

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
профессионального образования «Ивановский государственный энергетический
университет имени В. И. Ленина».

Кафедра систем управления

Отчет по лабораторному практикуму
**«Проектирование баз данных и разработка приложений в СУБД
Microsoft SQL Server»**

Дисциплина: Системы управления базами данных

Вариант 10

Выполнили: студенты группы 2–43

Балашов И.

Ковшов Д.

Проверил: ассистент

Трифонов А. Ю.

Цель работы

Разработка и реализация базы данных для конкретной предметной области с использованием методов моделирования, таких как создание концептуальной, логической и физической моделей. В рамках работы необходимо определить таблицы пользовательских ролей и их функции, а также установить ограничения на вводимые данные, значения по умолчанию и правила удаления. Также требуется разработать и выполнить запросы на модификацию данных (Insert, Update, Delete) и запросы на выборку с использованием различных операторов и функций SQL, включая агрегацию, подзапросы и объединения. В процессе работы также предстоит создать хранимые процедуры и триггеры для автоматизации обработки данных и обеспечения целостности базы данных.

Задание

Оператор сотовой связи

Оператор сотовой связи МТС предоставляет услуги сотовой связи физическим и юридическим лицам по всей России. Юридические лица могут подключать только корпоративные тарифы, а физические – только некорпоративные. Для каждого клиента известно количество средств на счету. Для каждого тарифа указаны стоимость перехода на тариф, стоимость минуты разговоров (по городу, для междугородних и международных звонков). Все звонки фиксируются: кто звонит, кому звонят, время звонка, продолжительность разговора (целое количество минут), тип соединения (по городу, междугородний, международный). После окончания разговора со счета звонящего списывается определенная сумма. Каждый клиент вправе запросить распечатку звонков за любой период.

Проектирование

Создание модели предметной области начинается с определения абстракций, существующих в реальном мире, то есть ключевых концептуальных объектов, присутствующих в системе. Концептуальная модель представляет собой описание понятий через призму предметной области. Она отражает структуру проектируемой системы на высоком уровне абстракции, не привязанном к её физической реализации. При построении концептуальной модели предпочтительнее излишняя детализация, чем недостаточная проработка, поэтому в неё включаются все понятия предметной области, даже те, которые не планируется использовать при разработке базы данных.

Создадим к нашему заданию таблицу с ролями пользователей и их функциями.

Таблица 1. Роли пользователей и их функции

| Роль | Функции |
|-----------------------------|---|
| Физическое лицо | Просмотр собственного баланса, просмотр истории звонков, подключение/смена некорпоративного тарифа |
| Юридическое лицо | Просмотр баланса компании, просмотр звонков по всем номерам компании, подключение корпоративных тарифов |
| Оператор колл-центра | Просмотр данных клиентов, тарификации, истории звонков, смена тарифов по заявке клиента |
| Служба биллинга | Расчёт стоимости звонков, списание средств, ведение лога звонков, формирование отчетов |
| Администратор БД | Полный доступ ко всем таблицам и объектам базы данных, управление пользователями и ролями |

Теперь перейдём к созданию диаграмм.

1) Создание диаграммы классов (рис. 1)

В UML концептуальная модель изображается в виде статической диаграммы классов без указания операций. Такая диаграмма описывает типы объектов системы и различные виды статических связей между ними.

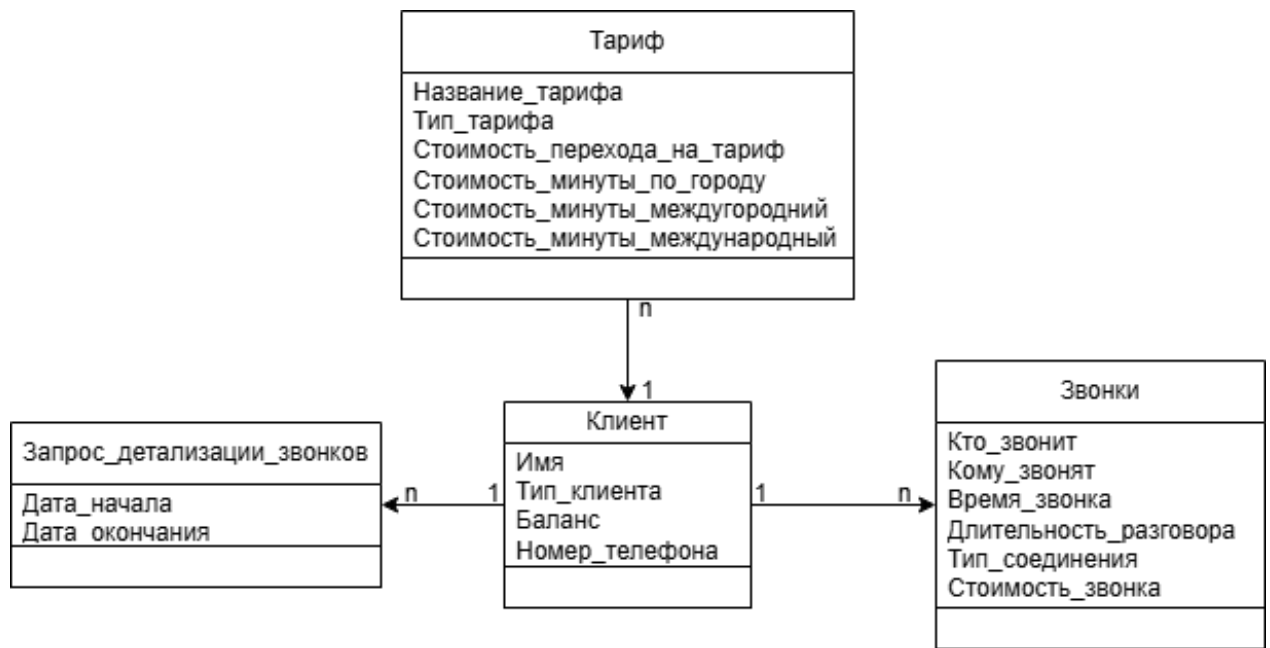


Рис. 1. Диаграмма классов.

2) Создание концептуальной модели базы данных (рис. 2)

Концептуальная модель базы данных описывает объекты предметной области, их атрибуты и взаимосвязи между ними в той степени, в которой они подлежат непосредственному сохранению в базе данных системы.

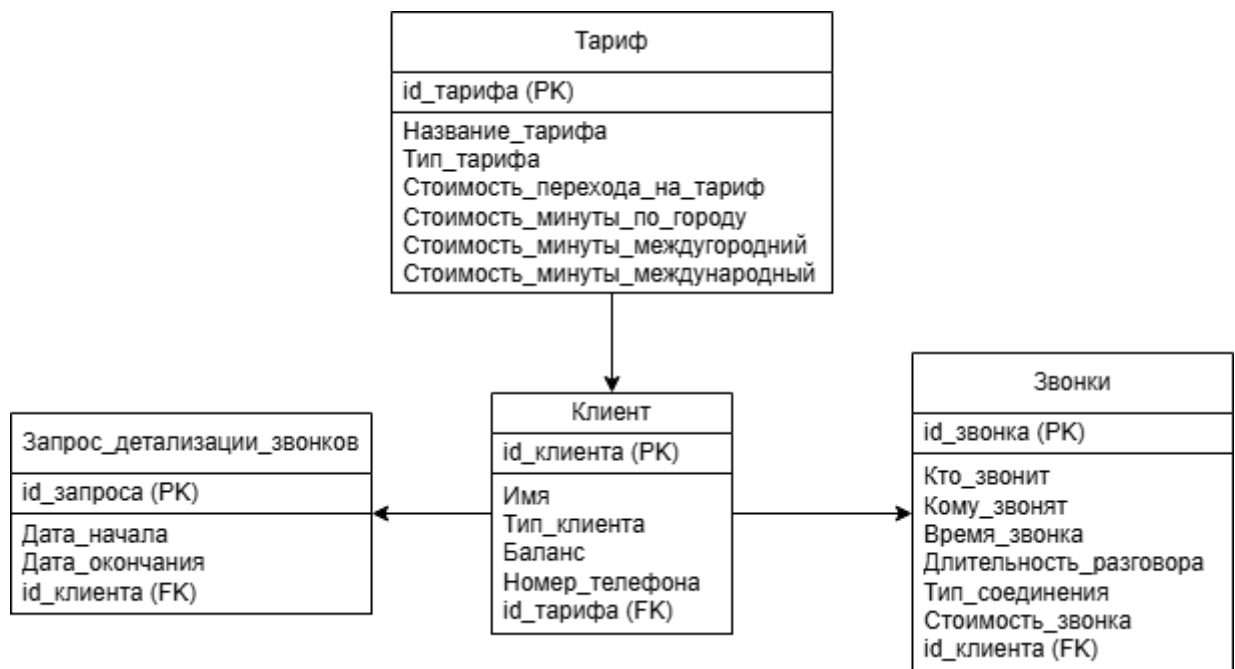


Рис. 2. Концептуальная модель базы данных

3) Создание физической модели базы данных (рис. 3)

Физическая модель включает в себя все необходимые детали для конкретной СУБД, обеспечивающие эффективное определение данных, такие как наименования столбцов, типы данных, описание первичных и внешних ключей и другие элементы.

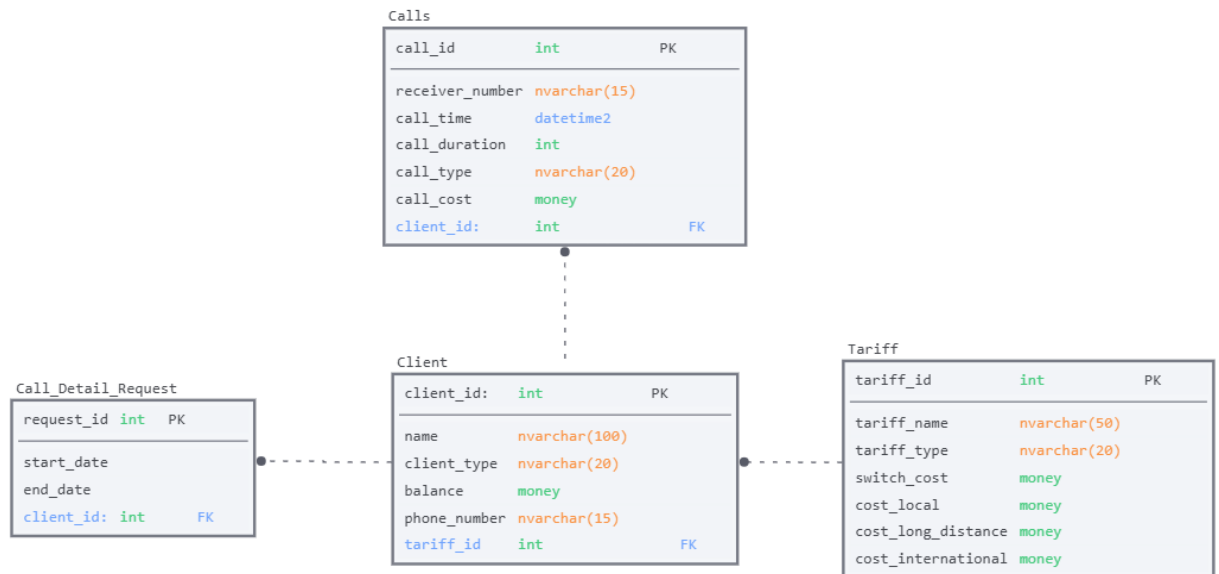


Рис.3. Физическая модель базы данных.

Создание БД в Microsoft SQL Server

Чтобы диаграмма была построена в Microsoft SQL Server, нужно создать БД и создать там таблицы, с необходимыми нам значениями. Будем использовать SQL-запросы.

Создаём БД с названием MTS (рис. 4).

```

-- Создание базы данных MTS
CREATE DATABASE MTS;
GO
  
```

Сообщения

Выполнение команд успешно завершено.

Рис. 4. Создание БД

Теперь перейдём к созданию необходимых нам таблиц с параметрами.

