|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ |  |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |  |

**Институт информационных технологий**

Институт Информационных технологий (ИТ)

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ (ПИ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

**по дисциплине**

**«Разработка баз данных»**

**тема**

**«Сеть ресторанов быстрого питания»**

Выполнил студент группы ИКБО-30-20 Бессмертных И.В.

Принял Корнеев М.С.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_ 2022г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_ 2022г.

Москва 2022

ОГЛАВЛЕНИЕ

[Глава 1 3](#_Toc118816732)

[Постановка задачи: создание логической модели базы данных. Описание ER-модели, генерация на ее основе реляционной модели данных. 3](#_Toc118816733)

[Выполнение работы 3](#_Toc118816734)

[Глава 2 10](#_Toc118816735)

[Постановка задачи: реализация модели в СУБД. 10](#_Toc118816736)

[Выполнение работы 10](#_Toc118816737)

[Глава 3 13](#_Toc118816738)

[Постановка задачи: заполнение базы данных. 13](#_Toc118816739)

[Выполнение работы 13](#_Toc118816740)

[Глава 4 16](#_Toc118816741)

[Постановка задачи: создать различные запросы на получение данных. Для каждой из операций (исключая деление) нужно показать минимум три запроса. 16](#_Toc118816742)

[Выполнение работы 16](#_Toc118816743)

[Глава 5 26](#_Toc118816744)

[Постановка задачи: создать хранимые процедуры и триггеры для обеспечения серверной части работы с данными. 26](#_Toc118816745)

[Выполнение работы 26](#_Toc118816746)

[Выводы 30](#_Toc118816747)

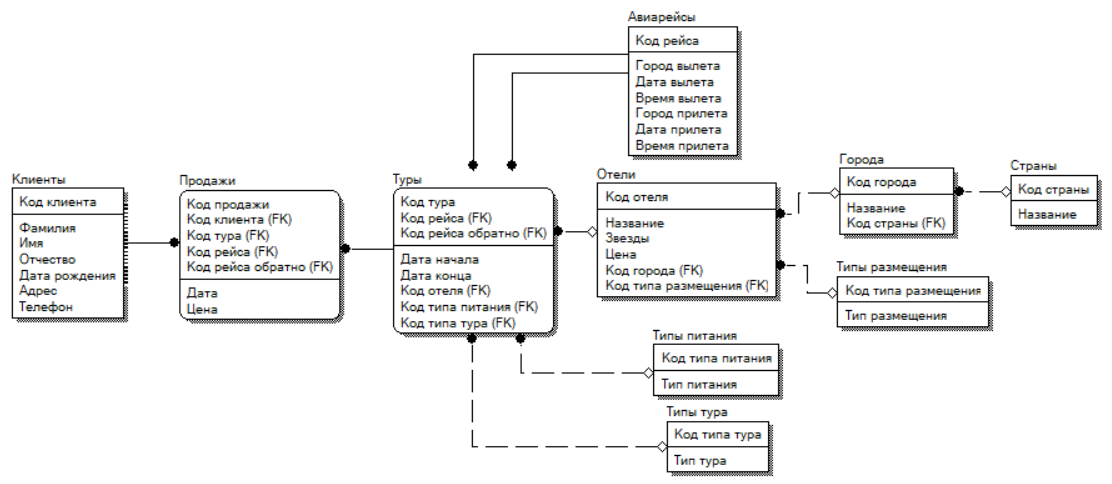
[Список литературы 31](#_Toc118816748)

# Глава 1

## Постановка задачи: создание логической модели базы данных. Описание ER-модели, генерация на ее основе реляционной модели данных.

## Выполнение работы

Создание модели данных начинается с разработки логической модели (Рисунок 1.1), которая должна представлять состав сущностей предметной области с перечнем атрибутов и отношений между ними.



**Рисунок 1.1 – Логическая модель базы данных**

Построим описание логической модели нашей предметной области “Турфирма”. Отразим все сущности в таблицах 1.1 – 1.10.

*Таблица 1.1 – сущность “Клиенты”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код клиента | Первичный ключ клиента |
| Фамилия | Фамилия клиента |
| Имя | Имя клиента |
| Отчество | Отчество клиента |
| Дата рождения | Дата рождения клиента |
| Адрес | Адрес клиента |
| Телефон | Номер телефона клиента |

*Таблица 1.2 – сущность “Продажи”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код продажи | Первичный ключ продажи |
| Код клиента | Внешний ключ, связан с первичным ключом сущности “Клиенты”  Тип связи: One to Many |
| Код тура | Внешний ключ, связан с первичным ключом сущности “Туры”  Тип связи: One to Many |
| Код рейса | Внешний ключ, связан с первичным ключом сущности “Рейсы”  Тип связи: One to Many |
| Код рейса обратно | Внешний ключ, связан с первичным ключом сущности “Рейсы”  Тип связи: One to Many |
| Дата | Дата продажи |
| Цена | Цена продажи |

*Таблица 1.3 – сущность “Туры”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код тура | Первичный ключ тура |
| Код рейса | Внешний ключ, связан с первичным ключом сущности “Рейсы”  Тип связи: One to Many |
| Код рейса | Внешний ключ, связан с первичным ключом сущности “Рейсы”  Тип связи: One to Many |
| Код отеля | Внешний ключ, связан с первичным ключом сущности “Отели”  Тип связи: Не идентифицирующая  Тип связи: One to Many |
| Код типа питания | Внешний ключ, связан с первичным ключом сущности “Типы питания”  Тип связи: Не идентифицирующая  Тип связи: One to Many |
| Код типа тура | Внешний ключ, связан с первичным ключом сущности “Типы тура”  Тип связи: Не идентифицирующая  Тип связи: One to Many |
| Дата начала | Дата, когда начинается тур |
| Дата конца | Дата, когда заканчивается тур |

*Таблица 1.4 – сущность “Авиарейсы”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код рейса | Первичный ключ рейса |
| Город вылета | Город, откуда вылетает самолет |
| Дата вылета | Дата, когда вылетает самолет |
| Время вылета | Время, когда вылетает самолет |
| Город прилета | Город, куда прилетает самолет |
| Дата прилета | Дата, когда прилетает самолет |
| Время прилета | Время, когда прилетает самолет |

*Таблица 1.5 – сущность “Отели”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код отеля | Первичный ключ отеля |
| Код города | Внешний ключ, связан с первичным ключом сущности “Города”  Тип связи: Не идентифицирующая  Тип связи: One to Many |
| Код типа размещения | Внешний ключ, связан с первичным ключом сущности “Типы размещения”  Тип связи: Не идентифицирующая  Тип связи: One to Many |
| Название | Название отеля |
| Звезды | Количество звезд у отеля |
| Цена | Цена номера за сутки |

*Таблица 1.6 – сущность “Типы питания”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код типа питания | Первичный ключ типа питания |
| Тип питания | Название типа питания |

*Таблица 1.7 – сущность “Типы тура”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код типа тура | Первичный ключ типа тура |
| Тип тура | Названия типа тура |

*Таблица 1.8 – сущность “Города”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код горда | Первичный ключ города |
| Код страны | Внешний ключ, связан с первичным ключом сущности “Страны”  Тип связи: Не идентифицирующая  Тип связи: One to Many |
| Название | Название города |

