|  |  |
| --- | --- |
|  | **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 «Фундаментальные науки»

КАФЕДРА «Математическое моделирование»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент Ириневич Сергей Георгиевич

*фамилия, имя, отчество*

Группа ФН12-21М

Тип практики научно-исследовательская работа

Название предприятия МГТУ им. Н.Э. Баумана

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Ириневич С.Г.

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Вишняков И.Э.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2024 г.*

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)

Кафедра «Математическое моделирование» (ФН12-21М)

**ЗАДАНИЕ**

**на прохождение производственной практики**

на предприятии МГТУ им. Н.Э. Баумана

Студент Ириневич Сергей Георгиевич / Ириневич С.Г. / ФН12-21М

(фамилия, имя, отчество; инициалы; индекс группы)

Во время прохождения производственной практики студент должен:

1. Построить модель «сущность-связь» для выбранной предметной области.

2. Преобразовать модель «сущность-связь» в реляционную модель.

3. Реализовать БД средствами СУБД SQL Server.

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Руководитель практики от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /** Ириневич С.Г.

(подпись, дата) (Фамилия И.О.)

**Оглавление**

[Введение 4](#_Toc168536410)

[1. Проектирование ER-модели 5](#_Toc168536411)

[2. Преобразование ER-модели в реляционную 8](#_Toc168536412)

[3. Реализация модели средствами SQL Server 14](#_Toc168536413)

[Заключение 18](#_Toc168536414)

[Приложение А 19](#_Toc168536415)

# **Введение**

База данных и система управления базой данных (СУБД) являются важными компонентами информационных систем предприятия. Процесс создания базы данных включает несколько шагов: от словесного описания информации до формализации объектов в рамках определенной модели. В общем случае выделяют следующие этапы проектирования: анализ и описание предметной области, концептуальное моделирование, разработку логической структуры и физическую реализацию базы данных.

Цель данной научно-исследовательской работы – разработка модели «сущность-связь» и её реализация в SQL Server.

Для достижения поставленной цели необходимо решить следующие задачи:

* Выбрать предметную область, соответствующую 4-5 сущностям.
* Сформировать требования к предметной области.
* Создать модель «сущность-связь» для предметной области с обоснованием выбора кардинальных чисел связей.
* Преобразовать созданную модель «сущность-связь» в реляционную модель согласно процедуре преобразования.
* Обосновать выбор типов данных, ключей, правил обеспечения ограничений минимальной кардинальности.
* Создать базу данных, используя средства SQL Server.
* Составить запросы к базе данных, используя возможности языка Transact-SQL.

# **Проектирование ER-модели**

В качестве предметной области была выбрана система доставки еды из кафе и ресторанов. Примерами таких систем, действующих на территории РФ, являются Яндекс.Еда, Самокат и Delivery Club. Клиент делает заказ из ресторана, после чего этот заказ ему доставляет курьер. Каждый заказ делает только один клиент и доставляется только одним курьером.

На основе описанной предметной области была построена модель сущность-связь (рис. 1), включающая шесть сущностей:

* *Клиент* – сущность клиента с идентификатором «Телефон» и атрибутами: «Фамилия», «Имя», «Отчество»;
* *Заказ* – сущность, являющаяся абстракцией сделанного заказа с идентификатором «№ заказа» и атрибутами: «Адрес доставки», «Стоимость заказа», «Стоимость доставки», «Дата и время заказа», «Дата и время доставки»;
* *Курьер* – сущность курьера с идентификатором «Телефон» и атрибутами: «Фамилия», «Имя», «Отчество», «З./п.»;
* *Ресторан* – сущность ресторана (конкретной торговой точки в случае сети ресторанов) с составным идентификатором, состоящим из атрибутов «Наименование организации» и «Адрес», а также атрибутами «Телефон» и «e-mail»;
* *Пункт меню ресторана* – идентификационно-зависимая сущность, являющаяся абстракцией товара из меню ресторана с составным идентификатором, состоящим из атрибутов «Наименование организации», «Адрес» и «Название товара», а также атрибутами «Цена», «Состав», «Пищевая ценность», «Энергетическая ценность», «Масса нетто»;
* *Пункт заказа* – идентификационно-зависимая сущность, являющаяся абстракцией товара, выбранного для заказа, с составным идентификатором, состоящим из атрибутов «№ заказа», «Наименование организации», «Адрес» и «Название товара», а также имеющая атрибуты «Цена» и «Количество».