Таблица для Тарифов (рис. 5)

```

CREATE TABLE Tariff (
    tariff_id INT IDENTITY(1,1) PRIMARY KEY, -- Идентификатор тарифа
    tariff_name NVARCHAR(50) NOT NULL, -- Название тарифа
    tariff_type NVARCHAR(20) CHECK(tariff_type IN ('Корпоративный', 'Некорпоративный')), -- Тип тарифа
    switch_cost MONEY NOT NULL, -- Стоимость перехода на тариф
    cost_local MONEY NOT NULL, -- Стоимость минуты по городу
    cost_long_distance MONEY NOT NULL, -- Стоимость минуты междугородний
    cost_international MONEY NOT NULL -- Стоимость минуты международный
);
GO
  
```

Рис. 5. Структура таблицы Тарифов

Таблица для Клиентов (рис. 6)

```

CREATE TABLE Client (
    client_id INT IDENTITY(1,1) PRIMARY KEY,          -- Идентификатор клиента
    name NVARCHAR(100) NOT NULL,                      -- Имя клиента
    client_type NVARCHAR(20) CHECK(client_type IN ('Физическое', 'Юридическое')), -- Тип клиента
    balance MONEY NOT NULL DEFAULT 0,                 -- Баланс клиента (по умолчанию 0)
    phone_number NVARCHAR(15) UNIQUE NOT NULL,        -- Номер телефона клиента
    tariff_id INT,                                     -- Идентификатор тарифа клиента
    FOREIGN KEY (tariff_id) REFERENCES Tariff(tariff_id) -- Связь с тарифом
    ON DELETE SET NULL -- Если тариф удалён, ставим NULL в поле тариф клиента
);
GO

```

Рис. 6. Структура таблицы Клиентов

Таблица для Звонков (рис. 7)

```

CREATE TABLE Calls (
    call_id INT IDENTITY(1,1) PRIMARY KEY,            -- Идентификатор звонка
    client_id INT,                                    -- Идентификатор клиента
    receiver_number NVARCHAR(15) NOT NULL,            -- Номер абонента, которому звонили
    call_time DATETIME2 NOT NULL,                     -- Время звонка (используем DATETIME2)
    call_duration INT NOT NULL,                       -- Длительность звонка (в минутах)
    call_type NVARCHAR(20) CHECK(call_type IN ('По городу', 'Междугородный', 'Международный')), -- Тип соединения
    call_cost MONEY NOT NULL DEFAULT 0,               -- Стоимость звонка (по умолчанию 0)
    FOREIGN KEY (client_id) REFERENCES Client(client_id) -- Связь с клиентом
    ON DELETE CASCADE -- Если клиент удалён, все его звонки тоже удаляются
);
GO

```

Рис. 7. Структура таблицы Звонков

Таблица для Запросов детализации звонков (рис. 8)

```

-- Таблица для запросов детализации звонков
CREATE TABLE Call Detail Request (
    request_id INT IDENTITY(1,1) PRIMARY KEY,        -- Идентификатор запроса
    client_id INT,                                    -- Идентификатор клиента
    start_date DATE NOT NULL,                         -- Начальная дата для детализации
    end_date DATE NOT NULL,                           -- Конечная дата для детализации
    FOREIGN KEY (client_id) REFERENCES Client(client_id) -- Связь с клиентом
    ON DELETE CASCADE -- Если клиент удалён, запросы его детализации тоже удаляются
);
GO

```

Рис. 8. Структура таблицы Детализации звонков

Выполняем запрос, для БД MTS (рис. 9).

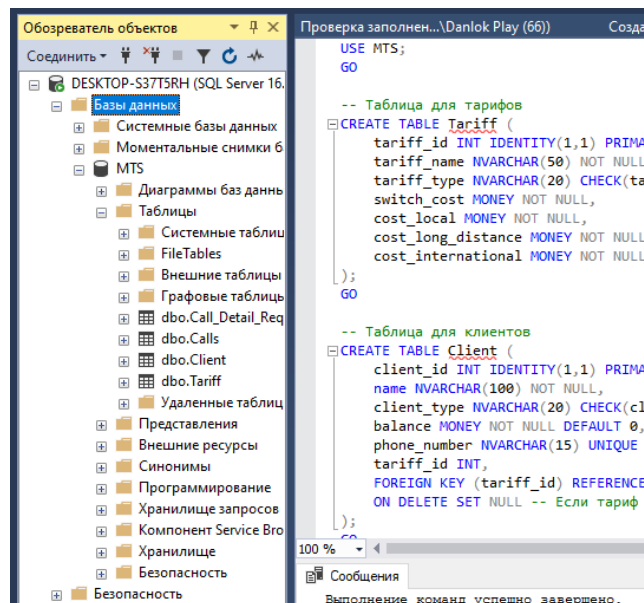


Рис. 9. Обзорщик объектов

Как видим, в обзорщике решений появилась БД с нашими таблицами.

Теперь можем сделать диаграмму в Microsoft SQL Server, и посмотреть, что получилось (рис. 10).

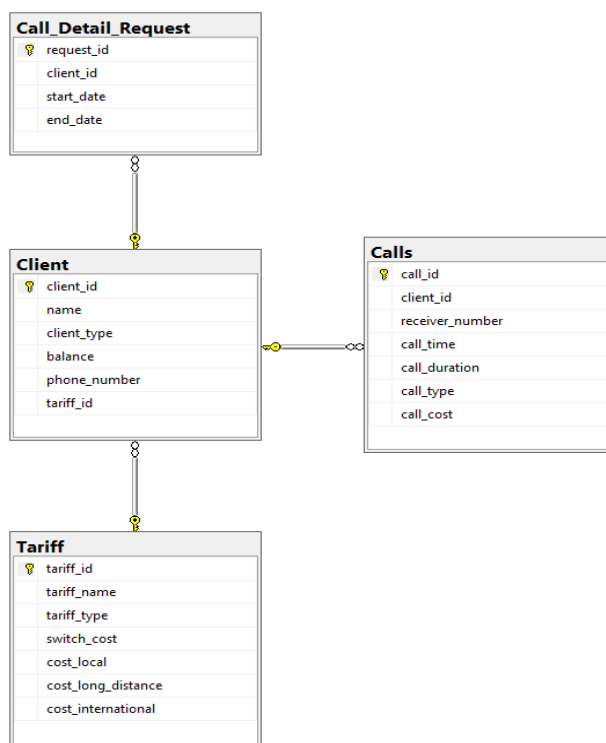


Рис. 10. Диаграмма в Microsoft SQL Server

Описание ограничений

При создании таблиц мы использовали следующие ограничения:

1) Значения по умолчанию. DEFAULT

У нас будет начальный баланс у клиента 0, и по умолчанию стоимость совершённого звонка, также будет 0 (рис. 11).

```

balance MONEY NOT NULL DEFAULT 0, -- Баланс клиента (по умолчанию 0)
call_cost MONEY NOT NULL DEFAULT 0, -- Стоимость звонка (по умолчанию 0)
  
```

Рис. 11 Значения по умолчанию.

2) Ограничения на вводимые данные. CHECK

У нас будут ограничения на вводимые данные для Типа тарифа ('Корпоративный', 'Некорпоративный'), Типа клиента ('Физическое', 'Юридическое') и Типа звонка ('По городу', 'Междугородний', 'Международный') (рис. 12.)

```

tariff_type NVARCHAR(20) CHECK(tariff_type IN ('Корпоративный', 'Некорпоративный')), -- Тип тарифа (Проверка ограничения на вводимые данные)
client_type NVARCHAR(20) CHECK(client_type IN ('Физическое', 'Юридическое')), -- Тип клиента (Проверка ограничения на вводимые данные)
call_type NVARCHAR(20) CHECK(call_type IN ('По городу', 'Междугородний', 'Международный')), -- Тип соединения (Проверка ограничения на вводимые данные)
  
```

Рис. 12. Ограничения на вводимые данные

3) Правила удаления. DELETE

```
ON DELETE CASCADE -- Если клиент удалён, все его звонки тоже удаляются  
ON DELETE CASCADE -- Если клиент удалён, запросы его детализации тоже удаляются
```

Рис. 13. ON DELETE CASCADE

Здесь в таблицах Calls и Call_Detail_Request используется ON DELETE CASCADE для связи с Client. Это значит: если удалить клиента из таблицы Client, то все его звонки (Calls) и запросы на детализацию (Call_Detail_Request) тоже автоматически удалятся.

```
ON DELETE SET NULL -- Если тариф удалён, ставим NULL в поле тариф клиента
```

Рис. 14. ON DELETE SET NULL

Здесь в таблице Client установлено правило ON DELETE SET NULL для tariff_id. Это значит: если удалить тариф из таблицы Tariff, у всех клиентов, которые были на этом тарифе, поле tariff_id станет NULL, но сами записи клиентов останутся в таблице.

Запросы на модификацию данных

Таблица 2. Команды на модификацию данных

| Команда | Назначение |
|---------|------------------------------|
| INSERT | Добавить новую запись |
| UPDATE | Изменить существующую запись |
| DELETE | Удалить запись |
| SELECT | Получить записи из таблицы |

1) INSERT

Добавление тарифа (рис. 15)

```
INSERT INTO Tariff (tariff_name, tariff_type, switch_cost, cost_local, cost_long_distance, cost_international)  
VALUES (N'МТС Бизнес', N'Корпоративный', 150.00, 1.50, 3.00, 7.50);
```

100 %

Сообщения

(затронута одна строка)

Рис. 15. Добавление тарифа

Добавление клиента (рис 16.)

```
INSERT INTO Client (name, client_type, balance, phone_number, tariff_id)  
VALUES (N'Иванов Иван Иванович', N'Физическое', 500.00, N'+79001234567', 1);
```

100 %

Сообщения

(затронута одна строка)

Рис. 16. Добавление клиента

Добавление звонка (рис. 17)

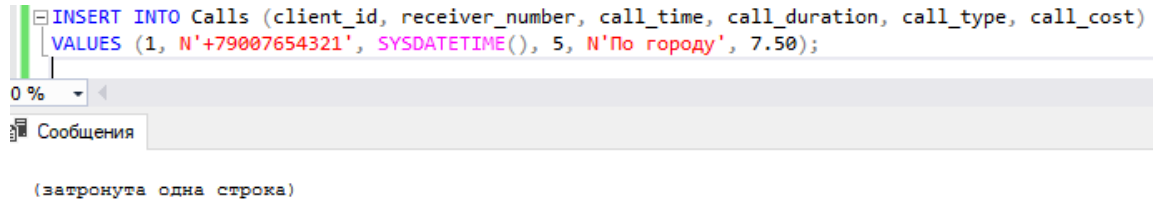


Рис. 17. Добавление звонка

Просмотрим теперь содержание таблиц.

Для этого выполним следующий запрос (рис. 18)

```
USE MTS;
GO

-- Проверка данных в таблице Tariff
SELECT * FROM Tariff;
GO

-- Проверка данных в таблице Client
SELECT * FROM Client;
GO

-- Проверка данных в таблице Calls
SELECT * FROM Calls;
GO

-- Проверка данных в таблице Call_Detail_Request
SELECT * FROM Call_Detail_Request;
GO
```

Рис. 18. Просмотр данных в таблицах

Результат (рис. 19).

| Результаты | | | | | | | |
|------------|-------------|---------------|-------------|------------|--------------------|--------------------|--|
| tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international | |
| 1 | МТС Бизнес | Корпоративный | 150,00 | 1,50 | 3,00 | 7,50 | |

| client_id | name | client_type | balance | phone_number | tariff_id |
|-----------|----------------------|-------------|---------|--------------|-----------|
| 1 | Иванов Иван Иванович | Физическое | 500,00 | +79001234567 | 1 |

| call_id | client_id | receiver_number | call_time | call_duration | call_type | call_cost |
|---------|-----------|-----------------|-----------------------------|---------------|-----------|-----------|
| 1 | 1 | +79007654321 | 2025-04-28 02:37:13.2403970 | 5 | По городу | 7,50 |

| request_id | client_id | start_date | end_date |
|------------|-----------|------------|----------|
|------------|-----------|------------|----------|

Рис. 19. Данные в таблицах.

2) UPDATE

Обновить баланс клиента. Этот запрос увеличит баланс клиента с номером +79001234567 на 200 рублей (рис. 20).

```
UPDATE Client
SET balance = balance + 200
WHERE phone_number = N'+79001234567';
```

Рис. 20. Обновление баланса клиента.

