Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых »
(ВлГУ)**

Кафедра Физики и Прикладной математики

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по курсовому проектированию

по дисциплине

«БАЗЫ ДАННЫХ»

Владимир 2013 г.

ЦЕЛЬ И ЗАДАЧИ КУРСОВОГО ПРОЕКТИРОВАНИЯ

Целью курсового проектирования является закрепление и углубление теоретических знаний, приобретение практических навыков проектирования систем обработки данных, т.е. умения по поставленным задачам сформулировать конкретные технико-экономические требования к объекту проектирования, умения обоснованно выбрать из ряда возможных технических решений одно, наиболее целесообразное, умения выполнять необходимые расчеты, умения коротко и ясно изложить в письменной форме обоснования принятых решений, умения оформлять результаты проектирования.

ВЫБОР И ЗАКРЕПЛЕНИЕ ТЕМЫ КУРСОВОЙ РАБОТЫ (ПРЕКТА)

Тема и задание на курсовую работу выбирается студентом в начале семестра из приведенного ниже перечня, однако студент в праве предложить свою тему, которая должна быть согласована с ведущим преподавателем и утверждена на кафедре. Допускается выдача комплект-задания нескольким студентам при условии четкого выделения задания каждому из них. Тематику типовых курсовых работ разрабатывает и корректирует кафедра. В отдельных случаях курсовое проектирование может быть проведено в интересах научно-исследовательских работ кафедры или заинтересованных предприятий и организаций, с которыми кафедра имеет постоянные творческие связи. Все темы курсовых работ должны быть утверждены кафедрой до начала проектирования.

В соответствии с утвержденной темой курсовой работы кафедрой назначается руководитель, который в начале семестра выдает студентам задание.

СТУКТУРА КУРСОВОЙ РАБОТЫ И ЕЕ ОФОРМЛЕНИЕ

Пояснительная записка к курсовой работе должна включать в себя следующие разделы:

1) титульный лист (см. приложение 1);
2) задание на курсовую работу;
3) оглавление;
4) введение;
5) основные разделы работы;
6) заключение;
7) список использованной литературы;
8) приложения.

Текстовая часть работы должна содержать анализ решаемых в ходе курсового проектирования задач, описание предложенных алгоритмов, структурных и функциональных схем и расчеты в соответствии с выданным заданием. Текст пояснительной записки оформляется в соответствии с ГОСТ. Полный объем текстовой части вместе с с иллюстративным материалом не должен превышать 30 – 35 страниц рукописного текста. К иллюстративным материалам обычно относятся схемы алгоритмов, графики, диаграммы и другие материалы, поясняющие этот текст. Текст пишется с одной стороны на листах бумаге формата А4 (297х210 мм) с оставлением полей у кромок листа.

Содержание основных разделов пояснительной записки рекомендуется разбивать на главы, отражающие основные этапы курсового проектирования.

Все листы записки, кроме титульного, должны иметь сквозную нумерацию. Каждый раздел ее должен быть снабжен соответствующим заголовком, отдельные части разделов могут иметь подзаголовки.

Нумерация таблиц и рисунков сквозная, двойная (по номеру главы и номерам рисунков или таблицы в пределах данной главы, напрмер рис. 1.2 или табл. 1.2), причем подписи к рисункам помещаются под ними, а к таблицам – над ними.

Все использованные в расчетах формулы, выводы которых не приводятся, должны сопровождаться ссылками на литературные источники, из которых они заимствованы. То же самое следует делать и в отношении используемых в записке справочных данных, иллюстраций и графиков. При ссылке на литературный источник в тексте указывается номер источника, помещенный в списке литературы, который заключается в квадратные скобки, например [6].

В заключение пояснительной записки необходимо привести основные результаты проектирования и оценить, насколько полученные проектные решения соответствуют требованиям задания.

Справочный и вспомогательный текстовый материал, алгоритмы и листинги программ, затрудняющие чтение основного текста пояснительной записки, помещаются в приложении. При этом на каждом листе приложения в правом верхнем углу пишется слово «Приложение» и ставится его номер. Ссылка на приложение в тексте указывается в скобках (приложение №).

ЗАЩИТА КУРСОВОЙ РАБОТЫ

После завершения необходимых расчетов и оформления записки в виде вшитого в твердый переплет материала курсовая работа представляется на проверку руководителю. Правильно выполненная и оформленная работа допускается к защите. Защиту работы принимает комиссия, включающая не менее двух преподавателей, ведущих курсовое проектирования по данной дисциплине, назначенная руководителем кафедры.

Защита курсовой работы заключается в докладе о проделанной работе, направленной на достижение основной цели, и решения задач, оговоренных в задании, а также в ответах на поставленные комиссией вопросы, и выступления рецензента проекта. По времени доклад не должен превышать 7 мин, в течение которых докладчику необходимо:

изложить цель и основные задачи проектирования;

на основе анализа технических решений обосновать выбор своего решения, исходя из заданных требований и уровня развития современной вычислительной техники, программных средств и технологии обработки информации.

изложить основные результаты проведенных расчетов;

кратко описать предлагаемую структуру таблиц базы данных, схему данных и алгоритмы обработки информации, используя представленные графические материалы;

провести анализ результатов проектирования, на основе которого сделать выводы, насколько полно удовлетворены поставленные в задании требования.

Курсовое проектирование в некоторых случаях может предусматривать взаимное рецензирование работ самими студентами под руководством одного из преподавателей, ведущих курсовое проектирование по данной тематике. В таком случае рецензия должна содержать:

общую характеристику рецензируемой работы и оценку принятых решений;

оценку достоинств и недостатков предложенных решений, алгоритмов обработки информации в свете основных тенденций развития программного обеспечения и аппаратных средств вычислительной техники;

оценку качества выполнения пояснительной записки и графических материалов в свете требований современных ГОСТов;

заключение, содержащее общую оценку курсовой работы.

За рецензирование курсовой работы студенту-рецензенту комиссия выставляет оценку, которая учитывается при выставлении ему окончательной оценки его курсовой работы, качество доклада и ответы на вопросы по теме рассмотренных в работе предложений.

ВАРИАНТЫ ЗАДАНИЙ НА КУРСОВУЮ РАБОТУ

В а р и а н т 1.

Тема «Проектирование и программная реализация базы данных интернет-провайдера».

Основные разделы:

1. Постановка задачи курсового проектирования.
2. Анализ задачи.
3. Информационная модель.
4. Реляционная модель.
5. Реализация бизнес-логики приложения.
6. Описание демонстрационных данных и инициализация базы данных.
7. Имитация работы клиентского приложения.
8. Интерфейс пользователя.

В а р и а н т 2.

Тема « Разработка клиент-серверной базы данных для интернет-магазина цифровой техники»

Основные разделы:

1. Анализ предметной области.
2. Реляционная модель.
3. Схема данных.
4. Основные команды языка SQL.
5. Поставщики данных .NET Framework (ADO.NET).
6. Основные объекты поставщиков данных .NET Framework.
7. Интерфейс IDataReader и IDbCommand.
8. Программная реализация
   - 8.1. Серверная часть БД.
     - 8.1.1. Создание таблиц БД.
     - 8.1.2. Создание первичных ключей БД.
     - 8.1.3. Создание вторичных ключей БД.
     - 8.1.4. Ввод данных.
   - 8.2. Клиентская часть БД.
     - 8.2.1. Создание формы соединение с сервером.
     - 8.2.2. Создание формы для работа с БД.
9. Тестирование приложения.
   - 9.1. Соединение с сервером.
   - 9.2. Добавление данных.

В а р и а н т 3.

Тема «Информационная система поддержки работы менеджера магазина компьютерной техники».

Основные разделы:

1. Анализ предметной области
1.1. Функциональные требования к системе
1.2. Словарь предметной области
1.3. Краткое описание приложения
2. Реляционная модель
2.1. Схема данных
2.2. Хранимые процедуры и триггеры
3. Программная реализация системы
3.1. Структура приложения
3.2. Программный интерфейс
3.3. Пользовательский интерфейс
3.4. Руководство пользователя

В а р и а н т 4.

Тема «База данных банковской системы обслуживания клиентов с электронными кредитными картами»

Основные разделы:

1. Постановка задачи курсового проектирования
2. Анализ задачи.
3. Информационная модель.
4. Реляционная модель.
5. Реализация бизнес-логики приложения.
6. Описание демонстрационных данных и инициализация базы данных.
7. Имитация работы клиентского приложения.
8. Интерфейс пользователя.

СПИСОК ЛИТЕРАТУРЫ

1. Райордан Р. Основы реляционных баз данных / Пер. с англ.- М.:Издатель-ско-торговый дом «Русская Редакция», 2001.- 384. ISBN 5-7502-0150-3.
2. Кониолли Томас, Бегг Каролин. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.:Пер. с англ..- М.: Изда-тельский дом «Вильямс»2003.- 1440 с. . ISBN 5-8459-0527-3.
3. Виейра Роберт. Программирование баз данных Microsoft SQL Server 2005. Базовый курс.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2007.- 892 с.
   . ISBN 978-5-8459-2.
4. Фролов А.В., Фролов Г.В. Базы данных в Интернете: практическое руководство по созданию Web-приложений с базами данных.- М.: Издательско-торговый дом «Русская Редакция», 2004.- 448 с. . ISBN 5-7502-0174-0.
5. Дж. Грофф, П.Вайнберг. SQL: Полное руководство: Пер. с англ.- 2-е изд.- К.: Издательская группа BHV, 2001.- 816 с. . ISBN 966-552-073-3
6. Михеев Р.Н. MS SQL Server 2005 для администраторов.- СПб.: БХВ- Петербург, 2007.- 544 с. . ISBN 978-5-94157-796-5.
7. Хендерсон К. Профессиональное руководство по SQL Server: Хранимые процедуры, XML, HTML.- СПб.: Питер, 2005.- 620 с. . ISBN 5-469-00046-5.
8. Троеслен Э. C# и платформа NET. Библиотека программиста.- СПб.: Питер, 2004.- 796 с. ISBN 5-318-00750-3
9. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ.- М.: Издательско-торговый дом «Русская Редакция», 2002.- 576 с. ISBN 5-7502-0210-0.
10. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ.- М.: Издательско-торговый дом «Русская Редакция», 2002.- 624 с. ISBN 5-7502-0220-8.
11. Сеппа Д. Microsoft ADO.NET / Пер. С англ.- М.: .: Издательско-торговый дом «Русская Редакция», 2003.- 640 с. ISBN 5-7502-0223-2.
12. Малик Сахил. Microsoft ADO.NET 2.0 для профессионалов.: Пер. с англ.- М.: ООО «И.Д. Вильямс», 2006.- 560 с. ISBN 5-84559-1080-3.

ПРИЛОЖЕНИЕ 1.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых »
(ВлГУ

Кафедра Физики и Прикладной математики

КУРСОВАЯ РАБОТА (ПРОЕКТ)
на тему _____
_____

по дисциплине

БАЗЫ ДАННЫХ

Студент_____
(ф.и.о)
Группа_____
Руководитель работы
_____
(должность, ф.и.о)

Дата сдачи законченной
работы на проверку
«____»_____201__ г.
Защищена «___»_____201__г.
Оценка «_____». Подписи членов комиссии_____

Владимир, 201__ г.

ПРИЛОЖЕНИЕ 2.

Пример выполнения курсовой работы на тему «Проектирование и программная реализация базы данных интернет-провайдера».

1. Постановка задачи

Разработать базу данных пользователей интернет-провайдера с возможностью ведения истории соединений и расчета стоимости.

2. Анализ задачи

Бизнес-логика разрабатываемого приложения должна соответствовать всем указанным замечаниям, а именно должны быть разработаны бизнес-процессы для оперирования списками клиентов, тарифов, подключений и платежей, а также отображения статистики по ним. Перечень процессов выглядит так:

- добавление и удаление тарифных планов;
- добавление и удаление клиентов;
- изменение тарифных планов для клиентов;
- пополнение счета клиентами;
- инициация соединений клиентами;
- контроль текущего баланса клиентов;
- статистика по соединениям и клиентам;
- отображение информации по соединениям, клиентам, платежам и тарифным планам.

3. Информационная модель

Информационная модель предполагает наличие следующих сущностей.

1. Клиент – представляет пользователя услуг интернет-провайдера.
   Атрибуты:
   - идентификатор клиента;
   - имя клиента;
   - физический адрес клиента;
   - ip-адрес клиента;
   - номер договора данного клиента;
   - текущий баланс.
2. Тариф – представляет собой атрибут услуг провайдера.
   Атрибуты:

- идентификатор тарифа;
- название тарифа;
- цена интернет трафика за 1 Мб;
- стартовая цена подключения к данному тарифу.

