МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

Работа допущена к защите

\_\_\_\_\_\_\_\_\_\_\_\_\_Руководитель

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

## КУРСОВОЙ ПРОЕКТ

По дисциплине «Проетная деятельность»

На тему «Разработка информационной системы для аэропорта»

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_Будейра М.Р.

Шифр 190361

Институт приборостроения,автоматизации и информационных технологий

Направление подготовки 09.03.04 «программная инженерия»

Группа 92-ПГ

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лукьянов П.В.

Оценка: «\_\_\_\_\_\_\_\_\_\_\_\_\_» дата\_\_\_\_\_\_\_\_\_\_\_\_\_

Орел 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

## УТВЕРЖДАЮ:

## \_\_\_\_\_\_\_\_\_\_\_\_зав.кафедрой

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

## ЗАДАНИЕ

## На курсовой проект

## По дисциплине «Проетная деятельность»

## Студент Будейра М.Р.. Шифр 190361

Институт приборостроения,автоматизации и информационных технологий

Направление подготовки 09.03.04 «программная инженерия»

Группа 92-ПГ

1 Тема курсового проекта

## «Разработка информационной системы для аэропорта»

## 2 срок сдачи студентом законченной работы «\_\_\_» \_\_\_\_\_\_\_\_\_20\_\_\_

## 3 Исходные данные

## «Разработка информационной системы для аэропорта»

## 4 Содержание курсового проекта

## Описание модели

## Построение модели

## Реализация запросов

## Описание работы приложения

## 5 Отчетный материал курсовогог проекта

## Пояснительная записка курсового проекта; презентация

## Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лукьянов П.В.

## Задание принял к исполнению: «\_\_\_» \_\_\_\_\_\_\_\_\_20\_\_\_

## Подпись студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Содержание

## Введение .......................................................................................................5

## 1. Описание модели .....................................................................................6

## 2. Построение модели...................................................................................7

## 2.1 Этап концептуального проектирования................................................7

## 2.2 Этап логического проектирования........................................................7

## 3. Реализация запросов...............................................................................10

## 3.1 Получение информации по рейсам.....................................................10

## 3.2 Поиск рейсов по определенным критериям .......................................13

## 3.3 Получение списка заказов....................................................................13

## 3.4 Заказ билетов на выбранный рейс.......................................................14

## 3.5 Удаление заказа.....................................................................................14

## 3.6 Добавление рейса..................................................................................15

## 3.7 Редактирование рейса...........................................................................16

## 3.8 Удаление рейса......................................................................................17

## 3.8 Добавление пользователя .....................................................................17

## 3.9 Удаление пользователя..........................................................................18

## 4. Описание работы приложения................................................................ 19

## Заключение.....................................................................................................21

## Список использованных источников...........................................................22

## Приложение А................................................................................................23

## Введение

Целью данной курсовой работы является разработка многопользовательской системы бронирования авиабилетов.

Для этого необходимо решить следующие задачи:

- исследовать предметную область;

- построить концептуальную модель предметной области;

- построить даталогическую модель организации данных;

- реализовать соответствующее WEB-приложение;

- произвести развертывание и тестирование системы.

Объектом исследования является деятельность авиа-агентства. Агентство предоставляет услуги по заказу билетов на авиарейсы различных авиакомпаний. Каждый рейс следует из пункта отправления в пункт назначения. Рейс имеет дату и время вылета, дату и время прибытия. Каждый рейс выполняется самолетом определенной модели, в салоне которого есть места первого и второго класса. В зависимости от класса, билет имеет разную цену.

Для каждой модели самолета имеется определенное число мест каждого класса. Каждая модель самолета характеризуется авиакомпанией-производителем.

Ставится задача разработки многопользовательской системы, предназначенной для поиска и заказа билетов на авиарейсы. Каждому пользователю должна предоставляться возможность найти интересующие его рейсы, получить информацию о времени вылета и прибытия, авиакомпании, обслуживающей данный рейс, а также сделать заказ определенного количества билетов на выбранный рейс. Так же система должна предоставлять администратору системы WEB-интерфейс для её сопровождения

1. Описание модели

Наиболее важными элементами модели выбраны рейсы (flights) и заказы (orders).

Каждый рейс характеризуется следующей информацией:

- самолет, обслуживающий рейс,

- город отправления,

- город прибытия,

- дата отправления,

- дата прибытия,

- стоимость билетов первого класса,

- стоимость билетов второго класса.

- Заказ описывается следующим набором полей:

- пользователь, оформивший заказ,

- рейс, на который оформлен заказ,

- количество заказанных билетов первого класса,

- количество заказанных билетов второго класса,

- номер кредитной карты.

Отдельно хранится информация о самолетах с указанием модели самолета, количества свободных мест первого и второго класса, а также о компаниях, которым принадлежат данные самолеты.

Для входа в систему пользователь должен ввести свое имя. При первом входе в систему информация о пользователе сохраняется в отдельной таблице базы данных и используется при повторных входах в систему.

В системе предполагается использовать следующие роли пользователей:

Администратор: имеет права на просмотр полного списка рейсов, включая заказы.

Клиент: имеет право на отбор рейсов по определенным параметрам и на заказ билета.

Доступ к системе предоставляется только зарегистрированным пользователям.

2. Построение модели

2.1 Этап концептуального проектирования

На этапе концептуального проектирования использовалась модель «сущность-связь». Как видно из рисунка 1, в предметной области выделено 6 сущностей: пользователи (users), заказы (orders), рейсы (flights), города (cities), самолеты (aircrafts) и авиакомпании (companies). Связи между сущностями также изображены на рисунке. Атрибуты сущностей и их типы более подробно будут рассмотрены на этапе логического проектирования.



