



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 (ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине «Базы Данных»

НА ТЕМУ:

База данных "Заказ банкета"

Студент

ИУ6-41Б

(Группа)

(Подпись, дата)

М. А. Тарасова

(И.О. Фамилия)

Руководитель

(Подпись, дата)

М. А. Скворцова

(И.О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка состоит из 36 страниц, включающих в себя 11 рисунков, 3 источника.

Объектом разработки является база данных агентства по организации банкетов и интерфейс для работы с имеющимися в ней данными.

Цель работы – создание базы данных с простым и понятным интерфейсом для работы с ней агентству.

Для реализации используется реляционная база данных PostgreSQL 15.2, графический клиент PgAdmin 6.20, язык разработки Ruby 3.2.0, интегрированная среда разработки Microsoft Visual Studio 2022.

База данных состоит из 10 связанных таблиц. Интерфейс позволяет посмотреть все записи в каждой таблице и получить данные по 7-ми select-запросам.

ОГЛАВЛЕНИЕ

Введение.....	5
1. Анализ предметной области.....	6
2. Модели базы данных.....	8
2.1. Инфологическая модель базы данных.....	8
2.2. Дatalogическая модель базы данных.....	9
3. Создание и заполнение базы данных.....	14
3.1. Скрипт для создания базы данных.....	14
3.2. Заполнение таблиц в VS Code на Ruby.....	17
4. Работа в Microsoft Visual Studio.....	23
4.1. Страница просмотра информации.....	23
4.2. Страница сложных запросов.....	26
Заключение.....	37
Список используемой литературы.....	38

ВВЕДЕНИЕ

Создаваемая система предназначена для обеспечения работы агентства по организации банкетов. Разрабатываемая база данных обеспечивает доступность информации о заказах, блюдах, клиентах, менеджерах и залах, а также о продуктах, из которых состоят блюда. Система дает возможность легко добавлять, обновлять и учитывать всю необходимую для работы агентства информацию.

Актуальность данной системы состоит в том, что она позволяет производить учёт заказов, учёт чаще заказываемых блюд, залов и каких продуктов нужно больше закупать в какой период.

Для работы базы данных была проведена её реализация в интегрированной среде разработки Microsoft Visual Studio 2022.

1. Анализ предметной области

Предметной областью разработанной базы данных является агентство по организации банкета. Для того, чтобы сервис функционировал, база данных должна обеспечивать контроль над заказами, хранить информацию о клиентах и менеджерах, с помощью неё аналитики агентства должны суметь понять, какие блюда и залы чаще всего заказывают люди и на какие праздники, что должно показать предпочтения клиентов и благодаря этому увеличить эффективность работы агентства.

В ходе анализа были выявлены следующие сущности:

1. Клиент

У каждого клиента есть свой персональный идентификатор, фамилия и инициалы, адрес проживания, телефонный номер и электронная почта.

2. Менеджер

У менеджера, как и у клиента, есть персональный идентификатор, фамилия с инициалами, адрес проживания, телефонный номер и почта. Но кроме этого у него также есть дата его принятия на работу и дата увольнения, которая может быть пустой, если менеджер ещё работает в агентстве.

3. Зал

У каждого зала есть свой персональный номер, название, количество мест, которое есть в этом зале, для понимания, какое количество гостей он может вместить, и стоимость самого зала.

4. Блюдо

Каждое блюдо имеет свой персональный идентификатор, название и стоимость.

5. Тип банкета

Данная сущность необходима для определения типа банкета и его стартовой цены. Каждый тип имеет свой персональный идентификатор, название, начальная стоимость и описание того, что входит в начальную стоимость.

6. Продукт

Эта сущность необходима для определения, какие продукты входят в блюда. Каждый продукт имеет свой идентификатор, название, дату поставки и количество его на складе.

7. Заказ

Центральной сущностью в нашей базе данных является заказ. В нём есть идентификатор клиента, дата, когда будет банкет, количество гостей банкета, идентификаторы менеджера и типа банкета, начальная цена, предоплата и конечная цена.

Главной сущностью в базе данных является заказ. Для обеспечения её полноценного существования были выделены второстепенные сущности, указанные выше.

2. Модели базы данных

Для создания полноценной базы данных нужно сначала спроектировать две её модели: инфологическую и даталогическую.

2.1. Инфологическая модель базы данных

Исходя из сущностей и их свойств, определенных в пункте 1, возможно построить инфологическую модель базы данных.

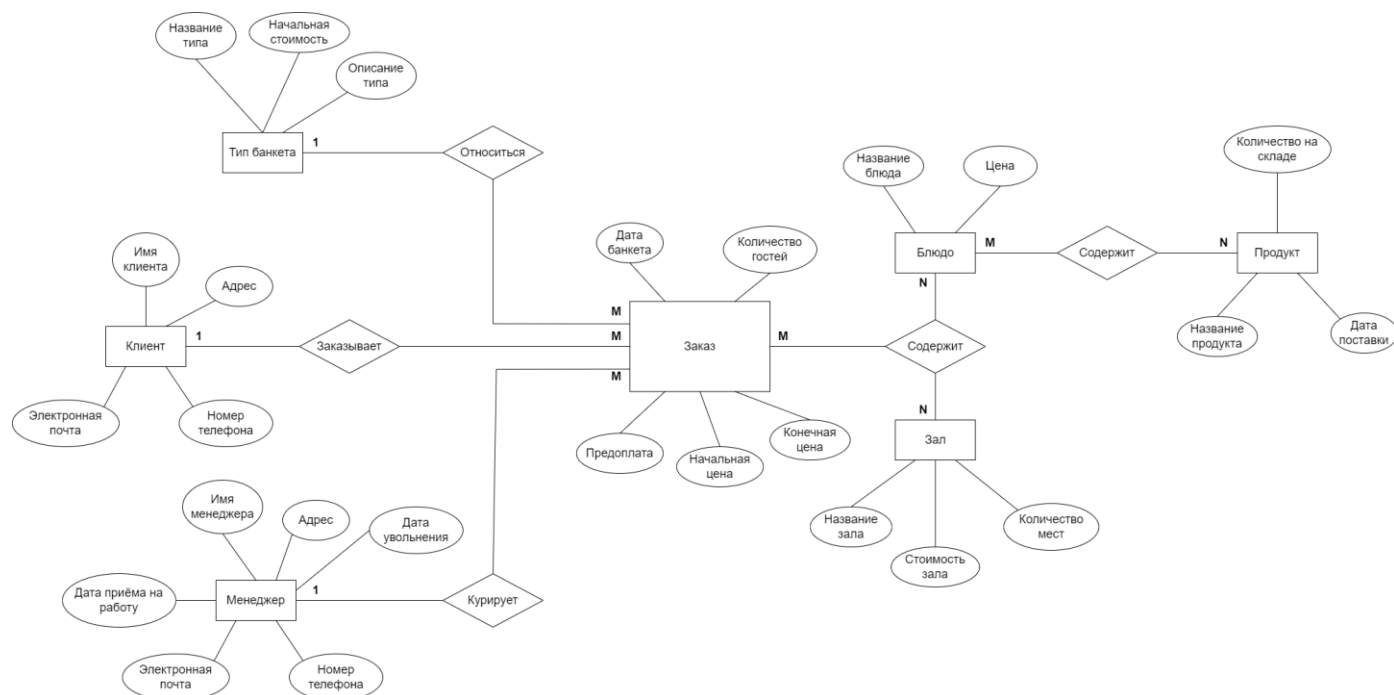


Рисунок 1 – инфологическая модель базы данных

Во время разработки инфологической модели были выделены главная и второстепенные сущности и взаимодействия между ними.

