ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

**ВЫСШЕГО ОБРАЗОВАНИЯ**

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

*Высшая Школа Бизнеса*

**Проектирование и разработка базы данных**

**для авиакомпании**

Курсовая работа студента

Гасанова Ильяса Запировича

2 курса, направление подготовки: 38.03.05 Бизнес-информатика

образовательной программы «Бизнес-информатика»

Научный руководитель

ст.преп.

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

К. С. Гоменюк

Москва 2021

Оглавление

[Введение 3](#_Toc133949690)

[Глава 1. Теоретические основы 5](#_Toc133949691)

[1.1 Теория проектирования 5](#_Toc133949692)

[1.2 Теория разработки 7](#_Toc133949693)

[Глава 2. Инструментальные методы 8](#_Toc133949694)

[2.1 Описание предметной области 8](#_Toc133949695)

[2.1.1 Сущности и атрибуты 9](#_Toc133949696)

[2.1.2 Связи 10](#_Toc133949697)

[2.2 Технические и функциональные требования 11](#_Toc133949698)

[Глава 3. Реализация 13](#_Toc133949699)

[3.1 Физическая схема 13](#_Toc133949700)

[3.2 Хранимые процедуры 14](#_Toc133949701)

[3.3 Триггеры 15](#_Toc133949702)

[3.4 Графический интерфейс 16](#_Toc133949703)

[Заключение 23](#_Toc133949704)

[Список литературы 24](#_Toc133949705)

[Приложение 25](#_Toc133949706)

# Введение

В настоящее время компании сталкиваются со всё большими объёмами данных, растущих в геометрической прогрессии. Если раньше хранение и использование информации могло осуществляться в бумажном виде, то сегодня в большинстве компаний это уже не представляется возможным. Данных попросту слишком много, чтобы хранить их без потерь и противоречий, а также уметь эффективно взаимодействовать с ними. Инструментом решения данной проблемы являются базы данных (БД) и сопутствующие им системы управления базами данных (СУБД), позволяющие структурировать информацию и управлять ею. Такое умение, безусловно, актуально уже не первый десяток лет, и потребность в нём будет только расти.

Стоит отметить, что при переходе с бумажных носителей на электронные, принцип работы с данными меняется: всегда, как правило, с информацией взаимодействовал сотрудник компании, теперь же – ещё и пользователь. Это означает, что следует организовать безопасность хранения данных и аккуратно задать для пользователя уровень их владения. Поэтому помимо разработки самой базы данных в рамках этой работы мы реализовали и приложение для пользователя. Помимо всего прочего, через интерфейс работать с данными гораздо проще и понятнее даже на интуитивном уровне.

**Объектом исследования** данной работы является авиакомпания.

**Предметом исследования** является структура авиакомпании и взаимодействие между её частями.

**Цель исследования** заключается в проектировании и разработке базы данных и графического интерфейса для взаимодействия пользователя с данными.

**Задачи исследования:**

* Изучение предметной области авиакомпании
* Изучение технических и функциональных требований
* Проектирование базы данных авиакомпании
* Реализация базы данных
* Реализация графического интерфейса для управления базой данных пользователем

**Используемые технологии:**

1. Инструмент моделирования баз данных ERD Plus
2. Приложение для управления базами данных DBeaver
3. Среда разработки PyCharm

**Структура работы:**

Сначала рассмотрены теоретические основы проектирования и разработки, затем описаны предметная область и технические требования. В результате реализованы физическая схема и приложение для пользователя.

# Глава 1. Теоретические основы

## 1.1 Теория проектирования

В данной работе рассматривается проектирование реляционной базы данных, являющейся на сегодняшний день самой распространённой. В такой модели данные хранятся только в табличном виде, за счёт чего они приобретают структурированность, независимость и компактность. В реляционной модели также для пользователя должны присутствовать методы создания новых таблиц на основе имеющихся. Например, пользователю будут доступны такие операции над таблицами, как проекция и выборка, позволяющие сократить размер исходной таблицы при указании определённых ограничений на данные. В нашем случае ограничением может стать желание пользователя приобрести авиабилет в конкретный промежуток времени, а не в любой день.

На этапе построения концептуальной модели мы ещё не знаем всех вариантов использования данных, поэтому она не должна зависеть от таких аспектов, как, например, СУБД, язык программирования и требования пользователя. Строгое разделение между моделью и её реализацией помогает создать эту независимость от данных.

Итак, для решения задачи проектирования необходимо определить сущности и списки их атрибутов. Типом сущности является таблица, а атрибутами – поля в ней. Также не менее важной задачей на данном этапе исследования ставится соблюдение целостности данных. Данные должны соответствовать внутренней логике, структуре и явно заданным правилам. К примеру, часто контролируются уникальность некоторых атрибутов, принадлежность значений атрибутов к возможному диапазону значений и отсутствие неопределённых значений.

Для устранения аномалий, а также для повышения производительности и удобства управления данными существует метод нормализации. Нормализация удаляет из базы избыточные функциональные зависимости (ФЗ) между атрибутами. От лишних ФЗ стоит избавляться, потому что они являются ограничениями целостности, соответствие которым будет проверяться в СУБД при каждом обновлении данных.

Именно базы данных в третьей нормальной форме чаще всего используются в любой сфере, связанной с большим количеством бизнес-процессов и необходимостью хранения информации в оптимальном виде, поэтому будем добиваться 3НФ. Для этого необходимо сначала иметь 2НФ и 1НФ. Чтобы получить первую нормальную форму все атрибуты должны хранить единственное значение (быть атомарными), а также должны отсутствовать повторяющиеся строки. Далее проследим за наличием первичного ключа и за тем, что все атрибуты зависят от всего первичного ключа, а не от его части – получили вторую нормальную форму. Наконец, если все атрибуты зависят только от ПК, но не от других атрибутов, то мы имеем 3НФ.

**Основными этапами проектирования являются:**

1. **Инфологическое** – заключается в построении семантической модели, также называется концептуальным проектированием. Определяются сущности и связи между ними. На этом этапе построенная нами ER-модель помогла визуализировать все данные и избежать потенциальных ошибок в самом начале работы.
2. **Даталогическое** – представляет собой создание схемы БД на основе модели данных. Добавляются атрибуты и их тип данных. В нашем случае с реляционной БД происходит нормализация отношений.
3. **Физическое** – схема БД ориентируется на особенности выбранной СУБД. Определяется физический способ хранения данных, создаются дополнительные структуры (например, индексы), добавляются триггеры.

