БИБЛИОТЕКА РАЗРАБОТЧИКА

М.Г. Радченко
Е.Ю. Хрусталева

1С:ПРЕДПРИЯТИЕ 8.2
ПРАКТИЧЕСКОЕ ПОСОБИЕ РАЗРАБОТЧИКА

+CD

ПРИМЕРЫ
И ТИПОВЫЕ ПРИЕМЫ

1С ПАБЛИШИНГ

БИБЛИОТЕКА РАЗРАБОТЧИКА

М.Г. Радченко
Е.Ю. Хрусталева

1С:ПРЕДПРИЯТИЕ 8.2

ПРАКТИЧЕСКОЕ ПОСОБИЕ РАЗРАБОТЧИКА

+CD

ПРИМЕРЫ
И ТИПОВЫЕ ПРИЕМЫ

1С ПАБЛИШИНГ

М.Г. Радченко, Е.Ю. Хрусталева

1С:Предприятие 8.2. Практическое пособие разработчика
Примеры и типовые приемы

М.Г. Радченко, Е.Ю. Хрусталева

1С:Предприятие 8.2. Практическое пособие разработчика
Примеры и типовые приемы

Электронная книга в формате ePub; ISBN 978-5-9677-1933-2.

Версия издания от 16.09.2013.

*Электронный аналог издания "1С:Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые приемы "
(ISBN978-5-9677-1147-3, М.: ООО "1С-Паблишинг", 2009;
артикул печатной книги по прайс-листу фирмы "1С": 4601546069627;
по вопросам приобретения печатных изданий издательства "1С-Паблишинг" обращайтесь к партнеру "1С",
обслуживающему вашу организацию, или к другим партнерам фирмы "1С", в магазины "1С Интерес", а также в книжные и интернет-магазины).*

Книга представляет собой пособие, позволяющее быстро освоить приемы разработки и модификации прикладных решений на платформе 1С:Предприятие 8.2.
На примере создания реального прикладного решения показана структура различных объектов системы, их назначение и методика использования. Приведены процедуры на встроенном языке, в том числе с применением языка запросов, которые снабжены подробными комментариями.
Книга может быть использована и как практическое руководство, и как справочное пособие. Рассматриваемое в книге прикладное решение учитывает накопленный опыт разработки в системе 1С:Предприятие 8 и демонстрирует многие новые возможности и механизмы, предоставляемые версией 8.2.
Материал рассчитан на начинающих разработчиков, не знакомых с системой 1С:Предприятие 8.

Дополнительные материалы

Приложение к книге включает демонстрационные конфигурации, иллюстрирующие примеры, рассматриваемые в книге. Таким образом, можно самостоятельно воспроизвести или доработать любой пример из книги, используя имеющиеся готовые решения. Демонстрационные конфигурации можно установить на коммерческую или учебную версию 1С:Предприятия.

Скачайте материалы и учебную версию на странице http://its.1c.ru/book_demo/,

раскройте архив и следуйте инструкциям по установке.

Интернет-конференция для начинающих
разработчиков http://devtrainingforum.v8.1c.ru/forum

Фирма "1С"
123056, Москва, а/я 64, Селезневская ул., 21.
Тел.: (495) 737-92-57, факс: (495) 681-44-07.
1c@1c.ru, http://www.1c.ru/
Издательство ООО "1С-Паблишинг"
127473, Москва, ул. Достоевского, 21/1, строение 1.
Тел.: (495) 681-02-21, факс: (495) 681-44-07.
publishing@1c.ru, http://books.1c.ru/

Предисловие

Издание этой книги подготовлено специально к выпуску финальной версии платформы «1С:Предприятие 8.2».

«1С:Предприятие 8.2» – это принципиальное изменение архитектуры платформы версии 8, наиболее существенное с момента ее выпуска.

Книга, которую вы держите в руках, обладает двумя значительными преимуществами.

С одной стороны, в ней показаны все основные моменты, которые важны для разработки в новой версии платформы «1С:Предприятие 8.2»: конструирование управляемого интерфейса, разработка управляемых форм, использование вариантов отчетов и новых возможностей их настройки, новая методика проведения документов, настройка рабочего стола, командного интерфейса, использование функциональных опций и др.

С другой стороны, книга дополнена и переработана с расчетом на специалистов, не знакомых с системой «1С:Предприятие 8». При этом использовались вопросы, мнения, пожелания, которые высказывали читатели этой книги в интернет-конференции http://devtrainingforum.v8.1c.ru.

Например, книга дополнена специальным разделом, подробно описывающим методику использования синтакс-помощника и различные способы использования отладчика для анализа имеющегося кода или написания собственного. Эта информация, безусловно, поможет начинающим разработчикам в самостоятельном освоении всего многообразия встроенного языка «1С:Предприятия 8.2».

Также в книге показаны примеры установки ограничений прав доступа на уровне записей и полей базы данных. В предыдущих изданиях книги таких примеров не было.

Предыдущие издания этой книги хорошо зарекомендовали себя в среде разработчиков «1С:Предприятия 8». Благодаря этой книге десятки тысяч разработчиков смогли освоить версии 8.0 и 8.1 платформы «1С:Предприятия». Общий тираж предыдущих изданий составил более 60 000 экземпляров.

Авторы надеются, что книга будет полезна каждому, кто хочет научиться разрабатывать прикладные решения на новой платформе «1С:Предприятия 8.2».

Если же в процессе выполнения примеров вы столкнетесь с трудностями, добро пожаловать в интернет-конференцию http://devtrainingforum.v8.1c.ru. Авторы книги и другие читатели обязательно помогут вам разобраться со всеми

непонятными вопросами.

Максим Радченко, Елена Хрусталева

Кому предназначена эта книга

В основу книги положен реальный пример разработки прикладного решения для небольшой фирмы, оказывающей бытовые услуги. По мере изучения этой книги вы научитесь основным приемам разработки в системе «1С:Предприятие 8», освоите различные области автоматизации хозяйственной деятельности, включая бухгалтерский учет, расчет зарплаты и т. д.

Почему был выбран именно такой пример?

С одной стороны, область оказания услуг хорошо знакома большинству из нас. Так или иначе, но с разнообразными услугами мы сталкиваемся постоянно. Это ремонт различной бытовой техники, обслуживание автомобиля, стирка и химчистка, парикмахерские и косметические услуги и многое другое.

С другой стороны, деятельность ремонтной фирмы хорошо подходит для демонстрации возможностей «1С:Предприятия 8». Здесь есть разнообразные услуги, оказываемые клиентам, снабжение фирмы необходимыми материалами и их расход при оказании услуг. Работа такого предприятия позволяет

рассмотреть учет персонала и расчет заработной платы сотрудников. Есть возможность проиллюстрировать ведение бухгалтерского учета. Это разнообразие видов деятельности позволяет довольно широко показать возможности формирования различных отчетов и итоговых данных на основе имеющейся информации.

Книга обращена в большей степени к начинающим разработчикам, делающим первые шаги в разработке прикладных решений. Пояснения, приведенные в книге, подробны и доступны даже для тех, кто лишь отдаленно знаком с азами программирования.

Если вы только начинаете работу с «1С:Предприятием» или даже совсем не знакомы с этой системой, но очень хотите научиться, то эта книга – для вас. Цель книги – «провести вас за руку» по основным этапам разработки простого прикладного решения в системе «1С:Предприятие 8» и показать, что нет ничего недоступного для человека с интеллектом.

Более опытным разработчикам эта книга также будет полезна и позволит вспомнить или подробнее изучить отдельные моменты разработки.

Как читать

Эта книга максимально приближена к учебному пособию и построена в виде

отдельных занятий. В начале каждого занятия дается примерный хронометраж, чтобы вы представляли, сколько времени в среднем необходимо потратить на это занятие.

В конце занятия приводится список контрольных вопросов, позволяющий читателю оценить, насколько он усвоил данное занятие.

В начале книги содержится краткое оглавление по занятиям с указанием продолжительности каждого занятия. В самих занятиях находится более подробное оглавление, которое поможет вам быстро переходить к отдельным фрагментам прошлых занятий. В конце книги находится полное подробное оглавление всех занятий.

Каждое занятие является логически законченной частью разработки прикладного решения. Поэтому, хотя занятия различаются по своей продолжительности, настоятельно рекомендуется выполнять их целиком, от начала до конца. Иначе вам как начинающему разработчику будет сложно восстановить ход своих действий с середины занятия.

Занятия построены по принципу от простого к сложному. Они последовательно описывают основные приемы и охватывают различные области разработки в системе «1С:Предприятие 8».

Книга содержит большое количество рисунков и примеров кода на встроенном языке, снабженных подробными комментариями. Если вам они покажутся лишними или слишком подробными, можно их пропустить.

Занятия имеют теоретические вставки, которые можно читать сразу по ходу занятия, а можно оставить на потом. В любом случае на выполнение примера разработки, рассмотренного в книге, это не повлияет.

На специальном теоретическом занятии № 5 подробно рассматриваются примеры работы с отладчиком и синтакс-помощником. Это поможет вам в дальнейшем самостоятельно осваивать встроенный язык и разбираться с ошибками, допущенными в ходе выполнения заданий.

Конфигурация, которая создается на протяжении всей книги, содержится в демонстрационной информационной базе, прилагающейся к книге на компакт-диске. К ней можно обращаться в тех случаях, когда необходимо проверить правильность самостоятельного выполнения примеров из книги.

Поскольку пример, разбираемый в книге, довольно большой, на диске содержится не одна, а четыре информационные базы. По состоянию после выполнения 8, 13, 20 и 27-го занятия. Это поможет быстрее находить нужные фрагменты конфигурации.

Благодарности

Авторы выражают признательность Корсаковой Веронике, которая стала первым читателем этой книги и протестировала ее материалы с точки зрения человека, не имеющего опыта программирования.

Благодаря ее советам и замечаниям книга стала более понятной и доступной для неопытных читателей.

Что находится на компакт-диске

К книге прилагается компакт-диск, который содержит материалы, предназначенные для самостоятельного изучения и использования.

Прежде всего, это четыре варианта демонстрационной конфигурации, иллюстрирующие состояние разрабатываемого прикладного решения на момент прочтения соответствующей главы.

Все демонстрационные конфигурации содержатся на компакт-диске в виде одного дистрибутива. После запуска исполняемого файла шаблоны конфигураций устанавливаются в текущий каталог шаблонов. Конфигурации созданы в версии 8.2.9.260 «1С:Предприятия».

Конфигурации не являются законченными (с предметной точки зрения) прикладными решениями и не предназначены для реального ведения учета. Они являются исключительно набором примеров, поясняющих текст книги.

Также компакт-диск содержит все фрагменты листингов, приведенных в книге. Использование этих фрагментов может быть полезным как при чтении книги, так и в дальнейшей работе. Поэтому фрагменты оформлены в виде файла шаблонов текста «1С:Предприятия 8» – *Example.st*.

Этот файл вы можете подключить к любой конфигурации с помощью команды *Сервис* > *Шаблоны текста* > *Действия* > *Настройка шаблонов* > *Добавить* (рис. 0.1).

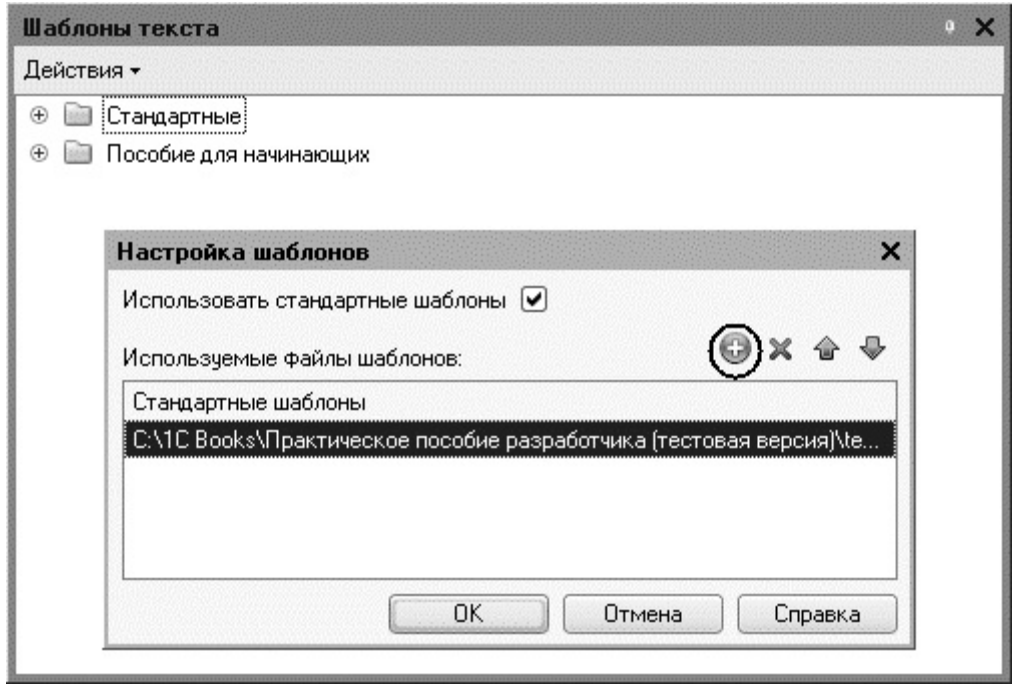


Рис. 0.1. Подключение шаблонов

Фрагменты кода сгруппированы по занятиям (рис. 0.2).

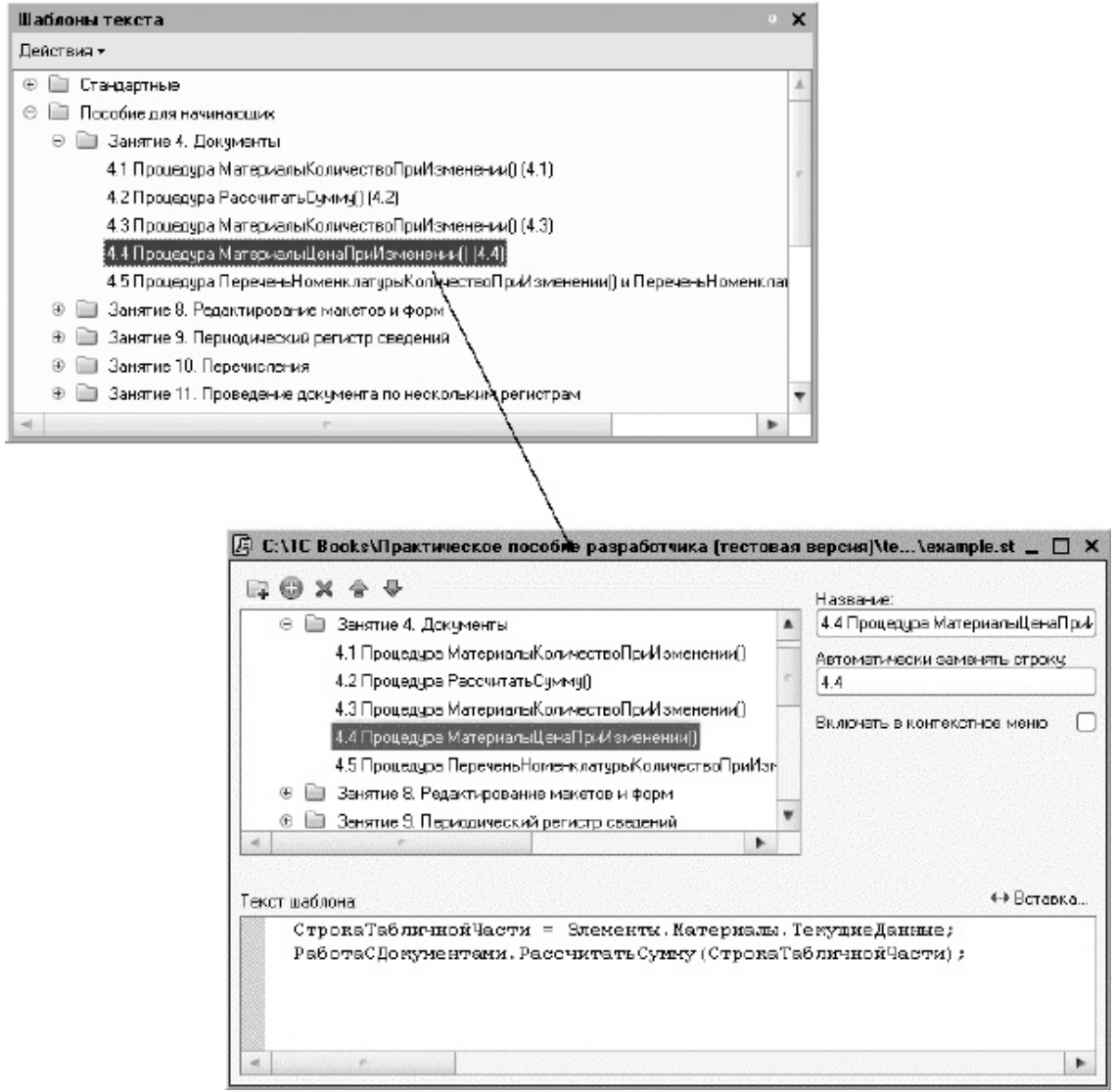

Шаблоны текста
Действия
Стандартные
Пособие для начинающих
Занятие 4. Документы
4.1 Процедура МатериалыКоличествоПриИзменении() (4.1)
4.2 Процедура РассчитатьСумму() (4.2)
4.3 Процедура МатериалыКоличествоПриИзменении() (4.3)
4.4 Процедура МатериалыЦенаПриИзменении() (4.4)
4.5 Процедура ПеречеьНоменклатурыКоличествоПриИзменении() и ПеречеьНоменклат
Занятие 8. Редактирование макетов и форм
Занятие 9. Периодический регистр сведений
Занятие 10. Перечисления
Занятие 11. Проведение документа по нескольким регистрам
C:\1C Books\Практическое пособие разработчика (тестовая версия)\te...\example.st
Занятие 4. Документы
4.1 Процедура МатериалыКоличествоПриИзменении()
4.2 Процедура РассчитатьСумму()
4.3 Процедура МатериалыКоличествоПриИзменении()
4.4 Процедура МатериалыЦенаПриИзменении()
4.5 Процедура ПеречеьНоменклатурыКоличествоПриИзм
Занятие 8. Редактирование макетов и форм
Занятие 9. Периодический регистр сведений
Название:
4.4 Процедура МатериалыЦенаПри4
Автоматически заменять строку:
4.4
Включать в контекстное меню
Текст шаблона
Вставка...
СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

Рис. 0.2. Шаблоны тестов листингов

Для каждого фрагмента кода в качестве строковой последовательности, которая будет заменяться при вводе текста, указывается номер листинга, содержащего соответствующий фрагмент кода. Замена строковой последовательности может производиться автоматически, если установлен режим автозамены, или вручную, с помощью комбинации клавиш *Ctrl* + *Q*. Режим автозамены устанавливается командой *Сервис* > *Параметры* > *Тексты модулей* > *Автозамена*. Также любой шаблон текста может быть просто перенесен мышью в произвольное место модуля.

Также на диске в папке *Image* находятся файлы картинок, использующиеся при создании подсистем конфигурации.

Ограничения учебной версии платформы

Эта книга может продаваться отдельно или входить в состав продукта «1С:Предприятие 8.2. Версия для обучения программированию».

К книге, продаваемой отдельно, не прикладывается платформа «1С:Предприятие 8.2». Предполагается, что вы уже имеете один из продуктов системы «1С:Предприятие 8.2», который позволяет изменять (конфигурировать) прикладные решения. Такой возможностью обладают

большинство программных продуктов «1С:Предприятия 8.2», исключение составляют лишь базовые версии.

Если же вы приобрели эту книгу в составе продукта «1С:Предприятие 8.2». Версия для обучения программированию», то в этот продукт входит учебная версия платформы «1С:Предприятие 8.2». Ее можно использовать в том случае, если в вашем распоряжении нет других продуктов системы «1С:Предприятие 8.2».

Учебная версия платформы обладает ограничениями, которые не позволят вам полностью выполнить примеры, приведенные в этой книге. Таких ограничений немного, и они не носят принципиального характера. Однако сказать о них необходимо.

В 24-м занятии рассматривается обмен данными. Учебная версия платформы не позволит проверить в работе вторую часть примера – распределенную информационную базу. Однако первую, более общую, часть (универсальный механизм обмена) вы сможете изучить полностью.

В 20-м занятии рассматривается механизм регламентных заданий. Учебная версия платформы не позволит запустить одновременно два сеанса работы – планировщика регламентных заданий и обычный пользовательский сеанс. Но вы сможете запустить их по очереди и убедиться, что каждый из них работает

правильно.

В 22-м занятии создается список пользователей системы. Учебная версия платформы не позволит задать пароли для пользователей и не позволит установить аутентификацию средствами операционной системы. Но это не имеет принципиального значения для изучения, т. к. вы все равно сможете запустить систему от имени каждого из созданных пользователей. Только ни у одного из них не будет пароля.

Занятие 1 (0:40). Знакомство, создание информационной базы

продолжительность

Ориентировочная продолжительность занятия – 40 минут.

Наше первое занятие будет посвящено знакомству с системой «1С:Предприятие 8» и главным инструментом разработчика – конфигуратором.

Вы узнаете, что обозначается терминами платформа, конфигурация и прикладное решение. Познакомитесь с различными режимами запуска системы «1С:Предприятие 8».

Узнаете, что такое объект конфигурации, как можно создать новый объект и задать его свойства.

В заключение вы создадите новую пустую информационную базу для разработки нашей учебной конфигурации.

Программирование или разработка?

Что же я делаю?! Такой вопрос периодически возникает у всех, кто

сталкивался или просто интересовался разработками на «1С:Предприятии».

«Пишу программу», – вот наиболее частый ответ. «На чем?» – «На 1С». «На чем вы работаете?» – «На 1С». «На чем это написано?» – «На 1С». «Требуется бухгалтер со знанием 1С», «требуется программист «1С» на неполный рабочий день…» и т. д.

Такие фразы можно встретить постоянно, и вам они наверняка хорошо знакомы. Для человека непосвященного в них нет ничего особенного. Однако тех, кто имеет представление о разработке на «1С:Предприятии», такие вопросы зачастую могут поставить в тупик, потому что в этих фразах термином *1С* обозначаются совершенно разные предметы, а термин *программа* и вовсе сбивает с толку…

Если ваша цель – научиться «программировать на 1С», то эта цель не совсем верная. В системе «1С:Предприятие 8» есть встроенный язык, но он занимает далеко не главное место в процессе разработки. И эта книга не учит программированию в общепринятом понимании этого слова. Эта книга учит *разработке прикладных решений на основе платформы «1С:Предприятия 8»* – процессу, в котором программирование, безусловно, присутствует, но лишь как один из инструментов разработки.

Это важно понимать с самого начала, еще до того, как вы начнете делать

первые шаги в «1С:Предприятии 8».

А чтобы было понятно, что именно мы будем создавать с вами на протяжении этой книги, объясним сначала, что представляет собой система «1С:Предприятие 8» вообще.

Общие сведения о системе

«1С:Предприятие» является универсальной системой автоматизации экономической и организационной деятельности предприятия. Поскольку такая деятельность может быть довольно разнообразной, система «1С:Предприятие» может приспосабливаться к особенностям конкретной области деятельности, в которой она применяется. Для обозначения такой способности используется термин *конфигурируемость*, то есть возможность настройки системы на особенности конкретного предприятия и класса решаемых задач.

Это достигается благодаря тому, что «1С:Предприятие» – это не просто программа, существующая в виде набора неизменяемых файлов, а совокупность различных программных инструментов, с которыми работают разработчики и пользователи. Логически всю систему можно разделить на две большие части, которые тесно взаимодействуют друг с другом: *конфигурацию и платформу*, которая управляет работой конфигурации.

Для того чтобы легче понять взаимодействие этих частей системы, сравним ее с проигрывателем компакт-дисков. Как вы хорошо знаете, проигрыватель служит для того, чтобы слушать музыку. На вкус и цвет товарищей нет, поэтому существует множество разнообразных компакт-дисков, на которых записаны музыкальные произведения на любой вкус.

Чтобы прослушать какую-либо композицию, нужно вставить компакт-диск в проигрыватель, и проигрыватель воспроизведет записанное на диске музыкальное произведение. Более того, современный проигрыватель компакт-дисков даже позволит вам записать собственную подборку музыкальных произведений, то есть создать новый компакт-диск.

Сам по себе проигрыватель совершенно бесполезен без компакт-диска, точно так же, как компакт-диск не может принести нам никакой пользы (кроме как стать подставкой под чашку кофе), если у нас нет проигрывателя.

Возвращаясь к системе «1С:Предприятие», можно сказать, что платформа является своеобразным «проигрывателем», а конфигурация – «компакт-диском». Платформа обеспечивает работу конфигурации и позволяет вносить в нее изменения или создавать собственную конфигурацию.

Существует одна платформа («1С:Предприятие 8») и множество конфигураций. Для функционирования какого-либо прикладного решения всегда необходима

платформа и какая-либо (одна) конфигурация (рис. 1.1).

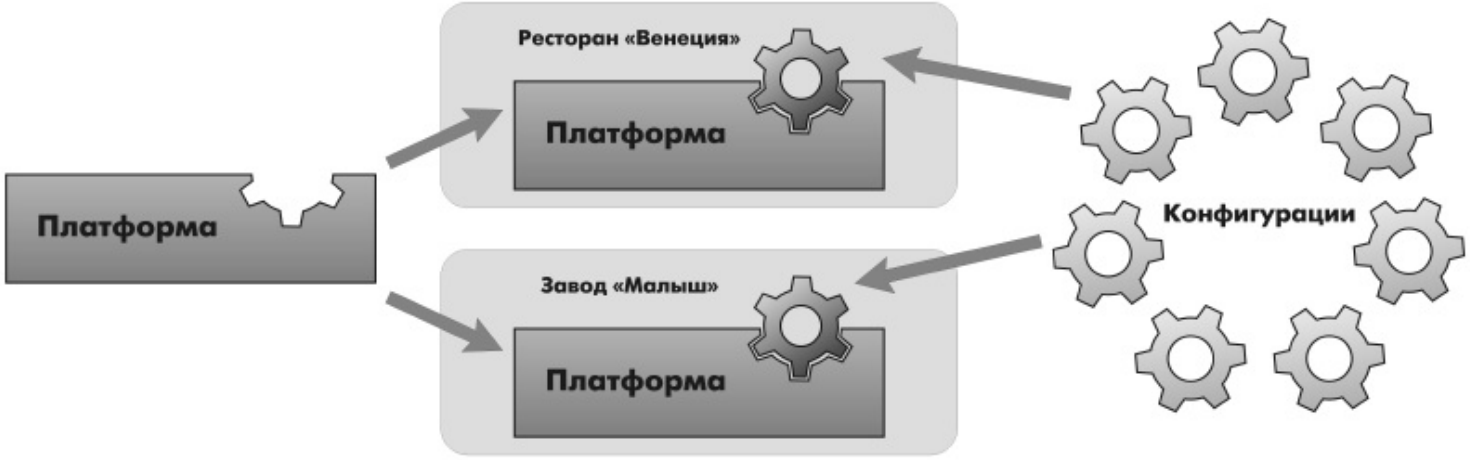


Рис. 1.1. Конфигураций много, а платформа одна

Сама по себе платформа не может выполнить никаких задач автоматизации, так как она создана для обеспечения работы какой-либо конфигурации. То же самое с конфигурацией: чтобы выполнить те задачи, для которых она создана, необходимо наличие платформы, управляющей ее работой.

Конфигурация и прикладное решение

Наконец-то мы можем ответить на вопрос, который был задан в предыдущем разделе: в процессе чтения этой книги и выполнения демонстрационного

примера мы разработаем *конфигурацию*.

Здесь следует сказать о небольшой двойственности терминологии, которая будет использоваться в дальнейшем. Двойственность заключается в употреблении разных терминов для обозначения одного и того же предмета: *конфигурация и прикладное решение*.

Эти термины обозначают ту часть системы «1С:Предприятие», которая работает под управлением платформы и которую видят все пользователи. Бывает, конечно, что пользователи работают и с инструментальными средствами платформы, но это продвинутые пользователи. Употребление одного или другого термина зависит от контекста, в котором ведется изложение.

Если речь идет о действиях разработчика, то употребляется термин конфигурация, поскольку это точный термин «1С:Предприятия». Термин прикладное решение, напротив, является более общепринятым и понятным для пользователя системы «1С:Предприятие».

Итак, поскольку задачи автоматизации, как было упомянуто выше, могут быть самыми разными, фирма «1С» и ее партнеры выпускают прикладные решения, каждое из которых предназначено для автоматизации одной определенной области человеческой деятельности. В качестве примера существующих

прикладных решений можно перечислить следующие типовые решения:

- «1С:Управление небольшой фирмой 8»,
- «1С:Бухгалтерия 8»,
- «1С:Предприятие 8. Управление торговлей»,
- «1С:Зарплата и управление персоналом 8»,
- «1С:Предприятие 8. Управление производственным предприятием»,
- «1С:Консолидация 8».

Существует также множество других типовых прикладных решений. Более подробно о них можно узнать на сайте *http://v8.1c.ru/solutions/applied_solutions.htm*.

Типовое прикладное решение является, по сути, универсальным, и способно удовлетворить потребности самых разных предприятий, работающих в одной области деятельности. И это хорошо.

С другой стороны, такая универсальность неизбежно приведет к тому, что на конкретном предприятии будут использоваться далеко не все возможности прикладного решения, а каких-то возможностей в нем будет недоставать (нельзя угодить всем).

Вот тут и выходит на передний план *конфигурируемость* системы, поскольку платформа, помимо управления работы конфигурацией, содержит средства, позволяющие вносить изменения в используемую конфигурацию. Более того, платформа позволяет создать свою собственную конфигурацию с нуля, если по каким-либо причинам использование типовой конфигурации представляется нецелесообразным.

Обратите внимание, как мы в одном абзаце перешли от *прикладного решения* к *конфигурации*. Ничего не поделаешь, для пользователя понятнее так, а для разработчика – по-другому.

Таким образом, если вернуться к сравнению с проигрывателем компакт-дисков, мы можем изменять по своему вкусу мелодии, которые были ранее записаны на компакт-диске, и даже создавать диски со своими собственными музыкальными произведениями. При этом нам не потребуются какие-либо музыкальные инструменты – все необходимое для создания мелодий есть в нашем проигрывателе компакт-дисков.

Режимы работы системы

Для того чтобы обеспечить такие возможности, система «1С:Предприятие» имеет различные режимы работы: *1С:Предприятие* и *Конфигуратор*.

Режим *1С:Предприятие* является основным и служит для работы пользователей системы. В этом режиме пользователи вносят данные, обрабатывают их и получают итоговые результаты.

Режим *Конфигуратор* используется разработчиками и администраторами информационных баз. Именно этот режим и предоставляет инструменты, необходимые для модификации существующей или создания новой конфигурации.

Поскольку задача нашей книги состоит в том, чтобы научить вас создавать собственные конфигурации и изменять существующие, дальнейшее повествование будет в основном посвящено работе с системой в режиме *Конфигуратор*. И лишь иногда, чтобы проверить результаты нашей работы, мы будем запускать систему в режиме *1С:Предприятие*.

Изучение этой книги предполагает, что у вас уже установлена на компьютере система «1С:Предприятие 8». Если это не так, то сейчас самое время это сделать, так далее будет непосредственно описываться последовательность работы с программой.

Создание новой информационной базы

При установке системы «1С:Предприятие» у вас не должно возникнуть никаких

трудностей. Процесс установки подробно описан в документации «1С:Предприятие 8.2. Руководство администратора».

Также у вас не должно возникнуть трудностей при запуске системы и создании информационной базы, которая содержит пустую конфигурацию.

Будьте внимательны! Для выполнения примера, содержащегося в книге, вам потребуется информационная база для разработки новой конфигурации, а не база, созданная из шаблона. Для этого вам нужно выполнить следующие действия.

Запустите «1С:Предприятие». В открывшемся диалоге вы увидите список информационных баз, с которыми вы работаете. Если этот список пуст, система сама предложит вам создать новую базу. Если же в списке информационных баз содержится какая-либо база, например, у вас установлена демонстрационная конфигурация, то для создания новой базы нажмите кнопку *Добавить* (рис. 1.2).

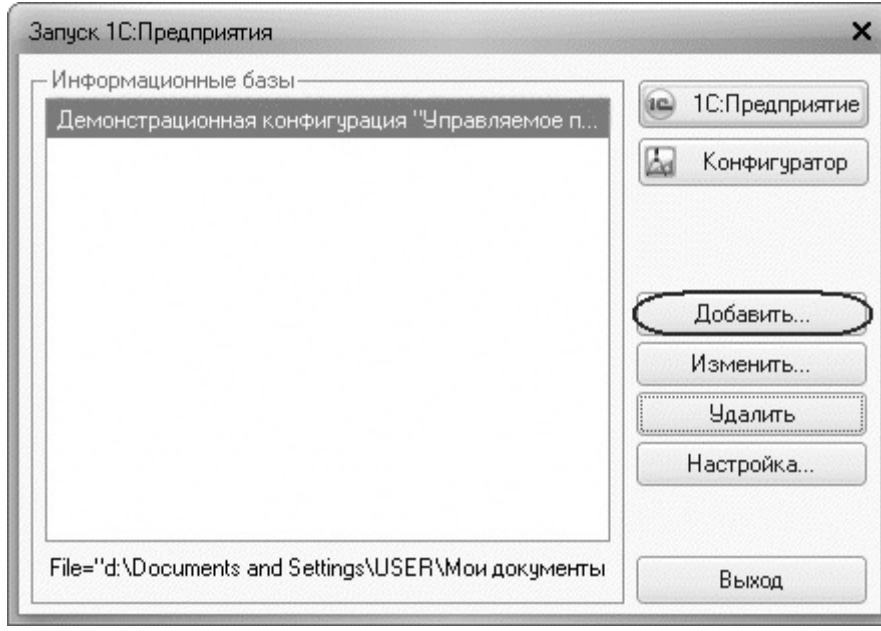


Рис. 1.2. Создание новой информационной базы. Шаг 1

В открывшемся диалоге выберите пункт *Создание новой информационной базы* (рис. 1.3).

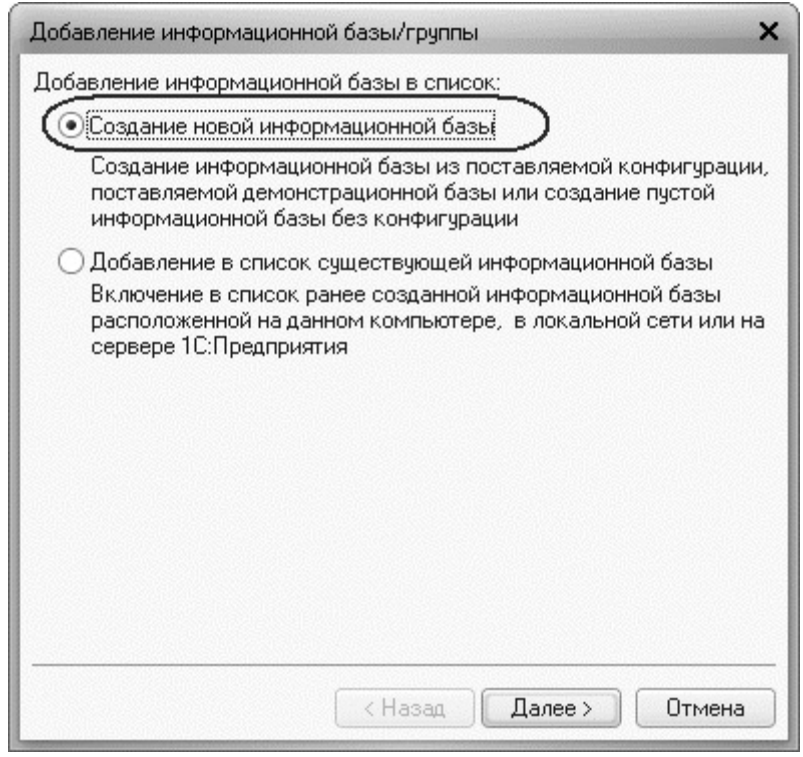


Рис. 1.3. Создание новой информационной базы. Шаг 2

Нажмите кнопку *Далее*. На следующем шаге выберите пункт *Создание информационной базы без конфигурации...* (рис. 1.4).

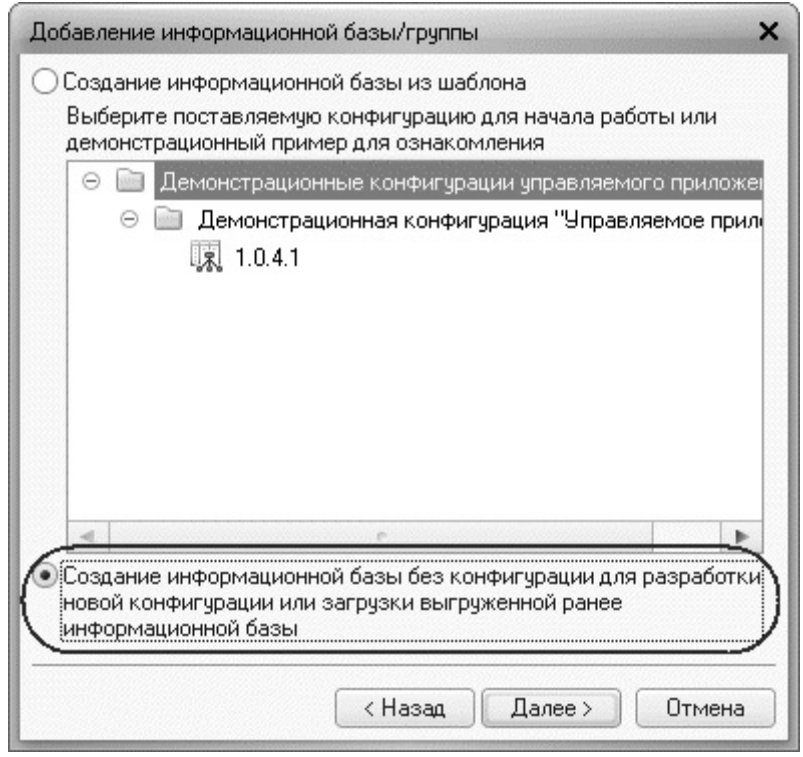


Рис. 1.4. Создание новой информационной базы. Шаг 3

Нажмите кнопку *Далее*. На следующем шаге задайте наименование вашей информационной базы и выберите тип ее расположения *На данном компьютере…* (рис. 1.5).

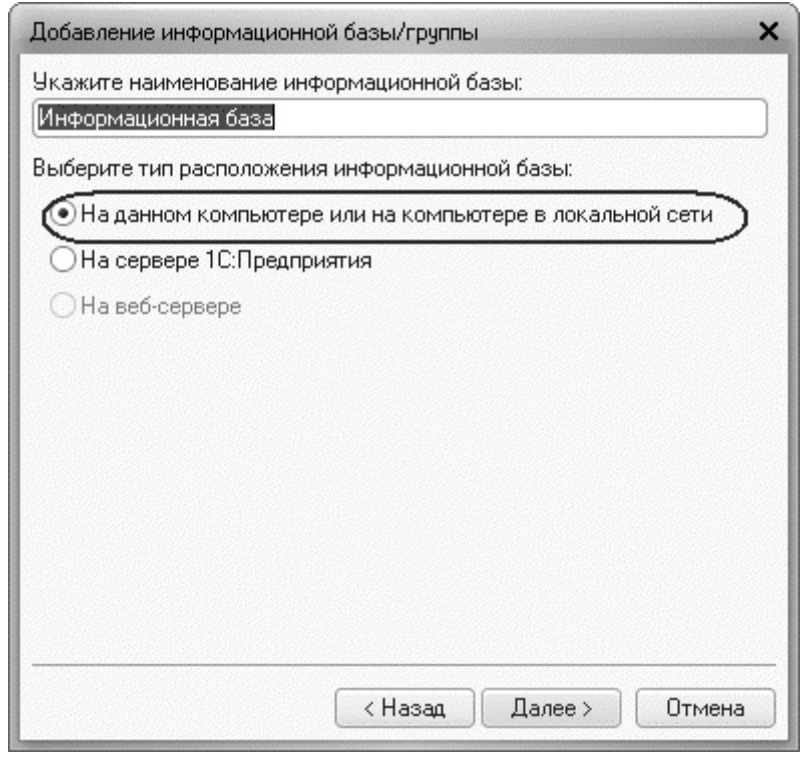


Рис. 1.5. Создание новой информационной базы. Шаг 4

Нажмите кнопку *Далее*. На следующем шаге укажите каталог для расположения вашей информационной базы. Язык по умолчанию установлен в значение *Русский* (рис. 1.6).

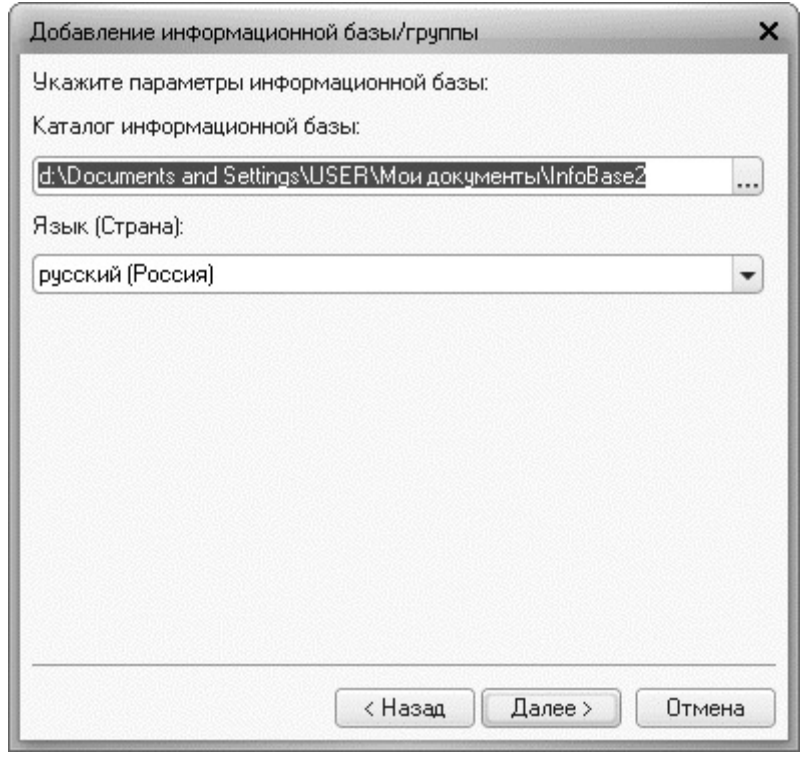


Рис. 1.6. Создание новой информационной базы. Шаг 5

Нажмите кнопку *Далее*. На следующем шаге нажмите кнопку *Готово* (рис. 1.7).

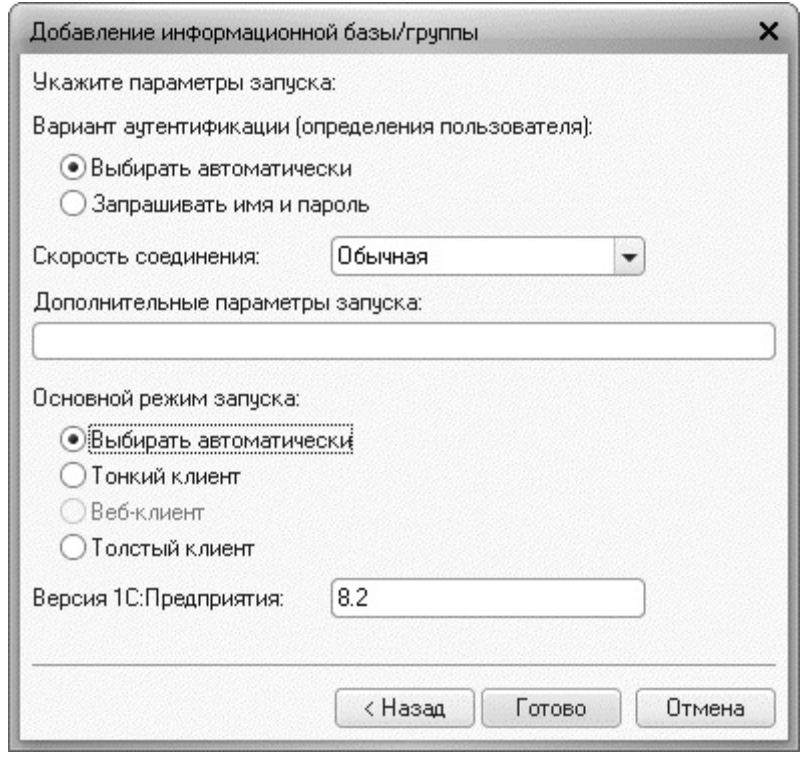


Рис. 1.7. Создание новой информационной базы. Шаг 6

В диалоге запуска «1С:Предприятия», в списке информационных баз вы увидите созданную вами новую пустую базу (рис. 1.8).

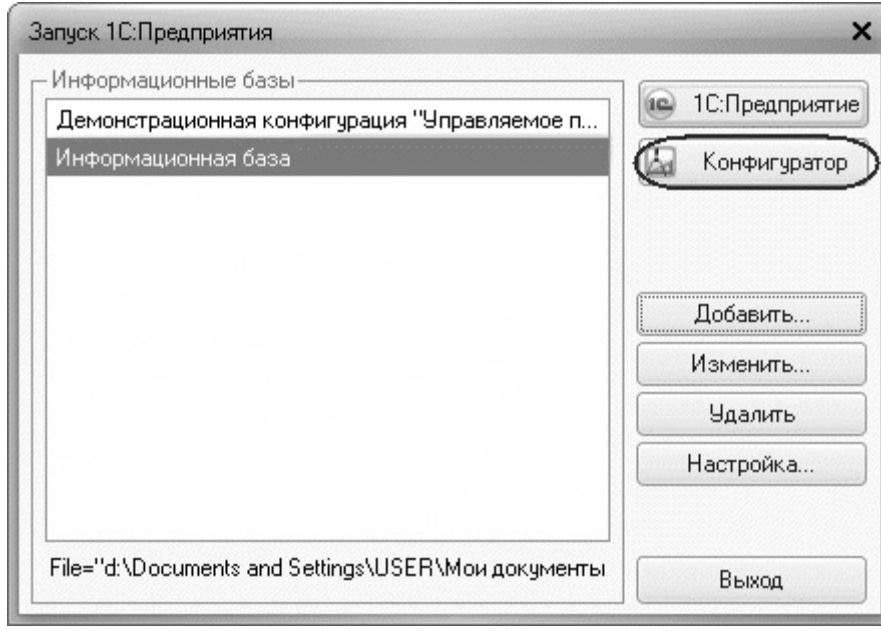


Рис. 1.8. Запуск «1С:Предприятия» в режиме «Конфигуратор»

В режиме «Конфигуратор»

Знакомство с конфигуратором

Итак, запустим «1С:Предприятие» в режиме *Конфигуратор*. Для этого нажмем кнопку *Конфигуратор* в диалоге запуска системы (см. рис. 1.8).

Перед вами окно конфигуратора (рис. 1.9).

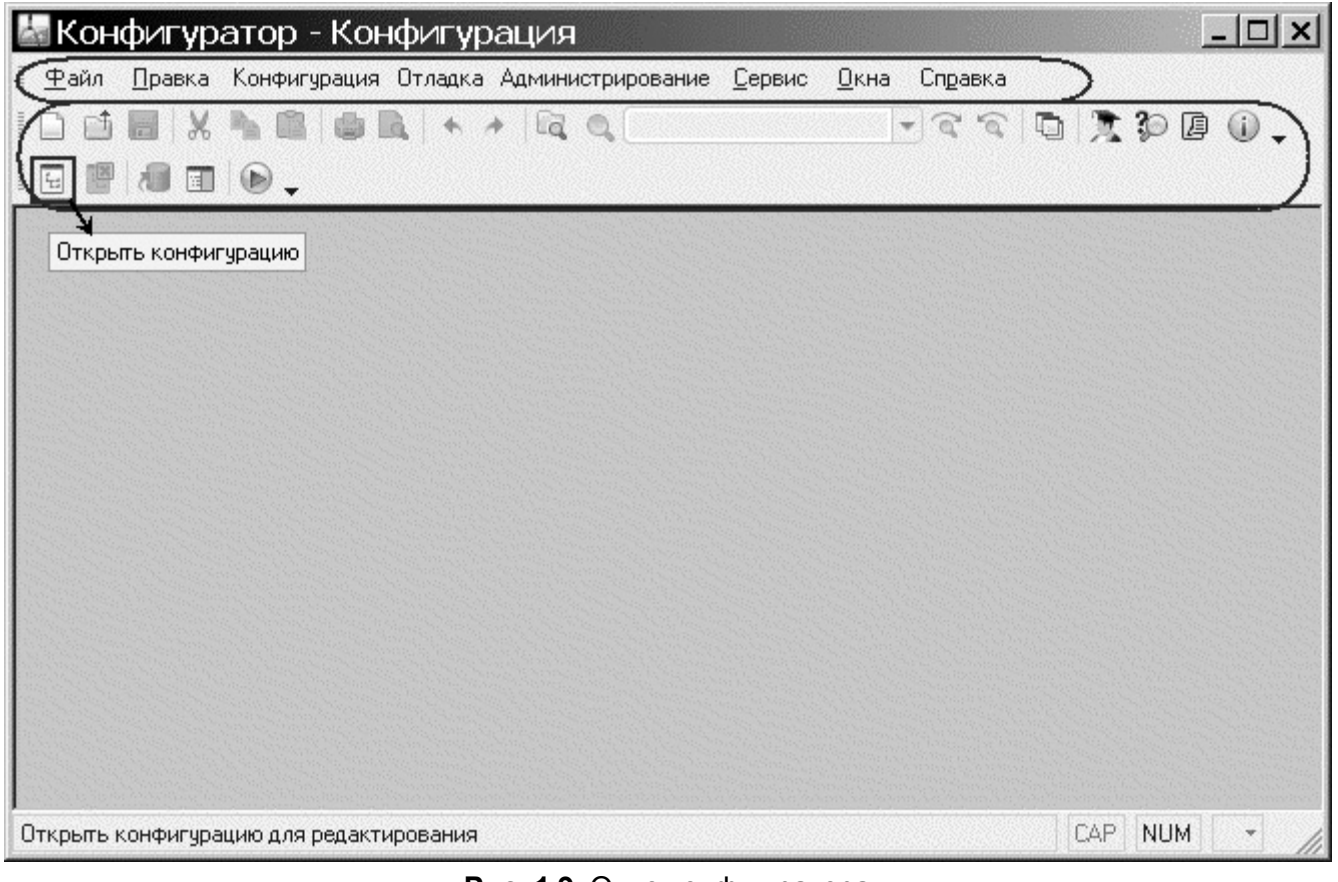


Рис. 1.9. Окно конфигуратора

Именно с помощью этого инструмента мы будем создавать нашу конфигурацию. Сразу под заголовком окна находится главное меню конфигуратора, содержащее пункты *Файл*, *Правка*, *Конфигурация*,

Администрирование и т. п. В каждом из этих пунктов содержится много подпунктов, вызов которых обеспечивает выполнение различных действий конфигуратора.

Ниже находится панель инструментов конфигуратора, в которую в виде кнопок-пиктограмм помещены наиболее часто используемые действия, вызываемые из меню.

Таким образом, одни и те же действия можно выполнить двумя разными способами: вызвав определенный пункт меню или нажав соответствующую ему кнопку на панели инструментов.

Большое количество незнакомых пиктограмм часто смущает начинающего разработчика. Не следует этого бояться – со временем вы сможете свободно ориентироваться среди них. Просто подведите к какой-либо кнопке мышь, задержите ее на несколько секунд, и появится всплывающая подсказка, поясняющая назначение этой кнопки (см. рис. 1.9).

Вероятно, сначала вы будете пользоваться пунктами меню, но постепенно ваша работа сама собой переместится на панель инструментов, так как это удобнее. Со временем в случае надобности вы сможете настраивать панель инструментов под себя, удаляя или добавляя нужные вам кнопки (рис. 1.10).

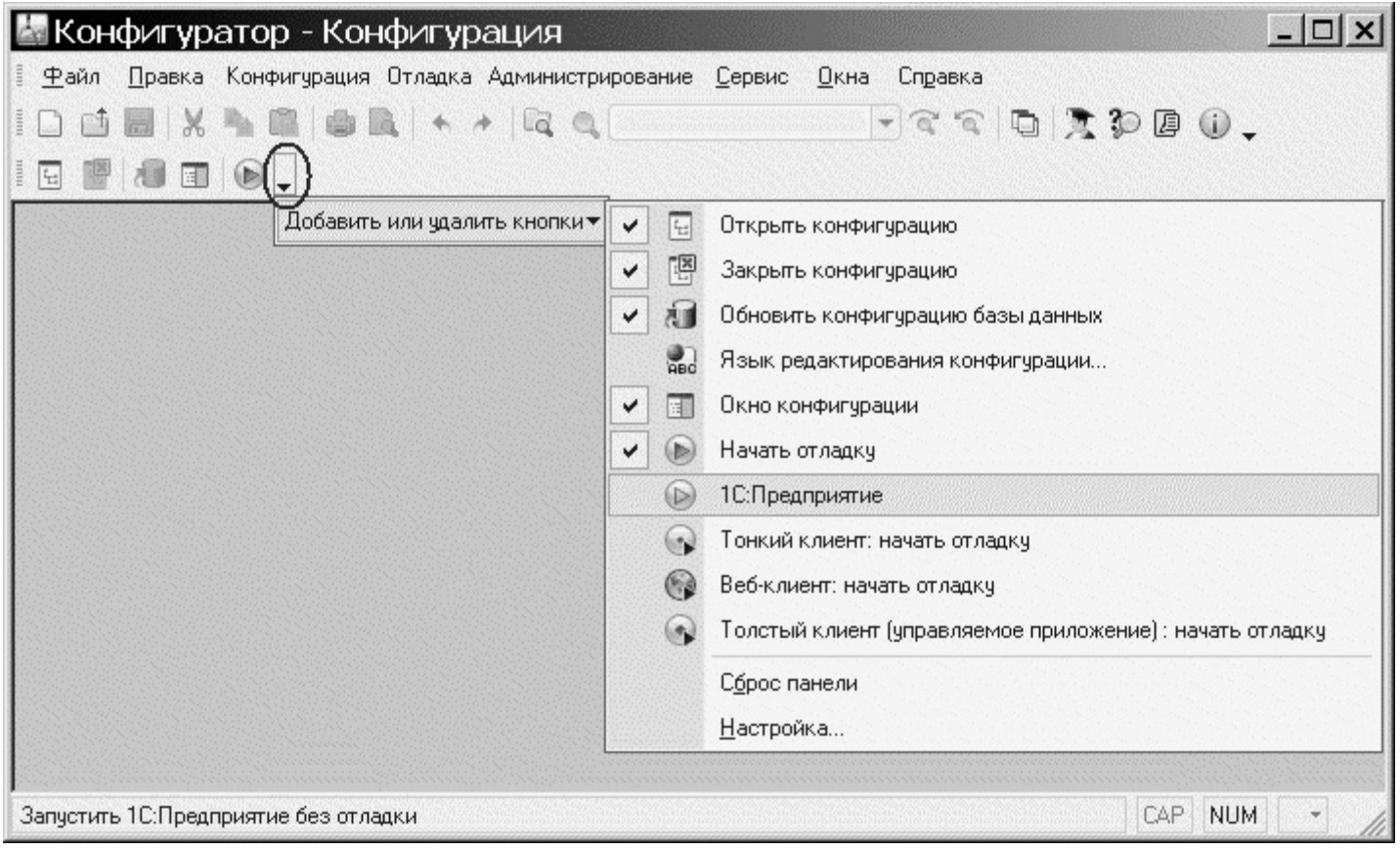


Рис. 1.10. Настройка панели инструментов конфигуратора

Дерево объектов конфигурации

Выполним первую команду, с которой начинается работа с любой конфигурацией, – откроем конфигурацию с помощью пункта меню

Конфигурация > *Открыть конфигурацию* или соответствующей кнопки на панели инструментов (см. рис. 1.9).

На экране откроется *дерево объектов конфигурации* (рис. 1.11).

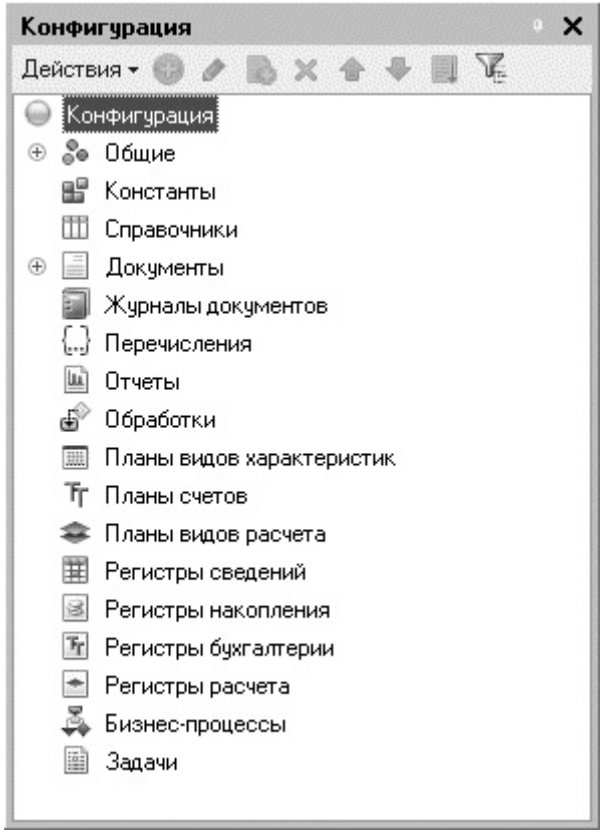


Рис. 1.11. Дерево конфигурации

Можно сказать, что дерево объектов конфигурации – основной инструмент, с которым работает разработчик. Дерево объектов конфигурации содержит в себе практически всю информацию о том, из чего состоит конфигурация.

Наверняка у вас уже возник вопрос: почему в дереве что-то есть, если мы пока еще ничего не создавали?

Дело в том, что для облегчения работы разработчика все, из чего состоит конфигурация сгруппировано, и сейчас дерево и показывает вам эти группы. Если вы будете перемещаться по дереву и нажимать на +, то увидите, что ни в одной группе ничего нет. Исключение составит лишь группа *Общие* > *Языки*, в которой вы обнаружите «нечто» под названием «Русский». Этот «Русский» платформа создала для вас сама, поскольку в данном случае конфигуратор использует русскоязычный интерфейс.

Хотелось бы уже начать что-нибудь делать, но прежде следует определиться с терминами. Вы наверняка уже заметили, что, говоря о содержимом конфигурации, мы сознательно избегали использования каких-либо терминов. Но теперь настало время, когда можно определиться с терминологией и рассказать про *объекты конфигурации*.

Что такое объекты конфигурации

Конфигурация представляет собой описание. Она описывает структуру данных, которые пользователь будет использовать в режиме работы *1С:Предприятие*.

Кроме этого, конфигурация описывает всевозможные алгоритмы обработки этих данных, содержит информацию о том, как эти данные должны будут выглядеть на экране и на принтере и т. д. В дальнейшем платформа «1С:Предприятие» на основании этого описания создаст базу данных, которая будет иметь необходимую структуру и предоставит пользователю возможность работать с этой базой данных.

Для того чтобы систему «1С:Предприятие» можно было быстро и легко настраивать на нужные прикладные задачи, все описание, которое содержит конфигурация, состоит из неких логических единиц, называемых объектами конфигурации. Возможно, вы уже успели заглянуть в книгу документации «1С:Предприятие 8.2. Руководство разработчика», в которой дается краткое описание объекта конфигурации.

Мы не будем дублировать это определение в настоящей книге, поскольку наша задача не изложить концепцию построения системы «1С:Предприятие» как структуры метаданных, описанной в терминах классов проблемно-ориентированных бизнес-сущностей, а научить вас методически правильно и грамотно использовать возможности «1С:Предприятия».

Поэтому что такое объекты конфигурации, мы объясним на бытовом уровне. Однако это даст вам возможность правильно понимать назначение объектов применительно к тем задачам, которые мы будем решать.

С одной стороны, объекты конфигурации представляют собой детали «конструктора», из которого собирается конфигурация. Обычно в конструкторе существует некоторый набор деталей. Детали могут быть разного вида: длинные, короткие, квадратные, прямоугольные и т. д. Теперь представьте, что деталей каждого вида мы можем создавать столько, сколько нам нужно (скажем, 5 длинных и 3 короткие). Мы можем соединять детали между собой различными способами.

То же и с объектами конфигурации. Мы можем создавать только объекты определенных видов. Но каждого вида объектов мы можем создать столько, сколько нам нужно. Объекты одного вида отличаются от объектов другого вида тем, что имеют разные свойства (точнее говоря, разный набор свойств). Объекты могут взаимодействовать друг с другом, и мы можем описать такое взаимодействие.

В чем еще сходство объектов конфигурации с деталями конструктора? В конструкторе обычно есть блоки, которые можно скрепить между собой, и есть другие детали, например колеса, которые скрепить между собой нельзя, зато их можно соединить с осью, и тогда колеса будут вращаться. То есть разные

детали конструктора по-разному ведут себя.

Объекты конфигурации также обладают различным поведением, и оно зависит от вида объекта. Одни объекты могут выполнять какие-то действия, другие этих действий выполнять не могут, зато у них есть свой собственный набор действий.

Следующую особенность объектов конфигурации можно продемонстрировать на примере автомобиля. Автомобиль состоит из большого количества деталей. Одна из деталей автомобиля – это двигатель. Но двигатель, в свою очередь, тоже состоит из набора деталей, причем в разных двигателях могут использоваться одни и те же детали.

Также «сложные» объекты конфигурации состоят из более «простых», и одни и те же «простые» объекты могут входить в состав сложных объектов. Такая структура позволяет упростить работу с объектами конфигурации, поскольку если мы знаем, как работать с каким-либо «простым» объектом, то в любом «сложном» объекте, в состав которого он входит, мы будем работать с ним все тем же образом.

И, наконец, самое важное качество объектов конфигурации – это их прикладная направленность. Объекты конфигурации не просто некие абстрактные конструкции, при помощи которых разработчик пытается описать поставленную

перед ним задачу. Они представляют собой аналоги реальных объектов, которыми оперирует предприятие в ходе своей работы.

Например, на каждом предприятии существуют различные документы, с помощью которых оно фиксирует факты совершения хозяйственных операций. Точно так же в конфигурации существуют объекты вида *Документ*.

Кроме этого, на каждом предприятии обязательно ведется список сотрудников, справочник номенклатуры или товаров. В конфигурации тоже есть специальные объекты вида *Справочник*, которые позволяют разработчику создавать компьютерные аналоги таких списков.

Как мы уже говорили, на основе объектов конфигурации платформа создает в базе данных таблицы, в которых будут храниться данные. В литературе, как правило, объект конфигурации и соответствующий ему набор таблиц базы данных принято называть одинаково.

Например, если в конфигурации существует объект *Справочник Сотрудники*, то набор таблиц, созданный платформой на основе этого объекта конфигурации, также называют *Справочником Сотрудники*.

Мы отойдем от такого «размытого» стиля изложения и в тех местах, где речь пойдет о конфигурации, будем использовать явное уточнение – *объект*

конфигурации справочник Сотрудники. Там же, где речь пойдет о базе данных, мы будем говорить просто: *справочник Сотрудники*.

Как добавить объект конфигурации

Прежде чем мы приступим к добавлению первых объектов конфигурации, нужно иметь в виду, что для разработки собственной конфигурации, автоматизирующей хозяйственную деятельность предприятия, разработчик может использовать только ограниченный набор объектов конфигурации, «жестко зашитый» в платформе. Ему не дано возможности создавать собственные объекты конфигурации. Он только может добавлять в конфигурацию какой-либо из стандартных объектов, поставляемых системой.

Перед началом работы следует объяснить некоторые приемы работы с конфигуратором.

Для того чтобы открыть и закрыть конфигурацию, следует использовать пункты меню *Конфигурация* > *Открыть конфигурацию* и *Конфигурация* > *Закрыть конфигурацию* или соответствующие им кнопки на панели инструментов.

После того как конфигурация открыта, ее состав появляется в окне дерева конфигурации (см. рис. 1.11). Это окно вы можете закрыть, как любое другое окно Windows, при этом конфигурация останется открытой (то есть доступной

для редактирования). Чтобы снова отобразить на экране окно дерева конфигурации, следует воспользоваться командой меню *Конфигурация* > *Окно конфигурации*.

Добавить новый объект конфигурации можно несколькими способами, и вы можете использовать наиболее понятный и удобный для вас.

Первый способ. Необходимо установить курсор на ту ветку объектов конфигурации, которая вас интересует, и в командной панели окна конфигурации нажать кнопку *Действия* > *Добавить* (рис. 1.12).

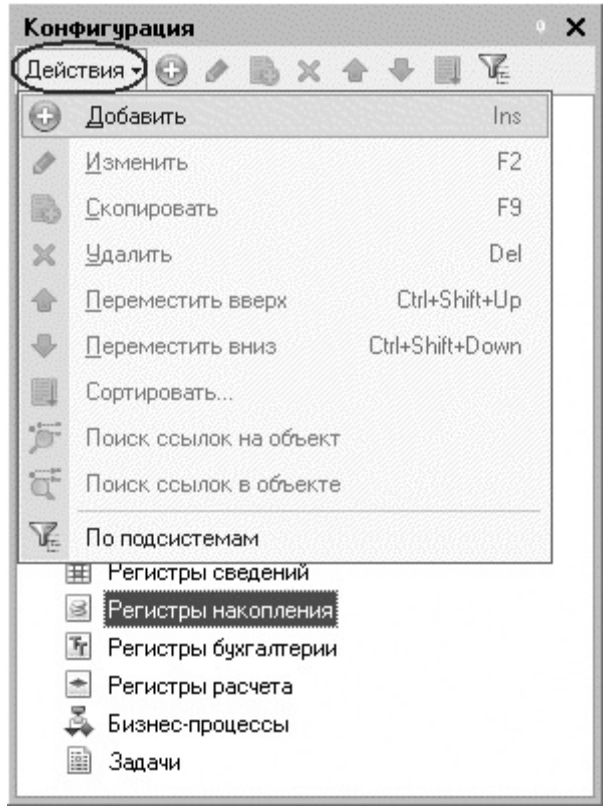


Рис. 1.12. Добавление нового объекта конфигурации

Второй способ. Вы можете воспользоваться контекстным меню, которое вызывается при нажатии на правую клавишу мыши. Установите курсор на интересующую вас ветку объектов конфигурации и нажмите правую клавишу мыши. В появившемся меню выберите пункт *Добавить* (рис. 1.13).

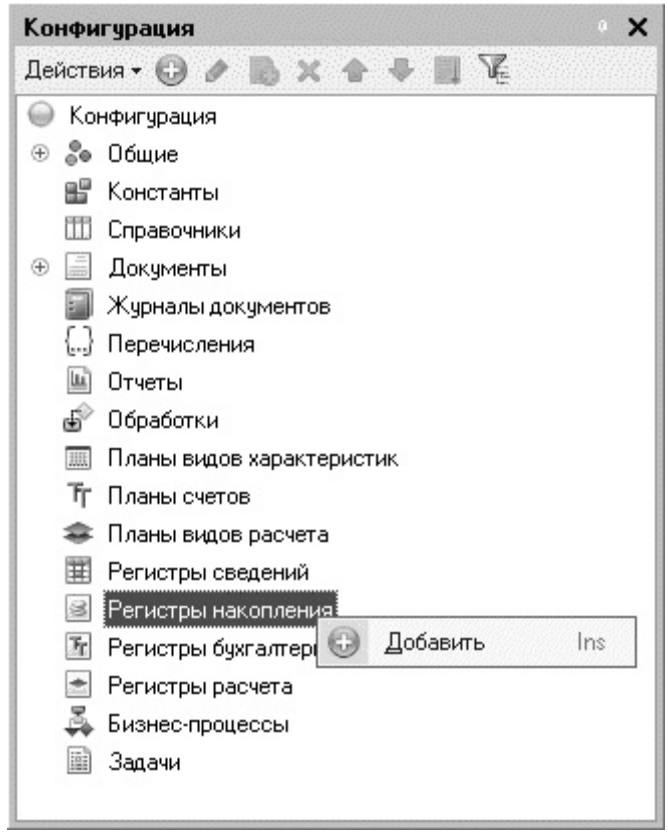


Рис. 1.13. Добавление нового объекта конфигурации

Третий способ. Установите курсор на интересующую вас ветку объектов конфигурации и в командной панели окна конфигурации нажмите кнопку *Добавить* (с пиктограммой +), рис. 1.14.

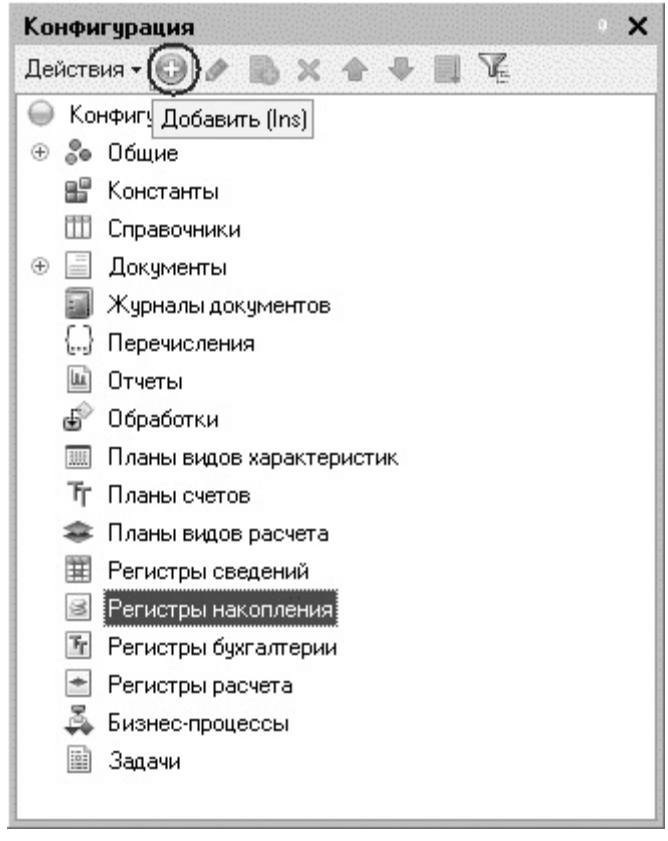


Рис. 1.14. Добавление нового объекта конфигурации

Последний способ, на наш взгляд, наиболее удобен, поэтому в основном мы будем использовать именно его.

Палитра свойств

Итак, мы начинаем!

Зададим имя нашей конфигурации и на этом примере познакомимся с палитрой свойств, с помощью которой разработчик может задавать свойства создаваемых им объектов конфигурации.

Палитра свойств – это специальное служебное окно, которое позволяет редактировать все свойства объекта конфигурации и другую связанную с ним информацию. Поскольку разные объекты конфигурации имеют самые разные свойства, содержимое этого окна будет меняться в зависимости от того, какой объект является текущим (на каком объекте конфигурации установлен курсор).

Выделим в дереве объектов конфигурации корневой элемент *Конфигурация* и двойным щелчком мыши откроем его палитру свойств.

Зададим имя конфигурации *ПособиеДляНачинающих*.

Соответствующий ему синоним устанавливается автоматически, но его можно изменить по своему усмотрению. В дальнейшем именно его мы будем видеть в рабочем окне «1С:Предприятия» (рис. 1.15).

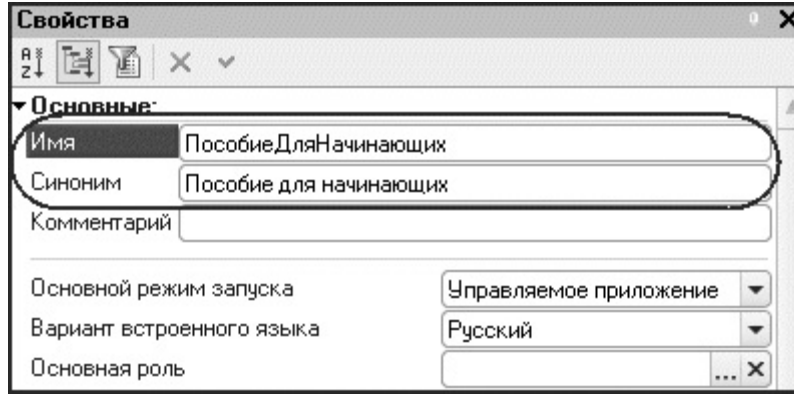


Рис. 1.15. Палитра свойств конфигурации

При некоторых действиях разработчика палитра свойств открывается автоматически. Но разработчик всегда может открыть палитру свойств объекта конфигурации самостоятельно, воспользовавшись пунктом *Свойства* контекстного меню правой кнопки мыши.

В этом случае, как и сейчас, палитра свойств откроется и будет закреплена на рабочей области конфигуратора. То есть при выделении какого-либо объекта конфигурации окно его свойств всегда будет открыто. Однако есть удобная возможность открепить палитру свойств, используя символ кнопки в заголовке окна палитры свойств (рис. 1.16).

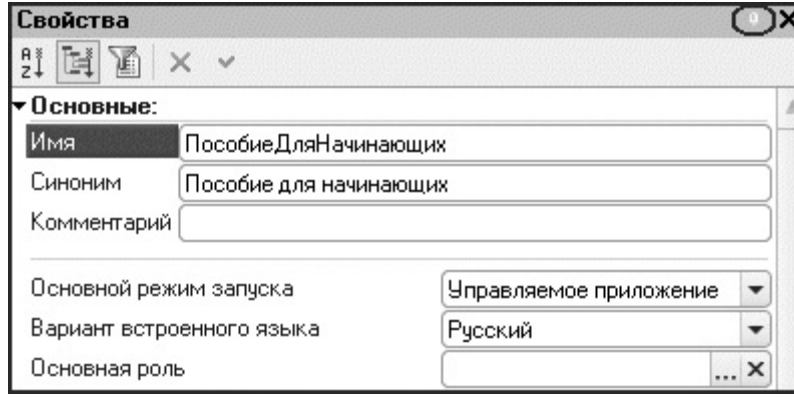


Рис. 1.16. «Открепим» палитру свойств

В этом состоянии, при наведении курсора мыши на любое другое окно, палитра свойств будет сворачиваться на дополнительную панель в правой части экрана (рис. 1.17).

Рис. 1.17. Кнопка на дополнительной панели

При наведении курсора мыши на символ свернутой палитры свойств она будет открываться.

Подобным поведением (возможностью быть прикрепленным, прячущимся и т. д.) обладает не только окно палитры свойств, но и другие окна конфигуратора (например, окно дерева конфигурации).

Запуск отладки в режиме «1С:Предприятие»

Теперь проверим наши первые изменения в режиме *1С:Предприятие*.

Для этого выполним пункт меню *Отладка* > *Начать отладку* или нажмем соответствующую кнопку  на панели инструментов конфигуратора. Система сама анализирует наличие изменений в конфигурации и выдает соответствующий вопрос об обновлении конфигурации базы данных (рис. 1.18).

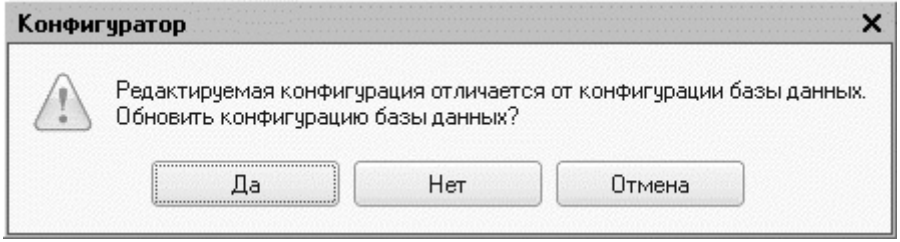


Рис. 1.18. Вопрос об обновлении конфигурации

Мы не будем пока останавливаться подробно на том, почему это происходит, а

рассмотрим этот вопрос в разделе «Основная конфигурация и конфигурация базы данных».

На вопрос конфигуратора ответим *Да*, и на экране появится окно «1С:Предприятия» (рис. 1.19).

В режиме «1С:Предприятие»

Внешний вид интерфейса прикладного решения

В заголовке окна мы видим название нашей конфигурации. Пустое пространство – это *рабочая область* приложения, которая пока ничем не заполнена.

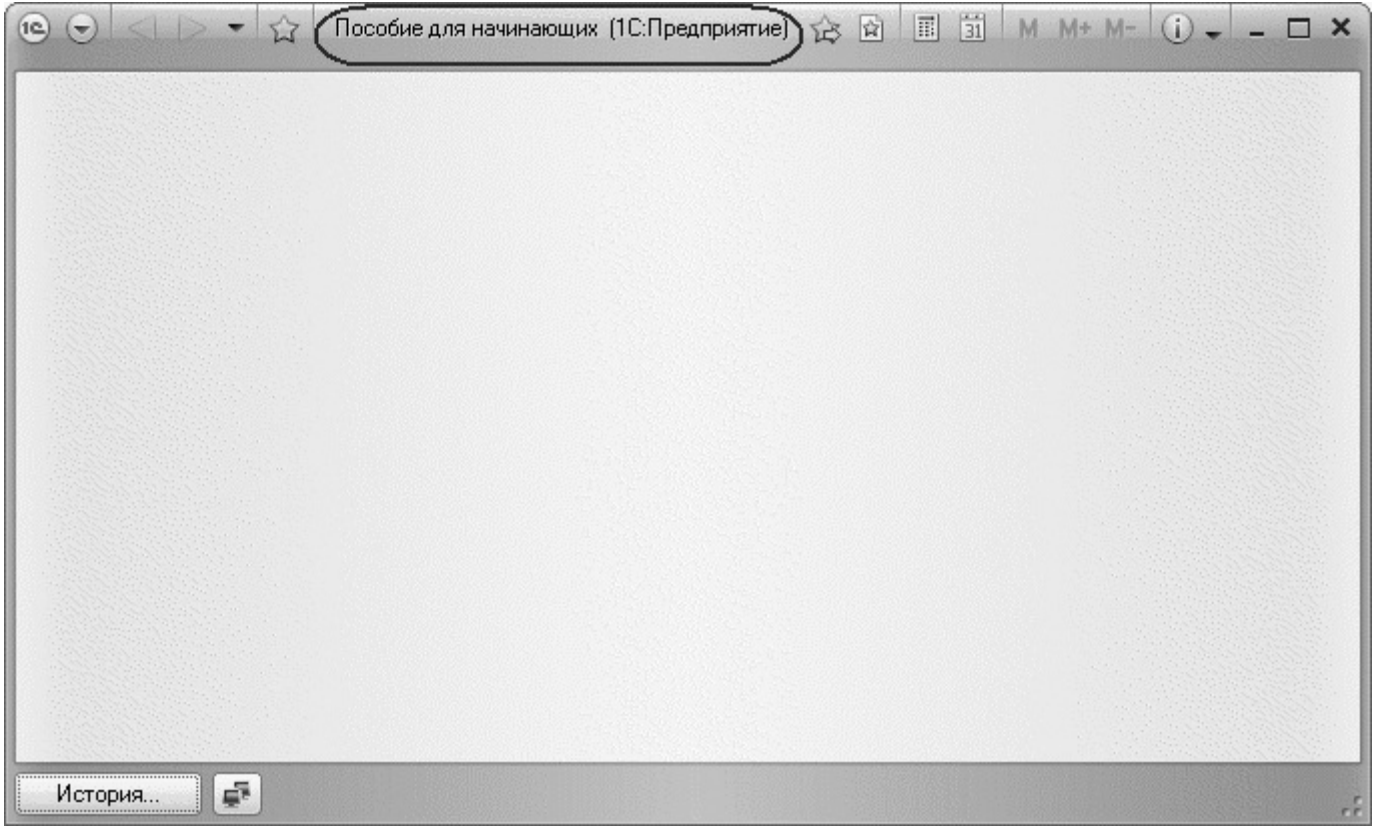


Рис. 1.19. «1С:Предприятие»

Кроме заголовка конфигурации в окне «1С:Предприятия» ничего не появилось. И этого следовало ожидать.

Мы еще не создали никаких объектов конфигурации и не создали никаких

подсистем, в которых бы эти объекты отображались.

О подсистемах как основе разработки интерфейса «1С:Предприятия» и пойдет речь на следующем занятии. А пока, взглянув на список кратких итогов первого занятия, проверьте, насколько хорошо вы поняли изложенный материал.

Контрольные вопросы

- *Что такое конфигурируемость системы «1С:Предприятие».*
- *Из каких основных частей состоит система.*
- *Что такое платформа и что такое конфигурация.*
- *Для чего используются разные режимы запуска системы «1С:Предприятие».*
- *Что такое дерево объектов конфигурации.*
- *Что такое объекты конфигурации.*
- *Что создает система на основе объектов конфигурации.*
- *Какими способами можно добавить новый объект конфигурации.*
- *Зачем нужна палитра свойств.*
- *Как запустить «1С:Предприятие» в режиме отладки.*

Занятие 2 (0:45). Подсистемы

Продолжительность

Ориентировочная продолжительность занятия – 45 минут.

На этом занятии мы познакомимся с объектом конфигурации *Подсистема* как основой декларативного описания интерфейса «1С:Предприятия 8».

Мы создадим несколько подсистем, определяющих логическую структуру прикладного решения, настроим их внешний вид и порядок их следования в интерфейсе прикладного решения.

Что такое подсистема

Подсистемы – это основные элементы для построения интерфейса «1С:Предприятия». Поэтому первое, с чего следует начинать разработку конфигурации, – это проектирование состава подсистем.

При этом перед разработчиком стоит важная и ответственная задача – тщательно продумать состав подсистем, и затем аккуратно и осмысленно привязать к подсистемам те объекты конфигурации, которые он будет создавать.

В простых прикладных решениях можно не использовать подсистемы, но мы рассмотрим общий случай, когда подсистемы используются.

Объекты конфигурации *Подсистема* позволяют выделить в конфигурации функциональные части, на которые логически разбивается создаваемое прикладное решение.

Эти объекты располагаются в ветке объектов *Общие* и позволяют строить древовидную структуру, состоящую из подсистем и подчиненных подсистем (рис. 2.1).

Рис. 2.1. Структура подсистем конфигурации

Подсистемы верхнего уровня являются основными элементами интерфейса, так как образуют разделы прикладного решения (рис. 2.2).

Рис. 2.2. Разделы прикладного решения

Каждый объект конфигурации может быть включен в одну или сразу несколько подсистем, в составе которых он будет отображаться.

Забегая вперед, скажем, что с помощью подсистем, используя видимость по

ролям, можно предоставить пользователю удобный и функциональный интерфейс, не содержащий лишних элементов. Например, кладовщик должен иметь возможность принять и выдать товар, и ему совсем не нужно видеть все, что относится к области бухгалтерского учета и оказанию услуг.

Таким образом, наличие подсистем определяет структуру прикладного решения, организует весь пользовательский интерфейс, позволяет рассортировать различные документы, справочники и отчеты по логически связанным с ними разделам, в которых пользователю будет проще их найти и удобнее с ними работать. При этом каждому конкретному пользователю будут видны лишь те разделы, то есть та функциональность прикладного решения, которые ему нужны в процессе работы.

Даже в такой небольшой конфигурации, как наша, можно выделить несколько функциональных частей, представляющих собой отдельные предметные области.

Так, можно выделить в отдельную подсистему все, что имеет отношение к бухгалтерскому учету.

Кроме этого, отдельной предметной областью является расчет зарплаты сотрудников предприятия.

Всю производственную деятельность нашей фирмы ООО «На все руки мастер» можно разделить на учет материалов и оказание услуг.

А кроме этого, для выполнения специальных административных функций с базой данных нам нужно иметь отдельную подсистему, в которую будет иметь доступ только администратор.

Поэтому сейчас мы создадим в нашей конфигурации пять новых объектов конфигурации *Подсистема*, которые будут иметь имена: *Бухгалтерия*, *РасчетЗарплаты*, *УчетМатериалов*, *ОказаниеУслуг* и *Предприятие*. Чтобы это сделать, выполним следующие действия.

Добавление подсистемы

В режиме «Конфигуратор»

Закроем приложение и вернемся в конфигуратор. Чтобы создать новые подсистемы, раскроем ветвь *Общие* в дереве объектов конфигурации, нажав на + слева от нее.

Затем выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Добавить* или нажмем соответствующую кнопку в командной панели окна конфигурации (рис. 2.3).

Рис. 2.3. Добавление новой подсистемы в дерево объектов конфигурации

После этого система откроет *окно редактирования объекта конфигурации*.

Оно предназначено специально для сложных объектов конфигурации и позволяет путем выполнения последовательных действий быстро создавать такие объекты.

Для того чтобы придерживаться правильной последовательности действий, в нижней части окна имеются кнопки *Далее* и *Назад*. Кнопка *Далее* позволяет задавать свойства объекта в нужной последовательности, чтобы ничего не пропустить и не проскочить вперед, где потребуются данные, которые должны были быть введены ранее. Кнопка *Назад* позволяет вернуться на несколько шагов назад, если вы обнаружили, что ранее ввели не все или ошибочные данные. Впоследствии вы сможете задавать свойства объектов, сразу выделяя нужную вам закладку, например, *Данные*. При открытии окна редактирования объекта конфигурации мы попадаем на закладку *Основные*.

ПРИМЕЧАНИЕ

Чтобы изменить свойства объекта в процессе разработки, часто бывает нужно повторно открыть окно редактирования объекта конфигурации. Для этого следует выделить нужный элемент в дереве объектов конфигурации и нажать в

командной панели окна конфигурации кнопку ***Изменить текущий элемент (F2)*** *или дважды щелкнуть мышью по выделенному элементу.*

Зададим *имя* подсистемы – *Бухгалтерия*. На основании имени платформа автоматически создаст *синоним* – *Бухгалтерия* (рис. 2.4).

Рис. 2.4. Установка имени и синонима подсистемы

Имя и синоним объекта конфигурации

Имя является основным свойством любого объекта конфигурации. При создании нового объекта система автоматически присваивает ему некоторое имя.

Можно использовать имя, присвоенное системой, но лучше заменить его своим, понятным именем. Имя можно задавать любое, главное, чтобы оно начиналось с буквы и не содержало некоторых специальных символов (например, пробел).

Для удобства чтения конфигурации принято составлять интуитивно понятные имена и, если они состоят из нескольких слов, удалять пробелы между словами и каждое слово начинать с большой буквы. Имя объекта является уникальным и служит для обращения к свойствам и методам объекта на встроенном языке.

Свойство *Синоним* также есть у любого объекта конфигурации. Оно предназначено для хранения альтернативного наименования объекта конфигурации, которое будет использовано в элементах интерфейса нашей программы, то есть будет показано пользователю. Поэтому на синоним практически нет никаких ограничений, и его можно задавать в привычном для человека виде.

Картинка подсистемы

В целях усовершенствования интерфейса приложения мы можем также задать картинку для отображения подсистемы.

Нажмем кнопку выбора в поле *Картинка* (см. рис. 2.4). В окне выбора картинки добавим картинку в список на закладке *Из конфигурации*. Для этого нажмем кнопку *Добавить* (рис. 2.5).

Рис. 2.5. Выбор картинки для представления подсистемы

Система создаст объект конфигурации *Общая картинка* и откроет окно редактирования его свойств.

Дадим картинке имя *Бухгалтерия*. Чтобы задать саму картинку, нажмем кнопку *Выбрать из файла* (рис. 2.6).

Рис. 2.6. Окно редактирования объекта конфигурации «ОбщаяКартинка»

Далее на диске, прилагающемся к книге, выберем папку *Image*, содержащую картинки, и укажем нужный файл с изображением.

Для просмотра изображений установим флажок *Просмотр*.

Отметим файл *Бухгалтерия* и нажмем кнопку *Открыть* (рис. 2.7).

Рис. 2.7. Выбор картинки для представления подсистемы

Выбранная нами картинка появится в окне редактирования общей картинки.

Закроем окно редактирования объекта конфигурации Общая картинка и вернемся в окно выбора картинки для подсистемы *Бухгалтерия*. Мы видим, что в списке картинок на закладке *Из конфигурации* появилась добавленная нами картинка. Нажмем *ОК* (рис. 2.8).

Рис. 2.8. Выбор картинки для представления подсистемы

После наших действий в дереве объектов конфигурации в ветке *Общие*

картинки появилась картинка *Бухгалтерия*, которую мы можем редактировать и использовать в дальнейшем в нашей конфигурации (рис. 2.9).

Рис. 2.9. Общие картинки в дереве объектов конфигурации

Итак, мы вернулись в окно редактирования объекта конфигурации *Подсистема Бухгалтерия*. Мы видим, что выбранная нами одноименная картинка установилась в качестве картинки для подсистемы (рис. 2.10).

Рис. 2.10. Общие картинки в дереве объектов конфигурации

Таким образом, в интерфейсе «1С:Предприятия» в качестве названия раздела будет показан синоним подсистемы, и над ним будет выводиться указанная картинка.

Отсутствие картинки у подсистемы не препятствует отображению раздела в интерфейсе. В этом случае рядом с названием раздела отображается стандартная картинка по умолчанию.

Снова выделим ветвь *Подсистемы*, нажмем кнопку *Добавить* в дереве объектов конфигурации и создадим подсистемы с именами *УчетМатериалов* и *ОказаниеУслуг*. Установим для них в качестве картинок соответственно общие

картинки *Материалы* и *Услуги*, добавив их из файлов *Материалы* и *Услуги* так же, как мы это делали для подсистемы *Бухгалтерия*.

Теперь воспользуемся другим способом для добавления подсистем. Вызовем контекстное меню одной из созданных подсистем. Выберем в нем пункт *Добавить*. Он разбивается на два подпункта. Выбор подпункта *Подсистема* позволяет добавить подсистему того же уровня иерархии, что и выделенная. Выбор подпункта *Подчиненная Подсистема* позволяет добавить подсистему, подчиненную выделенной (рис. 2.11).

Рис. 2.11. Добавление новой подсистемы в дерево объектов конфигурации

Поскольку в нашей конфигурации не планируется сложной многоуровневой структуры, выберем первый вариант и добавим подсистему *РасчетЗарплаты*. Установим для нее в качестве картинки общую картинку *Зарплата*, добавив ее из файла *Зарплата*.

В заключение добавим подсистему *Предприятие* для доступа к административным и сервисным функциям.

Панель разделов прикладного решения

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и увидим результат наших изменений. Вид разрабатываемого нами приложения изменился (рис. 2.12).

Рис. 2.12. «1С:Предприятие»

Сразу под главным меню располагается *панель разделов* приложения, где и отражены созданные нами подсистемы. Причем все разделы выводятся с выбранными в их свойствах картинками.

Разделы представлены в форме гиперссылок, нажав на которые пользователь может открыть связанные с ними документы, справочники, отчеты и т. п. Сейчас состав разделов пуст, так как мы еще не создали наполняющих их объектов конфигурации.

ПРИМЕЧАНИЕ

Обратите внимание, что раздел Рабочий стол формируется платформой по умолчанию. Он предназначен для размещения наиболее часто используемых пользователем документов, отчетов и т. п.

Порядок разделов

В режиме «Конфигуратор»

Однако порядок расположения подсистем нас не совсем устраивает. Изменим его.

Закроем приложение и вернемся в конфигуратор. Выделим корень дерева объектов конфигурации *ПособиеДляНачинающих*, нажатием правой кнопки мыши вызовем контекстное меню и выберем пункт *Открыть командный интерфейс конфигурации* (рис. 2.13).

Рис. 2.13. Вызов окна настройки командного интерфейса конфигурации

В открывшемся окне *Командный интерфейс* вы увидите список созданных вами подсистем (разделов приложения). С помощью кнопок *Вверх*, *Вниз* изменим порядок расположения разделов в этом списке.

Расположим сначала подсистемы, отражающие производственную деятельность нашей фирмы: *Учет материалов* и *Оказание услуг*, затем бухгалтерскую деятельность и расчет зарплаты сотрудников: *Бухгалтерия* и *Расчет зарплаты*, а затем подсистему *Предприятие* (рис. 2.14).

Рис. 2.14. Окно настройки подсистем

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и увидим, что порядок расположения подсистем в панели разделов приложения изменился так, как мы его задали (рис. 2.15).

Рис. 2.15. «1С:Предприятие»

Закроем приложение и вернемся в конфигуратор. На следующем занятии мы начнем создавать первые объекты конфигурации, привязывать их к различным подсистемам и продемонстрируем их конкретное применение в интерфейсе «1С:Предприятия».

ВНИМАНИЕ!

После успешного завершения каждого занятия мы рекомендуем сохранять конфигурацию, выполнив команду главного меню ***Администрирование > Выгрузить информационную базу****. Это полезно на случай, если вы запутаетесь в своих действиях и захотите вернуться к работающему варианту. Это можно сделать, выполнив команду* ***Администрирование > Загрузить информационную базу****.*

«Теория». Окно редактирования объекта конфигурации и

палитра свойств

На первый взгляд окно редактирования объекта и палитра свойств дублируют друг друга. В самом деле в палитре свойств отображены все свойства объекта конфигурации. Зачем было создавать еще и окно редактирования объекта? И если существует окно редактирования объекта, то зачем тогда палитра свойств, которая содержит все то же самое, только в другом виде?

Окно редактирования объекта конфигурации предназначено в первую очередь для быстрого создания новых объектов. Быстрое создание подразумевает ввод исчерпывающей информации об объекте. Значит, нужно очень хорошо знать структуру объекта, а на это требуется время… Выходит, быстро создать объект не получится?

Получится! Окно редактирования объекта имеет в своей основе механизм «мастеров»: разработчику в нужной последовательности предлагается ввести необходимые данные. Последовательность ввода данных разработана таким образом, чтобы предыдущие данные могли служить основой для ввода последующих. Движение управляется кнопками *Далее* и *Назад*. На каждом шаге предлагается ввести группу логически связанных между собой данных.

Но, предположим, вы уже освоились со структурой объектов, или вам просто нужно изменить несколько свойств объекта. Чтобы при этом опять не

«прокручивать» все с самого начала, окно редактирования объекта содержит закладки, позволяющие вам перейти непосредственно к тому шагу, на котором вводятся интересующие вас данные. Таким образом, окно редактирования объекта помогает быстро создать незнакомый объект конфигурации и в то же время обеспечивает удобный доступ к нужным свойствам при редактировании существующих объектов.

Что же касается палитры свойств, то она предоставляет одну абсолютно незаменимую возможность. Дело в том, что она не привязана по своей структуре к какому-то конкретному виду объектов конфигурации. Ее содержимое меняется в зависимости от того, какой объект является текущим. За счет этого она может запоминать, какое свойство объекта в ней выбрано, и при переходе в дереве к другому объекту будет подсвечивать у себя все то же свойство, но уже другого объекта.

Такая способность палитры свойств абсолютно незаменима, когда, например, среди трех десятков справочников конфигурации вам нужно быстро найти подчиненные какому-нибудь другому. В этом случае вы выбираете мышью в палитре свойств свойство *Владелец* любого справочника, затем переходите в дерево объектов конфигурации и просто пробегаете его при помощи стрелок *?* или *?*.

Контрольные вопросы

- *Для чего используется объект конфигурации «Подсистема».*
- *Как описать логическую структуру конфигурации при помощи объектов «Подсистема».*
- *Как управлять порядком вывода и отображением подсистем в конфигурации.*
- *Что такое окно редактирования объекта конфигурации и в чем его отличие от палитры свойств.*

Занятие 3 (2:10). Справочники

Продолжительность

Ориентировочная продолжительность занятия – 2 часа 10 минут.

На этом занятии мы познакомимся с объектом конфигурации *Справочник*. Вы узнаете, для чего используется этот объект, какова его структура и какими основными свойствами он обладает.

На практических примерах вы научитесь создавать справочники, описывать наиболее важные элементы их структуры и заполнять их данными.

Кроме этого, вы узнаете еще об одном объекте конфигурации – *Форма*. Узнаете, какие виды форм существуют у объекта конфигурации Справочник, и в каких ситуациях они используются.

В заключение в конце занятия будет сделано теоретическое отступление, касающееся механизма внесения изменений в конфигурацию.

Что такое справочник

Объект конфигурации *Справочник* предназначен для работы со списками данных. Как правило, в работе любой фирмы используются списки сотрудников,

списки товаров, списки клиентов, поставщиков и т. д. Свойства и структура этих списков описываются в объектах конфигурации Справочник, на основе которых платформа создает в базе данных таблицы для хранения информации из этих справочников.

Справочник состоит из *элементов*. Например, для справочника сотрудники элементом является сотрудник, для справочника товаров – товар и т. д. Пользователь в процессе работы может самостоятельно добавлять новые элементы в справочник: например, добавить новых сотрудников, создать новый товар или внести нового клиента.

В базе данных каждый элемент справочника представляет собой отдельную запись в основной таблице, хранящей информацию из этого справочника (рис. 3.1).

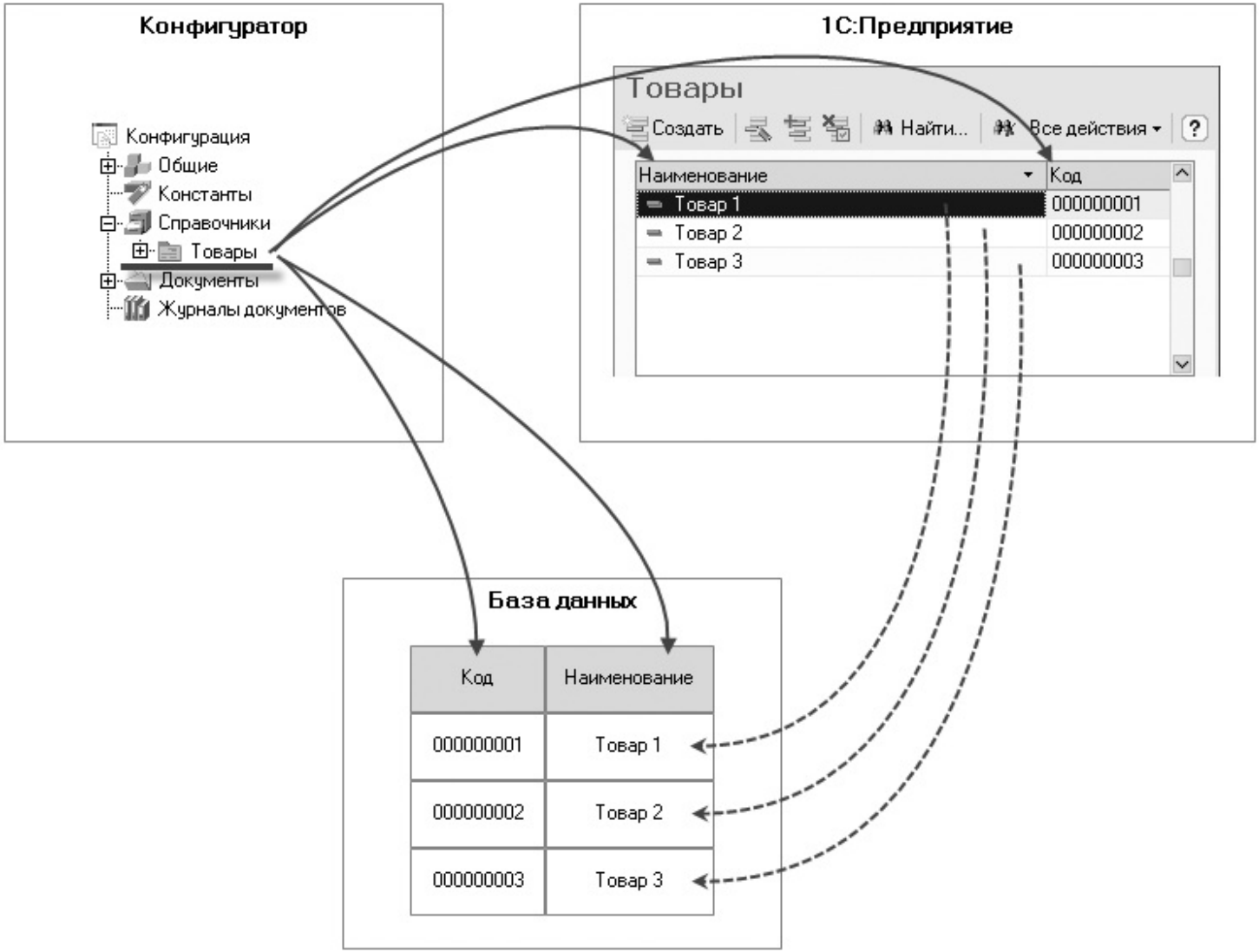

Конфигуратор
Конфигурация
Общие
Константы
Справочники
Товары
Документы
Журналы документов
1С:Предприятие
Товары
Создать
Найти...
Все действия
Наименование
Код
Товар 1
000000001
Товар 2
000000002
Товар 3
000000003
База данных
Код
Наименование
000000001
Товар 1
000000002
Товар 2
000000003
Товар 3

Рис. 3.1. Справочник «Товары» в режиме «Конфигуратор», в режиме «1С:Предприятие» и в базе данных

Каждый элемент справочника, как правило, содержит некоторую дополнительную информацию, которая подробнее описывает этот элемент. Например, все элементы справочника *Товары* могут содержать дополнительную информацию о производителе, сроке годности и др. Набор такой информации является одинаковым для всех элементов справочника, и для описания такого набора используются *реквизиты* объекта конфигурации *Справочник*, которые также, в свою очередь, являются объектами конфигурации (рис. 3.2).

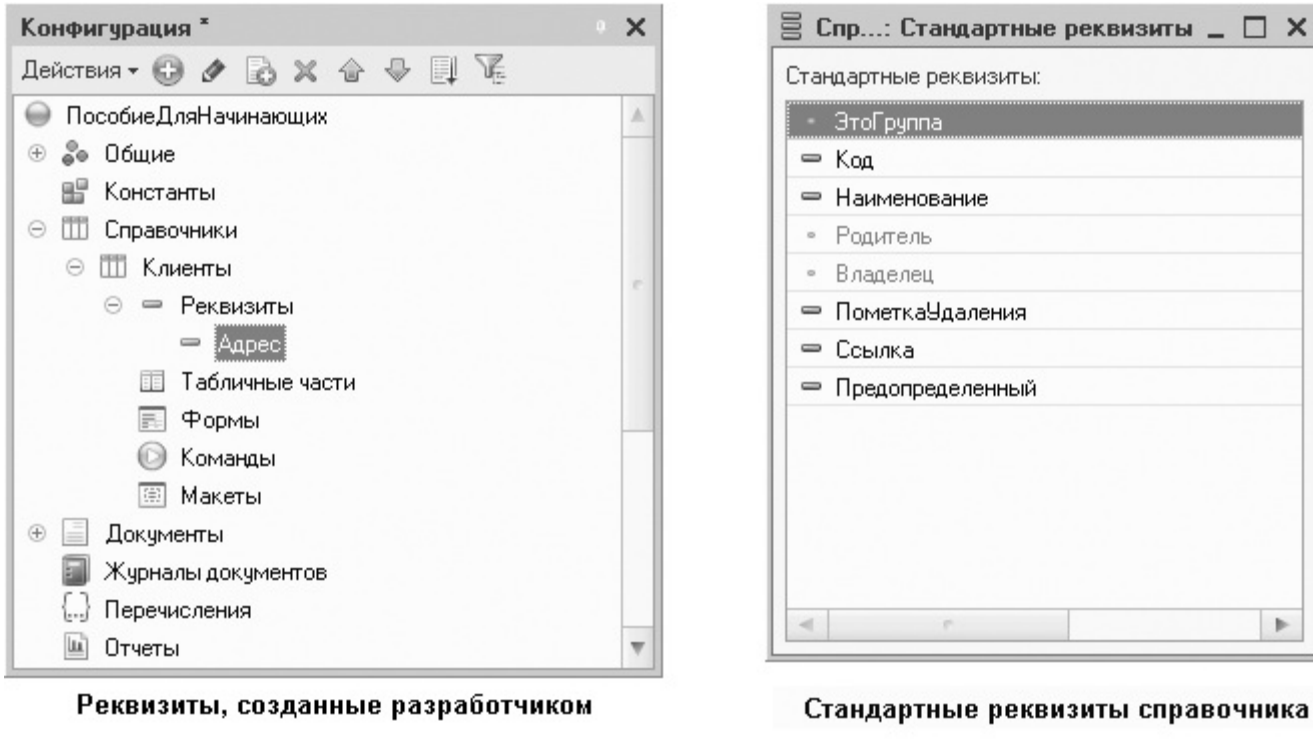


Рис. 3.2. Стандартные реквизиты справочника и реквизиты, созданные разработчиком

Поскольку эти объекты конфигурации логически связаны с объектом Справочник, они называются *подчиненными* этому объекту.

Большинство реквизитов разработчик создает самостоятельно, однако у каждого объекта конфигурации Справочник по умолчанию существует набор

стандартных реквизитов: *Код* и *Наименование* и пр. (см. рис. 3.2). Причем доступность стандартных реквизитов зависит от свойств справочника.

Например, если справочник иерархический, у него будет доступен стандартный реквизит *Родитель*. Если справочник подчинен другому объекту конфигурации, у него будет доступен реквизит *Владелец*. Если установить длину стандартного реквизита *Код* равной нулю, то у справочника будет недоступен этот реквизит. То же самое относится к реквизиту *Наименование*. Однако как минимум либо *Код*, либо *Наименование* должны присутствовать в реквизитах справочника, иначе такой справочник не имеет смысла.

Таким образом, в базе данных справочник хранится в виде таблицы, в строках которой расположены элементы списка, а каждому реквизиту (стандартному или созданному разработчиком) в этой таблице соответствует отдельный столбец. Соответственно, в ячейках этой таблицы хранится значение конкретного реквизита для конкретного элемента справочника (рис. 3.3).

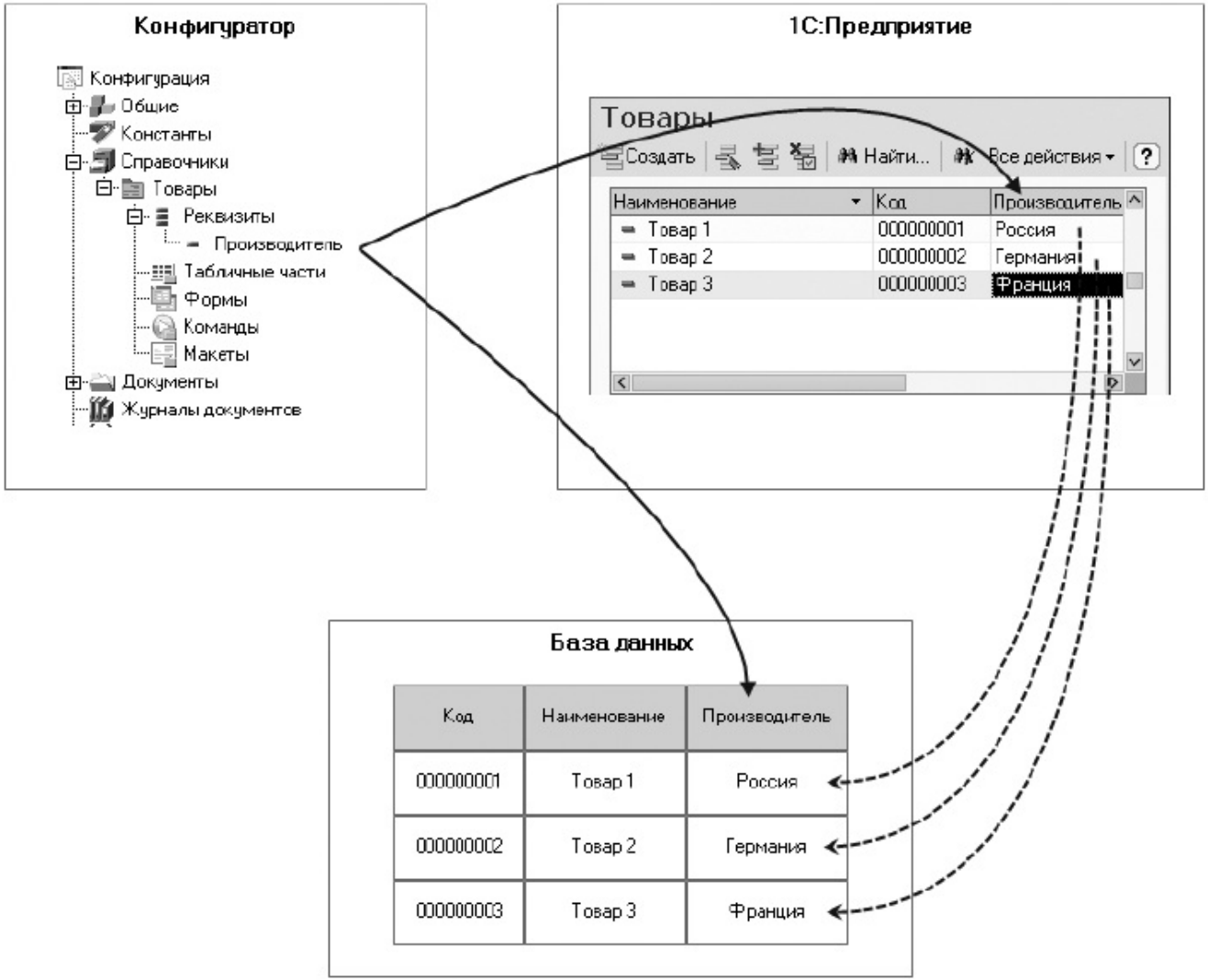

Конфигуратор
Конфигурация
Общие
Константы
Справочники
Товары
Реквизиты
Производитель
Табличные части
Формы
Команды
Макеты
Документы
Журналы документов
1С:Предприятие
Товары
Создать
Найти...
Все действия
Наименование
Код
Производитель
Товар 1
000000001
Россия
Товар 2
000000002
Германия
Товар 3
000000003
Франция
База данных
Код
Наименование
Производитель
000000001
Товар 1
Россия
000000002
Товар 2
Германия
000000003
Товар 3
Франция

Рис. 3.3. Справочник «Товары» в режиме «Конфигуратор», в режиме «1С:Предприятие» и в базе данных

Кроме этого, каждый элемент справочника может содержать некоторый набор информации, которая одинакова по своей структуре, но различна по количеству и предназначена для разных элементов справочника.

Так, например, каждый элемент справочника *Сотрудники* может содержать информацию о составе семьи сотрудника. Для одного сотрудника это будет только супруга, а у другого семья может состоять из супруги, сына и дочери.

Для описания подобной информации могут быть использованы *табличные части* объекта конфигурации *Справочник*, являющиеся подчиненными ему объектами конфигурации. В этом случае в базе данных будут созданы дополнительные таблицы для хранения табличных частей, подчиненных конкретному элементу справочника (рис. 3.4).

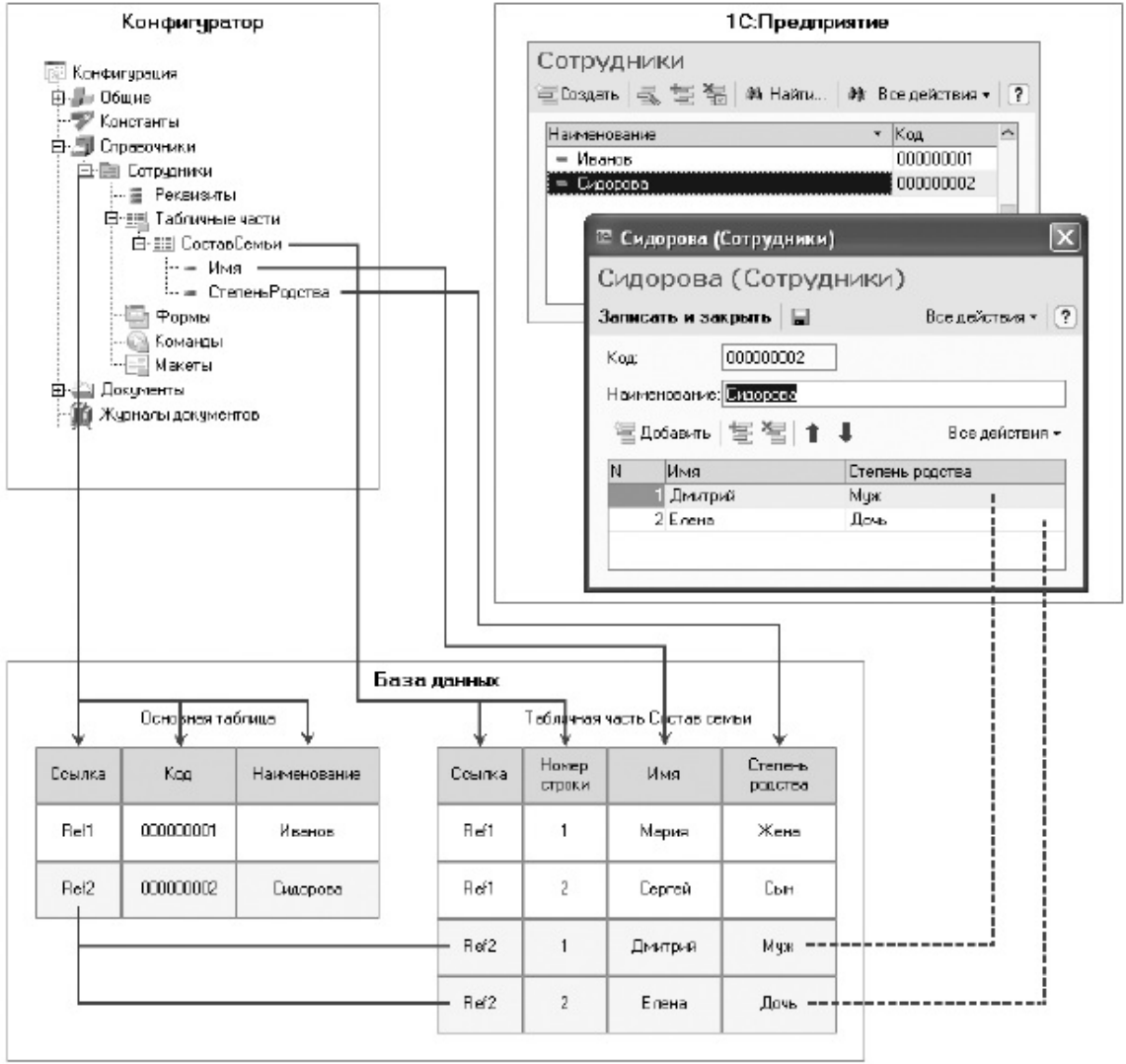

Конфигуратор
Конфигурация
Общие
Константы
Справочники
Сотрудники
Реквизиты
Табличные части
СоставСемьи
Имя
СтепеньРодства
Формы
Команды
Макеты
Документы
Журналы документов
1С:Предприятие
Сотрудники
Создать
Найти...
Все действия
Наименование
Код
Иванов
000000001
Сидорова
000000002
Сидорова (Сотрудники)
Сидорова (Сотрудники)
Записать и закрыть
Все действия
Код:
000000002
Наименование:
Сидорова
Добавить
Все действия
N
Имя
Степень родства
1
Дмитрий
Муж
2
Елена
Дочь
База данных
Основная таблица
Табличная часть Состав семьи
Ссылка
Код
Наименование
Ref1
000000001
Иванов
Ref2
000000002
Сидорова
Ссылка
Номер строки
Имя
Степень родства
Ref1
1
Мария
Жена
Ref1
2
Сергей
Сын
Ref2
1
Дмитрий
Муж
Ref2
2
Елена
Дочь

Рис. 3.4. Справочник «Товары» в режиме «Конфигуратор», в режиме «1С:Предприятие» и в базе данных

Причем система скрывает от разработчика всю «техническую» часть, связанную с хранением данных: в базе данных для справочника создаются несколько таблиц, эти таблицы связываются по уникальному полю (*Ссылка*), поля таблиц имеют определенные типы и т. д. Все это система делает сама. Нам лишь нужно добавить в объект конфигурации Справочник подчиненный ему объект *Табличная часть*.

Для удобства использования элементы справочника могут быть сгруппированы пользователем по какому-либо принципу.

Например, в справочнике *Бытовая техника* могут быть созданы следующие *группы*: *Холодильники*, *Телевизоры*, *Стиральные машины* и т. д. Возможность создания таких групп в справочнике задается свойством *Иерархический* объекта конфигурации *Справочник*. В этом случае элемент справочника, представляющий собой группу, будет являться *родителем* для всех элементов и групп, входящих в эту группу. Такой вид иерархии называется *иерархией групп и элементов* (рис. 3.5).

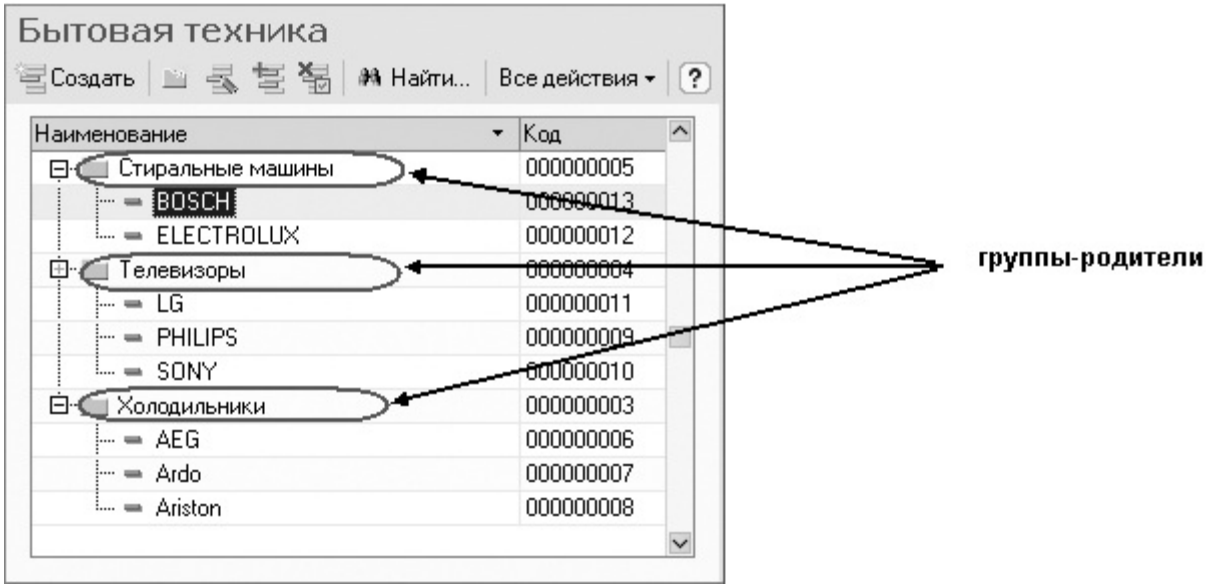


Рис. 3.5. Иерархический справочник с иерархией групп и элементов

Возможен и другой вид иерархии – *иерархия элементов*. В этом случае в качестве родителя выступает не группа элементов справочника, а непосредственно один из его элементов. Например, такой вид иерархии можно использовать при создании справочника *Подразделения*, когда одно подразделение является родителем для нескольких других, входящих в его состав (рис. 3.6).

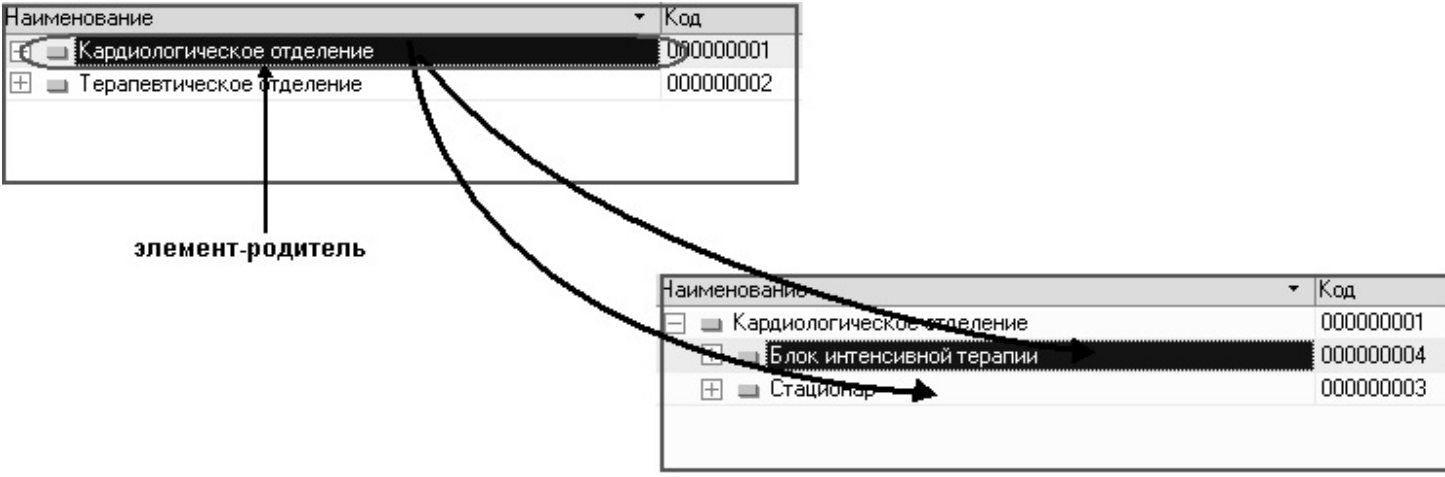


Рис. 3.6. Иерархический справочник с иерархией элементов

Элементы одного справочника могут быть *подчинены* элементам или группам другого справочника. Например, справочник *ЕдиницыИзмерения* может быть подчинен справочнику *Товары*. Тогда для каждого элемента справочника *Товары* мы сможем указать единицы измерения, в которых этот товар поступает на склад.

В системе «1С:Предприятие» это достигается путем указания списка *владельцев* справочника для каждого объекта конфигурации *Справочник*. В данном случае справочник *Товары* будет владельцем справочника *ЕдиницыИзмерения* (рис. 3.7).

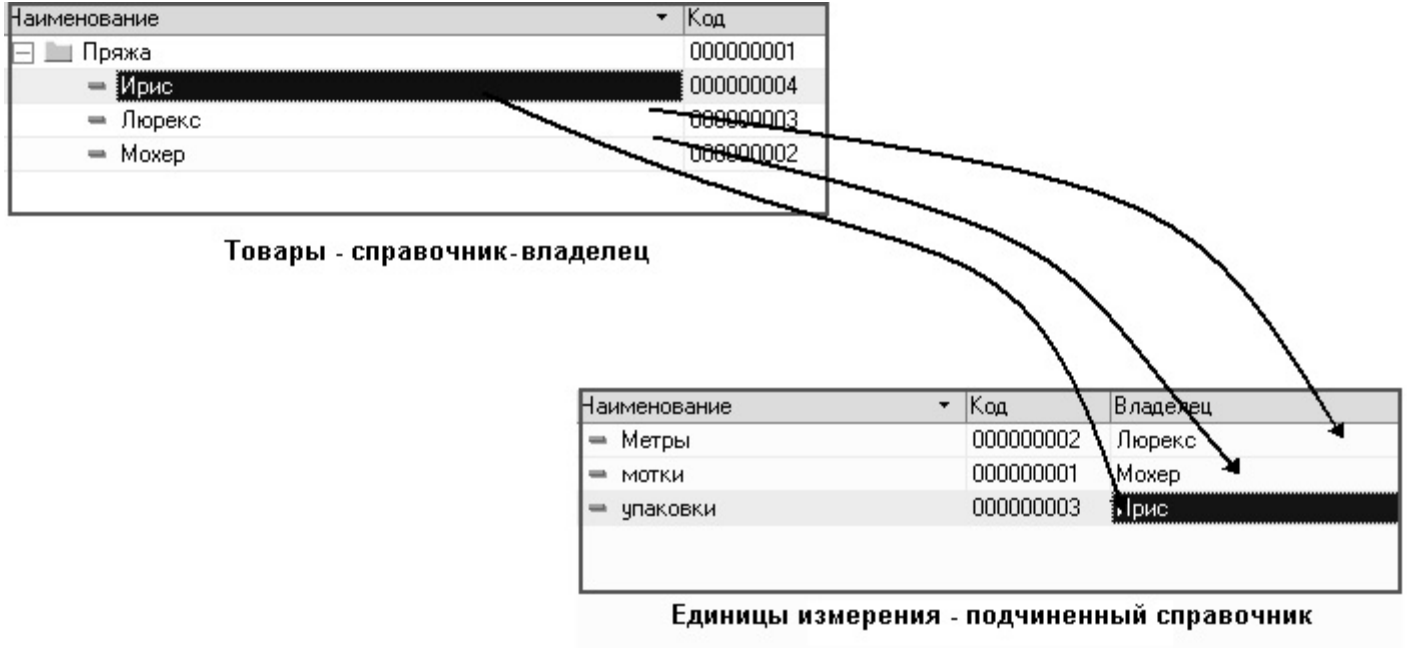


Рис. 3.7. Справочник «Товары» – владелец справочника «Единицы измерения»

Порой возникают ситуации, когда необходимо, чтобы в справочнике некоторые элементы существовали всегда, независимо от действий пользователя. Допустим, логика бизнес-процессов на предприятии такова, что все товары сначала поступают на основной склад, а затем по мере надобности перемещаются на другие склады. В этом случае в справочнике *Склады* всегда должен существовать склад *Основной*, иначе приходование товаров будет выполнено неправильно. Объект конфигурации *Справочник* позволяет описать любое количество таких элементов справочника. Они называются

предопределенными элементами справочника (рис. 3.8).

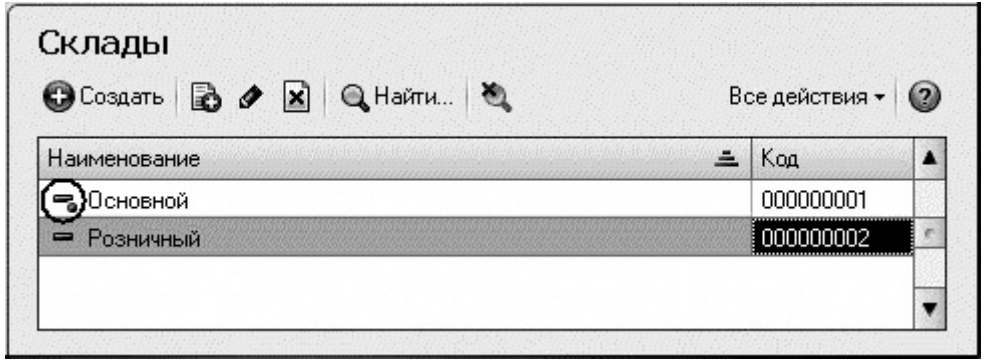


Рис. 3.8. Справочник «Склады» с предопределенным элементом «Основной»

Предопределенные элементы отличаются от обычных тем, что они создаются в конфигураторе и что пользователь не может их удалить. Все остальные действия с ними он делать может, в том числе и переименовывать. В интерфейсе предопределенные элементы справочника помечены специальной пиктограммой (см. рис. 3.8).

Формы справочника

В зависимости от того, какие действия мы хотим выполнять со справочником, нам требуется изображать справочник в «разном виде». Например, для того чтобы выбрать некоторый элемент справочника, удобнее представить справочник в виде списка, а для того чтобы изменить какой-то элемент

справочника, удобнее представить все реквизиты этого элемента справочника в одной форме (рис. 3.9).

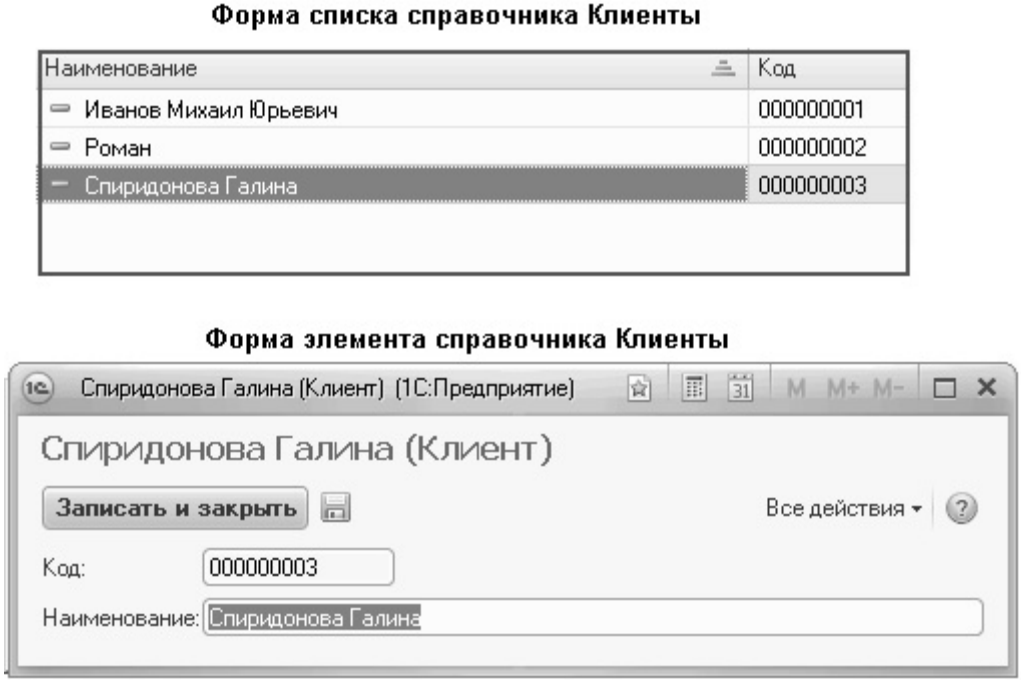


Рис. 3.9. Форма списка и форма редактирования элемента справочника «Клиенты»

Система может самостоятельно сгенерировать все формы, которые нужны для представления данных, содержащихся в справочнике. Причем система знает, какие именно формы нужно использовать в каких ситуациях.

Вообще говоря, для отображения справочника в различных ситуациях требуется максимум пять форм для справочника.

Обратите внимание, что в различных местах конфигуратора одни и те же формы называются немного по-разному (табл. 3.1). Следующая таблица представляет различные названия форм:

- в контекстном меню справочника (открыть основную форму…), в дереве конфигурации и в палитре свойств справочника;
- в конструкторе форм;
- на закладке *Формы* окна редактирования справочника.

Таблица 3.1. Формы справочника

В контекстном меню и в палитре свойств (рис. 3.12)	В конструкторе форм (рис. 3.11)	На закладке формы (рис. 3.10)
Форма объекта	Форма элемента справочника	Элемента
Форма группы	Форма группы справочника	Группы
Форма списка	Форма списка справочника	Списка

Форма для выбора	Форма выбора справочника	Выбора
Форма для выбора группы	Форма выбора группы справочника	Выбора группы

Дело в том, что в контекстном меню и палитре свойств отображаются свойства объектов конфигурации. Они одинаковые для всех объектов конфигурации. А в конструкторе форм и на закладке формы отображаются представления этих свойств в виде, более понятном разработчику. Они разные для разных объектов конфигурации (рис. 3.10, 3.11, 3.12).

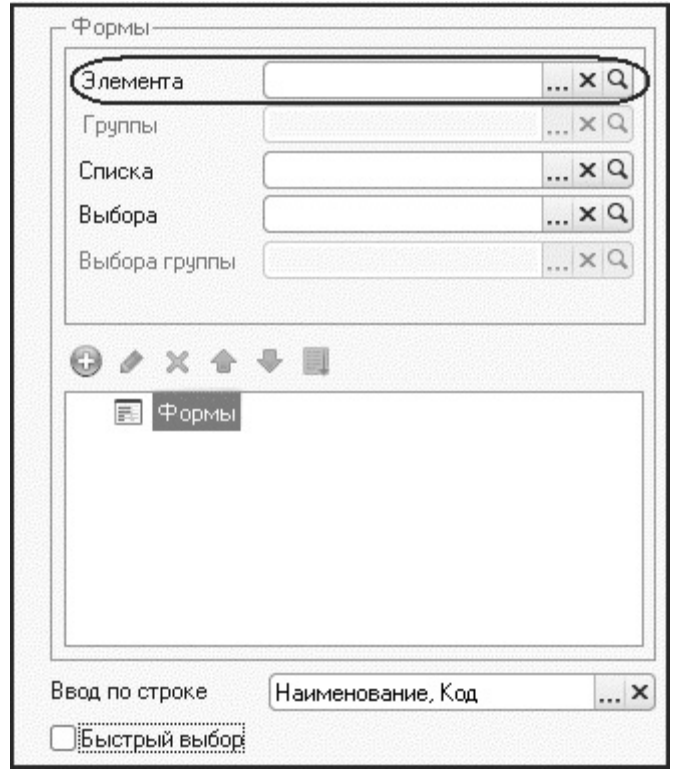


Рис. 3.10. Названия форм справочника на закладке «Формы»

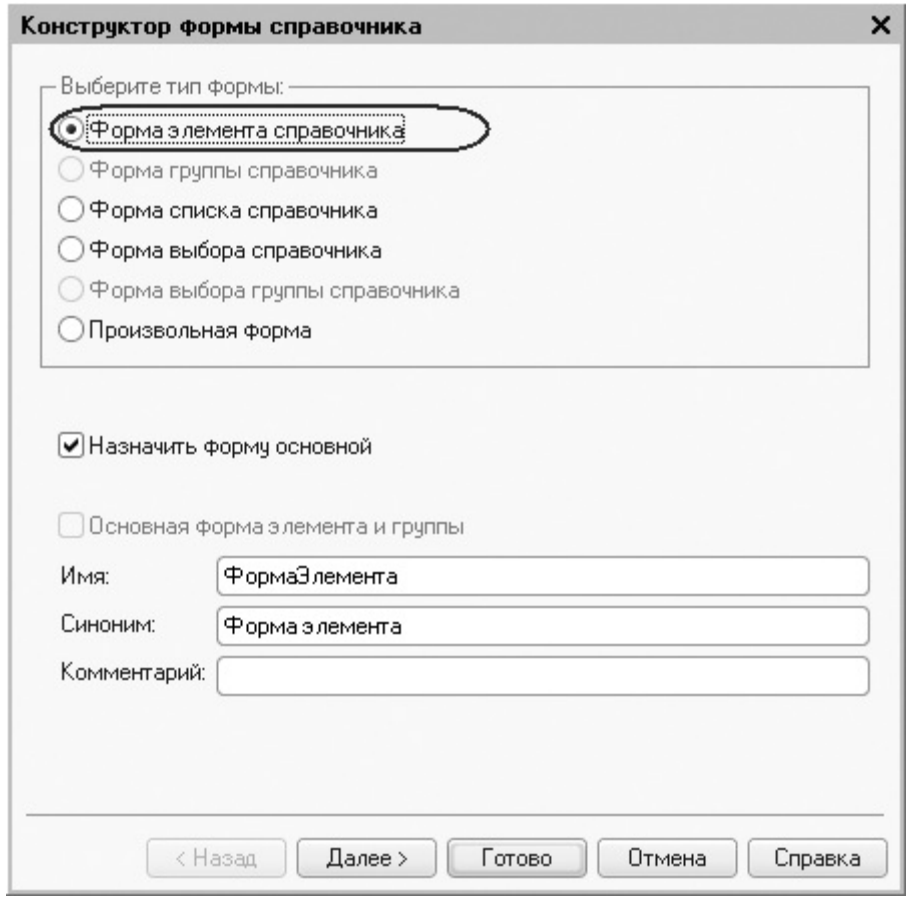


Рис. 3.11. Названия форм справочника в конструкторе форм

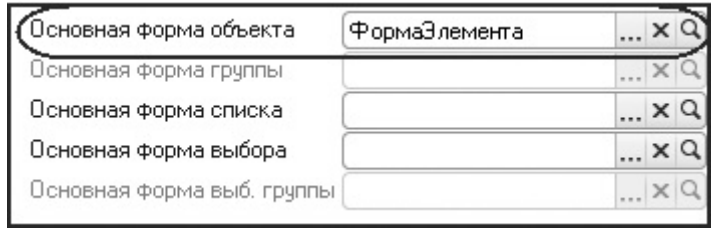


Рис. 3.12. Названия форм справочника в палитре свойств

Форма элемента используется для редактирования или создания элемента справочника.

Форма группы используется для редактирования или создания группы справочника. Группа, как правило, содержит гораздо меньше информации, чем сам элемент справочника. Поэтому для нее нужна отдельная форма, отличная от формы элемента (рис. 3.13).

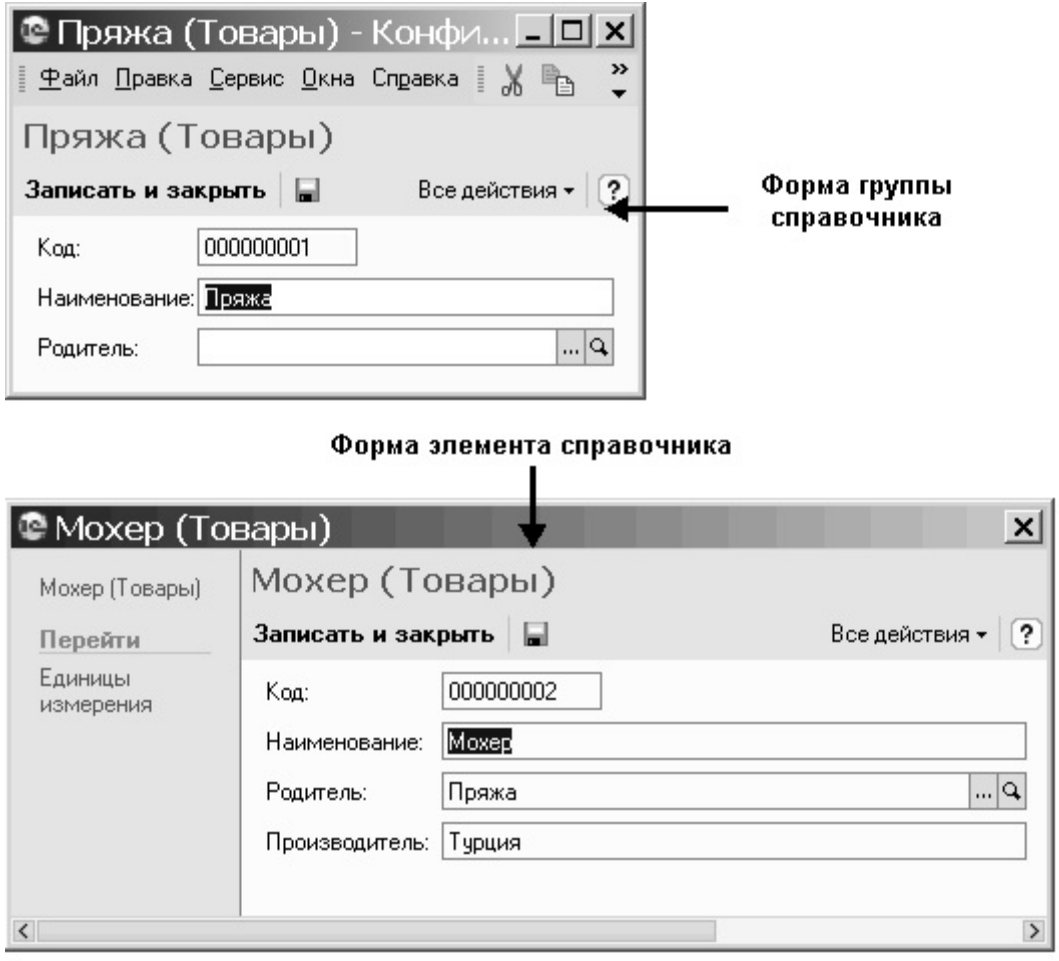


Рис. 3.13. Форма группы и форма элемента справочника

Форма списка используется для отображения списка элементов справочника.

Форма выбора используется для того, чтобы в поле некоторой формы выбрать один из элементов справочника. При этом форма выбора проще, чем форма списка, так как в форме списка может показываться много реквизитов. А при выборе элемента (в документе, например) нам нужно знать только наименование. Поэтому можно для выбора использовать отдельную более простую форму (рис. 3.14).

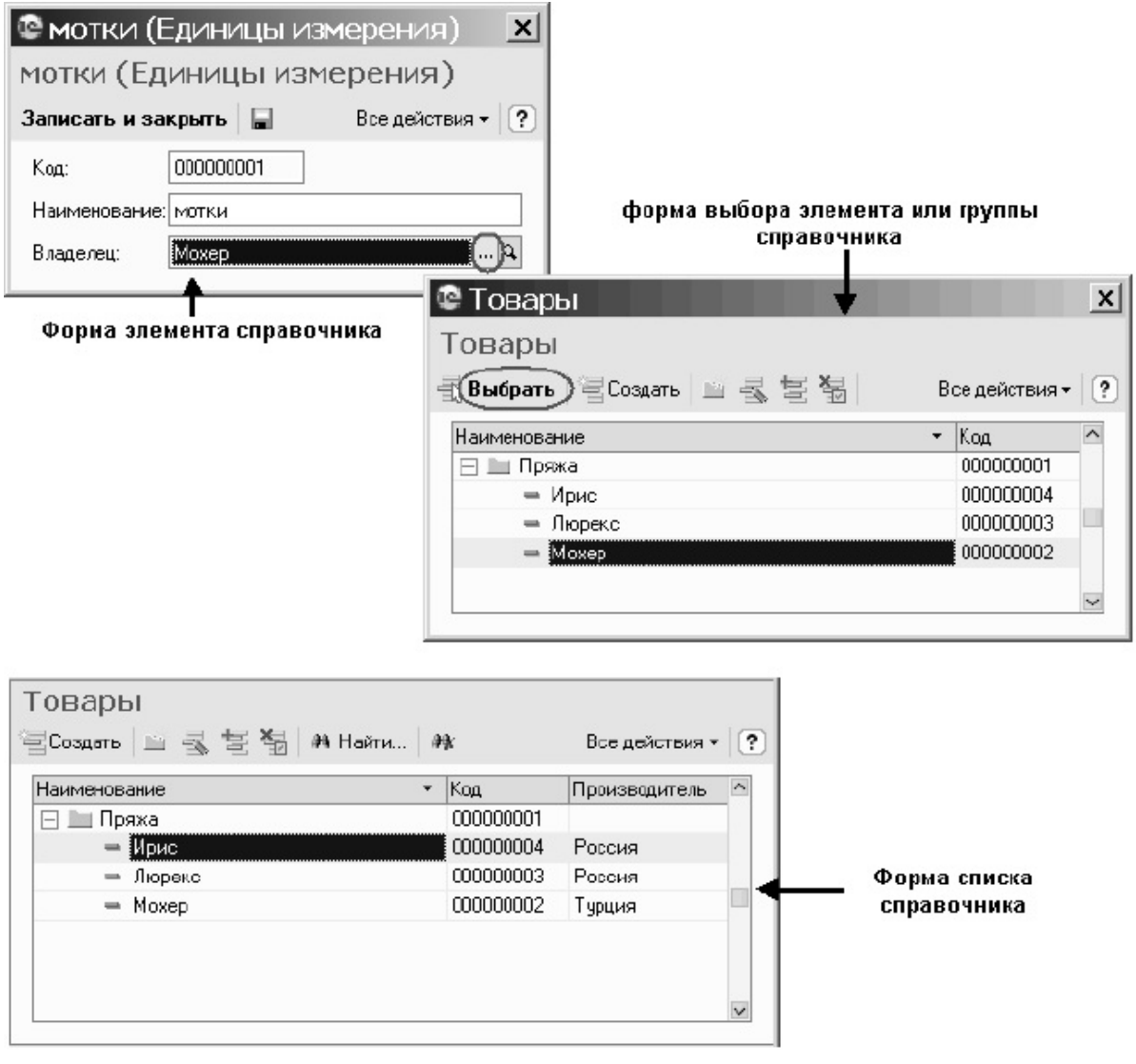

мотки (Единицы измерения)
мотки (Единицы измерения)
Записать и закрыть
Все действия
Код:
000000001
Наименование:
мотки
Владелец:
Мохер
Форма элемента справочника
форма выбора элемента или группы справочника
Товары
Товары
Выбрать
Создать
Все действия
Наименование
Код
Пряжа
000000001
Ирис
000000004
Люрекс
000000003
Мохер
000000002
Товары
Создать
Найти...
Все действия
Наименование
Код
Производитель
Пряжа
000000001
Ирис
000000004
Россия
Люрекс
000000003
Россия
Мохер
000000002
Турция
Форма списка справочника

Рис. 3.14. Форма выбора и форма списка справочника

Форма выбора группы используется, когда в поле некоторой формы нужно выбрать не просто элемент справочника, а одну из его групп. При этом форма выбора группы проще, чем форма выбора элемента, так как группа, как правило, содержит гораздо меньше информации, чем сам элемент справочника.

При этом для всех ссылочных объектов конфигурации (справочников, документов и т. д.) будет использоваться форма объекта, но нужно понимать, что под объектом здесь понимается объект информационной базы, то есть «элемент» того, что хранит этот объект конфигурации. Для справочника это будет элемент справочника, для документа – документ, для плана счетов – счет и т. д.

Любая форма может быть описана в конфигураторе. Для создания такого описания существует подчиненный объект конфигурации *Форма* (рис. 3.15).

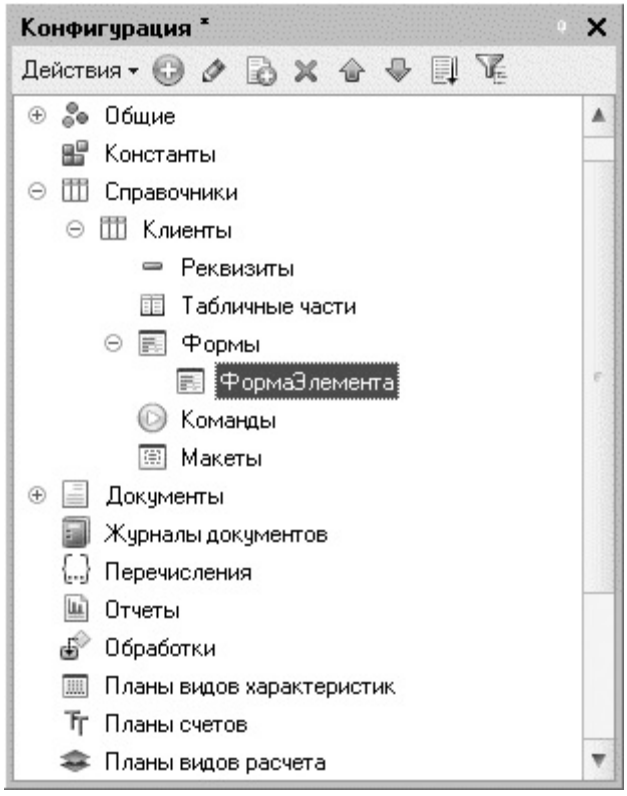


Рис. 3.15. Форма элемента справочника в конфигураторе

Как правило, объект конфигурации *Форма* подчинен одному из прикладных объектов, но может существовать и самостоятельно.

На основании описания, содержащегося в объекте конфигурации *Форма*, в нужный момент работы платформа «1С:Предприятие» создаст программный

объект *Форма*, с которым и будет работать пользователь.

Таким образом, форма служит для визуализации данных, находящихся в базе данных. Она представляет эти данные в удобном для пользователя виде и позволяет описать алгоритмы, которые будут сопровождать работу пользователя с данными, показанными в форме.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы со справочниками, можно прочитать в разделе «Краткий справочник разработчика. Справочники»*.*

Простой справочник

Теперь, когда мы немного познакомились с возможностями объекта конфигурации Справочник, создадим несколько таких объектов, чтобы описать справочники, которые будут использоваться в нашей базе данных.

Так как наше ООО «На все руки мастер» оказывает услуги по ремонту бытовой техники, очевидно, что для ведения учета нам потребуется хранить некоторую списочную информацию.

Для начала нам понадобится список сотрудников предприятия, которые будут оказывать услуги.

Затем нам будет нужен список клиентов, с которыми работает наше ООО «На все руки мастер».

После этого нам понадобится перечень услуг, которые может оказывать наше предприятие, и список материалов, которые могут быть израсходованы. Кроме этого, нам потребуется список складов, на которых могут находиться материалы ООО «На все руки мастер».

Начнем с простых вещей – списка сотрудников и списка клиентов. Сначала создадим справочник, в котором будут храниться наименования наших клиентов.

В режиме «Конфигуратор»

Откроем в конфигураторе нашу учебную конфигурацию, выделим в дереве объектов конфигурации ветвь *Справочники* и нажмем кнопку *Добавить* в командной панели окна конфигурации (рис. 3.16).

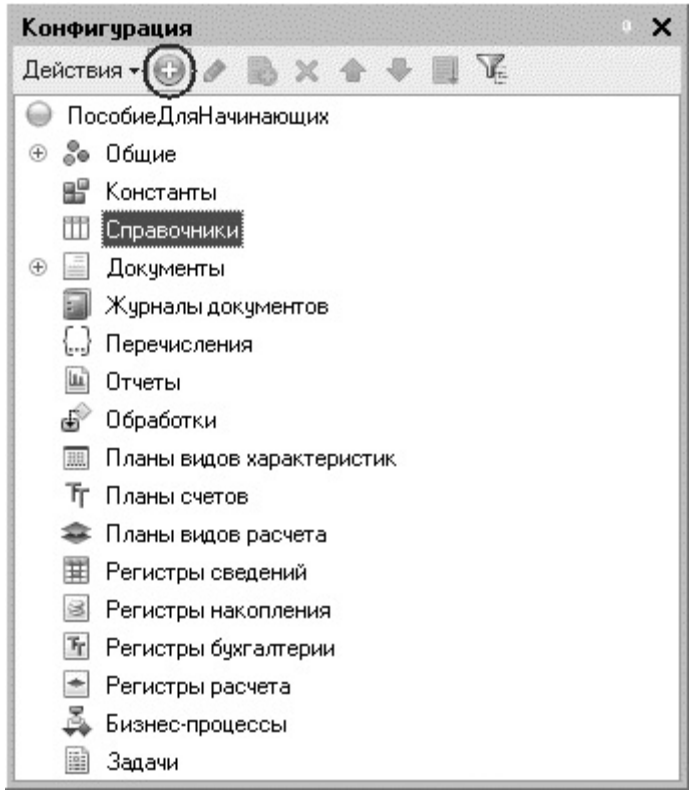


Рис. 3.16. Создание нового объекта конфигурации «Справочник»

В открывшемся окне редактирования объекта конфигурации зададим имя справочника – *Клиенты*. На основании имени платформа автоматически создаст синоним – *Клиенты*. Напомним, что свойство *Синоним* служит для представления объекта в интерфейсе нашей программы.

Также у разработчика есть возможность установки дополнительных свойств, определяющих пользовательское представление объектов. Эти свойства задавать не обязательно. Если они не заданы, то для представления объекта в интерфейсе «1С:Предприятия» используется синоним объекта конфигурации *Справочник*. Но, как мы увидим дальше, это не всегда хорошо.

Представления объекта конфигурации

Представление объекта определяет название объекта в единственном числе и используется в названии стандартной команды, например, команды создания объекта – *Клиент: создать*. Представление объекта нужно задавать тогда, когда синоним объекта конфигурации задан во множественном числе, или когда он описывает множество объектов, потому что в интерфейсе автоматически формируются команды открытия списка справочника и команды создания нового элемента справочника.

Если синоним задан во множественном числе, то для команды открытия списка это вполне подходит – *Клиенты*, то есть посмотреть всех клиентов. Но для команды создания элемента справочника – одного клиента – это неудачный вариант.

Для этой команды нужно задать представление в единственном числе – *Клиент*. Представление объекта как раз и используется для того, чтобы описать, как будет выглядеть в интерфейсе команда добавления нового

клиента. Также оно будет использовано в заголовке формы клиента (если не указано расширенное представление объекта) и в представлении ссылки на клиента.

Расширенное представление объекта определяет заголовок формы объекта, например формы для создания нового элемента справочника. Если это свойство не задано, то вместо него используется свойство *Представление объекта*.

Представление списка определяет название списка объектов и используется в названии стандартной команды, например, команды открытия списка объектов – *Клиенты: открыть*. Представление списка нужно задавать тогда, когда синоним задан в единственном числе.

Например, это часто бывает у документов (*Приходная накладная*). Тогда в представлении списка нужно указывать название объекта конфигурации во множественном числе (*Приходные накладные*).

Расширенное представление списка определяет заголовок формы списка, например формы списка справочника. Если это свойство не задано, то вместо него используется свойство *Представление списка*.

Зададим два свойства *Представление объекта – Клиент* и *Представление*

списка – *Клиенты*. Последнее можно было и не задавать, так как синоним справочника совпадает со свойством *Представление списка* (рис. 3.17).

В представлении списка вроде бы подразумевается название *Список клиентов*, но идущие подряд строки *Список сотрудников, Список клиентов, Список складов* не очень хорошо смотрятся в интерфейсе приложения.

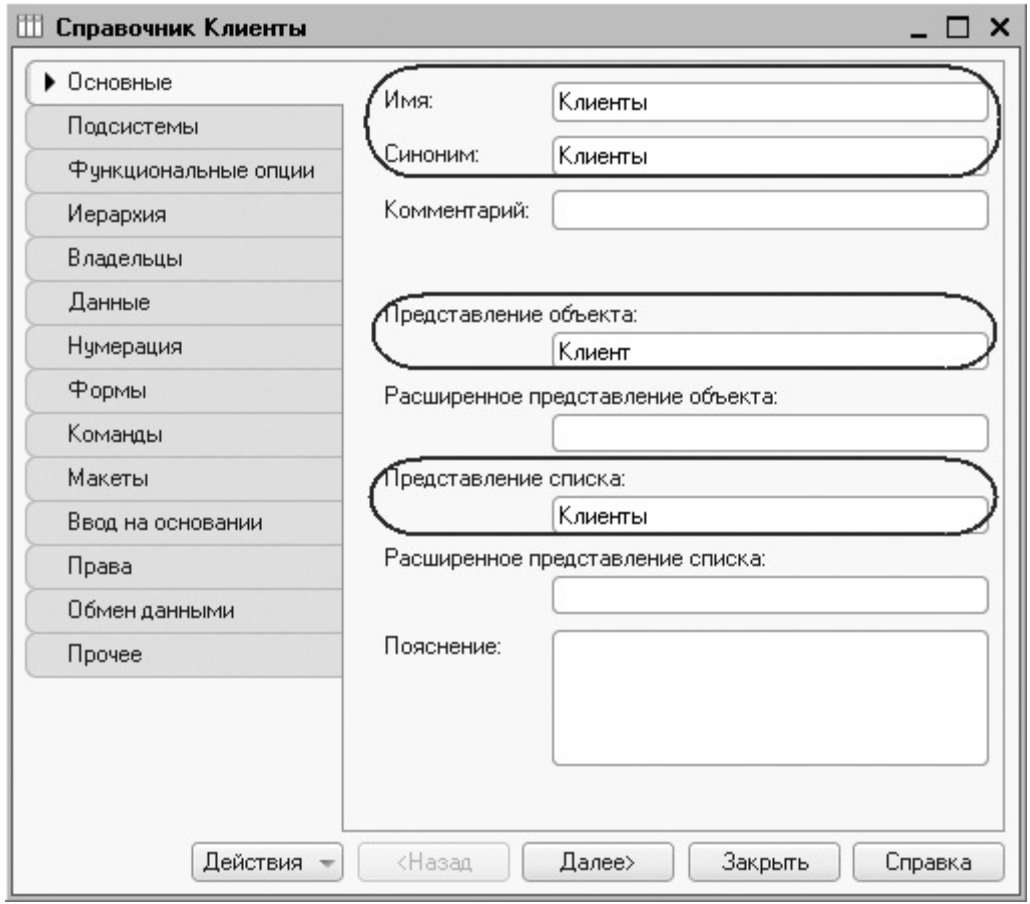


Рис. 3.17. Установка основных свойств справочника

Принадлежность объекта к подсистемам

Нажмем кнопку *Далее* и перейдем на закладку *Подсистемы* окна

редактирования объекта конфигурации *Справочник*. На этой закладке определяется, в каких подсистемах будет отображаться данный справочник.

В списке подсистем мы видим подсистемы, созданные нами ранее при определении структуры приложения. Логично предположить, что список клиентов должен быть доступен в разделе *Оказание услуг*, так как оказываемые услуги относятся к определенному клиенту. Бухгалтерская отчетность, формируемая в разделе *Бухгалтерия,* также может быть представлена в разрезе клиентов.

Поэтому отметим в списке подсистемы *Бухгалтерия* и *ОказаниеУслуг* (рис. 3.18).

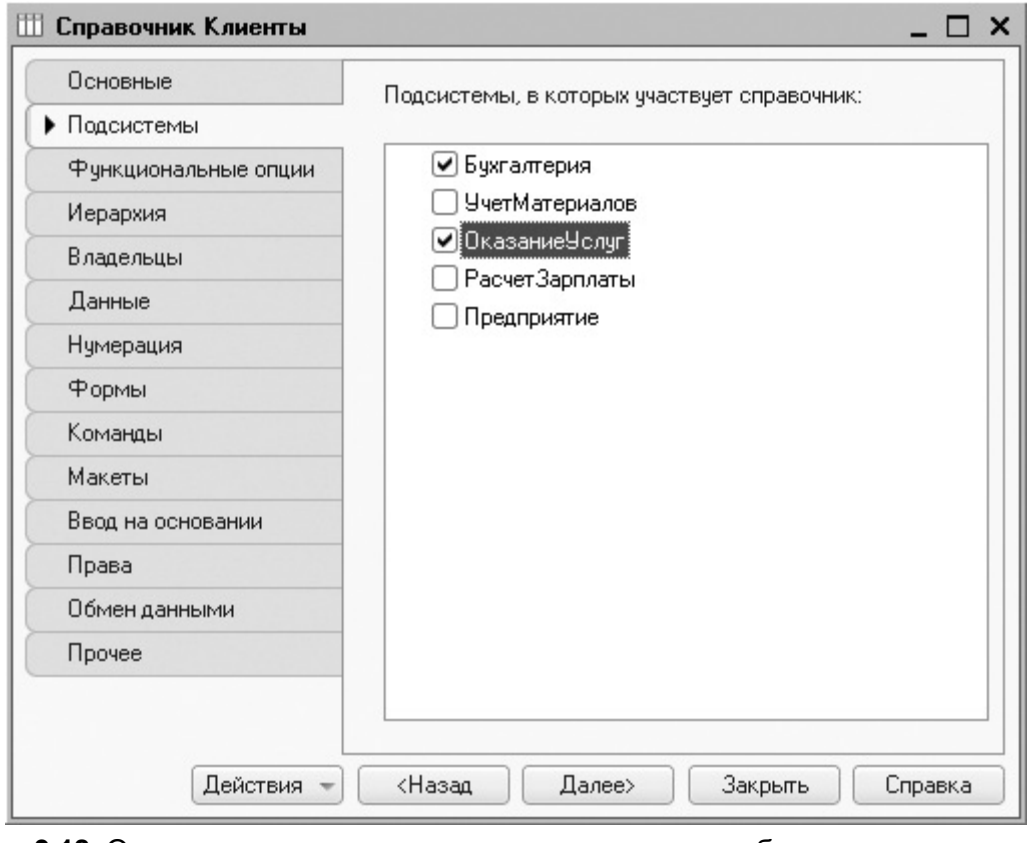


Рис. 3.18. Определение списка подсистем, в которых отображается справочник

Теперь откроем окно редактирования одной из отмеченных подсистем, например *Бухгалтерия,* и перейдем на закладку *Состав*. Мы видим, что в составе объектов этой подсистемы появился новый объект конфигурации *СправочникКлиенты* (рис. 3.19).

Обратите внимание, что на закладке *Состав* также можно изменять список объектов, входящих в подсистему.

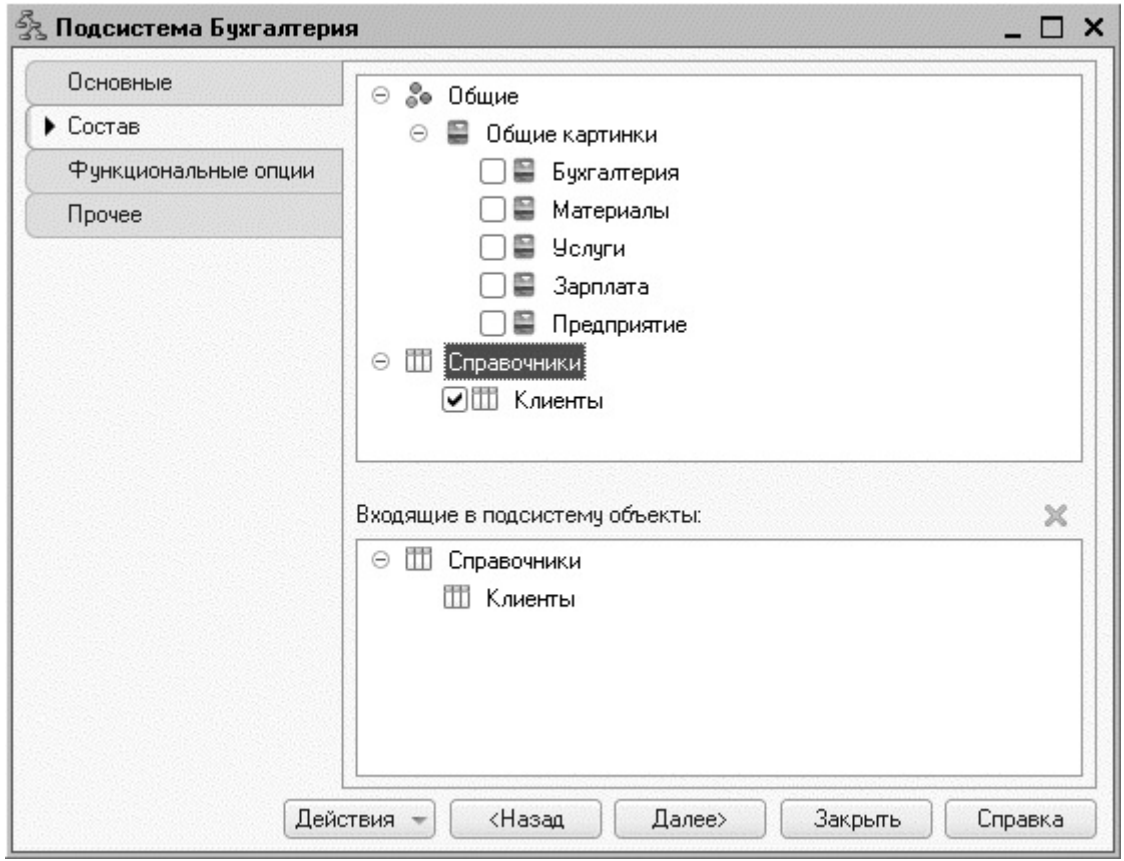


Рис. 3.19. Состав объектов, входящих в подсистему

Код и наименование справочника

Теперь вернемся к окну редактирования объекта конфигурации Справочник и нажмем на закладку *Данные*.

Здесь для нас представляют интерес длина кода и длина наименования.

Длина кода – важное свойство справочника. Как правило, код справочника используется для идентификации элементов справочника и содержит уникальные для каждого элемента справочника значения. Платформа может сама контролировать уникальность кодов и поддерживать автоматическую нумерацию элементов справочника. Поэтому от длины кода будет зависеть количество элементов, содержащихся в справочнике.

Длина кода – 9 символов. В результате мы сможем использовать коды от 1 до 999999999 – этого вполне достаточно для нашего небольшого ООО «На все руки мастер».

Перейдем к длине наименования. 25 символов для нас явно мало, увеличим длину наименования до 50 (рис. 3.20).

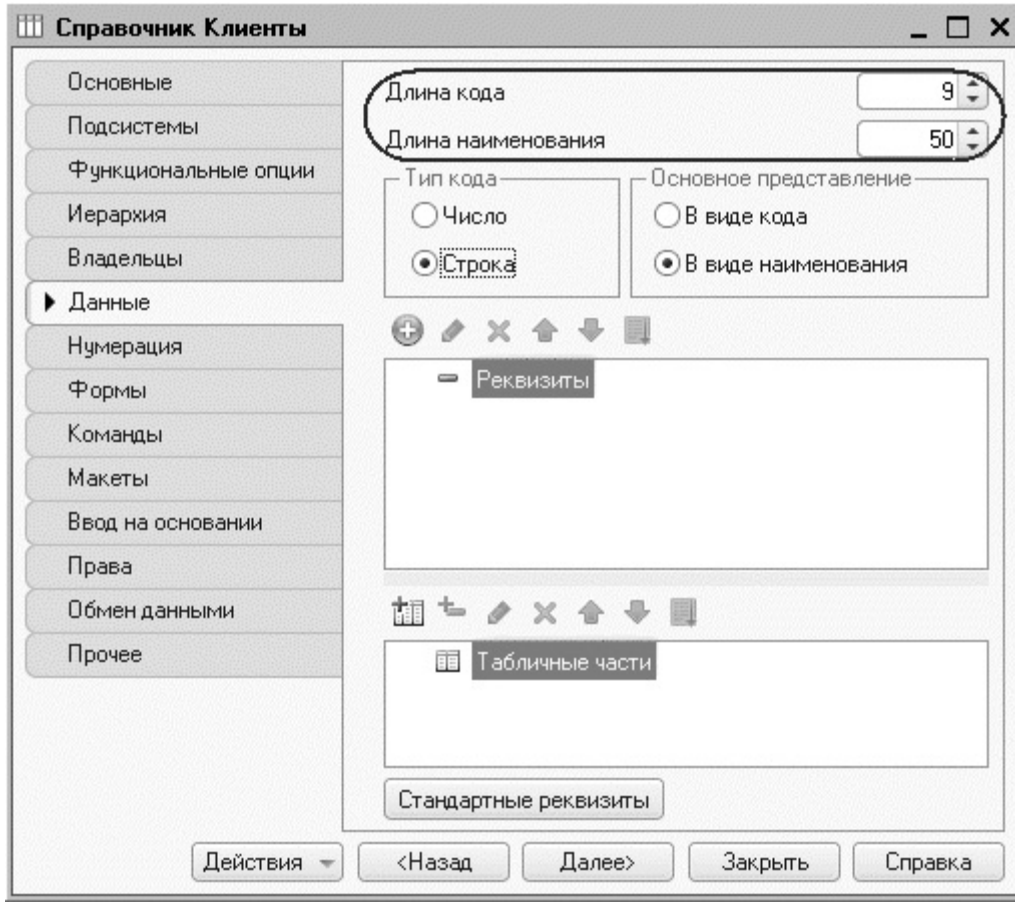


Рис. 3.20. Установка длины кода и наименования справочника

Команда добавления нового элемента

Прежде чем запускать «1С:Предприятие», настроим интерфейс приложения,

чтобы нам было удобнее вводить новые элементы справочника.

Дело в том, что для размещения стандартных команд открытия списков и создания новых объектов конфигурации в интерфейсе «1С:Предприятия» существует общий стандартный алгоритм, который мы сейчас объясним на примере справочников. Но это справедливо и для документов, планов счетов и т. п.

Команда для открытия списка справочника, как и команда для создания его новых элементов, добавляется в интерфейс тех разделов (подсистем), в которых будет отображаться справочник. Но команда создания новых элементов по умолчанию невидима в интерфейсе приложения.

Это объясняется тем, что возможность просматривать списки справочника нужна, как правило, всегда. А возможность создания новых элементов справочника используется не так часто. Поэтому соответствующую команду следует включать только для тех справочников (объектов конфигурации), создание новых элементов которых является основной деятельностью для пользователей в данном разделе прикладного решения.

Сделаем доступной в панели действий раздела *ОказаниеУслуг* стандартную команду для создания новых клиентов.

Для этого в дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы* (рис. 3.21).

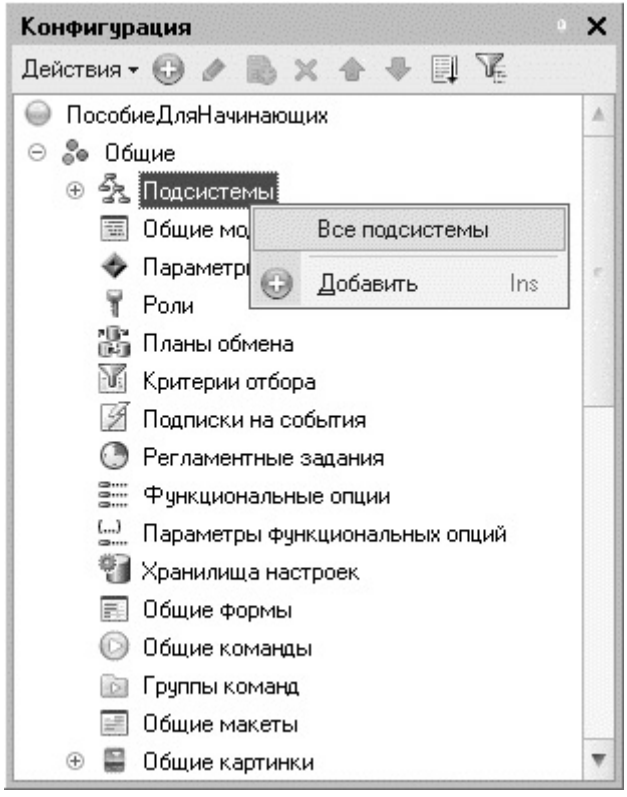


Рис. 3.21. Вызов окна настройки подсистем

В открывшемся окне *Все подсистемы* слева в списке *Подсистемы* выделим подсистему *ОказаниеУслуг*. Справа в списке *Командный интерфейс*

отразятся все команды выбранной подсистемы.

При создании справочника в группу *Панель навигации.Обычное* добавилась команда *Клиенты* для открытия этого списка. Она включена по умолчанию. В группу *Панель действий.Создать* добавилась команда *Клиент: создать* для создания нового элемента справочника, но она невидима по умолчанию.

Включим видимость у этой команды (рис. 3.22).

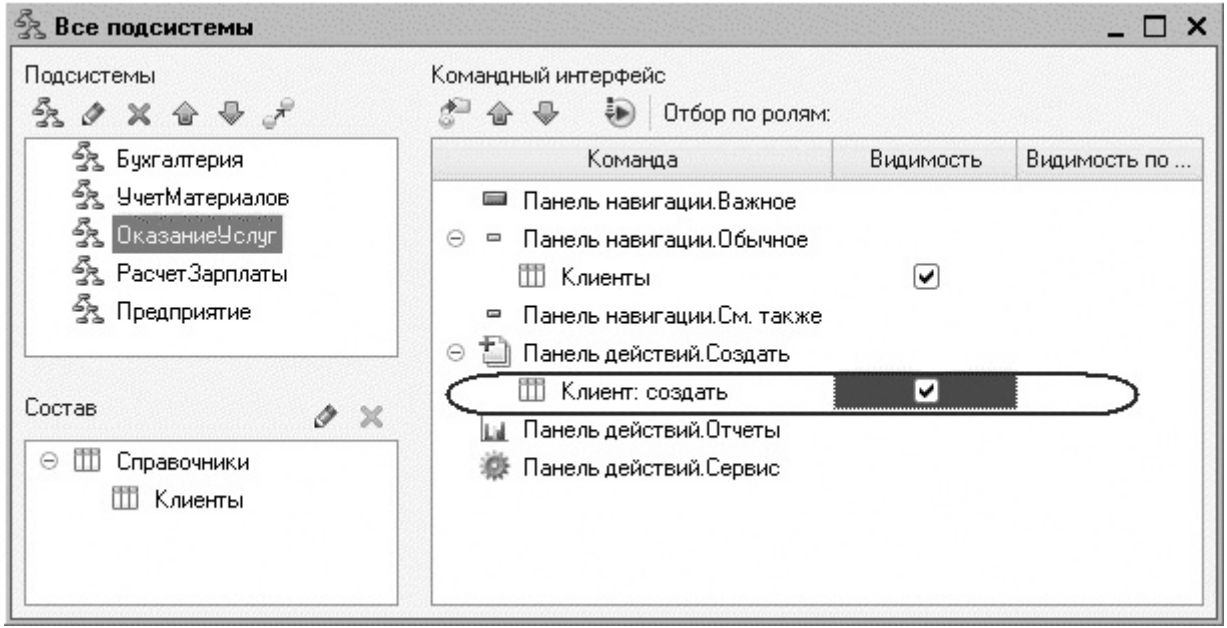


Рис. 3.22. Окно настройки подсистем

Для подсистемы *Бухгалтерия* никаких команд добавлять в панель действий не будем, так как это определяется прикладной логикой работы.

В данном случае мы предполагаем, что основную ежедневную работу с клиентами ведет менеджер, занимающийся оказанием услуг. В том числе он создает в базе новых клиентов, если они появляются. А бухгалтерия просто обрабатывает имеющиеся в базе данные для получения регламентированной отчетности.

Именно поэтому команду создания нового клиента мы отражаем в подсистеме *Оказание услуг*, где работает менеджер, а для бухгалтерии она невидима, так как не предполагается, что бухгалтеры будут вводить новых клиентов.

Однако это не лишает бухгалтера такой возможности – он может создать нового клиента, используя список клиентов (открыть список клиентов и добавить нового клиента). Наличие команды создания нового элемента без использования списка элементов – это вопрос удобства работы, а не ограничения прав пользователя, и мы предоставляем эту удобную возможность менеджеру, а не бухгалтеру.

Закроем окно редактирования справочника *Клиенты* и запустим «1С:Предприятие» в режиме отладки. Ответим утвердительно на запрос конфигуратора об обновлении конфигурации и увидим окно, содержащее список

изменений в структуре конфигурации, автоматически сгенерированный платформой. В данном случае мы добавили справочник *Клиенты*.

Нажмем кнопку *Принять* (рис. 3.23).

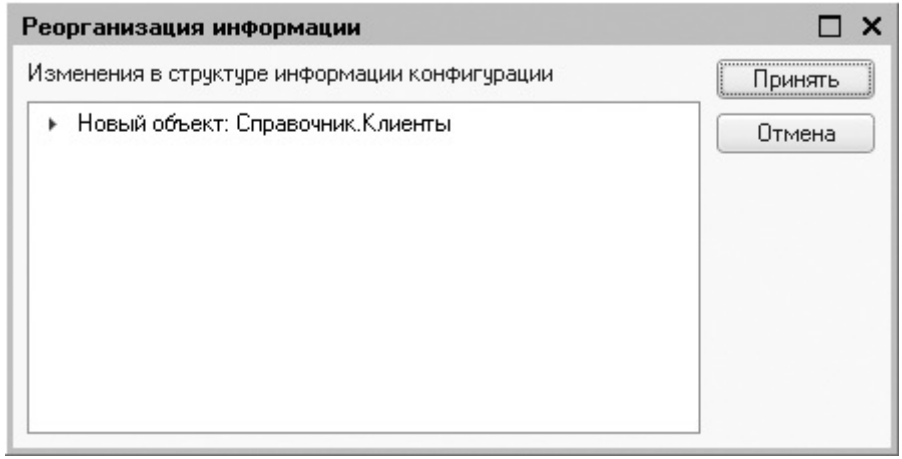


Рис. 3.23. Список изменений в структуре конфигурации

В режиме «1С:Предприятие»

Панель навигации и панель разделов

Перед нами откроется окно системы в режиме *1С:Предприятие*. Мы видим, что если перейти в раздел *Оказание услуг* или *Бухгалтерия*, то слева в вертикальной области окна появится *панель навигации* (рис. 3.24).

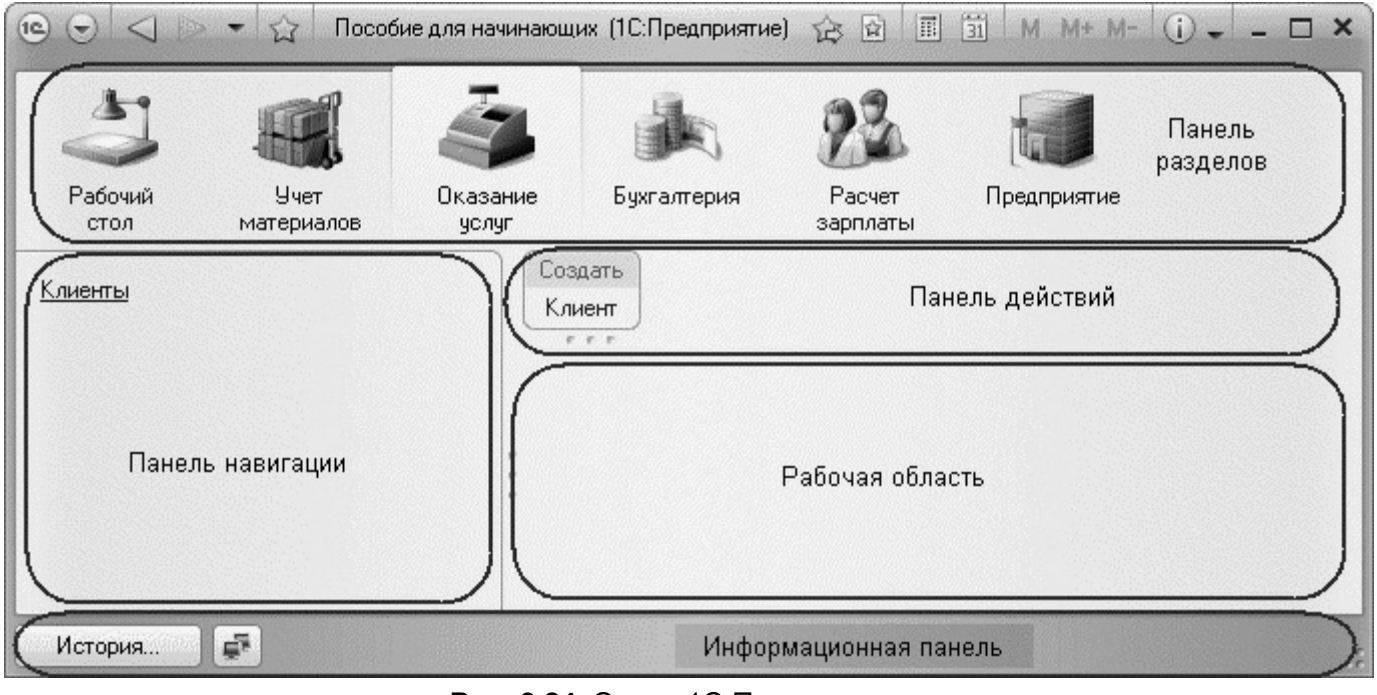


Рис. 3.24. Окно «1С:Предприятия»

Панель навигации отображает структуру выбранного раздела. Как правило, панель навигации предназначена для быстрого перехода к различным спискам в пределах выбранного раздела программы.

Сейчас она содержит команду для открытия нашего первого списка – *Клиенты*. Обратите внимание, что название команды *Клиенты* определяется свойством *Представление списка*, которое мы задали для этого справочника. Если это

свойство не задано, то для представления списка используется значение синонима объекта конфигурации *Справочник*.

Также в разделе *Оказание услуг* появилась панель действий (см. рис. 3.24). *Панель действий* содержит команды, которые соответствуют текущему разделу, выбранному в панели разделов. Эти команды объединены в стандартные группы: *Создать, Отчеты, Сервис* и группы, созданные разработчиком. Группа *Создать* включает в себя команды создания новых объектов информационной базы, например, документов или элементов справочников.

Сейчас в панели действий раздела *Оказание услуг* в группе *Создать* доступна команда для создания элементов нашего первого справочника *Клиенты*, которую мы сделали видимой в интерфейсе этого раздела. Этой командой мы и воспользуемся для создания новых элементов справочника, не открывая при этом списка клиентов.

Обратите внимание, что название стандартной команды создания нового элемента определяется свойством *Представление объекта*, которое мы задали для этого справочника. Если бы мы это свойство не задали, то в названии команды использовался бы синоним объекта конфигурации Справочник *Клиенты* – *Клиенты*. Это неудобно, так как ничем не отличается от команды открытия списка, и не совсем верно – ведь при создании элемента

справочника мы создаем только одного нового клиента.

Заметьте, что у раздела *Бухгалтерия* нет панели действий, так как для этой подсистемы мы не устанавливали видимость команды создания новых элементов из группы команд *Создать*. Кроме этого, нет ни одной другой видимой команды из групп *Отчеты* или *Сервис*. Соответственно, раз нет команд, входящих в группу, значит, невидима вся группа, а раз нет ни одной группы, значит, невидима вся панель действий.

Создание элементов справочника

Пока наш справочник пуст, поэтому добавим в него несколько элементов. Для этого выполним команду *Клиент* в панели действий раздела *Оказание услуг*.

Перед нами откроется форма для создания элемента справочника – основная форма объекта (рис. 3.25).

Внесем наименование нового клиента *Иванов Михаил Юрьевич*. Код вносить не будем, так как он генерируется автоматически.

Нажмем *Записать и закрыть*.

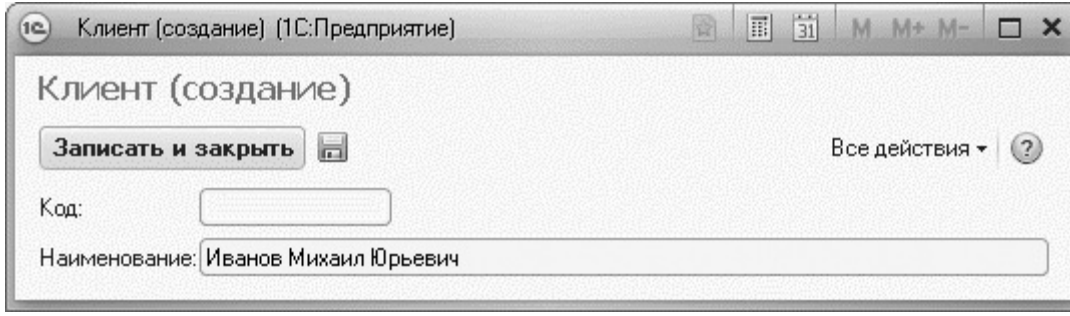


Рис. 3.25. Создание нового элемента справочника

При этом в правом нижнем углу появится информационное сообщение о том, какой элемент был создан либо изменен.

Кроме этого, нажав на ссылку в *информационной панели* (в нижней части окна приложения), можно открыть этот элемент. В этой панели (см. рис. 3.24) автоматически отображается информация о последних действиях, выполненных в системе. Это позволяет не пользоваться списком для того, чтобы убедиться, что нужный элемент записан.

Добавим еще одного клиента с наименованием *Роман*.

Последнего клиента с наименованием *Спиридонова Галина* добавим, пользуясь формой списка клиентов.

Для этого выполним команду *Клиенты*, расположенную в панели навигации

раздела *Оказание услуг*. Справа от панели навигации в рабочей области окна приложения откроется основная форма списка (рис. 3.26).

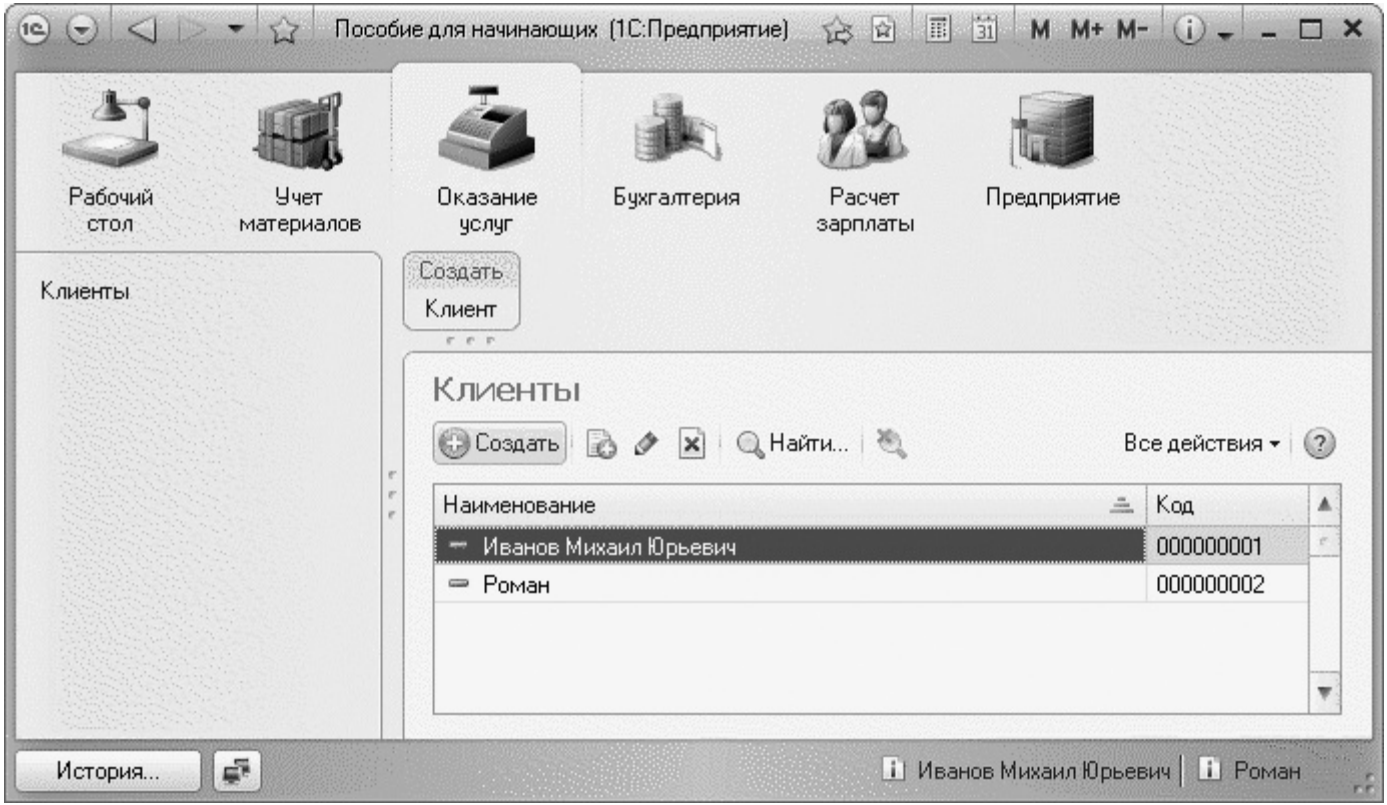


Рис. 3.26. Основная форма списка клиентов

Добавить новый элемент в справочник можно при помощи кнопки *Создать* в

командной панели формы или клавишей *Insert*.

Нажмем кнопку *Создать*.

Обратите внимание, что поле *Наименование* при вводе нового клиента подсвечено красным пунктиром. Это значит, что для этого поля по умолчанию выполняется проверка заполнения. Если это поле оставить пустым и попытаться записать клиента, то будет получено сообщение об ошибке (рис. 3.27).

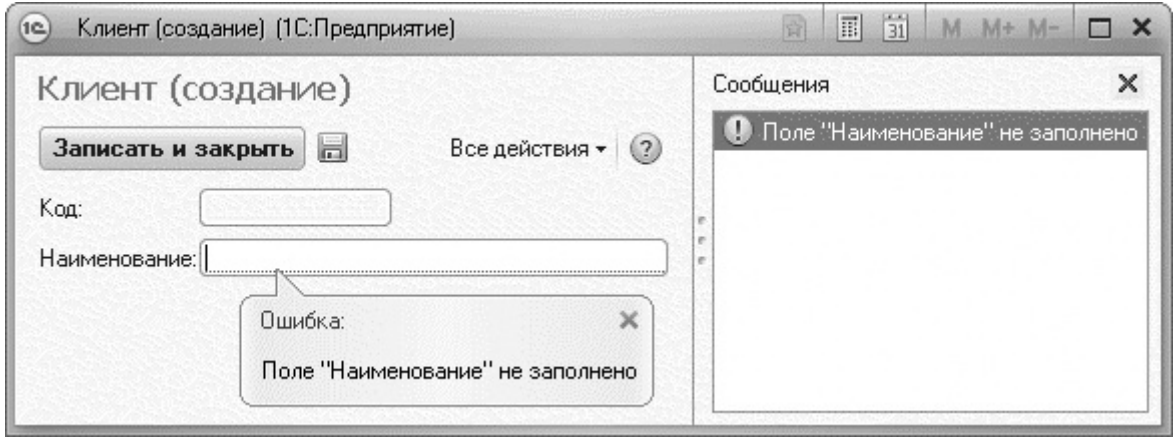


Рис. 3.27. Сообщение об ошибке при вводе нового элемента справочника

Так происходит потому, что система автоматически устанавливает проверку заполнения у некоторых стандартных реквизитов объектов, например у

наименования справочника (если основное представление справочника в виде наименования).

Внесем наименование клиента – *Спиридонова Галина*.

После добавления элементов справочник будет выглядеть следующим образом (рис. 3.28).

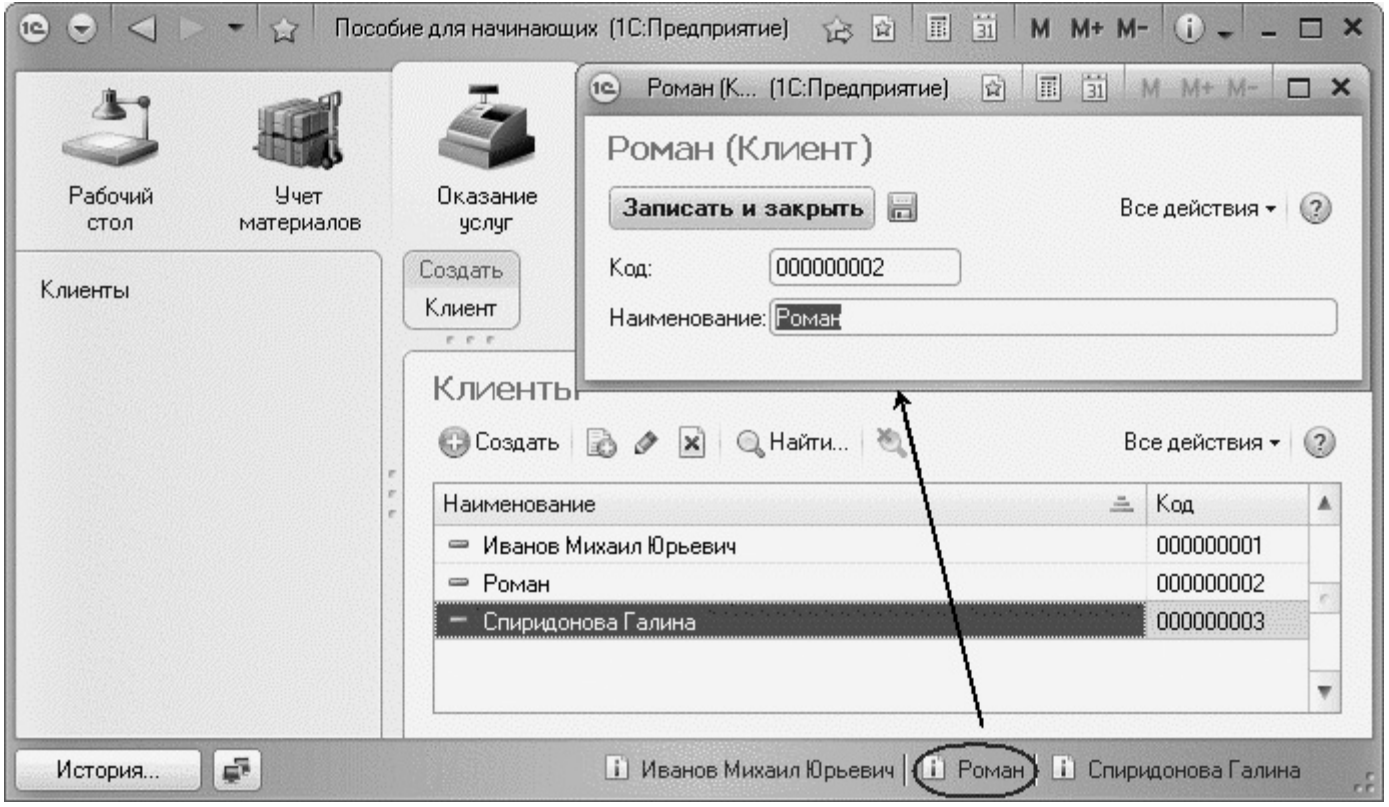


Рис. 3.28. Список клиентов

Чтобы открыть существующий элемент справочника для редактирования, нужно дважды щелкнуть на нем мышью. А кроме этого, нажав на ссылку в информационной панели, мы тоже можем открыть для редактирования один из последних измененных элементов справочника (см. рис. 3.28).

«Теория».Проверка заполнения стандартных реквизитов

Закроем «1С:Предприятие» и вернемся в конфигуратор.

Чтобы посмотреть состав и свойства стандартных реквизитов справочника, в окне редактирования объекта конфигурации *Справочник Клиенты* на закладке *Данные* нажмем кнопку *Стандартные реквизиты*.

Выделим в списке реквизит *Наименование*, вызовем его контекстное меню и выберем пункт *Свойства* (рис. 3.29).

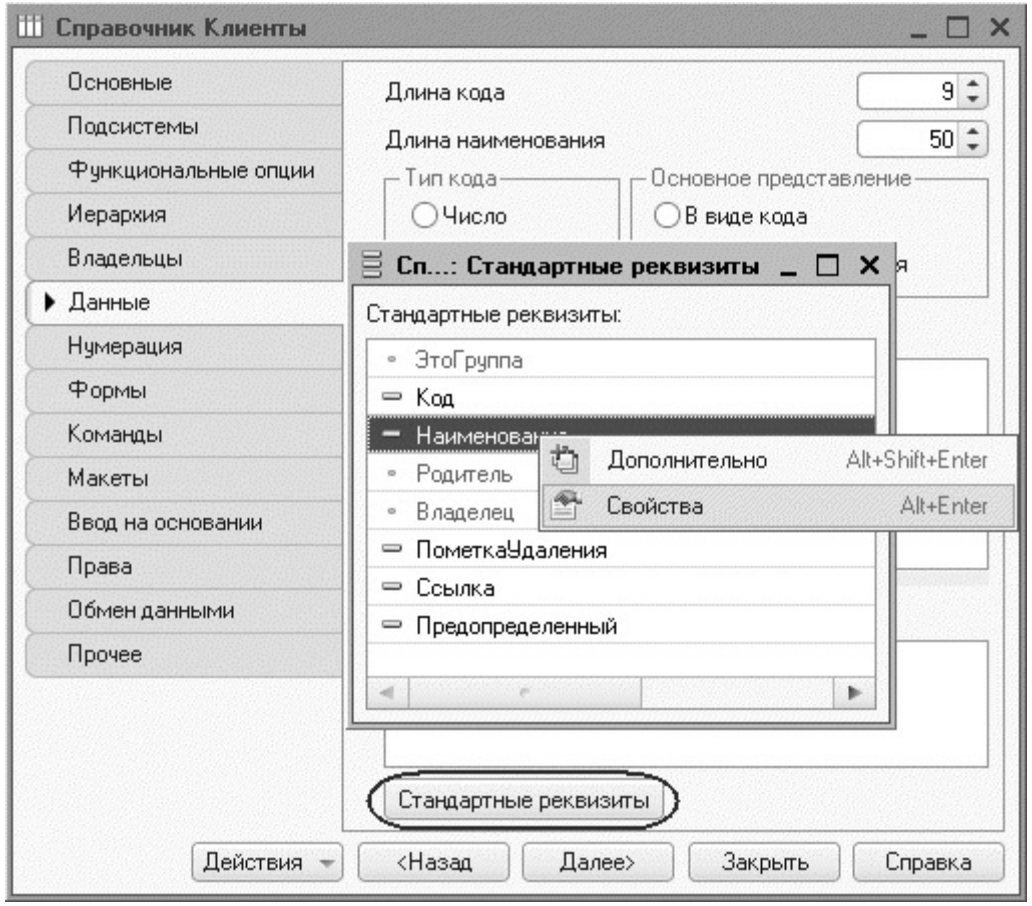


Рис. 3.29. Состав стандартных реквизитов справочника

В палитре свойств стандартного реквизита *Наименование* мы видим, что свойство *Проверка заполнения* по умолчанию установлено в значение

Выдавать ошибку (рис. 3.30).

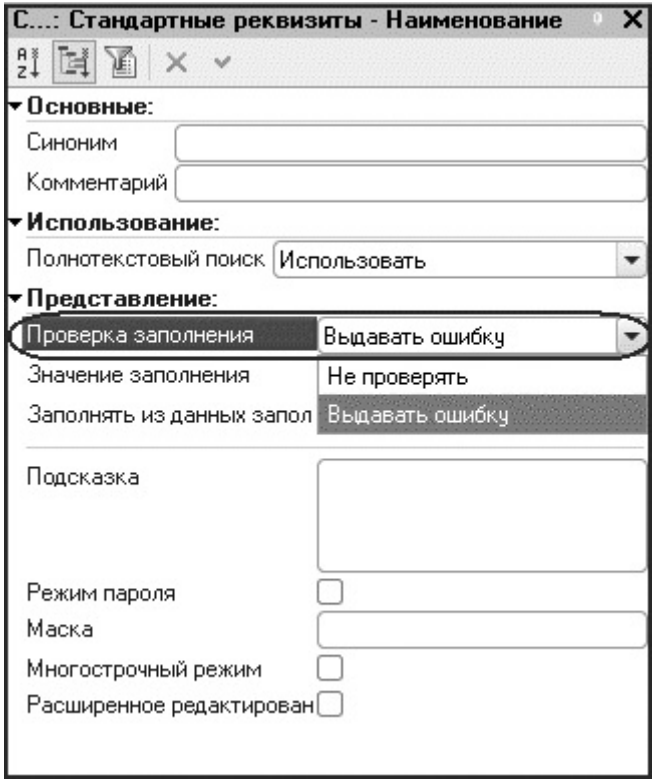


Рис. 3.30. Палитра свойств стандартного реквизита «Наименование»

Это означает, что если реквизит *Наименование* не заполнен, то будет выведено сообщение об ошибке (см. рис. 3.27).

Справочник с табличной частью

Теперь мы можем перейти к созданию второго справочника, который будет использоваться в нашей конфигурации, – справочника *Сотрудники*.

Этот справочник будет устроен несколько сложнее, чем справочник *Клиенты*. Дело в том, что в нем мы будем хранить не только фамилию, имя и отчество сотрудника, но и информацию о его прошлой трудовой деятельности.

Эта информация однородна по своей структуре (организация, начало, окончание работы, занимаемая должность), но количество предыдущих мест работы у разных сотрудников может быть различным. Поэтому для хранения такой информации мы будем использовать табличную часть справочника.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Справочник*. Назовем его *Сотрудники*.

На основании имени платформа автоматически заполнит его синоним.

Зададим *Представление объекта* как *Сотрудник*.

Представление списка устанавливать не будем, а *Расширенное представление списка* зададим как *Список сотрудников* (рис. 3.31).

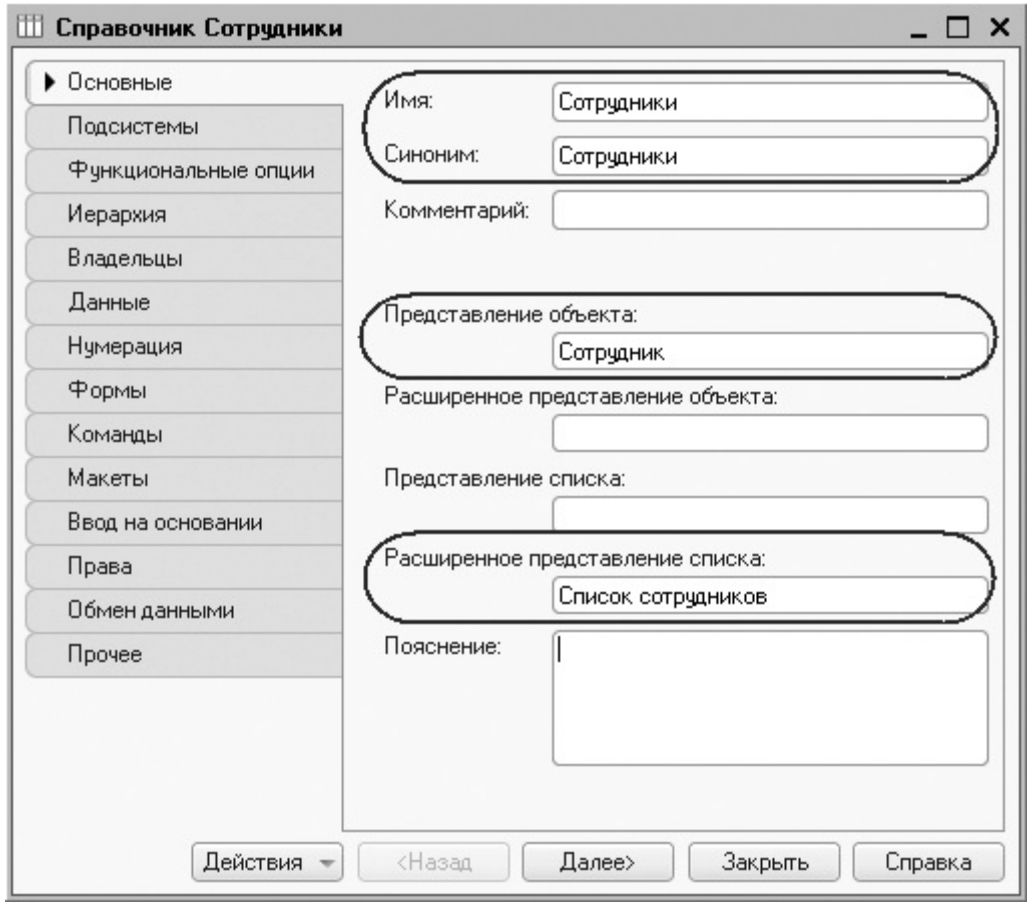


Рис. 3.31. Установка основных свойств справочника

Нажмем кнопку *Далее* и перейдем на закладку *Подсистемы*.

По логике нашей конфигурации список сотрудников должен быть доступен в разделах *Оказание услуг* и *Расчет зарплаты*. Действительно, при оказании услуг должен быть указан сотрудник, оказавший эти услуги, и по результатам этой работы мы будем начислять зарплату каждому сотруднику.

Поэтому отметим в списке подсистем *Оказание услуг* и *Расчет зарплаты* (рис. 3.32).

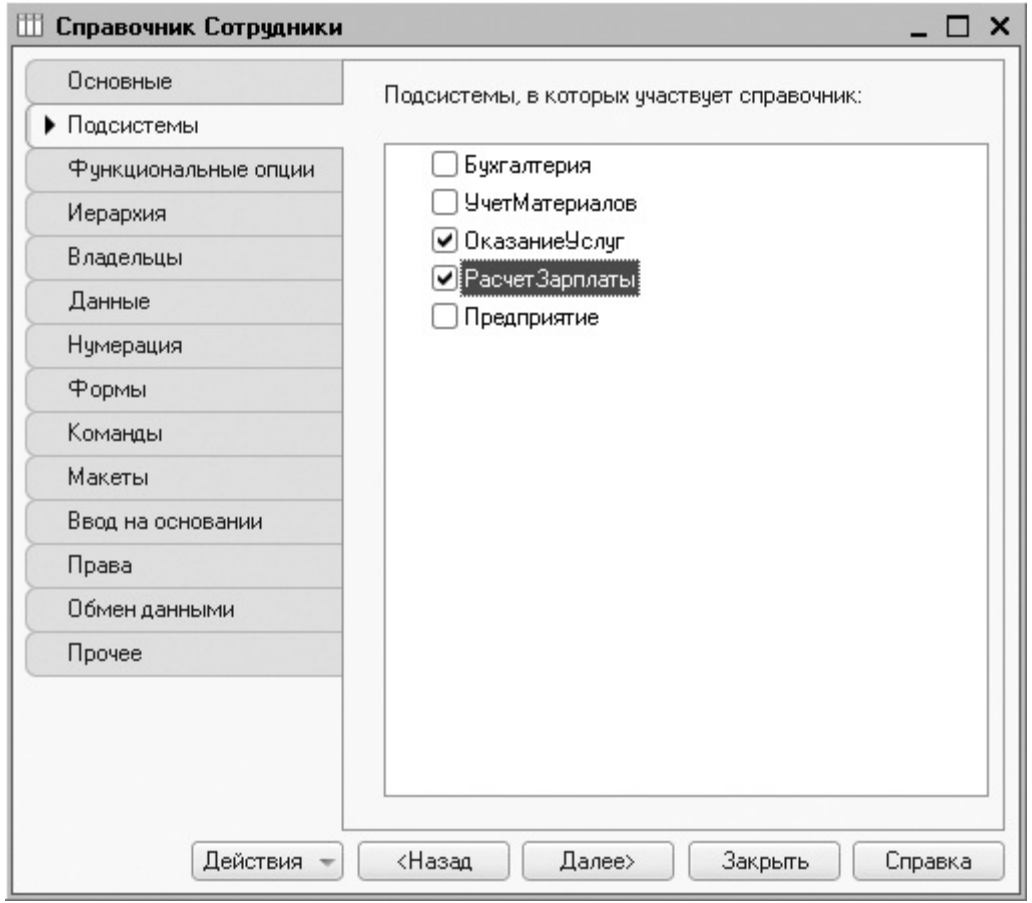


Рис. 3.32. Определение списка подсистем, в которых отображается справочник

Перейдем на закладку *Данные*. Оставим по умолчанию длину и тип кода, длину наименования справочника зададим равной *50* символам.

Табличная часть

Наша задача – создать справочник, имеющий табличную часть. Поэтому добавим в справочник новую табличную часть с именем *ТрудоваяДеятельность*.

Для этого нажмем кнопку *Добавить табличную часть* над списком табличных частей справочника (рис. 3.33).

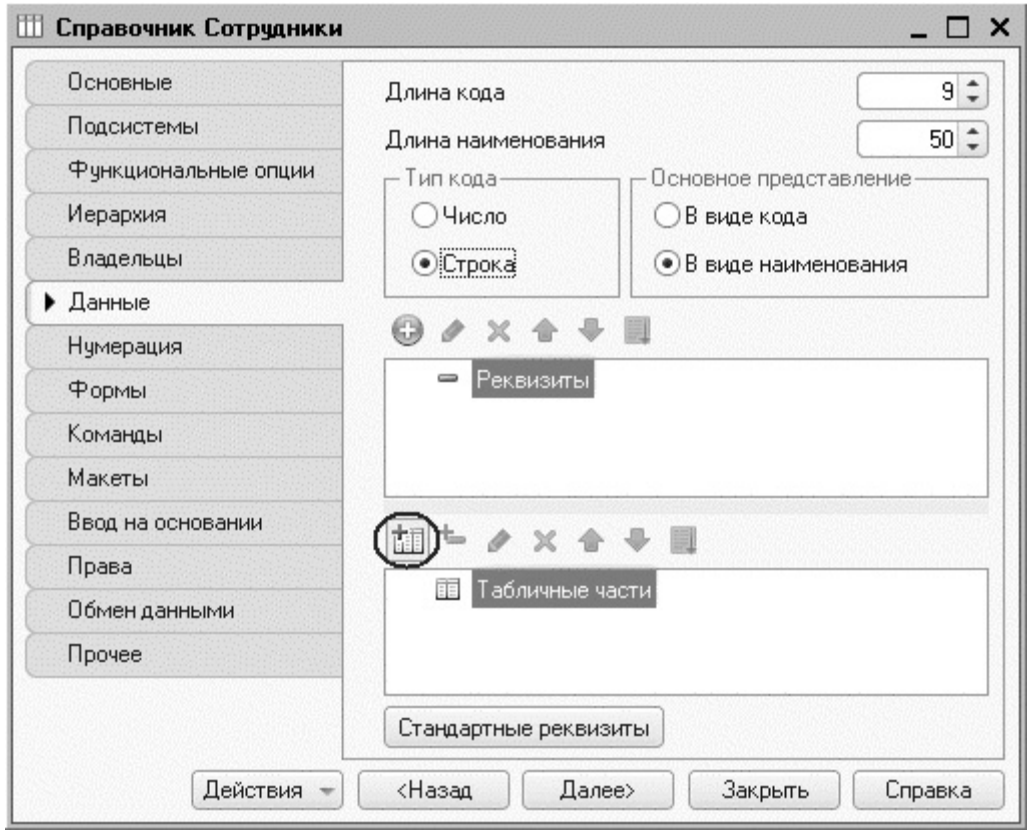


Рис. 3.33. Добавление новой табличной части справочника

Зададим имя табличной части – *ТрудоваяДеятельность* (рис. 3.34).

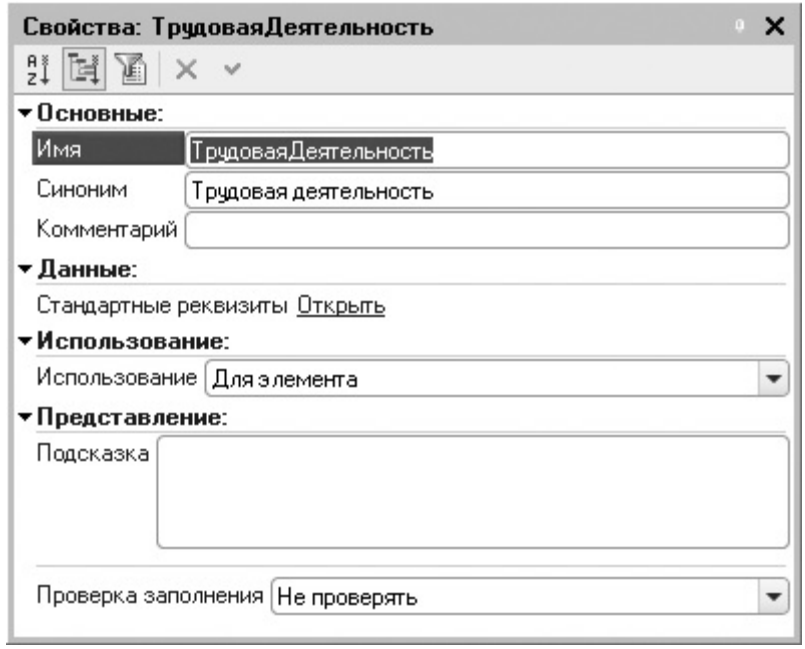


Рис. 3.34. Добавление новой табличной части справочника

Создадим реквизиты табличной части *ТрудоваяДеятельность.*

Для этого нажмем кнопку *Добавить реквизит* над списком табличных частей справочника (рис. 3.35):

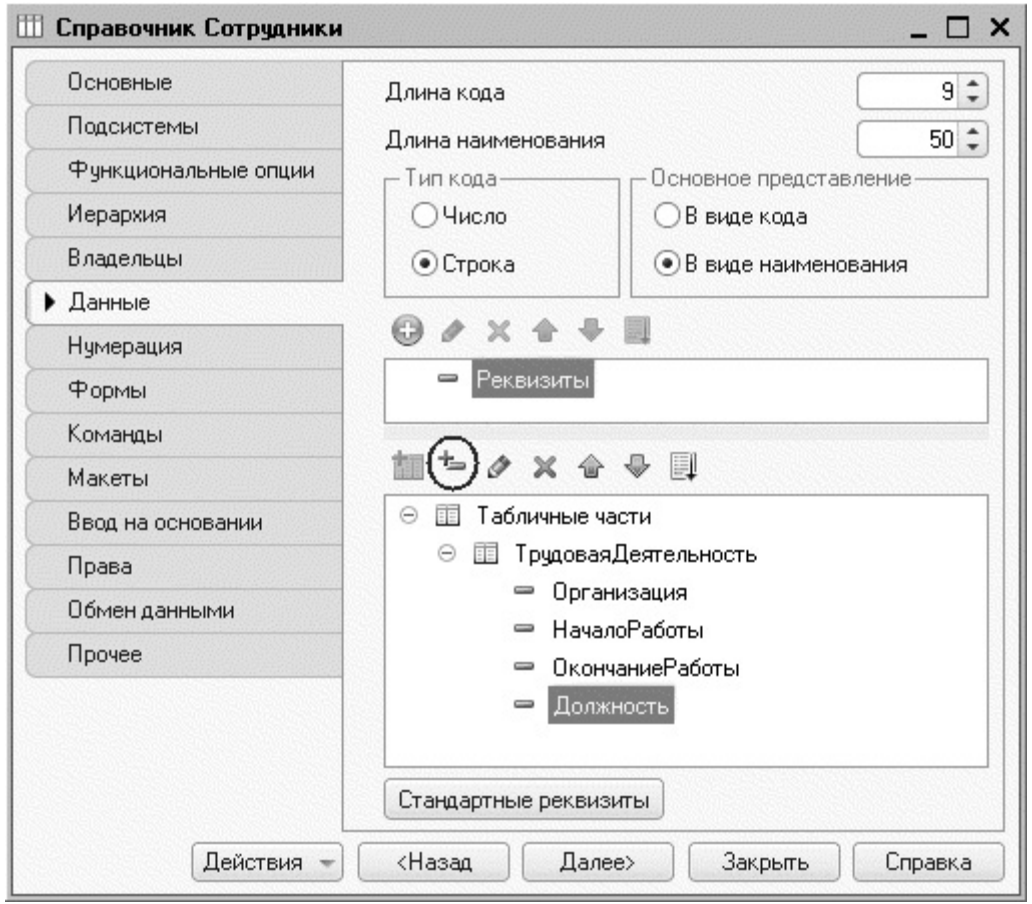


Рис. 3.35. Добавление нового реквизита в табличную часть справочника

Добавим следующие реквизиты:

- *Организация* – тип *Строка*, длина *100*;
- *НачалоРаботы* – тип *Дата*, состав даты – *Дата*;
- *ОкончаниеРаботы* – тип *Дата,* состав даты – *Дата*;
- *Должность* – тип *Строка*, длина *100*.

Для реквизитов *НачалоРаботы* и *ОкончаниеРаботы* мы выбрали состав даты – *Дата* (рис. 3.36), поскольку в системе «1С:Предприятие 8» значения типа *Дата* содержат как дату, так и время. В данном случае время начала и окончания работы нам безразлично.

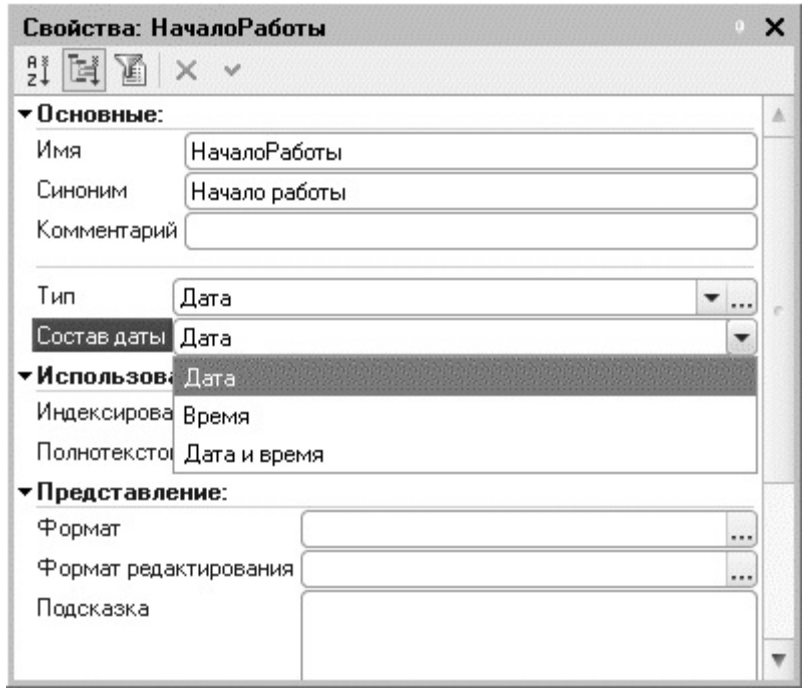


Рис. 3.36. Свойства реквизита табличной части справочника

В заключение отредактируем командный интерфейс, чтобы нам было удобнее вводить новые элементы справочника. Сделаем видимой в панели действий подсистемы *РасчетЗарплаты* стандартную команду для создания новых сотрудников.

Для этого в дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*.

В открывшемся окне слева в списке *Подсистемы* выделим подсистему *РасчетЗарплаты*.

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель действий.Создать* включим видимость у команды *Сотрудник: создать*.

Также мы видим, что в группу *Панель навигации.Обычное* добавилась команда *Сотрудники* для открытия этого списка. Она включена по умолчанию (рис. 3.37).

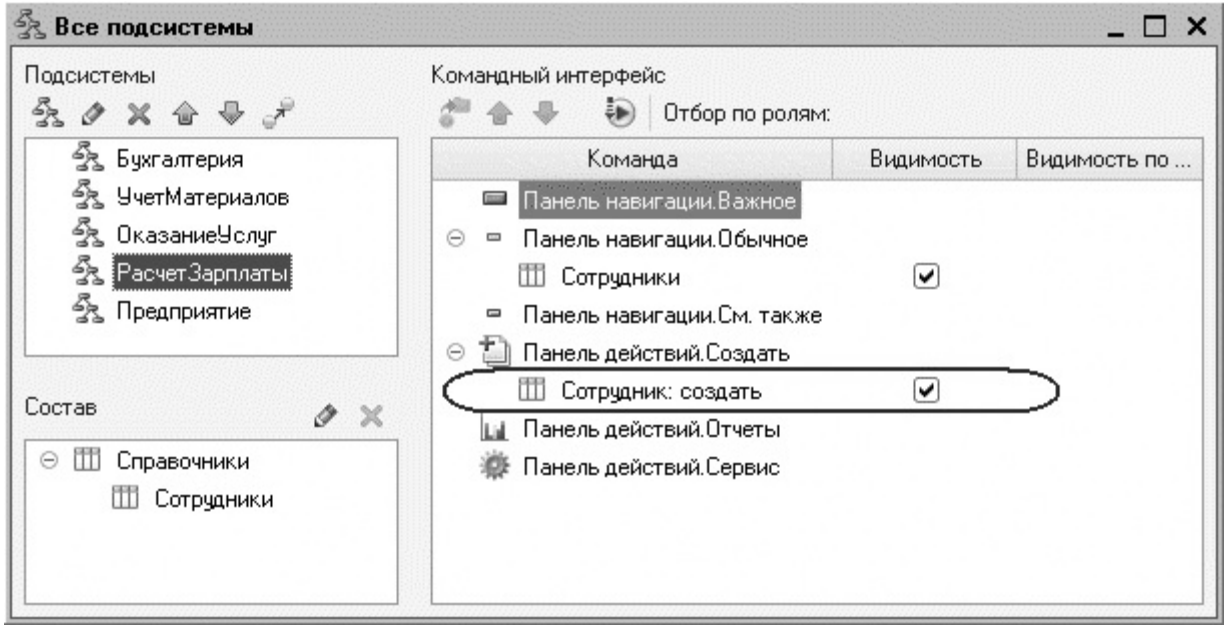


Рис. 3.37. Окно настройки подсистем

Для подсистемы *ОказаниеУслуг* никаких команд добавлять в панель действий не будем, так как вряд ли понадобится пополнять список сотрудников в этом разделе.

На этом создание справочника *Сотрудники* завершено.

Закроем окно редактирования справочника *Сотрудники* и запустим «1С:Предприятие» в режиме отладки.

Ответим утвердительно на запрос конфигуратора об обновлении конфигурации и увидим окно, содержащее список изменений в структуре конфигурации, автоматически сгенерированный платформой.

В данном случае мы добавили справочник *Сотрудники*. Нажмем кнопку *Принять* (рис. 3.38).

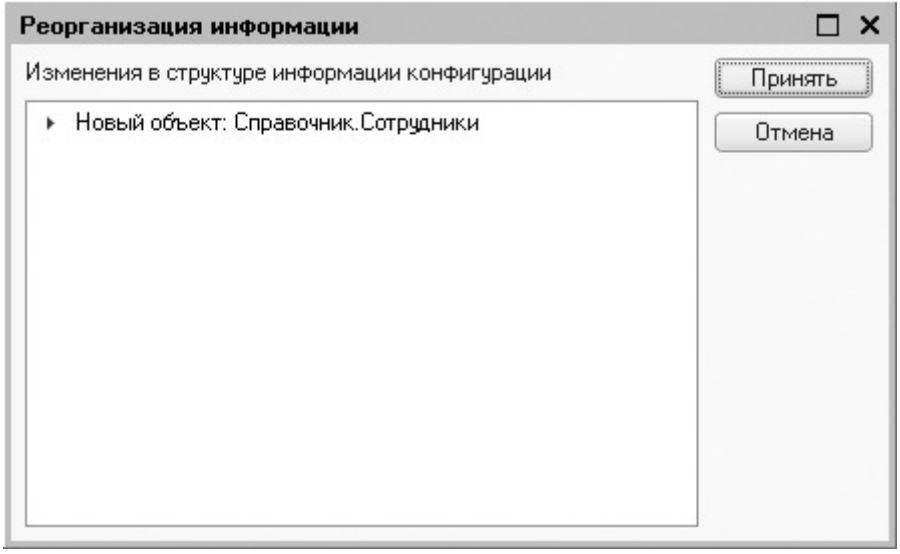


Рис. 3.38. Список изменений в структуре конфигурации

В режиме «1С:Предприятие»

В открывшемся окне «1С:Предприятия» мы видим, что в панели навигации

разделов *Оказание услуг* и *Расчет зарплаты* появилась команда *Сотрудники* для открытия списка сотрудников (рис. 3.39).

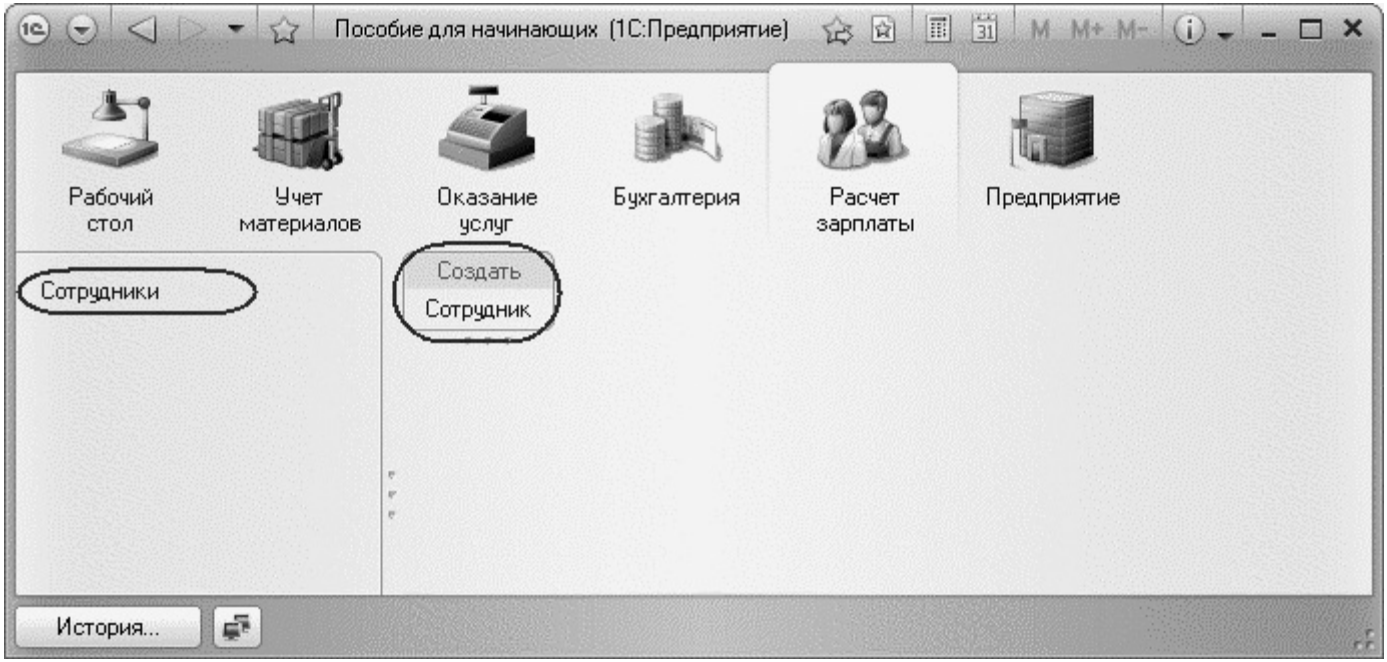


Рис. 3.39. Раздел «Расчет зарплаты»

Название этой команды определяется синонимом объекта, так как *Представление списка* мы для этого справочника не задавали.

Также в панели действий раздела *Расчет зарплаты* появилась команда

Сотрудник для создания новых сотрудников (см. рис. 3.39). Название этой команды определяется свойством *Представление объекта*, которое мы задали для этого справочника.

Этой командой мы и воспользуемся для создания новых элементов справочника, не открывая при этом списка сотрудников.

Заполнение табличной части

Выполним команду *Сотрудник*.

Перед нами откроется форма для создания элемента справочника – основная форма объекта. Заголовок этой формы определяется свойством *Представление объекта*.

Эта форма содержит табличную часть с реквизитами, которые мы описали в конфигураторе для этого справочника.

Создадим следующих сотрудников (рис. 3.40, 3.41, 3.42):

Гусаков Николай Дмитриевич:

Трудовая деятельность:

- Организация – *ЗАО «НТЦ»,*
- Начало работы – *01.02.2000,*
- Окончание работы – *16.04.2003,*
- Должность – *Ведущий специалист.*

Деловой Иван Сергеевич:

Трудовая деятельность:

- *1:*
  - Организация *– ООО «Автоматизация»,*
  - Начало работы *– 22.01.1996,*
  - Окончание работы *– 31.12.2002,*
  - Должность *– Инженер.*
- *2:*
- Организация – *ЗАО «НПО СпецСвязь»,*
- Начало работы – *20.06.1986,*
- Окончание работы – *21.01.1995,*
- Должность – *Начальник производства.*

Симонов Валерий Михайлович:

Трудовая деятельность:

- Организация – *ООО «СтройМастер»,*
- Начало работы – *06.02.2001,*
- Окончание работы – *03.04.2004,*
- Должность – *Прораб.*

Строки табличной части справочника можно добавлять кнопкой *Добавить* (рис. 3.40) и располагать в произвольном порядке, используя кнопки *Вверх*, *Вниз* в командной панели табличной части (рис. 3.41).

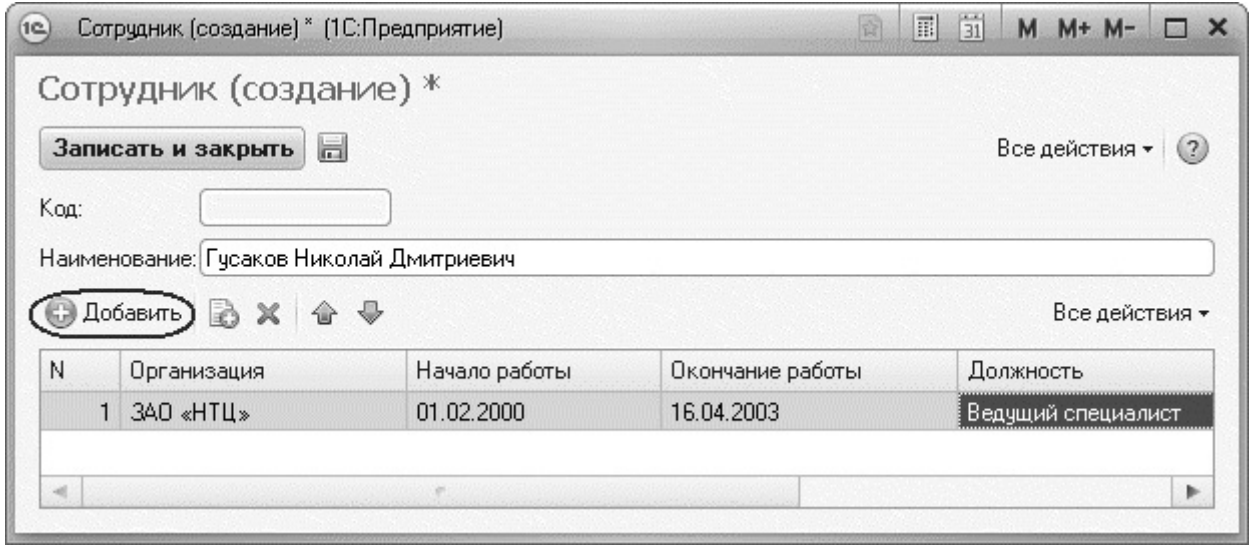


Рис. 3.40. Заполнение элемента справочника «Сотрудники»

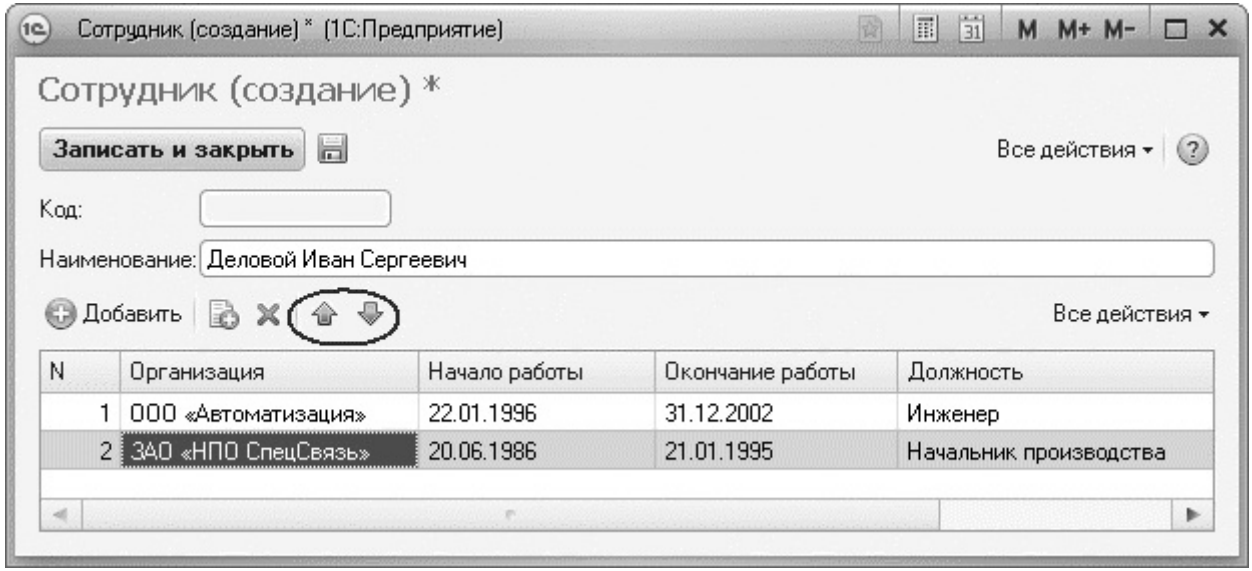


Рис. 3.41. Заполнение элемента справочника «Сотрудники»

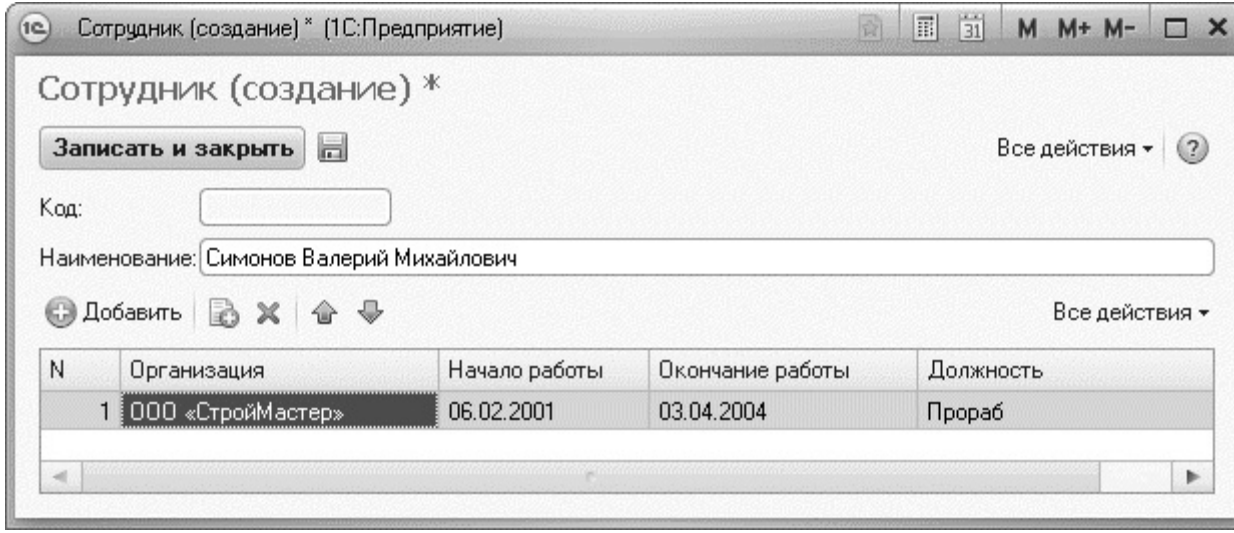


Рис. 3.42. Заполнение элемента справочника «Сотрудники»

Чтобы просмотреть список добавленных сотрудников, выполним команду *Сотрудники*, расположенную в панели навигации раздела *Расчет зарплаты*.

Справа от панели навигации в рабочей области окна приложения откроется основная форма списка.

Обратите внимание, что заголовок этой формы определяется свойством *Расширенное представление списка*, которое мы задали для этого справочника (рис. 3.43).

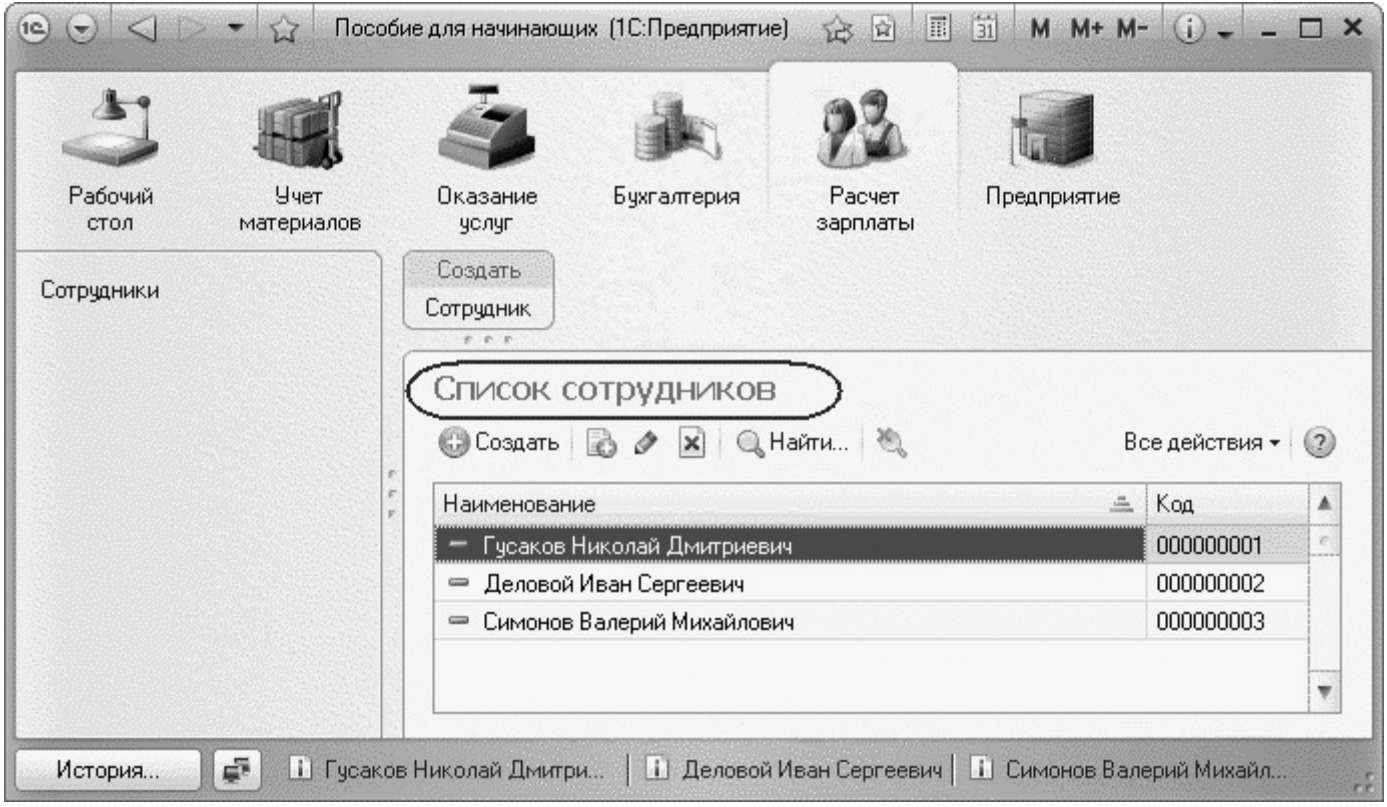


Рис. 3.43. Список сотрудников

Теперь мы можем приступить к созданию следующего справочника – *Номенклатура*.

Иерархический справочник

Справочник *Номенклатура* будет содержать информацию об услугах, которые оказывает ООО «На все руки мастер», и о тех материалах, которые при этом могут быть использованы.

Этот справочник не будет сложным. Единственная особенность, которой он будет обладать, – наличие иерархической структуры. Для того чтобы справочником было удобно пользоваться, мы сгруппируем услуги в одну группу, а материалы – в другую.

Кроме этого, поскольку ООО «На все руки мастер» оказывает самые разные услуги, они также будут логически собраны в несколько групп. То же самое можно сказать и про материалы.

В режиме «Конфигуратор»

Создадим новый объект конфигурации *Справочник* и назовем его *Номенклатура*. На основании имени платформа автоматически заполнит его синоним.

Поскольку понятие *Номенклатура* не имеет единственного числа, больше никаких свойств, определяющих представление объекта в интерфейсе приложения, задавать не будем. Вместо *Представления объекта* и *Представления списка* будет использоваться *Синоним* объекта –

Номенклатура.

Перейдем на закладку *Подсистемы*.

По логике нашей конфигурации список номенклатуры должен быть доступен в разделах *Учет материалов, Оказание услуг* и *Бухгалтерия*. Действительно, к первым двум разделам этот справочник имеет прямое отношение, а для бухгалтерского анализа всегда может понадобиться список материалов и услуг. Поэтому отметим в списке подсистем эти подсистемы (рис. 3.44).

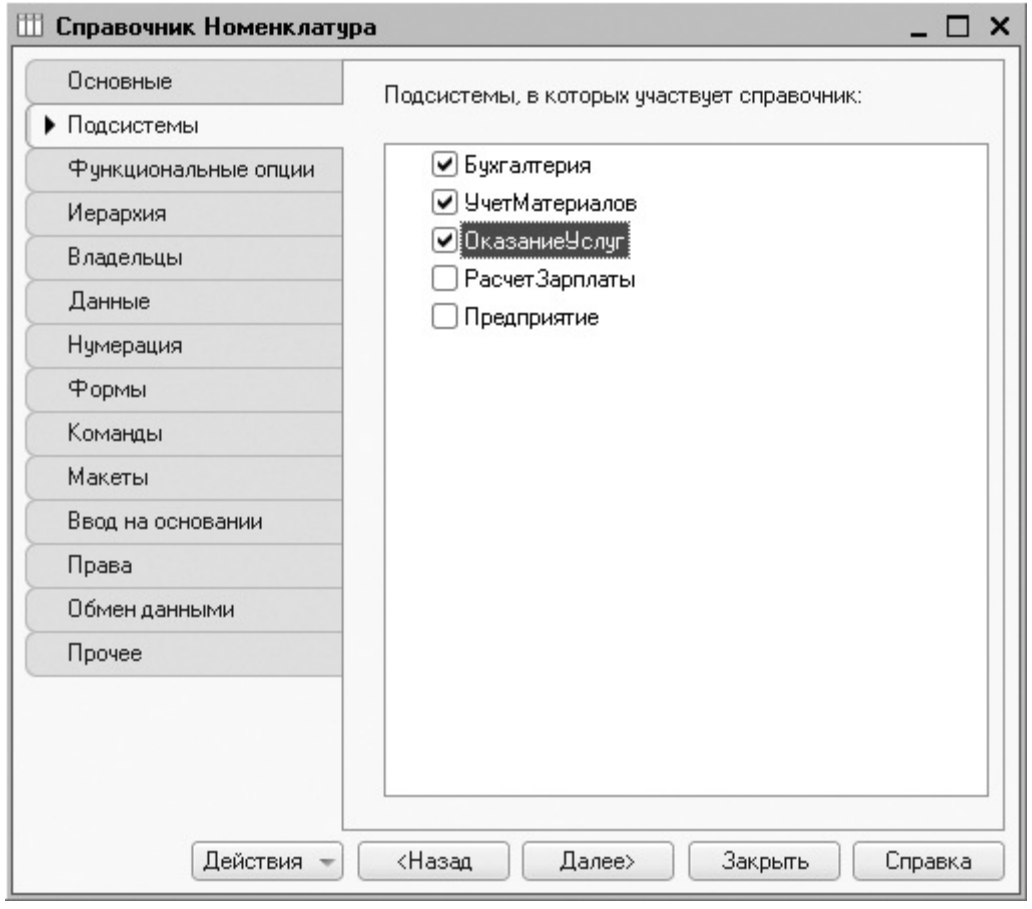


Рис. 3.44. Определение списка подсистем, в которых отображается справочник

Наша задача состоит в создании иерархического справочника. Перейдем на закладку *Иерархия* и установим флажок *Иерархический справочник* (рис. 3.45).

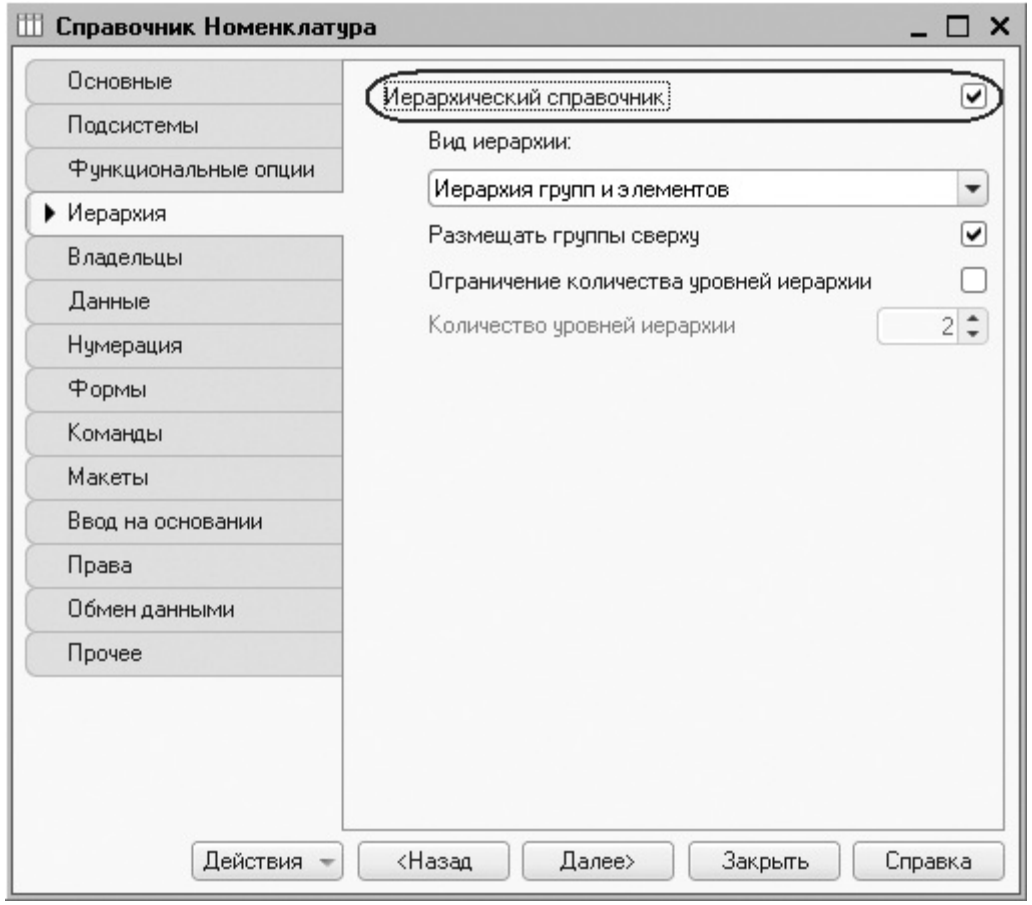


Рис. 3.45. Установка признака иерархического справочника

На закладке *Данные* оставим по умолчанию длину и тип кода, длину наименования справочника зададим равной *100* символам.

Прежде чем запускать «1С:Предприятие», настроим интерфейс приложения, чтобы нам было удобнее вводить новые элементы справочника. Сделаем доступной в панели действий разделов *УчетМатериалов* и *ОказаниеУслуг* стандартную команду для создания новых элементов списка номенклатуры.

Для этого в дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*.

В открывшемся окне слева в списке *Подсистемы* выделим подсистему *УчетМатериалов*.

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель действий.Создать* включим видимость у команды *Номенклатура: создать*.

Также мы видим, что в группу *Панель навигации.Обычное* добавилась команда *Номенклатура* для открытия этого списка. Она включена по умолчанию (рис. 3.46).

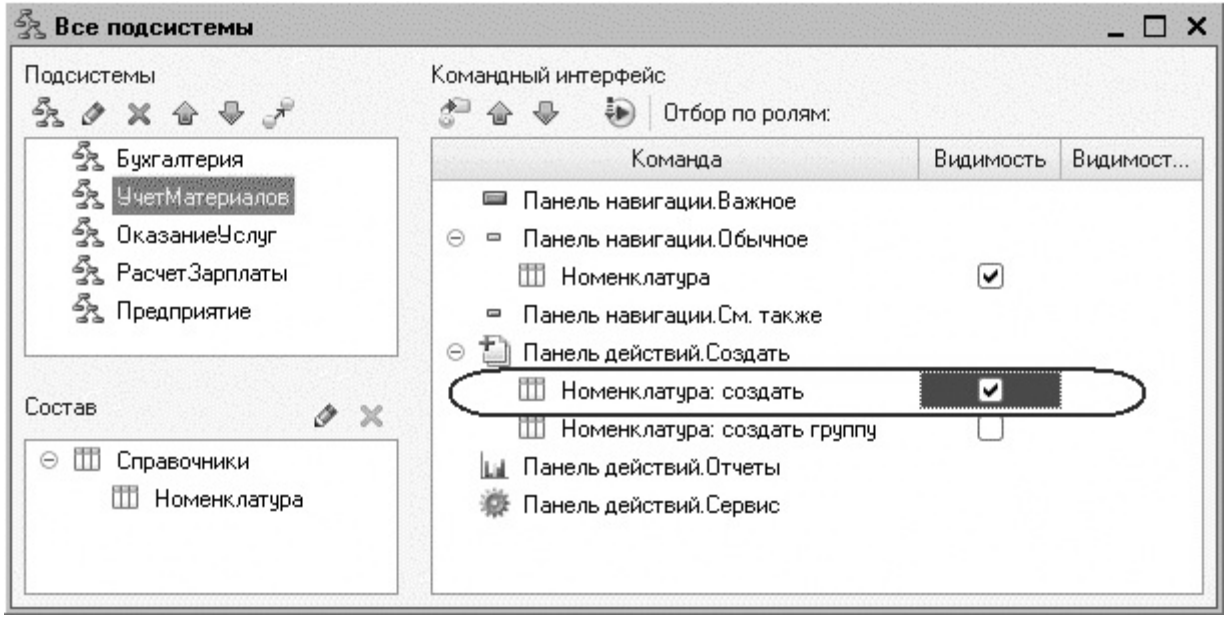


Рис. 3.46. Окно настройки подсистем

Выделив в списке подсистем *ОказаниеУслуг*, проделаем те же действия.

А для подсистемы *Бухгалтерия* никаких команд добавлять в панель действий не будем, так как вряд ли понадобится пополнять список номенклатуры в этом разделе.

ПРИМЕЧАНИЕ

Вызвать и настроить фрагмент командного интерфейса для конкретной

подсистемы можно также из окна редактирования этой подсистемы, нажав кнопку ***Командный интерфейс****.*

Теперь заполним справочник *Номенклатура*. В процессе заполнения мы покажем, как создавать группы и переносить элементы из одной группы в другую.

Закроем окно редактирования справочника *Номенклатура* и запустим «1С:Предприятие» в режиме отладки.