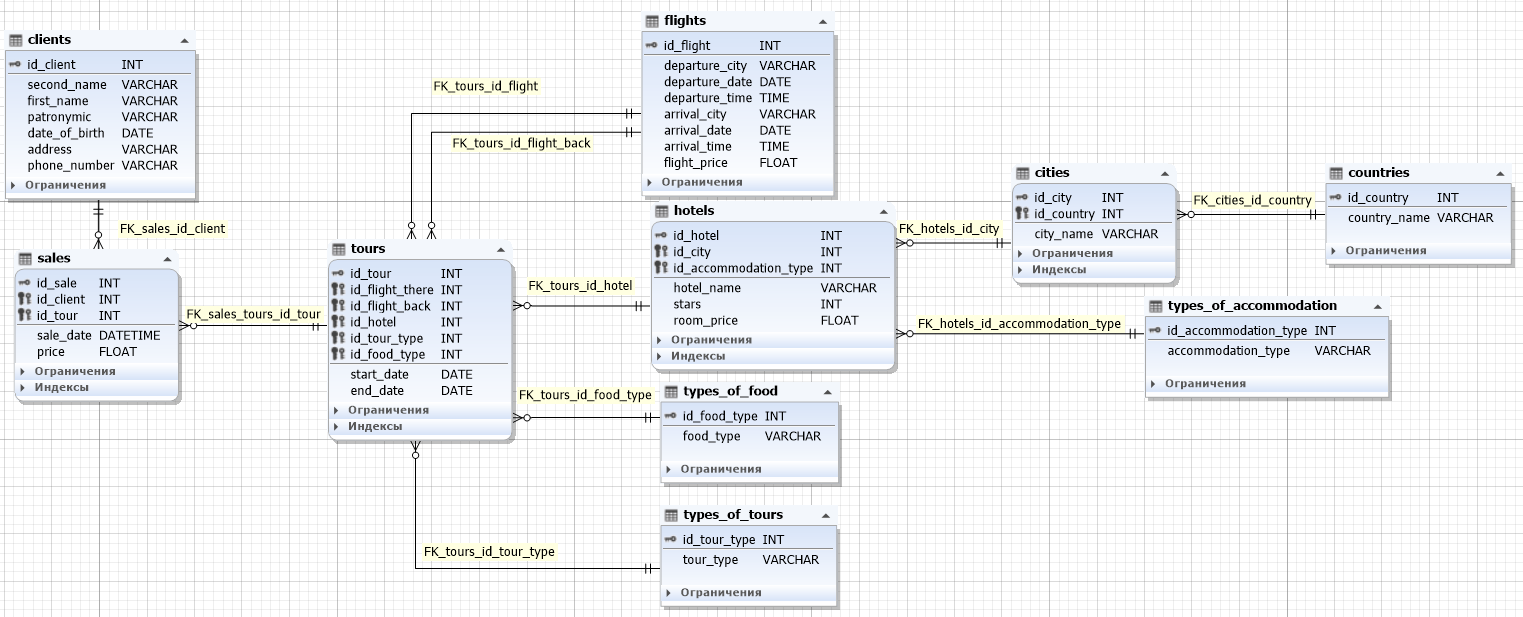
*Таблица 1.9 – сущность “Типы размещения”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код типа размещения | Первичный ключ типа питания |
| Тип размещения | Название типа питания |

*Таблица 1.10 – сущность “Страны”*

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Код страны | Первичный ключ типа тура |
| Название страны | Названия типа тура |

Создание физической модели является вторым шагом построения модели данных системы. В ней уже прописываются типы данных. Физический уровень представления модели зависит от конкретной реализации СУБД, поэтому необходимо предварительно осуществить ее выбор. Сама модель была реализована в dbForge Studio (Рисунок 1.2).



**Рисунок 1.2 – физическая модель базы данных**

Опишем типы данных каждой сущности (Таблицы 1.11 – 1.21)

*Таблица 1.11 – сущность “clients”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_client | INT(11) |
| second\_name | VARCHAR(255) |
| first\_name | VARCHAR(255) |
| patronymic | VARCHAR(255) |
| date\_of\_birth | DATE |
| address | VARCHAR(255) |
| phone\_number | VARCHAR(11) |

*Таблица 1.12 – сущность “sales”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_sale | INT(11) |
| id\_client | INT(11) |
| id\_tour | INT(11) |
| sale\_date | DATETIME |
| price | FLOAT |

*Таблица 1.13 – сущность “tours”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_tour | INT(11) |
| id\_flight\_there | INT(11) |
| id\_flight\_back | INT(11) |
| id\_hotel | INT(11) |
| id\_tour\_type | INT(11) |
| id\_food\_type | INT(11) |
| start\_date | DATE |
| end\_date | DATE |

*Таблица 1.14 – сущность “flights”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_flight | INT(11) |
| departure\_city | VARCHAR(255) |
| departure\_date | DATE |
| departure\_time | TIME |
| arrival\_city | VARCHAR(255) |
| arrival\_date | DATE |
| arrival\_time | TIME |
| flight\_price | FLOAT |

*Таблица 1.15 – сущность “hotels”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_hotel | INT(11) |
| id\_city | INT(11) |
| id\_accommodation\_type | INT(11) |
| hotel\_name | VARCHAR(255) |
| stars | INT(11) |
| room\_price | FLOAT |

*Таблица 1.16 – сущность “types\_of\_food”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_food\_type | INT(11) |
| food\_type | VARCHAR(255) |

*Таблица 1.17 – сущность “type\_of\_tours”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_tour\_type | INT(11) |
| tour\_type | VARCHAR(255) |

*Таблица 1.18 – сущность “cities”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_city | INT(11) |
| id\_country | INT(11) |
| city\_name | VARCHAR(255) |

*Таблица 1.19 – сущность “type\_of\_accommodation”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_accommodation\_type | INT(11) |
| accommodation\_type | VARCHAR(255) |

*Таблица 1.20 – сущность “countries”*

|  |  |
| --- | --- |
| **Свойство** | **Тип данных** |
| id\_country | INT(11) |
| country\_name | VARCHAR(255) |

# 

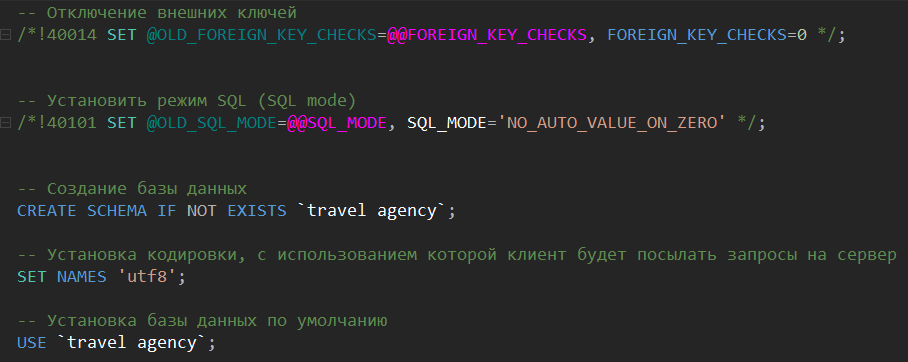
# Глава 2

## Постановка задачи: реализация модели в СУБД.

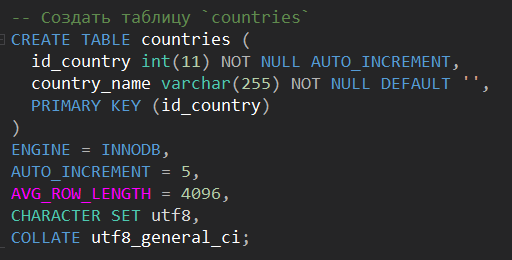
## Выполнение работы

ERD или ER-диаграмма или IDEF1X — это разновидность блок-схемы, где показано, как разные сущности связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

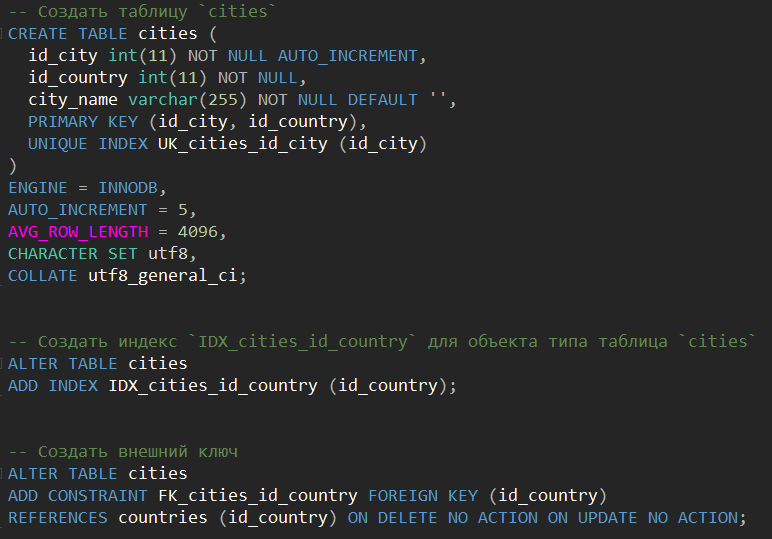
В качестве СУБД выберем MySQL. Возьмем физическую модель из главы 1 и перенесем ее в SQL-скрипт, на основе которого и созданы основные таблицы и связи (Рисунки 2.1 – 2.13).



