МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
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.

**КУРСОВАЯ РАБОТА**

по дисциплине «Базы данных»

на тему: «Разработка и реализация базы данных для онлайн магазина продажи авиабилетов»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шорин В.Д.

Шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) Промышленная разработка программного обеспечения

Группа 71-ПГ

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рыженков Д.В.

Оценка: «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_» Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Орел 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

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 курсовую работу**

по дисциплине «Базы данных»

Студент Круглов И.Н. Шифр 190953

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Группа 91ИВТ

1 Тема курсовой работы

«Разработка и реализация базы данных для онлайн магазина продажи авиабилетов»

2 Срок сдачи студентом законченной работы «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 202\_\_

3 Исходные данные

Предметной областью курсовой работы является онлайн-магазин продажи авиабилетов.

4 Содержание курсовой работы

Анализ предметной области онлайн-магазина продажи авиабилетов;

Проектирование схем базы данных;

Реализация базы данных.

5 Отчетный материал курсовой работы

Пояснительная записка курсовой работы, SQL скрипты для создания и работы с базой данных.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рыженков Д.В.

Задание принял к исполнению: «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 202\_\_

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

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc74404852)

[1 Описание предметной области онлайн магазина продажи авиабилетов 5](#_Toc74404853)

[2 Проектирование схем базы данных для онлайн магазина продажи авиабилетов 6](#_Toc74404854)

[2.1 Проектирование концептуальной схемы 6](#_Toc74404855)

[2.2 Проектирование логической схемы базы данных 9](#_Toc74404856)

[2.3 Построение физической схемы базы данных для онлайн магазина продажи авиабилетов 12](#_Toc74404857)

[3 Реализация базы данных для онлайн магазина продажи авиабилетов 15](#_Toc74404858)

[3.1 Создание базы данных 15](#_Toc74404859)

[3.2 Разработка триггеров к базе данных 17](#_Toc74404860)

[3.3 Заполнение базы данных 19](#_Toc74404861)

[3.4 Разработка запросов базы данных 20](#_Toc74404862)

[3.5 Разработка хранимых процедур для базы данных для онлайн магазина продажи авиабилетов 21](#_Toc74404863)

[Заключение 24](#_Toc74404864)

[Список литературы 25](#_Toc74404865)

[Приложение А 26](#_Toc74404866)

[Листинг базы данных 26](#_Toc74404867)

# Введение

Одной из наиболее актуальных задача в сфере IT в настоящее время является разработка баз данных.

Целью курсовой работы является построение базы данных для онлайн магазина продажи авиабилетов.

Задачами курсовой работы являются:

1. Изучение предметной области процесса продажи авиабилетов в онлайн магазине;
2. Разработка модели базы данных для онлайн магазина продажи авиабилетов;
3. Выбор системы управления базой данных (СУБД);
4. Создание и пробное наполнение базы данных для онлайн магазина продажи авиабилетов;
5. Разработка триггеров к базе данных для онлайн магазина продажи авиабилетов;
6. Разработка процедур для базы данных для онлайн магазина продажи авиабилетов;
7. Построение запросов на извлечение информации из базы данных для онлайн магазина продажи авиабилетов.

# Описание предметной области онлайн магазина продажи авиабилетов

Онлайн магазин — сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или мобильном устройстве, сформировать заказ на покупку, выбрать способ оплаты и получения заказа. В данной работе будет рассмотрен онлайн магазина продажи авиабилетов. Онлайн магазин является своего рода витриной, на которой в качестве товара выставляются авиабилеты. Авиабилеты делятся, в основном, по классу.

Страница заказа определенного авиабилета содержит следующую информацию о нем: компания, дата и время отправления, дата и время прибытия, место отправления, место прибытия, цена, максимальный вес багажа.

Просто просматривать сайт, искать авиабилеты и просматривать о них информацию может каждый незарегистрированный пользователь. Однако, если он захочет заказать какой-либо конкретный билет, то ему придется зарегистрироваться, указав следующие свои данные: электронную почту, фамилию, имя, отчество, дату рождения, город проживания, тип документа и его номер, а также один номер мобильного телефона для обратной связи.

Когда зарегистрированный пользователь заполнил всю необходимую информацию касательно билета, он переходит на страницу подтверждения заказа для оплаты товара. В заказе о каждом авиабилете хранится вся его информация. Как только пользователь оплатил заказ, ему на указанную почту высылается штрих-код, по которому, в последствие, он может уже в аэропорту подтвердить регистрацию на полет.

# Проектирование схем базы данных для онлайн магазина продажи авиабилетов

## Проектирование концептуальной схемы

Первым этапом проектирования базы данных после описания предметной области является создание концептуальной схемы базы данных.

Концептуальная модель базы данных – это наглядная диаграмма, нарисованная в принятых обозначениях и показывающая основные связи между объектами и их характеристиками. Создается концептуальная модель для дальнейшего проектирования базы данных и перевод ее, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

Для единообразия программирования баз данных введены следующие понятия для концептуальных баз данных:

1. Объект или сущность. Это фактическая вещь или объект (для людей) за которой пользователь (заказчик) хочет наблюдать. Например, Иванов Иван Иванович.
2. Атрибут – это характеристика объекта, соответствующая его сущности. Например. Задаем себе вопрос: Какую информацию нужно хранить об Иванове Иване Ивановиче? Ответы на этот вопрос и будут атрибуты объекта Иванов Иван Иванович.
3. Третье понятие в проектировании концептуальной базы данных это связь или отношения между объектами [1].