Ответим утвердительно на запрос конфигуратора об обновлении конфигурации и увидим окно, содержащее список изменений в структуре конфигурации, автоматически сгенерированный платформой. В данном случае мы добавили справочник *Номенклатура*.

Нажмем кнопку *Принять* (рис. 3.47).

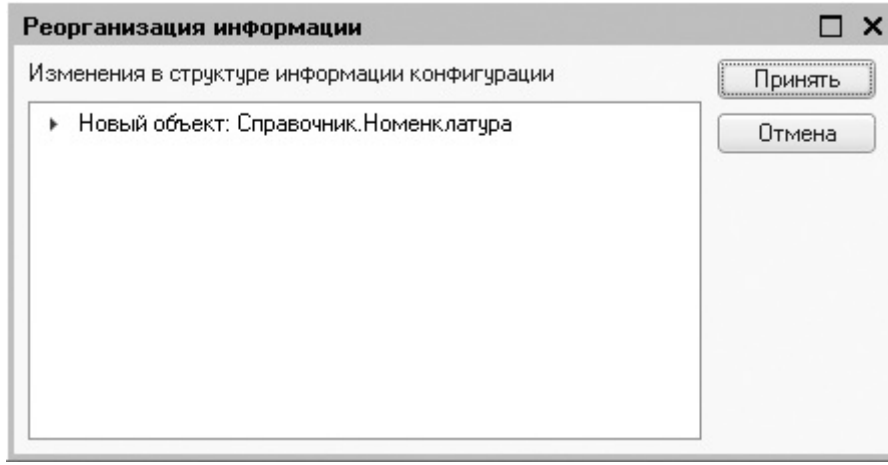


Рис. 3.47. Список изменений в структуре конфигурации

В режиме «1С:Предприятие»

В открывшемся окне «1С:Предприятия» мы видим, что в панели навигации разделов *Учет материалов, Оказание услуг* и *Бухгалтерия* появилась команда *Номенклатура* (см. рис. 3.48).

Название этой команды определяется синонимом объекта, так как других представлений мы для этого справочника не задавали.

Выполним команду *Номенклатура* в панели навигации раздела *Учет материалов*.

Справа от панели навигации в рабочей области окна приложения откроется основная форма списка.

Создание элементов в иерархическом справочнике

Создадим две группы в корне справочника: *Материалы* и *Услуги*.

Для этого нажмем кнопку *Создать группу* в командной панели формы списка (рис. 3.48).

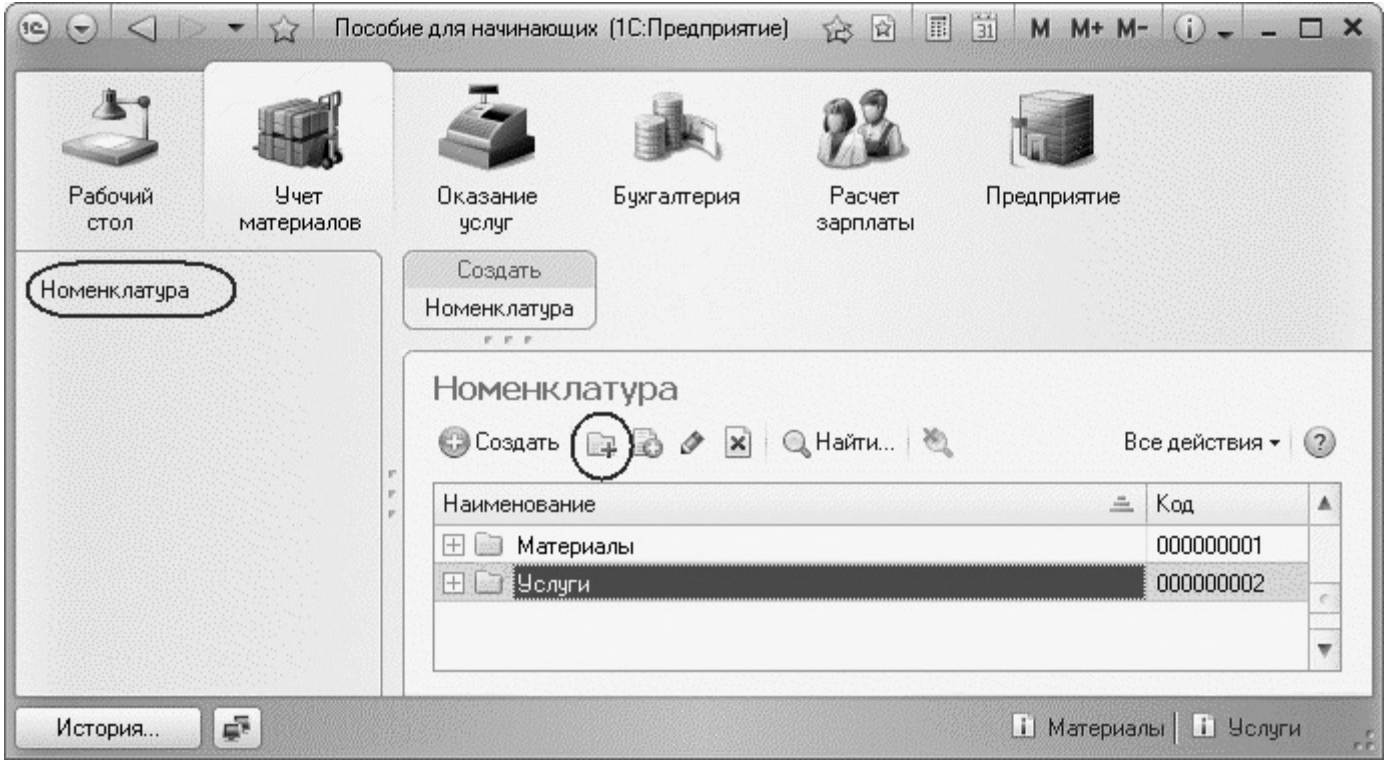


Рис. 3.48. Создание групп справочника «Номенклатура»

Зададим наименование групп: *Материалы*, *Услуги*. Поля *Родитель* и *Код* заполнять не будем.

Затем раскроем группу *Материалы* нажатием на + и создадим в ней пять элементов:

- *Строчный трансформатор Samsung*,
- *Строчный трансформатор GoldStar*,
- *Транзистор Philips 2N2369*,
- *Шланг резиновый*,
- *Кабель электрический*.

Для добавления элемента в открытую группу справочника нажмем кнопку *Создать* в командной панели формы списка справочника.

Перед нами откроется форма для создания элемента справочника – основная форма объекта. Причем если новый элемент добавляется из формы списка в некоторую открытую группу, то система автоматически подставляет в качестве родителя эту группу.

В данном случае родителем является группа *Материалы* (рис. 3.49).

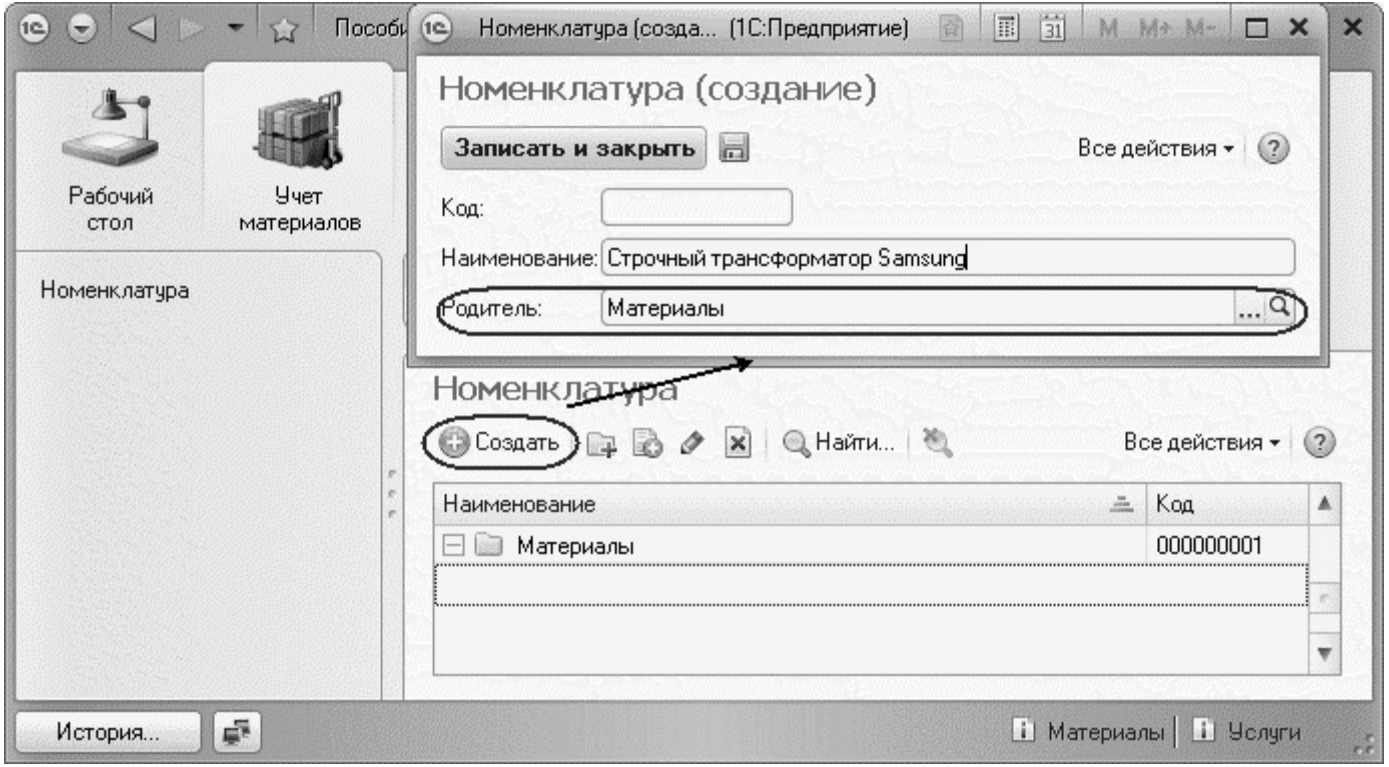


Рис. 3.49. Создание элементов в группе «Материалы»

Для создания нового элемента справочника можно также воспользоваться командой *Номенклатура*, которая появилась в панели действий разделов *Учет материалов* и *Оказание услуг* (см. рис. 3.48).

В этом случае, если элемент добавляется командой из панели действий, она

никак не связана со списком номенклатуры. Поэтому система не знает, в какую группу добавлять элемент, и родителя нужно указывать вручную (рис. 3.50).

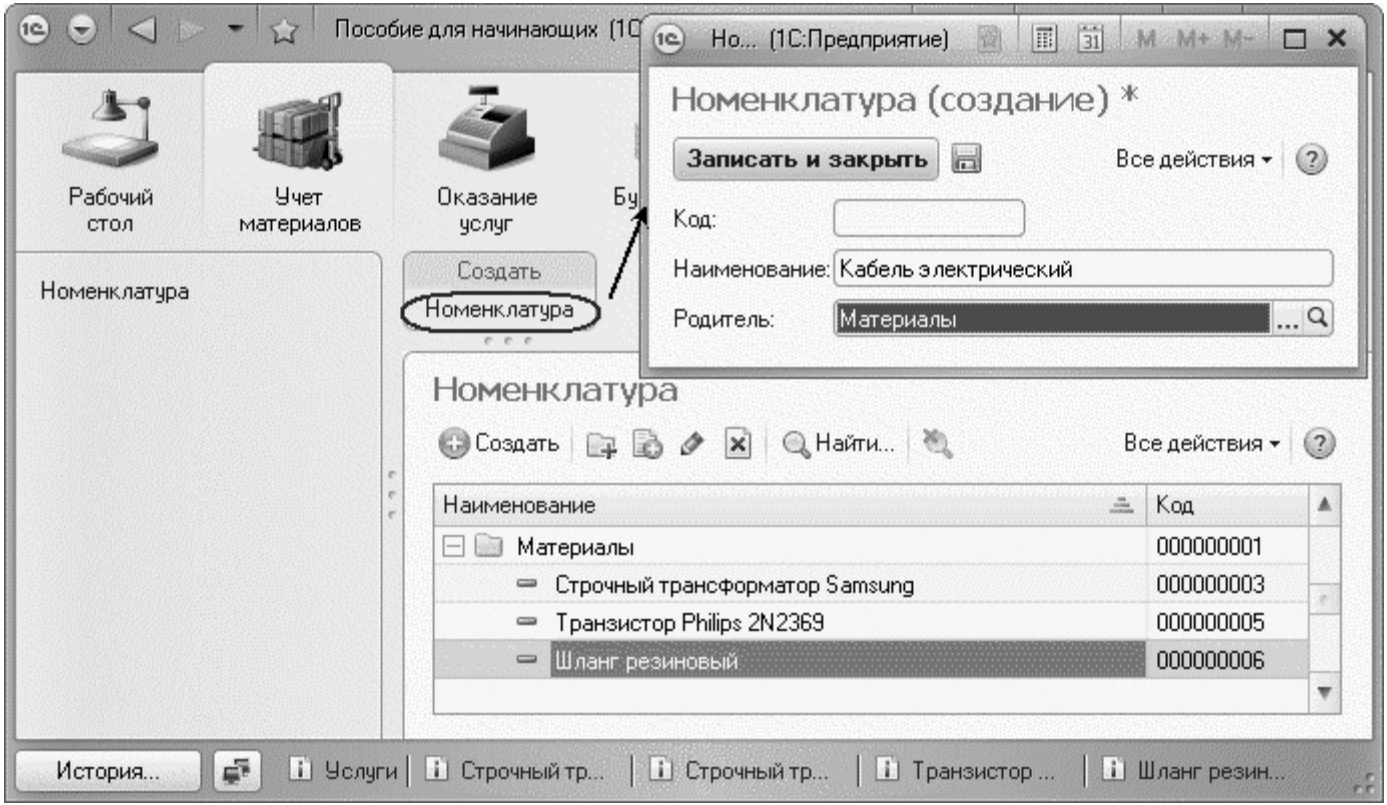


Рис. 3.50. Создание элементов в группе «Материалы»

При создании новых материалов или услуг список номенклатуры открывать

необязательно. Факт добавления элемента в справочник отражается в сообщении в нижнем углу приложения.

Затем раскроем группу *Услуги* и тоже создадим в ней несколько элементов – услуги по ремонту телевизоров (рис. 3.51).

- *Диагностика,*
- *Ремонт отечественного телевизора*,
- *Ремонт импортного телевизора*.

И услуги по установке стиральных машин:

- *Подключение воды,*
- *Подключение электричества*.

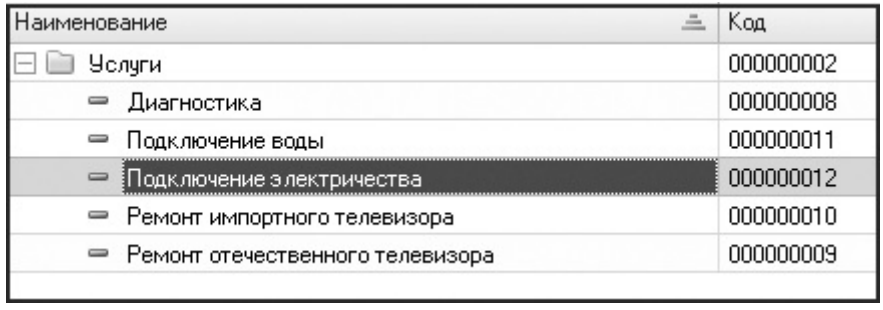


Рис. 3.51. Создание элементов в группе «Услуги»

Перенос элементов в другие группы

Теперь разнесем услуги по двум смысловым группам: услуги по ремонту телевизоров и услуги по установке стиральных машин.

Для этого в группе *Услуги* создадим еще две группы: *Телевизоры* и *Стиральные машины* (рис. 3.52).

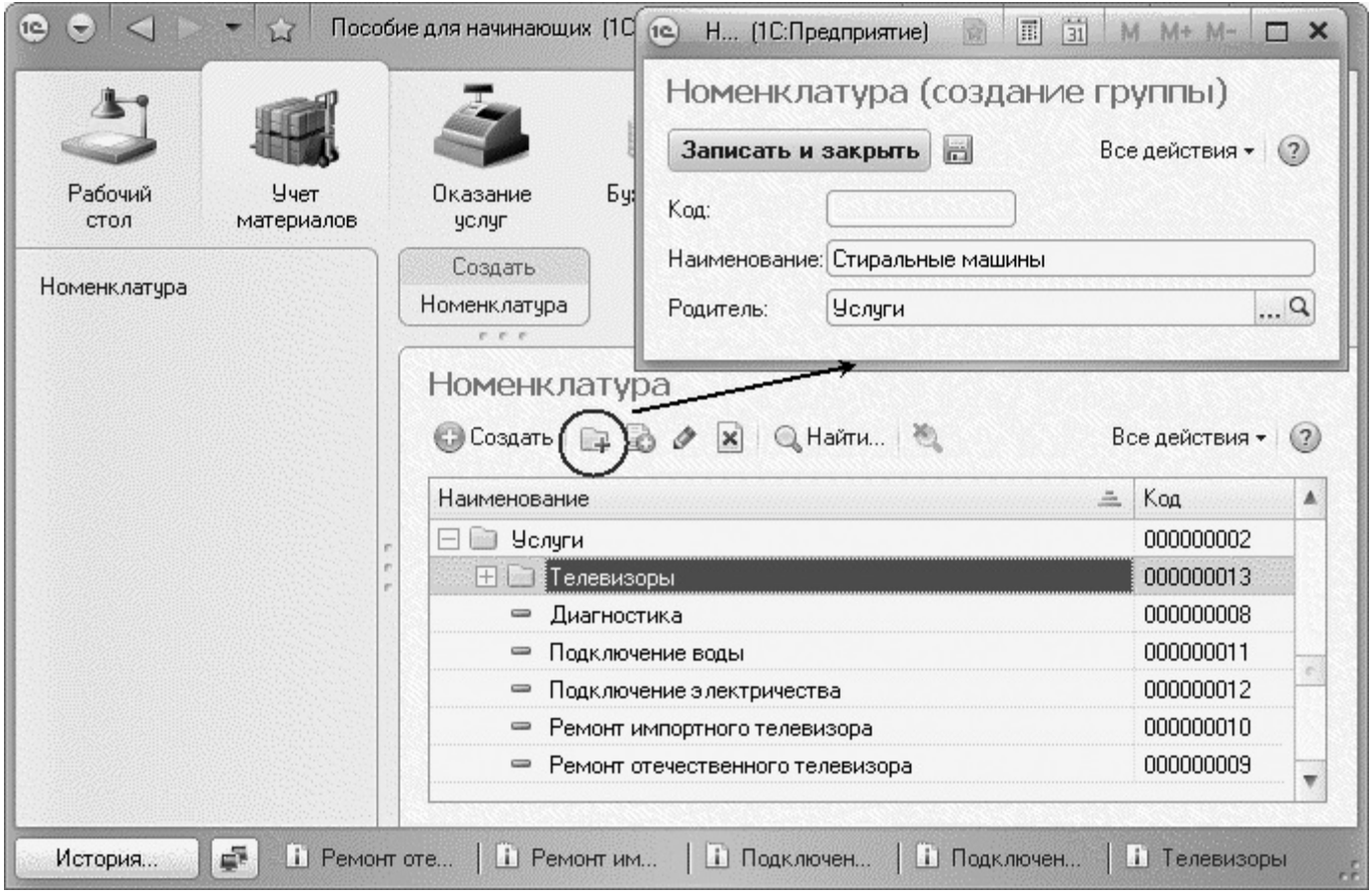


Рис. 3.52. Создание новых групп в группе «Услуги»

Для того чтобы переместить услуги в соответствующие группы, в окне списка установим курсор на ту услугу, которую мы хотим переместить, и выполним

команду *Все действия > Переместить в группу*. В открывшемся окне выберем новую группу.

Можно выделить в списке сразу несколько элементов (левой кнопкой мыши, удерживая при этом клавишу *Ctrl*) и переместить их все сразу. Или же можно мышью перетащить выделенный элемент справочника в нужную группу.

Можно также открыть для редактирования выделенный элемент справочника и изменить поле *Родитель* (рис. 3.53).

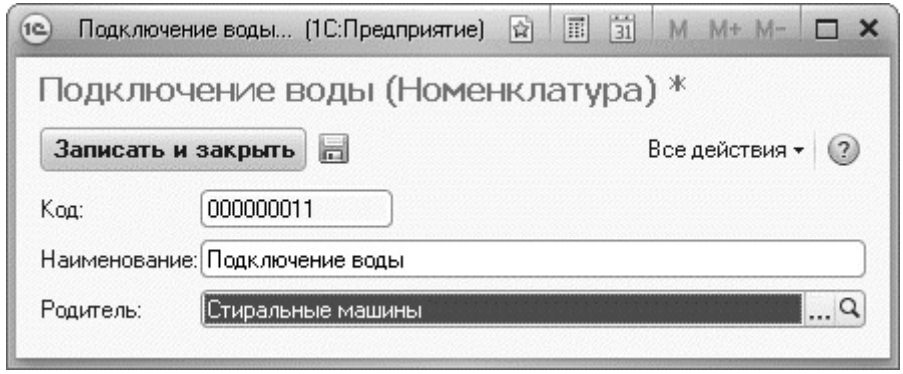


Рис. 3.53. Перенос элемента справочника в другую группу

Переместим в группу *Телевизоры* услуги *Диагностика, Ремонт отечественного телевизора* и *Ремонт импортного телевизора*.

Услуги *Подключение воды* и *Подключение электричества* переместим в группу *Стиральные машины*.

Затем в группе *Материалы* создадим две группы: *Радиодетали* и *Прочее*. В группу *Прочее* поместим *Кабель электрический* и *Шланг резиновый*. Остальные материалы переместим в группу *Радиодетали*.

Если теперь переключить представление списка в виде дерева (*Все действия – Режим просмотра – Дерево*), то мы увидим, что список номенклатуры будет представлен в виде следующего дерева (рис. 3.54).

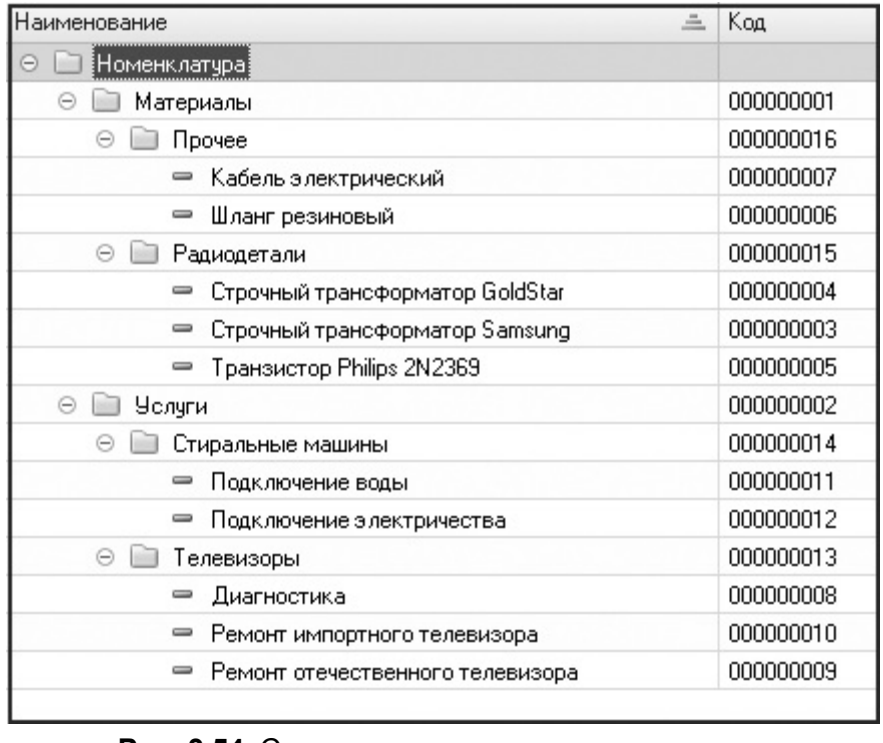


Рис. 3.54. Список номенклатуры в виде дерева

Справочник с предопределенными элементами

В заключение мы создадим справочник *Склады*, который будет содержать информацию о складах, используемых ООО «На все руки мастер».

Этот справочник будет включать в себя один предопределенный элемент –

склад *Основной*, на который будут поступать все материалы.

Наша задача – создать справочник, содержащий предопределенные элементы.

В режиме «Конфигуратор»

Откроем конфигуратор и создадим новый объект конфигурации Справочник с именем *Склады*. На основании имени платформа автоматически заполнит его синоним.

Зададим *Представление объекта* как *Склад*. Вместо свойства *Представления списка* будет использоваться *Синоним* объекта – *Склады* (рис. 3.55).

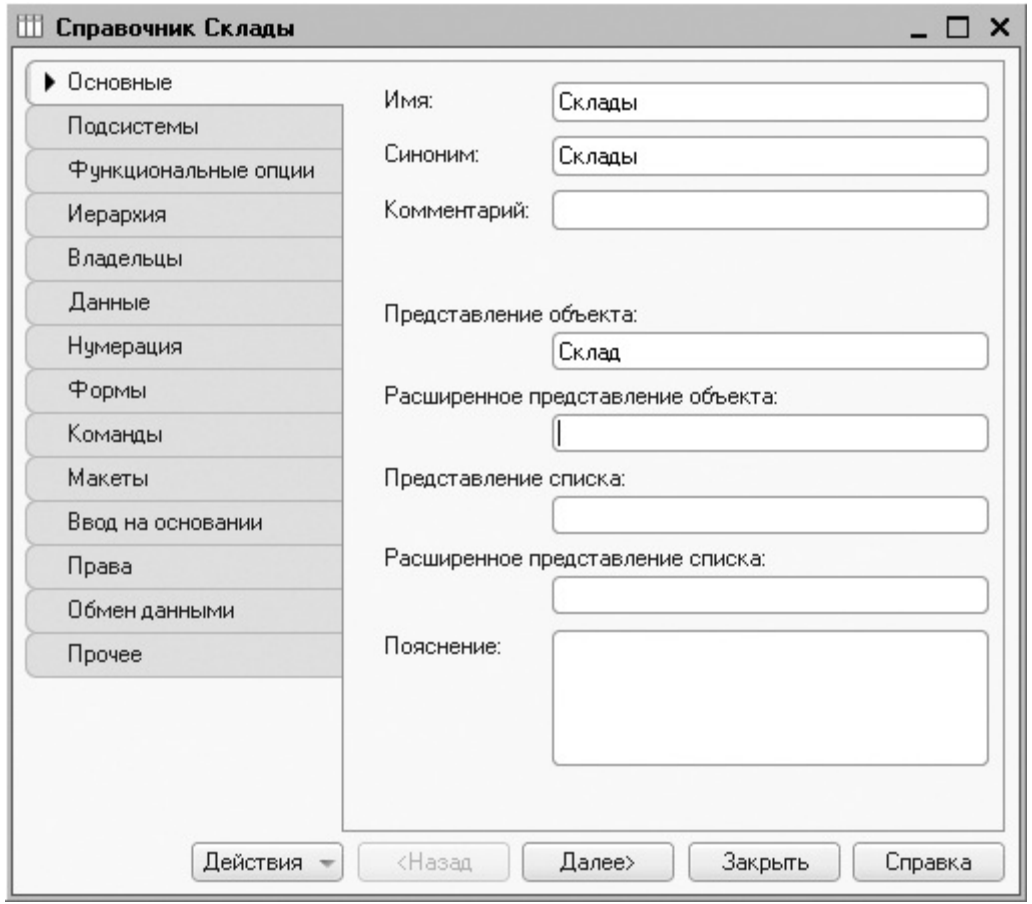


Рис. 3.55. Установка основных свойств справочника

Перейдем на закладку *Подсистемы*.

По логике нашей конфигурации список складов должен быть доступен в разделах *Оказание услуг* и *Учет материалов*, так как поступление материалов и оказание услуг, как правило, учитываются в разрезе складов. Поэтому отметим в списке подсистем эти подсистемы (рис. 3.56).

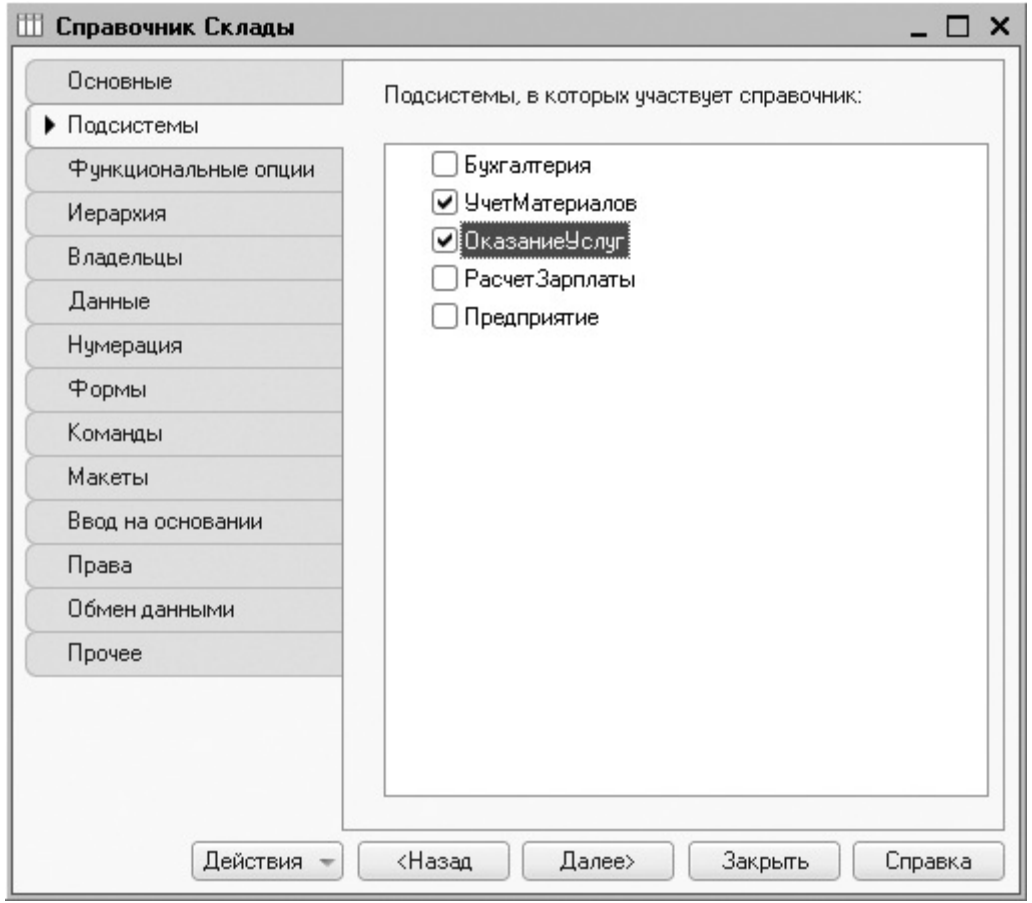


Рис. 3.56. Определение списка подсистем, в которых отображается справочник

Свойство «Быстрый выбор»

Заполним еще одно свойство справочника *Склады – Быстрый выбор.* Для

этого перейдем на закладку *Формы* и установим соответствующий флажок (рис. 3.57).

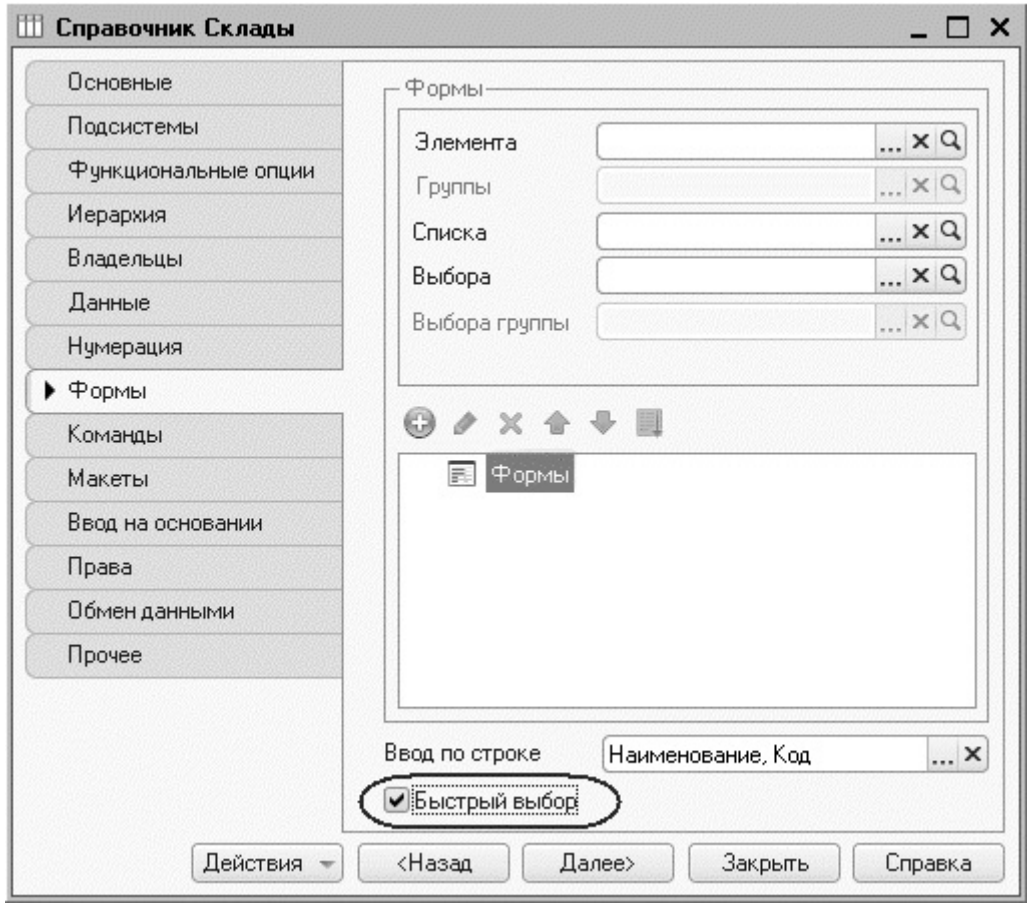


Рис. 3.57. Установка свойства «Быстрый выбор»

Дело в том, что по умолчанию при нажатии кнопки выбора ... в поле, содержащем ссылку на элемент справочника, открывается основная форма выбора элемента справочника. Она может быть не всегда удобна, особенно в том случае, когда справочник неиерархический и заведомо содержит небольшое количество элементов.

Свойство *Быстрый выбор* как раз позволяет выбирать элементы не из отдельной формы, а из небольшого выпадающего списка, заполненного элементами этого справочника (рис. 3.58).

Рис. 3.58. Выбор склада из выпадающего списка

Этот вариант наиболее удобен для списка складов, так как их, вероятно, будет немного.

Для остальных справочников свойство *Быстрый выбор* мы не устанавливали, так как *Номенклатура* – иерархический справочник, и, следовательно, быстрый выбор для него не имеет смысла. А список сотрудников и, особенно, список клиентов может быть очень большим, и выпадающий список в этом случае будет неудобно прокручивать.

Предопределенные элементы

Перейдем на закладку *Прочее* и нажмем кнопку *Предопределенные*.

Система откроет список предопределенных элементов справочника.

Сейчас он пуст, поэтому нажмем кнопку *Добавить* и создадим предопределенный элемент с именем *Основной* (рис. 3.59).

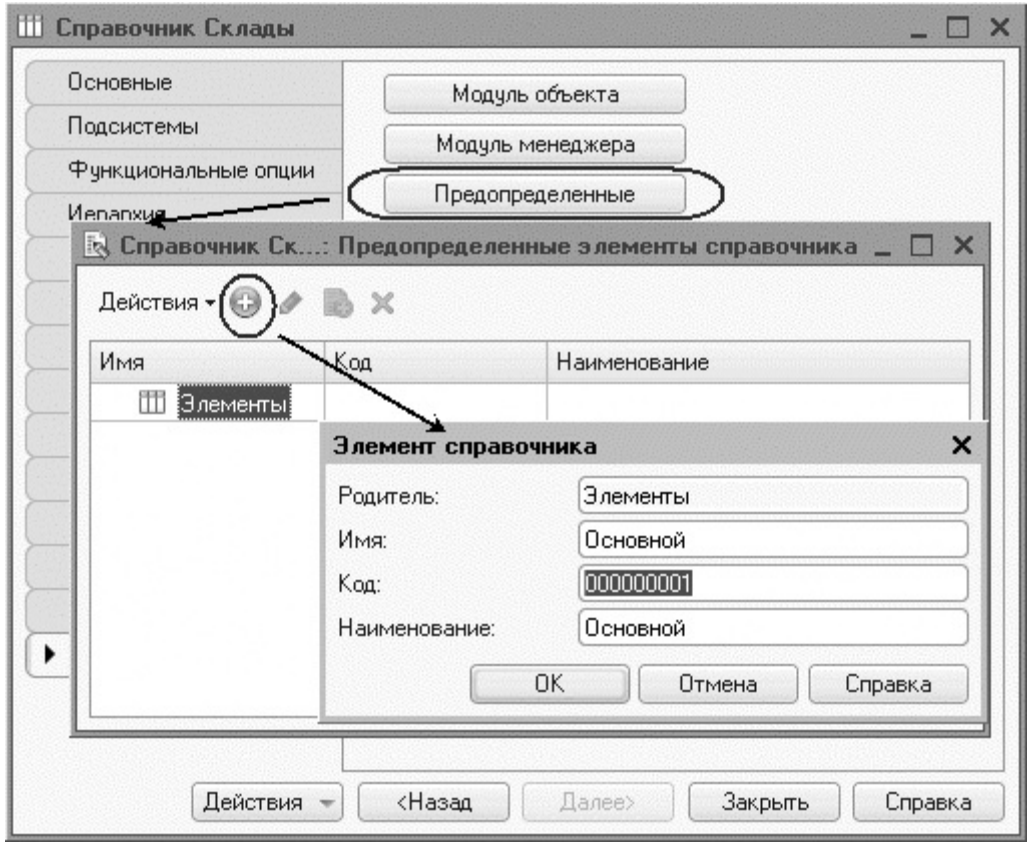


Рис. 3.59. Заполнение предопределенного элемента справочника

Обратите внимание на то, что помимо наименования мы задали еще и *имя* предопределенного элемента справочника.

В дальнейшем, когда мы будем использовать встроенный язык, мы сможем

обратиться к этому элементу справочника, используя имя, присвоенное ему в конфигураторе. Дело в том, что наименование предопределенного элемента справочника пользователь может изменить, а имя пользователь не видит и изменить не может.

Прежде чем запускать «1С:Предприятие», настроим интерфейс приложения, чтобы нам было удобнее вводить новые элементы справочника. В панели действий разделов *УчетМатериалов* сделаем доступной стандартную команду для создания новых складов.

Для этого в дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*.

В открывшемся окне слева в списке *Подсистемы* выделим подсистему *УчетМатериалов*.

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель действий.Создать* включим видимость у команды *Склад: создать*.

Также мы видим, что в группу *Панель навигации.Обычное* добавилась команда

Склады для открытия этого списка. Она включена по умолчанию (рис. 3.60).

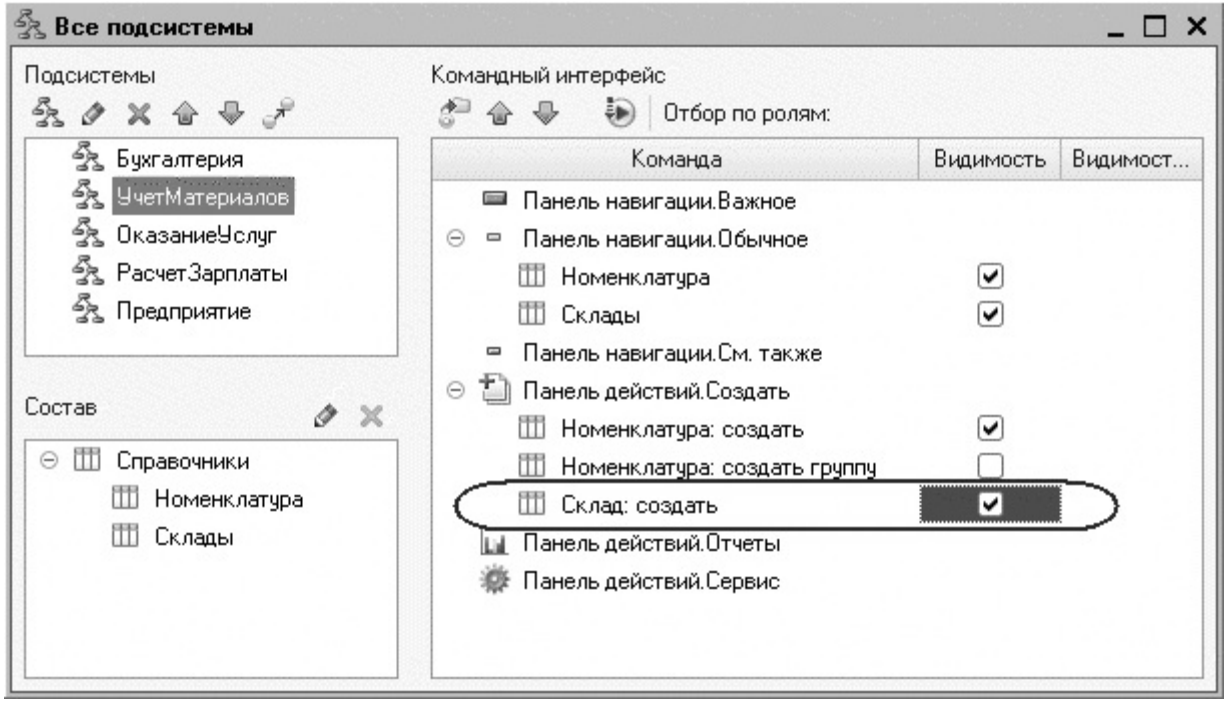


Рис. 3.60. Окно настройки подсистем

Закроем окно редактирования справочника *Склады* и запустим «1С:Предприятие» в режиме отладки.

Ответим утвердительно на запрос конфигуратора об обновлении конфигурации и увидим окно, содержащее список изменений в структуре конфигурации,

автоматически сгенерированный платформой. В данном случае мы добавили справочник *Склады*.

Нажмем кнопку *Принять* (рис. 3.61).

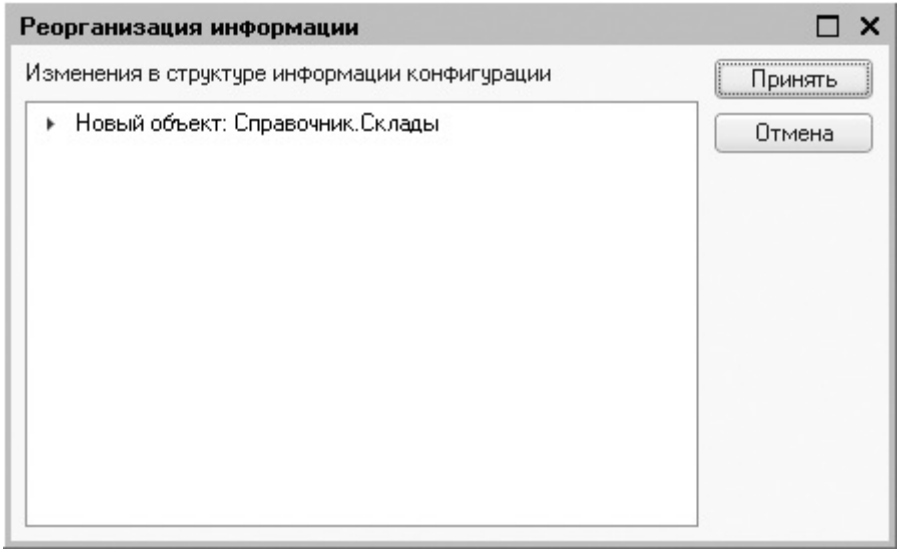


Рис. 3.61. Список изменений в структуре конфигурации

В режиме «1С:Предприятие»

В открывшемся окне «1С:Предприятия» мы видим, что в панели действий раздела *Учет материалов* появилась команда *Склад* для создания новых складов.

Название этой команды определяется свойством *Представление объекта*, которое мы задали для этого справочника (рис. 3.62).

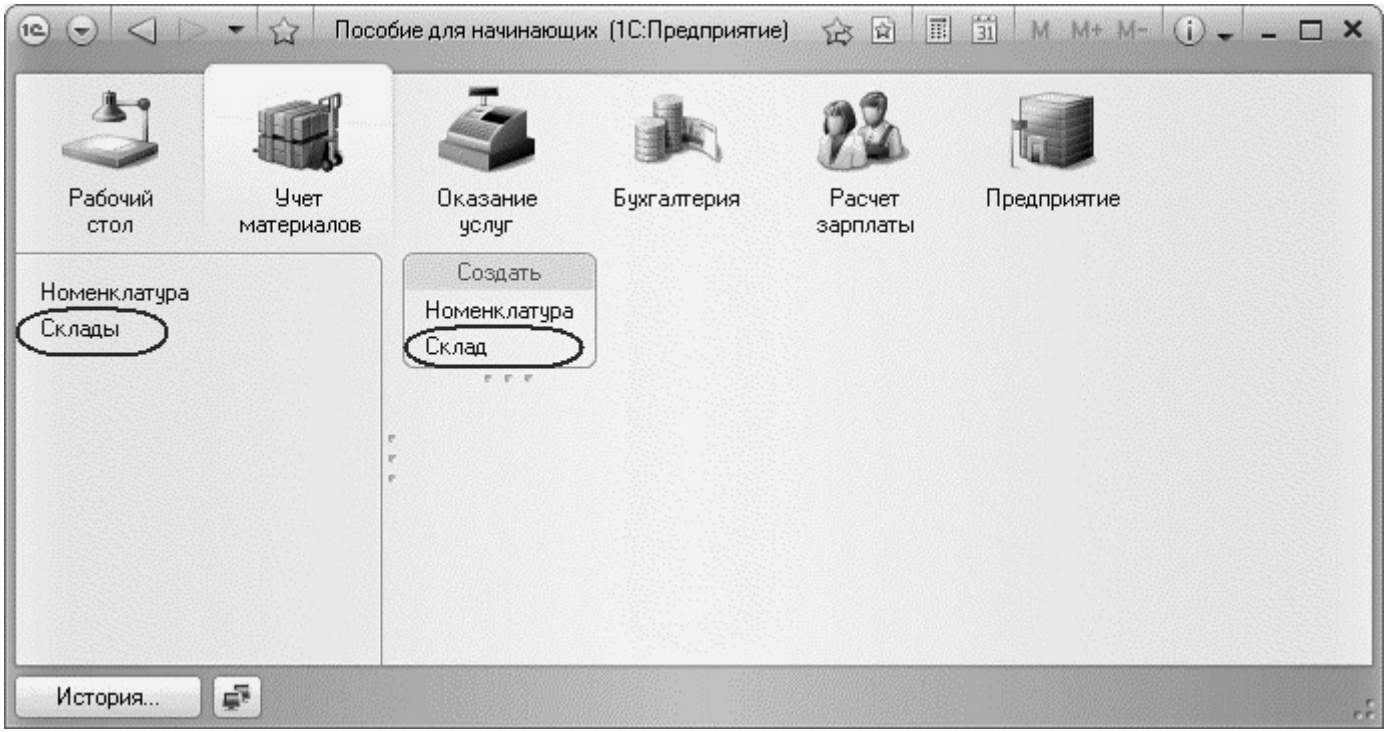


Рис. 3.62. Раздел «Учет материалов»

Также в панели навигации разделов *Оказание услуг* и *Учет материалов* появилась команда *Склады* для открытия списка складов.

Название этой команды определяется синонимом объекта, так как *Представление списка* мы для этого справочника не задавали (см. рис. 3.62).

Выполним команду *Склады* в панели навигации раздела *Учет материалов*.

Справа от панели навигации в рабочей области окна приложения откроется основная форма списка.

В списке складов уже есть один элемент с наименованием *Основной*. Это предопределенный элемент, который мы создали в конфигураторе.

Выполнив команду *Склад* в панели действий, добавим в справочник еще один склад, который назовем *Розничный* (рис. 3.63).

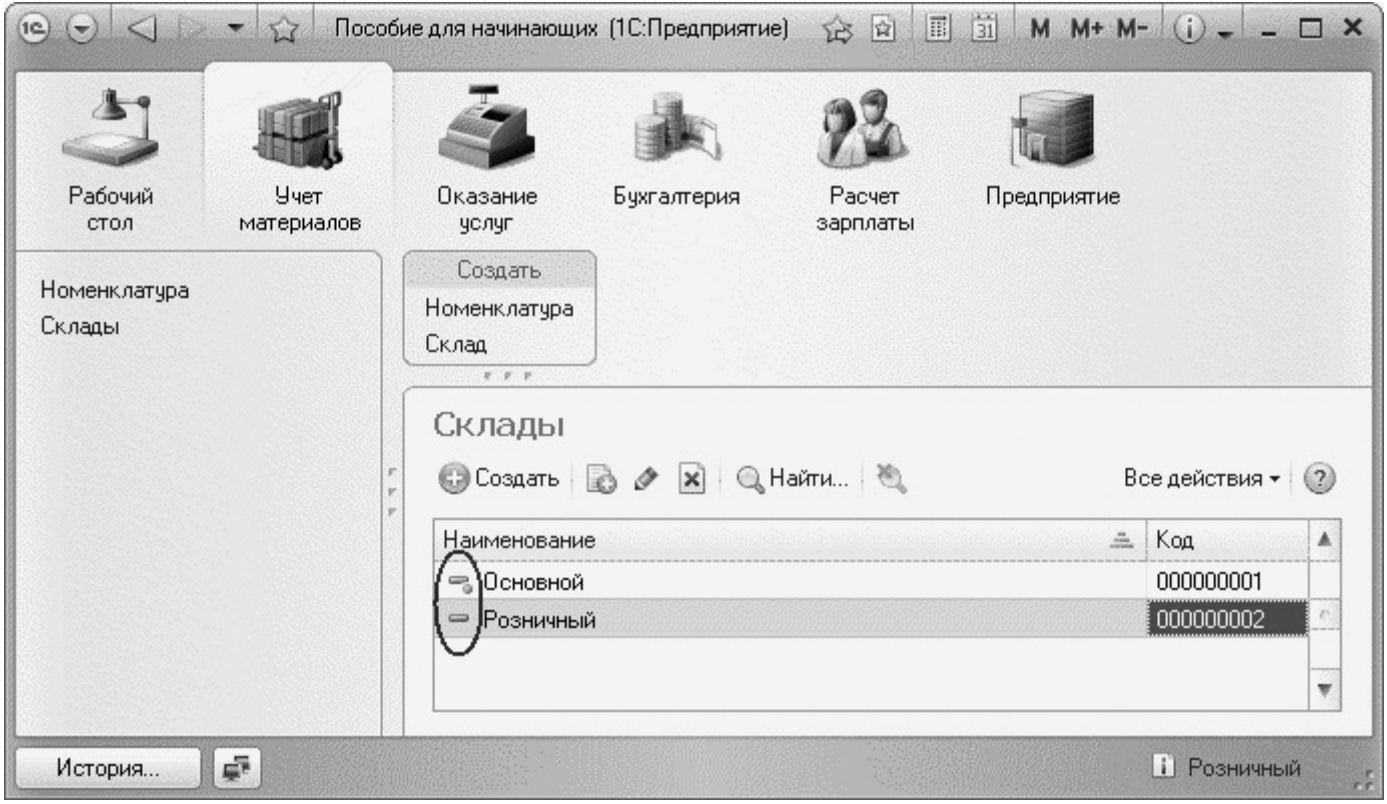


Рис. 3.63. Элементы справочника «Склады»

На этом мы завершим подготовительную работу по созданию справочников и сделаем теоретическое отступление, касающееся предопределенных элементов и тех вопросов, которые постоянно появляются у вас на экране при запуске и продолжении отладки.

«Теория». Предопределенные элементы

Обратите внимание, что система отмечает различными пиктограммами обычный и предопределенный элементы справочника.

Несмотря на то, что можно изменить код или наименование у обоих элементов, пометка на удаление (или удаление) возможна только для обычных элементов справочника. При попытке пометить на удаление предопределенный элемент система выдаст предупреждение (рис. 3.64).

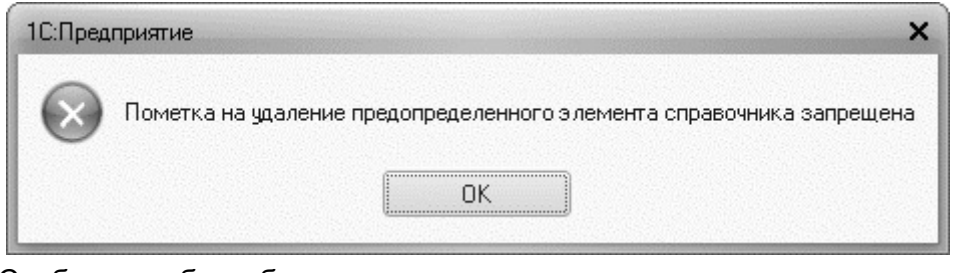


Рис. 3.64. Сообщение об ошибке при удалении предопределенного элемента справочника

Таким образом мы можем обозначить две характерные особенности предопределенных элементов:

- на предопределенные элементы могут опираться алгоритмы работы конфигурации (т. к. возможно обращение к ним из встроенного языка по имени);

- предопределенные элементы являются объектами базы данных, которые нельзя удалить в режиме *1С:Предприятие*.

Из этого видно, в чем заключается принципиальная с точки зрения конфигурации разница между обычными и предопределенными элементами справочника.

Обычные элементы непостоянны для конфигурации. В процессе работы пользователя они могут появиться, исчезнуть. Поэтому конфигурация хоть и может отличить их друг от друга, но рассчитывать на них в выполнении каких-либо алгоритмов она не может в силу их непостоянства.

Предопределенные элементы, напротив, постоянны. В процессе работы пользователя они находятся всегда на своих местах и исчезнуть не могут. Поэтому конфигурация может работать с ними вполне уверенно и опираться на них при отработке различных алгоритмов. По этой причине каждый из предопределенных элементов имеет уникальное имя для того, чтобы к нему можно было обратиться средствами встроенного языка.

Основная конфигурация и конфигурация базы данных

До сих пор мы не углублялись в структуру системы «1С:Предприятие 8», но теперь пришло время сказать об этом несколько слов.

Вспомните, с точки зрения пользователя, «программа 1С» состоит из платформы и конфигурации. Мы говорили, что в каждом конкретном случае используется одна из множества возможных конфигураций. Настало время сказать, что это не совсем так.

Почему *не* так? Потому что в каждой информационной базе существуют как минимум две конфигурации.

Почему не *совсем* так? Потому что пользователь действительно работает всегда только с одной конфигурацией. Вторая конфигурация предназначена для разработчика или человека, который должен вносить изменения в конфигурацию (например, администратора базы данных). Для пользователя она «не видна».

Конфигурация, предназначенная для разработчика, называется *Основная конфигурация* (или просто *Конфигурация* – та, которую мы редактировали в конфигураторе).

Конфигурация, с которой работают пользователи, называется *Конфигурация базы данных*.

Основную конфигурацию можно редактировать. Конфигурацию базы данных редактировать нельзя, можно только произвести обновление конфигурации

базы данных на основе основной конфигурации.

Однако у вас может возникнуть естественный вопрос: если у нас есть две конфигурации – одна, которую можно редактировать, и другая, с которой работают пользователи, то почему же тогда основной называется редактируемая конфигурация? Ведь с точки зрения конечного продукта основной является именно конфигурация, с которой работают пользователи!

Дело в том, что в данном случае название «основная» дано с точки зрения разработчика, и это имеет глубокий практический смысл.

В общем случае информационная база «1С:Предприятия» может хранить более двух конфигураций: основную конфигурацию, конфигурацию базы данных и несколько конфигураций поставщиков.

Кроме этого, вне информационной базы может существовать хранилище. В нем находится конфигурация, предназначенная для групповой разработки.

Вне информационной базы может существовать также некоторое количество файлов конфигураций, в том числе файлы новой поставки (рис. 3.65).

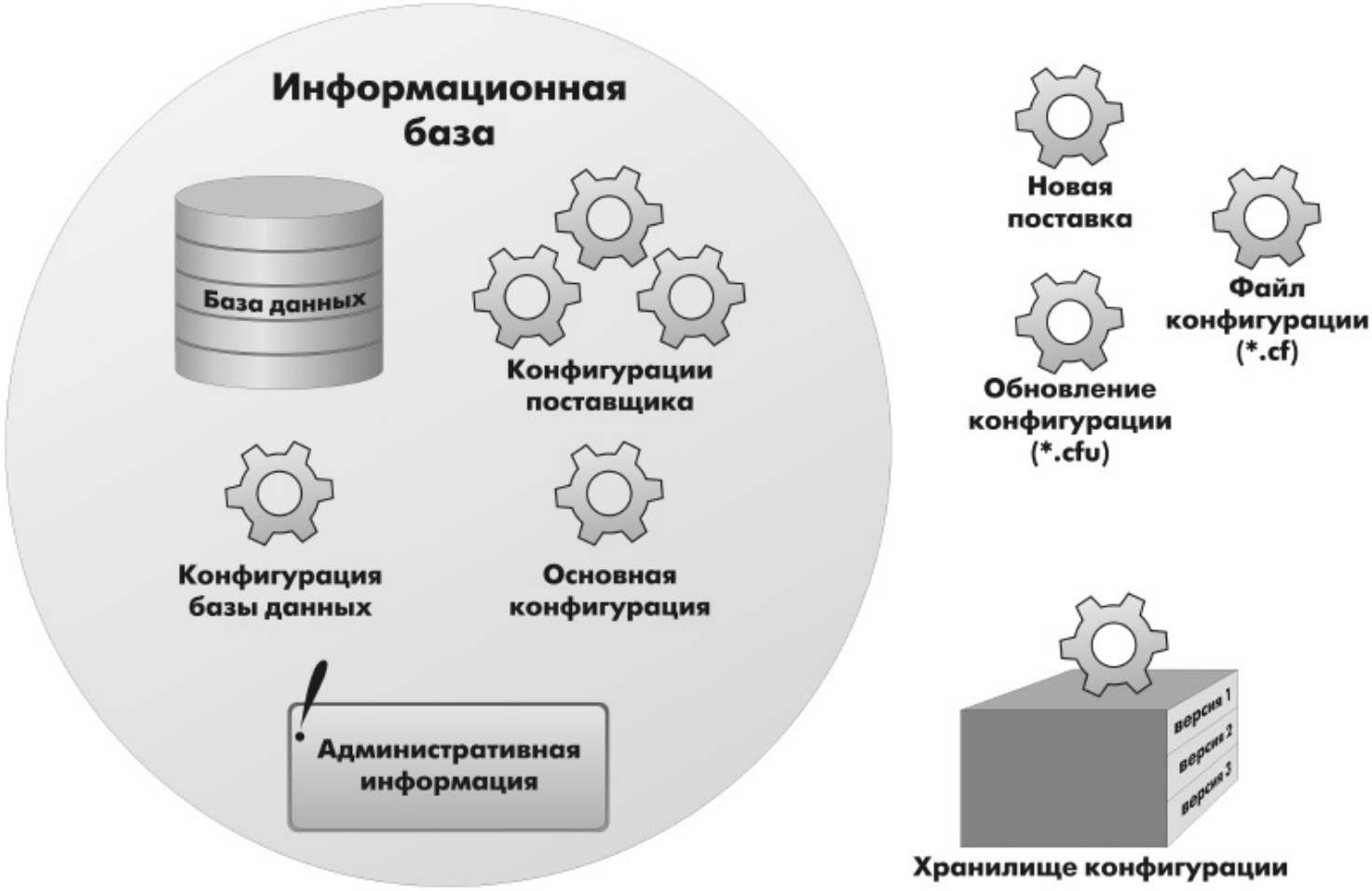


Рис. 3.65. Структура конфигурации

Конфигурация поставщика, находящаяся в информационной базе, содержит

предыдущее состояние поставки. Возможна ситуация, когда конфигурация находится на поддержке одновременно у нескольких поставщиков, каждый из которых поддерживает только свою часть в виде отдельной конфигурации. В этом случае информационная база будет хранить несколько конфигураций поставщиков (состояние предыдущей поставки для каждого поставщика отдельно).

Файлы новой поставки могут существовать в виде файлов конфигураций (*полная поставка*) и файлов обновлений (*поставка обновлений*).

Хранилище конфигурации содержит конфигурацию, предназначенную для групповой разработки. Она хранится не в виде единой конфигурации, а в виде отдельных объектов в разрезе версий конфигурации. Таким образом, мы можем получить из хранилища конфигурацию любой версии – для этого она «собирается» из объектов нужной версии.

Теперь представьте, что между всеми этими видами конфигураций существует возможность сравнения и обновления. В этом случае очень легко запутаться, и название *Основная конфигурация* как нельзя лучше отражает конечную цель всех изменений.

Теперь, возвращаясь к основной конфигурации и конфигурации базы данных, нужно заметить, что внутреннее разделение на две конфигурации позволяет

вносить изменения, не прерывая работы пользователей, потому что изменения вносятся в основную конфигурацию, с которой пользователи не работают.

Затем, когда разработчик будет уверен в том, что все изменения, которые он внес, верны, можно будет быстро произвести обновление конфигурации базы данных, используя основную конфигурацию.

Если эти изменения не затрагивают структуру базы данных (например, если не нужно изменять таблицы, если поменялся только программный код в каком-то модуле), то обновить конфигурацию базы данных можно не прерывая работы пользователей. Это так называемое *динамическое обновление*.

Пользователи увидят изменения только после того, как перезапустят свое приложение. Используя метод встроенного языка *КонфигурацияБазыДанныхИзменена Динамически(),* можно определить программно, нужно ли перезапускать приложение.

Но если изменения касаются структуры базы данных, например, добавился новый реквизит у справочника или изменился тип существующего реквизита, то тогда нужно завершить работу всех пользователей.

Разработчик всегда может сравнить основную конфигурацию и конфигурацию базы данных, может вернуться к исходному состоянию основной конфигурации,

используя конфигурацию базы данных (если, например, совсем запутался в своих изменениях).

Таким образом, взаимодействие двух конфигураций можно представить следующим образом (рис. 3.66).

Рис. 3.66. Взаимодействие двух конфигураций

Когда разработчик работает с основной конфигурацией, система всегда подсказывает ему, отличается ли его вариант основной конфигурации от того, который сохранен, и отличается ли сохраненный вариант основной конфигурации от конфигурации базы данных.

Если разработчик редактирует основную конфигурацию и редактируемый вариант основной конфигурации отличается от того, который сохранен, в заголовке окна дерева конфигурации появляется признак модифицированности конфигурации (*), рис. 3.67.

Рис. 3.67. Заголовок окна дерева конфигурации

Если сохраненный вариант основной конфигурации отличается от конфигурации базы данных, в заголовке окна дерева конфигурации появляется знак отличия конфигураций (<*!*>), рис. 3.68.

Рис. 3.68. Заголовок окна дерева конфигурации

Для сохранения основной конфигурации следует использовать команду *Конфигурация* > *Сохранить конфигурацию*, а для обновления конфигурации базы данных – команду *Конфигурация* > *Обновить конфигурацию базы данных*.

При выполнении команды *Отладка* > *Начать отладку* система сама сначала сохраняет основную конфигурацию, а затем производит ее сравнение с конфигурацией базы данных. Если конфигурации отличаются, выдается запрос на обновление конфигурации базы данных, который вы видели в предыдущих

примерах.

При выполнении команды *Отладка* > *Продолжить* система, после описанных выше действий, предлагает еще и перезапустить приложение, чтобы прекратить текущий сеанс, запущенный в режиме отладки.

Таким образом, система старается облегчить жизнь разработчика и автоматизировать часто выполняемые операции.

Важным фактом является то, что именно в момент обновления конфигурации базы данных система создает (модифицирует) в базе данных те структуры хранения данных, которые мы описали в виде объектов конфигурации.

Таким образом, обычные элементы справочника пользователь добавляет в ту структуру базы данных, которую создала система на основе объекта конфигурации *Справочник*.

Предопределенные элементы этого справочника система добавляет в эту структуру сама, на основе все того же описания этой структуры, которым является объект конфигурации *Справочник*.

Отсюда следует немаловажный факт (о котором говорилось в предыдущем разделе), что если простые элементы справочника «безразличны» для

конфигурации, то предопределенные элементы важны для нее, поскольку на них могут быть завязаны алгоритмы работы конфигурации.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Справочник».*
- *Каковы характерные особенности справочника.*
- *Для чего используются реквизиты и табличные части справочника.*
- *Зачем нужны иерархические справочники и что такое родитель.*
- *Зачем нужны подчиненные справочники и что такое владелец.*
- *Какие основные формы существуют у справочника.*
- *Что такое предопределенные элементы.*
- *Чем с точки зрения конфигурации отличаются обычные элементы справочника от предопределенных элементов.*
- *Как пользователь может отличить обычные элементы справочника от предопределенных элементов.*
- *Как создать объект конфигурации «Справочник» и описать его структуру.*
- *Как добавить новые элементы в справочник.*
- *Как создать группу справочника.*

- *Как переместить элементы из одной группы справочника в другую.*
- *Зачем нужна основная конфигурация и конфигурация базы данных.*
- *Как изменить конфигурацию базы данных.*
- *Как связаны объекты конфигурации и объекты базы данных.*
- *Что такое подчиненные объекты конфигурации.*
- *Зачем нужна проверка заполнения у реквизитов справочника.*
- *Что такое быстрый выбор и когда его использовать.*
- *Как отобразить справочник и определить его представление в различных разделах интерфейса приложения.*
- *Как отобразить команды создания нового элемента справочника в интерфейсе подсистем.*
- *Как редактировать командный интерфейс подсистем.*

Занятие 4 (1:30). Документы

Продолжительность

Ориентировочная продолжительность занятия – 1 час 30 минут.

На этом занятии мы познакомимся с объектом конфигурации *Документ*. Вы узнаете, для чего он нужен, какова его структура и какими основными свойствами он обладает.

Затем мы создадим несколько документов и покажем, каким образом разработчик может задавать собственные алгоритмы выполнения тех или иных действий, связанных с работой документа.

Кроме этого, вы узнаете, как создать форму документа, познакомитесь с некоторыми конструкциями встроенного языка и узнаете, что такое типообразующие объекты конфигурации.

Что такое документ

Объект конфигурации *Документ* предназначен для описания информации о совершенных хозяйственных операциях или о событиях, произошедших в жизни организации вообще. Как правило, в работе любой фирмы используются такие документы, как приходные накладные, приказы о приеме на работу, платежные

поручения, счета и т. д. Свойства и структура этих документов описываются в объектах конфигурации *Документ*, на основе которых платформа создает в базе данных таблицы для хранения информации из этих документов.

Логика работы документов отличается от логики работы других объектов конфигурации. Документ обладает способностью *проведения*. Факт проведения документа означает, что событие, которое он отражает, повлияло на состояние учета.

До тех пор, пока документ не проведен, состояние учета неизменно, и документ – не более чем черновик, заготовка. Как только документ будет проведен, изменения, вносимые документом в учет, вступят в силу и состояние учета будет изменено.

Поскольку документ вносит изменения в состояние учета, он всегда «привязан» к конкретному моменту времени. Это позволяет отражать в базе данных фактическую последовательность событий.

Следующим важным фактом, вытекающим из двух предыдущих, является то, что система «1С:Предприятие» имеет механизмы, позволяющие отслеживать правильность состояния учета. Предположим, что мы изменили один из проведенных ранее документов и снова провели его задним числом. В этом случае система «1С:Предприятие» способна отследить, повлияют ли внесенные

нами изменения на последующие проведенные документы, и, если это так, система способна перепровести необходимые документы.

В процессе работы пользователь может самостоятельно создавать новые документы – приходные и расходные накладные, счета и т. п.

В базе данных каждый документ представляет собой отдельную запись в основной таблице, хранящей информацию об этом виде документов (рис. 4.1).

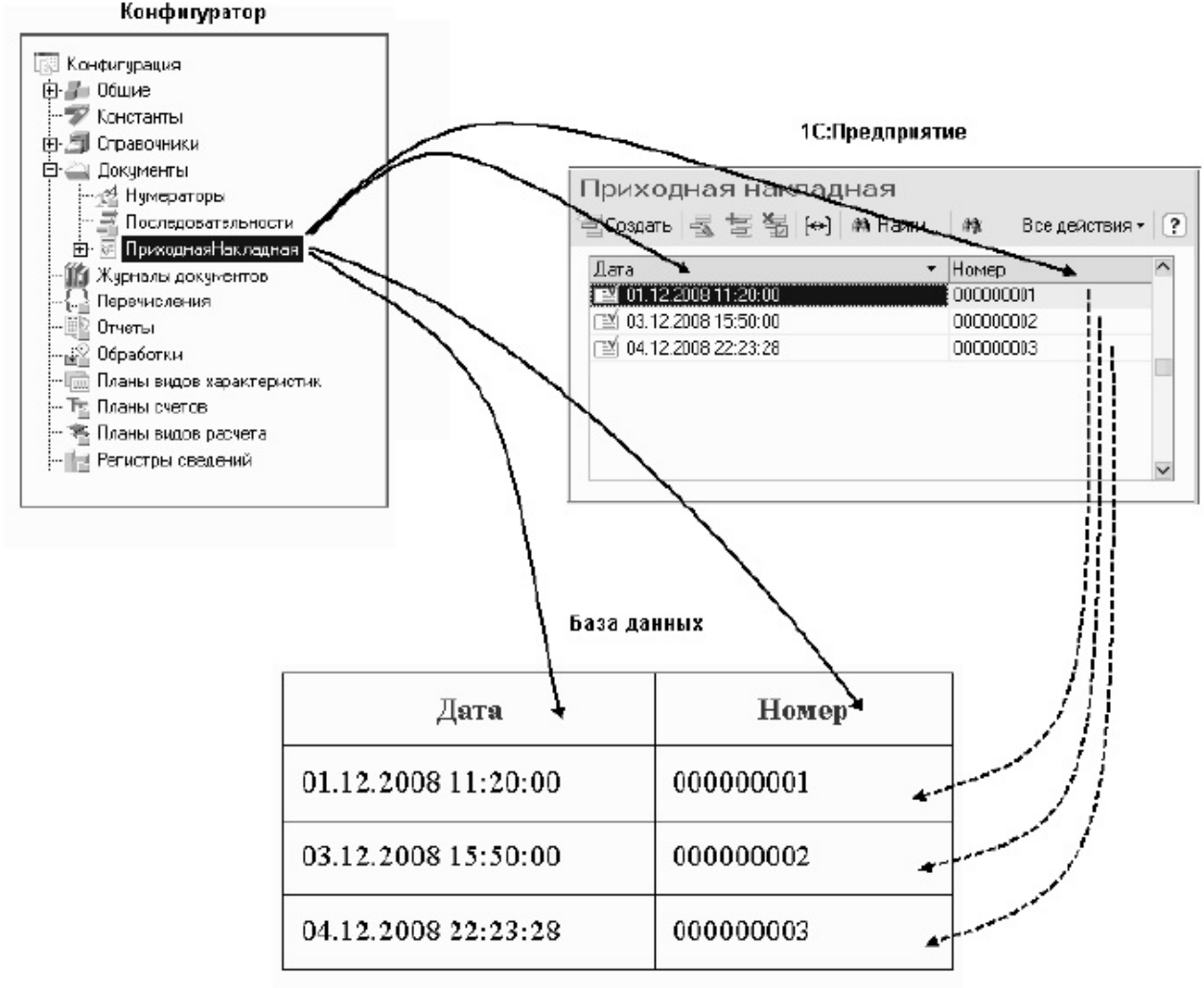
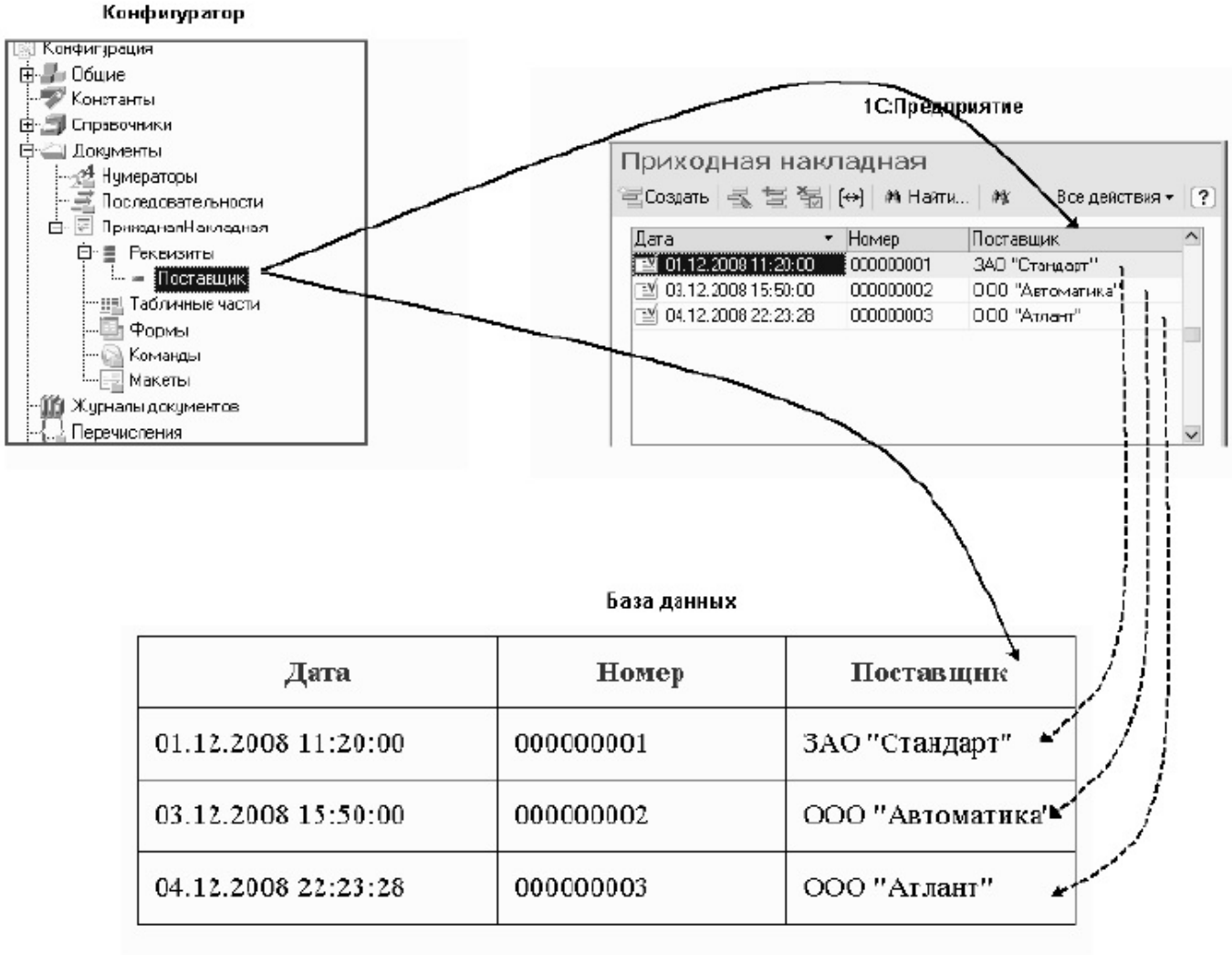

Конфигуратор
Конфигурация
Общие
Константы
Справочники
Документы
Нумераторы
Последовательности
ПриходнаяНакладная
Журналы документов
Перечисления
Отчеты
Обработки
Планы видов характеристик
Планы счетов
Планы видов расчета
Регистры сведений
1С:Предприятие
Приходная накладная
Создать
Найти
Все действия
Дата
Номер
01.12.2008 11:20:00
000000001
03.12.2008 15:50:00
000000002
04.12.2008 22:23:28
000000003
База данных
Дата
Номер
01.12.2008 11:20:00
000000001
03.12.2008 15:50:00
000000002
04.12.2008 22:23:28
000000003

Рис. 4.1. Стандартные реквизиты документа «Приходная накладная» в режиме «Конфигуратор», в режиме «1С:Предприятие» и в базе данных

Каждый документ, как правило, содержит информацию, которая подробнее описывает этот документ. Например, каждый документ *Приходная накладная* может содержать информацию о поставщике товаров, складе, на который приходуется товар, и т. д.

Набор такой информации является одинаковым для всех документов одного вида, и для описания такого набора используются реквизиты объекта конфигурации *Документ*, являющиеся подчиненными объектами конфигурации. Большинство реквизитов разработчик создает самостоятельно, однако у каждого объекта конфигурации *Документ* по умолчанию существуют стандартные реквизиты. Два наиболее важных из них – это *Дата* и *Номер*. Поскольку тип данных *Дата* содержит дату и время с точностью до секунды, этот реквизит и определяет в основном положение документа на оси времени (рис. 4.2).



Дата	Номер	Поставщик
01.12.2008 11:20:00	000000001	ЗАО "Стандарт"
03.12.2008 15:50:00	000000002	ООО "Автоматика"
04.12.2008 22:23:28	000000003	ООО "Атлант"

Рис. 4.2. Реквизит «Поставщик» документа «Приходная накладная» в режиме «Конфигуратор», в режиме «1С:Предприятие» и в базе данных

Кроме этого, каждый документ содержит, как правило, некоторый набор информации, которая одинакова по своей структуре, но различна по количеству и предназначена для разных документов. Так, например, каждый документ *Приходная накладная* может содержать список приходуемых товаров.

Для описания подобной информации служат табличные части объекта конфигурации *Документ*. В этом случае в базе данных будут созданы дополнительные таблицы для хранения табличных частей, подчиненных конкретному документу (рис. 4.3).

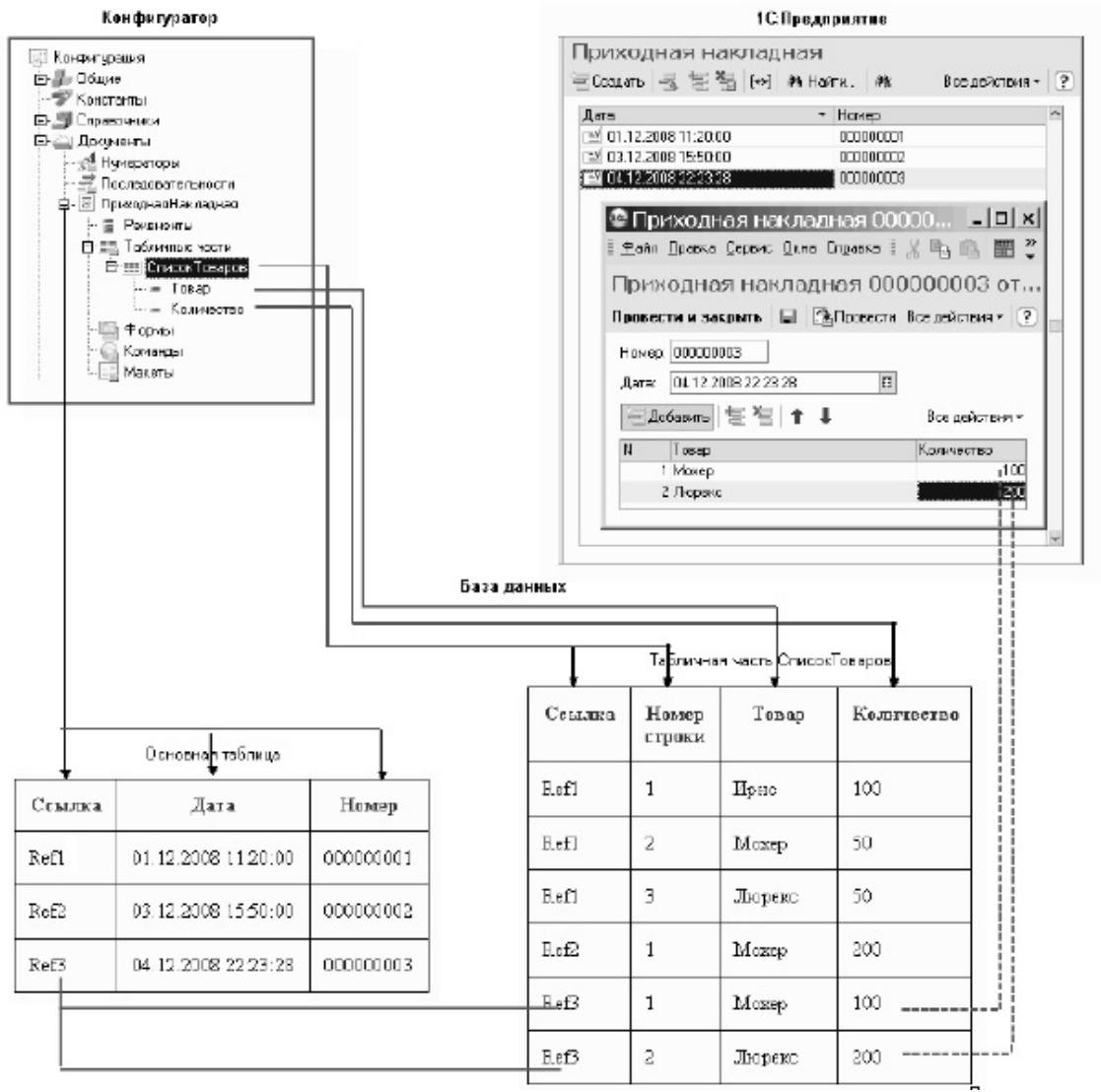

Конфигуратор
Конфигурация
Общие
Константы
Справочники
Документы
Нумераторы
Последовательности
ПриходнаяНакладная
Реквизиты
Табличные части
СписокТоваров
Товар
Количество
Формы
Команды
Макеты
1С:Предприятие
Приходная накладная
Создать
Найти...
Все действия
Дата
Номер
01.12.2008 11:20:00 000000001
03.12.2008 15:50:00 000000002
04.12.2008 22:23:28 000000003
Приходная накладная 00000...
Файл Правка Сервис Окно Справка
Приходная накладная 000000003 от...
Провести и закрыть
Провести
Все действия
Номер 000000003
Дата 04.12.2008 22:23:28
Добавить
Все действия
N
Товар
Количество
1 Мокер 100
2 Лорекс 200
База данных
Основная таблица
Ссылка
Дата
Номер
Ref1
01.12.2008 11:20:00
000000001
Ref2
03.12.2008 15:50:00
000000002
Ref3
04.12.2008 22:23:28
000000003
Табличная часть СписокТоваров
Ссылка
Номер строки
Товар
Количество
Ref1 1 Пресс 100
Ref1 2 Мокер 50
Ref1 3 Лорекс 50
Ref2 1 Мокер 200
Ref3 1 Мокер 100
Ref3 2 Лорекс 200

Рис. 4.3. Табличная часть «Список товаров» документа «Приходная накладная» в режиме «Конфигуратор», в режиме «1С:Предприятие» и в базе данных

Формы документа

Для визуализации документа существует несколько основных форм, которые, как мы уже говорили, имеют несколько вариантов названий (табл. 4.1).

Таблица 4.1. Основные формы документа

В контекстном меню и в палитре свойств	В конструкторе форм	На закладке формы
Форма объекта	Форма документа	Документа
Форма списка	Форма списка документа	Списка
Форма для выбора	Форма выбора документа	Выбора

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с документами, можно прочитать в разделе «Краткий справочник

разработчика. Документы».

«Теория». Типы данных. Типообразующие объекты конфигурации

Прежде чем мы приступим к практическому созданию документов, необходимо сделать отступление о том, какие типы данных могут использоваться в системе «1С:Предприятие».

На предыдущем занятии, когда мы создавали реквизиты справочников или табличных частей, мы всегда указывали тип значения, которое может принимать этот реквизит. Это были *примитивные* типы данных: *Число*, *Строка*, *Дата* и *Булево*. Примитивные типы данных изначально определены в системе, и их набор ограничен.

Наряду с такими изначально определенными в любой конфигурации типами могут существовать типы данных, определяемые только конкретной конфигурацией. То есть такие типы, которые не присутствуют в конфигурации постоянно, а появляются в результате того, что добавлены некоторые объекты конфигурации.

Например, после того как мы создали объект конфигурации *Справочник Склады*, сразу же появилось несколько новых типов данных, связанных с этим

справочником. Среди них, например, *СправочникСсылка.Склады*. И если теперь мы укажем какому-либо реквизиту этот тип данных, то сможем хранить в нем ссылку на конкретный объект справочника *Склады*.

Объекты конфигурации, которые могут образовывать новые типы данных, называются *типообразующими*.

Например, после создания нового справочника *Номенклатура* становятся доступны следующие типы данных:

- *СправочникМенеджер.Номенклатура*,
- *СправочникСсылка.Номенклатура*,
- *СправочникОбъект.Номенклатура*,
- *СправочникВыборка.Номенклатура*.

Следует еще раз отметить, что эти типы данных не поддерживаются платформой изначально и существуют только в конкретном прикладном решении.

Это небольшое отступление было необходимо потому, что уже при создании первого документа мы столкнемся с использованием типов данных *СправочникСсылка.Склады* и *СправочникСсылка.Номенклатура,* которые

появились в нашей конфигурации в результате создания объектов конфигурации *Справочник Склады* и *Номенклатура*.

Документ «Приходная накладная»

После того как мы познакомились с объектом конфигурации *Документ*, создадим несколько таких объектов, чтобы иметь возможность фиксировать события, происходящие в нашем ООО «На все руки мастер».

Одними из самых популярных услуг нашего предприятия является ремонт телевизоров и установка стиральных машин. И в том, и в другом случае требуются некоторые материалы, которые расходуются в процессе оказания этих услуг. Поэтому двумя важнейшими событиями в хозяйственной жизни нашей организации будут являться поступление материалов и оказание услуг.

Для отражения этих событий в базе данных мы создадим два документа: *Приходная накладная* и *Оказание услуги*.

Документ *Приходная накладная* будет фиксировать факт поступления в нашу организацию необходимых материалов, а документ *Оказание услуги* – фиксировать оказание услуг и расход материалов, которые используются при оказании этих услуг.

В режиме «Конфигуратор»

Добавление документа

Откроем конфигуратор и добавим новый объект конфигурации *Документ*. На закладке *Основные* зададим имя документа – *ПриходнаяНакладная*. На основании имени платформа автоматически заполнит его синоним.

Здесь же определим, как будет представлен документ в интерфейсе «1С:Предприятия». *Представление объекта* задавать не будем, вместо него будет использоваться *Синоним* объекта. Это нам вполне подходит, так как мы задали его имя в единственном числе.

Представление списка, наоборот, зададим во множественном числе как *Приходные накладные* (рис. 4.4).

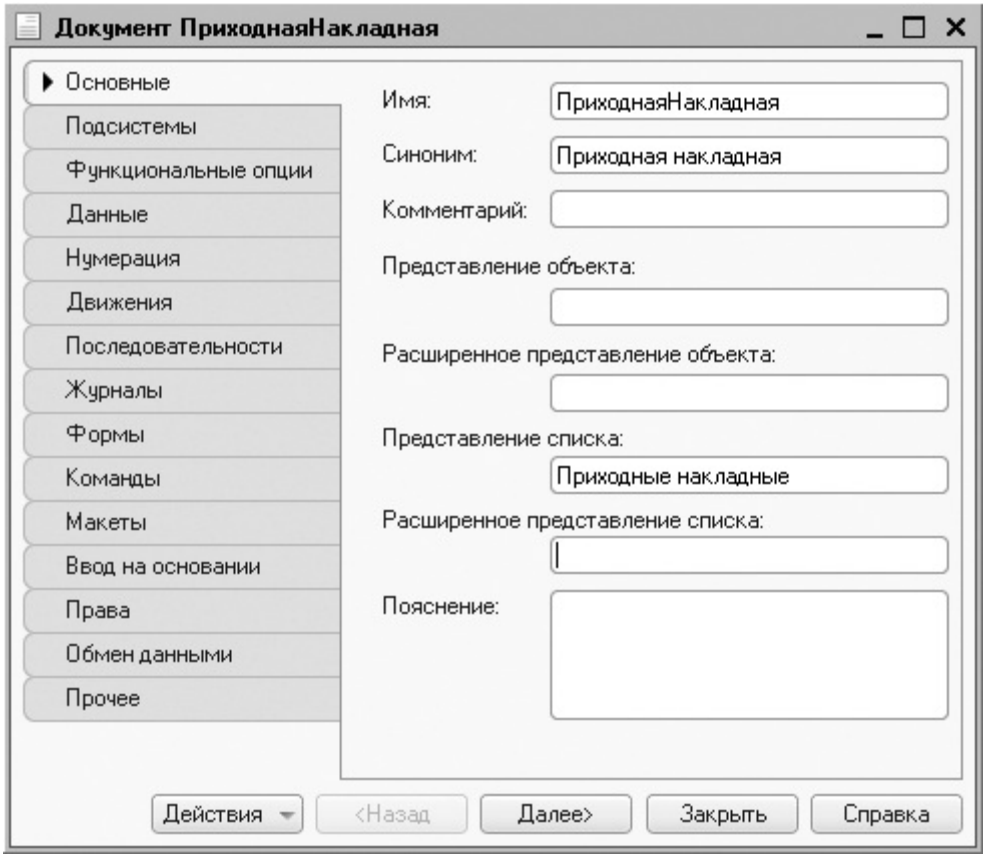


Рис. 4.4. Установка основных свойств документа

Нажмем *Далее* и перейдем на закладку *Подсистемы*.

По логике нашей конфигурации список приходных накладных должен быть

доступен в разделах *Учет материалов* и *Бухгалтерия*. Действительно, к первому разделу этот документ имеет прямое отношение, а для бухгалтерского анализа всегда может понадобиться список документов, отражающих поступление материалов.

Поэтому отметим в списке подсистем эти подсистемы (рис. 4.5).

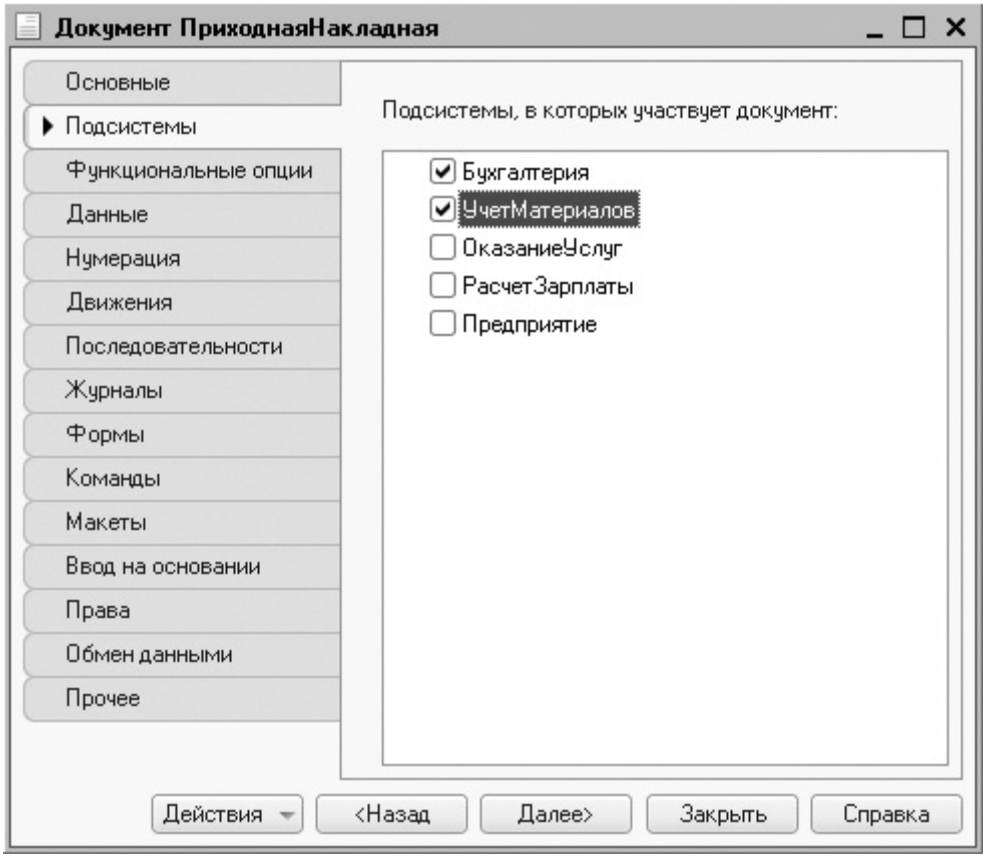


Рис. 4.5. Определение списка подсистем, в которых будет отражаться документ

Перейдем на закладку *Данные* и создадим реквизит документа с именем *Склад.* Для этого нажмем кнопку *Добавить* над списком реквизитов документа (рис. 4.6).

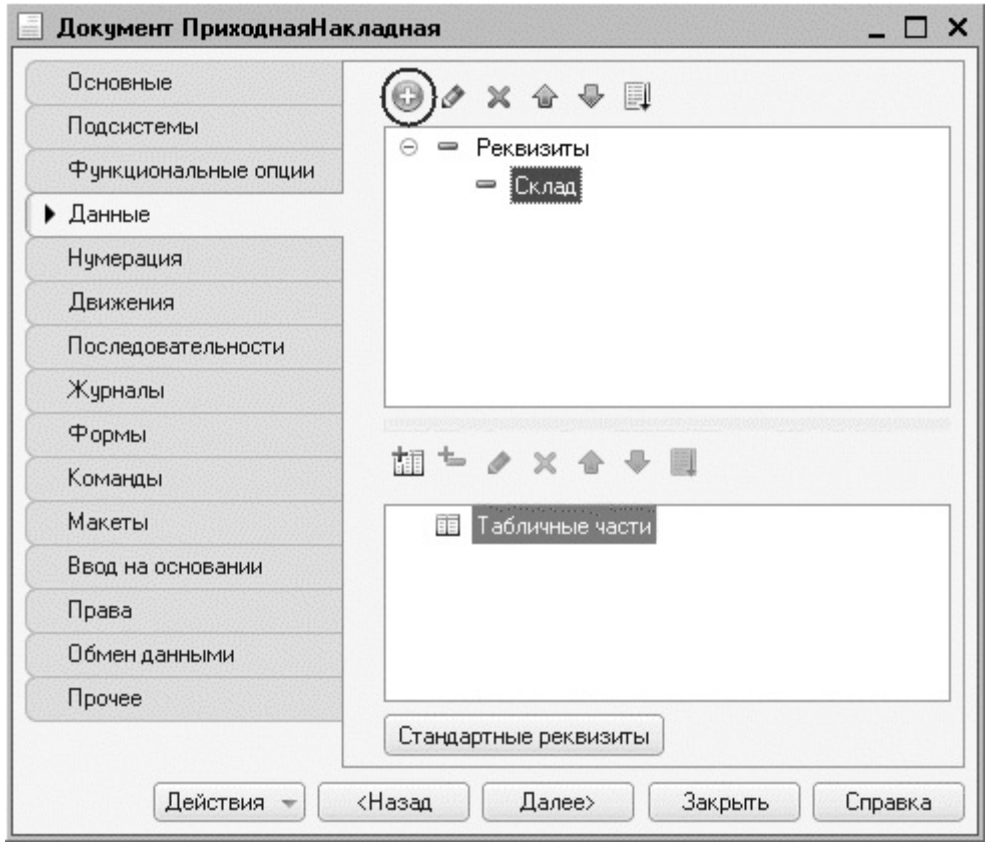


Рис. 4.6. Создание реквизита документа

Реквизиты ссылочного типа

Выберем для реквизита ссылочный тип данных *СправочникСсылка.Склады*. Этот тип стал доступен в конфигурации после создания объекта конфигурации *Справочник Склады* (рис. 4.7).

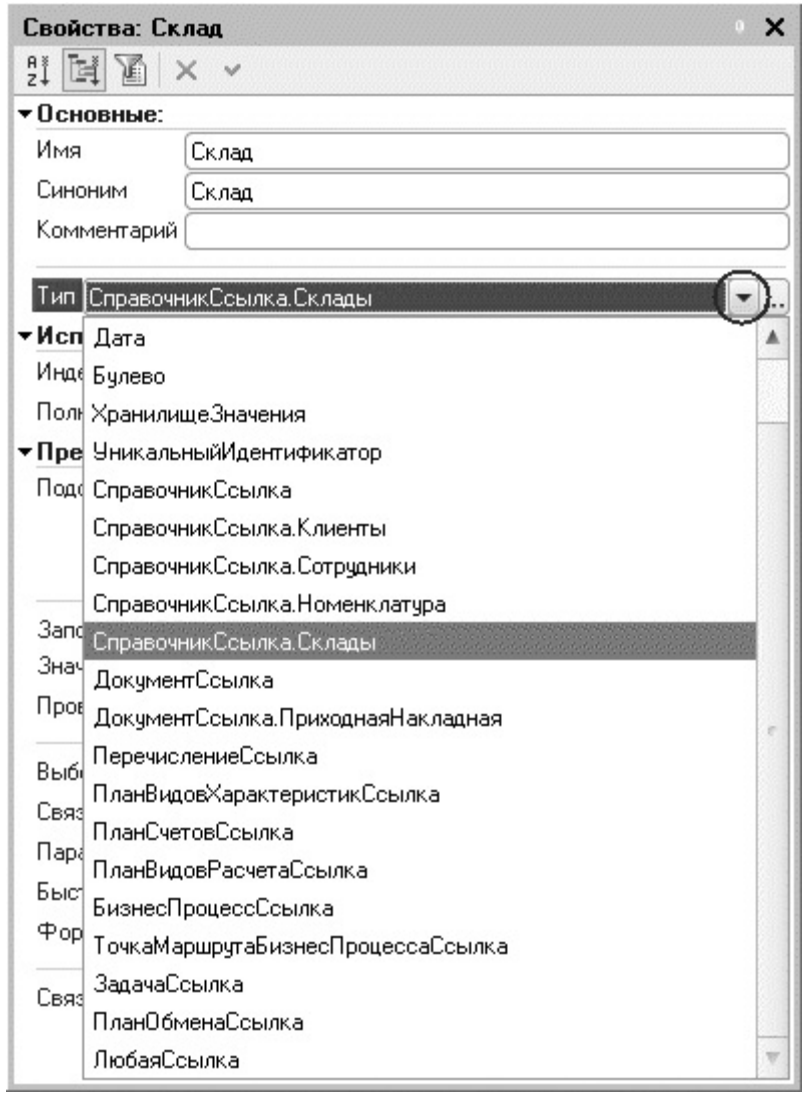

Свойства: Склад
Основные:
Имя
Склад
Синоним
Склад
Комментарий
Тип
СправочникСсылка.Склады
Дата
Булево
ХранилищеЗначения
УникальныйИдентификатор
СправочникСсылка
СправочникСсылка.Клиенты
СправочникСсылка.Сотрудники
СправочникСсылка.Номенклатура
СправочникСсылка.Склады
ДокументСсылка
ДокументСсылка.ПриходнаяНакладная
ПеречислениеСсылка
ПланВидовХарактеристикСсылка
ПланСчетовСсылка
ПланВидовРасчетаСсылка
БизнесПроцессСсылка
ТочкаМаршрутаБизнесПроцессаСсылка
ЗадачаСсылка
ПланОбменаСсылка
ЛюбаяСсылка

Рис. 4.7. Создание реквизита документа

Свойство «Значение заполнения» реквизита объекта конфигурации

Теперь покажем, как можно облегчить жизнь пользователя при приходовании материалов. Работа в автоматизируемой нами фирме построена таким образом, что, как правило, все поступающие товары приходуются на основной склад.

Поэтому в палитре свойств для созданного нами реквизита *Склад* документа найдем свойство *Значение заполнения.*

В качестве значения этого свойства выберем предопределенный элемент справочника *Склады – Основной*.

Таким образом, при создании нового документа склад будет сразу заполняться значением *Основной*, и пользователю не придется делать это вручную (рис. 4.8).

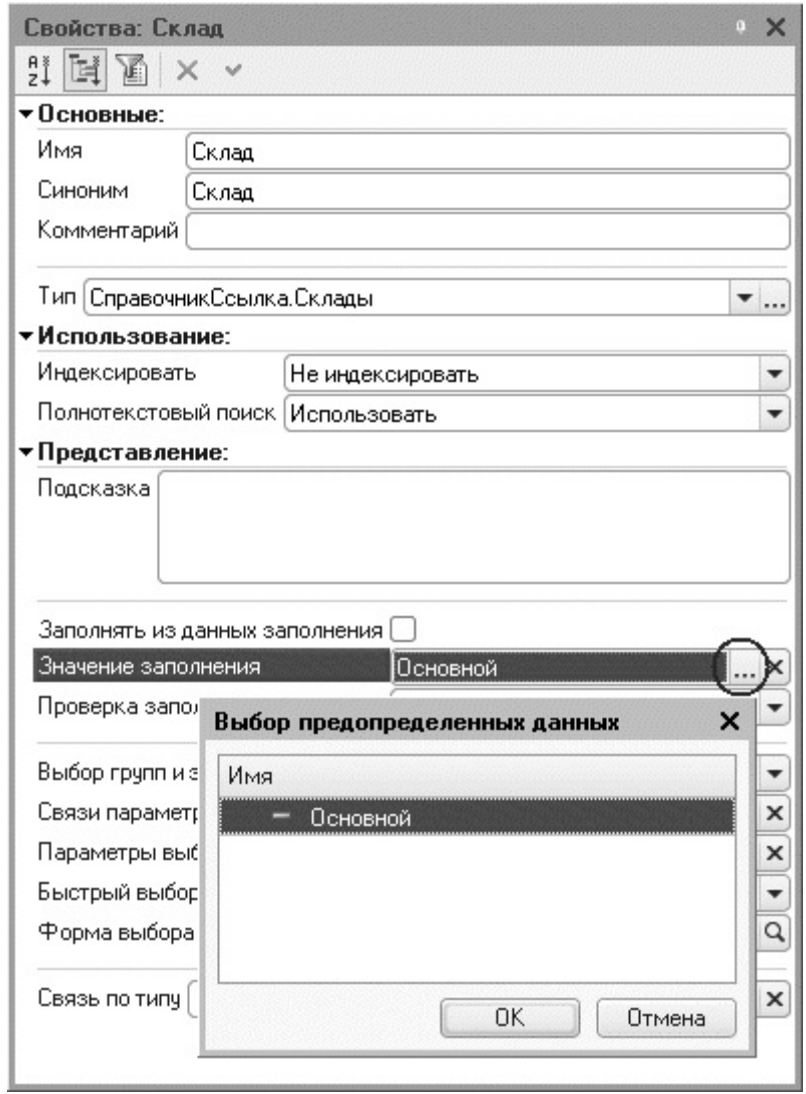

Свойства: Склад
Основные:
Имя
Склад
Синоним
Склад
Комментарий
Тип
СправочникСсылка.Склады
Использование:
Индексировать
Не индексировать
Полнотекстовый поиск
Использовать
Представление:
Подсказка
Заполнять из данных заполнения
Значение заполнения
Основной
Проверка запол
Выбор групп и э
Связи параметр
Параметры выб
Быстрый выбор
Форма выбора
Связь по типу
Выбор предопределенных данных
Имя
Основной
OK
Отмена

Рис. 4.8. Выбор значения заполнения по умолчанию для реквизита «Склад»

После этого добавим в документ табличную часть с именем *Материалы*. Для этого нажмем кнопку *Добавить табличную часть* над списком табличных частей документа (рис. 4.9).

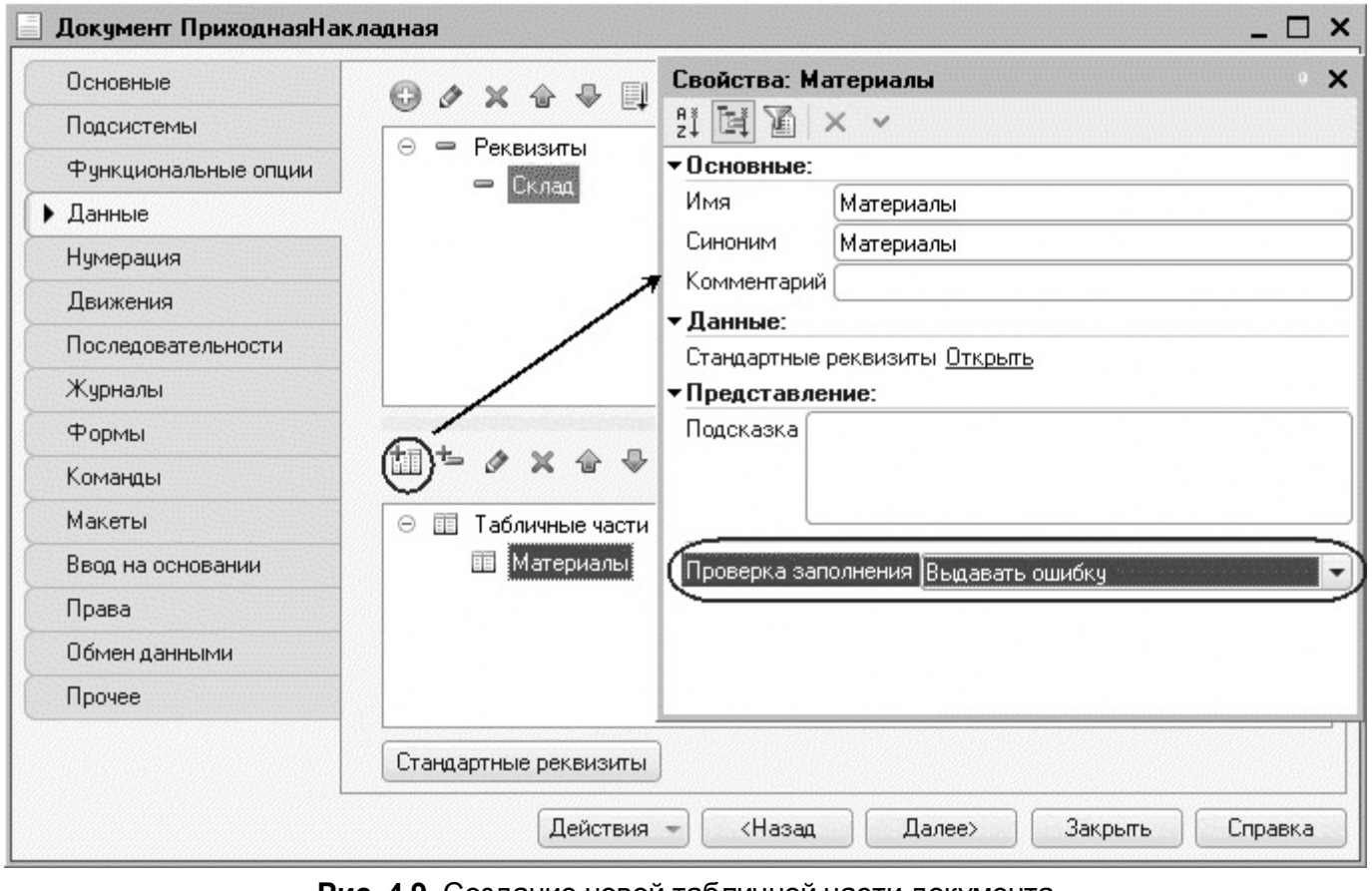


Рис. 4.9. Создание новой табличной части документа

Проверка заполнения табличной части

Кроме имени табличной части установим свойство *Проверка заполнения* в значение *Выдавать ошибку*. Тем самым мы задаем условие, что документ

Приходная накладная обязательно должен содержать табличную часть, то есть список приходуемых материалов. Иначе будет выдано сообщение об ошибке, и документ не будет сохранен.

Создадим реквизиты табличной части *Материалы.* Для этого нажмем кнопку *Добавить реквизит* в разделе описания табличных частей документа (рис. 4.10):

- *Материал*, тип *СправочникСсылка.Номенклатура*;
- *Количество*, тип *Число*, длина *15*, точность *3*, неотрицательное;
- *Цена*, тип *Число*, длина *15*, точность *2*, неотрицательное;
- *Сумма*, тип *Число*, длина *15*, точность *2*, неотрицательное.

Для каждого реквизита табличной части также установим свойство *Проверка заполнения* в значение *Выдавать ошибку*. Тем самым при записи документа будет проверяться на заполнение не только табличная часть в целом, но и ее отдельные реквизиты.

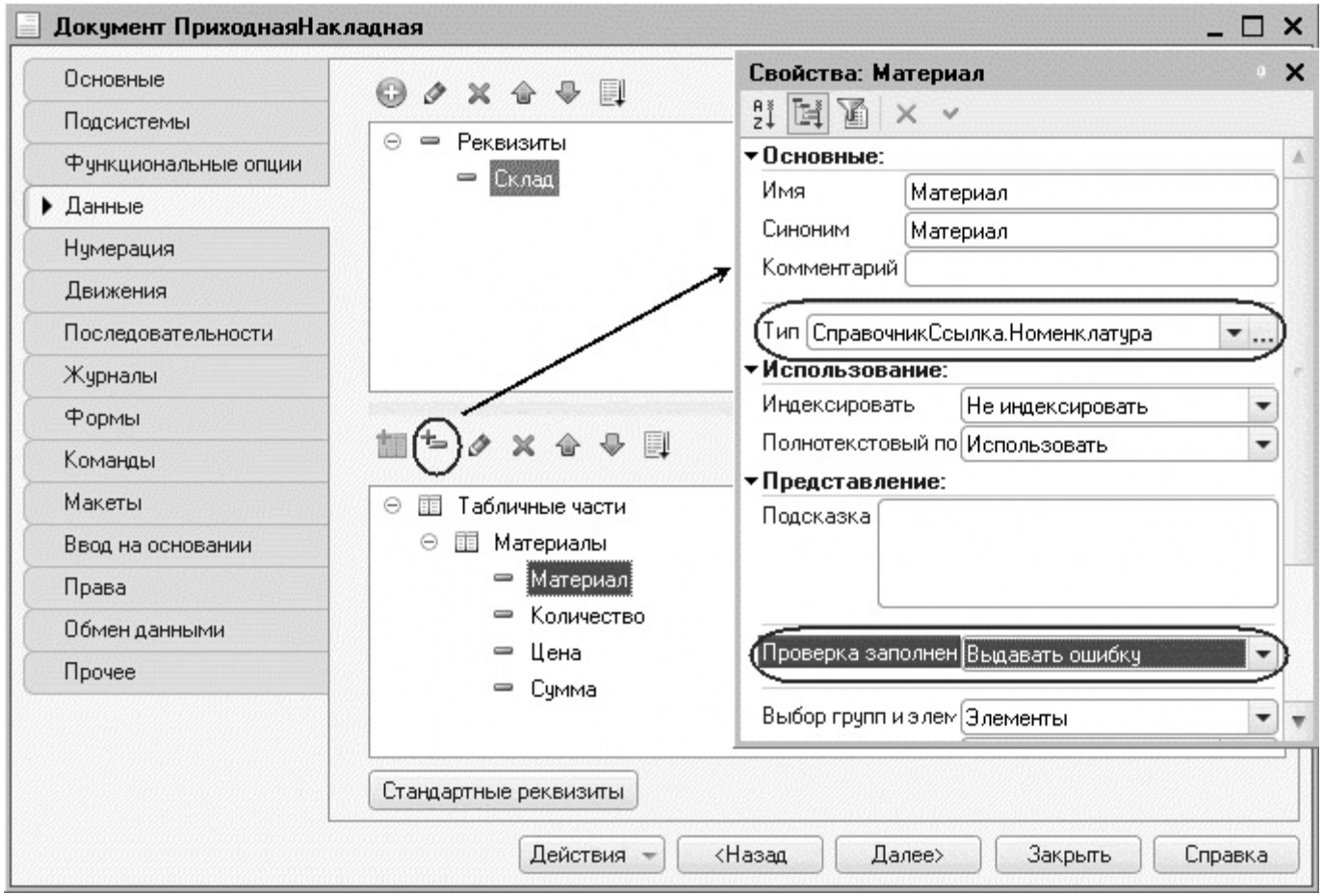


Рис. 4.10. Создание реквизитов табличной части документа

Перейдем на закладку *Нумерация* и убедимся, что свойство *Автонумерация* включено. Это обеспечит автоматическую генерацию уникальных номеров для

создаваемых нами документов.

В заключение отредактируем командный интерфейс, чтобы в подсистеме *Учет материалов* была доступна команда создания новых документов.

Для этого в дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*. В открывшемся окне слева в списке *Подсистемы* выделим подсистему *УчетМатериалов*.

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель действий.Создать* включим видимость у команды *Приходная накладная: создать* (рис. 4.11).

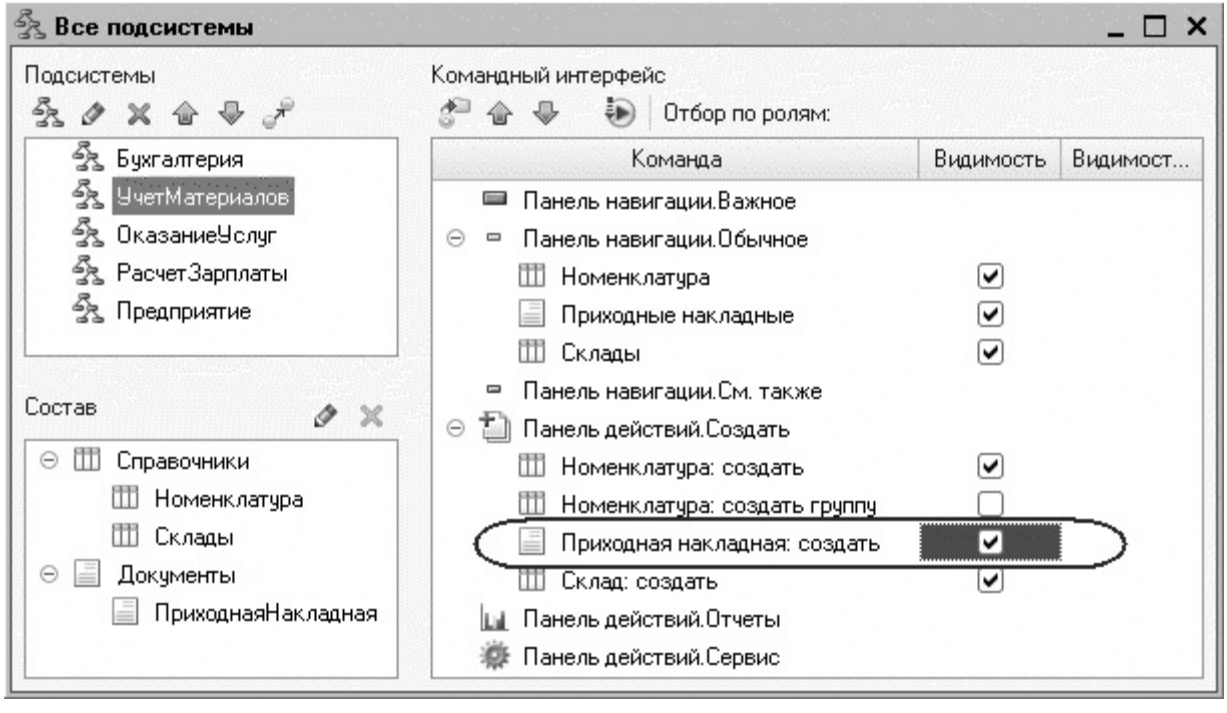


Рис. 4.11. Настройка командного интерфейса

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и протестируем получившийся документ.

В открывшемся окне «1С:Предприятия» мы видим, что в панели навигации разделов *Бухгалтерия* и *Учет материалов* появилась команда *Приходные*

накладные для открытия списка приходных накладных (рис. 4.12).

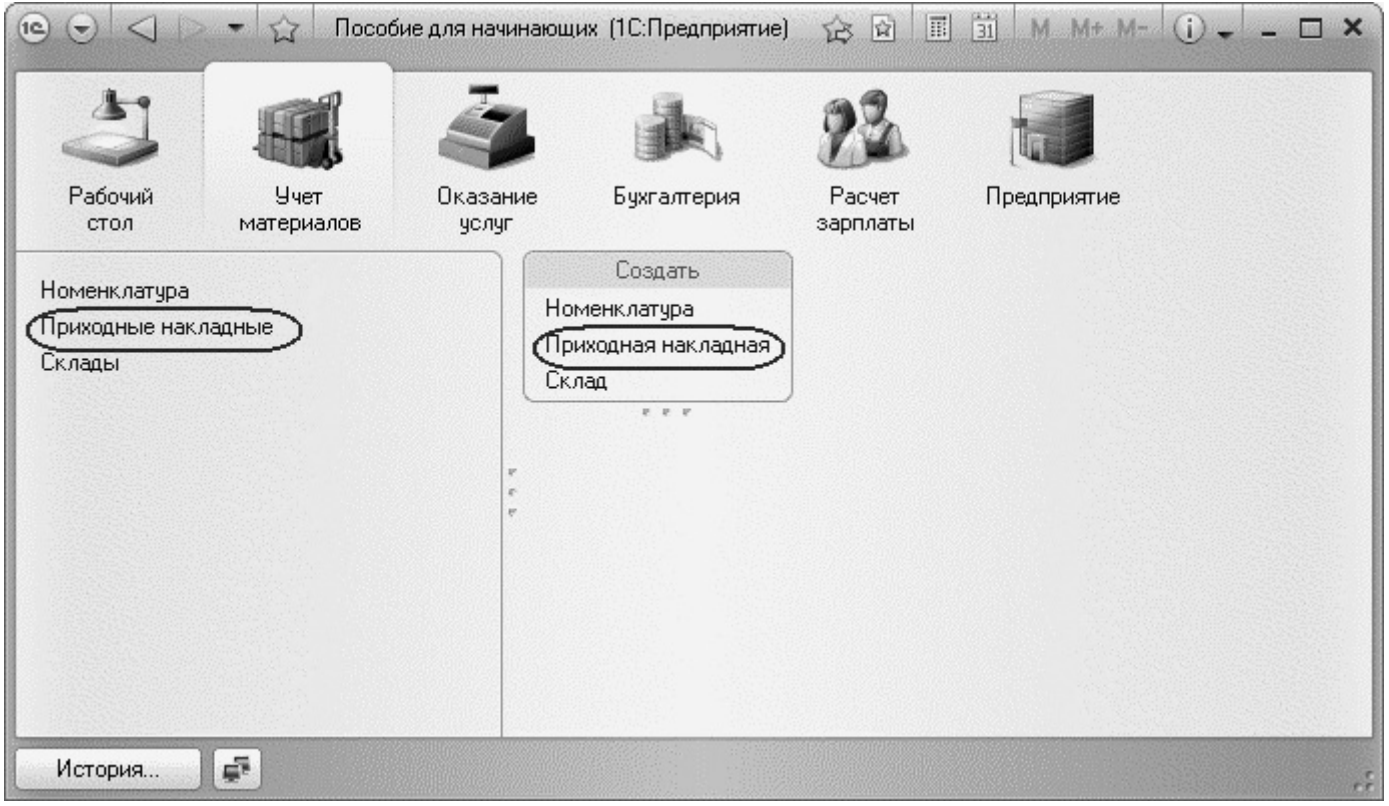


Рис. 4.12. Раздел «Учет материалов»

Название этой команды определяется свойством *Представление списка*, которое мы задали для этого документа.

Также в панели действий раздела *Учет материалов* появилась команда *Приходная накладная* для создания новых документов этого вида. Название этой команды определяется синонимом, так как *Представление объекта* мы для этого документа не задавали (см. рис. 4.12).

Добавление приходных накладных

Пока в нашей базе данных нет ни одного документа *Приходная накладная*, поэтому выполним команду *Приходная накладная* в панели действий раздела *Учет материалов* и создадим новую приходную накладную.

Перед нами откроется форма документа – основная форма объекта. Заголовок этой формы *Приходная накладная* совпадает с синонимом документа.

Система автоматически подставит текущую дату создания документа и нулевое время, так как документ еще не проведен. В качестве времени документа при оперативном проведении ему присваивается оперативная отметка времени.

Поле *Номер* не заполнено, но система сама сгенерирует для нового документа уникальный номер, так как свойство *Автонумерация* для документа включено по умолчанию. Новый номер будет сохранен в момент записи документа.

Обратите внимание, что склад уже заполнен значением *Основной*, как мы и задали в свойствах этого реквизита.

Нам осталось только заполнить табличную часть приходной накладной материалами для ремонта телевизоров так, как показано на рисунке (рис. 4.13).

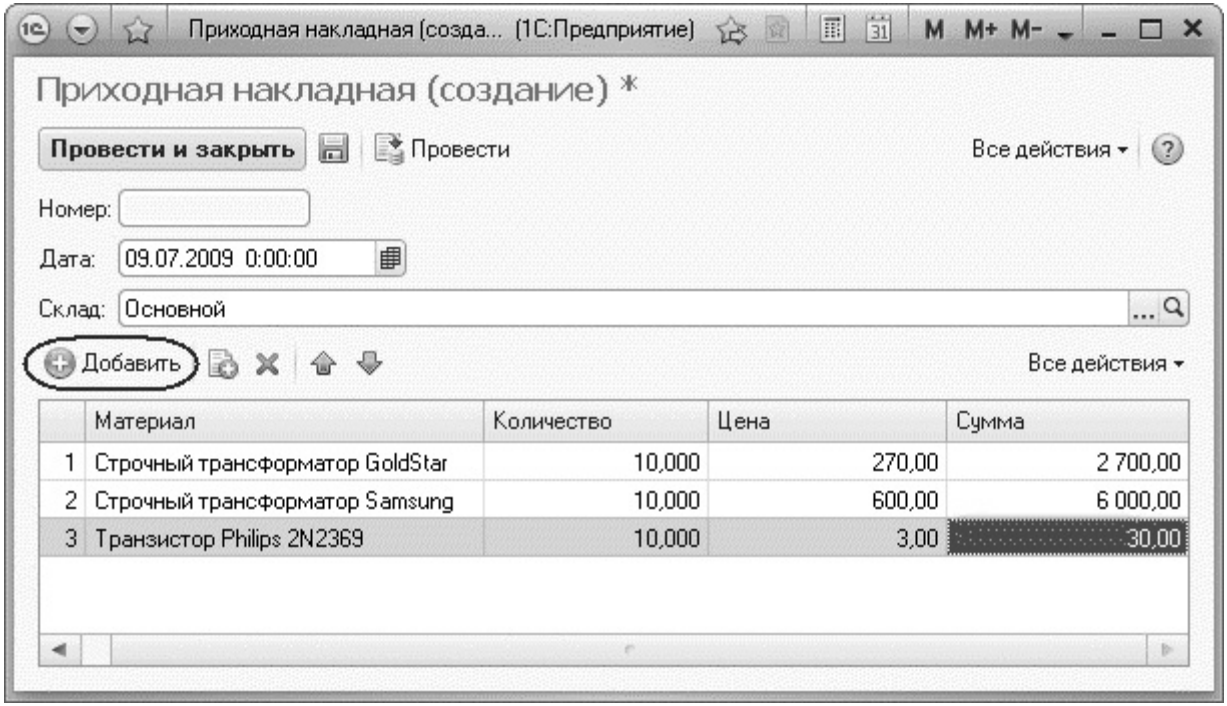


Рис. 4.13. Создание нового документа «Приходная накладная № 1»

Обратите внимание, что при нажатии кнопки выбора в поле *Материал* (в табличной части документа) открывается форма для выбора элементов справочника *Номенклатура*, так как этот реквизит имеет ссылочный тип данных

и ссылается на справочник *Номенклатура* (см. рис. 4.13).

Нажмем *Провести и закрыть*.

Документ будет сохранен и проведен, ему будет присвоен автоматически сгенерированный системой номер и текущее время проведения документа.

Аналогичным образом мы создадим второй документ, который будет приходовать на *Основной* склад материалы для установки стиральных машин. Но теперь не будем использовать кнопку выбора в поле *Материал*, а просто начнем вводить название материала в это поле. Платформа автоматически найдет материалы, наименование которых начинается с введенных нами символов, и предложит их нам для выбора.

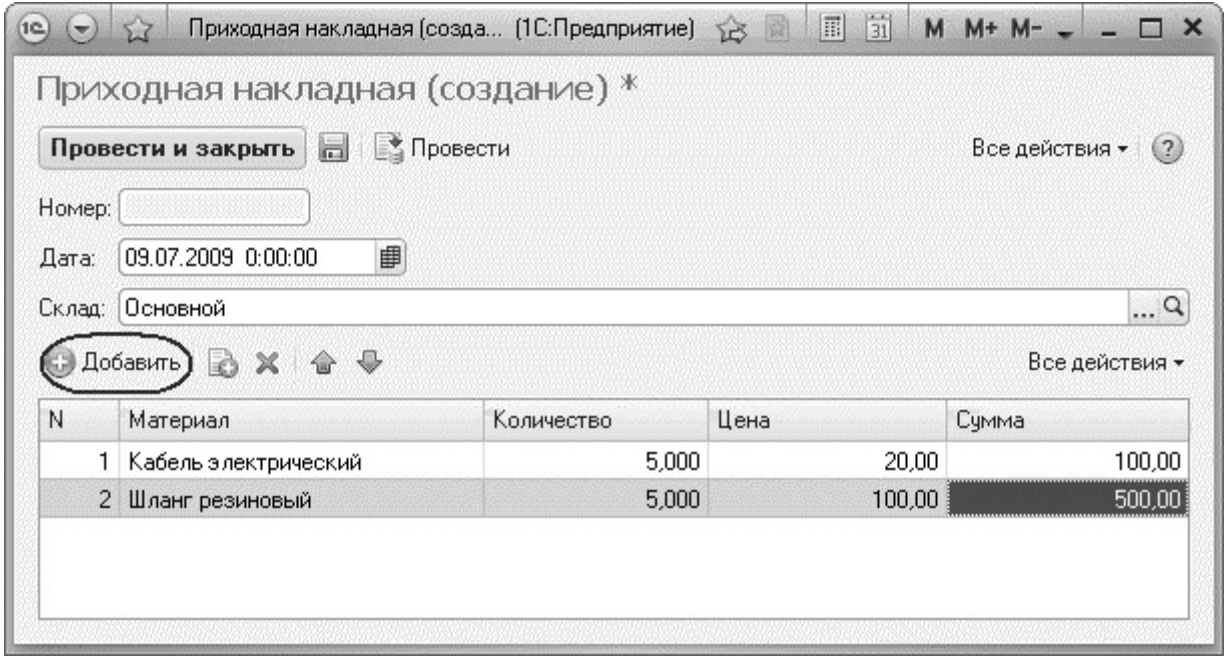


Рис. 4.14. Создание нового документа «Приходная накладная № 2»

Нажмем *Провести и закрыть*.

Документ будет сохранен и проведен, ему будет присвоен автоматически сгенерированный системой номер и текущее время проведения документа.

Обратите внимание, что при вводе нового документа табличная часть в целом и каждая ее колонка подсвечена красным пунктиром. Это значит, что для них

выполняется проверка заполнения. Если не ввести ни одной строки в табличную часть документа или оставить незаполненной какую-либо колонку табличной части и попытаться записать документ, то будет получено сообщение об ошибке (рис. 4.15).

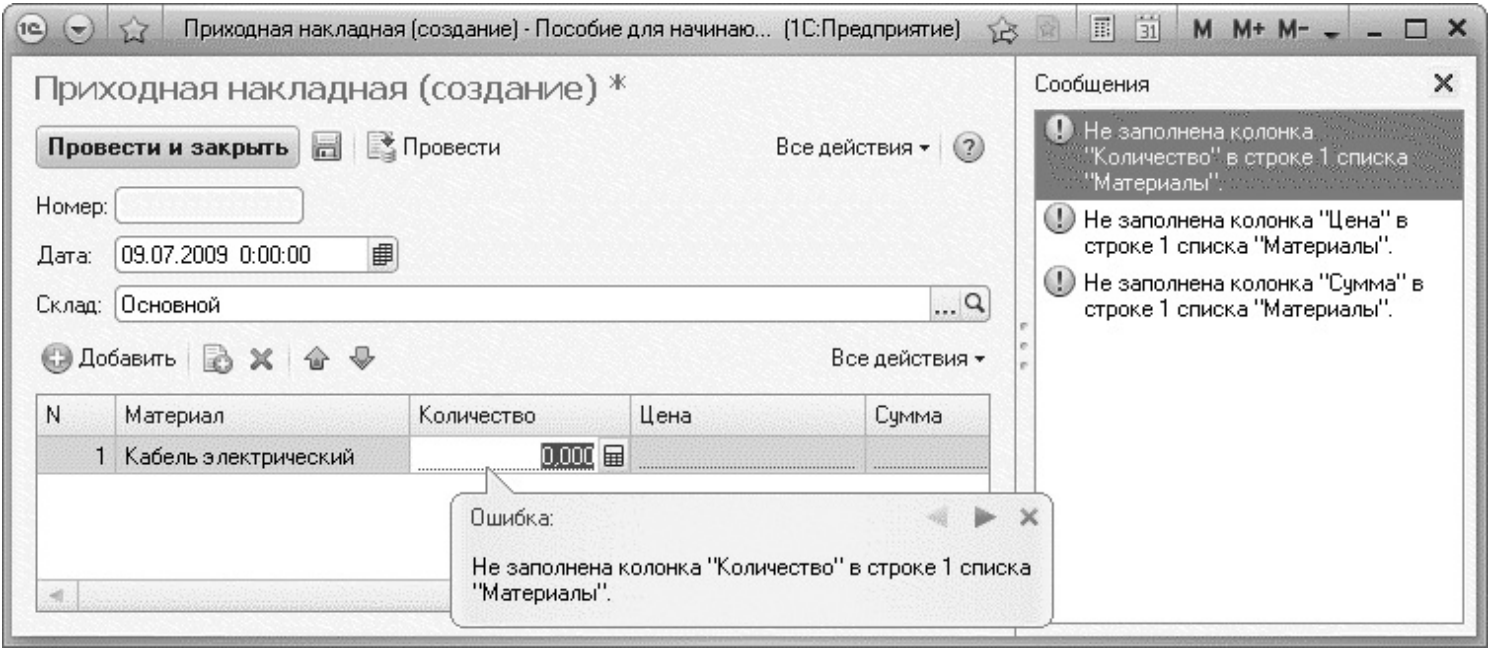


Рис. 4.15. Сообщение об ошибке при вводе нового элемента документа

Чтобы просмотреть список созданных документов, выполним команду *Приходные накладные* в панели навигации.

В форме списка, открывшейся в рабочей области окна приложения, мы видим два созданных нами документа, отмеченных пиктограммой, указывающей на то, что документы проведены (зеленая галочка в пиктограмме документа, рис. 4.16).

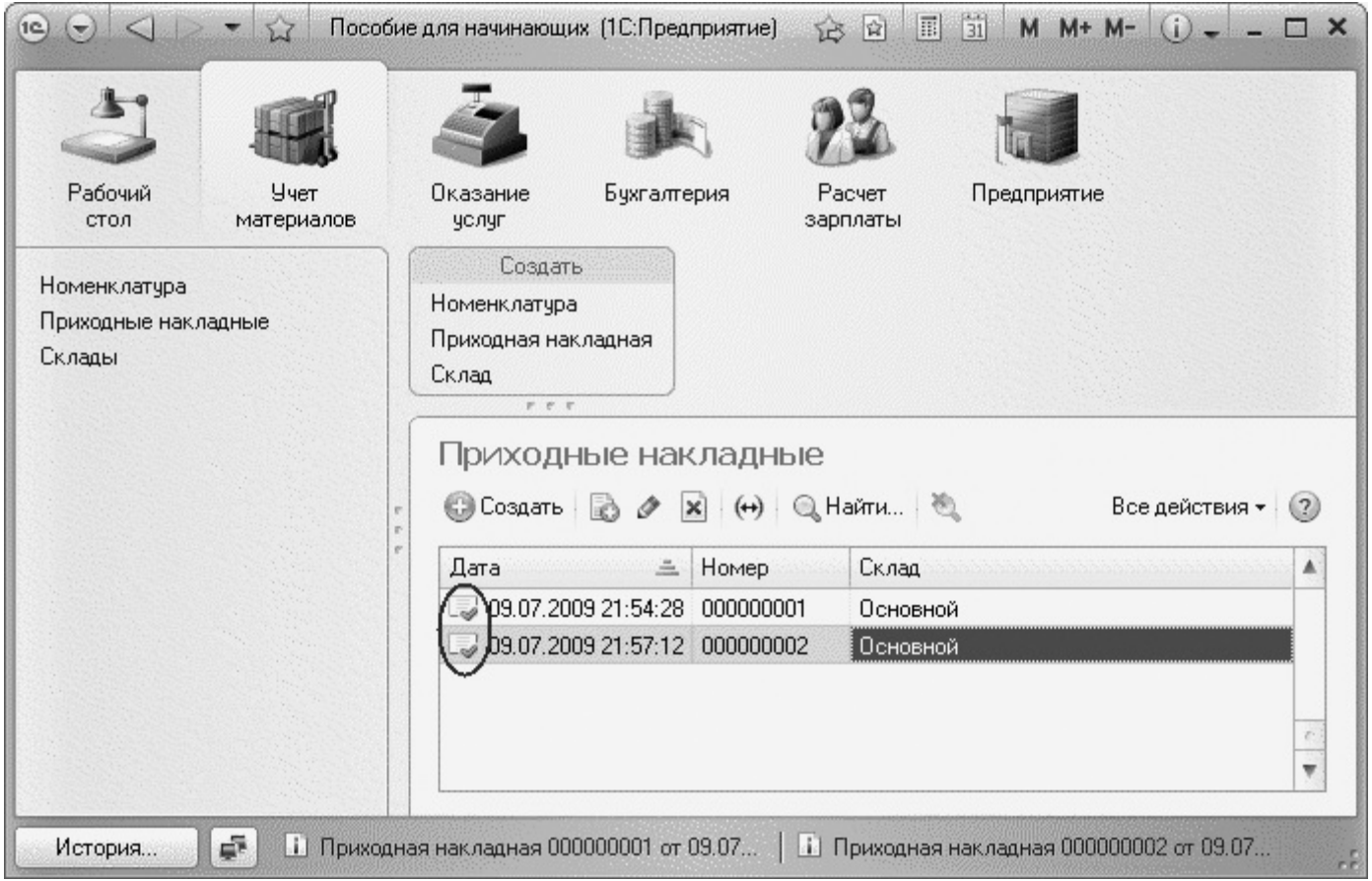


Рис. 4.16. Список приходных накладных

«Теория». Справочники и документы

Интересно обратить внимание на разницу в употреблении единственного и

множественного числа при именовании объектов вида *Справочник* и *Документ*.

Если вы откроете дерево типовой конфигурации, то увидите, что все объекты, расположенные в ветке *Справочники*, как правило, именованы во множественном числе. В ветке же *Документы*, как правило, в единственном числе.

Может сложиться впечатление, что, создавая объект конфигурации Справочник, мы делаем хранилище для элементов этого справочника, а создавая объект конфигурации *Документ*, – некий шаблон одного конкретного документа. На самом деле это не так.

Создавая в конфигураторе объект вида *Справочник*, мы даем ему наименование во множественном числе (*Товары*). При этом подразумевается, что в базе данных этот объект будет состоять из элементов, описывающих конкретные товары (в единственном числе).

Создавая в конфигураторе объект вида *Документ*, мы даем ему наименование в единственном числе (*ПриходнаяНакладная*), однако на самом деле мы создаем такое же хранилище, как и в случае со справочником. Каждая запись этого хранилища будет описывать один документ, одну приходную накладную (в единственном числе). Поэтому концептуально правильно было бы в конфигураторе задавать наименование объекта вида *Документ* во

множественном числе, подчеркивая тем самым описание набора документов этого вида (например, *ПриходныеНакладные*).

Однако психология человека такова, что, открывая ветку *Документы*, он ожидает увидеть перечисление их в единственном числе, а никак не во множественном. Так происходит потому, что в реальной жизни трудно найти подходящий термин для описания совокупности документов одного вида (совокупность записей одного вида обозначить гораздо проще – справочник, план и т. д.). Поэтому соответствующая ветка объектов конфигурации имеет название *Документы*, а объекты конфигурации, создаваемые в этой ветке, именуются в единственном числе, хотя, по сути, сама ветка содержит описания хранилищ документов разных видов, а каждый элемент в этой ветке описывает набор всех документов одного вида.

Автоматический пересчет суммы в строках документа

Наверняка вы обратили внимание на то, что при заполнении документа приходится вводить сумму в каждой строке. Это неудобно, и возникает естественное желание автоматизировать работу документа так, чтобы сумма вычислялась автоматически каждый раз при изменении цены или количества материалов в строке.

Это совсем не сложно, и для этого нам потребуется сначала создать

собственную форму документа, а затем воспользоваться возможностями встроенного языка.

Дело в том, что до сих пор мы использовали предопределенные формы объектов, которые система «1С:Предприятие» по умолчанию создавала для нас сама. Теперь же у нас возникла необходимость слегка изменить логику работы формы документа, поэтому нам нужно создать свою собственную форму документа *ПриходнаяНакладная* для того, чтобы в ней с помощью встроенного языка описать тот алгоритм, который нам нужен. И система будет использовать нашу форму вместо формы по умолчанию.

В режиме «Конфигуратор»

Форма документа

Вернемся в конфигуратор и откроем окно редактирования объекта конфигурации *Документ ПриходнаяНакладная*.

В этом окне нас интересует закладка *Формы*.

Как мы видим, ни одна из основных форм документа пока не задана. Для того чтобы создать форму документа, нажмем кнопку открытия  со значком лупы в поле ввода или кнопку *Добавить* над списком форм (рис. 4.17).

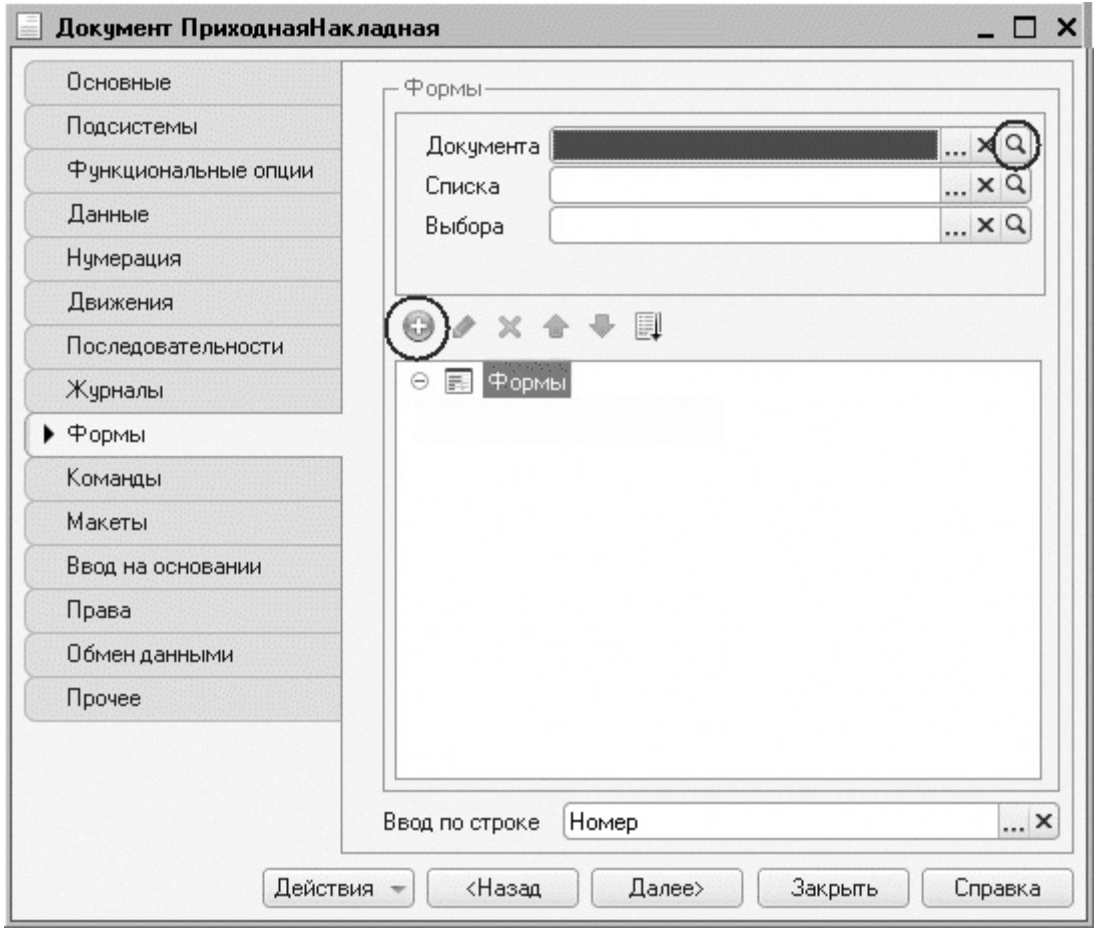


Рис. 4.17. Создание формы документа

Система вызовет еще один полезный инструмент разработчика – *конструктор форм* (рис. 4.18).

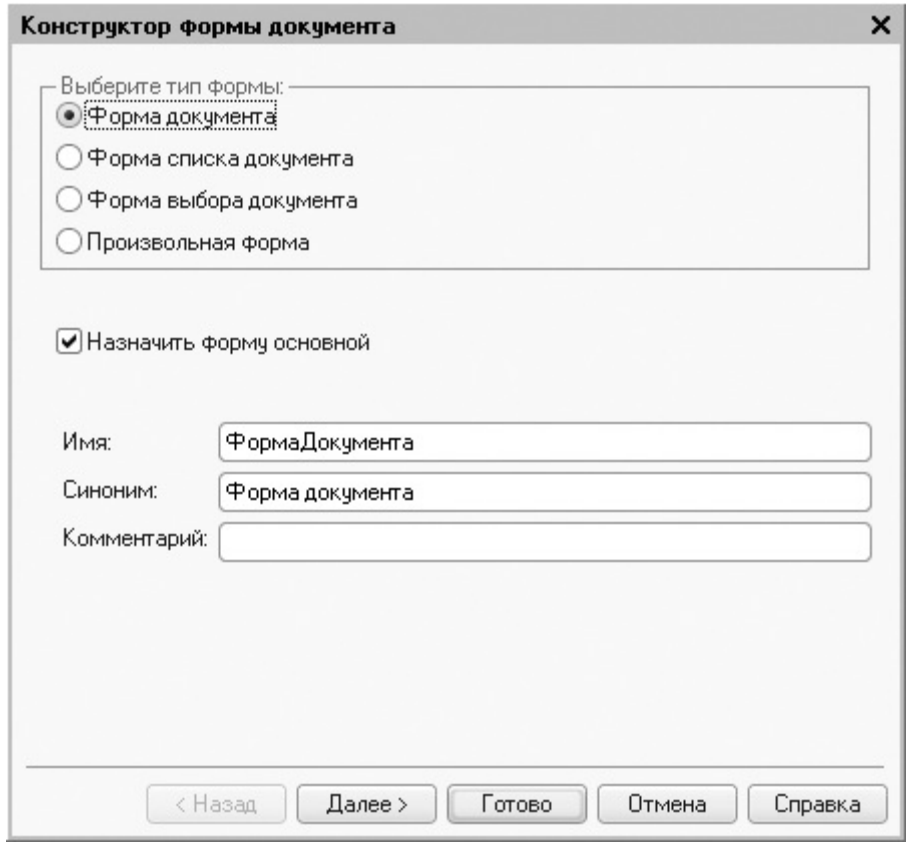


Рис. 4.18. Конструктор форм

Этот инструмент также построен по принципу «мастеров»: ввод данных в определенной последовательности и передвижение кнопками *Далее* и *Назад*.

Выберем тип формы *Форма документа* и нажмем кнопку *Готово*,

согласившись тем самым со всем, что нам предложила система (см. рис. 4.18).

Обратите внимание, что в дереве объектов конфигурации у объекта конфигурации *Документ ПриходнаяНакладная* появилась форма *ФормаДокумента* (рис. 4.19), а на экране открылось окно редактора форм, содержащее эту форму (рис. 4.20).

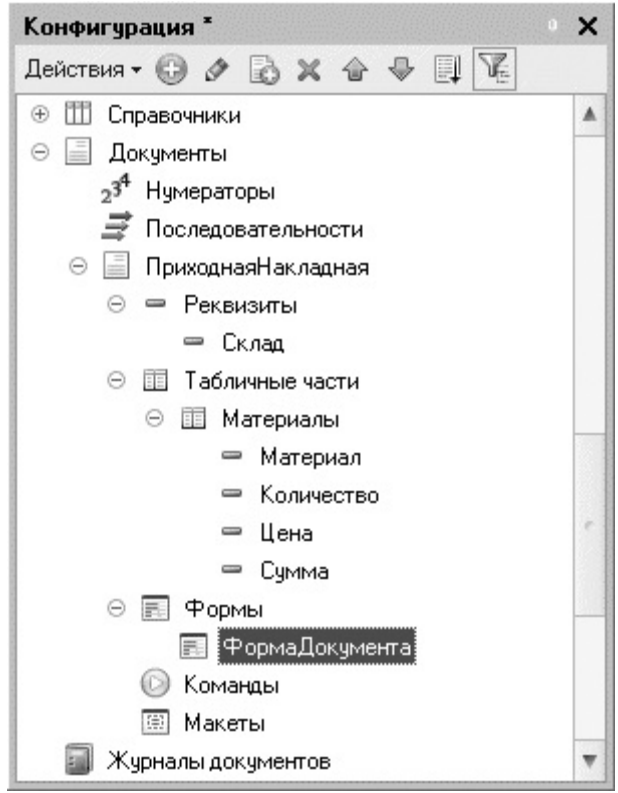


Рис. 4.19. Новая форма документа в конфигураторе

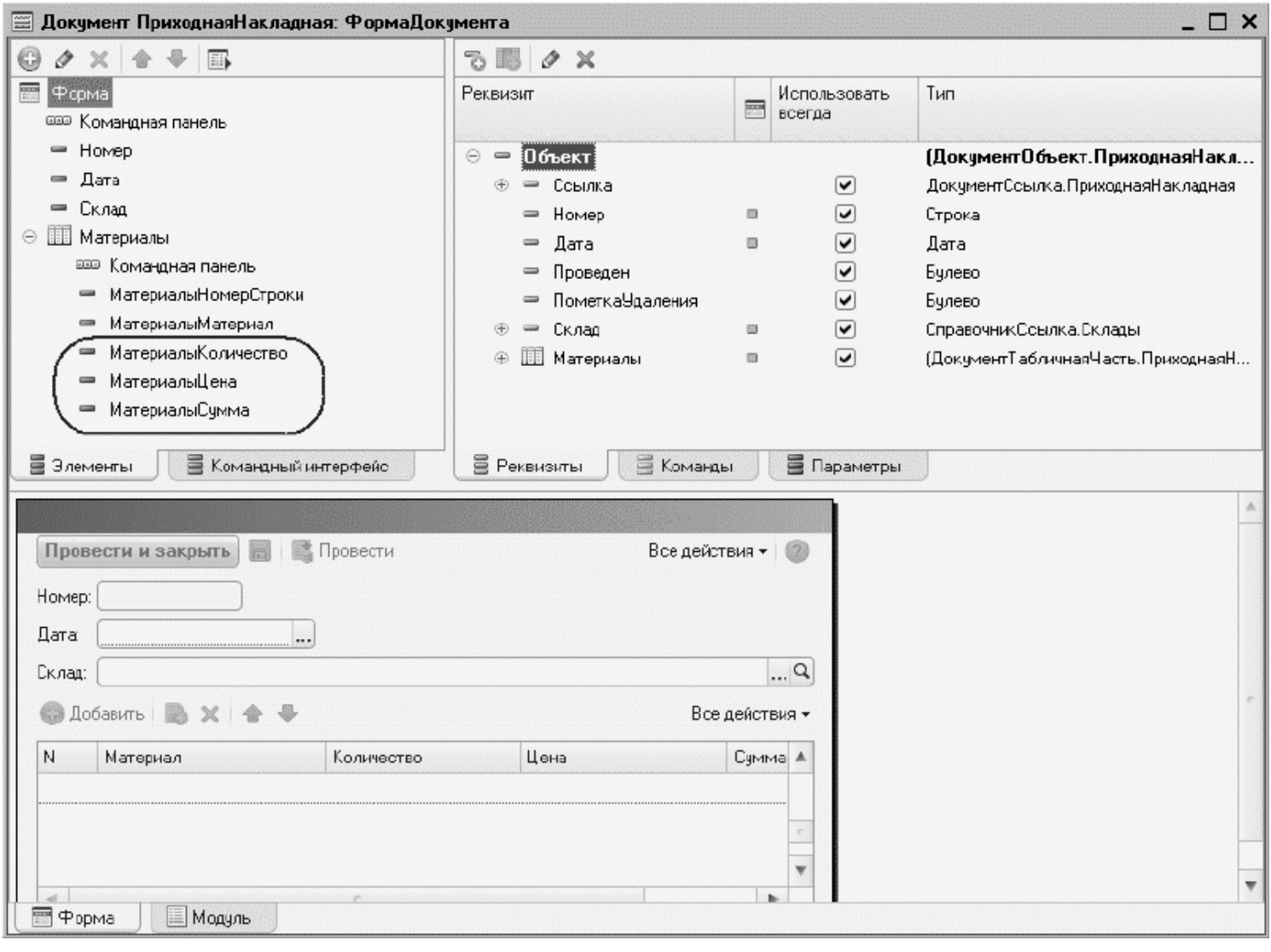

Документ ПриходнаяНакладная: ФормаДокумента
Форма
Командная панель
Номер
Дата
Склад
Материалы
Командная панель
МатериалыНомерСтроки
МатериалыМатериал
МатериалыКоличество
МатериалыЦена
МатериалыСумма
Элементы
Командный интерфейс
Реквизит
Использовать всегда
Тип
Объект
(ДокументОбъект.ПриходнаяНакл...
Ссылка
ДокументСсылка.ПриходнаяНакладная
Номер
Строка
Дата
Дата
Проведен
Булево
ПометкаУдаления
Булево
Склад
СправочникСсылка.Склады
Материалы
(ДокументТабличнаяЧасть.ПриходнаяН...
Реквизиты
Команды
Параметры
Провести и закрыть
Провести
Все действия
Номер:
Дата
Склад:
Добавить
Все действия
N
Материал
Количество
Цена
Сумма
Форма
Модуль

Рис. 4.20. Окно редактора форм

Редактор форм объединяет несколько окон взаимосвязанных между собой редакторов. Мы не будем здесь разбирать подробно работу с редактором форм, а коснемся только тех моментов, которые нужны для выполнения простейших действий, связанных с нашей задачей.

За более подробной информацией о работе с редактором форм следует обратиться к документации: «1С:Предприятие 8.2. Руководство разработчика», глава 21.1 «Инструменты разработки» – «Редактор формы».

Пока мы рассмотрим только окно формы документа в режиме просмотра, расположенное внизу, и окно редактора элементов формы, расположенное слева в верхней части окна редактора форм.

Прежде всего, необходимо понимать, что при разработке форм объектов конфигурации разработчик не имеет возможности нарисовать форму. Он может только указать, из каких элементов будет состоять форма, а система уже сама самостоятельно расположит эти элементы в форме.

Элементы формы в верхнем левом окне редактора форм образуют иерархическую структуру, из которой следует, что чем выше в списке находится элемент, тем выше и левее на форме он будет располагаться.

Эта структура редактируется на закладке *Элементы* и позволяет управлять отображением и редактированием данных в форме.

Мы видим, что на основе описания в конфигурации документа *ПриходнаяНакладная* система создала структуру элементов, которая определяет, как будет выглядеть форма.

Эти элементы имеют разное назначение и разное поведение. Однако все они служат для того, чтобы отображать информацию, хранящуюся в базе данных, и организовывать интерактивную работу с этой информацией.

Вы можете попробовать перетащить мышью поля в дереве элементов и поменять местами, например, реквизиты табличной части. Результат изменений сразу отразится в форме документа в нижней части редактора форм. При этом разработчику не нужно задумываться над вопросами конкретного (до пикселя) расположения того или иного элемента формы, его размеров и привязки к другим элементам. Это работу берет на себя система.

Но разработчик может через палитру свойств изменить свойства элемента, которые повлияют на его отображение в форме. Он может также изменить структуру элементов формы – создать новое поле, группу полей, добавить табличную часть, связав эти элементы с данными формы.

Но пока нам ничего этого не нужно делать. Нас интересуют три элемента табличной части: *МатериалыКоличество*, *МатериалыЦена* и *МатериалыСумма* (см. рис. 4.20).

Мы хотим, чтобы каждый раз, когда меняется значение в поле *Количество* или в поле *Цена*, в поле *Сумма* автоматически устанавливалось значение, равное *Количество*Цена*.

Очевидно, что для этого нужно написать на встроенном языке команду, похожую на *Сумма = Количество*Цена*, которая будет выполняться при изменении значения поля *Количество* или *Цена*. Но как «поймать» эти моменты изменения?

Обработчик события

Как мы уже видели, система сама умеет работать с теми объектами, которые описаны в дереве конфигурации: показывать их данные, добавлять новые элементы и пр. То есть у нее есть некие «стандартные представления» о том, как это все должно работать.

Но, как правило, разработчиков эти «стандартные представления» устраивают только в самых простых случаях. Реальные задачи гораздо разнообразнее. Поэтому у системы существуют *события*, которые связаны с самыми различными моментами ее «стандартного» функционирования. В том числе

события, связанные с функционированием форм и элементов, расположенных в этих формах.

Используя встроенный язык, разработчик может «вклиниться» в эти события и описать собственный алгоритм того, что должно происходить при наступлении этого события. Что мы сейчас и сделаем.

Дважды щелкнем на элементе формы *МатериалыКоличество* или правой кнопкой мыши откроем для него палитру свойств (пункт контекстного меню *Свойства*).

Прокрутив список до конца, мы увидим перечень событий, которые могут быть связаны с этим полем.

Очевидно, что нам нужно событие *ПриИзменении*, которое возникает после изменения значения поля. Найдем его в списке событий и нажмем кнопку открытия  со значком лупы в поле ввода (рис. 4.21).

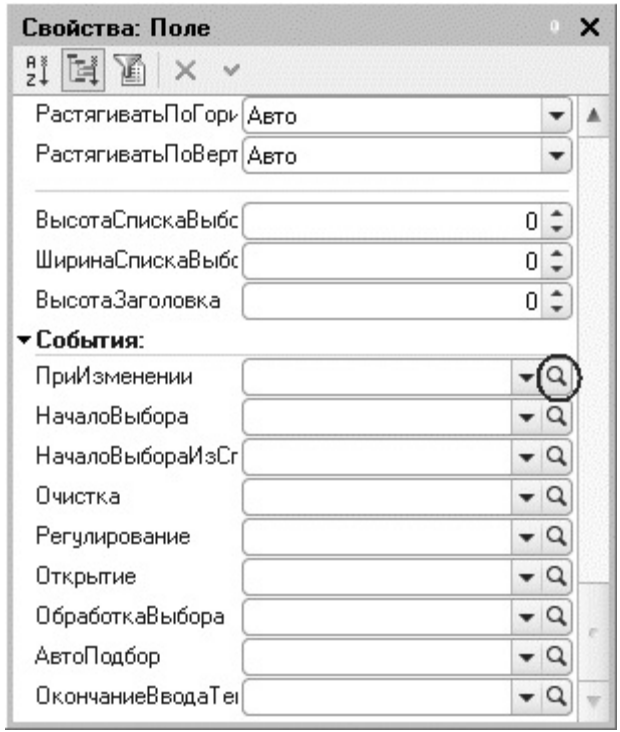


Рис. 4.21. Создание обработчика события «ПриИзменении» поля табличной части «Количество»

Система создаст шаблон процедуры *обработчика* этого события в модуле нашей формы и откроет закладку *Модуль* редактора формы (рис. 4.22).

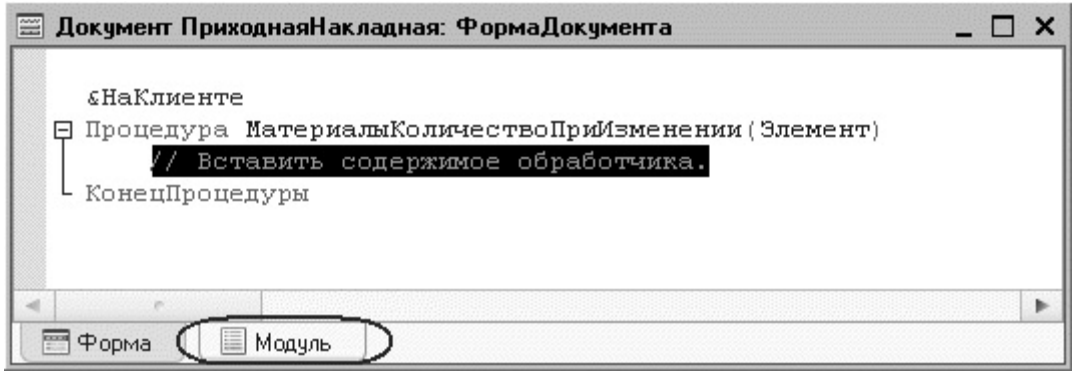


Рис. 4.22. Шаблон обработчика события «ПриИзменении» поля табличной части «Количество»

Модуль – это «хранилище» для текста программы на встроенном языке. В данном случае это модуль формы, так как обработчики всех интерактивных событий, связанных с элементами формы, помещаются именно в модуль формы.

В модуль формы, в процедуру *МатериалыКоличествоПриИзменении(),* мы и добавим следующий текст (листинг 4.1).

Листинг 4.1. Процедура «МатериалыКоличествоПриИзменении()»

```
СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество *
СтрокаТабличнойЧасти.Цена;
```

Объясним назначение этих строк.

В первой строке мы сначала создаем переменную *СтрокаТабличнойЧасти*, в которую будет помещен объект, содержащий данные, находящиеся в строке табличной части, которую нам нужно пересчитать.

Мягкая типизация данных встроенного языка позволяет сделать это, не объявляя переменную и ее тип заранее. Мы создаем переменную прямо по ходу работы, и ее тип определяется типом значения, которое она содержит.

Поскольку мы находимся в модуле формы, то в нем доступны все свойства и методы объекта встроенного языка *УправляемаяФорма*. Поэтому мы можем обращаться к ним напрямую. В данном случае после знака равенства мы обращаемся к коллекции элементов формы, используя одно из свойств объекта *УправляемаяФорма* – свойство *Элементы*.

Коллекция элементов формы является объектом встроенного языка *ВсеЭлементыФормы*, содержащим все элементы формы. То есть это программный аналог корня дерева элементов формы.

Каждый элемент формы можно получить, указав его имя в качестве свойства этого объекта, то есть через точку от него. В данном случае мы обращаемся к табличной части документа *Материалы* (*Элементы.Материалы*).

Табличная часть документа представляет собой объект встроенного языка *ТаблицаФормы*. Получить ту строку, в которой в настоящее время осуществляется редактирование, можно при помощи свойства программного объекта *ТаблицаФормы – ТекущиеДанные* (*Элементы.Материалы.ТекущиеДанные*).

Таким образом, в результате выполнения первой строки процедуры обработчика переменная *СтрокаТабличнойЧасти* будет содержать объект *ДанныеФормыСтруктура*. Этот объект содержит данные, находящиеся в текущей строке табличной части документа (*Элементы.Материалы.ТекущиеДанные*).

Получив этот объект, мы можем обратиться к данным конкретной колонки табличной части, указав имя колонки в качестве свойства объекта. Например, используя обращение *СтрокаТабличнойЧасти.Количество* мы получаем число, которое находится в редактируемой строке в колонке *Количество*.

Таким образом, во второй строке процедуры обработчика вычисляется значение колонки *Сумма* как произведение значений колонок *Количество* и *Цена*.

В режиме «1С:Предприятие»

Теперь посмотрим, как это работает. Запустим «1С:Предприятие» в режиме отладки, откроем список документов *Приходные накладные* и откроем любой из двух созданных нами документов. Если теперь вы поменяете количество в любой строке документа, то сумма в строке будет пересчитана автоматически.

Одна процедура для обработки нескольких событий

Итак, мы убедились, что при изменении количества в любой строке документа *Приходная накладная*, сумма в этой строке пересчитывается автоматически.

Замечательно. Но теперь хотелось бы и для поля *Цена* сделать то же самое. А если заглянуть вперед, то мы увидим, что подобное автоматическое заполнение поля *Сумма* может нам понадобиться и в других документах.

Поэтому лучше будет поместить расчет суммы в некоторое «общедоступное» место, чтобы разные документы, имеющие аналогичные реквизиты табличной части, могли использовать этот алгоритм.

Для описания таких «общедоступных» мест служат объекты конфигурации *Общий модуль*, расположенные в ветке *Общие* > *Общие модули*. Процедуры и функции, содержащиеся в этих модулях, могут быть доступны для любых объектов конфигурации.

Поэтому создадим общий модуль и перенесем в него нашу процедуру расчета суммы. А в документе просто оставим вызовы этой процедуры из общего модуля.

В режиме «Конфигуратор»

Общий модуль

Добавим объект конфигурации *Общий модуль*.

Для этого раскроем ветвь *Общие* в дереве объектов конфигурации, нажав на + слева от нее. Затем выделим ветвь *Общие модули* и нажмем кнопку *Добавить* в командной панели окна конфигурации (рис. 4.23).

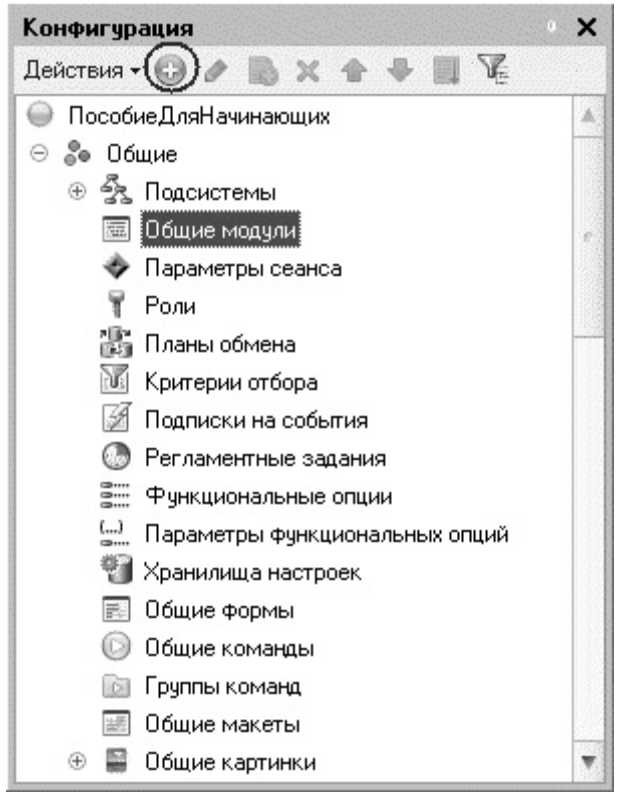


Рис. 4.23. Создание общего модуля в дереве объектов конфигурации

Назовем его *РаботаСДокументами* и установим в его свойствах флажок *Клиент (управляемое приложение)*, а флажок *Сервер* снимем. Это означает, что экземпляры этого модуля будут скомпилированы в контексте тонкого клиента и в контексте веб-клиента (рис. 4.24).

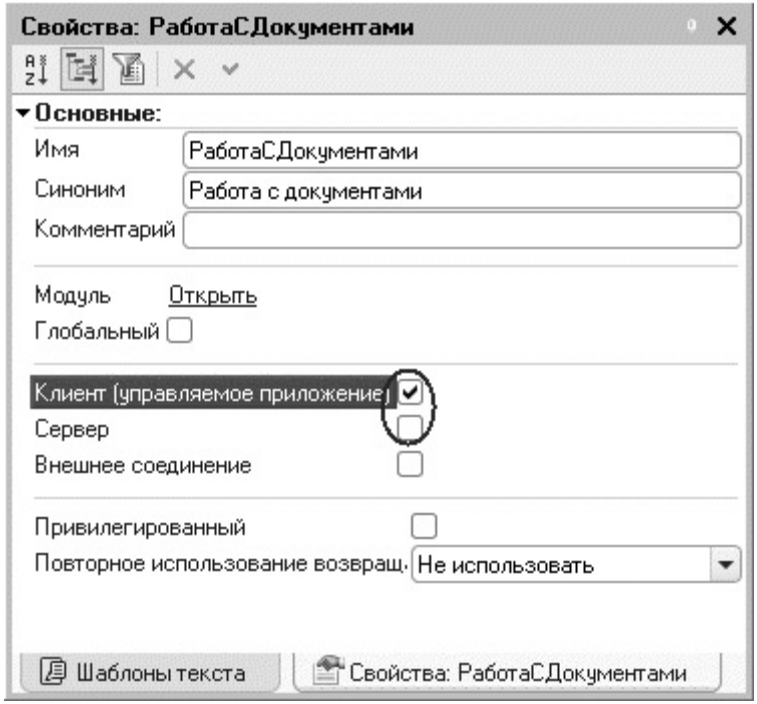


Рис. 4.24. Свойства общего модуля

Внесем в модуль следующий текст (листинг 4.2).

Листинг 4.2. Процедура «РассчитатьСумму()»

```
Процедура РассчитатьСумму(СтрокаТабличнойЧасти) Экспорт

  СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество *
  СтрокаТабличнойЧасти.Цена;
```

```
КонецПроцедуры
```

Прокомментируем этот код. В процедуру *РассчитатьСумму()* мы передаем переменную *СтрокаТабличнойЧасти*, которую мы определили в обработчике события *ПриИзменении* поля *Количество*. Она содержит данные редактируемой строки табличной части документа *ПриходнаяНакладная*.

Теперь, используя эту переменную, мы можем получить доступ к данным колонок табличной части и рассчитать сумму как произведение цены на количество.

Ключевое слово *Экспорт* в заголовке процедуры указывает на то, что эта процедура может быть доступна из других программных модулей.

Теперь в модуле нашей формы изменим текст обработчика (листинг 4.3).

Листинг 4.3. Процедура «МатериалыКоличествоПриИзменении()»

```
&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)

  СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
  РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
```

```
КонецПроцедуры
```

Мы видим, что первая строка процедуры осталась без изменений. А во второй строке вместо непосредственного расчета суммы мы вызываем процедуру *РассчитатьСумму()* из общего модуля *РаботаСДокументами* и передаем ей в качестве параметра текущую строку табличной части.

Проверим, как это работает, и убедимся, что ничего не изменилось.

Теперь осталось и для поля *Цена* установить такой же обработчик. Так как однажды мы уже написали в модуле формы нужную нам процедуру, то мы просто могли бы сопоставить ее также и другому событию другого элемента управления, расположенного в форме. Однако стандарты разработки конфигураций фирмы «1С» не допускают такого решения.

Узнай больше!

Согласно стандартам разработки фирмы «1С» у каждого события должен быть свой обработчик. Если одинаковые действия должны выполняться при изменении разных элементов управления (например, при нажатии нескольких кнопок), то в этом случае следует поступать следующим образом:

– создается отдельная процедура (функция), выполняющая необходимые действия;

– для каждого элемента управления создается отдельный обработчик с именем, назначаемым по умолчанию;

– из каждого обработчика вызывается требуемая процедура (функция).

Поэтому мы создадим обработчик события *ПриИзменении* для поля табличной части *МатериалыЦена* так же, как мы делали это для поля *МатериалыКоличество,* и повторим в нем вызов процедуры *РассчитатьСумму* из общего модуля (листинг 4.4).

Листинг 4.4. Процедура «МатериалыЦенаПриИзменении()»

```
&НаКлиенте
Процедура МатериалыЦенаПриИзменении(Элемент)

  СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
  РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры
```

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и убедимся, что теперь сумма в строках табличной части документов *ПриходнаяНакладная* пересчитывается как при изменении количества, так и при изменении цены.

Документ «Оказание услуги»

Теперь мы аналогичным образом создадим второй документ, необходимый нам, – *Оказание услуги*. Для этого потребуется выполнить уже знакомые нам действия, которые мы выполняли по созданию документа «Приходная накладная».

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Документ* и назовем его *ОказаниеУслуги*.

На закладке *Основные* определим, как будет представлен документ в интерфейсе «1С:Предприятия».

Представление объекта задавать не будем, вместо него будет использоваться *Синоним* объекта.

Представление списка зададим как *Оказание услуг*. На закладке *Подсистемы* отметим, что документ будет доступен в подсистемах *Оказание*

услуг и *Бухгалтерия*. На закладке *Данные* создадим реквизиты документа:

- *Склад*, тип *СправочникСсылка.Склады.* Выберем для свойства *Значение заполнения* предопределенный элемент *Основной* справочника *Склады*;
- *Клиент*, тип *СправочникСсылка.Клиенты*. Установим свойство *Проверка заполнения* в значение *Выдавать ошибку*;
- *Мастер*, тип *СправочникСсылка.Сотрудники*. Установим свойство *Проверка заполнения* в значение *Выдавать ошибку*.

Создадим табличную часть этого документа *ПеречеьНоменклатуры* с реквизитами:

- *Номенклатура*, тип *СправочникСсылка.Номенклатура*;
- *Количество*, тип *Число*, длина *15*, точность *3*, неотрицательное;
- *Цена*, тип *Число*, длина *15*, точность *2*, неотрицательное,
- *Сумма*, тип *Число*, длина *15*, точность *2*, неотрицательное.

Установим для табличной части в целом и для каждого ее реквизита свойство *Проверка заполнения* в значение *Выдавать ошибку*.

На закладке *Формы* создадим основную форму документа.

Для поля *ПеречеьНоменклатурыКоличество* создадим обработчик события *ПриИзменении*, в котором будем вызывать процедуру *РассчитатьСумму* из общего модуля *РаботаСДокументами*.

При этом откроется модуль формы с шаблоном обработчика события *ПеречеьНоменклатурыКоличествоПриИзменении*, который мы пока заполнять не будем, а перейдем в окно элементов формы на закладку *Форма* и аналогичным образом создадим обработчик события *ПеречеьНоменклатурыЦенаПриИзменении* для поля *ПеречеьНоменклатурыЦена*.

Далее модуль формы документа *ОказаниеУслуги* нужно заполнить следующим образом (листинг 4.5).

Листинг 4.5. Модуль формы документа «ОказаниеУслуги»

```
&НаКлиенте
Процедура ПеречеьНоменклатурыКоличествоПриИзменении(Элемент)

  СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;
  РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры

&НаКлиенте
```

```
Процедура ПеречеьНоменклатурыЦенаПриИзменении(Элемент)

  СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;
  РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры
```

В заключение отредактируем командный интерфейс, чтобы в подсистеме *Оказание услуг* была доступна команда создания новых документов.

Для разнообразия воспользуемся другим способом. Откроем окно редактирования объекта конфигурации *Подсистема ОказаниеУслуг* и нажмем кнопку *Командный интерфейс*.

В открывшемся окне отразятся все команды выбранной подсистемы.

В группе *Панель действий.Создать* включим видимость у команды *Оказание услуги: создать* (рис. 4.25).

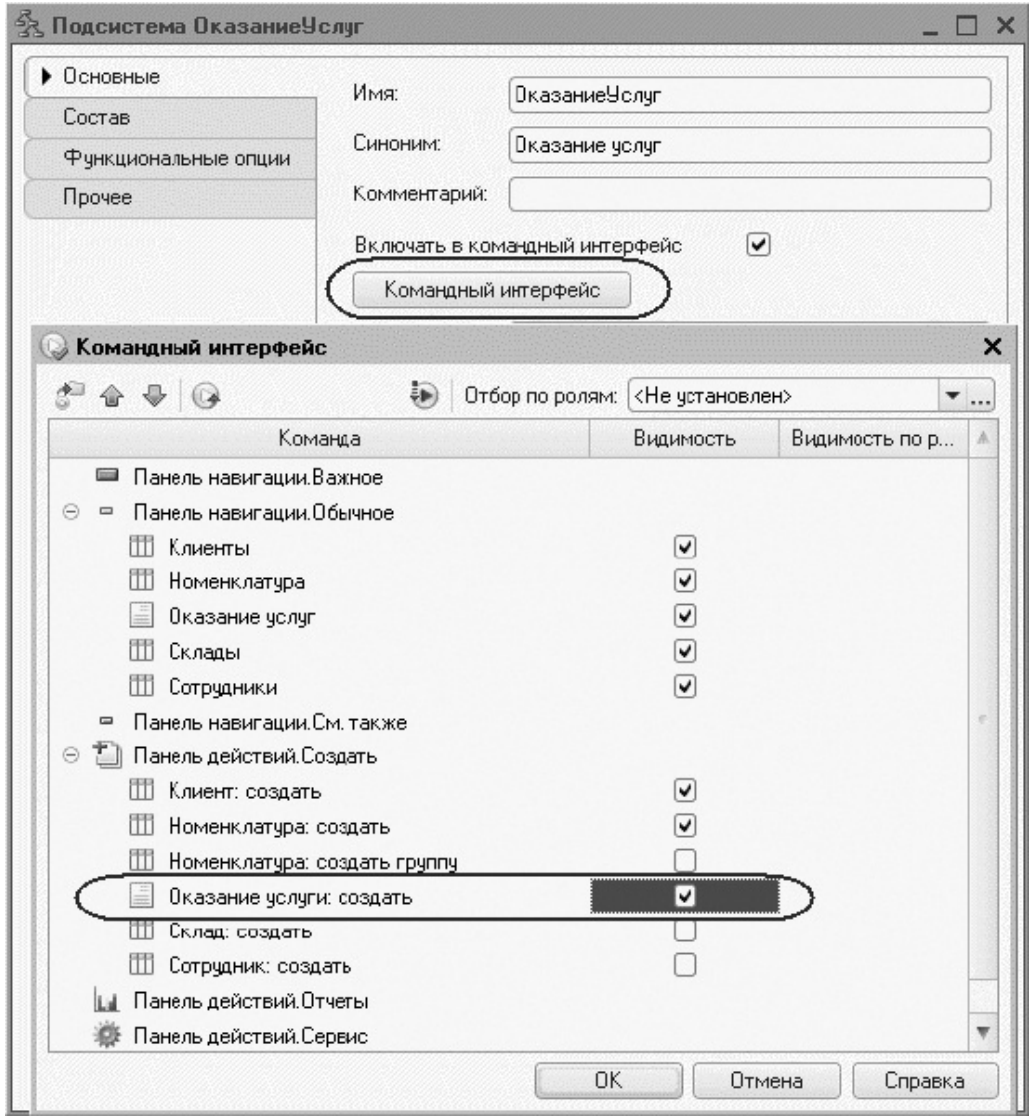

Подсистема ОказаниеУслуг
Основные
Состав
Функциональные опции
Прочее
Имя:
ОказаниеУслуг
Синоним:
Оказание услуг
Комментарий:
Включать в командный интерфейс
Командный интерфейс
Командный интерфейс
Отбор по ролям:
<Не установлен>
Команда
Видимость
Видимость по р...
Панель навигации.Важное
Панель навигации.Обычное
Клиенты
Номенклатура
Оказание услуг
Склады
Сотрудники
Панель навигации.См. также
Панель действий.Создать
Клиент: создать
Номенклатура: создать
Номенклатура: создать группу
Оказание услуги: создать
Склад: создать
Сотрудник: создать
Панель действий.Отчеты
Панель действий.Сервис
OK
Отмена
Справка

Рис. 4.25. Настройка командного интерфейса подсистем

В результате наших действий в дереве объектов конфигурации документ *ОказаниеУслуги* будет выглядеть следующим образом (рис. 4.26).

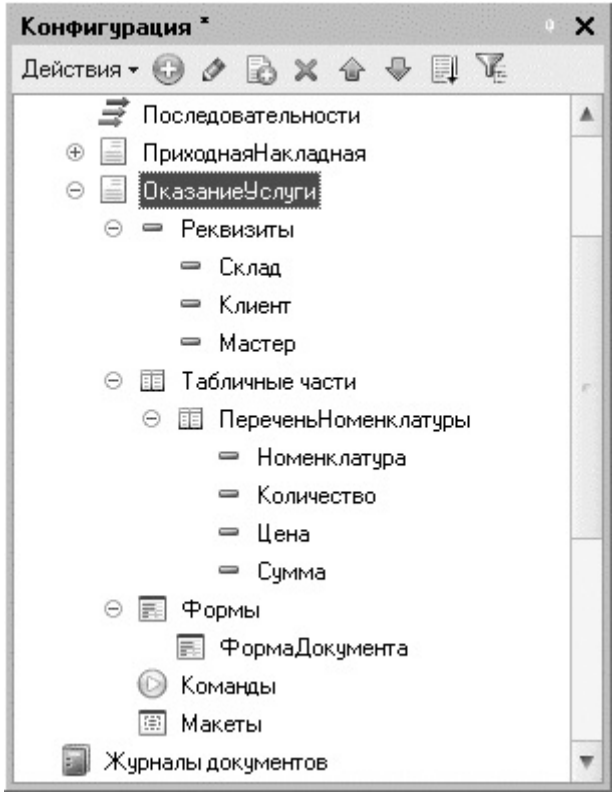


Рис. 4.26. Документ «ОказаниеУслуги» в дереве объектов конфигурации

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

В панели действий раздела *Оказание услуг* вызовем команду создания документа *Оказание услуги* и заполним его следующим образом (рис. 4.27).

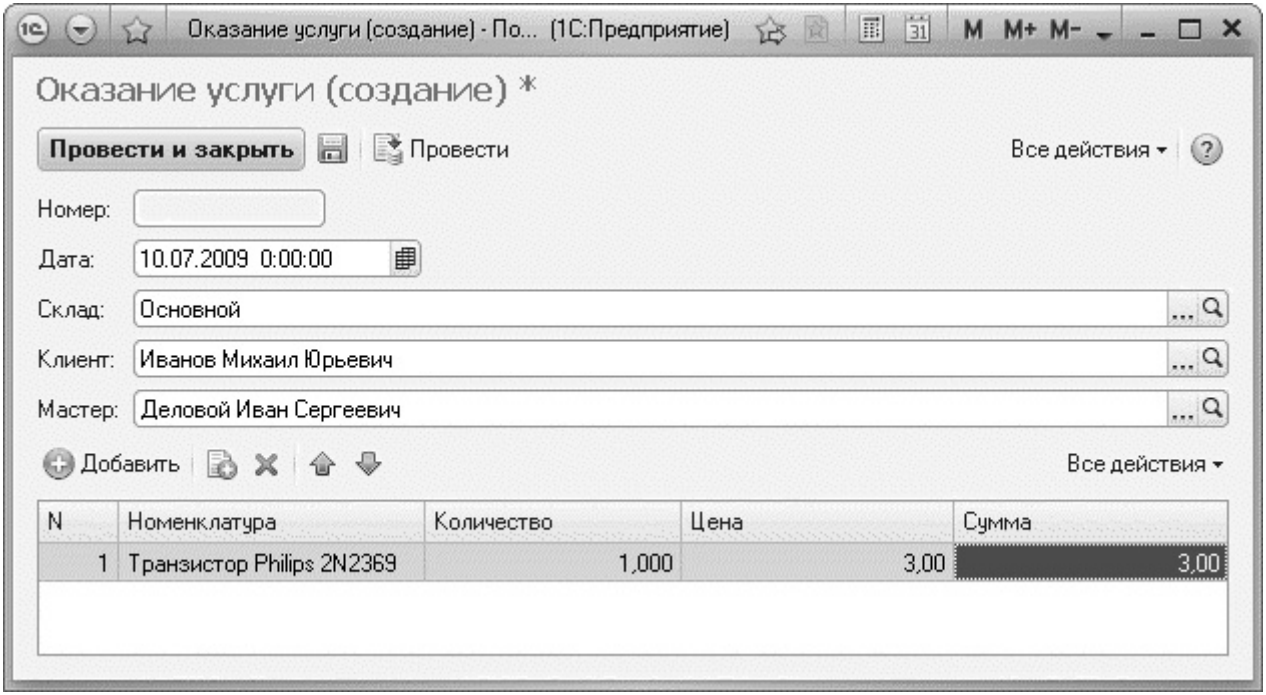


Рис. 4.27. Создание документа «Оказание услуги № 1»

Обратите внимание, что склад *Основной* подставляется по умолчанию, а для полей *Мастер* и *Клиент* выполняется проверка заполнения.

А также при вводе цены и количества в табличную часть документа *Оказание услуги* сумма пересчитывается по нашему алгоритму.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Документ».*
- *Какими характерными особенностями обладает документ.*
- *Для чего предназначены реквизиты и табличные части документа.*
- *Какие существуют основные формы документа.*
- *Что такое проведение документа.*
- *Как создать объект конфигурации «Документ» и описать его основную структуру.*
- *Как создать новый документ и заполнить его данными.*
- *Как создать собственную форму документа.*
- *Что такое конструктор форм.*
- *Что такое редактор форм.*
- *Что такое элементы формы.*
- *Что такое события и с чем они связаны.*

- *Что такое обработчик события и как его создать.*
- *Что такое модуль и для чего он нужен.*
- *Зачем нужны общие модули.*
- *Что такое типообразующие объекты.*

Занятие 5 (2:00). Теоретическое

Продолжительность

Ориентировочная продолжительность занятия – 2 часа.

Чтобы не усложнять восприятие и без того объемного предыдущего занятия «Документы», все теоретические моменты были вынесены в это отдельное занятие. На нем будут подробно рассмотрены вопросы, которые мы опускали или бегло проходили ранее.

Эта теория очень важна для более глубокого понимания, почему мы делали так, а не иначе. Она необходима читателям, желающим в будущем самостоятельно разрабатывать конфигурации, а не просто шаг за шагом следовать данному пособию.

Однако если прямо сейчас вы не в состоянии усвоить этот раздел, можно его пропустить и вернуться к нему позднее. На выполнение сквозного примера разработки учебной конфигурации, рассмотренного в данной книге, это никак не повлияет. Итак…

Механизм основных форм

На предыдущем занятии мы создали форму документа *Приходная накладная* и назначили эту форму основной. Что это значит?

У всех прикладных объектов конфигурации существует некоторое количество основных форм. Они служат для отображения данных объекта в том или ином виде.

Если разработчик не назначит в качестве основных форм объекта свои собственные, система будет генерировать необходимые формы объекта самостоятельно, в те моменты, когда к ним происходит обращение.

Наличие такого механизма позволяет разработчику не тратить время на создание форм для тестирования своей разработки, а воспользоваться тем, что платформа создаст по умолчанию.

Создание этих форм происходит динамически, в процессе работы системы. Форма создается в тот момент, когда к ней происходит обращение. Причем не важно, интерактивное это обращение или программное.

Так, форма списка для справочника *Клиенты* будет создана как при интерактивном выборе в меню *Все функции* > *Справочники* > *Клиенты*, так и при программном вызове глобального метода *ПолучитьФорму()* (листинг 5.1).

Листинг 5.1. Программный вызов метода «ПолучитьФорму()»

```
ФормаСписка = ПолучитьФорму("Справочник.Клиенты.ФормаСписка");
```

Также примечательным фактом является то, что состав основных форм, определенных для объекта конфигурации, может не совпадать с перечнем тех форм, которые вообще можно создать для данного объекта, используя конструктор формы.

Например, для большинства регистров в конфигураторе можно задать основную форму списка. Однако если открыть конструктор форм для регистра, вы увидите, что кроме формы списка предлагается создать и форму набора записей регистра, которая отсутствует в перечне основных форм.

Дело в том, что состав основных форм определяется исходя из того, какое представление данных может понадобиться в процессе интерактивной работы пользователя. Для этих представлений разработчик может создать свои формы и указать их в качестве основных, а может использовать те формы, которые система создаст автоматически.

Конструктор форм, напротив, исходит из потребностей разработчика. Если разработчик посчитает нужным использовать для какого-либо регистра вместо обычной формы списка форму набора записей, он сможет это сделать,

воспользовавшись конструктором и определив ее в качестве основной формы регистра. Но для логики работы системы это не будет иметь принципиального значения.

Обработчики событий

На предыдущем занятии мы создавали обработчики событий *ПриИзменении* у некоторых элементов формы. Что это такое?

При работе с событиями в платформе «1С:Предприятие 8» следует различать два типа событий: события, связанные с формой и ее элементами, и все остальные.

Разница заключается в том, что обработчики событий, связанных с формой и ее элементами, – назначаемые, а обработчики всех остальных событий – фиксированные.

Фиксированный обработчик события должен иметь имя, совпадающие с именем события. Только в этом случае он будет вызываться при возникновении соответствующего события.

Назначаемый обработчик может иметь произвольное имя. Если имя процедуры совпадает с именем события формы или ее элемента, этого совсем

недостаточно для вызова процедуры обработки события с таким именем. Требуется явное назначение процедуры обработчиком этого события в палитре свойств, в соответствующем событии.

Таким образом, любая процедура, расположенная в модуле формы, может быть назначена обработчиком любого события (или сразу нескольких событий) формы или ее элемента, расположенного в форме. Имя процедуры в этом случае не имеет значения. Важно лишь то, что она назначена обрабатывать какое-либо событие.

Назначение обработчика может выполняться интерактивно, при работе с формой в конфигураторе, или программно, используя методы формы и ее элементов – *УстановитьДействие()*.

Модули

На предыдущем занятии мы рассматривали код обработчиков событий. Мы узнали, что эти процедуры располагаются в модуле формы – неком хранилище текста программы на встроенном языке.

Теперь расскажем о модулях подробнее и внимательнее познакомимся с устройством модуля формы.

Виды модулей

В конфигурации существуют различные виды модулей. Они могут принадлежать некоторым объектам конфигурации (например, формам), а могут существовать сами по себе (принадлежать всей конфигурации в целом).

Текст программы, содержащийся в модулях, будет использоваться платформой в заранее известные моменты работы системы «1С:Предприятие» – *события*, о которых мы рассказывали ранее.

В «1С:Предприятии 8» существуют следующие виды модулей.

Модуль управляемого приложения. Модуль управляемого приложения выполняется при старте системы «1С:Предприятие» в режимах тонкого клиента и веб-клиента.

В нем возможно объявление переменных, а также объявление и описание процедур и функций, которые будут доступны в любом модуле конфигурации (кроме модуля внешнего соединения). Их доступность также обеспечивается для неглобальных общих модулей с установленным свойством *Клиент (управляемое приложение).* В контексте модуля управляемого приложения доступны экспортируемые процедуры и функции общих модулей.

Чтобы открыть модуль управляемого приложения, нужно выделить корень дерева объектов конфигурации (строку *ПособиеДляНачинающих*) и вызвать из контекстного меню команду *Открыть модуль управляемого приложения* (рис. 5.1)

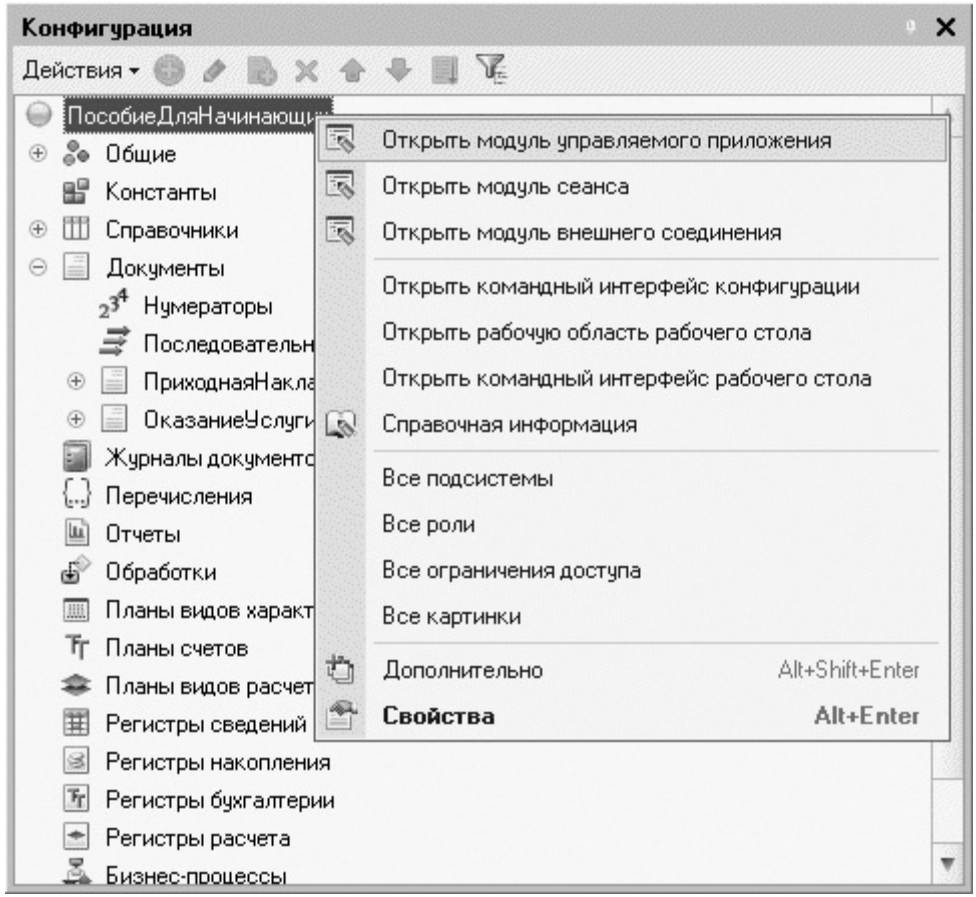


Рис. 5.1. Открытие модуля управляемого приложения

Общие модули. В общих модулях хранятся процедуры и функции, которые вызываются из других модулей системы. Сам по себе общий модуль не исполняется. Исполняются отдельные его процедуры/функции в момент их

вызова из других модулей.

Чтобы открыть общий модуль, нужно раскрыть ветвь *Общие* в дереве объектов конфигурации, затем раскрыть ветвь *Общие модули* и дважды щелкнуть мышью на нужном модуле (рис. 5.2).

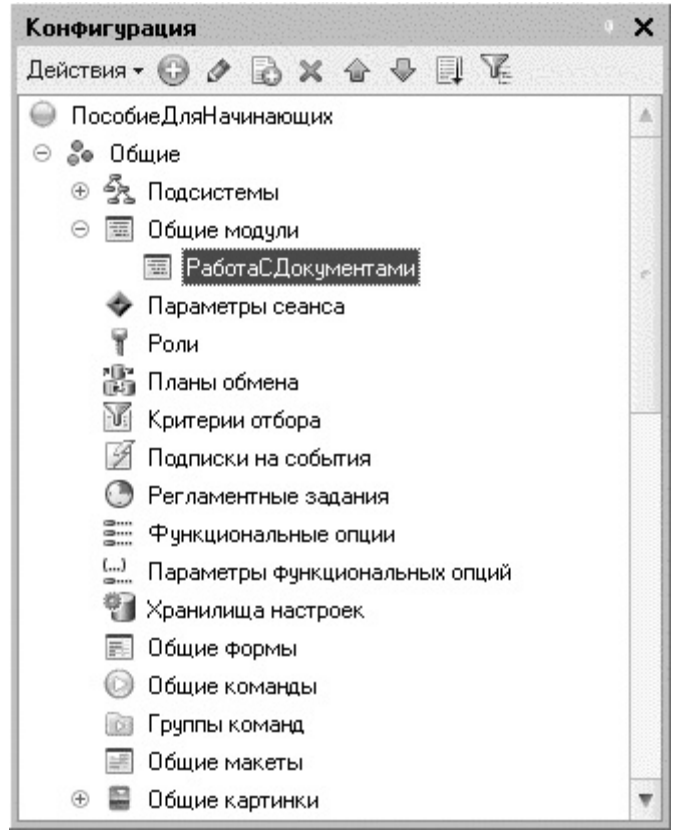


Рис. 5.2. Открытие общего модуля

Модули объектов. Модули объектов – это, например, модуль элемента справочника или модуль документа.

Эти модули вызываются тогда, когда либо программно создается этот объект средствами встроенного языка, например, методами *СоздатьЭлемент()* у менеджеров справочников или *СоздатьДокумент()* менеджеров документов, либо когда пользователь создает новый элемент справочника или документ интерактивно.

При записи измененных данных объекта в базу данных вызываются различные обработчики событий, которые располагаются в модуле объекта.

Более подробно о последовательности событий при записи объектов конфигурации в базу данных рассказывается в разделе «Краткий справочник разработчика».

Чтобы открыть модуль объекта, нужно в окне редактирования объекта конфигурации перейти на закладку *Прочее* и нажать кнопку *Модуль объекта* (рис. 5.3). Или, выделив нужный объект в дереве объектов конфигурации, вызвать из контекстного меню команду *Открыть модуль объекта*.

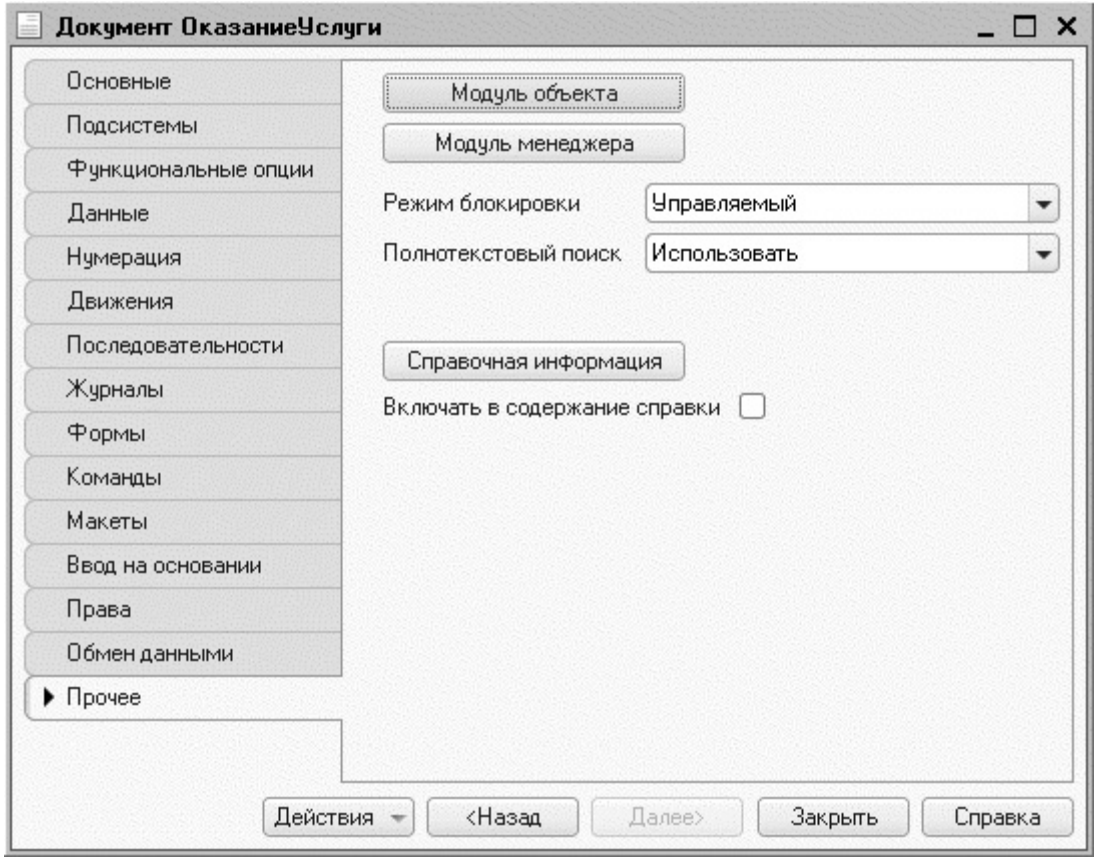


Рис. 5.3. Открытие модуля объекта

Модули форм. Каждая форма, определенная в конфигурации, имеет свой собственный модуль. Этот модуль исполняется при создании объекта *УправляемаяФорма* встроенного языка. Этот объект создается системой в

режиме *1С:Предприятие* в тот момент, когда мы программно (методами *ПолучитьФорму()* или *ОткрытьФорму()*) или интерактивно открываем форму некоторого элемента.

Чтобы открыть модуль формы, нужно открыть нужный объект конфигурации *Форма* объекта и в окне редактора форм перейти на закладку *Модуль* (рис. 5.4).

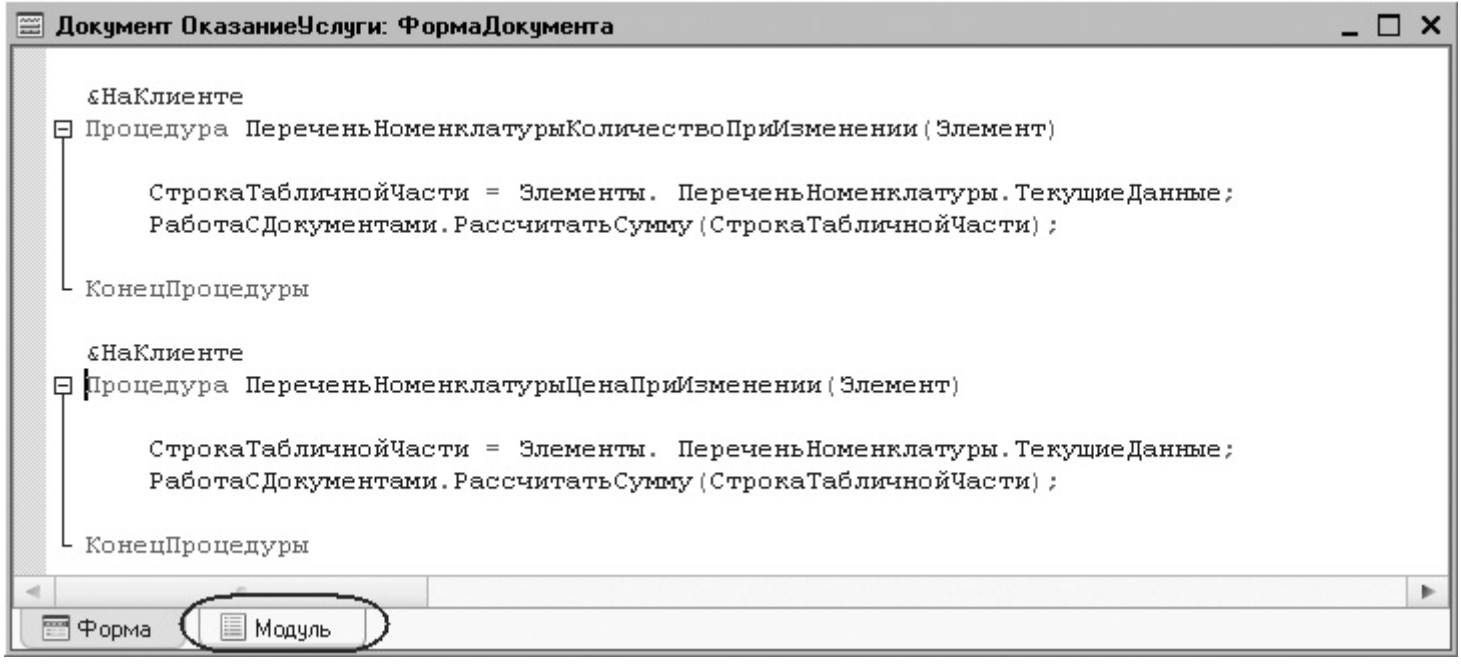


Рис. 5.4. Открытие модуля формы

Модуль сеанса. Модулем сеанса называется модуль, который автоматически выполняется при старте системы «1С:Предприятие 8» в момент загрузки конфигурации. Модуль сеанса предназначен для инициализации параметров сеанса и отработки действий, связанных с сеансом работы. Модуль сеанса не содержит экспортируемых процедур и функций и может использовать процедуры из общих модулей конфигурации.

Чтобы открыть модуль сеанса, нужно выделить корень дерева объектов конфигурации (строку *ПособиеДляНачинающих*) и вызвать из контекстного меню команду *Открыть модуль сеанса* (рис. 5.5).

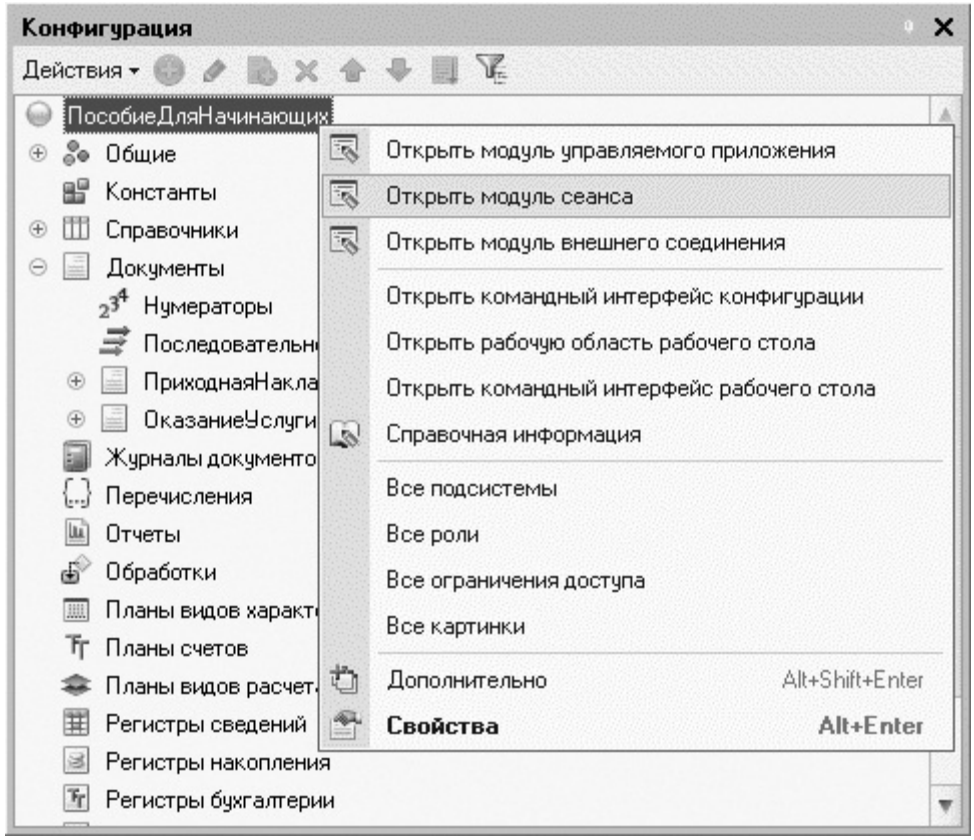


Рис. 5.5. Открытие модуля сеанса

Модуль внешнего соединения предназначен для размещения в нем текстов функций и процедур, которые могут вызываться в сессии внешнего соединения.

Чтобы открыть модуль сеанса, нужно выделить корень дерева объектов

конфигурации (строку *ПособиеДляНачинающих*) и вызвать из контекстного меню команду *Открыть модуль внешнего соединения* (см. рис. 5.5).

Модуль менеджеров. Для каждого прикладного объекта существует менеджер, предназначенный для управления этим объектом как объектом конфигурации. С помощью менеджера можно создавать объекты, работать с формами и макетами. Модуль менеджера позволяет расширить функциональность менеджеров, предоставляемых системой, за счет написания процедур и функций на встроенном языке.

Фактически это позволяет описать собственные методы для объекта конфигурации (например, справочника), которые относятся не к конкретному экземпляру объекта базы данных, а к самому объекту конфигурации.

Чтобы открыть модуль менеджера, нужно в окне редактирования объекта конфигурации перейти на закладку *Прочее* и нажать кнопку *Модуль менеджера* (рис. 5.6). Или, выделив нужный объект в дереве объектов конфигурации, вызвать из контекстного меню команду *Открыть модуль менеджера*.

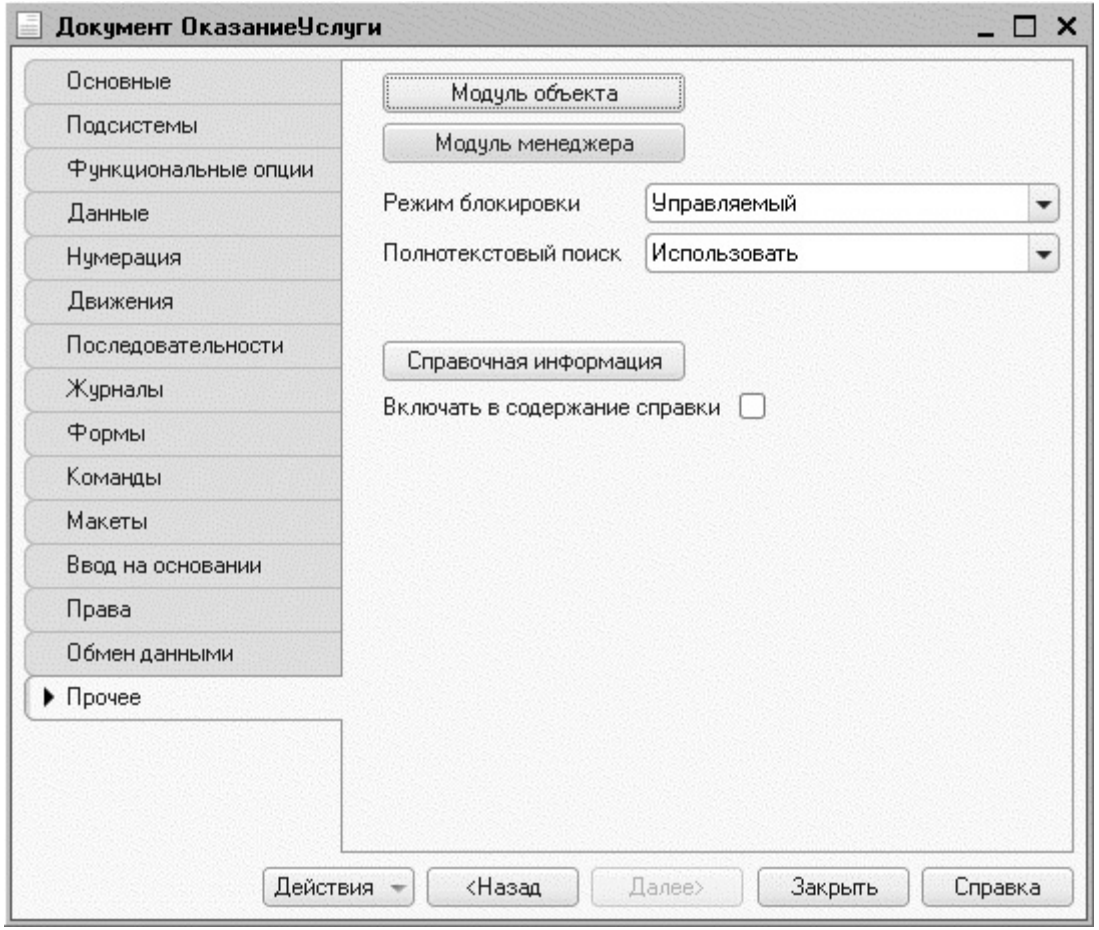


Рис. 5.6. Открытие модуля менеджера

Модуль команды. Как в самой конфигурации, так и у многих прикладных объектов могут существовать подчиненные объекты конфигурации – *Команды*.

У каждой команды существует модуль команды, в котором можно написать предопределенную процедуру *ОбработкаКоманды()* для выполнения этой команды.

Чтобы открыть модуль команды, подчиненной некоторому объекту конфигурации, нужно в окне редактирования объекта конфигурации перейти на закладку *Команды* и дважды щелкнуть мышью на нужной команде (рис. 5.7). Или, выделив нужную команду в дереве объектов конфигурации, вызвать из контекстного меню команду *Открыть модуль команды*.

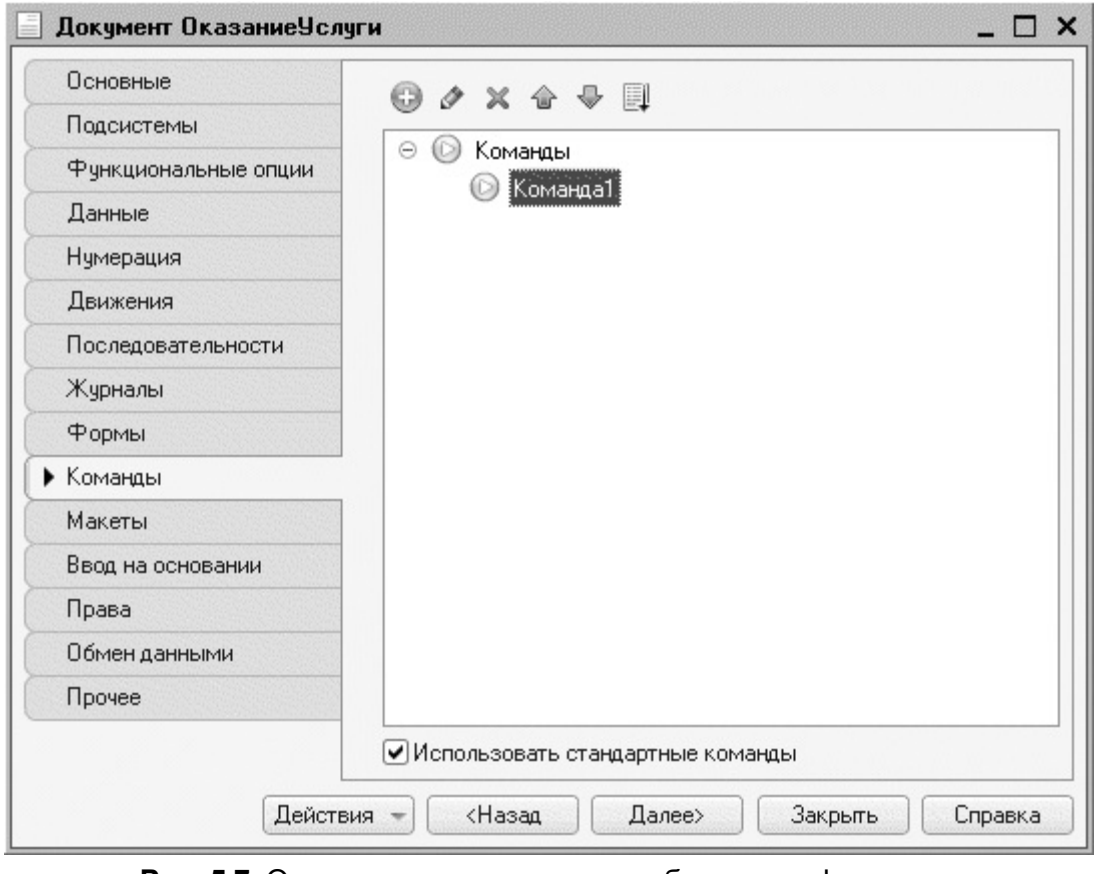


Рис. 5.7. Открытие модуля команды объекта конфигурации

Контекст модуля формы

Каждый модуль связан с остальной частью конфигурации. Эта связь называется *контекстом модуля*.

Контекст модуля определяет набор доступных во время выполнения модуля объектов, переменных, процедур и функций.

Поскольку дальше речь пойдет о том обработчике, который мы написали в модуле формы, рассмотрим подробнее, из чего складывается контекст модуля формы.

Контекст модуля формы образуется:

- локальным контекстом самого модуля формы;
- реквизитами формы, которой «принадлежит» модуль;
- свойствами и методами объекта *УправляемаяФорма* встроенного языка;
- свойствами и методами расширения формы, определяемого типом того объекта, данные которого содержатся в основном реквизите формы;
- глобальным контекстом, в том числе неглобальными общими модулями и экспортируемыми функциями и процедурами глобальных общих модулей;
- экспортируемыми переменными, процедурами и функциями модуля управляемого приложения.

Рассмотрим подробнее, что собой представляет каждая из перечисленных составляющих.

1. Локальный контекст модуля формы.

Локальный контекст модуля формы – это переменные, процедуры и функции, объявленные в этом модуле.

Например, внутри модуля формы можно непосредственно обращаться по имени к процедуре *ПолучитьСумму()*, объявленной в этом же модуле (листинг 5.2):

Листинг 5.2. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  ПолучитьСумму();

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ПолучитьСумму()
  …
КонецПроцедуры
```

Или внутри модуля формы можно непосредственно обращаться по имени к переменной *СлужебнаяПеременная*, объявленной в этом модуле (листинг 5.3):

Листинг 5.3. Модуль формы

```
&НаКлиенте
Перем СлужебнаяПеременная;

&НаКлиенте
Процедура Команда1()

  СлужебнаяПеременная = 3;

КонецПроцедуры
```

2. Реквизиты формы, которой «принадлежит» модуль.

Например, если у формы существует реквизит *СлужебныйРеквизит* (рис. 5.8), то к нему можно непосредственно обращаться по имени (листинг 5.4):

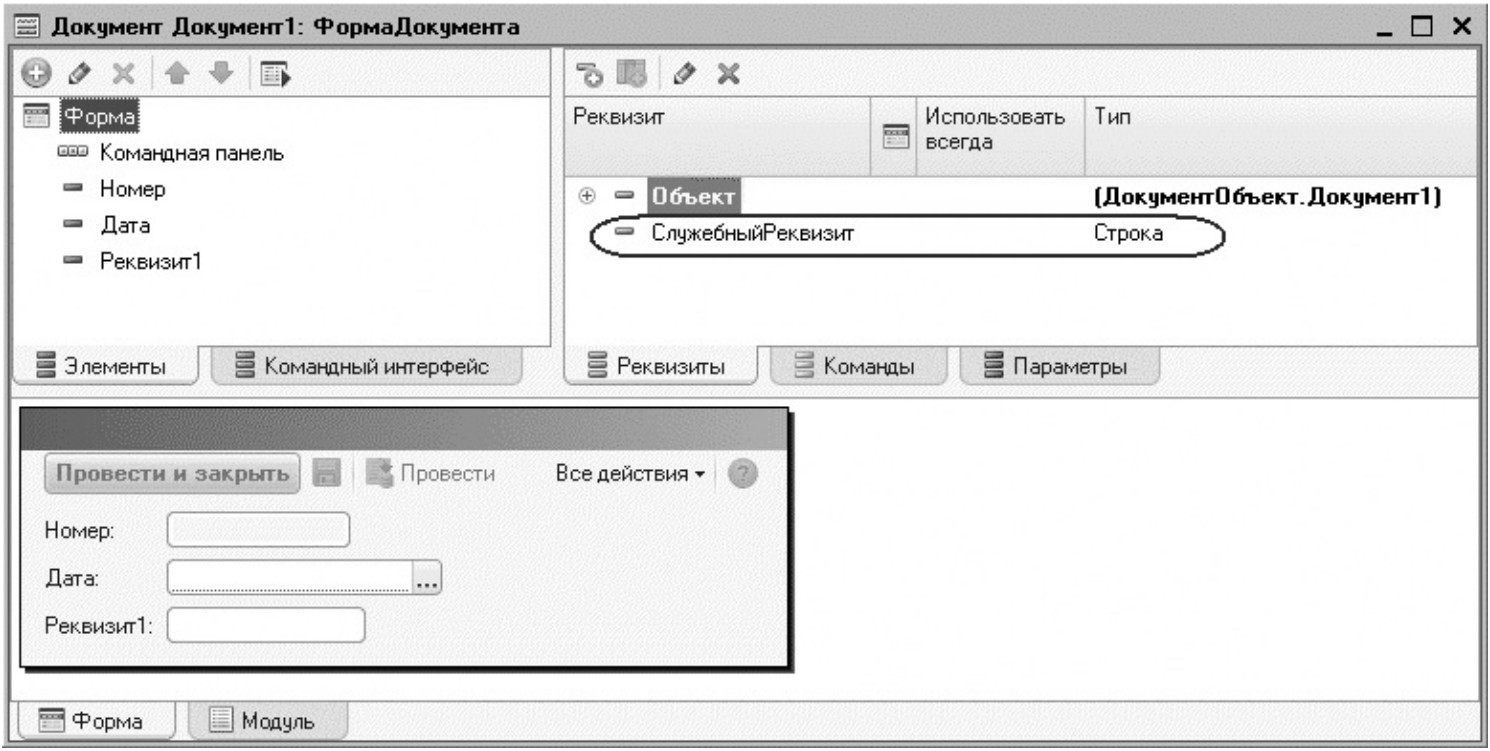


Рис. 5.8. Реквизит формы – «Служебный реквизит»

Листинг 5.4. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  СлужебныйРеквизит = 3;

КонецПроцедуры
```

3. Свойства и методы объекта «УправляемаяФорма»

Свойства и методы объекта встроенного языка *УправляемаяФорма* описаны в синтакс-помощнике: *Интерфейсные объекты управляемого приложения* > *УправляемаяФорма* (рис. 5.9).

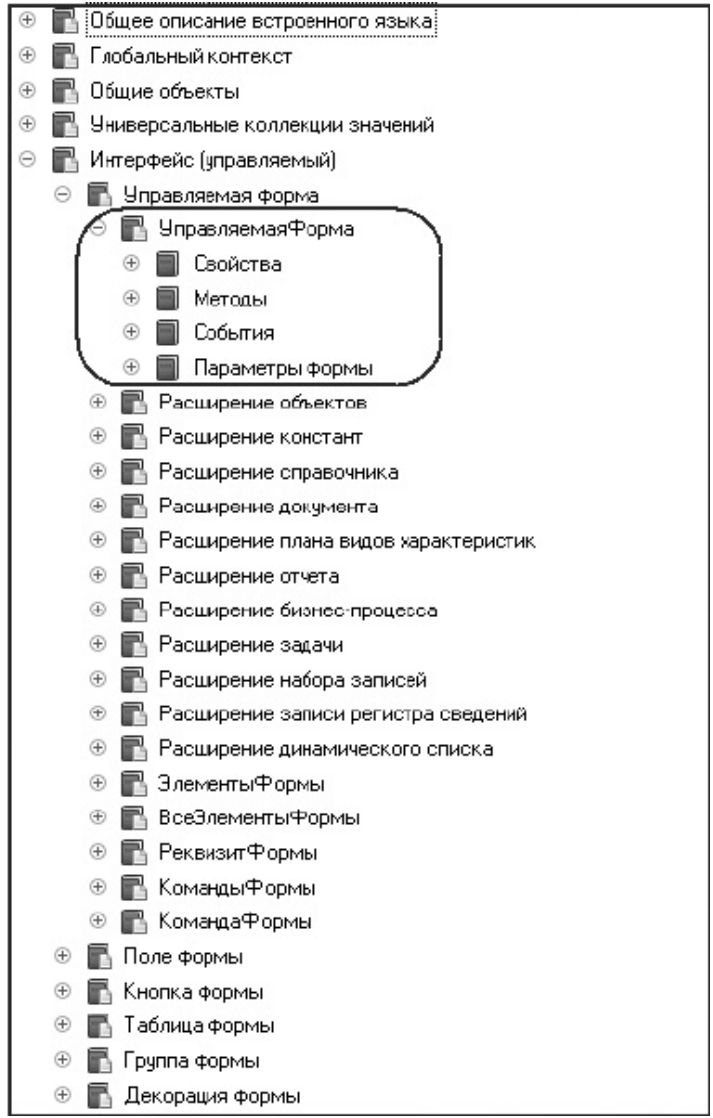

Общее описание встроенного языка
Глобальный контекст
Общие объекты
Универсальные коллекции значений
Интерфейс (управляемый)
Управляемая форма
УправляемаяФорма
Свойства
Методы
События
Параметры формы
Расширение объектов
Расширение констант
Расширение справочника
Расширение документа
Расширение плана видов характеристик
Расширение отчета
Расширение бизнес-процесса
Расширение задачи
Расширение набора записей
Расширение записи регистра сведений
Расширение динамического списка
ЭлементыФормы
ВсеЭлементыФормы
РеквизитФормы
КомандыФормы
КомандаФормы
Поле формы
Кнопка формы
Таблица формы
Группа формы
Декорация формы

Рис. 5.9. Свойства управляемой формы в синтакс-помощнике

К ним можно обращаться непосредственно по имени. Например, можно задать заголовок формы (листинг 5.5):

Листинг 5.5. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  Заголовок = "Новый заголовок формы";

КонецПроцедуры
```

Или можно закрыть форму (листинг 5.6):

Листинг 5.6. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  Закрыть();

КонецПроцедуры
```

4. Свойства и методы расширения формы, определяемого типом того

объекта, данные которого содержатся в основном реквизите формы.

Один из реквизитов формы может быть основным, в списке реквизитов он выделяется жирным. Как правило, основной реквизит формы содержит данные того объекта, который отображается в форме. Например, если это форма справочника, то основной реквизит будет содержать данные объекта *СправочникОбъект.<имя>* (рис. 5.10).

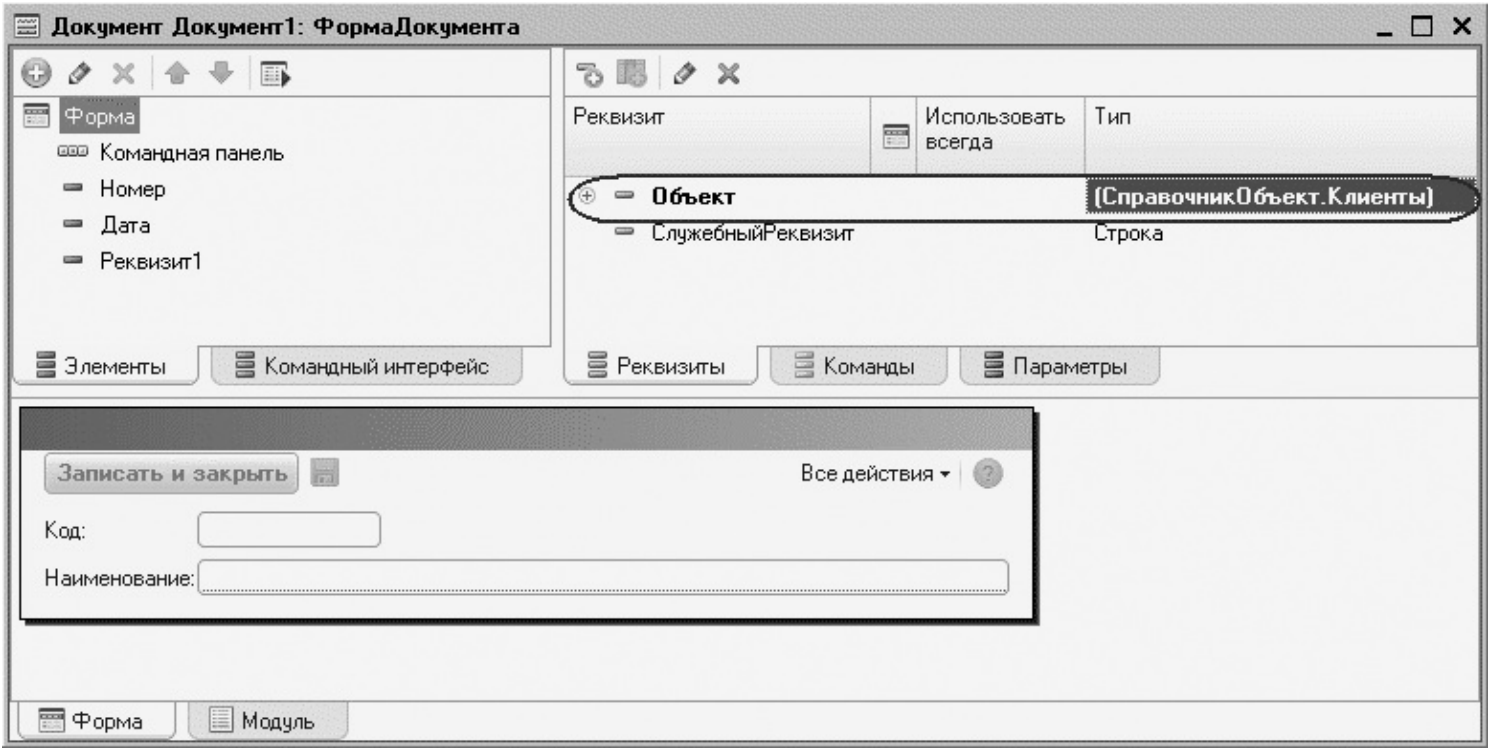


Рис. 5.10. Основной реквизит формы

А если это форма документа, то основной реквизит формы будет содержать данные объекта *ДокументОбъект.<имя>* (рис. 5.11).

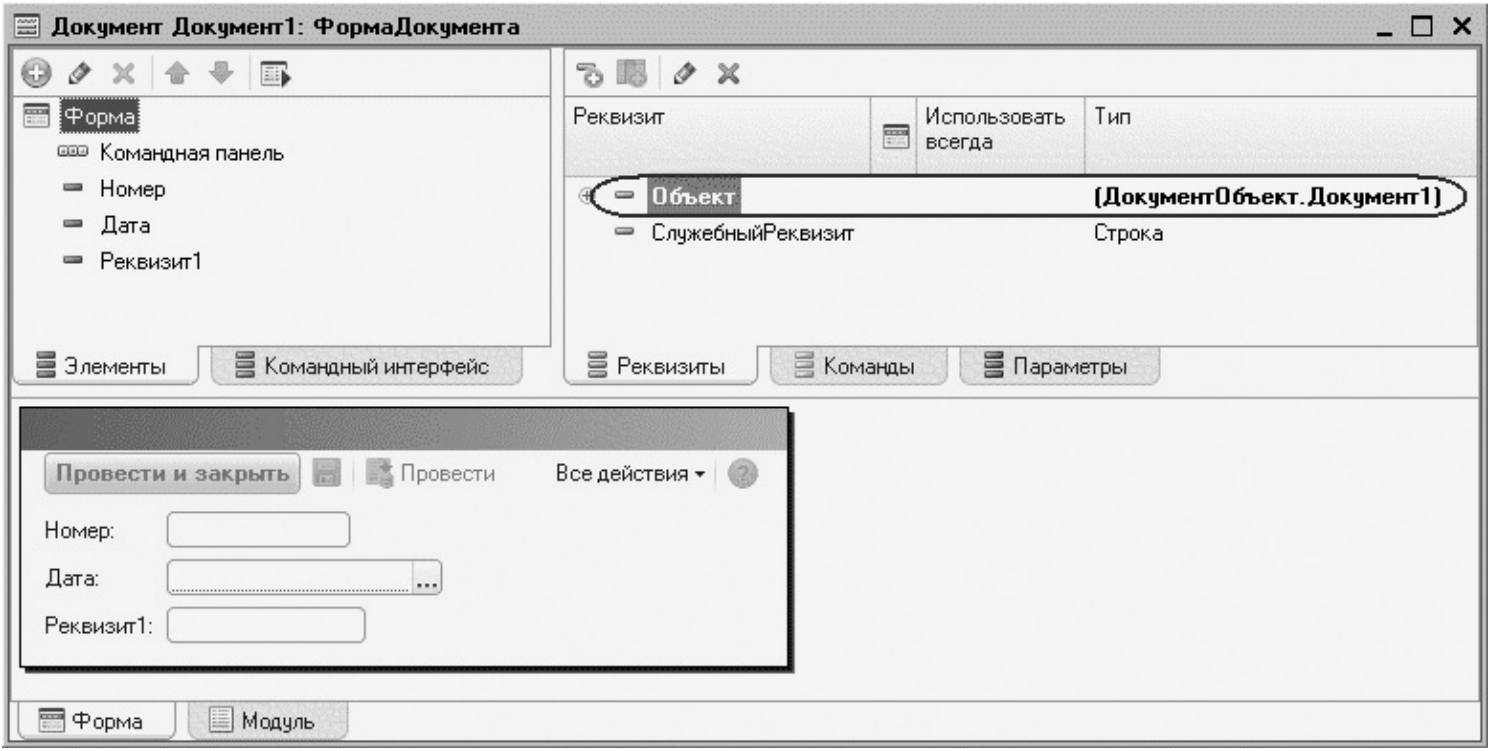


Рис. 5.11. Основной реквизит формы

Тут следует пояснить, почему в списке реквизитов тип основного реквизита указан в скобках: *(ДокументОбъект.Документ1)*. Дело в том, что это «ненастоящий» тип реквизита формы. Настоящий тип в данном случае будет *ДанныеФормыСтруктура* (рис. 5.12).

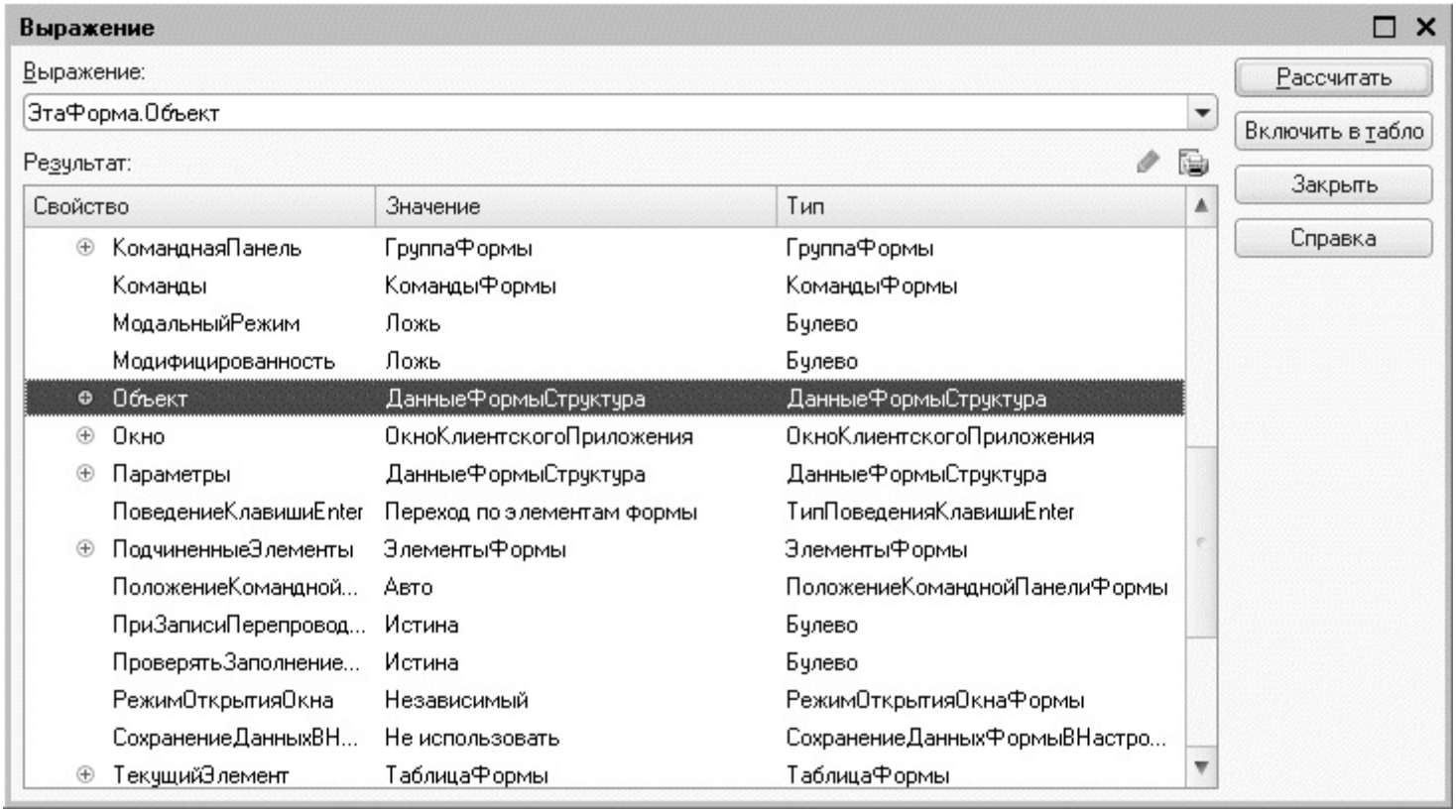


Рис. 5.12. Тип объекта основного реквизита формы

Но *ДанныеФормыСтруктура* – это универсальный тип, который может содержать данные различных прикладных объектов. Поэтому, чтобы в редакторе формы можно было «ориентироваться», данные какого же именно прикладного объекта отображает эта форма, в колонке *Тип* редактора

показывается не тип реквизита формы (*ДанныеФормыСтруктура*), а тип того объекта, данные которого содержатся в этом реквизите. И этот «ненастоящий» тип показывается в скобках.

От типа объекта, данные которого содержатся в основном реквизите формы, зависят некоторые особенности в поведении формы.

Например, если основной реквизит формы будет содержать данные документа, то при закрытии такой формы система будет запрашивать подтверждение записи и проведения этого документа. Если же основной реквизит формы будет содержать данные справочника, то подобного запроса подтверждения при закрытии формы возникать не будет.

В зависимости от того, какой тип имеет объект, данные которого содержатся в основном реквизите, к контексту программного объекта *УправляемаяФорма* добавляется контекст соответствующего расширения.

Например, если основной реквизит – *СправочникОбъект.<имя>*, то в модуле формы становятся доступны свойства, методы объекта встроенного языка *Расширение управляемой формы для справочника* (синтакс-помощник – *Интерфейс (управляемый)* > *Управляемая форма* > *Расширение справочника*), рис. 5.13.

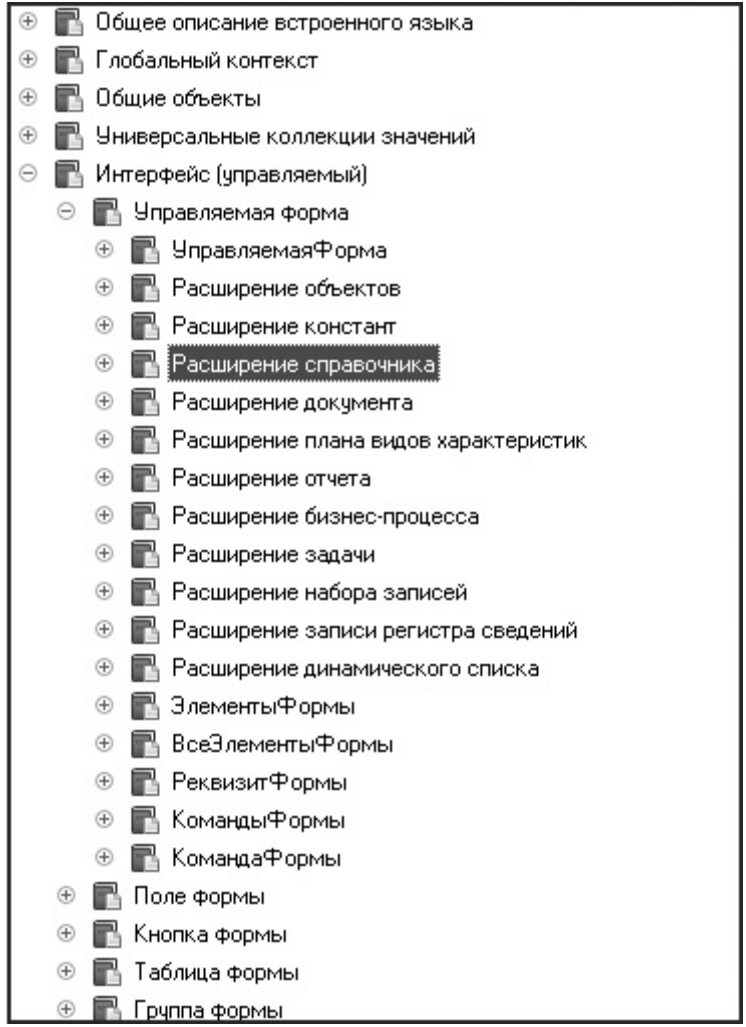


Рис. 5.13. Описание объектов в синтакс-помощнике

А если основной реквизит *ДинамическийСписок* (рис. 5.14), то в модуле формы становятся доступны свойства, методы объекта встроенного языка *Расширение управляемой формы для динамического списка* (синтакс-помощник – *Интерфейс (управляемый)* > *Управляемая форма* > *Расширение динамического списка*), рис. 5.15.

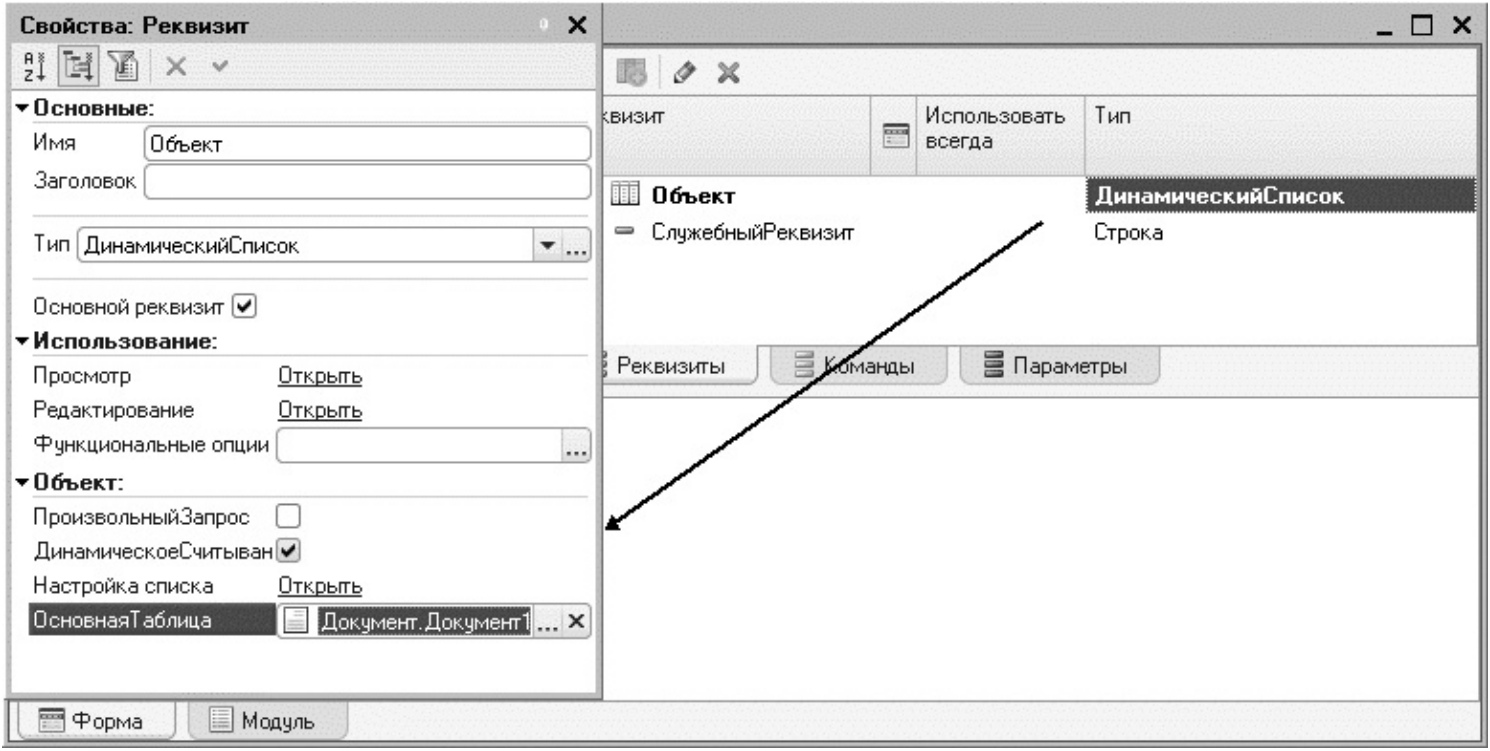


Рис. 5.14. Основной реквизит формы

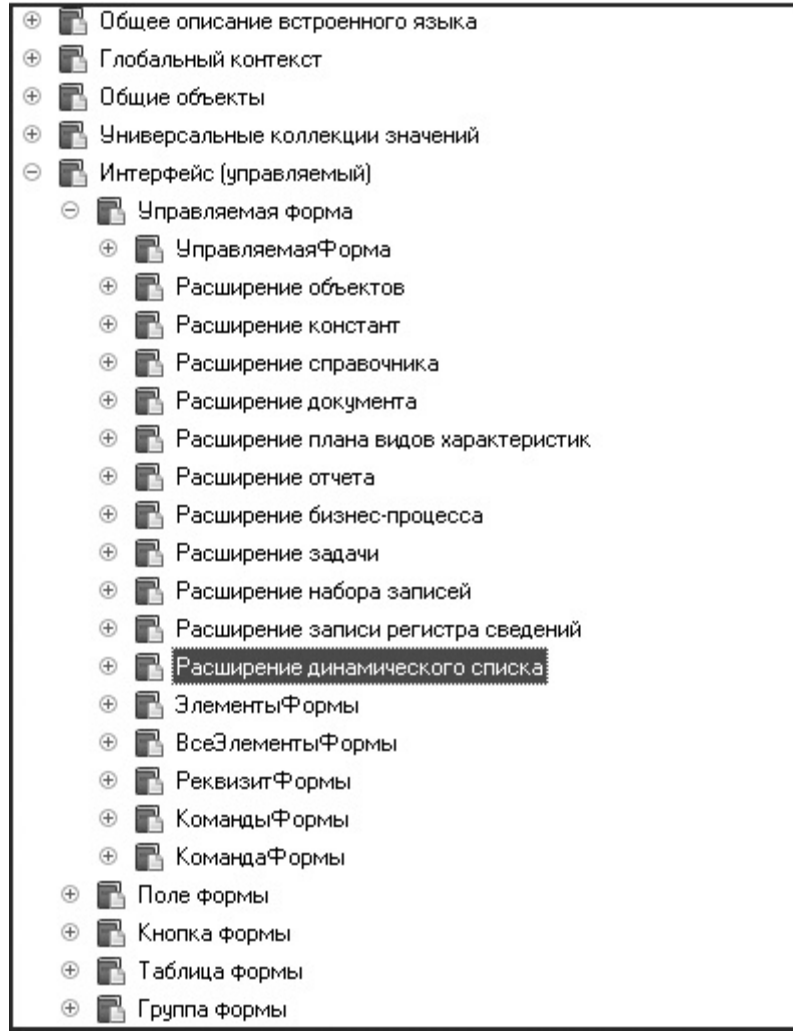


Рис. 5.15. Описание объектов в синтакс-помощнике

Таким образом, в модуле формы, где основной реквизит содержит данные документа (рис. 5.16), можно обратиться к свойству расширения управляемой формы для документа *АвтоВремя* (листинг 5.7).

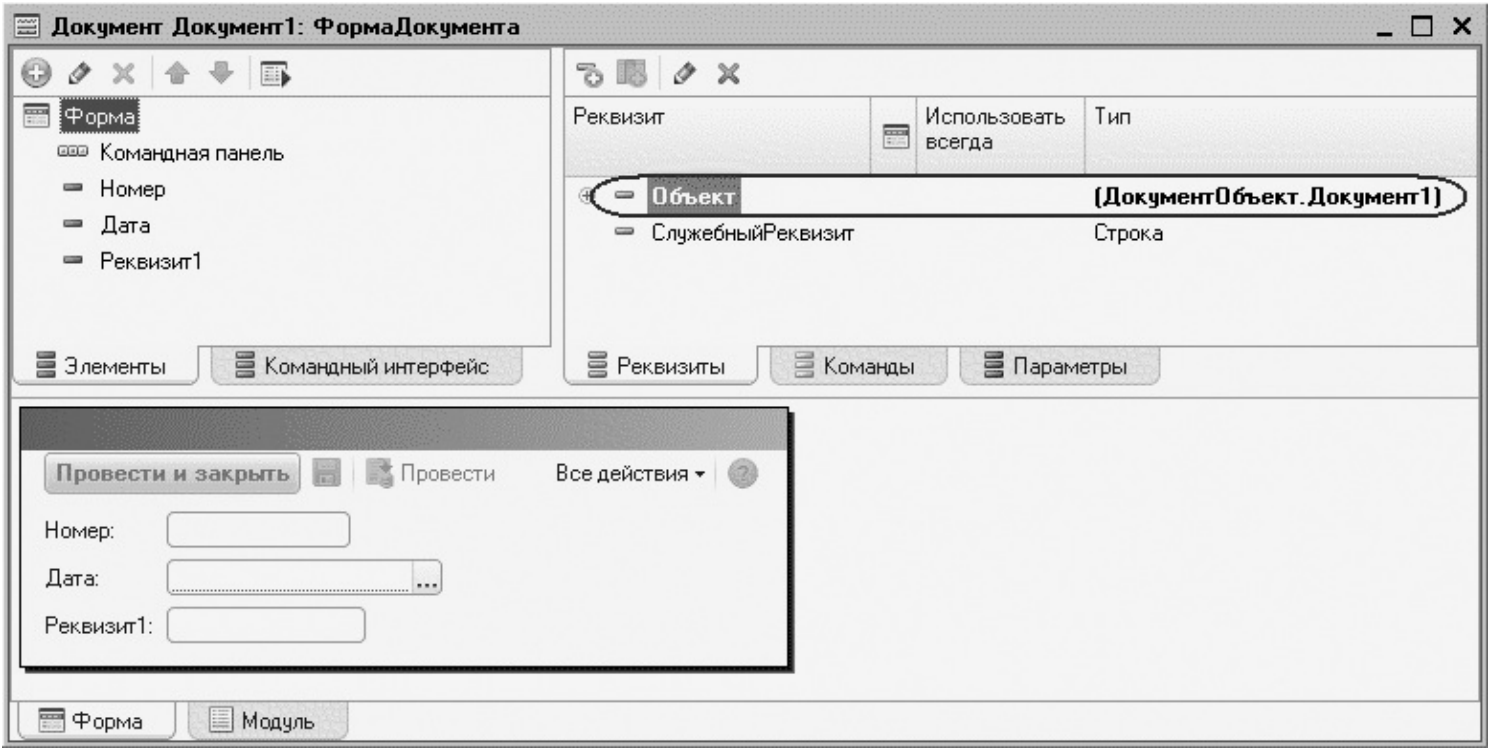


Рис. 5.16. Основной реквизит формы

Листинг 5.7. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  АвтоВремя = РежимАвтоВремя.Первым;

КонецПроцедуры
```

Или записать документ с помощью метода расширения управляемой формы для документа *Записать(),* листинг 5.8:

Листинг 5.8. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  Записать(РежимЗаписиДокумента.Проведение);

КонецПроцедуры
```

5. Глобальный контекст, в том числе неглобальные общие модули и экспортируемые функции и процедуры глобальных общих модулей.

В модуле формы можно получить системную дату, обратившись к встроенной функции *ТекущаяДата(),* листинг 5.9:

Листинг 5.9. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  Сообщить(ТекущаяДата());

КонецПроцедуры
```

Или получить историю работы пользователя, обратившись к свойству глобального контекста *ИсторияРаботыПользователя* (листинг 5.10):

Листинг 5.10. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  История = ИсторияРаботыПользователя.Получить();

КонецПроцедуры
```

Если в глобальном общем модуле (например, *ОбменДанными*) определена экспортная процедура *ПолучитьПрефиксНомера()* (листинг 5.11), то в модуле формы можно обращаться к ней по имени (листинг 5.12):

Листинг 5.11. Глобальный общий модуль

```
Функция ПолучитьПрефиксНомера() Экспорт

  Возврат Константы.ПрефиксНумерации.Получить();

КонецФункции
```

Листинг 5.12. Модуль формы

```
&НаКлиенте
Процедура Команда1(Префикс)

  Префикс = ПолучитьПрефиксНомера();

КонецПроцедуры
```

Если такой общий модуль – неглобальный (например, *РаботаСДокументами*), то при обращении к процедуре ее имя нужно указывать через точку от имени модуля (листинг 5.13):

Листинг 5.13. Модуль формы

```
&НаКлиенте
Процедура КоличествоПриИзменении(Элемент)

  РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры
```

Второй способ предпочтительнее, так как неглобальные общие модули компилируются по мере обращения к ним, а не при запуске системы, как глобальные.

Естественно, при этом нужно обеспечивать согласованность того, как описана процедура в модуле формы (*&НаКлиенте, &НаСервере* и т. д.), и того, какие флажки проставлены у общего модуля (*клиент (управляемое приложение), сервер* и т. д.).

6. Экспортируемые переменные, процедуры и функции модуля управляемого приложения.

Если в модуле приложения определена экспортная процедура *ТестовоеСообщение()* (листинг 5.14), то в модуле формы можно обращаться к ней по имени (листинг 5.15):

Листинг 5.14. Модуль приложения

```
Процедура ТестовоеСообщение() Экспорт

  Сообщить("Тестовое сообщение");

КонецПроцедуры
```

Листинг 5.15. Модуль формы

```
&НаКлиенте
Процедура Команда1()

  ТестовоеСообщение();

КонецПроцедуры
```

Форма как программный объект

Помимо того что форма внутри своего модуля предоставляет доступ к различным частям конфигурации, она также доступна из других частей конфигурации как программный объект.