Рисунок 1 - ER- модель разработанной системы

2.2 Этап логического проектирования

Использование реляционной модели данных в системах управления базами данных было предложено в 1970 г. доктором Э. Ф. Коддом. Одним из важных достоинств реляционного подхода является его простота, а отсюда и доступность для понимания конечным пользователем.

Рассмотрим схемы отношений, используемых в реляционной модели:

R1 = ORDERS (OrderID, CreditCard, Number1cl, Number2cl, UserID, FlightID),

где:

- OrderID - идентификатор заказа;

- CreditCard - номер кредитной карты;

- Number1cl - количество заказанных билетов первого класса;

- Number2cl - количество билетов второго класса;

- UserID - идентификатор пользователя, оформившего заказ;

- FlightID - идентификатор рейса, на который оформлен данный

заказ.

R2 = FLIGHTS (FlightID, DateDeparture, DateArrival, Price1, Price2, AircraftID, CityDepatrureID, CityArrivalID),

где:

- FlightID - идентификатор рейса;

- DateDeparture - дата отправления;

- DateArrival - дата прибытия;

- Price1 - стоимость билета первого класса;

- Price2 - стоимость билета второго класса;

- CompanyID - идентификатор компании, организующей рейс;

- AircraftID - идентификатор самолета;

- CityDepatrureID - идентификатор города отправления;

- CityArrivalID - идентификатор города прибытия.

R3 = CITIES (CityID, CityName),

где:

- CityID - идентификатор города;

- CityName - название города.

R4 = AIRCRAFTS (AircraftID, AircraftModel, Count1, Count2),

где:

- AircraftID - идентификатор самолета;

- AircraftModel - модель самолета;

- Count1 - общее количество мест первого класса;

- Count2 - общее количество мест второго класса.

R5= COMPANIES (ID, COMPANY\_NAME, COMPANY\_PHONE),

где:

- CompanyID - идентификатор компании;

- CompanyName- название авиакомпании.

R6 = USERS (ID, USER\_NAME),

где

- UserID - идентификатор пользователя;

- UserLogin - имя пользователя;

- Password - пароль пользователя;

- Email - e-mail пользователя.

Данная схема отношений находится в 1 НФ, так все входящие в нее атрибуты являются атомарными (неделимыми). Более того, данная схема находится в НФ Бойса-Кодда, так как она находится в 1 НФ и никакой атрибут не зависит транзитивно ни от одного ключа.

Даталогическая схема базы данных приведена на рисунке 2. На нем помимо отношений и связей между ними показаны также соответствующие атрибутам типы данных.



Рисунок 2 - Даталогическая модель базы данных

3. Реализация запросов

В системе реализованы следующие виды запросов:

3.1 Получение информации по рейсам

Результаты данного запроса включают в себя следующую информацию: id рейса, название авиакомпании, модель, дату и время отправления, продолжительность полета, пункт назначения, цены на билеты каждого класса, количество свободных мест каждого класса.

Для того, чтобы представить данные именно в такой форме необходимо выполнить дополнительные операции. Так в базе данных нет поля продолжительность полета, зато есть поля дата отправления и дата прибытия. По ним легко получить продолжительность. Для её вычисления была создана функция timeFlight:

CREATE FUNCTION [dbo].[timeFlight]

(

@dateArrival datetime,

@dateDeparture datetime

)

RETURNS char(5)

BEGIN

RETURN

convert(char(2),datediff(hh,@dateDeparture,@dateArrival))+':'+

convert(char(2),datediff(mi,@dateDeparture,@dateArrival)-

datediff(hh,@dateDeparture,@dateArrival)\*60)

END

Данная функция возвращает продолжительность полета в виде строки из 5 символов в формате hh:mm.

Так же в базе не хранится информации о количестве свободных мест первого и второго класса на заданный рейс. Но данное значение для соответствующего класса можно вычислить, взяв общее количество мест класса, характерное для данной модели самолета, и вычтя из него количество забронированных мест. Подсчет количества забронированных мест идет суммированием по таблице заказов. Для вычисления свободных мест первого и второго класса созданы еще 2 скалярные функии: CountEmptyPlaces1cl и CountEmptyPlaces2cl.

CREATE FUNCTION [dbo].[CountEmptyPlaces1cl]

(

@flightID int

)

RETURNS int

BEGIN

DECLARE @count int;

IF EXISTS (SELECT OrderID FROM Orders WHERE FlightID=@flightID

AND Number1cl>0)

SELECT @count=Aircrafts.Count1

-(SELECT SUM(Number1cl) FROM Orders WHERE FlightID=@flightID)

FROM Flights,Aircrafts WHERE Flights.FlightID=@flightID

AND Flights.AircraftID=Aircrafts.AircraftID

Благодаря созданию перечисленых серверных сущностей мы оптимизируем выполнение данных видов запросов, тем самым получая выигрышь в производительности. Кроме того использование представление, позволяет упростить наисание клиентского приложения, делает запросы более компактными и наглядными.

3.2 Поиск рейсов по определенным критериям

Система позволяет выполнять отбор рейсов с заданными параметрами.

Критериями поиска являются:

- пункт назначения;

- дата (критерий - равенство) и время вылета (критерий - до/после включительно);

- дата (критерий - равенство) и время прибытия (критерий - до/после включительно).

