И.В. Скобелев, К.Н. Грибанов, В.В. Фалев

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ НА ПЛАТФОРМЕ 1С: ПРЕДПРИЯТИЕ 8.2

УЧЕБНОЕ ПОСОБИЕ

ISBN 978-5-4217-0236-8

Курганский государственный университет

редакционно-издательский центр

43-38-36

Проект «Инженерные кадры Зауралья»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

И.В. Скобелев, К.Н. Грибанов, В.В. Фалев

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ НА ПЛАТФОРМЕ 1С: ПРЕДПРИЯТИЕ 8.2

Учебное пособие

Курган 2013

УДК 658.012.011.56: 004.42
ББК 65.29
С 44

Рецензенты:
инженер-программист ООО «Сенсор-Сервис», 1С-профессионал **М.А. Жоров**;
канд. экон. наук, доцент Курганской государственной сельскохозяйственной академии им. Т.С. Мальцева **А.Х. Голованова**

Печатается по решению методического совета Курганского государственного университета в рамках проекта «Инженерные кадры Зауралья».

С 44 Скобелев И.В., Грибанов К.Н., Фалев В.В. Объектно-ориентированное программирование автоматизированных систем управления на платформе 1С: Предприятие 8.2: учеб. пособие. - Курган: Изд-во Курганского гос. ун-та, 2013. - 107 с.

В учебном пособии в простой и доступной форме изложены вопросы работы в конфигураторе системы 1С: Предприятие. Рассмотрены механизмы создания базы данных, конфигураций, объектов конфигурации, а также внесения изменений и дополнений в уже существующие решения. Разделы снабжены списком контрольных вопросов и практических заданий.

Работа предназначена для студентов направления подготовки 220700.62 «Автоматизация технологических процессов и производств».

ISBN 978-5-4217-0236-8

СОДЕРЖАНИЕ

Введение

Автоматизация учетных и управленческих процессов на предприятии является одной из ключевых задач, стоящих перед руководством каждого предприятия. Для решения этих трудоемких задач в настоящие время существует несколько программных продуктов. Самым популярным из них является система 1С: Предприятие, имеющая достаточно гибкий пользовательский интерфейс и позволяющая разрабатывать новые решения. Программирование в среде 1С имеет некоторые особенности, связанные с предметной средой. Их изучение позволяет студентам получить навыки работы в области автоматизации производственных и управленческих процессов на предприятии. Платформа 1С Предприятия специально создавалась таким образом, чтобы максимально упростить создание, модификацию и поддержку прикладных решений с точки зрения непосредственно программирования, и позволить разработчику сфокусироваться на наиболее серьезных вопросах автоматизации предметной области.

Прежде всего, следует разделить создание (или существенное развитие) сложных прикладных решений, и небольшие доработки, которые необходимо выполнять в конкретной организации. На практике специалист, не имеющий большого опыта в разработке (продвинутый пользователь, системный администратор) достаточно быстро осваивает основные приемы разработки в 1С: Предприятии и может выполнять доработку прикладного решения, и даже создавать прикладные решения, предназначенные для автоматизации различных участков деятельности предприятия [1].

Вместе с тем разработка серьезных тиражных или заказных прикладных решений, а также существенная доработка бизнес-логики при внедрении типовых решений, безусловно, требует специалиста высокого профессионального уровня. Правильнее использовать не термин программист, а термин разработчик. При этом, в отличие от специалистов, работающих с универсальными средствами программирования, разработчики решений на платформе 1С: Предприятия являются профессионалами прежде всего в сфере автоматизации деятельности предприятия. Само средство разработки изолирует, насколько это возможно, разработчика от технических деталей, оставляя ему решение наиболее важных вопросов построения структур данных, обработки информации, построения интерфейса. Профессиональный рост специалистов направлен в сторону понимания принципов управления предприятиями, создания адекватных экономических моделей, построения сложных структур данных, реализации эффективных алгоритмов бизнес-логики, создания эргономичного интерфейса [1].

Платформа 1С: Предприятия 8.2. написана на MS Visual C++.

При создании платформы использована собственная компонентная архитектура. Сервер 1С: Предприятия создан с использованием технологии COM+ [2].

Учебное пособие предназначено для студентов, обучающихся по направлению 220700.62 «Автоматизация технологических процессов и

производств» и знакомит студентов с отечественной разработкой в области программных средств. Учебное пособие может быть использовано в курсе «Программирование и алгоритмизация» для обучения студентов программированию и конфигурировани в системе 1С: Предприятие 8.2.

Для освоения материала пособия студенты должны проделать подробно рассмотренные в разделах примеры в информационной базе, созданной на платформе 1С: Предприятие, и добиться понимания логики выполнения примеров. Материал построен с последовательным усложнением. и поэтому, переходить к следующему разделу, не усвоив материал предыдущего, не имеет смысла. К каждому разделу приведен список вопросов, направленных на повторение и закрепление основных моментов изложенного материала. Предложены также задания для самостоятельного выполнения по образу и подобию примеров, приведенных в разделе. Как и в любом деле, в программировании, достижение результатов тесно связано с получением навыков владения инструментом, в данном случае 1С: Предприятие 8.2, которые даются многократной тренировкой.

В приложении приведен перечень лабораторных работ и примерные темы курсовых работ, которые и позволят получить необходимый опыт.

1. Основные средства разработки в 1С

Состав средств разработки достаточно разнообразен и позволяет создать как свои объекты в готовой конфигурации, так и полностью свою индивидуальную конфигурацию. Основой программирования в 1С: Предприятии является визуальное программирование на основе уже созданных типовых объектов.

В состав средств разработки входит довольно большое количество конструкторов и редакторов. Конструкторы позволяют автоматизировать и облегчить создание некоторых часто используемых элементов прикладного решения, модификация которых может быть формализована. Как правило, эти элементы прикладного решения могут быть созданы и без использования конструктора, но конструктор позволяет сделать это быстрее и легче.

Например, текст запроса может быть полностью написан самим разработчиком. Для этого разработчик должен хорошо знать синтаксис языка запросов и понимать назначение различных предложений языка запросов. В то же время текст запроса можно создать с помощью конструктора запросов. При этом используется визуальное конструирование запроса, при котором с помощью мыши необходимо выбрать и переместить нужные таблицы, поля, установить связи между ними и т. д. После нажатия кнопки ОК конструктор запроса создаст синтаксически верный текст запроса [1].

Можно перечислить следующие конструкторы, присутствующие в интерфейсе разработки и администрирования:

- конструктор запросов;
- конструктор выходной формы;
- конструктор движений регистров;
- конструктор печати;
- конструктор ввода на основании;
- конструктор форм объектов конфигурации;
- конструктор меню;
- конструктор форматной строки.

Редакторы позволяют создавать и изменять те элементы прикладного решения, для которых модификацию трудно формализовать или формализация не имеет практического смысла.

Например, конструктор форм объектов конфигурации позволяет создать некоторую типовую форму объекта, разместить на ней необходимые поля, назначить источники данных и т. д. Этот процесс легко формализуется и потому может быть выполнен конструктором. Однако дальнейшее редактирование формы, как правило, является процессом творческим и формализации не поддается. Поэтому платформа предоставляет в распоряжение разработчика редактор форм, с помощью которого он придает форме нужный вид, располагает дополнительные элементы управления, настраивает, при необходимости, привязки и т.д. в соответствии с конкретным назначением формы [1].

Можно перечислить следующие редакторы, реализованные в интерфейсе разработки и администрирования:

- редактор форм;
- редактор текстов и модулей;
- редактор табличных документов;
- HTML-редактор;
- редактор интерфейсов;
- редактор картинок.

Все средства разработки доступны в режиме конфигуратора, и позволяют существенно сократить сроки разработки, а также повысить качество получаемых решений. Кроме визуальных средств разработки в руках программиста также остается язык программирования 1С, построенный на основе современных языков программирования и язык запросов, построенный на основе языка SQL-запросов. Операторы языков переведены на русский язык [1].

Вопросы для самоконтроля

1. Что подразумевается под средствами разработки?
2. Для чего нужны средства разработки?
3. Что означает визуальное программирование?
4. В чем разница между конструкторами и редакторами?

2. Конфигуратор - создание базы данных

Разработка прикладных решений и собственно программ осуществляется в режиме конфигурации, в специальной среде разработки. Все средства разработки и ее автоматизации входят в состав технологической платформы. Для начала работы с имеющейся конфигурацией или для создания новой конфигурации надо открыть 1С в режиме конфигуратора [3].

Рассмотрим подробно процесс начала разработки на примере создания своей конфигурации. Начать лучше всего с начала, т.е. создать пустую базу с пустой конфигурацией.

Алгоритм следующий:

1. Создаем пустую папку, называем ее NevB1. В этой папке на диске будем разрабатывать нашу учебную конфигурацию.

2. Запускаем 1С: Предприятие и в окне запуска нажимаем кнопку - «Добавить» (рис.2.1)

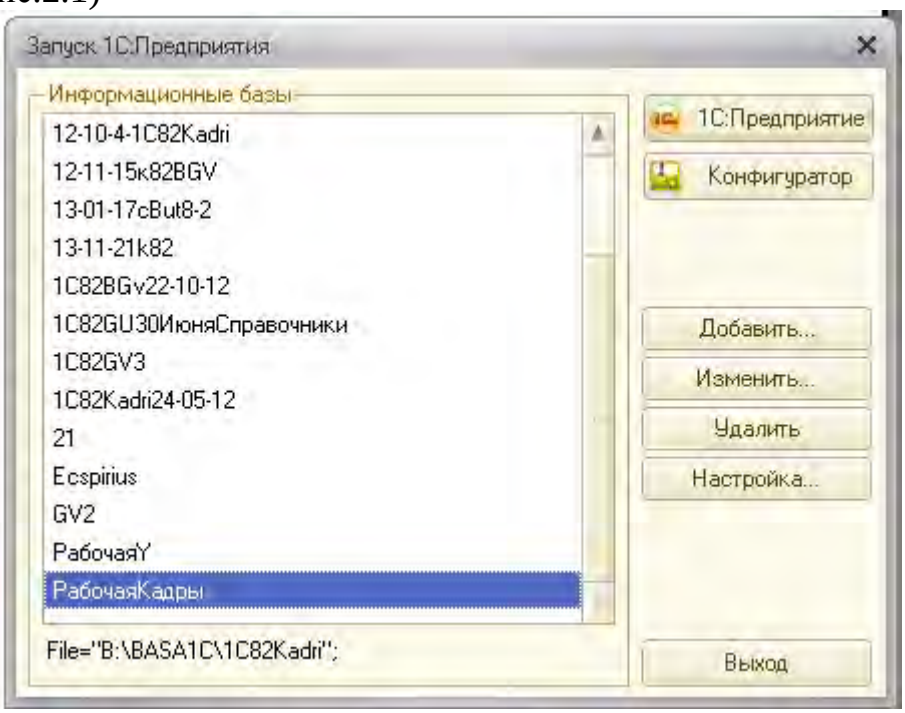


Рисунок 2.1. Окно запуска 1С

Кнопка «Изменить» служит для изменения в базе данных, присутствующей в интерфейсе. Ей изменить можно, в том числе и имя базы в интерфейсе, и «прицепить» интерфейсное имя к другой базе и т.д.

3. Далее выбираем - создание новой информационной базы (рис.2.2). Поскольку в окне запуска еще нет нашей базы, но база физически уже существует на диске, то мы можем дать ей свое интерфейсное имя и таким образом подключить к нашему интерфейсу. База появится в окне запуска 1С: Предприятие.

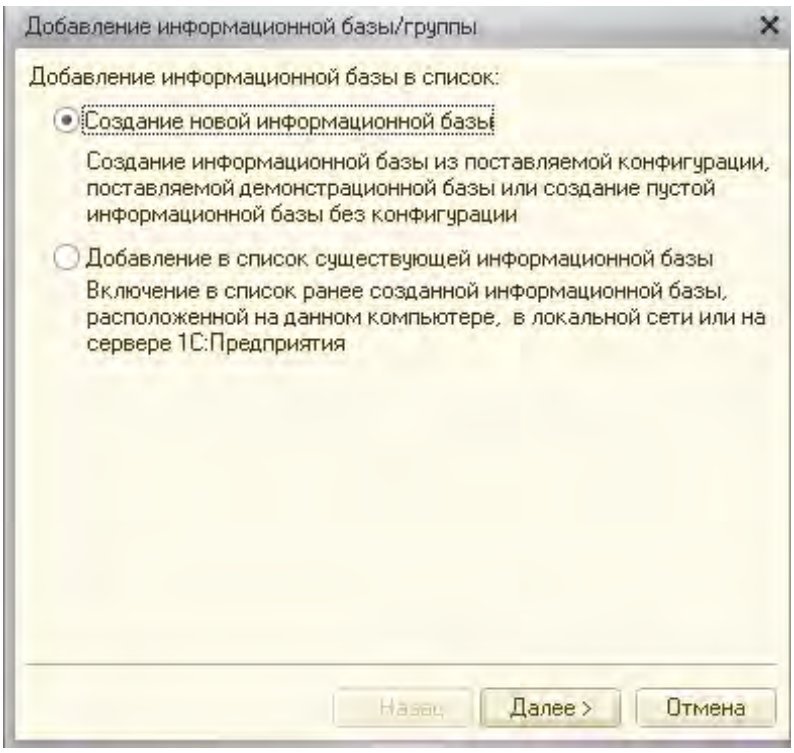


Рисунок 2.2. Окно добавления или подключения информационной базы.

4. Выбираем создание информационной базы без конфигурации для новой разработки (рис.2.3).

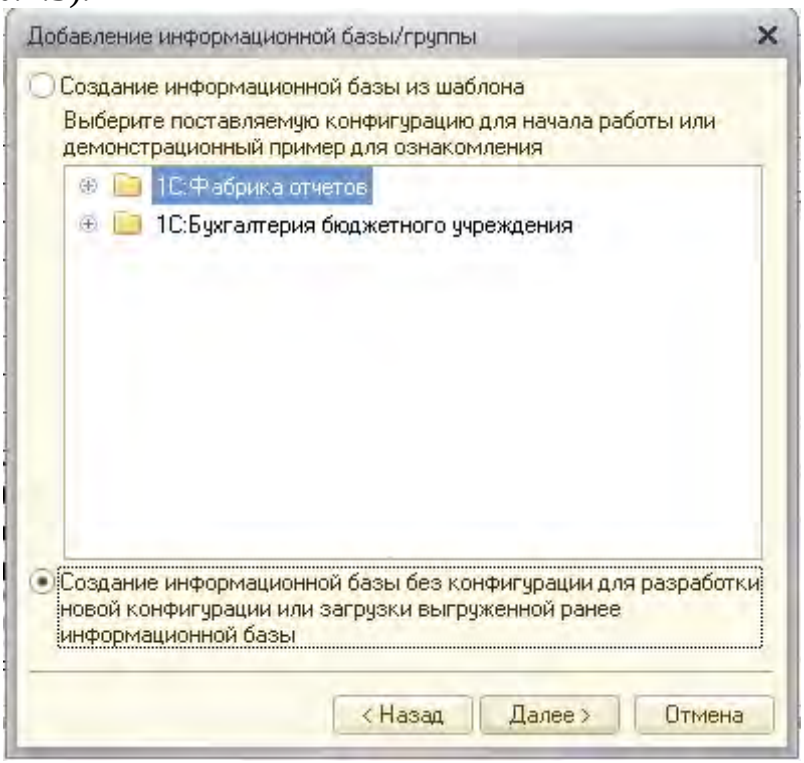


Рисунок 2. 3. Окно выбора типа создаваемой базы.

И нажимаем «Далее», чтобы дать свое интерфейсное имя.

5. Указываем имя такое, какое хотим иметь в интерфейсе. В данном случае: «Наша база» - это имя, которое будет в окне запуска 1С: Предприятие, связано с папкой NevB1 (рис.2.4.).

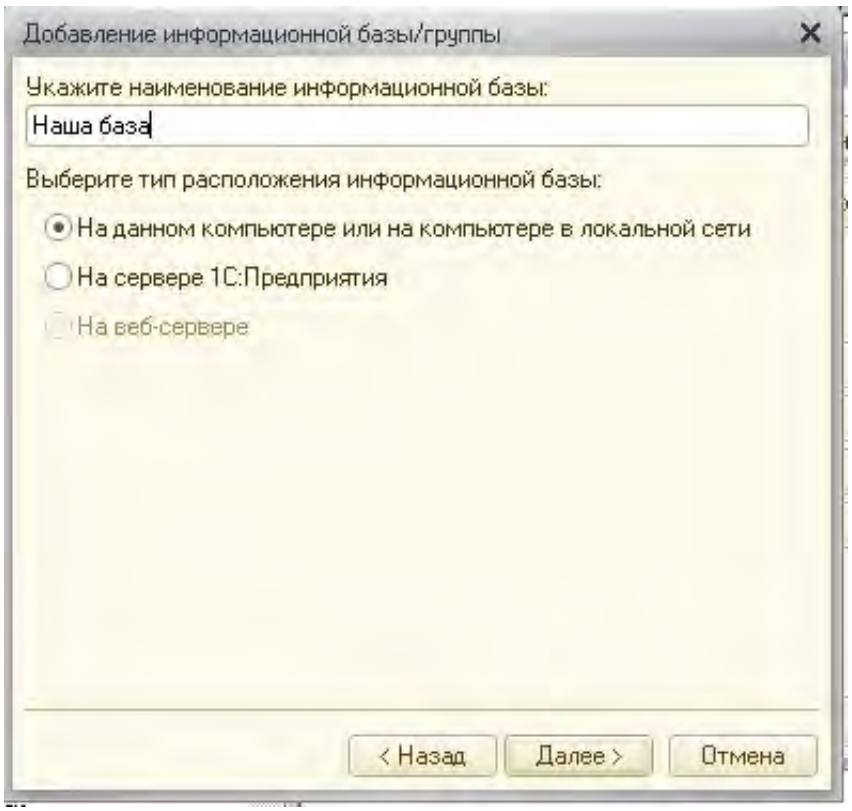


Рисунок 2.4. Присвоение имени базы в интерфейсе запуска 1С.

6. Связываем имя с реально существующей на диске папкой (2.5.).

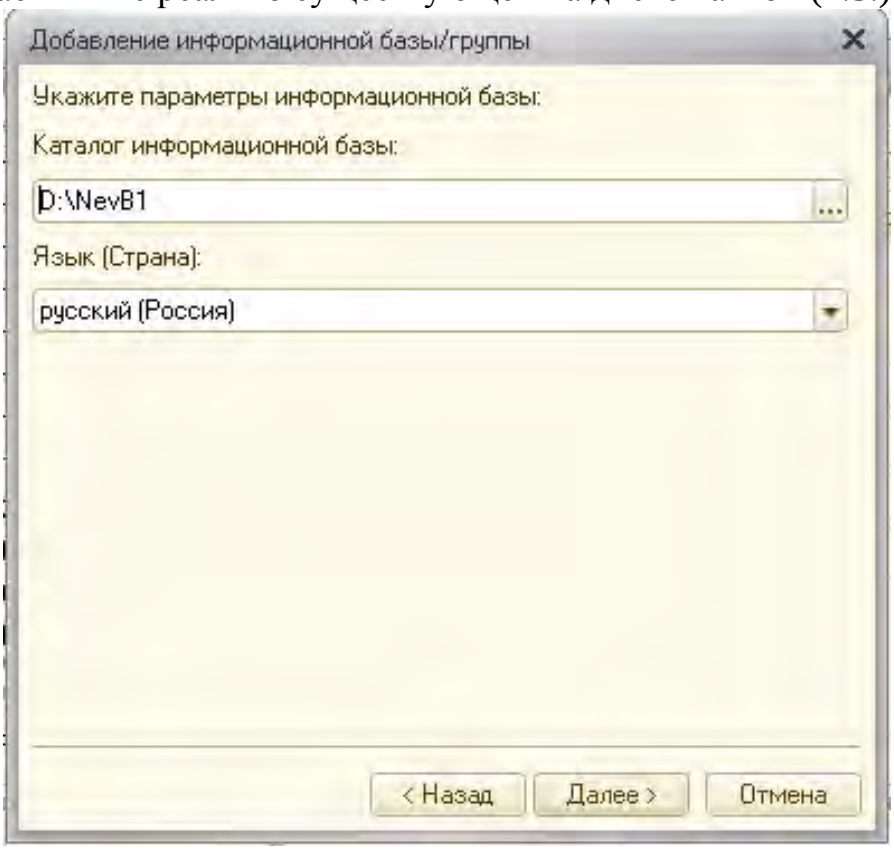


Рисунок 2.5. Установление связи с реально существующей папкой на диске, в которую будет помещена конфигурация.

7. Создание базы завершено (рис.2.6).

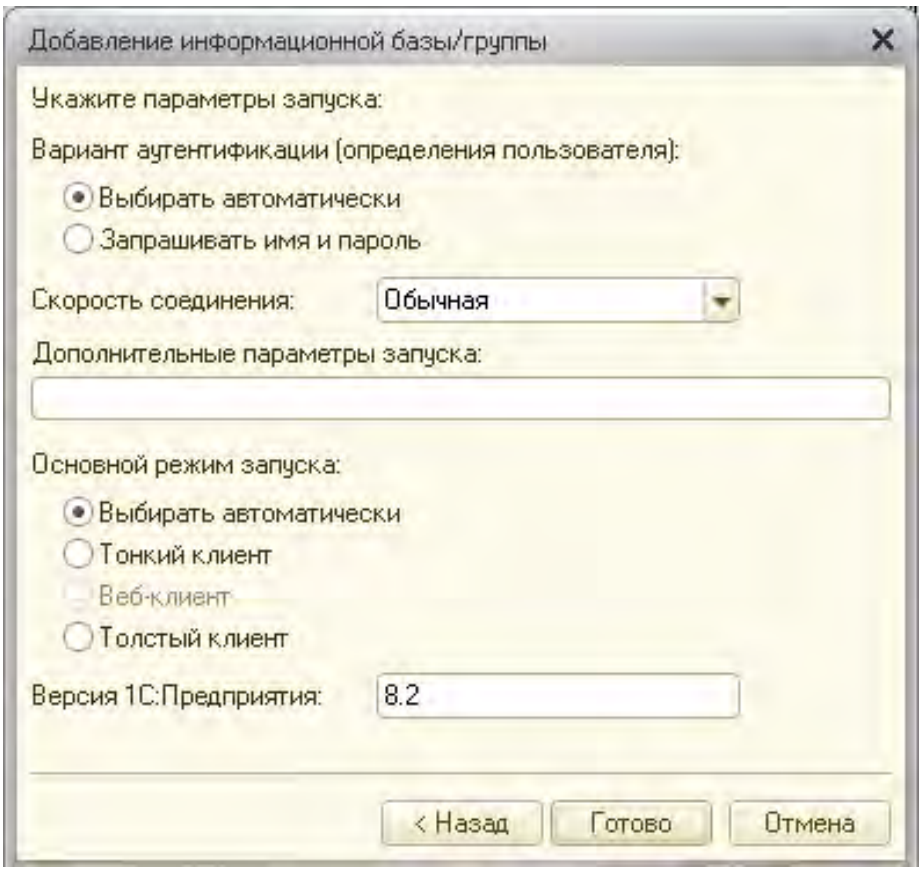


Рисунок 2.6. Последний шаг создания базы.

После завершения создания базы ее имя можно увидеть в интерфейсе (рис.2.7).

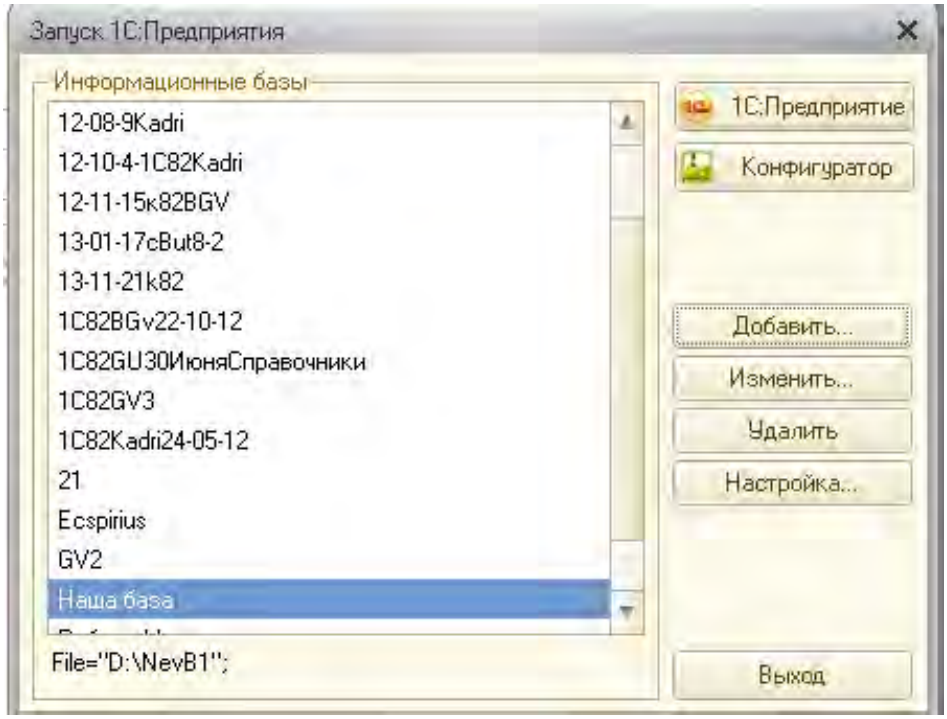


Рисунок 2.7. Результат создания пустой базы.

Теперь посмотрим, что получилось в результате наших усилий. После нажатия кнопки «конфигуратор» открывается окно «Конфигуратора». Это

рабочий инструмент программиста в 1С: Предприятие. Находим в меню пункт «Конфигурация» и подпункт «Открыть конфигурацию» (рис.2.7).

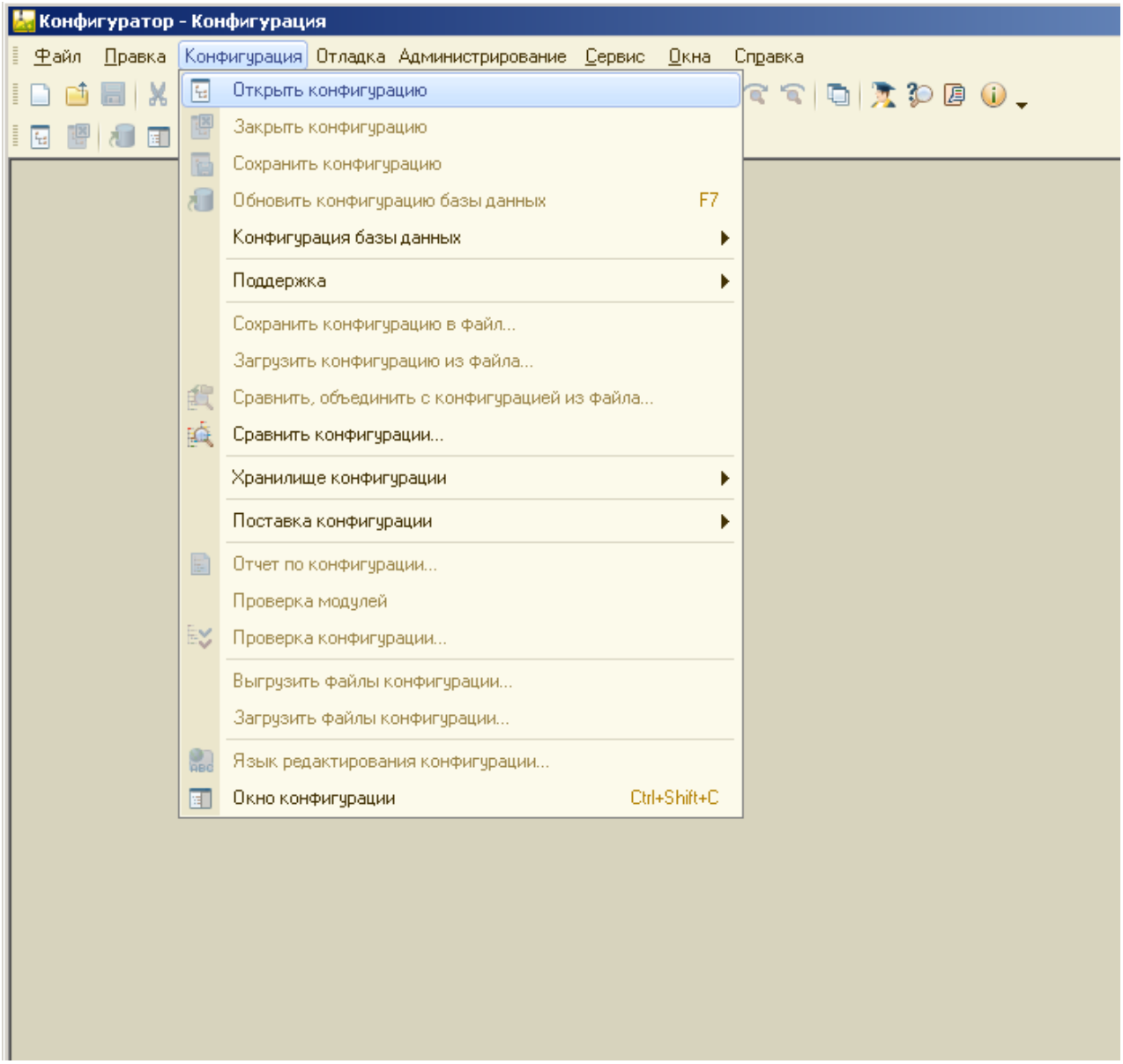


Рисунок 2.8. Открытие конфигурации.

В левой вкладке показаны множество типовых объектов конфигурации (рис.2.9), но пока это только заголовки. Чтобы наполнить их содержанием, надо потрудиться на ниве визуального и даже ручного программирования.

Собственно, создание своих объектов конфигурации на основе имеющихся шаблонов и является объектно-ориентированным программированием в 1С [4].

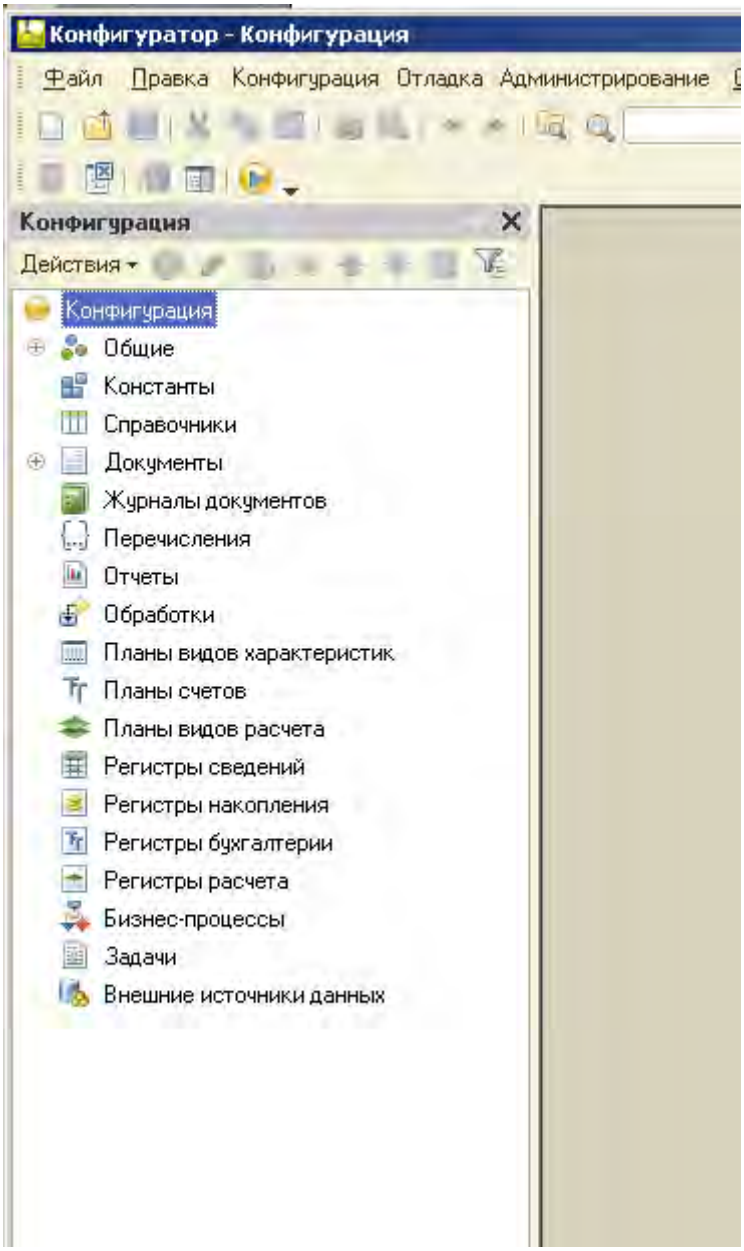


Рисунок 2.9. Объекты конфигурации.

На диске в созданной нами папке мы увидим созданную базу (рис.2.10):

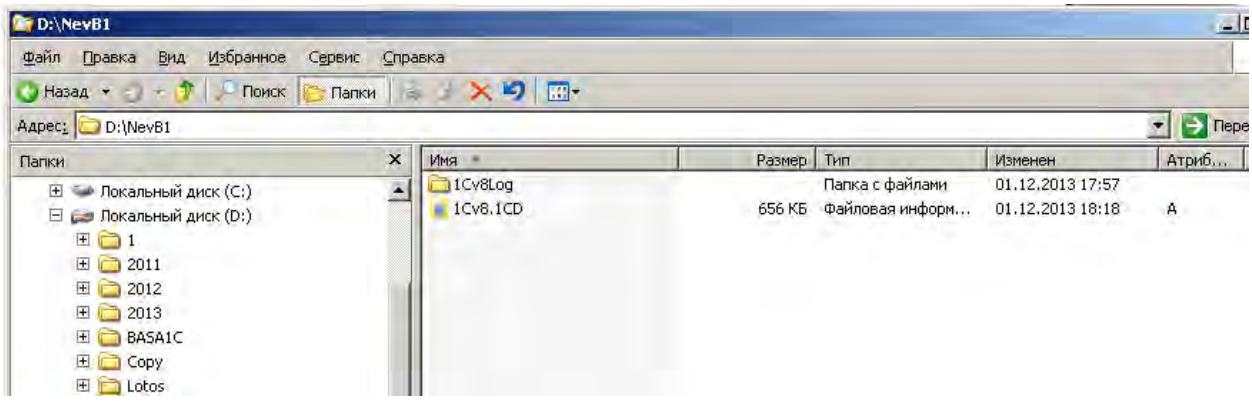


Рисунок 2.10. Содержание папки с нашей базой.

Создание конфигурации начинается с присвоения ей имени. Например, присваиваем имя «ФирмаРОГАиКОПЫТА». Запускаем конфигуратор, щелкаем правой клавишей на конфигурации и выходим на «Свойства» (рис.2.11).

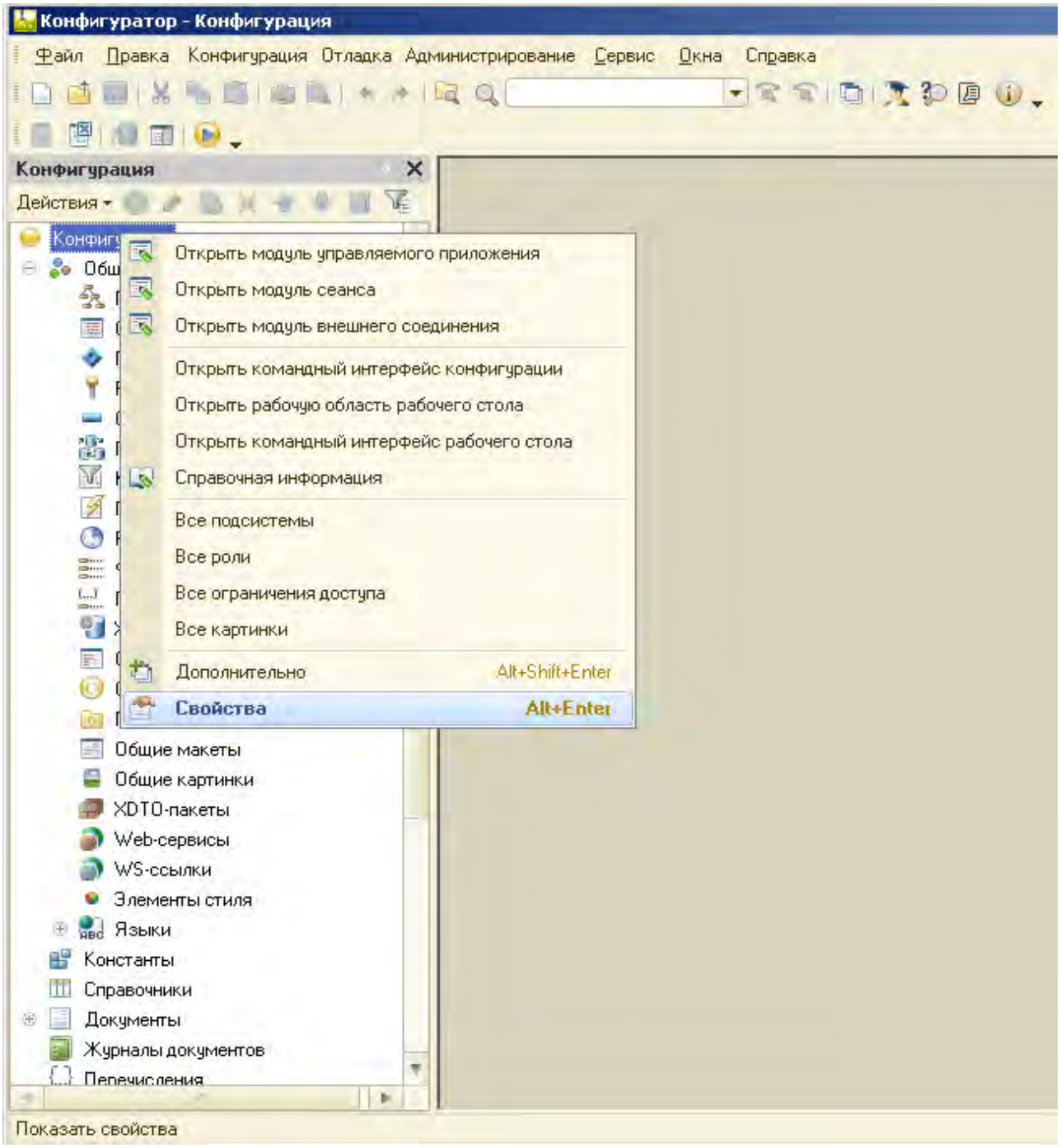


Рисунок 2.11. Доступ к свойствам объекта.

Записываем название: «ФИРМАРОГАиКОПЫТА». Если не записывать синоним, то автоматически повторится название. Здесь же можно указать «основной режим запуска», который определяет вид окна в режиме 1С: Предприятие. «Управляемое приложение» оставляем без изменений (рис.2.12).

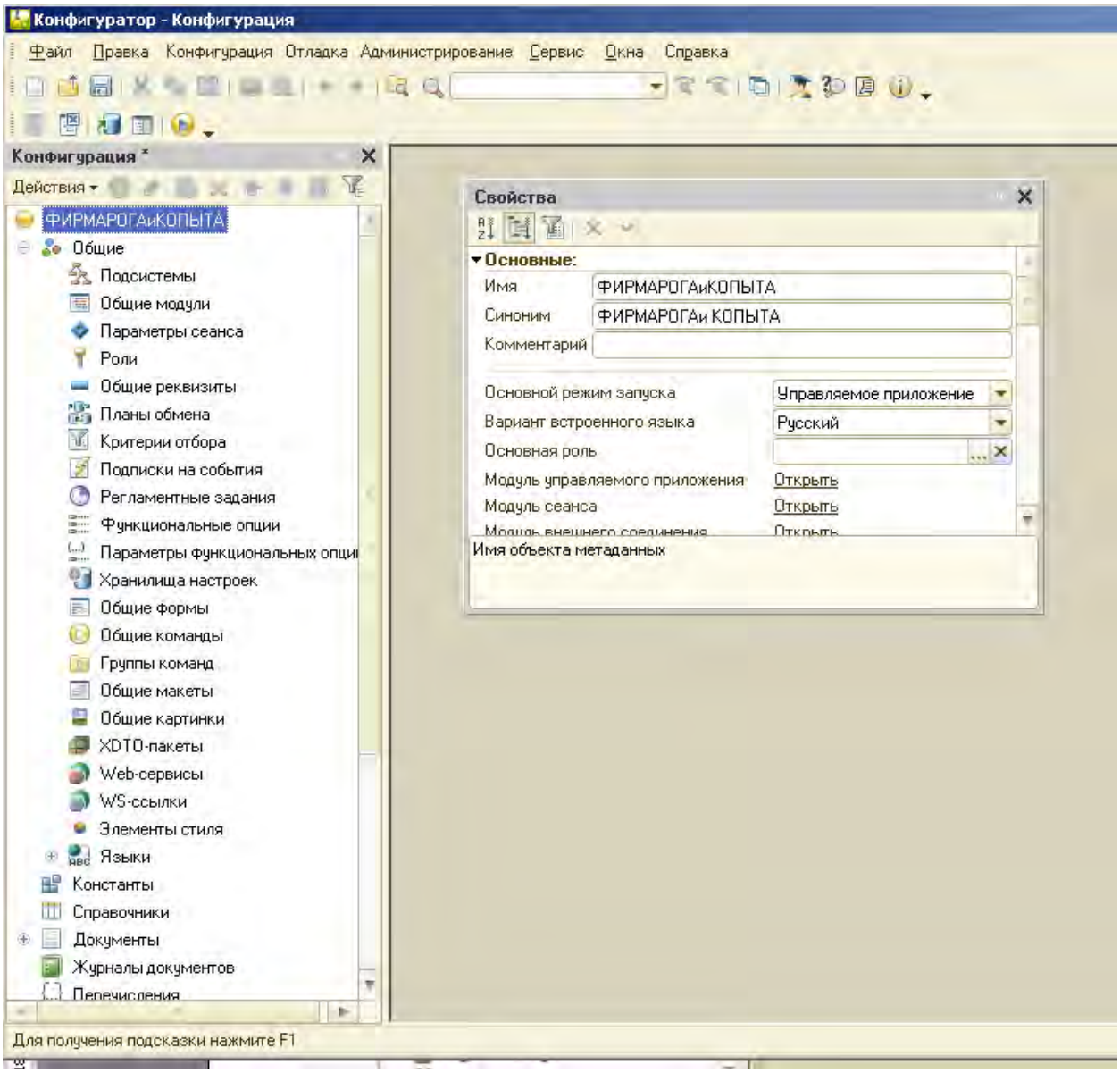


Рисунок 2.12. Задание имени объекта метаданных.

Таким образом, мы создали заготовку для разработки собственной конфигурации. Хотя конфигуратор и предоставляет нам широкие возможности по разработки объектов конфигурации, и во многих пособиях процесс разработки в 1С называют процессом конфигурирования, а не программирования, однако для получения положительных результатов знание основ языка необходимо [4].

Вопросы для самоконтроля

1. Для чего на форме запуска 1С две кнопки «Конфигуратор» и «1С Предприятие»?
2. Как добавить новую конфигурацию в список?
3. Как узнать в какой папке находится та или иная конфигурация из списка?
4. Как открыть конфигурацию для редактирования?
5. Для чего служит режим 1С: Предприятие?

6. Для чего служит режим «Конфигурации»?

7. Как присвоить имя конфигурации?

8. Как соотносятся между собой имя конфигурации, имя в интерфейсе запуска и имя папки на диске в котором находится база данных?

9. Можно ли поменять имя базы в интерфейсе? А имя в конфигурации?

Задание для самостоятельного выполнения

1. Создайте собственную конфигурацию и проделайте весь процесс от начала до конца.

2. Перепишите папку с созданной базой данных на другой диск, компьютер и свяжите с ней интерфейс 1С.

3. Попробуйте изменить «Основной режим запуска», посмотрите результат.

3. Основные понятия языка программирования

Сам по себе язык, на котором пишутся программные модули, мало чем отличается от других современных алгоритмических языков программирования. Базовым отличием можно считать разве что перевод всех терминов и операторов на русский язык, что до некоторой степени облегчает изучение и понимание 1С русскоязычными студентами.

Встроенный язык обладает некоторыми объектно-ориентированными возможностями, однако типизированные данные не могут создаваться средствами самого языка, а задаются с помощью конфигуратора. Переменные могут быть, как описаны явно, так и задаваться в неявном виде.

Программные модули в системе не являются самостоятельными программами в общепринятом смысле этого слова. Поэтому алгоритмы оформляются в виде процедур и функций, которые выполняются в локальном и глобальном контексте [4].

Локальный контекст определяется местом и переменными данного модуля, глобальный - всей конфигурацией.

Пример структуры программного модуля:

```
//****************** ОБЛАСТЬ ОБЪЯВЛЕНИЯ ПЕРЕМЕННЫХ
//это глобальная переменная, доступна из
//других модулей, описатель -Экспорт делает
//их таковыми
Перем КоличествоСотрудников Экспорт;

//это переменная модуля
Перем Фамилия, Имя, Отчество;

//это тоже переменная модуля и к ней можно обращаться
//из любой процедуры и функции нашего модуля
Перем ФИО;



//*** ОБЛАСТЬ ОПИСАНИЯ ПРОЦЕДУР И ФУНКЦИЙ
Процедура Процедура1 (параметр1)
//Итог это локальная переменная (переменная процедуры)

Итог = Фамилия + " " + Имя + " " + Отчество + параметр1;
//Итог это локальная переменная (переменная процедуры), объявлена
//  неявно.

КонецПроцедуры
```

```
Функция Функция1( )

    // операторы функции

    Возврат(Фамилия + " " + Имя);

КонецФункции


//******************** ОСНОВНОЙ ТЕКСТ ПРОГРАММЫ
Фамилия ="Иванов";
Имя = "Иван";
Отчество = "Иванович";
```

Здесь все, что идет после // это комментарии, которые на ход выполнения программы не влияют, но служат пояснением для человека [4].

Если посмотреть на данный пример, то можно заметить, что области действия переменных, аналогичны тому, что мы изучали в VC++. Отличие только в том, что переменные можно не объявлять явно.

В конкретном программном модуле любая из областей может отсутствовать.

Область объявления переменных размещается от начала текста модуля до первого оператора Процедура или оператора Функция или любого исполняемого оператора. В этом разделе могут находиться только операторы объявления переменных «Перем» [5].

Область описания процедур и функций размещается от первого оператора Процедура или оператора Функция до любого исполняемого оператора вне тела описания процедур или функций.

Область основного текста программы размещается от первого исполняемого оператора вне тела процедур или функций до конца модуля. В этом разделе могут находиться только исполняемые операторы. Область основного текста программы исполняется в момент инициализации модуля. Обычно в разделе основной программы имеет смысл размещать операторы инициализации переменных какими-либо конкретными значениями, которые необходимо присвоить до первого вызова процедур или функций модуля.

Программные модули располагаются в тех местах конфигурации, которые могут требовать описания специфических алгоритмов функционирования. Эти алгоритмы следует оформлять в виде процедур или функций, которые будут вызываться самой системой в заранее предусмотренных ситуациях (например, при открытии формы справочника, при нажатии кнопки в диалоговом окне, при изменении объекта и т.д.).

Каждый отдельный программный модуль воспринимается системой как единое целое, поэтому все процедуры и функции программного модуля выполняются в едином контексте.

Контекст выполнения модулей делится на клиентский и серверный. Кроме того, некоторые программные модули могут быть скомпилированы как на стороне клиента, так и на стороне сервера. Модуль приложения (управляемого или обычного) может содержать все 3 области, выполняется на стороне клиента, располагается в корневом разделе конфигурации.

В модуле приложения описываются процедуры (обработчики) событий, которые инициализируются при старте и окончании работы системы. Например, при начале работы приложения можно обновить какие-либо данные конфигурации, а при завершении работы - поинтересоваться, стоит ли вообще выходить из программы. Кроме того, в данном модуле перехватываются события от внешнего оборудования, например, торгового или фискального. Стоит отметить, что модуль приложения выполняется только в случае интерактивного запуска приложения, то есть когда запускается окно программы. Этого не происходит, если приложение запускается в режиме com-соединения [4].

В платформе 1С: Предприятие 8 существует два различных модуля приложения. Это модуль Обычного приложения и модуль Управляемого приложения. Они срабатывают при запуске различных клиентов. Так, модуль Управляемого приложения срабатывает при запуске веб-клиента, тонкого клиента и толстого клиента в режиме управляемого приложения. А модуль Обычного приложения срабатывает при запуске толстого клиента в режиме Обычного приложения. Настройка режима запуска приложения задается в свойстве конфигурации «Основной режим запуска», как было показано в разделе 2.

В модуле приложения могут располагаться все 3 раздела - объявления переменных, описания процедур и функций, а также основной текст программы. Модуль приложения компилируется на стороне клиента, что сильно ограничивает нас в использовании многих типов данных. Расширить контекст модуля приложения можно за счет методов общих модулей, для которых установлено свойство «Вызов сервера». Все переменные и методы программного модуля приложения, помеченные как экспортные, будут доступны в любом модуле конфигурации, работающем на стороне клиента. Однако, как бы ни было это заманчиво, не следует размещать здесь большое количество процедур и функций. Чем больше в данном модуле находится кода, тем длительнее время компиляции, а, следовательно, и время запуска приложения [4].

В 1С: Предприятие различают следующие типы модулей:

1. Общий модуль - в конфигурации может быть определено произвольное количество общих модулей, в том числе и ни одного. Контекст общего модуля образуется глобальным контекстом и локальным контекстом самого общего модуля, то есть процедурами и функциями, определенными внутри общего модуля.

2. Модуль приложения - выполняется при запуске системы в режиме 1С: Предприятие или при обращении к приложению как к Automation-серверу. Этот модуль предназначен для отработки действий, связанных с сеансом работы

конечного пользователя. Помимо описания переменных и основной программы, модуль приложения может содержать описание процедур-обработчиков событий, связанных с сеансом пользователя и прикладным решением в целом.

3. Модуль внешнего соединения выполняется при обращении к приложению как к COM-серверу (в режиме внешнего соединения). В режиме внешнего соединения запускается не полноценное приложение 1С: Предприятия, а «облегченный» вариант приложения, в котором недоступны все функции, так или иначе связанные с организацией пользовательского интерфейса. Поэтому в режиме внешнего соединения вместо модуля приложения исполняется модуль внешнего соединения. Этот модуль предназначен для отработки действий, связанных с сеансом работы с приложением 1С: Предприятия.

4. Модуль объекта имеется в каждом прикладном объекте конфигурации, данные которого могут быть модифицированы в режиме 1С: Предприятие. Этот модуль исполняется при создании объекта встроенного языка, который позволяет модифицировать данные объекта конфигурации. Соответствующий объект встроенного языка создается, например, при вводе нового объекта, при копировании, при получении данных существующего объекта и т. д. Для различных объектов конфигурации этот модуль имеет разное название.

5. Модуль формы имеет каждая форма, определенная в конфигурации. Этот модуль исполняется при создании объекта Форма встроенного языка. Этот объект может создаваться при открытии формы прикладного объекта.

Если переменные, процедуры или функции модуля формы определены как экспортируемые, то они будут доступны в качестве свойств и методов соответствующих объектов.

Язык программирования 1С: Предприятия оперирует различными величинами: числами, символами, объектами. Каждая величина имеет тип, который определяет возможные значения и набор доступных для них операций.

Существуют типы, определенные на уровне системы, и типы, создаваемые в конкретном прикладном решении. Например, на уровне системы определены примитивные типы, такие как Строка, Число, Булев и т. д. Также на уровне системы определены и другие типы, которые могут быть использованы в прикладном решении, например универсальные коллекции значений (Массив, Структура, СписокЗначений), общие типы (ТекстовыйДокумент, ТабличныйДокумент, ПостроительОтчета, АнализДанных) и др. Полный перечень типов значений, которые может использовать система 1С: Предприятие, приведен в описании встроенного языка и в синтакс-помошнике.

Переменные встроенного языка 1С: Предприятия динамически типизированы. Это значит, что тип переменной определяется типом того значения, которое хранится в переменной в данный момент [5].

В то же время данные 1С: Предприятия существуют не только в оперативной памяти компьютера, где они содержатся в объектах встроенного языка, но и в базе данных, где осуществляется долговременное хранение этих данных. База данных представляет собой совокупность некоторого количества таблиц, создаваемых в соответствии со структурой метаданных прикладного

решения. Таблицы базы данных состоят из полей, и для каждого поля обязательно должен быть указан тип значений, которые могут храниться в этом поле. По этой причине все объекты метаданных, которые «отвечают» за создание тех или иных полей в базе данных, должны иметь совершенно определенный тип. Такими объектами являются, например, реквизиты, измерения, ресурсы и т. д.

Так же как и в других языках программирования есть некоторое ограниченное количество примитивных типов данных:

1. Число - для операций над числами здесь нет множества различных типов числовых данных как в C++, все гораздо проще. Операции над числами стандартные, как и для большинства языков программирования.

2. Строка - для операций над текстовыми строками. Имеется набор стандартных функций для операций над строковыми данными, аналогичный стандартным функциям работы со строковыми данными языка C++.

3. Дата - для операций с датой и временем. Для операций с ними имеется значительный аппарат стандартных функций, их использование и назначение можно посмотреть в технической документации.

4. Булев тип - для логических операций.

Операторы языка программирования, как и в большинстве языков, отделяются друг от друга символом - «;».

Как и в большинстве языков программирования эти простейшие типы могут объединятся в составные типы: массивы, структуры и т.д. [4].

Примеры использования простых типов переменных:

A = 3; - При первом присвоении значения система создает данную переменную. Данная переменная имеет числовое значение.

Сумма = 23.5+12*2; - Переменной Сумма присваивается числовое значение. С данными числового типа можно выполнять арифметические операции: сложение, вычитание, умножение и деление. В качестве разделителя целой и дробной части используется точка!

A = -0.123; - Числовые значения могут быть отрицательными.

Предупреждение = "Внимание! - это предупреждение"; - Переменной Предупреждение присваиваем строковое значение. Значение строкового типа пишется в кавычках.

ФИО = "Иванов" + " " + "Кирилл" + " " + "Витальевич"; - Строки могут формироваться путем сложения.

Результат сложения строк: ФИО = "Иванов Кирилл Витальевич"

Символ " " мы прибавляем, чтобы между фамилией, именем и отчеством были пробелы.

ДатаНГ13 = '2013.01.01'; Переменная, которая хранит дату. Значение типа Дата записывается в одинарных кавычках.

КоличествоСекунд = '2013.01.02' - '2013.01.01';
КоличествоСекунд = 86400 Даты можно вычитать одну из другой. В результате получим разницу между датами, измеренную в секундах. В сутках 86 400 секунд (60 сек * 60 мин * 24 ч).
НДата = '2013.01.01' + 172800;
Результат: НДата = '2013.01.03'

Конструкции языка программирования:
Начало процедуры обозначается словом ПРОЦЕДУРА <имя процедуры>(<Список параметров>), конец словом КОНЕЦПРОЦЕДУРЫ. Процедура может получать или возвращать параметры, т.е. переменные перечисленные в скобках после имени процедуры
Например:

```
Процедура Вычислить (Цена, количество, НДС, стоимость)

    Стоимость = (Цена + цена * НДС / 100) * количество;

КонецПроцедуры
```

Вызов процедуры:

```
…..
…..
Цена = 450;
Количество = 10;
НДС = 10;
Вычислить (Цена, количество, НДС, стоимость)
Результат = Стоимость;
…..
```

В результате получим: стоимость = 4950.
Различают процедуры и функции, последние начинаются со слова ФУНКЦИЯ
Тот же пример с функцией:

```
Функция Вычислить (Цена, количество, НДС)

    Стоимость = (Цена + цена*НДС/100)* количество;
    Возврат Стоимость;
КонецФункции
```

Вызов процедуры:

```
…..
…..
Цена = 450;
Количество = 10;
НДС = 10;
Результат = Вычислить (Цена, количество, НДС);
…..
```

Всем знакомые конструкции разветвляющихся алгоритмов - оператор
ЕСЛИ <условие > Тогда
<Операторы, выполняющиеся, если условие истинно >
ИНАЧЕ
<Операторы, выполняющиеся, если условие ложно >
КОНЕЦЕСЛИ;
Например:

```
Если Доход > 20000 Тогда
  Результат = "Жить можно";
Иначе
  Результат = "Так жить нельзя!";
КонецЕсли;
```

Возможна сокращенная форма оператора, аналогичная тернарной операции в C++:

```
Результат = ?(Доход > 20000, "Жить можно ", "Так жить нельзя!");
```

.

