**ПРАВИТЕЛЬСТВО САНКТ-ПЕТЕРБУРГА**

**КОМИТЕТ ПО НАУКЕ И ВЫСШЕЙ ШКОЛЕ**

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Санкт-Петербургский технический колледж управления и коммерции»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование курса:** | | | | *МДК 01.01 Разработка программных модулей* | | | | | | | |
| **Тема:** | *«Разработка компьютерной игры* | | | | | | | | | |
|  | *для ООО «Моя Организация»»* | | | | | | | | | |
|  |  | | | | | | | | | |
| **Специальность:** | | *09.02.07 Информационные системы* | | | | | | | | |
|  | | *и программирование* | | | | | | | | |
|  | | | (шифр и наименование специальности) | | | | | | | |
|  | | |  | | | | | | | |
| **Студент** | *группы 9ПО-41* | | | |
|  |  | | *Д. Д. Проскурня* | |
|  | ***(подпись)*** | | | ***(И.О. Фамилия)*** |
| **Оценка** |  | | | |
|  | *(проставляется цифрами и в скобках прописью)* | | | |
| **Преподаватель** |  | *А. В. Смирнова* | | |
|  | ***(подпись)*** | ***(И.О. Фамилия)*** | | |

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc102579488)

[1.ПОСТАНОВКА ЗАДАЧИ 4](#_Toc102579489)

[2. СРЕДСТВА РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ 6](#_Toc102579490)

[3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ 7](#_Toc102579491)

[3.1 Инфологическое проектирование базы данных 8](#_Toc102579492)

[3.1.1 Определение сущностей 9](#_Toc102579493)

[3.2 Логическое (даталогическое) проектирование 10](#_Toc102579494)

[3.2.1 Преобразование ER-диаграммы в схему базы данных 11](#_Toc102579495)

[3.3 Физическое проектирование 12](#_Toc102579496)

[3.4. Создание таблиц базы данных 17](#_Toc102579497)

[3.5. Ввод тестовых данных 21](#_Toc102579498)

[3.6. Создание просмотров 24](#_Toc102579499)

[3.7. Создание триггеров 27](#_Toc102579500)

[3.8. Создание хранимых процедур 31](#_Toc102579501)

[4. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 33](#_Toc102579502)

[4.1. Разработка рабочего модуля ИС 33](#_Toc102579503)

[4.2. Демонстрация работы приложения 34](#_Toc102579504)

[ЗАКЛЮЧЕНИЕ 38](#_Toc102579505)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 39](#_Toc102579506)

## ВВЕДЕНИЕ

Игры – новый виток эволюции общества. В последние десятилетия компьютерные игры стали неотъемлемой частью нашей жизни. Миллионы людей по всему миру проводят часы за экранами, погружаясь в виртуальные миры. И хотя некоторые все еще считают игры просто развлечением для детей, все чаще и чаще появляются исследования, которые подтверждают их положительное влияние на наше психическое и эмоциональное благополучие.

В данном курсовой проекте рассматривается разработка компьютерной видеоигры “HYDROGENIUM” жанра платформер.

Одно из основных преимуществ компьютерных игр – возможность испытывать новые роли и переживания виртуальных миров. Они позволяют нам занять позицию героя, принимать сложные решения, сражаться с врагами и решать головоломки. Игры предоставляют нам безграничные возможности и полноту контроля над событиями, что может быть особенно привлекательно в реальности, где мы часто сталкиваемся с ограничениями и трудностями.

При работе барбершопа непременно возникает необходимость вести учет обращений клиентов, их предварительной записи, учета оказанных услуг и проданных сопутствующих товаров.

Целью курсового проектирования является прохождение уровня, собрав как можно больше очков к концу уровня для ООО «Моя Организация». Проект представляет собой компьютерную игру.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Создание концепта игры;
2. Разработать компьютерную игру.

## 1.ПОСТАНОВКА ЗАДАЧИ

В рамках данного курсового проекта необходимо разработать компьютерную видеоигру на языке *C#,* используя среду разработки Unity, и Microsoft Visual Studio. Игра должна обладать следующими функциональными возможностями:

* Авторизация пользователей;
* Добавление, просмотр, редактирование информации о клиентах барбершопа;
* Добавление, просмотр, редактирование информации о заказах на парикмахерские услуги и покупку сопутствующих товаров;
* Просмотр информации о сотрудниках барбершопа.
* Просмотр информации о предстоящих заказах.

Окна просмотра информации должны содержать:

* Информацию в виде таблицы с понятными пользователю заголовками;
* Возможность поиска информации по нескольким наиболее важным полям, в зависимости от назначения формы;
* Возможность сортировки записей по одному из нескольких полей, выбор которого осуществляется с помощью раскрывающегося списка.

Окна ввода и редактирования данных должны содержать:

* Поля для ввода данных;
* Пояснения к каждому из полей, раскрывающие их назначение;
* Кнопку «Сохранить».

На рисунке 1 представлена диаграмма вариантов использования (use-case диаграмма) для разрабатываемой подсистемы.

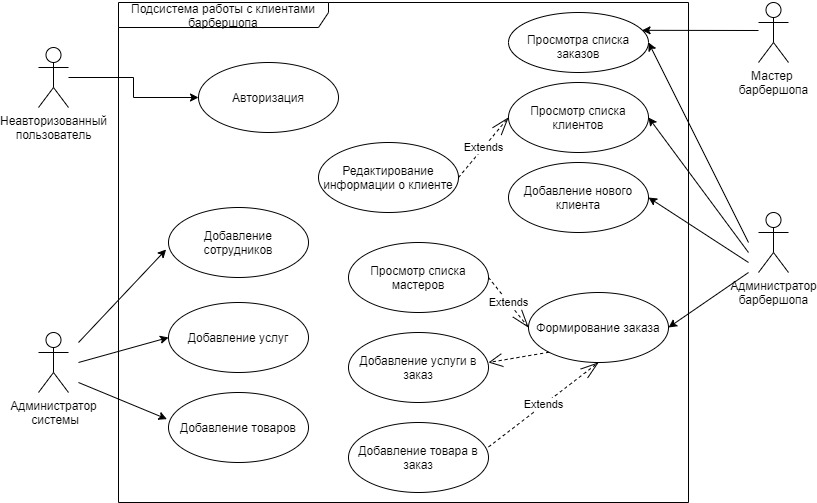


Рисунок 1 - Диаграмма вариантов использования

## 2. СРЕДСТВА РАЗРАБОТКИ КОМПЬЮТЕРНОЙ ИГРЫ

Правильно спроектированная игра должна удовлетворять следующим требованиям:

1. Минимальная избыточность;
2. Непротиворечивость;
3. Целостность данных;
4. Независимость данных;
5. Возможность ведения (добавления и удаления) и актуализации (корректировки, модификации) данных;
6. Безопасность и секретность;
7. Высокая производительность.

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой MS SQL Server. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. С помощью SSMS можно развертывать, отслеживать и обновлять компоненты уровня данных, используемые вашими приложениями, а также создавать запросы и скрипты.

Создание многопользовательской БД и получение одновременного доступа нескольких пользователей к общей базе данных возможно как в рамках модели сервера баз данных (предпочтительно), так и модели удаленного доступа к данным

## 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ

Проектирование БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. В общем случае можно выделить следующие этапы проектирования;

1. Системный анализ и словесное описание информационных объектов предметной области.

2. Проектирование инфологической модели предметной области - частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ЕR-модели.

3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели данных.

4. Физическое проектирование БД, то есть выбор эффективного размещения БД па внешних носителях для обеспечения наиболее эффективной работы системы.

Оценка проекта БД:

1. сокращение избыточности данных,
2. уменьшение затрат на многократное обновление полей,
3. устранение возможных противоречий информации из-за хранения в различных местах и получение «чистого» проекта.

Этапы проектирования БД:

1. Концептуальное проектирование - сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия:

* обследование предметной области, изучение ее информационной структуры
* выявление всех фрагментов, каждый из которых характеризуется пользовательским представлением, информационными объектами и связями между ними, процессами над информационными объектами
* моделирование и интеграция всех представлений

По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде модели "сущность-связь".

2. Логическое проектирование - преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

3. Физическое проектирование - определение особенностей хранения данных, методов доступа и т.д.

## 3.1 Инфологическое проектирование базы данных

Инфологический аспект имеется в виду при рассмотрении вопросов, связанных со смысловым содержанием данных, с их семантикой, независимо от способа представления в памяти компьютерной системы.

Концептуальное (инфологическое) проектирование – построение сематической модели предметной области, то есть информационной модели наиболее высоко уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Термины «сематическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контекст равноправно могут использоваться слова «модель базы данных» и «модель предметной области», поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

* Описание информационных объектов или понятий предметной области и связей между ними. Описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.
* Описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

## 3.1.1 Определение сущностей

Этапом инфологического проектирования, является определение того, какая информация о предметной области должна быть представлена в базе данных. Конкретная информационная подсистема разрабатывается для удовлетворения информационных потребностей определенного круга пользователей. Из анализа предметной области можно выделить несколько сущностей:

1. Роль пользователя;
2. Пользователь;
3. Должность;
4. Сотрудник;
5. Клиент;
6. Вид услуги;
7. Услуга;
8. Товар;
9. Заказ;
10. Заказанная услуга;
11. Заказанный сопутствующий товар.

3.1.2. Связи между сущностями

В реляционной БД отношение между объектами (сущностями) определяет связи между таблицами.

Всего существует 3 типа связей:

* Один к одному. Связь один к одному образуется, когда ключевой столбец (идентификатор) присутствует в другой таблице, в которой тоже является ключом либо свойствами столбца задана его уникальность (одно и тоже значение не может повторяться в разных строках).
* Один ко многим. В типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.
* Многие ко многим. Если нескольким записям соответствует несколько записей из другой таблицы, то связь называется «многие ко многим».

В результате концептуального проектирования, исходя из особенностей предметной области и выявленных в ней сущностей была сформирована концептуальная модель данных, представленная на рисунке 2.

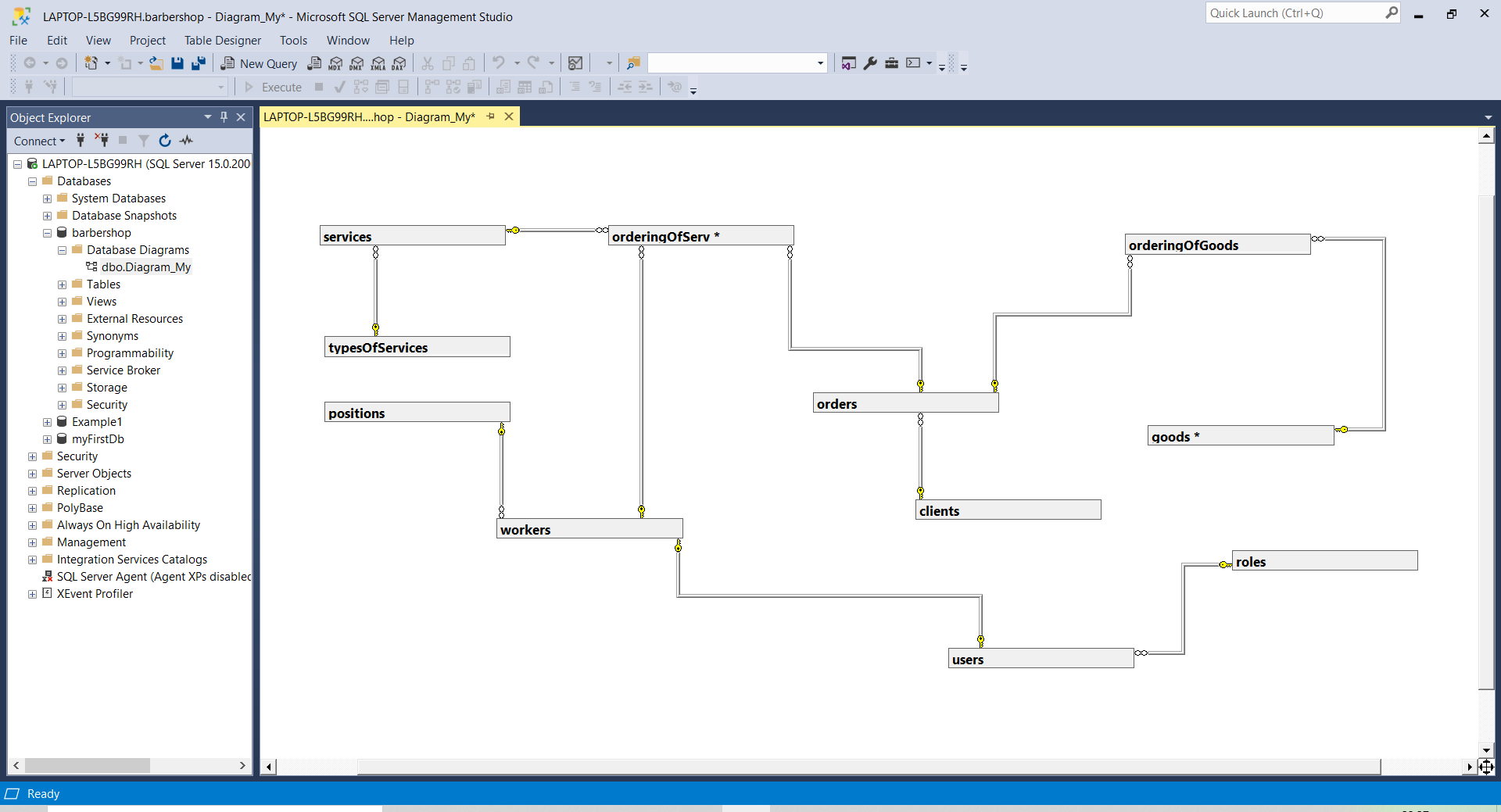


Рисунок 2 – Концептуальная модель данных

## 3.2 Логическое (даталогическое) проектирование

Логическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик (например, длины поля). Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД. В данном проекте будет использоваться реляционная модель данных, так как именно она поддерживается СУБД MS SQL Server.

## 3.2.1 Преобразование ER-диаграммы в схему базы данных

Преобразование ER–диаграммы в схему реляционной БД выполняется путем преобразования сущностей в отношения по определенным правилам. При этом, данные правила учитывают не только степень связи, но степень участия сущностей в формируемой связи. Исходя из этого получаем три правила для связи 1:1, два для связи 1:М, одно для связи М:М. Применяя данные правила последовательно к каждой паре сущностей, мы получим схему БД (ее логическую модель).

Связь типа 1:М (один-ко-многим) между отношениями реализуются через внешний ключ. Ключ вводиться для того отношения, к которому осуществляется множественная связь. Внешнему ключу должен соответствовать первичный или уникальный ключ основного (родительского) отношения.

Применив к диаграмме на рисунке 2 правила преобразования концептуальной модели данных в реляционную, нами была получена логическая модель, которая представлена на рисунке 3. На ней представлены отношения, связи между ними (с указанием степени связи), первичные ключи сформированных отношений.

Вероятно, при изучении логической модели могут возникнуть сомнения в организации связи между Пользователями и Сотрудниками. В данной базе данных выбрано такое решение исходя из того, что в дальнейшем, к системе смогут подключаться и зарегистрированные клиенты (не все клиенты захотят регистрироваться в системе, или вовсе не являются пользователями сети Интернет), что потребует добавление промежуточного отношения Клиенты- Пользователи. На данный момент все сотрудники барбершопа получают доступ к информационной системе, что объясняет их обязательную степень участия в данной связи. Предполагается, что степень участия пользователя- не обязательная, так как пользователи-клиенты не будут участвовать в связи с сотрудником. Исходя из этого, с учетом перспективы масштабирования системы было выбрано именно такое решение, при котором пользователи получают свой суррогатный ID, который указывается у сотрудника.

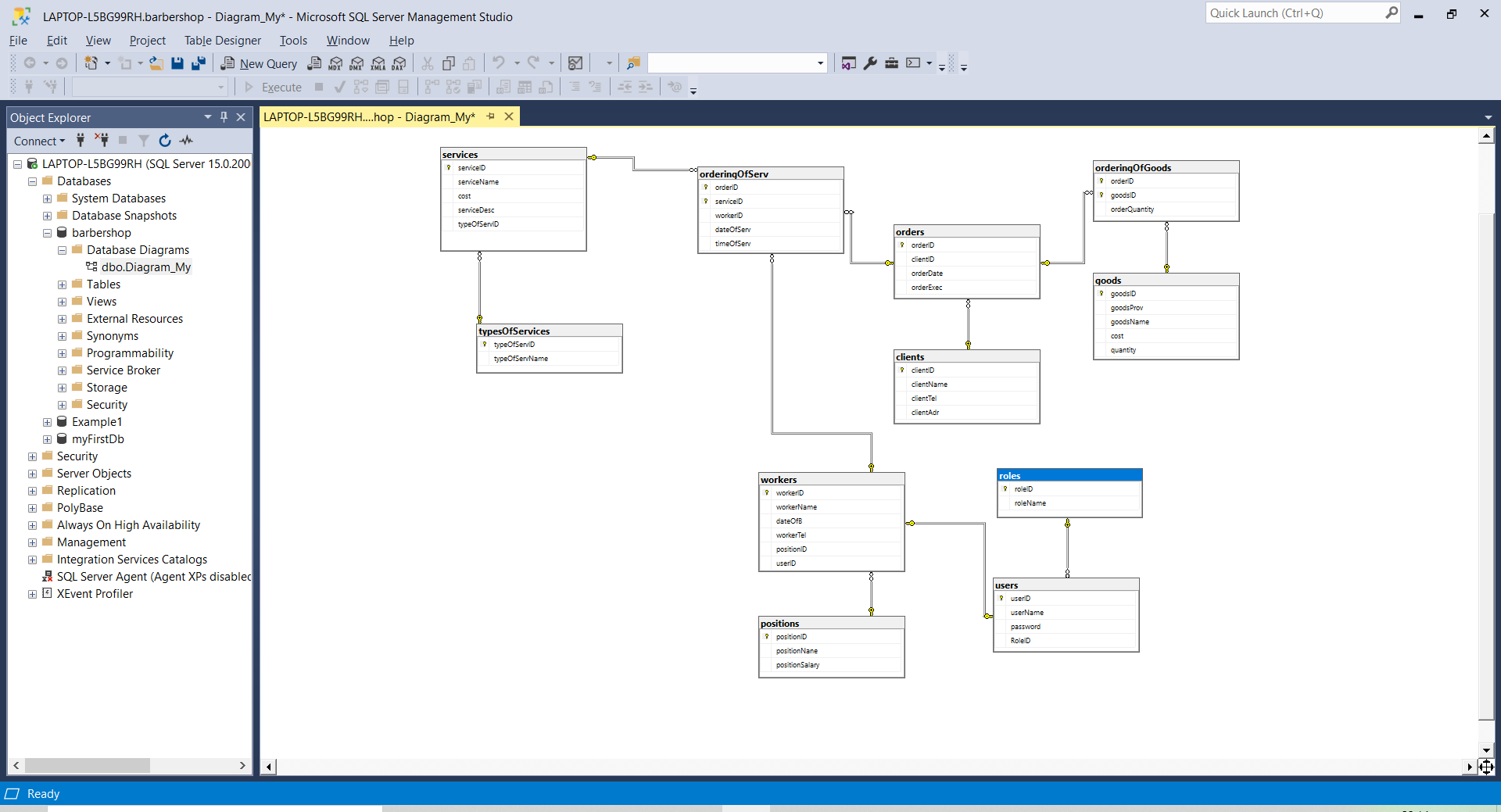


Рисунок 3 – Логическая модель данных

## 3.3 Физическое проектирование

На этапе физического проектирования для каждого отношения выбирается тип данных из тех, что предлагает конкретная СУБД.