Ни один из критериев не является обязательным для задания при выполнении поиска. Все критерии опциональны. Если ни один из критериев поиска не заполнен, то выводится полный список рейсов.

При задании критерия «дата», время вылета/прибытия может отсутствовать. В случае отсутствия даты, но указания времени выдается ошибка.

Эти критерии учитываются путем добавления необходимых условий к представлению FlightView в секцию WHERE. Формирование и выполнение данного запроса происходит в слое доступа к данным методом SearchFlights().

3.3 Получение списка заказов

Система предоставляет возможность просмотра заказов пользователя на выбранный рейс. В этом случае запрос осуществляется по идентификатору пользователя и рейса.

Если текущий пользователь имеет администраторские права, то он может просмотреть как заказы конкретного пользователя, так и все заказы на определенный рейс. В этом случае запрос осуществляется по идентификатору рейса.3.4 Заказ билетов на выбранный рейс

После выбора пользователем рейса, имеется возможность заказать определенное количество мест выбранного класса на данный рейс. Количество заказанных билетов вводится пользователем.

Система контролирует, чтобы количество заказанных билетов не превышало количество мест данного класса на рейсе. При количестве свободных мест равном нулю система запрещать бронировать билеты данного класса на данный рейс.

3.5 Удаление заказа

Пользователь может удалить свой заказ. Администратор имеет возможность удалить любой выбранный заказ. Билеты, забронированные данным заказом, переходят в разряд свободных.

Удаление заказа осуществляется посредством хранимой процедуры Delete\_Order:

CREATE PROCEDURE [dbo].[Delete\_Order]

@orderID int

AS

BEGIN

DELETE FROM Orders WHERE [OrderID]=@orderID

END

3.6 Добавление рейса

Добавление рейса осуществляется хранимой процедурой Insert\_FlightString:

CREATE PROCEDURE [dbo].[Insert\_FlightString]

@dateDeparture datetime,

@dateArrival datetime,

@price1 decimal(18,0),

@price2 decimal(18,0),

@company int,

@aircraft int,

@cityDeparture int,

@cityArrival int

AS

BEGIN

INSERT INTO [Flights]

([DateDeparture],[DateArrival],

[Price1],[Price2],

[CompanyID],[AircraftID],

[CityDepartureID],[CityArrivalID])

VALUES (

@dateDeparture,@dateArrival,

@price1,@price2,

@company,@aircraft,

@cityDeparture,@cityArrival)

END

Перед передачей в неё параметров осуществляется контроль их корректности. В частоности проверяется, чтобы город вылета и прибытия не совпадали. Цены за билеты должны быть положительными числами. Дата прилета должна быть больше даты вылета. Эти условия проверяются на клиентской стороне.

3.7 Редактирование рейса

Обновление информации о рейсе обеспечивается хранимой процедурой Update\_FlightString:

CREATE PROCEDURE [dbo].[Update\_FlightString]

@FlightID int,

@dateDeparture datetime,

@dateArrival datetime,

@price1 decimal(18,0),

@price2 decimal(18,0),

@companyID int,

@aircraftID int,

@cityDepartureID int,

@cityArrivalID int

AS

BEGIN

UPDATE Flights SET

dateDeparture = @dateDeparture,

dateArrival = @dateArrival,

price1 = @price1,

price2 = @price2,

cityDepartureID = @cityDepartureID,

cityArrivalID = @cityArrivalID,

companyID = @companyID,

aircraftID = @aircraftID

WHERE FlightID = @FlightID

END

При внесении изменений в рейс, на передаваемые параметры накладываются те же ограничении, как и при добавлении рейса.

3.8 Удаление рейса

При удалении рейса срабатывает триггер TR\_Flights\_Delete:

CREATE TRIGGER [TR\_Flights\_Delete] ON [dbo].[Flights] INSTEAD OF DELETE

AS

DELETE FROM Orders

WHERE Orders.FlightID= (SELECT top(1) deleted.FlightID FROM deleted)

DELETE FROM Flights

WHERE Flights.FlightID= (SELECT top(1) deleted.FlightID FROM deleted)

RETURN

Так как в базе данных существует связь FK\_Orders\_Flights между таблицами Flight и Orders, то невозможно удалить рейс, пока есть хотя бы одна заявка на него. Поэтому сначала должны быть удалены все связанные заявки, а уже потом - сам рейс. Эту логику и осуществляет даный триггер.

3.8 Добавление пользователя

При добавлении пользователя применяется хранимая процедура Insert\_User:

CREATE PROCEDURE [dbo].[Insert\_User]

@login nvarchar(20),

@password nvarchar(20),

@email nvarchar(50)

AS

BEGIN

INSERT Users

(UserLogin, Password, Email)

VALUES

(@login, @password, @email)

END

3.9 Удаление пользователя

Удаление пользователя осуществляется хранимой процедурой Delete\_User:

CREATE PROCEDURE [dbo].[Delete\_User]

@UserID int

AS

BEGIN

DELETE FROM Orders WHERE [UserID]=@UserID

DELETE FROM Users WHERE [UserID]=@UserID

END

Так как существует ограничение FK\_Orders\_Users, то перед удалением пользователя необходимо удалить все сделанные им заказы.

4. Описание работы приложения

Сайт позволяет работать с сохранением без регистрации на сайте.

После входа на сайт пользователь попадает на страницу поиска авиабилетов.

