«АКАКДЕМИЯ TOP»

**ПРОЕКТ**

По дисциплине: «Технология доступа к базам данных ADO.NET»

На тему: «Система управления заказами службы доставки еды.»

Выполнили: Михайлов, Кулягин(РПО 28/2)

Санкт-Петербург, 2025

**ВВЕДЕНИЕ**

**Цель проекта:**

Разработать систему управления заказами службы доставки еды, позволяющий отображать сам заказ, статус заказа, пользователей.

**Задачи проекта:**

1. Разработать саму базу данных,
2. Заполнить их нужными данными для работы программы,
3. Создать интерфейс позволяющий добавлять заказ,
4. Смотреть заказ, проверить его статус, редактировать
5. Провести тестирование.

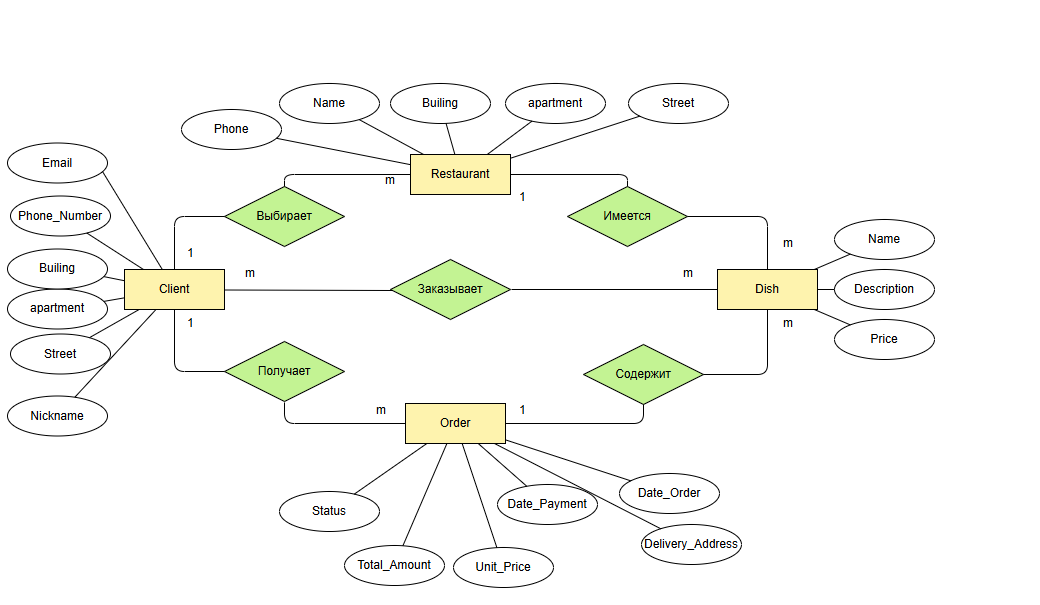
**Язык программирования:** C#

**СУБД:** SQLite

**Программа для реализации:** MS Visual Studio

**Объекты базы данных:** Client, Restaurant, Dish, Order, OrderStatus, OrderItem.

**ER Диаграмма**



**Описание ER-Диаграммы:**

**1. Client**

Client\_Id - уникальный идентификатор пользователя

Email - эл. почта

Building - номер здания

apartment - номер квартиры

street - улица

Nickname - имя пользователя

**2. Restaurant**

Restaurant\_Id - уникальный идентификатор ресторана

Name - название

Building - номер здания

Apartment - номер квартиры

Street - улица

Phone - номер телефона

**3. Dish**

Dish\_Id - уникальный идентификатор блюда

Name - название

Description - описание

Price - цена

**4. Order**

Order\_Id - уникальный идентификатор чека

Dish\_Id – уникальный идентификатор блюда

Status - Статус заказа

Total\_Amount - общая сумма заказа

Unit\_Price - цена каждой позиции отдельно