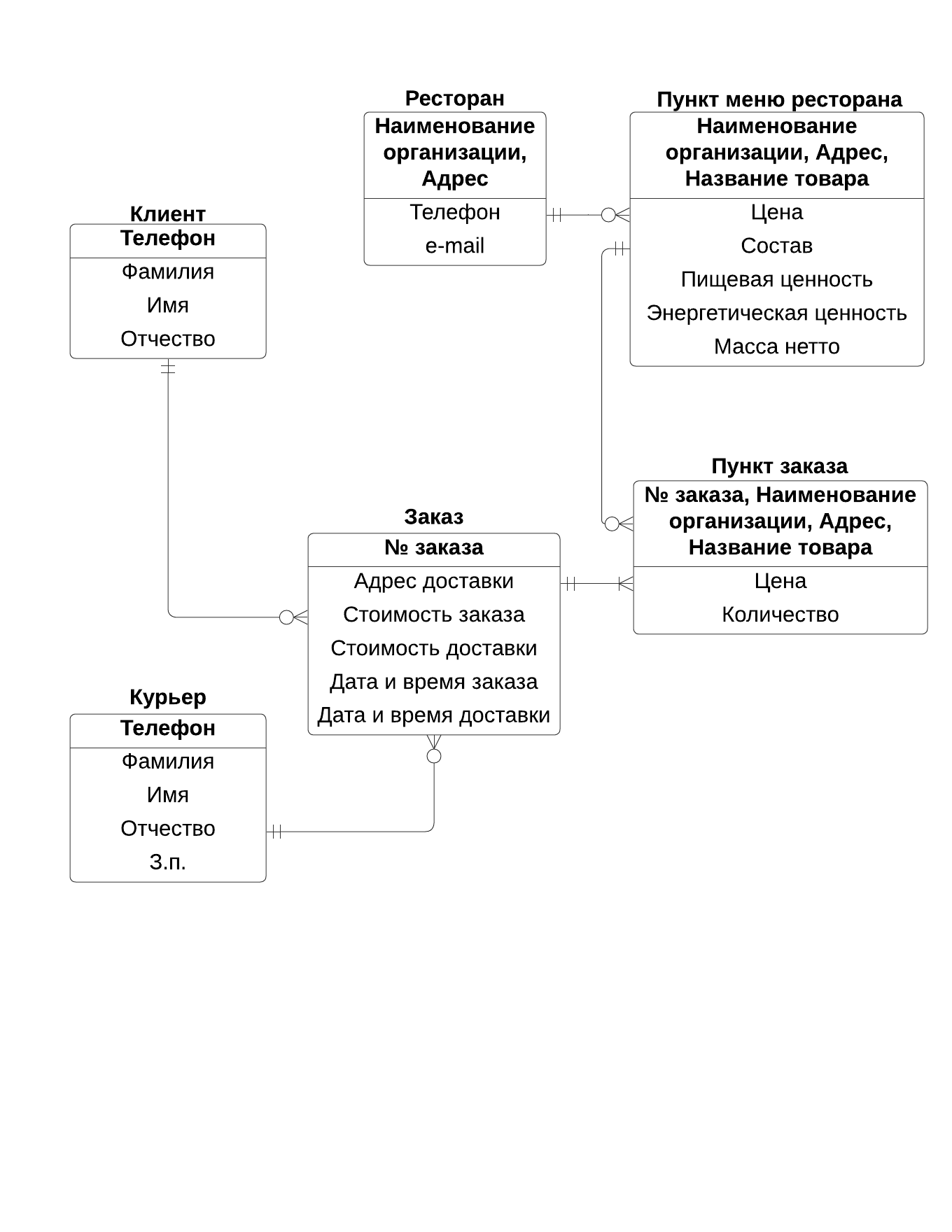


Рис. 1. Модель «Сущность – связь»

Так как с течением времени клиент может сделать много заказов, но каждый заказ соответствует одному и только одному клиенту, связь между сущностями «Клиент» и «Заказ» является связью типа «один-ко-многим», а минимальные кардинальные числа равны 1 и 0 соответственно.

Так как каждый курьер во время своей работы может относить много заказов, но каждый заказ несёт один и только один курьер, связь между сущностями «Курьер» и «Заказ» является связью типа «один-ко-многим», а минимальные кардинальные числа равны 1 и 0.

В связи с тем, что каждый заказ может иметь несколько наименований, причём пустой заказ не имеет смысла, но каждый пункт относится к конкретному заказу, связь между сущностями «Заказ» и «Пункт заказа» является связью типа «один-ко-многим», а минимальные кардинальные числа равны 1 и 1 соответственно.

Поскольку каждый пункт меню ресторана может попасть в большое количество заказов, но каждый пункт заказа связан с конкретным пунктом из меню, связь между сущностями «Пункт меню ресторана» и «Пункт заказа» является связью типа «один-ко-многим», а минимальные кардинальные числа равны 1 и 0 соответственно.

Так как каждый ресторан содержит в меню несколько наименований, но каждый пункт меню относится к конкретному ресторану (конкретной точке сети ресторанов, поскольку цена в разных точках может быть разной), связь между сущностями «Ресторан» и «Пункт меню ресторана» является связью типа «один-ко-многим», а минимальные кардинальные числа равны 1 и 0 соответственно.

# **Преобразование ER-модели в реляционную**

В соответствии с процедурой преобразования ER-модели в реляционную была получена модель, представленная на рис. 2.