Диаграмма сущность/отношения (объект/связь) называют ER-диаграммой или EDR (entity-relationship diagram). Сама модель сущность-связь была предложена профессором Peter Pin-Shen Chen (Питер Чен) в 1976 году. Правила написания и условные обозначения ER-диаграммы называют нотацией. Распространены две основные нотации ER-диаграмм:

* Нотация Питера Чена.
* Нотация Гордона Эверста под названием Crow’s Foot или Fork (вилка).

Чен предложил и это приняли следующие условные обозначения для ER-диаграмм:

* Сущность или объект обозначать прямоугольником.
* Отношения обозначать ромбом.
* Атрибуты объектов, обозначаются овалом.
* Если сущность связана с отношением, то их связь обозначается прямой линией со стрелкой. Необязательная связь обозначается пунктирной линией. Мощная связь обозначается двойной линией.

Каждый атрибут может быть связан с одним объектом (сущностью).

Для создания концептуальной схемы базы данных воспользуемся нотацией Чена. Перед проектированием необходимо выделить главные сущности будущей схемы.

Изучив предметную область, можно выделить следующие основные объекты: «Билет» и «Пассажир».

На основе выделенных объектов, можно выделить следующие сущности; билет, компания, класс, пассажир, документ.

Рассмотрим отношения между ними:

* В рейсе содержится информация об обслуживающей компании, типе самолета и билетах, поэтому он включается в информацию о пассажире;
* В билете содержится информация о классе полета и пассажире, поэтому они включаются в содержание билета;
* У пассажира имеется какой-либо документ, подтверждающий личность, поэтому он включается в информацию о пассажире.

На рис. 2.1 представлена концептуальная схема базы данных, построенная на основе анализа предметной области.

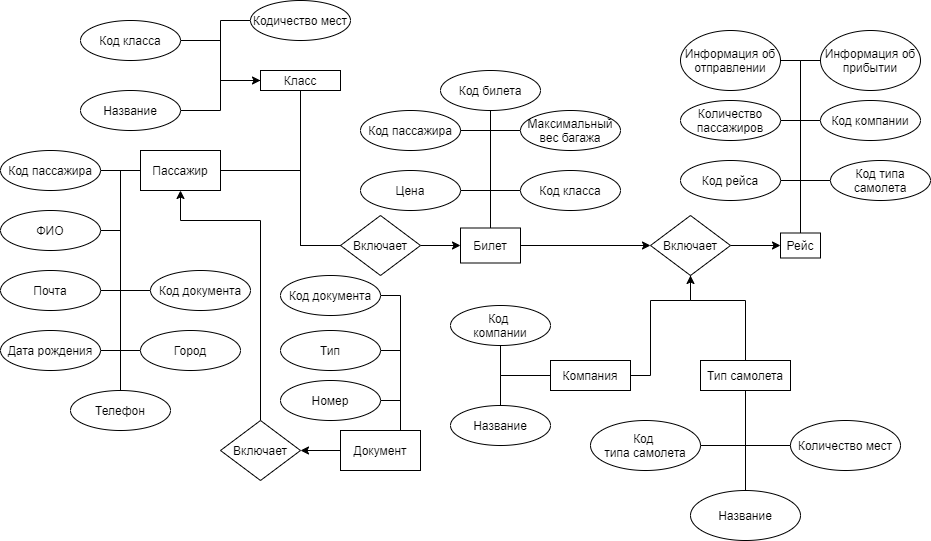


Рисунок 2.1 – Концептуальная схема базы данных

Теперь приступим к изучению атрибутов каждой из сущностей.

Сущность «Билет» имеет следующие атрибуты:

* Код билета;
* Цена;
* Максимальный вес багажа;
* Код класса;
* Код пассажира.

Сущность «Компания» имеет следующие атрибуты:

* Код компании;
* Название.

Сущность «Класс» имеет следующие атрибуты:

* Код класса;
* Название.

Сущность «Пассажир» имеет следующие атрибуты:

* Код пассажира;
* Фамилия;
* Имя;
* Отчество;
* Почта;
* Дата рождения;
* Код документа;
* Город;
* Телефон.

Сущность «Документ» имеет следующие атрибуты:

* Код документа;
* Тип;
* Номер.

Сущность «Рейс» имеет следующие атрибуты:

* Информация об отправлении;
* Информация о прибытии;
* Код компании;
* Код рейса;
* Количество пассажиров;
* Код типа самолета.

Сущность «Тип самолета» имеет следующие атрибуты:

* Код типа самолета;
* Количество мест;
* Название.

В сущности «Тип самолета» атрибут название указывает на категорию самолета, к которой он относится. Она определяет количество мест. Например, для самолета «Туполев Ту-134» оно равно «80», а для «Туполев Ту-154» составляет 150.

## Проектирование логической схемы базы данных

Логическая схема – это универсальное описание структуры данных, независимое от конечной реализации базы данных и аппаратной платформы. На основании полученной логической схемы переходят к физической схеме данных.