## 1.2 Теория разработки

Помимо самой базы данных к ней прилагается графический интерфейс. Для его разработки требуется выбрать подходящий инструмент – в нашем случае это графическая кроссплатформенная библиотека Tkinter для Python. Tkinter входит в стандартную библиотеку Python, предназначена для взаимодействия с пользователем с помощью окон. Рассчитана на небольшие приложения с простым интерфейсом, что вполне подходит для работы с нашей базой данных. Для доступа к серверу, на котором будут находиться все данные, необходимо подключение к VPN серверу ВШЭ, а также использование отдельного модуля Pyodbc, который позволяет использовать программный интерфейс ODBC с целью доступа к информации.

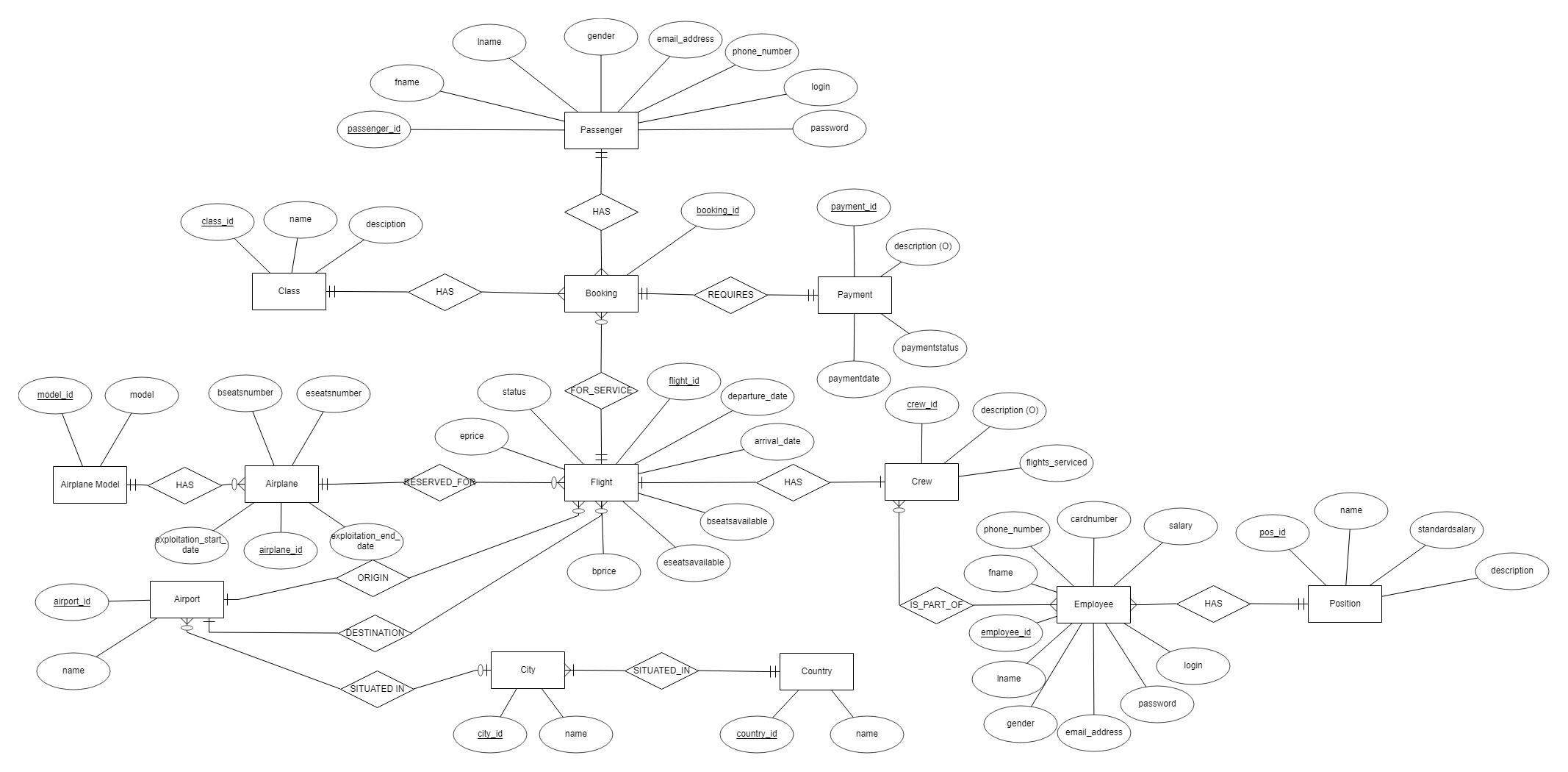
Подводя промежуточный итог, отметим, что мы рассмотрели задачу проектирования с теоретической точки зрения, выбрали реляционную модель базы данных в третьей нормальной форме, а также разделили задачу проектирования БД на три последовательных этапа. Кроме того, были выявлены особенности выбранных инструментов разработки.