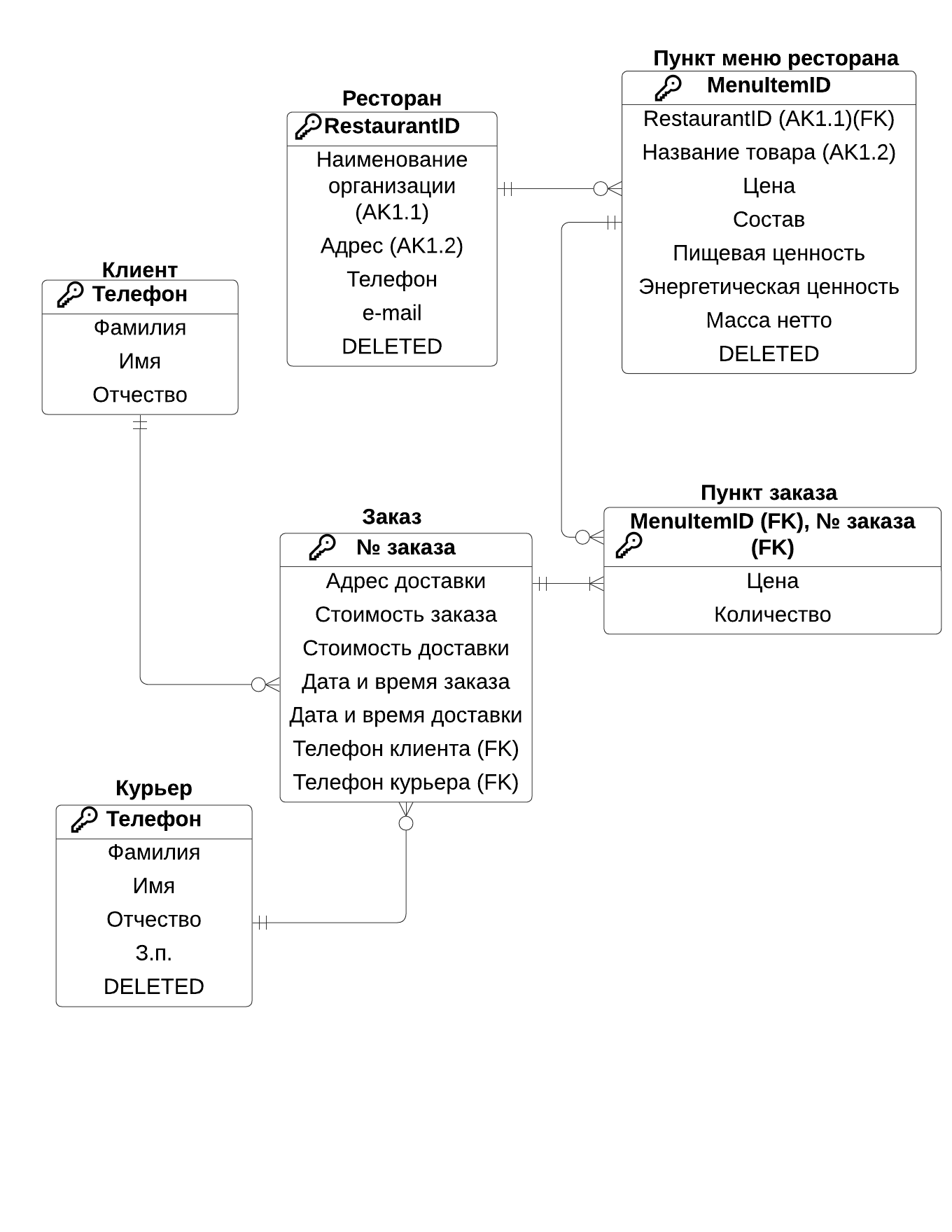


Рис. 2. Реляционная модель

В отношении «Ресторан» был добавлен суррогатный ключ «RestaurantID», так как исходный первичный ключ является нагроможденным и неудобным в использовании (мы его пометим как составной ключ-кандидат). По той же причине был добавлен суррогатный ключ «MenuItemID» в отношение «Пункт меню ресторана» и изменён первичный ключ отношения «Пункт заказа».

Типы данных столбцов и их NULL-статусы, а также типы ключей приведены в таблицах 1-6.

Таблица 1. Типы данных столбцов и NULL-статусы отношения «Клиент»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип данных | Ключ | NULL status | Замечания |
| Телефон | CHAR(12) | Primary Key | NOT NULL | Начинается с +7 |
| Фамилия | NVARCHAR(50) | - | NOT NULL |  |
| Имя | NVARHCAR(35) | - | NOT NULL |  |
| Отчество | NVARCHAR(50) | - | NULL |  |

Таблица 2. Типы данных столбцов и NULL-статусы отношения «Курьер»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип данных | Ключ | NULL status | Замечания |
| Телефон | CHAR(12) | Primary Key | NOT NULL | Начинается с +7 |
| Фамилия | NVARCHAR(50) | - | NOT NULL |  |
| Имя | NVARHCAR(35) | - | NOT NULL |  |
| Отчество | NVARCHAR(50) | - | NULL |  |
| З.п. | FLOAT | - | NULL |  |
| DELETED | BIT | - | NOT NULL | 1 – Уволен  0 – Работает |

Таблица 3. Типы данных столбцов и NULL-статусы отношения «Заказ»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип данных | Ключ | NULL status | Замечания |
| № заказа | BIGINT | Primary Key | NOT NULL | Суррогатный ключ |
| Адрес доставки | NVARCHAR(200) | - | NOT NULL |  |
| Стоимость заказа | FLOAT | - | NOT NULL |  |
| Стоимость доставки | FLOAT | - | NOT NULL |  |
| Дата и время заказа | DATETIME | - | NULL |  |
| Дата и время доставки | DATETIME | - | NULL |  |
| Телефон клиента | CHAR(12) | Foreign Key | NOT NULL |  |
| Телефон курьера | CHAR(12) | Foreign Key | NOT NULL |  |

Таблица 4. Типы данных столбцов и NULL-статусы отношения «Ресторан»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип данных | Ключ | NULL status | Замечания |
| RestaurantID | INT | Primary Key | NOT NULL | Суррогатный ключ |
| Наименование организации | NVARCHAR(100) | AK1.1 | NOT NULL |  |
| Адрес | NVARCHAR(200) | AK1.2 | NOT NULL |  |
| Телефон | CHAR(12) | - | NULL | Начинается с +7 |
| e-mail | VARCHAR(254) | - | NULL |  |
| DELETED | BIT | - | NOT NULL | 1 – ресторан закрылся |

Таблица 5. Типы данных столбцов и NULL-статусы отношения «Пункт меню ресторана»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип данных | Ключ | NULL status | Замечания |
| MenuItemID | INT | Primary Key | NOT NULL | Суррогатный ключ |
| RestaurantID | INT | AK1.1, Foreign Key | NOT NULL |  |
| Название товара | NVARCHAR(80) | AK1.2 | NOT NULL |  |
| Цена | FLOAT | - | NOT NULL |  |
| Состав | NVARCHAR(1000) | - | NULL |  |
| Пищевая ценность | NVARCHAR(1000) | - | NULL |  |
| Энергетическая ценность | INT | - | NULL | В ккал |
| Масса нетто | INT | - | NULL | В граммах |
| DELETED | BIT | - | NOT NULL | 1 – пункт удалён из меню ресторана |

Таблица 6. Типы данных столбцов и NULL-статусы отношения «Пункт заказа»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип данных | Ключ | NULL status | Замечания |
| MenuItemID | INT | Primary Key, Foreign Key | NOT NULL |  |
| № заказа | BIGINT | Primary Key, Foreign Key | NOT NULL |  |
| Цена | FLOAT | - | NOT NULL |  |
| Количество | INT | - | NOT NULL |  |

Правила ограничения минимальной кардинальности представлены в таблицах 7-11.

Таблица 7. Связь Клиент – Заказ (M – O)

|  |  |  |
| --- | --- | --- |
| Операция | Действие над родительской таблицей (PK) | Действие над дочерней таблицей (FK) |
| Вставка | Без ограничений | Подбор родителя |
| Изменение | Каскадное обновление | Запрет |
| Удаление | Позволено, если клиент не имеет ни одного заказа, иначе – запрет | Запрет |

Таблица 8. Связь Курьер – Заказ (M – O)

|  |  |  |
| --- | --- | --- |
| Операция | Действие над родительской таблицей (PK) | Действие над дочерней таблицей (FK) |
| Вставка | Без ограничений | Подбор родителя |
| Изменение | Каскадное обновление | Запрет |
| Удаление | Поле «DELETED» меняет своё значение на «1» | Запрет |

Таблица 9. Связь Заказ – Пункт заказа (M – M)

|  |  |  |
| --- | --- | --- |
| Операция | Действие над родительской таблицей (PK) | Действие над дочерней таблицей (FK) |
| Вставка | Создание заказа влечёт за собой создание соответствующих пунктов этого заказа | Создаётся при создании заказа |
| Изменение | Запрет (суррогатный ключ) | Запрет |
| Удаление | Запрет | Запрет |

Таблица 10. Связь Ресторан – Пункт меню ресторана (M – O)

|  |  |  |
| --- | --- | --- |
| Операция | Действие над родительской таблицей (PK) | Действие над дочерней таблицей (FK) |
| Вставка | Без ограничений | Подбор родителя |
| Изменение | Запрет (суррогатный ключ) | Запрет |
| Удаление | Поле «DELETED» меняет своё значение на «1». Также меняется значение поля «DELETED» на «1» у всех дочерних таблиц. | Поле «DELETED» меняет своё значение на «1» |

Таблица 11. Связь Пункт меню ресторана – Пункт заказа (M – O)

|  |  |  |
| --- | --- | --- |
| Операция | Действие над родительской таблицей (PK) | Действие над дочерней таблицей (FK) |
| Вставка | При корректном значении внешнего ключа | Подбор родителя |
| Изменение | Запрет (суррогатный ключ) | Запрет |
| Удаление | Поле «DELETED» меняет своё значение на «1» | Запрет |

# **Реализация модели средствами SQL Server**

Реализация описанной в предыдущих разделах базы данных была выполнена с использованием языка запросов Transact-SQL. Подробно с ней можно ознакомиться в Приложении.