2.2. Даталогическая модель базы данных

Построив инфологическую модель базы данных, мы можем с её помощью построить даталогическую модель.

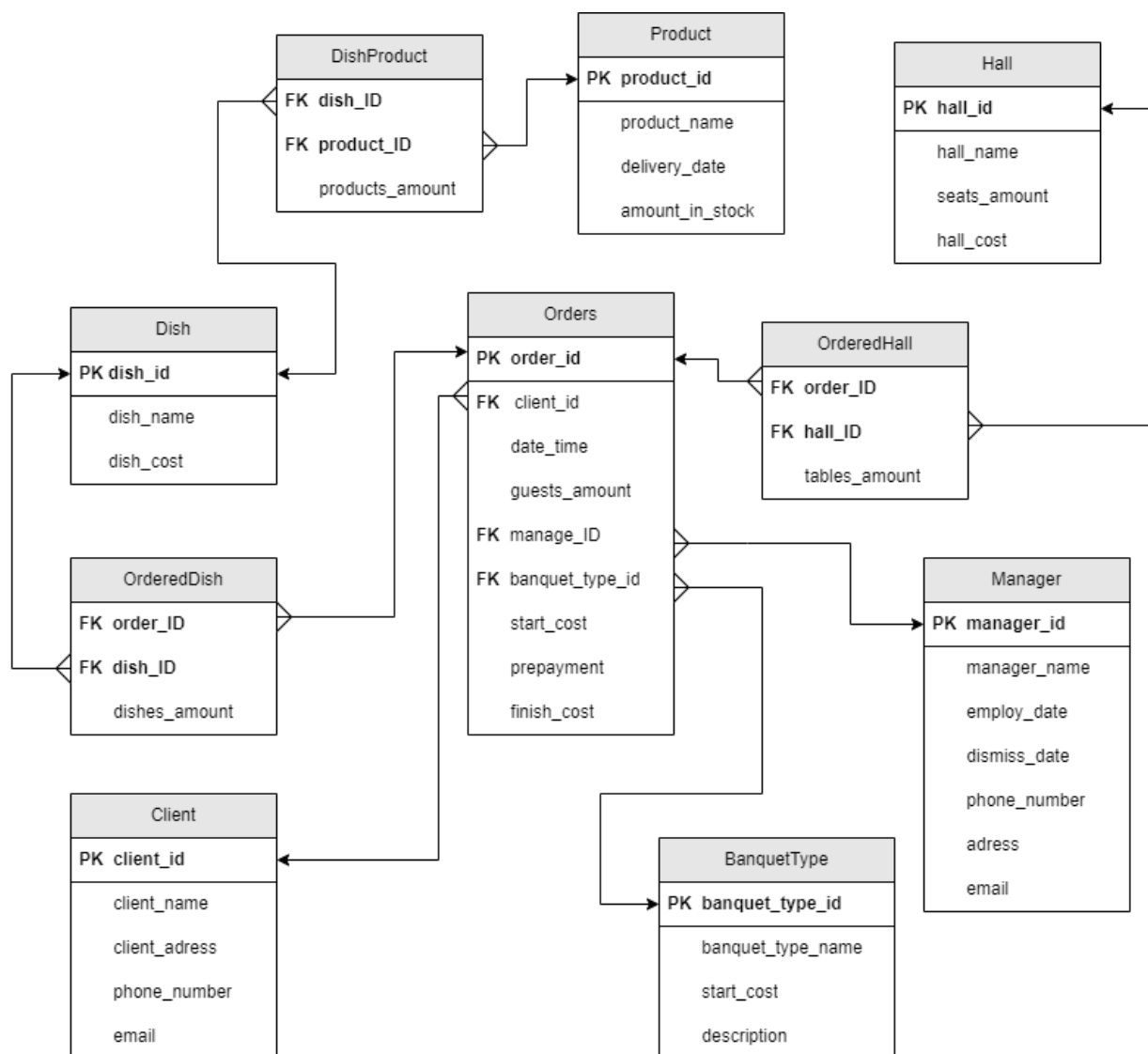


Рисунок 2 – Даталогическая модель базы данных

Ниже приведено описание каждой таблицы для лучшего понимания всей концепции базы данных:

Таблица 2.2.1

Описание таблицы «Client»

Название поля	Тип поля	Назначение поля
client_id	serial	Уникальный идентификатор

client_name	text	Имя клиента (Фамилия и инициалы)
client_adress	text	Адрес проживания клиента
phone_number	varchar(14)	Телефонный номер клиента
email	text	Электронная почта клиента

Таблица 2.2.2

Описание таблицы «Manager»

Название поля	Тип поля	Назначение поля
manager_id	serial	Уникальный идентификатор
manager_name	text	Имя менеджера (Фамилия и инициалы)
employ_date	date	Дата принятия на работу
dismiss_date	date	Дата увольнения
manager_adress	text	Адрес проживания менеджера
phone_number	varchar(14)	Телефонный номер менеджера
email	text	Электронная почта менеджера

Таблица 2.2.3

Описание таблицы «BanquetType»

Название поля	Тип поля	Назначение поля
banquet_type_id	serial	Уникальный идентификатор
banquet_type_name	text	Название типа банкета
start_cost	decimal(10,2)	Начальная стоимость банкета

description	text	Описание того, что входит в начальную стоимость
-------------	------	---

Таблица 2.2.4

Описание таблицы «Hall»

Название поля	Тип поля	Назначение поля
hall_id	serial	Уникальный идентификатор
hall_name	text	Название зала
seats_amount	int	Количество мест в зале
hall_cost	decimal(10,2)	Стоимость аренды зала на день

Таблица 2.2.5

Описание таблицы «Dish»

Название поля	Тип поля	Назначение поля
dish_id	serial	Уникальный идентификатор
dish_name	text	Название блюда
dish_cost	decimal(4,2)	Цена блюда

Таблица 2.2.6

Описание таблицы «Product»

Название поля	Тип поля	Назначение поля
product_id	serial	Уникальный идентификатор
product_name	text	Название продукта
delivery_date	date	Дата поставки
amount_in_stock	int	Количество продукта на складе

Таблица 2.2.7

Описание таблицы «OrderedHall»

Название поля	Тип поля	Назначение поля
order_id	int	Указатель на заказ (его номер в системе)
hall_id	int	Указатель на зал (его номер в системе)
tables_amount	int	Количество столов в зале

Таблица 2.2.8

Описание таблицы «OrderedDish»

Название поля	Тип поля	Назначение поля
order_id	int	Указатель на заказ (его номер в системе)
dish_id	int	Указатель на блюдо (его номер в системе)
dishes_amount	int	Количество блюд данного вида в заказе

Таблица 2.2.9

Описание таблицы «DishProduct»

Название поля	Тип поля	Назначение поля
dish_id	int	Указатель на блюдо (его номер в системе)
product_id	int	Указатель на продукт (его номер в системе)
products_amount	int	Количество продуктов в блюде

Описание таблицы «Orders»

Название поля	Тип поля	Назначение поля
order_id	serial	Уникальный идентификатор
client_id	int	Указатель на клиента (его номер в системе)
date_time	date	Дата банкета
guests_amount	int	Количество гостей банкета
manage_id	int	Указатель на менеджера (его номер в системе)
banquet_type_id	int	Указатель на тип банкета (его номер в системе)
start_cost	decimal(10,2)	Начальная стоимость банкета
prepayment	decimal(10,2)	Предоплата
finish_cost	decimal(10,2)	Конечная стоимость банкета

3. Создание и заполнение базы данных

Теперь, когда мы сделали обе модели базы данных, мы можем написать скрипт для создания базы в PostgreSQL 15.2

3.1. Скрипт для создания базы данных

Ниже в листинге 1 приведён скрипт для создания БД.