Проведем повторный анализ предметной области, учитывая полученную в п. 2.1 концептуальную схему, в контексте логической схемы базы данных и получим более подробное описание связей между сущностями, а также выделим у каждой из них первичные ключи.

Полученная логическая схема базы данных представлена на рис. 2.2.

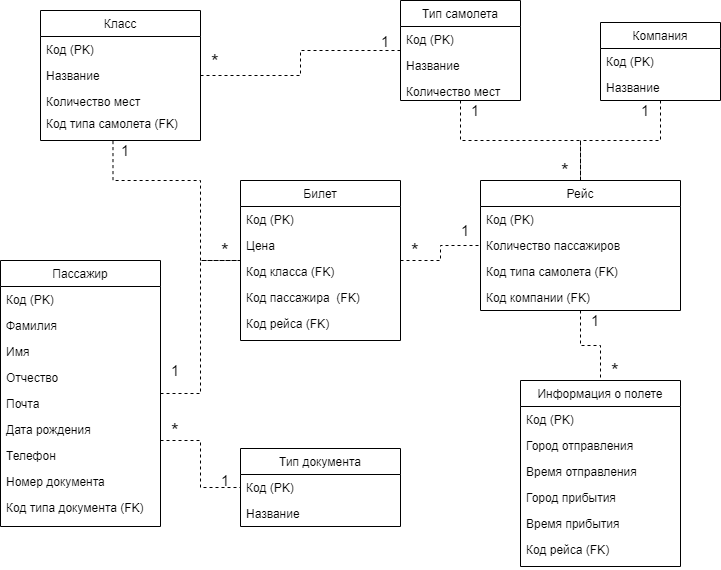


Рисунок 2.2 -Логическая схема базы данных

Сначала рассмотрим подробнее связи между данными сущностями.

Одна компания может обслуживать неограниченное количество рейсов, но рейс всегда связан только с одной компанией, потому между ними отношение «один ко многим».

Конкретный рейс может быть выполнен только на определенном типе самолета, но этот самолет может обслуживать различные рейсы, потому между ними отношение «один ко многим».

В одном самолете может быть несколько классов комфорта с определенным количеством мест, но такой класс комфорта всегда привязан только к конкретному типу самолета, потому между ними отношение «один ко многим».

Рейс может быть как прямым (из одного города в другой), так и с пересадками. Значит, у него может быть несколько информаций о полете, но такая информация уникальна для каждого рейса, потому между ними отношение «один ко многим».

К одному рейсу привязаны множество билетов, но один билет привязан только к одному рейсу, потому между ними отношение «один ко многим».

Билет может принадлежать только к одному классу комфорта в самолете, но класс комфорта самолета может относится к разным билетам, потому между ними отношение «один ко многим».

У пассажира может быть неограниченное количество билетов, но один билет может быть связан только с одним пассажиром, потому между ними отношение «один ко многим».

Пассажир может иметь несколько типов документов, удостоверяющих его личность, но один тип документ всегда связан и принадлежит только одному пассажиру, потому между ними отношение «один ко многим».

Сущность «Рейс» имеет первичный ключ «Код» и внешние ключи «Код компании» и «Код типа самолета», которые указывают на связь с сущностями «Компания» и «Тип самолета» соответственно.

Сущность «Информация о полете» имеет первичный ключ «Код» и внешний ключ «Код рейса», указывающий на связь с соответствующей таблицей.

Сущность «Компания» имеет первичный ключ «Код».

Сущность «Тип самолета» имеет первичный ключ «Код».

Сущность «Билет» имеет первичный ключ «Код» и внешние ключи «Код класса», «Код пассажира» и «Код рейса», которые указывают на связь данной сущности с сущностями «Класс», «Пассажир» и «Рейс» соответственно.

Сущность «Класс» имеет первичный ключ «Код» и внешний ключ «Код типа самолета», указывающий на связь с таблицей «Тип самолета».

Сущность «Пассажир» имеет первичный ключ «Код» и внешний ключ «Код типа документа», который указывает на связь с сущностью «Тип документа» соответственно.

Сущность «Документ» имеет первичный ключ «Код».

Проверим полученную логическую схему на соответствие первым трем нормальным формам [5].

**Первая нормальная форма** (1НФ) говорит, что каждый атрибут отношения должен хранить атомарное значение, каждое отношение (строка в таблице) должно содержать одинаковое количество атрибутов (столбцов), т.е.:

* запрещает повторяющиеся столбцы (содержащие одинаковую по смыслу информацию)
* запрещает множественные столбцы (содержащие значения типа списка и т.п.)
* требует определить первичный ключ для таблицы, то есть тот столбец или комбинацию столбцов, которые однозначно определяют каждую строку.

**Вторая нормальная форма** (2НФ) говорит, что отношение находится во второй нормальной форме, если оно находится в 1НФ, и при этом все не ключевые атрибуты зависят только от первичного ключа, т.е.:

* Вторая нормальная форма требует, чтобы не ключевые столбцы таблиц зависели от первичного ключа в целом, но не от его части.
* Если таблица находится в первой нормальной форме и первичный ключ у нее состоит из одного столбца, то она автоматически находится и во второй нормальной форме [2].