Существует оператор с множественным условием:

```
ЕСЛИ <условие1 > Тогда
        <Операторы выполняющиеся если условие1 истина >
ИНАЧЕ ЕСЛИ <условие2 > Тогда
        <Операторы выполняющиеся если условие2 истина >
ИНАЧЕ ЕСЛИ <условие3 > Тогда
        <Операторы выполняющиеся если условие3 истина >
…….
…….
ИНАЧЕ
        <Операторы, выполняющиеся если условие ложно >
КОНЕЦЕСЛИ;
```

Например:

```
ЕСЛИ доход >100 000 Тогда
        Результат = "Жизнь хороша и жить хорошо ";
ИНАЧЕ ЕСЛИ доход > 50 000 Тогда
        Результат = "Хорошо живем";
ИНАЧЕ ЕСЛИ доход > 20000 Тогда
        Результат = "Жить можно";
ИНАЧЕ
        Результат = "Так жить нельзя!";
КОНЕЦЕСЛИ;
```

Циклы - какой же язык программирования без циклов?

```
ПОКА <условие > Цикл
        <Операторы тела цикла >
```

КонецЦикла;
Цикл выполняется, пока условие истинно.

Например:
Н=1
Пока Н < 10 Цикл
    ……
    Н = Н + 1;
КонецЦикла;
Либо другой вариант:
ДЛЯ<Начальное значение счетчика> ПО <Конечное значение счетчика > Цикл
    <Операторы тела цикла >

КонецЦикла;

Пример:

ДЛЯ Н = 1 По 10 Цикл
    ……
КонецЦикла;
Результат один и тот же, но операторов меньше [6].

Вопросы для самоконтроля

1. Для чего нужны комментарии?
2. Какова структура программного модуля?
3. Какие разновидности программных модулей существуют в 1С?
4. Какие типы переменных могут быть в 1С?
5. Приведите примеры операций с числовыми данными.
6. Приведите примеры операций со строковыми данными.
7. Приведите примеры операций с датой и временем.
8. Какие операторы языка 1С Вы знаете?
9. Как отделяется один оператор от другого.
10. Объясните разницу между процедурой и функцией.
11. Приведите пример краткой формы оператора ЕСЛИ.
12. Какие операторы существуют для организации циклов?
13. Приведите пример оператора ЕСЛИ с множественным условием.
14. Можно ли организовать цикл, не используя операторы цикла?

4. Создание типовых объектов в конфигурации

Этот раздел посвящен созданию новых объектов конфигурации. Начнем с подсистем, которые служат для того, чтобы было удобнее пользоваться объектами конфигурации. Проектирование состава подсистем важная и ответственная задача, т.к. подсистема позволяет выделить в конфигурации функциональные части. Каждый объект конфигурации может быть включен в одну или несколько подсистем.

Любое предприятие как-то контактирует с внешним миром, следовательно, у него есть контакты и оно, конечно же, получает какие-то финансовые средства.

Создадим сначала две подсистемы: Контакты и Финансы. Если потребуется, то в конфигурацию можно добавить еще подсистемы или построить иерархическую структуру, состоящую из подсистем и их подсистем. Чтобы создать любой объект конфигурации, надо на вкладке выбрать этот объект и щелкнуть правой клавишей мыши - появится окно с пунктом меню «Добавить» (рис.4.1).

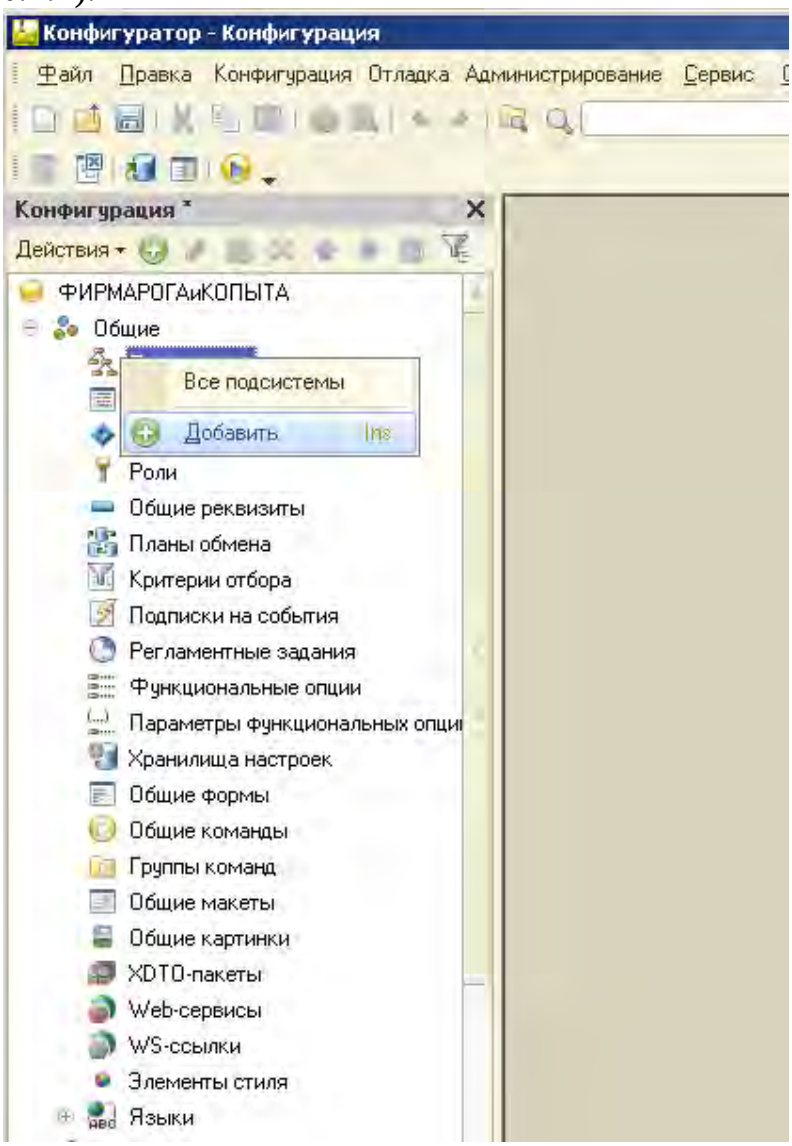


Рисунок 4.1 Добавление объектов типа подсистемы.

Таким образом, добавляем новую подсистему и на основной вкладке задаем ее имя (рис.4.2) [3].

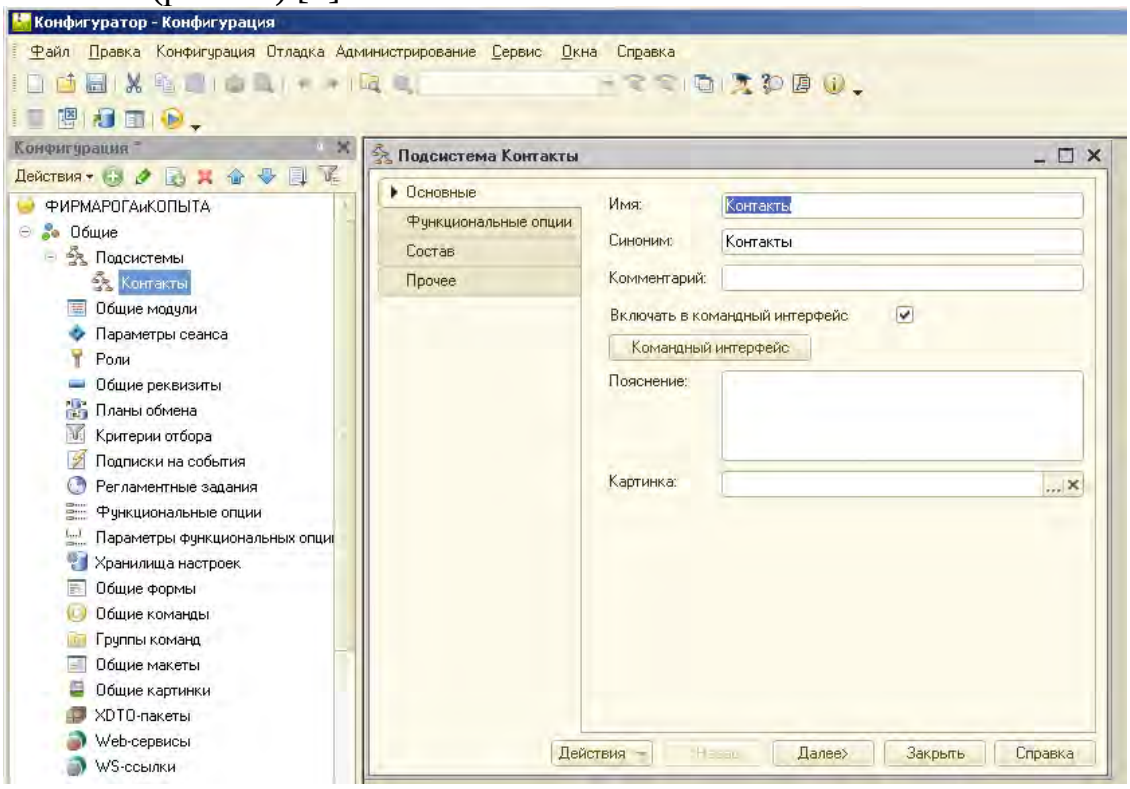


Рисунок 4.2. Задание имени подсистемы.

Наполнение подсистемы начинаем со справочников, как наиболее постоянных элементов. Справочник - это список данных, в конфигураторе мы описываем свойства и структуру этих списков. Так как предприятие покупает сырье у поставщиков и реализует товары покупателям, то нам необходим справочник «Контрагенты» (рис.4.3).

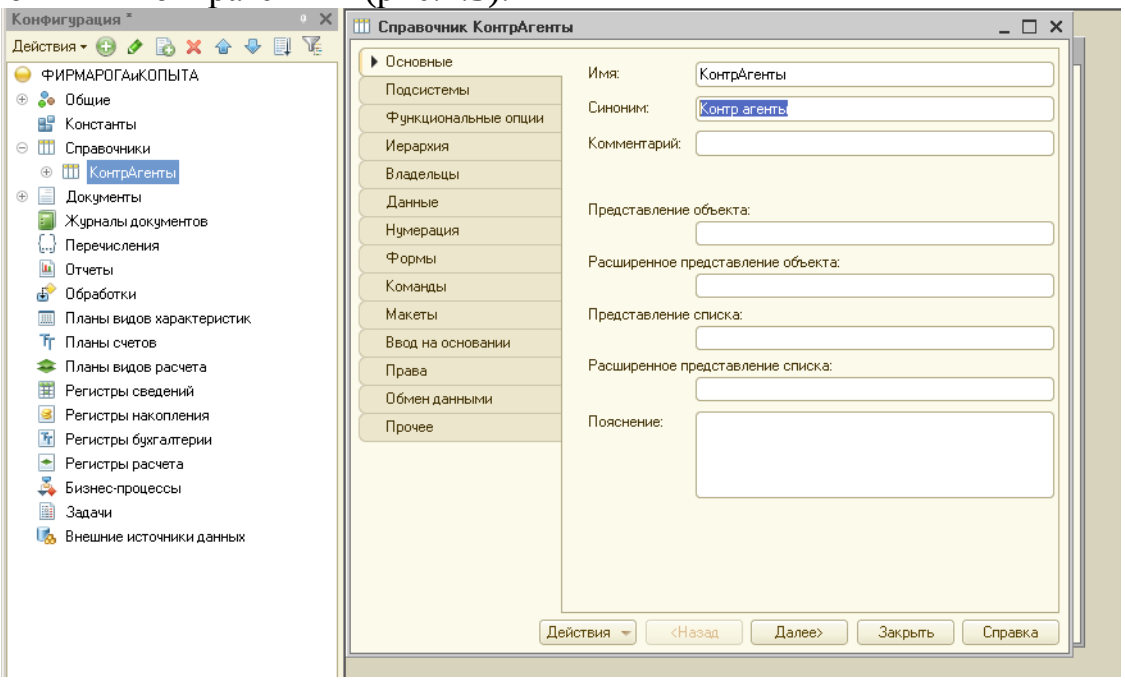


Рисунок 4.3. Создание справочника: «Контрагенты».

На второй вкладке отмечаем, что справочник будет принадлежать подсистеме контакты.

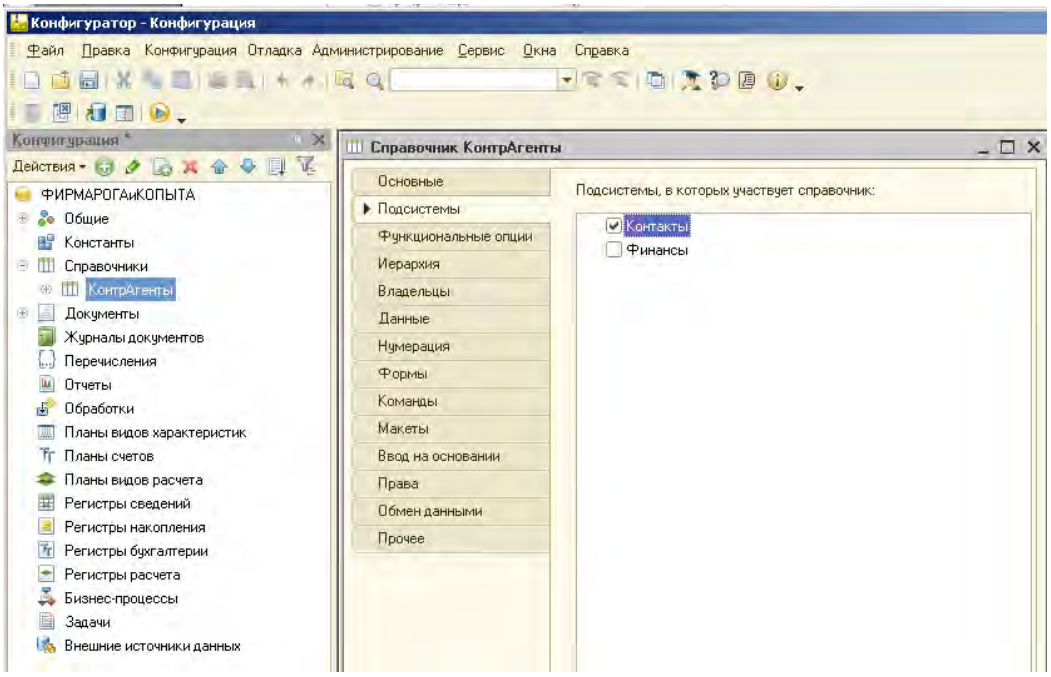


Рисунок 4.4. Связывание справочника с подсистемой.

После открытия состава подсистемы «Контакты» на вкладке Состав видим, что справочник «Контрагенты» действительно входит в подсистему (рис.4.5). Рассмотрим результат с точки зрения пользователя. Для этого запустим 1С: Предприятие в пользовательском режиме, для чего нажмем соответствующую кнопку «Начать отладку» или F5. Далее необходимо согласиться с произведенными изменениями в конфигурации базы данных. Процесс отображен на рисунках 4.6.- 4.8.

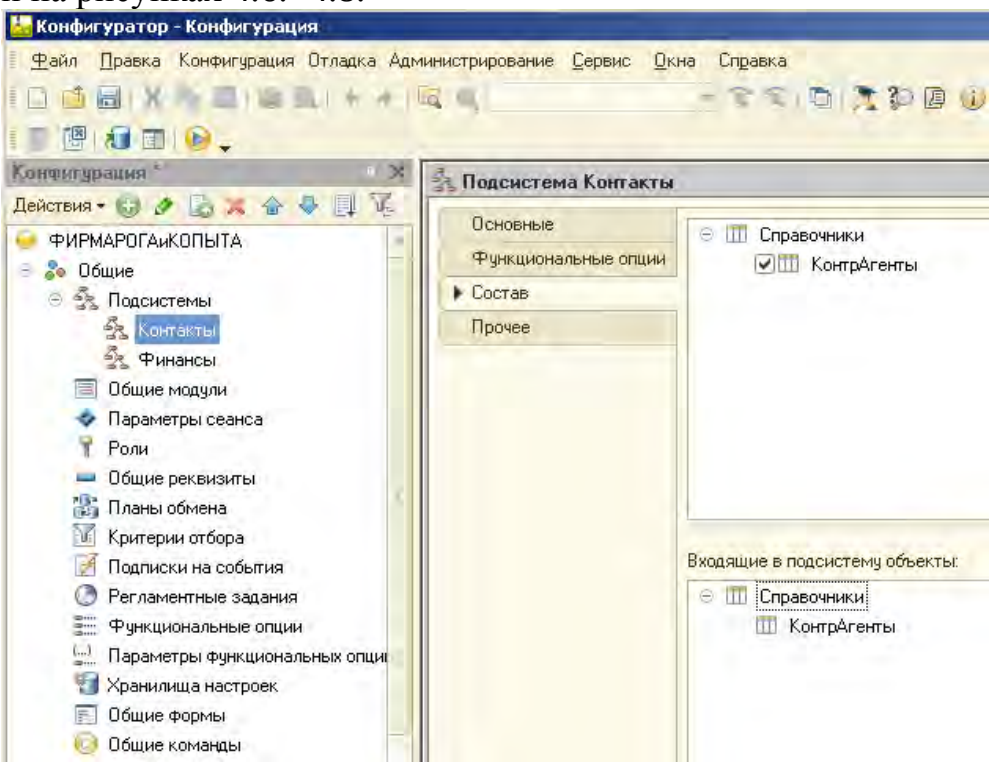


Рисунок 4.5. Состав подсистемы.

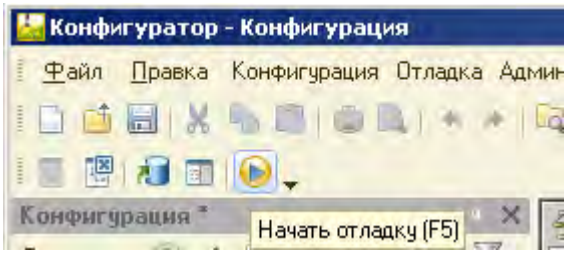


Рисунок 4.6. Запуск отладки.

Соглашаемся с необходимостью изменений:

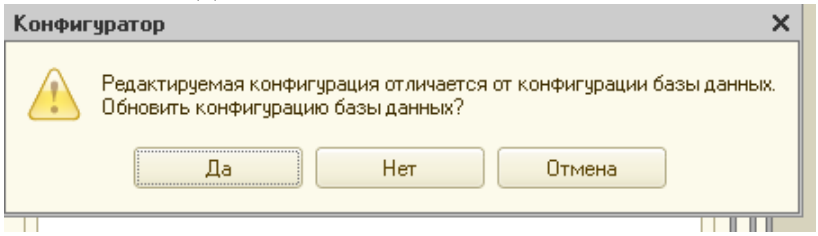


Рисунок 4.7. Предупреждение об изменении конфигурации.

И с тем, что мы новый справочник ввели в конфигурацию:

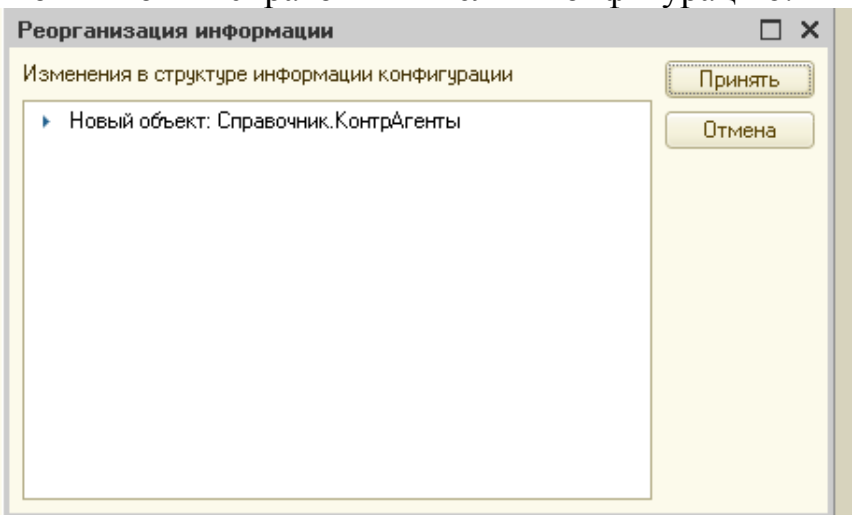


Рисунок 4.8. Сообщения о внесенных изменениях.

Получилось примерно так (рис.4.9):

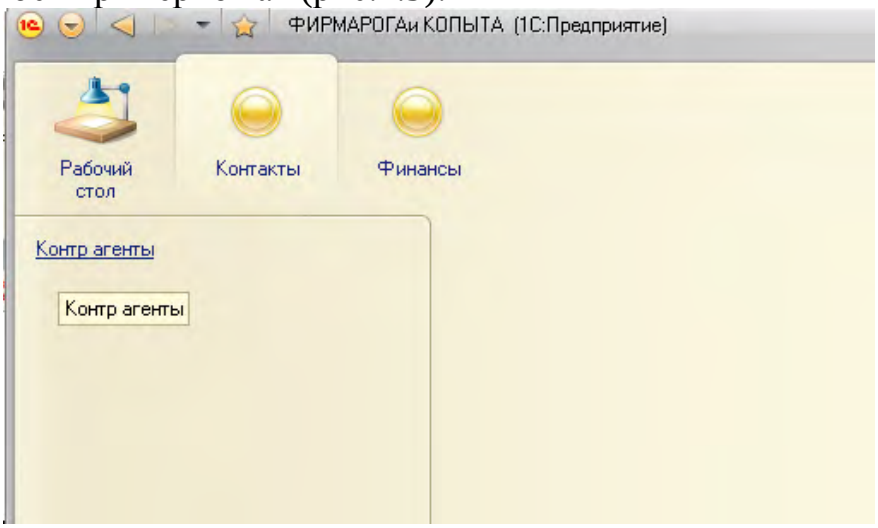


Рисунок 4.9. Окно пользовательского интерфейса.

После открытия приложения видим, что на панели разделов появились созданные нами подсистемы, а на панели навигации - пустой справочник «Контрагенты» с двумя обязательными реквизитами: «наименование» и «код» (рис.4.10).

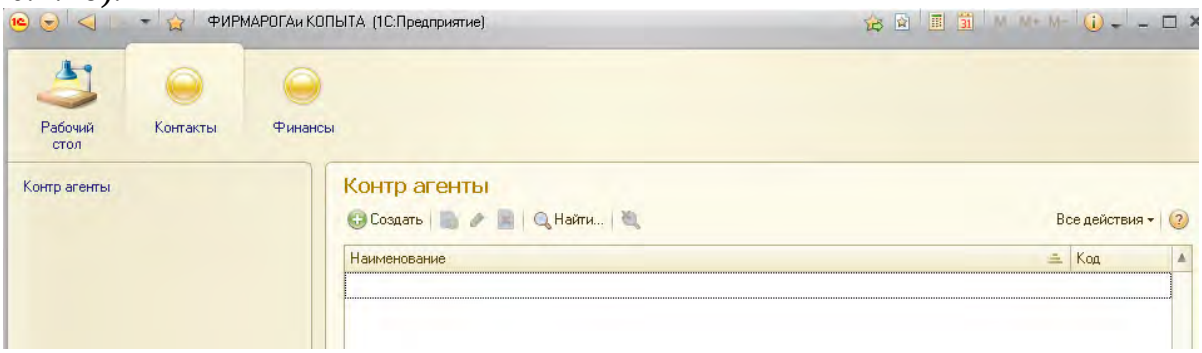


Рисунок 4.10. Отображение справочника «Контрагенты».

Созданный справочник является заготовкой для дальнейшего совершенствования конфигурации. Рассмотрим реальный справочник «Контрагенты» в типовой конфигурации (рис.4.11):

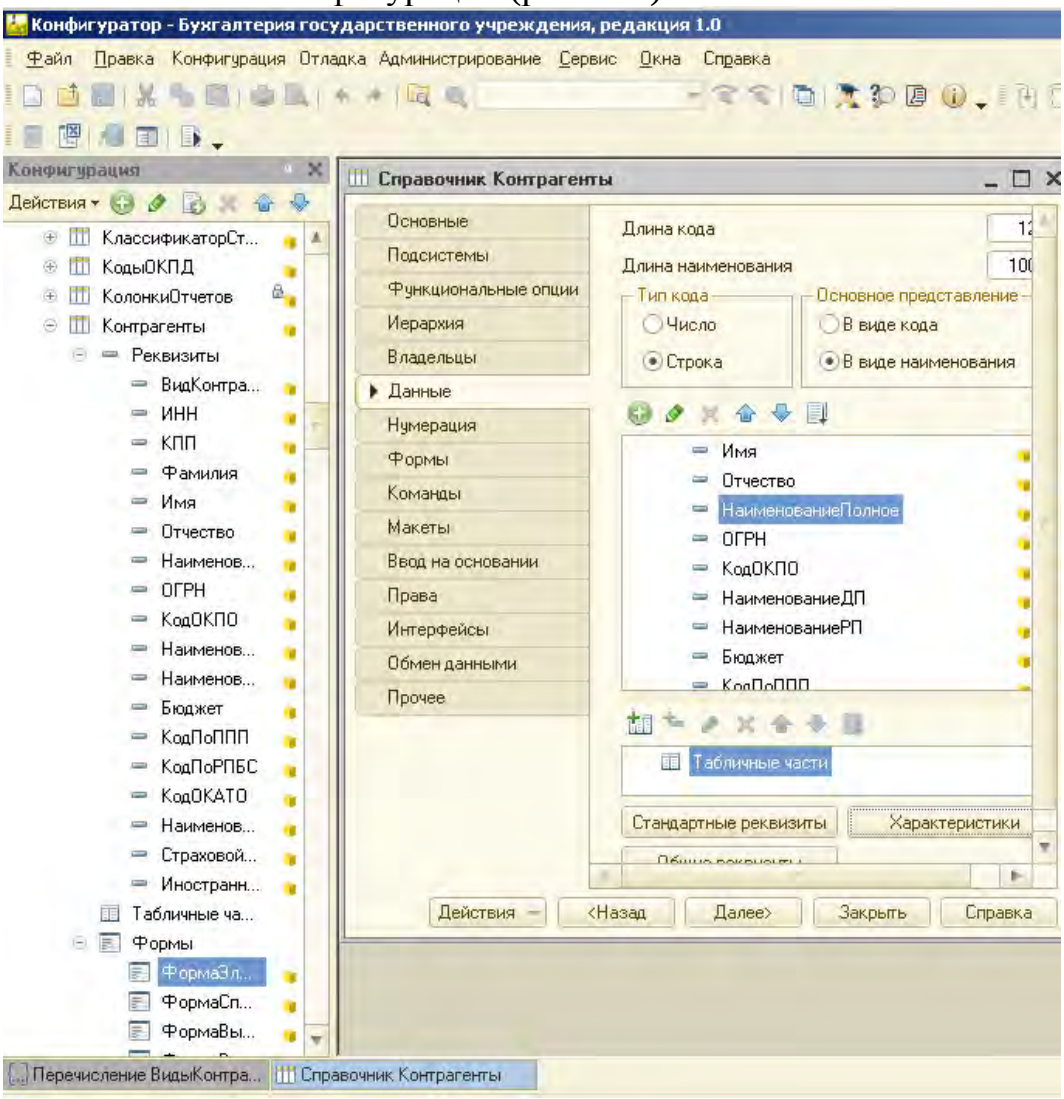


Рисунок 4.11. Справочник «Контрагенты» в типовой конфигурации.

В справочнике видны реквизиты и разнообразные формы (рис.4.12).

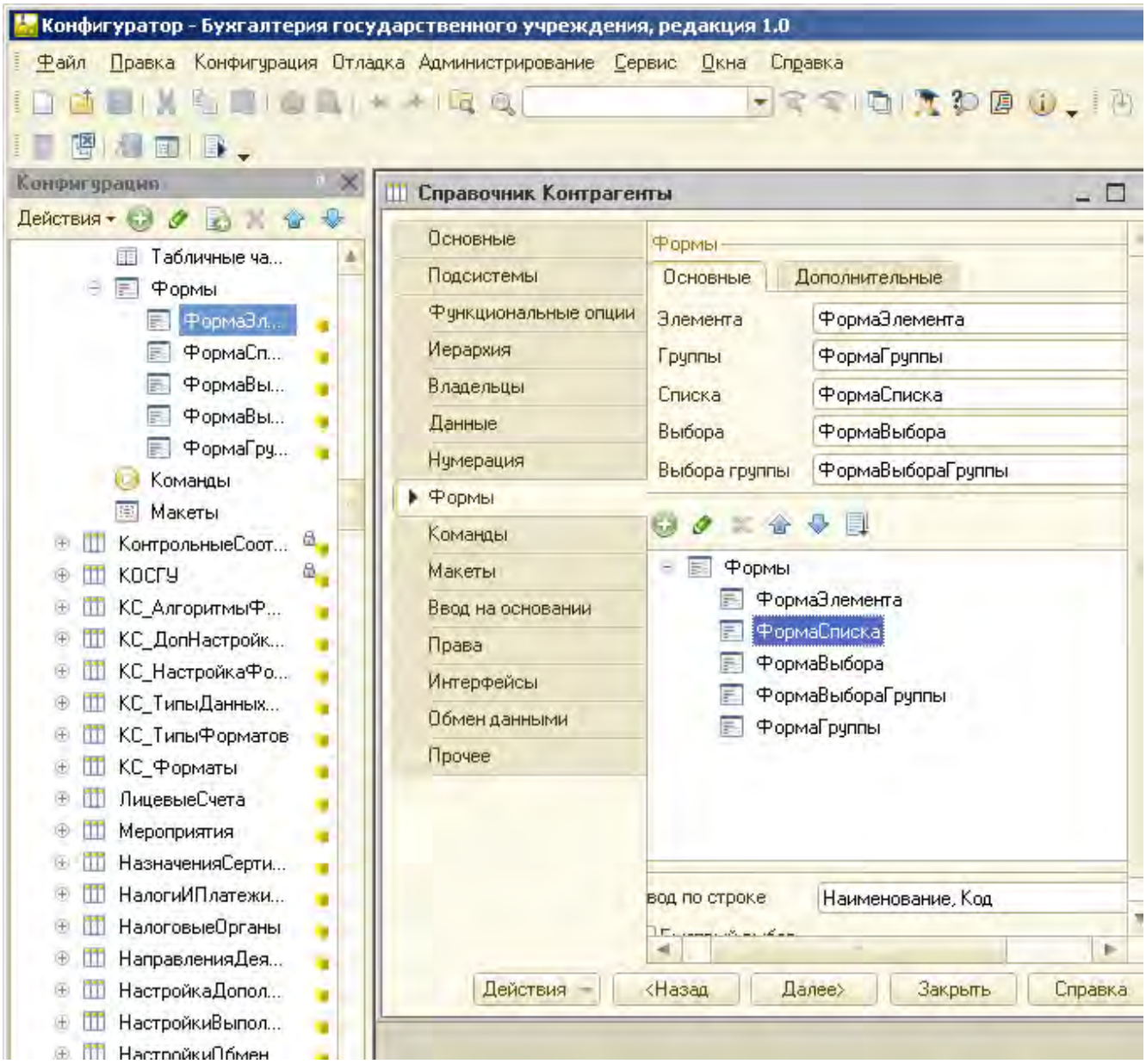


Рисунок 4.12. Формы справочника «Контрагенты» в реальной конфигурации.

Справочник содержит массу разных сведений:

ИНН, КПП, фамилия, имя, отчество, и т.д. Все это размещается на сложной форме с вкладками (рис.4.13).

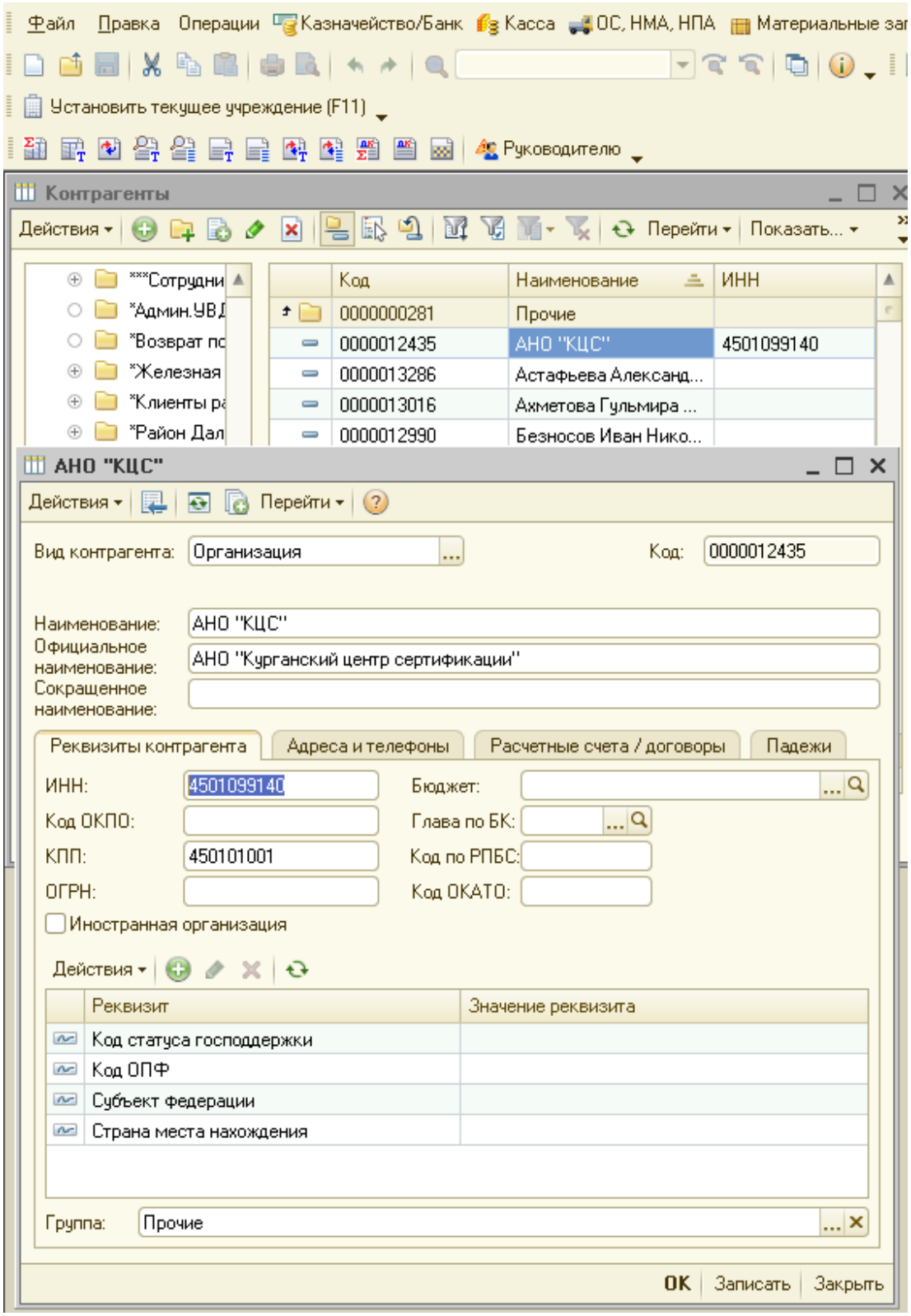


Рисунок 4.13. Содержание справочника «Контрагенты» в реальной конфигурации.

Для реализации такого справочника необходимо время, потому создадим более простой, чтобы понять сам процесс создания реквизитов и их заполнение.

Для начала сделаем справочник иерархическим (рис.4.14), в котором возможно упорядочение контрагентов по разным категориям, например, по району или области [4].

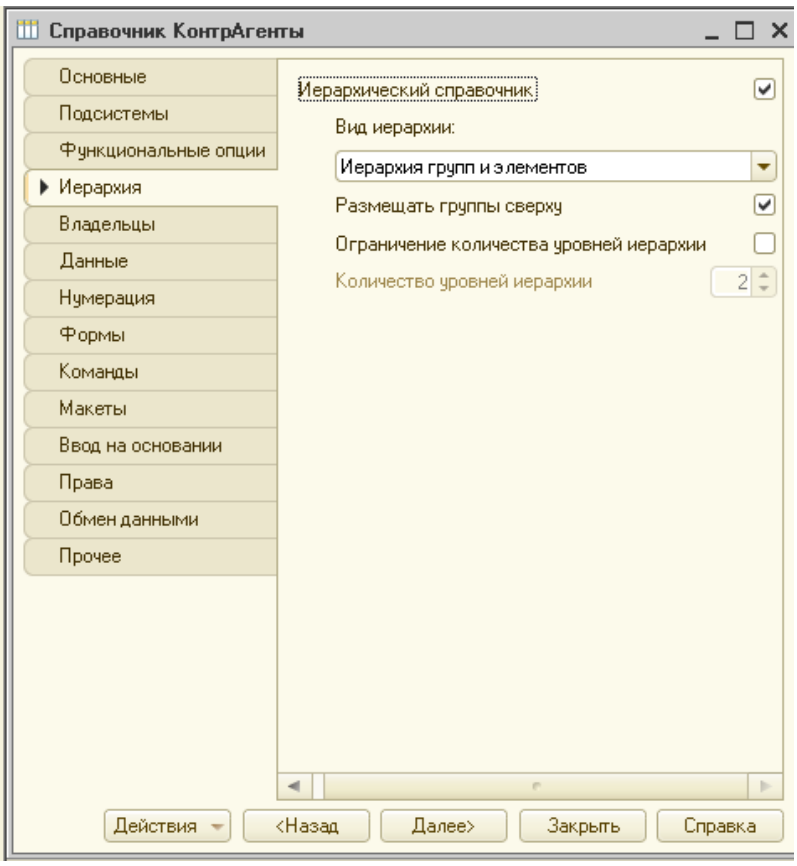


Рисунок 4.14. Придание иерархичности справочнику.

При редактировании данных можно изменить длину обязательных реквизитов, кода и наименования, а также их тип.

Для того, чтобы создать новый реквизит надо нажать на кнопку со значком + (рис.4.15).

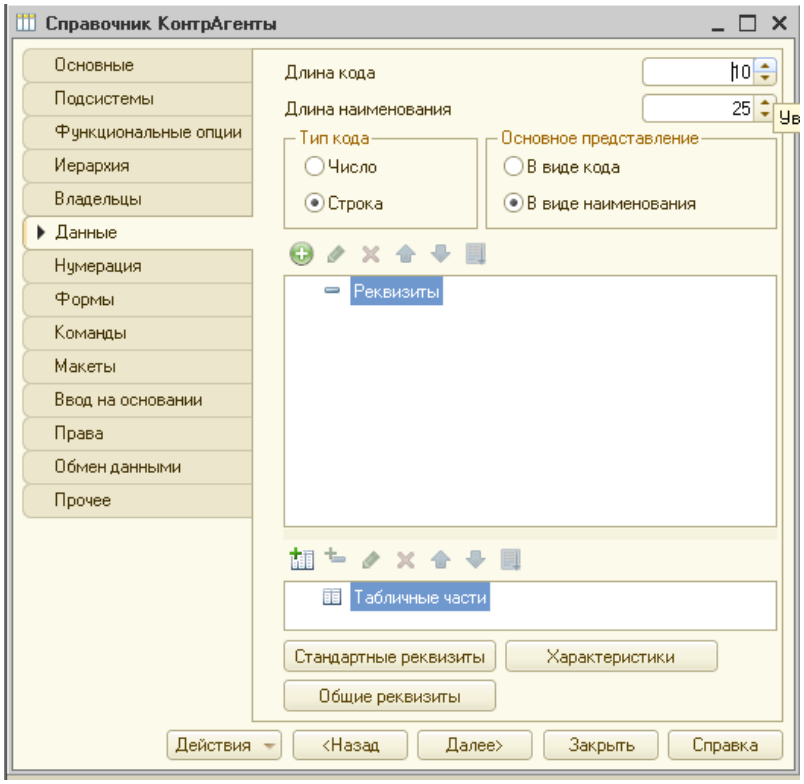


Рисунок 4.15. Вкладка данные.

Заведем простой строковый реквизит ИНН с длиной 12 символов (рис.4.16).

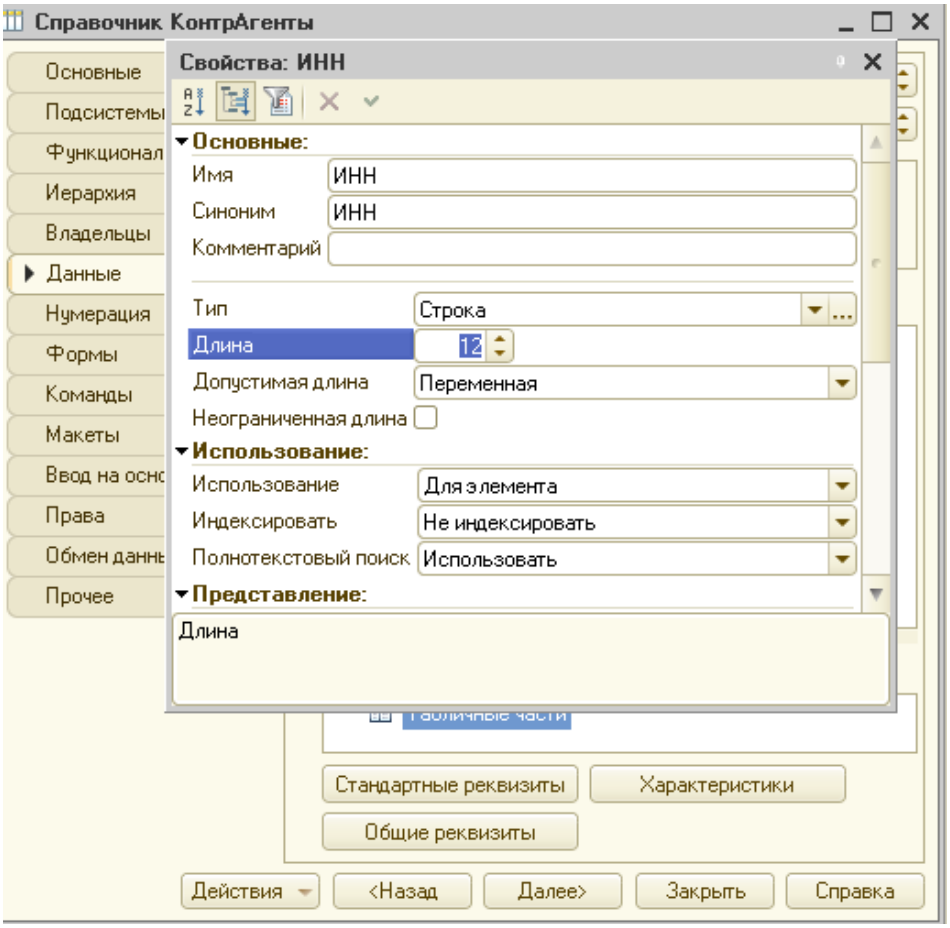


Рисунок 4.16. Окно свойств реквизита.

Элемент справочника может быть ссылкой на другие объекты конфигурации. Рассмотрим создание элемента - «ВидКонтрагента», который будет ссылаться на перечисление, т.е. будет принимать некоторое значение из заранее предопределенного списка.

На первом шаге создадим перечисление (рис.4.17).

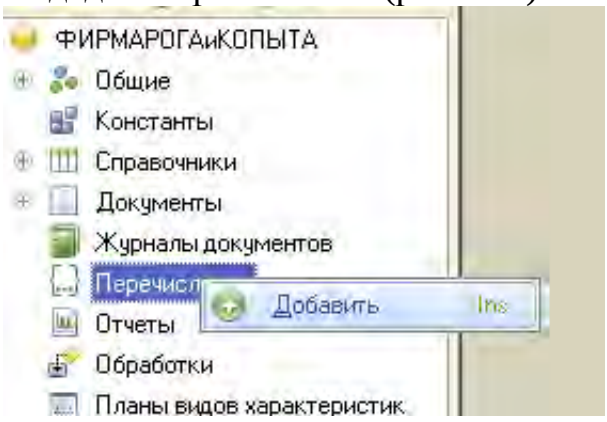


Рисунок 4.17. Добавление перечисления.

Имя перечисления будет «ВидКонтрагента» (рис.4.18)

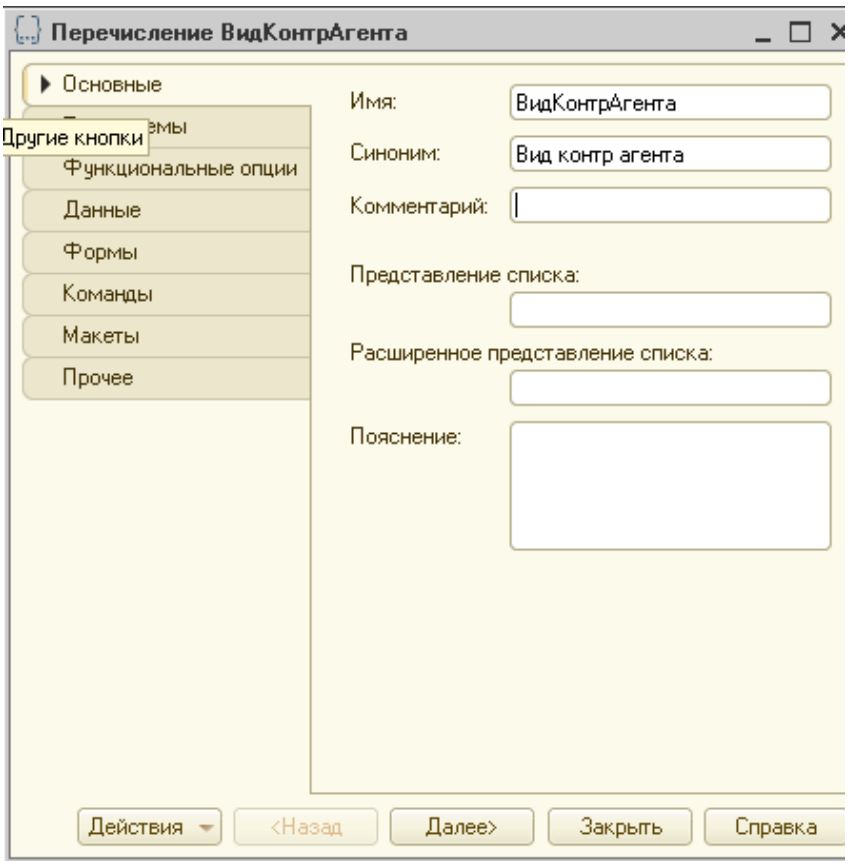


Рисунок 4.18. Создание перечисления.

Добавляем данные перечисления. Их будет два - «Физическое лицо» и «Предприятие» (рис.4.19).

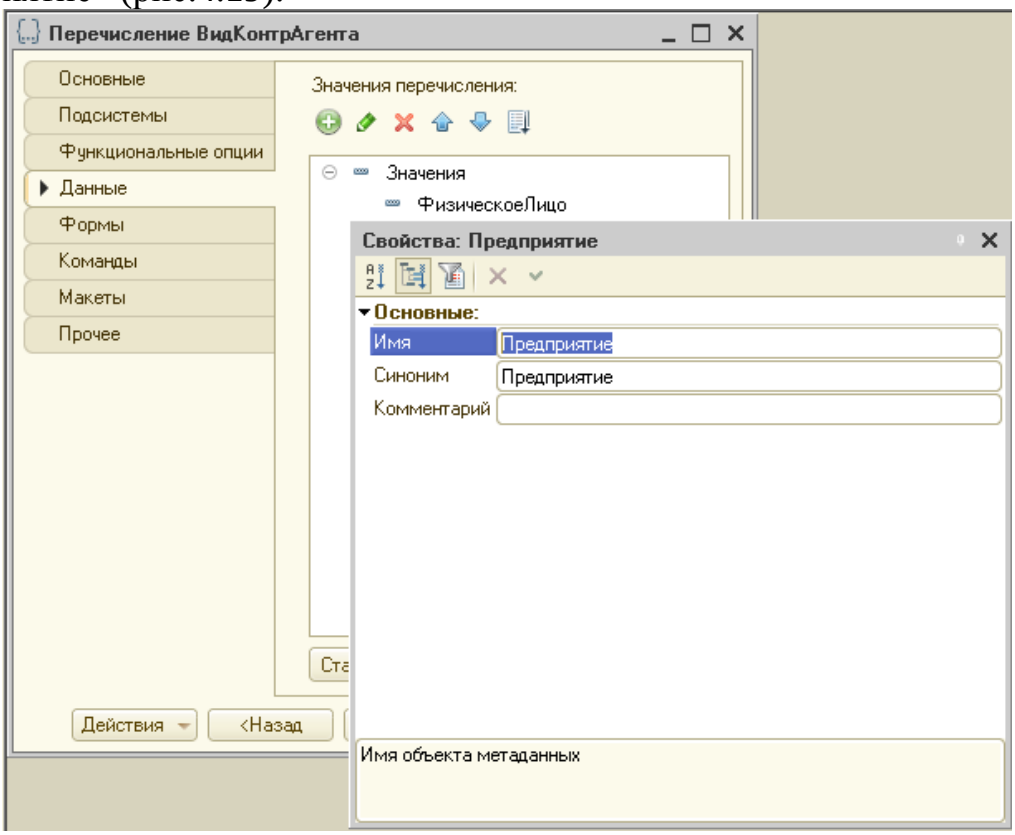


Рисунок 4.19. Задание перечисления.

Вернемся к справочнику контрагентов и создадим реквизит «ВидКонтрагента» (рис.4.20).

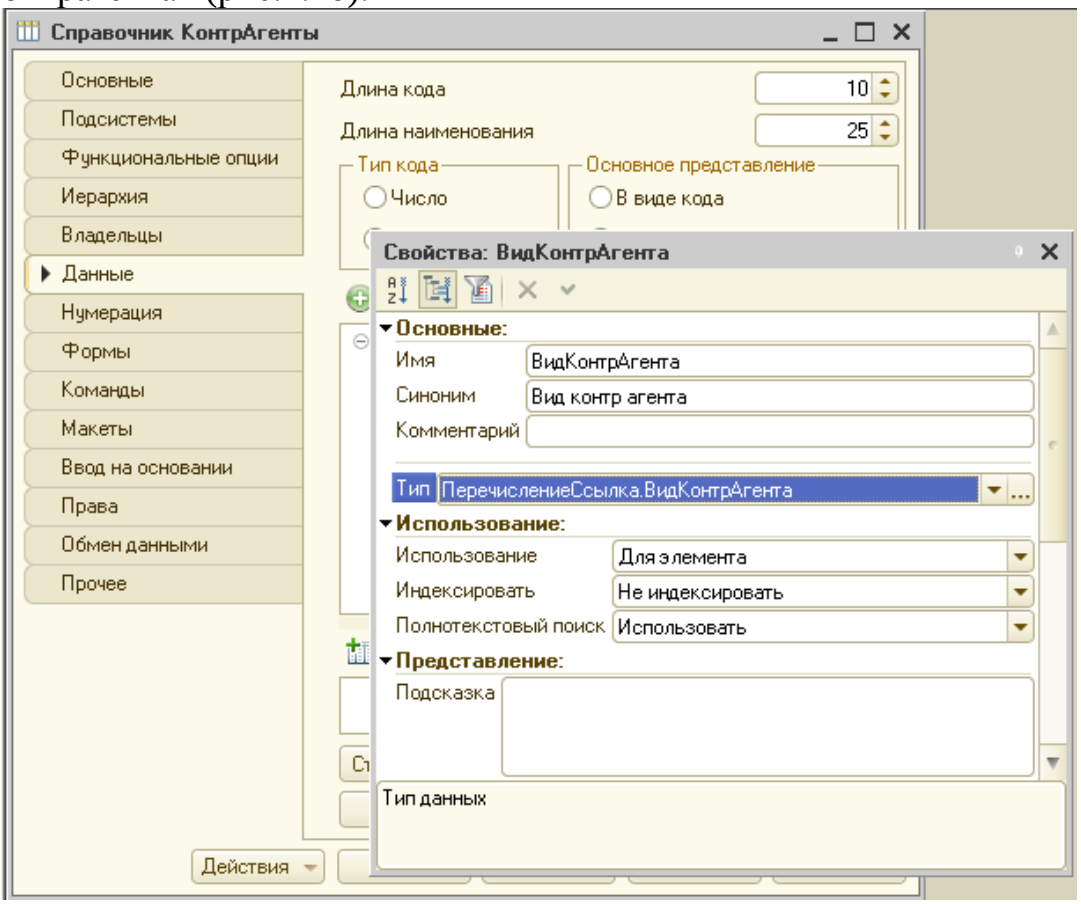


Рисунок 4.20. Создание реквизита с типом перечисление.

Рассмотрим функционирование в режиме 1С Предприятие.

Начинаем заполнять справочник (рис.4.21) с создания групп, так как наш справочник - иерархический. Создадим сначала две группы по названию населенных пунктов: Курган и Кетово, и будем помещать туда контрагентов по территориальному признаку (рис.4.22-4.23).

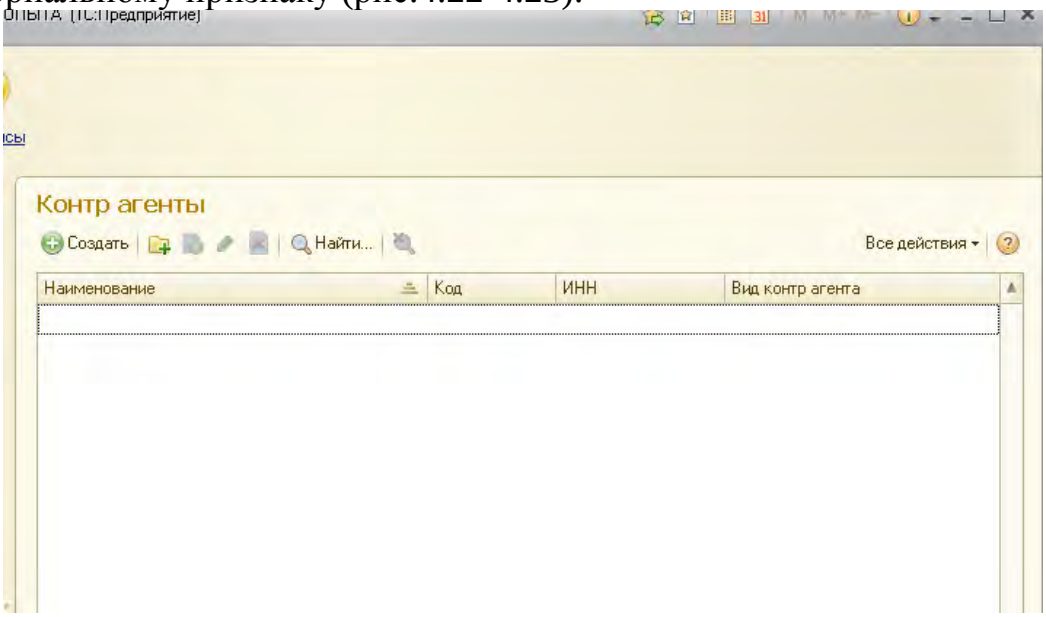


Рисунок 4.21. Вид пустого справочника Контрагенты.

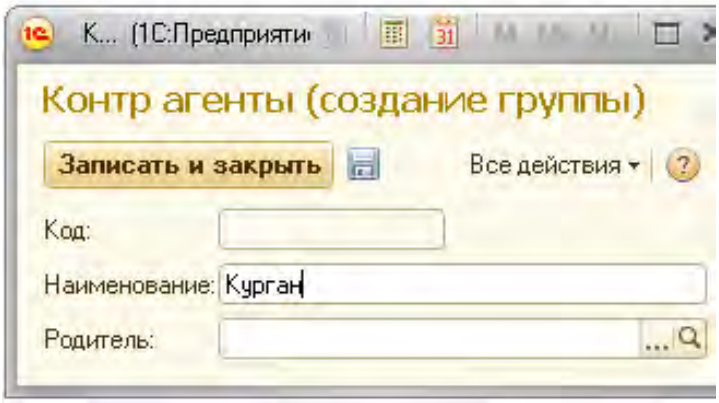


Рисунок 4.22. Создание группы.

