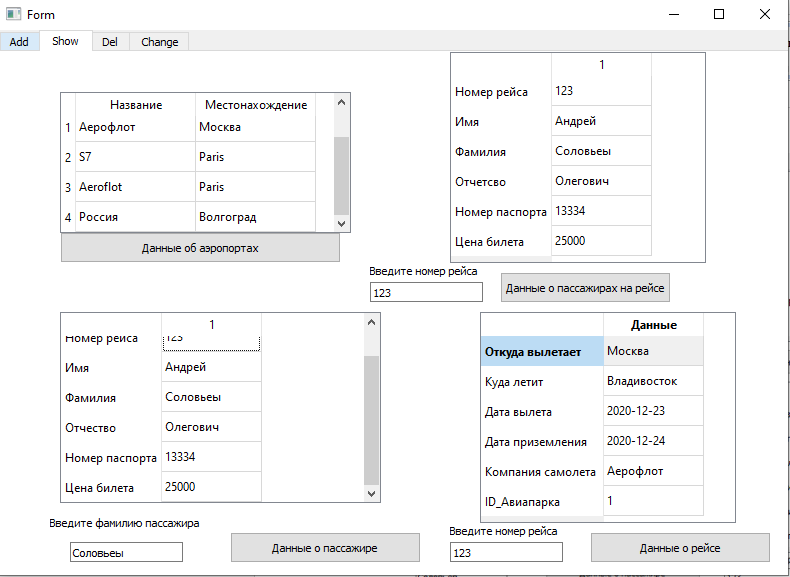
После запуска приложения появляется окно входа



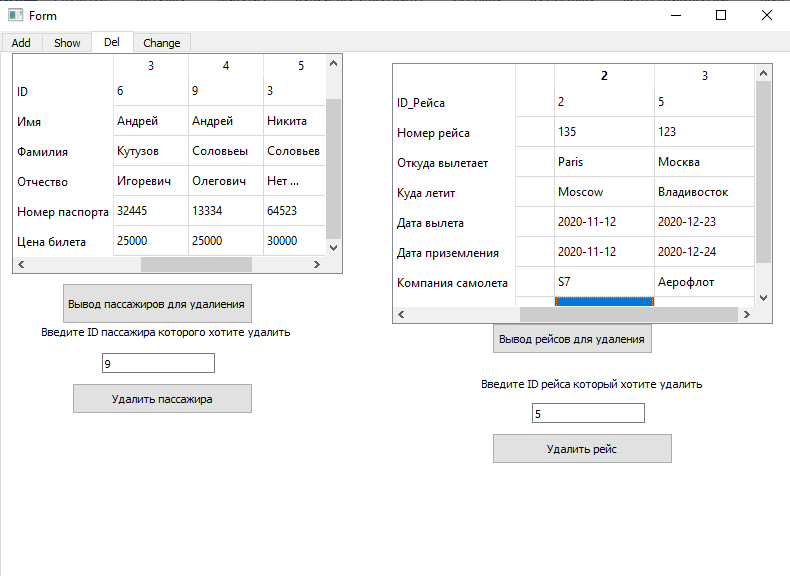
Заносим данные в таблицы



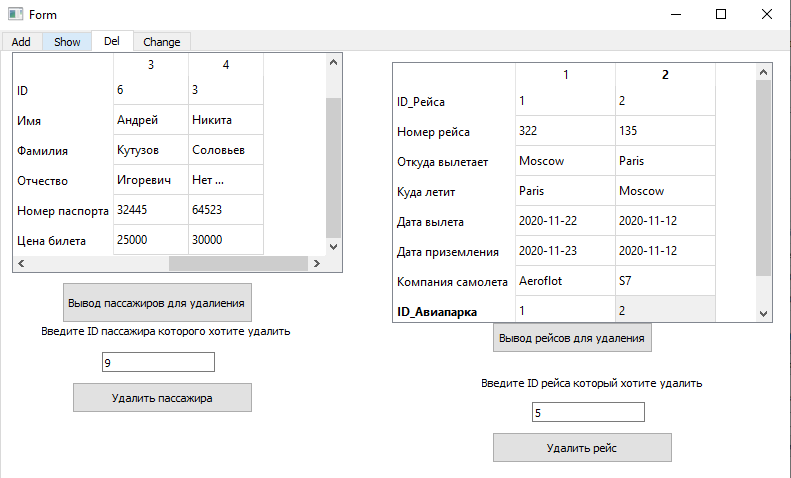
Проверим данные в окне Show



Удалим рейс, который добавили и пассажира на нем (Андрей Соловьев Олегович)