3. Подключение – представляет собой информацию об очередном сеансе работы пользователя в сети.
   Атрибуты:
   - идентификатор подключения;
   - дата и время подключения;
   - количество израсходованного интернет-трафика.
4. Платеж – представляет собой информацию о пополнении клиентом своего баланса.
   Атрибуты:
   - идентификатор платежа;
   - сумма;
   - дата совершения.

4. Реляционная модель

Преобразуем информационную модель в реляционную (рис. 4.1 Диаграмма).

Табл.1

Описание таблиц

Имя таблицы	Поле	Тип поля	Описание поля
Clients	clientID (PK)	INT IDENTITY(1,1)	идентификатор клиента
	clientName	NVARCHAR (30)	имя клиента
	address	NVARCHAR (40)	физический адрес
	ip	NVARCHAR (15)	ip-адрес клиента
	treatyNum	NVARCHAR (10)	номер договора

	balance	MONEY	текущий баланс
	tariff_id (FK)	INT	идентификатор тарифа
Tariffes	tariffID (PK)	INT IDENTITY(1,1)	идентификатор тарифа
	tariffName	NVARCHAR (20)	название тарифа
	cost	MONEY	цена 1 Мб трафика
	start_cost	MONEY	цена подключения к тарифу
Connections	connectionID (PK)	INT IDENTITY(1,1)	идентификатор соединения
	date	DATETIME	дата и время соединения
	mbCount	FLOAT	затраченный трафик (в Мб)
	client_id (FK)	INT	идентификатор клиента
Payments	paymentID (PK)	INT IDENTITY(1,1)	идентификатор платежа
	date	DATETIME	дата и время совершения платежа
	summ	MONEY	сумма
	client_id (FK)	INT	идентификатор клиента

Для привязки клиента к тарифному плану таблица Clients имеет внешний ключ tariff_id (связь между таблицами 1 (тариф) ко многим (клиентам)).

Для привязки соединения к клиенту таблица Connections имеет внешний ключ client_id (1 (клиент) ко многим (соединениям)).

Для привязки платежа к клиенту таблица Payments имеет внешний ключ client_id (1 (клиент) ко многим (платежам)).

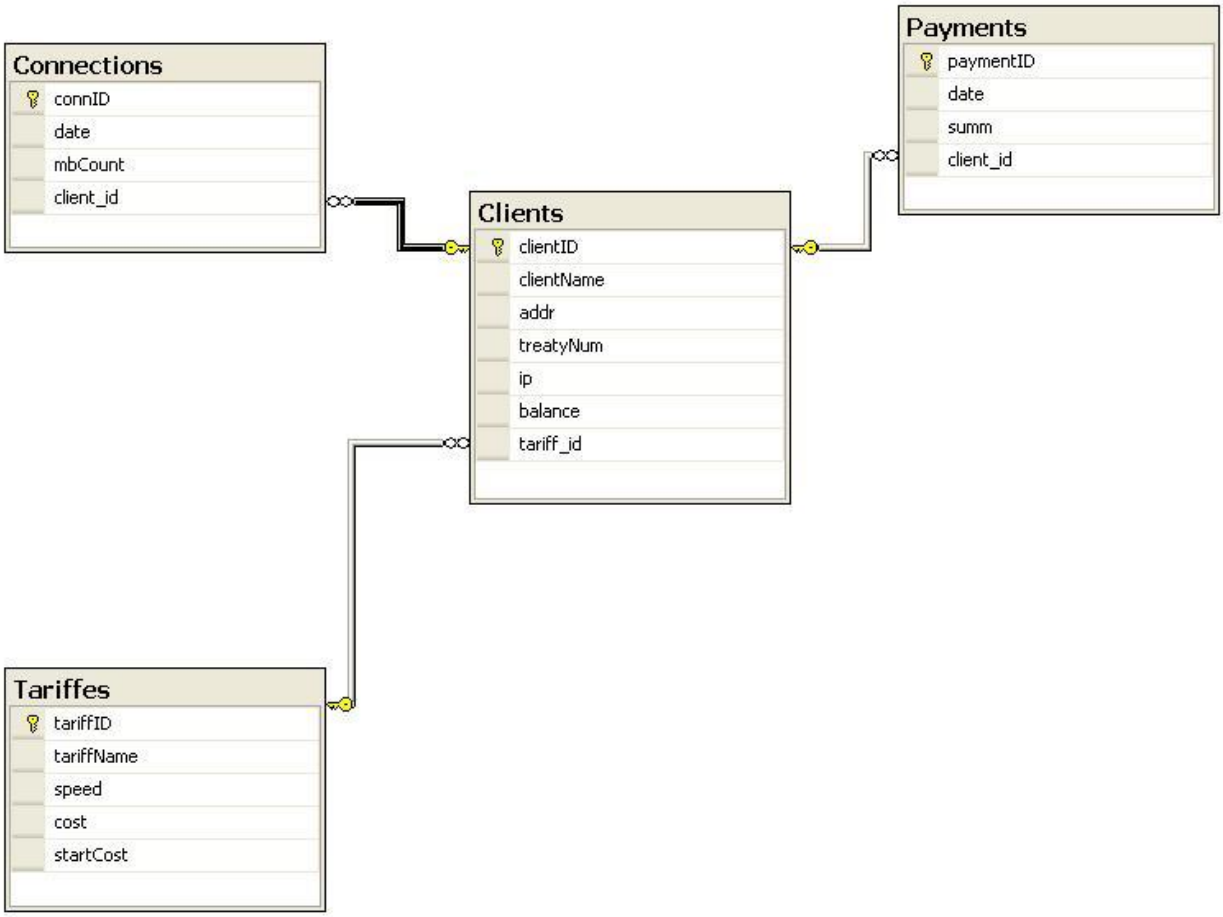


Рис. 4.1 Диаграмма

5. Реализация бизнес-логики приложения

Бизнес-логика приложения реализована с помощью пользовательских хранимых процедур, триггеров (которые являются особым типом хранимых процедур), а также представлений, предназначенных, прежде всего для формирования отчетов по таблицам. Хранимые процедуры созданы на каждое базовое действие пользователя. Здесь представлены описания назначения, а также входных/выходных параметров для каждого объекта. Реализацию см. в приложении.

```
1.

-- процедура добавления Клиента
-- аргументы:
```

```
--          @name        - имя клиента
--          @address     - физический адрес (м.б. NULL)
--          @ip                - ip-адрес
--          @treatyNum   - номер договора
--          @tariffName  - название тарифа, выбранного при заключении
контракта
-- возвращаемое значение:
--          идентификатор нового клиента (-1 - ошибка при выполнении
процедуры)


2.
-- процедура добавления Тарифа
-- аргументы:
--          @name       - название
--          @cost       - цена 1Мб трафика
-- возвращаемое значение:
--          идентификатор нового тарифа (-1 - ошибка при выполнении
процедуры)


3.
-- процедура добавления Платежа
-- аргументы:
--          @treatyNum        - номер договора плательщика
--          @date             - дата
--          @summ             - сумма платежа
-- возвращаемое значение:
--          идентификатор нового платежа (-1 - ошибка при выполнении
процедуры)


4.
-- процедура добавления Подключения
-- аргументы:
--          @ip                    - ip-адрес подключившегося клиента
--          @date             - дата
--          @summ             - сумма платежа
-- возвращаемое значение:
--          идентификатор нового платежа (-1 - ошибка при выполнении
процедуры)


5.
-- процедура удаления подключения
-- аргументы:
--          @connID -  идентификатор удаляемого подключения
-- возвращаемое значение:
--          0 - нет ошибки / -1 - ошибка при удалении


6.
-- процедура удаления платежа
-- аргументы:
```

```
--          @paymentID -  идентификатор удаляемого подключения
-- возвращаемое значение:
--          0 - нет ошибки / -1 - ошибка при удалении


7.
      -- процедура удаления клиента
      -- аргументы:
      --          @clientID -  идентификатор удаляемого клиента
      -- возвращаемое значение:
      --          0 - нет ошибки / -1 - ошибка при удалении


8.
-- процедура удаления тарифа
-- аргументы:
--          @tariffID -  идентификатор удаляемого тарифа
-- возвращаемое значение:
--          0 - нет ошибки / -1 - ошибка при удалении


9.
-- процедура изменения клиентом тарифного плана
-- аргументы:
--          @treatyNum        - номер договора клиента
--          @tariffName       - название нового тарифа
-- возвращаемое значение:
--          идентификатор нового тарифного плана (-1 - ошибка)


10.
-- триггер, связанный с пополнением клиентом своего счета
-- (увеличивает баланс клиента после добавления записи в Payments)


11.
-- триггер, связанный с использованием клиентом трафика
-- (уменьшает баланс клиента после добавления записи в таблицу Connections)


12.
-- триггер, связанный с изменением клиентом своего тарифного плана
-- (изменяет баланс клиента на разницу в стоимости между текущим и новым
--  тарифными планами)


13.
-- процедура подсчета суммарного потребления трафика заданным клиентом
--                 в течении заданного периода времени
-- аргументы:
--          @clientID         - идентификатор клиента
--          @dateBegin        - дата и время начала подсчета
--          @dateEnd          - дата и время конца подсчета
--          @count            - кол-во трафика (OUTPUT)
```

```
--  возвращаемое значение:
--          нет


14.
-- процедура вывода информации о заданным клиенте
-- аргументы:
--          @clientID         - идентификатор клиента
-- возвращаемое значение:
--          нет


15.
-- представление всех сеансов подключений всех клиентов


16.
-- представление всех пополнений счета всех клиентов


17.
-- представление данных о клиентах


18.
-- представление всех тарифных планов с указанием кол-ва клиентов
```

6. Описание демонстрационных данных и инициализация БД

Команды файла PopulateData.sql демонстрируют занесение тестовых демо-данных в БД:

- создается 6 тарифных планов (рис. 2);
- заносятся информация о 6 участниках (рис. 5);
- заносятся 10 записей о подключениях (рис. 3);
- заносятся 8 записей о процедурах пополнения счета (рис. 4).

	tariff...	tariffName	spe...	cost	startC...
1	1	Базовый	128	4,00	100,00
2	2	Продвинутый	256	3,00	300,00
3	3	Деловой	512	2,00	500,00
4	4	Экстра	1024	1,00	1000,00
5	5	Безлимитный-512	512	0,00	4000,00
6	6	Безлимитный-1024	1024	0,00	6000,00

Рис. 2 Выборка данных из таблицы Tariffes после занесения начальных данных.

	conn...	date	mbCo...	client...
1	1	2010-12-01 04:04:04.000	10	1
2	2	2010-12-04 12:34:00.000	3,1	1
3	3	2010-12-05 15:27:36.000	5,02	1
4	4	2010-12-02 14:08:12.000	10	2
5	5	2010-12-12 23:54:30.000	4,05	2
6	6	2010-10-23 18:57:44.000	8,02	3
7	7	2010-11-30 04:52:00.000	14,03	3
8	8	2010-03-13 03:19:28.000	44	4

Рис. 3 Выборка данных из таблицы Connections после занесения начальных данных.

	payment...	date	summ	client...
1	1	2010-10-01 12:14:32.000	100,00	1
2	2	2010-08-19 23:48:49.000	150,00	1
3	3	2010-06-04 15:16:46.000	450,00	1
4	4	2010-11-18 21:26:27.000	543,43	2
5	5	2010-09-04 14:19:58.000	500,00	4
6	6	2010-07-15 06:06:14.000	654,57	5
7	7	2010-06-21 18:21:12.000	95,23	5
8	8	2010-09-12 04:02:05.000	700,00	6

Рис. 4 Выборка данных из таблицы Payments после занесения начальных данных.

	clientID	clientName	addr	treatyN...	ip	balan...	tariff...
1	1	Ефимова Татьяна	ул. Растопчина	623и	253.18.17.01	600,00	2
2	2	Андреева Мария	Москва	3423	253.18.17.02	800,00	4
3	3	Климова Дарья	Владимир	923и	253.18.17.03	400,00	3
4	4	Крупенин Владимир	Радужный	154ил	253.18.17.04	200,00	5
5	5	Марков Андрей	ЛВЗ	3423и	253.56.29.01	100,00	6
6	6	Волкова Ольга	ЛВЗ	666тм	253.34.54.01	220,00	3

Рис. 5 Выборка данных из таблицы Clients после занесения начальных данных.

7. Имитация работы клиентского приложения

Имитация работы приложения заключена в 11 скриптах: Test01.sql, …, Test11.sql .