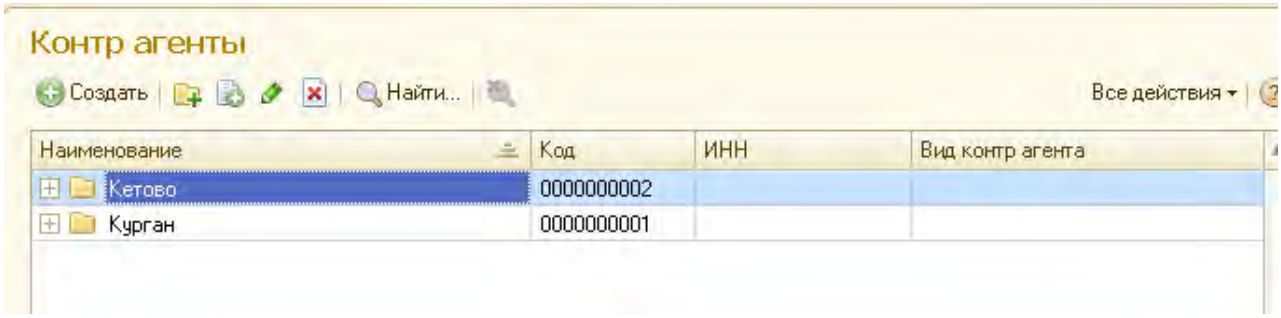


Рисунок 4.23. Созданы две группы.

Заполним группы контрагентами, одни будут физическими лицами, другие Предприятиями (рис.4.24).

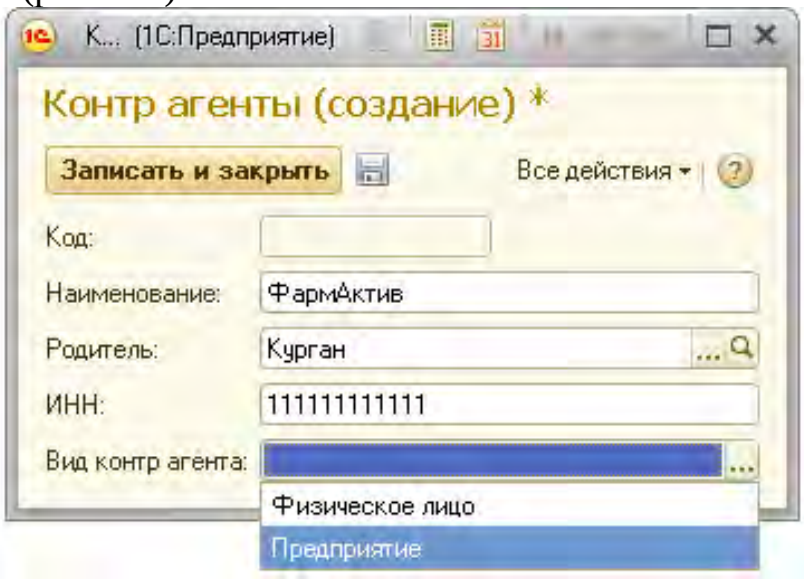


Рисунок 4.24. Создание контрагента из Кургана.

Теперь справочник наполнен некоторым количеством строк (рис.4.25).

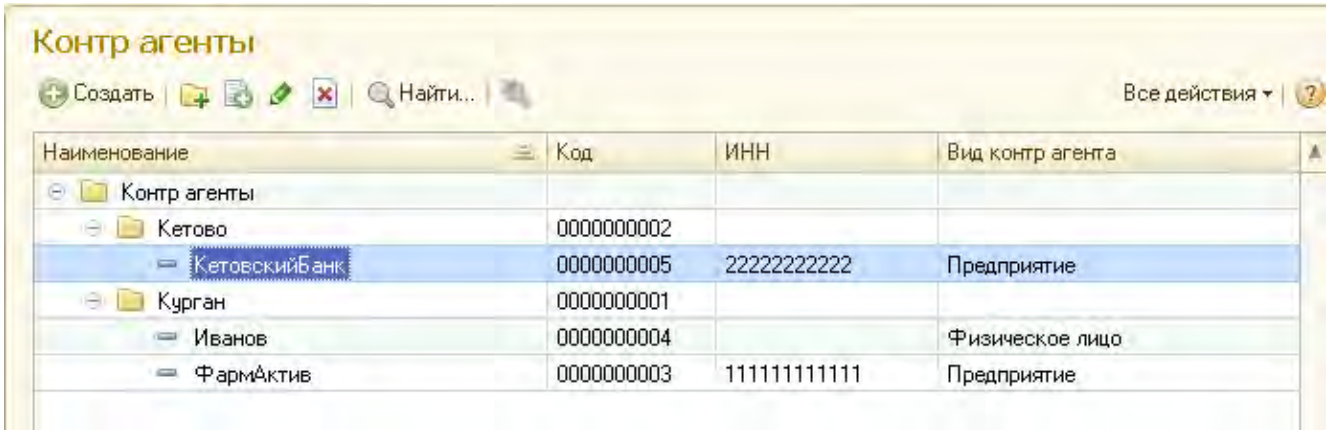


Рисунок 4.25 Содержимое справочника Контрагенты.

Нажав кнопку «Все действия», мы можем посмотреть все возможности, которые предоставляет нам стандартная форма для действий над элементами справочника и настроить представление его в более удобный вид (рис.4.26).

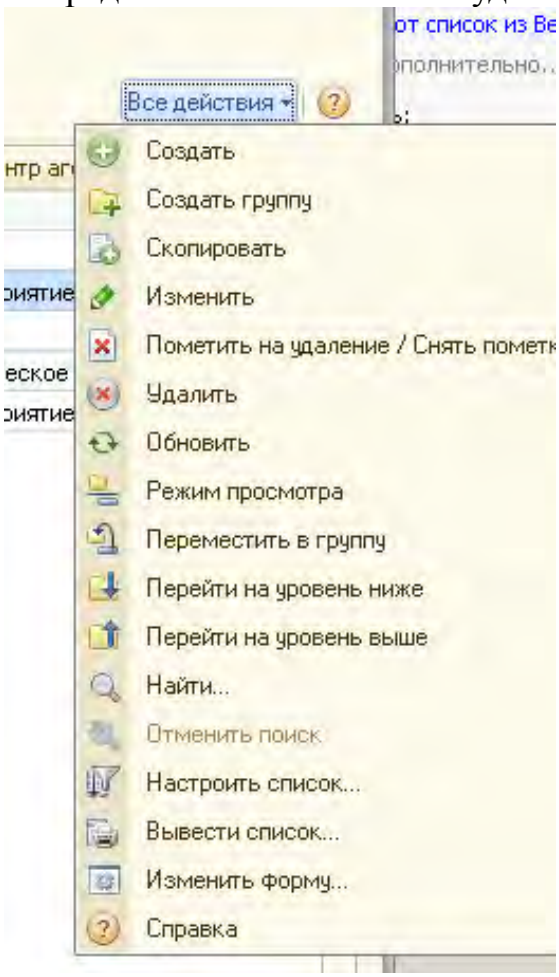


Рисунок 4.26. Стандартные возможности по изменению справочника.

В данном справочнике целесообразно иметь такой реквизит как «Контактное лицо» и где-то хранить сведения о человеке, с которым мы будем общаться. Можно, конечно, завести эту информацию прямо в справочнике «Контрагенты». Однако информация о людях может быть использована и в других местах нашей конфигурации, потому целесообразно завести отдельный

справочник «Физические лица». В справочнике будем хранить и собирать всю информацию о людях, с которыми приходится контактировать (рис.4.27).

Создаем данный справочник известным нам образом, и заводим для него реквизиты.

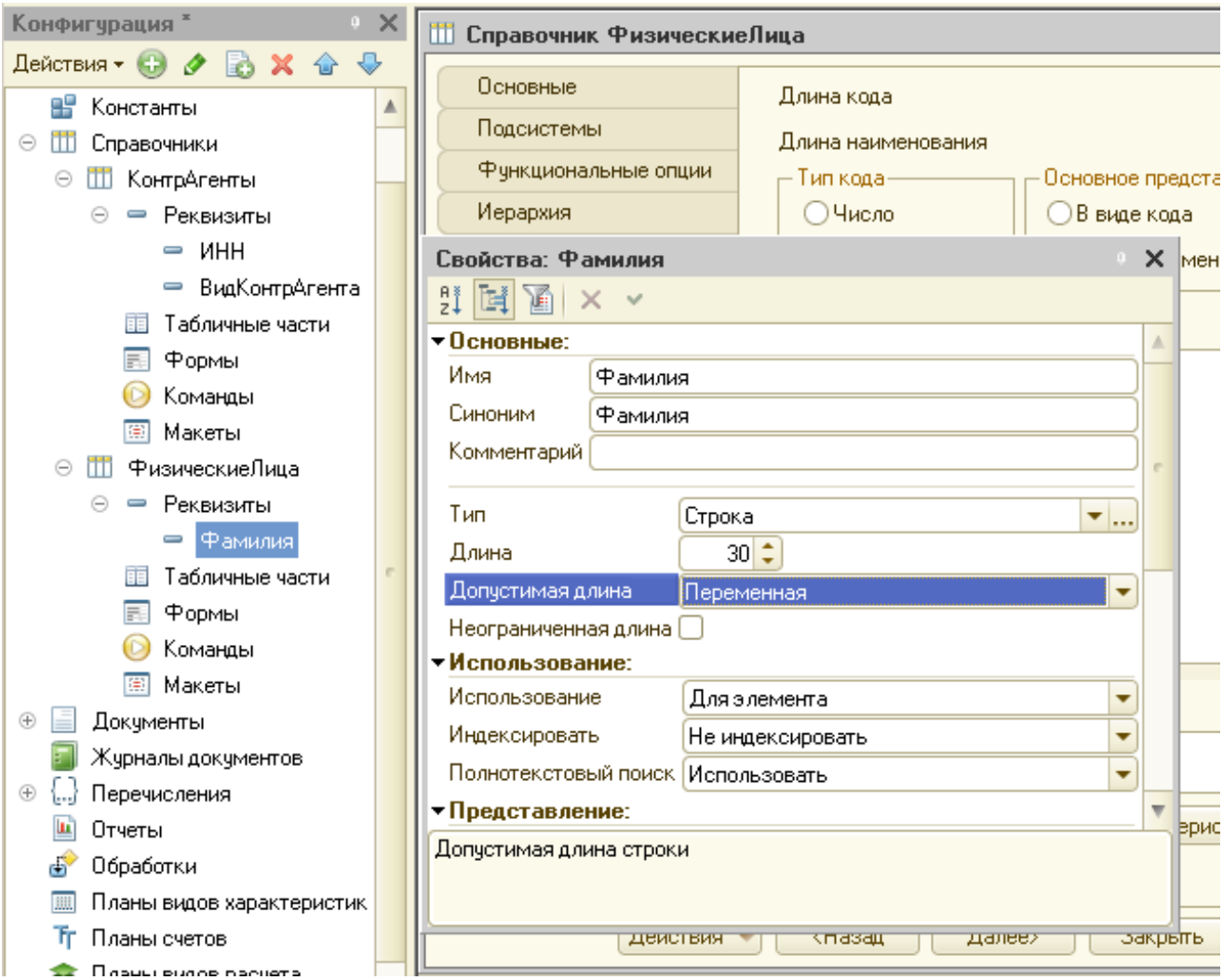


Рисунок 4.27. Создание справочника «Физические лица».

Заводим минимальный перечень реквизитов:
Фамилия, Имя, Отчество,
ИНН,
ДатаРождения,
Паспорт,
СтраховойНомерПНФ,
МестоРождения (рис.4.28-4.29).

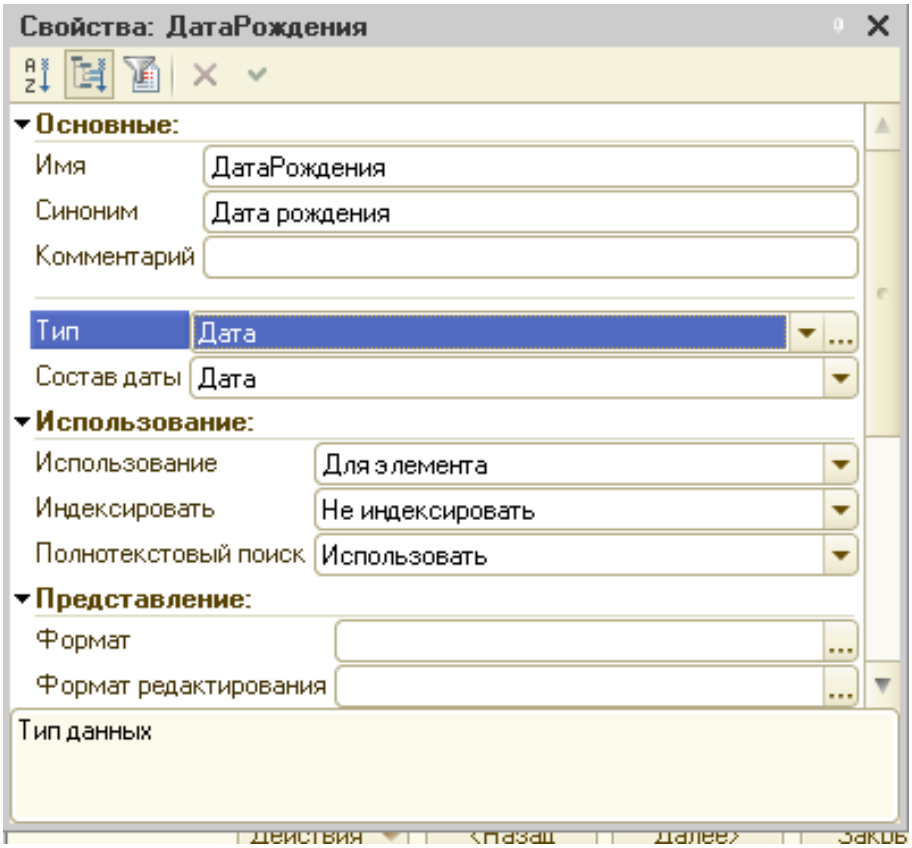


Рисунок 4.28. Реквизит дата рождения.

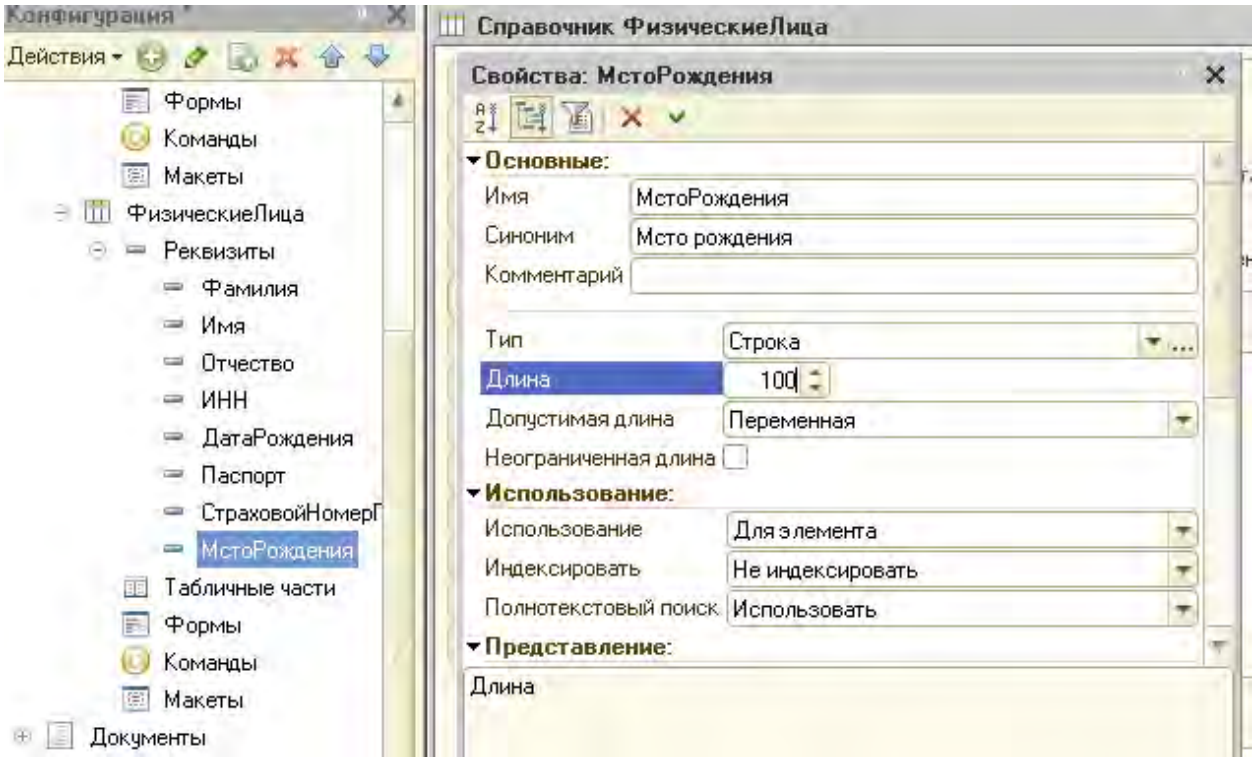


Рисунок 4.29 Реквизиты справочника «Физические лица».

Для учебного примера реквизитов вполне достаточно. Теперь создадим табличную часть «Трудовая деятельность». В ней будем хранить все известные сведения о месте работы, прошлом и нынешнем (рис.4.30-4.31).

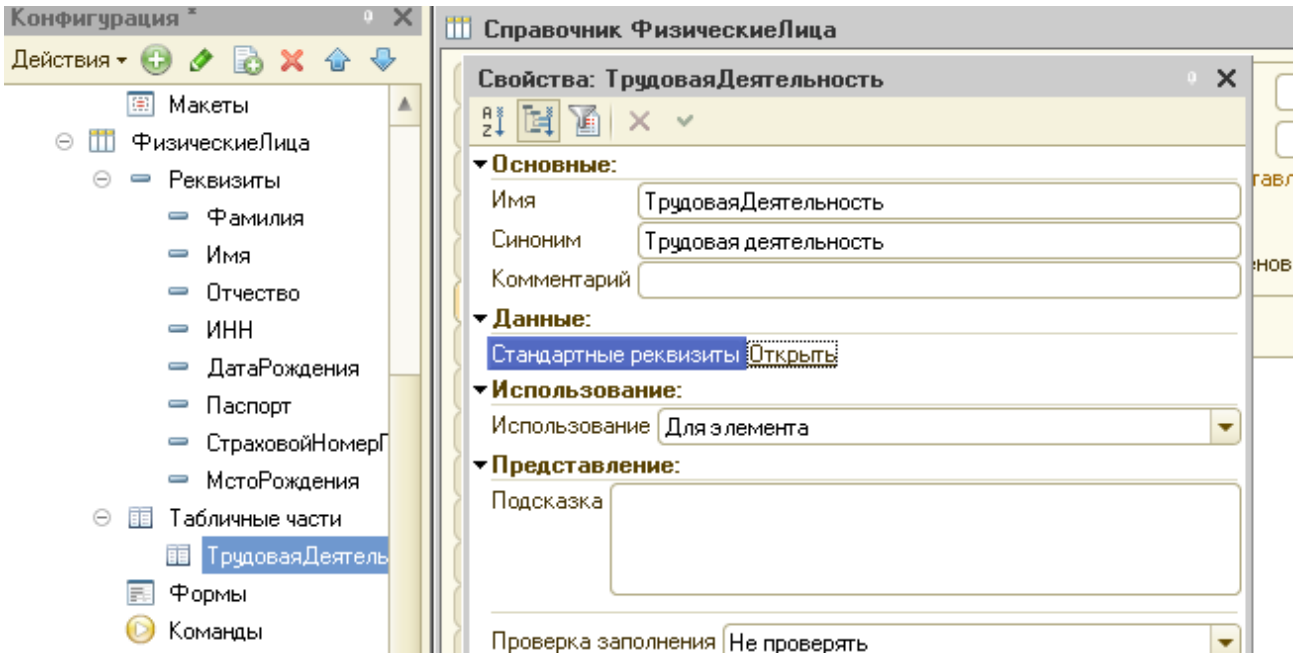


Рис.4.30. Создание табличной части

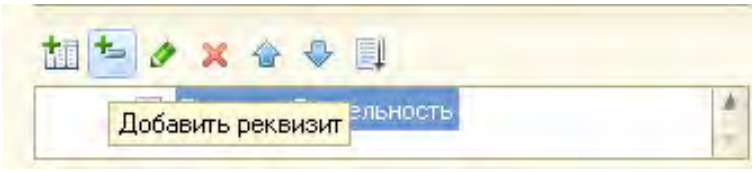


Рисунок 4.31. Добавление реквизитов в табличную часть.

В этом примере реквизитами могут быть: «Организация», «Должность», «ДатаНачалаработы», «ДатаОкончанияРаботы» (рис.4.32-4.33)

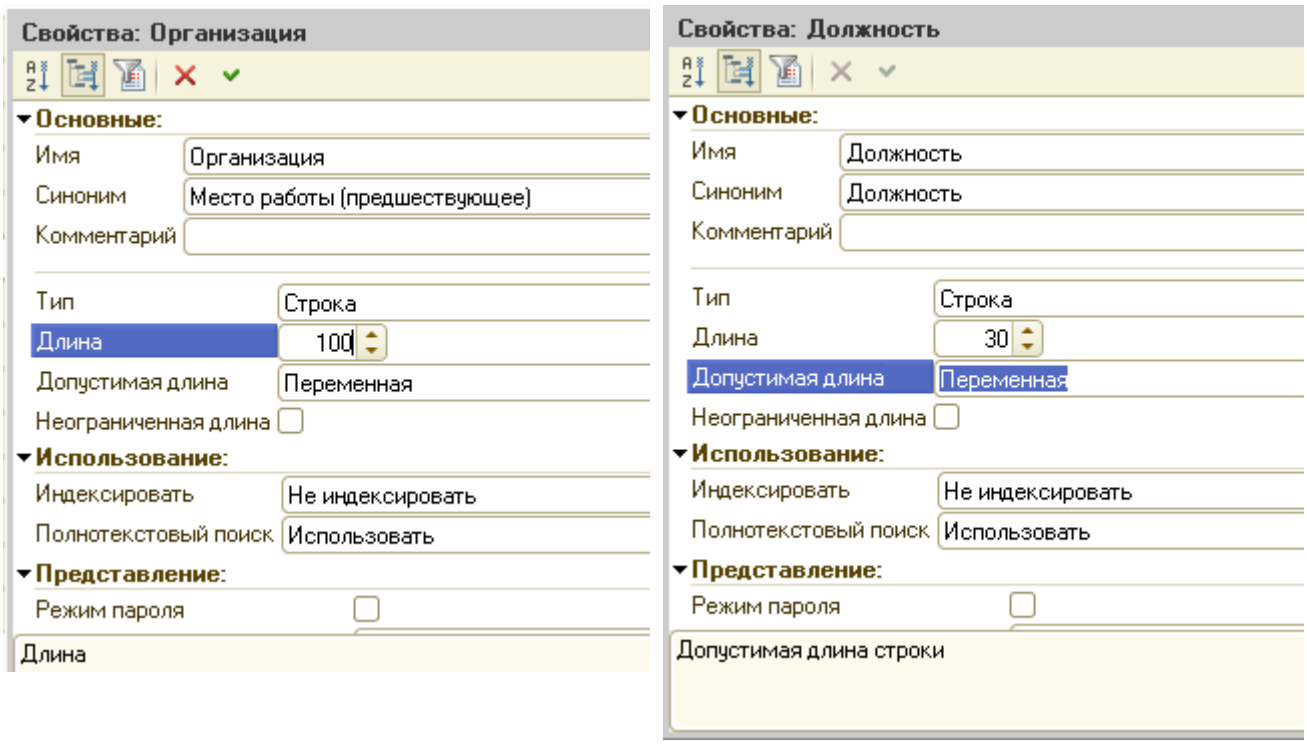


Рисунок 4.32. Реквизиты табличной части.

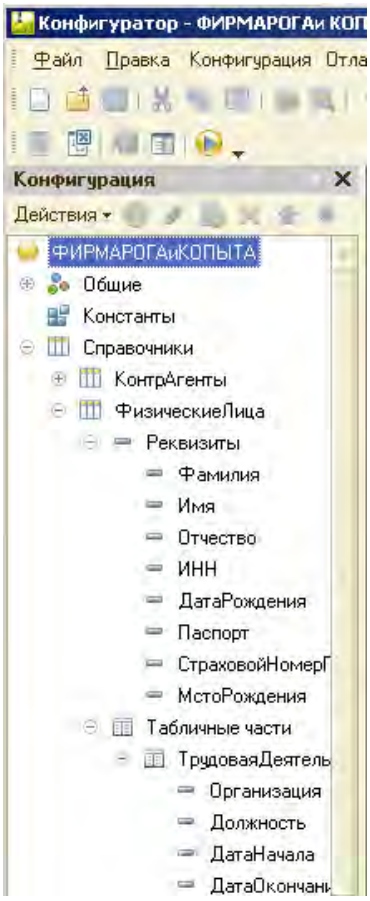


Рисунок 4.33. Все реквизиты справочника «Физические лица».

Теперь в справочнике «Контрагенты» достаточно указать реквизит «Контактное лицо», заставить его ссылаться на справочник физические лица, и информация будет связана с данным «Контрагентом» (рис.4.34).

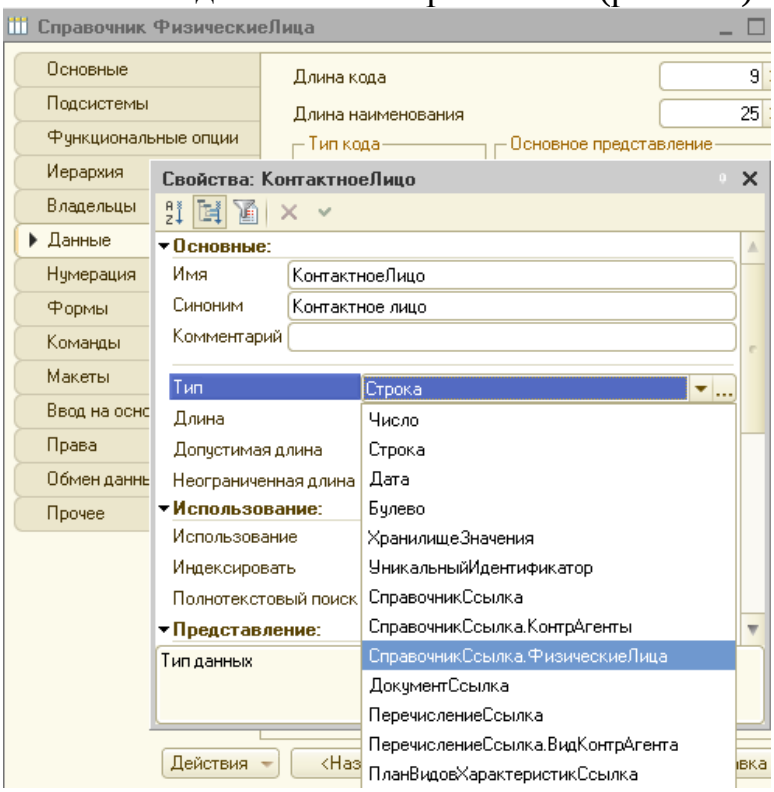


Рисунок 4.34. Создание реквизита типа справочник.

Запускаем 1С: Предприятие и заполняем справочник физические лица (рис.4.35). После того как в справочнике «Физические лица» появится несколько записей, можно перейти в справочник «Контрагенты» и заполнить вновь созданный реквизит «контактное лицо», путем выбора его из справочника физических лиц (рис.4.36).

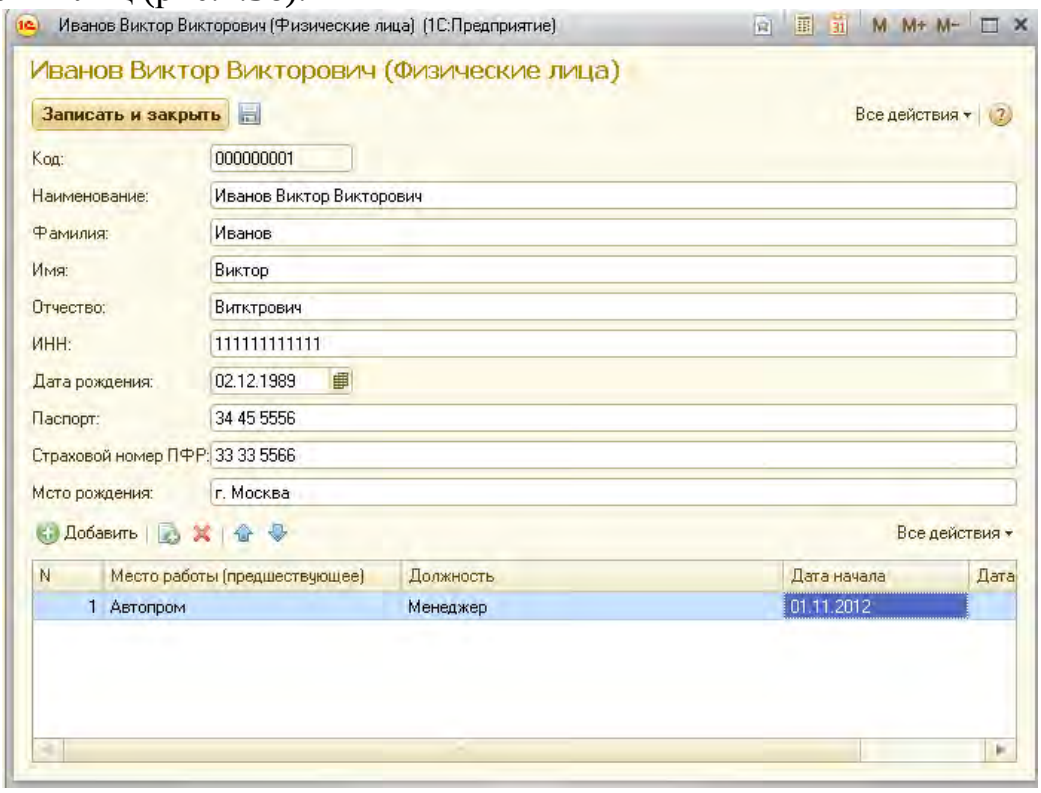


Рисунок 4.35. Заполнение справочника «Физические лица».

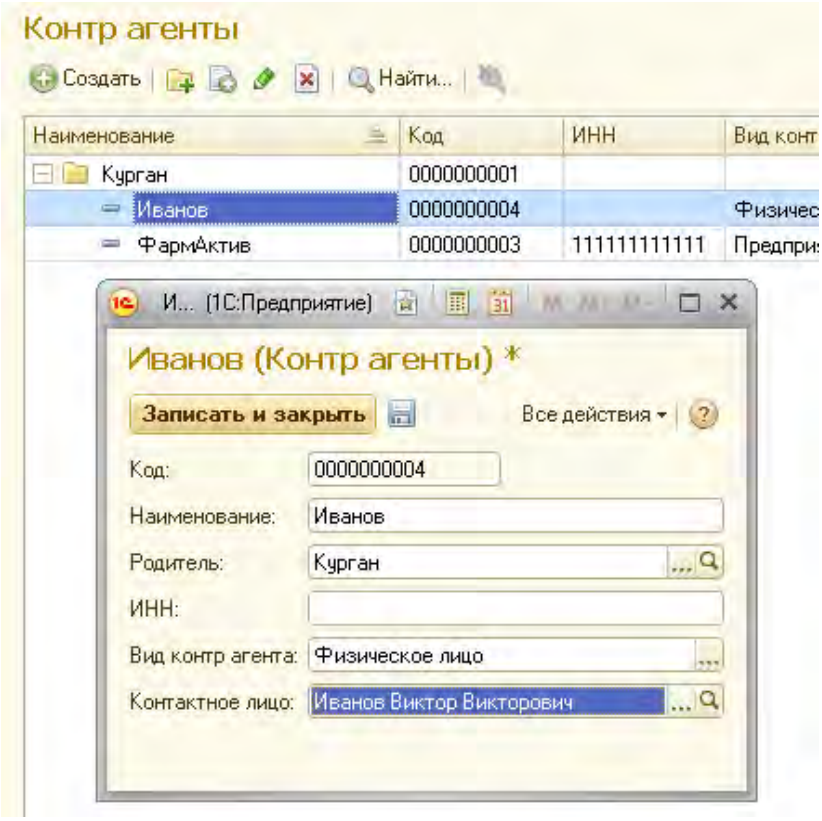


Рисунок 4.36. Редактирование справочника «Контрагенты».

До настоящего момента при редактировании и заполнении сведений в элементы справочника мы пользовались стандартными формами, которые предоставляет нам конфигурация. Рассмотрим возможность изменения этих форм или создания новых форм под наши требования. Например, мы хотим, чтобы некоторые сведения о контактном лице из справочника «Физические лица» отображались в форме справочника «Контрагенты».

Добавляем свою форму в список форм справочника (рис.4.37) «Контрагенты».

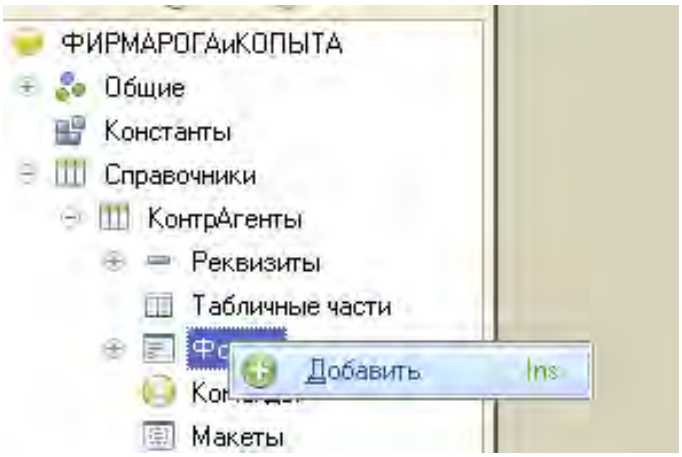


Рисунок 4.37. Создание формы.

Выбираем форму элемента справочника из множества предложенных форм (рис.4.38).

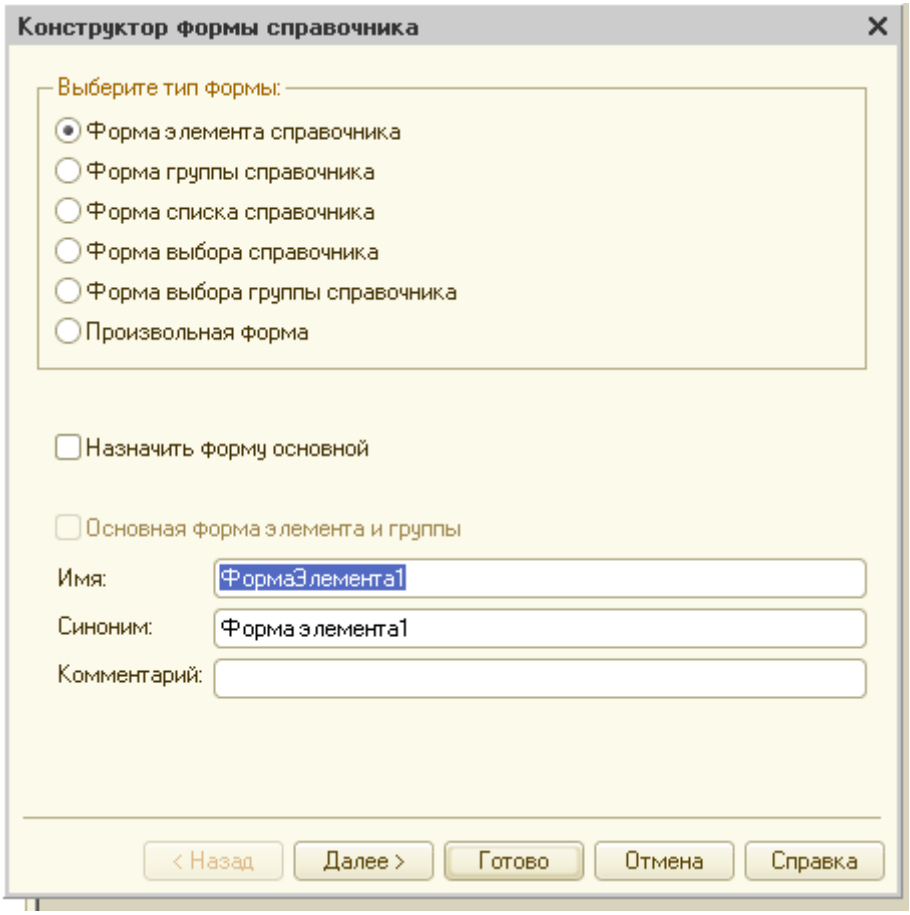


Рисунок 4.38 Форма элемента справочника.

Как видно из представленной формы, мы имеем возможность создавать большое количество разнообразных форм. После того, как определились с видом формы, переходим к ее конструированию. Из предложенного списка реквизитов выбираем те, что будут на ней отображаться (рис.4.39).

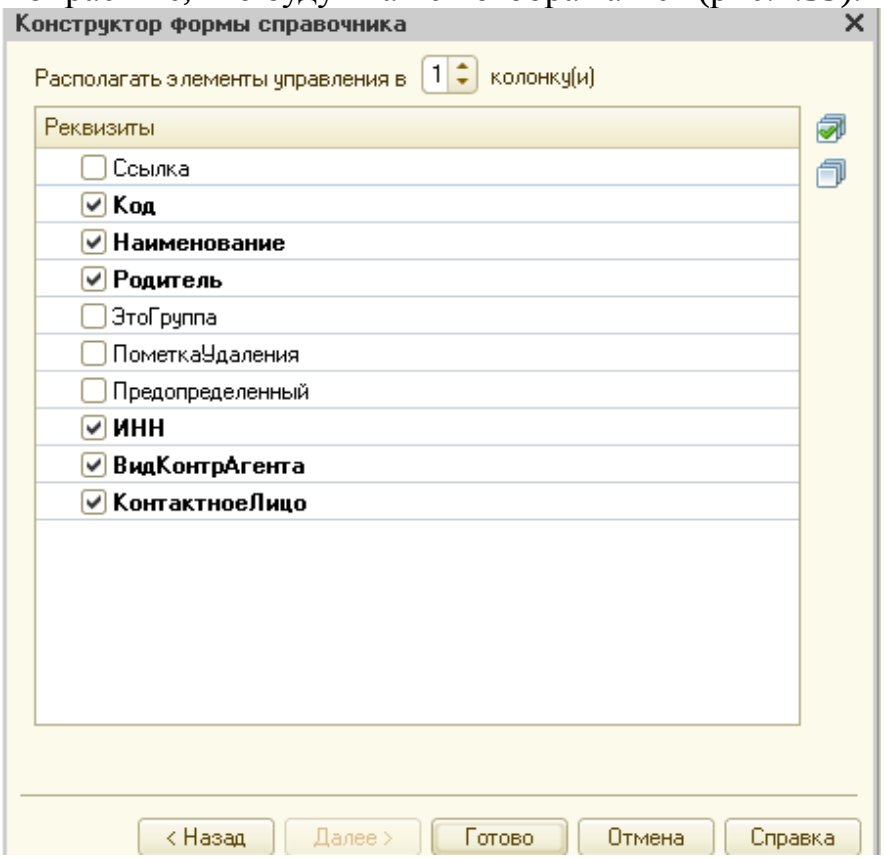


Рисунок 4.39. Выбор реквизитов справочника для отображения на форме

Далее в редакторе форм изменяем и дополняем предложенное системой окно формы (рис.4.40).

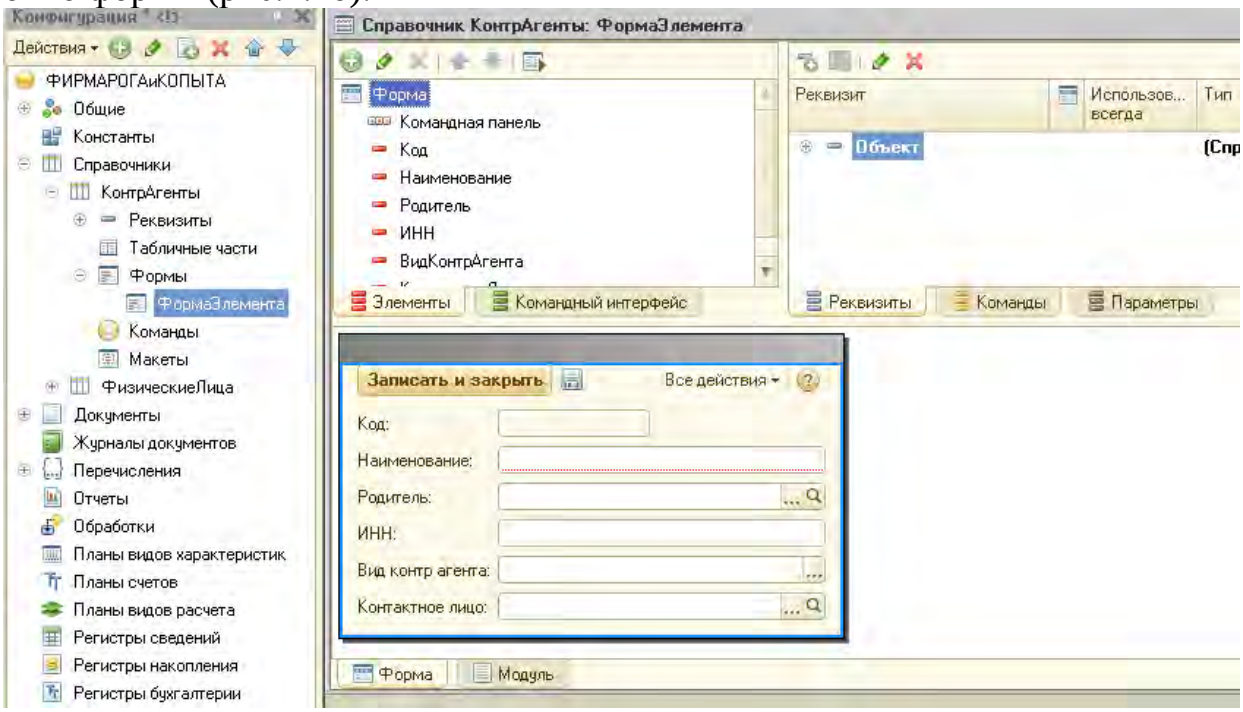


Рисунок 4.40. Окно редактирования формы.

Добавляем в наше окно формы надпись «Данные о контактном лице» (рис.4.41)

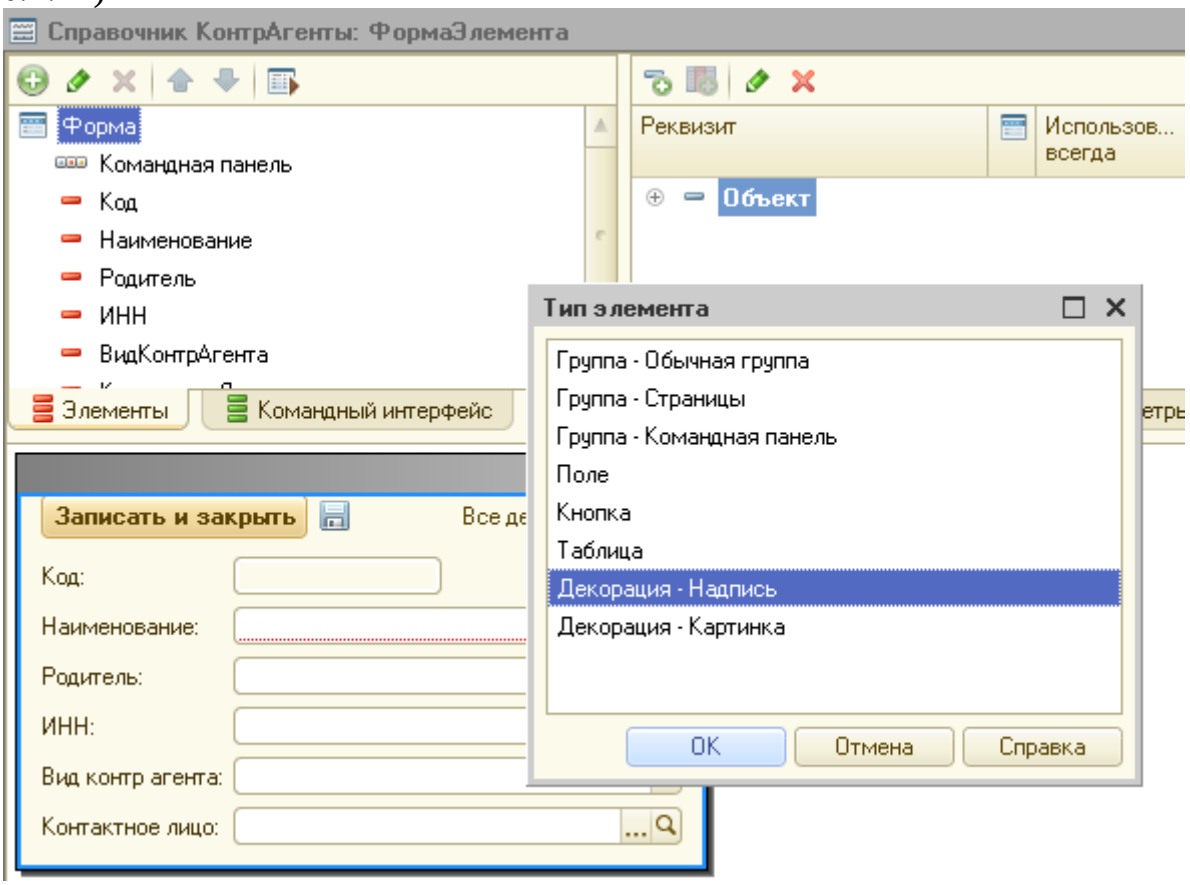


Рисунок 4.41. Добавление элемента на форму.

Добавляем элемент «Надпись» (рис.4.42).

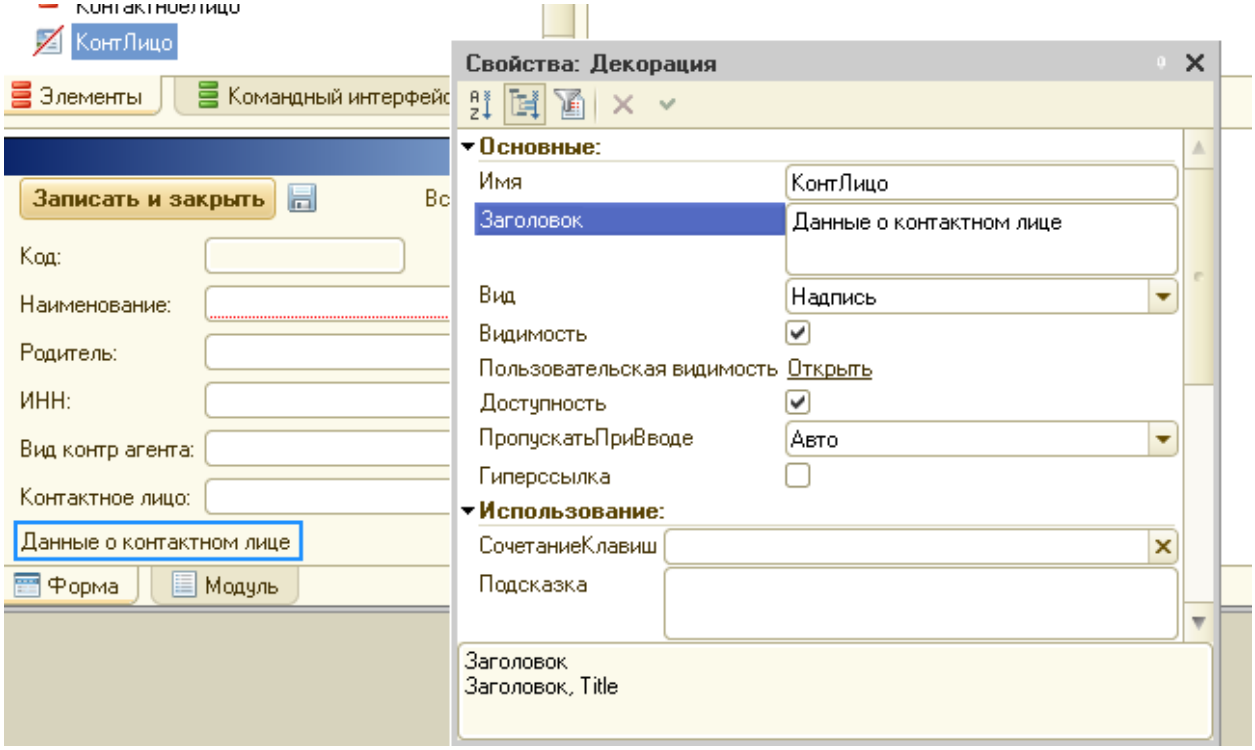


Рисунок 4.42. Элемент типа «Надпись»

Заполняем форму элементами типа «Поле», в которых будут отображаться сведения, полученные из справочника «Физические лица» (рис.4.43).

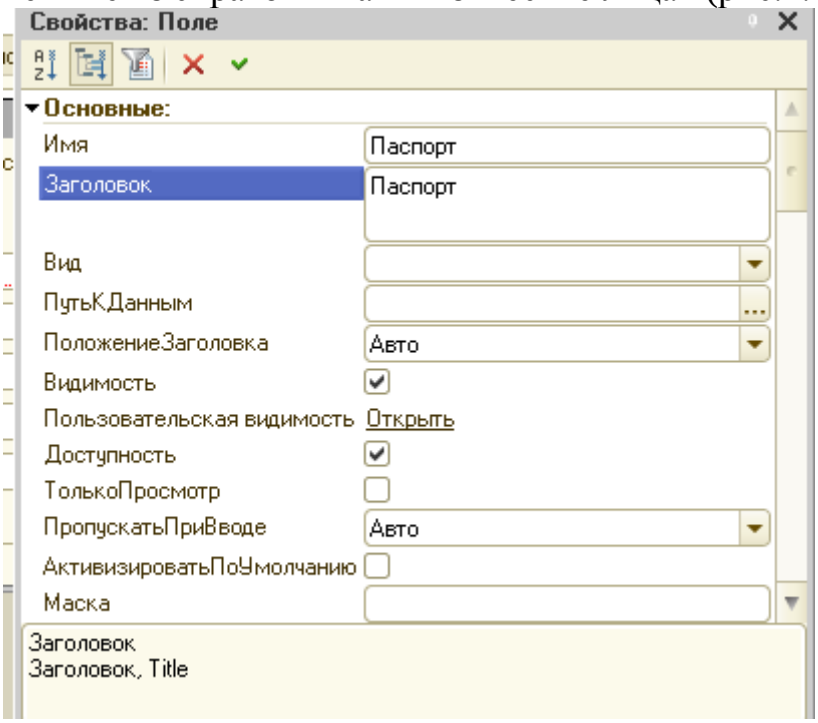


Рисунок 4.43 Элемент типа «Поле».

Здесь главное правильно показать путь к данным (рис.4.44).

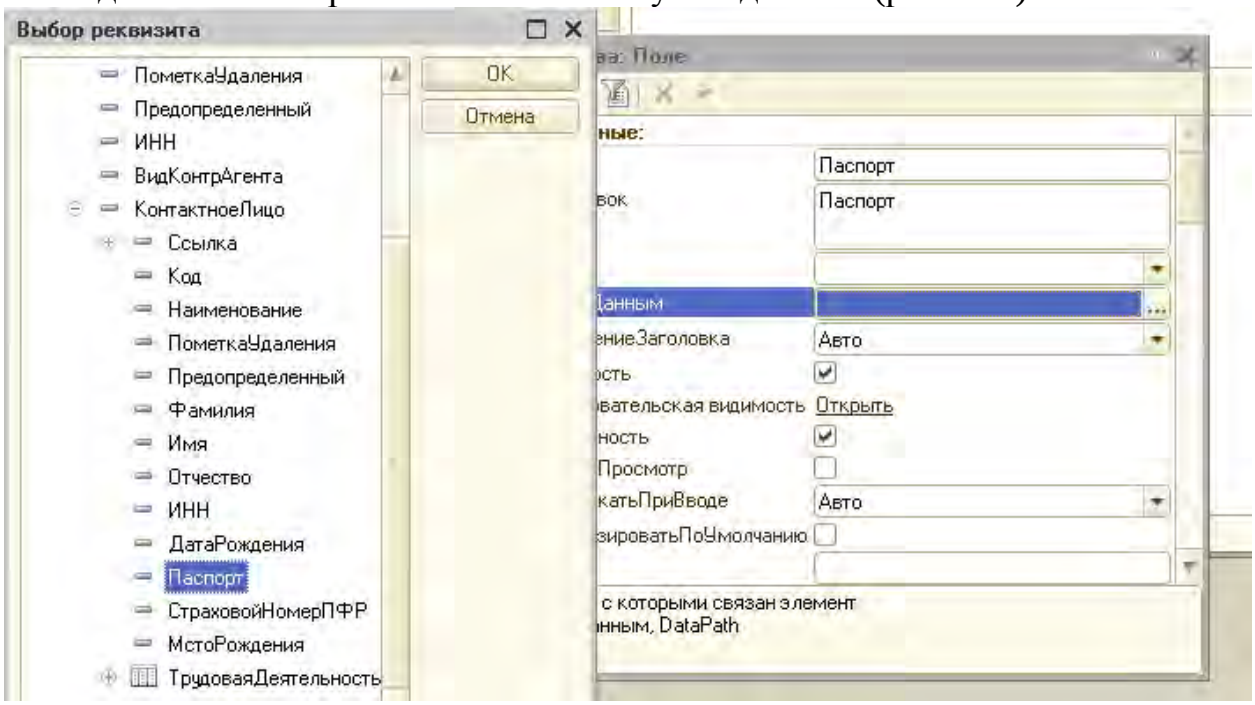


Рисунок 4.44. Выбор пути к данным.

После заполнения свойств нового поля, оно появится на разрабатываемой форме. В данном случае это поле предназначено для вывода номера паспорта. Аналогичным образом заполняем и другие поля, которые необходимы в нашей форме (рис.4.45-4.46).

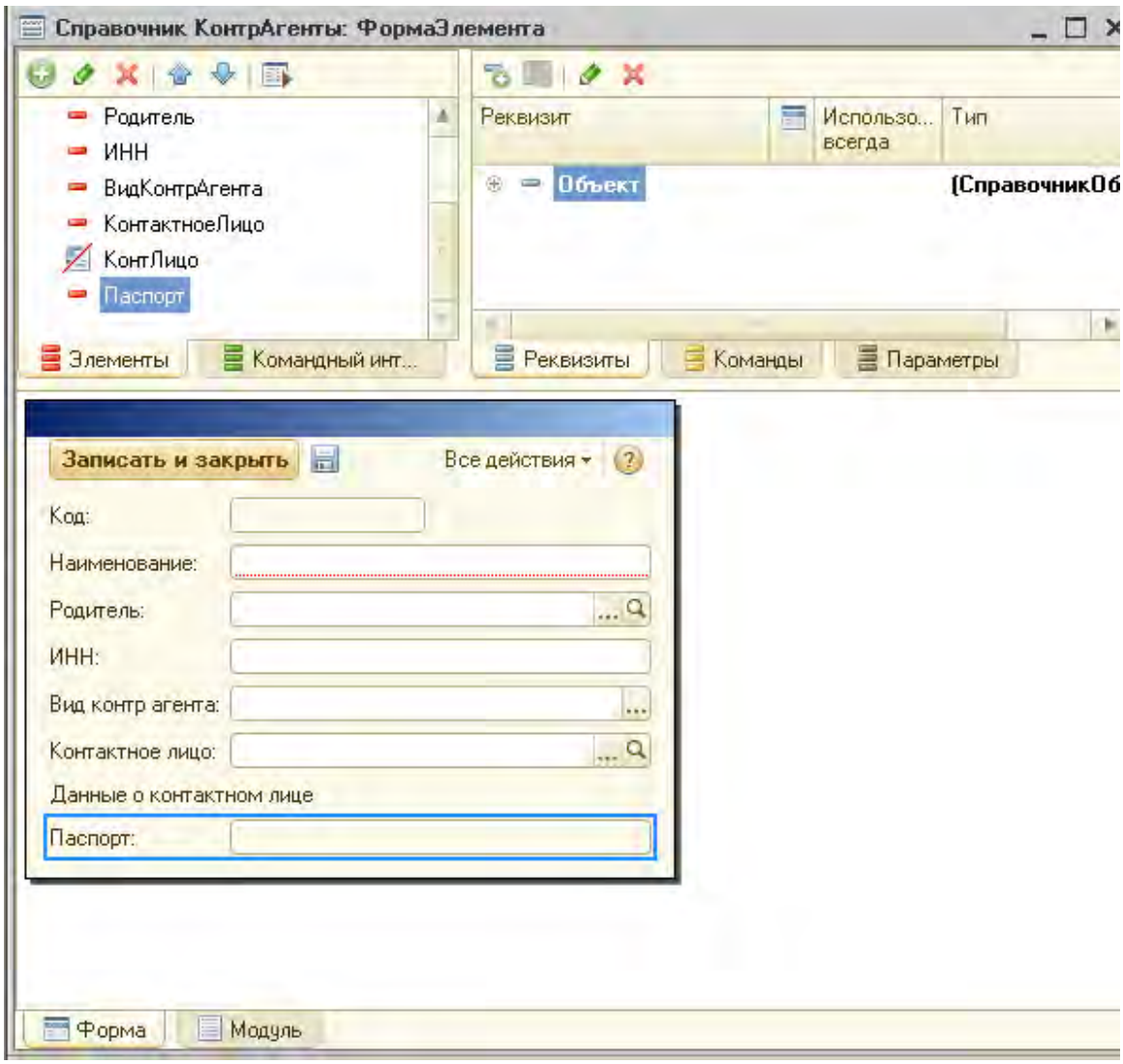


Рисунок 4.45. Новый вид формы.