Обеспечение ограничений минимальной кардинальности было выполнено при помощи предложений ON DELETE и ON UPDATE ограничений FOREIGN KEY, а также при помощи триггеров и транзакций.

Создадим несколько запросов к построенной базе данных, с помощью которых можно обращаться к данным, а заодно продемонстрируем возможности языка Transact-SQL. Для демонстрации в таблицы были внесены некоторые данные.

При этом были продемонстрированы такие возможности языка, как

* Удаление повторяющихся записей (DISTINCT);
* Выбор, упорядочивание и именование полей (создание псевдонимов для полей и таблиц);
* Соединение таблиц (INNER JOIN, LEFT JOIN, RIGHT JOIN и FULL OUTER JOIN);
* Условия выбора записей (NULL, LIKE, BETWEEN, IN, EXISTS);
* Сортировка записей (ORDER BY – ASC, DESC);
* Группировка записей (GROUP BY + HAVING, использование функций агрегирования – COUNT, AVG, SUM, MIN, MAX);
* Объединение результатов нескольких запросов (UNION, UNION ALL, EXCEPT, INTERSECT);
* Вложенные запросы.

Выполним следующие задания:

1. Вывести телефоны и ФИО курьеров, которые не доставили ни одного заказа:

SELECT c.[phone], c.first\_name, c.second\_name, c.patronymic FROM [order] o RIGHT JOIN [courier] c

ON c.[phone] = o.courier\_phone

WHERE o.orderID IS NULL;

1. Вывести рестораны, в меню которых есть позиции, не попавшие ни в один заказ, и количество таких позиций:

SELECT r.organization\_name, r.[address], COUNT(\*) AS [count] FROM [restaurant] r, menu\_item m LEFT JOIN order\_item o

ON m.MenuItemID = o.MenuItemID

WHERE r.restaurantID = m.restaurantID AND o.MenuItemID IS NULL

GROUP BY r.organization\_name, r.[address];

1. Показать данные о заказах вместе с клиентами, сделавшими эти заказы, и курьерами, которые их доставляли:

SELECT cl.[phone] AS client\_phone, cl.second\_name, cl.first\_name, o.[address], o.order\_cost, o.delivery\_cost,

co.[phone] AS courier\_phone, co.second\_name, co.first\_name

FROM client cl FULL OUTER JOIN [order] o

ON cl.[phone] = o.client\_phone FULL OUTER JOIN courier co

ON co.[phone] = o.courier\_phone;

1. Вывести среднюю стоимость «заказ+доставка» по каждому курьеру, отсортировав по убыванию:

SELECT c.[phone], AVG(o.order\_cost + o.delivery\_cost) AS mean\_cost FROM [courier] c INNER JOIN [order] o

ON c.[phone] = o.courier\_phone

GROUP BY c.[phone]

ORDER BY mean\_cost DESC;

1. Вывести данные курьеров с зарплатой между 38000 и 42000, отсортировав по зарплате по возрастанию:

SELECT \* FROM [courier]

WHERE salary BETWEEN 38000 AND 42000

ORDER BY salary ASC;

1. Вывести цену самого дешёвого товара в каждой из торговых точек, отсортировав по возрастанию:

SELECT r.organization\_name, r.[address], MIN(m.cost) AS min\_cost FROM [restaurant] r INNER JOIN menu\_item m

ON r.restaurantID = m.restaurantID

GROUP BY r.organization\_name, r.[address]

ORDER BY min\_cost ASC;

1. Вывести суммарное количество калорий в каждом заказе, выбрав те заказы, для которых это значение больше 1000:

SELECT o.orderID, o.client\_phone, c.second\_name, c.first\_name, SUM(m.energy\*i.[count]) AS sum\_energy

FROM client c, [order] o, order\_item i, menu\_item m

WHERE c.[phone] = o.client\_phone AND o.orderID = i.orderID AND i.MenuItemID = m.MenuItemID

GROUP BY o.orderID, o.client\_phone, c.second\_name, c.first\_name

HAVING SUM(m.energy\*i.[count]) > 1000;

1. Вывести самую высокую зарплату курьера:

SELECT MAX(salary) AS max\_salary FROM courier;

1. Вывести данные курьеров, которые доставляли заказы быстрее 20 минут:

SELECT DISTINCT \* FROM [courier]

WHERE [phone] IN

(SELECT courier\_phone FROM [order]

WHERE DATEDIFF(MINUTE, order\_datetime, delivery\_datetime) < 20);

1. Вывести данные клиентов, которые совершили хотя бы один заказ:

SELECT \* FROM [client] c

WHERE EXISTS(SELECT \* FROM [order] o WHERE c.[phone] = o.client\_phone);

1. Вывести данные курьеров, у которых фамилия начинается на «Ш» или имя начинается на «Э»:

SELECT \* FROM [courier]

WHERE second\_name LIKE N'Ш%'

UNION

SELECT \* FROM [courier]

WHERE first\_name LIKE N'Э%';

1. Вывести данные клиентов с фамилией «Иванов(-а)», имя которых начинается на «В»:

SELECT \* FROM [client]

WHERE second\_name LIKE N'Иванов%'

INTERSECT

SELECT \* FROM [client]

WHERE first\_name LIKE N'В%';

1. Показать пункты меню ресторанов, которые дешевле 120 рублей и калорийность которых не выше средней:

SELECT \* FROM menu\_item

WHERE cost < 120

EXCEPT

SELECT \* FROM menu\_item

WHERE energy > (SELECT AVG(energy) FROM menu\_item);

1. Вывести все рестораны, которые называются «McDonald’s», и те, у которых рабочая почта в Яндексе:

SELECT \* FROM [restaurant]

WHERE organization\_name = N'McDonald''s'

UNION ALL

SELECT \* FROM [restaurant]

WHERE e\_mail LIKE '%@yandex.ru';

# **Заключение**

В данной работе была определена предметная область для создания базы данных и установлены требования к ней. Далее была разработана ER-модель базы данных "Доставка еды из кафе и ресторанов" и обоснован выбор кардинальных чисел связи. После этого было произведено преобразование спроектированной ER-модели в реляционную модель с обоснованием типов данных, ключей, правил обеспечения минимальной кардинальности. Изучены основные возможности языка запросов Transact-SQL. База данных реализована с использованием СУБД SQL Server. В базе данных реализованы все запросы, определенные на этапе проектирования, а также описаны механизмы обеспечения целостности данных в выбранной системе управления базами данных. Таким образом, все задачи научно-исследовательской работы выполнены, поэтому её цель можно считать достигнутой.