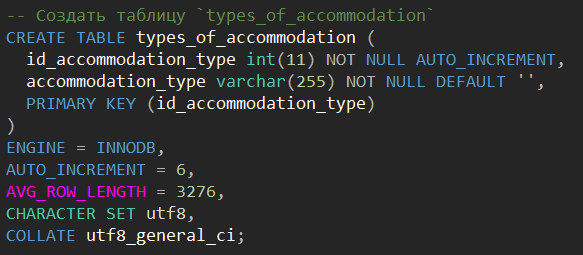
**Рисунок 2.1 – Создание базы данных**



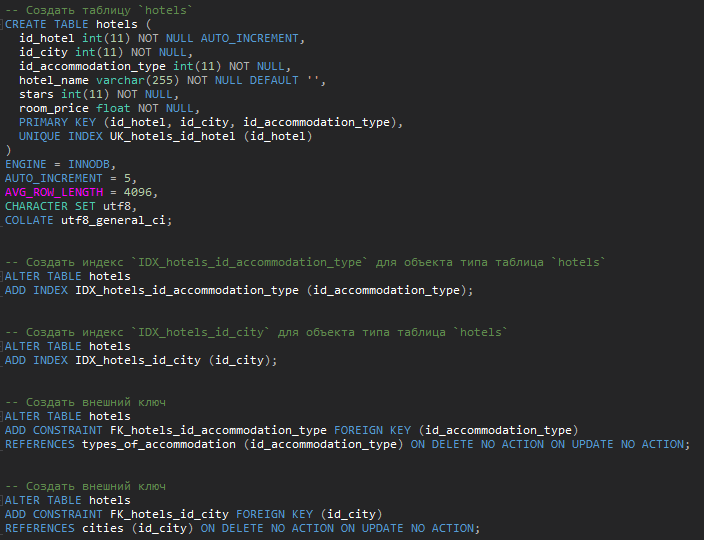
**Рисунок 2.2 – Создание таблицы стран**



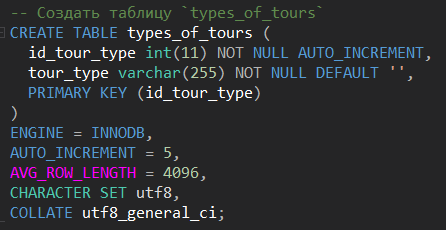
**Рисунок 2.3 – Создание таблицы городов и индексов и внешних ключей для неё**



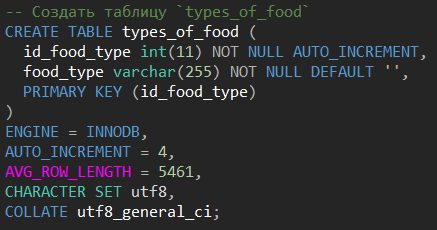
**Рисунок 2.4 – Создание таблицы типов размещения**



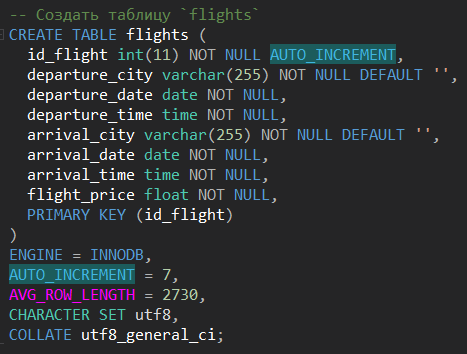
**Рисунок 2.5 – Создание таблицы отелей и индексов и внешних ключей для неё**



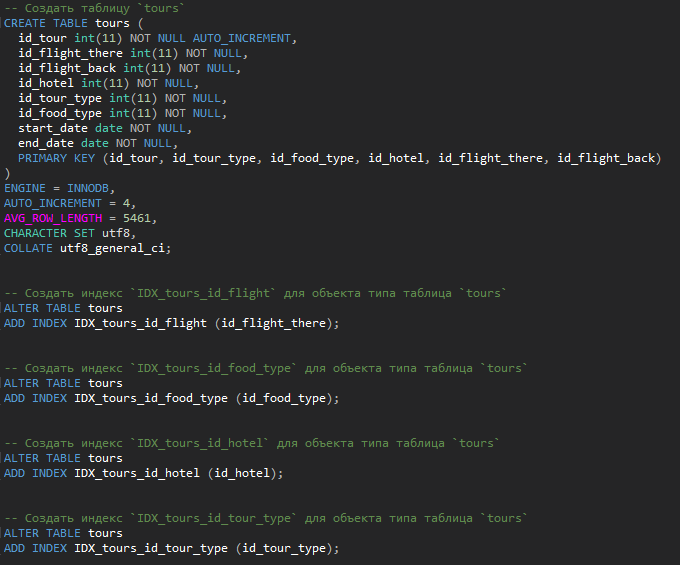
**Рисунок 2.6 – Создание таблицы типов туров**



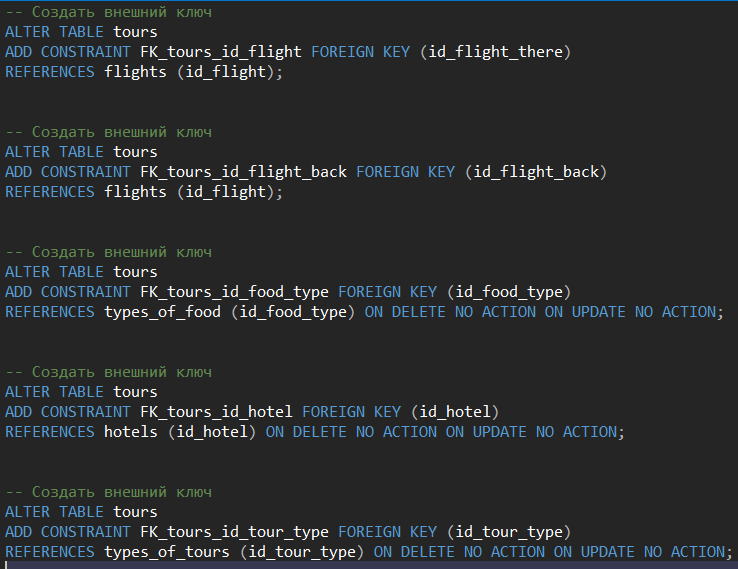
**Рисунок 2.7 – Создание таблицы типов питания**



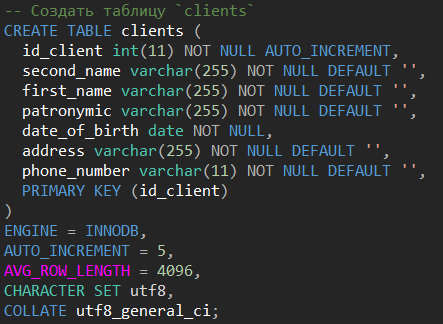
**Рисунок 2.8 – Создание таблицы авиарейсов**



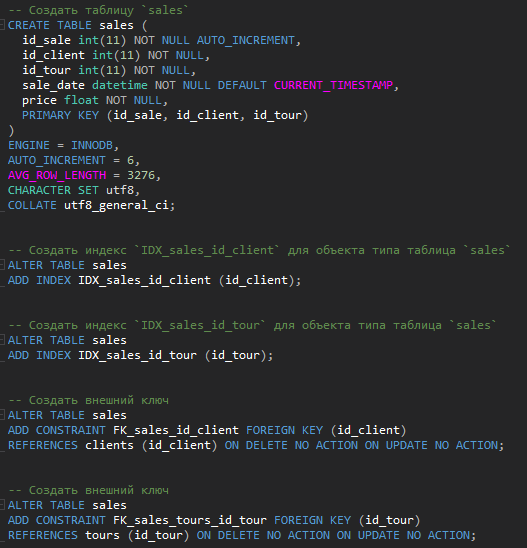
**Рисунок 2.9 – Создание таблицы авиарейсов и индексов для неё**



**Рисунок 2.10 – Создание внешних ключей для таблицы авиарейсов**

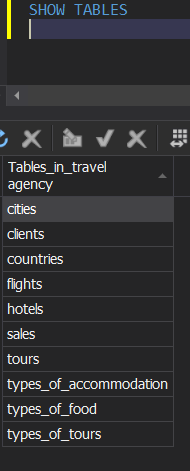


**Рисунок 2.11 – Создание таблицы клиентов**



**Рисунок 2.12 – Создание таблицы продаж**

Выполнив выше представленные скрипты, результатом будем база данных, с представленными ранее таблицами (Рисунок 2.13).