Листинг 1:

```
create table if not exists Hall (  
    hall_id serial primary key,  
    hall_name text NOT NULL,  
    seats_amount int NOT NULL CONSTRAINT positive_seats_amount CHECK  
(seats_amount > 0)  
);
```

```
create table if not exists Manager (  
    manager_id serial primary key,  
    manage_name text NOT NULL,  
    employ_date date NOT NULL,  
    dismiss_date date  
    phone_number varchar(13) NOT NULL,  
    address text NOT NULL,  
    email text NOT NULL  
);
```

```
create table if not exists Dish (  
    dish_id serial primary key,  
    dish_name text NOT NULL,  
    dish_cost decimal(10,2) NOT NULL CONSTRAINT positive_dish_cost  
CHECK(dish_cost > 0)
```

);

```
create table if not exists Orders (  
    order_id serial primary key,  
    client_id int NOT NULL,  
    date_time date NOT NULL,  
    guests_amount int NOT NULL CONSTRAINT positive_guests_amount CHECK  
(guests_amount > 0),  
    manager_ID int,  
    banquet_type_id int,  
    start_cost decimal(10,2) NOT NULL CONSTRAINT positive_start_cost CHECK  
(start_cost > 0),  
    prepayment decimal(10,2) NOT NULL CONSTRAINT positive_prepayment CHECK  
(prepayment > 0),  
    finish_cost decimal(10,2) NOT NULL CONSTRAINT positive_finish_cost CHECK  
(finish_cost > 0),  
    CHECK (finish_cost >= start_cost),  
    FOREIGN KEY (manager_ID) references Manager (manager_id) on delete SET  
NULL,  
    FOREIGN KEY (client_ID) references Client (client_id) on delete cascade,  
    FOREIGN KEY (banquet_type_id) references BanquetType (banquet_type_id) on  
delete SET NULL  
);
```

```
create table if not exists OrderedDish (  
    order_ID int NOT NULL,  
    dish_ID int NOT NULL,
```

```

        dishes_amount int NOT NULL CONSTRAINT positive_dishes_amount CHECK
(dishes_amount > 0),
        foreign key (dish_ID) references Dish (dish_id) on delete cascade,
        foreign key (order_ID) references Orders (order_id) on delete cascade
);

```

```

CREATE TABLE IF NOT EXISTS Product (
    product_id serial primary key,
    product_name text NOT NULL,
    delivery_date date NOT NULL,
    amount_in_stock int NOT NULL CONSTRAINT positive_amount_in_stock CHECK
(amount_in_stock > 0)
);

```

```

CREATE TABLE IF NOT EXISTS DishProduct (
    dish_ID int NOT NULL,
    product_ID int NOT NULL,
    products_amount int NOT NULL CONSTRAINT positive_products_amount CHECK
(products_amount > 0),
    FOREIGN KEY (dish_ID) references Dish (dish_id) on delete cascade,
    Foreign key (product_ID) references Product (product_id) on delete cascade
);

```

```

create table if not exists OrderedHall (
    order_id int not NULL,
    hall_id int not null,
    tables_amount int not null constraint positive_tables_amount
check(tables_amount > 0),

```



```
foreign key (order_id) references Orders(order_id) on delete cascade,  
foreign key (hall_id) references Hall(hall_id) on delete cascade  
);
```

```
create table if not exists Client (  
    client_id serial primary key,  
    client_name text NOT NULL,  
    client_adress text NOT NULL,  
    phone_number varchar(14) NOT NULL,  
    email text NOT NULL  
);
```

```
create table if not exists BanquetType (  
    banquet_type_id serial primary key,  
    banquet_type_name text NOT NULL,  
    start_cost decimal(10,2) NOT null,  
    description text  
);
```

3.2. Заполнение таблиц в VS Code на Ruby

После того, как мы написали скрипт и создали базу данных, мы должны её заполнить. Заполнение производилось в редакторе кода VS Code на языке Ruby 3.2.0. Ниже представлен листинг 2 с кодом заполнения.

Листинг 2:

```
# frozen_string_literal: true  
  
require 'pg'  
require 'faker'  
Faker::Config.locale = :ru  
conn = PG.connect(dbname: 'Ordering a banquet', user: 'postgres', password:  
'CrazybirD')
```

```

# Заполнение таблицы Manager

def manager_name()
  return Faker::Name.last_name + ' ' + Faker::Name.initials(number: 1) + '.' +
  Faker::Name.initials(number: 1) + '.'
end

def employ_date()
  return Faker::Date.between(from: '2017-09-23', to: '2020-05-06')
end

def dismiss_date()
  return Faker::Date.between(from: '2020-09-23', to: '2023-05-06')
end

def phone_number()
  return Faker::PhoneNumber.cell_phone_in_e164
end

def manager_values()
  return %("#{manager_name}", "#{employ_date}", "#{phone_number}")
end

conn.exec%(Insert into Manager (manager_name, employ_date, phone_number)
  Values #{100.times.collect{manager_values}.to_a.join(',')})

# Заполнение табицы Dish

def dish_name()
  return Faker::Food.dish
end

def dish_cost()
  return Faker::Number.decimal(l_digits: 3, r_digits: 2)
end

def dish_values()
  return %("#{dish_name}", "#{dish_cost}")
end

conn.exec%(Insert into Dish (dish_name, dish_cost)
  Values #{100.times.collect{dish_values}.to_a.join(',')})

# Заполнение таблицы Product

def product_name()
  return Faker::Food.ingredient
end

```

```

def delivery_date()
  return Faker::Date.between(from: '2023-05-01', to: '2023-05-06')
end

def number_in_stock()
  return Faker::Number.between(from: 50, to: 200)
end

def product_values()
  return %("#{product_name}", "#{delivery_date}", "#{number_in_stock}")
end

conn.exec(%(Insert into Product (product_name, delivery_date, amount_in_stock)
  Values #{100.times.collect{product_values}.to_a.join(',')}))

# Заполнение таблицы DishProduct

def dish_id()
  return Faker::Number.between(from: 1, to: 101)
end

def product_id()
  return Faker::Number.between(from: 1, to: 100)
end

def products_amount()
  return Faker::Number.between(from: 1, to: 5)
end

def dish_product_values()
  return %("#{dish_id}", "#{product_id}", "#{products_amount}")
end

conn.exec(%(Insert into DishProduct (dish_id, product_id, products_amount)
  Values #{100.times.collect{dish_product_values}.to_a.join(',')}))

# Заполняем таблицу Client

def client_email()
  return Faker::Internet.email(domain: 'yandex.ru')
end

def client_adress()
  return Faker::Address.street_address + ', Москва, Россия'
end

def client_values()
  return %("#{manager_name}", "#{client_adress}", "#{phone_number}",
  "#{client_email}")
end