Проверим (рис. 21).

| | client_id | name | client_type | balance | phone_number | tariff_id |
|---|-----------|----------------------|-------------|---------|--------------|-----------|
| 1 | 1 | Иванов Иван Иванович | Физическое | 500,00 | +79001234567 | 1 |

| | client_id | name | client_type | balance | phone_number | tariff_id |
|---|-----------|----------------------|-------------|---------|--------------|-----------|
| 1 | 1 | Иванов Иван Иванович | Физическое | 700,00 | +79001234567 | 1 |

Рис. 21. Баланс до/после

Изменить стоимость перехода на тариф. Этот запрос изменяет стоимость перехода (switch_cost) для тарифа с tariff_id = 1 на 100 рублей (рис. 22).

```
UPDATE Tariff
SET switch_cost = 100.00
WHERE tariff_id = 1;
```

Рис. 22. Изменение стоимости перехода на тариф.

Проверка (рис. 23).

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|---|-----------|-------------|---------------|-------------|------------|--------------------|--------------------|
| 1 | 1 | МТС Бизнес | Корпоративный | 150,00 | 1,50 | 3,00 | 7,50 |

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|---|-----------|-------------|---------------|-------------|------------|--------------------|--------------------|
| 1 | 1 | МТС Бизнес | Корпоративный | 100,00 | 1,50 | 3,00 | 7,50 |

Рис. 23. Стоимость перехода до/после

Изменить тариф у клиента. Этот запрос меняет тариф клиента **только если** тип клиента и тип тарифа совпадают (рис. 24):

Физическое лицо → Некорпоративный тариф

Юридическое лицо → Корпоративный тариф

Если не подходит — тариф не изменится.

```

UPDATE Client
SET tariff_id = 4
FROM Client c
JOIN Tariff t ON t.tariff_id = 4
WHERE c.client_id = 1
AND (
    (c.client_type = N'Физическое' AND t.tariff_type = N'Некорпоративный')
    OR
    (c.client_type = N'Юридическое' AND t.tariff_type = N'Корпоративный')
);

```

Рис. 24. Изменение тарифа у клиента.

Проверка (рис. 25).

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|---|-----------|-------------|-----------------|-------------|------------|--------------------|--------------------|
| 1 | 1 | МТС Бизнес | Корпоративный | 100,00 | 1,50 | 3,00 | 7,50 |
| 2 | 3 | МТС Мой | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |
| 3 | 4 | МТС Один | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |

| | client_id | name | client_type | balance | phone_number | tariff_id |
|---|-----------|----------------------|-------------|---------|--------------|-----------|
| 1 | 1 | Иванов Иван Иванович | Физическое | 700,00 | +79001234567 | 3 |

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|---|-----------|-------------|-----------------|-------------|------------|--------------------|--------------------|
| 1 | 1 | МТС Бизнес | Корпоративный | 100,00 | 1,50 | 3,00 | 7,50 |
| 2 | 3 | МТС Мой | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |
| 3 | 4 | МТС Один | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |

| | client_id | name | client_type | balance | phone_number | tariff_id |
|---|-----------|----------------------|-------------|---------|--------------|-----------|
| 1 | 1 | Иванов Иван Иванович | Физическое | 700,00 | +79001234567 | 4 |

Рис. 25. Изменение тарифа до/после

3) DELETE

Удаление всех звонков у клиента (рис. 26).

```

DELETE FROM Calls
WHERE client_id = 1;

```

Рис. 26. Удаление звонков у клиента.

Удаление клиента по номеру телефона (рис. 27).

```

DELETE FROM Client
WHERE phone_number = N'+79001234567';

```

Рис. 27. Удаление клиента по номеру телефона.

Удаление тарифа (рис. 28).

```

DELETE FROM Tariff
WHERE tariff_id = 3;

```

Рис. 28. Удаление тарифа.

Таблицы до выполнения запросов (рис. 29).

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|---|-----------|-------------|-----------------|-------------|------------|--------------------|--------------------|
| 1 | 1 | МТС Бизнес | Корпоративный | 100,00 | 1,50 | 3,00 | 7,50 |
| 2 | 3 | МТС Мой | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |
| 3 | 4 | МТС Один | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |

| | client_id | name | client_type | balance | phone_number | tariff_id |
|---|-----------|----------------------|-------------|---------|--------------|-----------|
| 1 | 1 | Иванов Иван Иванович | Физическое | 700,00 | +79001234567 | 4 |

| | call_id | client_id | receiver_number | call_time | call_duration | call_type | call_cost |
|---|---------|-----------|-----------------|-----------------------------|---------------|-----------|-----------|
| 1 | 1 | 1 | +79007654321 | 2025-04-28 02:37:13.2403970 | 5 | По городу | 7,50 |

Рис. 29. Таблицы до выполнения запросов.

После выполнения этих запросов (рис. 30).

| Результаты | | Сообщения | | | | | |
|------------|-----------|-------------|-----------------|-------------|------------|--------------------|--------------------|
| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
| 1 | 1 | МТС Бизнес | Корпоративный | 100,00 | 1,50 | 3,00 | 7,50 |
| 2 | 4 | МТС Один | Некорпоративный | 150,00 | 1,50 | 3,00 | 7,50 |

| client_id | name | client_type | balance | phone_number | tariff_id |
|-----------|------|-------------|---------|--------------|-----------|
|-----------|------|-------------|---------|--------------|-----------|

| call_id | client_id | receiver_number | call_time | call_duration | call_type | call_cost |
|---------|-----------|-----------------|-----------|---------------|-----------|-----------|
|---------|-----------|-----------------|-----------|---------------|-----------|-----------|

Рис. 30. Таблицы после выполнения команд.

Всё сработало корректно.

Запросы на выборку данных. 10 запросов SELECT

Перед выполнением этих запросов заполним таблицы.

Заполнение Тарифов (рис. 31)

```

USE MTS;
GO

-- Тарифы
INSERT INTO Tariff (tariff_name, tariff_type, switch_cost, cost_local, cost_long_distance, cost_international)
VALUES
('MTS-Бизнес', 'Корпоративный', 360, 1.5, 2.5, 4.0),
('MTS-Про', 'Корпоративный', 310, 1.6, 2.6, 3.8),
('MTS-Гига', 'Корпоративный', 140, 2.5, 3.5, 5.0),
('MTS-Гига-Про', 'Корпоративный', 400, 1.3, 2.3, 3.9),
('MTS-Смарт', 'Некорпоративный', 250, 1.6, 2.6, 3.8),
('MTS-Экспресс', 'Корпоративный', 100, 2.2, 3.2, 4.7),
('MTS-Стандарт', 'Некорпоративный', 200, 1.9, 2.9, 4.2),
('MTS-Люкс', 'Некорпоративный', 250, 2.0, 3.0, 4.5),
('MTS-Оптим+', 'Некорпоративный', 280, 1.6, 2.6, 4.1),
('MTS-Мега', 'Корпоративный', 260, 1.4, 2.4, 3.9),
('MTS-Гибрид', 'Некорпоративный', 130, 2.3, 3.3, 4.7),
('MTS-Смарт-Плюс', 'Корпоративный', 160, 2.5, 3.5, 4.9),
('MTS-Базовый', 'Корпоративный', 130, 2.3, 3.3, 4.9),
('MTS-Плюс', 'Некорпоративный', 180, 1.9, 2.9, 4.4),
('MTS-Супер', 'Корпоративный', 260, 1.9, 2.9, 4.7),
('MTS-Лайт', 'Корпоративный', 110, 2.1, 3.1, 4.6),
('MTS-Максим', 'Корпоративный', 220, 2.7, 3.7, 5.2),
('MTS-Комфорт', 'Корпоративный', 120, 2.4, 3.4, 4.7),
('MTS-Эконом', 'Некорпоративный', 160, 1.7, 2.7, 4.0),
('MTS-Гига', 'Некорпоративный', 130, 2.4, 3.4, 4.9),
('MTS-Люкс', 'Корпоративный', 260, 2.1, 3.1, 4.6),
('MTS-Смарт', 'Корпоративный', 100, 2.2, 3.2, 4.6),
('MTS-Про', 'Некорпоративный', 300, 1.5, 2.5, 3.7),
('MTS-Модуль', 'Некорпоративный', 180, 2.1, 3.1, 4.9),
('MTS-Модуль', 'Корпоративный', 190, 2.2, 3.2, 5.0),
('MTS-Смарт', 'Некорпоративный', 90, 2.1, 3.1, 4.5),
('MTS-Базовый', 'Некорпоративный', 120, 2.2, 3.2, 4.8),
('MTS-Ультра', 'Корпоративный', 410, 1.3, 2.1, 3.3),
('MTS-Смарт-Плюс', 'Некорпоративный', 150, 2.4, 3.4, 4.8),
('MTS-Плюс', 'Корпоративный', 190, 2.0, 3.0, 4.5),
('MTS-Экспресс', 'Некорпоративный', 90, 2.1, 3.1, 4.6),
('MTS-Мега', 'Некорпоративный', 250, 1.3, 2.3, 3.8),
('MTS-Гига', 'Некорпоративный', 130, 2.4, 3.4, 4.9),
('MTS-Максимум', 'Корпоративный', 220, 2.7, 3.7, 5.2),
('MTS-Тарифице', 'Корпоративный', 190, 2.3, 3.3, 4.8),
('MTS-Оптим+', 'Некорпоративный', 280, 1.6, 2.6, 3.7),

```

Рис. 31. Заполнение тарифов

Теперь заполним Пользователей (рис. 32)

```

USE MTS;
GO

-- Тарифы
INSERT INTO Tariff (tariff_name, tariff_type, switch_cost, cost_local, cost_long_distance, cost_international)
VALUES
('MTS-Бизнес', 'Корпоративный', 360, 1.5, 2.5, 4.0),
('MTS-Про', 'Корпоративный', 310, 1.6, 2.6, 3.8),
('MTS-Гига', 'Корпоративный', 140, 2.5, 3.5, 5.0),
('MTS-Гига-Про', 'Корпоративный', 400, 1.3, 2.3, 3.9),
('MTS-Смарт', 'Некорпоративный', 250, 1.6, 2.6, 3.8),
('MTS-Экспресс', 'Корпоративный', 100, 2.2, 3.2, 4.7),
('MTS-Стандарт', 'Некорпоративный', 200, 1.9, 2.9, 4.2),
('MTS-Люкс', 'Некорпоративный', 250, 2.0, 3.0, 4.5),
('MTS-Оптимал', 'Некорпоративный', 280, 1.6, 2.6, 4.1),
('MTS-Мега', 'Корпоративный', 260, 1.4, 2.4, 3.9),
('MTS-Гибрид', 'Некорпоративный', 130, 2.3, 3.3, 4.7),
('MTS-Смарт-Плюс', 'Корпоративный', 160, 2.5, 3.5, 4.9),
('MTS-Базовый', 'Корпоративный', 130, 2.3, 3.3, 4.9),
('MTS-Плюс', 'Некорпоративный', 180, 1.9, 2.9, 4.4),
('MTS-Супер', 'Корпоративный', 260, 1.9, 2.9, 4.7),
('MTS-Лайт', 'Корпоративный', 110, 2.1, 3.1, 4.6),
('MTS-Максим', 'Корпоративный', 220, 2.7, 3.7, 5.2),
('MTS-Комфорт', 'Корпоративный', 120, 2.4, 3.4, 4.7),
('MTS-Эконом', 'Некорпоративный', 160, 1.7, 2.7, 4.0),
('MTS-Гига', 'Некорпоративный', 130, 2.4, 3.4, 4.9),
('MTS-Люкс', 'Корпоративный', 260, 2.1, 3.1, 4.6),
('MTS-Смарт', 'Корпоративный', 100, 2.2, 3.2, 4.6),
('MTS-Про', 'Некорпоративный', 300, 1.5, 2.5, 3.7),
('MTS-Модуль', 'Некорпоративный', 180, 2.1, 3.1, 4.9),
('MTS-Модуль', 'Корпоративный', 190, 2.2, 3.2, 5.0),
('MTS-Смарт', 'Некорпоративный', 90, 2.1, 3.1, 4.5),
('MTS-Базовый', 'Некорпоративный', 120, 2.2, 3.2, 4.8),
('MTS-Ультра', 'Корпоративный', 410, 1.3, 2.1, 3.3),
('MTS-Смарт-Плюс', 'Некорпоративный', 150, 2.4, 3.4, 4.8),
('MTS-Плюс', 'Корпоративный', 190, 2.0, 3.0, 4.5),
('MTS-Экспресс', 'Некорпоративный', 90, 2.1, 3.1, 4.6),
('MTS-Мега', 'Некорпоративный', 250, 1.3, 2.3, 3.8),
('MTS-Гига', 'Некорпоративный', 130, 2.4, 3.4, 4.9),
('MTS-Максимум', 'Корпоративный', 220, 2.7, 3.7, 5.2),
('MTS-Тарифище', 'Корпоративный', 190, 2.3, 3.3, 4.8),
('MTS-Оптимал', 'Некорпоративный', 280, 1.6, 2.6, 3.7),
('MTS-Смарт', 'Некорпоративный', 100, 2.2, 3.2, 4.6),
('MTS-Ультра', 'Некорпоративный', 400, 1.2, 2.0, 3.2),
('MTS-Лайт', 'Некорпоративный', 100, 2.0, 3.0, 4.5),
('MTS-Супер', 'Некорпоративный', 250, 1.8, 2.8, 4.6),
('MTS-Бизнес', 'Некорпоративный', 350, 1.4, 2.4, 3.9),
('MTS-Максимум', 'Некорпоративный', 210, 2.6, 3.6, 5.1),
('MTS-Тарифище', 'Некорпоративный', 180, 2.2, 3.2, 4.7),
('MTS-Базовый', 'Некорпоративный', 120, 2.2, 3.2, 4.8),
('MTS-Смарт', 'Корпоративный', 250, 1.6, 2.6, 3.8),
('MTS-Стандарт', 'Корпоративный', 210, 2.0, 3.0, 4.3),
('MTS-Смарт', 'Корпоративный', 100, 2.2, 3.2, 4.6),
('MTS-Мега', 'Некорпоративный', 250, 1.3, 2.3, 3.8),
('MTS-Комфорт', 'Некорпоративный', 110, 2.3, 3.3, 4.6),
('MTS-Гибрид', 'Некорпоративный', 130, 2.3, 3.3, 4.7),
('MTS-лайт', 'Корпоративный', 110, 2.1, 3.1, 4.6),
('MTS-Гига', 'Корпоративный', 140, 2.5, 3.5, 5.0),
('MTS-Эконом', 'Некорпоративный', 160, 1.7, 2.7, 4.0),
('MTS-Смарт', 'Некорпоративный', 90, 2.1, 3.1, 4.5),