**Рисунок 2.13 - созданные на основе скрипта таблицы**

# Глава 3

## Постановка задачи: заполнение базы данных.

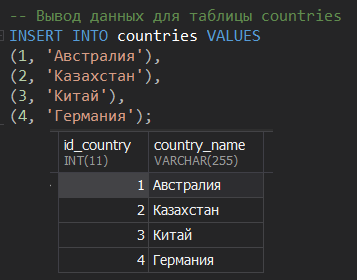
## Выполнение работы

Чтобы заполнить таблицы базы данных, воспользуемся командой *INSERT INTO ‘table\_name’ (columns) VALUES (values)*. INSERT добавляет строки в таблицу. Эта команда может добавить одну или несколько строк, сформированных выражениями значений, либо ноль или более строк, выданных дополнительным запросом. Значения, получаемые от предложения VALUES или запроса, связываются с явно или неявно определенным списком столбцов слева направо.

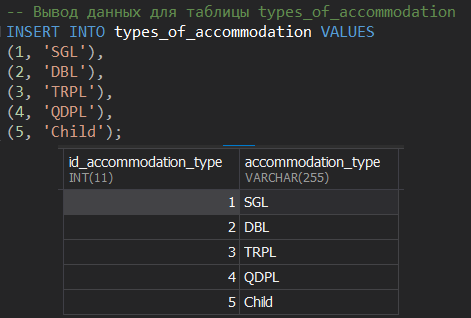
Все столбцы, не представленные в явном или неявном списке столбцов, получат значения по умолчанию, если для них заданы эти значения, либо NULL в противном случае.

Если выражение для любого столбца выдаёт другой тип данных, система попытается автоматически привести его к нужному.

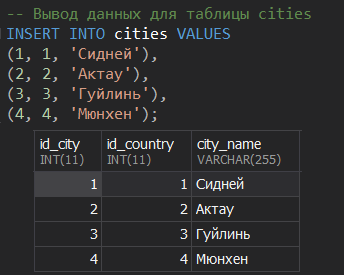
Используя выше представленные команды, получим заполненные таблицы базы данных (Рисунки 3.1 – 3.)



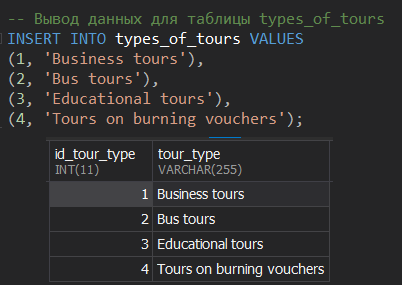
**Рисунок 3.1 – Заполненная таблица стран**



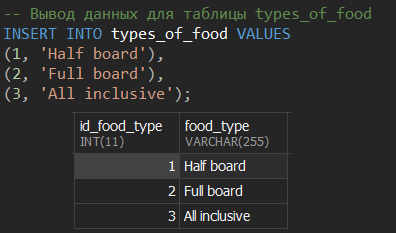
**Рисунок 3.2 – Заполненная таблица типов размещения**



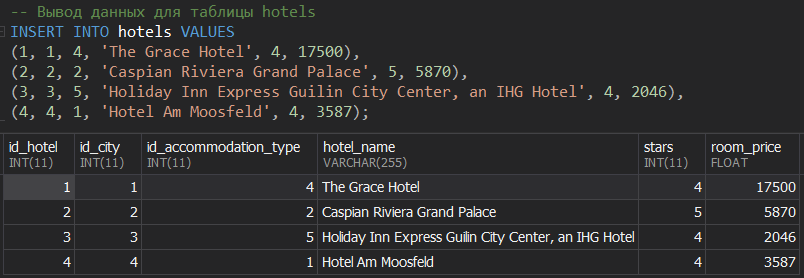
**Рисунок 3.3 – Заполненная таблица городов**



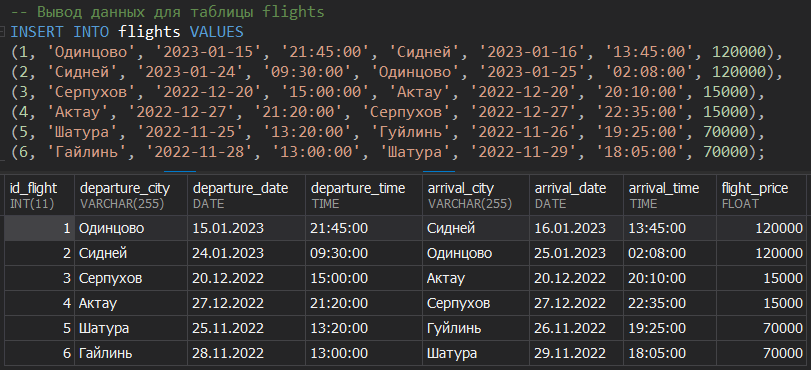
**Рисунок 3.4 – Заполненная таблица типов туров**



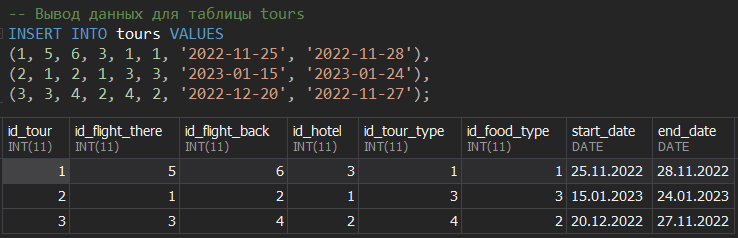
**Рисунок 3.5 – Заполненная таблица типов питания**



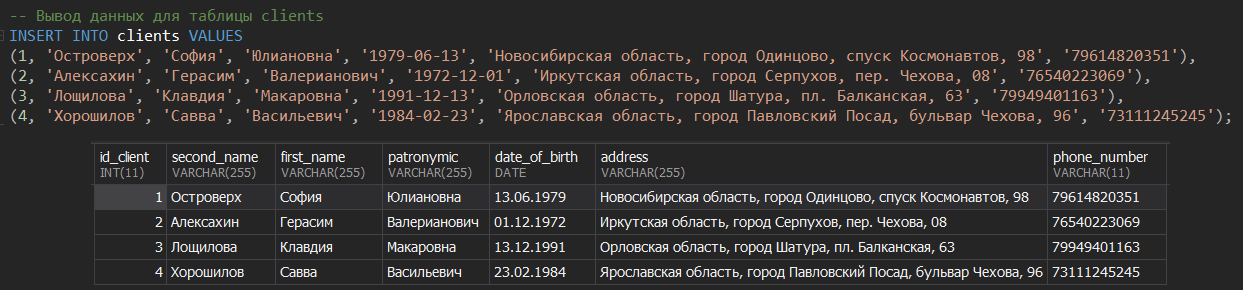
**Рисунок 3.6 – Заполненная таблица отелей**



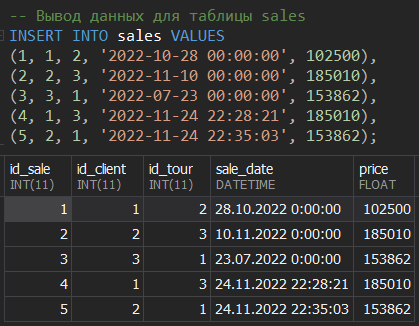
**Рисунок 3.7 – Заполненная таблица авиарейсов**



**Рисунок 3.8 – Заполненная таблица туров**



**Рисунок 3.9 – Заполненная таблица клиентов**



**Рисунок 3.10 – Заполненная таблица продаж**

# Глава 4

## Постановка задачи: создать различные запросы на получение данных.

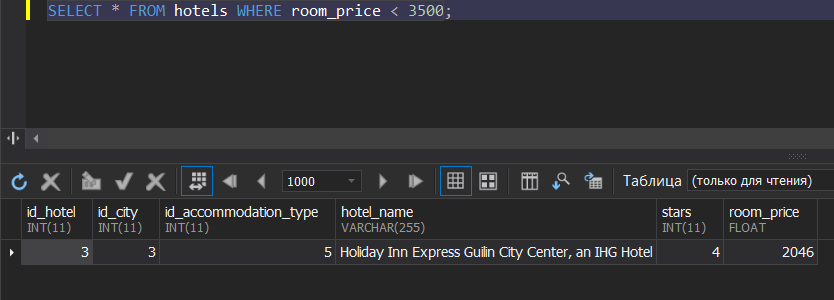
## Выполнение работы

Одной из основных задач, которую решают базы данных – это эффективный поиск необходимых данных. Универсальным подходом для решения этой задачи является применение специального оператора языка SQL - SELECT. Этот оператор достаточно сложный, имеет множество возможностей. Теоретической основой этого оператора является реляционная алгебра, которая доказывает возможность получения с помощью конечного набора операций любых возможных наборов данных.

