

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Алтайский государственный технический университет
им. И.И. Ползунова»

Факультет информационных технологий
Кафедра информатики, вычислительной техники и информационной
безопасности
Направление Информатика и вычислительная техника

Курсовая работа
защищена с оценкой отлично (80)
Л.Ю. Качесова
(подпись руководителя работы) (инициалы, фамилия)

“ 26 ” 12 2018 г.

КУРСОВАЯ РАБОТА

Разработка базы данных и клиентского приложения для менеджера
туристической фирмы
тема работы

Пояснительная записка
по дисциплине Базы данных
наименование дисциплины

КР 09.03.01.20.000 ПЗ
обозначение документа

Студент группы ИВТ-72 Романюк Анастасия Александровна
(фамилия, имя, отчество) (подпись) (дата)
Руководитель работы ст. преподаватель Л.Ю. Качесова
(должность, ученое звание) (подпись) (инициалы, фамилия)

Барнаул 2018

**РЕЦЕНЗИЯ
НА КУРСОВУЮ РАБОТУ**

Автор (студент/ка):

Романюк Анастасия Александровна

ИВТ-72

Тема:

Разработка базы данных и клиентского приложения для менеджера туристической фирмы

Факультет: Информационных технологий

Кафедра: ИВТи

Дисциплина: Базы данных

Рецензент: старший преподаватель Качесова Л.Ю.

Оценка курсовой работы

№	Показатели	Оценка (баллы)	Примечания
1	Актуальность тематики работы	80	
2	Степень полноты обзора состояния вопроса и корректность постановки задачи	80	
3	Уровень и корректность использования в работе литературных источников и методов исследований	80	
4	Соответствие разработанного ПО постановке задачи	80	
5	Работоспособность ПО	80	
6	Ясность, четкость, последовательность и обоснованность изложения	80	
7	Применение современных технологий в работе	80	
8	Соответствие использованных технологий и инструментария требованиям прикладной области	80	
9	Качество оформления (общий уровень грамотности, стиль изложения, соответствие требованиям стандартов)	80	

Отмеченные достоинства:

Клиентское приложение имеет удобный современный графический интерфейс. База данных спроектирована с использованием современного CASE средства (Toad Data Modeler). Для создания базы данных и клиентского приложения использовалось свободное ПО.

Отмеченные недостатки:

Наличие отклонений от требований стандарта к оформлению

Итоговая оценка

Рецензент

80 (80)

Л.Ю. Качесова

26.12.2018

Задание на курсовую работу

Учебная дисциплина: Базы данных

Студент: А.А. Романюк

Группа: ИВТ-72

Тема работы: Разработка базы данных и клиентского приложения
туристической фирмы

Дата выдачи задания: 01.09.2018

Дата защиты: 26.12.18

Руководитель Мсо ст. преподаватель кафедры ИВТиИБ Л.Ю. Качесова

Содержание

Введение.....	4
1 Описание предметной области.....	5
2 Проектирование базы данных.....	7
2.1 Построение концептуальной модели данных.....	7
2.2 Построение реляционной модели данных.....	11
3 Создание базы данных.....	18
4 Запросы. Представление. Триггеры и функции.....	20
4.1 Запросы.....	20
4.2 Представления в PostgreSQL.....	23
4.3 Функции в PostgreSQL.....	25
4.4 Триггеры в PostgreSQL.....	27
5 Описание и тестирование клиентского приложения.....	34
Заключение.....	42
Список использованных источников.....	43
Приложение А.....	44
Приложение Б.....	50

Введение

База данных - объективная форма представления и организации совокупности каких-либо данных (статей, нормативных актов, расчётов, информации о клиентах и сотрудниках и многое другое). Таким образом, база данных систематизирует материалы и обрабатывает их с помощью программы на компьютере.

Современное информационное поле состоит из массы событий, объектов и явлений. Оно охватывает такие объёмы информации, что без чётко действующей определённой системы, хранение всех этих данных могло быть неуправляемым и хаотичным.

База данных представляет собой способ управления хранимой информацией, и используется во всех сферах человеческой жизни.

Проектирование базы данных представляет собой сложный и трудоёмкий процесс отображения предметной области во внутреннюю модель данных, который под силу только профессионалам своего дела.

Подводя итоги, можно сказать, что база данных может хранить огромное количество систематизированной информации, и быстро предоставлять её после введения запроса пользователя.

Целью данной курсовой работы является создание базы данных для туристической фирмы. Задание также подразумевает создание необходимой системы управления этой базой данных СУБД.

Также целью данной работы является удовлетворение создания специализированной СУБД, рассчитанной на управление заранее определённой структурой информации и решение вполне определённого круга задач для туристической фирмы. Данная работа позволяет создать условия для будущих пользователей, позволяющие не отвлекаться на изучение вопросов, связанных с базами данных и средствами управления ими.

1 Описание предметной области

Требуется создать базу данных и программу обработки данных, предназначенную для работников туристической фирмы. База данных должна обеспечивать хранение сведений об имеющихся в продаже путевках и о клиентах фирмы. Сведения о путевке включают ее стоимость, время отправления и возвращения, маршрут, способы перемещения, места для проживания, экскурсии и прочие услуги, например, в стоимость путевки полностью или частично может входить питание. Путевка может предполагать посещение одной или нескольких стран, одного или нескольких населенных пунктов. Сведения о клиентах – это фамилия, имя, отчество, номер контактного телефона, паспортные данные, дата регистрации, особые замечания. Если у клиента есть загранпаспорт, то его номер, дата выдачи, срок действия должны быть зафиксированы в БД уже при регистрации. То же касается и виз: если клиент имеет визу, то в БД должны быть указаны сроки ее действия и тип визы. Если паспорта и/или визы у клиента нет, то работник туристической фирмы должен ее оформить по существующим расценкам в установленные сроки. В обязанности работника туристической фирмы входит продажа стандартных путевок, подбор индивидуальных туров для клиентов не предусмотрен. Клиент может высказать свои пожелания относительно сроков поездки, ее стоимости, стран, которые он хотел бы посетить. Для постоянных клиентов существует система скидок.

Работнику туристической фирмы могут потребоваться следующие данные:

- Какие есть путевки по цене, не превышающей ту, которую указал клиент?
- Можно ли отдохнуть в указанной стране в указанные сроки? Показать все возможные варианты.
- Сколько будет стоить оформление визы и паспорта при условии покупки указанной путевки?

- Какие путевки являются «горящими», то есть дата отправления, указанная в них, не более, чем на 5 дней больше текущей?
- Какие скидки возможны для постоянных клиентов фирмы?
- Какие путевки пользуются наибольшим спросом?

2 Проектирование базы данных

2.1 Построение концептуальной модели данных

Для проектирования концептуальной модели данных использовалась программа Toad Data Modeler.

На основе исследования предметной области «Туристическая фирма» было выявлено 9 сущностей : клиенты, путёвки, скидки, маршрут, страны, виза, заграничный паспорт, оформление визы, оформление паспорта. Подробное описание всех сущностей приведено в таблице 1.

Между сущностями установлены следующие связи. Рассмотрим связь между сущностями «Клиенты» и «Скидки». Клиенты могут не иметь скидки, либо иметь только одну, а скидки могут быть не только у одного клиента, но и у нескольких. Следовательно, связь между сущностями «Клиенты» и «Скидки» - один ко многим, неидентифицирующая.

Для создания связи между сущностями «Клиенты» и «Путёвки», используем дополнительную пустую сущность. Ведь один клиент может купить несколько путёвок, а путёвка может быть куплена разными клиентами, соответственно, связь между сущностями многие ко многим, но так как Toad Data Modeler не воспринимает такую связь, мы создаем дополнительную сущность. После проделанной операции, связь между данными сущностями - идентифицирующая, один ко многим.

Если рассматривать связь между сущностями «Путевки» и «Маршрут», то связь один к одному, неидентифицирующая. Так как у путевки должен быть один маршрут, ни больше, ни меньше. Так же и в обратную сторону: одному маршруту соответствует одна путевка.

Для создания связи между сущностями «Маршрут» и «Страны», используем дополнительную пустую сущность. Ведь один маршрут может проходить по нескольким странам, а маршрут в одну страну может проходить разными путями, соответственно, связь между сущностями многие ко многим, но так как Toad Data Modeler не воспринимает такую

связь, мы создаем дополнительную сущность. После проделанной операции, связь между данными сущностями - идентифицирующая, один ко многим.

Рассмотрим связь между сущностями «Клиенты» и «Загран.паспорт». Так как клиенты могут не иметь заграничного паспорта или иметь только один, а у заграничного паспорта обязательно должен быть владелец и только один. Следовательно, связь между сущностями «Клиенты» и «Загран.паспорт» - неидентифицирующая, один к одному.

Связь между сущностями «Клиенты» и «Оформление паспорта» - идентифицирующая, один к одному, зависимой сущностью является сущность «Оформление паспорта». Если у клиента нет заграничного паспорта, он должен его оформить и может оформить только один раз. А оформить паспорт может только один клиент.

Связь между сущностями «Загран.паспорт» и «Оформление паспорта» - идентифицирующая, один к одному, зависимой сущностью является сущность «Оформление паспорта». Если заграничного паспорта нет, то его нужно оформить и можно оформить только один раз.

Рассмотрим связь между сущностями «Клиенты» и «Виза». Так как клиенты могут не иметь визу или иметь только одну, а у визы обязательно должен быть владелец и только один. Следовательно, связь между сущностями «Клиенты» и «Виза» - неидентифицирующая, один к одному.

Связь между сущностями «Клиенты» и «Оформление визы» - идентифицирующая, один к одному, зависимой сущностью является сущность «Оформление визы». Если у клиента нет визы, он должен ее оформить и может оформить только один раз. А оформить визу может только один клиент.

Связь между сущностями «Виза» и «Оформление визы» - идентифицирующая, один к одному, зависимой сущностью является сущность «Оформление визы». Если визы нет, то ее нужно оформить и можно оформить только один раз.

На рисунке 1 размещена информационно-логическая модель данных предметной области «Туристическая фирма», созданной в Toad Data Modeler.

Таблица 1-Сущности и атрибуты

Сущность	Атрибут	Ключ
Скидки	Скидки в процентах	РК
Клиенты	Фамилия	
	Имя	
	Отчество	
	Серия и номер паспорта	РК
	Дата регистрации	
	Особые замечания	
Загран.паспорт	Номер	
	Дата выдачи	
	Срок действия	
Виза	Тип визы	
	Срок действия	
	Номер визы	РК
Оформление визы	Цена	
	Сроки	
Оформление паспорта	Цена	
	Сроки	
	Сроки	
Страны	Название	РК
	Посещение нескольких населенных пунктов	
	Экскурсии	
Маршрут	Дата и время отправления	
	Дата и время прибытия	
	Номер маршрута	РК
	Способ перемещения	
Путевки	Номер путевки	РК
	Стоимость	
	Места проживания	
	Прочие услуги	

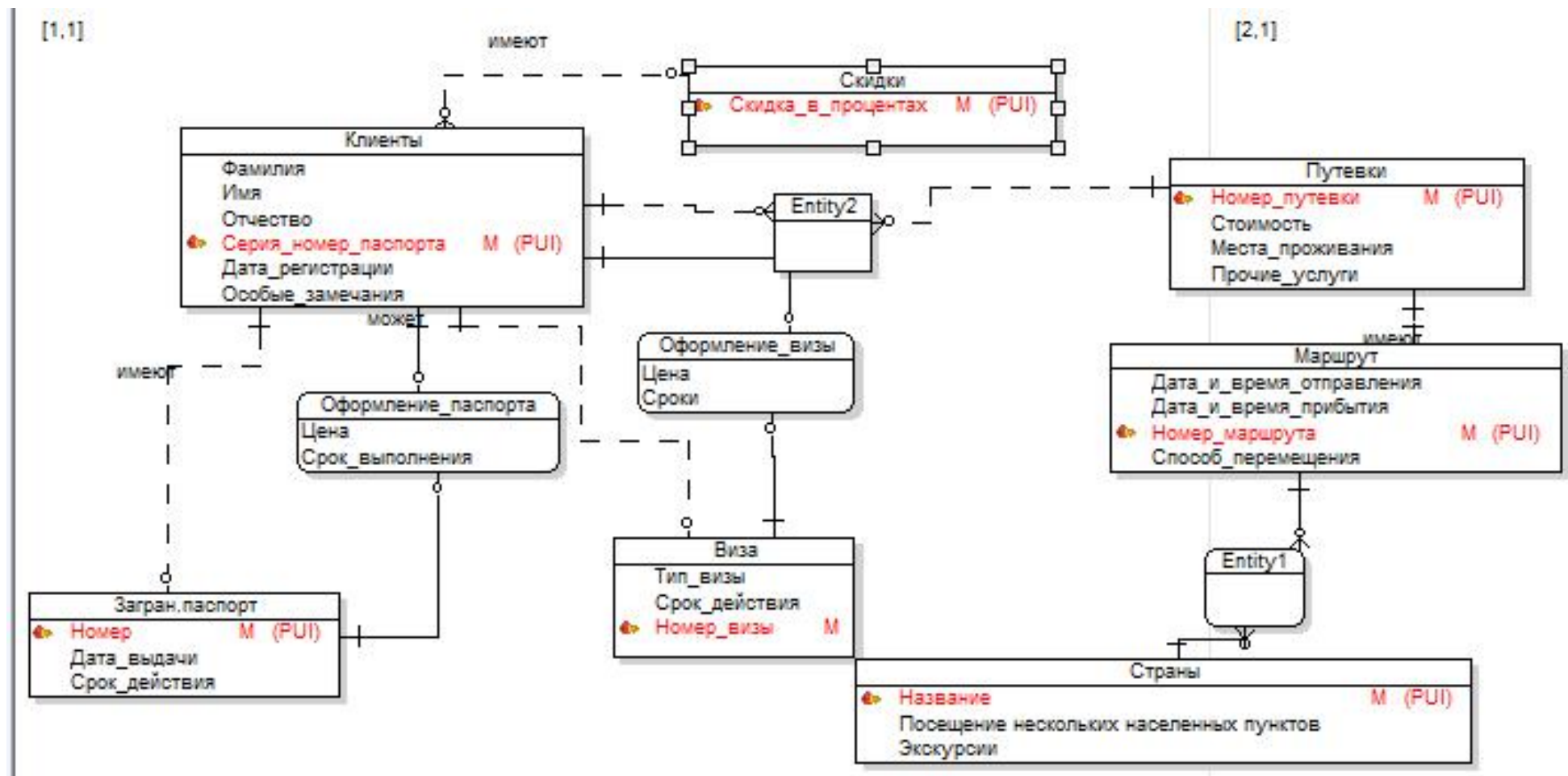


Рисунок 1 - Информационно-логическая модель данных предметной области «Туристическая фирма»

2.2 Проектирование реляционной базы данных

В Toad Data Modeler можно на основе логической модели данных получить физическую модель данных для выбранной СУБД. Преобразуем информационно-логическую модель данных предметной области «Музыкальные группы» в реляционную схему базы данных для СУБД PostgreSQL 9.6.5. Для этого в Toad Data Modeler необходимо в меню Model выбрать Convert Model -> Simple Conversion.