# Глава 2. Инструментальные методы

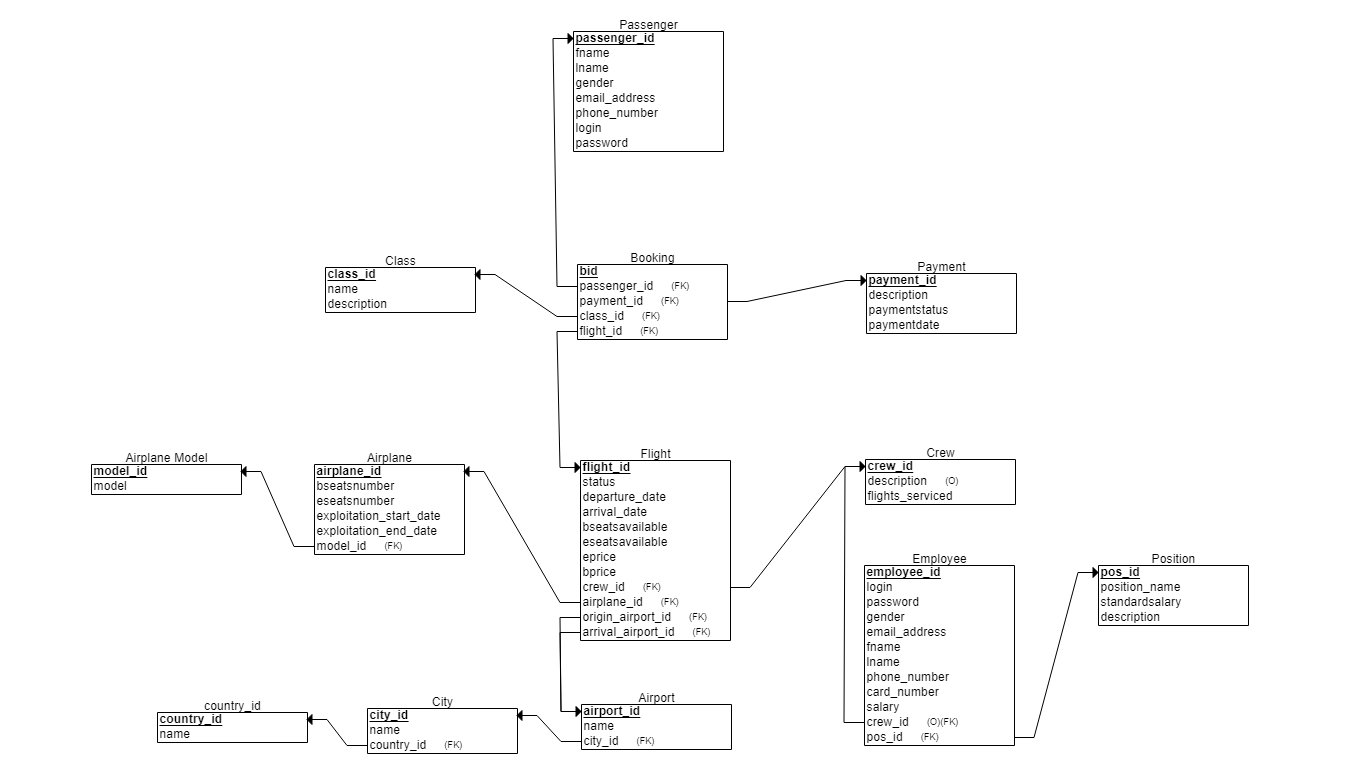
## 2.1 Описание предметной области

За рассматриваемую предметную область взята авиакомпания. Организация осуществляет внутренние и международные пассажирские авиаперелёты. Клиент выбирает дату и желаемое время полёта, после чего производится первичное бронирование билета. Если на выбранный рейс имеются места, на класс которых претендует потенциальный клиент, то на его имя выставляется счёт, без которого является невозможным будущий перелёт. На каждый рейс назначается отдельный самолёт, количество мест на котором может как совпадать со стандартным для определённого борта, так и быть уникальным. Для каждого рейса также хранится информация об аэропортах, которые используются в процессе перелёта. Также хранятся данные о команде, которая привязана и далее будет обслуживать данный рейс. Команда состоит из сотрудников разных должностей.

Для наглядного представления предметной области воспользуемся инструментом моделирования БД ERD Plus и осуществим первые два этапа проектирования (рис. 1 и 2).



*Рисунок 1 – инфологическая модель*



*Рисунок 2 – даталогическая модель*

### 2.1.1 Сущности и атрибуты

Под сущностями понимаются объекты, о которых необходимо хранить информацию. Мир, по сути, построен из сущностей.

Определённые нами названия сущностей почти всегда говорят сами за себя, но всё же следует задать список с точными значениями таблиц. Итак, в рамках нашего моделирования были созданы 13 сущностей:

1. Бронирование (booking) – данные о бронировании
2. Класс (class) – категория забронированного места: экономический/ бизнес-класс
3. Платёж (payment) – данные о платеже
4. Пассажир (passenger) – информация о пассажире
5. Перелёт (flight) – данные о полёте
6. Самолёт (airplane) – сведения о самолёте
7. Модель самолёта (airplanemodel) – данные о модели самолёта
8. Аэропорт (airport) – данные об аэропорте вылета/прилёта
9. Город (city) – город вылета/прилёта
10. Страна (country) – страна вылета/прилёта
11. Экипаж (crew) – сведения об экипаже
12. Сотрудник (employee) – информация о сотруднике
13. Должность (position) – сведения о должности сотрудника

Любая сущность имеет свойства. В терминологии проектирования БД эти свойства принято называть атрибутами, которые задают идентификатор сущности и прочие параметры с обязательными указанием типа этого атрибута. Некоторые атрибуты могут быть первичными и внешними ключами (Primary Key и Foreign Key). Под первичными ключами понимают атрибуты или наборы атрибутов, идентифицирующих сущность. При этом важно, чтобы PK были и минимальными ключами, так как это обеспечивает отсутствие избыточности в зависимостях от этих ключей. Избыточность может вызывать аномалии и неправильные результаты. А внешние ключи – это атрибуты, осуществляющие однозначную связь между таблицами. Некоторые поля в таблице могут быть опциональными и по умолчанию хранить пустое значение, они обозначаются как Optional (O). Всего для имеющихся сущностей было создано 67 атрибутов.

### 2.1.2 Связи

После задания сущностей и их атрибутов, для объединения их в одну систему, между сущностями строятся связи. Связи типа «один к одному» и «один ко многим» всегда могут быть представлены с помощью внешнего ключа, помещаемого в одну из переменных-отношений, участвующих в данной связи.

|  |  |  |  |
| --- | --- | --- | --- |
| **Связь** | **Отношение** | **Главная таблица** | **Подчинённая таблица** |
| бронирование – пассажир | многие к одному | booking | passenger |
| бронирование – полёт | один к одному | booking | flight |
| бронирование – класс | многие к одному | booking | class |
| бронирование – платёж | один к одному | booking | payment |
| перелёт – самолёт | многие к одному | flight | airplane |
| перелёт – аэропорт | многие к одному | flight | airport |
| перелёт – экипаж | один к одному | flight | crew |
| самолёт – модель самолёта | многие к одному | airplane | airplanemodel |
| аэропорт – город аэропорта | многие к одному | airport | city |
| город – страна, в которой он находится | многие к одному | city | country |
| сотрудник – экипаж | один к одному | crew | employee |
| сотрудник – должность | один к одному | employee | position |

*Таблица 1 – Связи между таблиц БД*

## 2.2 Технические и функциональные требования

Требуется разработать приложение с графическим интерфейсом, которым могут пользоваться и обычные пользователи, и сотрудники компании. При авторизации у сотрудника компании должен присутствовать доступ к редактированию данных, а у клиента, наоборот, – отсутствовать.