Отношение находится в **третьей нормальной форме** (3НФ), если оно находится во второй нормальной форме и каждый не ключевой атрибут зависит только от первичного ключа и не зависят друг от друга [4].

Поскольку все вышеперечисленные условия соблюдены, то мы можем сделать вывод о том, что полученная логическая схема базы данных соответствует первым трем нормальным формам и мы можем перейти к построению физической схемы и реализации базы данных. Поэтому в следующем пункте будет рассмотрен процесс создания физической схемы базы данных, в которой мы должны выбрать СУБД, в контексте которой она будет реализовываться, а также определить типы полей.

## Построение физической схемы базы данных для онлайн магазина продажи авиабилетов

В качестве СУБД, для которой будет строиться физическая схема базы данных выберем MS SQL Server.

MS SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

В контексте выбранной СУБД физическая схема нашей базы данных будет выглядеть так, как показано на рисунке 2.3.



Рисунок 2.3 – Физическая схема базы данных

Целостность базы данных – соответствие имеющейся в базе данных информации предметной области.

В реляционной модели данных определяют две базовые категории обеспечения целостности:

* Целостность ссылок.
* Целостность сущностей.

**Целостность ссылок** — необходимое качество реляционной базы данных, заключающееся в отсутствии в любом её отношении внешних ключей, ссылающихся на несуществующие кортежи.

**Целостность сущностей** — каждый кортеж любого отношения должен отличатся от любого другого кортежа этого отношения (т.е. любое отношение должно обладать первичным ключом).

Учитывая то, что наша схема удовлетворяет первым трем нормальным формам, можно сделать вывод, что целостность данных соблюдена.

Бизнес-правила определяют реакцию системы на добавление, изменение или удаление данных, обеспечивая непротиворечивость и ссылочную целостность БД. Многие бизнес-правила реализуются с помощью триггеров. В нашей базе данных необходимо будет реализовать следующие бизнес-правила:

* дата рождения пассажира не может быть больше даты отправки самолета;
* количество билетов не может превышать заявленное количество мест в самолете.

# Реализация базы данных для онлайн магазина продажи авиабилетов

## Создание базы данных

Для реализации базы данных будут использованы следующие языки:

DDL - (Data Definition Language) предложения для определения структуры базы данных или схемы.

DML - (Data Manipulation Language) предложения для управления данными.

К DDL относятся такие команды, как:

* CREATE - создает объекты базы данных (таблицы, представления и т.д.);
* ALTER - изменяет структуру и объекты базы данных;
* DROP - удаляет объекты базы данных;
* PRIMARY KEY – назначает столбец первичным ключом;
* IDENTITY – назначает столбец идентификатором;
* [FOREIGN KEY] REFERENCES главная\_таблица (столбец\_главной\_таблицы) – устанавливает внешние ключи для таблицы;
* TRUNCATE - удаляет все записи из таблицы;
* SP\_RENAME - переименовывает объект (sp\_rename 'Users', 'UserAccounts';).

К DML относятся такие команды, как:

* SELECT - возвращает данные из базы данных;
* INSERT - вставляет данные в таблицу;
* UPDATE - обновляет существующие данные в таблице;
* DELETE - удаляет все записи в таблице;
* WHERE – применяется для фильтрации данных в команде SELECT

Рассмотрим некоторые скрипты реализации базы данных для онлайн магазина продажи авиабилетов

На рис. 3.1 и 3.2 представлены скрипты для создания таблиц «Информация о полете» и «Пассажир» с помощью команды INSERT. Скрипты по созданию остальных таблиц представлены в Приложении А.

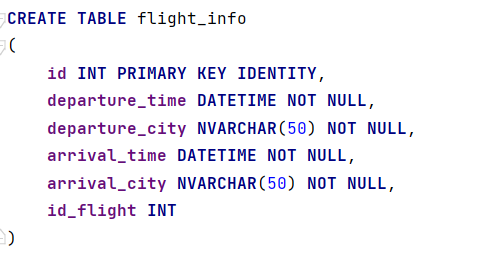


Рисунок 3.1 – Скрипт создания таблицы «Информация о полете»

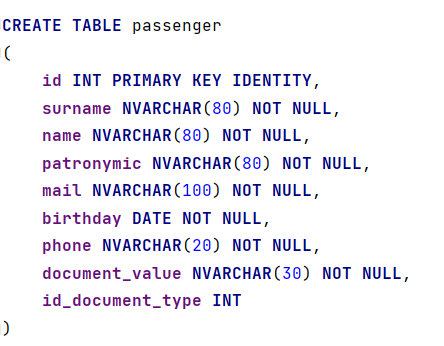


Рисунок 3.2 – Скрипт создания таблицы «Пассажир»

Для формирования внешних ключей в таблицах будем использовать команду ALTER TABLE [3]. На рис. 3.3 и .3.4 приведены скрипты по созданию внешних ключей для таблиц «Рейс» и «Билет». Скрипты по изменению остальных таблиц представлены в Приложении А.