Полученную физическую модель данных демонстрирует рисунок 2. Обратим внимание на наличие внешних ключей (FK), которые используются в реляционной модели данных для организации связей между таблицами.

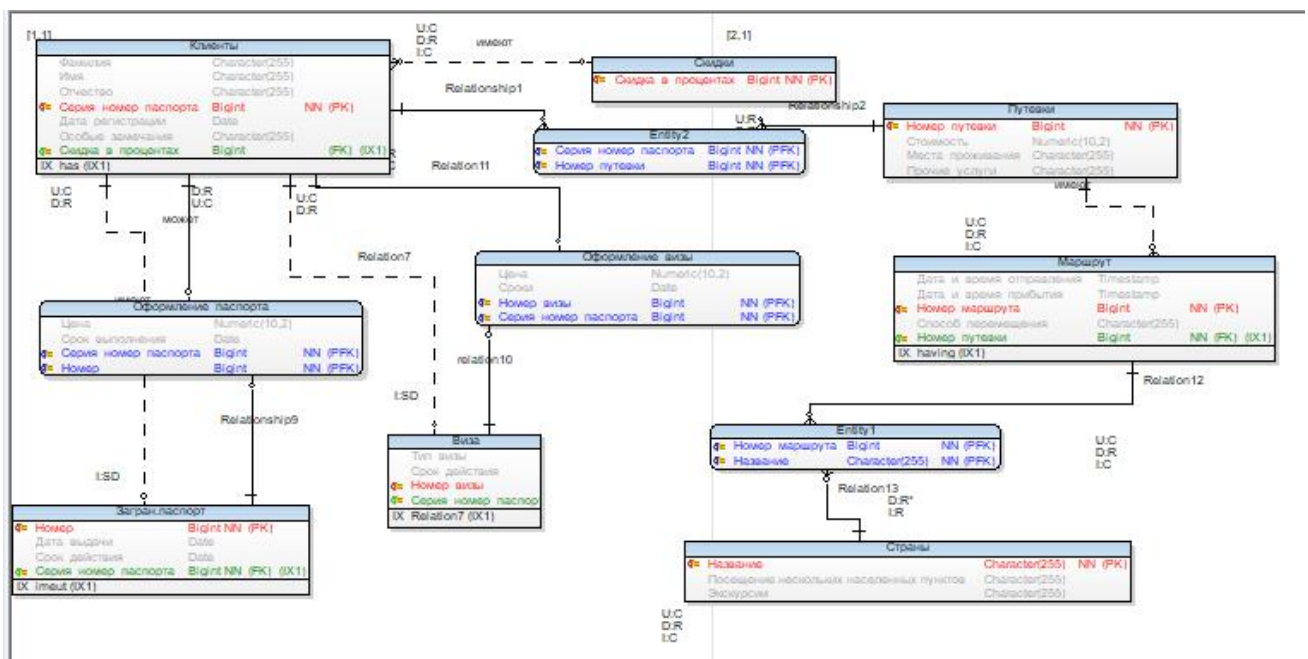


Рисунок 2 - Реляционная схема базы данных предметной области
«Туристическая фирма»

Сформулируем для связей правила поддержки ссылочной целостности.

Рассмотрим связь между таблицами «Клиенты» и «Загран.паспорт». Вставка строк в таблицу «Клиенты» всегда разрешена (стандартное правило поддержки ссылочной целостности), поскольку клиенты не обязаны иметь загран.паспорта. Удаление из таблицы «Клиенты» ограничено (Restrict), если за клиентом числится загран.паспорт, то запись о клиенте удалять нельзя, поскольку каждый загран.паспорт должен принадлежать какому-то клиенту. Когда меняется серия и номер паспорта клиента в таблице «Клиенты», нужно

распространить это изменение на одноименный столбец в таблице «Загран.паспорт», производя каскадное (Cascade) обновление. Рисунок 3 демонстрирует установку в Toad Data Modeler правил поддержки ссылочной целостности для связи между таблицами «Клиенты» и «Загран.паспорт».

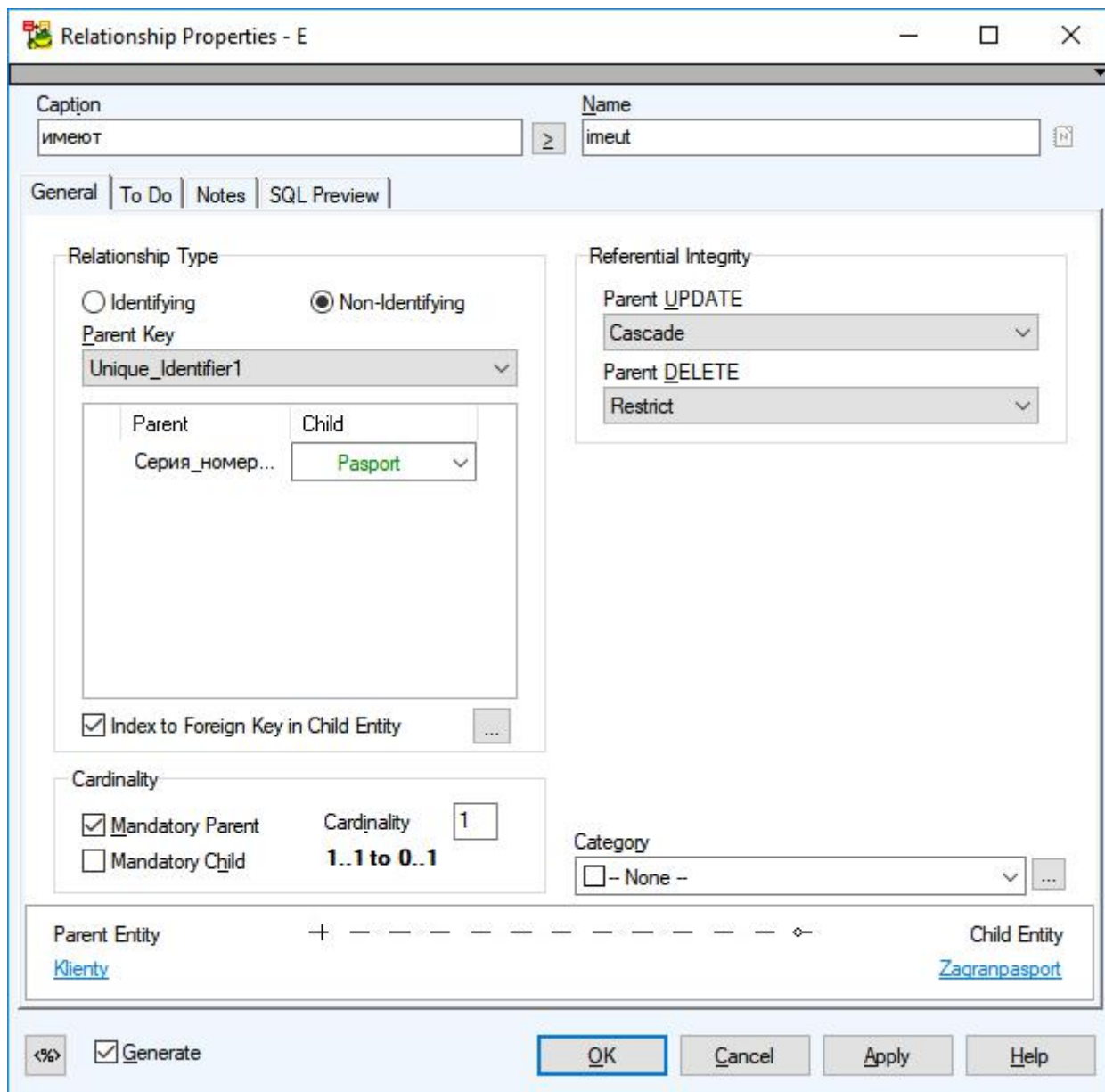


Рисунок 3 -Установка правил поддержки ссылочной целостности для связи между таблицами «Клиенты» и «Загран.паспорт» (см. Referential Integrity).

Когда новая строка вставляется в таблицу «Загран.паспорт», соответствующий загран.паспорт по умолчанию присваивается некоторому клиенту (Set Default). Это необходимо, поскольку всякий загран.паспорт должен числиться за каким-то клиентом. Правило Set Default в Toad Data

Modeler не поддерживается, его необходимо программировать в PostgreSQL с помощью триггеров.

При обновлении и удалении строк в таблице «загран.паспорт» подходят стандартные правила поддержки ссылочной целостности (удаление разрешено; обновление внешнего ключа запрещено, если обновленное значение внешнего ключа не соответствует ни одному значению первичного ключа в родительской таблице), поскольку клиенты не обязаны иметь заграничный паспорт.

Аналогичные правила поддержки ссылочной целостности можно сформулировать для связи между таблицами «Клиенты» и «Виза».

Рассмотрим идентифицирующие связи между таблицами: «Клиенты» и «Оформление паспорта», «Клиенты» и «Оформление визы». Со стороны родительских сущностей действуют правила каскадного обновления и ограниченного удаления (Restrict). Добавление строк в эти таблицы разрешено, так как у них необязательные потомки. Со стороны дочерних таблиц действуют стандартные правила поддержки ссылочной целостности (удаление разрешено; вставка новой строки запрещена, если значение внешнего ключа в новой строке не соответствует ни одному значению первичного ключа в родительской таблице; обновление внешнего ключа запрещено, если обновленное значение внешнего ключа не соответствует ни одному значению первичного ключа в родительской таблице).

Рассмотрим связь между таблицами «Путевки» и «Маршрут». В этой связи имеется обязательный потомок и обязательный родитель. Когда новая строка вставляется в таблицу «Путевки», соответствующей группе по умолчанию присваивается некоторый маршрут (Set Default). Это необходимо, поскольку путевка должна иметь, как минимум, один маршрут. Удаление из таблицы «путевка» ограничено (Restrict), если за путевкой числится какой-либо маршрут, то запись о путевке удалять нельзя (стандартное правило поддержки ссылочной целостности устанавливается в Toad Data Modeler. Когда меняется номер путевки в таблице «путевки», нужно распространить

это изменение на одноименный столбец в таблице «маршрут», производя каскадное (Cascade) обновление. Рисунок 4 демонстрирует установку в Toad Data Modeler правил поддержки ссылочной целостности для связи между таблицами «путевки» и «маршрут». Когда новая строка вставляется в таблицу «маршрут», соответствующий маршрут по умолчанию присваивается некоторой путевке (Set Default). Это необходимо, поскольку всякий маршрут должен числиться за какой-либо путевкой. Правило Set Default в Toad Data Modeler не поддерживается, его необходимо программировать в PostgreSQL с помощью триггеров.

При обновлении и удалении строк в таблице «маршрут» необходимо учитывать тот факт, что путевка должна иметь как минимум один маршрут. Соответственно, удалять маршрут запрещено, если он является единственным у путевки. Обновление внешнего ключа запрещено, если маршрут является единственным у путевки.

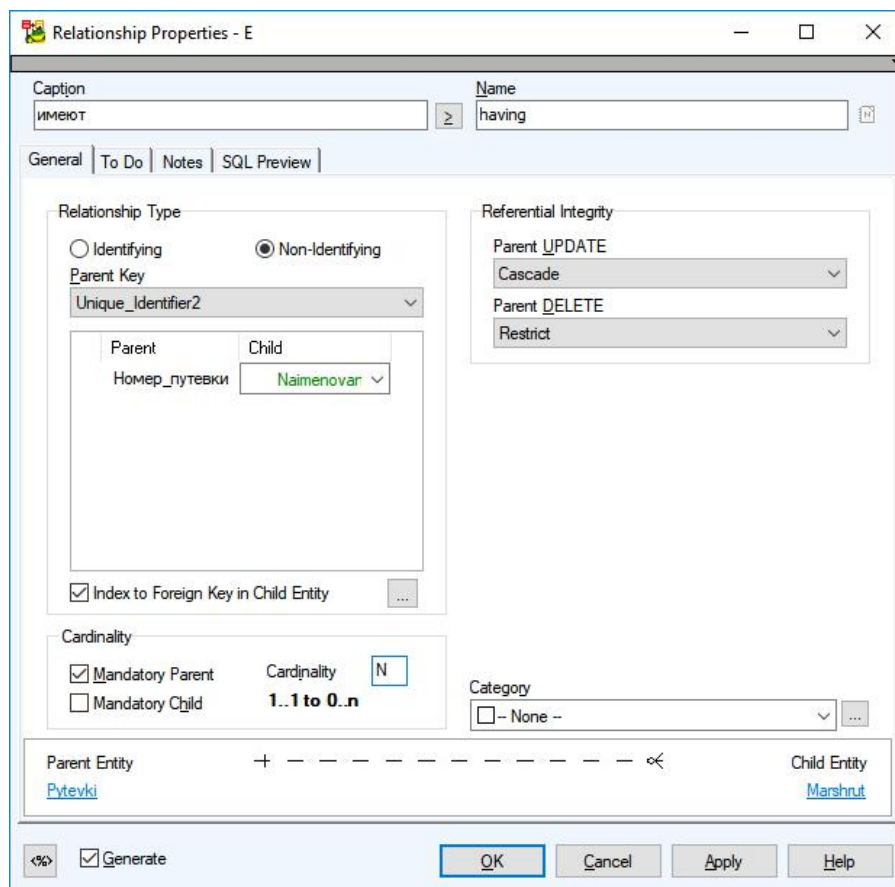


Рисунок 4 - Правила поддержки ссылочной целостности для связи между таблицами «Путевки» и «Маршрут»

Рассмотрим связь между таблицами: «Маршрут» и «Entity1». Здесь процедуры обеспечения ссылочной целостности более сложны, так как в этой связи имеется обязательный родитель.

Для таблицы «Маршрут» действуют следующие правила поддержки ссылочной целостности: Delete Restrict (если маршрут существует и в нем состоят страны, то запись об этом маршруте удалять нельзя – стандартное правило); Update Cascade (каскадное обновление номера маршрута); Insert Cascade (при добавлении нового маршрута при необходимости для него добавить страну в таблицу «Страны»). Правила Delete Restrict и Update Cascade можно установить в Toad Data Modeller. Правило Insert Cascade нужно программировать в СУБД, например, PostgreSQL с использованием триггеров. Рисунок 5 демонстрирует установку в Toad Data Modeler правил.

Для таблицы «Entity1» действуют следующие правила поддержки ссылочной целостности: запрещено добавление страны для не существующего маршрута (стандартное правило); если удаляемая или обновляемая страна (обновляется номер маршрута) является единственной в маршруте, то удалять её и обновлять для неё номер маршрута запрещено (эти правила нужно программировать в СУБД, например, PostgreSQL с использованием триггеров).