**Приложение должно иметь следующий функционал:**

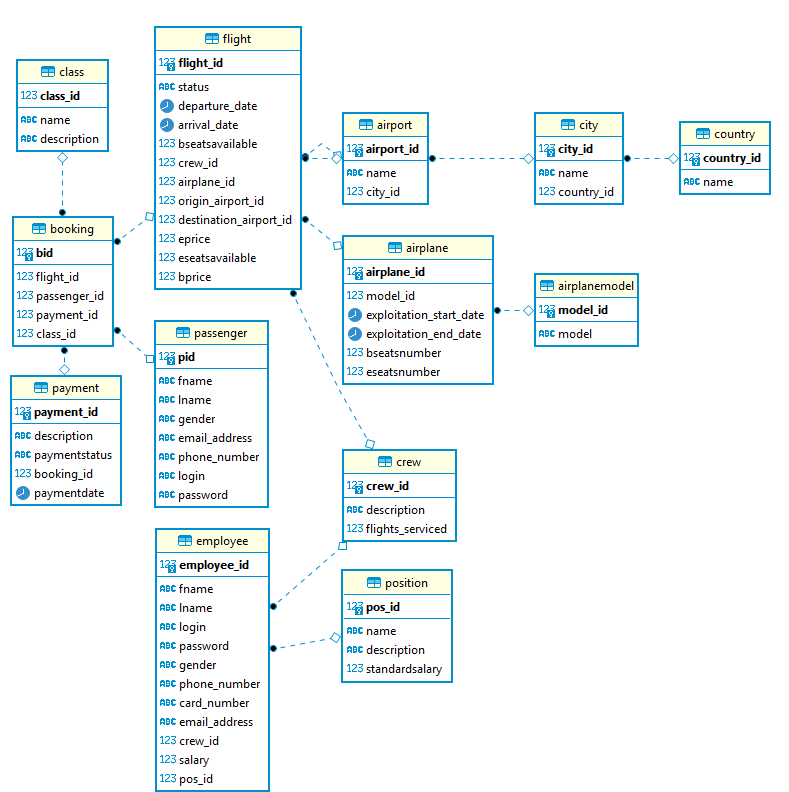
1. Авторизация пользователя
2. Регистрация нового пользователя
3. Вывод доступных перелётов за выбранный пользователем период времени
4. Первоначальное бронирование билета
5. Вывод на экран перелётов, который уже бронировал пользователь, вошедший в систему
6. Возможность отмены бронирования билета на выбранный пользователем рейс
7. Регулирование уровня доступа пользователя в зависимости от его статуса (клиент/сотрудник компании)
8. Добавление информации о новых рейсах: выбор места прибытия и отбытия из списка имеющегося списка, времени вылета и прибытия, цены билетов на места в зависимости от класса, а также выбор команды, которая будет обслуживать данный рейс (только для аккаунта сотрудника)
9. Обновление информации о существующих рейсах – также только для аккаунта сотрудника

Таким образом, на данный момент мы рассмотрели предметную область, построили инфологическую и даталогическую модели, подробно описали сущности, их атрибуты и связи. После этого мы определили технические и функциональные требования к приложению.

# Глава 3. Реализация

## 3.1 Физическая схема

Реализация базы данных начинается с построения физической схемы в СУБД.

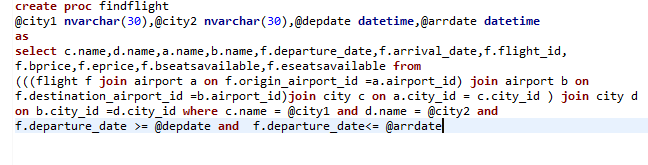


*Рисунок 3 – физическая модель*

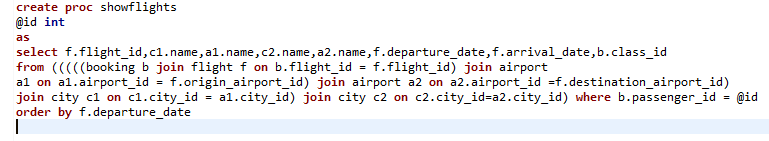
## 3.2 Хранимые процедуры

Данные процедуры используются во время работы программы. Вся информация вводится или выбирается непосредственно самим пользователем.

1. Процедура для поиска рейсов в БД

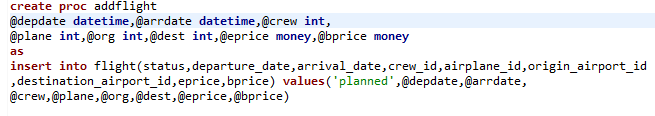
*Рисунок 4 – процедура 1*

1. Процедура для поиска рейсов, на которые зарегистрирован пользователь



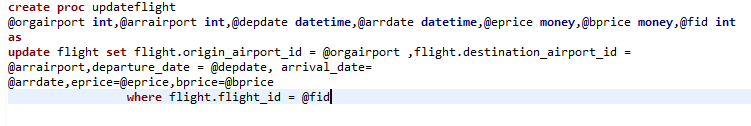
*Рисунок 5 – процедура 2*

1. Процедура для добавления в БД новых рейсов сотрудником компании



*Рисунок 6 – процедура 3*

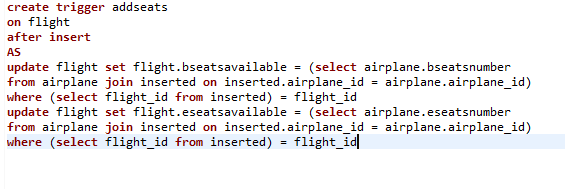
1. Процедура для обновления части информации об уже существующих рейсах



*Рисунок 7 – процедура 4*

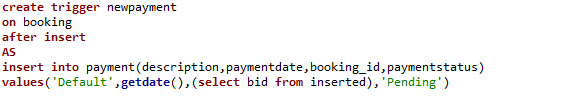
## 3.3 Триггеры

1. Триггер, добавляющий информацию о количестве мест на добавленный рейс в зависимости от выбранного самолёта.