```

Рис. 32. Заполнение пользователей

Заполнение Звонков (рис. 33).

```

USE MTS;
GO

-- Вставка данных в таблицу Calls (звонки)
DECLARE @j INT = 1;
DECLARE @ClientId INT;
DECLARE @TariffId INT;
DECLARE @CallType NVARCHAR(20);
DECLARE @RatePerMinute DECIMAL(10, 2);
DECLARE @CallDuration INT;
DECLARE @CallCost DECIMAL(10, 2);

WHILE @j <= 2337
BEGIN
    -- Генерация случайного client_id из существующих клиентов в таблице Client
    SELECT TOP 1 @ClientId = client_id FROM Client ORDER BY NEWID(); -- Случайный client_id

    -- Получаем тариф клиента
    SELECT @TariffId = tariff_id FROM Client WHERE client_id = @ClientId;

    -- Получаем тарифные данные для клиента из таблицы Tariff
    SELECT @RatePerMinute =
        CASE
            WHEN @CallType = 'По городу' THEN cost_local
            WHEN @CallType = 'Междугородний' THEN cost_long_distance
            ELSE cost_international
        END
    FROM Tariff
    WHERE tariff_id = @TariffId;

    -- Генерация случайного типа звонка
    SET @CallType =
        CASE
            WHEN RAND() < 0.33 THEN 'По городу'
            WHEN RAND() < 0.66 THEN 'Междугородний'
            ELSE 'Международный'
        END;

    -- Генерация длительности звонка (от 1 до 30 минут)
    SET @CallDuration = FLOOR(RAND() * 30) + 1;

    -- Рассчитываем стоимость звонка (стоимость за минуту * длительность звонка)
    SET @CallCost = @RatePerMinute * @CallDuration;

    -- Вставка данных о звонке
    INSERT INTO Calls (client_id, receiver_number, call_time, call_duration, call_type, call_cost)
    VALUES
        (@ClientId, -- Используем существующий client_id
        CONCAT('891612345', RIGHT('000' + CAST(FLOOR(RAND() * 1000) + 1 AS VARCHAR(3)), 3)),
        -- Генерация случайного времени звонка в диапазоне с 2024-09-01 по 2025-04-09
        DATEADD(MINUTE, FLOOR(RAND() * DATEDIFF(MINUTE, '2024-09-01 00:00:00', '2025-04-09 23:59:59')), '2024-09-01 00:00:00'),
        @CallDuration, -- Добавлена запятая
        @CallType,
        @CallCost); -- Стоимость звонка вычисляется на основе тарифа и длительности звонка

    SET @j = @j + 1;
END
GO

```

Рис. 33. Заполнение звонков

И заполним таблицу детализации звонков (рис. 34).

```

-- Вставка данных в таблицу Call_Detail_Request (Запросы детализации звонков)
DECLARE @k INT = 1; -- Используем новую переменную @k
DECLARE @ClientId INT;

WHILE @k <= 638
BEGIN
    -- Генерация случайного client_id из существующих клиентов в таблице Client
    SELECT TOP 1 @ClientId = client_id FROM Client ORDER BY NEWID(); -- Случайный client_id

    DECLARE @EndDate DATETIME;
    DECLARE @StartDate DATETIME;

    -- Генерация случайной конечной даты (в пределах до 09.04.2025)
    SET @EndDate = DATEADD(DAY, FLOOR(RAND() * DATEDIFF(DAY, '2025-03-01', '2025-04-09')), '2025-03-01');

    -- Генерация случайной начальной даты, которая не будет позднее конечной
    SET @StartDate = DATEADD(DAY, FLOOR(RAND() * DATEDIFF(DAY, '2024-09-01', @EndDate)), '2024-09-01');

    -- Вставка данных
    INSERT INTO Call_Detail_Request (client_id, start_date, end_date)
    VALUES
        (@ClientId, -- Используем существующий client_id
        @StartDate, -- Начальная дата
        @EndDate); -- Конечная дата

    SET @k = @k + 1;
END
GO

```

Рис. 34. Заполнение детализации звонков

Выведем результат, что получилось (рис. 35).

Результаты

Сообщения

| | | | | | | | |
|----|----|--------------|-----------------|--------|------|------|------|
| 59 | 59 | MTS-Стандарт | Некорпоративный | 200,00 | 1,90 | 2,90 | 4,20 |
| 60 | 60 | MTS-Про | Корпоративный | 310,00 | 1,60 | 2,60 | 3,80 |
| 61 | 61 | MTS-Гига-Про | Корпоративный | 400,00 | 1,30 | 2,30 | 3,90 |
| 62 | 62 | MTS-Люкс | Корпоративный | 260,00 | 2,10 | 3,10 | 4,60 |

| | | | | | | |
|-----|-----------|-------------------------------|--------------|---------|--------------|-----------|
| | client_id | name | client_type | balance | phone_number | tariff_id |
| 99 | 99 | АО "НейроСистемы" | Юридическ... | 1598,65 | 8916092531 | 13 |
| 100 | 100 | ООО "Системные Решения" | Юридическ... | 1683,55 | 8916096075 | 10 |
| 101 | 101 | ООО "ТехСистемы" | Юридическ... | 1355,45 | 8916093237 | 34 |
| 102 | 102 | ООО "ПрогрессГрупп" | Юридическ... | 1690,95 | 8916025904 | 16 |
| 103 | 103 | ИП "Новиков Александр Вале... | Юридическ... | 4176,78 | 8916041639 | 10 |
| 104 | 104 | АО "КосмоГрупп" | Юридическ... | 4057,48 | 8916095324 | 2 |
| 105 | 105 | АО "ТехноСтрим" | Юридическ... | 2462,98 | 8916014542 | 46 |

| | | | | | | | |
|------|---------|-----------|-----------------|-----------------------------|---------------|---------------|-----------|
| | call_id | client_id | receiver_number | call_time | call_duration | call_type | call_cost |
| 2329 | 2329 | 10 | 891612345265 | 2024-07-21 10:55:00.0000000 | 18 | Междугородний | 59,40 |
| 2330 | 2330 | 52 | 891612345237 | 2024-03-11 20:28:00.0000000 | 12 | По городу | 37,20 |
| 2331 | 2331 | 42 | 891612345201 | 2024-05-18 11:03:00.0000000 | 5 | По городу | 8,50 |
| 2332 | 2332 | 60 | 891612345749 | 2024-07-16 08:24:00.0000000 | 14 | Междугородний | 28,00 |
| 2333 | 2333 | 42 | 891612345616 | 2024-06-30 11:46:00.0000000 | 11 | Междугородний | 29,70 |

| | | | | |
|-----|------------|-----------|------------|------------|
| | request_id | client_id | start_date | end_date |
| 633 | 633 | 88 | 2024-04-21 | 2025-02-02 |
| 634 | 634 | 17 | 2024-03-21 | 2025-01-25 |
| 635 | 635 | 52 | 2024-03-08 | 2025-01-18 |
| 636 | 636 | 96 | 2024-04-16 | 2025-02-09 |
| 637 | 637 | 66 | 2024-03-05 | 2025-01-30 |
| 638 | 638 | 83 | 2024-04-22 | 2025-01-17 |

Рис. 35. Заполненные таблицы.

Теперь перейдём к 10 запросам SELECT.

1) WHERE

```
USE MTS;
GO
SELECT * FROM Client
WHERE balance > 1500;
```

100 %

Результаты Сообщения

| | client_id | name | client_type | balance | phone_number | tariff_id |
|---|-----------|------------------------------------|-------------|---------|--------------|-----------|
| 1 | 68 | АО "Технопарк" | Юридическое | 4268,02 | 8916092315 | 47 |
| 2 | 70 | ООО "Мегасистемы" | Юридическое | 4660,72 | 8916099690 | 22 |
| 3 | 71 | ИП "Тимофеев Михаил Александрович" | Юридическое | 3032,00 | 8916092477 | 30 |
| 4 | 72 | ИП "Смирнов Владимир Иванович" | Юридическое | 1976,33 | 8916046585 | 17 |
| 5 | 73 | ООО "ИнтерТех" | Юридическое | 4278,76 | 8916000887 | 22 |
| 6 | 74 | АО "Глобус-Тех" | Юридическое | 4719,27 | 8916039835 | 28 |
| 7 | 75 | ИП "Ковалев Алексей Владимирович" | Юридическое | 3753,44 | 8916003294 | 25 |

Рис. 36. WHERE

Запрос выводит список клиентов, у которых баланс превышает 1500. Условие WHERE используется для фильтрации данных, чтобы выбрать только те записи, которые соответствуют указанному критерию.

2) ORDER BY

```
USE MTS;
GO
SELECT * FROM Tariff
ORDER BY switch_cost ASC;
```

100 %

Результаты Сообщения

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|----|-----------|--------------|-----------------|-------------|------------|--------------------|--------------------|
| 1 | 26 | MTS-Смарт | Некорпоративный | 90,00 | 2,10 | 3,10 | 4,50 |
| 2 | 31 | MTS-Экспресс | Некорпоративный | 90,00 | 2,10 | 3,10 | 4,60 |
| 3 | 54 | MTS-Смарт | Некорпоративный | 90,00 | 2,10 | 3,10 | 4,50 |
| 4 | 37 | MTS-Смарт | Некорпоративный | 100,00 | 2,20 | 3,20 | 4,60 |
| 5 | 39 | MTS-Лайт | Некорпоративный | 100,00 | 2,00 | 3,00 | 4,50 |
| 6 | 47 | MTS-Смарт | Корпоративный | 100,00 | 2,20 | 3,20 | 4,60 |
| 7 | 22 | MTS-Смарт | Корпоративный | 100,00 | 2,20 | 3,20 | 4,60 |
| 8 | 6 | MTS-Экспресс | Корпоративный | 100,00 | 2,20 | 3,20 | 4,70 |
| 9 | 16 | MTS-Лайт | Корпоративный | 110,00 | 2,10 | 3,10 | 4,60 |
| 10 | 49 | MTS-Комфорт | Некорпоративный | 110,00 | 2,30 | 3,30 | 4,60 |
| 11 | 51 | MTS-Лайт | Корпоративный | 110,00 | 2,10 | 3,10 | 4,60 |

Рис. 37. ORDER BY

Запрос выводит все данные из таблицы Tariff, сортируя результаты по стоимости перехода на тариф (switch_cost) от наименьшей к наибольшей.