Результат физического проектирования представлен на рисунке 4.

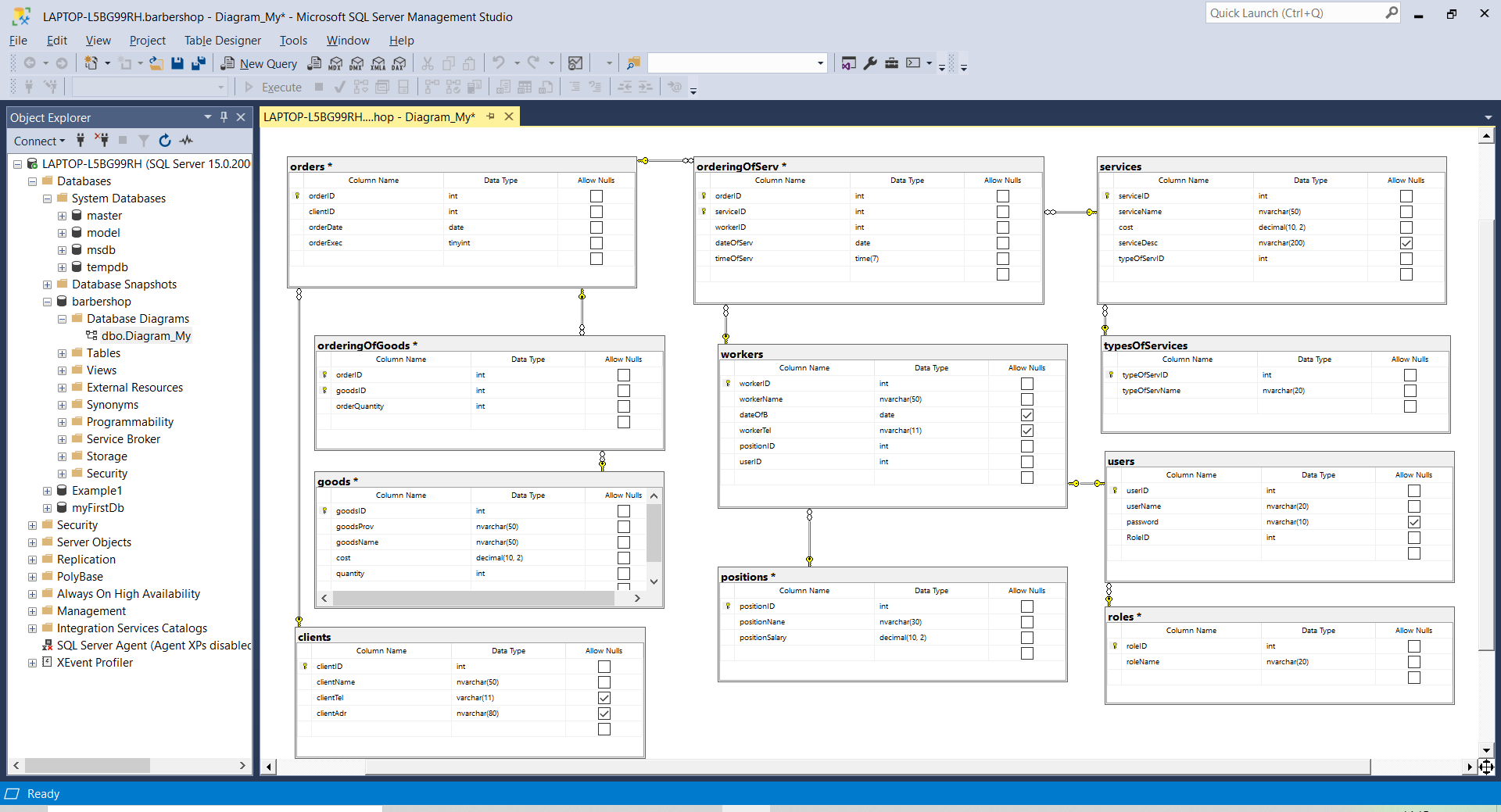


Рисунок 4 – Физическая модель базы данных

Ниже представлена структура каждой из сформированных таблиц с указанием их полей, их типов данных и обязательности заполнения.

Структура таблицы «Роли пользователей» представлена на рисунке 5.

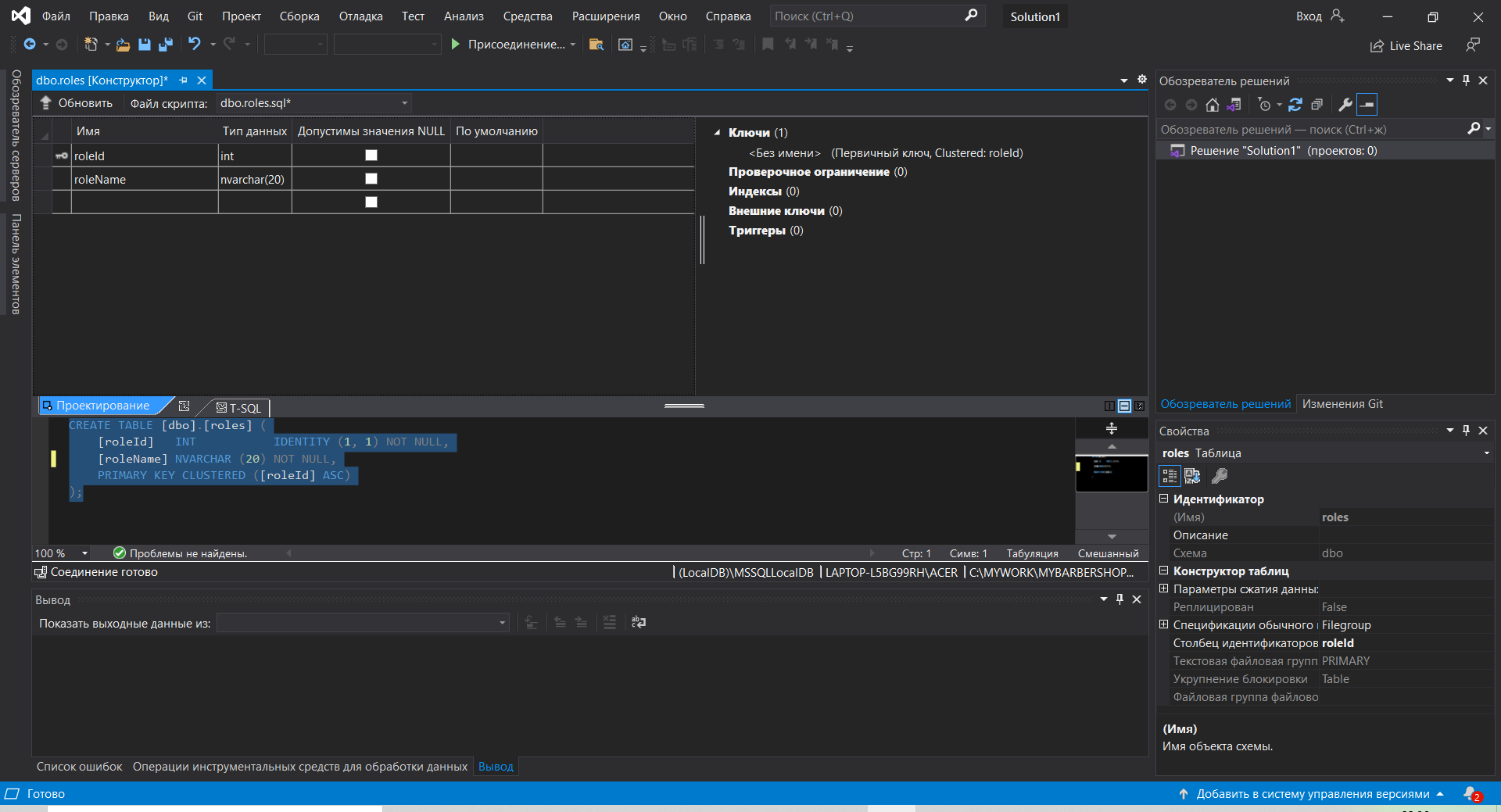


Рисунок 5 – Структура таблицы «Роли пользователей»

Структура таблицы «Пользователи» представлена на рисунке 6.

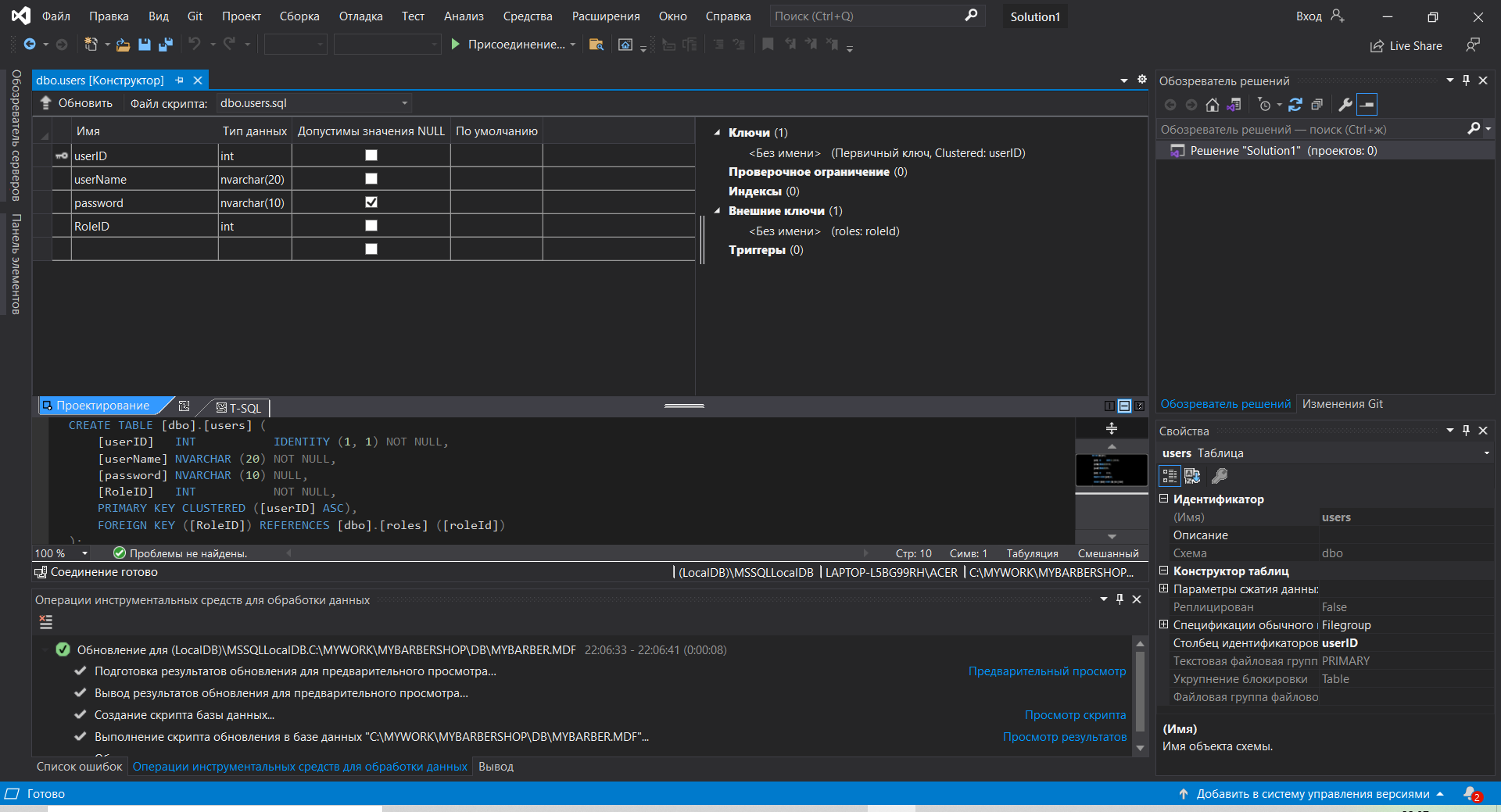


Рисунок 6 – Структура таблицы «Пользователи»

Структура таблицы «Должности» представлена на рисунке 7.

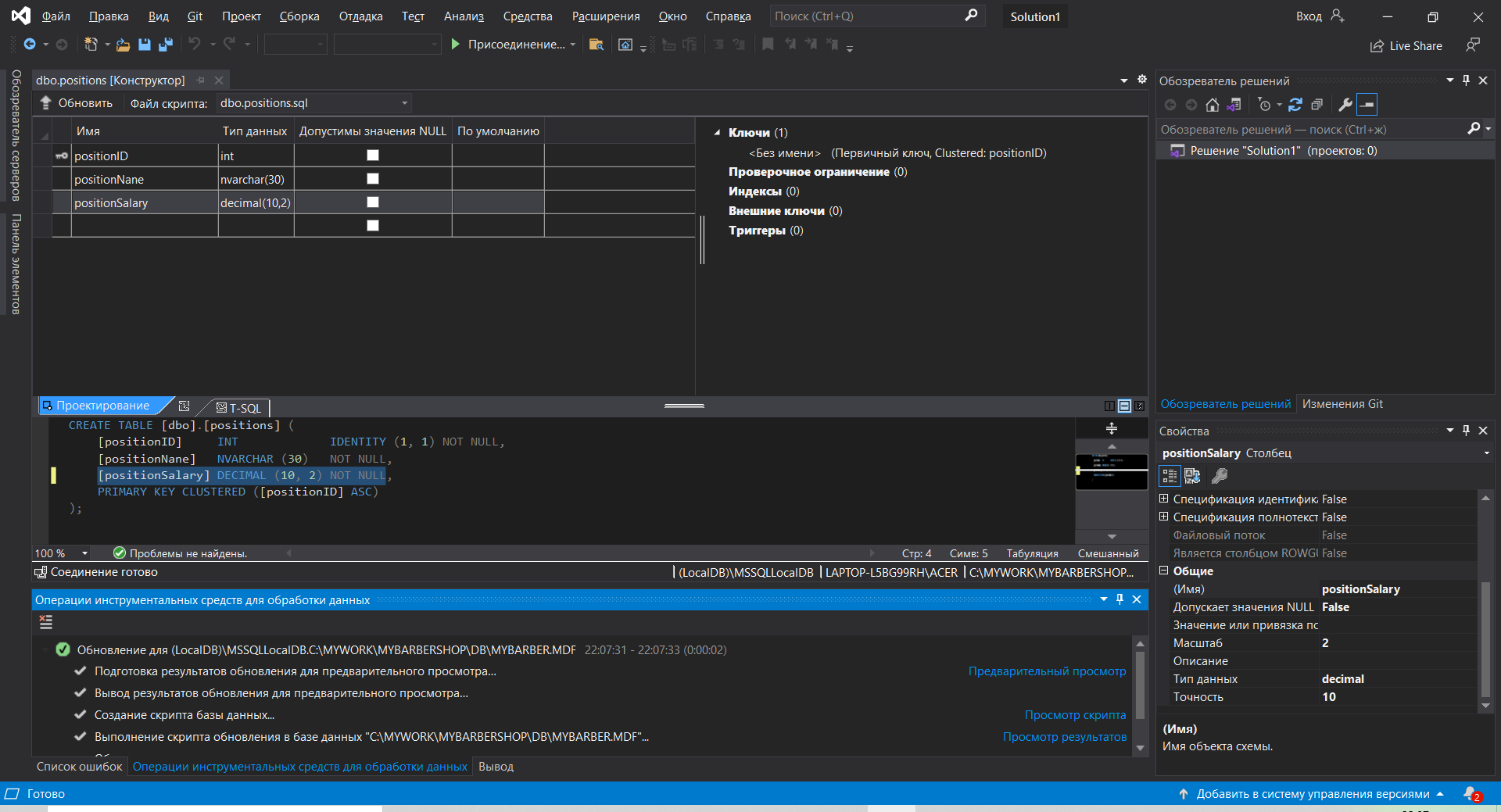


Рисунок 7 – Структура таблицы «Должности»

Структура таблицы «Сотрудники» представлена на рисунке 8.