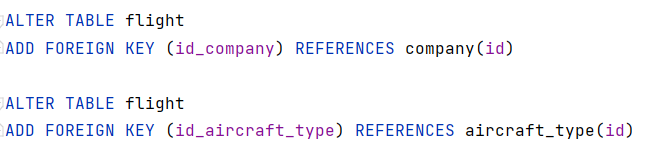


Рисунок 3.3 – Скрипт создания внешних ключей в таблице «Рейс»

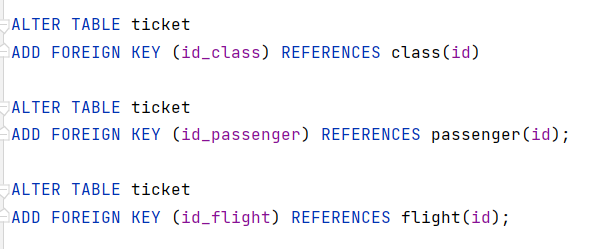


Рисунок 3.4 – Скрипт создания внешних ключей в таблице «Билет»

После создания базы данных для онлайн магазина продажи авиабилетов можно приступить к разработке триггеров, которые будут реализовывать бизнес-правила, определенные в п. 2.3.

## Разработка триггеров к базе данных

Триггеры представляют специальный тип хранимой процедуры, которая вызывается автоматически при выполнении определенного действия над таблицей или представлением, в частности, при добавлении, изменении или удалении данных, то есть при выполнении команд INSERT, UPDATE, DELETE.

Формальное определение триггера:

CREATE TRIGGER имя\_триггера

ON {имя\_таблицы | имя\_представления}

{AFTER | INSTEAD OF} [INSERT | UPDATE | DELETE]

AS выражения\_sql

Для создания триггера применяется выражение CREATE TRIGGER, после которого идет имя триггера. Как правило, имя триггера отражает тип операций и имя таблицы, над которой производится операция.

Каждый триггер ассоциируется с определенной таблицей или представлением, имя которых указывается после слова ON.

Затем устанавливается тип триггера. Мы можем использовать один из двух типов:

* AFTER: выполняется после выполнения действия.
* INSTEAD OF: выполняется вместо действия (то есть, по сути, действие - добавление, изменение или удаление - вообще не выполняется).

После типа триггера идет указание операции, для которой определяется триггер: INSERT, UPDATE или DELETE.

Для триггера AFTER можно применять сразу для нескольких действий, например, UPDATE и INSERT. В этом случае операции указываются через запятую. Для триггера INSTEAD OF можно определить только одно действие.

И затем после слова AS идет набор выражений SQL, которые, собственно, и составляют тело триггера.

Ниже представлен пример триггера, который реализует одно из бизнес-правил, названных выше, а именно дата рождения пассажира не может быть больше даты отправки самолета. Остальные скрипты триггеров представлены в приложении А.

CREATE TRIGGER Check\_birthday ON ticket

AFTER INSERT, UPDATE

AS

DECLARE @birthday date

DECLARE @departure\_date date

SET @birthday = (SELECT birthday FROM passenger WHERE passenger.id = (SELECT id\_passenger FROM inserted))

SET @departure\_date = (SELECT departure\_time FROM flight\_info WHERE flight\_info.id\_flight = (SELECT id\_flight FROM inserted))

IF @birthday > @departure\_date

BEGIN

RAISERROR ('Birthday must be less than departure date', 16, 1);

ROLLBACK TRAN;

RETURN;

END

ELSE

PRINT 'Successful operation' RETURN;

END

ELSE

PRINT 'Successful operation'

## Заполнение базы данных

Для пробного наполнения базы данных используем команду INSERT INTO. Ниже представлены скрипты для наполнения таблиц «Тип самолета» и «Пассажир». Остальные скрипты по наполнению таблиц представлены в Приложении А.

Скрипт наполнения таблицы «Тип самолета»:

INSERT INTO aircraft\_type (name, seats\_number)

VALUES

(N'Туполев Ту-134', '80'),

(N'Туполев Ту-154', '152'),

(N'Туполев Ту-204', '196'),

(N'Сухой Суперджет-100', '90'),

(N'Боинг -737 (Boeing-737)', '114 ')

Скрипт наполнения таблицы «Пассажир»:

INSERT INTO passenger (surname, name, patronymic, mail, birthday, phone, document\_value, id\_document\_type)

VALUES

(N'Иванов', N'Андрей', N'Иванович', 'mail@mail.com', '1998-05-14', '123456789', '123321123', '1'),

(N'Петров', N'Иван', N'Сергеевич', 'mail1@mail.com', '2009-06-24', '987654321', '234432234', '2'),

(N'Сидоров', N'Евгений', N'Петрович', 'mail2@mail.com', '1997-02-12', '123321123321', '345543345', '1'),

(N'Васюшкин', N'Антон', N'Викторович', 'mail3@mail.com', '1991-10-02', '34554372435', '654456654', '1'),

(N'Столов', N'Денис', N'Артемович', 'mail4@mail.com', '2000-01-01', '7123523452', '756675675567', '1')

Проверим работу триггера. Попробуем добавить новый билет, в котором дата рождения пассажира больше даты отправки самолета с помощью следующего кода:

INSERT INTO ticket (cost, id\_class, id\_passenger, id\_flight) VALUES

('55555, '1', '16', '1')

где 16 – код пассажира с датой рождения 2021-06-25,

1 – код рейса с датой вылета 2021-06-01.