# **Приложение А**

USE tempdb;

GO

DECLARE @SQL NVARCHAR(1000);

IF EXISTS (SELECT 1 FROM sys.databases WHERE [name] = N'food\_delivery')

BEGIN

SET @SQL = N'USE [food\_delivery];

ALTER DATABASE food\_delivery SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

USE [tempdb];

DROP DATABASE food\_delivery;';

EXEC (@SQL);

END;

GO --Удалить базу данных, если она существует

USE master;

GO

CREATE DATABASE food\_delivery

ON (NAME='food\_delivery\_dat', FILENAME='C:\databases\food\_delivery\food\_delivery\_dat.mdf',

SIZE=10, MAXSIZE=50, FILEGROWTH=5%)

LOG ON (NAME='food\_delivery\_log', FILENAME='C:\databases\food\_delivery\food\_delivery\_log.ldf',

SIZE=10, MAXSIZE=50, FILEGROWTH=5%);

GO

USE food\_delivery;

GO

CREATE TABLE [client] (

[phone] CHAR(12) NOT NULL PRIMARY KEY,

second\_name NVARCHAR(50) NOT NULL,

first\_name NVARCHAR(35) NOT NULL,

patronymic NVARCHAR(50) NULL,

CONSTRAINT constraint\_client1 CHECK ([phone] LIKE '+7%')

);

GO

CREATE TABLE [courier] (

[phone] CHAR(12) NOT NULL PRIMARY KEY,

second\_name NVARCHAR(50) NOT NULL,

first\_name NVARCHAR(35) NOT NULL,

patronymic NVARCHAR(50) NULL,

salary FLOAT NULL,

[deleted] BIT NOT NULL DEFAULT 0,

CONSTRAINT constraint\_courier1 CHECK ([phone] LIKE '+7%')

);

GO

CREATE TABLE [order] (

orderID BIGINT NOT NULL PRIMARY KEY,

[address] NVARCHAR(200) NOT NULL,

order\_cost FLOAT NOT NULL,

delivery\_cost FLOAT NOT NULL,

order\_datetime DATETIME NULL,

delivery\_datetime DATETIME NULL,

client\_phone CHAR(12) NOT NULL,

courier\_phone CHAR(12) NOT NULL,

CONSTRAINT constraint\_order1 FOREIGN KEY (client\_phone)

REFERENCES [client]([phone]) ON UPDATE CASCADE ON DELETE NO ACTION,

CONSTRAINT constraint\_order2 FOREIGN KEY (courier\_phone)

REFERENCES [courier]([phone]) ON UPDATE CASCADE

);

GO

CREATE TABLE [restaurant] (

restaurantID INT IDENTITY(1, 1) NOT NULL PRIMARY KEY,

organization\_name NVARCHAR(100) NOT NULL,

[address] NVARCHAR(200) NOT NULL,

[phone] CHAR(12) NULL,

e\_mail VARCHAR(254) NULL,

[deleted] BIT NOT NULL DEFAULT 0,

CONSTRAINT constraint\_restaurant1 CHECK ([phone] LIKE '+7%'),

CONSTRAINT constraint\_restaurant2 UNIQUE (organization\_name, [address])

);

GO

CREATE TABLE menu\_item (

MenuItemID INT IDENTITY(1, 1) NOT NULL PRIMARY KEY,

restaurantID INT NOT NULL,

product\_name NVARCHAR(80) NOT NULL,

cost FLOAT NOT NULL,

ingredients NVARCHAR(1000) NULL,

nutritional\_value NVARCHAR(1000) NULL,

energy INT NULL,

[weight] INT NULL,

[deleted] BIT NOT NULL DEFAULT 0,

CONSTRAINT constraint\_menu\_item1 UNIQUE (restaurantID, product\_name),

CONSTRAINT constraint\_menu\_item2 FOREIGN KEY (restaurantID) REFERENCES [restaurant](restaurantID)

);

GO

CREATE TABLE order\_item(

MenuItemID INT NOT NULL,

orderID BIGINT NOT NULL,

cost FLOAT NOT NULL,

[count] INT NOT NULL,

CONSTRAINT constraint\_order\_item1 PRIMARY KEY (MenuItemID, orderID),

CONSTRAINT constraint\_order\_item2 FOREIGN KEY (MenuItemID) REFERENCES menu\_item(MenuItemID),

CONSTRAINT constraint\_order\_item3 FOREIGN KEY (orderID) REFERENCES [order](orderID)

);

GO

CREATE TRIGGER order\_trigger1

ON [order]

AFTER DELETE

AS

BEGIN

RAISERROR(N'Невозможно удалить заказ (Business rule)', 16, 1);

ROLLBACK;

END;

GO

CREATE TRIGGER order\_trigger2

ON [order]

AFTER UPDATE

AS

BEGIN

IF UPDATE(client\_phone) OR UPDATE(courier\_phone)

BEGIN

RAISERROR(N'Невозможно изменить внешний ключ заказа', 16, 1);

ROLLBACK;

END;

IF UPDATE(orderID)

BEGIN

RAISERROR(N'Невозможно изменить первичный ключ заказа', 16, 1);

ROLLBACK;

END;

END;

GO

CREATE TRIGGER courier\_trigger1

ON [courier]

INSTEAD OF DELETE

AS

BEGIN

UPDATE [courier] SET [deleted] = 1 WHERE [courier].phone IN (SELECT phone FROM deleted);

END;

GO

CREATE TRIGGER order\_item\_trigger1

ON order\_item

AFTER DELETE

AS

BEGIN

RAISERROR(N'Невозможно удалить пункт заказа (Business rule)', 16, 1);