3) Функция агрегации

```
USE MTS;
GO
SELECT SUM(call_cost) AS total_revenue FROM Calls;
```

100 %

Результаты Сообщения

| | total_revenue |
|---|---------------|
| 1 | 106620.80 |

Рис. 38. Функция агрегации

Запрос подсчитывает общую сумму всех звонков (их стоимости) в таблице Calls и выводит её как total_revenue (общий доход).

4) GROUP BY + HAVING

```
USE MTS;
GO
SELECT call_type, COUNT(*) AS call_count
FROM Calls
GROUP BY call_type
HAVING COUNT(*) >= 750;
```

100 %

Результаты Сообщения

| | call_type | call_count |
|---|---------------|------------|
| 1 | Междугородний | 1016 |
| 2 | По городу | 756 |

Рис. 39. GROUP BY + HAVING

Запрос выводит количество звонков для каждого типа звонка, но только для тех типов, где количество звонков больше или равно 750.

5) CAST

```
USE MTS;
GO
SELECT name, CAST(balance AS INT) AS balance_int
FROM Client;
```

| | name | balance_int |
|---|------------------------------|-------------|
| 1 | Иванов Иван Иванович | 802 |
| 2 | Петров Петр Петрович | 560 |
| 3 | Сидоров Сергей Сергеевич | 866 |
| 4 | Кузнецов Николай Николаевич | 622 |
| 5 | Васильев Владимир Викторович | 910 |
| 6 | Морозов Михаил Михайлович | 505 |
| 7 | Попов Алексей Александрович | 507 |
| 8 | Александров Андрей Андреевич | 236 |
| 9 | Михайлов Денис Денисович | 505 |

Рис. 40. CAST

Запрос выводит имена клиентов и их баланс, при этом баланс преобразуется в целое число (отбрасываются дробные значения, если они есть).

6) CASE

```
USE MTS;
GO
SELECT call_id, call_cost,
CASE
WHEN call_cost < 20 THEN 'Дешевый'
WHEN call_cost BETWEEN 20 AND 45 THEN 'Средний'
ELSE 'Дорогой'
END AS call_category
FROM Calls;
```

| | call_id | call_cost | call_category |
|----|---------|-----------|---------------|
| 1 | 1 | 78,20 | Дорогой |
| 2 | 2 | 11,20 | Дешевый |
| 3 | 3 | 64,00 | Дорогой |
| 4 | 4 | 96,00 | Дорогой |
| 5 | 5 | 15,60 | Дешевый |
| 6 | 6 | 10,80 | Дешевый |
| 7 | 7 | 13,60 | Дешевый |
| 8 | 8 | 59,40 | Дорогой |
| 9 | 9 | 37,80 | Средний |
| 10 | 10 | 16,90 | Дешевый |
| 11 | 11 | 34,50 | Средний |
| 12 | 12 | 9,60 | Дешевый |

Рис. 41. CASE

Запрос выводит идентификатор звонка, стоимость звонка и категорию звонка (Дешевый, Средний или Дорогой) в зависимости от стоимости звонка.

7) EXISTS

```

USE MTS;
GO
SELECT t.tariff_id, t.tariff_name
FROM Tariff t
WHERE EXISTS (
    SELECT 1 FROM Client c WHERE c.tariff_id = t.tariff_id
);

```

| | tariff_id | tariff_name |
|----|-----------|--------------|
| 1 | 1 | MTS-Бизнес |
| 2 | 2 | MTS-Про |
| 3 | 4 | MTS-Гига-Про |
| 4 | 5 | MTS-Смарт |
| 5 | 7 | MTS-Стандарт |
| 6 | 8 | MTS-Люкс |
| 7 | 9 | MTS-Оптимал |
| 8 | 10 | MTS-Мега |
| 9 | 11 | MTS-Гибрид |
| 10 | 13 | MTS-Базовый |
| 11 | 14 | MTS-Плюс |
| 12 | 15 | MTS-Супер |
| 13 | 16 | MTS-Лайт |
| 14 | 17 | MTS-Максим |
| 15 | 18 | MTS-Комфорт |
| 16 | 19 | MTS-Эконом |
| 17 | 20 | MTS-Гига |

Рис. 42. EXISTS

Запрос выводит тарифы, которые используются хотя бы одним клиентом. Если у тарифа есть хотя бы один клиент, он будет включён в результат.

8) Подзапрос

```

USE MTS;
GO
SELECT * FROM Client
WHERE balance > (SELECT AVG(balance) FROM Client);

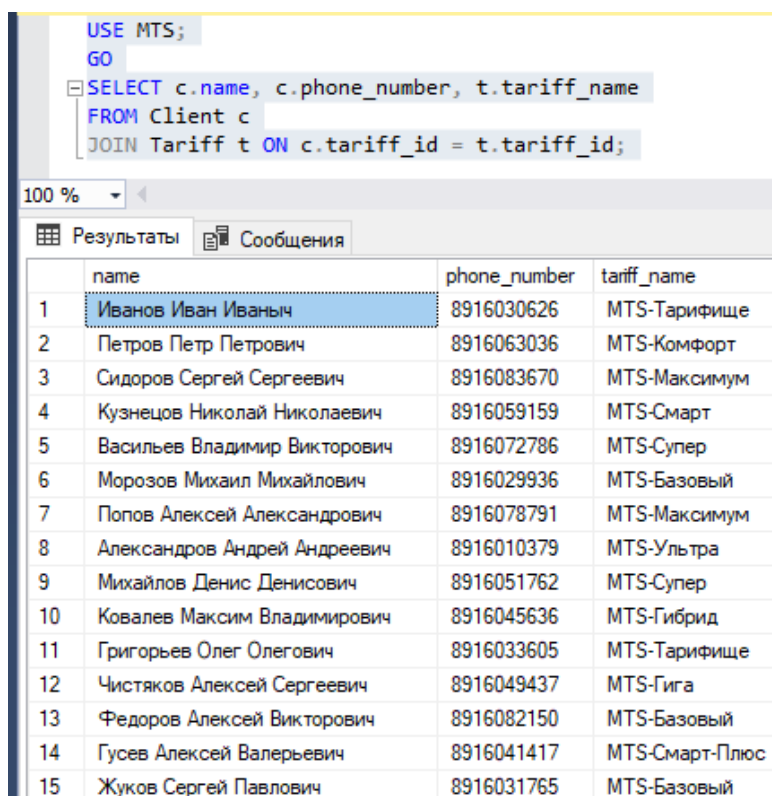
```

| | client_id | name | client_type | balance | phone_number | tariff_id |
|----|-----------|------------------------------------|-------------|---------|--------------|-----------|
| 1 | 68 | АО "Технопарк" | Юридическое | 4268,02 | 8916092315 | 47 |
| 2 | 70 | ООО "Мегасистемы" | Юридическое | 4660,72 | 8916099690 | 22 |
| 3 | 71 | ИП "Тимофеев Михаил Александрович" | Юридическое | 3032,00 | 8916092477 | 30 |
| 4 | 72 | ИП "Смирнов Владимир Иванович" | Юридическое | 1976,33 | 8916046585 | 17 |
| 5 | 73 | ООО "ИнтерТех" | Юридическое | 4278,76 | 8916000887 | 22 |
| 6 | 74 | АО "Глобус-Тех" | Юридическое | 4719,27 | 8916039835 | 28 |
| 7 | 75 | ИП "Ковалев Алексей Владимирович" | Юридическое | 3753,44 | 8916003294 | 25 |
| 8 | 76 | ООО "ПроектСистемы" | Юридическое | 4533,09 | 8916000912 | 4 |
| 9 | 78 | ООО "ЭкспертГрупп" | Юридическое | 4551,14 | 8916079899 | 35 |
| 10 | 79 | АО "ПроГрупп" | Юридическое | 2091,45 | 8916069016 | 4 |
| 11 | 80 | ООО "ВекторТех" | Юридическое | 3229,08 | 8916088315 | 10 |
| 12 | 81 | ООО "Вектор-Проект" | Юридическое | 4429,93 | 8916067529 | 1 |

Рис. 43. Подзапрос

Запрос выводит всех клиентов, чьи балансы больше, чем средний баланс всех клиентов в базе данных.

9) JOIN



```
USE MTS;  
GO  
SELECT c.name, c.phone_number, t.tariff_name  
FROM Client c  
JOIN Tariff t ON c.tariff_id = t.tariff_id;
```

100 %

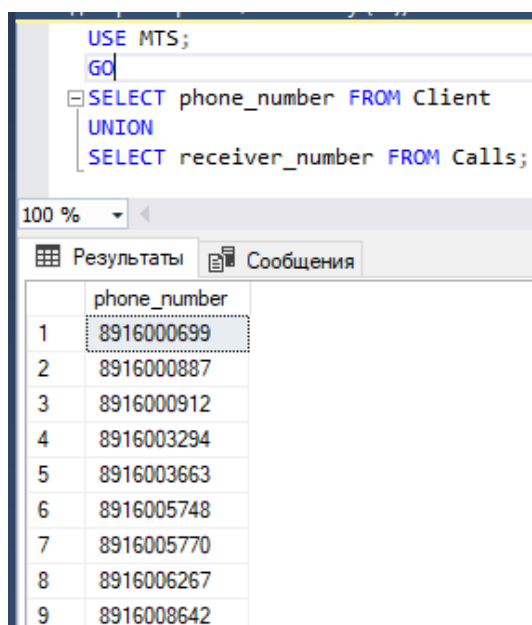
Результаты Сообщения

| | name | phone_number | tariff_name |
|----|------------------------------|--------------|----------------|
| 1 | Иванов Иван Иванович | 8916030626 | MTS-Тарифище |
| 2 | Петров Петр Петрович | 8916063036 | MTS-Комфорт |
| 3 | Сидоров Сергей Сергеевич | 8916083670 | MTS-Максимум |
| 4 | Кузнецов Николай Николаевич | 8916059159 | MTS-Смарт |
| 5 | Васильев Владимир Викторович | 8916072786 | MTS-Супер |
| 6 | Морозов Михаил Михайлович | 8916029936 | MTS-Базовый |
| 7 | Попов Алексей Александрович | 8916078791 | MTS-Максимум |
| 8 | Александров Андрей Андреевич | 8916010379 | MTS-Ультра |
| 9 | Михайлов Денис Денисович | 8916051762 | MTS-Супер |
| 10 | Ковалев Максим Владимирович | 8916045636 | MTS-Гибрид |
| 11 | Григорьев Олег Олегович | 8916033605 | MTS-Тарифище |
| 12 | Чистяков Алексей Сергеевич | 8916049437 | MTS-Гига |
| 13 | Федоров Алексей Викторович | 8916082150 | MTS-Базовый |
| 14 | Гусев Алексей Валерьевич | 8916041417 | MTS-Смарт-Плюс |
| 15 | Жуков Сергей Павлович | 8916031765 | MTS-Базовый |

Рис. 44. JOIN

Запрос выводит имена клиентов, их номера телефонов и название их тарифов. Для этого объединяются данные из таблиц Client и Tariff по полю tariff_id.

10) UNION



```
USE MTS;  
GO  
SELECT phone_number FROM Client  
UNION  
SELECT receiver_number FROM Calls;
```

100 %

Результаты Сообщения

| | phone_number |
|---|--------------|
| 1 | 8916000699 |
| 2 | 8916000887 |
| 3 | 8916000912 |
| 4 | 8916003294 |
| 5 | 8916003663 |
| 6 | 8916005748 |
| 7 | 8916005770 |
| 8 | 8916006267 |
| 9 | 8916008642 |

Рис. 45. UNION

Запрос выводит уникальные номера телефонов, которые присутствуют либо среди клиентов (phone_number из Client), либо среди получателей звонков (receiver_number из Calls).

Хранимые процедуры

1) Хранимая процедура GetMostActiveClients позволяет вывести информацию о самых активных клиентах за определённый период. Для этого нужно передать дату начала и окончания периода, а также количество клиентов, которых нужно вывести в результатах. Процедура возвращает топ клиентов, совершивших наибольшее количество звонков, и сортирует их по убыванию активности.

```
EXEC GetMostActiveClients @start_date = '2024-01-01', @end_date = '2024-08-01', @top_n = 5;
```

Рис. 46. Запрос к процедуре 1.