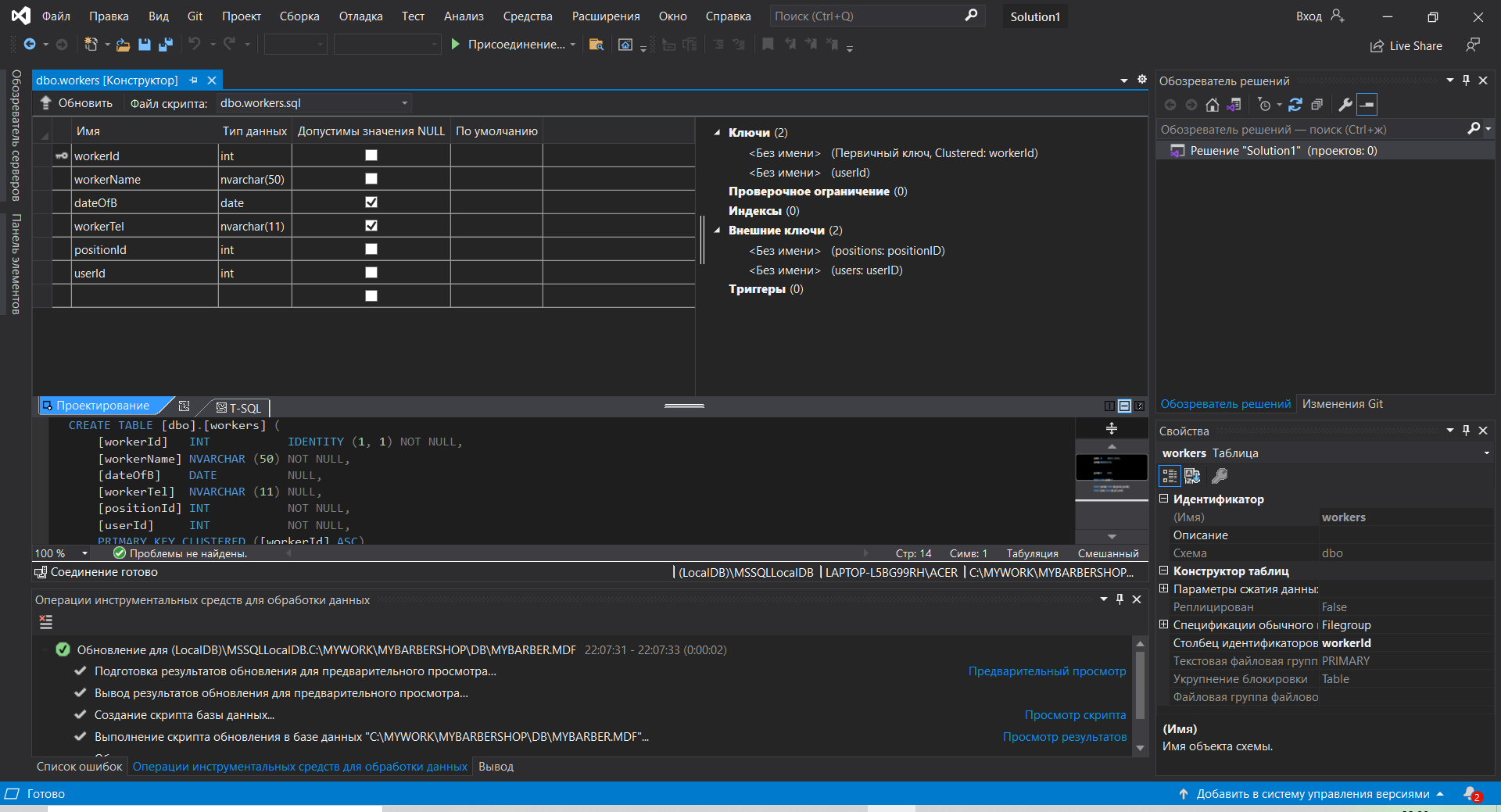


Рисунок 8 – Структура таблицы «Сотрудники»

Структура таблицы «Клиенты» представлена на рисунке 9.



Рисунок 9 – Структура таблицы «Клиенты»

Структура таблицы «Виды услуг» представлена на рисунке 10.

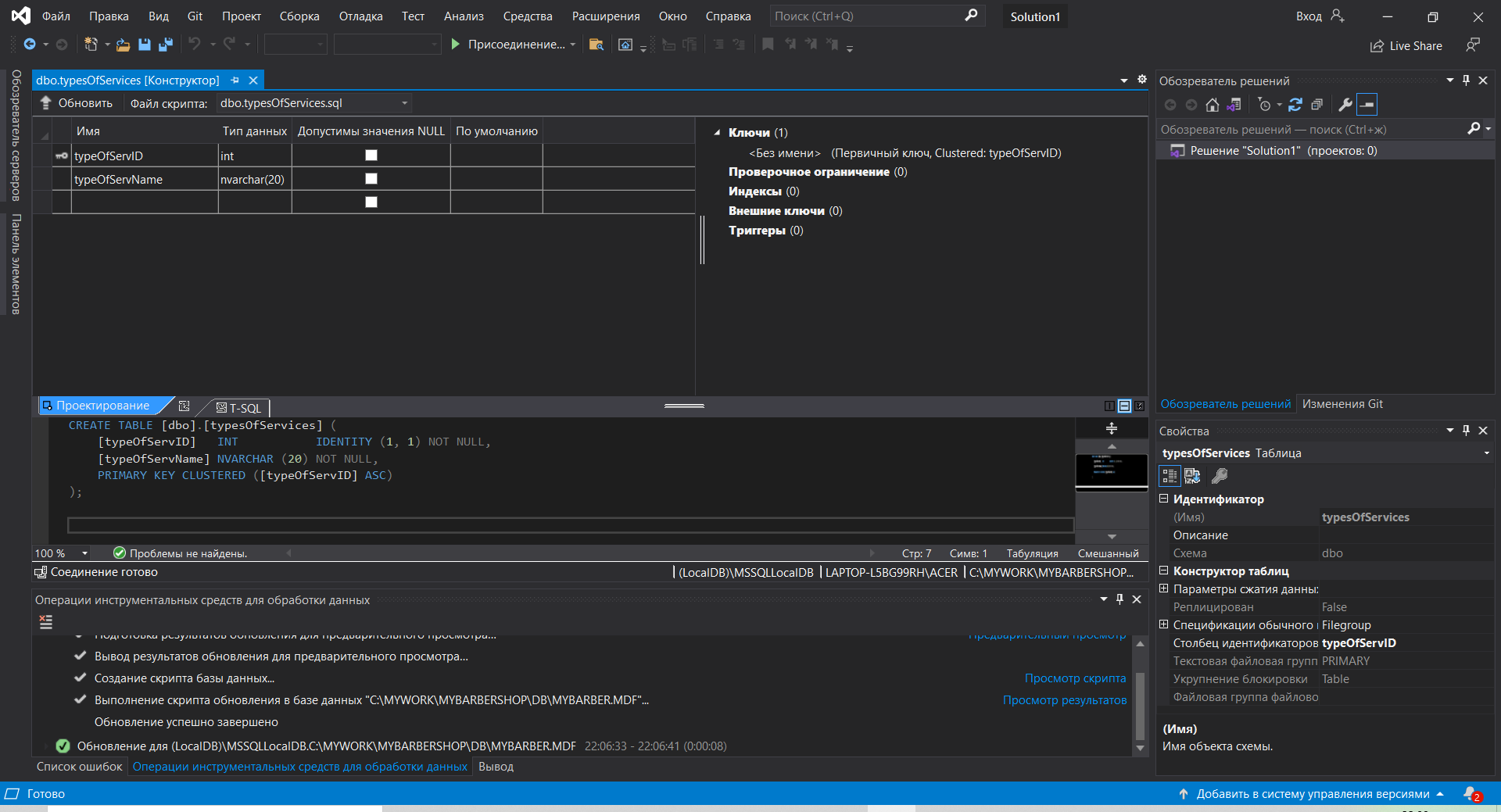


Рисунок 10 – Структура таблицы «Виды услуг»

Структура таблицы «Услуги» представлена на рисунке 11.

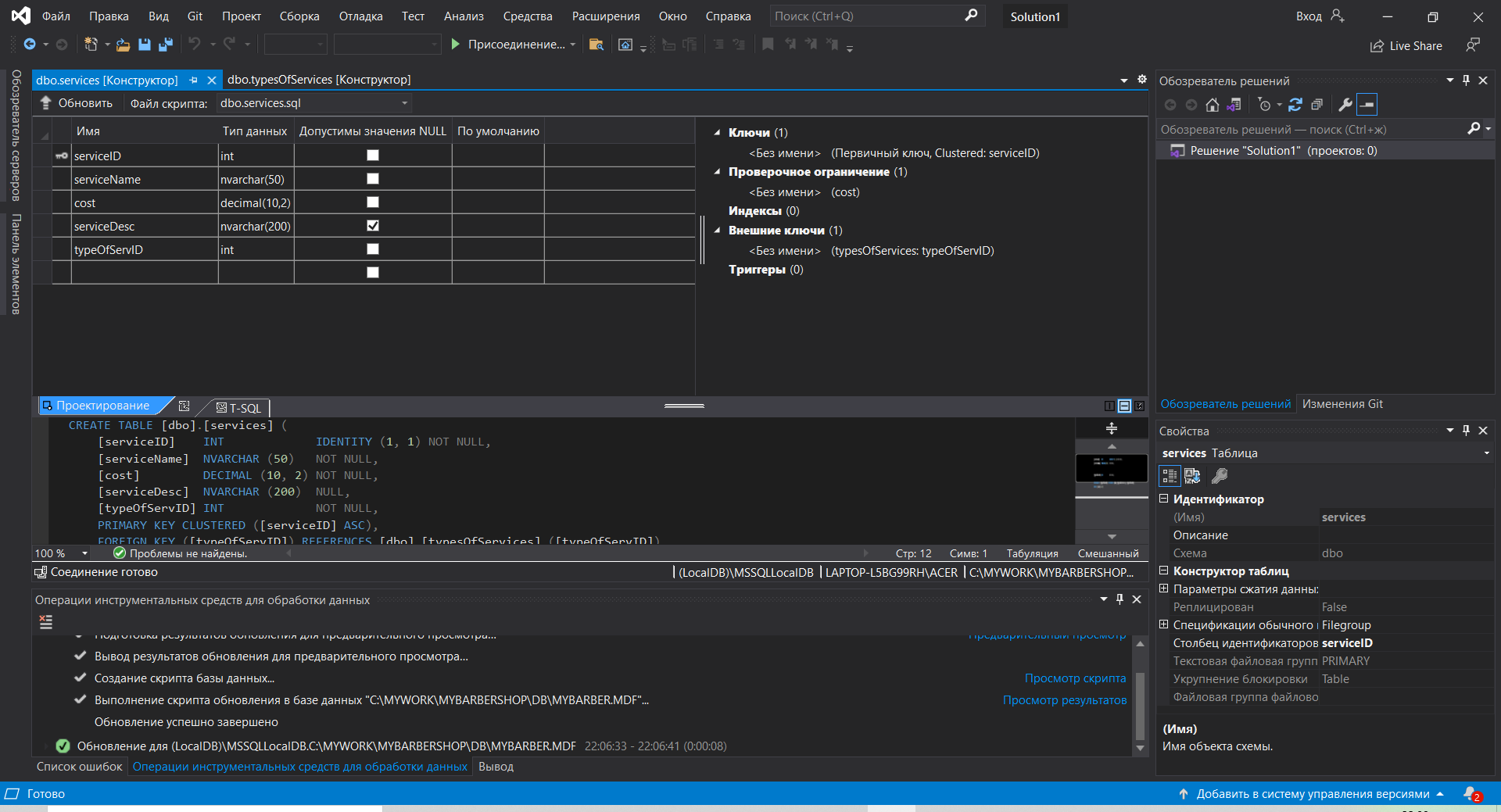


Рисунок 11 – Структура таблицы «Услуги»

Структура таблицы «Товары» представлена на рисунке 12.

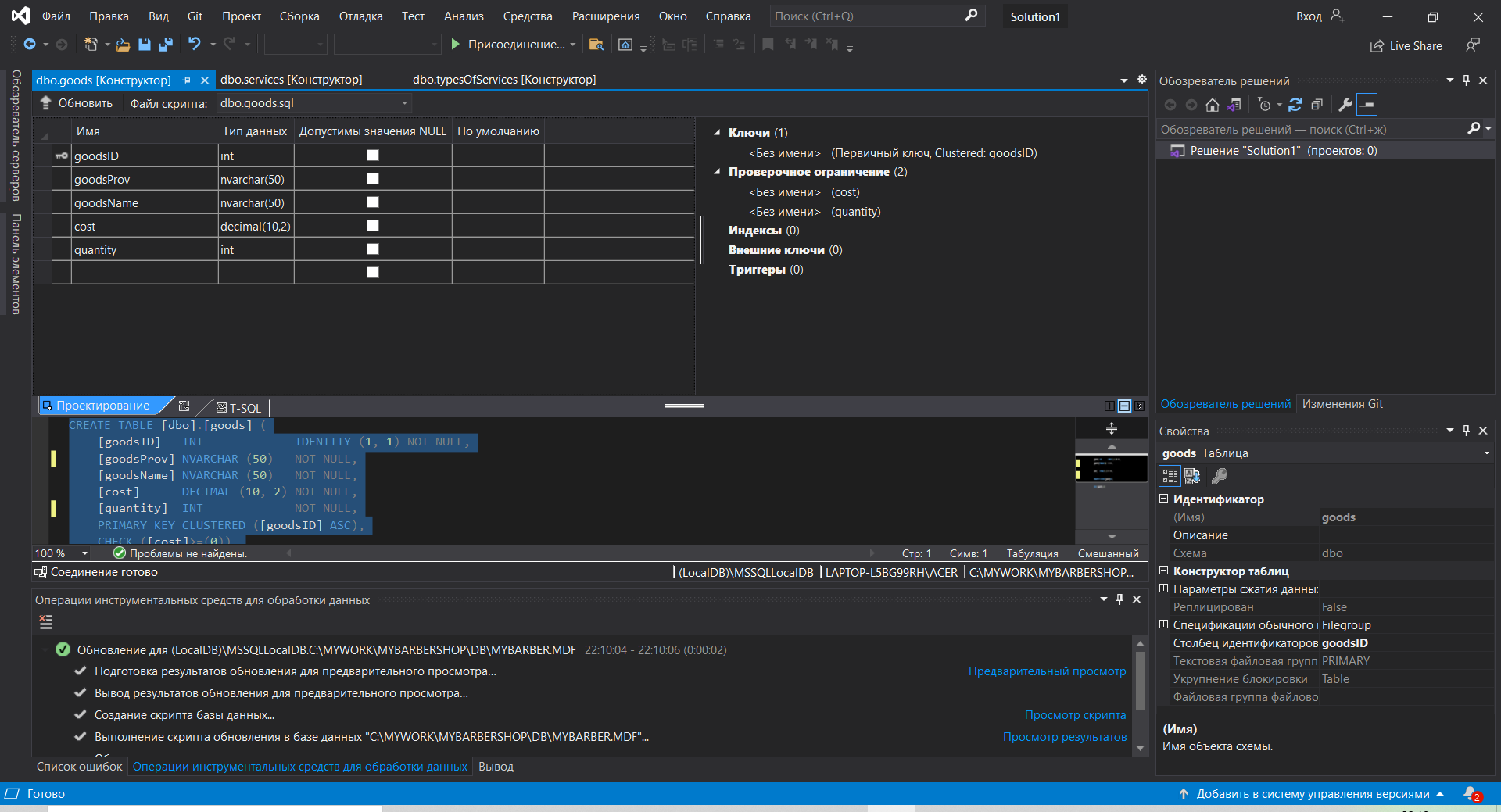


Рисунок 12 – Структура таблицы «Товары»

Структура таблицы «Заказы» представлена на рисунке 13.

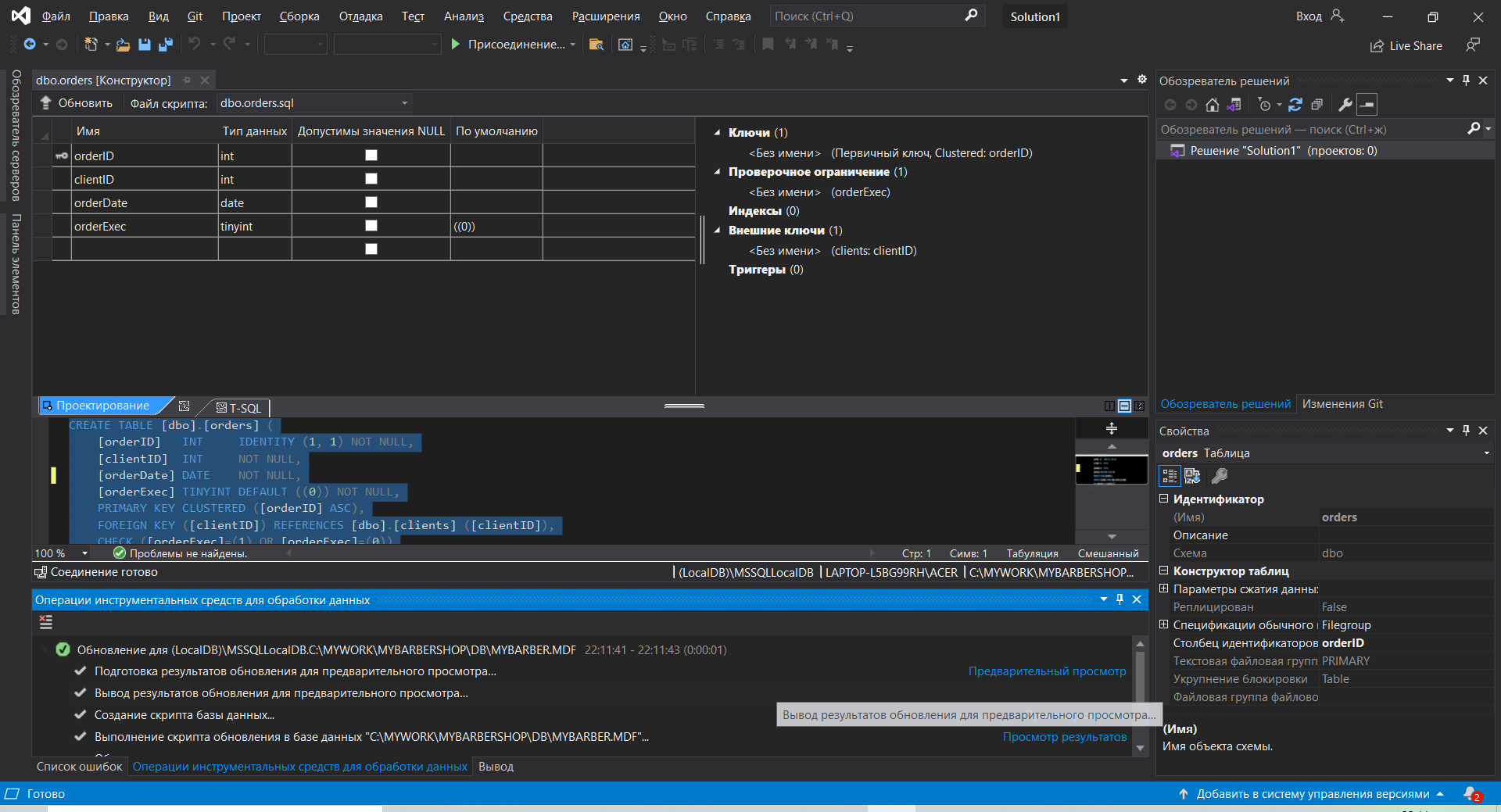


Рисунок 13 – Структура таблицы «Заказы»

Структура таблицы «Заказ услуг» представлена на рисунке 14.