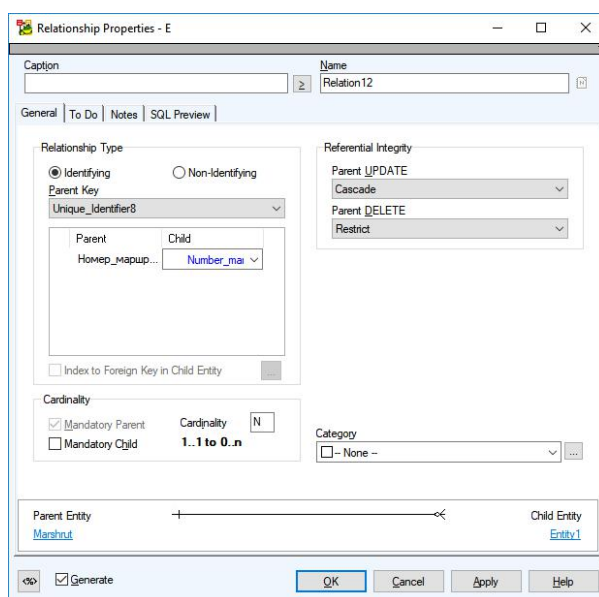


Рисунок 5 - Правила поддержки ссылочной целостности для связи между таблицами «Маршрут» и «Entity1»

Рассмотрим связь между таблицами: «Страны» и «Entity1». Здесь процедуры обеспечения ссылочной целостности более сложны, так как в этой связи имеется обязательный родитель.

Для таблицы «Страны» действуют следующие правила поддержки ссылочной целостности: Delete Restrict (если страна существует и относится к какому-либо маршруту, то запись об этой стране удалять нельзя – стандартное правило); Update Cascade (каскадное обновление номера маршрута); Insert Cascade (при добавлении новой страны при необходимости для неё добавить маршрут в таблицу «Маршруты»). Правила Delete Restrict и Update Cascade можно установить в Toad Data Modeller. Правило Insert Cascade нужно программировать в СУБД, например, PostgreSQL с использованием триггеров. Рисунок 6 демонстрирует установку в Toad Data Modeller правил.

Для таблицы «Entity1» действуют следующие правила поддержки ссылочной целостности: запрещено добавление маршрута для не существующей страны (стандартное правило); если удаляемый или обновляемый маршрут (обновляется название страны) является единственным для страны, то удалять его и обновлять для него название страны запрещено (эти правила нужно программировать в СУБД, например, MySQL с использованием триггеров).

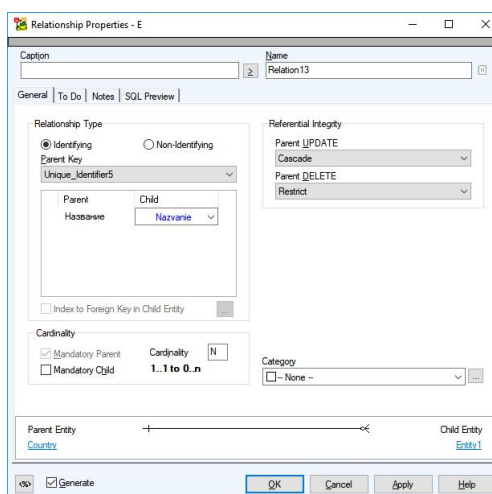


Рисунок 6 - Правила поддержки ссылочной целостности для связи между таблицами «Страны» и «Entity1»

Правила поддержки ссылочной целостности между таблицами «Клиенты» и «Путёвки» аналогичны правилам между таблицами «Маршрут» и «Страны», рассмотренных ранее.

3 Создание базы данных

Для создания базы данных необходимо установить СУБД PostgreSQL 9.6.5 и EMS SQL Manager for PostgreSQL.

Создадим пустую базу данных в EMS SQL Manager for PostgreSQL. Он предоставляет удобный диалоговый интерфейс для создания базы данных и работы с ней. Запустим мастер создания базы данных (База данных > Создать базу данных).

После создания базы данных автоматически появляется диалоговое окно регистрации базы данных. В диалоговом окне вводится информация, необходимая для регистрации базы данных.

Получим в Toad Data Modeler на основе логической модели предметной области «Туристическая фирма» (рисунок 1) SQL скрипт с описанием таблиц базы данных. Описание данного скрипта приведено в приложении А.

EMS SQL Manager for PostgreSQL предоставляет программы-мастера, которые предоставляют удобный диалоговый интерфейс для создания таблиц базы данных. Мы же воспользуемся другим инструментом (Инструменты → SQL-скрипт). С помощью инструмента «SQL-скрипт» выполним SQL-скрипт с описанием таблиц базы данных «Туристическая фирма» и связей между ними (SQL-скрипт с описанием таблиц и связей создан в Toad Data Modeller на основе логической модели (Model→Convert model→Run→PostgreSQL 9.5)).

После создания одиннадцати таблиц приступим к их заполнению, соблюдая все правила заполнения. Сначала заполняются таблицы, не содержащие внешних ключей, а потом таблицы с внешними ключами. Для заполнения можно использовать удобный интерфейс программы EMS SQL Manager for PostgreSQL. Также можно воспользоваться операторами INSERT, UPDATE, DELETE, которые выполняются с помощью SQL-скрипта.

На рисунке 7 представлена заполненная таблица базы данных «Туристическая фирма».

Chena	Sroki	Number	Pasport
5 000	05.06.2020	134	112 865 434
6 000	07.07.2020	498	117 849 376
10 000	31.12.2020	485	113 846 502

Рисунок 7 — таблица «Оформление паспорта», заполненная данными

4 Запросы. Представления. Триггеры и функции

4.1 Запросы

Создадим запросы, сформулированные в описании предметной области курсовой работы.

- Какие есть путевки по цене, не превышающей ту, которую указал клиент?

```
select * from "Pytevki"
```

```
where "Stoimost"<='50000';
```

№	Наименование	Стоимость	Питание	Прочие
1		30 000	нет	завтрак в номер
2		50 000	отель 4 звезды	шведский стол
4		45 000	отель 3 звезды	нет

Рисунок 8 - путевки по цене, не превышающей ту, которую указал клиент

- Можно ли отдохнуть в указанной стране в указанные сроки? Показать все возможные варианты.

```
select
```

```
"Country"."Nazvanie","Ekskursii","Marshrut"."Data_and_time_otpravleniya","  
Date_and_time_pribytia","Sposob_peremeshenia"
```

```
from "Country", "Marshrut","Entity1"
```

```
where "Country"."Nazvanie"="Entity1"."Nazvanie" and
```

```
"Marshrut"."Number_marshruta"="Entity1"."Number_marshruta"
```

```
and "Entity1"."Nazvanie"='Китай' and
```

```
"Data_and_time_otpravleniya">='12.12.2018' and
```

```
"Date_and_time_pribytia"<='20.12.2018';
```

Nazvanie	Ekskursii	Data_and_time_otpravleniya	Date_and_time_pribytia	Sposob_peremeshenia
Китай	историческая	12.12.2018	20.12.2018	самолет

Рисунок 9 - страны

- Сколько будет стоить оформление визы и паспорта при условии покупки указанной путевки?

```
select "Chen","Chena","Pytevki"."Naimenovanie","Klienty"."Pasport"
```

```
from "Oformlenie_pasporta","Oformlenie_visy","Pytevki","Entity2","Klienty"
```

where "Oformlenie_pasporta"."pasport"="Klienty"."Pasport" and
 "Klienty"."Pasport"="Oformlenie_visy"."pasport"
 and "Klienty"."Pasport"="Entity2"."Pasport" and
 "Pytevkki"."Naimenovanie"='4'

Chen	Chena	Naimeno'	Pasport
6 000	4 000	4	112 567 784
6 000	4 000	4	112 567 784
3 000	3 000	4	118 075 345

Рисунок 10 - стоимость оформления визы и паспорта, для указанной
 путевки

- Какие путевки являются «горящими», то есть дата отправления, указанная в них, не более, чем на 5 дней больше текущей?

```
select "Data_and_time_otpravlenia","Naimenovanie"
from "Marshrut"
where "Data_and_time_otpravlenia" BETWEEN '19.11.2018' and
'24.11.2018';
```

Data_and_time_otpravle	Naimeno'
20.11.2018	2

Рисунок 11 - «горящие» путевки

- Какие скидки возможны для постоянных клиентов фирмы?

```
select "Skidka","vremy_v_godax","vremy"
from "Skidki"
where "vremy"='любимчик' or "vremy"='постоянный';
```

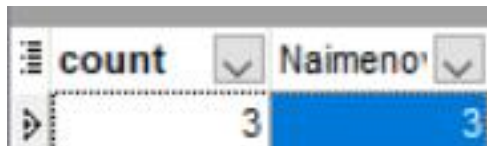
Skidka	vremy_v_	vremy
10	3	постоянный
15	5	любимчик

Рисунок 12 - скидки для постоянных клиентов

- Какие путевки пользуются наибольшим спросом?

```
select count(*),"Naimenovanie"
from "Entity2"
group by "Naimenovanie"
```

```
HAVING count(*)in(  
select count(*)total  
from "Entity2"  
group by "Naimenovanie"  
order by total desc  
limit 1);
```



count	Naimenovanie
3	3

Рисунок 13 - путёвки, пользующиеся большим спросом

4.2 Представления в PostgreSQL

В SQL Manager for PostgreSQL для работы с представлениями существует удобный интерфейс, который доступен из подменю «Представления» выбранной базы данных. Для сохранения представления используется команда «Компилировать». Так же можно использовать для создания представления инструмент SQL-скрипт.

Воспользуемся SQL-скриптом для создания представления для базы данных «Туристическая фирма». Представление будет отображать фамилии клиентов, номера путевок, способ перемещения, дату и время отправления, где дата отправления находится между 20.11.2018 и 31.12.2018.

```
CREATE OR REPLACE VIEW public.view(  
    "Familia",  
    "Naimenovanie",  
    "Sposob_peremeshenia",  
    "Data_and_time_otpravlenia")  
AS  
SELECT "Klienty"."Familia",  
    "Pyteyki"."Naimenovanie",  
    "Marshrut"."Sposob_peremeshenia",  
    "Marshrut"."Data_and_time_otpravlenia"  
FROM "Klienty",  
    "Pyteyki",  
    "Marshrut",  
    "Entity2"  
WHERE "Klienty"."Pasport" = "Entity2"."Pasport" AND  
    "Pyteyki"."Naimenovanie" = "Entity2"."Naimenovanie" AND  
    "Pyteyki"."Naimenovanie" = "Marshrut"."Naimenovanie" AND  
    "Marshrut"."Data_and_time_otpravlenia" >= '2018-11-20 00:00:00'::  
    timestamp without time zone AND  
    "Marshrut"."Data_and_time_otpravlenia" <= '2018-12-31 00:00:00'::
```


timestamp without time zone;

Familia	Naimeno	Sposob_pere meshenia	Data_and_time_otpravleni a
Моисеева	2	самолет	20.11.2018
Романюк	2	самолет	20.11.2018
Романюк	1	самолет	12.12.2018
Кондуров	1	самолет	12.12.2018
Кондуров	4	автобус	31.12.2018

Рисунок 14 - данные, которые отображает представление
Созданное представление является не обновляемым.

4.3 Функции в PostgreSQL

Создадим в EMS SQL Manager for PostgreSQL функцию для базы данных «Туристическая фирма». Для создания функции будем использовать SQL-скрипт, а для их выполнения удобный интерфейс, который доступен из подменю «Функции» выбранной базы данных.

Функция будет выдавать номер загранпаспорта клиента по номеру паспорта.

```
CREATE OR REPLACE FUNCTION public.func (  
    inout pas bigint  
)  
RETURNS bigint AS'  
SELECT number  
from zagranpasport,klienty  
where zagranpasport.pasport=pas and zagranpasport.pasport=klienty.pasport  
'LANGUAGE 'sql'  
VOLATILE  
CALLED ON NULL INPUT  
SECURITY INVOKER  
COST 100;
```

Функция | Параметры конфигурации | Зависимости | Описание | DDL | Привилегии

Имя: public.func

Возв. значение: ☒ Одно значение ☐ Набор значений ☐ Таблица ☐ Ничего ☐ Триггер

Возв. тип: BIGINT

Возв. таблица:

Имя столбца	Тип столбца
-------------	-------------

Язык: sql

Аргументы:

Имя	Тип	Режим	Значение по ум.
1 pas	BIGINT	Входной/выходной	

Параметры планировщика (по умолчанию 0):

Ожидаемая стоимость выполнения: 100

Примерное кол-во строк: 0

Определение:

```
1 select number  
2 from zagranpasport,klienty  
3 where zagranpasport.pasport=pas and zagranpasport.pasport=klienty.pasport
```

Рисунок 15 - параметры функции

Введите значения параметра

pas
bigint

☐ NULL

112830732

OK Отмена

Рисунок 16 - ввод значения

```
Запрос ОК (выполнено: 0 мс; всего: 141 мс)  
Возвращаемое значение: 567789
```

Рисунок 17 - результат

4.4 Триггеры PostgreSQL

Для создания триггеров воспользуемся SQL-скриптом в EMS SQL Manager for PostgreSQL для базы данных «Туристическая фирма».

Создадим триггеры для событий UPDATE и DELETE в таблице "Путёвка-Маршрут". Исходя из бизнес-правила курсовой работы, при удалении маршрута или обновлении внешнего ключа, мы не можем оставить маршрут без единой путёвки. У любого маршрута должна быть, как минимум, одна путёвка.