Этот запрос вернёт топ-5 самых активных клиентов, которые совершили наибольшее количество звонков в период с 1 января 2024 года по 1 августа 2024 года.

```
USE MTS;
GO

IF OBJECT_ID('GetMostActiveClients', 'P') IS NOT NULL
    DROP PROCEDURE GetMostActiveClients;
GO

CREATE PROCEDURE GetMostActiveClients
    @start_date DATETIME2,
    @end_date DATETIME2,
    @top_n INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT TOP(@top_n)
        c.client_id,
        c.name,
        c.phone_number,
        COUNT(cl.call_id) AS total_calls
    FROM Client c
    LEFT JOIN Calls cl ON c.client_id = cl.client_id
    WHERE cl.call_time BETWEEN @start_date AND @end_date
    GROUP BY c.client_id, c.name, c.phone_number
    ORDER BY total_calls DESC;
END;
GO
```

Рис. 47. Процедура 1.

Результат:

| | client_id | name | phone_number | total_calls |
|---|-----------|-----------------------------|--------------|-------------|
| 1 | 16 | Крылов Игорь Николаевич | 8916011756 | 33 |
| 2 | 29 | Новиков Дмитрий Михайлович | 8916015485 | 31 |
| 3 | 33 | Смирнов Борис Евгеньевич | 8916006267 | 30 |
| 4 | 34 | Тимошенко Михаил Викторович | 8916008642 | 30 |
| 5 | 94 | ООО "НейроГрупп" | 8916086174 | 29 |

Рис. 48. Результат выполнения 1-ой процедуры

2) Процедура `GetClientCallCostAnalysis` анализирует стоимость звонков клиентов за определённый период, который задаётся двумя параметрами — начальной и конечной датой. Она собирает информацию о стоимости звонков каждого клиента, разделяя их на городские, междугородние и международные. Для каждого клиента рассчитывается общая стоимость всех его звонков в указанный период. Результаты выводятся в виде списка с именем клиента, его номером телефона и детализацией стоимости звонков по категориям, с указанием общей стоимости. Данные сортируются по общей стоимости звонков в порядке убывания, то есть сначала идут клиенты с наибольшими расходами.

```
USE MTS;
GO

IF OBJECT_ID('GetClientCallCostAnalysis', 'P') IS NOT NULL
    DROP PROCEDURE GetClientCallCostAnalysis;
GO

CREATE PROCEDURE GetClientCallCostAnalysis
    @start_date DATE,          -- Начальная дата для анализа
    @end_date DATE             -- Конечная дата для анализа
AS
BEGIN
    SET NOCOUNT ON;

    -- Основной запрос для получения информации о стоимости звонков по клиентам
    SELECT
        c.client_id,
        c.name AS client_name,
        c.phone_number,
        SUM(CASE
            WHEN ca.call_type = 'По городу' THEN ca.call_cost
            ELSE 0
        END) AS local_calls_cost, -- Стоимость городских звонков
        SUM(CASE
            WHEN ca.call_type = 'Междугородний' THEN ca.call_cost
            ELSE 0
        END) AS long_distance_calls_cost, -- Стоимость междугородних звонков
        SUM(CASE
            WHEN ca.call_type = 'Международный' THEN ca.call_cost
            ELSE 0
        END) AS international_calls_cost, -- Стоимость международных звонков
        SUM(ca.call_cost) AS total_calls_cost -- Общая стоимость всех звонков
    FROM Client c
    JOIN Calls ca ON c.client_id = ca.client_id
    WHERE ca.call_time BETWEEN @start_date AND @end_date
    GROUP BY c.client_id, c.name, c.phone_number
    ORDER BY total_calls_cost DESC;
END;
GO
```

Рис. 49. Процедура 2.

EXEC GetClientCallCostAnalysis @start_date = '2024-01-01', @end_date = '2024-10-30';

| | client_id | client_name | phone_number | local_calls_cost | long_distance_calls_cost | international_calls_cost | total_calls_cost |
|----|-----------|--------------------------------|--------------|------------------|--------------------------|--------------------------|------------------|
| 1 | 52 | Мельников Сергей Александрович | 8916019556 | 545.00 | 799.30 | 476.00 | 1820.30 |
| 2 | 16 | Крылов Игорь Николаевич | 8916011756 | 472.10 | 702.90 | 508.90 | 1683.90 |
| 3 | 70 | ООО "Мегасистемы" | 8916099690 | 546.80 | 586.40 | 464.20 | 1597.40 |
| 4 | 31 | Павлов Валерий Викторович | 8916039575 | 270.50 | 661.40 | 637.70 | 1569.60 |
| 5 | 29 | Новиков Дмитрий Михайлович | 8916015485 | 435.10 | 784.30 | 344.50 | 1563.90 |
| 6 | 46 | Егорова Светлана Петровна | 8916029286 | 560.60 | 520.60 | 430.00 | 1511.20 |
| 7 | 94 | ООО "НейроГрупп" | 8916086174 | 545.30 | 281.00 | 669.70 | 1496.00 |
| 8 | 54 | Тимофеев Павел Анатольевич | 8916005748 | 404.60 | 419.00 | 669.70 | 1493.30 |
| 9 | 57 | Петрова Ирина Викторовна | 8916018057 | 482.90 | 691.70 | 271.50 | 1446.10 |
| 10 | 65 | Степанов Сергей Анатольевич | 8916011228 | 707.60 | 582.30 | 152.80 | 1442.70 |
| 11 | 33 | Смирнов Борис Евгеньевич | 8916006267 | 280.20 | 884.80 | 271.70 | 1436.70 |

Рис. 50. Результат запроса к процедуре 2.

Триггеры

Теперь создадим 2 триггера.

Триггер 1. **trg_PreventTariffChangeWithNegativeBalance.**

Этот триггер срабатывает после обновления записи в таблице Client. Его задача — запретить изменение тарифа (tariff_id), если у клиента отрицательный баланс.

Как работает: Если при обновлении обнаружено, что тариф изменён (tariff_id до и после разные), и при этом значение поля balance меньше нуля — происходит откат транзакции и вывод сообщения об ошибке: 'Нельзя сменить тариф при отрицательном балансе.'

Таким образом, триггер обеспечивает контроль целостности логики: клиент с отрицательным балансом не может перейти на другой тариф.

```
CREATE TRIGGER trg_PreventTariffChangeWithNegativeBalance
ON Client
AFTER UPDATE
AS
BEGIN
    -- Проверка: была ли попытка смены тарифа при отрицательном балансе
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN deleted d ON i.client_id = d.client_id
        WHERE i.tariff_id <> d.tariff_id AND i.balance < 0
    )
    BEGIN
        RAISERROR('Нельзя сменить тариф при отрицательном балансе.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

%

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-20T10:50:47.5273951+03:00

Рис. 51. Триггер 1

Триггер 2. **trg_PreventCallWithNegativeBalance.**

Этот триггер срабатывает после вставки записи в таблицу Calls. Он блокирует добавление звонка, если у клиента, который совершает звонок, отрицательный баланс.

Как работает: Триггер проверяет каждую вставляемую строку. Если у соответствующего клиента (client_id) баланс меньше нуля, выполнение вставки отменяется с помощью ROLLBACK, и выводится сообщение об ошибке: 'Нельзя совершить звонок при отрицательном балансе.'

Этот механизм предотвращает возможность совершения звонков при нехватке средств, что обеспечивает реалистичное поведение системы биллинга.

```
CREATE TRIGGER trg_PreventCallWithNegativeBalance
ON Calls
AFTER INSERT
AS
BEGIN
    -- Проверяем, есть ли среди вставленных звонков такие, где у клиента отрицательный баланс
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN Client c ON i.client_id = c.client_id
        WHERE c.balance < 0
    )
    BEGIN
        RAISERROR('Нельзя совершить звонок при отрицательном балансе.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-20T10:51:46.5551142+03:00

Рис. 52. Триггер 2

Проверим триггер. У нас у клиента с id 21 – отрицательный баланс (рис. 50).

| | tariff_id | tariff_name | tariff_type | switch_cost | cost_local | cost_long_distance | cost_international |
|---|-----------|--------------|-----------------|-------------|------------|--------------------|--------------------|
| 1 | 1 | MTS-Бизнес | Корпоративный | 360,00 | 1,50 | 2,50 | 4,00 |
| 2 | 2 | MTS-Про | Корпоративный | 310,00 | 1,60 | 2,60 | 3,80 |
| 3 | 3 | MTS-Гига | Корпоративный | 140,00 | 2,50 | 3,50 | 5,00 |
| 4 | 4 | MTS-Гига-Про | Корпоративный | 400,00 | 1,30 | 2,30 | 3,90 |
| 5 | 5 | MTS-Смарт | Некорпоративный | 250,00 | 1,60 | 2,60 | 3,80 |
| 6 | 6 | MTS-Экспресс | Корпоративный | 100,00 | 2,20 | 3,20 | 4,70 |
| 7 | 7 | MTS-Стандарт | Некорпоративный | 200,00 | 1,90 | 2,90 | 4,20 |
| 8 | 8 | MTS-Плюс | Некорпоративный | 250,00 | 2,00 | 3,00 | 4,50 |

| | client_id | name | client_type | balance | phone_number | tariff_id |
|----|-----------|--------------------------------|-------------|---------|--------------|-----------|
| 17 | 17 | Тимофеев Василий Григорьевич | Физическое | 594,90 | 8916020211 | 58 |
| 18 | 18 | Барсуков Александр Вячеслав... | Физическое | 189,09 | 8916029585 | 26 |
| 19 | 19 | Шмидт Михаил Вячеславович | Физическое | 789,07 | 8916010033 | 54 |
| 20 | 20 | Голубев Николай Сергеевич | Физическое | 711,62 | 8916048999 | 19 |
| 21 | 21 | Гончаров Алексей Артемович | Физическое | -226,63 | 8916021039 | 9 |
| 22 | 22 | Дмитриев Александр Викторо... | Физическое | 821,46 | 8916005770 | 59 |
| 23 | 23 | Емельянов Кирилл Игоревич | Физическое | 164,14 | 8916014480 | 48 |
| 24 | 24 | Зайцев Павел Сергеевич | Физическое | 509,84 | 8916087707 | 27 |

Рис. 53. Клиент 21

Выполним запрос на изменение тарифа (рис. 51).

```
USE MTS;
GO
UPDATE Client
SET tariff_id = 5
FROM client c
JOIN Tariff t ON t.tariff_id = 5
WHERE c.client_id = 21
AND (
    (c.client_type = N'Физическое' AND t.tariff_type = N'Некорпоративный')
    OR
    (c.client_type = N'Юридическое' AND t.tariff_type = N'Корпоративный')
);
```

Сообщения

сообщение: 50000, уровень: 16, состояние: 1, процедура: trg_PreventTariffChangeWithNegativeBalance, строка: 14 [строка начала пакета: 2]
 Нельзя сменить тариф при отрицательном балансе.
 Сообщение 3609, уровень: 16, состояние: 1, строка: 3
 Транзакция завершилась в триггере. Выполнение пакета прервано.

Рис. 54. Проверка триггера

Как видим, триггер сработал.

Приложение

В Microsoft Visual Studio создадим Приложение Windows Forms (.Net Framework). Назовём проект MTS и запустим проект (рис. 55).

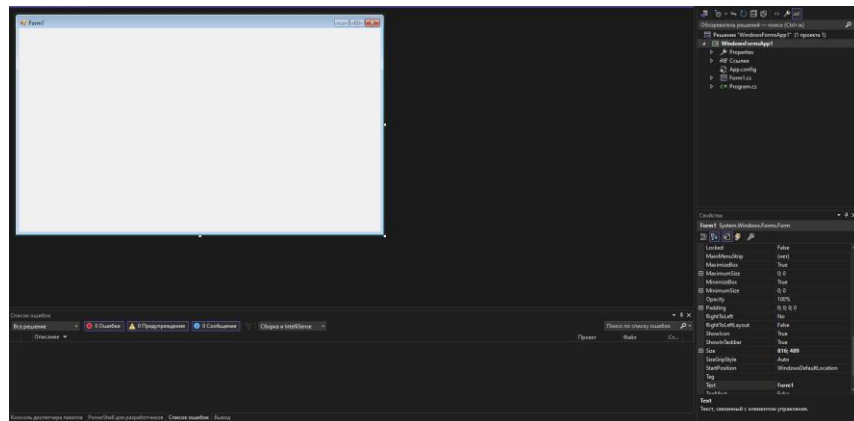


Рис. 55.