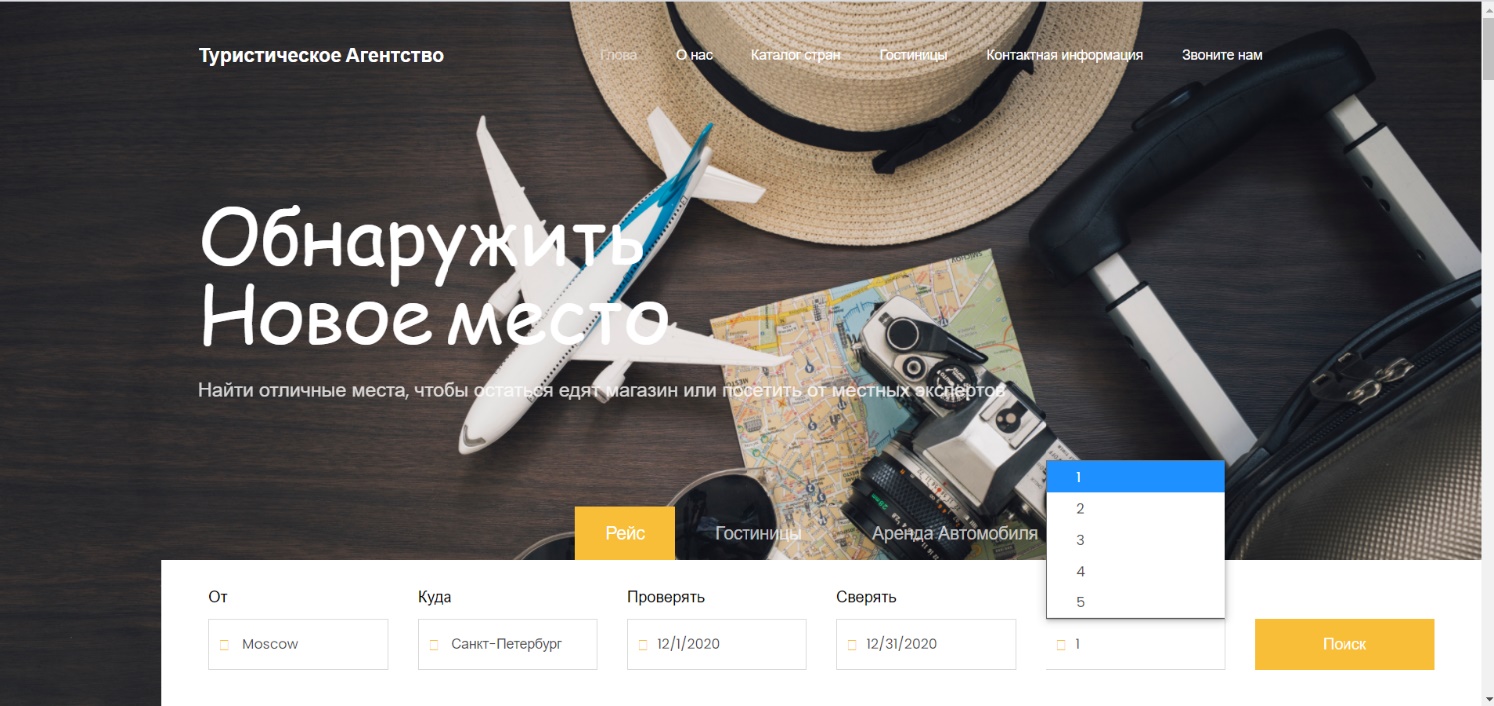


Рисунок 4 - Страница Flights пользовательского интерфейса

Страницы специальных возможностей предоставляют возможность поиска рейсов по таким критериям, как город отправления, и поиск рейсов по таким критериям, как город отправления, выбор и вы. Необязательно включать все параметры. Вам вообще не нужно об этом упоминать. В этом случае будут выбраны все рейсы. Однако выбор критерия может улучшить результаты запроса. Из-за того, что пользователь сам не вводит значения, а выбирает их из загруженных списков, вероятность ввода неверных значений уменьшается.

На параметры поиска налагается одно ограничение: время вылета или прибытия не может быть указано без соответствующей даты. При нарушении этого условия появляется сообщение об ошибке