Триггер на событие UPDATE и DELETE в таблице «Путёвка-Маршрут» :

```
CREATE TRIGGER trigger7
```

```
AFTER UPDATE OF "Naimenovanie" OR DELETE
```

```
ON public."Marshrut" FOR EACH ROW
```

```
EXECUTE PROCEDURE public.ed();
```

```
DECLARE
```

```
    p integer;
```

```
BEGIN
```

```
    SELECT COUNT(OLD."Naimenovanie")into p
```

```
    FROM "Marshrut"
```

```
    WHERE "Naimenovanie" =OLD."Naimenovanie";
```

```
    IF (p=1)THEN raise exception 'Данный маршрут имеет единственную  
путёвку!';
```

```
    END IF;
```

```
    return OLD;
```

```
END;
```

Перетащите заголовок столбца сюда, чтобы группировать по нему

Data_and_time_otpravleni	Date_and_time_pribyti	Number	Sposob_premeshenia	Naimenovanie
18.12.2018	31.12.2018	1	самолет	1
11.12.2018	11.01.2019	2	самолет	3
15.01.2019	30.01.2019	3	поезд	2
11.11.2018	30.11.2018	4	самолет	4
23.12.2018	13.01.2019	5	самолет	1

Рисунок 18 — у маршрута 5 одна путёвка

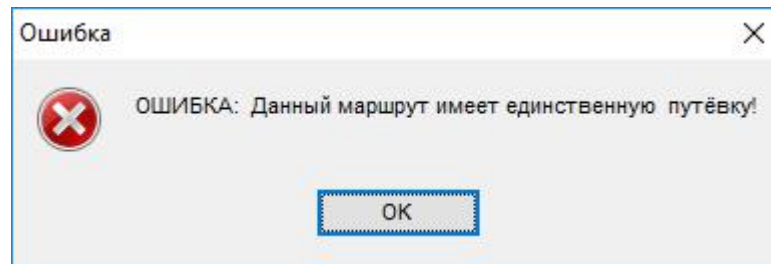


Рисунок 19 — сообщение об ошибке при удалении маршрута

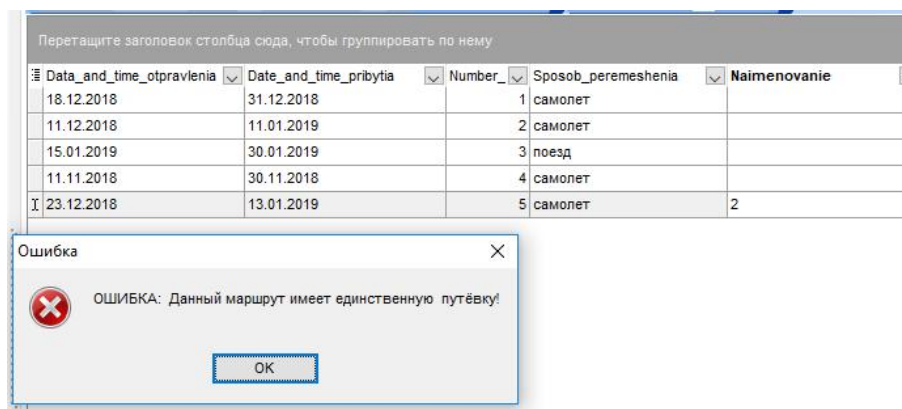


Рисунок 20 — сообщение об ошибке при обновлении внешнего ключа

Создадим триггеры для событий UPDATE и INSERT в таблице "Загран.паспорт". Исходя из бизнес-правила курсовой работы, при вставке паспорта или обновлении внешнего ключа, мы не можем допустить, чтобы у клиента было 2 или более заграничных паспортов. У любого клиента должен быть только один заграничный паспорт.

```
CREATE TRIGGER trigger1
AFTER INSERT OR UPDATE OF "Pasport"
ON public."Zagranpasport" FOR EACH ROW
EXECUTE PROCEDURE public.vstavka();
DECLARE
```

```

k BIGINT;

BEGIN

select count("Pasport")into k

from "Zagranpasport"

where "Pasport"="Pasport";

if (k>=1)then raise exception 'Клиент не может иметь больше одного
паспорта!';

end if;

return null;

END;

```

Перетащите заголовок столбца сюда, чтобы группировать по н

iii	Number	Data_vadachi	Srok	Pasport
	11	17.07.2018	17.07.2022	112 984 857
	12	17.07.2018	17.07.2022	112 830 753
	13	12.12.2018	12.12.2022	117 654 321

Рисунок 21 - данные до изменения

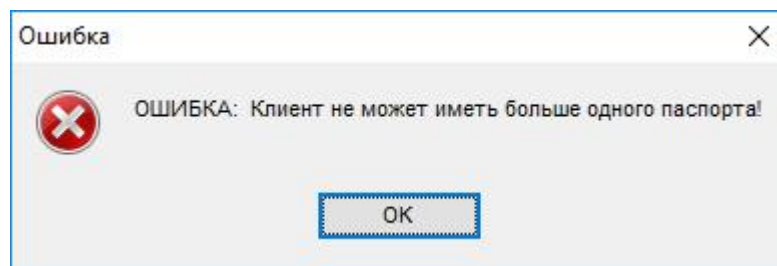


Рисунок 22 - сообщение об ошибке на обновление ключа с такими же параметрами

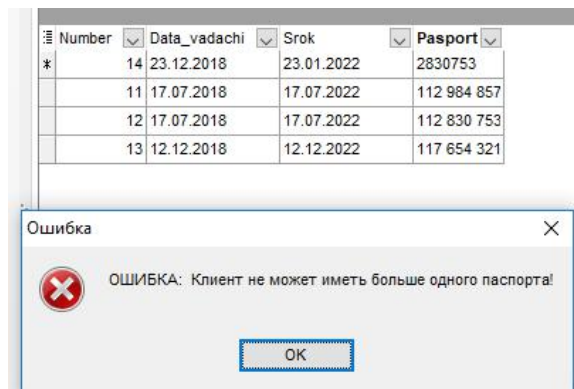


Рисунок 23 - вставка нового паспорта для одного и того же клиента

Такие же триггеры нужны для таблиц «Виза», «Оформление визы», «Оформление паспорта».

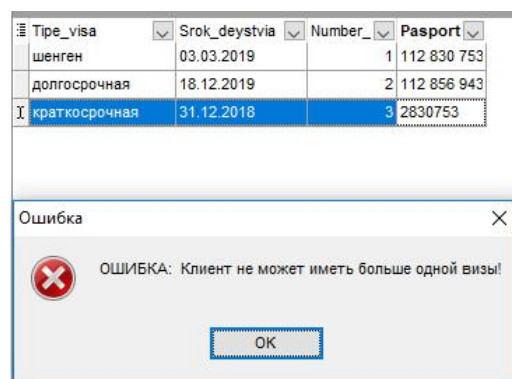


Рисунок 24 - попытка обновления паспорта у клиента, уже имеющего визу

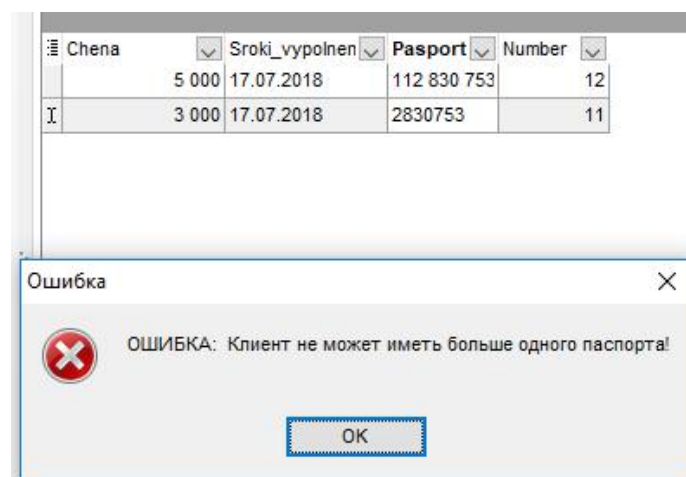


Рисунок 25 - сообщение об ошибке в таблице «Оформление паспорта»

Создадим триггер, который запрещает удаление клиента, если он имеет загран.паспорт :

CREATE TRIGGER trigger6

```

BEFORE DELETE

ON public."Klienty" FOR EACH ROW

EXECUTE PROCEDURE public.delete();

DECLARE p BIGINT;

BEGIN

select count( OLD."Pasport") into p

from "Zagranpasport"

where "Pasport"=OLD."Pasport";

if (p=1)then

    raise exception 'У клиента есть загранпаспорт. Удалять информацию о
клиенте нельзя!';

end if;

return null;

END;

```

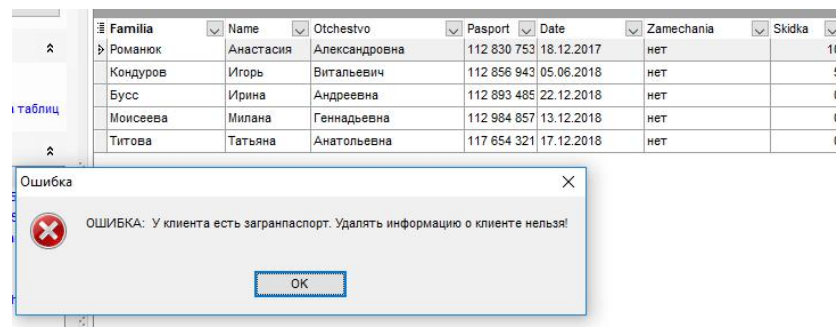


Рисунок 26 - попытка удаления клиента, имеющего загран.паспорт

Создадим триггер, который запрещает удаление или обновление маршрута, если он имеет единственную путёвку :

```

CREATE TRIGGER trigger7

AFTER UPDATE OF "Naimenovanie" OR DELETE

ON public."Marshrut" FOR EACH ROW

EXECUTE PROCEDURE public.ed();

```



```

DECLARE

p integer;

BEGIN

SELECT COUNT(OLD."Naimenovanie")into p

FROM "Marshrut"

WHERE "Naimenovanie" =OLD."Naimenovanie";

IF (p=1)THEN raise exception 'Данный маршрут имеет единственную
путёвку!';

END IF;

return OLD;

END;

```

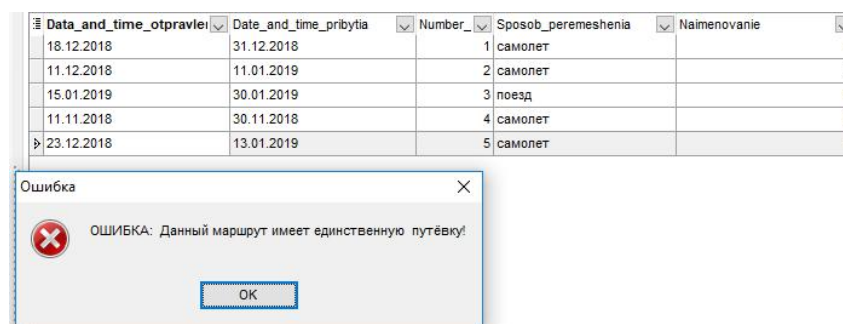


Рисунок 27 - удаление маршрута, имеющего только одну путёвку

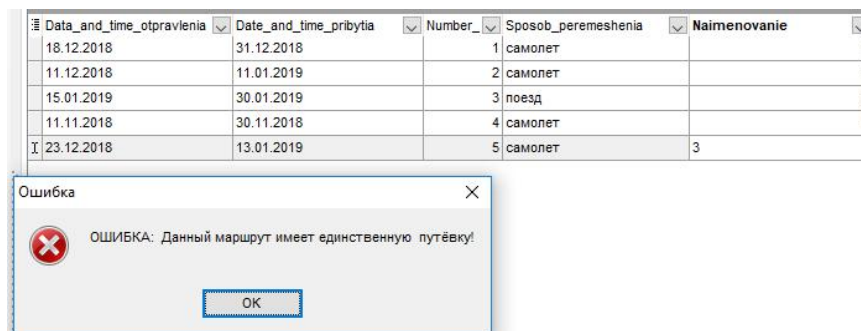


Рисунок 28 - изменение номера путёвки, являющейся единственной у маршрута

Допустим, у туристической фирмы есть бизнес-правило: стоимость путёвки не должна быть меньше 30000 рублей. Поэтому необходимо создать триггер на вставку и обновление стоимости :

```
CREATE TRIGGER trigger3
```

```
AFTER INSERT OR UPDATE OF "Stoimost"
```

```
ON public."Pytevki" FOR EACH ROW
```

```
EXECUTE PROCEDURE public.stoim('NEW');
```

```
BEGIN
```

```
IF (NEW."Stoimost"<30000)
```

```
then
```

```
raise EXCEPTION'Стоимость путевки не может быть меньше 30000
рублей!' ;
```

```
END IF;
```

```
return null;
```

```
END;
```

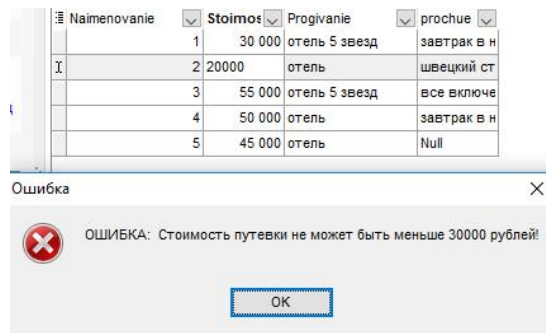


Рисунок 29 - обновление стоимости

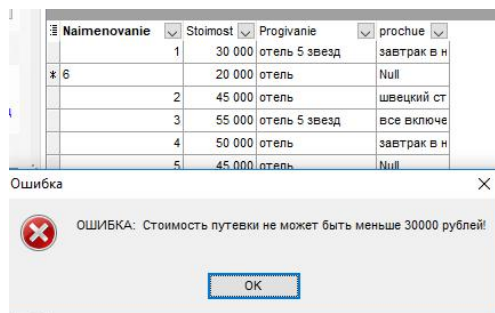


Рисунок 30 - вставка новой путёвки

5 Описание и тестирование клиентского приложения

Клиентское приложение написано в среде Visual Studio 2017 на языке программирования высокого уровня C#.

Исходный код программы находится в приложении В. Подключение к базе данных осуществляется при помощи ссылки на библиотеку .Net Data Provider for PostgreSQL. Она предоставляет унифицированные средства взаимодействия приложения-клиента с сервером БД. Был установлен коннектор Npgsql 2.2.3.0-bin-ms.net3.5sp1.

Файл Form1.cs содержит в себе следующие методы:

- 1) mainMenu1_Table1_Select - описание таблицы «Клиенты»;
- 2) mainMenu1_Table2_Select - описание таблицы «Путёвки»;
- 3) mainMenu1_Table3_Select - описание таблицы «Клиенты - Путёвки»;
- 4) mainMenu1_Table4_Select - описание таблицы «Маршрут»;
- 5) mainMenu1_Table5_Select - описание таблицы «Страны»;
- 6) mainMenu1_Table6_Select - описание таблицы «Маршрут - Страны»;
- 7) mainMenu1_Table7_Select - описание таблицы «Виза»;
- 8) mainMenu1_Table8_Select - описание таблицы «Загран.паспорт»;
- 9) mainMenu1_Table9_Select - описание таблицы «Оформление заграничного паспорта»;
- 10) mainMenu1_Table10_Select - описание таблицы «Оформление визы»;
- 11) mainMenu1_Save_Select - описание сохранения данных;
- 12) mainMenu1_Del_Select - описание удаления данных.

С mainMenu1_Zapros1_Select по mainMenu1_Zapros6_Select описание запросов.

При запуске программы появляется окно, в котором нам доступно меню управления программой (рисунок 31).