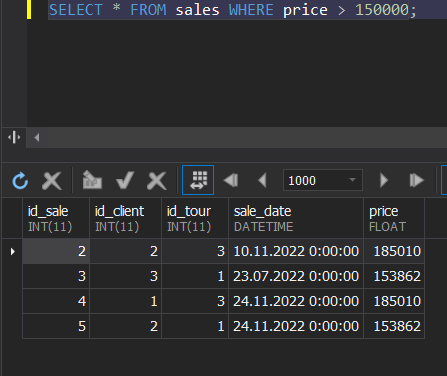
Приведем несколько примеров запросов к таблицам базы данных, демонстрирующих различные возможности языка SQL и различные операции реляционной алгебры.

**Операции проекции и селекции**

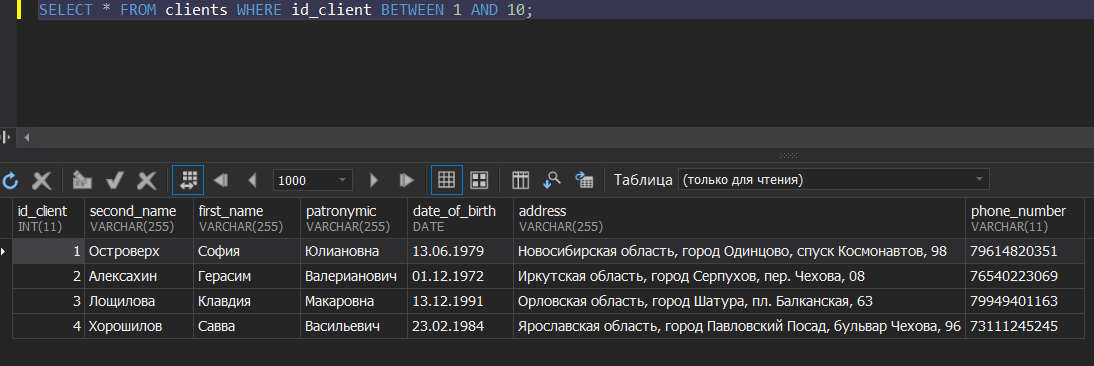
При проекции, осуществляется выбор только части полей таблицы, а при селекции осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию. Создадим несколько запросов и результаты выполнения запросов отображены на рисунках 4.1 - 4.4.



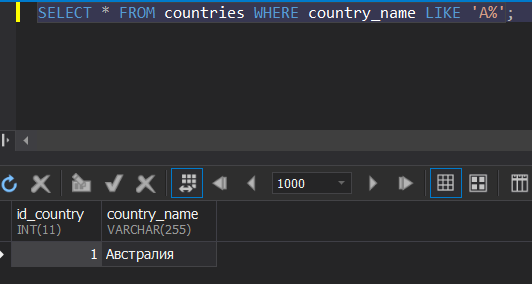
**Рисунок 4.1 – Выбор всех записей из таблицы hotels где цена комнаты за сутки меньше 3500**



**Рисунок 4.2 - Выбор всех записей из таблицы sales, где цена тура больше 150000**



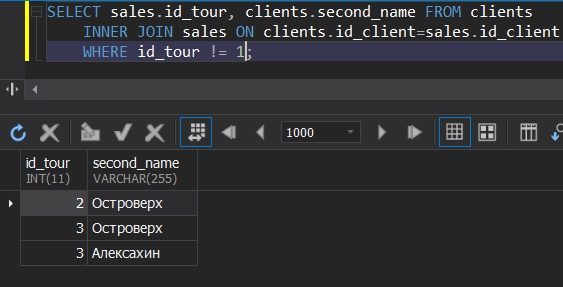
**Рисунок 4.3 - Выбор всех клиентов с id между 1 и 10**

****

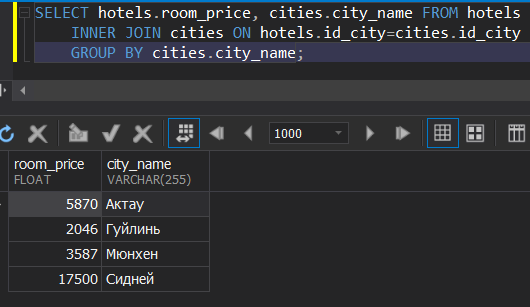
**Рисунок 4.4 - Выбор всех записей из таблицы countries, где страны начинаются на букву «А»**

**Операция объединения**

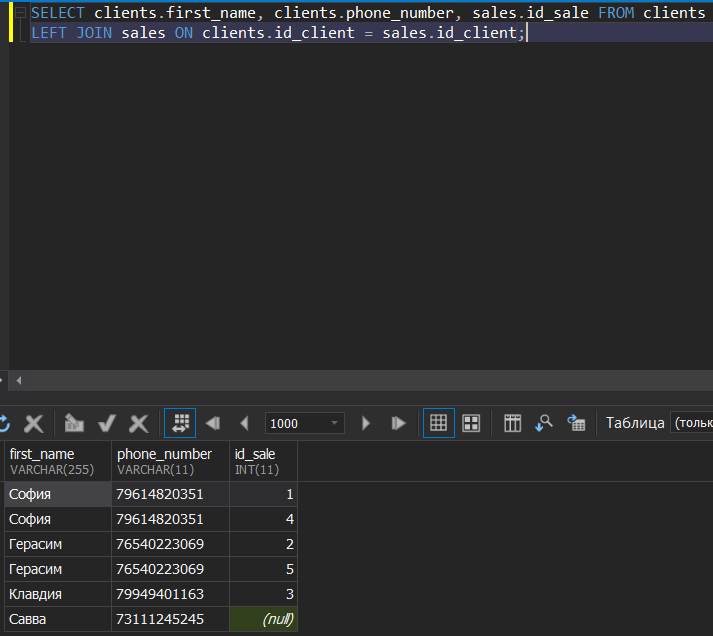
Теоретико-множественные операции часто можно записать с помощью логических операций, примененных в конструкции WHERE запроса. Объединение можно задать с помощью логического ИЛИ. Создадим несколько запросов и результаты выполнения запросов отображены на рисунках 4.5 - 4.7.



**Рисунок 4.5 - Выбор некоторых записей из таблицы client, объединяя с таблицей sales, где id\_tour не 1**



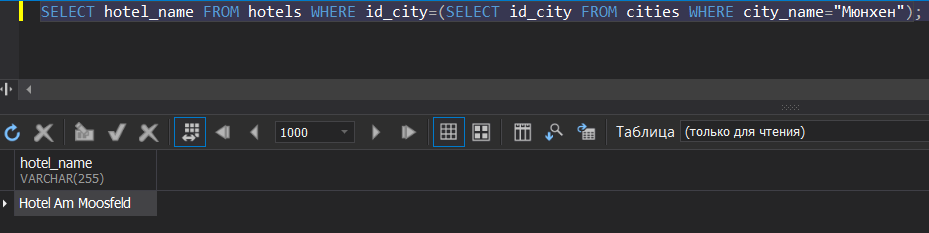
**Рисунок 4.6 - Выбор некоторых записей из таблицы hotels, объединяя с таблицей cities, группируя по названию города**

****

**Рисунок 4.7 - Выбор некоторых записей из таблицы clients, объединяя с таблицей sales**

**Вложенные запросы**

Вложенный запрос — это запрос на выборку, который используется внутри инструкции SELECT, INSERT, UPDATE или DELETE или внутри другого вложенного запроса. Подзапрос может быть использован везде, где разрешены выражения. Результаты представлены на рисунке 4.8.