Date\_Payment - дата оплаты

Date\_Order - дата оформления заказа

Delivery\_Address - адрес доставки

**Связи между сущностями:**

Client M:M Dish - много клиентов может заказать разные блюда

Client 1:M Order - клиент может иметь много чеков, но не наоборот

Client 1:M Restaurant - клиент выбирает ресторан

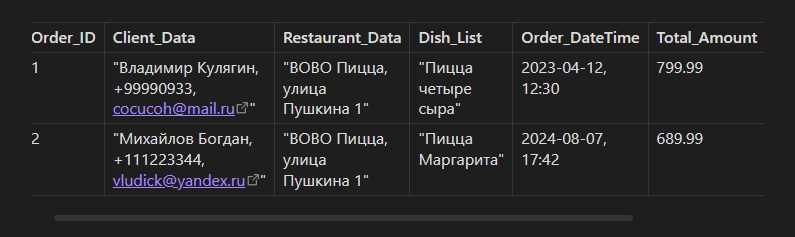
Restaurant 1:M Dish - Один ресторан имеет много блюд

Order 1:M Dish - В одном чеке может быть много блюд

**ПРОЦЕСС НОРМАЛИЗАЦИИ БАЗЫ ДАННЫХ**

**1. НЕНОРМАЛИЗОВАННАЯ БАЗА ДАННЫХ**

На начальном этапе все данные хранились в одной большой таблице. Это приводило к множеству проблем.



**Проблемы ненормализованной формы:**

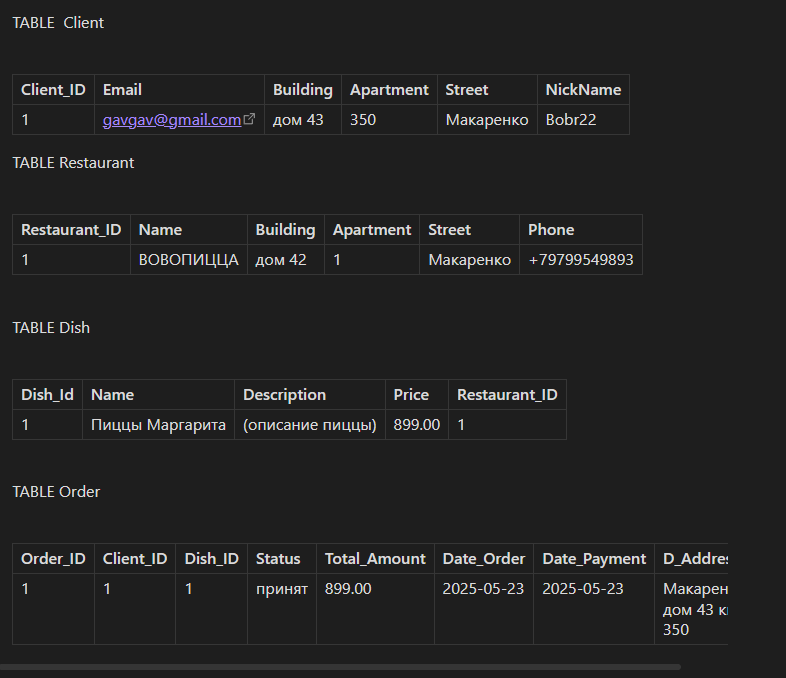
1. **Избыточность данных:** Данные о клиенте и ресторане дублировались в каждом заказе.
2. **Аномалии обновления:** При смене номера телефона ресторана "ВОВО Пицца" необходимо было обновить его в каждой строке таблицы.
3. **Аномалии удаления:** Если удалить все заказы для ресторана "ВОВО Пицца", то информация о самом ресторане безвозвратно терялась.
4. **Аномалии добавления:** Нельзя было добавить в систему новый ресторан, пока ему не сделали ни одного заказа.

**2. 1NF**

Цель: Устранить повторяющиеся группы, обеспечить атомарность данных.

Шаги:

1. Выделить отдельные сущности
2. Создать первичные ключи
3. Убедиться в атомарности всех полей



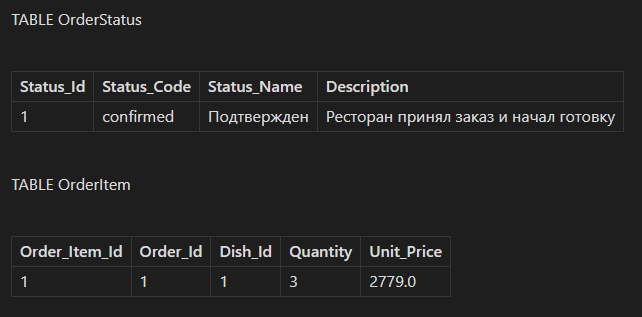
Проблема после 1NF: В таблице Order поле Dish\_Id нарушает 1NF - один заказ может содержать несколько блюд.