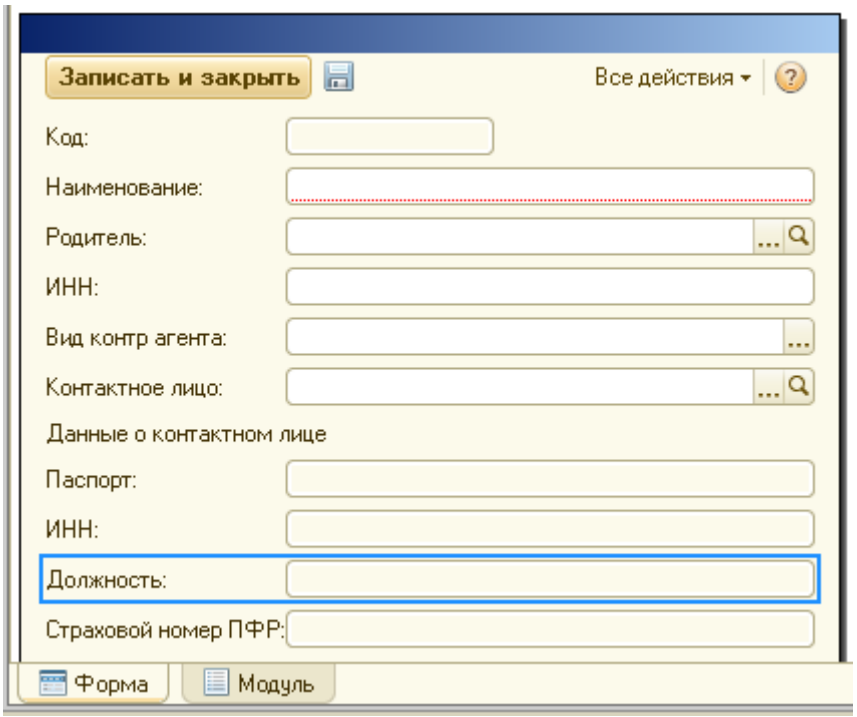


Рисунок 4.46. Форма с добавленными полями.

Работа в конфигураторе закончилась. Посмотрим, как все это выглядит в режиме 1С: Предприятие. Откроем справочник «Контрагенты» и заполним реквизит «Контактное лицо» для элемента «ФармАктив» (рис.4.47).

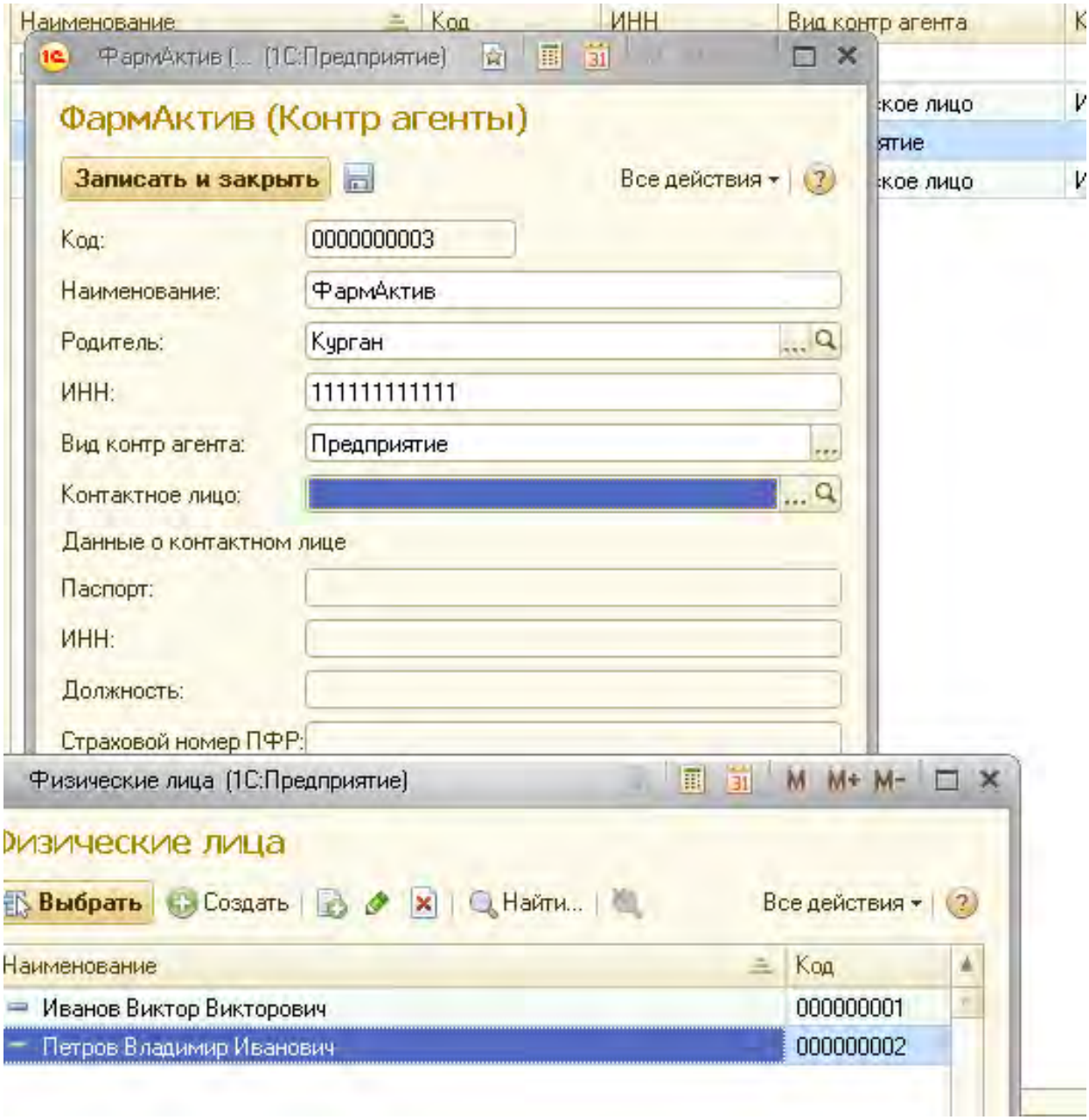


Рисунок 4.47. Выбор контактного лица фирмы.

После заполнения реквизита данные из справочника «Физические лица» появятся на нашей форме (рис.4.48).

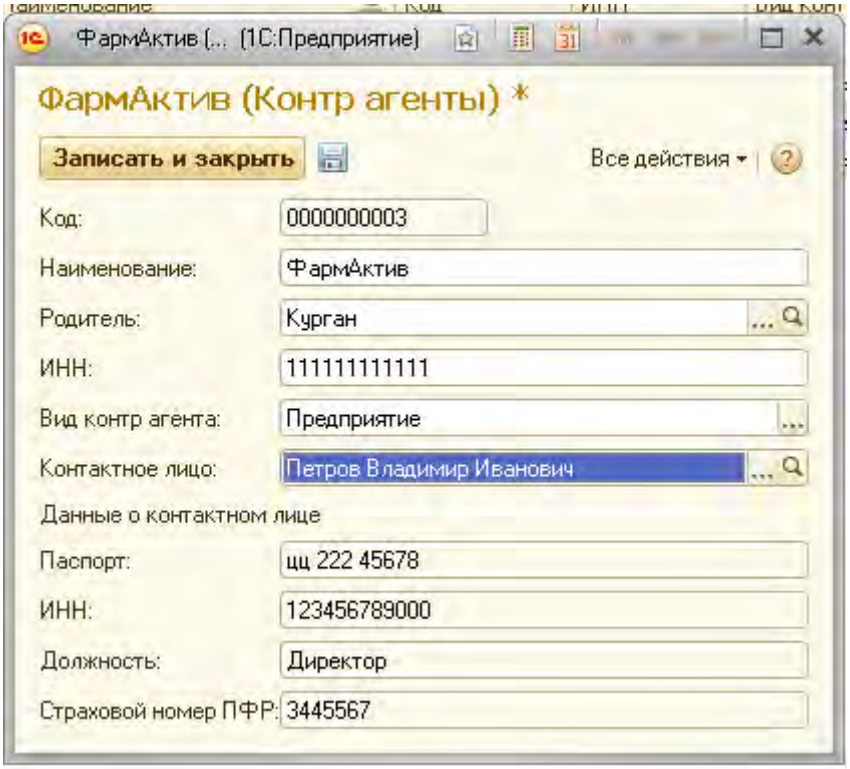


Рисунок 4.48. Отображение данных в нашей форме.

Вопросы для самоконтроля

1. Как создать новый объект конфигурации стандартного типа?
2. Для чего нужны объекты типа подсистема?
3. Что такое справочник в конфигурации?
4. Для чего используются объекты типа справочник?
5. Как отнести объект к подсистеме?
6. Как создать реквизит справочника?
7. Что может быть реквизитом справочника?
8. Для чего нужны таблицы в справочнике?
9. Как создать или изменить форму элемента справочника?
10. Для чего нужна форма элемента справочника?

Задание для самостоятельного выполнения

1. Создать справочник «Номенклатура» (рис.4.49).
2. Создать справочник «Организации» (рис.4.50).
3. Создать справочник «Основные средства» (рис.4.51)

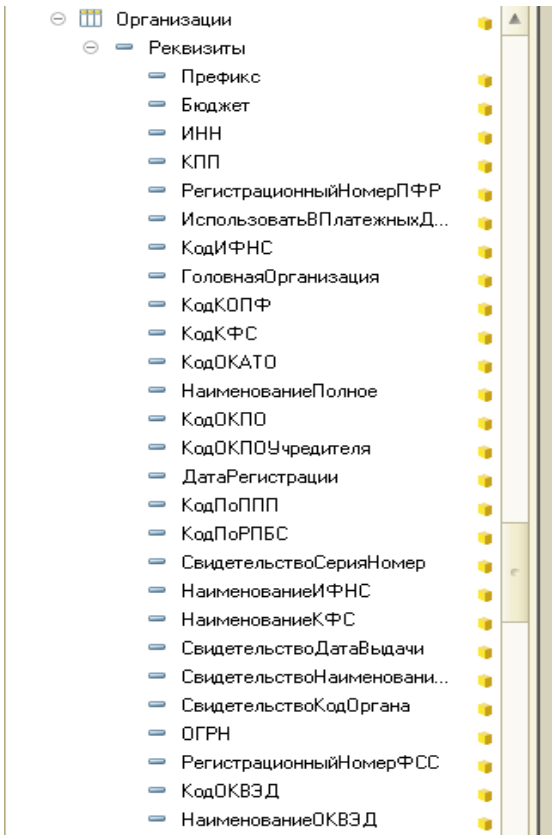


Рисунок 4.49. Примерный перечень реквизитов справочника «Номенклатура»

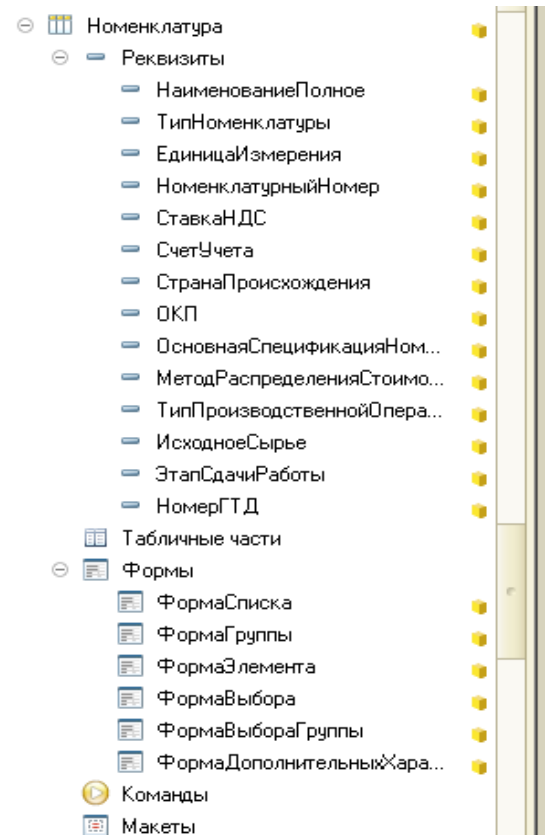


Рисунок 4.50. Примерный перечень реквизитов справочника «Организации».

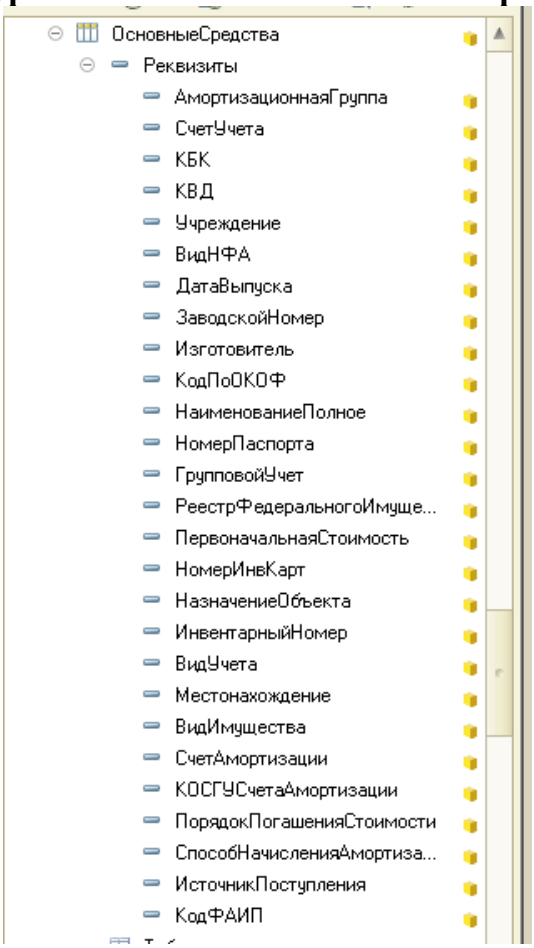


Рисунок 4.51. Примерный перечень реквизитов справочника «Основные средства»

5. Документы как объект конфигурации и учет их движения

Учет в любом предприятии строится на анализе движения документов. Так сложилось еще с давних времен, когда документы были бумажными. В документах отражается вся финансово-хозяйственная деятельность предприятия, поступление денег от реализации товаров и услуг, затраты на закупку необходимых материалов и оборудования, заработную плату, расчеты за услуги, оказываемые сторонними организациями и т.д. В документах также отражается и изменение в материальных активах предприятия. Можно сказать, что без документов невозможен учет на современном предприятии [1].

Объект конфигурации Документ предназначен для хранения информации из документов. Логика работы документа отличается от логики работы других объектов конфигурации, тем, что документ имеет способность проводиться. Факт проведения документа означает, что событие уже свершилось и повлияло на учет. Если документ не проведен, то состояние учета не изменилось и документ существует только в черновом варианте. Каждый документ привязан к конкретному моменту времени [4].

В базе данных документы представляют собой записи в основной таблице, хранящей информацию об этом виде документов.

В реальных конфигурациях присутствует огромное количество видов документов. Научимся создавать документы разных типов. Для начала создадим документ «Накладная» (рис.5.1)

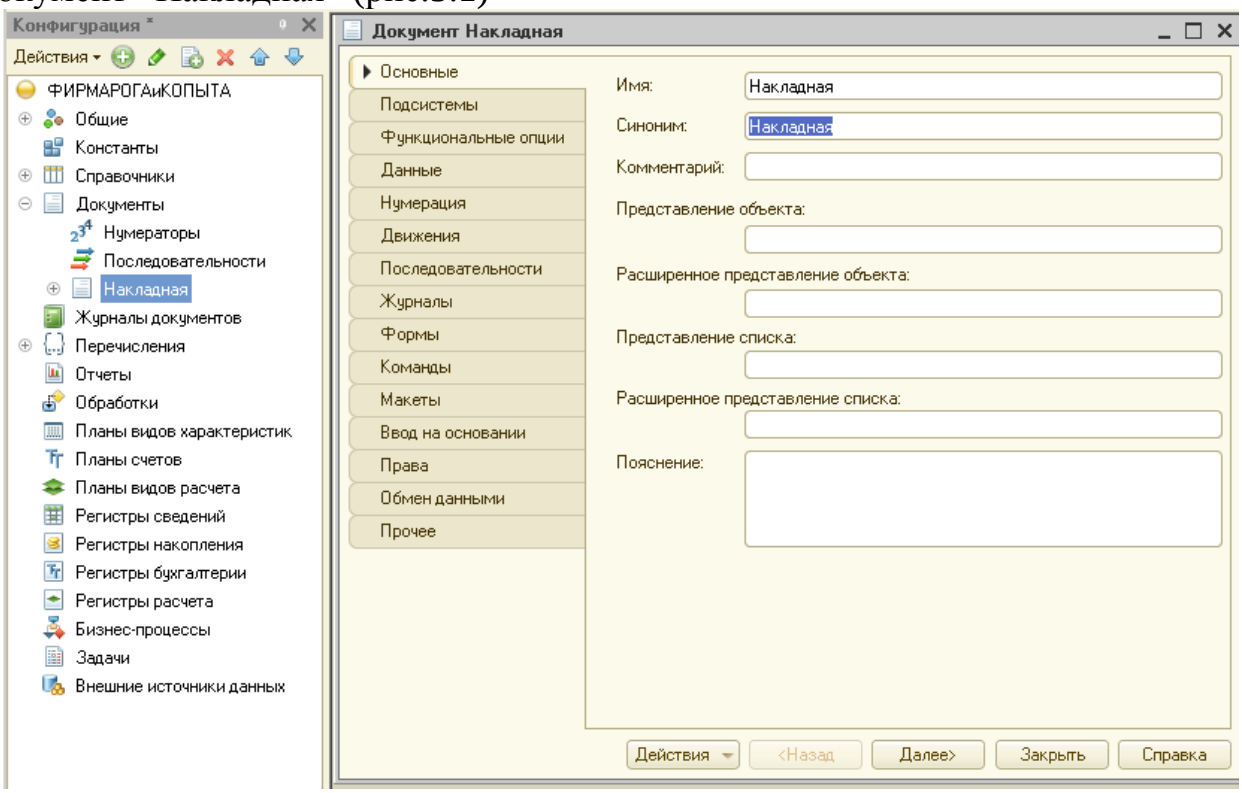


Рисунок 5.1 Создание нового документа «Накладная»

Данный документ будет фиксировать поступление оборудования и материалов на предприятие и соответственно фиксировать убывание денег за них уплаченных. Отнесем это документ к двум нашим подсистемам, так как с одной стороны, он будет фиксировать наши контакты с контрагентами, от которых мы будем получать необходимые нам инструменты и материалы. С другой стороны, документ будет финансовым, так как в нем будут отражаться суммы, уплаченные за эти материалы (рис.5.2).

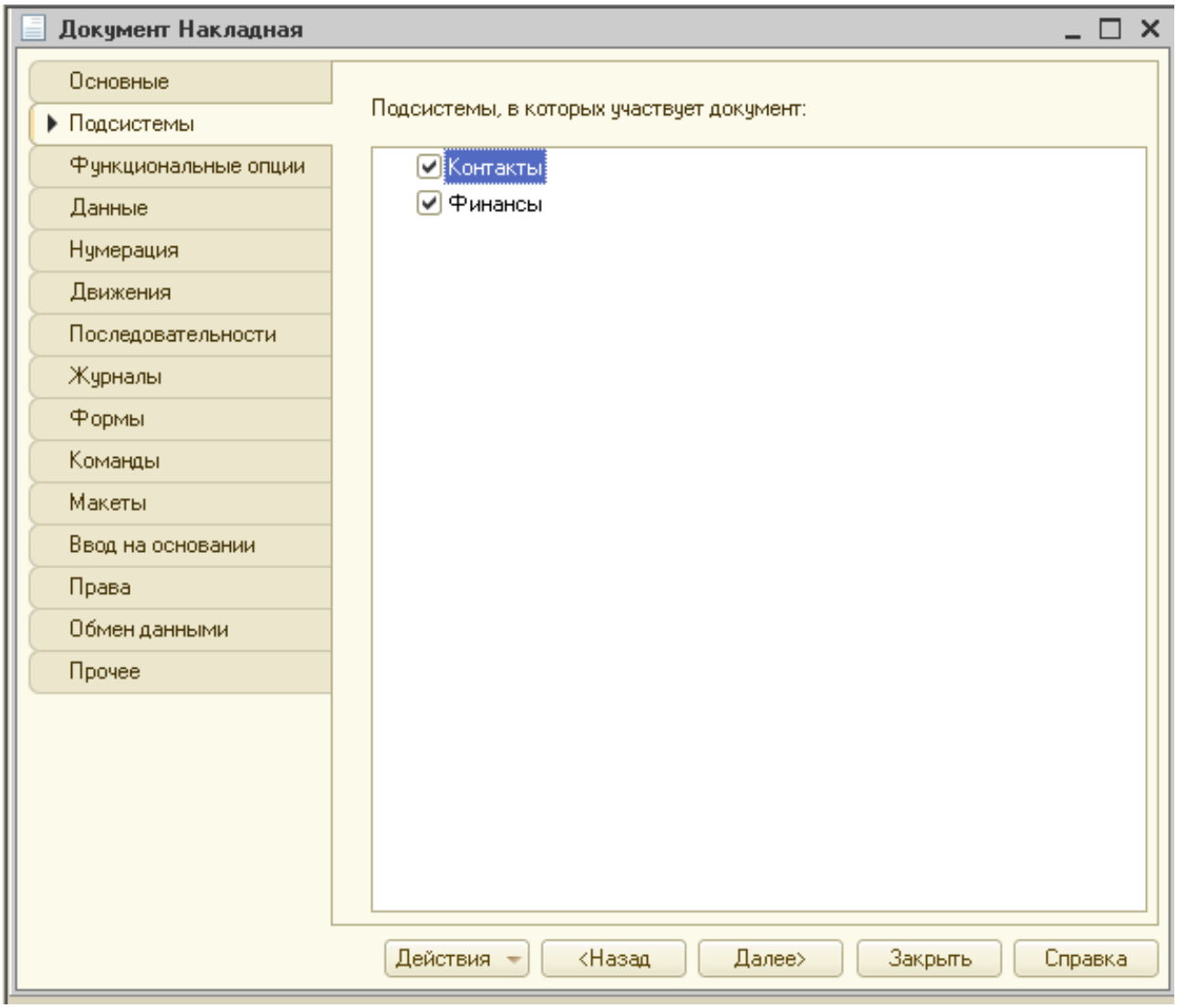


Рисунок 5.2. Участие документа в подсистемах.

Необходимым реквизитом накладной является «Контрагент». Это то предприятие или физическое лицо, которое поставляет нам инструмент или материал. Данный аргумент будет ссылочного типа, и ссылаться будет на справочник «Контрагенты» (рис.5.3).

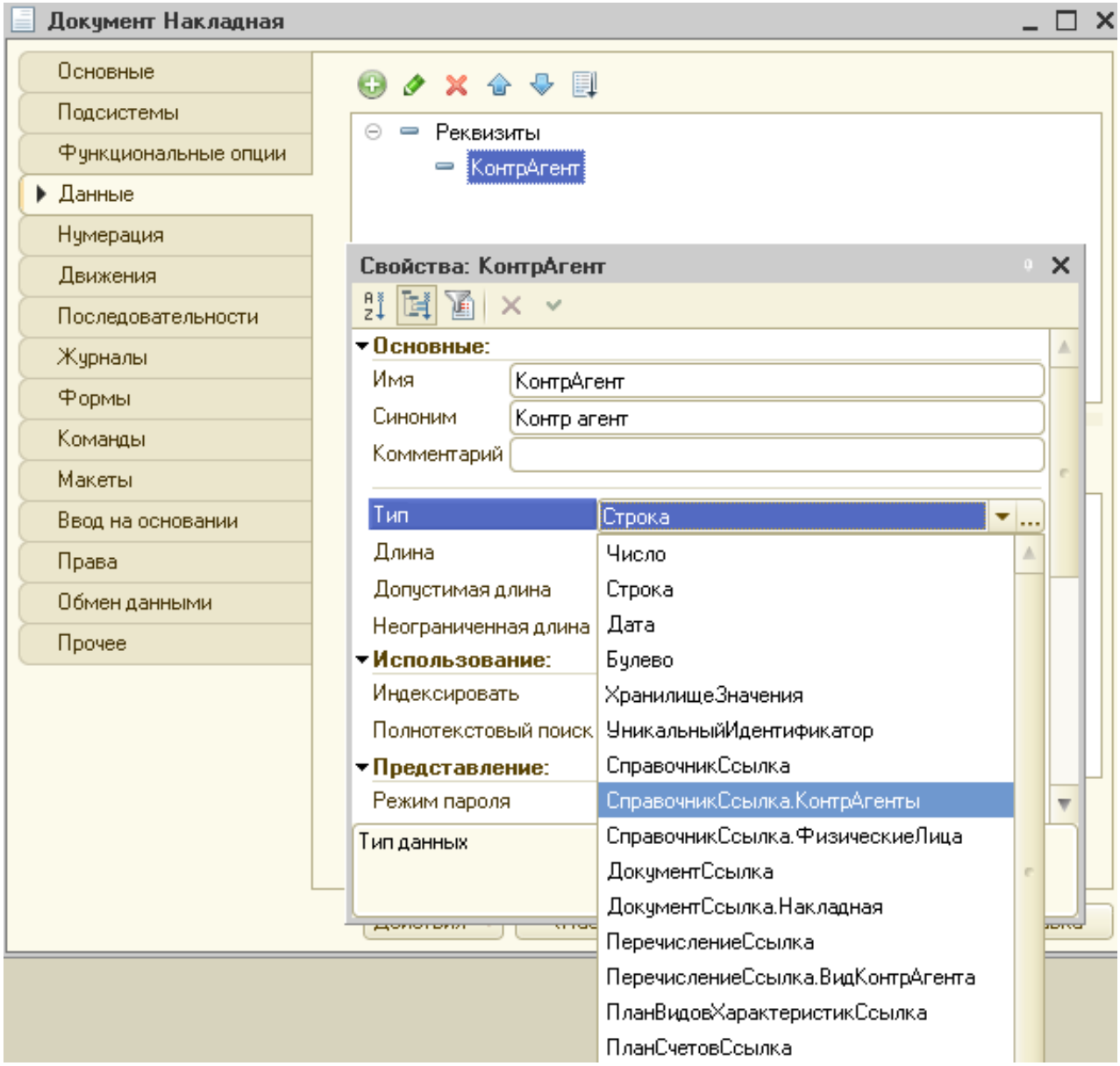


Рисунок 5.3. Реквизит ссылочного типа.

Теперь переходим к формированию табличной части. Кроме имени табличной части целесообразно установить свойство «Проверка заполнения» в режим «Выдавать ошибку», ибо документ без табличной части в данном случае не имеет смысла и, следовательно, не должен проводиться. Установим это свойство и удостоверимся в режиме 1С: Предприятие, что, действительно, система документ не пропустит (рис.5.4).

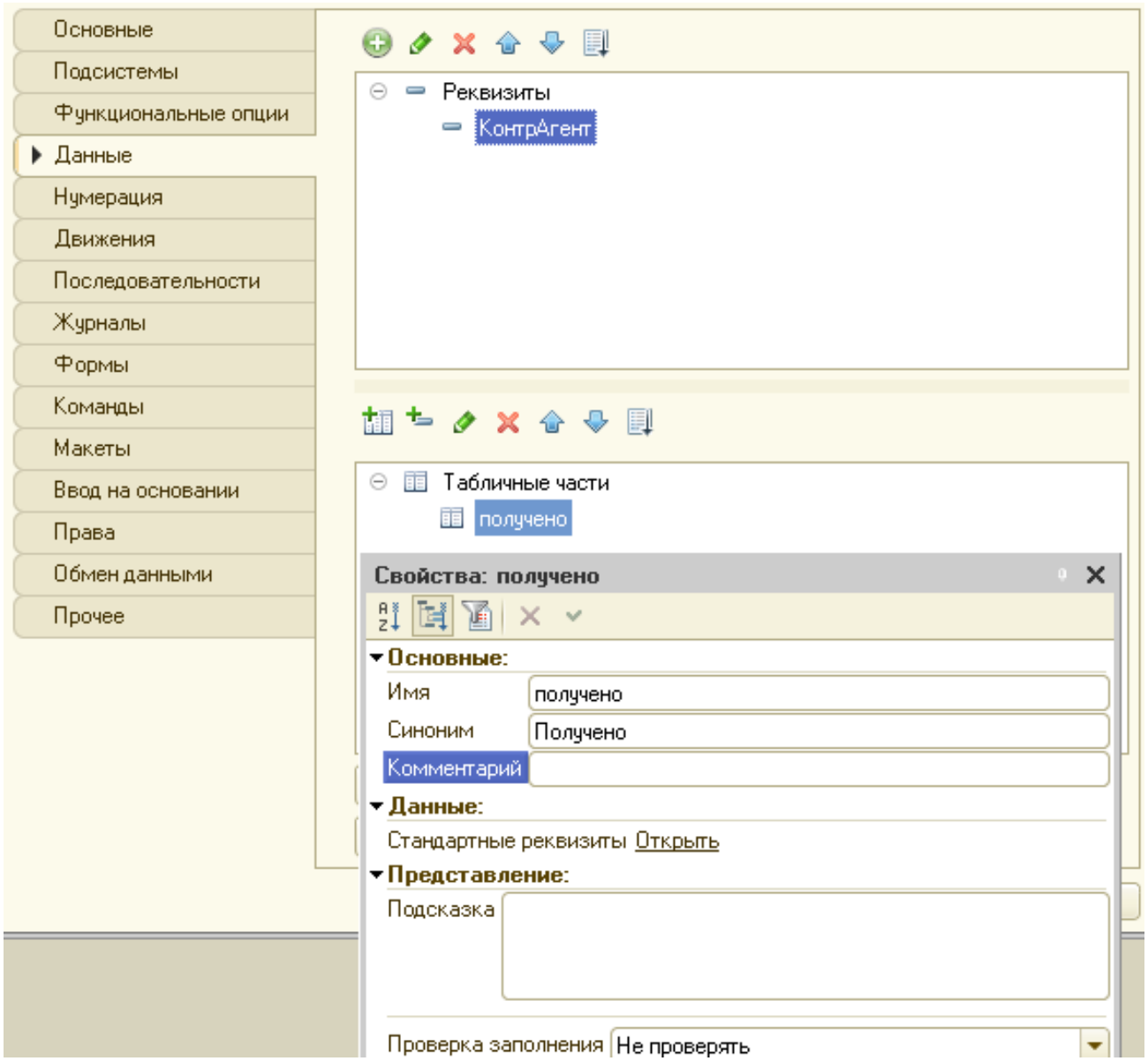


Рисунок 5.4.Создание табличной части.

Создаем минимально необходимый набор реквизитов табличной части: «Материал», «Количество», «Цена», «Сумма».

«Материал» - реквизит ссылочного типа, из справочника «Номенклатура».

«Количество», «Цена», «Сумма», реквизиты числовые.

Указываем длину и точность - количество знаков после запятой. Для каждого реквизита установим свойство «Проверка заполнения» - «Выдавать ошибку» (рис.5.5-5.9).

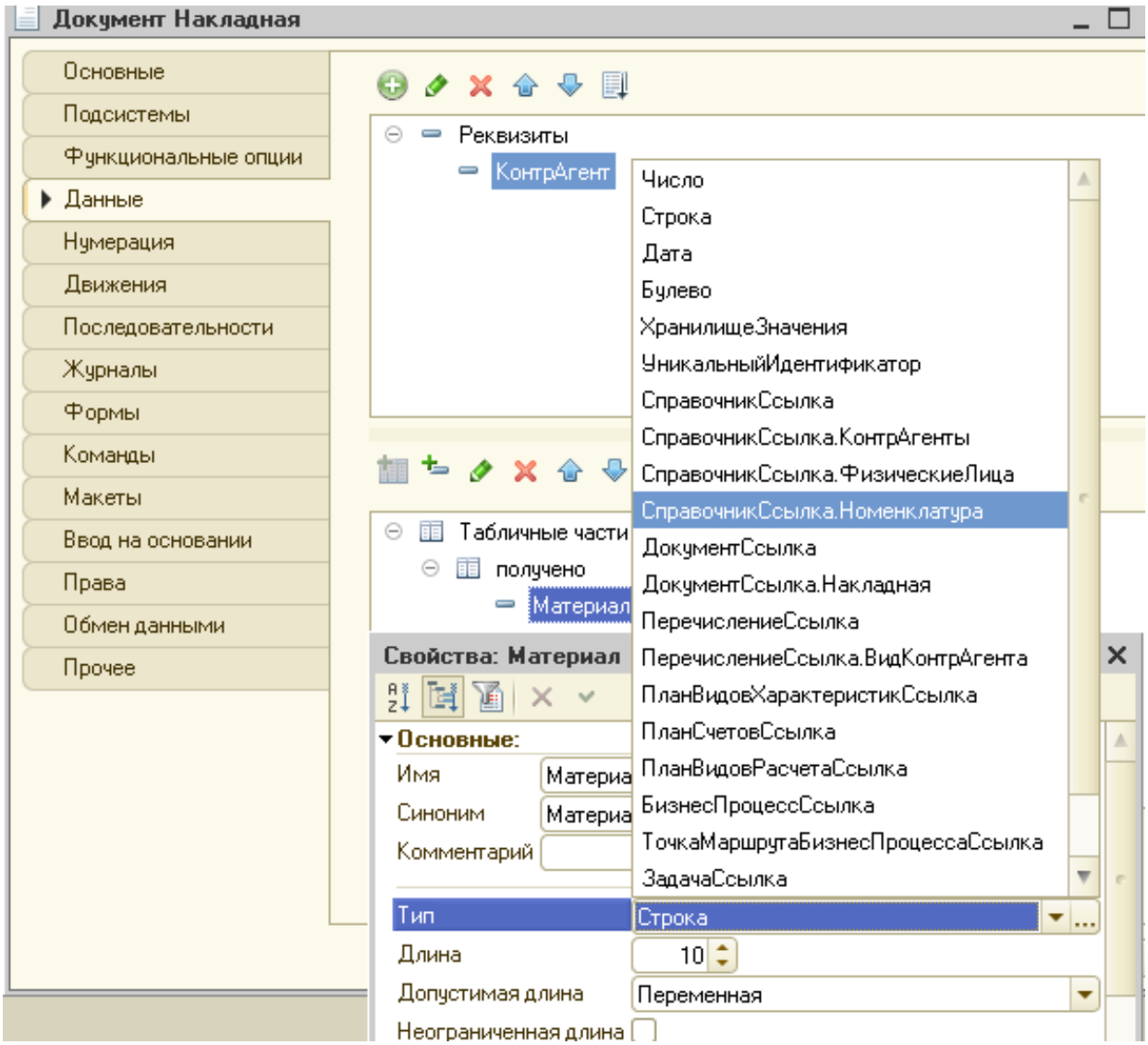


Рисунок 5.5. Реквизит табличной части «Материал».

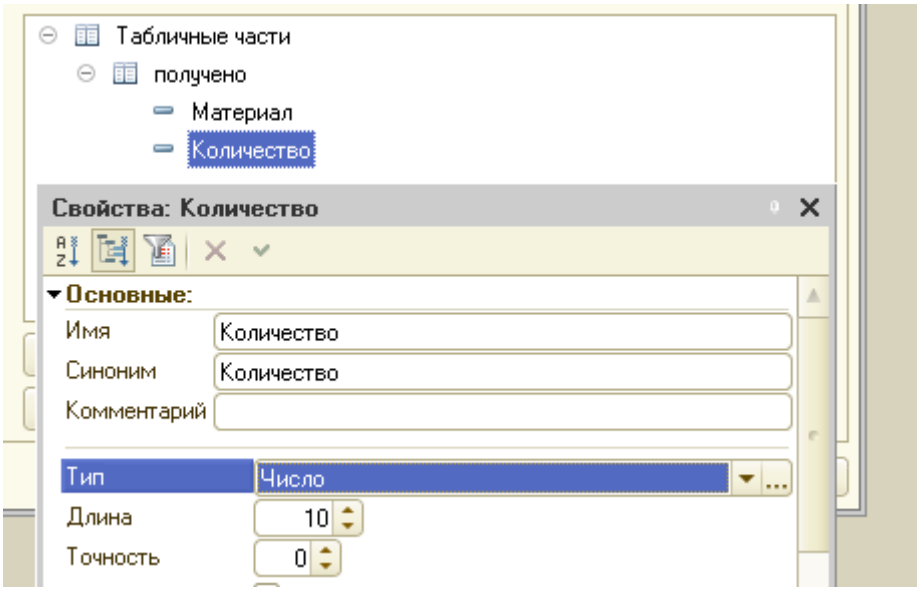


Рисунок 5.6. Реквизит табличной части «Количество»

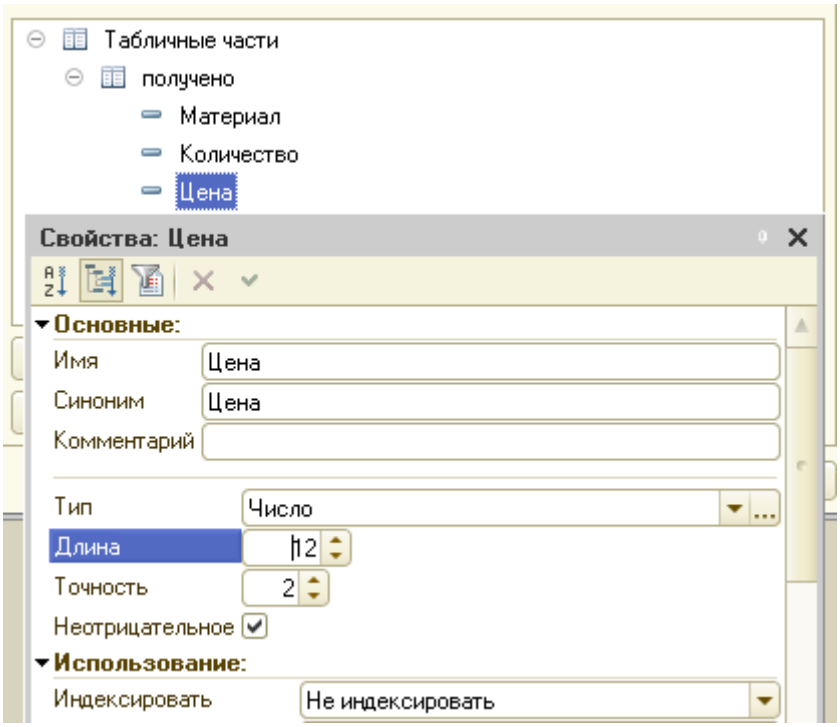


Рисунок 5.7. Реквизит табличной части «Цена»

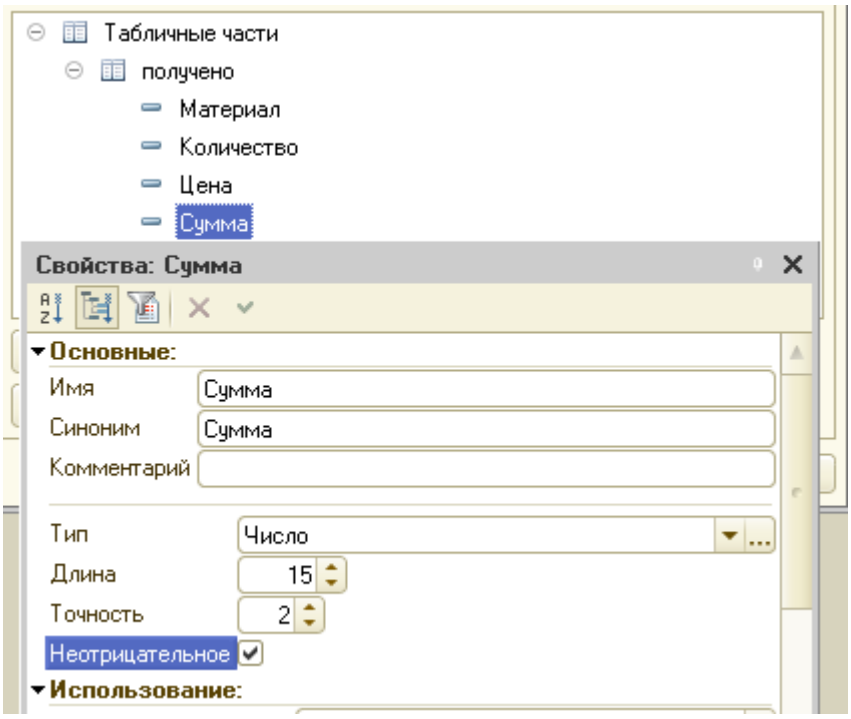


Рисунок 5.8. Реквизит табличной части «Сумма».

Дальше переходим на закладку Нумерация и устанавливаем свойства «Автонумерация», ставим галочку и определяем период, в пределах которого номер документа должен быть уникальным (рис.5.9). Это важное свойство. Во-первых, автонумерация исключает появление документов с одинаковыми номерами в пределах указанного периода, а во-вторых, пользователю не надо задумываться какой номер присвоить документу.

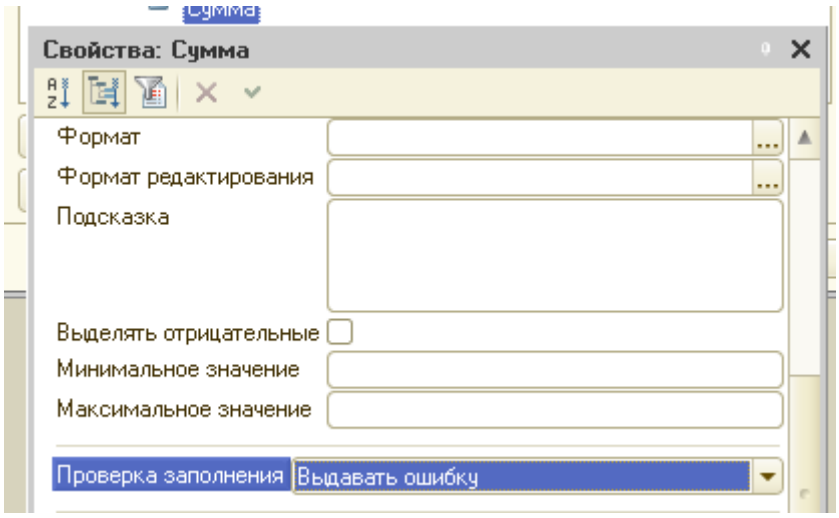


Рисунок 5.9. Свойство «Проверка заполнения».

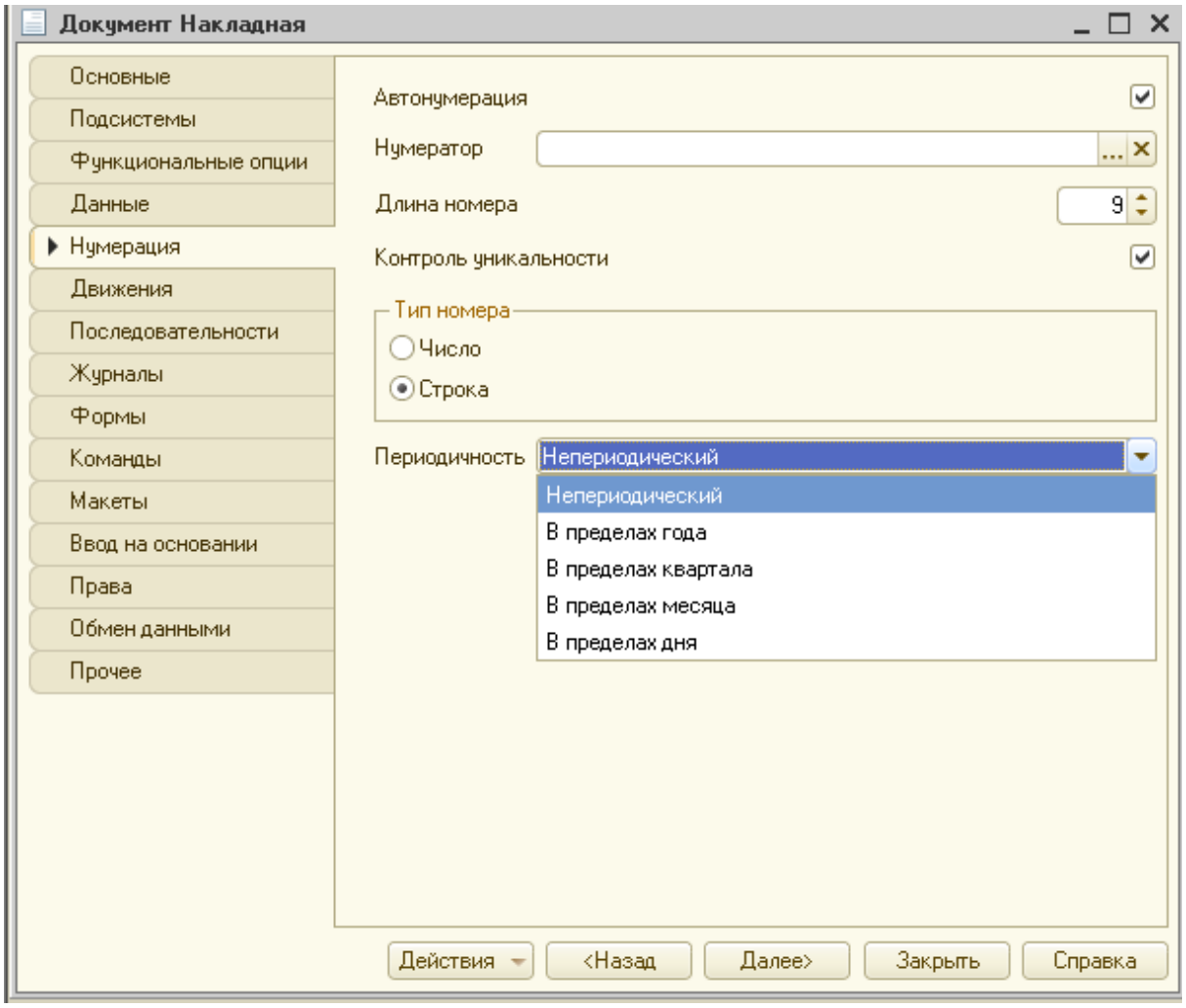


Рисунок 5.10. Выбор порядка нумерации документа.

В реальных системах учета на предприятии циркулирует большое количество видов различных документов. Для того чтобы облегчить работу пользователей по поиску и анализу документооборота, документы разных типов могут учитываться в журналах. В нашей учебной конфигурации это конечно не имеет практического значения. Однако научиться создавать журналы и

регистрировать в них документы надо. Создаем журнал «ЖурналНакладная» (рис.5.11-5.12).

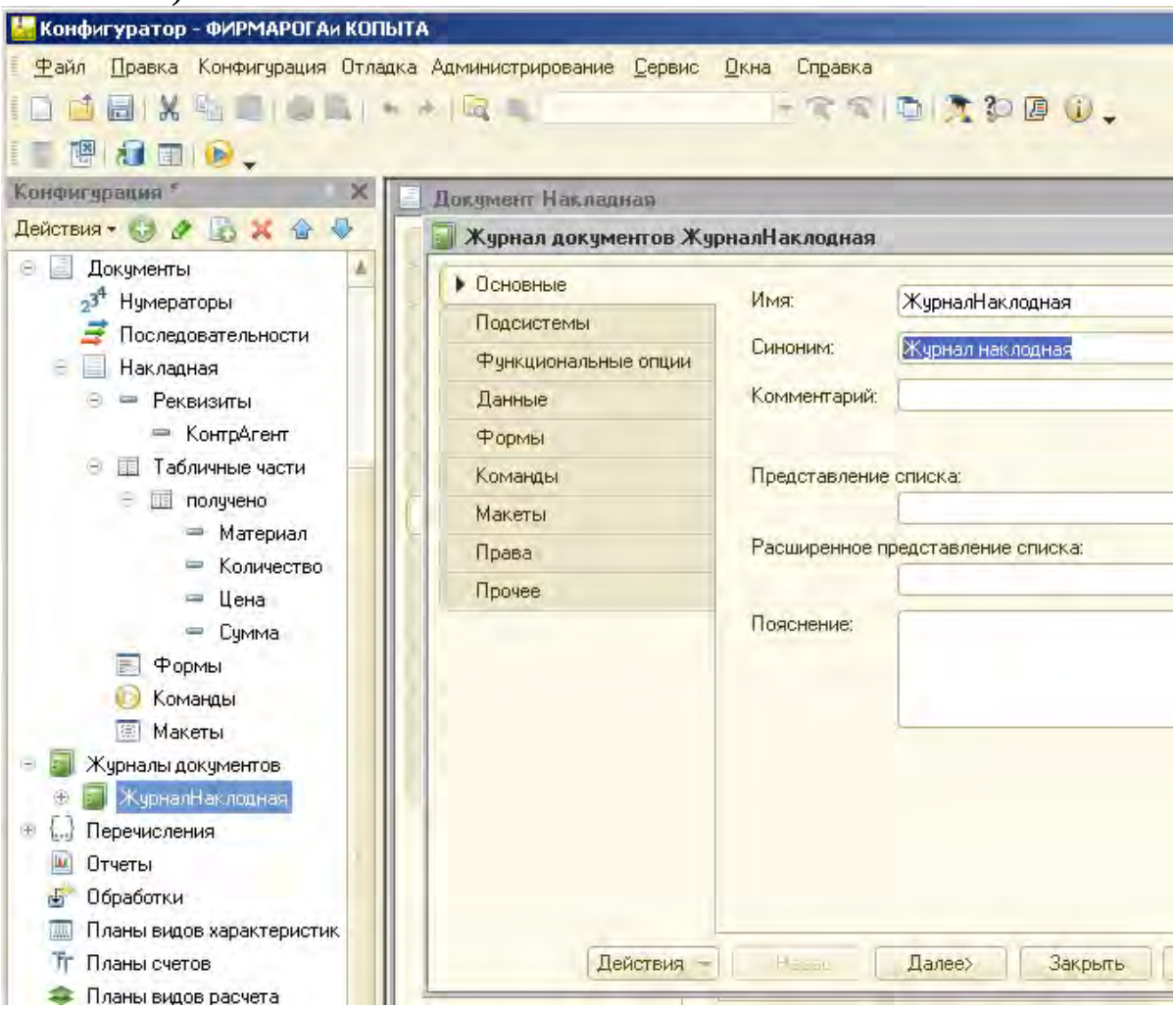


Рисунок 5.11. Создание журнала

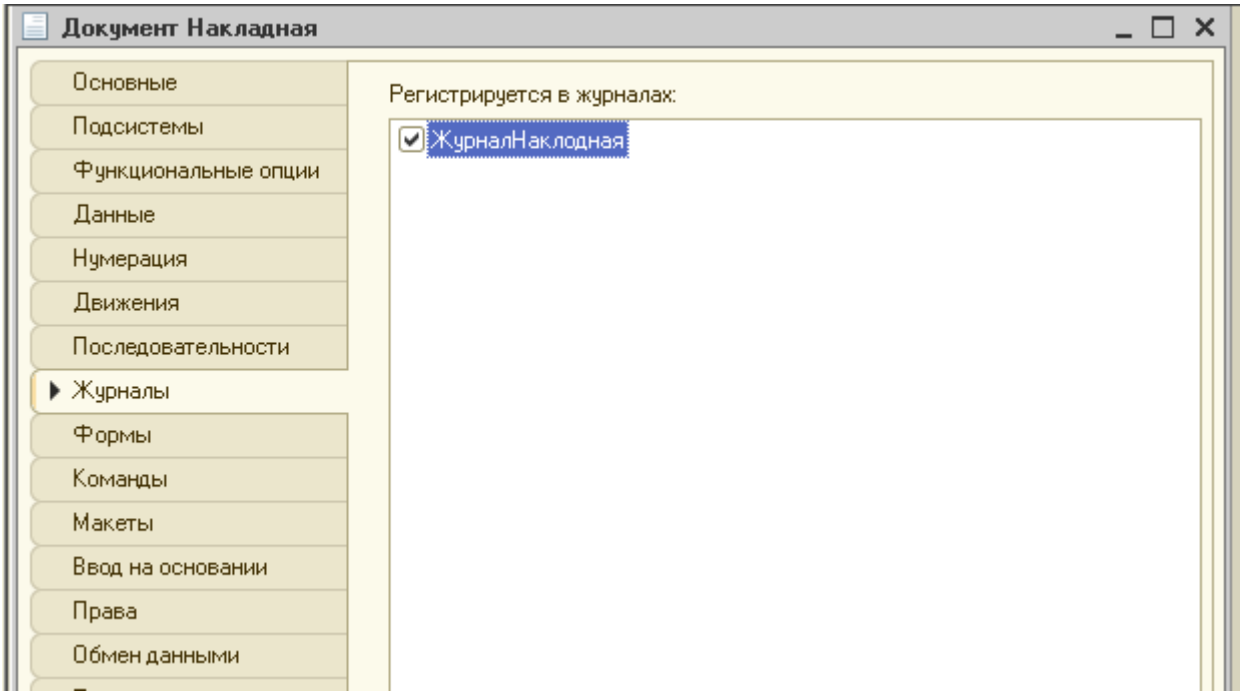


Рисунок 5.12. Регистрация в журнале.

Переходим к созданию, заполнению и регистрации накладной (рис.5.13-5.16).