При этом помимо стандартных свойств и методов объекта встроенного языка *УправляемаяФорма*, у нее могут существовать и другие свойства и методы, определенные разработчиком.

Например, если в модуле формы *ФормаДокумента* документа *ПриходнаяНакладная* описана экспортируемая процедура *МатериалыКоличествоПриИзменении()* (листинг 5.16), то может быть использован следующий вызов этой процедуры (листинг 5.17):

Листинг 5.16. Модуль формы

```
&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)

  СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
  РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры
```

Листинг 5.17. Модуль формы

```
Форма = ПолучитьФорму("Документ.ПриходнаяНакладная.Форма.ФормаДокумента");
Форма.МатериалыКоличествоПриИзменении();
```

Процедуры – обработчики событий в модуле формы

Помимо описания переменных и основной программы, модуль формы может содержать описание процедур-обработчиков событий, связанных с формой. Основными событиями, которые могут обрабатываться в модуле формы, являются события открытия и закрытия окна формы (листинг 5.18):

Листинг 5.18. Модуль формы

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

  // Вставить содержимое обработчика.
```

```
КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)

  // Вставить содержимое обработчика.

КонецПроцедуры
```

Важным моментом здесь является то, что имена этих процедур не фиксированы, они могут иметь произвольные имена. Поэтому недостаточно, например, написать в модуле определения процедур с именами *ПриСозданииНаСервере* или *ПриОткрытии*. Кроме этого, их обязательно нужно связать в конфигураторе с соответствующими событиями формы (рис. 5.17).

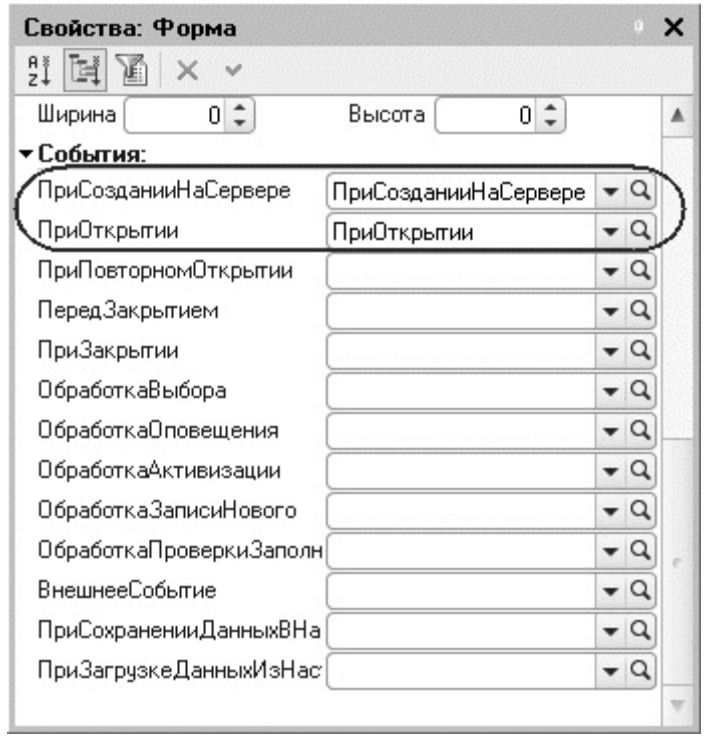


Рис. 5.17. События формы

Начинающие разработчики зачастую забывают об этом. Чтобы не попадать в такие ситуации, для автоматического создания таких процедур лучше использовать кнопки открытия  со значком лупы в панели свойств (рис. 5.18).

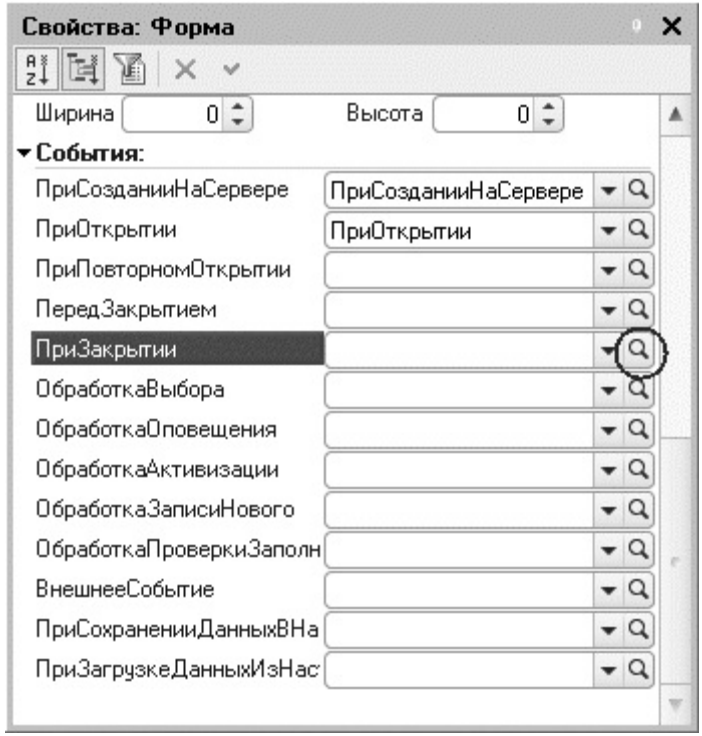


Рис. 5.18. События формы

Или можно выбрать их из выпадающего списка, доступного, когда вы находитесь в модуле формы (рис. 5.19).

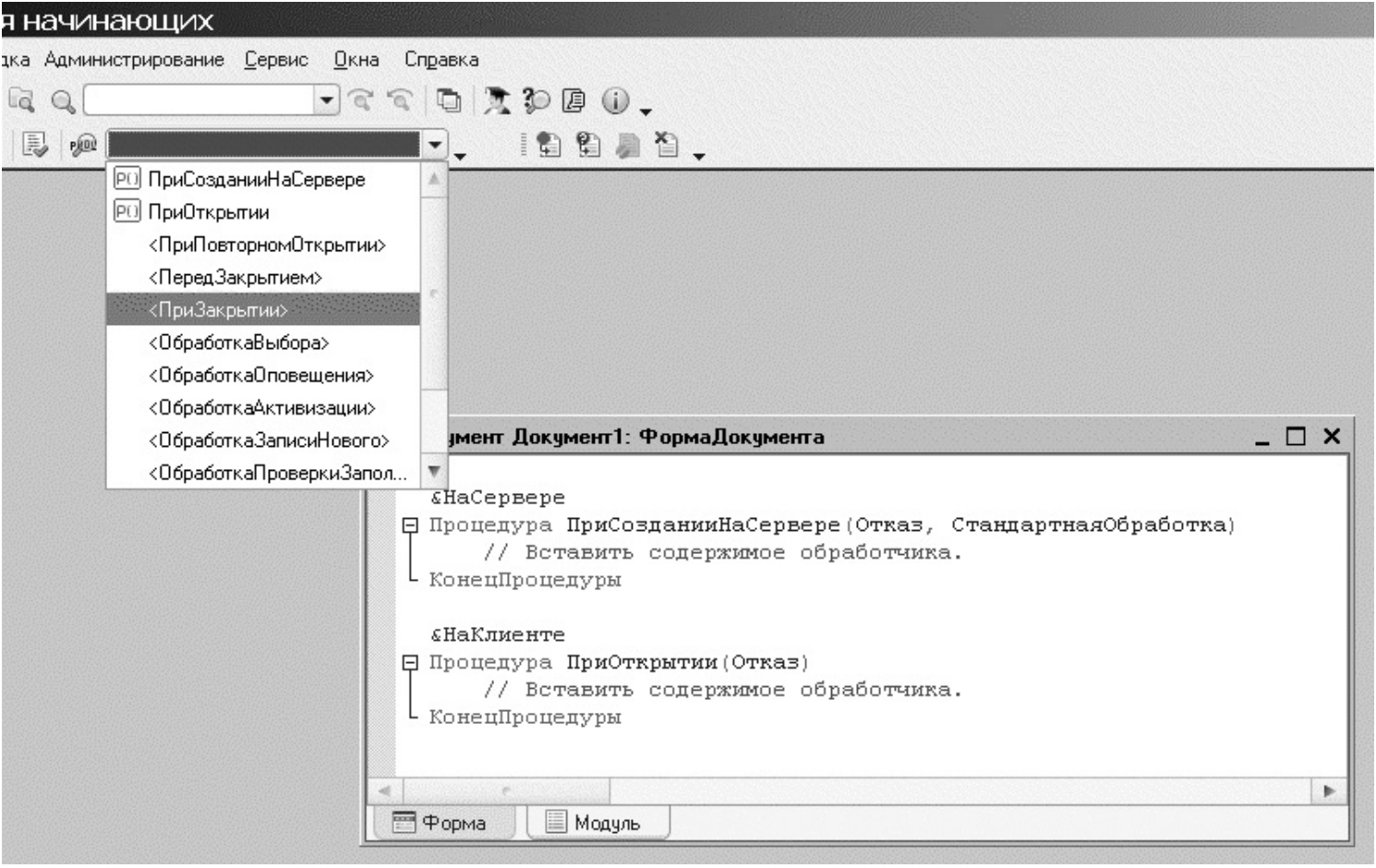


Рис. 5.19. Список событий формы

В этом случае система автоматически создаст определения этих процедур и свяжет их с соответствующими свойствами формы.

Как понять то, что написано в модуле формы

В заключение кратко опишем общий алгоритм, позволяющий разобраться с кодом, написанным в модуле формы. Рассматривать его будем на конкретном примере нашего обработчика события *МатериалыКоличествоПриИзменении*.

Допустим, в модуле формы нам встретилось выражение:
СтрокаТабличнойЧасти = ЭлементыФормы.Материалы.ТекущиеДанные.

Как понять, что такое *СтрокаТабличнойЧасти*? Нужно вспомнить, из чего состоит контекст формы:

- локальный контекст самого модуля формы;
- реквизиты формы, которой «принадлежит» модуль;
- свойства и методы объекта *УправляемаяФорма* встроенного языка;
- свойства и методы расширения формы, определяемого типом того объекта, данные которого содержатся в основном реквизите формы;
- глобальный контекст, в том числе неглобальные общие модули и экспортируемые функции и процедуры глобальных общих модулей;
- экспортируемые переменные, процедуры и функции модуля управляемого приложения.

Далее по порядку проверить:

1. Объявлена ли в модуле формы переменная *СтрокаТабличнойЧасти*? Нет.
2. Есть ли у формы реквизит *СтрокаТабличнойЧасти*? Нет.
3. Есть ли у объекта *УправляемаяФорма* свойство *СтрокаТабличнойЧасти*? Нет.
4. Есть ли у расширения формы свойство *СтрокаТабличнойЧасти*? Нет.
5. Есть ли свойство глобального контекста *СтрокаТабличнойЧасти*? Нет.
6. Есть ли в модуле управляемого приложения экспортная переменная *СтрокаТабличнойЧасти*? Нет.

Значит *СтрокаТабличнойЧасти* – это локальная переменная, определяемая непосредственно в этом операторе присваивания.

Как понять работу кода на встроенном языке

На предыдущем занятии мы писали код обработчика события *МатериалыКоличествоПриИзменении* (листинг 4.1) и кратко объясняли смысл написанного.

Теперь мы покажем два способа, как самому разобраться с множеством

незнакомых свойств и методов объектов конфигурации, чтобы в будущем самостоятельно изучать фрагменты кода или создавать свои собственные процедуры на встроенном языке.

Синтакс-помощник – инструмент, созданный для помощи разработчику, содержащий описание всех программных объектов, которые использует система, их методов, свойств, событий и пр.

Чтобы открыть синтакс-помощник, нужно нажать соответствующую кнопку  на панели инструментов конфигуратора или выполнить команду главного меню *Справка* > *Синтакс-помощник* (рис. 5.20).

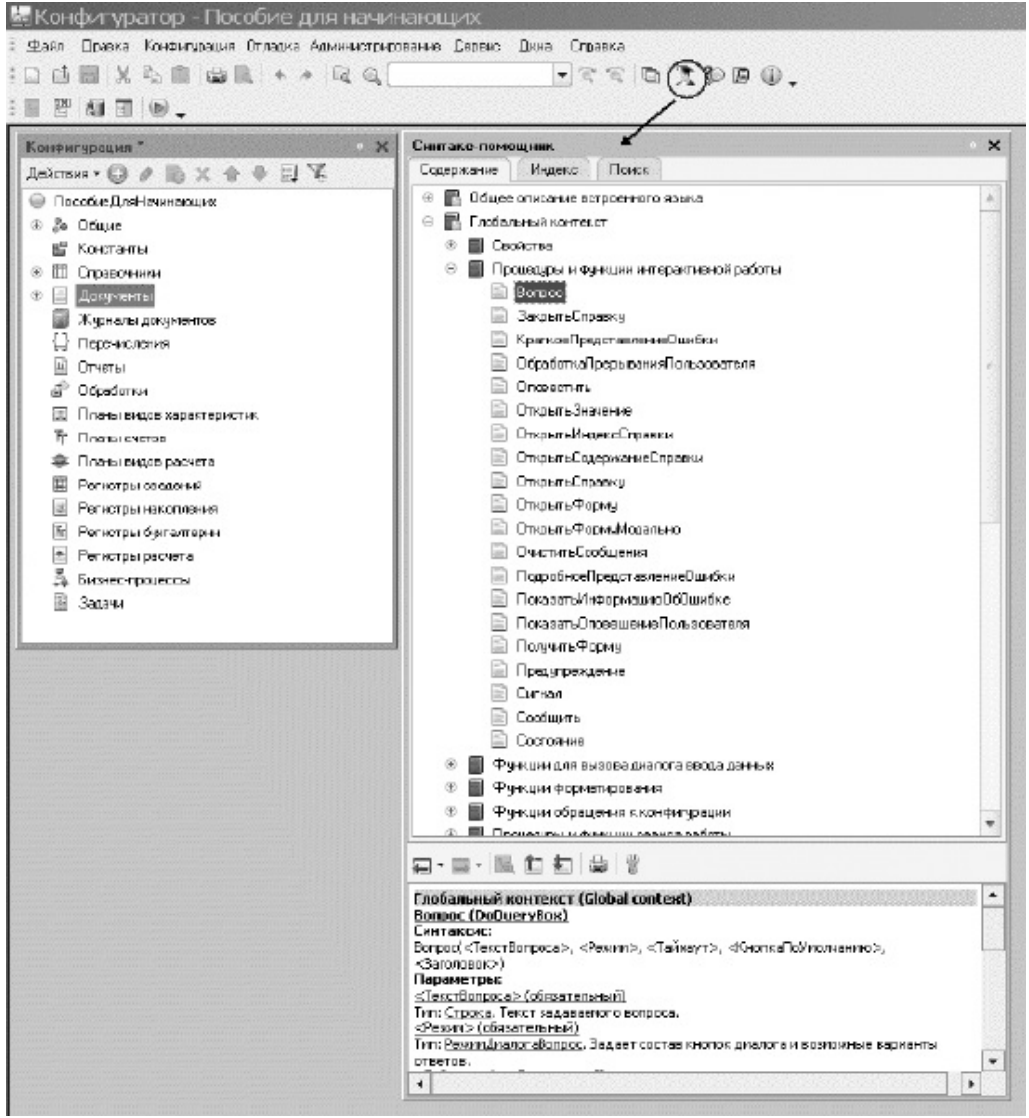

Конфигуратор - Пособие для начинающих
Файл Правка Конфигурация Отладка Администрирование Сервис Окна Справка
Конфигурация
Действия
ПособиеДляНачинающих
Общие
Константы
Справочники
Документы
Журналы документов
Перечисления
Отчеты
Обработки
Планы видов характеристик
Планы счетов
Планы видов расчета
Регистры сведений
Регистры накопления
Регистры бухгалтерии
Регистры расчета
Бизнес-процессы
Задачи
Синтакс-помощник
Содержание
Индекс
Поиск
Общее описание встроенного языка
Глобальный контекст
Свойства
Процедуры и функции интерактивной работы
Вопрос
ЗакрытьСправку
КраткоеПредставлениеОшибки
ОбработкаПрерыванияПользователя
Оповестить
ОткрытьЗначение
ОткрытьИндексСправки
ОткрытьСодержаниеСправки
ОткрытьСправку
ОткрытьФорму
ОткрытьФормуМодально
ОчиститьСообщения
ПодробноеПредставлениеОшибки
ПоказатьИнформациюОбОшибке
ПоказатьОповещениеПользователя
ПолучитьФорму
Предупреждение
Сигнал
Сообщить
Состояние
Функции для вызова диалога ввода данных
Функции форматирования
Функции обращения к конфигурации
Глобальный контекст (Global context)
Вопрос (DoQueryBox)
Синтаксис:
Вопрос(<ТекстВопроса>, <Режим>, <Таймаут>, <КнопкаПоУмолчанию>, <Заголовок>)
Параметры:
<ТекстВопроса> (обязательный)
Тип: Строка. Текст задаваемого вопроса.
<Режим> (обязательный)
Тип: РежимДиалогаВопрос. Задает состав кнопок диалога и возможные варианты ответов.

Рис. 5.20. Вызов синтакс-помощника

Как и любая другая справочная система, он представляет собой древовидную структуру, состоящую из глав, разделов, подразделов и т. п. Содержание синтакс-помощника полностью дублирует описание встроенного языка в семи томах, входящих в стандартный комплект поставки «1С:Предприятия». Однако пользоваться синтакс-помощником, на наш взгляд, удобнее, так как он находится сразу под рукой и имеет возможность контекстной помощи (*Ctrl + F1*).

Анализ кода с помощью синтакс-помощника

Пользоваться синтакс-помощником удобно в тех случаях, когда нужно разобраться в уже написанном незнакомом коде. На примере нашего обработчика события *МатериалыКоличествоПриИзменении* (см. листинг 4.1) продемонстрируем, как понять код обработчика, используя синтакс-помощник.

Первый способ

Первый способ – найти нужный раздел в содержании и спускаться вниз по дереву, раскрывая нужные подразделы, свойства, ссылки и т. п.

Итак, перед нами первая строка нашего обработчика (листинг 5.19).

Листинг 5.19. Процедура «МатериалыКоличествоПриИзменении» (первая строка)

```
СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
```

Чтобы понять этот код, нужно прежде всего понимать, в каком контексте он исполняется. Программный контекст зависит от того, в каком модуле располагается код. В данном случае процедура обработчика находится в модуле формы, следовательно, мы находимся в контексте модуля формы.

Будем изучать нашу строку последовательно слева направо. Что такое *СтрокаТабличнойЧасти*? Слева от оператора присваивания (=) находится либо какое-то свойство, доступное нам непосредственно в этом контексте, либо переменная.

Согласно алгоритму, изложенному в предыдущем разделе, мы должны проверить:

- Объявлена ли в модуле формы переменная *СтрокаТабличнойЧасти*? Откроем модуль формы (о том, как это сделать рассказано здесь). Мы не видим здесь строки описания переменной (*Перем СтрокаТабличнойЧасти;*), значит это не переменная модуля формы.
- Есть ли у формы реквизит *СтрокаТабличнойЧасти*? Откроем форму документа *ПриходнаяНакладная* и перейдем в окно реквизитов формы, расположенное справа вверху редактора форм (рис. 5.21).

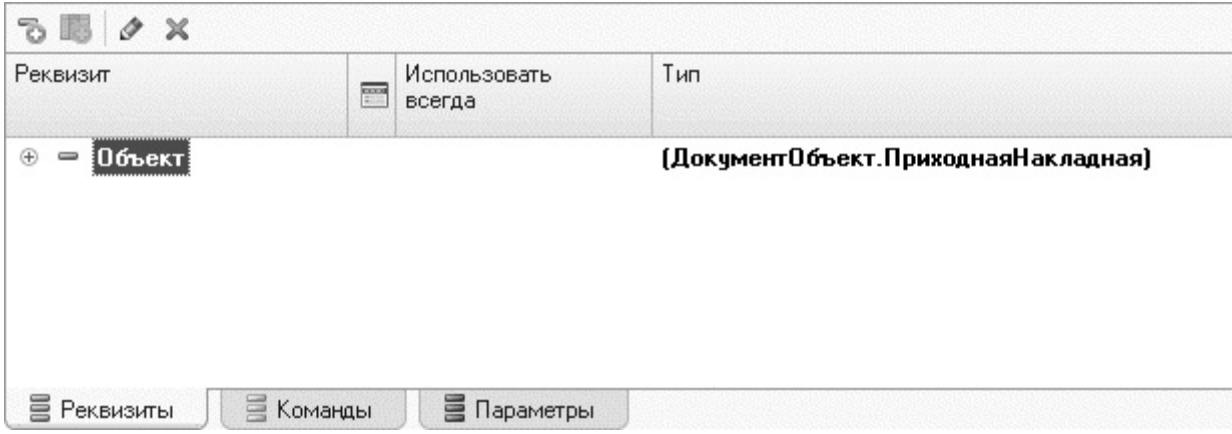


Рис. 5.21. Список реквизитов формы документа «Приходная накладная»

Мы видим, что у формы один основной (он выделен жирным шрифтом) реквизит *Объект*. Значит, реквизита *СтрокаТабличнойЧасти* у формы нет.

- Есть ли у объекта *УправляемаяФорма* свойство *СтрокаТабличнойЧасти*? Посмотрим в синтакс-помощнике свойства управляемой формы. Откроем синтакс-помощник на закладке *Содержание*. Управляемая форма – это объект интерфейса управляемого приложения, поэтому раскроем раздел *Интерфейс (управляемый)* > *Управляемая форма*. Затем раскроем объект *УправляемаяФорма* и его *Свойства* (рис. 5.22).

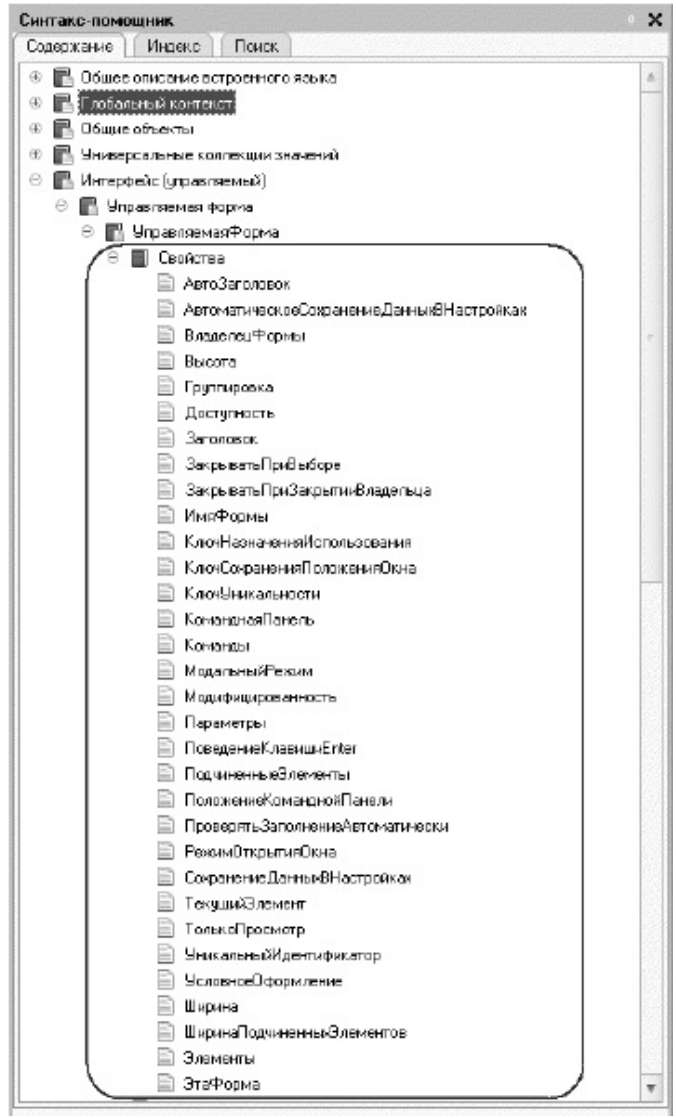

Синтакс-помощник
Содержание
Индекс
Поиск
Общее описание встроенного языка
Глобальный контекст
Общие объекты
Универсальные коллекции значений
Интерфейс (управляемый)
Управляемая форма
УправляемаяФорма
Свойства
АвтоЗаголовок
АвтоматическоеСохранениеДанныхВНастройках
ВладелецФормы
Высота
Группировка
Доступность
Заголовок
ЗакрыватьПриВыборе
ЗакрыватьПриЗакрытииВладельца
ИмяФормы
КлючНазначенияИспользования
КлючСохраненияПоложенияОкна
КлючУникальности
КоманднаяПанель
Команды
МодальныйРежим
Модифицированность
Параметры
ПоведениеКлавишиEnter
ПодчиненныеЭлементы
ПоложениеКоманднойПанели
ПроверятьЗаполнениеАвтоматически
РежимОткрытияОкна
СохранениеДанныхВНастройках
ТекущийЭлемент
ТолькоПросмотр
УникальныйИдентификатор
УсловноеОформление
Ширина
ШиринаПодчиненныхЭлементов
Элементы
ЭтаФорма

Рис. 5.22. Список свойств объекта «УправляемаяФорма» в синтакс-помощнике

Свойства расположены в алфавитном порядке. Мы видим, что среди них нет свойства *СтрокаТабличнойЧасти*.

- Есть ли у расширения формы свойство *СтрокаТабличнойЧасти*? Мы знаем, что основной реквизит формы содержит данные объекта *ДокументОбъект.ПриходнаяНакладная* (см. рис. 5.21). Следовательно, в модуле формы становятся доступны свойства, методы объекта встроенного языка *Расширение управляемой формы для документа* (синтакс-помощник – *Интерфейс (управляемый)* > *Управляемая форма* > *Расширение документа*). Посмотрим на них (рис. 5.23).

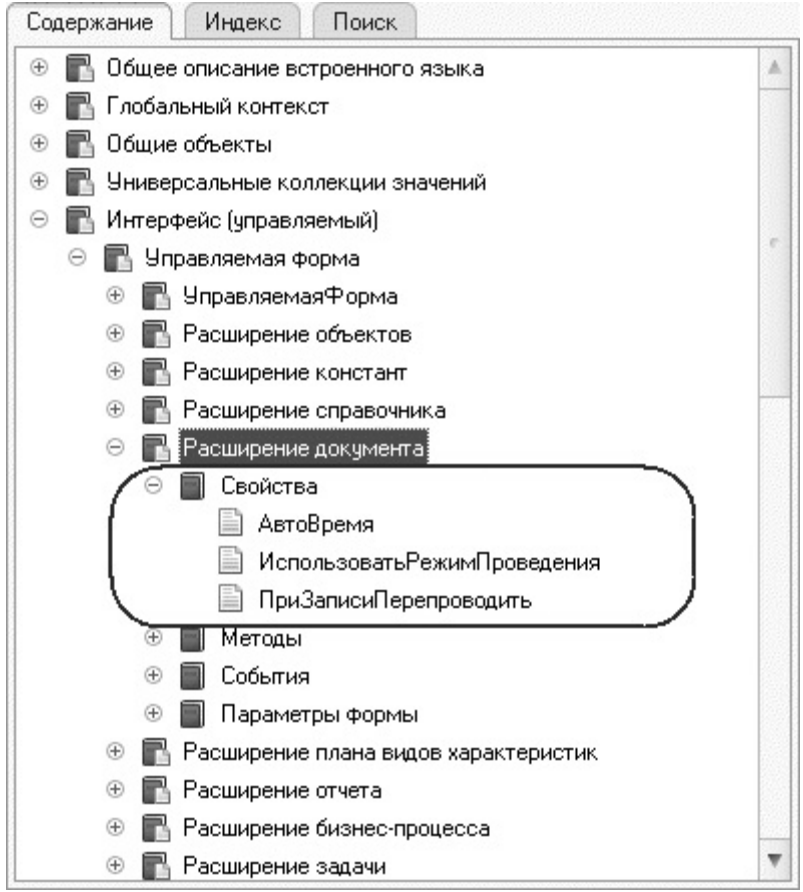


Рис. 5.23. Список свойств объекта «Расширение» управляемой формы для документа в синтакс-помощнике

Мы видим, что среди них нет свойства *СтрокаТабличнойЧасти*.

- Есть ли свойство глобального контекста *СтрокаТабличнойЧасти*? Откроем в синтакс-помощнике свойства глобального контекста (рис. 5.24).

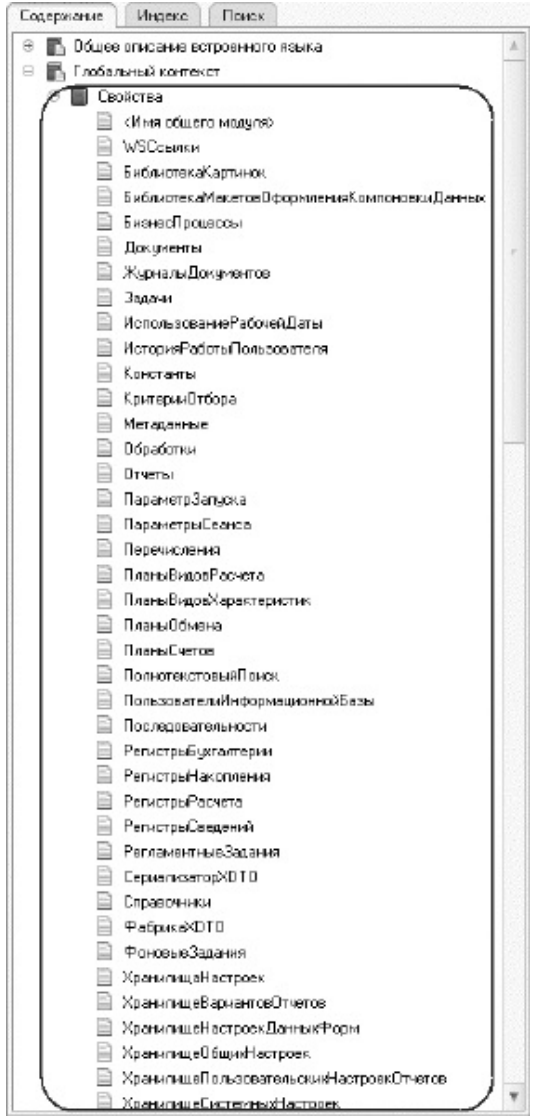

Содержание
Индекс
Поиск
Общее описание встроенного языка
Глобальный контекст
Свойства
<Имя общего модуля>
WSСсылки
БиблиотекаКартинок
БиблиотекаМакетовОформленияКомпоновкиДанных
БизнесПроцессы
Документы
ЖурналыДокументов
Задачи
ИспользованиеРабочейДаты
ИсторияРаботыПользователя
Константы
КритерииОтбора
Метаданные
Обработки
Отчеты
ПараметрЗапуска
ПараметрыСеанса
Перечисления
ПланыВидовРасчета
ПланыВидовХарактеристик
ПланыОбмена
ПланыСчетов
ПолнотекстовыйПоиск
ПользователиИнформационнойБазы
Последовательности
РегистрыБухгалтерии
РегистрыНакопления
РегистрыРасчета
РегистрыСведений
РегламентныеЗадания
СериализаторXDTO
Справочники
ФабрикаXDTO
ФоновыеЗадания
ХранилищеНастроек
ХранилищеВариантовОтчетов
ХранилищеНастроекДанныхФорм
ХранилищеОбщихНастроек
ХранилищеПользовательскихНастроекОтчетов
ХранилищеСистемныхНастроек

Рис. 5.24. Список свойств глобального контекста в синтакс-помощнике

Мы видим, что среди них нет свойства *СтрокаТабличнойЧасти*. Выражение *СтрокаТабличнойЧасти* также не может быть именем неглобального общего модуля, так как к его процедурам следует обращаться через точку (*СтрокаТабличнойЧасти.*). Также это выражение не может быть экспортируемой процедурой глобального общего модуля, так как в этом случае мы могли бы только вызвать эту процедуру как *СтрокаТабличнойЧасти ()*, а не присваивать ей что-то.

- Есть ли в модуле управляемого приложения экспортная переменная *СтрокаТабличнойЧасти*? Откроем модуль управляемого приложения (о том, как это сделать рассказано здесь). Мы не видим здесь строки описания переменных (*Перем СтрокаТабличнойЧасти Экспорт;*), значит, это не переменная модуля управляемого приложения.

Таким образом, понятно, что выражение *СтрокаТабличнойЧасти* – это локальная переменная процедуры *МатериалыКоличествоПриИзменении*. В процессе выполнения программы ей присваивается какое-то значение. Переменные во встроенном языке не типизированные, поэтому в любой момент ей можно присвоить значение любого типа. Если переменная локальная, то есть используется только в контексте данной процедуры, то не требуется и ее явного объявления. Она объявляется в момент первого ее использования.

Справа от оператора присваивания находится выражение *Элементы.Материалы.ТекущиеДанные*. Чтобы понять, что такое *Элементы*, пройдемся еще раз по тому же алгоритму, что и в случае с локальной переменной *СтрокаТабличнойЧасти*.

- Объявлена ли в модуле формы переменная *Элементы*? Нет.
- Есть ли у формы реквизит *Элементы*? Нет.
- Есть ли у объекта *УправляемаяФорма* свойство *Элементы*? Посмотрим еще раз на список свойств объекта *УправляемаяФорма*. Найдем в нем строку *Элементы*, значит – это одно из свойств управляемой формы. Чтобы узнать, что это такое, дважды щелкнем мышью на этой строке (рис. 5.25).

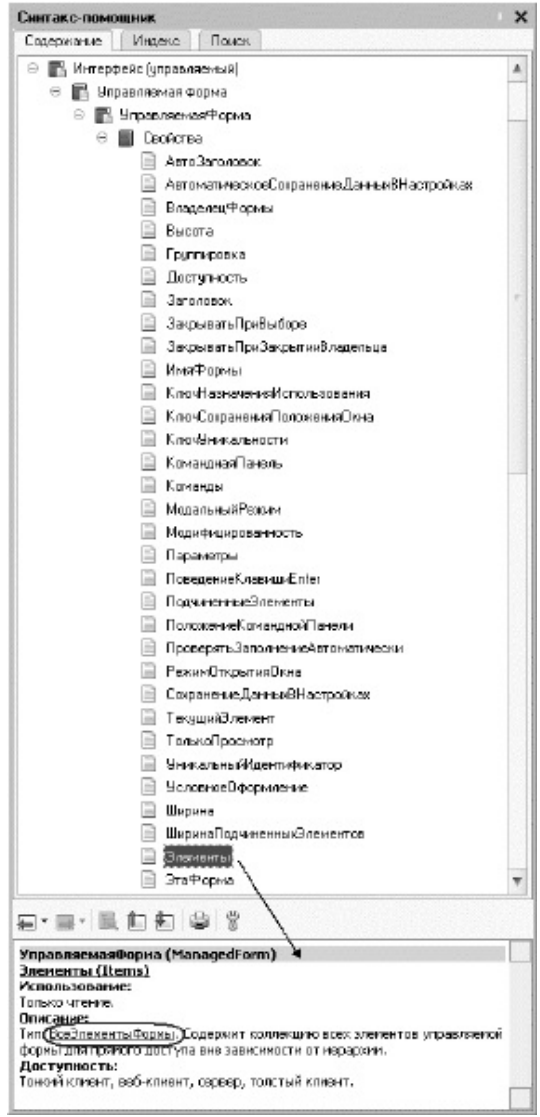

Синтакс-помощник
Содержание
Индекс
Поиск
Интерфейс (управляемый)
Управляемая форма
УправляемаяФорма
Свойства
АвтоЗаголовок
АвтоматическоеСохранениеДанныхВНастройках
ВладелецФормы
Высота
Группировка
Доступность
Заголовок
ЗакрыватьПриВыборе
ЗакрыватьПриЗакрытииВладельца
ИмяФормы
КлючНазначенияИспользования
КлючСохраненияПоложенияОкна
КлючУникальности
КоманднаяПанель
Команды
МодальныйРежим
Модифицированность
Параметры
ПоведениеКлавиши Enter
ПодчиненныеЭлементы
ПоложениеКоманднойПанели
ПроверятьЗаполнениеАвтоматически
РежимОткрытияОкна
СохранениеДанныхВНастройках
ТекущийЭлемент
ТолькоПросмотр
УникальныйИдентификатор
УсловноеОформление
Ширина
ШиринаПодчиненныхЭлементов
Элементы
ЭтаФорма
УправляемаяФорма (ManagedForm)
Элементы (Items)
Использование:
Только чтение.
Описание:
Тип: ВсеЭлементыФормы. Содержит коллекцию всех элементов управляемой формы для прямого доступа вне зависимости от иерархии.
Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент.

Рис. 5.25. Описание свойства «Элементы» объекта «УправляемаяФорма» в синтакс-помощнике

В нижней части окна синтакс-помощника появится описание выделенного нами свойства. Из этого описания следует, что, используя свойство *Элементы*, мы получаем объект *ВсеЭлементыФормы*, который содержит коллекцию всех элементов формы. Чтобы узнать, что это такое, нажмем на соответствующую ссылку. Откроется описание коллекции *ВсеЭлементыФормы* (рис. 5.26).

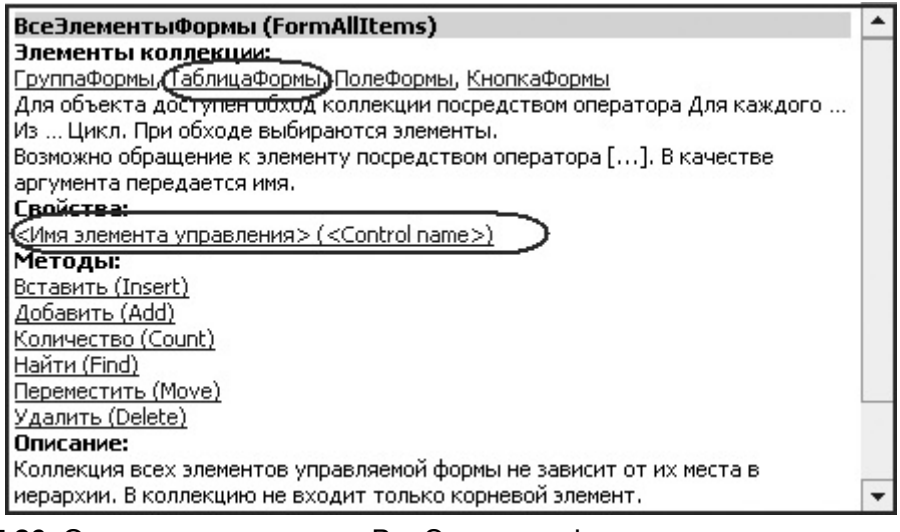


Рис. 5.26. Описание коллекции «ВсеЭлементыФормы» в синтакс-помощнике

ПРИМЕЧАНИЕ

При этом дерево синтакс-помощника, отображаемое вверху, не изменилось. Если вы хотите найти, в какой ветке дерева находится открытое сейчас описание,

нужно нажать кнопку ***Найти текущий элемент в дереве***  *, находящуюся над окном описания объектов синтакс-помощника.*

Эта коллекция содержит элементы управляемой формы, размещенные на форме. Доступ к элементу осуществляется по имени.

Итак, что такое *Элементы,* мы знаем. Дальше через точку от этого объекта у нас написано: *Элементы.Материалы*. У коллекции есть такое свойство – *<имя элемента управления>. Материалы* – это имя некоторого элемента формы. Посмотрим на структуру элементов формы. Откроем форму документа *ПриходнаяНакладная* и перейдем в окно элементов формы, расположенное слева вверху редактора форм (рис. 5.27).

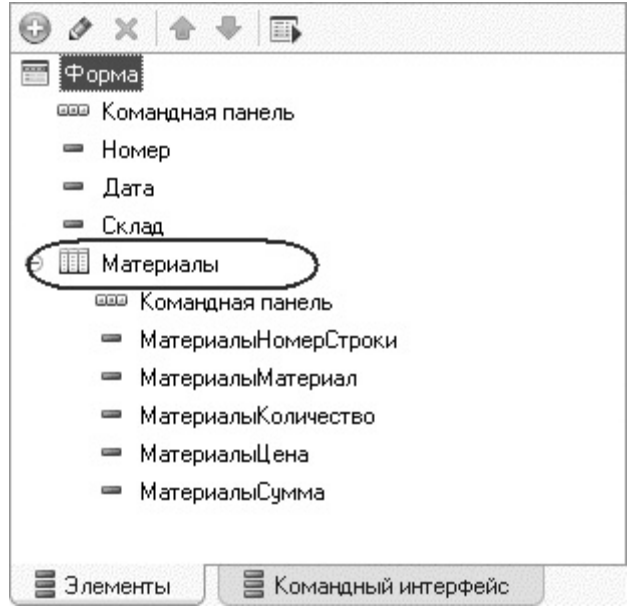


Рис. 5.27. Структура элементов формы документа «Приходная накладная»

В структуре элементов формы мы видим таблицу – *Материалы*. Открыв палитру свойств этой таблицы, в заголовке мы видим – *Свойства: Таблица* (рис. 5.28).

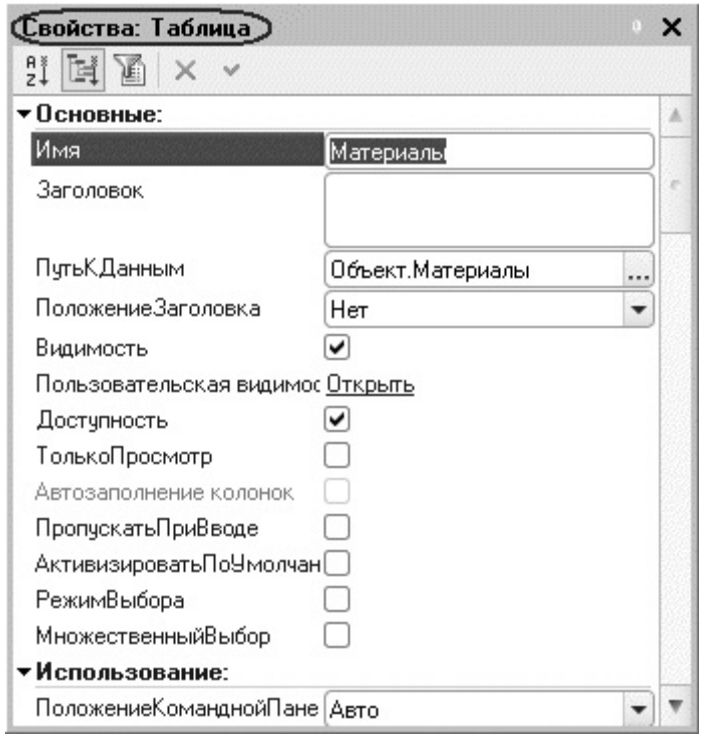


Рис. 5.28. Палитра свойств элемента формы – таблица

Значит, этот элемент формы является таблицей, и, следовательно, нам нужен объект коллекции *ТаблицаФормы*. Чтобы узнать, что это такое, нажмем на соответствующую ссылку (см. рис. 5.26). Мы увидим список свойств объекта *ТаблицаФормы*. Свойства расположены в алфавитном порядке (рис. 5.29).


ТаблицаФормы (FormTable)
Свойства:
АвтоВводНезаполненного (AutoAddIncomplete)
АвтоВводНовойСтроки (AutoInsertNewRow)
АвтоОтметкаНезаполненного (AutoMarkIncomplete)
АктивизироватьПоУмолчанию (DefaultItem)
ВертикальнаяПолосаПрокрутки (VerticalScrollBar)
ВертикальныеЛинии (VerticalLines)
Видимость (Visible)
Вывод (Output)
ВыделенныеСтроки (SelectedRows)
Высота (Height)
ВысотаВСтрокахТаблицы (HeightInTableRows)
ВысотаЗаголовка (TitleHeight)
ВысотаПодвала (FooterHeight)
ВысотаШапки (HeaderHeight)
ГоризонтальнаяПолосаПрокрутки (HorizontalScrollBar)
ГоризонтальныеЛинии (HorizontalLines)
Доступность (Enabled)
Заголовок (Title)
ИзменятьПорядокСтрок (ChangeRowOrder)
ИзменятьСоставСтрок (ChangeRowSet)
Имя (Name)
КартинкаСтрок (RowsPicture)
КоманднаяПанель (CommandBar)
КонтекстноеМеню (ContextMenu)
МножественныйВыбор (MultipleChoice)
НачальноеОтображениеДерева (InitialTreeView)
НачальноеОтображениеСписка (InitialListView)
ОтметкаНезаполненного (MarkIncomplete)
Отображение (Representation)
Подвал (Footer)
Подсказка (ToolTip)
ПодчиненныеЭлементы (ChildItems)
ПоложениеЗаголовка (TitleLocation)
ПоложениеКоманднойПанели (CommandBarLocation)
ПропускатьПриВводе (SkipOnInput)
ПутьКДанным (DataPath)
ПутьКДаннымКартинкиСтроки (RowPictureDataPath)
РазрешитьНачалоПеретаскивания (EnableStartDrag)
РазрешитьПеретаскивание (EnableDrag)
РастягиватьПоВертикали (VerticalStretch)
РастягиватьПоГоризонтали (HorizontalStretch)
РежимВводаСтрок (RowInputMode)
РежимВыбора (ChoiceMode)
РежимВыделения (SelectionMode)
РежимВыделенияСтроки (RowSelectionMode)
Родитель (Parent)
СочетаниеКлавиш (Shortcut)
ТекущаяСтрока (CurrentRow)
ТекущиеДанные (CurrentData)
ТекущийРодитель (CurrentParent)
ТекущийЭлемент (CurrentItem)
ТолькоПросмотр (ReadOnly)
ФиксацияСлева (FixedLeft)
ФиксацияСправа (FixedRight)
ЦветРамки (BorderColor)
ЦветТекста (TextColor)
ЦветТекстаЗаголовка (TitleTextColor)
ЦветФона (BackColor)
ЧередованиеЦветовСтрок (UseAlternationRowColor)
Шапка (Header)
Ширина (Width)
Шрифт (Font)
ШрифтЗаголовка (TitleFont)

Рис. 5.29. Список свойств объекта «ТаблицаФормы» в синтакс-помощнике

Итак, что такое *Элементы.Материалы,* мы знаем. Дальше через точку от этого объекта у нас написано: *Элементы.Материалы.ТекущиеДанные*. Прокрутив список свойств таблицы управляемой формы вниз, мы видим свойство *ТекущиеДанные*. Значит, это одно из свойств объекта *ТаблицаФормы*. Чтобы узнать, что это такое, дважды щелкнем мышью на этой строке (рис. 5.30)

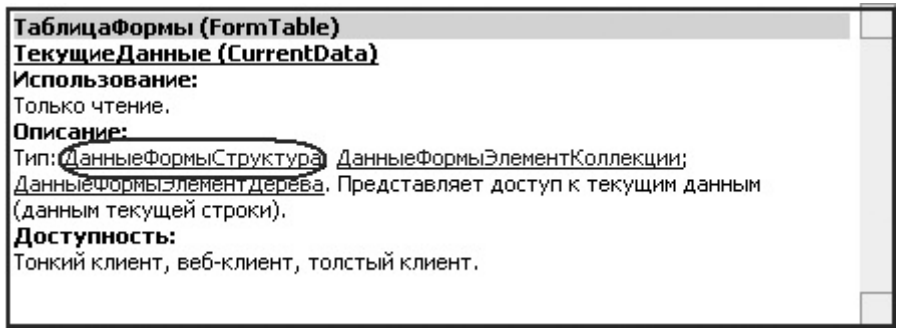


Рис. 5.30. Описание свойства «Текущие Данные» объекта «ТаблицаФормы» в синтакс-помощнике

В нижней части окна синтакс-помощника появится описание выделенного нами свойства. Из этого описания следует, что, используя свойство *ТекущиеДанные*, мы получаем объект *ДанныеФормыСтруктура*, который содержит данные, находящиеся в текущей строке таблицы.

Значит, в результате выполнения первой строки обработчика

СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные. В переменной *СтрокаТабличнойЧасти* у нас окажется объект типа *ДанныеФормыСтруктура*.

Теперь посмотрим на следующую строку обработчика (листинг 5.20).

Листинг 5.20. Процедура «МатериалыКоличествоПриИзменении» (вторая строка)

```
СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество *
СтрокаТабличнойЧасти.Цена;
```

Логично предположить, что *Сумма, Количество* и *Цена* – это какие-то свойства объекта *ДанныеФормыСтруктура*, который после выполнения первой строки находится в переменной *СтрокаТабличнойЧасти*. Чтобы узнать, что это такое, нажмем на соответствующую ссылку (см. рис. 5.30, 5.31).

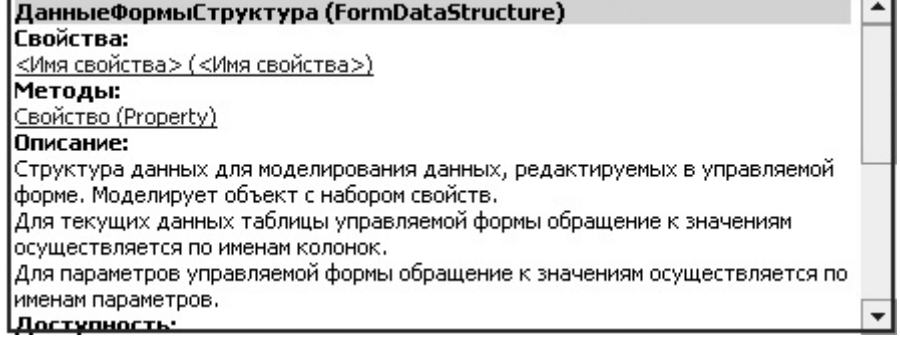


Рис. 5.31. Описание объекта «ДанныеФормыСтруктура» в синтакс-помощнике

Из описания этого объекта следует, что, используя объект *ДанныеФормыСтруктура*, мы можем обратиться к данным конкретной колонки табличной части, указав имя колонки в качестве свойства объекта. То есть, используя выражение *СтрокаТабличнойЧасти.Сумма,* мы обращаемся к данным, которые находятся в колонке *Сумма* текущей строки таблицы. И в них помещается произведение данных, содержащихся в колонке *Количество*, на данные в колонке *Цена*.

Второй способ

Второй способ – воспользоваться контекстной помощью синтакс-помощника. Для этого нужно открыть программный модуль, установить курсор на интересующую вас конструкцию встроенного языка и нажать одновременно клавиши *Ctrl* и *F1*. Откроем форму документа *ПриходнаяНакладная*, перейдем на закладку *Модуль*, откроем текст процедуры *МатериалыКоличествоПриИзменении*.

Немного выше мы установили, что выражение *СтрокаТабличнойЧасти*, слева от оператора присваивания – это локальная переменная.

Справа от оператора присваивания находится выражение *Элементы.Материалы.ТекущиеДанные*. Установим курсор на выражении

Элементы и нажмем *Ctrl + F1*. Синтакс-помощник откроется на закладке *Индекс*, и выражение *Элементы* будет помещено в строку поиска (рис. 5.32). Среди конструкций встроенного языка, отсортированных по алфавиту, будет произведен поиск этого выражения.

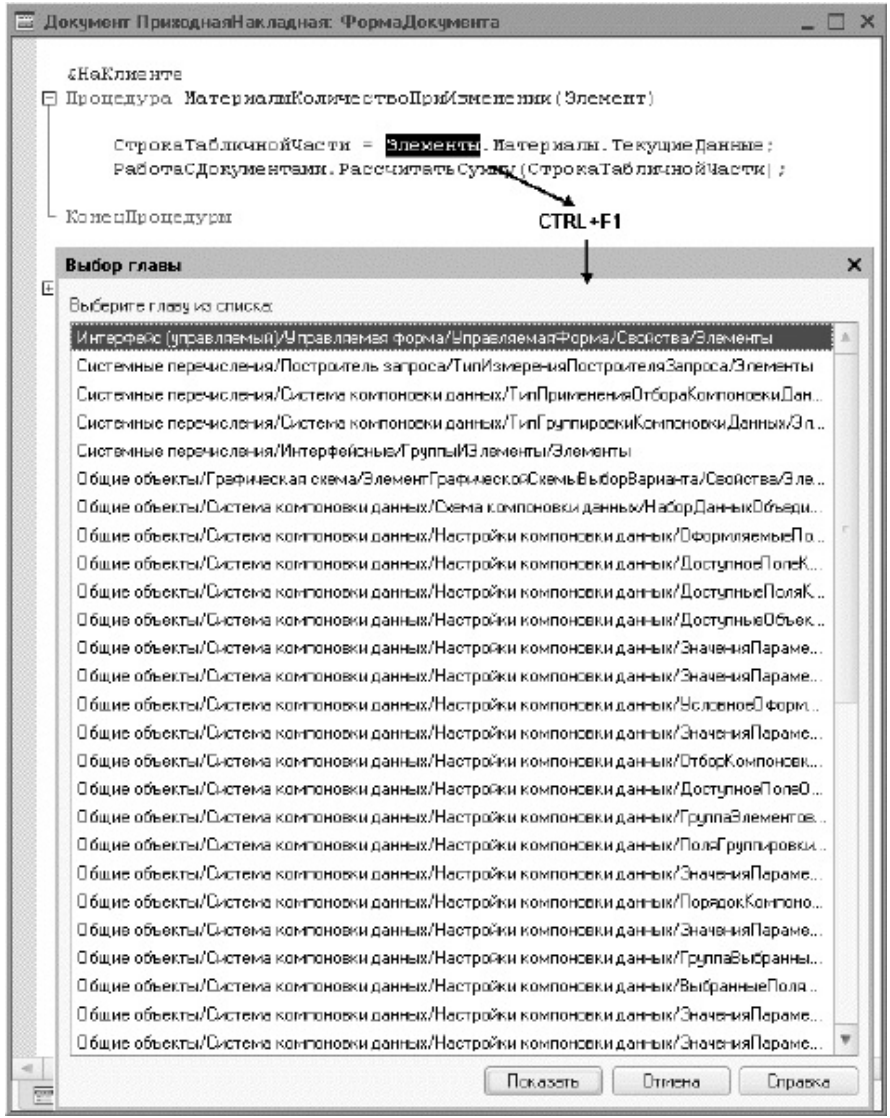

Документ ПриходнаяНакладная: ФормаДокумента
&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)
СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
КонецПроцедуры
CTRL+F1
Выбор главы
Выберите главу из списка:
Интерфейс (управляемый)/Управляемая форма/УправляемаяФорма/Свойства/Элементы
Системные перечисления/Построитель запроса/ТипИзмеренияПостроителяЗапроса/Элементы
Системные перечисления/Система компоновки данных/ТипПримененияОтбораКомпоновкиДан...
Системные перечисления/Система компоновки данных/ТипГруппировкиКомпоновкиДанных/Эл...
Системные перечисления/Интерфейсные/ГруппыИЭлементы/Элементы
Общие объекты/Графическая схема/ЭлементГрафическойСхемыВыборВарианта/Свойства/Эле...
Общие объекты/Система компоновки данных/Схема компоновки данных/НаборДанныхОбъеди...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ОформляемыеПо...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ДоступноеПолеК...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ДоступныеПоляК...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ДоступныеОбъек...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/УсловноеОформ...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ОтборКомпоновк...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ДоступноеПолеО...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ГруппаЭлементов...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ПоляГруппировки...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ПорядокКомпоно...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ГруппаВыбранны...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ВыбранныеПола...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Показать
Отмена
Справка

Рис. 5.32. Вызов контекстной помощи в синтакс-помощнике

Мы видим, что выражение найдено, так как появилось окно со списком глав, в которых это выражение используется. Каждая глава в списке содержит полный путь к искомому выражению, начиная от корня структуры содержания синтакс-помощника. Уровни этой структуры отделены друг от друга символом «/» (слеш). Таким образом, подведя мышь к какому-либо элементу списка глав, мы сможем оценить, та ли это глава, которая нам нужна.

Однако список глав, в которых используется искомое выражение, достаточно большой, и вручную искать нужную главу довольно утомительно. Чтобы найти свойство *Элементы*, которое используется в управляемой форме, нажмем *Ctrl + F*, находясь в окне списка, и введем в окно поиска выражение «УправляемаяФорма». Нажмем кнопку *Искать*. После этого в списке глав будет выделена нужная глава (рис. 5.33).

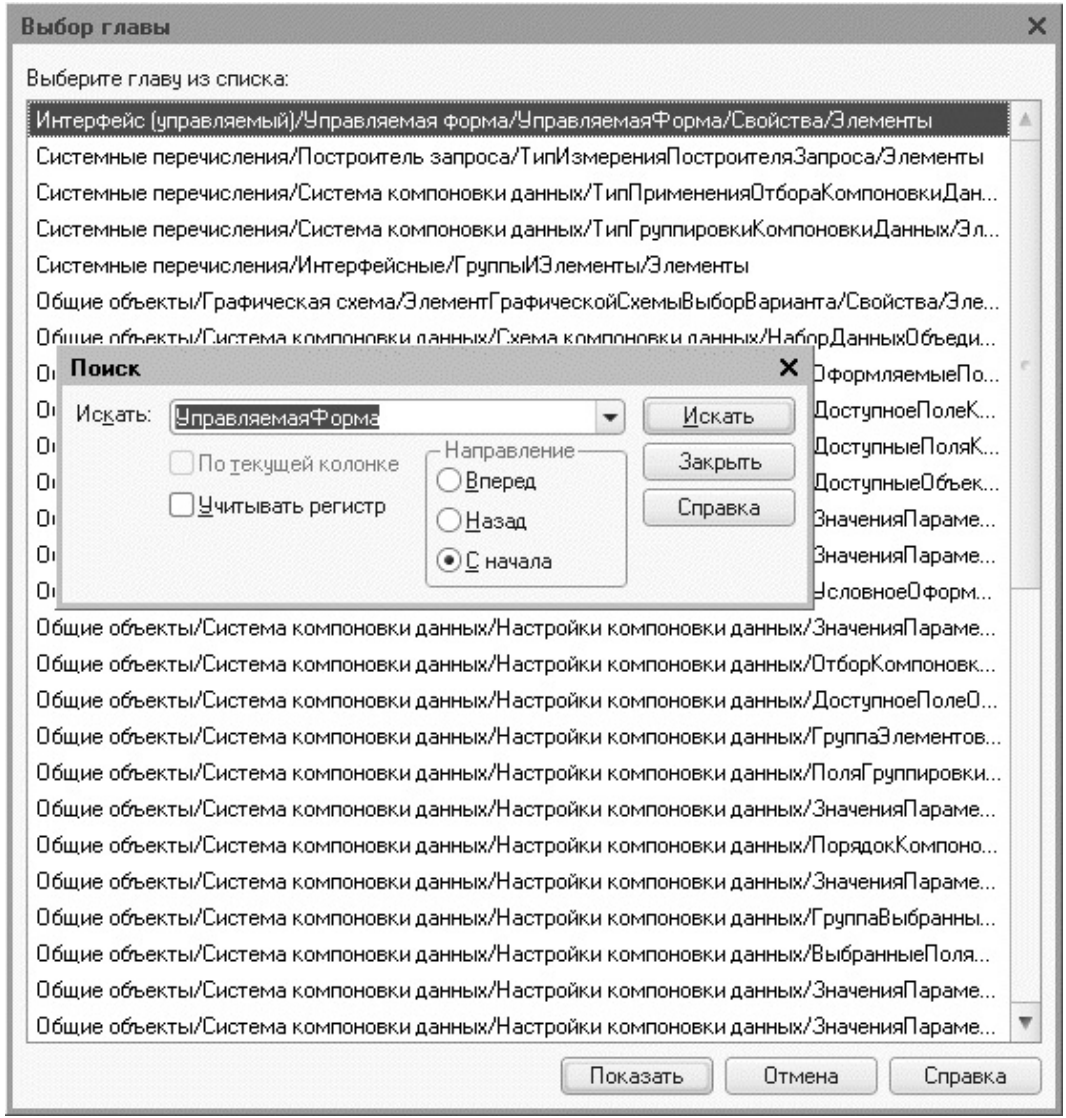

Выбор главы
Выберите главу из списка:
Интерфейс (управляемый)/УправляемаяФорма/УправляемаяФорма/Свойства/Элементы
Системные перечисления/Построитель запроса/ТипИзмеренияПостроителяЗапроса/Элементы
Системные перечисления/Система компоновки данных/ТипПримененияОтбораКомпоновкиДан...
Системные перечисления/Система компоновки данных/ТипГруппировкиКомпоновкиДанных/Эл...
Системные перечисления/Интерфейсные/ГруппыИЭлементы/Элементы
Общие объекты/Графическая схема/ЭлементГрафическойСхемыВыборВарианта/Свойства/Эле...
Общие объекты/Система компоновки данных/Схема компоновки данных/НаборДанныхОбъеди...
Поиск
Искать:
УправляемаяФорма
Искать
Закрыть
Справка
По текущей колонке
Учитывать регистр
Направление
Вперед
Назад
С начала
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ОтборКомпоновк...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ДоступноеПолеО...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ГруппаЭлементов...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ПоляГруппировки...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ПорядокКомпоно...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ГруппаВыбранны...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ВыбранныеПоля...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Общие объекты/Система компоновки данных/Настройки компоновки данных/ЗначенияПараме...
Показать
Отмена
Справка

Рис. 5.33. Поиск в списке глав при контекстном поиске

Нажмем кнопку *Показать*. Описание выбранной главы откроется на закладке *Индекс* в нижнем окне синтакс-помощника (рис. 5.34). При этом дерево синтакс-помощника, отображаемое вверху, не изменится. Чтобы найти, в какой ветке дерева находится открытое сейчас описание, нажмем кнопку *Найти текущий элемент в дереве*  , находящуюся над окном описания объектов синтакс-помощника.

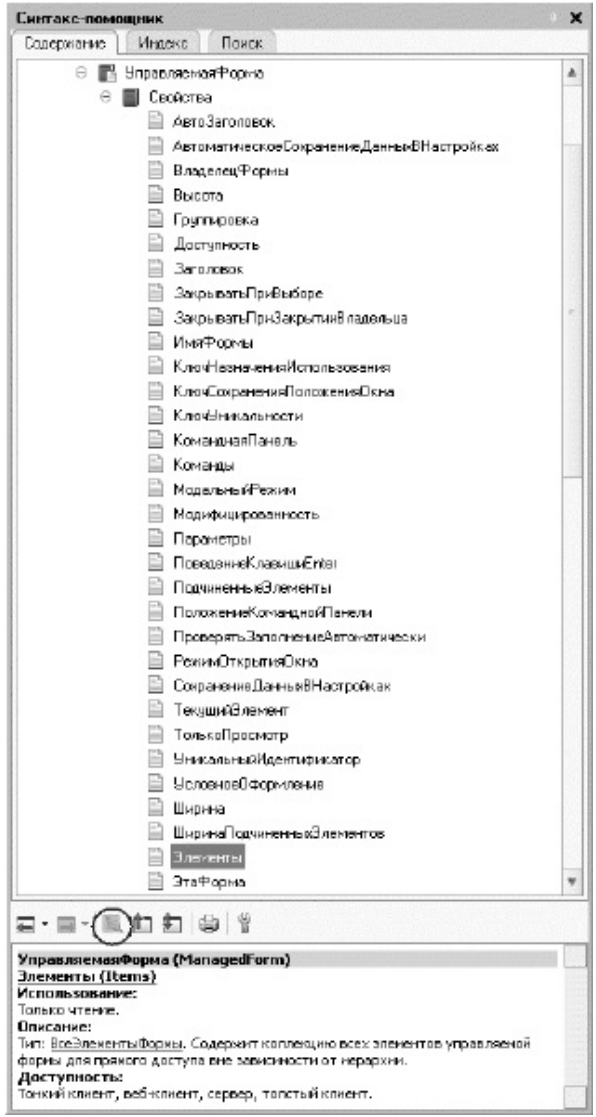

Синтакс-помощник
Содержание
Индекс
Поиск
УправляемаяФорма
Свойства
АвтоЗаголовок
АвтоматическоеСохранениеДанныхВНастройках
ВладелецФормы
Высота
Группировка
Доступность
Заголовок
ЗакрыватьПриВыборе
ЗакрыватьПриЗакрытииВладельца
ИмяФормы
КлючНазначенияИспользования
КлючСохраненияПоложенияОкна
КлючУникальности
КоманднаяПанель
Команды
МодальныйРежим
Модифицированность
Параметры
ПоведениеКлавишиEnter
ПодчиненныеЭлементы
ПоложениеКоманднойПанели
ПроверятьЗаполнениеАвтоматически
РежимОткрытияОкна
СохранениеДанныхВНастройках
ТекущийЭлемент
ТолькоПросмотр
УникальныйИдентификатор
УсловноеОформление
Ширина
ШиринаПодчиненныхЭлементов
Элементы
ЭтаФорма
УправляемаяФорма (ManagedForm)
Элементы (Items)
Использование:
Только чтение.
Описание:
Тип: ВсеЭлементыФормы. Содержит коллекцию всех элементов управляемой формы для прямого доступа вне зависимости от иерархии.
Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент.

Рис. 5.34. Описание свойства «Элементы» объекта «УправляемаяФорма»

На закладке *Содержание* синтакс-помощника будет показан раздел, соответствующий текущему описанию. Таким образом, в дереве содержания мы видим, что *Элементы* – это свойство объекта *УправляемаяФорма*. Далее, раскрывая соответствующие ссылки, как описано в первом способе работы с синтакс-помощником, мы поймем, что содержится в переменной *СтрокаТабличнойЧасти*, и как можно обращаться к данным в ее колонках. И так далее.

Есть также еще одна полезная возможность использования синтакс-помощника. Можно ограничить состав объектов, которые будут отображаться в нем. Так как мы находимся на клиенте, в форме, имеет смысл ограничиться только объектами встроенного языка, доступными в режимах *Тонкий клиент* и *Веб-клиент.* Для этого нужно выполнить команду главного меню *Сервис* > *Параметры* или нажать кнопку *Открыть режим настройки параметров*  , находящуюся над окном описания объектов синтакс-помощника. На закладке *Справка* окна *Параметры* можно снять или поставить отметку у нужных режимов исполнения (рис. 5.35).

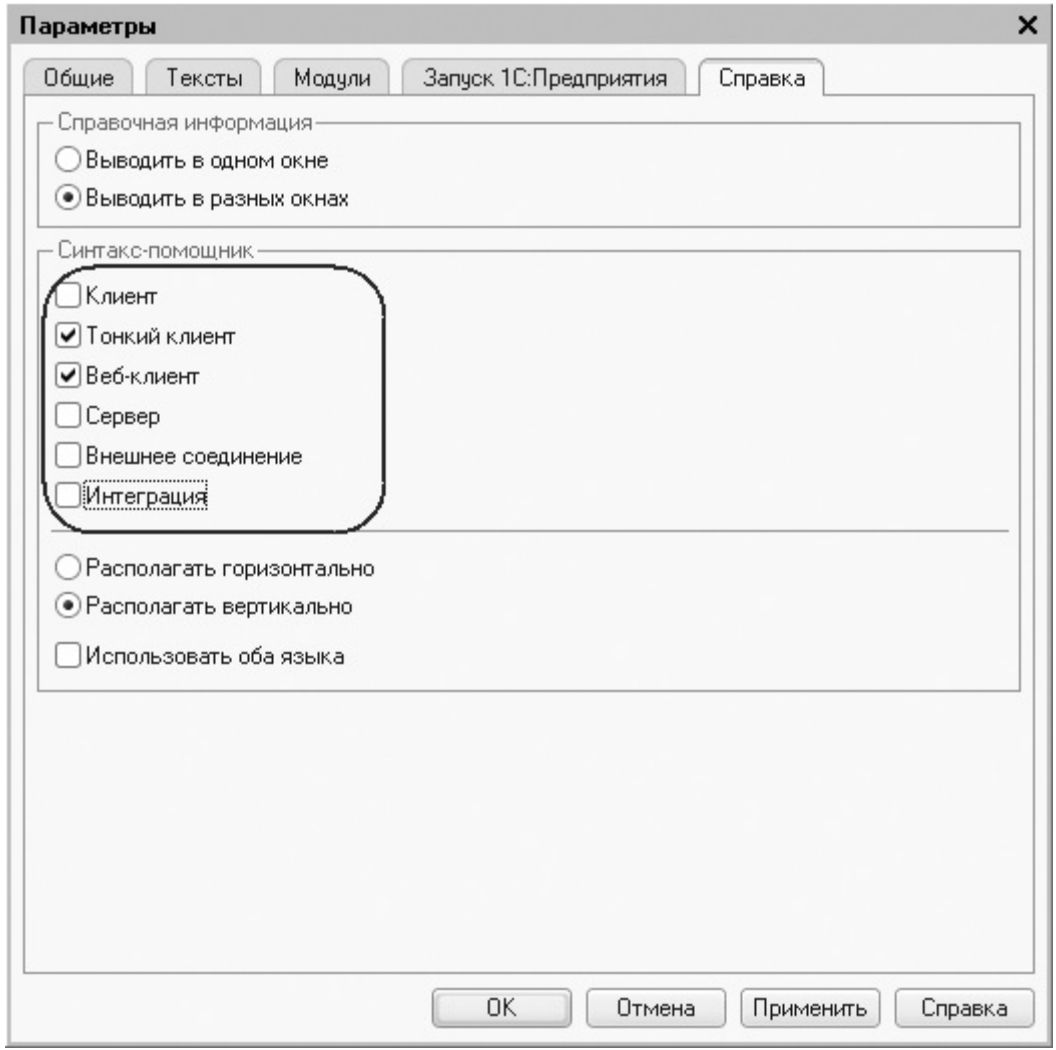


Рис. 5.35. Фильтрация объектов, показываемых в синтакс-помощнике

ВНИМАНИЕ!

Когда вы пишите процедуру, исполняемую, например, на сервере, и хотите воспользоваться синтакс-помощником, не забудьте установить соответствующий флажок режима исполнения для отображения объектов встроенного языка.

Анализ кода с помощью отладчика

Пользоваться отладчиком наиболее удобно в тех случаях, когда нужно написать какой-то собственный код. Потому что в отличие от синтакс-помощника, где нужно, вообще говоря, хорошо представлять контексты исполнения, структуру объектов и пр., с помощью отладчика ничего этого представлять не нужно.

Можно просто остановиться в конкретном месте программы и посмотреть, какие же свойства здесь доступны или какие программные объекты здесь используются.

Отладчик — вспомогательный инструмент, облегчающий разработку и отладку программных модулей системы «1С:Предприятие». Отладчик предоставляет следующие возможности:

- пошаговое выполнение модуля,

- расстановка точек останова,
- прерывание и продолжение выполнения модуля,
- возможность отладки нескольких модулей одновременно,
- вычисление выражений для анализа состояния переменных,
- просмотр стека вызовов процедур и функций,
- возможность остановки по возникновению ошибки,
- возможность редактирования модуля в процессе отладки.

Но мы пока не будем подробно останавливаться на всех этих возможностях, а рассмотрим использование отладчика для того, чтобы разобраться с обработчиком события *МатериалыКоличествоПриИзменении*, приведенном в листинге 4.1.

Если в режиме *Конфигуратор* редактируется текст модуля, то становятся доступными команды пункта главного меню *Отладка*, позволяющие расставлять и убирать точки останова. *Точки останова* позволяют прерывать выполнение программы в тех местах, где они установлены. Затем разработчик может проанализировать значение и тип выражений и переменных модуля в момент остановки и продолжить выполнение программы до следующей точки останова и т. д.

Итак, откроем форму документа *ПриходнаяНакладная*, перейдем на закладку

Модуль, откроем текст процедуры *МатериалыКоличествоПриИзменении*. Мы видим, что в пункте главного меню *Отладка* и на панели инструментов конфигуратора стали доступны команды для работы с точками останова (рис. 5.36).

Рис. 5.36. Панель инструментов точки останова

Установить точку останова можно двойным щелчком мыши в служебной области слева от нужной строки. При этом в служебной области, располагающейся вдоль левой границы, появится значок точки останова.

ПРИМЕЧАНИЕ

Для установки точки останова можно также установить указатель в любое место строки модуля, в которой требуется выполнить остановку, и выполнить команду ***Отладка > Точка останова*** *или нажать соответствующую кнопку*  *на панели инструментов* ***Точки останова****.*

Снять точку останова можно двойным щелчком мыши на этой точке в служебной области.

ПРИМЕЧАНИЕ

Для снятия точки останова можно также установить указатель в любое место

строки модуля и выполнить команду ***Отладка > Отключить точку останова*** *или нажать соответствующую кнопку*  *на панели инструментов* ***Точки останова***.

Для снятия всех точек останова нужно выполнить команду *Отладка > Отключить все точки останова* или нажать соответствующую кнопку на панели инструментов *Точки останова*.

Итак, дважды щелкнем в служебной области слева от первой строки процедуры *МатериалыКоличествоПриИзменении* (рис. 5.37).

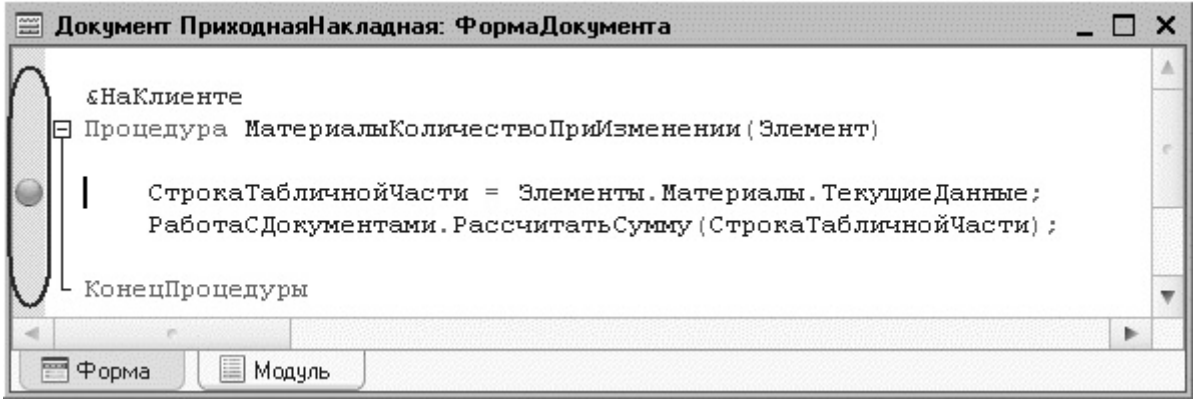


Рис. 5.37. Установка точки останова в процедуре «МатериалыКоличествоПриИзменении»

Чтобы иметь возможность отлаживать код на встроенном языке, нужно обеспечить запуск приложения, в котором исполняется код, в отладочном

режиме. Для начала отладки выполним команду *Отладка > Начать отладку* или нажмем соответствующую кнопку на панели инструментов конфигуратора. Конфигуратор запустит «1С:Предприятие» в отладочном режиме. На самом деле мы так делали и раньше, но не устанавливали точек останова, и программа не прерывалась.

Откроем список документов *Приходные накладные* и откроем любой из двух созданных нами документов. Изменим поле *Количество* в любой строке документа. Выполнение программы прервется, и в конфигураторе будет открыта процедура *МатериалыКоличествоПриИзменении* в точке останова. Рядом с ней появится стрелка, указывающая на текущую исполняемую строку модуля (рис. 5.38).

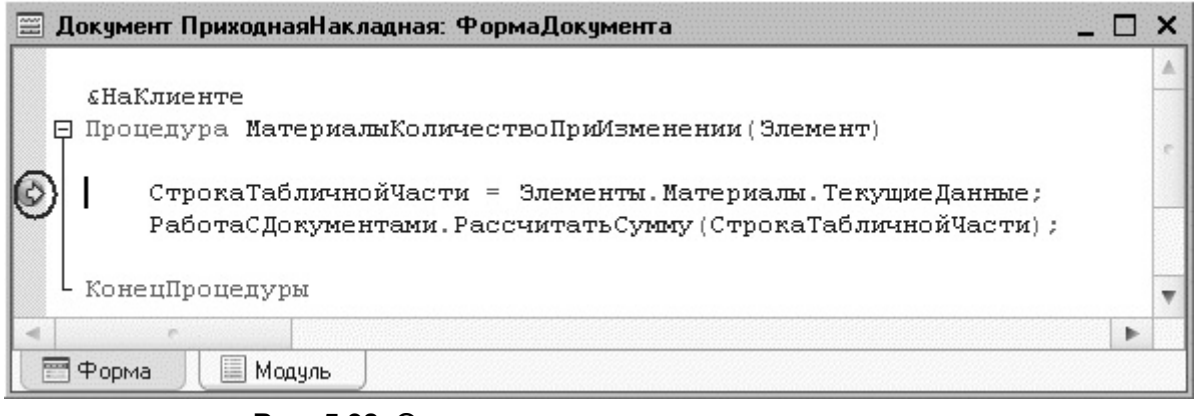


Рис. 5.38. Остановка программы в точке останова

Также мы видим, что в пункте главного меню *Отладка* и на панели инструментов конфигуратора стали доступны команды для работы с конфигурацией в процессе отладки (рис. 5.39).

Рис. 5.39. Панель инструментов «Отладка конфигурации»

С помощью этих команд и кнопок в панели инструментов *Отладка конфигурации* можно пошагово выполнять программу (кнопки *Шагнуть через, Шагнуть в, Шагнуть из*) или продолжить отладку (кнопка *Продолжить отладку*) до следующей точки останова. С помощью кнопок *Табло* и *Вычислить выражение* можно получить значения интересующих вас выражений в каждый момент остановки программы. С помощью кнопки *Стек вызовов* можно проследить последовательность вызова процедур и функций.

Но мы сейчас уже остановились в интересующей нас строке процедуры *МатериалыКоличествоПриИзменении*. Заметим, что эта строка еще не выполнялась, поэтому значения переменных будут еще не заполнены. Чтобы увидеть их после выполнения строки, можно нажать кнопку *Шагнуть через*.

Итак, мы хотим понять, что означает выражение *Элементы.Материалы.ТекущиеДанные* и что содержит этот объект в момент остановки программы. Двойным щелчком выделим слово *Элементы* и нажмем

кнопку *Вычислить выражение (Shift + F9)*  на панели инструментов *Отладка конфигурации*.

В поле *Выражение* попадет выделенное нами слово *Элементы*. В соответствующих колонках мы увидим значение и тип этого объекта. Мы видим, что объект *Элементы* является коллекцией значений *ВсеЭлементыФормы*, содержащей все элементы формы. Раскроем этот объект (рис. 5.40).

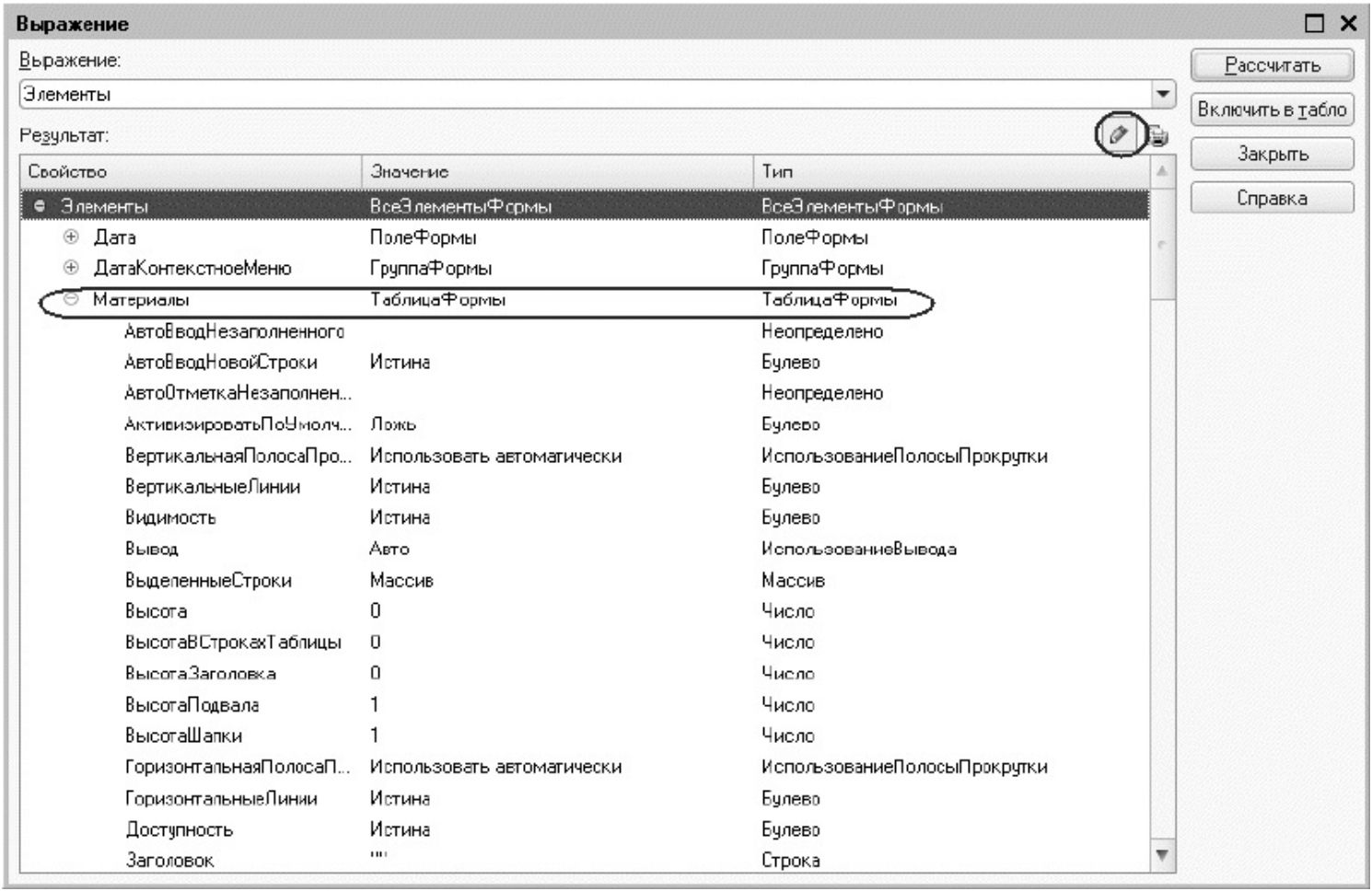


Рис. 5.40. Объект «Элементы»

Над окном *Результат* находится кнопка *Показать значения в отдельном*

окне, по нажатию которой (или *F2*) можно будет просматривать содержимое коллекций.

Дальше нас интересует выражение *Материалы*. Найдем его в списке элементов формы. Мы видим, что это объект *ТаблицаФормы*. Раскроем этот объект и увидим его свойства. Нас интересует свойство *ТекущиеДанные*. Найдем в списке и раскроем этот объект. Мы увидим данные текущей строки нашей табличной части, их значение и тип. Причем *Количество* содержит только что измененное нами значение (рис. 5.41).

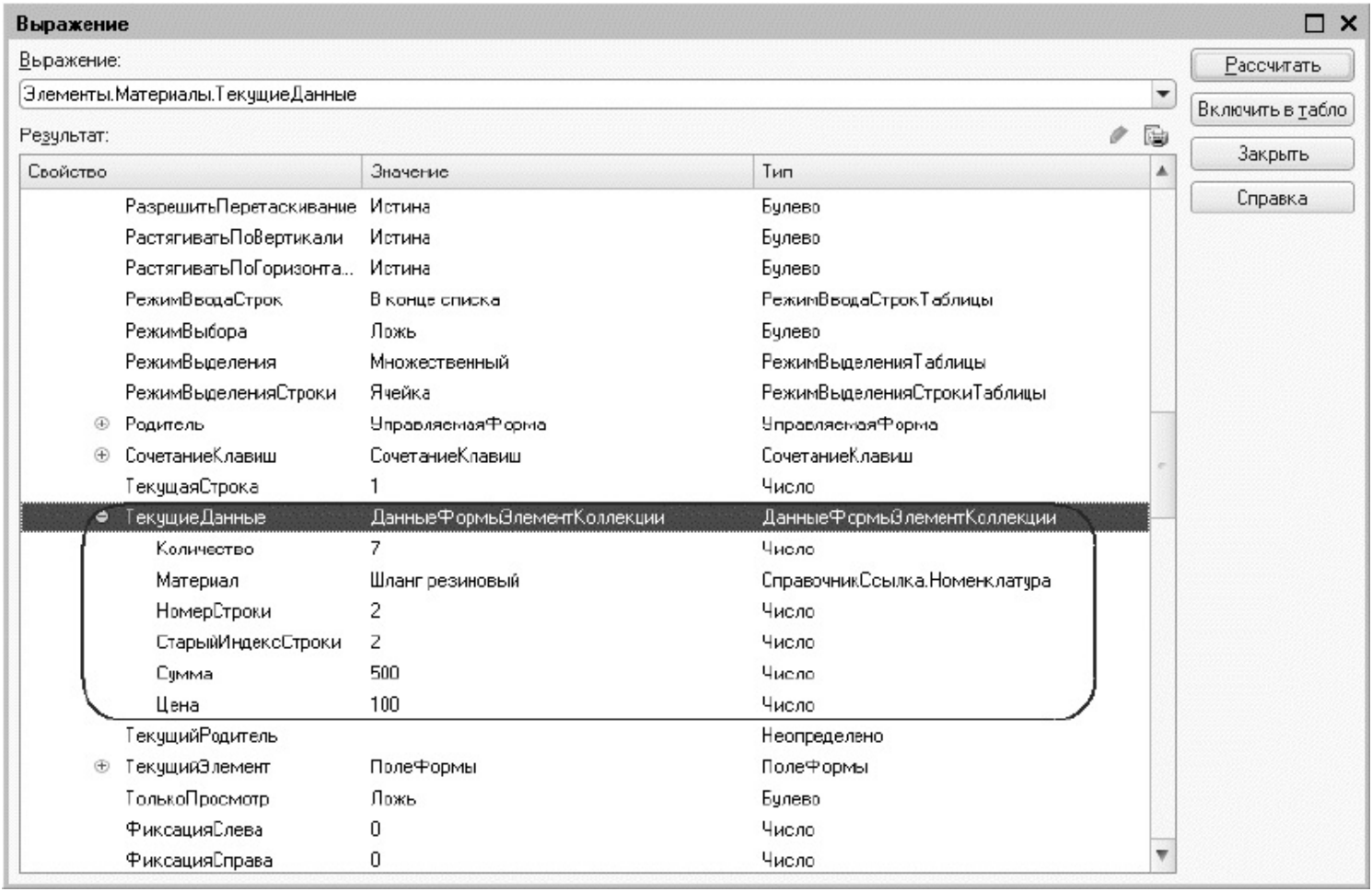


Рис. 5.41. Объект «Элементы.Материалы.ТекущиеДанные»

Сейчас значение переменной *СтрокаТабличнойЧасти* не определено, так как

мы остановились до выполнения строки, то есть до присвоения этой переменной объекта *Элементы.Материалы.ТекущиеДанные*. Закроем окно *Выражение* и нажмем кнопку *Шагнуть через* на панели инструментов *Отладка конфигурации*. Программа остановится на следующей строке, на которую указывает стрелка. Затем нажмем кнопку *Шагнуть в* на этой же панели, так как нам нужно шагнуть в процедуру общего модуля *РаботаСДокументами*, где вычисляется значение переменной *СтрокаТабличнойЧасти*. Программа перейдет в процедуру *РассчитатьСумму* общего модуля *РаботаСДокументами*. Двойным щелчком выделим выражение *СтрокаТабличнойЧасти* и нажмем кнопку *Вычислить выражение*. Раскроем объект *СтрокаТабличнойЧасти* (рис. 5.42).

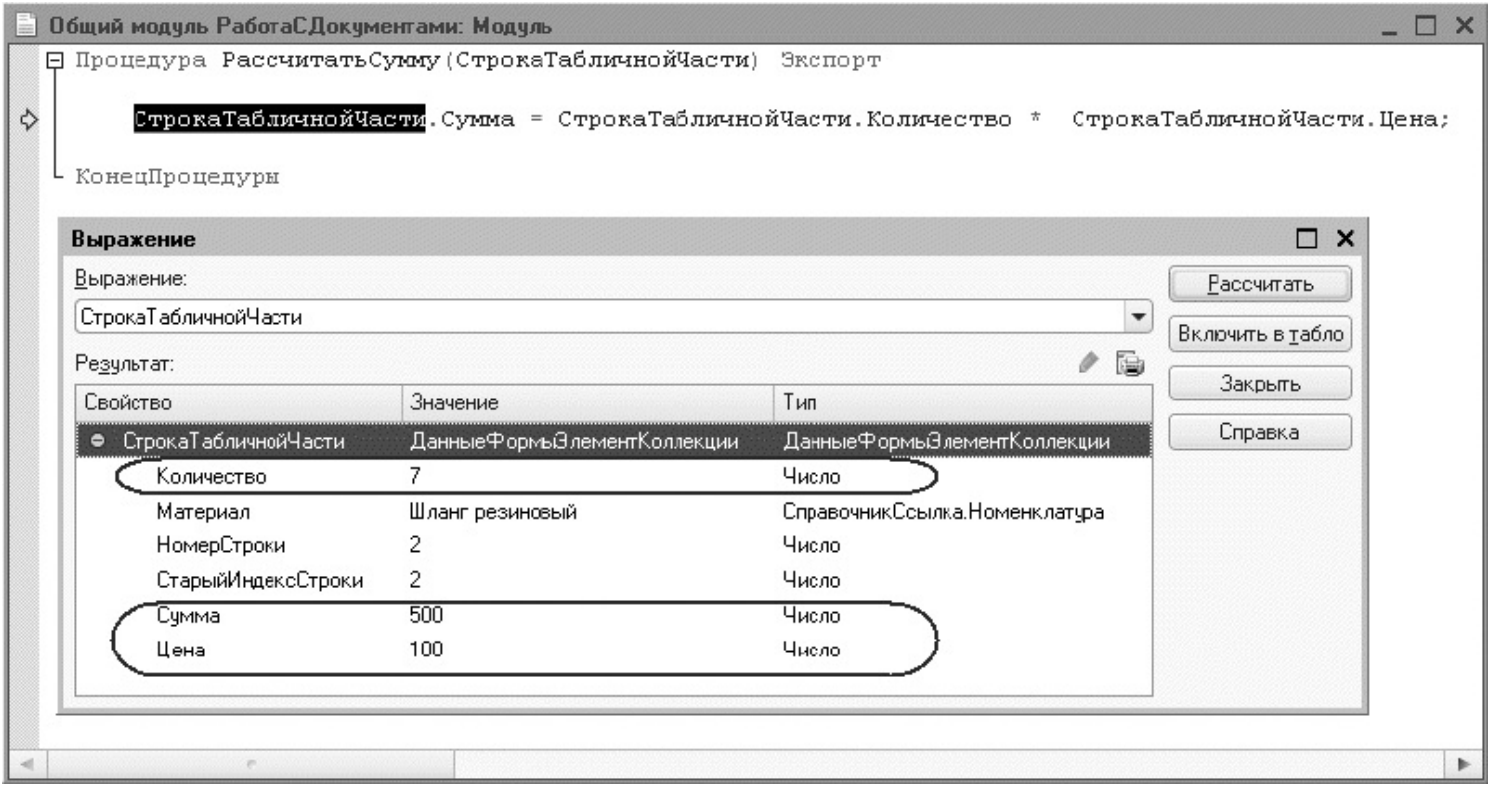


Рис. 5.42. Объект «СтрокаТабличнойЧасти»

Мы видим, что теперь переменная *СтрокаТабличнойЧасти* содержит объект *ДанныеФормыЭлементКоллекции*. Но значения колонки *Сумма* еще не пересчитаны, так как вторая строка кода еще не исполнялась. Закроем окно *Выражение* и еще раз нажмем кнопку *Шагнуть через* на панели инструментов *Отладка конфигурации*. Программа выполнит процедуру *РассчитатьСумму*

общего модуля *РаботаСДокументами* и остановится в ее конце. Теперь можно подвести курсор к колонке *Количество* или *Сумма,* и система во всплывающей подсказке покажет текущее значение (рис. 5.43).

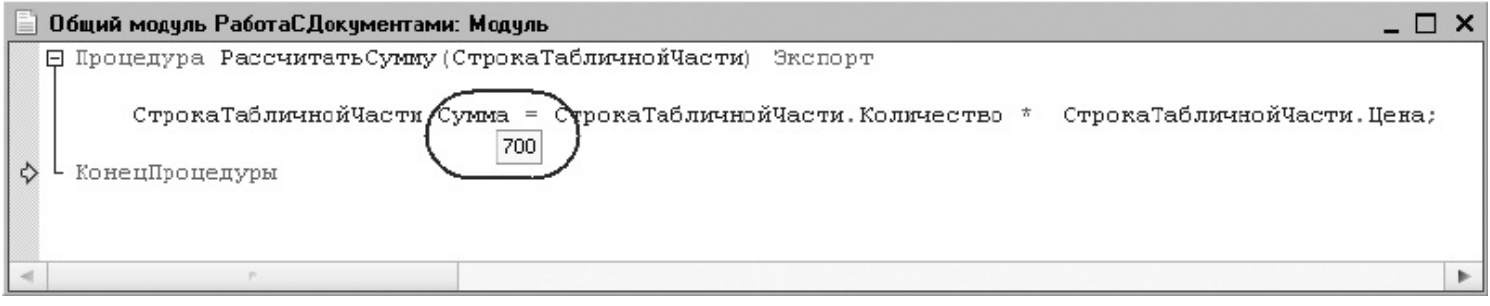


Рис. 5.43. Объект «СтрокаТабличнойЧасти»

Мы видим, что значения колонки *Сумма* пересчитались по заданному алгоритму, и значит, объект *Элементы.Материалы.ТекущиеДанные* содержит новые значения.

Таким образом, можно поставить точку останова в тело цикла и смотреть значения переменных по мере его выполнения и т. д.

После анализа переменных и выражений в момент остановки можно продолжить отладку, нажав кнопку *Продолжить отладку*. Если необходимо прервать отладку, нужно выполнить команду *Отладка > Завершить*. В процессе отладки допускается редактирование текущей конфигурации и сохранение

изменений.

ВНИМАНИЕ!

Хотя в процессе отладки возможно редактирование отлаживаемого модуля, отладчик не производит компилирование измененного кода – продолжается отладка кода конфигурации базы данных (на момент запуска отладчика или подключения). Для отладки изменений, внесенных в конфигурацию, необходимо выполнить обновление конфигурации базы данных.

В заключение скажем о следующих полезных приемах работы с отладчиком.

Когда вы находитесь в модуле формы и вам нужно написать какой-то обработчик, можно использовать свойство *ЭтаФорма*, чтобы посмотреть свойства контекста этой конкретной формы со всеми теми расширениями, которые у нее есть и пр. Для этого, после того как вы остановились в какой-либо точке останова, нужно нажать кнопку *Вычислить выражение (Shift + F9)*  на панели инструментов *Отладка конфигурации*. В поле *Выражение* ввести слово *ЭтаФорма* и нажать кнопку *Рассчитать* (рис. 5.44).

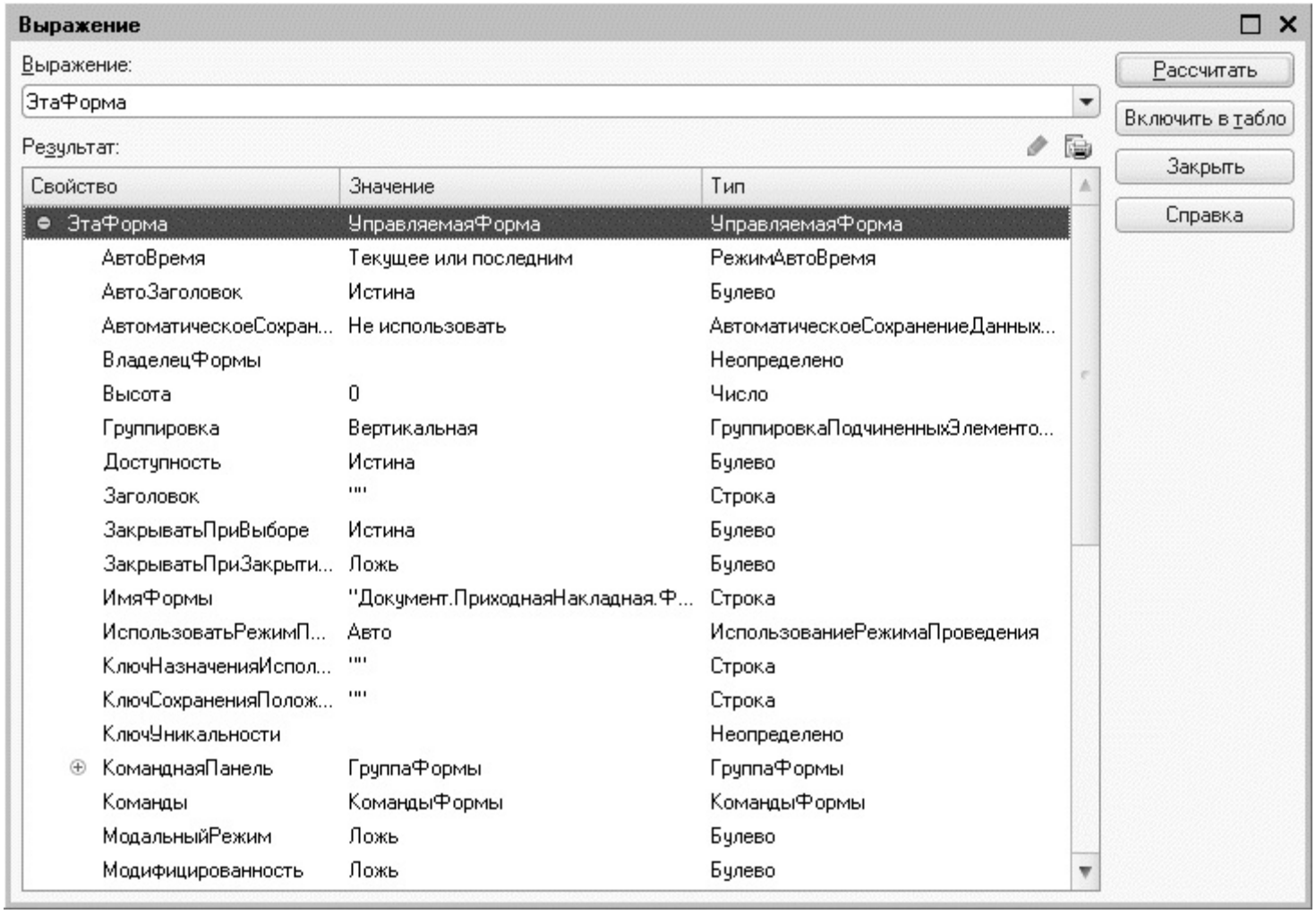


Рис. 5.44. Объект «ЭтаФорма»

Раскрыв этот объект, вы увидите тип и свойства объектов встроенного языка, которые используются в момент останова.

Аналогичным образом, когда вы находитесь в модуле объекта или набора записей, можно использовать свойство *ЭтотОбъект*, чтобы посмотреть свойства контекста модуля объекта или набора записей.

Объекты, объекты, объекты…

Что такое объект в терминах «1С:Предприятия»? Этот вопрос зачастую ставит в тупик не только начинающих разработчиков, но и людей, имеющих определенный опыт разработки на платформе «1С:Предприятие».

Основная трудность заключается в том, что всегда нужно ясно представлять себе, в каком контексте употребляется этот термин.

Как правило, термин *объект* употребляется в одном из трех контекстов:

- конфигурация,
- база данных,
- встроенный язык.

Говоря о конфигурации, термином *объект конфигурации* мы обозначаем

некоторую совокупность описания данных и алгоритмов работы с этими данными. Например, в конфигурации может существовать объект *Справочник Сотрудники*.

На основании каждого объекта конфигурации в базе данных создается информационная структура, в которой будут храниться данные.

Так вот, когда мы говорим о базе данных, термином объект мы обозначаем всего лишь некий элемент такой информационной структуры. Характерной особенностью такого элемента является то, что на него (как на совокупность данных) существует ссылка, которая может являться значением какого-либо поля другой информационной структуры.

Например, в базе данных существует справочник *Сотрудники*, в котором есть сотрудник *Иванов*. В этом случае элемент справочника, содержащий информацию о сотруднике Иванове, будет являться объектом базы данных. И если в документе *ПриходнаяНакладная* будет существовать реквизит *ОтветственноеЛицо*, то тип значения этого реквизита будет ссылкой на объект базы данных, то есть на элемент справочника, содержащий информацию об Иванове.

Если же мы начинаем говорить о встроенном языке и о том, каким образом средствами встроенного языка работать со справочниками, то термином объект

мы обозначаем тип данных, позволяющий получить доступ к данным и обладающий набором свойств и методов.

Существует целый ряд объектов встроенного языка, позволяющих работать со справочниками (*СправочникиМенеджер*, *СправочникМенеджер.<имя>*, *СправочникСсылка.<имя>* и т. д.). Среди них есть один объект, который предоставляет доступ к объекту справочника в базе данных, – *СправочникОбъект.<имя>* (рис. 5.45).

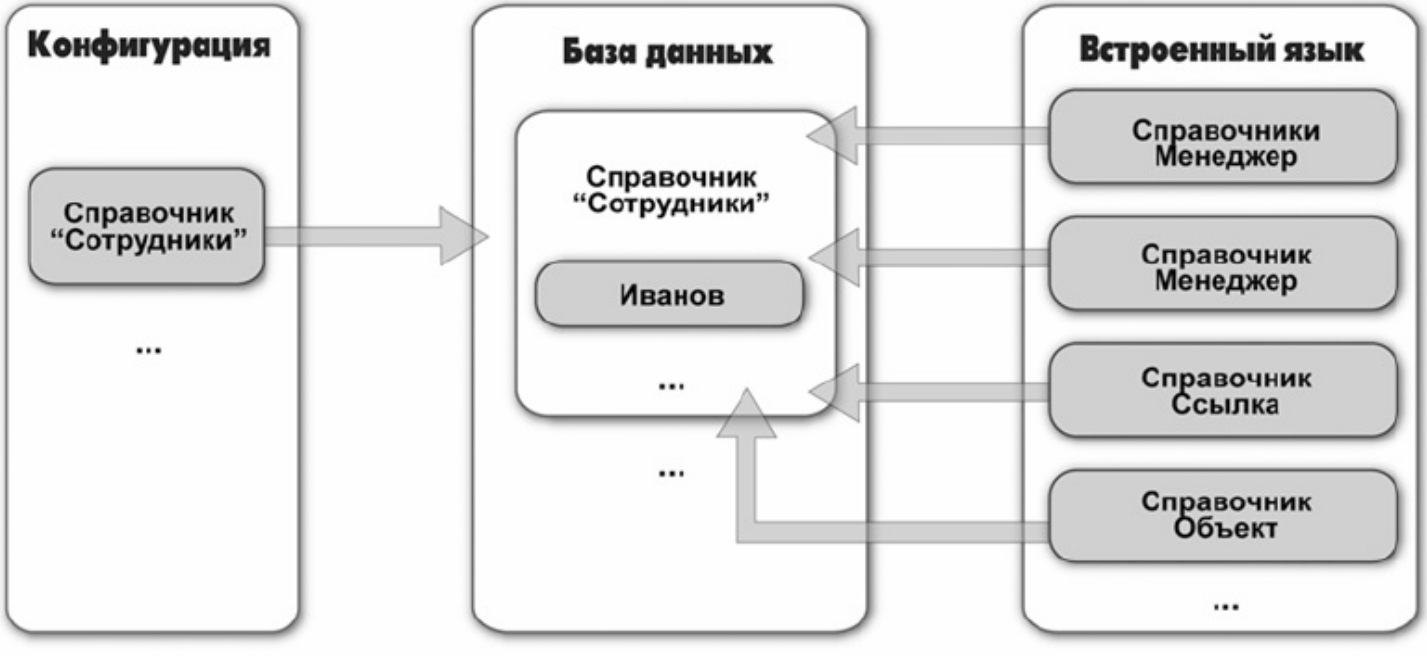


Рис. 5.45. Объект конфигурации, объект базы данных, объекты встроенного языка

Сервер и клиенты

На предыдущем занятии мы создали одну процедуру для обработки нескольких событий и разместили ее в общем модуле *РаботаСДокументами*. В свойствах этого общего модуля мы устанавливали флажки *Клиент* и *Сервер*. Объясним подробнее, откуда в «1С:Предприятии» взялись вообще какие-то «клиенты» и

«серверы».

Система «1С:Предприятие» поддерживает два варианта работы системы: файловый и клиент-серверный.

Файловый вариант работы с информационной базой рассчитан на персональную работу одного пользователя или работу небольшого количества пользователей в локальной сети. В этом варианте все данные информационной базы (конфигурация, база данных, административная информация) располагаются в одном файле (рис. 5.46).

Рис. 5.46. Файловый вариант работы

Основное назначение файлового варианта – быстро и легко установить систему и работать с ней. Например, чтобы что-то посмотреть или доработать дома или на ноутбуке. В файловом варианте тоже можно вести реальную учетную работу, но при этом нужно понимать, что он не предоставляет абсолютно всех тех же возможностей по масштабируемости, защите данных, какие имеет клиент-серверный вариант. Поэтому, если вы самостоятельно ведете бухгалтерский учет, или у вас небольшой коллектив сотрудников, и вам не требуется гарантированная защита данных от несанкционированного использования сотрудниками, и вы имеете относительно небольшой объем

данных, можно работать в файловом варианте. В остальных же случаях нужно использовать клиент-серверный вариант.

Клиент-серверный вариант предназначен для использования в рабочих группах или в масштабе предприятия. Он реализован на основе трехуровневой архитектуры «клиент-сервер» (рис. 5.47).


1С:Предприятие 8
(клиентская часть)
Кластер серверов
1С:Предприятия 8
Сервер базы данных
• Microsoft SQL Server
• PostgreSQL
• IBM DB2
• Oracle Database
База данных

Рис. 5.47. Трехуровневая архитектура «1С:Предприятия»

Клиент-серверный вариант работы – это основной вариант для работы в многопользовательской среде с большим объемом данных. Он предоставляет абсолютно все возможности по масштабируемости, администрированию и защите данных. Однако он требует значительных усилий по установке и администрированию.

При этом физически серверная и клиентские части системы «1С:Предприятие 8» могут располагаться как на разных компьютерах, так и на одном. Главное, что пользователь не имеет непосредственного доступа к серверу баз данных, и это позволяет обеспечивать безопасность данных. А в файловом варианте база данных должна находиться на некотором общем сетевом ресурсе, доступном всем пользователям.

Система «1С:Предприятие» изначально рассчитана на клиент-серверный вариант работы. Хотя сейчас вы разрабатываете свою учебную конфигурацию в файловом варианте работы, она будет работать и в клиент-серверном варианте без ваших дополнительных усилий.

Прикладные решения разрабатываются один раз и одинаково работают, что в одном, что в другом варианте. То есть переход с одного варианта на другой не требует переделки конфигурации.

Это достигается за счет того, что конфигурация разрабатывается всегда исходя из клиент-серверной архитектуры. В системе «1С:Предприятие» просто нет возможности разрабатывать ее по-другому. И в том случае, когда используется файловый вариант работы, система при исполнении прикладного решения просто «имитирует» наличие сервера на клиентском компьютере.

Клиент-серверная архитектура разделяет всю работающую систему на три различные части, определенным образом взаимодействующие между собой – клиент, сервер «1С:Предприятия» и сервер баз данных.

Клиентское приложение – это программа, часть системы «1С:Предприятие». Основное ее назначение – организация пользовательского интерфейса, отображение данных с возможностью их изменения. Кроме этого, клиентское приложение может исполнять код на встроенном языке (то есть какие-то алгоритмы разработчика), но оперирует при этом лишь очень ограниченным пространством типов встроенного языка. Такой подход позволяет клиентскому приложению быть очень «легким», не требовать много ресурсов, «путешествовать» по Интернету и работать даже в среде веб-браузеров.

Клиентское приложение взаимодействует с сервером «1С:Предприятия». *Сервер «1С:Предприятия»* – это тоже программа, часть системы «1С:Предприятие».

Одна из основных задач этой программы – передавать запросы от клиентского приложения к серверу баз данных и возвращать обратно клиенту результаты этих запросов.

Другая задача сервера – исполнение большинства алгоритмов на встроенном языке, подготовка данных для отображения форм, отчетов и т. д. То есть все сложные вычисления, требующие непосредственной работы с данными, исполняются именно на сервере. При этом на сервере доступно практически все пространство типов встроенного языка «1С:Предприятия» (за исключением, естественно, чисто интерфейсных типов, потому что у сервера нет никакой интерфейсной части, так как он общается не с пользователями, а только с другими программами: клиентским приложением и с сервером баз данных).

Сервер баз данных – это тоже программа. Она уже не является частью системы «1С:Предприятие», это специализированная программа, поставляемая сторонними производителями. Ее основное назначение – это организация и ведение баз данных – структурированных организованных наборов данных, описывающих характеристики каких-либо физических или виртуальных систем.

В настоящее время система «1С:Предприятие» может работать со следующими серверами баз данных:

- Microsoft SQL Server,

- PostgreSQL,
- IBM DB2,
- Oracle Database.

Компиляция общих модулей

На предыдущем занятии мы создали одну процедуру для обработки нескольких событий и разместили ее в общем модуле *РаботаСДокументами*. У этого модуля, как и у всякого общего модуля конфигурации, существует набор свойств: *Клиент (управляемое приложение), Сервер и Внешнее соединение*. Значения этих свойств (истина/ложь) определяют, где будут скомпилированы экземпляры этих модулей.

Расскажем подробнее о том, что происходит, когда мы устанавливаем те или иные флажки у общего модуля.

Прежде всего, необходимо понимать, зачем необходима компиляция. Дело в том, что все, что мы разработали и написали в конфигурации, – пока только некая «заготовка». Платформа, когда мы запускаем ее в режиме *1С:Предприятие*, превращает все это в программу, которую уже можно исполнить на компьютере, – компилирует. При этом, как мы уже сказали ранее, есть разные части системы, в которых исполняется код – сервер, клиентские приложения. Поэтому для общих модулей мы можем и должны в явном виде

указать, где, на какой «стороне» они должны быть скомпилированы – на сервере или клиенте.

В общем модуле *РаботаСДокументами* мы установили свойство *Клиент (управляемое приложение),* см. рис. 4.24. Это значит, что экземпляры этого модуля будут скомпилированы только на стороне клиента в контекстах тонкого клиента и веб-клиента (рис. 5.48).

Рис. 5.48. Компиляция общего модуля на стороне сервера или клиента

Если у модуля установлено только свойство *Сервер*, то модуль будет скомпилирован только на стороне сервера.

Клиентских приложений два: *Тонкий клиент* и *Веб-клиент*.

Тонкий клиент – это файл 1cv8c.exe. Именно его запускает платформа автоматически, когда мы начинаем отладку.

Веб-клиент не существует в виде файла, потому что он исполняется не в среде операционной системы, а в среде интернет-браузера. Пользователю достаточно всего лишь запустить свой браузер, ввести адрес веб-сервера, на котором опубликована информационная база, и веб-клиент скачается к нему на компьютер и начнет исполняться.

Директивы компиляции

На предыдущем занятии мы создали обработчики событий *ПриИзменении* у некоторых элементов формы. Когда система создавала объявления процедур, в которых располагается код этих обработчиков, она предваряла их директивами *&НаКлиенте*. Расскажем подробнее о том, что это значит.

Дело в том, что форма существует и на сервере, и на клиенте одновременно. Поэтому для каждой процедуры, которая существует в модуле формы, нужно указывать в явном виде контекст ее исполнения, где она будет исполняться: на сервере или на клиенте. Наличие директив *&НаКлиенте* или *&НаСервере* или *&НаСервереБезКонтекста* связано с тем, что при использовании встроенного языка в модуле формы клиентский код должен быть четко отделен от серверного. Таким образом, указав одну из директив, разработчик в явном виде программирует серверную или клиентскую части.

В модуле формы одновременно можно поместить процедуры с различными директивами исполнения и передавать выполнение кода с клиента на сервер. Также из клиентской процедуры модуля формы можно вызвать процедуру общего модуля (как и сделано в нашем примере), которая может выполняться как на клиенте, так и на сервере, в зависимости от установленных свойств модуля *Тонкий клиент* или *Сервер*.

Исполнение кода на клиенте и на сервере

На предыдущем занятии мы создали одну процедуру для обработки нескольких событий и разместили ее в общем модуле *РаботаСДокументами*. При этом в его свойствах мы поставили флажок *Клиент (управляемое приложение)* и сняли флажок *Сервер*. Эта процедура вызывалась из модуля формы, в котором для обработчика события также была указана директива,

определяющая контекст исполнения этого обработчика – *&НаКлиенте*. Расскажем подробнее, как именно происходит исполнение кода в этом случае.

После запуска прикладного решения выполнение кода всегда начинается на клиенте. В процессе работы выполнение кода может быть передано на сервер посредством вызова процедуры общего модуля, скомпилированного на сервере.

Дело в том, что при вызове процедуры или функции ее поиск осуществляется сначала на клиенте. Если скомпилированный контекст клиента не содержит данную процедуру, то поиск продолжается на стороне сервера. Если вызываемая процедура будет найдена, то выполнение кода будет передано на сервер. После завершения процедуры выполнение кода продолжится на клиенте.

Например, при изменении цены в форме приходной накладной вызывается процедура модуля формы *ЦенаПриИзменении()*. Она исполняется на стороне клиента в режиме тонкого клиента. Во второй строке этого обработчика вызывается процедура *РассчитатьСумму()*. Эта процедура находится в общем модуле *РаботаСДокументами*, скомпилированном также на клиенте. Она исполняется в режиме тонкого клиента. После выполнения этой процедуры управление опять передается в модуль формы (рис. 5.49).

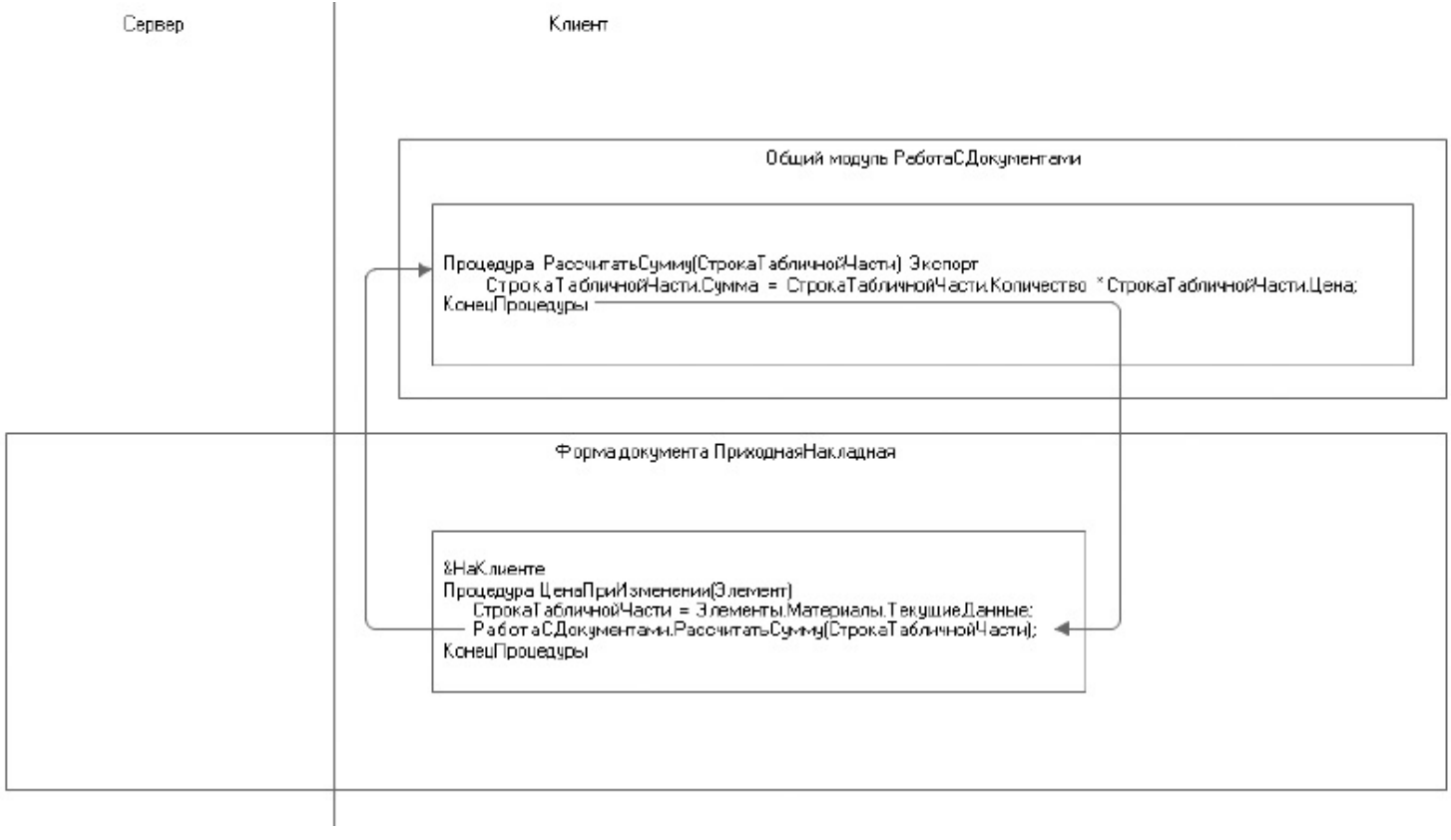


Рис. 5.49. Передача выполнения кода между модулями на клиенте

Занятие 6 (0:50). Регистры накопления

Продолжительность

Ориентировочная продолжительность занятия – 50 минут.

На этом занятии мы познакомимся с объектом конфигурации *Регистр накопления*. Вы узнаете, для чего используется этот объект, какой структурой он обладает и каковы его отличительные особенности.

Затем мы создадим с вами один из регистров накопления, который будет использоваться в нашей конфигурации и отражать изменение данных в процессе работы ранее созданных нами документов.

Зачем нужен регистр накопления

Итак, мы с вами подошли к одному из главных моментов разработки любой конфигурации – созданию механизма учета накопления данных.

Казалось бы, все необходимое мы с вами уже создали: у нас есть что расходовать и приходовать (справочники), и у нас есть чем расходовать и приходовать (документы). Осталось только построить несколько отчетов, и автоматизация ООО «На все руки мастер» будет закончена.

Однако это не так.

Во-первых, путем анализа документов можно, конечно, получить требуемые нам выходные данные. Но представьте, что завтра ООО «На все руки мастер» решит немного изменить свои бизнес-процессы, и нам потребуется ввести в конфигурацию еще один документ (или несколько документов).

Например, сейчас мы полагаем, что товары поступают в ООО «На все руки мастер» и затем расходуются. Руководство захотело усилить материальный контроль и решило приходовать товары на основной склад организации и затем выдавать их материально ответственным лицам. В этом случае нам придется добавить в конфигурацию еще один документ, который будет фиксировать перемещение материалов между основным складом и материально ответственными лицами. И очевидно, нам придется переработать все отчеты, которые были нами созданы к этому моменту с тем, чтобы они учитывали изменения, вносимые новым документом. А представьте, если в нашей конфигурации не два, а двадцать документов?!

Во-вторых, отчеты, анализирующие документы, будут работать довольно медленно, что будет вызывать раздражение пользователей и недовольство руководителей.

Поэтому в системе «1С:Предприятие» есть несколько объектов конфигурации,

которые позволяют создавать в базе данных структуры, предназначенные для накопления информации в удобном для последующего анализа виде. Использование таких хранилищ данных позволяет нам, с одной стороны, накапливать в них данные, поставляемые различными документами (или другими объектами базы данных), а с другой стороны, легко создавать нужные нам отчеты или использовать эти данные в алгоритмах работы конфигурации (рис. 6.1).

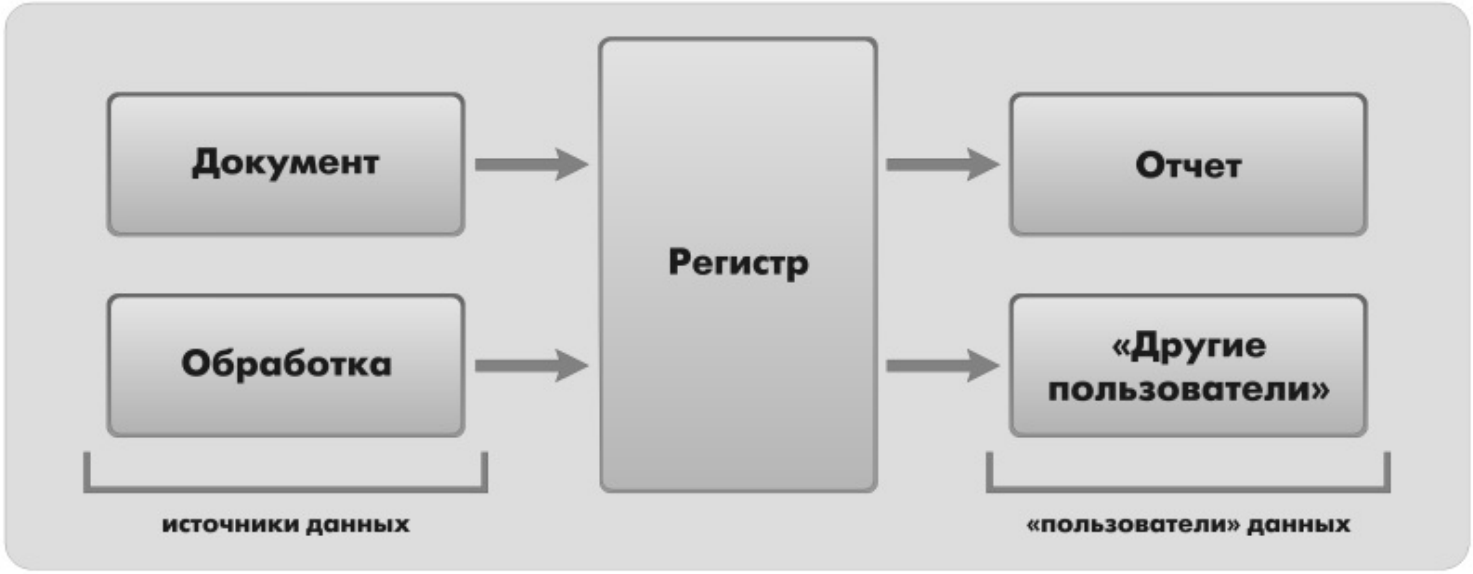


Рис. 6.1. Алгоритм работы конфигурации

В конфигурации существует несколько объектов, называемых *регистрами*,

для описания подобных хранилищ. Сейчас мы рассмотрим один из них.

Что такое регистр накопления

Объект конфигурации *Регистр накопления* предназначен для описания структуры накопления данных. На основе объекта конфигурации *Регистр накопления* платформа создает в базе данных таблицы, в которых будут накапливаться данные, поставляемые различными объектами базы данных.

Эти данные будут храниться в таблицах в виде отдельных записей, каждая из которых имеет одинаковую заданную в конфигураторе структуру (рис. 6.2).

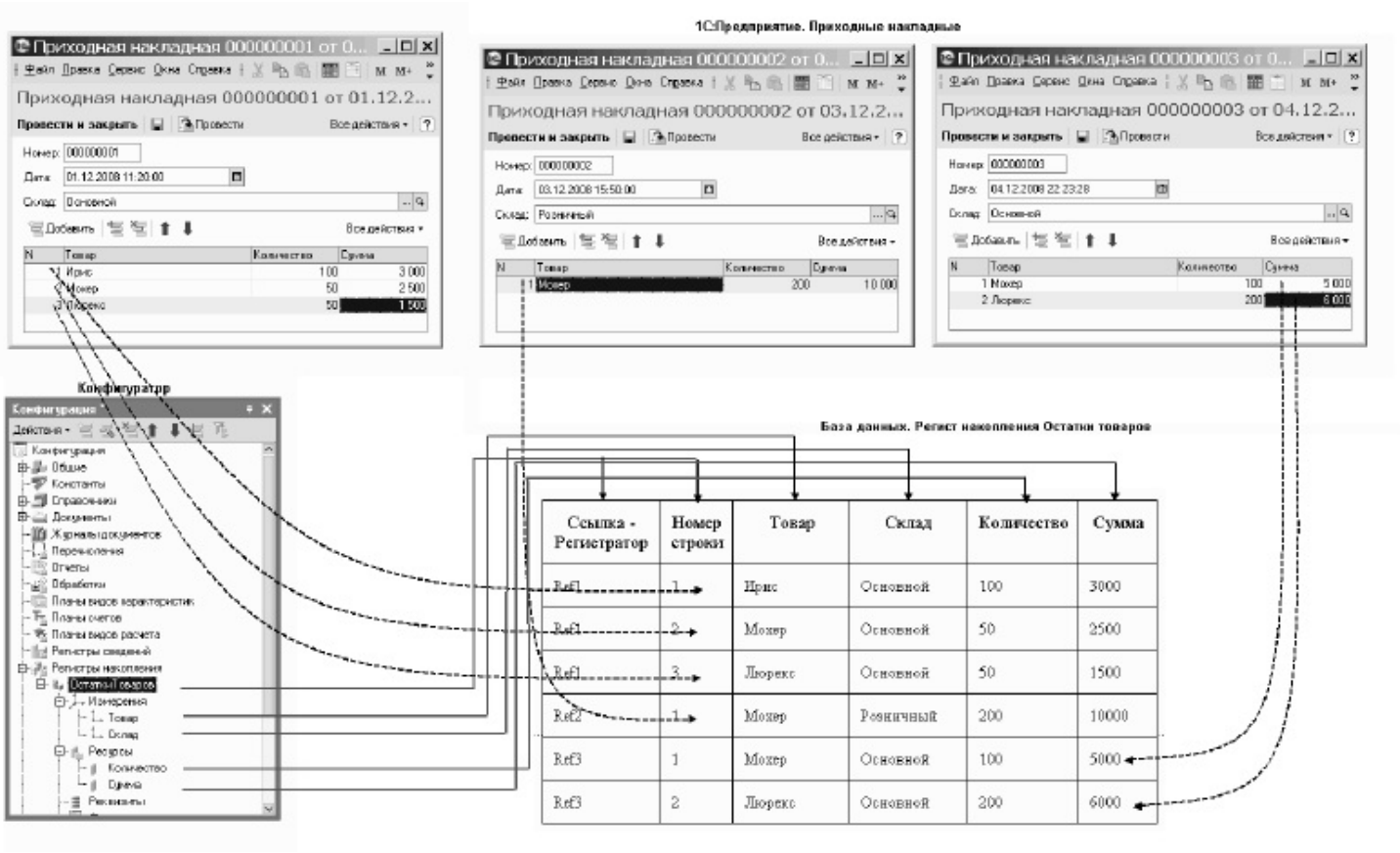


Рис. 6.2. Регистр накопления «Остатки товаров» в конфигураторе и в базе данных

На основании таблицы движений регистра накопления система рассчитывает таблицу итогов регистра, которая хранит в базе данных итоги на момент времени последнего движения (актуальные итоги).

Отличительной особенностью регистра накопления является то, что он не предназначен для интерактивного редактирования пользователем.

Разработчик может при необходимости предоставить пользователю возможность редактировать регистр накопления. Но предназначение регистра накопления заключается в том, чтобы его модификация производилась на основе алгоритмов работы других объектов базы данных, а не в результате непосредственных действий пользователя.

Основным назначением регистра накопления является накопление числовой информации в разрезе нескольких *измерений*, которые описываются разработчиком в соответствующем объекте конфигурации Регистр накопления и являются подчиненными объектами конфигурации.

Виды числовой информации, накапливаемой регистром накопления, называются *ресурсами,* также являются подчиненными объектами и описываются в конфигураторе.

Например, регистр накопления может накапливать информацию о количестве и сумме товаров на складах. В этом случае он будет иметь измерения *Товар* и *Склад* и ресурсы *Количество* и *Сумма* (см. рис. 6.2).

Изменение состояния регистра накопления происходит, как правило, при

проведении документа и заключается в том, что в регистр добавляется некоторое количество записей. Каждая запись содержит значения измерений, значения приращений ресурсов, ссылку на документ, который вызвал эти изменения (регистратор), и «направление» приращения (приход или расход). Такой набор записей называется *движениями* регистра накопления. Каждому движению регистра накопления всегда должен соответствовать *регистратор* – объект информационной базы (как правило, документ), который произвел эти движения.

Кроме этого, регистр накопления может хранить дополнительную информацию, описывающую каждое движение. Набор такой дополнительной информации задается разработчиком при помощи *реквизитов* объекта конфигурации *Регистр накопления*.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с регистрами накопления, можно прочитать в разделе [«Краткий справочник разработчика. Регистры накопления»](#)*.*

Добавление регистра накопления

Теперь, когда мы знаем, для чего предназначены регистры накопления,

посмотрим, как можно их использовать в нашем примере.

Прежде всего, нас интересует информация о том, сколько и каких материалов есть у нас на складах. Для накопления такой информации мы создадим регистр *ОстаткиМатериалов*.

В режиме «Конфигуратор»

Откроем в конфигураторе нашу учебную конфигурацию и добавим новый объект конфигурации *Регистр накопления*.

Для этого выделим в дереве объектов конфигурации ветвь *Регистры накопления* и нажмем кнопку *Добавить* в командной панели окна конфигурации.

В открывшемся окне редактирования объекта конфигурации на закладке *Основные* зададим имя регистра – *ОстаткиМатериалов*.

Также зададим и *Расширенное представление списка* как *Движения по регистру Остатки материалов*. Этот заголовок будет отображаться в окне списка записей регистра.

Нажмем *Далее* и перейдем на закладку *Подсистемы*.

По логике нашей конфигурации данный регистр должен быть доступен в разделах *Учет материалов*, *Оказание услуг* и *Бухгалтерия*. Поэтому отметим в списке подсистем эти подсистемы (рис. 6.3).

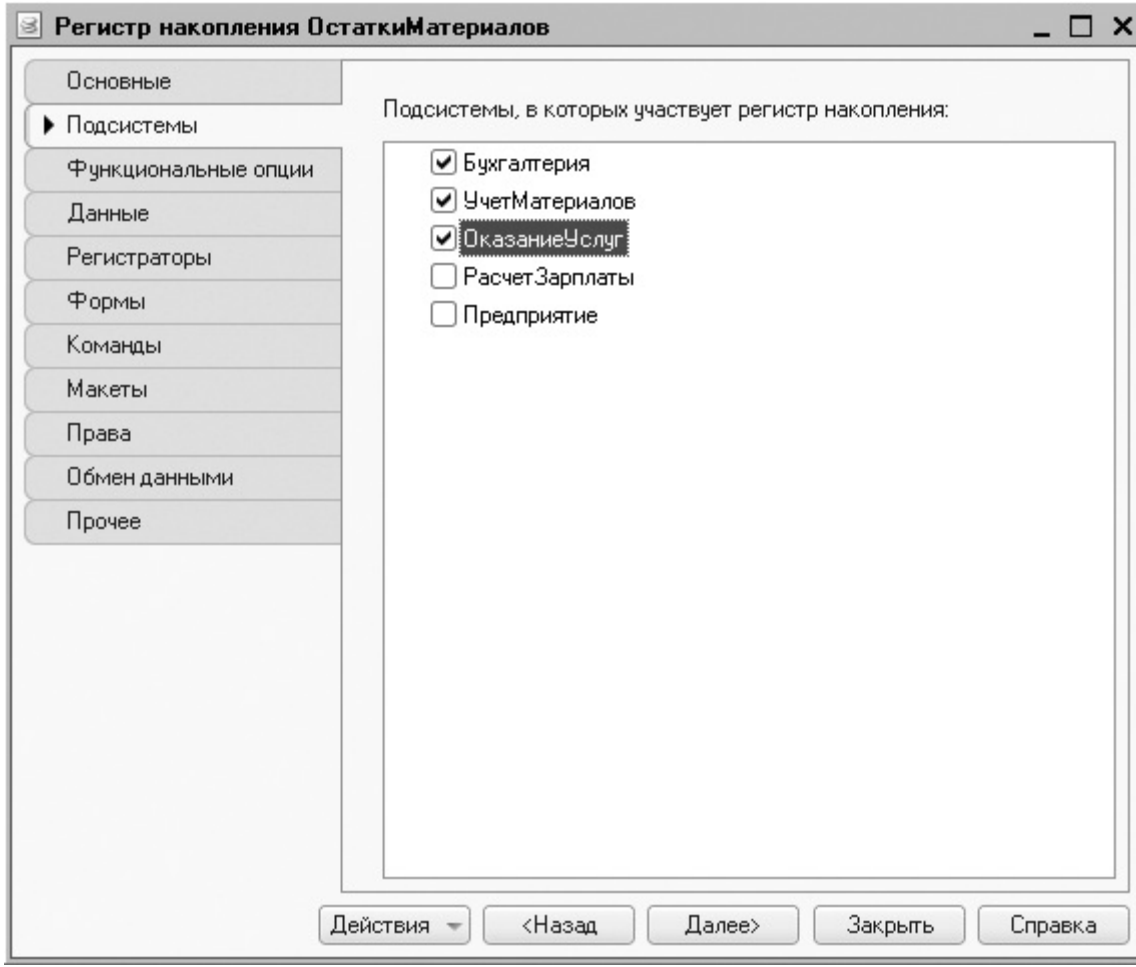


Рис. 6.3. Определение списка подсистем, в которых будет отражаться регистр

Выделим закладку *Данные* и перейдем к созданию структуры регистра.

Создадим измерения регистра:

- *Материал*, тип *СправочникСсылка.Номенклатура*;
- *Склад*, тип *СправочникСсылка.Склады*.

Для этого выделим ветвь *Измерения* и нажмем кнопку *Добавить* в командной панели окна (рис. 6.4).

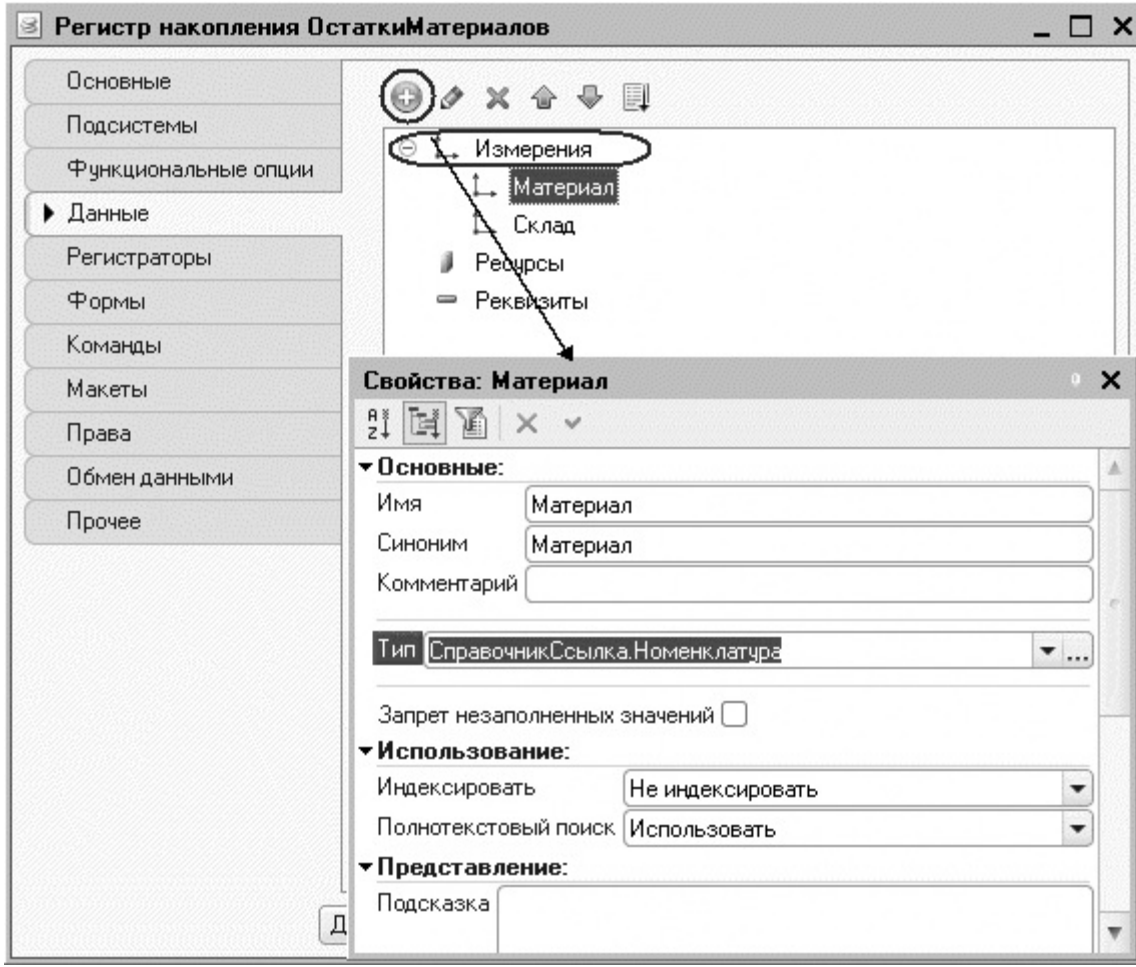


Рис. 6.4. Создание измерений регистра

Затем создадим ресурс *Количество* с длиной *15* и точностью *3*. Для этого

выделим ветвь *Ресурсы* и нажмем кнопку *Добавить* в командной панели окна (рис. 6.5).

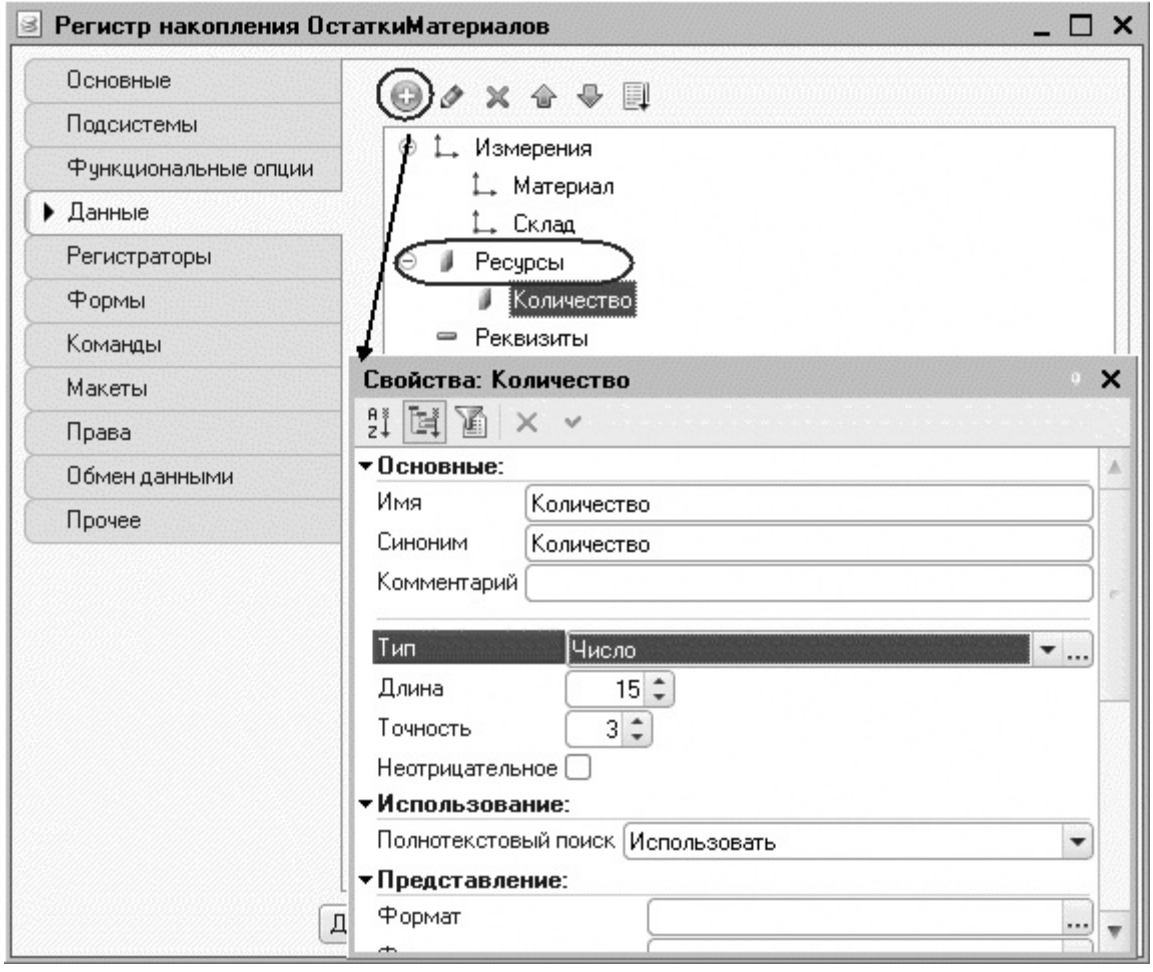


Рис. 6.5. Создание ресурсов регистра

В результате этих действий регистр *ОстаткиМатериалов* должен иметь следующий вид (рис. 6.6).

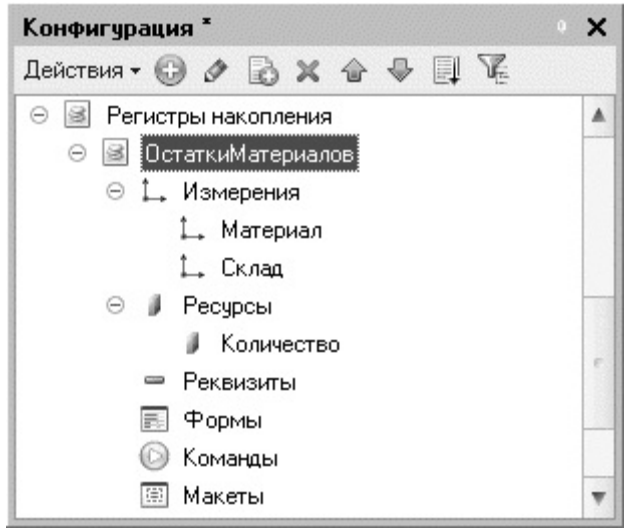


Рис. 6.6. Регистр «ОстаткиМатериалов»

Если вы сейчас попытаетесь запустить «1С:Предприятие» в режиме отладки, то система выдаст сообщение об ошибке: *«РегистрНакопления.ОстаткиМатериалов: Ни один из документов не является регистратором для регистра»*. Это сообщение еще раз подтверждает тот факт, что назначение регистра накопления в том, чтобы аккумулировать данные, поставляемые различными документами.

Поэтому мы сформируем движения регистра накопления *ОстаткиМатериалов* в процессе проведения двух созданных нами документов *ПриходнаяНакладная* и *ОказаниеУслуг*.

Движения документа

Движения документа – это записи в регистрах, которые создаются в процессе проведения документа и отражают изменения, производимые документом.

Откроем окно редактирования объекта конфигурации Документ *ПриходнаяНакладная*. Перейдем на закладку *Движения*, раскроем список *Регистры накопления* и отметим регистр накопления *ОстаткиМатериалов* (рис. 6.7).

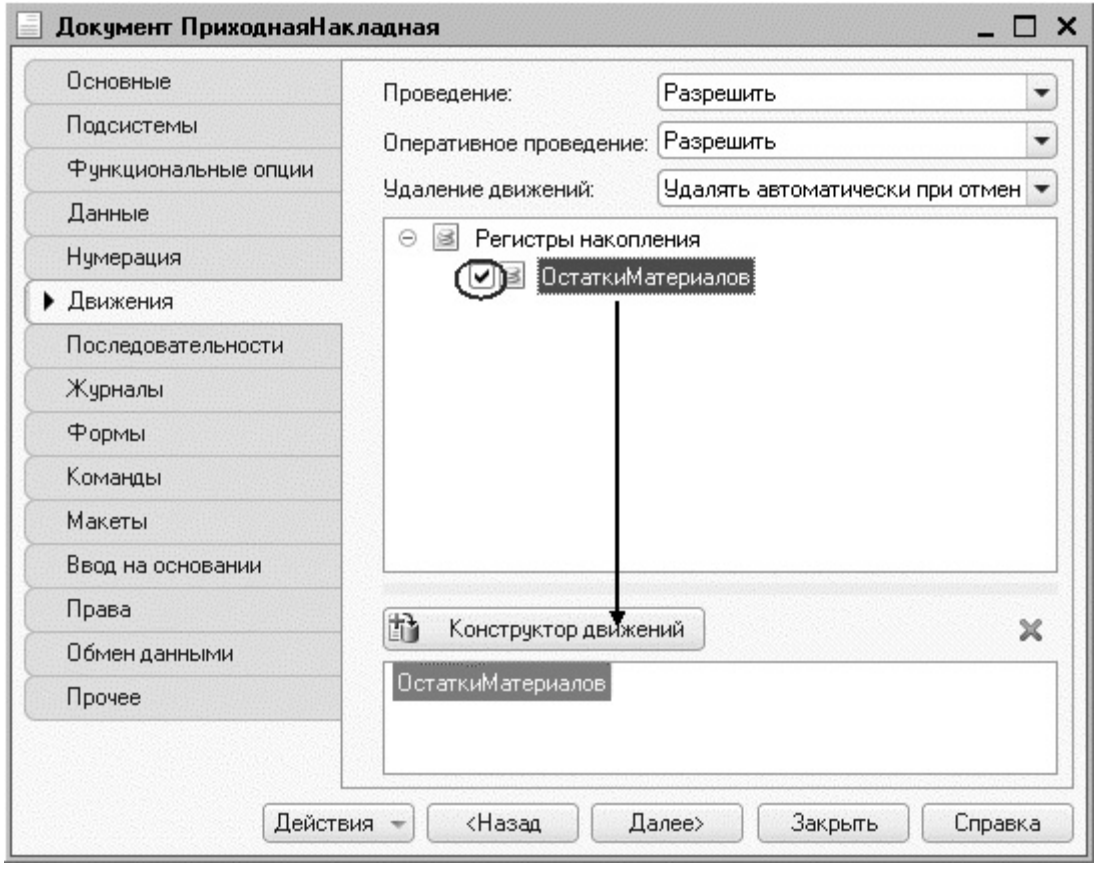


Рис. 6.7. Создание движений документа «ПриходнаяНакладная» в регистре «ОстаткиМатериалов»

Обратите внимание, что сразу после отметки выбранного регистра становится доступной кнопка *Конструктор движений*. Нажмем ее и воспользуемся этим конструктором.

Конструктор устроен просто. В списке *Регистры* перечислены регистры, в которых документ может создавать движения. В нашем случае там пока один регистр *ОстаткиМатериалов*.

В списке *Реквизиты документа* должны находиться исходные данные для создания движений – реквизиты документа *ПриходнаяНакладная*.

А в таблице *Поле – Выражение* должны быть заданы формулы, по которым будут вычисляться значения измерений и ресурсов регистра при записи движений (рис. 6.8).


Регистр, для которого конструируем движения (приход или расход)
Откуда берем данные
Конструктор движения регистров
Регистры
РегистрНакопления.ОстаткиМатериалов
Реквизиты документа
Дата
Номер
Склад
Тип движения регистра:
Приход
Расход
Табличная часть:
<Назад
Далее>
Поле
Выражение
Материал
Склад
Количество
Заполнить выражения
Очистить выражения
OK
Отмена
Справка
Что записываем в измерения и ресурсы регистра

Рис. 6.8. Конструктор движений

Обратите внимание, что по умолчанию конструктор предлагает нам создавать движения прихода (*Тип движения регистра – Приход,* символ + рядом с названием регистра) по регистру *ОстаткиМатериалов*. Это нас вполне устраивает, ведь документ *ПриходнаяНакладная* и должен приходовать материалы.

В поле выбора *Табличная часть* выберем табличную часть нашего документа – *Материалы*.

Список реквизитов документа, который уже заполнен реквизитами шапки документа, автоматически дополнится реквизитами нашей табличной части.

Теперь нажмем кнопку *Заполнить выражения*.

В нижнем окне сформируется соответствие полей (измерений и ресурсов) регистра и выражений для их расчета (рис. 6.9).

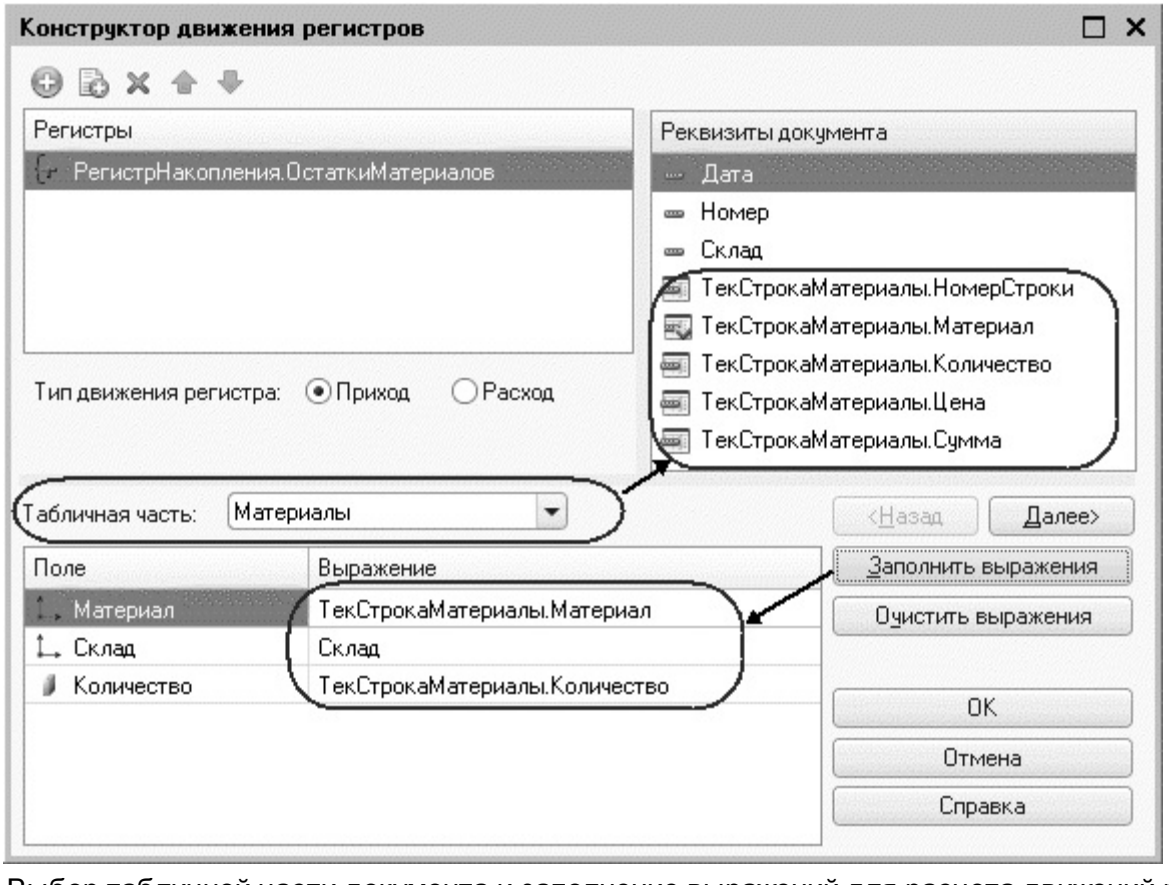


Рис. 6.9. Выбор табличной части документа и заполнение выражений для расчета движений регистра

Как видите, конструктор движений установил соответствия подходящим образом: в качестве материала в регистр будет записан материал из табличной части документа, в качестве склада – склад, указанный в шапке документа, а в

качестве количества – количество из табличной части документа.

Нажмем кнопку *OK* и посмотрим, какой текст сформировал конструктор в модуле документа *ПриходнаяНакладная* (листинг 6.1).

Листинг 6.1. Процедура «ОбработкаПроведения()»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
  // Данный фрагмент построен конструктором.
  // При повторном использовании конструктора внесенные вручную изменения будут
  утеряны!!!
  Движения.ОстаткиМатериалов.Записывать = Истина;

  Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
    // регистр ОстаткиМатериалов Приход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаМатериалы.Количество;
  КонецЦикла;
  //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

КонецПроцедуры
```

Конструктор создал обработчик события *ОбработкаПроведения* объекта конфигурации *Документ ПриходнаяНакладная*, поместил его в модуль объекта и открыл текст модуля.

Событие *ОбработкаПроведения* является одним из важнейших событий, связанных с документом. Это событие возникает при проведении документа. Основное назначение обработчика данного события – генерация движений по документу. Выполнение различных операций с данными в процедуре обработчика влияет на состояние учета. Таким образом, именно в эту процедуру разработчик должен поместить собственные алгоритмы преобразования данных, выполняемые в момент проведения документа.

Поясним текст процедуры обработчика.

Объект встроенного языка *ДокументОбъект* имеет свойство *Движения*. Оно возвращает объект *КоллекцияДвижений*, содержащий коллекцию наборов записей регистров, по которым этот документ может формировать движения.

К конкретному набору записей этой коллекции можно обратиться, указав через точку имя регистра, которому принадлежит этот набор записей. Например, *Движения.ОстаткиМатериалов*.

Затем через точку можно использовать различные методы набора записей

регистра, например, *Движения.ОстаткиМатериалов.Добавить()*. Метод *Добавить()* добавляет новую запись в набор записей.

В первой строке процедуры мы устанавливаем свойство *Записывать* набора записей регистра в значение *Истина*. То есть в явном виде указываем, что после завершения обработки проведения платформа должна будет записать этот набор записей в базу данных.

Внутри обработчика расположен цикл *Для Каждого … Из … Цикл*. Он предназначен для перебора строк табличной части нашего документа.

В цикле обращение к табличной части документа происходит по имени (*Материалы*). Переменная *ТекСтрокаМатериалы* содержит объект с данными текущей строки табличной части документа. Эта переменная создается в начале цикла и меняется по мере его прохождения.

В первой строке тела цикла, используя метод *Добавить()*, мы добавляем к набору записей, который создает наш документ в регистре, новую запись. Тем самым мы создаем объект *РегистрНакопленияЗапись* и сохраняем его в переменной *Движение*.

Используя этот объект, мы можем обратиться к полям этой записи, указав имя поля через точку от этой переменной (например, *Движение.Количество*).

Причем *Движение.Материал*, *Движение.Склад* – это измерения регистра, *Движение.Количество* – это ресурс регистра, а *Движение.ВидДвижения* и *Движение.Период* – стандартные реквизиты регистра, которые создаются автоматически.

Чтобы присвоить полям новой записи регистра соответствующие значения полей документа, мы обращаемся к полям табличной части, указав имя поля через точку от переменной *ТекСтрокаМатериалы* (например, *ТекСтрокаМатериалы.Материал*).

Заметим, что *Склад* – это реквизит шапки документа, а *Дата* – стандартный реквизит документа, который создается автоматически. Причем в цикле меняются только значения полей табличной части документа – *ТекСтрокаМатериалы.Материал* и *ТекСтрокаМатериалы.Количество*. Остальные поля не меняются, так как относятся к документу в целом и не зависят от текущей строки табличной части документа.

ВидДвиженияНакопления.Приход – это значение системного перечисления, которое определяет вид движения регистра накопления как *Приход*.

Таким образом, мы присваиваем нужные значения всем полям новой записи. После перебора всех строк документа (после завершения цикла) в этом наборе записей (*Движения.ОстаткиМатериалов*) будет содержаться столько

записей, сколько строк в табличной части проводимого документа.

Если мы теперь откроем окно редактирования объекта конфигурации Регистр накопления *ОстаткиМатериалов* и перейдем на закладку *Регистраторы*, то в списке документов, созданных в конфигурации, мы увидим отмеченный документ *ПриходнаяНакладная*, так как мы задали в модуле этого документа формирование движений в регистре *ОстаткиМатериалов* (рис. 6.10).

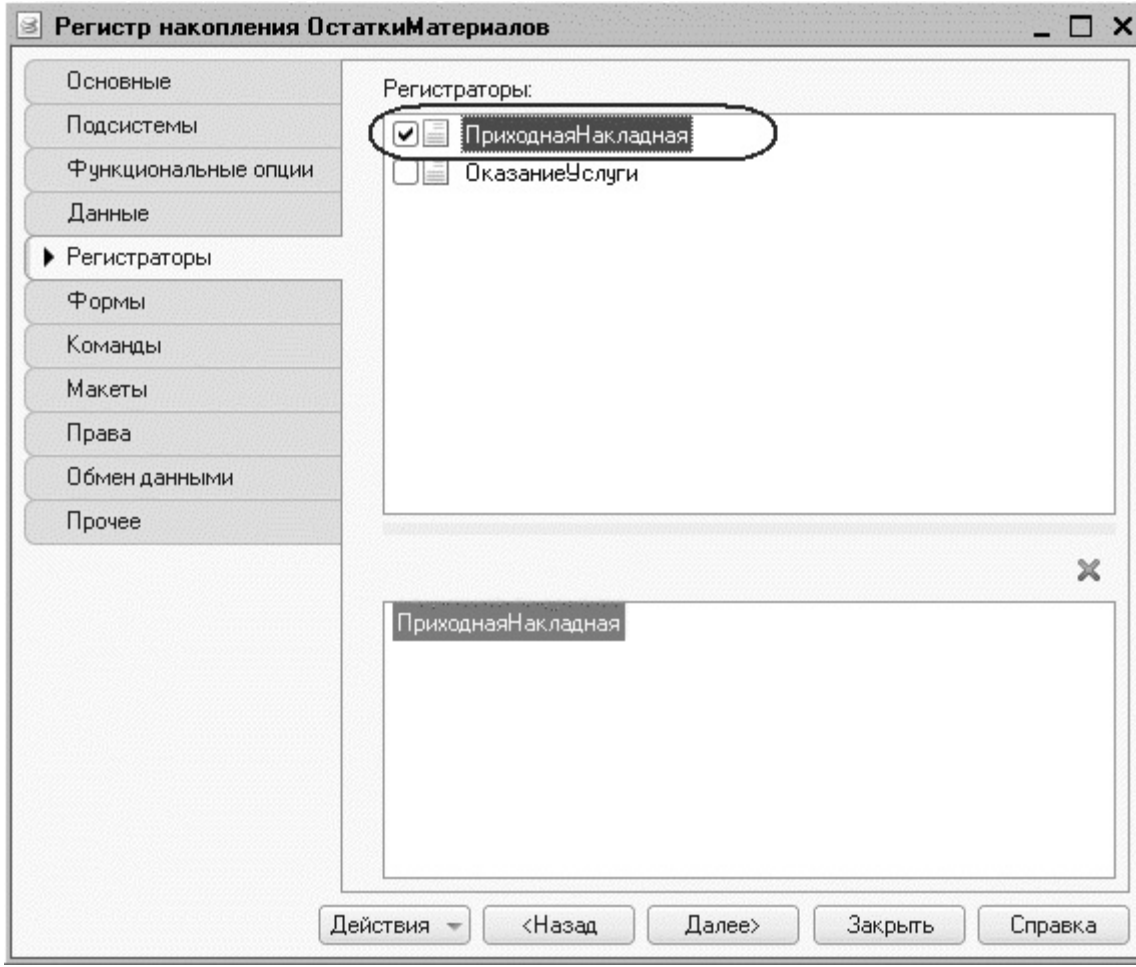


Рис. 6.10. Регистраторы регистра «ОстаткиМатериалов»

В заключение отредактируем командный интерфейс, чтобы в подсистемах

Бухгалтерия, *Оказание услуг* и *Учет материалов* была доступна ссылка для просмотра записей нашего регистра накопления.

Дело в том, что команды открытия регистров также добавляются в панель навигации подсистем, но по умолчанию они невидимы, в отличие от команд открытия справочников и документов.

В дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*. В открывшемся окне слева в списке *Подсистемы* выделим подсистему *УчетМатериалов*. Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель навигации.Обычное* включим видимость у команды *Остатки материалов* и мышью перетащим ее в группу *Панель навигации.См.также* (рис. 6.11).

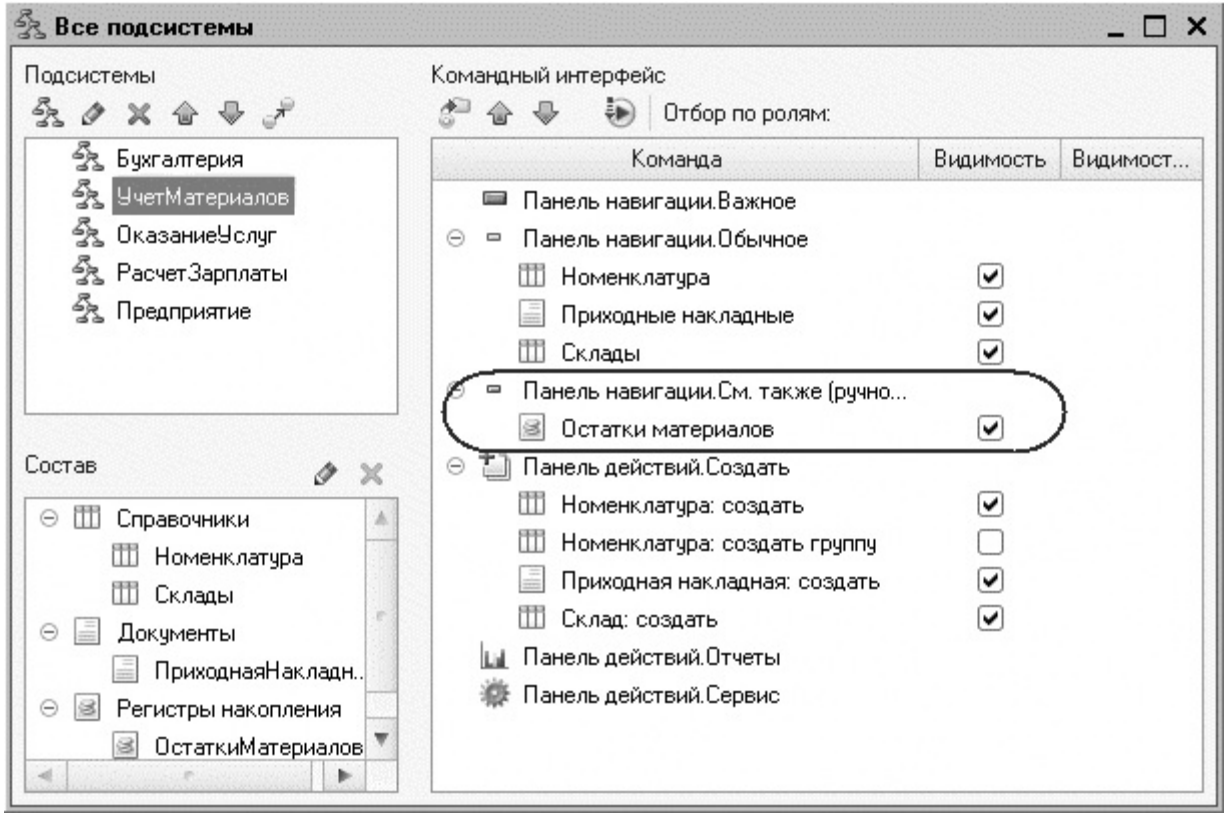


Рис. 6.11. Настройка командного интерфейса подсистем

Действительно, команды открытия регистров накопления не так часто используются, и поэтому лучше их перенести в группу *См.также* панели навигации разделов интерфейса.

Аналогично, выделив подсистемы *ОказаниеУслуг* и *Бухгалтерия*, в панели навигации в группе *Обычное* включим видимость у команды *Остатки материалов* и перенесем ее в группу *См.также*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и протестируем внесенные нами изменения.

В открывшемся окне «1С:Предприятия» мы видим, что в панели навигации в группе *См.также* разделов *Бухгалтерия*, *Оказание услуг* и *Учет материалов* появилась команда для открытия списка регистра *Остатки материалов* (рис. 6.12).

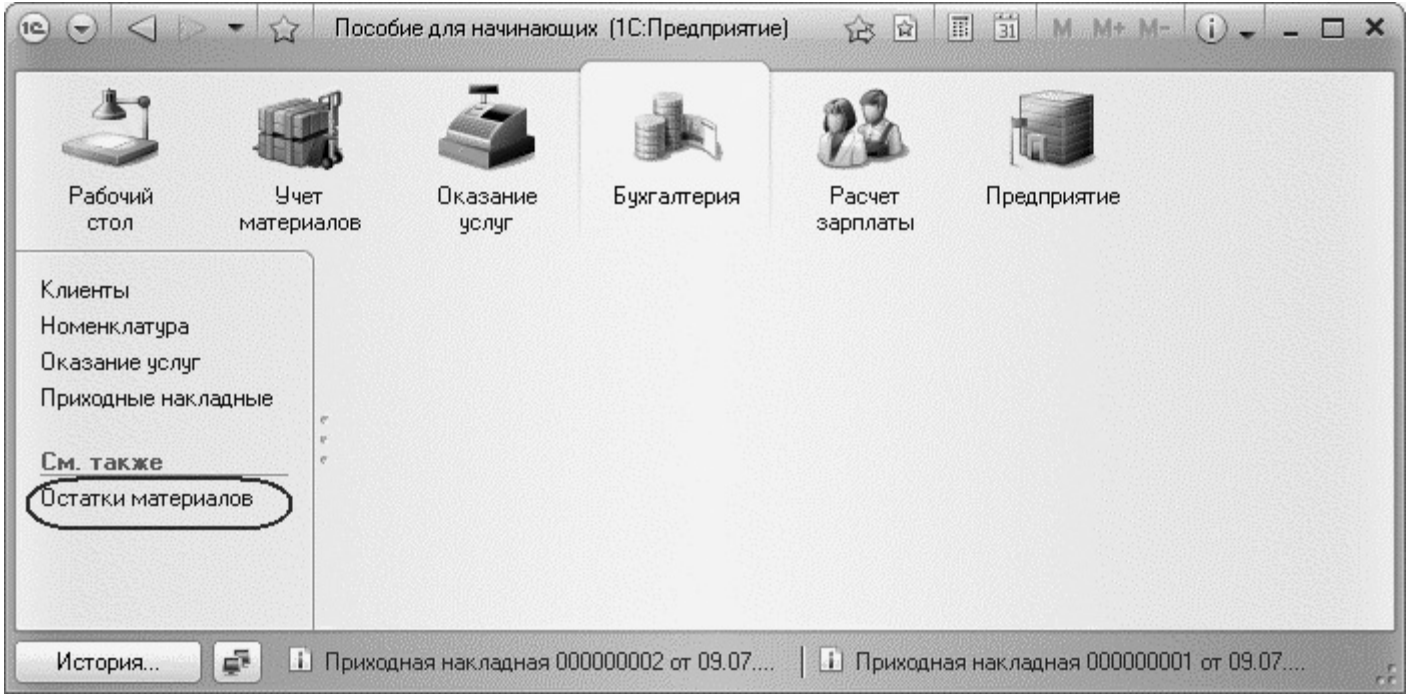


Рис. 6.12. Список регистра накопления «ОстаткиМатериалов»

Чтобы проследить связь между проведением документа и накоплением информации в регистре, откроем список приходных накладных, выполнив команду *Приходные накладные* разделе *Бухгалтерия*.

Откроем *Приходную накладную № 1* и нажмем *Провести и закрыть*, то есть перепроведем ее. То же самое сделаем для *Приходной накладной № 2*.

Перепровести документы можно и не открывая документов. Для этого нужно выделить нужный документ в списке (или выделить мышью группу документов, удерживая клавишу *Ctrl*), нажать кнопку *Все действия* в командной панели формы списка и выбрать пункт *Провести* (рис. 6.13).

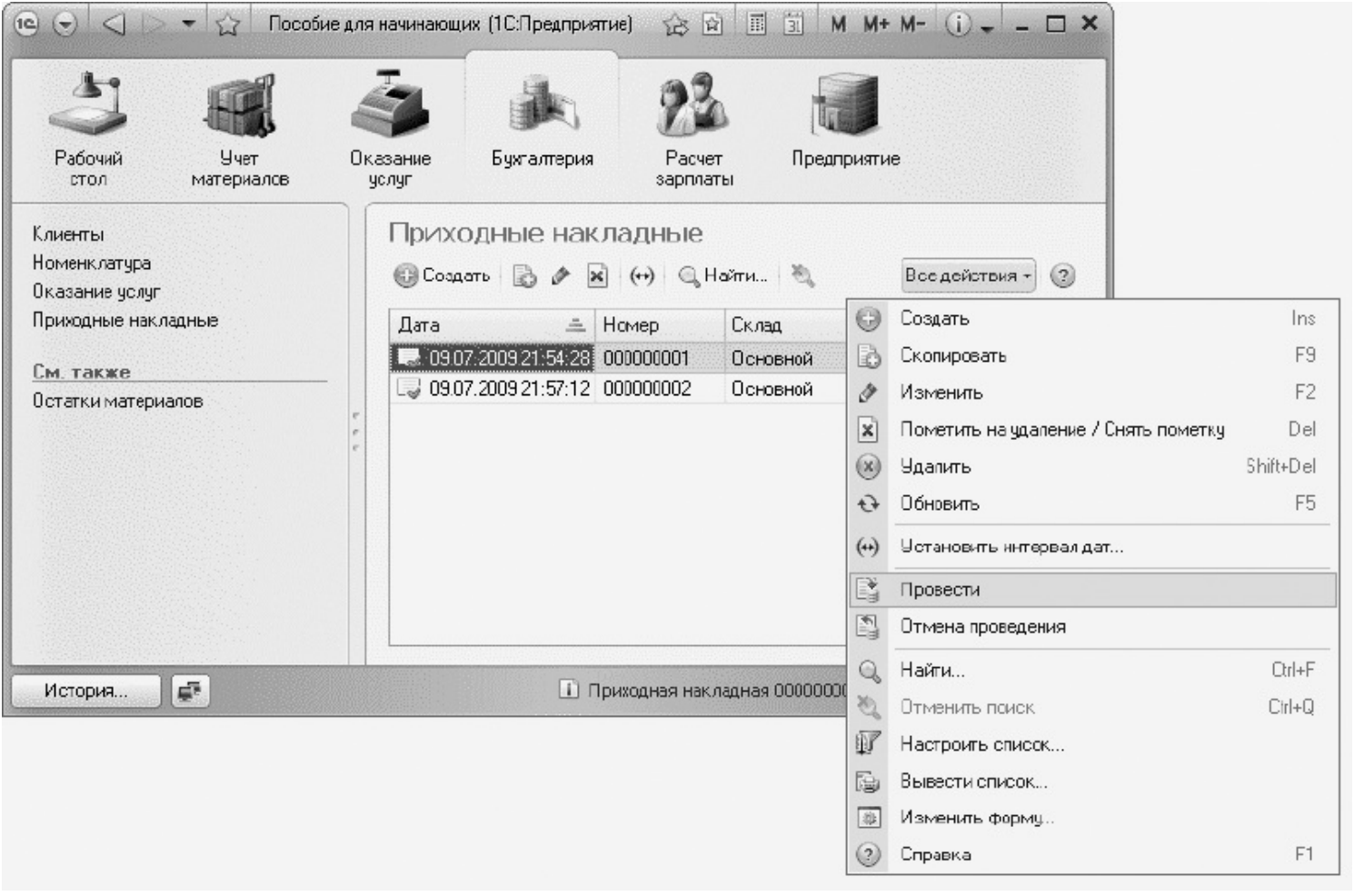


Рис. 6.13. Проведение документа

Теперь выполним команду *Остатки материалов* и откроем список нашего

регистра накопления (рис. 6.14).

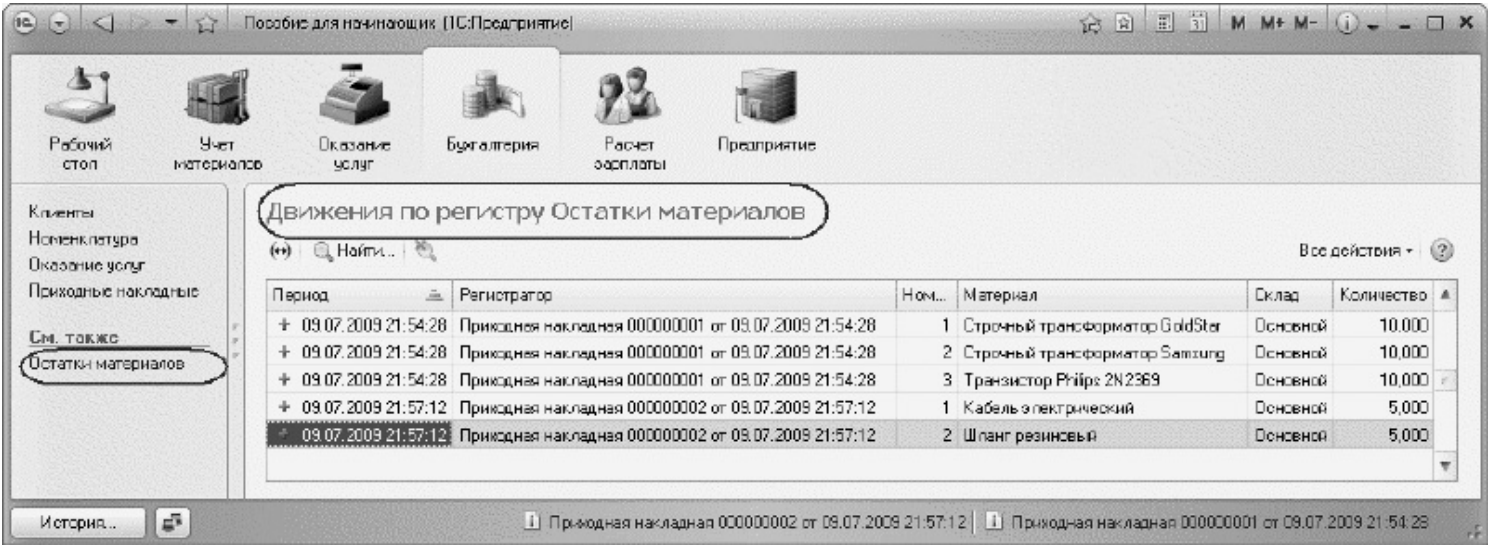


Рис. 6.14. Список регистра накопления «ОстаткиМатериалов»

Мы видим, что при проведении приходных накладных появляются соответствующие записи в регистре накопления *Остатки материалов*. Обратите внимание, что добавилось пять записей – первые три после проведения первого документа, что соответствует количеству строк в его табличной части, и последние две после проведения второго документа.

Все поля регистра заполнились данными документов так, как мы задали в обработчике проведения документа *ПриходнаяНакладная*. Пиктограмма со

знаком + слева от каждой записи указывает на тип движения – *Приход*.

Как мы видим, заголовок формы списка записей регистра соответствует заданному нами в свойстве *Расширенное представление списка* для этого регистра.

Команда перехода к движениям в форме документа

В режиме «Конфигуратор»

При реальной работе записей в регистре *ОстаткиМатериалов* будет много, и будет трудно понять, какие записи относятся к определенному документу.

Поэтому наряду с общим списком регистра хотелось бы иметь возможность вызывать из формы документа список регистра, в котором показаны движения, произведенные только этим документом.

Чтобы реализовать такую возможность, вернемся в конфигуратор и откроем форму документа *ПриходнаяНакладная*.

В левом верхнем окне перейдем на закладку *Командный интерфейс*.

В разделе *Панель навигации* раскроем группу *Перейти* и увидим команду для

открытия списка регистра накопления *Остатки материалов*. Эта команда была автоматически помещена в панель навигации формы документа, так как он является регистратором, то есть создает движения в нашем регистре.

Установим свойство *Видимость* для этой команды (рис. 6.15).

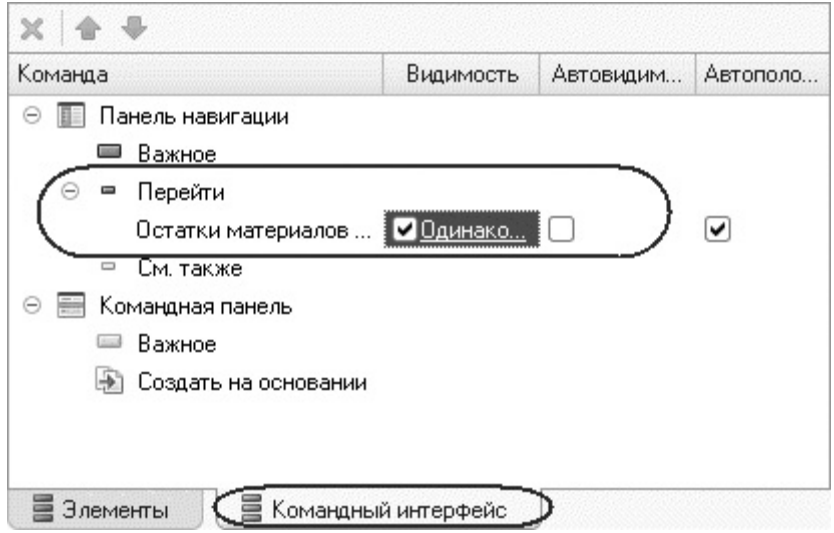


Рис. 6.15. Настройка командного интерфейса формы документа

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем *Приходную накладную № 2* (рис. 6.16).

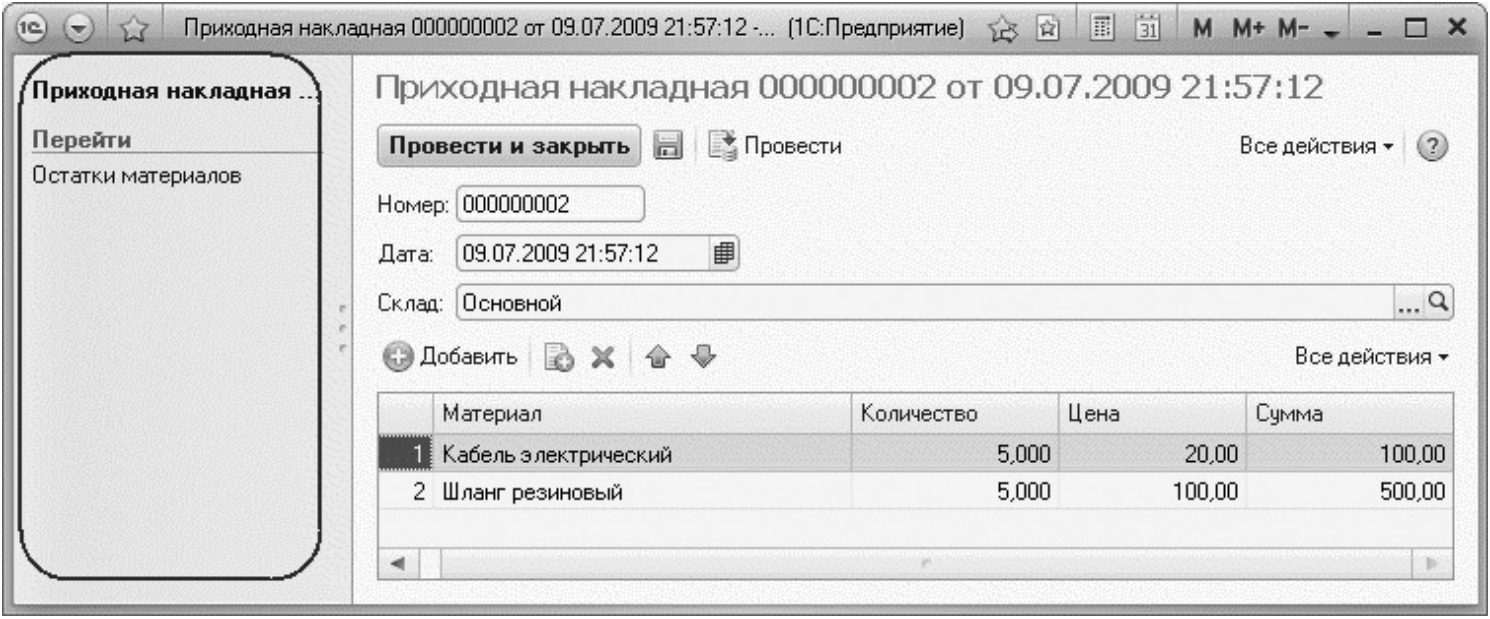


Рис. 6.16. Панель навигации документа «Приходная накладная»

В форме документа появилась панель навигации, в которой мы можем переходить к списку записей регистра *Остатки Материалов*, связанному с документом (рис. 6.17), и обратно к содержимому документа.

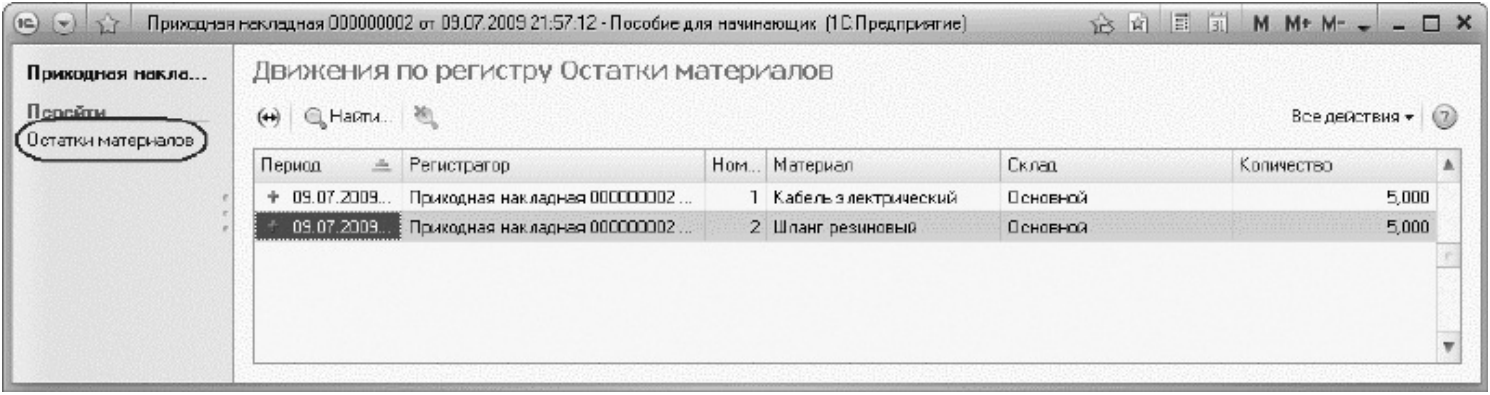


Рис. 6.17. Переход к регистру накопления из формы документа

Обратите внимание, что до этого панель навигации в форме приходной накладной была не видна, так как в ней не было отображено ни одной команды.

Движения документа «Оказание услуги»

Теперь мы аналогичным образом создадим движения документа *ОказаниеУслуги*. Для этого потребуется выполнить уже знакомые нам действия.

В режиме «Конфигуратор»

Откроем окно редактирования объекта конфигурации *Документ ОказаниеУслуги*.

Перейдем на закладку *Движения* и в списке регистров конфигурации отметим регистр накопления *ОстаткиМатериалов*.

Нажмем кнопку *Конструктор движений*.

В открывшемся окне конструктора изменим тип движения регистра на *Расход*, так как документ *ОказаниеУслуги* должен расходовать материалы. Пиктограмма слева от названия регистра изменится на знак –. В поле выбора *Табличная часть* выберем табличную часть нашего документа – *ПеречеьНоменклатуры*.

Список реквизитов документа, который уже заполнен реквизитами шапки документа, автоматически дополнится реквизитами нашей табличной части.

Теперь нажмем кнопку *Заполнить выражения*.

В нижнем окне сформируется соответствие полей (измерений и ресурсов) регистра и выражений для их расчета. Однако при автоматическом заполнении поле *Материал* не заполнится.

Так происходит потому, что имя поля табличной части – *Номенклатура* не совпадает с именем измерения регистра – *Материал*. Если мы оставим это так, как есть, то в регистре накопления в строках с типом *Движение регистра*

– *расход* номенклатура фиксироваться не будет.

Чтобы избежать этого, нужно выделить поле регистра *Материал* и в окне *Реквизиты документа* дважды щелкнуть по строке *ТекСтрокаПеречеьНомеклатуры.Номенклатура*.

Таким образом, номенклатура для движений регистра накопления будет выбираться из табличной части документа (рис. 6.18).

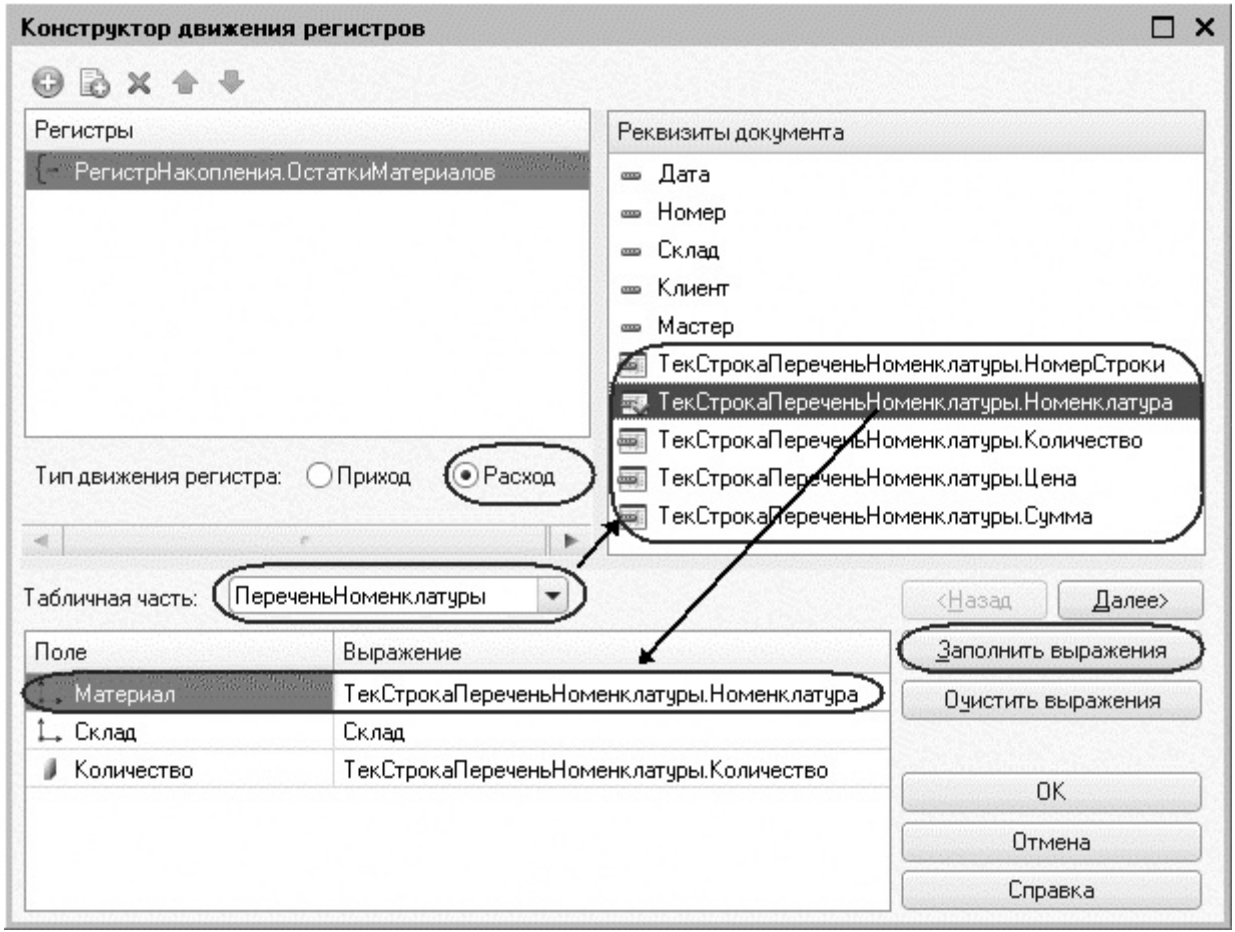


Рис. 6.18. Выбор табличной части документа и заполнение выражений для расчета движений регистра

Нажмем кнопку *OK*.

Конструктор создал обработчик события *ОбработкаПроведения* объекта конфигурации *Документ ОказаниеУслуги* и поместил его в модуль объекта (листинг 6.2).

Листинг 6.2. Процедура «ОбработкаПроведения()»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
  // Данный фрагмент построен конструктором.
  // При повторном использовании конструктора внесенные вручную изменения будут
  утеряны!!!
  Движения.ОстаткиМатериалов.Записывать = Истина;

  Для Каждого ТекСтрокаПеречеНьНоменклатуры Из ПеречеНьНоменклатуры Цикл
    // регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаПеречеНьНоменклатуры.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаПеречеНьНоменклатуры.Количество;
  КонецЦикла;
  //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

КонецПроцедуры
```

Обратите внимание, что строка *Движение.ВидДвижения* =

ВидДвиженияНакопления.Расход определяет вид движения регистра накопления, производимый этим документом как *Расход*, а в остальном процедура обработчика документа *ОказаниеУслуги* идентична обработчику документа *ПриходнаяНакладная* (см. листинг 6.1), подробно разобранному нами ранее.

В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра *Остатки Материалов*, связанному с документом.

Для этого откроем форму документа *ОказаниеУслуги*.

В левом верхнем окне перейдем на закладку *Командный интерфейс*.

В разделе *Панель навигации* раскроем группу *Перейти* и установим видимость для команды открытия регистра накопления *Остатки материалов*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и в разделе *Оказание услуг* откроем документ *Оказание услуги № 1* и нажмем *Провести и закрыть*, то есть перепроведем его.

Теперь выполним команду *Остатки материалов* и откроем список нашего регистра накопления (рис. 6.19).

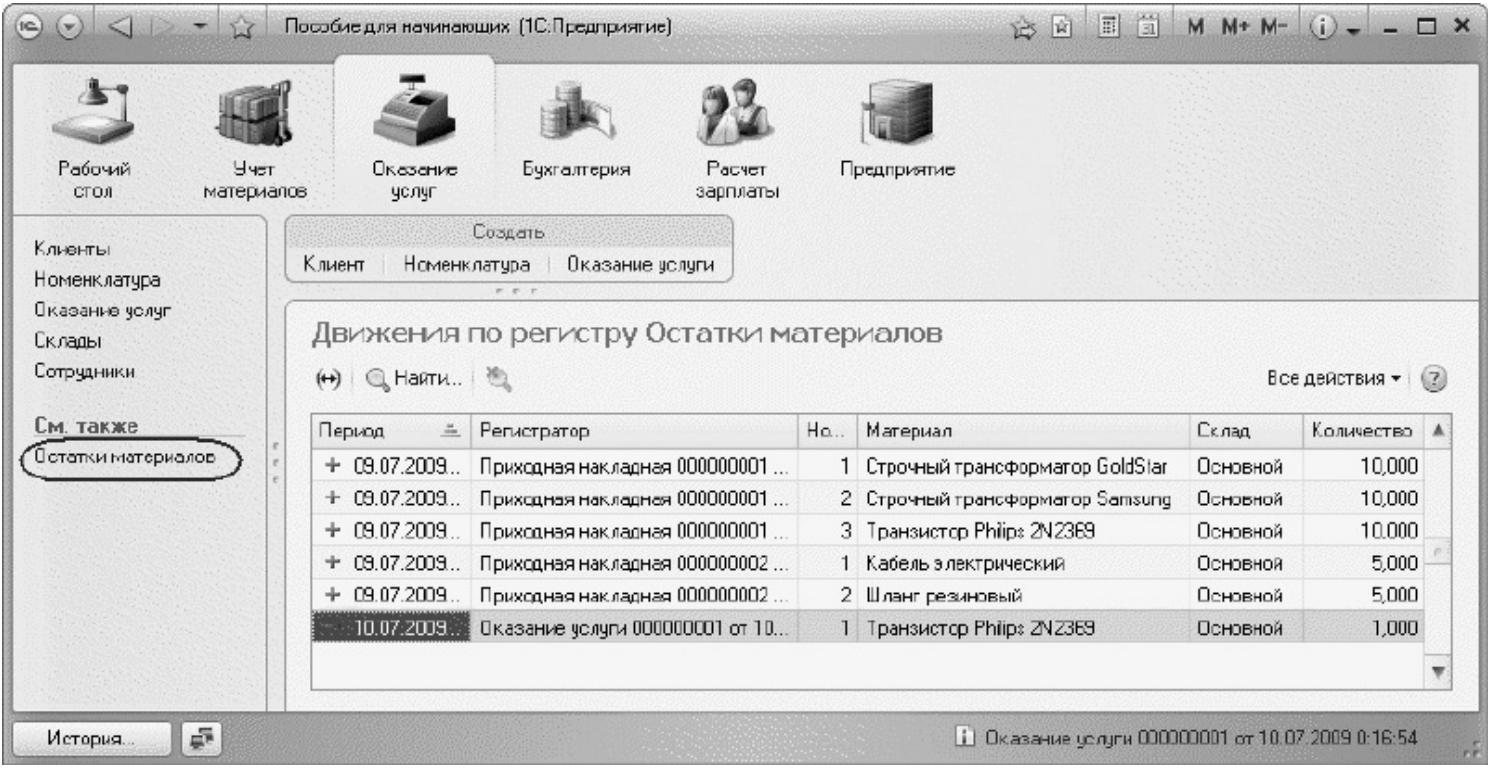


Рис. 6.19. Список регистра накопления «ОстаткиМатериалов»

Мы видим, что в регистре накопления *Остатки материалов* появилась еще одна запись, что соответствует количеству строк в табличной части

проведенного документа.

Все поля регистра заполнились данными документа так, как мы задали в обработчике проведения документа *Оказание услуги*.

Пиктограмма со знаком минус слева от записи указывает на тип движения – *Расход* (см. рис. 6.19).

Сейчас мы видим весь список движений регистра. Открыв этот список из формы документа, мы можем отфильтровать движения по документу-регистратору.

Для этого откроем еще раз документ *Оказание услуги № 1*.

В форме документа появилась панель навигации, в которой мы можем переходить к списку записей регистра *Остатки материалов*, связанному с документом, и обратно к содержимому документа (рис. 6.20).

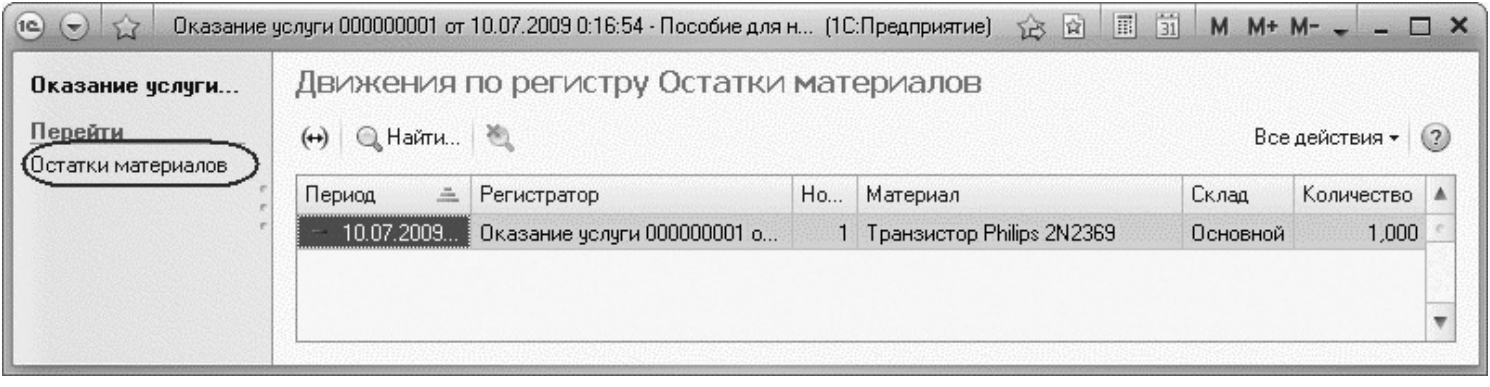


Рис. 6.20. Переход к регистру накопления из формы документа

Сформированные таким образом движения этого документа будут не совсем правильны.

Дело в том, что в документе *Оказание услуги*, в отличие от документа *Приходная накладная,* могут содержаться не только расходуемые материалы, но и услуги. Поэтому в регистр *Остатки материалов* будут попадать записи и о расходуемых услугах, что неправильно.

Пока мы ничего не будем делать с движениями, которые сформировал конструктор, но как только познакомимся с перечислениями, мы внесем в обработчик проведения необходимые изменения.

«Теория». Способы работы с коллекцией

В процессе формирования движений документов, когда в цикле обходили табличные части документов *ПриходнаяНакладная* и *ОказаниеУслуги*, мы столкнулись с одним из объектов встроенного языка, который является коллекцией.

Многие объекты встроенного языка являются коллекциями. Коллекция представляет собой совокупность объектов. Существуют общие принципы работы с любой коллекцией.

Во-первых, доступ к каждому объекту коллекции возможен путем перебора элементов коллекции в цикле. Для этого используется конструкция языка *Для Каждого Из … Цикл …* (листинг 6.3).

Листинг 6.3. Перебор элементов коллекции в цикле

```
Для Каждого СтрокаТабличнойЧасти Из ТабличнаяЧасть Цикл
  Сообщить(СтрокаТабличнойЧасти.Услуга);

КонецЦикла;
```

В этом примере *ТабличнаяЧасть* – это коллекция строк табличной части объекта конфигурации. При каждом проходе цикла в переменной *СтрокаТабличнойЧасти* будет содержаться очередная строка из этой коллекции.

Во-вторых, существует доступ напрямую к элементу коллекции, без перебора коллекции в цикле. Здесь возможны различные комбинации двух обращений.

1. Во встроенном языке бывают именованные коллекции. То есть коллекции, в которых каждый элемент имеет некоторое уникальное имя. В этом случае обращение к элементу коллекции возможно по этому имени (листинг 6.4).

Листинг 6.4. Обращение к элементу коллекции

```
Справочники.Сотрудники;
Справочники["Сотрудники"];
```

В этом примере *Справочники* – это коллекция менеджеров всех справочников, содержащихся в конфигурации. Так как каждый справочник конфигурации имеет свое уникальное имя, то к конкретному элементу этой коллекции (к менеджеру конкретного справочника) можно обратиться, указав имя этого справочника: *Справочники["Сотрудники"]*.

2. Если нет смысла в «персонификации» элементов коллекции (коллекция неименованная), тогда обращение к элементу коллекции возможно по индексу (индекс первого элемента коллекции – ноль), листинг 6.5.

Листинг 6.5. Обращение к элементу коллекции по индексу

```
ТабличнаяЧасть[0];
```

В этом примере *ТабличнаяЧасть* – это коллекция строк табличной части объекта конфигурации. И мы обращаемся к первому элементу этой коллекции, указывая его индекс – 0.

Следует отметить, что существуют коллекции, сочетающие оба вида обращений. Например, к коллекции колонок таблицы значений можно обращаться как по именам колонок, так и по индексу.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Регистр накопления».*
- *Почему следует использовать регистры, хотя необходимая информация содержится в других объектах.*
- *Для чего нужны измерения регистра, ресурсы и реквизиты.*
- *Что такое движения регистра и что такое регистратор.*
- *Как создать новый регистр накопления и описать его структуру.*
- *Как создать движения документа с помощью конструктора движений.*
- *Как средствами встроенного языка обойти табличную часть документа и обратиться к ее данным.*

- *Как показать команды открытия списка регистра в интерфейсе конфигурации и в интерфейсе формы.*

Занятие 7 (0:25). Простой отчет

Продолжительность

Ориентировочная продолжительность занятия – 25 минут.

На этом занятии мы познакомимся с вами с объектом конфигурации *Отчет*. Вы узнаете, для чего он используется, и создадите отчет, который будет показывать движения и остатки материалов на нашем предприятии.

Данное занятие преследует цель лишь проиллюстрировать на простом примере механизм создания отчетов. Поэтому здесь мы не будем углубляться в систему компоновки данных, с помощью которой строится любой отчет. Более подробно этот вопрос будет рассмотрен в занятии «Отчеты».

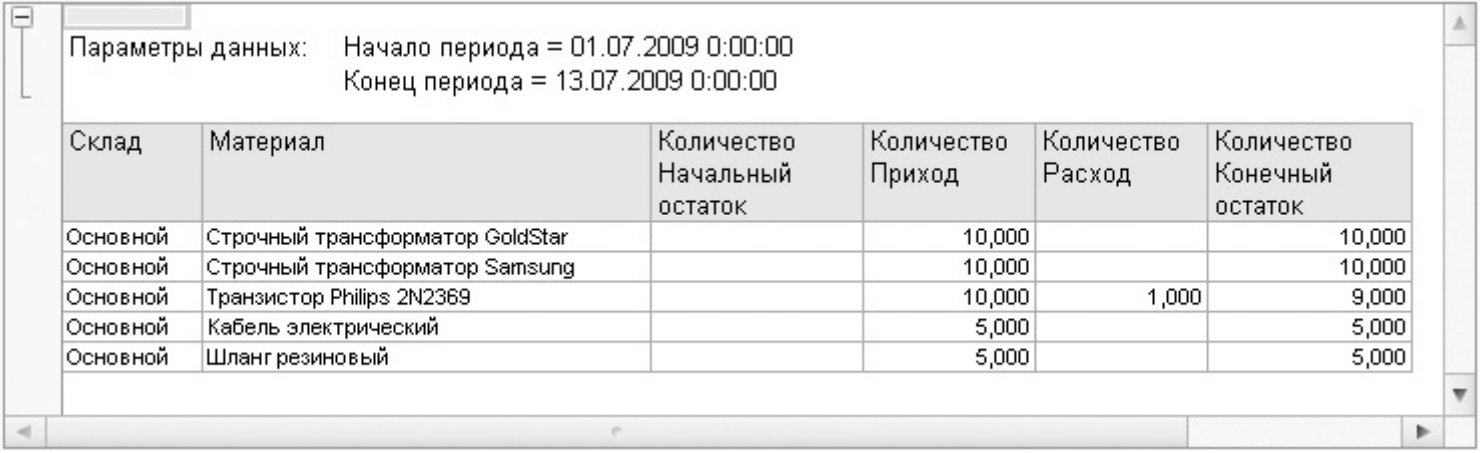
Что такое отчет

Объект конфигурации *Отчет* предназначен для описания алгоритмов, при помощи которых пользователь сможет получать необходимые ему выходные данные. Алгоритм формирования выходных данных описывается при помощи визуальных средств или с использованием встроенного языка. В реальной жизни объектам конфигурации *Отчет* соответствуют всевозможные таблицы выходных данных, сводных данных, диаграммы и пр.

Добавление отчета

В режиме «Конфигуратор»

Теперь у нас все готово для того, чтобы можно было получать выходные данные. Поэтому приступим к созданию отчета, который будет показывать нам приход, расход и остатки материалов (рис. 7.1).



Параметры данных: Начало периода = 01.07.2009 0:00:00
Конец периода = 13.07.2009 0:00:00

Склад	Материал	Количество Начальный остаток	Количество Приход	Количество Расход	Количество Конечный остаток
Основной	Строчный трансформатор GoldStar		10,000		10,000
Основной	Строчный трансформатор Samsung		10,000		10,000
Основной	Транзистор Philips 2N2369		10,000	1,000	9,000
Основной	Кабель электрический		5,000		5,000
Основной	Шланг резиновый		5,000		5,000

Рис. 7.1. Результат отчета

На этом примере мы покажем, как быстро и легко разработать отчет с использованием только визуальных средств разработки «без единой строчки кода».

Откроем в конфигураторе нашу учебную конфигурацию и добавим новый объект конфигурации *Отчет*.

Для этого выделим в дереве объектов конфигурации ветвь *Отчеты* и нажмем кнопку *Добавить* в командной панели окна конфигурации (рис. 7.2).

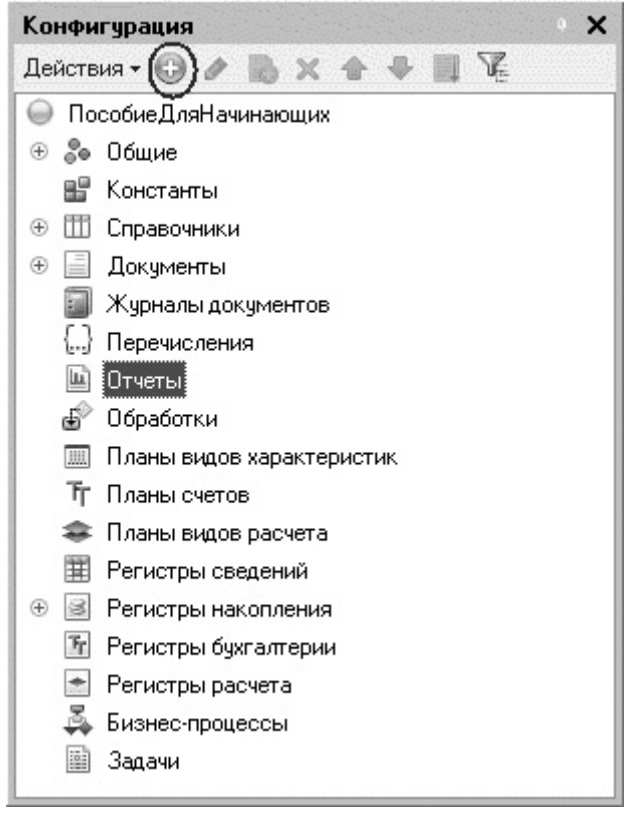


Рис. 7.2. Создание нового объекта конфигурации «Отчет»

В открывшемся окне редактирования объекта конфигурации на закладке *Основные* зададим имя отчета – *Материалы*.

Больше никаких свойств, определяющих представление объекта в интерфейсе приложения, задавать не будем. Вместо них будет использоваться *Синоним* объекта, который создается автоматически на основании имени объекта.

Создадим основу для построения любого отчета – *схему компоновки данных*. Для этого нажмем кнопку *Открыть схему компоновки данных* или кнопку открытия  со значком лупы (рис. 7.3).

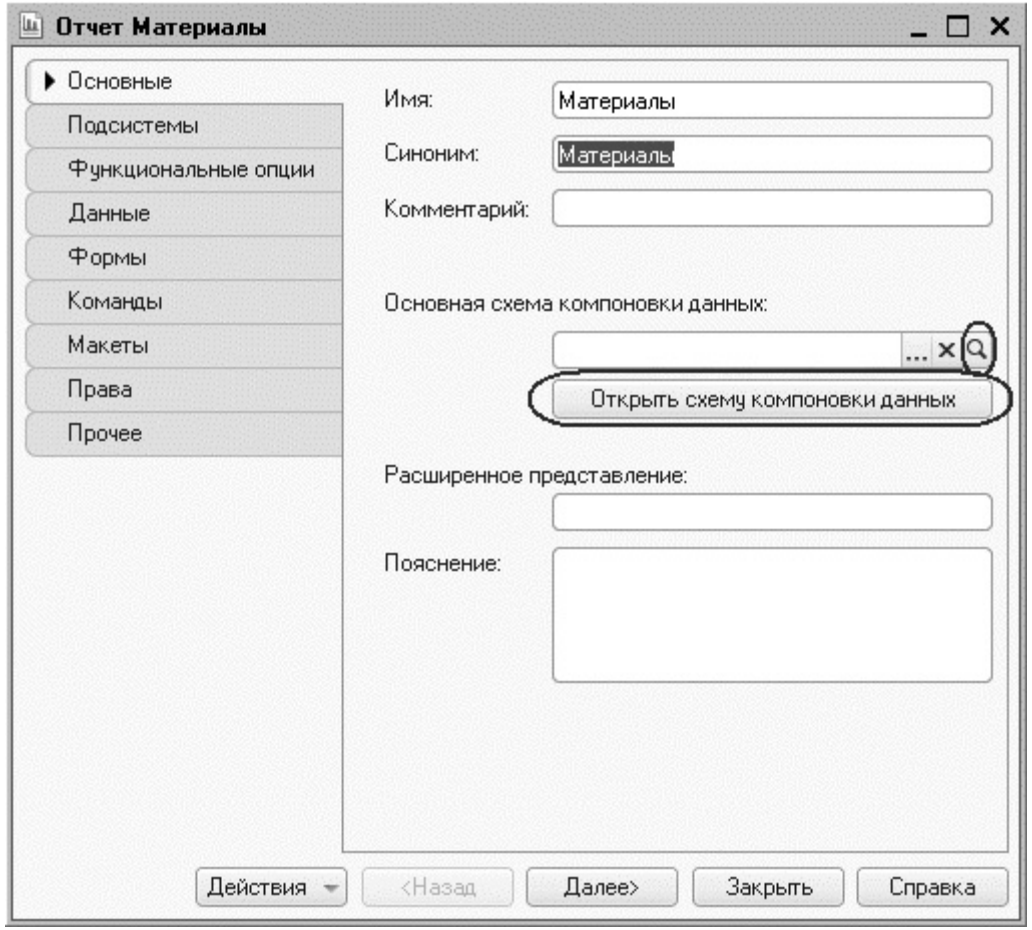


Рис. 7.3. Создание схемы компоновки данных отчета

Макет

Так как у отчета, который мы создаем, еще не существует схемы компоновки

данных, платформа предложит создать новую схему. Схема компоновки данных с точки зрения конфигурации является макетом, поэтому будет открыт конструктор макета, предлагающий выбрать единственный тип макета – *Схема компоновки данных* (рис. 7.4).

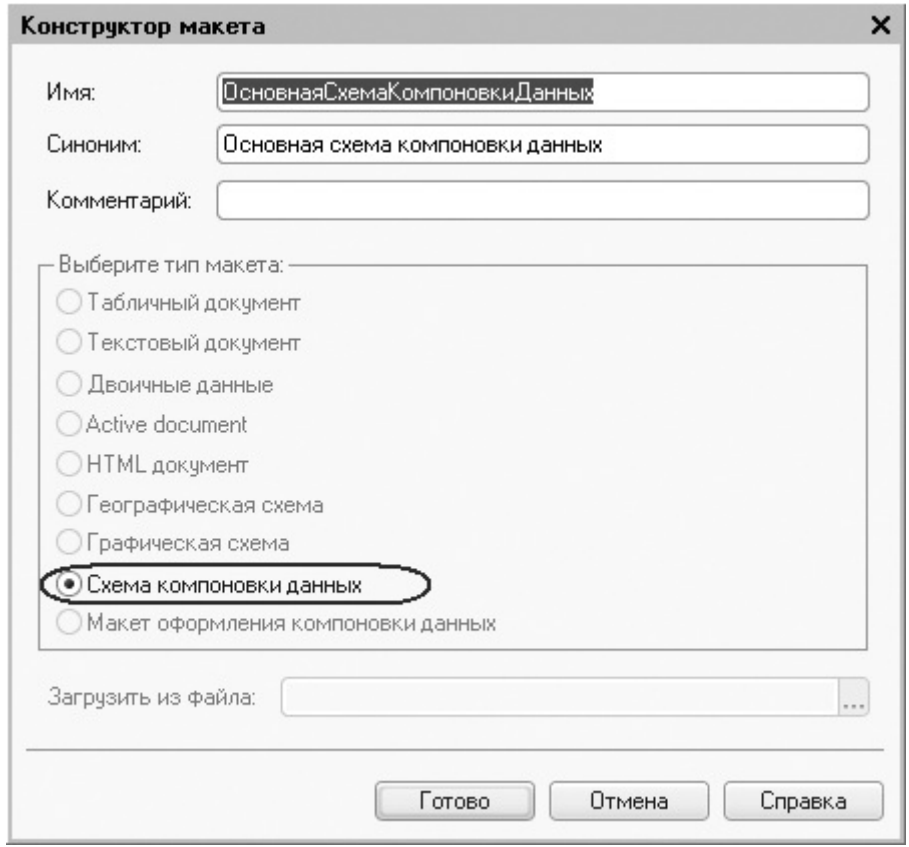


Рис. 7.4. Создание схемы компоновки данных отчета

Нажмем кнопку *Готово*.

Схема компоновки данных

Платформа создаст новый макет, содержащий схему компоновки данных, и сразу же откроет конструктор схемы компоновки данных.

Конструктор обладает большим количеством возможностей для визуального проектирования отчетов, но мы сейчас воспользуемся только самыми простыми его возможностями и определим те данные, которые хотим видеть в результате работы нашего отчета.

Набор данных

Добавим новый *набор данных – запрос*. Для этого нажмем кнопку *Добавить* или вызовем контекстное меню ветки *Наборы данных* (рис. 7.5).

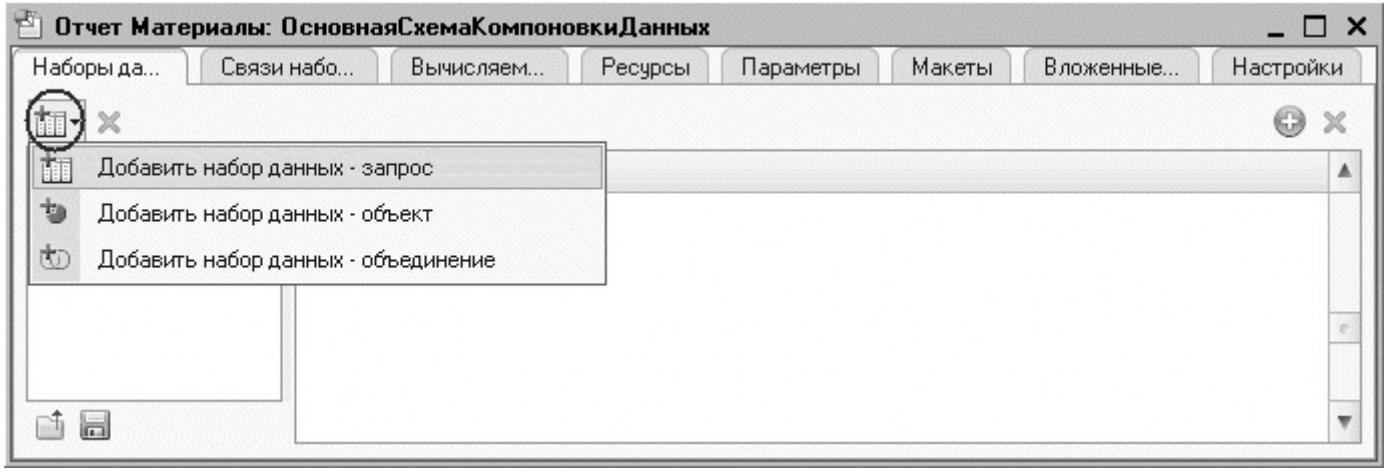


Рис. 7.5. Добавление набора данных в конструкторе схемы компоновки

Текст запроса

Для того чтобы создать текст запроса, запустим конструктор запроса – нажмем кнопку *Конструктор запроса* (рис. 7.6).

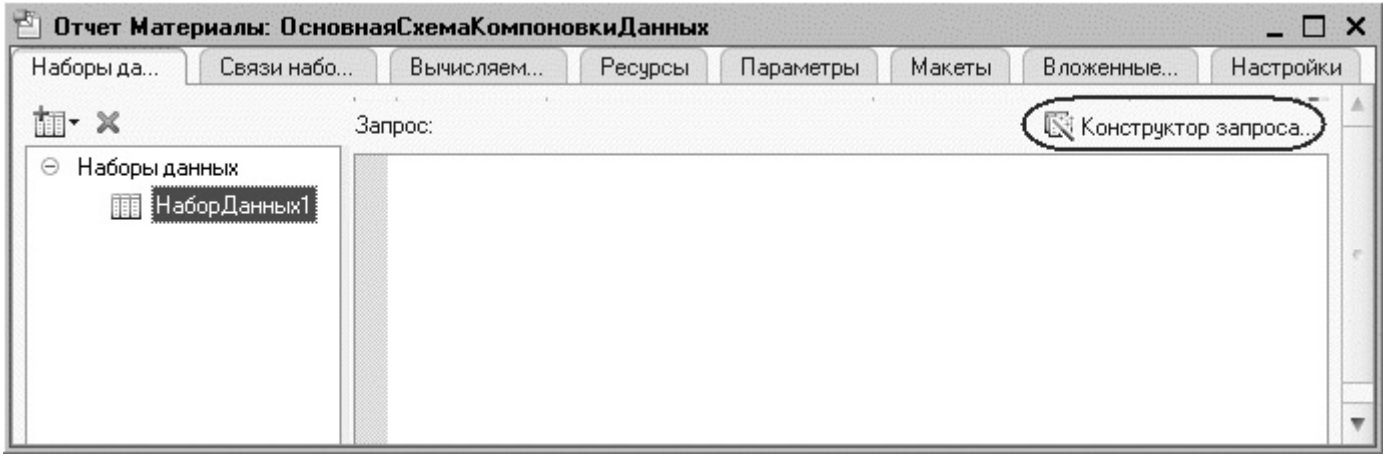


Рис. 7.6. Вызов конструктора запроса

Конструктор запроса – инструмент, созданный для помощи разработчику, позволяющий визуально конструировать запрос. Даже пользователь, не знакомый с языком запросов, может с помощью конструктора создать синтаксически правильный запрос.

В списке *База данных* представлены таблицы для создания запроса. На основе их данных мы имеем возможность построить отчет.

Если раскрыть ветку *РегистрыНакопления*, то мы увидим, что кроме таблицы регистра *ОстаткиМатериалов* в этой ветке присутствуют еще несколько *виртуальных таблиц*, которые формирует система (рис. 7.7).

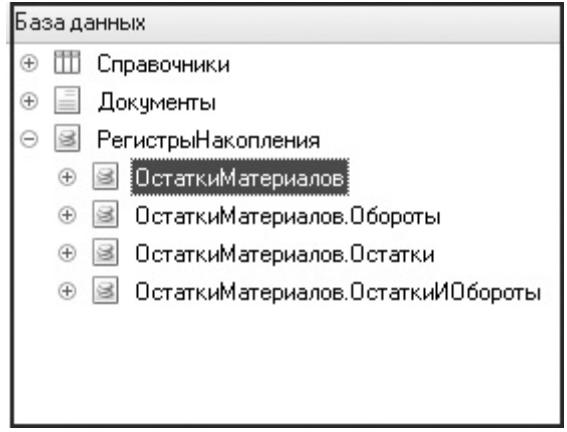


Рис. 7.7. Таблицы для создания запроса

Эти виртуальные таблицы, создаваемые платформой для регистров, и используются в основном для построения различных отчетов. Поскольку мы хотим видеть как остатки материалов, так и информацию об их поступлении и расходовании, нас будет интересовать виртуальная таблица *ОстаткиМатериалов.ОстаткиИОбороты*. Раскроем ее (рис. 7.8).

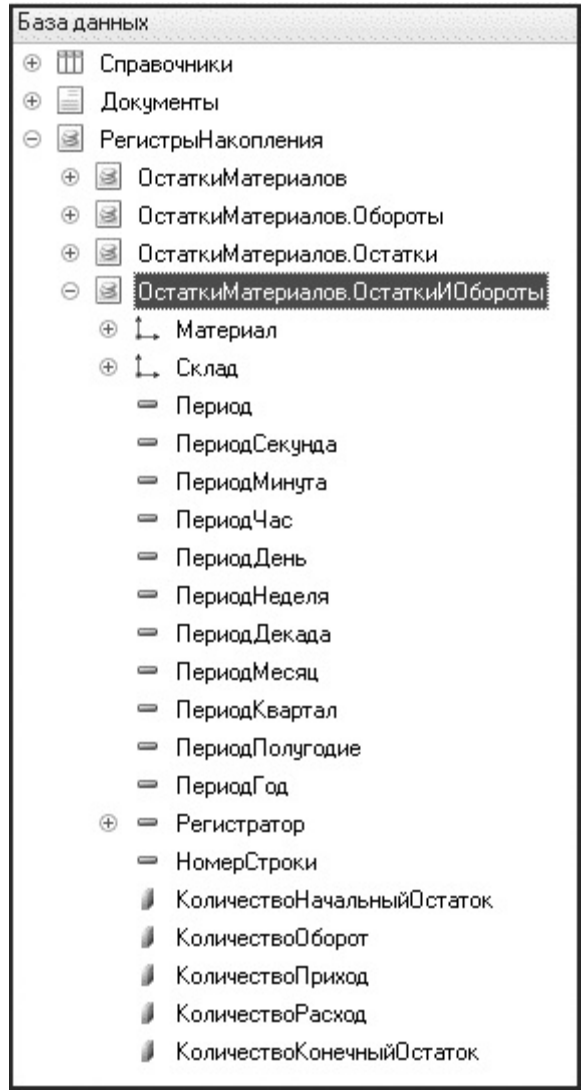

База данных
Справочники
Документы
РегистрыНакопления
ОстаткиМатериалов
ОстаткиМатериалов.Обороты
ОстаткиМатериалов.Остатки
ОстаткиМатериалов.ОстаткиИОбороты
Материал
Склад
Период
ПериодСекунда
ПериодМинута
ПериодЧас
ПериодДень
ПериодНеделя
ПериодДекада
ПериодМесяц
ПериодКвартал
ПериодПолугодие
ПериодГод
Регистратор
НомерСтроки
КоличествоНачальныйОстаток
КоличествоОборот
КоличествоПриход
КоличествоРасход
КоличествоКонечныйОстаток

Рис. 7.8. Таблица «ОстаткиМатериалов.ОстаткиИОбороты»

Как вы видите, эта таблица содержит измерения регистра *ОстаткиМатериалов* – *Материал*, *Склад* и кроме этого начальные и конечные остатки, а также значения прихода, расхода и оборотов для всех ресурсов регистра *ОстаткиМатериалов*.

Начнем выбирать поля таблицы в нужном нам порядке двойным щелчком мыши.

Сначала выберем *Склад* и *Материал*. Затем выберем *КоличествоНачальныйОстаток*, *КоличествоПриход, КоличествоРасход.* В заключение выберем *КоличествоКонечныйОстаток*.

ПРИМЕЧАНИЕ

Выделенные элементы можно перенести из одного списка в другой перетаскиванием мышью или двойным щелчком на них. Либо можно использовать кнопки >, <, >>, <<.

В результате окно *Поля* должно быть заполнено следующим образом (рис. 7.9).

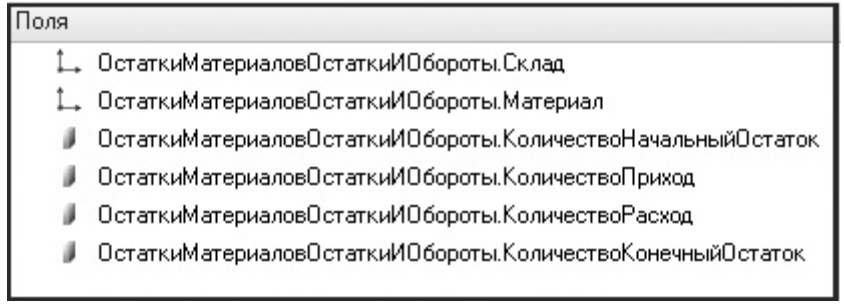


Рис. 7.9. Выбранные поля

Нажмем *OK* и вернемся в конструктор схемы компоновки данных (рис. 7.10).

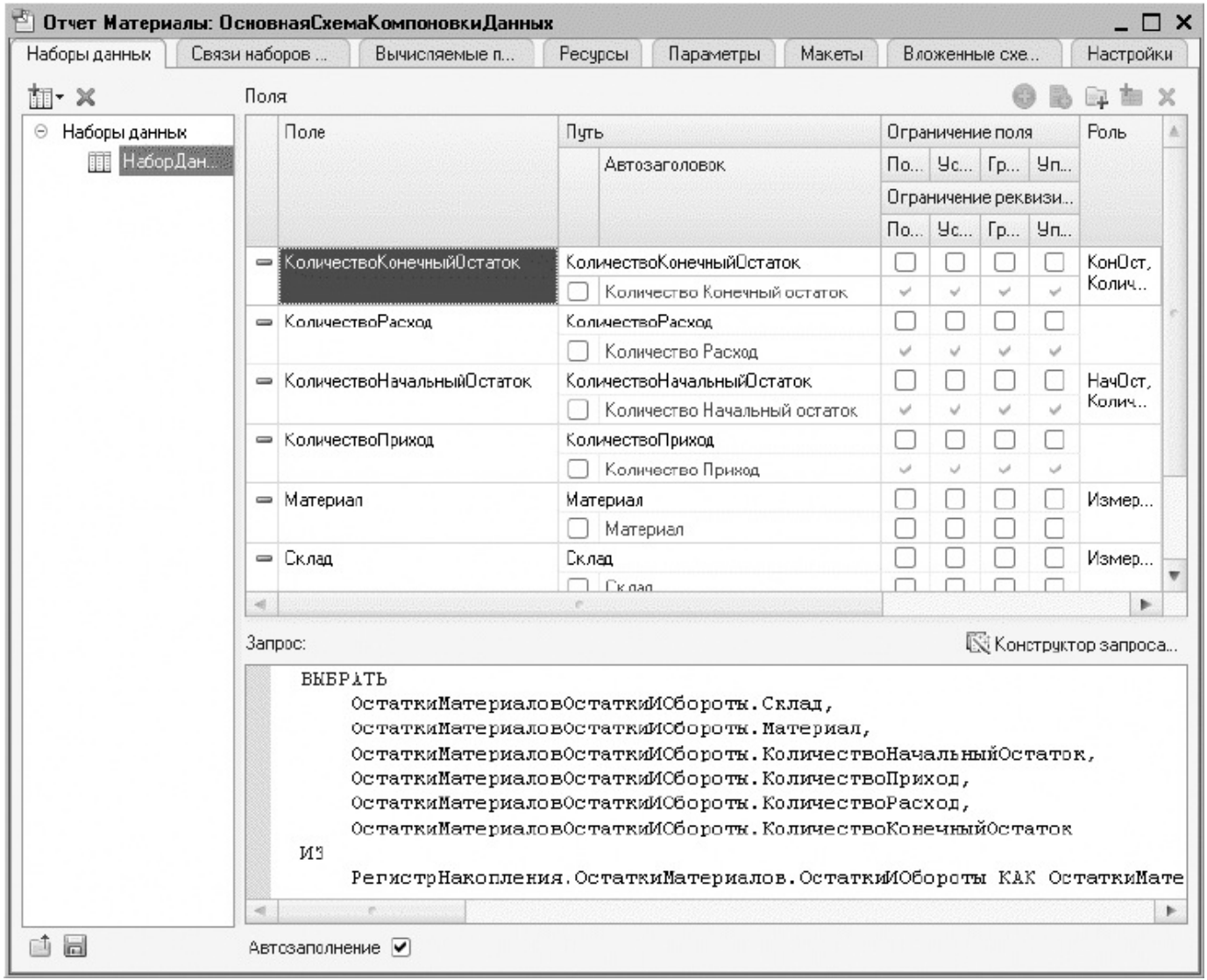

Отчет Материалы: ОсновнаяСхемаКомпоновкиДанных
Наборы данных
Связи наборов ...
Вычисляемые п...
Ресурсы
Параметры
Макеты
Вложенные схе...
Настройки
Поля
Наборы данных
НаборДан...
Поле
Путь
Автозаголовок
Ограничение поля
По...
Ус...
Гр...
Уп...
Ограничение реквизи...
Роль
КоличествоКонечныйОстаток
КоличествоКонечныйОстаток
Количество Конечный остаток
КонОст, Колич...
КоличествоРасход
КоличествоРасход
Количество Расход
КоличествоНачальныйОстаток
КоличествоНачальныйОстаток
Количество Начальный остаток
НачОст, Колич...
КоличествоПриход
КоличествоПриход
Количество Приход
Материал
Материал
Материал
Измер...
Склад
Склад
Склад
Измер...
Запрос:
Конструктор запроса...
ВЫБРАТЬ
ОстаткиМатериаловОстаткиИОбороты.Склад,
ОстаткиМатериаловОстаткиИОбороты.Материал,
ОстаткиМатериаловОстаткиИОбороты.КоличествоНачальныйОстаток,
ОстаткиМатериаловОстаткиИОбороты.КоличествоПриход,
ОстаткиМатериаловОстаткиИОбороты.КоличествоРасход,
ОстаткиМатериаловОстаткиИОбороты.КоличествоКонечныйОстаток
ИЗ
РегистрНакопления.ОстаткиМатериалов.ОстаткиИОбороты КАК ОстаткиМате
Автозаполнение

Рис. 7.10. Конструктор схемы компоновки данных

Текст запроса, который был создан с помощью конструктора, платформа поместит в поле *Запрос*.

Это поле представляет собой текстовый редактор, в котором можно вручную отредактировать существующий запрос. Кроме того, можно снова вызвать конструктор запроса и отредактировать запрос при помощи него.

Для упрощения восприятия материала мы не будем здесь приводить листинг запроса и разбирать его. Более подробно этот вопрос будет рассмотрен в разделе о языке запросов на занятии «Отчеты». В данном случае, как и для многих других отчетов, можно не анализировать и не редактировать текст запроса.

Обратите внимание на список полей системы компоновки данных, который платформа заполнила в верхней части конструктора. В нем отображаются поля, которые доступны у текущего набора данных. В нашем случае система «1С:Предприятие» заполнила данный список автоматически, из текста запроса, и нет необходимости в его ручной настройке.

Итак, мы описали, каким образом будут извлекаться данные для отчета. Но пока мы не создадим стандартных настроек нашего отчета, мы ничего не

увидим в результате. Поэтому создадим самые простые настройки отчета для отображения обычных детальных записей информационной базы.

В нашем случае это будут записи виртуальной таблицы регистра накопления *ОстаткиМатериалов*, выбранные в линейном порядке по мере попадания их в эту таблицу.

Настройки отчета

Перейдем на закладку *Настройки*. В верхнем правом окне будет находиться иерархическая структура нашего отчета.

Для добавления нового элемента выделим в дереве структуры отчета корневой элемент *Отчет* и вызовем его контекстное меню. Можно также нажать кнопку *Добавить* в командной панели окна или нажать клавишу *Ins*.

Добавим в отчет группировку (контекстное меню – *Новая группировка*). При этом не станем указывать поле группировки, а просто нажмем *ОК*.

Тем самым мы определили, что в отчет будут выводиться *детальные* записи из информационной базы – записи, получаемые в результате выполнения запроса без итогов (рис. 7.11).

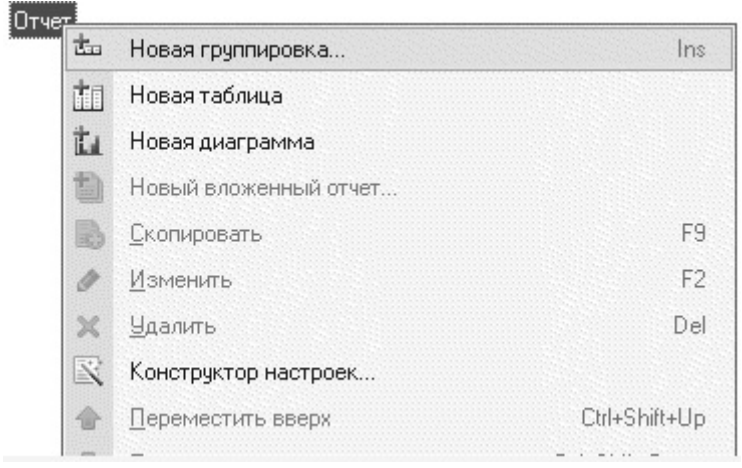


Рис. 7.11. Добавление новой группировки в отчет

В структуре отчета появится группировка *Детальные записи*.

Теперь настроим поля, которые будут выводиться в результат отчета.

Для этого перейдем в нижнем окне настроек на закладку *Выбранные поля* и перенесем мышью из списка доступных полей:

- *Склад,*
- *Материал,*
- *КоличествоНачальныйОстаток,*
- *КоличествоПриход,*

- *КоличествоРасход,*
- *КоличествоКонечныйОстаток.*

ПРИМЕЧАНИЕ

Добавление доступных полей в список выбранных полей можно осуществить перетаскиванием мышью, двойным щелчком на доступных полях либо нажатием кнопки ***Добавить*** *справа от списка выбранных полей. Порядок выбранных полей можно изменить позже кнопками* ***Вверх****,* ***Вниз*** *или перетаскиванием мышью.*

В результате окно настроек отчета должно иметь вид (рис. 7.12).

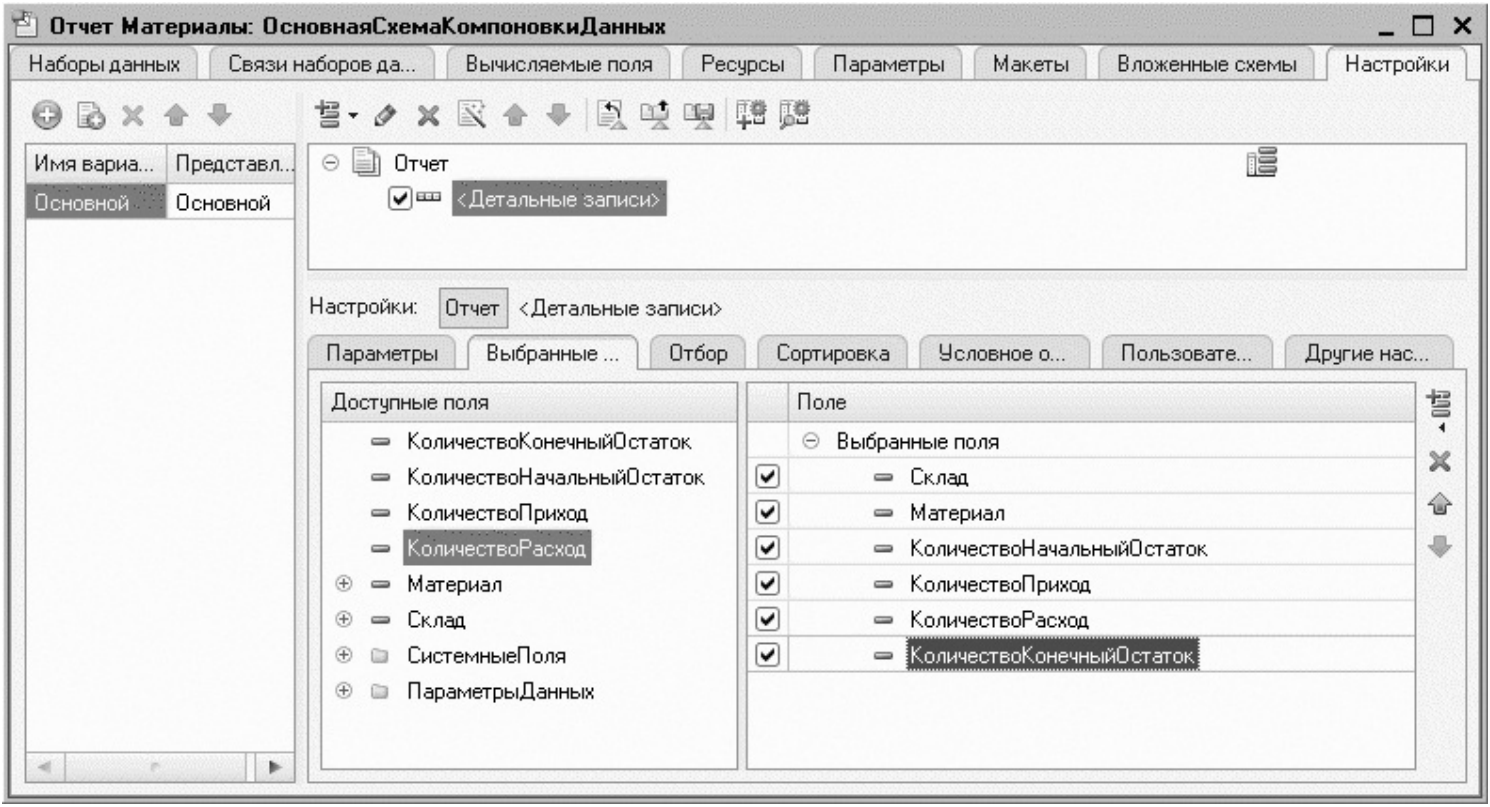


Рис. 7.12. Окно настроек отчета

Затем перейдем на закладку *Параметры* и укажем, что параметры отчета *Дата начала* и *Дата окончания* будут включены в состав пользовательских настроек, и эти настройки будут находиться непосредственно в форме отчета, то есть будут быстрыми настройками.

Сначала укажем, что оба эти параметра будут использоваться в отчете – установим флажки в первой колонке.

Затем выделим каждый из параметров, нажмем кнопку *Свойства элемента пользовательских настроек* и установим флажок *Включать в пользовательские настройки* (рис. 7.13).

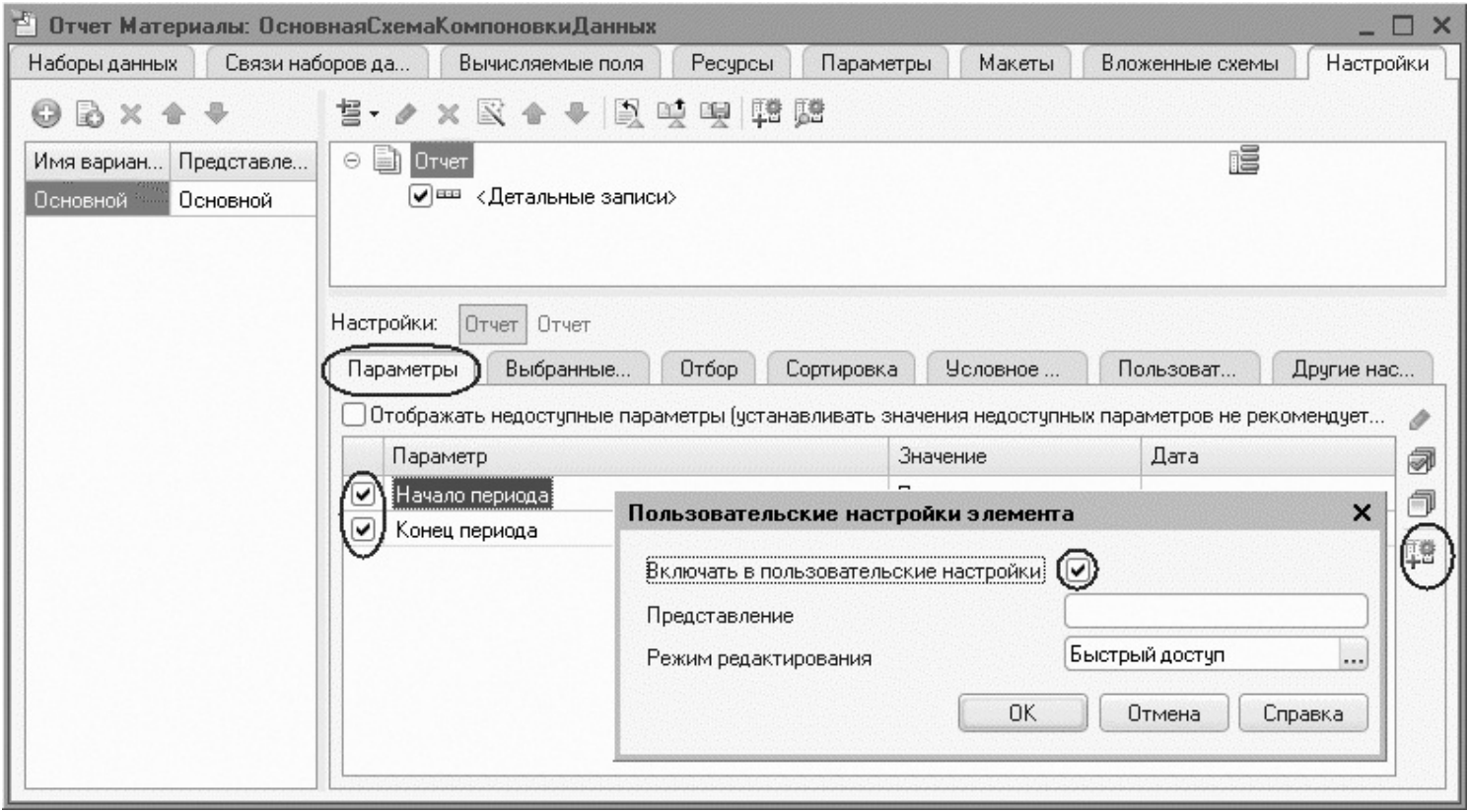


Рис. 7.13. Определение списка подсистем, в которых будет отражаться отчет

Таким образом, перед формированием отчета пользователь сможет задать отчетный период. Более подробно о параметрах и пользовательских настройках отчета будет рассказано позже на занятии «Отчеты».

В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации Отчет *Материалы* перейдем на закладку *Подсистемы*.

Отметим в списке подсистем конфигурации ветви *Учет материалов*, *Оказание услуг* и *Бухгалтерия*.

Таким образом, ссылка на наш отчет автоматически попадет в панель действий этих подсистем (рис. 7.14).

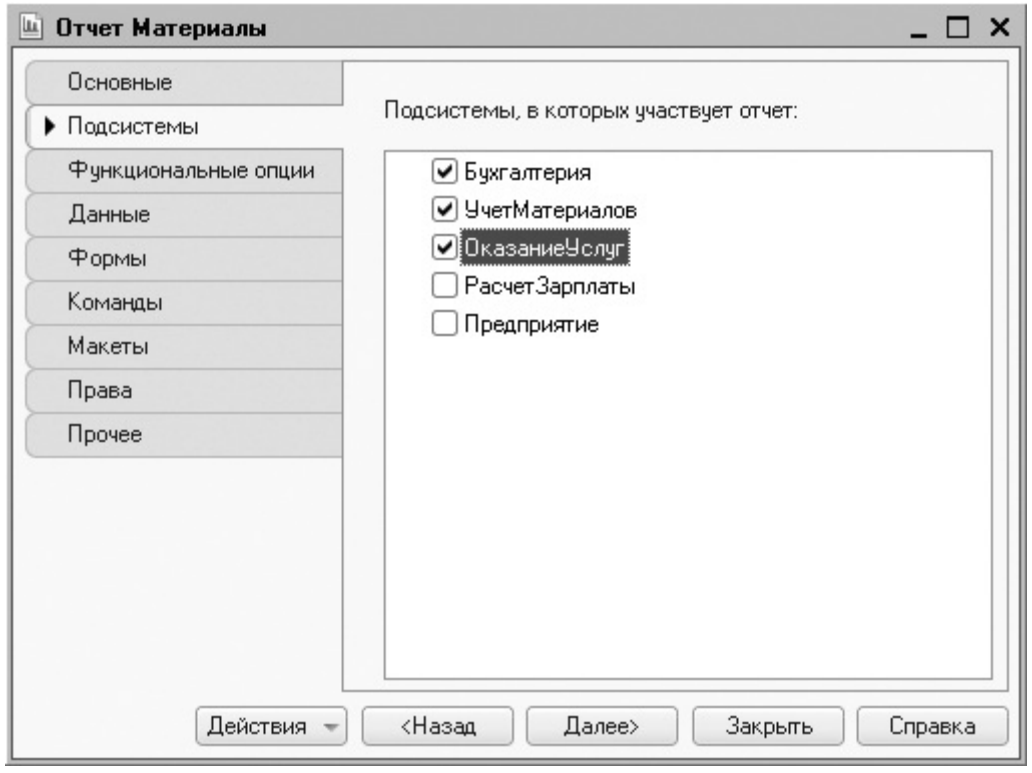


Рис. 7.14. Определение списка подсистем, в которых будет отражаться отчет

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает отчет.

В открывшемся окне «1С:Предприятия» мы видим, что в панели действий

разделов *Бухгалтерия*, *Оказание услуг* и *Учет материалов* появилась новая группа команд для выполнения отчетов и в ней команда для формирования отчета *Материалы*. Выполним ее (рис. 7.15).

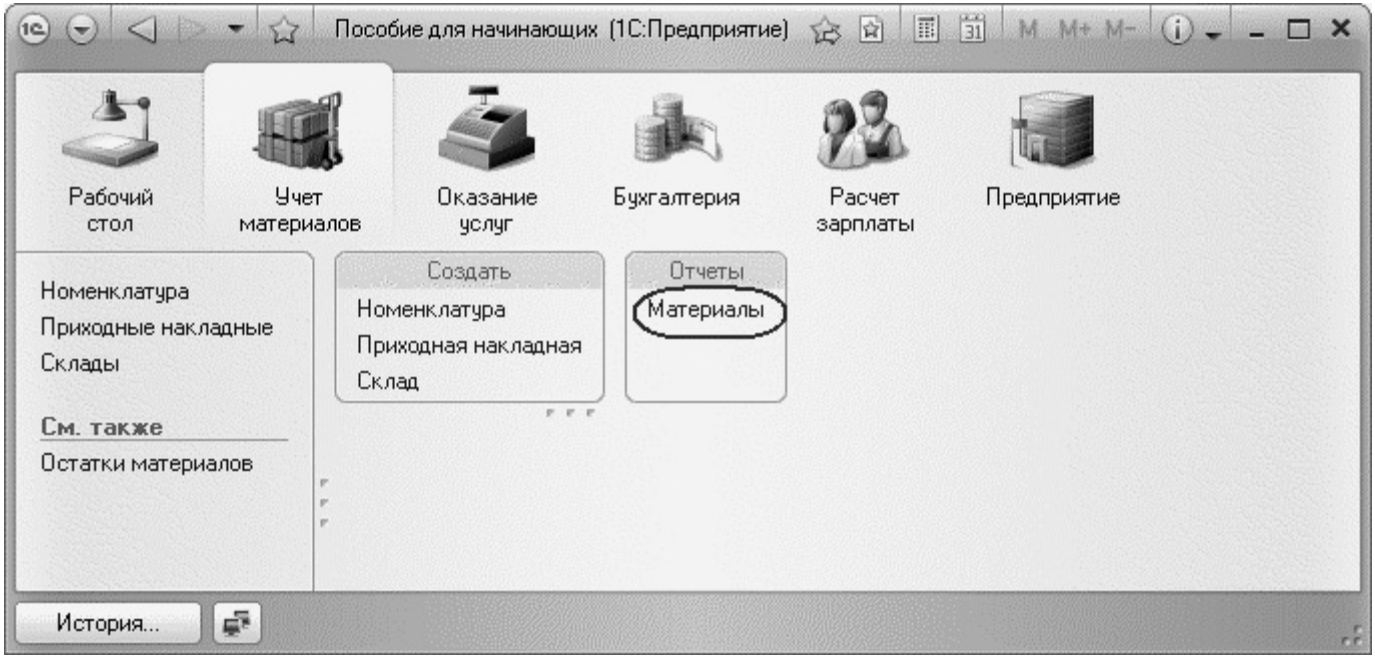
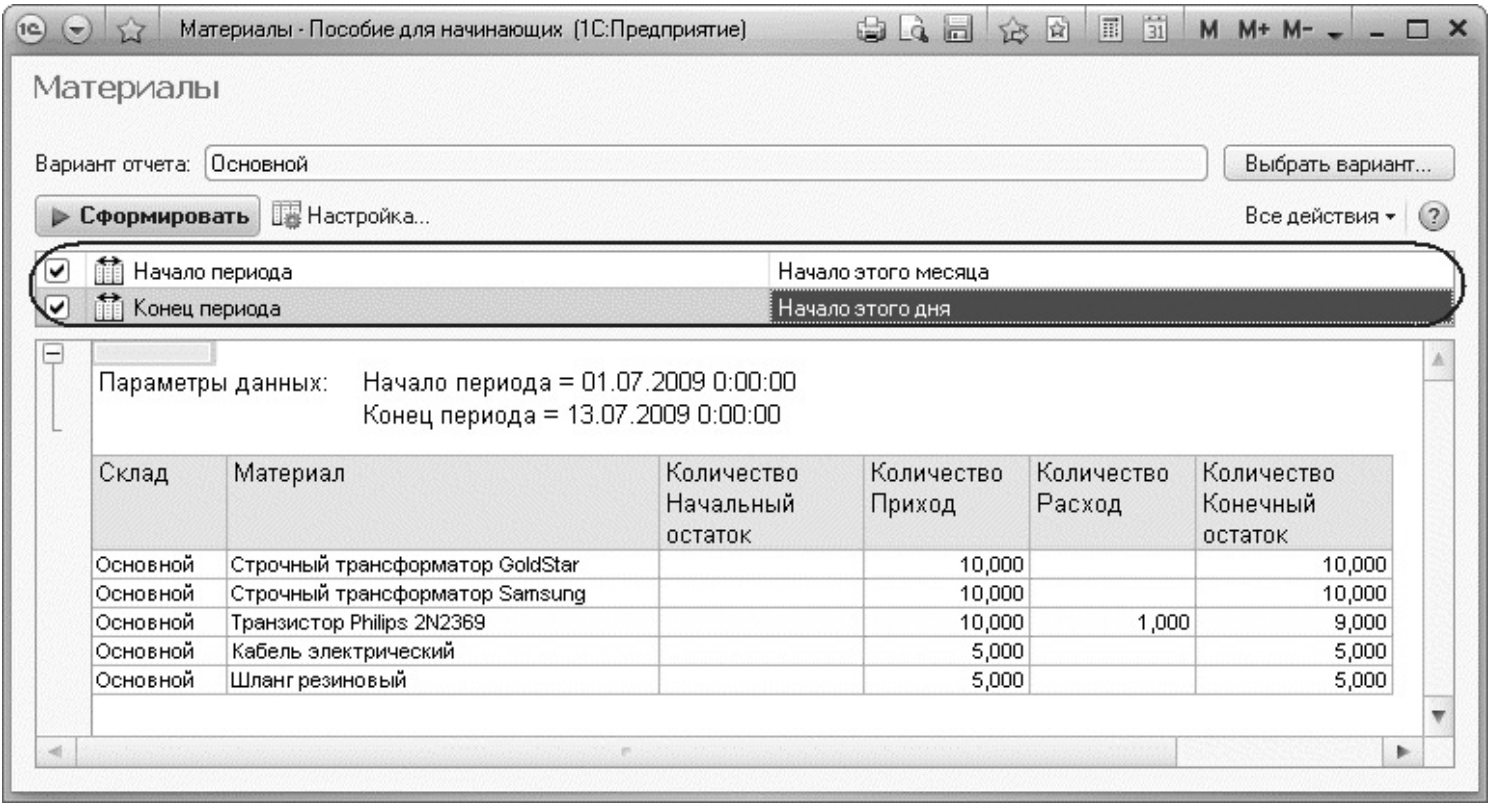


Рис. 7.15. Команда для формирования отчета «Материалы»

Перед нами откроется автоматически сформированная системой форма отчета.

Зададим даты начала и окончания отчетного периода и нажмем кнопку

Сформировать (рис. 7.16).



Склад	Материал	Количество Начальный остаток	Количество Приход	Количество Расход	Количество Конечный остаток
Основной	Строчный трансформатор GoldStar		10,000		10,000
Основной	Строчный трансформатор Samsung		10,000		10,000
Основной	Транзистор Philips 2N2369		10,000	1,000	9,000
Основной	Кабель электрический		5,000		5,000
Основной	Шланг резиновый		5,000		5,000

Рис. 7.16. Отчет «Материалы»

Как видите, наш отчет вполне «презентабелен» и полностью отражает движение материалов, произошедшее в нашей организации.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Отчет».*
- *Как создать отчет с помощью конструктора схемы компоновки данных.*
- *Как отобразить отчет в разделах прикладного решения.*

Занятие 8 (1:10). Макеты. Редактирование макетов и форм

Продолжительность

Ориентировочная продолжительность занятия – 1 час 10 минут.

На этом занятии вы познакомитесь с очередным новым объектом конфигурации – *Макет*. Вы узнаете о его назначении и создадите макет документа, на основе которого будет формироваться печатная форма документа.

Что такое макет

Объект конфигурации *Макет* предназначен для хранения различных форм представления данных, различных данных, которые могут потребоваться каким-либо объектам конфигурации или всему прикладному решению в целом.

Макет может содержать табличный или текстовый документ, двоичные данные, HTML-документ или Active Document, графическую или географическую схему, схему компоновки данных или макет оформления схемы компоновки данных.

Макеты могут существовать как сами по себе (общие макеты), так и быть подчинены какому-либо объекту конфигурации.

Одно из предназначений макета, подчиненного объекту конфигурации и содержащего табличный документ, – создание печатной формы этого объекта. Создание печатной формы заключается в конструировании ее составных частей – именованных областей, из которых затем «собирается» готовая печатная форма.

Порядок заполнения областей данными и вывода их в итоговую форму описывается при помощи встроенного языка. Печатная форма может включать в себя различные графические объекты: картинки, OLE-объекты, диаграммы и т. д.

Помимо создания макета «вручную» конфигуратор предоставляет разработчику возможность воспользоваться специальным инструментом – *конструктором печати*, который берет на себя большинство рутинной работы по созданию макета.

Макет печатной формы

В режиме «Конфигуратор»

Наша цель будет заключаться в создании печатной формы документа *Оказание услуги*.

Откроем в конфигураторе окно редактирования объекта конфигурации *Документ ОказаниеУслуги*.

Перейдем на закладку *Макеты* и запустим конструктор печати (рис. 8.1).

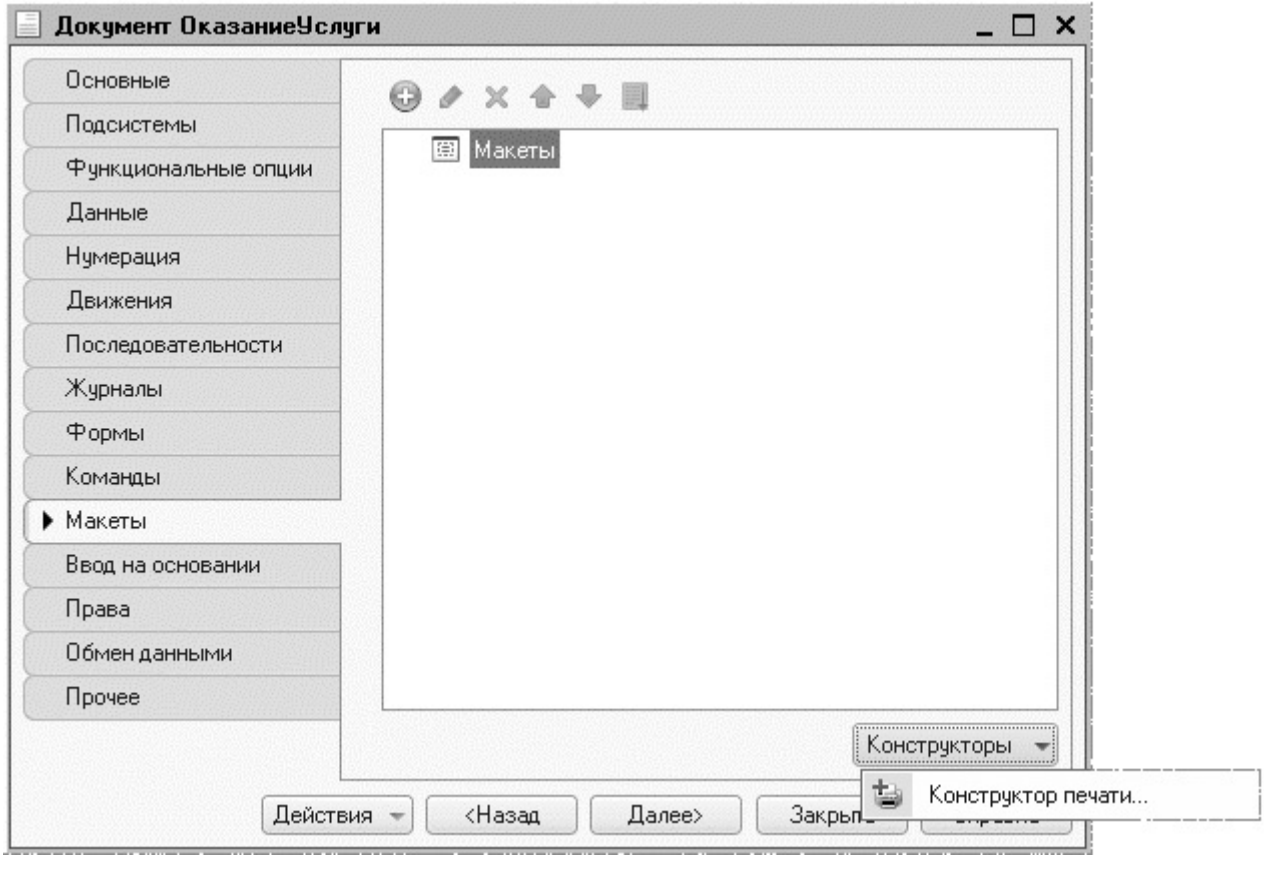


Рис. 8.1. Запуск конструктора печати

В открывшемся окне конструктора на первом шаге укажем, что будет создана новая команда *Печать* для формирования печатной формы документа (рис. 8.2).

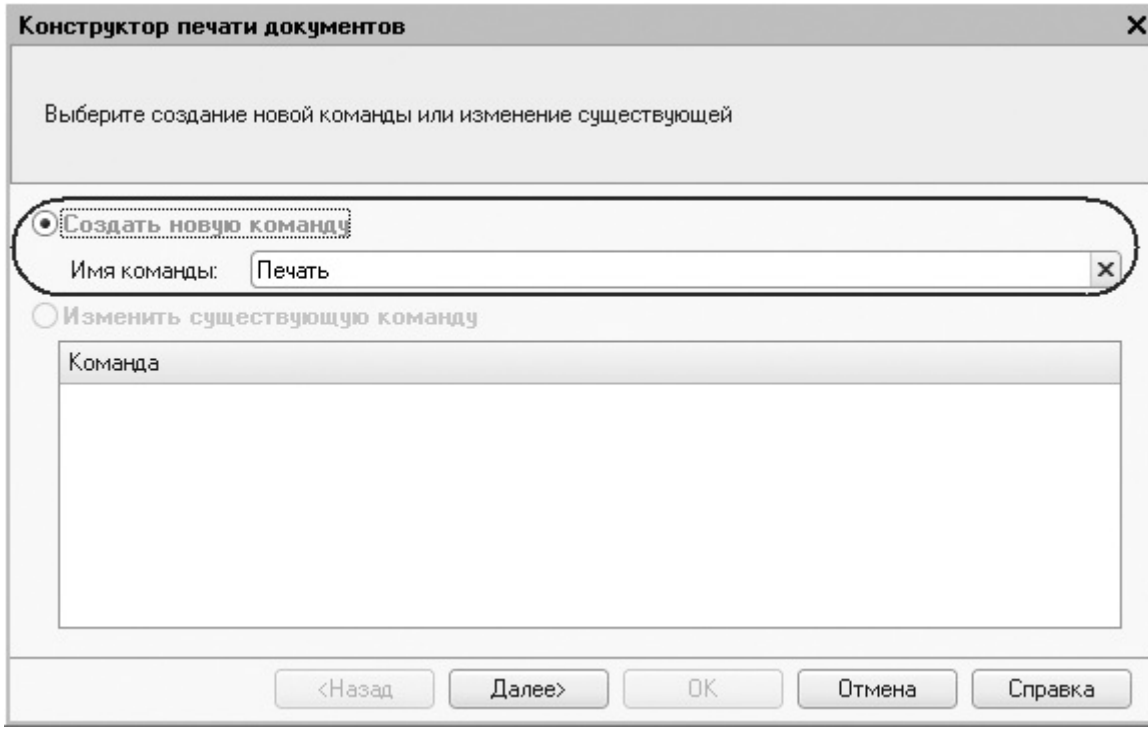


Рис. 8.2. Конструктор печати. Шаг 1

Нажмем *Далее*.

На втором шаге нажатием кнопки >> определим, что все реквизиты нашего документа будут отображены в шапке печатной формы (рис. 8.3).

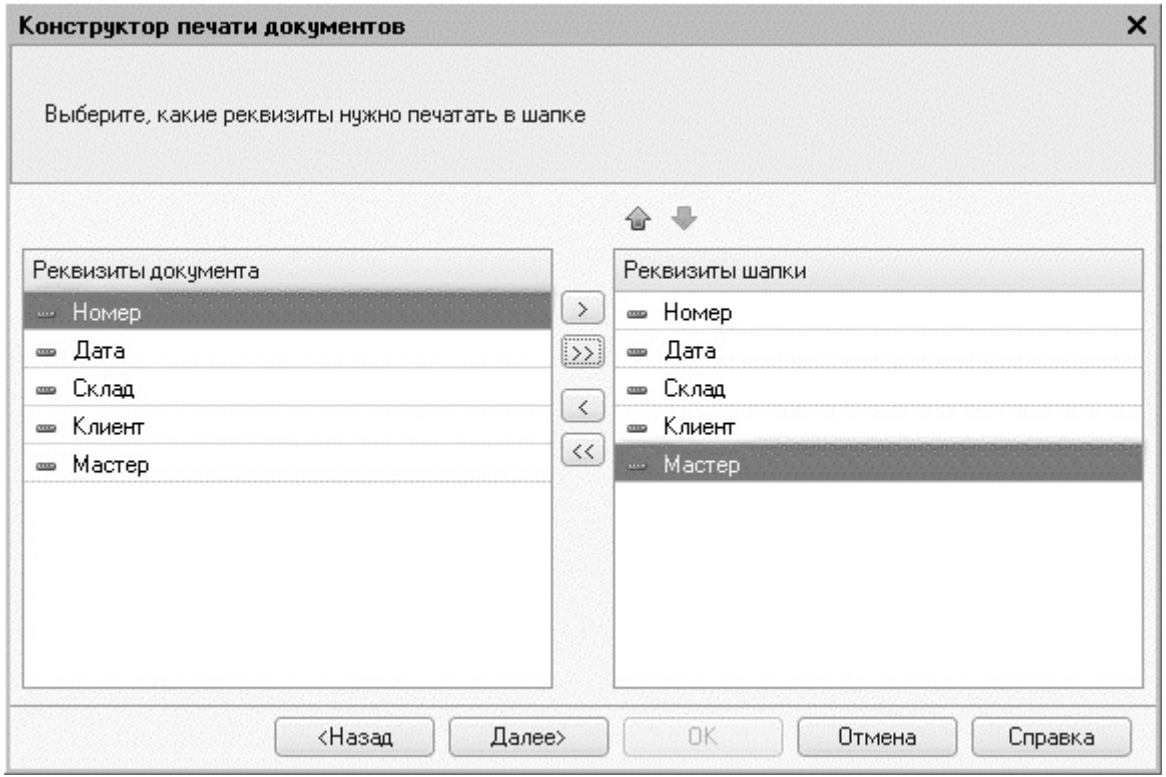


Рис. 8.3. Конструктор печати. Шаг 2

Нажмем *Далее*.

На третьем шаге точно так же определим, что все реквизиты табличной части документа будут отображены в печатной форме (рис. 8.4).

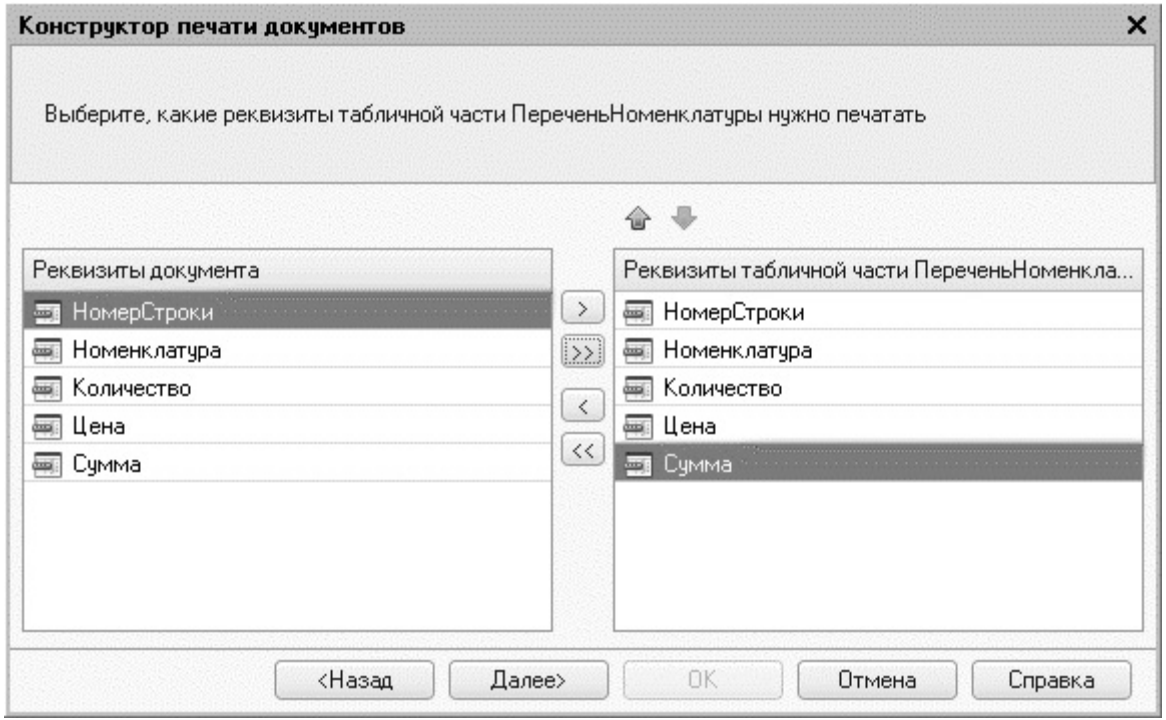


Рис. 8.4. Конструктор печати. Шаг 3

Нажмем *Далее*.

На четвертом шаге конструктор предложит сформировать нам подвал (нижнюю часть) печатной формы. Мы не станем ничего указывать (подвал в данном случае использовать не будем), нажмем *Далее* и перейдем к пятому шагу (рис. 8.5).

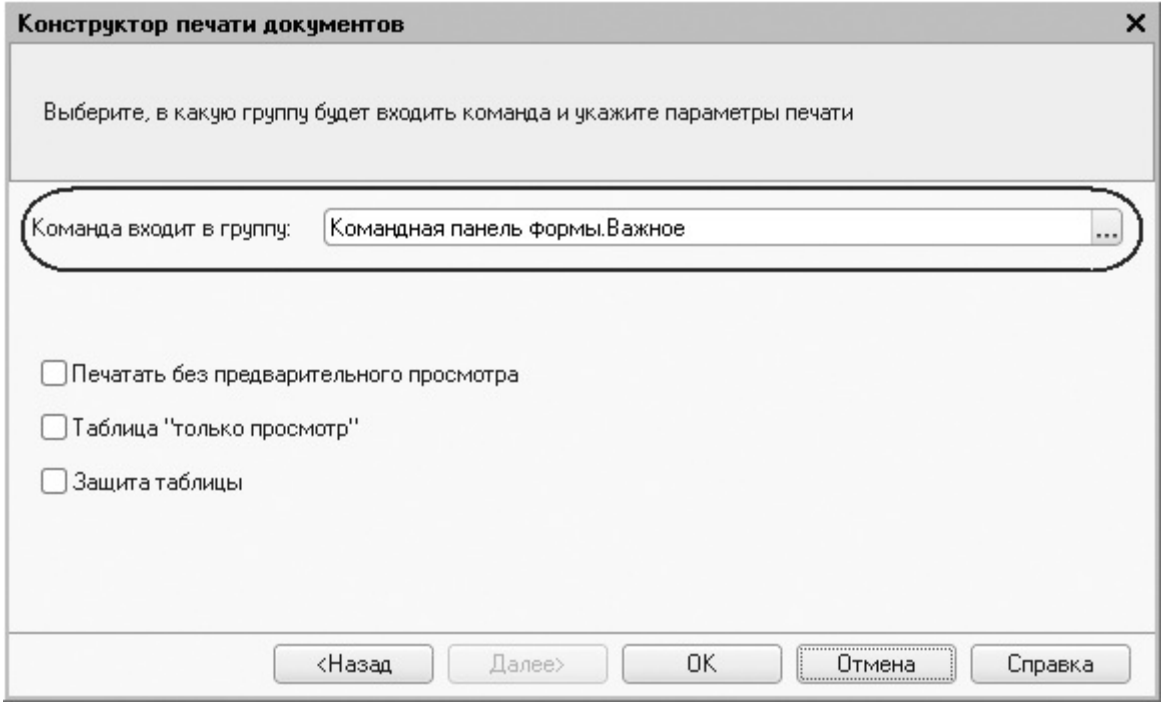


Рис. 8.5. Конструктор печати. Шаг 5

Здесь ничего изменять не будем. Тем самым согласимся с тем, что команда

для вызова процедуры формирования печатной формы будет помещена в командную панель формы, в раздел *Важное*.

Нажмем *ОК*.

В конфигураторе откроется модуль команды *Печать*, модуль менеджера документа *ОказаниеУслуги* и макет этого документа (рис. 8.6).

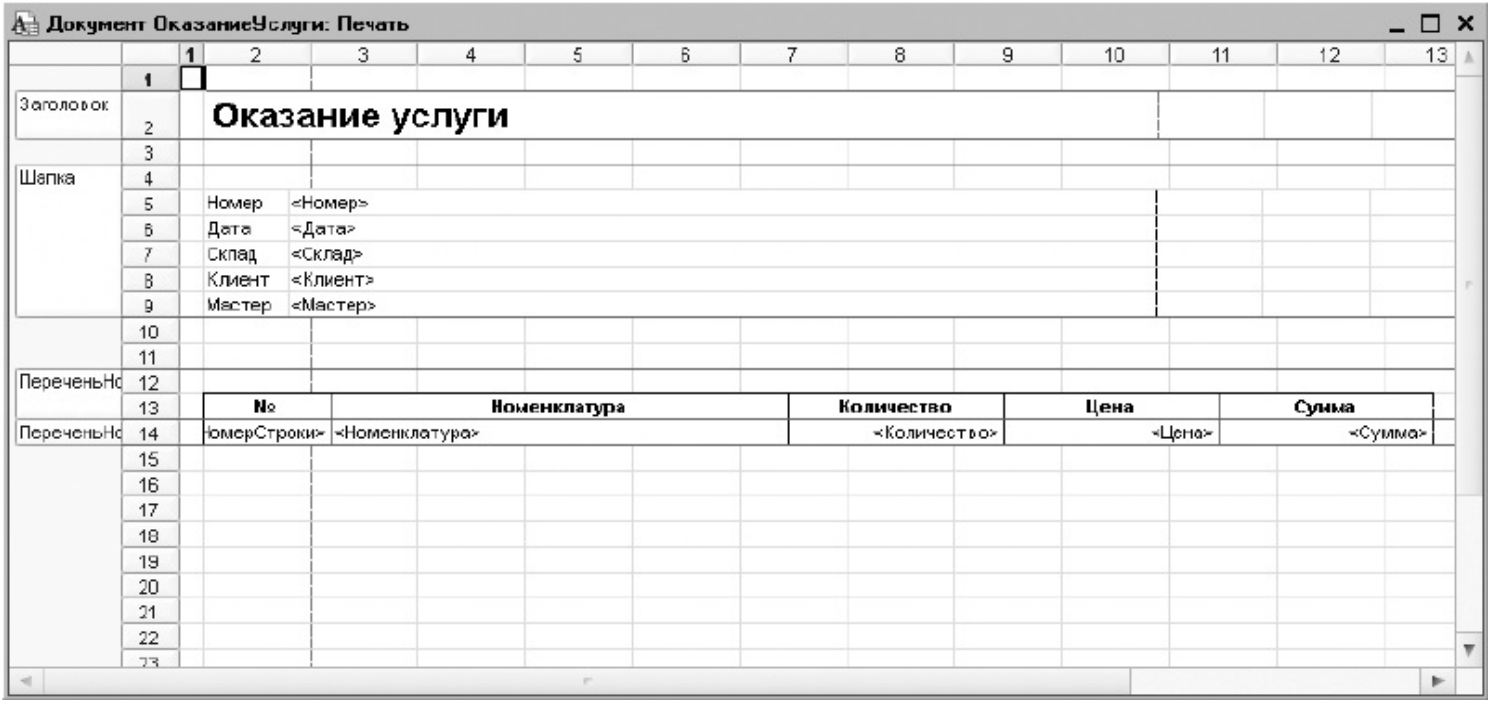


Рис. 8.6. Макет документа «Оказание услуги»

Заметим, что разработчик может создать макет печатной формы с нуля, и для ее вывода создать соответствующую команду и кнопку в форме документа, но в данном случае всю работу сделал за нас конструктор печати:

- Создан макет печатной формы документа *ОказаниеУслуги* с именем *Печать* (см. рис. 8.6).
- Создана команда документа *ОказаниеУслуги* с именем *Печать*. В модуль этой команды помещен обработчик, вызывающий процедуру печати документа, выполняющуюся на сервере. Сама процедура печати помещена в модуль менеджера документа *ОказаниеУслуги* (рис. 8.7).

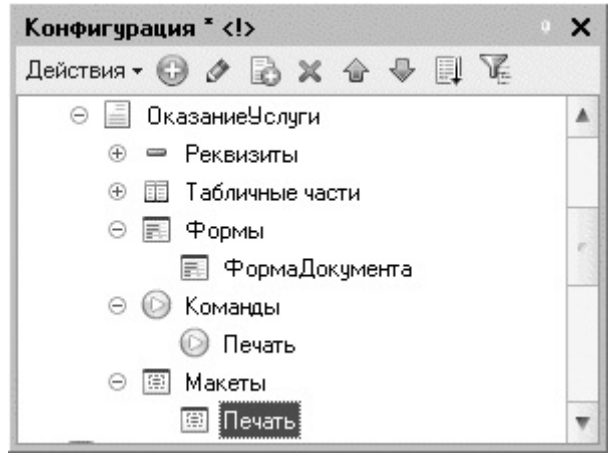


Рис. 8.7. Структура документа «Оказание услуги» в дереве объектов конфигурации

- В командную панель формы документа *ОказаниеУслуги* помещена команда *Печать* для формирования печатной формы документа (рис. 8.8).

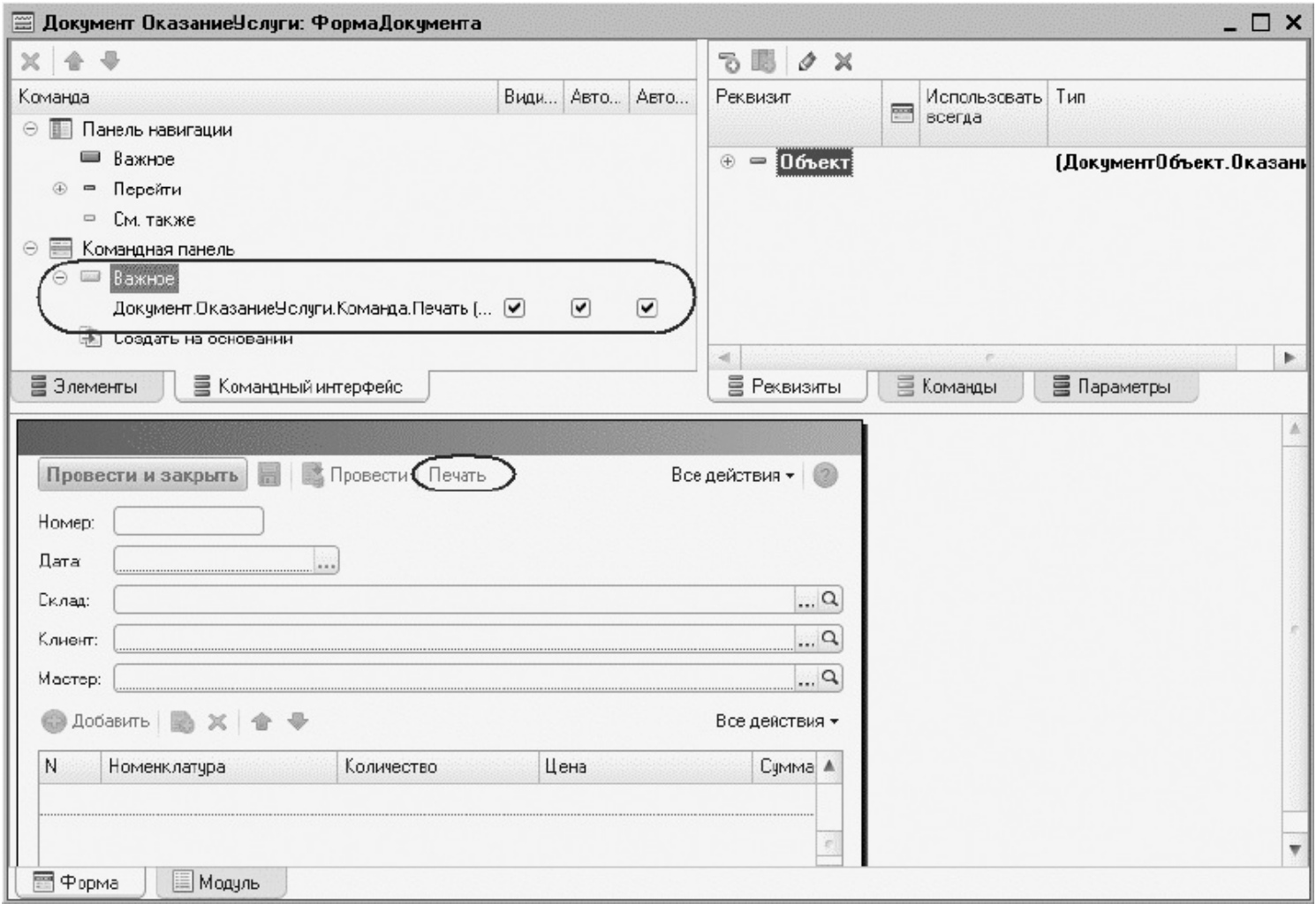


Рис. 8.8. Макет документа «Оказание услуги»

Причем поскольку команда *Печать* принадлежит документу *ОказаниеУслуги* в целом, а не конкретной его форме, эту команду можно будет помещать в любую форму, созданную для документа.

В будущем мы будем самостоятельно создавать процедуры обработчиков команд и размещать соответствующие им кнопки в форме, но пока воспользуемся результатами работы конструктора печати и проверим макет в работе.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем документ *Оказание услуги № 1*.

Обратите внимание, что в командной панели документа появилась новая кнопка *Печать* (рис. 8.9).

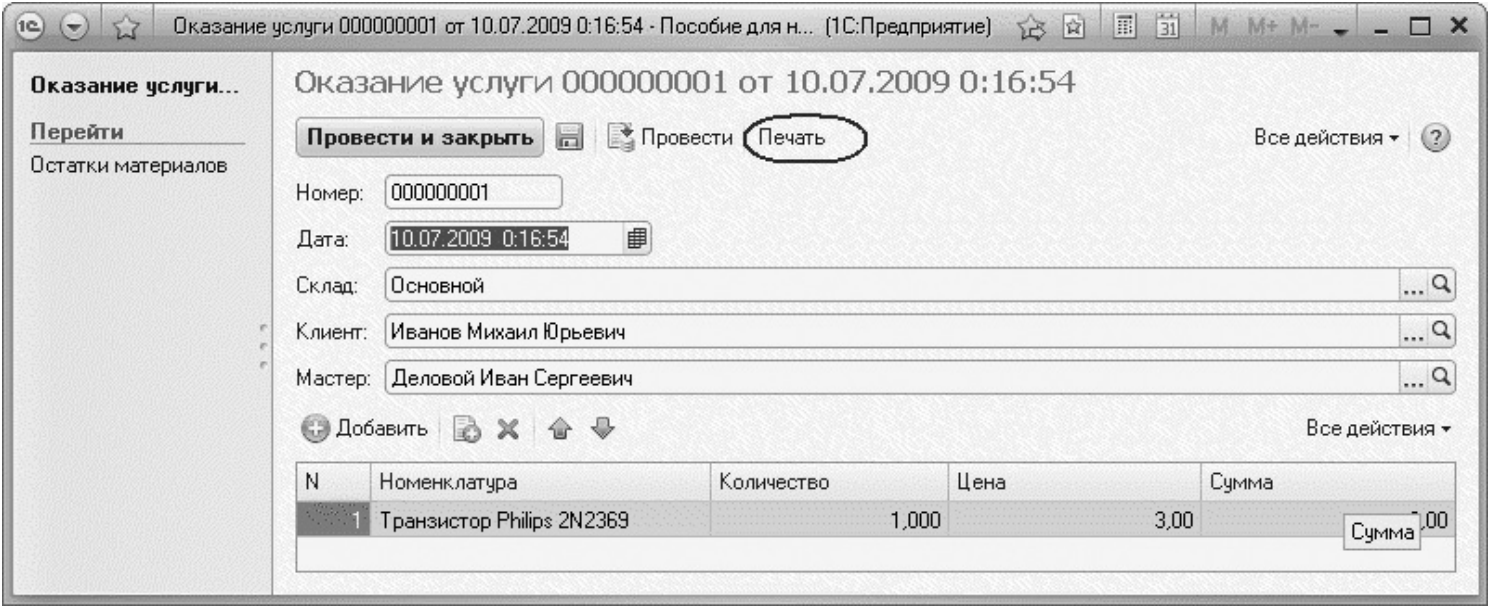


Рис. 8.9. Форма документа «Оказание услуги» с новой кнопкой «Печать»

Нажмем на нее и увидим печатную форму нашего документа (рис. 8.10).

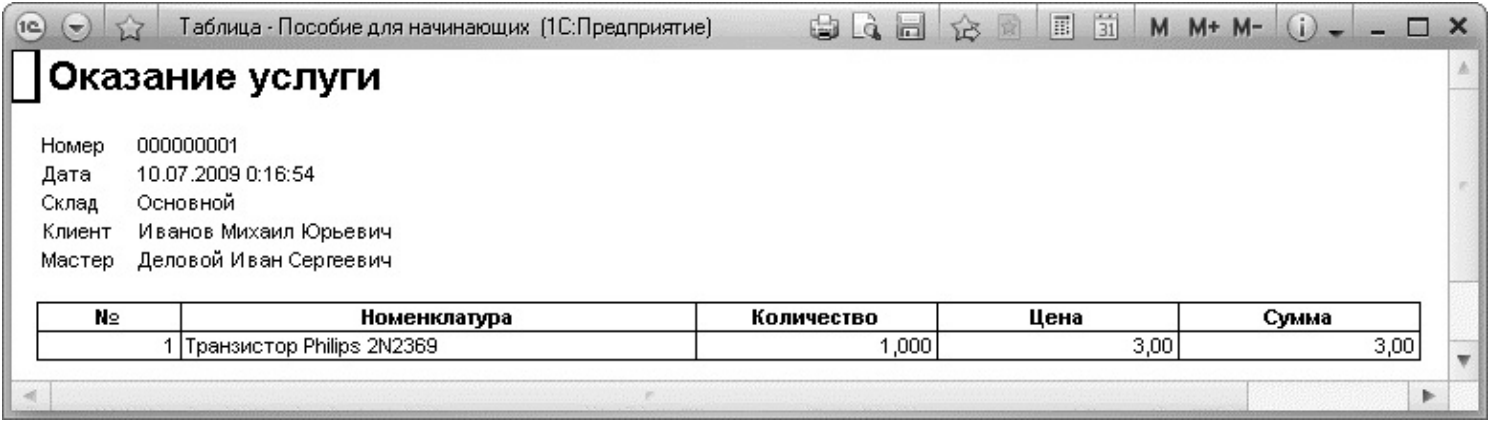


Рис. 8.10. Печатная форма документа «Оказание услуги»

Как видите, конструктор сформировал вполне подходящую печатную форму для нашего документа. Единственное, чего не хватает в данной форме, это итоговой суммы документа.

В следующем разделе на примере добавления итоговой суммы документа мы познакомимся с тем, как можно редактировать макеты и формы объектов конфигурации.

Редактирование макета

В режиме «Конфигуратор»

Прежде всего, добавим итоговую сумму в печатную форму документа

ОказаниеУслуги.

Откроем конфигуратор, раскроем дерево документа *Оказание Услуги* и дважды щелкнем на макете *Печать*.

Как видите, макет документа состоит из именованных областей, которые в определенном порядке выводятся на печать.

Те именованные области, которые мы видим слева, были созданы с помощью конструктора. Но разработчик может сам создавать или удалять области, переименовывать их и т. п.

Добавим новую область для вывода итоговой суммы документа. Выделим мышью две пустые строки под табличной частью документа и выполним пункт главного меню *Таблица* > *Имена* > *Назначить имя*… (рис. 8.11).

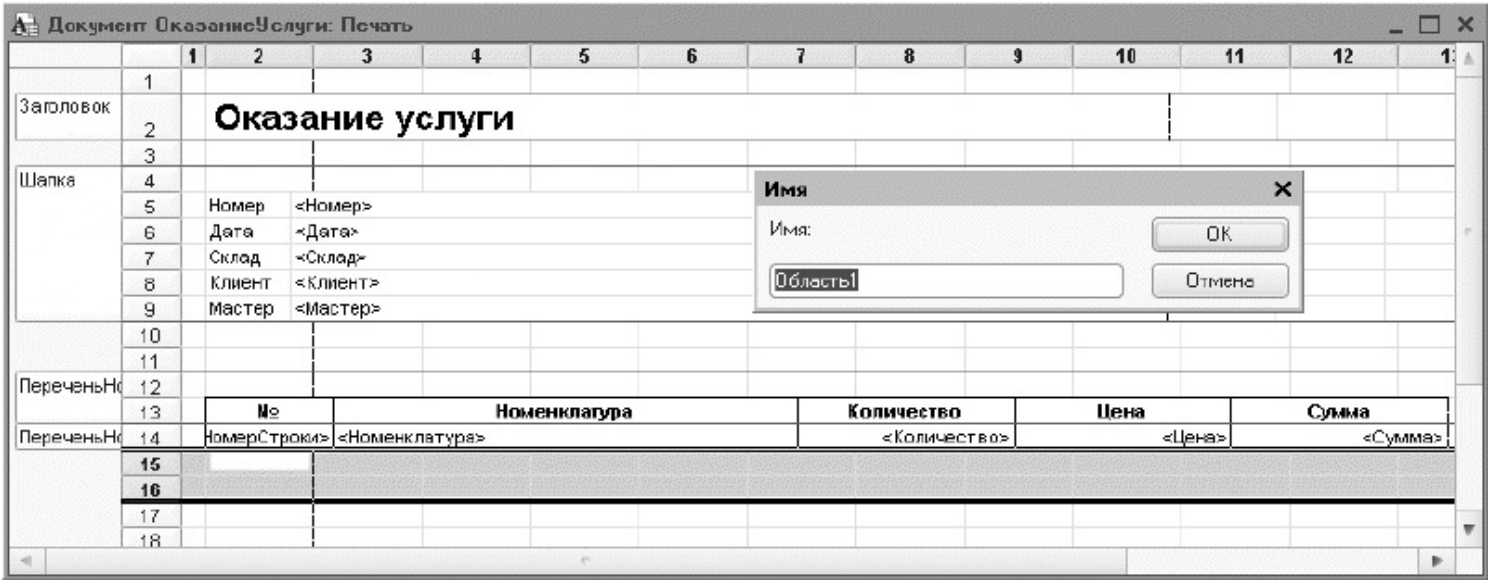


Рис. 8.11. Создание области ячеек для вывода итоговой строки

Назовем область *Всего*, нажмем *ОК*.

Чтобы формат добавленных нами строк совпадал с имеющимся форматом заголовка и табличной части документа, изменим ширину колонок.

Для этого потянем мышью в заголовке таблицы за правую границу колонки 2 так, чтобы ее ширина совпала с шириной колонки № в шапке документа. Отпустим мышь.

Платформа предложит создать новый формат для выделенных строк. Согласимся.

Аналогичные действия выполним и для колонок 3, 4, 5 и 6.

В созданной области, в колонке *Цена*, напишем *ВСЕГО*, а в колонке *Сумма* напишем *ВсегоПоДокументу* (рис. 8.12).

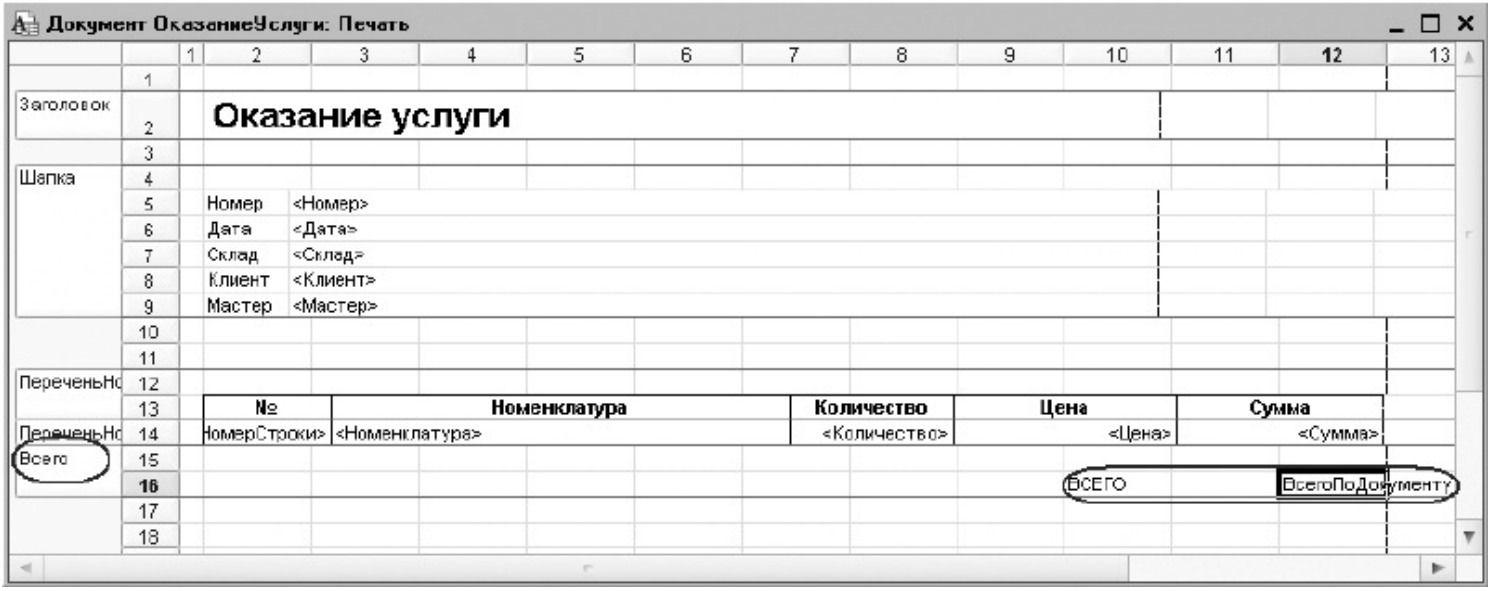


Рис. 8.12. Создание ячеек для вывода итога

Вызвав палитру свойств для последней заполненной нами ячейки (контекстное

меню – *Свойства*), в свойстве *Заполнение* укажем, что в этой ячейке будет находиться не текст, а параметр (рис. 8.13).

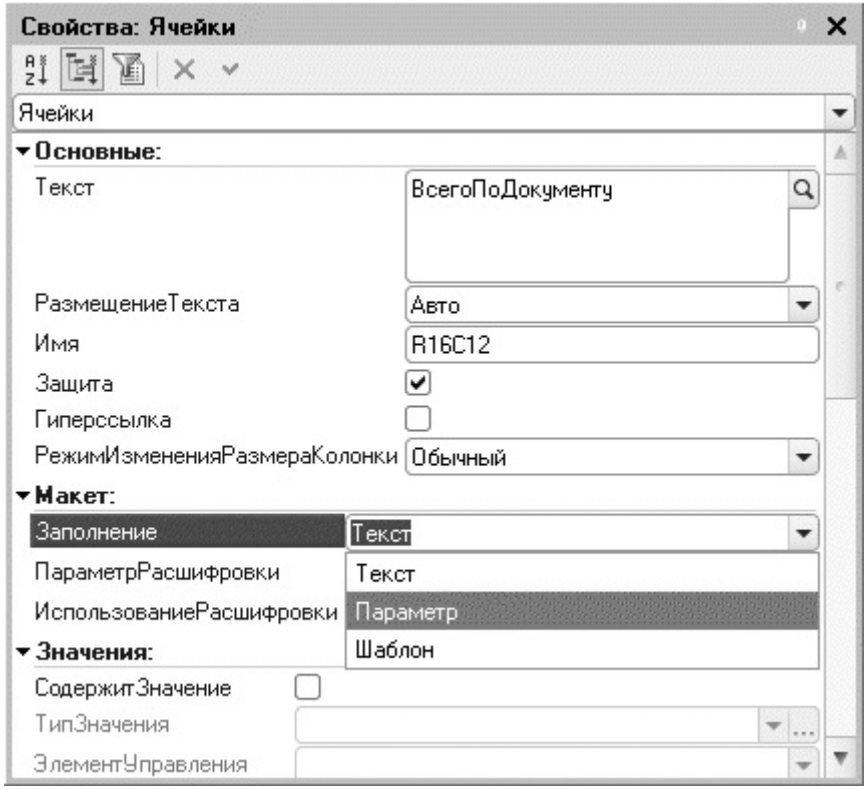


Рис. 8.13. Свойства ячейки «ВсегоПоДокументу»

Каждая ячейка редактируемого нами табличного документа может содержать либо текст, либо некоторый параметр, либо шаблон.

Текст, содержащийся в ячейке, будет показан на экране.

Параметр будет заменен некоторым значением, которое может быть присвоено ему средствами встроенного языка. Текст, содержащийся в ячейке, является именем этого параметра.

Шаблон представляет собой текстовую строку, в определенные места которой будут вставлены значения параметров.

Поэтому, указав для ячейки в качестве заполнения *Параметр*, мы определили параметр области с именем *ВсегоПоДокументу*, которому присвоим нужное нам значение при формировании печатной формы.

Откроем модуль менеджера документа *ОказаниеУслуги.*

Для этого перейдем на закладку *Прочее* окна редактирования объекта конфигурации Документ *ОказаниеУслуги* и нажмем кнопку *Модуль менеджера* (рис. 8.14).

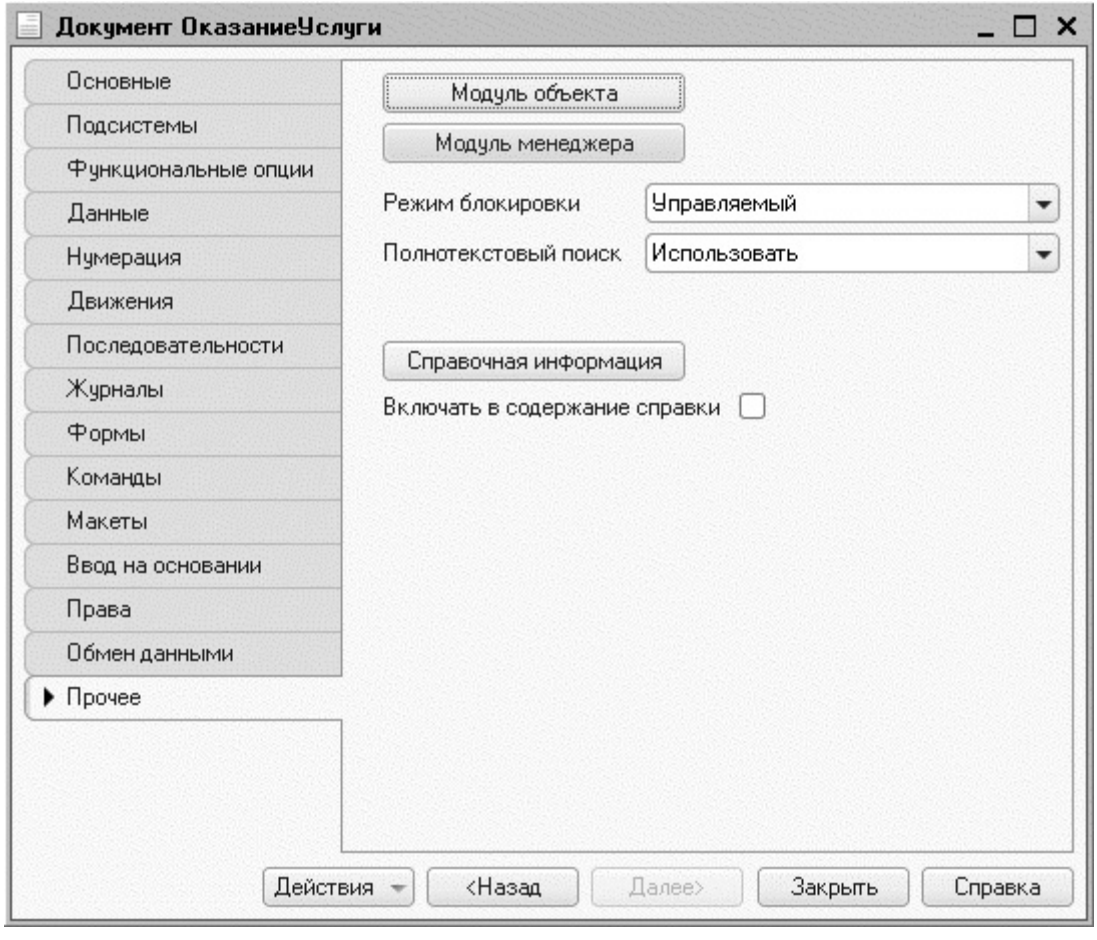


Рис. 8.14. Открытие модуля менеджера документа «ОказаниеУслуги»

Найдем в нем процедуру *Печать* и отредактируем ее следующим образом (новые строки выделены жирным шрифтом), листинг 8.1.

Листинг 8.1. Печать формы документа (фрагмент)