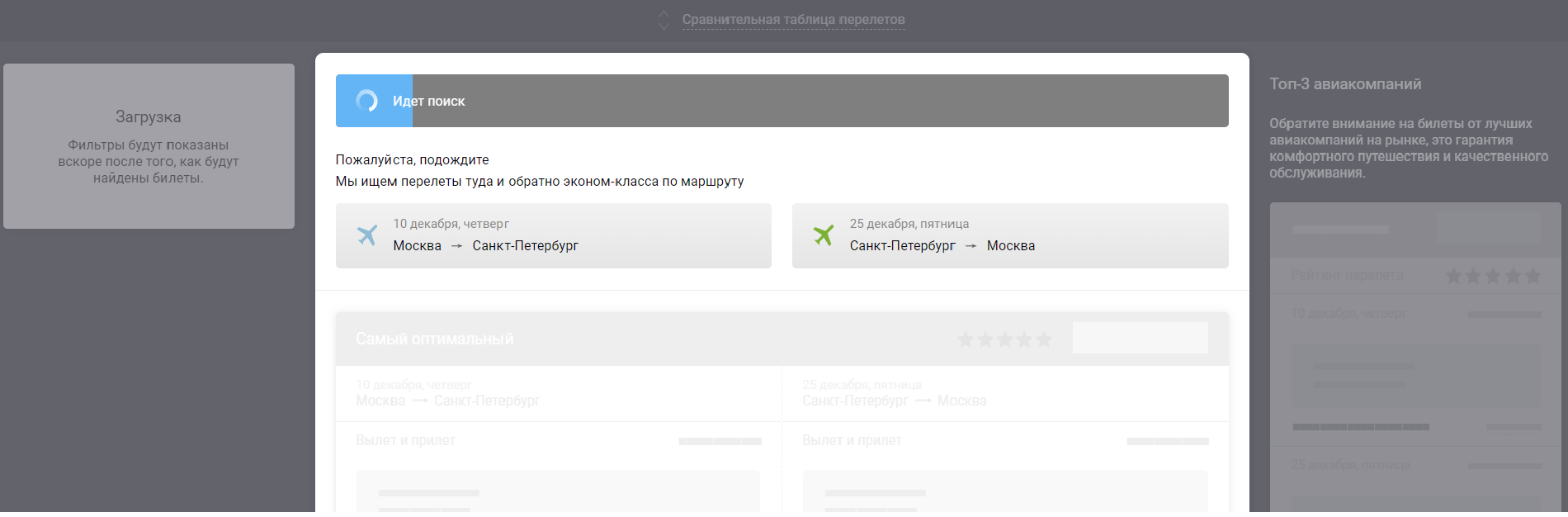


Рисунок 5 - Интерфейс поиска рейсов

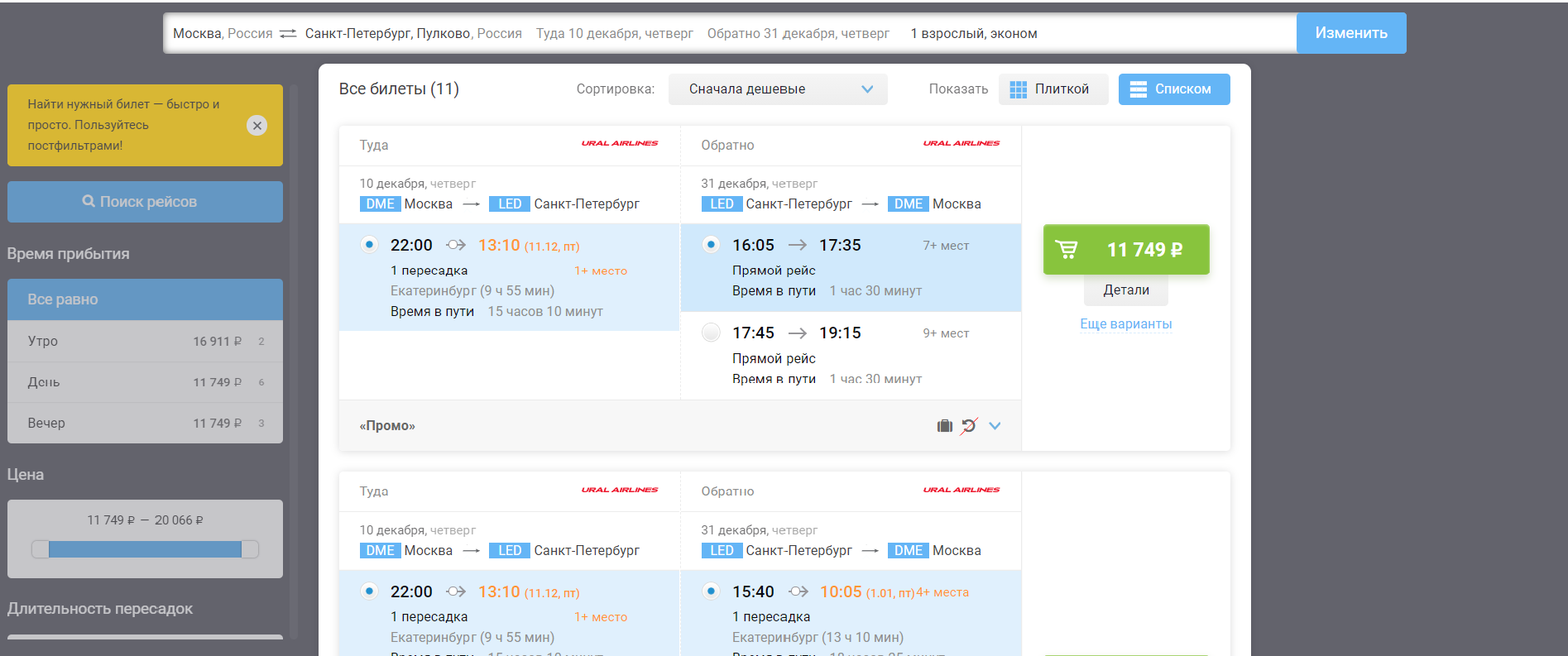


Рисунок 6 - Платежный интерфейс

Выбрав рейс, пользователь может сделать заказ на некоторое количество билетов первого и/или второго класса. Для этого ему необходимо на странице Orders указать нужное количество билетов и номер кредитной карты. Информацию о заявке невозможно будет сохранить, пока он не укажет количество билетов, не превышающее число свободных мест заданного класса и корректный номер кредитной карты (16 цифр). В случае неправильного заполнения полей система выводит соответствующие сообщения.

**Заключение**

В рамках данной курсовой работы была разработана система бронирования авиабилетов: спроектирована база данных для MS SQL Server 2005, а так же реализованы слой доступа к данным и WEB-интерфейс. При проектировании базы данных акцент делался на серверную часть. На стороне сервера были применены индексы, хранимые процедуры, функции, триггеры, представления. Клиентская часть была реализована посредством языка С# на платформе.NET. В частности: доступ к данным был обеспечен провайдером ADO.NET, а WEB-интерфейс построен с помощью ASP.NET. Дополнительно при реализации WEB-представления были использованы HTML, CSS, JavaScript, AJAX.