```

```

conn.exec(%(Insert into Client (client_name, client_adress, phone_number, email)
  Values #{100.times.collect{client_values}.to_a.join(',')}))

# заполняем таблицу Hall

def hall_name()
  return Faker::Restaurant.name
end

def seats_amount()
  return Faker::Number.between(from: 30, to: 250)
end

def hall_cost()
  return Faker::Number.between(from: 60000, to: 200000)
end

def hall_values()
  return %("#{hall_name}", #{seats_amount}, #{hall_cost})
end

conn.exec(%(Insert into Hall (hall_name, seats_amount, hall_cost)
  Values #{100.times.collect{hall_values}.to_a.join(',')}))

# Заполнение таблицы Orders

def client_id()
  return Faker::Number.between(from: 101, to: 200)
end

def date_time()
  return Faker::Date.between(from: '2016-02-01', to: '2023-05-06')
end

def guests_amount()
  return Faker::Number.between(from: 20, to: 200)
end

def manager_id()
  return Faker::Number.between(from: 1, to: 101)
end

def start_cost()
  return Faker::Number.between(from: 80000, to: 300000)
end

def order_values()
  return %("#{client_id}", "#{date_time}", #{guests_amount}, #{manager_id},
  #{start_cost})

```

```

end

conn.exec(%(Insert into orders (client_id, date_time, guests_amount, manager_id,
start_cost)
  Values #{1000000.times.collect{order_values}.to_a.join(',')}))

# Заполняем таблицу OrderedDish

def order_id()
  return Faker::Number.between(from: 1, to: 10000100)
end

def ordered_dish_values()
  return %((#{order_id}, #{dish_id}))
end

conn.exec(%(Insert into orderreddish (order_id, dish_id)
  Values #{100000.times.collect{ordered_dish_values}.to_a.join(',')}))

# Заполняем таблицу OrderedHall

def hall_id()
  return Faker::Number.between(from: 1, to: 100)
end

def ordered_hall_values()
  return %((#{order_id}, #{hall_id}))
end

conn.exec(%(Insert into orderedhall (order_id, hall_id)
  Values #{100000.times.collect{ordered_hall_values}.to_a.join(',')}))

for i in 1..101 do
conn.exec(%(Update Manager set email = '#{client_email}' where manager_id = #{i}))
end

def banquet_type()
  return %((#{Faker::Number.between(from: 1, to: 6)}))
end

conn.exec(%(Insert into orders (banquet_type_id)
  Values #{10000000.times.collect{banquet_type}.to_a.join(',')}))

for i in 1..10000000 do
conn.exec(%(Update orders set banquet_type_id = '#{Faker::Number.between(from: 1,
to: 6)}' where order_id = #{i}))
end

```

Триггеры и функции, выполняющиеся при заполнении БД:

1. Триггер для заполнения начальной стоимости и предоплаты из BanquetType в Orders (представлен в листинге 4):

Листинг 4:

```
CREATE OR REPLACE FUNCTION set_start_cost() returns
trigger as
$$
begin
UPDATE orders
    set start_cost = banquettype.start_cost
    from banquettype
    where order_id = new.order_id and orders.banquet_type_id =
banquettype.banquet_type_id;
Update orders
    set prepayment = round(start_cost/2,2)
    where order_id = new.order_id;
update orders
    set finish_cost = start_cost
    where order_id = new.order_id;
return new;
end;
$$
language 'plpgsql';

create or replace trigger add_start_cost
after insert on orders for each row
EXECUTE PROCEDURE set_start_cost();
```

2. Функция заполнения конечной цены с учётом стоимости залов (представлена в листинге 5):

Листинг 5:

```
update orders
set finish_cost = finish_cost + query_in.hall_cost
from (select order_id, hall_cost
      from hall inner join orderedhall using(hall_id)) query_in
where orders.order_id = query_in.order_id;
```

3. Функция заполнения конечной цены с учётом стоимости блюд (представлена в листинге 6):

Листинг 6:

```
update orders
set finish_cost = finish_cost + query_in.dishes_cost
from (select order_id, dish_cost*dishes_amount as dishes_cost
      from dish inner join orderreddish using(dish_id)) query_in
where orders.order_id = query_in.order_id;
```

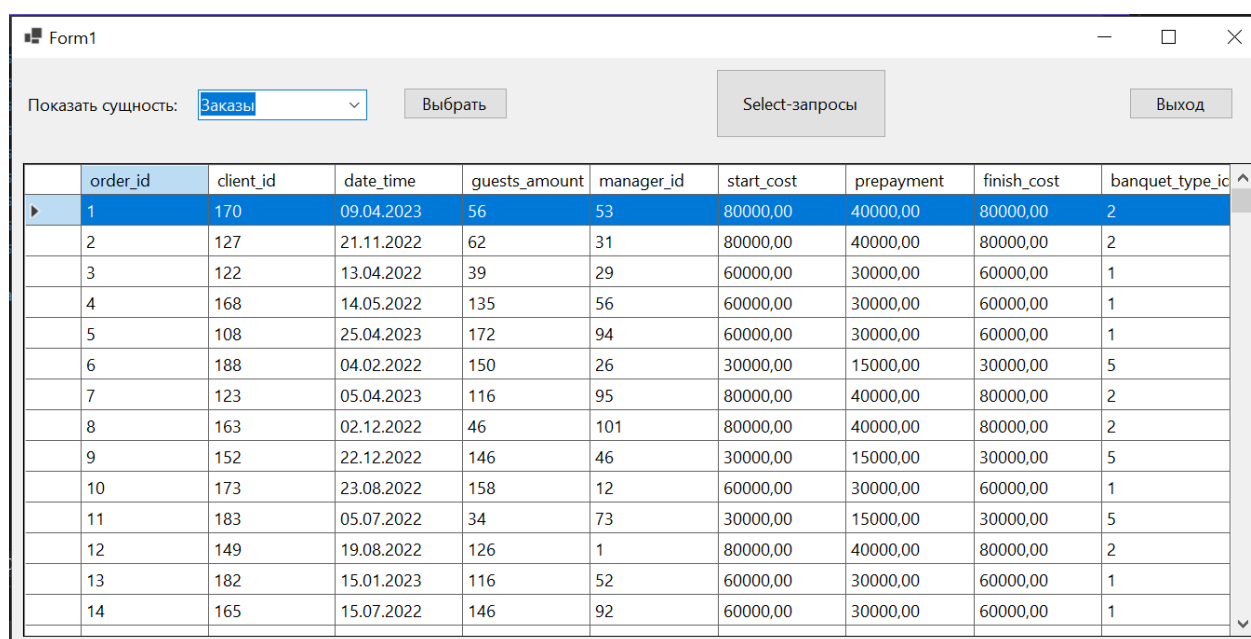
4. Работа в Microsoft Visual Studio

После заполнения БД можно переходить к визуализации интерфейса.

В данной курсовой работе визуализация производилась с помощью форм Microsoft Visual Studio 2022 и языка C#.