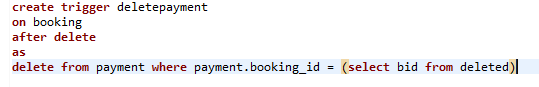
*Рисунок 8 – триггер 1*

1. Триггер, добавляющий информацию о новом платеже.



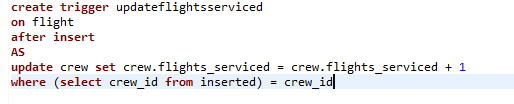
*Рисунок 9 – триггер 2*

1. Триггер, удаляющий информацию о платеже в случае отмены бронирования.



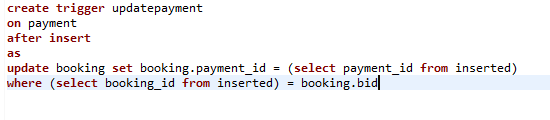
*Рисунок 10 – триггер 3*

1. Триггер, обновляющий информацию о количестве обслуженных рейсов определенной командой.



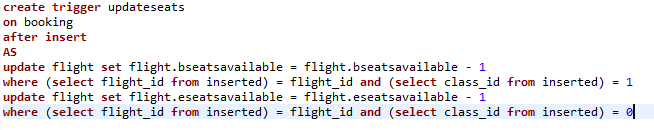
*Рисунок 11 – триггер 4*

1. Триггер, связывающий определенное бронирование с соответствующим платежом.



*Рисунок 12 – триггер 5*

1. Триггер, необходимый для обновления информации о количестве мест на определенном рейсе после очередного бронирования.

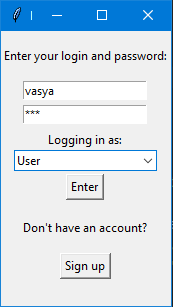


*Рисунок 13 – триггер 6*

## 3.4 Графический интерфейс

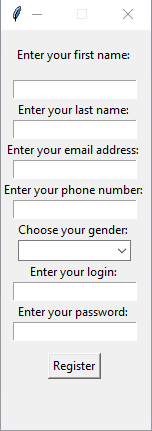
Интерфейс реализован на языке Python с использованием библиотеки Tkinter. Для подключения к базе данных требуется модуль Pyodbc, а также Microsoft ODBC Driver for SQL Server, предназначенный для использования всех возможных функций базы данных.

Для использования основного функционала программы необходимо пройти этап авторизации, во время которого проверяется легитимность вводимых данных, а также определяются доступные функции программы.



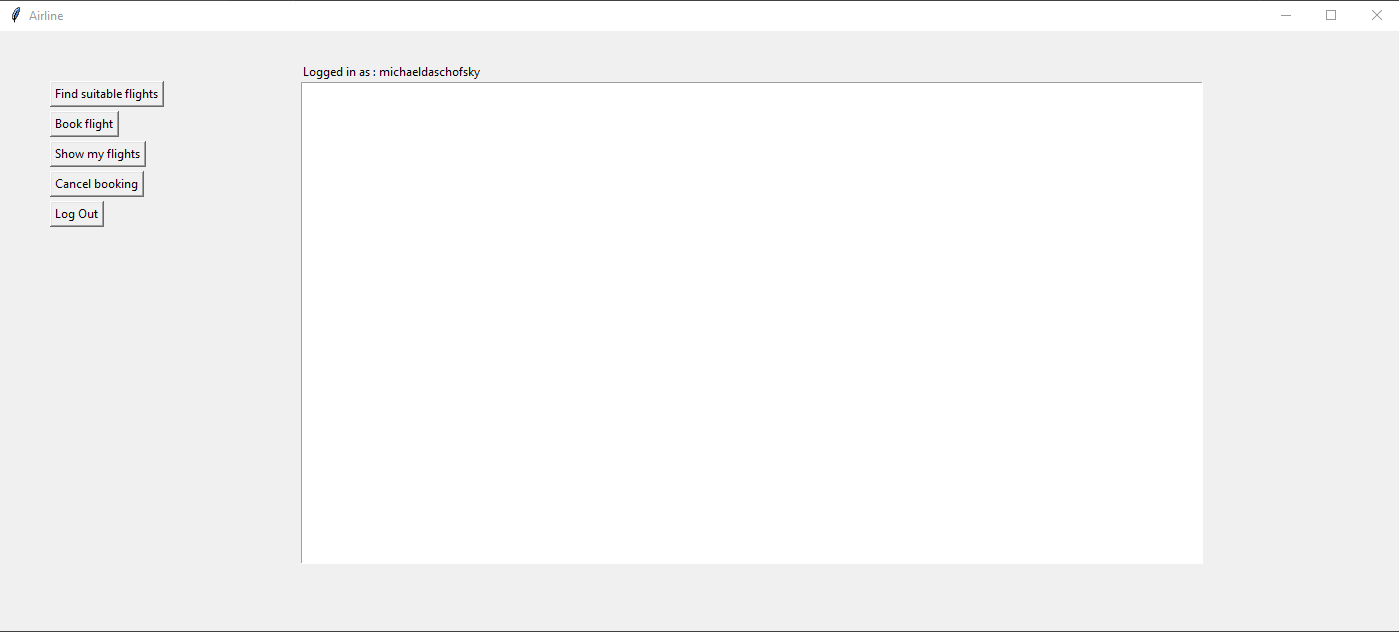
*Рисунок 14 – окно входа*

В случае, если у пользователя нет аккаунта, открывается окно регистрации, в котором клиент вводит некоторые данные, и, после базовой проверки на корректность введённой информации, получает полностью рабочую учетную запись, все данные о которой хранятся в БД.