****

**Рисунок 4.8 – Выбор названия отеля из таблицы hotels, где название города “Мюнхен”**

# Глава 5

## Постановка задачи: создать хранимые процедуры, функции и триггеры для обеспечения серверной части работы с данными.

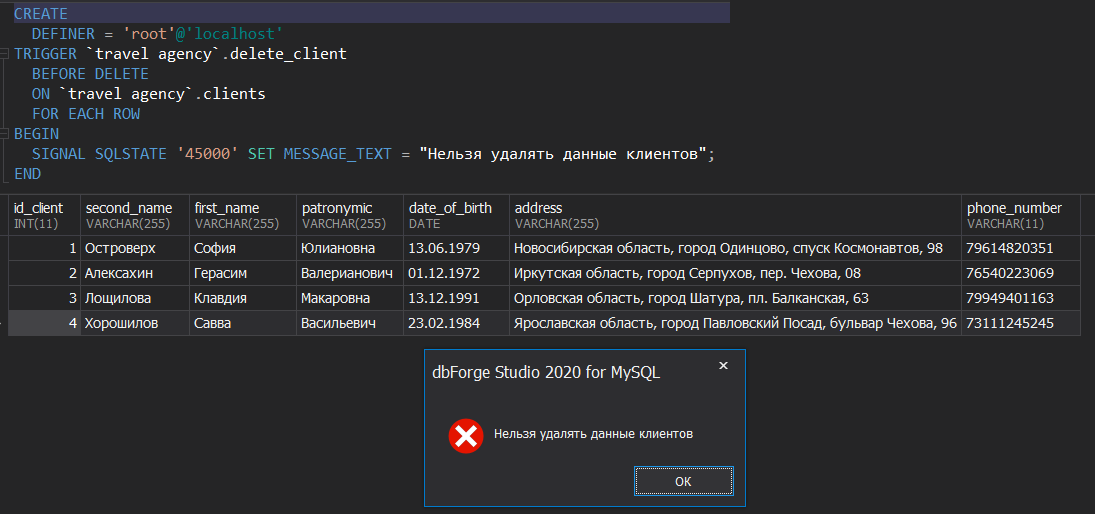
## Выполнение работы

Хранимые процедуры, функции и триггеры вводятся в базу данных для обеспечения бизнес-логики приложения на уровне серверной его компоненты. Обычно хранимые процедуры и функции представляют собой утилиты, которые определенным образом обрабатывают данные или реализуют достаточно сложный алгоритм вычисления некоторых показателей.

Функция – это хранимая программа, в которую вы можете передавать параметры и возвращать значение.

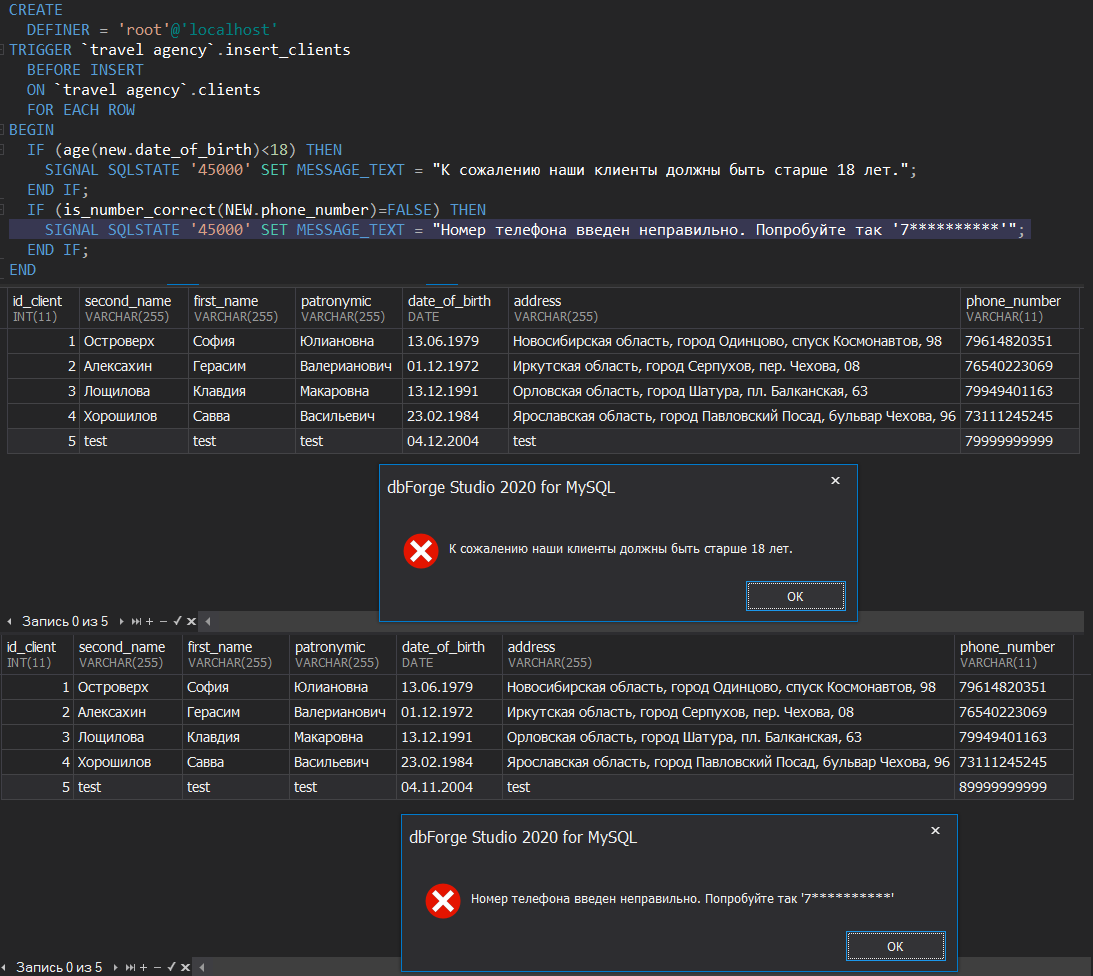
Триггеры – это частный случай хранимой процедуры, который выполняется автоматически при выполнении команд обновления данных (INSERT, DELETE, UPDATE). Триггеры привязываются к конкретным таблицам базы данных. Для каждой команды должны быть свои триггеры.

Создадим триггер, который при попытке удалить клиента из базы данных выдаст ошибку и не даст этого сделать. После создания триггера – протестируем его (Рисунок 5.1).



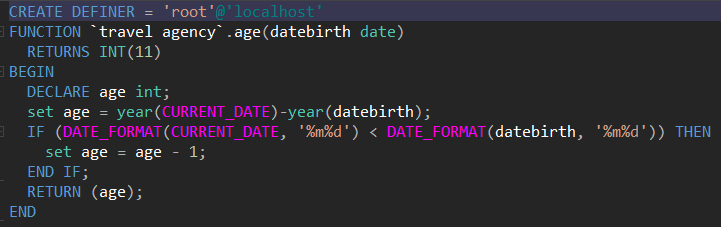
**Рисунок 5.1 – Создание и тестирование триггера**

Следующий триггер сработает при добавлении нового клиента. Он проверит является ли новый клиент совершеннолетним, а также проверит по правильной ли маске введен его номер телефона (Рисунок 5.2).



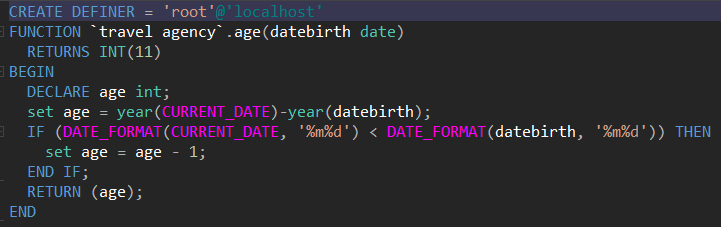
**Рисунок 5.2 – Создание и тестирование триггера**

Данный триггер использует две функции. Рассмотри их по порядку. Функция “age”: в переменную записываем разницу между нынешним годом и годом рождения клиента, который мы ей передали, а затем проверяем был ли у него в этом году день рождения, если не, то вычитаем ещё один год и возвращаем переменную (Рисунок 5.3).



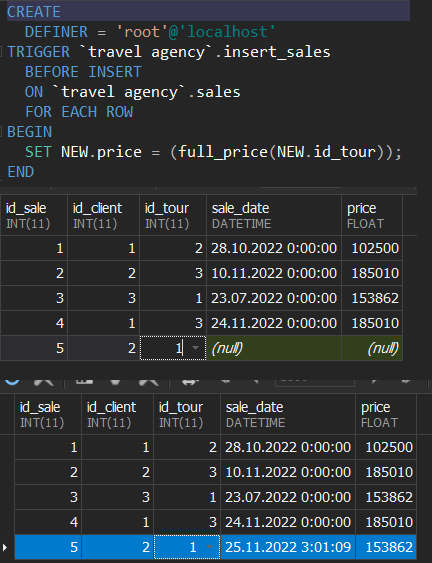
**Рисунок 5.3 – Функция “age”**

Функция “is\_number\_correct”: проверяет начинается ли номер телефона, который мы в неё передаем, с цифры 7 и состоит ли этот номер только из цифр и ничего более, если да, то возвращает true, иначе возвращает false (Рисунок 5.4).