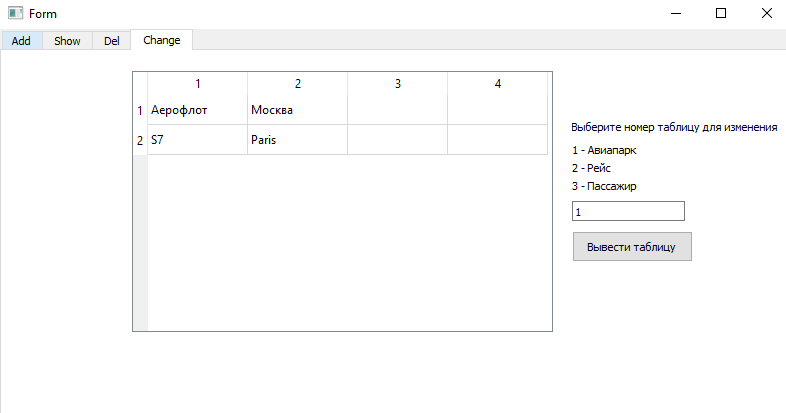
До нажатия кнопок удаления

После нажатия

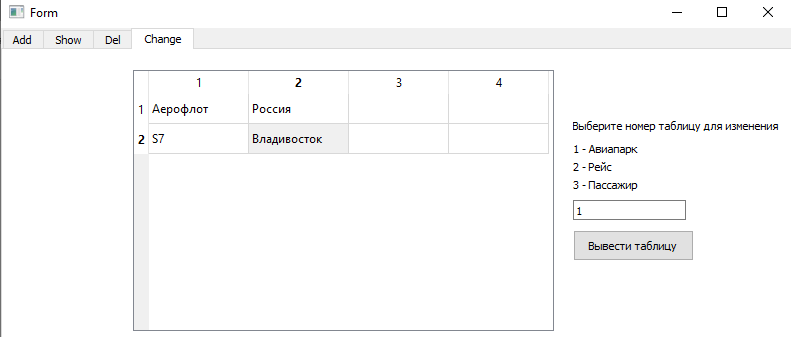


Изменение в таблице Авиапарка

Изначальный вид

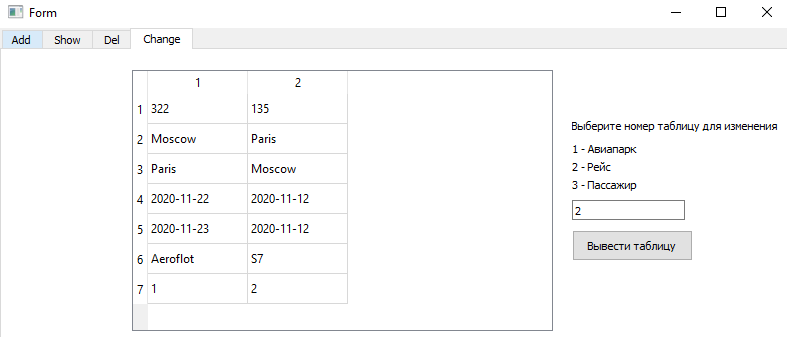


После изменений

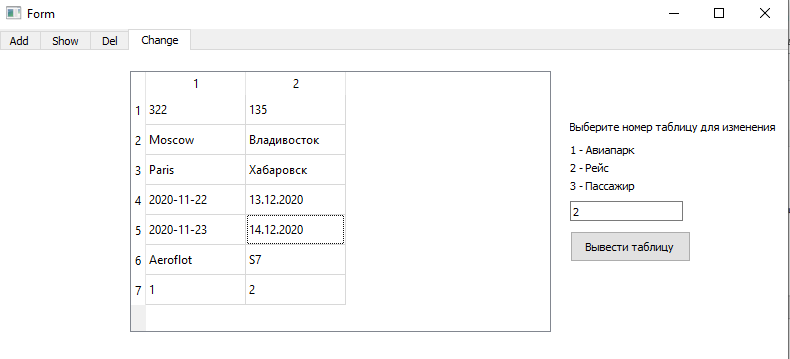


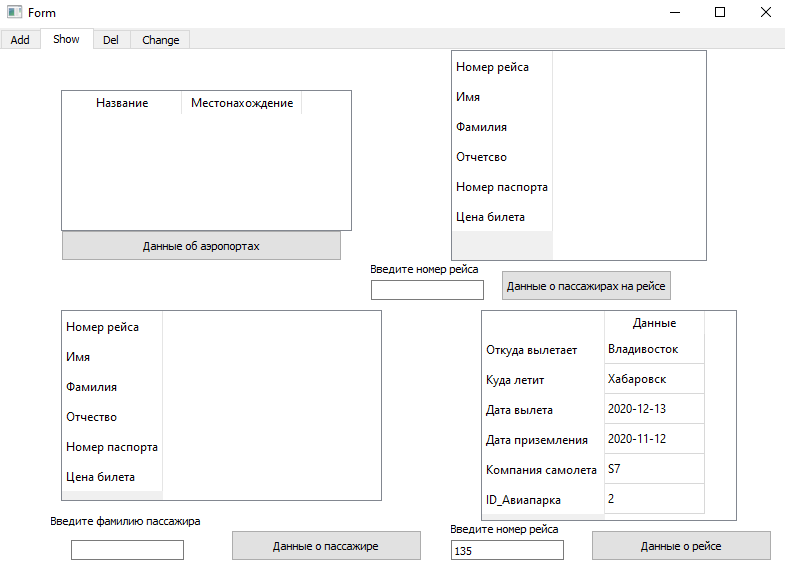
Изменение в таблице Рейса

До изменения



После изменения



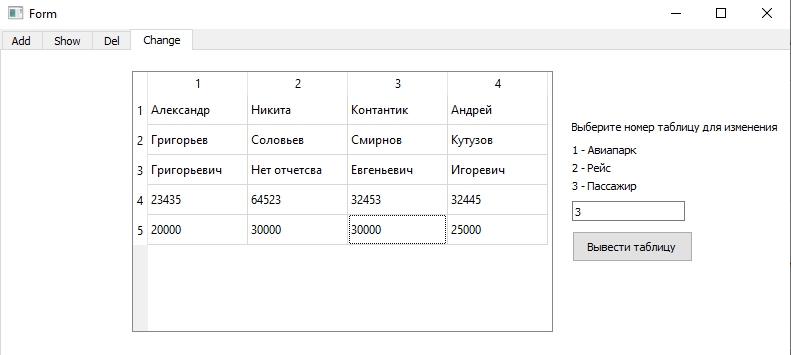
Проверка во вкладке Show

Изменение в таблице пассажиров

Изначальный вид



После изменений



Проверка во вкладке Show