```
…
ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
Шапка = Макет.ПолучитьОбласть("Шапка");
ОбластьПеречеьНоменклатурыШапка =
Макет.ПолучитьОбласть("ПеречеьНоменклатурыШапка");
ОбластьПеречеьНоменклатуры = Макет.ПолучитьОбласть("ПеречеьНоменклатуры");
ОбластьИтог = Макет.ПолучитьОбласть("Всего");
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(ОбластьПеречеьНоменклатурыШапка);

  ВыборкаПеречеьНоменклатуры = Выборка.ПеречеьНоменклатуры.Выбрать();
  СуммаИтог = 0;

  Пока ВыборкаПеречеьНоменклатуры.Следующий() Цикл
    ОбластьПеречеьНоменклатуры.Параметры.Заполнить(ВыборкаПеречеьНоменклатуры);
    ТабДок.Вывести(ОбластьПеречеьНоменклатуры,
```

```
    ВыборкаПереченьНоменклатуры.Уровень());
    СуммаИтог = СуммаИтог + ВыборкаПереченьНоменклатуры.Сумма;

  КонецЦикла;

  ОбластьИтог.Параметры.ВсегоПоДокументу = СуммаИтог;
  ТабДок.Вывести(ОбластьИтог);

  ВставлятьРазделительСтраниц = Истина;

КонецЦикла;
…
```

Смысл добавленного фрагмента прост. Мы обращаемся к макету документа *ОказаниеУслуги* по его имени – *Макет.*

Используя его метод *ПолучитьОбласть(),* получаем область *Всего* (ту, которую мы только что добавили к макету) и сохраняем ее в переменной *ОбластьИтог*.

В цикле обхода строк табличной части документа, полученных в результате выполнения запроса, мы накапливаем в переменной *СуммаИтог* значение суммы табличной части документа по колонке *Сумма*.

Затем мы обращаемся к параметру *ВсегоПоДокументу*

(*ОбластьИтог.Параметры.ВсегоПоДокументу*), находящемуся в области *Всего*, и присваиваем ему значение переменной *СуммаИтог*.

В заключение мы выводим итоговую область в табличный документ, который будет показан на экране и распечатан пользователем – *ТабДок.Вывести(ОбластьИтог)*.

Отображение табличного документа на экране выполняется в обработчике команды *Печать*, в модуле этой команды на клиенте, в то время как сама процедура печати, описанная в модуле менеджера документа, выполняется на сервере.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим результат наших изменений (рис. 8.15).

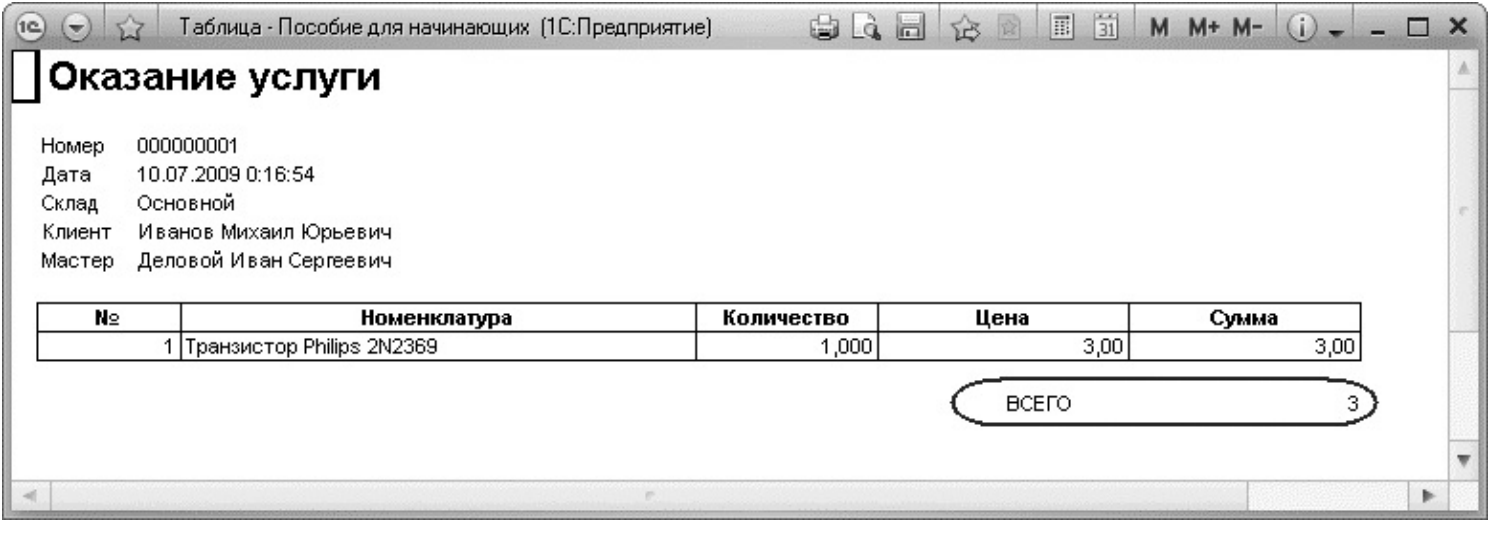


Рис. 8.15. Печатная форма документа «Оказание услуги»

Подобным образом, создавая именованные области и ячейки макета, используя их свойства и управляя порядком их вывода с помощью встроенного языка, разработчик имеет возможность создать печатную форму любого дизайна.

А теперь, для того чтобы наш документ *ОказаниеУслуги* выглядел вполне законченным, добавим итоговую сумму по документу и на экранную форму, чтобы пользователь мог видеть ее в процессе заполнения табличной части документа.

Редактирование формы

В режиме «Конфигуратор»

После того как мы вывели итоговую сумму по табличной части в печатную форму документа, возникло естественное желание видеть такую же итоговую сумму и в форме документа. Чтобы в процессе его создания можно было оперативно, не печатая документ, знать итоговую сумму по документу.

Для этого мы внесем небольшие изменения в форму документа *ОказаниеУслуги*.

Для редактирования формы документа откроем конфигуратор, раскроем дерево документа *Оказание услуги* и дважды щелкнем на форме *ФормаДокумента*.

Дважды щелкнем на элементе *ПеречеьНоменклатуры* в дереве элементов формы или правой кнопкой мыши откроем для него палитру свойств (пункт контекстного меню *Свойства*), рис. 8.16.

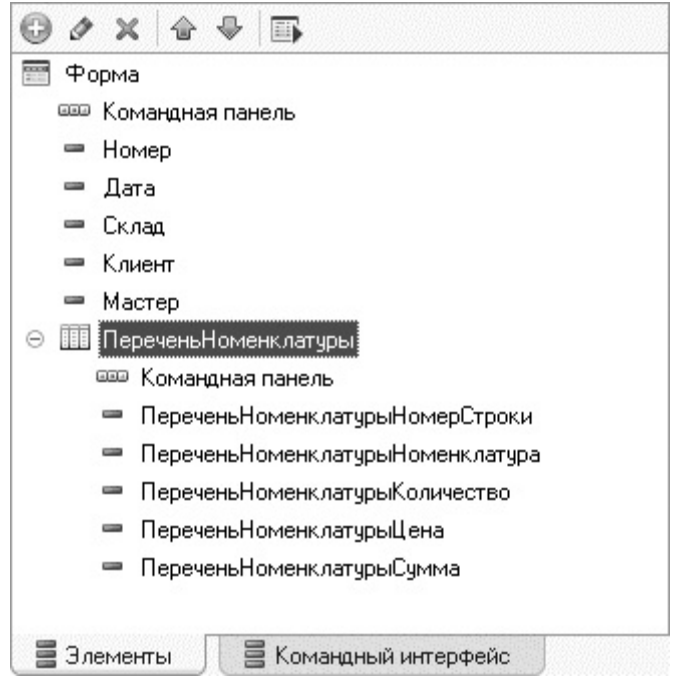


Рис. 8.16. Вызов палитры свойств табличного поля

Установим свойство *Подвал*, которое определяет наличие подвала у таблицы формы (рис. 8.17).

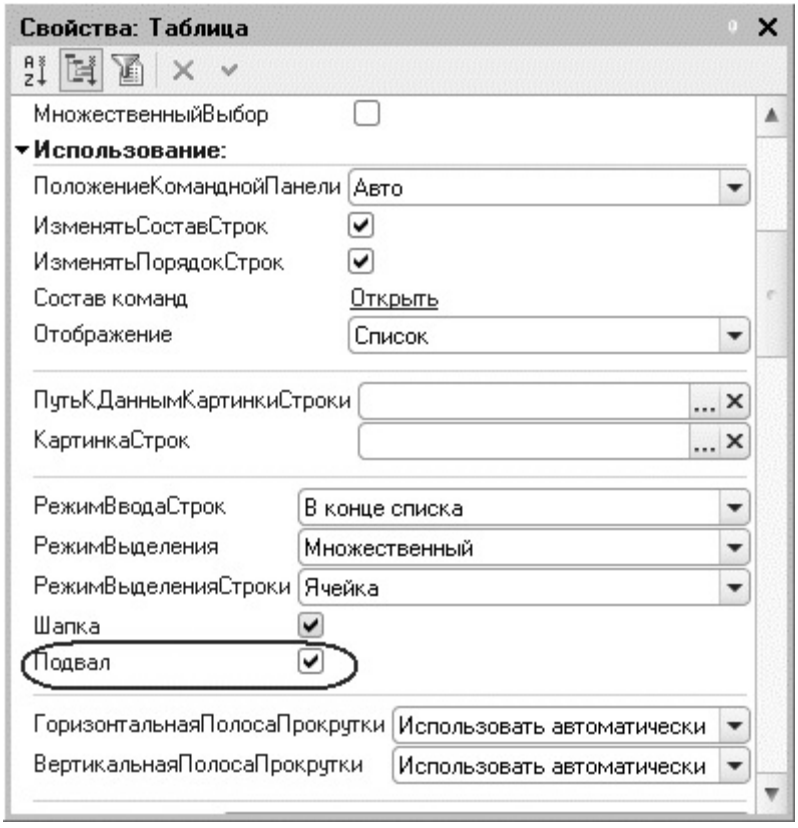


Рис. 8.17. Установка видимости подвала у табличной части

Затем откроем свойства элемента формы *ПеречеьНоменклатурыЦена* и установим:

- *Текст подвала – Всего:* (рис. 8.18);

- *Горизонтальное положение в подвале – Право* (рис. 8.19).

В свойстве *Шрифт подвала* изменим начертание на *Жирный* (рис. 8.20).

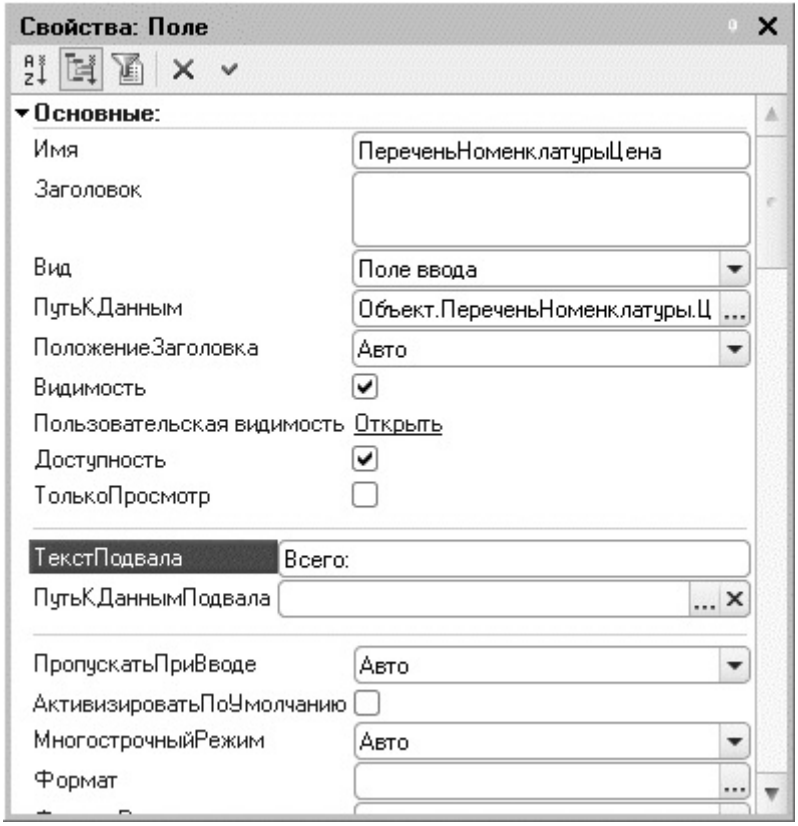


Рис. 8.18. Установка свойств колонки таблицы в подвале

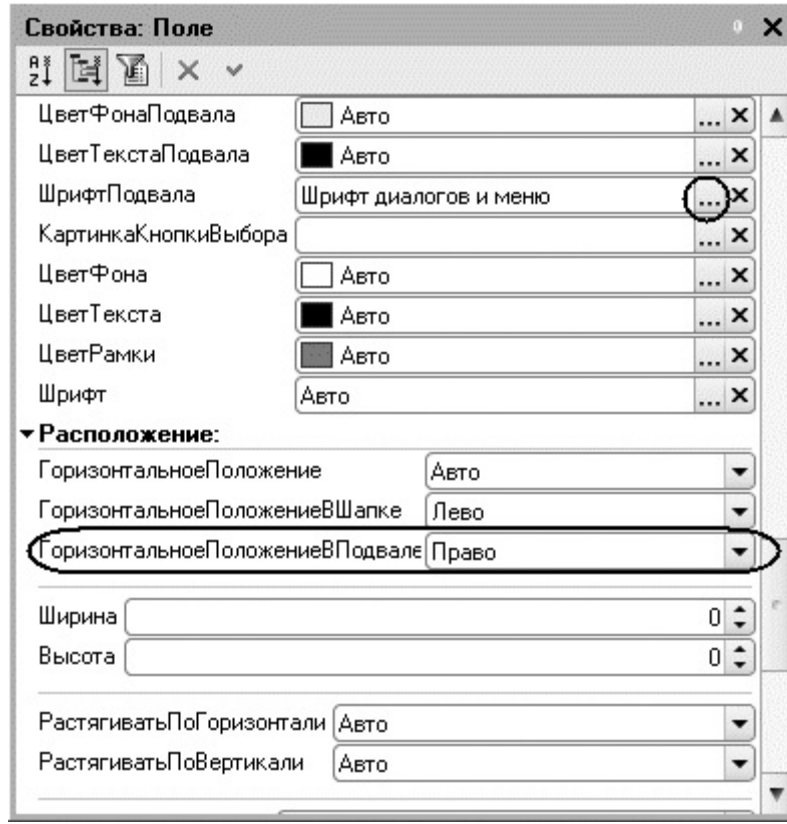


Рис. 8.19. Установка свойств колонки таблицы в подвале

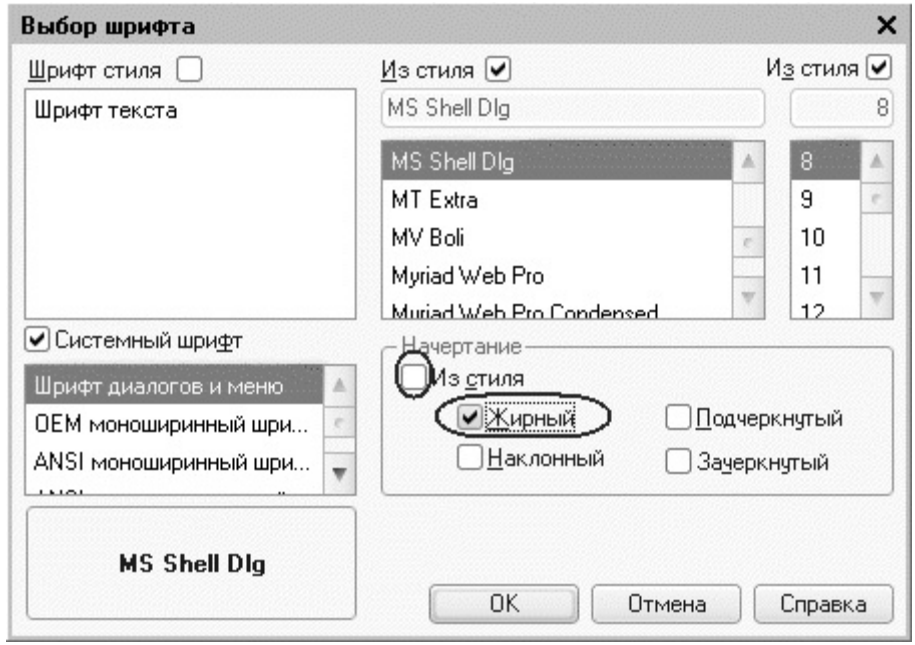


Рис. 8.20. Установка свойств колонки таблицы в подвале

После этого откроем свойства элемента *ПереченьНоменклатурыСумма*, установим: *Горизонтальное положение в подвале – Право,* в свойстве *Шрифт подвала* тоже изменим начертание на *Жирный*.

Для того чтобы в подвале колонки *Сумма* отображался итог по ней, нажмем кнопку выбора в поле *ПутьКДаннымПодвала* (рис. 8.21).

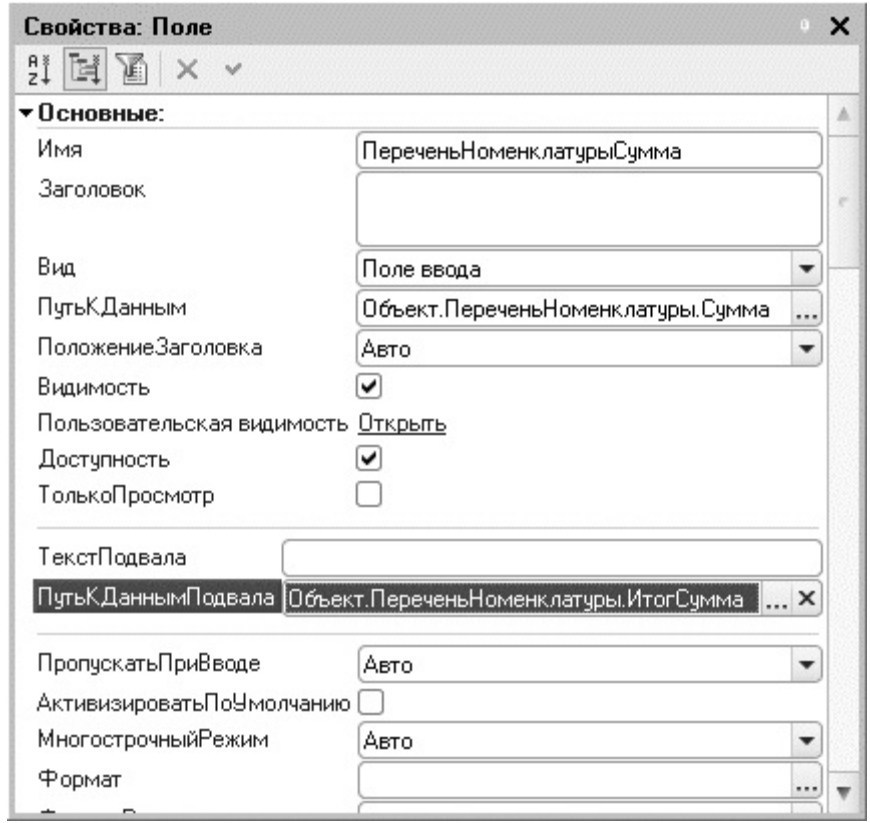


Рис. 8.21. Выбор данных подвала для колонки «Сумма»

Раскроем дерево реквизитов объекта и выберем элемент *ИтогСумма* (рис. 8.22).

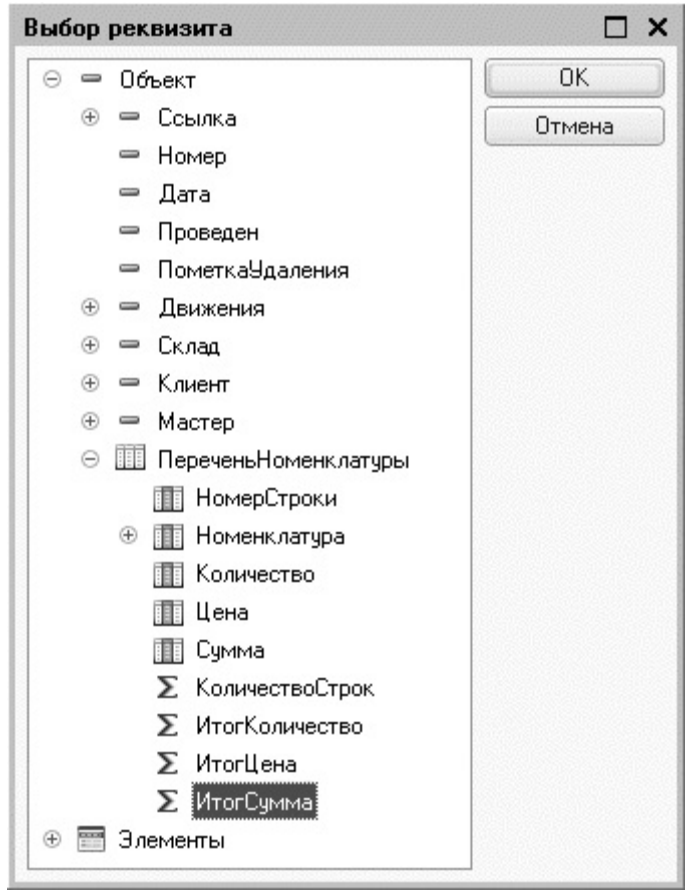


Рис. 8.22. Выбор данных подвала для колонки «Сумма»

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как теперь

выглядит форма документа *Оказание услуги № 1* (рис. 8.11).

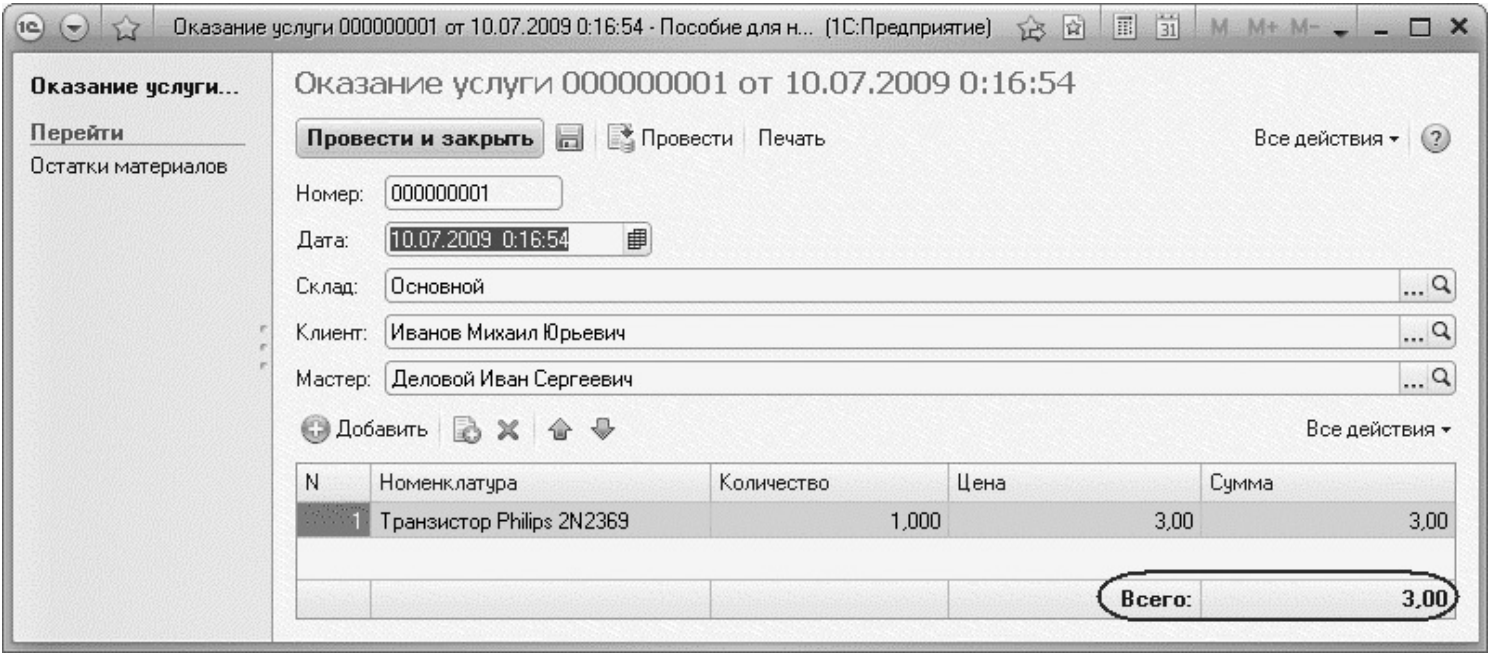


Рис. 8.23. Форма документа «Оказание услуги» с итоговой строкой

Мы видим, что по колонке *Сумма* в табличной части документа, подсчитывается общий итог документа.

Это небольшое изменение, которое мы сделали в форме, на самом деле сильно улучшает ее пользовательский интерфейс и делает ее более удобной в применении. Подобным образом, используя свойства элементов формы,

задавая их значения и изменяя командный интерфейс формы, разработчик имеет возможность создать экранную форму нужного дизайна и поведения.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Макет».*
- *Что такое конструктор печати.*
- *Как создать макет с помощью конструктора печати.*
- *Как изменить табличный документ.*
- *Какая разница в заполнении ячейки табличного документа текстом, параметром и шаблоном.*
- *Как с помощью встроенного языка вывести в табличный документ новую область.*
- *Как изменить внешний вид и поведение элемента формы.*
- *Как отобразить сумму по колонке таблицы.*

Занятие 9 (0:50). Периодические регистры сведений

Продолжительность

Ориентировочная продолжительность занятия – 50 минут.

На этом занятии мы с вами познакомимся с объектом конфигурации *Регистр сведений*, а точнее с одним из его видов – периодическим регистром сведений. Вы узнаете, для чего предназначен этот объект конфигурации и какова его структура.

Мы создадим с вами один периодический регистр сведений, который будет использоваться в нашей конфигурации, и покажем, каким образом можно использовать его данные средствами встроенного языка.

Зачем нужен периодический регистр сведений

Начнем мы с того, что обратим ваше внимание на документ *Оказание услуги*. Как вы помните, в этом документе мы выбираем услугу, которая оказывается, и затем указываем цену.

Очевидно, что в ООО «На все руки мастер» существует перечень услуг,

который определяет стоимость каждой услуги. Казалось бы, стоимость услуги является неотъемлемым свойством самой услуги, и поэтому ее следует добавить в качестве реквизита справочника *Номенклатура*.

Однако стоимость услуг имеет особенность меняться со временем. И может сложиться такая ситуация, когда нам потребуется внести изменения или уточнения в один из ранее проведенных документов *Оказание услуги*. В этом случае мы не сможем получить правильную стоимость услуги, поскольку в реквизите справочника будет храниться последнее введенное значение.

Кроме этого, не исключено, что руководство ООО «На все руки мастер» пожелает видеть зависимость прибыли предприятия от изменения стоимости оказываемых услуг. И тогда просто необходимо будет иметь возможность анализировать изменение стоимости услуг во времени.

Поэтому для хранения стоимости услуг мы используем новый пока еще для нас объект – *Регистр сведений*.

Что такое регистр сведений

Объект конфигурации *Регистр сведений* предназначен для описания структуры хранения данных в разрезе нескольких измерений. На основе объекта конфигурации *Регистр сведений* платформа создает в базе данных таблицу, в

которой может храниться произвольная информация, «привязанная» к набору измерений (рис. 9.1).

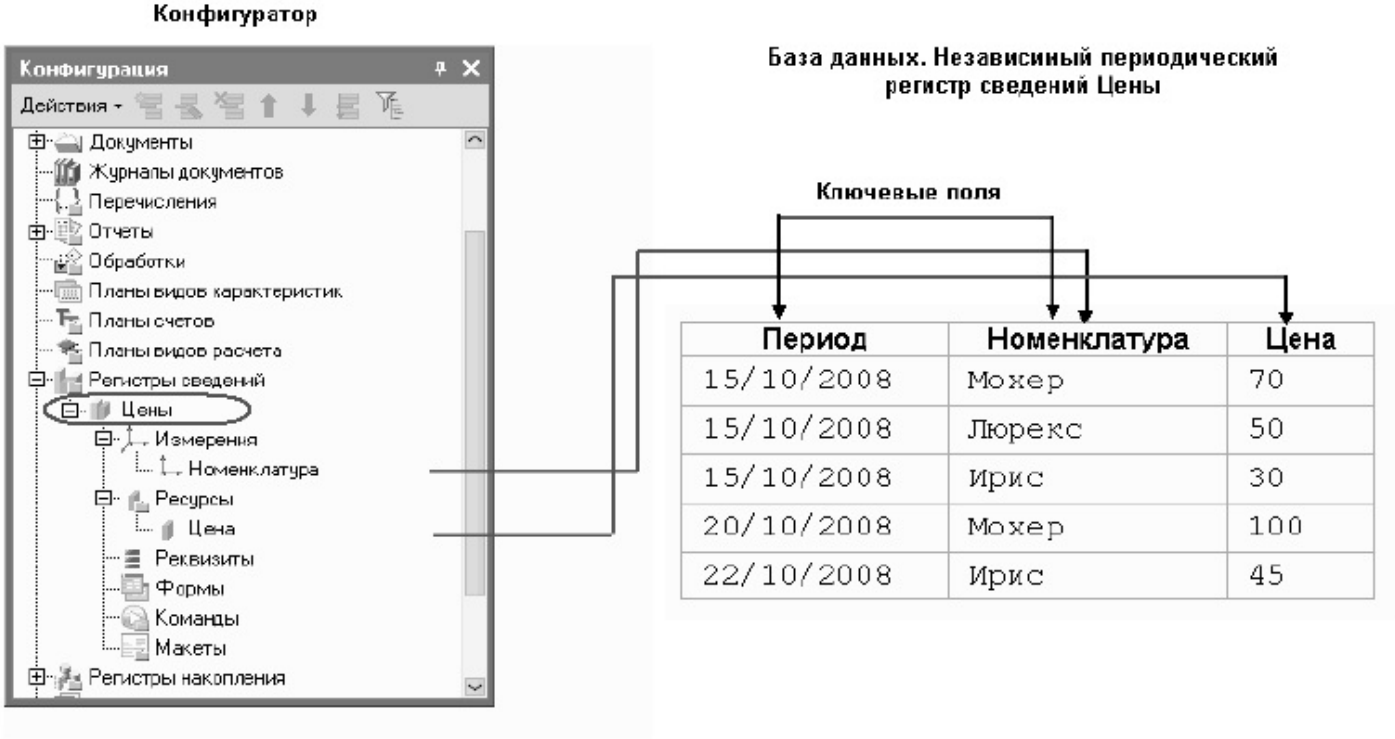


Рис. 9.1. Независимый периодический регистр сведений «Цены» в конфигураторе и в базе данных

Принципиальное отличие регистра сведений от регистра накопления заключается в том, что каждое движение регистра сведений устанавливает

новое значение ресурса, в то время как движение регистра накопления изменяет существующее значение ресурса. По этой причине регистр сведений может хранить любые данные (а не только числовые, как регистр накопления).

Следующей важной особенностью регистра сведений является его способность (при необходимости) хранить данные с привязкой ко времени. Благодаря этому регистр сведений может хранить не только актуальные значения данных, но и историю их изменения во времени. Регистр сведений, использующий привязку ко времени, называют *периодическим регистром сведений*.

Периодичность регистра сведений можно определить одним из следующих значений:

- в пределах секунды;
- в пределах дня;
- в пределах месяца;
- в пределах квартала;
- в пределах года;
- в пределах регистратора (если установлен режим записи *Подчинение регистратору*).

Периодический регистр сведений всегда содержит служебное поле *Период*,

добавляемое системой автоматически. Оно имеет тип *Дата* и служит для указания факта принадлежности записи к какому-либо периоду. При записи данных в регистр платформа всегда приводит значение этого поля к началу того периода, в который он попадает.

Например, если в регистр сведений с периодичностью в пределах месяца записать данные, в которых период указан как 08.04.2004, то регистр сохранит эти данные со значением периода, равным 01.04.2004.

Как и для других регистров, система контролирует уникальность записей для регистра сведений. Однако если для прочих регистров уникальным идентификатором записи является регистратор и номер строки, то для регистра сведений применяется другой принцип формирования ключевого значения.

Ключом записи, однозначно идентифицирующим запись, является в данном случае совокупность значений измерений регистра и периода (в случае, если регистр сведений периодический). Регистр сведений не может содержать несколько записей с одинаковыми ключами.

Если продолжать сравнение с регистром накопления, то можно сказать, что регистр сведений предоставляет больше свободы в редактировании хранимых данных. Наряду с возможностью использования в режиме подчинения регистратору (когда записи регистра сведений «привязаны» к документу-

регистратору) регистр сведений может применяться и в независимом режиме, в котором пользователю предоставляется полная свобода интерактивной работы с данными регистра. Регистр сведений, не использующий подчинение регистратору, называют *независимым регистром сведений*.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с регистрами сведений, можно прочитать в разделе «Краткий справочник разработчика. Регистры сведений».

Добавление периодического регистра сведений

Приступим к созданию периодического регистра сведений, который будет хранить развернутые во времени розничные цены материалов и стоимости услуг, оказываемых нашим ООО «На все руки мастер».

В режиме «Конфигуратор»

Откроем в конфигураторе нашу учебную конфигурацию и добавим новый объект конфигурации *Регистр сведений*.

Для этого выделим в дереве объектов конфигурации ветвь *Регистры сведений* и нажмем кнопку *Добавить* в командной панели окна конфигурации.

В открывшемся окне редактирования объекта конфигурации на закладке *Основные* зададим имя регистра – *Цены*.

Установим свойство *Периодичность* этого регистра – *В пределах секунды*.

Такую периодичность мы выбрали для того, чтобы иметь возможность отслеживать цены несколько раз в течение дня. Если же так часто не предполагается изменять цены, то можно выбрать, вообще говоря, в пределах дня.

Здесь же определим представление объекта в интерфейсе приложения.

Зададим свойства *Представление записи* как *Цена*, а *Представление списка* как *Цены на номенклатуру* (рис. 9.2).

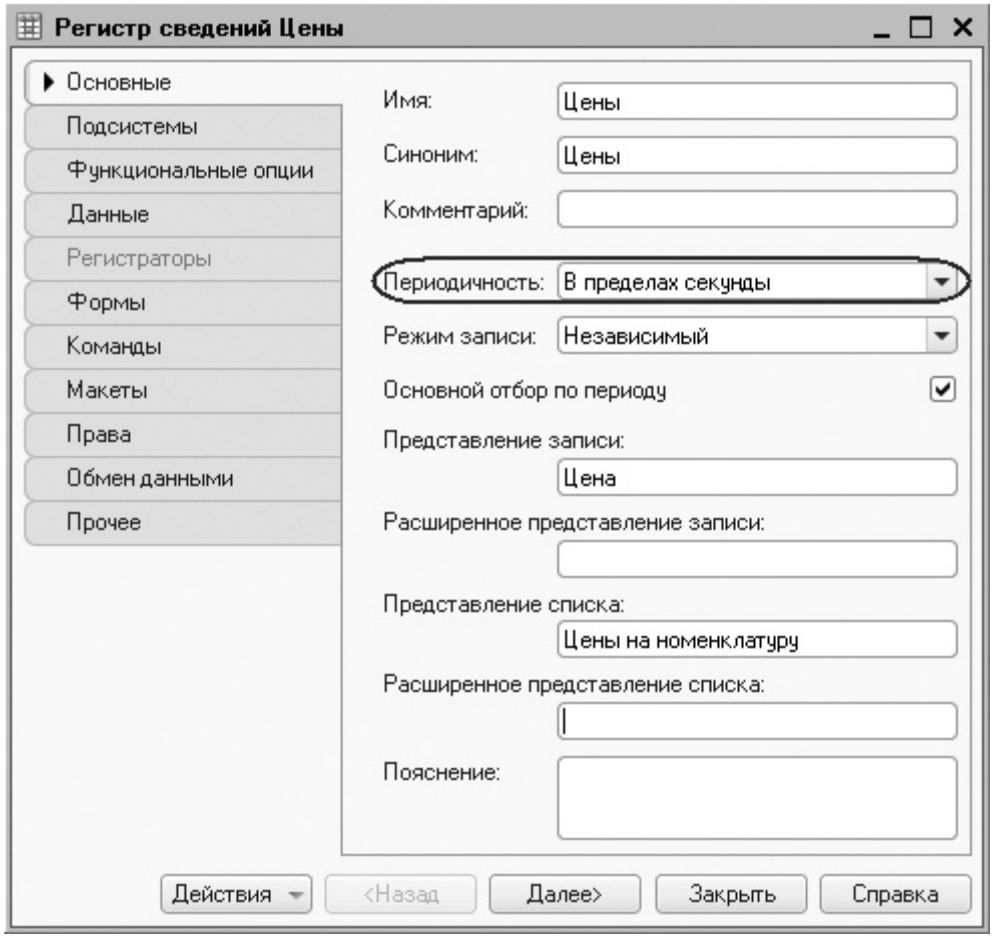


Рис. 9.2. Основные свойства регистра сведений «Цены»

Обратите внимание на свойство *Режим записи*. По умолчанию оно имеет значение – *Независимый*, то есть мы создаем независимый регистр сведений и

сможем в дальнейшем вводить в него данные без использования регистратора, вручную.

Нажмем *Далее* и перейдем на закладку *Подсистемы*.

По логике нашей конфигурации данный регистр должен быть доступен в разделах *Учет материалов*, *Оказание услуг* и *Бухгалтерия*.

Поэтому отметим в списке подсистем эти подсистемы (рис. 9.3).

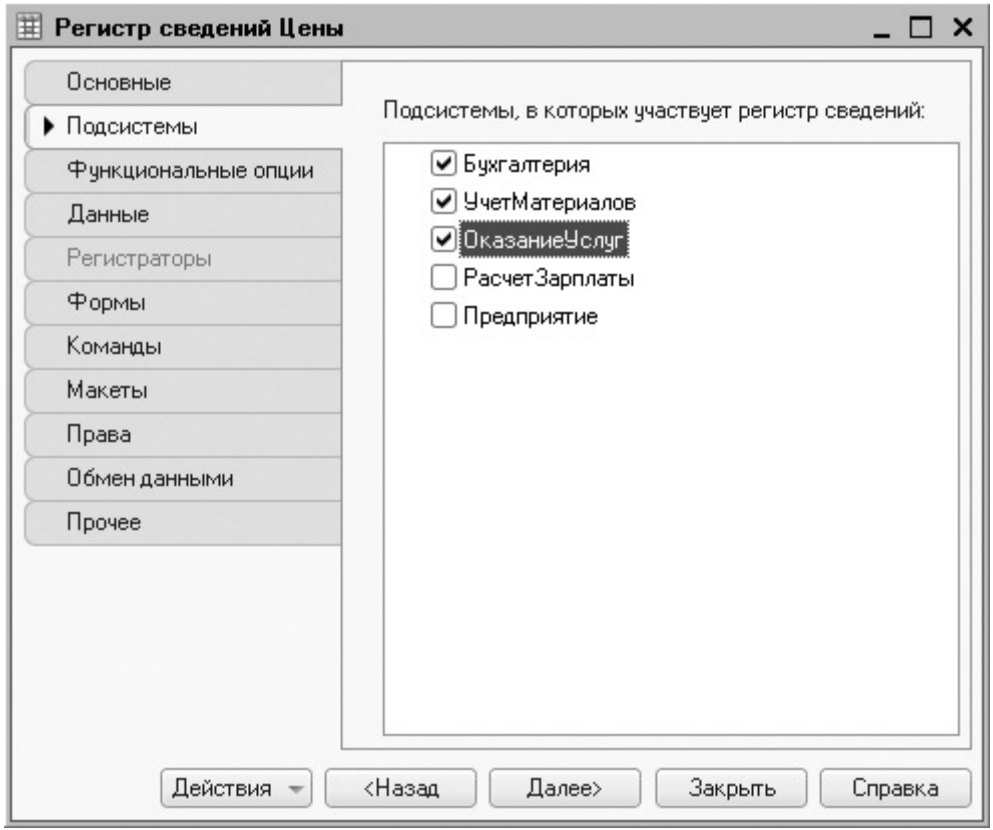


Рис. 9.3. Определение списка подсистем, в которых будет отражаться регистр

Измерения и ресурсы

Перейдем на закладку *Данные* и создадим измерение *Номенклатура* с типом *СправочникСсылка.Номенклатура*.

Для этого выделим ветвь *Измерения* и нажмем кнопку *Добавить* в командной панели окна.

Укажем, что это измерение будет ведущим (рис. 9.4).

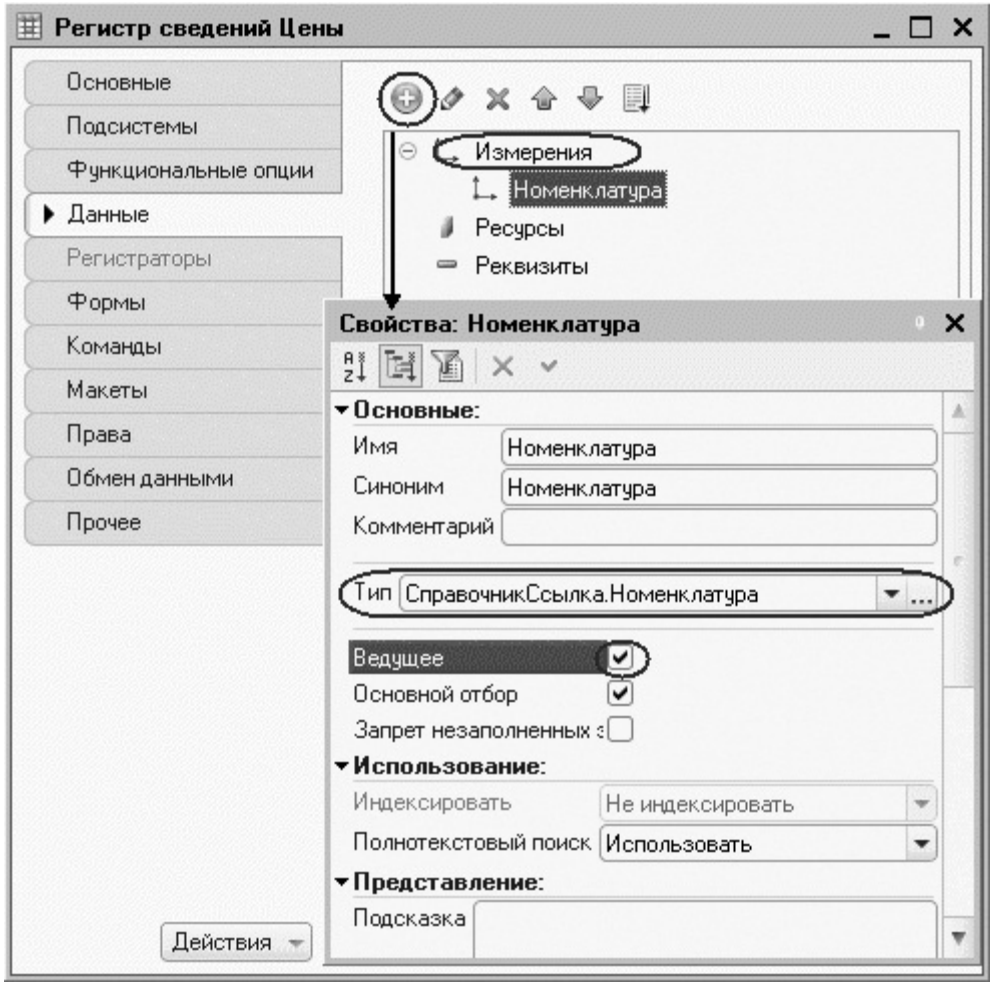


Рис. 9.4. Создание ведущего измерения регистра сведений

Свойство *Ведущее* имеет смысл использовать лишь тогда, когда измерение

имеет тип ссылки на объект базы данных. Установка свойства *Ведущее* будет говорить о том, что запись регистра сведений представляет интерес, пока существует тот объект, ссылка на который выбрана в качестве значения этого измерения в этой записи. При удалении объекта все записи регистра сведений по этому объекту тоже будут автоматически удалены.

Также в результате того, что это измерение регистра мы сделали ведущим, в форме элемента справочника *Номенклатура*, в панели навигации в группе *Перейти*, появится ссылка. По ней возможен переход к записям этого регистра, которые содержат в измерении *Номенклатура* ссылку на этот элемент справочника.

Затем создадим ресурс *Цена*, тип *Число*, длина *15*, точность *2*, неотрицательное.

Для этого выделим ветвь *Ресурсы* и нажмем кнопку *Добавить* в командной панели окна (рис. 9.5).

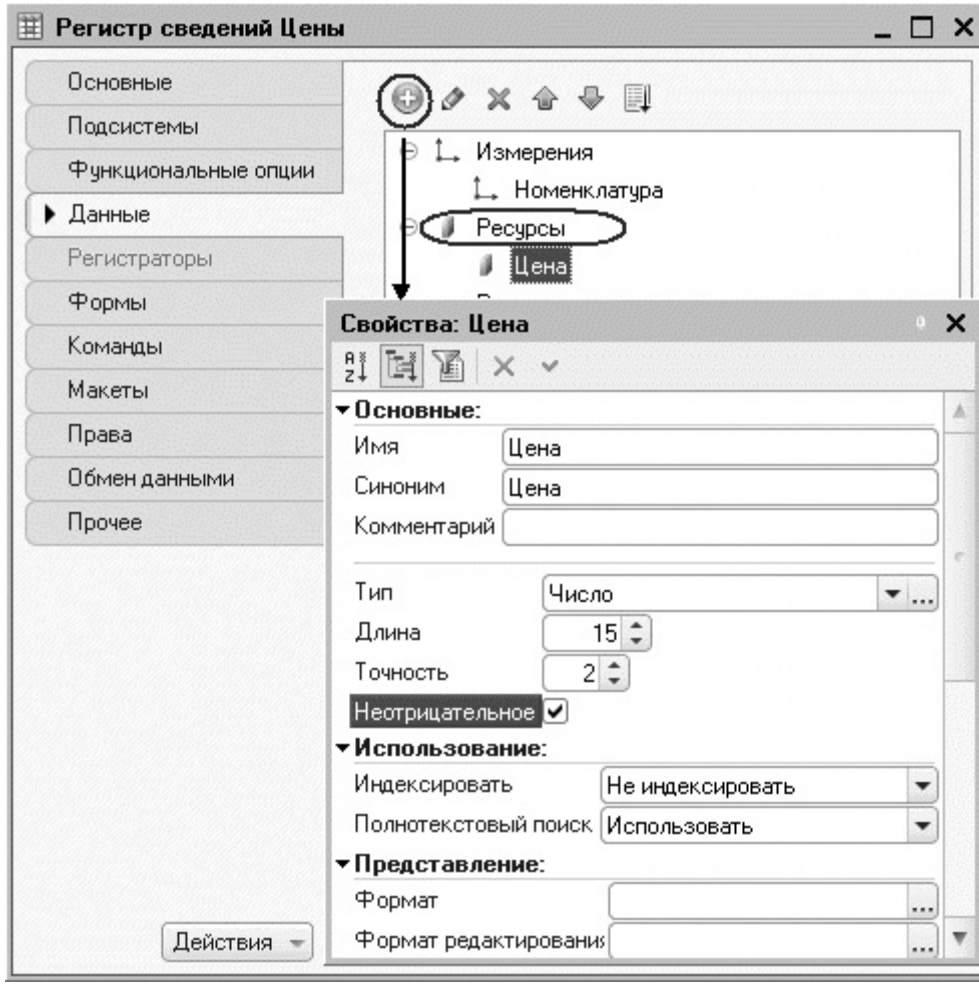


Рис. 9.5. Создание ресурса регистра сведений

В режиме «1С:Предприятие»

Теперь запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает наш периодический регистр сведений *Цены*.

В открывшемся окне «1С:Предприятия» мы видим, что в панели навигации разделов *Бухгалтерия*, *Оказание услуг* и *Учет материалов* появилась команда для открытия списка регистра *Цены на номенклатуру* (рис. 9.6).

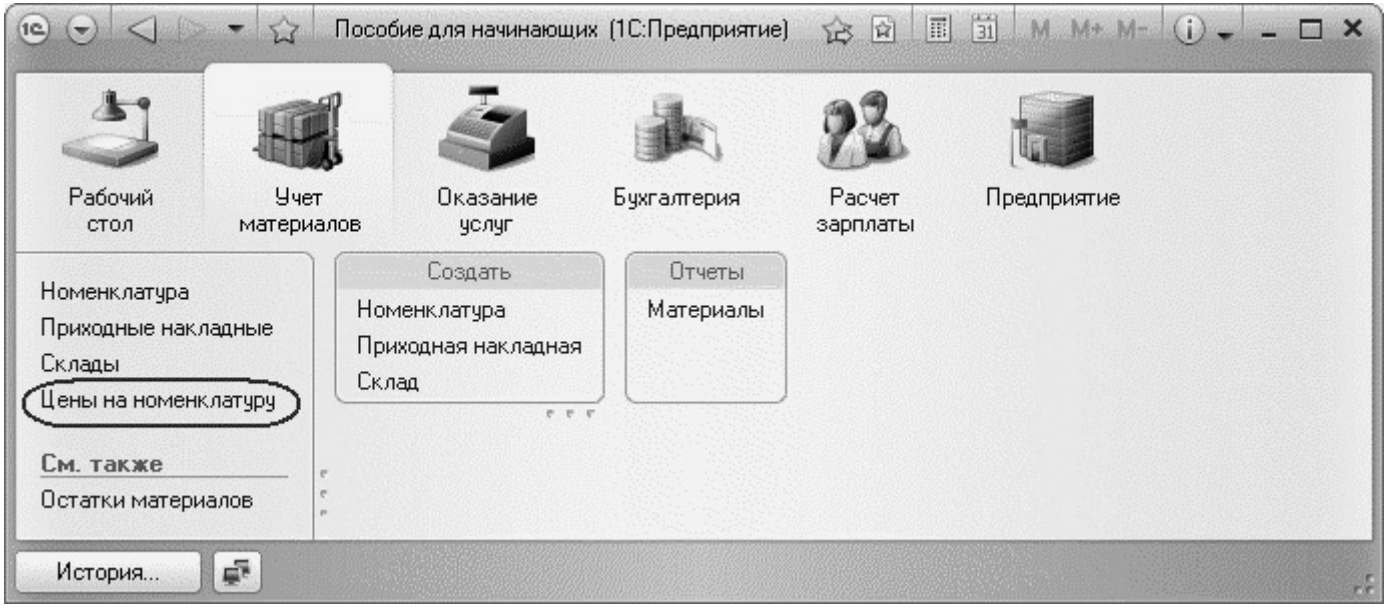


Рис. 9.6. Команда для открытия периодического регистра сведений «Цены»

Команда для открытия регистра сведений по умолчанию доступна в интерфейсе разделов, в которых отображается регистр, так как в отличие от регистров

накопления предполагается изменение данных регистра пользователем.

Создание записей в регистре сведений

Чтобы добавить новую запись в регистр сведений, нажмем кнопку *Создать.*

Зададим стоимость услуг ООО «На все руки мастер» следующим образом (рис. 9.7).

При этом период зададим задним числом, так как он должен быть меньше или равен дате создания документа об оказании услуг, в нашем случае 10.07.2009.

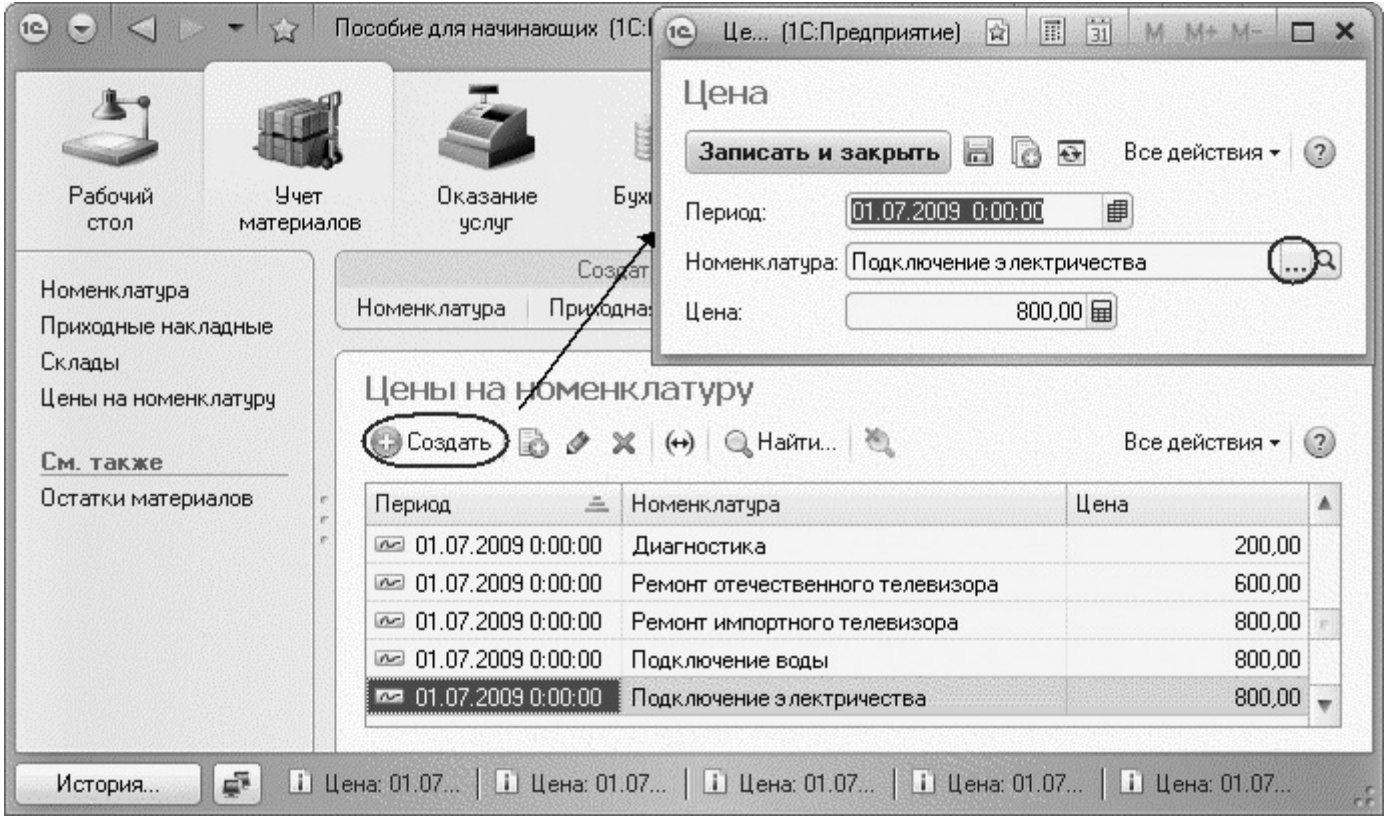


Рис. 9.7. Цены на услуги в регистре сведений «Цены»

После этого зададим розничные цены на материалы (рис. 9.8).

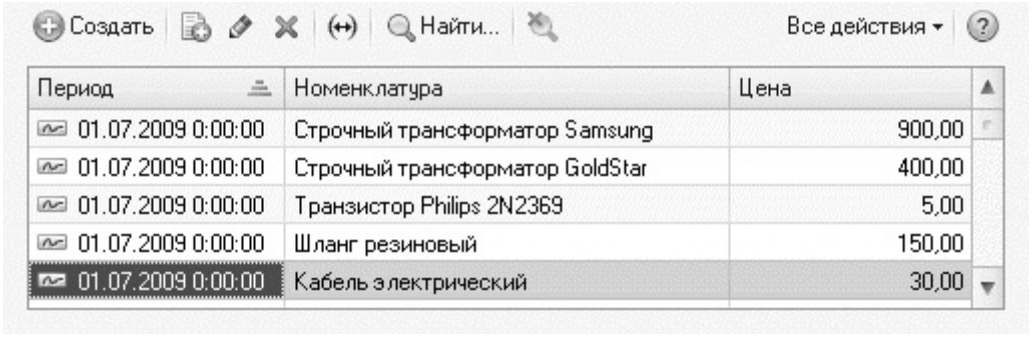


Рис. 9.8. Цены на материалы в регистре сведений «Цены»

Итак, мы с вами имеем очень полезную возможность в нашей программе – установка цен на услуги и материалы. Поскольку цены хранятся с привязкой к дате, мы можем заранее установить новые цены и быть уверены в том, что новые цены вступят в действие не раньше указанного для них времени.

Автоматическая подстановка цены в документ при выборе номенклатуры

Наша задача заключается в следующем. Цена номенклатуры у нас теперь хранится в отдельном регистре сведений. Когда мы создаем или изменяем документ *ОказаниеУслуги* и добавляем в табличную часть какую-либо номенклатуру, нам хочется, чтобы одновременно с этим в документ подставлялась бы сразу и актуальная цена этой номенклатуры, полученная из регистра сведений и соответствующая дате документа.

Для этого нам нужно сделать две вещи.

Сначала написать некую функцию, которая будет возвращать нам актуальную цену номенклатуры, а затем вызвать эту функцию в тот момент, когда в документ добавляется номенклатура, и подставить в документ цену номенклатуры, которую вернет эта функция.

Поскольку такой сервис понадобится нам, скорее всего, не только в этом документе, но и в других документах, которые содержат в табличной части номенклатуру, мы разместим функцию в некотором общедоступном месте – в общем модуле.

В режиме «Конфигуратор»

Функция, возвращающая цену номенклатуры

Сначала мы создадим функцию *РозничнаяЦена()*, которая будет возвращать нам актуальную розничную цену номенклатуры, и поместим ее в общий модуль конфигурации.

Откроем конфигуратор, в ветке *Общие* > *Общие модули* добавим новый объект конфигурации *Модуль* и назовем его *РаботаСоСправочниками*.

Мы видим, что у модуля по умолчанию установлен флажок *Сервер*. Это

означает, что экземпляры этого модуля будут скомпилированы только на стороне сервера.

Установим флажок *Вызов сервера* для того, чтобы экспортные процедуры и функции этого модуля можно было вызывать с клиента (рис. 9.9).

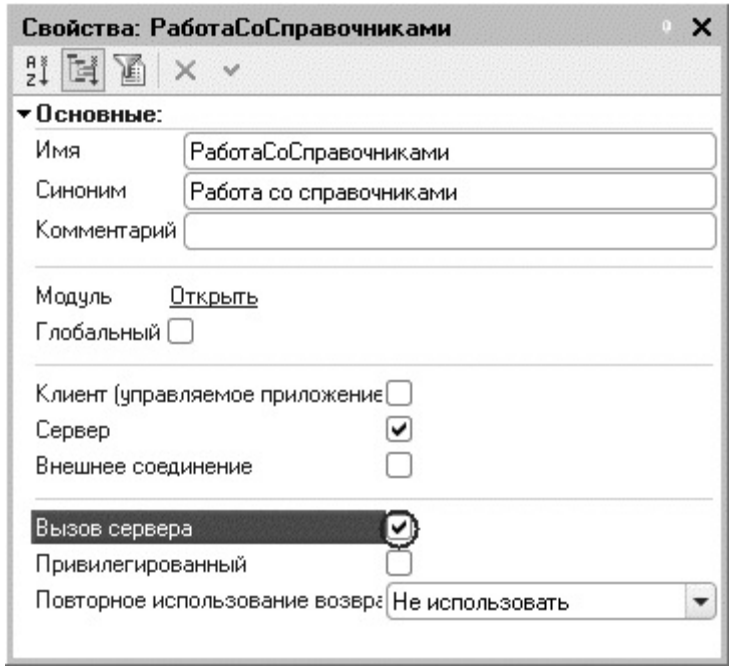


Рис. 9.9. Свойства общего модуля

Поместим в него следующий текст (листинг 9.1).

Листинг 9.1. Функция «РозничнаяЦена()»

```
Функция РозничнаяЦена(АктуальнаяДата, ЭлементНоменклатуры) Экспорт

  // Создать вспомогательный объект "Отбор".
  Отбор = Новый Структура("Номенклатура", ЭлементНоменклатуры);

  // Получить актуальные значения ресурсов регистра.
  ЗначенияРесурсов = РегистрыСведений.Цены.ПолучитьПоследнее(АктуальнаяДата,
  Отбор);

  Возврат ЗначенияРесурсов.Цена;

КонецФункции
```

Поясним эту функцию.

Для получения розничной цены мы будем передавать в функцию два параметра:

- *АктуальнаяДата* – параметр типа *Дата*, определяет точку на оси времени, в которой нас интересует значение розничной цены;
- *ЭлементНоменклатуры* – ссылка на элемент справочника *Номенклатура*, для которого мы хотим получить розничную цену.

В теле функции мы сначала создаем вспомогательный объект *Отбор*.

Это структура, содержащая отбор по измерениям регистра. С его помощью определяем, что нас будут интересовать записи регистра, в которых измерение регистра *Номенклатура* равно переданной в функцию ссылке на элемент справочника.

Имя ключа структуры (*"Номенклатура"*) должно совпадать с именем измерения регистра, заданного в конфигураторе, а значение элемента структуры (*ЭлементНоменклатуры*) задает отбираемое по данному измерению значение.

Во второй строке мы обращаемся к менеджеру регистра сведений *Цены* (*РегистрыСведений.Цены*) и выполняем метод *ПолучитьПоследнее()*, который возвращает нам значения ресурсов самой поздней записи регистра, соответствующей передаваемой в функцию дате (*АктуальнаяДата*) и значениям измерений регистра (*Отбор*).

Метод *ПолучитьПоследнее* возвращает структуру, содержащую значения ресурсов, которая сохраняется в переменной *ЗначенияРесурсов*. Вообще говоря, у регистра может быть несколько ресурсов. В нашем регистре ресурс один, но все равно будет возвращена структура, содержащая единственный элемент.

Поэтому в следующей строке мы получаем искомую нами розничную цену, просто указав имя нужного нам ресурса регистра через точку

(*ЗначенияРесурсов.Цена*) и возвращаем ее при выполнении функции.

Теперь эту функцию нужно вызвать в некоторый момент работы документа.

Вызов функции при выборе номенклатуры и заполнение цены в документе

Итак, задача, которая перед нами стоит, заключается в следующем. При редактировании документа *ОказаниеУслуги* нам необходимо обеспечить автоматическое заполнение поля *Цена* после того, как пользователь выберет услугу. Причем цена услуги должна определяться исходя из даты создаваемого документа.

Найдем в конфигураторе документ *ОказаниеУслуги* и откроем его форму *ФормаДокумента*.

Дважды щелкнем на элементе формы *ПеречеьНоменклатурыНоменклатура* или правой кнопкой мыши откроем для него палитру свойств (пункт контекстного меню *Свойства*). Прокрутив список до конца, найдем событие *ПриИзменении*, которое возникает после изменения значения поля.

Нажмем кнопку открытия  со значком лупы в поле ввода.

Система создаст шаблон процедуры обработчика этого события в модуле нашей формы и откроет закладку *Модуль* редактора формы.

Внесем в него следующий текст (листинг 9.2).

Листинг 9.2. Процедура ПеречеьНоменклатурыНоменклатураПриИзменении()

```
// Получить текущую строку табличной части.
СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;

// Установить цену.
СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена(Объект.Дата,
СтрокаТабличнойЧасти.Номенклатура);

// Пересчитать сумму строки
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
```

Прокомментируем содержимое обработчика.

Первая строка обработчика вам уже знакома по процедурам *ПеречеьНоменклатурыКоличествоПриИзменении* и *ПеречеьНоменклатурыЦенаПриИзменении*. Сначала мы получаем текущую строку табличной части документа, так как она нам понадобится в дальнейшем, и сохраняем ее в переменной *СтрокаТабличнойЧасти*.

Затем мы вызываем нашу функцию *РозничнаяЦена()* из общего модуля *РаботаСоСправочниками*.

Первым параметром мы передаем в эту функцию дату документа, на которую необходимо получить цену. Дату документа мы получаем из основного реквизита формы – *Объект.Дата*.

Вторым параметром мы передаем ссылку на элемент справочника *Номенклатура*, который содержится в текущей строке табличной части документа (*СтрокаТабличнойЧасти.Номенклатура*).

Функция возвращает последнее значение цены, и это значение мы присваиваем полю *Цена* в текущей строке табличной части документа (*СтрокаТабличнойЧасти.Цена*).

Затем мы вызываем процедуру *РассчитатьСумму* из общего модуля *РаботаСДокументами*. Эту процедуру мы создали с вами на предыдущих занятиях для того, чтобы при изменении цены или количества в документе пересчитывать сумму в строке документа.

Заметьте, что сама процедура *ПереченьНоменклатурыНоменклатураПриИзменении()* начинает работать в модуле формы на стороне клиента, так как это обработчик интерактивного события формы. Создавая заготовку этой процедуры, платформа автоматически разместила перед описанием процедуры директиву компиляции *&НаКлиенте*.

Затем вызывается функция *РозничнаяЦена()*. Поскольку эта функция не будет найдена на стороне клиента, то исполнение будет передано в общий модуль *РаботаСоСправочниками*, который выполняется на сервере. После завершения функции программный код продолжит исполняться на клиенте.

Почему в данном случае использована такая хитрость? Зачем было передавать исполнение кода на сервер?

Дело в том, что любая работа с базой данных (чтение данных, запись) возможна только на сервере. В данном случае нам необходимо было прочитать последние данные из регистра сведений для некоторой номенклатуры.

Такие действия можно выполнить только на сервере, и если посмотреть в синтакс-помощнике описание метода *ПолучитьПоследнее()* регистра сведений, то можно заметить, что этот метод доступен только на сервере, в толстом клиенте и во внешнем соединении.

Толстый клиент и внешнее соединение – это клиентские приложения прежней версии платформы, которые существуют для совместимости с прежними прикладными решениями.

Мы же с вами разрабатываем совершенно новое прикладное решение, которое работает в тонком клиенте или в веб-клиенте. Поэтому в нашем случае для

получения каких-либо данных из базы данных нужно передать исполнение кода на сервер, там получить нужные данные и вернуть эти данные на клиента. Что мы и сделали.

В режиме «1С:Предприятие»

Проверим, как теперь работает наш документ.

Запустим «1С:Предприятие» в режиме отладки и откроем регистр сведений *Цены*.

Для транзистора *Philips* добавим другим числом новую цену (рис. 9.10).

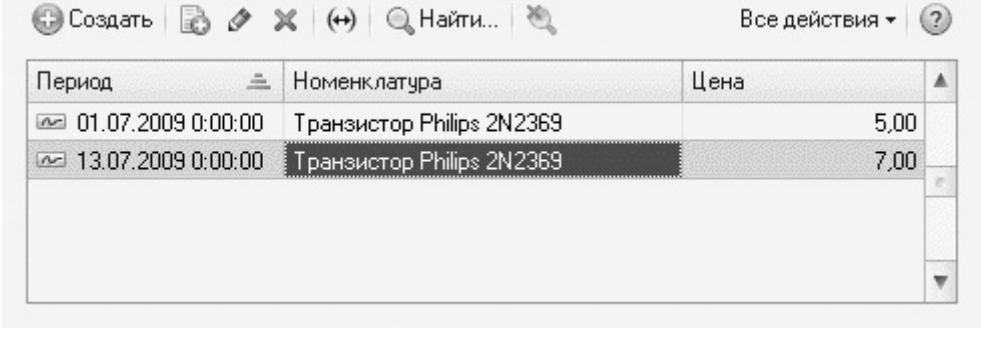


Рис. 9.10. Регистр сведений «Цены»

Теперь откроем документ *Оказание услуги № 1*. Как вы помните, этим документом мы как раз «израсходовали» один такой транзистор.

Оставим дату документа без изменения и повторим выбор транзистора в колонке *Номенклатура* табличной части документа. Автоматически установится значение цены транзистора от 01.07.2009. Это последнее значение цены на дату документа (рис. 9.11).

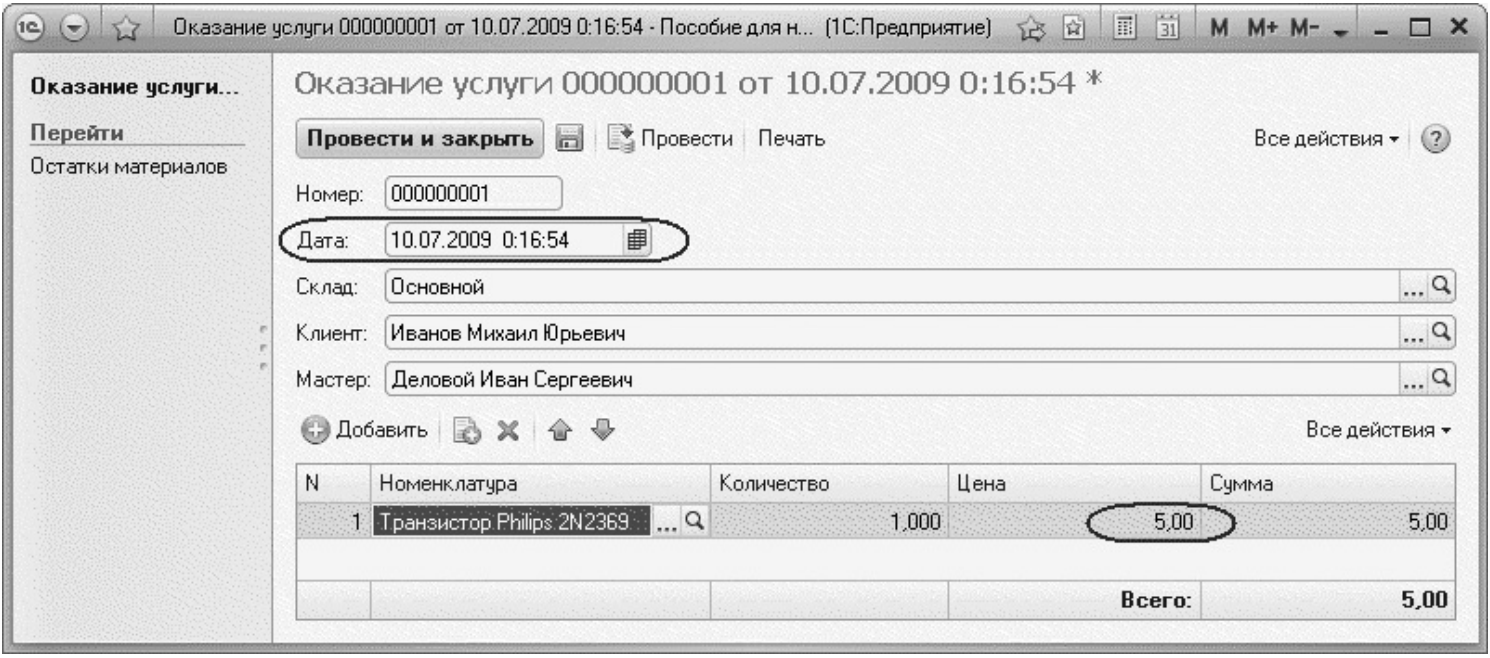


Рис. 9.11. Заполнение документа «Оказание услуги»

Теперь изменим дату документа на 13.07.2009 и снова повторим выбор транзистора. Будет установлено новое значение цены, последнее на эту дату (рис. 9.12).

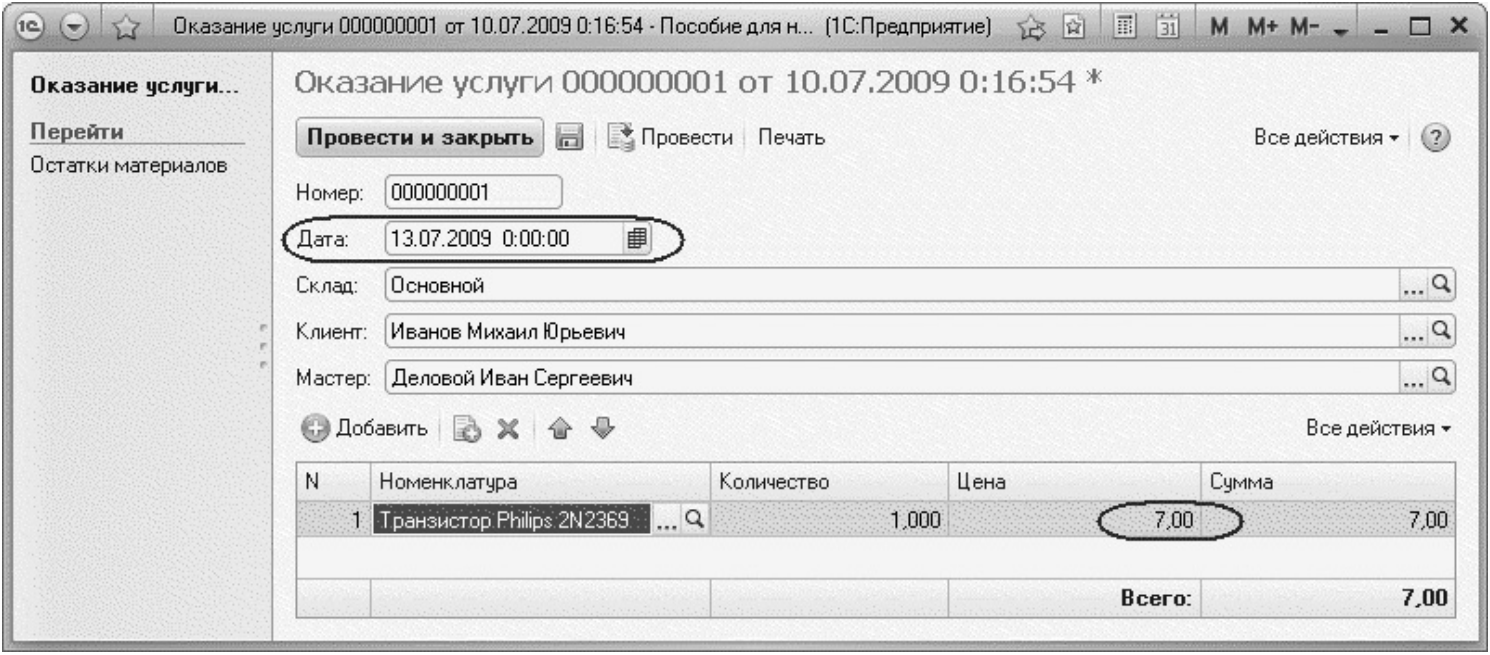


Рис. 9.12. Заполнение документа «Оказание услуги»

Таким образом, в документе появляется актуальная на момент создания документа цена услуги.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Регистр сведений».*
- *Какими особенностями обладает объект конфигурации «Регистр*

сведений».

- *В чем главные отличия регистра сведений от регистра накопления.*
- *Какие поля определяют ключ уникальности регистра накопления.*
- *Что такое периодический регистр сведений и что такое независимый регистр сведений.*
- *Как создать периодический регистр сведений.*
- *Что такое ведущее измерение регистра.*
- *Как получить значения ресурсов наиболее поздних записей регистра средствами встроенного языка.*

Занятие 10 (0:30). Перечисления

Продолжительность

Ориентировочная продолжительность занятия – 30 минут.

До сих пор мы с вами не обращали внимания на то, что у нас нет никакого признака, по которому мы могли бы сказать, чем является конкретный элемент справочника *Номенклатура*: материалом или услугой. То, что все элементы справочника разложены у нас по некоторым группам, не может являться надежным критерием: группы можно удалить, переименовать, сгруппировать элементы по другим принципам…

Поэтому нам требуется некоторый признак, позволяющий однозначно определять принадлежность элемента справочника к материалам или услугам, независимо от изменения иерархической структуры справочника.

На этом занятии мы создадим у справочника *Номенклатура* специальный реквизит, тип значения которого образуется новым пока еще для нас объектом конфигурации *Перечисление*. Это поможет нам в дальнейшем легко определять, чем является элемент справочника *Номенклатура*: услугой или материалом.

Кроме этого, мы скорректируем процедуру проведения документа *Оказание услуги* и покажем, как работать с перечислением средствами встроенного языка.

Что такое перечисление

Объект конфигурации *Перечисление* предназначен для описания структуры хранения постоянных наборов значений, не изменяемых в процессе работы конфигурации. На основе объекта конфигурации Перечисление платформа создает в базе данных таблицу, в которой может храниться набор некоторых постоянных значений.

В реальной жизни этому объекту может соответствовать, например, перечисление вариантов указания цены («включая НДС», «без НДС»). Набор всех возможных значений, которые содержит перечисление, задается при конфигурировании системы, и пользователь не может изменять их, удалять или добавлять новые.

Из этого следует важная особенность перечисления: значения перечисления не «обезличены» для конфигурации, на них могут опираться алгоритмы работы программы.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с перечислениями, можно прочитать в разделе «Краткий справочник разработчика. Перечисления».

Добавление перечисления

В режиме «Конфигуратор»

Откроем конфигуратор и создадим сначала новый объект конфигурации *Перечисление* с именем *ВидыНоменклатуры*.

На закладке *Данные* добавим два значения перечисления: *Материал* и *Услуга*.

Для этого нажмем кнопку *Добавить* над списком значений перечисления (рис. 10.1).

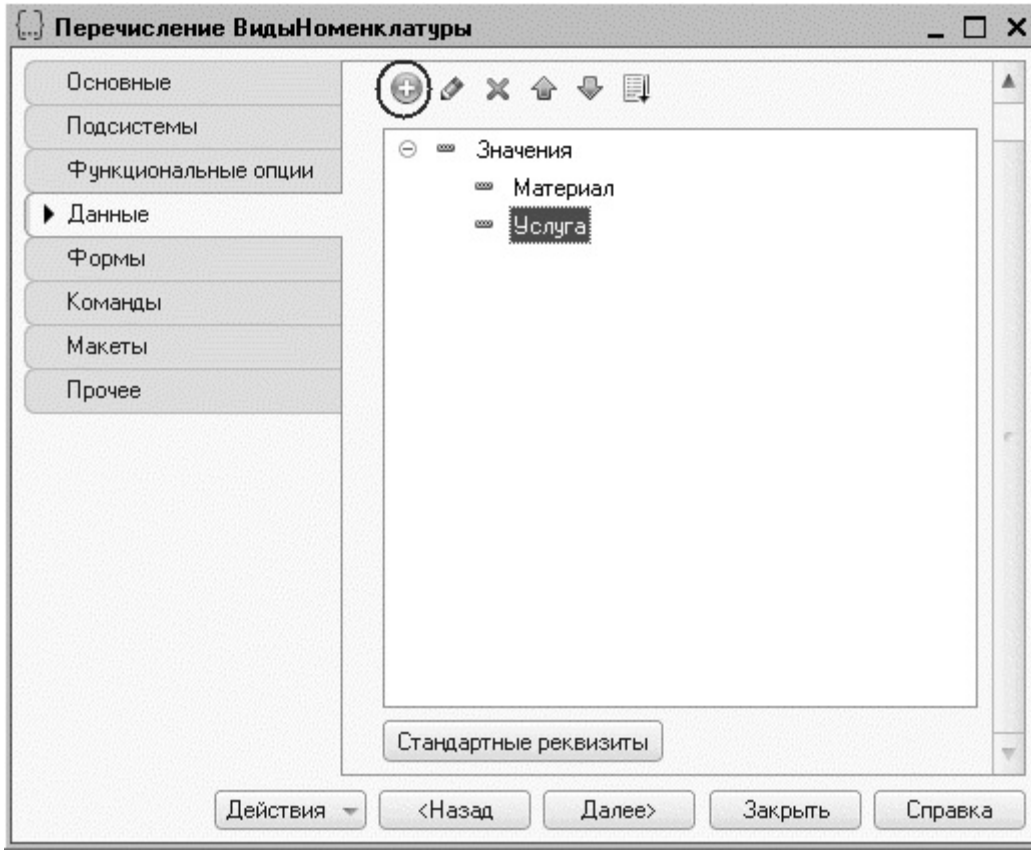


Рис. 10.1. Данные перечисления «ВидыНоменклатуры»

Привязка номенклатуры к значениям перечисления «ВидНоменклатуры»

Чтобы привязать номенклатуру к значениям перечисления, мы сделаем следующее:

- в режиме *Конфигуратор* создадим у справочника *Номенклатура* реквизит, который будет хранить значение перечисления;
- в режиме *1С:Предприятие* проставим нужные значения этого реквизита для всех элементов справочника *Номенклатура*.

В режиме «Конфигуратор»

Добавим в справочник *Номенклатура* новый реквизит *ВидНоменклатуры* с типом *ПеречислениеСсылка.ВидыНоменклатуры*.

Для этого откроем окно редактирования объекта конфигурации Справочник *Номенклатура* и на закладке *Данные* нажмем кнопку *Добавить* над списком реквизитов справочника (рис. 10.2).

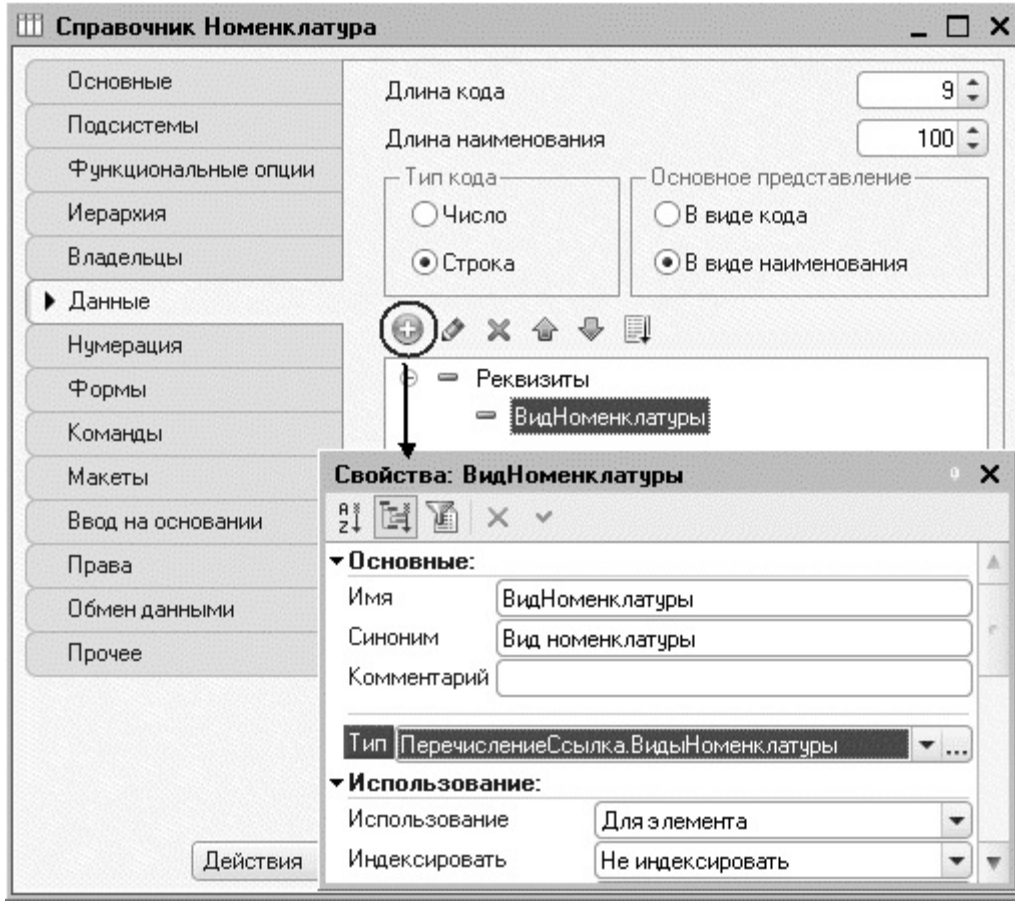


Рис. 10.2. Данные справочника «Номенклатура»

В режиме «1С:Предприятие»

После этого запустим «1С:Предприятие» в режиме отладки.

Платформа выдаст предупреждающее сообщение о том, что наше перечисление не включено ни в одну подсистему. Проигнорируем его и примем изменения конфигурации.

В режиме *1С:Предприятие* зададим для каждого элемента справочника *Номенклатура* соответствующее значение реквизита *Вид номенклатуры* (рис. 10.3).

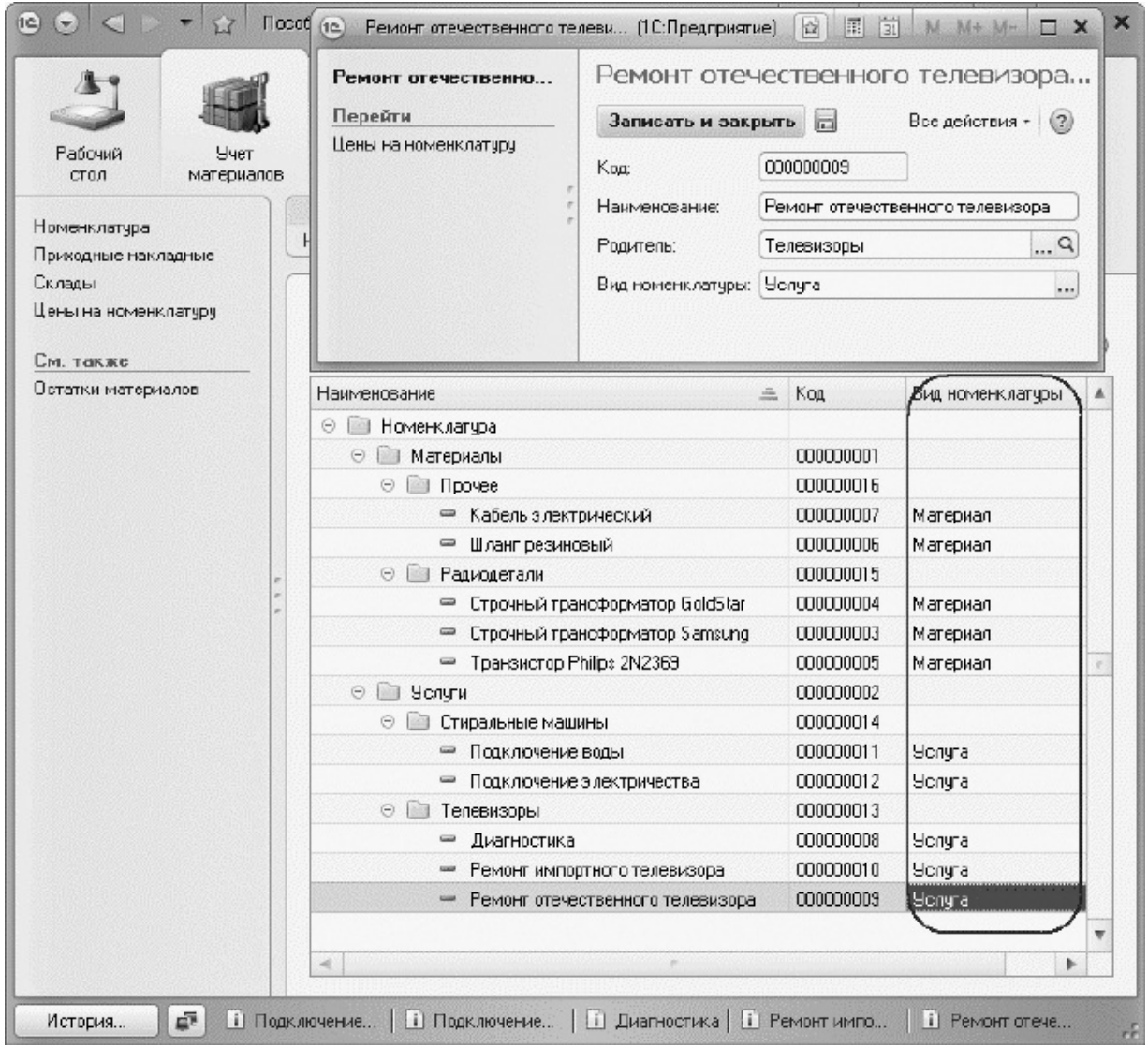

Рабочий стол
Учет материалов
Номенклатура
Приходные накладные
Склады
Цены на номенклатуру
См. также
Остатки материалов
Ремонт отечественного телеви... (1С:Предприятие)
Ремонт отечественно...
Перейти
Цены на номенклатуру
Ремонт отечественного телевизора...
Записать и закрыть
Все действия
Код: 000000009
Наименование: Ремонт отечественного телевизора
Родитель: Телевизоры
Вид номенклатуры: Услуга
Наименование
Код
Вид номенклатуры
Номенклатура
Материалы 000000001
Прочее 000000016
Кабель электрический 000000007 Материал
Шланг резиновый 000000006 Материал
Радиодетали 000000015
Строчный трансформатор GoldStar 000000004 Материал
Строчный трансформатор Samsung 000000003 Материал
Транзистор Philips 2N2369 000000005 Материал
Услуги 000000002
Стиральные машины 000000014
Подключение воды 000000011 Услуга
Подключение электричества 000000012 Услуга
Телевизоры 000000013
Диагностика 000000008 Услуга
Ремонт импортного телевизора 000000010 Услуга
Ремонт отечественного телевизора 000000009 Услуга
История...
Подключение...
Подключение...
Диагностика
Ремонт импо...
Ремонт отече...

Рис. 10.3. Данные справочника «Номенклатура»

Теперь посмотрим, как можно применить новые данные, полученные благодаря использованию перечисления *ВидыНоменклатуры*.

Регистрация расхода только той номенклатуры, которая является материалом

Если вы помните, на шестом занятии, когда создавались движения документа *ОказаниеУслуги* по регистру накопления *ОстаткиМатериалов*, мы сказали, что они не совсем правильные, поскольку в регистр будут попадать не только записи об израсходованных материалах, но и записи об оказанных услугах (листинг 10.1).

Листинг 10.1. Процедура «ОбработкаПроведения()» документа «ОказаниеУслуги»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
  // Данный фрагмент построен конструктором.
  // При повторном использовании конструктора внесенные вручную изменения будут
  утеряны!!!
  Движения.ОстаткиМатериалов.Записывать = Истина;

  Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл
    // регистр ОстаткиМатериалов Расход
```

```
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаПеречеНьНоменклатуры.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаПеречеНьНоменклатуры.Количество;
  КонецЦикла;
  //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

КонецПроцедуры
```

Теперь мы доработаем документ таким образом, чтобы в регистре появлялись только записи, относящиеся к расходу материалов.

Для этого мы сначала в режиме *Конфигуратор* изменим процедуру проведения документа так, чтобы в регистр попадали записи только о той номенклатуре, которая является материалом, а потом в режиме *1С:Предприятие* заново проведем (перепроведем) все документы *Оказание услуги*, чтобы данные в регистре изменились в соответствии с новым алгоритмом проведения документа.

Эта доработка будет не совсем эффективна с точки зрения производительности, зато позволит нам получить нужные данные в регистре *ОстаткиМатериалов*.

ПРИМЕЧАНИЕ

Более эффективный вариант обработки проведения этого документа мы рассмотрим после изучения занятия №14, рассказывающего о механизме запросов «1С:Предприятия».

В режиме «Конфигуратор»

Скорректируем движения документа, исключив из обработки те строки табличной части, в которых находятся услуги.

Для этого откроем в конфигураторе модуль документа *ОказаниеУслуги* (контекстное меню документа – *Открыть модуль объекта*) и добавим в обработчик события *ОбработкаПроведения* это условие.

Текст следует добавить в начало цикла обхода табличной части документа после строки *Для Каждого ТекСтрокаПеречеьНоменклатуры Из ПеречеьНоменклатуры Цикл*.

В результате процедура *ОбработкаПроведения* должна выглядеть следующим образом (листинг 10.2).

Листинг 10.2. Движения документа «ОказаниеУслуги»

```
Процедура ОбработкаПроведения(Отказ, Режим)
```

```
  //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
  // Данный фрагмент построен конструктором.
  // При повторном использовании конструктора внесенные вручную изменения будут
  утеряны!!!
  Движения.ОстаткиМатериалов.Записывать = Истина;

  Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

    Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
    Перечисления.ВидыНоменклатуры.Материал Тогда
      // регистр ОстаткиМатериалов Расход
      Движение = Движения.ОстаткиМатериалов.Добавить();
      Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
      Движение.Период = Дата;
      Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
      Движение.Склад = Склад;
      Движение.Количество = ТекСтрокаПереченьНоменклатуры.Количество;

    КонецЕсли;

  КонецЦикла;
  //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

КонецПроцедуры
```

Добавленный текст исключает выполнение операторов цикла для тех строк табличной части документа, в которых номенклатура не является материалом. Поясним это условие.

В переменной *ТекСтрокаПеречеНоменклатуры* содержатся на каждом шаге цикла данные текущей строки табличной части *ПеречеНоменклатуры*.

Указывая через точку имя колонки *Номенклатура* (*ТекСтрокаПеречеНоменклатуры.Номенклатура*), мы обращаемся к ссылке на элемент номенклатуры, которая содержится в этой строке табличной части.

Затем, указывая через точку *ВидНоменклатуры* *ТекСтрокаПеречеНоменклатуры.Номенклатура.ВидНоменклатуры*), мы обращаемся к реквизиту *ВидНоменклатуры* этого элемента справочника *Номенклатура*.

Полученное значение с помощью оператора сравнения (=) мы сравниваем со значением *Материал* перечисления *ВидНоменклатуры* (*Перечисления.ВидыНоменклатуры.Материал*).

Если значения совпадают, операторы цикла выполняются. Если нет, мы переходим к следующей итерации цикла, к следующей строке табличной части.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим работу процедуры

проведения документа *Оказание услуги*.

Откроем список документов, выполнив команду *Оказание услуг* в панели навигации раздела *Оказание услуг*.

Откроем документ *Оказание услуги № 1* и внесем в него следующие изменения:

- удалим из табличной части строку, содержащую *Транзистор Philips*;
- добавим услугу – *Подключение воды*;
- добавим материал – *Шланг резиновый* (рис. 10.4).

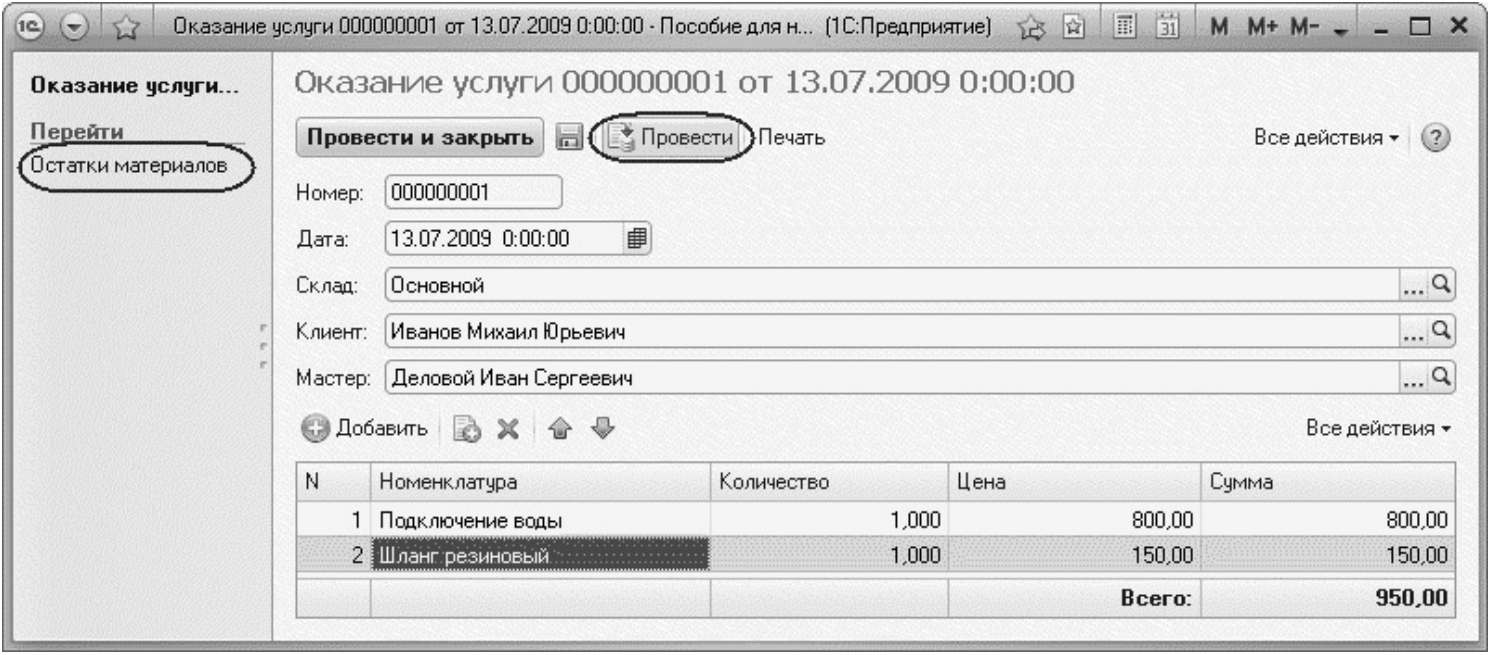


Рис. 10.4. Измененный документ «Оказание услуги № 1»

Обратите внимание, что цены подставляются автоматически из регистра сведений *Цены*.

Нажмем кнопку *Провести* в командной панели формы документа.

Затем выполним команду *Остатки материалов* в панели навигации формы, чтобы перейти к записям регистра *Остатки материалов*, связанным с данным документом (рис. 10.5).

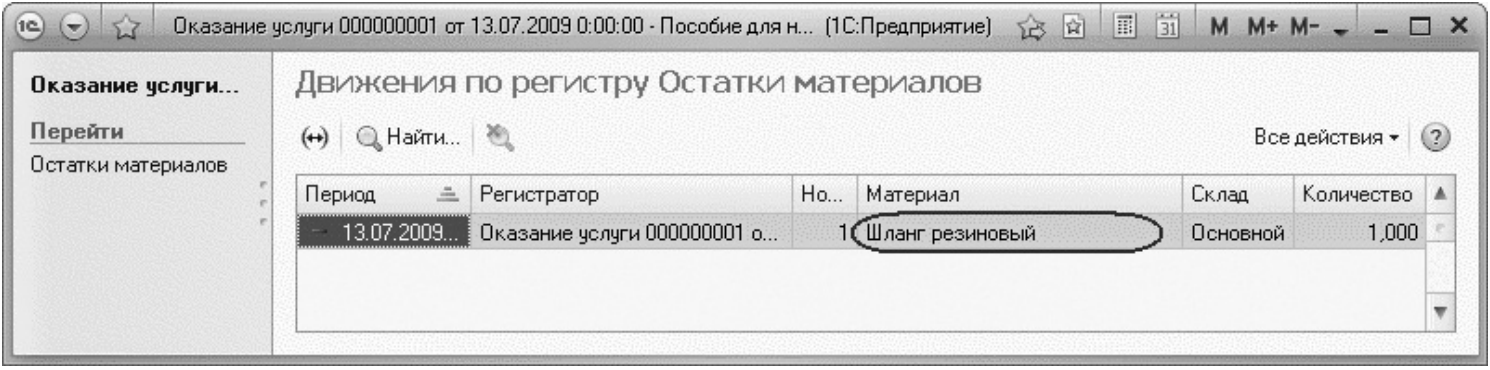


Рис. 10.5. Записи регистра «Остатки материалов»

Как мы видим, в движения по регистру *Остатки материалов* включаются только строки, содержащие материалы. Запись про услугу *Подключение воды* в движения не попала.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Перечисление».*
- *Как создать новое перечисление.*
- *Как с помощью перечисления задать принадлежность элементов справочника к той или иной смысловой группе.*
- *Как обратиться к значению перечисления средствами встроенного языка.*

Занятие 11 (1:20). Проведение документа по нескольким регистрам

Продолжительность

Ориентировочная продолжительность занятия – 1 час 20 минут.

Это занятие будет посвящено тому, как один и тот же документ может поставлять информацию в различные регистры конфигурации и для чего может понадобиться такая возможность.

На этом занятии мы создадим еще один регистр накопления нашей конфигурации и изменим процедуру проведения документов так, чтобы они записывали необходимые данные как в один, так и в другой регистр.

Также мы подготовим базу для изучения следующей главы.

Зачем нужно проведение документа по нескольким регистрам

До сих пор мы с вами учитывали только количественное движение материалов в ООО «На все руки мастер». Для этих целей мы создали регистр накопления

ОстаткиМатериалов.

Однако как вы, наверное, догадываетесь, одного только количественного учета совершенно недостаточно для нужд нашего предприятия. Очевидно, что необходимо также знать, какие денежные средства были затрачены на приобретение тех или иных материалов и каковы материальные запасы ООО «На все руки мастер» в денежном выражении.

После того как мы начали автоматизировать наше предприятие, руководство ООО «На все руки мастер» высказало пожелание, чтобы весь суммовой учет материалов велся бы теперь по средней стоимости.

То есть при закупке материалов они должны учитываться в ценах приобретения, а при расходе – по средней стоимости, которая рассчитывается исходя из общей суммы закупок данного материала и общего количества этого материала, находящегося в ООО «На все руки мастер».

Поскольку подобная информация имеет совершенно другую структуру, нежели количественный учет, для хранения данных об общей стоимости тех или иных материалов мы будем использовать еще один регистр накопления *СтоимостьМатериалов*.

Таким образом, документы *ПриходнаяНакладная* и *ОказаниеУслуги* должны

будут создавать движения не только в регистре *ОстаткиМатериалов*, но одновременно и в регистре *СтоимостьМатериалов*, отражая изменения суммового учета.

Добавление еще одного регистра накопления

В режиме «Конфигуратор»

Регистр *СтоимостьМатериалов* совсем не сложен, поэтому мы не будем подробно останавливаться на его создании.

Создадим новый объект конфигурации *Регистр накопления* с именем *СтоимостьМатериалов*.

Расширенное представление списка зададим как *Движения по регистру Стоимость материалов*. Этот заголовок будет отображаться в окне списка записей регистра.

На закладке *Подсистемы* отметим, что этот регистр будет отображаться в подсистемах *Бухгалтерия, Учет материалов* и *Оказание услуг*.

На закладке *Данные* создадим для регистра одно измерение – *Материал* с типом *СправочникСсылка.Номенклатура* и один ресурс – *Стоимость* с

длиной *15* и точностью *2*.

После создания регистр *СтоимостьМатериалов* должен выглядеть в дереве конфигурации следующим образом (рис. 11.1).

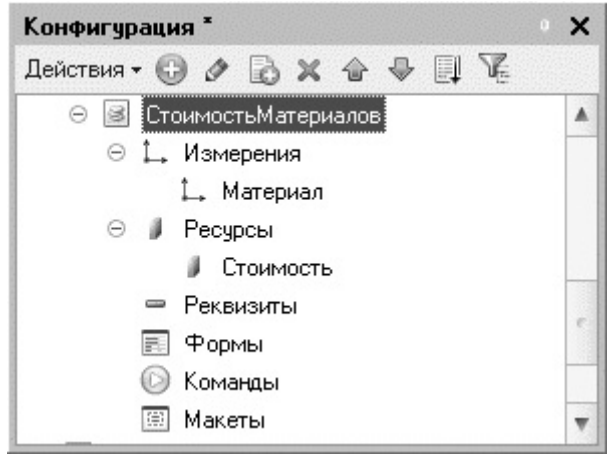


Рис. 11.1. Регистр накопления «Стоимость материалов»

Теперь отредактируем командный интерфейс, чтобы в подсистемах *Бухгалтерия*, *Оказание услуг* и *Учет материалов* была доступна ссылка для просмотра нашего регистра накопления.

В дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*.

В открывшемся окне слева в списке *Подсистемы* выделим подсистему *Бухгалтерия*.

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель навигации.Обычное* включим видимость у команды *Стоимость материалов* и мышью перетащим ее в группу *Панель навигации.См.также* (рис. 11.2).

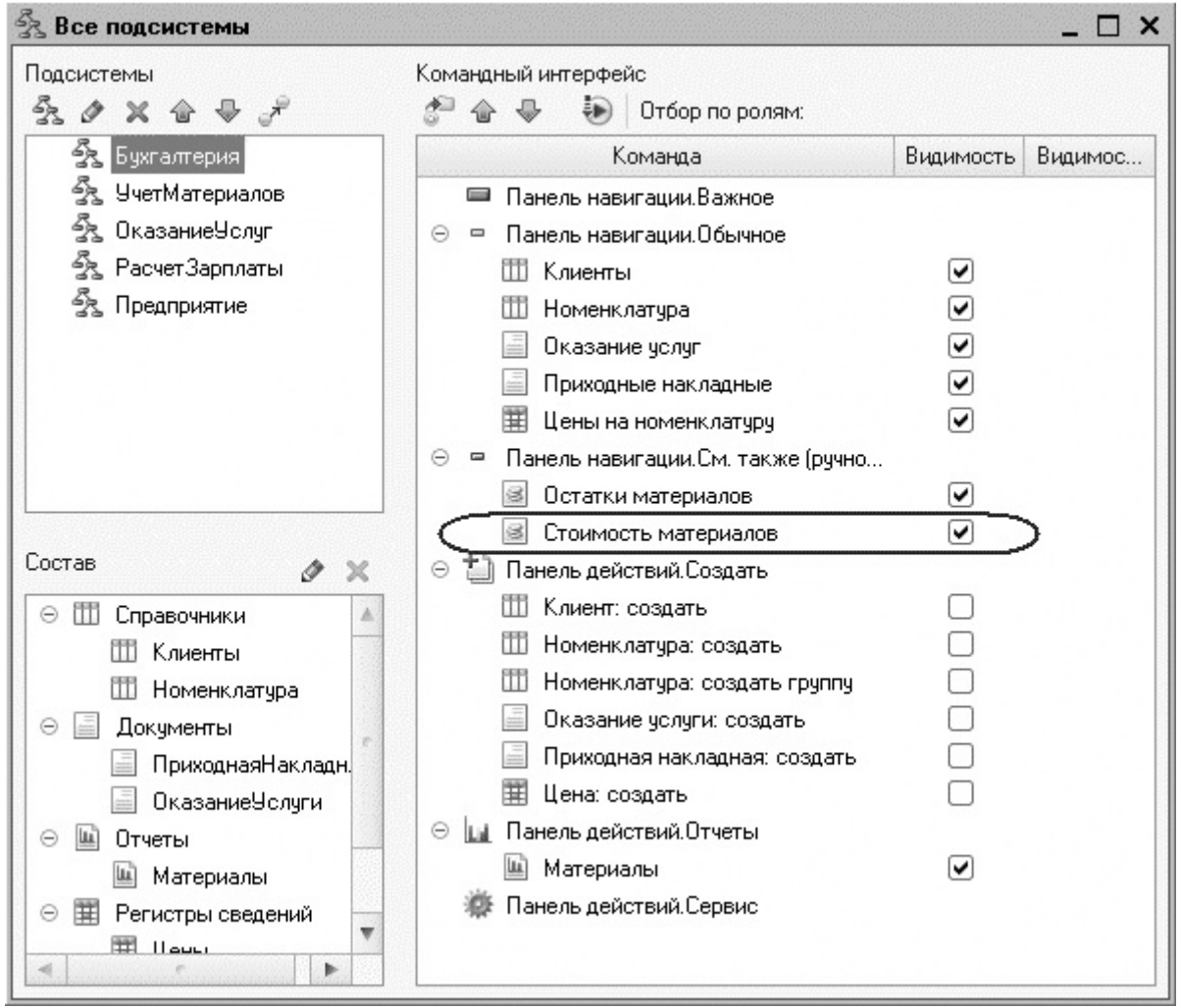


Рис. 11.2. Настройка командного интерфейса подсистем

Аналогично, выделив подсистемы *ОказаниеУслуг* и *УчетМатериалов*, в панели навигации в группе *Обычное* включим видимость у команды *Стоимость материалов* и перенесем ее в группу *См. также*.

Теперь мы можем приступить к внесению изменений в процедуры проведения наших документов.

Начнем с самого простого – документа *Приходная накладная*.

Проведение приходной накладной по двум регистрам

В режиме «Конфигуратор»

Изменение процедуры проведения

Откроем в конфигураторе окно редактирования объекта конфигурации Документ *ПриходнаяНакладная* и перейдем на закладку *Движения*.

В списке регистров отметим, что документ будет создавать теперь движения и по регистру *СтоимостьМатериалов* (рис. 11.3).

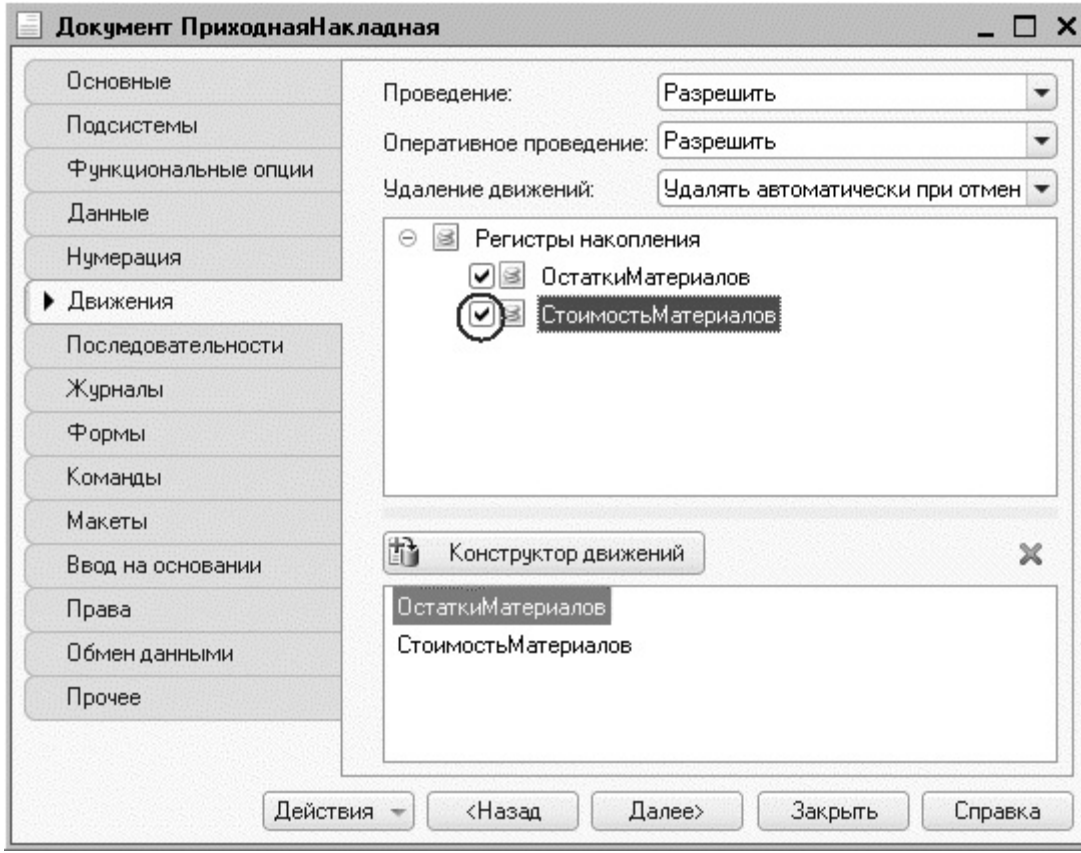


Рис. 11.3. Создание движений документа «ПриходнаяНакладная» в регистре «Стоимость материалов»

На этот раз мы не будем использовать конструктор движений документа, а внесем необходимые дополнения прямо в обработчик события *ОбработкаПроведения* документа *ПриходнаяНакладная*.

Дело в том, что с помощью конструктора можно создавать движения одновременно и в нескольких регистрах (в конструкторе движений можно добавлять регистры). Но когда процедура проведения документа уже написана, использование конструктора приведет к тому, что имеющаяся процедура сохранена не будет. То есть сейчас при использовании конструктора пришлось бы заново описывать движения как для одного, так и для другого регистра. Поэтому проще внести изменения в существующую процедуру вручную.

Перейдем на закладку *Прочее* и откроем модуль объекта. Для этого нажмем кнопку *Модуль объекта* (рис. 11.4).

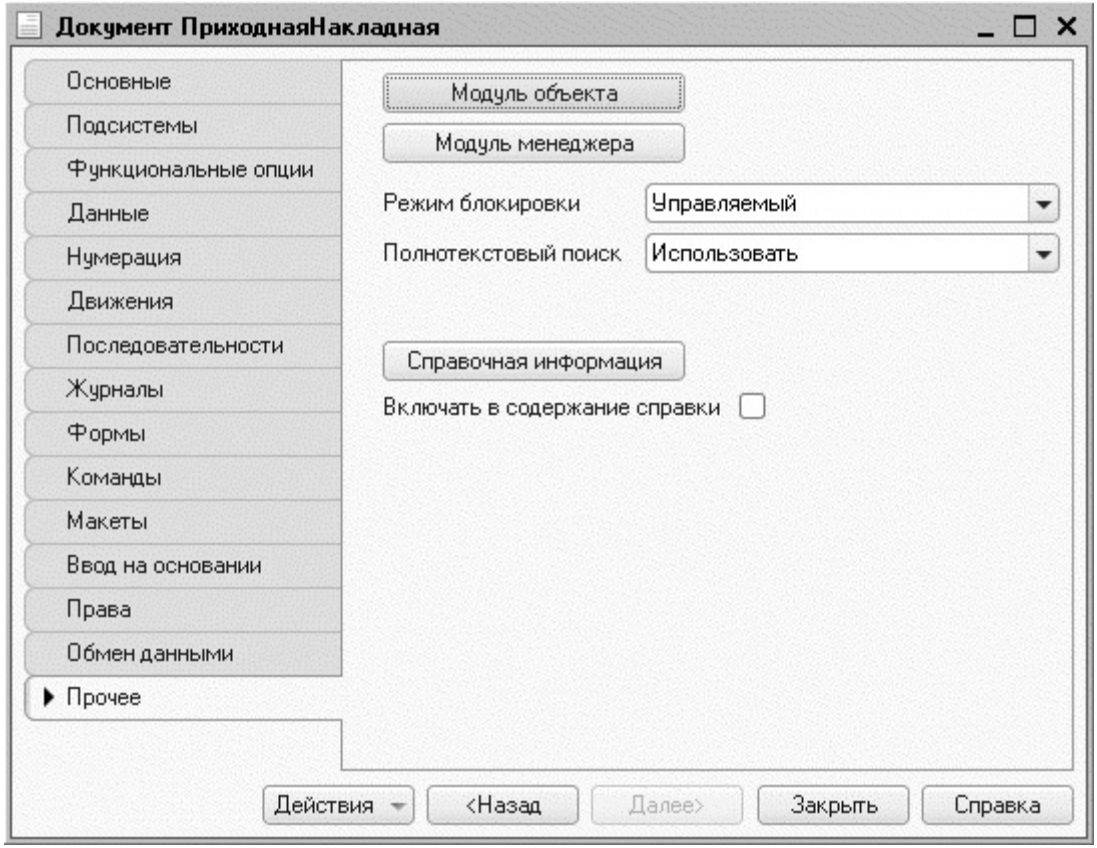


Рис. 11.4. Открытие модуля объекта

Откроем процедуру обработчика события *ОбработкаПроведения*.

В самом конце цикла перед строкой *КонецЦикла* добавим строки кода,

создающие движения в регистре *СтоимостьМатериалов* (листинг 11.1).

Листинг 11.1. Движения документа «ПриходнаяНакладная» (фрагмент)

```
// регистр Стоимость Материалов Приход
Движение = Движения.СтоимостьМатериалов.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
Движение.Период = Дата;
Движение.Материал = ТекСтрокаМатериалы.Материал;
Движение.Стоимость = ТекСтрокаМатериалы.Сумма;
```

Перед началом цикла установим для набора записей движений по этому регистру свойство *Записывать* в значение *Истина*, чтобы платформа автоматически записала созданные нами движения после выхода из процедуры проведения документа. И еще удалим комментарии, внесенные конструктором.

В результате процедура *ОбработкаПроведения* будет выглядеть следующим образом (листинг 11.2).

Листинг 11.2. Движения документа «ПриходнаяНакладная»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  Движения.ОстаткиМатериалов.Записывать = Истина;
  Движения.СтоимостьМатериалов.Записывать = Истина;
```

```
  Для Каждого ТекСтрокаМатериалы Из Материалы Цикл

    // Регистр ОстаткиМатериалов Приход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаМатериалы.Количество;

    // Регистр Стоимость Материалов Приход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.Стоимость = ТекСтрокаМатериалы.Сумма;

  КонецЦикла;

КонецПроцедуры
```

Команда перехода к записям регистра

В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра *СтоимостьМатериалов*, связанному с документом.

Для этого откроем форму документа *ПриходнаяНакладная*. В левом верхнем

окне перейдем на закладку *Командный интерфейс*. В разделе *Панель навигации* раскроем группу *Перейти* и увидим команду для открытия регистра накопления *Стоимость материалов*. Установим свойство *Видимость* для этой команды (рис. 11.5).

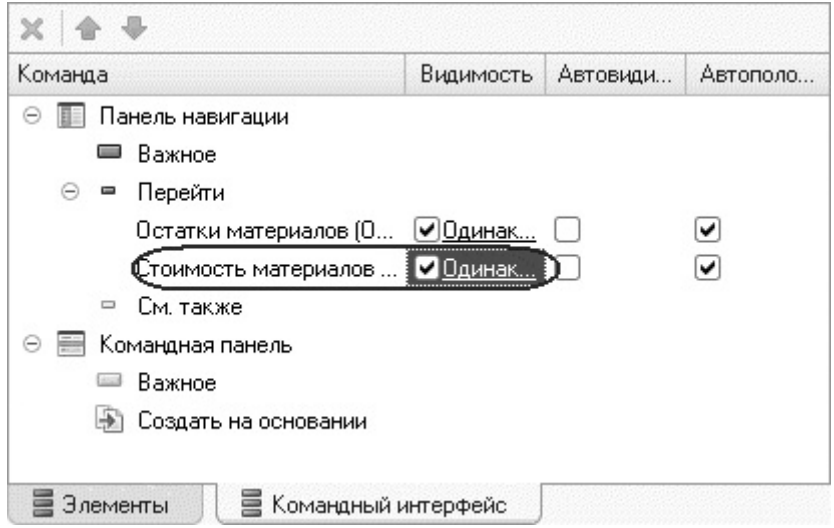


Рис. 11.5. Настройка командного интерфейса формы документа «ПриходнаяНакладная»

В режиме «1С:Предприятие»

В режиме *1С:Предприятие* наша задача будет заключаться в том, чтобы провести еще раз (перепровести) все приходные накладные. Это необходимо для того, чтобы эти документы создали новые записи в регистрах, в соответствии с алгоритмом проведения, который мы только что изменили.

Запустим «1С:Предприятие» в режиме отладки. Откроем список документов, выполнив команду *Приходные накладные* в панели навигации раздела *Учет материалов*.

Выделим одновременно, используя клавишу *Ctrl,* все приходные накладные и перепроведем их, выполнив команду *Все действия* > *Провести*.

Затем откроем первый документ (рис. 11.6) и, выполнив команды перехода к регистрам *Остатки материалов* и *Стоимость материалов*, убедимся, что документ создает желаемые записи как в одном (рис. 11.7), так и в другом регистре накопления (рис. 11.8).

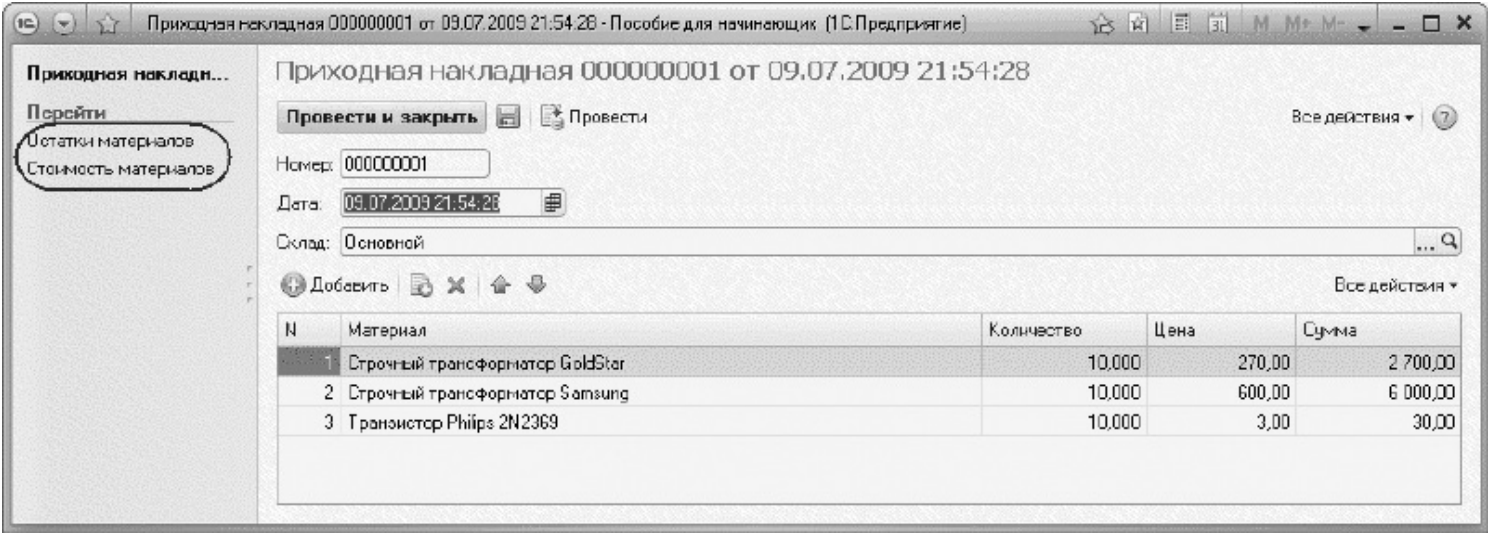


Рис. 11.6. Приходная накладная № 1

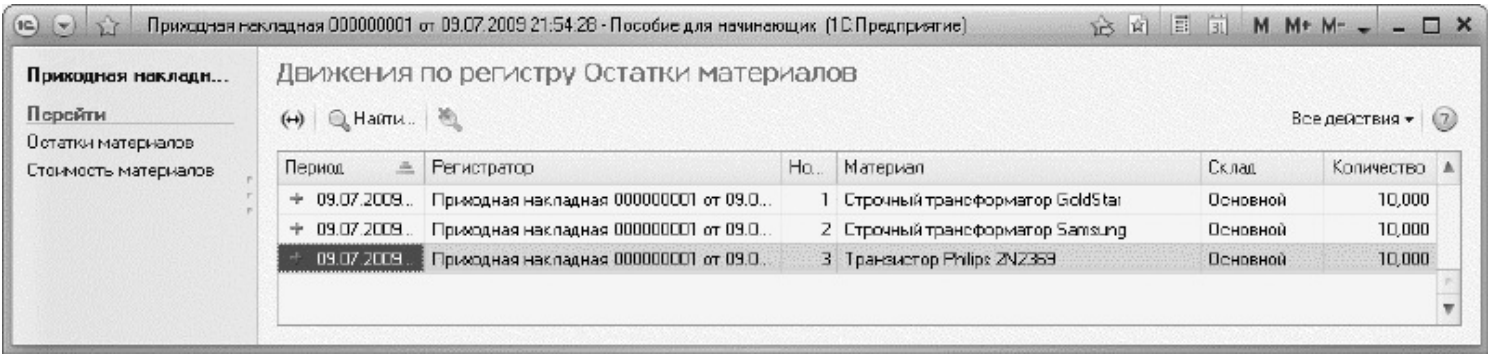


Рис. 11.7. Записи регистра «Остатки материалов»

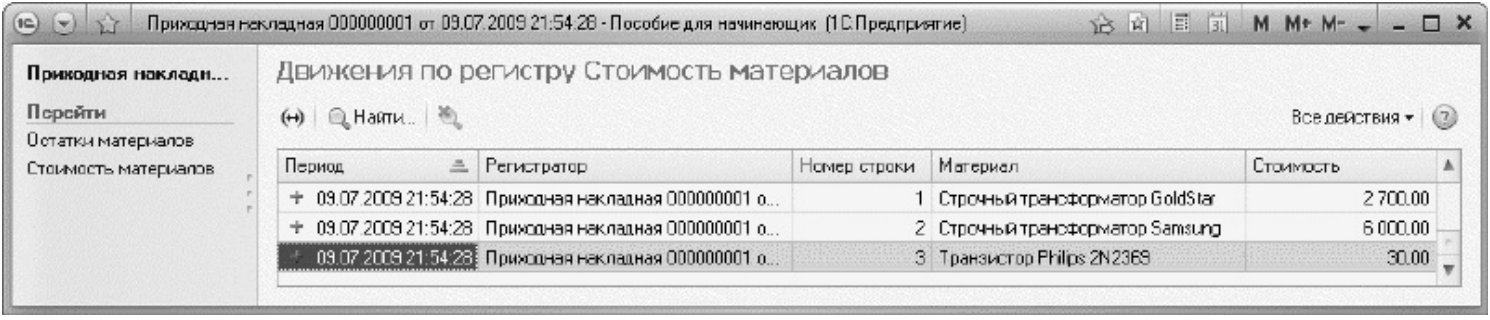


Рис. 11.8. Записи регистра «Стоимость материалов»

Проведение документа «Оказание услуги» по двум регистрам

В заключение мы внесем изменения в процедуру обработки проведения

документа *ОказаниеУслуги*.

При этом мы будем исходить из пожелания, высказанного руководством ООО «На все руки мастер». Суть его заключается в том, что на первом этапе, при списании материалов, израсходованных в процессе оказания услуги, должна быть возможность указывать различную стоимость для одного и того же материала, которая рассчитана руководством исходя из текущих конъюнктурных соображений.

Поскольку в документе *ОказаниеУслуги* у нас отражена только цена номенклатуры, нам понадобится:

- Добавить в табличную часть документа еще один реквизит, в котором будет указываться стоимость номенклатуры.
- После этого изменить процедуру проведения документа *ОказаниеУслуги.*
- И в заключение в режиме *1С:Предприятие* перепровести все эти документы, чтобы отработал новый измененный нами алгоритм проведения документов *Оказание услуги*.

В режиме «Конфигуратор»

Новый реквизит документа

Откроем в конфигураторе окно редактирования объекта конфигурации

Документ ОказаниеУслуги и перейдем на закладку *Данные*.

Создадим новый реквизит табличной части документа с именем *Стоимость*, типом *Число*, длиной *15* и точностью *2*, неотрицательное (рис. 11.9).

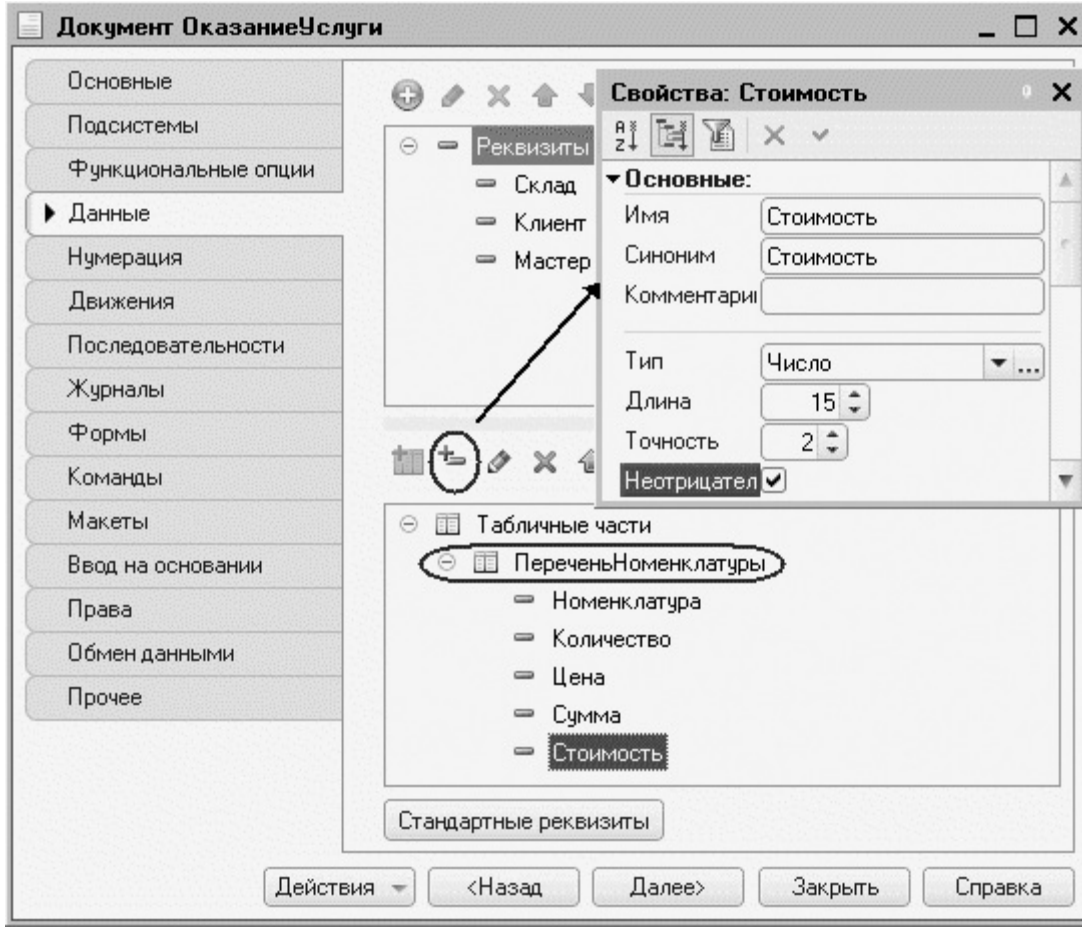


Рис. 11.9. Изменение документа «ОказаниеУслуги»

После этого откроем форму *ФормаДокумента* документа *ОказаниеУслуги* и добавим в табличную часть *ПеречеьНоменклатуры* поле, отображающее

новый реквизит *Стоимость*.

Для этого в правом верхнем окне редактора форм на закладке *Реквизиты* раскроем реквизит формы *Объект* (рис. 11.10).

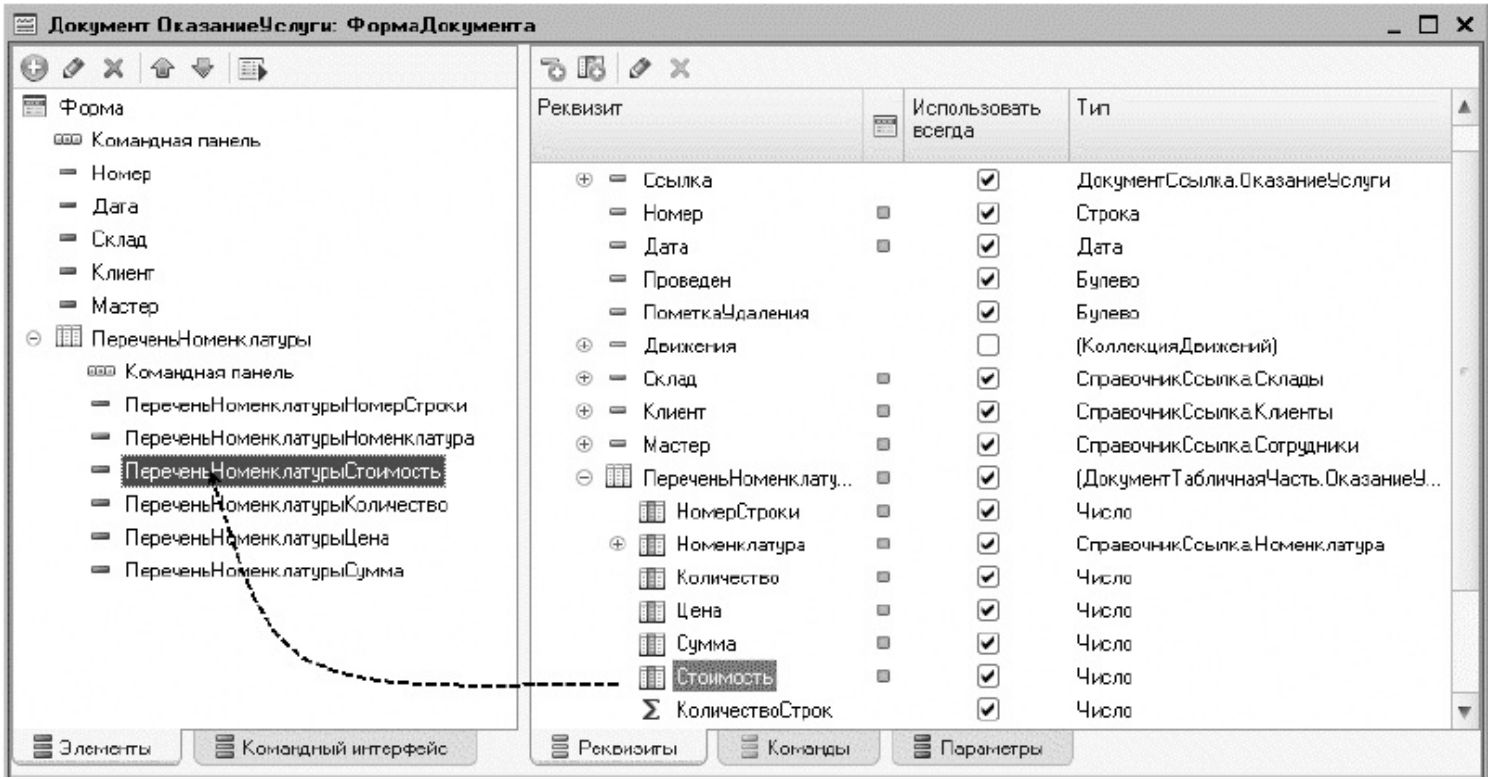


Рис. 11.10. Изменение формы документа «ОказаниеУслуги»

Мы видим, что он содержит все реквизиты документа *ОказаниеУслуги*.

Найдем в табличной части реквизит *Стоимость* и с помощью мыши перетащим его в окно элементов формы, расположенное слева в верхней части редактора форм.

Новый элемент расположим в структуре элементов формы после поля *Номенклатура*. Оставим свойства элемента формы, предложенные по умолчанию.

Новый реквизит сразу же отобразится в форме документа, расположенной в левом нижнем окне редактора форм (рис. 11.11).

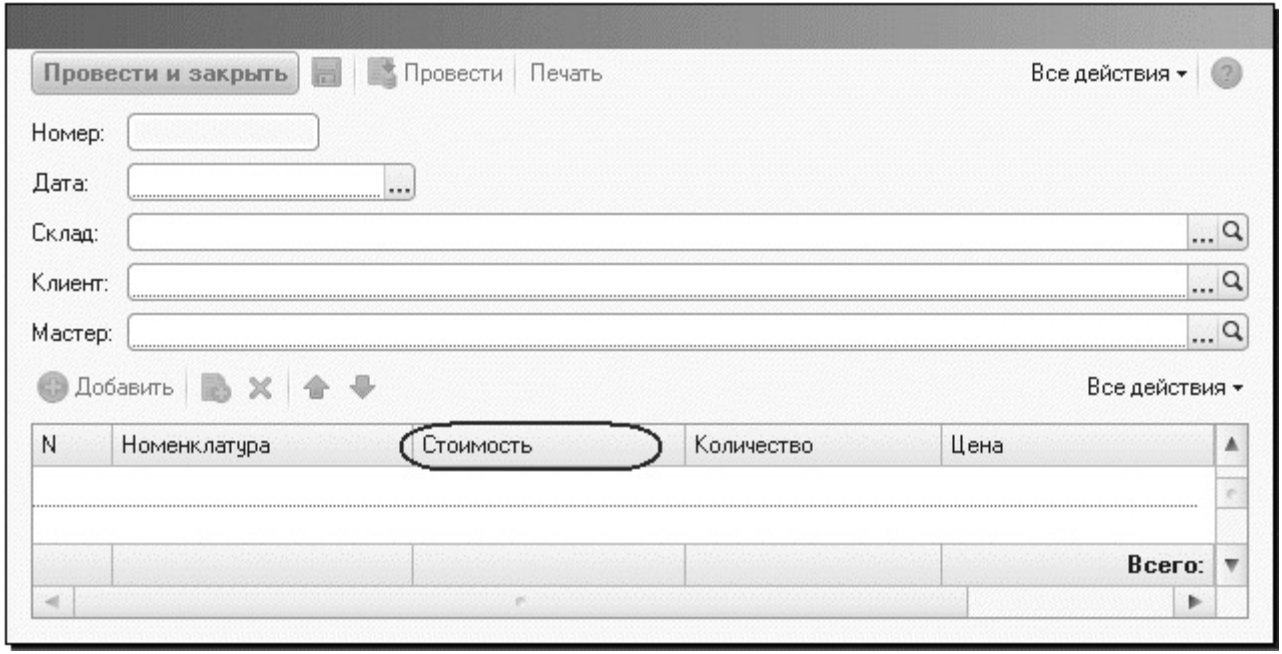


Рис. 11.11. Изменение формы документа «ОказаниеУслуги»

Изменение процедуры проведения

Теперь создадим движения документа *ОказаниеУслуги* таким же образом, как мы делали это для документа *ПриходнаяНакладная*.

В окне редактирования объекта конфигурации *Документ ОказаниеУслуги* перейдем на закладку *Движения*. В списке регистров отметим, что документ будет создавать теперь движения и по регистру *СтоимостьМатериалов*. Перейдем на закладку *Прочее* и откроем модуль объекта. Для этого нажмем

кнопку *Модуль объекта*. Откроем процедуру обработчика события *ОбработкаПроведения*.

В самом конце цикла перед строкой *КонецЕсли* добавим строки кода, создающие движения регистра *СтоимостьМатериалов*, производимые документом *ОказаниеУслуги* (листинг 11.3).

Листинг 11.3. Движения документа «ОказаниеУслуги» (фрагмент)

```
// регистр СтоимостьМатериалов Расход
Движение = Движения.СтоимостьМатериалов.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
Движение.Период = Дата;
Движение.Материал = ТекСтрокаПеречеНоменклатуры.Номенклатура;
Движение.Стоимость = ТекСтрокаПеречеНоменклатуры.Количество *
ТекСтрокаПеречеНоменклатуры.Стоимость;
```

Перед началом цикла установим свойство *Записывать* набора записей движений по этому регистру в значение *Истина*. Удалим комментарии, внесенные конструктором.

В результате процедура *ОбработкаПроведения* будет выглядеть следующим образом (листинг 11.4).

Листинг 11.4. Движения документа «ОказаниеУслуги»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  Движения.ОстаткиМатериалов.Записывать = Истина;
  Движения.СтоимостьМатериалов.Записывать = Истина;

  Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

    Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
    Перечисления.ВидыНоменклатуры.Материал Тогда

      // Регистр ОстаткиМатериалов Расход
      Движение = Движения.ОстаткиМатериалов.Добавить();
      Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
      Движение.Период = Дата;
      Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
      Движение.Склад = Склад;
      Движение.Количество = ТекСтрокаПереченьНоменклатуры.Количество;

      // Регистр СтоимостьМатериалов Расход
      Движение = Движения.СтоимостьМатериалов.Добавить();
      Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
      Движение.Период = Дата;
      Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
      Движение.Стоимость = ТекСтрокаПереченьНоменклатуры.Количество *
      ТекСтрокаПереченьНоменклатуры.Стоимость;
    КонецЕсли;

  КонецЦикла;

КонецПроцедуры
```

Обратите внимание, что измерение регистра *Стоимость* вычисляется как произведение стоимости и количества, указанных в табличной части документа.

В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра *Стоимость Материалов*, связанному с документом.

Для этого откроем форму документа *ОказаниеУслуги*. В левом верхнем окне перейдем на закладку *Командный интерфейс*. В разделе *Панель навигации* раскроем группу *Перейти* и увидим команду для открытия регистра накопления *Стоимость материалов*. Установим свойство *Видимость* для этой команды.

В режиме «1С:Предприятие»

В режиме *1С:Предприятие* наша задача будет заключаться в том, чтобы провести еще раз (перепровести) наш документ оказания услуги. Это необходимо для того, чтобы этот документ создал новые записи в регистрах в соответствии с алгоритмом проведения, который мы только что изменили.

Запустим «1С:Предприятие» в режиме отладки и откроем список документов, выполнив команду *Оказание услуг* в панели навигации раздела *Оказание услуг*.

Откроем документ *Оказание услуги № 1* и укажем в нем стоимость резинового шланга – 100 (рис. 11.12).

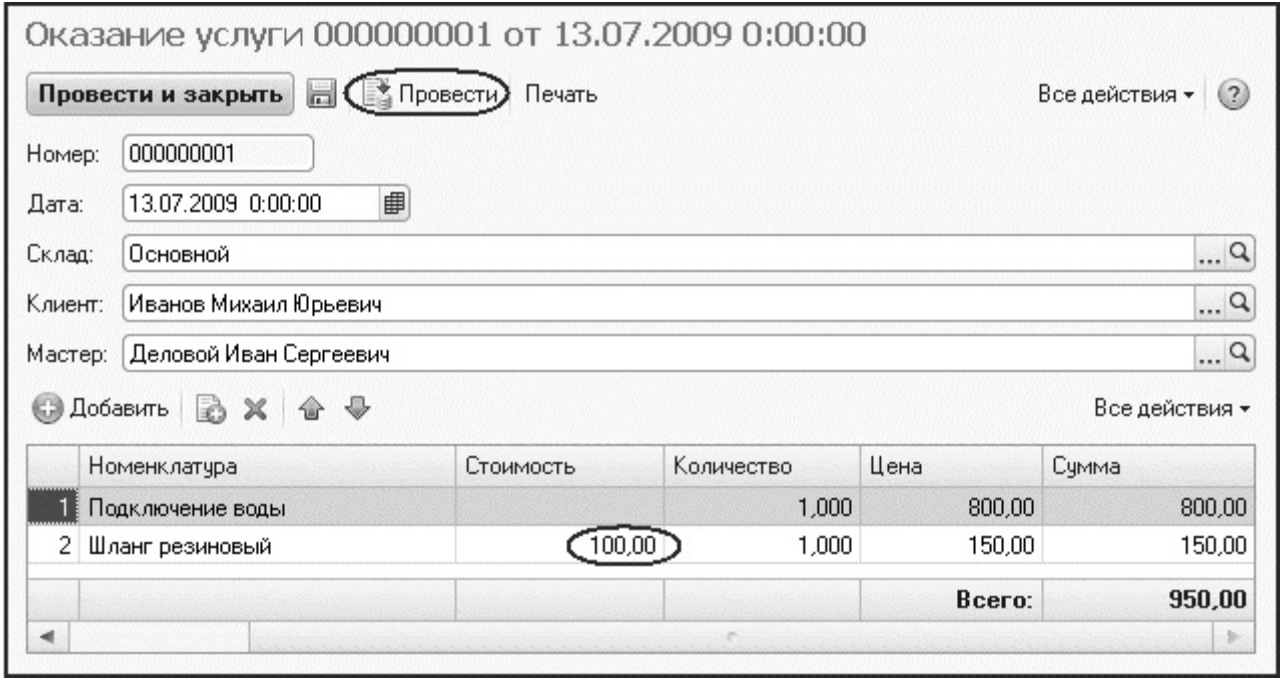


Рис. 11.12. Документ «Оказание услуги № 1»

Проведем документ *Оказание услуги № 1* и посмотрим на движения этого документа по регистру *Стоимость материалов*.

Для этого нажмем кнопку *Провести* и выполним команду перехода к регистру

Стоимость материалов (рис. 11.13).

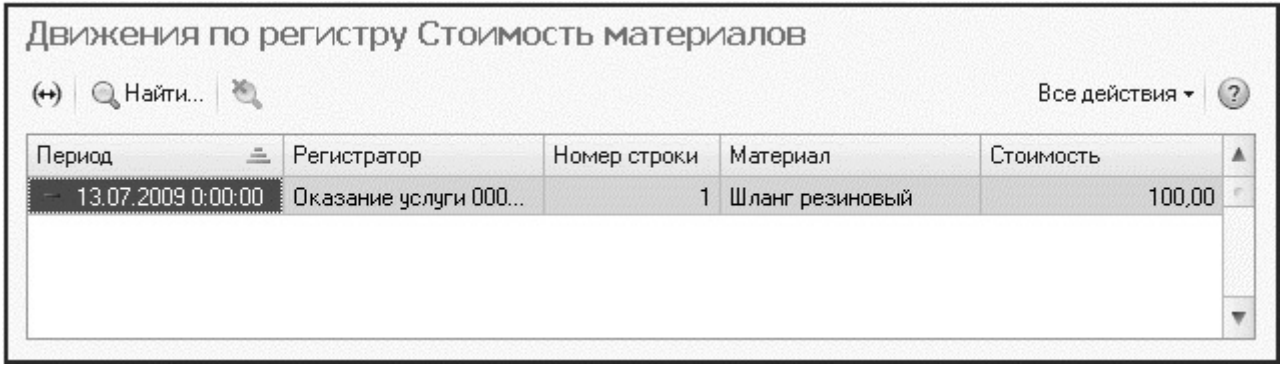


Рис. 11.13. Записи регистра «Стоимость материалов»

Теперь создадим и проведем еще два документа *Оказание услуги.*

Для этого в форме списка документов нажмем кнопку *Создать* или в панели действий раздела *Оказание услуг* выполним команду *Оказание Услуги* (рис. 11.14).

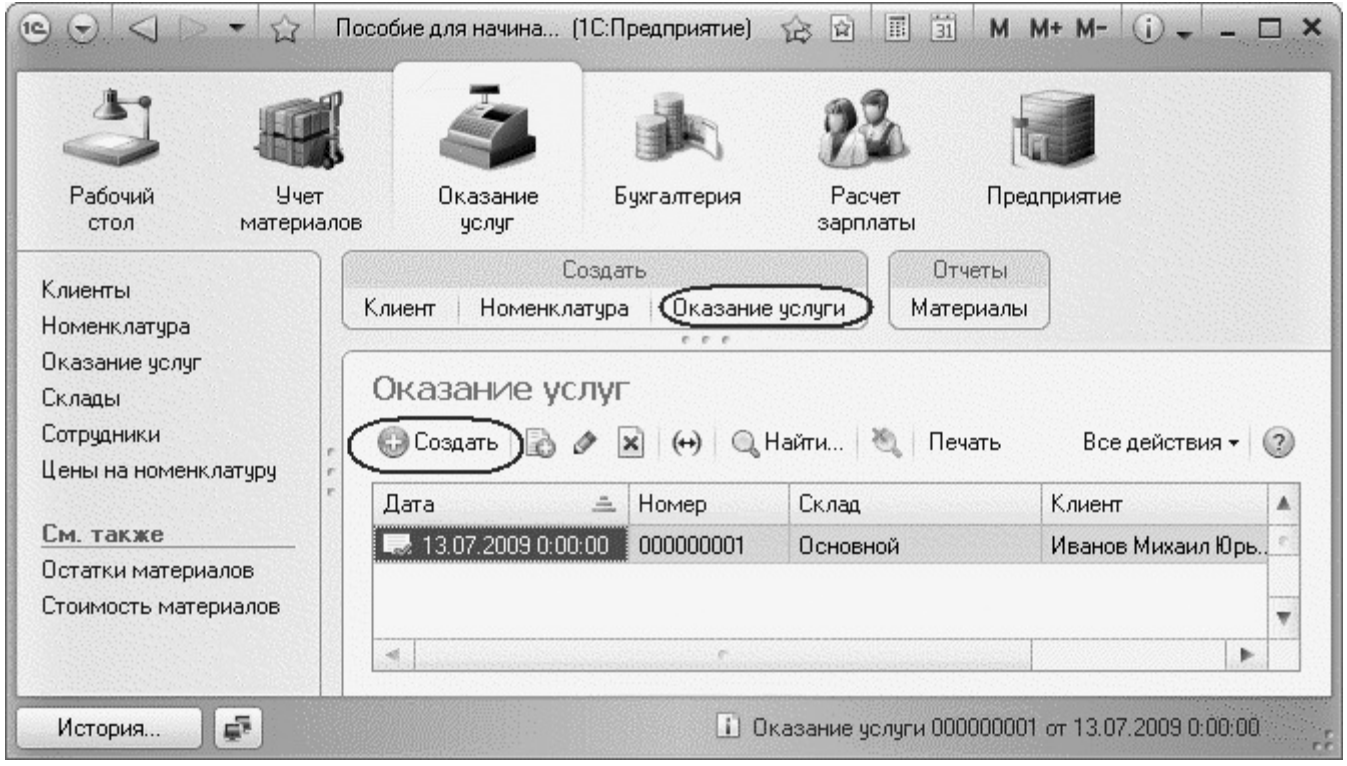


Рис. 11.14. Создание новых документов

Эти документы понадобятся нам в дальнейшем, поэтому будьте внимательны и обратите внимание на то, что эти документы созданы другими датами (рис. 11.15, 11.16).

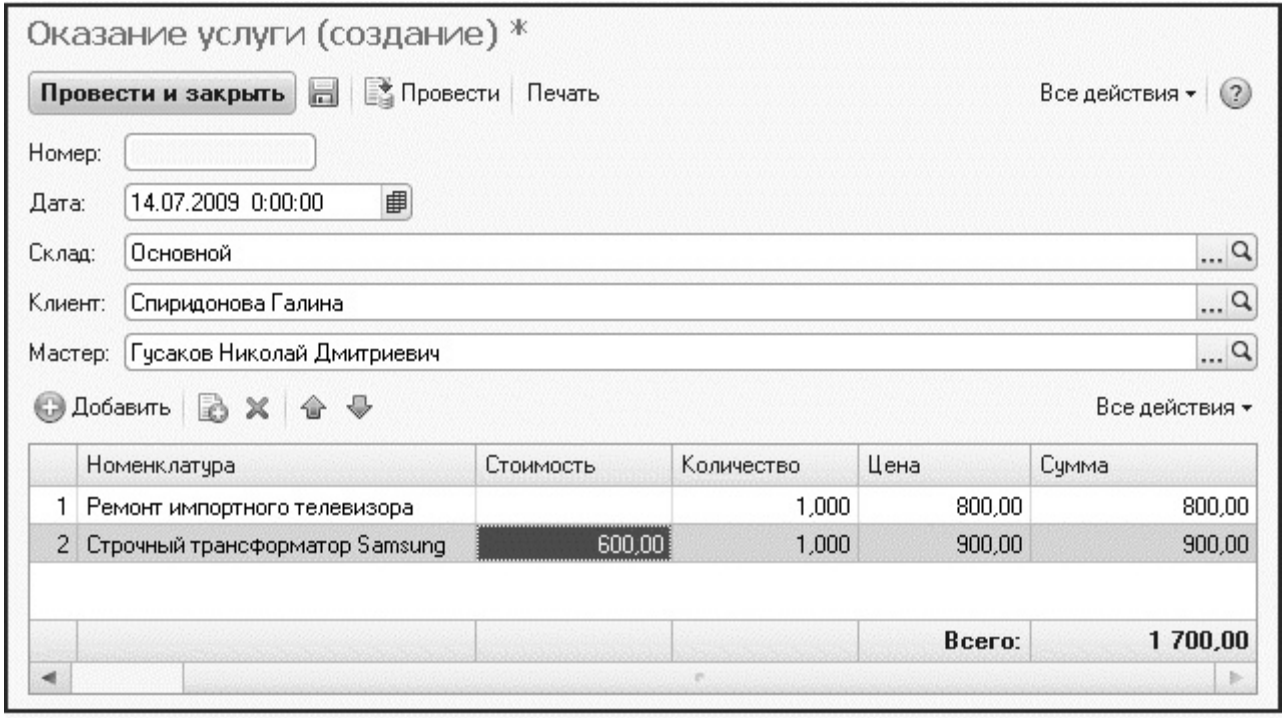


Рис. 11.15. Документ «Оказание услуги № 2»

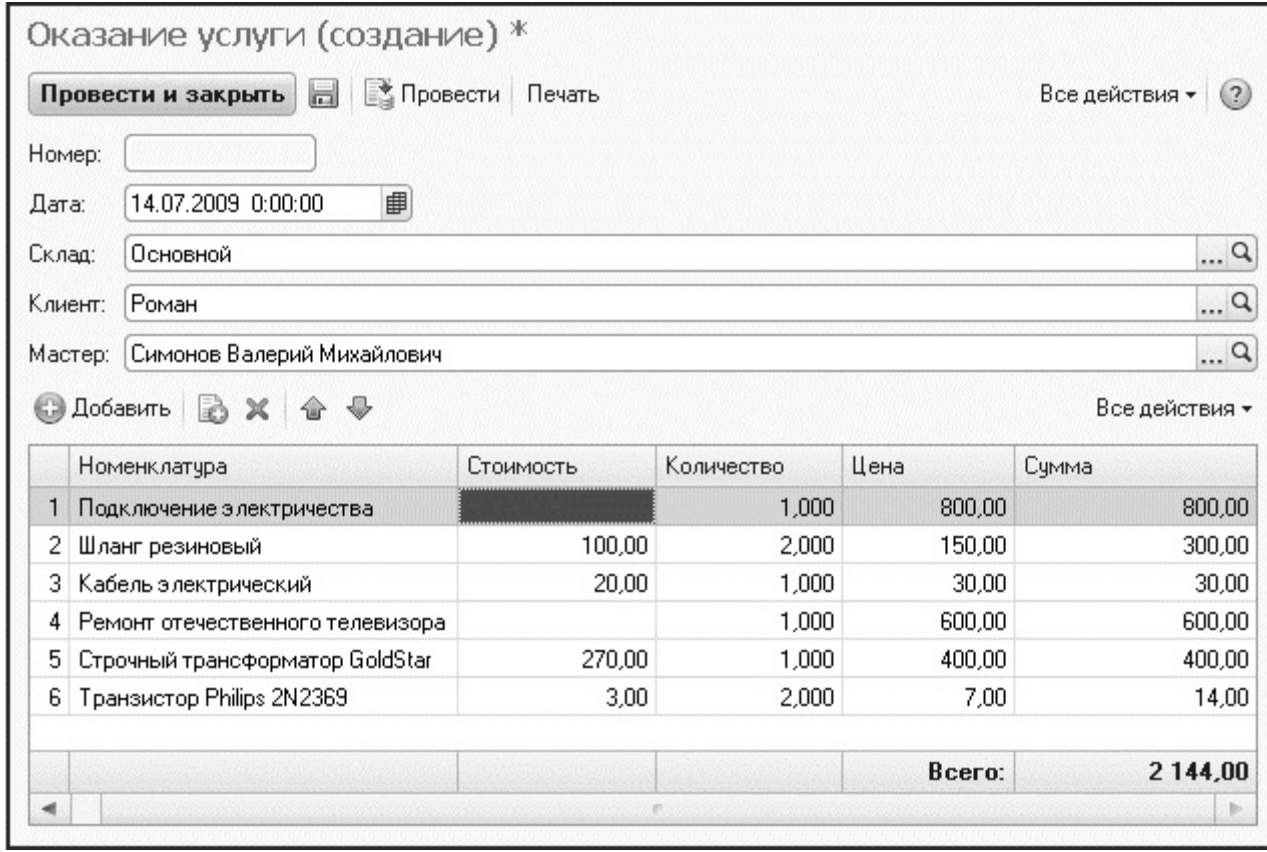


Рис. 11.16. Документ «Оказание услуги № 3»

Движения документов *Оказание услуги № 2* и *№ 3* по регистру *Стоимость материалов* должны выглядеть соответственно следующим образом (рис. 11.17, 11.18).

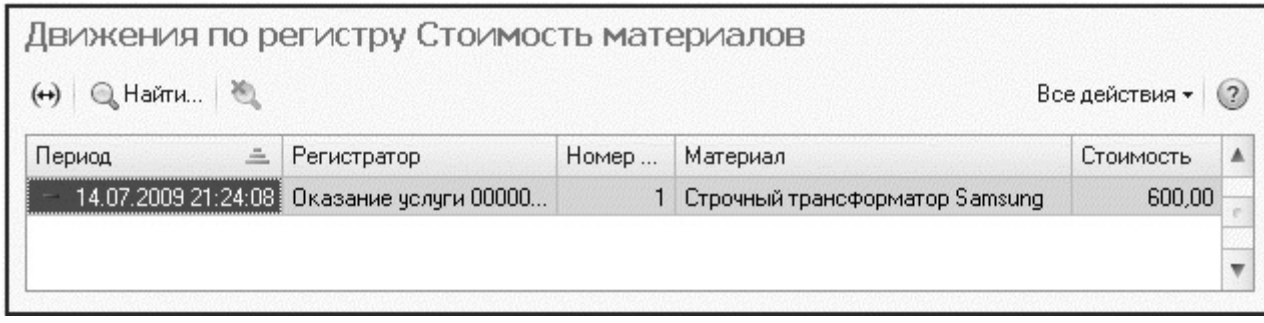


Рис. 11.17. Движения документа «Оказание услуги № 2»

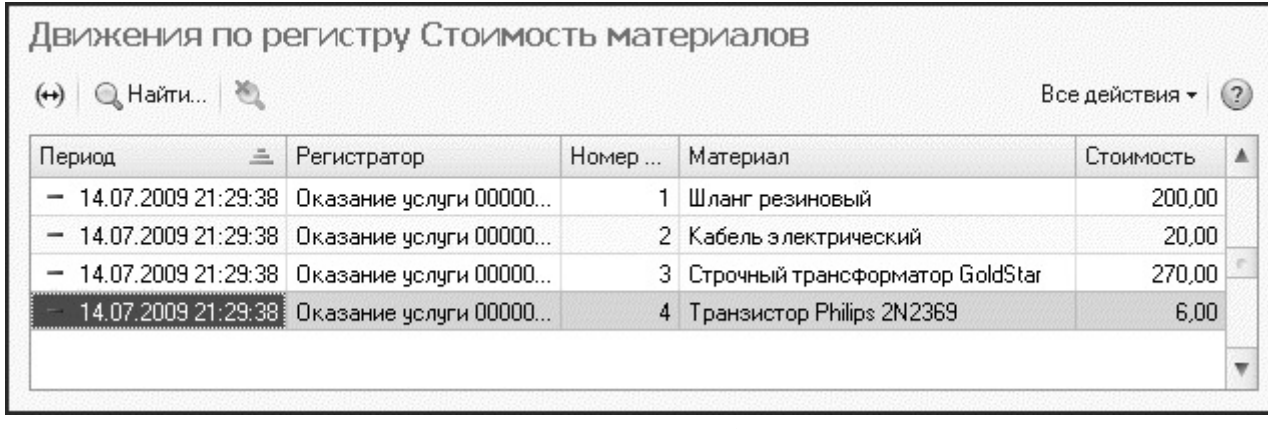


Рис. 11.18. Движения документа «Оказание услуги № 3»

Контрольные вопросы

- *Для чего может понадобиться проведение документа по нескольким регистрам.*

- *Как создать движения документа по нескольким регистрам в обработчике проведения документа.*
- *Как создать движения документа без использования конструктора движений.*
- *Как средствами встроенного языка сформировать и записать движения документа в регистр накопления.*
- *Как добавить в форму документа новый реквизит.*

Занятие 12 (0:40). Оборотные регистры накопления

Продолжительность

Ориентировочная продолжительность занятия – 40 минут.

На этом занятии мы с вами познакомимся с еще одним видом регистра накопления – оборотным регистром накопления.

Вы узнаете о некоторых важных принципах выбора измерений и реквизитов регистров накопления.

Мы с вами создадим оборотный регистр накопления и добавим в один из наших документов движения еще и по этому регистру.

Зачем нужно создавать еще один регистр

Продолжим рассматривать работу нашего документа *ОказаниеУслуги*.

До сих пор мы создавали в регистрах накопления движения только для строк документа, которые содержат материалы. Услуги, содержащиеся в документе, мы никак не учитывали.

Дело в том, что при учете услуг важны совершенно другие критерии, нежели при учете материалов.

Прежде всего, бессмысленно говорить о том, сколько услуг было и сколько их осталось, важна только сумма и количество услуг, которые были оказаны за определенный промежуток времени.

Кроме этого, интересны следующие моменты:

- какие именно услуги были оказаны (чтобы составить рейтинг услуг);
- какому именно клиенту оказывались услуги (чтобы, например, предоставить ему скидку от объема оплаченных ранее услуг);
- какой мастер предоставлял услуги (чтобы начислить ему заработную плату).

Очевидно, что существующие регистры накопления совершенно не подходят для решения таких задач.

Поэтому мы создадим еще одно хранилище данных, которое будет использоваться в нашей программе, – оборотный регистр накопления *Продажи*.

Что такое оборотный регистр накопления

Когда мы создавали регистры *ОстаткиМатериалов* и *СтоимостьМатериалов*, мы специально не останавливались на видах регистров накопления, которые существуют в системе «1С:Предприятие». Сейчас пришло время сказать об этом несколько слов.

Регистры накопления могут быть *регистрами остатков* и *регистрами оборотов*.

Существующие в нашей учебной конфигурации регистры *ОстаткиМатериалов* и *СтоимостьМатериалов* являются регистрами остатков.

Если вы помните, при создании отчета *Материалы* в конструкторе запроса мы видели, что для таких регистров система создает три виртуальные таблицы: таблица остатков, оборотов и совокупная таблица остатков и оборотов.

Оборотный регистр накопления очень похож на знакомый уже нам регистр остатков, но для него понятие «остаток» не имеет смысла. Оборотный регистр накапливает только обороты, остатки ему безразличны. Поэтому единственной виртуальной таблицей, которую будет создавать система для такого регистра, будет таблица оборотов. В остальном оборотный регистр ничем не отличается от регистра остатков.

Следует сказать об одной особенности конструирования регистров накопления,

напрямую связанной с возможностью получения остатков. При создании оборотного регистра накопления нет особой сложности в определении того, какие именно данные должны являться измерениями регистра – мы можем назначить в качестве его измерений любые нужные нам данные.

Совсем иная ситуация в случае регистра накопления, поддерживающего накопление остатков. Для него выбор измерений должен выполняться исходя из того, что движения регистра могут быть осуществлены в две стороны: приход и расход. Таким образом, в качестве измерений нужно выбирать те данные, по которым движения точно будут осуществляться как в одну, так и в другую сторону.

Например, если ведется учет материалов в разрезах номенклатуры и склада, очевидно, что и номенклатура, и склад могут быть измерениями, поскольку как приход, так и расход материалов всегда будут осуществляться с указанием конкретной номенклатуры и конкретного склада. Если же в этой ситуации появляется желание отразить учет материалов еще и в разрезе поставщика, то здесь уже нужно исходить из конкретной схемы учета, принятой на предприятии.

Скорее всего, при поступлении материалов поставщик будет указан, а вот при расходовании материалов, с большой долей вероятности, поставщик указываться не будет. В большинстве случаев это совершенно лишняя информация.

Значит, поставщика следует добавить не как измерение, а как реквизит регистра накопления.

Если же при расходе материалов поставщик будет указываться наверняка, имеет смысл добавить поставщика в измерения регистра.

Иными словами, по каждому из измерений регистра накопления остатков ресурсы обязательно должны изменяться в обе стороны: приход и расход. Не должно существовать таких измерений, по которым осуществляется только приход или только расход.

Нарушение этого принципа построения регистров накопления будет вести к непроизводительному использованию ресурсов системы и как следствие к замедлению работы и падению производительности.

Для реквизитов же регистра этот принцип не важен. По реквизитам регистра ресурсы могут только приходоваться или только расходоваться.

Добавление оборотного регистра накопления

В режиме «Конфигуратор»

Теперь, когда мы знаем практически все о регистрах накопления, откроем

конфигуратор и создадим новый объект конфигурации *Регистр накопления*.

Назовем его *Продажи* и определим вид регистра – *Обороты*.

Кроме этого, зададим *Расширенное представление списка* как *Движения по регистру Продажи*. Этот заголовок будет отображаться в окне списка записей регистра (рис. 12.1).

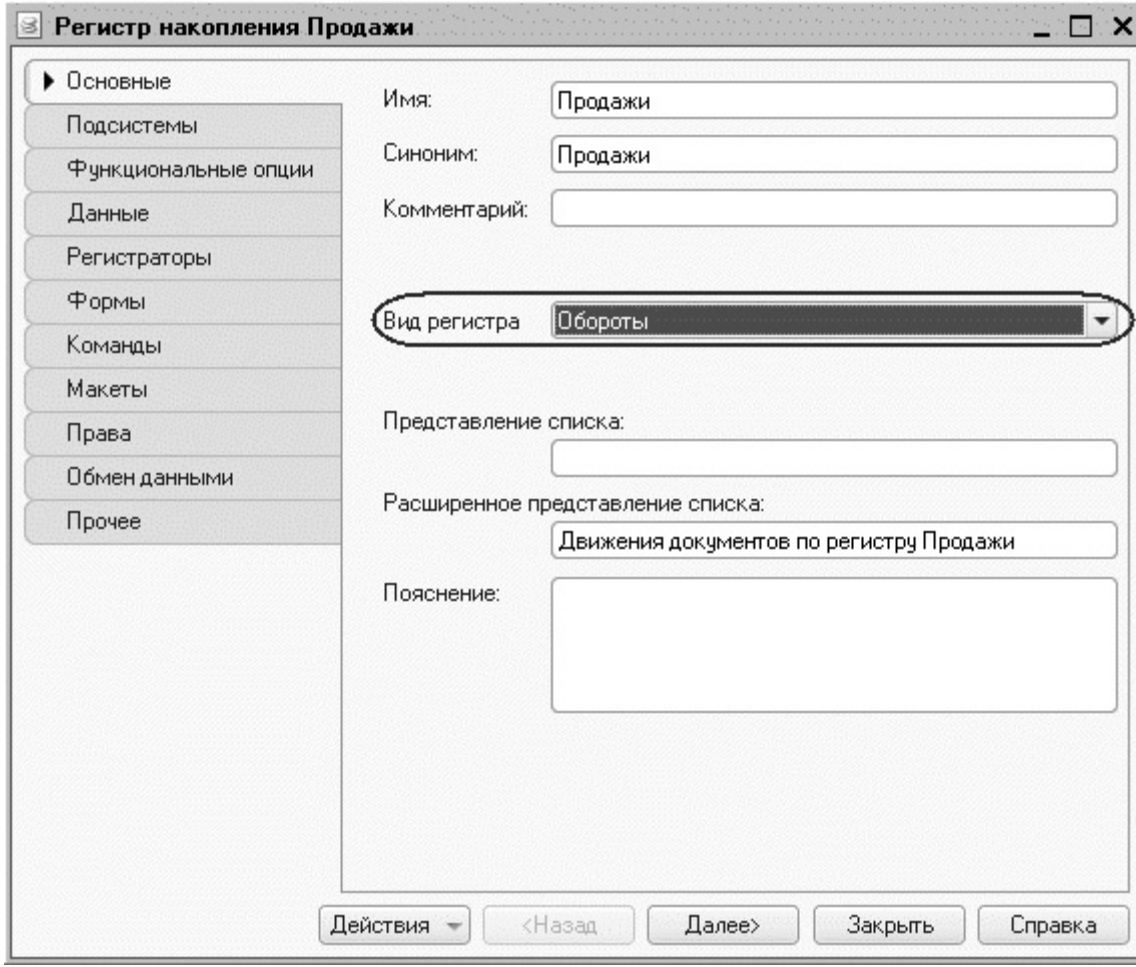


Рис. 12.1. Создание оборотного регистра накопления

На закладке *Подсистемы* отметим, что этот регистр будет отображаться в

подсистемах *Бухгалтерия, Учет материалов* и *Оказание услуг*.

На закладке *Данные* создадим измерения регистра:

- *Номенклатура*, тип *СправочникСсылка.Номенклатура*;
- *Клиент*, тип *СправочникСсылка.Клиенты*;
- *Мастер*, тип *СправочникСсылка.Сотрудники*.

У регистра будет три ресурса:

- *Количество*, тип *Число*, длина *15*, точность *3*;
- *Выручка*, тип *Число*, длина *15*, точность *2*;
- *Стоимость*, тип *Число*, длина *15*, точность *2*.

После создания регистр *Продажи* должен выглядеть в дереве конфигурации следующим образом (рис. 12.2).

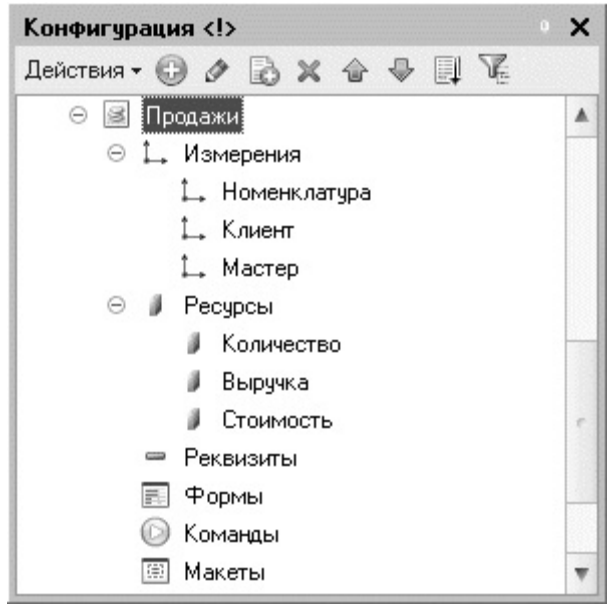


Рис. 12.2. Оборотный регистр накопления «Продажи»

Теперь отредактируем командный интерфейс, чтобы в подсистемах *Бухгалтерия*, *Оказание услуг* и *Учет материалов* была доступна ссылка для просмотра нашего оборотного регистра накопления.

В дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*.

В открывшемся окне слева в списке *Подсистемы* выделим подсистему *Бухгалтерия*.

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

В группе *Панель навигации.Обычное* включим видимость у команды *Продажи* и мышью перетащим ее в группу *Панель навигации.См.также.*

Аналогично, выделив подсистемы *ОказаниеУслуг* и *УчетМатериалов*, в панели навигации включим видимость у команды *Продажи* и перенесем ее в группу *См.также*.

Проведение документа «Оказание услуги» по трем регистрам

В этом разделе мы сначала изменим процедуру проведения документа *ОказаниеУслуги*, а затем в режиме *1С:Предприятие* перепроведем все эти документы, чтобы отработал новый, измененный нами алгоритм проведения документов *Оказание услуги*.

В режиме «Конфигуратор»

Откроем окно редактирования объекта конфигурации *Документ ОказаниеУслуги* и на закладке *Движения* укажем, что этот документ будет создавать движения еще и по регистру *Продажи*.

Перейдем на закладку *Прочее* и откроем модуль документа. Для этого нажмем кнопку *Модуль объекта*.

Откроем процедуру обработчика события *ОбработкаПроведения*.

В конце цикла после строки *КонецЕсли* и перед строкой *КонецЦикла* добавим строки кода, создающие движения регистра *Продажи*, производимые документом *ОказаниеУслуги* (листинг 12.1).

Листинг 12.1. Движения документа «ОказаниеУслуги» (фрагмент)

```
// Регистр Продажи
Движение = Движения.Продажи.Добавить();
Движение.Период = Дата;
Движение.Номенклатура = ТекСтрокаПеречеНоменклатуры.Номенклатура;
Движение.Клиент = Клиент;
Движение.Мастер = Мастер;
Движение.Количество = ТекСтрокаПеречеНоменклатуры.Количество;
Движение.Выручка = ТекСтрокаПеречеНоменклатуры.Сумма;
Движение.Стоимость = ТекСтрокаПеречеНоменклатуры.Стоимость *
ТекСтрокаПеречеНоменклатуры.Количество;
```

Перед началом цикла установим свойство *Записывать* набора записей движений по этому регистру в значение *Истина*.

В результате процедура *ОбработкаПроведения* будет выглядеть следующим образом (листинг 12.2).

Листинг 12.2. Движения документа «ОказаниеУслуги»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  Движения.ОстаткиМатериалов.Записывать = Истина;
  Движения.СтоимостьМатериалов.Записывать = Истина;
  Движения.Продажи.Записывать = Истина;

  Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

    Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
    Перечисления.ВидыНоменклатуры.Материал Тогда

      // Регистр ОстаткиМатериалов Расход
      Движение = Движения.ОстаткиМатериалов.Добавить();
      Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
      Движение.Период = Дата;
      Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
      Движение.Склад = Склад;
      Движение.Количество = ТекСтрокаПереченьНоменклатуры.Количество;

      // Регистр СтоимостьМатериалов Расход
      Движение = Движения.СтоимостьМатериалов.Добавить();
      Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
      Движение.Период = Дата;
      Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
      Движение.Стоимость = ТекСтрокаПереченьНоменклатуры.Количество *
```

```
        ТекСтрокаПеречеНьНоменклатуры.Стоимость;

    КонецЕсли;

    // Регистр Продажи
    Движение = Движения.Продажи.Добавить();
    Движение.Период = Дата;
    Движение.Номенклатура = ТекСтрокаПеречеНьНоменклатуры.Номенклатура;
    Движение.Клиент = Клиент;
    Движение.Мастер = Мастер;
    Движение.Количество = ТекСтрокаПеречеНьНоменклатуры.Количество;
    Движение.Выручка = ТекСтрокаПеречеНьНоменклатуры.Сумма;
    Движение.Стоимость = ТекСтрокаПеречеНьНоменклатуры.Стоимость *
    ТекСтрокаПеречеНьНоменклатуры.Количество;

  КонецЦикла;

КонецПроцедуры
```

Все добавленные конструкции вам уже хорошо известны.

Обратите внимание лишь на то, что у оборотного регистра отсутствует свойство *ВидДвижения*, поскольку отражение вида движения (приход или расход) имеет смысл лишь при учете остатков. В случае регистра оборотов нас интересует только значение, которое должно быть записано в ресурс регистра.

Также заметьте, что мы разместили команды, создающие движения в регистре

Продажи, в конце цикла обхода строк табличной части документа, после условия выполнения цикла только для материалов. Это важно, так как движения в этом регистре создаются как для материалов, так и для услуг.

В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра *Продажи*, связанному с документом.

Для этого откроем форму документа *ОказаниеУслуги*.

В левом верхнем окне перейдем на закладку *Командный интерфейс*.

В разделе *Панель навигации* раскроем группу *Перейти* и увидим команду для открытия регистра накопления *Продажи*.

Установим свойство *Видимость* для этой команды.

В режиме «1С:Предприятие»

В режиме *1С:Предприятие* нам нужно перепровести все документы оказания услуг и проверить, что они создают правильные движения в регистре *Продажи*.

Запустим «1С:Предприятие» в режиме отладки и откроем по очереди каждый

документ *Оказание услуги*.

Нажмем *Провести* и перейдем к списку движений этих документов по регистру *Продажи*. Они должны иметь следующий вид (рис. 12.3а, 12.3б, 12.4а, 12.4б, 12.5а, 12.5б).

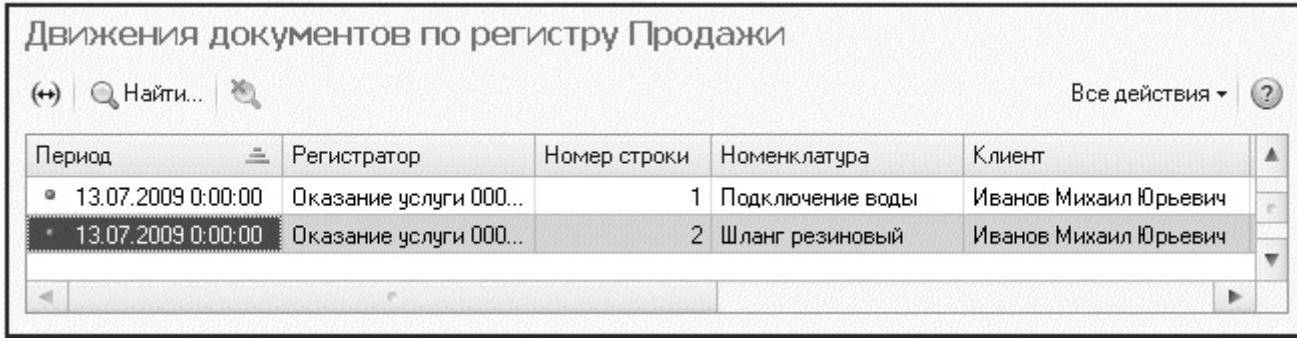


Рис. 12.3а. Движения документа «Оказание услуги № 1» в регистре «Продажи»

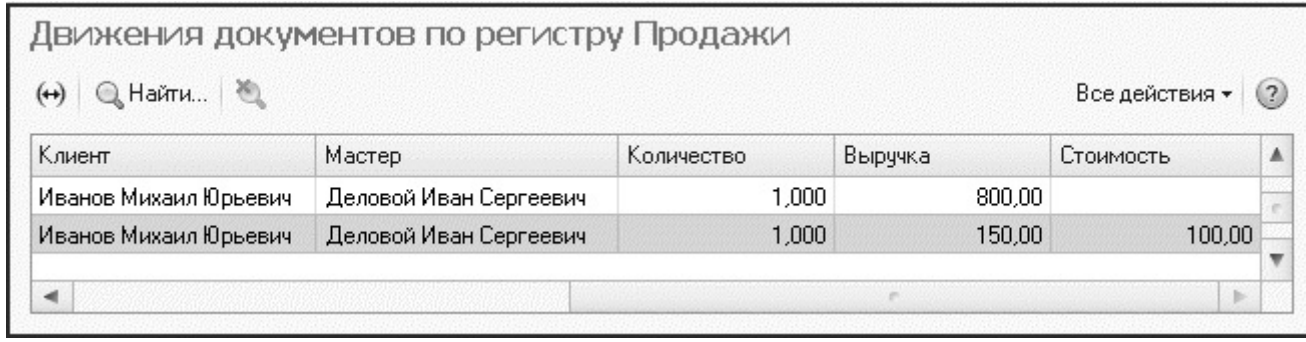


Рис. 12.3б. Движения документа «Оказание услуги № 1» в регистре «Продажи»

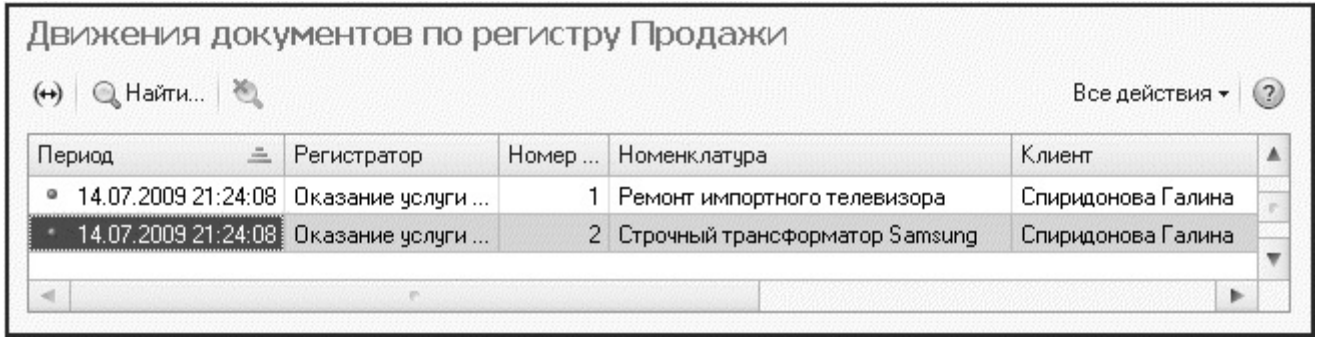


Рис. 12.4а. Движения документа «Оказание услуги № 2» в регистре «Продажи»

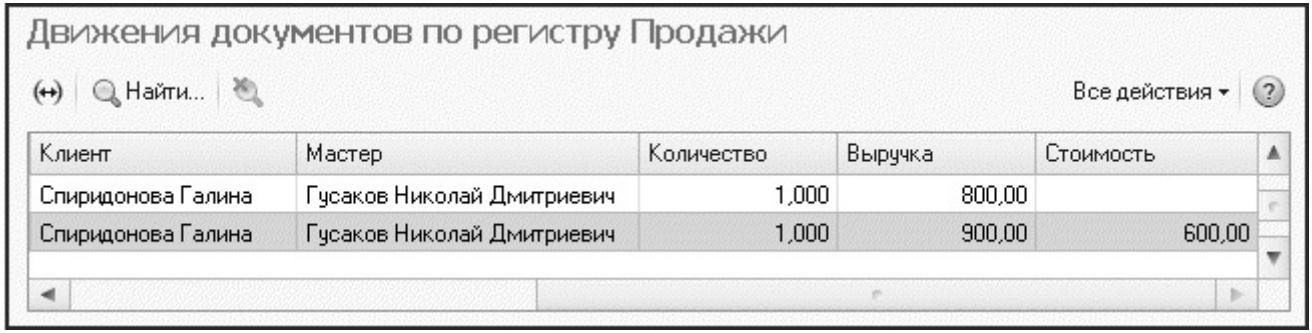


Рис. 12.4б. Движения документа «Оказание услуги № 2» в регистре «Продажи»

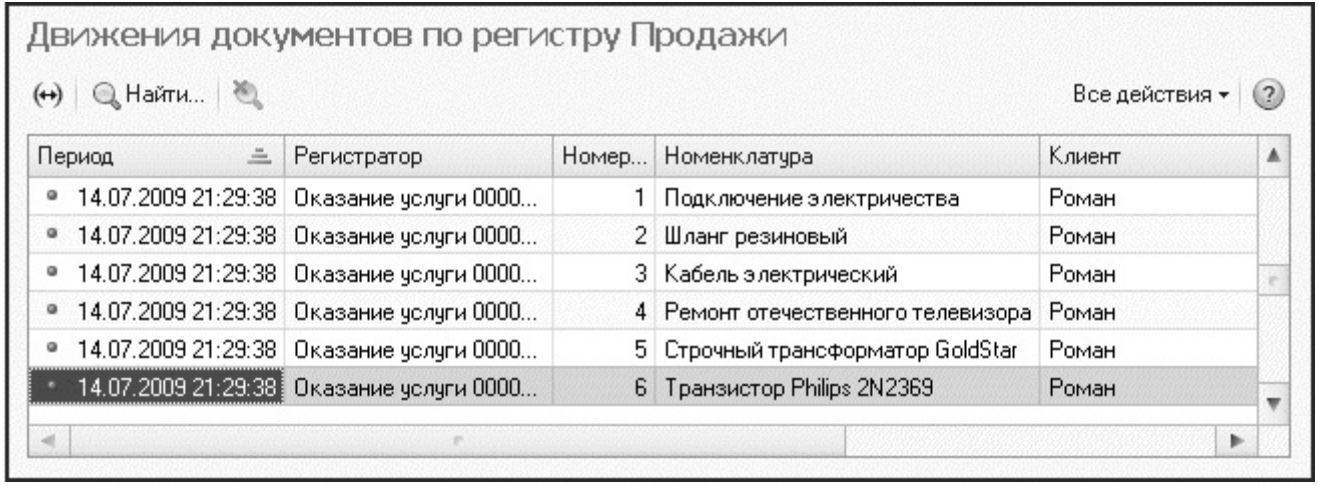


Рис. 12.5а. Движения документа «Оказание услуги № 3» в регистре «Продажи»

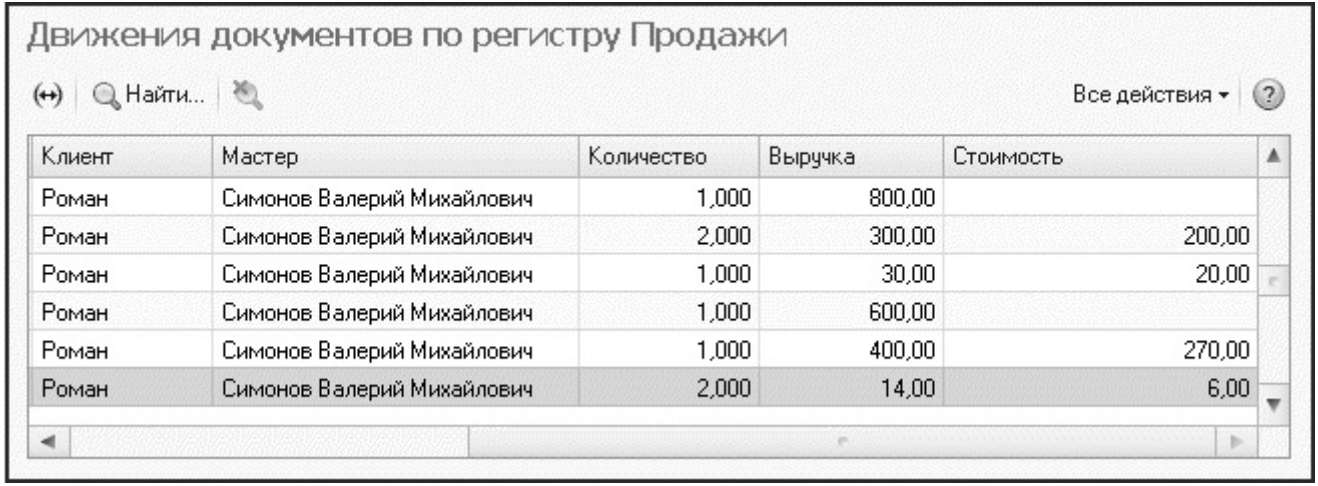


Рис. 12.5б. Движения документа «Оказание услуги № 3» в регистре «Продажи»

Теперь у нас есть практически вся необходимая информация для анализа деятельности ООО «На все руки мастер».

На следующем занятии мы займемся с вами тем, что создадим несколько отчетов, представляющих нам итоговую информацию о работе предприятия.

Контрольные вопросы

- *Что такое оборотный регистр накопления.*
- *В чем отличие между регистром накопления остатков и оборотным регистром накопления.*
- *Как выбирать реквизиты и измерения при создании регистров накопления.*
- *Как создать оборотный регистр накопления.*

Занятие 13 (4:30). Отчеты

Продолжительность

Ориентировочная продолжительность занятия – 4 часа 30 минут.

Поскольку общая продолжительность занятия довольно велика, то можно делать это занятие частями, прерываясь после каждого из шести отчетов, разрабатываемых на этом занятии.

Настало время, чтобы познакомиться с одним важным инструментом платформы «1С:Предприятие» – *системой компоновки данных*. На этом занятии мы рассмотрим построение нескольких отчетов, которые будут использоваться в нашей конфигурации, и на их примере объясним основные возможности системы компоновки данных.

Любой отчет, как правило, подразумевает получение сложной выборки данных, сгруппированных и отсортированных определенным образом. Система компоновки данных представляет собой мощный и гибкий механизм, позволяющий выполнить все необходимые действия – от получения данных из различных источников до представления этих данных в виде, удобном для пользователя.

Чаще всего исходные данные, необходимые для отчета, находятся в базе данных. Для того чтобы указать системе компоновки данных, какая информация и откуда должна быть получена, используется язык запросов системы «1С:Предприятие 8».

На этапе разработки отчета можно задать стандартные настройки отчета для того, чтобы пользователь мог сразу же запустить отчет на выполнение. В то же время пользователь может самостоятельно изменить настройки отчета и выполнить его. При этом система компоновки данных сгенерирует другой запрос и другим образом представит конечные данные – в соответствии с новыми настройками, заданными пользователем.

В начале этого занятия мы познакомимся с общими сведениями о языке запросов системы «1С:Предприятие» и о системе компоновки данных.

Затем на примерах создания конкретных отчетов мы научимся использовать систему компоновки данных для решения различных практических задач.

«Теория». Способы доступа к данным

Система «1С:Предприятие 8» поддерживает два способа доступа к данным, хранящимся в базе данных:

- объектный (для чтения и записи);
- табличный (для чтения).

Объектный способ доступа к данным реализован посредством использования объектов встроенного языка.

С некоторыми из этих объектов мы уже познакомились на предыдущих занятиях.

Важной особенностью объектного способа доступа к данным является то, что, обращаясь к какому-либо объекту встроенного языка, мы обращаемся к некоторой совокупности данных, находящихся в базе данных, как к единому целому.

Например, объект *ДокументОбъект.ОказаниеУслуги* будет содержать значения всех реквизитов документа *Оказание услуги* и всех его табличных частей.

Объектная техника обеспечивает сохранение целостности объектов, кеширование объектов, вызов соответствующих обработчиков событий и т. д.

Табличный доступ к данным в «1С:Предприятии 8» реализован с помощью запросов к базе данных, которые составляются на *языке запросов*.

В этой технике разработчик получает возможность оперировать отдельными полями таблиц базы данных, в которых хранятся те или иные данные.

Табличная техника предназначена для получения информации из базы данных по некоторым условиям (отбор, группировка, сортировка, объединение нескольких выборок, расчет итогов и т. д.). Табличная техника оптимизирована для обработки больших объемов информации, расположенной в базе данных, и получения данных, отвечающих заданным критериям.

Работа с запросами

Для работы с запросами используется объект встроенного языка *Запрос*. Он позволяет получать информацию, хранящуюся в полях базы данных, в виде выборки, сформированной по заданным правилам.

Источники данных запросов

Исходную информацию запрос получает из набора таблиц. Эти таблицы представляют разработчику данные реальных таблиц базы данных в удобном для анализа виде.

Все таблицы, которыми оперирует язык запросов, можно разделить на две большие группы: реальные таблицы и виртуальные таблицы (рис. 13.1).

Рис. 13.1. Таблицы запросов

Посмотреть состав таблиц, доступных для запроса и их описание можно в синтакс-помощнике в разделе *Работа с запросами* > *Таблицы запросов*.

Отличительной особенностью реальных таблиц является то, что они содержат данные какой-либо одной реальной таблицы, хранящейся в базе данных.

Например, реальной является таблица *Справочник.Клиенты*, соответствующая справочнику *Клиенты,* или таблица *РегистрНакопления.ОстаткиМатериалов*, соответствующая регистру накопления *ОстаткиМатериалов*.

Виртуальные таблицы формируются в основном из данных нескольких таблиц базы данных.

Например, виртуальной является таблица *РегистрНакопления.ОстаткиМатериалов.ОстаткиИОбороты*, формируемая из нескольких таблиц регистра накопления *Остатки Материалов*.

Иногда виртуальные таблицы могут формироваться и из одной реальной таблицы (например, виртуальная таблица *Цены.СрезПоследних* формируется на основе таблицы регистра сведений *Цены*).

Однако общим для всех виртуальных таблиц является то, что им можно задать ряд параметров, определяющих, какие данные будут включены в эти виртуальные таблицы. Набор таких параметров может быть различным для разных виртуальных таблиц и определяется данными, хранящимися в исходных таблицах базы данных.

Реальные таблицы подразделяются на *объектные* (ссылочные) и *необъектные* (нессылочные).

В объектных (ссылочных) таблицах представлена информация ссылочных типов данных (справочники, документы, планы видов характеристик и т. д.). А в

необъектных (нессылочных) – всех остальных типов данных (константы, регистры и т. д.).

Отличительной особенностью объектных (ссылочных) таблиц является то, что они включают в себя поле *Ссылка*, содержащее ссылку на текущую запись. Кроме этого, для таких таблиц возможно получение пользовательского представления объекта. Эти таблицы могут быть иерархическими, и поля таких таблиц могут содержать вложенные таблицы (табличные части).

Язык запросов

Алгоритм, по которому данные будут выбраны из исходных таблиц запроса, описывается на специальном языке – *языке запросов*.

Текст запроса может состоять из нескольких частей:

- описание запроса,
- объединение запросов,
- упорядочивание результатов,
- автоупорядочивание,
- описание итогов.

Обязательной частью запроса является только первая – описание запроса.

Все остальные присутствуют по необходимости.

Описание запроса определяет источники данных, поля выборки, группировки и т. д.

Объединение запросов определяет, как будут объединены результаты выполнения нескольких запросов.

Упорядочивание результатов определяет условия упорядочивания строк результата запроса.

Автоупорядочивание позволяет включить режим автоматического упорядочивания строк результата запроса.

Описание итогов определяет, какие итоги необходимо рассчитывать в запросе и каким образом группировать результат.

Следует заметить, что в случае, когда язык запросов используется для описания источников данных в системе компоновки данных, секция описания итогов языка запросов не используется. Это связано с тем, что система компоновки данных самостоятельно рассчитывает итоги на основании тех настроек, которые сделаны разработчиком или пользователем.

Применение различных синтаксических конструкций языка запросов подробно описано во встроенной справке в режиме *Конфигуратор*: *Справка > Содержание справки > «1С:Предприятие» > Встроенный язык > Работа с запросами*, а также в документации «1С:Предприятие 8.2. Руководство разработчика», глава 8 «Работа с запросами».

Детальнее с языком запросов мы познакомимся далее, в процессе создания конкретных отчетов.

Мы не будем писать запросы руками. Для большинства отчетов, разрабатываемых с помощью системы компоновки данных, запрос можно создать при помощи конструктора запросов.

Поэтому наша задача на этом занятии – научиться читать и понимать тексты этих запросов, чтобы в дальнейшем иметь возможность изменять их.

Система компоновки данных

Система компоновки данных предназначена для создания произвольных отчетов в системе «1С:Предприятие» и состоит из нескольких основных частей.

Исходные данные для компоновки отчета содержит в себе *схема компоновки данных.* Это наборы данных и методы работы с ними (рис. 13.2).

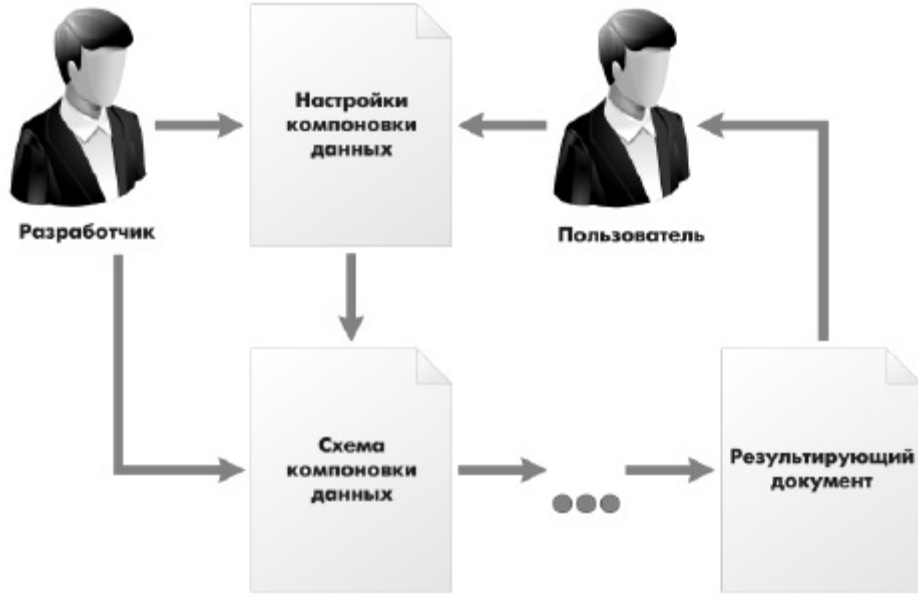


Рис. 13.2. Общая схема работы с системой компоновки данных

Разработчик создает схему компоновки данных, в которой описывает текст запроса, наборы данных, связи между ними, доступные поля, параметры получения данных, и задает первоначальные настройки компоновки – структуру отчета, макет оформления данных и др.

Например, схема компоновки может содержать следующий набор данных (рис. 13.3).

Наборы данных | Связи наборов данн... | Вычисляемые поля | Ресурсы | Параметры | Макеты | Вложенные схемы | Настройки

Поля:

- Наборы данных
  - НаборДанных1

Источник данных

Поле	Путь	Ограничение поля				Роль	Выражени...	Проверка иер...
	Автозаголовок	По...	Ус...	Гр...	Уп...		Выражения упорядочи...	Набор данных
		Ограничение реквизи...						Параметр
		По...	Ус...	Гр...	Уп...			
ВыручкаОборот	ВыручкаОборот	☐	☐	☐	☐			
	☐ Выручка Оборот	✓	✓	✓	✓			
Клиент	Клиент	☐	☐	☐	☐	Измерение		
	☐ Клиент	☐	☐	☐	☐			
КоличествоОборот	КоличествоОборот	☐	☐	☐	☐			
	☐ Количество Оборот	✓	✓	✓	✓			
Мастер	Мастер	☐	☐	☐	☐	Измерение		
	☐ Мастер	☐	☐	☐	☐			
Номенклатура	Номенклатура	☐	☐	☐	☐	Измерение		
	☐ Номенклатура	☐	☐	☐	☐			
СтоимостьОборот	СтоимостьОборот	☐	☐	☐	☐			
	☐ Стоимость Оборот	✓	✓	✓	✓			

Запрос: Конструктор запроса...