ROLLBACK;

END;

GO

CREATE TRIGGER order\_item\_trigger2

ON order\_item

AFTER UPDATE

AS

BEGIN

IF UPDATE(MenuItemID) OR UPDATE(orderID)

BEGIN

RAISERROR(N'Невозможно изменить внешний ключ пункта заказа', 16, 1);

ROLLBACK;

END

END;

GO

CREATE TRIGGER menu\_item\_trigger1

ON menu\_item

AFTER UPDATE

AS

BEGIN

IF UPDATE(restaurantID)

BEGIN

RAISERROR(N'Невозможно изменить внешний ключ пункта меню ресторана', 16, 1);

ROLLBACK;

END

END;

GO

CREATE TRIGGER restaurant\_trigger1

ON [restaurant]

INSTEAD OF DELETE

AS

BEGIN

UPDATE [restaurant] SET [deleted] = 1 WHERE restaurantID IN (SELECT restaurantID FROM deleted);

UPDATE menu\_item SET [deleted] = 1 WHERE restaurantID IN (SELECT restaurantID FROM deleted);

END;

GO

CREATE TRIGGER menu\_item\_trigger2

ON menu\_item

INSTEAD OF DELETE

AS

BEGIN

UPDATE menu\_item SET [deleted] = 1 WHERE MenuItemID IN (SELECT MenuItemID FROM deleted);

END;

GO

INSERT INTO [client]([phone], second\_name, first\_name, patronymic) VALUES

('+78005553535', N'Иванов', N'Владимир', N'Михайлович'),

('+79163654400', N'Медведев', N'Валерий', N'Дмитриевич'),

('+79807432247', N'Пупкин', N'Василий', N'Иванович'),

('+79223411268', N'Иванова', N'Мария', N'Ивановна'),

('+79175552211', N'Кириллова', N'Дарья', N'Николаевна'),

('+79803476723', N'Смирнов', N'Михаил', N'Константинович'),

('+79182351726', N'Кузнецова', N'Юлия', N'Михайловна'),

('+79192223343', N'Сергеев', N'Сергей', N'Валентинович'),

('+78003431526', N'Сергеева', N'Валентина', N'Витальевна'),

('+79163542219', N'Иванов', N'Валентин', N'Юрьевич'),

('+74953211234', N'Ткачёва', N'Кристина', N'Владимировна');

GO

INSERT INTO [restaurant](organization\_name, [address], [phone], e\_mail) VALUES

(N'McDonald''s', N'Москва, Манежная площадь, 1, стр. 2', '+78006000770', 'mcduck@mail.ru'),

(N'McDonald''s', N'Москва, Никольская улица, 10', '+78006000770', 'mcduck@mail.ru'),

(N'McDonald''s', N'Москва, Газетный пер., 17', '+78006000770', 'mcduck\_t@mail.ru'),

(N'McDonald''s', N'Москва, Большая Бронная улица, 29', '+78006000770', 'mcduck\_t@mail.ru'),

(N'Вкусно - и точка', N'Москва, Театральный пр., 5, стр. 1, этаж 6', '+78006000770', 'vkustochka@yandex.ru'),

(N'Вкусно - и точка', N'Москва, улица Маросейка, 9', '+78006000770', 'vkustochka@yandex.ru'),

(N'Вкусно - и точка', N'Москва, улица Новый Арбат, 17', '+78006000770', 'vkustochka@yandex.ru'),

(N'Rostic''s', N'Москва, Измайловское шоссе, 71 А', '+79225000880', 'yurest@yandex.ru'),

(N'Rostic''s', N'Москва, шоссе Энтузиастов, 20', '+79225000880', 'yurest@yandex.ru'),

(N'Rostic''s', N'Москва, Верхняя Красносельская улица, 3', '+79225000880', 'yurest\_ro@yandex.ru'),

(N'Шоколадница', N'Москва, Измайловское шоссе, 71 А', '+78002223231', 'marketing@shoko.ru'),

(N'Шоколадница', N'Москва, Большая Семёновская улица, 2', '+78002223231', 'marketing@shoko.ru'),

(N'Крошка Картошка', N'Москва, Семёновская площадь, 1', '+74951390200', 'kartoshka@gmail.com');

GO

INSERT INTO [courier]([phone], second\_name, first\_name, patronymic, salary) VALUES

('+74953223231', N'Рамзанов', N'Алибек', N'Нурсултанович', 40000),

('+79003488327', N'Цзухвиле', N'Азарбан', N'Магометович', 35000),

('+79253332325', N'Шумарев', N'Ахмед', N'Арбазанович', 38000),

('+79183261689', N'Назаров', N'Ахмед', NULL, 42000),

('+79802314767', N'Иванов', N'Василий', N'Дмитриевич', 45000),

('+79342185445', N'Кузнецов', N'Амир', N'Арамазанович', 41000),

('+78003211221', N'Шархимуллин', N'Юлдаш', NULL, 39000),

('+79162361600', N'Смирнова', N'Эльмира', 'Кирилловна', 38500);

GO

INSERT INTO menu\_item(restaurantID, product\_name, cost, ingredients, nutritional\_value, energy, [weight]) VALUES

(1, N'Биг Мак', 200, N'Бифштекс, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 220),

(1, N'Биг Чикен Бургер', 190, N'Куриная грудка, булочка, сыр, помидоры, соус, листья салата',

N'Белки: 20г, Жиры: 35г, Углеводы: 50г', 450, 300),

(1, N'Картофель Фри', 70, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(1, N'МакФлурри', 100, N'Мороженое с клубничным топингом и кусочками ягод',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 180),

(2, N'Биг Мак', 212, N'Бифштекс, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 220),

(2, N'Картофель Фри', 80, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(2, N'МакФлурри', 120, N'Мороженое с клубничным топингом и кусочками ягод',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 180),

(2, N'Пирожок Вишнёвый', 70, N'Тесто, начинка из вишни',

N'Белки: 3г, Жиры: 10г, Углеводы: 30г', 230, 80),

(3, N'Биг Мак', 260, N'Бифштекс, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 220),

(3, N'Биг Чикен Бургер', 180, N'Куриная грудка, булочка, сыр, помидоры, соус, листья салата',

N'Белки: 20г, Жиры: 35г, Углеводы: 50г', 450, 320),

(3, N'Картофель Фри', 65, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(3, N'МакФлурри', 115, N'Мороженое с клубничным топингом и кусочками ягод',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 180),