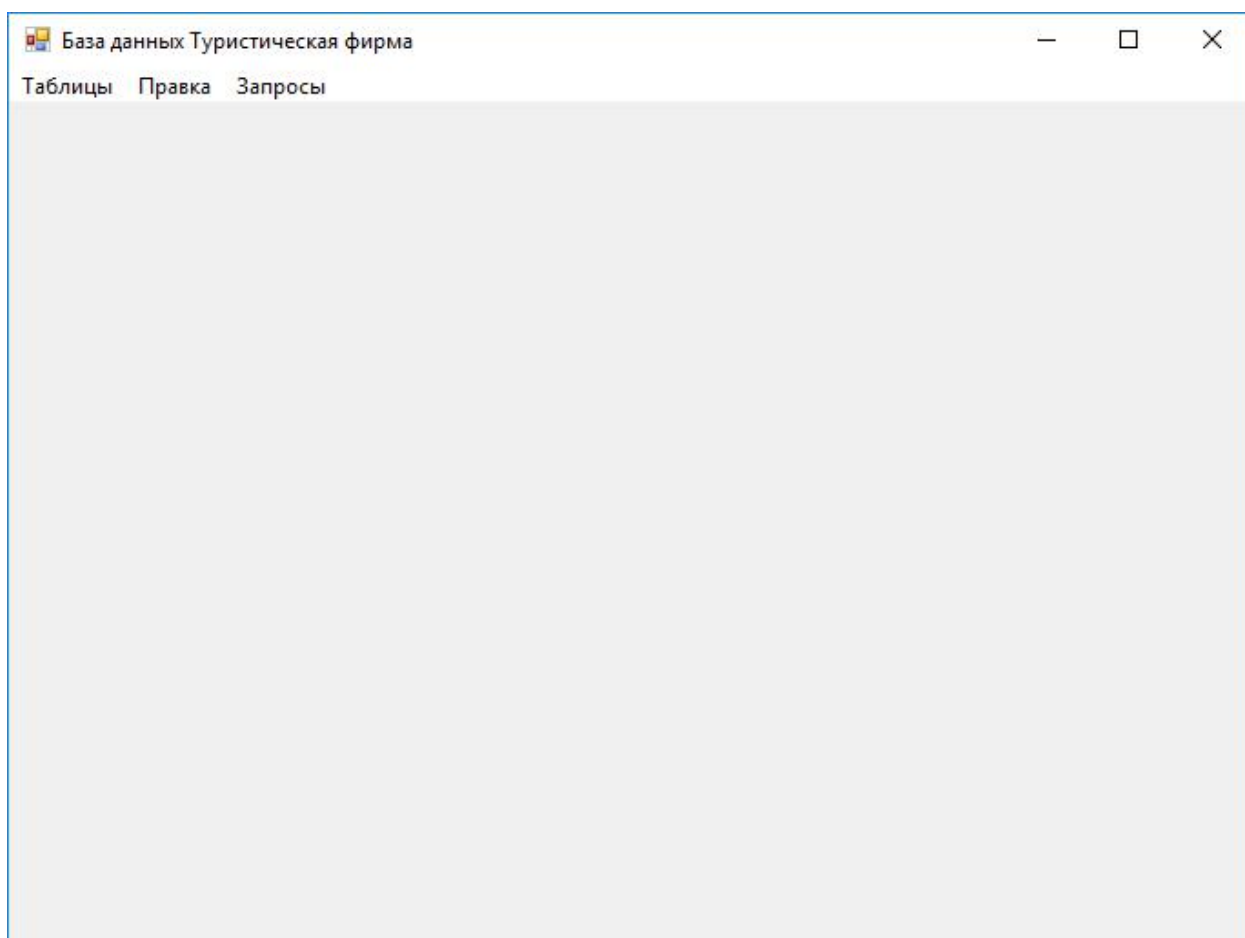


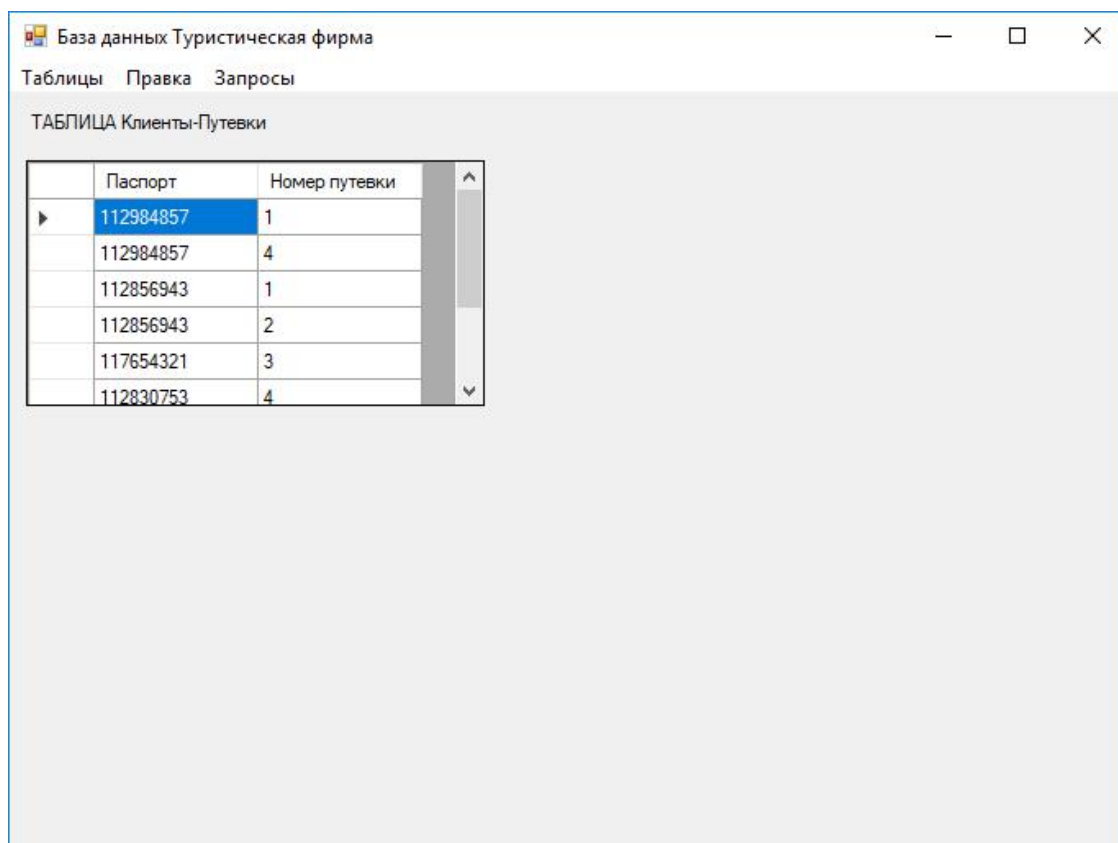
Рисунок 31 - главное окно программы

При нажатии на пункт меню «Таблицы», можно выбрать любую таблицу базы данных «Туристическая фирма» и вывести её на экран. Выведем таблицу «Клиенты» (рисунок 32).

	Фамилия	Имя	Отчество	Паспорт	Дата	Замечания
▶	Моисеева	Милана	Геннадьевна	112984857	13.12.2018	нет
	Кондров	Игорь	Витальевич	112856943	05.06.2018	нет
	Романюк	Анастасия	Александровна	112830753	18.12.2017	нет
	Титова	Татьяна	Анатольевна	117654321	17.12.2018	нет
	kldjv	klnd	kjlbdf	123	20.12.2018	kdfjv
	jfgk	kjfv	kplv	856		pvikej
	Бусс	Ирина	Андреевна	112893485	22.12.2018	нет

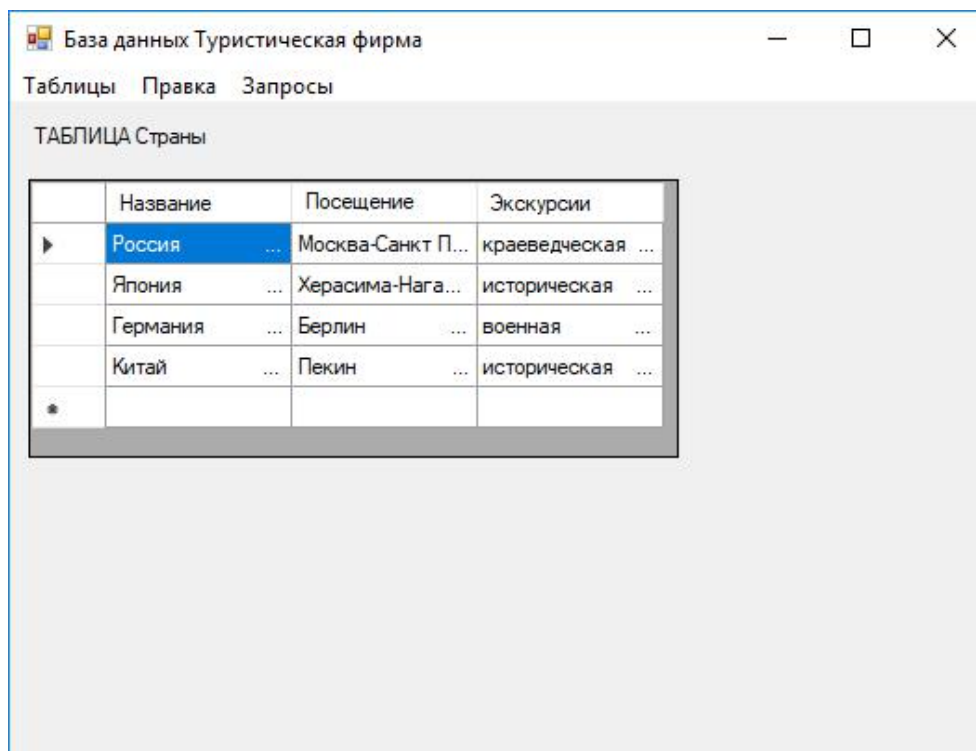
Рисунок 32 - таблица «Клиенты»

Несколько других таблиц, выведенных на экран клиентского приложения, приведены на рисунке 33-35.



	Паспорт	Номер путевки
▶	112984857	1
	112984857	4
	112856943	1
	112856943	2
	117654321	3
	112830753	4

Рисунок 33 - таблица «Клиенты-Путёвки»



	Название	Посещение	Экскурсии
▶	Россия	Москва-Санкт П...	краеведческая ...
	Япония	Херасима-Нага...	историческая ...
	Германия	Берлин	военная ...
	Китай	Пекин	историческая ...
*			

Рисунок 34 - таблица «Страны»

	Тип визы	Срок действия	Номер	Паспорт
▶	шенген	03.03.2019	1	112830753
	долгосрочная	18.12.2019	2	112856943
	краткосрочная	31.12.2018	3	112984857
✱				

Рисунок 35 - таблица «Виза»

При работе с базой данных в приложении пользователь может добавить данные в таблицу. Пример добавления приведён на рисунке 36.

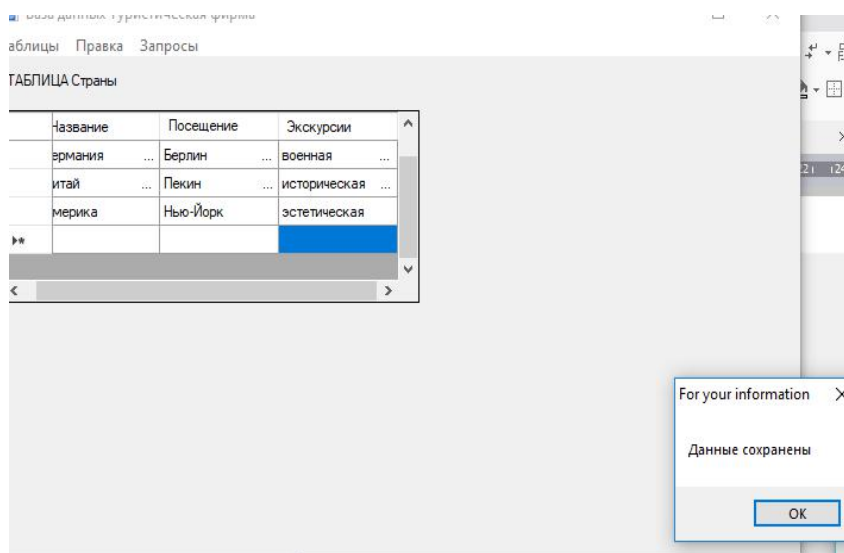


Рисунок 36 - сохранение изменённых данных

При использовании приложения, пользователь может внести изменения в базу данных (изменить, удалить) , при этом сохранив их. Сохранённые данные распространяются и на саму базу данных PostgreSQL.

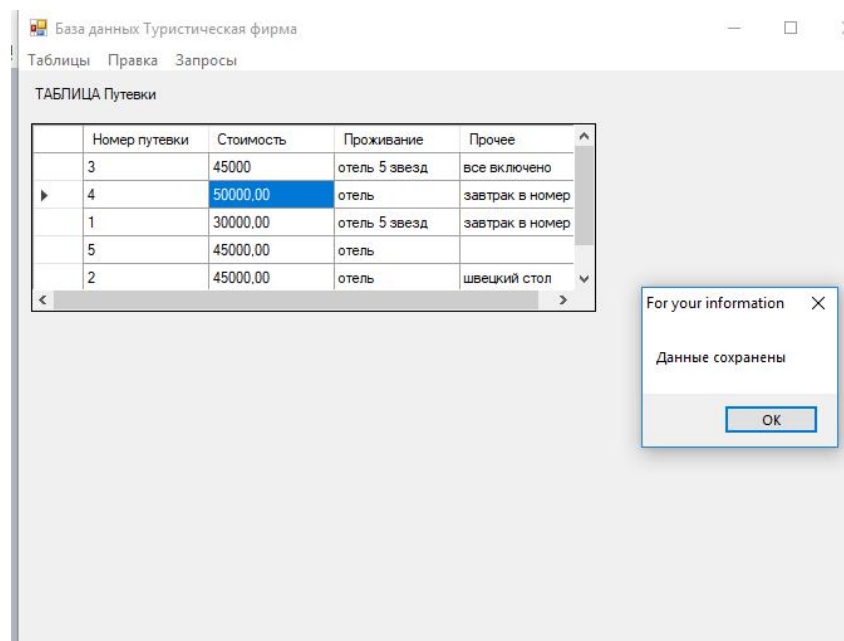


Рисунок 37 - обновление данных

Также в клиентском приложении можно выполнить любой запрос из предметной области «Туристическая фирма». Пример выполнения запроса «Какие есть путёвки по цене не больше той, которую указал клиент» приведён на рисунке 38.