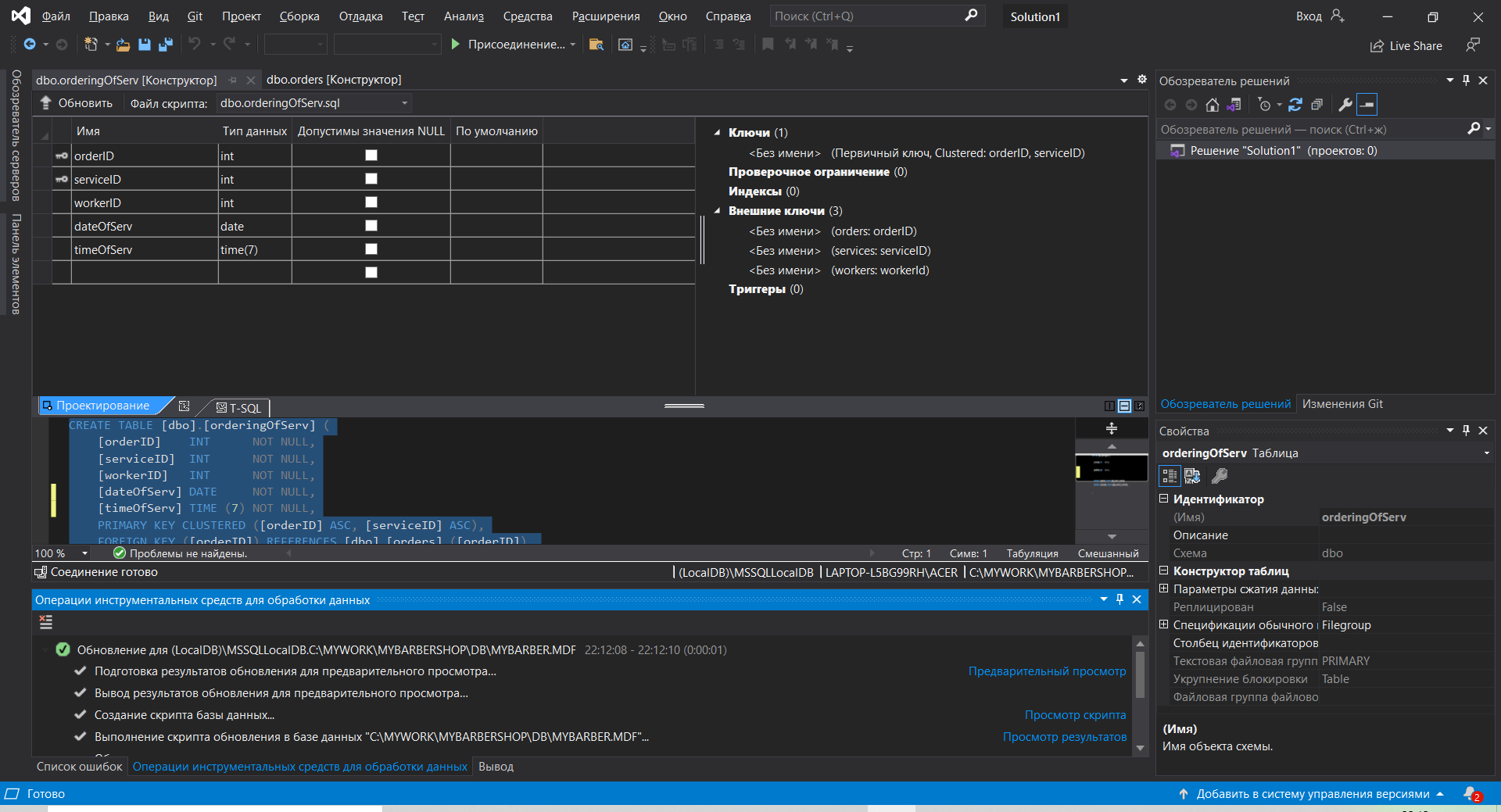


Рисунок 14 – Структура таблицы «Заказ услуг»

Структура таблицы «Заказ товаров» представлена на рисунке 15.

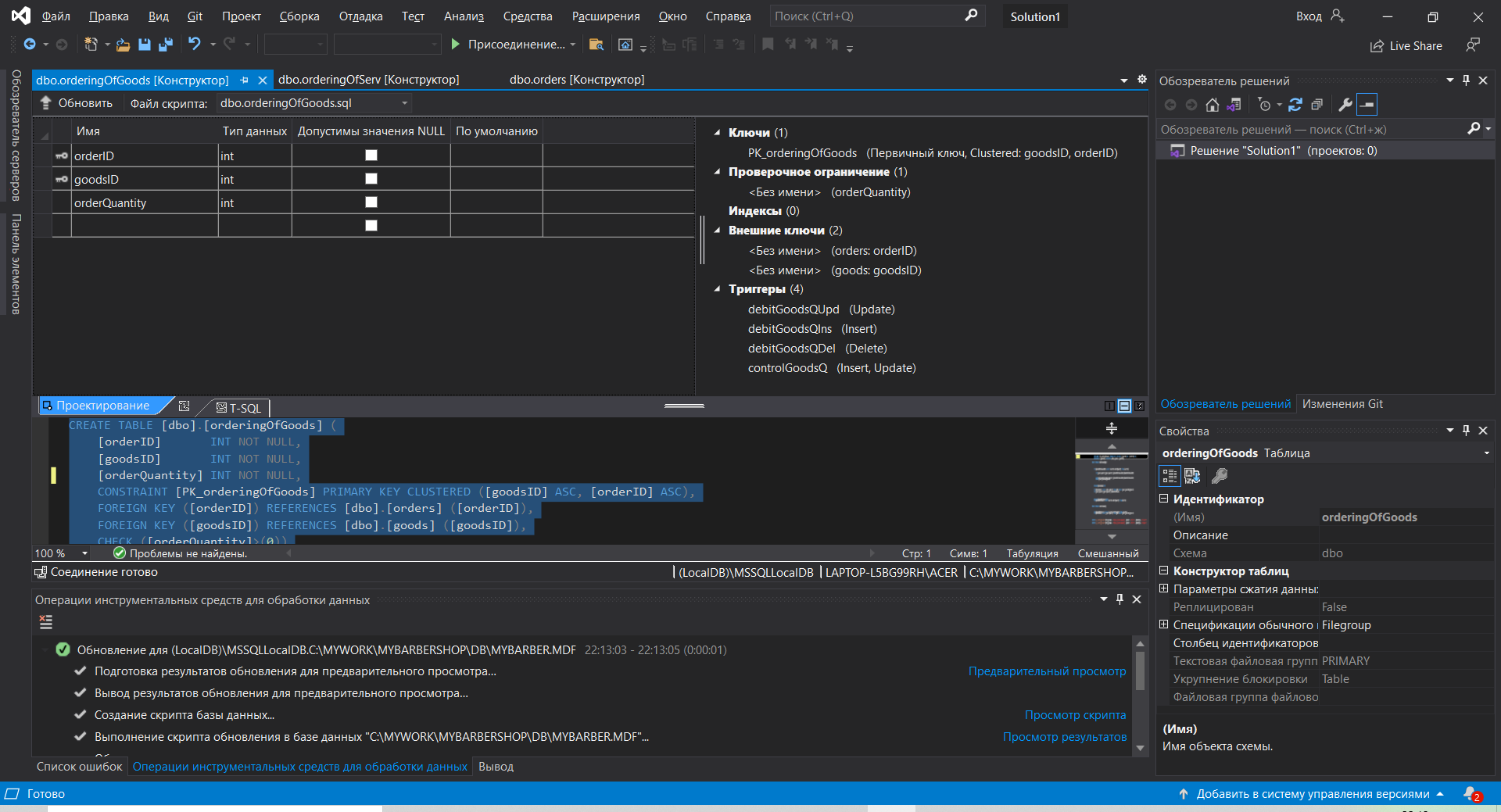


Рисунок 15 – Структура таблицы «Заказ товаров»

## 3.4. Создание таблиц базы данных

Для создания определенных ранее таблиц использовался оператор SQL create table.

Ниже представлен оператор для создания таблицы «Роли пользователей»:

create table roles (

roleID int identity not null primary key,

roleName nvarchar(20) not null

);

Ниже представлен оператор для создания таблицы «Пользователи»:

create table users (

userID int identity not null primary key,

userName nvarchar(20) not null,

password nvarchar(10),

roleID int not null,

foreign key(roleID) references roles(roleID));

Ниже представлен оператор для создания таблицы «Должности»:

create table positions (

positionID int identity not null primary key,

positionNane nvarchar(30) not null,

positionSalary decimal(10, 2) not null

);

Ниже представлен оператор для создания таблицы «Сотрудники»:

create table workers (

workerID int identity not null primary key,

workerName nvarchar(50) not null,

dateofb date,

workerTel nvarchar (11),

positionID int not null,

userID int not null,

unique(userID),

foreign key(positionID) references positions (positionID),

foreign key(userID) references users(userID)

);

Ниже представлен оператор для создания таблицы «Товары»:

create table goods(

goodsID int identity not null primary key,

goodsProv nvarchar(50) not null,

goodsName nvarchar(50) not null,

cost decimal(10,2) not null check(cost>=0),

quantity int not null check(quantity>=0)

);

Ниже представлен оператор для создания таблицы «Клиенты»:

create table clients(

clientID int identity not null primary key,

clientName nvarchar(50) not null,

clientTel varchar(11),

clientAdr nvarchar(80)

);

Ниже представлен оператор для создания таблицы «Виды услуг»:

create table typesOfServices(

typeOfServID int identity not null primary key,

typeOfServName nvarchar(20) not null

);

Ниже представлен оператор для создания таблицы «Заказы»:

create table orders(

orderID int identity not null primary key,

clientID int not null,

orderDate date not null,

orderExec tinyint not null default 0

check(orderExec in (0,1)),

foreign key(clientID) references clients(clientID)

);

Ниже представлен оператор для создания таблицы «Заказ товаров»:

create table orderingOfGoods(

orderID int not null,

goodsID int not null,

orderQuantity int not null check(orderQuantity>0),

primary key(orderID,goodsID),

foreign key(orderID) references orders(orderID),

foreign key(goodsID) references goods(goodsID)

);

Ниже представлен оператор для создания таблицы «Услуги»:

create table [services](

serviceID int identity not null primary key,

serviceName nvarchar(50) not null,

cost decimal(10,2) not null check(cost>=0),

serviceDesc nvarchar(200),

typeOfServID int not null,

foreign key(typeOfServID) references typesOfServices(typeOfServID)

);

Ниже представлен оператор для создания таблицы «Заказ услуг»:

create table orderingOfServ(

orderID int not null,

serviceID int not null,

workerID int not null,

dateOfServ date not null,

timeOfServ time not null,

primary key(orderID,serviceID),

foreign key(orderID) references orders(orderID),

foreign key(serviceID) references [services](serviceID),

foreign key(workerID) references workers(workerID)

);

В результате выполнения всех этих запросов были созданы таблицы базы данных. Их список представлен на рисунке 16.

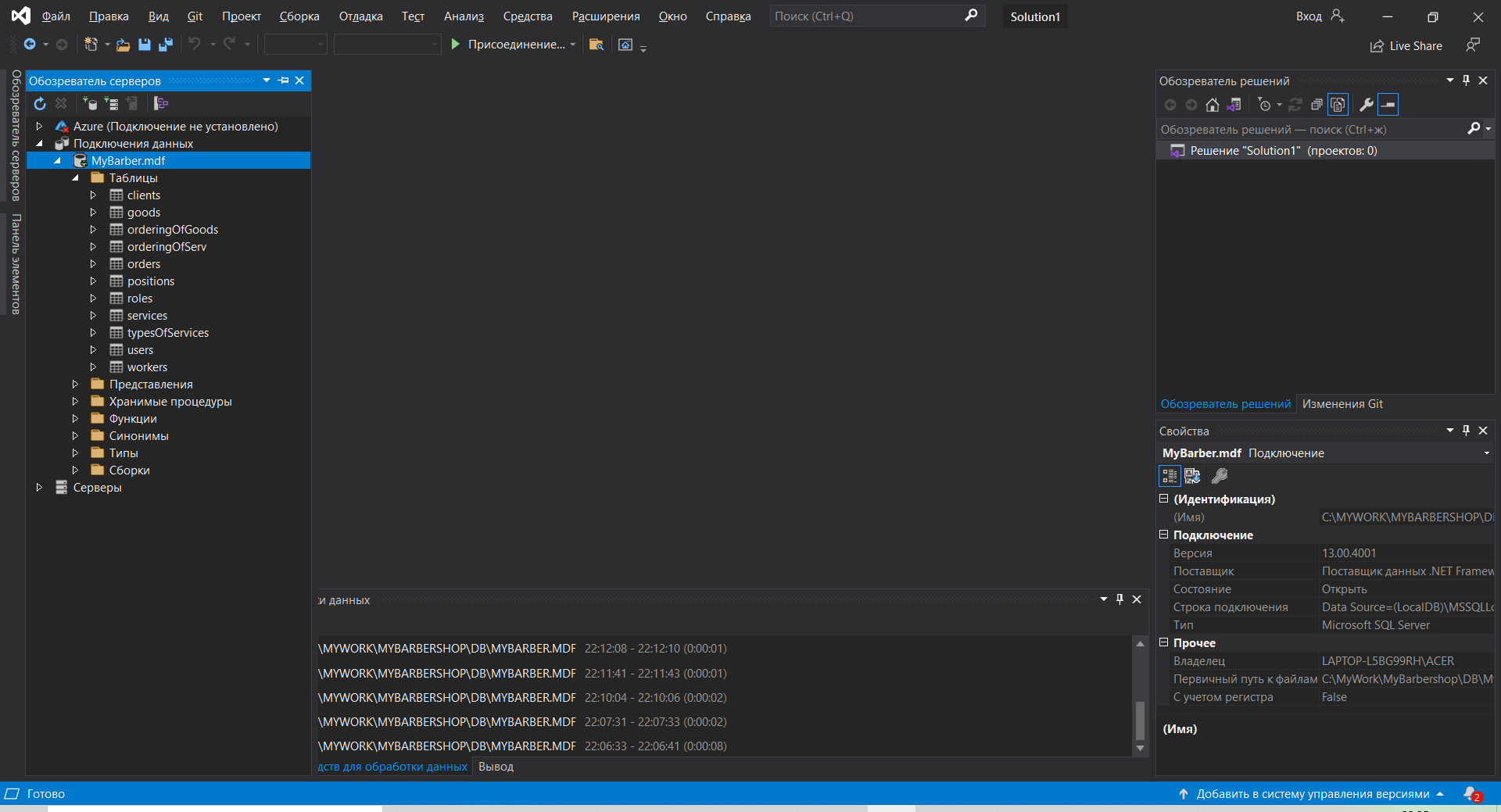


Рисунок 16 – Список созданных в базе данных таблиц

## 3.5. Ввод тестовых данных

После создания таблиц в них были внесены тестовые данные, позволяющие в дальнейшем оценить работу приложения. Внесенные данные представлены ниже (рисунки 17- 27 ).

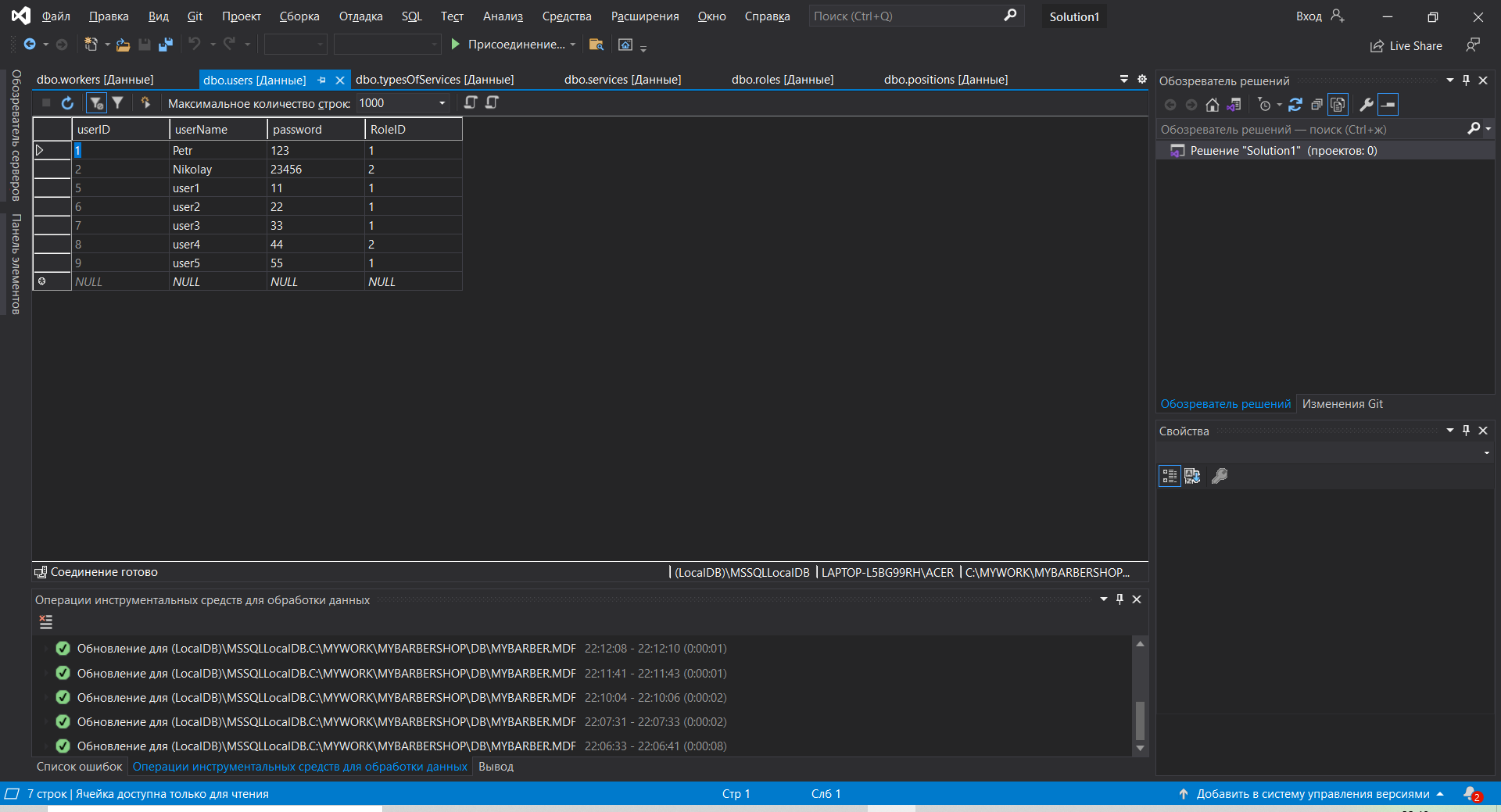


Рисунок 17 – Данные в таблице «Пользователи»