(3, N'Молочный коктейль', 120, N'Молоко, Клубничный сироп',

N'Белки: 7г, Жиры: 5г, Углеводы: 45г', 180, 270),

(4, N'Биг Мак', 212, N'Бифштекс, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 220),

(4, N'Картофель Фри', 80, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(4, N'МакФлурри', 120, N'Мороженое с клубничным топингом и кусочками ягод',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 180),

(4, N'Пирожок Вишнёвый', 70, N'Тесто, начинка из вишни',

N'Белки: 3г, Жиры: 10г, Углеводы: 30г', 230, 80),

(5, N'Биг Хит', 210, N'Куриная грудка, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 250),

(5, N'Биг Чикен Бургер', 180, N'Куриная грудка, булочка, сыр, помидоры, соус, листья салата, маринованные огурцы',

N'Белки: 20г, Жиры: 35г, Углеводы: 50г', 450, 320),

(5, N'Картофель Фри', 50, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(5, N'Мороженое', 120, N'Мороженое с клубничным топингом и кусочками фруктов',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 170),

(6, N'Биг Хит', 215, N'Куриная грудка, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 250),

(6, N'Картофель Фри', 80, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(6, N'Мороженое', 120, N'Мороженое с клубничным топингом и кусочками фруктов',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 170),

(7, N'Биг Хит', 215, N'Куриная грудка, булочка, сыр, соус, горчица, листья салата, маринованные огурцы',

N'Белки: 8г, Жиры: 20г, Углеводы: 40г', 500, 250),

(7, N'Картофель Фри', 80, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(7, N'Мороженое', 120, N'Мороженое с клубничным топингом и кусочками фруктов',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 170),

(7, N'Чай чёрный', 100, N'Чай чёрный Greenfield', N'Белки: 0г, Жиры: 0г, Углеводы: 15г', 50, 250),

(8, N'Чизбургер', 100, N'Стрипсы из куриного филе, булочка, сыр, кетчуп, маринованные огурцы',

N'Белки: 15г, Жиры: 10г, Углеводы: 25г', 230, 150),

(8, N'Острые крылышки', 350, N'Сочная курица, мука пшеничная, растительное масло, пищевая добавка',

N'Белки: 25г, Жиры: 21г, Углеводы: 10г', 300, 200),

(8, N'Наггетсы', 260, N'Наггетсы оригинальные, растительное масло', N'Белки: 12г, Жиры: 15г, Углеводы: 15г', 230, 100),

(8, N'Картофель Фри', 50, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(9, N'Чизбургер', 110, N'Стрипсы из куриного филе, булочка, сыр, кетчуп, маринованные огурцы',

N'Белки: 15г, Жиры: 10г, Углеводы: 25г', 230, 150),

(9, N'Острые крылышки', 340, N'Сочная курица, мука пшеничная, растительное масло, пищевая добавка',

N'Белки: 25г, Жиры: 21г, Углеводы: 10г', 300, 200),

(9, N'Картофель Фри', 60, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(9, N'Мороженое', 115, N'Мороженое с клубничным топингом и кусочками фруктов',

N'Белки: 10г, Жиры: 5г, Углеводы: 40г', 250, 170),

(10, N'Острые крылышки', 355, N'Сочная курица, мука пшеничная, растительное масло, пищевая добавка',

N'Белки: 25г, Жиры: 21г, Углеводы: 10г', 300, 200),

(10, N'Наггетсы', 265, N'Наггетсы оригинальные, растительное масло', N'Белки: 12г, Жиры: 15г, Углеводы: 15г', 230, 100),

(10, N'Картофель Фри', 70, N'Отборный картофель, соль, растительное масло',

N'Белки: 5г, Жиры: 10г, Углеводы: 40г', 200, 100),

(11, N'Борщ с говядиной', 200, N'Говядина отварная, свёкла, капуста, картофель, морковь, лук, сметана, укроп',

N'Белки: 5г, Жиры: 2г, Углеводы: 2г', 80, 250),

(11, N'Запеканка творожная с изюмом', 100, N'Творог, сахар, яйцо, изюм', N'Белки: 8г, Жиры: 2г, Углеводы: 5г', 100, 300),

(11, N'Горячий шоколад', 120, N'Какао-порошок, сахар, молоко', N'Белки: 6г, Жиры: 5г, Углеводы: 25г', 240, 250),

(11, N'Блинчик с шоколадом', 130, N'Мука, соль, сахар, яйцо, молоко, сливочное масло, шоколад',

N'Белки: 4г, Жиры: 10г, Углеводы: 23г', 195, 250),

(12, N'Борщ с говядиной', 210, N'Говядина отварная, свёкла, капуста, картофель, морковь, лук, сметана, укроп',

N'Белки: 5г, Жиры: 2г, Углеводы: 2г', 80, 250),

(12, N'Горячий шоколад', 110, N'Какао-порошок, сахар, молоко', N'Белки: 6г, Жиры: 5г, Углеводы: 25г', 240, 250),

(12, N'Блинчик с шоколадом', 135, N'Мука, соль, сахар, яйцо, молоко, сливочное масло, шоколад',

N'Белки: 4г, Жиры: 10г, Углеводы: 23г', 195, 250),

(13, N'Картошка с укропом и растительным маслом', 150, N'Картофель отварной, укроп, растительное масло',

N'Белки: 3г, Жиры: 8г, Углеводы: 10г', 160, 260),

(13, N'Суп лапша куриная', 130, N'Лапша отварная, растительное масло, соль, морковь, картофель, лук, петрушка, соль, курица',

N'Белки: 1г, Жиры: 2г, Углеводы: 3г', 28, 270),

(13, N'Пирожное картошка', 120, N'Мука пшеничная, сахар, какао-порошок, соль, масло сливочное, натуральные добавки',

N'Белки: 7г, Жиры: 10г, Углеводы: 45г', 300, 80);

GO

CREATE SEQUENCE seq1

START WITH 1 INCREMENT BY 1;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, Измайловское шоссе, 70', 800, 400, '2022-06-01T12:54:00', '2022-06-01T13:15:00', '+79163654400', '+79003488327');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(30, @id, 700, 2),

(32, @id, 100, 2);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, Большая Семёновская улица, 15', 345, 400, '2022-07-15T13:38:00', '2022-07-15T13:50:00', '+78003431526', '+74953223231');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(44, @id, 210, 1),