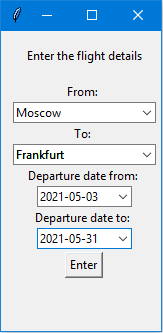
*Рисунок 15 – окно регистрации*

После успешного входа пользователь получает доступ к основному окну, где видны все доступные ему функции.



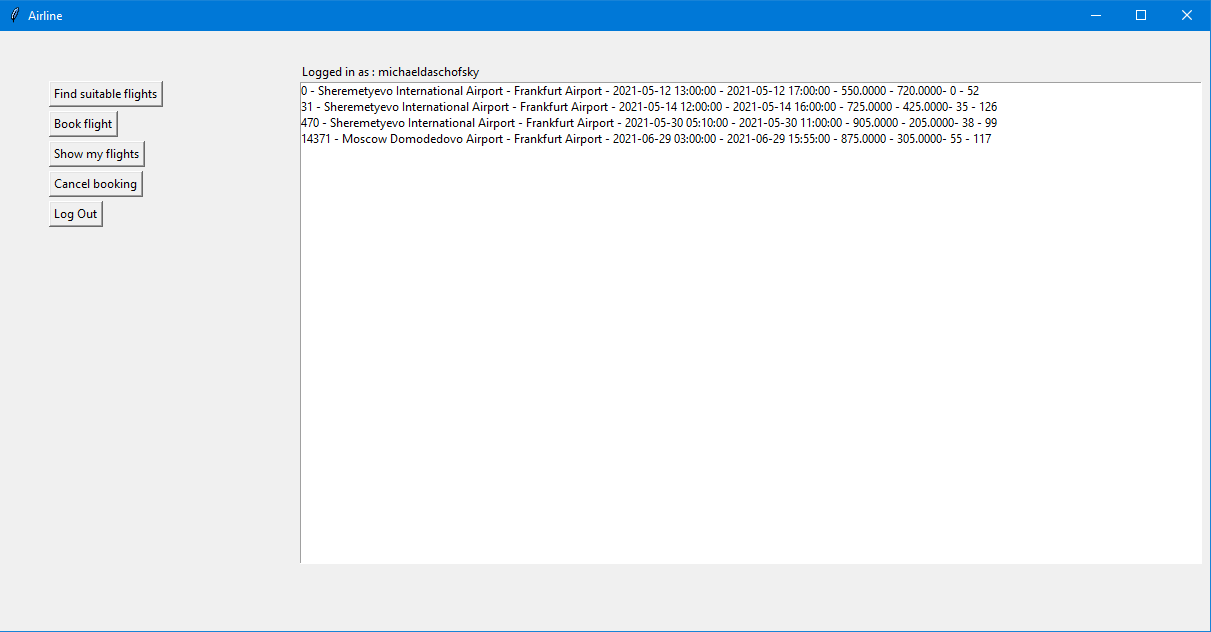
*Рисунок 16 – основное окно*

Основная цель приложения с точки зрения пользователя – поиск подходящих рейсов.



*Рисунок 17 – поиск рейсов*

Итогом выполнения становится обновление информации в основном окне, в котором появляются данные о подходящих рейсах.



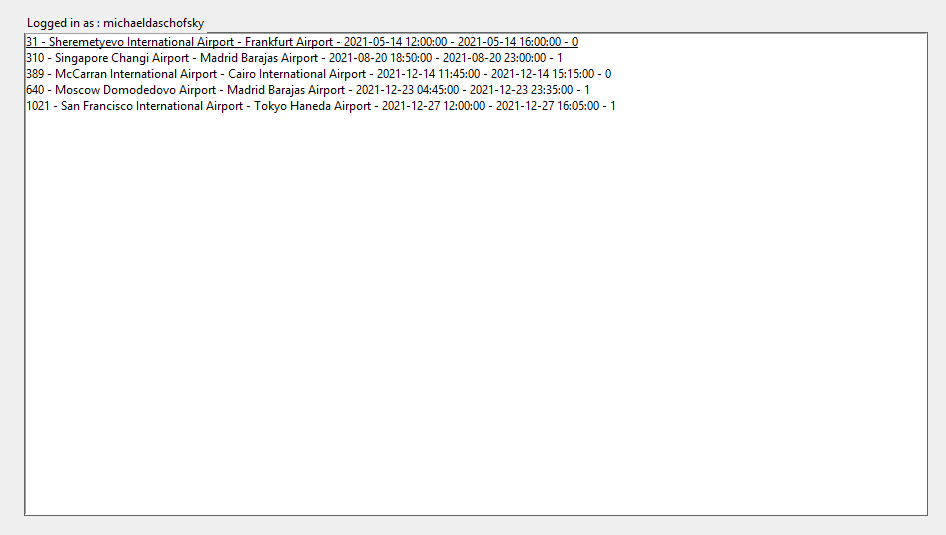
*Рисунок 18 – вывод информации*

После выбора какого-либо рейса, клиент может забронировать билет на него, при условии, что имеются места того класса, который его интересует. В случае, если на выбранном рейсе нет свободных мест, выводится сообщение о невозможности бронирования билета. Если же места есть, то пользователь успешно покупает билет, а специальный триггер обновляет количество мест на борту. При успешной покупке билета срабатывает ещё один триггер, который создаёт новые данные в сущности, ответственной за хранение информации о всех платежах.