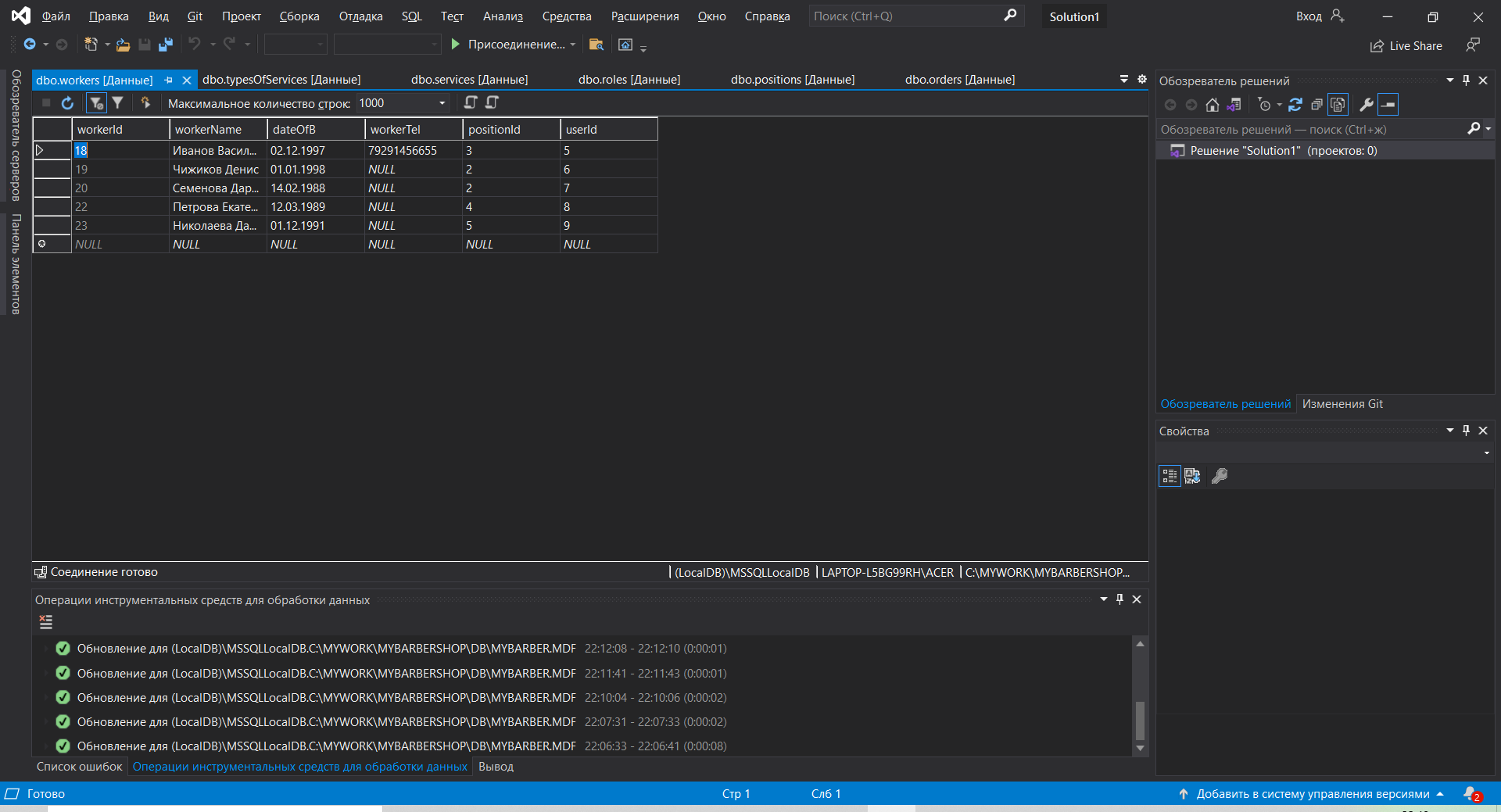


Рисунок 18 – Данные в таблице «Сотрудники»

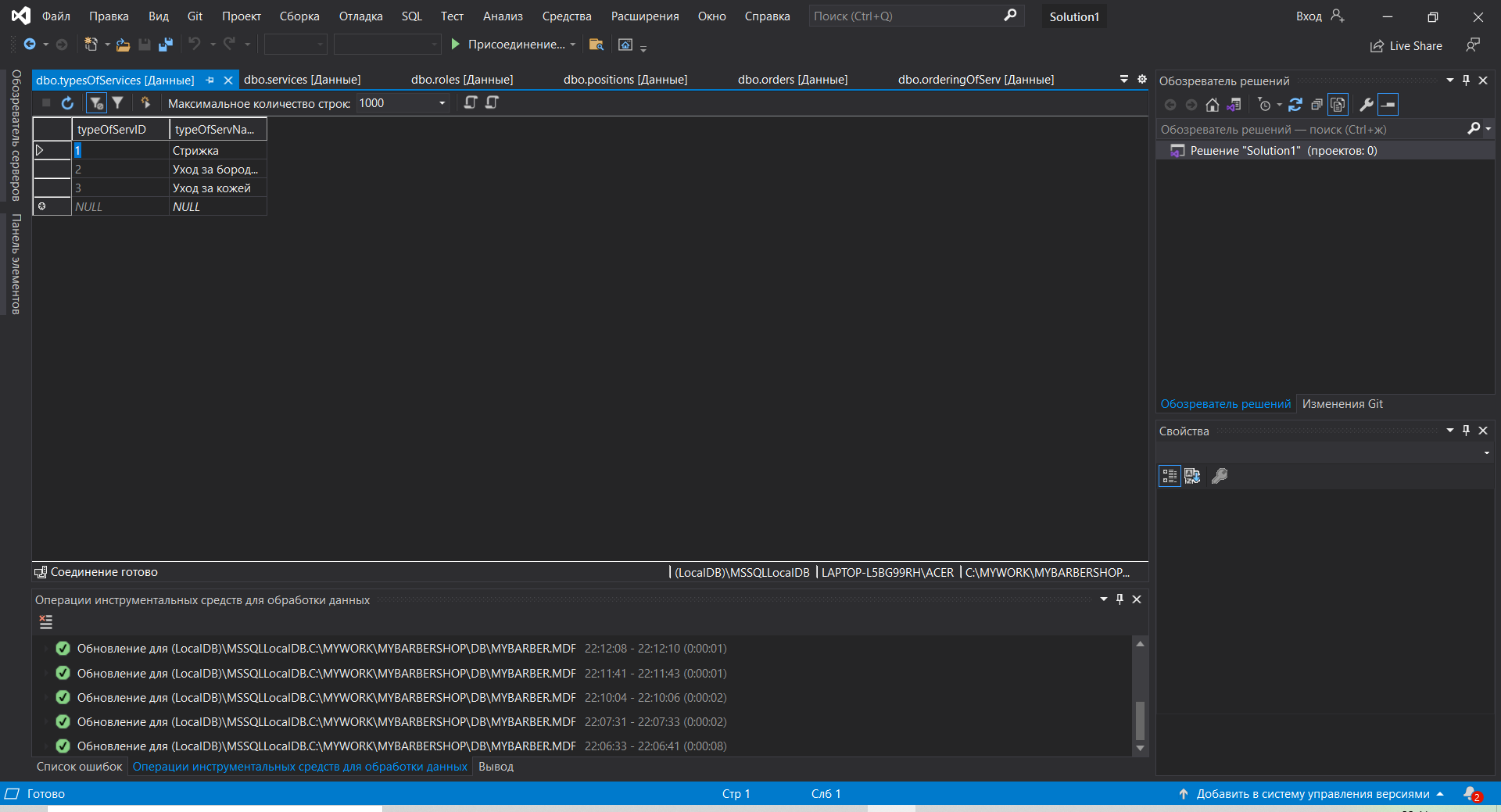


Рисунок 19 – Данные в таблице «Виды услуг»

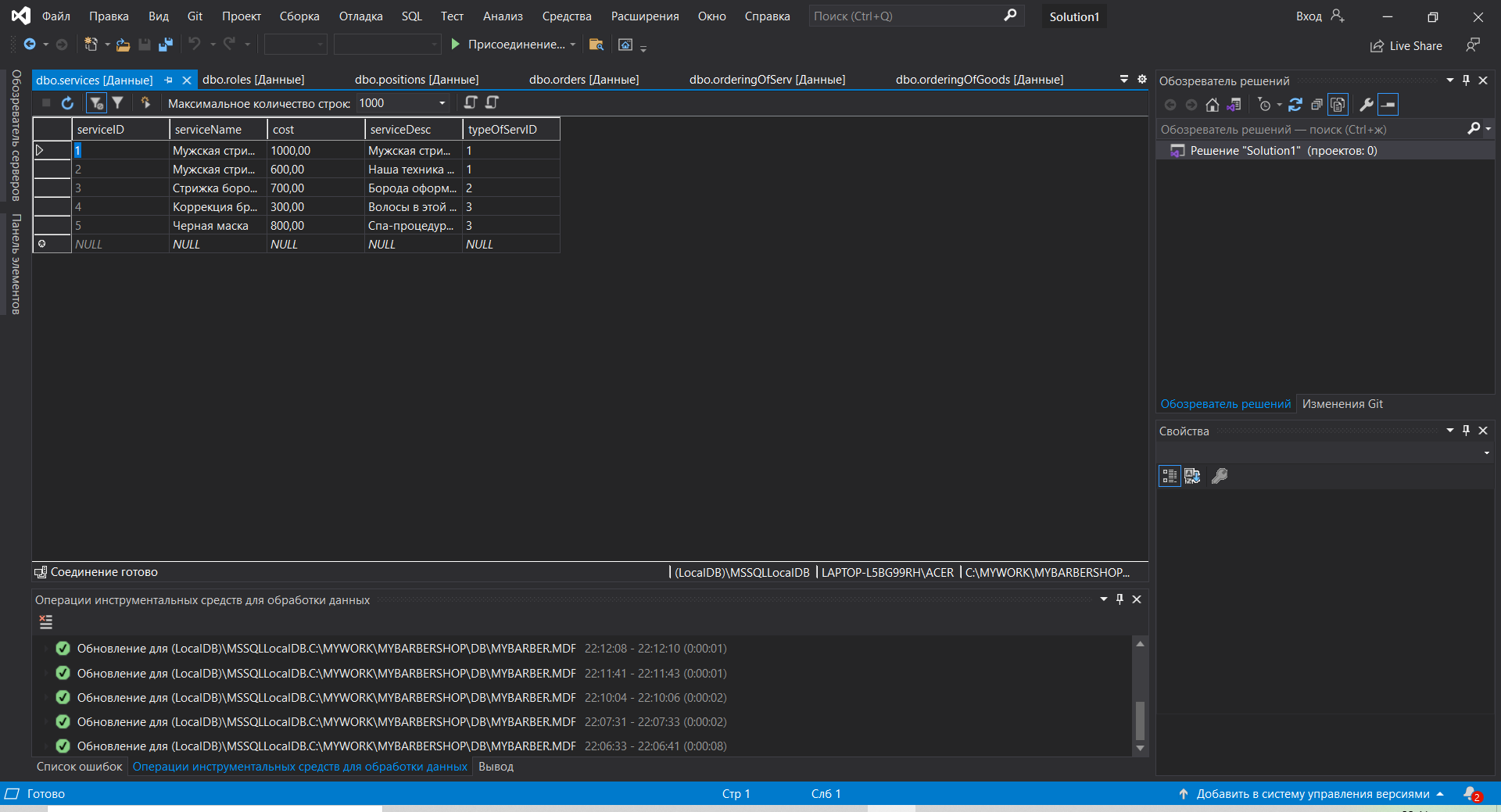


Рисунок 20 – Данные в таблице «Услуги»

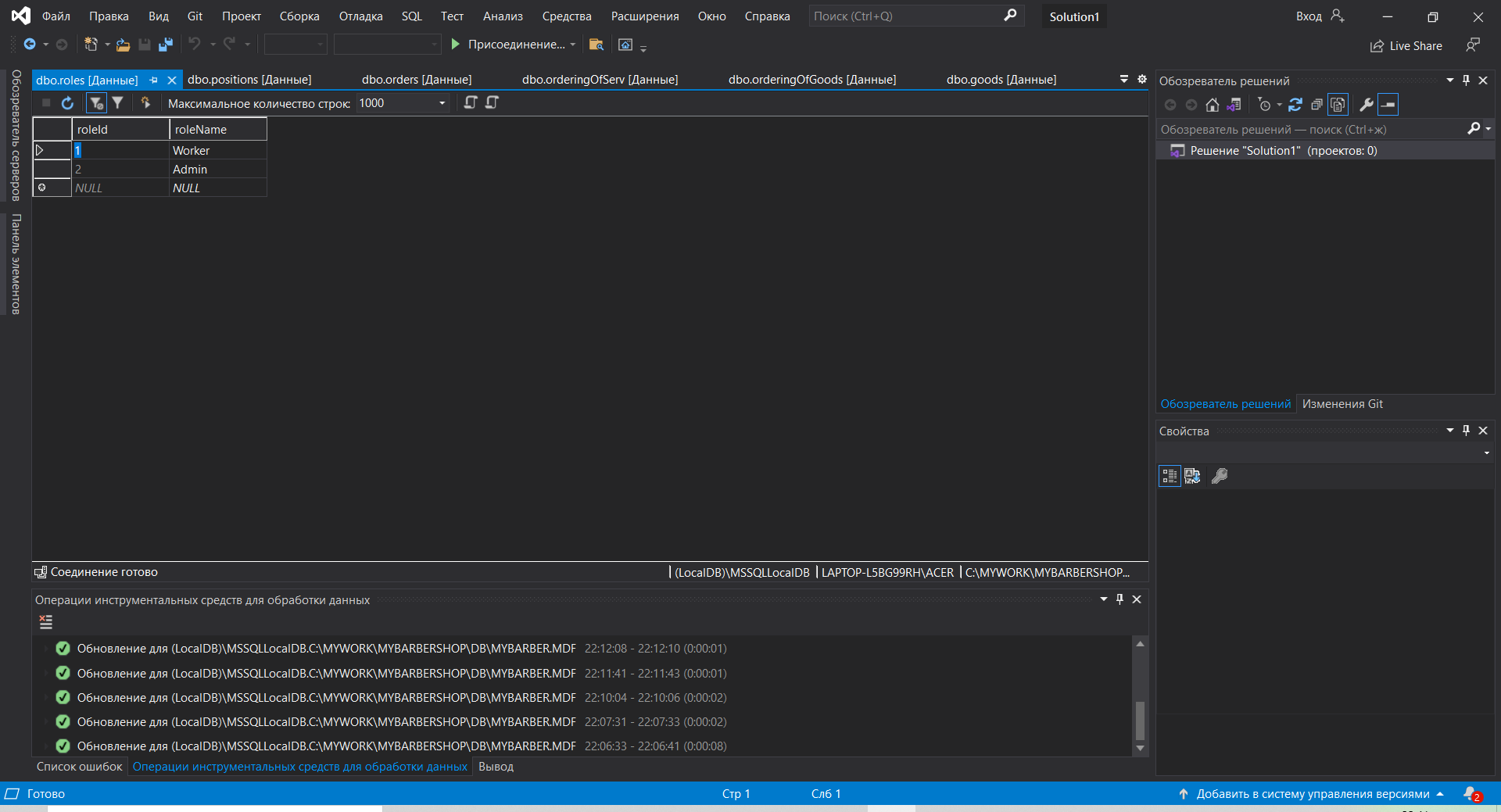


Рисунок 21 – Данные в таблице «Роли пользователей»

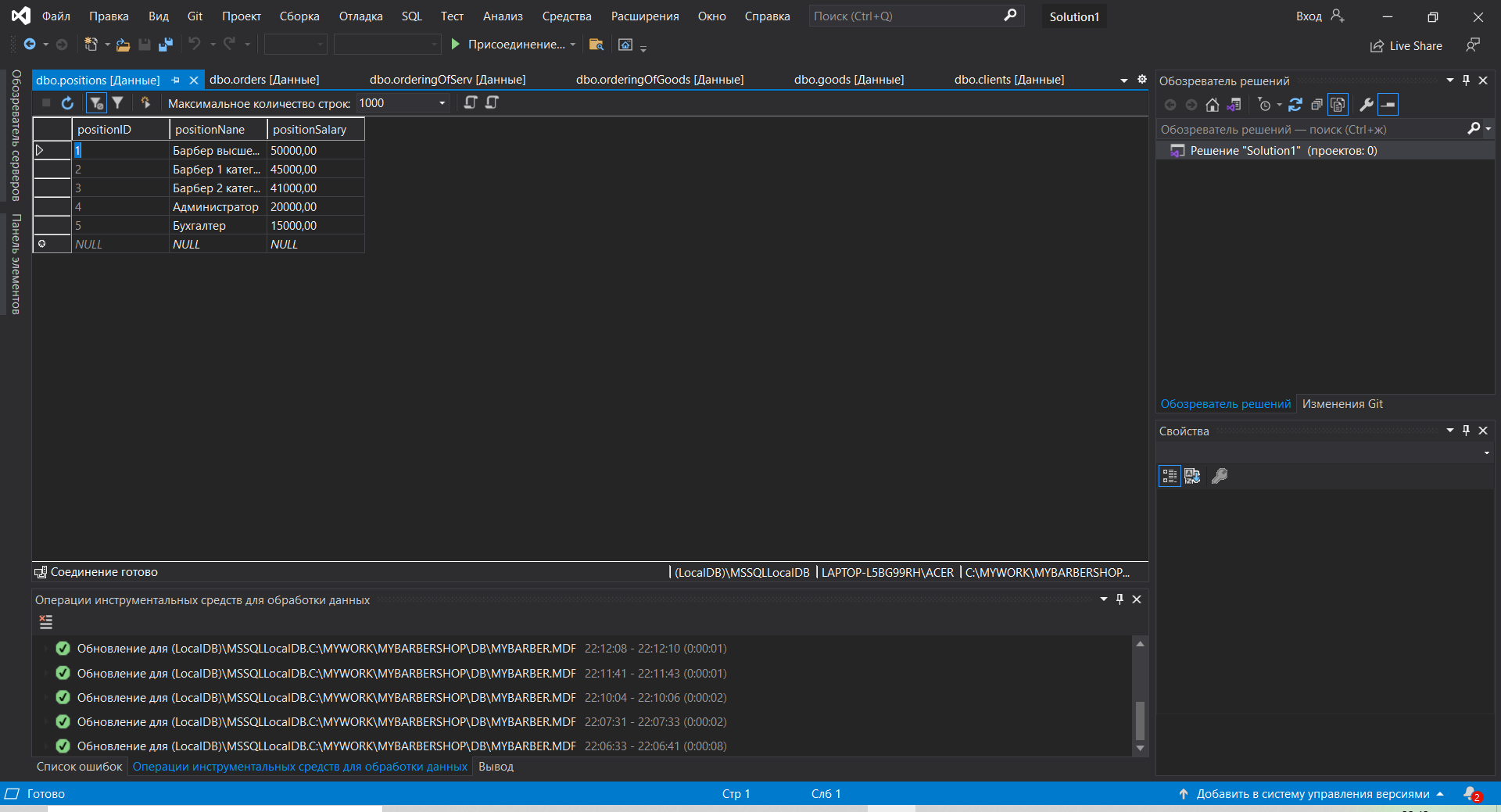


Рисунок 22 – Данные в таблице «Должности»