```
ВЫБРАТЬ
    ПродажиОбороты.Номенклатура,
    ПродажиОбороты.Клиент,
    ПродажиОбороты.Мастер,
    ПродажиОбороты.КоличествоОборот,
    ПродажиОбороты.ВыручкаОборот,
    ПродажиОбороты.СтоимостьОборот
ИЗ
    РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
```

Текст запроса

Автозаполнение ☑

Рис. 13.3. Пример схемы компоновки (набор данных и запрос, его использующий)

На приведенном рисунке показано окно конструктора схемы компоновки данных, в котором содержится источник данных, текст запроса и поля, выбранные запросом.

Отчет системы компоновки данных, который получит пользователь, представляет собой не просто таблицу записей, удовлетворяющих условиям запроса.

Отчет системы компоновки имеет сложную иерархическую структуру и может состоять из различных элементов, таких как группировки, таблицы и диаграммы.

При этом пользователь может изменить существующую структуру отчета или вообще создать совершенно новую структуру отчета. Может настроить необходимый ему отбор, оформление элементов структуры отчета, получить расшифровку по каждому элементу и т. д.

Например, может быть задана такая структура отчета, состоящая из одной таблицы и одной диаграммы (рис. 13.4).

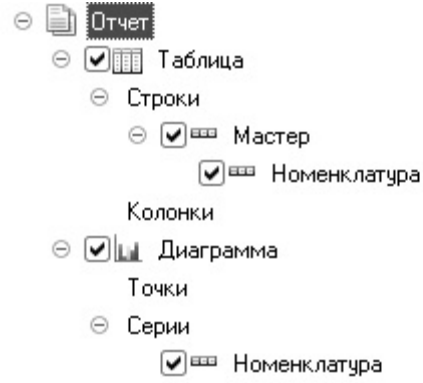


Рис. 13.4. Возможная структура отчета

В этом случае сформированный отчет будет иметь следующий вид (рис. 13.5).

Мастер / Номенклатура	Итого Выручка Оборот
Гусаков Николай Дмитриевич	1 700,00
Ремонт импортного телевизора	800,00
Строчный трансформатор Samsung	900,00
Деловой Иван Сергеевич	950,00
Подключение воды	800,00
Шланг резиновый	150,00
Симонов Валерий Михайлович	2 144,00
Кабель электрический	30,00
Подключение электричества	600,00
Ремонт отечественного телевизора	600,00
Строчный трансформатор GoldStar	400,00
Транзистор Philips 2N2369	14,00
Шланг резиновый	300,00
Итого	4 794,00

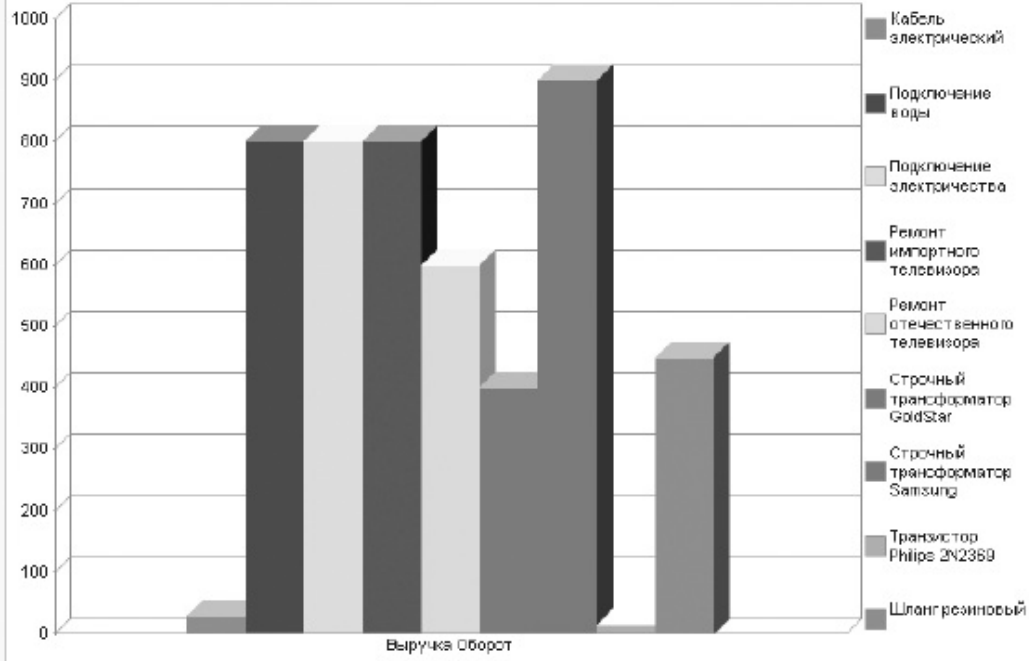


Рис. 13.5. Пример отчета

В представленном отчете таблица будет состоять из записей регистра накопления *ПродажиОбороты* о клиентах и оказанных им услугах. Эти записи сгруппированы по мастерам, которые выполняли заказы. А в группировке будет выведен список услуг, оказанных данным мастером, и затраченных на это материалов.

Как мы уже говорили в начале раздела, система компоновки данных представляет собой совокупность нескольких объектов. При формировании и исполнении отчета происходит последовательная передача данных от одного объекта системы компоновки данных к другому, до получения конечного результата – документа, показанного пользователю.

Алгоритм взаимодействия этих объектов выглядит следующим образом:

- Разработчик создает *схему компоновки данных* и *настройки по умолчанию*. В общем случае на основе одной схемы компоновки данных может быть создано большое количество различных отчетов. Настройки компоновки данных, создаваемые разработчиком или изменяемые пользователем, определяют, какой именно отчет будет получен в конкретном случае.
- На основе схемы компоновки и имеющихся настроек *компоновщик макета*

создает *макет*. Это этап подготовки к исполнению отчета. Макет компоновки данных является уже готовым заданием для выполнения процессором компоновки. Он содержит необходимые запросы, макеты областей отчета и др.

- *Процессор компоновки* данных выбирает данные из информационной базы согласно макету компоновки, агрегирует и оформляет эти данные.
- *Результат компоновки* обрабатывается *процессором вывода*, и в итоге пользователь получает результирующий табличный документ.

Эту последовательность работы можно представить в виде следующей схемы (рис. 13.6).


Схема компоновки данных
Компоновщик макета
Макет компоновки данных
Настройки компоновки данных
Информационная база
Процессор компоновки данных
Результирующий документ
Процессор вывода
Результат компоновки

Рис. 13.6. Схема работы системы компоновки

Выбор данных из одной таблицы

Создадим отчет *Реестр документов оказание услуги*, используя систему компоновки данных.

На примере этого отчета мы покажем, как выбрать данные из одной таблицы базы данных и как вывести их в определенном порядке. Также мы познакомимся с тем, как использовать расшифровку в готовом отчете.

Этот отчет будет выводить список существующих в базе данных документов *ОказаниеУслуги* в порядке их дат и номеров (рис. 13.7).

Документ	Склад	Мастер	Клиент
Оказание услуги 000000001 от 13.07.2009 0:00:00	Основной	Деловой Иван Сергеевич	Иванов Михаил Юрьевич
Оказание услуги 000000002 от 14.07.2009 21:24:08	Основной	Гусаков Николай Дмитриевич	Спиридонова Галина
Оказание услуги 000000003 от 14.07.2009 21:29:38	Основной	Симонов Валерий Михайлович	Роман

Рис. 13.7. Результат отчета

В режиме «Конфигуратор»

Добавим в конфигураторе объект конфигурации *Отчет*. Повторим первые

шаги по созданию отчета, описанные нами в занятии № 6.

На закладке *Основные* зададим имя отчета – *РеестрДокументовОказаниеУслуги*.

Установим свойство *Расширенное представление* как *Список оказанных услуг* для представления отчета в интерфейсе программы.

Создадим схему компоновки данных для отчета. Для этого нажмем кнопку *Открыть схему компоновки данных* или кнопку открытия  со значком лупы (рис. 13.8).

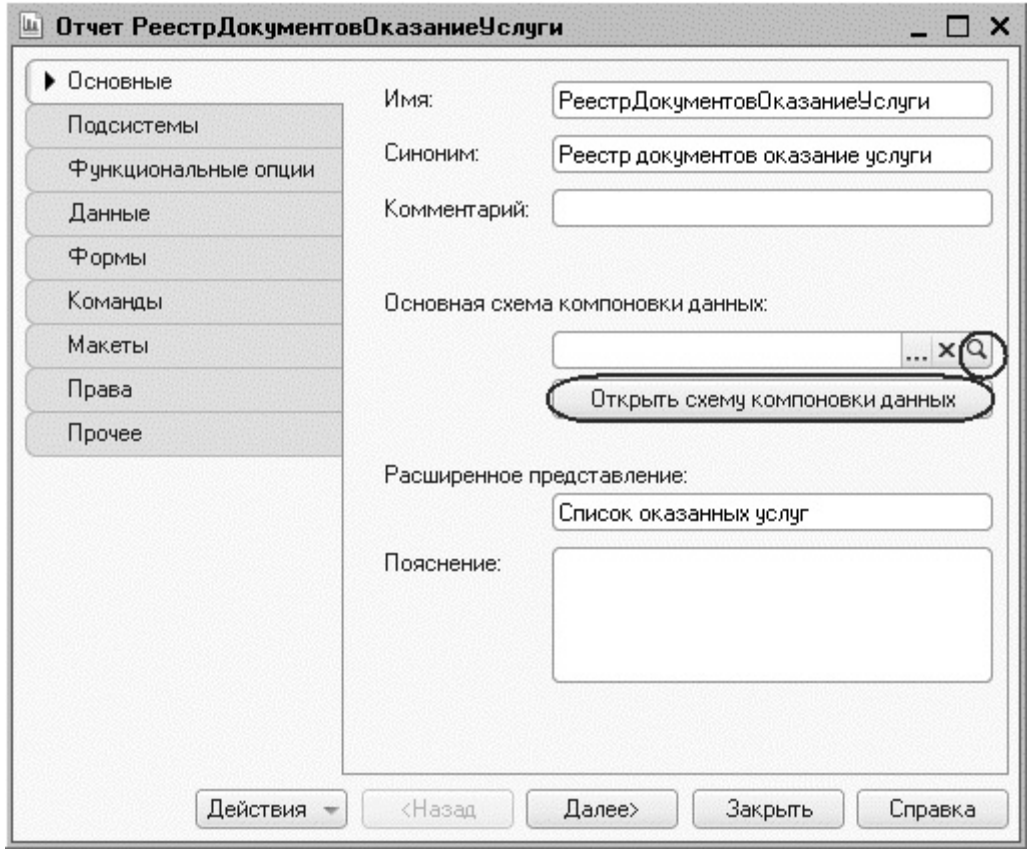


Рис. 13.8. Основные свойства отчета

В открывшемся диалоговом окне конструктора макета нажмем *Готово*. В конструкторе схемы компоновки данных создадим *Набор данных – запрос* (рис. 13.9).

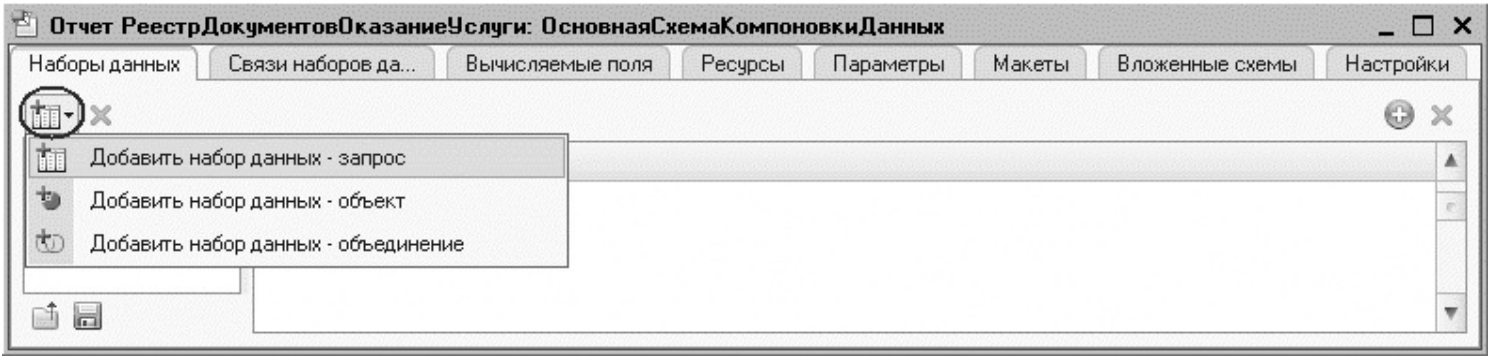


Рис. 13.9. Создание набора данных – запрос

Запрос для набора данных

Нажав кнопку *Конструктор запроса*, запустим конструктор запроса.

В качестве источника данных для запроса выберем объектную (ссылочную) таблицу документа *ОказаниеУслуги*.

Из этой таблицы выберем следующие поля (рис. 13.10):

- *Склад,*
- *Мастер,*
- *Клиент,*
- *Ссылка.*

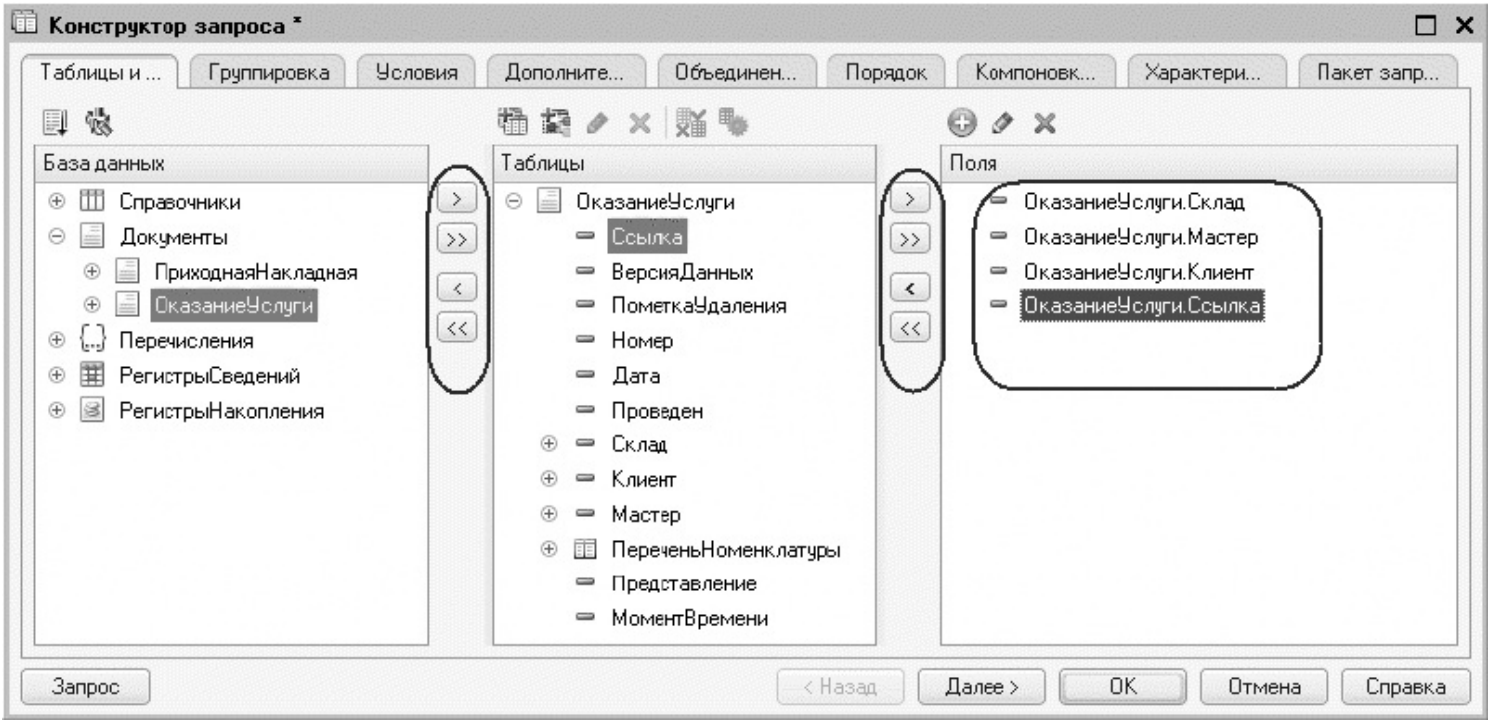


Рис. 13.10. Выбранные поля для запроса

ПРИМЕЧАНИЕ

Выделенные элементы можно перенести из одного списка в другой перетаскиванием мышью или двойным щелчком на них. Либо можно использовать кнопки >, <, >>, <<.

Псевдонимы полей

Перейдем на закладку *Объединения/Псевдонимы* и укажем, что поле *Ссылка* будет иметь псевдоним *Документ* (рис. 13.11).

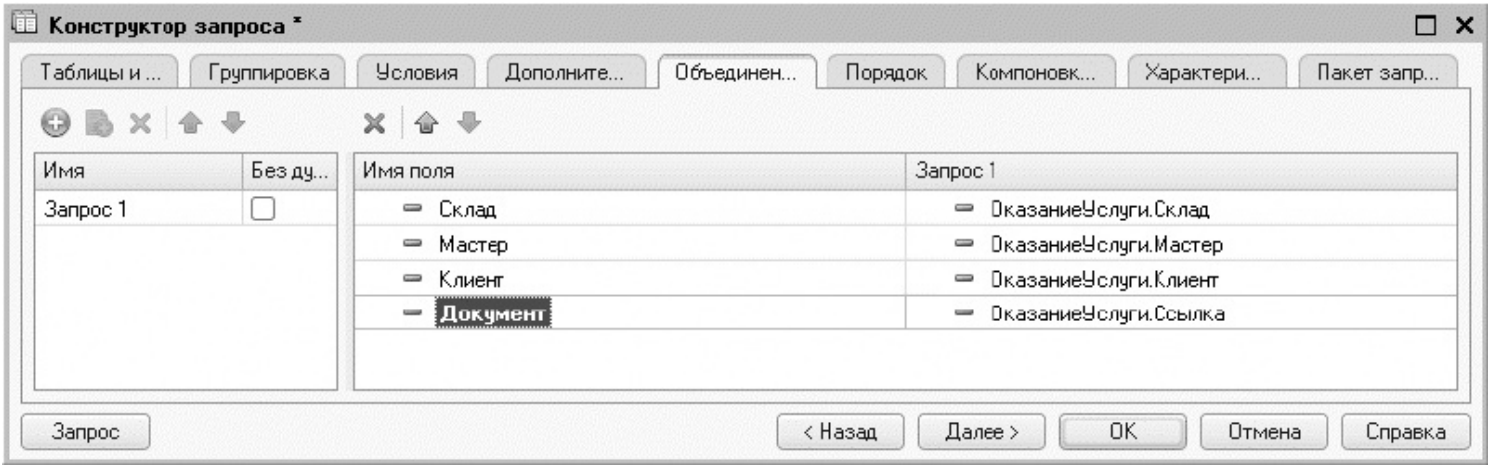


Рис. 13.11. Установка псевдонимов полей запроса

СОВЕТ

Имена полей лучше изменять в запросе, так как в этом случае в схему компоновки данных они перенесутся сразу в три колонки: Поле, Путь и Заголовок, и не нужно будет лишний раз их изменять.

Порядок записей

После этого перейдем на закладку *Порядок* и укажем, что результат запроса должен быть упорядочен по значению поля *Документ* (рис. 13.12).

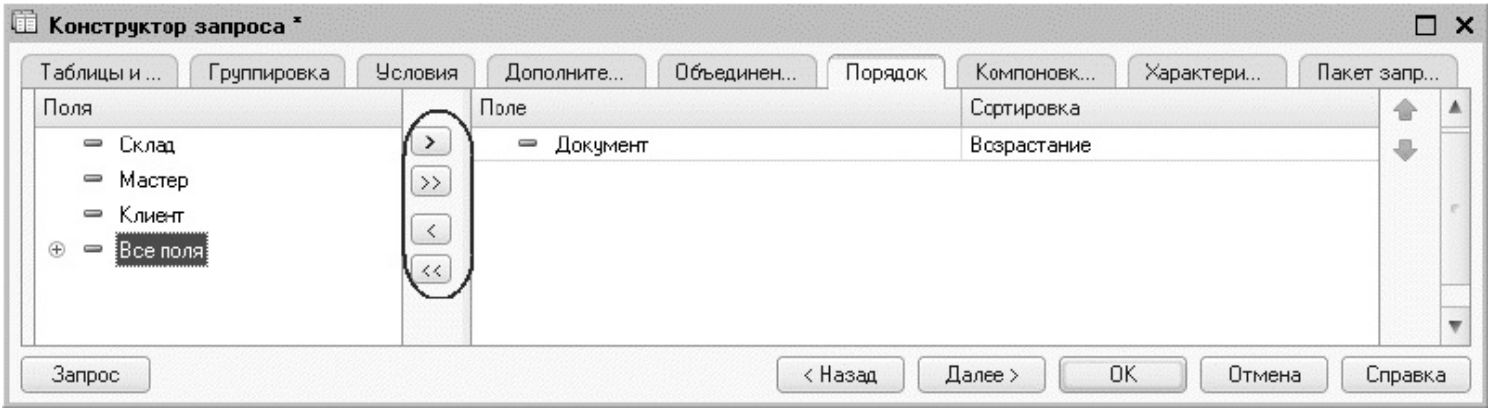


Рис. 13.12. Порядок записей в запросе

Анализ текста запроса

Нажмем *OK* и посмотрим, какой запрос сформировал конструктор запроса (листинг 13.1).

Листинг 13.1. Текст запроса

```
ВЫБРАТЬ
  ОказаниеУслуги.Склад,
  ОказаниеУслуги.Мастер,
  ОказаниеУслуги.Клиент,
  ОказаниеУслуги.Ссылка КАК Документ
ИЗ
  Документ.ОказаниеУслуги КАК ОказаниеУслуги
УПОРЯДОЧИТЬ ПО
  Документ
```

Текст запроса начинается, как мы говорили выше, с *части описания запроса* (листинг 13.2).

Листинг 13.2. Описание запроса

```
ВЫБРАТЬ
  ОказаниеУслуги.Склад,
  ОказаниеУслуги.Мастер,
  ОказаниеУслуги.Клиент,
  ОказаниеУслуги.Ссылка КАК Документ
ИЗ
  Документ.ОказаниеУслуги КАК ОказаниеУслуги
```

Описание запроса начинается с обязательного ключевого слова *ВЫБРАТЬ*.

Затем следует *список полей выборки*. В нем описываются поля, которые должны содержаться в результате запроса. Этот список может содержать как собственно поля, так и некоторые выражения, вычисляемые на основе значений полей.

После ключевого слова *ИЗ* указываются *источники данных* – исходные таблицы запроса, содержимое которых обрабатывается в запросе.

В данном случае это объектная (ссылочная) таблица

Документ.ОказаниеУслуги.

После ключевого слова *КАК* указывается *псевдоним источника данных*.

В нашем случае это *ОказаниеУслуги*. В дальнейшем к этому источнику данных можно будет обращаться в тексте запроса, используя его псевдоним.

Такое обращение мы видим в описании полей выборки (листинг 13.3).

Листинг 13.3. Описание полей выборки

```
ВЫБРАТЬ
  ОказаниеУслуги.Склад,
  ОказаниеУслуги.Мастер,
  ОказаниеУслуги.Клиент,
  ОказаниеУслуги.Ссылка КАК Документ
```

Поля выборки также могут иметь псевдонимы, по которым в дальнейшем в тексте запроса можно обращаться к этому полю. В нашем случае это псевдоним *Документ* у поля *Ссылка*.

После части описания запроса в нашем примере следует *часть упорядочивания результатов* (листинг 13.4).

Листинг 13.4. Упорядочивание результатов запроса

```
УПОРЯДОЧИТЬ ПО
  Документ
```

Предложение *УПОРЯДОЧИТЬ ПО* позволяет сортировать строки в результате запроса. После этого ключевого предложения располагается выражение упорядочивания, которое в общем случае представляет собой перечисление полей (выражений) и порядка вывода.

В нашем случае упорядочивание будет выполняться по полю *Документ*, оно же поле *ОказаниеУслуги.Ссылка*. Порядок сортировки будет по возрастанию (если порядок сортировки не указан явно, выполняется сортировка по возрастанию).

На этом закончим изучение текста запроса и перейдем к настройке схемы компоновки данных.

Настройки

Перейдем на закладку *Настройки* и создадим стандартные настройки, определяющие, как будет выводиться информация в отчет.

Иерархическая структура отчета может содержать в различных сочетаниях три

основных элемента:

- *Группировка* – для вывода информации в виде обычного линейного отчета;
- *Таблица* – для вывода информации в виде таблицы;
- *Диаграмма* – для вывода информации в виде диаграммы.

Для добавления нового элемента, в нашем случае группировки, выделим в дереве структуры отчета корневой элемент *Отчет* и вызовем его контекстное меню.

Можно также нажать кнопку *Добавить*, расположенную в командной панели окна, или нажать клавишу *Ins* (рис. 13.13).

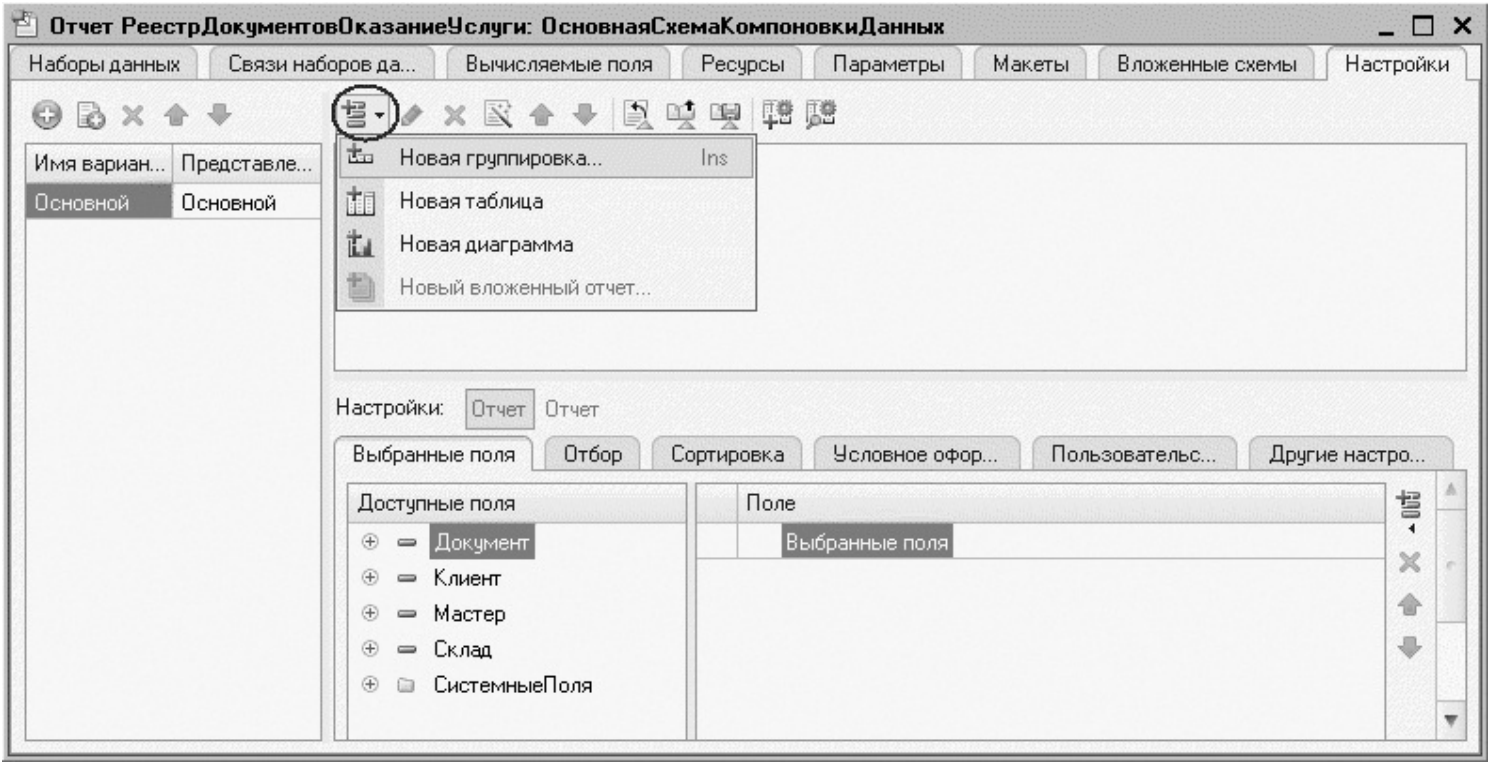


Рис. 13.13. Добавление новой группировки в отчет

В окне выбора поля группировки просто нажмем *OK* (тем самым мы указываем, что в группировке будут выводиться детальные записи из информационной базы).

В структуре отчета появится группировка *Детальные записи*.

На закладке *Выбранные поля* перенесем мышью из списка доступных полей те поля, которые будут выводиться в отчет:

- *Документ,*
- *Склад,*
- *Мастер,*
- *Клиент.*

ПРИМЕЧАНИЕ

Добавление доступных полей в список выбранных полей можно осуществить перетаскиванием мышью, двойным щелчком на доступных полях либо нажатием кнопки ***Добавить*** *справа от списка выбранных полей. Порядок выбранных полей можно изменить позже кнопками* ***Вверх****,* ***Вниз*** *или перетаскиванием мышью.*

В результате окно настроек отчета должно иметь вид (рис. 13.14).

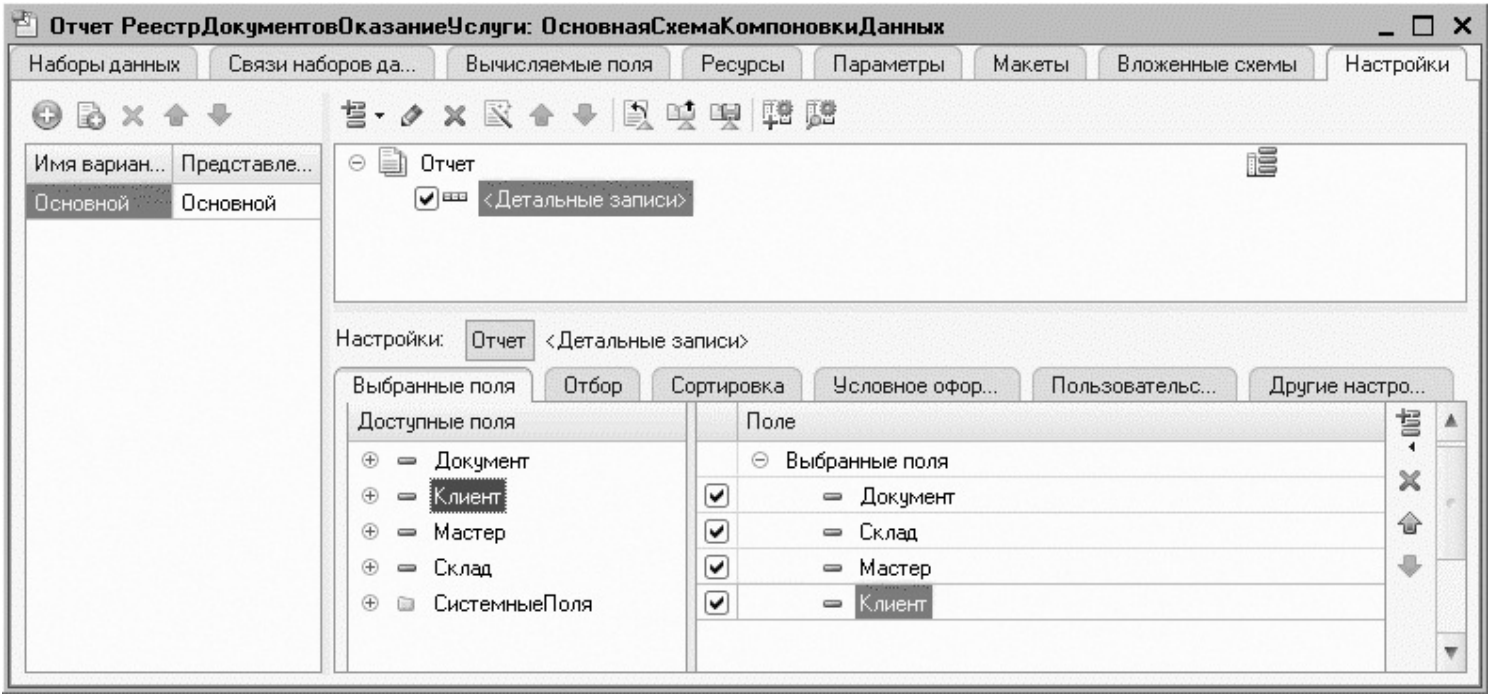


Рис. 13.14. Окно настроек отчета

На этом создание отчета закончено.

В заключение определим, в каких подсистемах будет отображаться наш отчет. Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации Отчет *РеестрДокументовОказаниеУслуги* перейдем на закладку *Подсистемы*. Отметим в списке подсистему *Оказание услуг*.

Таким образом, ссылка на наш отчет автоматически попадет в панель действий этой подсистемы (рис. 13.15).

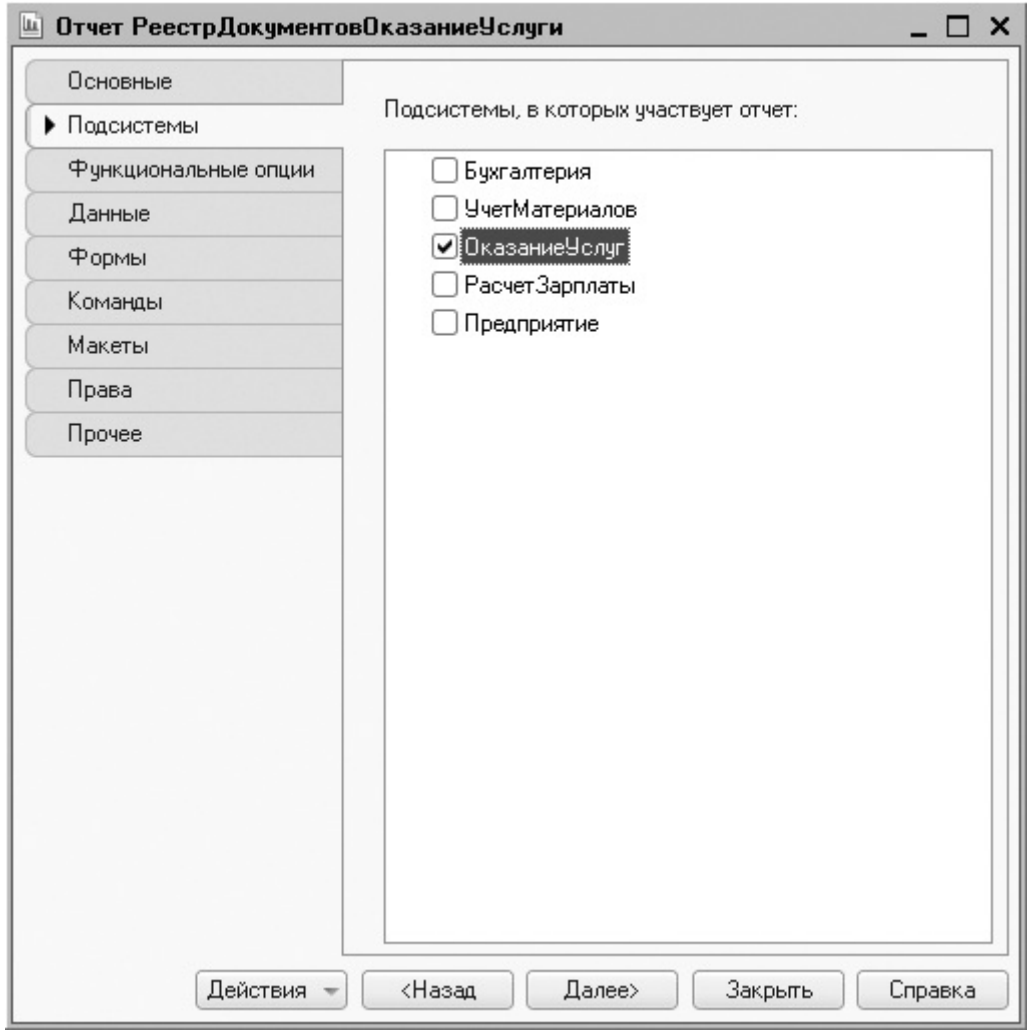


Рис. 13.15. Определение списка подсистем, в которых будет отражаться отчет

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает отчет.

В открывшемся окне «1С:Предприятия» мы видим, что в панели действий раздела *Оказание услуг* в группе команд для выполнения отчетов появилась команда для формирования отчета *Реестр документов оказание услуги*.

Причем если подвести к ней мышь, то появится всплывающая подсказка *Список оказанных услуг*, которая определяется свойством *Расширенное представление*, заданное нами для отчета.

Выполним эту команду (рис. 13.16).

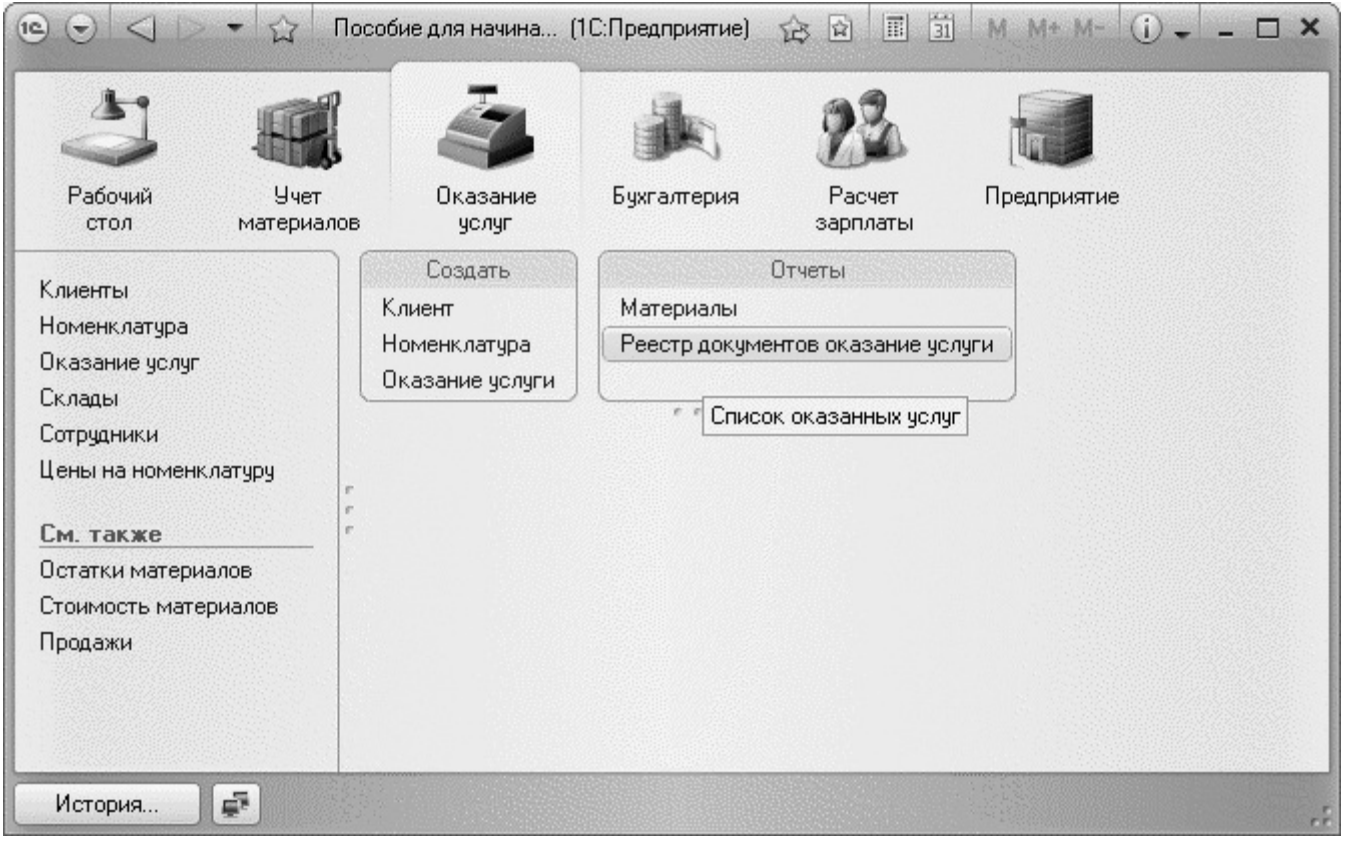


Рис. 13.16. Команда для формирования отчета

Перед нами откроется форма отчета, автоматически сформированная системой.

Заметьте, что заголовок этой формы также называется *Список оказанных*

услуг и определяется свойством *Расширенное представление*.

Нажмем кнопку *Сформировать* (рис. 13.17).

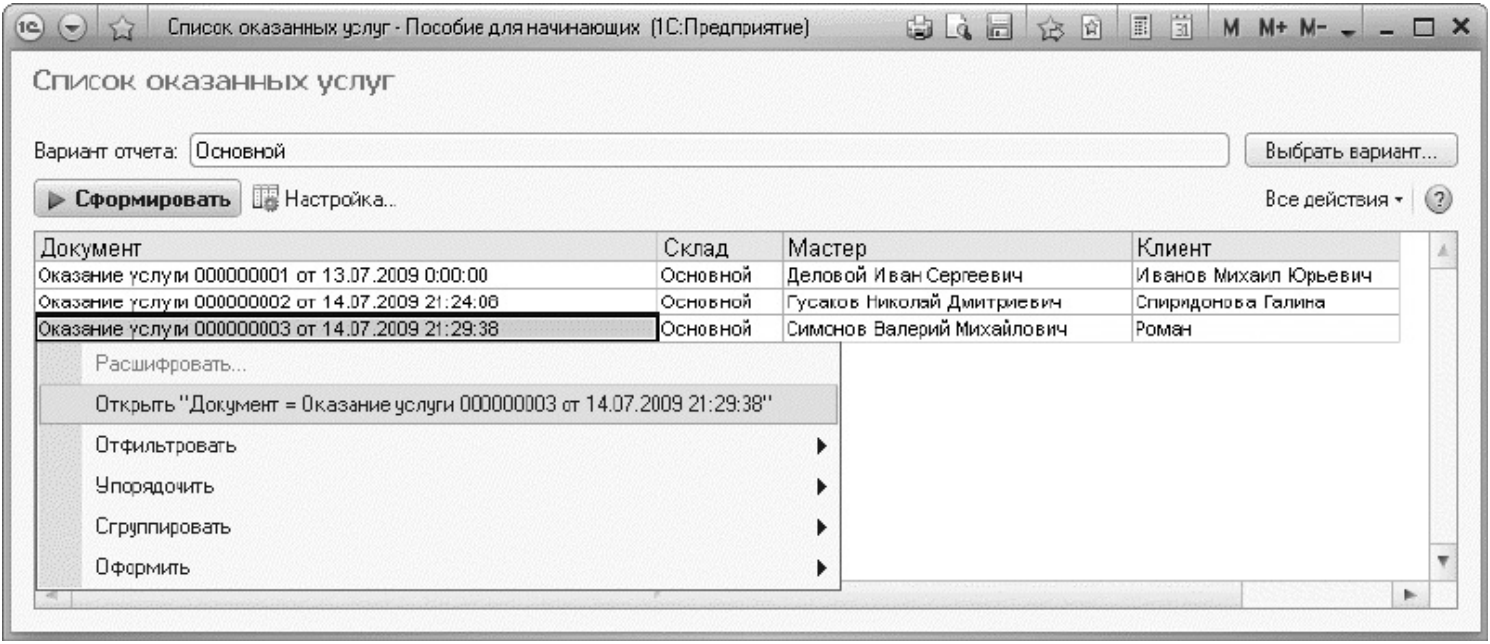


Рис. 13.17. Результат отчета

Мы видим, что отчет содержит реестр документов *Оказание услуги*.

Причем двойным щелчком мыши на поле *Документ* мы можем открыть исходный документ, а также выполнить другие действия «расшифровки»,

которые предоставляет нам система компоновки данных.

Таким образом, на примере этого отчета мы продемонстрировали, как использовать конструктор схемы компоновки данных, и познакомились с некоторыми основными конструкциями языка запросов.

Выбор данных из двух таблиц

Отчет *Рейтинг услуг* будет содержать информацию о том, выполнение каких услуг принесло ООО «На все руки мастер» наибольшую прибыль в указанном периоде (рис. 13.18).

Рейтинг услуг

Параметры данных: Дата начала = 01.07.2009
Дата окончания = 15.07.2009

Услуга	Выручка
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00
Ремонт отечественного телевизора	600,00
Диагностика	
Итого	3 000,00

Рис. 13.18. Результат отчета

На примере отчета *Рейтинг услуг* мы проиллюстрируем, как отбирать данные в некотором периоде, как задавать параметры запроса, как использовать в запросе данные из нескольких таблиц и как включать в результат запроса все данные одного из источников.

Также мы узнаем, как работать с параметрами системы компоновки данных, как использовать стандартные даты, и познакомимся с быстрыми пользовательскими настройками отчетов.

Кроме этого, мы научимся более детально настраивать отбор и условное оформление в отчетах.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Отчет.*

Назовем его *РейтингУслуг* и запустим конструктор схемы компоновки данных.

Добавим новый *Набор данных – запрос* и вызовем конструктор запроса.

Запрос для набора данных

Левое соединение двух таблиц

В качестве источника данных для запроса выберем объектную (ссылочную)

таблицу *Номенклатура* и виртуальную таблицу регистра накопления *Продажи.Обороты*.

Чтобы исключить неоднозначность имен в запросе, переименуем таблицу *Номенклатура* в *спрНоменклатура*.

Для этого выделим ее в списке *Таблицы*, вызовем ее контекстное меню и выберем пункт *Переименовать таблицу* (рис. 13.19).

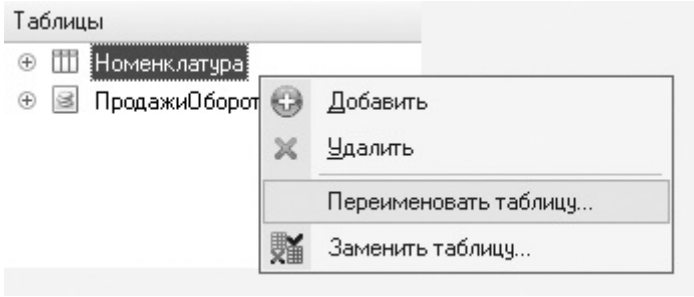


Рис. 13.19. Переименование таблицы в запросе

В список полей перенесем поля *СпрНоменклатура.Ссылка* и *ПродажиОбороты.ВыручкаОборот* из этих таблиц (рис. 13.20).

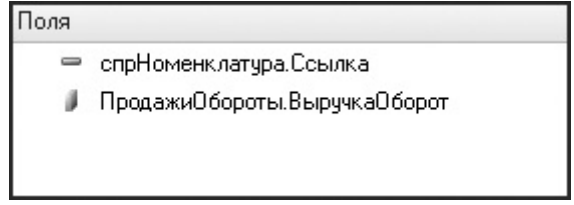


Рис. 13.20. Выбранные поля

Перейдем на закладку *Связи*.

Так как в запросе теперь участвуют несколько таблиц, требуется определить связь между ними.

По умолчанию платформой уже будет создана связь по полю *Номенклатура*. То есть значение измерения *Номенклатура* регистра *Продажи* должно быть равно ссылке на элемент справочника *Номенклатура*.

Но нам нужно снять флажок *Все* у таблицы *ПродажиОбороты* и установить его у таблицы *спрНоменклатура*.

Тем самым мы задаем тип связи как *Левое соединение*, то есть в результат запроса будут включены все записи справочника *Номенклатура* и те записи регистра *Продажи*, которые удовлетворяют условию связи по полю *Номенклатура.*

Таким образом, в результате запроса будут присутствовать все услуги, и для некоторых из них будут указаны обороты выручки. Для тех услуг, которые не производились в выбранном периоде, не будет указано ничего.

Описанную связь двух таблиц схематично можно представить следующим примером (рис. 13.21).

Ссылка
Материалы
Услуги
Строчный трансформатор Samsung
Строчный трансформатор GoldStar
Транзистор Philips 2N2369
Шланг резиновый
Кабель электрический
Диагностика
Ремонт отечественного телевизора
Ремонт импортного телевизора
Подключение воды
Подключение электричества
Телевизоры
Стиральные машины
Радиодетали
Прочее

Номенклатура	Выручка (Оборот)
Строчный трансформатор Samsung	900,00
Строчный трансформатор GoldStar	400,00
Транзистор Philips 2N2369	14,00
Шланг резиновый	450,00
Кабель электрический	30,00
Ремонт отечественного телевизора	600,00
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00

Ссылка	Выручка (Оборот)
Материалы	
Услуги	
Строчный трансформатор Samsung	900,00
Строчный трансформатор GoldStar	400,00
Транзистор Philips 2N2369	14,00
Шланг резиновый	450,00
Кабель электрический	30,00
Диагностика	
Ремонт отечественного телевизора	600,00
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00
Телевизоры	
Стиральные машины	
Радиодетали	
Прочее	

Рис. 13.21. Связь записей таблиц в запросе

В результате описанных выше действий закладка *Связи* будет иметь следующий вид (рис. 13.22).

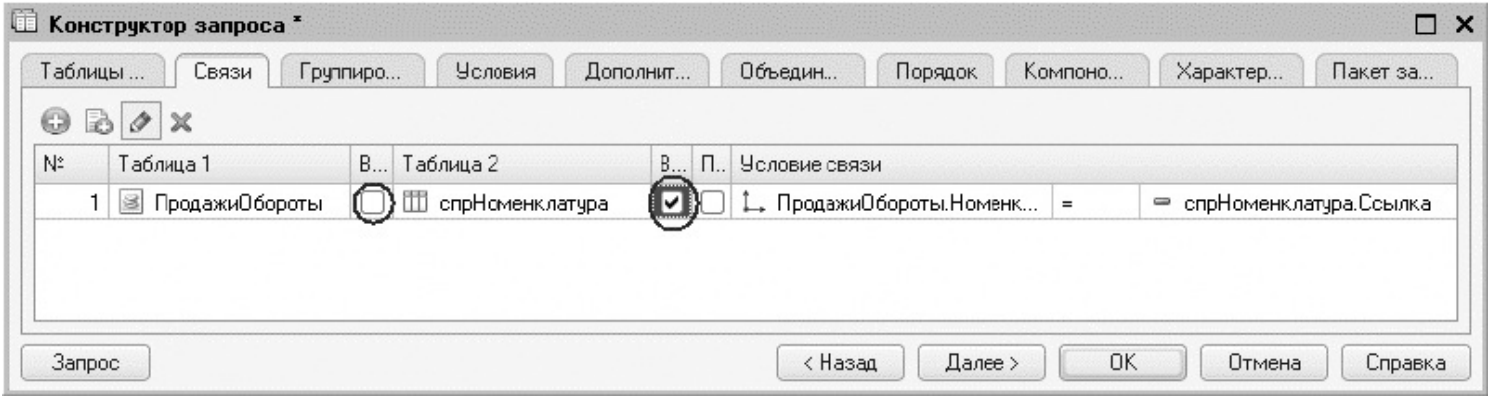


Рис. 13.22. Определение связи между таблицами

Условие отбора записей

Перейдем на закладку *Условия* и установим отбор, чтобы группы справочника *Номенклатура* не попадали в отчет.

Для этого раскроем таблицу *спрНоменклатура*, перетащим мышью поле *ЭтоГруппа* в список условий, установим флажок *Произвольное* и напишем в поле *Условие* следующий текст (листинг 13.5).

Листинг 13.5. Условие запроса

```
спрНоменклатура.ЭтоГруппа = ЛОЖЬ
```

Тем самым мы указали, что из базы данных нужно выбрать только те записи справочника *Номенклатура*, которые не являются группами.

Работу этого условия можно проиллюстрировать на следующем примере. Слева – исходная таблица справочника *Номенклатура*, а справа – записи, которые будут выбраны из этой таблицы (рис. 13.23).

Ссылка	Выручка (Оборот)
Материалы	
Услуги	
Строчный трансформатор Samsung	900,00
Строчный трансформатор GoldStar	400,00
Транзистор Philips 2N2369	14,00
Шланг резиновый	450,00
Кабель электрический	30,00
Диагностика	
Ремонт отечественного телевизора	600,00
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00
Телевизоры	
Стиральные машины	
Радиодетали	
Прочее	

Ссылка	Выручка (Оборот)
Строчный трансформатор Samsung	900,00
Строчный трансформатор GoldStar	400,00
Транзистор Philips 2N2369	14,00
Шланг резиновый	450,00
Кабель электрический	30,00
Диагностика	
Ремонт отечественного телевизора	600,00
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00

Рис. 13.23. Отбор записей номенклатуры в запросе

Вторым условием должно быть то, что выбранный элемент является услугой. Это *Простое условие.* Чтобы его создать, перетащим мышью поле *ВидНоменклатуры* в список условий.

Платформа автоматически сформирует условие, согласно которому вид номенклатуры должен быть равен значению параметра *ВидНоменклатуры*.

В дальнейшем перед выполнением запроса мы передадим в параметр *ВидНоменклатуры* значение перечисления – *Услуга*.

Работу этого условия тоже можно проиллюстрировать на примере. Слева – записи справочника *Номенклатура*, выбранные согласно первому условию. Справа – только те записи, которые являются услугами (рис. 13.24).

Ссылка	Выручка (Оборот)
Строчный трансформатор Samsung	900,00
Строчный трансформатор GoldStar	400,00
Транзистор Philips 2N2369	14,00
Шланг резиновый	450,00
Кабель электрический	30,00
Диагностика	
Ремонт отечественного телевизора	600,00
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00

Ссылка	Выручка (Оборот)
Диагностика	
Ремонт отечественного телевизора	600,00
Ремонт импортного телевизора	800,00
Подключение воды	800,00
Подключение электричества	800,00

Рис. 13.24. Отбор записей номенклатуры в запросе

В результате закладка *Условия* примет вид (рис. 13.25).

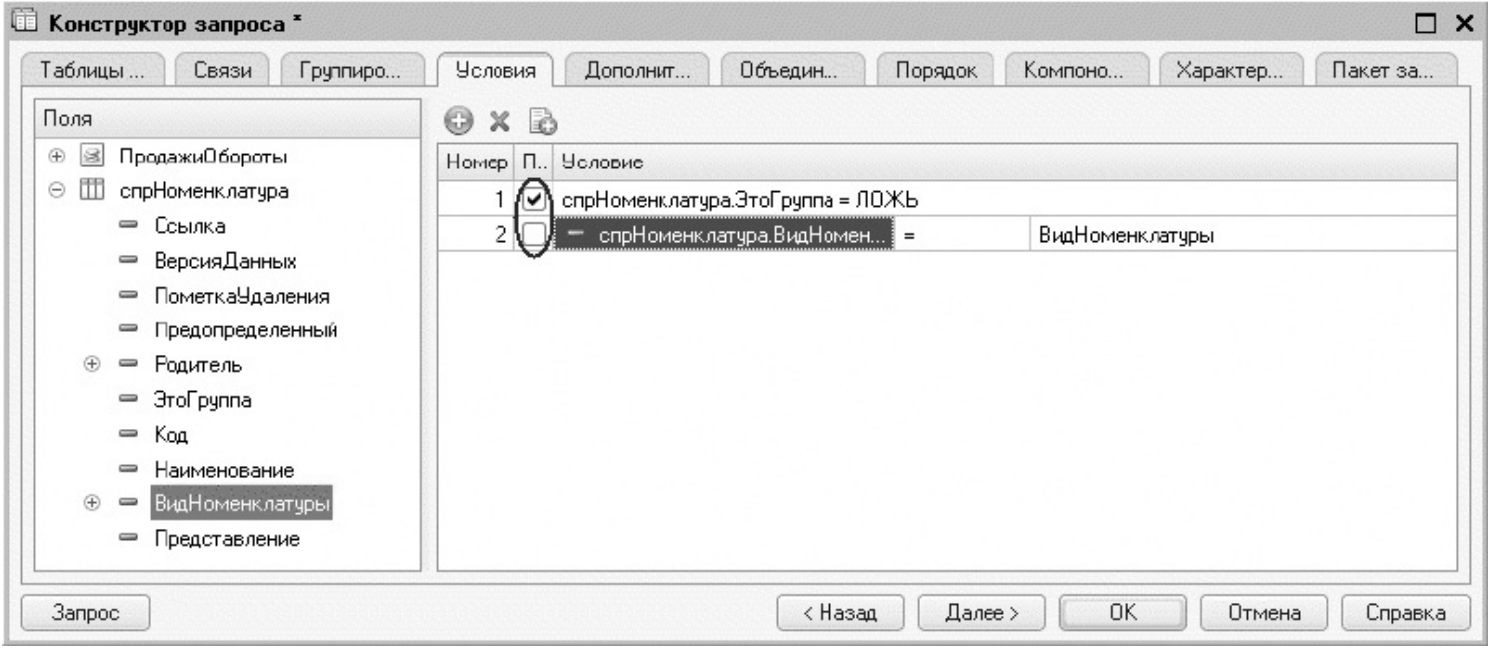


Рис. 13.25. Создание условия запроса

СОВЕТ

Отбор можно применять и в самом запросе, и в настройках отчета. То же касается и сортировки и группировки. Отбор лучше применять в запросе, если записи, не удовлетворяющие условию запроса, наверняка не понадобятся в отчете. Сортировку и группировку лучше применять уже в настройках отчета,

чтобы сделать его более гибким.

Псевдонимы полей

Перейдем на закладку *Объединения/Псевдонимы* и укажем, что представление элемента справочника (поле *Ссылка*) будет иметь псевдоним *Услуга*, а поле регистра будет иметь псевдоним *Выручка* (рис. 13.26).

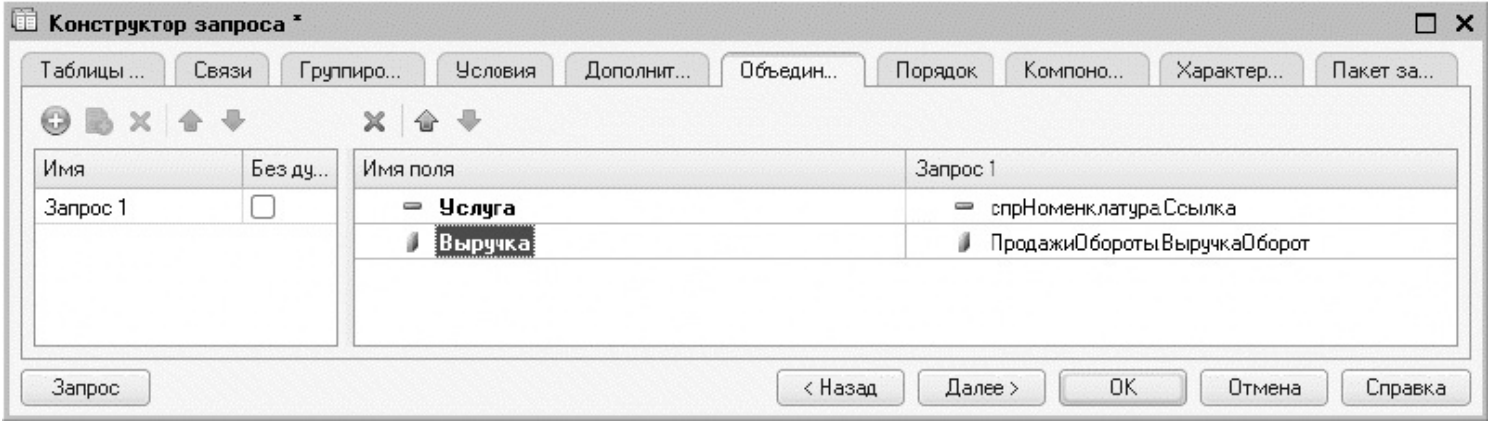


Рис. 13.26. Установка псевдонимов полей запроса

Порядок записей

Перейдем на закладку *Порядок* и укажем, что результат запроса должен быть отсортирован по убыванию значения поля *Выручка* (рис. 13.27).

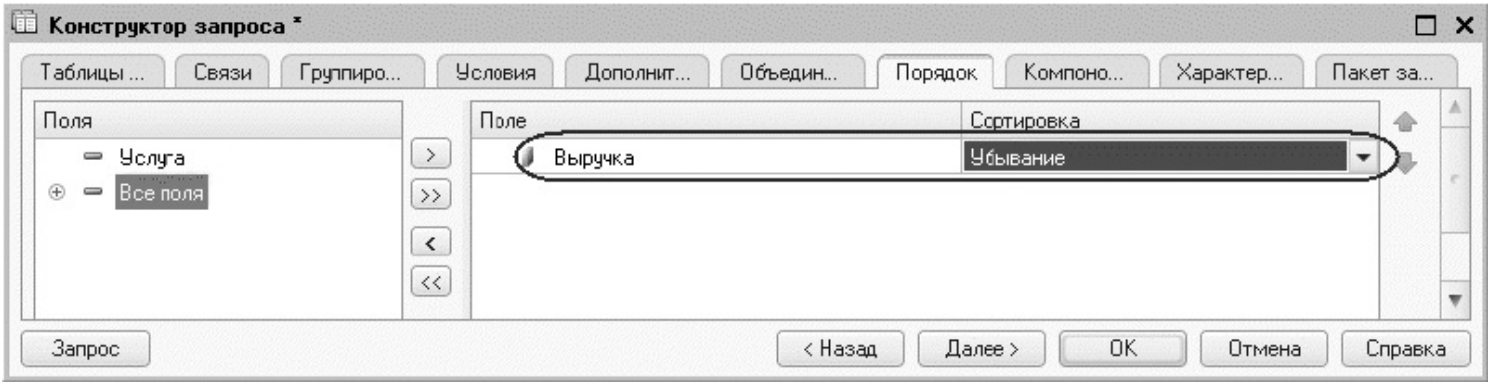


Рис. 13.27. Порядок записей запроса

Создание запроса закончено, нажмем кнопку *OK*. Вернемся в конструктор схемы компоновки данных.

Анализ текста запроса

Текст запроса, сформированный платформой, примет вид (листинг 13.6).

Листинг 13.6. Текст запроса

```
ВЫБРАТЬ
  СпрНоменклатура.Ссылка КАК Услуга,
  ПродажиОбороты.ВыручкаОборот КАК Выручка
ИЗ
  Справочник.Номенклатура КАК СпрНоменклатура
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
    ПО ПродажиОбороты.Номенклатура = СпрНоменклатура.Ссылка
ГДЕ
```

```
  СпрНоменклатура.ЭтоГруппа = ЛОЖЬ
  И СпрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры
УПОРЯДОЧИТЬ ПО
  Выручка УБЫВ
```

Сначала, как обычно, идет часть описания запроса, и в ней есть новые для нас конструкции.

При описании источников запроса (после ключевого слова *ИЗ*) использована возможность определения нескольких источников запроса (листинг 13.7).

Листинг 13.7. Определение нескольких источников запроса

```
ИЗ
  Справочник.Номенклатура КАК СпрНоменклатура
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
    ПО ПродажиОбороты.Номенклатура = СпрНоменклатура.Ссылка
```

В данном случае выбираются записи из двух источников: *СпрНоменклатура* и *ПродажиОбороты*, причем ключевым предложением *ЛЕВОЕ СОЕДИНЕНИЕ … ПО* описан способ, которым будут скомбинированы между собой записи этих двух источников.

ЛЕВОЕ СОЕДИНЕНИЕ означает, что в результат запроса нужно включить

комбинации записей из обоих источников, которые соответствуют указанному после ключевого слова *ПО* условию. Кроме этого, в результат запроса нужно включить еще и записи из первого (указанного слева от слова *СОЕДИНЕНИЕ*) источника, для которых не найдено соответствующих условию записей из второго источника.

Продолжим рассматривать текст запроса. В части описания запроса есть еще одна новая для нас конструкция – задание условий отбора данных из исходных таблиц (листинг 13.8).

Листинг 13.8. Задание условий отбора

```
ГДЕ
  СпрНоменклатура.ЭтоГруппа = ЛОЖЬ
  И СпрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры
```

Условию отбора всегда предшествует ключевое слово *ГДЕ*. После него описывается само условие. Обратите внимание, что поля исходных таблиц, на которые накладывается условие, могут и не входить в список выборки (как в нашем случае). Кроме того, в нашем условии использован параметр запроса *ВидНоменклатуры* (перед именем параметра указывается символ *&* – амперсанд).

Теперь, когда мы закончили знакомиться с текстом запроса, продолжим

формирование нашей схемы компоновки данных.

Ресурсы

В нашем отчете мы хотим видеть итоговые значения выручки для каждой услуги. Для этого нам нужно определить поля ресурсов отчета.

Под *ресурсами* в системе компоновки данных подразумеваются поля, значения которых рассчитываются на основании детальных записей, входящих в группировку. По сути ресурсы являются групповыми или общими итогами отчета.

Итоговые данные формируются на закладке *Ресурсы*.

Перейдем на эту закладку и нажмем кнопку >>, чтобы конструктор выбрал все доступные ресурсы, по которым можно вычислять итоги. В нашем случае это единственный ресурс *Выручка*.

Платформа автоматически предложить рассчитывать сумму значений этого поля, что нам и нужно (рис. 13.28).

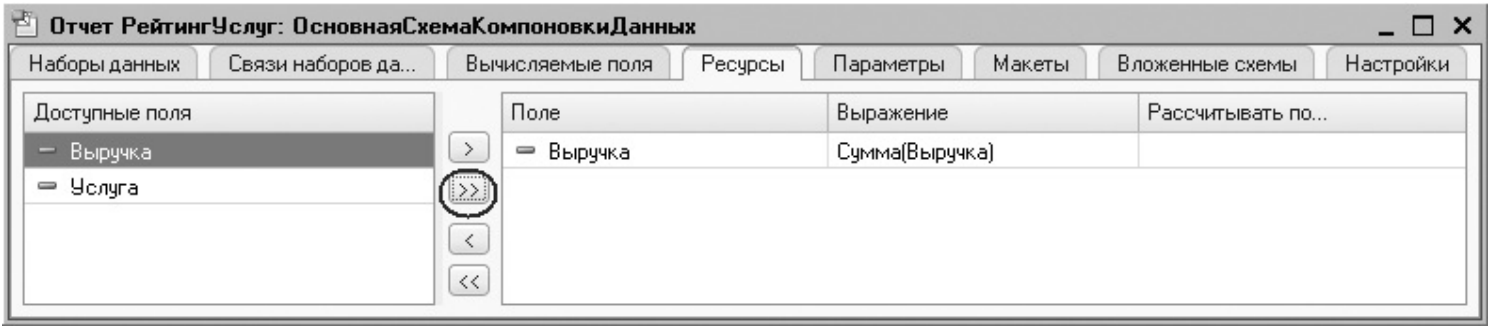


Рис. 13.28. Ресурсы схемы компоновки данных

Параметры

Пользователя, как правило, интересуют данные о хозяйственной деятельности за определенный период. Поэтому практически в любом отчете используются параметры, задающие начало и конец отчетного периода.

Параметры отчета задают условия отбора записей в отчет. В схеме компоновки данных параметры отчета задаются на закладке *Параметры* (рис. 13.29).

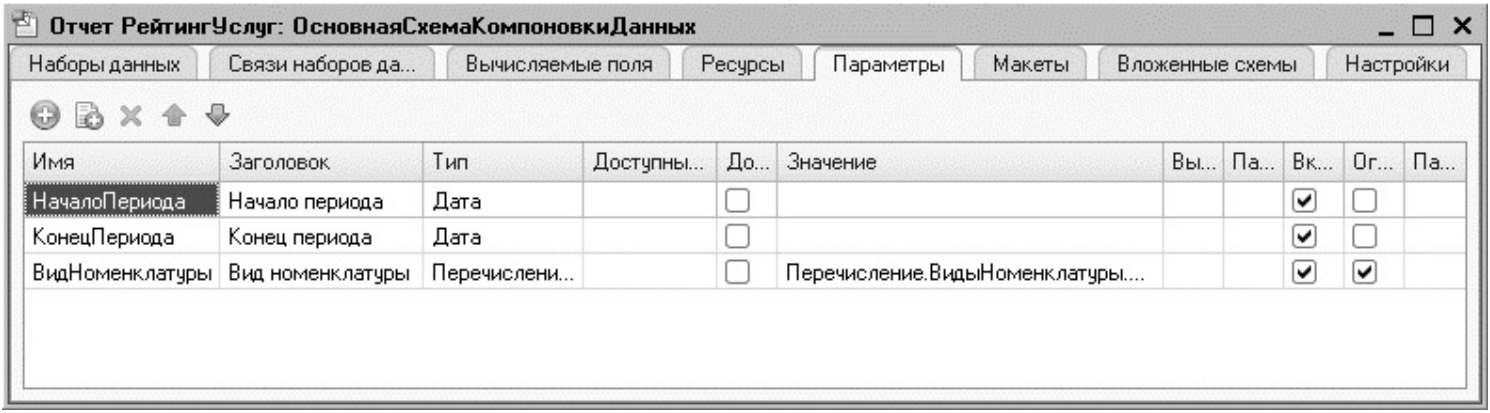


Рис. 13.29. Параметры компоновки данных

На этой закладке мы увидим три параметра: *НачалоПериода*, *КонецПериода* и *ВидНоменклатуры*. Вы можете спросить: почему параметра три, хотя в запросе мы задавали всего один – *ВидНоменклатуры*?

Все дело в том, что система компоновки данных самостоятельно анализирует текст запроса и помимо тех параметров, которые указаны в нем в явном виде (*ВидНоменклатуры*), предоставляет возможность настроить также и параметры виртуальных таблиц, которые участвуют в запросе.

Такими параметрами являются *НачалоПериода* и *КонецПериода*. Это первые два параметра виртуальной таблицы *РегистрНакопления.Продажи.Обороты*, которую мы использовали в запросе, в левом соединении.

Если в конструкторе запроса выделить в списке таблиц эту таблицу и нажать кнопку *Параметры виртуальной таблицы*, то появится диалог, где мы увидим параметры *НачалоПериода* и *КонецПериода* (рис. 13.30).

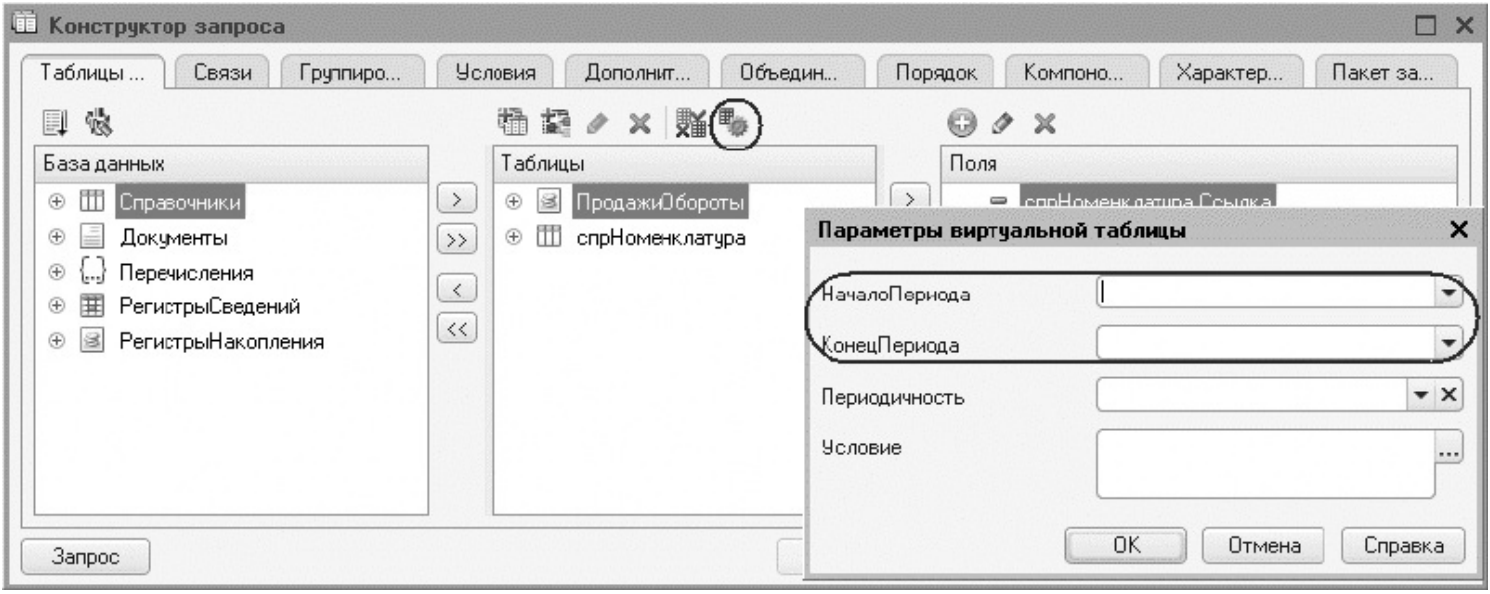


Рис. 13.30. Параметры виртуальной таблицы

Первым параметром передается начало периода расчета итогов, вторым – конец периода. В результате исходная таблица будет содержать только обороты, рассчитанные в переданном периоде.

Здесь всегда следует помнить, что если мы передаем в качестве этих

параметров дату (а в нашем случае так и будет), то дата содержит и время с точностью до секунды.

Допустим, заранее известно, что пользователя не будут интересовать результаты работы отчета в периодах, указанных с точностью до секунды. В этом случае следует учесть две особенности.

Во-первых, пользователя нужно избавить от необходимости указывать время при вводе даты периода, за который формируется отчет.

Для этого мы изменим существующее описание типа для параметра *НачалоПериода*.

Вернемся на закладку *Параметры схемы компоновки данных* и дважды щелкнем в ячейке *Тип*, соответствующей параметру *НачалоПериода*.

Затем нажмем кнопку выбора  и в нижней части окна редактирования типа данных установим *Состав даты* в значение *Дата*. Нажмем *ОК* (рис. 13.31).

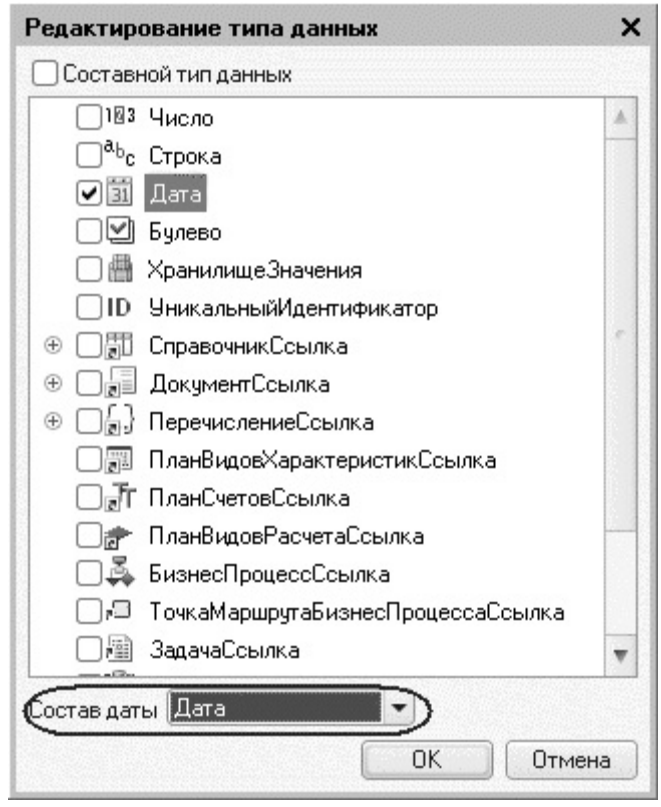


Рис. 13.31. Редактирование состава даты

Вторая особенность заключается в том, что по умолчанию время в дате установлено 00:00:00. Поэтому если пользователь задаст период отчета с 01.07.2009 по 14.07.2009, итоги регистра будут рассчитаны с начала дня 01.07.2009 00:00:00 по начало дня 14.07.2009 00:00:00. Таким образом, данные

за 14-е число, отличные от начала дня, в расчет не войдут, что сильно удивит пользователя.

Для того чтобы исключить эту ситуацию, мы добавим еще один параметр, в который пользователь будет вводить дату окончания. А значение параметра *КонецПериода* будем рассчитывать автоматически таким образом, чтобы оно указывало на конец дня даты, введенной пользователем.

Поэтому для параметра *КонецПериода* установим флажок *Ограничение доступности* (рис. 13.32).

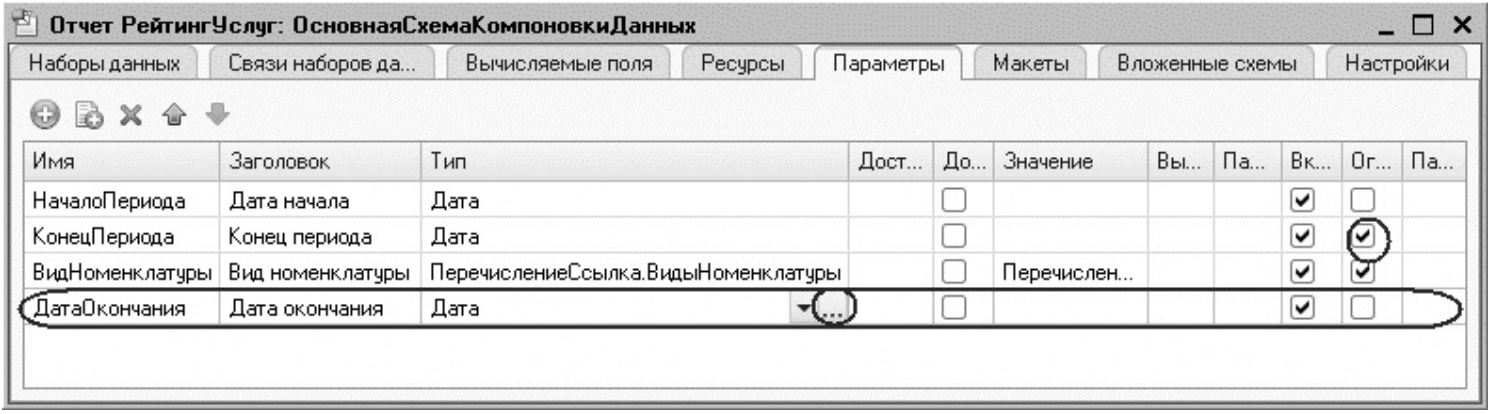


Рис. 13.32. Добавление параметра «ДатаОкончания»

Если этот флажок не установлен, то параметр будет доступен для настройки пользователем. Если же установить этот флажок, то пользователь не увидит

этот параметр.

Затем с помощью кнопки *Добавить* в командной панели добавим новый параметр с именем *ДатаОкончания* (см. рис. 13.32). Для этого параметра платформа автоматически сформирует заголовок – *Дата окончания*. Оставим его без изменений.

Зададим тип значения параметра – *Дата*. При этом, как и для параметра *Начало Периода,* укажем состав даты – *Дата*. А также для параметра *НачалоПериода* зададим заголовок, который будет отображаться пользователю, – *Дата начала*.

Обратите внимание, что по умолчанию добавленный нами параметр доступен для пользователя (ограничение доступности в колонке снято). Нас это вполне устраивает.

Перейдем к параметру *КонецПериода*. Для него мы установили флажок *Ограничение доступности*, поскольку значение этого параметра мы собираемся вычислять на основании значения, установленного пользователем для параметра *ДатаОкончания*.

Чтобы задать формулу, по которой будет вычисляться значение параметра *КонецПериода*, воспользуемся языком выражений системы компоновки данных.

В нем есть функция *КонецПериода()*, которая позволяет получить дату, соответствующую концу какого-либо периода, например, указанного дня.

В ячейке *Выражение* зададим для параметра *КонецПериода* следующее выражение (листинг 13.9).

Листинг 13.9. Выражение для расчета значения параметра «КонецПериода»

```
КонецПериода(&ДатаОкончания, "День")
```

ПРИМЕЧАНИЕ

Подробное описание языка выражений системы компоновки данных содержится во встроенной справке системы в разделе ***Справка > Содержание справки > Встроенный язык > Общие объекты > Система компоновки данных > Язык выражений системы компоновки данных***.

В результате перечисленных действий параметры компоновки будут иметь следующий вид (рис. 13.33).

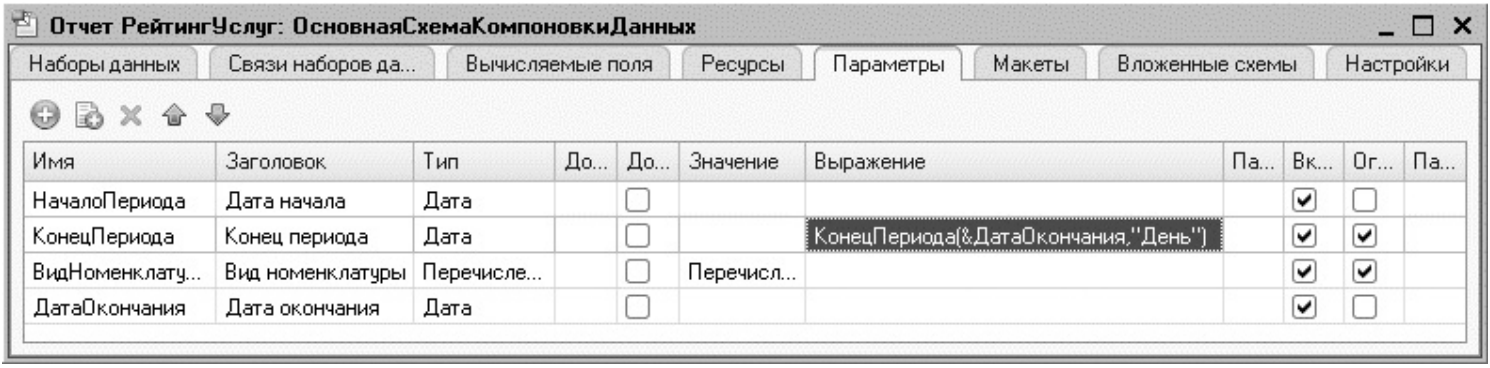


Рис. 13.33. Параметры системы компоновки

И в заключение настроим параметр *ВидНоменклатуры*.

Поскольку отчет должен отображать выручку, полученную только от реализации услуг, значение параметра *ВидНоменклатуры* пользователь изменять не должен. Оно должно быть задано непосредственно в схеме компоновки как *Перечисление.ВидыНоменклатуры.Услуга*.

Флажок ограничения использования у параметра *ВидНоменклатуры* платформа установила по умолчанию, поэтому нам остается только указать нужное значение перечисления *ВидыНоменклатуры* в ячейке *Значение*, соответствующей параметру *ВидНоменклатуры*.

Воспользуемся кнопкой выбора ... и выберем это значение из списка

перечисления видов номенклатуры – *Услуга* (рис. 13.34).

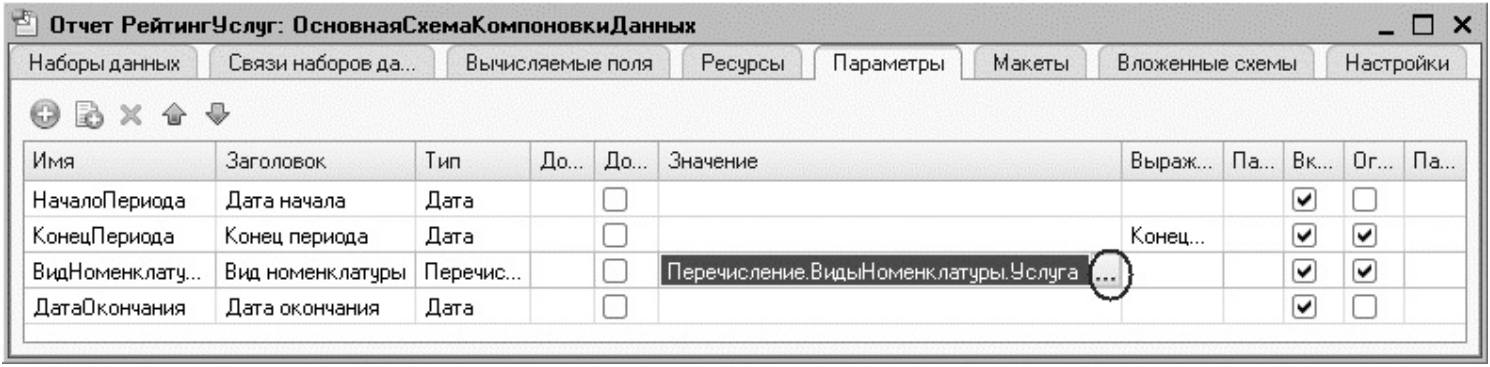


Рис. 13.34. Установка значения параметра «ВидНоменклатуры»

Настройки

Перейдем к формированию структуры отчета. На закладке *Настройки* добавим группировку и снова не укажем поле группировки. На закладке *Выбранные поля* укажем поля *Услуга* и *Выручка* (рис. 13.35).

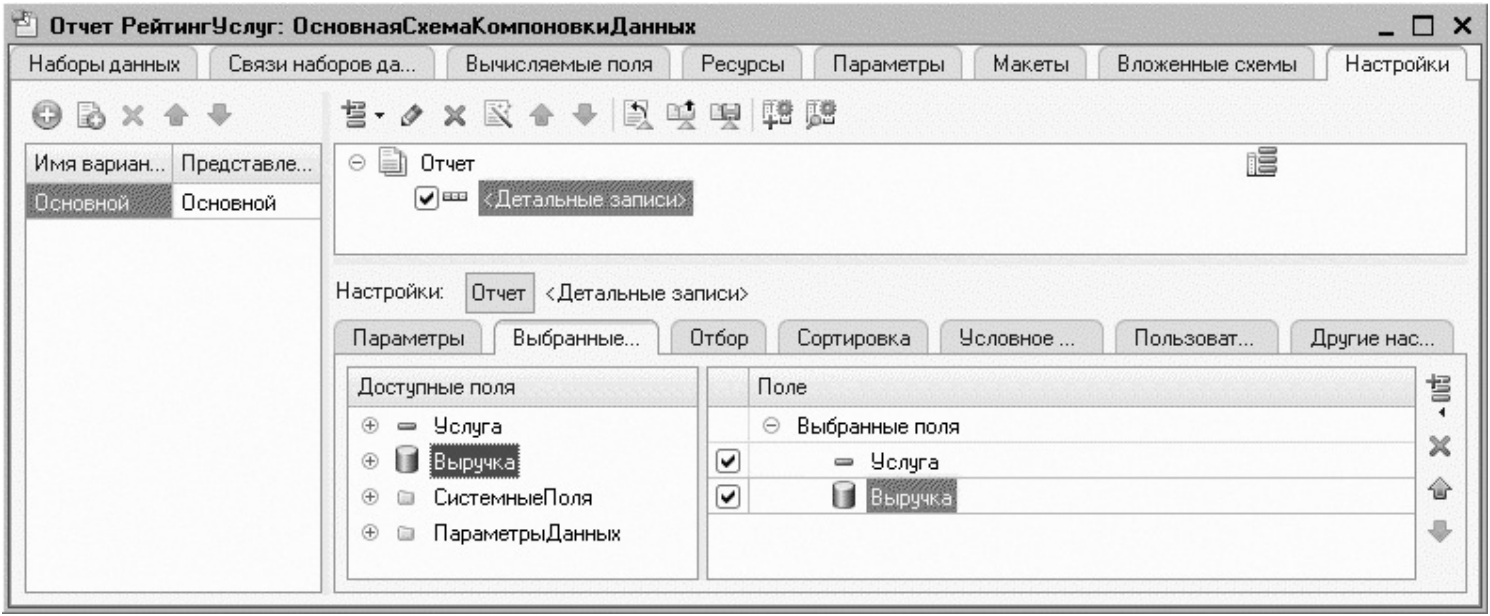


Рис. 13.35. Структура отчета «РейтингУслуг»

Затем перейдем на закладку *Другие настройки* и зададим заголовок отчета – *Рейтинг услуг* (рис. 13.36).

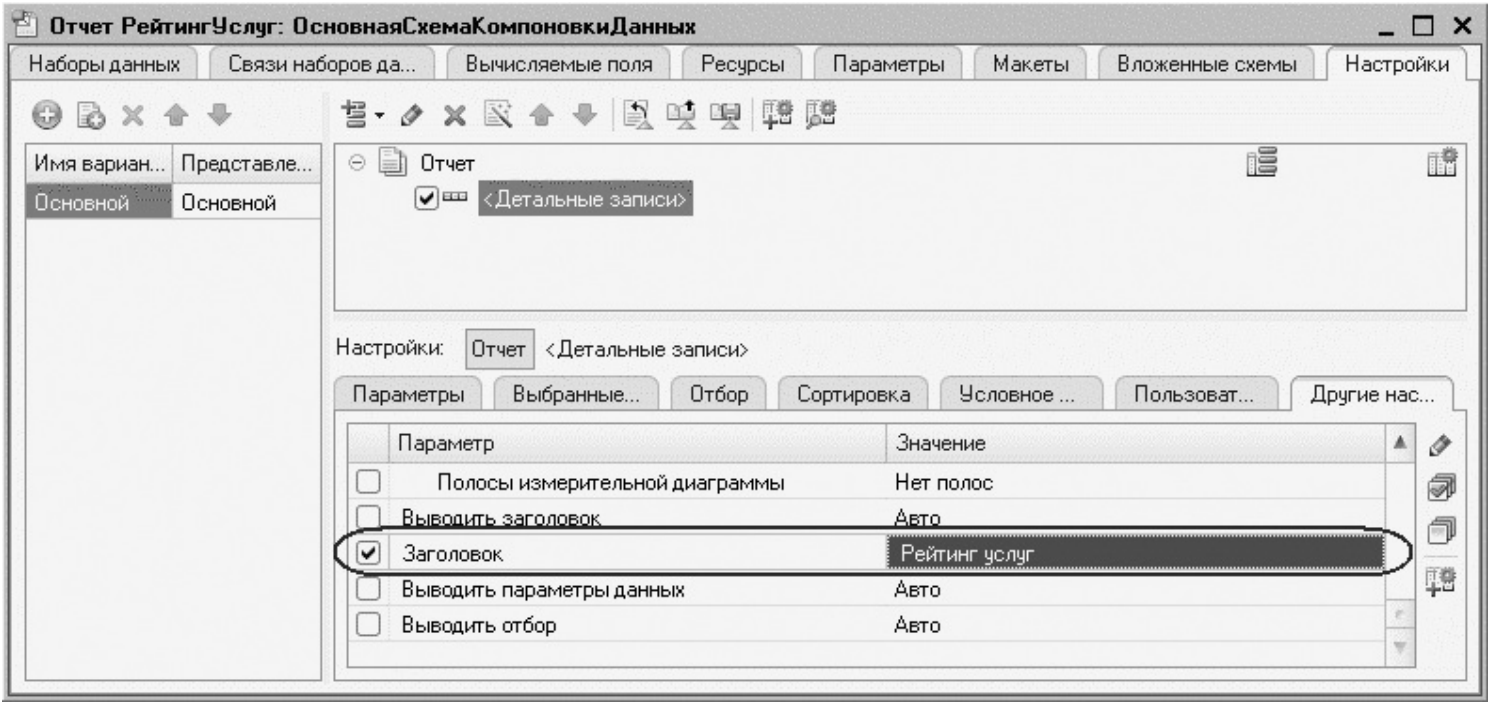


Рис. 13.36. Установка заголовка отчета

ПРИМЕЧАНИЕ

При изменении параметров настроек, которые предполагают выбор некоторого значения, нужно выделить двойным щелчком поле ***Значение****, и, нажав кнопку выбора* ..., *выбрать из списка значений нужный вариант. При этом флажок использования значения появится автоматически. Этот флажок можно также снять и установить вручную.*

Быстрые пользовательские настройки

В заключение мы должны предоставить пользователю возможность задавать отчетный период перед формированием отчета. То есть параметры *Дата начала* и *Дата окончания* должны быть включены в состав пользовательских настроек.

Причем поскольку задавать отчетный период требуется практически всегда, эти настройки должны находиться непосредственно в форме отчета.

На закладке *Параметры* мы видим параметры, для которых мы установили возможность их изменения пользователем, то есть сняли флажок *Ограничение доступности* (рис. 13.37).

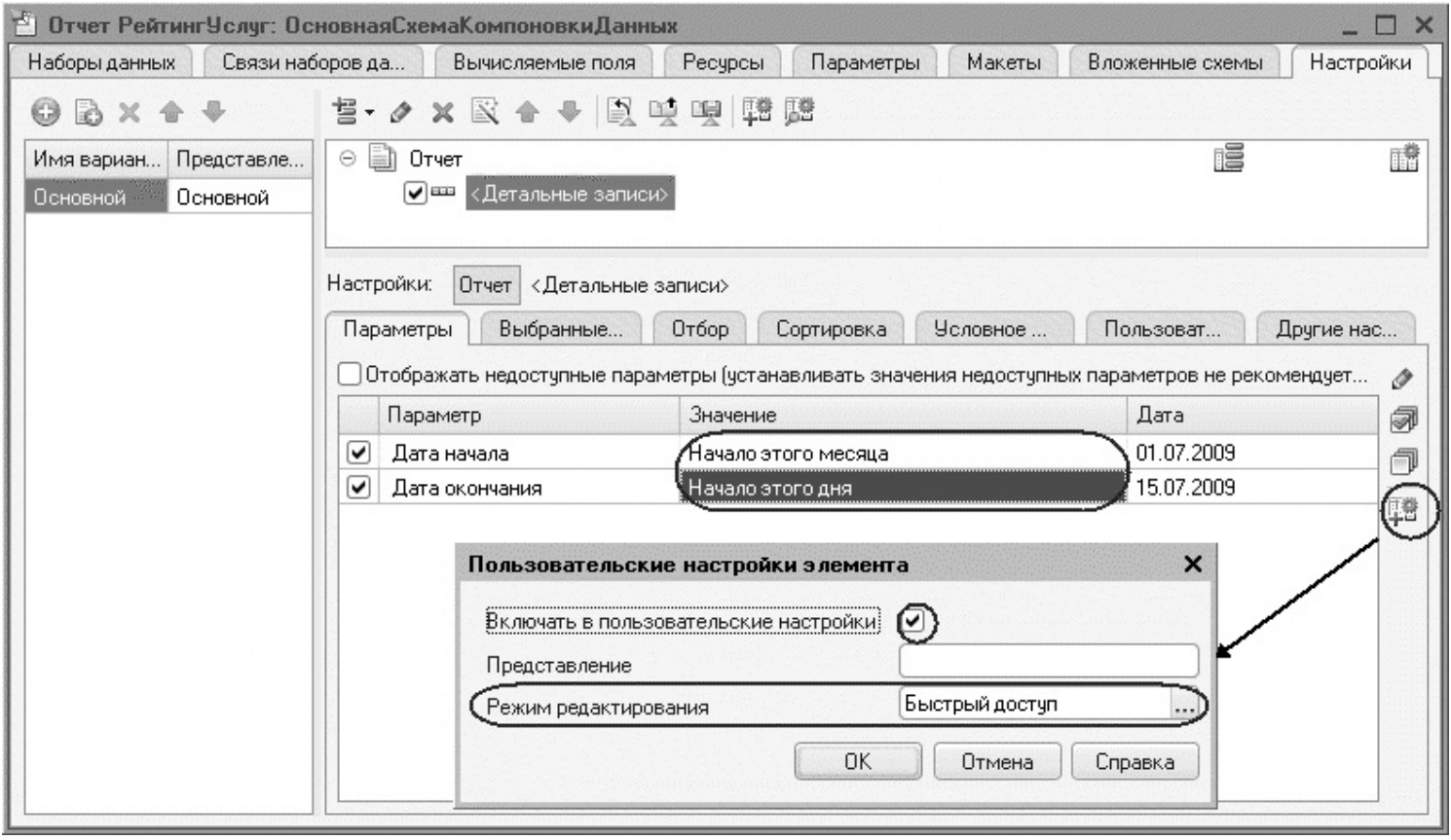


Рис. 13.37. Определение пользовательских настроек

Выделим по очереди каждый из параметров и нажмем кнопку *Свойства элемента пользовательских настроек*, расположенную в правом нижнем углу окна настроек.

Установим флажок *Включать в пользовательские настройки* и оставим предложенное по умолчанию для свойства *Режим редактирования* значение *Быстрый доступ*.

Поясним, что флажок *Включать в пользовательские настройки* означает, что эта настройка будет доступна пользователю в отдельном окне (2) при нажатии кнопки *Настройка* (то есть такая настройка, которой он может пользоваться, но не очень часто, рис. 13.38).

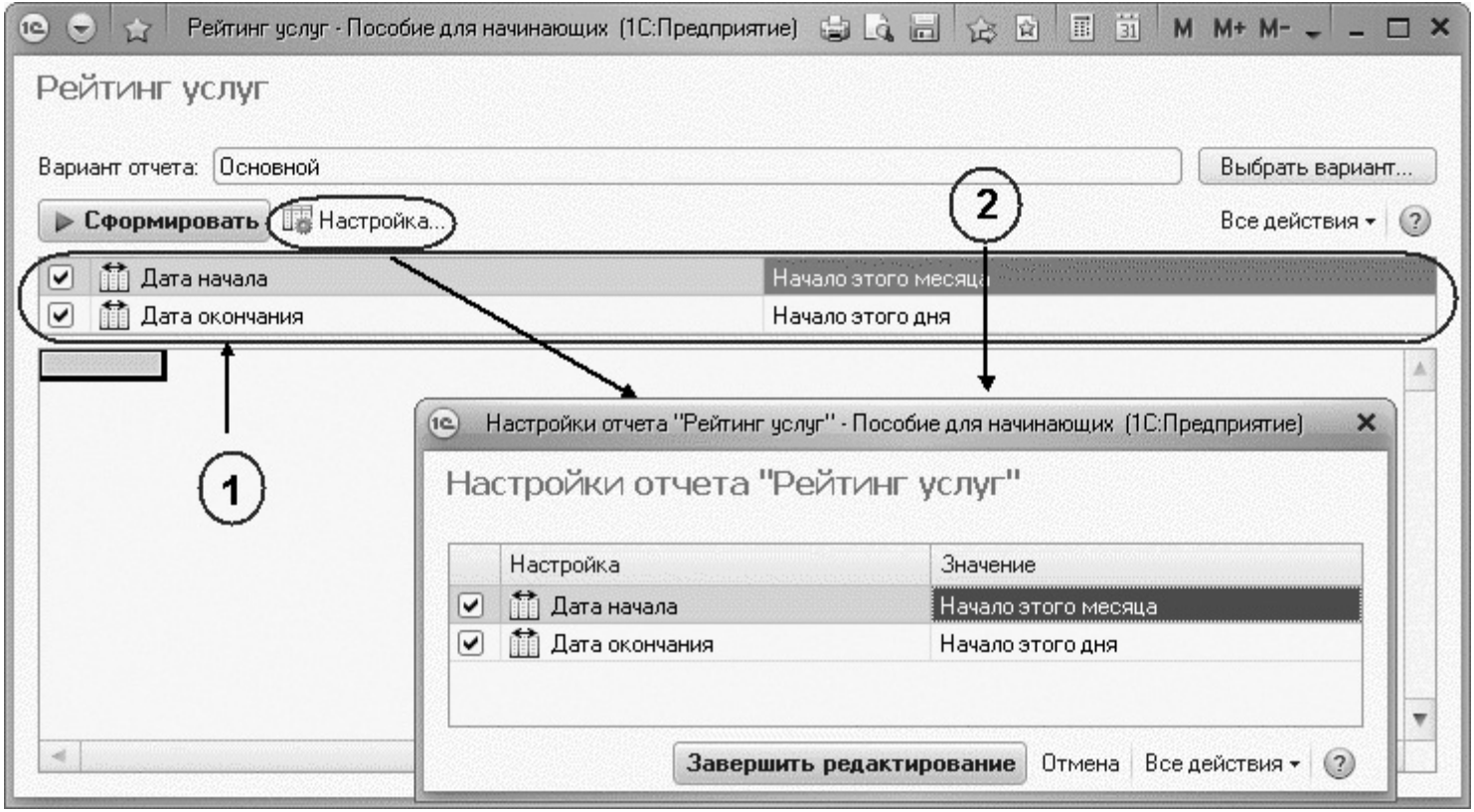


Рис. 13.38. Быстрые (1) и обычные (2) пользовательские настройки

А режим редактирования, установленный в значение *Быстрый доступ,* означает, что эта настройка также будет автоматически отображаться непосредственно в отчетной форме (1). Это *быстрая пользовательская настройка* – такая настройка, которая нужна пользователю постоянно, чуть ли

не при каждом запуске отчета. Поэтому она всегда на виду.

Кроме того, чтобы улучшить интерфейс пользователя, зададим для параметров *Дата начала* и *Дата окончания* в качестве начальных значений соответственно *Начало этого месяца* и *Начало этого дня* (см. рис. 13.37).

Таким образом, при выполнении отчета даты начала и окончания отчетного периода будут динамически меняться и показывать период с начала текущего месяца по сегодняшнее число, и пользователю, возможно, не придется менять их вручную.

В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации Отчет *РейтингУслуг* перейдем на закладку *Подсистемы*. Отметим в списке подсистем конфигурации подсистемы *Оказание услуг* и *Бухгалтерия*.

Таким образом, ссылка на наш отчет автоматически попадет в панель действий этих подсистем (рис. 13.39).

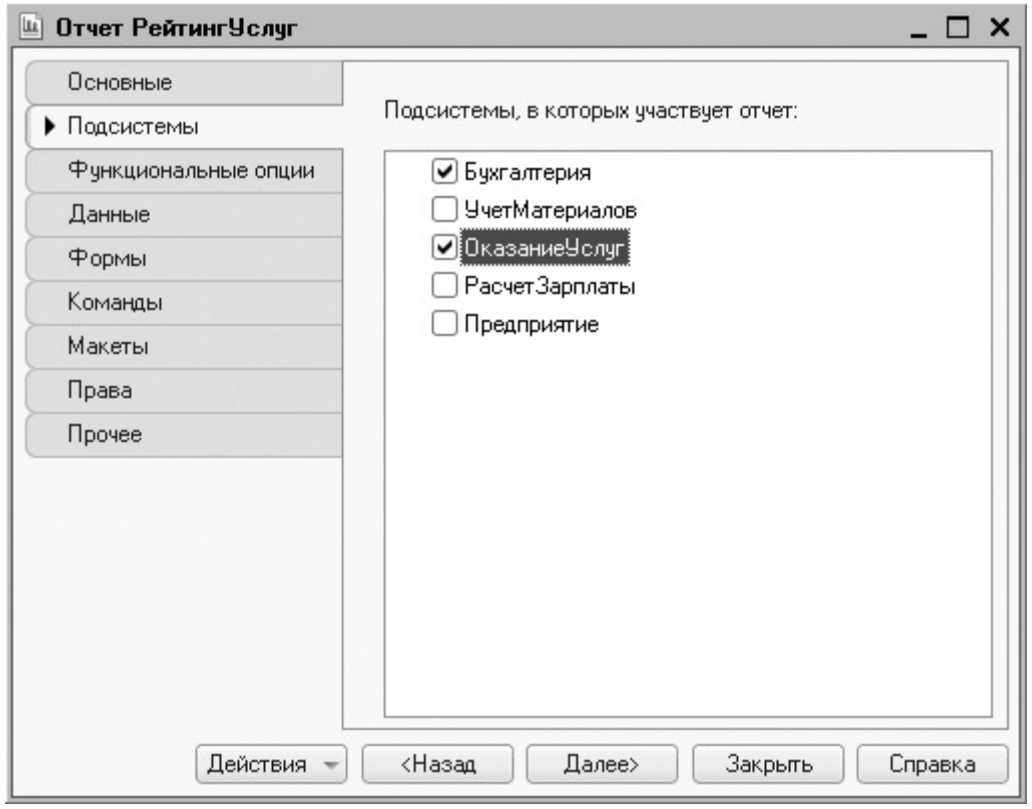


Рис. 13.39. Подсистемы, в которых отображается отчет

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает отчет.

В открывшемся окне «1С:Предприятия» мы видим, что в панели действий разделов *Оказание услуг* и *Бухгалтерия* в группе команд для выполнения отчетов появилась команда для формирования отчета *Рейтинг услуг* (рис. 13.40).

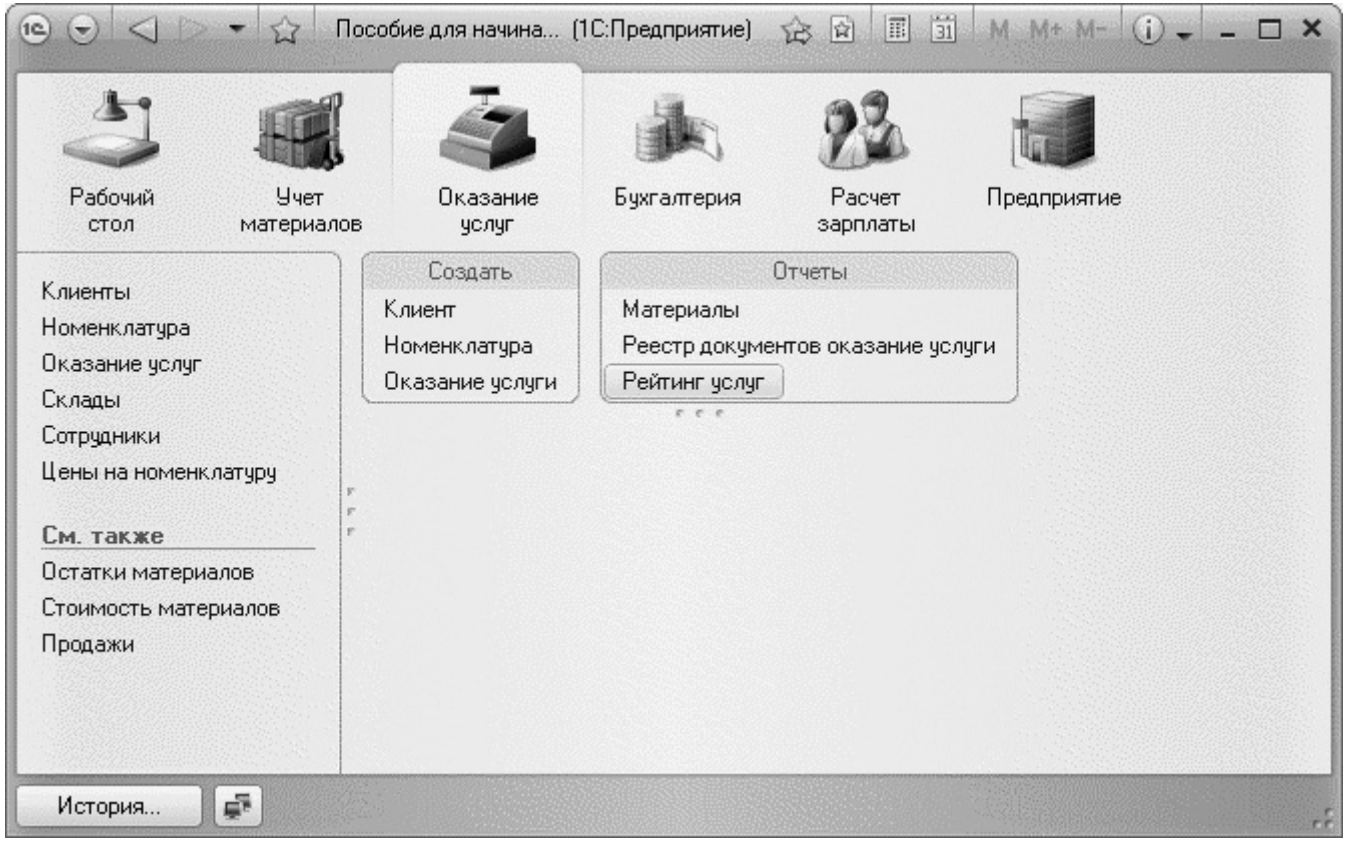


Рис. 13.40. Команда для формирования отчета

Выполним эту команду.

Перед нами откроется форма отчета, автоматически сформированная системой.

В окне отчета мы видим параметры, определяющие отчетный период. Он по умолчанию задан – с начала месяца по сегодняшнее число. Но можно при желании изменить его, воспользовавшись кнопкой календаря.

Нажмем кнопку *Сформировать*. Результат будет выглядеть следующим образом (рис. 13.41).

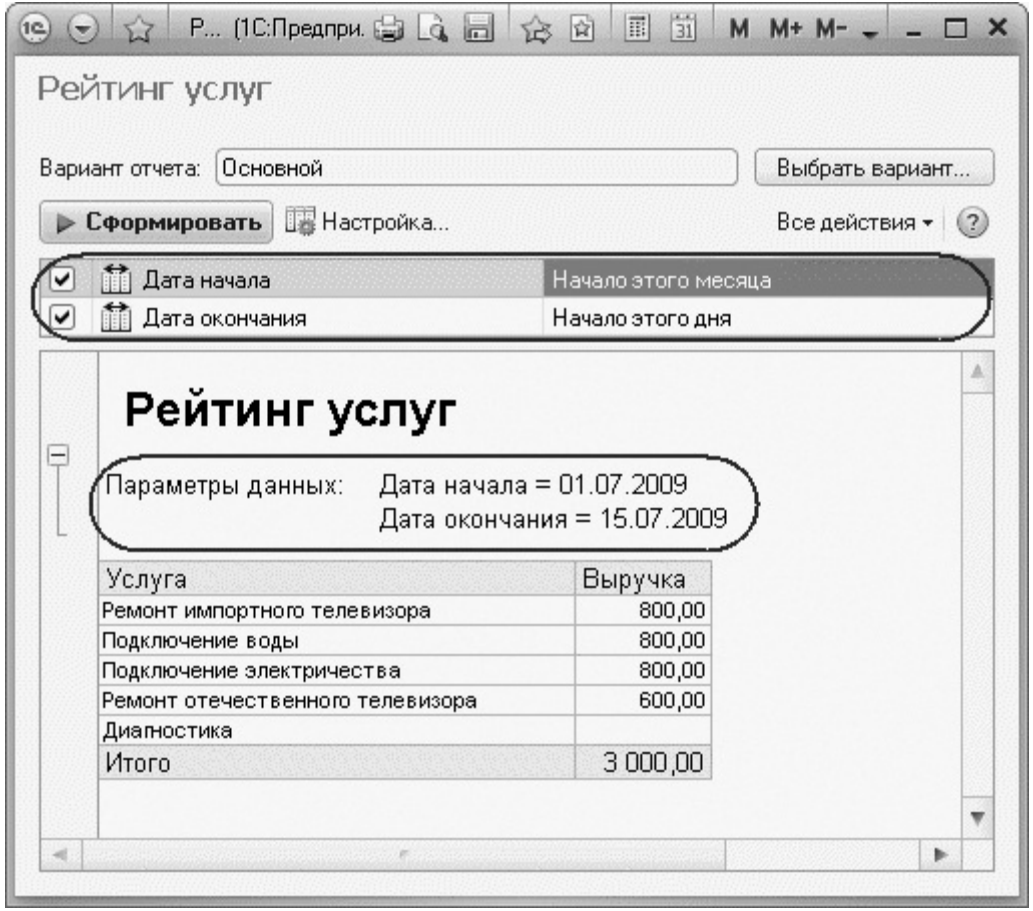


Рис. 13.41. Результат выполнения отчета

То есть выручка от услуг из всех трех введенных нами документов *Оказание услуги* попала в отчет.

Заметьте, что вверху окна результата отчета выводится заданный нами заголовок и параметры, определяющие отчетный период.

Теперь изменим дату окончания на *13.07.2009*. Данные за 13 июля из документа *Оказание услуги № 1* (то есть последнее число отчетного периода) попадают в отчет (рис. 13.42).

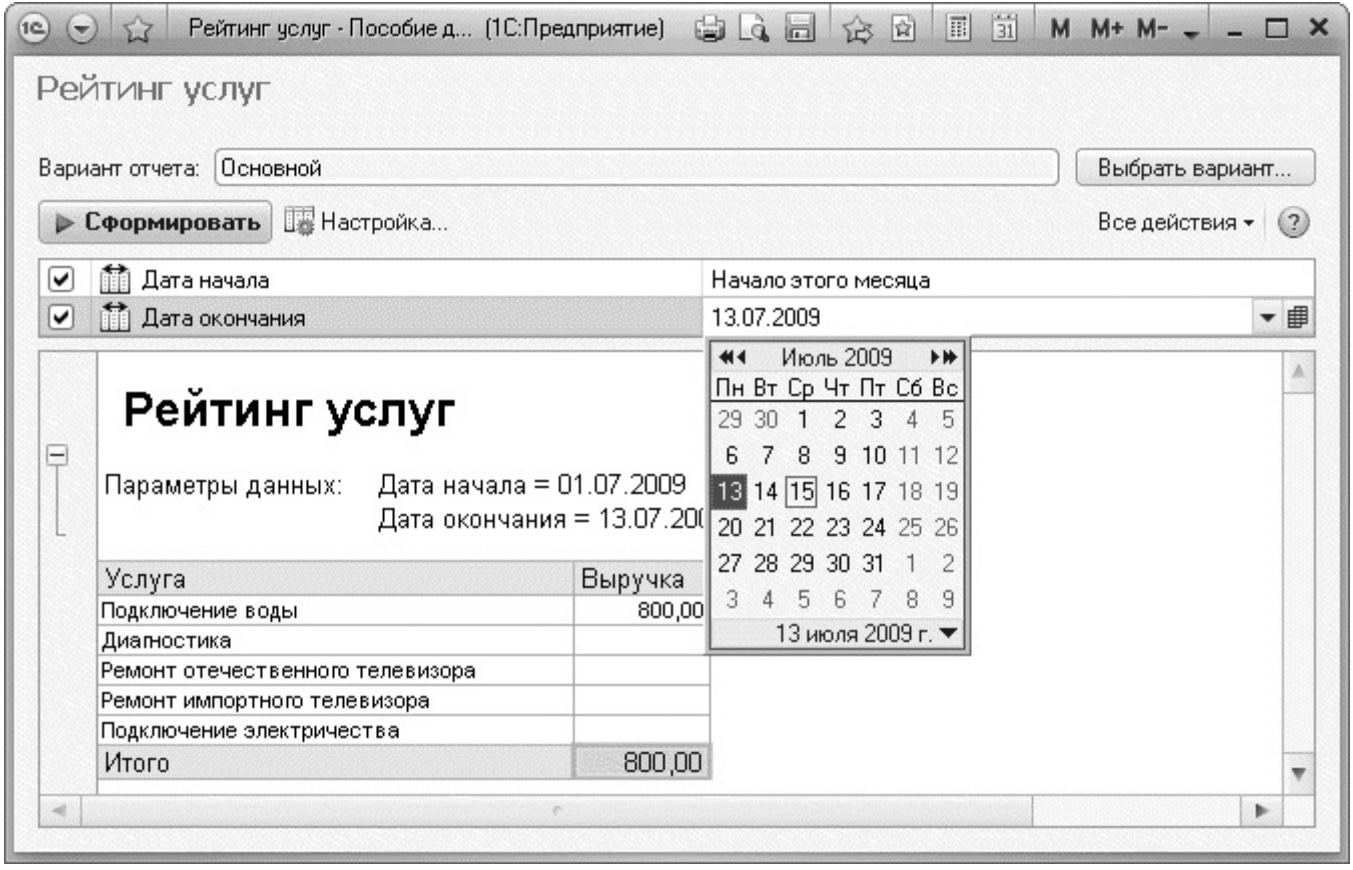


Рис. 13.42. Результат выполнения отчета

Причем поскольку в запросе данных для отчета таблица номенклатуры связана левым соединением с таблицей регистра продаж, то услуги, для которых нет данных о продажах, все равно показаны в отчете.

Настройки в конфигураторе и в режиме «1С:Предприятие»

Теперь на примере этого отчета покажем создание и использование других настроек отчета – *Условное оформление* и *Отбор*.

В процессе создания этих настроек мы будем выполнять некоторые действия в конфигураторе и затем переходить в режим *1С:Предприятие*, чтобы посмотреть, что получилось.

На самом деле все то же самое, что мы будем настраивать в режиме Конфигуратор, можно настроить и в режиме *1С:Предприятие* по команде *Все действия* > *Изменить вариант*. Разница вот в чем. Те настройки, которые мы будем делать сейчас в конфигураторе, называются *стандартными настройками* и будут сохранены в самой схеме компоновки данных, то есть будут являться частью конфигурации. Это означает, что любой пользователь конфигурации будет видеть отчет именно в таком виде, как мы его настроили в конфигураторе.

Все то же самое можно настроить и в режиме *1С:Предприятие*, но эта настройка уже не будет являться частью конфигурации и будет доступна только одному конкретному пользователю конкретной информационной базы.

Примечание

В конфигурации может быть разработан механизм, позволяющий обмениваться настройками между различными пользователями. Однако это непростая задача, и в рамках данной книги мы ее рассматривать не будем, но в принципе такая возможность существует.

Возможность изменения варианта отчета в режиме *1С:Предприятие* не предназначена для рядового пользователя (для него – быстрые настройки и пользовательские настройки). Она предназначена для разработчика, осуществляющего внедрение, или для администратора, или для очень опытного пользователя.

Настройки, сделанные в режиме *1С:Предприятие,* естественно «перекрывают» стандартные настройки. И если пользователь настолько все перестроил в отчете так, что его не узнать, то всегда можно вернуться к стандартным настройкам по команде *Все действия > Стандартные настройки*.

Итак, сейчас мы хотим настроить отчет для любых пользователей, которые будут им пользоваться, поэтому делаем это в конфигураторе.

Но если завтра главный бухгалтер попросит вас «сделать отчет красивым», вы сможете повторить все то же самое, не меняя конфигурацию и прямо у нее на глазах.

Условное оформление

В таком отчете, как *Рейтинг услуг*, было бы удобно выделять цветом записи отчета, содержащие услуги с наименьшей или с наибольшей выручкой, или еще по какому-либо условию.

В режиме Конфигуратор

Для этого вернемся в конфигуратор и откроем схему компоновки данных на закладке *Настройки*.

В нижней части окна перейдем на закладку *Условное оформление* и нажмем кнопку *Добавить*, расположенную в правом верхнем углу окна настроек (рис. 13.43).

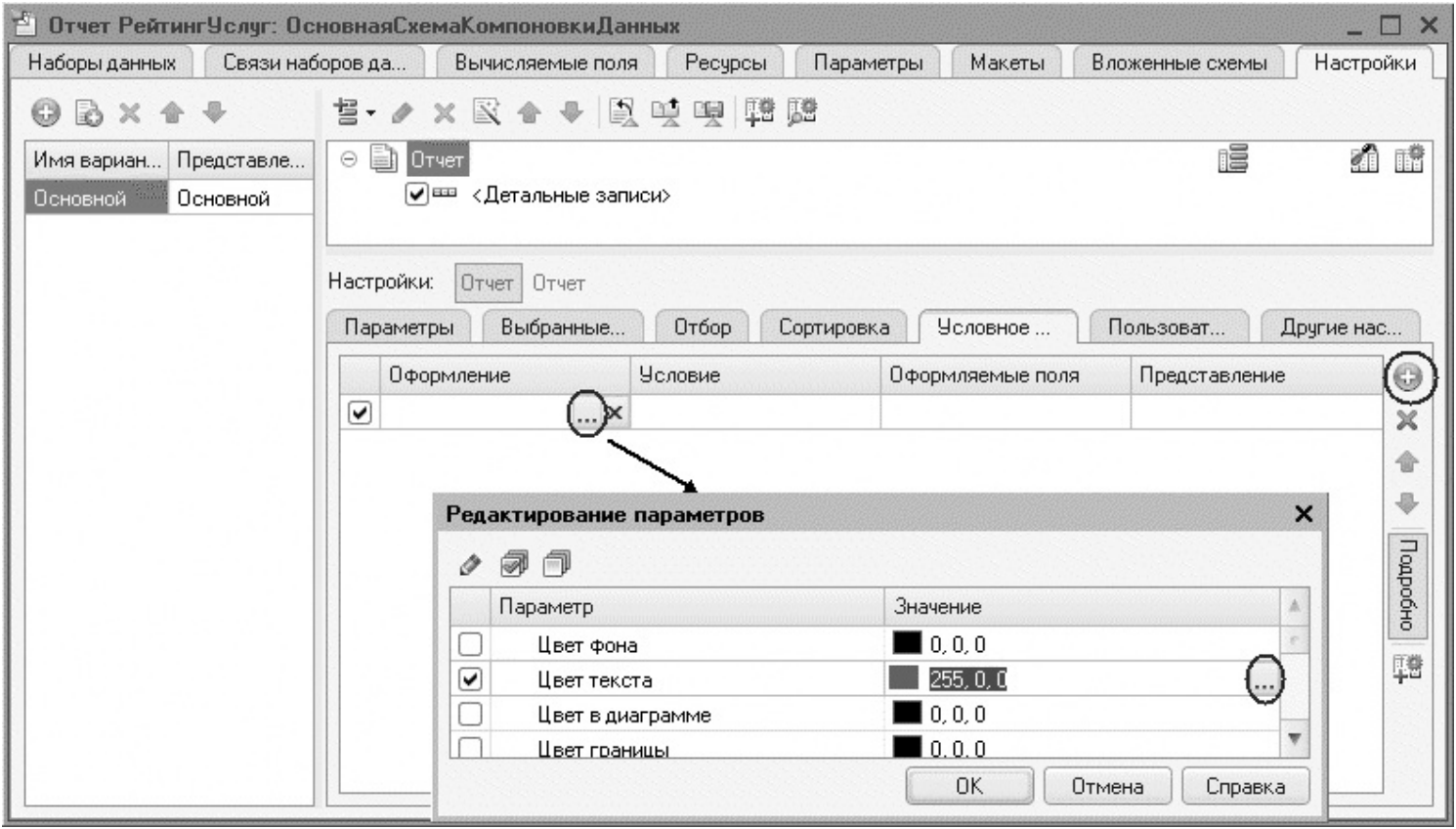


Рис. 13.43. Настройка условного оформления

Сначала укажем *Оформление*, то есть то, каким образом должны выделяться интересующие нас поля.

Нажмем кнопку выбора ... в поле *Оформление* и установим красный цвет текста. Нажмем *ОК*.

Затем укажем *Условие*, при наступлении которого будет применяться оформление, то есть когда в нашем случае текст будет становиться красным.

Нажмем кнопку выбора ... в поле *Условие* и в появившемся окне добавим *Новый элемент* отбора.

Каждый элемент отбора задает одно условие. Условий может быть несколько (рис. 13.44).

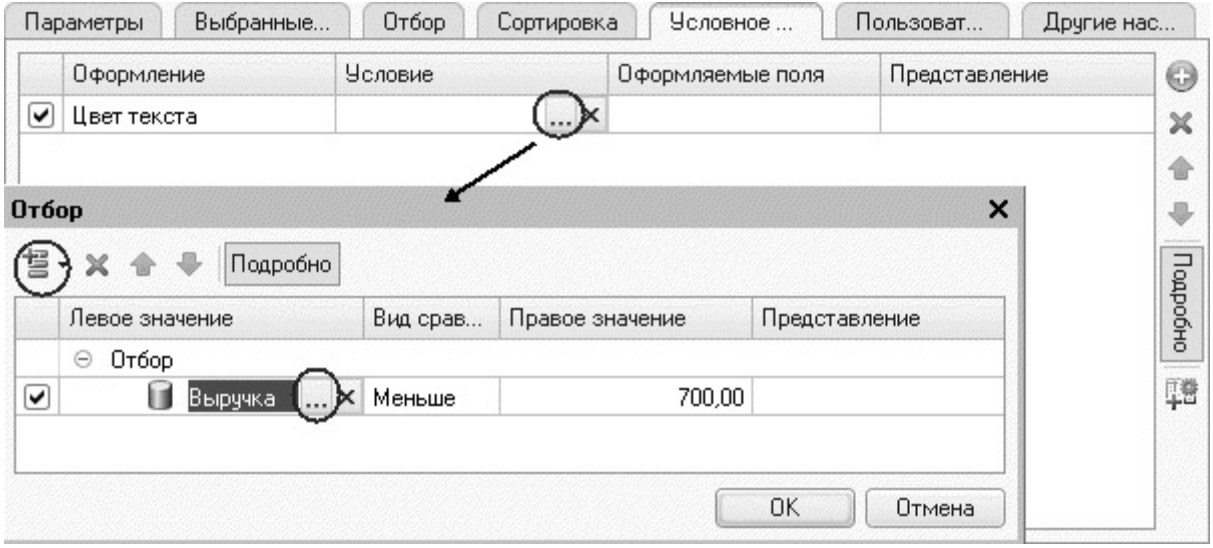


Рис. 13.44. Настройка условного оформления

Для этого нажмем кнопку *Добавить* и укажем в графе *Левое значение* – поле *Выручка*, в графе *Вид сравнения – Меньше*, а в графе *Правое значение – 700*. Нажмем *ОК*.

То есть когда в поле *Выручка* окажется значение, меньше 700 «что-то» будет выделено красным цветом текста.

Теперь укажем это «что-то», то есть зададим список *оформляемых полей*.

Если мы хотим выделять всю строку отчета, то можно оставить этот список

пустым. Или же нажать кнопку выбора ... в поле *Оформляемые поля* и в появившемся окне, нажимая кнопку *Добавить*, можно выбрать поля *Услуга* и *Выручка* (рис. 13.45).

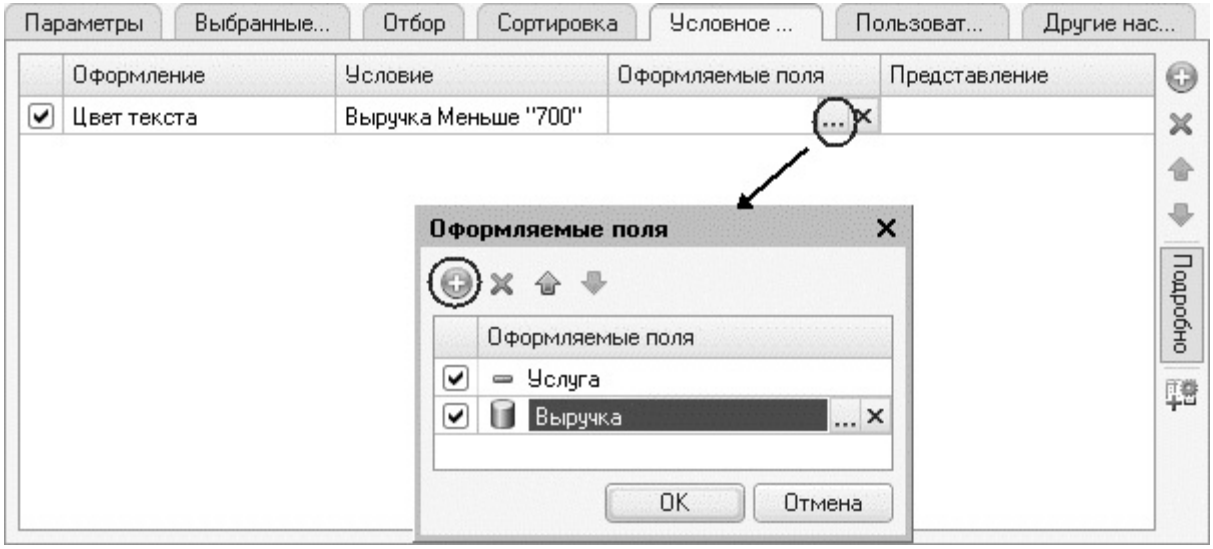


Рис. 13.45. Настройка условного оформления

В нашем случае можно было бы этого не делать, так как *Услуга* и *Выручка* и есть все поля отчета. Нажмем *ОК*.

В заключение зададим *Представление* условного оформления как *Непопулярная услуга* (рис. 13.46).

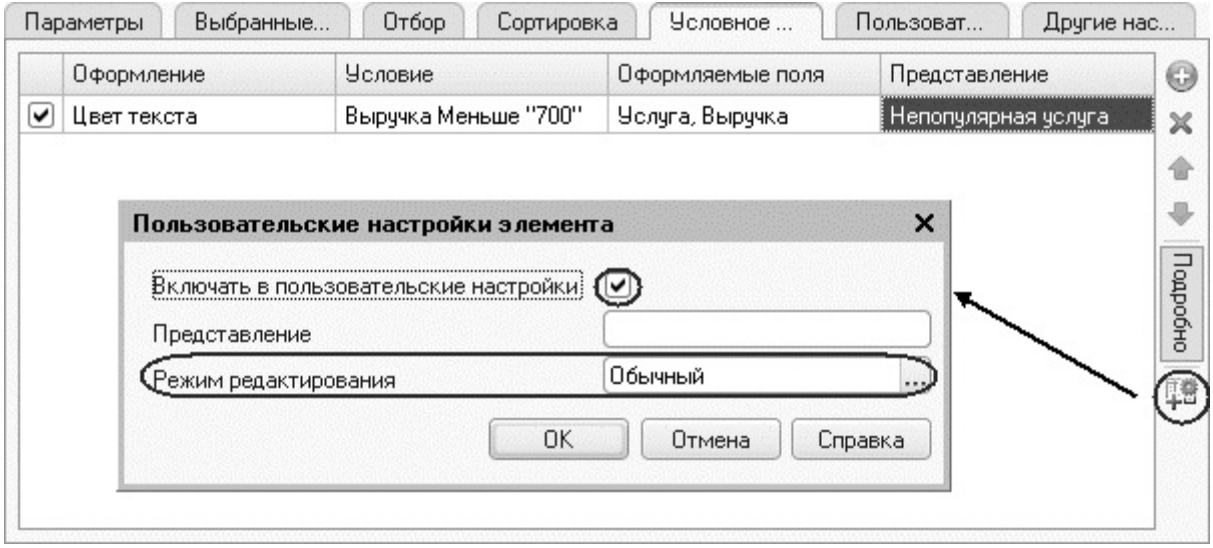


Рис. 13.46. Настройка условного оформления

Непопулярная услуга – это то, что увидит пользователь в своих настройках. То есть вместо пугающей строки «Выручка меньше 700…» пользователь увидит осмысленное выражение, которое задано в поле *Представление*.

Итак, мы задали условное оформление отчета, по которому все услуги с выручкой менее 700 руб. будут считаться «непопулярными» и выделяться красным цветом.

Теперь добавим это условие в пользовательские настройки. Нажмем кнопку *Свойства элемента пользовательских настроек*, расположенную в правом

нижнем углу окна настроек (см. рис. 13.46). Установим флажок *Включать в пользовательские настройки* и установим свойство *Режим редактирования* в значение *Обычный*.

Тем самым мы включили созданную нами настройку условного оформления в обычные пользовательские настройки. Эти настройки, в отличие от быстрых настроек, расположены не в форме отчета, а вызываются нажатием кнопки *Настройка* и появляются в отдельном окне, так эти настройки используются значительно реже, чем, например, настройки отчетного периода.

В режиме «1С:Предприятие»

Перейдем в режим *1С:Предприятие*. Вызовем отчет.

Зададим *Дату окончания* отчетного периода как *Начало этого дня* и нажмем кнопку *Сформировать* (рис. 13.47).

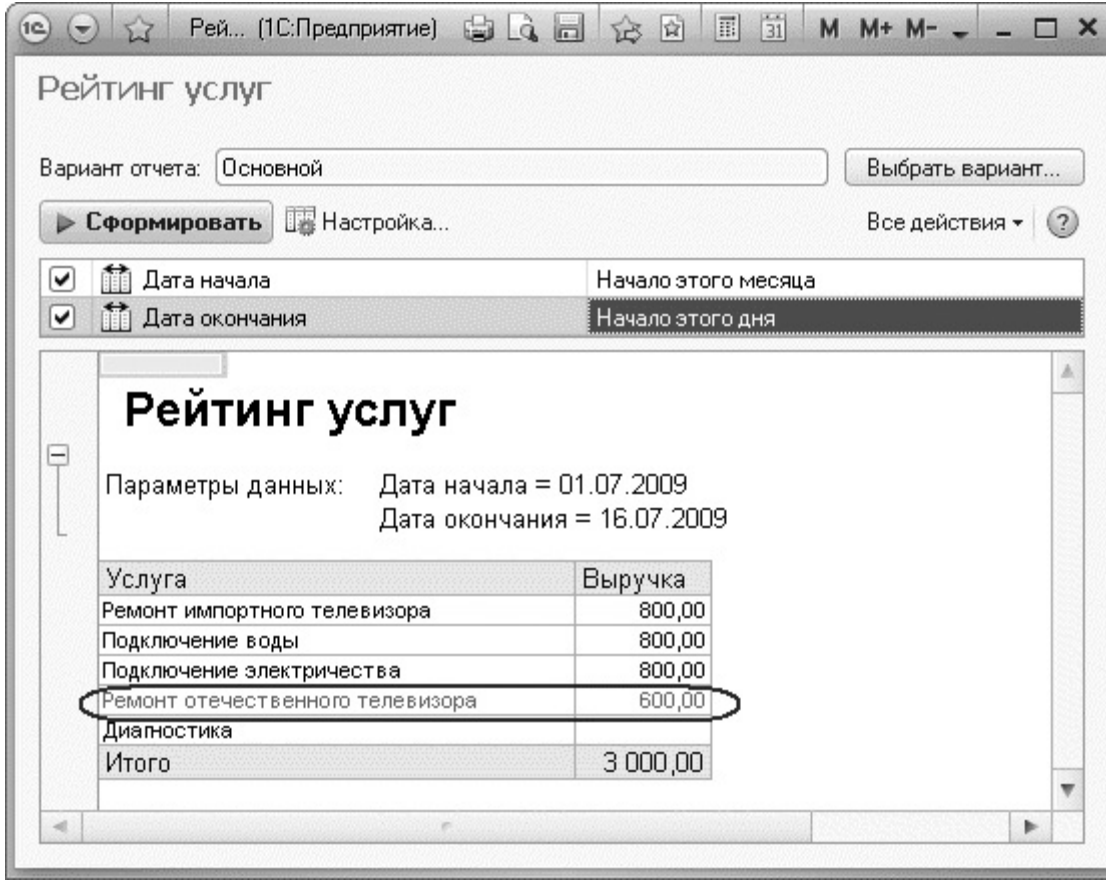


Рис. 13.47. Результат выполнения отчета

Мы видим, что суммы услуг менее 700 руб. выделены красным цветом. Нажмем кнопку *Настройка*.

Перед нами появится окно пользовательских настроек отчета, содержащее параметры отчетного периода и настройку условного оформления *Непопулярная услуга*. Мы можем снять флажок использования этой настройки, нажать кнопку *Завершить редактирование* (рис. 13.48) и снова выполнить отчет.

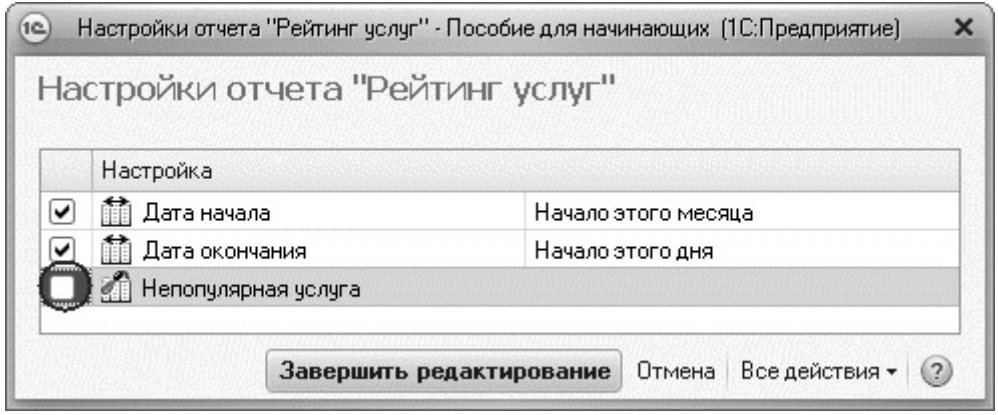


Рис. 13.48. Окно пользовательских настроек

Выделение цветом исчезнет. Настройка *Непопулярная услуга* не видна в форме отчета, так как мы установили для нее в качестве режима редактирования *Обычный*, а не *Быстрый доступ*.

Однако данная настройка условного оформления задана жестко, и пользователь может лишь включить или выключить признак ее использования.

Но для более подготовленных пользователей мы можем предоставить более полную свободу в использовании настроек, то есть возможность, например, самостоятельно задавать настройки отчета: отбор, порядок, условное оформление и пр.

Рассмотрим это в следующем примере.

Пользовательские настройки

В режиме «Конфигуратор»
Вернемся в конфигуратор.

На закладке *Настройки* схемы компоновки данных содержатся полные настройки отчета, которые задает разработчик. Часть из них может быть представлена пользователю для создания произвольного отбора, условного оформления отчета и пр.

Для этого нажмем кнопку *Свойства элемента пользовательских настроек*, расположенную вверху в командной панели окна настроек (рис. 13.49).

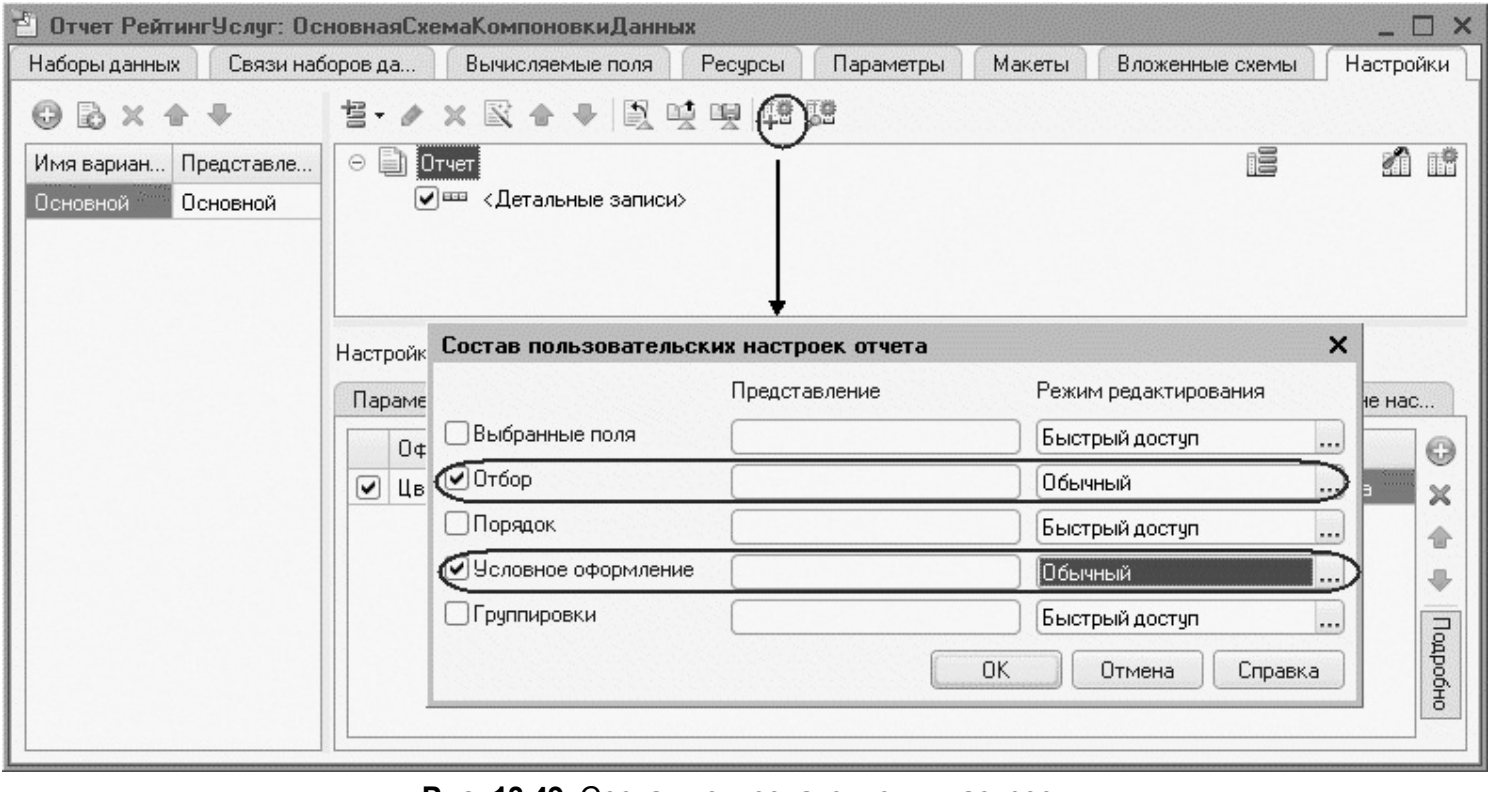


Рис. 13.49. Состав пользовательских настроек

В появившемся окне мы можем редактировать состав пользовательских настроек отчета.

Установим признак использования для настроек *Отбор* и *Условное оформление* и установим для них свойство *Режим редактирования* в значение

Обычный.

Таким образом, мы включили настройки отбора и условного оформления в состав пользовательских настроек и предоставили пользователю возможность задавать их в отдельном окне, вызываемом кнопкой *Настройка*.

Отбор

В режиме «Конфигуратор»

Теперь создадим настройку отбора в отчете. Для этого в нижней части окна настроек перейдем на закладку *Отбор*. Слева мы видим список доступных полей отчета. Раскроем поле *Услуга* и двойным щелчком мыши на поле *Родитель* перенесем его в список условий отбора в правой части окна (рис. 13.50).

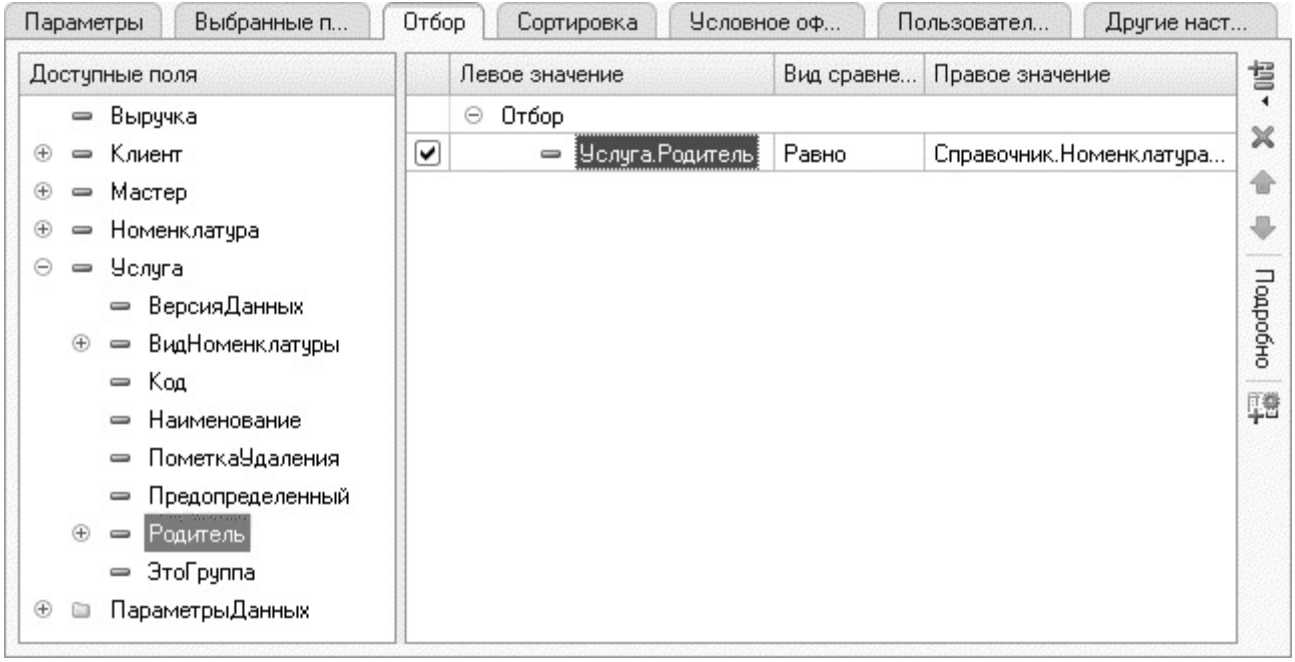


Рис. 13.50. Настройка отбора

Таким образом, мы создали возможность отбора по группам услуг, которые пользователь может задать в режиме *1С:Предприятие*.

В режиме «1С:Предприятие»

Откроем отчет в режиме *1С:Предприятие* и нажмем кнопку *Настройка*.

В окне пользовательских настроек отчета появились настройки *Отбор* и

Условное оформление, которые мы только что отметили (рис. 13.51).

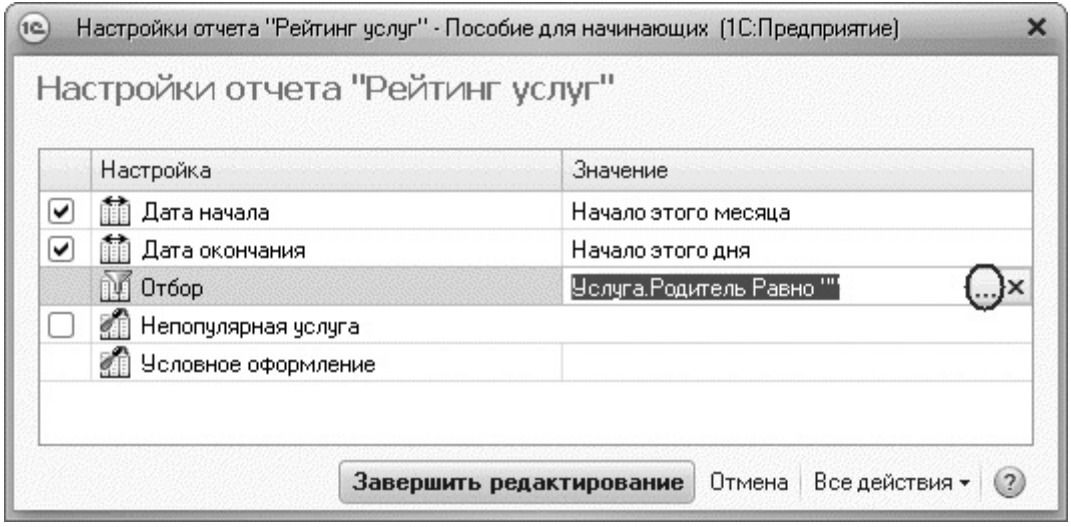


Рис. 13.51. Окно пользовательских настроек

На самом деле здесь присутствуют две настройки условного оформления.

Настройку *Непопулярная услуга* мы заранее создали в конфигураторе. А теперь, добавив настройку условного оформления «вообще», мы предоставили пользователю возможность создавать любое количество собственных условий для условного оформления аналогично тому, как мы это делали сами в конфигураторе. Сейчас мы это делать не будем, но самостоятельно вы можете попробовать.

Сейчас мы зададим отбор в отчете так, чтобы в него попадали только услуги, относящиеся к установке стиральных машин. Для этого нажмем кнопку выбора ... в окне пользовательских настроек в строке *Отбор* (см. рис. 13.51).

В открывшемся окне *Редактирование отбора* мы видим созданное нами ранее в конфигураторе условие отбора. Нам остается только нажать кнопку выбора ... в строке *Значение* и, раскрыв группу *Услуги*, выбрать группу *Стиральные машины* из справочника *Номенклатура* (рис. 13.52).

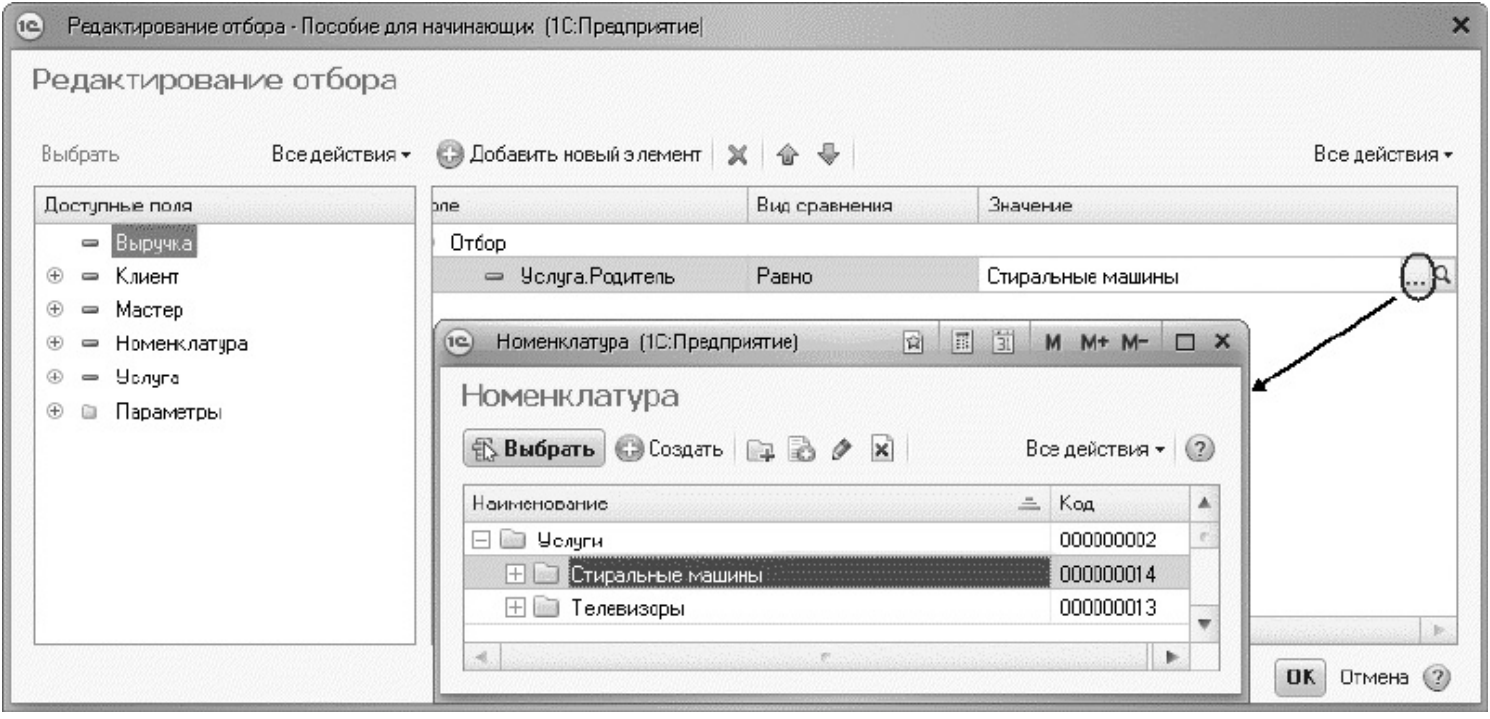


Рис. 13.52. Настройка отбора

Нажмем *ОК*.

Таким образом, мы задали отбор по услугам, родителем которых является группа *Стиральные машины* справочника *Номенклатура*.

В окне пользовательских настроек нажмем кнопку *Завершить редактирование*

и выполним отчет, нажав кнопку *Сформировать* (рис. 13.53).

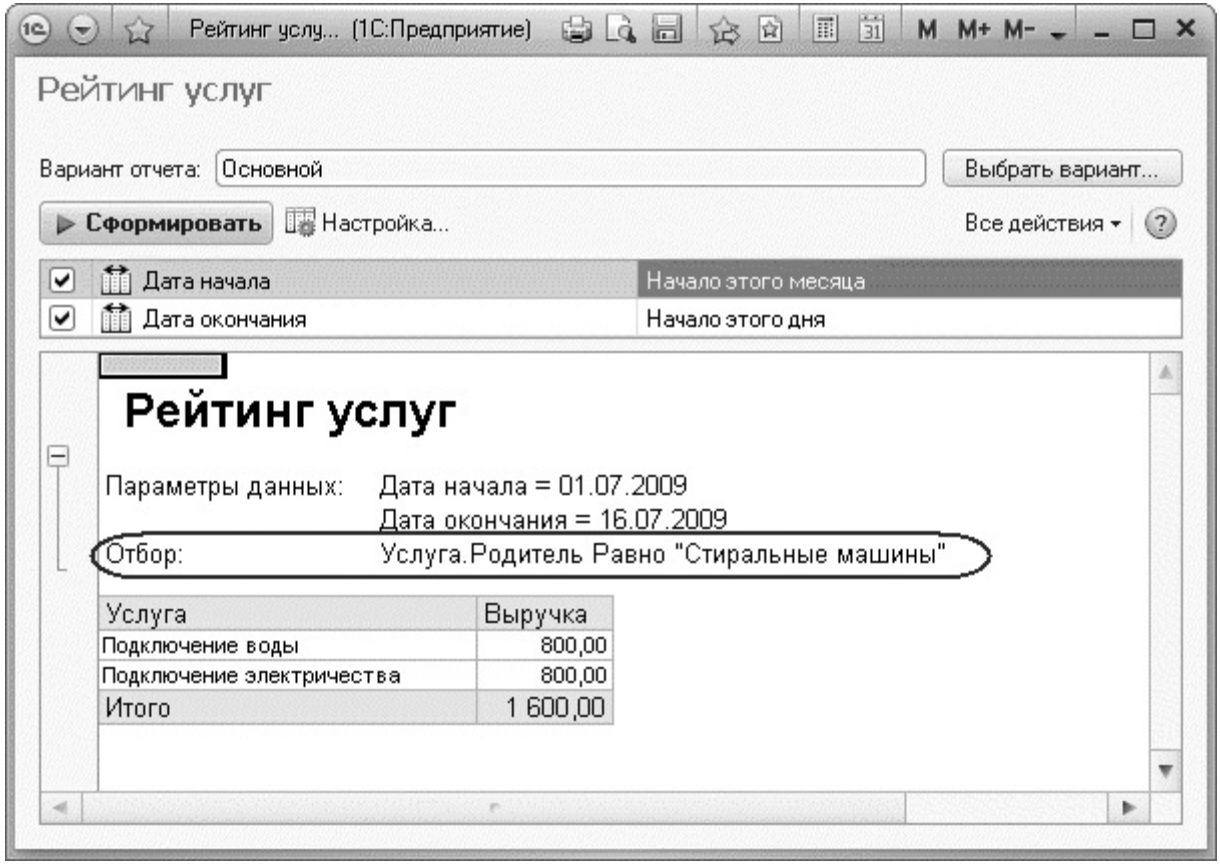


Рис. 13.53. Результат выполнения отчета

Мы видим, что в отчет включены только услуги по установке стиральных машин

и в заголовке отчета отражена информация об отборе.

Вызвав окно настроек, мы можем очистить настройку отбора, нажав кнопку очистки ×, или создать ее по другому критерию, нажав кнопку выбора ... в строке *Отбор* (рис. 13.54).

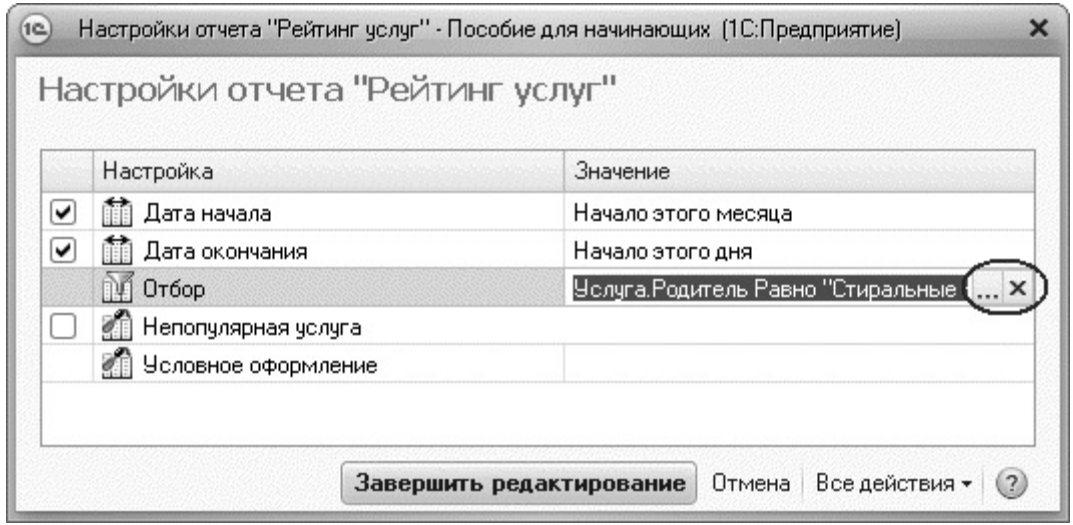


Рис. 13.54. Окно пользовательских настроек

Таким образом, пользователь сможет при наличии определенной квалификации задавать многие настройки по своему желанию.

Если же такого желания или соответствующих знаний у него нет, лучше

задавать эти настройки жестко, а пользователю останется только включать или выключать их использование.

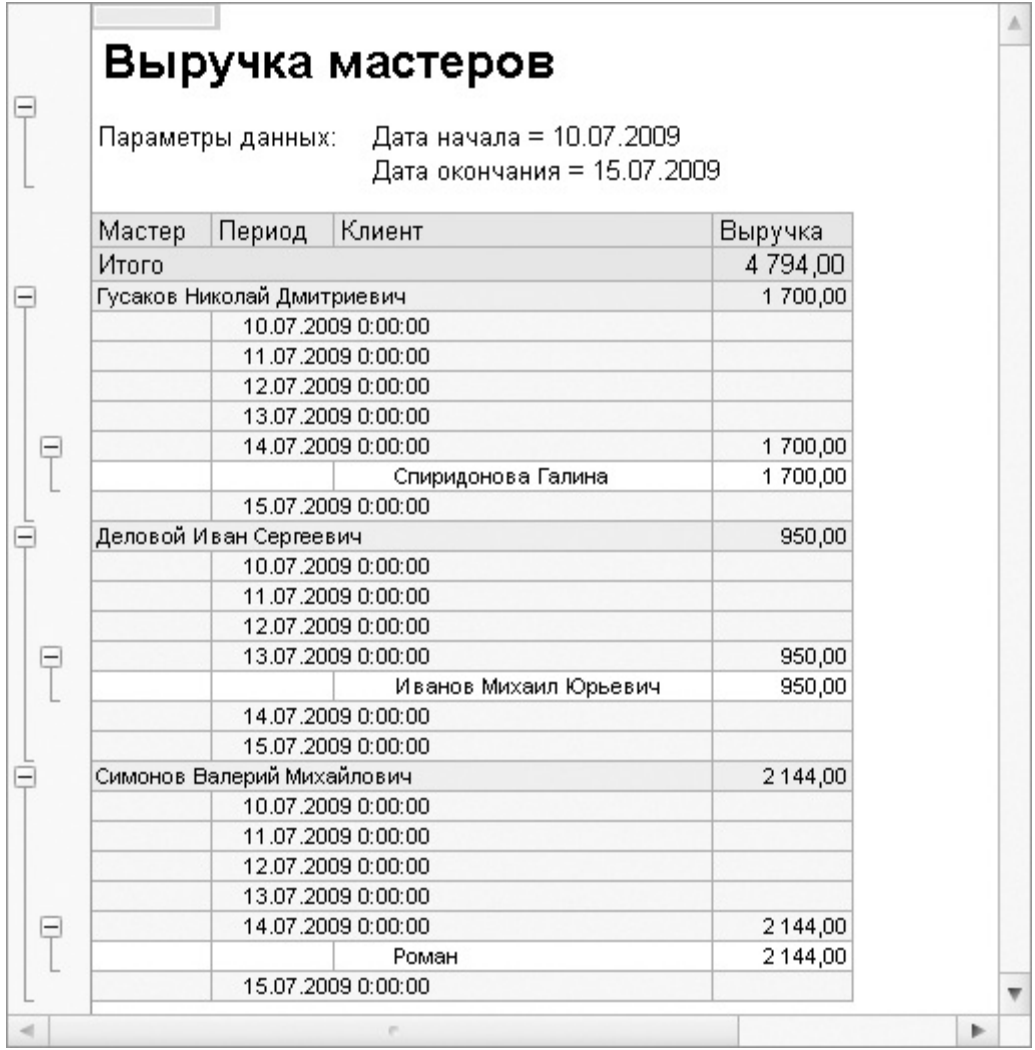
Да собственно часто достаточно только отчетного периода или еще какой-то жизненно важной настройки, и такие настройки, конечно, нужно размещать непосредственно в отчетной форме.

Теперь вернемся в конфигуратор и снимем признак использования у настройки отбора. Это нам понадобится в дальнейших примерах.

Вывод данных по всем дням в выбранном периоде

Следующий отчет, который мы добавим, будет называться *Выручка мастеров*.

Он будет содержать информацию о том, какая выручка была получена ООО «На все руки мастер» благодаря работе каждого из мастеров, с детализацией по всем дням в выбранном периоде и разворотом по клиентам, обслуженным в каждый из дней (рис. 13.55).



Выручка мастеров

Параметры данных: Дата начала = 10.07.2009
Дата окончания = 15.07.2009

Мастер	Период	Клиент	Выручка
Итого			4 794,00
Гусаков Николай Дмитриевич			1 700,00
	10.07.2009 0:00:00		
	11.07.2009 0:00:00		
	12.07.2009 0:00:00		
	13.07.2009 0:00:00		
	14.07.2009 0:00:00		1 700,00
		Спиридонова Галина	1 700,00
	15.07.2009 0:00:00		
Деловой Иван Сергеевич			950,00
	10.07.2009 0:00:00		
	11.07.2009 0:00:00		
	12.07.2009 0:00:00		
	13.07.2009 0:00:00		950,00
		Иванов Михаил Юрьевич	950,00
	14.07.2009 0:00:00		
	15.07.2009 0:00:00		
Симонов Валерий Михайлович			2 144,00
	10.07.2009 0:00:00		
	11.07.2009 0:00:00		
	12.07.2009 0:00:00		
	13.07.2009 0:00:00		
	14.07.2009 0:00:00		2 144,00
		Роман	2 144,00
	15.07.2009 0:00:00		

Рис. 13.55. Результат отчета

На примере этого отчета мы проиллюстрируем, как строить многоуровневые группировки в запросе и как обходить все даты в выбранном периоде.

Также продемонстрируем настройку отдельных элементов структуры отчета, научимся выводить данные в диаграмму и создавать несколько вариантов отчета в конфигураторе.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Отчет*.

Назовем его *ВыручкаМастеров* и запустим конструктор схемы компоновки данных.

Добавим новый *Набор данных – запрос* и вызовем конструктор запроса.

В качестве источника данных для запроса выберем виртуальную таблицу регистра накопления *Продажи.Обороты*.

Запрос для набора данных

Параметры виртуальной таблицы

Зададим один из параметров этой виртуальной таблицы – *Периодичность*.

Для этого перейдем в поле *Таблицы*, выделим таблицу и нажмем кнопку *Параметры виртуальной таблицы* (рис. 13.56).

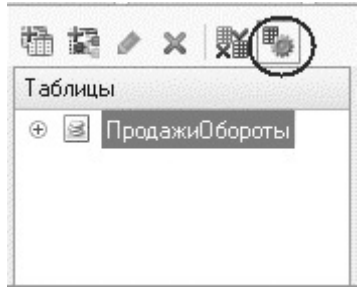


Рис. 13.56. Изменение параметров виртуальной таблицы

В открывшемся окне параметров зададим значение параметра *Периодичность* – *День* (рис. 13.57).

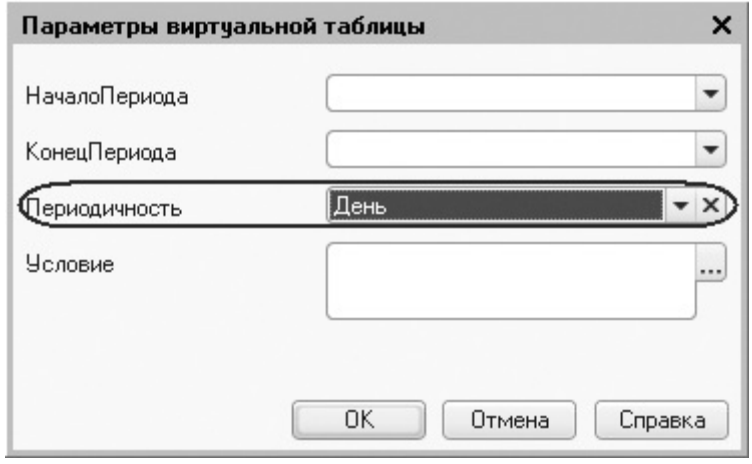


Рис. 13.57. Параметры виртуальной таблицы

Нажмем *OK*. После этого выберем из таблицы следующие поля (рис. 13.58):

- *ПродажиОбороты.Мастер,*
- *ПродажиОбороты.Период,*
- *ПродажиОбороты.Клиент,*
- *ПродажиОбороты.ВыручкаОборот*.

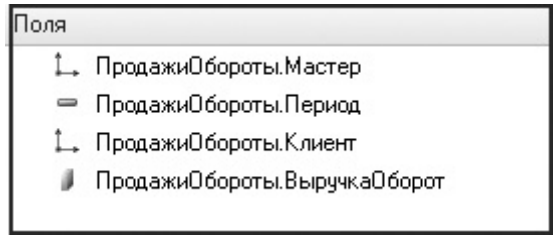


Рис. 13.58. Выбранные поля

Теперь перейдем на закладку *Объединения/Псевдонимы* и зададим псевдоним *Выручка* для поля *ПродажиОбороты.ВыручкаОборот* (рис. 13.59).

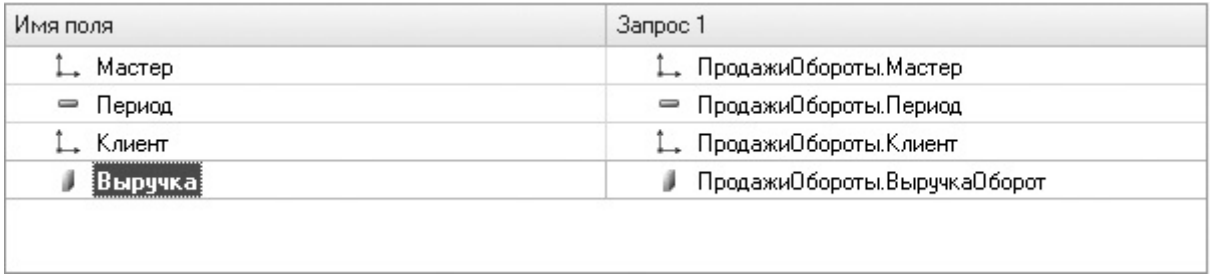


Рис. 13.59. Объединения/Псевдонимы

Анализ текста запроса

Нажмем *ОК* и рассмотрим текст запроса, сформированный конструктором (листинг 13.10).

Листинг 13.10. Текст запроса

```
ВЫБРАТЬ
  ПродажиОбороты.Мастер,
  ПродажиОбороты.Период,
  ПродажиОбороты.Клиент,
  ПродажиОбороты.ВыручкаОборот КАК Выручка
ИЗ
  РегистрНакопления.Продажи.Обороты( , , День, ) КАК ПродажиОбороты
```

В части описания запроса обратите внимание, что у источника данных задана периодичность выбираемых данных – *День* (листинг 13.11).

Листинг 13.11. Задание периодичности виртуальной таблицы

```
ИЗ
  РегистрНакопления.Продажи.Обороты( , , День, ) КАК ПродажиОбороты
```

Именно благодаря этому у нас появляется возможность описать среди выбранных полей поле *Период*.

Ресурсы

Теперь перейдем к редактированию схемы компоновки данных.

На закладке *Ресурсы* нажмем кнопку  и убедимся, что конструктор выбрал единственный имеющийся у нас ресурс – *Выручка*.

Параметры

На закладке *Параметры* выполним те же действия, что и при создании предыдущего отчета.

Для параметра *НачалоПериода* зададим заголовок *Дата начала*. В поле *Тип* зададим состав даты – *Дата*.

Затем добавим еще один параметр – *ДатаОкончания*, установим его тип как *Дата*, состав даты – *Дата*.

Для параметра *КонецПериода* зададим выражение (листинг 13.12) и в поле *Ограничение доступности* установим флажок ограничения доступности.

Листинг 13.12. Выражение для расчета значения параметра «КонецПериода»

```
КонецПериода(&ДатаОкончания, "День")
```

В результате перечисленных действий параметры компоновки данных будут иметь следующий вид (рис. 13.60).

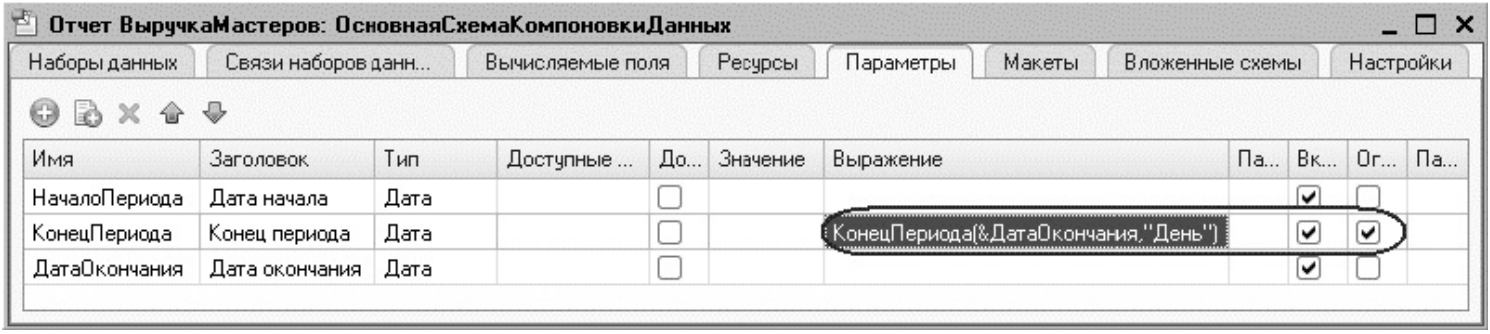


Рис. 13.60. Параметры компоновки данных

Настройки

Теперь создадим структуру отчета.

На закладке *Настройки* последовательно создадим две вложенные группировки:

- верхнего уровня – по полю *Мастер*;
- вложенная в нее – по полю *Период*.

Для этого сначала выделим корневой элемент *Отчет* в структуре отчета, нажмем кнопку *Добавить* в командной панели окна настроек, добавим новую группировку и укажем поле группировки *Мастер* (рис. 13.61).

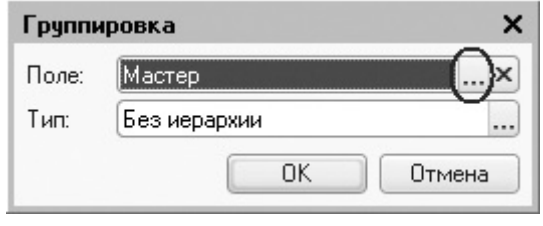


Рис. 13.61. Поле группировки

Затем добавим в группировку *Мастер* вложенную группировку по полю *Период*.

Для этого выделим группировку *Мастер*, нажмем кнопку *Добавить*, добавим новую группировку и укажем поле группировки *Период*.

Затем добавим еще одну группировку, вложенную в группировку по полю *Период, – Детальные записи* (без указания группировочного поля). Для этого

выделим группировку *Период*, нажмем кнопку *Добавить* и добавим новую группировку без указания группировочного поля. После этого перейдем на закладку *Выбранные поля* и добавим в список выбранных полей поля *Клиент* и *Выручка*.

Поля *Мастер* и *Период* мы не задаем, так как по этим полям производится группировка данных и их значение будет выведено автоматически.

В результате структура отчета будет иметь вид (рис. 13.62).

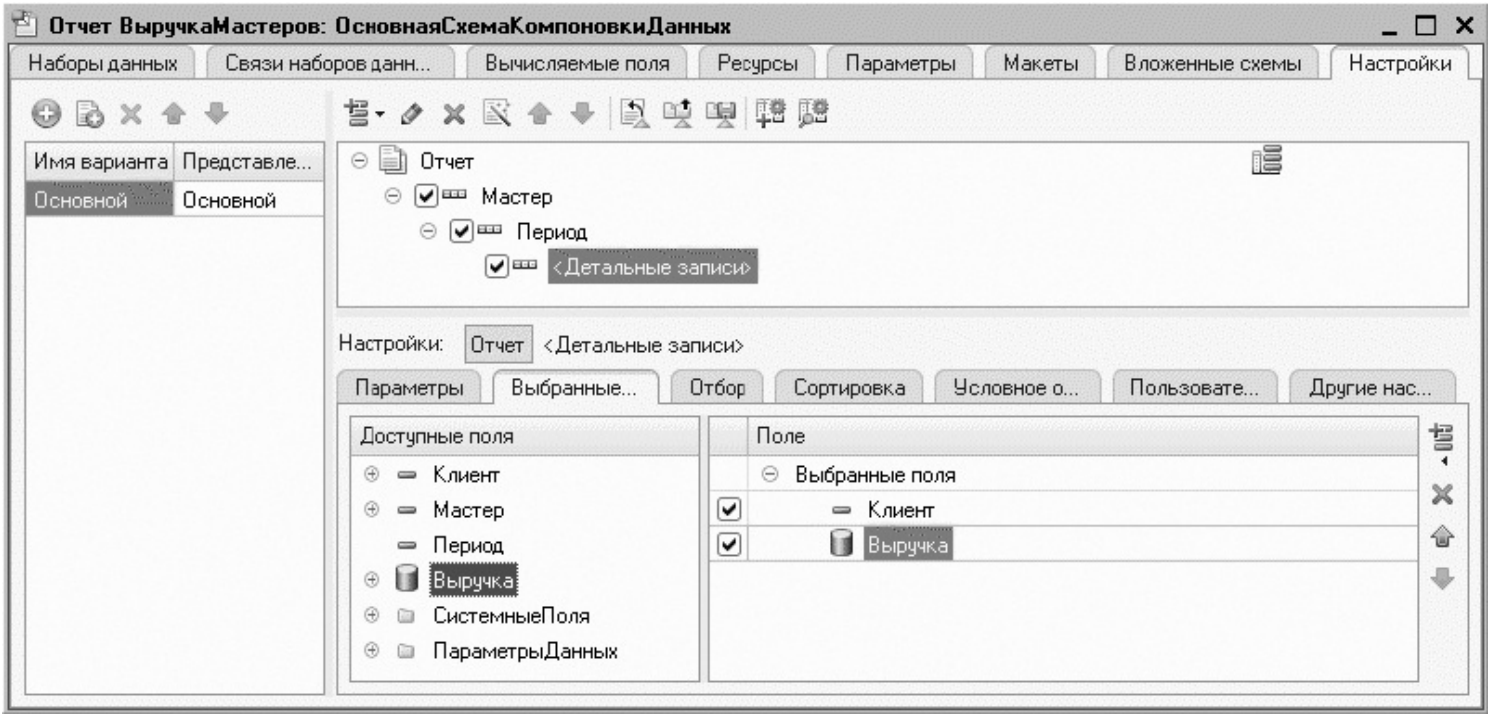


Рис. 13.62. Структура и поля отчета

В заключение перейдем на закладку *Другие настройки* и изменим следующие параметры.

Для параметра *Расположение полей группировок* установим значение *Отдельно и только в итогах*.

По умолчанию поля группировок в отчете располагаются вертикально друг под другом (рис. 13.63).

Мастер Период Клиент	Выручка
Гусаков Николай Дмитриевич	1 700,00
14.07.2009 0:00:00	1 700,00
Спиридонова Галина	1 700,00
Деловой Иван Сергеевич	950,00
13.07.2009 0:00:00	950,00
Иванов Михаил Юрьевич	950,00
Симонов Валерий Михайлович	2 144,00
14.07.2009 0:00:00	2 144,00
Роман	2 144,00
Итого	4 794,00

Рис. 13.63. Расположение полей группировок и итогов по вертикали по умолчанию

Установка этого свойства в значение *Отдельно и только в итогах* означает, что каждая группировка будет располагаться в отдельной области отчета слева направо и ее наименование будет выводиться только в данной группировке (рис. 13.64).

Мастер	Период	Клиент	Выручка
Гусаков Николай Дмитриевич			1 700,00
	14.07.2009 0:00:00		1 700,00
		Спиридонова Галина	1 700,00
Деловой Иван Сергеевич			950,00
	13.07.2009 0:00:00		950,00
		Иванов Михаил Юрьевич	950,00
Симонов Валерий Михайлович			2 144,00
	14.07.2009 0:00:00		2 144,00
		Роман	2 144,00
Итого			4 794,00

Рис. 13.64. Расположение полей группировок «Отдельно и только в итогах»

Для параметра *Расположение общих итогов по вертикали* зададим значение *Начало*.

По умолчанию итоги по вертикали располагаются в конце (см. рис. 13.63). Установка этого свойства означает, что общие итоги будут отображаться в начале перед строками группировки (рис. 13.65).

Мастер	Период	Клиент	Выручка
Итого			4 794,00
Гусаков Николай Дмитриевич			1 700,00
	14.07.2009 0:00:00		1 700,00
		Спиридонова Галина	1 700,00
Деловой Иван Сергеевич			950,00
	13.07.2009 0:00:00		950,00
		Иванов Михаил Юрьевич	950,00
Симонов Валерий Михайлович			2 144,00
	14.07.2009 0:00:00		2 144,00
		Роман	2 144,00

Рис. 13.65. Расположение итогов по вертикали в начале

В результате другие настройки отчета примут вид (рис. 13.66).

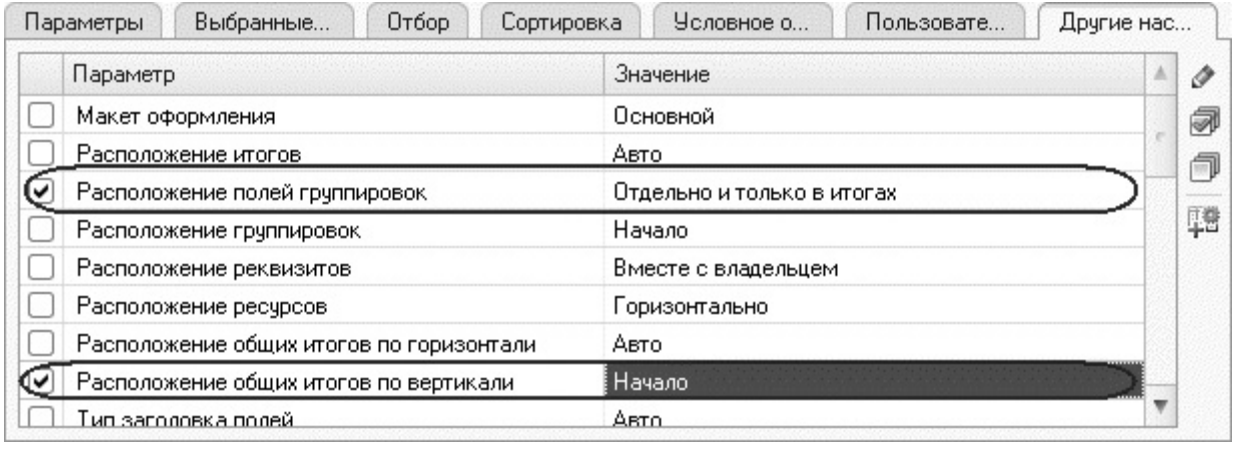


Рис. 13.66. Параметры настроек вывода отчета

Здесь же для параметра *Заголовок* зададим значение *Выручка мастеров.*

Затем укажем, что параметры *Дата начала* и *Дата окончания* будут включены в состав пользовательских настроек, и эти настройки будут находиться непосредственно в отчетной форме, то есть будут «быстрыми» настройками.

Таким образом, перед формированием отчета пользователь сможет задать отчетный период (рис. 13.67).

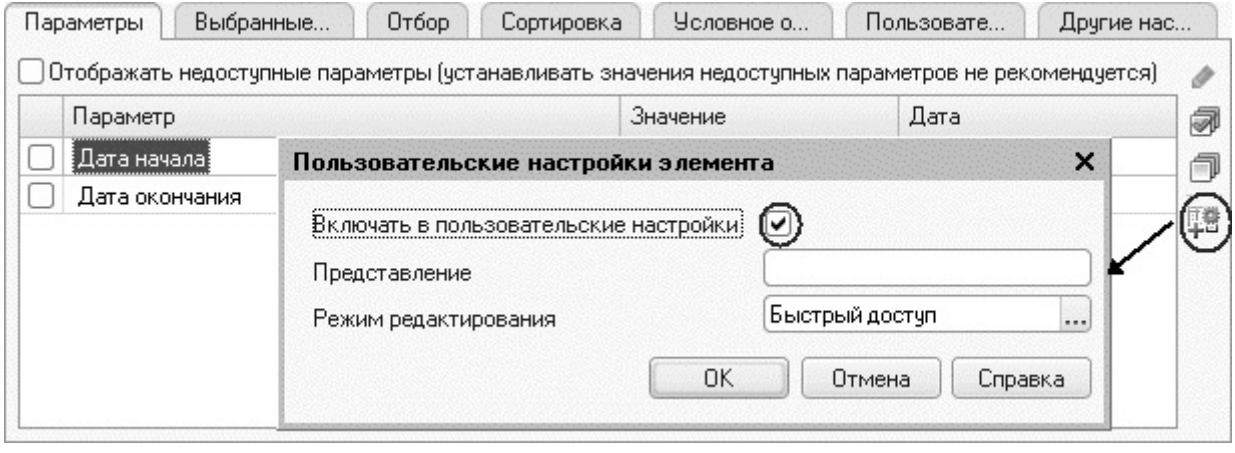


Рис. 13.67. Создание быстрых настроек отчетного периода

В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации *Отчет ВыручкаМастеров* перейдем на закладку *Подсистемы*.

Отметим в списке подсистем конфигурации подсистемы *Оказание услуг* и *Расчет зарплаты*. Таким образом, ссылка на наш отчет автоматически попадет в панель действий этих подсистем.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает отчет.

В открывшемся окне «1С:Предприятия» мы видим, что в панели действий разделов *Оказание услуг* и *Расчет зарплаты* в группе команд для выполнения отчетов появилась команда для формирования отчета *Выручка мастеров*.

Выполним эту команду. Зададим отчетный период с *01.07.2009* по *15.07.2009* и сформируем отчет (рис. 13.68).

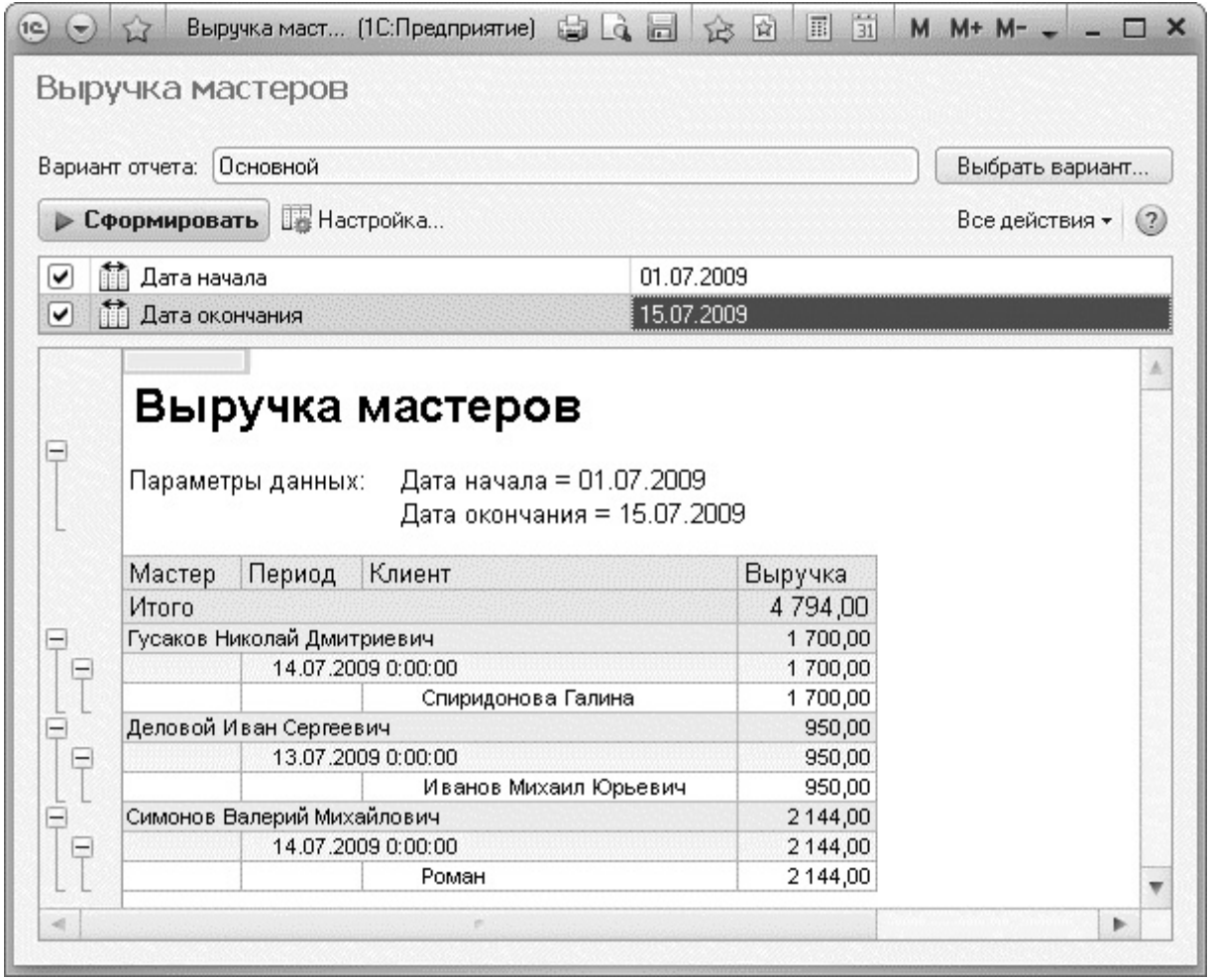


Рис. 13.68. Результат выполнения отчета

Вывод всех дат в выбранном периоде

Если вы помните, в начале раздела мы говорили, что этот отчет должен показывать данные с детализацией по всем дням в выбранном периоде.

У нас же отображаются только те дни, для которых существуют ненулевые записи в таблице регистра накопления *Продажи*.

Для детализации данных в отчете система компоновки данных позволяет указывать для группировок *дополнение* периодов с заданной периодичностью в указанном интервале.

Поэтому сейчас мы изменим настройки отчета таким образом, чтобы в отчет попадала каждая дата из периода, за который сформирован отчет.

В режиме «Конфигуратор»

Вернемся в режим *Конфигуратор* и выполним более тонкую настройку структуры отчета. Откроем схему компоновки данных на закладке *Настройки*.

До сих пор все настройки структуры, которые мы выполняли, относились ко всему отчету в целом. Но система компоновки данных позволяет настраивать также и каждый элемент структуры в отдельности.

ВНИМАНИЕ!

При установке настроек отчета в средней части окна, под деревом структуры отчета, должна быть выделена кнопка, соответствующая режиму настроек. Кнопка Отчет *– для настройки отчета в целом или кнопка с именем группировки, например* Детальные записи*, если настройки относятся только к ней.*

В нашем случае потребуется изменить настройку группировки *Период*.

Для того чтобы перейти к настройкам именно этой группировки, в поле структуры отчета установим курсор на эту группировку, а затем нажмем кнопку *Период* в командной панели окна.

В нижней части окна будут отображены настройки, доступные для данной группировки.

Перейдем на закладку *Поля группировки*. Для поля *Период* установим *Тип дополнения – День* (рис. 13.69).

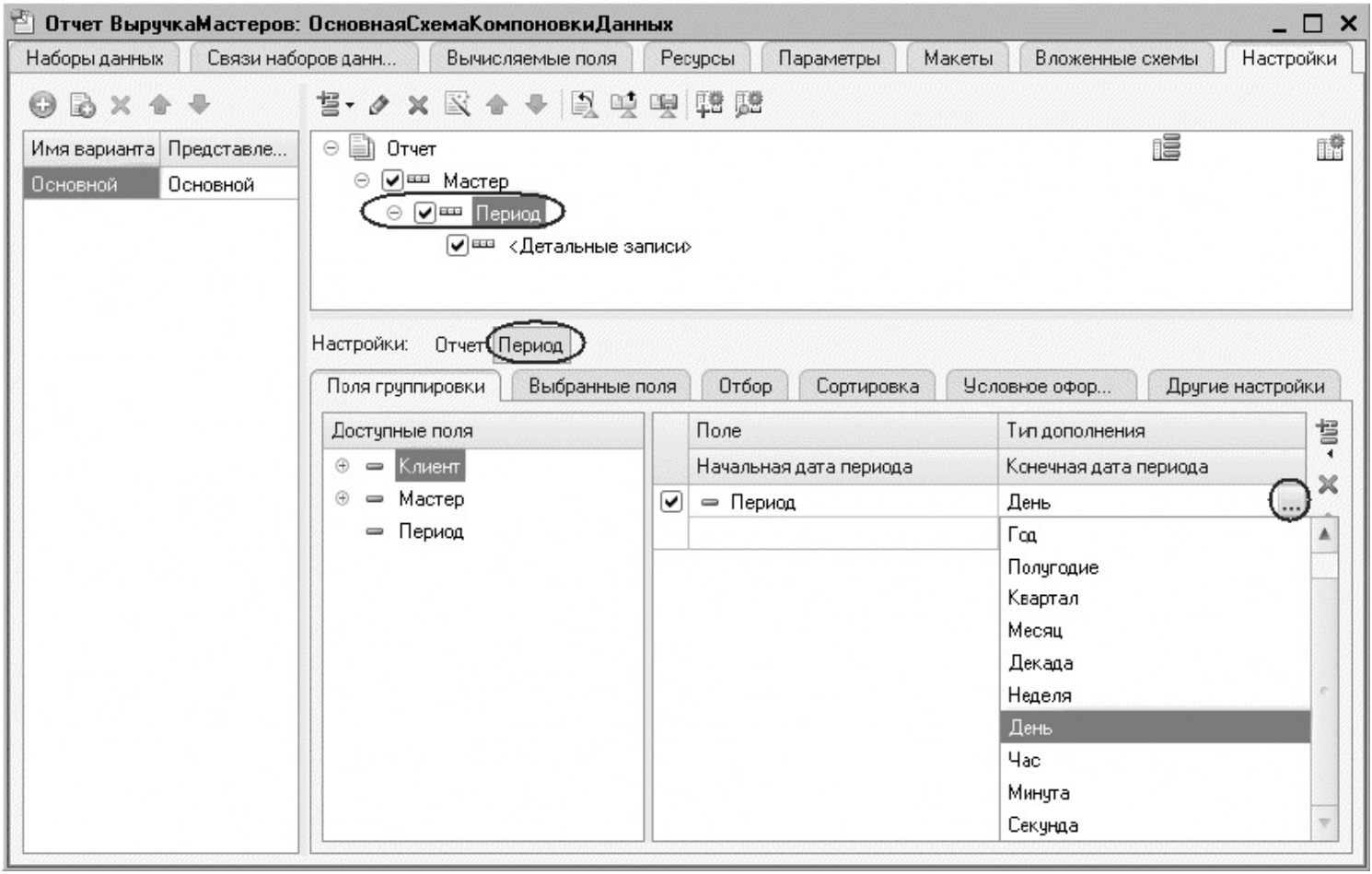


Рис. 13.69. Установка типа дополнения периода

Тем самым мы укажем, что для этой группировки существующие записи с

ненулевым значением ресурса будут дополняться записями для каждого из дней.

После этого следует указать, в каком именно периоде будет выполняться такое дополнение.

В поля, расположенные строчкой ниже, можно ввести даты начала и окончания этого периода. Но указание дат в явном виде нас не устраивает, так как пользователь может сформировать отчет за произвольный период. И нам нужно, чтобы дополнение дат выполнялось не в некотором фиксированном периоде, а именно в том периоде, который выбрал пользователь для всего отчета.

Для того чтобы обеспечить именно такую работу отчета, войдем в режим редактирования поля *Начальная дата периода*, дважды кликнув на нем, и нажмем кнопку очистки ×.

После этого, нажав кнопку выбора типа данных T, мы сможем выбрать тип данных, отображаемых в этом поле. Выберем *Поле компоновки данных* (рис. 13.70).

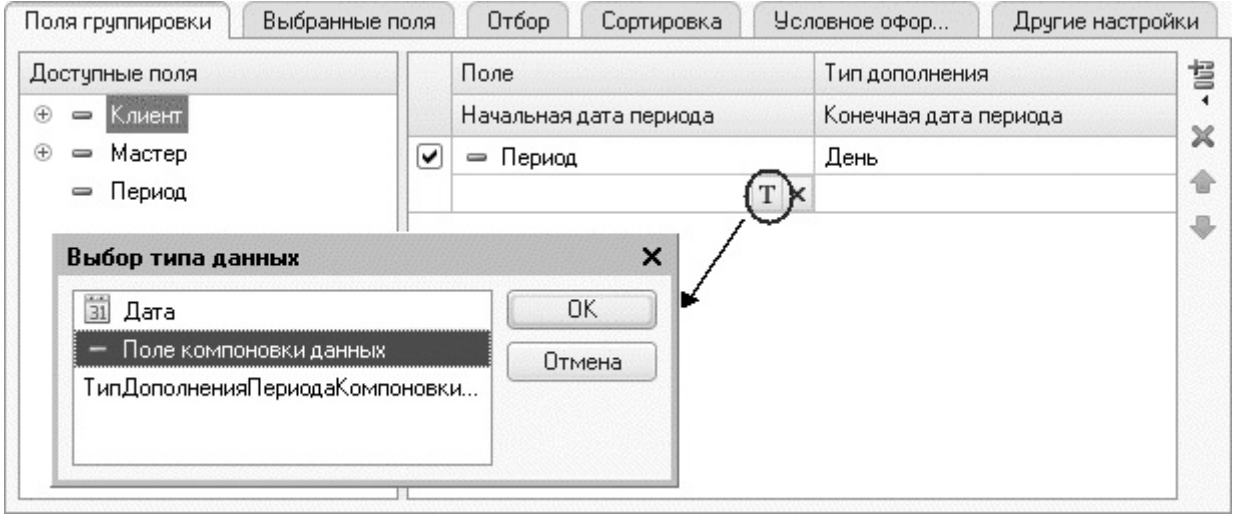


Рис. 13.70. Выбор типа данных

Нажмем *OK*.

Теперь нажмем в поле ввода кнопку выбора ... и в открывшемся окне выбора поля отметим параметр *НачалоПериода* (рис. 13.71). Нажмем *OK*.

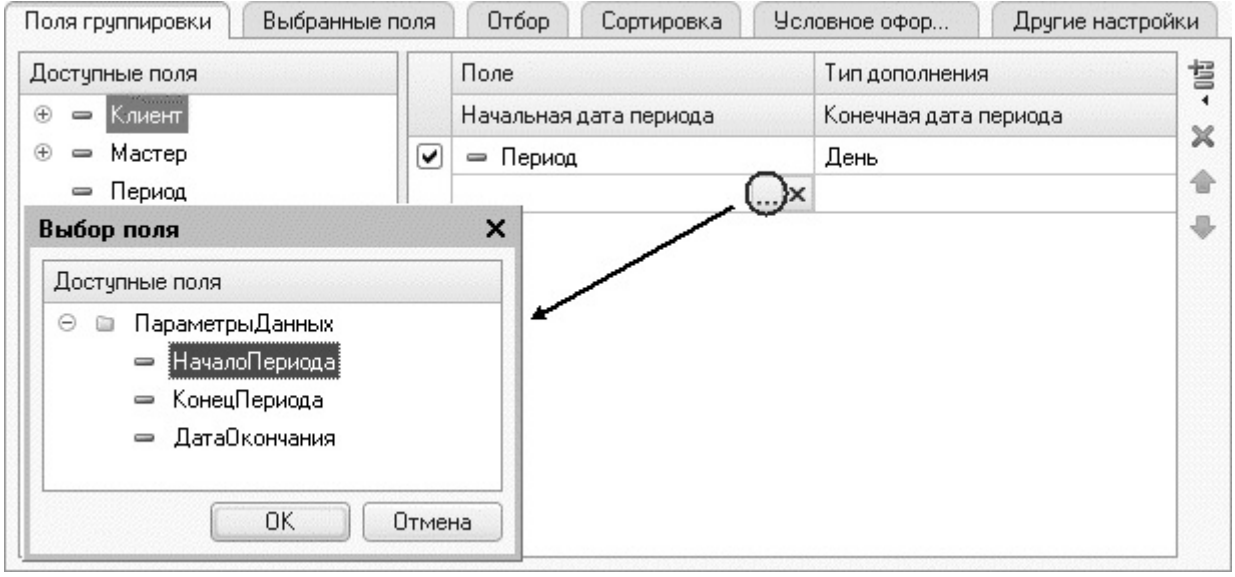


Рис. 13.71. Выбор поля

Для второго поля ввода аналогичным образом укажем, что дата окончания периода будет получена из параметра *ДатаОкончания* (рис. 13.72).

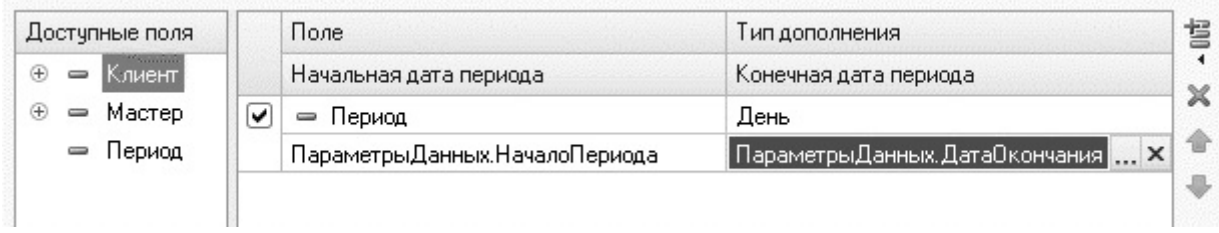


Рис. 13.72. Настройки группировки «Период»

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и выполним отчет *Выручка мастеров* за период с *10.07.2009* по *15.07.2009* (рис. 13.73).

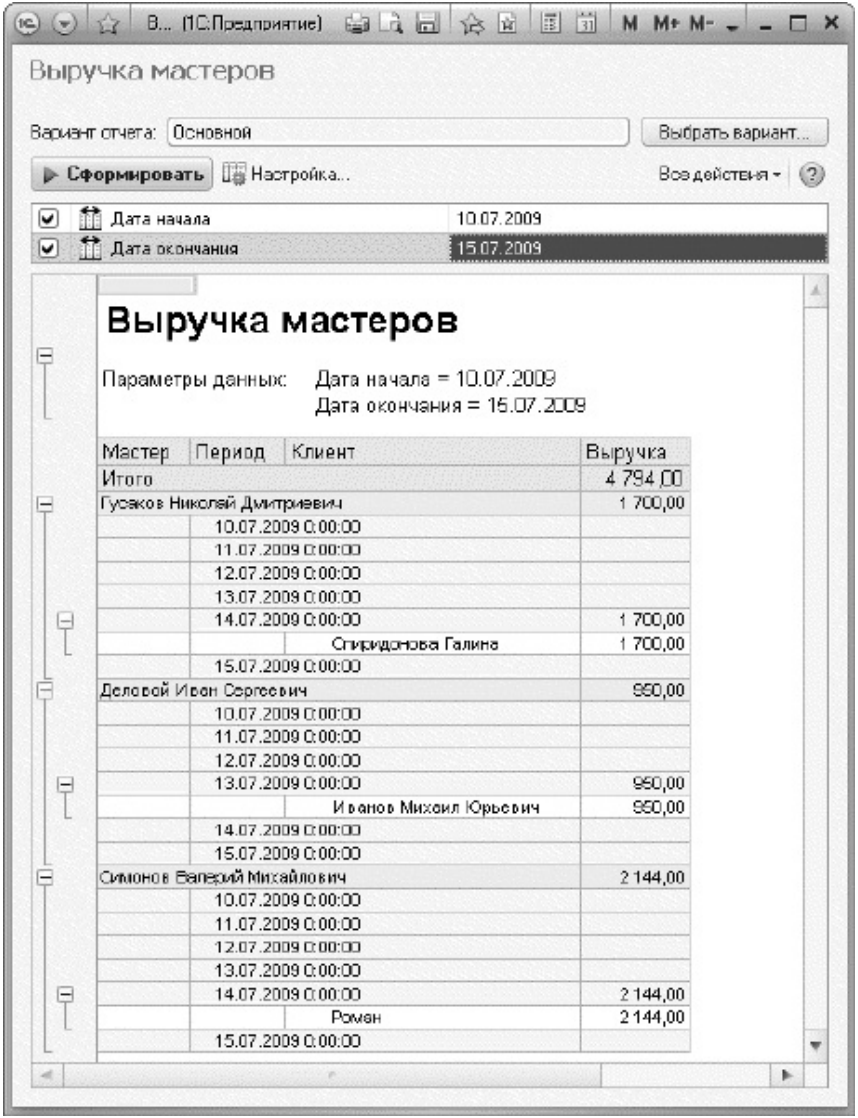

В... (1С:Предприятие)
Выручка мастеров
Вариант отчета: Основной
Выбрать вариант...
Сформировать
Настройка...
Все действия
Дата начала 10.07.2009
Дата окончания 15.07.2009
Выручка мастеров
Параметры данных: Дата начала = 10.07.2009
Дата окончания = 15.07.2009
Мастер | Период | Клиент | Выручка
Итого 4 794,00
Гусаков Николай Дмитриевич 1 700,00
10.07.2009 0:00:00
11.07.2009 0:00:00
12.07.2009 0:00:00
13.07.2009 0:00:00
14.07.2009 0:00:00 1 700,00
Спиридонова Галина 1 700,00
15.07.2009 0:00:00
Деловой Иван Сергеевич 950,00
10.07.2009 0:00:00
11.07.2009 0:00:00
12.07.2009 0:00:00
13.07.2009 0:00:00 950,00
Иванов Михаил Юрьевич 950,00
14.07.2009 0:00:00
15.07.2009 0:00:00
Симонов Валерий Михайлович 2 144,00
10.07.2009 0:00:00
11.07.2009 0:00:00
12.07.2009 0:00:00
13.07.2009 0:00:00
14.07.2009 0:00:00 2 144,00
Роман 2 144,00
15.07.2009 0:00:00

Рис. 13.73. Результат выполнения отчета

Новый вариант отчета

Для анализа работы мастеров за определенный период может понадобиться представить ту же информацию в другом, более наглядном виде. Например, директору при начислении зарплаты, чтобы понять, какой из мастеров лучше работает, вполне может понадобиться увидеть диаграмму, отражающую вклад каждого мастера в общую выручку предприятия за период.

Поэтому мы создадим другой вариант отчета *ВыручкаМастеров*, представляющий данные в виде диаграммы.

Диаграмма

Диаграмма предназначена для размещения в таблицах и формах диаграмм и графиков различного вида.

Логически диаграмма является совокупностью точек, серий и значений серий в точке (рис. 13.74).

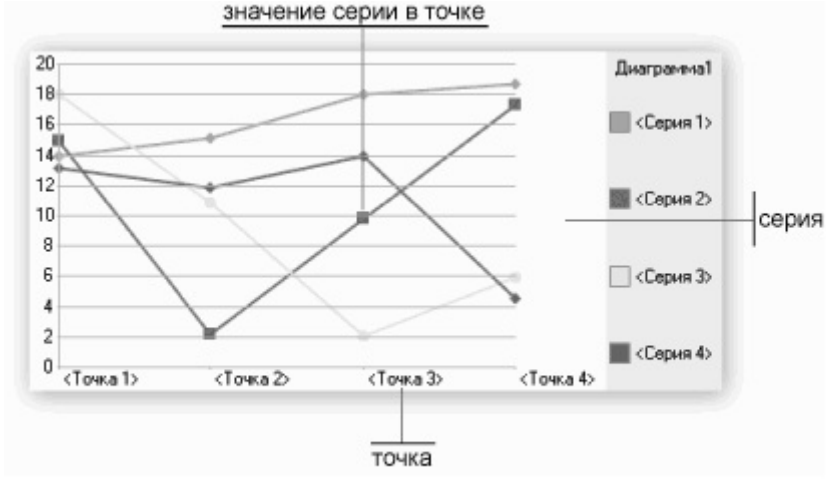


Рис. 13.74. Пример диаграммы

Как правило, в качестве *точек* используются моменты или объекты, для которых мы получаем значения характеристик, а в качестве *серий* – характеристики, значения которых нас интересуют. На пересечении серии и точки находится *значение* диаграммы.

Например, диаграмма продаж видов номенклатуры по месяцам будет состоять из точек – месяцев, серий – видов номенклатуры и значений – оборотов продаж.

Диаграмма как объект встроенного языка имеет три области, которые

позволяют управлять оформлением диаграммы: область построения, область заголовка и область легенды (рис. 13.75).

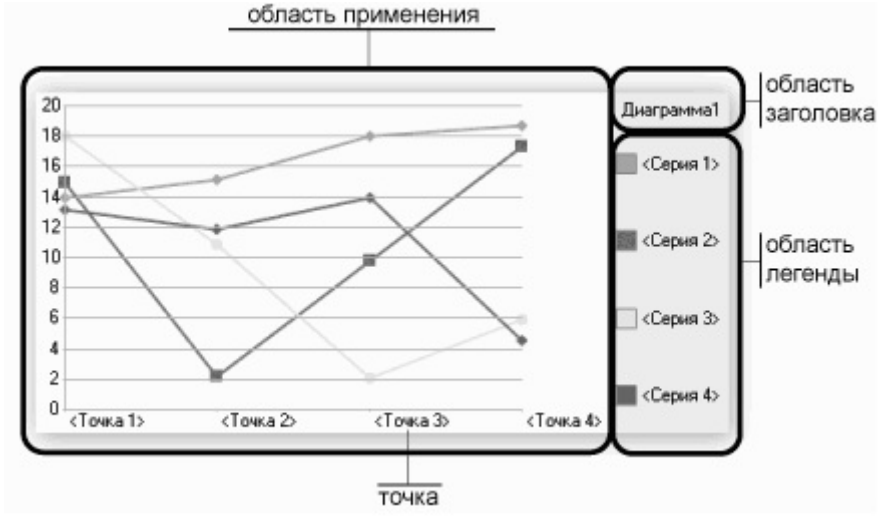


Рис. 13.75. Области диаграммы

Диаграмма может быть вставлена в структуру отчета как отдельный элемент. В следующем варианте настроек отчета *ВыручкаМастеров* мы будем использовать диаграмму в структуре настроек схемы компоновки данных.

В режиме «Конфигуратор»

Вернемся в конфигуратор и откроем схему компоновки данных на закладке *Настройки*.

В левой части окна находится список вариантов отчета.

При создании настроек отчета в первый раз система компоновки данных по умолчанию создает *Основной* вариант настроек. И мы видим его в списке вариантов нашего отчета. Чтобы добавить новый вариант, нажмем кнопку *Добавить* над этим списком. Зададим имя варианта – *ОбъемВыручки* (рис. 13.76).

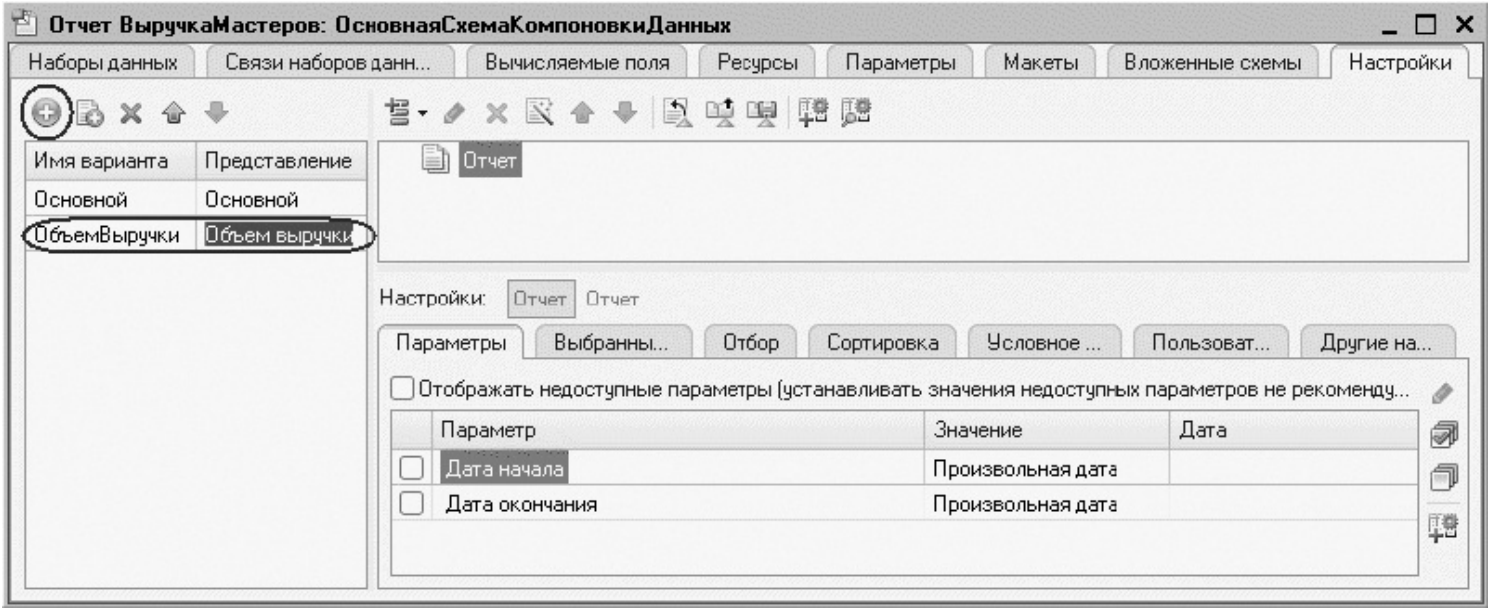


Рис. 13.76. Добавление нового варианта настроек

Мы видим, что структура отчета и все его настройки очистились.

Но они не пропали, а стали невидимы, так как относятся к *Основному* варианту настроек.

Если у отчета есть несколько вариантов, то мы видим и можем изменять настройки того варианта, который выделен в данный момент. Причем вся остальная информация в схеме компоновке данных (ресурсы, параметры, наборы данных) осталась без изменений. Данные для отчета будут получены с помощью того же запроса к базе данных. Изменятся лишь настройки, которые определят, как будет представлен отчет.

Добавим в структуру отчета диаграмму. Для этого выделим корневой элемент *Отчет*, вызовем его контекстное меню и добавим диаграмму (рис. 13.77).

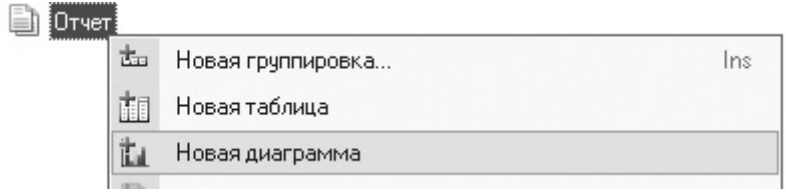


Рис. 13.77. Добавление диаграммы в структуру отчета

Затем выделим ветку *Точки* и добавим в нее группировку по полю *Мастер*. *Серии* диаграммы оставим без изменений.

Для демонстрации вклада мастеров в общий объем выручки хорошо подойдет измерительная диаграмма, которую мы хотим показать. Для этого вида диаграммы достаточно задать только точки, поэтому серии мы не задаем.

В значения диаграммы всегда выводится один из ресурсов отчета. У нас всего один ресурс – *Выручка* (поле ресурса помечено соответствующей пиктограммой и отличается от обычных полей).

Поэтому перейдем на закладку *Выбранные поля,* перейдем на уровень настроек отчета в целом (нажав кнопку *Отчет*) и выберем поле *Выручка* для вывода в отчет.

Структура отчета должна принять следующий вид (рис. 13.78).

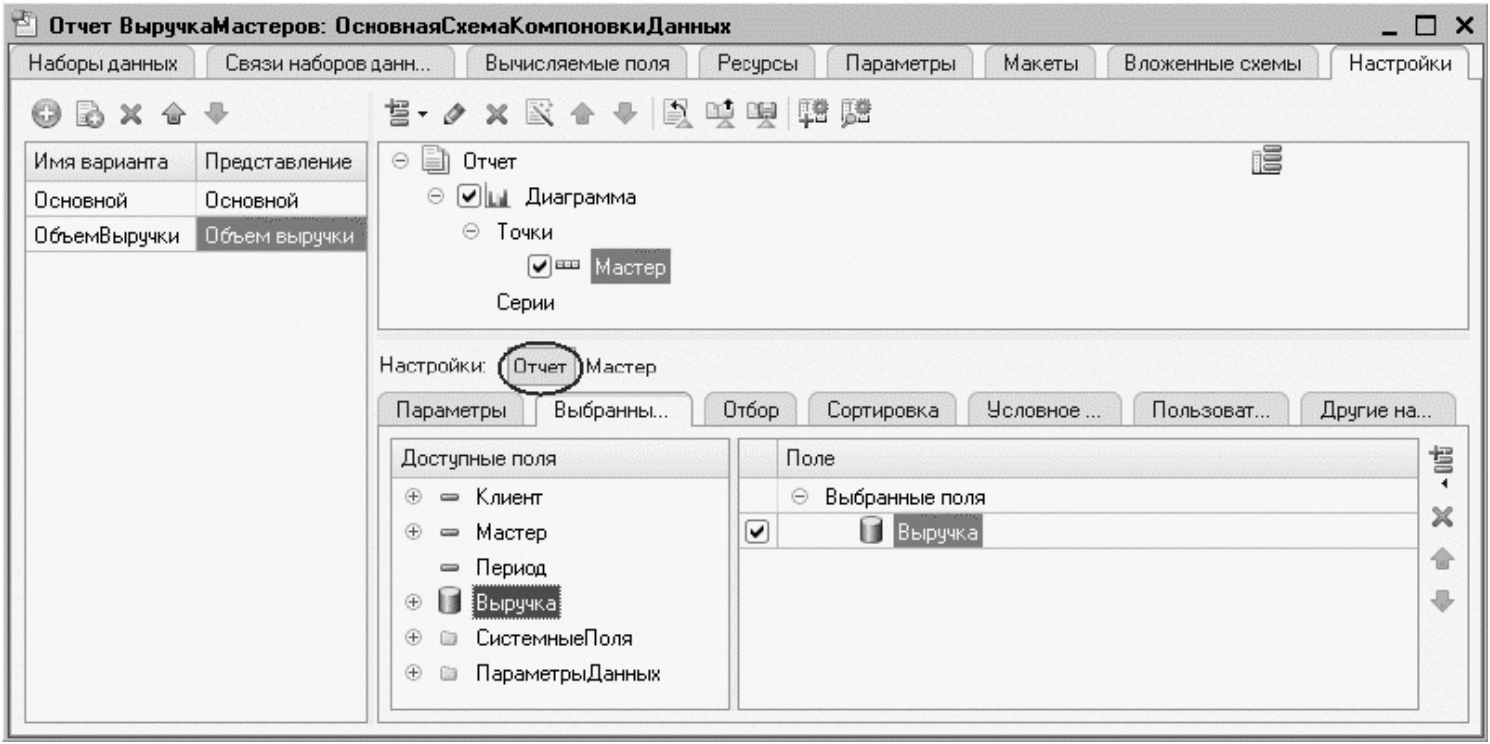


Рис. 13.78. Структура отчета и настройки диаграммы

ВНИМАНИЕ!

В диаграмме обязательно должен выводиться ресурс отчета, иначе будет получена ошибка.

На закладке *Другие настройки* выберем тип диаграммы – *Измерительная* (рис. 13.79).

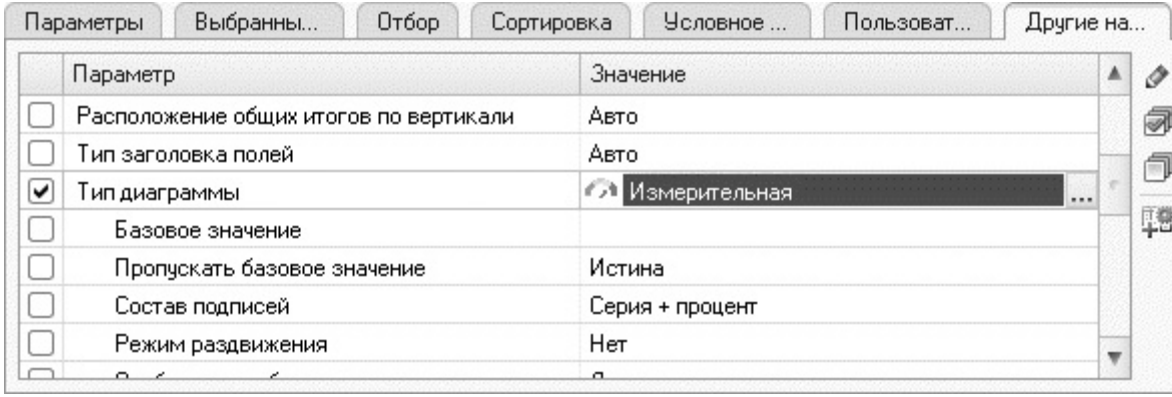


Рис. 13.79. Настройка типа диаграммы

Прокрутив вниз список свойств измерительной диаграммы, зададим ее полосы – *Плохо*, *Хорошо* и *Отлично* (рис. 13.80).

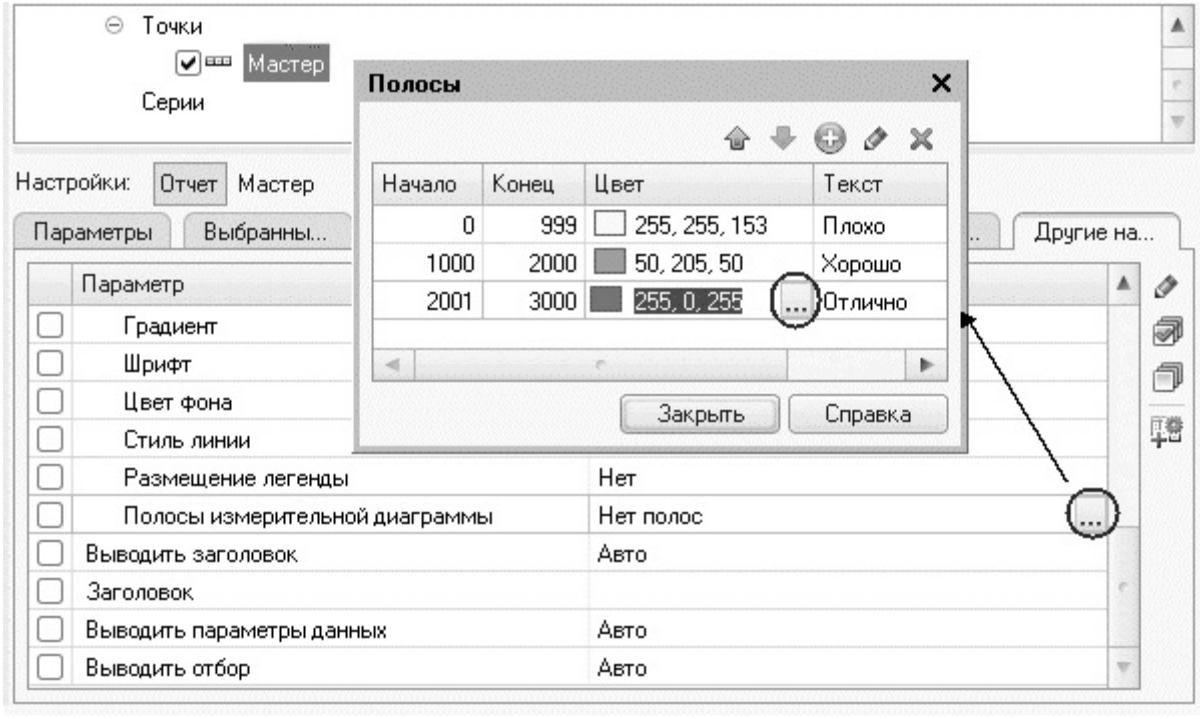


Рис. 13.80. Настройка полос измерительной диаграммы

В заключение включим параметры *Дата начала* и *Дата окончания* в состав пользовательских настроек и установим для них *Режим редактирования – Быстрый доступ*.

ВНИМАНИЕ!

Состав пользовательских настроек для каждого варианта отчета нужно

настраивать заново, поскольку у каждого варианта отчета – свои пользовательские настройки.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и выполним команду *Выручка мастеров* в панели действий раздела *Расчет зарплаты*. В открывшемся окне отчета нажмем кнопку *Выбрать вариант* (рис. 13.81).

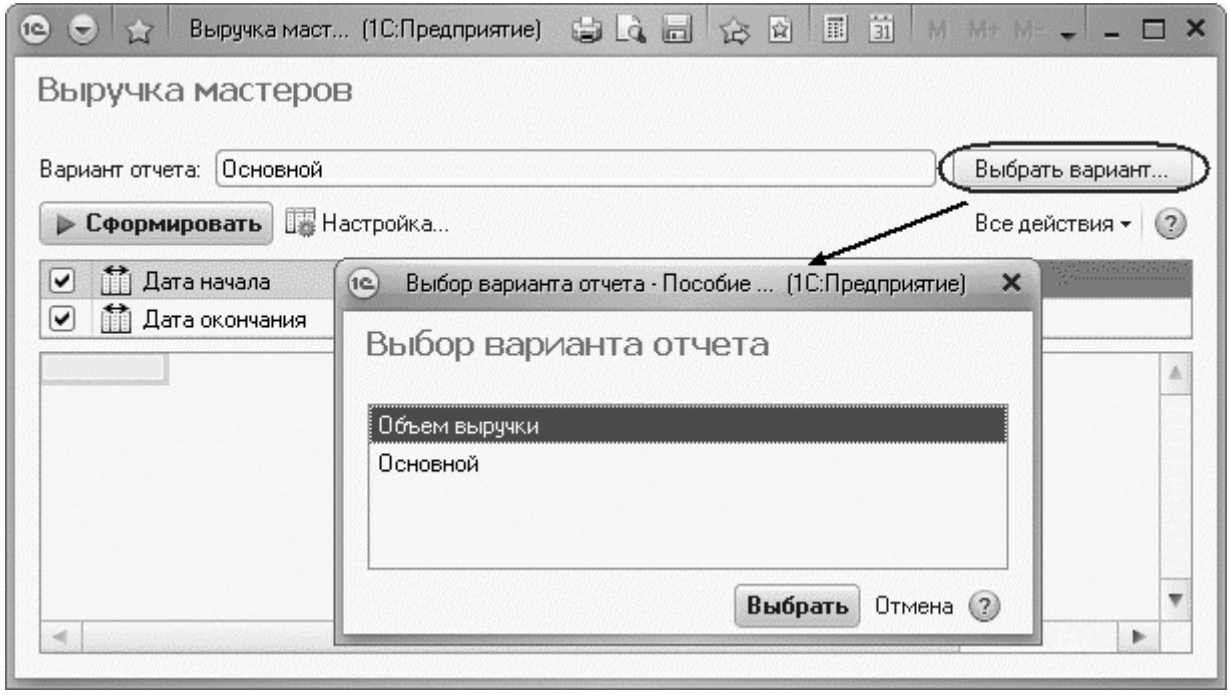


Рис. 13.81. Выбор варианта отчета

В окне вариантов отчета мы видим теперь два варианта – *Основной* и только что созданный нами вариант *Объем выручки*. Выделим его и нажмем кнопку *Выбрать*. Зададим отчетный период с *01.07.2009* по *15.07.2009* и сформируем отчет (рис. 13.82).

Выручка мастеров - 13 занятий - демонстрационна... (1С:Предприятие)

Выручка мастеров

Вариант отчета: Объем выручки | Выбрать вариант...

▶ Сформировать | Настройка... | Все действия

☑	Дата начала	01.07.2009
☑	Дата окончания	15.07.2009

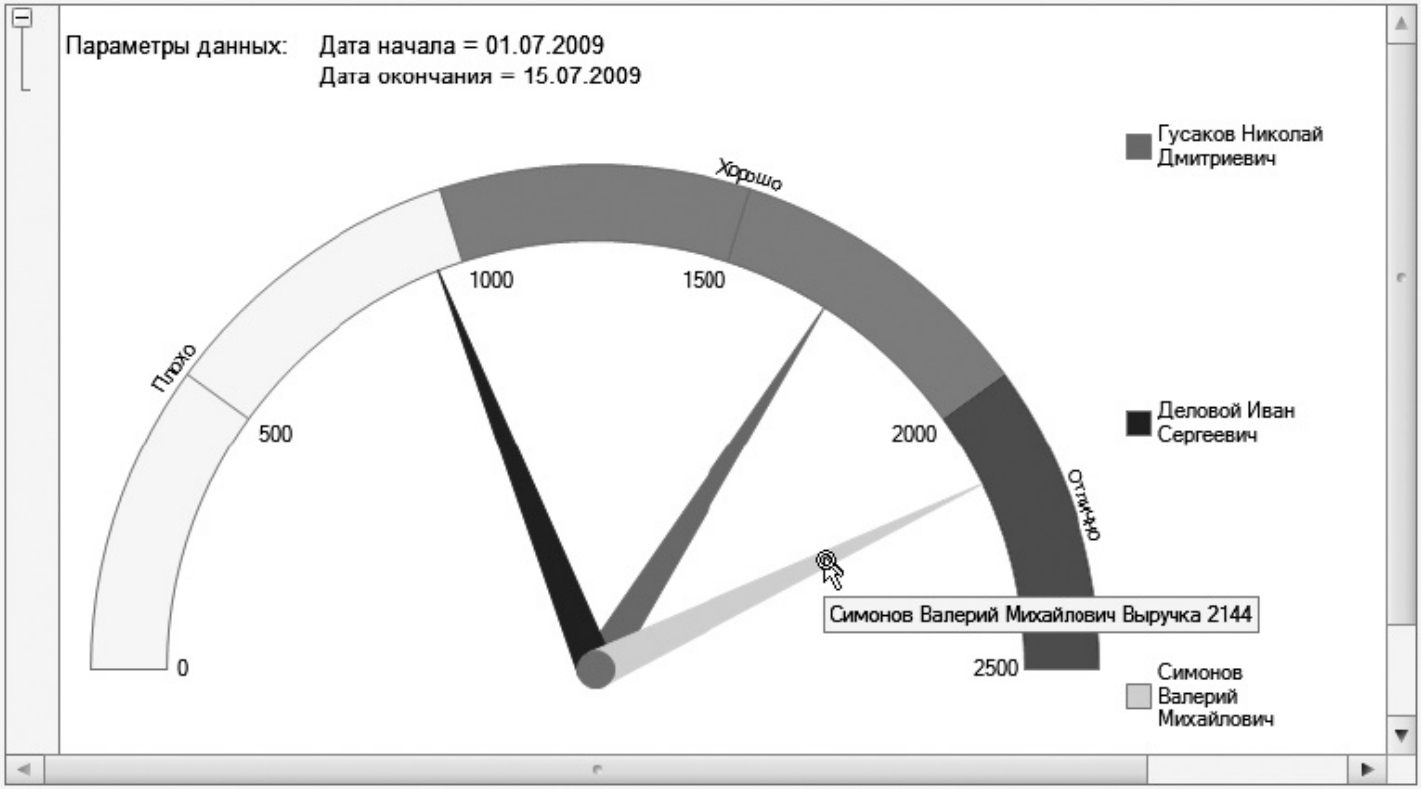


Рис. 13.82. Результат выполнения отчета

В результате мы видим те же данные, что и в основном варианте отчета, представленные в виде измерительной диаграммы. На диаграмме хорошо видна доля каждого мастера в общем объеме выручки. Обратите внимание, что при наведении курсора на стрелку диаграммы появляется подсказка.

Если же понадобится просмотреть данные о работе какого-либо мастера с разбивкой по дням и клиентам, достаточно выбрать *Основной* вариант отчета и переформировать отчет.

Таким образом, на примере отчета *Выручка мастеров* мы показали создание и использование различных вариантов отчета в целях наилучшего представления информации о работе мастеров.

Получение актуальных значений из периодического регистра сведений

Следующий отчет – *Перечень услуг* – будет содержать информацию о том, какие услуги и по какой цене оказывает ООО «На все руки мастер» (рис. 13.83).

Перечень услуг

Параметры данных: Дата отчета = 07.07.2009

Группа услуг	Услуга	Цена
Услуги		
Стиральные машины		
	Подключение воды	800,00
	Подключение электричества	800,00
Телевизоры		
	Диагностика	200,00
	Ремонт отечественного телевизора	600,00
	Ремонт импортного телевизора	800,00

Рис. 13.83. Результат отчета

На его примере мы познакомимся с возможностью получения последних значений из периодического регистра сведений и с возможностью вывода иерархических справочников.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Отчет*. Назовем его *ПеречеьУслуг* и запустим конструктор схемы компоновки данных. Добавим новый *Набор данных – запрос* и вызовем конструктор запроса.

Запрос для набора данных

В качестве источника данных для запроса выберем объектную (ссылочную) таблицу справочника *Номенклатура* и виртуальную таблицу регистра сведений *Цены.СрезПоследних*.

Для того чтобы исключить неоднозначность имен в запросе, переименуем таблицу *Номенклатура* в *СпрНоменклатура*. Для этого выделим ее в списке *Таблицы*, вызовем ее контекстное меню и выберем пункт *Переименовать таблицу*.

Параметры виртуальной таблицы

Вызовем диалог ввода параметров виртуальной таблицы *ЦеныСрезПоследних* и укажем, что период будет передан в параметре *ДатаОтчета.* Для этого выделим эту таблицу в списке *Таблицы* и нажмем кнопку *Параметры виртуальной таблицы* (рис. 13.84).

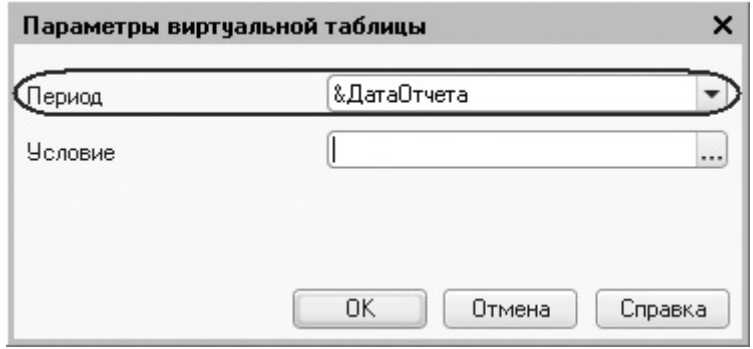


Рис. 13.84. Параметры виртуальной таблицы

Затем выберем из таблиц следующие поля (рис. 13.85):

- *СпрНоменклатура.Родитель,*
- *СпрНоменклатура.Ссылка,*
- *ЦеныСрезПоследних.Цена*.

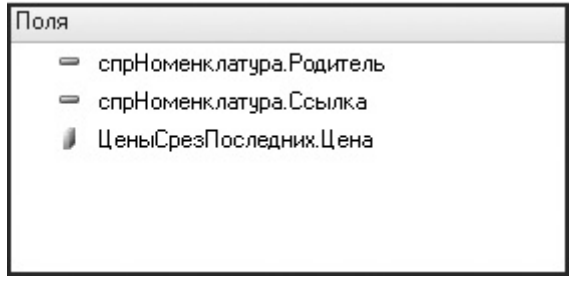


Рис. 13.85. Выбранные поля

Левое соединение таблиц

Перейдем на закладку *Связи* и укажем в поле *Условие связи*, что значение измерения *Номенклатура* регистра сведений должно быть равно ссылке на элемент справочника *Номенклатура*.

А также сбросим флажок *Все* у таблицы регистра и установим его у таблицы справочника, тем самым установив вид связи как левое соединение для таблицы справочника (рис. 13.86).

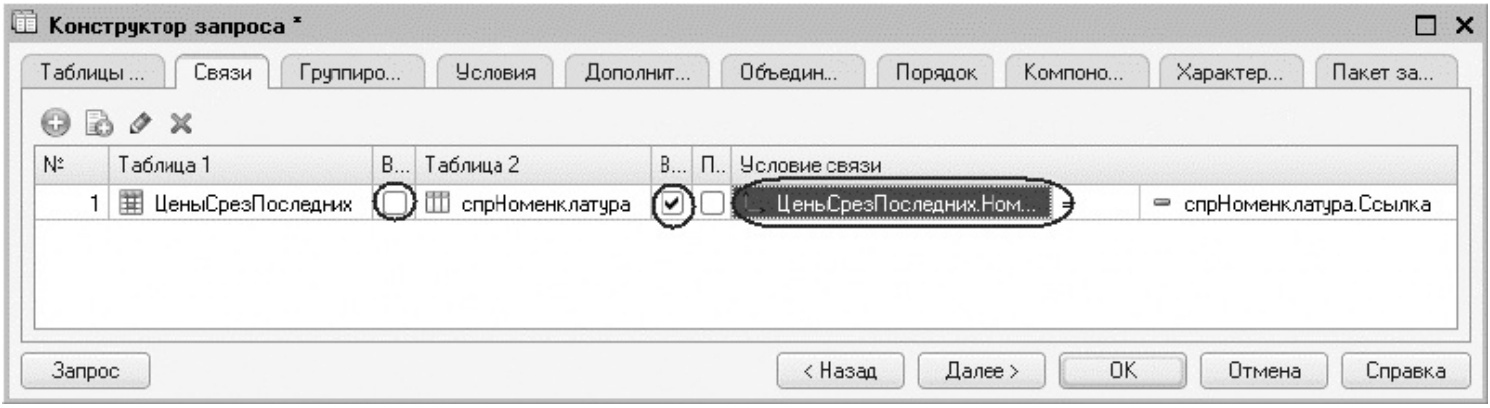


Рис. 13.86. Связь таблиц в запросе

На закладке *Условия* зададим условие выбора элементов справочника *Номенклатура* – выбираемые элементы должны соответствовать виду номенклатуры, переданному в параметре запроса *Вид Номенклатуры* (рис. 13.87).

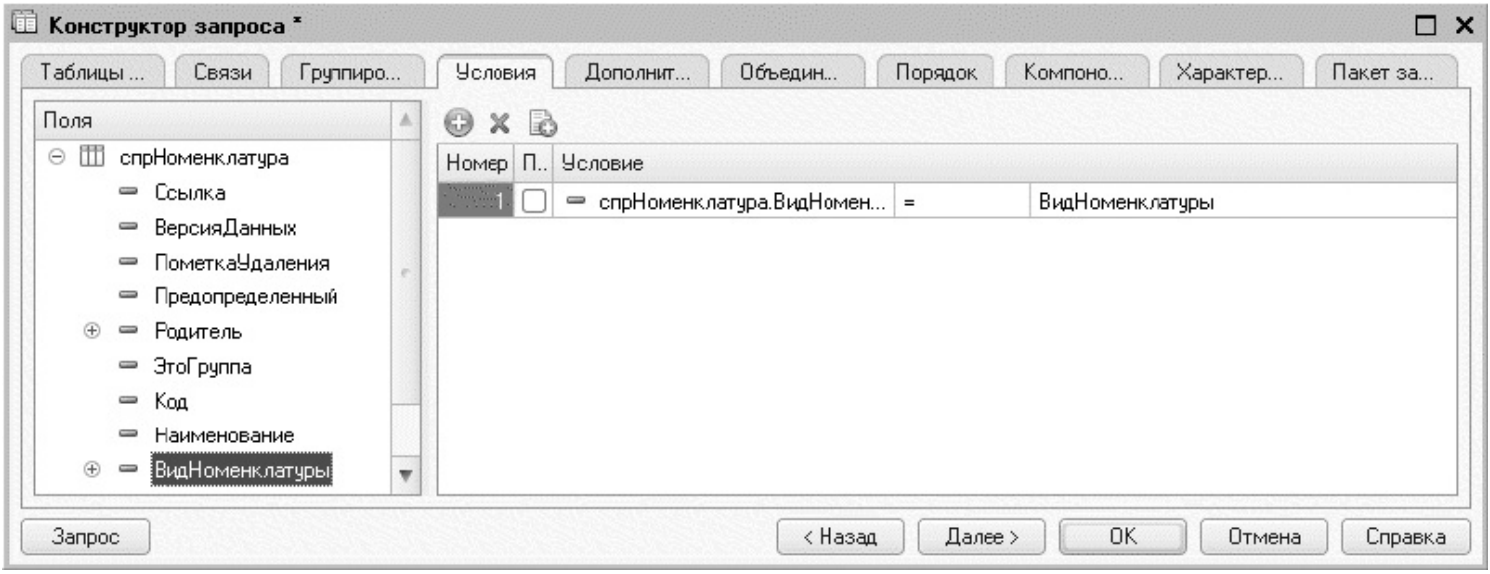


Рис. 13.87. Условия выбора элементов

Псевдонимы полей

На закладке *Объединения/Псевдонимы* укажем, что поле *Родитель* будет иметь псевдоним *ГруппаУслуг,* а поле *Ссылка – Услуга* (рис. 13.88).

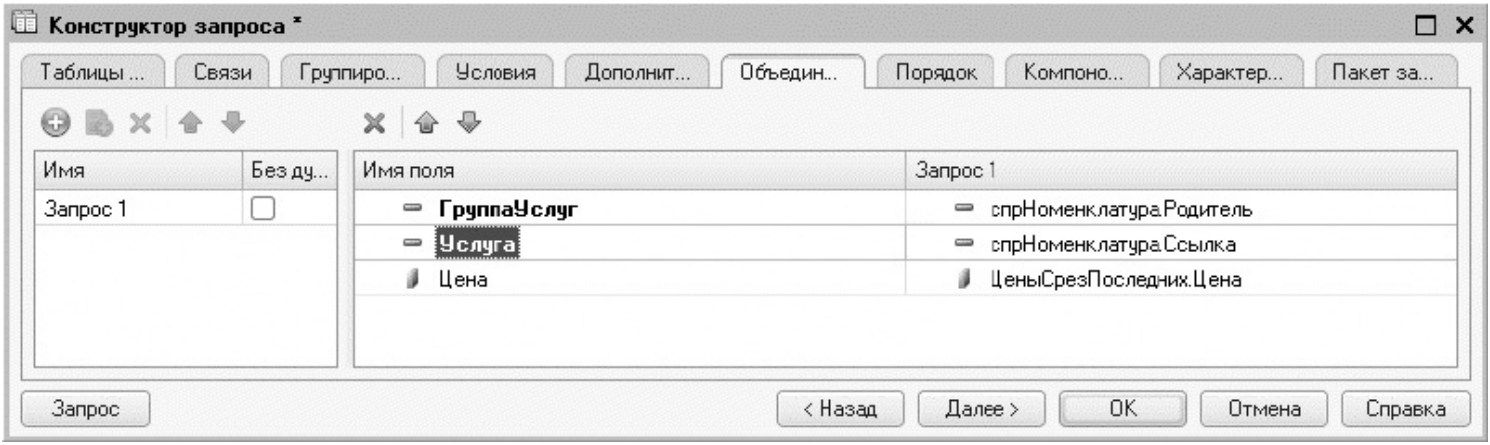


Рис. 13.88. Объединения/Псевдонимы

На этом создание запроса завершено, нажмем *OK*.

Анализ текста запроса

Теперь рассмотрим текст запроса, сформированный конструктором (листинг 13.13).

Листинг 13.13. Текст запроса

```
ВЫБРАТЬ
  СпрНоменклатура.Родитель КАК ГруппаУслуг,
  СпрНоменклатура.Ссылка КАК Услуга,
  ЦеныСрезПоследних.Цена
ИЗ
  Справочник.Номенклатура КАК СпрНоменклатура
```