Test01.sql

Добавляем в Базу новый тарифный план с названием ‘Top Speed’, максимальной скоростью соединения 4 Мбит/с, стоимостью трафика 0.5 руб. за Мб и начальной стоимостью подключения 1000 руб.

```
EXECUTE insertTariff 'Top Speed', 4096, 0.5, 1000
GO
SELECT * FROM Tariffes
GO
```

	tariff...	tariffName	spe...	cost	startC...
1	1	Базовый	128	4,00	100,00
2	2	Продвинутый	256	3,00	300,00
3	3	Деловой	512	2,00	500,00
4	4	Экстра	1024	1,00	1000,00
5	5	Безлимитный-512	512	0,00	4000,00
6	6	Безлимитный-1024	1024	0,00	6000,00
7	7	Top Speed	4096	0,50	1000,00

Рис. 6 Выборка данных из таблицы Tariffes после выполнения скрипта Test01.sql

Test02.sql

Добавляем в Базу нового клиента с именем Владимир Ильич, местом жительства Мавзолей, ip-адресом 273.133.212.201, номером договора 1917ил, начальным балансом 1200 руб. и начальным тарифным планом Базовый.

```
EXECUTE insertClient 'Владимир Ильич', 'Мавзолей', '273.133.212.201',
'1917ил', 1200.0, 'Базовый'
GO
SELECT * FROM Clients
GO
```

	clientID	clientName	addr	treatyN...	ip	balance	tariff...
1	1	Ефимова Татьяна	ул. Растопчина	623и	253.18.17.01	1245,64	2
2	2	Андреева Мария	Москва	3423	253.18.17.02	1329,38	4
3	3	Климова Дарья	Владимир	923и	253.18.17.03	355,90	3
4	4	Крупенин Владимир	Радужный	154ил	253.18.17.04	700,00	5
5	5	Марков Андрей	ЛВЗ	3423и	253.56.29.01	849,80	6
6	6	Волкова Ольга	ЛВЗ	666тм	253.34.54.01	895,70	3
7	7	Владимир Ильич	Мавзолей	1917ил	273.133.212.201	1200,00	1

Рис. 7 Выборка данных из таблицы Clients после выполнения скрипта Test02.sql

Test03.sql

Изменяем тарифный план клиенту с номером договора 1917ил на Top Speed. При этом изменяется также и баланс клиента – с него снимается стоимость подключения к новому тарифу, но компенсируется стоимость уже подключенного тарифа. Реализация такого контроля осуществляется через триггер.

```
EXECUTE changeTariff '1917ил', 'Top Speed'
GO
SELECT * FROM Clients
GO
```

	clientID	clientName	addr	treatyN...	ip	balance	tariff...
1	1	Ефимова Татьяна	ул. Растопчина	623и	253.18.17.01	1245,64	2
2	2	Андреева Мария	Москва	3423	253.18.17.02	1329,38	4
3	3	Климова Дарья	Владимир	923и	253.18.17.03	355,90	3
4	4	Крупенин Владимир	Радужный	154ил	253.18.17.04	700,00	5
5	5	Марков Андрей	ЛВЗ	3423и	253.56.29.01	849,80	6
6	6	Волкова Ольга	ЛВЗ	666тм	253.34.54.01	895,70	3
7	7	Владимир Ильич	Мавзолей	1917ил	273.133.212.201	300,00	7

Рис. 8 Выборка данных из таблицы Clients после выполнения скрипта Test03.sql

Test04.sql

Добавляем сеанс работы пользователя в сети – Connection. Соединения добавляется для пользователя с ip-адресом 273.133.212.201. Дата и время начала работы пользователя - 2009-12-12 04:04:04. Количество затраченного при этом траффика составляет 54.87 Мб.

```
EXECUTE insertConnection '273.133.212.201', '2009-12-12 04:04:04', 54.87

SELECT * FROM Connections
GO
SELECT * FROM Clients

GO
```

	conn...	date	mbCo...	client...
1	1	2010-12-01 04:04:04.000	10	1
2	2	2010-12-04 12:34:00.000	3,1	1
3	3	2010-12-05 15:27:36.000	5,02	1
4	4	2010-12-02 14:08:12.000	10	2
5	5	2010-12-12 23:54:30.000	4,05	2
6	6	2010-10-23 18:57:44.000	8,02	3
7	7	2010-11-30 04:52:00.000	14,03	3
8	8	2010-03-13 03:19:28.000	44	4
9	9	2010-04-18 16:23:06.000	5,09	5
10	10	2010-06-05 12:15:17.000	12,15	6
11	11	2009-12-12 04:04:04.000	54,87	7

Рис. 9 Выборка данных из таблиц Connections и Clients после выполнения скрипта Test04.sql

Test05.sql

Добавление информации о пополнении клиентом своего счета. Клиент с номером договора 1917ил пополнил свой баланс 2009-11-13 12:14:32 на сумму 100 руб.

```
EXECUTE insertPayment '1917èë', '2009-11-13 12:14:32', 100.0

SELECT * FROM Payments
GO
SELECT * FROM Clients

GO
```

	payment...	date	summ	client...
1	1	2010-10-01 12:14:32.000	100,00	1
2	2	2010-08-19 23:48:49.000	150,00	1
3	3	2010-06-04 15:16:46.000	450,00	1
4	4	2010-11-18 21:26:27.000	543,43	2
5	5	2010-09-04 14:19:58.000	500,00	4
6	6	2010-07-15 06:06:14.000	654,57	5
7	7	2010-06-21 18:21:12.000	95,23	5
8	8	2010-09-12 04:02:05.000	700,00	6
9	9	2009-11-13 12:14:32.000	100,00	7

Рис. 10 Выборка данных из таблиц Payments и Clients после выполнения скрипта Test05.sql

Test06.sql

Выводим сводную информацию по тарифам, использую соответствующее представление.

```
SELECT * FROM ViewOfTariffes

GO
```

	Название тарифа	Стоимость 1 Мб трафф...	Стоимость активац...	Число абонен...
1	Базовый	4,00	100,00	0
2	Продвинутый	3,00	300,00	1
3	Деловой	2,00	500,00	2
4	Экстра	1,00	1000,00	1
5	Безлимитный-512	0,00	4000,00	1
6	Безлимитный-1024	0,00	6000,00	1
7	Top Speed	0,50	1000,00	1

Рис. 11 Выборка данных из представления ViewOfTariffes после выполнения скрипта Test06.sql

Test07.sql

Выводим сводную информацию по клиентам.

```
SELECT * FROM ViewOfClients

GO
```

	ID	Имя клиента	Адрес клиента	Баланс	Номер догово...	IP-адрес	Тариф
1	1	Ефимова Татьяна	ул. Растопчина	1245,64	623и	253.18.17.01	Продвинутый
2	2	Андреева Мария	Москва	1329,38	3423	253.18.17.02	Экстра
3	3	Климова Дарья	Владимир	355,90	923и	253.18.17.03	Деловой
4	4	Крупенин Владимир	Радужный	700,00	154ил	253.18.17.04	Безлимитный-512
5	5	Марков Андрей	ЛВЗ	849,80	3423и	253.56.29.01	Безлимитный-1024
6	6	Волкова Ольга	ЛВЗ	895,70	666тм	253.34.54.01	Деловой
7	7	Владимир Ильич	Мавзолей	372,565	1917ил	273.133.212.201	Top Speed

Рис. 12 Выборка данных из представления ViewOfClients после выполнения скрипта Test07.sql

Test08.sql

Выводим сводную информацию по соединениям.

```
SELECT * FROM ViewOfConnections

GO
```

	ID	Имя клиента	IP-адрес	Время подключения	Затраченный траффик (...
1	1	Ефимова Татьяна	253.18.17.01	2010-12-01 04:04:04.000	10
2	1	Ефимова Татьяна	253.18.17.01	2010-12-04 12:34:00.000	3,1
3	1	Ефимова Татьяна	253.18.17.01	2010-12-05 15:27:36.000	5,02
4	2	Андреева Мария	253.18.17.02	2010-12-02 14:08:12.000	10
5	2	Андреева Мария	253.18.17.02	2010-12-12 23:54:30.000	4,05
6	3	Климова Дарья	253.18.17.03	2010-10-23 18:57:44.000	8,02
7	3	Климова Дарья	253.18.17.03	2010-11-30 04:52:00.000	14,03
8	4	Крупенин Владимир	253.18.17.04	2010-03-13 03:19:28.000	44
9	5	Марков Андрей	253.56.29.01	2010-04-18 16:23:06.000	5,09
10	6	Волкова Ольга	253.34.54.01	2010-06-05 12:15:17.000	12,15
11	7	Владимир Ильич	273.133.212.201	2009-12-12 04:04:04.000	54,87

Рис. 13 Выборка данных из представления ViewOfConnections после выполнения скрипта Test08.sql

Test09.sql

Выводим сводную информацию по платежам.

```
SELECT * FROM ViewOfPayments

GO
```

	Имя клиента	Номер догово...	Время платежа	Сумма
1	Крупенин Владимир	154ил	2010-09-04 14:19:58.000	500,00
2	Владимир Ильич	1917ил	2009-11-13 12:14:32.000	100,00
3	Андреева Мария	3423	2010-11-18 21:26:27.000	543,43
4	Марков Андрей	3423и	2010-06-21 18:21:12.000	95,23
5	Марков Андрей	3423и	2010-07-15 06:06:14.000	654,57
6	Ефимова Татьяна	623и	2010-06-04 15:16:46.000	450,00
7	Ефимова Татьяна	623и	2010-08-19 23:48:49.000	150,00
8	Ефимова Татьяна	623и	2010-10-01 12:14:32.000	100,00
9	Волкова Ольга	666тм	2010-09-12 04:02:05.000	700,00

Рис. 14 Выборка данных из представления ViewOfPayments после выполнения скрипта Test09.sql

Test10.sql

Подсчитываем и выводим суммарное количество трафика, израсходованного клиентом в заданный период времени.

```
DECLARE @count1 FLOAT,
            @clientID1 INT,
            @dateBegin1 DATETIME,
            @dateEnd1 DATETIME

SET @clientID1 = 1
SET @dateBegin1 = '2009-12-04 12:34:00'
SET @dateEnd1 = '2009-12-05 15:27:36'

EXECUTE TrafficOfClient
@clientID = @clientID1,
@dateBegin = @dateBegin1,
@dateEnd = @dateEnd1,
@count = @count1 OUTPUT

PRINT 'Summary traffic for client with ID ' + CAST(@clientID1 as NVARCHAR) +
            ' between (' + CAST(@dateBegin1 as NVARCHAR) + ') and (' +
CAST(@dateEnd1 as NVARCHAR) +
            ') is ' + CAST(@count1 as NVARCHAR)

GO
```

```
Summary traffic for client with ID 2 between (Dec  4 2009 12:34PM) and (Dec  5 2009  3:27PM) is 8.12
```

Рис. 15 Результат выполнения скрипта Test10.sql

Test11.sql

Вывод главной (с т.з. провайдера) информации о клиенте с заданным ID.

```
EXECUTE InfoOfClient
@clientID = 7

GO
```

	Имя клиента	Адрес клие...	Номер догово...	IP-адрес	Тариф	Баланс	Суммарный трафф...
1	Владимир Ильич	Мавзолей	1917ил	273.133.212.201	Top Speed	372,565	54,87

Рис. 16 Результат выполнения скрипта Test11.sql

8. Интерфейс пользователя

Интерфейс написан в среде разработки Visual Studio 2008 на языке C# с активным использованием технологии ADO.NET.

ADO.NET предоставляет возможность подключения к различным источникам данных с использованием соответствующих драйверу источника классов. Так для подключения к SQL Server оптимально использовать класс SqlConnection (*System.Data.SqlClient*.SqlConnection), который представляет установленное подключение.

ADO.NET включает два основных уровня:

- connected layer (.NET Data Provider) - набор компонентов специфичных для СУБД для работы в режиме с активным подключением к источнику данных, например, SqlConnection, SqlCommand, SqlDataReader;
- disconnected layer - набор универсальных компонентов абстрагированных от конкретного провайдера данных для работы с данными в режиме без подключения, например, DataSet, DataTable и др.

Хранение данных осуществляется в компонентах DataSet и DataTable., работа с которыми может вестись без активного подключения к источнику. Однако сам источник данных при этом должен быть выбран. Обмен данными с БД производится через класс SqlDataAdapter.

Главная форма пользовательского приложения (рис. 19,20,21) предоставляет основную функциональность при работе с Базой Данных: добавление, удаление, и, частично, модификация записей в таблицах – реализованы вызовами хранимых процедур; просмотр основных сведений по всем таблицам – реализация через представления.