4.1. Страница просмотра информации

Чтобы можно было просмотреть все записи во всех таблицах, была создана страница просмотра таблиц, представленная на рисунке 3.



	order_id	client_id	date_time	guests_amount	manager_id	start_cost	prepayment	finish_cost	banquet_type_ic
▶	1	170	09.04.2023	56	53	80000,00	40000,00	80000,00	2
	2	127	21.11.2022	62	31	80000,00	40000,00	80000,00	2
	3	122	13.04.2022	39	29	60000,00	30000,00	60000,00	1
	4	168	14.05.2022	135	56	60000,00	30000,00	60000,00	1
	5	108	25.04.2023	172	94	60000,00	30000,00	60000,00	1
	6	188	04.02.2022	150	26	30000,00	15000,00	30000,00	5
	7	123	05.04.2023	116	95	80000,00	40000,00	80000,00	2
	8	163	02.12.2022	46	101	80000,00	40000,00	80000,00	2
	9	152	22.12.2022	146	46	30000,00	15000,00	30000,00	5
	10	173	23.08.2022	158	12	60000,00	30000,00	60000,00	1
	11	183	05.07.2022	34	73	30000,00	15000,00	30000,00	5
	12	149	19.08.2022	126	1	80000,00	40000,00	80000,00	2
	13	182	15.01.2023	116	52	60000,00	30000,00	60000,00	1
	14	165	15.07.2022	146	92	60000,00	30000,00	60000,00	1

Рисунок 3 – Страница просмотра таблиц

Код создания данной страницы представлен в листинге 7.

Листинг 7:

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TermWork_2._0
{
    public partial class Form1 : Form
    {
        public string connString = String.Format("Server={0}; Port = {1};" +
            "User ID = {2}; Password = {3}; Database = {4}", "localhost", 5432, "postgres",
            "CrazybirD", "Ordering a banquet");

        private NpgsqlConnection conn;
        private string sql;
        private NpgsqlCommand cmd;
        private DataTable dt;
        private string tableChoice;

        public Form1()
        {
            InitializeComponent();

            private void Form1_Load(object sender, EventArgs e)
            {
                conn = new NpgsqlConnection(connString);
                sql = @"select * from orders order by 1 limit 500";
                Choose();
                comboBox1.Items.Insert(0, "Заказы");
                comboBox1.Items.Insert(1, "Менеджеры");
                comboBox1.Items.Insert(2, "Клиенты");
                comboBox1.Items.Insert(3, "Типы банкетов");
                comboBox1.Items.Insert(4, "Залы");
                comboBox1.Items.Insert(5, "Блюда");
                comboBox1.Items.Insert(6, "Продукты");
                comboBox1.Items.Insert(7, "Арендованные залы");
                comboBox1.Items.Insert(8, "Заказанные блюда");
                comboBox1.Items.Insert(9, "Продукты в блюдах");
            }

            private void Choose()
            {
                try
                {
                    conn.Open();
                    cmd = new NpgsqlCommand(sql, conn);
                    dt = new DataTable();
                    dt.Load(cmd.ExecuteReader());
                    conn.Close();
                    dataGridView1.DataSource = null;
                    dataGridView1.DataSource = dt;
                }
                catch (Exception ex)
                {
                    conn.Close();
                    MessageBox.Show("Error: " + ex.Message);
                }
            }

            private void button1_Click(object sender, EventArgs e)
            {
                Close();
            }

            private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)

```



```

{
}

private void button2_Click(object sender, EventArgs e)
{
    tableChoice = comboBox1.SelectedIndex.ToString();
    switch (tableChoice)
    {
        case "0":
        {
            sql = @"select * from orders where finish_cost is not null
order by 1 limit 500";
            break;
        }
        case "1":
        {
            sql = @"select * from Manager";
            break;
        }
        case "2":
        {
            sql = @"select * from Client";
            break;
        }
        case "3":
        {
            sql = @"select * from BanquetType";
            break;
        }
        case "4":
        {
            sql = @"select * from Hall";
            break;
        }
        case "5":
        {
            sql = @"select * from Dish";
            break;
        }
        case "6":
        {
            sql = @"select * from Product";
            break;
        }
        case "7":
        {
            sql = @"select * from OrderedHall";
            break;
        }
        case "8":
        {
            sql = @"select * from OrderedDish";
            break;
        }
        case "9":
        {
            sql = @"select * from DishProduct";
            break;
        }
        default:
            sql = @"select * from Orders";
            break;
    }
    Choose();
}

```

```

    }

    private void button3_Click(object sender, EventArgs e)
    {
        Form2 f2 = new Form2();
        f2.ShowDialog();
    }
}

```

4.2. Страница сложных запросов

Отдельно сделана страница сложных SELECT-запросов, чтоб не загромождать интерфейс первой страницы. Её дизайн представлен на рисунке 4.

The screenshot shows a Windows application window titled 'Form2'. At the top, there is a horizontal bar with eight buttons: 'Select 1', 'Select 2', 'Select 3' (which is highlighted with a blue border), 'Select 4', 'Select 5', 'Select 6', 'Select 7', and 'Выход в Меню'. Below the buttons is a text area containing the text: 'Какие заказы включают блюда, использующие самые редкие продукты? Вывести номер заказа, название блюда, название продукта'. Below the text area is a table with three columns: 'order_id', 'dish_name', and 'product_name'. The table contains 13 rows of data. The first row is highlighted in blue.

order_id	dish_name	product_name
6406938	Pasta Carbonara	Potato Flour
5176700	French Fries with Sausages	Coriander Seed
6387094	Chilli con Carne	Agar
4959107	Chilli con Carne	Agar
3831708	Pork Sausage Roll	Oyster Sauce
2954664	Cheeseburger	Quinoa
3058248	Caprese Salad	Radish
5561845	Mushroom Risotto	Pear Juice
4346136	Caprese Salad	Radish
2629167	French Fries with Sausages	Coriander Seed

Рисунок 4 – Страница сложных SELECT-запросов

Код для реализации страницы SELECT-запросов представлен в листинге 8.

Листинг 8:

```

using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TermWork_2._0
{
    public partial class Form2 : Form
    {
        private NpgsqlConnection conn;
        private string sql;
    }
}

```

```

private NpgsqlCommand cmd;
private DataTable dt;

public Form2()
{
    InitializeComponent();
}

private void Choose()
{
    try
    {
        conn.Open();
        cmd = new NpgsqlCommand(sql, conn);
        dt = new DataTable();
        dt.Load(cmd.ExecuteReader());
        conn.Close();
        dataGridView1.DataSource = null;
        dataGridView1.DataSource = dt;
    }
    catch (Exception ex)
    {
        conn.Close();
        MessageBox.Show("Error: " + ex.Message);
    }
}

private void Form2_Load(object sender, EventArgs e)
{
    Form1 form1 = new Form1();
    conn = new NpgsqlConnection(form1.connString);
}

private void button1_Click(object sender, EventArgs e)
{
    Close();
}

private void button2_Click(object sender, EventArgs e)
{
    richTextBox1.Text = "Вывести номер и конечную стоимость заказов, включающих  
самое популярное блюдо. Название блюда и его цену указать в двух последних колонках.";
    sql = @"select order_id, finish_cost, dish_name, dish_cost
            from orders
                inner join orderreddish using (order_id)
                inner join Dish on dish.dish_id = orderreddish.dish_id
            where dish.dish_id in (select dish_id
                                   from orderreddish
                                   group by dish_id
                                   having sum(dishes_amount) =
(select max(sum_dishes_amount)
 from (select dish_id, sum(dishes_amount) as sum_dishes_amount
        from orderreddish
        group by dish_id
        order by sum_dishes_amount desc) query_1))
            order by order_id";
    Choose();
}