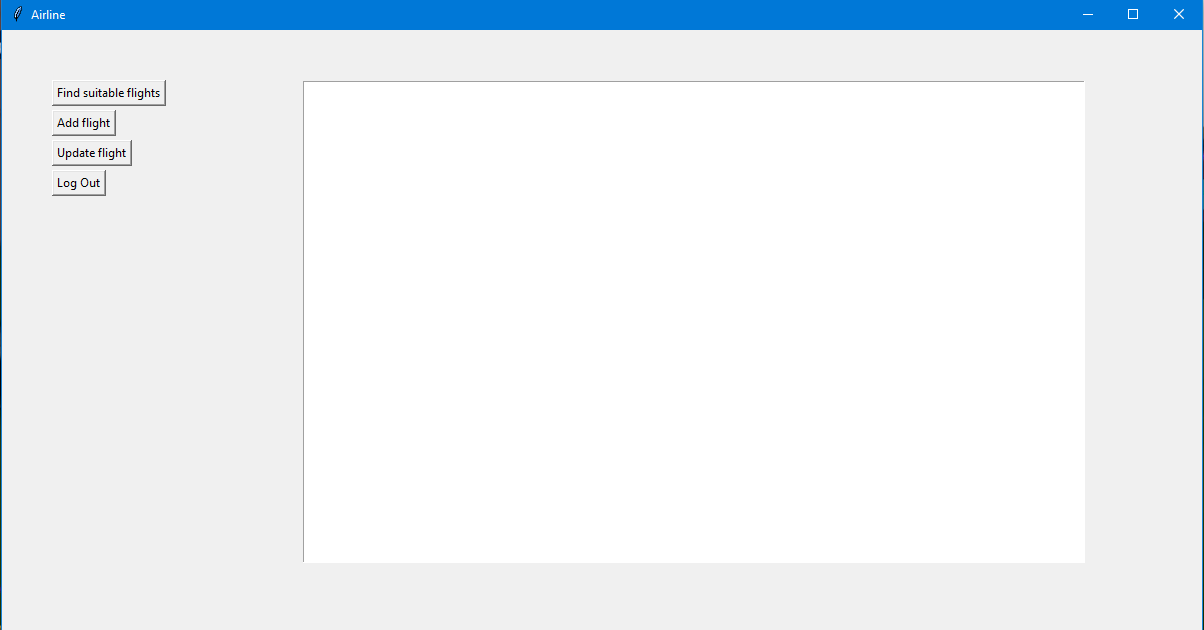
*Рисунок 19 – бронирование билета*

Последующий вызов функции отображения полётов покажет все рейсы, на которые зарегистрирован пользователь, отсортированные по дате вылета.



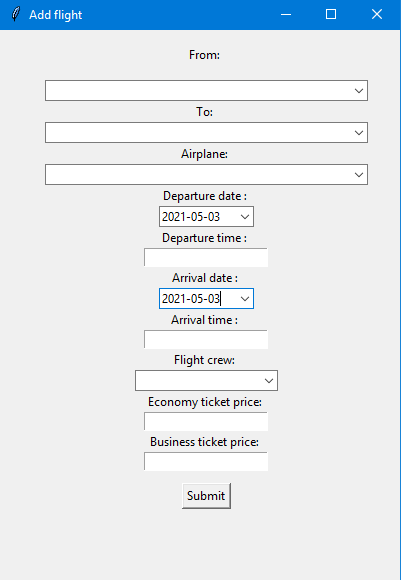
*Рисунок 20 – обновление полетов пользователя*

Интерфейс программы для сотрудника компании похож на обычный, но имеет несколько другие функции.



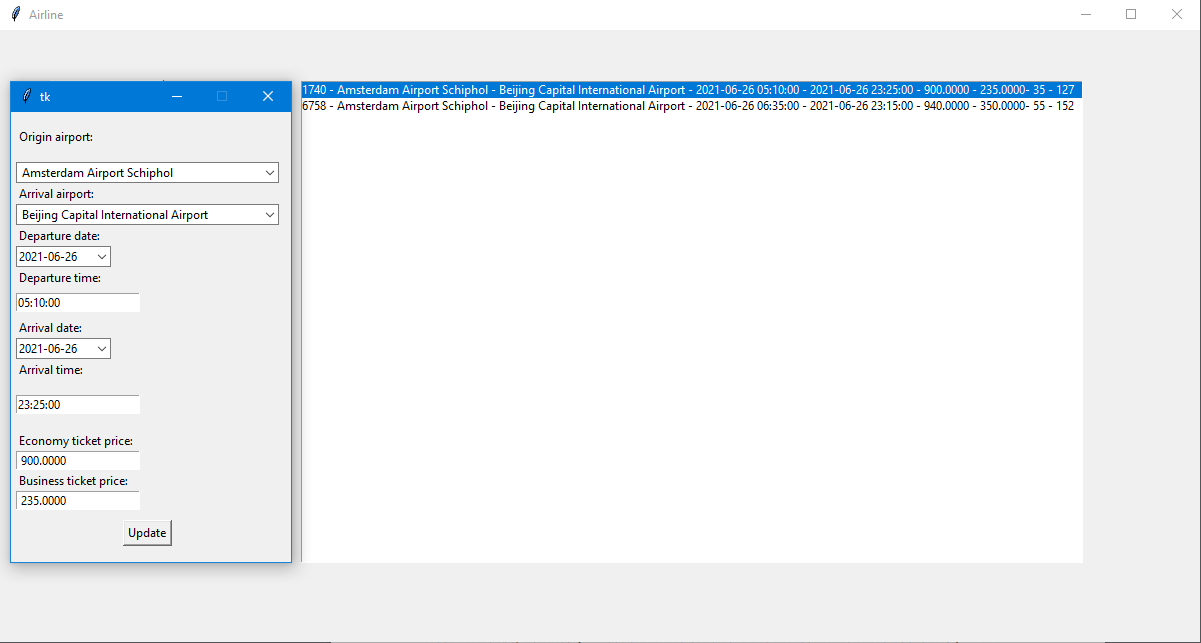
*Рисунок 21 – интерфейс работника компании*

Основной функцией сотрудника, работающим с приложением, является добавление и обновление информации о различных рейсах. Дополнение БД новыми рейсами требует введения определённого количества информации: время отправления и прибытия, используемые аэропорты, команда, ответственная за полёт, а также стоимость билетов различных классов. При создании нового рейса срабатывает триггер, который задаёт определённое количество мест на борту для данного полёта в зависимости от выбранного воздушного судна.



*Рисунок 22 – добавление нового рейса*

Другой важной функцией является изменение информации о каком-либо рейсе, нуждающемся в обновлении. Начальные данные в окне заполняются автоматически, а любые изменения сразу также имеют эффект на саму базу данных.