Форма аутентификации (рис. 17,18) предоставляет возможность выбрать источник данных, сервер, тип аутентификации, а также, если требуется, ввести логин и пароль пользователя.

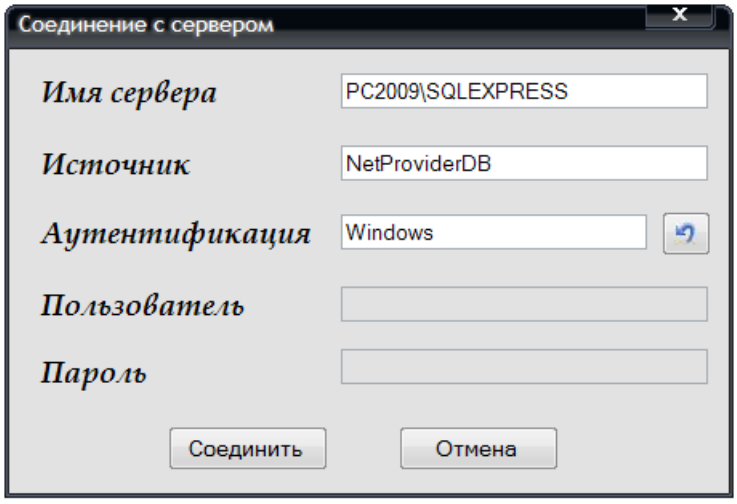


Рис. 17 Внешний вид формы соединения с сервером (аутентификации) пользовательского приложения

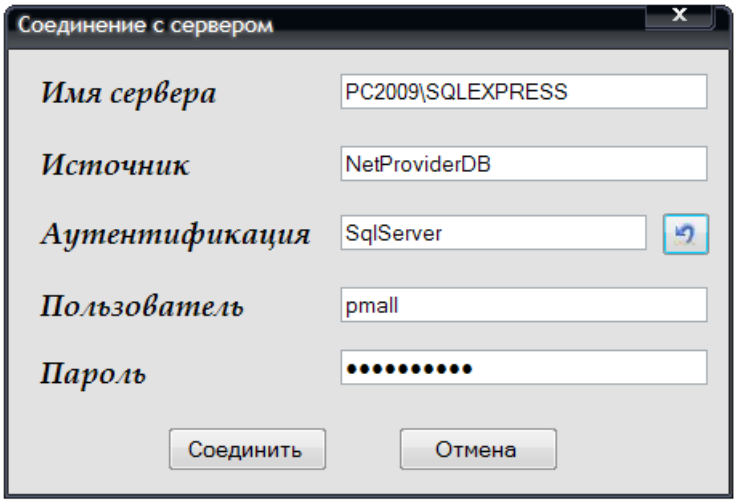
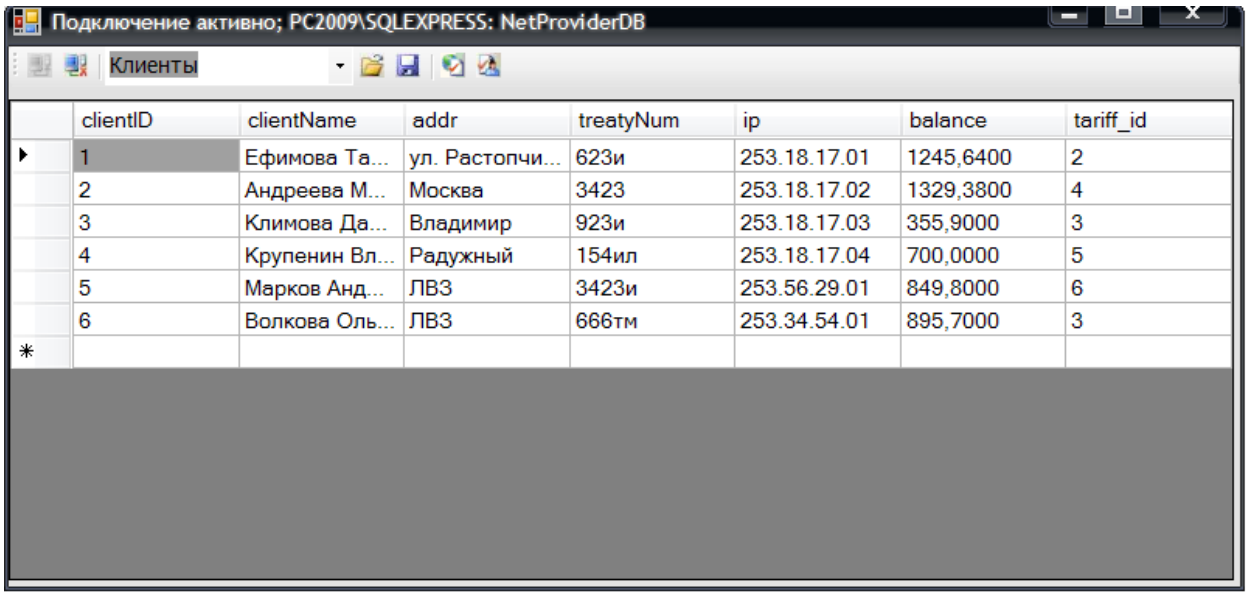
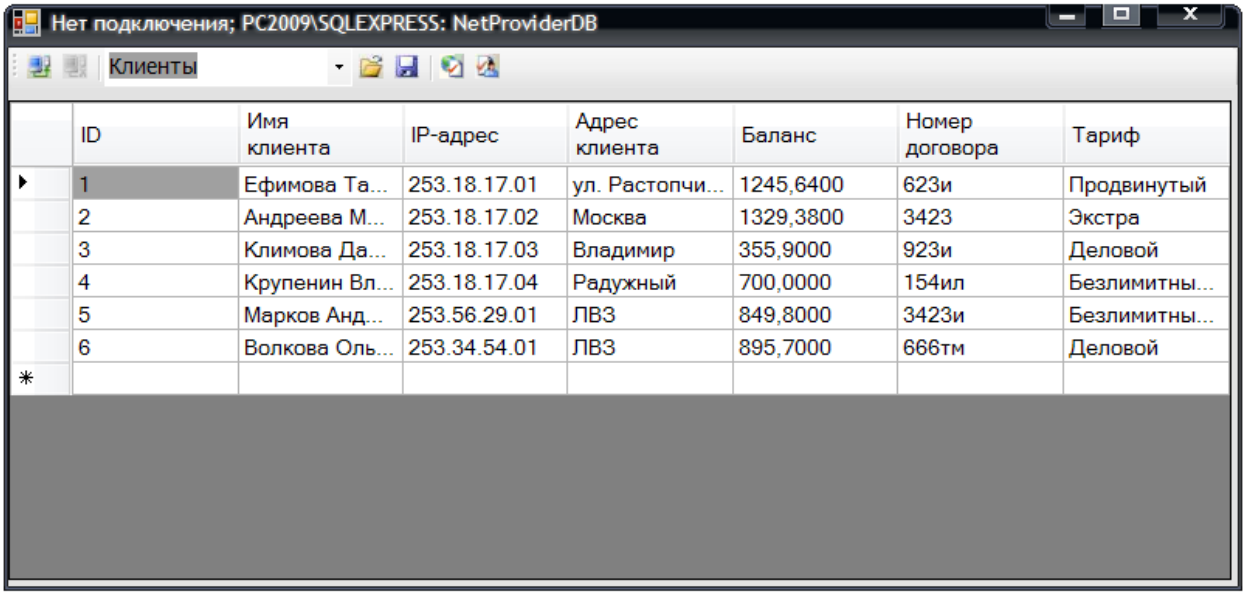


Рис. 18 Внешний вид формы соединения с сервером при выборе аутентификации SqlServer



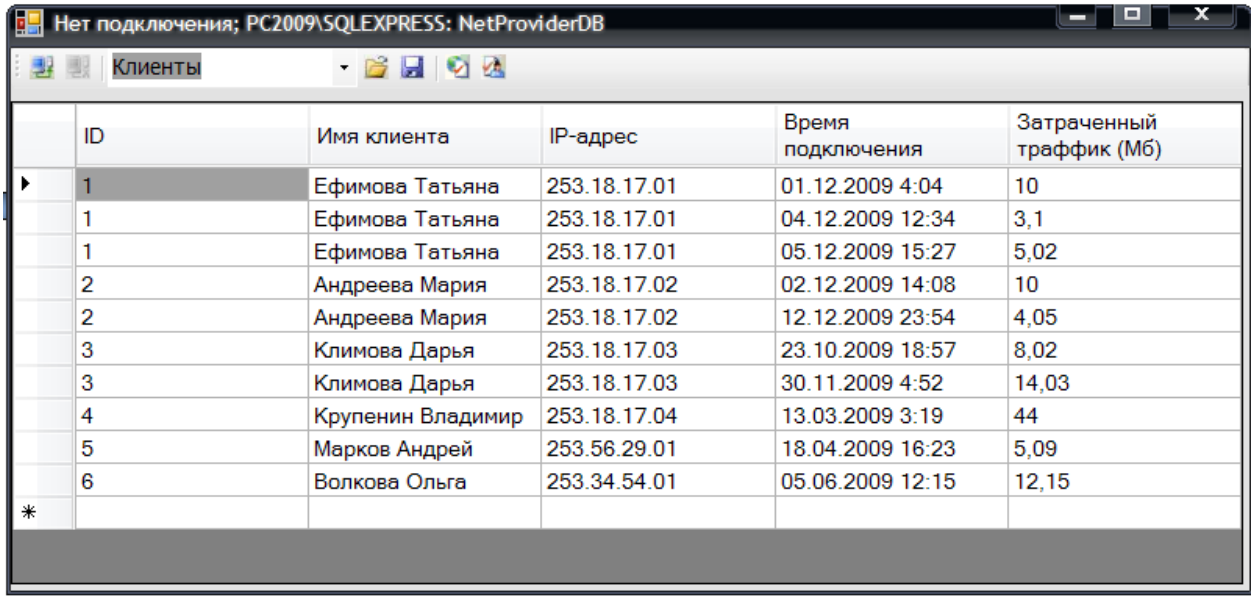
clientID	clientName	addr	treatyNum	ip	balance	tariff_id
1	Ефимова Та...	ул. Растопчи...	623и	253.18.17.01	1245,6400	2
2	Андреева М...	Москва	3423	253.18.17.02	1329,3800	4
3	Климова Да...	Владимир	923и	253.18.17.03	355,9000	3
4	Крупенин Вл...	Радужный	154ил	253.18.17.04	700,0000	5
5	Марков Анд...	ЛВЗ	3423и	253.56.29.01	849,8000	6
6	Волкова Оль...	ЛВЗ	666тм	253.34.54.01	895,7000	3

Рис. 19 Внешний вид главной формы пользовательского приложения при работе в подключенном режиме



ID	Имя клиента	IP-адрес	Адрес клиента	Баланс	Номер договора	Тариф
1	Ефимова Та...	253.18.17.01	ул. Растопчи...	1245,6400	623и	Продвинутый
2	Андреева М...	253.18.17.02	Москва	1329,3800	3423	Экстра
3	Климова Да...	253.18.17.03	Владимир	355,9000	923и	Деловой
4	Крупенин Вл...	253.18.17.04	Радужный	700,0000	154ил	Безлимитны...
5	Марков Анд...	253.56.29.01	ЛВЗ	849,8000	3423и	Безлимитны...
6	Волкова Оль...	253.34.54.01	ЛВЗ	895,7000	666тм	Деловой

Рис. 20 Внешний вид главной формы пользовательского приложения при работе в отключенном режиме



ID	Имя клиента	IP-адрес	Время подключения	Затраченный траффик (Мб)
1	Ефимова Татьяна	253.18.17.01	01.12.2009 4:04	10
1	Ефимова Татьяна	253.18.17.01	04.12.2009 12:34	3,1
1	Ефимова Татьяна	253.18.17.01	05.12.2009 15:27	5,02
2	Андреева Мария	253.18.17.02	02.12.2009 14:08	10
2	Андреева Мария	253.18.17.02	12.12.2009 23:54	4,05
3	Климова Дарья	253.18.17.03	23.10.2009 18:57	8,02
3	Климова Дарья	253.18.17.03	30.11.2009 4:52	14,03
4	Крупенин Владимир	253.18.17.04	13.03.2009 3:19	44
5	Марков Андрей	253.56.29.01	18.04.2009 16:23	5,09
6	Волкова Ольга	253.34.54.01	05.06.2009 12:15	12,15

Рис. 21 Внешний вид главной формы после запроса списка с историей соединений всех пользователей данного интернет-провайдера