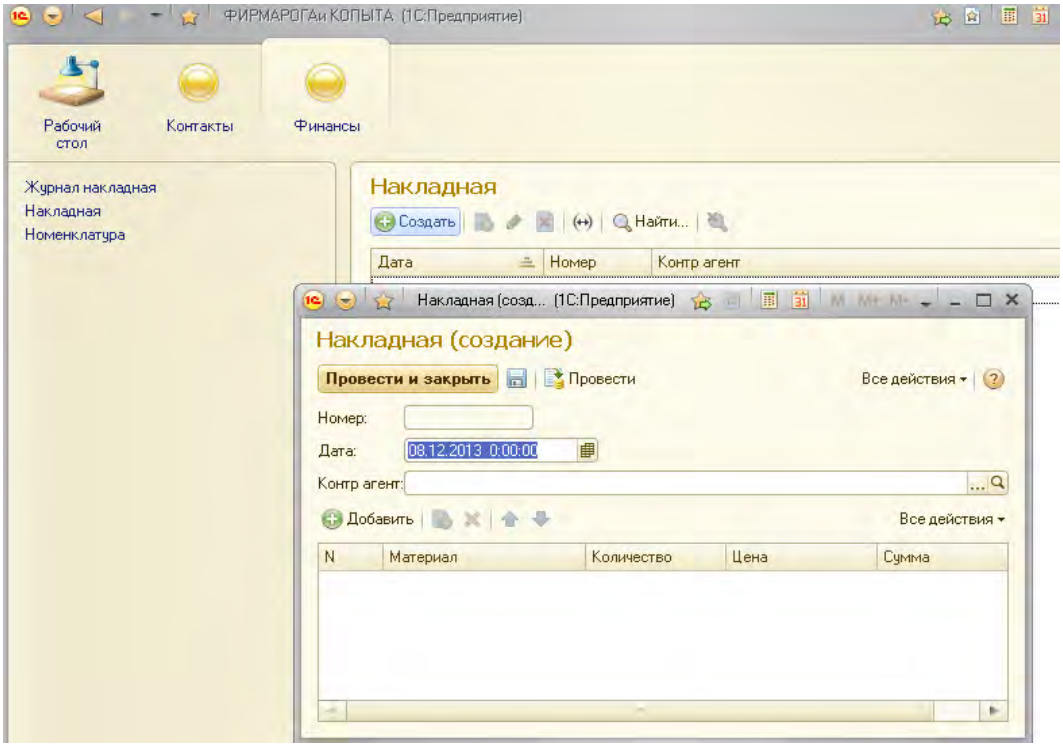


Рисунок 5.13. Начало заполнения документа.

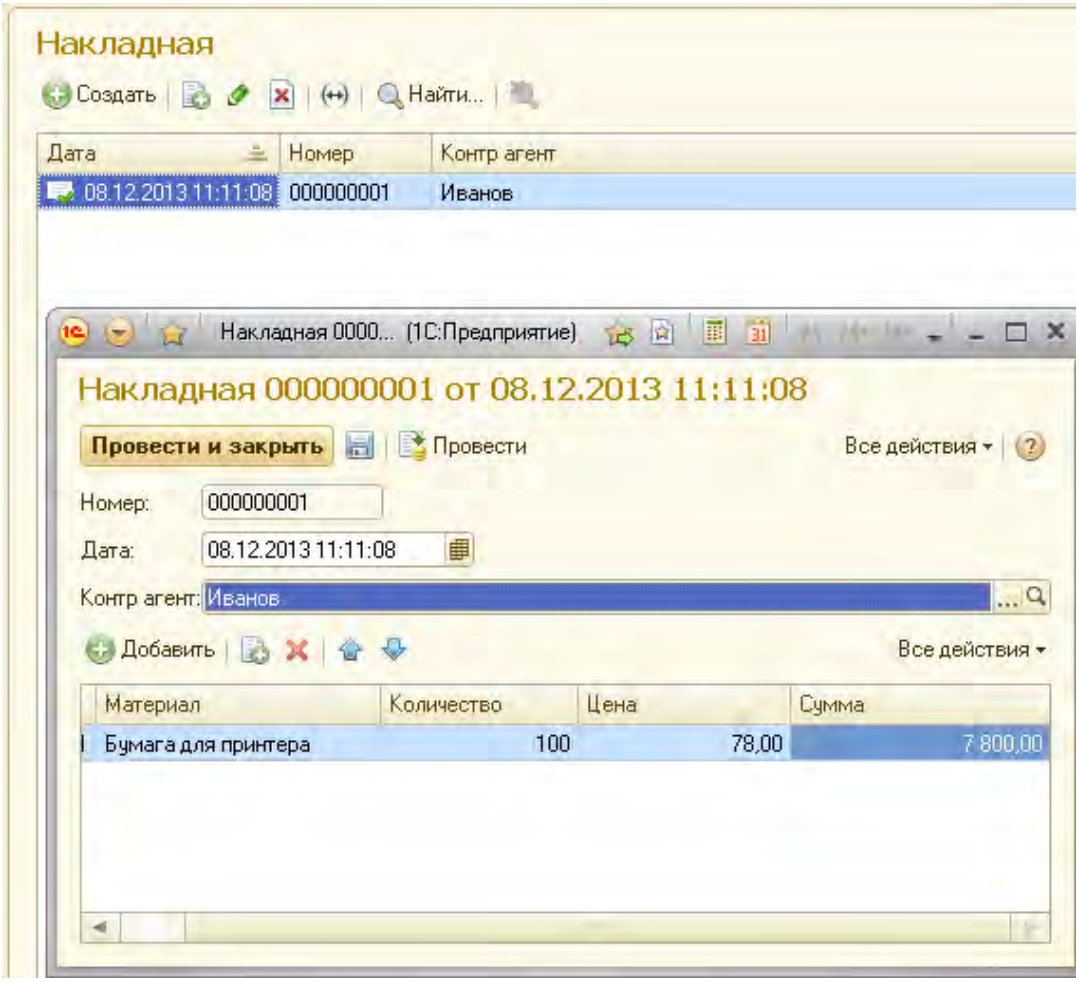


Рисунок 5.14. Создание и проведение накладной.

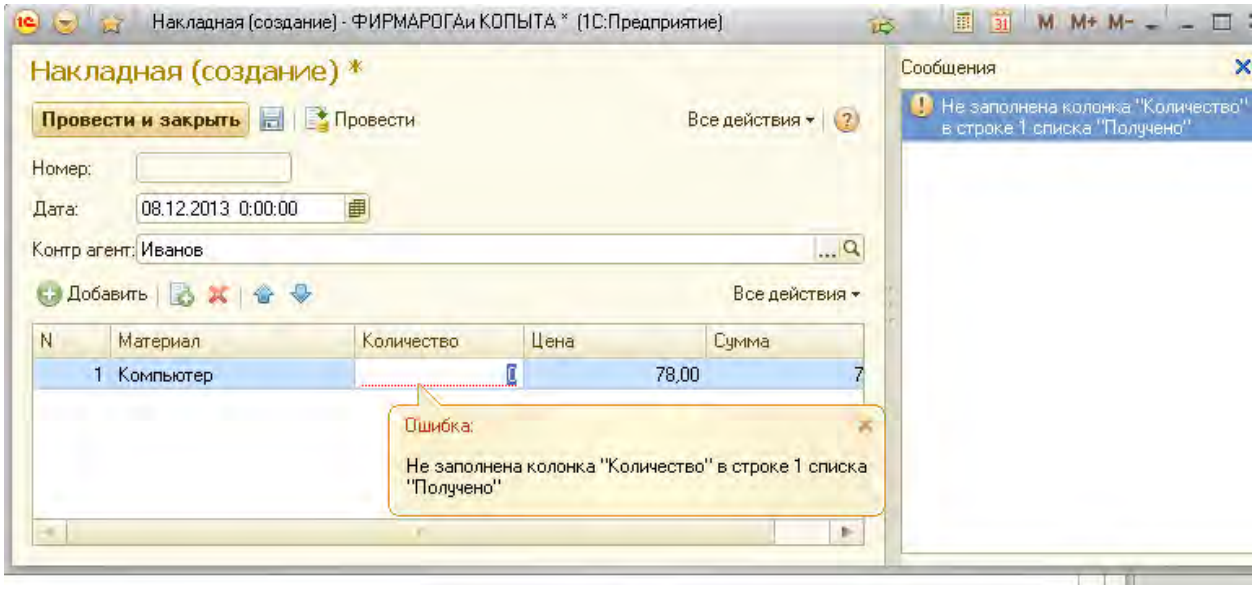


Рисунок 5.15. Сообщение об ошибке.

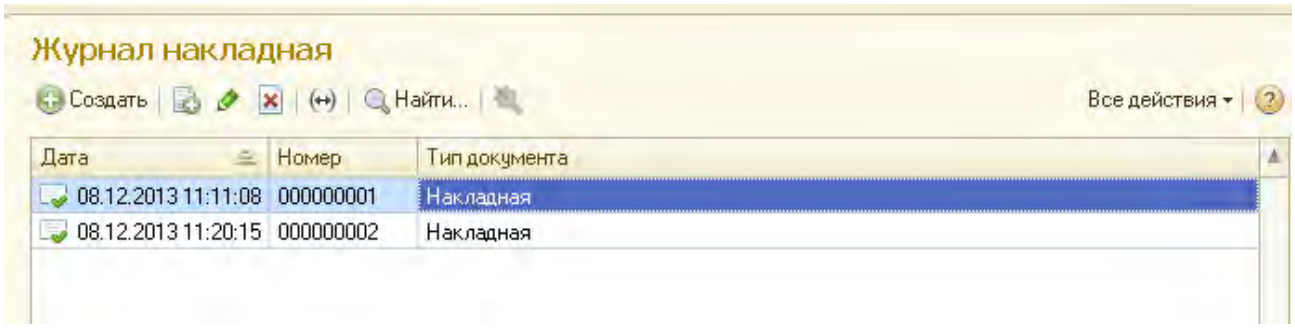


Рисунок 5.16. Журнал накладных

Немного поработав и создав документы, можно отметить их существенный недостаток - сумму приходится считать самому. Для устранения указанного недостатка вместо стандартной формы, которую мы использовали до сих пор, создадим собственную форму (рис.5.17). Пример создания формы для элемента справочника был рассмотрен в предыдущем разделе.

Используем конструктор форм документов и редактор форм. Визуально тут все просто, но нам надо достичь автоматического пересчета суммы при изменении в графах «Количество» и «Цена» (рис.5.18) [1].

В системе существуют «события», связанные с самыми различными моментами ее функционирования. Эти «события» связаны также с элементами формы. Используя встроенный язык, можно написать собственную программу обработки того или иного события. Надо сказать, этот механизм мало чем отличается от механизма событий в Windows.

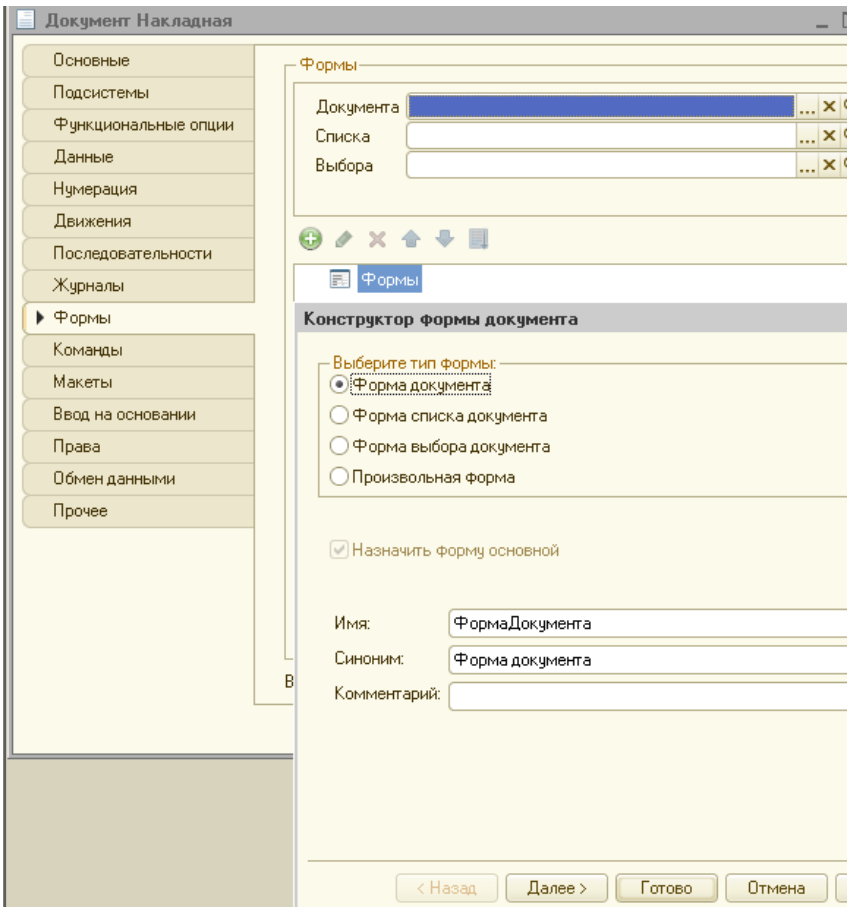


Рисунок 5.17. Создание формы документа

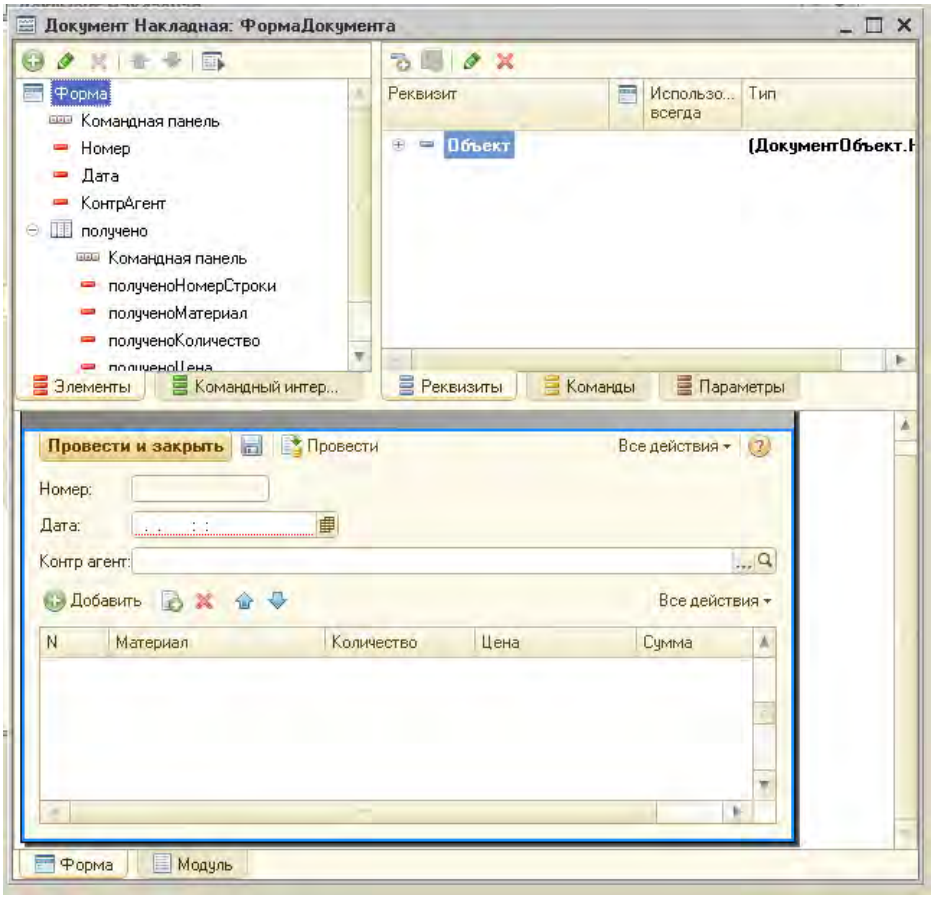


Рисунок 5.18. Форма документа в окне редактора форм.

Начнем с поля «Количество», в окне свойств этого поля найдем перечень событий, которые могут быть связаны с этим элементом (рис.5.19). Существует много разных событий: «ПриИзменении», «НачалоВыбораИзСписка», «Очистка», «Регулирование» и т.д. Нам подойдет событие «ПриИзменении». Нажимаем на кнопку с лупой, и система создаст нам заготовку для написания обработчика события (рис.5.20-5.22).

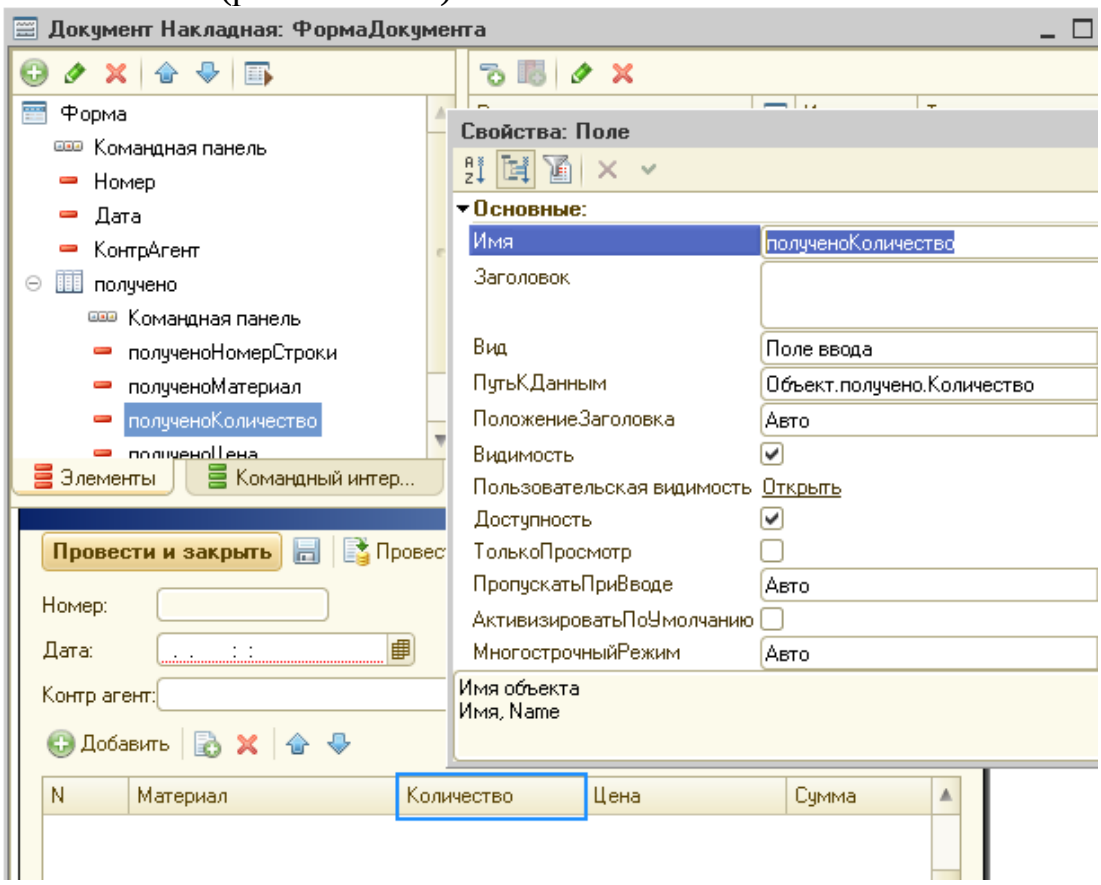


Рисунок 5.19. Окно свойств поля «Количество»

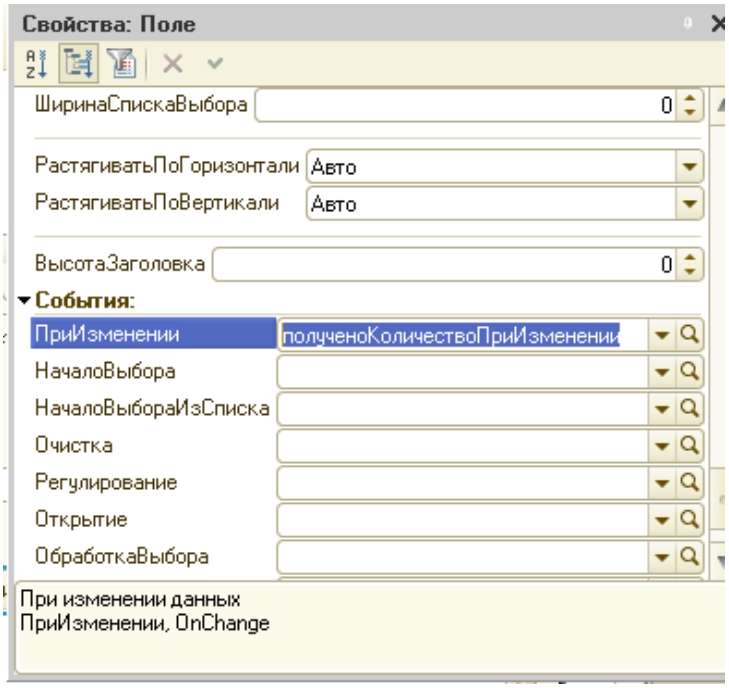


Рисунок 5.20. Создание обработчика события.

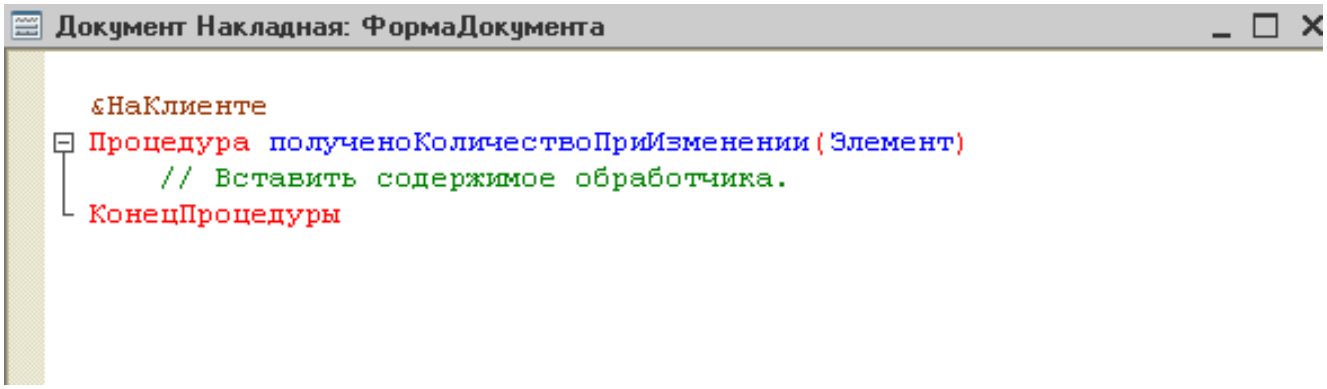


Рисунок 5.21. Модуль документа, открытый для редактирования.

Аналогично программированию на VC. Далее написать текст обработчика.

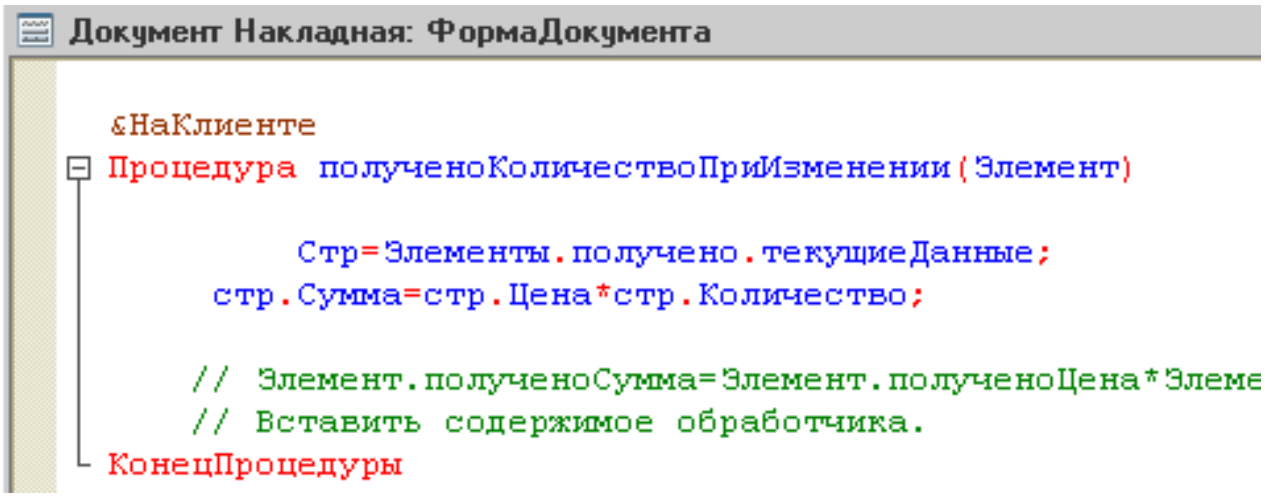


Рисунок 5.22.Текст обработчика.

Вот всего две строки.

В первой мы создаем переменную «Стр», и присваиваем ей значение объекта, содержащего данные строки табличного поля, причем указываем, что надо брать данные текущей строки (свойство «ТекущиеДанные»).

Каждый элемент формы можно получить в качестве свойства, указав через точку его имя. Во второй строке просто перемножаем полученные значения и присваиваем результат реквизиту «Сумма».

После того как написали код на встроенном языке, желательно проверить его синтаксическую правильность. Появится область с ошибками, если они есть или как в нашем примере с надписью: «синтаксических ошибок не обнаружено» (рис.5.23). Аналогия с программированием в VC++ и тут просматривается.

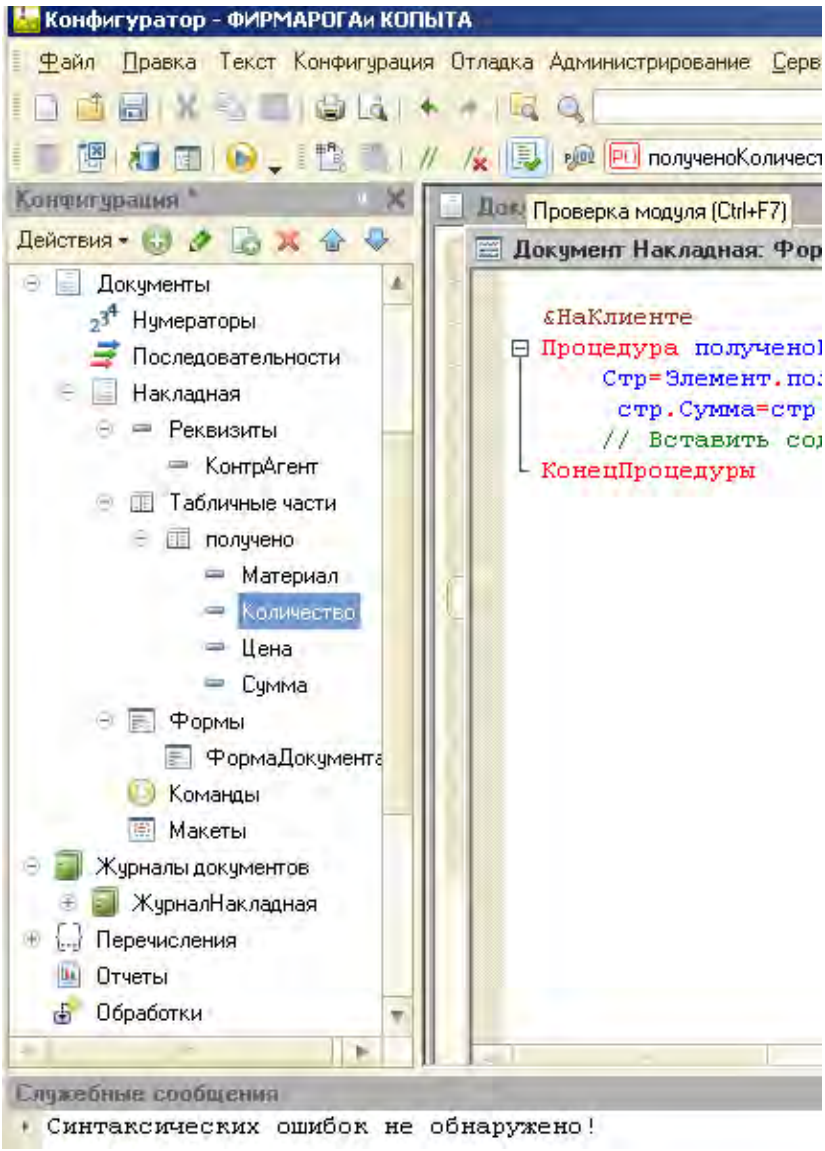


Рисунок 5.23. Проверка на синтаксические ошибки.

Делаем аналогичный обработчик для «Цены», но несколько другим способом, так как всегда можно сделать одно и то же действие разными способами (рис.5.24-5.26).

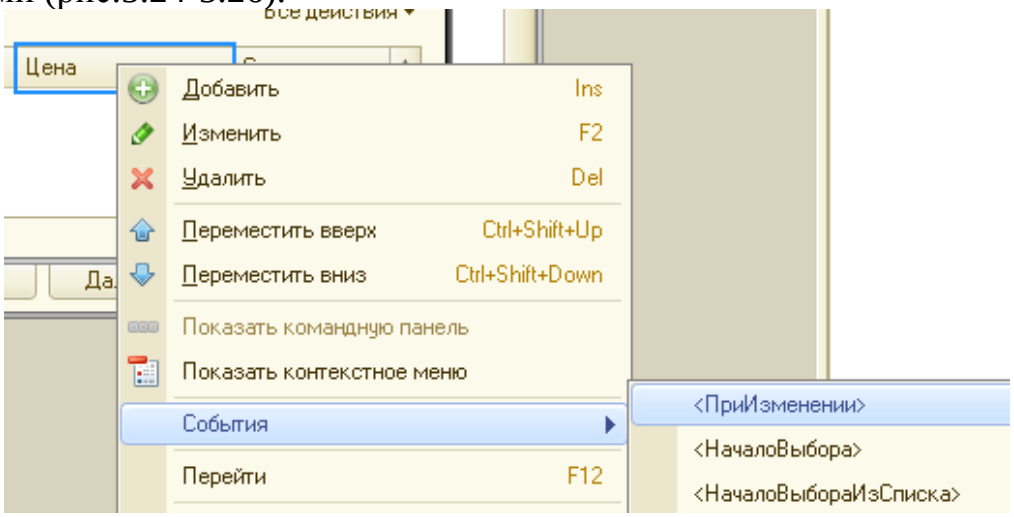


Рисунок 5.24. Обработчик для «Цены»

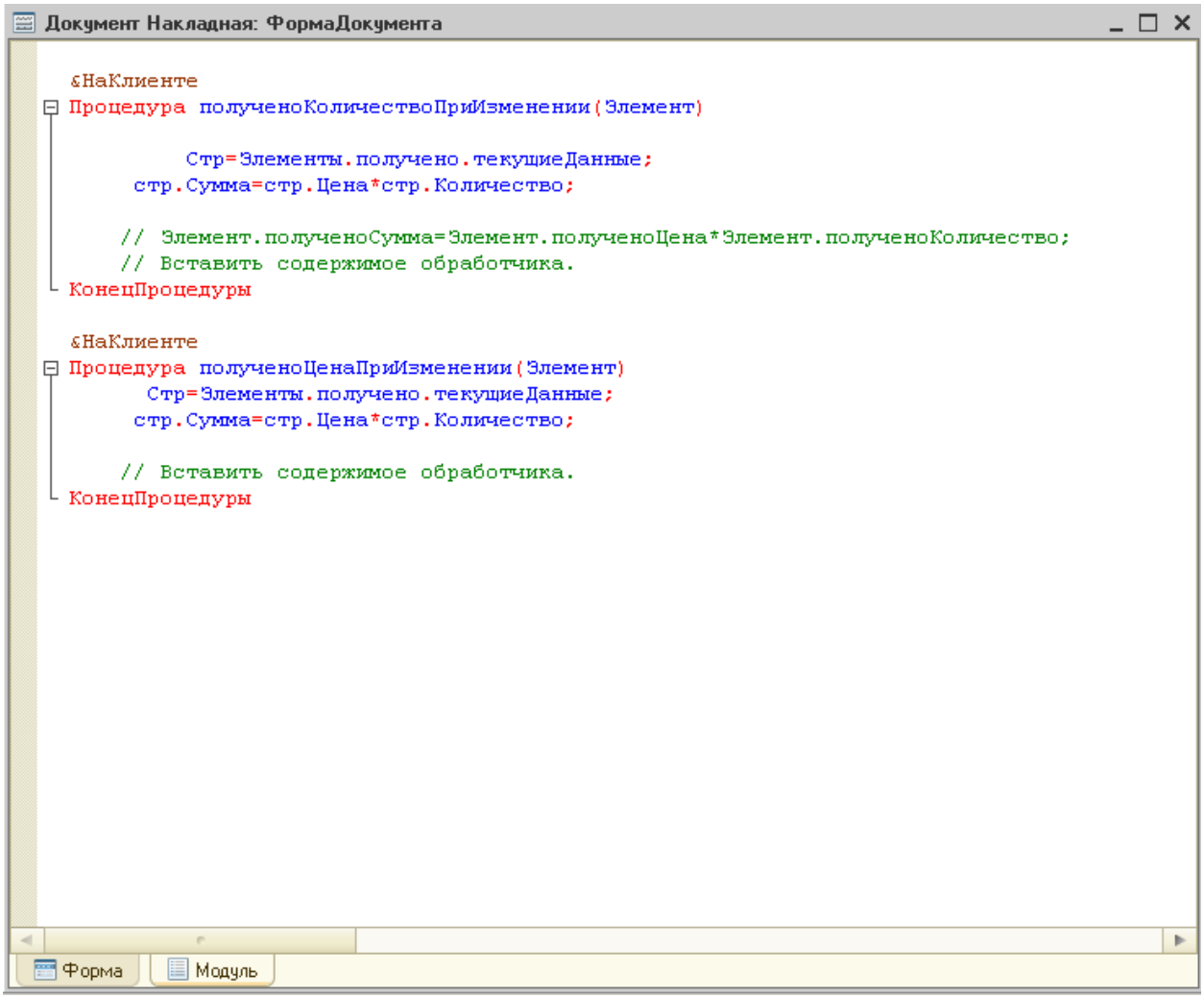


Рисунок 5.25. Модуль формы документа с двумя одинаковыми обработчиками.

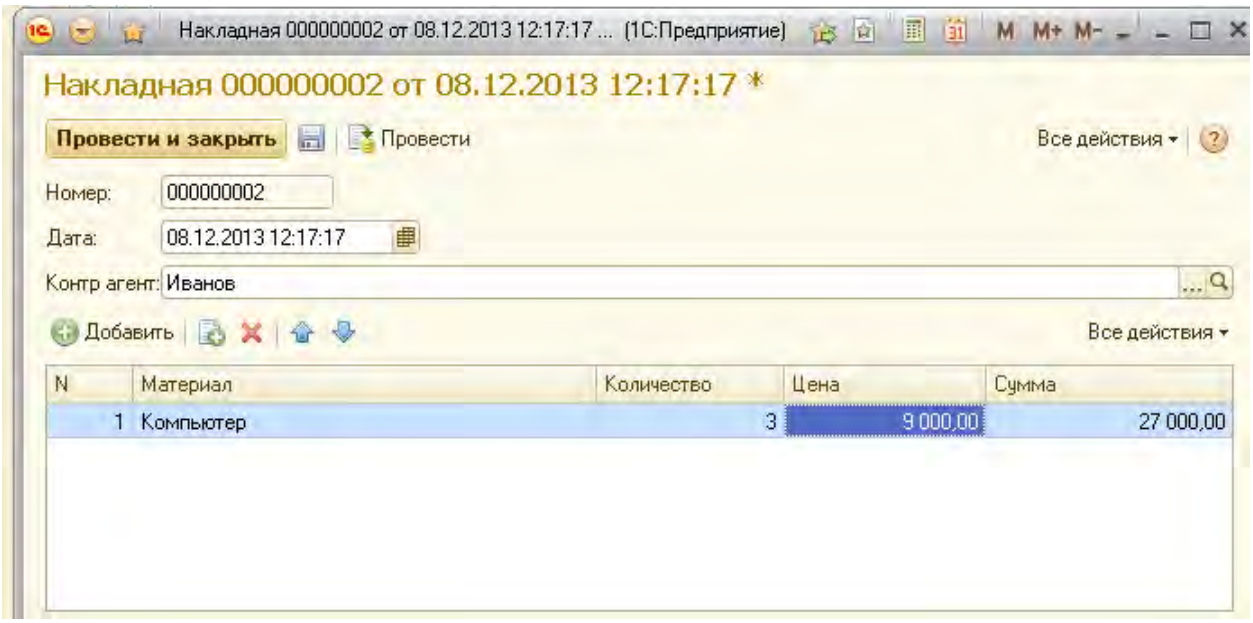


Рисунок 5.26.Пресчет суммы при изменении «Цены» или «Количества».

Создадим еще один документ, аналогичный документу накладная, который назовем «Счет». Этот документ будет регистрировать продажу нашей продукции и оказания услуг для сторонней организации. Поскольку документ аналогичен созданному, то можно получить его копированием (рис.5.27-5.31).

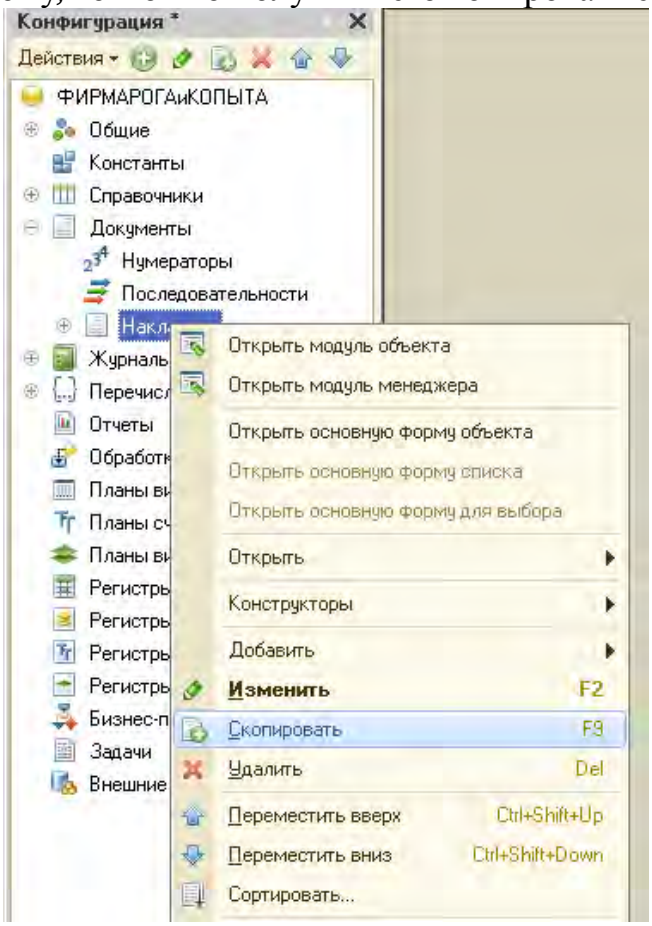


Рисунок 5.27. Копирование объекта конфигурации.

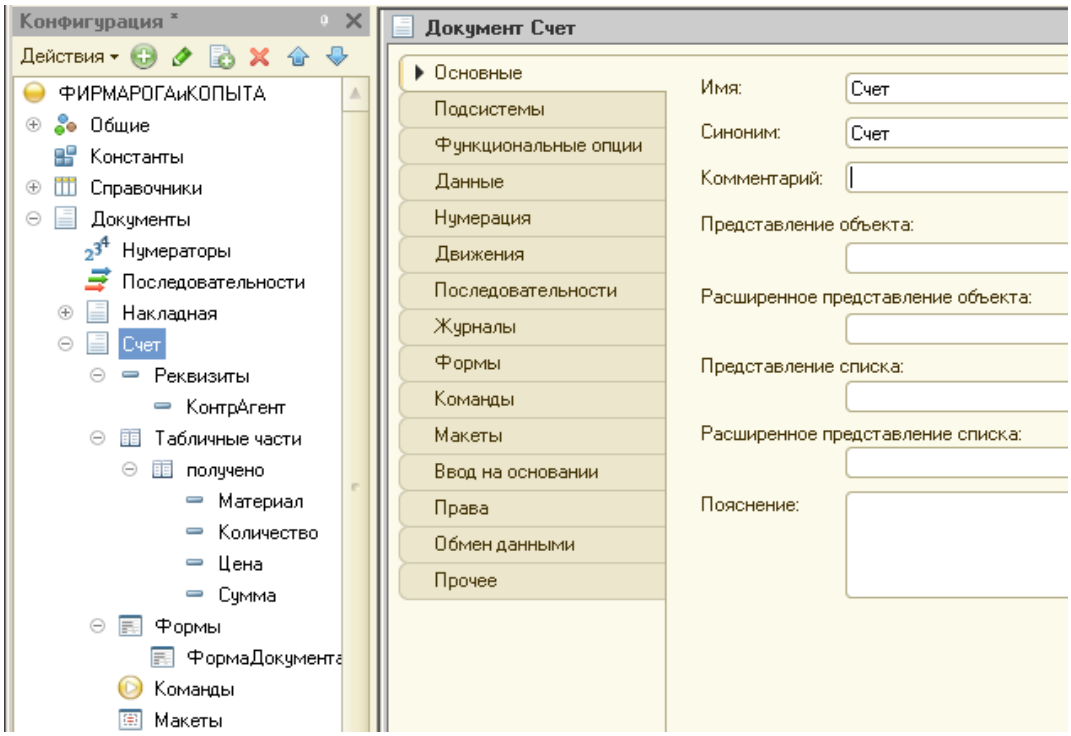


Рисунок 5.28. Новый документ «Счет».

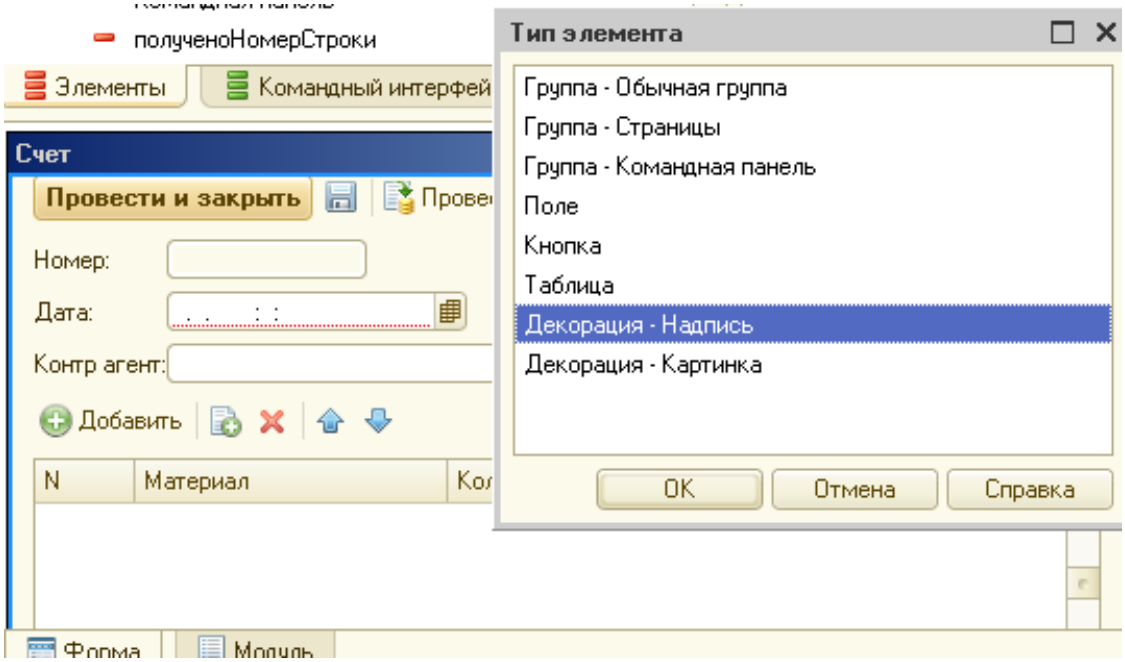


Рисунок 5.29. Добавление на форму надписи.

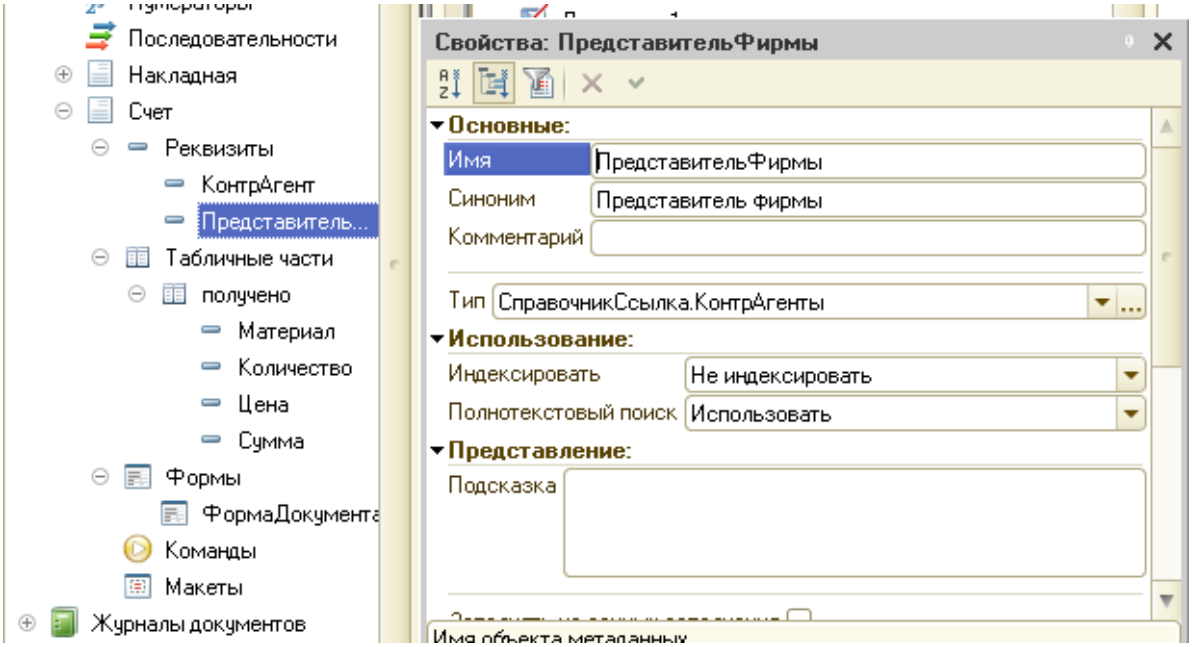


Рисунок 5.30. Добавление реквизита.

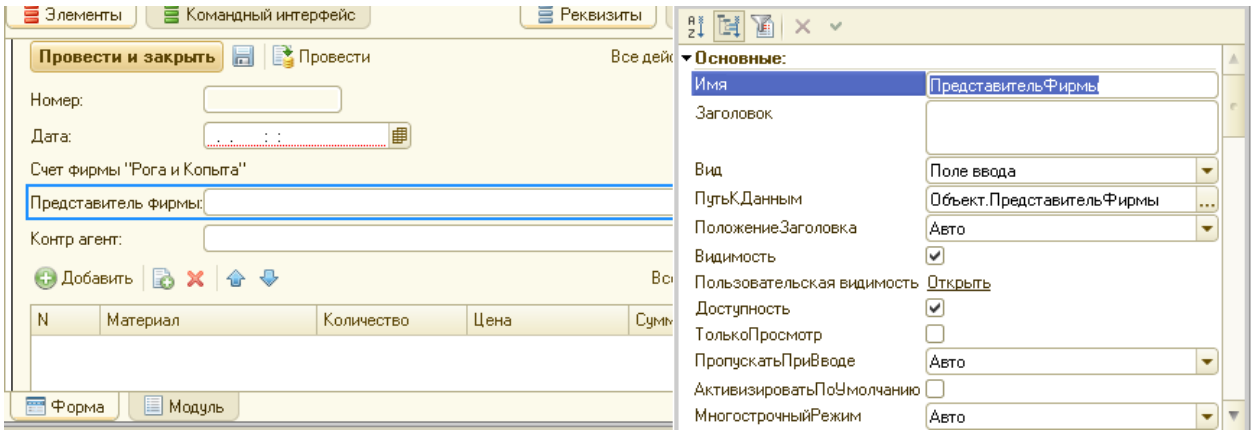


Рисунок 5.31 Добавление реквизита «Представитель Фирмы» на форму.

В результате получен документ (рис.5.32).

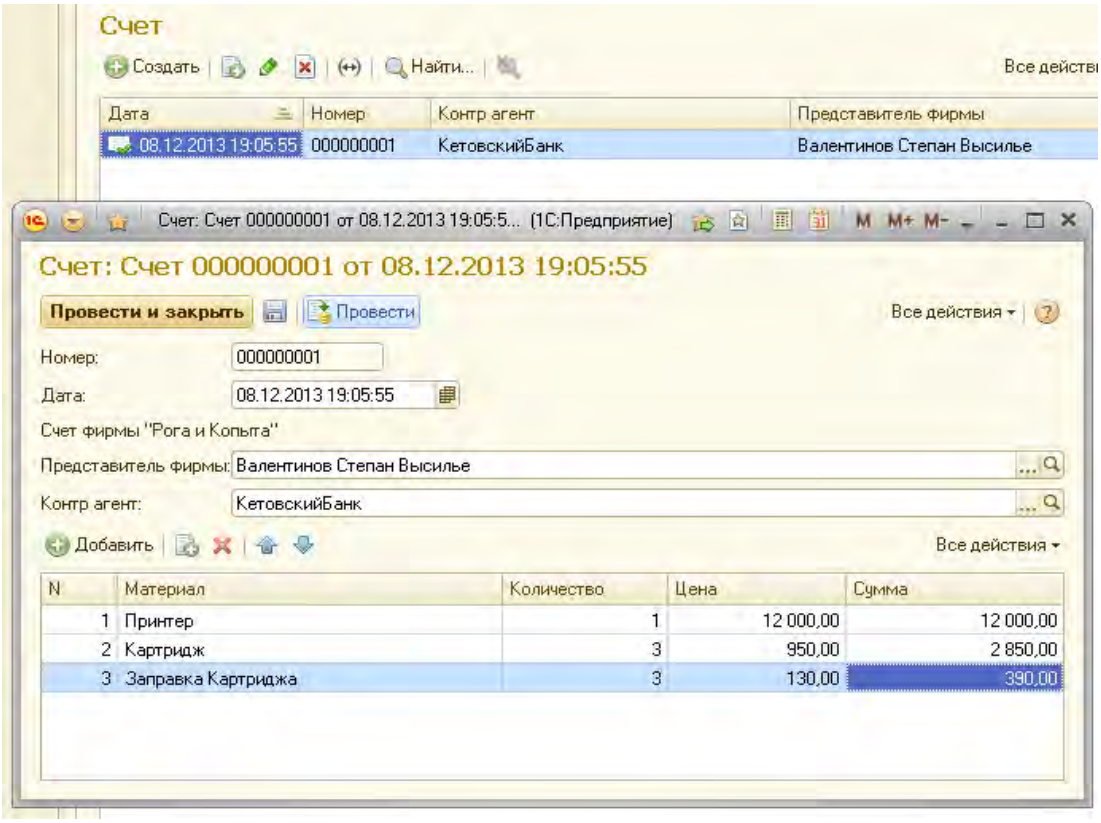


Рисунок 5.32. Документ типа «Счет» в 1С: Предприятии.

Добавляем итоговую сумму по документу (рис.5.33).

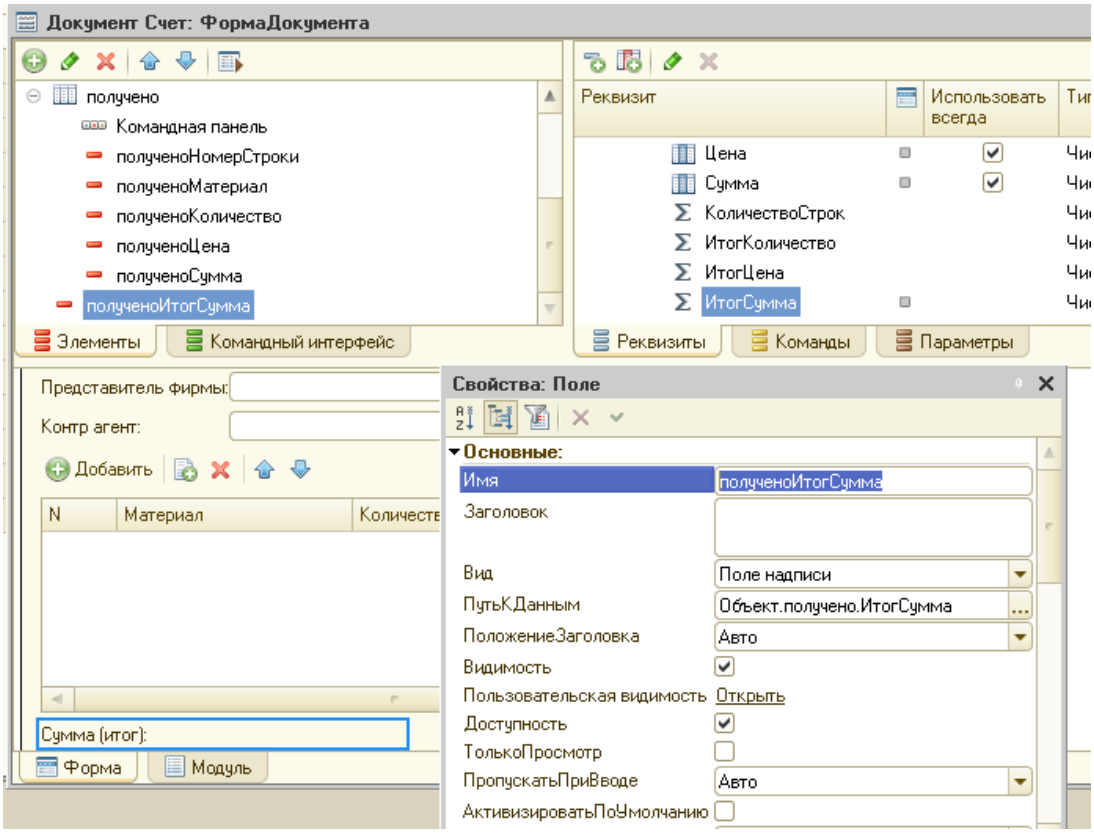


Рисунок 5.33. Добавление итоговой суммы

Получен счет с итоговой суммой (рис.5.34).

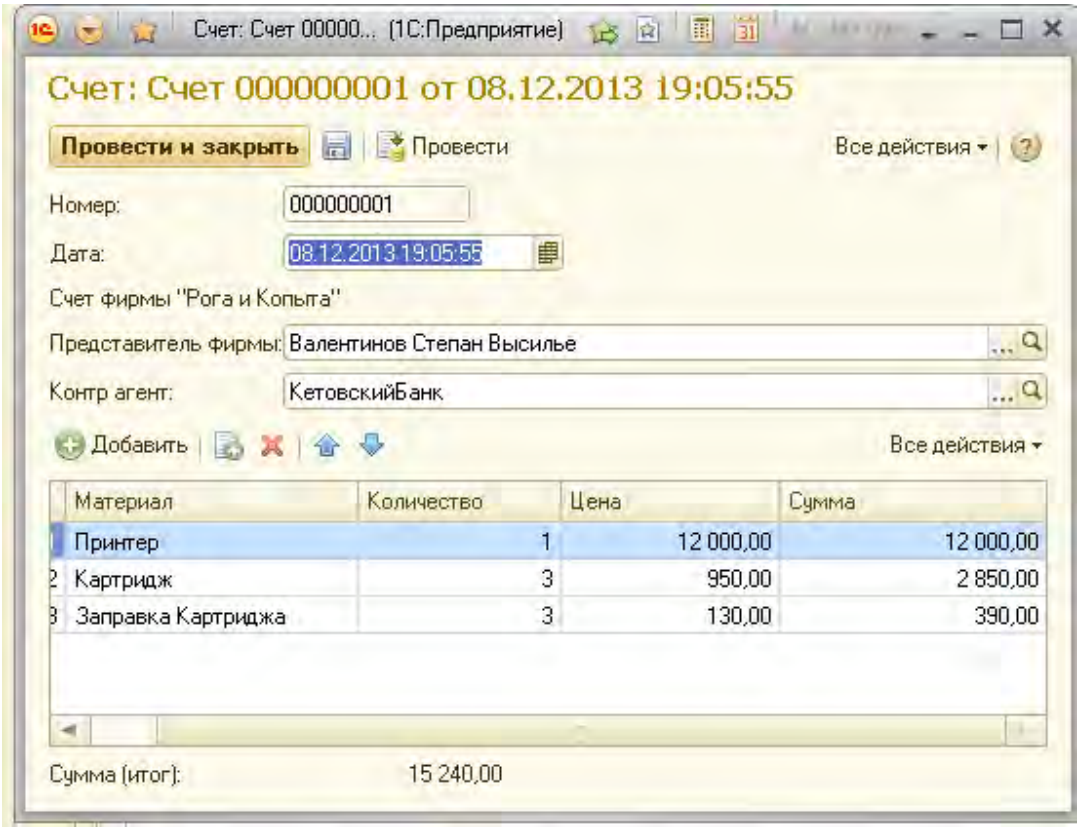


Рисунок 5.34. Счет с итоговой суммой.