На рис. 3.5 представлен результат работы данного скрипта. Как и ожидалось, мы получили сообщение об ошибке с определенным нами текстом.



Рисунок 3.5 – Результат работы скрипта вставки неправильных значений

## Разработка запросов базы данных

При работе с таблицами можно в любой момент выбрать из базы данных необходимую информацию с помощью запросов. Запрос – это обращение к БД для поиска или изменения в базе данных информации, соответствующей заданным критериям.

В языке SQL запрос представлен командой SELECT, которая может выводить значения из всех таблиц, нескольких выборочных или только одной.

К примеру, нам необходимо вывести на экран информацию о билете, некоторую информацию о пассажире и полете для рейса, значение кода которого равно «1». Данные, полученные в результате работы данного скрипта, представлены на рисунке 3.6. Скрипты остальных запросов к базе данных представлены в Приложении А.

Скрипт запроса к базе данных:

SELECT ticket.cost, class.name,

passenger.name, passenger.surname,

passenger.document\_value, document\_type.name

FROM ticket

INNER JOIN class

ON ticket.id\_class = class.id

INNER JOIN passenger

ON ticket.id\_passenger = passenger.id

INNER JOIN document\_type

ON passenger.id\_document\_type = document\_type.id

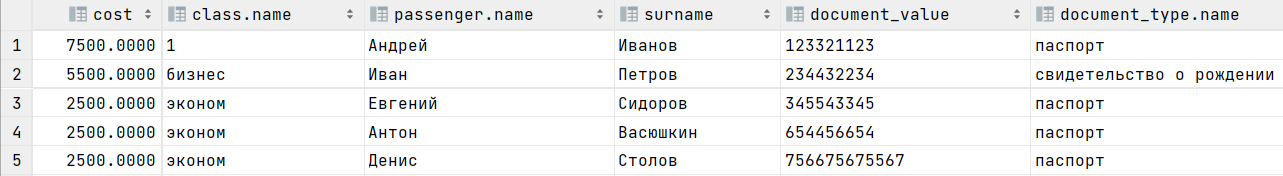


Рисунок 3.6 – Результат запроса к базе данных

## Разработка хранимых процедур для базы данных для онлайн магазина продажи авиабилетов

Хранимая процедура – это способ инкапсуляции повторяющихся действий. В хранимых процедурах можно объявлять переменные, управлять потоками данных, а также применять другие техники программирования. Причина их создания ясна и подтверждается частым использованием.

Хранимые процедуры представляют набор инструкций, которые выполняются как единое целое. Тем самым хранимые процедуры позволяют упростить комплексные операции и вынести их в единый объект. То есть процедура также упрощает управление кодом.

Также хранимые процедуры позволяют ограничить доступ к данным в таблицах и тем самым уменьшить вероятность преднамеренных или неосознанных нежелательных действий в отношении этих данных.

И еще один важный аспект - производительность. Хранимые процедуры обычно выполняются быстрее, чем обычные SQL-инструкции. Все, потому что код процедур компилируется один раз при первом ее запуске, а затем сохраняется в скомпилированной форме.

Для создания хранимой процедуры применяется команда CREATE PROCEDURE или CREATE PROC.

Таким образом, хранимая процедура имеет три ключевых особенности: упрощение кода, безопасность и производительность.

Ниже представлена хранимая процедура boing\_flights, которая возвращает список всех рейсов, которые осуществлялись с помощью самолета Боинг-737. Остальные процедуры представлены в Приложении А.

CREATE PROCEDURE boing\_flights AS

SELECT \* FROM flight

INNER JOIN aircraft\_type

ON aircraft\_type.id = flight.id\_aircraft\_type

WHERE aircraft\_type.name = N'Боинг -737 (Boeing-737)'

Для выполнения хранимой процедуры вызывается команда EXEC или EXECUTE:

EXECUTE boing\_flights

На рис. 3.6 представлен результат выполнения хранимой процедуры.

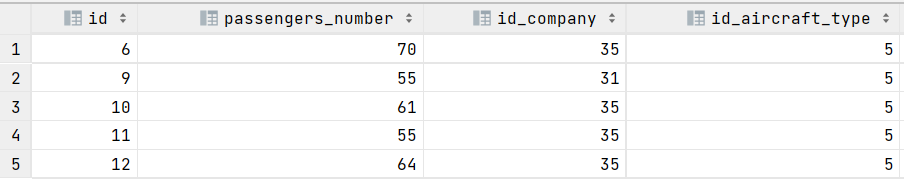


Рисунок 3.6. Результат выполнения хранимой процедуры

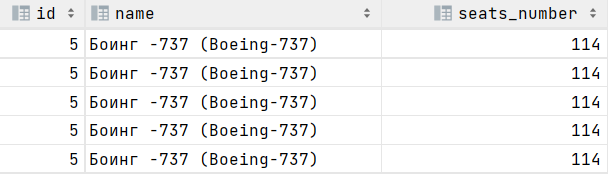


Рисунок 3.6. Результат выполнения хранимой процедуры (продолжение)

# Заключение

В результате выполнения курсовой работы была достигнута поставленная цель и выполнены все поставленные задачи:

* изучен процесс продажи авиабилетов в онлайн магазине;
* разработаны схемы базы данных и приведены к нормальным формам, выбрана СУБД;
* создана база данных для онлайн магазина по продаже авиабилетов;
* разработаны триггеры;
* разработаны хранимые процедуры;
* построены запросы.