Теперь панель элементов добавим 4 DataGridView и 1 Button, при нажатии на которую будут значения в таблицах обновляться (рис. 56)

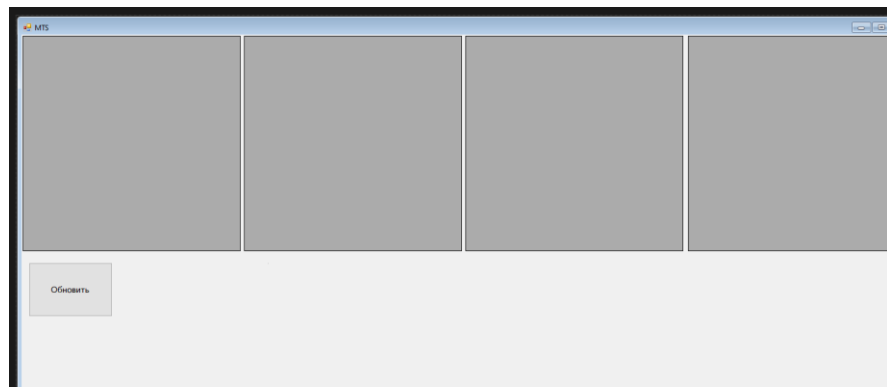


Рис. 56.

Добавим ещё 2 формы, в 1-ой мы будем добавлять нового пользователя, а во 2-ой делать запрос на детализацию звонков за определённый период.

Итоговая 1-я форма (рис. 57).

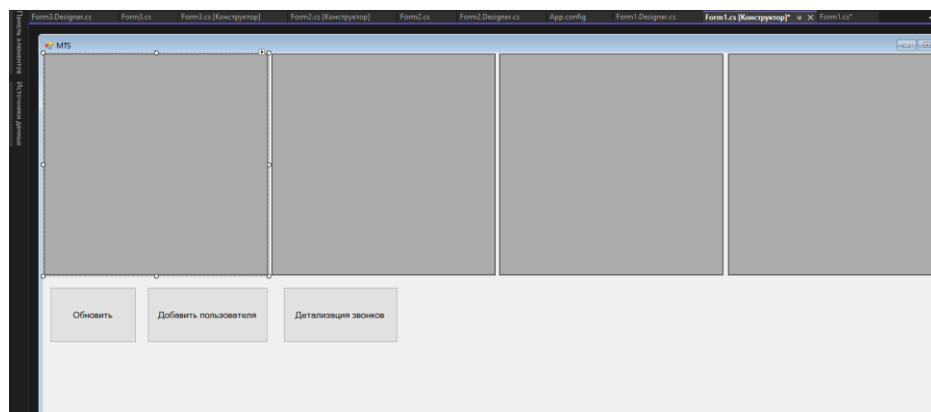


Рис. 57.

Вторая форма (рис. 58)

Рис. 58.

Третья форма (рис. 59).

Рис. 59.

У всех объектов есть свои имена, которые уже используются в коде.

Код для начальной формы.

Таблица 3. Код Form1.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace MTS
{
    public partial class MTS : Form
    {
        // Строка подключения с аутентификацией Windows
        private string connectionString = @"Server=localhost;Database=MTS;Trusted_Connection=True;";

        public MTS()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;

            // Привязка обработчика к кнопке
            buttonRefresh.Click += ButtonRefresh_Click;
            buttonAddClient.Click += buttonAddClient_Click;
            buttonCallDetails.Click += buttonCallDetails_Click;
        }

        private void MTS_Load(object sender, EventArgs e)
        {
            LoadAllData();
        }
    }
}
```

```

    }

    private void ButtonRefresh_Click(object sender, EventArgs e)
    {
        LoadAllData();
    }

    private void LoadAllData()
    {
        LoadData("SELECT * FROM Tariff", dataGridViewTariff);
        LoadData("SELECT * FROM Client", dataGridViewClient);
        LoadData("SELECT * FROM Calls", dataGridViewCalls);
        LoadData("SELECT * FROM Call_Detail_Request", dataGridViewCallDetailRequest);
    }

    private void LoadData(string query, DataGridView dgv)
    {
        try
        {
            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
                DataTable dt = new DataTable();
                adapter.Fill(dt);
                dgv.DataSource = dt;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void buttonAddClient_Click(object sender, EventArgs e)
    {
        using (AddClientForm addClientForm = new AddClientForm())
        {
            if (addClientForm.ShowDialog() == DialogResult.OK)
            {
                // После добавления клиента обновляем данные на главной форме
                LoadData("SELECT * FROM Client", dataGridViewClient);
            }
        }
    }

    private void buttonCallDetails_Click(object sender, EventArgs e)
    {
        using (CallDetailsForm form = new CallDetailsForm())
        {
            form.StartPosition = FormStartPosition.CenterScreen;
            form.ShowDialog();