9. Список литературы

1. Грубер М. «Понимание SQL» -1993. – 420 стр.
2. Сеппа Д. «Программирование на Microsoft ADO.NET 2.0» - Спб: Питер, 2007. – 784 стр.: ил. – ISBN 978-5-91180-686-6.
3. Ульман Л. «MySQL» - М.: ДМК Пресс, Спб: Питер, 2004. – 352 стр. – ISBN 5-94074-229-7.
4. Microsoft Corporation «Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCSE» - М.: «Русская редакция», 2001. – 704 стр.: ил. – ISBN 5-7502-0149-X

10. ПРИЛОЖЕНИЕ

CreateSchema.sql

```
/* ---------------------------------------------------------------------------- */
/* ------------------------------------ БД ------------------------------------ */
/* ---------------------------------------------------------------------------- */
/*IF  EXISTS (SELECT name FROM sys.databases WHERE name = N'NetProviderDB')
DROP DATABASE [NetProviderDB]
GO*/

-- создаем [заново]
/*CREATE DATABASE NetProviderDB
GO*/

USE NetProviderDB
GO

------------------------------------------------------------------------------
----------------------- Удаляем старое содержимое ----------------------------
------------------------------------------------------------------------------
-- таблицы

IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID('Payments'))
      DROP TABLE Payments
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID('Connections'))
      DROP TABLE Connections
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID('Clients'))
      DROP TABLE Clients
GO

IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID('Tariffes'))
      DROP TABLE Tariffes
GO

-- процедуры управления
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insertClient')
    DROP PROC insertClient
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insClient')
    DROP PROC insClient
GO

IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insertTariff')
    DROP PROC insertTariff
GO
```

```
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insertPayment')
    DROP PROC insertPayment
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insPayment')
    DROP PROC insPayment
GO

IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insertConnection')
    DROP PROC insertConnection


GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'insConnection')
    DROP PROC insConnection
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'changeTariff')
    DROP PROC changeTariff
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'changeTariffForID')
    DROP PROC changeTariffForID

GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'deleteClient')
    DROP PROC deleteClient
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'deleteTariff')
    DROP PROC deleteTariff
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'deletePayment')
    DROP PROC deletePayment
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'deleteConnection')
    DROP PROC deleteConnection

GO
-- процедуры статистики
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'TrafficOfClient')
    DROP PROC TrafficOfClient
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'P' AND NAME = 'InfoOfClient')
    DROP PROC InfoOfClient
GO


-- представления
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'V' AND NAME = 'ViewOfConnections')
    DROP VIEW ViewOfConnections
GO

IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'V' AND NAME = 'ViewOfPayments')
    DROP VIEW ViewOfPayments
GO

IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'V' AND NAME = 'ViewOfClients')
    DROP VIEW ViewOfClients
GO
IF EXISTS (SELECT * FROM sys.objects WHERE TYPE = 'V' AND NAME = 'ViewOfTariffes')
    DROP VIEW ViewOfTariffes
GO
```

```
-------------------------------------------------------------------------------
------------------------- Таблицы -----------------------------------------
-------------------------------------------------------------------------------

------------ Платежи ----------------
CREATE TABLE Payments
(
	-- идентификатор платежа - первичный ключ - кластерный индекс
	paymentID INT IDENTITY(1,1) PRIMARY KEY,



	-- дата платежа (поле должно быть заполнено)
	date DATETIME NOT NULL,
	-- сумма платежа (поле д.б. заполнено)
    summ MONEY NOT NULL,
	-- идентификатор плательщика (д.б. заполнен) - внешний ключ
	client_id INT NOT NULL
)
GO

-- некластерный индекс для столбца с датой платежа
CREATE INDEX DateOfPayment ON Payments (date)
GO
-- некластерный индекс для столбца с идентификатором клиента, совершившего платеж
CREATE INDEX PaymentOfClientID ON Payments (client_id)
GO

----- Подключения(сеансы связи) -----
CREATE TABLE Connections
(
	-- идентификатор сеанса - первичный ключ
	connID INT IDENTITY(1,1) PRIMARY KEY,
	-- дата подключения (поле должно быть заполнено)
	date DATETIME NOT NULL,
	-- кол-во скачанных Мб (поле д.б. заполнено)
    mbCount FLOAT NOT NULL,
	-- идентификатор пользователя (д.б. заполнен) - внешний ключ
	client_id INT NOT NULL
)
GO

CREATE INDEX DateOfConnection ON Connections (date)
GO
CREATE INDEX ConnectionOfClientID ON Connections (client_id)
GO


-------------- Клиенты --------------
CREATE TABLE Clients
(
	-- идентификатор клиента - первичный ключ - кластерный индекс
	clientID INT IDENTITY(1,1) PRIMARY KEY,
	-- имя клиента (поле должно быть заполнено)
	clientName NVARCHAR(30) NOT NULL,
	-- адрес клиента (необязательное поле)
    addr NVARCHAR(50) NOT NULL DEFAULT 'N/A',
	-- номер договора (д.б. введен и уникален) - некластерный индекс
    treatyNum NVARCHAR(10) NOT NULL UNIQUE,
	--  ip-адрес  (м.б.  не  назначен,  но  все  назначенные  д.б.  уникальны)  -
некластерный индекс
```

```
    ip NVARCHAR(15) UNIQUE,
       -- текущий баланс (д.б. заполнен)
    balance MONEY NOT NULL,
       -- идентификатор тарифного плана (д.б. заполнен) - внешний ключ
    tariff_id INT NOT NULL
)
GO

CREATE INDEX TariffOfClient ON Clients (tariff_id)
GO


/*CREATE INDEX BalanceOfClient ON Clients (balance)
GO**/
/*CREATE INDEX IPOfClient ON Clients (ip)
GO
CREATE INDEX TreatyNumOfClient ON Clients (treatyNum)
GO*/




-------------- Тарифы -----------------
CREATE TABLE Tariffes
(
       -- идентификатор тарифа - первичный ключ - кластерный индекс
       tariffID INT IDENTITY(1,1) PRIMARY KEY,
       -- название (поле должно быть заполнено и уникально) - некластерный индекс
       tariffName NVARCHAR(20) NOT NULL UNIQUE,
       -- скорость
       speed INT NOT NULL,
       -- стоимость 1мб траффика (поле д.б. заполнено)
    cost MONEY NOT NULL,
       -- предоплата (стоимость подключения тарифа)
       startCost MONEY NOT NULL
)
GO

-------------- Связи между таблицами ---------------------

--      Клиенты N --<>-- 1 Тарифы
ALTER TABLE Clients ADD FOREIGN KEY (tariff_id) REFERENCES Tariffes (tariffID)
GO
--  Подключения N --<>-- 1 Клиенты
ALTER TABLE Connections ADD FOREIGN KEY (client_id) REFERENCES Clients (clientID)
GO
--      Платежи N --<>-- 1 Клиенты
ALTER TABLE Payments ADD FOREIGN KEY (client_id) REFERENCES Clients (clientID)
GO



/* ---------------------------------------------------------------- */
/* ------------------- хранимые процедуры ------------------------- */
/* ---------------------------------------------------------------- */

/* ----------------- Добавление и удаление ------------------------ */

-- процедура добавления Клиента
-- аргументы:
--           @name         - имя клиента
```

```
--           @address     - физический адрес (м.б. NULL)
--           @ip                  - ip-адрес
--           @treatyNum   - номер договора
--           @tariffName  - название тарифа, выбранного при заключении контракта
-- возвращаемое значение:
--           идентификатор нового клиента (-1 - ошибка при выполнении процедуры)

CREATE PROCEDURE insertClient
      @name NVARCHAR(30),
      @address NVARCHAR(50),
      @ip NVARCHAR(15),

      @treatyNum NVARCHAR (10),
      @balance FLOAT,
      @tariffName NVARCHAR(20)
AS
BEGIN
      SET NOCOUNT ON

      -- узнаем идентификатор выбранного тарифа
      DECLARE @tariff_id INT
      SET  @tariff_id  =  (SELECT  tariffID  FROM  Tariffes  WHERE  tariffName  =
@tariffName)

      -- проверяем, существует ли тариф с таким названием
      IF @@ROWCOUNT = 0
      BEGIN
            PRINT 'Error: Tariff with such name is not exists'
            RETURN (-1)
      END

      BEGIN TRY
            -- пробуем вставить новую запись
            INSERT INTO Clients(clientName, addr, ip, treatyNum, tariff_id, balance)
                  VALUES(@name, @address, @ip, @treatyNum, @tariff_id, @balance)
            PRINT 'Client was inserted successfully'
            RETURN SCOPE_IDENTITY()
      END TRY
      BEGIN CATCH
            -- ловим исключения при добавлении записи и отображаем инфо об ошибке
            PRINT ERROR_MESSAGE()
            RETURN (-1)
      END CATCH
END
GO

-- процедура добавления Клиента
-- то же самое, но учитывается не название, а идентификатор тарифа

CREATE PROCEDURE insClient
      @name NVARCHAR(30),
      @address NVARCHAR(50),
      @ip NVARCHAR(15),
      @treatyNum NVARCHAR (10),
      @balance FLOAT,
      @tariff_id INT
AS
BEGIN
      SET NOCOUNT ON

      IF ((SELECT COUNT(*) FROM Tariffes WHERE tariffID = @tariff_id) = 0)
```

```
	BEGIN
		PRINT 'Error: Tariff with such id is not exists'
		RETURN (-1)
	END

	BEGIN TRY
		INSERT INTO Clients(clientName, addr, ip, treatyNum, tariff_id, balance)
			VALUES(@name, @address, @ip, @treatyNum, @tariff_id, @balance)
		PRINT 'Client was inserted successfully'
		RETURN SCOPE_IDENTITY()


	END TRY
	BEGIN CATCH
		-- ловим исключения при добавлении записи и отображаем инфо об ошибке
		PRINT ERROR_MESSAGE()
		RETURN (-1)
	END CATCH
END
GO

-----------------------------------------------------

-- процедура добавления Тарифа
-- аргументы:
--		@name		- название
--		@cost		- цена 1Мб трафика
-- возвращаемое значение:
--		идентификатор нового тарифа (-1 - ошибка при выполнении процедуры)

CREATE PROCEDURE insertTariff
	@name NVARCHAR(20),
	@speed INT,
	@cost MONEY,
	@startCost MONEY
AS
BEGIN
	SET NOCOUNT ON

	IF((SELECT COUNT(*) FROM Tariffes WHERE tariffName = @name) > 0)
	BEGIN
		PRINT 'Tariff with such name already exists'
		RETURN (-1)
	END

	BEGIN TRY
		-- пробуем вставить новую запись
		INSERT INTO Tariffes (tariffName, speed, cost, startCost)
			VALUES (@name, @speed, @cost, @startCost)
		PRINT 'Tariff was inserted successfully'
		RETURN SCOPE_IDENTITY()
	END TRY
	BEGIN CATCH
		-- ловим исключения при добавлении записи и отображаем инфо об ошибке
		PRINT ERROR_MESSAGE()
		RETURN (-1)
	END CATCH
END
GO

-------------------------------------------------
```

```
-- процедура добавления Платежа
-- аргументы:
--           @treatyNum         - номер договора плательщика
--           @date              - дата
--           @summ              - сумма платежа
-- возвращаемое значение:
--           идентификатор нового платежа (-1 - ошибка при выполнении процедуры)

CREATE PROCEDURE insertPayment


      @treatyNum NVARCHAR(10),
      @date DATETIME,
      @summ MONEY
AS
BEGIN
      SET NOCOUNT ON

      DECLARE @client_id INT
      SET @client_id = (SELECT clientID FROM Clients WHERE treatyNum = @treatyNum)

      -- проверяем, существует ли клиент с таким номером договора
      IF @@ROWCOUNT = 0
      BEGIN
            PRINT 'Error: Client with such treaty number is not exists'
            RETURN (-1)
      END

      BEGIN TRY
            -- пробуем вставить новую запись
            INSERT INTO Payments(client_id, date, summ)
                  VALUES(@client_id, @date, @summ)