**3. 2NF**

Цель: Устранить частичные зависимости от составного ключа.

Шаги:

1. Вынести статусы в отдельную таблицу
2. Создать таблицу для позиций заказа (OrderItem)



Что улучшилось:

1. Статусы вынесены в отдельную таблицу
2. Один заказ теперь может содержать несколько блюд через OrderItem
3. Устранены частичные зависимости

**4. 3NF**

На этом этапе мы проверили, что в получившихся таблицах неключевые поля зависят непосредственно от первичного ключа, а не друг от друга.

* **Нет избыточности данных** - каждая сущность в своей таблице
* **Минимизированы аномалии обновления** - изменение адреса клиента не влияет на исторические заказы
* **Правильные связи** - внешние ключи обеспечивают целостность
* **Отсутствие транзитивных зависимостей** - все атрибуты зависят только от первичного ключа своей таблицы

**БАЗА ДАННЫХ:**

**СТАТУС (STATUS)**

CREATE TABLE OrderStatus (

Status\_Id INTEGER PRIMARY KEY AUTOINCREMENT,

Status\_Code TEXT UNIQUE NOT NULL,

Status\_Name TEXT NOT NULL,

Description TEXT

);

**КЛИЕНТ (CLIENT)**

CREATE TABLE Client (

Client\_Id INTEGER PRIMARY KEY AUTOINCREMENT,

Email TEXT NOT NULL UNIQUE,

Building TEXT NOT NULL,

Apartment TEXT NOT NULL,

Street TEXT NOT NULL,

Nickname TEXT NOT NULL,

);

**РЕСТОРАН (RESTAURANT)**

CREATE TABLE Restaurant (

Restaurant\_Id INTEGER PRIMARY KEY AUTOINCREMENT,

Name TEXT NOT NULL,

Building TEXT NOT NULL,

Apartment TEXT,

Street TEXT NOT NULL,

Phone TEXT NOT NULL,

);

**БЛЮДО (DISH)**

CREATE TABLE Dish (

Dish\_Id INTEGER PRIMARY KEY AUTOINCREMENT,

Name TEXT NOT NULL,

Description TEXT,

Price REAL NOT NULL CHECK (Price >= 0),

Restaurant\_Id INTEGER NOT NULL,

FOREIGN KEY (Restaurant\_Id) REFERENCES Restaurant(Restaurant\_Id) ON DELETE CASCADE

);

**ЗАКАЗ (ORDER)**

CREATE TABLE "Order" (

Order\_Id INTEGER PRIMARY KEY AUTOINCREMENT,

Client\_Id INTEGER NOT NULL,

Status\_Id INTEGER NOT NULL DEFAULT 1,

Restaurant\_Id INTEGER NOT NULL,

Total\_Amount REAL NOT NULL CHECK (Total\_Amount >= 0),

Date\_Order DATETIME DEFAULT CURRENT\_TIMESTAMP,

Date\_Payment DATETIME,

Delivery\_Address TEXT NOT NULL,

Created\_At DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (Client\_Id) REFERENCES Client(Client\_Id) ON DELETE CASCADE,

FOREIGN KEY (Status\_Id) REFERENCES OrderStatus(Status\_Id),

FOREIGN KEY (Restaurant\_Id) REFERENCES Restaurant(Restaurant\_Id)

);

**СОСТАВ ЗАКАЗА (ORDERITEM)**

CREATE TABLE OrderItem (

Order\_Item\_Id INTEGER PRIMARY KEY AUTOINCREMENT,

Order\_Id INTEGER NOT NULL,

Dish\_Id INTEGER NOT NULL,

Quantity INTEGER NOT NULL DEFAULT 1 CHECK (Quantity > 0),

Unit\_Price REAL NOT NULL CHECK (Unit\_Price >= 0),

FOREIGN KEY (Order\_Id) REFERENCES "Order"(Order\_Id) ON DELETE CASCADE,

FOREIGN KEY (Dish\_Id) REFERENCES Dish(Dish\_Id) ON DELETE CASCADE

);