Однако этот счет надо будет выставить для оплаты, в России традиционно любят бумажные документы, значит желательно иметь возможность получить печатную форму документа (рис.5.35-5.38).

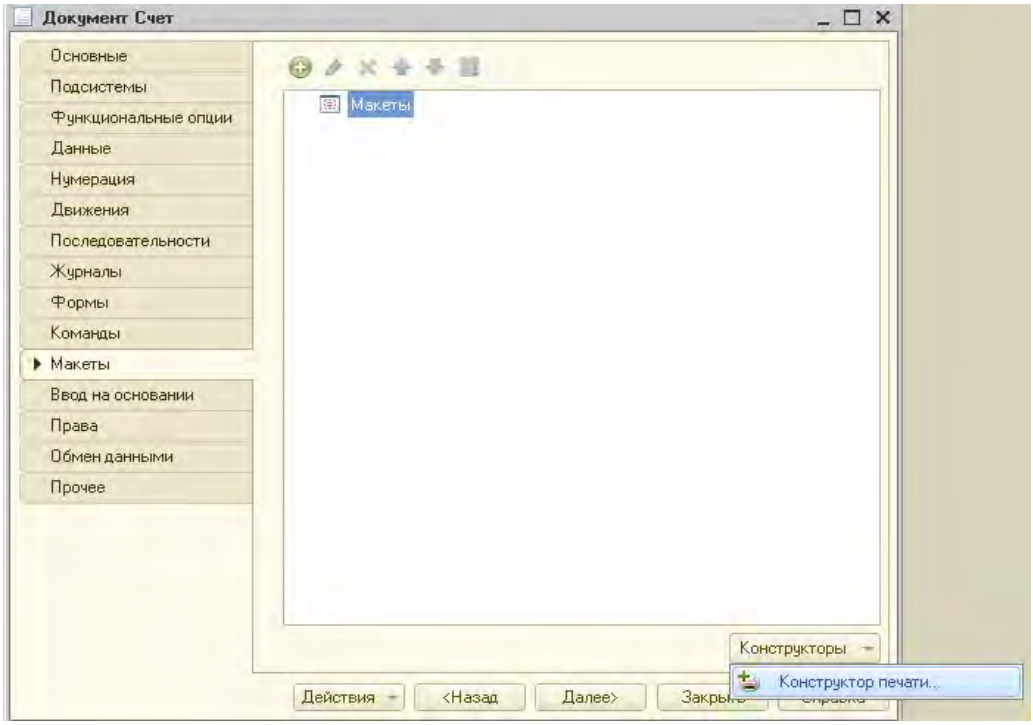


Рисунок 5.35.Запуск конструктора печатных форм

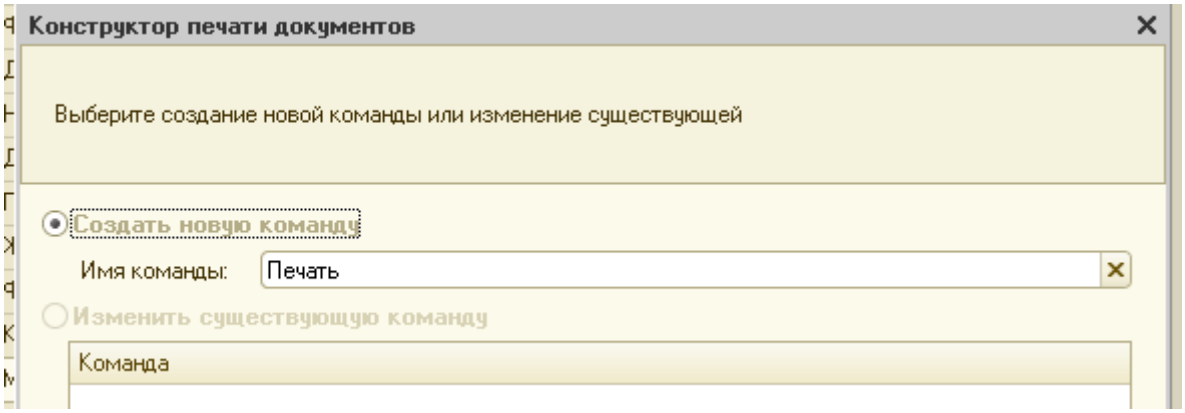


Рисунок 5.36. Создание команды печать.

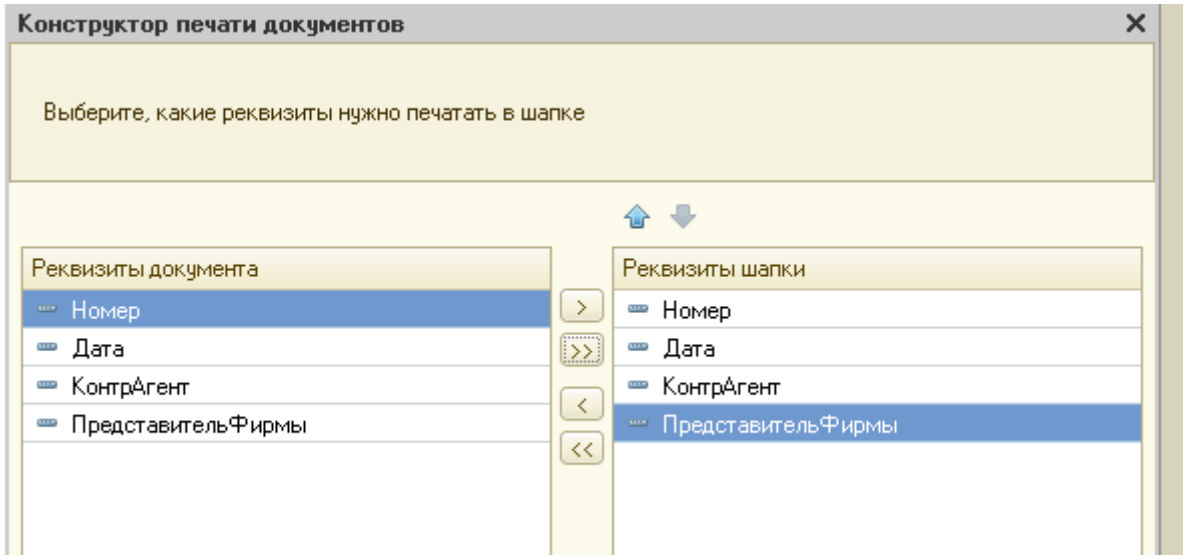


Рисунок 5.37.Выбор реквизитов шапки

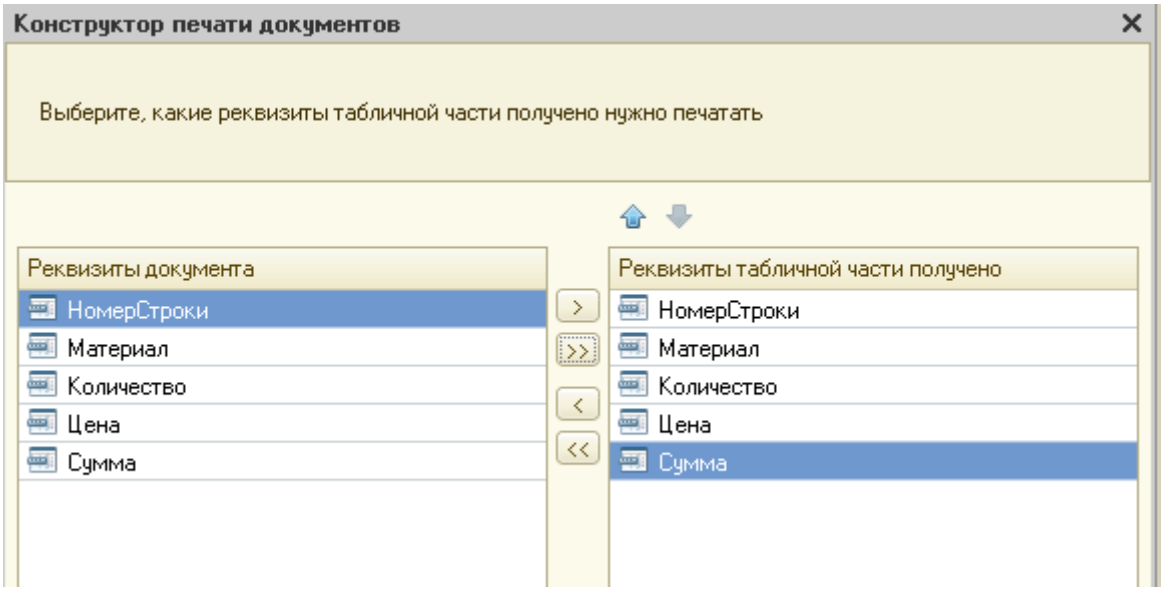


Рисунок 5.38. Выбор реквизитов таблицы.

Получаем макет печатной формы и два модуля (рис.5.39).

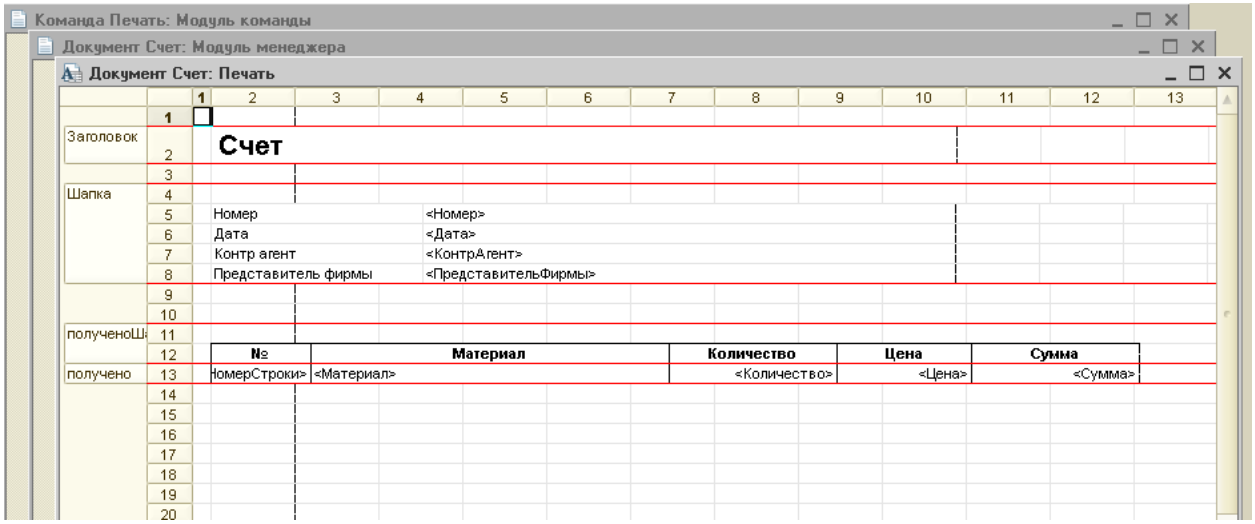


Рисунок 5.39. Макет печатной формы.

В полученной печатной форме нет итоговой строки. Создадим итоговую строку и подпись директора фирмы (рис.5.40-5.41).

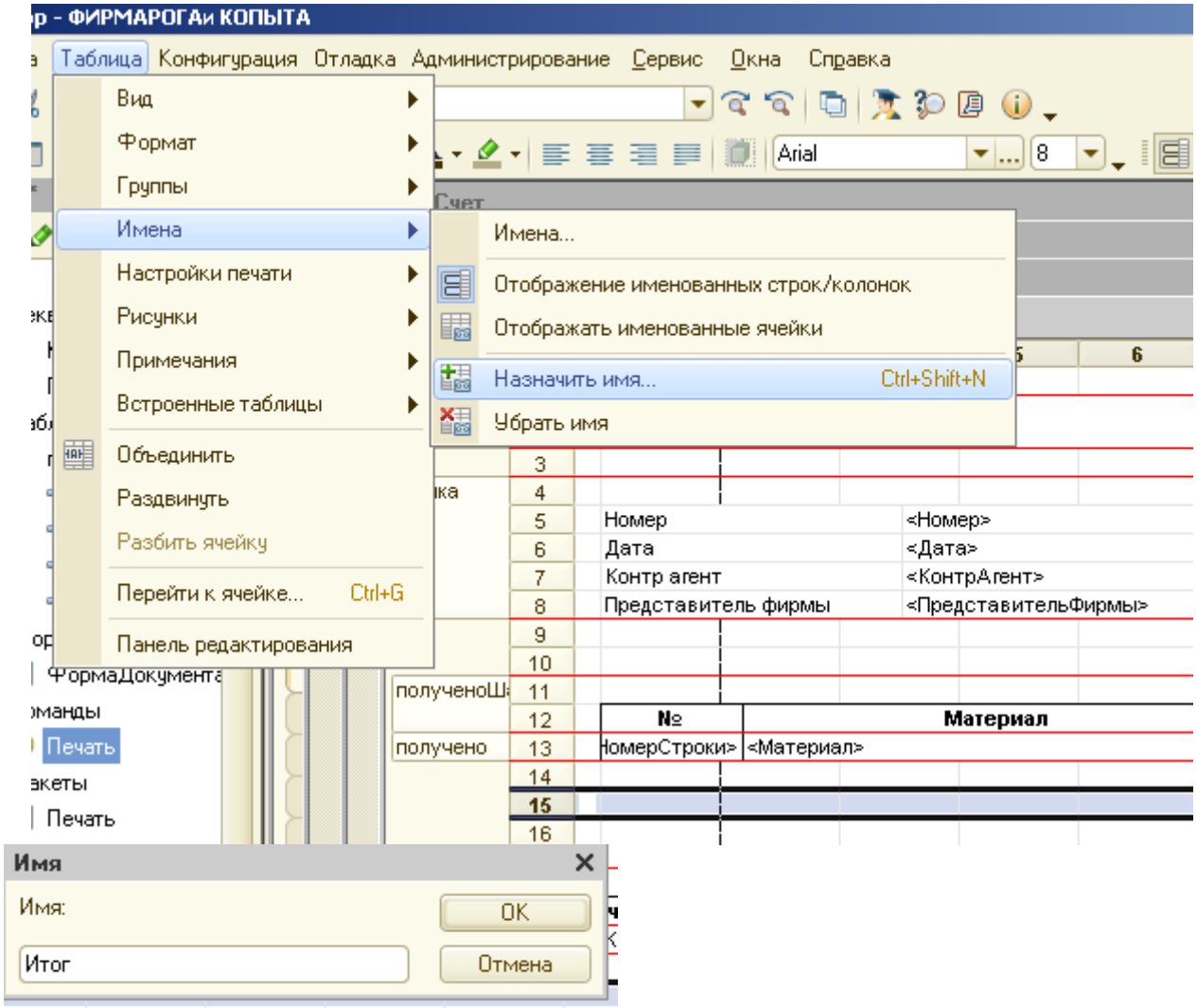


Рисунок 5.40.Назначение имени итоговой строки.

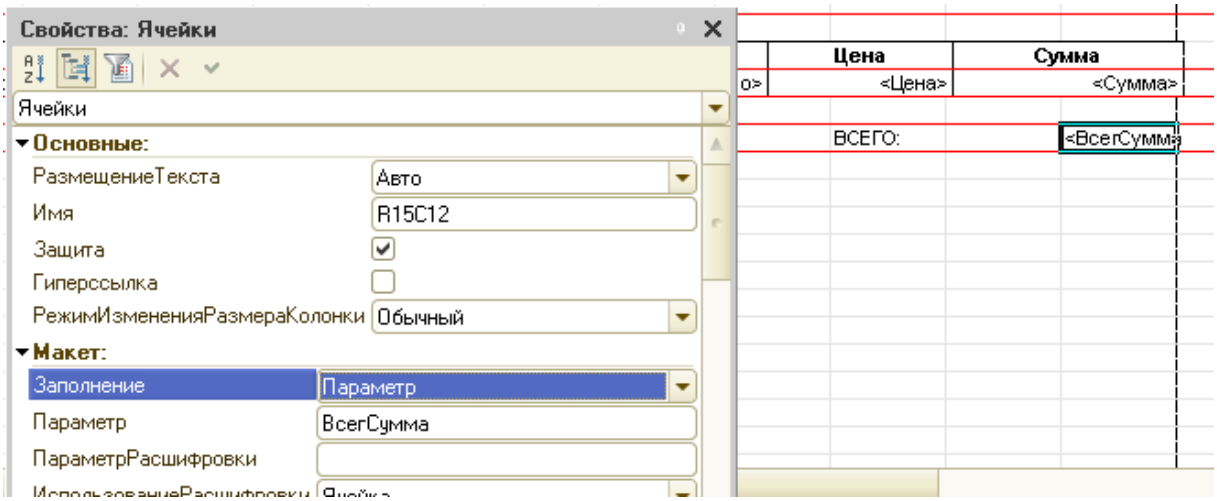


Рисунок 5.41. Формирование строки макета «Итог»

Аналогично добавляем еще одну строку для подписи директора. Нназовем ее подвал (рис.5.42) и открываем модуль для редактирования (рис.5.43).

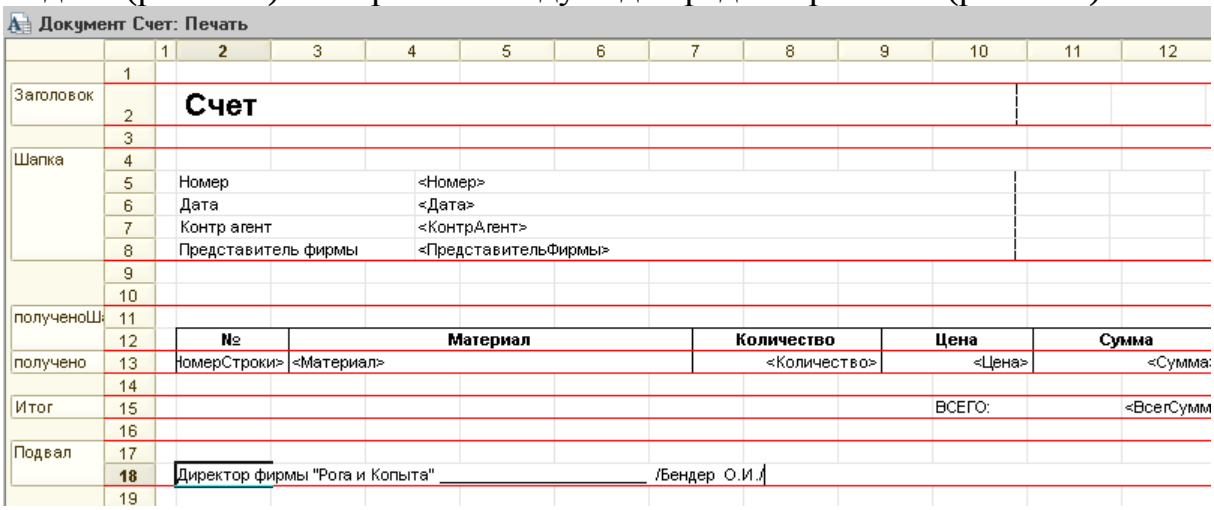


Рисунок 5.42. Отредактированный макет.

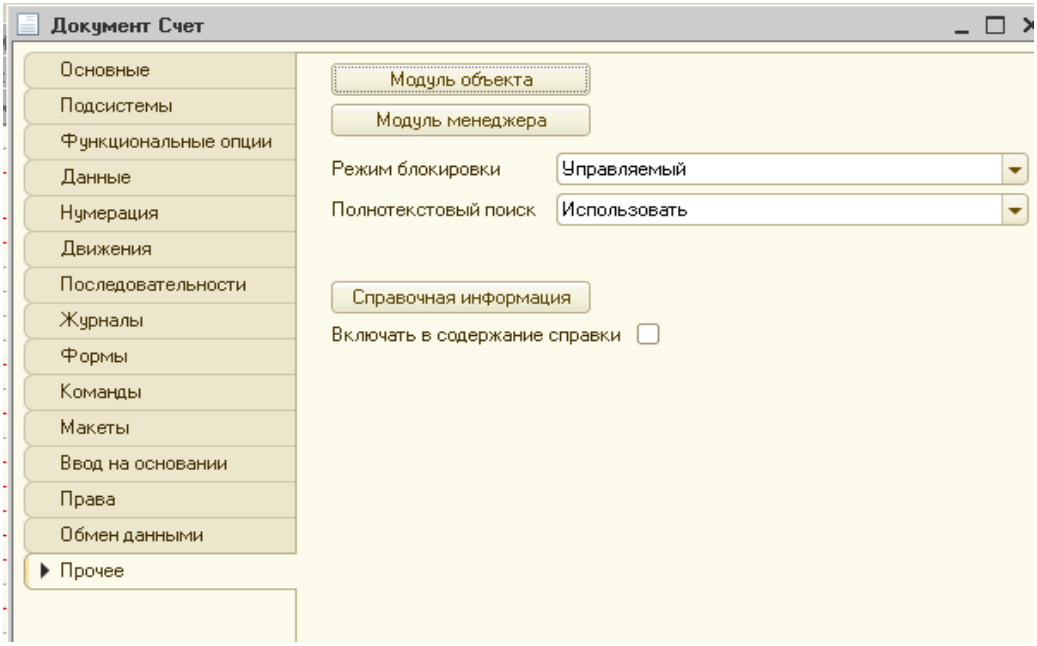


Рисунок 5.43. Редактирование «Модуля менеджера».

Видим код (рис.5.44) и вносим изменения (рис.5.45):

Документ Счет: Модуль менеджера

```
|        Количество,
|        Цена,
|        Сумма
|    )
|ИЗ
|    Документ.Счет КАК Счет
|ГДЕ
|    Счет.Ссылка В (&Ссылка)";
Запрос.Параметры.Вставить("Ссылка", Ссылка);
Выборка = Запрос.Выполнить().Выбрать();

ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
Шапка = Макет.ПолучитьОбласть("Шапка");
ОбластьполученоШапка = Макет.ПолучитьОбласть("полученоШапка");
Областьполучено = Макет.ПолучитьОбласть("получено");
ТабДок.Очистить();

ВставлятьРазделительСтраниц = Ложь;
Пока Выборка.Следующий() Цикл
    Если ВставлятьРазделительСтраниц Тогда
        ТабДок.ВывестиГоризонтальныйРазделительСтраниц();
    КонецЕсли;

    ТабДок.Вывести(ОбластьЗаголовок);

    Шапка.Параметры.Заполнить(Выборка);
    ТабДок.Вывести(Шапка, Выборка.Уровень());

    ТабДок.Вывести(ОбластьполученоШапка);
    Выборкаполучено = Выборка.получено.Выбрать();
    Пока Выборкаполучено.Следующий() Цикл
        Областьполучено.Параметры.Заполнить(Выборкаполучено);
        ТабДок.Вывести(Областьполучено, Выборкаполучено.Уровень());
    КонецЦикла;

    ВставлятьРазделительСтраниц = Истина;
КонецЦикла;
//}}
КонецПроцедуры
```

Рисунок 5.44. Фрагмент кода «Модуля менеджера».

Документ Счет: Модуль менеджера

```
|    )
|ИЗ
|    Документ.Счет КАК Счет
|ГДЕ
|    Счет.Ссылка В (&Ссылка)";
Запрос.Параметры.Вставить("Ссылка", Ссылка);
Выборка = Запрос.Выполнить().Выбрать();

ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
Шапка = Макет.ПолучитьОбласть("Шапка");
ОбластьполученоШапка = Макет.ПолучитьОбласть("полученоШапка");
Областьполучено = Макет.ПолучитьОбласть("получено");
 ОбластьИтог=Макет.ПолучитьОбласть("Итог");  //****наша строка
  ОбластьПодвал=Макет.ПолучитьОбласть("Подвал");      //****наша строка
    ТабДок.Очистить();
ВставлятьРазделительСтраниц = Ложь;
Пока Выборка.Следующий() Цикл
    Если ВставлятьРазделительСтраниц Тогда
        ТабДок.ВывестиГоризонтальныйРазделительСтраниц();
    КонецЕсли;

    ТабДок.Вывести(ОбластьЗаголовок);

    Шапка.Параметры.Заполнить(Выборка);
    ТабДок.Вывести(Шапка, Выборка.Уровень());

    ТабДок.Вывести(ОбластьполученоШапка);
    Выборкаполучено = Выборка.получено.Выбрать();
    СуммаИ=0;  //****наша строка

    Пока Выборкаполучено.Следующий() Цикл
        Областьполучено.Параметры.Заполнить(Выборкаполучено);
        ТабДок.Вывести(Областьполучено, Выборкаполучено.Уровень());
        СуммаИ=СуммаИ+Выборкаполучено.Сумма;    //****наша строка
    КонецЦикла;
     ОбластьИтог.Параметры.ВсегоСумма=СуммаИ;  //****наша строка
     ТабДок.Вывести(ОбластьИтог);              //****наша строка
         ТабДок.Вывести(ОбластьПодвал);        //****наша строка
```

Рисунок 5.45.Фрагмент кода с внесенными изменениями.

В результате получаем улучшенный счет, а, нажав кнопку «Печать» (рис.5.46), получаем печатную форму для вывода на бумагу и предъявления клиенту (рис.5.47).

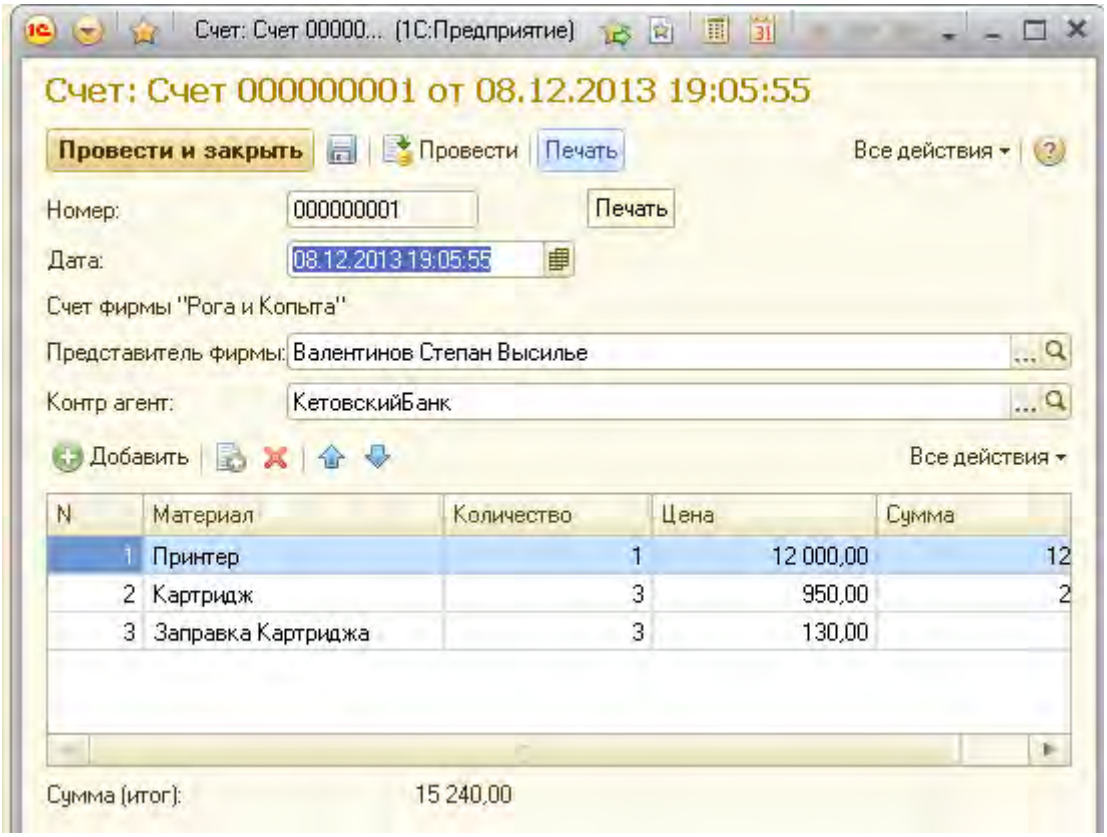


Рисунок 5.46. Форма с кнопкой «Печать».

Рисунок 5.47. Счет.

Вопросы для самоконтроля

1. Что такое документ и для чего он предназначен?
2. Что означает проведение документа?
3. Для чего предназначены реквизиты документа?
4. Как создать документ?
5. Что такое табличная часть документа?
6. Как создать форму документа?
7. Что такое конструктор форм?

8. Как создать элементы формы?
9. Что такое события?
10. Что такое обработчик событий?
11. Как создать обработчик событий?
12. Что такое модуль и для чего он нужен?
13. Для чего нужен объект конфигурации макет?
14. Что такое конструктор печати?
15. Как создать макет?
16. Как редактировать макет?
17. Как создавать и выводить на печать новые области?
18. Что такое модуль менеджера?

Задание для самостоятельного выполнения:

1. Создать документ «Приходный кассовый ордер (ПКО)», его формы и печатный макет (рис.5.48).

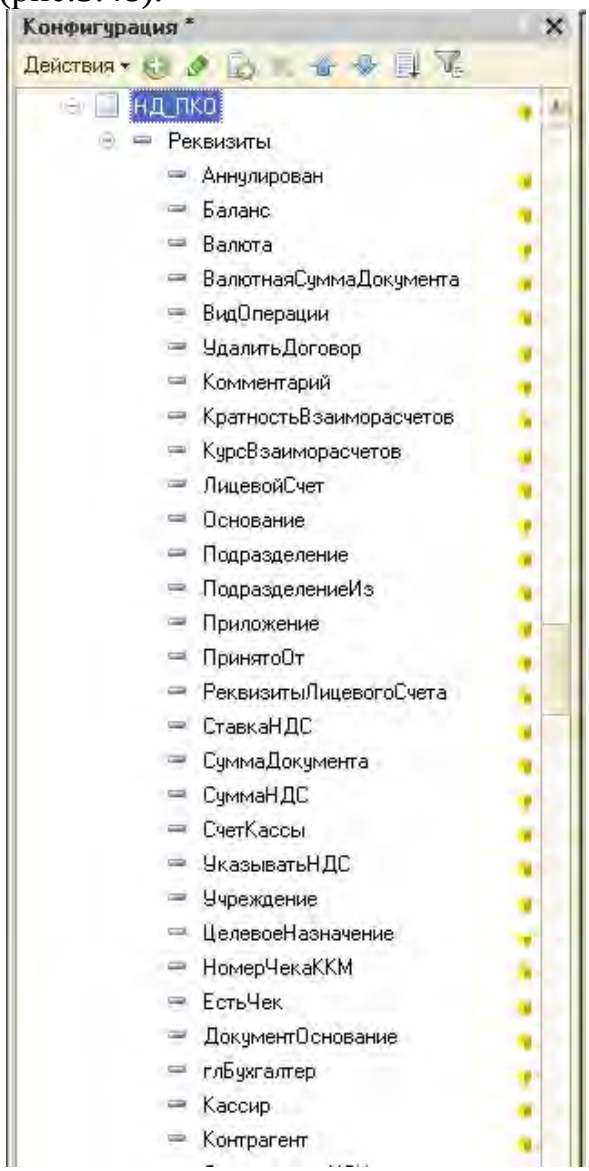


Рисунок 5.48. Примерный перечень реквизитов «Приходного кассового ордера (ПКО)»

2. Создать документ «Счет», его формы и печатный макет (рис.5.49).

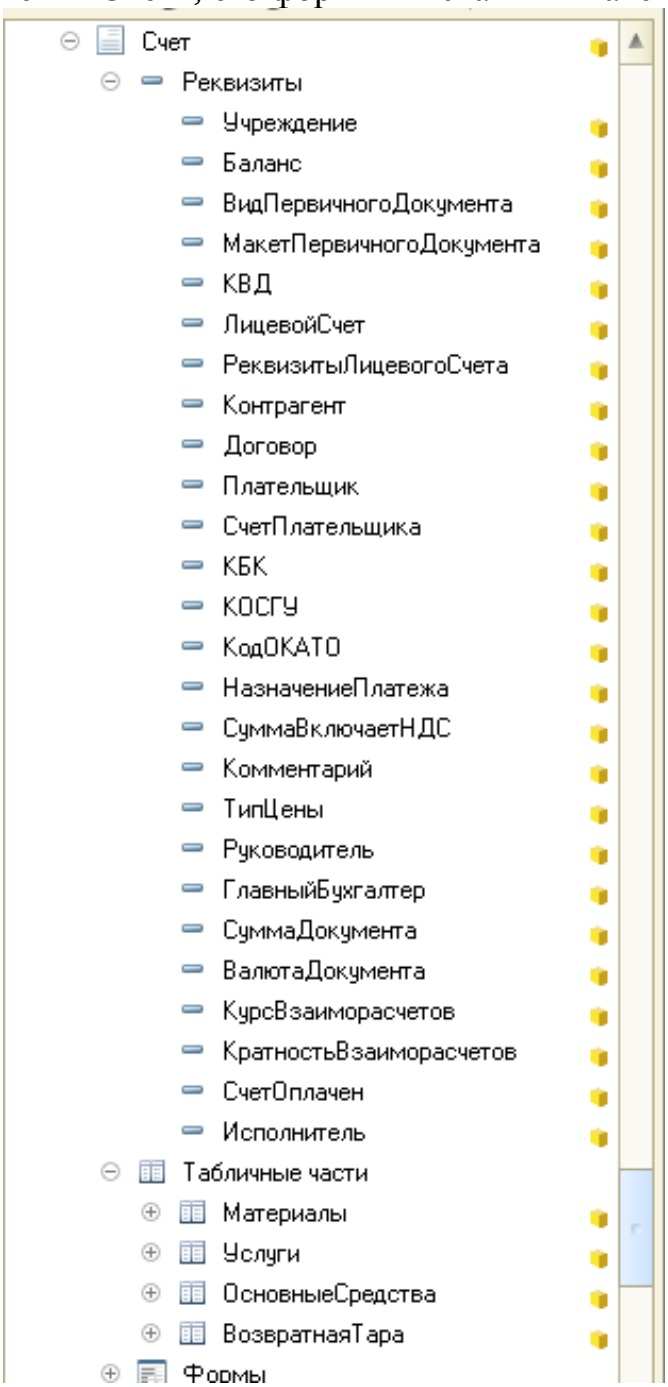


Рисунок 5.49. Примерный перечень реквизитов документа «Счет».

3. Создать документ оказания услуг организации «УслугиСторонниxОрганизаций», его формы и печатный макет (рис.5.50).

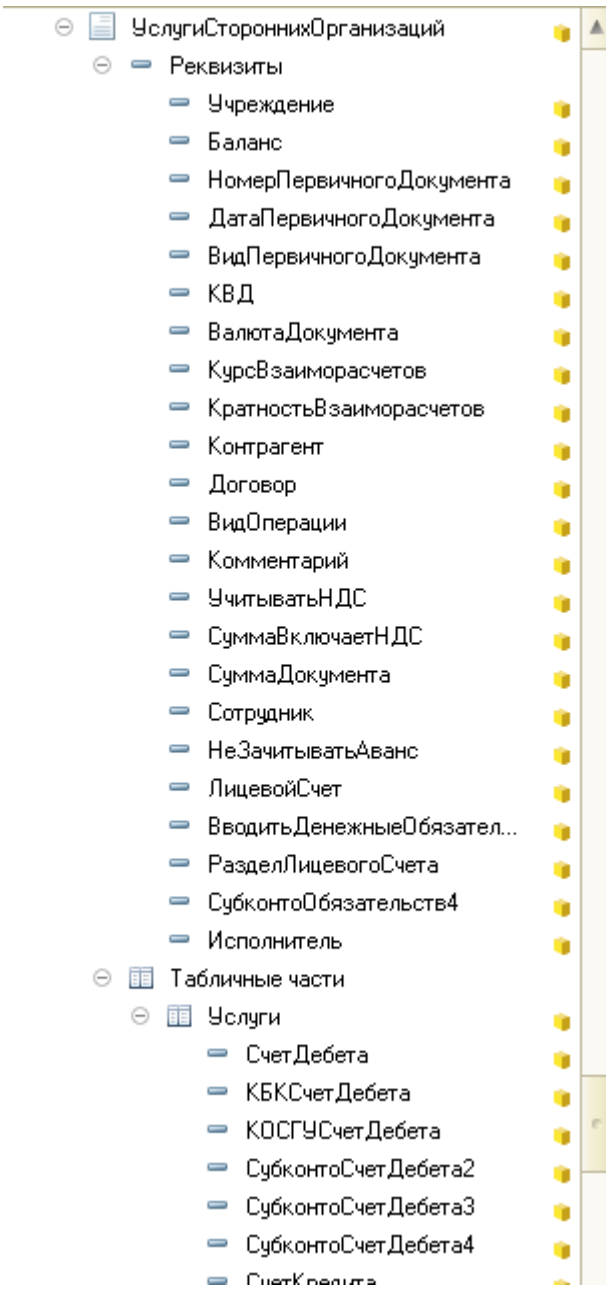


Рисунок 5.50. Примерный перечень реквизитов документа «Услуги Сторонних Организаций».

6. Учет движения документов, регистры накопления

В технологических платформах 1С: Предприятие 8.2 среди предопределенных объектов метаданных имеется в арсенале 4 типа регистров: Регистр накопления, Регистр сведений, Регистр бухгалтерии, Регистр расчета.

Регистры накопления служат для описания структуры накопления данных. В регистре хранятся данные, поставляемые различными объектами конфигурации. Казалось бы, можно обойтись и без регистров накопления, вся нужная информация хранится в документах, достаточно их проанализировать и мы получим отчеты. Однако надо учесть тот факт, что на практике часто возникает необходимость введения дополнительных документов в конфигурацию и тогда для изменения учета необходимо внесение изменений во все использующие их отчеты. Если же мы используем регистр, то достаточно будет включить документ в этот регистр. И с другой стороны если база большая, то анализ документов может занять значительное время, регистр в этом случае работает существенно быстрее [1].

Особенностью регистра, является, то, что он не предназначен для редактирования пользователем, однако разработчик может предоставить такую возможность пользователям.

Накопление данных в регистре происходит в разрезе нескольких измерений, которые описываются в конфигураторе. В измерениях мы можем учитывать движение интересующих нас ресурсов, которые должны быть числовыми. Изменение регистра происходит, как правило, при проведении документа. В регистр добавляются новые записи, содержащие значение измерений и приращение ресурсов, и ссылку на документ «регистратор». Рассмотрим пример создания регистра накопления (рис.6.1) [4].

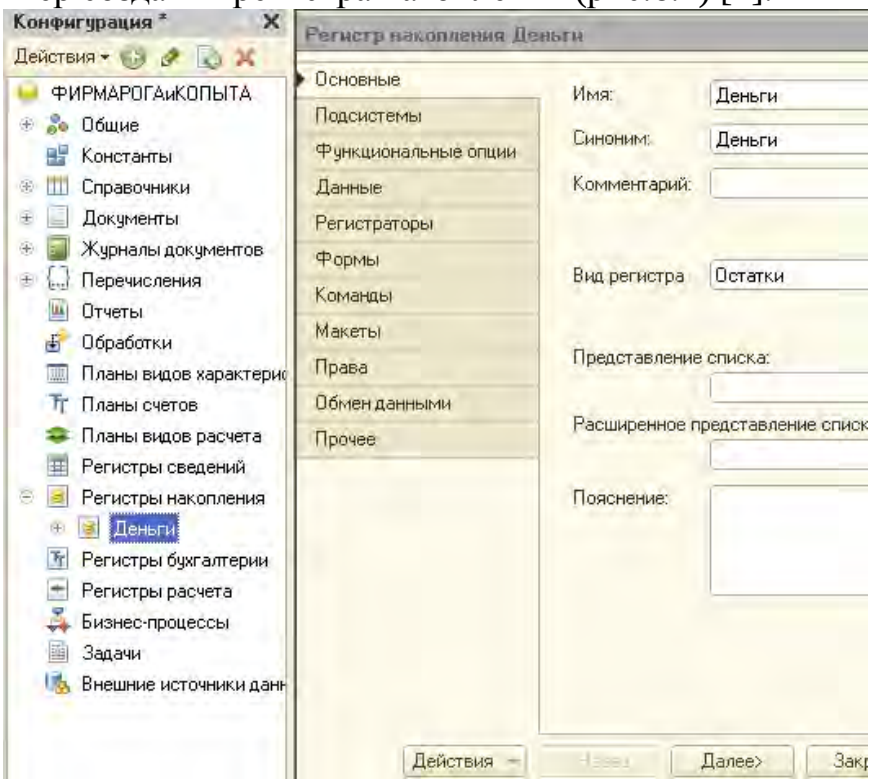


Рисунок 6.1. Создание регистра накопления.

Назовем регистр «Деньги», в данном случае нас будет интересовать поступление и расходование денежных средств (рис.6.2).

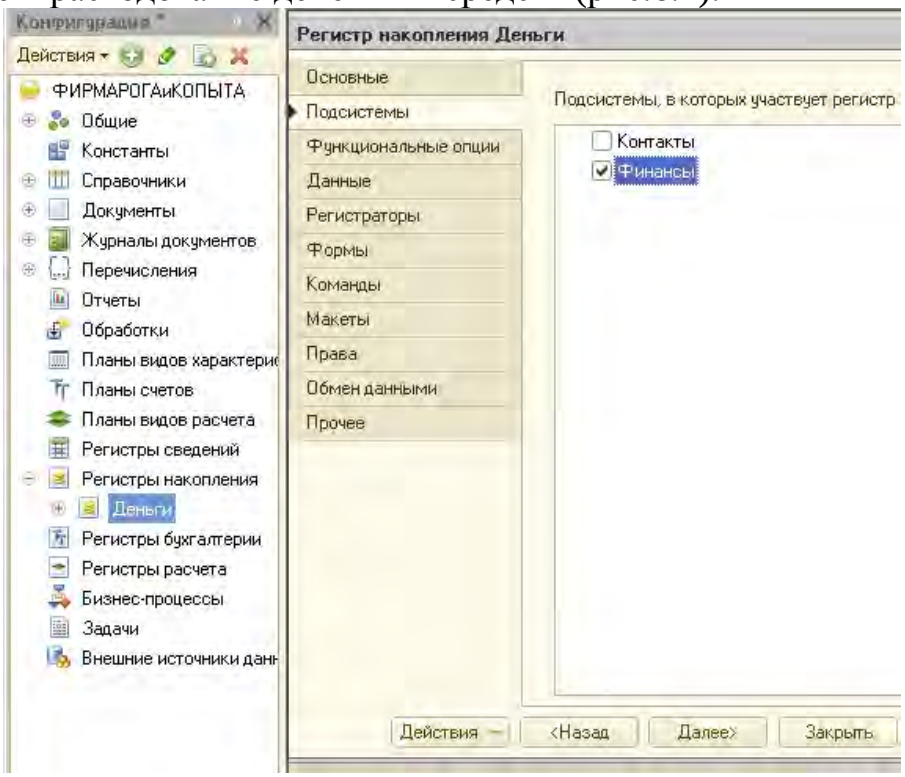


Рисунок 6.2. Назначение подсистемы.

Теперь выбираем измерения, по которым нам интересно учитывать движение денежных средств. Очень полезно знать, на что деньги тратились и от продажи чего мы получали доход, потому измерение «Материал» который является ссылкой на справочник «Номенклатура» (рис.6.3).

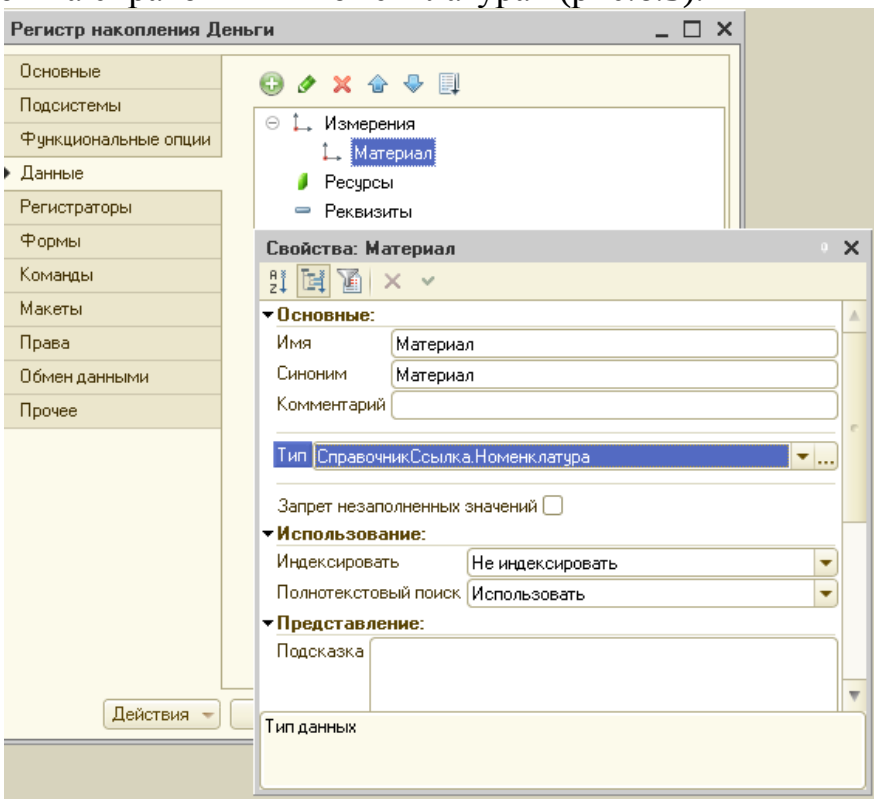


Рисунок 6.3. Назначение измерения «Материал».

Другое измерение, которое обязательно заинтересует нас - это откуда к нам деньги поступали и куда уходили, значит, измерение «Контрагент» со ссылкой на соответствующий справочник (рис.6.4).

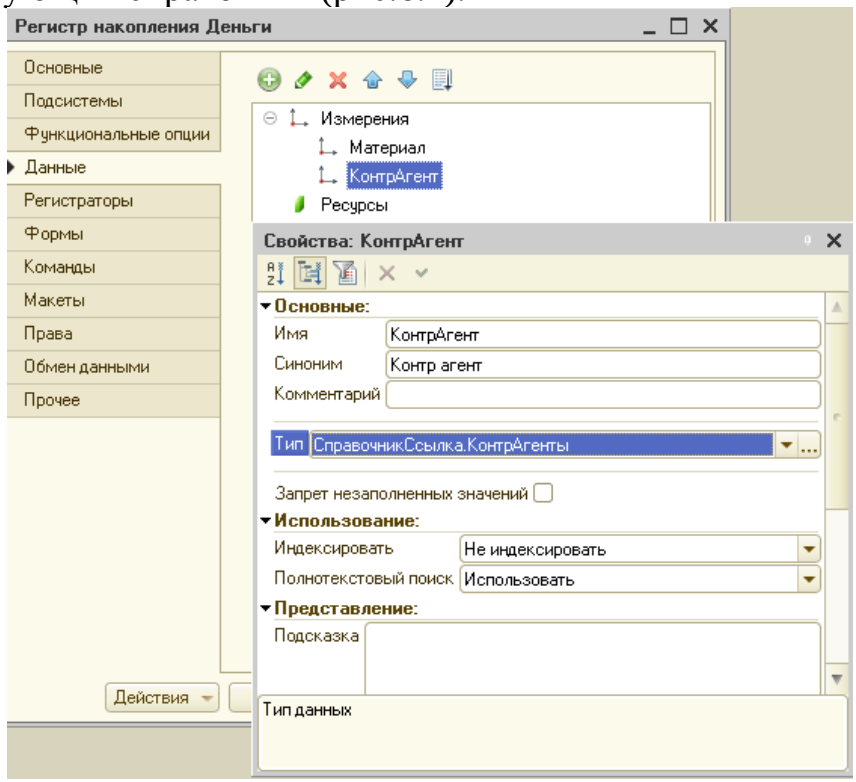


Рисунок 6.4. Назначение измерения «Контрагент».

С ресурсом пока думать особо не приходится, нас интересуют денежные потоки, следовательно, называем «Сумма» и выбираем подходящий числовой формат (рис.6.5).

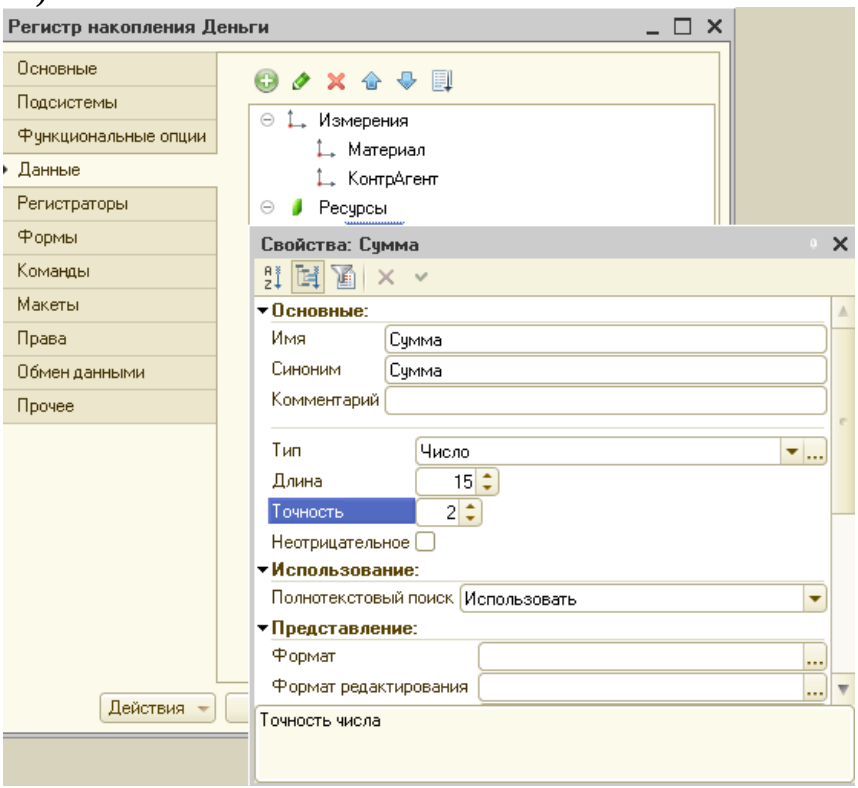


Рисунок 6.5. Назначение ресурса «Сумма».

Теперь перейдем к документам и назначим им движение (рис.6.6-6.8).

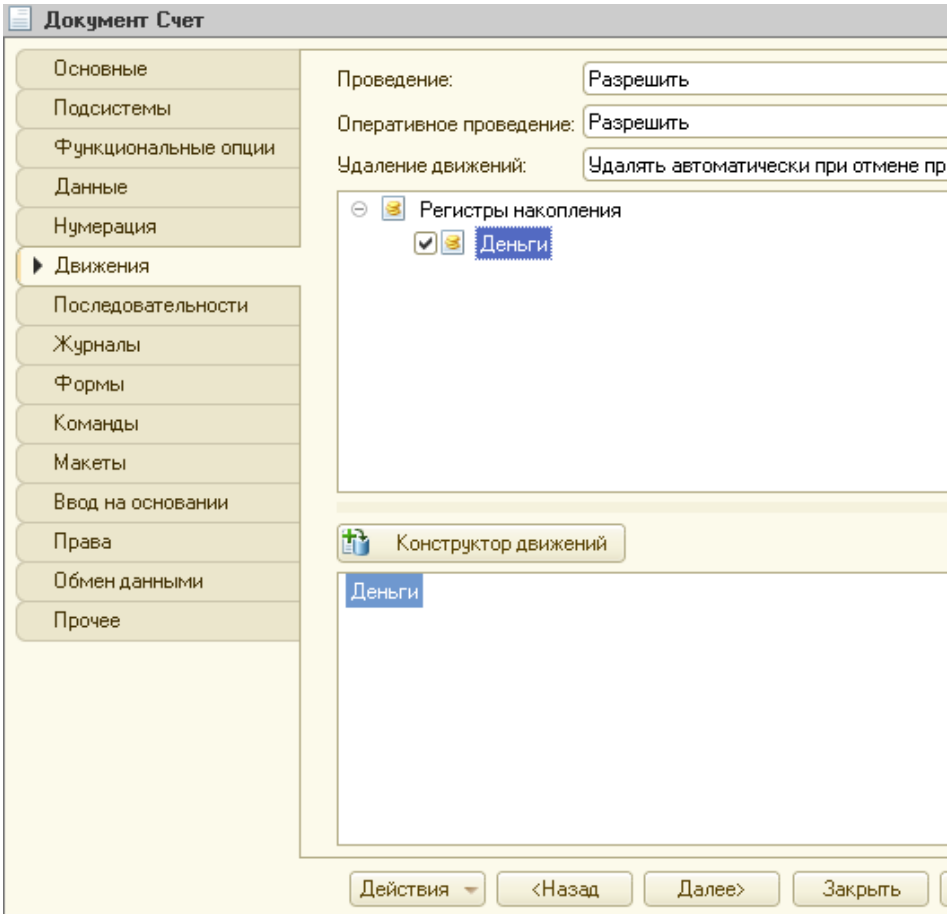


Рисунок 6.6. Регистрация движения документа «Счет».

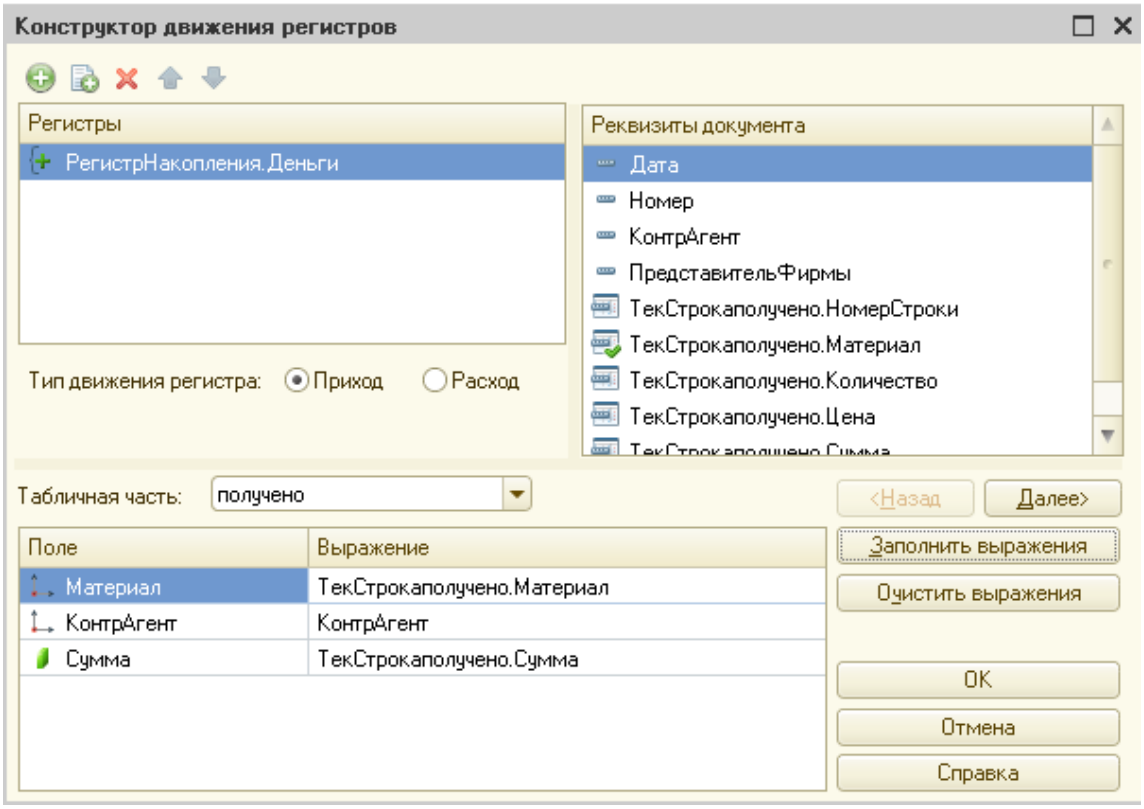


Рисунок 6.7. Работа конструктора движения регистров.