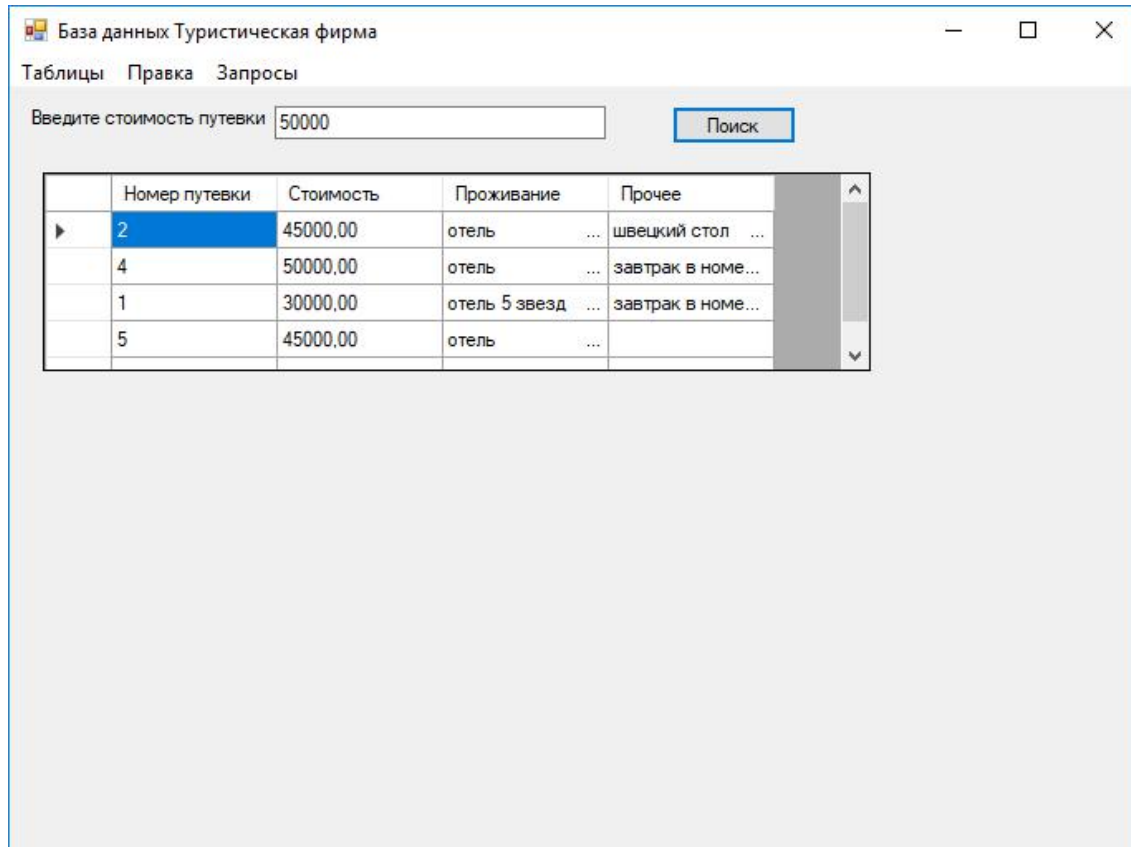
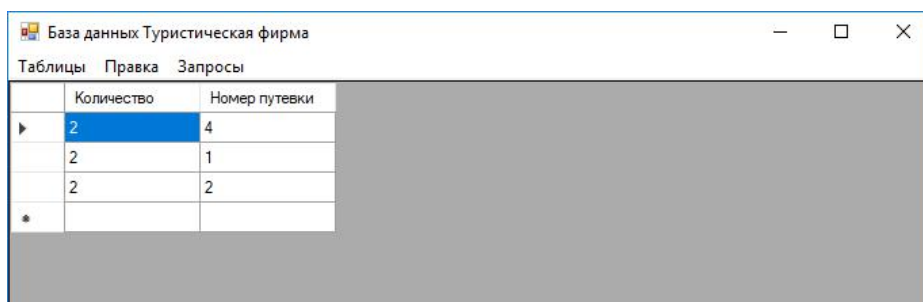


Рисунок 38 - выполнение запроса

В клиентском приложении реализованы запросы с без параметров. При выполнении запроса «Какие путёвки пользуются наибольшим спросом». Данный пример приведён на рисунке 39.



	Количество	Номер путевки
▶	2	4
	2	1
	2	2
*		

Рисунок 39 - выполнение запроса без параметров

Остальные запросы можно запустить в пункте меню «Запросы» (рисунок 40).

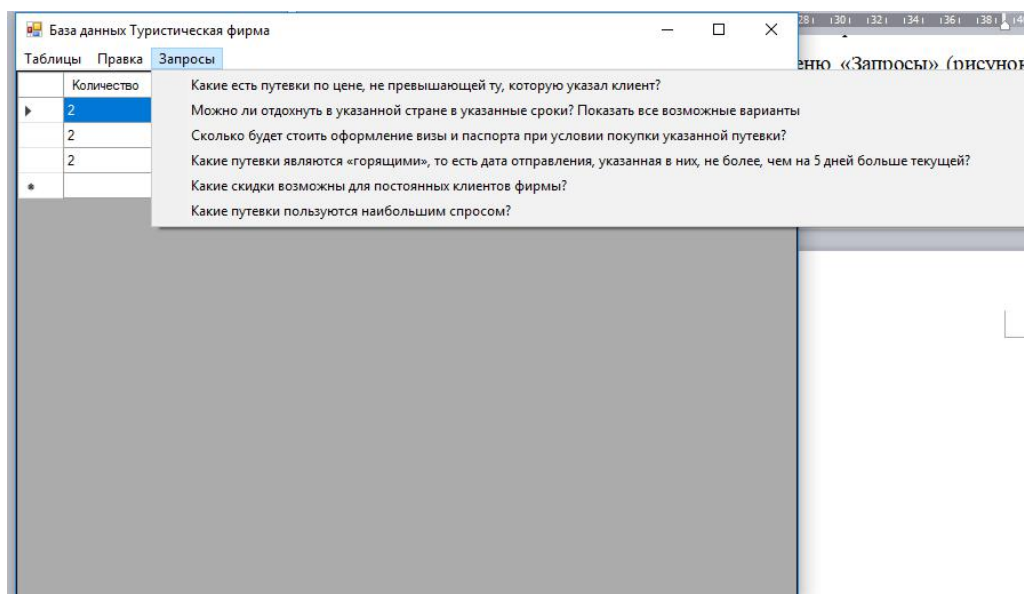


Рисунок 40 - запросы в клиентском приложении

Так как путёвки могут иметь много маршрутов, то мы не можем оставить маршрут без единой путёвки. Пример данного триггера изображён на рисунке 41.

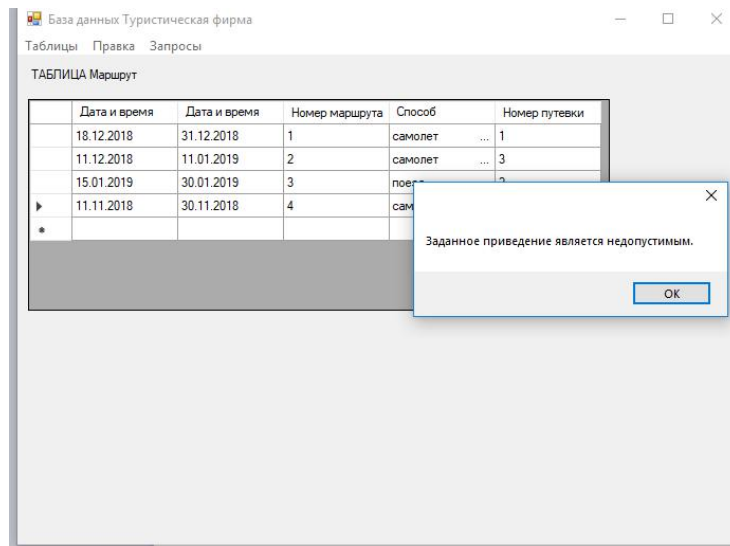


Рисунок 41 - пример удаления единственной путёвки маршрута 5

Клиент не может иметь больше одной визы. Проверим данный триггер в приложении:

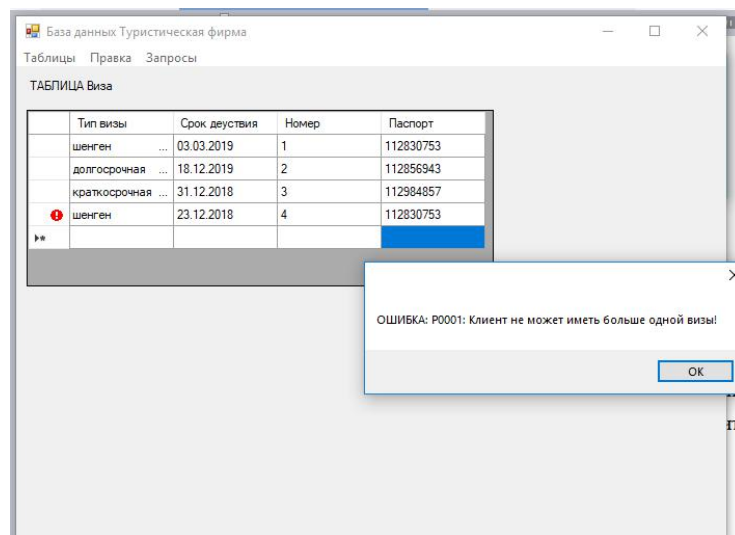


Рисунок 42 - вставка новой визы клиенту, уже имеющему визу

Бизнес-правило туристической фирмы гласит: стоимость путёвки не должна быть меньше 30000 рублей.

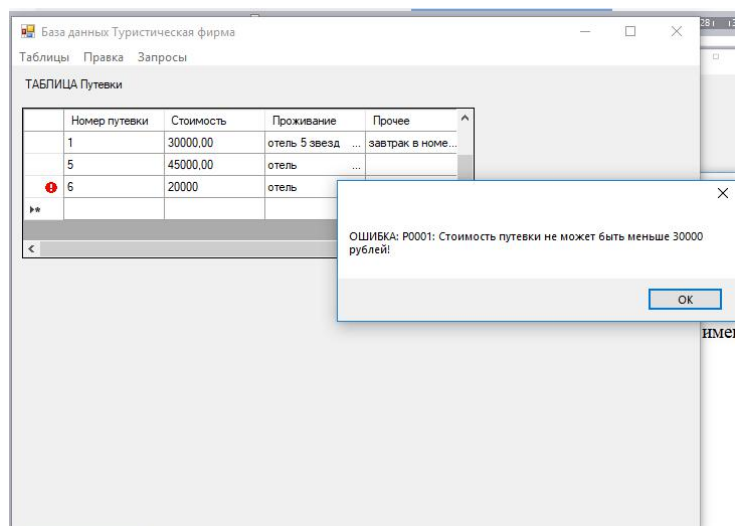


Рисунок 43 - работа триггера

Заключение

В процессе создания курсовой работы была спроектирована база данных для предметной области «Туристическая фирма». Была создана логическая и реляционная модель базы данных. На основе логической модели генерировался скрипт и на основе уже него создавалась основная база данных.

Для дальнейшей работы с базой данных она была заполнена данными, для неё были разработаны запросы, представления, функции и триггеры. Также для работы с ней было разработано клиентское приложение на языке программирования C#.

Список использованных источников

1. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304с.
2. Тиори Т., Фрай Дж. Проектирование структур баз данных: В 2-х кн. Кн. 1. Пер. с англ. – М.: Мир, 1985. – 287с.
3. Чамберлин Д.Д., Астрахан М.М., Эсваран К.П., Грифитс П.П., Лори Р.А., Мел Д.В., Райшер П., Вейд Б.В. SEQUEL 2: унифицированный подход к определению, манипулированию и контролю данных //СУБД. - 1996. - №1. - С.144-159.
4. Чаудхари С. Методы оптимизации запросов в реляционных системах //СУБД. - 1998. - №3. - С.22-36.
5. Чен П. Модель "сущность-связь" - шаг к единому представлению о данных //СУБД. - 1995. - №3. - С.137-158.

Приложение А

/*

Created: 04.12.2018

Modified: 04.12.2018

Model: tur2

Database: PostgreSQL 9.5

*/

-- Create tables section -----

-- Table Klienty

```
CREATE TABLE "Klienty"(  
  "Familia" Character(255),  
  "Name" Character(255),  
  "Otchestvo" Character(255),  
  "Pasport" Bigint NOT NULL,  
  "Date" Date,  
  "Zamechania" Character(255),  
  "Skidka" Bigint  
)  
;
```

-- Create indexes for table Klienty

```
CREATE INDEX "IX_has" ON "Klienty" ("Skidka")  
;
```

-- Add keys for table Klienty

```
ALTER TABLE "Klienty" ADD CONSTRAINT "Unique_Identifier1" PRIMARY  
KEY ("Pasport")  
;
```

-- Table Pyteyki

```
CREATE TABLE "Pyteyki"(  
  "Naimenovanie" Bigint NOT NULL,  
  "Stoimost" Numeric(10,2),  
  "Progivanie" Character(255),  
  "prochue" Character(255)  
)  
;
```

-- Add keys for table Pyteyki

```
ALTER TABLE "Pyteyki" ADD CONSTRAINT "Unique_Identifier2" PRIMARY  
KEY ("Naimenovanie")  
;
```

-- Table Zagranpasport

```
CREATE TABLE "Zagranpasport"(  
  "Number" Bigint NOT NULL,  
  "Data_vadachi" Date,  
  "Srok" Date,  
  "Pasport" Bigint NOT NULL  
)  
;
```

-- Create indexes for table Zagranpasport

```
CREATE INDEX "IX_imeut" ON "Zagranpasport" ("Pasport")  
;
```

-- Add keys for table Zagranpasport

```
ALTER TABLE "Zagranpasport" ADD CONSTRAINT "Unique_Identifier3"  
PRIMARY KEY ("Number")  
;
```

-- Table Visa

```
CREATE TABLE "Visa"(  
  "Tipe_visa" Character(255),  
  "Srok_deystvia" Date,  
  "Number_visa" Bigint NOT NULL,  
  "Pasport" Bigint NOT NULL  
)  
;
```

-- Create indexes for table Visa

```
CREATE INDEX "IX_Relation7" ON "Visa" ("Pasport")  
;
```

-- Add keys for table Visa

```
ALTER TABLE "Visa" ADD CONSTRAINT "Unique_Identifier4" PRIMARY  
KEY ("Number_visa")  
;
```

-- Table Country

```
CREATE TABLE "Country"(  
  "Nazvanie" Character(255) NOT NULL,  
  "Poseshenie punktov" Character(255),  
  "Ekskursii" Character(255)  
)  
;
```

-- Add keys for table Country

```
ALTER TABLE "Country" ADD CONSTRAINT "Unique_Identifier5"  
PRIMARY KEY ("Nazvanie")  
;
```

-- Table Oformlenie_visy

```
CREATE TABLE "Oformlenie_visy"(  
  "Chena" Numeric(10,2),  
  "Sroki" Date,  
  "Number_visa" Bigint NOT NULL,  
  "Pasport" Bigint NOT NULL  
)  
;
```

-- Add keys for table Oformlenie_visy

```
ALTER TABLE "Oformlenie_visy" ADD CONSTRAINT "Unique_Identifier6"  
PRIMARY KEY ("Number_visa","Pasport")  
;
```

-- Table Oformlenie_pasporta

```
CREATE TABLE "Oformlenie_pasporta"(  
  "Chena" Numeric(10,2),  
  "Sroki_vypolnenia" Date,  
  "Pasport" Bigint NOT NULL,  
  "Number" Bigint NOT NULL  
)  
;
```

-- Add keys for table Oformlenie_pasporta

```
ALTER TABLE "Oformlenie_pasporta" ADD CONSTRAINT  
"Unique_Identifier7" PRIMARY KEY ("Pasport","Number")  
;
```

-- Table Marshrut

```
CREATE TABLE "Marshrut"(  
  "Data_and_time_otpravleniya" Timestamp,  
  "Date_and_time_pribytiya" Timestamp,  
  "Number_marshruta" Bigint NOT NULL,  
  "Sposob_peremesheniya" Character(255),  
  "Naimenovanie" Bigint NOT NULL  
)  
;
```

-- Create indexes for table Marshrut

```
CREATE INDEX "IX_having" ON "Marshrut" ("Naimenovanie")  
;
```

-- Add keys for table Marshrut

```
ALTER TABLE "Marshrut" ADD CONSTRAINT "Unique_Identifier8"  
PRIMARY KEY ("Number_marshruta")  
;
```

-- Table Skidki

```
CREATE TABLE "Skidki"(  
  "Skidka" Bigint NOT NULL  
)  
;
```

-- Add keys for table Skidki

```
ALTER TABLE "Skidki" ADD CONSTRAINT "Unique_Identifier9" PRIMARY  
KEY ("Skidka")  
;
```

-- Table Entity1

```
CREATE TABLE "Entity1"(  
  "Number_marshruta" Bigint NOT NULL,
```



```
"Nazvanie" Character(255) NOT NULL
)
;
```