*Рисунок 23 – обновление информации о рейсе*

# Заключение

Данную работу мы начали с теории проектирования и разработки базы данных. Мы обсудили теоретические понятия, которыми апеллируют при проектировании, а также рассмотрели основные этапы проектирования и пользу приведения данных к третьей нормальной форме. В качестве СУБД мы выбрали DBeaver, а для реализации графического интерфейса отдали предпочтение инструменту Tkinter. После этого мы описали предметную область авиакомпании, создали инфологическую, даталогическую и физическую модели, сформировали требования к приложению. За время работы над моделями мы осознали, что достаточно непросто, но в то же время крайне важно правильно отобразить сущности и связи между ними, так как база данных – это фундамент, на основе которого будет строится приложение.

Разработка приложения включила в себя окно для регистрации и авторизации, страницу для поиска, бронирования, отмены и отображения существующих бронирований на рейсы. Для этого нам понадобилось написать различные хранимые процедуры и триггеры, которые позволили автоматизировать некоторые задачи. Например, были созданы триггеры, добавляющие информацию о количестве мест на рейс и о новых платежах.

Таким образом, выбранная тема помогла не только изучить авиакомпании и лучше понять их процессы и задачи, но и создать ИТ-продукт с нуля, увидев картину разработки в целом.

# Список литературы

Алан, Б. (2016). Изучаем SQL. СПб., М.:«Символплюс.

Дейт, К. Д. (2005). Введение в системы баз данных, 8-е издание. М.: Издательский дом" Вильямс.с.

Лутц, М. (2011). Программирование на Python, том II, 4-е издание. Пер. с англ.–СПб.: Символ-Плюс.

Hughes, P. (2000). Python and tkinter programming. Linux Journal, 2000(77es), 23-es.

# Приложение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сущность** | **Атрибут** | **Тип** | **PK/FK/O** | **Описание** |
| booking | bid | int | Primary Key | Идентификатор бронирования |
| flight\_id | int | Foreign Key | Идентификатор полёта |
| passenger\_id | int | Foreign Key | Идентификатор пассажира |
| payment\_id | int | Foreign Key | Идентификатор платежа |
| class\_id | int | Foreign Key | Идентификатор класса |
| class | class\_id | int | Primary Key | Идентификатор класса |
| name | nvarchar(20) |  | Название класса (эконом/бизнес) |
| description | nvarchar(100) |  | Информация о классе |
| payment | payment\_id | int | Primary Key | Идентификатор платежа |
| description | nvarchar(100) |  | Информация о платеже |
| paymentdate | datetime |  | Дата платежа |
| paymentstatus | nvarchar(30) |  | Статус платежа |
| booking\_id | int | Foreign Key | Идентификатор бронирования |
| passenger | pid | int | Primary Key | Идентификатор пассажира |
| fname | nvarchar(20) |  | Имя пассажира |
| lname | nvarchar(20) |  | Фамилия пассажира |
| gender | nvarchar(10) |  | Пол пассажира |
| email\_address | nvarchar(30) |  | Электронная почта пассажира |
| phone\_number | nvarchar(20) |  | Номер телефона пассажира |
| login | nvarchar(30) |  | Логин для авторизации пассажира |
| password | nvarchar(30) |  | Пароль для авторизации пассажира |
| flight | flight\_id | int | Primary Key | Идентификатор полёта |
| status | nvarchar(20) |  | Статус полёта |
| departure\_date | datetime |  | Дата и время отправления |
| arrival\_date | datetime |  | Дата и время прибытия |
| bseatsavailable | int |  | Количество свободных мест бизнес-класса |
| crew\_id | int | Foreign Key | Идентификатор экипажа |
| airplane\_id | int | Foreign Key | Идентификатор самолёта |
| origin\_airport\_id | int | Foreign Key | Идентификатор аэропорта отправления |
| destination\_airport\_id | int | Foreign Key | Идентификатор аэропорта прибытия |
| eprice | money |  | Стоимость перелёта экономическим классом |
| eseatsavailable | int |  | Количество свободных мест экономического класса |
| bprice | money |  | Стоимость перелёта бизнес-классом |
| airplane | airplane\_id | int | Primary Key | Идентификатор самолёта |
| model\_id | int | Foreign Key | Идентификатор модели самолёта |
| exploitation\_start\_date | date |  | Дата начала эксплуатации самолёта |
| exploitation\_end\_date | date |  | Дата конца эксплуатации самолёта |
| bseatsnumber | int |  | Число мест бизнес-класса в самолёте |
| eseatsnumber | int |  | Число мест экономического класса в самолёте |
| airplanemodel | model\_id | int | Primary Key | Идентификатор модели самолёта |
| model | nvarchar(20) |  | Модель самолёта |
| airport | airport\_id | int | Primary Key | Идентификатор аэропорта |
| name |  |  | Название аэропорта |
| city\_id | int | Foreign Key | Идентификатор города, в котором находится аэропорт |
| city | city\_id | int | Primary Key | Идентификатор города |
| name | nvarchar(20) |  | Наименование города |
| country\_id | int | Foreign Key | Идентификатор страны, в которой находится город |
| country | country\_id | int | Primary Key | Идентификатор страны |
| name | nvarchar(20) |  | Наименование страны |
| crew | crew\_id | int | Primary Key | Идентификатор экипажа |
| description | nvarchar(100) | Optional | Информация об экипаже |
| flights\_serviced | int |  | Количество совершённых перелётов экипажем |
| employee | employee\_id | int | Primary Key | Идентификатор сотрудника |
| fname | nvarchar(20) |  | Имя сотрудника |
| lname | nvarchar(20) |  | Фамилия сотрудника |
| login | nvarchar(30) |  | Логин для авторизации сотрудника |
| password | nvarchar(30) |  | Пароль для авторизации сотрудника |
| gender | nvarchar(10) |  | Пол сотрудника |
| phone\_number | nvarchar(20) |  | Номер телефона сотрудника |
| card\_number | nvarchar(20) |  | Номер карты сотрудника |
| email\_address | nvarchar(20) |  | Электронная почта сотрудника |
| position | nvarchar(20) |  | Должность сотрудника |
| crew\_id | int | Foreign Key, Optional | Идентификатор экипажа, в котором работает сотрудник |
| salary | money |  | Заработная плата сотрудника |
| position | name | nvarchar(20) | Primary Key | Наименование должности |
| description | nvarchar(100) |  | Информация о должности |
| standardsalary | money |  | Стандартная заработная плата на данной должности |

*Таблица 2. Сущности и их атрибуты*