            PRINT 'Payment was inserted successfully'
            RETURN SCOPE_IDENTITY()
      END TRY
      BEGIN CATCH
            -- ловим исключения при добавлении записи и отображаем инфо об ошибке
            PRINT ERROR_MESSAGE()
            RETURN (-1)
      END CATCH
END
GO

-- процедура добавления Платежа
-- то же самое, но учитываем идентификатор клиента

CREATE PROCEDURE insPayment
      @client_id INT,
      @date DATETIME,
      @summ MONEY
AS
BEGIN
      SET NOCOUNT ON

      IF((SELECT COUNT(*) FROM Clients WHERE clientID = @client_id) = 0)
      BEGIN
            PRINT 'Error: Client with such id is not exists'
            RETURN (-1)
      END
```

```
	BEGIN TRY
		-- пробуем вставить новую запись
		INSERT INTO Payments(client_id, date, summ)
			VALUES(@client_id, @date, @summ)
		PRINT 'Payment was inserted successfully'
		RETURN SCOPE_IDENTITY()
	END TRY
	BEGIN CATCH
		-- ловим исключения при добавлении записи и отображаем инфо об ошибке
		PRINT ERROR_MESSAGE()
		RETURN (-1)

	END CATCH
END
GO

-------------------------------------------------

-- процедура добавления Подключения
-- аргументы:
--		@ip			- ip-адрес подключившегося клиента
--		@date		- дата
--		@summ		- сумма платежа
-- возвращаемое значение:
--		идентификатор нового платежа (-1 - ошибка при выполнении процедуры)

CREATE PROCEDURE insertConnection
	@ip NVARCHAR(15),
	@date DATETIME,
	@mbCount FLOAT
AS
BEGIN
	SET NOCOUNT ON

	DECLARE @client_id INT
	SET @client_id = (SELECT clientID FROM Clients WHERE ip = @ip)

	-- проверяем, существует ли клиент с таким id
	IF @@ROWCOUNT = 0
	BEGIN
		PRINT 'Error: Client with such ip-address is not exists'
		RETURN (-1)
	END

	BEGIN TRY
		-- пробуем вставить новую запись
		INSERT INTO Connections(client_id, date, mbCount)
			VALUES(@client_id, @date, @mbCount)
		PRINT 'Connection was inserted successfully'
		RETURN SCOPE_IDENTITY()
	END TRY
	BEGIN CATCH
		-- ловим исключения при добавлении записи и отображаем инфо об ошибке
		PRINT ERROR_MESSAGE()
		RETURN (-1)
	END CATCH
END
GO

-- процедура добавления Подключения
-- то же самое, но учитываем идентификатор клиента
```

```
CREATE PROCEDURE insConnection
	@client_id INT,
	@date DATETIME,
	@mbCount FLOAT
AS
BEGIN
	SET NOCOUNT ON

	IF((SELECT COUNT(*) FROM Clients WHERE clientID = @client_id) = 0)
	BEGIN
```



```
		PRINT 'Error: Client with such id is not exists'
		RETURN (-1)
	END

	BEGIN TRY
		INSERT INTO Connections(client_id, date, mbCount)
			VALUES(@client_id, @date, @mbCount)
		PRINT 'Connection was inserted successfully'
		RETURN SCOPE_IDENTITY()
	END TRY
	BEGIN CATCH
		-- ловим исключения при добавлении записи и отображаем инфо об ошибке
		PRINT ERROR_MESSAGE()
		RETURN (-1)
	END CATCH
END
GO




--------------------------------------------------

-- процедура удаления подключения
-- аргументы:
--		@connID -  идентификатор удаляемого подключения
-- возвращаемое значение:
--		0 - нет ошибки / -1 - ошибка при удалении

CREATE PROCEDURE deleteConnection
	@connID INT
AS
BEGIN
	SET NOCOUNT ON

	IF((SELECT COUNT(*) FROM Connections WHERE connID = @connID) > 0)
	BEGIN
		DELETE FROM Connections WHERE connID = @connID
		PRINT 'Connection was deleted'
		RETURN 0
	END

	PRINT 'Connection with such ID is not exists'
	RETURN (-1)
END
GO

-- процедура удаления платежа
```

```
-- аргументы:
--           @paymentID -  идентификатор удаляемого подключения
-- возвращаемое значение:
--           0 - нет ошибки / -1 - ошибка при удалении

CREATE PROCEDURE deletePayment
      @paymentID INT
AS
BEGIN
      SET NOCOUNT ON



      IF((SELECT COUNT(*) FROM Payments WHERE paymentID = @paymentID) > 0)
      BEGIN
            DELETE FROM Payments WHERE paymentID = @paymentID
            PRINT 'Payment was deleted'
            RETURN 0
      END

      PRINT 'Payment with such ID is not exists'
      RETURN (-1)
END
GO

-- процедура удаления клиента
-- аргументы:
--           @clientID -  идентификатор удаляемого клиента
-- возвращаемое значение:
--           0 - нет ошибки / -1 - ошибка при удалении

CREATE PROCEDURE deleteClient
      @clientID INT
AS
BEGIN
      SET NOCOUNT ON

      IF((SELECT COUNT(*) FROM Clients WHERE clientID = @clientID) > 0)
      BEGIN
            DELETE FROM Clients WHERE clientID = @clientID
            PRINT 'Client was deleted'
            RETURN 0
      END

      PRINT 'Client with such ID is not exists'
      RETURN (-1)
END
GO

-- процедура удаления тарифа
-- аргументы:
--           @tariffID -  идентификатор удаляемого тарифа
-- возвращаемое значение:
--           0 - нет ошибки / -1 - ошибка при удалении

CREATE PROCEDURE deleteTariff
      @tariffID INT
AS
BEGIN
      SET NOCOUNT ON
```

```
	IF((SELECT COUNT(*) FROM Tariffes WHERE tariffID = @tariffID) > 0)
	BEGIN
		DELETE FROM Tariffes WHERE tariffID = @tariffID
		PRINT 'Tariff was deleted'
		RETURN 0
	END

	PRINT 'Tariff with such ID is not exists'
	RETURN (-1)



END
GO

/* ----------------------------- Другие операции ---------------------------- */

-- процедура изменения клиентом тарифного плана
-- аргументы:
--		@treatyNum		- номер договора клиента
--		@tariffName		- название нового тарифа
-- возвращаемое значение:
--		идентификатор нового тарифного плана (-1 - ошибка)

CREATE PROCEDURE changeTariff
	@treatyNum NVARCHAR(10),
	@tariffName NVARCHAR(20)
AS
BEGIN
	SET NOCOUNT ON

	DECLARE @tariff_id INT
	SET @tariff_id = (SELECT tariffID FROM Tariffes WHERE tariffName = 
@tariffName)

	-- проверяем, существует ли тариф с таким названием
	IF @@ROWCOUNT = 0
	BEGIN
		PRINT 'Error: Tariff with such name is not exists'
		RETURN (-1)
	END

	BEGIN TRY
		UPDATE Clients SET tariff_id = @tariff_id WHERE treatyNum = @treatyNum
		PRINT 'Tariff changed successfully'
		RETURN @tariff_id
	END TRY
	BEGIN CATCH
		PRINT ERROR_MESSAGE()
		RETURN (-1)
	END CATCH
END
GO

-- процедура изменения клиентом тарифного плана
-- аргументы:
--		@clientID		- идентификатор клиента
--		@tariffID		- идентификатор тарифа
-- возвращаемое значение:
--		идентификатор нового тарифного плана (-1 - ошибка)
```

```
CREATE PROCEDURE changeTariffForID
	@clientID INT,
	@tariffID INT
AS
BEGIN
	SET NOCOUNT ON

	IF((SELECT COUNT(*) FROM Tariffes WHERE tariffID = @tariffID) = 0)
	BEGIN
		PRINT 'Error: Tariff with such id is not exists'


		RETURN (-1)
	END

	IF((SELECT COUNT(*) FROM Clients WHERE clientID = @clientID) = 0)
	BEGIN
		PRINT 'Error: Client with such id is not exists'
		RETURN (-1)
	END

	BEGIN TRY
		UPDATE Clients SET tariff_id = @tariffID WHERE clientID = @clientID
		PRINT 'Tariff changed successfully'
		RETURN @tariffID
	END TRY
	BEGIN CATCH
		PRINT ERROR_MESSAGE()
		RETURN (-1)
	END CATCH
END
GO

-----------------------------------------------------

-- триггер, связанный с пополнением клиентом своего счета

CREATE TRIGGER increaseBalance
ON Payments
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON

	UPDATE Clients SET balance = balance +
	(
		SELECT summ
		FROM Inserted
		WHERE Inserted.client_id = Clients.clientID
	)
	WHERE clientId IN (SELECT client_id FROM Inserted)
END
GO

-- триггер, связанный с использованием клиентом траффика

CREATE TRIGGER decreaseBalance
ON Connections
AFTER INSERT
AS
BEGIN
```

```
	SET NOCOUNT ON

	UPDATE Clients SET balance = balance -
	(
		SELECT i.mbCount * t.cost
		FROM Inserted i
			INNER JOIN Clients c ON (c.clientID = i.client_id)
			INNER JOIN Tariffes t ON (t.tariffID = c.tariff_id)
	)
	WHERE clientId IN (SELECT client_id FROM Inserted)


END
GO

-- триггер, связанный с изменением клиентом своего тарифного плана

CREATE TRIGGER checkTariffchange
ON Clients
AFTER UPDATE
AS
BEGIN
	SET NOCOUNT ON

	-- если изменился идентификатор тарифа
	IF UPDATE (tariff_id)
	BEGIN
		-- снимаем с клиента разницу в стоимости нового и текущего тарифного
планов
		UPDATE Clients SET balance = balance -
		(
			(SELECT startCost FROM Tariffes WHERE tariffID IN (SELECT
tariff_id FROM Inserted)) -
			(SELECT startCost FROM Tariffes WHERE tariffID IN (SELECT
tariff_id FROM Deleted))
		)
		WHERE clientId IN (SELECT clientID FROM Inserted)

		-- если в итоге баланс ушел в минус, отменяем транзакцию
		IF ((SELECT balance FROM Clients WHERE clientID IN (SELECT
/*tariff_id*/clientID FROM Inserted)) < 0)
			ROLLBACK TRANSACTION
	END
END
GO

/* ------------------------- Статистика ----------------------------- */

-- процедура подсчета суммарного потребления траффика заданным клиентом
--			в течении заданного периода времени
-- аргументы:
--		@clientID		- идентификатор клиента
--		@dateBegin		- дата и время начала подсчета
--		@dateEnd		- дата и время конца подсчета
--		@count			- кол-во траффика (OUTPUT)
-- возвращаемое значение:
--		нет

CREATE PROCEDURE TrafficOfClient
	@clientID INT,
	@dateBegin DATETIME,
```

```
	@dateEnd DATETIME,
	@count FLOAT OUTPUT
AS
BEGIN
	SET NOCOUNT ON

	IF((SELECT COUNT(*) FROM Clients WHERE clientID = @clientID) = 0)
	BEGIN
		PRINT 'Error: Client with such id is not exists'
		RETURN (-1)
	END

	SET @count = (SELECT SUM(mbCount) FROM Connections
				WHERE client_id = @clientID AND date >= @dateBegin AND
date <= @dateEnd)
END
GO

-- процедура вывода информации о заданным клиенте
-- аргументы:
--		@clientID		- идентификатор клиента
-- возвращаемое значение:
--		нет

CREATE PROCEDURE InfoOfClient
	@clientID INT
AS
BEGIN

	IF((SELECT COUNT(*) FROM Clients WHERE clientID = @clientID) = 0)
	BEGIN
		PRINT 'Error: Client with such id is not exists'
		RETURN (-1)
	END

	SELECT c.clientName as 'Имя клиента', c.addr as 'Адрес клиента',
		c.treatyNum as 'Номер договора', c.ip as 'IP-адрес',
		t.tariffName as 'Тариф', c.balance as 'Баланс',
		SUM(conn.mbCount) as 'Суммарный траффик'
	FROM Clients c
		INNER JOIN Tariffes t ON (c.tariff_id = t.tariffID)
		INNER JOIN Connections conn ON (client_id = @clientID)
	WHERE clientID = @clientID
	GROUP BY clientName, addr, treatyNum, ip, tariffName, balance
END
GO

/* ----------------------------------------------------------------
   --------------------- Представления ----------------------------
   ---------------------------------------------------------------- */

-- представление всех сеансов подключений всех клиентов
CREATE VIEW ViewOfConnections
AS
	SELECT cl.clientID as 'ID', cl.clientName as 'Имя клиента', cl.ip as 'IP-
адрес',
		conn.date as 'Время подключения', conn.mbCount as 'Затраченный
траффик (Мб)'
	FROM Connections conn
		INNER JOIN Clients cl ON (cl.clientID = conn.client_id)
	GROUP BY clientID, clientName, ip, date, mbCount
GO
```

```
-- представление всех пополнений счета всех клиентов
CREATE VIEW ViewOfPayments
AS
	SELECT c.clientName as 'Имя клиента', c.treatyNum as 'Номер договора',
		p.date as 'Время платежа', p.summ as 'Сумма'
	FROM Payments p
		INNER JOIN Clients c ON (c.clientID = p.client_id)
	GROUP BY clientName, treatyNum, date, summ
GO

-- представление данных о клиентах
CREATE VIEW ViewOfClients
AS
	SELECT c.clientID as 'ID', c.clientName as 'Имя клиента', c.addr as 'Адрес
клиента', c.balance as 'Баланс',
		c.treatyNum as 'Номер договора', c.ip as 'IP-адрес', t.tariffName as
'Тариф'
	FROM Clients c
		INNER JOIN Tariffes t ON (c.tariff_id = t.tariffID)
GO

-- представление всех тарифных планов с указанием кол-ва клиентов
CREATE VIEW ViewOfTariffes
AS
	SELECT t.tariffName as 'Название тарифа', t.cost as 'Стоимость 1 Мб траффика',
		t.startCost as 'Стоимость активации', COUNT(c.tariff_id) as 'Число
абонентов'
	FROM Tariffes t
		LEFT JOIN Clients c ON (c.tariff_id = t.tariffID)
	GROUP BY cost, startCost, tariffName
GO
```