Документ Счет: Модуль объекта

```
Процедура ОбработкаПроведения(Отказ, Режим)
    //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
    // Данный фрагмент построен конструктором.
    // При повторном использовании конструктора, внесенные вру

    // регистр Деньги Приход
    Движения.Деньги.Записывать = Истина;
    Для Каждого ТекСтрокаполучено Из получено Цикл
        Движение = Движения.Деньги.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Период = Дата;
        Движение.Материал = ТекСтрокаполучено.Материал;
        Движение.КонтрАгент = КонтрАгент;
        Движение.Сумма = ТекСтрокаполучено.Сумма;
    КонецЦикла;

    //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Рисунок 6.8. Модуль проведения документа «Счет».

Отразим расход денег с использованием документа «Накладная» (рис.6.9-6.11).

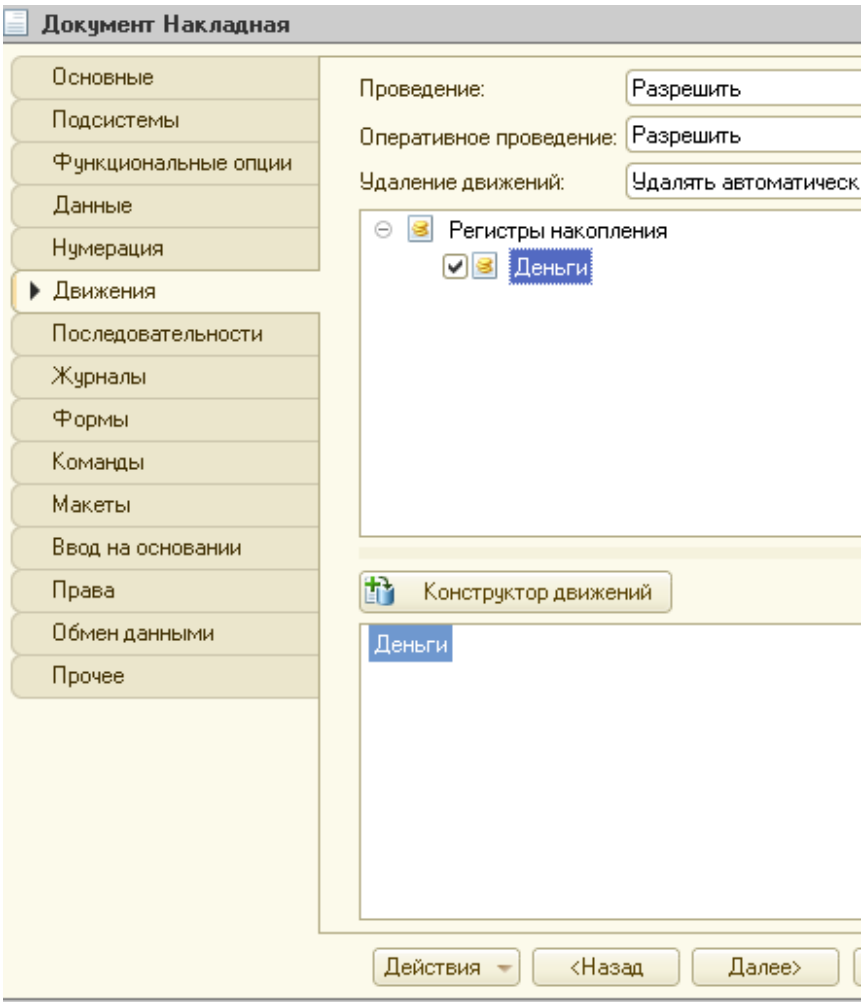


Рисунок 6.9 Формирование движения документа «Накладная».

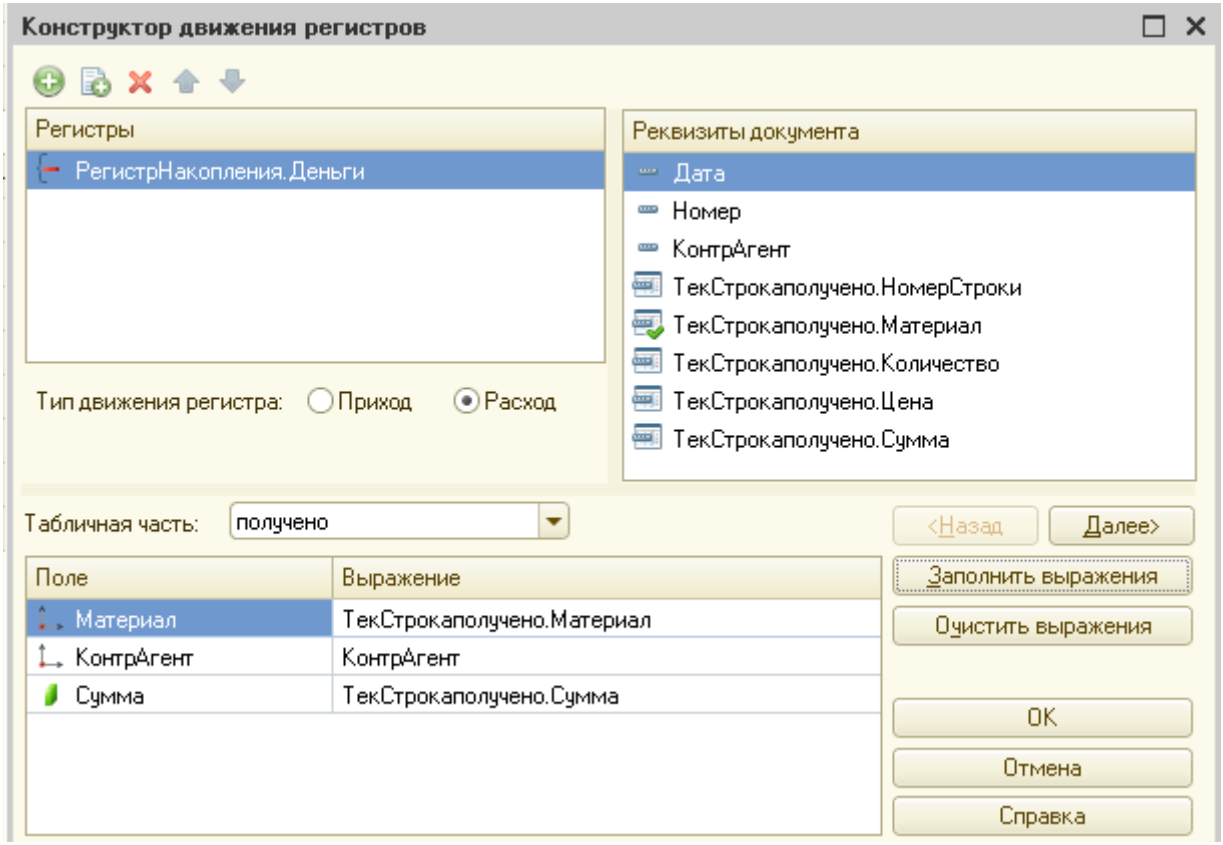


Рисунок 6.10. Конструктор движения документа «Накладная».

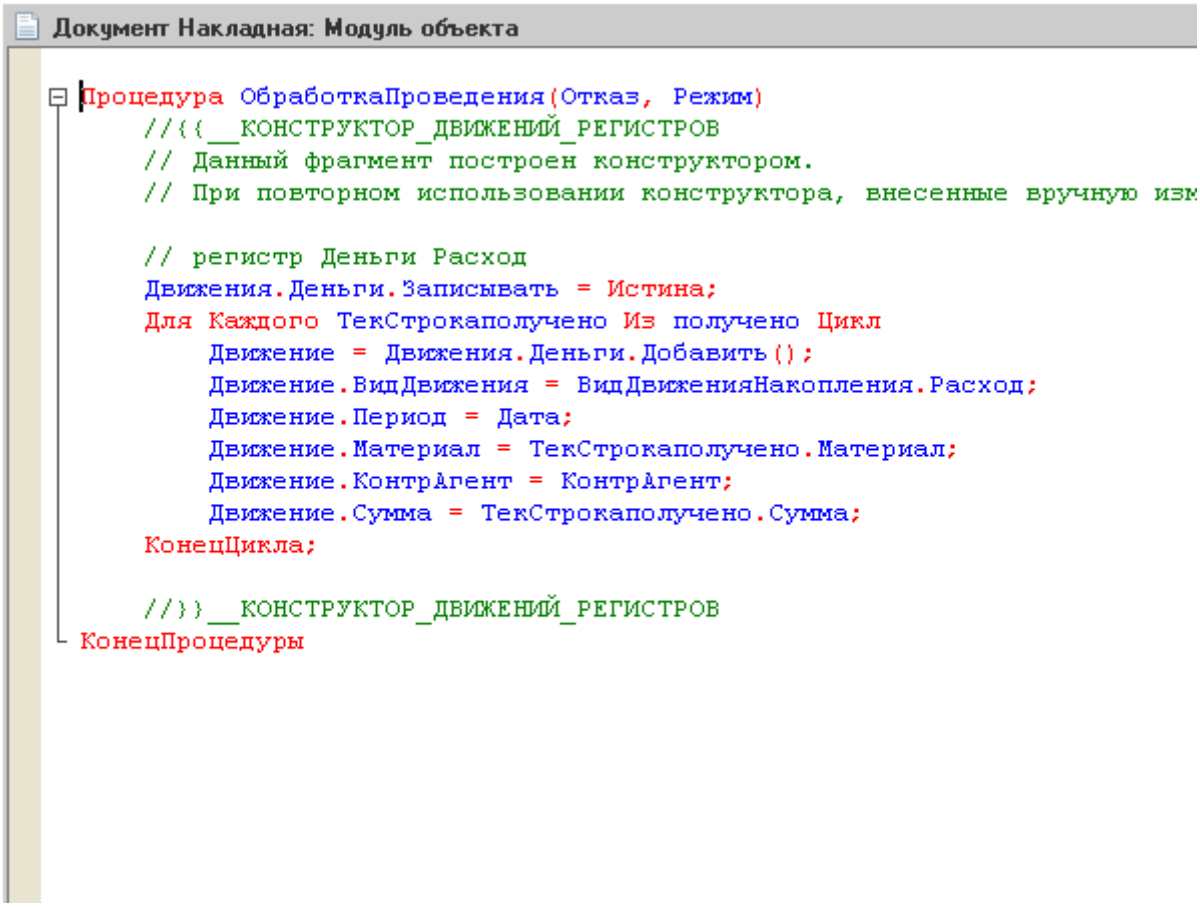
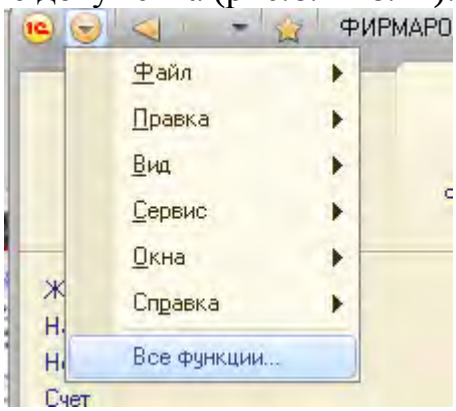


Рисунок 6.11. Модуль проведения документа «Накладная».

Проверяем проведение документа (рис.6.12-6.17):



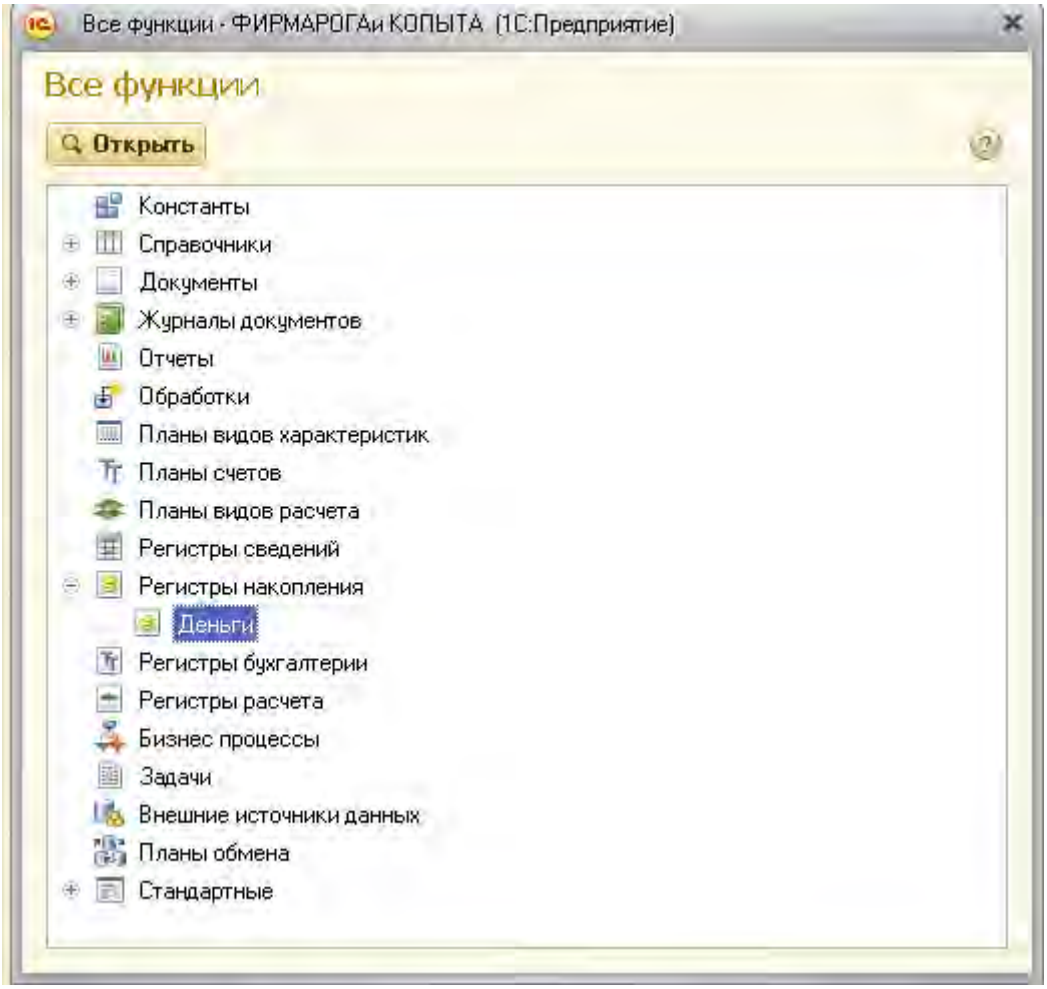


Рисунок 6.12. Доступ из главного меню к регистру.

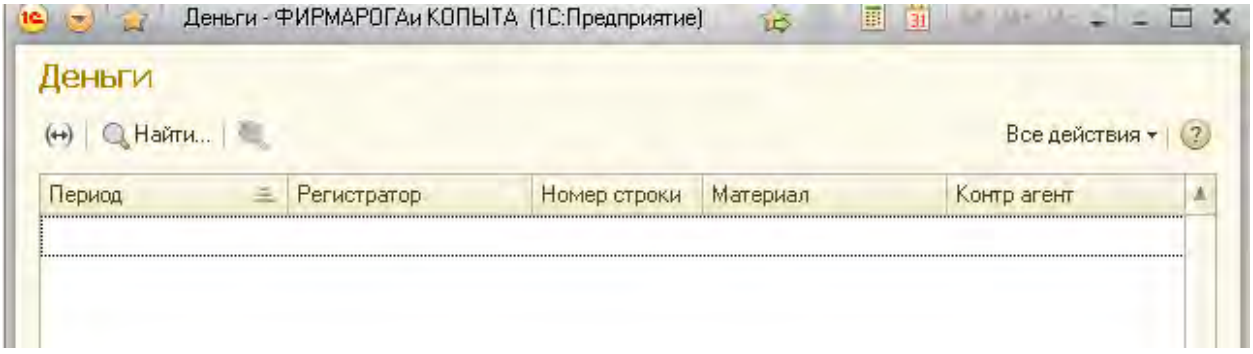


Рисунок 6.13. Пустой регистр (нет проведенных документов)

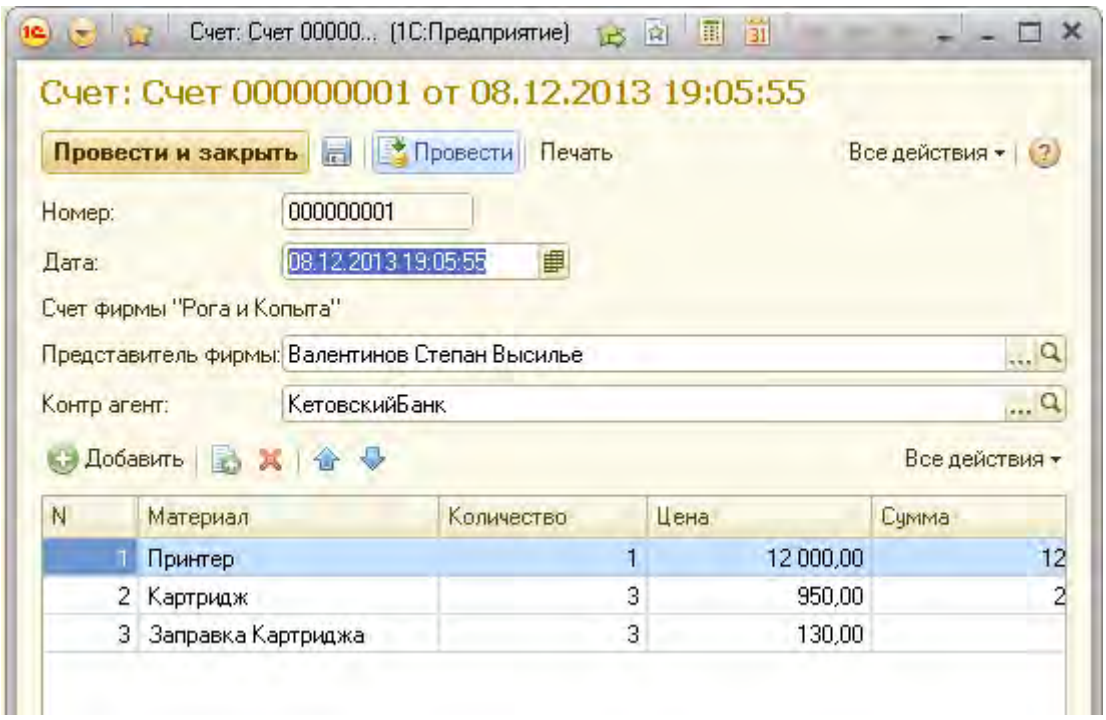


Рисунок 6.14 Проведение документа «Счет».

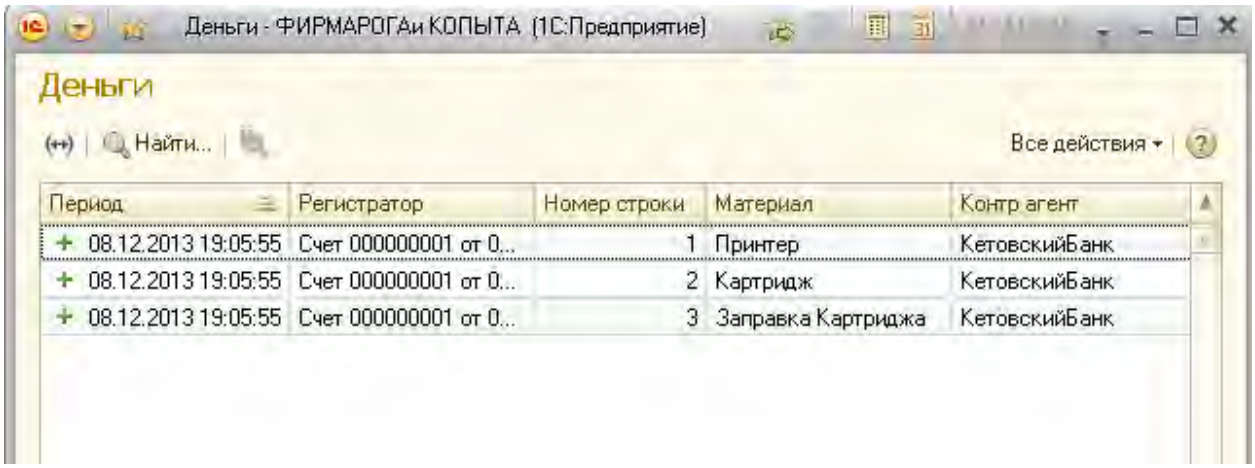


Рисунок 6.15. Данные, записанные в регистр.

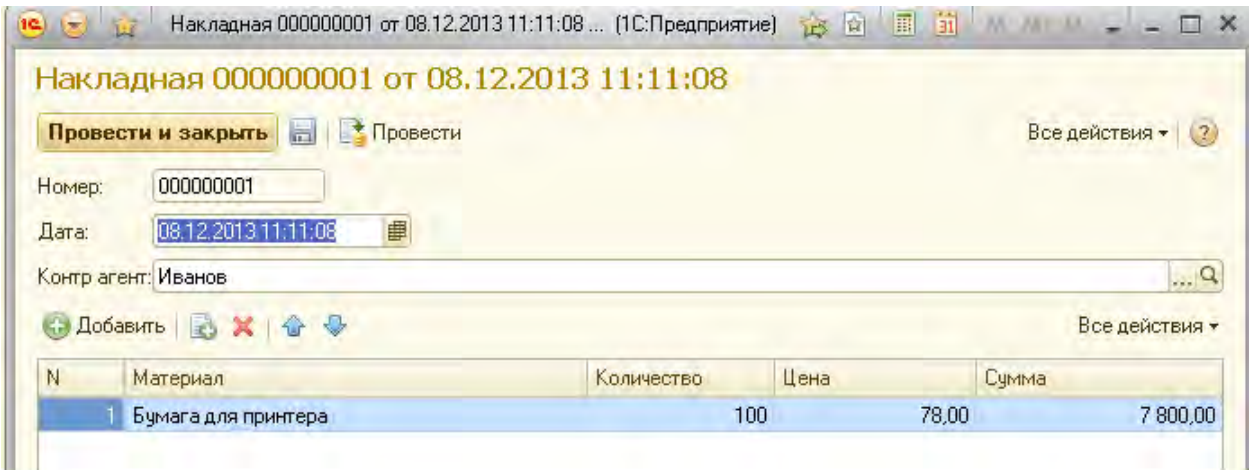


Рисунок 6.16. Проведение документа «Накладная».

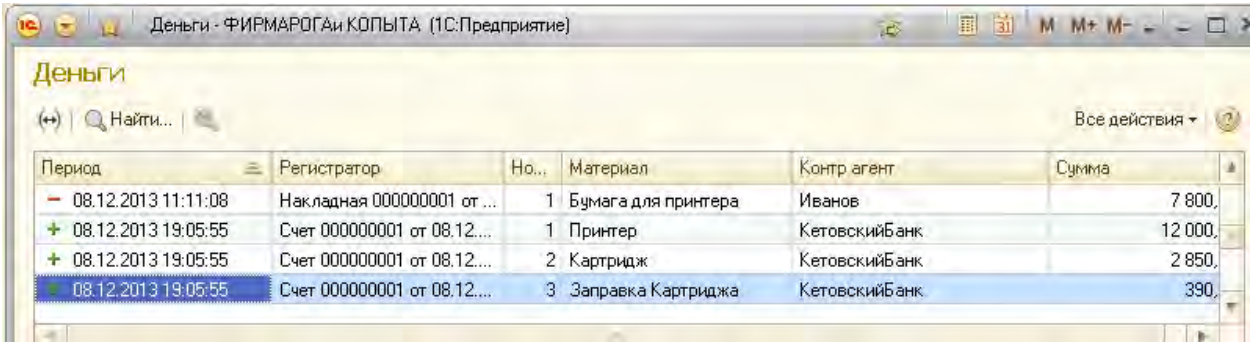


Рисунок 6.17. Регистр с новыми данными.

Вопросы для самоконтроля

1. Что такое регистры накопления?
2. Для чего следует использовать этот объект конфигурации?
3. Что такое измерения регистра накопления?
4. В чем преимущество использования регистров накопления?
5. Как создать движение документа?
6. Как создать ресурс регистра?

Задание для самостоятельного выполнения

Создайте регистр для учета движения количества.

7. Отчеты

Отчеты дают возможность пользователю получить информацию в удобной для него форме и проводить анализ деятельности предприятия. В реально работающем предприятии именно отчеты служат основой анализа деятельности предприятия и принятия управленческих решений.

Дальше открываем схему компоновки данных (рис.7.1).

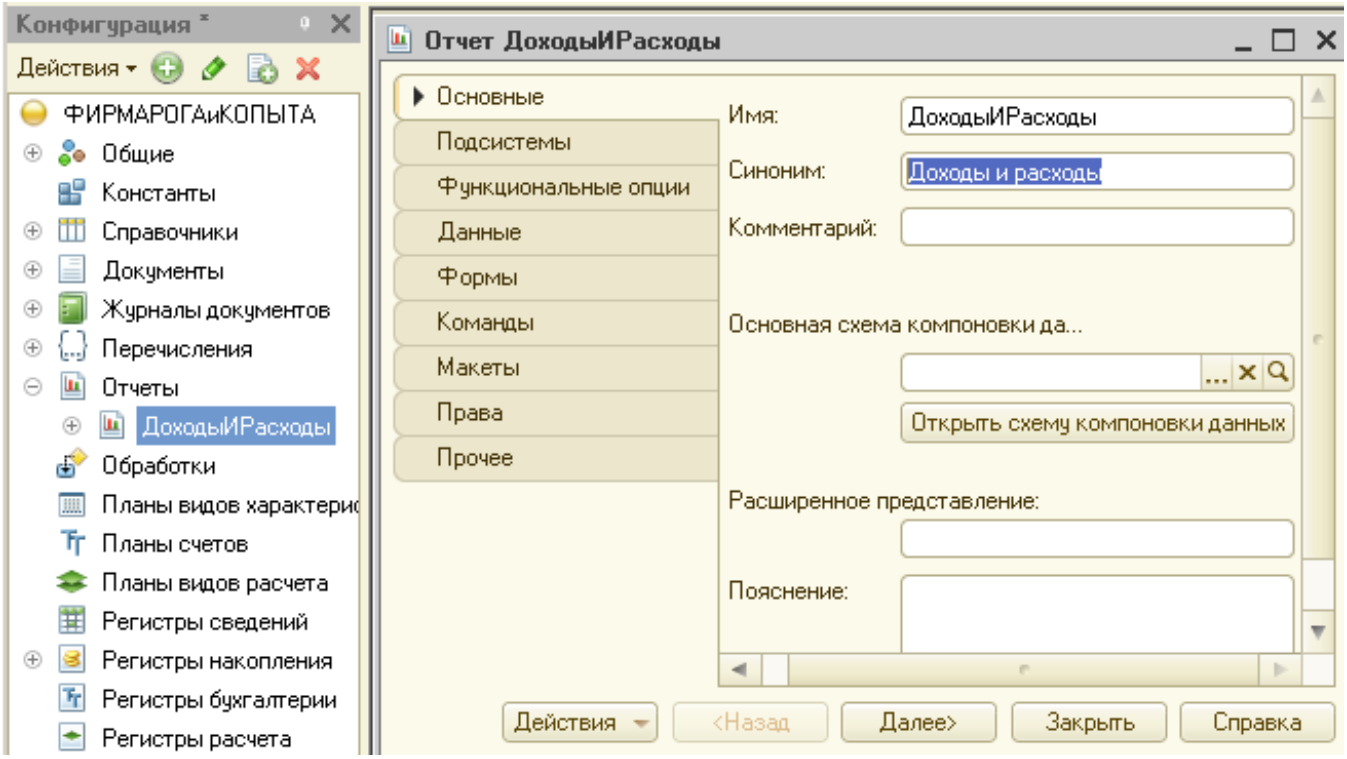


Рисунок 7.1. Создание отчета.

Выбираем источники данных (рис.7.2).

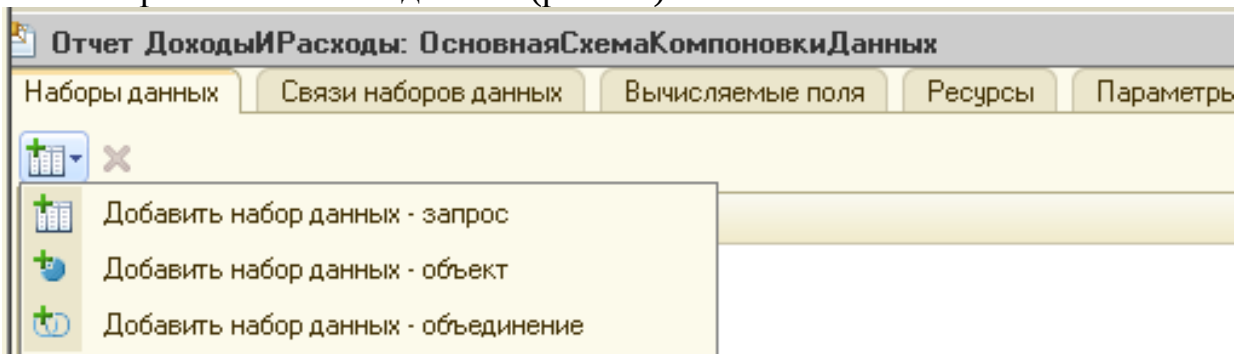


Рисунок 7.2. Выбор набора данных.

Из предложенных вариантов выбираем запрос, т. е. будем формировать запрос к объектам базы данных (рис.7.3-7.4).

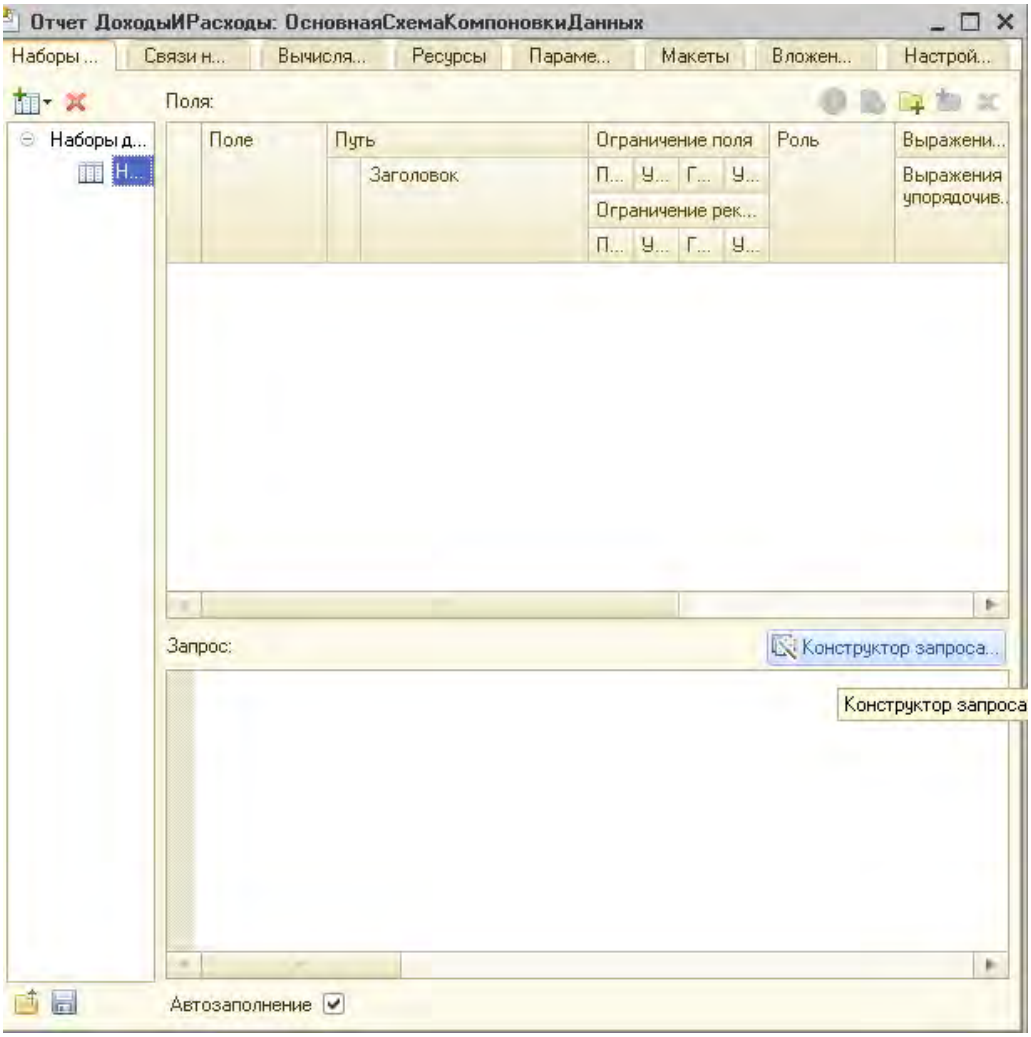


Рисунок 7.3. Вызов конструктора запросов.

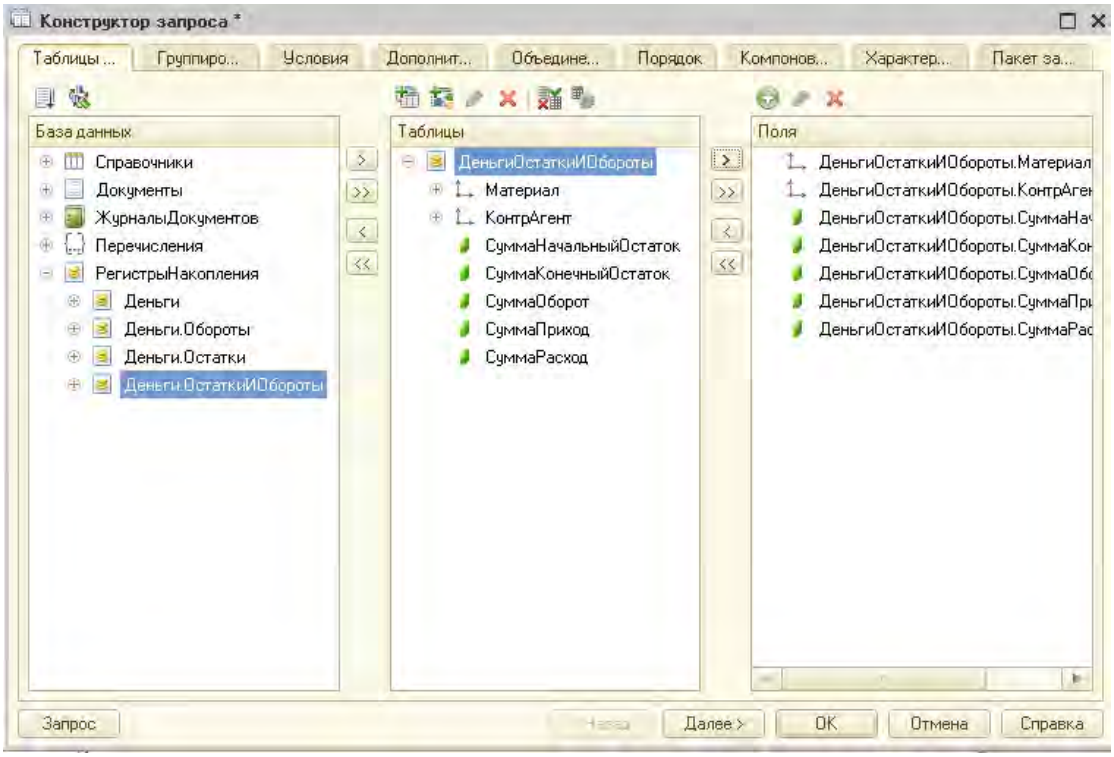


Рисунок 7.4. Таблицы в конструкторе запросов.

Выбираем регистры накопления и в нем регистр «Деньги», в котором «Деньги.ОстаткиОбороты». Нажимаем на кнопку «>>», и дальше из средней таблицы то же самое - кнопка «>>» (рис.7.5).

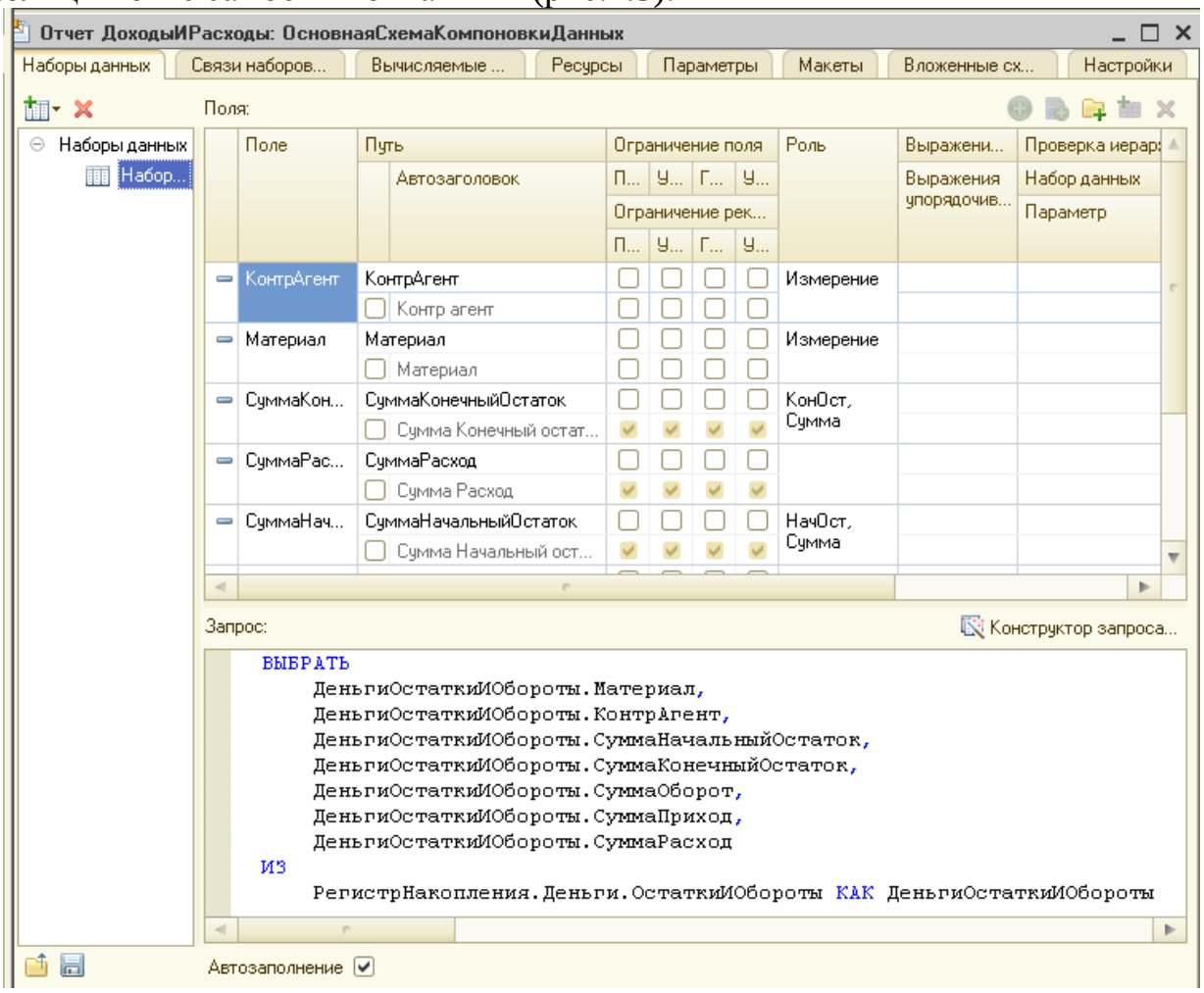


Рисунок 7.5. Текст запроса.

На вкладке ресурсы выбираем ресурсы (рис.7.6), вызываем конструктор надстроек (рис.7.7), выбираем данные и формируем структуру отчета (рис.7.8.-7.11.).

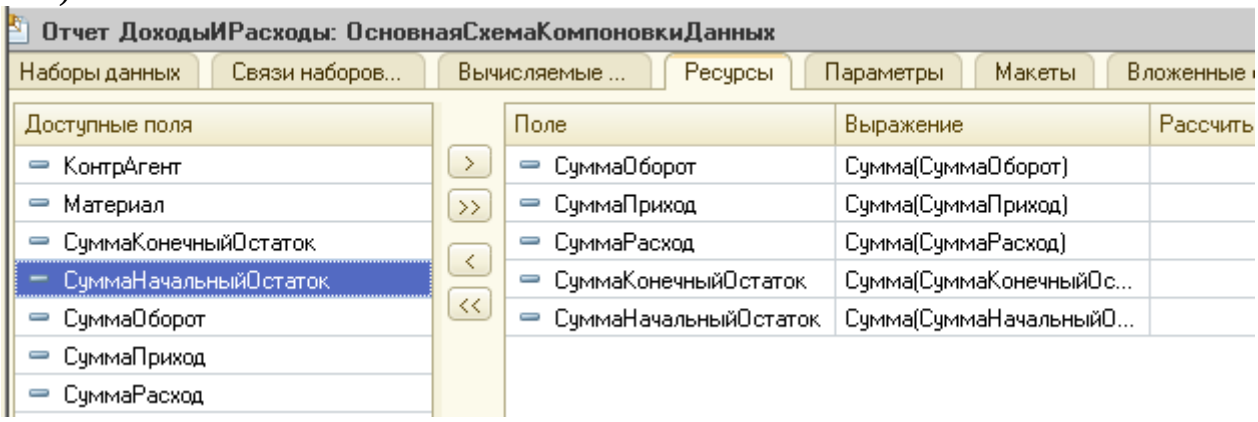


Рисунок 7.6. Выбор ресурсов.

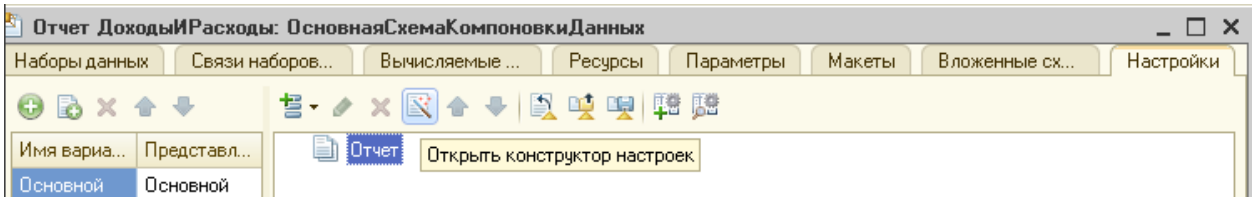


Рисунок 7.7. Вызов конструктора настроек.

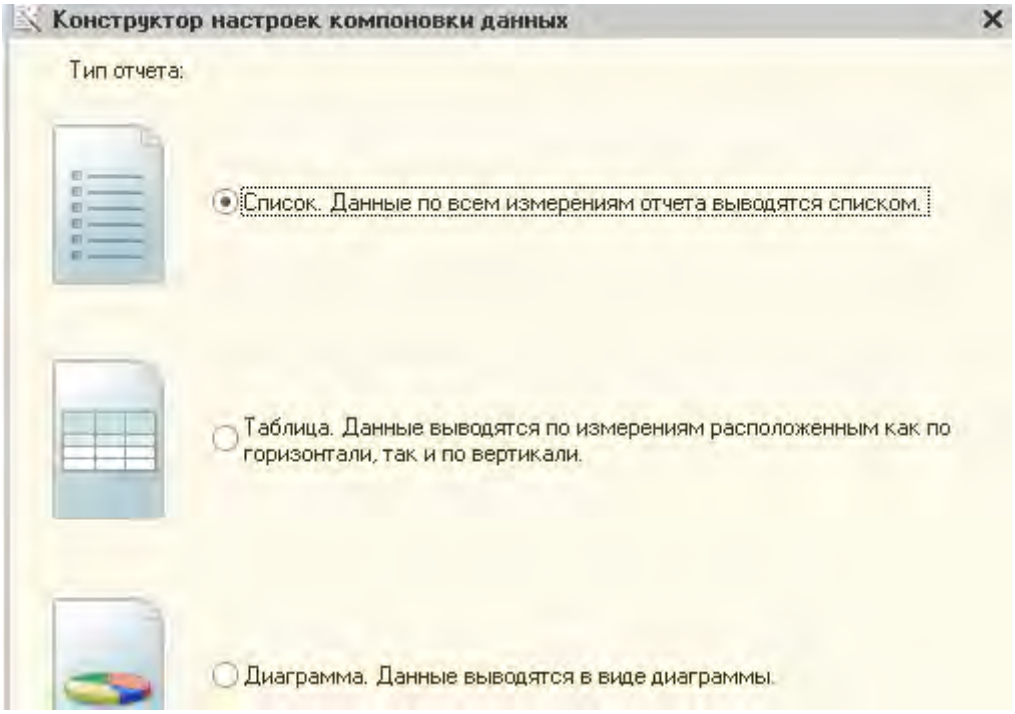


Рисунок 7.8. Выбор списка.

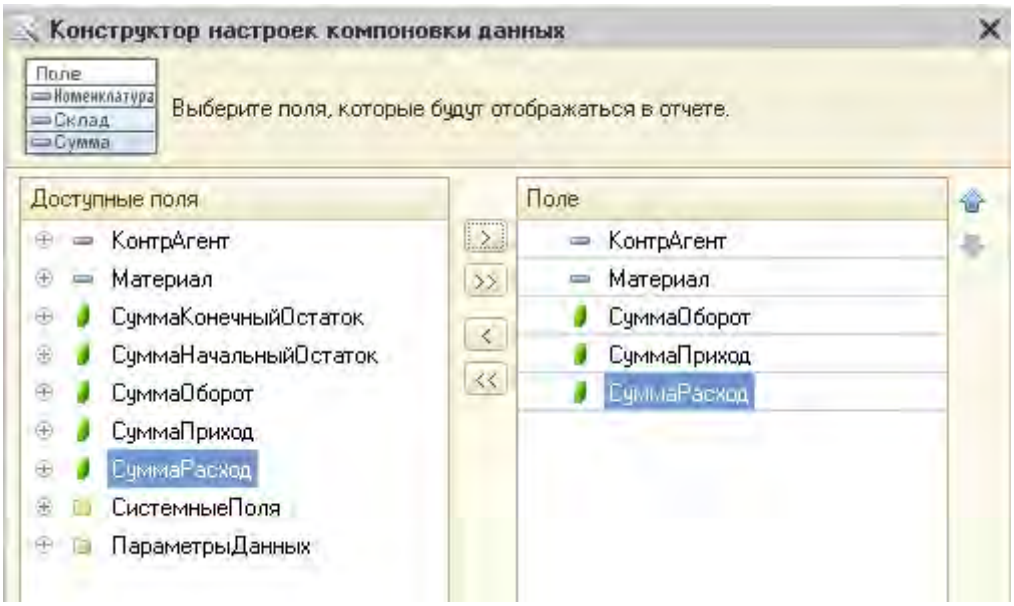


Рисунок 7.9. Выбор полей в отчет.

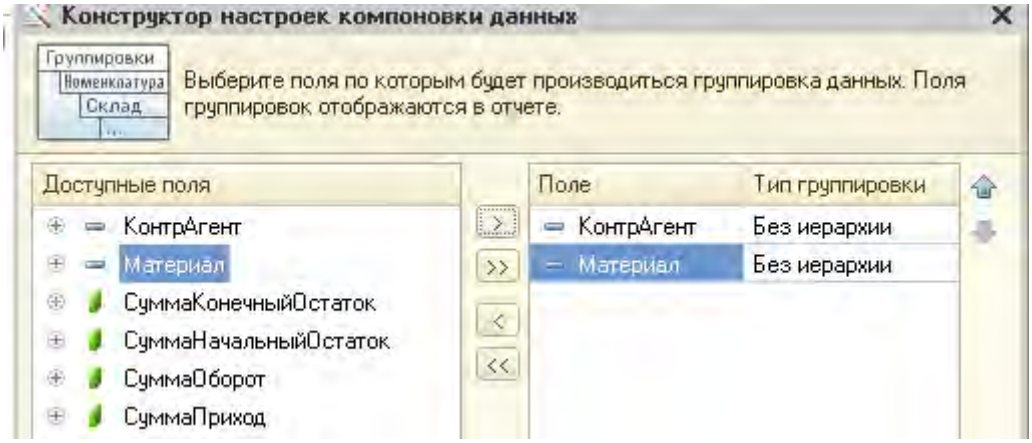


Рисунок 7.10 Выбор полей группировки данных.

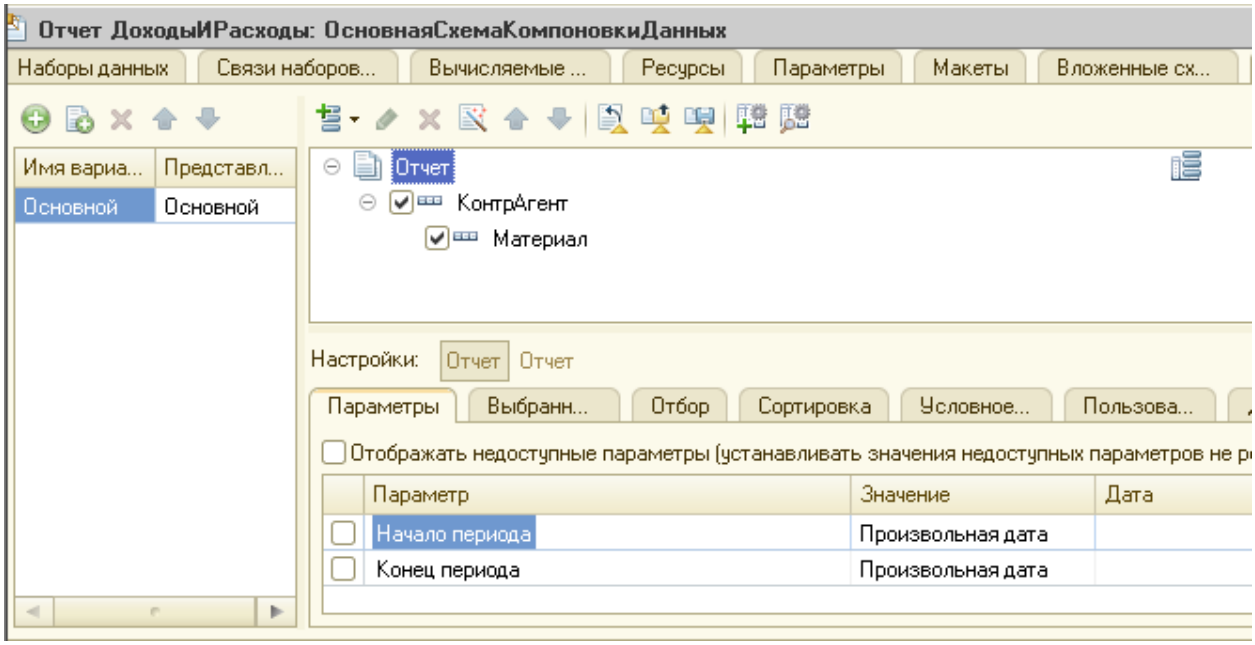


Рисунок 7.11. Структура отчета.

Осталось задать возможность для пользователя выбирать период. Правой клавишей на «Начало периода», и выбрать «Включить в пользовательский интерфейс» (рис.7.12).

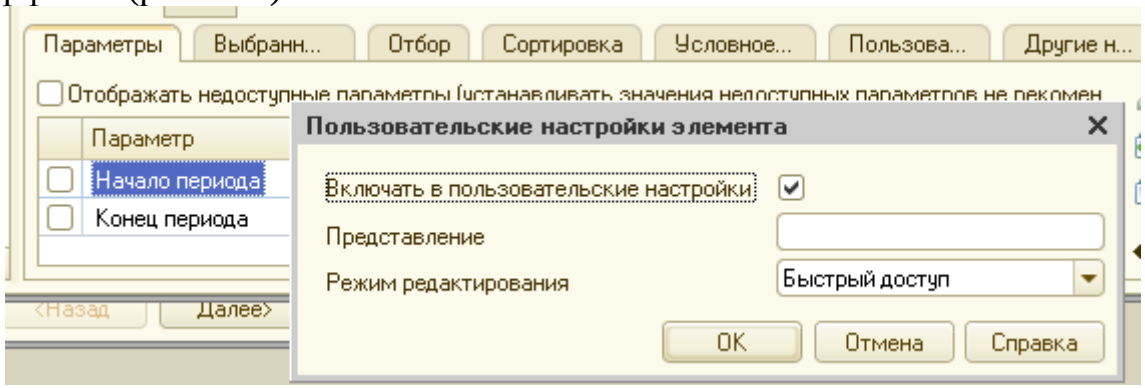


Рисунок 7.12. Изменение свойств поля.

Проверяем работу в режиме 1С: Предприятие (рис.7.13.-7.14).

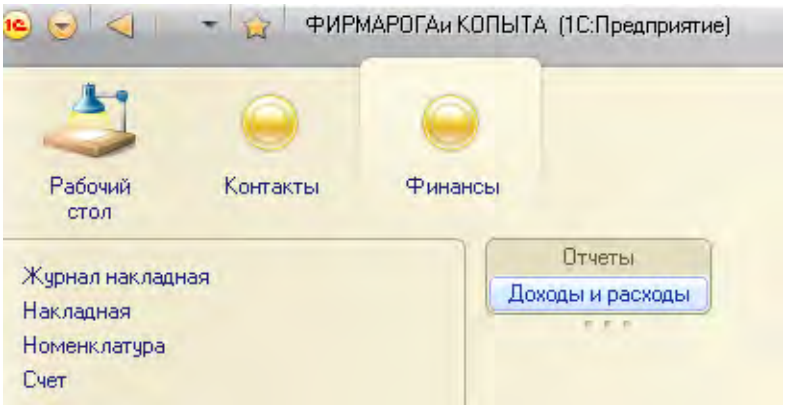
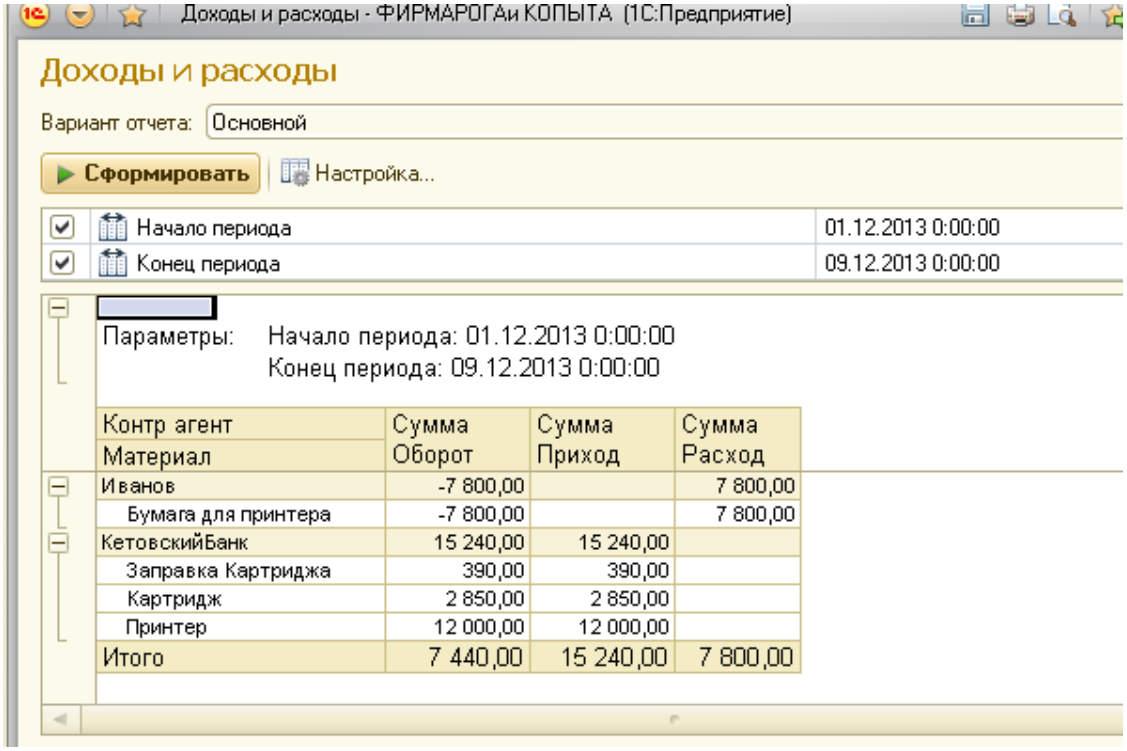


Рисунок 7.13. Запуск отчета.



Контр агент / Материал	Сумма Оборот	Сумма Приход	Сумма Расход
Иванов	-7 800,00		7 800,00
Бумага для принтера	-7 800,00		7 800,00
КетовскийБанк	15 240,00	15 240,00	
Заправка Картриджа	390,00	390,00	
Картридж	2 850,00	2 850,00	
Принтер	12 000,00	12 000,00	
Итого	7 440,00	15 240,00	7 800,00

Рисунок 7.14. Отчет сформирован.

В данной главе рассмотрен способ создания отчета в автоматическом режиме с использованием регистра накопления. В реальных конфигурациях, где задействованы множество разнообразных документов и справочников, создание отчетов, наглядно отображающих движение товарно-материальных ценностей и работу сотрудников, - основная работа сопровождающего программиста.

Вопросы для самоконтроля

1. Для чего предназначен объект конфигурации «Отчет»?
2. Как создать простейший отчет?
3. Как пользоваться конструктором отчетов?

Задание для самостоятельного выполнения

Создайте отчет по приходу расходу материалов.

8. Периодические регистры сведений

Часто необходимо сохранять изменяющиеся со временем данные. Например: цена какого либо изделия может изменяться во времени. Желательно хранить это изменение цены. В ранних версиях системы 1С для этого использовались периодические реквизиты, в версии 8 от периодических реквизитов отказались и заменили их на регистры сведений. Рассмотрим использование периодических регистров.

Регистр сведений предназначен для описания структуры хранения данных в разрезе нескольких измерений. Регистры и измерения были рассмотрены выше. Принципиальным отличием регистра сведений от регистра накопления является то, что каждое движение устанавливает новое значение ресурса. Таким образом, в нем можно хранить любые данные, а не только числовые. Регистр может хранить данные с привязкой к периоду времени и, следовательно, хранит историю изменения данных [1]. Добавим регистр (рис.8.1).

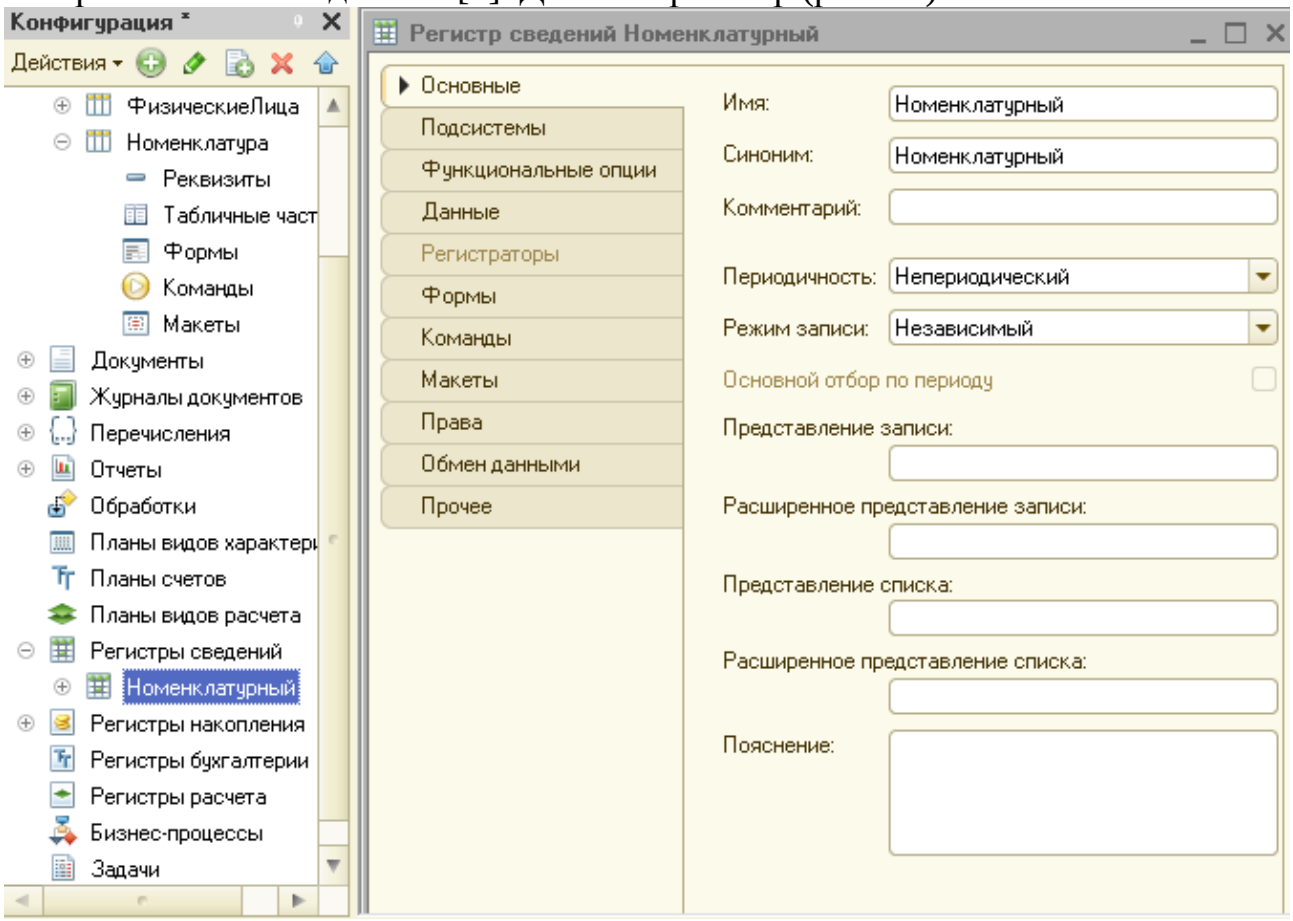


Рисунок 8.1. Создание периодического регистра сведений.

Задаем регистр периодическим и выбираем период, который может измеряться в секундах, в днях, в месяцах, кварталах, годах или задаваться регистратором. В данном случае выбираем в пределах секунды (рис.8.2). Оставим режим записи как Независимый, чтобы можно было записывать и изменять регистр без использования документа регистратора.

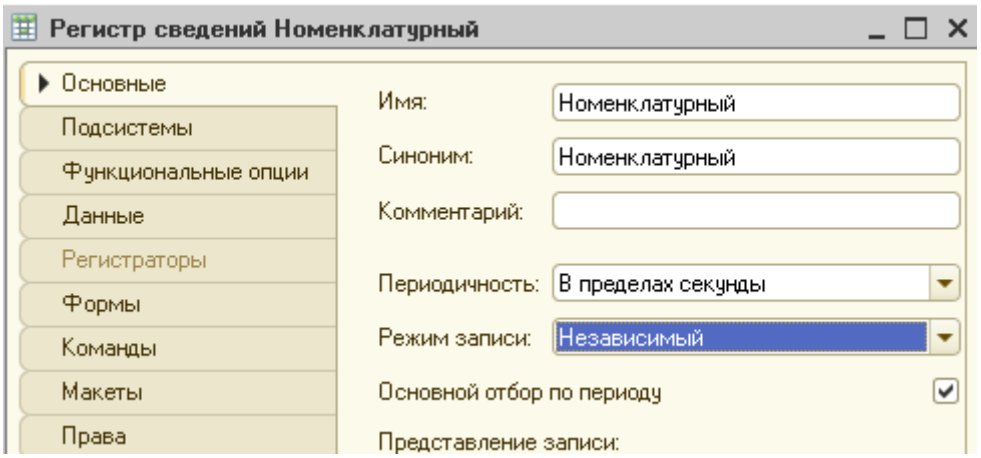


Рисунок 8.2. Периодичность регистра.

Переходим на вкладку данных и создаем измерение Номенклатура, ссылающееся на справочник, для которого мы и создаем регистр (рис.8.3-8.4).

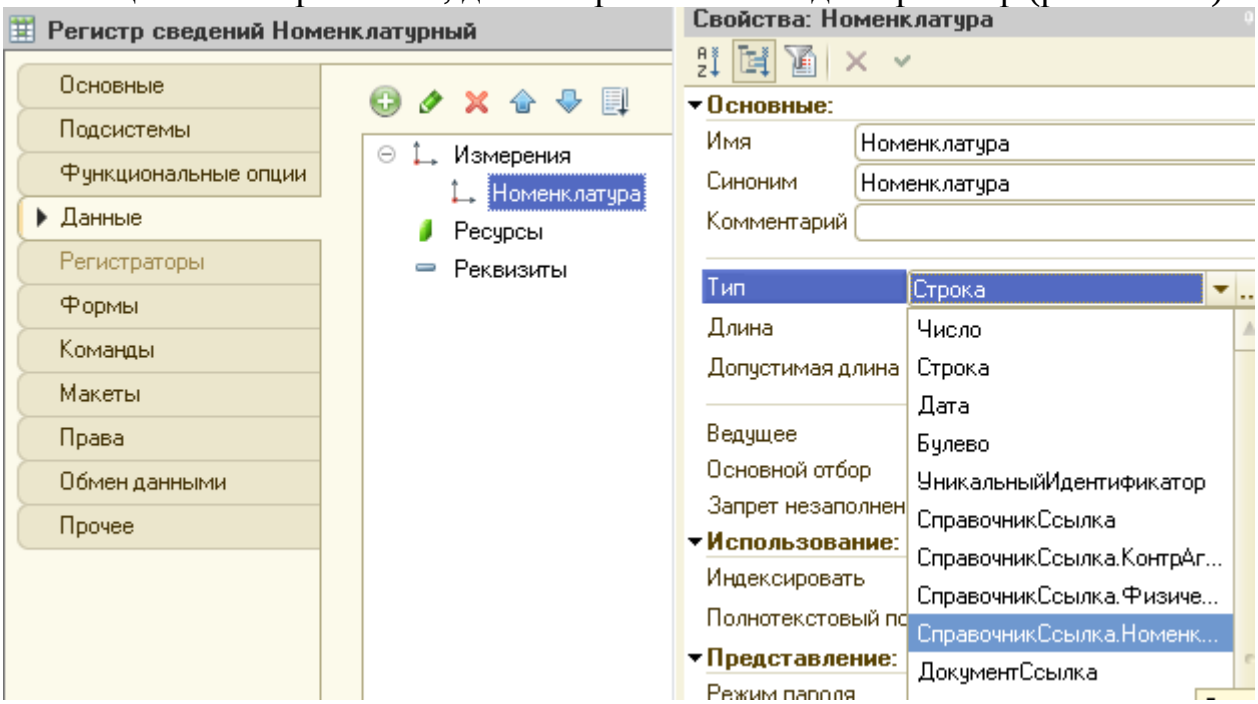


Рисунок 8.3. Задание измерения.

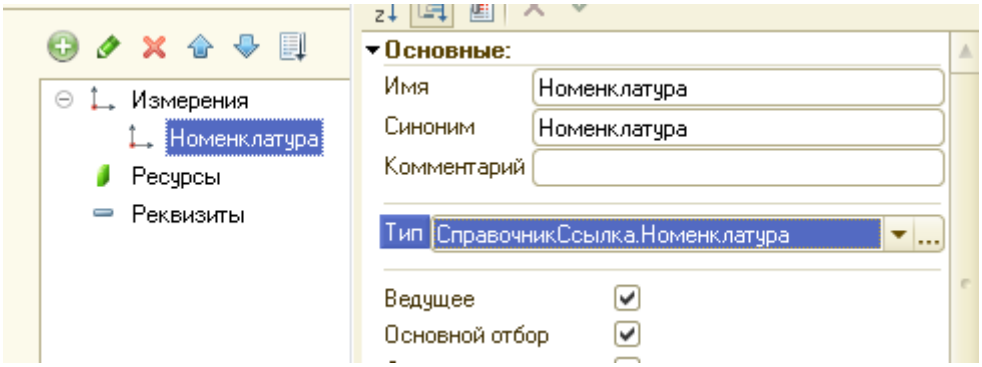


Рисунок 8.4. Задание свойства «Ведущее».

Отмечаем свойство «Ведущее». Это важное свойство говорит о том, что сведения регистра существуют пока существует объект, на который есть ссылка.

В нашем случае этим объектом является запись справочника «Номенклатура». При удалении записи автоматически удаляются и связанные с ней записи регистра. Задаем ресурсы (рис.8.5-8.6).

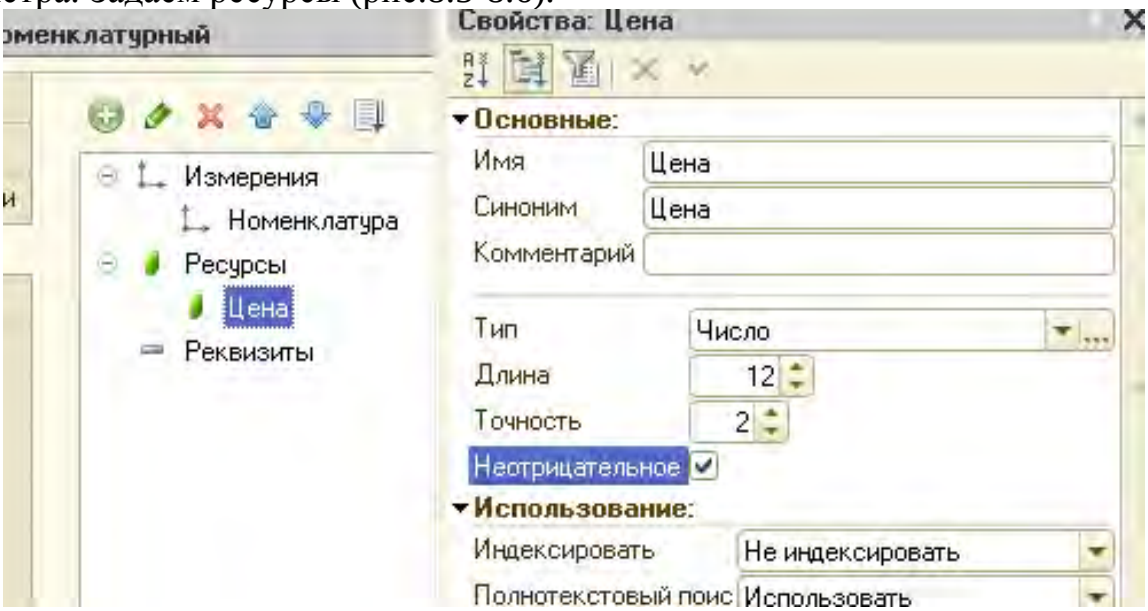


Рисунок 8.5. Задание ресурса «Цена».

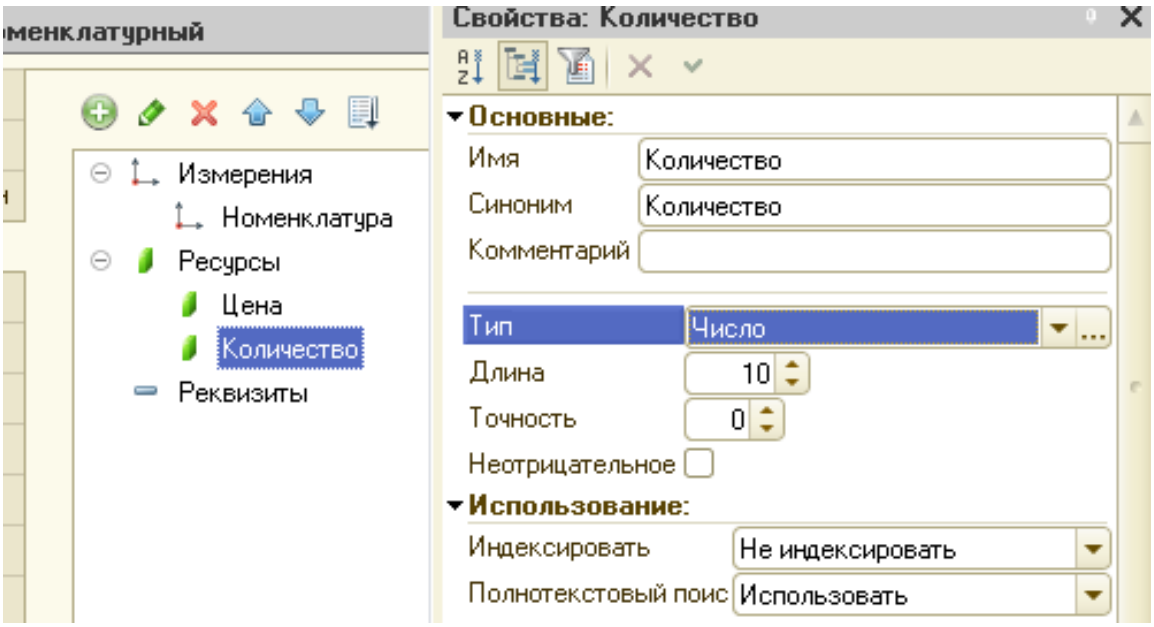


Рисунок 8.6. Задание ресурса «Количество».

В реальной жизни часто надо отслеживать и изменение количества товаров, например, на складе. В наших учебных целях используем два ресурса.

Посмотрим, как все это выглядит в режиме 1С: Предприятие. Заходим в справочник «Номенклатура», выбираем элемент. В открывшейся форме появилась возможность перейти к редактированию регистра «Номенклатурный» (рис.8.7.-8.9).

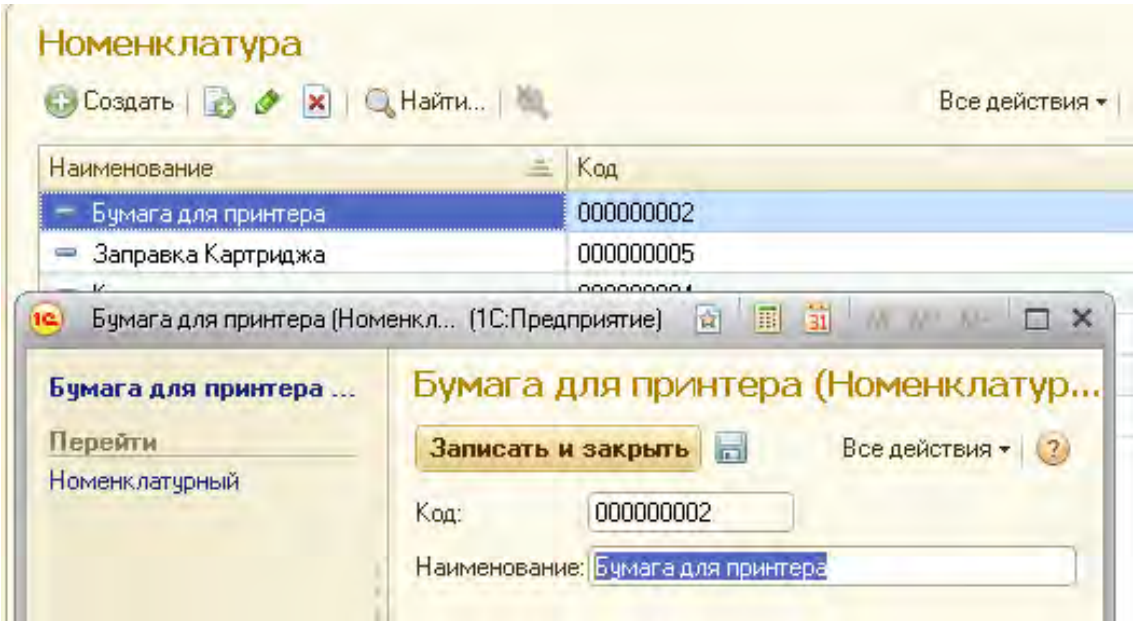


Рисунок 8.7. Переход к регистру «Номенклатурный».

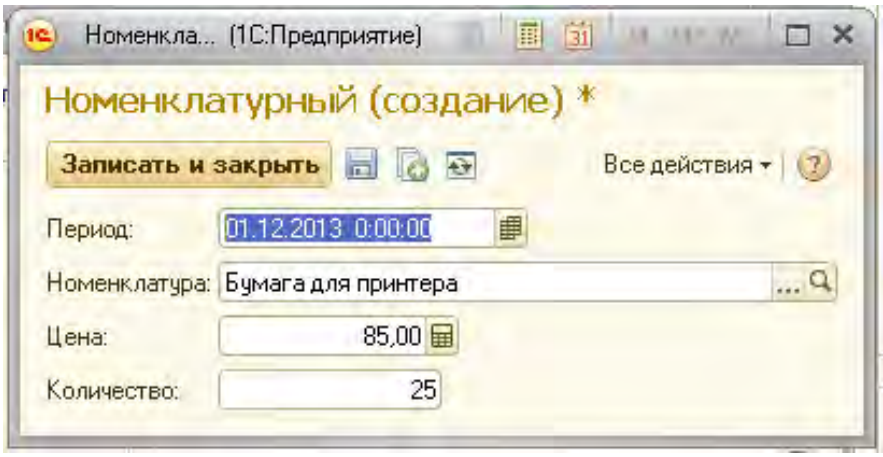


Рисунок 8.8. Создание записи в регистр.

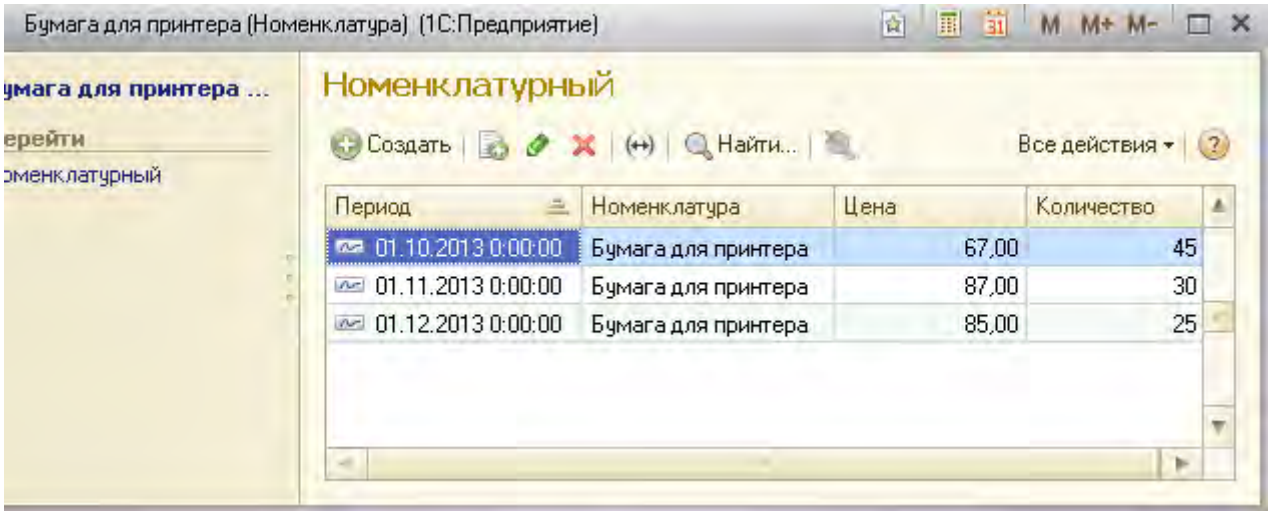


Рисунок 8.9. Создание нескольких записей в периодическом регистре.

Теперь попробуем использовать наш периодический регистр для автоматической подстановки цены в документы. При этом цена должна зависеть от даты создания документа.

Применим регистр для улучшения документа «Счет». Открываем форму документа и создаем обработчик для события «ПриИзменении» материала, аналогично тому, как это делалось в разделе 5 для изменения цены и количества (рис.8.10).

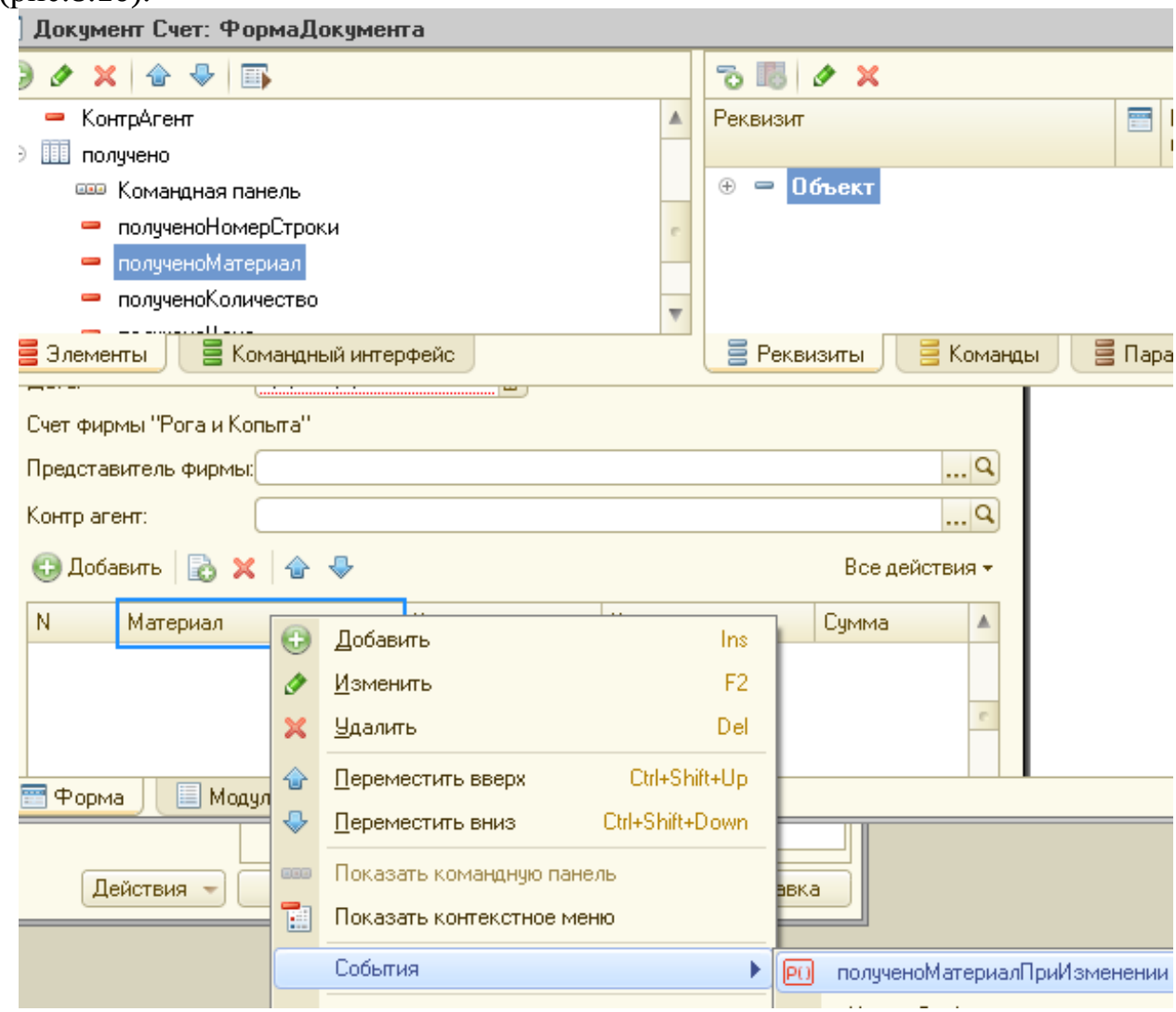


Рисунок 8.10. Создание обработчика.

В результате попадаем в знакомый нам модуль, где у нас уже есть два обработчика событий, теперь предстоит написать третий (рис.8.11-8.13).

Сначала напишем для него функцию, которая будет выполняться на сервере и получать значение ресурсов из периодического регистра. Назовем ее:

Функция ЦенаАктуальная (ДатаТек,ЭлементНоменклатуры)

Здесь параметры : ДатаТек - это текущая дата документа.

ЭлементНоменклатуры - это элемент, для которого ставится цена.

Очевидно, что это функция двух переменных, которая возвращает ресурсы из нашего регистра.

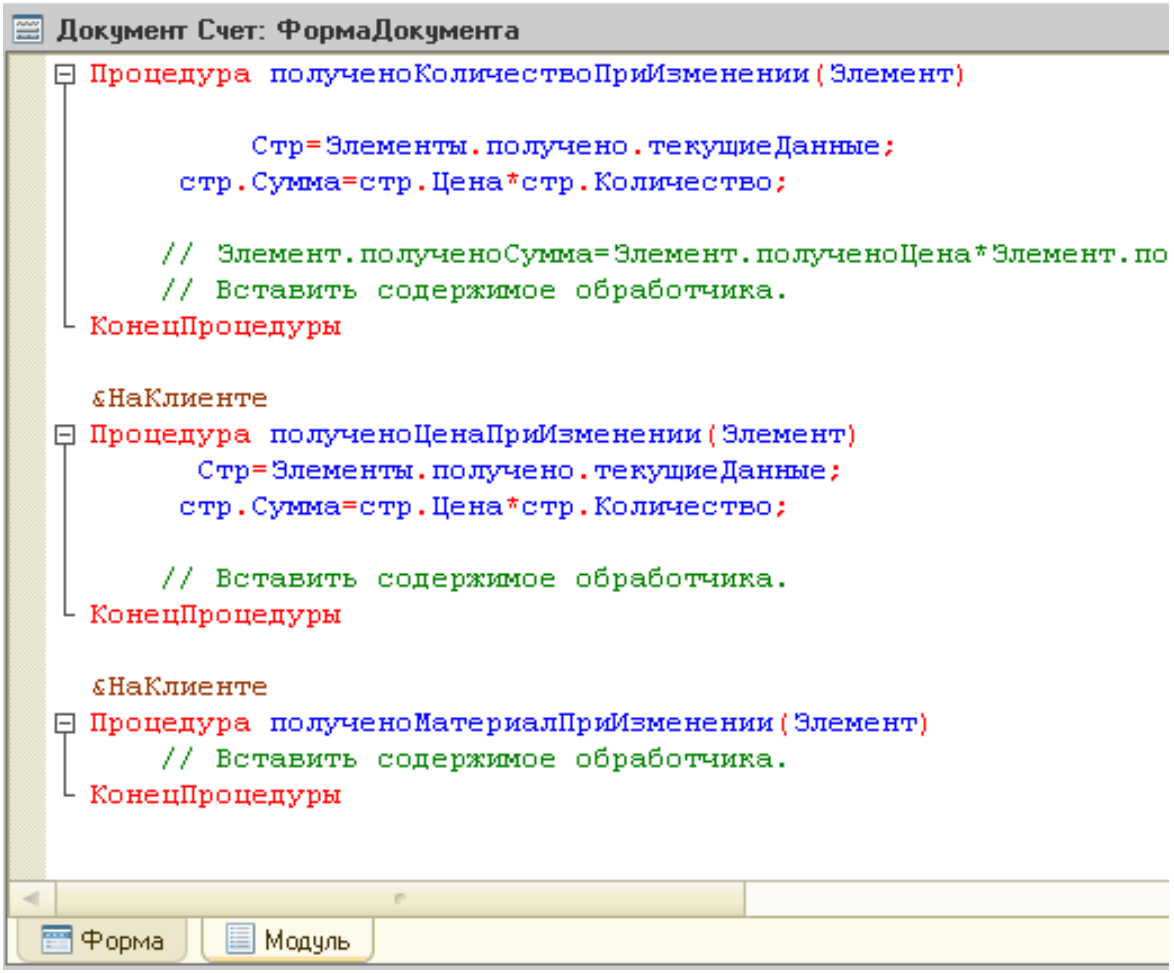


Рисунок 8.11. Модуль формы документа «Счет».

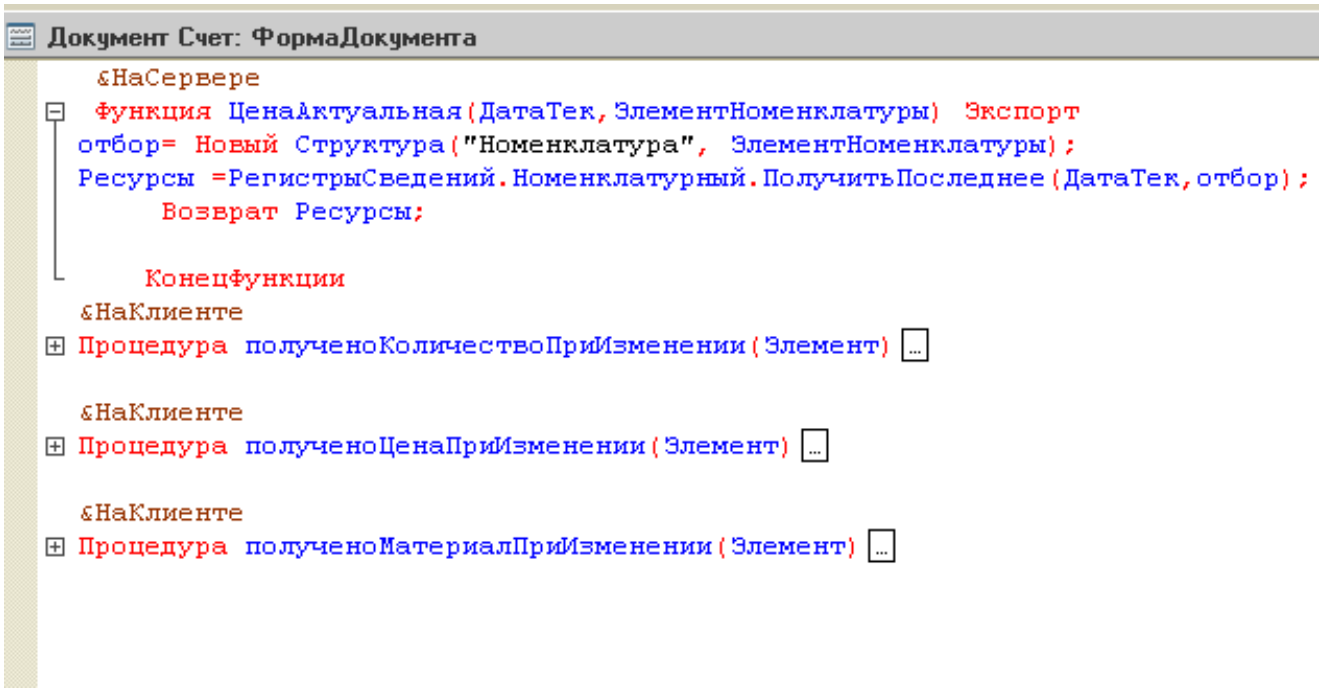


Рисунок 8.12. Функция для получения ресурсов из регистра.

Документ Счет: ФормаДокумента