В итоге, в соответствии с поставленной задачей, мы получили систему бронирования авиабилетов. Данное приложение предоставляет удобный, интуитивно понятный интерфейс для клиентов компании. Кроме того, администратор получает возможность выполнять многие операции по сопровождению системы: по добавлению, удалении, редактированию пользователей, рейсов, заказов посредством того же (но уже расширенного в соответствии с его ролью) WEB-интерфейса.

Список использованных источников

1. Жилинский А.А. Самоучитель Microsoft SQL Server 2005 - СПб.: БХВ-Петербург, 2007. - 224 с.

2. Рихтер ДЖ. CLR via C#. Программирование на платформе.NET Framework 2.0 на языке С#. Мастер-класс./ Пер. с англ. - М.: Издательство «Русская редакция»; СПб.; Питер, 2007. - 656 с.

3. Экспозито Д. Microsoft ASP.NET 2.0. Базовый курс. Мастер-класс/ Пер. с англ. - М. Издательство «Русская редакция»; СПб.; Питер, 2007. - 688 с.

4. Экспозито Д. Microsoft ASP.NET 2.0. Углубленное изучение/ Пер. с англ. - М. Издательство «Русская редакция»; СПб.; Питер, 2007. - 592 с.

5. Microsoft Corporation. Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCAD/MSCE, MCDMA/ Пер. с англ. - 2-е изд., испр. - М.: Издательско-торговый дом «Русская редакция», 2003. - 512с.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**РЕАЛИЗАЦИЯ ПРОГРАММЫ**

Скрипт для создания базы данных

CREATE DATABASE [Group0703b]

GO

USE [Group0703b]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Companies]') AND type in (N'U'))

BEGIN

CREATE TABLE [dbo].[Companies](

[CompanyID] [int] IDENTITY(1,1) NOT NULL,

[CompanyName] [nvarchar](40) NULL,

CONSTRAINT [PK\_Companies] PRIMARY KEY CLUSTERED

(

[CompanyID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Users]') AND type in (N'U'))

BEGIN

CREATE TABLE [dbo].[Users](

[UserID] [int] IDENTITY(1,1) NOT NULL,

[UserLogin] [nvarchar](20) NOT NULL,

[Password] [nvarchar](20) NOT NULL,

[Email] [nvarchar](50) NULL,

CONSTRAINT [PK\_Users\_1] PRIMARY KEY CLUSTERED

(

[UserID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Cities]') AND type in (N'U'))

BEGIN

CREATE TABLE [dbo].[Cities](

[CityID] [int] IDENTITY(1,1) NOT NULL,

[CityName] [nvarchar](40) NULL,

CONSTRAINT [PK\_Cities] PRIMARY KEY CLUSTERED

(

[CityID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Aircrafts]') AND type in (N'U'))

BEGIN

CREATE TABLE [dbo].[Aircrafts](

[AircraftID] [int] IDENTITY(1,1) NOT NULL,

[AircraftModel] [nvarchar](40) NULL,

[Count1] [int] NULL,

[Count2] [int] NULL,

CONSTRAINT [PK\_Aicrafts] PRIMARY KEY CLUSTERED

(

[AircraftID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[timeFlight]') AND type in (N'FN', N'IF', N'TF', N'FS', N'FT'))

BEGIN

execute dbo.sp\_executesql @statement = N'CREATE FUNCTION [dbo].[timeFlight]

(

@dateArrival datetime,

@dateDeparture datetime

)

RETURNS char(5)

BEGIN

RETURN

convert(char(2),datediff(hh,@dateDeparture,@dateArrival))+'':''+

convert(char(2),datediff(mi,@dateDeparture,@dateArrival)-datediff(hh,@dateDeparture,@dateArrival)\*60)

END'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Flights]') AND type in (N'U'))

BEGIN

CREATE TABLE [dbo].[Flights](

[FlightID] [int] IDENTITY(1,1) NOT NULL,

[DateDeparture] [datetime] NULL,

[DateArrival] [datetime] NULL,

[Price1] [decimal](18, 0) NULL,

[Price2] [decimal](18, 0) NULL,

[CompanyID] [int] NULL,

[AircraftID] [int] NULL,

[CityDepartureID] [int] NULL,

[CityArrivalID] [int] NULL,

CONSTRAINT [PK\_Flights] PRIMARY KEY CLUSTERED

(

[FlightID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

GO

IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE object\_id = OBJECT\_ID(N'[dbo].[Flights]') AND name = N'IX\_Fligts\_AircraftID')

CREATE NONCLUSTERED INDEX [IX\_Fligts\_AircraftID] ON [dbo].[Flights]

(

[AircraftID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

GO

IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE object\_id = OBJECT\_ID(N'[dbo].[Flights]') AND name = N'IX\_Fligts\_CityArrivalID')

CREATE NONCLUSTERED INDEX [IX\_Fligts\_CityArrivalID] ON [dbo].[Flights]

(

[CityArrivalID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

GO

IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE object\_id = OBJECT\_ID(N'[dbo].[Flights]') AND name = N'IX\_Fligts\_CityDepartureID')

CREATE NONCLUSTERED INDEX [IX\_Fligts\_CityDepartureID] ON [dbo].[Flights]

(

[CityDepartureID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

GO

IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE object\_id = OBJECT\_ID(N'[dbo].[Flights]') AND name = N'IX\_Fligts\_CompanyID')