PopulateData.sql

```
USE NetProviderDB
GO

-- удаляем старые данные
DELETE FROM Payments
DELETE FROM Connections
DELETE FROM Clients
DELETE FROM Tariffes
GO

-- заполняем линейку тарифных планов
EXECUTE insertTariff 'Базовый',               128,  4.0, 100
EXECUTE insertTariff 'Продвинутый',           256,  3.0, 300
EXECUTE insertTariff 'Деловой',               512,  2.0, 500
EXECUTE insertTariff 'Экстра',                1024, 1.0, 1000
EXECUTE insertTariff 'Безлимитный-512',  512,  0.0, 4000
EXECUTE insertTariff 'Безлимитный-1024', 1024, 0.0, 6000
GO

SELECT * FROM Tariffes
GO

-- вносим в базу первых клиентов
```

```
EXECUTE insertClient 'Ефимова Татьяна',        'ул. Растопчина', '253.18.17.01',
'623и',  600.0, 'Продвинутый'
EXECUTE insertClient 'Андреева Мария',   'Москва',    '253.18.17.02', '3423',  800.0,
'Экстра'
EXECUTE insertClient 'Климова Дарья', 'Владимир',           '253.18.17.03', '923и',
400.0, 'Деловой'
EXECUTE insertClient 'Крупенин Владимир', 'Радужный',       '253.18.17.04', '154ил',
200.0,  'Безлимитный-512'
EXECUTE insertClient 'Марков Андрей',    'ЛВЗ',                    '253.56.29.01',
'3423и', 100.0,  'Безлимитный-1024'
EXECUTE insertClient 'Волкова Ольга',          'ЛВЗ',

'253.34.54.01', '666тм', 220.0,  'Деловой'

GO

SELECT * FROM Clients
GO

-- вносим инфо о подключениях
EXECUTE insertConnection '253.18.17.01', '2010-12-01 04:04:04', 10.00
EXECUTE insertConnection '253.18.17.01', '2010-12-04 12:34:00', 3.1
EXECUTE insertConnection '253.18.17.01', '2010-12-05 15:27:36', 5.02
EXECUTE insertConnection '253.18.17.02', '2010-12-02 14:08:12', 10.0
EXECUTE insertConnection '253.18.17.02', '2010-12-12 23:54:30', 4.05
EXECUTE insertConnection '253.18.17.03', '2010-10-23 18:57:44', 8.02
EXECUTE insertConnection '253.18.17.03', '2010-11-30 04:52:00', 14.03
EXECUTE insertConnection '253.18.17.04', '2010-03-13 03:19:28', 44.0
EXECUTE insertConnection '253.56.29.01', '2010-04-18 16:23:06', 5.09
EXECUTE insertConnection '253.34.54.01', '2010-06-05 12:15:17', 12.15
GO

SELECT * FROM Connections
GO
SELECT * FROM Clients
GO

-- вносим инфо о платежах
EXECUTE insertPayment '623и',  '2010-10-01 12:14:32', 100.0
EXECUTE insertPayment '623и',  '2010-08-19 23:48:49', 150.00
EXECUTE insertPayment '623и',  '2010-06-04 15:16:46', 450.00
EXECUTE insertPayment '3423',  '2010-11-18 21:26:27', 543.43
EXECUTE insertPayment '154ил', '2010-09-04 14:19:58', 500.00
EXECUTE insertPayment '3423и', '2010-07-15 06:06:14', 654.57
EXECUTE insertPayment '3423и', '2010-06-21 18:21:12', 95.23
EXECUTE insertPayment '666тм', '2010-09-12 04:02:05', 700.00
GO

SELECT * FROM Payments
GO
SELECT * FROM Clients
GO
```

Test01.sql

```
USE NetProviderDB
GO

-- добавляем в БД новый тарифный план
EXECUTE insertTariff 'Top Speed', 4096, 0.5, 1000
GO
```

```
SELECT * FROM Tariffes
GO
```

Test02.sql

```
USE NetProviderDB
GO

-- добавляем в БД нового клиента
EXECUTE insertClient 'Владимир Ильич', 'Мавзолей', '273.133.212.201', '1917ил',


 1200.0, 'Базовый'
GO
SELECT * FROM Clients
GO
```

Test03.sql

```
/* --- изменение тарифного плана для клиента --- */

-- изменяем тариф клиенту с номером договора '1917ил' на 'Top Speed'

USE NetProviderDB
GO
EXECUTE changeTariff '1917ил', 'Top Speed'
GO
SELECT * FROM Clients
GO
```

Test04.sql

```
/* --- добавление соединения --- */

USE NetProviderDB
GO

EXECUTE insertConnection '273.133.212.201', '2009-12-12 04:04:04', 54.87

SELECT * FROM Connections
GO
SELECT * FROM Clients
GO
```

Test05.sql

```
/* --- добавление платежа --- */

USE NetProviderDB
GO

EXECUTE insertPayment '1917ил', '2009-11-13 12:14:32', 100.0

SELECT * FROM Payments
GO
SELECT * FROM Clients
GO
```

Test06.sql

```
/* --- отображение статистики по тарифам --- */
```

```
USE NetProviderDB
GO

SELECT * FROM ViewOfTariffes
GO
```

Test07.sql

```
/* --- отображение статистики по клиентам --- */



USE NetProviderDB
GO

SELECT * FROM ViewOfClients
GO
```

Test08.sql

```
/* --- отображение статистики по подключениям --- */

USE NetProviderDB
GO
SELECT * FROM ViewOfConnections
GO
```

Test09.sql

```
/* --- отображение статистики по платежам --- */

USE NetProviderDB
GO
SELECT * FROM ViewOfPayments
GO
```

Test10.sql

```
/* --- подсчет суммарного потребления траффика заданным клиентом в течении заданного
периода времени --- */

USE NetProviderDB
GO

DECLARE @count1 FLOAT,
            @clientID1 INT,
            @dateBegin1 DATETIME,
            @dateEnd1 DATETIME

SET @clientID1 = 2
SET @dateBegin1 = '2009-12-04 12:34:00'
SET @dateEnd1 = '2009-12-05 15:27:36'

EXECUTE TrafficOfClient
@clientID = @clientID1,
@dateBegin = @dateBegin1,
@dateEnd = @dateEnd1,
@count = @count1 OUTPUT

PRINT 'Summary traffic for client with ID ' + CAST(@clientID1 as NVARCHAR) +
```

```
            '  between  ('  +  CAST(@dateBegin1  as  NVARCHAR)  +  ')  and  ('  +
CAST(@dateEnd1 as NVARCHAR) +
            ') is ' + CAST(@count1 as NVARCHAR)
GO
```

Test11.sql

```
/* --- вывод информации о клиенте с заданным ID --- */

USE NetProviderDB
GO

EXECUTE InfoOfClient
@clientID = 8
GO
```

ConnectionsForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace NetProvider
{
  public partial class ConnectionForm : Form
  {
    // тип аутентификации
    private enum AuthType { Windows, SqlServer };
    private AuthType authType;

    // ссылка на главную форму (для обратной связи)
    private MainForm mainForm = null;

    // true, если пользователь нажал на acceptBtn,
    // если он пытается закрыть форму любым другим способом - false.
    // соотв. если true, то пытаемся выполнить подключение
    private bool accept;
    // ---------------------------------------------------------------------

    public ConnectionForm(MainForm mainForm)
    {
        InitializeComponent();
        this.mainForm = mainForm;
    }

    private void ConnectionForm_Shown(object sender, EventArgs e)
    {
      serverTBx.Text = "PC2009\\SQLEXPRESS";//"amd-
s02.laser.hq.corp.vlsu.ru\\STUDENT";
      dbNameTBx.Text = "NetProviderDB";

      authType = AuthType.Windows;
      authTBx.Text = "Windows";
```

```
  loginTBx.Text = "";
  loginTBx.Enabled = false;
  passTBx.Text = "";
  passTBx.Enabled = false;

  accept = false;
}

// подключение отменено пользователем
private void cancelBtn_Click(object sender, EventArgs e)


{
  accept = false;   // закрытие формы без подключения
  Close();
}

// кнопка "подключение"
private void acceptBtn_Click(object sender, EventArgs e)
{
  accept = true;    // закрытие формы с попыткой подключения
  Close();          // закрываем форму
}

private void authBtn_Click(object sender, EventArgs e)
{
  if (authType == AuthType.Windows)
  {
    authType = AuthType.SqlServer;
    authTBx.Text = "SqlServer";
    loginTBx.Text = "pmall";
    passTBx.Text = "0987PaSS!$";
    loginTBx.Enabled = true;
    passTBx.Enabled = true;
  else
  {
    authType = AuthType.Windows;
    authTBx.Text = "Windows";
    loginTBx.Enabled = false;
    passTBx.Enabled = false;
  }
}

// форма "собирается" закрыться
private void ConnectionForm_FormClosing(object sender, FormClosingEventArgs e)
{
  if (accept)
  {
    // если пользователь нажал на кнопку acceptBtn,
    // пытаемся выполнить подключение к БД
    // и передать ссылку на созданное подключение
    // главной форме

    // строка подключения
    SqlConnectionStringBuilder connStrBuilder =
      new SqlConnectionStringBuilder();
    connStrBuilder.DataSource = serverTBx.Text;
    connStrBuilder.InitialCatalog = dbNameTBx.Text;

    if (authType == AuthType.Windows)
    {
```

```
          connStrBuilder.IntegratedSecurity = true;
        }
        else
        {
          connStrBuilder.IntegratedSecurity = false;
          connStrBuilder.UserID = loginTBx.Text;
          connStrBuilder.Password = passTBx.Text;
        }




        // объект подключения
        SqlConnection sqlConn =
          new SqlConnection(connStrBuilder.ConnectionString);

        try
        {
          // подключаемся
          sqlConn.Open();
          MessageBox.Show("Подключение успешно выполнено");
          // передаем ссылку на подключение главной форме
          mainForm.InitConnection(sqlConn);

          // форма закрывается
        }
        catch (SqlException ex)
        {
          // если пытались подключиться, но возникла ошибка,
          // выводим сообщение и отменяем закрытие формы
          //MessageBox.Show("Ошибка подключения к базе данных");
          //MessageBox.Show(ex.ToString());
          e.Cancel = true;
          accept = false;
        }
        catch (Exception ex)
        {
          MessageBox.Show("Неизвестная ошибка при подключения к базе данных\n"
                          + ex.Message);
          e.Cancel = true;
          accept = false;
        }
      }
      else
      {
        // если пользователь закрывает форму любым другим способом,
        // объект подключения не создается
        mainForm.InitConnection(null);