private void button3_Click(object sender, EventArgs e)
{

```

```

        richTextBox1.Text = "В каких залах чаще всего празднуют Дни рождения?
Вывести название зала и количество проводимых Дней рождения для 10 верхних записей";
        sql = @"select hall_name, count(order_id) as Количество_праздников
        from orderedhall
            inner join hall using(hall_id)
        where order_id in (select order_id
            from orders
            where banquet_type_id = (select
banquet_type_id
from banquettype
where banquet_type_name like 'Birthday'))
        group by hall_name
        order by Количество_праздников desc
        limit 10";
        Choose();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        richTextBox1.Text = "Какие заказы включают блюда, использующие самые редкие
продукты? Вывести номер заказа, название блюда, название продукта";
        sql = @"select order_id, dish_name, product_name
        from orderreddish
            inner join dish using(dish_id)
            inner join dishproduct using(dish_id)
            inner join product using(product_id)
        group by order_id, dish_name, product_name
        having sum(products_amount) = (select min(Количество_штук)
            from (select
product_id, sum(products_amount) as Количество_штук
from dishproduct
group by product_id
order by Количество_штук) query_1)";
        Choose();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        richTextBox1.Text = "Какие клиенты какие залы чаще всего заказывают?";
        sql = @"select client_name as Имя_клиента, (array_agg(hall_name))[1] as
Чаще_заказываемый_зал, (array_agg(replicas_amount))[1] as Количество_праздников
        from (select client_name, hall_name, count(hall_name)
            from orders
            inner join orderedhall
            inner join client using(client_id)
            inner join hall using(hall_id)
            group by client_name, hall_name
            order by 1, 3 desc) query_1
        group by client_name";
        Choose();
    }

    private void button6_Click(object sender, EventArgs e)
    {

```

```

        richTextBox1.Text = "Для клиентов у которых средняя конечная стоимость
заказов выше средней конечной стоимости по суммам заказов клиентов (общей стоимости
всех заказов клиентов), вывести имя, общую конечную стоимость всех заказов, количество
заказов.";
        sql = @"select client_name, count(order_id) as orders_amount,
round(sum(finish_cost)/count(order_id),2) as middle_finish_cost
        from orders
            inner join client using(client_id)
        group by client_name
        having round(sum(finish_cost)/count(order_id),2) > (select
round(avg(finish_cost),2) as middle_cost
from orders)";
        Choose();
    }

    private void button7_Click(object sender, EventArgs e)
    {
        richTextBox1.Text = "Какие клиенты, какие блюда чаще всего заказывают на
Новый год?";
        sql = @"select client_name as Имя_клиента, (array_agg(dish_name))[1] as
Чаще_заказываемое_блюдо, (array_agg(replicas_amount))[1] as Количество_заказов
        from (select client_name, dish_name, count(dish_name)
        as replicas_amount
        from orders
            inner join orderreddish
        using(order_id)
            inner join client using(client_id)
            inner join dish using(dish_id)
        where banquet_type_id = (select
banquet_type_id
        from banquettype
        where banquet_type_name like 'New Year%')
        group by client_name, dish_name
        order by 1, 3 desc) query_1
        group by client_name";
        Choose();
    }

    private void button8_Click(object sender, EventArgs e)
    {
        richTextBox1.Text = "--Вывести статистику по клиенту за определённый период
(какие блюда для каких праздников чаще всего заказывались)\r\n";
    }
}
}

```

Разберём каждый запрос по отдельности:

- 1) Вывести номер и конечную стоимость заказов, включающих самое популярное блюдо. Название блюда и его цену указать в двух последних колонках. (Представлен в листинге 9)

Листинг 9:

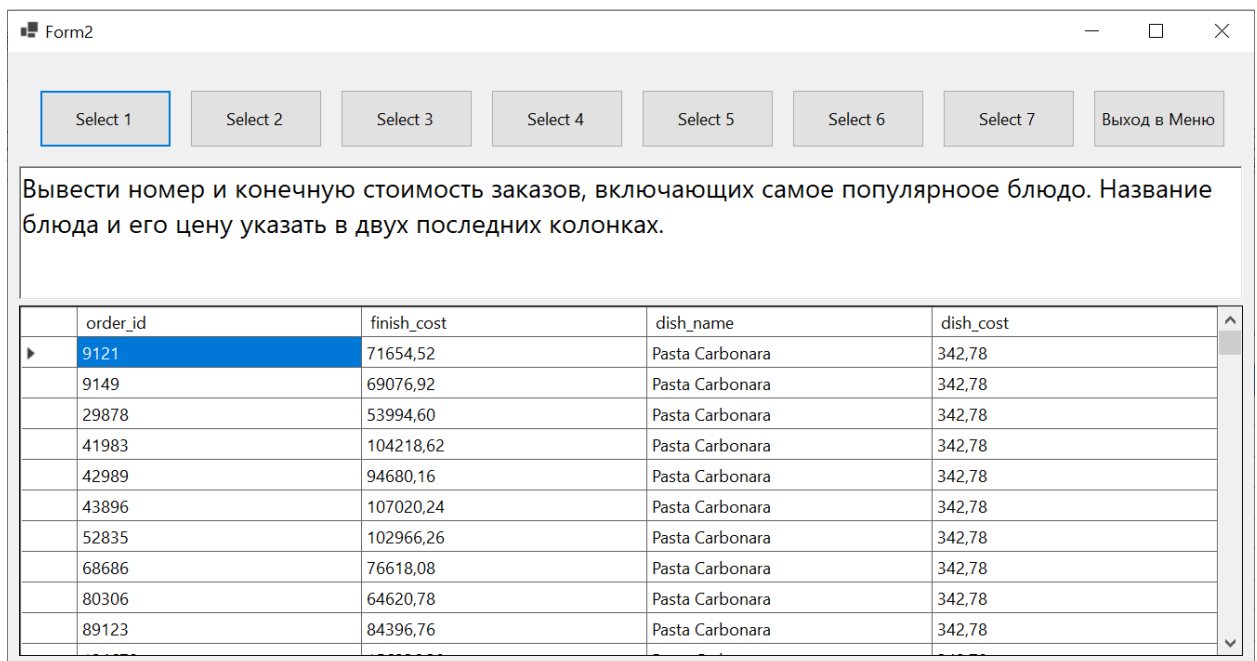
```

select order_id, finish_cost, dish_name, dish_cost
from orders
    inner join ordereddish using (order_id)
    inner join Dish on dish.dish_id = ordereddish.dish_id
where dish.dish_id in (select dish_id
                        from ordereddish
                        group by dish_id
                        having sum(dishes_amount) = (select
max(sum_dishes_amount)
from (select dish_id, sum(dishes_amount) as sum_dishes_amount

        from ordereddish
        group by dish_id
        order by sum_dishes_amount desc) query_1))
order by order_id

```

На рисунке 5 представлен результат данного запроса:



order_id	finish_cost	dish_name	dish_cost
9121	71654,52	Pasta Carbonara	342,78
9149	69076,92	Pasta Carbonara	342,78
29878	53994,60	Pasta Carbonara	342,78
41983	104218,62	Pasta Carbonara	342,78
42989	94680,16	Pasta Carbonara	342,78
43896	107020,24	Pasta Carbonara	342,78
52835	102966,26	Pasta Carbonara	342,78
68686	76618,08	Pasta Carbonara	342,78
80306	64620,78	Pasta Carbonara	342,78
89123	84396,76	Pasta Carbonara	342,78

Рисунок 5 – Результат первого SELECT-запроса

- 2) В каких залах чаще всего празднуют Дни рождения? Вывести название зала и количество проводимых Дней рождения для 10 верхних записей. (Представлен в листинге 10)

Листинг 10:

```

select hall_name, count(order_id) as Количество_праздников

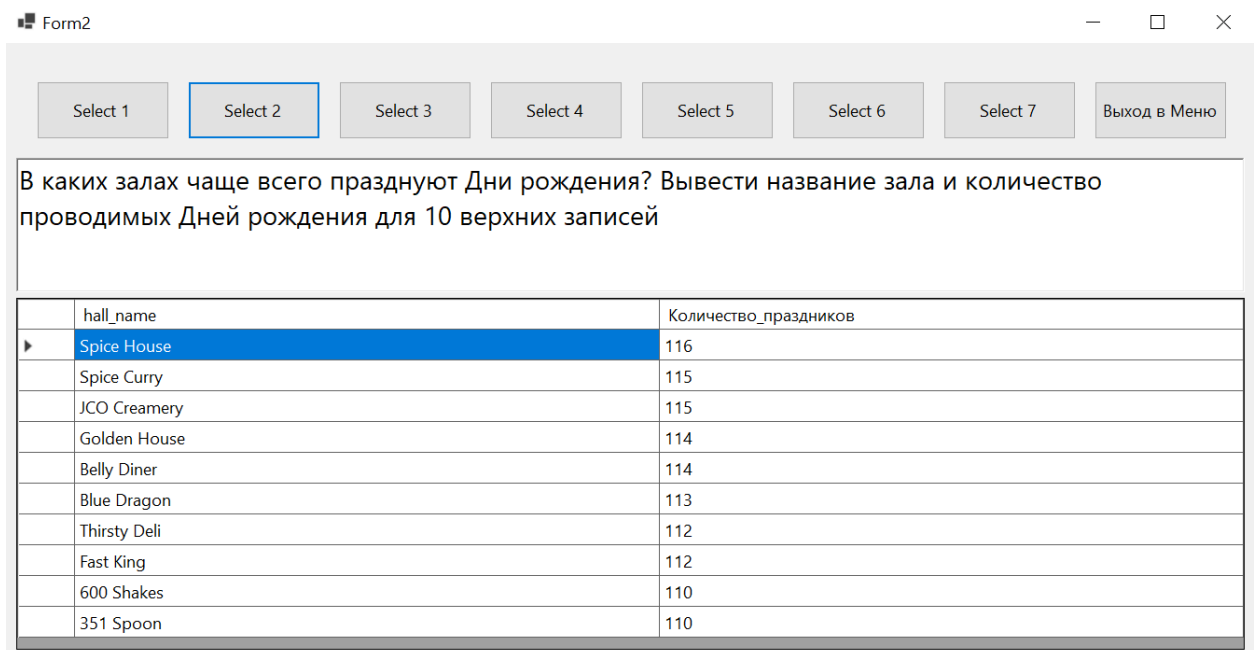
```

```

from orderedhall
inner join hall using(hall_id)
where order_id in (select order_id
                   from orders
                   where banquet_type_id = (select banquet_type_id
                                           from banquettype
                                           where banquet_type_name like 'Birthday'))
group by hall_name
order by Количество_праздников desc
limit 10

```

Результат данного запроса представлен на рисунке 6:



hall_name	Количество_праздников
Spice House	116
Spice Curry	115
JCO Creamery	115
Golden House	114
Belly Diner	114
Blue Dragon	113
Thirsty Deli	112
Fast King	112
600 Shakes	110
351 Spoon	110

Рисунок 6 – Результат второго SELECT-запроса

3) Какие заказы включают блюда, использующие самые редкие продукты?

Вывести номер заказа, название блюда, название продукта. (Представлен в листинге 11)

Листинг 11:

```

select order_id, dish_name, product_name
from orderreddish
    inner join dish using(dish_id)
    inner join dishproduct using(dish_id)
    inner join product using(product_id)
group by order_id, dish_name, product_name

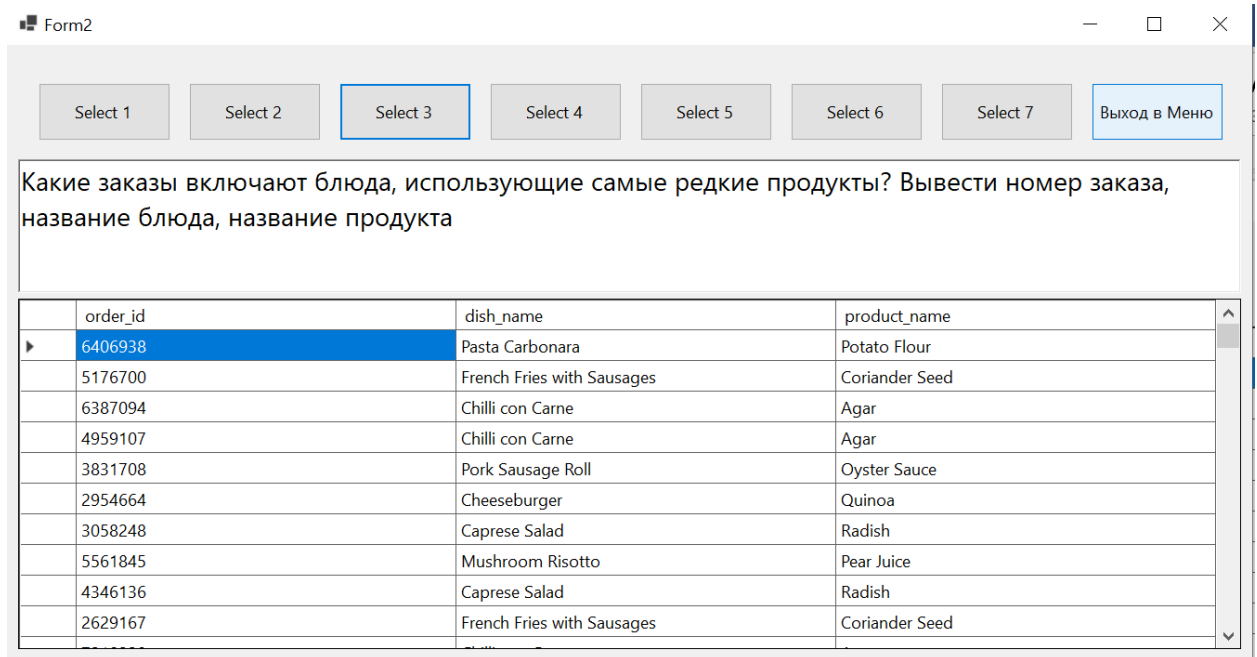
```

```

having sum(products_amount) = (select min(Количество_штук)
                                from (select product_id, sum(products_amount) as
Количество_штук
                                from dishproduct
                                group by product_id
                                order by Количество_штук) query_1)

```

Результат данного запроса представлен на рисунке 7:



order_id	dish_name	product_name
6406938	Pasta Carbonara	Potato Flour
5176700	French Fries with Sausages	Coriander Seed
6387094	Chilli con Carne	Agar
4959107	Chilli con Carne	Agar
3831708	Pork Sausage Roll	Oyster Sauce
2954664	Cheeseburger	Quinoa
3058248	Caprese Salad	Radish
5561845	Mushroom Risotto	Pear Juice
4346136	Caprese Salad	Radish
2629167	French Fries with Sausages	Coriander Seed

Рисунок 7 – Результат третьего SELECT-запроса

4) Какие клиенты какие залы чаще всего заказывают? (Представлен в листинге 12)

Листинг 12:

```

select client_name as Имя_клиента, (array_agg(hall_name))[1] as
Чаще_заказываемый_зал, (array_agg(replicas_amount))[1] as
Количество_праздников
from (select client_name, hall_name, count(hall_name) as replicas_amount
      from orders
           inner join orderedhall using(order_id)
           inner join client using(client_id)
           inner join hall using(hall_id)
      group by client_name, hall_name
      order by 1, 3 desc) query_1
group by client_name

```


Результат данного запроса представлен на рисунке 8:

Имя_клиента	Чаще_заказываемый_зал	Количество_праздников
Aufderhar L.S.	Blue Plate Pub	20
Bashirian R.M.	Golden Pub	20
Bednar B.X.	Belly Sushi	21
Bednar E.V.	PEW Brasserie	21
Beer P.I.	9329 BBQ	17
Block J.A.	Big Cafe	20
Bogisich Q.I.	Orange Grill & Tap	21
Boyle B.F.	Blue Plate Eats	22
Breitenberg X.G.	Blue Plate Grill	22
Collier I.A.	Fast Coffee	21

Рисунок 8 – Результат четвёртого SELECT-запроса

- 5) Для клиентов, у которых средняя конечная стоимость заказов выше средней конечной стоимости по суммам заказов клиентов (общей стоимости всех заказов клиентов), вывести имя, общую конечную стоимость всех заказов, количество заказов. (Представлен в листинге 13)

Листинг 13:

```
select client_name, count(order_id) as orders_amount,
round(sum(finish_cost)/count(order_id),2) as middle_finish_cost
from orders
    inner join client using(client_id)
group by client_name
having round(sum(finish_cost)/count(order_id),2) > (select round(avg(finish_cost),2)
as middle_cost
from orders)
```

Результат данного запроса представлен на рисунке 9:

Form2

Select 1 Select 2 Select 3 Select 4 **Select 5** Select 6 Select 7 Выход в Меню

Для клиентов у которых средняя конечная стоимость заказов выше средней конечной стоимости по суммам заказов клиентов (общей стоимости всех заказов клиентов), вывести имя, общую конечную стоимость всех заказов, количество заказов.

	client_name	orders_amount	middle_finish_cost
▶	Bogisich Q.I.	99951	71095,00
	Boyle B.F.	100141	71146,80
	Breitenberg X.G.	99980	71104,79
	Collier I.A.	100153	71074,89
	Crist Y.S.	99970	71133,33
	Cummerata I.P.	100200	71145,52
	Davis U.P.	100325	71107,88
	DuBuque Q.F.	100027	71065,97
	Emmerich F.N.	100666	71136,18
	Fisher J.J.	99680	71101,40

Рисунок 9 – Результат пятого SELECT-запроса

6) Какие клиенты, какие блюда чаще всего заказывают на Новый год?

(Представлен в листинге 14)

Листинг 14:

```
select client_name as Имя_клиента, (array_agg(dish_name))[1] as
Чаще_заказываемое_блюдо, (array_agg(replicas_amount))[1] as
Количество_заказов
from (select client_name, dish_name, count(dish_name) as replicas_amount
      from orders
        inner join orderreddish using(order_id)
        inner join client using(client_id)
        inner join dish using(dish_id)
       where banquet_type_id = (select banquet_type_id
                                from banquettype
                               where banquet_type_name like 'New Year%')

      group by client_name, dish_name
      order by 1, 3 desc) query_1

group by client_name
```

Результат данного запроса представлен на рисунке 10:

Form2

Select 1 Select 2 Select 3 Select 4 Select 5 **Select 6** Select 7 Выход в Меню

Какие клиенты, какие блюда чаще всего заказывают на Новый год?

	Имя_клиента	Чаще_заказываемое_блюдо	Количество_заказов
▶	Aufderhar L.S.	French Fries with Sausages	4
	Bashirian R.M.	Som Tam	5
	Bednar B.X.	Pasta Carbonara	5
	Bednar E.V.	Caprese Salad	5
	Beer P.I.	Cheeseburger	3
	Block J.A.	Pasta Carbonara	6
	Bogisich Q.I.	Katsu Curry	7
	Boyle B.F.	Kebab	5
	Breitenberg X.G.	Katsu Curry	5
	Collier I.A.	Mushroom Risotto	5

Рисунок 10 – Результат шестого SELECT-запроса

- 7) Вывести статистику по клиенту за определённый период (какие блюда для каких праздников чаще всего заказывались). (Представлен в листинге 15)

Листинг 15:

```
select  client_name,  banquet_type_name,  (array_agg(dish_name))[1]  as
Самое_заказываемое_блюдо,  (array_agg(dish_count))[1]  as
Количество_заказов_блюда
from(select  client_name,  banquet_type_name,  dish_name,  count(dish_id)  as
dish_count
from orders
      join client using(client_id)
      join BanquetType using(banquet_type_id)
      join orderreddish using(order_id)
      join dish using(dish_id)
where client_id = 101
      and date_time between '2017-08-05' and '2021-09-30'
group by client_name, banquet_type_name, dish_name
order by 2, 4 desc) query_1
group by client_name, banquet_type_name
```

Результат данного запроса представлен на рисунке 11:

Form2

Select 1

Select 2

Select 3

Select 4

Select 5

Select 6

Select 7

Выход в Меню

Вывести статистику по клиенту за определённый период (какие блюда для каких праздников чаще всего заказывались)

	client_name	banquet_type_name	Самое_заказываемое_блюдо	Количество_заказов_блюда
►	Cummerata I.P.	Birthday	Lasagne	4
	Cummerata I.P.	Christmas party	Chilli con Carne	6
	Cummerata I.P.	Easter	Meatballs with Sauce	6
	Cummerata I.P.	New Year Party	Stinky Tofu	5
	Cummerata I.P.	Wedding	Pasta Carbonara	11
	Cummerata I.P.	Company party	Som Tam	4

Рисунок 11 – Результат седьмого SELECT-запроса

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы было выполнено следующее:

- проведён анализ предметной области, выделены основные сущности и процессы;
- спроектирована база данных, созданы её инфологическая и даталогическая модели;
- написан скрипт для создания таблиц базы данных, а также написаны скрипты для триггеров и функций, обеспечивающих целостность и согласованность данных;
- разработан пользовательский интерфейс;
- разработаны и написаны сложные SQL-запросы для получения статистической информации;

В результате курсовой работы была реализована система, позволяющая агентству по организации банкетов вносить, изменять данные, а также собирать статистику по заказам. Данная система предоставляет доступ к необходимой информации и помогает улучшить работу агентства.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Скворцова М. А., Замула М. И. Методические указания по выполнению курсовой работы по курсу «Базы данных» [Электронный ресурс] URL:

https://e-learning.bmstu.ru/iu6/pluginfile.php/19147/mod_resource/content/1/%D0%9C%D0%A0_%D0%9A%D0%A0_%D0%91%D0%94_2%20%D0%BA%D1%83%D1%80%D1%81.pdf

2. Документация Faker Ruby [Электронный ресурс] URL:

<https://github.com/faker-ruby/faker>

3. Скворцова М. А., Лапшин А. В. Методические указания к лабораторным работам по курсу «Базы данных» [Электронный ресурс] URL:

https://e-learning.bmstu.ru/iu6/pluginfile.php/18672/mod_resource/content/1/%D0%9C%D0%A0_%D0%9B%D0%A0%D0%A3_%D0%91%D0%94.pdf