-- Add keys for table Entity1

```
ALTER TABLE "Entity1" ADD CONSTRAINT "Unique_Identifier10"
PRIMARY KEY ("Number_marshruta","Nazvanie")
;
```

-- Table Entity2

```
CREATE TABLE "Entity2"(
  "Pasport" Bigint NOT NULL,
  "Naimenovanie" Bigint NOT NULL
)
;
```

-- Add keys for table Entity2

```
ALTER TABLE "Entity2" ADD CONSTRAINT "Key1" PRIMARY KEY
("Pasport","Naimenovanie")
;
```

-- Create foreign keys (relationships) section

```
ALTER TABLE "Klienty" ADD CONSTRAINT "has" FOREIGN KEY ("Skidka")
REFERENCES "Skidki" ("Skidka") ON DELETE RESTRICT ON UPDATE
RESTRICT
;
```

```
ALTER TABLE "Marshrut" ADD CONSTRAINT "having" FOREIGN KEY
("Naimenovanie") REFERENCES "Pyteyki" ("Naimenovanie") ON DELETE
RESTRICT ON UPDATE CASCADE
;
```

```
ALTER TABLE "Zagranpasport" ADD CONSTRAINT "imeut" FOREIGN KEY
("Pasport") REFERENCES "Klienty" ("Pasport") ON DELETE RESTRICT ON
UPDATE CASCADE
;
```

```
ALTER TABLE "Oformlenie_pasporta" ADD CONSTRAINT "moget" FOREIGN
KEY ("Pasport") REFERENCES "Klienty" ("Pasport") ON DELETE RESTRICT
ON UPDATE CASCADE
;
```

```
ALTER TABLE "Visa" ADD CONSTRAINT "Relation7" FOREIGN KEY  
("Pasport") REFERENCES "Klienty" ("Pasport") ON DELETE RESTRICT ON  
UPDATE CASCADE  
;
```

```
ALTER TABLE "Oformlenie_pasporta" ADD CONSTRAINT "Relationship9"  
FOREIGN KEY ("Number") REFERENCES "Zagranpasport" ("Number") ON  
DELETE RESTRICT ON UPDATE RESTRICT  
;
```

```
ALTER TABLE "Oformlenie_visy" ADD CONSTRAINT "relation10" FOREIGN  
KEY ("Number_visa") REFERENCES "Visa" ("Number_visa") ON DELETE  
RESTRICT ON UPDATE RESTRICT  
;
```

```
ALTER TABLE "Oformlenie_visy" ADD CONSTRAINT "Relation11"  
FOREIGN KEY ("Pasport") REFERENCES "Klienty" ("Pasport") ON DELETE  
RESTRICT ON UPDATE CASCADE  
;
```

```
ALTER TABLE "Entity1" ADD CONSTRAINT "Relation12" FOREIGN KEY  
("Number_marshruta") REFERENCES "Marshrut" ("Number_marshruta") ON  
DELETE RESTRICT ON UPDATE CASCADE  
;
```

```
ALTER TABLE "Entity1" ADD CONSTRAINT "Relation13" FOREIGN KEY  
("Nazvanie") REFERENCES "Country" ("Nazvanie") ON DELETE RESTRICT  
ON UPDATE CASCADE  
;
```

```
ALTER TABLE "Entity2" ADD CONSTRAINT "Relationship1" FOREIGN KEY  
("Pasport") REFERENCES "Klienty" ("Pasport") ON DELETE RESTRICT ON  
UPDATE CASCADE  
;
```

```
ALTER TABLE "Entity2" ADD CONSTRAINT "Relationship2" FOREIGN KEY  
("Naimenovanie") REFERENCES "Pyteyki" ("Naimenovanie") ON DELETE  
RESTRICT ON UPDATE CASCADE  
;
```

Приложение Б

Код программы на C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Npgsql;
using System.Data.Common;

namespace kurs
{
    public partial class Form1 : Form
    {
        DataGridView dg;
        NpgsqlDataAdapter da;
        TextBox TextBox1;
        TextBox TextBox2;

        String Connect =
"Server=localhost;Port=5432;User=postgres;Password=user;Database=kurs;";
        public Form1()
        {
            InitializeComponent();
            NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
// Представляет подключение к базе данных SQL Server
            npgSqlConnection.Open();

            this.Text = "База данных Туристическая фирма";
            this.Width = 700;
            this.Height = 500;

            MainMenu mainMenu1 = new MainMenu();
            MenuItem menuItem1 = new MenuItem("Таблицы");
            menuItem1.MenuItems.Add("Клиенты",
                new EventHandler(mainMenu1_Table1_Select));
            menuItem1.MenuItems.Add("Путевки",
                new EventHandler(mainMenu1_Table2_Select));
```

```
menuItem1.MenuItems.Add("Клиенты-Путевки",
    new EventHandler(mainMenu1_Table3_Select));
menuItem1.MenuItems.Add("Маршрут",
    new EventHandler(mainMenu1_Table4_Select));
menuItem1.MenuItems.Add("Страны",
    new EventHandler(mainMenu1_Table5_Select));
menuItem1.MenuItems.Add("Маршрут-Страны",
    new EventHandler(mainMenu1_Table6_Select));
menuItem1.MenuItems.Add("Виза",
    new EventHandler(mainMenu1_Table7_Select));
menuItem1.MenuItems.Add("Загран.паспорт",
    new EventHandler(mainMenu1_Table8_Select));
menuItem1.MenuItems.Add("Оформление загран.паспорта",
    new EventHandler(mainMenu1_Table9_Select));
menuItem1.MenuItems.Add("Оформление визы",
    new EventHandler(mainMenu1_Table10_Select));
```

```
mainMenu1.MenuItems.Add(menuItem1);
```

```
MenuItem menuItem2 = new MenuItem("Правка");
menuItem2.MenuItems.Add("Сохранить изменения",
    new EventHandler(mainMenu1_Save_Select));
menuItem2.MenuItems.Add("Удалить",
    new EventHandler(mainMenu1_Del_Select));
mainMenu1.MenuItems.Add(menuItem2);
```

```
MenuItem menuItem3 = new MenuItem("Запросы");
menuItem3.MenuItems.Add("Какие есть путевки по цене, не
превышающей ту, которую указал клиент?",
    new EventHandler(mainMenu1_Zapros1_Select));
menuItem3.MenuItems.Add("Можно ли отдохнуть в указанной стране в
указанные сроки? Показать все возможные варианты",
    new EventHandler(mainMenu1_Zapros2_Select));
menuItem3.MenuItems.Add("Сколько будет стоить оформление визы и
паспорта при условии покупки указанной путевки?",
    new EventHandler(mainMenu1_Zapros3_Select));
menuItem3.MenuItems.Add("Какие путевки являются «горящими», то
есть дата отправления, указанная в них, не более, чем на 5 дней больше
текущей?",
    new EventHandler(mainMenu1_Zapros4_Select));
menuItem3.MenuItems.Add("Какие скидки возможны для постоянных
клиентов фирмы?",
    new EventHandler(mainMenu1_Zapros5_Select));
menuItem3.MenuItems.Add("Какие путевки пользуются наибольшим
спросом?",
```

```

        new EventHandler(mainMenu1_Zapros6_Select));

        mainMenu1.MenuItems.Add(menuItem3);
        this.Menu = mainMenu1;
    }
    /* private void dataGridView1_DefaultValuesNeeded(object sender,
System.Windows.Forms.DataGridDataRowEventArgs e)
    {
        e.Row.Cells["Pasport"].Value =
Convert.ToInt32(comboBox1.SelectedValue);
    }*/
    private void mainMenu1_Table1_Select(object sender, System.EventArgs e)
// таблица Клиенты
    {
        this.Controls.Clear();

        Label label1 = new Label();
        label1.Text = "ТАБЛИЦА Клиенты";
        label1.Width = 200;
        label1.Location = new Point(10, 10);
        this.Controls.Add(label1);

        dg = new DataGridView();
        dg.Width = 660;
        dg.Height = 190;
        dg.Location = new Point(10, 40);
        this.Controls.Add(dg);

        NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
        npgSqlConnection.Open();
        da = new NpgsqlDataAdapter(@"SELECT * FROM ""Klienty""",
Connect);

        NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);
        cb.GetUpdateCommand();

        DataTable dt = new DataTable();

        da.Fill(dt);

        dg.DataSource = dt;
        dg.Columns[0].HeaderText = "Фамилия";
        dg.Columns[1].HeaderText = "Имя";
        dg.Columns[2].HeaderText = "Отчество";
        dg.Columns[3].HeaderText = "Паспорт";

```

```

        dg.Columns[4].HeaderText = "Дата";
        dg.Columns[5].HeaderText = "Замечания";
        dg.Columns[6].HeaderText = "Скидка";
    }
    private void mainMenu1_Table2_Select(object sender, System.EventArgs e)
// таблица Путевки
    {
        this.Controls.Clear();

        Label label1 = new Label();
        label1.Text = "ТАБЛИЦА Путевки";
        label1.Width = 200;
        label1.Location = new Point(10, 10);
        this.Controls.Add(label1);

        dg = new DataGridView();
        dg.Width = 450;
        dg.Height = 150;
        dg.Location = new Point(10, 40);
        this.Controls.Add(dg);

        NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
        npgSqlConnection.Open();
        da = new NpgsqlDataAdapter(@"SELECT * FROM ""Pyteyki""",
Connect);

        NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

        DataTable dt = new DataTable();

        da.Fill(dt);

        dg.DataSource = dt;
        dg.Columns[0].HeaderText = "Номер путевки";
        dg.Columns[1].HeaderText = "Стоимость";
        dg.Columns[2].HeaderText = "Проживание";
        dg.Columns[3].HeaderText = "Прочее";
    }
    private void mainMenu1_Table3_Select(object sender, System.EventArgs e)
// таблица Клиенты-Путевки
    {
        this.Controls.Clear();

        Label label1 = new Label();
        label1.Text = "ТАБЛИЦА Клиенты-Путевки";

```

```

label1.Width = 200;
label1.Location = new Point(10, 10);
this.Controls.Add(label1);

dg = new DataGridView();
dg.Width = 280;
dg.Height = 150;
dg.Location = new Point(10, 40);
this.Controls.Add(dg);

NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
npgSqlConnection.Open();
da = new NpgsqlDataAdapter(@"SELECT * FROM ""Entity2""",
Connect);

NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

DataTable dt = new DataTable();

da.Fill(dt);

dg.DataSource = dt;
dg.Columns[0].HeaderText = "Паспорт";
dg.Columns[1].HeaderText = "Номер путевки";
}
private void mainMenu1_Table4_Select(object sender, System.EventArgs e)
// таблица Маршрут
{
    this.Controls.Clear();

    Label label1 = new Label();
    label1.Text = "ТАБЛИЦА Маршрут";
    label1.Width = 200;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    dg = new DataGridView();
    dg.Width = 550;
    dg.Height = 200;
    dg.Location = new Point(10, 40);
    this.Controls.Add(dg);

    NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);

```

```

        npgSqlConnection.Open();
        da = new NpgsqlDataAdapter(@"SELECT * FROM ""Marshrut""",
Connect);

        NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

        DataTable dt = new DataTable();

        da.Fill(dt);

        dg.DataSource = dt;
        dg.Columns[0].HeaderText = "Дата и время отправления";
        dg.Columns[1].HeaderText = "Дата и время прибытия";
        dg.Columns[2].HeaderText = "Номер маршрута";
        dg.Columns[3].HeaderText = "Способ перемещения";
        dg.Columns[4].HeaderText = "Номер путевки";
    }
    private void mainMenu1_Table5_Select(object sender, System.EventArgs e)
// таблица Страны
    {
        this.Controls.Clear();

        Label label1 = new Label();
        label1.Text = "ТАБЛИЦА Страны";
        label1.Width = 200;
        label1.Location = new Point(10, 10);
        this.Controls.Add(label1);

        dg = new DataGridView();
        dg.Width = 350;
        dg.Height = 150;
        dg.Location = new Point(10, 40);
        this.Controls.Add(dg);

        NpgSqlConnection npgSqlConnection = new NpgSqlConnection(Connect);
        npgSqlConnection.Open();
        da = new NpgsqlDataAdapter(@"SELECT * FROM ""Country""",
Connect);

        NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

        DataTable dt = new DataTable();

        da.Fill(dt);

```



```

        dg.DataSource = dt;
        dg.Columns[0].HeaderText = "Название";
        dg.Columns[1].HeaderText = "Посещение пунктов ";
        dg.Columns[2].HeaderText = "Экскурсии";
    }
    private void mainMenu1_Table6_Select(object sender, System.EventArgs e)
// таблица Маршрут-Страны
    {
        this.Controls.Clear();

        Label label1 = new Label();
        label1.Text = "ТАБЛИЦА Маршрут-Страны";
        label1.Width = 200;
        label1.Location = new Point(10, 10);
        this.Controls.Add(label1);

        dg = new DataGridView();
        dg.Width = 250;
        dg.Height = 170;
        dg.Location = new Point(10, 40);
        this.Controls.Add(dg);

        NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
        npgSqlConnection.Open();
        da = new NpgsqlDataAdapter(@"SELECT * FROM ""Entity1""",
Connect);

        NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

        DataTable dt = new DataTable();

        da.Fill(dt);

        dg.DataSource = dt;
        dg.Columns[0].HeaderText = "Номер маршрута";
        dg.Columns[1].HeaderText = "Название";
    }
    private void mainMenu1_Table7_Select(object sender, System.EventArgs e)
// таблица Виза
    {
        this.Controls.Clear();

```

```

Label label1 = new Label();
label1.Text = "ТАБЛИЦА Виза";
label1.Width = 200;
label1.Location = new Point(10, 10);
this.Controls.Add(label1);

dg = new DataGridView();
dg.Width = 450;
dg.Height = 170;
dg.Location = new Point(10, 40);
this.Controls.Add(dg);

NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
npgSqlConnection.Open();
da = new NpgsqlDataAdapter(@"SELECT * FROM ""Visa""", Connect);

NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

DataTable dt = new DataTable();

da.Fill(dt);

dg.DataSource = dt;
dg.Columns[0].HeaderText = "Тип визы";
dg.Columns[1].HeaderText = "Срок действия";
dg.Columns[2].HeaderText = "Номер";
dg.Columns[3].HeaderText = "Паспорт";
}
private void mainMenu1_Table8_Select(object sender, System.EventArgs e)
// таблица Загран.паспорт
{
    this.Controls.Clear();

    Label label1 = new Label();
    label1.Text = "ТАБЛИЦА Загран.паспорт";
    label1.Width = 200;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    dg = new DataGridView();
    dg.Width = 450;
    dg.Height = 100;
    dg.Location = new Point(10, 40);
    this.Controls.Add(dg);

```

```

NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
npgSqlConnection.Open();
da = new NpgsqlDataAdapter(@"SELECT * FROM ""Zagranpasport""",
Connect);

NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

DataTable dt = new DataTable();

da.Fill(dt);

dg.DataSource = dt;
dg.Columns[0].HeaderText = "Номер";
dg.Columns[1].HeaderText = "Дата выдачи";
dg.Columns[2].HeaderText = "Срок";
dg.Columns[3].HeaderText = "Паспорт";
}
private void mainMenu1_Table9_Select(object sender, System.EventArgs e)
// таблица оформление Загран.паспорт
{
    this.Controls.Clear();