CREATE NONCLUSTERED INDEX [IX\_Fligts\_CompanyID] ON [dbo].[Flights]

(

[CompanyID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.triggers WHERE object\_id = OBJECT\_ID(N'[dbo].[TR\_Flights\_Delete]'))

EXEC dbo.sp\_executesql @statement = N'CREATE TRIGGER [dbo].[TR\_Flights\_Delete] ON [dbo].[Flights] INSTEAD OF DELETE

AS

DELETE FROM Orders

WHERE Orders.FlightID= (SELECT top(1) deleted.FlightID FROM deleted)

DELETE FROM Flights

WHERE Flights.FlightID= (SELECT top(1) deleted.FlightID FROM deleted)

RETURN'

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Orders]') AND type in (N'U'))

BEGIN

CREATE TABLE [dbo].[Orders](

[OrderID] [int] IDENTITY(1,1) NOT NULL,

[CreditCard] [varchar](16) NULL,

[Number1cl] [int] NULL,

[Number2cl] [int] NULL,

[UserID] [int] NULL,

[FlightID] [int] NULL,

CONSTRAINT [PK\_Orders] PRIMARY KEY CLUSTERED

(

[OrderID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

END

GO

IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE object\_id = OBJECT\_ID(N'[dbo].[Orders]') AND name = N'IX\_Orders\_UserID')

CREATE NONCLUSTERED INDEX [IX\_Orders\_UserID] ON [dbo].[Orders]

(

[UserID] ASC

)WITH (PAD\_INDEX = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Insert\_User]') AND type in (N'P', N'PC'))

BEGIN

EXEC dbo.sp\_executesql @statement = N'-- Вставка новой записи в таблицу Users

CREATE PROCEDURE [dbo].[Insert\_User]

@login nvarchar(20),

@password nvarchar(20),

@email nvarchar(50)

AS

BEGIN

INSERT Users

(UserLogin, Password, Email)

VALUES

(@login, @password, @email)

END

;

'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Delete\_User]') AND type in (N'P', N'PC'))

BEGIN

EXEC dbo.sp\_executesql @statement = N'-- Удаление записи из таблицы Users

CREATE PROCEDURE [dbo].[Delete\_User]

@UserID int

AS

BEGIN

DELETE FROM Users WHERE [UserID]=@UserID

END

;

'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[CountEmptyPlaces1cl]') AND type in (N'FN', N'IF', N'TF', N'FS', N'FT'))

BEGIN

execute dbo.sp\_executesql @statement = N'CREATE FUNCTION [dbo].[CountEmptyPlaces1cl]

(

@flightID int

)

RETURNS int

BEGIN

DECLARE @count int;

IF EXISTS (SELECT OrderID FROM Orders WHERE FlightID=@flightID AND Number1cl>0)

SELECT @count=Aircrafts.Count1

-(SELECT SUM(Number1cl) FROM Orders WHERE FlightID=@flightID)

FROM Flights,Aircrafts WHERE Flights.FlightID=@flightID

AND Flights.AircraftID=Aircrafts.AircraftID

ELSE

SELECT @count=Aircrafts.Count1

FROM Flights,Aircrafts WHERE Flights.FlightID=@flightID

AND Flights.AircraftID=Aircrafts.AircraftID

RETURN @count

END'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[CountEmptyPlaces2cl]') AND type in (N'FN', N'IF', N'TF', N'FS', N'FT'))

BEGIN

execute dbo.sp\_executesql @statement = N'CREATE FUNCTION [dbo].[CountEmptyPlaces2cl]

(

@flightID int

)

RETURNS int

BEGIN

DECLARE @count int;

IF EXISTS (SELECT OrderID FROM Orders WHERE FlightID=@flightID AND Number2cl>0)

SELECT @count=Aircrafts.Count2

-(SELECT SUM(Number2cl) FROM Orders WHERE FlightID=@flightID)

FROM Flights,Aircrafts WHERE Flights.FlightID=@flightID

AND Flights.AircraftID=Aircrafts.AircraftID

ELSE

SELECT @count=Aircrafts.Count2

FROM Flights,Aircrafts WHERE Flights.FlightID=@flightID

AND Flights.AircraftID=Aircrafts.AircraftID

RETURN @count

END'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Insert\_FlightString]') AND type in (N'P', N'PC'))

BEGIN

EXEC dbo.sp\_executesql @statement = N'-- Вставка новой записи в таблицу Flight

CREATE PROCEDURE [dbo].[Insert\_FlightString]

@dateDeparture datetime,

@dateArrival datetime,

@price1 decimal(18,0),

@price2 decimal(18,0),

@company int,

@aircraft int,

@cityDeparture int,

@cityArrival int

AS

BEGIN

INSERT INTO [Flights]

([DateDeparture],[DateArrival],

[Price1],[Price2],

[CompanyID],[AircraftID],

[CityDepartureID],[CityArrivalID])

VALUES (

@dateDeparture,@dateArrival,

@price1,@price2,

@company,@aircraft,

@cityDeparture,@cityArrival)

END

;

'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Update\_FlightString]') AND type in (N'P', N'PC'))

BEGIN

EXEC dbo.sp\_executesql @statement = N'-- Обновление записи в таблице Flights

CREATE PROCEDURE [dbo].[Update\_FlightString]