```
    &НаСервере
  Функция ЦенаАктуальная(ДатаТек,ЭлементНоменклатуры)  Экспорт
 отбор= Новый Структура("Номенклатура",  ЭлементНоменклатуры);
 Ресурсы =РегистрыСведений.Номенклатурный.ПолучитьПоследнее(ДатаТек,отбор);
       Возврат Ресурсы;

     КонецФункции
  &НаКлиенте
 Процедура полученоКоличествоПриИзменении(Элемент) ...

  &НаКлиенте
 Процедура полученоЦенаПриИзменении(Элемент) ...

  &НаКлиенте
 Процедура полученоМатериалПриИзменении(Элемент)

        Стр=Элементы.получено.ТекущиеДанные;
           стр.Цена= ЦенаАктуальная(Объект.Дата,стр.Материал).цена;
         полученоЦенаПриИзменении(Элемент);
        // Вставить содержимое обработчика.
 КонецПроцедуры
```

Рисунок 8.13. Обработчик, вставляющий цену.

В режиме 1С: Предприятие (рис.8.14).

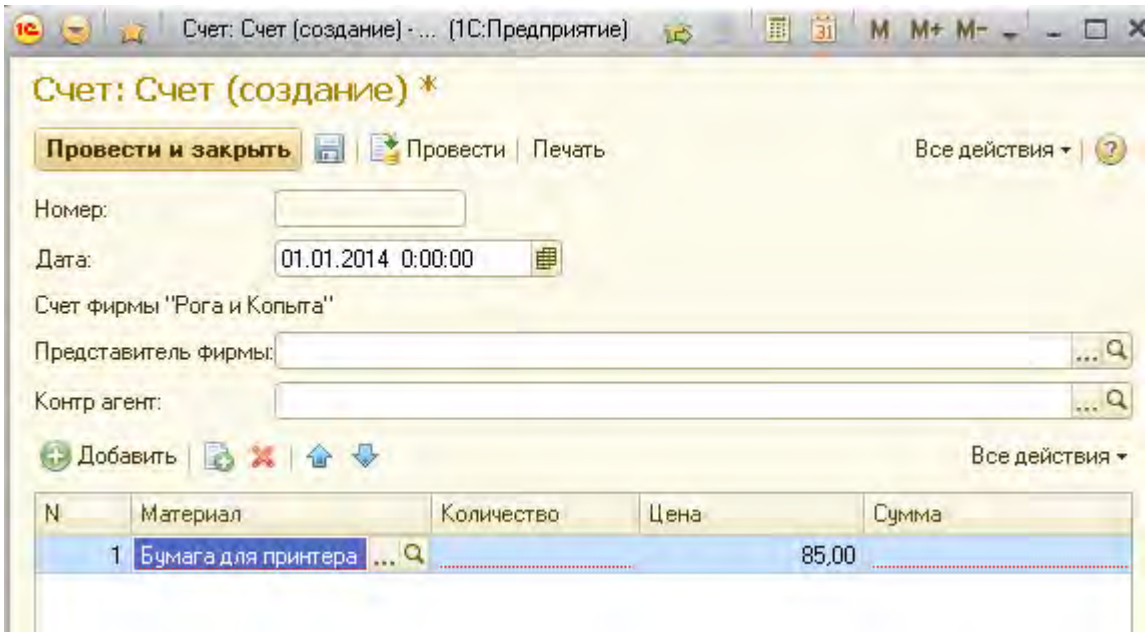


Рисунок 8.14. Формирование цены в зависимости от материала.

Добавим возможность актуальной цены при изменении даты документа (рис.8.15-8.16).

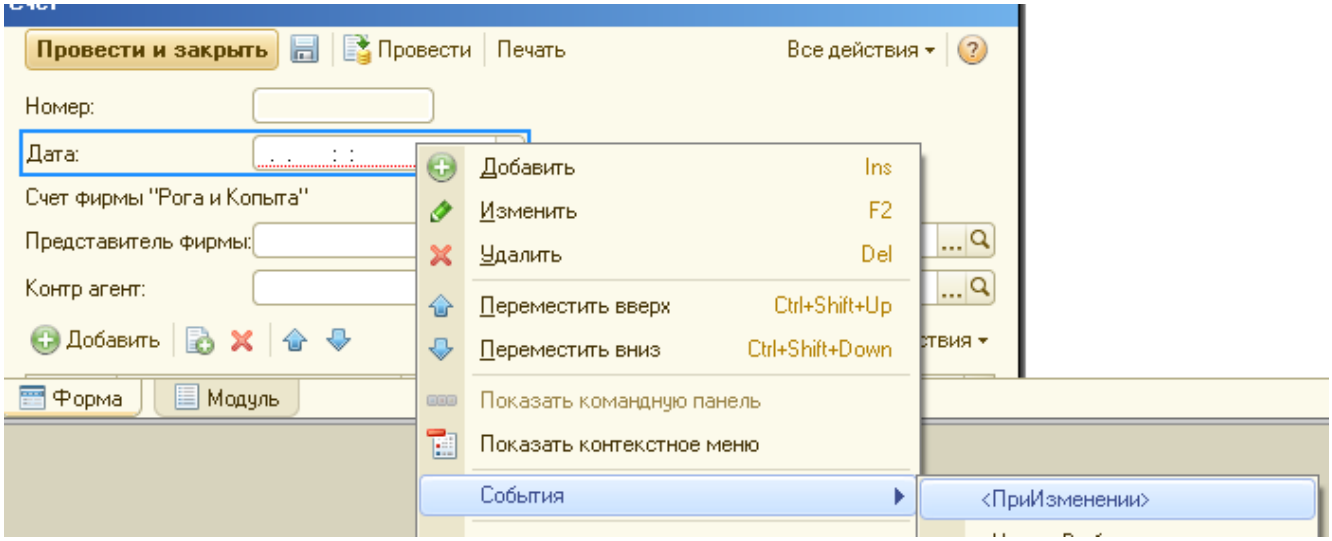


Рисунок 8.15. Обработчик для даты.

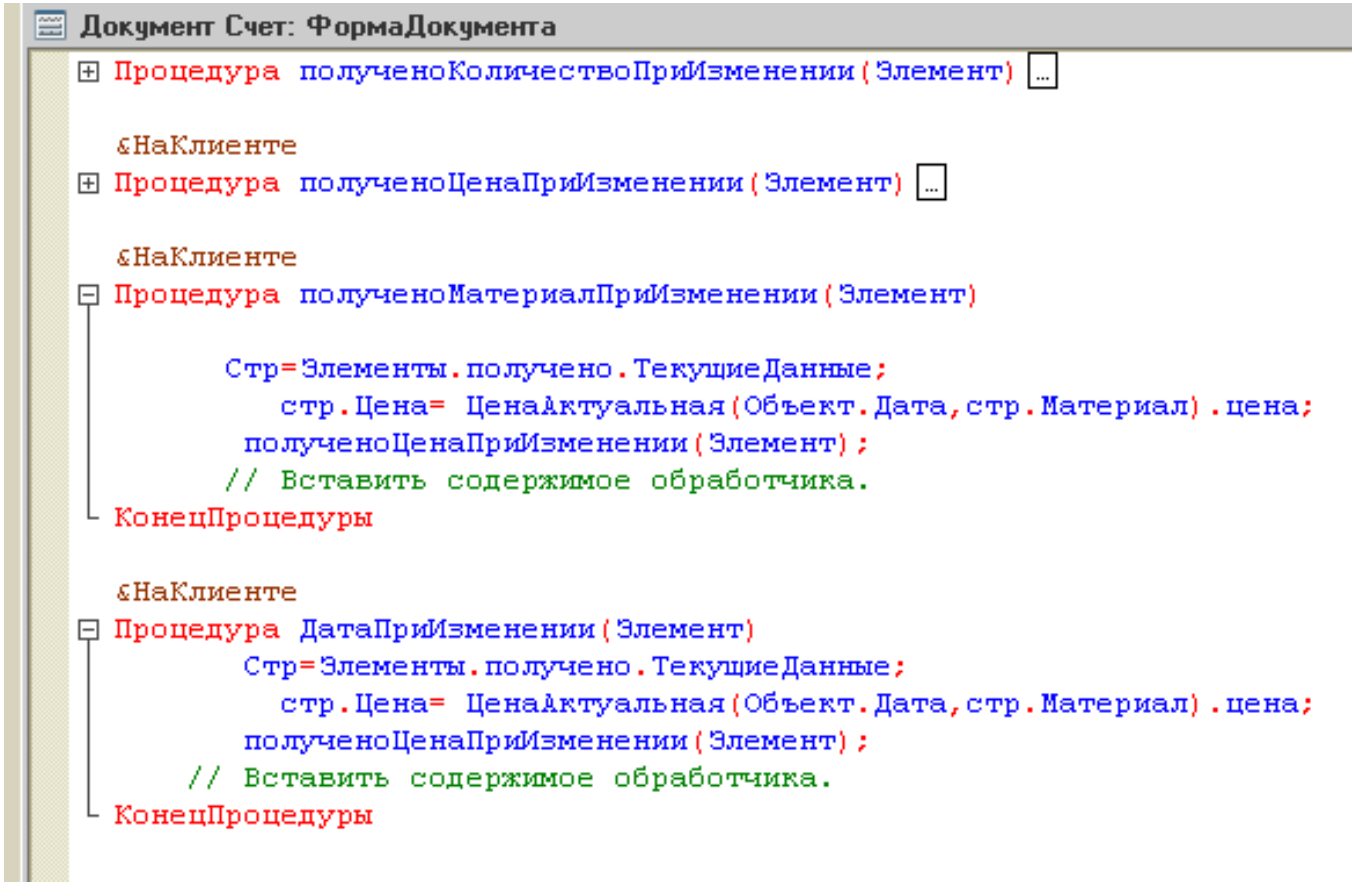


Рисунок 8.16. Модуль формы с обработчиком изменения даты.

Разработанный обработчик идентичен предыдущим (рис.8.17).

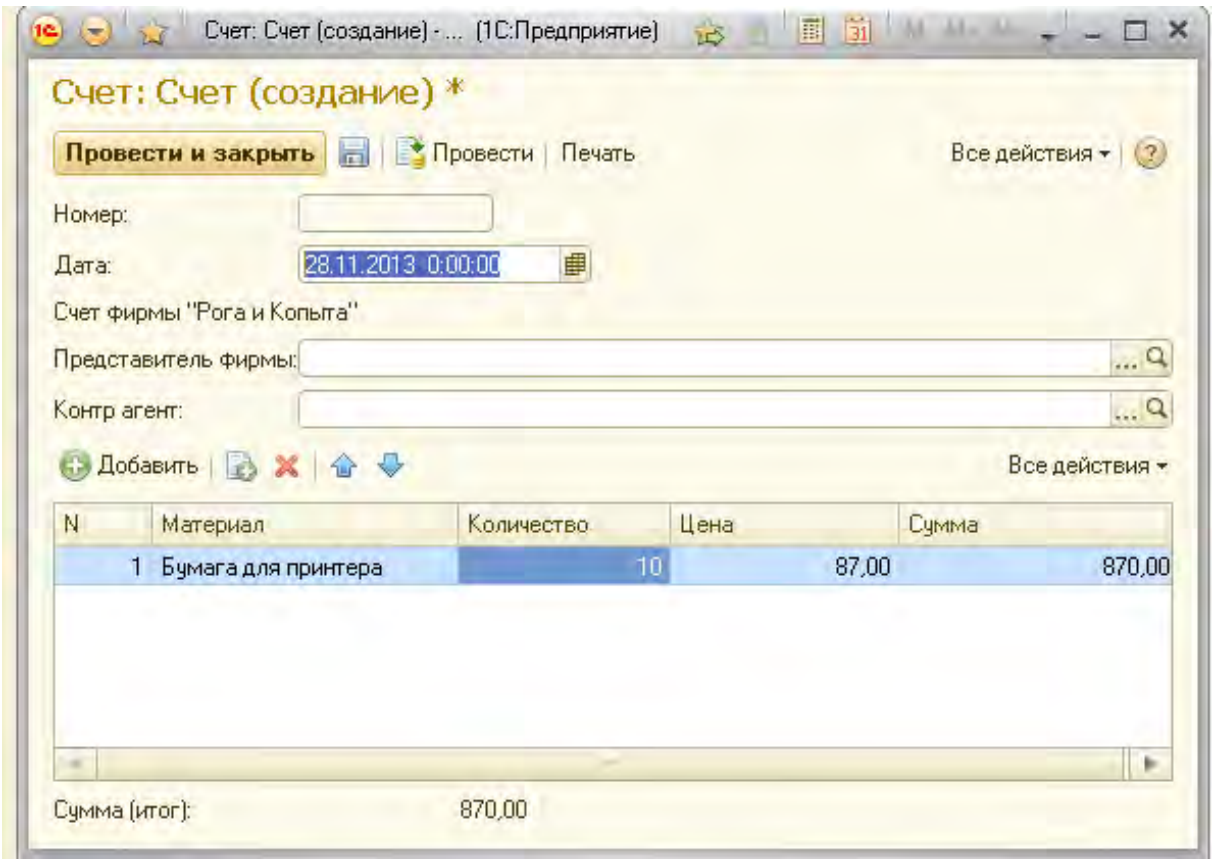


Рисунок 8.17. Результат работы обработчика.

В случае, если на складе нет такого количества материала, которое пытается указать оператор, то его следует предупредить. Следовательно, необходимо дописать в обработчик изменения количества соответствующий код (рис.8.18).

```
&НаКлиенте
Процедура полученоКоличествоПриИзменении(Элемент)

    Стр=Элементы.получено.текущиеДанные;
    СкладКол=ЦенаАктуальная(Объект.Дата,стр.Материал).количество;
    Если стр.Количество> СкладКол тогда
        Сообщить("На складе нет такого количества, там всего "+СкладКол);
        стр.Количество=СкладКол;
        конецЕсли;
         стр.Сумма=стр.Цена*стр.Количество;

    КонецПроцедуры
```

Рисунок 8.18. Текст обработчика ошибочной ситуации.

Для проверки обработчика откроем счет и попытаемся указать количество 100. Полученное сообщение показывает невозможность такого действия (8.19).

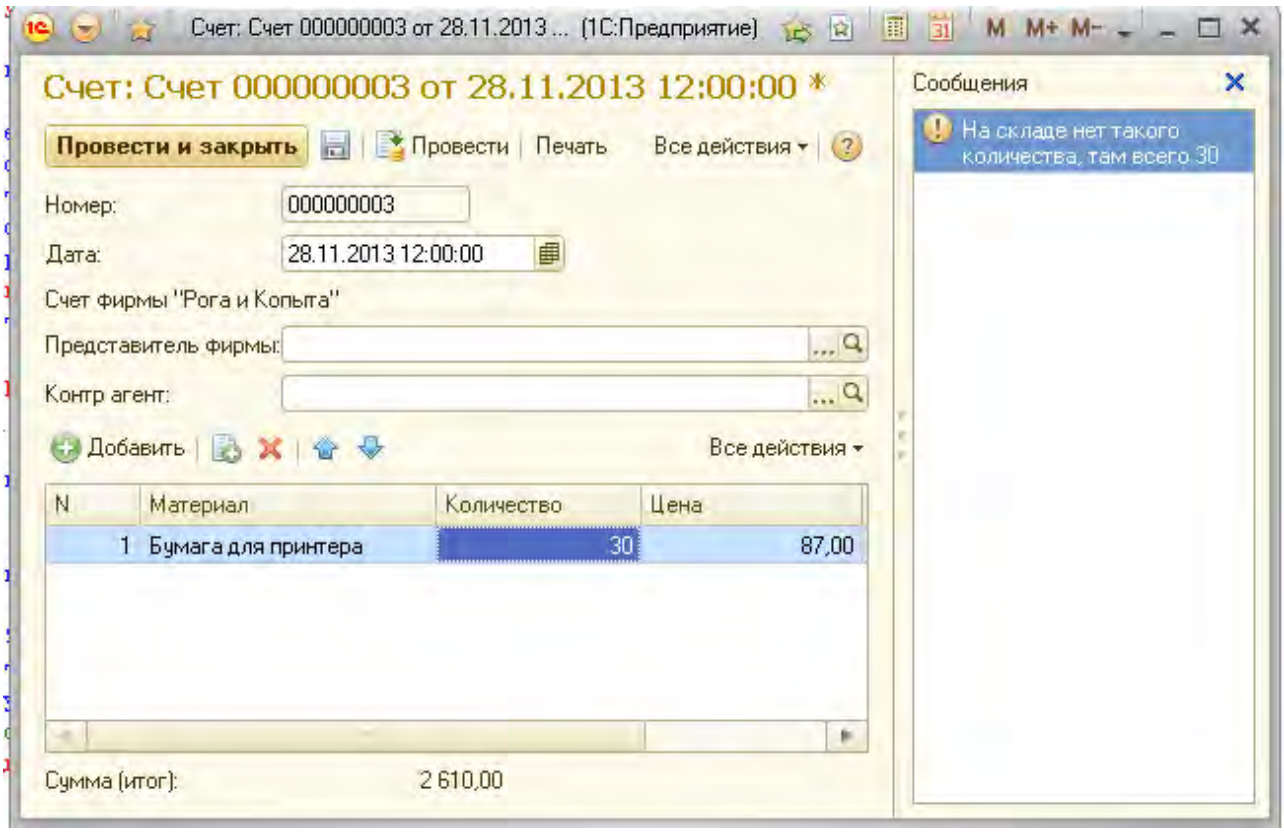


Рисунок 8.19. Результат работы обработчика.

Вопросы для самоконтроля

1. Что такое регистр сведений?
2. Чем регистр сведений отличается от регистра накопления?
3. Для чего предназначены регистры сведений?
4. Что значит периодический регистр сведений?
5. Как создать периодический регистр?
6. Что такое обработчик событий?
7. Где находится модуль формы?
8. Как программно получить данные из периодического регистра?

Список рекомендуемых источников

1. Профессиональная разработка в среде 1С Предприятие 8. / под ред. Радченко М.Г. - « 1С-Паблибшинг», СПб.: Питер, 2006.
2. http://v8.1c.ru/predpriyatie/QuestionsPlatform.htm?printversion=1
3. Рыбалко В.В. HELLO-1С. Пример быстрой разработки приложений на платформе 1С:Предприятие 8.2. М.: ООО « 1С-Паблибшинг», 2009.
4. Радченко М.Г. 1С:Предприятие 8.2 практическое пособие разработчика. Примеры и типовые решения / М.Г. Радченко, Е.Ю. Хрусталева - М.: ООО « 1С-Паблибшинг», 2009.
5. 1С:Предприятие 8.2 Руководство разработчика. Ч.1, Ч.2. Москва. Фирма «1С». 2009.
6. http://1c-uroki.ru/

Приложение А

Список лабораторных работ

Лабораторная работа №1. Знакомство с платформой 1С: Предприятие.
Цель работы: изучить способы создания баз данных, состав конфигурации и задание имен в интерфейсе, связь между базой на диске и в интерфейсе конфигурации.
Продолжительность 2 час.

Лабораторная работа № 2. Создание объектов конфигурации.
Цель работы: изучить способы создания объектов конфигурации, научиться создавать подсистемы, справочники, реквизиты объектов, формы справочников, взаимодействие реквизитов.
Продолжительность 4 часа.

Лабораторная работа № 3. Документы как объекты конфигурации.
Цель работы: научиться работать с документами - создавать, изменять их, добавлять реквизиты, формы, макеты печатных форм.
Продолжительность 4 часа.

Лабораторная работа № 4. Регистрация движения документов
Цель работы: изучить объекты конфигурации регистры, журналы, научиться создавать их и вносить исправления.
Продолжительность 4 часа.

Лабораторная работа № 5. Отчеты.
Цель работы: изучить объекты конфигурации отчеты, научиться создавать запросы, формы и макеты печатных форм.
Продолжительность 4 часа.

Лабораторная работа № 6. Периодические регистры сведений
Цель работы: изучить объекты конфигурации периодические регистры, научиться их создавать и использовать для учета изменяющейся во времени информации.
Продолжительность 4 часа.

Приложение Б

Примерные темы курсового проекта

Курсовая работа является самостоятельной работой студента призванная продемонстрировать навыки конфигурирования и программирования полученные ходе освоения курса.

Студент должен продемонстрировать умение создавать объекты конфигурации:

- Подсистемы.
- Справочники.
- Перечисления.
- Регистры.
- Документы.
- Отчеты.

Курсовая работа должна содержать:

1. Созданные студентом объекты конфигурации в виде программных объектов.
2. Пояснительную записку, оформленную в соответствии с ГОСТ

Пояснительная записка должна обязательно содержать:

1. Техническое задание, т.е. конкретизация выбранной темы.
2. Общую блок-схему алгоритма.
3. Описание работы программного продукта с приведением необходимого кода, форм, структуры файлов и т.д..
4. Инструкцию программисту, содержащую всю необходимую информацию об ограничениях и выявленных при тестировании ошибочных ситуациях. Кроме того, данная инструкция содержит информацию о возможности модернизации программного продукта.
5. Инструкцию оператору, содержащую порядок действий оператора при работе с программным продуктом, в том числе в случае возникновения нештатных ситуаций.
6. Инструкцию по проверке и тестированию программного обеспечения с тестовыми данными.
7. Выводы по проделанной работе.
8. Список используемой литературы.

Примерный перечень тем курсовой работы:

1. Решение задачи сбора первичной информации о деятельности предприятия, на основе счетов-фактур.
2. Создание базы данных контрагентов предприятия.
3. Создание базы данных о сотрудниках предприятия.
4. Исследование задачи транспортно-складской операции в автоматизированном производстве.
5. Решение задачи оптимизации технологических процессов в автоматизированном машиностроительном производстве.
6. Нахождение оптимального решения в реальном производственном процессе.
7. Решение задачи анализа деятельности предприятия.
8. Автоматизация анализа текстовых документов .
9. Задача организации сбора, хранения и поиска информации на предприятии.

Учебное издание

Скобелев Игорь Вадимович

Грибанов Константин Николаевич

Фалев Вячеслав Владимирович

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ НА ПЛАТФОРМЕ 1С: ПРЕДПРИЯТИЕ 8.2

Учебное пособие

В авторской редакции

Подписано к печати 26.03.14 Формат 60x84 1/16 Бумага тип. № 1
Печать цифровая Усл.печ.л. 6,69 Уч.изд.л. 6,69
Заказ 90 Тираж 100

Редакционно-издательский центр КГУ.
640669, г. Курган, ул. Гоголя, 25.
Курганский государственный университет.