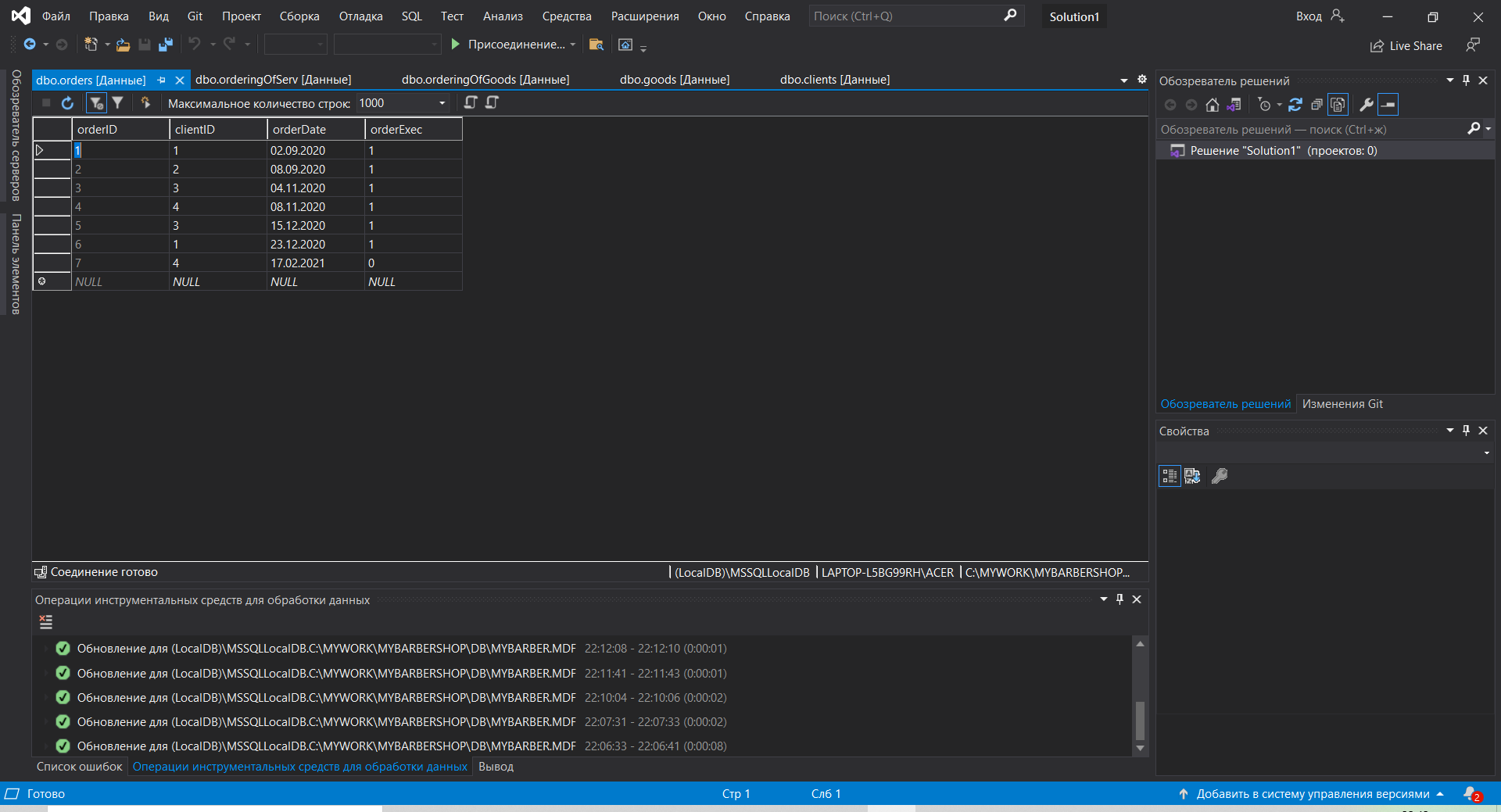


Рисунок 23 – Данные в таблице «Заказы»

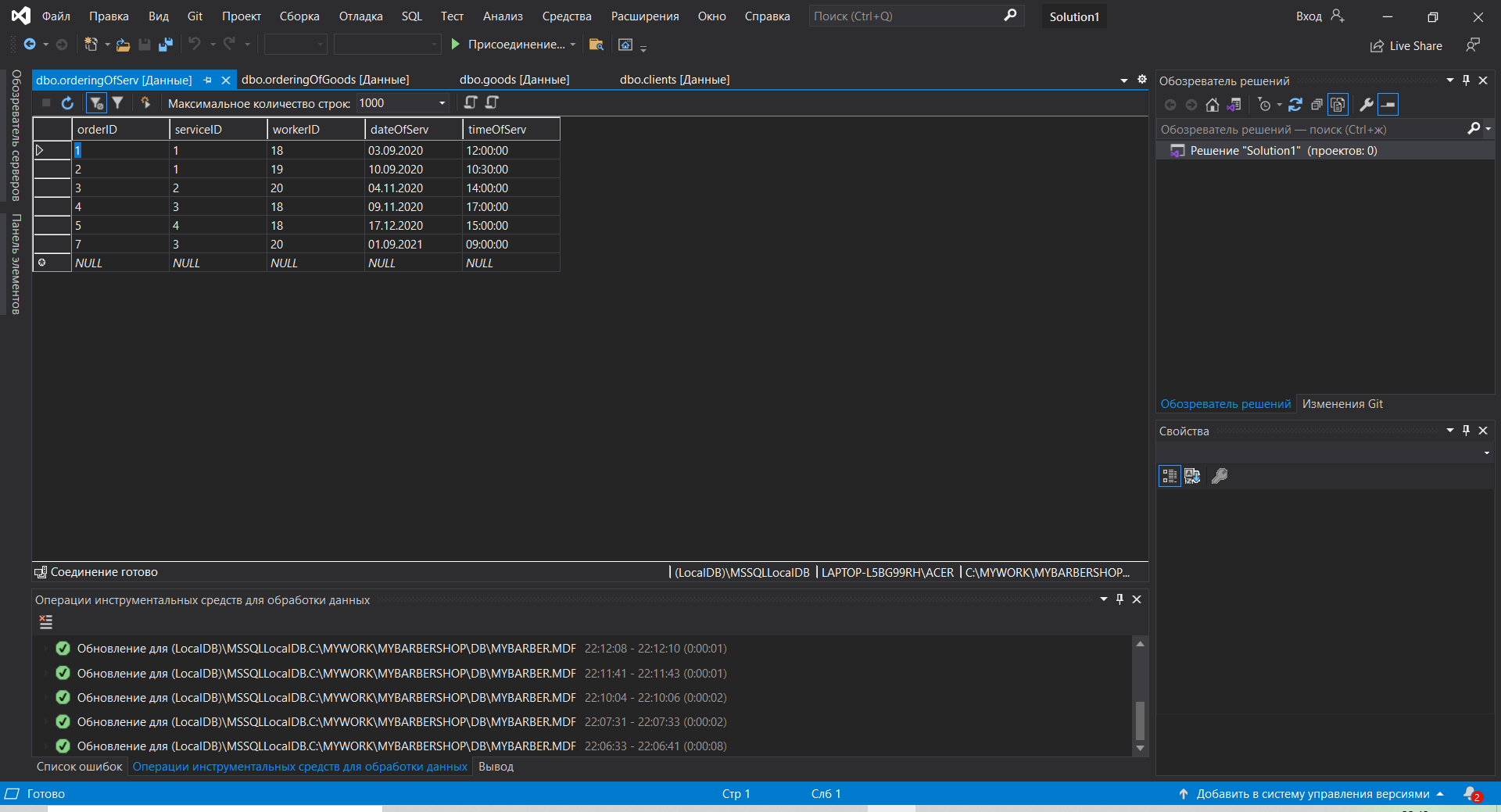


Рисунок 24 – Данные в таблице «Заказ услуг»

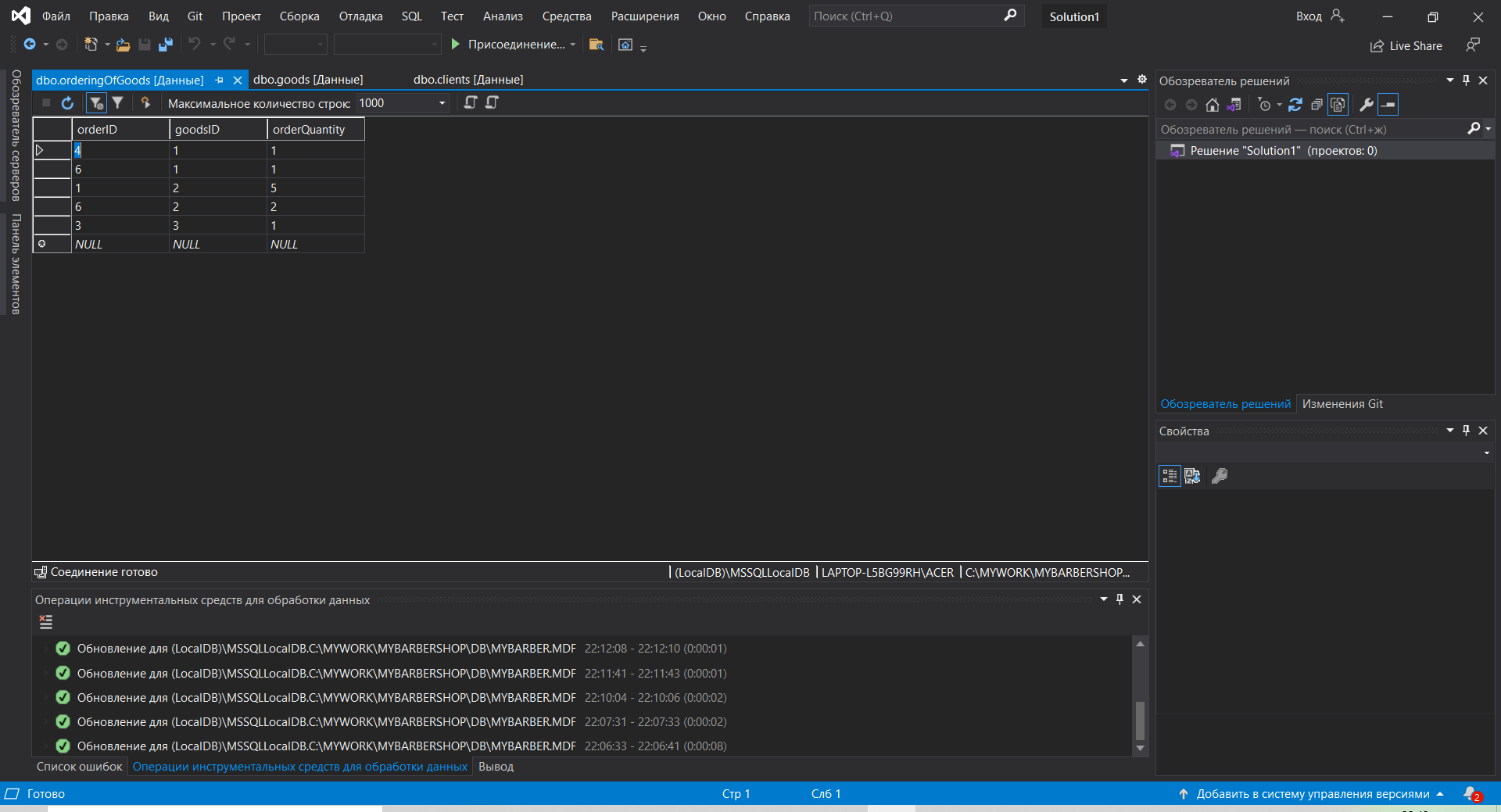


Рисунок 25 – Данные в таблице «Заказ товаров»

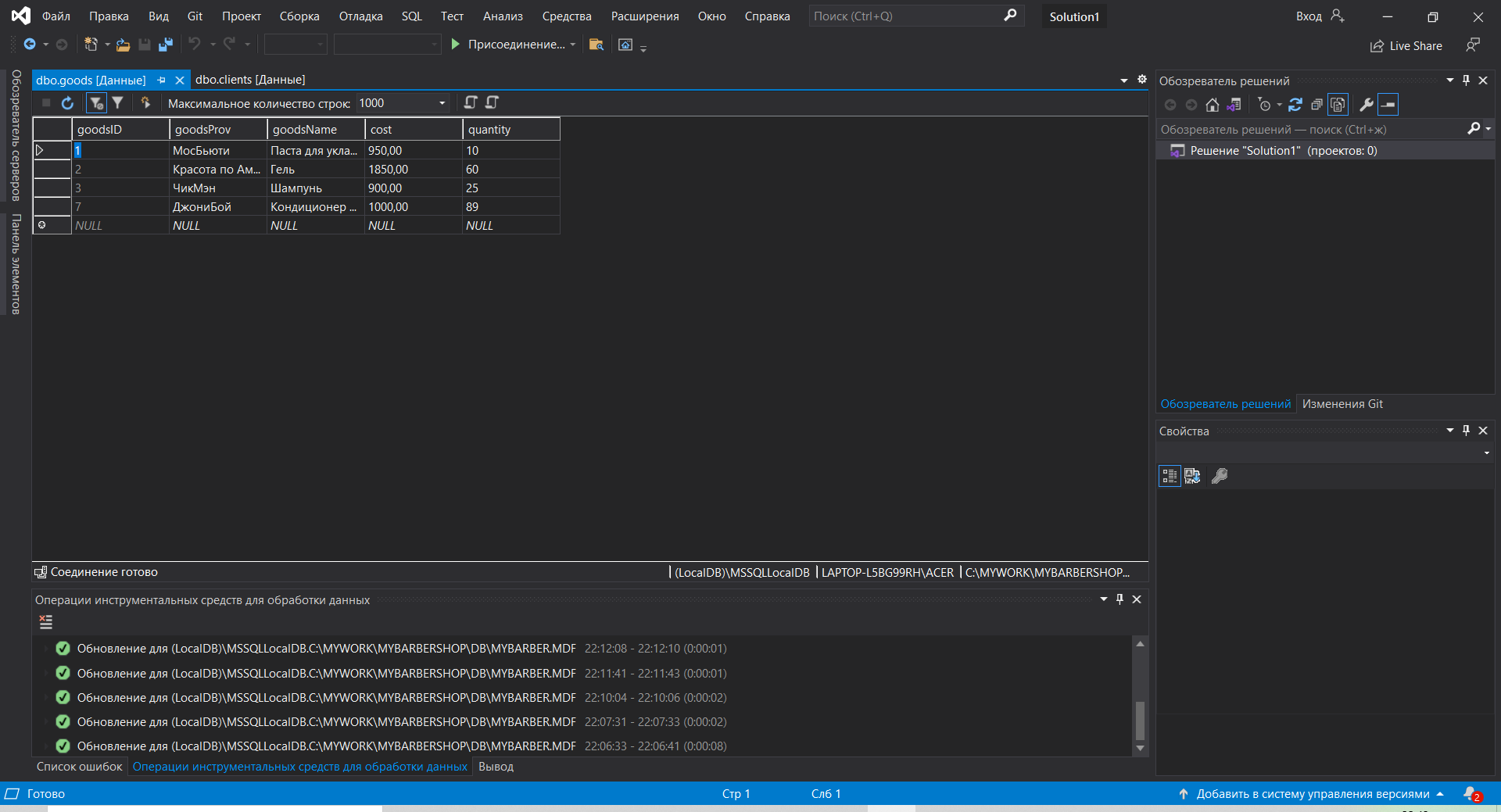


Рисунок 26 – Данные в таблице «Товары»

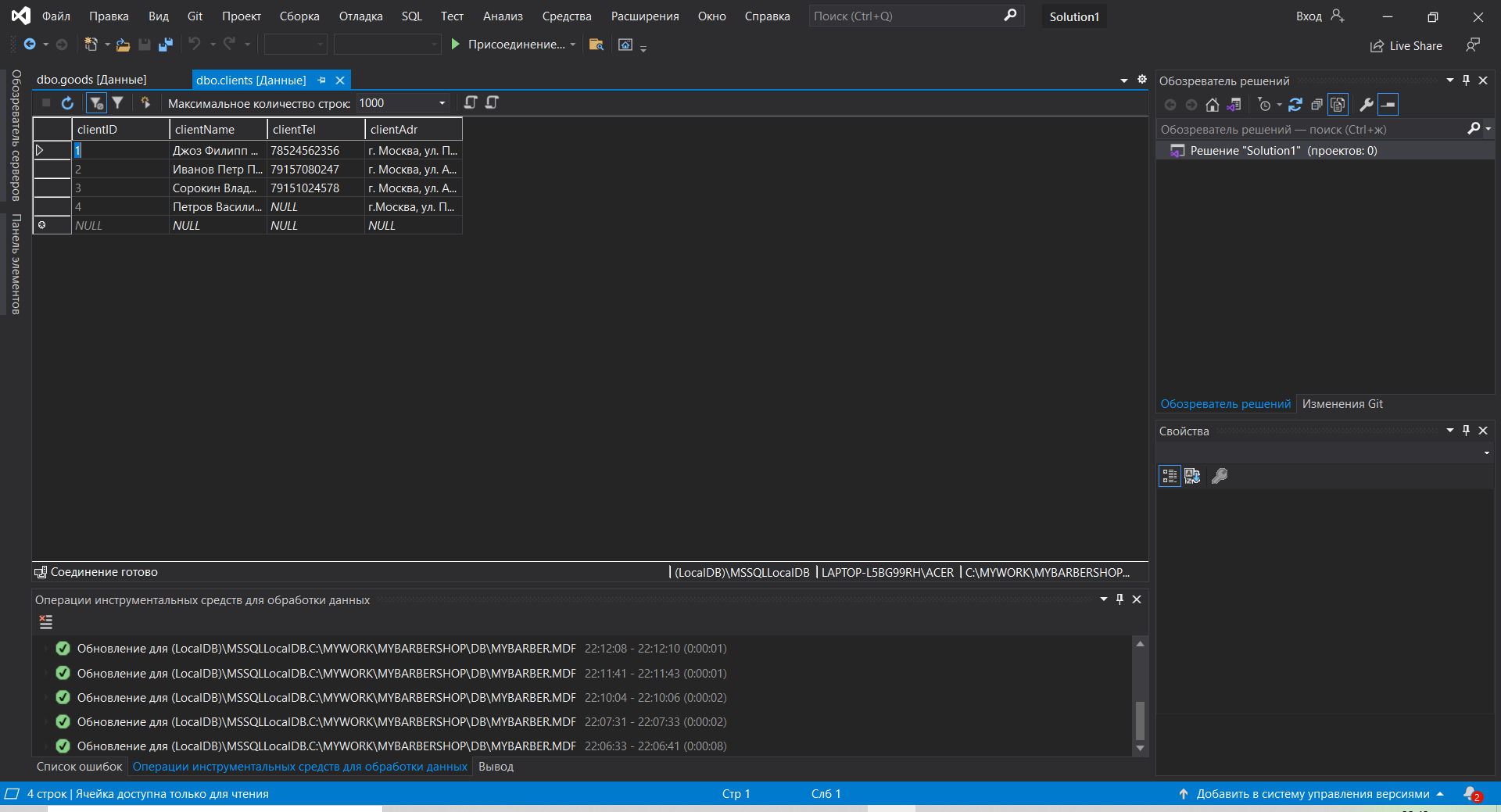


Рисунок 27 – Данные в таблице «Клиенты»

## 3.6. Создание просмотров

Представления или просмотры (views) представляют собой виртуальные таблицы, в которых, в отличии от обычных таблиц в базе данных, содержатся запросы select, динамически извлекающие используемые данные.