@FlightID int,

@dateDeparture datetime,

@dateArrival datetime,

@price1 decimal(18,0),

@price2 decimal(18,0),

@companyID int,

@aircraftID int,

@cityDepartureID int,

@cityArrivalID int

AS

BEGIN

UPDATE Flights SET

dateDeparture = @dateDeparture,

dateArrival = @dateArrival,

price1 = @price1,

price2 = @price2,

cityDepartureID = @cityDepartureID,

cityArrivalID = @cityArrivalID,

companyID = @companyID,

aircraftID = @aircraftID

WHERE FlightID = @FlightID

END

;

'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Delete\_Order]') AND type in (N'P', N'PC'))

BEGIN

EXEC dbo.sp\_executesql @statement = N'-- Удаление записи из таблицы Orders по OrderID

CREATE PROCEDURE [dbo].[Delete\_Order]

@orderID int

AS

BEGIN

DELETE FROM Orders WHERE [OrderID]=@orderID

END

;

'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Delete\_OrderByUserID]') AND type in (N'P', N'PC'))

BEGIN

EXEC dbo.sp\_executesql @statement = N'-- Удаление записей из таблицы Orders по UserID

CREATE PROCEDURE [dbo].[Delete\_OrderByUserID]

@UserID int

AS

BEGIN

DELETE FROM Orders WHERE [UserID]=@UserID

END

;

'

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

IF NOT EXISTS (SELECT \* FROM sys.views WHERE object\_id = OBJECT\_ID(N'[dbo].[FlightView]'))

EXEC dbo.sp\_executesql @statement = N'CREATE VIEW [dbo].[FlightView] AS

SELECT

FlightID,

DateDeparture,

DateArrival,

dbo.timeFlight(DateArrival,DateDeparture) AS TimeFlight,

CityDepartureID,

CityArrivalID,

DepartureCities.CityName AS CityDeparture,

ArrivalCities.CityName AS CityArrival,

CompanyName,

AircraftModel,

Price1,

Price2,

dbo.CountEmptyPlaces1cl(FlightID) AS EmptyPlace1cl,

dbo.CountEmptyPlaces2cl(FlightID) AS EmptyPlace2cl

FROM dbo.Flights INNER JOIN dbo.Companies

ON Group0703b.dbo.Flights.CompanyID = Group0703b.dbo.Companies.CompanyID

INNER JOIN Group0703b.dbo.Aircrafts

ON Group0703b.dbo.Flights.AircraftID = Group0703b.dbo.Aircrafts.AircraftID

LEFT OUTER JOIN Group0703b.dbo.Cities AS DepartureCities

ON Group0703b.dbo.Flights.CityDepartureID = DepartureCities.CityID

LEFT OUTER JOIN Group0703b.dbo.Cities AS ArrivalCities

ON Group0703b.dbo.Flights.CityArrivalID = ArrivalCities.CityID '

GO

IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE object\_id = OBJECT\_ID(N'[dbo].[FK\_Flights\_Aircrafts]') AND parent\_object\_id = OBJECT\_ID(N'[dbo].[Flights]'))

ALTER TABLE [dbo].[Flights] WITH CHECK ADD CONSTRAINT [FK\_Flights\_Aircrafts] FOREIGN KEY([AircraftID])

REFERENCES [dbo].[Aircrafts] ([AircraftID])

GO

ALTER TABLE [dbo].[Flights] CHECK CONSTRAINT [FK\_Flights\_Aircrafts]

GO

IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE object\_id = OBJECT\_ID(N'[dbo].[FK\_Flights\_Cities\_Arrival]') AND parent\_object\_id = OBJECT\_ID(N'[dbo].[Flights]'))

ALTER TABLE [dbo].[Flights] WITH CHECK ADD CONSTRAINT [FK\_Flights\_Cities\_Arrival] FOREIGN KEY([CityArrivalID])

REFERENCES [dbo].[Cities] ([CityID])

GO

ALTER TABLE [dbo].[Flights] CHECK CONSTRAINT [FK\_Flights\_Cities\_Arrival]

GO

IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE object\_id = OBJECT\_ID(N'[dbo].[FK\_Flights\_Cities\_Departure]') AND parent\_object\_id = OBJECT\_ID(N'[dbo].[Flights]'))

ALTER TABLE [dbo].[Flights] WITH CHECK ADD CONSTRAINT [FK\_Flights\_Cities\_Departure] FOREIGN KEY([CityDepartureID])

REFERENCES [dbo].[Cities] ([CityID])

GO

ALTER TABLE [dbo].[Flights] CHECK CONSTRAINT [FK\_Flights\_Cities\_Departure]

GO

IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE object\_id = OBJECT\_ID(N'[dbo].[FK\_Flights\_Companies]') AND parent\_object\_id = OBJECT\_ID(N'[dbo].[Flights]'))

ALTER TABLE [dbo].[Flights] WITH CHECK ADD CONSTRAINT [FK\_Flights\_Companies] FOREIGN KEY([CompanyID])

REFERENCES [dbo].[Companies] ([CompanyID])

GO

ALTER TABLE [dbo].[Flights] CHECK CONSTRAINT [FK\_Flights\_Companies]

GO

IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE object\_id = OBJECT\_ID(N'[dbo].[FK\_Orders\_Flights]') AND parent\_object\_id = OBJECT\_ID(N'[dbo].[Orders]'))

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_Flights] FOREIGN KEY([FlightID])

REFERENCES [dbo].[Flights] ([FlightID])

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_Flights]

GO

IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE object\_id = OBJECT\_ID(N'[dbo].[FK\_Orders\_Users]') AND parent\_object\_id = OBJECT\_ID(N'[dbo].[Orders]'))

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_Users] FOREIGN KEY([UserID])

REFERENCES [dbo].[Users] ([UserID])

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_Users]