```
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних(&ДатаОтчета, ) КАК
    ЦеныСрезПоследних
    ПО ЦеныСрезПоследних.Номенклатура = СпрНоменклатура.Ссылка
ГДЕ
  СпрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры
```

Практически все конструкции, использованные в этом запросе, нам уже известны. Перейдем к редактированию схемы компоновки данных. На закладке *Ресурсы* нажатием кнопки  выберем единственный доступный ресурс – *Цена*.

Параметры

На закладке *Параметры* зададим значение параметра *ВидНоменклатуры* как *Перечисление.ВидыНоменклатуры.Услуга*.

Кроме этого, снимем ограничение доступности для параметра *ДатаОтчета*.

В поле *Тип* этого параметра зададим состав даты – *Дата*. Для параметра *Период,* наоборот, установим ограничение доступности (рис. 13.89).

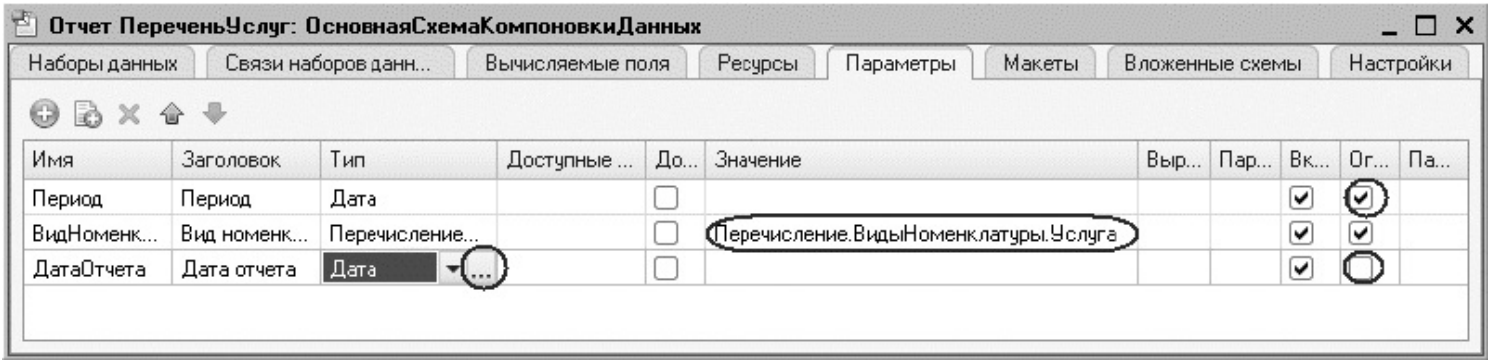


Рис. 13.89. Параметры схемы компоновки

Настройки

Приступим к созданию структуры отчета. Перейдем на закладку *Настройки* и создадим группировку по полю *ГруппаУслуг*, указав тип группировки *Иерархия* (рис. 13.90).

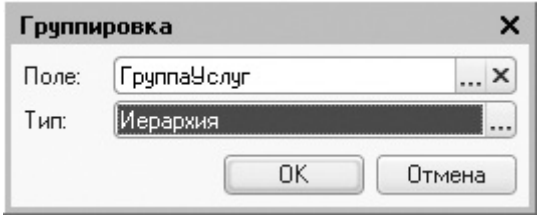


Рис. 13.90. Выбор поля и типа группировки

ПРИМЕЧАНИЕ

*До сих пор мы использовали тип иерархии по умолчанию – **Без иерархии**.*

Существуют следующие типы иерархии для группировок отчета:

- Без иерархии – в группировке выводятся только неиерархические записи;

- Иерархия – в группировке выводятся как неиерархические, так и иерархические записи;

- Только иерархия – в группировке выводятся только иерархические (родительские) записи.

Внутри этой группировки создадим еще одну группировку без указания группового поля. Она будет содержать детальные записи отчета.

Перейдем на закладку *Выбранные поля* и укажем, что в отчет будут выводиться поля *Услуга* и *Цена* (рис. 13.91).

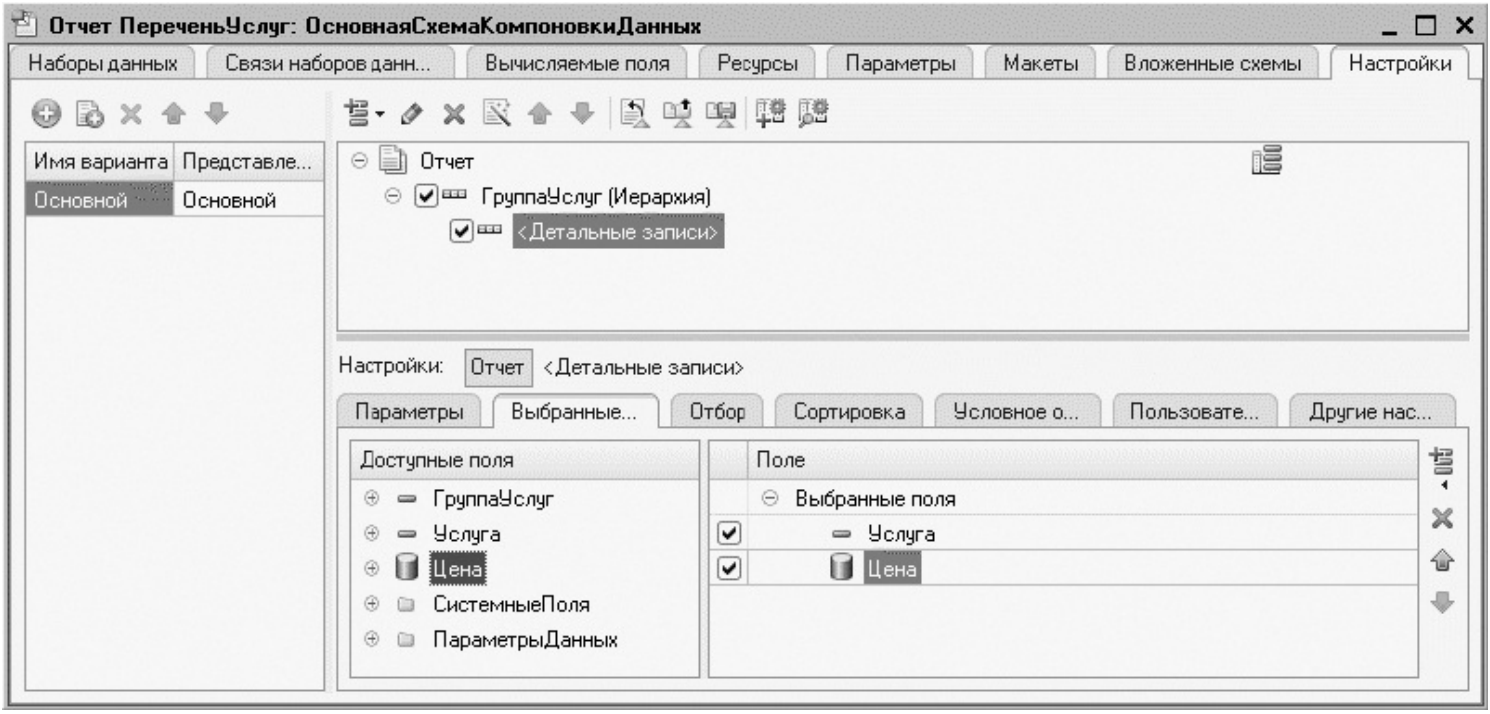


Рис. 13.91. Структура и поля отчета

Теперь настроим внешний вид отчета на закладке *Другие настройки*.

Так как наш отчет будет представлять собой просто список оказываемых услуг, в котором интересны цены на конкретные услуги, выводить значения ресурса *Цена* для каждой из группировок и для всего отчета в целом не имеет смысла. Запретим вывод общих итогов в группировке *ГруппаУслуг* и в отчете в целом.

Сначала перейдем к настройкам конкретной группировки – *ГруппаУслуг*. Для параметра *Расположение итогов* этой группировки укажем значение *Нет* (рис. 13.92).

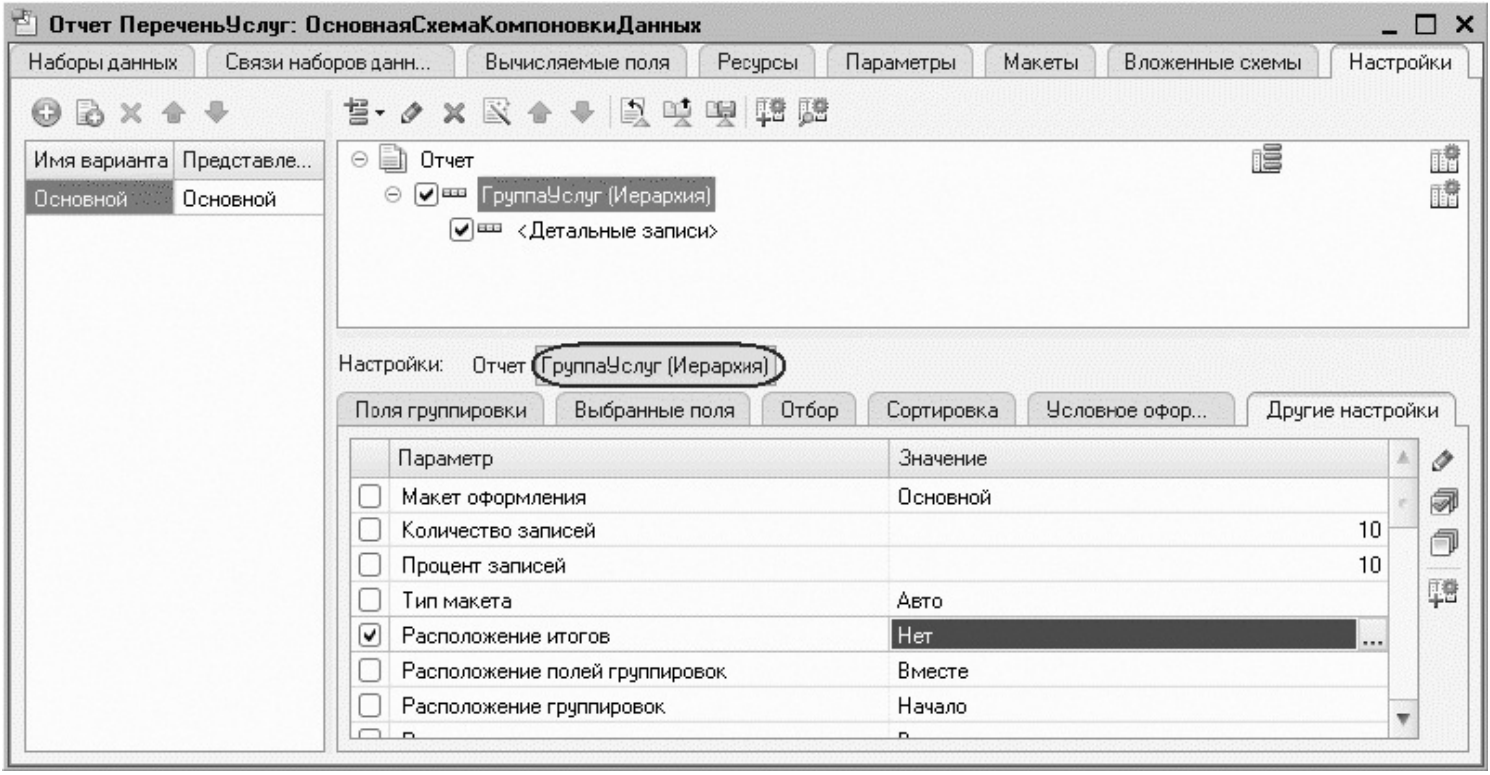


Рис. 13.92. Настройка вывода общих итогов для группировки «ГруппаУслуг»

Вернемся к настройкам всего отчета в целом. Чтобы запретить вывод общих

итогов в отчете, установим параметр *Расположение общих итогов по вертикали* в значение *Нет* (см. рис. 13.93).

Для параметра *Расположение полей группировок* укажем значение *Отдельно и только в итогах* (так наш отчет будет лучше читаться). Также зададим заголовок отчета – *Перечень услуг*.

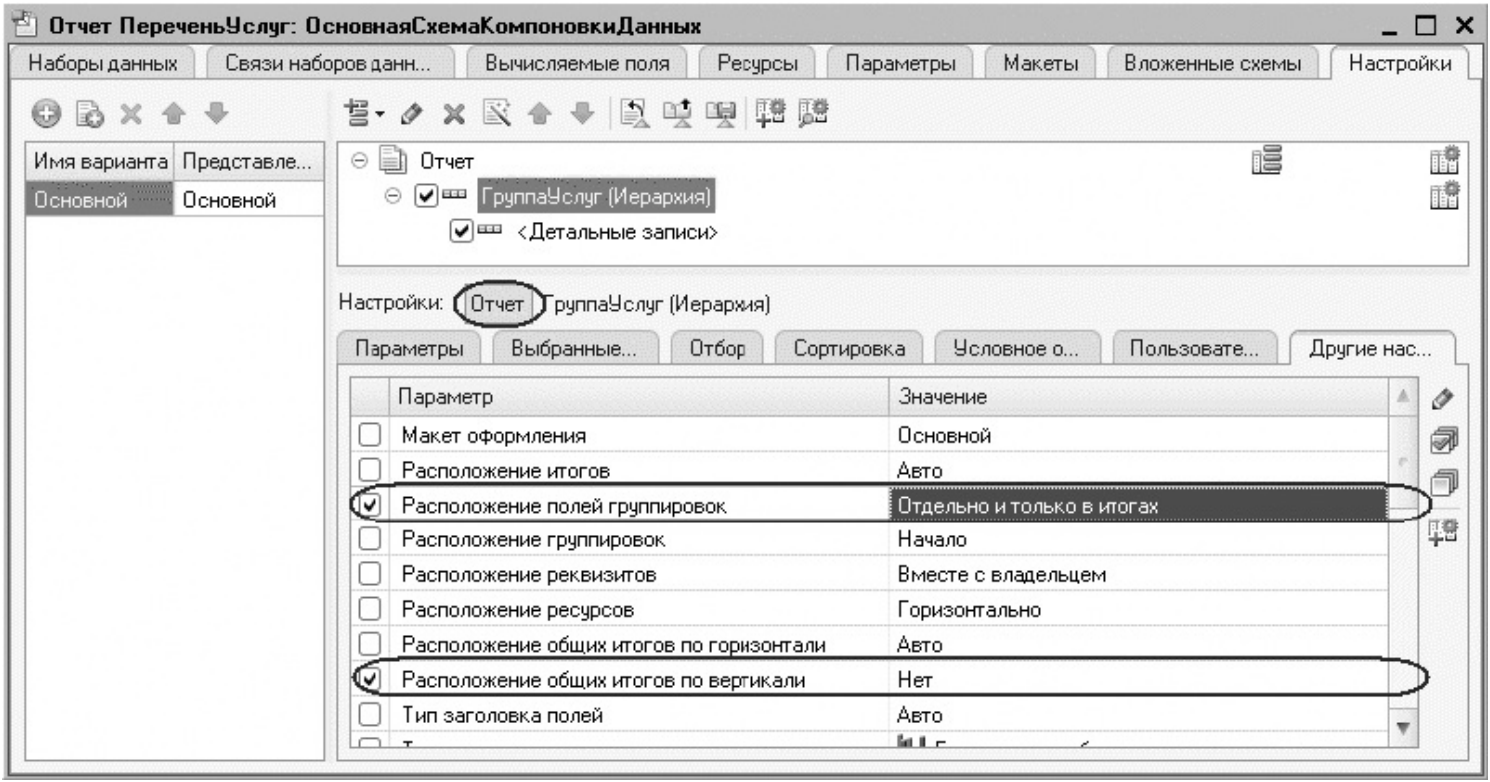


Рис. 13.93. Настройки вывода итогов для глобального отчета

В заключение включим параметр *Дата отчета* в состав пользовательских настроек и установим для него *Режим редактирования – Быстрый доступ*. Также определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации *Отчет ПеречеьУслуг* перейдем на закладку *Подсистемы*. Отметим в списке подсистем конфигурации подсистемы *Оказание услуг* и *Бухгалтерия*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и прежде всего откроем периодический регистр *Цены*.

Добавим в него еще одно значение для услуги *Диагностика*: новая цена услуги на *10.07.2009 – 350* (рис. 13.94). Это позволит нам протестировать отчет.

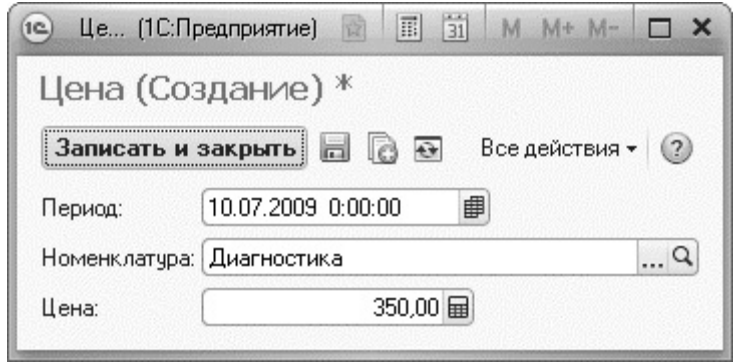


Рис. 13.94. Добавление новой записи регистра «Цены» для услуги «Диагностика»

Теперь выполним отчет *Перечень услуг* по состоянию на *07.07.2009* (рис. 13.95).

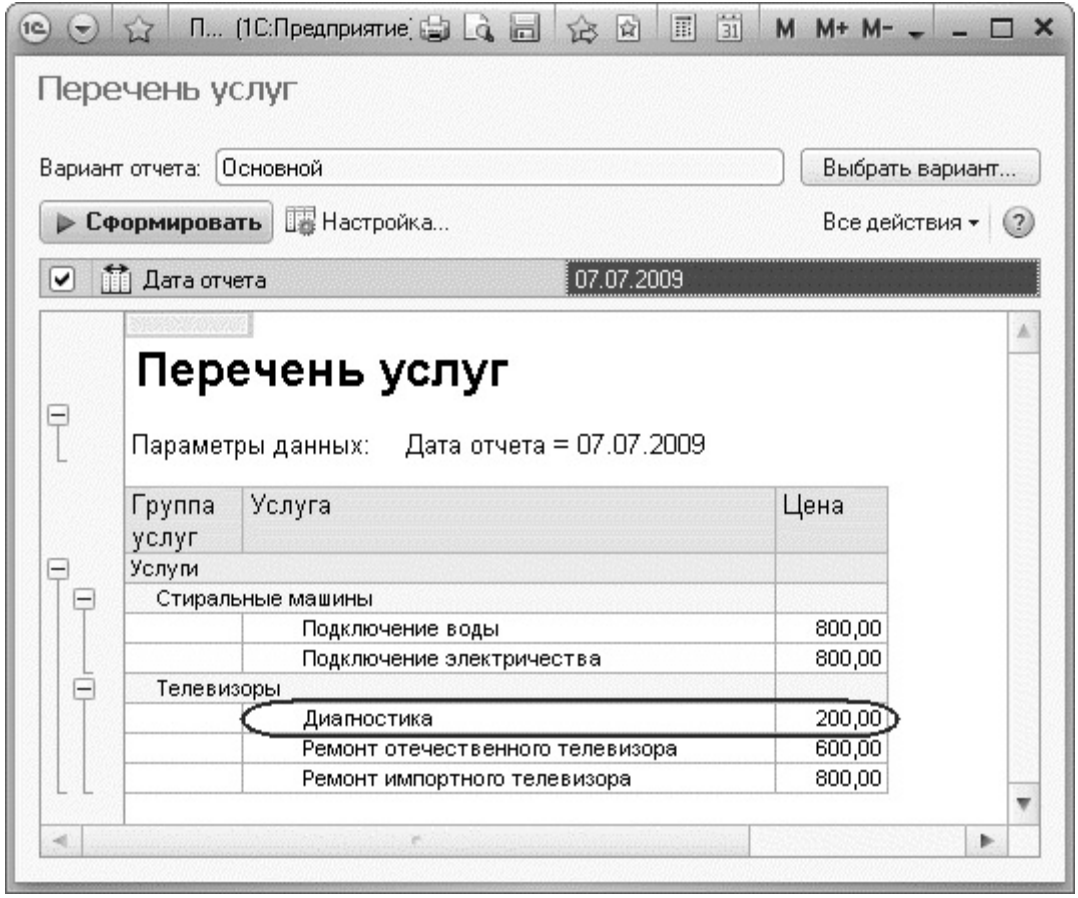


Рис. 13.95. Результат выполнения отчета

Наш отчет правильно отражает цену услуги *Диагностика* на *07.07.2009 – 200* руб.

Еще раз выполним отчет, но теперь уже на другую дату – *10.07.2009* (рис. 13.96).

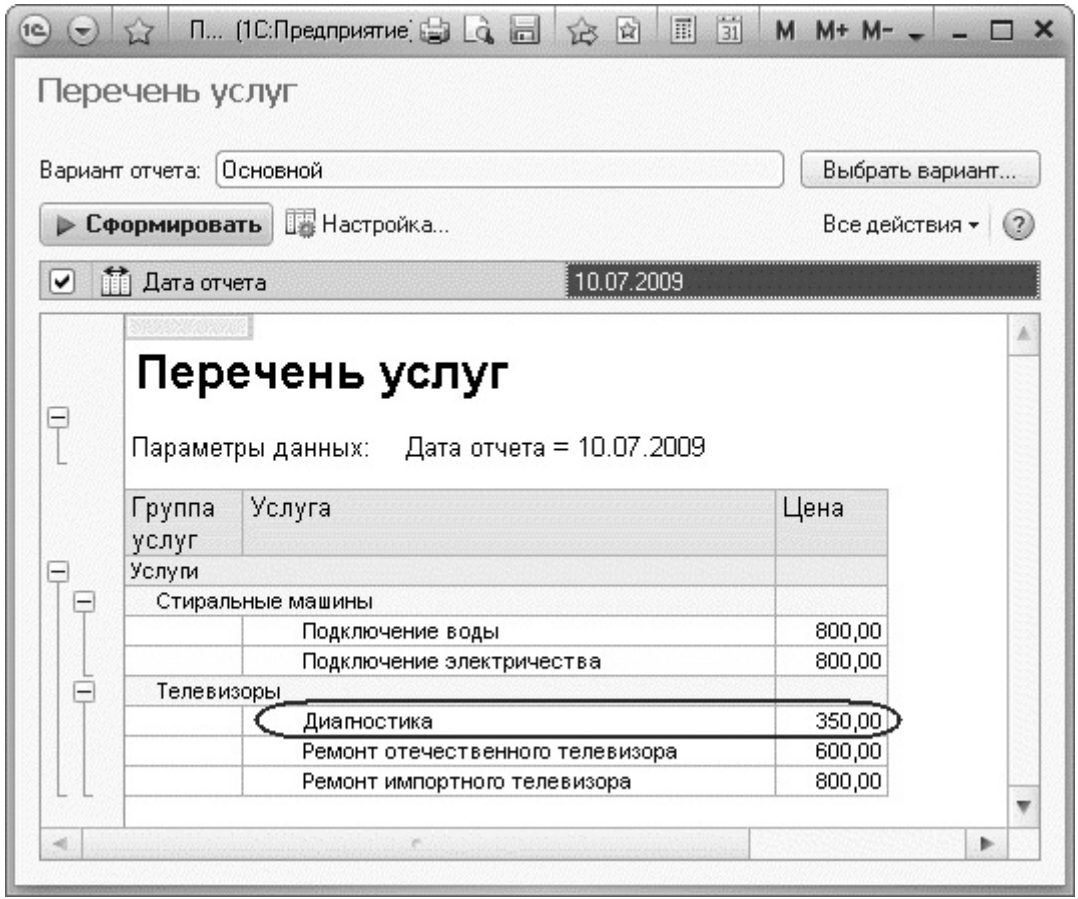


Рис. 13.96. Результат выполнения отчета

Как видите, показана новая цена услуги *Диагностика – 350* руб.

Таким образом, на примере этого отчета мы показали, как система компоновки данных получает последние значения из периодического регистра сведений и как вывести группировки по иерархии справочника.

Использование вычисляемого поля в отчете

Следующий отчет – *Рейтинг клиентов* – будет показывать в графическом виде, каков доход от оказания услуг каждому из клиентов за все время работы ООО «На все руки мастер» (рис. 13.97).

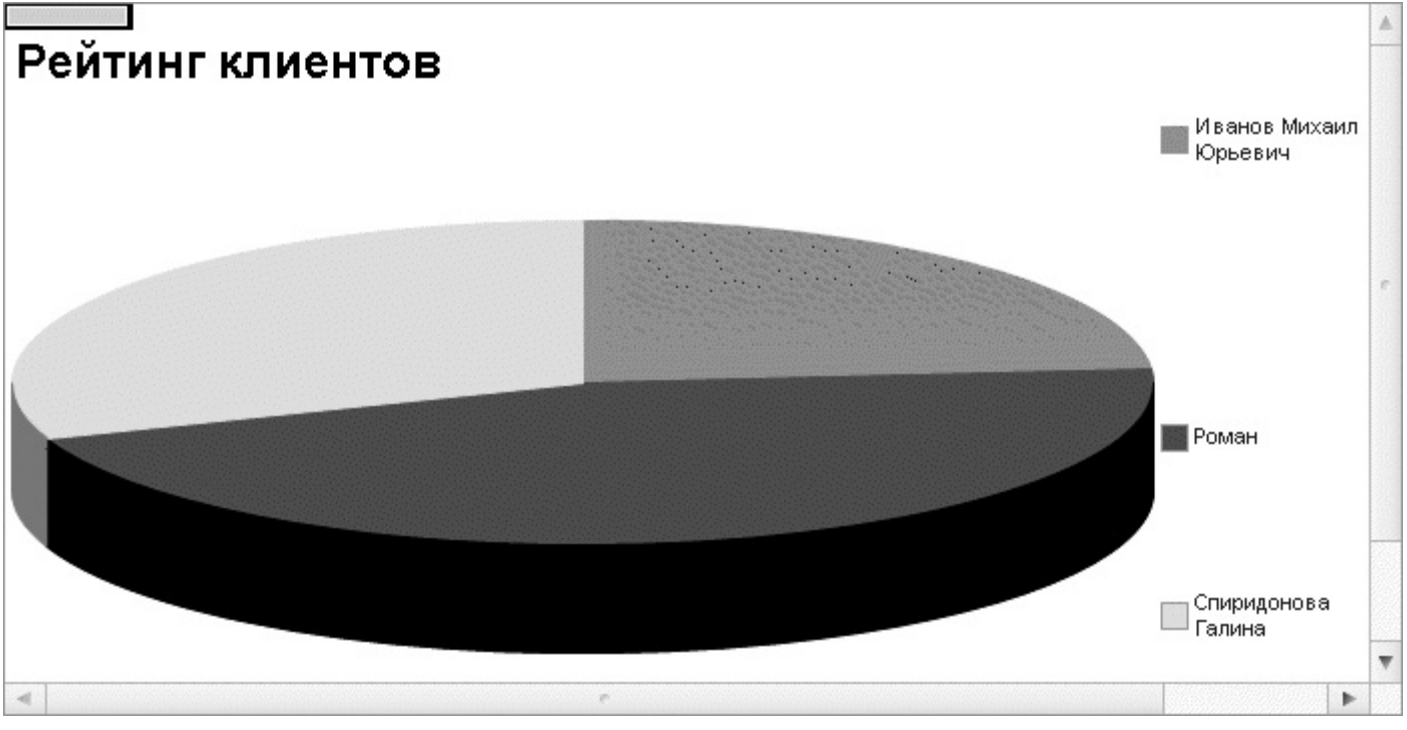


Рис. 13.97. Результат отчета

На его примере мы продемонстрируем возможность использования вычисляемого поля и вывод результата в виде круговой диаграммы и в виде гистограммы.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Отчет*. Назовем его *РейтингКлиентов*

и запустим конструктор схемы компоновки данных. Создадим новый *Набор данных – запрос* и вызовем конструктор запроса.

Запрос для набора данных

В качестве источника данных для запроса выберем виртуальную таблицу регистра накопления *Продажи.Обороты*. Затем выберем из нее следующие поля (рис. 13.98):

- *ПродажиОбороты.Клиент,*
- *ПродажиОбороты.ВыручкаОборот,*
- *ПродажиОбороты.СтоимостьОборот.*

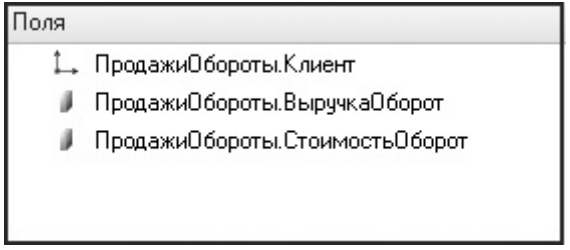


Рис. 13.98. Выбранные поля

На закладке *Объединения/Псевдонимы* укажем, что поле *ВыручкаОборот* будет иметь псевдоним *Выручка,* а поле *СтоимостьОборот – Стоимость*. На этом создание запроса завершено, нажмем *OK*.

Ничего нового и непонятного в этом запросе для нас нет, поэтому не будем его подробно рассматривать. Перейдем к редактированию схемы компоновки данных.

Вычисляемые поля

На этом этапе мы столкнулись с необходимостью отразить в отчете поле, которого нет в наборе данных. Раньше мы использовали в отчете те поля, которые описывались в наборе данных. Теперь, чтобы отобразить доход от оказания услуг в разрезе клиентов, нам необходимо дополнительное поле, рассчитанное как разница между выручкой и стоимостью оказания услуг.

Для этого в системе компоновки данных есть возможность определения вычисляемого поля.

Вычисляемые поля представляют собой дополнительные поля схемы компоновки данных, значения которых будут вычисляться по некоторой формуле.

Перейдем на закладку *Вычисляемые поля* схемы компоновки данных и, нажав кнопку *Добавить*, добавим вычисляемое поле.

Дадим ему имя (*Путь к данным*) – *Доход*, в колонку *Выражение* введем выражение для расчета вычисляемого поля (листинг 13.14).

Листинг 13.14. Выражение для расчета вычисляемого поля «Доход»

```
Выручка - Стоимость
```

Заголовок вычисляемого поля, который будет отображаться в шапке отчета, задается по умолчанию, но можно его изменить (рис. 13.99).

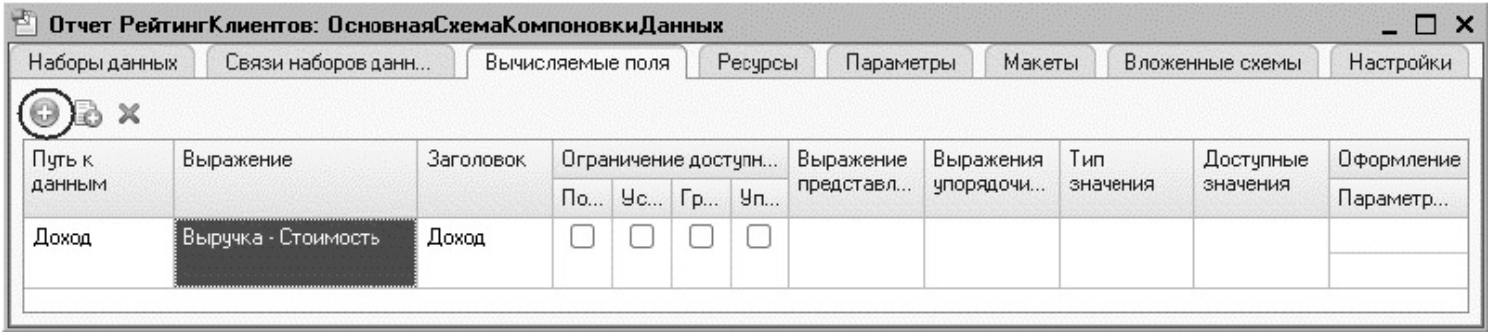


Рис. 13.99. Создание вычисляемого поля

Вычисляемое поле можно добавить в ресурсы отчета, чтобы вычислять по нему групповые и общие итоги.

Ресурсы

На закладке *Ресурсы* нажатием кнопки >> выберем все доступные ресурсы отчета. Как мы видим, вычисляемое поле *Доход* также добавилось в список ресурсов (рис. 13.100).

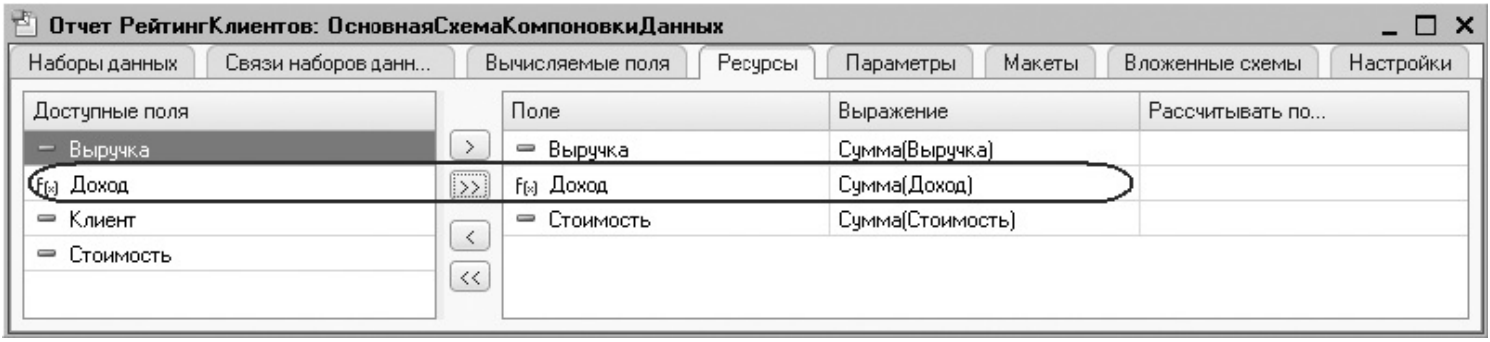


Рис. 13.100. Ресурсы схемы компоновки данных

Настройки

На закладке *Настройки* добавим в структуру отчета диаграмму. Для этого нажмем кнопку *Добавить* в командной панели окна настроек и добавим диаграмму (рис. 13.101).

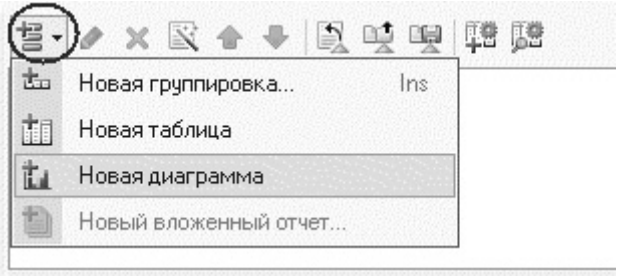


Рис. 13.101. Добавление диаграммы в структуру отчета

Затем выделим ветку *Точки* и добавим в нее группировку по полю *Клиент*.

Серии диаграммы оставим без изменений.

Дело в том, что для демонстрации рейтинга клиентов хорошо подойдет круговая диаграмма, которую мы хотим показать. Для этого вида диаграммы достаточно задать только точки, поэтому серии мы не задаем.

В значения диаграммы всегда выводится один из ресурсов отчета. Перейдем на закладку *Выбранные поля* и выберем поле *Доход* для вывода в отчет.

Структура отчета должна принять следующий вид (рис. 13.102).

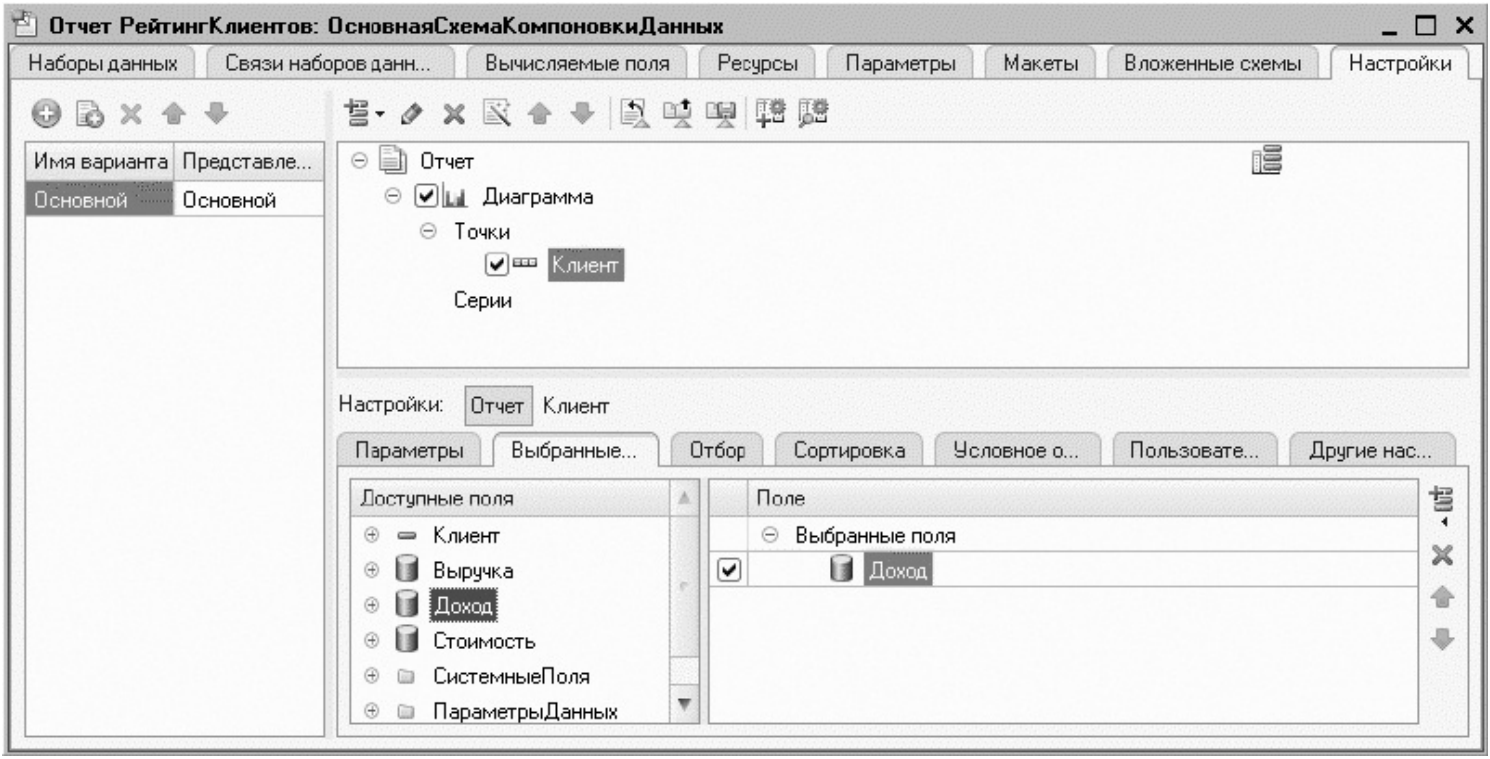


Рис. 13.102. Структура отчета и настройки диаграммы

На закладке *Другие настройки* выберем тип диаграммы – *Круговая объемная* и зададим заголовок отчета – *Рейтинг клиентов*.

В заключение определим, в каких подсистемах будет отображаться наш отчет. В окне редактирования объекта конфигурации Отчет *РейтингКлиентов*

перейдем на закладку *Подсистемы*. Отметим в списке подсистем конфигурации подсистемы *Оказание услуг* и *Бухгалтерия*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и выполним команду *Рейтинг клиентов* в панели действий раздела *Бухгалтерия*. Нажмем *Сформировать*.

Мы видим данные о доходе от оказания услуг по каждому из клиентов, представленные в виде круговой диаграммы (рис. 13.103).

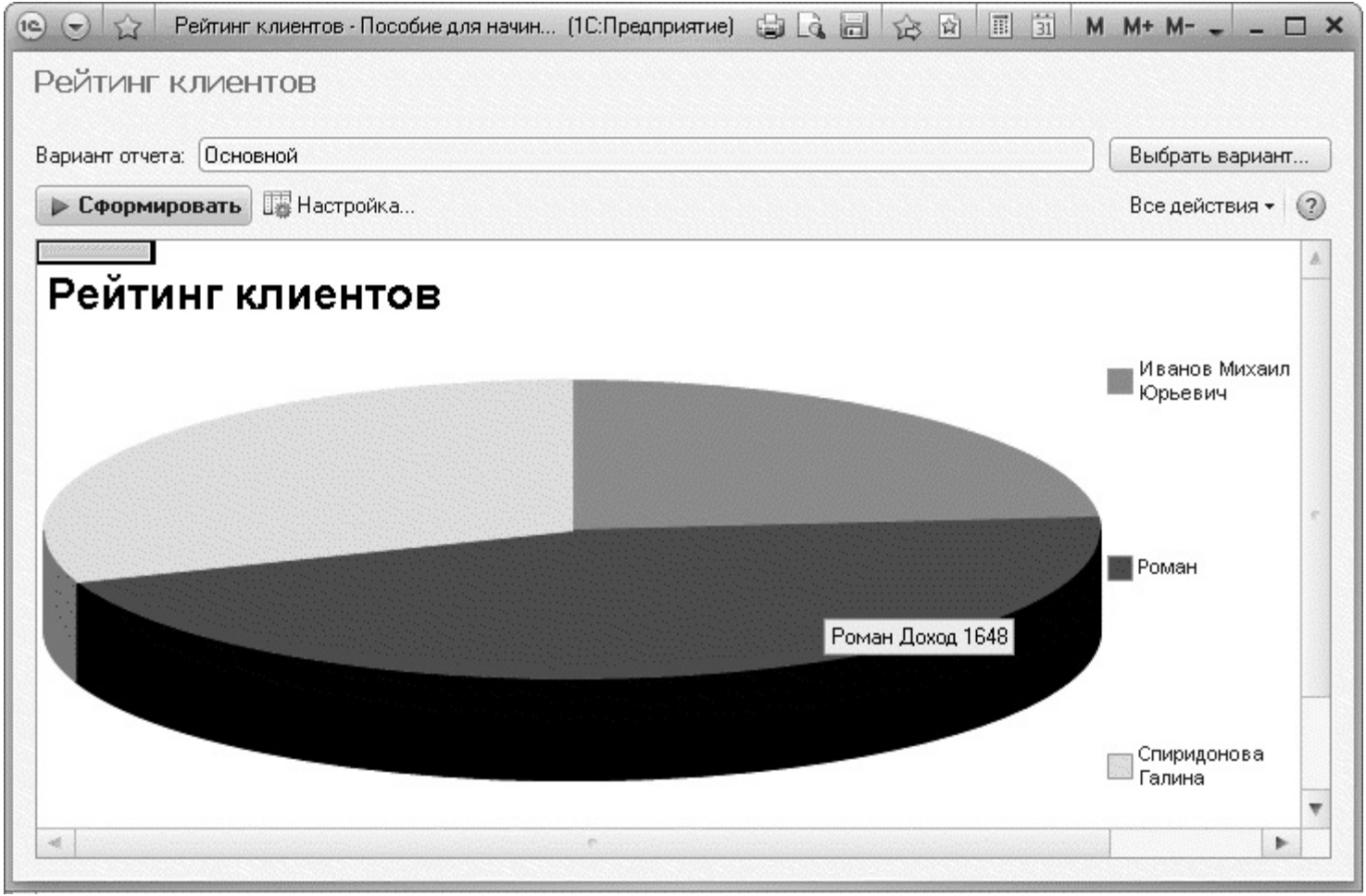


Рис. 13.103. Круговая объемная диаграмма

Вернемся теперь в конфигуратор и изменим тип диаграммы на *Гистограмма*

объемная. Заново сформируем отчет (рис. 13.104).

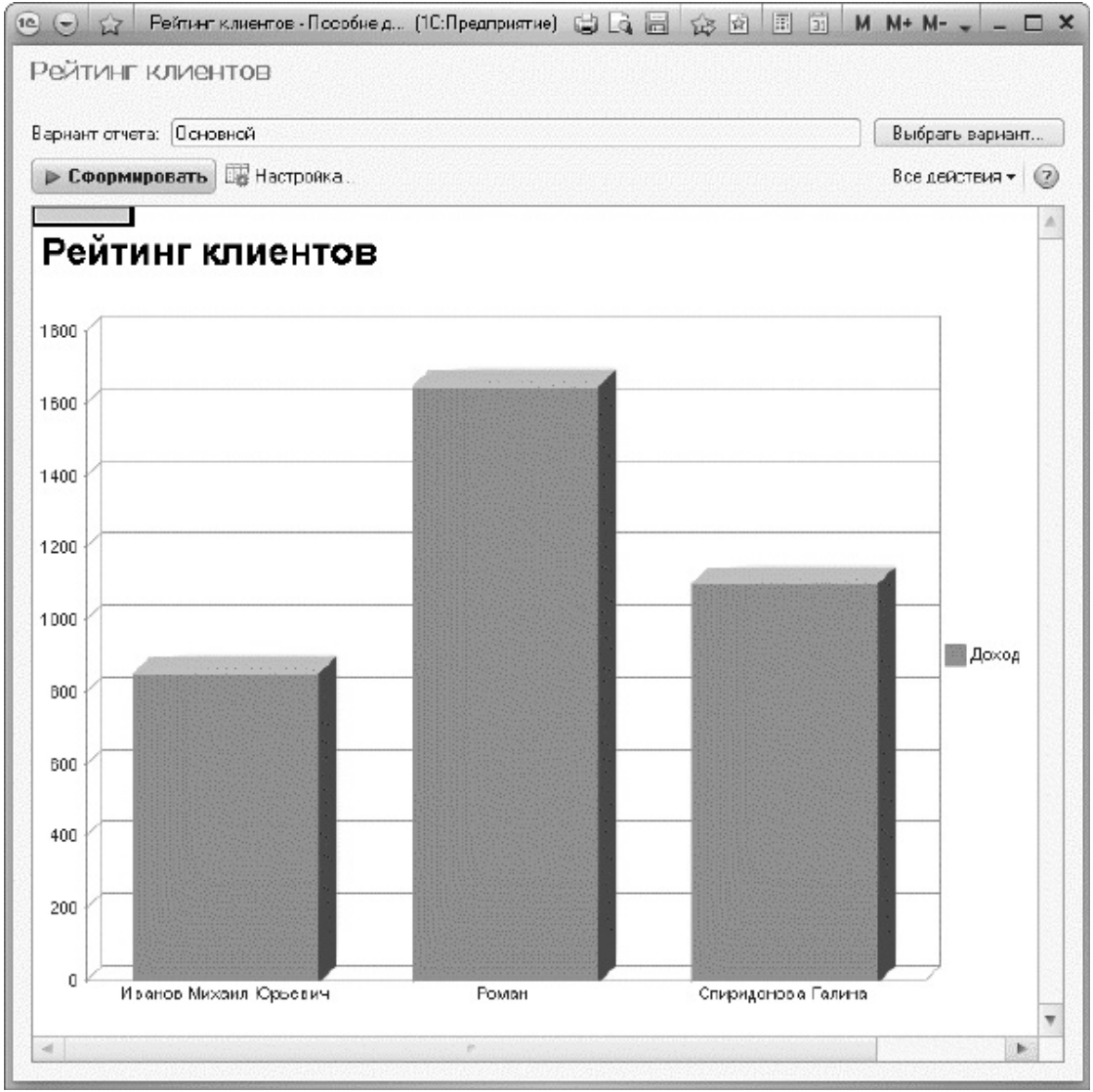

Рейтинг клиентов - Пособие д... (1С:Предприятие)
Рейтинг клиентов
Вариант отчета: Основной
Выбрать вариант...
Сформировать
Настройка...
Все действия
Рейтинг клиентов
1800
1600
1400
1200
1000
800
600
400
200
0
Иванов Михаил Юрьевич
Роман
Спиридонова Галина
Доход

Рис. 13.104. Гистограмма объемная

Таким образом, мы продемонстрировали, как можно использовать различные виды диаграмм для визуализации данных отчета.

Вывод данных в таблицу

На примере создания универсального отчета мы продемонстрируем вывод данных в таблицу (рис. 13.105).

Номенклатура	Гусаков Николай Дмитриевич	Деловой Иван Сергеевич	Симонов Валерий Михайлович	Итого
	Выручка Оборот	Выручка Оборот	Выручка Оборот	Выручка Оборот
Материалы	900,00	150,00	744,00	1 794,00
Прочее		150,00	330,00	480,00
Кабель электрический			30,00	30,00
Шланг резиновый		150,00	300,00	450,00
Радиодетали	900,00		414,00	1 314,00
Строчный трансформатор GoldStar			400,00	400,00
Строчный трансформатор Samsung	900,00			900,00
Транзистор Philips 2N2369			14,00	14,00
Услуги	800,00	800,00	1 400,00	3 000,00
Стиральные машины		800,00	800,00	1 600,00
Подключение воды		800,00		800,00
Подключение электричества			800,00	800,00
Телевизоры	800,00		600,00	1 400,00
Ремонт импортного телевизора	800,00			800,00
Ремонт отечественного телевизора			600,00	600,00
Итого	1 700,00	950,00	2 144,00	4 794,00

Рис. 13.105. Результат отчета

Мы покажем, как сделать отчет максимально универсальным, чтобы позволить пользователю в режиме *1С:Предприятие*, не обращаясь к полным настройкам отчета (не выполняя *Все действия > Изменить вариант*), изменять его структуру и внешний вид. Например, поменять местами строки и колонки таблицы или изменить данные, выводящиеся в ячейках таблицы.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Отчет*. Назовем его *Универсальный* и запустим конструктор схемы компоновки данных. Создадим новый *Набор данных – запрос* и вызовем конструктор запроса.

Запрос для набора данных

В качестве источника данных для запроса выберем виртуальную таблицу регистра накопления *Продажи.Обороты*. Затем выберем из нее все поля (рис. 13.106).

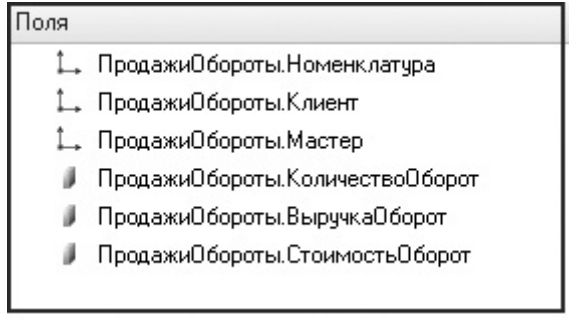


Рис. 13.106. Выбранные поля

Анализ текста запроса

Нажмем *OK* и посмотрим на текст, сформированный конструктором запроса (листинг 13.15).

Листинг 13.15. Текст запроса

```
ВЫБРАТЬ
  ПродажиОбороты.Номенклатура,
  ПродажиОбороты.Клиент,
  ПродажиОбороты.Мастер,
  ПродажиОбороты.КоличествоОборот,
  ПродажиОбороты.ВыручкаОборот,
  ПродажиОбороты.СтоимостьОборот
ИЗ
  РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
```

Ресурсы

На закладке *Ресурсы* нажатием кнопки » выберем все доступные ресурсы отчета.

Настройки

На закладке *Настройки* добавим в структуру отчета таблицу. Для этого нажмем кнопку *Добавить* в командной панели окна настроек и добавим таблицу (рис. 13.107).

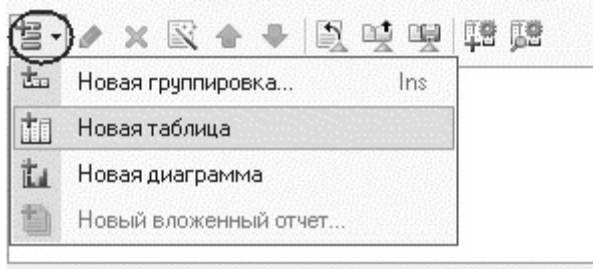


Рис. 13.107. Добавление таблицы в структуру отчета

Мы не будем здесь задавать строки и колонки этой таблицы, а также список выбранных полей, так как хотим предоставить полную свободу пользователю в этих действиях. Для этого выделим в структуре элементов отчета элемент *Таблица* и нажмем кнопку *Свойства элемента пользовательских настроек*, расположенную вверху в командной панели окна настроек.

В появившемся окне мы можем редактировать состав пользовательских настроек таблицы.

Установим признак использования для настроек *Выбранные поля, Группировки строк* и *Группировки колонок* и оставим для них по умолчанию свойство *Режим редактирования* в значении *Быстрый доступ* (рис. 13.108).

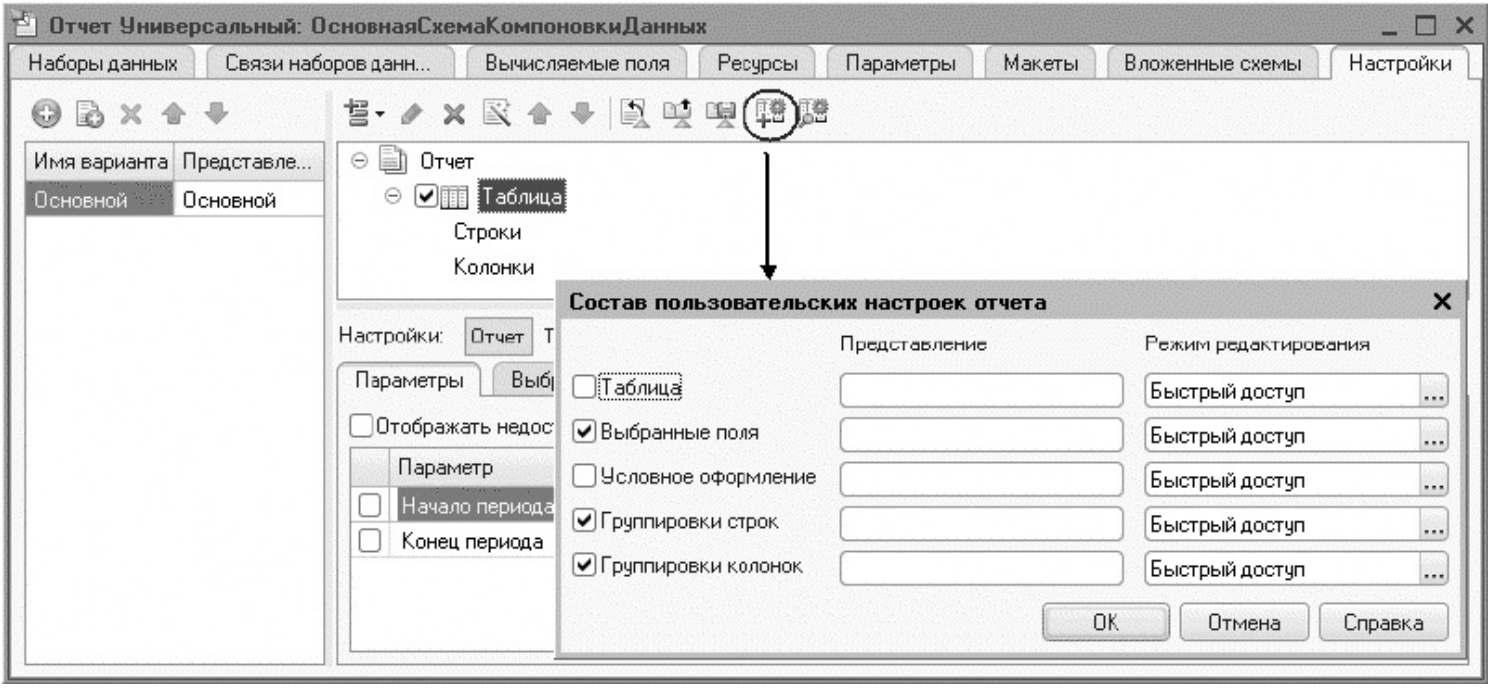


Рис. 13.108. Состав пользовательских настроек

Таким образом, мы предоставили пользователю возможность самостоятельно определять состав выбранных полей, группировок строк и колонок таблицы непосредственно в отчетной форме перед формированием отчета.

В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования

объекта конфигурации Отчет *Универсальный* перейдем на закладку *Подсистемы*. Отметим в списке подсистем конфигурации подсистему *Оказание услуг*.

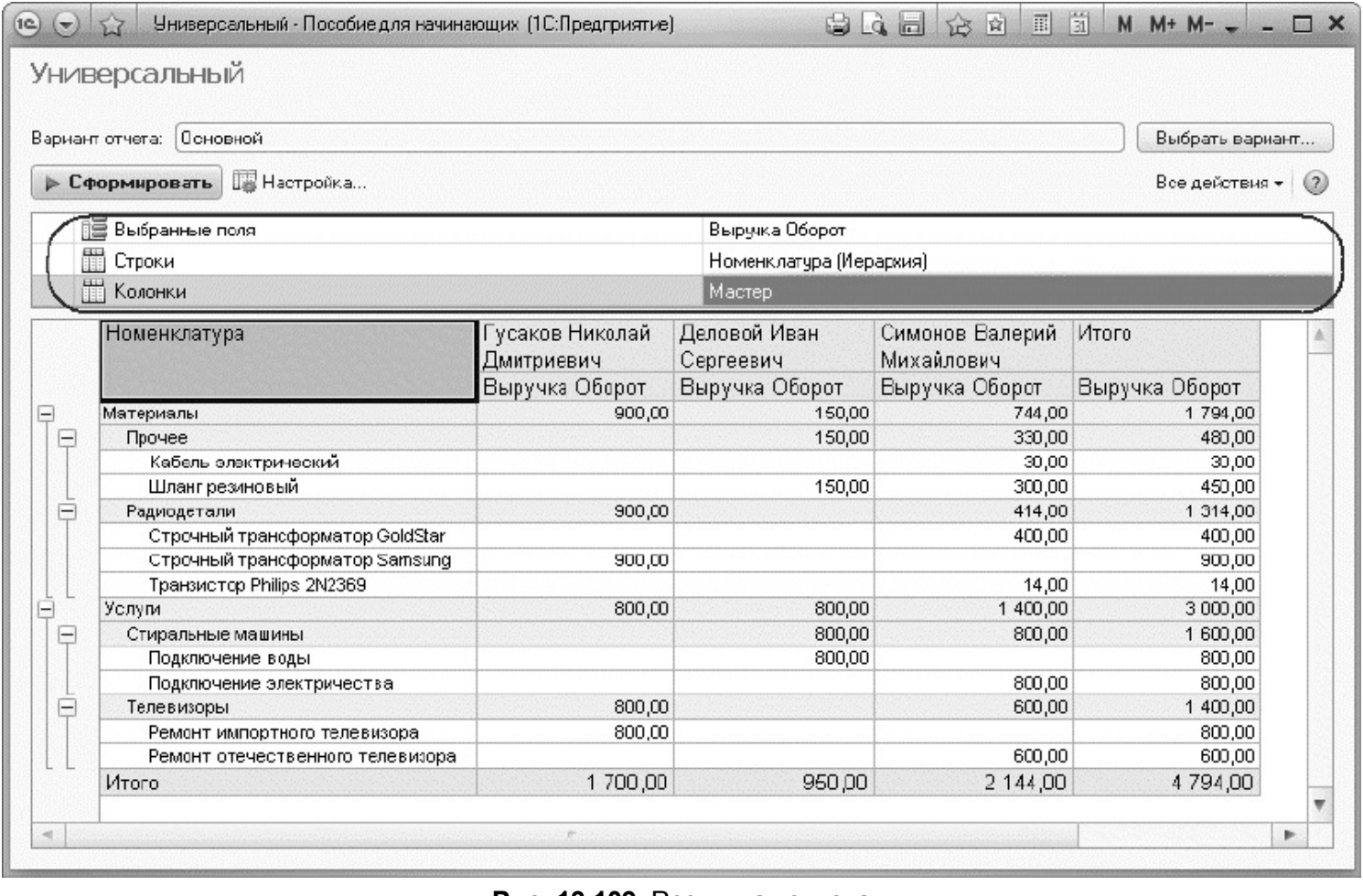
В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и выполним команду *Универсальный* в панели действий раздела *ОказаниеУслуг*.

Если мы сейчас нажмем *Сформировать*, то ничего не увидим в результате, так список выбранных полей, группировок строк и колонок таблицы пуст. Заполним эти быстрые пользовательские настройки.

Нажмем кнопку выбора в строке *Выбранные поля* и выберем из доступных полей поле *ВыручкаОборот*. Нажмем кнопку выбора в строке *Строки* и добавим в строки таблицы группировку по полю *Номенклатура* с типом *Иерархия*. Нажмем кнопку выбора в строке *Колонки* и добавим в колонки таблицы группировку по полю *Мастер*. Нажмем *Сформировать*.

Отчет примет следующий вид (рис. 13.109).



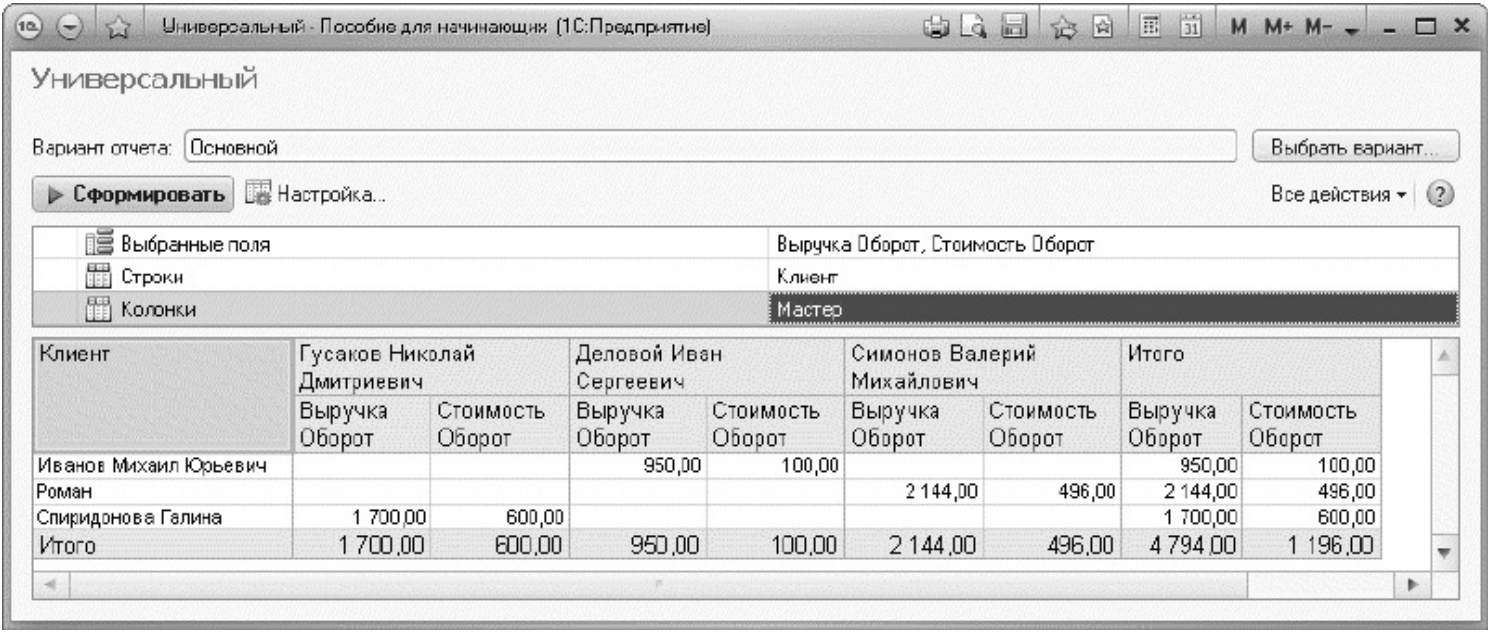
Номенклатура	Гусаков Николай Дмитриевич	Деловой Иван Сергеевич	Симонов Валерий Михайлович	Итого
	Выручка Оборот	Выручка Оборот	Выручка Оборот	Выручка Оборот
Материалы	900,00	150,00	744,00	1 794,00
Прочее		150,00	330,00	480,00
Кабель электрический			30,00	30,00
Шланг резиновый		150,00	300,00	450,00
Радиодетали	900,00		414,00	1 314,00
Строчный трансформатор GoldStar			400,00	400,00
Строчный трансформатор Samsung	900,00			900,00
Транзистор Philips 2N2369			14,00	14,00
Услуги	800,00	800,00	1 400,00	3 000,00
Стиральные машины		800,00	800,00	1 600,00
Подключение воды		800,00		800,00
Подключение электричества			800,00	800,00
Телевизоры	800,00		600,00	1 400,00
Ремонт импортного телевизора	800,00			800,00
Ремонт отечественного телевизора			600,00	600,00
Итого	1 700,00	950,00	2 144,00	4 794,00

Рис. 13.109. Результат отчета

Теперь добавим в список выбранных полей поле *СтоимостьОборот*. В строки

таблицы вместо группировки по полю *Номенклатура* поместим группировку по полю *Клиент*.

В результате отчет примет следующий вид (рис. 13.110).



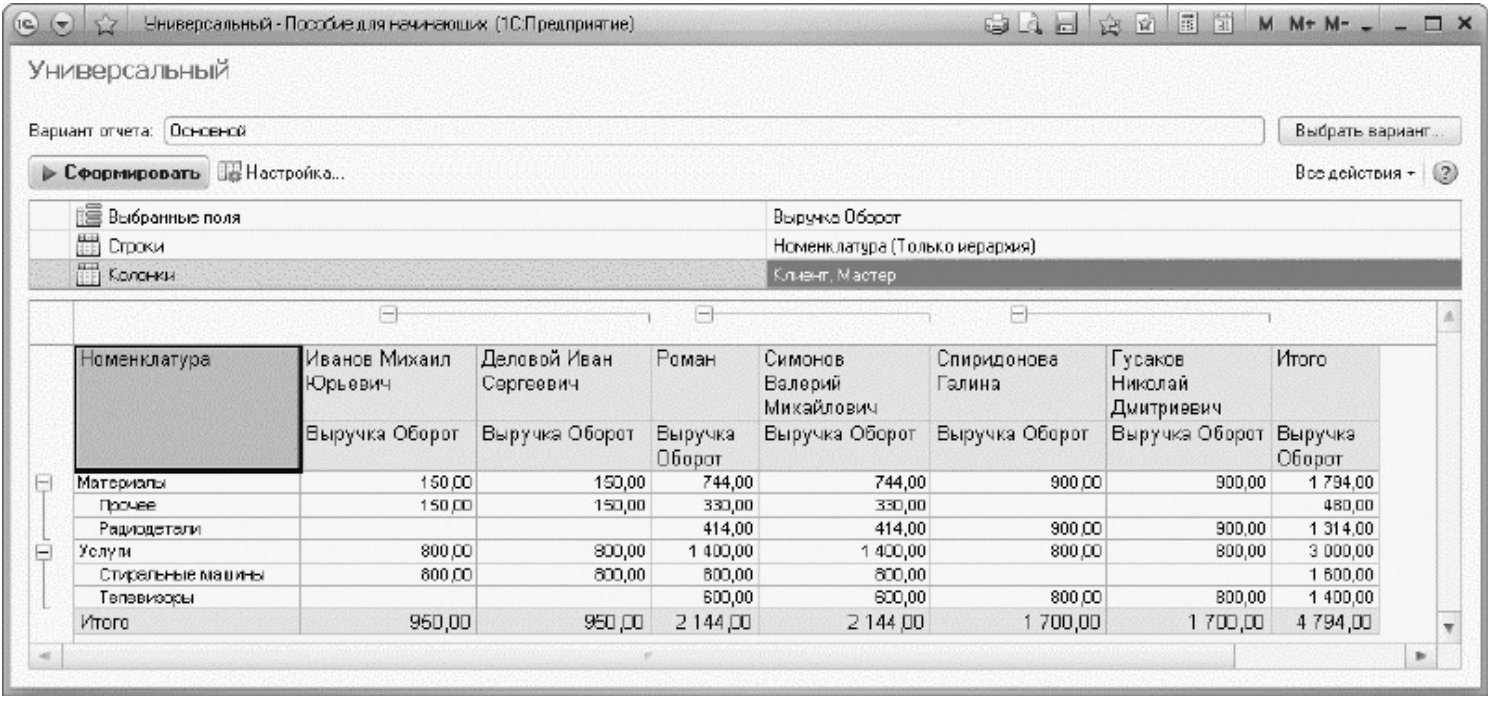
Клиент	Гусаков Николай Дмитриевич		Деловой Иван Сергеевич		Симонов Валерий Михайлович		Итого	
	Выручка Оборот	Стоимость Оборот	Выручка Оборот	Стоимость Оборот	Выручка Оборот	Стоимость Оборот	Выручка Оборот	Стоимость Оборот
Иванов Михаил Юрьевич			950,00	100,00			950,00	100,00
Роман					2 144,00	496,00	2 144,00	496,00
Спиридонова Галина	1 700,00	600,00					1 700,00	600,00
Итого	1 700,00	600,00	950,00	100,00	2 144,00	496,00	4 794,00	1 196,00

Рис. 13.110. Результат отчета

Теперь исключим из списка выбранных полей поле *СтоимостьОборот*. В строках таблицы заменим прежнюю группировку на группировку по полю *Номенклатура* с типом *ТолькоИерархия*. В колонки таблицы добавим

группировку по полю *Клиент* и поместим ее первой в списке группировок.

В результате отчет примет следующий вид (рис. 13.111).



Номенклатура	Иванов Михаил Юрьевич		Деловой Иван Сергеевич		Роман		Симонов Валерий Михайлович		Спиридонова Галина		Гусаков Николай Дмитриевич		Итого	
	Выручка	Оборот	Выручка	Оборот	Выручка	Оборот	Выручка	Оборот	Выручка	Оборот	Выручка	Оборот	Выручка	Оборот
Материалы	150,00		150,00		744,00		744,00		900,00		900,00		1 794,00	
Прочее	150,00		150,00		330,00		330,00						480,00	
Радиодетали					414,00		414,00		900,00		900,00		1 314,00	
Услуги	800,00		800,00		1 400,00		1 400,00		800,00		800,00		3 000,00	
Стиральные машины	800,00		800,00		800,00		800,00						1 600,00	
Телевизоры					600,00		600,00		800,00		800,00		1 400,00	
Итого	950,00		950,00		2 144,00		2 144,00		1 700,00		1 700,00		4 794,00	

Рис. 13.111. Результат отчета

Таким образом, используя этот отчет, мы предоставили пользователю альтернативную возможность самостоятельно формировать отчет по регистру *Продажи*.

«Теория». Виртуальные таблицы запросов

Как вы теперь знаете, при создании запроса платформа предоставляет нам в качестве источников данных некоторое количество виртуальных таблиц. Название «виртуальные» полностью соответствует их сути, поскольку эти таблицы, в свою очередь, также являются результатом запроса, который система формирует в момент выполнения соответствующего участка кода.

По большому счету разработчик может самостоятельно получить те же самые данные, которые система предоставляет ему в качестве виртуальных таблиц, однако алгоритм получения этих данных не будет оптимизирован в силу следующих двух причин.

Во-первых, все виртуальные таблицы параметризованы, то есть разработчику предоставляется возможность задать некоторые параметры, которые система будет использовать при формировании запроса создания виртуальной таблицы.

Примечательным здесь является то, что задание параметров виртуальной таблицы далеко не всегда приводит к простой подстановке указанных разработчиком значений в текст запроса. В зависимости от того, какие параметры виртуальной таблицы указаны разработчиком, система может формировать РАЗЛИЧНЫЕ запросы для получения одной и той же виртуальной таблицы, причем они будут оптимизированы с точки зрения переданных

параметров.

Во-вторых, не всегда разработчик имеет возможность получить доступ к тем данным, к которым имеет доступ система. Например, при использовании виртуальных таблиц регистров сведений разработчику доступна по большому счету вся та же информация о данных регистров, которую использует система при формировании запроса виртуальной таблицы.

Совсем иная картина с виртуальными таблицами регистров накопления. Здесь система динамически формирует запрос в зависимости не только от переданных параметров, но и от периода рассчитанных итогов регистра, причем в запросе она использует данные рассчитанных итогов, которые просто недоступны для разработчика при создании запроса.

Конечно, разработчик может самостоятельно перебрать все записи регистра накопления и в итоге получить те же самые данные, которые система предоставляет в виде виртуальной таблицы. Однако очевидно, что такой запрос будет менее эффективным и потребует от разработчика гораздо больше трудозатрат.

Контрольные вопросы

- *Для чего предназначен объект встроенного языка «Запрос».*
- *Для чего предназначена система компоновки данных.*
- *Для чего предназначена схема компоновки данных.*
- *Для чего предназначены настройки компоновки данных.*
- *В чем отличие между реальными и виртуальными таблицами.*
- *Из каких частей состоит текст запроса, какие из них являются обязательными.*
- *Каковы основные синтаксические конструкции языка запросов.*
- *Что является источником данных запроса.*
- *Что такое псевдонимы в языке запросов.*
- *Что такое параметры запроса.*
- *Что такое параметры виртуальной таблицы.*
- *Что такое левое соединение.*
- *Как использовать конструктор запроса.*
- *Как выбрать данные в некотором периоде для отчета.*
- *Как упорядочить данные в отчете.*
- *Как использовать в отчете данные нескольких таблиц.*
- *Как использовать группировки в структуре отчета.*
- *Как получить последние значения регистра сведений.*
- *Как вывести в отчет иерархические данные.*

- *Как управлять выводом итогов по группировкам и общих итогов.*
- *Как создать отчет, содержащий диаграмму.*
- *Как использовать параметры в системе компоновки данных.*
- *Что такое ресурсы в системе компоновки данных.*
- *Что такое вычисляемые поля в системе компоновки данных.*
- *Как дополнить данные отчета всеми датами в группировке по периоду.*
- *Как создать пользовательские настройки отчета.*
- *В чем отличие «быстрых» настроек от остальных пользовательских настроек.*
- *Как определить состав пользовательских настроек отчета.*
- *Как вывести данные в виде таблицы.*
- *Как сделать отчет универсальным.*

Занятие 14 (3:20). Оптимизация проведения документа «Оказание услуги»

Продолжительность

Ориентировочная продолжительность занятия – 3 часа 20 минут.

После изучения предыдущего занятия вы уже достаточно хорошо знакомы с языком запросов, и мы наконец-то можем приступить к одному из самых важных занятий нашей книги – к оптимизации документа *ОказаниеУслуги* и, в частности, к полному изменению его обработчика события *ОбработкаПроведения*.

«Зачем это нужно?» – можете спросить вы. Тому есть три причины.

Во-первых, в обработчике события *ОбработкаПроведения* мы используем обращение к реквизиту *ВидНоменклатуры* справочника *Номенклатура* «через точку». Такое обращение может сильно замедлить скорость выполнения процедуры при больших объемах табличной части документа.

Во-вторых, руководство ООО «На все руки мастер» решило наконец-то завершить «эксперименты» по ручному вводу стоимости расходуемых материалов и перейти на автоматический расчет стоимости расходуемых материалов «по среднему».

В-третьих, при проведении документа *ОказаниеУслуги* необходимо контролировать остатки расходуемых товаров на складе. Если товаров не хватает, выдавать предупреждение и не проводить документ.

Поэтому изменения, вносимые нами в документ *ОказаниеУслуги*, будут преследовать три цели:

- повышение скорости выполнения процедуры;
- автоматическое определение стоимости расходуемых материалов при проведении документа;
- разделение алгоритма проведения документа на оперативный и неоперативный режимы и контроль остатков в случае оперативного проведения документа.

Прежде чем мы приступим непосредственно к каким-либо действиям, следует сказать несколько слов об особенностях хранения и использования ссылочных данных в системе «1С:Предприятие 8».

«Теория». Особенности использования ссылочных данных

В этом разделе мы поговорим об особенностях использования ссылочных

данных, так как, используя доступ к этим данным с помощью запросов, мы можем значительно повысить скорость проведения документа и оптимизировать этот процесс.

Термином ссылочные данные мы будем обозначать данные, хранящиеся в базе данных, доступ к которым возможен при помощи объектов встроенного языка вида *Ссылка*: *СправочникСсылка.<имя>, ДокументСсылка.<имя>* и т. д. Для того чтобы дальнейшее изложение было понятнее, мы построим объяснение на примере получения ссылки на вид номенклатуры при проведении документа *ОказаниеУслуги*.

Не все данные, хранящиеся в базе данных, являются ссылочными. Это связано с тем, что в модели данных «1С:Предприятия 8» существует деление на данные, представляющие объектные сущности (справочники, планы счетов, документы и т. д.), и данные, представляющие необъектные сущности (регистры сведений, регистры накопления и т. д.).

С точки зрения платформы некоторая совокупность объектных данных определяется не только значениями своих полей, но и самим фактом своего существования. Другими словами, удалив из базы некоторую совокупность объектных данных, мы не сможем вернуть систему в то же состояние, которое было до удаления. Даже если мы заново создадим ту же самую совокупность объектных данных с теми же самыми значениями полей, с точки зрения системы

это будет ДРУГАЯ совокупность объектных данных.

Каждую такую совокупность объектных данных, уникальную с точки зрения системы, называют объектом базы данных.

Для того чтобы система могла отличить один объект базы данных от другого, каждый объект базы данных (совокупность объектных данных) имеет внутренний идентификатор. Различные объекты базы данных всегда будут иметь разные внутренние идентификаторы. Этот идентификатор хранится вместе с остальными данными объекта в специальном поле *Ссылка*.

Необъектные данные хранятся в виде записей и с точки зрения системы определяются исключительно значениями своих полей. Таким образом, удалив некоторую запись и записав после этого новую, с точно такими же значениями всех полей, мы получим то же самое состояние базы данных, которое было до удаления.

Таким образом, поскольку мы можем однозначно указать на каждый объект базы данных, у нас появляется возможность хранить такой указатель в полях других таблиц базы данных, выбирать его в поле ввода, указывать в параметрах запроса при поиске по ссылке и т. д.

Во всех этих случаях как раз и будет использоваться объект встроенного языка

вида *Ссылка*. Фактически этот объект хранит только внутренний идентификатор, находящийся в поле *Ссылка*.

Например, если взять наш документ *ОказаниеУслуги*, то в поле, хранящем реквизит табличной части *Номенклатура,* на самом деле находится внутренний идентификатор, указывающий на элемент справочника *Номенклатура* (рис. 14.1).

Рис. 14.1. Ссылка на элемент справочника «Номенклатура»

Когда в обработчике события *ОбработкаПроведения* документа *ОказаниеУслуги* мы присваиваем значение реквизита табличной части *Номенклатура* какой-либо переменной, мы имеем дело с объектом встроенного языка *ДокументОбъект.ОказаниеУслуги*.

Этот объект содержит в себе значения всех реквизитов документа и реквизитов его табличных частей.

Поэтому обращение (листинг 14.1) приводит к тому, что мы просто читаем данные, хранящиеся в оперативной памяти, в этом самом объекте встроенного языка (рис. 14.2).

Листинг 14.1. Обращение к реквизиту объекта

```
Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
```


Ссылка на Номенклатуру?
Ссылка
Оперативная память
ДокументОбъект. Оказание Услуги
База данных
Документ ОказаниеУслуги
Ссылка
Номер
Ссылка
Номенклатура
1

Рис. 14.2. Чтение данных из оперативной памяти

Однако когда мы обращаемся к виду номенклатуры как к реквизиту того элемента справочника, ссылка на который указана в табличной части документа (листинг 14.2), происходит буквально следующее (рис. 14.3).

Листинг 14.2. Обращение к реквизиту ссылки

```
Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
Перечисления.ВидыНоменклатуры.Материал Тогда
```


Ссылка на ВидНоменклатуры?
Ссылка
Оперативная память
ДокументОбъект. ОказаниеУслуги
Ссылка на элемент справочника Номенклатура
Кэш объектов
База данных
Справочник Номенклатура
Ссылка
Код
Наименование
11
Подключение воды

Рис. 14.3. Использование кеша объектов

Поскольку в объекте *ДокументОбъект.ОказаниеУслуги* есть только ссылка на элемент справочника *Номенклатура* и больше никаких данных об этом элементе нет, платформа возьмет эту ссылку и обратится по ней в кеш объектов в надежде найти там данные того объекта, ссылка на который у нее есть.

Если кеш объектов не будет иметь нужных данных, он обратится к базе данных с тем, чтобы прочитать все данные объекта, ссылкой на который он обладает.

После того как все данные, хранящиеся в реквизитах нужного элемента справочника и в реквизитах его табличных частей, будут считаны в кеш объектов, кеш объектов вернет запрашиваемую ссылку, хранящуюся в реквизите *ВидНоменклатуры* справочника *Номенклатура*.

Как несложно догадаться, подобное обращение к базе данных требует большего количества времени, нежели просто чтение из оперативной памяти. При интерактивном заполнении документа подобные задержки ничтожно малы, по сравнению со скоростью работы пользователя. Однако при выполнении большого количества расчетов (например, при проведении больших документов, содержащих несколько тысяч строк) разница во времени может быть довольно заметной.

Из всего вышесказанного можно сделать следующий вывод: если алгоритм проведения документа использует только те данные, которые присутствуют в реквизитах документа (и его табличных частей), вполне достаточно использовать конструктор движений документа (как это было у нас в случае с документом *Приходная Накладная*).

Если же в алгоритме проведения требуется анализировать дополнительные реквизиты объектов, ссылки на которые содержатся в документе, а также использовать результаты расчета итогов регистров, следует использовать запросы для более быстрой выборки данных из базы данных.

То же самое справедливо в отношении выполнения любых участков программы, критичных по производительности. Механизм запросов лучше «читает» информационную базу и может за один раз выбрать только те данные, которые необходимы. Поэтому, например, в типовых решениях вы практически не увидите использования объекта встроенного языка *СправочникВыборка.<имя>*. Вместо этого повсеместно используются запросы к базе данных.

Повышение скорости проведения

Первое, чем мы займемся на этом занятии, – избавимся от «вредной» конструкции *ТекСтрокаПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры*.

В режиме «Конфигуратор»

Откроем модуль документа *ОказаниеУслуги*.

Напомним, как выглядит сейчас процедура проведения этого документа (листинг 14.3).

Листинг 14.3. Процедура «ОбработкаПроведения»

```
Движения.ОстаткиМатериалов.Записывать = Истина;
Движения.СтоимостьМатериалов.Записывать = Истина;
Движения.Продажи.Записывать = Истина;

Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

  Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаПереченьНоменклатуры.Количество;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
```

```
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаПеречеьНоменклатуры.Номенклатура;
    Движение.Стоимость = ТекСтрокаПеречеьНоменклатуры.Количество *
    ТекСтрокаПеречеьНоменклатуры.Стоимость;

  КонецЕсли;

  // Регистр Продажи
  Движение = Движения.Продажи.Добавить();
  Движение.Период = Дата;
  Движение.Номенклатура = ТекСтрокаПеречеьНоменклатуры.Номенклатура;
  Движение.Клиент = Клиент;
  Движение.Мастер = Мастер;
  Движение.Количество = ТекСтрокаПеречеьНоменклатуры.Количество;
  Движение.Выручка = ТекСтрокаПеречеьНоменклатуры.Сумма;
  Движение.Стоимость = ТекСтрокаПеречеьНоменклатуры.Стоимость *
  ТекСтрокаПеречеьНоменклатуры.Количество;

КонецЦикла;
```

Другими словами, все данные, необходимые для проведения документа, мы получаем из самого документа, и только для определения того, чем является номенклатура (товаром или услугой), мы обращаемся к базе данных, читая данные всего объекта *Номенклатура* (листинг 14.4).

Листинг 14.4. Обращение к объекту «Номенклатура»

```
Если ТекСтрокаПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры
```

Забегая вперед, скажем, что это не единственные данные, которые не содержатся в самом документе и которые в то же время будут нужны нам для правильного его проведения.

Поэтому поступим следующим образом: все данные, связанные с номенклатурой, которая содержится в табличной части документа, мы будем получать с помощью запроса к базе данных. А данные, связанные с самим документом (например, дата документа, склад), мы по-прежнему будем получать из документа. Такой подход позволит нам читать только нужные данные и за счет этого максимально ускорить проведение документа.

Итак, запросом мы будем получать:

- номенклатуру,
- количество,
- сумму,
- стоимость.

Из документа мы возьмем следующие данные:

- дата,

- клиент,
- мастер,
- склад.

Приступим к созданию запроса. Установим курсор перед циклом обхода табличной части документа и из контекстного меню выберем пункт *Конструктор запроса с обработкой результата* (рис. 14.4).

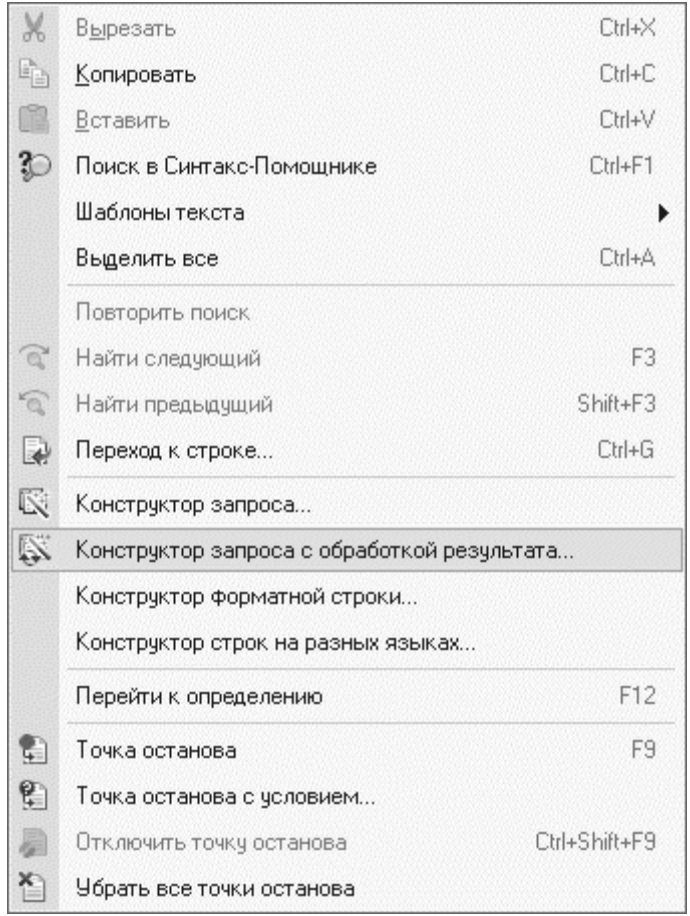


Рис. 14.4. Вызов конструктора запроса

Подтвердим, что мы хотим создать новый запрос.

В окне конструктора запросов перейдем на закладку *Таблицы и поля* и выберем таблицу *ОказаниеУслугиПереченьНоменклатуры* – это табличная часть документа *ОказаниеУслуги*.

Из этой таблицы нам нужны поля – *Номенклатура*, *Номенклатура.ВидНоменклатуры, Количество, Сумма* и *Стоимость* (рис. 14.5).

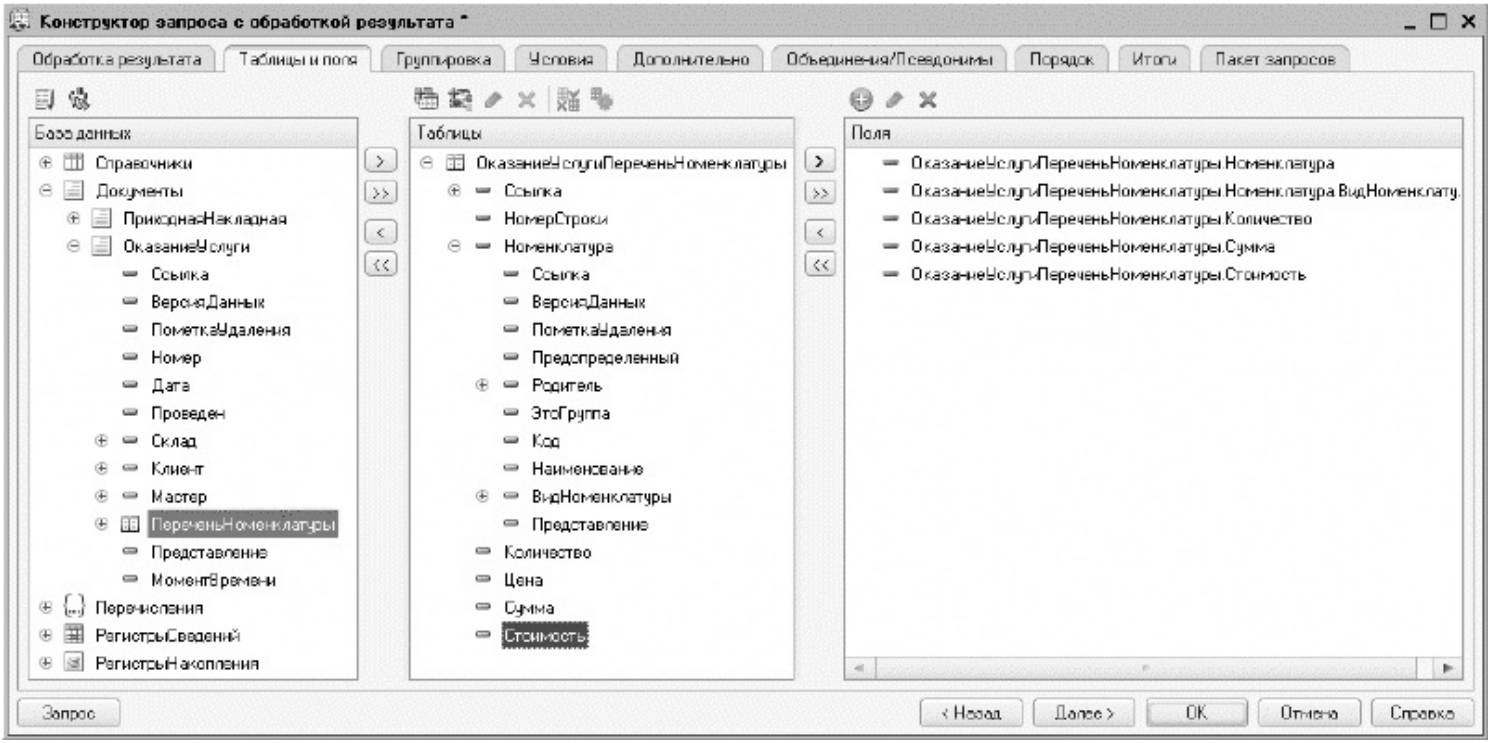


Рис. 14.5. Выбранные поля

Но нам нужны не все записи этой таблицы, а только те, которые относятся к нашему документу.

Поэтому перейдем на закладку *Условия* и зададим условие отбора из таблицы документа только строк проводимого документа.

Для этого перетащим поле *Ссылка* в список условий запроса (листинг 14.5).

Листинг 14.5. Условие отбора из таблицы документа

```
ОказаниеУслугиПереченьНоменклатуры.Ссылка = &Ссылка
```

Ссылка на этот документ будет передана в параметр запроса *Ссылка* (рис. 14.6).

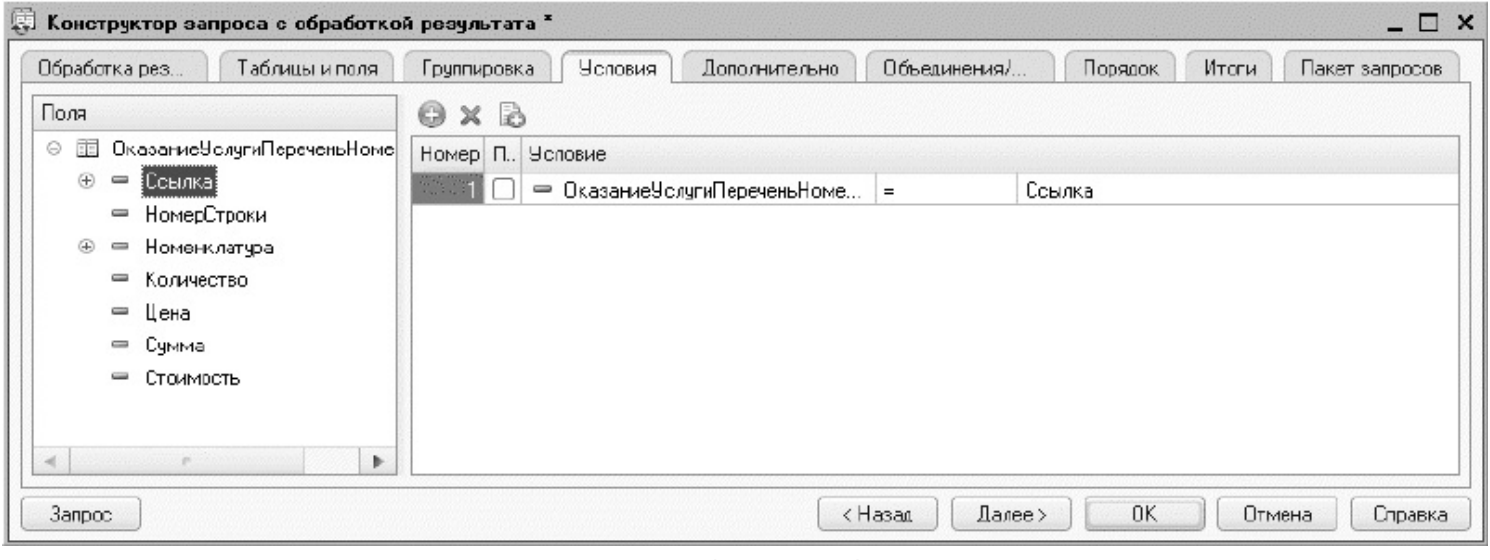


Рис. 14.6. Условие отбора из таблицы документа

Также следует учесть, что в табличной части документа одна и та же

номенклатура может встречаться несколько раз.

Поэтому на закладке *Группировка* сгруппируем наши записи по полю *Номенклатура* и *НоменклатураВидНоменклатуры*, а рассчитывать будем сумму значений для полей *Количество* и *Сумма*.

Благодаря этому в результате значения номенклатуры повторяться не будут, и для каждого из них будут посчитаны суммарные значения по полям *Количество* и *Сумма*, если в табличной части документа содержится несколько строк с одинаковой номенклатурой.

Также в состав суммируемых полей включим и поле *Стоимость*. По нему будем рассчитывать, например, функцию *Максимум*.

Мы подразумеваем, что для разных строк одной и той же номенклатуры стоимость будет одинаковой, поэтому функция *Максимум* нужна нам лишь для того, чтобы получить одно из имеющихся значений стоимости (рис. 14.7).

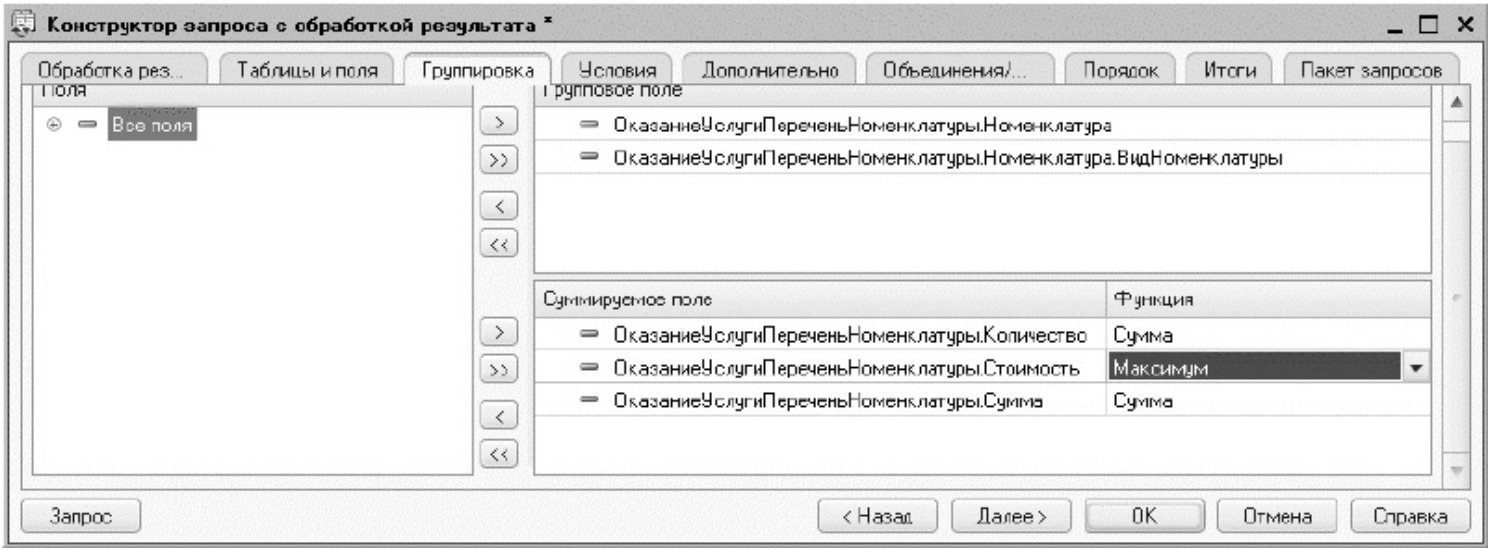


Рис. 14.7. Группировка строк таблицы документа

На закладке *Объединения/Псевдонимы* зададим псевдонимы для полей *Количество* и *Сумма – КоличествоВДокументе* и *СуммаВДокументе*, а для поля *НоменклатураВидНоменклатуры* зададим псевдоним *ВидНоменклатуры* просто для облегчения чтения запроса (рис. 14.8).

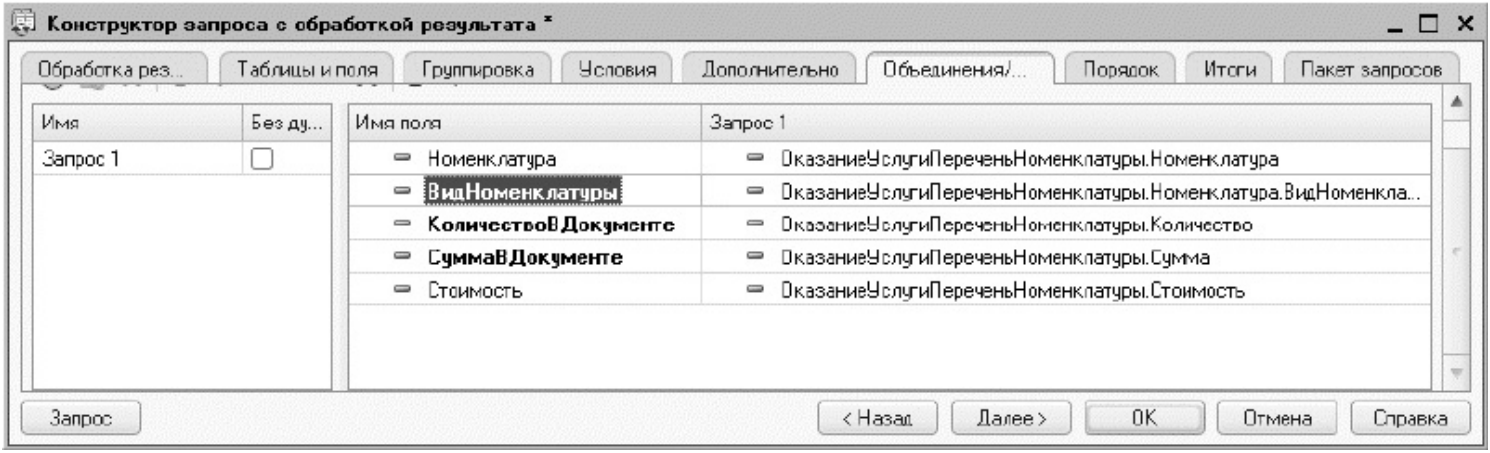


Рис. 14.8. Псевдонимы полей

Нажмем *ОК* и посмотрим, какой текст запроса сформировал конструктор (листинг 14.6).

Листинг 14.6. Текст запроса

```
//{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
// Данный фрагмент построен конструктором.
// При повторном использовании конструктора внесенные вручную изменения будут
утеряны!!!

Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
  |        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры КАК
```

```
  ВидНоменклатуры,
  |        СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Количество) КАК
  КоличествоВДокументе,
  |        СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Сумма) КАК СуммаВДокументе,
  |        МАКСИМУМ(ОказаниеУслугиПеречеНьНоменклатуры.Стоимость) КАК Стоимость
  |ИЗ
  |        Документ.ОказаниеУслуги.ПеречеНьНоменклатуры КАК
  ОказаниеУслугиПеречеНьНоменклатуры
  |ГДЕ
  |        ОказаниеУслугиПеречеНьНоменклатуры.Ссылка = &Ссылка
  |
  |СГРУППИРОВАТЬ ПО
  |        ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
  |        ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры";

Запрос.УстановитьПараметр("Ссылка", Ссылка);

Результат = Запрос.Выполнить();

ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
  // Вставить обработку выборки ВыборкаДетальныеЗаписи
КонецЦикла;

//}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
```

Комментарии конструктора запроса в начале и конце фрагмента можно удалить.

Поскольку для построения запроса мы использовали *Конструктор запроса с обработкой результата*, конструктор написал за нас код для выполнения и обхода записей запроса. Прокомментируем этот код.

Как вы уже знаете, для работы с запросами используется объект встроенного языка *Запрос*. Вначале создается новый объект *Запрос* и помещается в переменную *Запрос*. Затем в свойство *Текст* объекта *Запрос* помещается сам текст запроса (*Запрос.Текст = …*).

После этого устанавливается значение параметра запроса *&Ссылка* как ссылка на тот документ, в модуле которого мы сейчас находимся (листинг 14.7).

Листинг 14.7. Установка параметра запроса

```
Запрос.УстановитьПараметр("Ссылка", Ссылка);
```

Затем запрос выполняется (*Запрос.Выполнить()*), получается объект *РезультатЗапроса,* и выполняется его метод *Выбрать()*, который формирует выборку записей из результата запроса.

Таким образом, получается объект *ВыборкаИзРезультатаЗапроса*, который помещается в переменную *ВыборкаДетальныеЗаписи*.

Далее, используя метод этого объекта *Следующий*() (*ВыборкаДетальныеЗаписи.Следующий()*), мы будем в цикле обходить выборку записей запроса.

Выполняя метод выборки запроса *ВыборкаДетальныеЗаписи.Следующий()*, мы на каждом шаге цикла позиционируем указатель на следующую запись выборки, пока не будет достигнут конец выборки.

Чтобы в цикле получить значение какого-либо поля выборки из результата запроса, мы будем обращаться к полям запроса через точку от переменной *ВыборкаДетальныеЗаписи*, которая содержит текущую строку выборки запроса. Например, так: *ВыборкаДетальныеЗаписи.Номенклатура*.

Теперь нам осталось перенести существовавшие ранее в этом модуле строки, описывающие движения регистров, внутрь цикла обхода результата запроса (листинг 14.8).

Листинг 14.8. Цикл обхода записей запроса

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
  // Вставить обработку выборки ВыборкаДетальныеЗаписи
КонецЦикла;
```

Сначала вместо комментария «// Вставить обработку выборки

ВыборкаДетальныеЗаписи» перенесем условие проверки и весь код, формирующий движения по регистрам *ОстаткиМатериалов* и *СтоимостьМатериалов* (листинг 14.9).

Листинг 14.9. Формирование движений регистров

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ТекСтрокаПеречеНоменклатуры.Номенклатура.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаПеречеНоменклатуры.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаПеречеНоменклатуры.Количество;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаПеречеНоменклатуры.Номенклатура;
    Движение.Стоимость = ТекСтрокаПеречеНоменклатуры.Количество *
    ТекСтрокаПеречеНоменклатуры.Стоимость;

  КонецЕсли;
```

```
КонецЦикла;
```

В условии заменим *ТекСтрокаПеречеНоменклатуры.Номенклатура* на *ВыборкаДетальныеЗаписи*, так как вид номенклатуры мы теперь получаем из запроса.

В движениях также заменим *ТекСтрокаПеречеНоменклатуры* на *ВыборкаДетальныеЗаписи* (листинг 14.10).

Листинг 14.10. Формирование движений регистров

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ВыборкаДетальныеЗаписи.Количество;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
```

```
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.Количество *
    ВыборкаДетальныеЗаписи.Стоимость;

  КонецЕсли;

КонецЦикла;
```

ПРИМЕЧАНИЕ

Для упрощения восприятия новый текст в листингах выделен жирным шрифтом.

Не забудем, что для поля *Количество* мы задали псевдоним в запросе, поэтому заменим его на *КоличествоВДокументе* (листинг 14.11).

Листинг 14.11. Формирование движений регистров

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
```

```
    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе *
    ВыборкаДетальныеЗаписи.Стоимость;

  КонецЕсли;

КонецЦикла;
```

Теперь перенесем формирование движений по регистру *Продажи* (листинг 14.12).

Листинг 14.12. Формирование движений регистров

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
```

```
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе *
    ВыборкаДетальныеЗаписи.Стоимость;

  КонецЕсли;

  // Регистр Продажи
  Движение = Движения.Продажи.Добавить();
  Движение.Период = Дата;
  Движение.Номенклатура = ТекСтрокаПереченьНоменклатуры.Номенклатура;
  Движение.Клиент = Клиент;
  Движение.Мастер = Мастер;
  Движение.Количество = ТекСтрокаПереченьНоменклатуры.Количество;
  Движение.Выручка = ТекСтрокаПереченьНоменклатуры.Сумма;
  Движение.Стоимость = ТекСтрокаПереченьНоменклатуры.Стоимость *
  ТекСтрокаПереченьНоменклатуры.Количество;

КонецЦикла;
```

Здесь произведем аналогичные замены. *ТекСтрокаПереченьНоменклатуры* заменим на *ВыборкаДетальныеЗаписи* (листинг 14.13).

Листинг 14.13. Формирование движений регистров

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе *
    ВыборкаДетальныеЗаписи.Стоимость;

  КонецЕсли;

  // Регистр Продажи
  Движение = Движения.Продажи.Добавить();
  Движение.Период = Дата;
  Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
  Движение.Клиент = Клиент;
  Движение.Мастер = Мастер;
```

```
  Движение.Количество = ВыборкаДетальныеЗаписи.Количество;
  Движение.Выручка = ВыборкаДетальныеЗаписи.Сумма;
  Движение.Стоимость = ВыборкаДетальныеЗаписи.Стоимость *
  ВыборкаДетальныеЗаписи.Количество;

КонецЦикла;
```

Поля запроса *Сумма* и *Количество* заменим на *СуммаВДокументе* и *КоличествоВДокументе* (листинг 14.14).

Листинг 14.14. Формирование движений регистров

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
```

```
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе *
    ВыборкаДетальныеЗаписи.Стоимость;

  КонецЕсли;

  // Регистр Продажи
  Движение = Движения.Продажи.Добавить();
  Движение.Период = Дата;
  Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
  Движение.Клиент = Клиент;
  Движение.Мастер = Мастер;
  Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;
  Движение.Выручка = ВыборкаДетальныеЗаписи.СуммаВДокументе;
  Движение.Стоимость = ВыборкаДетальныеЗаписи.Стоимость *
  ВыборкаДетальныеЗаписи.КоличествоВДокументе;

КонецЦикла;
```

Оставшийся цикл обхода табличной части можно удалить (листинг 14.15).

Листинг 14.15. Ненужные строки

```
Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

КонецЦикла;
```

В результате процедура проведения примет следующий вид (листинг 14.16).

Листинг 14.16. Процедура «ОбработкаПроведения»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  Движения.ОстаткиМатериалов.Записывать = Истина;
  Движения.СтоимостьМатериалов.Записывать = Истина;
  Движения.Продажи.Записывать = Истина;

  Запрос = Новый Запрос;
  Запрос.Текст =
  "ВЫБРАТЬ
  |        ОказаниеУслугиПеречеНоменклатуры.Номенклатура,
  |        ОказаниеУслугиПеречеНоменклатуры.Номенклатура.ВидНоменклатуры КАК
  ВидНоменклатуры,
  |        СУММА(ОказаниеУслугиПеречеНоменклатуры.Количество) КАК
  КоличествоВДокументе,
  |        СУММА(ОказаниеУслугиПеречеНоменклатуры.Сумма) КАК СуммаВДокументе,
  |        МАКСИМУМ(ОказаниеУслугиПеречеНоменклатуры.Стоимость) КАК Стоимость
  |ИЗ
  |        Документ.ОказаниеУслуги.ПеречеНоменклатуры КАК
  ОказаниеУслугиПеречеНоменклатуры
  |ГДЕ
  |        ОказаниеУслугиПеречеНоменклатуры.Ссылка = &Ссылка
  |
  |СГРУППИРОВАТЬ ПО
  |        ОказаниеУслугиПеречеНоменклатуры.Номенклатура,
  |        ОказаниеУслугиПеречеНоменклатуры.Номенклатура.ВидНоменклатуры";
```

```
Запрос.УстановитьПараметр("Ссылка", Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе *
    ВыборкаДетальныеЗаписи.Стоимость;

  КонецЕсли;

  // Регистр Продажи
  Движение = Движения.Продажи.Добавить();
  Движение.Период = Дата;
  Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
```

```
    Движение.Клиент = Клиент;
    Движение.Мастер = Мастер;
    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;
    Движение.Выручка = ВыборкаДетальныеЗаписи.СуммаВДокументе;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.Стоимость *
    ВыборкаДетальныеЗаписи.КоличествоВДокументе;

  КонецЦикла;

КонецПроцедуры
```

В режиме «1С:Предприятие»

Теперь нужно запустить «1С:Предприятие» в режиме отладки, перепровести документы *Оказание услуги* и проверить, что ничего не изменилось.

Таким образом, мы выполнили первый пункт нашего «плана» – избавились в процедуре проведения от считывания всех данных объекта *Номенклатура* и тем самым оптимизировали выполнение процедуры проведения.

Автоматический расчет стоимости

Теперь приступим ко второму этапу нашего плана.

До сих пор стоимость расходуемых материалов мы вписывали в документ

Оказание услуги вручную, при его создании.

Теперь же будем определять стоимость номенклатуры по среднему: для каждой номенклатуры делить ее общую, суммарную стоимость на то количество этой номенклатуры, которое у нас имеется, и таким образом получать среднюю стоимость одной единицы этой номенклатуры.

Чтобы выполнить такой расчет, нам понадобятся дополнительные данные, которых у нас сейчас нет.

Для каждой номенклатуры из табличной части нам понадобятся:

- ее стоимость, хранящаяся в регистре *СтоимостьМатериалов;*
- общее ее количество на всех складах, хранящееся в регистре *ОстаткиМатериалов*.

Поэтому нам нужно будет доработать наш запрос таким образом, чтобы он получал из базы данных и эти данные тоже.

Таким образом, нам хотелось бы, чтобы запрос возвращал следующие поля для каждой номенклатуры, которая есть в документе (рис. 14.9).

Номен-клатура	Кол-во в документе	Сумма в документе	Вид номен-клатуры	Стои-мость	Кол-во на всех складах

Рис. 14.9. Описание полей запроса

Первые четыре поля мы можем получить (и уже получаем) из табличной части самого документа, а вот последние два нужно будет получить из других таблиц базы данных:

- стоимость – из регистра *СтоимостьМатериалов*;
- остатки на всех складах – из регистра *ОстаткиМатериалов* (рис. 14.10).

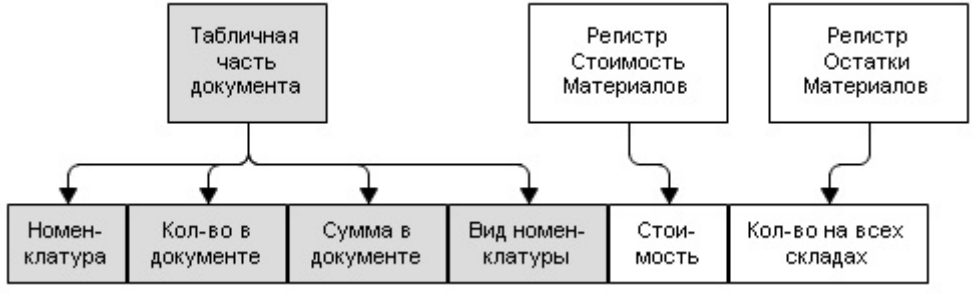


Рис. 14.10. Описание полей и таблиц запроса

Это значит, что наш запрос должен содержать два левых соединения таблицы документа с другими таблицами: одно – с таблицей *РегистрНакопления.СтоимостьМатериалов.Остатки*, другое – с таблицей *РегистрНакопления.ОстаткиМатериалов.Остатки*.

Казалось бы, все готово для того, чтобы составить запрос.

Но обратите внимание на важную деталь: в предложенной схеме виртуальные таблицы будут возвращать стоимость и остатки номенклатуры абсолютно для всей номенклатуры. А нас интересует только та номенклатура, которая указана в нашем документе.

На маленькой базе эта особенность может почти не проявляться – количество различных элементов номенклатуры в справочнике сравнимо с количеством различных элементов номенклатуры в документе.

Но представьте реальную базу. В справочнике *Номенклатура* – 15 000 наименований, например. А в документе – всего 5 наименований. Получится, что сначала виртуальная таблица остатков будет усиленно трудиться и рассчитывать нам стоимость (или остатки) по всем 15 000 наименованиям номенклатуры, а в результате, когда мы левым соединением станем соединять ее с таблицей документа, мы из этих 15 000 строк стоимости (или остатков) возьмем всего лишь 5 строк – для той номенклатуры, которая указана в документе. Остальные 14 995 строк будут просто отброшены – система напрасно их рассчитывала.

Естественно, такая расточительность непозволительна в реальных условиях, и такой запрос является очень неоптимальным. Поэтому во все виртуальные

таблицы, которые мы будем использовать, нужно добавить условие отбора только той номенклатуры, которая содержится в табличной части нашего документа. В этом случае стоимость и остатки будут рассчитаны не для всей номенклатуры вообще, а только для нужной нам номенклатуры.

В результате схема нашего запроса будет выглядеть следующим образом (рис. 14.11).

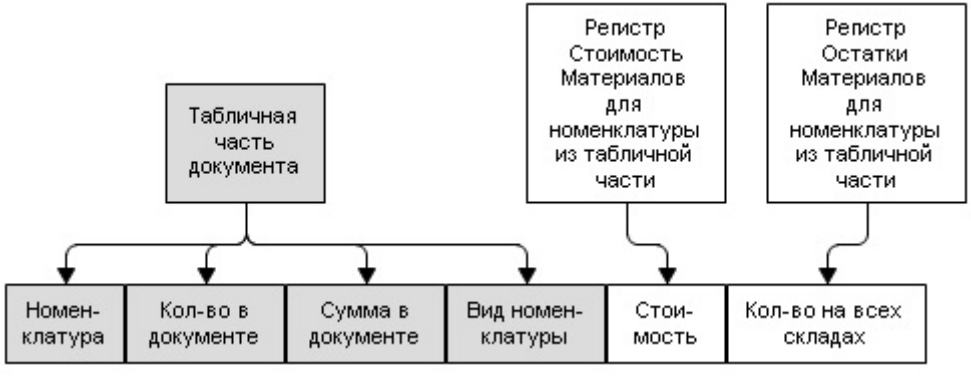


Рис. 14.11. Схема запроса

Обратите внимание, что в обеих виртуальных таблицах используется список номенклатуры из табличной части документа (по нему ограничиваются рассчитываемые данные).

Кроме этого, из таблиц документа тоже берутся не все данные, а только

данные, относящиеся к одному нашему конкретному документу. Чтобы не получать этот список три раза (для документа и в каждой виртуальной таблице заново), мы можем сформировать его заранее и затем уже использовать в нужных нам условиях запроса.

Выполнить эту задачу нам помогут временные таблицы.

Временные таблицы – это программные объекты, которые разработчик может создать и заполнить данными, а запросы могут использовать данные временных таблиц для своих нужд. Например, для наложения некоторого сложного условия, как в нашем случае.

Таким образом, схема нашего запроса приобретает следующий вид (рис. 14.12).

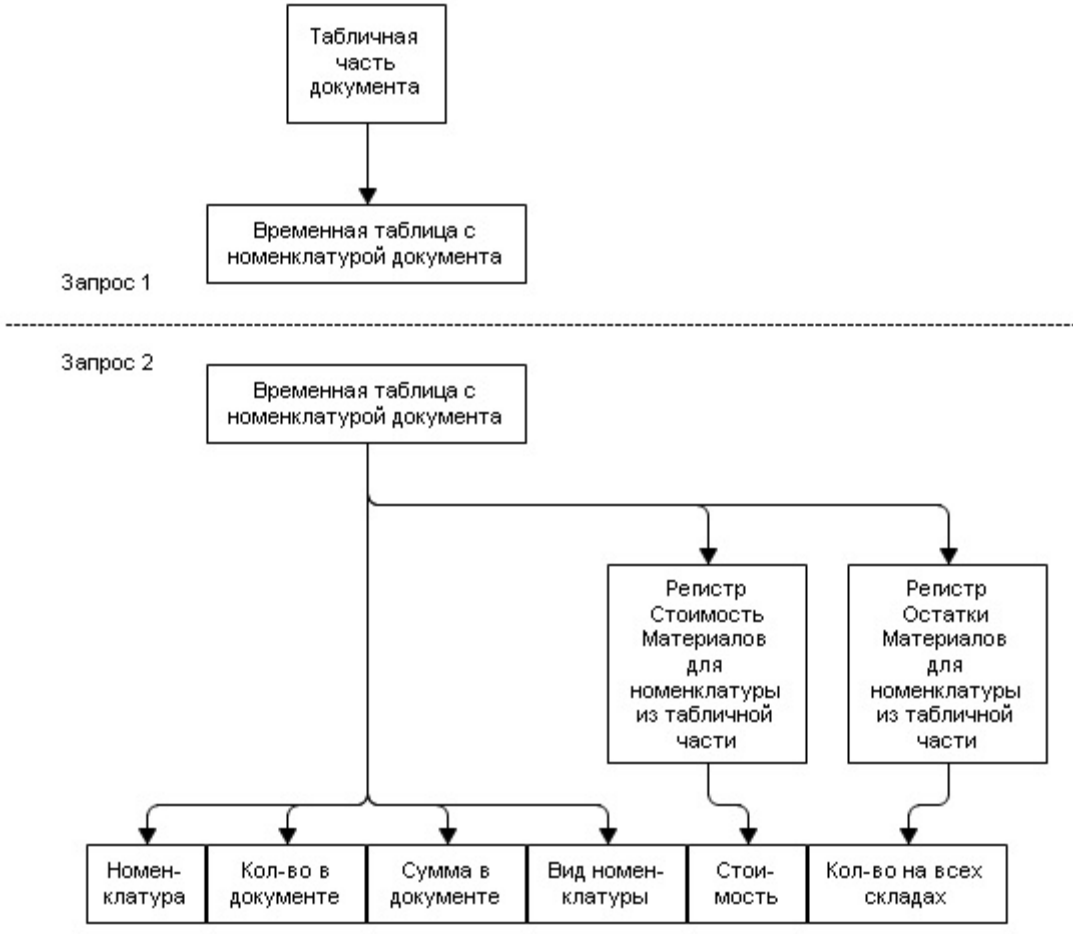


Рис. 14.12. Схема запроса

Итак, приступим.

В режиме «Конфигуратор»

Первое, что мы сделаем, – удалим реквизит табличной части *Стоимость* документа *ОказаниеУслуги*, который нам больше не понадобится.

Для этого откроем в конфигураторе окно редактирования объекта конфигурации Документ *ОказаниеУслуги*, перейдем на закладку *Данные*, раскроем список реквизитов табличной части документа, выделим реквизит *Стоимость* и нажмем кнопку *Удалить* в командной панели (рис. 14.13).

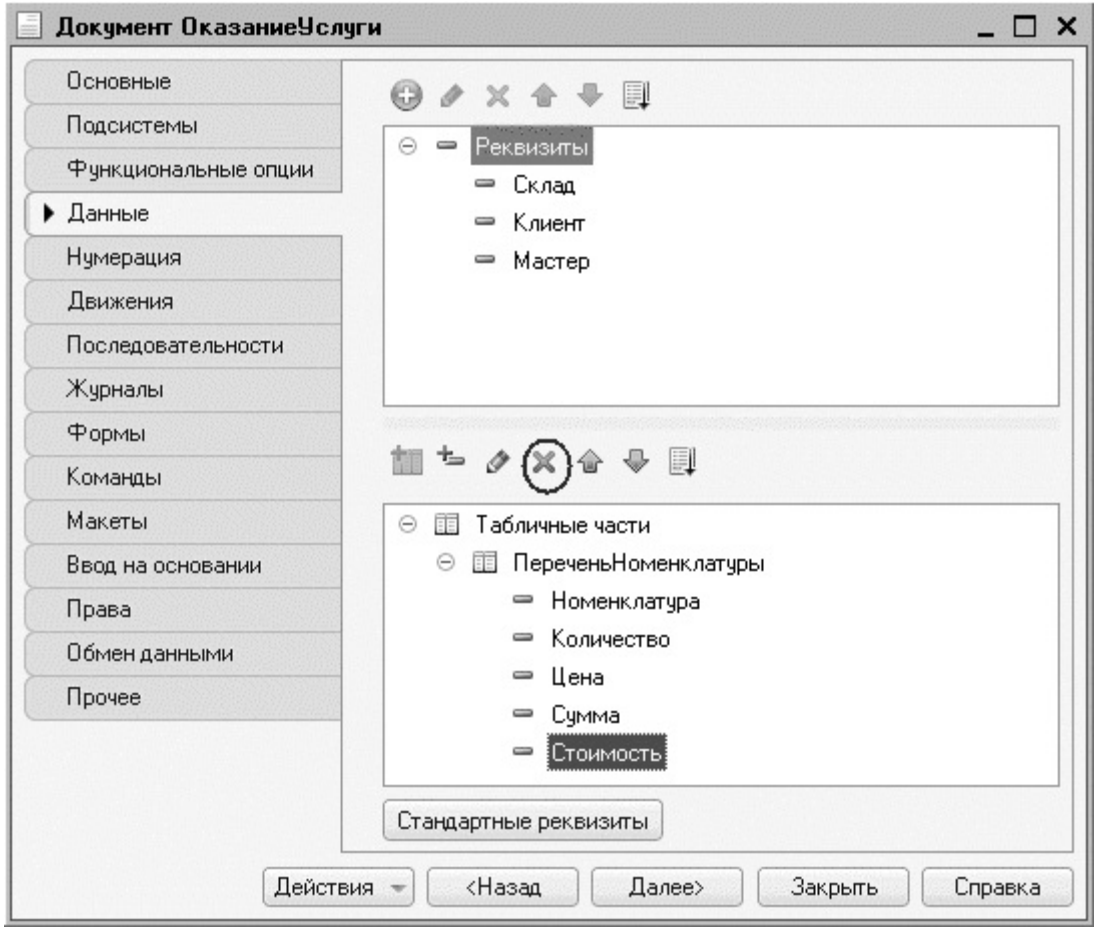


Рис. 14.13. Удаление реквизита табличной части

Также следует удалить соответствующее поле из таблицы *ПереченьНоменклатуры*, расположенной в форме.

Для этого откроем форму *ФормаДокумента* документа *ОказаниеУслуги* и в окне структуры элементов формы выделим поле таблицы *ПереченьНоменклатурыСтоимость* и нажмем кнопку *Удалить* в командной панели (рис. 14.14).

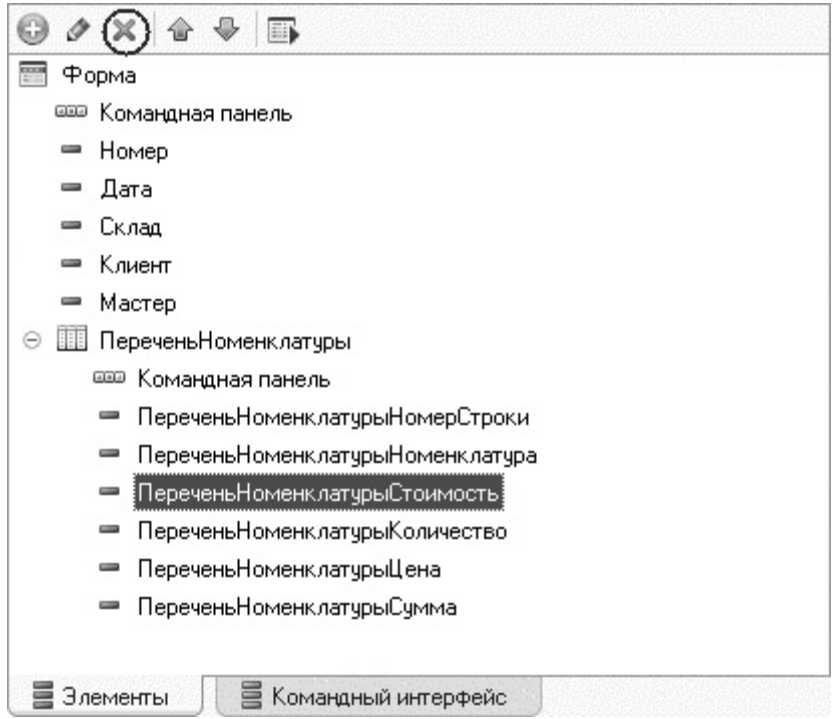


Рис. 14.14. Удаление поля табличной части

Теперь займемся запросом.

Временную таблицу мы сформируем с помощью того запроса, который у нас уже написан.

Откроем модуль документа *ОказаниеУслуги*.

В процедуре *ОбработкаПроведения()* перед созданием запроса создадим менеджер временных таблиц и укажем, что этот запрос будет использовать созданный менеджер временных таблиц (листинг 14.17).

Листинг 14.17. Использование менеджера временных таблиц

```
Движения.ОстаткиМатериалов.Записывать = Истина;
Движения.СтоимостьМатериалов.Записывать = Истина;
Движения.Продажи.Записывать = Истина;

// Создать менеджер временных таблиц
МенеджерВТ = Новый МенеджерВременныхТаблиц;

Запрос = Новый Запрос;

// Укажем, какой менеджер временных таблиц использует этот запрос
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос.Текст =
  "ВЫБРАТЬ
  |       ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
```

Теперь изменим запрос таким образом, чтобы он создавал временную таблицу, которая будет храниться в нашем менеджере временных таблиц *МенеджерВТ*.

Чтобы конструктор запроса смог открыть наш запрос, удалим из него строку (поля *Стоимость* у нас больше нет), листинг 14.18.

Листинг 14.18. Изменение запроса

```
|        МАКСИМУМ(ОказаниеУслугиПереченьНоменклатуры.Стоимость) КАК Стоимость
```

Также удалим запятую в конце предыдущей строки (листинг 14.19).

Листинг 14.19. Изменение запроса

```
|        СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК СуммаВДокументе
```

Теперь установим курсор внутрь текста запроса, например на слове *ВЫБРАТЬ,* и выполним команду контекстного меню *Конструктор запроса*. Существующий текст запроса будет показан в форме конструктора запросов (рис. 14.15).

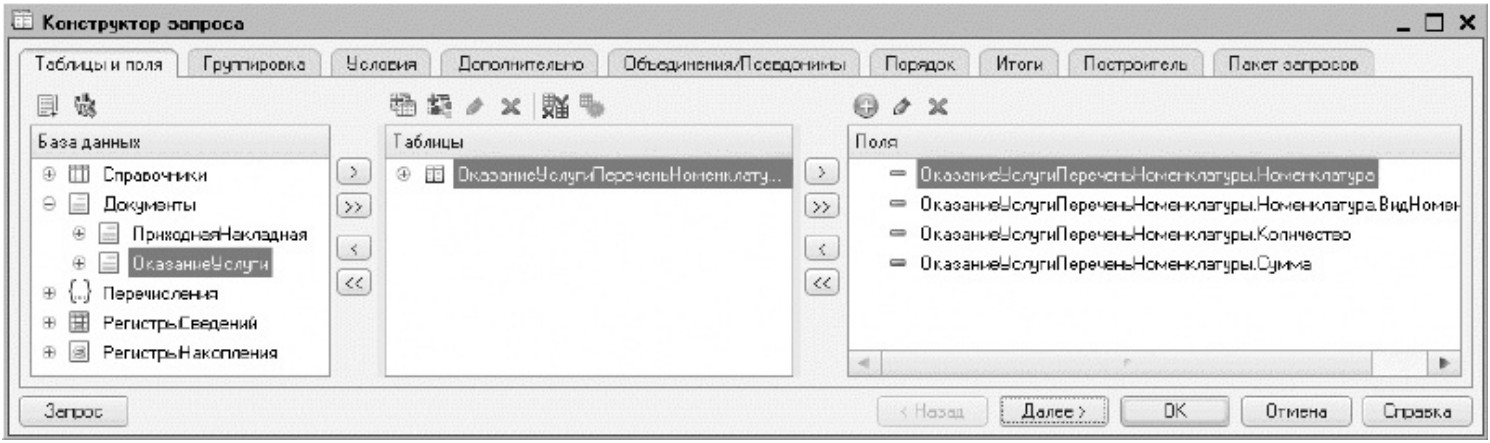


Рис. 14.15. Конструктор запроса

Чтобы результат запроса поместить во временную таблицу, перейдем на закладку *Дополнительно* и отметим пункт *Создание временной таблицы*.

Зададим имя временной таблицы – *НоменклатураДокумента* (рис. 14.16).

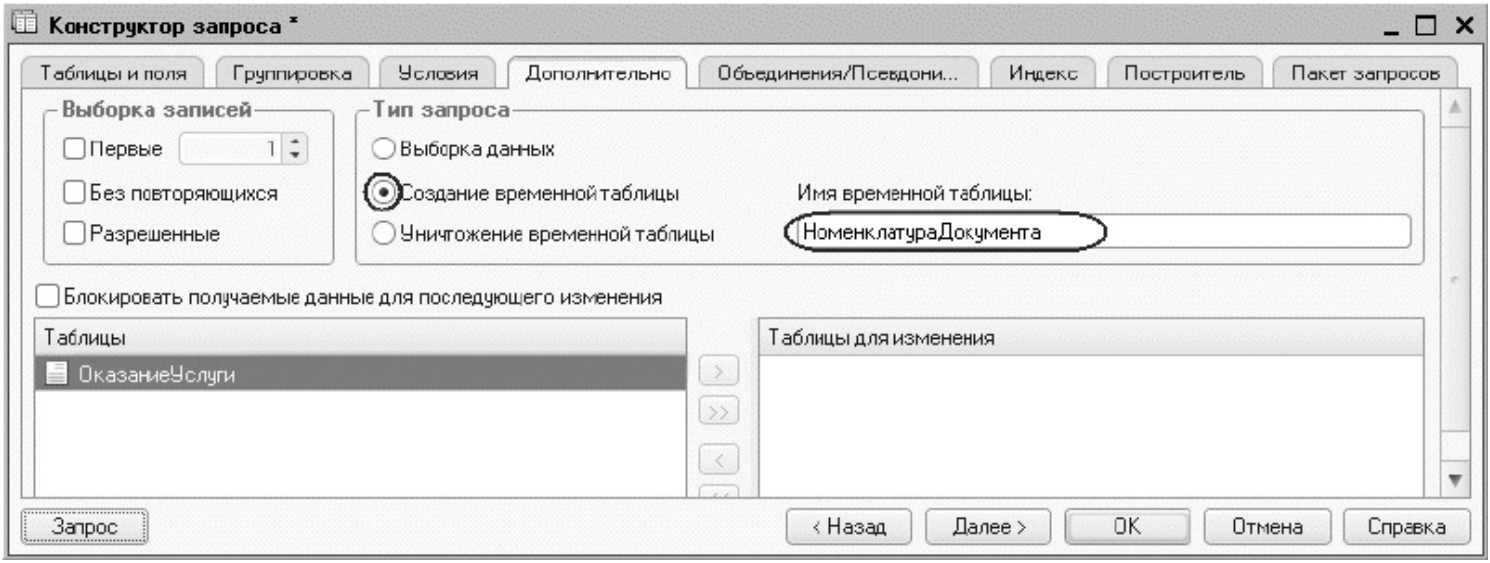


Рис. 14.16. Создание временной таблицы

Нажмем *ОК* и посмотрим, какой текст сформировал конструктор запроса (листинг 14.20).

Листинг 14.20. Текст запроса

```
"ВЫБРАТЬ
|        ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
|        ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры КАК
ВидНоменклатуры,
|        СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Количество) КАК
КоличествоВДокументе,
|        СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Сумма) КАК СуммаВДокументе
```

```
|ПОМЕСТИТЬ НоменклатураДокумента
|ИЗ
|        Документ.ОказаниеУслуги.ПереченьНоменклатуры КАК
ОказаниеУслугиПереченьНоменклатуры
|ГДЕ
|        ОказаниеУслугиПереченьНоменклатуры.Ссылка = &Ссылка
|СГРУППИРОВАТЬ ПО
|        ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
|        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры";
```

Новым здесь является только строка (листинг 14.21).

Листинг 14.21. Создание временной таблицы

```
|ПОМЕСТИТЬ НоменклатураДокумента
```

Это означает, что результат запроса будет сохранен во временной таблице *НоменклатураДокумента*.

Теперь если мы для другого запроса укажем этот же самый менеджер временных таблиц *МенеджерВТ*, то в этом другом запросе мы сможем обратиться к данным этой временной таблицы.

Таким образом, мы выполнили первую часть нашего плана – создали запрос, помещающий данные табличной части документа во временную таблицу (рис.

14.17).

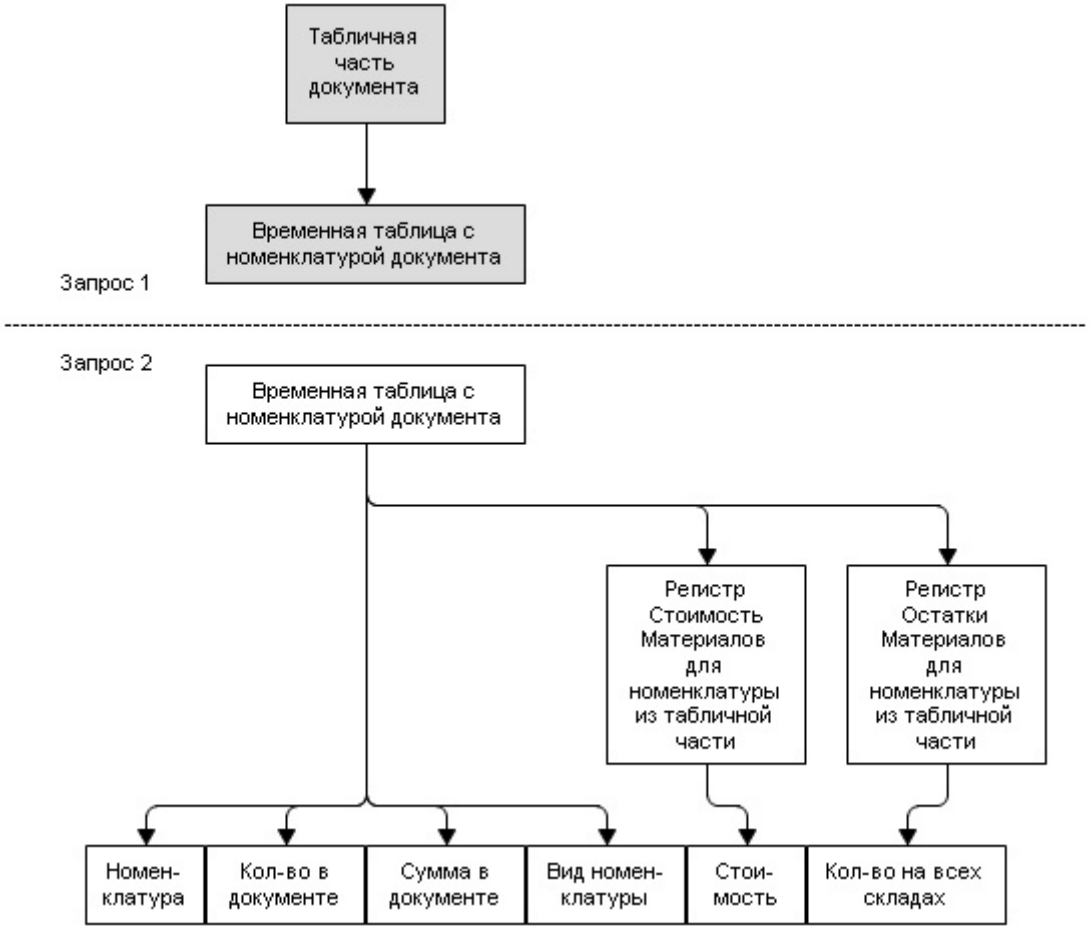


Рис. 14.17. Создание первого запроса

Теперь займемся конструированием второго запроса.

Установим курсор на следующую строку после оператора *Результат = Запрос.Выполнить();* (именно здесь выполняется создание временной таблицы) и напишем заготовку будущего запроса (листинг 14.22).

Листинг 14.22. Создание второго запроса

```
Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос2.Текст = "";
```

Мы создали новый объект *Запрос* и назначили ему тот же самый менеджер временных таблиц, чтобы иметь возможность обращаться к созданной нами ранее временной таблице.

Теперь установим курсор внутрь кавычек и выполним команду контекстного меню *Конструктор запроса*. Согласимся на создание нового запроса.

Поскольку мы собираемся выбирать данные из нашей временной таблицы, создадим в запросе описание этой временной таблицы. Для этого над списком *Таблицы* нажмем кнопку *Создать описание временной таблицы* (рис. 14.18).

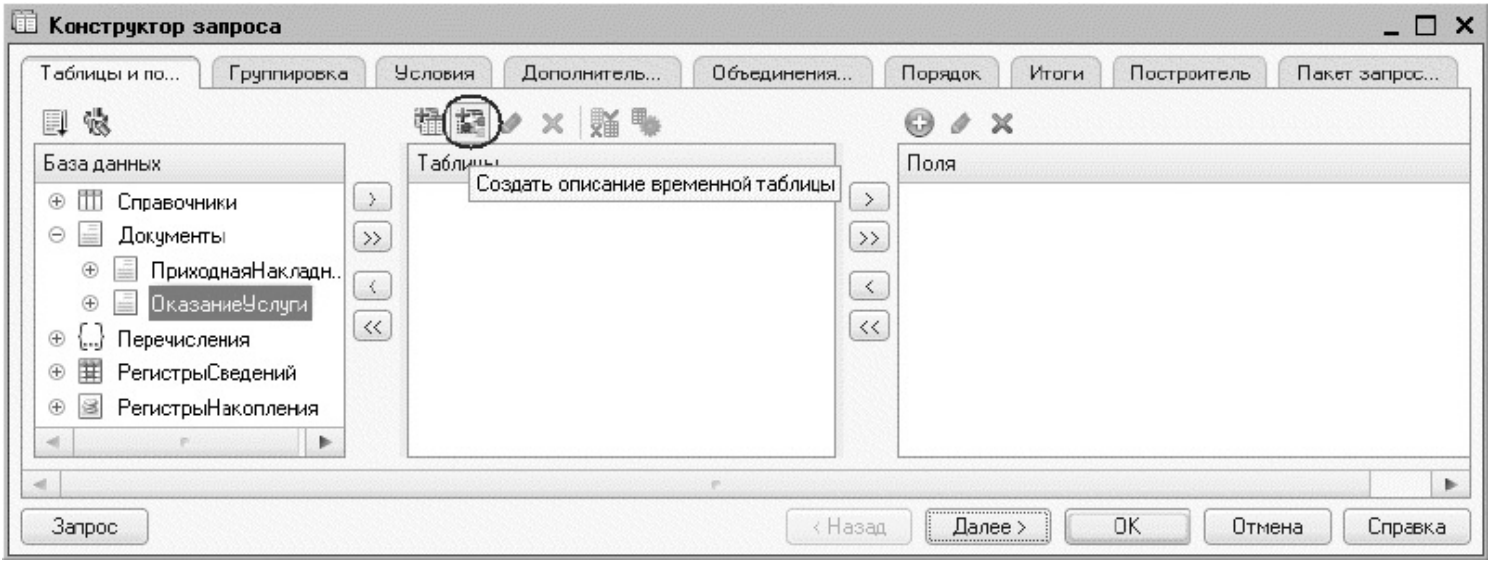


Рис. 14.18. Создание описания временной таблицы

В открывшемся окне введем имя нашей временной таблицы *НоменклатураДокумента* и добавим описание полей:

- *Номенклатура,* тип *СправочникСсылка.Номенклатура;*
- *ВидНоменклатуры,* тип *ПеречислениеСсылка.ВидыНоменклатуры;*
- *КоличествоВДокументе,* тип *Число, 15, 3;*
- *СуммаВДокументе,* тип *Число, 15, 2.*

Нажмем *ОК*.

В результате у нас получится следующее описание временной таблицы (рис. 14.19).

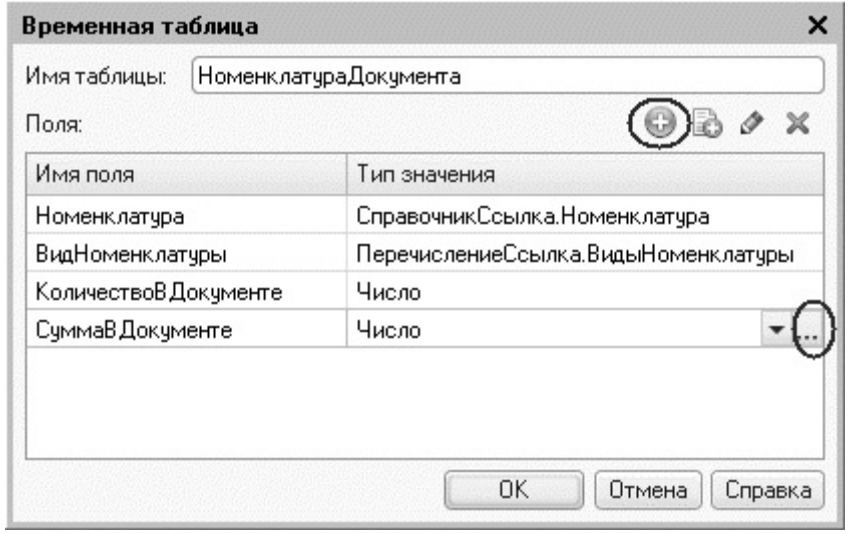


Рис. 14.19. Создание описания временной таблицы

Выберем из этой таблицы все поля и нажмем кнопку *Запрос* (рис. 14.20).

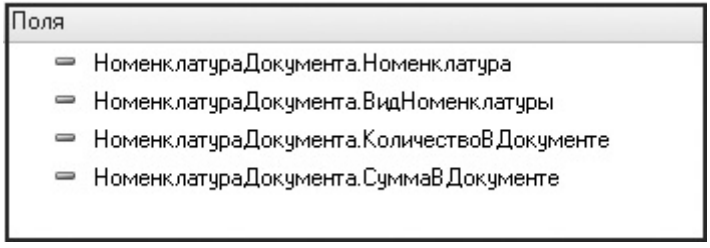


Рис. 14.20. Выбранные поля временной таблицы

Текст запроса будет иметь вид (листинг 14.23).

Листинг 14.23. Текст второго запроса

```
ВЫБРАТЬ
  НоменклатураДокумента.Номенклатура,
  НоменклатураДокумента.ВидНоменклатуры,
  НоменклатураДокумента.КоличествоВДокументе,
  НоменклатураДокумента.СуммаВДокументе
ИЗ
  НоменклатураДокумента КАК НоменклатураДокумента
```

Итак, мы создали первую часть второго запроса – выбрали информацию из временной таблицы (рис. 14.21).

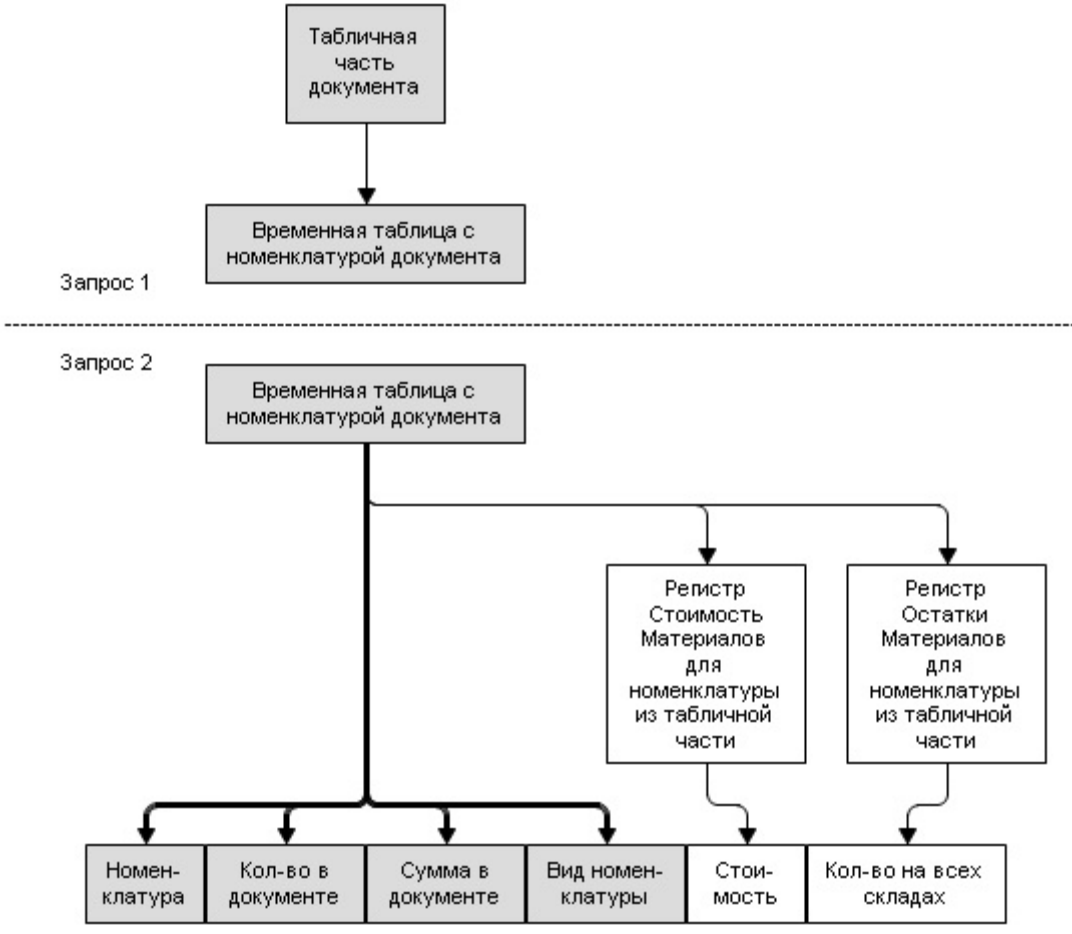


Рис. 14.21. Создание второго запроса

Теперь будем соединять эту конструкцию левыми соединениями с таблицами остатков.

Начнем со стоимости материалов.

Добавим в список таблиц запроса виртуальную таблицу *РегистрНакопления.СтоимостьМатериалов.Остатки*. Из нее выберем поле *СтоимостьОстаток*. Перейдем на закладку *Связи* и зададим связь между таблицами.

Из временной таблицы будем выбирать все записи, и поле *Номенклатура* временной таблицы должно быть равно полю *Материал* таблицы остатков (рис. 14.22).

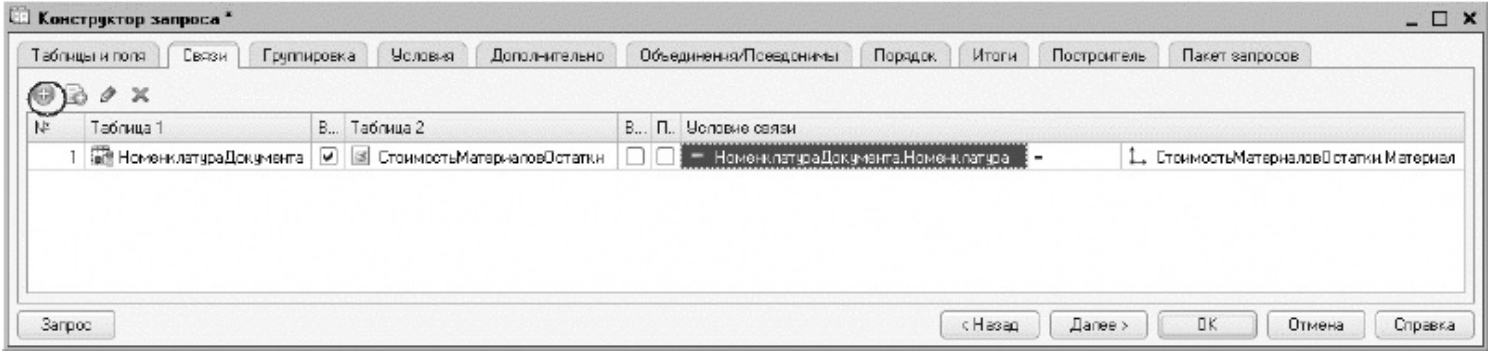


Рис. 14.22. Связи между таблицами

Нужно не забыть ограничить виртуальную таблицу только той номенклатурой, которая есть в нашей временной таблице.

Поэтому вернемся на закладку *Таблицы и поля*, выделим в списке таблиц таблицу *СтоимостьМатериаловОстатки* и нажмем кнопку *Параметры виртуальной таблицы*, расположенную над списком таблиц.

Зададим параметр *Условие* следующим образом (листинг 14.24).

Листинг 14.24. Условие виртуальной таблицы

```
Материал В (ВЫБРАТЬ НоменклатураДокумента.Номенклатура ИЗ
НоменклатураДокумента)
```

То есть материал должен быть среди номенклатуры, выбранной из временной таблицы.

Узнай больше!

Следует внимательно подходить к использованию виртуальных таблиц запросов. В частности, необходимо уделять особое внимание максимально возможному использованию параметров этих таблиц.

Например, в нашем случае можно было бы и не использовать параметр Условие, а ограничить выбранные поля уже в самом запросе, указав в условии ПО равенство номенклатуры из документа и материала из

таблицы остатков. Формально мы получили бы тот же самый результат, однако по производительности этот способ сильно отличался бы от того, который мы используем.

В самом деле, в нашем варианте виртуальная таблица предоставит нам ровно столько записей, сколько различных элементов номенклатуры содержится в проводимом документе. Если же не указывать условие, виртуальная таблица предоставит нам записи по абсолютно всем элементам номенклатуры, информация о которых есть в регистре накопления. И уже в самом нашем запросе мы будем отбирать из этой огромной массы записей лишь несколько, которые нам действительно нужны.

Очевидно, что второй вариант будет работать дольше, и время выполнения такого запроса будет зависеть в основном не от количества данных, содержащихся в документе (т. е. реального количества обрабатываемой информации), а от размера регистра накопления.

Кроме того что подобный вариант снижает производительность конфигурации, могут возникать ситуации, когда результаты, полученные одним и другим способом, будут различны. Такое, например, вполне возможно при использовании виртуальной таблицы регистра сведений СрезПоследних. Подробнее можно прочитать об этом на диске ИТС

(информационно-технологического сопровождения) в статье «Использование отборов в запросах с виртуальными таблицами».

Нажмем кнопку *Запрос* и посмотрим, какой текст запроса сформировал конструктор (листинг 14.25).

Листинг 14.25. Текст запроса

```
ВЫБРАТЬ
  НоменклатураДокумента.Номенклатура,
  НоменклатураДокумента.ВидНоменклатуры,
  НоменклатураДокумента.КоличествоВДокументе,
  НоменклатураДокумента.СуммаВДокументе,
  СтоимостьМатериаловОстатки.СтоимостьОстаток
ИЗ
  НоменклатураДокумента КАК НоменклатураДокумента
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СтоимостьМатериалов.Остатки( , Материал
    В
      (ВЫБРАТЬ
        НоменклатураДокумента.Номенклатура
      ИЗ
        НоменклатураДокумента)) КАК СтоимостьМатериаловОстатки
    ПО НоменклатураДокумента.Номенклатура = СтоимостьМатериаловОстатки.Материал
```

Тем самым мы добавили к выбранным ранее полям стоимость номенклатуры (рис. 14.23).

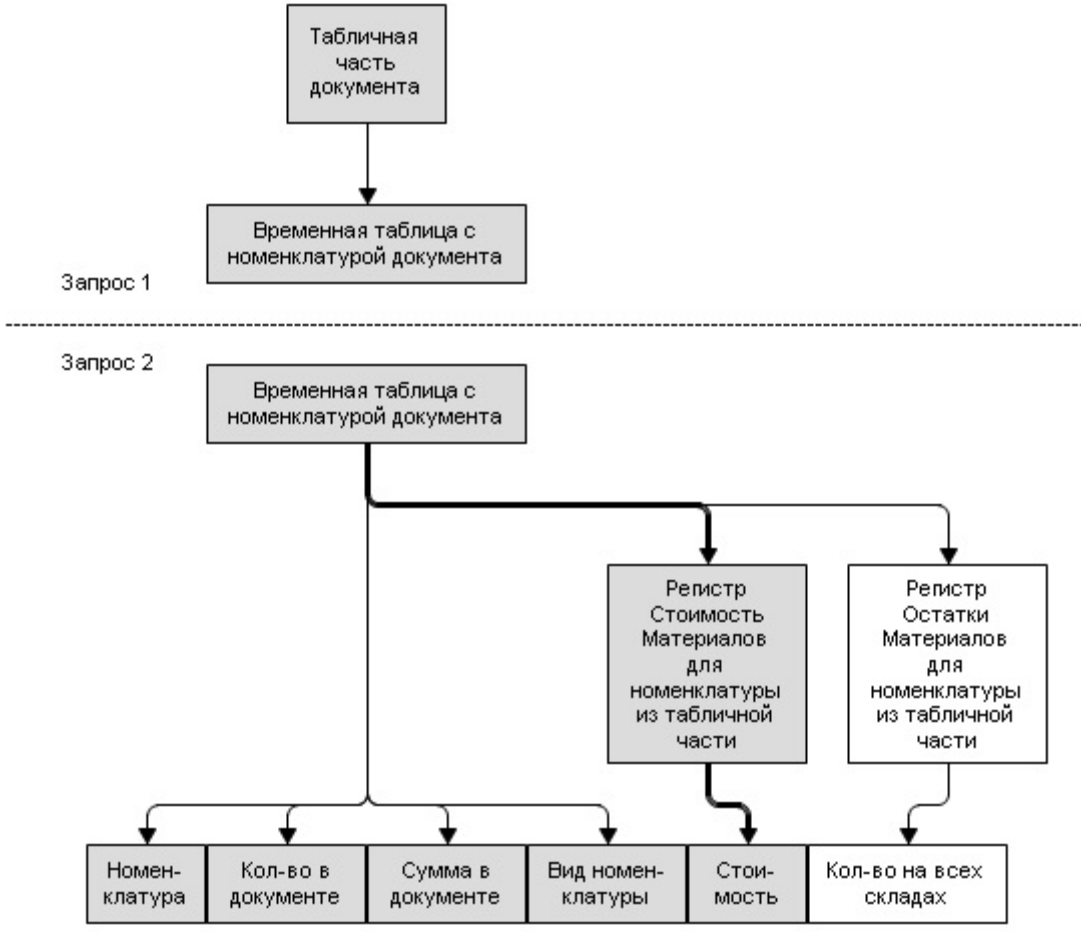


Рис. 14.23. Создание второго запроса

Теперь добавим виртуальную таблицу остатков регистра

ОстаткиМатериалов.Остатки, из которой выберем поле *КоличествоОстаток*.

Перейдем на закладку *Связи* и зададим связь между таблицами.

Из временной таблицы будем выбирать все записи, и поле *Номенклатура* временной таблицы должно быть равно полю *Материал* таблицы остатков (рис. 14.24).

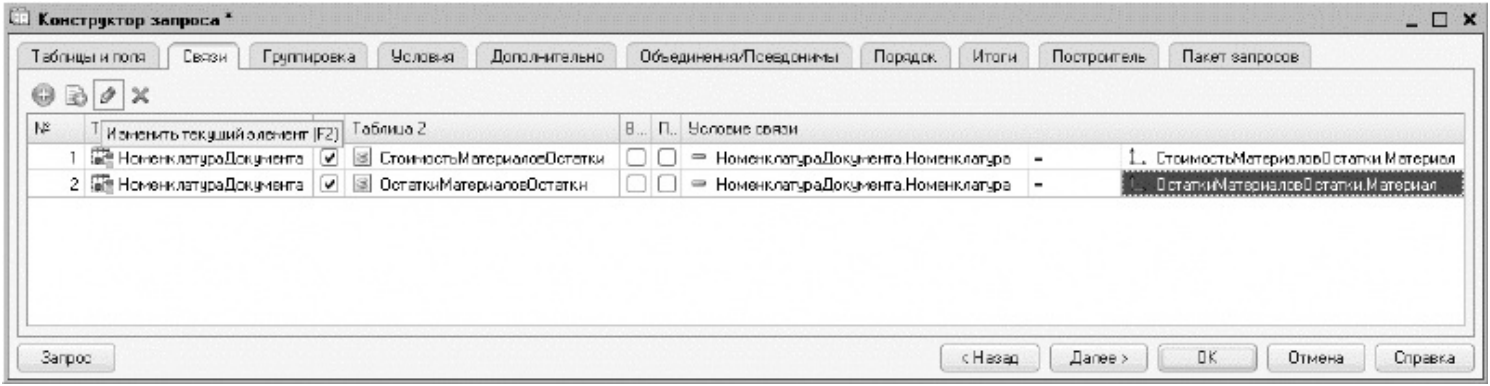


Рис. 14.24. Связи между таблицами

Также зададим параметры виртуальной таблицы *ОстаткиМатериаловОстатки*.

В параметр *Условие* внесем следующий текст (листинг 14.26).

Листинг 14.26. Условие виртуальной таблицы

```
Материал В (ВЫБРАТЬ НоменклатураДокумента.Номенклатура ИЗ
НоменклатураДокумента)
```

В результате мы получим следующий текст запроса (листинг 14.27).

Листинг 14.27. Текст запроса

```
ВЫБРАТЬ
  НоменклатураДокумента.Номенклатура,
  НоменклатураДокумента.ВидНоменклатуры,
  НоменклатураДокумента.КоличествоВДокументе,
  НоменклатураДокумента.СуммаВДокументе,
  СтоимостьМатериаловОстатки.СтоимостьОстаток,
  ОстаткиМатериаловОстатки.КоличествоОстаток
ИЗ
  НоменклатураДокумента КАК НоменклатураДокумента
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СтоимостьМатериалов.Остатки( , Материал
    В (
      ВЫБРАТЬ
        НоменклатураДокумента.Номенклатура
      ИЗ
        НоменклатураДокумента)) КАК СтоимостьМатериаловОстатки
    ПО НоменклатураДокумента.Номенклатура = СтоимостьМатериаловОстатки.Материал
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиМатериалов.Остатки( , Материал В
    (
      ВЫБРАТЬ
        НоменклатураДокумента.Номенклатура
```

```
    ИЗ
      НоменклатураДокумента)) КАК ОстаткиМатериаловОстатки
  ПО НоменклатураДокумента.Номенклатура = ОстаткиМатериаловОстатки.Материал
```

Тем самым мы добавили к выбранным ранее полям остатки номенклатуры на всех складах (рис. 14.25).

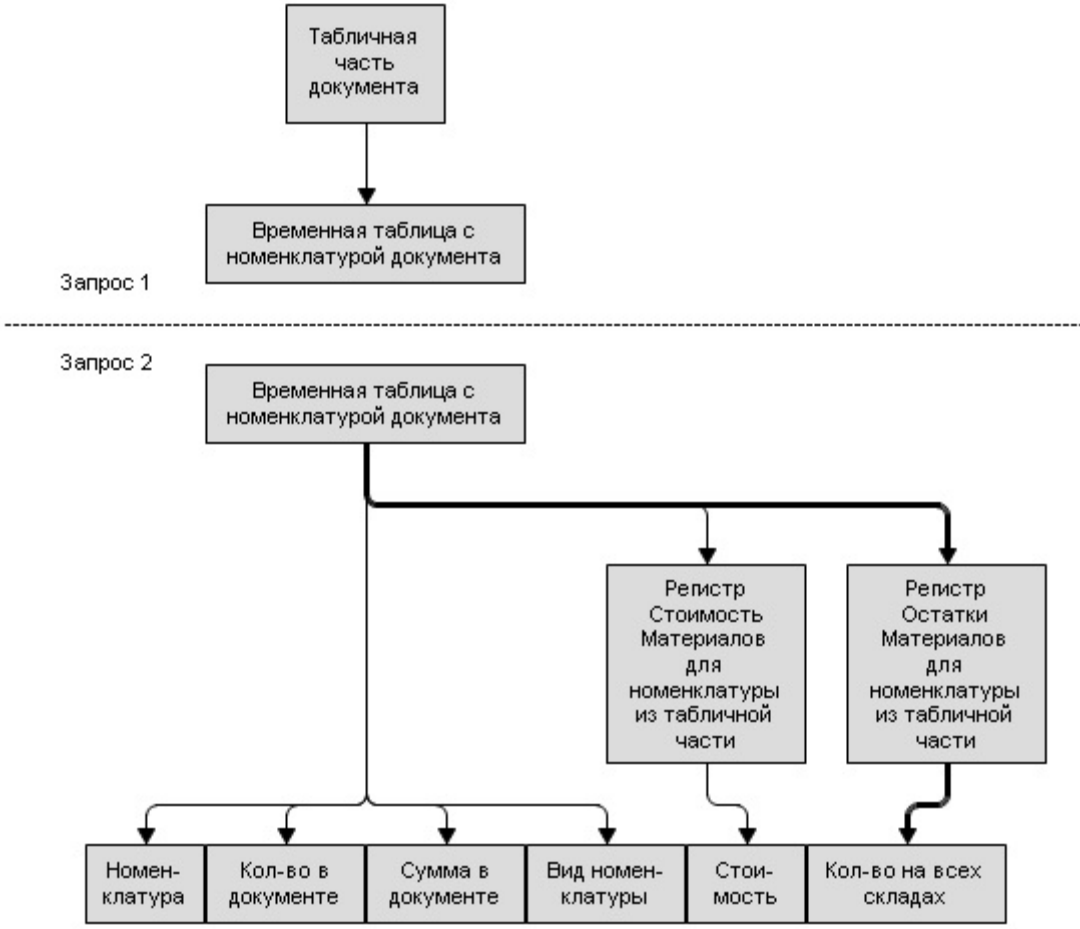


Рис. 14.25. Создание второго запроса

В заключение перейдем на закладку *Объединения/Псевдонимы* и зададим следующие псевдонимы полей (рис. 14.26):

- *СтоимостьОстаток – Стоимость;*
- *КоличествоОстаток – Количество.*

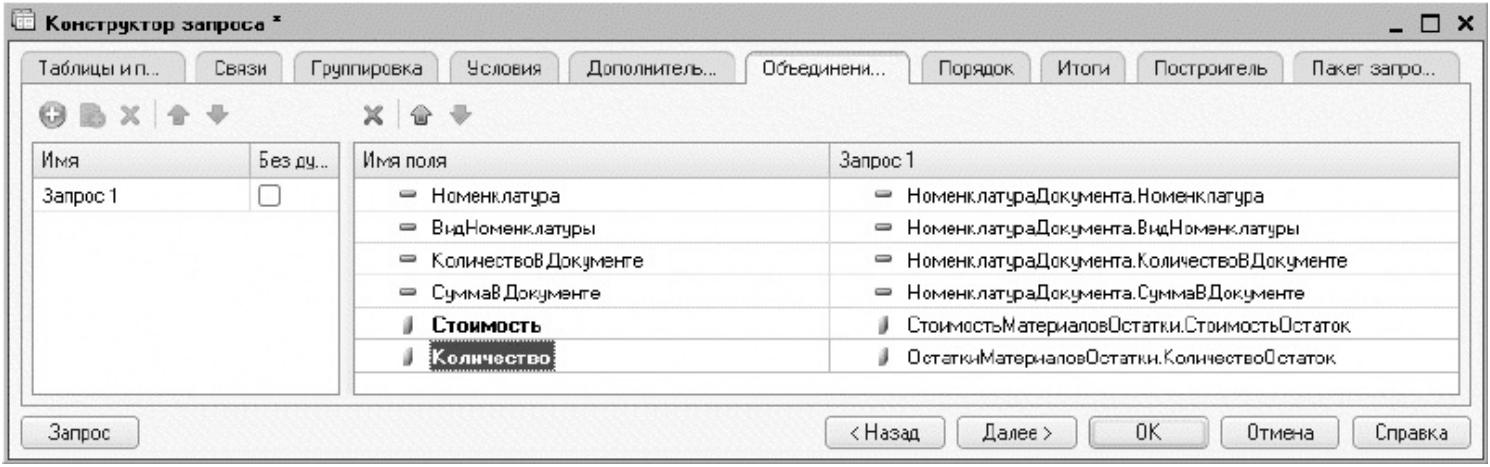


Рис. 14.26. Псевдонимы полей

В результате мы получим следующий текст запроса (листинг 14.28).

Листинг 14.28. Текст запроса

```
ВЫБРАТЬ
  НоменклатураДокумента.Номенклатура,
  НоменклатураДокумента.ВидНоменклатуры,
  НоменклатураДокумента.КоличествоВДокументе,
  НоменклатураДокумента.СуммаВДокументе,
  СтоимостьМатериаловОстатки.СтоимостьОстаток КАК Стоимость,
  ОстаткиМатериаловОстатки.КоличествоОстаток КАК Количество
```

```
ИЗ
  НоменклатураДокумента КАК НоменклатураДокумента
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СтоимостьМатериалов.Остатки( , Материал
    В (
      ВЫБРАТЬ
        НоменклатураДокумента.Номенклатура
      ИЗ
        НоменклатураДокумента)) КАК СтоимостьМатериаловОстатки
    ПО НоменклатураДокумента.Номенклатура = СтоимостьМатериаловОстатки.Материал
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиМатериалов.Остатки( , Материал В
    (
      ВЫБРАТЬ
        НоменклатураДокумента.Номенклатура
      ИЗ
        НоменклатураДокумента)) КАК ОстаткиМатериаловОстатки
    ПО НоменклатураДокумента.Номенклатура = ОстаткиМатериаловОстатки.Материал
```

В качестве последнего штриха нашей работы с запросом нужно предусмотреть тот случай, когда номенклатура в справочнике есть, но у нее нет ни остатков, ни стоимости. Это может быть, например, в том случае, когда номенклатуру создали в справочнике, но она еще не поступала в нашу фирму.

В такой ситуации левые соединения с виртуальными таблицами не вернут ничего. На языке запросов это значит, что в полях *Стоимость* и *Количество* будут значения *NULL*.

Чтобы в дальнейшем нам было удобно работать с результатом нашего запроса, сразу же в самом запросе избавимся от этих значений.

Для этого мы применим функцию *ЕСТЬNULL()* к полям *Стоимость* и *Количество*. Если значение этого поля будет *NULL*, функция вернет *0*. В остальных случаях функция вернет само значение этого поля.

Перейдем на закладку *Таблицы и поля*, выделим поле *СтоимостьМатериаловОстатки.СтоимостьОстаток* и нажмем кнопку *Изменить текущий элемент* (рис. 14.27).

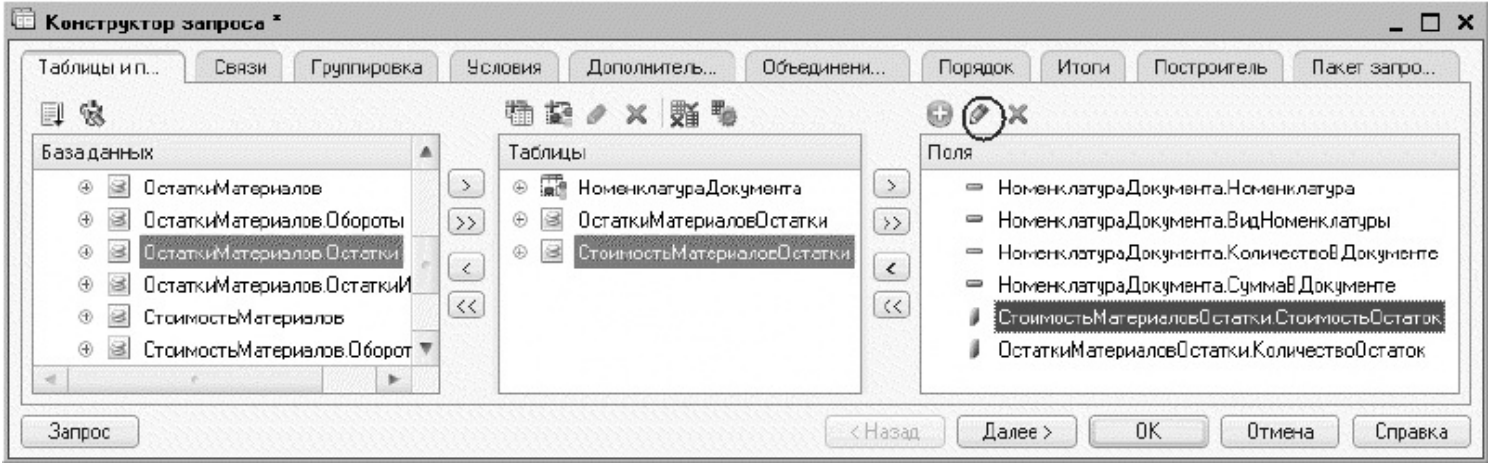


Рис. 14.27. Изменение значения поля в запросе

В открывшемся окне отредактируем значение поля следующим образом (листинг 14.29), рис. 14.28.

Листинг 14.29. Выражение для расчета поля в запросе

```
ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток, 0)
```

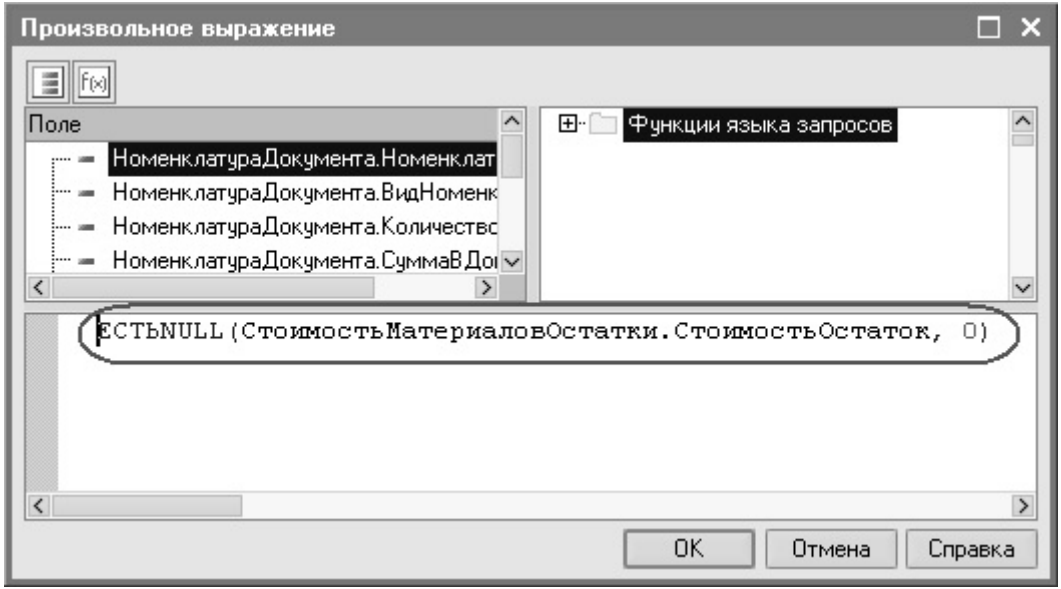


Рис. 14.28. Изменение значения поля в запросе

Аналогично поступим и с другим полем: *ОстаткиМатериаловОстатки.КоличествоОстаток*.

Нажмем *ОК* – текст запроса будет вставлен в модуль (листинг 14.30).

Листинг 14.30. Текст запроса

```
Запрос2.Текст = "ВЫБРАТЬ
   |        НоменклатураДокумента.Номенклатура,
   |        НоменклатураДокумента.ВидНоменклатуры,
   |        НоменклатураДокумента.КоличествоВДокументе,
   |        НоменклатураДокумента.СуммаВДокументе,
   |        ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток, 0) КАК
   Стоимость,
   |        ЕСТЬNULL(ОстаткиМатериаловОстатки.КоличествоОстаток, 0) КАК
   Количество
   |ИЗ
   |        НоменклатураДокумента КАК НоменклатураДокумента
   |                ЛЕВОЕ СОЕДИНЕНИЕ
   РегистрНакопления.СтоимостьМатериалов.Остатки( , Материал В (
   |                        ВЫБРАТЬ
   |                                НоменклатураДокумента.Номенклатура
   |                        ИЗ
   |                                НоменклатураДокумента)) КАК
   СтоимостьМатериаловОстатки
   |                ПО НоменклатураДокумента.Номенклатура =
   СтоимостьМатериаловОстатки.Материал
   |                ЛЕВОЕ СОЕДИНЕНИЕ
   РегистрНакопления.ОстаткиМатериалов.Остатки( , Материал В (
   |                        ВЫБРАТЬ
   |                                НоменклатураДокумента.Номенклатура
   |                        ИЗ
   |                                НоменклатураДокумента)) КАК
```

```
  ОстаткиМатериаловОстатки
  |                ПО НоменклатураДокумента.Номенклатура =
  ОстаткиМатериаловОстатки.Материал";
```

Нам останется всего лишь дописать после него оператор выполнения запроса (листинг 14.31).

Листинг 14.31. Фрагмент процедуры «ОбработкаПроведения()»

```
…
Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос2.Текст = "ВЫБРАТЬ
  |        НоменклатураДокумента.Номенклатура,
…
  |                ПО НоменклатураДокумента.Номенклатура =
  СтоимостьМатериаловОстатки.Материал";

Результат = Запрос2.Выполнить();

ВыборкаДетальныеЗаписи = Результат.Выбрать();
…
```

Теперь разберемся с записью движений.

Все операторы, которые были написаны нами ранее, будут работать без

изменений.

Единственное, что потребуется изменить, – это способ получения стоимости. Раньше мы просто брали ее из документа, теперь же нам нужно ее рассчитать на основании тех данных, которые мы получили запросом.

Стоимость материала равна частному от деления всей стоимости, полученной запросом (*Стоимость*) на общее количество материала на всех складах (*Количество*).

Но, как мы уже сказали, возможна ситуация, когда поле *Количество* у нас будет равно *0*, а на *0* делить нельзя. Поэтому сразу после начала цикла обхода результата запроса рассчитаем стоимость для текущей номенклатуры (листинг 14.32).

Листинг 14.32. Фрагмент процедуры «ОбработкаПроведения()»

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.Количество = 0 Тогда
    СтоимостьМатериала = 0;

  Иначе
    СтоимостьМатериала = ВыборкаДетальныеЗаписи.Стоимость /
    ВыборкаДетальныеЗаписи.Количество;
```

```
  КонецЕсли;

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда
…
```

Теперь заменим расчет стоимости в движениях регистров *СтоимостьМатериалов* и *Продажи* (листинг 14.33).

Листинг 14.33. Фрагмент процедуры «ОбработкаПроведения()»

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  Если ВыборкаДетальныеЗаписи.Количество = 0 Тогда
    СтоимостьМатериала = 0;

  Иначе
    СтоимостьМатериала = ВыборкаДетальныеЗаписи.Стоимость /
    ВыборкаДетальныеЗаписи.Количество;

  КонецЕсли;

  Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
  Перечисления.ВидыНоменклатуры.Материал Тогда

    // Регистр ОстаткиМатериалов Расход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
```

```
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Склад = Склад;
    Движение.Количество = ВыборкаДетальныеЗаписи.Количество;

    // Регистр СтоимостьМатериалов Расход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе *
    СтоимостьМатериала;

  КонецЕсли;

  // Регистр Продажи
  Движение = Движения.Продажи.Добавить();
  Движение.Период = Дата;
  Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
  Движение.Клиент = Клиент;
  Движение.Мастер = Мастер;
  Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;
  Движение.Выручка = ВыборкаДетальныеЗаписи.СуммаВДокументе;
  Движение.Стоимость = СтоимостьМатериала *
  ВыборкаДетальныеЗаписи.КоличествоВДокументе;

КонецЦикла;
```

Теперь все вроде бы хорошо, но остался один важный момент.

Если проводить этот документ в первый раз, то результат получится правильный.

Однако если документ был проведен ранее и мы заново решим провести его, мы получим неправильный результат.

Дело в том, что когда мы находимся в обработчике проведения документа и этот документ был уже проведен ранее, то в базе данных существуют движения этого документа.

Таким образом, читая из базы данных стоимость и остатки материалов, мы прочитаем их с учетом тех движений, которые документ выполнил ранее. А это неправильно.

Чтобы в обработчике проведения документа прочитать данные базы данных без учета предыдущих движений, которые мог выполнять документ, нужно перед чтением записать пустые наборы записей в те регистры, из которых мы собираемся читать.

В нашем случае такими регистрами являются регистры накопления *СтоимостьМатериалов* и *ОстаткиМатериалов*.

Поэтому перед выполнением второго запроса добавим следующие две строки:

Листинг 14.33а. Фрагмент процедуры «ОбработкаПроведения()»

```
…
|                ПО НоменклатураДокумента.Номенклатура =
СтоимостьМатериаловОстатки.Материал";

// Запишем пустые наборы записей, чтобы читать остатки без учета данных в
документе
Движения.СтоимостьМатериалов.Записать();
Движения.ОстаткиМатериалов.Записать();

Результат = Запрос2.Выполнить();
```

В режиме «1С:Предприятие»

Теперь нужно запустить «1С:Предприятие» в режиме отладки, перепровести все документы *Оказание услуги* и проверить, что данные правильно заносятся в регистры.

«Теория». Как быстро посмотреть результат запроса

В процессе изменения процедуры проведения документа *ОказаниеУслуги* мы с вами подошли уже к написанию довольно сложных запросов.

Зачастую возникает необходимость быстро убедиться в правильности тех данных, которые читаются из базы данных с помощью запроса, просмотреть их

в виде таблицы.

Есть простой способ сделать это в конфигураторе. Рассмотрим его на примере запроса из нашей обработки проведения.

После выполнения запроса (*Запрос2*) нужно выгрузить результат запроса в таблицу значений (*ТЗ*), листинг 14.34.

Листинг 14.34. Фрагмент процедуры «ОбработкаПроведения()»

```
…
Результат = Запрос2.Выполнить();

ТЗ = Результат.Выгрузить();

ВыборкаДетальныеЗаписи = Результат.Выбрать();
…
```

После этого установить точку останова на следующем операторе (*ВыборкаДетальныеЗаписи = Результат.Выбрать()*).

Запустить «1С:Предприятие» в режиме отладки и перепровести один из документов *Оказание услуги*. Например, документ № 2.

После того как исполнение кода будет остановлено, нужно двойным щелчком

выделить слово *ТЗ* и нажать кнопку *Вычислить выражение (Shift + F9)* на панели инструментов *Отладка конфигурации*.

Откроется окно просмотра выражений, в котором будет находиться наша таблица значений *ТЗ*.

Таблица значений является коллекцией, поэтому, чтобы просмотреть ее содержимое, выделим строку *ТЗ* в окне *Результат* и нажмем кнопку *Показать значения в отдельном окне* (или *F2*) над окном результата (рис. 14.29).

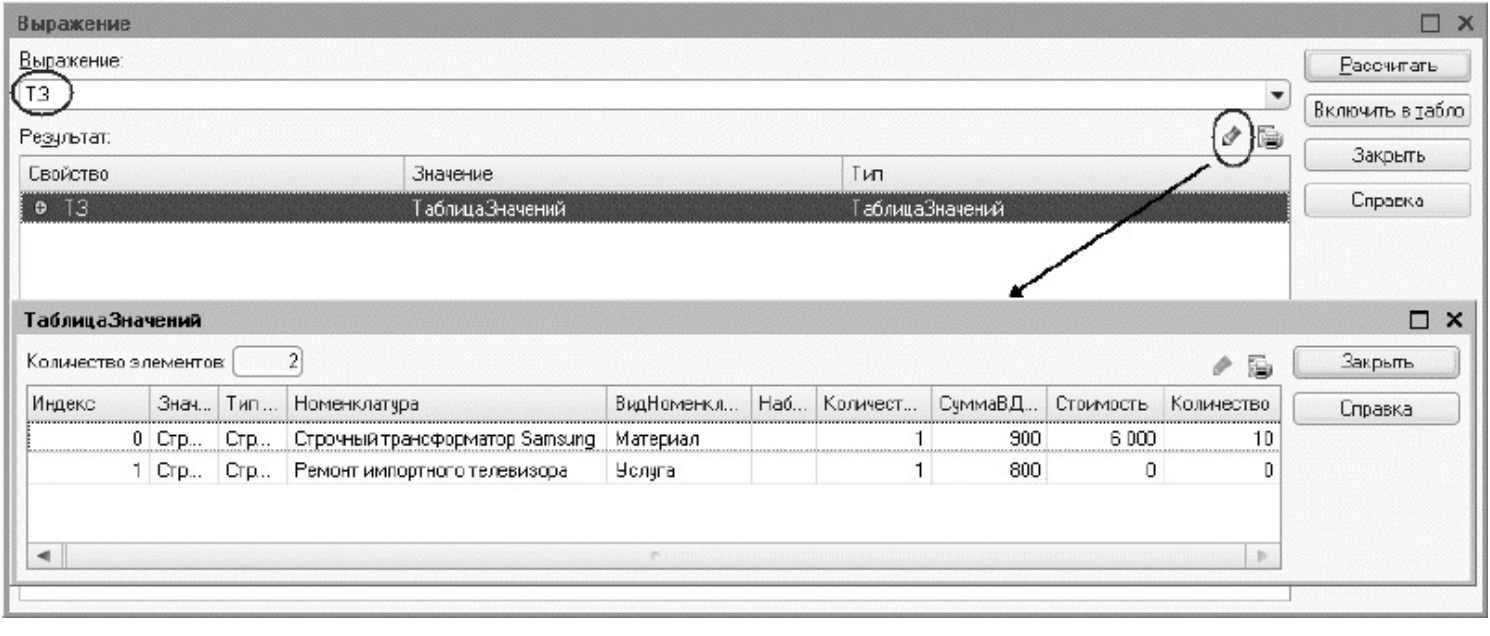


Рис. 14.29. Просмотр таблицы значений, содержащей результат запроса

Мы увидим всю таблицу значений, которая будет содержать результат выполнения нашего запроса.

С помощью кнопки *Вывести список* можно вывести эту таблицу значений, например, в табличный документ, если требуется какой-то анализ этих данных или если эти данные нужно сохранить для сравнения.

После просмотра таким образом результата запроса нужно не забыть снять точку останова в процедуре проведения документа и закомментировать (или удалить) добавленную нами строку, выгружающую результат запроса в таблицу значений, поскольку для нормальной работы документа она не нужна.

Оперативное и неоперативное проведение документов

Теперь мы поговорим о некоторых особенностях, связанных с тем, что в обработке проведения мы будем контролировать наличие достаточного количества материалов на складе при проведении документа.

Делать это мы будем не всегда. А только в том случае, когда документ проводится оперативно. Если документ проводится неоперативно, то мы это делать не будем. Так для чего же предназначено оперативное и неоперативное проведение документа? Как устроен в системе механизм оперативного

проведения документов?

При разработке конфигураций на платформе «1С:Предприятие 8» используется концепция оперативного и неоперативного проведения документов. Эта концепция используется при решении задач оперативного учета (например, при складском учете товаров) и позволяет организовать корректную работу с документами в условиях реальной действительности. Однако можно отключить этот механизм (запретить оперативное проведение документов) и реализовывать собственные алгоритмы, проверять актуальность документов на оси событий и т. д.

Эта концепция подразумевает, что работа пользователей может происходить в двух принципиально разных по своей сути режимах.

Оперативное проведение документов пользователями выполняется в режиме «реального времени», то есть отображает изменения, факты, свершающиеся в настоящее время. Оперативное проведение особенно актуально при многопользовательской работе. Поэтому при этом способе проведения документов следует осуществлять максимум проверок, способных исключить ошибки при вводе данных пользователями.

Например, при оперативном проведении следует выполнять контроль остатков на складе списываемой номенклатуры с тем, чтобы исключить одновременную

продажу одного товара несколькими продавцами.

Оперативность или неоперативность проведения документа определяется по его дате. Если дата проводимого документа совпадает с текущей датой, то система будет проводить такой документ в оперативном режиме, не задавая вопросов, и в обработке проведения об этом можно узнать, чтобы выстроить определенный алгоритм проведения документа.

Если дата проводимого документа меньше текущей, то такой документ система будет проводить в неоперативном режиме. При этом перед проведением она напомнит, что оперативно она этот документ провести не может (вдруг пользователь ошибся), и, если пользователь подтвердит, что хочет провести его неоперативно, прошлой датой, только тогда система выполнит неоперативное проведение.

Неоперативное проведение документов подразумевает отражение в базе данных фактов, которые свершились в прошлом или которые точно будут совершены в будущем. Поэтому задача неоперативного проведения документов – просто отразить в информационной базе данные о совершенных операциях.

При неоперативном проведении документов не имеет смысла производить целый ряд проверок, в частности контроль остатков. Подразумевается, что

если в процессе неоперативного проведения документов были допущены ошибки (например, списано такое количество номенклатуры, которого не было на складе на дату проведения документа), то анализ полученного состояния базы данных является отдельной задачей, не относящейся к неоперативному проведению и выполняющейся не в момент проведения документа, а тогда, когда в базе имеются достаточные данные для анализа, например, когда введены более ранние документы, приходующие товары.

Таким образом, оперативное проведение служит для того, чтобы в реальном режиме многопользовательской работы определить возможность или невозможность выполнения той или иной операции (и выполнить ее, если возможно). Неоперативное проведение предназначено для безусловного отражения в базе операций, которые уже были совершены (или точно будут совершены).

Зачастую возникает желание провести документ будущей датой, чтобы отразить какие-то события, которые точно наступят в будущем. В этом случае если документу разрешено использовать оперативное проведение, то система не даст провести такой документ будущей датой. Она просто сообщит, что не может оперативно провести такой документ, и не даст никаких вариантов выбора. Пользователю останется только поменять дату документа или на текущую дату (тогда документ будет проведен в оперативном режиме), или прошедшую (тогда документ будет проведен в неоперативном режиме после

дополнительного подтверждения от пользователя). Поэтому если логика учета подразумевает, что какой-то документ должен проводиться будущей датой, для такого документа механизм оперативного проведения должен быть отключен в метаданных (на закладке *Движения* окна редактирования объекта конфигурации).

С оперативным проведением документов связано понятие *оперативной отметки времени* и понятие *момента времени*. Прямо сейчас нам эта информация не понадобится, но раз уж зашла речь об оперативном проведении, есть подходящий случай рассказать об этом.

Понятие момента времени

Для определения положения документа на оси времени используется реквизит документа *Дата*. Дата содержит время с точностью до секунды. Это позволяет контролировать последовательность записи документов. Однако при большом объеме создаваемых документов вероятна ситуация, когда несколько документов будут иметь одинаковое значение даты (т. е. будут созданы в течение одной секунды). Как в этом случае определить последовательность созданных документов?

Для обработки подобных ситуаций было введено понятие момент времени. *Момент времени* представляет собой совокупность даты, времени и ссылки на

объект базы данных. Он позволяет однозначно идентифицировать любой объект ссылочного типа базы данных на оси событий, но имеет смысл в основном только для документов. Кроме того, момент времени позволяет идентифицировать и необъектные данные, например, записи регистров, подчиненных регистратору.

Понятие момента времени реализовано во встроенном языке при помощи универсального объекта *МоментВремени*. Этот объект имеет свойства *Дата* и *Ссылка*, которые позволяют получить «составляющие» момента времени, и один метод – *Сравнить()*, при помощи которого возможно сравнение двух моментов времени между собой. Кроме этого, объект *МоментВремени* имеет конструктор и может быть создан в явном виде для любого объекта базы данных ссылочного типа.

Для нескольких документов, имеющих одинаковую дату и время, последовательность их на оси событий определяется системой исходя из ссылок на эти документы. Она может не совпадать с последовательностью создания документов и она недоступна для изменения пользователем, то есть нельзя каким-либо образом повлиять на последовательность документов внутри одной секунды или вычислить, что один документ создан раньше, а другой – позже.

Оперативная отметка времени создается системой каждый раз при

оперативном проведении документа. Ее значение формируется исходя из текущей даты сеанса и последней созданной оперативной отметки.

Если последняя оперативная отметка меньше текущей даты сеанса, в качестве новой оперативной отметки принимается текущая дата сеанса.

Если последняя оперативная отметка равна или больше текущей даты сеанса, в качестве новой оперативной отметки принимается значение на одну секунду большее, чем старая оперативная отметка времени.

Таким образом, если у объекта конфигурации *Документ* установлено свойство оперативного проведения (рис. 14.30), последовательность действий системы будет следующей:

- при создании нового документа система будет устанавливать ему текущую дату и «нулевое» время;
- при проведении такого документа (с текущей датой) система установит в качестве даты документа оперативную отметку времени.
- если отменить проведение документа и затем провести его снова (не изменяя даты), система установит документу новую оперативную отметку времени;
- если попытаться перепровести документ, то система также автоматически

установит документу новую оперативную отметку времени и проведет его;

- при попытке проведения (или перепроведения) оперативно проводимого документа с датой, меньшей текущей, документ будет проведен неоперативно.
- если попытаться провести (или перепровести) оперативно проводимый документ будущей датой, то система не даст выполнить такое действие.

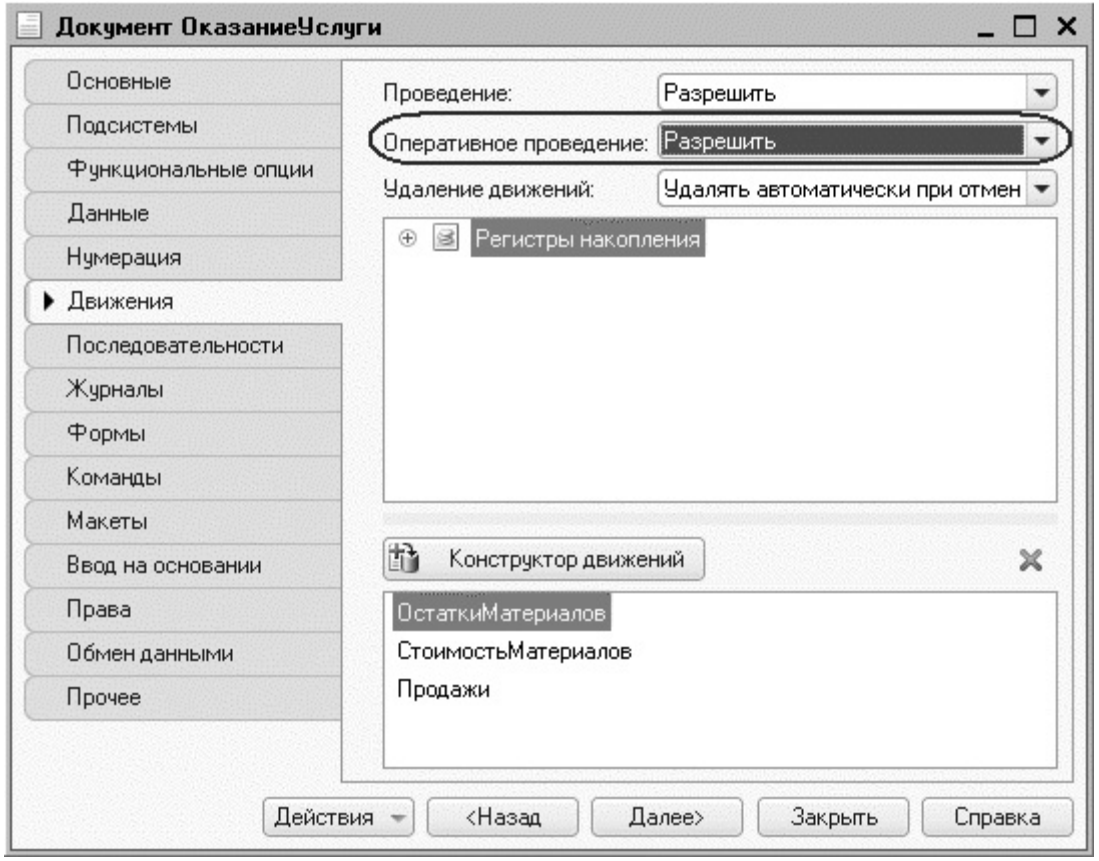


Рис. 14.30. Разрешение оперативного проведения документа

Контроль остатков

Общая методика контроля остатков при проведении документа заключается в

следующем: сначала, не глядя ни на что, нужно записать движения документа, а затем, когда движения уже записаны, прочитать из базы данных остатки.

Если появились отрицательные остатки, значит, такой документ проводить нельзя. Нужно сообщить пользователю, каких материалов не хватает, и отменить проведение документа.

Если же отрицательных остатков не появилось, тогда можно смело проводить документ.

Полдела у нас уже сделано: мы формируем движения документа и записываем их. Единственное, что нам осталось, – в случае оперативного проведения проконтролировать, что получилось, и, если появились отрицательные остатки, отменить проведение документа.

В режиме «Конфигуратор»

Сделаем заготовку. После цикла обхода результата запроса и перед концом процедуры напишем следующие строки (листинг 14.35).

Листинг 14.35. Фрагмент процедуры «ОбработкаПроведения()»

```
…
   КонецЦикла;
```

```
  Движения.Записать();

  Если Режим = РежимПроведенияДокумента.Оперативный Тогда

    // Проверить отрицательные остатки

  КонецЕсли;

КонецПроцедуры
```

Сначала мы записываем движения в регистры.

Затем определяем режим проведения документа. При выполнении процедуры *ОбработкаПроведения()* вторым параметром (*Режим*) в нее передается режим проведения документа, и значение этой переменной сравнивается со значением системного перечисления *РежимПроведенияДокумента*. В случае оперативного проведения мы будем выполнять контроль остатков.

Теперь сделаем заготовку запроса для проверки отрицательных остатков.

Так как нам придется снова получать остатки только для той номенклатуры, которая в документе, укажем, что этот запрос будет использовать тот же самый менеджер временных таблиц *МенеджерВТ* (листинг 14.36).

Листинг 14.36. Фрагмент процедуры «ОбработкаПроведения()»

```
Если Режим = РежимПроведенияДокумента.Оперативный Тогда

  // Проверить отрицательные остатки
  Запрос3 = Новый Запрос;
  Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
  Запрос3.Текст = "";

КонецЕсли;
```

Установим курсор внутрь кавычек и вызовем конструктор запроса. Подтвердим создание нового запроса.

Выберем таблицу *ОстаткиМатериалов.Остатки* и из нее два поля: *Материал* и *КоличествоОстаток*.

Зададим параметры этой таблицы. В параметре *Условие* напишем:

Листинг 14.37. Условие виртуальной таблицы

```
Материал В (
  ВЫБРАТЬ
    НоменклатураДокумента.Номенклатура
  ИЗ
    НоменклатураДокумента)
И Склад = &Склад
```

То есть мы получаем итоги только для той номенклатуры, которая содержится в нашей временной таблице, и только по складу, который указан в документе.

Затем на закладке *Условия* перенесем в список условий поле *ОстаткиМатериаловОстатки.КоличествоОстаток*, установим флажок *Произвольное* и укажем, что нас интересуют только отрицательные остатки (листинг 14.38).

Листинг 14.38. Условие запроса

```
ОстаткиМатериаловОстатки.КоличествоОстаток
```

Нажмем *ОК*. Текст запроса будет выглядеть следующим образом (листинг 14.39).

Листинг 14.39. Текст запроса

```
// Проверить отрицательные остатки
Запрос3 = Новый Запрос;
Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос3.Текст = "
  |ВЫБРАТЬ
  |        ОстаткиМатериаловОстатки.Материал,
  |        ОстаткиМатериаловОстатки.КоличествоОстаток
  |ИЗ
  |        РегистрНакопления.ОстаткиМатериалов.Остатки(     , Материал В (
```

```
   |                ВЫБРАТЬ
   |                        НоменклатураДокумента.Номенклатура
   |                ИЗ
   |                        НоменклатураДокумента)
   |                И Склад = &Склад) КАК ОстаткиМатериаловОстатки
   |ГДЕ
   |        ОстаткиМатериаловОстатки.КоличествоОстаток ;
```

Теперь осталось только установить параметр запроса, обойти результат запроса и вывести сообщения об отрицательных остатках (листинг 14.40).

Листинг 14.40. Фрагмент процедуры «ОбработкаПроведения()»

```
Запрос3 = Новый Запрос;
Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос3.Текст = "
  |ВЫБРАТЬ
  |        ОстаткиМатериаловОстатки.Материал,
…
  |ГДЕ
  |        ОстаткиМатериаловОстатки.КоличествоОстаток ;

Запрос3.УстановитьПараметр("Склад", Склад);

Результат = Запрос3.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
  Сообщение = Новый СообщениеПользователю();
```

```
    Сообщение.Текст = "Не хватает " + Строка(-
    ВыборкаДетальныеЗаписи.КоличествоОстаток) + " единиц материала """ +
    ВыборкаДетальныеЗаписи.Материал + """";
    Сообщение.Сообщить();

    Отказ = Истина;

    КонецЦикла;

КонецЕсли;
```

Кратко поясним добавленный текст.

При выполнении проверки в запрос в параметре *Склад* передается склад, указанный в документе.

Затем выполняется запрос для получения отрицательных остатков номенклатуры, содержащейся во временной таблице и на складе, указанном в параметре *Склад*.

После этого выборка записей запроса обходится в цикле, и, если есть такие записи, они выводятся в сообщениях пользователю.

При этом параметру *Отказ* процедуры проведения документа присваивается значение *Истина*, то есть документ не проводится, начатая транзакция

отменяется, и состояние данных, измененных в процессе проведения, возвращается в исходное, до начала проведения документа.

Блокировка данных, которые читаются и изменяются при проведении

Казалось бы, это все? Но нет, есть очень важный момент, о котором мы не позаботились.

Сейчас схема нашей процедуры такова:

1. Выполняем первый запрос с именем *Запрос*. В результате мы формируем временную таблицу из перечня номенклатуры документа.
2. Выполняем второй запрос с именем *Запрос2*. В результате мы читаем стоимость и остатки для номенклатуры, содержащейся в табличной части документа.
3. Записываем движения регистров (*Движения.Записать()*).
4. Выполняем третий запрос с именем *Запрос3*. Тем самым мы проверяем наличие отрицательных остатков.

Обратите внимание, что, начиная с выполнения второго запроса и до конца процедуры, нам необходимо обеспечить неизменность стоимости и остатков

номенклатуры, с которой мы работаем, и запретить другим транзакциям даже читать эти данные. Сама система, естественно, заблокирует изменение этих данных, но лишь начиная с того момента, когда мы запишем движения.

Однако может возникнуть следующая ситуация. Выполняя второй запрос, мы прочитали, что есть 2 шт. некоторого материала. И другая транзакция (другой пользователь), которая собирается списывать материалы, тоже прочитала, что есть 2 шт. этого материала. После этого мы записали движения, и система заблокировала эти данные. Другая транзакция ждет, когда мы освободим данные. Мы провели документ, списали 2 шт. материала и освободили данные. Другая транзакция пытается тоже списать 2 шт. материала, но его уже нет!

Аналогичная ситуация может возникнуть и между п. 3 и п. 4, в результате чего контроль остатков будет работать неверно.

Поэтому, чтобы не происходило таких коллизий, нам необходимо заблокировать остатки от чтения другими транзакциями еще до выполнения второго запроса. То есть прежде чем читать что-то, что мы собираемся изменять, нужно запретить чтение этих данных другими транзакциями до тех пор, пока мы не закончим свои изменения (или пока не откажемся от проведения документа).

В режиме «Конфигуратор»

Как это сделать? Хороший вопрос. Давайте посмотрим на свойство нашей конфигурации *Режим управления блокировкой данных*. Оно установлен в значение *Управляемый* (рис. 14.31).

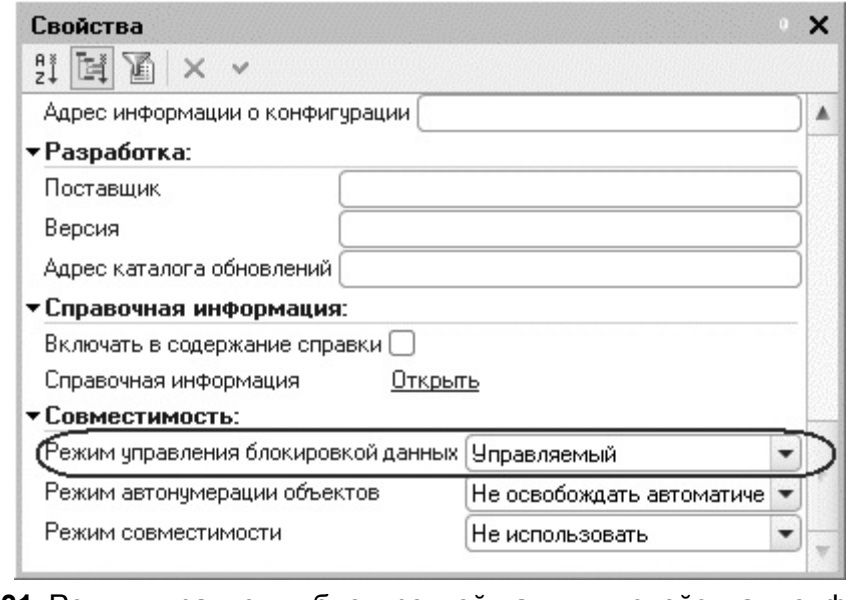


Рис. 14.31. Режим управления блокировкой данных в свойствах конфигурации

Это значит, что нам нужно использовать управляемые блокировки, которые устанавливаются средствами встроенного языка.

Необходимо заблокировать те данные, которые мы собираемся читать и впоследствии изменять. Для этого у наборов записей регистров есть свойство

БлокироватьДляИзменения, которым мы и воспользуемся.

Вставим этот код перед записью пустых наборов записей (листинг 14.41).

Листинг 14.41. Фрагмент процедуры «ОбработкаПроведения()»

```
…
|                ПО Номенклатура.Номенклатура =
СтоимостьМатериаловОстатки.Материал";

// Установим необходимость блокировки данных в регистрах СтоимостьМатериалов и
ОстаткиМатериалов
Движения.СтоимостьМатериалов.БлокироватьДляИзменения = Истина;
Движения.ОстаткиМатериалов.БлокироватьДляИзменения = Истина;

// Запишем пустые наборы записей, чтобы читать остатки без учета данных в
документе
Движения.СтоимостьМатериалов.Записать();
Движения.ОстаткиМатериалов.Записать();

Результат = Запрос2.Выполнить();
```

Управляемая блокировка будет установлена в момент записи этих наборов записей, то есть как раз перед выполнением второго запроса. Что нам и требовалось.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим работу нового обработчика события *ОбработкаПроведения*, перепроведя все документы *Оказание услуги*.

В результате все работает точно так же, с точки зрения пользователя, но проведение документов организовано методически правильно и более эффективно с точки зрения доступа к данным.

«Теория». Устройство кеша

В разделе «Особенности использования ссылочных данных» мы упомянули о том, что в платформе есть некий кеш, который хранит в себе данные объектов, читаемых из базы данных. Теперь расскажем о работе этого механизма подробнее.

Система «1С:Предприятие 8» использует механизм кеширования данных объектов, считанных из базы данных при использовании объектной техники.

Таким образом, для получения реквизитов какого-либо объекта через ссылку выполняется обращение к кешу объектов, расположенному в оперативной памяти.

Кеш объектов состоит из двух частей: транзакционного кеша и обычного кеша. В зависимости от того, происходит ли обращение в рамках транзакции или нет, в действие вступает тот или иной кеш (рис. 14.32).

Рис. 14.32. Кеш объектов

Все данные, находящиеся в кеше, предназначены только для чтения (*ReadOnly*). Таким образом, чтение любых данных, получаемых через ссылку, выполняется только через кеш объектов, а запись – механизмами самих программных объектов.

Обычный кеш

Если при обращении к обычному кешу требуемых данных в нем нет, то выполняется чтение данных объекта из базы данных и сохранение их в кеше. Уникальным идентификатором для кеша в данном случае будет являться ссылка на объект базы данных. Поэтому данные каждого считанного объекта могут существовать в кеше в одном из двух видов: либо все данные объекта, либо представление объекта.

Таким образом, если мы обратимся к кешу для получения представления объекта и в кеше есть информация для нашей ссылки, данные будут взяты из кеша (если в кеше весь объект, нужное представление будет получено из данных объекта).

Если в кеше нет информации для нашей ссылки, из базы данных в кеш будут считаны только поля, необходимые для формирования представления объекта.

Если мы обратимся к кешу для получения реквизита объекта, и в кеше есть информация для нашей ссылки, дальнейшие действия будут зависеть от того, что находится в кеше.

Если в кеше весь объект, значение реквизита будет получено из кеша. Если в кеше представление объекта, оно будет удалено из кеша, и в кеш будут

считаны все данные объекта. Если же при получении реквизита объекта в кеше нет информации для нашей ссылки, из базы данных будут считаны все поля объекта.

Считанные данные будут находиться в кеше до тех пор, пока не наступит одно из следующих событий:

- считанные данные будут вытеснены из кеша другими считанными данными других объектов (переполнение кеша);
- при очередном обращении к кешу окажется, что считанные данные были изменены в базе данных;
- закончится интервал времени в 20 минут.

Все считанные данные помещаются в последовательную очередь, и, поскольку объем кеша ограничен, наиболее старые данные будут вытесняться из кеша последними считанными.

При повторном обращении к кешу за данными уже считанного объекта будет анализироваться интервал времени, прошедший с момента появления данных в кеше.

Если обращение происходит в пределах 20 секунд после поступления данных в кеш, данные считаются верными (валидными). Если интервал превысил 20

секунд, будет выполняться проверка на соответствие версии данных, хранящихся в кеше, версии данных, находящихся в базе данных.

Если окажется, что версии данных не совпадают (т. е. произошло изменение данных в базе данных), данные, находящиеся в кеше, будут удалены из него, и будет выполнено повторное считывание данных из базы данных. Начиная с этого момента, идет отсчет следующего 20-секундного интервала валидности этих данных.

Кроме всех вышеперечисленных событий считанные данные будут удалены из кеша по истечении 20 минут после их последнего считывания из базы данных.

Таким образом, при последовательном выполнении двух операторов (листинг 14.42), где *Номенклатура* – это ссылка на объект справочника, на выполнение второго оператора будет тратиться гораздо меньше времени, поскольку в первом случае будет выполняться обращение к базе данных, а во втором – чтение из оперативной памяти (кеша объектов).

Листинг 14.42. Последовательность операторов

```
A = Номенклатура.Наименование;
B = Номенклатура.ВидНоменклатуры;
```

Транзакционный кеш

Если обращение к данным происходит в рамках транзакции, то оно переадресуется транзакционному кешу. В рамках транзакции в «1С:Предприятии 8» выполняются все операции, приводящие к изменению данных в базе данных. Например, в рамках транзакции выполняется обработка проведения документа.

Транзакция – это неделимая последовательность манипулирования данными, переводящая базу данных из одного целостного состояния в другое. Если по каким-то причинам одно из действий транзакции невыполнимо, база данных возвращается в то состояние, которое было до начала транзакции (происходит откат транзакции – *Rollback*).

Транзакционный кеш по сути представляет собой ту же последовательную очередь, что и обычный кеш. Разница заключается в том, что все данные, находящиеся в транзакционном кеше, являются валидными (гарантированно актуальными).

При считывании данных в транзакционный кеш устанавливается блокировка на данные в базе данных, поэтому они гарантированно не могут быть изменены до окончания транзакции.

Транзакционный кеш хранит считанные данные до тех пор, пока они не будут вытеснены более поздними, или пока не закончится транзакция. По окончании транзакции кеш очищается, однако действия, выполняемые при этом, зависят от состояния завершения транзакции.

Если транзакция завершена успешно (*Commit*), данные всех объектов, содержащиеся в транзакционном кеше, переносятся в обычный кеш, а транзакционный кеш очищается (рис. 14.33).

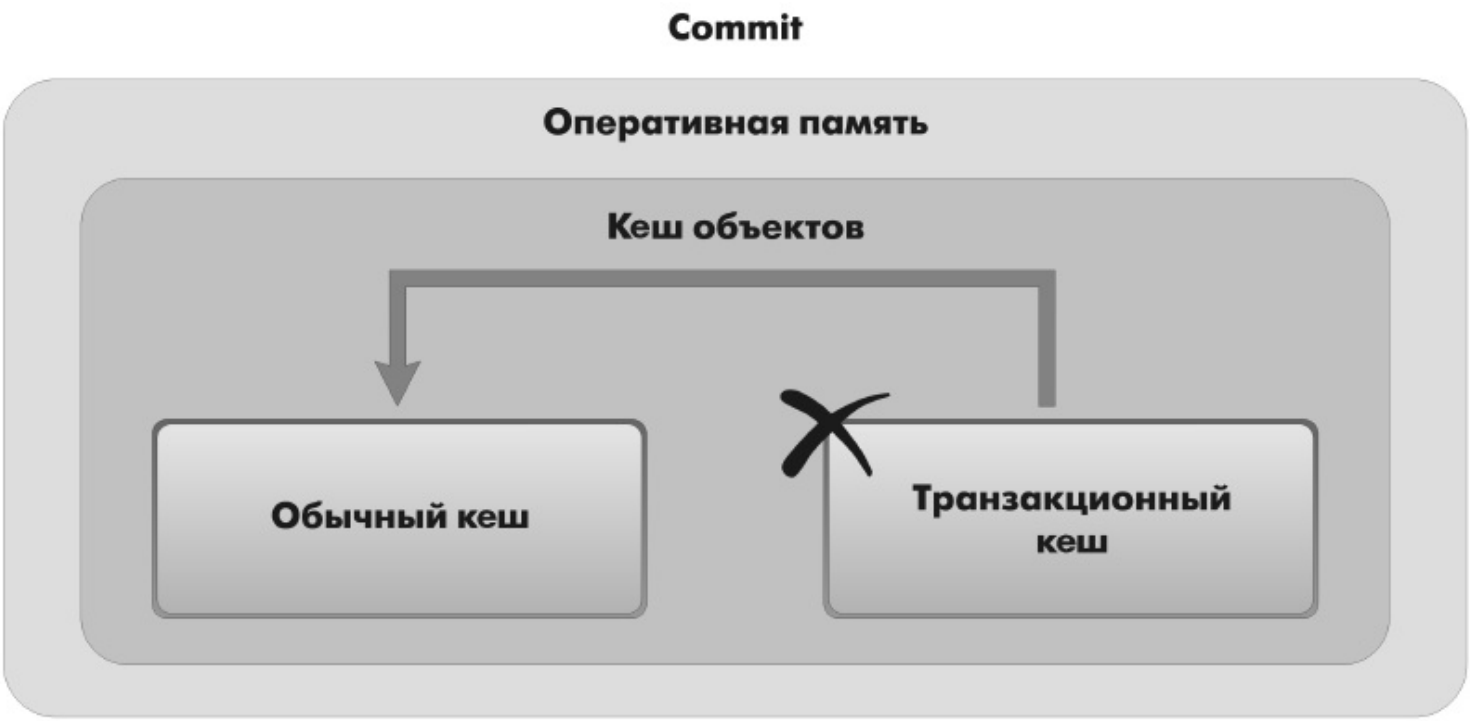


Рис. 14.33. Транзакция завершена успешно

Если был выполнен отказ от изменений (*Rollback*), то просто очищается транзакционный кеш (рис. 14.34).

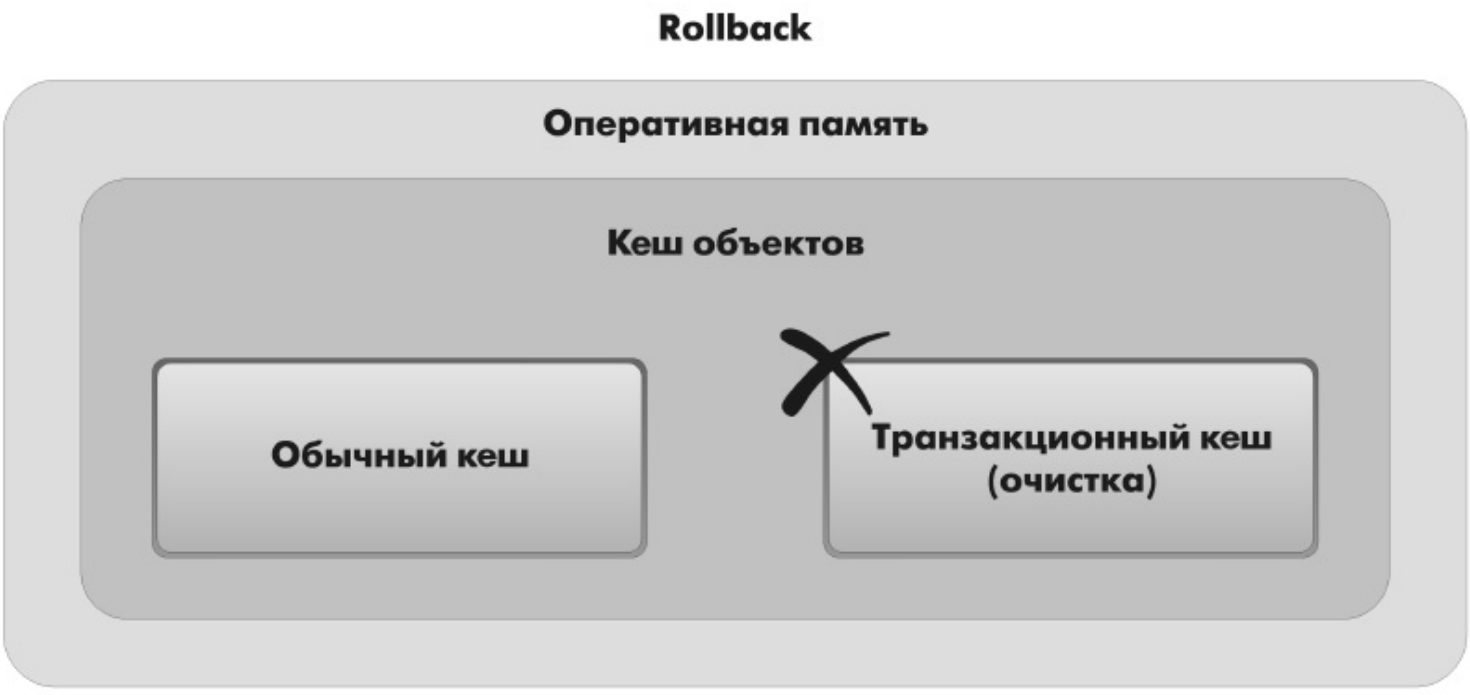


Рис. 14.34. Очистка транзакционного кеша при отказе от изменений

Контрольные вопросы

- *Как система «1С:Предприятие» выполняет обращение к ссылочным данным.*
- *Как используется кеш объектов.*
- *Почему для доступа к массивам данных информационной базы*

предпочтительнее использовать запросы.

- *Что такое момент времени.*
- *Чем отличается оперативное проведение документов от неоперативного.*
- *Что такое оперативная отметка времени.*
- *Как запросом получить остатки регистра накопления.*
- *На что следует обращать внимание при задании параметров виртуальных таблиц запросов.*
- *Почему при неоперативном проведении документов не нужно контролировать остатки.*
- *Что такое временные таблицы и зачем их использовать.*
- *Что такое менеджер запросов.*
- *Как и зачем можно использовать временные таблицы в параметрах виртуальных таблиц.*
- *Как программно блокировать данные.*
- *Как посмотреть в отладчике результат запроса.*

Занятие 15 (2:50). План видов характеристик

Продолжительность

Ориентировочная продолжительность занятия – 2 часа 50 минут.

На этом занятии мы познакомимся с новым объектом конфигурации – *План видов характеристик* – и узнаем, каким образом можно использовать этот объект для расширения возможностей нашей конфигурации.

Постановка задачи

Задача, которую мы перед собой поставим, будет заключаться в следующем: мы создадим механизм, позволяющий пользователю произвольным образом описывать материалы и, что самое главное, вести учет в разрезе всех тех описаний, которые могут быть заданы пользователем.

Описывать материалы пользователь сможет следующим образом: для каждого материала будет возможность создать некоторые (произвольные) характеристики этого материала (например, цвет, производитель и пр.). Затем, при поступлении материалов, можно будет задать конкретные значения интересующих характеристик (например, при поступлении электрических кабелей можно будет указать, что они белого цвета и их сечение равно 2,5 мм?, а при поступлении резиновых шлангов указать, что они черного цвета и

произведены на фирме «Fagumit Sp. z o.o.»).

В дальнейшем всегда можно будет получить информацию о том, сколько и каких материалов есть у нас, скажем, белого цвета или сколько было израсходовано черных резиновых шлангов.

Поскольку заранее неизвестно, какими именно характеристиками пользователь захочет описать тот или иной материал, мы должны предоставить ему некоторый механизм, позволяющий создавать любые характеристики и, что самое важное, указывать, какой тип значения должен быть у этих характеристик. Тогда при задании значений определенной характеристики пользователь сможет выбирать значения строго в соответствии с указанным типом.

Такую возможность описания характеристик как раз и обеспечивает объект конфигурации План видов характеристик, с которым мы сейчас познакомимся.

Что такое план видов характеристик

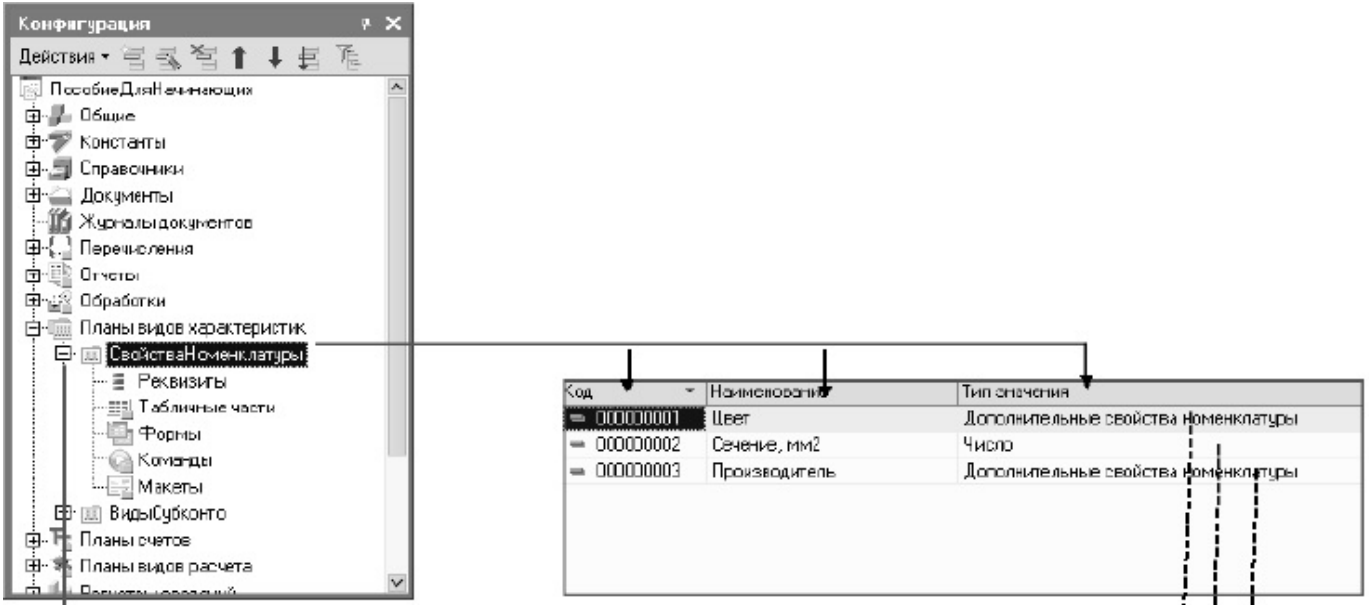
Объект конфигурации *План видов характеристик* предназначен для описания структуры хранения информации о характеристиках, создаваемых пользователем. На основе объекта конфигурации *План видов характеристик* платформа создает в базе данных набор таблиц, в которых будет храниться

информация о существующих видах характеристик и типе значения характеристики каждого вида.

В сущности, план видов характеристик очень напоминает справочник, однако имеет более узкую «специализацию»: хранит, по сути, информацию только о том, какими видами характеристик может описываться какой-либо объект базы данных.

План видов характеристик состоит из видов характеристик. Каждый вид характеристики обязательно описывается наименованием и типом значения.

Разработчик и, что самое важное, пользователь могут задать в нем любое необходимое им количество видов характеристик (рис. 15.1).



Код	Наименование	Тип значения
000000001	Цвет	Дополнительные свойства номенклатуры
000000002	Сечение, мм2	Число
000000003	Производитель	Дополнительные свойства номенклатуры

Рис. 15.1. План видов характеристик в конфигураторе, в базе данных и в режиме «1С:Предприятие»

Для того чтобы разработчик мог задать некий набор возможных типов значений, которые могут принимать виды характеристик, у объекта конфигурации *План видов характеристик* существует свойство *Тип значения характеристик*.

Это свойство определяет составной тип данных, куда входят все типы, которые могут понадобиться при указании типа значения характеристики (рис. 15.2).

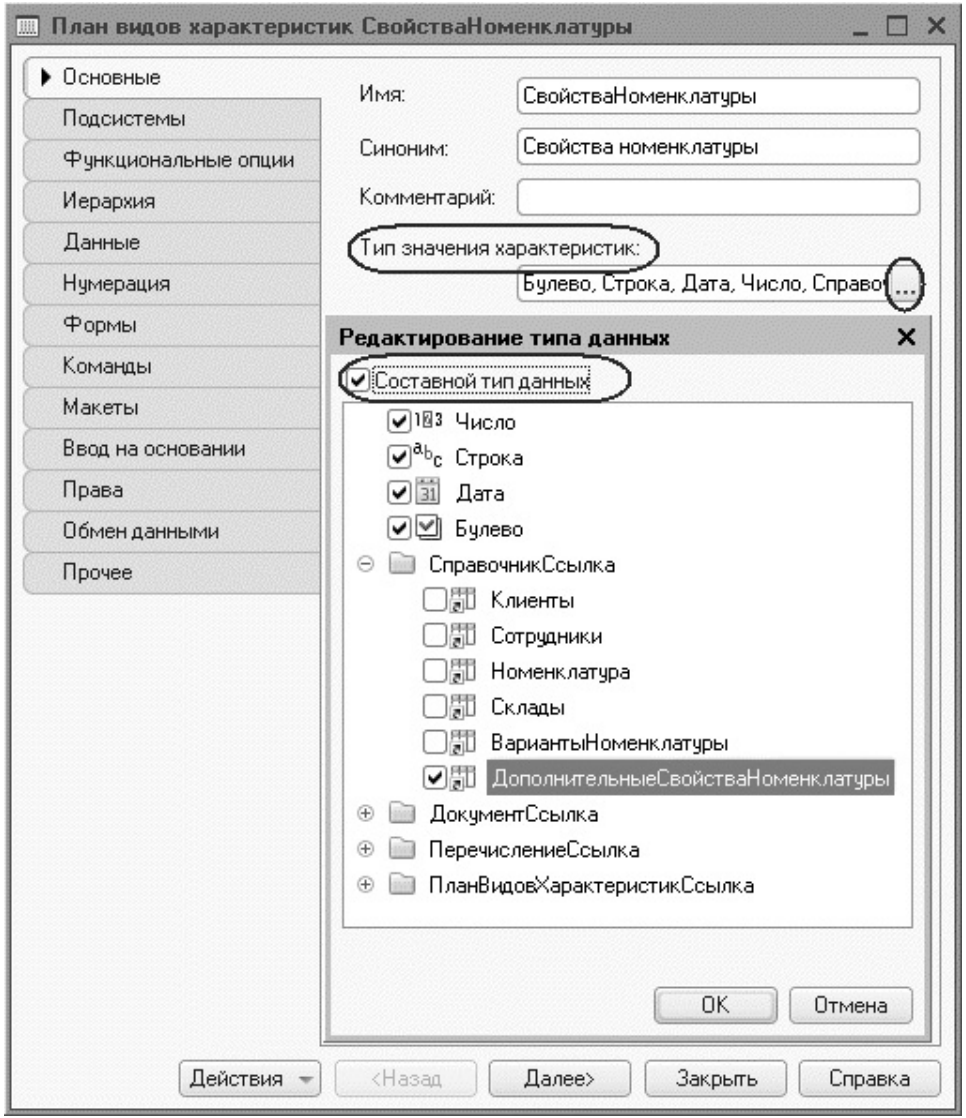

План видов характеристик СвойстваНоменклатуры
Основные
Подсистемы
Функциональные опции
Иерархия
Данные
Нумерация
Формы
Команды
Макеты
Ввод на основании
Права
Обмен данными
Прочее
Имя:
СвойстваНоменклатуры
Синоним:
Свойства номенклатуры
Комментарий:
Тип значения характеристик:
Булево, Строка, Дата, Число, Справо
...
Редактирование типа данных
Составной тип данных
Число
Строка
Дата
Булево
СправочникСсылка
Клиенты
Сотрудники
Номенклатура
Склады
ВариантыНоменклатуры
ДополнительныеСвойстваНоменклатуры
ДокументСсылка
ПеречислениеСсылка
ПланВидовХарактеристикСсылка
OK
Отмена
Действия
<Назад
Далее>
Закрыть
Справка

Рис. 15.2. Свойство «Тип значения характеристик»

Кроме этого, может случиться так, что пользователю станет недостаточно тех типов данных, которые существуют в конкретной конфигурации.

Например, он захочет вести учет в разрезе цвета товаров, а справочник *Цвет* в конфигурации отсутствует.

В этом случае он сможет воспользоваться специальным вспомогательным справочником, который разработчик создаст заблаговременно и укажет в качестве свойства объекта конфигурации *План видов характеристик* – *Дополнительные значения характеристик* (рис. 15.3).

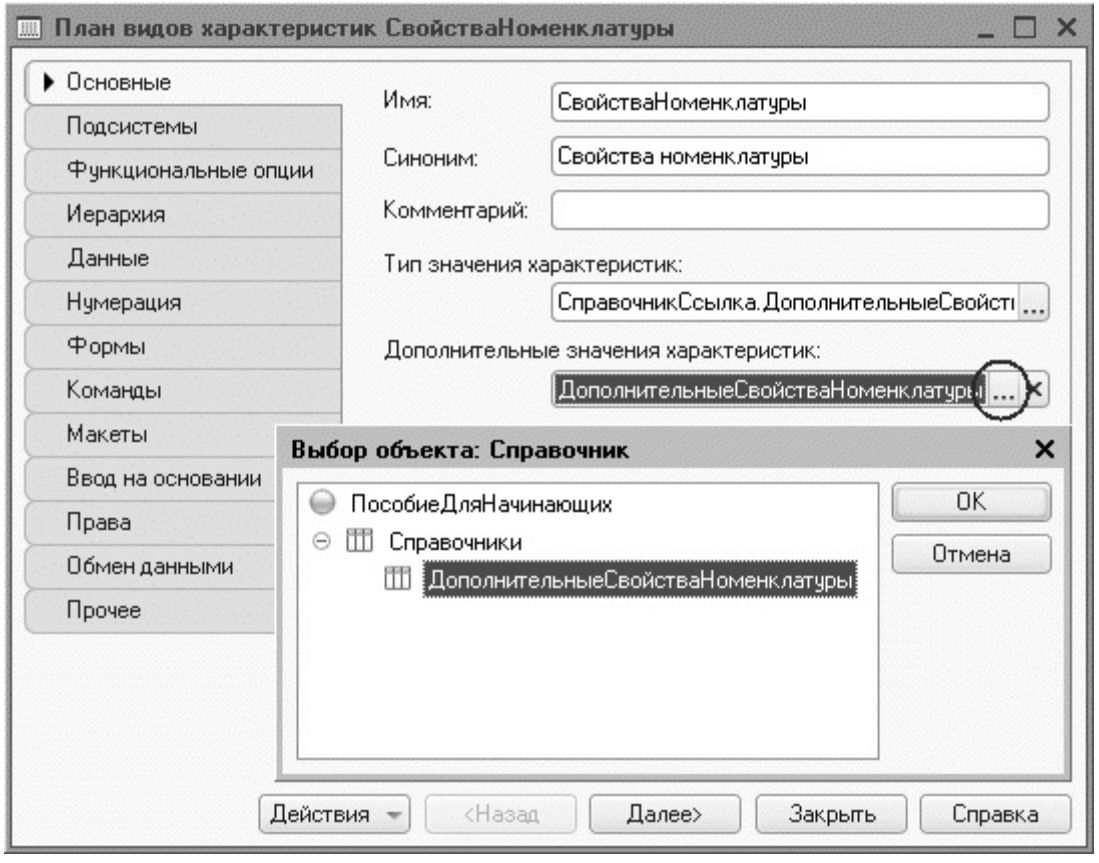


Рис. 15.3. Свойство «Дополнительные значения характеристик»

Тогда пользователь, создав новый вид характеристики *Цвет*, сможет задать необходимые значения цвета в справочнике дополнительных значений характеристик.

Примечательно, что этот справочник является подчиненным плану видов характеристик. Таким образом, если затем пользователь пожелает создать новый вид характеристик *Запах* и его значения, он будет создавать их в том же самом справочнике дополнительных характеристик, и они не будут «смешиваться» со значениями цвета.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с планами видов характеристик, можно прочитать в разделе «Краткий справочник разработчика. Планы видов характеристик».

План видов характеристик не имеет внутренних предопределенных механизмов привязки вида характеристики к тому объекту, который он должен описывать. Он лишь предоставляет возможность разработчику и пользователю описать некий набор характеристик и задать их тип.

Каким образом хранить соответствие конкретного вида характеристик или значения характеристик конкретному объекту базы данных, решает сам разработчик в зависимости от создаваемого прикладного решения.

С точки зрения реализации пример, который мы будем рассматривать далее, не является простым.

Поэтому сначала мы объясним логическую связь между объектами, которые будут использоваться в этом примере.

Логическая связь объектов

Для реализации этого примера нам понадобятся три новых объекта конфигурации.

Прежде всего, это *План видов характеристик*. Он будет хранить виды характеристик, которыми в принципе можно описывать материалы.

Кроме этого, нам понадобится специальный справочник, подчиненный справочнику *Номенклатура*. Элементы этого справочника будут идентифицировать партии материалов с некоторым фиксированным набором значений характеристик.

И третий объект – это регистр сведений, в котором собственно и будет храниться соответствие конкретных значений характеристик некоторому варианту материала (см. рис. 15.4).

Рис. 15.4. Логическая связь объектов

В результате использования такой логической структуры объектов мы получим возможность описывать каждую поступающую партию материала любым количеством видов характеристик, поскольку это соответствие будет храниться в регистре сведений.

И вместе с этим мы получим возможность вести учет в разрезе видов характеристик, добавив в регистры накопления еще одно измерение для хранения ссылки на элемент справочника, подчиненного справочнику *Номенклатура* (рис. 15.4).

В результате для того, чтобы узнать остатки материалов, обладающих некоторым значением характеристики, достаточно будет выбрать из регистра сведений все элементы подчиненного справочника с этим значением характеристики и затем по ним и их владельцам получить остатки регистра накопления.

Создание новых объектов конфигурации

В режиме «Конфигуратор»

Как мы уже говорили, нам понадобится создать несколько новых объектов конфигурации:

- справочник *ВариантыНоменклатуры,* чтобы описывать партии материалов;
- справочник *ДополнительныеСвойстваНоменклатуры,* чтобы задавать значения видов характеристик, для которых нет подходящих типов в конфигурации;
- план видов характеристик *СвойстваНоменклатуры,* чтобы создавать виды характеристик;
- регистр сведений *ЗначенияСвойствНоменклатуры,* чтобы хранить значения видов характеристик для различных партий материалов.

Сначала создадим объект конфигурации *Справочник* с именем *ВариантыНоменклатуры* и укажем, что он будет подчинен справочнику *Номенклатура*. Для этого на закладке *Владельцы* добавим справочник *Номенклатура* в список владельцев справочника *ВариантыНоменклатуры*.

Затем создадим еще один объект конфигурации *Справочник* с именем *ДополнительныеСвойстваНоменклатуры*.

После этого создадим объект конфигурации *План видов характеристик* с именем *СвойстваНоменклатуры*.

Установим его свойство *Тип значения характеристик*.

Для этого нажмем кнопку выбора ... и зададим составной тип данных следующим образом (рис. 15.5):

- *Число*, длина *15*, точность *3*;
- *Строка*, длина *25*;
- *Дата*;
- *Булево*;
- *СправочникСсылка.ДополнительныеСвойстваНоменклатуры.*

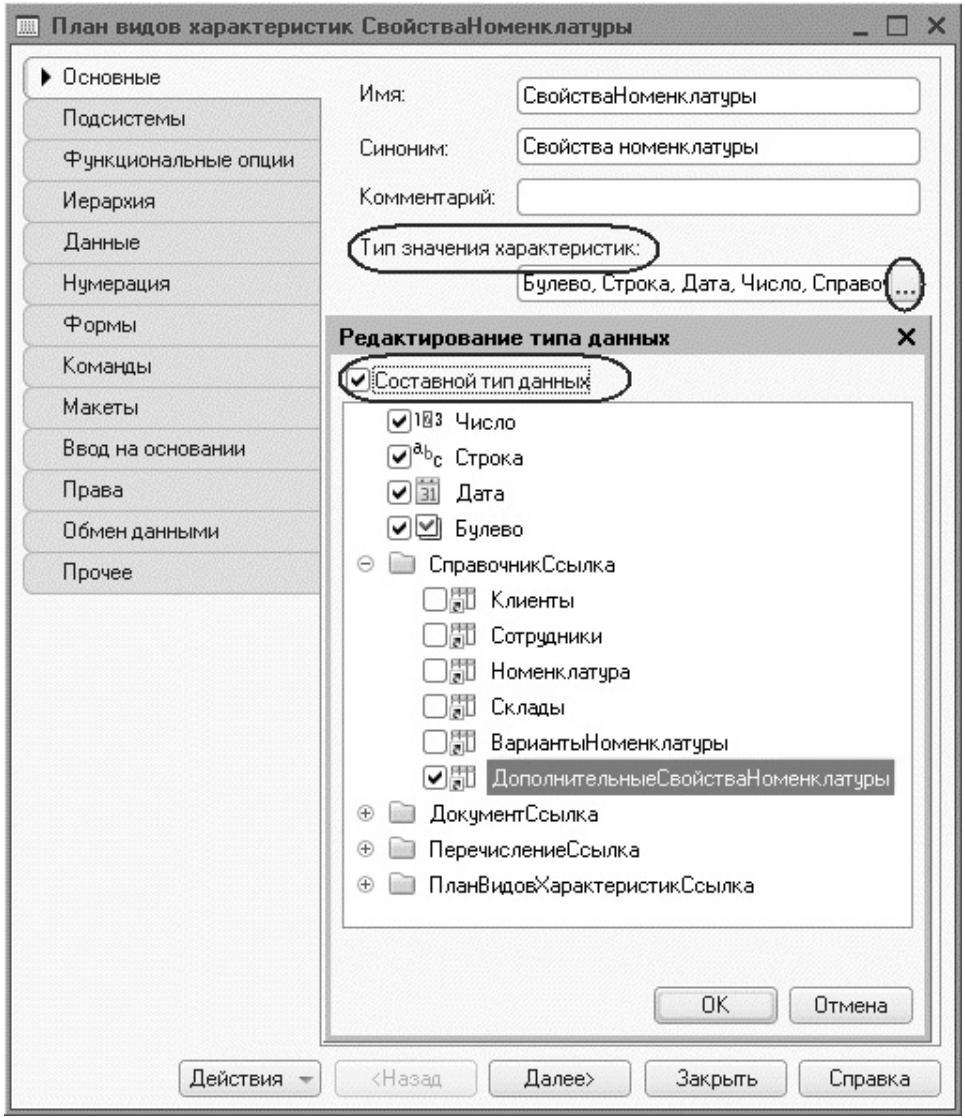

План видов характеристик СвойстваНоменклатуры
Основные
Подсистемы
Функциональные опции
Иерархия
Данные
Нумерация
Формы
Команды
Макеты
Ввод на основании
Права
Обмен данными
Прочее
Имя:
СвойстваНоменклатуры
Синоним:
Свойства номенклатуры
Комментарий:
Тип значения характеристик:
Булево, Строка, Дата, Число, Справо
...
Редактирование типа данных
Составной тип данных
Число
Строка
Дата
Булево
СправочникСсылка
Клиенты
Сотрудники
Номенклатура
Склады
ВариантыНоменклатуры
ДополнительныеСвойстваНоменклатуры
ДокументСсылка
ПеречислениеСсылка
ПланВидовХарактеристикСсылка
OK
Отмена
Действия
<Назад
Далее>
Закрыть
Справка

Рис. 15.5. Определение составного типа данных для типа значения характеристик плана видов характеристик

Затем справочнику *ДополнительныеСвойстваНоменклатуры* укажем владельца – план видов характеристик *СвойстваНоменклатуры* (рис. 15.6).

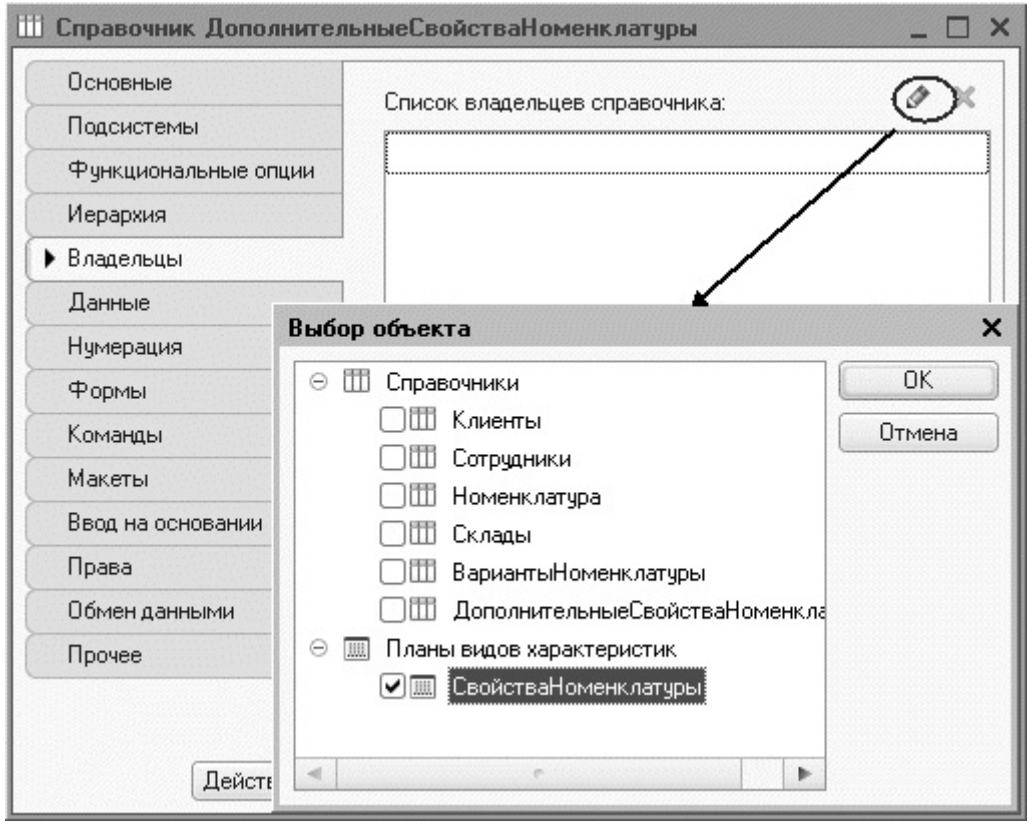


Рис. 15.6. Установка владельца справочника

После этого определим, что дополнительные значения характеристик плана видов характеристик будут располагаться в справочнике *ДополнительныеСвойстваНоменклатуры* (рис. 15.7).

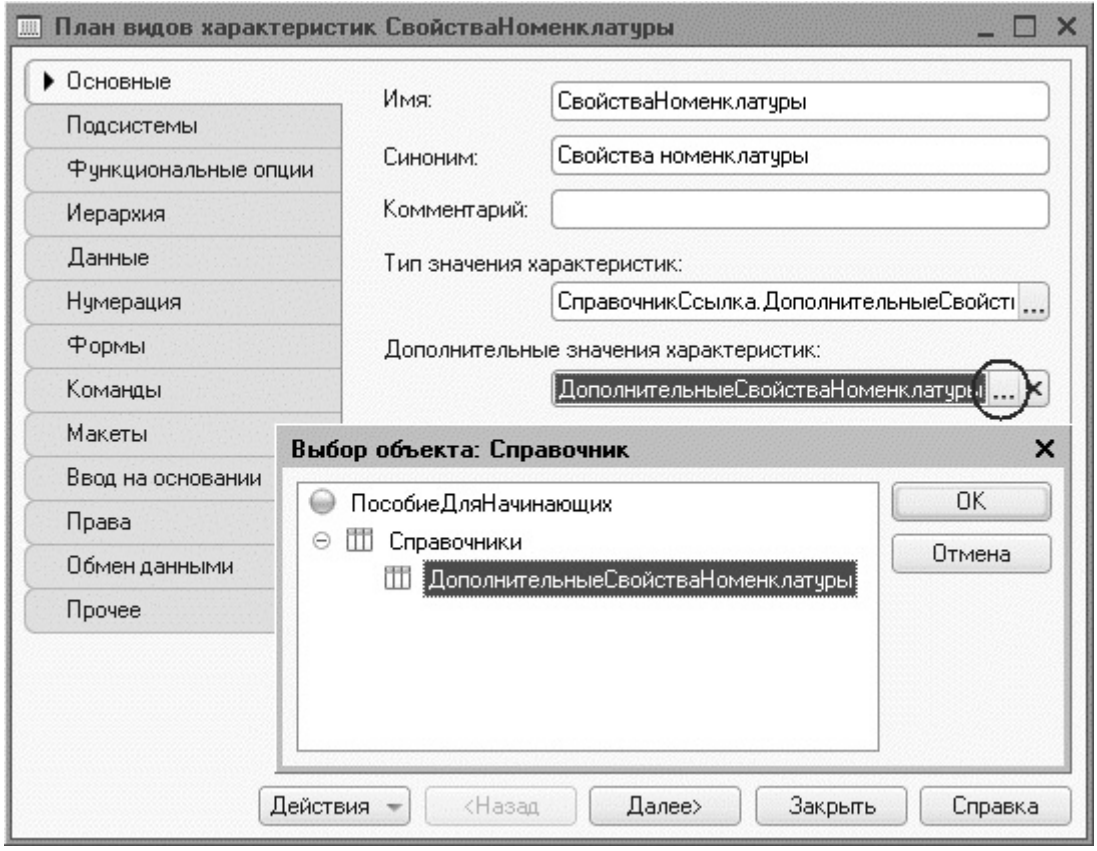


Рис. 15.7. Окно редактирования плана видов характеристик

Теперь создадим объект конфигурации *Регистр сведений* с именем *ЗначенияСвойствНоменклатуры*.

На закладке *Данные* создадим измерения регистра:

- *НаборСвойств*, *Ведущее*, тип *СправочникСсылка.ВариантыНоменклатуры*;
- *ВидСвойства*, тип *ПланВидовХарактеристикСсылка.СвойстваНоменклатуры*.

Затем создадим ресурс регистра (рис. 15.8):

- *Значение*, тип *Характеристика.СвойстваНоменклатуры*.

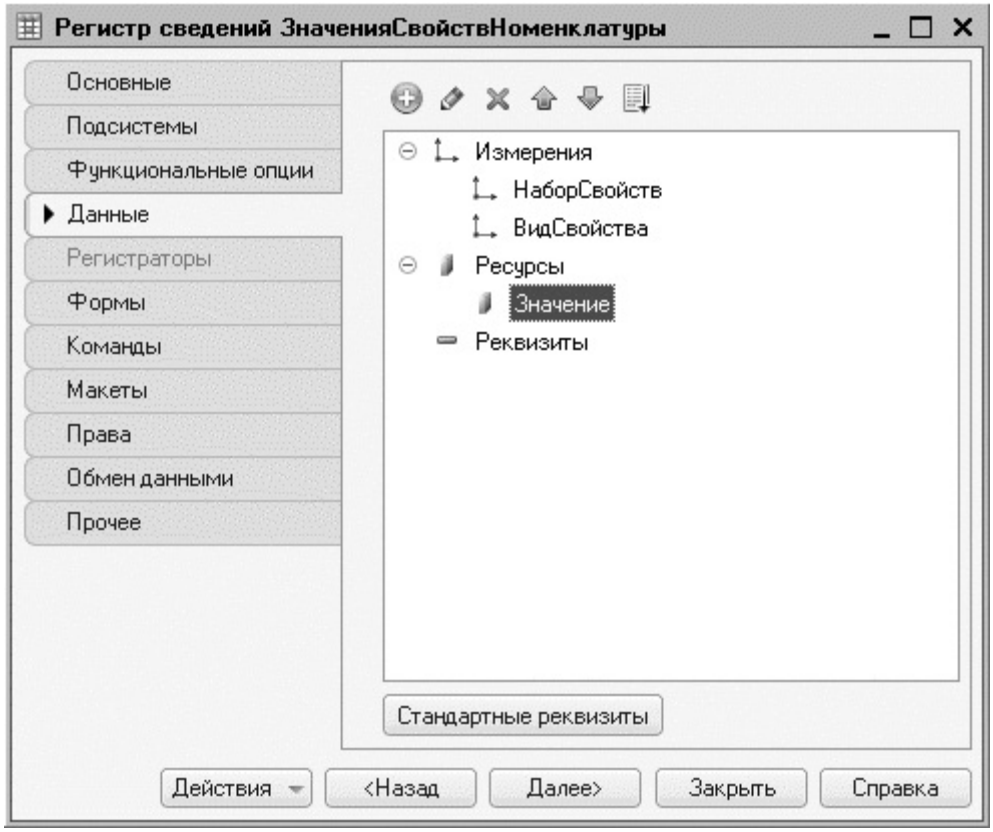


Рис. 15.8. Окно редактирования регистра сведений

Обратите внимание, что мы имеем возможность определить тип значения ресурса регистра как *Характеристика.<имя>*. По сути, это определение представляет собой составной тип данных, как он задан в типе значения соответствующего плана видов характеристик. То есть ресурс регистра может

иметь значение любого типа из тех, которые описаны в типе значения плана видов характеристик.

Кроме этого, зададим в свойстве *Связь по типу* этого ресурса измерение регистра *ВидСвойства*. Связь по типу будет обеспечивать нам соответствие типа значений, вводимых в это поле, и типа характеристики, выбранной в поле *Вид свойства*. А также заполним еще одно свойство – *Связи параметров выбора*.

Для этого нажмем кнопку выбора ... у этого свойства и перенесем из списка доступных реквизитов в список параметров измерение регистра *ВидСвойства*.

Установка свойства *Связи параметров выбора* обеспечит нам то, что при выборе значений, содержащихся в справочнике *Дополнительные свойства номенклатуры,* для выбора будут предлагаться только те значения, которые относятся к выбранной характеристике, а не все, которые есть в этом справочнике (рис. 15.9).

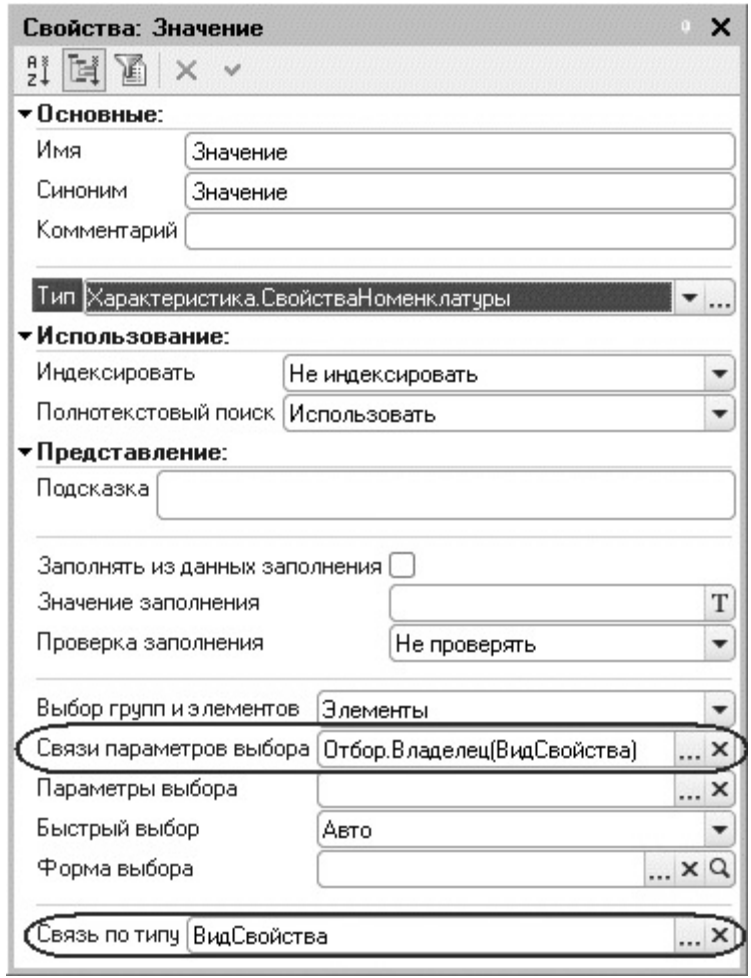


Рис. 15.9. Свойство ресурса «Значение регистра сведений»

В заключение для справочника *ВариантыНоменклатуры* опишем, где

хранятся свойства вариантов номенклатуры и как получить значения этих свойств. Это описание платформа будет использовать автоматически при выполнении отчетов и при формировании различных динамических списков, в которых задействуются варианты номенклатуры.

В контекстном меню справочника *ВариантыНоменклатуры* выберем команду *Характеристики* (рис. 15.9а).

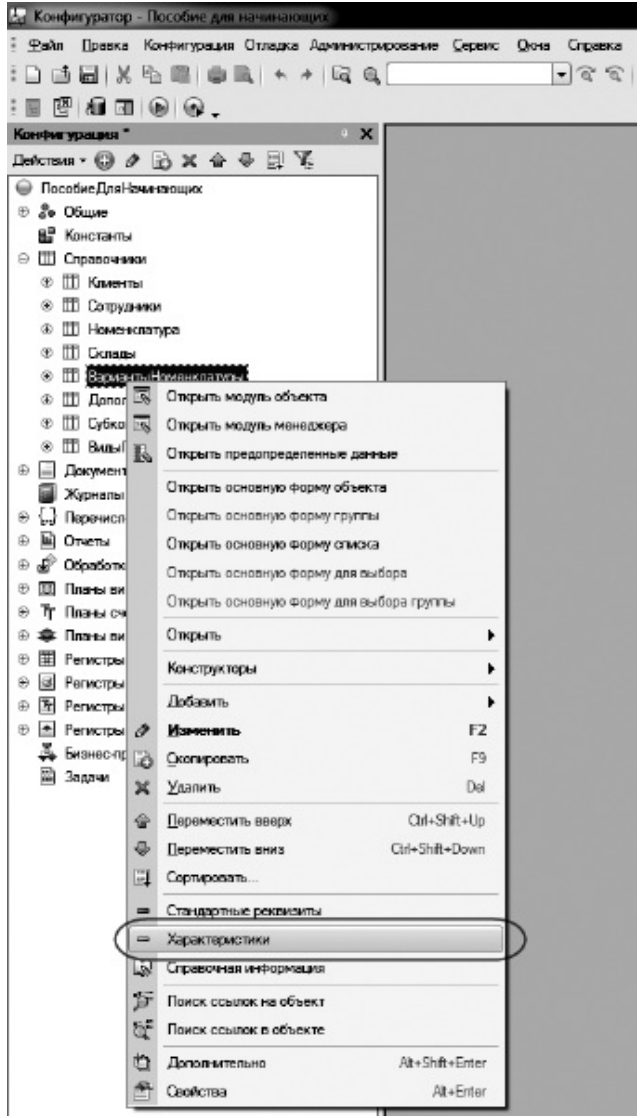

Конфигуратор - Пособие для начинающих
Файл Правка Конфигурация Отладка Администрирование Сервис Окна Справка
Конфигурация *
Действия
ПособиеДляНачинающих
Общие
Константы
Справочники
Клиенты
Сотрудники
Номенклатура
Склады
Открыть модуль объекта
Открыть модуль менеджера
Открыть предопределенные данные
Открыть основную форму объекта
Открыть основную форму группы
Открыть основную форму списка
Открыть основную форму для выбора
Открыть основную форму для выбора группы
Открыть
Конструкторы
Добавить
Изменить F2
Скопировать F9
Удалить Del
Переместить вверх Ctrl+Shift+Up
Переместить вниз Ctrl+Shift+Down
Сортировать...
Стандартные реквизиты
Характеристики
Справочная информация
Поиск ссылок на объект
Поиск ссылок в объекте
Дополнительно Alt+Shift+Enter
Свойства Alt+Enter

Рис. 15.9a. Переход к характеристикам справочника «ВариантыНоменклатуры»

Откроется диалог описания характеристик, сейчас он пуст. С помощью кнопки командной панели добавим в него новую запись (рис. 15.9b).

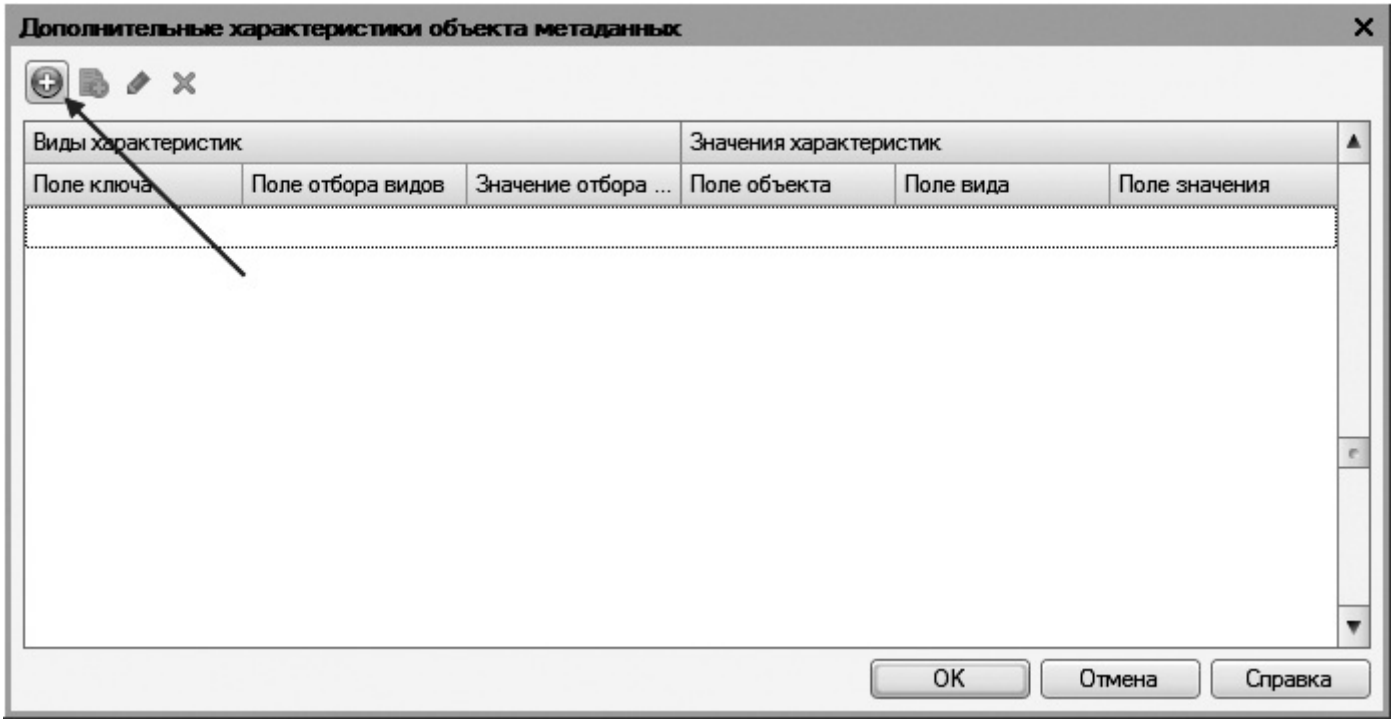


Рис. 15.9b. Переход к характеристикам справочника «ВариантыНоменклатуры»

В качестве источника характеристик выберем план видов характеристик *СвойстваНоменклатуры*. Платформа автоматически определит, что полем

ключа будет являться поле *Ссылка* этого объекта конфигурации (рис. 15.9с).

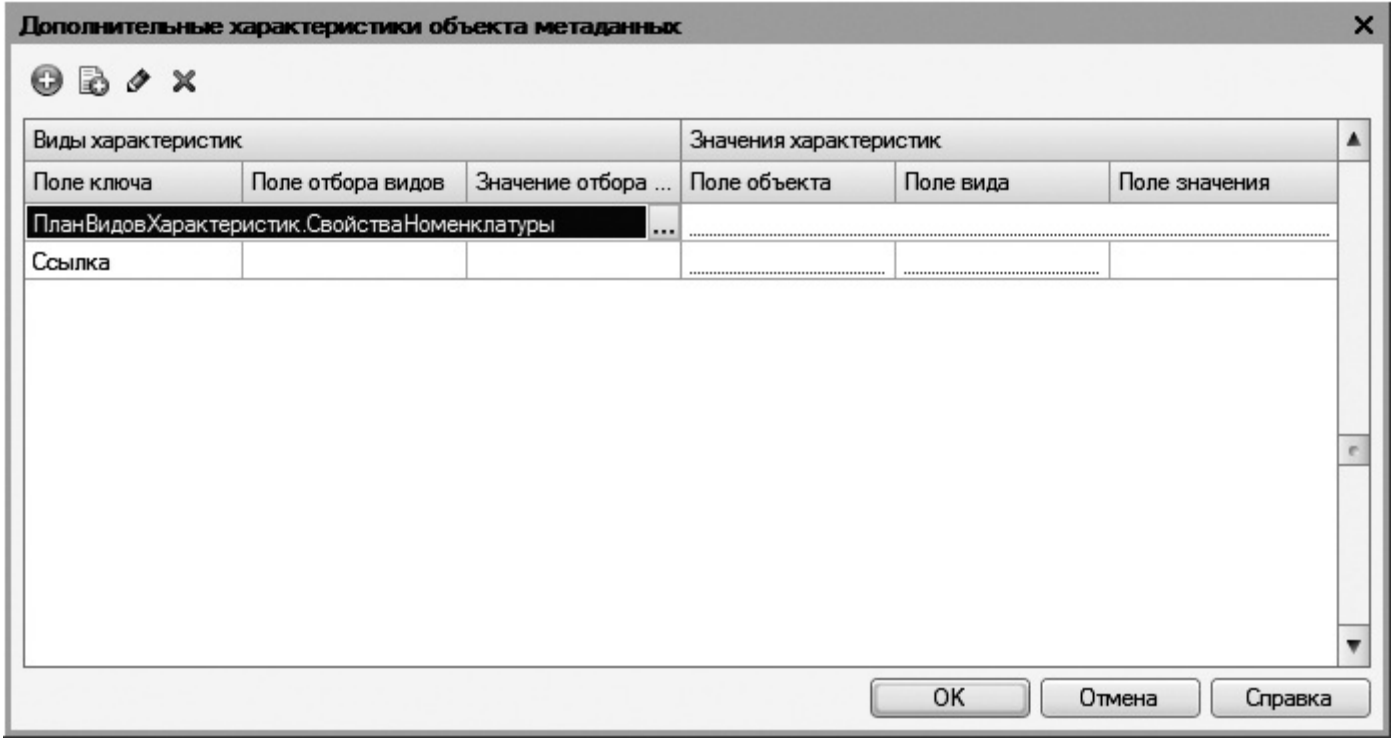


Рис. 15.9с. Описание источника видов характеристик

Два оставшихся поля, *Поле отбора видов* и *Значение отбора*, оставим пустыми. В нашем случае эти поля не понадобятся.

Перейдем к описанию того, где и как хранятся значения наших свойств. В

качестве источника значений характеристик выберем регистр сведений *ЗначенияСвойствНоменклатуры*. Платформа автоматически определит, что в этом регистре полем объекта является измерение *НаборСвойств*, а полем вида – измерение *ВидСвойства*.

Поэтому единственное, что нам останется указать самостоятельно, что значения свойств хранятся в ресурсе *Значение*. В результате описание характеристик для справочника *ВариантыНоменклатуры* будет выглядеть следующим образом (рис. 15.9d).

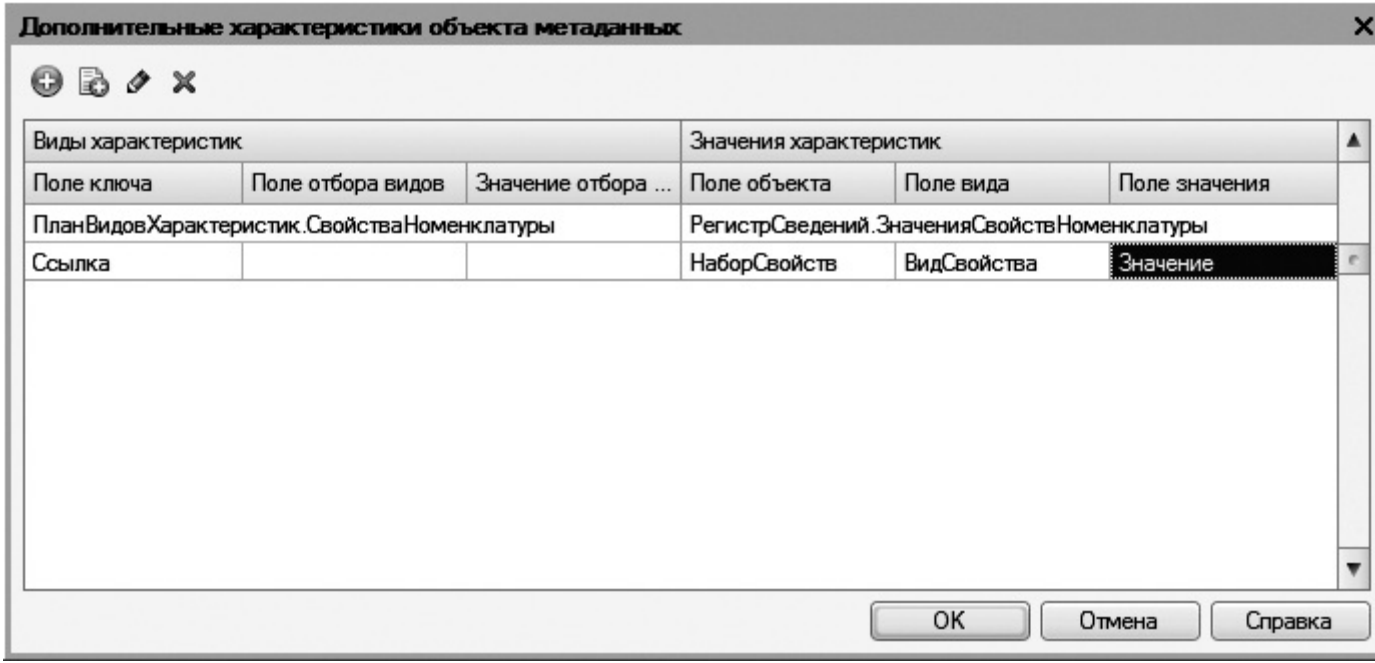


Рис. 15.9d. Описание характеристик для справочника «ВариантыНоменклатуры»

Доработка объектов конфигурации

Итак, мы создали новые объекты конфигурации и задали их основные свойства, необходимые для реализации нашей задачи.

Но, как мы дальше увидим, не все свойства нас полностью устраивают. И вообще, при разработке невозможно предусмотреть все заранее. Часто какие-

то недочеты становятся видны лишь в процессе работы. То есть, увидев промежуточный результат в режиме *1С:Предприятие*, важно уметь оценить недостатки и исправить их прямо по ходу работы.

Поэтому на этом занятии мы продемонстрируем процесс разработки от обратного. Это тоже очень ценный опыт, который, мы надеемся, будет полезен читателю.

Справочник «Номенклатура»

В режиме «1С:Предприятие»

Итак, запустим «1С:Предприятие» в режиме отладки и посмотрим, как взаимодействуют логически связанные объекты конфигурации *Справочник Номенклатура*, *Справочник ВариантыНоменклатуры*, *План видов характеристик СвойстваНоменклатуры* и *Регистр сведений ЗначенияСвойствНоменклатуры*.

Обратите внимание, что мы не указывали для этих объектов подсистем, к которым они относятся. Дело в том, что отображение этих объектов вне их логической связи друг с другом не имеет особого смысла. Поскольку мы задали владельцев справочников, ведущее измерение регистра сведений и т. п., то нужные объекты автоматически попадут в панель навигации форм своих владельцев как подчиненная информация.

Поэтому проигнорируем появившееся системное сообщение об отсутствии привязки созданных нами объектов к подсистемам.

Итак, по условию нашей задачи мы хотим создать наборы свойств и составляющие их характеристики для отдельных элементов номенклатуры. Наборы свойств, как мы уже говорили, будут храниться в справочнике *ВариантыНоменклатуры*, подчиненном справочнику *Номенклатура*.

В разделе *Учет материалов* откроем справочник *Номенклатура* и его элемент *Кабель электрический* из группы *Материалы > Прочее* (рис. 15.10).

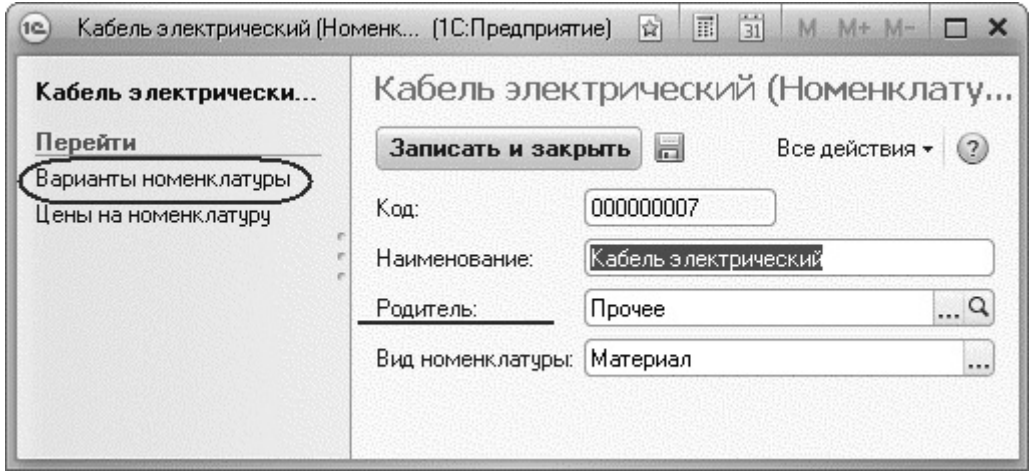


Рис. 15.10. Форма элемента справочника «Номенклатура»

Поскольку справочник *Номенклатура* является владельцем справочника *ВариантыНоменклатуры*, мы видим в панели навигации формы ссылку для перехода к подчиненному списку. Это значит, что при открытии этого списка мы будем видеть только наборы свойств, относящиеся к редактируемому элементу справочника *Номенклатура*.

Это система сделала за нас, и нас это устраивает. Но название стандартного реквизита *Родитель* не совсем понятно. Более понятно и естественно было бы назвать его *Группа номенклатуры*, так как родитель для элемента – это и есть группа номенклатуры, к которой он относится.

Поскольку в интерфейсе приложения отражаются синонимы объектов, нам нужно изменить синоним стандартного реквизита справочника, который по умолчанию совпадает с его именем *Родитель*.

В режиме «Конфигуратор»
Для этого вернемся в конфигуратор и откроем окно редактирования объекта конфигурации *Справочник Номенклатура*.

На закладке *Данные* нажмем кнопку *Стандартные реквизиты*, в списке этих реквизитов дважды щелкнем на реквизите *Родитель* и в открывшейся палитре свойств зададим *Синоним* реквизита *Группа номенклатуры* (рис. 15.11).

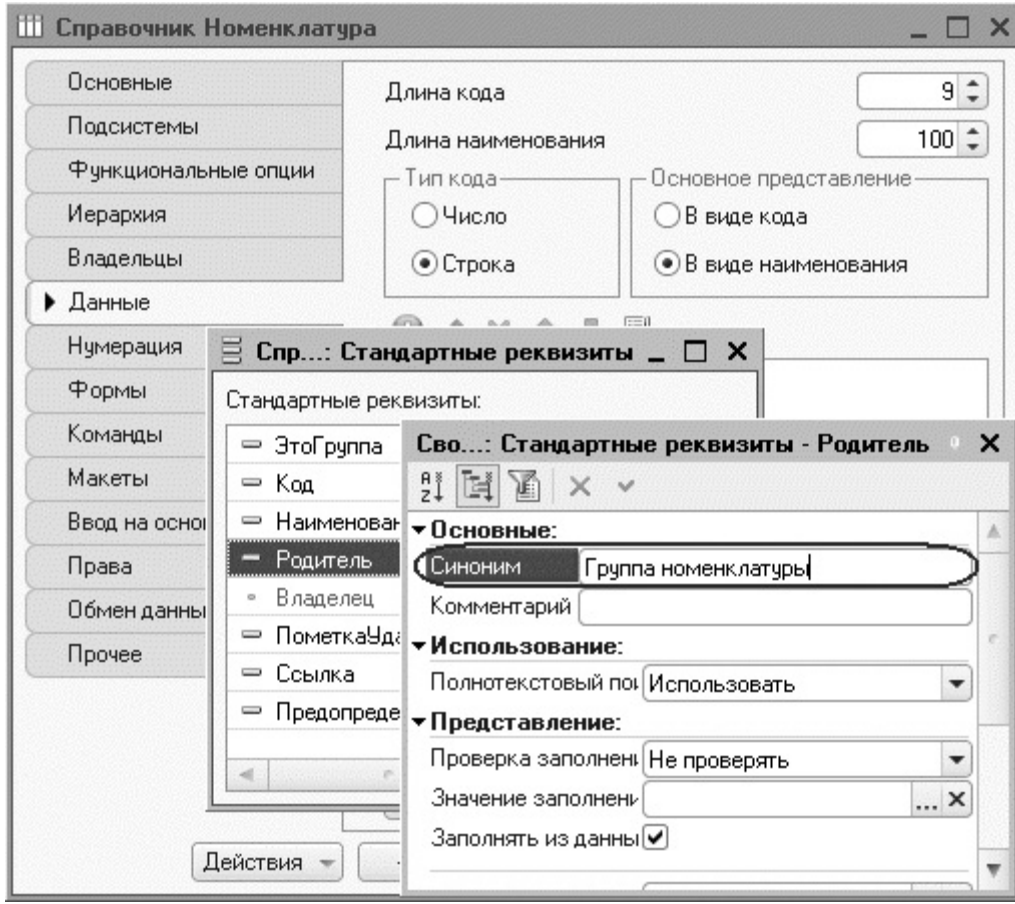


Рис. 15.11. Установка синонима стандартного реквизита

Обратите внимание, что мы изменили синоним реквизита объекта конфигурации, а не реквизита формы. В данном случае форма элемента справочника

Номенклатура вообще сгенерирована системой автоматически.

Теперь во всех видах форм данный реквизит будет иметь установленный синоним, если, конечно, разработчик не захочет его изменить при создании своей собственной формы.

В режиме «1С:Предприятие»

Проверим результат изменений в режиме *1С:Предприятие*.

Откроем форму того же элемента номенклатуры и вместо названия *Родитель* мы увидим *Группа номенклатуры* (рис. 15.12).

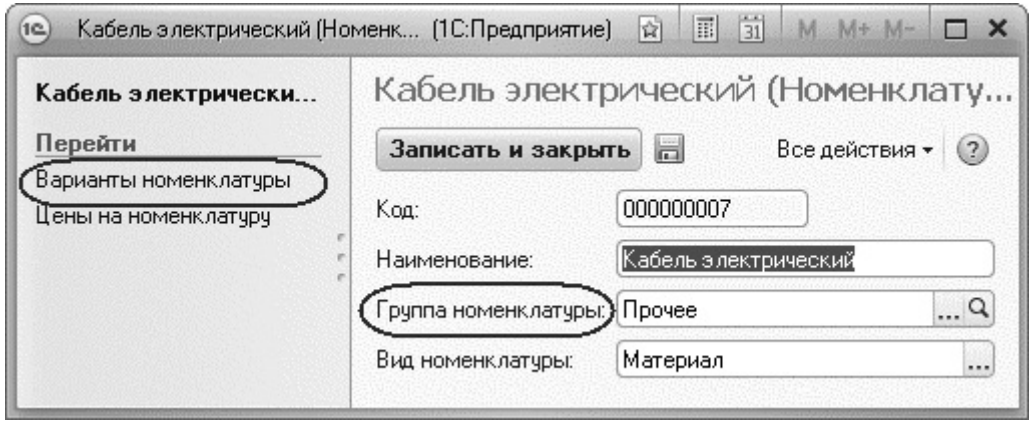


Рис. 15.12. Форма элемента справочника «Номенклатура»

Справочник «Варианты номенклатуры»

В режиме «1С:Предприятие»

Теперь мы хотим создать набор свойств для элемента номенклатуры *Кабель электрический*.

Для этого выполним команду *Варианты номенклатуры* для перехода к списку, где будут храниться наборы свойств элементов номенклатуры (рис. 15.13).

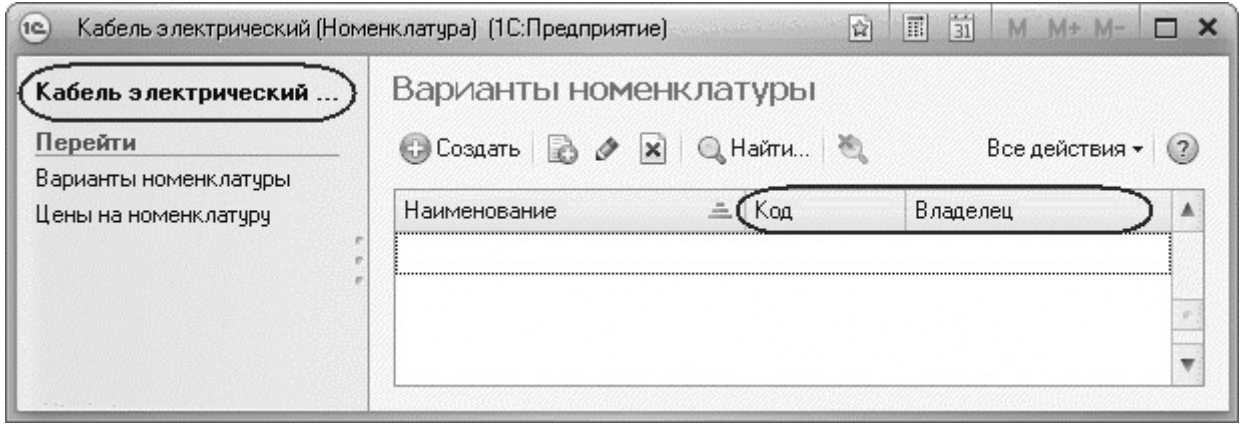


Рис. 15.13. Список вариантов номенклатуры

Открывшаяся форма списка вариантов номенклатуры не совсем нас устраивает – столбцы *Код* и *Владелец* явно лишние.

Код нового варианта номенклатуры генерируется автоматически и ни о чем

пользователю не говорит.

Владелец варианта номенклатуры отражен в левом верхнем углу панели навигации формы и тоже в списке не имеет смысла.

Чтобы сделать эти колонки невидимыми, нам нужно создать форму списка справочника *ВариантыНоменклатуры* и при ее создании проанализировать, откуда она открывается (это можно понять по значению параметра формы *Отбор*).

Если установлен отбор по владельцу (то есть она открывается из списка номенклатуры), то мы будем в ней скрывать колонки *Код* и *Владелец*.

Если же форма открывается другими способами, то эти колонки могут понадобиться, поэтому просто удалить их из формы было бы неправильно.

Поскольку форма создается на сервере, делать это нужно в обработчике события формы *ПриСозданииНаСервере*.

В режиме «Конфигуратор»
Вернемся в конфигуратор и устраним недостатки формы списка.

Для создания формы откроем окно редактирования объекта конфигурации

Справочник ВариантыНоменклатуры, перейдем на закладку *Формы*, нажмем кнопку открытия и создадим основную форму списка (рис. 15.14).

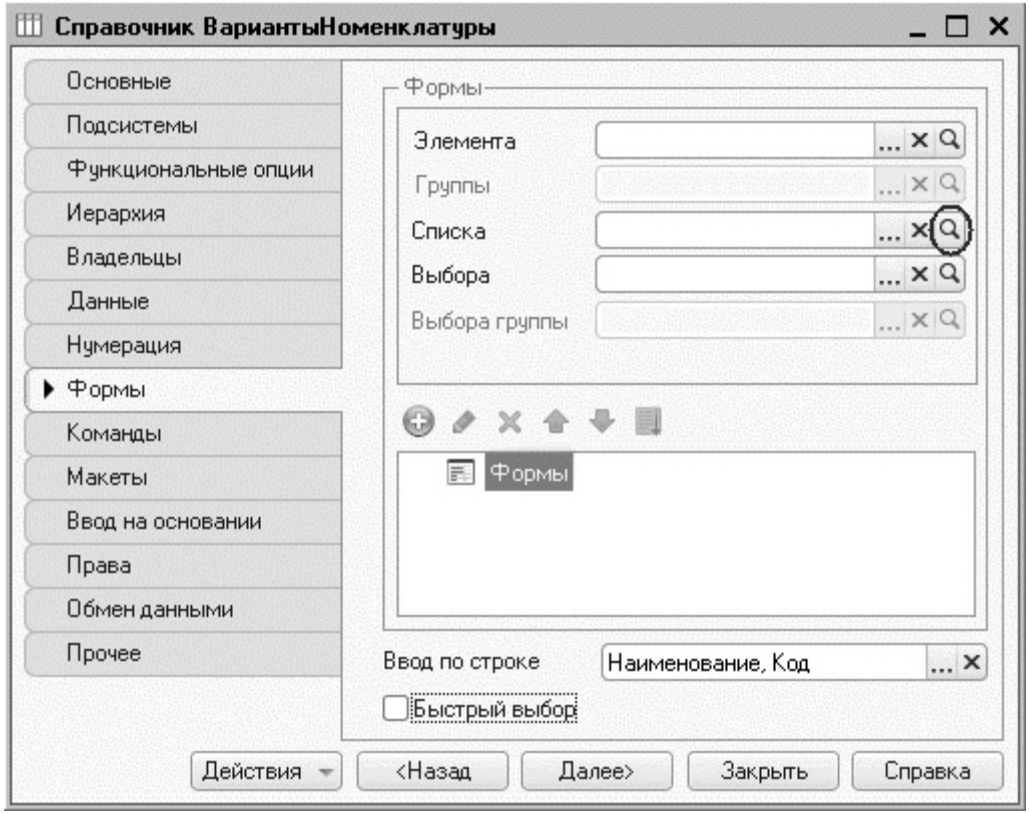


Рис. 15.14. Создание основной формы списка

В открывшемся окне конструктора нажмем *Готово*.

Форма, созданная конструктором, в отличие от автогенерируемой формы, не содержит поля *Владелец*. Поэтому наша задача даже упрощается: нам нужно будет скрыть только одно поле – *Код*.

В открывшемся окне редактора форм вверху слева расположено окно элементов формы. Выделим в нем элемент *Форма* (поскольку нам нужно событие формы в целом) и двойным щелчком мыши откроем палитру свойств этого элемента.

Прокрутив вниз список свойств формы, найдем событие *ПриСозданииНаСервере* и нажмем кнопку открытия  (рис. 15.15).

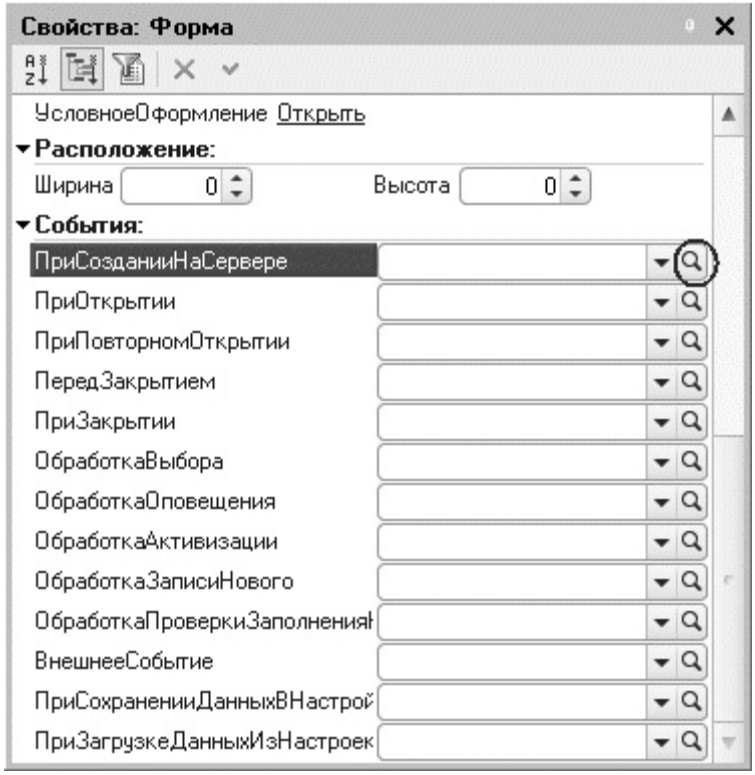


Рис. 15.15. Создание обработчика события формы «При создании на сервере»

В модуле формы будет создан обработчик события формы *ПриСозданииНаСервере*, в который мы внесем следующий текст (листинг 15.1).

Листинг 15.1. Обработчик события формы «ПриСозданииНаСервере()»