        // форма закрывается
      }
    }
  }
}
```

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace NetProvider
{
  public partial class MainForm : Form
  {



    // объект подключения
    private SqlConnection sqlConn = null;

    // таблицы
    private String[] tableNames = { "Clients", "Tariffes", "Payments",
"Connections"};

    private DataTable clientsTable = null;
    private DataTable tariffesTable = null;
    private DataTable paymentsTable = null;
    private DataTable connectionsTable = null;

    // адаптер для таблиц
    private SqlDataAdapter clientsAdapter = null;
    private SqlDataAdapter tariffesAdapter = null;
    private SqlDataAdapter paymentsAdapter = null;
    private SqlDataAdapter connectionsAdapter = null;

    // значение, получаемое из формы InputForm - служит параметром запросов по
статистике
    private Int32 value;

    // ---------------------------------------------------------------------

    public MainForm()
    {
      InitializeComponent();

      clientsTable = new DataTable(tableNames[0]);
      tariffesTable = new DataTable(tableNames[1]);
      paymentsTable = new DataTable(tableNames[2]);
      connectionsTable = new DataTable(tableNames[3]);

      dataSet.Tables.Add(tariffesTable);
      dataSet.Tables.Add(clientsTable);
      dataSet.Tables.Add(paymentsTable);
      dataSet.Tables.Add(connectionsTable);

      value = -1;
    }

    // ---------------------------------------------------------------------

    // загрузка формы
    private void MainForm_Load(object sender, EventArgs e)
    {
      // пытаемся подключиться к БД
      connectBtn_Click(sender, e);
    }
```

```
    // функция инициализации объекта подключения (д.б. вызвана формой аутентификации)
    public void InitConnection(SqlConnection sqlConn)
    {
      this.sqlConn = sqlConn;
    }

    // событие "форма закрылась"
    private void MainForm_FormClosed(object sender, FormClosedEventArgs e)
    {


      // если подключение было установлено - закрываем его
      if (sqlConn != null) sqlConn.Close();
    }

    // кнопка "подключиться к БД"
    private void connectBtn_Click(object sender, EventArgs e)
    {
      // передаем ссылку на себя в форму подключения
      ConnectionForm connForm = new ConnectionForm(this);
      connForm.ShowDialog();

      // если подключились
      if (sqlConn != null)
      {
        try
        {
          dataSet.Clear();
          dataSet.DataSetName = sqlConn.Database;
          dataSet.SchemaSerializationMode =
            SchemaSerializationMode.IncludeSchema;

          // SELECT
          tariffesAdapter = new SqlDataAdapter("SELECT * FROM Tariffes", sqlConn);
          tariffesAdapter.FillSchema(dataSet, SchemaType.Source, "Tariffes");

          clientsAdapter = new SqlDataAdapter("SELECT * FROM Clients", sqlConn);
          clientsAdapter.FillSchema(dataSet, SchemaType.Source, "Clients");

          paymentsAdapter = new SqlDataAdapter("SELECT * FROM Payments", sqlConn);
          paymentsAdapter.FillSchema(dataSet, SchemaType.Source, "Payments");

          connectionsAdapter = new SqlDataAdapter("SELECT * FROM Connections",
sqlConn);
          connectionsAdapter.FillSchema(dataSet, SchemaType.Source, "Connections");

          SqlParameter p;

          // INSERT
          tariffesAdapter.InsertCommand = new SqlCommand("insertTariff", sqlConn);
          tariffesAdapter.InsertCommand.CommandType = CommandType.StoredProcedure;
          p = tariffesAdapter.InsertCommand.Parameters.Add("@name",
SqlDbType.NVarChar, 0, "tariffName");
          p = tariffesAdapter.InsertCommand.Parameters.Add("@speed", SqlDbType.Int,
0, "speed");
          p = tariffesAdapter.InsertCommand.Parameters.Add("@cost", SqlDbType.Money,
0, "cost");
          p = tariffesAdapter.InsertCommand.Parameters.Add("@startCost",
SqlDbType.Money, 0, "startCost");
```

```
          clientsAdapter.InsertCommand = new SqlCommand("insClient", sqlConn);
          clientsAdapter.InsertCommand.CommandType = CommandType.StoredProcedure;
          p = clientsAdapter.InsertCommand.Parameters.Add("@name",
SqlDbType.NVarChar, 0, "clientName");
          p = clientsAdapter.InsertCommand.Parameters.Add("@address",
SqlDbType.NVarChar, 0, "addr");
          p = clientsAdapter.InsertCommand.Parameters.Add("@ip", SqlDbType.NVarChar,
0, "ip");
          p = clientsAdapter.InsertCommand.Parameters.Add("@treatyNum",
SqlDbType.NVarChar, 0, "treatyNum");
          p = clientsAdapter.InsertCommand.Parameters.Add("@balance",


SqlDbType.Money, 0, "balance");
          p = clientsAdapter.InsertCommand.Parameters.Add("@tariff_id",
SqlDbType.Int, 0, "tariff_id");

          paymentsAdapter.InsertCommand = new SqlCommand("insPayment", sqlConn);
          paymentsAdapter.InsertCommand.CommandType = CommandType.StoredProcedure;
          p = paymentsAdapter.InsertCommand.Parameters.Add("@client_id",
SqlDbType.Int, 0, "client_id");
          p = paymentsAdapter.InsertCommand.Parameters.Add("@date",
SqlDbType.NVarChar, 0, "date");
          p = paymentsAdapter.InsertCommand.Parameters.Add("@summ", SqlDbType.Float,
0, "summ");

          connectionsAdapter.InsertCommand = new SqlCommand("insConnection",
sqlConn);
          connectionsAdapter.InsertCommand.CommandType = CommandType.StoredProcedure;
          p = connectionsAdapter.InsertCommand.Parameters.Add("@client_id",
SqlDbType.Int, 0, "client_id");
          p = connectionsAdapter.InsertCommand.Parameters.Add("@date",
SqlDbType.NVarChar, 0, "date");
          p = connectionsAdapter.InsertCommand.Parameters.Add("@mbCount",
SqlDbType.Float, 0, "mbCount");

          // DELETE
          tariffesAdapter.DeleteCommand = new SqlCommand("deleteTariff", sqlConn);
          tariffesAdapter.DeleteCommand.CommandType = CommandType.StoredProcedure;
          p = tariffesAdapter.DeleteCommand.Parameters.Add("@tariffID",
SqlDbType.Int, 0, "tariffID");

          clientsAdapter.DeleteCommand = new SqlCommand("deleteClient", sqlConn);
          clientsAdapter.DeleteCommand.CommandType = CommandType.StoredProcedure;
          p = clientsAdapter.DeleteCommand.Parameters.Add("@clientID", SqlDbType.Int,
0, "clientID");

          paymentsAdapter.DeleteCommand = new SqlCommand("deletePayment", sqlConn);
          paymentsAdapter.DeleteCommand.CommandType = CommandType.StoredProcedure;
          p = paymentsAdapter.DeleteCommand.Parameters.Add("@paymentID",
SqlDbType.Int, 0, "paymentID");

          connectionsAdapter.DeleteCommand = new SqlCommand("deleteConnection",
sqlConn);
          connectionsAdapter.DeleteCommand.CommandType = CommandType.StoredProcedure;
          p = connectionsAdapter.DeleteCommand.Parameters.Add("@connID",
SqlDbType.Int, 0, "connID");

          // UPDATE
          clientsAdapter.UpdateCommand = new SqlCommand("changeTariffForID",
sqlConn);
```

```
          clientsAdapter.UpdateCommand.CommandType = CommandType.StoredProcedure;
          p = clientsAdapter.UpdateCommand.Parameters.Add("@clientID", SqlDbType.Int,
0, "clientID");
          p = clientsAdapter.UpdateCommand.Parameters.Add("@tariffID", SqlDbType.Int,
0, "tariff_id");

          // выводим название сервера и БД
          this.Text = "Подключение активно; ";
          this.Text += sqlConn.DataSource + ": " + sqlConn.Database;
          connectBtn.Enabled = false;
          disconnectBtn.Enabled = true;


          connectionsOfClientBtn.Enabled = true;
          infoOfClientBtn.Enabled = true;
          tablesCombo.Enabled = true;
          fillBtn.Enabled = true;
          updateBtn.Enabled = true;

          // загружаем и отображаем данные из текущей таблицы
          fillBtn_Click(sender, e);
        }
        catch (Exception ex)
        {
          MessageBox.Show(ex.Message);
        }
      }
      else
      {
        this.Text = "Нет подключения; Не выбран источник данных";
        connectBtn.Enabled = true;
        disconnectBtn.Enabled = false;
        connectionsOfClientBtn.Enabled = false;
        infoOfClientBtn.Enabled = false;
        tablesCombo.Enabled = false;
        fillBtn.Enabled = false;
        updateBtn.Enabled = false;
      }
    }

    // кнопка "отключиться от БД"
    private void disconnectBtn_Click(object sender, EventArgs e)
    {
      sqlConn.Close();
      connectBtn.Enabled = true;
      disconnectBtn.Enabled = false;
      this.Text = "Нет подключения";
      if (sqlConn.DataSource == "" || sqlConn.Database == "")
        this.Text += "; Не выбран источник данных";
      else
        this.Text += "; " + sqlConn.DataSource + ": " + sqlConn.Database;
    }

    // ---------------------------------------------------------------------

    private void tablesCombo_Click(object sender, EventArgs e)
    {

    }

    // событие "выбор новой таблицы в выпадающем списке"
    private void tablesCombo_TextChanged(object sender, EventArgs e)
```

```
{
  if (tablesCombo.Text == "") return;
  // автоматически загружаем данные из таблицы
  fillBtn_Click(sender, e);
}

// кнопка "загрузить данные из таблицы"
private void fillBtn_Click(object sender, EventArgs e)
{
  try
  {

    tariffesAdapter.Fill(dataSet.Tables["Tariffes"]);
    clientsAdapter.Fill(dataSet.Tables["Clients"]);
    paymentsAdapter.Fill(dataSet.Tables["Payments"]);
    connectionsAdapter.Fill(dataSet.Tables["Connections"]);
    sqlConn.Close();

    bindingSource.DataSource = dataSet;
    bindingSource.DataMember = tableNames[tablesCombo.SelectedIndex];
    gridView.DataSource = bindingSource;
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}

// кнопка "сохранить изменения в таблице"
private void updateBtn_Click(object sender, EventArgs e)
{
  try
  {
    switch (tablesCombo.SelectedIndex)
    {
      case 0:
        clientsAdapter.Update(dataSet.Tables["Clients"]);
        break;
      case 1:
        tariffesAdapter.Update(dataSet.Tables["Tariffes"]);
        break;
      case 2:
        paymentsAdapter.Update(dataSet.Tables["Payments"]);
        break;
      case 3:
        connectionsAdapter.Update(dataSet.Tables["Connections"]);
        break;
    }
  }
  catch (InvalidOperationException)
  {
    MessageBox.Show("Для данной таблицы не предусмотрена операция Update");
  }
  catch (SqlException ex)
  {
    MessageBox.Show(ex.Message);
    //MessageBox.Show(ex.ToString());
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.Message);
```

```
  }
  finally
  {
    sqlConn.Close();
  }
}


// --------------------- статистика --------------------------------



public void InitValue(Int32 value)
{
  this.value = value;
}

private void DisplayView(String view)
{
  // отображаем форму для ввода параметров запроса
  InputForm inputForm = new InputForm(this);
  inputForm.ShowDialog();

  try
  {
    DataSet ds = new DataSet();
    ds.Tables.Add(new DataTable());
    dataSet.DataSetName = sqlConn.Database;

    // формируем запрос с использованием одного представлений БД
    String query = "SELECT * FROM " + view;
    if (value < 0)
      return;
    if (value > 0)
      query += " WHERE ID = " + value.ToString();

    // извлекаем данные
    SqlDataAdapter ad = new SqlDataAdapter(query, sqlConn);
    ad.Fill(ds.Tables[0]);
    sqlConn.Close();

    // отображаем
    gridView.AutoGenerateColumns = true;
    bindingSource.DataSource = ds.Tables[0];
    gridView.DataSource = bindingSource;
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}

private void connectionsOfClientBtn_Click(object sender, EventArgs e)
{
  DisplayView("ViewOfConnections");
}

private void infoOfClientBtn_Click(object sender, EventArgs e)
{
  DisplayView("ViewOfClients");
}
```

```
    // ------------------------------------------------------------------

  }
}
```

InputForm.cs

```
using System;
using System.Collections.Generic;

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace NetProvider
{
  public partial class InputForm : Form
  {
    private MainForm mainForm;

    private bool accept;
    private Int32 value;

    // ----------------------------------------

    public InputForm(MainForm mainForm)
    {
      InitializeComponent();
      this.mainForm = mainForm;
      accept = false;
    }

    // закрытие формы
    private void InputForm_FormClosed(object sender, FormClosedEventArgs e)
    {
      // было подтверждение ввода данных, передаем параметры как есть
      if (accept)
        mainForm.InitValue(value);
      // подтверждения не было, передаем -1, как признак отмены операции
      else
        mainForm.InitValue(-1);
    }

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

    private void acceptBtn_Click(object sender, EventArgs e)
    {
      try
      {
        value = Convert.ToInt32(textBx1.Text);
      }
      catch (Exception)
      {
```

```
        MessageBox.Show("Неверный параметр");
        return;
      }

      accept = true;
      Close();
    }
  }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace NetProvider
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```