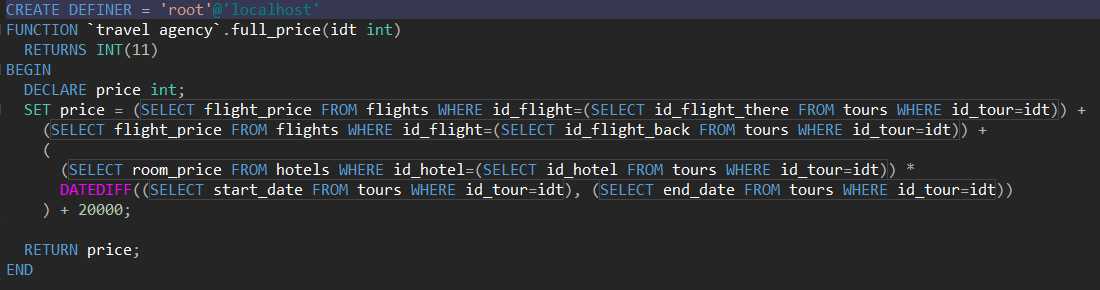
**Рисунок 5.4 – Функция “is\_number\_correct”**

Следующий триггер устанавливает цену при создании продажи, при выборе тура (Рисунок 5.5)



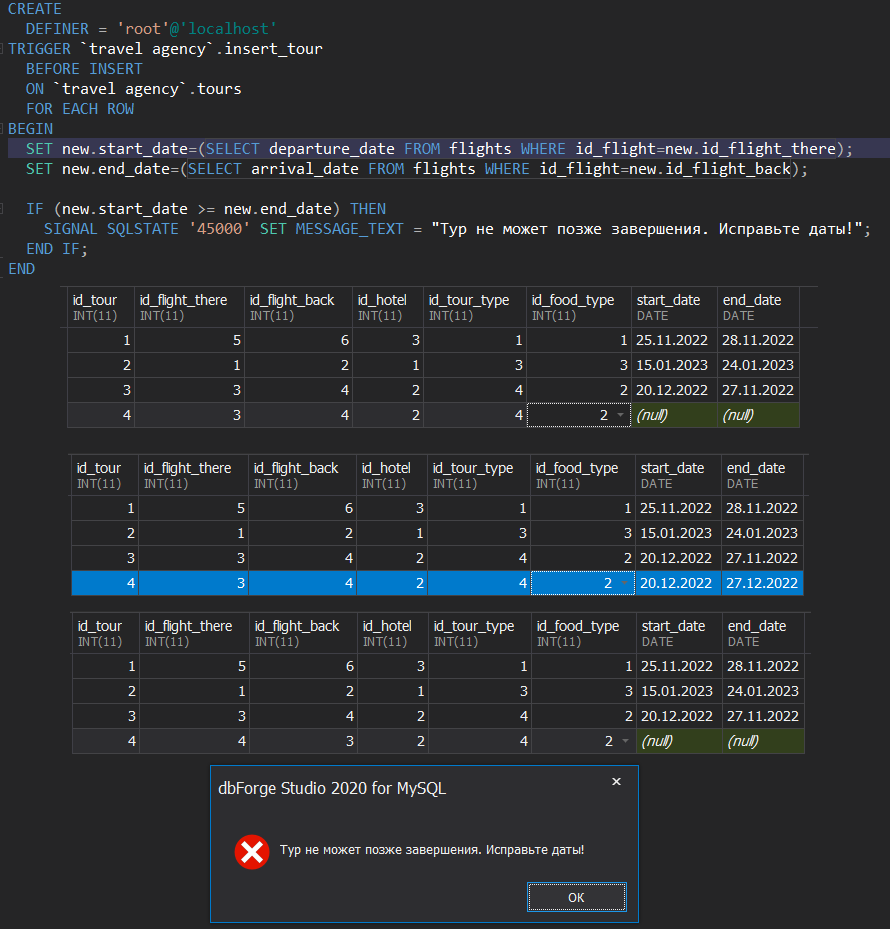
**Рисунок 5.5 – Создание и тестирование триггера**

Он также использует функцию. Функция “full\_price” принимает на вход id тура и с помощью него устанавливает цену. Сначала она добавляет цену билетов за перелет туда и обратно, затем узнает цену комнаты в отеле за сутки и умножает её на количество суток, которое клиент в данном отеле проведет, а затем прибавляет комиссию организации за организацию тура (наша прибыль). В самом конце данная функция возвращает полную цену тура (Рисунок 5.6).



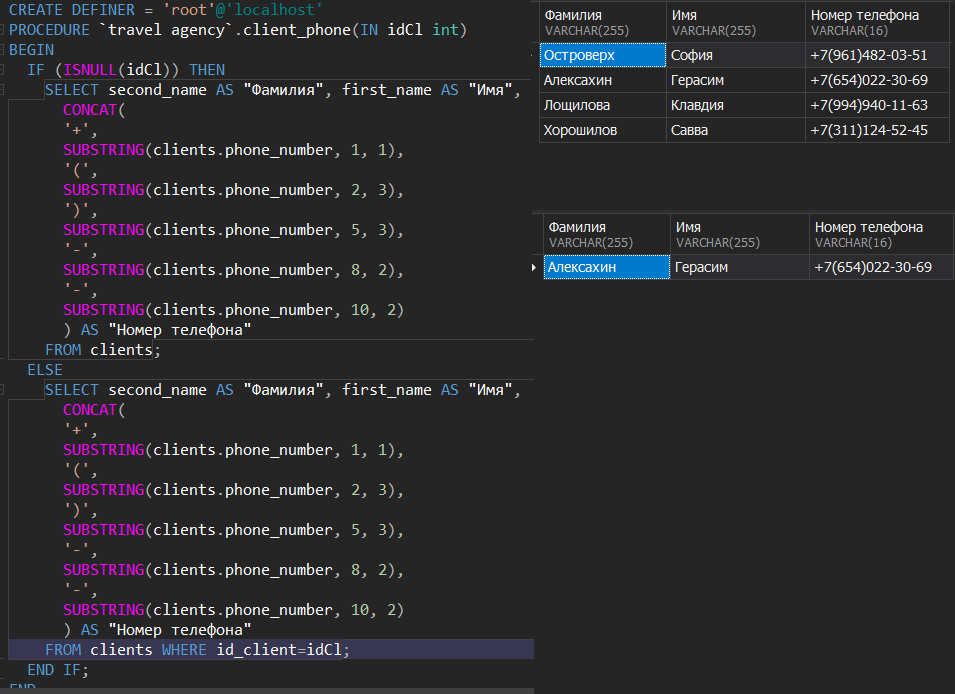
**Рисунок 5.6 – Функция “full\_price”**

Последний триггер устанавливает дату начала и конца тура, при добавлении тура в базу, из даты вылета и даты прилета авиарейсов туда и обратно соответственно. Также данный триггер проверяет правильность введенных дат и если дата старта позже даты завершения тура, то выдает ошибку и не дает сохранить данные (Рисунок 5.7).



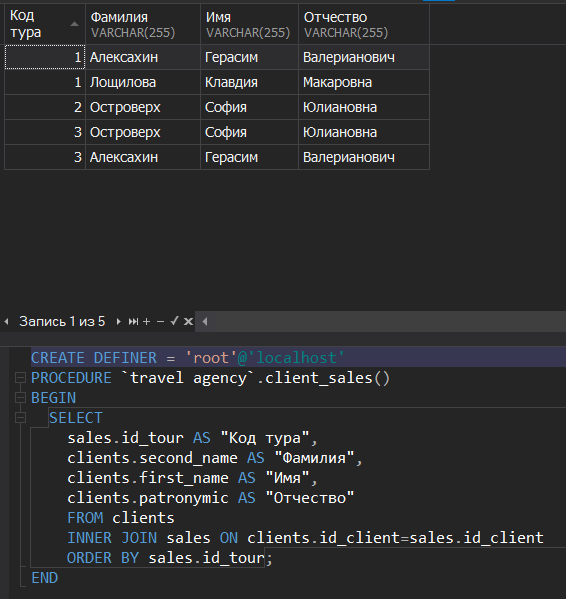
**Рисунок 5.7 – Создание и тестирование триггера**

Теперь займемся процедурами. Процедура “client phone” выводит фамилию и имя клиента, и его номер телефона, но номер она разбивает на регистры и перед определенными регистрами ставит знаки ‘+’, ‘(’, ‘)’ и ‘-’. Так же в данной процедуре предусмотрена фильтрация, если нам нужен конкретный клиент. Мы можем просто передать в процедуру его id и получим информацию только о нем (Рисунок 5.8).