```
Если Параметры.Отбор.Свойство("Владелец") Тогда
```

```
    Элементы.Код.Видимость = Ложь;

КонецЕсли;
```

Прокомментируем этот код.

Параметры – это свойство управляемой формы, в модуле которой мы находимся. Используя это свойство, мы получаем объект, который содержит коллекцию параметров формы.

К элементу этой коллекции *Отбор* мы обращаемся по имени. Используя метод *Свойство* структуры элементов отбора, мы определяем, установлен ли отбор по полю *Владелец*.

Если такой отбор установлен, то мы устанавливаем видимость поля *Код* в значение *Ложь*, то есть скрываем это поле. Здесь *Элементы* – это свойство управляемой формы, которое позволяет получить доступ ко всем элементам формы.

В режиме «1С:Предприятие»
Проверим результат изменений в режиме *1С:Предприятие*.

Форма списка вариантов номенклатуры будет иметь следующий вид (рис.

15.16).

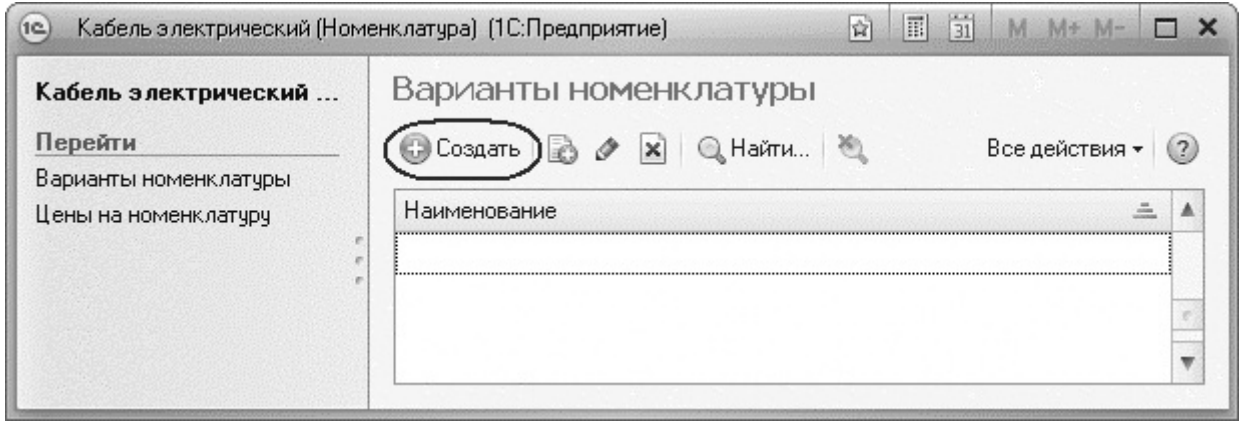


Рис. 15.16. Список вариантов номенклатуры

Мы видим, что добились желаемого результата (см. рис. 15.13): было три колонки, а теперь только одна – *Наименование*.

Теперь нажмем кнопку *Создать*, чтобы создать новый набор свойств для элемента номенклатуры.

Откроется форма элемента справочника *ВариантыНоменклатуры* (рис. 15.17).

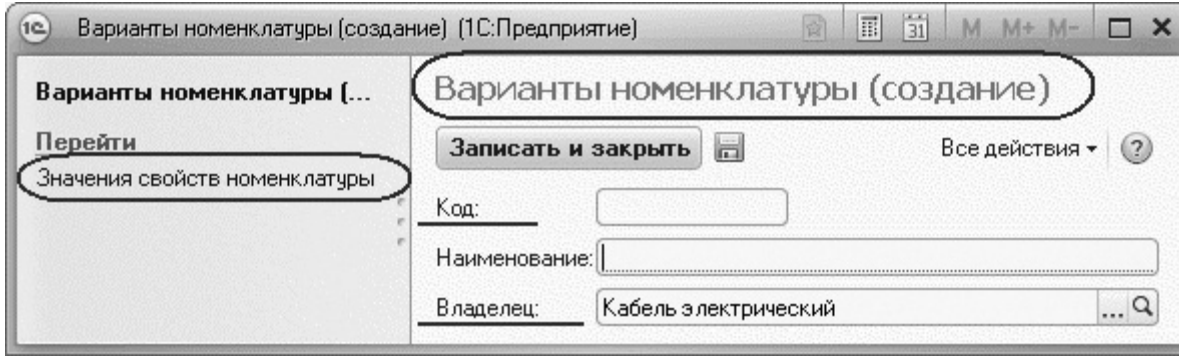


Рис. 15.17. Форма элемента справочника «Варианты номенклатуры»

Эта форма сгенерирована системой автоматически. Но в ней также есть недостатки:

- заголовок формы должен быть задан в единственном числе;
- лишние поля *Код* и *Владелец;*
- команду перехода к подчиненной информации нужно переименовать в более понятную.

Вернемся в конфигуратор и исправим их.

В режиме «Конфигуратор»

Во-первых, нужно переименовать заголовок формы, чтобы было понятно, что мы создаем в данный момент один вариант номенклатуры.

Для этого в окне редактирования объекта конфигурации *Справочник ВариантыНоменклатуры* на закладке *Основные* зададим *Представление объекта* в единственном числе как *Вариант номенклатуры* (рис. 15.18).

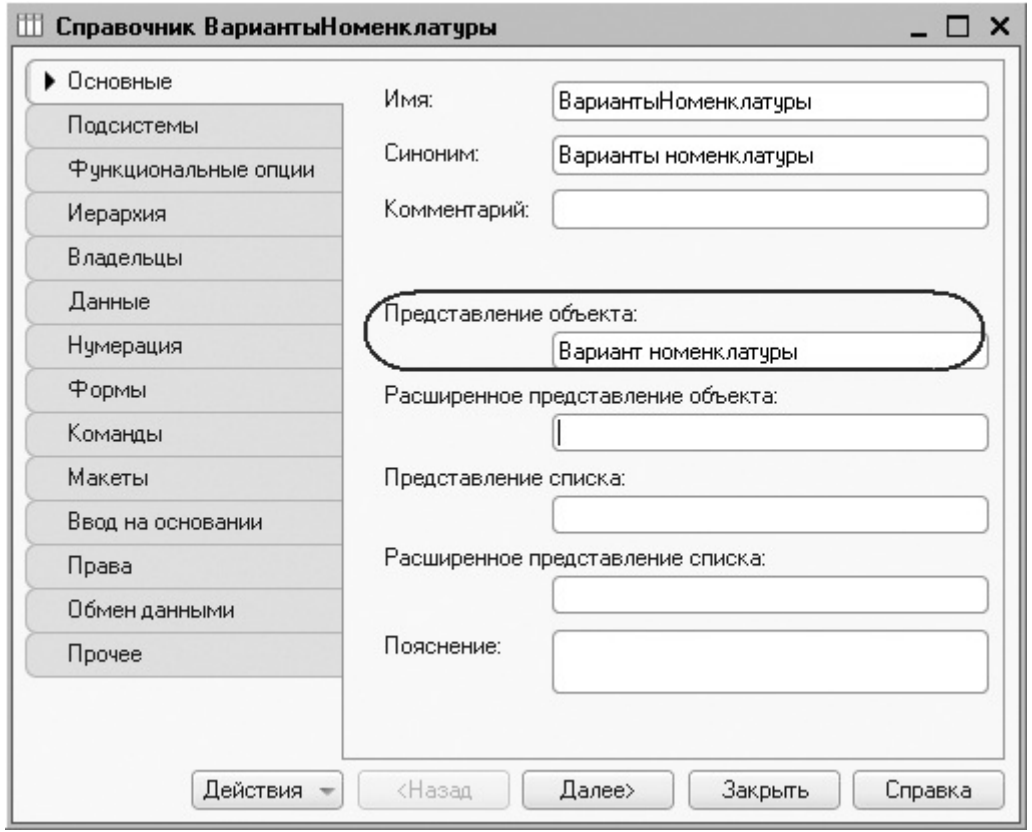


Рис. 15.18. Установка представления объекта

Это свойство будет использоваться в интерфейсе «1С:Предприятия» как заголовок формы элемента справочника.

Во-вторых, нужно убрать поля *Код* и *Владелец* из этой формы.

Для этого в окне редактирования объекта конфигурации *Справочник ВариантыНоменклатуры* перейдем на закладку *Формы*, нажмем кнопку открытия  и создадим основную форму элемента.

В окне структуры элементов формы выделим поочередно эти элементы и, нажимая кнопку *Удалить* в командной панели, удалим их из формы (рис. 15.19).

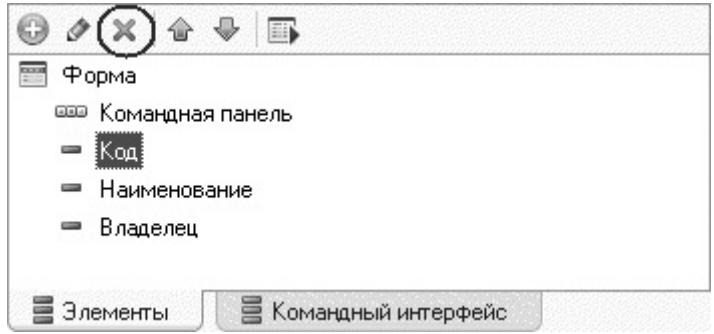


Рис. 15.19. Удаление элементов формы

В результате в форме элемента будет отображен только один реквизит

справочника – *Наименование*.

Его представление мы тоже немного поправим.

На закладке *Данные* в окне редактирования объекта конфигурации Справочник *ВариантыНоменклатуры* нажмем кнопку *Стандартные реквизиты*, в списке этих реквизитов дважды щелкнем на реквизите *Наименование* и в открывшейся палитре свойств зададим *Синоним* реквизита *Название*.

В-третьих, не вяжутся друг с другом заголовок формы *Вариант номенклатуры* и подчиненная ему информация – *Значения свойств номенклатуры* (см. рис. 15.17). Это записи одноименного регистра, к которым можно перейти из формы элемента.

Поэтому в окне редактирования объекта конфигурации *Регистр сведений ЗначенияСвойствНоменклатуры* на закладке *Основные* зададим *Представление списка* как *Состав варианта номенклатуры* (рис. 15.20).

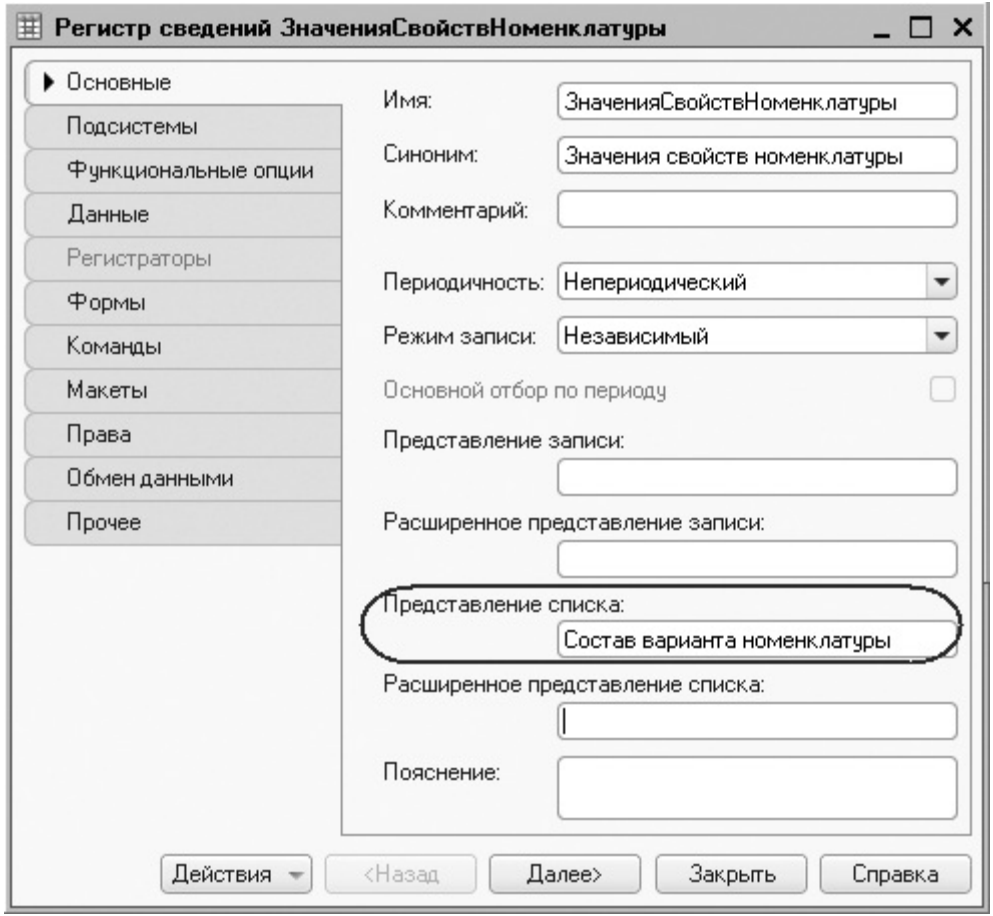


Рис. 15.20. Установка представления списка регистра

Это свойство будет использоваться в интерфейсе «1С:Предприятия» как заголовок формы списка регистра.

В режиме «1С:Предприятие»
Проверим результат изменений в режиме *1С:Предприятие*.

Итак, в разделе *Учет материалов* откроем справочник *Номенклатура* и его элемент *Кабель электрический* из группы *Материалы > Прочее*.

В форме элемента выполним команду *Варианты номенклатуры* для перехода к списку наборов свойств данного элемента номенклатуры. Пока этот список пуст.

Нажмем кнопку *Создать*. Теперь в открывшейся форме варианта номенклатуры нас все устраивает.

Регистр «Значения свойств номенклатуры»

В режиме «1С:Предприятие»
Создадим вариант номенклатуры *Белые кабели* (рис. 15.21).

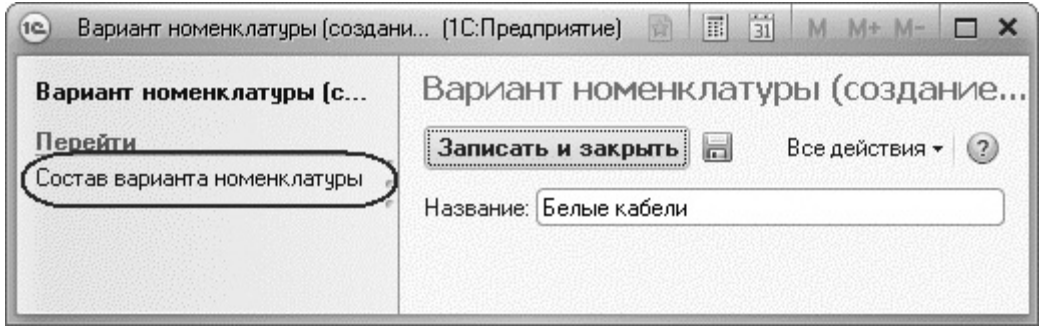


Рис. 15.21. Форма элемента справочника «Вариант номенклатуры»

Выполним команду *Состав варианта номенклатуры* для перехода к составу редактируемого варианта номенклатуры. Если вариант номенклатуры еще не записан, то появится вопрос о записи данных, на который мы ответим утвердительно (рис. 15.22).

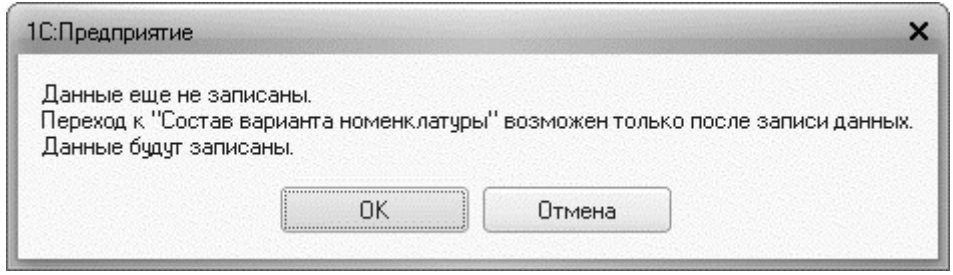


Рис. 15.22. Подтверждение записи данных

После этого откроется форма списка регистра *Значения свойств номенклатуры*, которая также генерируется по умолчанию (рис. 15.23).

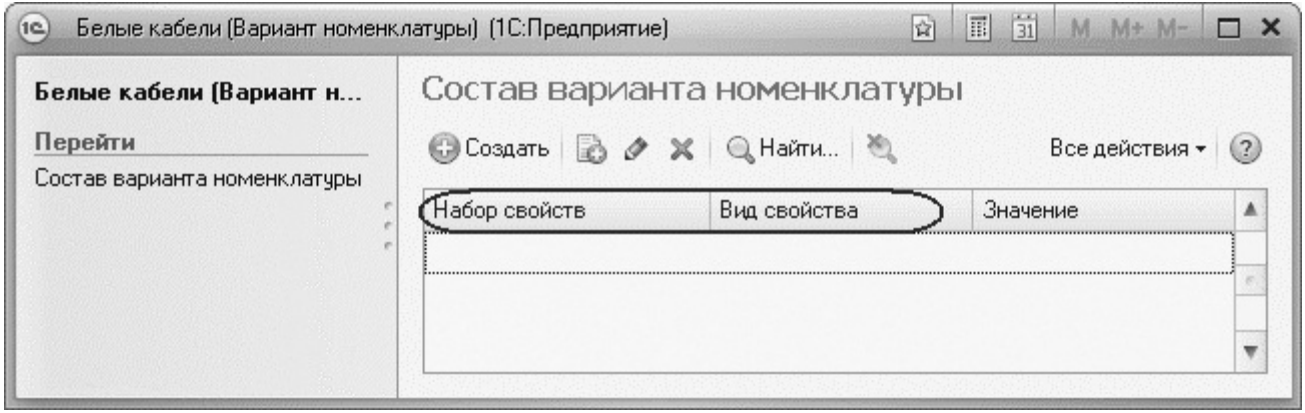


Рис. 15.23. Форма списка регистра «Состав варианта номенклатуры»

В этой форме нас также не все устраивает:

- заголовок колонки *ВидСвойства* лучше переименовать;
- лишняя колонка *НаборСвойств.*

Вернемся в конфигуратор и устраним недостатки формы списка.

В режиме «Конфигуратор»
Во-первых, название колонки *Вид свойства* лучше переименовать в *Свойство*.

Для этого в окне редактирования объекта конфигурации Регистр сведений *ЗначенияСвойствНоменклатуры* на закладке *Данные* откроем палитру

свойств измерения *ВидСвойства* и зададим его *Синоним* как *Свойство* (рис. 15.24).

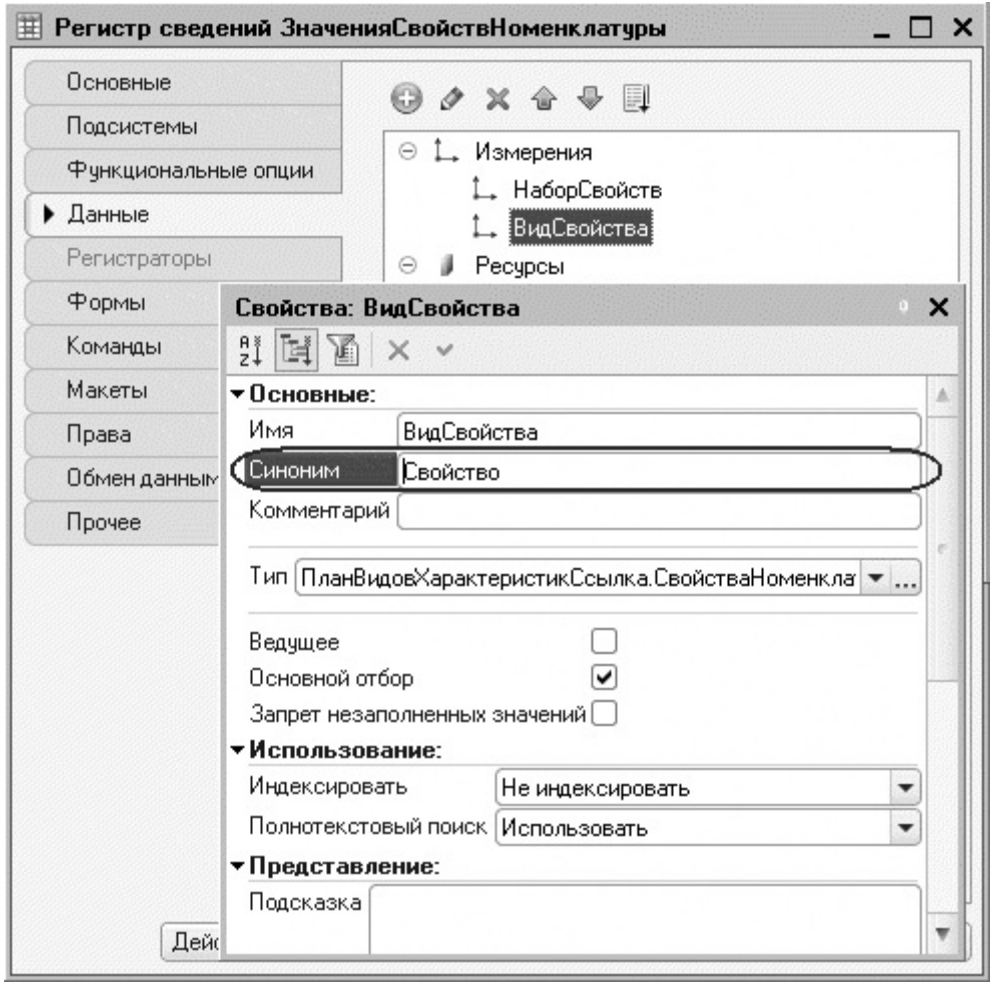


Рис. 15.24. Установка синонима для измерения регистра

Во-вторых, поскольку регистр имеет ведущее измерение *НаборСвойств* типа

СправочникСсылка.ВариантыНоменклатуры, поле *Набор свойств* – лишнее, так как владелец данного набора свойств отражен в левом верхнем углу панели навигации формы.

Поэтому создадим обработчик события *ПриСозданииНаСервере* формы списка регистра и в нем сделаем колонку *НаборСвойств* невидимой в случае открытия формы с отбором по этому полю, то есть если форма списка регистра открыта из формы элемента справочника *Варианты номенклатуры*.

Для создания этого обработчика откроем окно редактирования объекта конфигурации *Регистр сведений ЗначенияСвойствНоменклатуры,* перейдем на закладку *Формы*, нажмем кнопку открытия  и создадим основную форму списка.

Затем создадим для формы обработчик события формы *ПриСозданииНаСервере*, в который мы внесем следующий текст (листинг 15.2).

Листинг 15.2. Обработчик события формы «ПриСозданииНаСервере()»

```
Если Параметры.Отбор.Свойство("НаборСвойств") Тогда
  Элементы.НаборСвойств.Видимость = Ложь;

КонецЕсли;
```

Этот код аналогичен коду, приведенному выше в листинге 15.1, поэтому в комментариях не нуждается.

В режиме «1С:Предприятие»

Проверим результат изменений в режиме *1С:Предприятие*.

В результате форма списка регистра *Состав варианта номенклатуры* примет вид (рис. 15.25).

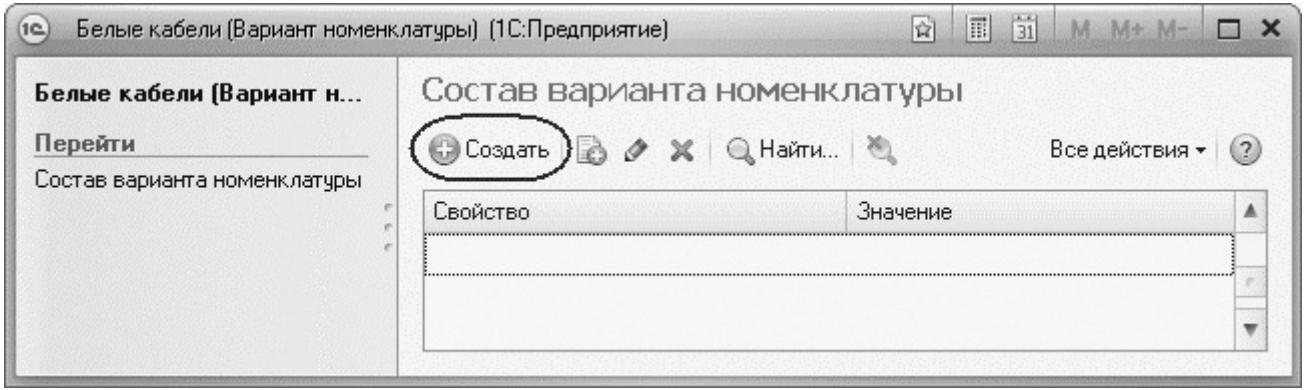


Рис. 15.25. Форма списка регистра «Состав варианта номенклатуры»

Теперь, если нажать кнопку *Создать*, чтобы ввести новую запись в состав варианта номенклатуры, откроется форма записи регистра *ЗначенияСвойствНоменклатуры* (рис. 15.26).

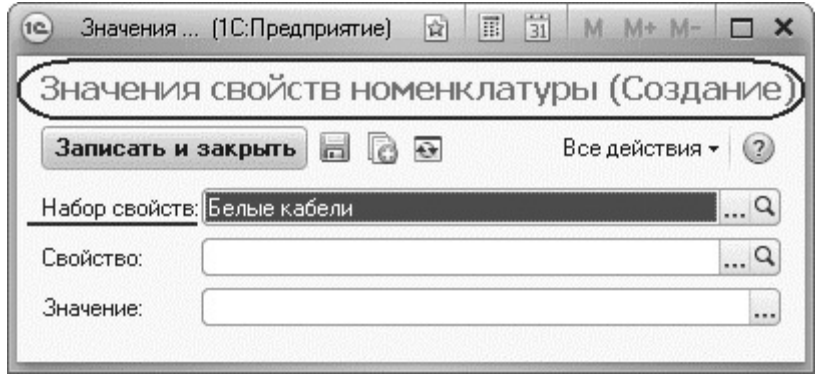


Рис. 15.26. Форма записи регистра «Значения свойств номенклатуры»

Эта форма сгенерирована системой автоматически. Но в ней также есть недостатки:

- заголовок формы должен быть задан в единственном числе;
- лишняя колонка *НаборСвойств.*

Вернемся в конфигуратор и исправим их.

В режиме «Конфигуратор»

Во-первых, нужно переименовать заголовок формы, чтобы было понятно, что мы создаем в данный момент одно свойство и его значение в составе варианта номенклатуры.

Для этого в окне редактирования объекта конфигурации *Регистр сведений ЗначенияСвойствНоменклатуры* на закладке *Основные* зададим *Представление записи* как *Свойство и значение* (рис. 15.27).

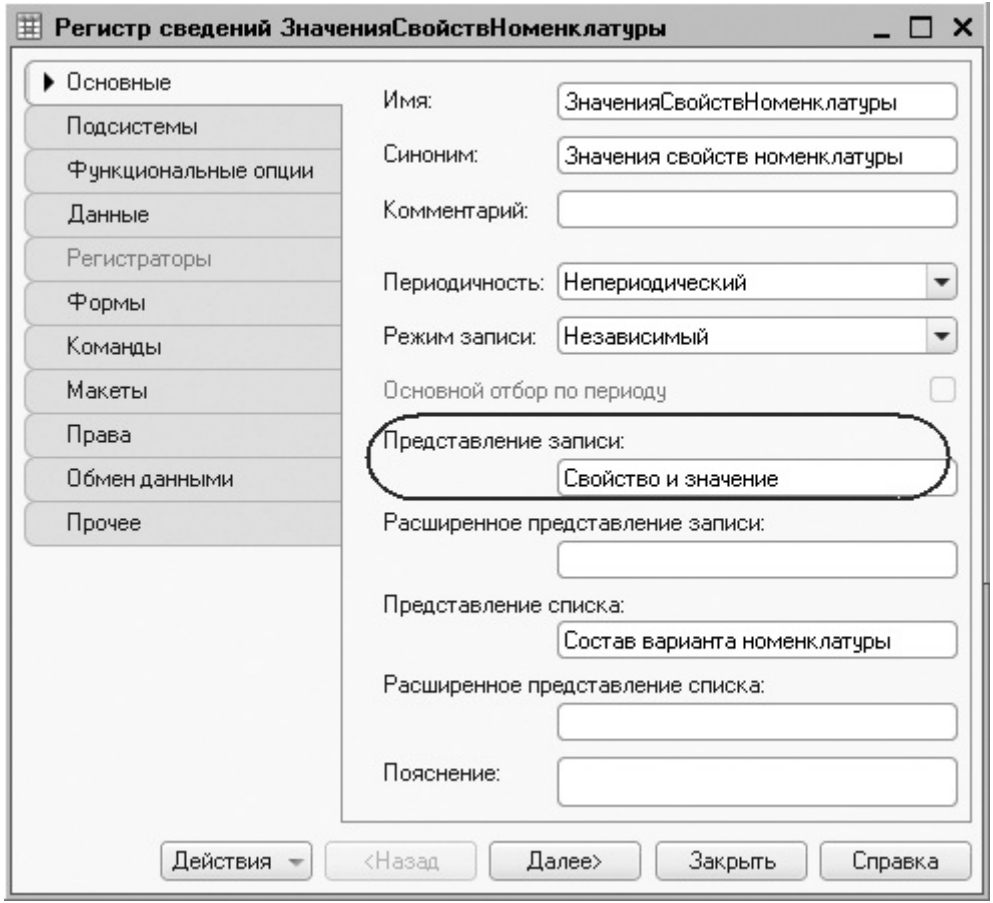


Рис. 15.27. Установка представления записи регистра

Это свойство будет использоваться в интерфейсе «1С:Предприятия» как заголовок формы записи регистра.

Во-вторых, нужно убрать поле *НаборСвойств* из этой формы. Для этого в окне редактирования объекта конфигурации *Регистр сведений ЗначенияСвойствНоменклатуры* перейдем на закладку *Формы*, нажмем кнопку открытия  и создадим основную форму записи.

В окне структуры элементов формы выделим этот элемент и, нажав кнопку *Удалить* в командной панели, удалим его из формы.

В режиме «1С:Предприятие»

Проверим результат изменений в режиме *1С:Предприятие*. В результате форма записи регистра *ЗначенияСвойствНоменклатуры* примет вид (рис. 15.28).

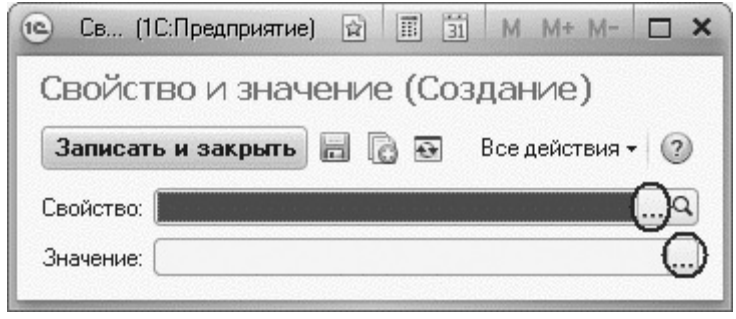


Рис. 15.28. Форма записи регистра «Значения свойств номенклатуры»

Создание характеристик номенклатуры

В режиме «1С:Предприятие»

Теперь создадим различные варианты номенклатуры в режиме *1С:Предприятие*.

Итак, в разделе *Учет материалов* откроем справочник *Номенклатура* и его элемент *Кабель электрический* из группы *Материалы* > *Прочее*.

В форме элемента номенклатуры выполним команду *Варианты номенклатуры* для перехода к списку наборов свойств данного элемента номенклатуры.

В форме списка вариантов номенклатуры откроем набор свойств *Белые кабели*, который мы создали ранее.

В форме варианта номенклатуры выполним команду *Состав варианта номенклатуры* для перехода к составу редактируемого варианта номенклатуры. Этот список пока пуст.

Нажмем кнопку *Создать*. В открывшейся форме (см. рис. 15.28) создадим свойство *Цвет* со значением *Белый*. Для этого нажмем кнопку выбора ... в поле *Свойство*.

Измерение *ВидСвойства*(*Свойство*) регистра *ЗначенияСвойствНоменклатуры* имеет тип *ПланВидовХарактеристикСсылка.СвойстваНоменклатуры*. Поэтому перед нами появится форма выбора этого плана видов характеристик. Список характеристик пока пуст.

Нажмем кнопку *Создать*. В открывшемся окне формы элемента плана видов характеристик введем наименование характеристики – *Цвет*, *Тип значения* этой характеристики оставим по умолчанию – *Дополнительные свойства номенклатуры* (рис. 15.29).

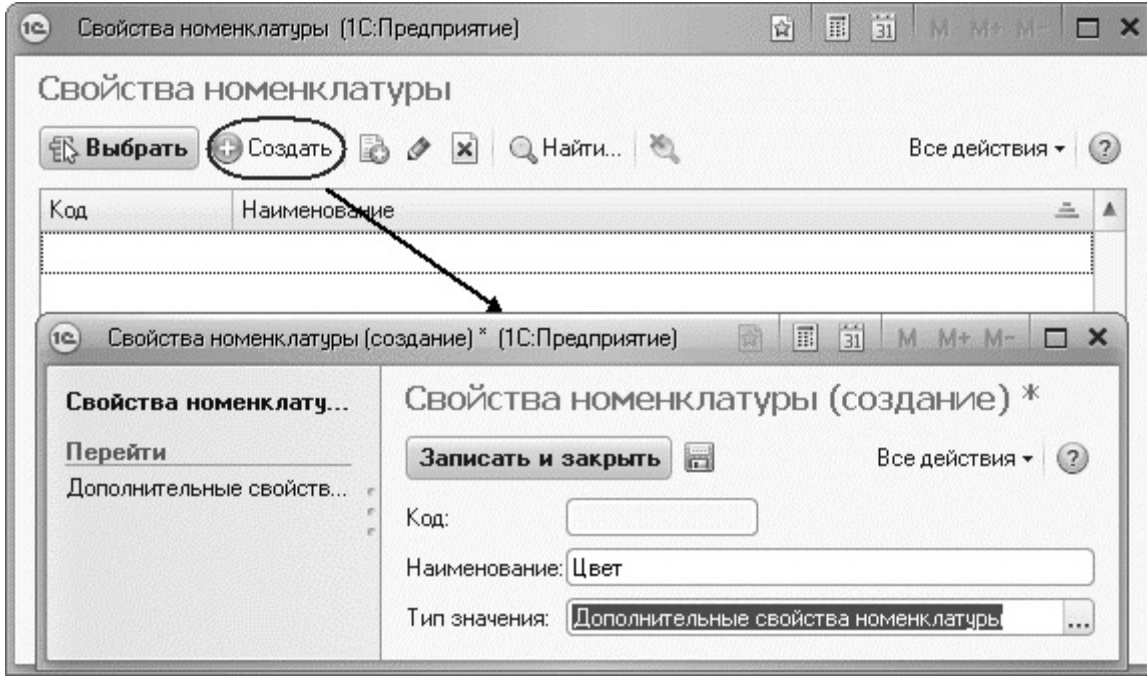


Рис. 15.29. Создание характеристики в плане видов характеристик

Обратите внимание, что в форме элемента плана видов характеристик (см. рис. 15.29) и в форме элемента справочника дополнительных характеристик номенклатуры (см. рис. 15.30) также есть лишнее поле *Код*.

Мы не стали дорабатывать эти формы, так как уже показывали подобные действия ранее. Вы можете сделать это самостоятельно аналогично тому, как это показано для формы элемента справочника *ВариантыНоменклатуры*.

Нажмем *Записать и закрыть*. В окне выбора плана видов характеристик появится созданная нами характеристика.

Нажмем кнопку *Выбрать*. В результате мы вернемся в форму записи состава варианта номенклатуры с заголовком *Свойство и значение*.

Нажмем кнопку выбора ... в поле *Значение*.

Ресурс *Значение* регистра *ЗначенияСвойствНоменклатуры* имеет тип *Характеристика.СвойстваНоменклатуры*. Это составной тип данных, который описан в свойстве *Тип значения характеристик* плана видов характеристик *СвойстваНоменклатуры*.

Так как для характеристики *Цвет* мы задали тип значения *СправочникСсылка.ДополнительныеСвойстваНоменклатуры*, то перед нами появится форма выбора этого справочника. Список свойств пока пуст.

Нажмем кнопку *Создать*. В открывшемся окне формы элемента дополнительных свойств номенклатуры введем тип значения *Белый*, в поле *Владелец* оставим имеющееся значение – *Цвет* (рис. 15.30).

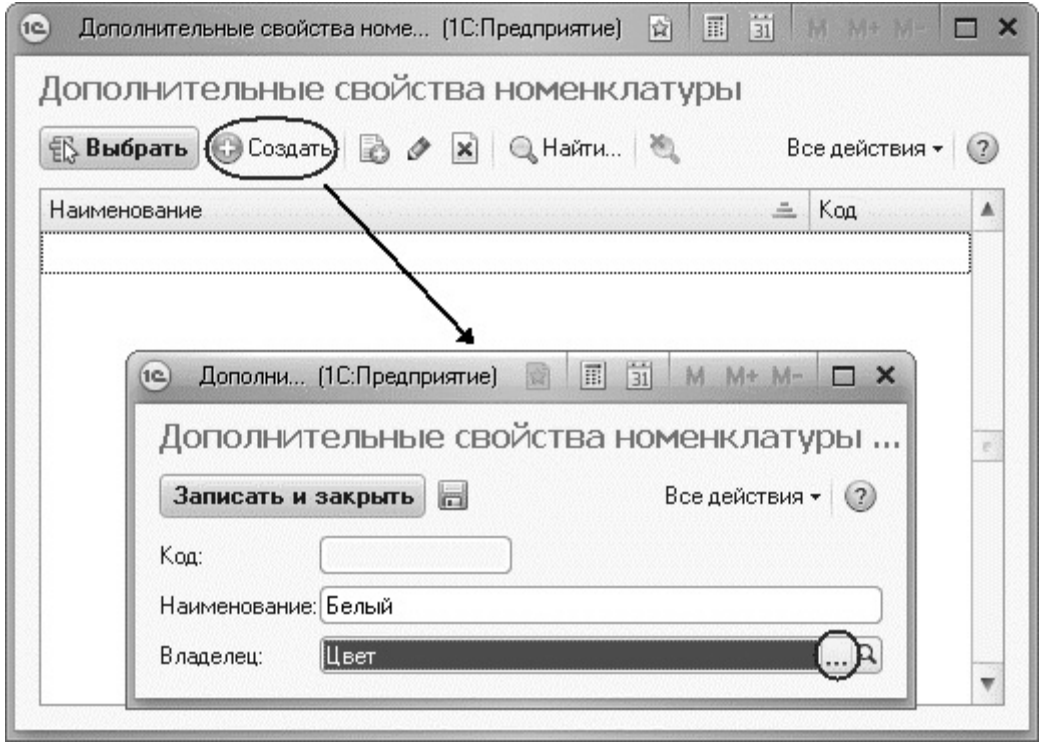


Рис. 15.30. Создание дополнительных свойств номенклатуры

Нажмем *Записать и закрыть*. В окне выбора дополнительных свойств номенклатуры появится созданное нами значение.

Нажмем кнопку *Выбрать*. Мы вернемся в форму записи состава варианта номенклатуры с заголовком *Свойство и значение* и увидим там созданное нами свойство *Цвет* со значением *Белый* (рис. 15.31).

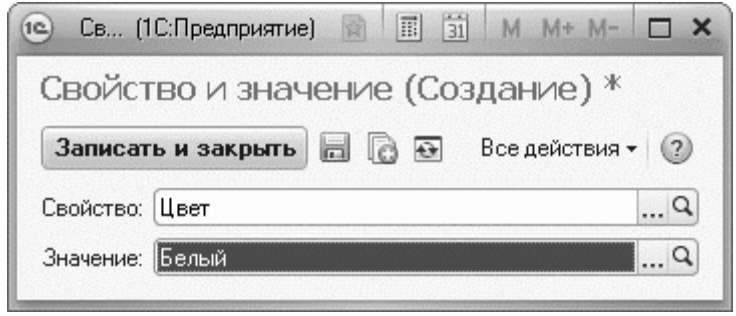


Рис. 15.31. Свойство и значение в составе варианта номенклатуры

Нажмем *Записать и закрыть*. Мы вернемся в форму списка состава варианта номенклатуры.

Создадим еще одно свойство – *Сечение, мм2* – в составе варианта номенклатуры *Белые кабели*. Для этого повторим только что выполненные действия.

Нажмем кнопку *Создать* (рис. 15.32).

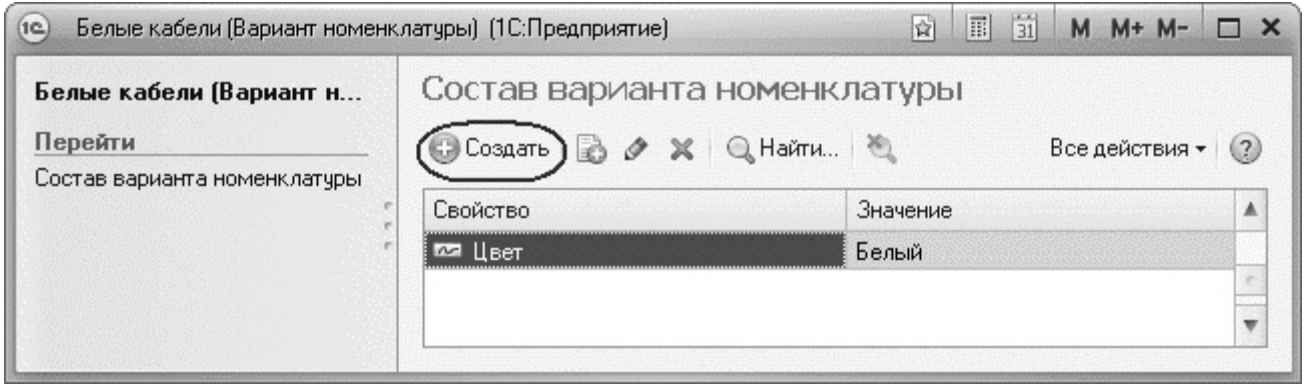


Рис. 15.32. Создание нового свойства в составе варианта номенклатуры

В открывшейся форме записи состава варианта номенклатуры нажмем кнопку выбора ... в поле *Свойство*.

В форме выбора плана видов характеристик нажмем кнопку *Создать*.

В открывшемся окне формы элемента плана видов характеристик введем наименование характеристики – *Сечение, мм2* и выберем *Тип значения* этой характеристики – *Число*, длина *15*, точность *3* (рис. 15.33).

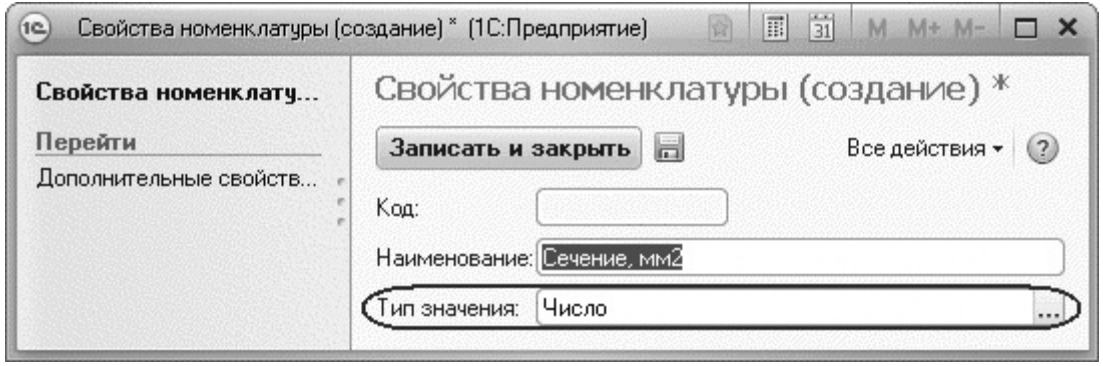


Рис. 15.33. Создание характеристики в плане видов характеристик

Нажмем *Записать и закрыть*. В окне выбора плана видов характеристик появится созданная нами характеристика.

Нажмем кнопку *Выбрать*. Мы вернемся в форму записи состава варианта номенклатуры с заголовком *Свойство и значение*.

Введем число *2,5* в поле *Значение* (рис. 15.34).

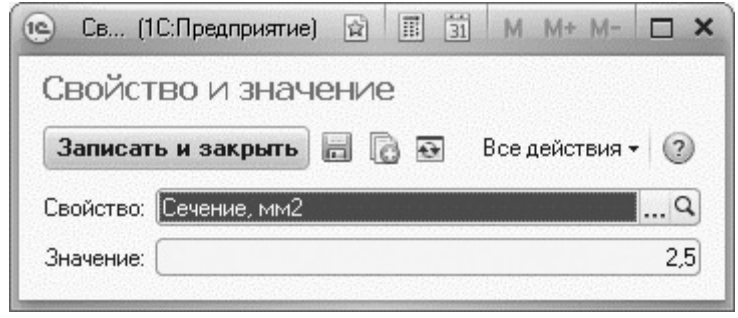


Рис. 15.34. Свойство и значение в составе варианта номенклатуры

Нажмем *Записать и закрыть*. Мы вернемся в форму списка состава варианта номенклатуры.

Итак, мы видим два свойства и их значения, которые мы создали для варианта номенклатуры *Белые кабели* (рис. 15.35).

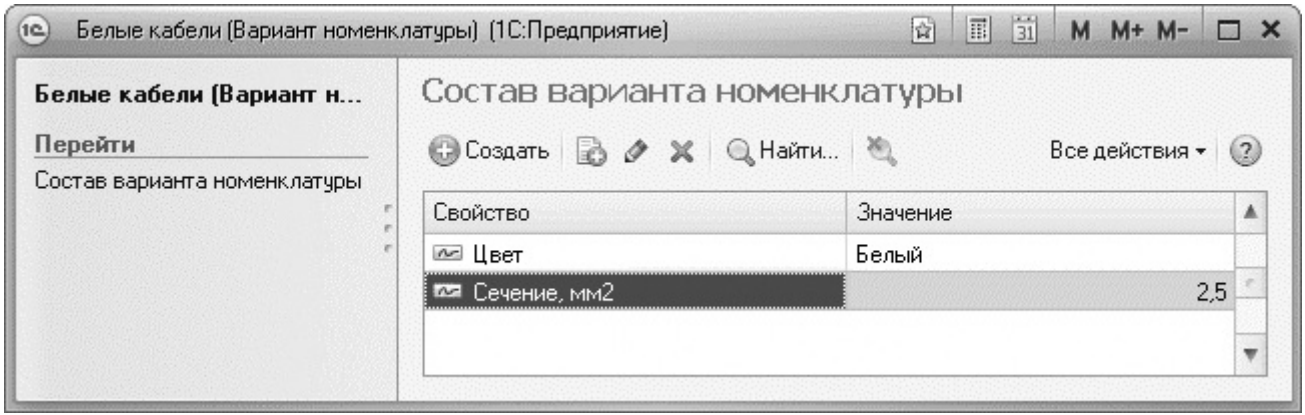


Рис. 15.35. Свойства и значения в составе варианта номенклатуры

Теперь аналогичным образом создадим набор свойств для элемента справочника *Номенклатура – Шланг резиновый*.

Этот набор свойств будет называться *Польша* (рис. 15.36) и состоять из следующих свойств (рис. 15.37):

- *Цвет – Черный*;
- *Производитель – Fagumit*.

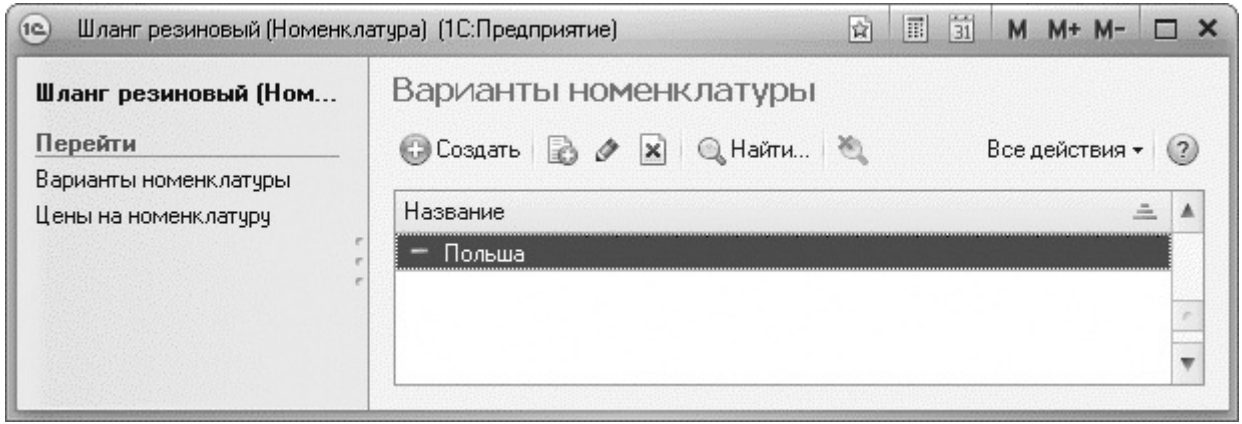


Рис. 15.36. Вариант номенклатуры для элемента номенклатуры «Шланг резиновый»

При создании свойства *Цвет* выберем его из уже имеющихся свойств в плане видов характеристик.

Значение этой характеристики – *Черный*, сначала добавим в справочник дополнительных свойств номенклатуры и затем выберем из него.

При создании свойства *Производитель* с типом значения *Дополнительные свойства номенклатуры* сначала добавим это свойство в план видов характеристик (тип значения – *Дополнительные свойства номенклатуры*), а затем выберем из него.

Значение этой характеристики – *Fagumit*, сначала добавим в справочник дополнительных свойств номенклатуры и затем выберем из него.

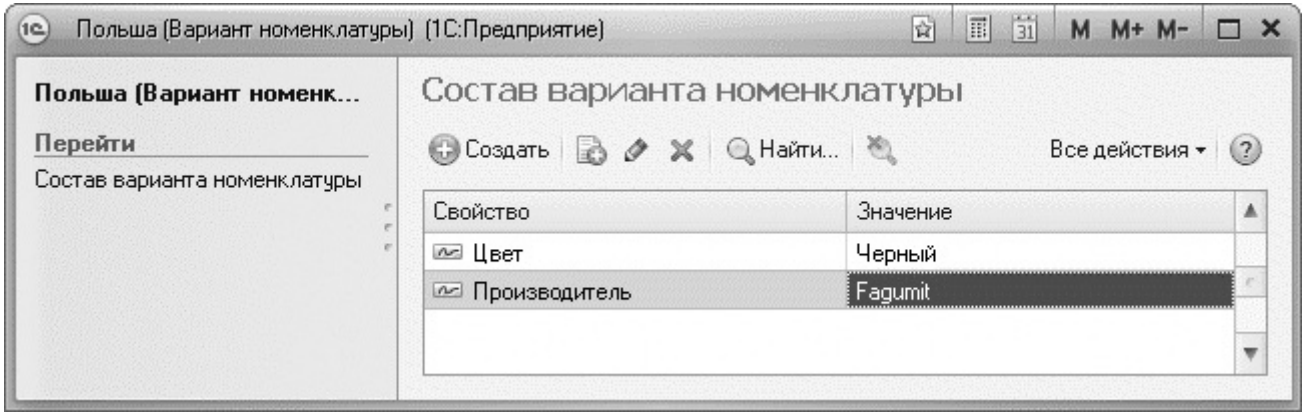


Рис. 15.37. Свойства и значения в составе варианта номенклатуры «Польша»

Теперь посмотрим на все, что мы создали не с точки зрения пользователя, а с точки зрения разработчика.

Перейдем в главное меню программы. Для этого нажмем на пиктограмму с черным треугольником, расположенную в заголовке основного окна программы рядом с символом «1С». Выполним команду главного меню *Все функции*. Поочередно откроем все объекты конфигурации, в которых хранится информация о созданных нами характеристиках номенклатуры.

В справочнике *Варианты номенклатуры* хранятся созданные нами наборы свойств номенклатуры. При этом каждый набор свойств подчинен конкретному элементу номенклатуры.

В плане видов характеристик *Свойства номенклатуры* хранятся созданные нами характеристики номенклатуры:

- *Цвет*, тип значения *Дополнительные свойства номенклатуры;*
- *Сечение, мм2*, тип значения *Число;*
- *Производитель*, тип значения *Дополнительные свойства номенклатуры.*

В справочнике *Дополнительные свойства номенклатуры* хранятся значения этих характеристик (за исключением характеристики *Сечение* типа *Число*). А в регистре сведений *ЗначенияСвойствНоменклатуры* хранятся соответствия характеристик и их значений в разрезе наборов свойств.

Взаимодействие этих объектов конфигурации представлено на следующей

схеме (рис. 15.38).

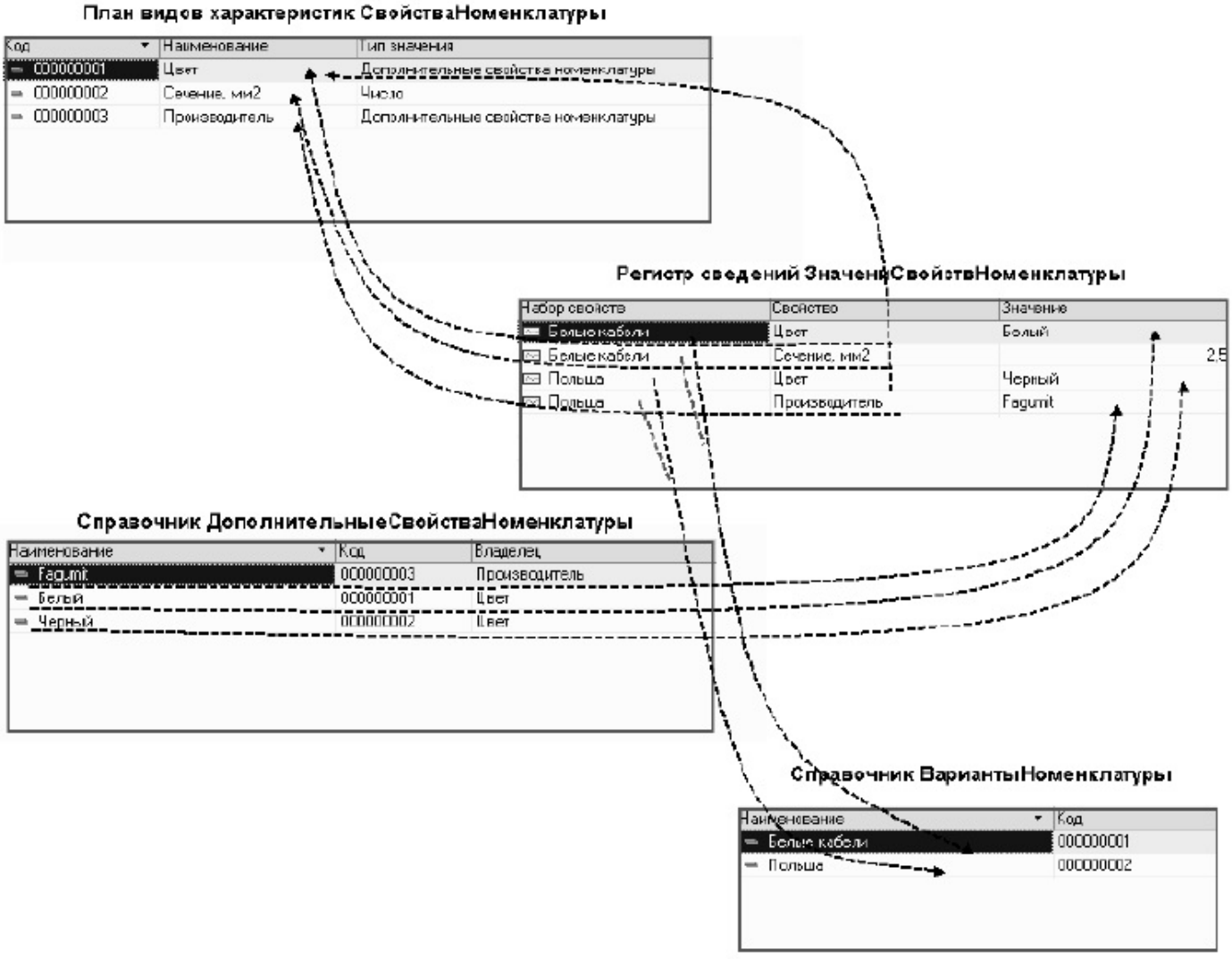

План видов характеристик СвойстваНоменклатуры
Код
Наименование
Тип значения
00000001
Цвет
Дополнительные свойства номенклатуры
00000002
Сечение, мм2
Число
00000003
Производитель
Дополнительные свойства номенклатуры
Регистр сведений ЗначенияСвойствНоменклатуры
Набор свойств
Свойство
Значение
Белые кабели
Цвет
Белый
Белые кабели
Сечение, мм2
2,5
Польша
Цвет
Черный
Польша
Производитель
Fagumit
Справочник ДополнительныеСвойстваНоменклатуры
Наименование
Код
Владелец
Fagumit
00000003
Производитель
Белый
00000001
Цвет
Черный
00000002
Цвет
Справочник ВариантыНоменклатуры
Наименование
Код
Белые кабели
00000001
Польша
00000002

Рис. 15.38. Объекты конфигурации, в которых хранится информация о характеристиках номенклатуры

Доработка учетных механизмов

Итак, мы добавили возможность указывать произвольные характеристики для номенклатуры и создали несколько таких характеристик – вариантов номенклатуры.

Но это лишь часть работы. Теперь хотелось бы иметь возможность еще и учитывать номенклатуру в разрезе этих характеристик. А именно:

- приходовать товар, указывая характеристики;
- расходовать товар, указывая характеристики;
- получать отчеты не просто по номенклатуре, а по номенклатуре с определенными характеристиками.

Для этого потребуется доработать имеющиеся регистры и создать новый отчет, который позволит получать данные в разрезе свойств номенклатуры.

Регистр «Остатки материалов»

В режиме «Конфигуратор»

Чтобы обеспечить учет материалов по значениям характеристик, необходимо

изменить структуру регистра накопления *ОстаткиМатериалов*, чтобы хранить в нем данные еще и в разрезе наборов свойств номенклатуры.

Для этого откроем окно редактирования объекта конфигурации *Регистр накопления ОстаткиМатериалов* и на закладке *Данные* добавим в него новое измерение *НаборСвойств* с типом *СправочникСсылка.ВариантыНоменклатуры* (рис. 15.39).

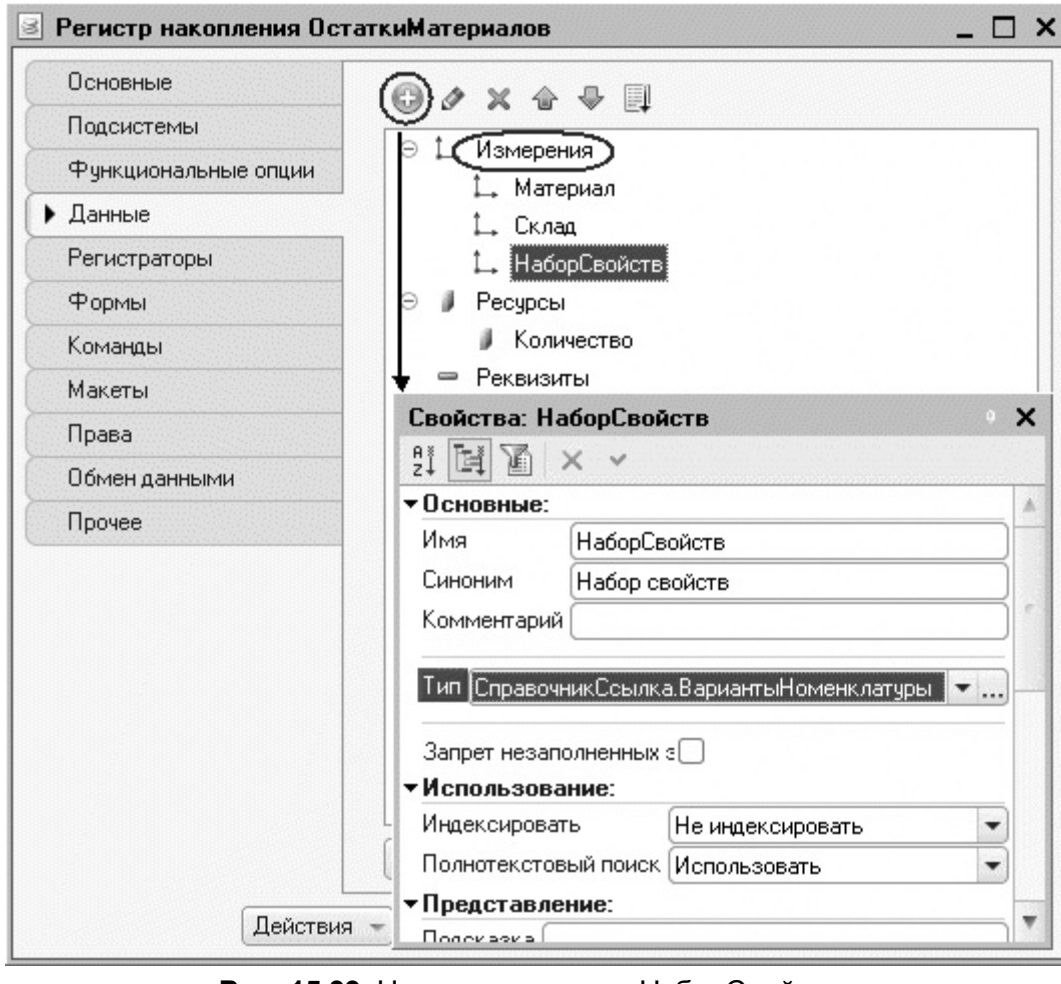


Рис. 15.39. Новое измерение «НаборСвойств»

Документ «Приходная накладная»

В режиме «Конфигуратор»

Теперь нам нужно доработать документ *ПриходнаяНакладная,* чтобы при приходовании материалов можно было указать набор свойств и чтобы этот набор свойств записывался в регистры при проведении документа.

Для этого откроем окно редактирования объекта конфигурации *Документ ПриходнаяНакладная* и на закладке *Данные* добавим в табличную часть документа новый реквизит *НаборСвойств* с типом *СправочникСсылка.ВариантыНоменклатуры* (рис. 15.40).

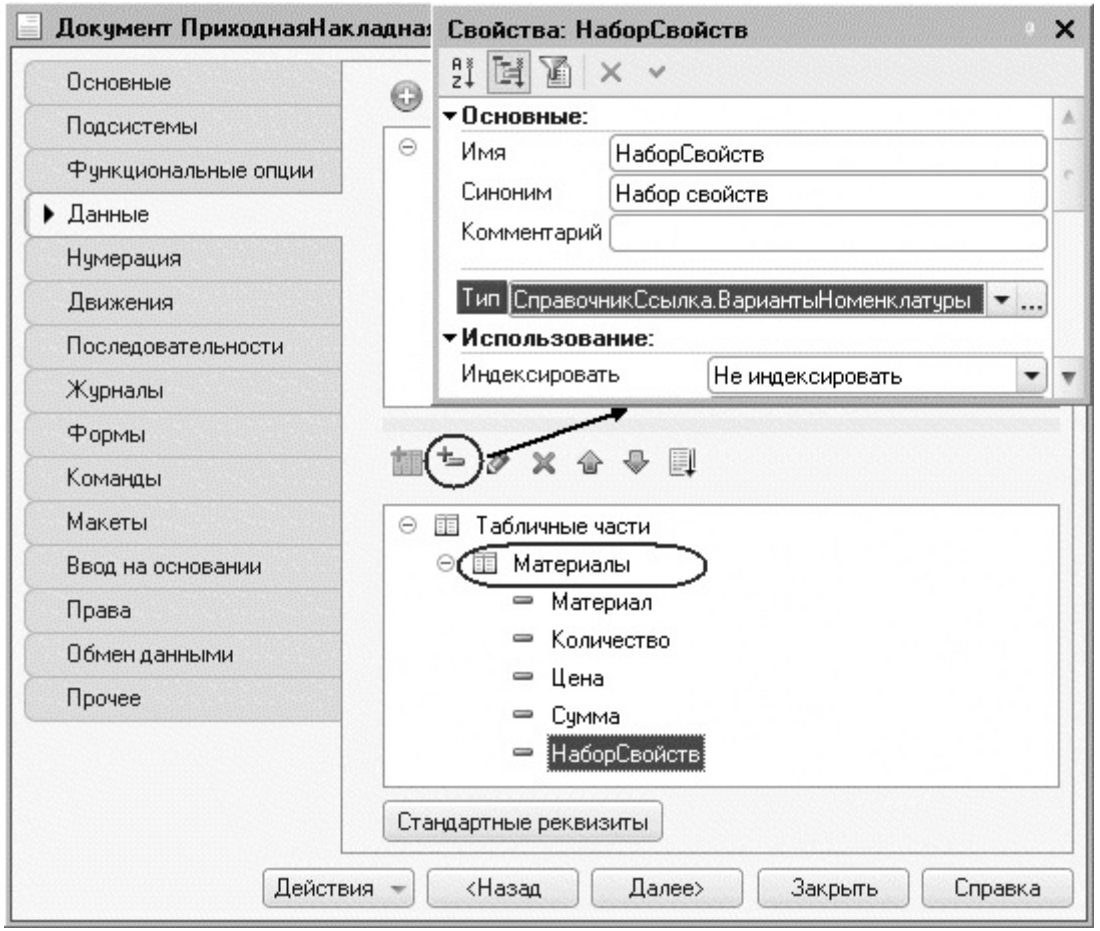


Рис. 15.40. Новый реквизит «НаборСвойств»

У этого реквизита необходимо заполнить свойство *Связи параметров выбора,* чтобы после выбора номенклатуры в этом свойстве выбирать только среди тех

наборов свойств, которые относятся к данной номенклатуре.

Найдем в палитре свойств свойство *Связи параметров выбора* и нажмем кнопку выбора ....

Перенесем из списка доступных реквизитов в список параметров реквизит *Материалы.Материал* (рис. 15.41).

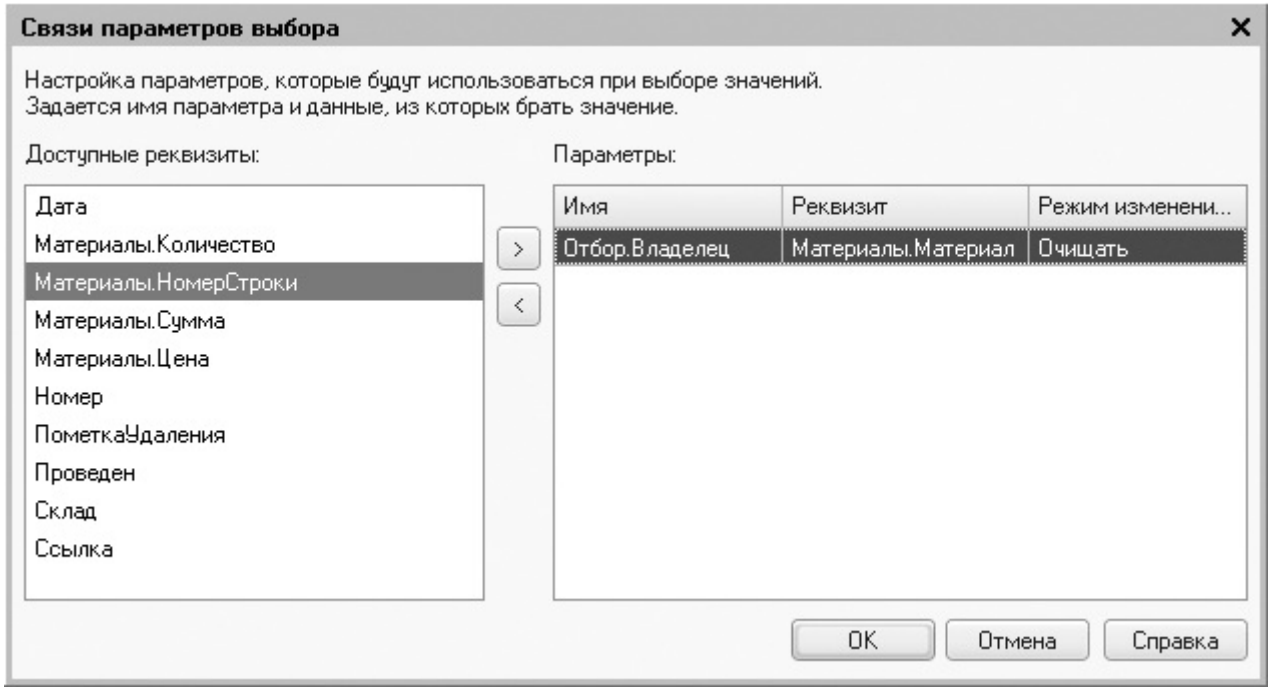


Рис. 15.41. Связи параметров выбора

Тем самым мы задали, что при выборе в поле *НаборСвойств* будет всегда открываться список элементов справочника *Варианты номенклатуры*, подчиненных материалу, выбранному в колонке *Материал*.

После этого расположим этот реквизит в табличной части формы документа.

Для этого перейдем на закладку *Формы* и двойным щелчком мыши на строке *ФормаДокумента* в списке форм откроем форму документа.

Затем в правом верхнем окне редактора форм на закладке *Реквизиты* раскроем реквизит формы *Объект*.

Мы видим, что он содержит все реквизиты документа *ПриходнаяНакладная*.

Найдем в табличной части реквизит *НаборСвойств* и с помощью мыши перетащим его в окно элементов формы, расположенное слева в верхней части редактора форм, в таблицу *Материалы*.

Новый элемент расположим в структуре элементов формы после поля *Материал* (рис. 15.42).

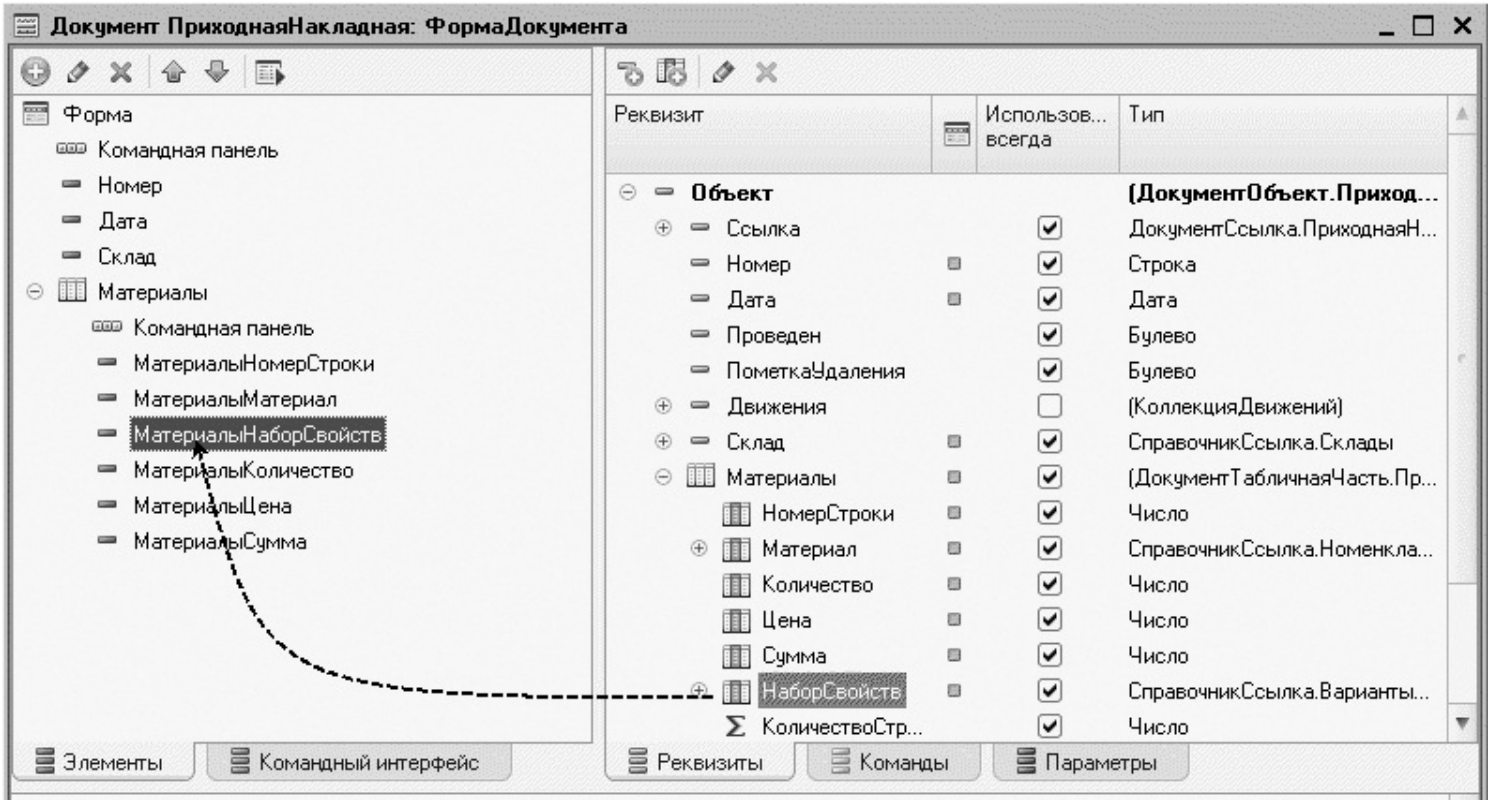


Рис. 15.42. Изменение формы документа «Приходная накладная»

Обратите внимание, что в открывшейся палитре свойств элемента формы *НаборСвойств* в свойстве *ПутьКДанным* уже указан реквизит табличной части *НаборСвойств*, так как мы перетаскивали реквизит в форму с помощью мыши, и оно заполнилось автоматически.

Свойство *ПутьКДанным* устанавливает связь элемента формы с реквизитом формы, то есть с отображаемыми данными. Это свойство обязательно должно быть заполнено, иначе элемент формы не будет показан!

ПРИМЕЧАНИЕ

При добавлении элемента формы с помощью кнопки ***Добавить*** *свойство* ***ПутьКДанным****, устанавливающее связь элемента с реквизитом формы, необходимо заполнять вручную.*

В заключение в окне редактирования объекта конфигурации Документ *ПриходнаяНакладная* на закладке *Прочее* откроем модуль объекта.

Откроем процедуру обработчика события *ОбработкаПроведения* и добавим к формируемым движениям присвоение значения измерению *НаборСвойств* регистра *ОстаткиМатериалов* (листинг 15.3).

Листинг 15.3. Фрагмент процедуры «ОбработкаПроведения()»

```
…
// регистр ОстаткиМатериалов Приход
…
Движение.Материал = ТекСтрокаМатериалы.Материал;
Движение.НаборСвойств = ТекСтрокаМатериалы.НаборСвойств;
Движение.Склад = Склад;
…
```

Документ «Оказание услуги»

В режиме «Конфигуратор»

Теперь аналогичным образом доработаем документ *ОказаниеУслуги*.

Для того чтобы при расходовании материалов пользователь мог указывать набор свойств для каждого расходуемого материала, откроем окно редактирования объекта конфигурации *Документ ОказаниеУслуги* и на закладке *Данные* добавим в табличную часть документа новый реквизит *НаборСвойств* с типом *СправочникСсылка.ВариантыНоменклатуры*.

У этого реквизита заполним свойство *Связи параметров выбора*. Перенесем из списка доступных реквизитов в список параметров реквизит *ПеречеьНоменклатуры.Номенклатура*. Тем самым мы задали, что при выборе в поле *НаборСвойств* будет всегда открываться список элементов справочника *Варианты номенклатуры*, подчиненных материалу, выбранному в колонке *Номенклатура*.

После этого расположим этот реквизит в табличной части формы документа. Откроем форму документа и с помощью мыши перетащим его из окна реквизитов формы в окно элементов формы. Новый элемент расположим в структуре элементов формы после поля *Номенклатура*.

В заключение в окне редактирования объекта конфигурации Документ *ОказаниеУслуги* на закладке *Прочее* откроем модуль объекта.

Откроем процедуру обработчика события *ОбработкаПроведения* и добавим к формируемым движениям присвоение значения измерению *НаборСвойств* регистра *ОстаткиМатериалов* (листинг 15.4).

Листинг 15.4. Фрагмент процедуры «ОбработкаПроведения()»

```
…
// регистр ОстаткиМатериалов Расход
…
Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
Движение.НаборСвойств = ВыборкаДетальныеЗаписи.НаборСвойств;
Движение.Склад = Склад;
…
```

Поскольку на предыдущем занятии мы оптимизировали процедуру проведения документа и получали все данные документа с помощью запроса, то в текст запроса нужно также добавить строки для получения нового реквизита документа (листинг 15.5).

Листинг 15.5. Фрагмент процедуры «ОбработкаПроведения()»

```
…
Запрос = Новый Запрос;
```

```
// Укажем, какой менеджер временных таблиц использует этот запрос
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос.Текст =
  "ВЫБРАТЬ
  |        ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
  |        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры КАК
  ВидНоменклатуры,
  |        ОказаниеУслугиПереченьНоменклатуры.НаборСвойств,
  |        СУММА(ОказаниеУслугиПереченьНоменклатуры.Количество) КАК
  КоличествоВДокументе,
  |        СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК СуммаВДокументе
  |ПОМЕСТИТЬ НоменклатураДокумента
  |ИЗ
  |        Документ.ОказаниеУслуги.ПереченьНоменклатуры КАК
  ОказаниеУслугиПереченьНоменклатуры
  |ГДЕ
  |        ОказаниеУслугиПереченьНоменклатуры.Ссылка = &Ссылка
  |СГРУППИРОВАТЬ ПО
  |        ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
  |        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры,
  |        ОказаниеУслугиПереченьНоменклатуры.НаборСвойств";
...
Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос2.Текст = "ВЫБРАТЬ
  |        НоменклатураДокумента.Номенклатура,
  |        НоменклатураДокумента.ВидНоменклатуры,
  |        НоменклатураДокумента.НаборСвойств,
  |        НоменклатураДокумента.КоличествоВДокументе,
```

```
   |        НоменклатураДокумента.СуммаВДокументе,
   |        ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток, 0) КАК
  Стоимость,
   |        ЕСТЬNULL(ОстаткиМатериаловОстатки.КоличествоОстаток, 0) КАК
  Количество
   |ИЗ
   |        НоменклатураДокумента КАК НоменклатураДокумента
…
```

Кроме этого, понадобится изменить последний запрос, который при оперативном проведении проверяет, не появились ли отрицательные остатки. Теперь мы будем получать остатки не «вообще» для номенклатуры из табличной части документа, а для номенклатуры именно с тем набором свойств, который указан в строках документа (листинг 15.5а).

Листинг 15.5а. Контроль отрицательных остатков при оперативном проведении

```
…
Запрос3.Текст = "ВЫБРАТЬ
   |        ОстаткиМатериаловОстатки.Материал,
   |        ОстаткиМатериаловОстатки.НаборСвойств,
   |        ОстаткиМатериаловОстатки.КоличествоОстаток
   |ИЗ
   |        РегистрНакопления.ОстаткиМатериалов.Остатки( , (Материал,
  НаборСвойств) В
   |                (ВЫБРАТЬ
   |                        НоменклатураДокумента.Номенклатура,
```

```
    |                            НоменклатураДокумента.НаборСвойств
    |                ИЗ
    |                            НоменклатураДокумента) И Склад = &Склад)
    |          КАК ОстаткиМатериаловОстатки
    |ГДЕ
    |          ОстаткиМатериаловОстатки.КоличествоОстаток ;
…
```

Приход/расход номенклатуры с учетом характеристик

В режиме «1С:Предприятие»

Теперь запустим «1С:Предприятие» в режиме отладки и укажем наборы свойств при приходовании материалов.

Откроем документ *Приходная накладная № 2* и укажем, что был закуплен белый электрический кабель в количестве 2 шт. и польский резиновый шланг.

Затем скопируем первую строку документа и укажем, что был закуплен еще и черный электрический кабель в количестве 3 шт. (в процессе ввода нам придется создать еще один набор свойств для электрического кабеля – *Черные кабели*, у которого *Цвет – Черный* и *Сечение – 2,5*), рис. 15.43.

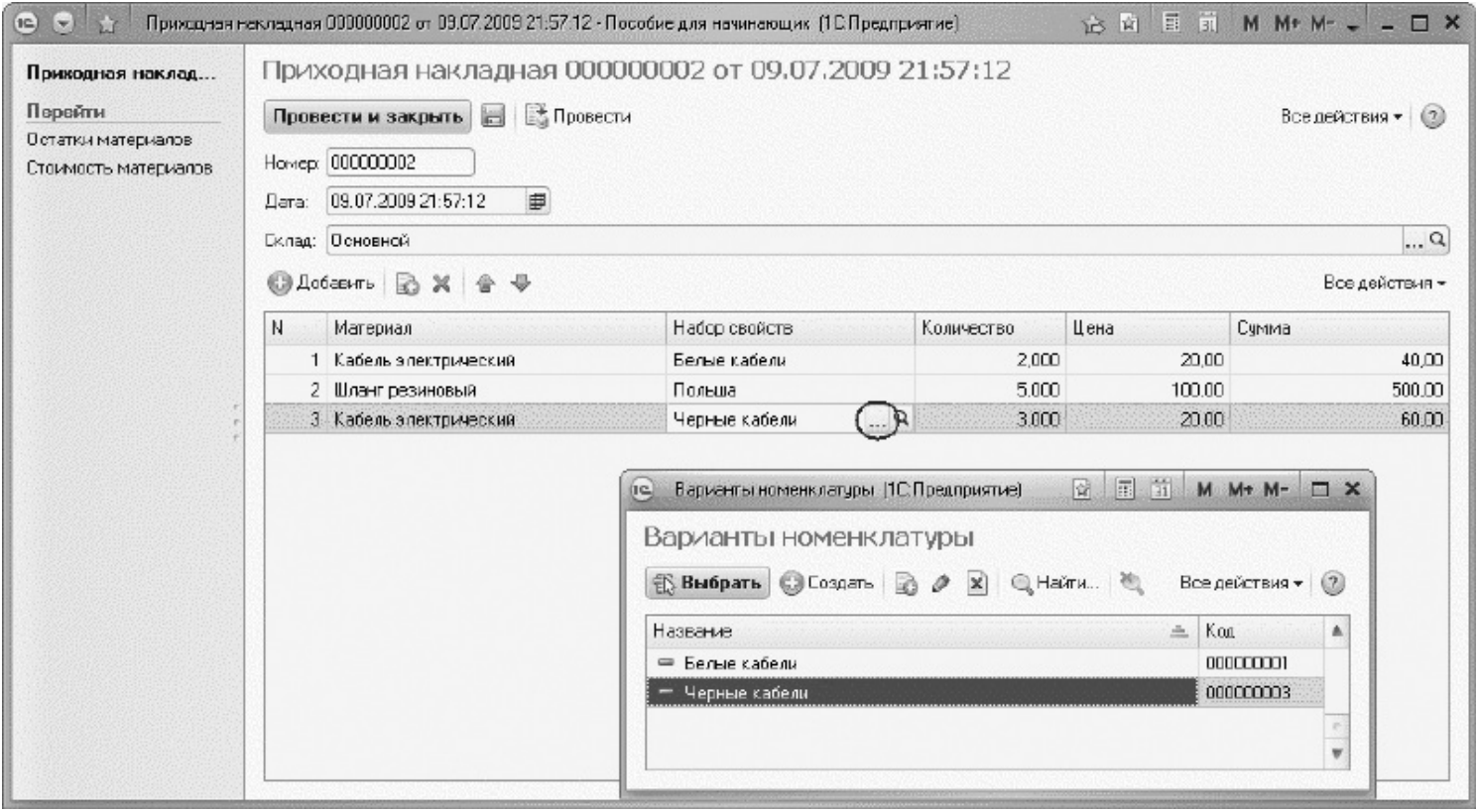


Рис. 15.43. Документ «Приходная накладная № 2»

Нажмем *Провести* и, выполнив команду *Остатки материалов* в панели навигации формы документа, проверим движения документа по регистру *ОстаткиМатериалов* (рис. 15.44).

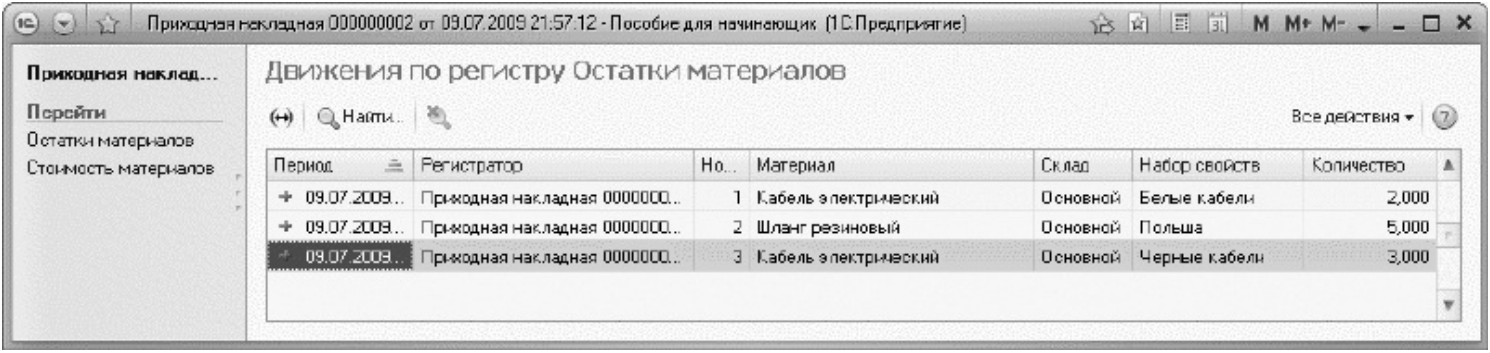


Рис. 15.44. Движения документа «Приходная накладная № 2» по регистру «Остатки материалов»

Теперь откроем документ *Оказание услуги № 1* и укажем, что был израсходован польский резиновый шланг (рис. 15.45).

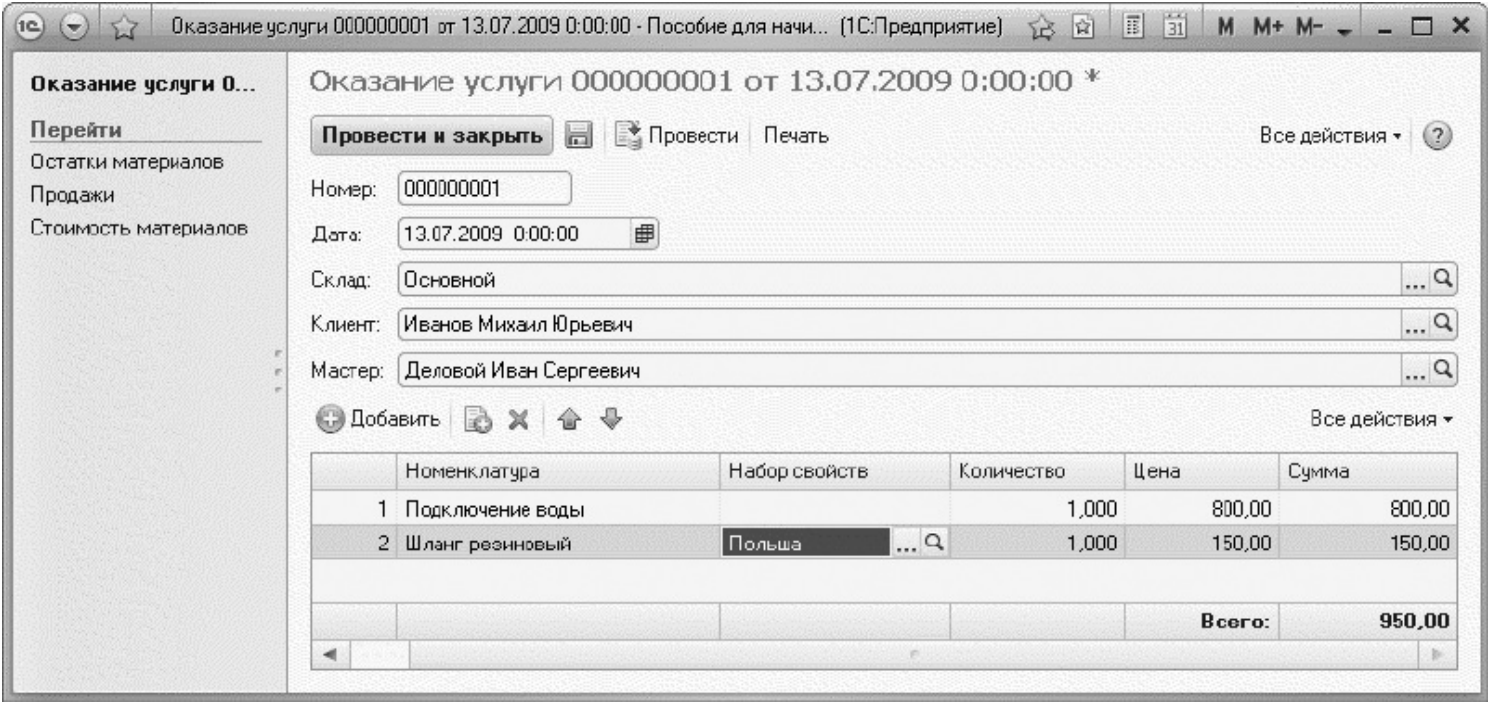


Рис. 15.45. Документ «Оказание услуги № 1»

Нажмем *Провести* и, выполнив команду *Остатки материалов* в панели навигации формы документа, проверим движения документа по регистру *ОстаткиМатериалов* (рис. 15.46).

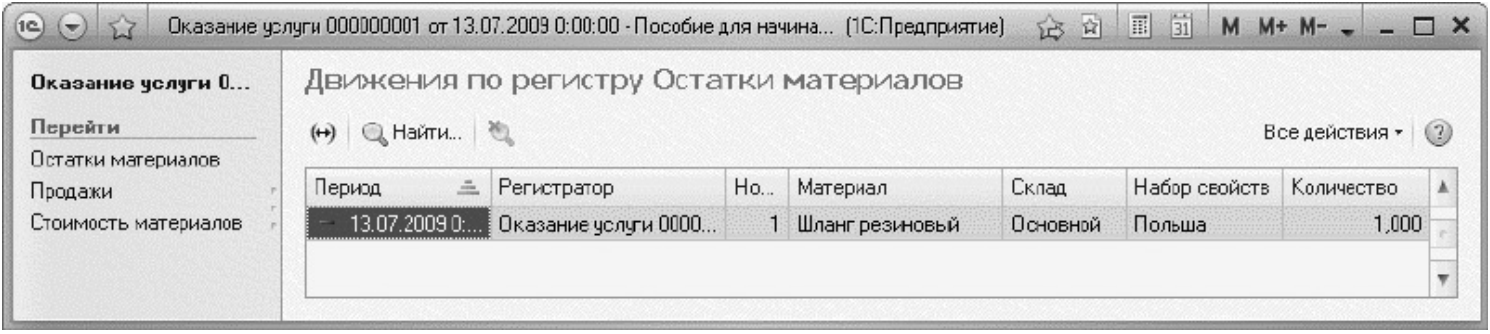
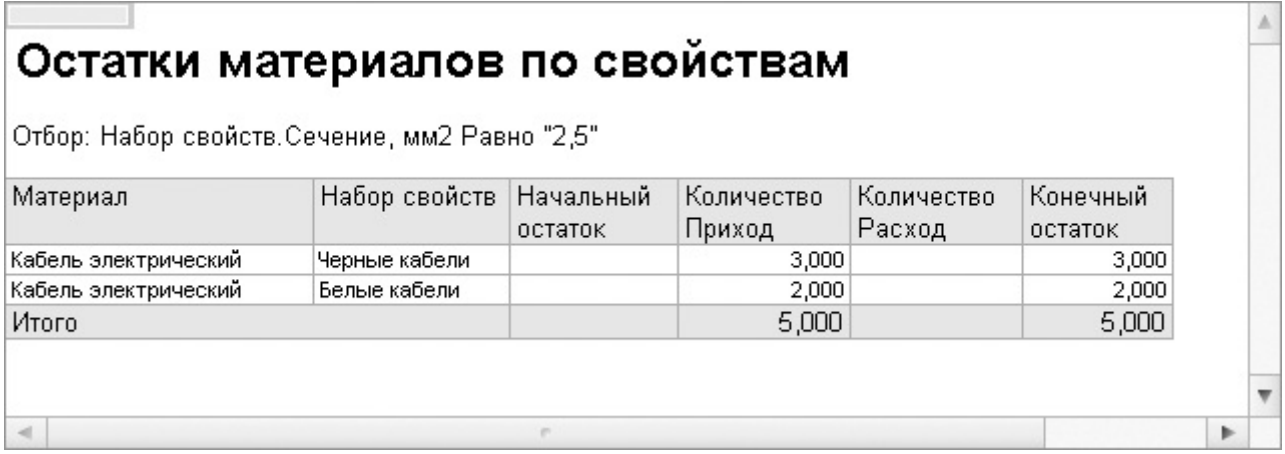


Рис. 15.46. Движения документа «Оказание услуги № 1» по регистру «Остатки материалов»

Отчет, использующий характеристики

Для полного завершения картины мы создадим отчет, который будет показывать нам наличие материалов с теми или иными свойствами. При создании этого отчета мы используем те возможности, которые предоставляет нам система компоновки данных для работы с характеристиками (рис. 15.47).



Остатки материалов по свойствам

Отбор: Набор свойств.Сечение, мм2 Равно "2,5"

Материал	Набор свойств	Начальный остаток	Количество Приход	Количество Расход	Конечный остаток
Кабель электрический	Черные кабели		3,000		3,000
Кабель электрический	Белые кабели		2,000		2,000
Итого			5,000		5,000

Рис. 15.47. Результат отчета

Коротко говоря, набором данных для системы компоновки данных будет довольно простой запрос к регистру *ОстаткиМатериалов*. А свойства вариантов номенклатуры платформа задействует в этом отчете автоматически, на основании того описания, которое мы создали у справочника *ВариантыНоменклатуры*.

Система компоновки данных сама сформирует достаточно понятный и удобный интерфейс для работы с характеристиками и в зависимости от значений, выбранных пользователем, будет формировать необходимые запросы к базе данных.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Отчет*. Назовем его *ОстаткиМатериаловПоСвойствам* и запустим конструктор схемы компоновки данных. Добавим новый *Набор данных – запрос* и вызовем конструктор запроса.

Запрос для набора данных

В качестве источника данных для запроса выберем виртуальную таблицу регистра накопления *ОстаткиМатериалов.ОстаткиИОбороты*. Из этой таблицы выберем следующие поля (рис. 15.48):

- *Материал,*
- *НаборСвойств,*
- *КоличествоНачальныйОстаток,*
- *КоличествоПриход,*
- *КоличествоРасход,*
- *КоличествоКонечныйОстаток.*

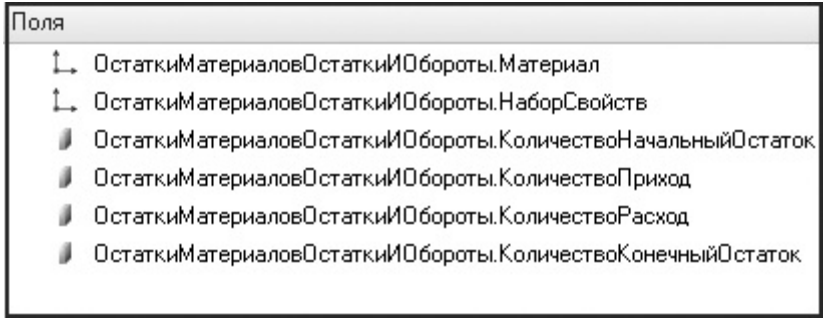


Рис. 15.48. Выбранные поля

После этого на закладке *Объединения/Псевдонимы* зададим псевдонимы числовых полей без слова *Количество* (рис. 15.49).

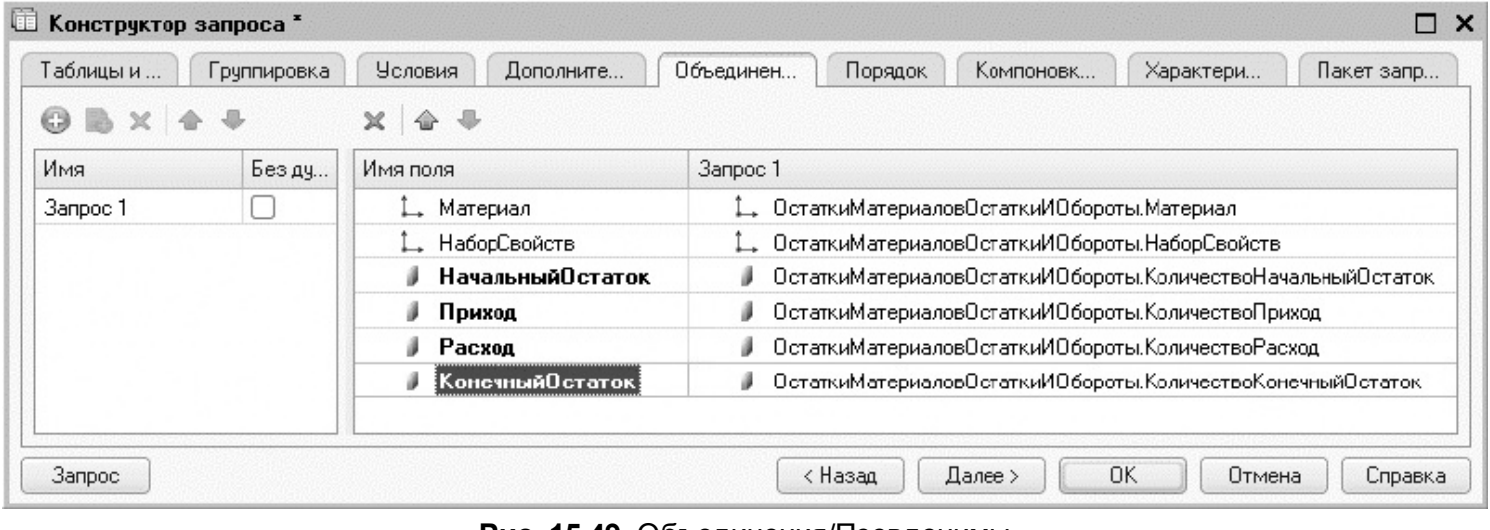


Рис. 15.49. Объединения/Псевдонимы

На этом создание запроса закончено. Нажмем *OK.*

Ресурсы

Приступим к редактированию схемы компоновки данных.

Прежде всего, на закладке *Ресурсы* выберем все доступные ресурсы (рис. 15.50).

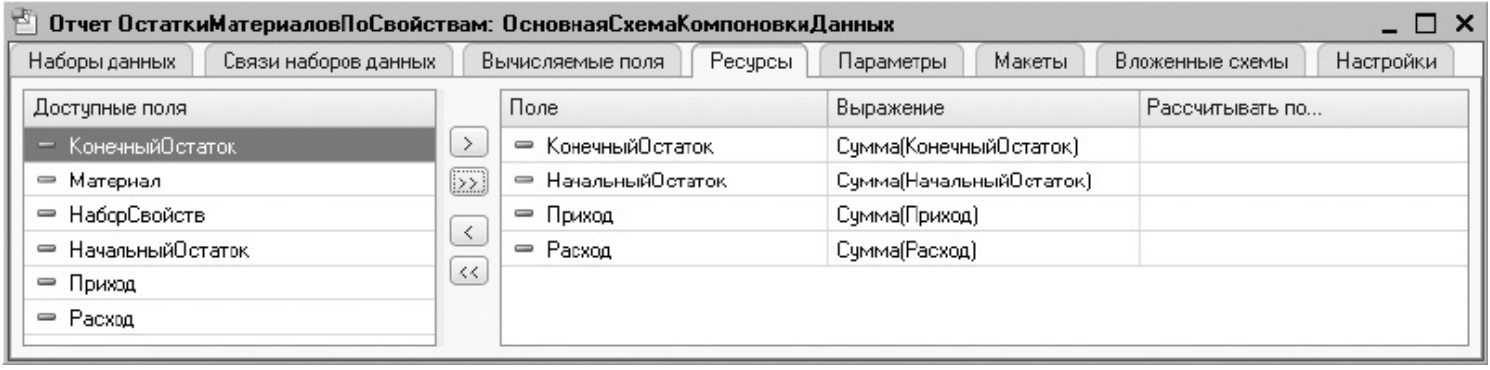


Рис. 15.50. Описание ресурсов

Настройки

Перейдем на закладку *Настройки*. Создадим структуру отчета – добавим группировку *Детальные записи*.

Затем на закладке *Выбранные поля* выберем те поля, которые будут

выводиться в отчет: *Материал*, *НаборСвойств*, *НачальныйОстаток*, *Приход*, *Расход* и *КонечныйОстаток* (рис. 15.51).

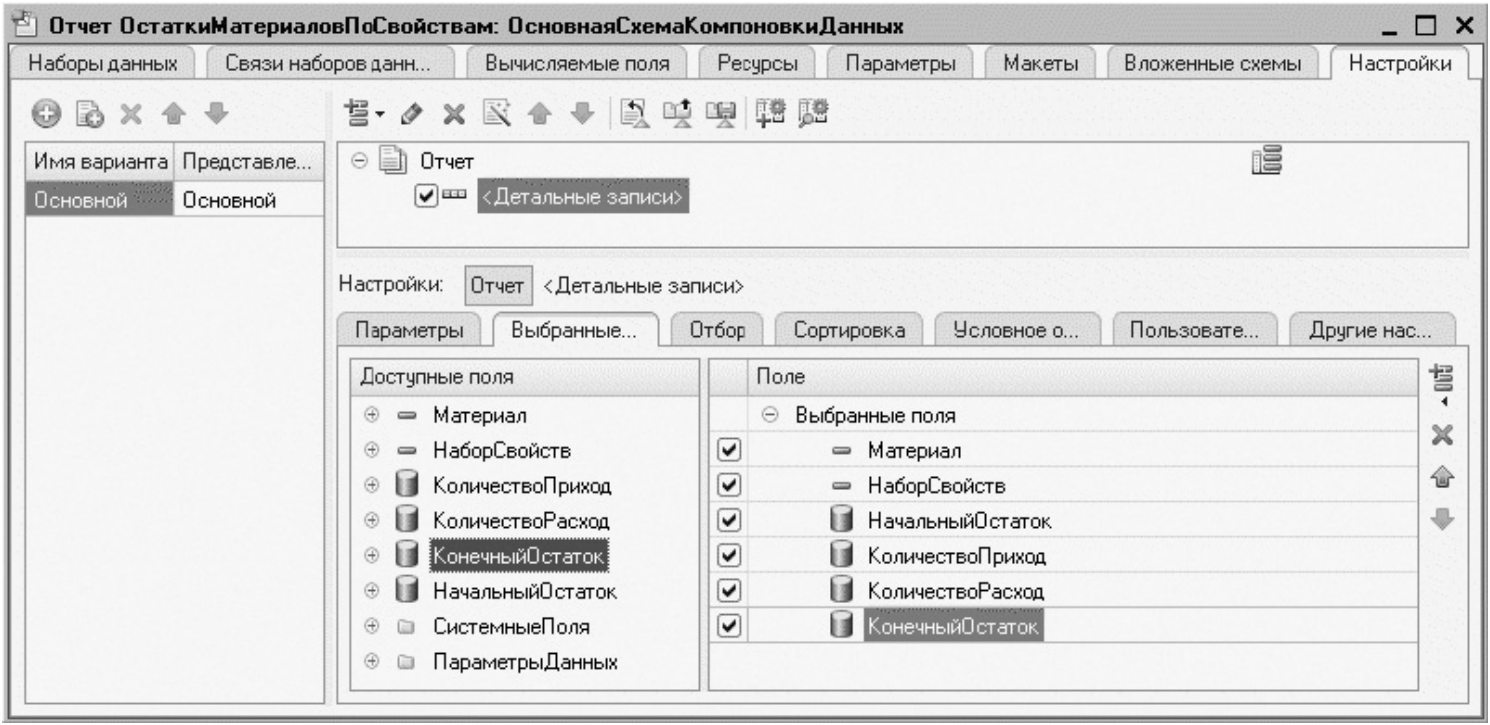


Рис. 15.51. Группировки и поля отчета

Затем перейдем на закладку *Другие настройки* и зададим заголовок отчета – *Остатки материалов по свойствам*.

Чтобы иметь возможность протестировать наш отчет, включим настройку *Отбор* в состав быстрых пользовательских настроек.

Для этого нажмем кнопку *Свойства элемента пользовательских настроек*, расположенную вверху в командной панели окна настроек.

В появившемся окне мы можем редактировать состав пользовательских настроек отчета. Установим признак использования для настройки *Отбор* и оставим предложенное для нее по умолчанию свойство *Режим редактирования* в значении *Быстрый* (рис. 15.52).

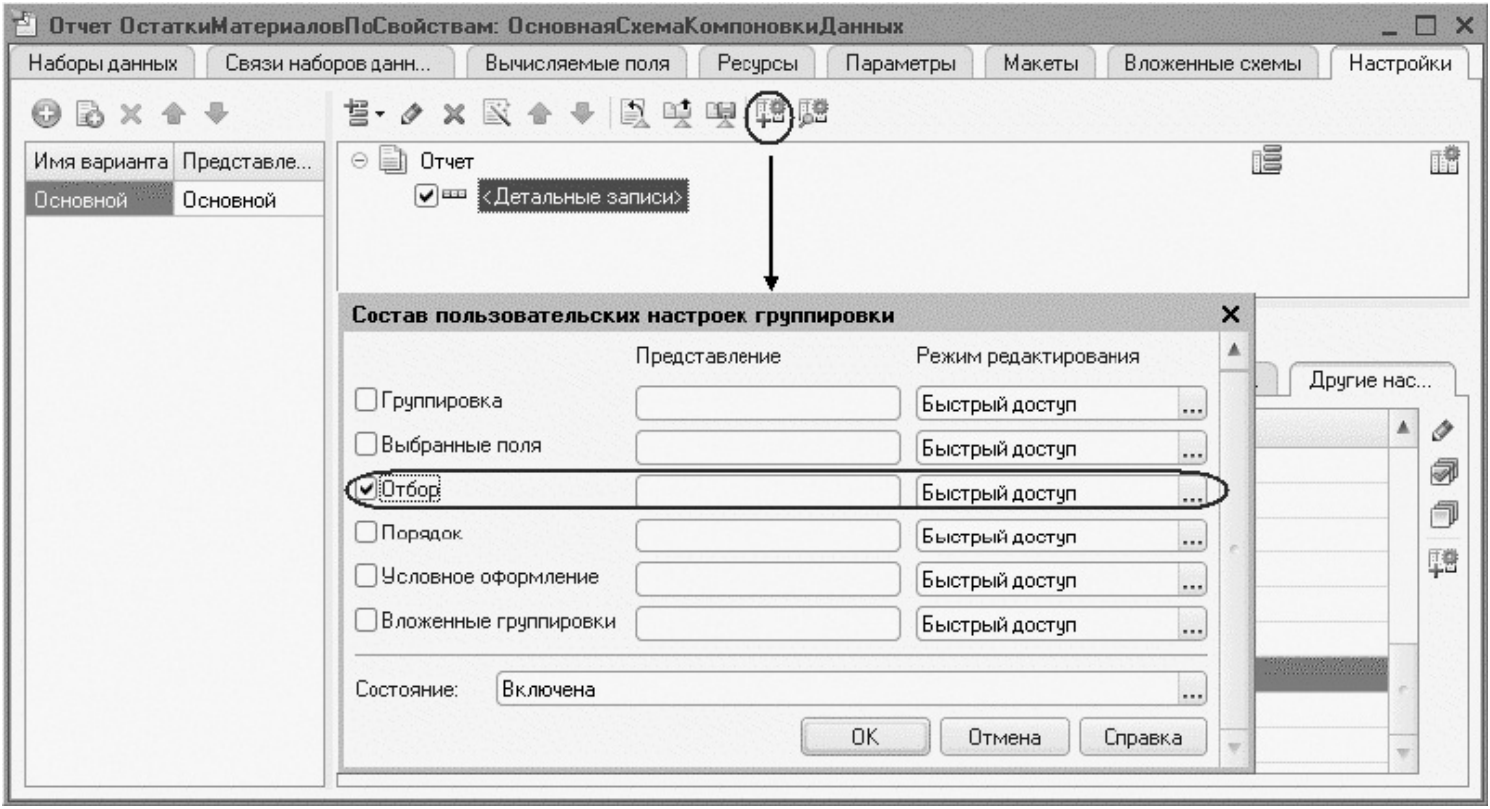


Рис. 15.52. Состав пользовательских настроек

В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации *Отчет ОстаткиМатериаловПоСвойствам* перейдем

на закладку *Подсистемы*.

Отметим в списке подсистем конфигурации подсистемы *Учет материалов* и *Бухгалтерия*.

На этом создание отчета завершено.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, какие результаты можно получить с помощью нашего отчета.

В разделе *Учет материалов* выполним команду открытия отчета *Остатки материалов по свойствам* (рис. 15.53).

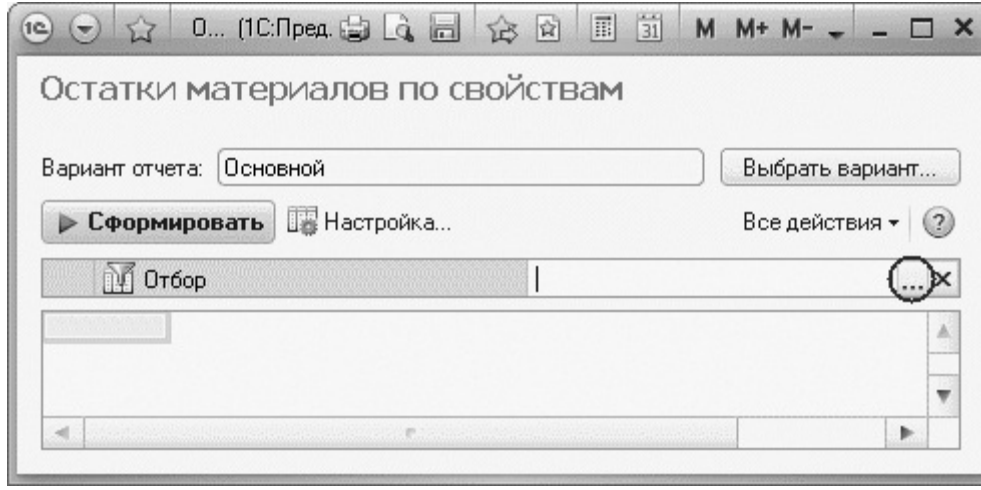


Рис. 15.53. Форма отчета

Мы видим настройку *Отбор*, расположенную в отчетной форме, с помощью которой мы можем получать остатки материалов в разрезе их характеристик.

Сначала посмотрим, какие у нас есть материалы с сечением 2,5 мм?.

Для этого в поле настройки *Отбор* нажмем кнопку выбора ... (см. рис. 15.53). В появившемся окне *Редактирование отбора* слева мы видим список доступных полей отчета.

Раскроем поле *Набор свойств* (рис. 15.54).

Обратите внимание, что к стандартным реквизитам справочника *ВариантыНоменклатуры* система компоновки данных добавила все характеристики, которые определены нами для различных наборов свойств в базе данных: *Производитель*, *Сечение* и *Цвет*. Таким образом, отбор в отчете по значениям каких-либо характеристик является достаточно простым и интуитивно понятным.

Чтобы узнать, какие у нас есть материалы с сечением 2,5 мм?, достаточно выбрать поле *Сечение, мм2* и задать для него условие равенства 2,5.

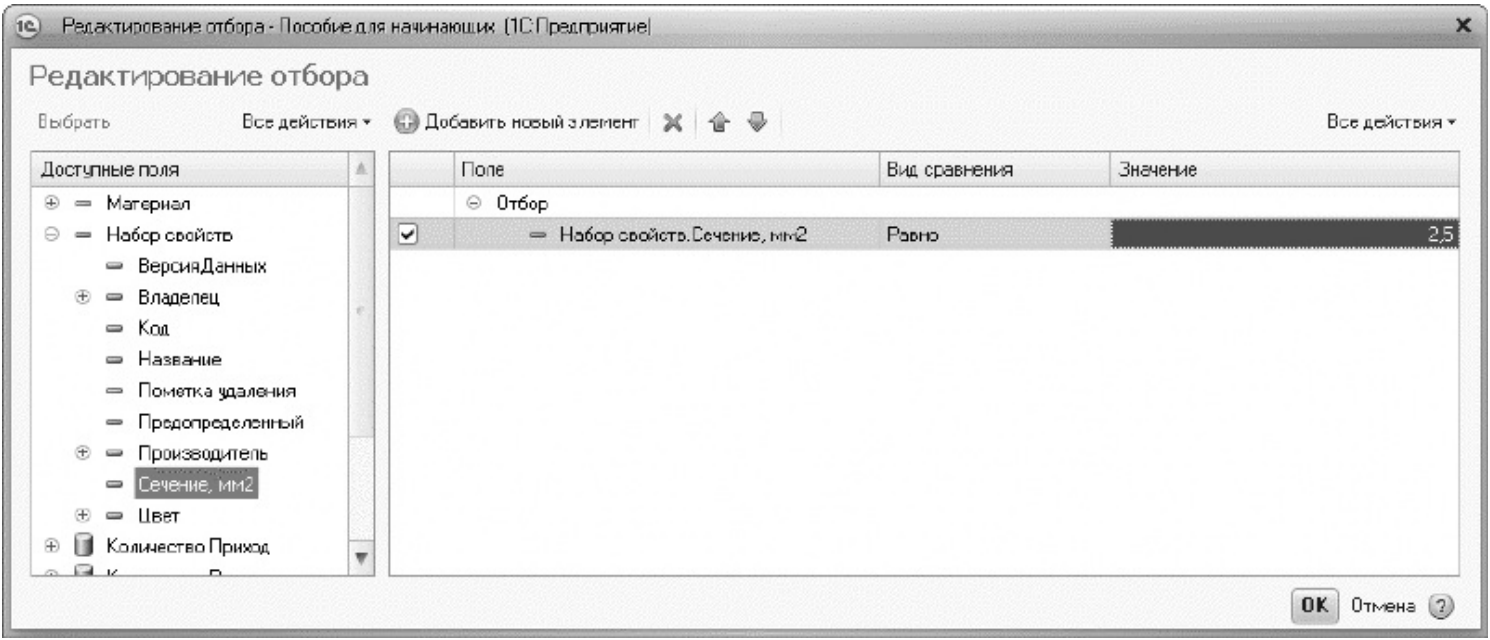


Рис. 15.54. Создание отбора

Нажмем *ОК*. В окне отчета нажмем *Сформировать* и получим следующий результат (рис. 15.55).

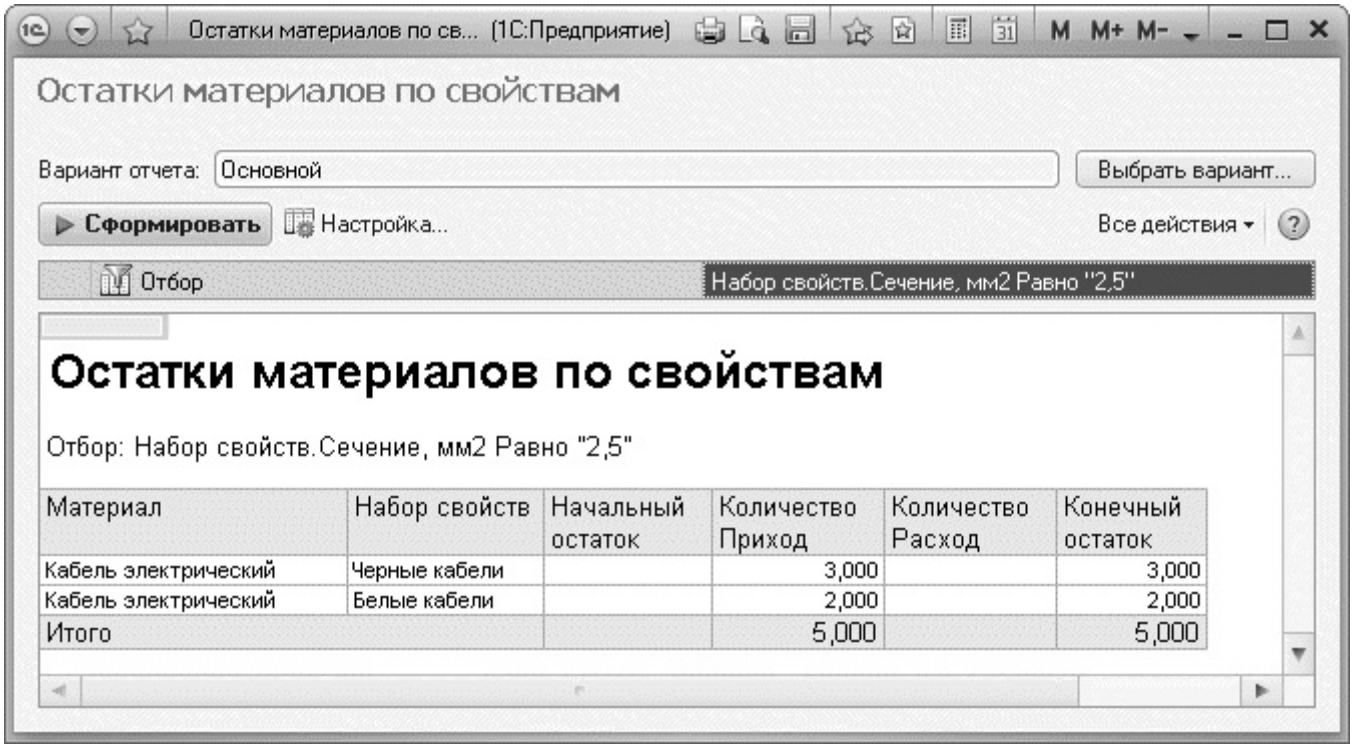


Рис. 15.55. Результат отчета

Затем посмотрим, какие у нас есть материалы черного цвета. Для этого в поле

настройки *Отбор* еще раз нажмем кнопку выбора ... и удалим прежний отбор кнопкой *Удалить* над списком условий отбора.

Затем двойным щелчком мыши выберем из списка доступных полей поле *Цвет*. Затем в поле *Значение* нажмем кнопку выбора ... и выберем из списка дополнительных свойств номенклатуры значение *Черный* (рис. 15.56).

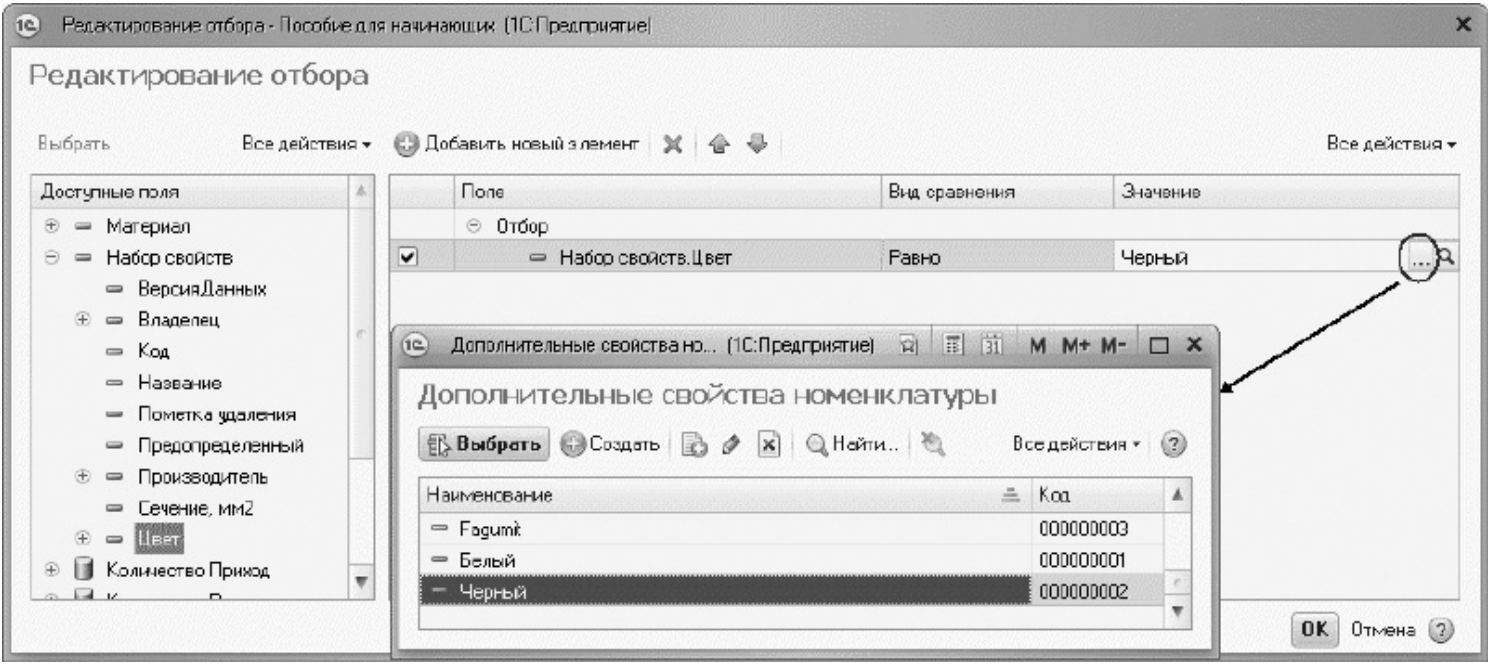


Рис. 15.56. Создание отбора

Нажмем *ОК*. В окне отчета нажмем *Сформировать* и получим следующий результат (рис. 15.57).

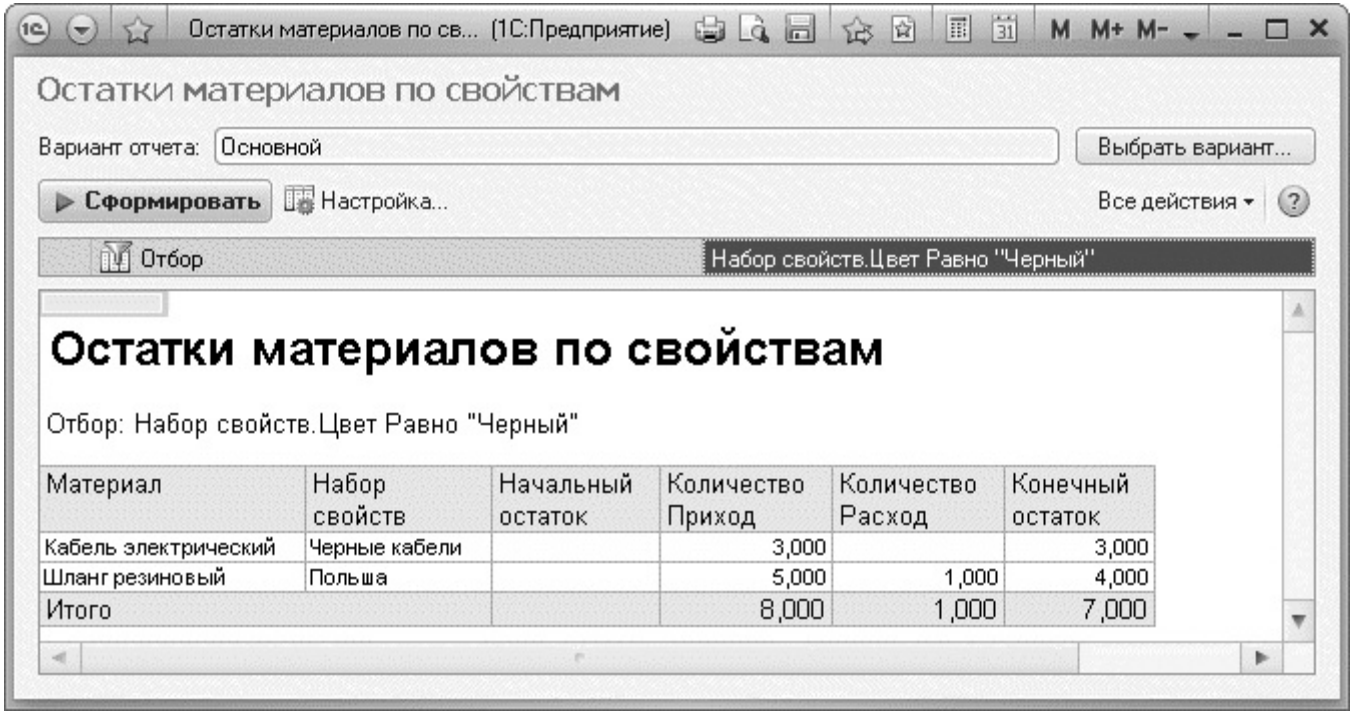


Рис. 15.57. Результат отчета

И в заключение, чтобы убедиться в правильности работы отчета, посмотрим, сколько у нас резиновых шлангов черного цвета.

В поле настройки *Отбор* еще раз нажмем кнопку выбора ... и добавим еще один элемент отбора. Для этого двойным щелчком мыши выберем из списка доступных полей поле *Материал*.

Затем в поле *Значение* нажмем кнопку выбора ... и выберем из списка номенклатуры значение *Шланг резиновый* (рис. 15.58).

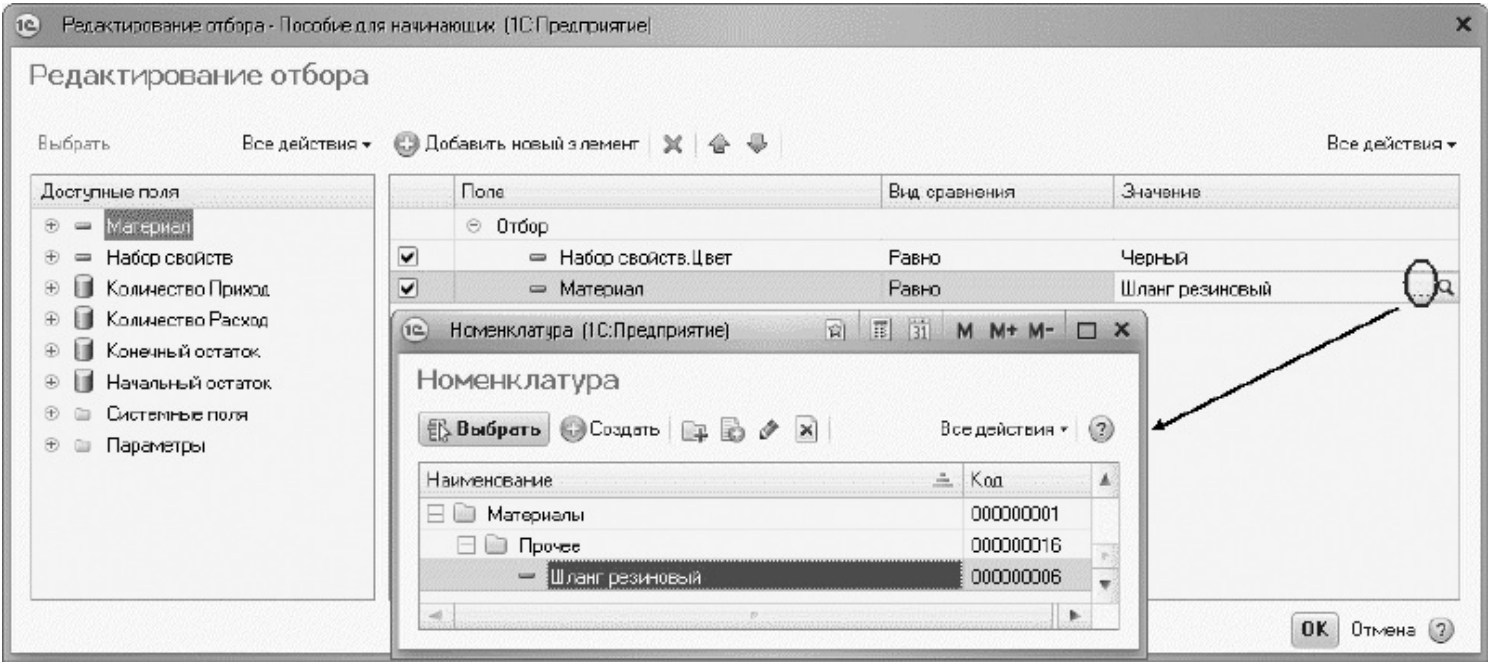


Рис. 15.58. Создание отбора

Нажмем *ОК*.

В окне отчета нажмем *Сформировать* и получим следующий результат (рис. 15.59).

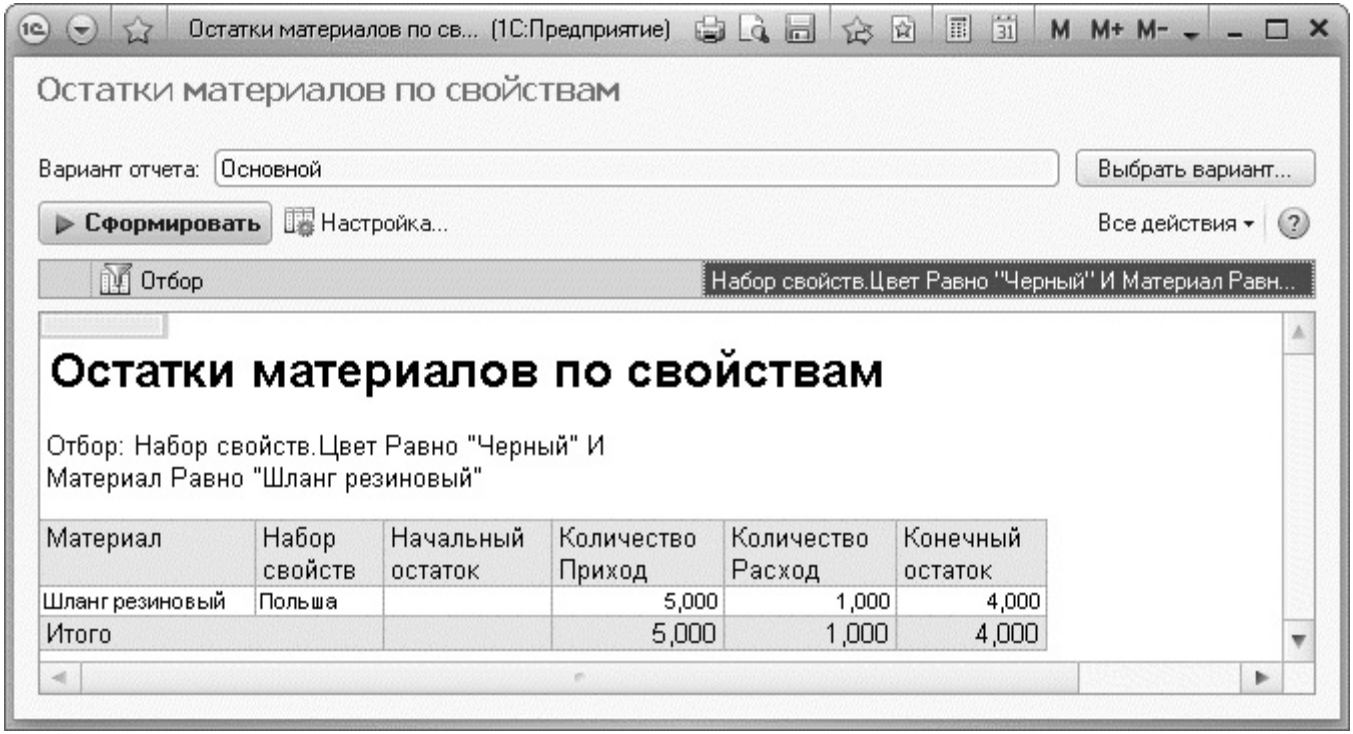


Рис. 15.59. Результат отчета

Таким образом, мы убедились в том, что при использовании данной логической

схемы мы имеем теперь возможность вести учет материалов в произвольном количестве разрезов свойств и их значений.

Следует заметить, что пример, рассмотренный нами в этой главе, не является законченным решением для данной конфигурации. Мы лишь продемонстрировали возможность ведения такого учета. Для того чтобы наша конфигурация могла полноценно использовать свойства материалов, необходимо внести соответствующие изменения в остальные регистры, документы и некоторые отчеты.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «План видов характеристик».*
- *В чем принципиальное отличие плана вида характеристик от справочника.*
- *Что такое тип значения характеристик.*
- *Зачем нужны дополнительные значения характеристик.*
- *Как, используя план видов характеристик, организовать учет по переменному количеству характеристик.*
- *Как создать план видов характеристик.*

- *Что такое связь по параметрам выбора.*
- *Как задать синоним стандартного реквизита.*
- *Как изменить заголовок формы.*
- *Как скрывать элементы формы с подчиненной информацией при ее создании.*
- *Как описать характеристики в схеме компоновки данных.*
- *Как использовать характеристики при выполнении отчета.*

Занятие 16 (1:50). Бухгалтерский учет

Продолжительность

Ориентировочная продолжительность занятия – 1 час 50 минут.

На этом занятии мы проиллюстрируем возможность ведения бухгалтерского учета средствами «1С:Предприятия». В рамках этого занятия мы не будем объяснять и рассматривать основы бухгалтерского учета. Поэтому если у вас нет знаний бухгалтерии, то, конечно, лучше сначала прочитать какую-нибудь популярную литературу о том, как вообще устроен бухгалтерский учет в нашей стране.

Для организации бухгалтерского учета мы используем уже знакомый нам план видов характеристик и два новых объекта конфигурации – *План счетов* и *Регистр бухгалтерии*.

Регистр бухгалтерии будет использоваться нами для накопления данных о совершенных хозяйственных операциях.

С помощью плана счетов мы будем описывать счета, в разрезе которых ведется учет, а план видов характеристик будет служить для описания объектов аналитического учета, в разрезе которых должен вестись учет на

счетах.

Сразу оговоримся, что план счетов, который мы будем использовать в нашей учебной конфигурации, очень сильно упрощен. Он содержит всего несколько условных счетов, которые, однако, позволят нам познакомиться с основными методами организации бухгалтерского учета средствами «1С:Предприятия».

План видов характеристик в бухгалтерском учете

Объект конфигурации *План видов характеристик* был подробно рассмотрен нами на предыдущем занятии (см. «Что такое План видов характеристик»), поэтому сейчас мы проиллюстрируем только использование этого объекта в контексте бухгалтерского учета.

Бухгалтерский учет, как правило, подразумевает ведение аналитического учета на большинстве счетов. Для обозначения разрезов аналитического учета мы будем использовать термин *виды субконто*. То есть на каждом счете учет может вестись в разрезе нескольких видов субконто.

А для обозначения конкретных объектов аналитического учета мы будем использовать термин *субконто*.

Например, на *41* счете (*Товары*) учет ведется обычно в разрезе

Номенклатуры и *Складов*, которые являются видами субконто. А вот конкретная номенклатура *Паста шоколадная* и конкретный склад *Основной*, указанные для некоторой проводки по *41* счету, – это субконто.

Так вот, частным случаем использования плана видов характеристик является применение его для описания видов субконто. То есть все разрезы аналитического учета описываются в соответствующем плане видов характеристик, и там же задаются типы значений, которые могут принимать те или иные субконто.

Добавление плана видов характеристик

В режиме «Конфигуратор»

Приступим к созданию плана видов характеристик, который будет содержать описания разрезов аналитического учета – видов субконто.

Откроем конфигуратор и добавим новый объект конфигурации *План видов характеристик*. Зададим его имя – *ВидыСубконто*. На закладке *Подсистемы* укажем, что план счетов будет отображаться в подсистеме *Бухгалтерия*.

Поскольку нам понадобится некий вспомогательный справочник, в котором пользователи будут осуществлять «свободное творчество» по созданию

значений новых объектов аналитического учета, добавим объект конфигурации Справочник и назовем его *Субконто*.

Затем на закладке *Владельцы* укажем, что этот справочник будет подчинен плану видов характеристик *ВидыСубконто*.

Для этого на закладке *Владельцы* нажмем кнопку *Добавить* и выберем в качестве владельца справочника план видов характеристик *ВидыСубконто* (рис. 16.1).

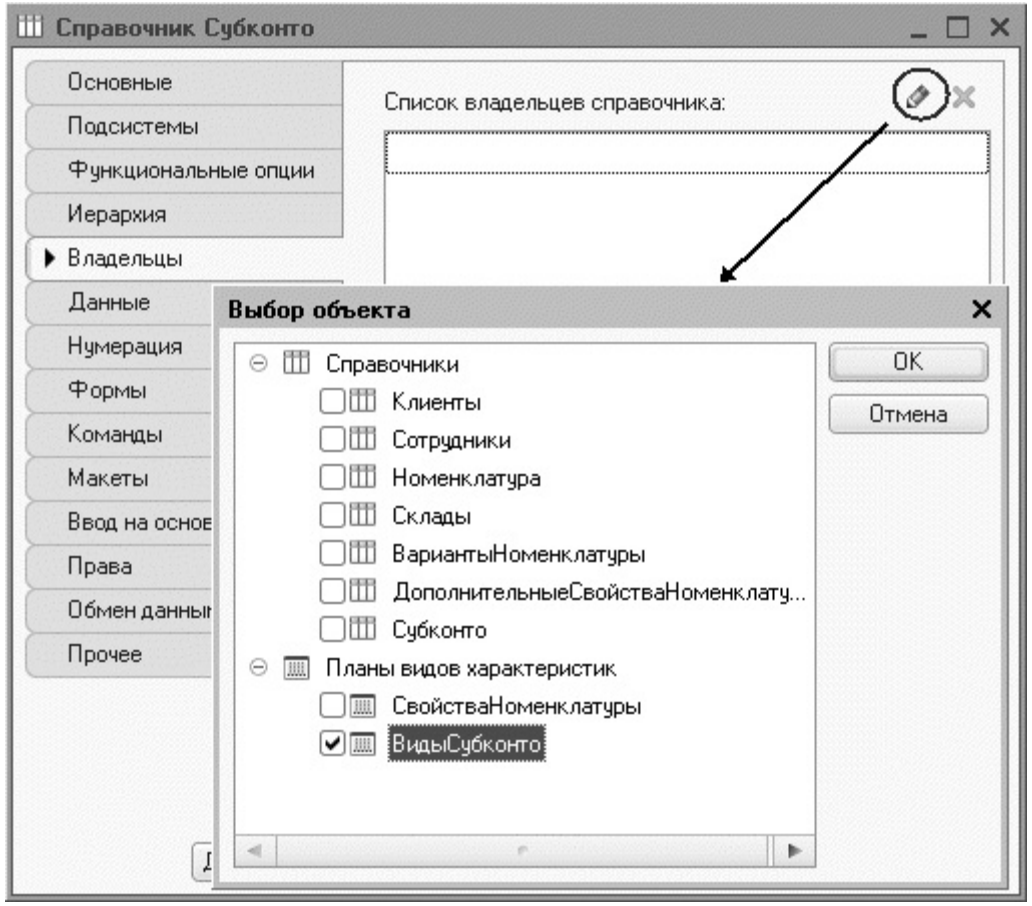


Рис. 16.1. Окно редактирования справочника «Субконто»

Закроем окно редактирования справочника и вернемся к нашему плану видов характеристик.

На закладке *Основные* установим свойство *Тип значения характеристик*. Нажмем кнопку выбора и зададим составной тип данных следующим образом (рис. 16.2):

- *СправочникСсылка.Клиенты,*
- *СправочникСсылка.Номенклатура,*
- *СправочникСсылка.Субконто.*

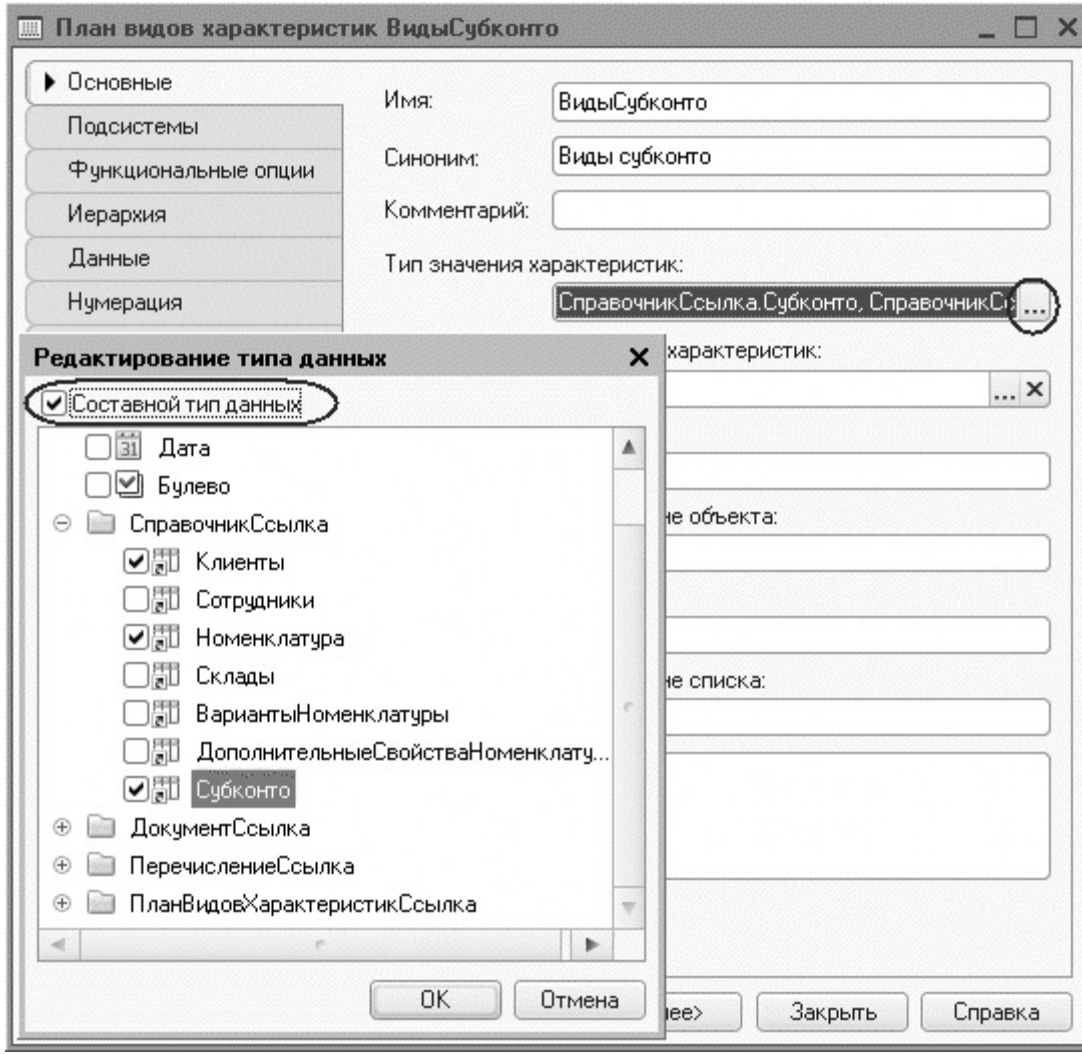


Рис. 16.2. Выбор типа значения характеристик плана вида характеристик

Бухгалтерия нашего ООО «На все руки мастер» ведет учет движения денежных средств только в разрезе материалов и клиентов, но не исключено, что в дальнейшем понадобится дополнительная аналитика (поэтому мы и используем справочник *Субконто*).

Обратите внимание, что тот справочник, который будет использован в качестве дополнительных значений характеристик, тоже должен входить в составной тип данных типа значений характеристик, иначе конфигуратор выдаст сообщение об ошибке.

Затем укажем, что дополнительные значения характеристик будут находиться в справочнике *Субконто*.

После этого перейдем на закладку *Прочее* и, нажав кнопку *Предопределенные*, начнем ввод предопределенных значений плана видов характеристик (рис. 16.3). То есть тех видов аналитического учета, в разрезе которых мы будем вести учет.

Рис. 16.3. Предопределенные виды характеристик

Нажимая кнопку *Добавить*, создадим предопределенный вид субконто *Материалы* с кодом *000000001* и типом *СправочникСсылка.Номенклатура.*

Затем создадим вид субконто *Клиенты* с кодом *000000002* и типом *СправочникСсылка.Клиенты*.

На этом создание видов субконто завершено, и мы можем перейти к знакомству со следующим объектом конфигурации, который будет использован нами, – *План счетов*.

Что такое «План счетов»

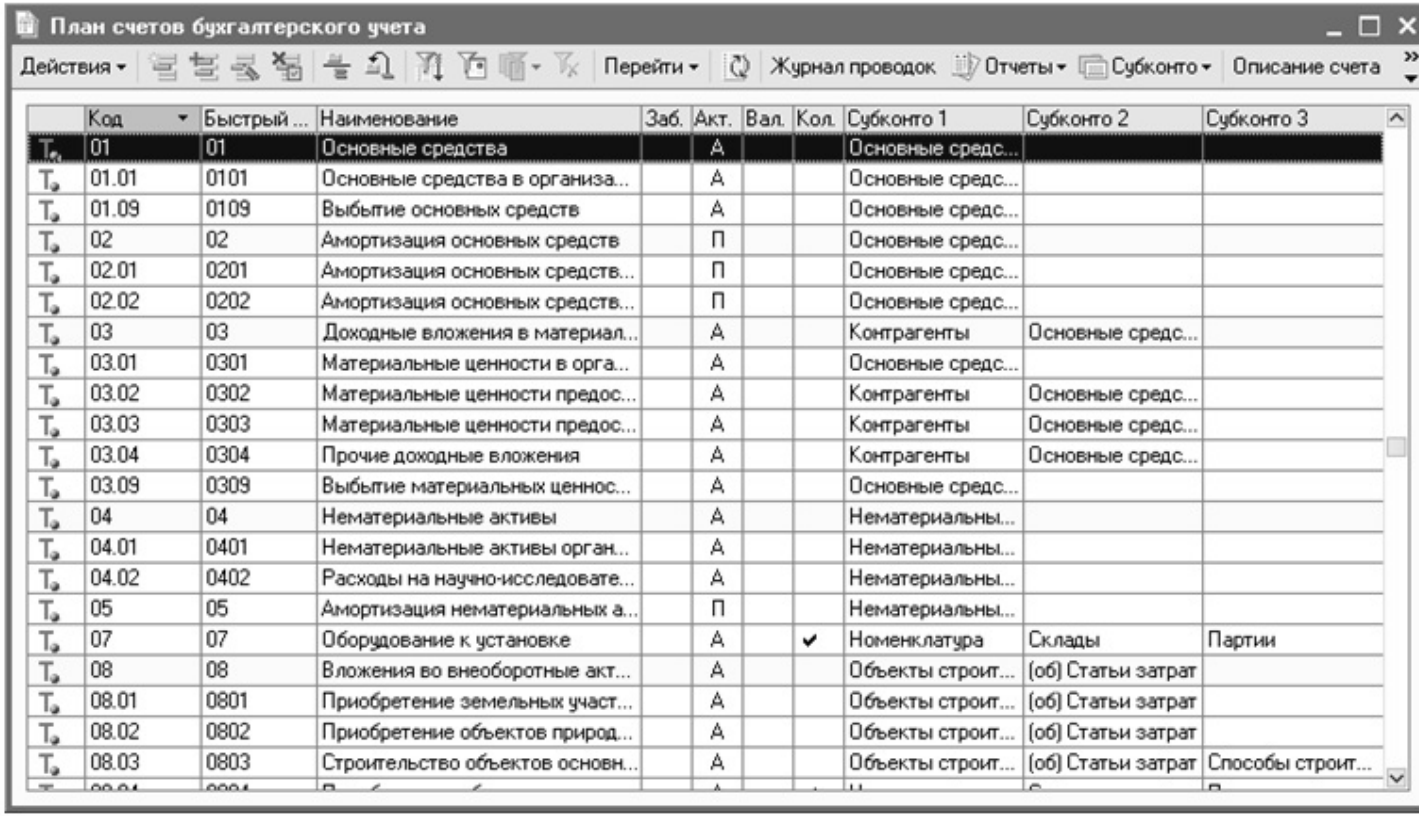
Объект конфигурации *План счетов* предназначен для описания структуры хранения информации о совокупности синтетических счетов предприятия, которые созданы для группировки данных о его хозяйственной деятельности.

На основе объекта конфигурации *План счетов* платформа создает в базе данных таблицы, в которых будет храниться информация о том, какие счета и каким образом будет использовать предприятие.

Это может быть система бухгалтерских счетов, установленная государством, план управленческих счетов или произвольный набор счетов, используемых для анализа тех или иных видов деятельности предприятия.

План счетов в системе «1С:Предприятие» поддерживает иерархию субсчетов: к каждому счету первого уровня может быть открыто несколько субсчетов, которые, в свою очередь, могут иметь свои субсчета, и так далее.

Например, законодательно утвержденный план счетов для ведения бухучета в России имеет следующий вид (рис. 16.4).



	Код	Быстрый ...	Наименование	Заб.	Акт.	Вал.	Кол.	Субконто 1	Субконто 2	Субконто 3
Т	01	01	Основные средства		А			Основные средс...		
Т	01.01	0101	Основные средства в организа...		А			Основные средс...		
Т	01.09	0109	Выбытие основных средств		А			Основные средс...		
Т	02	02	Амортизация основных средств		П			Основные средс...		
Т	02.01	0201	Амортизация основных средств...		П			Основные средс...		
Т	02.02	0202	Амортизация основных средств...		П			Основные средс...		
Т	03	03	Доходные вложения в материал...		А			Контрагенты	Основные средс...	
Т	03.01	0301	Материальные ценности в орга...		А			Основные средс...		
Т	03.02	0302	Материальные ценности предос...		А			Контрагенты	Основные средс...	
Т	03.03	0303	Материальные ценности предос...		А			Контрагенты	Основные средс...	
Т	03.04	0304	Прочие доходные вложения		А			Контрагенты	Основные средс...	
Т	03.09	0309	Выбытие материальных ценнос...		А			Основные средс...		
Т	04	04	Нематериальные активы		А			Нематериальны...		
Т	04.01	0401	Нематериальные активы орган...		А			Нематериальны...		
Т	04.02	0402	Расходы на научно-исследовате...		А			Нематериальны...		
Т	05	05	Амортизация нематериальных а...		П			Нематериальны...		
Т	07	07	Оборудование к установке		А		✔	Номенклатура	Склады	Партии
Т	08	08	Вложения во внеоборотные акт...		А			Объекты строит...	(об) Статьи затрат	
Т	08.01	0801	Приобретение земельных участ...		А			Объекты строит...	(об) Статьи затрат	
Т	08.02	0802	Приобретение объектов природ...		А			Объекты строит...	(об) Статьи затрат	
Т	08.03	0803	Строительство объектов основн...		А			Объекты строит...	(об) Статьи затрат	Способы строит...

Рис. 16.4. Российский план счетов

По любому счету или субсчету может вестись аналитический учет в разрезе субконто, описанных в плане видов характеристик. Связь между планом счетов и планом видов характеристик задается разработчиком на этапе конфигурирования.

Для описания используемых субконто система создает в плане счетов специальную табличную часть *ВидыСубконто*, которая не видна в конфигураторе (но доступна средствами встроенного языка).

Для каждого счета есть возможность задать несколько видов учета (например, количественный и валютный). Виды учета задаются при помощи подчиненных объектов конфигурации *признак учета*.

Также существует возможность определить несколько видов учета субконто (например, суммовой, валютный или количественный). Виды учета субконто задаются при помощи подчиненных объектов конфигурации *признак учета субконто*.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с планами счетов, можно прочитать в разделе «Краткий справочник разработчика. Планы счетов»*.*

Помимо всего вышеперечисленного каждый счет может иметь набор свойств, которые задаются в качестве реквизитов объекта конфигурации План счетов. Они позволяют определять уникальные свойства элементов плана счетов

(например, реквизит *ЗапретитьИспользоватьВПроводках*).

Добавление плана счетов

В режиме «Конфигуратор»

Приступим к созданию плана счетов ООО «На все руки мастер».

Как мы говорили в начале этого занятия, бухгалтерский учет в нашем ООО «На все руки мастер» сильно упрощен. Поэтому план счетов, по которому работает бухгалтерия, содержит всего четыре счета:

- *Товары*,
- *РасчетыСПоставщиками*,
- *Капитал,*
- *Дебиторская задолженность*.

Добавим новый объект конфигурации *План счетов*. Присвоим ему имя – *Основной*. Свойство *Представление списка* зададим как *Основной план счетов*. На закладке *Подсистемы* укажем, что план счетов будет отображаться в подсистеме *Бухгалтерия*. На закладке *Данные* выделим группу реквизитов *Признаки учета* и, нажав кнопку *Добавить*, создадим признак учета *Количественный* (рис. 16.5).

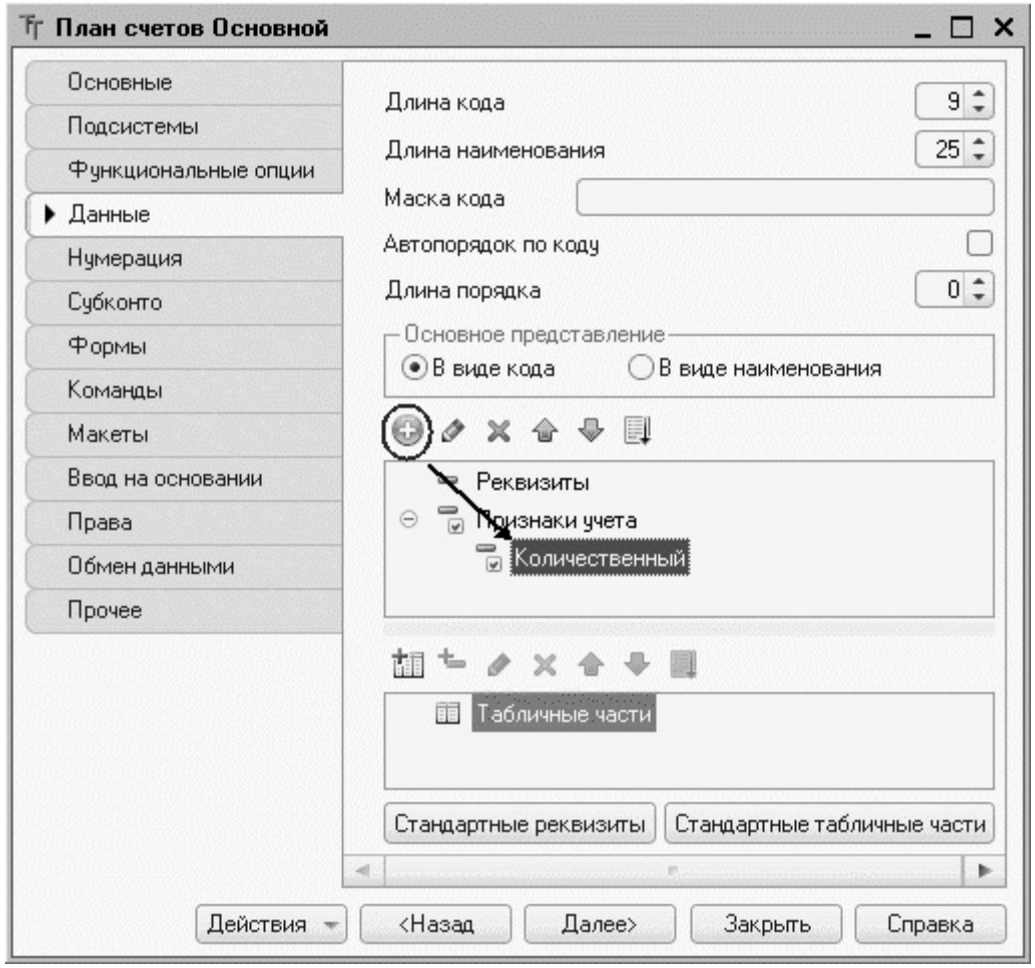


Рис. 16.5. Создание реквизита плана счетов в группе «Признаки учета»

Перейдем на закладку *Субконто* и укажем, что виды субконто для этого плана

счетов будут находиться в плане видов характеристик *ВидыСубконто*.

Максимальное количество субконто на счете установим равным двум.

Также создадим признак учета субконто *Количественный* (рис. 16.6).

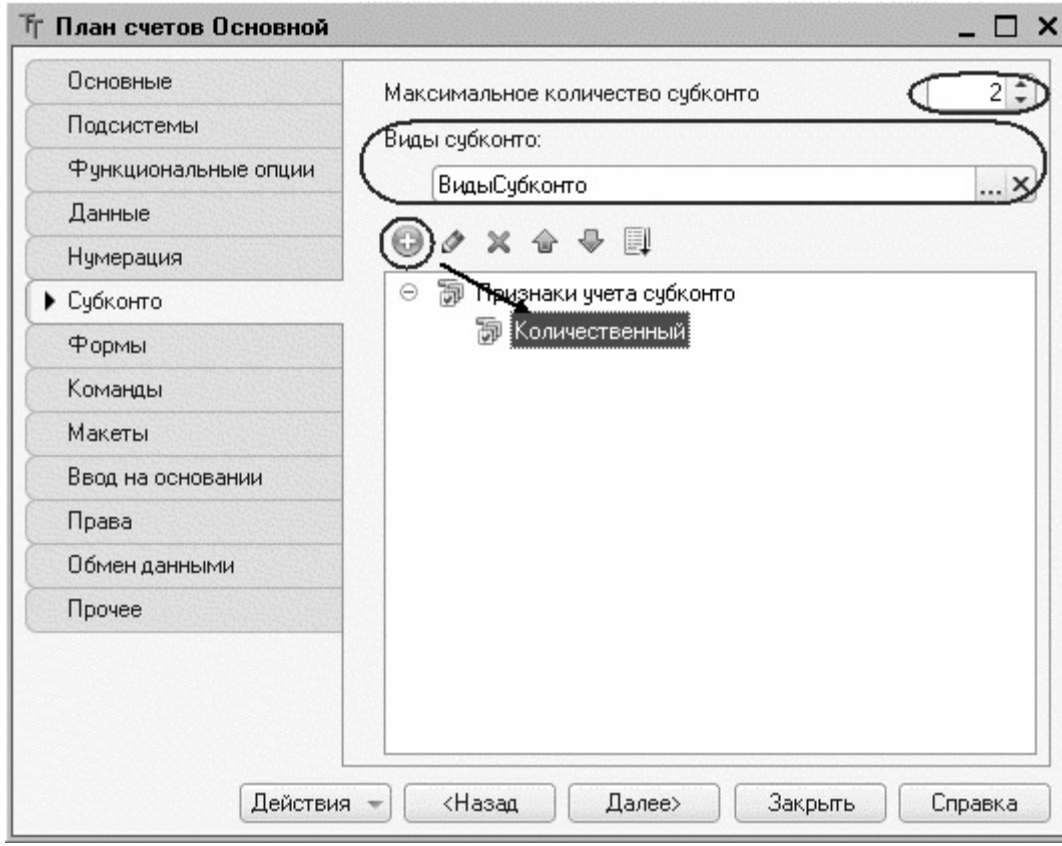


Рис. 16.6. Установка свойств «Субконто для плана счетов»

Затем откроем закладку *Прочее*. Нажмем кнопку *Предопределенные* и создадим четыре предопределенных счета (при создании каждого счета, перед тем как нажать кнопку *Добавить*, нужно выделить корень структуры счетов – строку *Счета*):

- *Товары*, код *41*, активный, с количественным учетом в разрезе материалов (рис. 16.7).

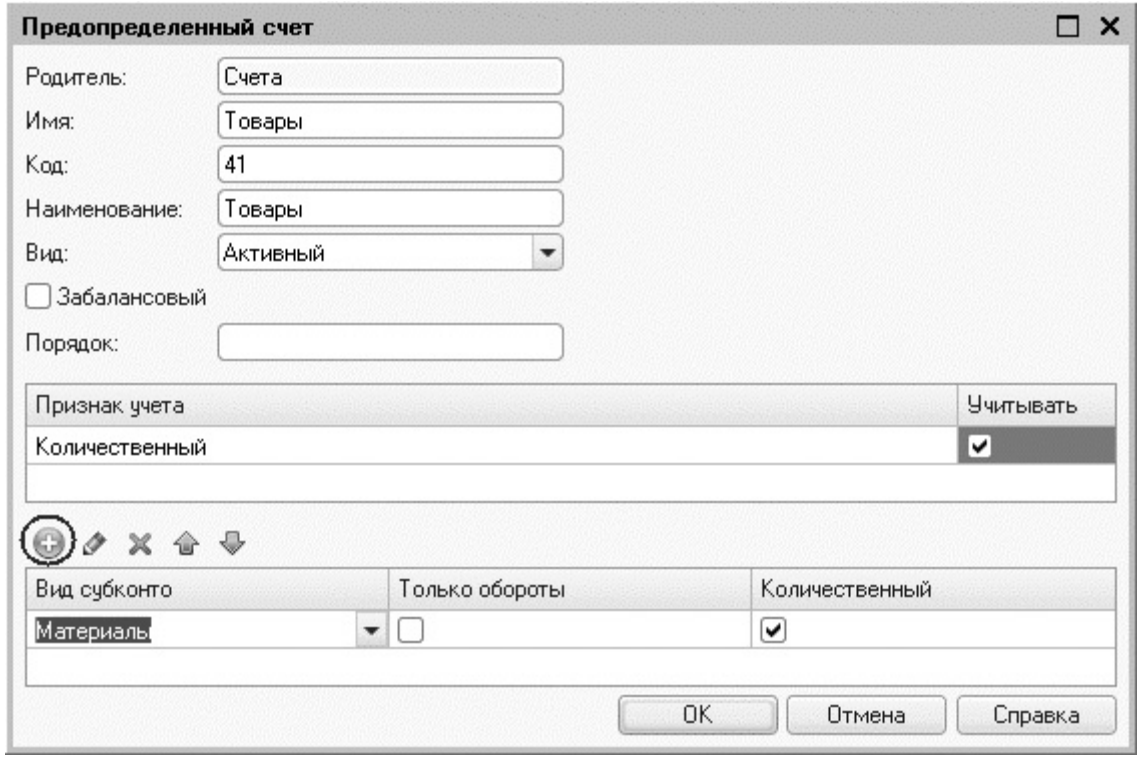


Рис. 16.7. Предопределенный счет «Товары»

- *РасчетыСПоставщиками*, код *60*, активный/пассивный (рис. 16.8).

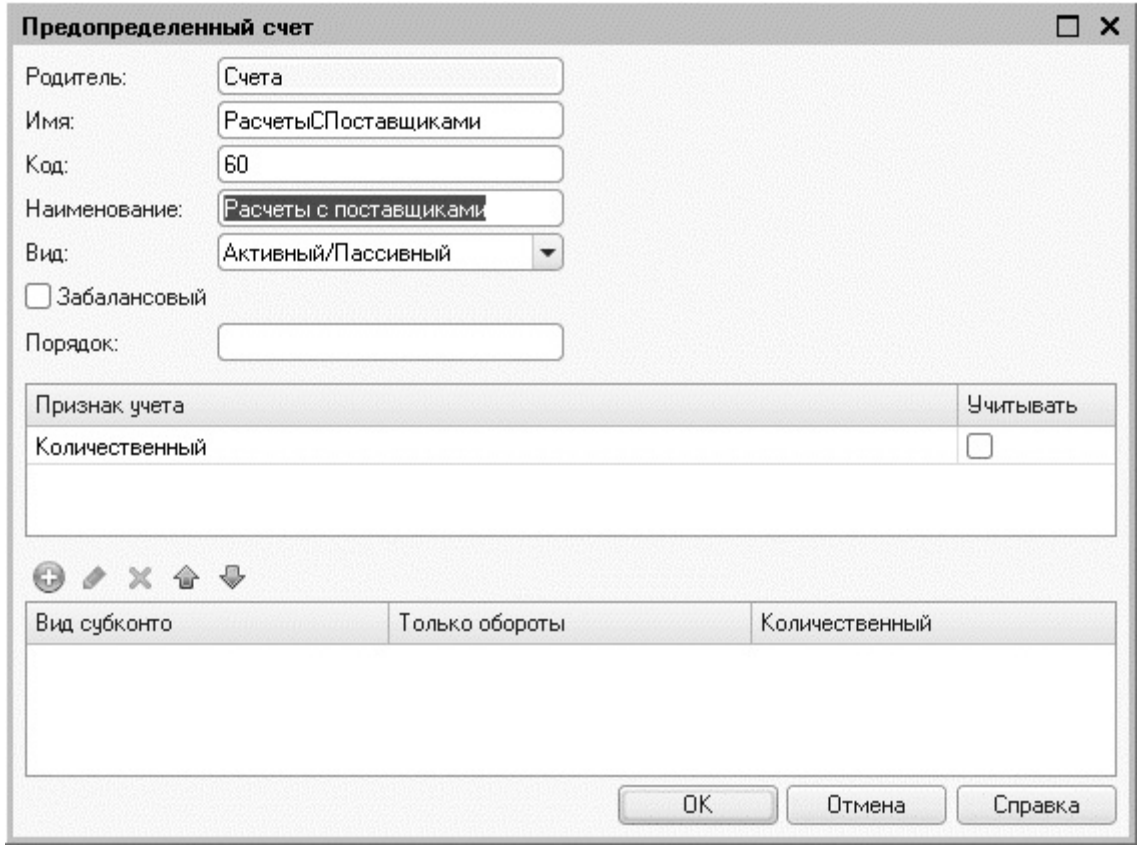


Рис. 16.8. Предопределенный счет «РасчетыСПоставщиками»

- *ДебиторскаяЗадолженность*, код *62*, активный/пассивный, в разрезе клиентов (рис. 16.9).

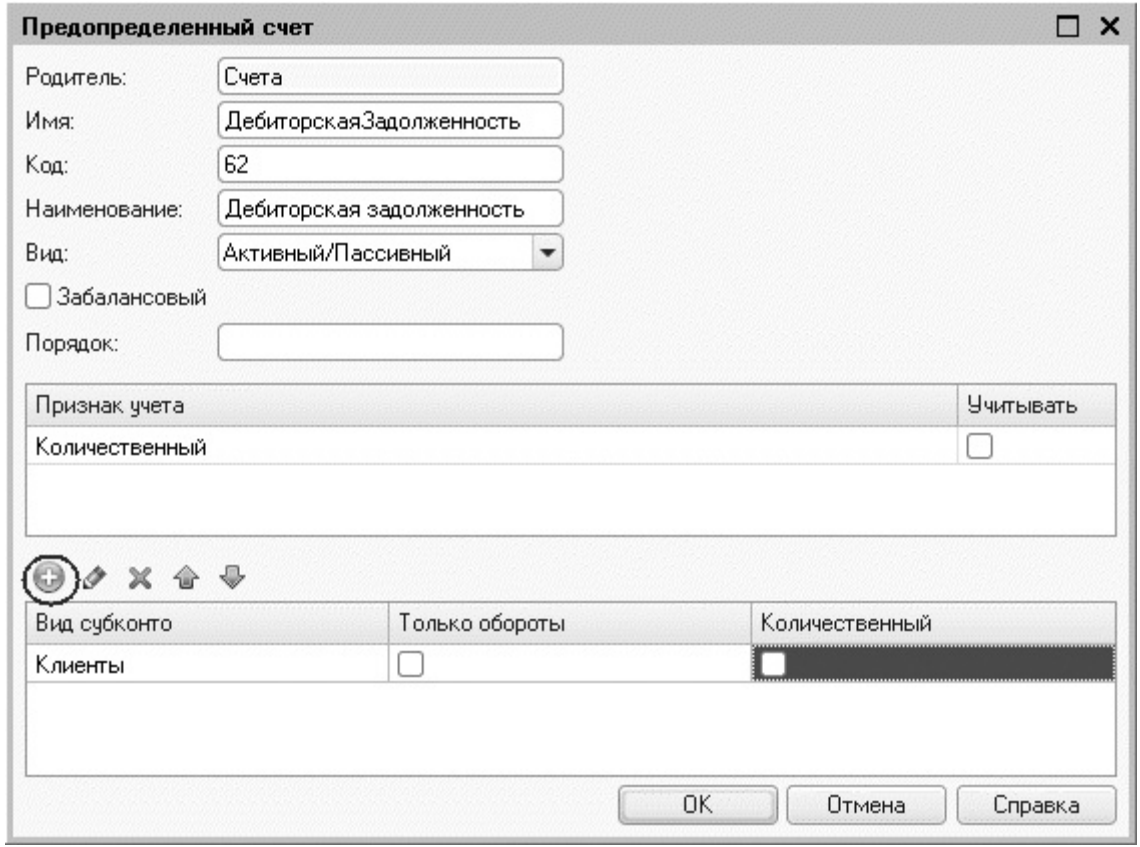


Рис. 16.9. Предопределенный счет «ДебиторскаяЗадолженность»

- *Капитал*, код *90*, активный/пассивный (рис. 16.10).

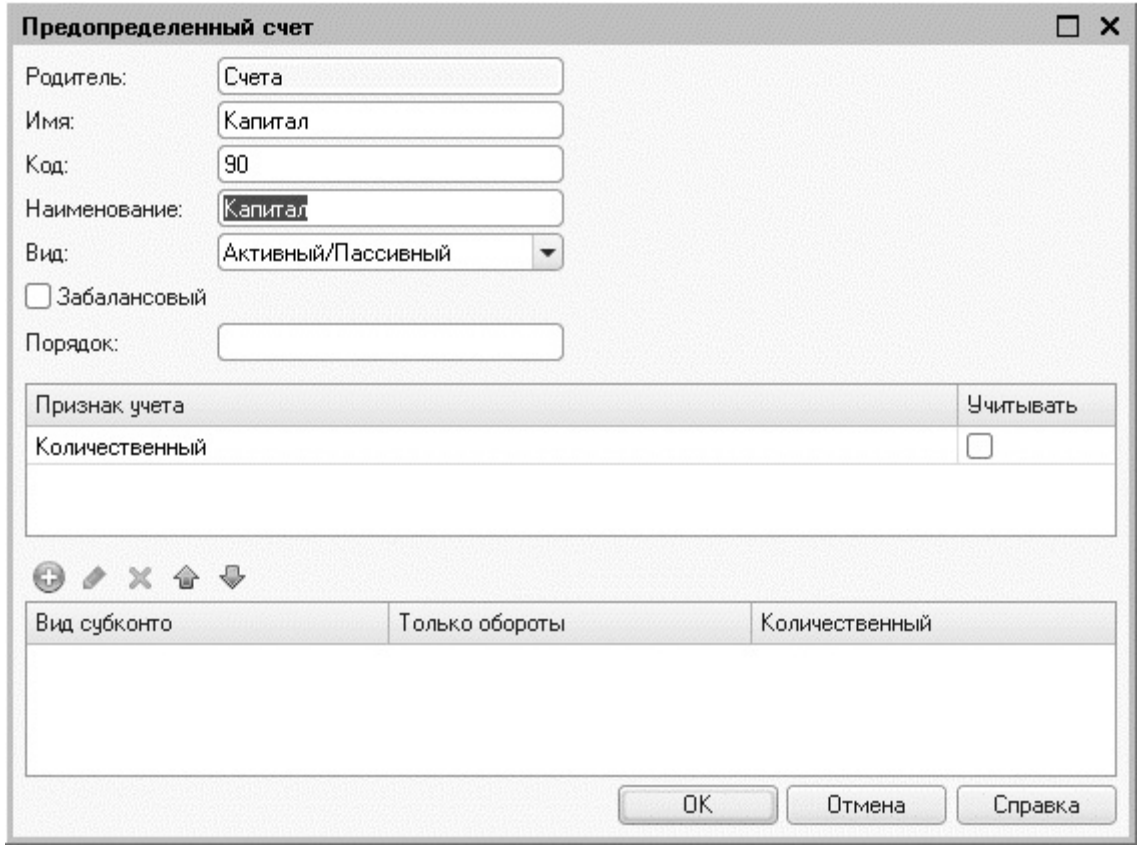


Рис. 16.10. Предопределенный счет «Капитал»

В результате план счетов нашего ООО «На все руки мастер» будет выглядеть следующим образом (рис. 16.11).

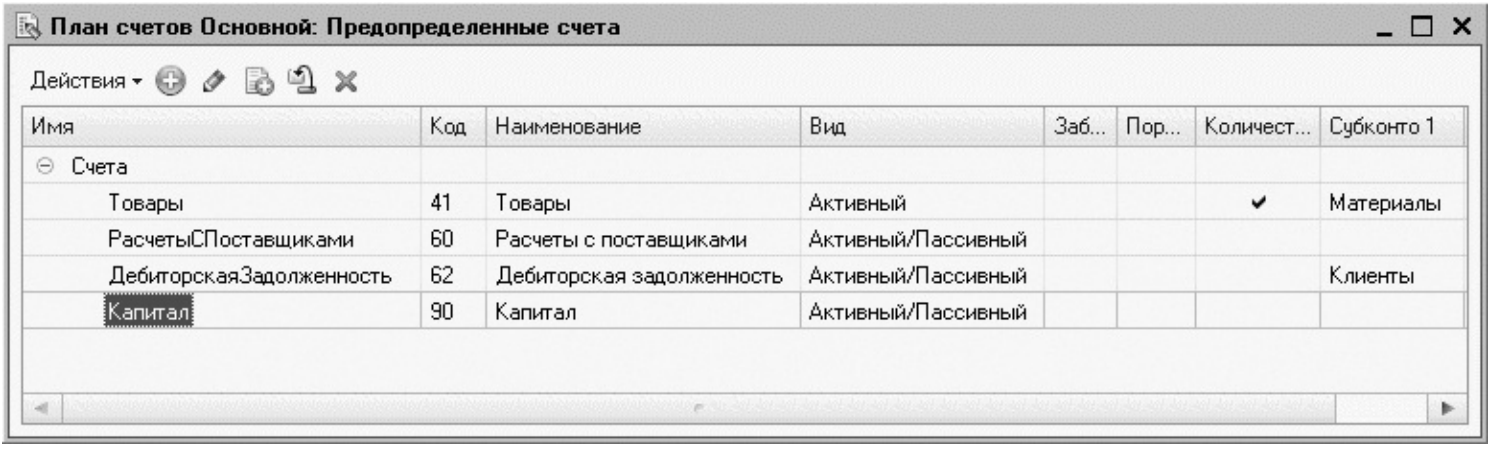


Рис. 16.11. План счетов «Основной»

Теперь мы можем перейти к знакомству с последним объектом конфигурации, который понадобится нам для организации бухгалтерского учета, – *Регистром бухгалтерии*.

Узнай больше!

Для плана счетов можно установить свойство «Автопорядок по коду». Это свойство используется для того, чтобы указать системе, что упорядочивание по полю «Порядок» должно всегда подставляться в тех случаях, когда пользователь или разработчик выбирает упорядочивание по коду. Его нужно использовать прежде всего тогда, когда с точки зрения пользователя нужно упорядочивать план счетов по коду с учетом

разделителей кода счета. Например, если счета 10.11 и 10.2 упорядочивать по коду счета, то счета будут располагаться так:

10.11

10.2

Это правильно с точки зрения сортировки строк, но не соответствует логическому смыслу кодов.

Но если заданы значения поля «Порядок» 10.11 и 10. 2 (перед 2 - пробел) и установлено свойство «Автопорядок по коду», то при выборе упорядочивания по коду пользователь будет фактически получать порядок, учитывающий разделители:

10.2

10.11

Если свойство не устанавливать, то нужно будет в явном виде выбирать упорядочивание по полю «Порядок».

Что такое регистр бухгалтерии

Объект конфигурации *Регистр бухгалтерии* предназначен для описания структуры накопления данных, учет которых ведется исходя из некоторого плана счетов. На основе объекта конфигурации *Регистр бухгалтерии* платформа создает в базе данных таблицу, в которой будут накапливаться данные о хозяйственных операциях, отображаемых в бухгалтерском учете.

По своему виду регистр бухгалтерии напоминает регистр накопления – он также имеет ресурсы, может иметь измерения и реквизиты.

Измерения позволяют разделять ведение учета (например, используя измерение *Организация*, можно вести учет в разрезе нескольких юридических лиц).

Реквизиты служат признаком, по которому одни записи регистра можно отделить от других (например, в качестве реквизита может использоваться номер журнала, что позволит отбирать проводки, имеющие одинаковый смысл).

Значительное отличие от регистра накопления заключается в том, что регистр бухгалтерии имеет жесткую связь с используемым планом счетов. Поэтому каждая запись регистра бухгалтерии содержит дополнительные поля, определяемые настройкой используемого плана счетов.

Например, запись регистра может содержать дополнительные поля для указания корреспондирующих счетов, сумм, объектов аналитического учета (субконто), количества, вида валюты и т. д.

Кроме этого, отличительной чертой регистра бухгалтерии является возможность поддержки механизма двойной записи, при которой каждая запись регистра содержит обязательные поля для указания счета дебета и счета кредита.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с регистрами бухгалтерии, можно прочитать в разделе [«Краткий справочник разработчика. Регистры бухгалтерии»](#)*.*

Добавление регистра бухгалтерии

В режиме «Конфигуратор»

Создадим новый объект конфигурации «Регистр бухгалтерии». Зададим его имя – *Управленческий*. Свойство *Расширенное представление списка* зададим как *Движения в регистре Управленческий*. Укажем, что с ним будет связан план счетов *Основной*. Установим флажок *Корреспонденция* (рис. 16.12).

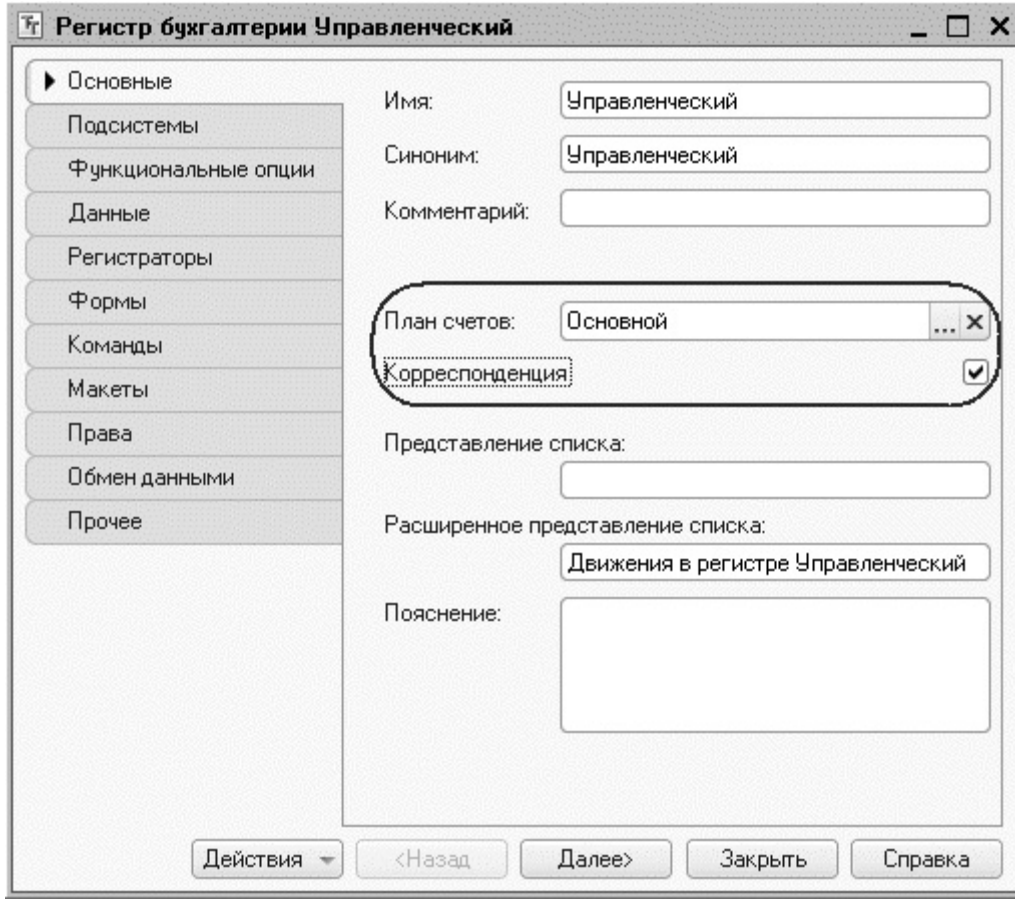


Рис. 16.12. Основные свойства регистра бухгалтерии

Флажок *Корреспонденция* будет говорить о том, что создаваемый нами регистр поддерживает корреспонденции. Это означает, что каждая запись регистра

имеет дебетовую и кредитовую часть, что позволит нам получать информацию не только об остатках и оборотах по счетам, но и о корреспонденциях между счетами.

Регистры, не поддерживающие корреспонденцию, применяются тогда, когда не нужно использовать принцип двойной записи, регламентированный в бухгалтерском учете, и, соответственно, контролировать баланс хозяйственных средств и их источников.

Теперь перейдем на закладку *Данные* и создадим два ресурса:

- *Сумма*, длина *15*, точность *2*, *Балансовый*;
- *Количество*, длина *15*, точность *3*, *Небалансовый*, признак учета – *Количественный*, признак учета субконто – *Количественный* (рис. 16.13).

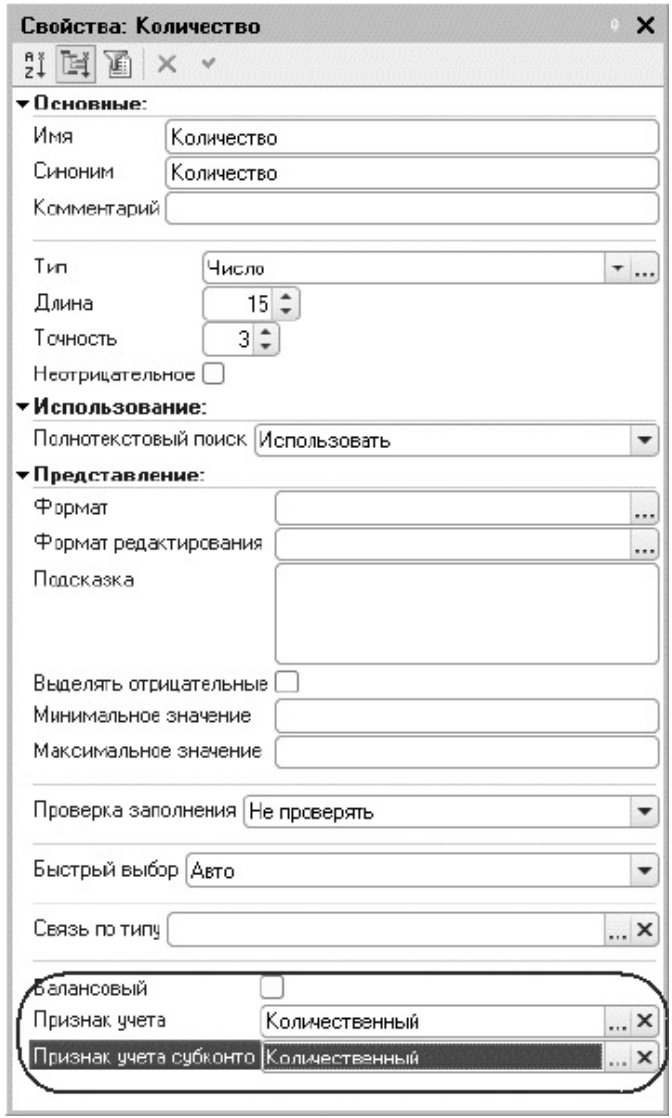

Свойства: Количество
Основные:
Имя
Количество
Синоним
Количество
Комментарий
Тип
Число
Длина
15
Точность
3
Неотрицательное
Использование:
Полнотекстовый поиск
Использовать
Представление:
Формат
Формат редактирования
Подсказка
Выделять отрицательные
Минимальное значение
Максимальное значение
Проверка заполнения
Не проверять
Быстрый выбор
Авто
Связь по типу
Балансовый
Признак учета
Количественный
Признак учета субконто
Количественный

Рис. 16.13. Свойства ресурса «Количество регистра бухгалтерии»

На этом создание нашего регистра бухгалтерии завершено.

Теперь настало время познакомиться с тем, каким образом используется созданный нами регистр бухгалтерии *Управленческий*.

Доработка приходной накладной

Что нас ждет дальше?

Сначала мы доработаем оба наши документа (*ПриходнаяНакладная* и *ОказаниеУслуги)* так, чтобы они «поставляли» данные не только для регистров накопления, но и для регистра бухгалтерии.

Затем мы создадим бухгалтерский отчет *Оборотно-сальдовая ведомость*, который будет показывать нам состояние товародвижения в ООО «На все руки мастер», основываясь на данных регистра бухгалтерии.

При проведении наши документы будут создавать следующие бухгалтерские проводки – записи в регистре бухгалтерии (таблица 16.1).

Таблица 16.1. Проводки, создаваемые документами

Документы	Проводки				
	Дебет		Кредит		Сумма
Приходная накладная	41	Товары	60	Расчеты с поставщиками	Стоимость
Оказание услуги	62	Дебиторская задолженность	90	Капитал	Выручка
	90	Капитал	41	Товары	Стоимость

Эти проводки мы отразим при создании движений документов в регистре бухгалтерии.

Итак, сначала изменим процедуру проведения документа *ПриходнаяНакладная*, а затем в режиме *1С:Предприятие* перепроведем все эти документы, чтобы отработал новый, измененный нами алгоритм проведения документа *Приходная накладная*.

В режиме «Конфигуратор»

В окне редактирования объекта конфигурации *Документ ПриходнаяНакладная* перейдем на закладку *Движения*. В списке регистров отметим, что документ будет создавать теперь движения и по регистру бухгалтерии *Управленческий*.

Перейдем на закладку *Прочее* и откроем модуль объекта. Откроем процедуру обработчика события *ОбработкаПроведения*. В самом конце цикла перед строкой *КонецЦикла* добавим строки кода, создающие движения регистра *Управленческий* (листинг 16.1).

Листинг 16.1. Движения документа «ПриходнаяНакладная» (фрагмент)

```
// Регистр Управленческий
Движение = Движения.Управленческий.Добавить();
Движение.СчетДт = ПланыСчетов.Основной.Товары;
Движение.СчетКт = ПланыСчетов.Основной.РасчетыСПоставщиками;
Движение.Период = Дата;
Движение.Сумма = ТекСтрокаМатериалы.Сумма;
Движение.КоличествоДт = ТекСтрокаМатериалы.Количество;
Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Материалы] =
ТекСтрокаМатериалы.Материал;
```

Перед началом цикла установим свойство *Записывать* набора записей регистра *Управленческий* в значение *Истина* для записи изменений регистра в базу данных.

В результате процедура *ОбработкаПроведения* будет выглядеть следующим образом (листинг 16.2).

Листинг 16.2. Движения документа «ПриходнаяНакладная»

```
Процедура ОбработкаПроведения(Отказ, Режим)

  Движения.ОстаткиМатериалов.Записывать = Истина;
  Движения.СтоимостьМатериалов.Записывать = Истина;
  Движения.Управленческий.Записывать = Истина;

  Для Каждого ТекСтрокаМатериалы Из Материалы Цикл

    // Регистр ОстаткиМатериалов Приход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.НаборСвойств = ТекСтрокаМатериалы.НаборСвойств;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаМатериалы.Количество;

    // Регистр Стоимость Материалов Приход
    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.Стоимость = ТекСтрокаМатериалы.Сумма;

    // Регистр Управленческий
    Движение = Движения.Управленческий.Добавить();
    Движение.СчетДт = ПланыСчетов.Основной.Товары;
    Движение.СчетКт = ПланыСчетов.Основной.РасчетыСПоставщиками;
    Движение.Период = Дата;
    Движение.Сумма = ТекСтрокаМатериалы.Сумма;
```

```
    Движение.КоличествоДт = ТекСтрокаМатериалы.Количество;
    Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Материалы] =
    ТекСтрокаМатериалы.Материал;

  КонецЦикла;

КонецПроцедуры
```

Прокомментируем этот код.

Как видите, движения для регистра бухгалтерии формируются таким же образом, как и для регистра накопления.

Но платформа (при создании таблицы хранения данных) добавила к созданным нами реквизитам регистра еще ряд полей, которые явились следствием использования плана счетов *Основной*.

Прежде всего, это поля *СчетДт* и *СчетКт*. В этих полях указываются счета, дебет и кредит которых затрагивает данная проводка.

Кроме этого, для измерений и ресурсов регистра, связанных с признаками учета, платформа создает пару полей для хранения значения каждого ресурса отдельно по дебету и отдельно по кредиту проводки – *КоличествоДт* и *КоличествоКт*. А также для счетов, по которым ведется учет в разрезе

субконто, платформа создает коллекции *СубконтоДт* и *СубконтоКт*.

Если обратиться к таблице 16.1, то при проведении приходной накладной счетом по дебету должен быть счет *Товары* (41), а счетом по кредиту – счет *РасчетыСПоставщиками* (60).

К счету мы обращаемся с помощью свойства глобального контекста *ПланыСчетов*. Оно предоставляет доступ ко всем планам счетов, созданным в конфигурации. Через точку от него мы указываем имя нужного нам плана счетов – *Основной*. А далее, тоже через точку, указываем имя предопределенного счета в этом плане счетов – *Товары*. Этот счет (и три других) мы создали в конфигураторе.

Так как количественный учет у нас ведется только для счета *Товары* (41), то поле регистра *КоличествоДт* заполняется количеством товара из табличной части документа. Поле регистра *КоличествоКт* не заполняется, так как по счету кредита проводки (*РасчетыСПоставщиками*) количественный учет не ведется.

Теперь рассмотрим последнюю строку цикла, в которой присваивается значение субконто дебета.

Дело в том, что количество субконто на счете дебета и на счете кредита в

каждой проводке будет различное, в зависимости от того, как определены счета в используемом плане счетов. Поэтому для каждой записи движения регистра бухгалтерии платформа хранит две коллекции значений: коллекцию субконто дебета и коллекцию субконто кредита. Каждая из них содержит ровно столько элементов, сколько видов субконто указано использовать для соответствующего счета (дебета или кредита) в плане счетов.

Обратиться к элементу коллекции можно, указав в квадратных скобках ссылку на соответствующий вид субконто (*ПланыВидовХарактеристик.ВидыСубконто.Материалы*).

Другой способ – в явном виде указать имя предопределенного вида субконто через точку от коллекции субконто дебета.

Другими словами, запись *Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Материа* равносильна записи *Движение.СубконтоДт.Материалы*.

Коллекция регистра *СубконтоКт* не заполняется, так как по счету кредита проводки (*РасчетыСПоставщиками*) учет в разрезе субконто у нас не ведется.

В заключение отредактируем командный интерфейс формы документа, чтобы в

панели навигации формы иметь возможность переходить к списку записей регистра *Управленческий*, связанному с документом.

Для этого откроем форму документа *ПриходнаяНакладная*. В левом верхнем окне перейдем на закладку *Командный интерфейс*. В разделе *Панель навигации* раскроем группу *Перейти* и установим видимость для команды открытия регистра бухгалтерии *Управленческий*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

Платформа предупредит нас, что регистр бухгалтерии *Управленческий* и справочник *Субконто* не включены ни в одну подсистему. Проигнорируем это сообщение.

Откроем документ *Приходная накладная № 1* и нажмем *Провести*.

Выполним команду перехода к регистру *Управленческий* и посмотрим, какие движения сформировал документ в регистре бухгалтерии (рис. 16.14, 16.15).

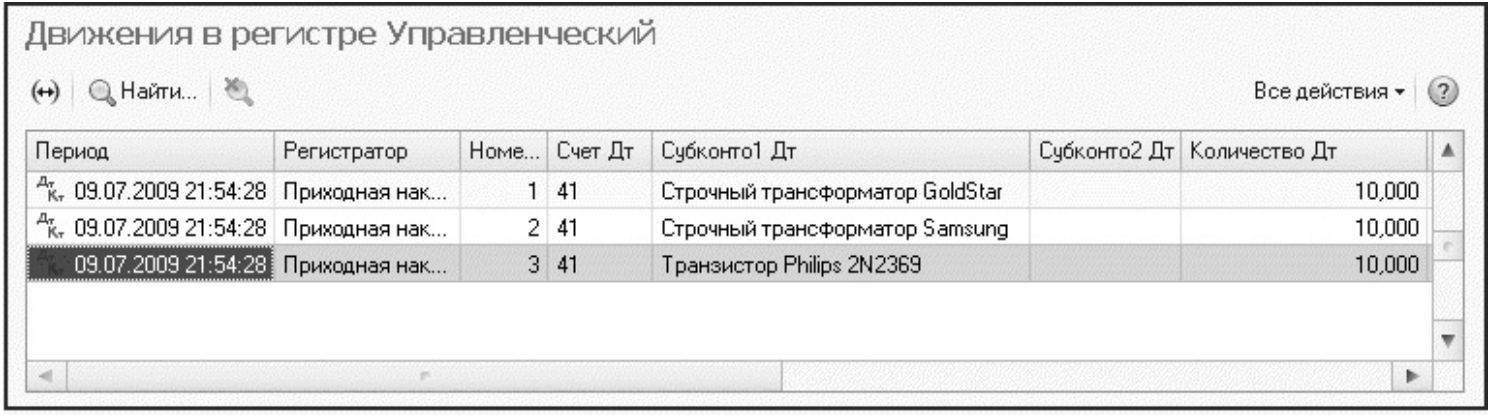


Рис. 16.14. Движения документа «Приходная накладная № 1» в регистре бухгалтерии «Управленческий»

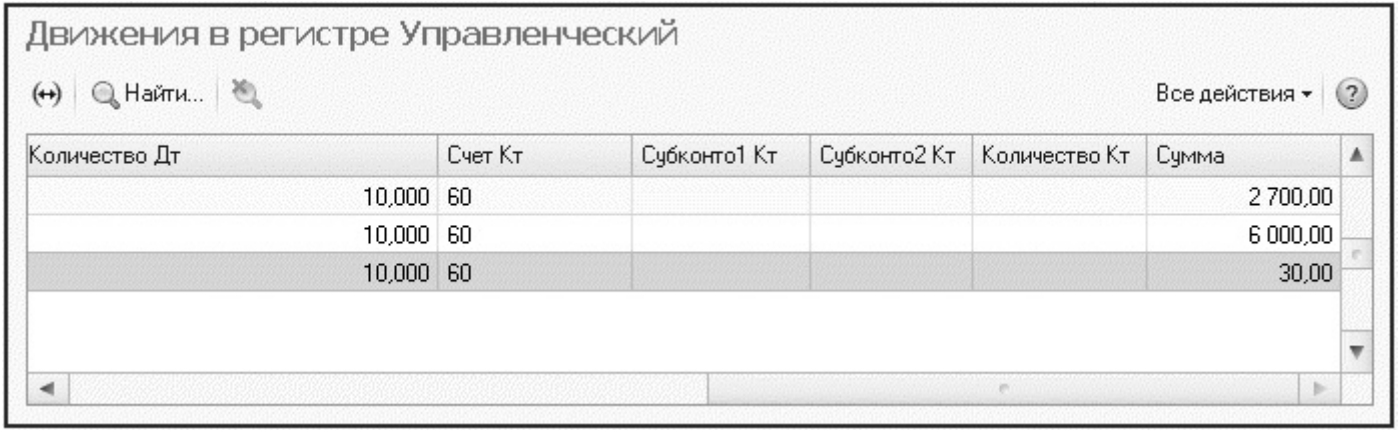


Рис. 16.15. Движения документа «Приходная накладная № 1» в регистре бухгалтерии «Управленческий»

Обратите внимание: поскольку на счете 60 (*РасчетыСПоставщиками*) отсутствует аналитика и ведется только суммовой учет, в записях движений

регистра *СубконтоКт1*, *СубконтоКт2* и *КоличествоКт* не указаны.

После этого перепроведем документ *Приходная накладная № 2* и убедимся, что он тоже формирует правильные проводки по регистру бухгалтерии *Управленческий*.

Теперь перейдем к более сложной задаче: добавлению движений по регистру *Управленческий* в документ *ОказаниеУслуги*.

Доработка документа «Оказание услуги»

Сначала мы изменим процедуру проведения документа *ОказаниеУслуги*, а затем в режиме *1С:Предприятие* перепроведем все эти документы, чтобы отработал новый, измененный нами алгоритм проведения документов *Оказание услуги*.

В режиме «Конфигуратор»

В отличие от документа *ПриходнаяНакладная*, который создавал всего одну бухгалтерскую проводку, документ *ОказаниеУслуги* будет создавать уже две.

Напомним, что бухгалтерия нашего ООО «На все руки мастер» не совсем похожа на настоящую, потому что для облегчения своей работы она учитывает только движения материалов. Услуги, которые оказывает организация, для нее

как бы не существуют.

Поэтому документ *ОказаниеУслуги* будет формировать движения по регистру бухгалтерии только в той части, которая касается расходования материалов.

В окне редактирования объекта конфигурации *Документ ОказаниеУслуги* перейдем на закладку *Движения*. В списке регистров отметим, что документ будет создавать теперь движения и по регистру *Управленческий*. Перейдем на закладку *Прочее* и откроем модуль объекта. Откроем процедуру обработчика события *ОбработкаПроведения*.

Поскольку нас интересует только движение материалов, для внесения дополнений подойдет тело условия *Если …*, в котором мы формировали движения по регистрам *ОстаткиМатериалов* и *СтоимостьМатериалов*.

Добавим в конец условия, перед строкой *КонецЕсли*, движения по регистру бухгалтерии *Управленческий* (листинг 16.3).

Листинг 16.3. Движения документа «ОказаниеУслуги» (фрагмент)

```
…
Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
Перечисления.ВидыНоменклатуры.Материал Тогда

  // Регистр ОстаткиМатериалов Расход
```

```
  …

  // Регистр СтоимостьМатериалов Расход
  …

  // Регистр Управленческий
  // Первая проводка:          Д 62(ДебиторскаяЗадолженность) – К 90 (Капитал)
  Розничная сумма
  Движение = Движения.Управленческий.Добавить();
  Движение.СчетДт = ПланыСчетов.Основной.ДебиторскаяЗадолженность;
  Движение.СчетКт = ПланыСчетов.Основной.Капитал;
  Движение.Период = Дата;
  Движение.Сумма = ВыборкаДетальныеЗаписи.СуммаВДокументе;
  Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Клиенты] = Клиент;

  // Вторая проводка: Д 90 (Капитал) – К 41 (Товары) – себестоимость
  Движение = Движения.Управленческий.Добавить();
  Движение.СчетДт = ПланыСчетов.Основной.Капитал;
  Движение.СчетКт = ПланыСчетов.Основной.Товары;
  Движение.Период = Дата;
  Движение.Сумма = СтоимостьМатериала *
  ВыборкаДетальныеЗаписи.КоличествоВДокументе;
  Движение.КоличествоКт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;
  Движение.СубконтоКт[ПланыВидовХарактеристик.ВидыСубконто.Материалы] =
  ВыборкаДетальныеЗаписи.Номенклатура;

КонецЕсли;
…
```

В первой проводке мы указываем розничную сумму материала из документа и субконто дебета, поскольку на счете *Дебиторская задолженность* ведется учет в разрезе клиентов.

Во второй проводке мы указываем стоимость материала, количество и субконто кредита, поскольку на счете *Товары* ведется количественный учет в разрезе материалов.

Теперь в самом начале процедуры установим свойство *Записывать* регистра бухгалтерии в значение *Истина* для записи изменений регистров в базу данных (листинг 16.4).

Листинг 16.4. Запись движений регистров

```
Процедура ОбработкаПроведения(Отказ, Режим)

  Движения.ОстаткиМатериалов.Записывать = Истина;
  Движения.СтоимостьМатериалов.Записывать = Истина;
  Движения.Продажи.Записывать = Истина;
  Движения.Управленческий.Записывать = Истина;
  …
```

В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей

регистра *Управленческий*, связанному с документом. Для этого откроем форму документа *ОказаниеУслуги*. В левом верхнем окне перейдем на закладку *Командный интерфейс*. В разделе *Панель навигации* раскроем группу *Перейти* и установим видимость для команды открытия регистра бухгалтерии *Управленческий*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки, откроем документ *Оказание услуги № 1* и нажмем *Провести*.

Выполним команду перехода к регистру *Управленческий* и посмотрим, какие движения сформировал документ в регистре бухгалтерии (рис. 16.16, 16.17).

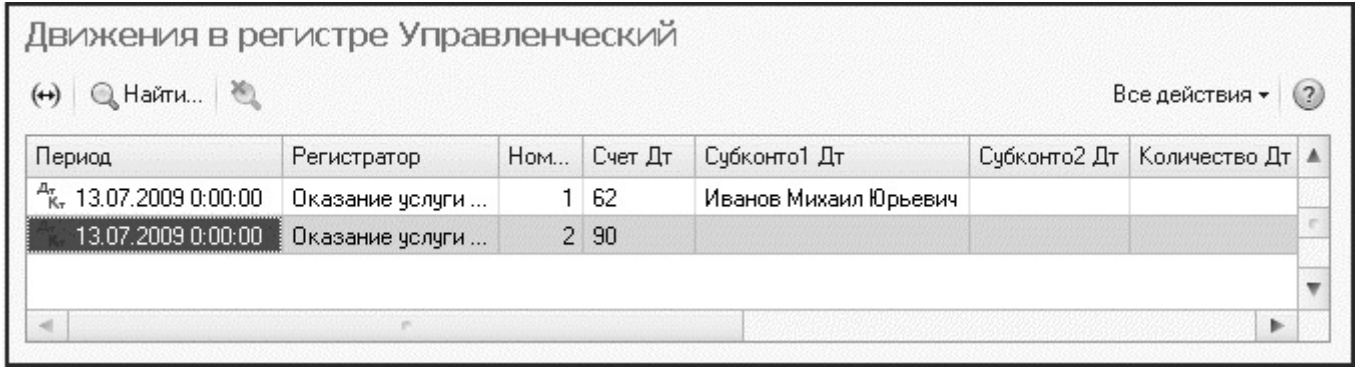


Рис. 16.16. Движения документа «Оказание услуги № 1» в регистре бухгалтерии «Управленческий»

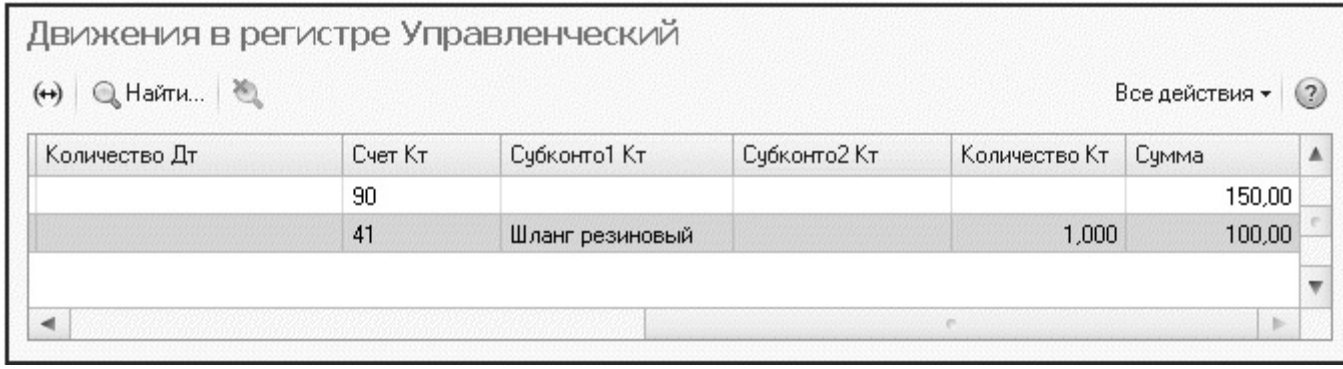


Рис. 16.17. Движения документа «Оказание услуги № 1» в регистре бухгалтерии «Управленческий»

После этого перепроведем остальные документы *Оказание услуги.*

Оборотно-сальдовая ведомость

Теперь нам только осталось создать отчет для бухгалтерии предприятия «На все руки мастер», и наше знакомство с использованием регистра бухгалтерии будет закончено.

Единственный отчет, которым пользуется бухгалтерия нашего предприятия, – это отчет *Оборотно-сальдовая ведомость* (рис. 16.18).

Оборотно-сальдовая ведомость

Параметры данных: Период = 01.07.2009 - 31.07.2009

Счет, Наименование	Сальдо нач дт	Сальдо нач кт	Оборот дт	Оборот кт	Сальдо кон дт	Сальдо кон кт
Итого			12 320,00	12 320,00	9 928,00	9 928,00
41, Товары			9 330,00	1 196,00	8 134,00	
60, Расчеты с поставщиками				9 330,00		9 330,00
62, Дебиторская задолженность			1 794,00		1 794,00	
90, Капитал			1 196,00	1 794,00		598,00
Итого			12 320,00	12 320,00	9 928,00	9 928,00

Рис. 16.18. Результат отчета

В режиме «Конфигуратор»

Для того чтобы его создать, откроем конфигуратор и добавим новый объект конфигурации Отчет с именем *ОборотноСальдоваяВедомость*. Создадим новую схему компоновки данных и добавим *Набор данных – запрос*. Откроем конструктор запроса.

Запрос для набора данных

Бухгалтерский отчет *Оборотно-сальдовая ведомость* представляет собой таблицу, в строках которой перечислены все имеющиеся в плане счетов счета, а в колонках – начальное сальдо, оборот и конечное сальдо по дебету и

кредиту каждого счета.

Поэтому нам для построения такого отчета понадобятся две исходные таблицы:

- объектная (ссылочная) таблица плана счетов *Основной*;
- виртуальная таблица регистра бухгалтерии *Управленческий.ОстаткиИОбороты* (рис. 16.19).

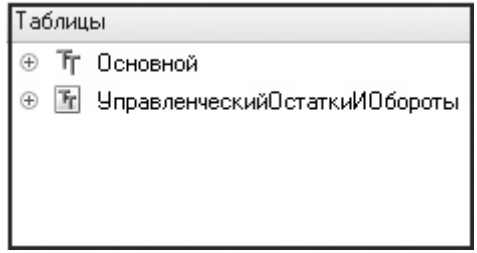


Рис. 16.19. Таблицы запроса

Из таблицы *Основной* мы выберем поле *Ссылка*, а из таблицы *УправленческийОстаткиИОбороты* возьмем следующие поля (рис. 16.20):

- *СуммаНачальныйРазвернутыйОстатокДт,*
- *СуммаНачальныйРазвернутыйОстатокКт,*
- *СуммаОборотДт,*

- *СуммаОборотКт,*
- *СуммаКонечныйРазвернутыйОстатокДт,*
- *СуммаКонечныйРазвернутыйОстатокКт.*

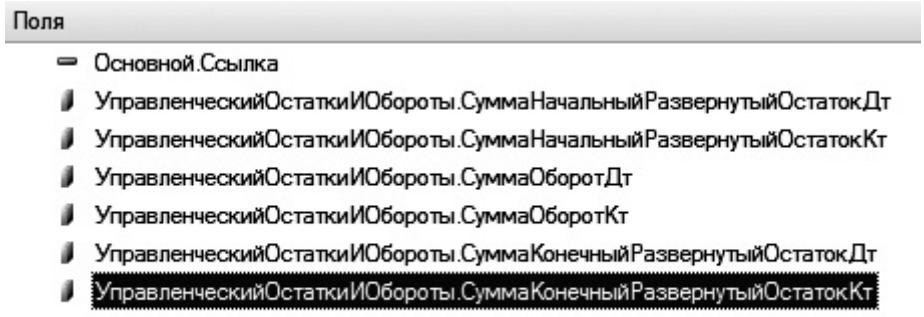


Рис. 16.20. Выбранные поля

Перейдем на закладку *Связи* и укажем, что из таблицы *Основной* мы будем выбирать все записи, а из таблицы регистра – только те, которые соответствуют условию связи (рис. 16.21).

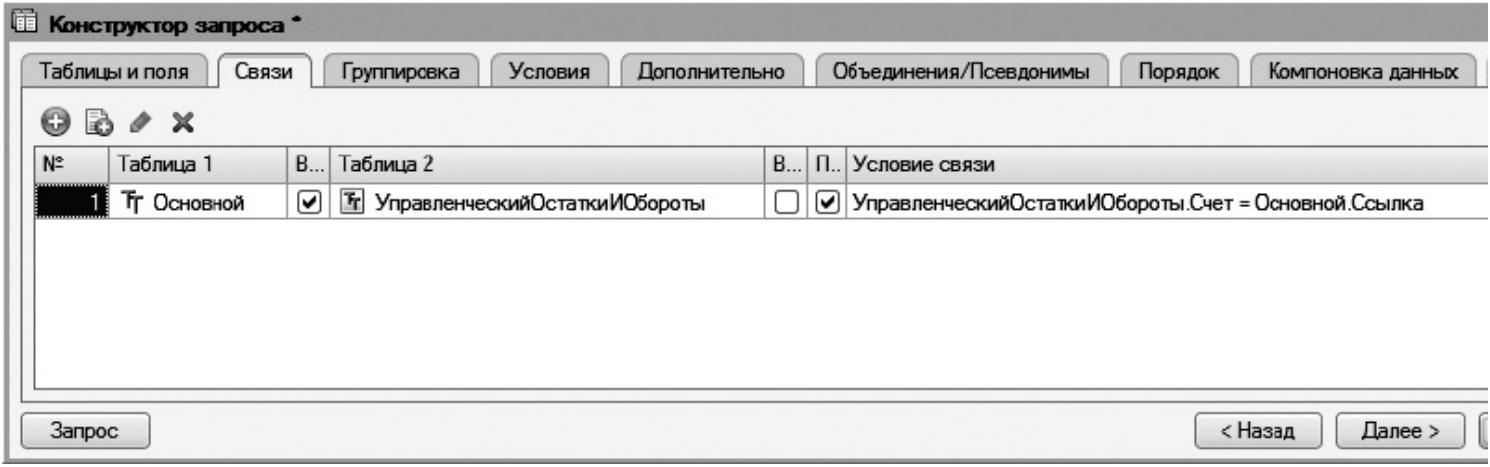


Рис. 16.21. Условие связи таблиц

Затем на закладке *Объединения/Псевдонимы* зададим псевдонимы полей отчета: *Счет, Счет.Наименование, СальдоНачДт*, *СальдоНачКт*, *ОборотДт*, *ОборотКт*, *СальдоКонДт* и *СальдоКонКт* (рис. 16.22).

Имя поля	Запрос 1
Счет	Основной.Ссылка
СальдоНачДт	УправленческийОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокДт
СальдоНачКт	УправленческийОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокКт
ОборотДт	УправленческийОстаткиИОбороты.СуммаОборотДт
ОборотКт	УправленческийОстаткиИОбороты.СуммаОборотКт
СальдоКонДт	УправленческийОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокДт
СальдоКонКт	УправленческийОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокКт

Рис. 16.22. Объединения/Псевдонимы

На этом создание запроса закончено, нажмем *OK*.

Ресурсы

Перейдем на закладку *Ресурсы* и с помощью кнопки *Добавить все ресурсы* (>>) выберем все доступные ресурсы.

Параметры

Бухгалтерские отчеты, как правило, формируются для определенного периода: месяц, квартал, год и т. д. Поэтому на примере нашего отчета продемонстрируем использование стандартного периода для указания периода отчета.

На закладке *Параметры* добавим параметр с именем *Период* типа *СтандартныйПериод,* а для параметров *НачалоПериода* и *КонецПериода* укажем *Выражение* для расчета и запретим их редактирование пользователем (листинг 16.5).

Листинг 16.5. Выражение для расчета параметров «НачалоПериода» и «КонецПериода»