# Список литературы

1. Дейт К. Дж. Введение в системы баз данных [Текст] / Дж. К. Дейт — 8-е изд. — М.: Вильямс, 2005. — 1328 с.;
2. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. — 3-е изд. [Текст] / Т. Конноли, К. Бегг. — М.: Вильямс, 2003. — 1436 с.;
3. Metanit – Введение в MS SQL Server и T-SQL [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/sqlserver/1.1.php. – дата обращения: 01.06.2021;
4. Кузнецов С. Д. Основы баз данных. — 2-е изд. — [Текст] / С.Д. Кузнецов.- М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с.;
5. Учебно-методический комплекс по дисциплине «Базы данных. Системы управления базами данных» - StudFiles.net [Электронный ресурс].- Режим доступа: https://studfiles.net/preview/5150865/page:14/. – дата обращения: 01.06.2021.

# Приложение А

**(**обязательное**)**

## Листинг базы данных

**Скрипты создания**

CREATE TABLE company

(

id INT PRIMARY KEY IDENTITY,

name NVARCHAR(30) NOT NULL

)

CREATE TABLE aircraft\_type

(

id INT PRIMARY KEY IDENTITY,

name NVARCHAR(50) NOT NULL,

seats\_number INT NOT NULL

)

CREATE TABLE flight

(

id INT PRIMARY KEY IDENTITY,

passengers\_number INT NOT NULL,

id\_company INT,

id\_aircraft\_type INT,

)

CREATE TABLE flight\_info

(

id INT PRIMARY KEY IDENTITY,

departure\_time DATETIME NOT NULL,

departure\_city NVARCHAR(50) NOT NULL,

arrival\_time DATETIME NOT NULL,

arrival\_city NVARCHAR(50) NOT NULL,

id\_flight INT

)

CREATE TABLE ticket

(

id INT PRIMARY KEY IDENTITY,

cost MONEY NOT NULL,

id\_class INT,

id\_passenger INT,

id\_flight INT,

)

CREATE TABLE class

(

id INT PRIMARY KEY IDENTITY,

name NVARCHAR (20) NOT NULL,

seats\_number INT NOT NULL,

id\_aircraft\_type INT

)

CREATE TABLE passenger

(

id INT PRIMARY KEY IDENTITY,

surname NVARCHAR(80) NOT NULL,

name NVARCHAR(80) NOT NULL,

patronymic NVARCHAR(80) NOT NULL,

mail NVARCHAR(100) NOT NULL,

birthday DATE NOT NULL,

phone NVARCHAR(20) NOT NULL,

document\_value NVARCHAR(30) NOT NULL,

id\_document\_type INT

)

CREATE TABLE document\_type

(

id INT PRIMARY KEY IDENTITY,

name NVARCHAR(30) NOT NULL,

)

**Скрипты изменения**

ALTER TABLE flight

ADD FOREIGN KEY (id\_company) REFERENCES company(id)

ALTER TABLE flight

ADD FOREIGN KEY (id\_aircraft\_type) REFERENCES aircraft\_type(id)

ALTER TABLE flight\_info

ADD FOREIGN KEY (id\_flight) REFERENCES flight(id)

ALTER TABLE ticket

ADD FOREIGN KEY (id\_class) REFERENCES class(id)

ALTER TABLE ticket

ADD FOREIGN KEY (id\_passenger) REFERENCES passenger(id);

ALTER TABLE ticket

ADD FOREIGN KEY (id\_flight) REFERENCES flight(id);

ALTER TABLE class

ADD FOREIGN KEY (id\_aircraft\_type) REFERENCES aircraft\_type(id)

ALTER TABLE passenger

ADD FOREIGN KEY (id\_document\_type) REFERENCES document\_type(id);

**Скрипты наполнения**

INSERT INTO company (name)

VALUES

('company1'),

('company2'),

('company3'),

('company4'),

('company5')

INSERT INTO aircraft\_type (name, seats\_number)

VALUES

(N'Туполев Ту-134', '80'),

(N'Туполев Ту-154', '152'),

(N'Туполев Ту-204', '196'),

(N'Сухой Суперджет-100', '90'),

(N'Боинг -737 (Boeing-737)', '114 ')

INSERT INTO document\_type (name)

VALUES

(N'паспорт'),

(N'свидетельство о рождении')

INSERT INTO flight (passengers\_number, id\_company, id\_aircraft\_type)

VALUES

('50', '31', '1'),

('60', '32', '2'),

('70', '33', '3'),

('80', '34', '4'),

('70', '35', '5'),

('60', '31', '4'),

('40', '31', '3'),

('55', '31', '5'),

('61', '35', '5'),

('55', '35', '5'),

('64', '35', '5')

INSERT INTO flight\_info (departure\_time, departure\_city, arrival\_time, arrival\_city, id\_flight)

VALUES

('2021-06-01 21:00:00', N'Москва', '2021-06-02 01:15:00', N'Санкт-Петербруг', '2'),