Использование просмотров имеет свои преимущества: они ускоряют выполнение часто сложных SQL-запросов (т.к. просмотр - это объект базы данных, хранящийся на сервере в готовом к исполнению виде), защищают данные (просмотр обеспечивает доступ к части полей таблицы), позволяют получать данные в нужном и удобном для пользователя форме. У разных групп пользователей свои потребности в информации и в формате их представления. Именно просмотр позволяет различным пользователям иметь свое представление данных из общей информационной системы.

Для создания представления используется SQL- оператор CREATE VIEW, имеющий следующий формат:

CREATE VIEW название\_просмотра[(столбец\_1,…, столбец\_N)]

AS оператор\_SELECT

В ходе разработки базы данных были созданы несколько просмотров, имеющих значение для пользователей информационной подсистемы барбершопа.

Ниже представлен код представления, позволяющий получить список заказов на исполнение услуг, которые еще не оказаны (клиент лишь записался к мастеру). Данное представление в дальнейшем может послужить исходными данными для хранимой процедуры, выбора предстоящих заказов на определенный день (дата будет передаваться в качестве параметра процедуры).

CREATE VIEW vExpectedOrders

AS

SELECT orders.orderID,orderingOfServ.dateOfServ,

orderingOfServ.timeOfServ,clients.clientName,

workers.workerName,[services].serviceName

FROM orders,orderingOfServ,clients,workers, [services]

WHERE orders.orderID=orderingOfServ.orderID

AND orders.clientID=clients.clientID

AND orderingOfServ.workerID=workers.workerId

AND orderingOfServ.serviceID=[services].serviceID

AND orders.orderExec=0;

В данном представлении для простоты восприятия использовалось внутреннее соединение таблиц в разделе WHERE. Его можно заменить конструкцией JOIN… ON в разделе FROM.

Результат вызова данного представления представлен на рисунке 28.

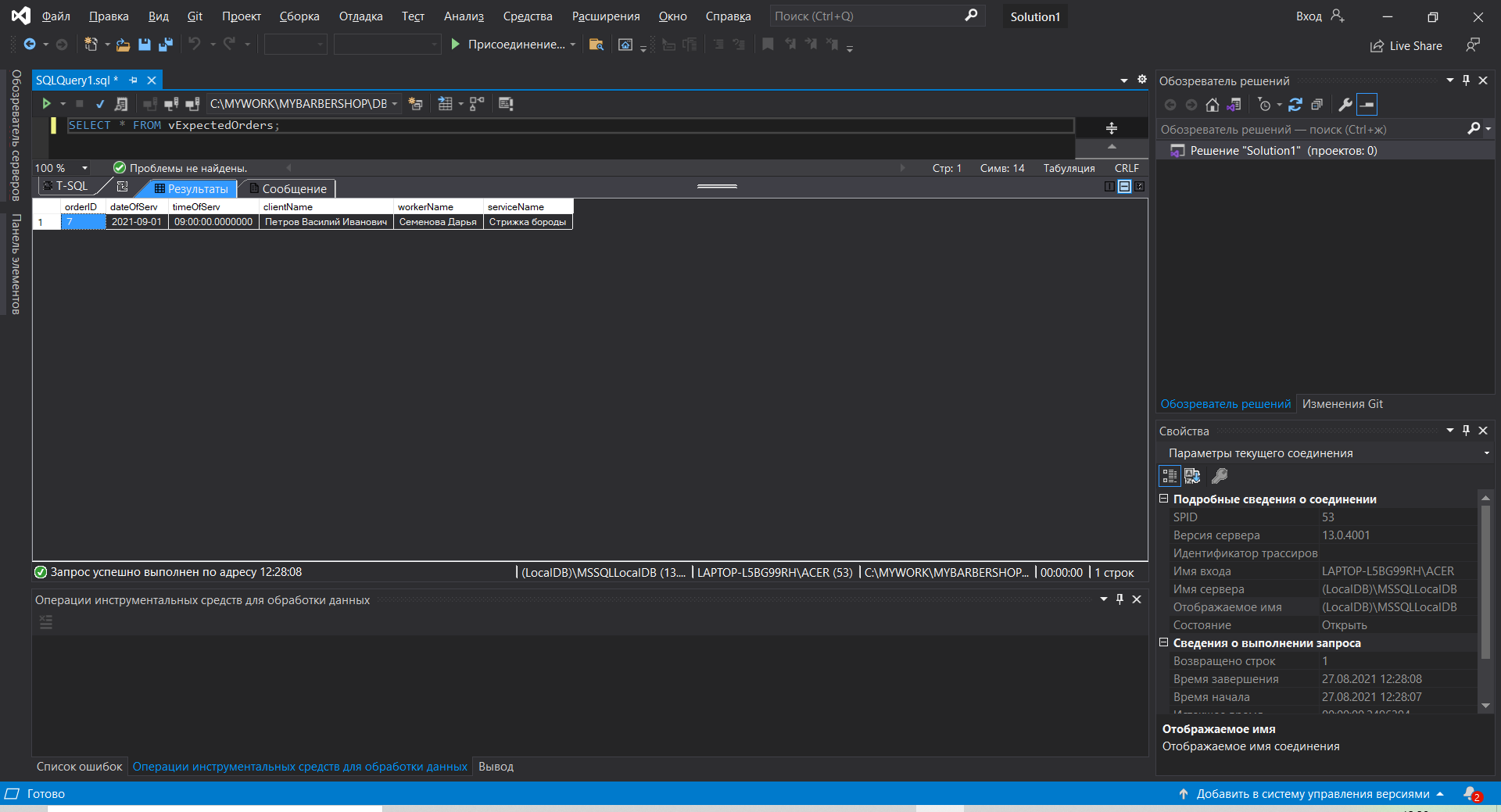


Рисунок 28 – Результат вызова представления vExpectedOrders

Префикс v (View) в имени представления позволяет программисты легче ориентироваться во множестве имен объектов.

В ходе разработки было создано еще одно представление, формирующее список заказов с указанием их общей стоимости. В данном представлении учитываются только те заказы, которые помечены как исполненные. Результат вызов данного просмотра представлен на рисунке 29.

CREATE VIEW vAmountOfOrders

AS

SELECT orders.orderID, ISNULL(ordG.sG,0)+ISNULL(ordS.sS,0) AS 'Amount'

FROM ((orders LEFT JOIN

(SELECT orderingOfGoods.orderID as oGID,

sum(orderingOfGoods.orderQuantity\*goods.cost) as sG

FROM orderingOfGoods JOIN goods ON

orderingOfGoods.goodsID=goods.goodsID

GROUP BY orderingOfGoods.orderID) ordG

ON orders.orderID=ordG.oGID) LEFT JOIN

(SELECT orderingOfServ.orderID as oSID,

sum([services].cost) as sS

FROM orderingOfServ JOIN [services] ON

orderingOfServ.serviceID=[services].serviceID

GROUP BY orderingOfServ.orderID) ordS

ON orders.orderID=ordS.oSID)

WHERE orders.orderExec=1;

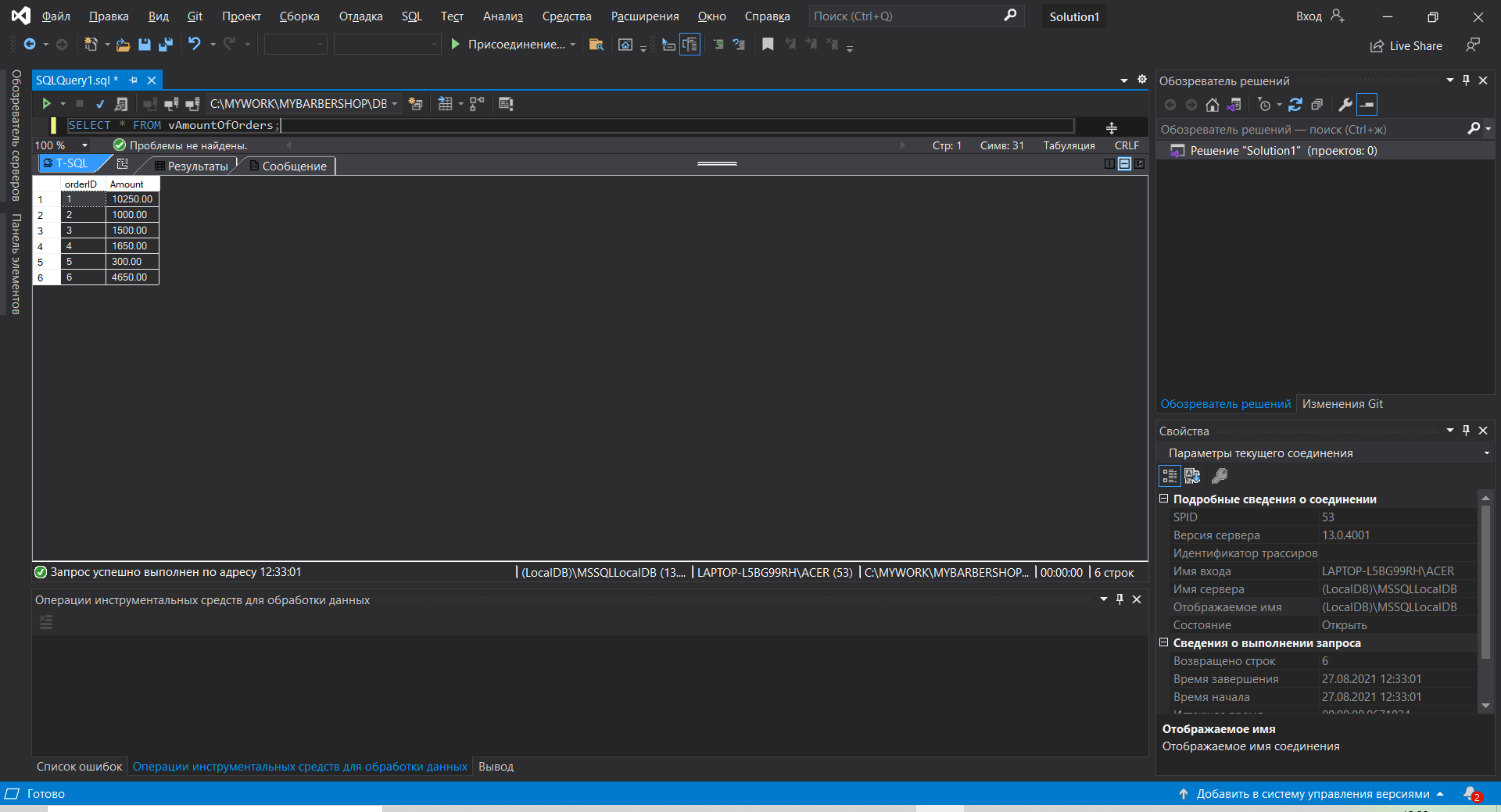


Рисунок 29 – Результат вызова представления vAmountOfOrders

Обращение к представлению осуществляется аналогично выборке данных из физически существующих таблиц.

## 3.7. Создание триггеров

Триггер представляет собой разновидность хранимой процедуры, отличающуюся от нее тем, что триггер не вызывается пользователем, а выполняется автоматически при возникновении того или иного события (добавление, удаление, изменение), связанного с таблицей, к которой он привязан. Он может запускаться после совершения события, либо вместо него (существуют СУБД, где триггер может выполняться перед событием). Триггеры служат для реализации тех ограничений, которые не реализованы (либо невозможно реализовать) на физическом уровне (в момент создания таблицы). Такие ограничения чаще всего связаны либо с обеспечением целостности данных, либо с теми правилами предметной области (бизнес-правилами), которые затрагивают данные из нескольких таблиц.

В барбершопе таким ограничением является ограниченный запас сопутствующих товаров на складе. При добавлении товара в заказ или при изменении числа товаров, необходимо в первую очередь проверить, не превышает ли заказанное число товара запас на складе, в дальнейшем- списать товар со склада.

При реализации данного ограничения возникла необходимость создать целый ряд триггеров, операторы создания которых представлены ниже.

Триггер controlGoodsQ проверяет достаточность товара на складе.

CREATE TRIGGER controlGoodsQ

ON orderingOfGoods

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @countGoods INT

DECLARE @goodsId INT

DECLARE @countOrderGoods INT

SET @countOrderGoods=(select inserted.orderQuantity from inserted)

SET @goodsId=(select goodsID from inserted)

SET @countGoods=(select goods.quantity from goods where goods.goodsID=@goodsId)

IF @countOrderGoods>@countGoods

BEGIN

PRINT 'CountOrder>CountGoods';

THROW 60000, 'CountOrder>CountGoods', 1;

END

END

Представленные ниже вызовы системной хранимой процедуры sp\_settriggerorder задают очередность выполнения триггеров: триггер контроля достаточности товара на складе controlGoodsQ всегда должен выполняться первым, как при добавлении товара в заказ, так и при редактировании объема заказанного товара.

exec sp\_settriggerorder @triggername = 'controlGoodsQ', @order = 'first', @stmttype = 'INSERT';

exec sp\_settriggerorder @triggername = 'controlGoodsQ', @order = 'first', @stmttype = 'UPDATE';

Представленный ниже триггер выполняет списание товара со склада при добавлении его в заказ.

CREATE TRIGGER debitGoodsQIns

ON orderingOfGoods

AFTER INSERT

AS

BEGIN

DECLARE @goodsId INT

DECLARE @countOrderGoods INT

SET @countOrderGoods=(select inserted.orderQuantity from inserted)

SET @goodsId=(select goodsID from inserted)

UPDATE goods

SET goods.quantity=goods.quantity-@countOrderGoods

WHERE goods.goodsID=@goodsId;

END

Представленный ниже триггер возвращает товар на склад при удалении его из заказа (сопутствующий товар не понадобился при оказании услуг, или клиент выбрал иной товар).

CREATE TRIGGER debitGoodsQDel

ON orderingOfGoods

AFTER DELETE

AS

BEGIN

DECLARE @goodsId INT

DECLARE @countOrderGoods INT

SET @countOrderGoods=(select deleted.orderQuantity from deleted)

SET @goodsId=(select goodsID from deleted)

UPDATE goods

SET goods.quantity=goods.quantity+@countOrderGoods

WHERE goods.goodsID=@goodsId;

END

Представленный ниже триггер выполняет редактирование объема товара на складе при редактировании заказа.

CREATE TRIGGER debitGoodsQUpd

ON orderingOfGoods

AFTER UPDATE

AS

BEGIN

DECLARE @goodsId INT

DECLARE @countOrderGoodsOLD INT

DECLARE @countOrderGoodsNEW INT

SET @countOrderGoodsOLD=(select deleted.orderQuantity from deleted)

SET @countOrderGoodsNEW=(select inserted.orderQuantity from inserted)

SET @goodsId=(select goodsID from deleted)

IF @countOrderGoodsOLD>@countOrderGoodsNEW

BEGIN

UPDATE goods

SET goods.quantity=goods.quantity+(@countOrderGoodsOLD-@countOrderGoodsNEW)

WHERE goods.goodsID=@goodsId;

END;

IF @countOrderGoodsOLD<@countOrderGoodsNEW

BEGIN

UPDATE goods

SET goods.quantity=goods.quantity-(@countOrderGoodsNEW-@countOrderGoodsOLD)

WHERE goods.goodsID=@goodsId;

END;

END

Таким образом созданный набор триггеров реализует на сервере необходимое для заданной предметной области ограничение.

## 3.8. Создание хранимых процедур

В ходе разработки информационной подсистемы возникла необходимость формировать список заказов на определенный день. Для этого было принято решение создать отдельную хранимую процедуру, принимающую дату в качестве параметра. Таким образом, вызвав ее мы может получить список заказов на указанный нами день.

Ниже представлен SQL-оператор создания такой процедуры. В качестве исходных данных процедура используется созданный ранее просмотр, так как он уже содержит все необходимые нам поля и необходимые объединения таблиц.

CREATE PROCEDURE xpExpectedOrders

@expDate DATE

AS

SELECT \* FROM vExpectedOrders

WHERE vExpectedOrders.dateOfServ=@expDate;

Оператор вызова данной процедуры имеет вид:

exec xpExpectedOrders '2021-09-01';

Результат вызова представлен на рисунке 30.