            // Обновляем таблицу после возможного добавления запроса
            LoadData("SELECT * FROM Call_Detail_Request", dataGridViewCallDetailRequest);
        }
    }
}

```

Код для второй формы с добавлением пользователя.

Таблица 4. Код Form2.cs

| | |
|---|--|
| <pre> using System; using System.Data.SqlClient; using System.Linq; using System.Windows.Forms; namespace MTS { public partial class AddClientForm : Form { private string connectionString = @"Server=localhost;Database=MTS;Trusted_Connection=True;"; public AddClientForm() { InitializeComponent(); this.StartPosition = FormStartPosition.CenterScreen; comboBoxClientType.SelectedIndexChanged += ComboBoxClientType_SelectedIndexChanged; Load += AddClientForm_Load; buttonSave.Click += ButtonSave_Click; // Настройка маски для maskedTextBoxPhone maskedTextBoxPhone.Mask = "8 (999) 000-00-00"; maskedTextBoxPhone.SkipLiterals = true; // курсор пропускает скобки и пробелы maskedTextBoxPhone.Text = "8 "; // +7 по умолчанию } private void AddClientForm_Load(object sender, EventArgs e) { comboBoxClientType.Items.Clear(); comboBoxClientType.Items.AddRange(new string[] { "Физическое", "Юридическое" }); comboBoxClientType.SelectedIndex = 0; } private void ComboBoxClientType_SelectedIndexChanged(object sender, EventArgs e) { string selectedType = comboBoxClientType.SelectedItem.ToString(); LoadTariffsForClientType(selectedType); } private void LoadTariffsForClientType(string clientType) { string tariffType = (clientType == "Юридическое") ? "Корпоративный" : "Некорпоративный"; try { using (SqlConnection conn = new SqlConnection(connectionString)) { conn.Open(); string query = "SELECT tariff_id, tariff_name FROM Tariff WHERE tariff_type = @" + tariffType"; using (SqlCommand cmd = new SqlCommand(query, conn)) { </pre> | <pre> // Проверяем уникальность номера if (PhoneExists(phone)) { MessageBox.Show("Клиент с таким номером телефона уже существует"); return; } string name = textBoxName.Text.Trim(); string clientType = comboBoxClientType.SelectedItem.ToString(); int tariffId = (int)((ComboBoxItem)comboBoxTariff.SelectedItem).Value; try { using (SqlConnection conn = new SqlConnection(connectionString)) { conn.Open(); string insertQuery = @"INSERT INTO Client (name, client_type, balance, phone_number, tariff_id) VALUES (@name, @client_type, @balance, @phone_number, @tariff_id)"; using (SqlCommand cmd = new SqlCommand(insertQuery, conn)) { cmd.Parameters.AddWithValue("@name", name); cmd.Parameters.AddWithValue("@client_type", clientType); cmd.Parameters.AddWithValue("@balance", balance); cmd.Parameters.AddWithValue("@phone_number", phone); cmd.Parameters.AddWithValue("@tariff_id", tariffId); cmd.ExecuteNonQuery(); } } MessageBox.Show("Клиент успешно добавлен"); this.DialogResult = DialogResult.OK; this.Close(); } catch (Exception ex) { MessageBox.Show("Ошибка при добавлении клиента: " + ex.Message); } } </pre> |
|---|--|

| | |
|--|--|
| <pre> cmd.Parameters.AddWithValue("@tariffType", tariffType); using (SqlDataReader reader = cmd.ExecuteReader()) { comboBoxTariff.Items.Clear(); while (reader.Read()) { comboBoxTariff.Items.Add(new ComboboxItem() { Text = reader["tariff_name"].ToString(), Value = reader["tariff_id"] }); } } if (comboBoxTariff.Items.Count > 0) comboBoxTariff.SelectedIndex = 0; } catch (Exception ex) { MessageBox.Show("Ошибка загрузки тарифов: " + ex.Message); } private void ButtonSave_Click(object sender, EventArgs e) { // Проверяем имя if (string.IsNullOrEmpty(textBoxName.Text)) { MessageBox.Show("Введите имя клиента"); return; } // Проверяем выбран тариф if (comboBoxTariff.SelectedItem == null) { MessageBox.Show("Выберите тариф"); return; } // Проверяем баланс if (!decimal.TryParse(textBoxBalance.Text, out decimal balance)) { MessageBox.Show("Введите корректный баланс"); return; } // Получаем номер телефона из maskedTextBox и очищаем от лишних символов string phoneRaw = maskedTextBoxPhone.Text; string phone = "+" + new string(phoneRaw.Where(char.IsDigit).ToArray()); if (!IsValidPhoneNumber(phone)) { MessageBox.Show("Номер телефона должен быть в формате +7 (XXX) XXX-XX-XX"); return; } </pre> | <pre> private bool IsValidPhoneNumber(string phone) { // Проверяем, что строка в формате 8 и 11 цифр (8 + 10) if (phone.Length != 11) return false; if (!phone.StartsWith("8")) return false; for (int i = 2; i < phone.Length; i++) { if (!char.IsDigit(phone[i])) return false; } return true; } private bool PhoneExists(string phone) { try { using (SqlConnection conn = new SqlConnection(connectionString)) { conn.Open(); string query = "SELECT COUNT(*) FROM Client WHERE phone_number = @phone"; using (SqlCommand cmd = new SqlCommand(query, conn)) { cmd.Parameters.AddWithValue("@phone", phone); int count = (int)cmd.ExecuteScalar(); return count > 0; } } } catch { // В случае ошибки считаем, что номер существует, // чтобы избежать дублирования return true; } } // Класс для удобного хранения пары текст/значение в ComboBox public class ComboboxItem { public string Text { get; set; } public object Value { get; set; } public override string ToString() { return Text; } } </pre> |
|--|--|

Код для третьей формы с запросом звонков от пользователя за определённый период.

Таблица 5. Код Form3.cs

| | |
|--|--|
| <pre> using System; using System.Data.SqlClient; using System.Linq; using System.Windows.Forms; namespace MTS { public partial class CallDetailsForm : Form { private string connectionString = @"Server=localhost;Database=MTS;Trusted_Connection=True;"; public CallDetailsForm() { InitializeComponent(); this.StartPosition = FormStartPosition.CenterScreen; // Настройка маски maskedTextBoxPhone.Mask = "8 (000) 000-00-00"; maskedTextBoxPhone.Text = "8 "; maskedTextBoxPhone.TextChanged += MaskedTextBoxPhone_TextChanged; listBoxSuggestions.Visible = false; listBoxSuggestions.Click += ListBoxSuggestions_Click; buttonSubmitRequest.Click += buttonSubmitRequest_Click; } private void MaskedTextBoxPhone_TextChanged(object sender, EventArgs e) { string rawInput = string(maskedTextBoxPhone.Text.Where(char.IsDigit).ToArray()); if (rawInput.Length < 1) { listBoxSuggestions.Visible = false; return; } try { using (SqlConnection conn = new SqlConnection(connectionString)) { conn.Open(); string query = "SELECT phone_number FROM Client WHERE phone_number LIKE @pattern"; using (SqlCommand cmd = new SqlCommand(query, conn)) { cmd.Parameters.AddWithValue("@pattern", rawInput + "%"); using (SqlDataReader reader = cmd.ExecuteReader()) { listBoxSuggestions.Items.Clear(); while (reader.Read()) </pre> | <pre> int? clientId = GetClientIdByPhone(phone); if (clientId == null) { MessageBox.Show("Клиент с таким номером не найден"); return; } try { using (SqlConnection conn = new SqlConnection(connectionString)) { conn.Open(); // Вставка запроса на детализацию string insertQuery = @"INSERT INTO Call_Detail_Request (client_id, start_date, end_date) VALUES (@client_id, @start_date, @end_date)"; using (SqlCommand insertCmd = new SqlCommand(insertQuery, conn)) { insertCmd.Parameters.AddWithValue("@client_id", clientId.Value); insertCmd.Parameters.AddWithValue("@start_date", startDate); insertCmd.Parameters.AddWithValue("@end_date", endDate); insertCmd.ExecuteNonQuery(); } // Получение статистики по звонкам клиента за указанный период string statsQuery = @" SELECT COUNT(*) AS CallCount, ISNULL(SUM(call_cost), 0) AS TotalCost FROM Calls WHERE client_id = @client_id AND call_time >= @start_date AND call_time <= @end_date"; using (SqlCommand statsCmd = new SqlCommand(statsQuery, conn)) { statsCmd.Parameters.AddWithValue("@client_id", clientId.Value); statsCmd.Parameters.AddWithValue("@start_date", startDate); statsCmd.Parameters.AddWithValue("@end_date", endDate); using (SqlDataReader reader = statsCmd.ExecuteReader()) { if (reader.Read()) { int callCount = reader.GetInt32(0); decimal totalCost = reader.GetDecimal(1); MessageBox.Show(\$"Запрос на детализацию успешно добавлен.\n\n" + </pre> |
|--|--|

| | |
|---|--|
| <pre> { listBoxSuggestions.Items.Add(reader.GetString(0)); } } } listBoxSuggestions.Visible = listBoxSuggestions.Items.Count > 0; } catch { listBoxSuggestions.Visible = false; } } private void listBoxSuggestions_Click(object sender, EventArgs e) { if (listBoxSuggestions.SelectedItem != null) { string selectedPhone = listBoxSuggestions.SelectedItem.ToString(); maskedTextBoxPhone.Text = FormatPhone(selectedPhone); listBoxSuggestions.Visible = false; } } private string FormatPhone(string phone) { if (phone.Length != 11) return phone; return \$"8 ({phone.Substring(1, 3)}) {phone.Substring(4, 3)}-{ phone.Substring(7, 2)}-{phone.Substring(9, 2)}"; } private void buttonSubmitRequest_Click(object sender, EventArgs e) { string phoneRaw = maskedTextBoxPhone.Text; string phone = "8" + new string(phoneRaw.Where(char.IsDigit).Skip(1).ToArray()); if (phone.Length != 11 !phone.StartsWith("8")) { MessageBox.Show("Введите корректный номер телефона в формате 8 (XXX) XXX-XX-XX"); return; } DateTime startDate = dateTimePickerStart.Value.Date; DateTime endDate = dateTimePickerEnd.Value.Date; if (endDate < startDate) { MessageBox.Show("Конечная дата не может быть раньше начальной"); return; } </pre> | <pre> \$"Период: {startDate:dd.MM.yyyy} {endDate:dd.MM.yyyy}\n" + \$"Количество звонков: {callCount}\n" + \$"Общая стоимость звонков: {totalCost:C}", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information); } } this.DialogResult = DialogResult.OK; this.Close(); } } catch (Exception ex) { MessageBox.Show("Ошибка при добавлении запроса: " + ex.Message); } private int? GetClientIdByPhone(string phone) { try { using (SqlConnection conn = new SqlConnection(connectionString)) { conn.Open(); string query = "SELECT client_id FROM Client WHERE phone_number = @phone"; using (SqlCommand cmd = new SqlCommand(query, conn)) { cmd.Parameters.AddWithValue("@phone", phone); var result = cmd.ExecuteScalar(); return result != null ? Convert.ToInt32(result) : (int?)null; } } } catch { return null; } } } </pre> |
|---|--|

Теперь пройдемся по работе приложения. Запустим приложение (рис. 60).

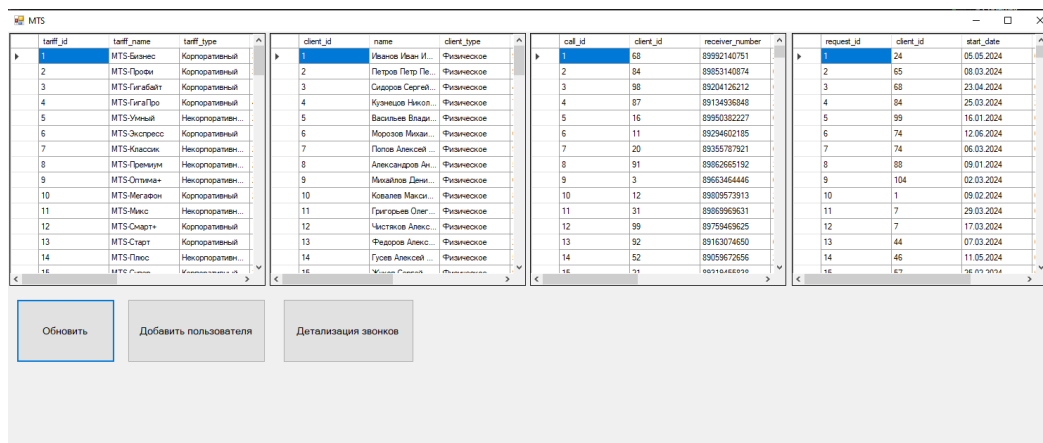


Рис. 60. Вход в приложение

Теперь добавим пользователя (рис. 61).

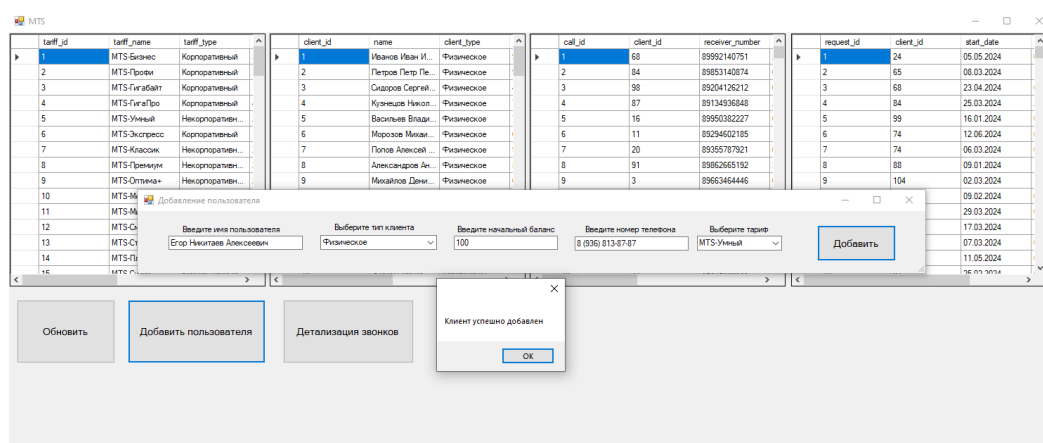


Рис. 61. Добавление пользователя

Посмотрим в таблице (рис. 62).

| | name | client_type | balance |
|---|-------------------|-------------|------------|
| | ООО "НейроГру... | Юридическое | 2274,3100 |
| | ООО "Инноваци... | Юридическое | 3415,4000 |
| | ИП "Лисенко В... | Юридическое | 3225,7100 |
| | АО "РеалТехПро" | Юридическое | 3100,0100 |
| ▶ | АО "Технодрайв" | Юридическое | 3693,4600 |
| | АО "НейроСист... | Юридическое | 1314,4300 |
| | ООО "Системн... | Юридическое | 3425,1400 |
| | ООО "ТехСисте... | Юридическое | 4858,7200 |
| | ООО "Прогресс... | Юридическое | 1026,2900 |
| | ИП "Новиков А... | Юридическое | 4339,2000 |
| | АО "КосмоГрупп" | Юридическое | 3738,4000 |
| | АО "ТехноСтрим" | Юридическое | -3768,6100 |
| | Егор Никитаев ... | Физическое | 100,0000 |
| * | | | |

Рис. 62. Проверка добавленного пользователя

Добавился успешно. Теперь закажем детализацию звонков (рис. 63).

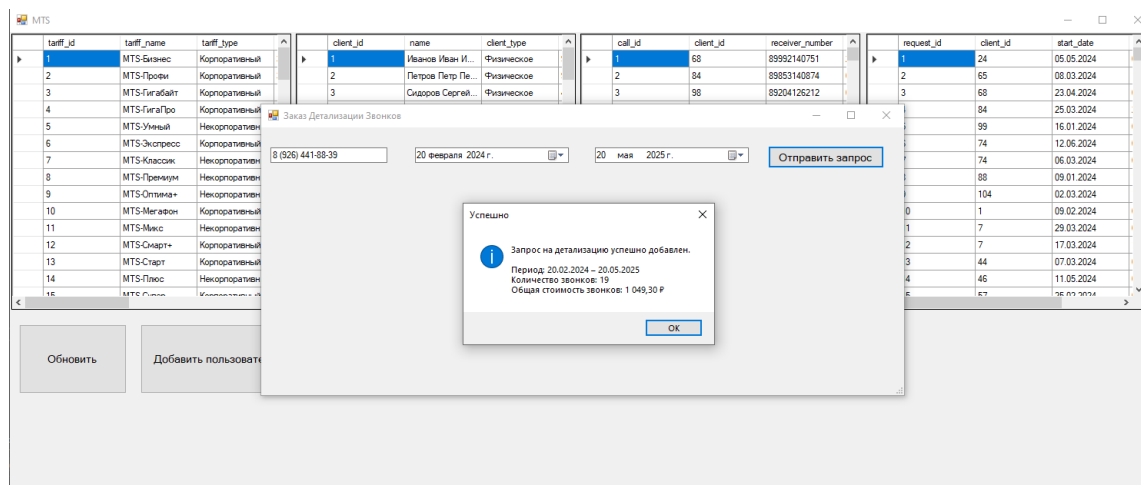


Рис. 63. Заказ детализации

Проверим данный запрос в таблице (рис. 64).

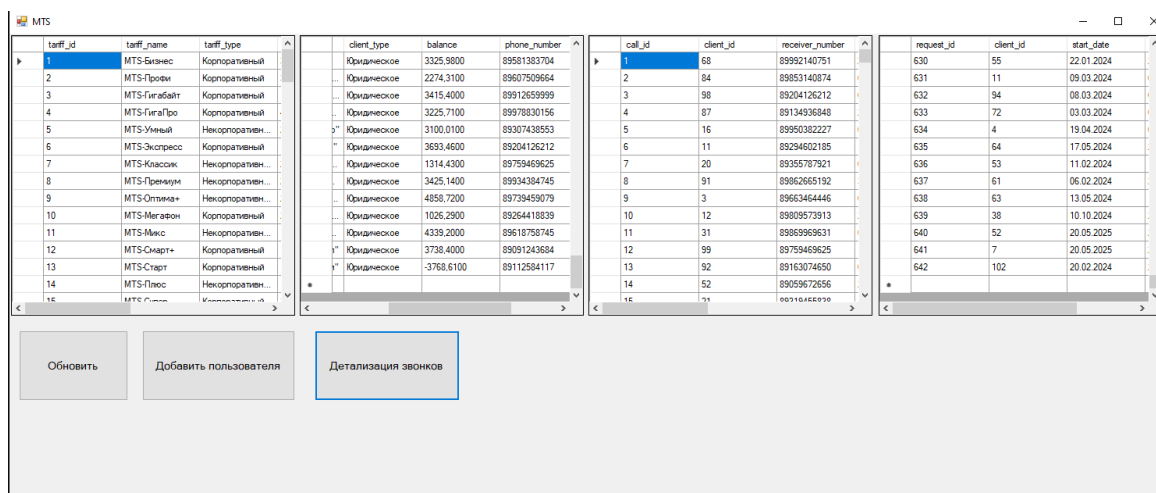


Рис. 64. Проверка детализации запроса

Всё корректно заполнилось.

Вывод

В ходе курсового проекта была разработана система для учета клиентов и звонков телекоммуникационной компании. На основе анализа предметной области создана база данных в MS SQL Server с проработанными связями, ограничениями целостности и бизнес-логикой, реализованной через хранимые процедуры и триггеры. Для работы с базой данных создано Windows Forms-приложение на языке C#, позволяющее регистрировать клиентов, выбирать тарифы, оформлять запросы на детализацию звонков и просматривать статистику по ним. Интерфейс предусматривает маску ввода номера телефона, автоподсказки и валидацию данных. Разработка охватывает все основные аспекты построения и использования реляционных баз данных и демонстрирует интеграцию клиентского приложения с серверной частью.