    Label label1 = new Label();
    label1.Text = "ТАБЛИЦА Оформление заграничного паспорта";
    label1.Width = 200;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    dg = new DataGridView();
    dg.Width = 450;
    dg.Height = 100;
    dg.Location = new Point(10, 40);
    this.Controls.Add(dg);

    NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
    npgSqlConnection.Open();
    da = new NpgsqlDataAdapter(@"SELECT * FROM
""Oformlenie_pasporta""", Connect);

    NpgsqlCommandBuilder cb = new NpgsqlCommandBuilder(da);

    DataTable dt = new DataTable();

```

```

da.Fill(dt);

dg.DataSource = dt;
dg.Columns[0].HeaderText = "Цена";
dg.Columns[1].HeaderText = "Срок выполнения";
dg.Columns[2].HeaderText = "Паспорт";
dg.Columns[3].HeaderText = "Номер";
}
private void mainMenu1_Table10_Select(object sender, System.EventArgs e)
// таблица оформление визы
{
    this.Controls.Clear();

    Label label1 = new Label();
    label1.Text = "ТАБЛИЦА Оформление визы";
    label1.Width = 200;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    dg = new DataGridView();
    dg.Width = 450;
    dg.Height = 100;
    dg.Location = new Point(10, 40);
    this.Controls.Add(dg);

    NpgsqlConnection npgSqlConnection = new NpgsqlConnection(Connect);
    npgSqlConnection.Open();
    da = new NpgsqlDataAdapter(@"SELECT * FROM ""Oformlenie_visy"",
Connect);

    DataTable dt = new DataTable();

    da.Fill(dt);

    dg.DataSource = dt;
    dg.Columns[0].HeaderText = "Цена";
    dg.Columns[1].HeaderText = "Срок";
    dg.Columns[2].HeaderText = "Номер";
    dg.Columns[3].HeaderText = "Паспорт";
}
private void mainMenu1_Save_Select(object sender, System.EventArgs e)
{
    try

```

```

    {
        da.Update((DataTable)dg.DataSource);
        MessageBox.Show
            ("Данные сохранены", "For your information",
MessageButtons.OK);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private void mainMenu1_Del_Select(object sender, System.EventArgs e)
{
    DataRow current =
((DataRowView)dg.CurrentRow.DataBoundItem).Row;
    current.Delete();
}
private void mainMenu1_Zapros1_Select(object sender, System.EventArgs
e)//Какие есть путевки по цене, не превышающей ту, которую указал клиент?
{
    this.Controls.Clear();
    Label label1 = new Label();
    label1.Text = "Введите стоимость путевки :";
    label1.Width = 150;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    TextBox1 = new TextBox();
    TextBox1.Location = new Point(160, 10);
    TextBox1.Width = 200;
    TextBox1.Text = "";
    this.Controls.Add(TextBox1);

    Button button1 = new Button();
    button1.Text = "Поиск";
    button1.Location = new Point(400, 10);
    button1.Click += new EventHandler(button1_Click);
    this.Controls.Add(button1);

    dg = new DataGridView();
    dg.Width = 500;
    dg.Height = 120;
    dg.Location = new Point(20, 50);
    this.Controls.Add(dg);
}

```

```

private void button1_Click(object sender, System.EventArgs e)
{
    dg.Rows.Clear();
    dg.Columns.Clear();

    NpgsqlConnection myConnection = new NpgsqlConnection(Connect);
    myConnection.Open();

    string CommandText = @"select * from ""Pytevkki"" where
""Stoimost""<="" + TextBox1.Text + """;
    NpgsqlCommand myCommand = new NpgsqlCommand(CommandText,
myConnection);
    NpgsqlDataReader myDataReader = myCommand.ExecuteReader();
    try
    {
        int k = 0;
        if (myDataReader.HasRows)
        {
            dg.Columns.Add("Naimenovanie", "Номер путевки");
            dg.Columns.Add("Stoimost", "Стоимость");
            dg.Columns.Add("Progivanie", "Проживание");
            dg.Columns.Add("prochue", "Прочее");

        }
        while (myDataReader.Read())
        {
            k++;
            dg.Rows.Add(myDataReader[0].ToString(),
myDataReader[1].ToString(),
myDataReader[2].ToString(), myDataReader[3].ToString());
        }
        if (k == 0) MessageBox.Show("Ошибка: нет такой путевки");
    }
    catch (Exception ex) { MessageBox.Show("Ошибка"); }
    myConnection.Close();
}

private void mainMenu1_Zapros2_Select(object sender, System.EventArgs e)
//Можно ли отдохнуть в указанной стране в указанные сроки?
{
    this.Controls.Clear();
    Label label1 = new Label();
    label1.Text = "Введите дату отправления в Китай :";
    label1.Width = 200;
    label1.Location = new Point(10, 10);
}

```

```

this.Controls.Add(label1);

TextBox1 = new TextBox();
TextBox1.Location = new Point(210, 10);
TextBox1.Width = 200;
TextBox1.Text = "";
this.Controls.Add(TextBox1);

Button button1 = new Button();
button1.Text = "Поиск";
button1.Location = new Point(410, 10);
button1.Click += new EventHandler(button2_Click);
this.Controls.Add(button1);

dg = new DataGridView();
dg.Width = 500;
dg.Height = 120;
dg.Location = new Point(20, 50);
this.Controls.Add(dg);
}

private void button2_Click(object sender, System.EventArgs e)
{
    dg.Rows.Clear();
    dg.Columns.Clear();

    NpgsqlConnection myConnection = new NpgsqlConnection(Connect);
    myConnection.Open();

    string CommandText = @"select
""Country"". ""Nazvanie"", ""Ekskursii"", ""Marshrut"". ""Data_and_time_otpravlen
ia"", ""Date_and_time_pribytia"", ""Sposob_peremeshenia""
from ""Country"", ""Marshrut"", ""Entity1""
where ""Country"". ""Nazvanie""=""Entity1"". ""Nazvanie"" and
""Marshrut"". ""Number_marshruta""=""Entity1"". ""Number_marshruta""
and ""Entity1"". ""Nazvanie""='Китай'and
""Marshrut"". ""Data_and_time_otpravlenia""= " + TextBox1.Text + "";

    NpgsqlCommand myCommand = new NpgsqlCommand(CommandText,
myConnection);
    NpgsqlDataReader myDataReader = myCommand.ExecuteReader();
    try
    {
        int k = 0;
        if (myDataReader.HasRows)
        {

```

```

        dg.Columns.Add("Nazvanie", "Название страны");
        dg.Columns.Add("Ekskursii", "Экскурсии");
        dg.Columns.Add("Data_and_time_otpravleniia", "Дата
отправления");
        dg.Columns.Add("Date_and_time_pribytia", "Дата прибытия");
        dg.Columns.Add("Sposob_peremesheniia", "Способ перемещения");

    }
    while (myDataReader.Read())
    {
        k++;
        dg.Rows.Add(myDataReader[0].ToString(),
myDataReader[1].ToString(),
            myDataReader[2].ToString(), myDataReader[3].ToString(),
            myDataReader[4].ToString());
    }
    if (k == 0) MessageBox.Show("Ошибка: нет такого маршрута");
}
catch (Exception ex) { MessageBox.Show("Ошибка"); }
myConnection.Close();
}

private void mainMenu1_Zapros3_Select(object sender, System.EventArgs
e)//Сколько будет стоить оформление визы и паспорта при условии покупки
указанной путевки?
{
    this.Controls.Clear();
    Label label1 = new Label();
    label1.Text = "Введите номер путевки:";
    label1.Width = 150;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    TextBox1 = new TextBox();
    TextBox1.Location = new Point(160, 10);
    TextBox1.Width = 200;
    TextBox1.Text = "";
    this.Controls.Add(TextBox1);

    Button button1 = new Button();
    button1.Text = "Поиск";
    button1.Location = new Point(400, 10);
    button1.Click += new EventHandler(button3_Click);
    this.Controls.Add(button1);

    dg = new DataGridView();

```



```

        dg.Width = 500;
        dg.Height = 120;
        dg.Location = new Point(20, 50);
        this.Controls.Add(dg);
    }

    private void button3_Click(object sender, System.EventArgs e)
    {
        dg.Rows.Clear();
        dg.Columns.Clear();

        NpgsqlConnection myConnection = new NpgsqlConnection(Connect);
        myConnection.Open();

        string CommandText = @"select

        ""Oformlenie_pasporta"". ""Chena"", ""Oformlenie_visy"". ""Chen"", ""Entity2"". ""
        Naimenovanie"", ""Klienty"". ""Pasport""
        from ""Oformlenie_pasporta"", ""Oformlenie_visy"", ""Entity2"", ""Klienty""
        where ""Oformlenie_pasporta"". ""Pasport""=""Klienty"". ""Pasport"" and
        ""Klienty"". ""Pasport""=""Oformlenie_visy"". ""Pasport""
        and ""Klienty"". ""Pasport""=""Entity2"". ""Pasport"" and
        ""Entity2"". ""Naimenovanie""="" + TextBox1.Text + """;

        NpgsqlCommand myCommand = new NpgsqlCommand(CommandText,
        myConnection);
        NpgsqlDataReader myDataReader = myCommand.ExecuteReader();
        try
        {
            int k = 0;
            if (myDataReader.HasRows)
            {
                dg.Columns.Add("Chena", "Цена паспорта");
                dg.Columns.Add("Chen", "Цена визы");
                dg.Columns.Add("Naimenovanie", "Номер путевки");
                dg.Columns.Add("Pasport", "Паспорт");

            }
            while (myDataReader.Read())
            {
                k++;
                dg.Rows.Add(myDataReader[0].ToString(),
                myDataReader[1].ToString(),
                myDataReader[2].ToString(), myDataReader[3].ToString());
            }
            if (k == 0) MessageBox.Show("Ошибка: нет такой путевки");
        }
    }

```

```

    }
    catch (Exception ex) { MessageBox.Show("Ошибка"); }
    myConnection.Close();
}

private void mainMenu1_Zapros4_Select(object sender, System.EventArgs
e)//Какие путевки являются «горящими», то есть дата отправления, указанная
в них, не более, чем на 5 дней больше текущей?
{
    this.Controls.Clear();
    Label label1 = new Label();
    label1.Text = "Введите текущую дату :";
    label1.Width = 150;
    label1.Location = new Point(10, 10);
    this.Controls.Add(label1);

    TextBox1 = new TextBox();
    TextBox1.Location = new Point(160, 10);
    TextBox1.Width = 200;
    TextBox1.Text = "";
    this.Controls.Add(TextBox1);

    Button button1 = new Button();
    button1.Text = "Поиск";
    button1.Location = new Point(400, 10);
    button1.Click += new EventHandler(button4_Click);
    this.Controls.Add(button1);

    dg = new DataGridView();
    dg.Width = 500;
    dg.Height = 120;
    dg.Location = new Point(20, 50);
    this.Controls.Add(dg);
}

private void button4_Click(object sender, System.EventArgs e)
{
    dg.Rows.Clear();
    dg.Columns.Clear();

    NpgsqlConnection myConnection = new NpgsqlConnection(Connect);
    myConnection.Open();

    string CommandText = @"select
""Data_and_time_otpravlenia"", ""Number_marshruta"", ""Naimenovanie"" from

```

```

""Marshrut"" where ""Data_and_time_otpravlenia""between"" + TextBox1.Text +
""and ""+ TextBox1.Text +""+432000";
NpgsqlCommand myCommand = new NpgsqlCommand(CommandText,
myConnection);
NpgsqlDataReader myDataReader = myCommand.ExecuteReader();
try
{
    int k = 0;
    if (myDataReader.HasRows)
    {
        dg.Columns.Add("Data_and_time_otpravlenia", "Дата и время
отправления");
        dg.Columns.Add("Number_marshruta", "Номер маршрута");
        dg.Columns.Add("Naimenovanie", "Номер путевки");

    }
    while (myDataReader.Read())
    {
        k++;
        dg.Rows.Add(myDataReader[0].ToString(),
myDataReader[1].ToString(),
myDataReader[2].ToString());
    }
    if (k == 0) MessageBox.Show("Ошибка: нет такого маршрута");
}
catch (Exception ex) { MessageBox.Show("Ошибка"); }
myConnection.Close();
}
private void mainMenu1_Zapros5_Select(object sender, System.EventArgs e)
{

    this.Controls.Clear();
    dg = new DataGridView();
    dg.Width = 450;
    dg.Height = 150;
    dg.Dock = DockStyle.Fill;
    this.Controls.Add(dg);

    dg.Rows.Clear();
    dg.Columns.Clear();

    NpgsqlConnection myConnection = new NpgsqlConnection(Connect);
    myConnection.Open();

    string CommandText = @"select ""Skidka"", ""Date""

```

```

        from ""Klienty""
        where ""Date""<='18.12.2017';
        NpgsqlCommand myCommand = new NpgsqlCommand(CommandText,
myConnection);
        NpgsqlDataReader myDataReader = myCommand.ExecuteReader();
        try
        {
            int k = 0;
            if (myDataReader.HasRows)
            {
                dg.Columns.Add("Skidka", "Скидка");
                dg.Columns.Add("Date", "Дата регистрации");
            }
            while (myDataReader.Read())
            {
                k++;
                dg.Rows.Add(myDataReader[0].ToString(),
myDataReader[1].ToString());

            }
            if (k == 0) MessageBox.Show("Ошибка: нет такой путевки");
        }
        catch (Exception ex) { MessageBox.Show("Ошибка"); }
        myConnection.Close();
    }
    private void mainMenu1_Zapros6_Select(object sender, System.EventArgs e)
//Какие путевки пользуются наибольшим спросом?
    {
        this.Controls.Clear();
        dg = new DataGridView();
        dg.Width = 450;
        dg.Height = 150;
        dg.Dock = DockStyle.Fill;
        this.Controls.Add(dg);

        dg.Rows.Clear();
        dg.Columns.Clear();

        NpgsqlConnection myConnection = new NpgsqlConnection(Connect);
        myConnection.Open();

        string CommandText = @"select count(*),""Naimenovanie""
        from ""Entity2""
        group by ""Naimenovanie""
        HAVING count(*)in(

```

```

        select count(*)total
        from ""Entity2""
        group by ""Naimenovanie""
        order by total desc
        limit 1);
        NpgsqlCommand myCommand = new NpgsqlCommand(CommandText,
myConnection);
        NpgsqlDataReader myDataReader = myCommand.ExecuteReader();
        try
        {
            int k = 0;
            if (myDataReader.HasRows)
            {
                dg.Columns.Add("count(*)", "Количество");
                dg.Columns.Add("Naimenovanie", "Номер путевки");
            }
            while (myDataReader.Read())
            {
                k++;
                dg.Rows.Add(myDataReader[0].ToString(),
myDataReader[1].ToString());

            }
            if (k == 0) MessageBox.Show("Ошибка: нет такой путевки");
        }
        catch (Exception ex) { MessageBox.Show("Ошибка"); }
        myConnection.Close();
    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }
}
}

```