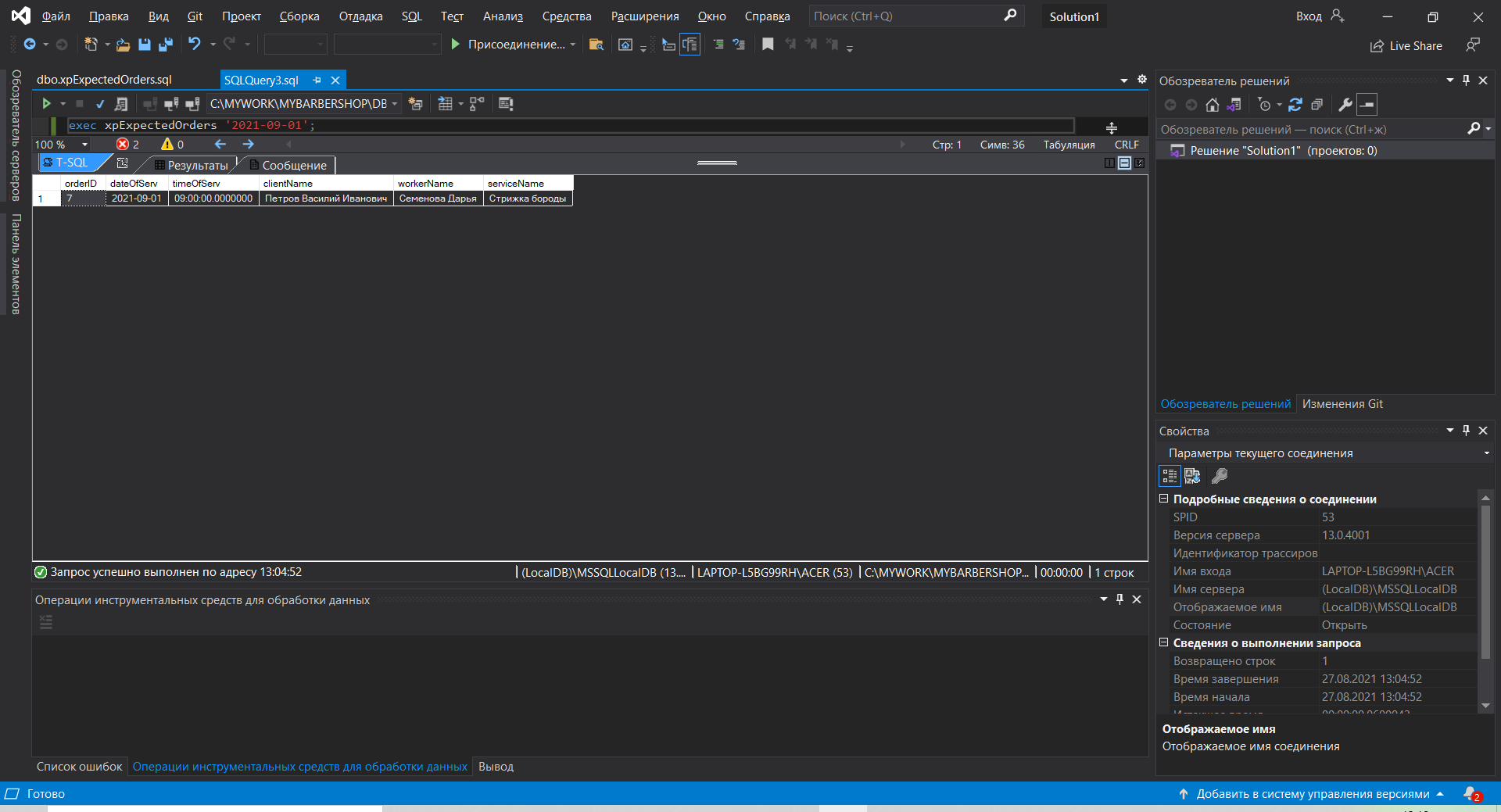


Рисунок 30 - Результат вызова хранимой процедуры xpExpectedOrders

На рисунке 31 представлен еще один результат вызова процедуры, который не содержит строк, так как никто из клиентов не записан на 25 сентября 2021 года.

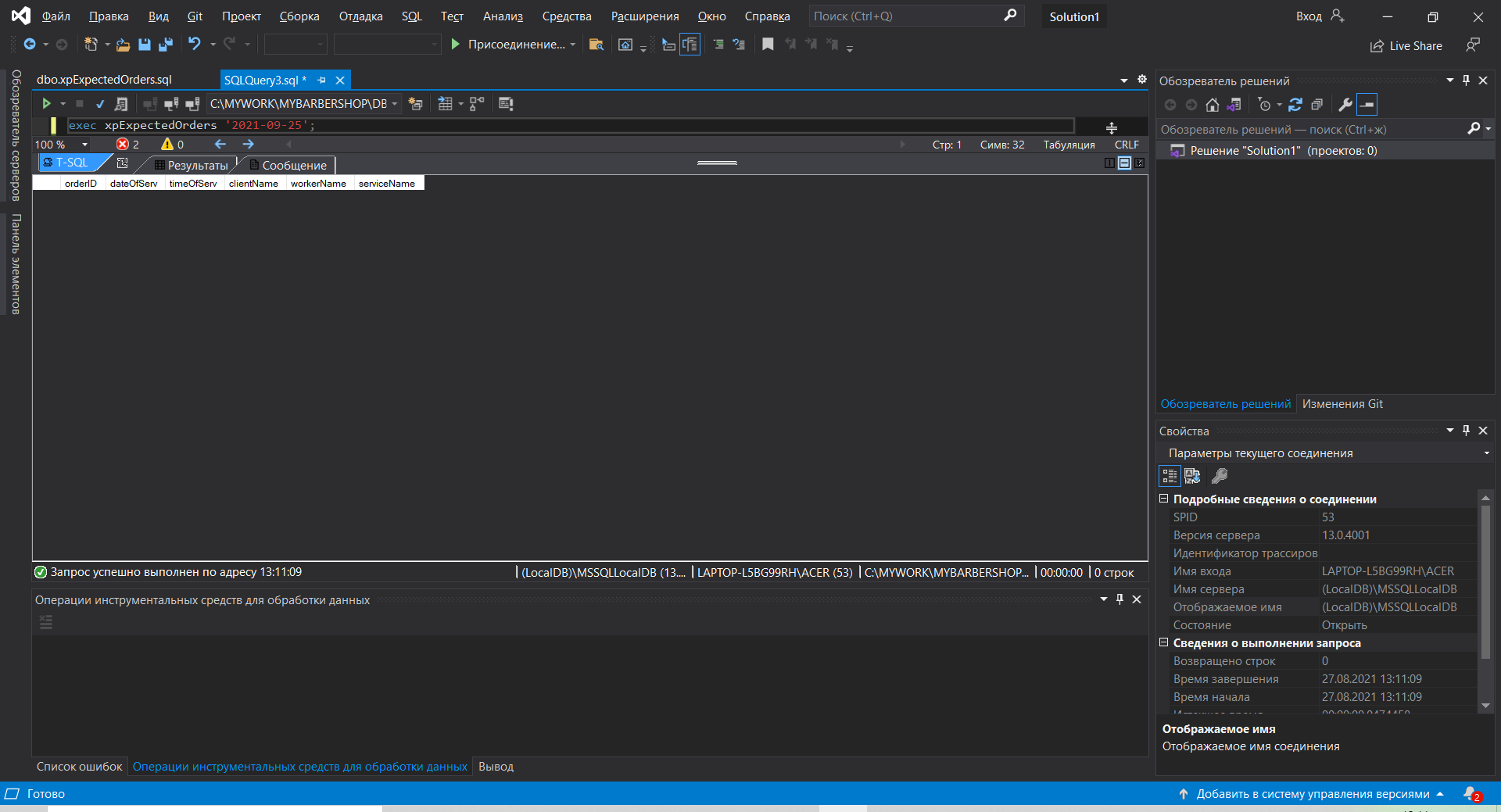


Рисунок 31 - Результат вызова хранимой процедуры xpExpectedOrders при отсутствии записанных в указанный день клиентов

## 4. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

В данном разделе курсового проекта будет описана разработка клиентского приложения для администратора барбершопа, проведенная в ходе выполнения проекта.

## 4.1. Разработка рабочего модуля ИС

При разработке приложения были выделены модули, отвечающие за презентационные функции приложения (модули форм) и модуль, отвечающий за взаимодействие с базой данных. В него были помещены те функции, которые необходимы для обращения к объектам базы данных, выполнения запросов. Это обеспечивает независимость визуальной части приложения от логики работы с базой данных, в которую могут вноситься изменения, независимо от изменений пользовательского интерфейса. Данное требование является довольно существенным, так как информационная подсистема в дальнейшем будет совершенствоваться, будут расширяться (либо изменяться) ее функциональные возможности.

Ниже будет приведен фрагмент модуля-класса взаимодействия с базой данных:

class MyConnection

{

//Класс отвечающий за взаимодействие с базой

//данных

//в этой строке подключения указывается путь к файлу базы данных

string conStr = "Data Source=(LocalDB)[\\MSSQLLocalDB](file:///\\MSSQLLocalDB);

AttachDbFilename=C:\\MyWork\\MyBarbershop\\DB\\MyBarber.mdf;Integrated Security=True;Connect Timeout=30";

public int roleId;

public SqlConnection connection;

public MyConnection()

{

connection = new SqlConnection(conStr);

}

public bool AuthorizeUser(string userName,string password)

{

//метод авторизации пользователя

}

public void ShowWorkers(System.Windows.Forms.DataGridView dg,int typeOfShow)

{

//метод отображения информации о сотрудниках

//Параметр typeOfShow позволяет задать режим отображения: все данные, с //сортировкой по ФИО, дате рождения.

}

public void FindWorker(System.Windows.Forms.DataGridView dg, string workerTel)

{

//метод поиска информации о сотруднике

}

//далее в классе по аналогичному принципу реализованы методы работы с клиентами и

//их заказами

}

Во всем проекте сохранен единый стиль наименований, отражающий функциональное назначение методов, переменных, компонентов.

Проект включает в себя комментарии как на уровне классов, так и методов. Внутри методов комментарии поясняют назначение существенных для понимания фрагментов кода.

## 4.2. Демонстрация работы приложения

После запуска приложения пользователю отображается окно авторизации, представленное на рисунке 32.

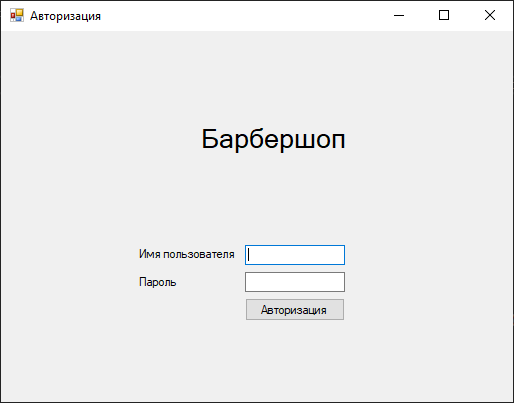


Рисунок 32 – Окно «Авторизация»

После успешной авторизации пользователь переходит к предназначенному для него меню, включающее в себя доступные пользователю функции.

Для демонстрации в рамках курсового проекта выберем функцию «Просмотр списка сотрудников». Форма представлена на рисунке 33.

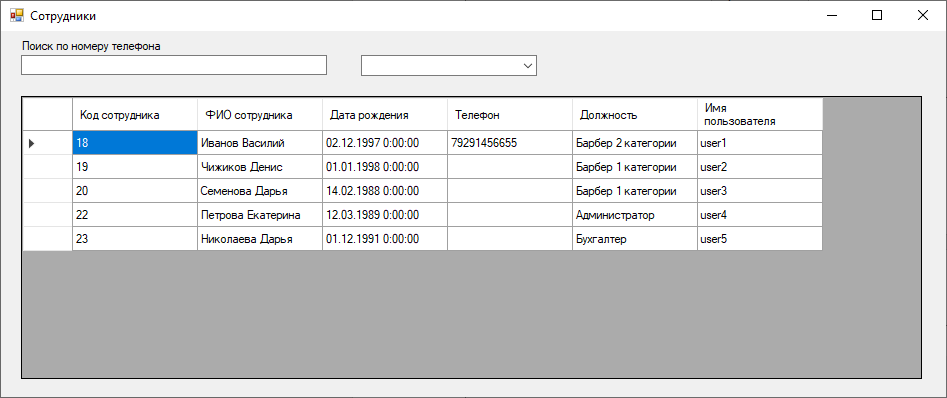


Рисунок 33 – Окно «Сотрудники»

Форма «Сотрудники» имеет простой и понятый пользовательский интерфейс, позволяющий пользователю легко ориентироваться в доступных ему функциональных возможностях.

С помощью данного окна пользователь получает возможность:

* просматривать список сотрудников;
* осуществлять поиск по номеру телефона (как по полному, так и частичному совпадению);
* осуществлять сортировку данных по фамилии сотрудников, их дате рождения.

На рисунке 34 представлен результат сортировки в алфавитном порядке по ФИО сотрудника.

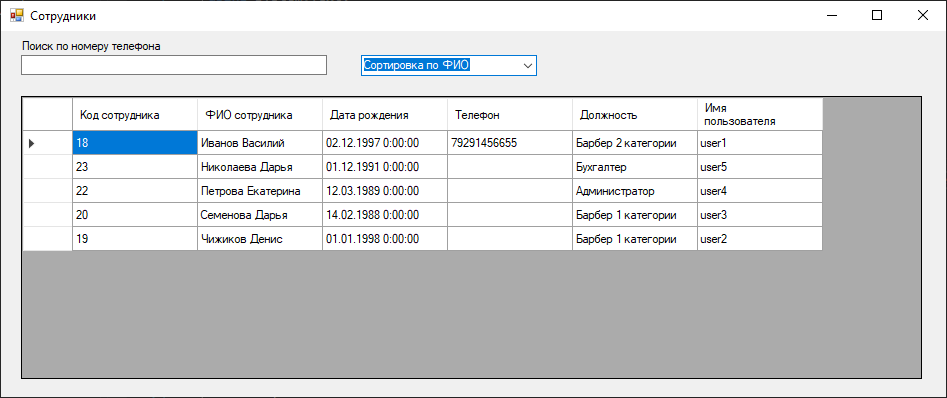


Рисунок 34 – Результат сортировки по ФИО сотрудника

На рисунке 35 представлен результат сортировки по дате рождения сотрудника.

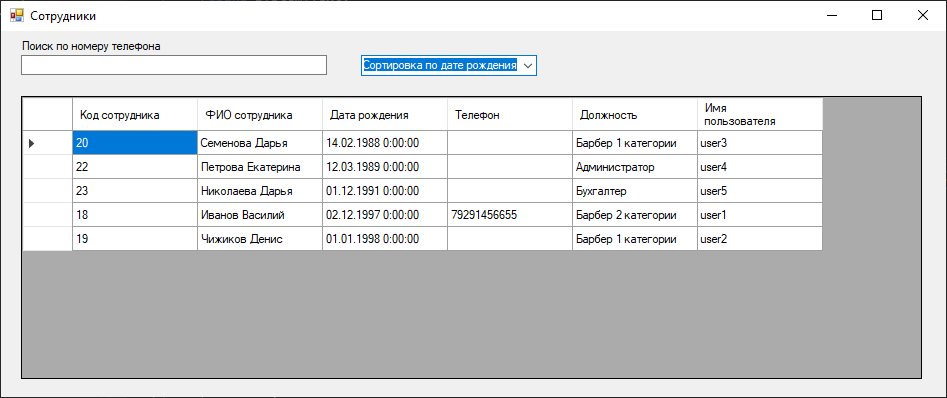


Рисунок 35 – Результат сортировки по дате рождения сотрудника

На рисунке 36 представлен результат поиска по части номера телефона сотрудника.

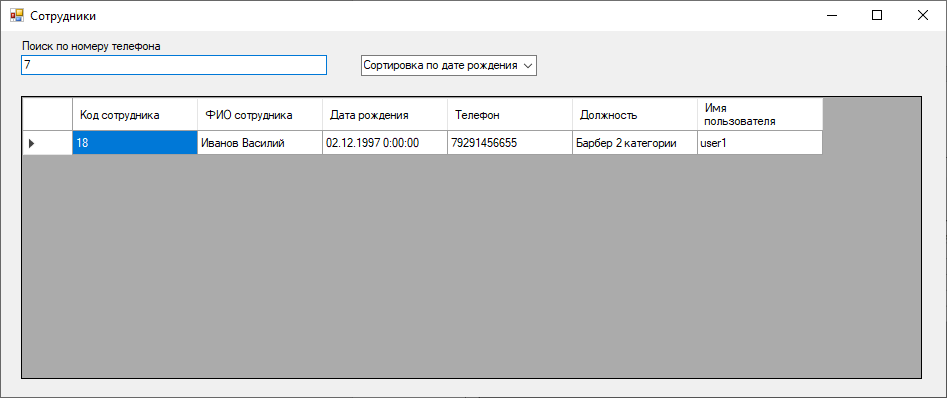


Рисунок 36 – Результат поиска по номеру телефона сотрудника

Остальные формы приложения выполнены в аналогичном стиле и обладают схожей функциональностью.

## ЗАКЛЮЧЕНИЕ

Целью курсового проектирования является автоматизация хранения и обработки данных в информационной системе для вашей организации. Для достижения поставленной цели были решены следующие задачи:

1. Проведен анализ предметной области;
2. Разработаны концептуальная и логическая модели базы данных;
3. Разработана серверная часть информационной подсистемы;
4. Разработано клиентское приложение.

Учитывая, что для разработки подсистемы был использован учебный (упрощенный) вариант модели предметной области, в разработанной подсистеме учтены не все данные (как информационные объекты, так и поля) которые имеют место в реальной работе барбершопа.

Разработанная информационная подсистема отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации.

Созданная информационная подсистема позволяет решать поставленные перед ней задачи.

Разработанная подсистема может быть в дальнейшем доработана в части расширения функциональных возможностей, так как в рамках курсового проекта были реализованы лишь базовые функции.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Роберт Э. Уолтерс, Майкл Коулс. SQL Server 2008: ускоренный курс для профессионалов = Accelerated SQL Server 2008. — М.: «Вильямс», 2008.
2. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс = Beginning Microsoft SQL Server 2005 Programming. — М.: «Диалектика», 2007.
3. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Освоение Microsoft SQL Server 2005 = Mastering Microsoft SQL Server 2005. — М.: «Диалектика», 2007
4. Microsoft® SQL Server™ 2005.Реализация и обслуживание. Учебный курс Microsoft (Экзамен 70-431). — М.: «Питер», 2007.
5. Петкович Д. Microsoft SQL Server 2008. Руководство для начинающих. — C.: БХВ-Петербург, 2009.
6. Малыхина М. П., Базы данных: основы, проектирование, использование, Санкт-Петербург: БХВ-Петербург, 2006.
7. Агальцов В. П. Базы данных. В 2-х кн. Кн. 2. Распределенные и удаленные базы данных: Учебник / В.П. Агальцов. — М.: ИД ФОРУМ: НИЦ Инфра-М, 2013.
8. Шустова Л. И. Базы данных: учебник / Л.И. Шустова, О.В. Тараканов. — М.: НИЦ ИНФРА-М, 2016.
9. Фуфаев Э.В. Базы данных — М.: ИЦ Академия, 2012.
10. Голицына О.Л., Максимов Н.В., Попов И.И. Базы данных — М: Форум, 2015
11. Кумскова И. А. Базы данных — М: КноРус, 2016
12. Советов Б. Я. Базы данных — М: Высшая школа, 2018
13. Федорова Г. Н. Основы проектирования баз данных — М: Академия, 2018
14. Илюшечкин В. М. Основы использования и проектирования баз данных — М: Высшая школа, 2018
15. Федорова Г. Н. Разработка, администрирование и защита баз данных —   
    М: Академия, 2018
16. Рябинина Л. Важность компьютерных игр - какие потребности они закрывают? / Рябинина Л. [Электронный ресурс] // b17.ru : [сайт]. — URL: https://www.b17.ru/blog/409586/?ysclid=m370hj71jz47839303 (дата обращения: 07.11.2024)