(46, @id, 135, 1);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, Верхняя Красносельская улица, 19', 680, 350, '2022-07-28T15:40:00', '2022-07-28T16:03:00', '+79803476723', '+78003211221');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(5, @id, 410, 2),

(6, @id, 150, 2),

(7, @id, 120, 1);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, Щёлковское шоссе, 23к1', 480, 350, '2022-08-01T18:15:00', '2022-08-01T18:32:00', '+79175552211', '+79802314767');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(2, @id, 380, 2),

(4, @id, 100, 1);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, Бауманская улица, 18', 420, 380, '2022-08-03T19:25:00', '2022-08-03T19:53:00', '+74953211234', '+79253332325');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(47, @id, 300, 2),

(49, @id, 120, 1);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, улица Земляной Вал, 32к1', 410, 350, '2022-08-05T10:29:00', '2022-08-05T10:46:00', '+79807432247', '+79183261689');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(18, @id, 210, 1),

(20, @id, 90, 2),

(21, @id, 110, 1);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, 1-я останкинская улица, 15к3', 372, 380, '2022-08-09T11:15:00', '2022-08-09T11:43:00', '+79163654400', '+79802314767');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(5, @id, 212, 1),

(6, @id, 80, 2);

COMMIT;

GO

BEGIN TRANSACTION

DECLARE @id INT;

SET @id = NEXT VALUE FOR seq1;

INSERT INTO [order](orderID, [address], order\_cost, delivery\_cost, order\_datetime, delivery\_datetime, client\_phone, courier\_phone) VALUES

(@id, N'Москва, проспект Мира, 3', 300, 380, '2022-08-12T16:17:00', '2022-08-12T16:31:00', '+79803476723', '+74953223231');

INSERT INTO order\_item(MenuItemID, orderID, cost, [count]) VALUES

(1, @id, 200, 1),

(4, @id, 100, 1);

COMMIT;

GO

--Вывести телефоны и ФИО курьеров, которые не доставили ни одного заказа

SELECT c.[phone], c.first\_name, c.second\_name, c.patronymic FROM [order] o RIGHT JOIN [courier] c

ON c.[phone] = o.courier\_phone

WHERE o.orderID IS NULL;

GO

--Вывести рестораны, в меню которых есть позиции, не попавшие ни в один заказ, и количество таких позиций

SELECT r.organization\_name, r.[address], COUNT(\*) AS [count] FROM [restaurant] r, menu\_item m LEFT JOIN order\_item o

ON m.MenuItemID = o.MenuItemID

WHERE r.restaurantID = m.restaurantID AND o.MenuItemID IS NULL

GROUP BY r.organization\_name, r.[address];

GO

--Показать данные о заказах вместе с клиентами, сделавшими эти заказы, и курьерами, которые их доставляли

SELECT cl.[phone] AS client\_phone, cl.second\_name, cl.first\_name, o.[address], o.order\_cost, o.delivery\_cost,

co.[phone] AS courier\_phone, co.second\_name, co.first\_name

FROM client cl FULL OUTER JOIN [order] o

ON cl.[phone] = o.client\_phone FULL OUTER JOIN courier co

ON co.[phone] = o.courier\_phone;

GO

--Вывести среднюю стоимость заказ+доставка по каждому курьеру, отсортировав по убыванию

SELECT c.[phone], AVG(o.order\_cost + o.delivery\_cost) AS mean\_cost FROM [courier] c INNER JOIN [order] o

ON c.[phone] = o.courier\_phone

GROUP BY c.[phone]

ORDER BY mean\_cost DESC;

GO

--Вывести данные курьеров с зарплатой между 38000 и 42000, отсортировав по зарплате по возрастанию

SELECT \* FROM [courier]

WHERE salary BETWEEN 38000 AND 42000

ORDER BY salary ASC;

GO

--Вывести цену самого дешёвого товара в каждой из торговых точек, отсортировав по возрастанию

SELECT r.organization\_name, r.[address], MIN(m.cost) AS min\_cost FROM [restaurant] r INNER JOIN menu\_item m

ON r.restaurantID = m.restaurantID

GROUP BY r.organization\_name, r.[address]

ORDER BY min\_cost ASC;

GO

--Вывести суммарное количество калорий в каждом заказе, выбрав те заказы, для которых это значение больше 1000

SELECT o.orderID, o.client\_phone, c.second\_name, c.first\_name, SUM(m.energy\*i.[count]) AS sum\_energy

FROM client c, [order] o, order\_item i, menu\_item m

WHERE c.[phone] = o.client\_phone AND o.orderID = i.orderID AND i.MenuItemID = m.MenuItemID

GROUP BY o.orderID, o.client\_phone, c.second\_name, c.first\_name

HAVING SUM(m.energy\*i.[count]) > 1000;

GO

--Вывести самую высокую зарплату курьера

SELECT MAX(salary) AS max\_salary FROM courier;

GO

--Вывести данные курьеров, которые доставляли заказы быстрее 20 минут

SELECT DISTINCT \* FROM [courier]

WHERE [phone] IN

(SELECT courier\_phone FROM [order]

WHERE DATEDIFF(MINUTE, order\_datetime, delivery\_datetime) < 20);

GO

--Вывести данные клиентов, которые совершили хотя бы один заказ

SELECT \* FROM [client] c

WHERE EXISTS(SELECT \* FROM [order] o WHERE c.[phone] = o.client\_phone);

--Вывести данные курьеров, у которых фамилия начинается на "Ш" или имя начинается на "Э"

SELECT \* FROM [courier]

WHERE second\_name LIKE N'Ш%'

UNION

SELECT \* FROM [courier]

WHERE first\_name LIKE N'Э%';

GO

--Вывести данные клиентов с фамилией "Иванов(-а)", имя которых начинается на "В"

SELECT \* FROM [client]

WHERE second\_name LIKE N'Иванов%'

INTERSECT

SELECT \* FROM [client]

WHERE first\_name LIKE N'В%';

GO

--Показать пункты меню ресторанов, которые дешевле 120 рублей и калорийность которых не выше средней

SELECT \* FROM menu\_item

WHERE cost < 120

EXCEPT

SELECT \* FROM menu\_item

WHERE energy > (SELECT AVG(energy) FROM menu\_item);

GO

--Вывести все рестораны, которые называются "McDonald's", и те, у которых рабочая почта в Яндексе

SELECT \* FROM [restaurant]

WHERE organization\_name = N'McDonald''s'

UNION ALL

SELECT \* FROM [restaurant]

WHERE e\_mail LIKE '%@yandex.ru';

GO