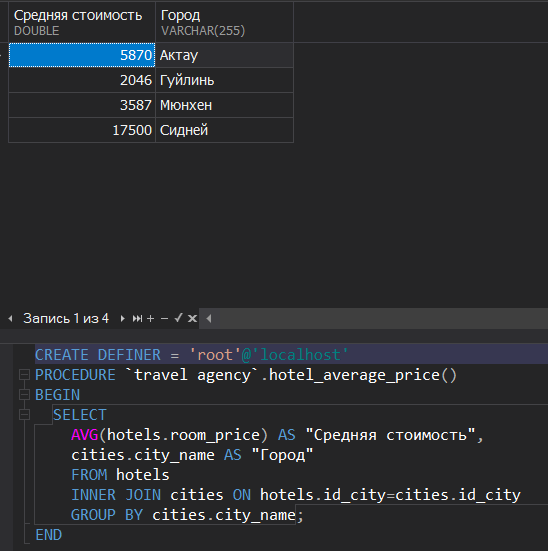
**Рисунок 5.8 – Создание и тестирование процедуры**

Процедура “client\_sales” выводит список клиентов которые купили тур, проходя по каждому туру и объединяет по его id (Рисунок 5.9).



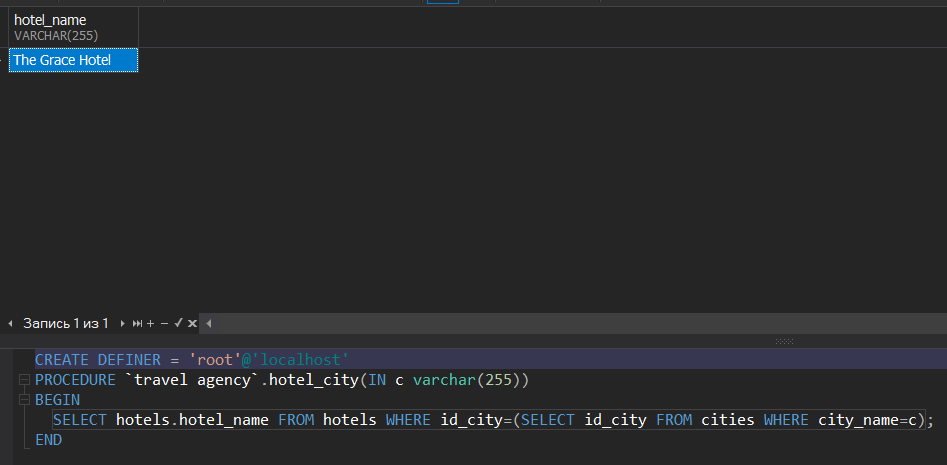
**Рисунок 5.9 – Создание и тестирование процедуры**

Процедура “hotel\_average\_price” выводит среднюю стоимость суток в отеле, разделяя по городам (Рисунок 5.10).



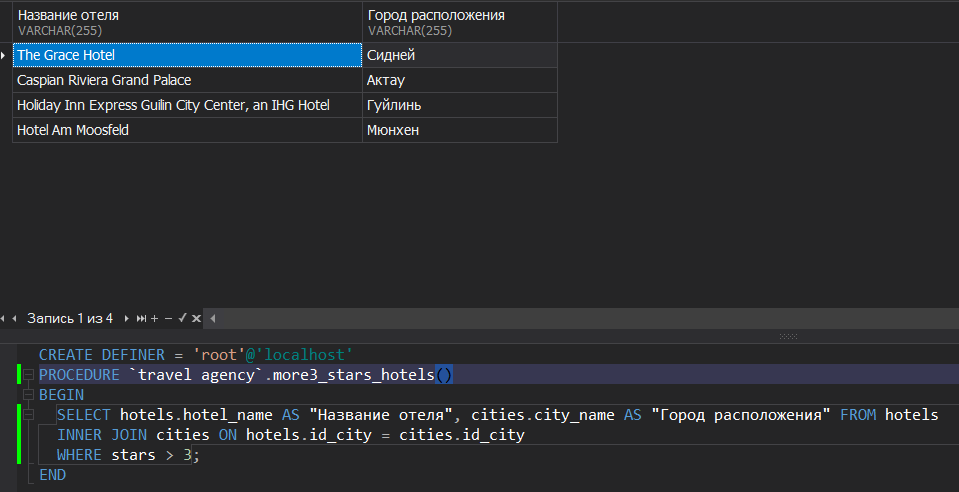
**Рисунок 5.10 – Создание и тестирование процедуры**

Следующая процедура “hotel city” выводит список отелей в городе, который мы передадим в неё (Рисунок 5.11).



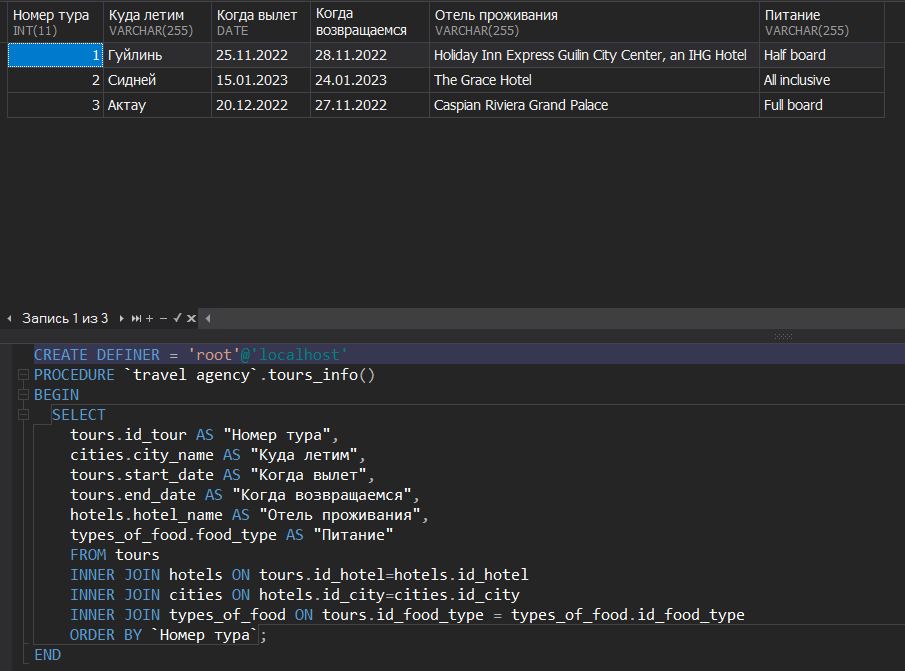
**Рисунок 5.11 – Создание и тестирование процедуры**

Процедура “more3\_stars\_hotel” выводит список отелей у которых рейтинг больше 3 звезд и город в котором они располагаются (Рисунок 5.12).



**Рисунок 5.12 – Создание и тестирование процедуры**

Последняя процедура выводит список всех туров с информацией о том какой у тура номер, в какой город полетят по этому туру, дату начала и конца тура, а также отель и тип питания в нем (Рисунок 5.13).



**Рисунок 5.13 – Создание и тестирование процедуры**

# Выводы

В данной практической работе были изучены методы и технологии разработки баз данных. Были изучены методы построения IDEF1X диаграмм, и создания таблиц базы данных на основе построенных представлений. Также были изучены основные запросы к таблицам баз данных и методы написания триггеров и функций.

# Список литературы

1. Лекционный материал по дисциплине «Разработка баз данных» Богомольная Г. В.
2. Методология IDEF1X // studme ScienceDirect [Электронный ресурс]. - https://studme.org/87187/ekonomika/metodologiya\_idef1x (дата обращения: 08.09.2022)
3. DataFlowDiagrams // ScienceDirect [Электронный ресурс]. - https://www.sciencedirect.com/science/article/pii/B0122272404000319 – (дата обращения: 08.09.2022)
4. Основы IDEF3 //CFIN [Электронный ресурс]. - https://www.cfin.ru/vernikov/idef/idef3.shtml – (дата обращения: 09.09.2022)
5. SQL запросы быстро. Часть 1// habr [Электронный ресурс]. - https://habr.com/ru/post/480838/ – (дата обращения: 23.09.2022)
6. Accessing data with MySQL //spring.io [Электронный ресурс]. - https://spring.io/guides/gs/accessing-data-mysql/ (дата обращения: 27.09.2022)