('2021-06-02 01:00:00', N'Москва', '2021-06-02 04:15:00', N'Санкт-Петербруг', '3'),

('2021-06-03 22:00:00', N'Москва', '2021-06-04 02:15:00', N'Санкт-Петербруг', '4'),

('2021-06-04 23:00:00', N'Москва', '2021-06-04 05:15:00', N'Санкт-Петербруг', '5'),

('2021-06-05 00:00:00', N'Санкт-Петербург', '2021-06-02 05:10:00', N'Казань', '2'),

('2021-06-06 04:00:00', N'Санкт-Петербург', '2021-06-04 07:10:00', N'Казань', '3'),

('2021-06-05 11:00:00', N'Санкт-Петербург', '2021-06-05 14:10:00', N'Казань', '8'),

('2021-06-05 12:00:00', N'Санкт-Петербург', '2021-06-05 15:10:00', N'Казань', '9'),

('2021-06-04 13:00:00', N'Казань', '2021-06-04 16:25:00', N'Москва', '10'),

('2021-06-04 14:00:00', N'Казань', '2021-06-04 17:25:00', N'Москва', '11')

INSERT INTO passenger (surname, name, patronymic, mail, birthday, phone, document\_value, id\_document\_type)

VALUES

(N'Иванов', N'Андрей', N'Иванович', 'mail@mail.com', '1998-05-14', '123456789', '123321123', '1'),

(N'Петров', N'Иван', N'Сергеевич', 'mail1@mail.com', '2009-06-24', '987654321', '234432234', '2'),

(N'Сидоров', N'Евгений', N'Петрович', 'mail2@mail.com', '1997-02-12', '123321123321', '345543345', '1'),

(N'Васюшкин', N'Антон', N'Викторович', 'mail3@mail.com', '1991-10-02', '34554372435', '654456654', '1'),

(N'Столов', N'Денис', N'Артемович', 'mail4@mail.com', '2000-01-01', '7123523452', '756675675567', '1')

INSERT INTO class (name, seats\_number, id\_aircraft\_type)

VALUES

(N'1', '10', '1'),

(N'бизнес', '20', '1'),

(N'эконом', '70', '1'),

(N'1 класс', '15', '2'),

(N'бизнес', '30', '2'),

(N'бизнес', '25', '3'),

(N'эконом', '60', '3'),

(N'эконом', '80', '4'),

(N'эконом', '90', '2')

INSERT INTO ticket (cost, id\_class, id\_passenger, id\_flight)

VALUES

('7500', '1', '1', '1'),

('5500', '2', '2', '1'),

('2500', '3', '3', '1'),

('2500', '3', '4', '1'),

('2500', '3', '5', '1'),

('2500', '4', '1', '2')

**Скрипты триггеров**

CREATE TRIGGER Check\_birthday ON ticket

AFTER INSERT, UPDATE

AS

DECLARE @birthday date

DECLARE @departure\_date date

SET @birthday = (SELECT birthday FROM passenger WHERE passenger.id = (SELECT id\_passenger FROM inserted))

SET @departure\_date = (SELECT departure\_time FROM flight\_info WHERE flight\_info.id\_flight = (SELECT id\_flight FROM inserted))

IF @birthday > @departure\_date

BEGIN

RAISERROR ('Birthday must be less than departure date', 16, 1);

ROLLBACK TRAN;

RETURN;

END

ELSE

PRINT 'Successful operation'

CREATE TRIGGER Check\_tickets\_count ON ticket

AFTER INSERT, UPDATE

AS

DECLARE @tickets\_count int;

DECLARE @max\_seats int;

DECLARE @id\_flight int

SET @id\_flight = (SELECT id\_flight FROM inserted);

SET @tickets\_count = (SELECT count(\*) FROM ticket WHERE ticket.id\_flight = @id\_flight)

SET @max\_seats = (SELECT aircraft\_type.seats\_number FROM aircraft\_type

INNER JOIN flight

ON flight.id = @id\_flight

INNER JOIN aircraft\_type

ON aircraft\_type.id = flight.id\_aircraft\_type

WHERE flight.id = @id\_flight)

IF @tickets\_count > @max\_seats

BEGIN

RAISERROR ('Count of Tickets must be less than max seats in aircraft', 16, 1);

ROLLBACK TRAN;

RETURN;

END

ELSE

PRINT 'Successful operation'

**Скрипты процедур**

CREATE PROCEDURE boing\_flights AS

SELECT \* FROM flight

INNER JOIN aircraft\_type

ON aircraft\_type.id = flight.id\_aircraft\_type

WHERE aircraft\_type.name = N'Боинг -737 (Boeing-737)'

**Скрипты запросов**

SELECT ticket.cost, class.name,

passenger.name, passenger.surname,

passenger.document\_value, document\_type.name

FROM ticket

INNER JOIN class

ON ticket.id\_class = class.id

INNER JOIN passenger

ON ticket.id\_passenger = passenger.id

INNER JOIN document\_type

ON passenger.id\_document\_type = document\_type.id

SELECT \* from passenger

WHERE birthday > '1999-01-01'

SELECT arrival\_city, arrival\_time

FROM flight\_info

INNER JOIN flight

ON flight\_info.id\_flight = flight.id

INNER JOIN ticket

ON ticket.id\_flight = flight.id