```
&Период.ДатаНачала
&Период.ДатаОкончания
```

В результате параметры компоновки данных примут вид (рис. 16.26).

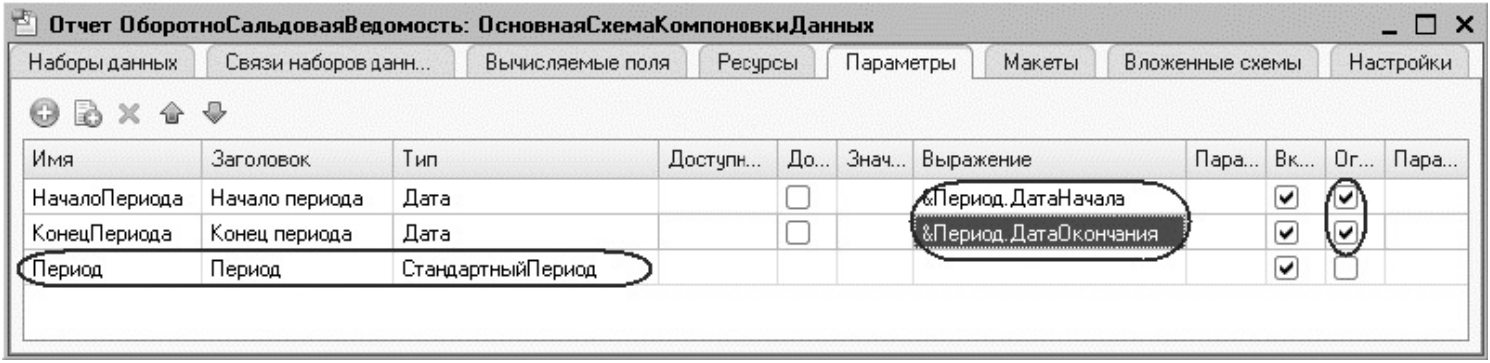


Рис. 16.23. Параметры схемы компоновки данных

Заметим, что даты начала и конца стандартного периода также содержат и время. Однако здесь, в отличие от параметров *НачалоПериода* и *КонецПериода*, начальная дата имеет время 00:00:00, а конечная дата – 23:59:59. Таким образом, последний день включается в отчет, и не нужно использовать функцию *КонецПериода()*.

Настройки

В заключение перейдем на закладку *Настройки* и создадим структуру отчета. Добавим группировку, содержащую детальные записи. Затем на закладке *Выбранные поля* выберем все поля для вывода в отчет и разместим их в следующем порядке (рис. 16.24).

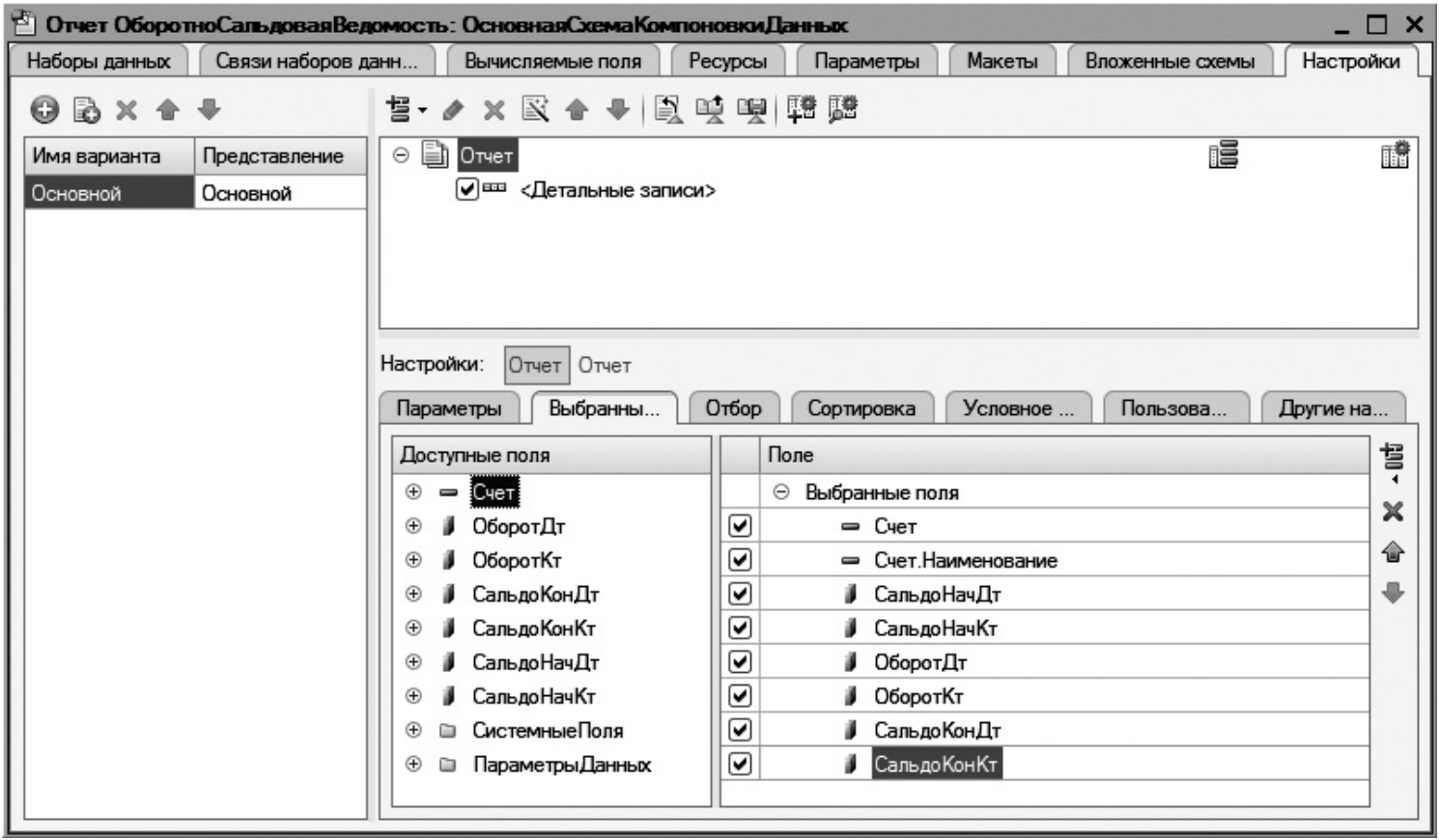


Рис. 16.24. Поля и группировки отчета

На закладке *Другие настройки* укажем заголовок отчета – *Оборотно-сальдовая ведомость.* Для параметра *Расположение общих итогов по вертикали* укажем значение *Начало и конец*. Затем на закладке *Параметры*

выберем для параметра *Период* значение из списка стандартных периодов – *Этот месяц* (рис. 16.25).

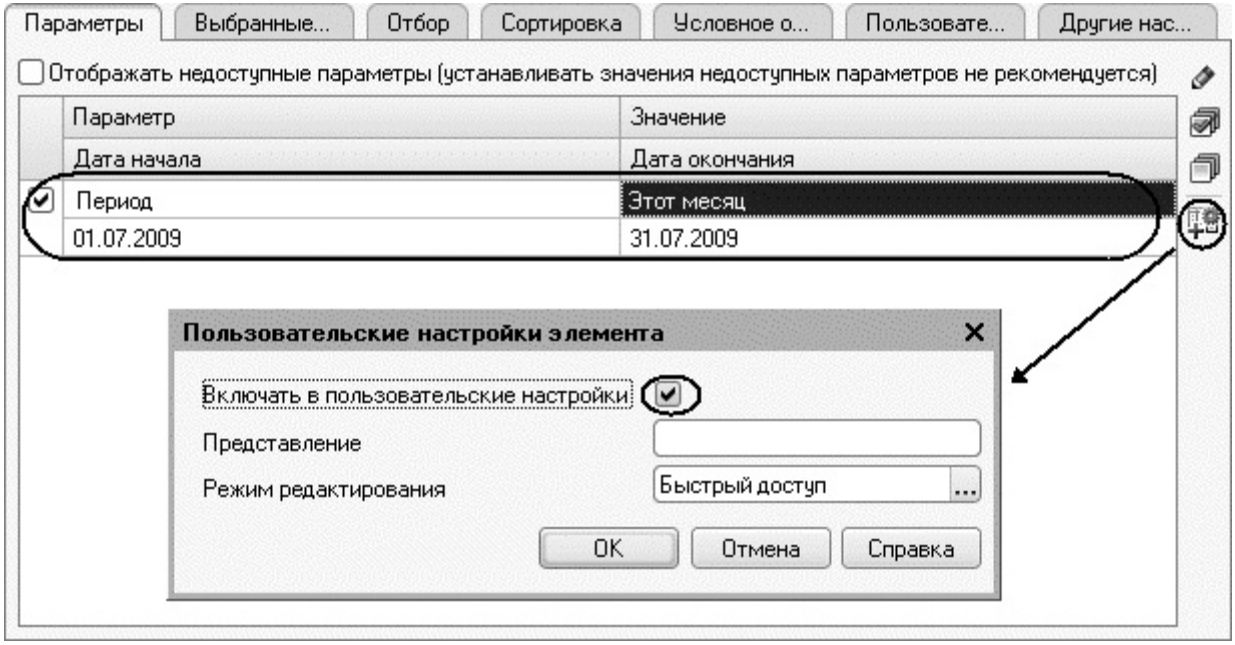


Рис. 16.25. Создание быстрых настроек отчетного периода

Тем самым мы обеспечим, что при открытии формы отчета в настройке отчетного периода всегда будет указан текущий месяц. Причем даты начала и конца периода будут динамически меняться в зависимости от даты выполнения отчета. А также, нажав кнопку *Свойства элемента пользовательских настроек*, укажем, что параметр *Период* будет включен в состав

пользовательских настроек, и эта настройка будет находиться непосредственно в отчетной форме (режим редактирования *Быстрый доступ*).

В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закроем конструктор схемы компоновки данных и в окне редактирования объекта конфигурации *Отчет ОборотноСальдоваяВедомость* перейдем на закладку *Подсистемы*.

Отметим в списке подсистем конфигурации подсистему *Бухгалтерия*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает отчет.

В разделе *Бухгалтерия* откроем отчет и нажмем *Сформировать* (рис. 16.26).

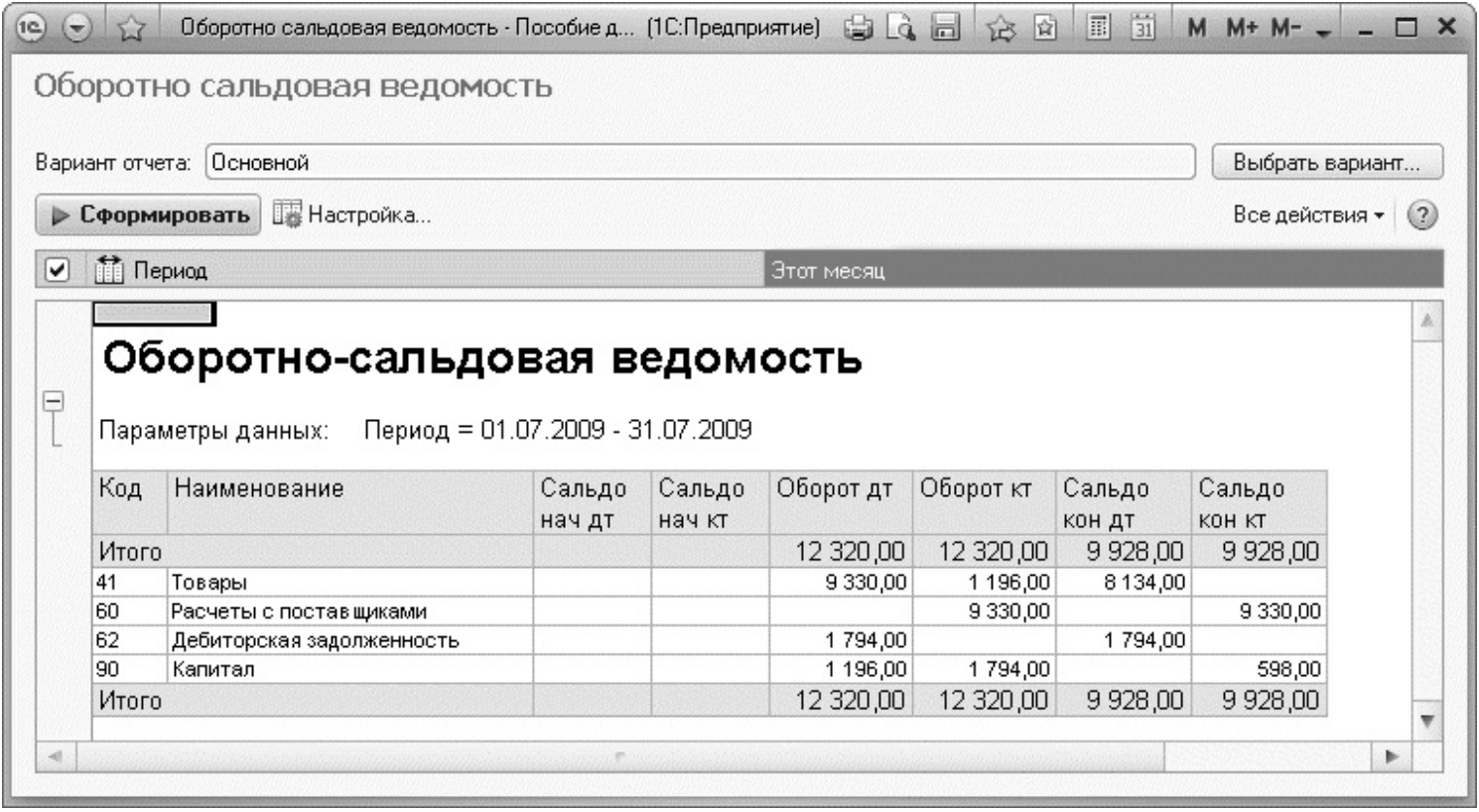


Рис. 16.26. Результат отчета

Мы видим, что стандартный период отчета задан по умолчанию – *Этот месяц*. Пользоваться стандартным периодом отчета очень удобно, когда пользователь регулярно выполняет отчет за определенный интервал времени. Тогда можно заранее установить в настройках нужный период, и пользователю не придется

задавать его перед формированием отчета. Но, при желании в режиме *1С:Предприятие* пользователь может выбрать другой период из списка стандартных периодов.

Контрольные вопросы

- *Как использовать план видов характеристик для организации ведения бухгалтерского учета.*
- *Что такое субконто.*
- *Для чего предназначен объект конфигурации «План счетов».*
- *Как создать план счетов.*
- *Для чего предназначен «Регистр бухгалтерии».*
- *Как создать регистр бухгалтерии и настроить параметры учета.*
- *Как создать движения документа по регистру бухгалтерии средствами встроенного языка.*
- *Как получить данные из регистра бухгалтерии запросом.*
- *Как создать отчет на основании данных из регистра бухгалтерии с помощью системы компоновки.*
- *Как задать роли и тип бухгалтерского остатка полям в схеме компоновки данных.*
- *Как задать стандартный период для выполнения отчета.*

Занятие 17 (1:00). План видов расчета, регистр расчета

Продолжительность

Ориентировочная продолжительность занятия – 1 час.

На этом занятии мы познакомимся с объектами конфигурации *План видов расчета* и *Регистр расчета* и узнаем об основных понятиях, используемых при создании сложных периодических расчетов.

В конце занятия мы создадим план видов расчета и регистр расчета, на основе которых на следующем занятии продемонстрируем работу механизмов периодических расчетов.

Зачем нужен план видов расчета и регистр расчета?

На этом занятии мы рассмотрим, какие возможности для автоматизации сложных периодических расчетов предоставляет система «1С:Предприятие».

Такие расчеты используются прежде всего при расчете заработной платы. Поэтому дальнейшее их рассмотрение мы будем строить на примере расчета заработной платы сотрудников, которые работают в нашем ООО «На все руки

мастер».

В общем случае сумма заработной платы сотрудника складывается из множества частей (например, оплата по окладу, премии, штрафы, оплаты по больничному листу, разовые выплаты и т. д.). Каждая из этих частей рассчитывается по некоторому алгоритму, присущему только этой части.

Например, сумма штрафа может определяться просто фиксированной суммой, сумма премии может рассчитываться как процент от оклада, а сумма оплаты по окладу рассчитывается исходя из количества рабочих дней в месяце и количества дней, отработанных сотрудником. Поэтому для обозначения каждой такой части мы будем использовать термин *вид расчета*.

Алгоритм каждого вида расчета опирается в общем случае на две категории параметров: период, за который нужно получить конечные данные, и набор некоторых исходных данных, используемых при расчете.

Как правило, в реальной жизни различные виды расчета существуют не сами по себе, а оказывают некоторое влияние на другие виды расчета. Исходя из того, что вид расчета опирается на две различные категории параметров, такое влияние тоже имеет двойственный характер.

Зависимость по базовому периоду

Это может быть влияние на исходные данные, используемые при расчете.

В качестве примера можно привести начисление премии в виде процента от оплаты по окладу. При изменении оплаты по окладу размер премии тоже должен быть пересчитан, исходя из новой суммы начисленного оклада. Другими словами, сумма начисленного оклада является базой для расчета премии.

Причем поскольку оклад рассчитывается за некоторый период, при расчете премии нам интересно знать не значение оклада вообще, а сумму, начисленную в том периоде, который влияет на расчет премии. Такой период мы будем называть *базовым*, а подобную зависимость между видами расчета – *зависимостью по базовому периоду*.

В качестве примера рассмотрим начисление премии за апрель. Премия должна начисляться в размере 10 % от суммы, начисленной в качестве оплаты по окладу. Следовательно, необходимо проанализировать все записи о начислениях оплаты по окладу, которые попадают в интересующий нас базовый период, а именно апрель.

Допустим, общая сумма таких начислений составила 8 000 рублей – в этом случае премия должна быть начислена в размере 800 рублей (рис. 17.1).

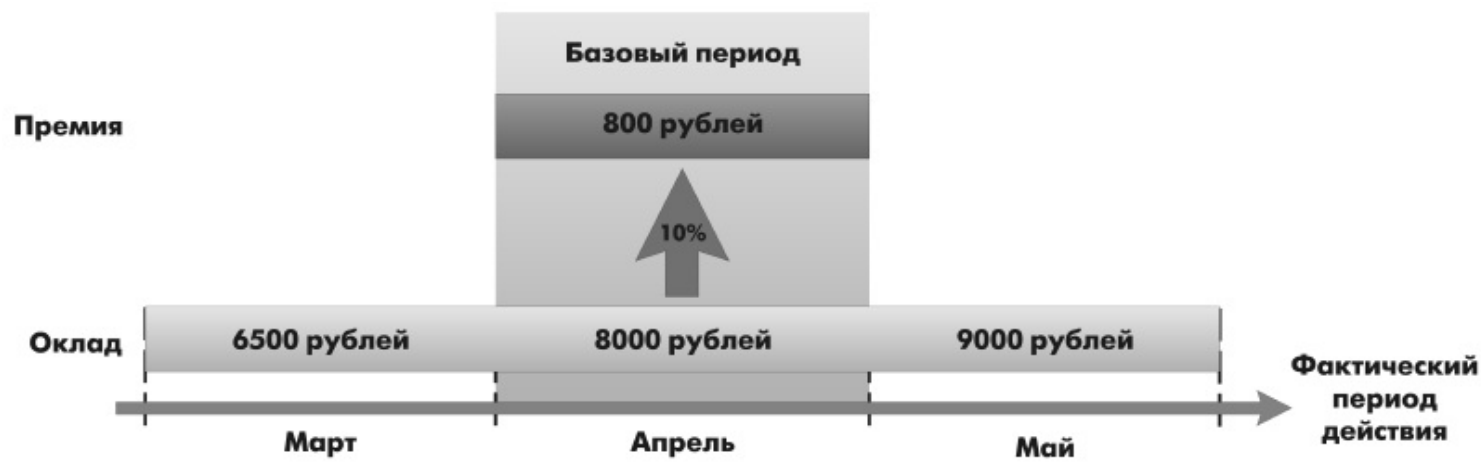


Рис. 17.1. Зависимость премии от оклада по базовому периоду

Вытеснение по периоду действия

Это влияние может быть не на исходные данные, а на сам период, за который производится расчет.

В качестве примера можно привести расчет оплаты по окладу и невыход на работу. Предположим, мы начислили сотруднику оплату по окладу за март месяц. В этом случае период действия такого расчета будет с 01.03.2009 по 31.03.2009.

После этого мы получили информацию от руководителя отдела, что,

оказывается, сотрудник отсутствовал на работе с 1 по 10 марта по неизвестной причине. В этом случае нам нужно будет произвести расчет *Невыход* (в котором можно рассчитать какие-то удержания с сотрудника).

Но кроме этого необходимо будет пересчитать и оклад сотрудника, исходя из того, что фактический период действия расчета *Оклад* стал теперь с 11.03.2009 по 31.03.2009.

Такое влияние мы будем называть *вытеснением по периоду действия*.

В результате если за полный месяц работы сотруднику должно было быть начислено 9 300 рублей, то теперь, за фактический период работы, начисление составит 6 300 рублей (рис. 17.2).

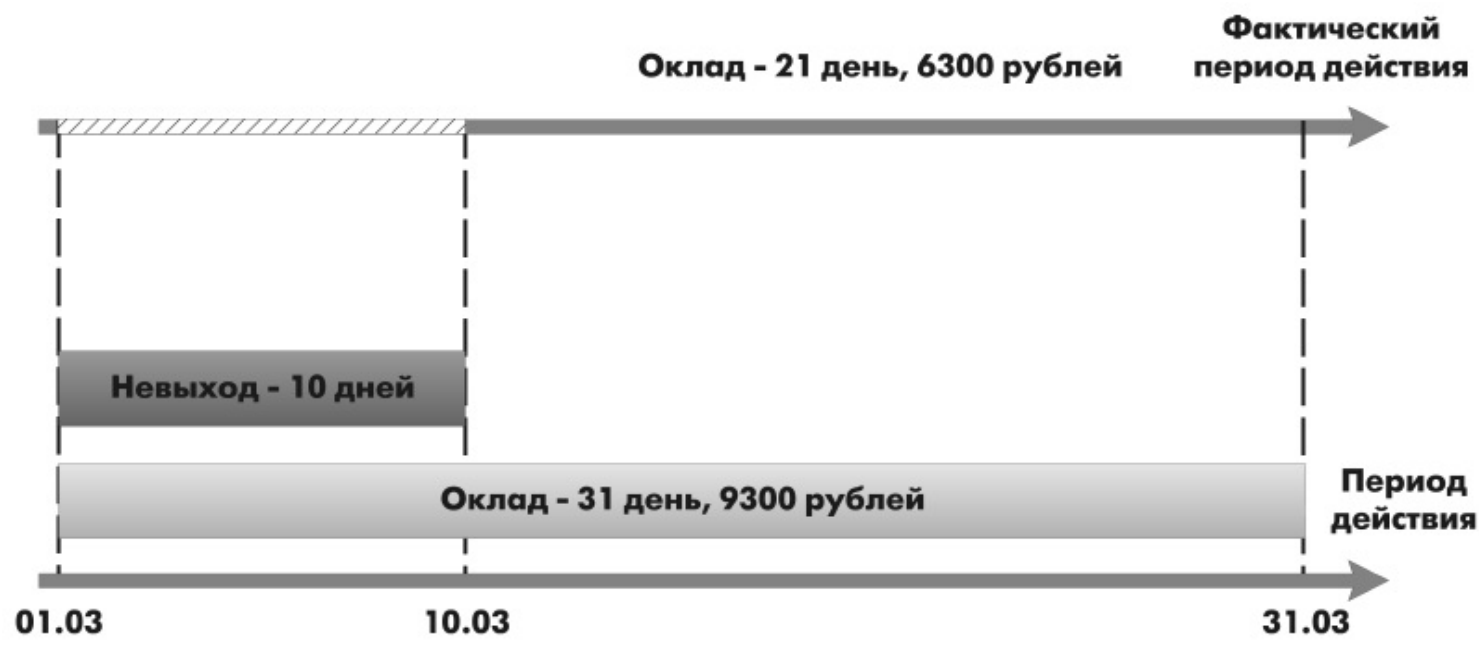


Рис. 17.2. Запись расчета «Невыход» вытесняет запись расчета «Оклад» по периоду действия

Таким образом, исходя из двух видов взаимного влияния расчетов, можно сказать, что в общем случае с каждым видом расчета будет связано три периода: период действия, фактический период и базовый период.

- *Период действия* является «запрашиваемым». То есть, указывая период действия, мы говорим: «Мы хотели бы, чтобы результат действовал в этом периоде».

- *Фактический период* – это то, что получилось из периода действия после анализа всех периодов действия расчетов, которые вытесняют наш по периоду действия.
- *Базовый период* – это период, в котором мы анализируем результаты других расчетов, влияющих на наш по базовому периоду.

Как видите, взаимное влияние между видами расчетов может быть довольно разнообразным и, что самое сложное, многоуровневым. То есть один вид расчета может влиять на другой, который, в свою очередь, влияет на третий и т. д.

Очевидно, что в этой ситуации требуется некий универсальный механизм, позволяющий описать каждый из видов расчетов (его алгоритм, влияние на другие виды расчетов, зависимость от других видов расчетов), обеспечить хранение данных, полученных в результате этих расчетов, и контроль необходимости перерасчета результатов зависимых расчетов в случае изменения результатов первичных расчетов.

В системе «1С:Предприятие» такой универсальный механизм реализован при помощи планов видов расчета и регистров расчета.

И первым объектом конфигурации, с которым мы начнем знакомиться на этом занятии, будет *План видов расчета*.

Что такое план видов расчета

Объект конфигурации *План видов расчета* предназначен для описания структуры хранения информации о возможных видах расчетов. На основе объекта конфигурации *План видов расчета* платформа создает в базе данных таблицу, в которой будет храниться информация о том, какие существуют виды расчета и каковы взаимосвязи между ними.

Отличительной особенностью плана видов расчета является то, что пользователь в процессе работы может добавлять новые виды расчета. Такая возможность делает механизм периодических расчетов более гибким и позволяет пользователю создавать собственные виды расчета, помимо тех, которые заданы разработчиком как предопределенные.

Объект конфигурации *План видов расчета* имеет свойство *Использует период действия*. С его помощью определяется, будут ли в этом плане находиться виды расчета, которые могут быть вытеснены по периоду действия.

Если это свойство установлено, то разработчик получает возможность указать для каждого вида расчета те виды, которые вытесняют его по периоду действия.

Следующим важным свойством объекта конфигурации *План видов расчета*

является *Зависимость от базы*. Оно определяет, будут ли в этом плане находиться зависимые по базовому периоду виды расчета.

Если это свойство установлено, появляется возможность указать, в каком плане видов расчета будут находиться базовые виды расчета и, кроме этого, как будет определяться эта зависимость.

Существует возможность указать один из двух видов зависимости от базы: *Зависимость по периоду действия* и *Зависимость по периоду регистрации*. Оба вида этой зависимости подробно рассмотрены в разделе «Что такое Регистр расчета».

Еще одной важной особенностью плана видов расчета является возможность создания предопределенных видов расчета и описания их взаимного влияния. При этом в общем случае разработчик имеет возможность указать три категории видов расчета, влияющих на предопределенный вид расчета:

- *Базовые* – их результаты должны быть использованы при перерасчете этого вида расчета;
- *Вытесняющие* – вытесняют этот вид расчета по периоду действия;
- *Ведущие* – изменение их результатов должно приводить к необходимости перерасчета этого вида расчета.

Здравый смысл подсказывает, что все базовые виды расчета должны быть включены и в категорию ведущих. Кроме этого, ведущие виды расчета могут содержать и некоторые другие виды, косвенно влияющие на данный вид расчета.

Например, мы имеем три вида расчета: невыход, оклад и премия. Невыход вытесняет оклад по периоду действия, а премия зависит от оклада по базовому периоду.

В этом случае для премии следует указать базовым видом расчета оклад, а ведущими – оклад и невыход, поскольку изменение результата расчета невыхода приведет к изменению результата оклада, что, в свою очередь, должно привести к изменению результата премии (рис. 17.3).

Рис. 17.3. Взаимное влияние видов расчетов

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с планом видов расчета, можно прочитать в разделе «Краткий справочник разработчика. Планы видов расчета».

Добавление плана видов расчета

Приступим теперь к созданию плана видов расчета *ОсновныеНачисления*,

который будет использоваться в нашей конфигурации.

В режиме «Конфигуратор»

Откроем конфигуратор и создадим новый объект конфигурации *План видов расчета*.

Зададим его имя – *ОсновныеНачисления*, а также зададим *Представление списка* как *Виды расчетов*.

На закладке *Подсистемы* укажем, что план видов расчета будет отображаться в подсистеме *РасчетЗарплаты*.

На закладке *Расчет* укажем, что он будет использовать период действия и зависеть от базы по периоду действия.

В качестве базового плана видов расчета укажем его самого, поскольку все наши виды расчетов будут храниться в единственном плане видов расчета (рис. 17.4).

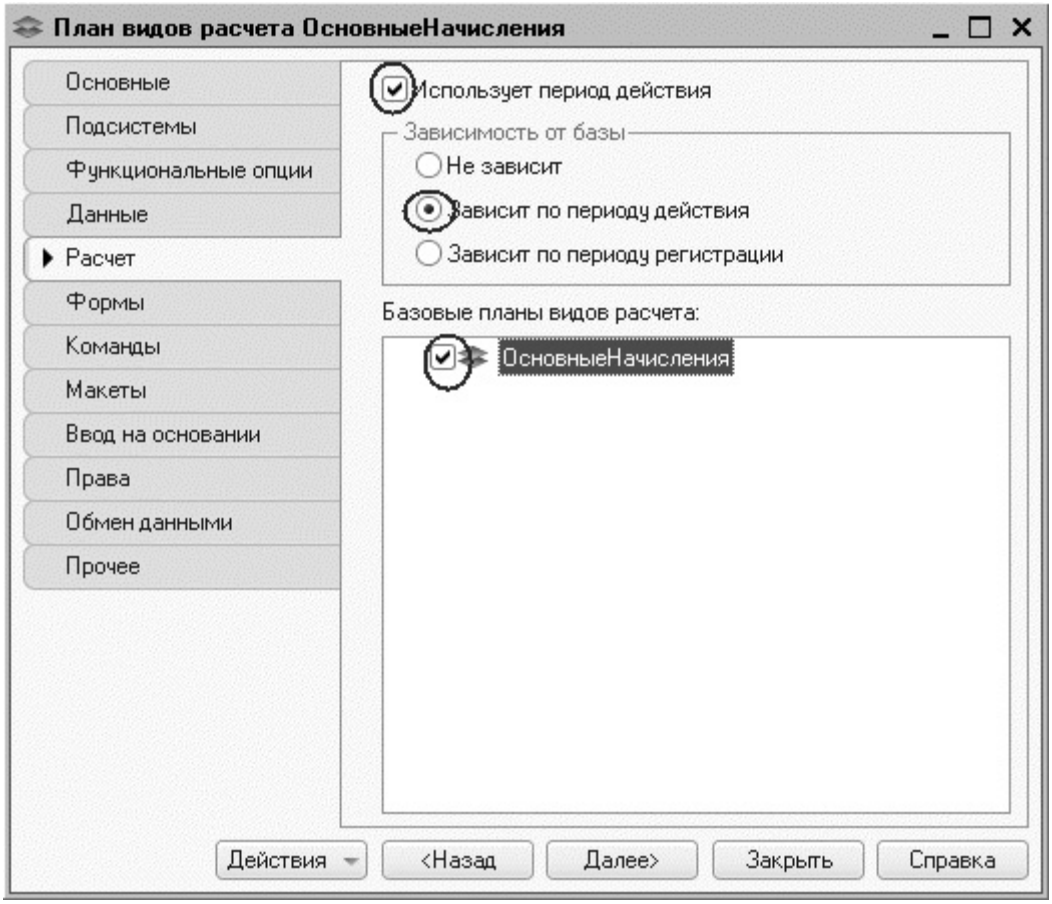


Рис. 17.4. Определим использование периода действия, зависимость от базы и базовые планы видов расчета

Перейдем на закладку *Прочее* и зададим предопределенные виды расчета.

Как и в случае с бухгалтерией, расчеты в нашем ООО «На все руки мастер» будут скромные, поэтому мы создадим всего три элемента (рис. 17.5):

- *Невыход* – с именем и наименованием *Невыход* и кодом *Невыход*;
- *Оклад* – с именем, кодом и наименованием *Оклад* и вытесняющим его видом расчета *Невыход*;
- *Премия* – с именем, кодом и наименованием *Премия*, с базовым видом расчета *Оклад* и ведущими видами расчета *Невыход* и *Оклад*.

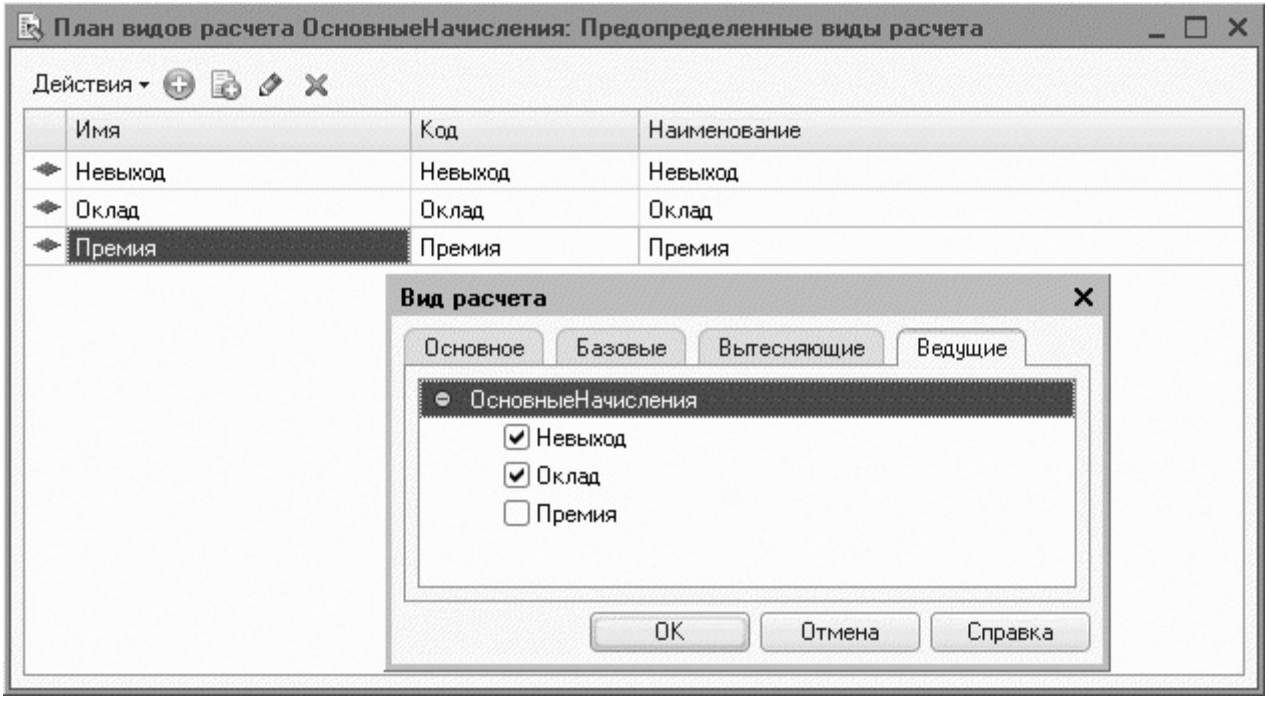


Рис. 17.5. Предопределенные виды расчета для плана видов расчета «ОсновныеНачисления»

Теперь мы перейдем к рассмотрению второго объекта, используемого при реализации механизмов сложных периодических расчетов, – регистра расчета.

Что такое регистр расчета

Объект конфигурации *Регистр расчета* предназначен для описания структуры накопления данных, являющихся результатами расчетов. На основе объекта

конфигурации *Регистр расчета* платформа создает в базе данных таблицы, в которых будут накапливаться данные, формируемые различными объектами базы данных.

Отличительной особенностью регистра расчета является то, что он не предназначен для интерактивного редактирования пользователем. Разработчик может при необходимости предоставить пользователю возможность редактировать регистр расчета, но предназначение регистра расчета заключается в том, чтобы его модификация производилась на основе алгоритмов работы объектов базы данных, а не в результате непосредственных действий пользователя.

Как и другие регистры, регистр расчета имеет ресурсы, в которых хранит числовые данные; имеет измерения, в разрезе которых можно получать значения ресурсов регистра; имеет реквизиты, которые характеризуют каждую запись регистра расчета.

Отличительными же особенностями регистра расчета является его периодичность, возможность использования механизмов вытеснения по периоду действия и зависимости по базовому периоду, а также связь с планом видов расчета. Рассмотрим все эти особенности по порядку.

Периодичность

Периодичность регистра расчета может быть определена одним из следующих значений:

- *День,*
- *Месяц,*
- *Квартал,*
- *Год.*

Периодичность регистра расчета определяет промежуток времени, к которому будет относиться каждая запись регистра.

Если указана периодичность *День*, то каждая запись регистра будет относиться к какому-либо дню; если периодичность – *Месяц*, то к какому-либо месяцу и т. д.

Для указания факта принадлежности записи к какому-либо периоду регистр имеет служебный реквизит *Период Регистрации* типа *Дата*. При записи данных в регистр платформа всегда приводит значение этого реквизита к началу того периода, в который он попадает.

Например, если в регистр расчета с периодичностью месяц записать данные, где *ПериодРегистрации* задан как *08.04.2004*, то регистр сохранит эти данные

со значением поля *ПериодРегистрации 01.04.2004* (рис. 17.6).


Документ
Дата:
08.04.2004
Модуль документа
...
Движения.ПериодРегистрации = Дата;
Движения. Записать ();
...
Таблица регистра расчета
(периодичность регистра – «Месяц»)
Регистратор
Период регистрации
Номер строки
...
01.04.2004 00:00:00
1
01.04.2004 00:00:00
...
01.04.2004 00:00:00
N

Рис. 17.6. Запись данных из документа в регистр расчета видов расчета

Если в этой же ситуации периодичность регистра будет год, сохраненное значение периода регистрации будет *01.01.2004* (рис. 17.7).


Документ
Дата:
08.04.2004
Модуль документа
...
Движения.ПериодРегистрации = Дата;
Движения. Записать ();
...
Таблица регистра расчета
(периодичность регистра – «Год»)
Регистратор
Период регистрации
Номер строки
...
01.01.2004 00:00:00
1
01.01.2004 00:00:00
...
01.01.2004 00:00:00
N

Рис. 17.7. Запись данных из документа в регистр расчета видов расчета

Вытеснение по периоду действия

Следующей важной особенностью регистра расчета является возможность использования механизма вытеснения одних записей другими по периоду действия.

При этом для каждой записи регистр расчета формирует фактический период действия, который является в общем случае совокупностью нескольких периодов, расположенных внутри периода действия (рис. 17.8).

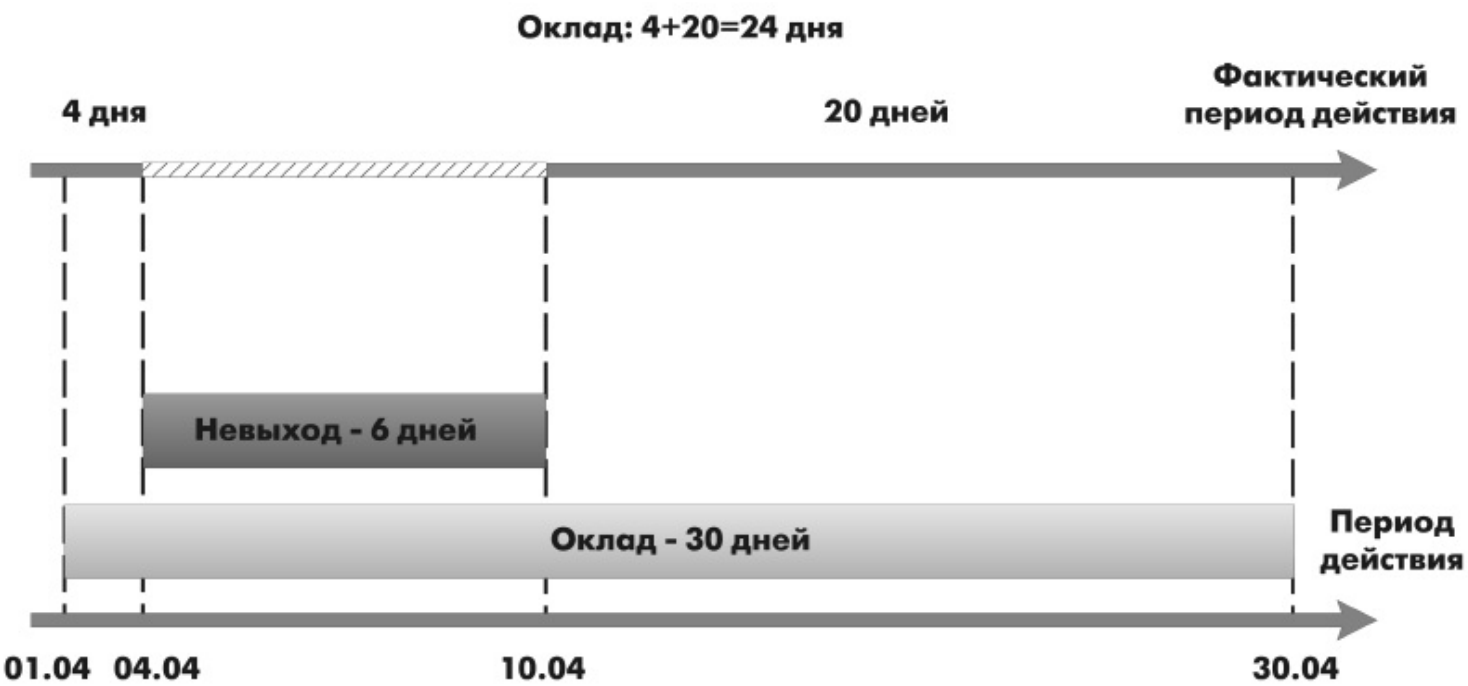


Рис. 17.8. Запись расчета «Невыход» вытесняет запись расчета «Оклад» по периоду действия

Если рассмотреть структуру записей таблиц регистра расчета, то после внесения записи о начислении по окладу таблицы регистра будут выглядеть следующим образом (таблицы 17.1, 17.2).

Таблица 17.1. Таблица регистра расчета

…	Начало периода действия	Конец периода действия	Вид расчета	…

…	…	…	…	…
…	01.04.2004 00:00:00	30.04.2004 23:59:59	Оклад	…
…	…	…	…	…

Таблица 17.2. Таблица фактического периода действия

…	Начало периода действия	Конец периода действия	Вид расчета	…
…	…	…	…	…
…	01.04.2004 00:00:00	30.04.2004 23:59:59	Оклад	…
…	…	…	…	…

После добавления в регистр записи вида расчета *Невыход*, который вытесняет вид расчета *Оклад* по периоду действия, записи о начислении по окладу примут следующий вид (таблицы 17.3, 17.4).

Таблица 17.3. Таблица регистра расчета

…	Начало периода действия	Конец периода действия	Вид расчета	…
…	…	…	…	…
…	01.04.2004 00:00:00	30.04.2004 23:59:59	Оклад	…
…	…	…	…	…

…	04.04.2004 00:00:00	10.04.2004 23:59:59	Невыход	…
…	…	…	…	…

Таблица 17.2. Таблица фактического периода действия

…	Начало периода действия	Конец периода действия	Вид расчета	…
…	…	…	…	…
…	01.04.2004 00:00:00	03.04.2004 23:59:59	Оклад	…
…	11.04.2004 00:00:00	31.04.2004 23:59:59	Оклад	…
…	…	…	…	…

Зависимость по базовому периоду

Другим механизмом, который поддерживает регистр расчета, является зависимость записей по базовому периоду. Этот механизм позволяет основывать расчет зависимых (вторичных) записей регистра на данных, полученных в результате расчета первичных записей.

Регистр расчета может поддерживать два вида зависимости от базы: зависимость по периоду действия и зависимость по периоду регистрации.

Зависимость по периоду действия

Зависимость по периоду действия означает, что при анализе базовых записей будут выбираться те, для которых найдено пересечение их фактического периода действия и указанного базового периода.

Например, в начале апреля производится расчет зарплаты за март. Премия за март должна быть начислена исходя из оплаты по окладу за март. В этом случае, как правило, используется зависимость по периоду действия (рис. 17.9).

Таблица регистра расчета

Регистратор	Номер строки	Начало периода действия	Конец периода действия	Начало базового периода	Конец базового периода	Вид расчета	Результат	...
...	...	...	...	...	...	...	...	...
(док.3)	3	01.03.2004 00:00:00	31.03.2004 23:59:59	01.03.2004 00:00:00	31.03.2004 23:59:59	(премия)	3 000 *X	...
...	...	...	...	...	...	...	...	...
(док.2)	5	01.03.2004 00:00:00	31.03.2004 23:59:59	...	...	(оклад)	3 000	...
...	...	...	...	...	...	...	...	...
(док.1)	2	01.04.2004 00:00:00	30.04.2004 23:59:59	...	...	(оклад)	5 000	...
...	...	...	...	...	...	...	...	...

Таблица фактического периода действия

Регистратор	Номер строки	Начало периода действия	Конец периода действия	...
...	...	...	...	...
(док.3)	3	01.03.2004 00:00:00	25.03.2004 23:59:59	...
...	...	...	...	...
(док.2)	5	01.03.2004 00:00:00	31.03.2004 23:59:59	...
...	...	...	...	...
(док.1)	2	02.04.2004 00:00:00	03.04.2004 23:59:59	...
(док.1)	2	05.04.2004 00:00:00	21.04.2004 23:59:59	...
(док.1)	2	23.04.2004 00:00:00	30.04.2004 23:59:59	...
...	...	...	...	...

Рис. 17.9. Зависимость по периоду действия

Следует сделать два замечания к приведенному рисунку.

Поля *Начало базового периода* и *Конец базового периода* имеют смысл только для записей тех видов расчета, для которых определена зависимость по базовому периоду (в нашем случае для записи расчета премии).

Значение базы, которая будет получена от конкретной влияющей записи, в общем случае не равно результату, который содержит эта запись. База будет рассчитана пропорционально тому, какую часть от фактического интервала влияющей записи составляет перекрывающийся с указанным базовым периодом участок. При этом будут использованы данные графика, связанного с записью.

Зависимость по периоду регистрации

Зависимость по периоду регистрации означает, что при анализе базовых записей будут выбираться те, которые попадают в указанный базовый период значением своего поля *Период регистрации*.

В качестве примера можно привести расчет штрафов при начислении зарплаты за март. В качестве базы для расчета суммы штрафов должны браться записи о прогулах, зарегистрированные в марте месяце (это могут быть как записи о

мартовских прогулах, так и записи о прогулах в феврале). В этом случае, как правило, используется зависимость по периоду регистрации (рис. 17.10).

Таблица регистра расчета
(периодичность регистр - «Месяц»)

Период регистрации	Начало периода действия	Конец периода действия	Начало базового периода	Конец базового периода	Вид расчета	Результат
...	...	...	...	...	...	...
01.04.2004 00:00:00	01.03.2004 00:00:00	31.03.2004 23:59:59	01.03.2004 00:00:00	31.03.2004 23:59:59	(штраф)	2*X
...	...	...	...	...	...	...
01.03.2004 00:00:00	26.02.2004 00:00:00	27.02.2004 23:59:59	...	...	(прогул)	2
...	...	...	...	...	...	...
01.02.2004 00:00:00	07.02.2004 00:00:00	10.02.2004 00:00:00	...	...	(прогул)	4
...	...	...	...	...	...	...

Рис. 17.10. Зависимость по периоду регистрации

Заключительной важной особенностью регистра расчета является его связь с планом видов расчета. Именно на основе этой связи работают механизмы вытеснения по периоду действия и зависимости по базовому периоду, поскольку в плане видов расчета описано взаимное влияние видов расчета друг на друга.

У регистра расчета могут существовать подчиненные объекты *Перерасчет*. Они предназначены для регистрации фактов появления в регистре записей, влияющих на результат расчета уже существующих записей регистра. Объект конфигурации *Перерасчет* может иметь несколько измерений, каждое из которых устанавливает связь между измерениями данного регистра расчета и влияющих регистров расчета. В частном случае это может быть один и тот же регистр.

В таблице, созданной в базе данных на основе объекта конфигурации *Перерасчет*, платформа хранит информацию о том, какие записи регистра подлежат перерасчету. Таблицы перерасчета заполняются автоматически как на основании записей регистров расчета, затронутых ведущими видами расчета, так и на основании записей регистра расчета, для которых изменился фактический период действия. Исходя из этой информации, разработчик может принимать решение о необходимости перерасчета записей регистра.

Последним замечанием, которое следует сделать, говоря о регистре расчета, является возможность установки связи регистра расчета с графиком времени. Такой график времени должен представлять собой регистр сведений (непериодический, с обязательным измерением типа *Дата* и ресурсом типа *Число*), в котором содержится временная схема исходных данных, участвующих в расчетах. Измерениями этого графика могут быть, например, график работы (ссылка на справочник) и дата, а ресурсом – количество рабочих часов в этой

дате. В этом случае можно будет связать запись регистра расчета с каким-либо конкретным графиком работы (указав в качестве реквизита записи ссылку на справочник *ВидыГрафиковРаботы*) и в дальнейшем средствами встроенного языка получать информацию о количестве рабочих часов в периоде действия, фактическом периоде действия или периоде регистрации этой записи.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с регистром расчета, можно прочитать в разделе [«Краткий справочник разработчика. Регистры расчета»](#)*.*

Добавление регистра расчета

Прежде чем мы начнем создавать объект конфигурации *Регистр расчета Начисления*, нам потребуется создать два дополнительных объекта конфигурации:

- регистр сведений *ГрафикиРаботы,*
- справочник *ВидыГрафиковРаботы*.

Справочник понадобится нам для хранения информации о том, какие графики

работы существуют в ООО «На все руки мастер», а регистр сведений – для указания того, какие дни в месяце являются рабочими, поскольку сумма оплаты по окладу будет рассчитываться исходя из того, сколько дней отработал сотрудник в расчетном месяце.

В режиме «Конфигуратор»

Откроем конфигуратор и создадим новый объект конфигурации *Справочник* с именем *ВидыГрафиковРаботы*.

На закладке *Подсистемы* укажем, что справочник будет отображаться в подсистеме *РасчетЗарплаты*.

На закладке *Прочее* создадим для справочника два предопределенных графика работы (рис. 17.11) – *ГрафикАдминистрации* и *ГрафикМастеров*.

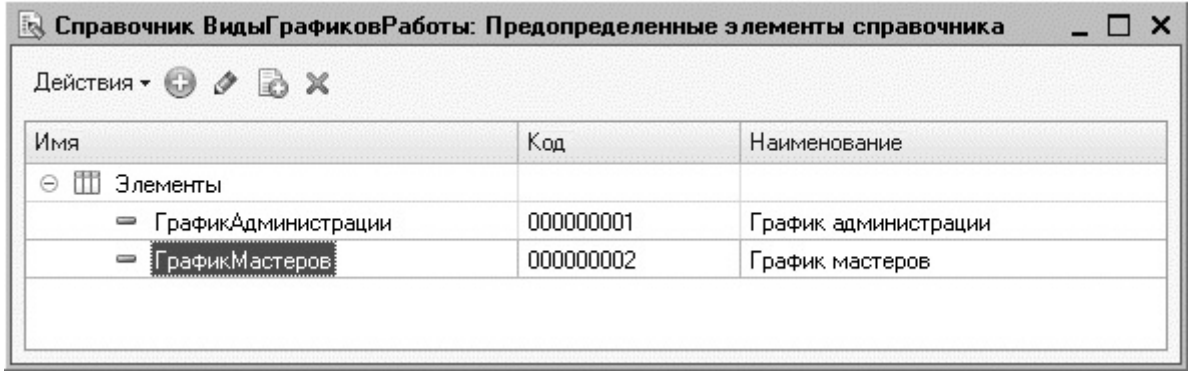


Рис. 17.11. Предопределенные графики работы

После этого создадим объект конфигурации *Регистр сведений* с именем *ГрафикиРаботы*.

Этот регистр будет иметь два измерения:

- *ГрафикРаботы*, тип *СправочникСсылка.ВидыГрафиковРаботы*;
- *Дата*, тип *Дата*.

Затем создадим единственный ресурс регистра – *Значение*, с типом *Число*, длиной *1*. На закладке *Подсистемы* укажем, что регистр сведений будет отображаться в подсистеме *РасчетЗарплаты*. Теперь заполним регистр сведений *ГрафикиРаботы* данными о рабочих днях июля графика мастеров.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и в разделе *Расчет зарплаты* выполним команду *Графики работы*. Поочередно создадим 31 запись в регистре.

Чтобы проще выполнить эту довольно однообразную работу, воспользуйтесь возможностью добавления элементов в регистр копированием текущего

элемента (кнопка *Создать новый элемент копированием текущего (F9)*). В качестве измерения *ГрафикРаботы* нашего регистра выберем предопределенный элемент *График мастеров* справочника *ВидыГрафиковРаботы*. В качестве ресурса *Значение* у рабочих дней проставим 1, а у выходных – 0 (рис. 17.12).

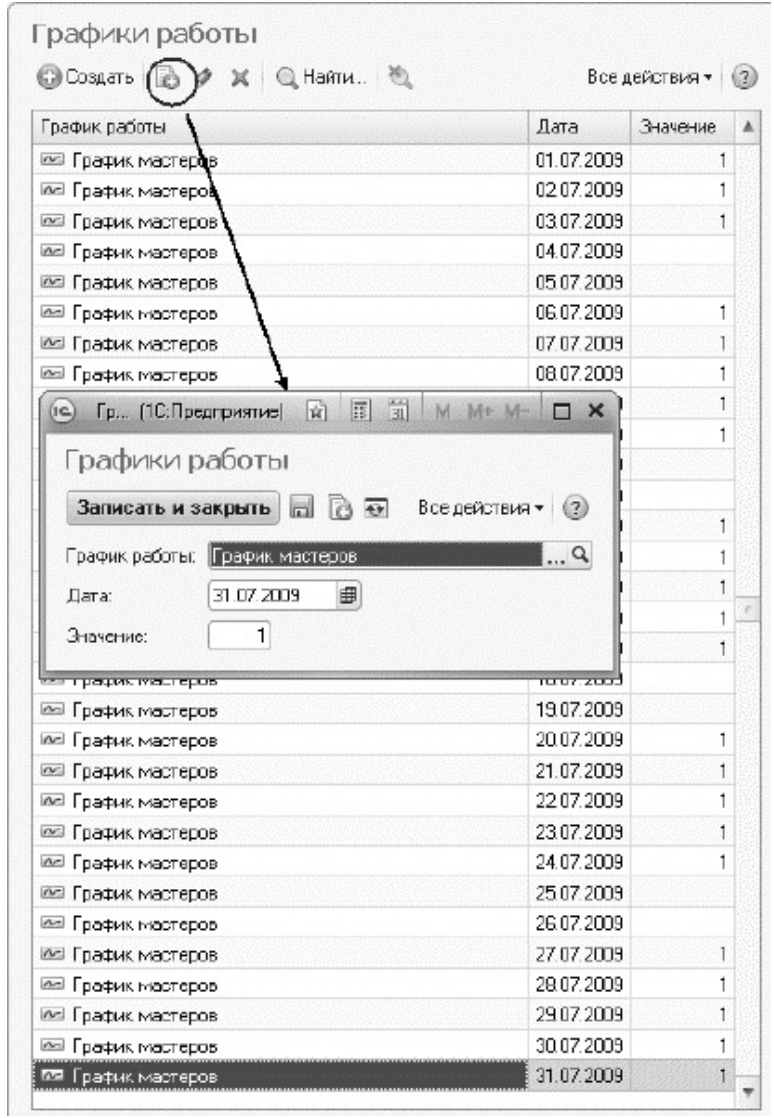

Графики работы
Создать
Найти...
Все действия
График работы
Дата
Значение
График мастеров 01.07.2009 1
График мастеров 02.07.2009 1
График мастеров 03.07.2009 1
График мастеров 04.07.2009
График мастеров 05.07.2009
График мастеров 06.07.2009 1
График мастеров 07.07.2009 1
График мастеров 08.07.2009 1
Гр... (1С:Предприятие)
Графики работы
Записать и закрыть
Все действия
График работы: График мастеров
Дата: 31.07.2009
Значение: 1
График мастеров 19.07.2009
График мастеров 20.07.2009 1
График мастеров 21.07.2009 1
График мастеров 22.07.2009 1
График мастеров 23.07.2009 1
График мастеров 24.07.2009 1
График мастеров 25.07.2009
График мастеров 26.07.2009
График мастеров 27.07.2009 1
График мастеров 28.07.2009 1
График мастеров 29.07.2009 1
График мастеров 30.07.2009 1
График мастеров 31.07.2009 1

Рис. 17.12. Записи регистра «Графики работы»

Теперь все готово для создания регистра расчета.

В режиме «Конфигуратор»

Добавим новый объект конфигурации *Регистр расчета* с именем *Начисления*. Зададим *Расширенное представление списка* как *Движения в регистре Начисления*. В качестве плана видов расчета, используемого регистром, выберем *ОсновныеНачисления*.

Установим, что регистр будет использовать период действия, график будет задаваться в регистре сведений *ГрафикиРаботы*, значение графика будет находиться в ресурсе *Значение*, а дата графика – в измерении *Дата*.

Укажем, что регистр расчета будет использовать базовый период и периодичность регистра будет *Месяц* (рис. 17.13).

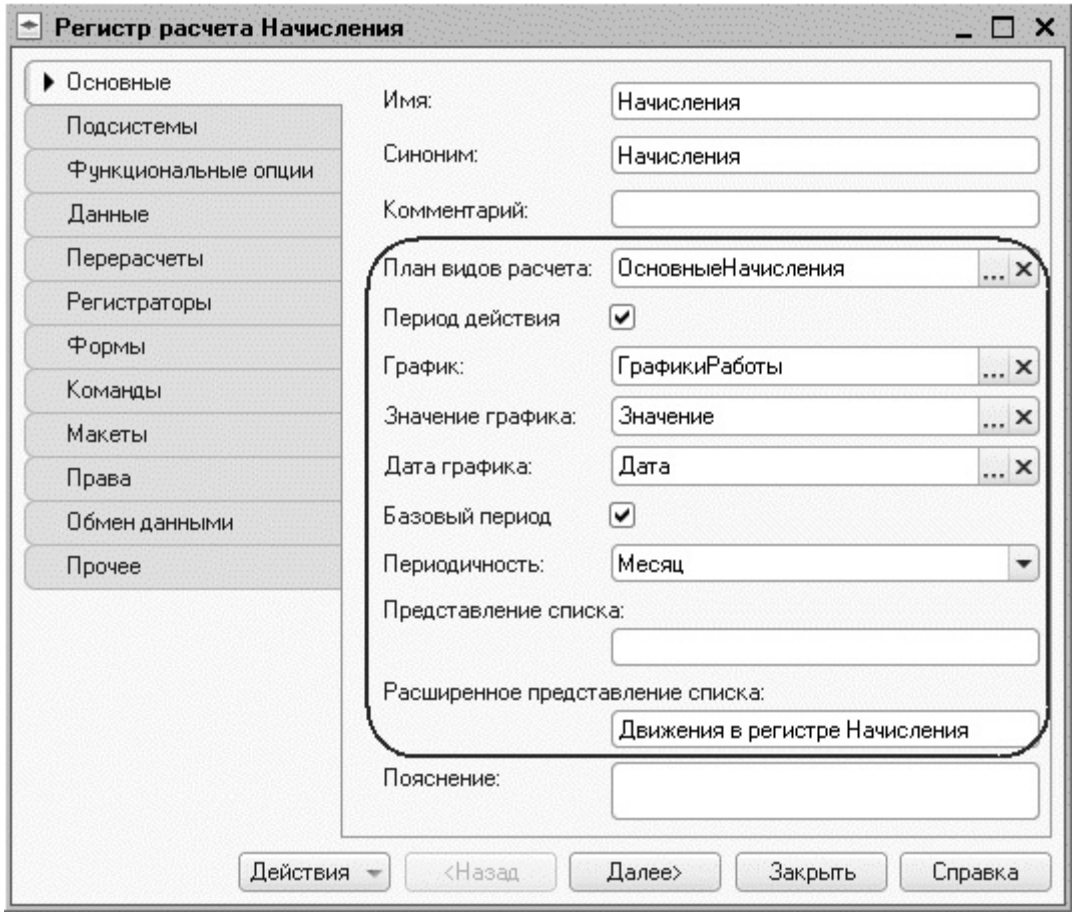


Рис. 17.13. Окно редактирования регистра расчета «Начисления»

На закладке *Подсистемы* укажем, что регистр расчета будет отображаться в подсистеме *РасчетЗарплаты*.

Затем перейдем на закладку *Данные* и создадим (рис. 17.14):

- измерение *Сотрудник*, тип *СправочникСсылка.Сотрудники*, *Базовое*;
- ресурс *Результат*, тип *Число*, длина *15*, точность *2*,
- реквизит *ГрафикРаботы*, тип *СправочникСсылка.ВидыГрафиковРаботы*, в разделе свойств *Данные* зададим свойство *Связь с графиком* по измерению *ГрафикРаботы*;
- реквизит *ИсходныеДанные*, тип *Число*, длина *15*, точность *2*.

Реквизит *ГрафикРаботы* мы будем использовать для того, чтобы связать запись регистра с используемым графиком работы, а реквизит *ИсходныеДанные* – чтобы хранить в нем данные, которые могут понадобиться при расчете или перерасчете (в нашем примере это будет расчет оклада).

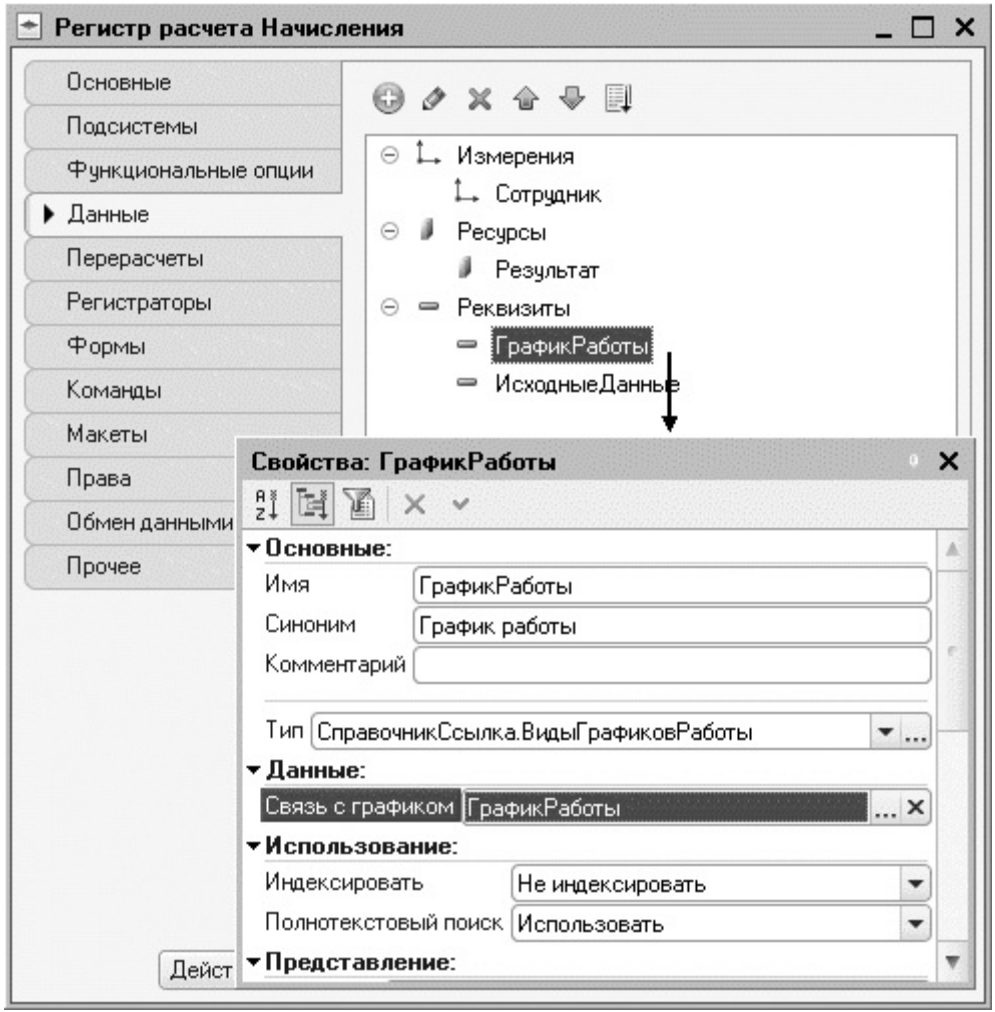


Рис. 17.14. Измерения, ресурсы и реквизиты регистра расчета

Теперь перейдем на закладку *Перерасчеты*. Создадим объект конфигурации *Перерасчет*, который так и назовем – *Перерасчет*.

У него будет единственное измерение – *Сотрудник*, для которого в разделе *Связь* мы укажем:

- измерение регистра – *Сотрудник*;
- данные ведущих регистров – выберем то же самое измерение *Сотрудник* регистра расчета *Начисления* (рис. 17.15).

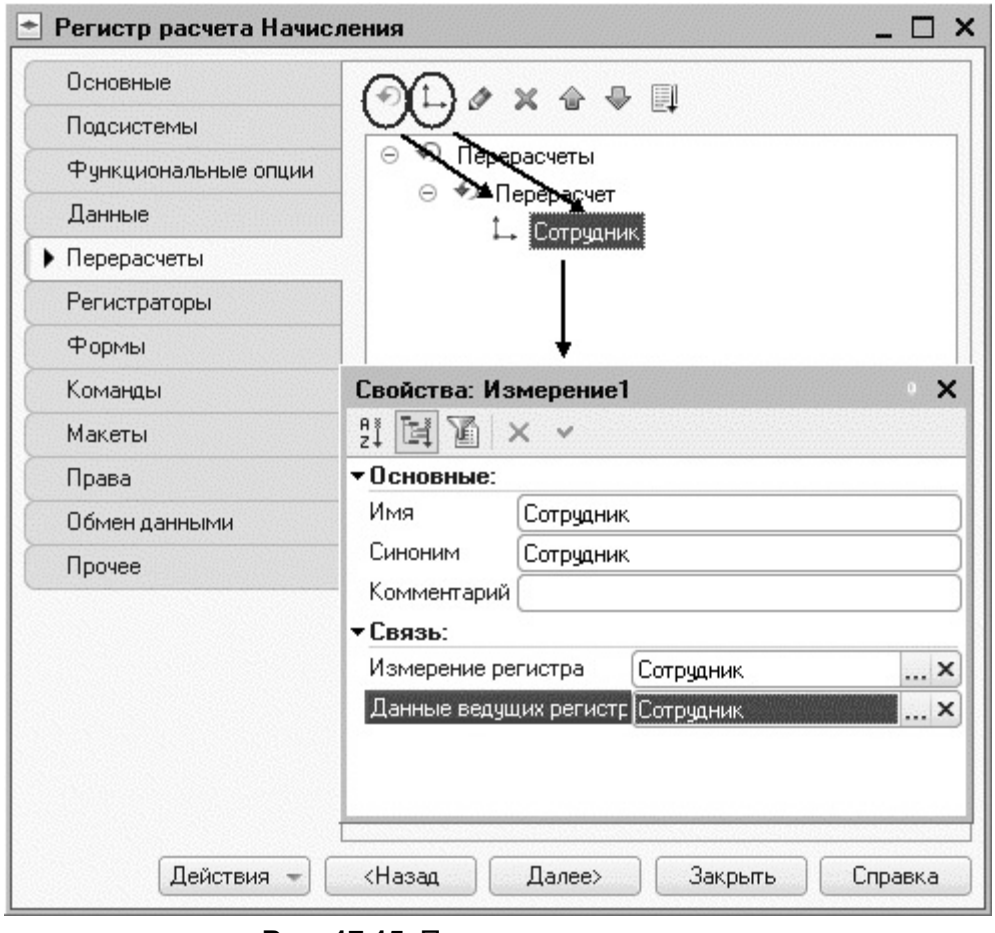


Рис. 17.15. Перерасчеты регистра

В заключение отредактируем командный интерфейс, чтобы в подсистеме *Расчет зарплаты* была доступна команда для просмотра записей регистра

расчета.

Для этого в дереве объектов конфигурации выделим подсистему *РасчетЗарплаты*, вызовем контекстное меню и выполним команду *Открыть командный интерфейс*.

В открывшемся окне командного интерфейса подсистемы в группе *Панель навигации.Обычное* включим видимость у команды *Начисления*.

На этом создание объекта конфигурации *Регистр расчета Начисления* завершено.

Контрольные вопросы

- *Что такое сложные периодические расчеты.*
- *Что такое вид расчета, база.*
- *Какая разница между базовым периодом, фактическим периодом и периодом действия.*
- *Что такое зависимость по базовому периоду.*
- *Что такое вытеснение по периоду действия.*
- *Для чего предназначен объект конфигурации «План видов расчета».*

- *Каковы основные свойства плана видов расчета.*
- *Какая разница между базовыми, вытесняющими и ведущими видами расчетов.*
- *Как создать план видов расчета.*
- *Что такое объект конфигурации «Регистр расчета».*
- *Каковы отличительные особенности регистра расчета.*
- *Что такое график времени.*
- *Что такое перерасчет.*
- *По какому принципу формируются записи перерасчета.*
- *Как создать регистр расчета.*

Занятие 18 (3:40). Использование регистра расчета

Продолжительность

Ориентировочная продолжительность занятия – 3 часа 40 минут.

Теперь у нас все готово для того, чтобы начать разработку системы расчета заработной платы ООО «На все руки мастер».

В этой главе мы создадим документ, с помощью которого будут выполняться различные виды начислений, посмотрим, как и когда платформа формирует записи перерасчета, увидим, как работают механизмы вытеснения по периоду действия и зависимости по базовому периоду.

Кроме этого, мы создадим отчет, показывающий начисления сотрудникам ООО «На все руки мастер», и сделаем так, чтобы данные расчетов можно было поддерживать в актуальном состоянии.

В заключение мы познакомимся с новым элементом формы – *Диаграмма Ганта* – и с его помощью наглядно проиллюстрируем работу некоторых механизмов расчета.

Добавление документа о начислениях

Для того чтобы иметь возможность регистрировать в базе данных начисления, производимые сотрудниками ООО «На все руки мастер», нам понадобится специальный документ.

В режиме «Конфигуратор»

Откроем конфигуратор и добавим новый объект конфигурации *Документ*. Назовем его *НачислениеСотрудникам*. Зададим представление объекта как *Начисление сотрудникам*. На закладке *Нумерация* установим:

- *Тип номера – Число*,
- *Длина номера – 5* (рис. 18.1).

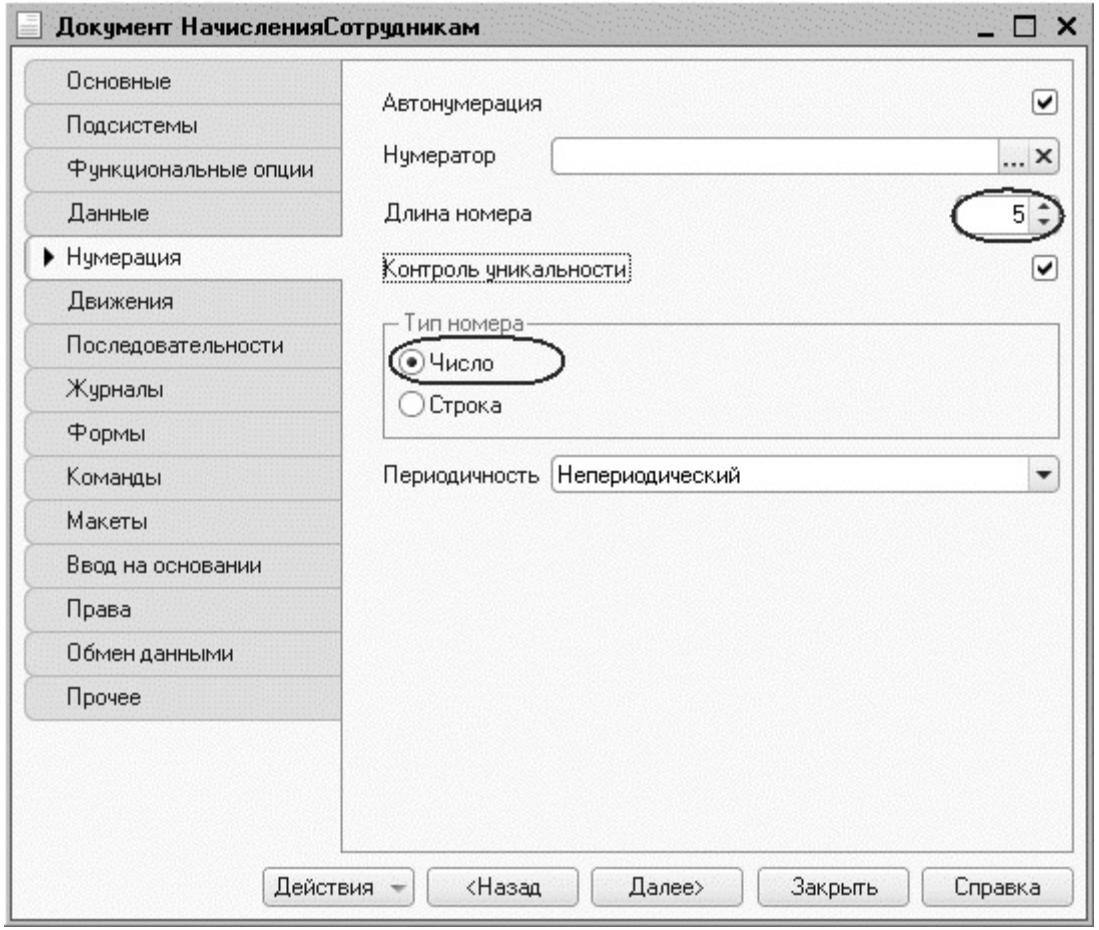


Рис. 18.1. Нумерация документа

На закладке *Подсистемы* укажем, что документ будет отображаться в подсистеме *РасчетЗарплаты*.

На закладке *Данные* укажем, что этот документ будет иметь табличную часть *Начисления*, содержащую следующие реквизиты:

- *Сотрудник*, тип *СправочникСсылка.Сотрудники*;
- *ГрафикРаботы*, тип *СправочникСсылка.ВидыГрафиковРаботы*;
- *ДатаНачала*, тип *Дата*;
- *ДатаОкончания*, тип *Дата*;
- *ВидРасчета*, тип *ПланВидовРасчетаСсылка.ОсновныеНачисления*;
- *Начислено*, *Число*, длина *15*, точность *2*.

Реквизиты *ДатаНачала* и *ДатаОкончания* понадобятся нам для того, чтобы задавать период, в котором должна действовать запись расчета.

На закладке *Движения* запретим оперативное проведение документа.

Отметим, что документ будет создавать движения по регистру расчета *Начисления,* и запустим конструктор движений (рис. 18.2).

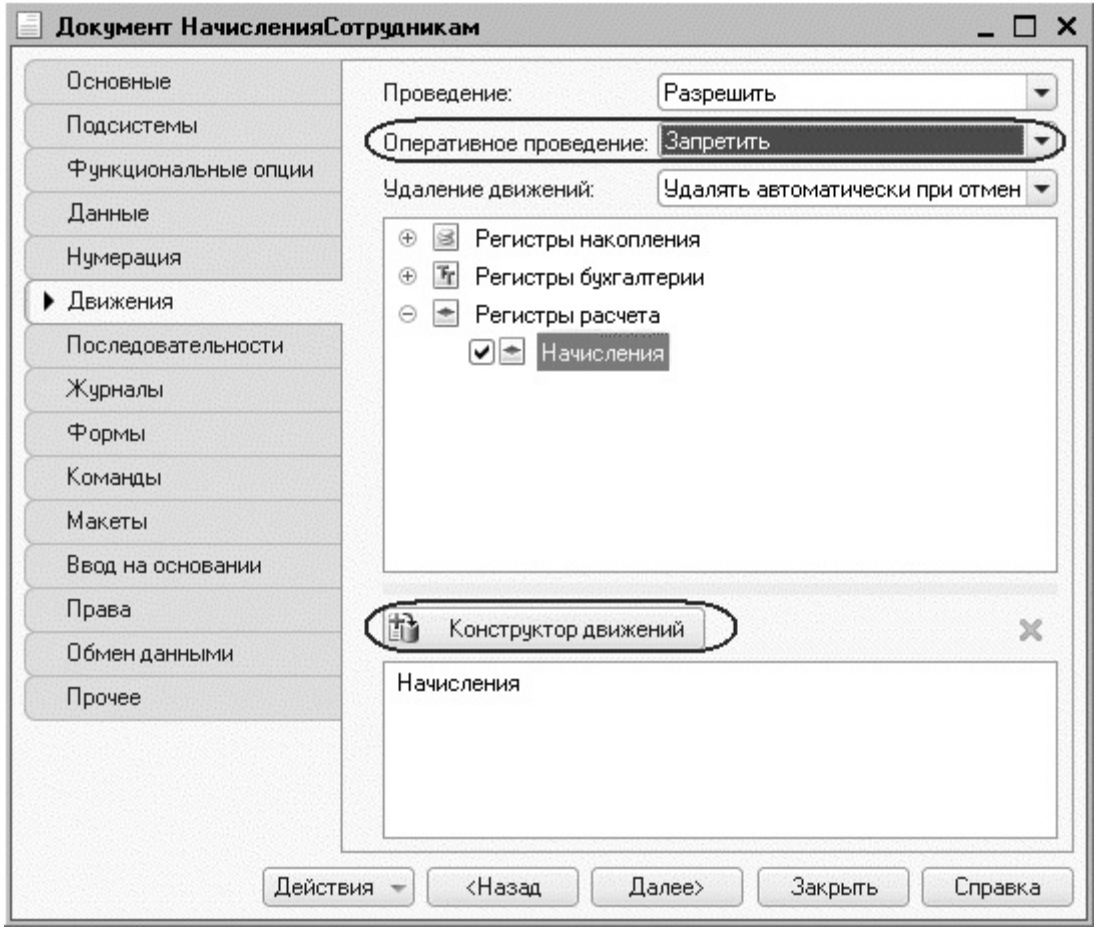


Рис. 18.2. Движения документа

В окне конструктора выберем табличную часть *Начисления* и нажмем *Заполнить выражения*.

Для реквизитов *ПериодДействияКонец* и *БазовыйПериодКонец* укажем выражение *КонецДня(ТекСтрокаНачисления.ДатаОкончания)*.

Для поля *ПериодРегистрации* укажем выражение *Дата.*

Реквизиту *ИсходныеДанные* поставим в соответствие реквизит табличной части *Начислено*, а для ресурса *Результат* оставим пустое выражение, так как мы будем его потом рассчитывать (рис. 18.3).

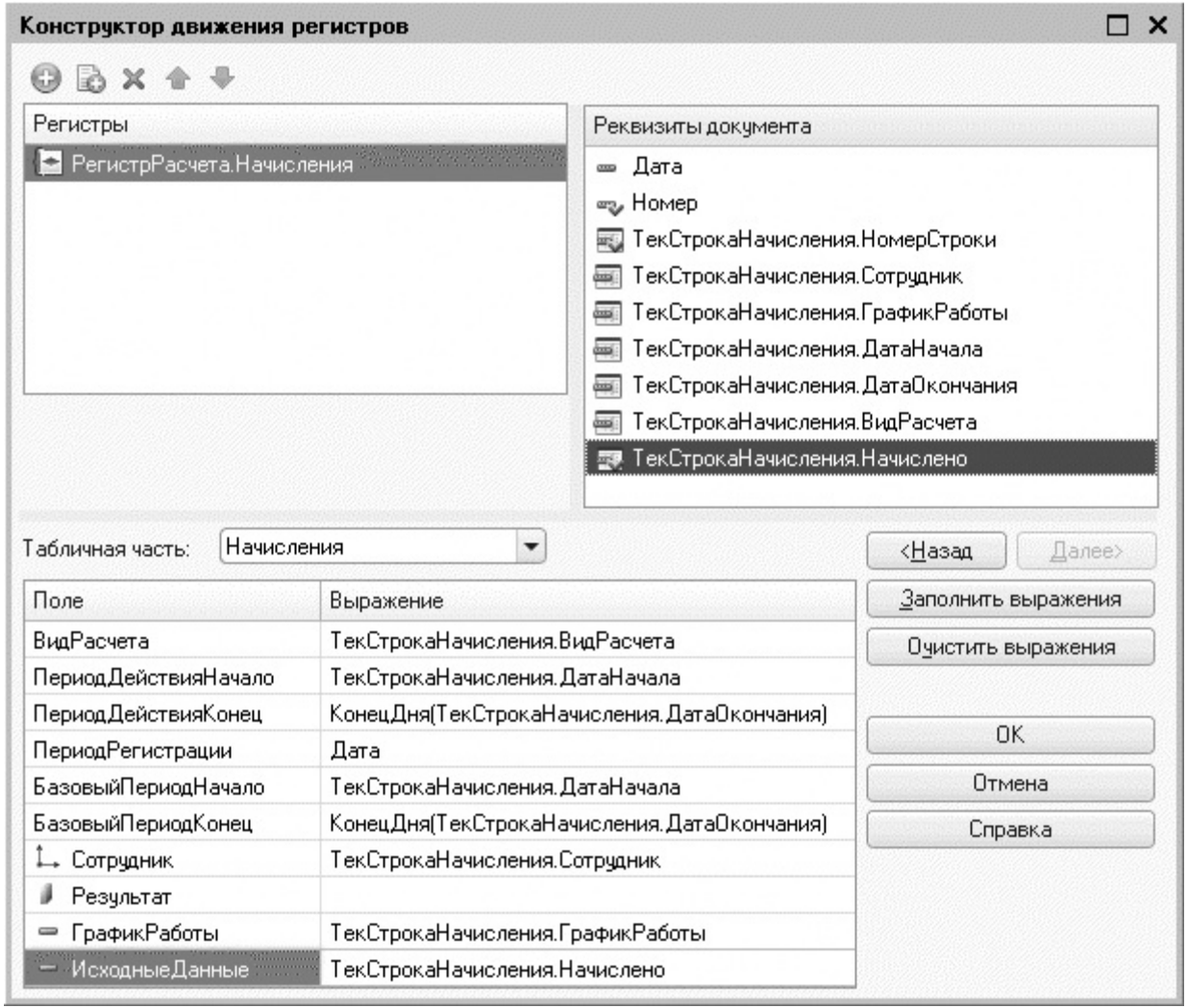


Рис. 18.3. Движения документа «НачисленияСотрудникам» по регистру расчета

Нажмем *OK* и посмотрим текст обработчика, созданный конструктором (листинг 18.1).

Листинг 18.1. Текст обработчика, созданный конструктором движений

```
Процедура ОбработкаПроведения(Отказ, Режим)

  //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
  // Данный фрагмент построен конструктором
  // При повторном использовании конструктора
  // внесенные вручную изменения будут утеряны!!!
  Движения.Начисления.Записывать = Истина;

  Для Каждого ТекСтрокаНачисления Из Начисления Цикл

    // Регистр Начисления
    Движение = Движения.Начисления.Добавить();
    Движение.Сторно = Ложь;
    Движение.ВидРасчета = ТекСтрокаНачисления.ВидРасчета;
    Движение.ПериодДействияНачало = ТекСтрокаНачисления.ДатаНачала;
    Движение.ПериодДействияКонец = КонецДня(ТекСтрокаНачисления.ДатаОкончания);
    Движение.ПериодРегистрации = Дата;
    Движение.БазовыйПериодНачало = ТекСтрокаНачисления.ДатаНачала;
    Движение.БазовыйПериодКонец = КонецДня(ТекСтрокаНачисления.ДатаОкончания);
    Движение.Сотрудник = ТекСтрокаНачисления.Сотрудник;
    Движение.ГрафикРаботы = ТекСтрокаНачисления.ГрафикРаботы;
    Движение.ИсходныеДанные = ТекСтрокаНачисления.Начислено;

  КонецЦикла;
  //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
```

```
КонецПроцедуры
```

Прокомментируем этот код.

В целом в нем нет ничего принципиально нового по сравнению с прежними обработчиками проведения документов, создающими движения регистров.

Напомним лишь, что *Начисления* – имя регистра расчета, и в выражениях *Движения.Начисления.Записывать()* и *Движения.Начисления.Добавить()* мы обращаемся к методам набора записей этого регистра.

В строке *Для Каждого ТекСтрокаНачисления Из Начисления Цикл* мы обращаемся по имени (*Начисления*) к табличной части нашего документа и организуем цикл обхода этой табличной части.

В цикле мы добавляем к движениям новую запись и присваиваем ее полям значения из табличной части документа.

Для присвоения значений полям *ПериодДействияКонец, БазовыйПериодКонец* мы используем функцию *КонецДня()*. Аналогичным образом мы поступали при создании отчетов, чтобы последний день периода попал в отчет.

В заключение отредактируем командный интерфейс, чтобы в подсистеме *Расчет зарплаты* была доступна команда создания новых документов.

Для этого в дереве объектов конфигурации выделим ветвь *Подсистемы*, вызовем ее контекстное меню и выберем пункт *Все подсистемы*.

В открывшемся окне слева в списке *Подсистемы* выделим подсистему *РасчетЗарплаты*.

В группе *Панель действий.Создать* включим видимость у команды *Начисление сотрудникам: создать* (рис. 18.4).

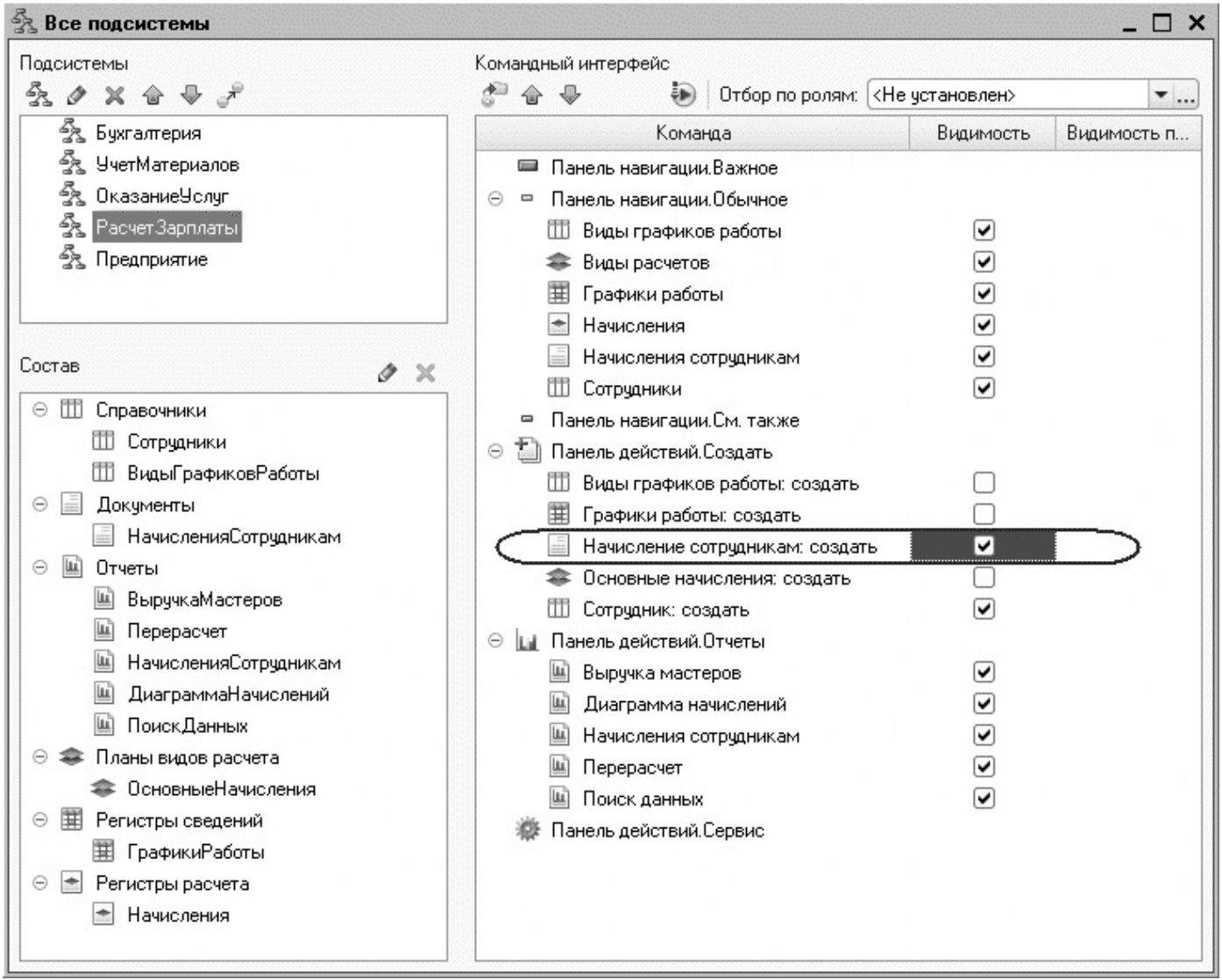

Все подсистемы
Подсистемы
Бухгалтерия
УчетМатериалов
ОказаниеУслуг
РасчетЗарплаты
Предприятие
Состав
Справочники
Сотрудники
ВидыГрафиковРаботы
Документы
НачислениеСотрудникам
Отчеты
ВыручкаМастеров
Перерасчет
НачисленияСотрудникам
ДиаграммаНачислений
ПоискДанных
Планы видов расчета
ОсновныеНачисления
Регистры сведений
ГрафикиРаботы
Регистры расчета
Начисления
Командный интерфейс
Отбор по ролям:
<Не установлен>
Команда
Видимость
Видимость п...
Панель навигации.Важное
Панель навигации.Обычное
Виды графиков работы
Виды расчетов
Графики работы
Начисления
Начисления сотрудникам
Сотрудники
Панель навигации.См. также
Панель действий.Создать
Виды графиков работы: создать
Графики работы: создать
Начисление сотрудникам: создать
Основные начисления: создать
Сотрудник: создать
Панель действий.Отчеты
Выручка мастеров
Диаграмма начислений
Начисления сотрудникам
Перерасчет
Поиск данных
Панель действий.Сервис

Рис. 18.4. Редактирование командного интерфейса

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим, как работает наш документ.

В панели действий раздела *Расчет зарплаты* выполним команду *Начисление сотрудникам* и начислим оклад за июль всем сотрудникам ООО «На все руки мастер» (рис. 18.5).

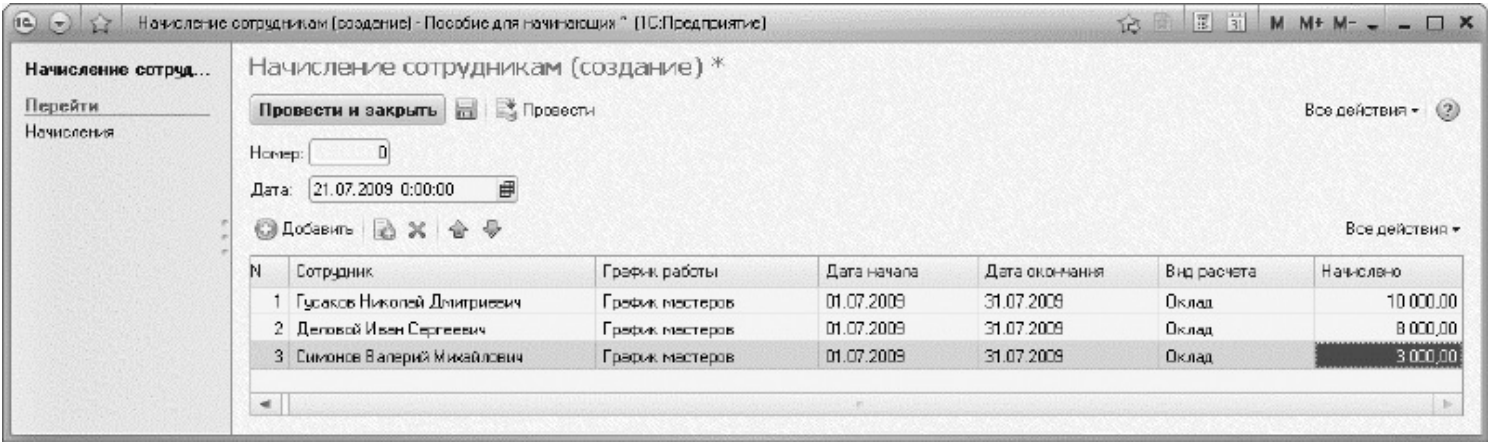


Рис. 18.5. Документ «Начисления сотрудникам № 1»

Проведем документ и посмотрим, какие движения он сформировал в регистре *Начисления* (рис. 18.6, 18.7).

Пери...	Регистратор	Ном...	Вид расчета	Сторно	Сотрудник	Результат	График работы	Исходные данные
01.07.2009...	Начисление сотрудни...	1	Оклад		Гусаков Николай Дмитриевич		График мастеров	10 000,00
01.07.2009...	Начисление сотрудни...	2	Оклад		Деловой Иван Сергеевич		График мастеров	8 000,00
01.07.2009...	Начисление сотрудни...	3	Оклад		Симонов Валерий Михайлович		График мастеров	3 000,00

Рис. 18.6. Движения документа «Начисление сотрудникам № 1» в регистре расчета «Начисления»

Период действия	Дата начала периода действия	Дата окончания периода действия	Дата начала базового периода	Дата окончания базового периода
01.07.2009 0:00:00	01.07.2009 0:00:00	31.07.2009 23:59:59	01.07.2009 0:00:00	31.07.2009 23:59:59
01.07.2009 0:00:00	01.07.2009 0:00:00	31.07.2009 23:59:59	01.07.2009 0:00:00	31.07.2009 23:59:59
01.07.2009 0:00:00	01.07.2009 0:00:00	31.07.2009 23:59:59	01.07.2009 0:00:00	31.07.2009 23:59:59

Рис. 18.7. Движения документа «Начисление сотрудникам № 1» в регистре расчета «Начисления»

Обратите внимание на то, что платформа привела период регистрации каждой записи к началу периода регистра расчета (в обработчике проведения мы указывали значение даты документа – *21.07.2009*).

Кроме этого, заметьте, что в каждой записи мы сохранили в реквизите *ИсходныеДанные* размер оклада сотрудника, введенный в документе, чтобы в дальнейшем рассчитать сумму оплаты по окладу.

Для дальнейшего изучения работы регистра расчета нам понадобится

служебный отчет, с помощью которого мы сможем посмотреть содержимое записей перерасчета.

Иллюстрация механизмов вытеснения и зависимости от базы

Отчет по перерасчетам

В режиме «Конфигуратор»

Создадим новый объект конфигурации *Отчет*. Назовем его *Перерасчет*. Создадим основную схему компоновки данных, добавим источник данных – запрос и откроем конструктор запроса.

В списке *База данных* раскроем ветвь *Перерасчеты* и из виртуальной таблицы перерасчета *Начисления.Перерасчет* выберем все поля:

- *ОбъектПерерасчета*,
- *ВидРасчета,*
- *Сотрудник* (рис. 18.8).

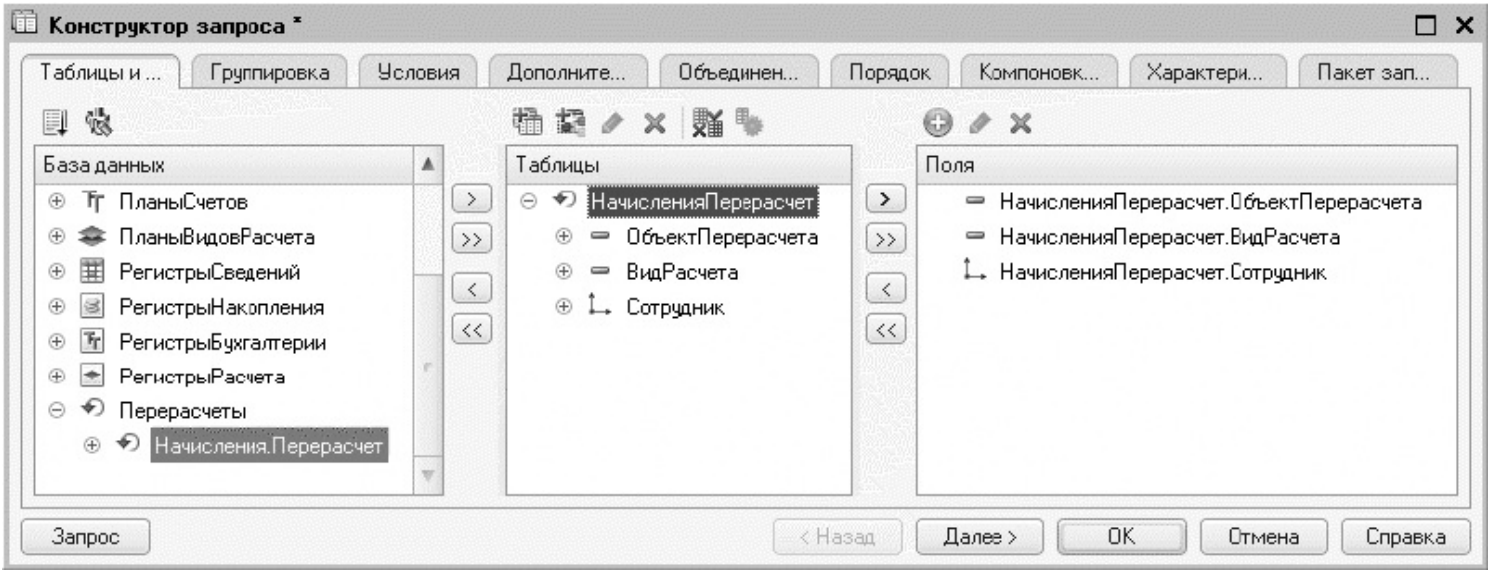


Рис. 18.8. Поля и таблицы запроса

На этом создание запроса закончено, нажмем *OK*.

Перейдем на закладку *Настройки* и добавим группировку детальных записей. На закладке *Выбранные поля* выберем для вывода в отчет поля *ОбъектПерерасчета*, *ВидРасчета* и *Сотрудник*.

На этом создание схемы компоновки данных закончено, закроем ее.

В окне редактирования объекта конфигурации *Отчет Перерасчет* на

закладке *Подсистемы* укажем, что отчет будет принадлежать подсистеме *РасчетЗарплаты*.

Зависимость по базовому периоду

В режиме «1С:Предприятие»

Если сейчас мы выполним отчет в режиме *1С:Предприятие*, то мы увидим, что ни один перерасчет еще не выполнялся.

Поэтому создадим новый документ *Начисление сотрудникам № 2*, в котором начислим премию за июль Гусакову и Деловому (рис. 18.9).

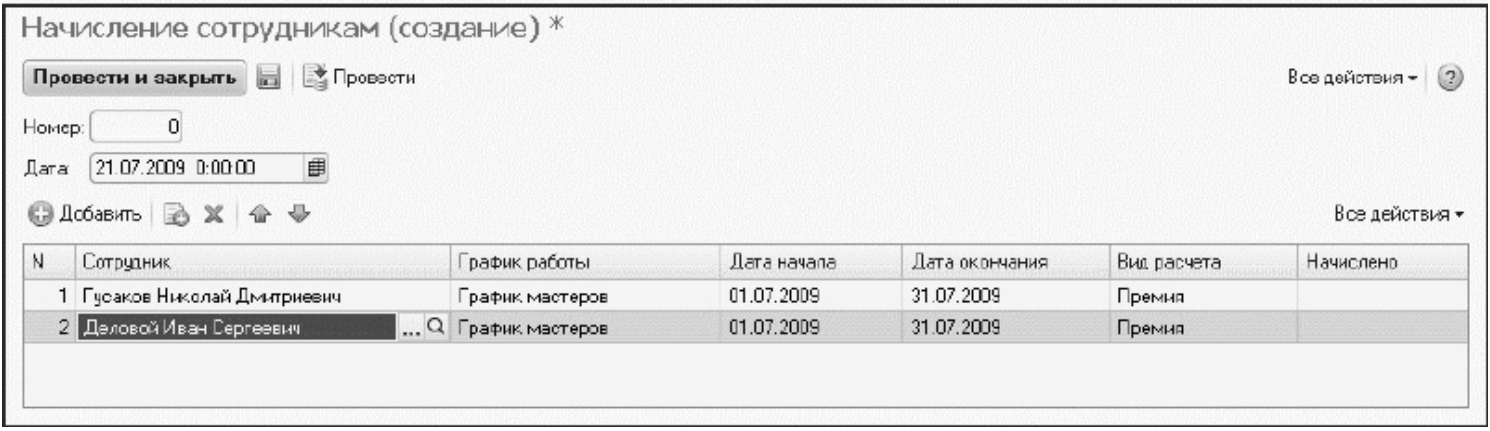


Рис. 18.9. Документ «Начисления сотрудникам № 2»

Этим документом мы зафиксируем тот факт, что сотрудникам Гусакову и

Деловому нужно начислить премию по итогам работы за июль. Поскольку размер премии нам неизвестен (он будет рассчитываться по некоторому алгоритму), поле *Начислено* мы оставляем пустым. Нажмем *Провести и закрыть*.

Теперь снова откроем документ *Начисление сотрудникам № 1* и изменим оклад Гусакова с 10 000 на 7 000. Нажмем *Провести и закрыть*.

Сформируем отчет *Перерасчет* (рис. 18.10).

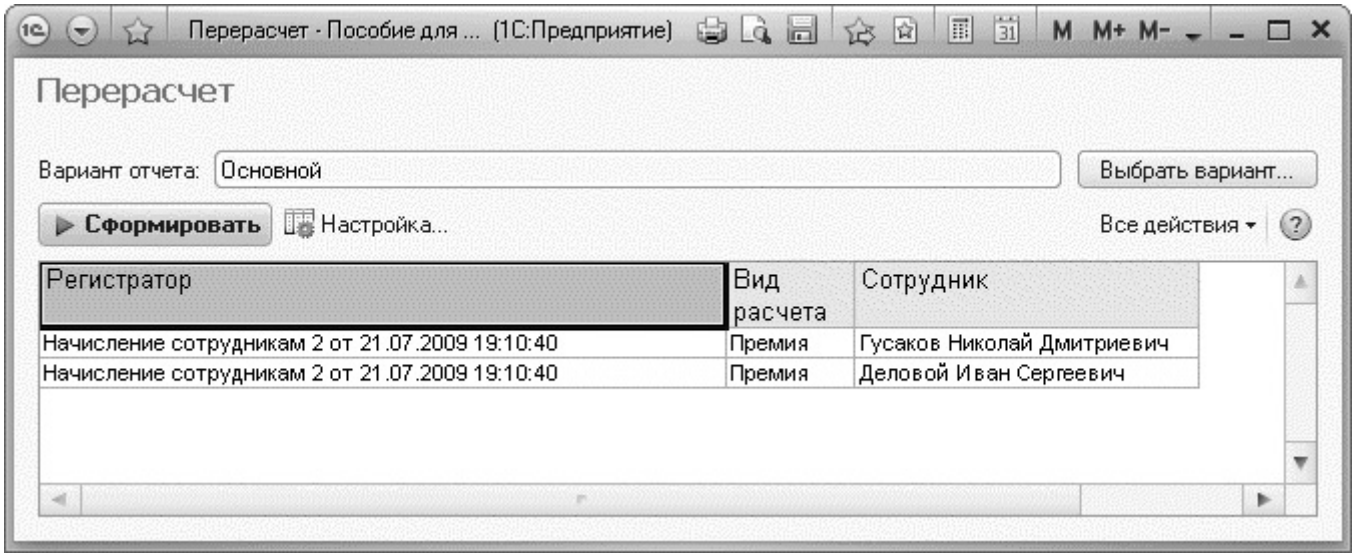


Рис. 18.10. Отчет «Перерасчет»

Как видите, отчет теперь содержит какие-то данные.

В самом деле вид расчета *Премия* зависит у нас по базовому периоду от вида расчета *Оклад*. Как только мы изменили существовавшие в регистре записи по виду расчета *Оклад,* платформа сразу же сформировала набор записей перерасчета, которые должны быть рассчитаны заново, так как изменилась их база.

Вы можете спросить: почему в перерасчет попали записи как про Делового, так и про Гусакова, хотя оклад мы меняли только Гусакову?

Дело в том, что платформа не отслеживает конкретные изменения, которые пользователь внес в записи документа. Она отслеживает лишь факт изменения набора записей регистра расчета в результате проведения (перепроведения) документа.

Поэтому в набор записей перерасчета она включает информацию обо ВСЕХ записях регистра, значение ресурсов которых МОЖЕТ измениться в результате перепроведения документа, создавшего базовые записи регистра.

Перепроведем документ *Начисления сотрудникам № 2* (которым мы начисляли премию) и сформируем отчет *Перерасчет*.

Он снова не содержит никаких данных – система отметила тот факт, что мы «пересчитали» зависимые записи, и очистила таблицу перерасчета.

На этом примере мы с вами познакомились с работой механизма поддержки зависимости по базовому периоду у регистра расчета.

Вытеснение по периоду действия

В режиме «1С:Предприятие»

Теперь посмотрим, как работает механизм вытеснения по периоду действия.

Для этого нам понадобится создать документ *Начисления сотрудникам № 3* (рис. 18.11).

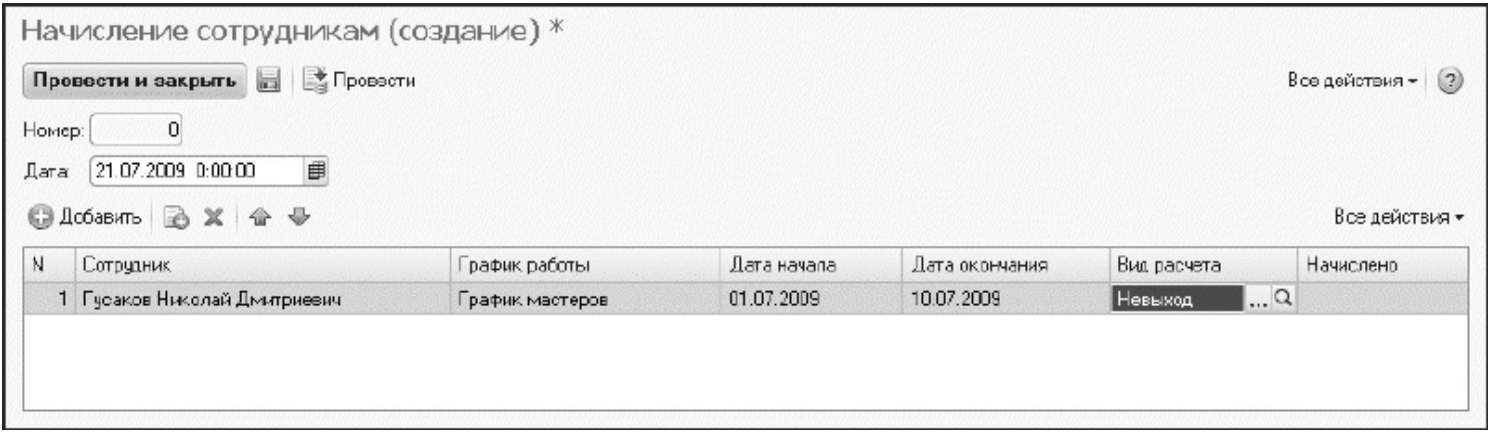


Рис. 18.11. Документ «Начисления сотрудникам № 3»

Этим документом мы зафиксируем тот факт, что Гусаков не выходил на работу с 1 по 10 июля.

Очевидно, что в этом случае потребуется пересчитать его оплату по окладу и как следствие начисленную премию.

Нажмем *Провести и закрыть* и затем сформируем отчет *Перерасчет* (рис. 18.12).

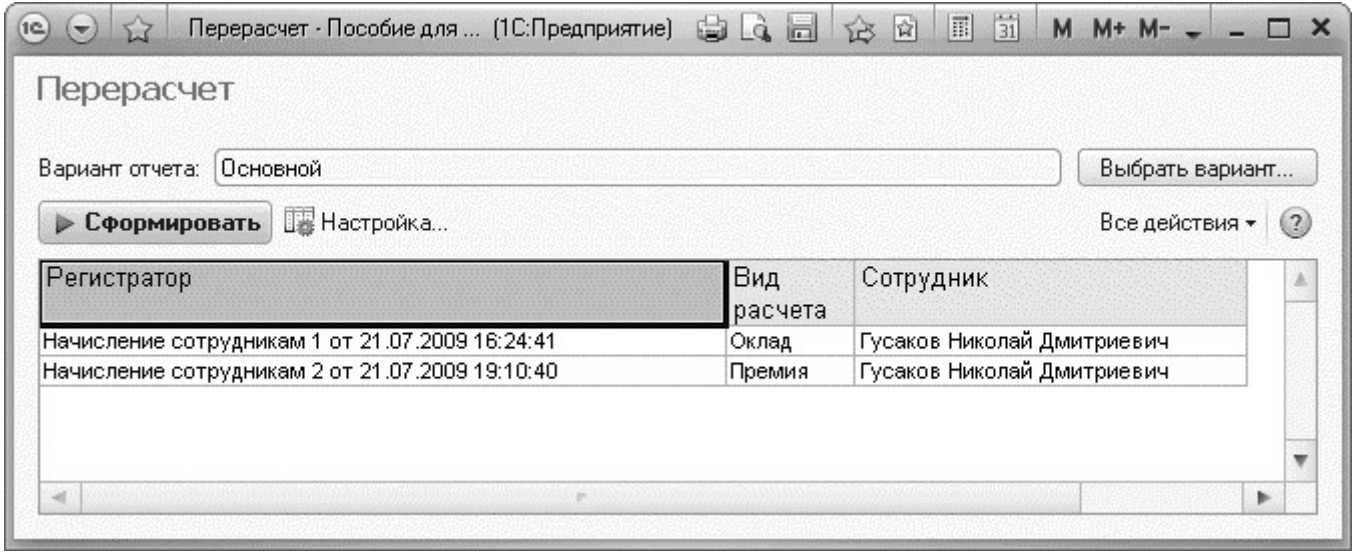


Рис. 18.12. Отчет «Перерасчет»

Как вы видите, в перерасчет попала запись о начислении оклада Гусакову. Это явилось результатом работы механизма вытеснения по периоду действия, ведь вид расчета *Невыход* вытесняет у нас вид расчета *Оклад*.

Обратите внимание, что в перерасчет попала и запись о начислении премии Гусакову.

Если вы помните, при создании предопределенных видов расчета мы указали, что результат вида расчета *Премия* будет зависеть от изменения результата вида расчета *Невыход*. Эта зависимость косвенная, но поскольку явно указали такую зависимость, платформа ее отследила.

Перепроведем документы *Начисления сотрудникам* № 1 и № 2 и убедимся, что таблица перерасчета очистилась.

Процедура расчета записей регистра расчета

В режиме «Конфигуратор»

До сих пор мы с вами просто заносили в регистр расчета *Начисления* записи о том, что необходимо выполнить какой-либо вид расчета. Но каким именно образом получать эти результаты, мы не говорили.

Теперь настало время описать алгоритмы формирования различных видов расчетов.

Поскольку эти алгоритмы нужно будет использовать не только в документе *Начисление сотрудникам*, удобнее всего будет разместить их в отдельном общем модуле.

Откроем в конфигураторе текст обработчика проведения документа *НачислениеСотрудникам* и добавим в него после завершения создания движений в регистре *Начисления* вызов процедуры *РассчитатьНачисления()* из общего модуля *ПроведениеРасчетов* (листинг 18.2).

Листинг 18.2. Обработчик проведения документа «НачислениеСотрудникам»

```
Процедура ОбработкаПроведения(Отказ, Режим)
  …
  КонецЦикла;
  //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

  // Записываем движения регистров
  Движения.Начисления.Записать();

  // Получим список всех сотрудников, содержащихся в документе
  Запрос = Новый Запрос(
  "ВЫБРАТЬ РАЗЛИЧНЫЕ
  |        НачислениеСотрудникамНачисления.Сотрудник
  |ИЗ
```

```
    |        Документ.НачисленияСотрудникам.Начисления
    | КАК НачисленияСотрудникамНачисления
    |
    |ГДЕ
    |        НачисленияСотрудникамНачисления.Ссылка = &ТекущийДокумент");

    Запрос.УстановитьПараметр("ТекущийДокумент", Ссылка);

    // Сформируем список сотрудников
    ТаблЗнач = Запрос.Выполнить().Выгрузить();
    МассивСотрудников = ТаблЗнач.ВыгрузитьКолонку("Сотрудник");

    // Вызов процедуры РассчитатьНачисления из общего модуля
    ПроведениеРасчетов.РассчитатьНачисления(Движения.Начисления,
    ПланыВидовРасчета.ОсновныеНачисления.Оклад, МассивСотрудников);
    Движения.Начисления.Записать( , Истина);

    ПроведениеРасчетов.РассчитатьНачисления(Движения.Начисления,
    ПланыВидовРасчета.ОсновныеНачисления.Премия, МассивСотрудников);
    Движения.Начисления.Записать( , Истина);

КонецПроцедуры
```

Обратите внимание: при проведении документа мы сначала записываем движения, сформированные документом, в регистр (*Движения.Начисления.Записать()*), а затем передаем этот набор записей регистра в процедуру расчета *РассчитатьНачисления()*, которую мы создадим в общем модуле *ПроведениеРасчетов*.

Эту процедуру мы вызываем сначала для расчета первичных записей (*Оклад*), а затем для расчета вторичных (*Премия*).

Процедура расчета на основе описанных в ней алгоритмов и данных, содержащихся в записях регистра, должна сформировать значения ресурсов регистра.

После того как ресурсы будут рассчитаны, мы перезаписываем набор записей регистра без формирования записей перерасчета (второй параметр в методе *Записать() – Истина*).

Перед вызовом процедуры из общего модуля мы с помощью запроса формируем массив сотрудников, перечисленных в документе, чтобы передать его в вызываемую процедуру.

Для параметра запроса *ТекущийДокумент* устанавливаем значение стандартного реквизита документа – *Ссылка*. Используя метод запроса *Запрос.Выполнить().Выгрузить()* выгружаем результат запроса в таблицу значений (переменную *ТаблЗнач*). Затем формируем массив *МассивСотрудников*, содержащий колонку *Сотрудник* из этой таблицы значений.

Теперь создадим в ветке *Общие* новый общий модуль *ПроведениеРасчетов*.

Установим флажок *Вызов сервера* для видимости его экспортных процедур и функций (рис. 18.13).

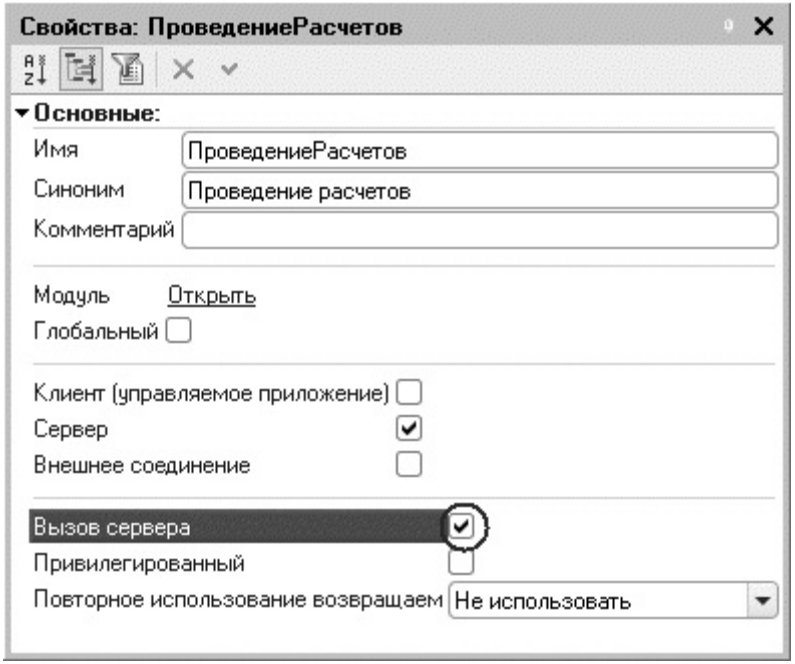


Рис. 18.13. Свойства общего модуля

Добавим в него заготовку процедуры *РассчитатьНачисления* (листинг 18.3).

Листинг 18.3. Заготовка процедуры «РассчитатьНачисления»

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
```

```
СписокСотрудников) Экспорт

   Регистратор = НаборЗаписейРегистра.Отбор.Регистратор.Значение;

   // Рассчитать первичные записи
   Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад Тогда

   // Рассчитать вторичные записи
   ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
   Тогда

   КонецЕсли;

КонецПроцедуры
```

Алгоритм расчета начислений будет различным при расчете первичных (вид расчета – *Оклад*) и вторичных записей (вид расчета – *Премия*), и каждая из его частей будет находиться в своей ветке условия *Если...*

При расчете первичных записей нам понадобятся данные графика из регистра расчета, поэтому добавим в первую ветку условия запроса по виртуальной таблице регистра расчета *РегистрРасчета.Начисления.ДанныеГрафика* (листинг 18.4).

Листинг 18.4. Изменение процедуры «РассчитатьНачисления»

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
```

```
СписокСотрудников) Экспорт

  Регистратор = НаборЗаписейРегистра.Отбор.Регистратор.Значение;

  // Рассчитать первичные записи
  Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад Тогда

    Запрос = Новый Запрос;
    Запрос.Текст =
      "ВЫБРАТЬ
      |        НачисленияДанныеГрафика.ЗначениеПериодДействия КАК Норма,
      |        НачисленияДанныеГрафика.ЗначениеФактическийПериодДействия КАК
      Факт,
      |        НачисленияДанныеГрафика.НомерСтроки КАК НомерСтроки
      |ИЗ
      |        РегистрРасчета.Начисления.ДанныеГрафика(Регистратор =
      &Регистратор И
      |                  ВидРасчета = &ВидРасчета И Сотрудник В
      (&СписокСотрудников))
      |        КАК НачисленияДанныеГрафика";

    Запрос.УстановитьПараметр("Регистратор", Регистратор);
    Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);
    Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);

    ВыборкаРезультата = Запрос.Выполнить().Выбрать();

  // Рассчитать вторичные записи
  ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
  Тогда
```

```
  КонецЕсли;

КонецПроцедуры
```

В этом запросе мы выбираем из виртуальной таблицы данных графика регистра расчета значение графика для периода действия и для фактического периода действия. При задании параметров виртуальной таблицы мы ограничиваем выборку регистратором, нужным нам видом расчета и списком сотрудников, по которым нужно получить значения графика.

Теперь добавим обход переданного в процедуру набора записей и расчет записей, для которых получены значения графика (листинг 18.5).

Листинг 18.5. Добавление обхода набора записей и расчета первичных записей

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
СписокСотрудников) Эспорт

  Регистратор = НаборЗаписейРегистра.Отбор.Регистратор.Значение;

  // Рассчитать первичные записи
  Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад Тогда
    …
    ВыборкаРезультата = Запрос.Выполнить().Выбрать();

    Для Каждого ЗаписьРегистра Из НаборЗаписейРегистра Цикл
```

```
    СтруктураНомер = Новый Структура("НомерСтроки");
    СтруктураНомер.НомерСтроки = ЗаписьРегистра.НомерСтроки;
    ВыборкаРезультата.Сбросить();

    Если ВыборкаРезультата.НайтиСледующий(СтруктураНомер) Тогда

      Если ВыборкаРезультата.Норма = 0 Тогда
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "Вид расчета: Оклад - Нет рабочих дней в заданном
        периоде";
        Сообщение.Сообщить();
        ЗаписьРегистра.Результат = 0;

      Иначе

        // Рассчитать оклад по фактическому периоду и исходным данным
        ЗаписьРегистра.Результат = (ЗаписьРегистра.ИсходныеДанные
        /ВыборкаРезультата.Норма) * ВыборкаРезультата.Факт;
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "Выполнен расчет" + ЗаписьРегистра.Регистратор + "
        - " + ЗаписьРегистра.ВидРасчета + " - " + ЗаписьРегистра.Сотрудник;
        Сообщение.Сообщить();

      КонецЕсли;

    КонецЕсли;

  КонецЦикла;

// Рассчитать вторичные записи
```

```
    ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
    Тогда

    КонецЕсли;

КонецПроцедуры
```

Для каждой записи из набора записей регистра расчета мы получаем номер строки, идентифицирующий начисление для конкретного сотрудника, и по этому номеру ищем соответствующую запись в выборке из результата запроса.

Если в результате запроса есть запись с таким номером строки, мы рассчитываем результат записи регистра расчета. То есть мы получаем начисление по окладу для каждого сотрудника как результат от деления начисленной суммы (поле регистра *ИсходныеДанные*) на количество рабочих дней в месяце (*Норма*) и умножения на фактически отработанные рабочие дни (*Факт*).

Добавим текст запроса во вторую ветку условия *Если…* с той лишь разницей, что теперь мы будем получать значения базы, используя виртуальную таблицу регистра расчета *РегистрРасчета.Начисления.БазаНачисления* (листинг 18.6).

Листинг 18.6. Добавление текста запроса во вторую ветку условия

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
СписокСотрудников) Экспорт

  Регистратор = НаборЗаписейРегистра.Отбор.Регистратор.Значение;

  // Рассчитать первичные записи
  Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад Тогда
…
  // Рассчитать вторичные записи
  ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
  Тогда

    Запрос = Новый Запрос;
    Запрос.Текст =
      "ВЫБРАТЬ
      |        НачисленияБазаНачисления.РезультатБаза КАК База,
      |        НачисленияБазаНачисления.НомерСтроки КАК НомерСтроки
      |ИЗ
      |        РегистрРасчета.Начисления.БазаНачисления(&Измерения Основного,
      |                &ИзмерениятБазового, , Регистратор = &Регистратор И
      ВидРасчета = &ВидРасчета И
      |                Сотрудник В (&СписокСотрудников))
      |        КАК НачисленияБазаНачисления";

    Измер = Новый Массив(1);
    Измер[0] = "Сотрудник";

    Запрос.УстановитьПараметр("ИзмеренияОсновного", Измер);
    Запрос.УстановитьПараметр("ИзмеренияБазового", Измер);
    Запрос.УстановитьПараметр("Регистратор", Регистратор);
```

```
    Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);
    Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);

    ВыборкаРезультата = Запрос.Выполнить().Выбрать();

  КонецЕсли;

КонецПроцедуры
```

В параметрах виртуальной таблицы запроса мы кроме привычных для нас регистратора, вида расчета и списка сотрудников задаем еще измерения основного и базового регистров. В нашем случае это будет один и тот же регистр *Начисления*, а нужное нам измерение – *Сотрудник*.

В заключение осталось добавить во второе условие *Если…* обход набора записей регистра расчета и вычисление результата вторичных записей (листинг 18.7).

Листинг 18.7. Добавление обхода набора записей регистра и вычисления результата вторичных записей

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
СписокСотрудников) Экспорт

  Регистратор = НаборЗаписейРегистра.Отбор.Регистратор.Значение;

  // Рассчитать первичные записи
  Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад Тогда
```

```
…
  // Рассчитать вторичные записи
  ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
  Тогда

    ВыборкаРезультата = Запрос.Выполнить().Выбрать();

    Для Каждого ЗаписьРегистра Из НаборЗаписейРегистра Цикл
      СтруктураНомер = Новый Структура("НомерСтроки");
      СтруктураНомер.НомерСтроки = ЗаписьРегистра.НомерСтроки;
      ВыборкаРезультата.Сбросить();

      Если ВыборкаРезультата.НайтиСледующий(СтруктураНомер) Тогда
        ЗаписьРегистра.Результат = ВыборкаРезультата.База * (10 / 100);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "Выполнен расчет" + ЗаписьРегистра.Регистратор + " -
        " + ЗаписьРегистра.ВидРасчета + " - " + ЗаписьРегистра.Сотрудник;
        Сообщение.Сообщить();

      КонецЕсли;

    КонецЦикла;

  КонецЕсли;

КонецПроцедуры
```

Сумму начисленной премии мы рассчитываем как 10 % от рассчитанной оплаты по окладу.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим правильность работы процедуры расчета.

Отменим проведение документа *Начисления сотрудникам № 3* (*Все действия* > *Отмена проведения*) и перепроведем документы *Начисления сотрудникам № 1* и *№ 2*. Регистр расчета *Начисления* должен выглядеть следующим образом (рис. 18.14, 18.15).

Движения в регистре Начисления

Найти... Все действия

Пери...	Регистратор	Ном...	Вид расчета	Сторно	Сотрудник	Результат	График работы
01.07.2009...	Начисление ...	1	Оклад		Гусаков Николай Дмитриевич	7 000,00	График мастеров
01.07.2009...	Начисление ...	2	Оклад		Деловой Иван Сергеевич	8 000,00	График мастеров
01.07.2009...	Начисление ...	3	Оклад		Симонов Валерий Михайлович	3 000,00	График мастеров
01.07.2009...	Начисление ...	1	Премия		Гусаков Николай Дмитриевич	700,00	График мастеров
01.07.2009...	Начисление ...	2	Премия		Деловой Иван Сергеевич	800,00	График мастеров

Рис. 18.14. Записи регистра «Начисления»

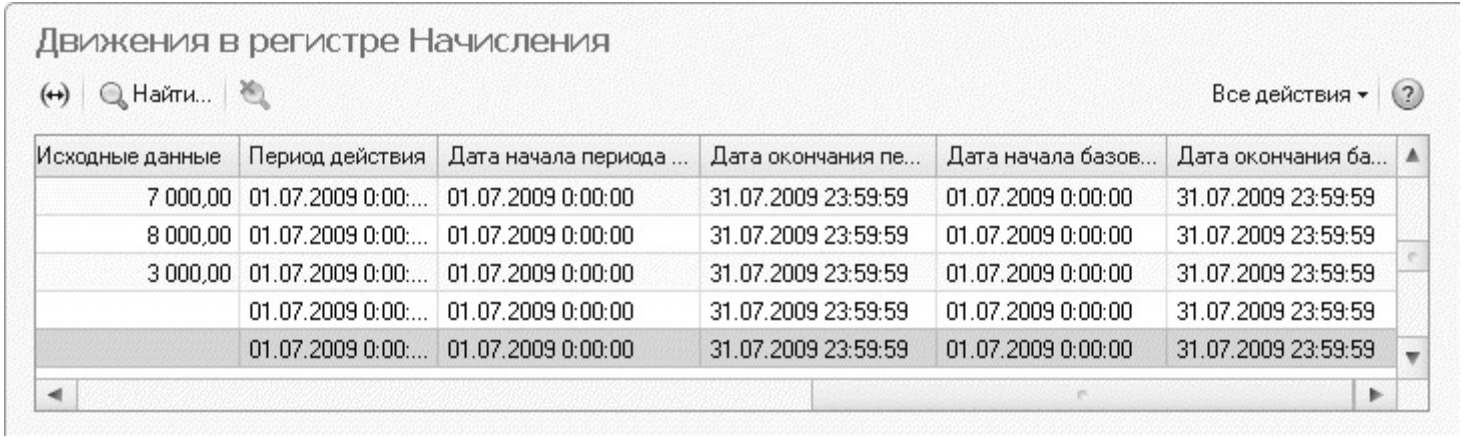


Рис. 18.15. Записи регистра «Начисления»

Мы видим, что всем сотрудникам произведены начисления по окладу (поле *Результат*) за полный месяц в соответствии с исходными данными (поле *Исходные данные*).

Сотрудникам Гусакову и Деловому начислена премия в размере 10 % от суммы начисления по окладу.

Проведем документ *Начисление сотрудникам № 3*, а затем *№ 1* и *№ 2*.

При этом отчет *Перерасчет* должен быть пуст.

Состояние регистра изменится следующим образом (рис. 18.16, 18.17).

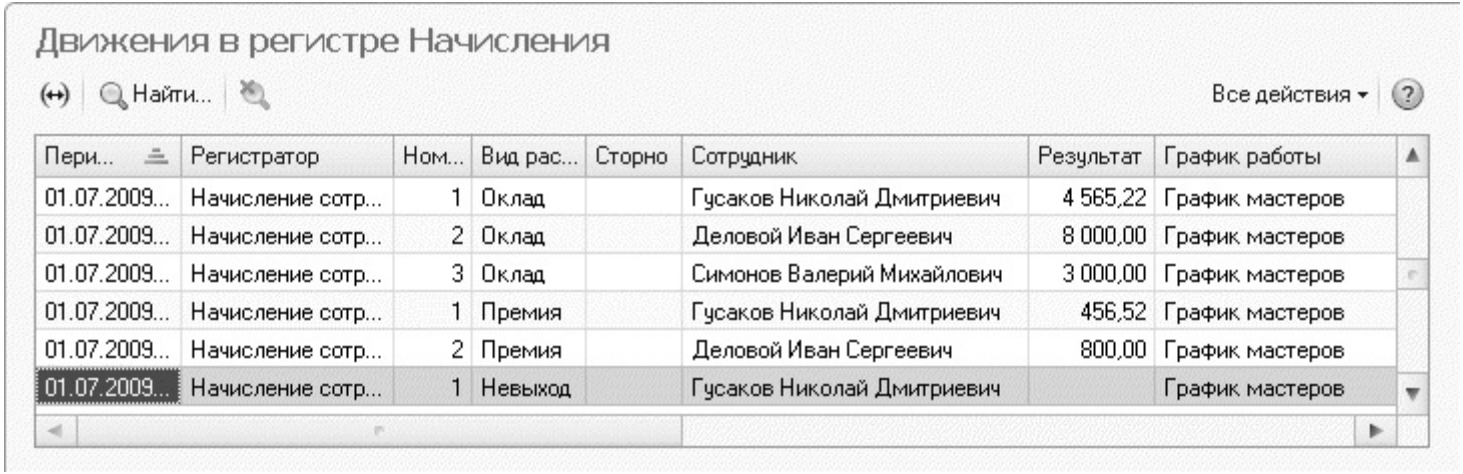


Рис. 18.16. Записи регистра «Начисления»

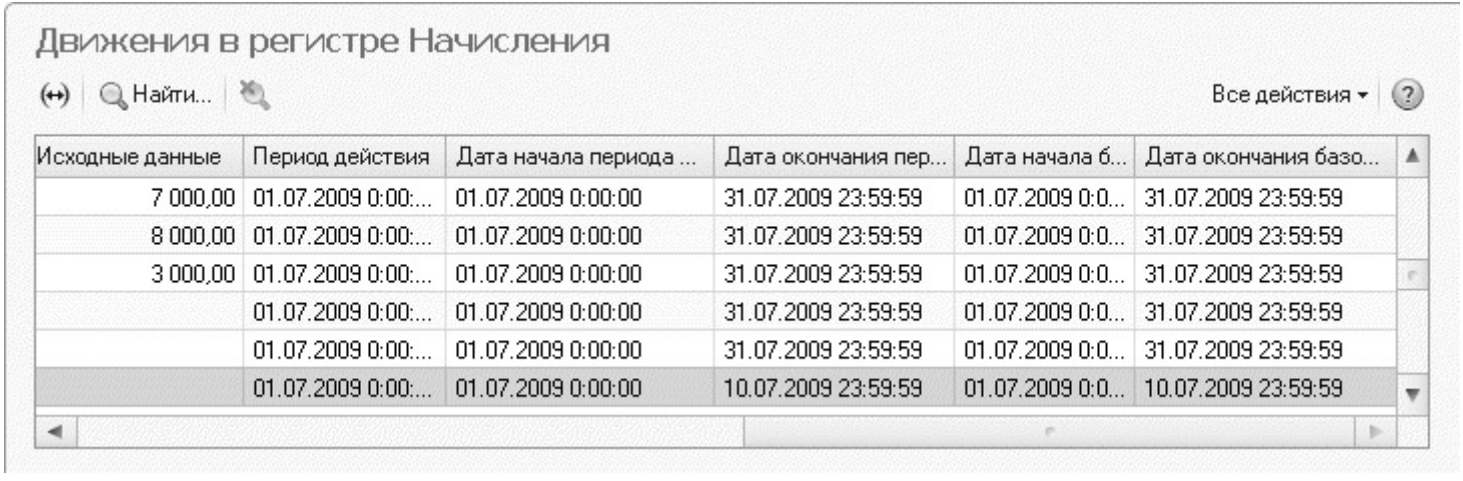
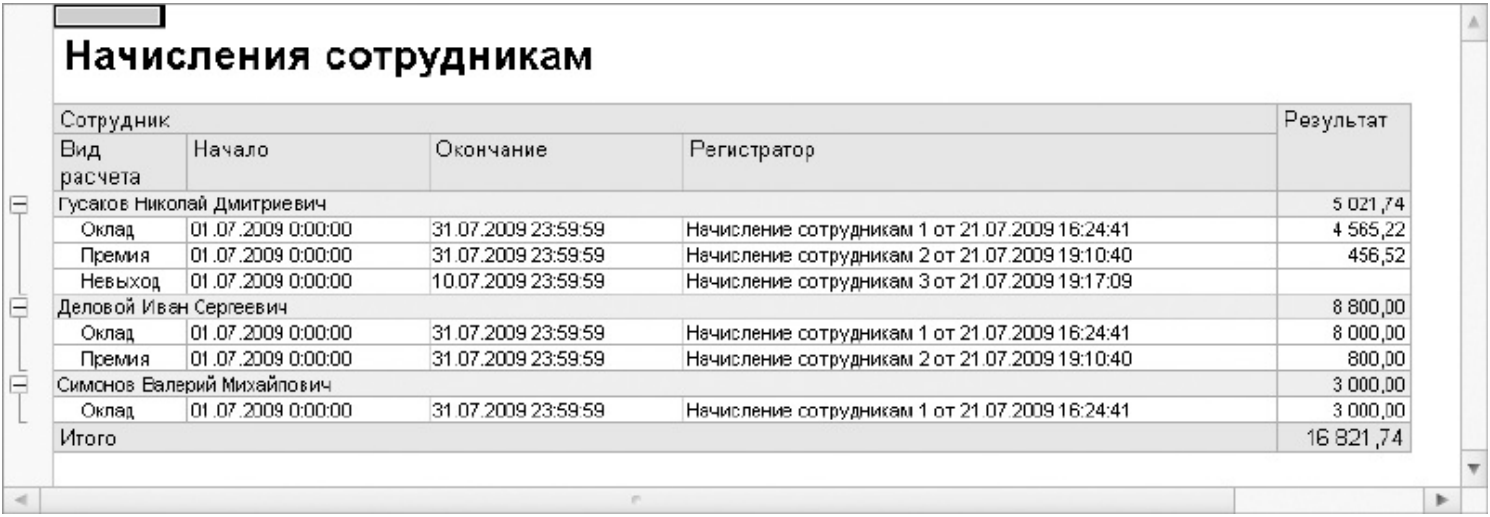


Рис. 18.17. Записи регистра «Начисления»

В результате невыхода Гусакова на работу сумма оплаты по окладу будет уменьшена и соответствующим образом уменьшится начисленная ему премия.

Отчет о начислениях сотрудникам

Теперь мы посмотрим, каким образом можно использовать данные, хранящиеся в регистре расчета, для получения в отчете итоговой информации о начислениях сотрудникам (рис. 18.18).



Начисления сотрудникам

Сотрудник / Вид расчета	Начало	Окончание	Регистратор	Результат
Гусаков Николай Дмитриевич				5 021,74
Оклад	01.07.2009 0:00:00	31.07.2009 23:59:59	Начисление сотрудникам 1 от 21.07.2009 16:24:41	4 565,22
Премия	01.07.2009 0:00:00	31.07.2009 23:59:59	Начисление сотрудникам 2 от 21.07.2009 19:10:40	456,52
Невыход	01.07.2009 0:00:00	10.07.2009 23:59:59	Начисление сотрудникам 3 от 21.07.2009 19:17:09	
Деловой Иван Сергеевич				8 800,00
Оклад	01.07.2009 0:00:00	31.07.2009 23:59:59	Начисление сотрудникам 1 от 21.07.2009 16:24:41	8 000,00
Премия	01.07.2009 0:00:00	31.07.2009 23:59:59	Начисление сотрудникам 2 от 21.07.2009 19:10:40	800,00
Симонов Валерий Михайлович				3 000,00
Оклад	01.07.2009 0:00:00	31.07.2009 23:59:59	Начисление сотрудникам 1 от 21.07.2009 16:24:41	3 000,00
Итого				16 821,74

Рис. 18.18. Результат отчета

В режиме «Конфигуратор»

Создадим в конфигураторе новый объект конфигурации *Отчет*. Назовем его *НачисленияСотрудникам*. Создадим основную схему компоновки данных отчета, добавим новый *Набор данных – запрос*, откроем конструктор запроса.

Запрос для набора данных

Выберем таблицу регистра расчета *Начисления* (рис. 18.19).

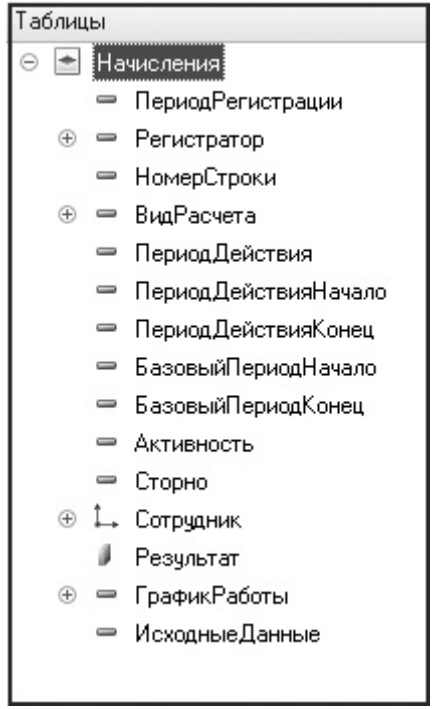


Рис. 18.19. Состав полей таблицы «Начисления»

Из нее выберем следующие поля:

- *Сотрудник*,
- *ВидРасчета*,
- *ПериодДействияНачало*,
- *ПериодДействияКонец*,
- *Регистратор,*
- *Результат* (рис. 18.20).

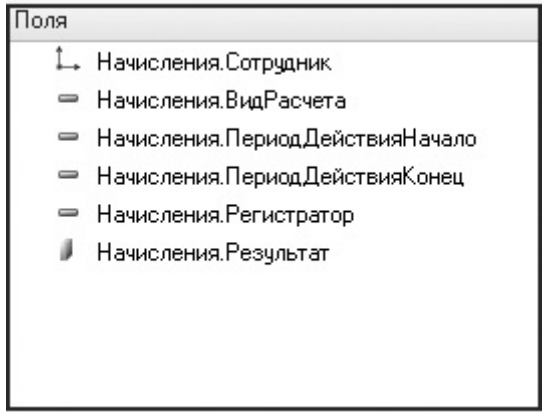


Рис. 18.20. Выбранные поля

На закладке *Объединения/Псевдонимы* определим следующие псевдонимы полей *ПериодДействияНачало* и *ПериодДействияКонец* (рис. 18.21).

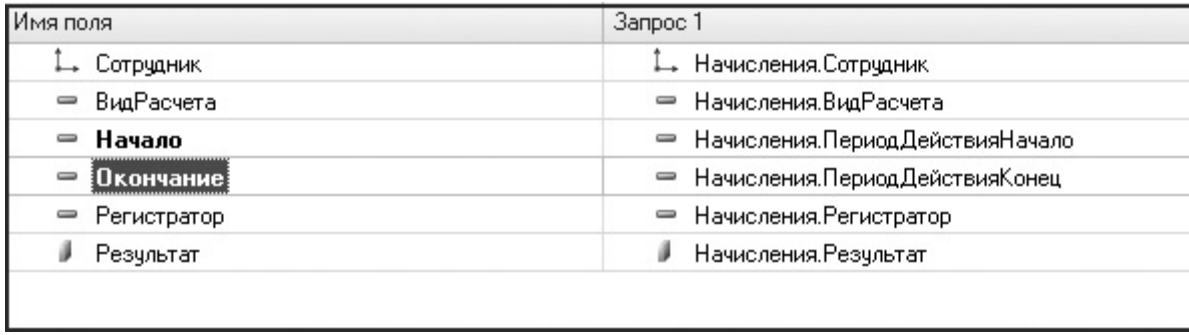


Рис. 18.21. Объединения/Псевдонимы

На этом создание запроса закончено, нажмем *OK*.

Ресурсы

Перейдем на закладку *Ресурсы* и укажем, что должна быть рассчитана сумма по полю *Результат*.

Настройки

После этого перейдем на закладку *Настройки* и создадим структуру отчета.

Добавим группировку по полю *Сотрудник* и в ней – подчиненную группировку детальных записей.

В качестве полей, выводимых в отчет, выберем поля *ВидРасчета*, *Начало*, *Окончание*, *Регистратор* и *Результат*.

В результате окно настроек отчета должно иметь вид (рис. 18.22).

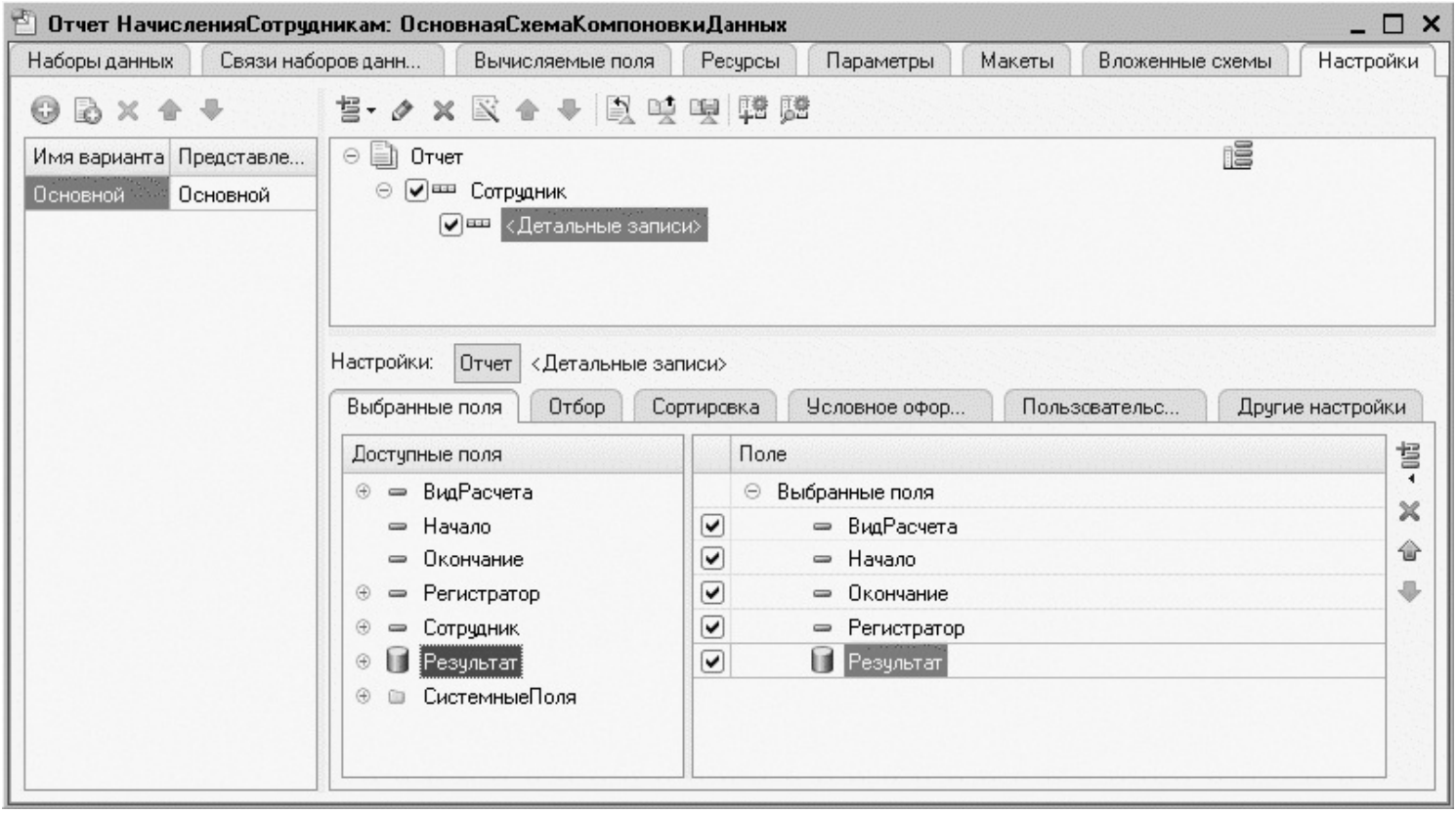


Рис. 18.22. Структура и выбранные поля отчета

На закладке *Сортировка* укажем, что сортировка должна выполняться по возрастанию значения поля *Сотрудник* и *Регистратор* (рис. 18.23).

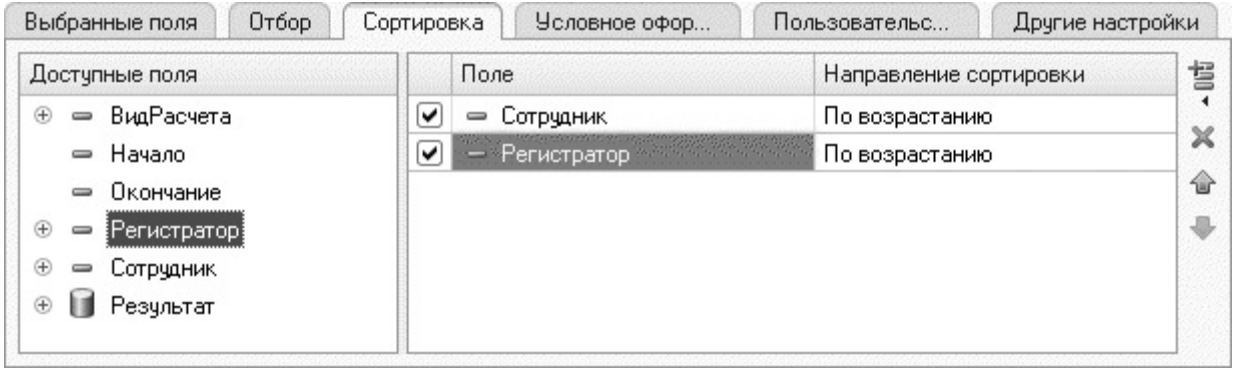


Рис. 18.23. Настройка сортировки отчета

И в заключение на закладке *Другие настройки* зададим заголовок отчета – *Начисления сотрудникам*.

На этом создание схемы компоновки данных закончено, закроем ее.

В окне редактирования объекта конфигурации Отчет *НачисленияСотрудникам* на закладке подсистемы укажем, что отчет будет вызываться из подсистем *РасчетЗарплаты* и *Бухгалтерия*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

В командной панели раздела *Расчет зарплаты* выполним команду *Начисления*

сотрудникам и сформируем отчет (рис. 18.24).

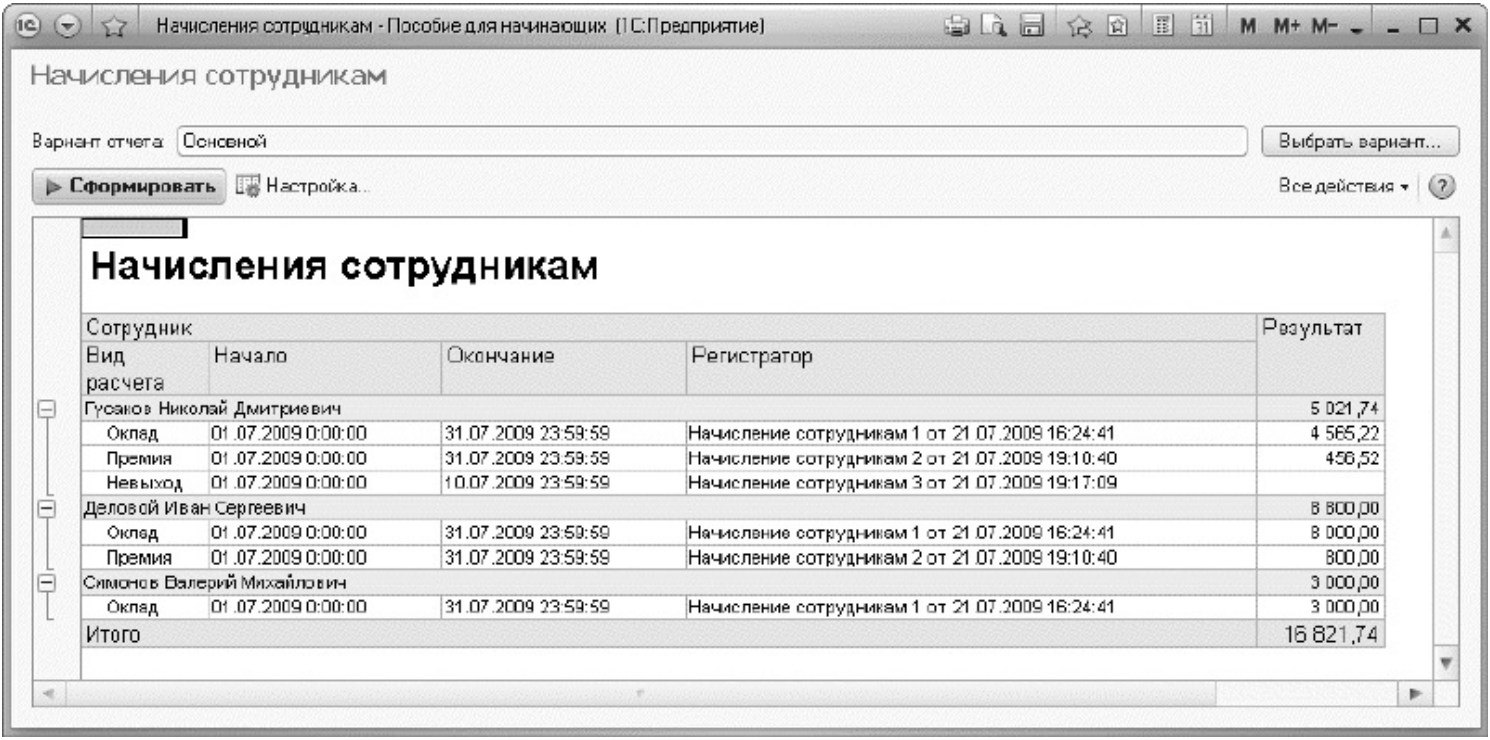


Рис. 18.24. Результат отчета

Перерасчет

Итак, в нашем алгоритме работы с данными расчета осталось одно узкое место – контроль актуальности данных, содержащихся в регистре расчета.

До сих пор мы с вами использовали служебный отчет *Перерасчет* для того, чтобы определить, являются ли данные в регистре расчета актуальными (если отчет пуст), или же они требуют перерасчета.

Теперь мы с вами создадим специальную процедуру, которая будет определять, требуется ли перерасчет данных регистра расчета, и, если такая необходимость есть, выполнять перерасчет.

Поскольку единственным способом получения итоговой информации о начислениях сотрудникам в нашей конфигурации является отчет *НачислениеСотрудникам*, для вызова этой процедуры мы создадим основную форму этого отчета и добавим в командную панель формы кнопку *Перерассчитать*, по которой будет выполняться перерасчет данных регистра.

В режиме «Конфигуратор»

Для этого в окне редактирования объекта конфигурации *Отчет НачислениеСотрудникам* перейдем на закладку *Формы*, нажмем кнопку открытия и создадим основную форму отчета.

Затем в правом верхнем окне редактора форм перейдем на закладку *Команды* и на закладке *Команды формы* создадим команду формы *Перерассчитать* (рис. 18.25).

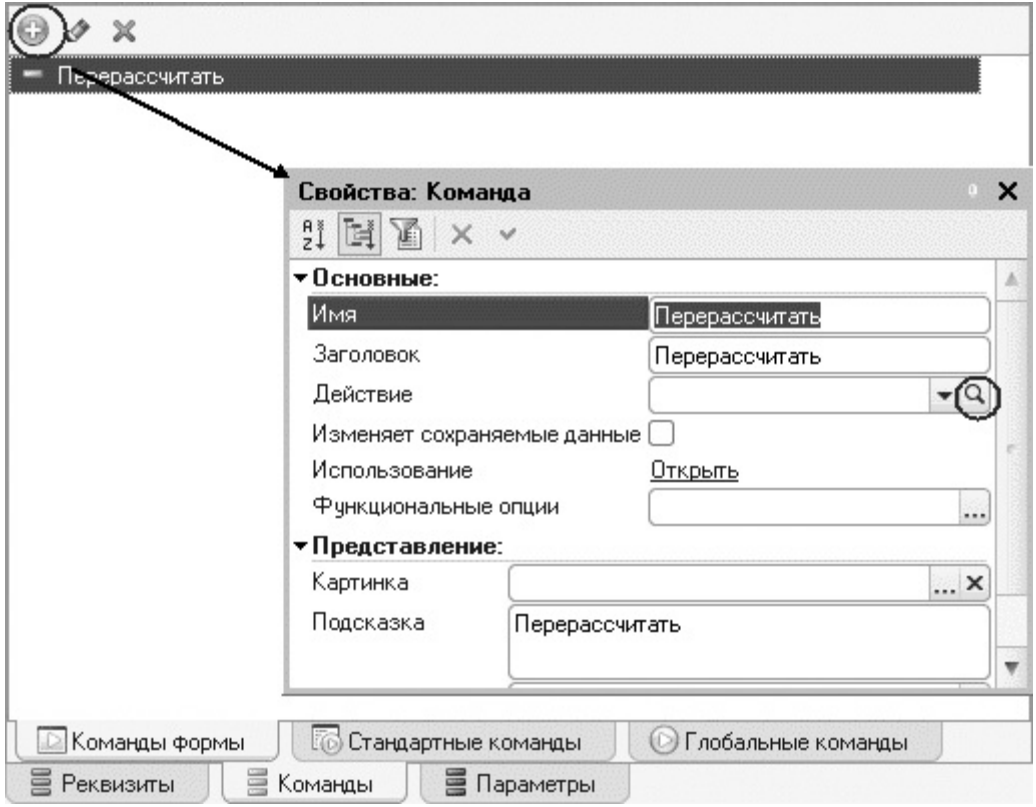


Рис. 18.25. Добавление команды формы

Теперь надо установить *Действие* для этой команды.

Для этого нажмем кнопку открытия в строке *Действие*.

В модуле формы будет создан шаблон процедуры *Перерассчитать()*, в которую мы поместим вызов процедуры *ПерерассчитатьНачисления()* из общего модуля *ПроведениеРасчетов* (листинг 18.8).

Листинг 18.8. Текст обработчика команды «Перерассчитать»

```
&НаКлиенте
Процедура Перерассчитать()

  ПроведениеРасчетов.ПерерассчитатьНачисления(ПредопределенноеЗначение("ПланВидов
  ПроведениеРасчетов.ПерерассчитатьНачисления(ПредопределенноеЗначение("ПланВидов

КонецПроцедуры
```

Саму процедуру перерасчета поместим в общем модуле *ПроведениеРасчетов* (листинг 18.9).

Листинг 18.9. Процедура перерасчета начислений

```
Процедура ПерерассчитатьНачисления(ТребуемыйВидРасчета) Экспорт

  // Здесь следует выбрать из набора записей перерасчета записи в следующей
  последовательности:
  // записи документа1 для сотрудников из списка,
  // записи документа2 для сотрудников из списка и т. д.
  Запрос = Новый Запрос(
    "ВЫБРАТЬ
```

```
    |        НачисленияПерерасчет.ОбъектПерерасчета,
    |        НачисленияПерерасчет.Сотрудник
    |ИЗ
    |        РегистрРасчета.Начисления.Перерасчет КАК НачисленияПерерасчет
    |ГДЕ
    |        НачисленияПерерасчет.ВидРасчета = &ТребуемыйВидРасчета
    |ИТОГИ ПО
    |        НачисленияПерерасчет.ОбъектПерерасчета");

Запрос.УстановитьПараметр("ТребуемыйВидРасчета", ТребуемыйВидРасчета);
СписокСотрудников = Новый СписокЗначений;

// Перебрать группировку по регистратору.
ВыборкаПоРегистратору =
Запрос.Выполнить().Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
Пока ВыборкаПоРегистратору.Следующий() Цикл
  Регистратор = ВыборкаПоРегистратору.ОбъектПерерасчета;

  // Перебрать группировку по сотрудникам для выбранного регистратора
  // и создать список сотрудников.
  ВыборкаПоСотрудникам = ВыборкаПоРегистратору.Выбрать();
  СписокСотрудников.Очистить();

  Пока ВыборкаПоСотрудникам.Следующий() Цикл
    СписокСотрудников.Добавить(ВыборкаПоСотрудникам.Сотрудник);

  КонецЦикла;

  // Получить набор записей регистра расчета для выбранного регистратора.
  НаборЗаписей = РегистрыРасчета.Начисления.СоздатьНаборЗаписей();
  НаборЗаписей.Отбор.Регистратор.Значение = Регистратор;
```

```
    НаборЗаписей.Прочитать();

    РассчитатьНачисления(НаборЗаписей, ТребуемыйВидРасчета, СписокСотрудников);
    НаборЗаписей.Записать( , Истина);

    // Очистить перерассчитанные записи в перерасчете.
    НаборЗаписейПерерасчета =
    РегистрыРасчета.Начисления.Перерасчеты.Перерасчет.СоздатьНаборЗаписей();
    НаборЗаписейПерерасчета.Отбор.ОбъектПерерасчета.Значение = Регистратор;
    НаборЗаписейПерерасчета.Записать();

  КонецЦикла;

КонецПроцедуры
```

В самом начале процедуры мы запросом выбираем данные о записях перерасчетов, содержащие переданный вид расчета и сгруппированные по объекту перерасчета.

Далее при обходе результата запроса мы формируем для каждого объекта перерасчета список сотрудников, читаем соответствующие записи регистра расчета и вызываем процедуру *РассчитатьНачисления*, которая использовалась нами при расчете записей документа *НачислениеСотрудникам*.

После того как расчет записей выполнен, мы записываем набор записей без

формирования записей перерасчета и очищаем записи перерасчета по тому объекту перерасчета, который только что обработали.

Вернемся в форму отчета *НачислениеСотрудникам*.

Итак, мы указали для команды *Перерассчитать* действие, то есть процедуру для ее выполнения. Но чтобы можно было воспользоваться этой командой, нужно создать в форме кнопку и связать ее с этой командой (в строке *Команда*).

Проще всего это сделать перетаскиванием команды из окна *Команды формы* в окно элементов формы.

Перетащим мышью команду *Перерассчитать* в группу элементов формы *ОсновнаяКомандная Панель*.

При этом в форме появится кнопка *Перерассчитать*, а связь кнопки с командой будет установлена автоматически (рис. 18.26).

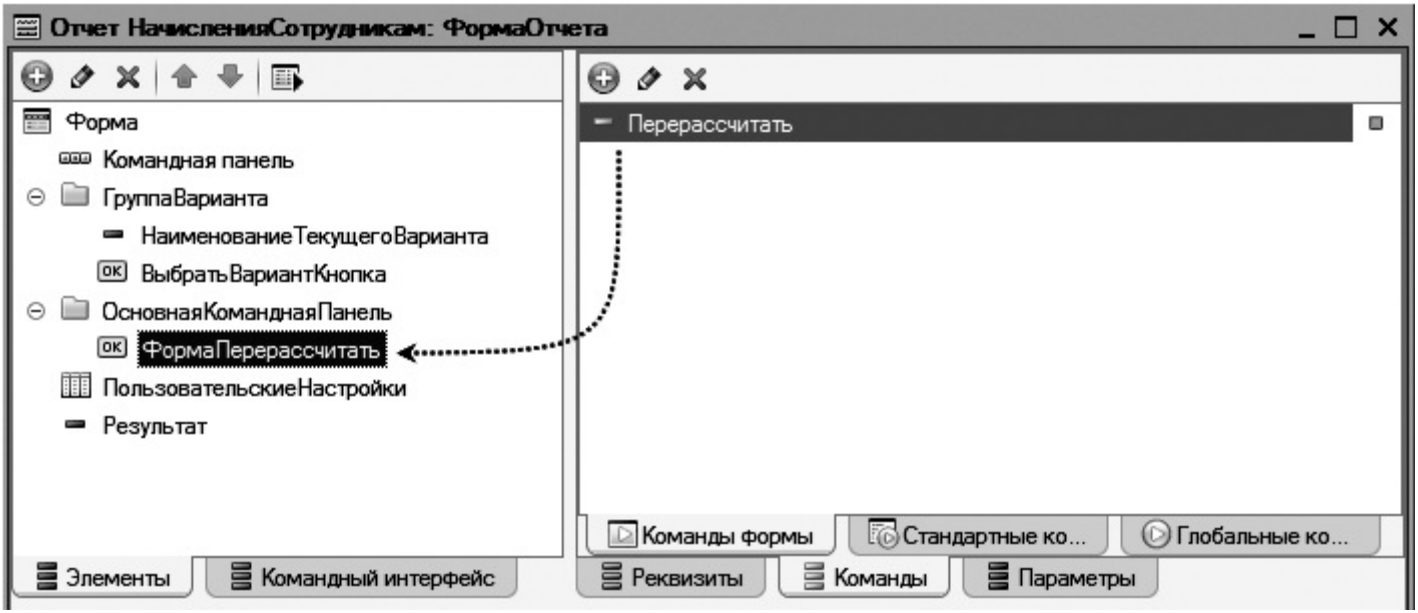


Рис. 18.26. Добавление кнопки в форму

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» и проверим, как выполняется перерасчет записей регистра расчета.

Отменим проведение всех документов *Начисления сотрудникам* и проведем документ *Начисления сотрудникам № 1* и затем *№ 2*.

Сформируем отчет *Начисления сотрудникам* (рис. 18.27).

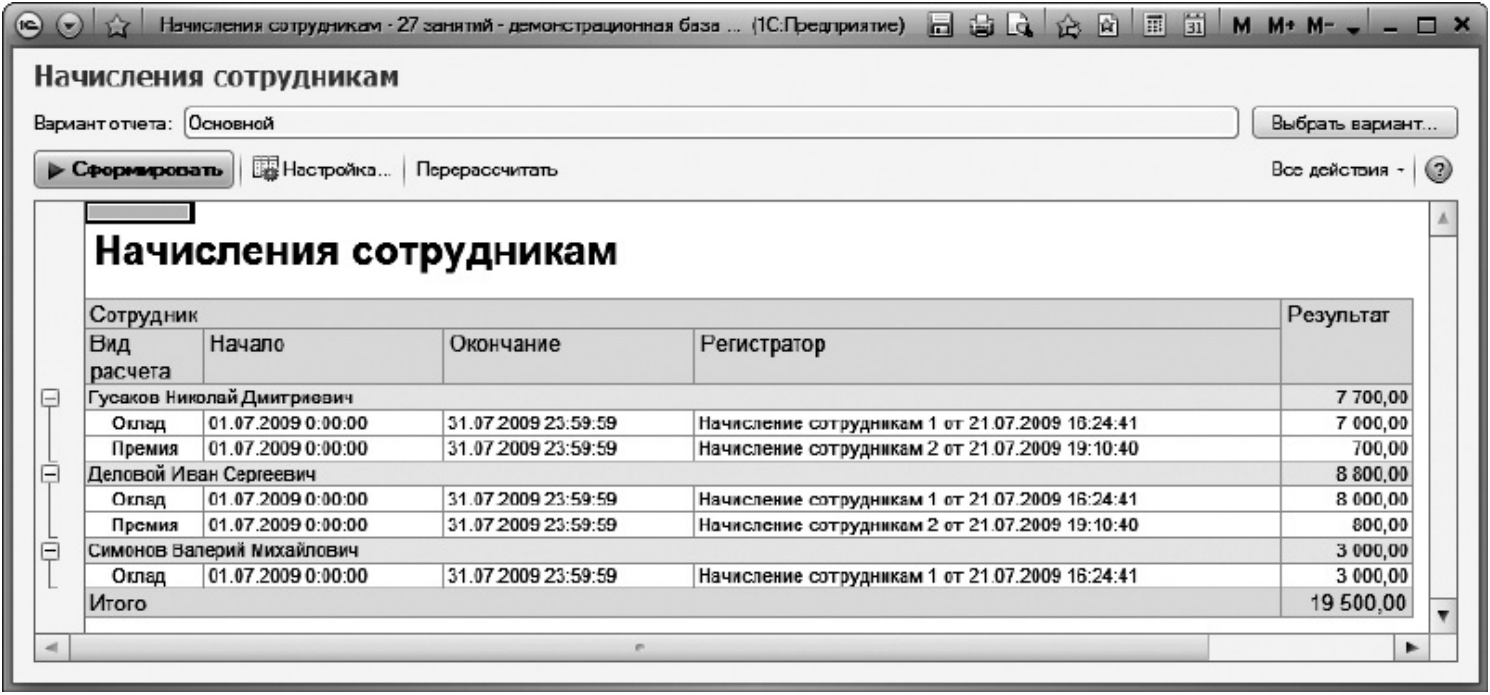


Рис. 18.27. Отчет «Начисления сотрудникам»

Теперь откроем документ *Начисления сотрудникам № 1*, изменим оклад Гусакова на 10 000 и проведем документ.

В отчете *Начисления сотрудникам* нажмем кнопку *Перерассчитать*.

Будет выполнен перерасчет начисления премии Гусакову и Деловому (рис. 18.28).

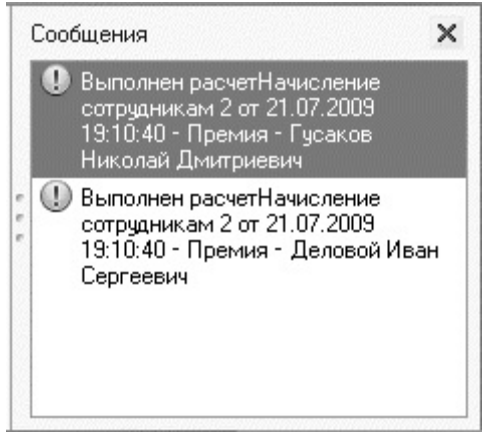


Рис. 18.28. Окно служебных сообщений

Чтобы увидеть в отчете актуальные данные, нажмем кнопку *Сформировать*.

Результат работы отчета будет содержать новые значения премии Гусакова (рис. 18.29).

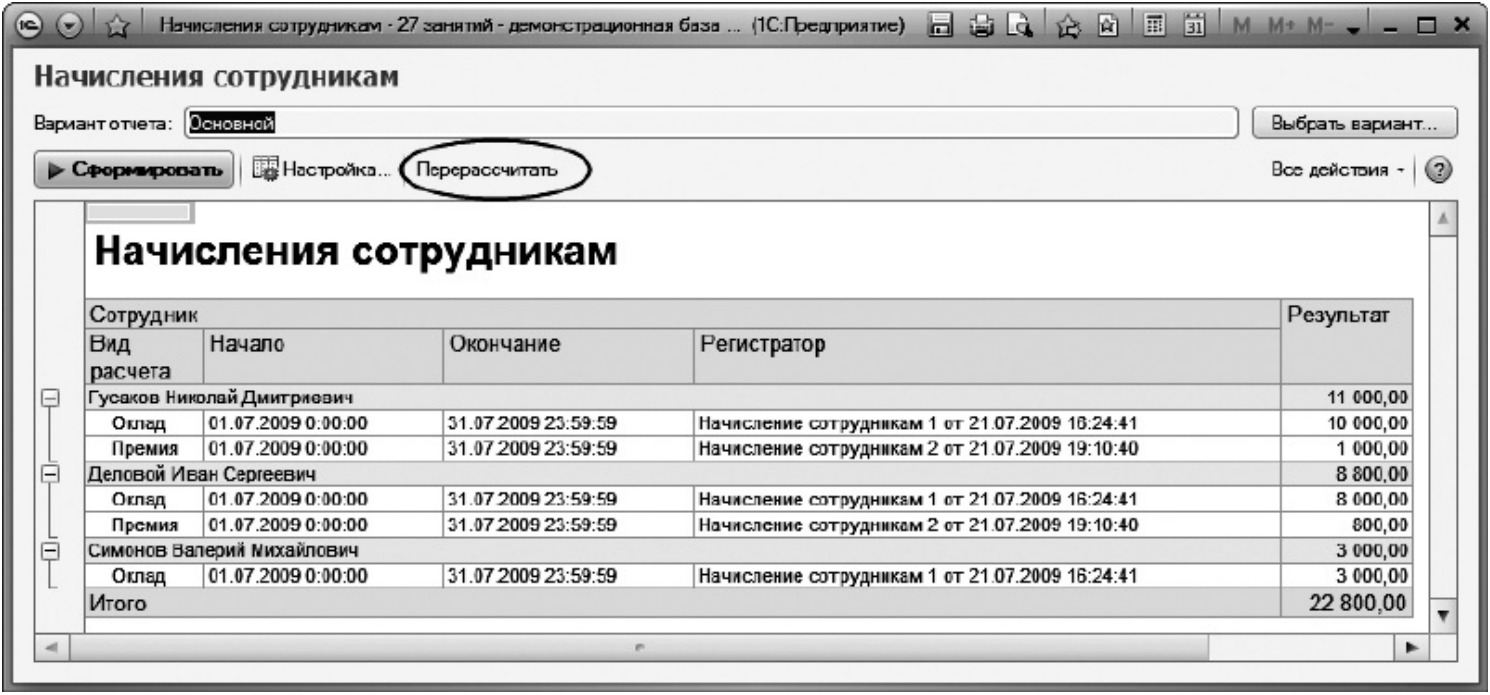


Рис. 18.29. Отчет «Начисления сотрудникам»

И, наконец, проведем документ *Начисления сотрудникам № 3* и нажмем *Перерассчитать* в отчете *Начисления сотрудникам*.

Снова будет произведен перерасчет оклада и премии Гусакова (рис. 18.30).

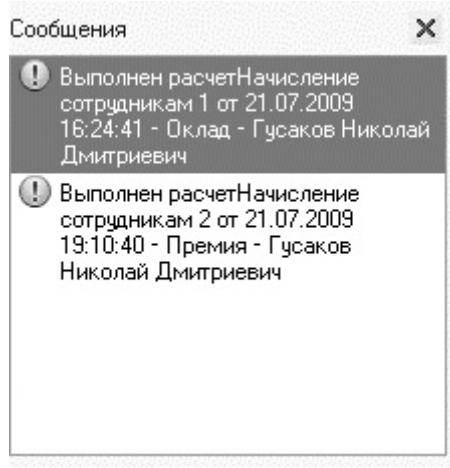


Рис. 18.30. Окно служебных сообщений

Нажмем кнопку *Перерассчитать*, а затем *Сформировать*. Данные отчета будут содержать актуальные значения начисления оклада и премии (рис. 18.31).

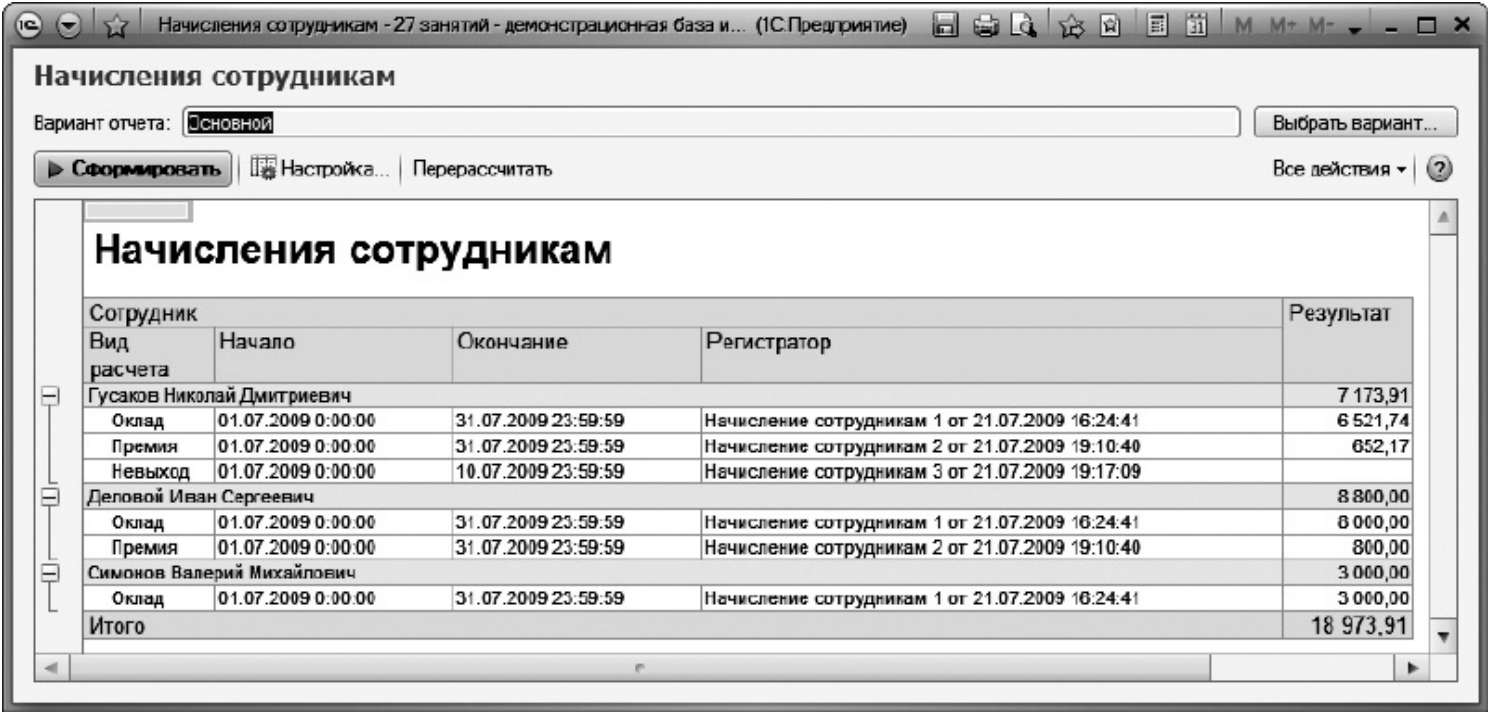


Рис. 18.31. Отчет «Начисления сотрудникам»

Диаграмма Ганта

В конце этой главы мы совместим приятное с полезным и создадим с вами отчет, который в графическом виде будет показывать нам фактический период действия записей расчета.

Помимо наглядной демонстрации работы механизма вытеснения записей по периоду действия этот отчет позволит нам познакомиться с элементом формы, позволяющим создавать *диаграммы Ганта*.

Диаграмма Ганта представляет собой диаграмму интервалов на шкале времени (рис. 18.32) и отражает использование объектами (точками) ресурсов (серий).

Чтобы проще было представить себе составные части диаграммы Ганта, изучим диаграмму, которая должна получиться в результате работы создаваемого нами отчета.

Как мы уже говорили, эта диаграмма будет отображать для каждого сотрудника фактический период действия записи по каждому из видов расчета, имеющих место для этого сотрудника.

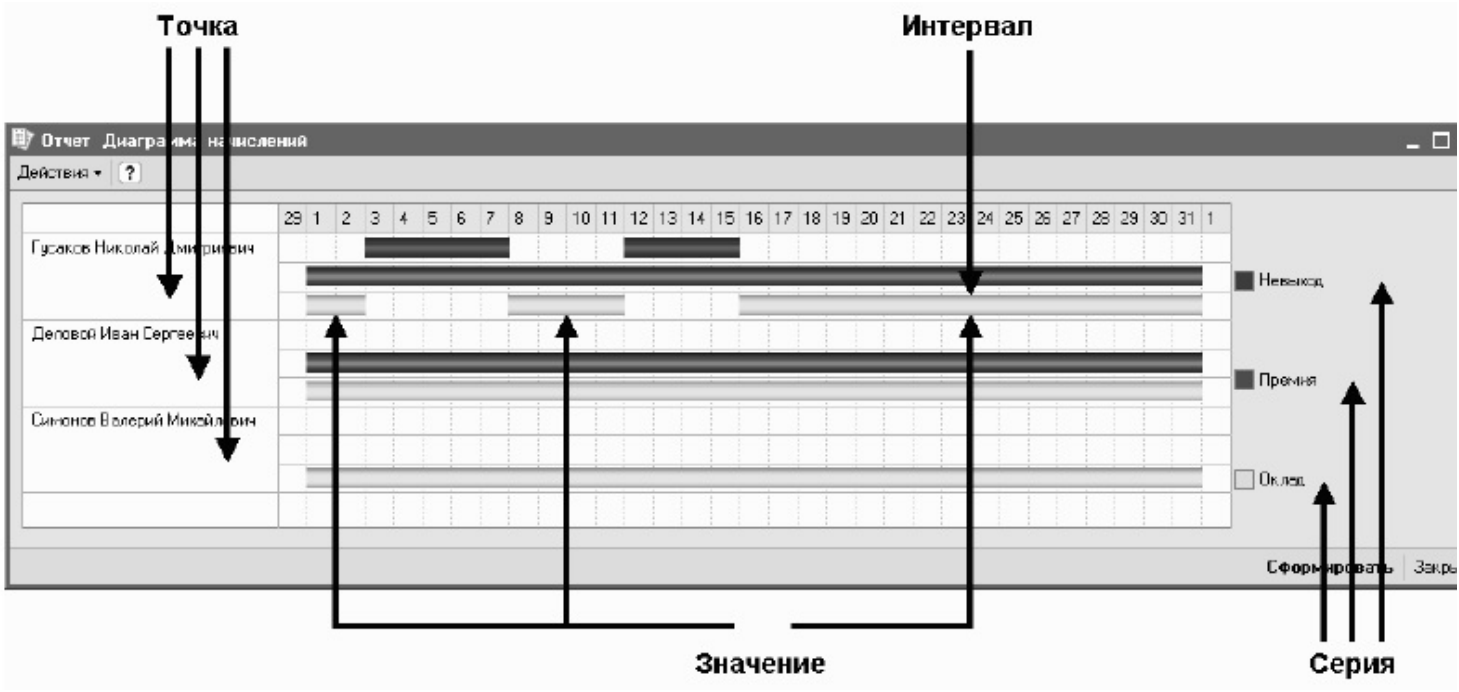


Рис. 18.32. Пример диаграммы Ганта

Итак, диаграмма Ганта представляет собой совокупность точек, серий и значений для каждой пары точка-серия.

В нашем случае точками диаграммы являются сотрудники, а сериями – виды расчетов. Таким образом, для каждого сотрудника существует некоторое значение диаграммы по каждой из серий, то есть по каждому из видов расчета.

Значение диаграммы Ганта представляет собой специальный объект, который автоматически формируется системой на основании того, какие точки и какие серии определены для данной диаграммы.

Этот объект является совокупностью (коллекцией) интервалов, то есть может содержать не один, а несколько интервалов, которые соответствуют паре серия-точка (создаваемый по умолчанию объект *ЗначениеДиаграммыГанта* не содержит ни одного интервала). Разработчик может получить значение диаграммы, указав интересующую его точку и серию, и затем добавить в коллекцию необходимое количество интервалов.

Все интервалы всех значений диаграммы располагаются с привязкой к единой оси времени, что дает возможность видеть их взаимное расположение.

Теперь коротко объясним последовательность наших дальнейших действий.

В качестве исходных данных для построения такой диаграммы мы возьмем данные регистра расчета *Начисления*. Каждая запись этого регистра уже содержит все необходимое для построения диаграммы: сотрудника, вид расчета, начало и конец интервала.

Нам останется только средствами встроенного языка разместить все это в диаграмме.

Итак, приступим.

В режиме «Конфигуратор»

Создадим новый объект конфигурации *Отчет* и назовем его *ДиаграммаНачислений*.

Для этого отчета мы не будем создавать схему компоновки данных, а создадим основную форму отчета и обеспечим формирование и настройку диаграммы Ганта с помощью кода на встроенном языке. В окне редактирования объекта конфигурации *Отчет ДиаграммаНачислений* перейдем на закладку *Формы*, нажмем кнопку открытия  и создадим основную форму отчета.

В правом верхнем окне редактора форм на закладке *Реквизиты* находятся реквизиты формы. Мы видим здесь основной реквизит формы *Отчет*, который был создан автоматически при создании формы.

Нажмем кнопку *Добавить* и добавим новый реквизит формы. Назовем его *ДиаграммаГанта* и выберем его тип *ДиаграммаГанта* (рис. 18.33).

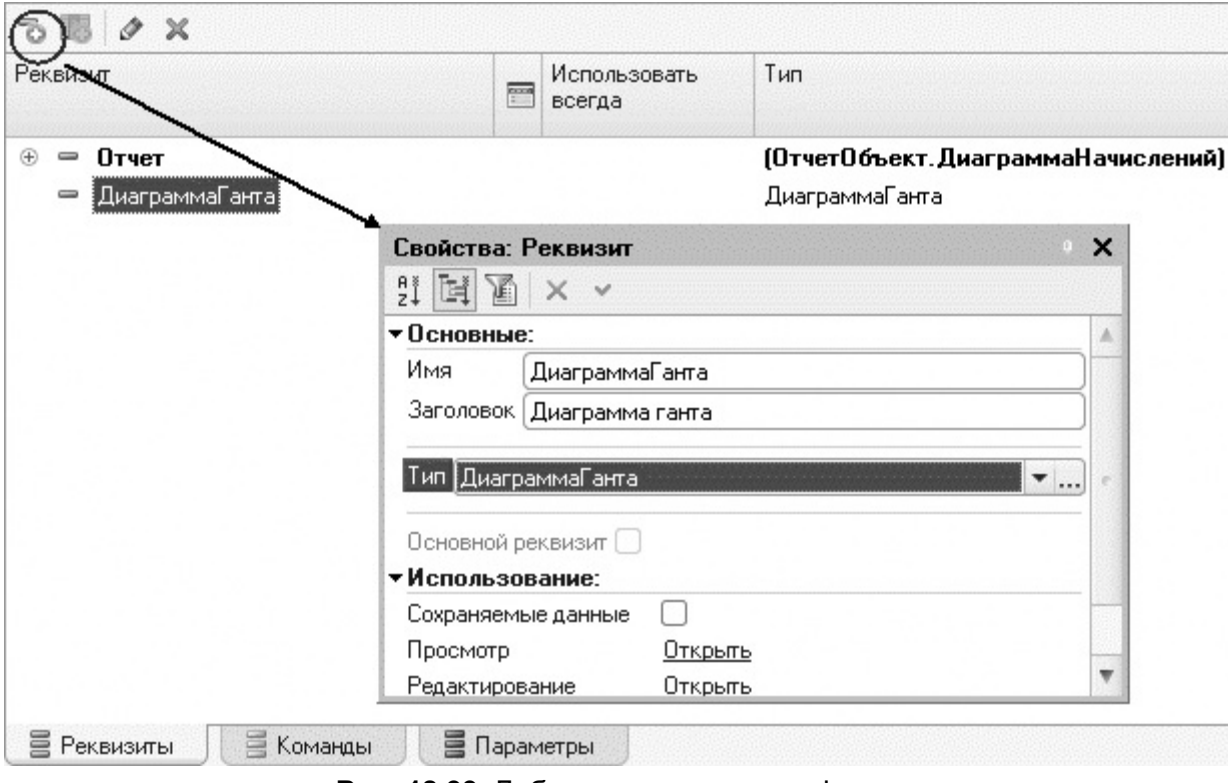


Рис. 18.33. Добавление реквизита формы

Теперь перетащим новый реквизит в окно элементов формы, которое пока пусто.

В окне элементов формы будет создано новое поле для отображения диаграммы Ганта, а нижнем окне просмотра формы мы сразу увидим поле

диаграммы (рис. 18.34).

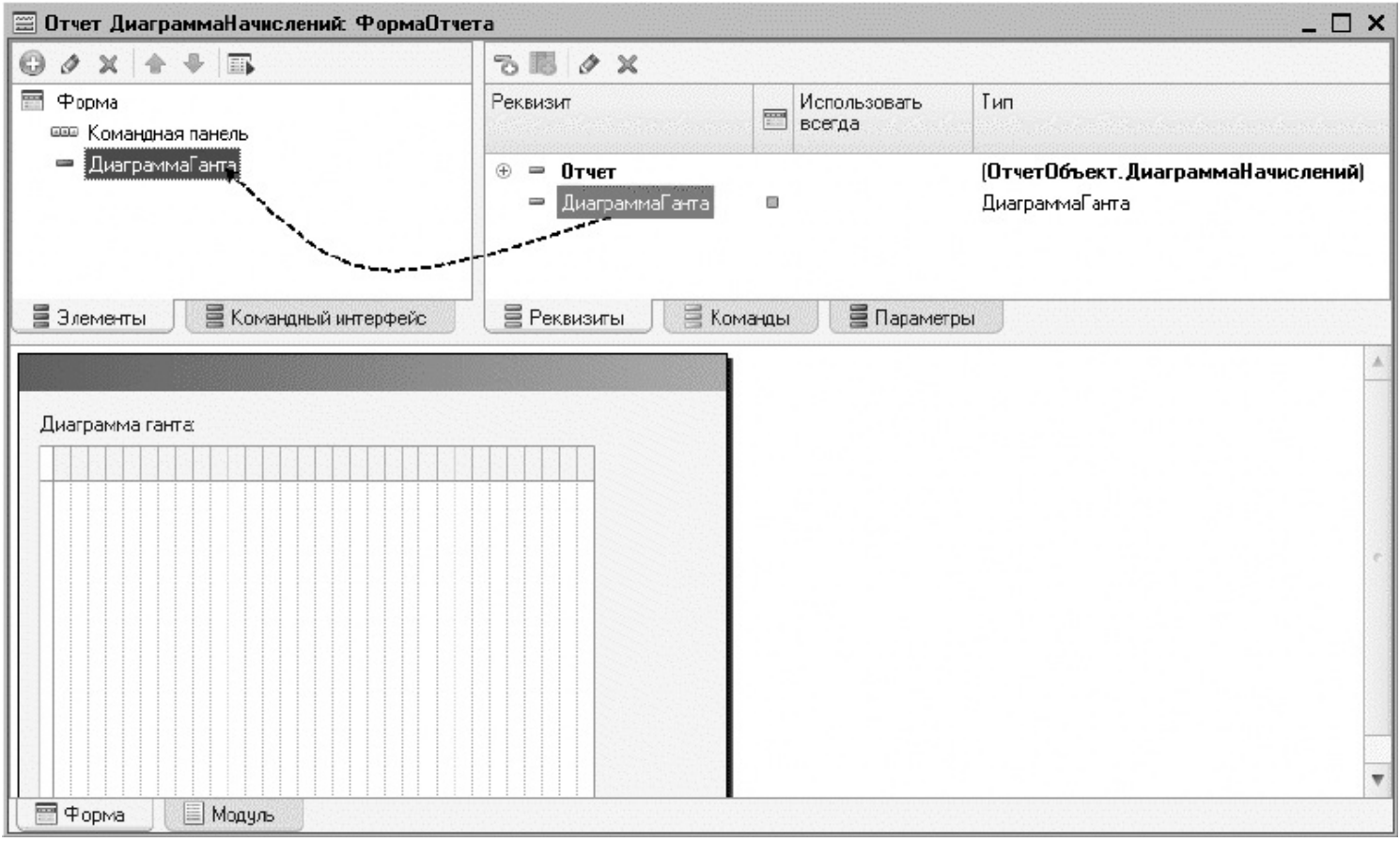


Рис. 18.34. Добавление диаграммы Ганта в форму

На закладке *Команды* создадим команду формы *Сформировать* (рис. 18.35).

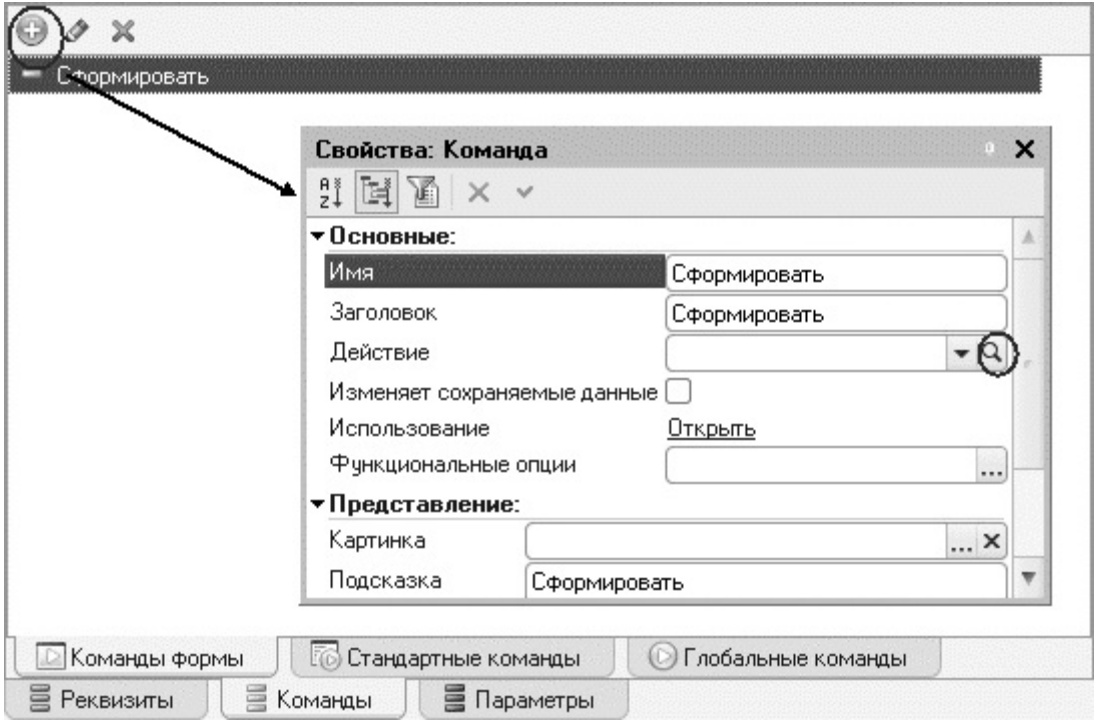


Рис. 18.35. Добавление команды формы

Теперь нужно установить *Действие* для этой команды.

Для этого нажмем кнопку открытия в строке *Действие*.

В модуле формы будет создан шаблон процедуры *Сформировать()*, в которую мы поместим вызов процедуры *СформироватьНаСервере()* и в

качестве параметра передадим в нее ссылку на реквизит формы *ДиаграммаГанта* (листинг 18.10).

Листинг 18.10. Текст обработчика команды «Сформировать»

```
&НаКлиенте
Процедура Сформировать()

  СформироватьНаСервере(ДиаграммаГанта);

КонецПроцедуры
```

Процедуру *СформироватьНаСервере()* мы поместим также в модуле формы и предварим ее директивой исполнения *&НаСервереБезКонтекста*. В нее мы вставим заготовку запроса (листинг 18.11).

Листинг 18.11. Процедура «СформироватьНаСервере()»

```
&НаСервереБезКонтекста
Процедура СформироватьНаСервере(Диаграмма)

  Запрос = Новый Запрос;
  Запрос.Текст = ;

КонецПроцедуры
```

Установим курсор перед точкой с запятой, вызовем контекстное меню, откроем конструктор запроса и создадим новый запрос.

Выберем виртуальную таблицу регистра расчета *Начисления.ФактическийПериодДействия*.

Из этой таблицы выберем следующие поля (рис. 18.36):

- *Сотрудник,*
- *ВидРасчета,*
- *ПериодДействияНачало,*
- *ПериодДействияКонец,*
- *Результат,*
- *Регистратор,*
- *Регистратор.Представление.*

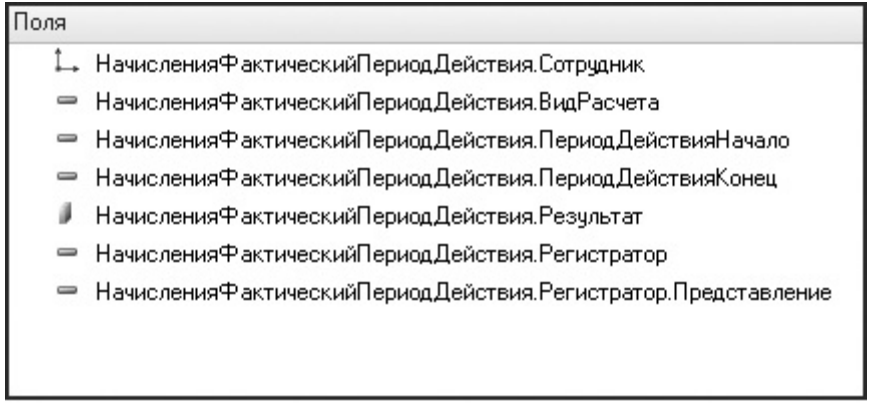


Рис. 18.36. Выбранные поля

Все, запрос готов.

Теперь нажмем *OK* и после текста запроса добавим в процедуру следующий текст (листинг 18.12).

Листинг 18.12. Процедура «СформироватьНаСервере()»

```
&НаСервереБезКонтекста
Процедура СформироватьНаСервере(Диаграмма)

  Запрос = Новый Запрос;
  Запрос.Текст =
  "ВЫБРАТЬ
  |        НачисленияФактическийПериодДействия.Сотрудник,
  |        НачисленияФактическийПериодДействия.ВидРасчета,
```

```
|        НачисленияФактическийПериодДействия.ПериодДействияНачало,
|        НачисленияФактическийПериодДействия.ПериодДействияКонец,
|        НачисленияФактическийПериодДействия.Результат,
|        НачисленияФактическийПериодДействия.Регистратор,
|        НачисленияФактическийПериодДействия.Регистратор.Представление
|ИЗ
|        РегистрРасчета.Начисления.ФактическийПериодДействия КАК
НачисленияФактическийПериодДействия";

ВыборкаРезультата = Запрос.Выполнить().Выбрать();

// Запретить обновление диаграммы.
Диаграмма.Обновление = Ложь;

Диаграмма.Очистить();
Диаграмма.ОтображатьЗаголовок = Ложь;

// Заполнить диаграмму.
Пока ВыборкаРезультата.Следующий() Цикл

  // Получить серию, точку и значение для них.
  ТекущаяСерия = Диаграмма.УстановитьСерию(ВыборкаРезультата.ВидРасчета);
  ТекущаяТочка = Диаграмма.УстановитьТочку(ВыборкаРезультата.Сотрудник);
  ТекущееЗначение = Диаграмма.ПолучитьЗначение(ТекущаяТочка, ТекущаяСерия);

  // Создать нужные нам интервалы в значении.
  ТекущийИнтервал = ТекущееЗначение.Добавить();
  ТекущийИнтервал.Начало = ВыборкаРезультата.ПериодДействияНачало;
  ТекущийИнтервал.Конец = ВыборкаРезультата.ПериодДействияКонец;
  ТекущийИнтервал.Текст = ВыборкаРезультата.РегистраторПредставление;
  ТекущийИнтервал.Расшифровка = ВыборкаРезультата.Регистратор;
```

```
  КонецЦикла;

  // Раскрасить серии своими цветами.
  Для Каждого Серия из Диаграмма.Серии Цикл
    Если Серия.Значение = ПланыВидовРасчета.ОсновныеНачисления.Оклад Тогда
      Серия.Цвет = WEBЦвета.Желтый;

    ИначеЕсли Серия.Значение = ПланыВидовРасчета.ОсновныеНачисления.Премия
    Тогда
      Серия.Цвет = WEBЦвета.Зеленый;

    ИначеЕсли Серия.Значение = ПланыВидовРасчета.ОсновныеНачисления.Невыход
    Тогда
      Серия.Цвет = WEBЦвета.Красный;

    КонецЕсли;

  КонецЦикла;

  // Разрешить обновление диаграммы.
  Диаграмма.Обновление = Истина;

КонецПроцедуры
```

Сначала мы запрещаем обновление диаграммы на то время, пока мы будем заполнять ее данными. Это нужно для того, чтобы в процессе заполнения не выполнялись пересчеты при каждом изменении данных диаграммы. После

окончания заполнения диаграммы мы разрешим обновление, и все пересчеты будут выполнены один раз.

Затем в цикле по выборке запроса мы заполняем диаграмму.

Сначала, используя методы *УстановитьСерию()* и *Установить Точку()*, мы получаем либо существующие, либо новые точку и серию. Точки и серии однозначно идентифицируются своими значениями, в качестве которых мы используем сотрудника и вид расчета из результата запроса.

После того как точка и серия получены, с помощью метода *ПолучитьЗначение()* мы получаем соответствующее им значение диаграммы.

Затем мы добавляем в значение диаграммы новый интервал, задаем его начало и конец, задаем текст интервала, который будет показываться во всплывающей подсказке, и задаем расшифровку интервала, которая будет выполняться при двойном щелчке мышью на этом интервале.

После того как все значения диаграммы сформированы, мы раскрашиваем серии своими цветами. Серии диаграммы представляют собой коллекцию значений, которую мы перебираем при помощи конструкции *Для Каждого … Цикл*.

Теперь вернемся в форму и добавим в нее кнопку для выполнения команды *Сформировать*.

Для этого перетащим мышью команду *Сформировать* из окна *Команды формы* в окно элементов формы, наведя мышь на строку *Форма* (рис. 18.37).

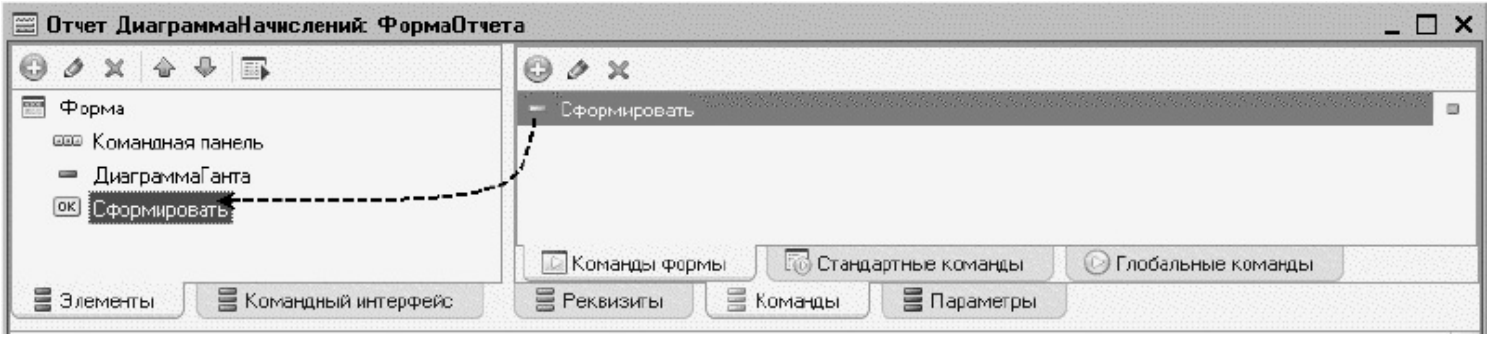


Рис. 18.37. Добавление кнопки в форму

В заключение в окне редактирования объекта конфигурации Отчет *ДиаграммаНачислений* на закладке подсистемы укажем, что отчет будет вызываться из подсистемы *РасчетЗарплаты*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и посмотрим на результат работы отчета (рис. 18.38).

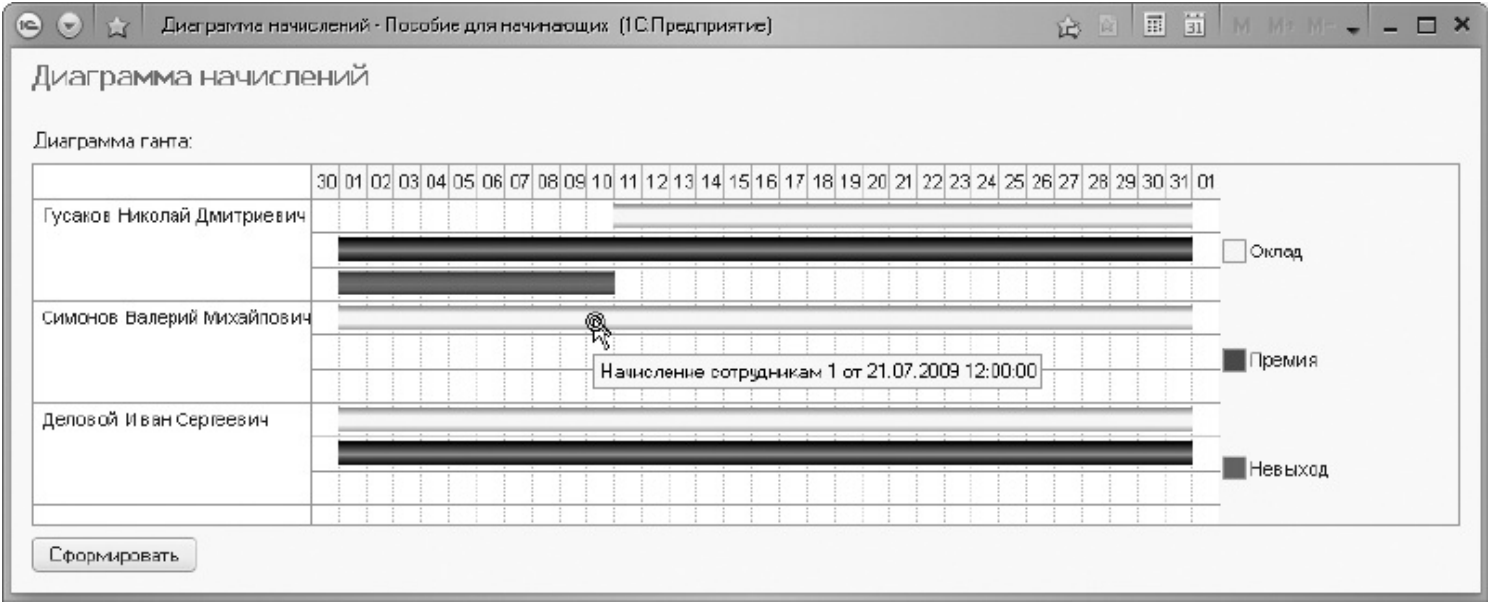


Рис. 18.38. Отчет «Диаграмма начислений»

А теперь посмотрим, как выглядит механизм вытеснения по периоду действия *в действии*.

Откроем документ *Начисления сотрудникам № 3* и вместо одного прогула с 1 по 10 число зададим Гусакову два прогула: с 3 по 7 число и с 12 по 15 число.

Проведем документ и снова нажмем *Сформировать* в нашем отчете (рис. 18.39).

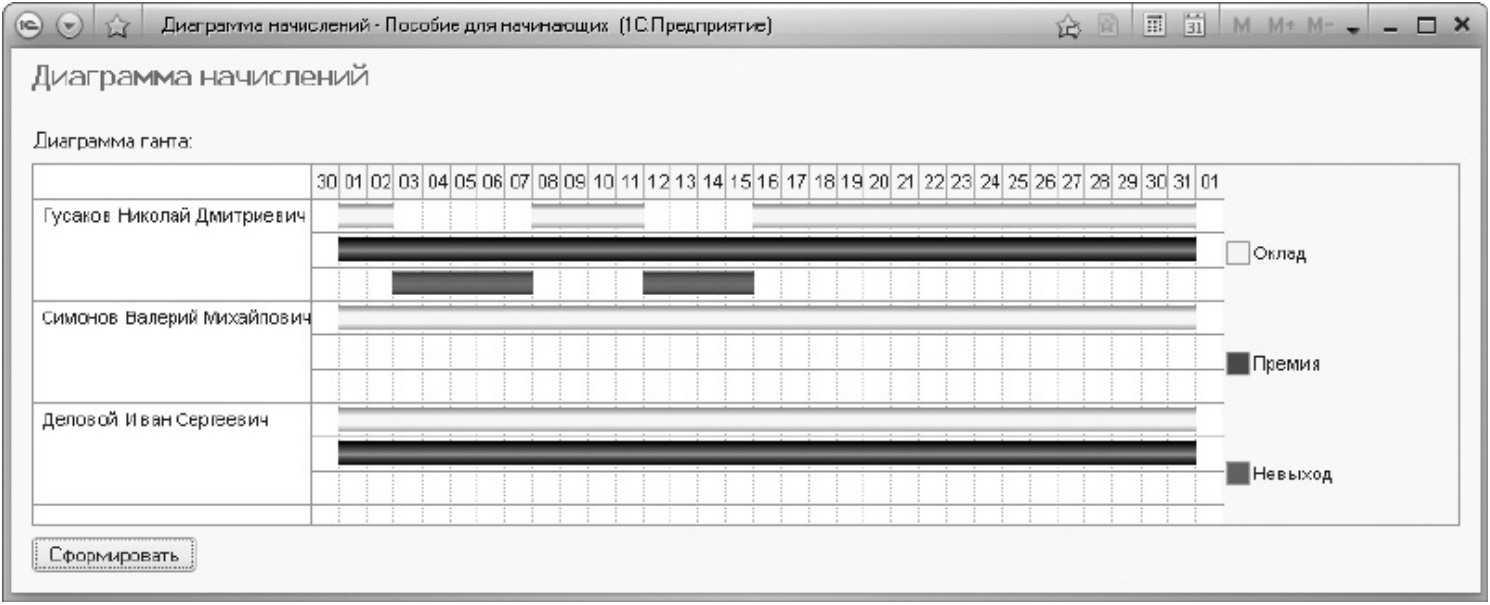


Рис. 18.39. Отчет «Диаграмма начислений»

Теперь вы наглядно видите, как записи вида расчета *Невыход* вытеснили по периоду действия запись расчета *Оклад*, изменив ее фактический период действия.

Следует отметить, что существует также возможность интерактивной настройки параметров диаграммы Ганта, доступная через пункт контекстного меню *Настройка…*

Контрольные вопросы

- *Как создать движения документа по регистру расчета.*
- *Как запросом получить записи перерасчета.*
- *Как работает перерасчет.*
- *Как рассчитать записи регистра расчета.*
- *Как запросом получить данные графика и базы.*
- *Как выполнить перерасчет отдельных записей регистра расчета.*
- *Как получить запросом записи регистра расчета.*
- *Как получить запросом фактический период действия записей регистра расчета.*
- *Для чего используется диаграмма Ганта.*
- *Как устроена диаграмма Ганта.*
- *Как заполнить диаграмму Ганта данными.*

Занятие 19 (1:30). Поиск в базе данных

Продолжительность

Ориентировочная продолжительность занятия – 1 час 30 минут.

На этом занятии вы познакомитесь с механизмом полнотекстового поиска в данных и разработаете отчет, который позволит выполнять поиск в базе данных на основе полнотекстового индекса. На примере этого отчета вы научитесь разрабатывать форму с нуля, наполняя ее реквизитами и командами.

Информационная база нашей фирмы ООО «На все руки мастер» пока еще очень мала. В самом деле, в процессе создания и проверки работы конфигурации мы добавили в нее всего лишь несколько элементов номенклатуры, провели небольшое количество документов.

Однако реальные информационные базы содержат гораздо больше разнообразной информации, и иногда поиск нужных данных становится достаточно сложной задачей, особенно для пользователя, плохо знакомого с номенклатурой товаров (услуг) или с перечнем контрагентов, с которыми работает фирма.

Специально для того, чтобы облегчить поиск незнакомой информации в базе

данных, система «1С:Предприятие 8» содержит механизм полнотекстового поиска в данных. Преимущества этого механизма заключаются в том, что он позволяет искать данные, вводя поисковый запрос в простой и естественной форме, например: «телефон абдулова». При этом можно использовать специальные операторы, наподобие тех, что применяются при поиске в Интернете (*И*, *ИЛИ*, *НЕ* и т. д.).

Полнотекстовый поиск чрезвычайно удобен в тех случаях, когда мы не знаем точно, где находятся нужные нам данные (например, в каком справочнике). Также полнотекстовый поиск оказывается незаменим тогда, когда мы не знаем точно, что нужно искать (например, не помним точное название номенклатуры или контрагента).

Кроме этих возможностей полнотекстовый поиск позволяет находить данные там, где другие методы поиска крайне трудоемки или требуют создания специальных алгоритмов и обработок. Например, полнотекстовый поиск хорошо умеет работать с текстовыми полями большой длины и полями типа *ХранилищеЗначения*.

На этом занятии мы познакомимся с общими сведениями о механизме полнотекстового поиска в данных, создадим полнотекстовый индекс и разработаем отчет, который позволит выполнять полнотекстовый поиск в базе данных нашего ООО «На все руки мастер».

Общие сведения о механизме полнотекстового поиска в данных

Механизм полнотекстового поиска «1С:Предприятия 8» основан на использовании двух составляющих:

- полнотекстового индекса,
- средств выполнения полнотекстового поиска.

Для того чтобы была возможность выполнять полнотекстовый поиск, обязательно должен существовать полнотекстовый индекс. Полнотекстовый индекс создается один раз и затем должен периодически обновляться.

Поиск осуществляется по тем данным, которые содержатся в полнотекстовом индексе. Таким образом, если ведется интенсивная работа с базой данных (например, данные изменяются, добавляются новые), то полнотекстовый индекс следует обновлять как можно чаще. Если же объем изменяемых или новых данных невелик, то обновление полнотекстового индекса можно выполнять реже, например раз в сутки, в период наименьшей загруженности системы.

Создание и обновление полнотекстового индекса может выполняться как

интерактивно, в режиме *1С:Предприятие*, так и программно, средствами встроенного языка. На этом занятии мы рассмотрим возможности интерактивного индексирования, а на следующем занятии вы узнаете, как можно обновлять полнотекстовый индекс в автоматическом режиме.

В процессе работы информационной базы система отслеживает факт изменения данных в тех объектах конфигурации, которые могут участвовать в полнотекстовом поиске. Такими объектами являются, например, планы обмена, справочники, документы, планы видов характеристик, планы счетов, планы видов расчета, регистры (сведений, накопления, бухгалтерии, расчета), бизнес-процессы и задачи.

Впоследствии при создании или обновлении полнотекстового индекса система анализирует данные, содержащиеся в реквизитах этих объектов, и включает эти данные в полнотекстовый индекс. При этом анализироваться могут не все реквизиты, а только те, которые имеют тип *Строка*, *Число*, *Дата*, *ХранилищеЗначения* или ссылочный тип (например, *СправочникСсылка.Номенклатура*).

Собственно сам полнотекстовый поиск выполняется средствами встроенного языка. Немного забегая вперед, необходимо отметить, что полнотекстовый поиск выполняется в соответствии с правами пользователя. Таким образом, если какая-то информация недоступна данному пользователю, он не сможет

получить ее и при помощи полнотекстового поиска.

Результаты полнотекстового поиска возвращаются порциями, и кроме этого, они отсортированы в определенном порядке. Это сделано для того, чтобы с большой долей вероятности пользователь получал требуемые ему данные в начале первой порции. Как показывает практика, при правильно составленном поисковом запросе требуемые данные возвращаются в первой тройке-пятерке результатов.

Теперь, когда мы в общем виде представляем себе, как работает полнотекстовый поиск, приступим к первой части необходимых действий – созданию полнотекстового индекса.

Второй частью будет создание отчета, который будет собственно выполнять полнотекстовый поиск, используя созданный нами индекс.

Полнотекстовый индекс

Прежде всего, познакомимся со свойствами конфигурации и ее объектов, которые отвечают за полнотекстовый поиск.

В режиме «Конфигуратор»

Каждый объект конфигурации, данные которого могут участвовать в

полнотекстовом индексировании, имеет свойство *ПолнотекстовыйПоиск*. По умолчанию при создании нового объекта это свойство установлено в значение *Использовать*.

Таким образом, в данный момент от нас не требуется вносить какие-либо изменения, но, тем не менее, для знакомства откроем палитру свойств объекта конфигурации *СправочникНоменклатура* (рис. 19.1).

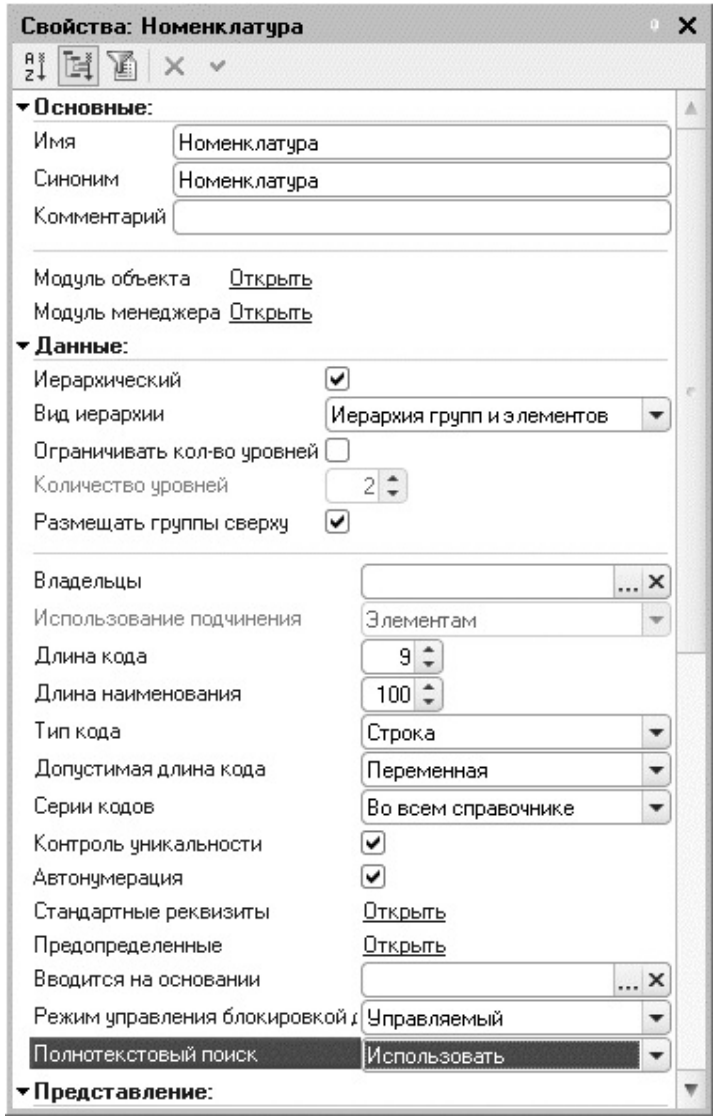

Свойства: Номенклатура
Основные:
Имя
Номенклатура
Синоним
Номенклатура
Комментарий
Модуль объекта
Открыть
Модуль менеджера
Открыть
Данные:
Иерархический
Вид иерархии
Иерархия групп и элементов
Ограничивать кол-во уровней
Количество уровней
2
Размещать группы сверху
Владельцы
Использование подчинения
Элементам
Длина кода
9
Длина наименования
100
Тип кода
Строка
Допустимая длина кода
Переменная
Серии кодов
Во всем справочнике
Контроль уникальности
Автонумерация
Стандартные реквизиты
Открыть
Предопределенные
Открыть
Вводится на основании
Режим управления блокировкой
Управляемый
Полнотекстовый поиск
Использовать
Представление:

Рис. 19.1. Свойство «Полнотекстовый поиск»

Кроме объектов конфигурации свойство *Полнотекстовый поиск* существует и у реквизитов этих объектов. Таким образом, мы имеем возможность указывать конкретные реквизиты, данные которых должны участвовать в полнотекстовом индексировании.

По умолчанию для новых реквизитов это свойство также устанавливается в значение *Использовать*, поэтому и в данном случае не требуется вносить каких-либо изменений.

Например, откроем палитру свойств реквизита *ВидНоменклатуры* справочника *Номенклатура* (рис. 19.2).

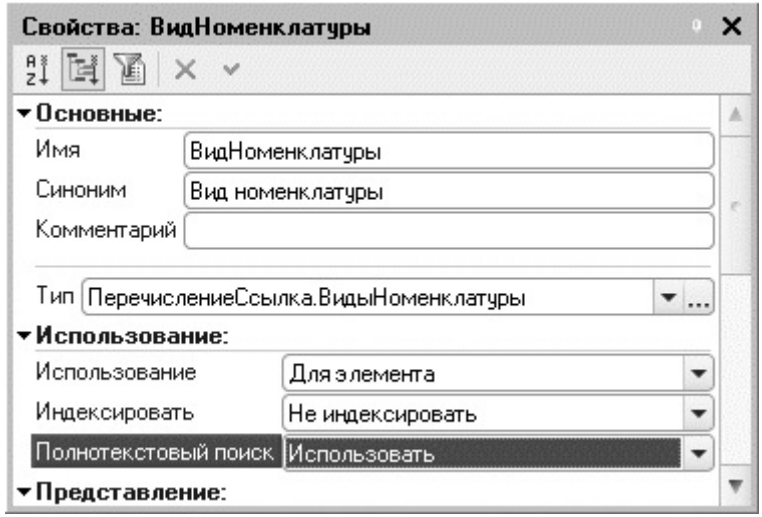


Рис. 19.2. Свойство «Полнотекстовый поиск»

Таким образом, по умолчанию в нашей конфигурации полнотекстовый поиск используется для всех возможных реквизитов всех возможных объектов конфигурации.

Перейдем в режим *1С:Предприятие*.

В режиме «1С:Предприятие»

Выполним команду меню *Все функции* > *Стандартные* > *Управление полнотекстовым поиском…*

В результате будет открыто окно управления полнотекстовым поиском (рис. 19.3).

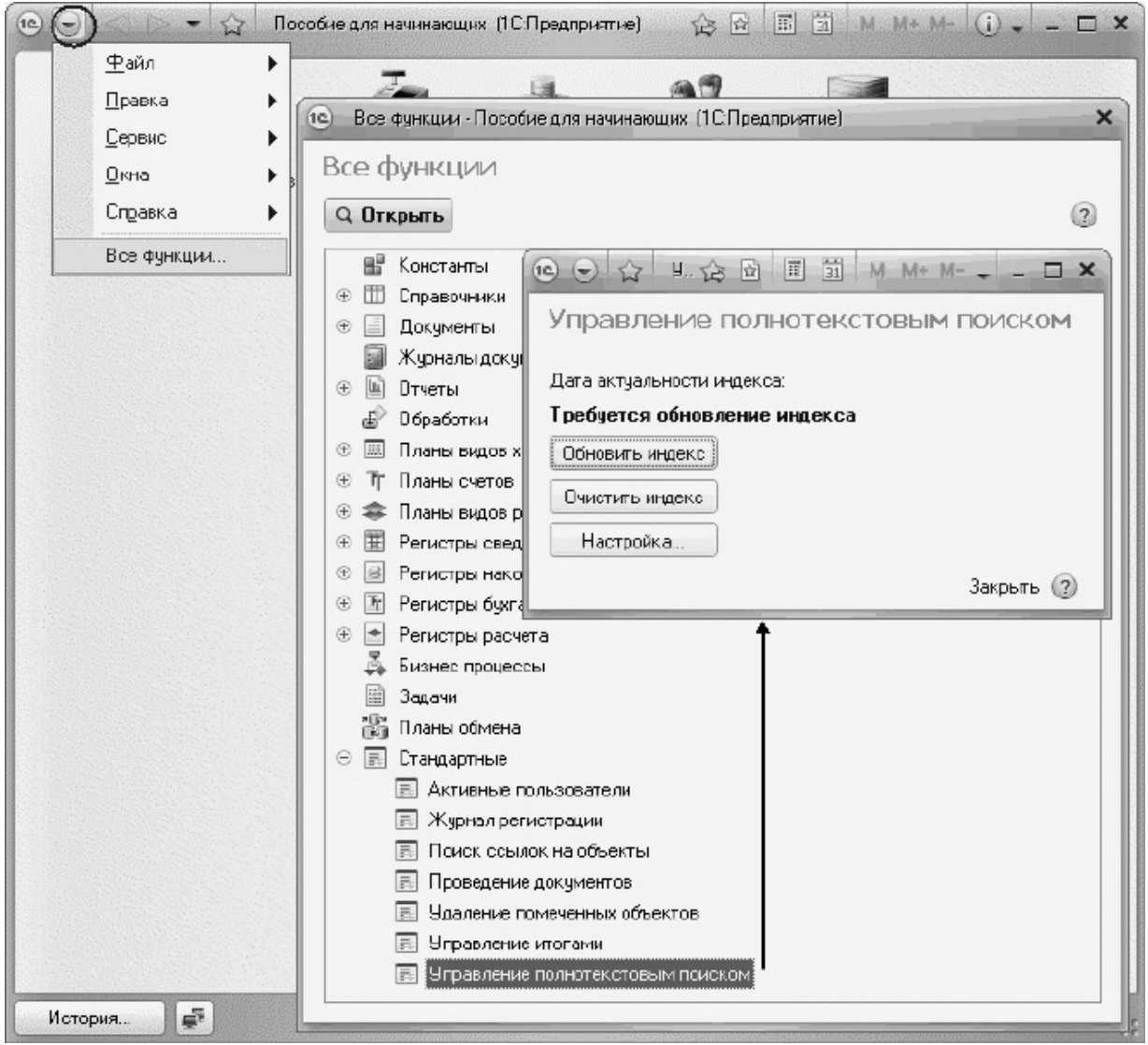

Пособие для начинающих (1С:Предприятие)
Файл
Правка
Сервис
Окно
Справка
Все функции...
Все функции - Пособие для начинающих (1С:Предприятие)
Все функции
Открыть
Константы
Справочники
Документы
Журналы докум
Отчеты
Обработки
Планы видов х
Планы счетов
Планы видов р
Регистры свед
Регистры нако
Регистры бухга
Регистры расчета
Бизнес процессы
Задачи
Планы обмена
Стандартные
Активные пользователи
Журнал регистрации
Поиск ссылок на объекты
Проведение документов
Удаление помеченных объектов
Управление итогами
Управление полнотекстовым поиском
Управление полнотекстовым поиском
Дата актуальности индекса:
Требуется обновление индекса
Обновить индекс
Очистить индекс
Настройка...
Закрыть
История...

Рис. 19.3. Управление полнотекстовым поиском

Это окно позволяет создавать и обновлять полнотекстовый индекс *интерактивно*. Кроме этого, оно позволяет разрешать или запрещать вообще все операции, связанные с полнотекстовым поиском: обновление, очистка полнотекстового индекса, полнотекстовый поиск.

Для того чтобы узнать, разрешены ли сейчас операции полнотекстового поиска, нажмем кнопку *Настройка…* Система откроет окно настройки полнотекстового поиска (рис. 19.4).

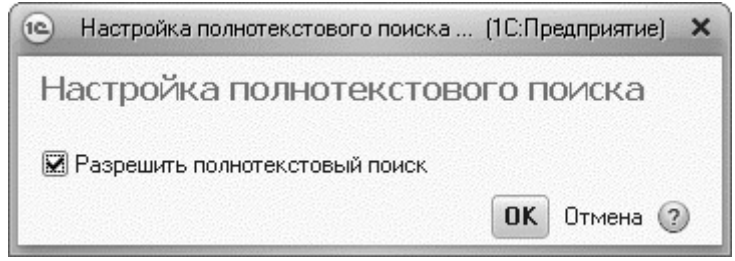


Рис. 19.4. Окно настройки полнотекстового поиска

Как вы видите, по умолчанию полнотекстовый поиск также разрешен, поэтому закроем это окно и вернемся к управлению полнотекстовым поиском.

Как видно из рисунка 19.3, система сообщает нам о том, что требуется обновление полнотекстового индекса. Это действительно так, потому что в

нашем случае индекс вообще отсутствует. Для того чтобы создать (или обновить) полнотекстовый индекс, нажмем кнопку *Обновить индекс*.

Обратите внимание, что при больших размерах информационной базы создание и обновление полнотекстового индекса могут занимать несколько минут. Поэтому в процессе обновления индекса в панель состояния «1С:Предприятия» выводится информация о том, какая часть данных в данный момент обрабатывается.

После того как создание полнотекстового индекса будет закончено, система сообщит об этом (рис. 19.5), и в окне управления полнотекстовым индексированием будет отображена дата актуальности полнотекстового индекса – дата, когда последний раз выполнялось обновление индекса (см. рис. 19.6).

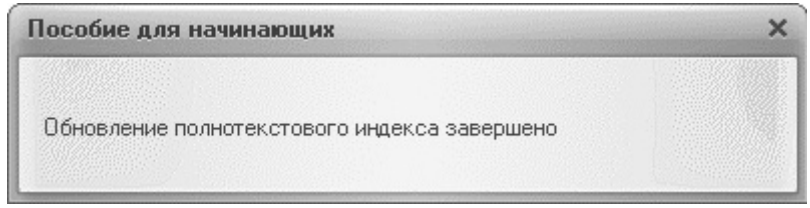


Рис. 19.5. Полнотекстовое индексирование завершено

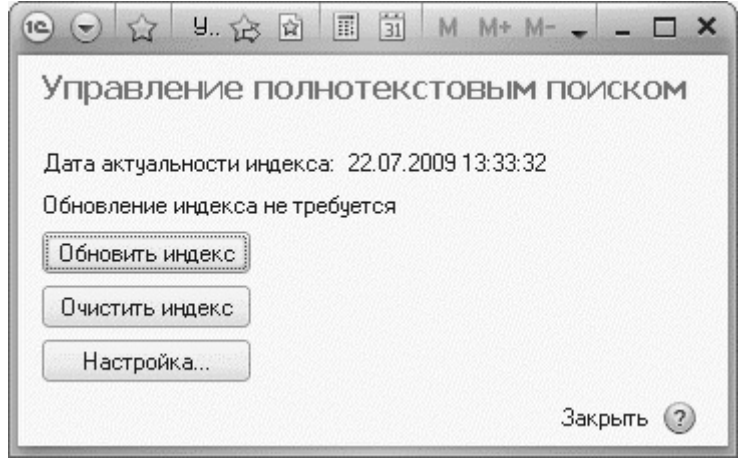


Рис. 19.6. Окно управления полнотекстовым поиском

Итак, мы создали полнотекстовый индекс для нашей информационной базы.

Теперь перейдем к созданию отчета, который позволит выполнять полнотекстовый поиск в базе данных (рис. 19.7).

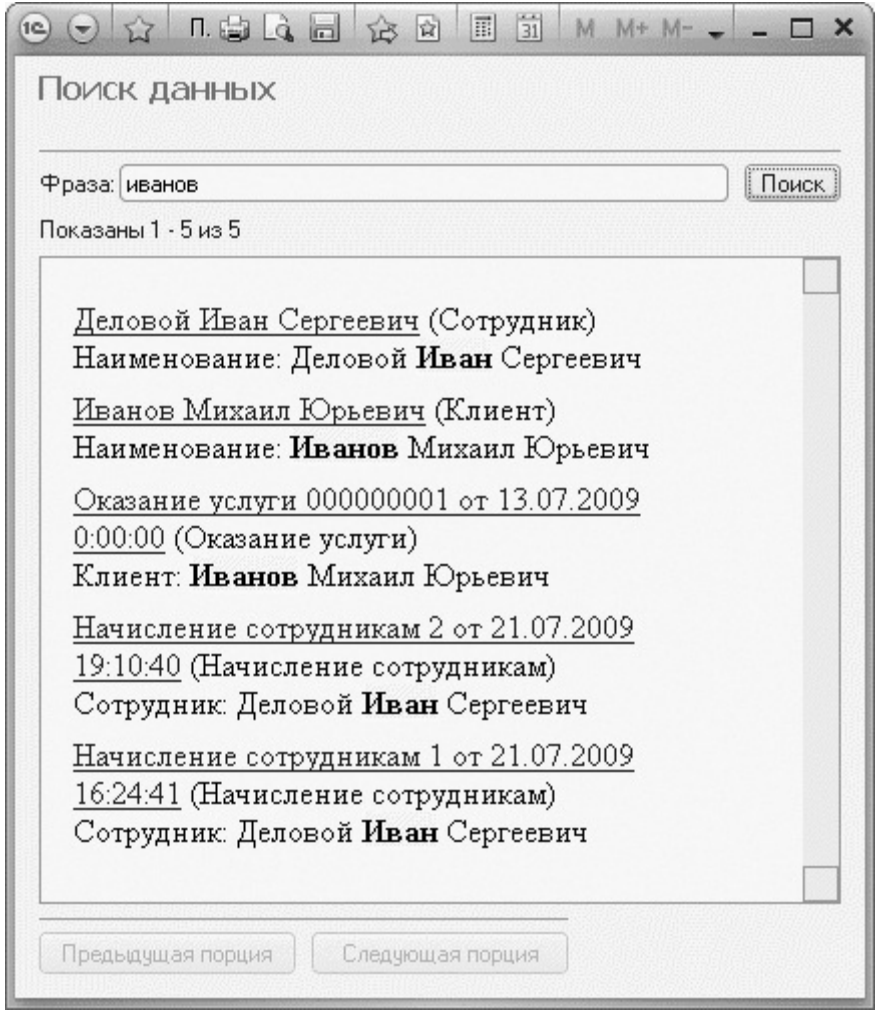


Рис. 19.7. Результат поиска в базе данных

Отчет для поиска данных

В режиме «Конфигуратор»

Добавим в конфигурацию новый объект *Отчет* с именем *ПоискДанных*.

Перейдем на закладку *Формы*, нажмем кнопку открытия  , создадим основную форму отчета и займемся ее редактированием.

Мы видим, что список элементов формы сейчас пуст. Начнем заполнять его.

Как мы уже говорили, элементы формы обязательно должны быть связаны с данными, иначе они не будут отображены. Поэтому сначала создадим соответствующие реквизиты и команды формы и затем перетащим их в окно элементов форм.

Итак, на закладке *Реквизиты* создадим реквизит *ПоисковоеВыражение* и перетащим его в окно элементов формы. В открывшейся палитре свойств поля *ПоисковоеВыражение* зададим его заголовок *Фраза*.

В поле *Вид* автоматически подставилось значение *Поле ввода*. Именно это нам и нужно (рис. 19.8).

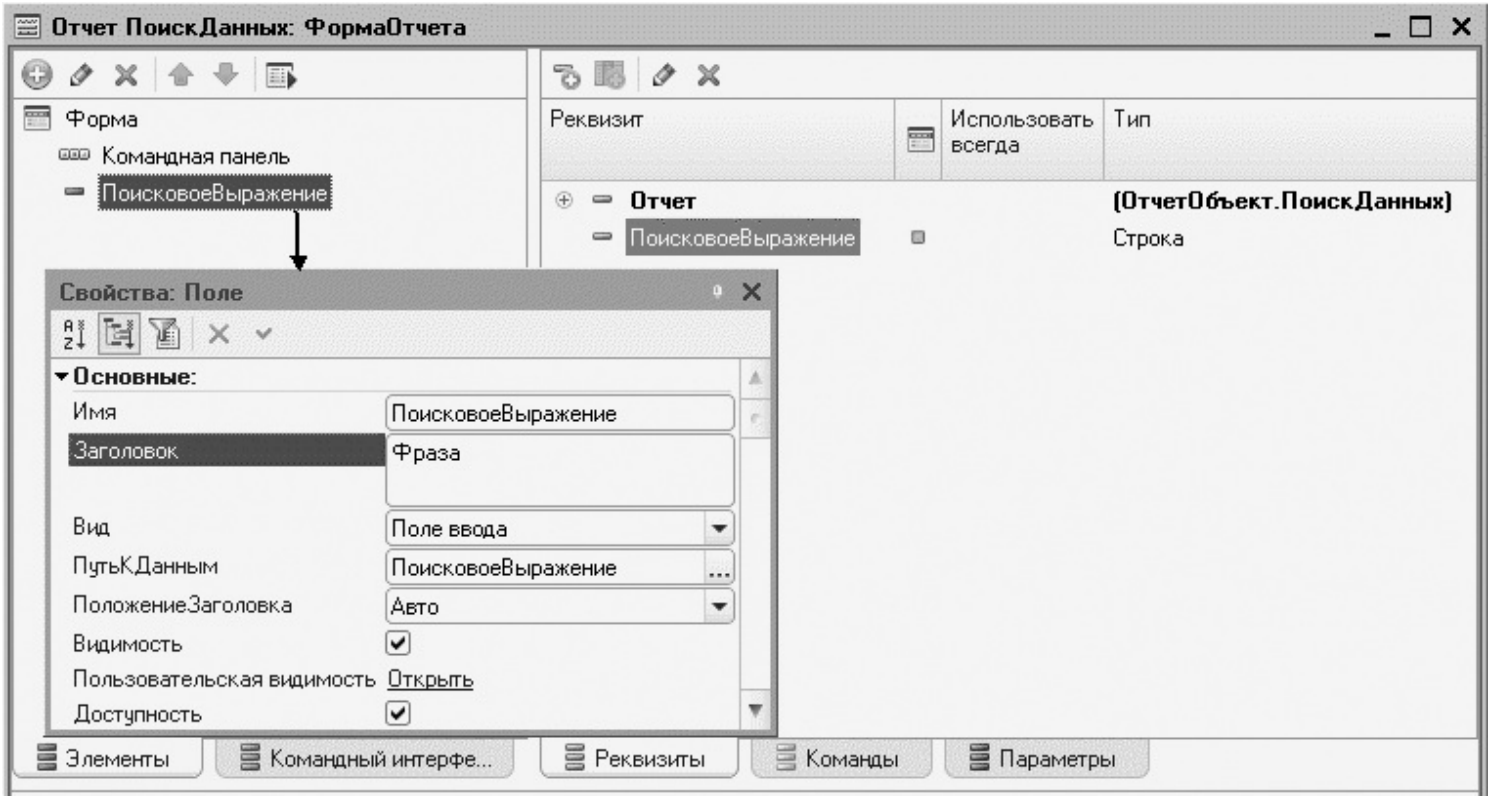


Рис. 19.8. Создание реквизитов, команд и элементов формы

В поле ввода *ПоисковоеВыражение* мы будем вводить фразу для поиска в базе данных. Затем на закладке *Команды* создадим команду *Поиск* и нажмем кнопку открытия в строке *Действие*.

Шаблон обработчика команды, открывшийся в модуле формы, пока заполнять

не будем, а перейдем на закладку *Форма* и перетащим эту команду в окно элементов формы.

В открывшейся палитре свойств кнопки *Поиск* установим флажок *КнопкаПоУмолчанию* (рис. 19.9).

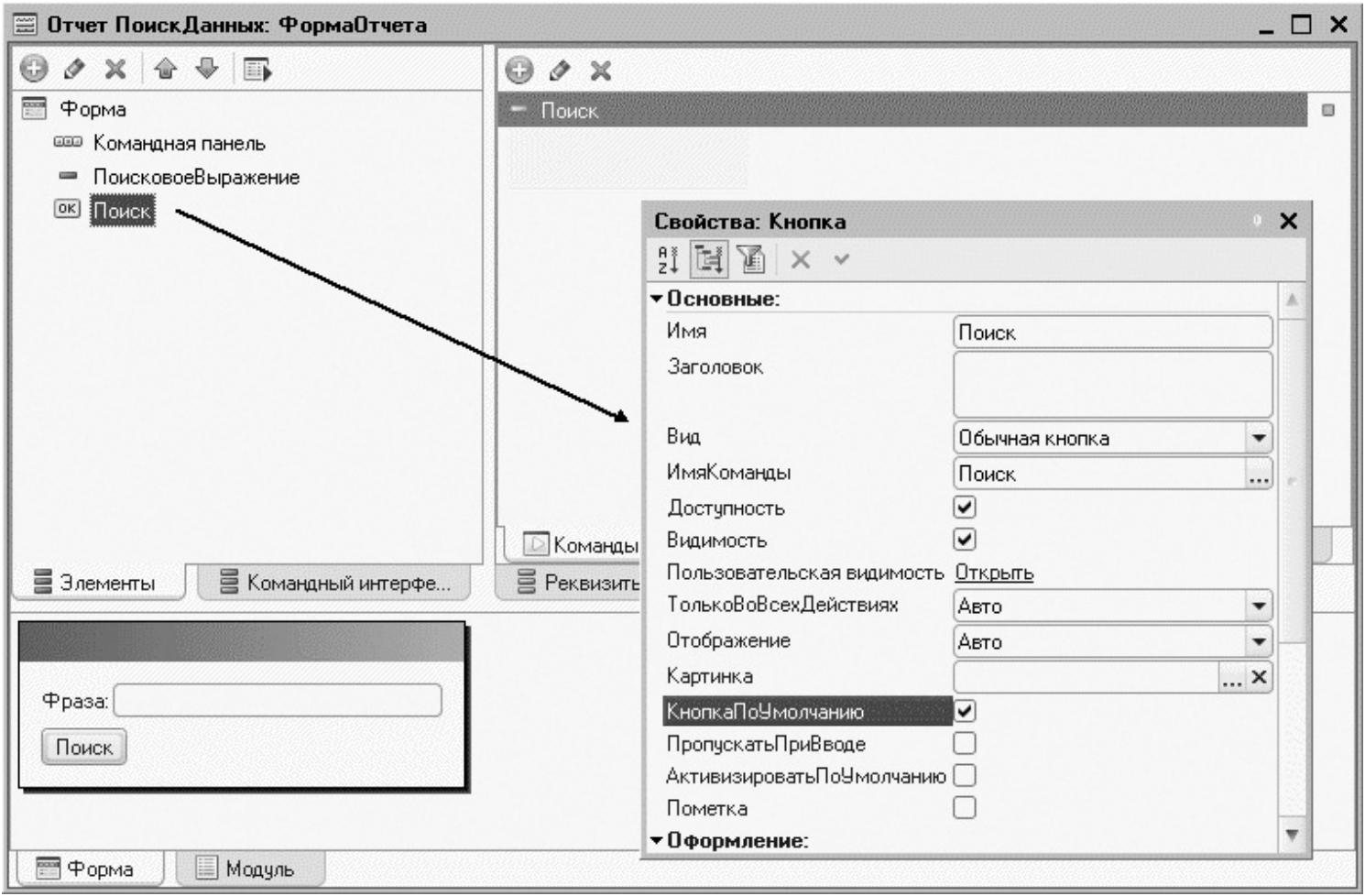


Рис. 19.9. Создание реквизитов, команд и элементов формы

Однако мы видим, что все добавляемые элементы формы располагаются

вертикально – друг под другом. Так происходит потому, что если посмотреть свойства формы в целом (строка *Форма*), то мы увидим, что группировка элементов формы по умолчанию *Вертикальная* (рис. 19.10).

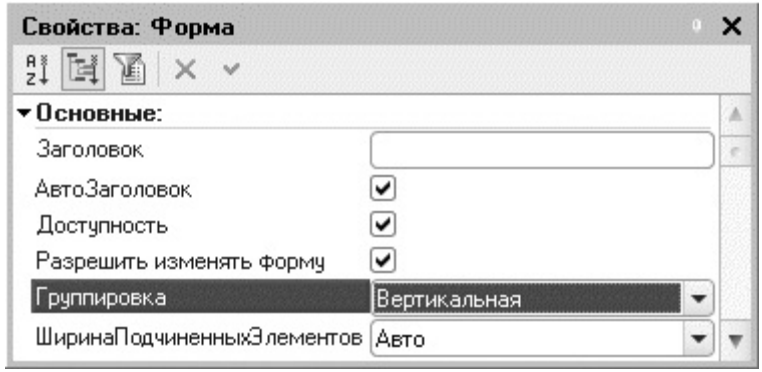


Рис. 19.10. Свойства формы

Нас это устраивает, но некоторые элементы формы, в частности поле *ПоисковоеВыражение* и кнопку *Найти*, хотелось бы расположить горизонтально – рядом друг с другом.

Для этого нужно добавить в форму группу и определить в ней тип группировки элементов *Горизонтальная*.

Выделим строку *Форма* в дереве элементов формы, нажмем кнопку *Добавить* в командной панели и выберем тип элемента *Группа – Обычная группа*.

В открывшейся палитре свойств группы зададим тип группировки *Горизонтальная* (рис. 19.11).

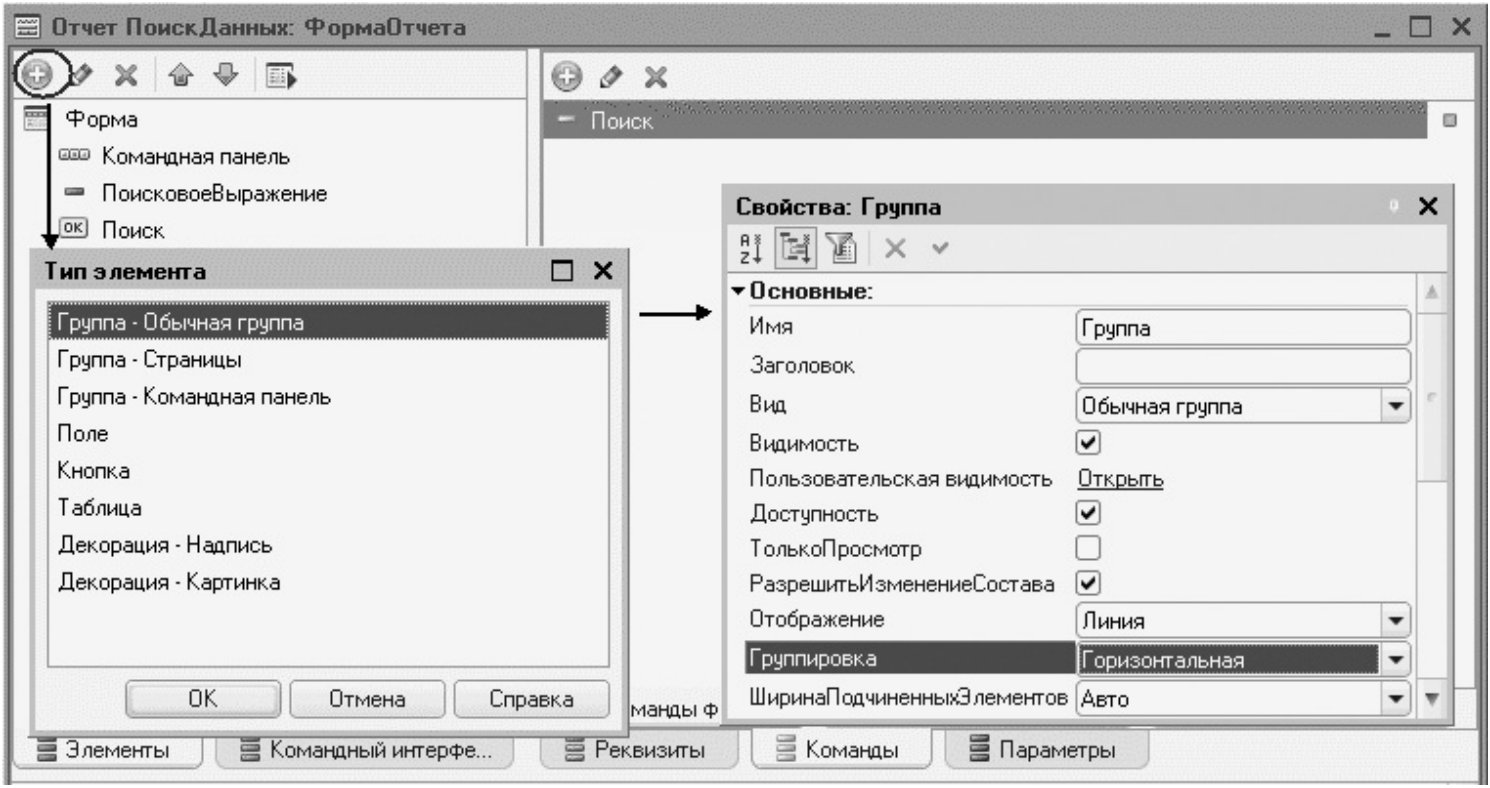


Рис. 19.11. Свойства группы формы

Затем мышью перетащим в эту группу элементы *ПоисковоеВыражение* и *Поиск*.

Теперь мы добились желаемого расположения элементов (рис. 19.12).

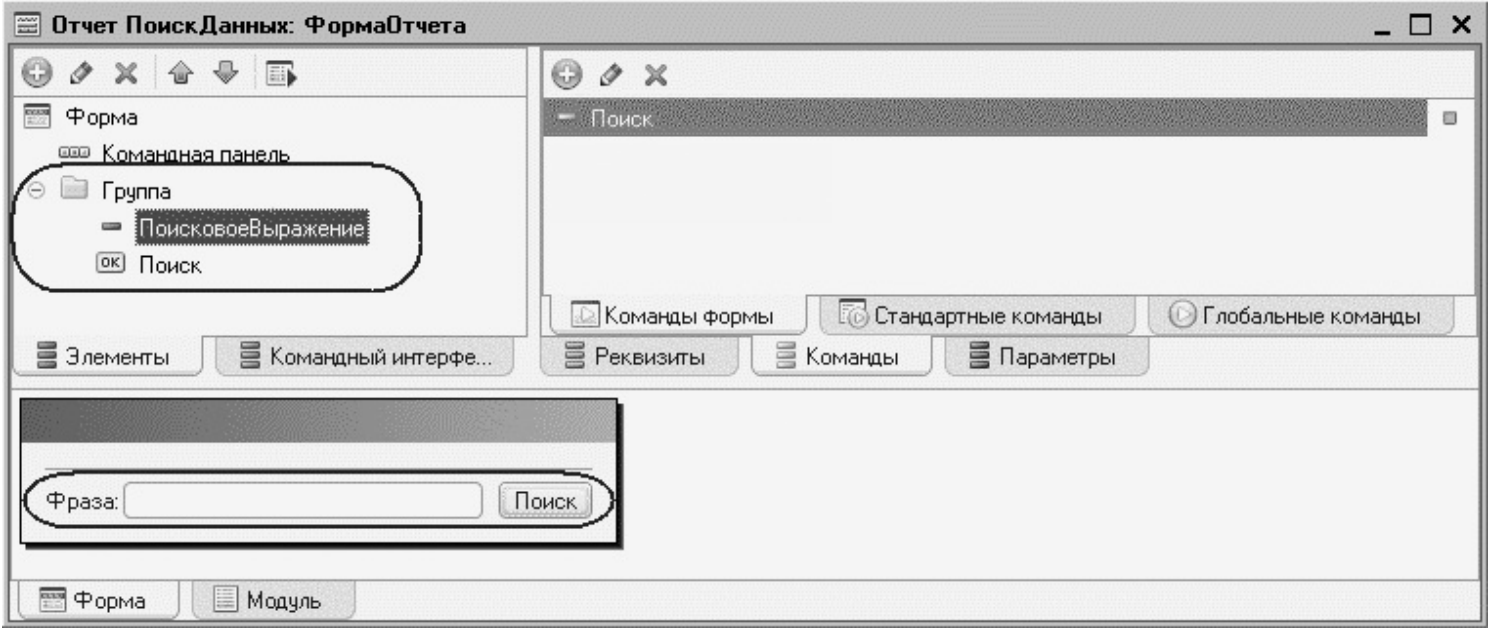


Рис. 19.12. Создание реквизитов, команд и элементов формы

Добавим в форму реквизит *СообщениеОРезультате* и перетащим его в окно элементов формы.

В открывшейся палитре свойств этого поля зададим *ПоложениеЗаголовка* в значение *Нет*. В поле *Вид* установим значение *Поле надписи* (рис. 19.13).

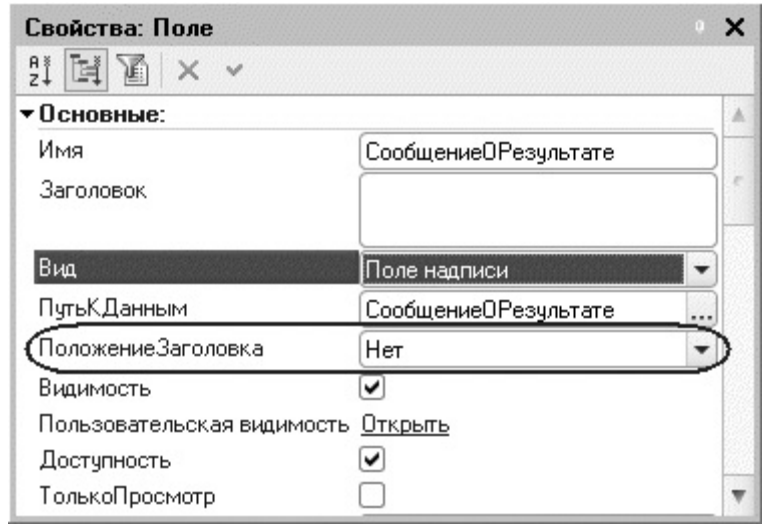


Рис. 19.13. Свойства поля

В поле надписи *СообщениеОРезультате* мы будем выводить сообщение о результате поиска. Добавим в форму реквизит *РезультатПоиска* и перетащим его в окно элементов формы. В открывшейся палитре свойств этого поля зададим *ПоложениеЗаголовка* в значение *Нет*. В поле *Вид* установим значение *Поле HTML документа* (рис. 19.14).

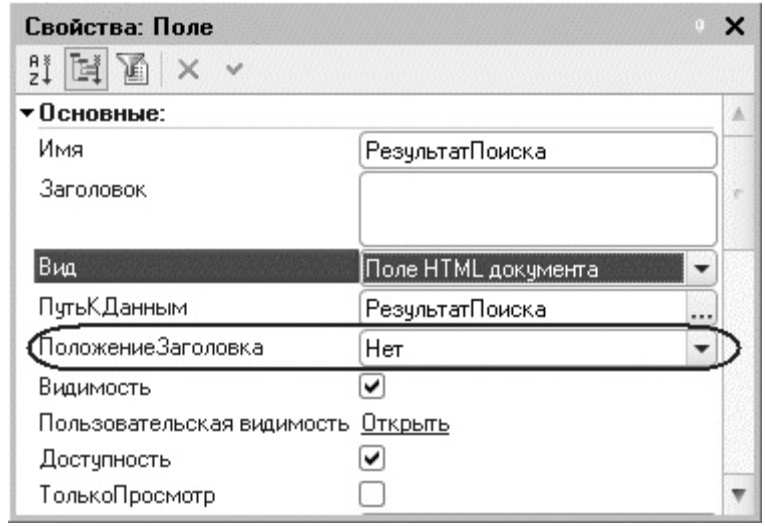


Рис. 19.14. Свойства поля

В поле HTML-документа *РезультатПоиска* мы будем выводить найденные элементы поиска. Затем на закладке *Команды* поочередно создадим команды *ПредыдущаяПорция* и *СледующаяПорция*.

Нажмем кнопку открытия в строке *Действие* для каждой команды. Шаблоны обработчиков событий пока заполнять не будем, а перейдем на закладку *Форма*, выделим корень дерева элементов и добавим новую группу. Зададим для этой группы тип группировки *Горизонтальная*. Затем поочередно перетащим наши команды в эту группу.

В результате форма примет вид (рис. 19.15).

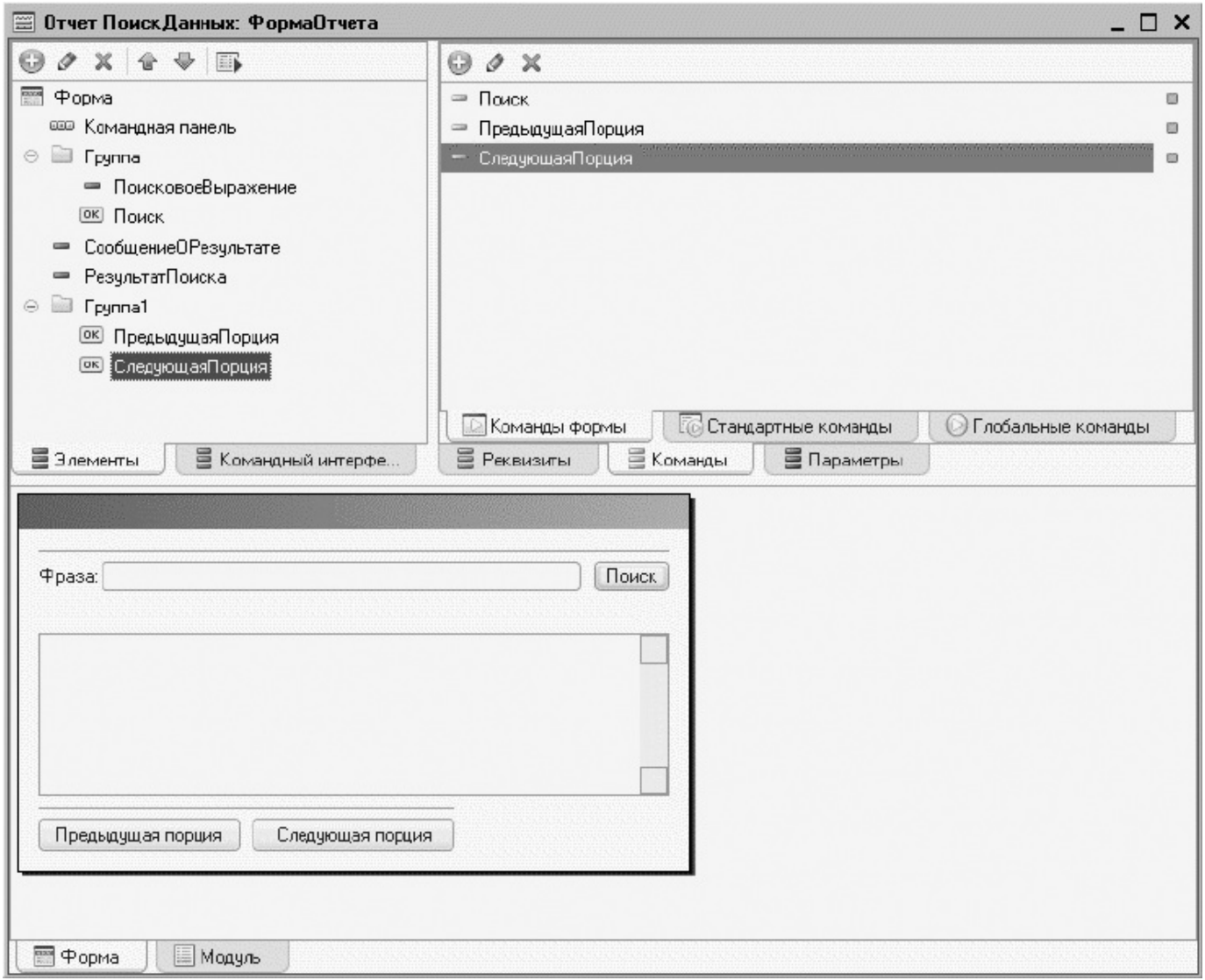

Отчет ПоискДанных: ФормаОтчета
Форма
Командная панель
Группа
ПоисковоеВыражение
Поиск
СообщениеОРезультате
РезультатПоиска
Группа1
ПредыдущаяПорция
СледующаяПорция
Элементы
Командный интерфе...
Поиск
ПредыдущаяПорция
СледующаяПорция
Команды формы
Стандартные команды
Глобальные команды
Реквизиты
Команды
Параметры
Фраза:
Поиск
Предыдущая порция
Следующая порция
Форма
Модуль

Рис. 19.15. Создание реквизитов, команд и элементов формы

Затем добавим в форму реквизит *РезультатыПоиска* типа *СписокЗначений* для хранения найденных элементов поиска. А также добавим в форму реквизит *ТекущаяПозиция* типа *Число* для хранения текущей позиции поиска.

Эти реквизиты имеют вспомогательную роль, и в форму их перетаскивать не нужно.

Теперь реализуем работу формы с помощью кода на встроенном языке.

Для обработчиков событий нажатия кнопок *Поиск*, *Предыдущая порция* и *Следующая порция* напишем код, который позволит нам выполнять поиск в соответствии с направлением поиска (искать сначала, искать вперед или назад), листинг 19.1.

Листинг 19.1. Обработчики событий нажатия кнопок «Поиск», «ПредыдущаяПорция», «СледующаяПорция»

```
&НаКлиенте
Процедура Поиск()

  Искать(0);

КонецПроцедуры
```

```
&НаКлиенте
Процедура ПредыдущаяПорция()

  Искать(-1);

КонецПроцедуры

&НаКлиенте
Процедура СледующаяПорция()

  Искать(1);

КонецПроцедуры
```

Все эти обработчики вызывают процедуру *Искать()*. В ней проверяется, задано ли выражение для поиска, и вызывается собственно процедура полнотекстового поиска, выполняющаяся на сервере *ИскатьСервер()*, в которую передается направление поиска (листинги 19.2, 19.3).

Листинг 19.2. Процедура поиска на клиенте

```
&НаКлиенте
// Процедура поиска, получение и отображение результата.
Процедура Искать(Направление)

  Если ПустаяСтрока(ПоисковоеВыражение) Тогда
    Предупреждение("Не задана строка поиска.");
```

```
    Возврат;

  КонецЕсли;

  ИскатьСервер(Направление);

КонецПроцедуры
```

Листинг 19.3. Процедура поиска на сервере

```
&НаСервере
Процедура ИскатьСервер(Направление) Экспорт

  СписокПоиска = ПолнотекстовыйПоиск.СоздатьСписок();
  СписокПоиска.СтрокаПоиска = ПоисковоеВыражение;

  Если Направление = 0 Тогда
    СписокПоиска.ПерваяЧасть();

  ИначеЕсли Направление = -1 Тогда
    СписокПоиска.ПредыдущаяЧасть(ТекущаяПозиция);

  ИначеЕсли Направление = 1 Тогда
    СписокПоиска.СледующаяЧасть(ТекущаяПозиция);

  КонецЕсли;

  РезультатыПоиска.Очистить();
  Для Каждого Результат Из СписокПоиска Цикл
    РезультатыПоиска.Добавить(Результат.Значение);
```

```
  КонецЦикла;

  РезультатПоиска =
  СписокПоиска.ПолучитьОтображение(ВидОтображенияПолнотекстовогоПоиска.HTMLТекст)
  ТекущаяПозиция = СписокПоиска.НачальнаяПозиция();
  ПолноеКоличество = СписокПоиска.ПолноеКоличество();

  Если СписокПоиска.Количество() > 0 Тогда
    СообщениеОРезультате = "Показаны " +   Строка(ТекущаяПозиция + 1) + " - " +
    Строка(ТекущаяПозиция + СписокПоиска.Количество()) + " из " +
    Строка(ПолноеКоличество);
    Элементы.СледующаяПорция.Доступность = (ПолноеКоличество - ТекущаяПозиция)
    > СписокПоиска.Количество();
    Элементы.ПредыдущаяПорция.Доступность = (ТекущаяПозиция > 0);

  Иначе
    СообщениеОРезультате = "Не найдено";
    Элементы.СледующаяПорция.Доступность = Ложь;
    Элементы.ПредыдущаяПорция.Доступность = Ложь;

  КонецЕсли;

КонецПроцедуры
```

Сначала в этой процедуре мы создаем список поиска, используя метод *СоздатьСписок()* объекта *ПолнотекстовыйПоиск,* и сохраняем его в переменной *СписокПоиска*.

Затем устанавливаем поисковое выражение, введенное пользователем, в качестве строки поиска для полнотекстового поиска. Затем в зависимости от направления поиска выполняем метод *ПервыйЧасть()*, *ПредыдущаяЧасть()* или *СледующаяЧасть(),* который собственно запускает полнотекстовый поиск и возвращает соответственно первую порцию результатов, либо предыдущую порцию, либо следующую порцию в зависимости от текущей позиции поиска. По умолчанию порция содержит 20 элементов.

Затем мы очищаем список значений *РезультатыПоиска* и заполняем его найденными элементами. Далее получаем результат полнотекстового поиска в виде HTML-текста и сохраняем этот текст в реквизите *РезультатПоиска*, имеющем тип HTML-документа.

После этого мы анализируем количество элементов в списке поиска. Если он не содержит ни одного элемента, то мы выводим в форму соответствующее сообщение. В противном случае мы формируем сообщение о том, какие элементы отображены и сколько всего элементов найдено. В зависимости от того, какая порция полученных результатов отображена, мы устанавливаем доступность кнопок *Предыдущая порция* и *Следующая порция*.

Заключительным штрихом будет создание обработчика события *ПриНажатии* поля HTML-документа *РезультатПоиска*, расположенного в форме.

Дело в том, что результат полнотекстового поиска, представленный в виде HTML-текста, содержит гиперссылки на номера элементов списка поиска. И нам хотелось бы, чтобы при переходе пользователя на эту ссылку система открывала бы форму того объекта, который содержится в этом элементе списка.

Для этого создадим обработчик события *ПриНажатии* поля HTML-документа *РезультатПоиска*, в котором будем получать номер элемента списка из гиперссылки и открывать форму соответствующего объекта (рис. 19.16, листинг 19.4).

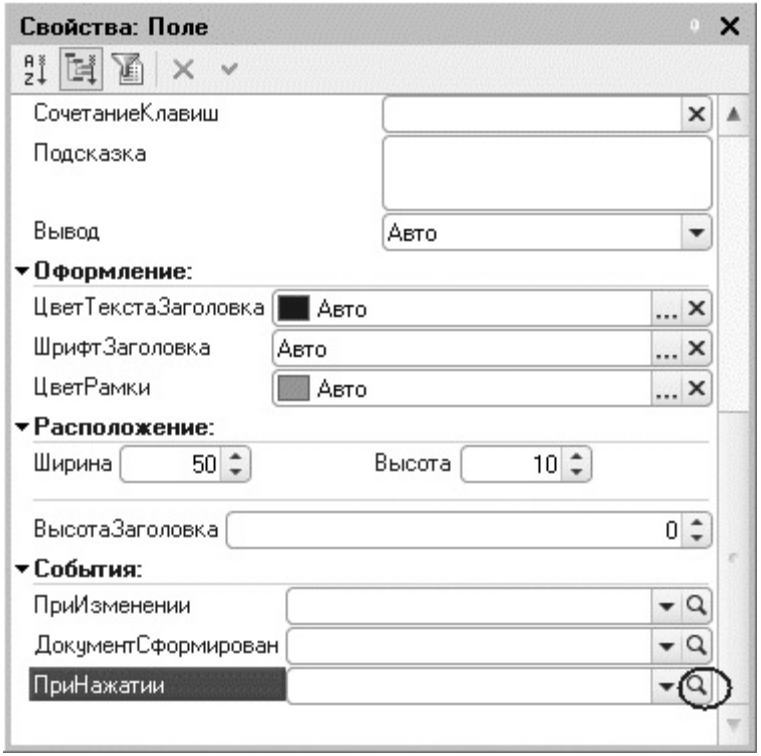


Рис. 19.16. Создание обработчика события «ПриНажатии» поля «РезультатПоиска»

Листинг 19.4. Обработчик события «ПриНажатии()» поля «РезультатПоиска»

```
&НаКлиенте
Процедура РезультатПоискаПриНажатии(Элемент, ДанныеСобытия,
СтандартнаяОбработка)

  ЭлементHTML = ДанныеСобытия.Event.srcElement;
  Если (ЭлементHTML.id = "FullTextSearchListItem") Тогда
```

```
    // Получить имя файла (номер строки списка поиска), содержащегося в
    гиперссылке.
    НомерВСписке = Число(ЭлементHTML.nameProp);

    // Получить строку списка поиска по номеру.
    ВыбраннаяСтрока = РезультатыПоиска[НомерВСписке].Значение;

    // Открыть форму найденного объекта.
    ОткрытьЗначение(ВыбраннаяСтрока);
    СтандартнаяОбработка = Ложь;

  КонецЕсли;

КонецПроцедуры
```

В заключение в окне редактирования объекта конфигурации *Отчет ПоискДанных* на закладке *Подсистемы* мы отметим все подсистемы, чтобы все пользователи в соответствии с их правами могли пользоваться поиском данных.

В режиме «1С:Предприятие»

На этом создание нашего отчета закончено, запустим «1С:Предприятие» в режиме отладки и посмотрим, какие возможности представляет наш отчет.

Количество данных в нашей базе данных невелико, но даже и на них мы

сможем познакомиться с основными возможностями полнотекстового поиска.

Для начала попробуем найти данные, связанные с Ивановым. Введем *иванов* (рис. 19.17).

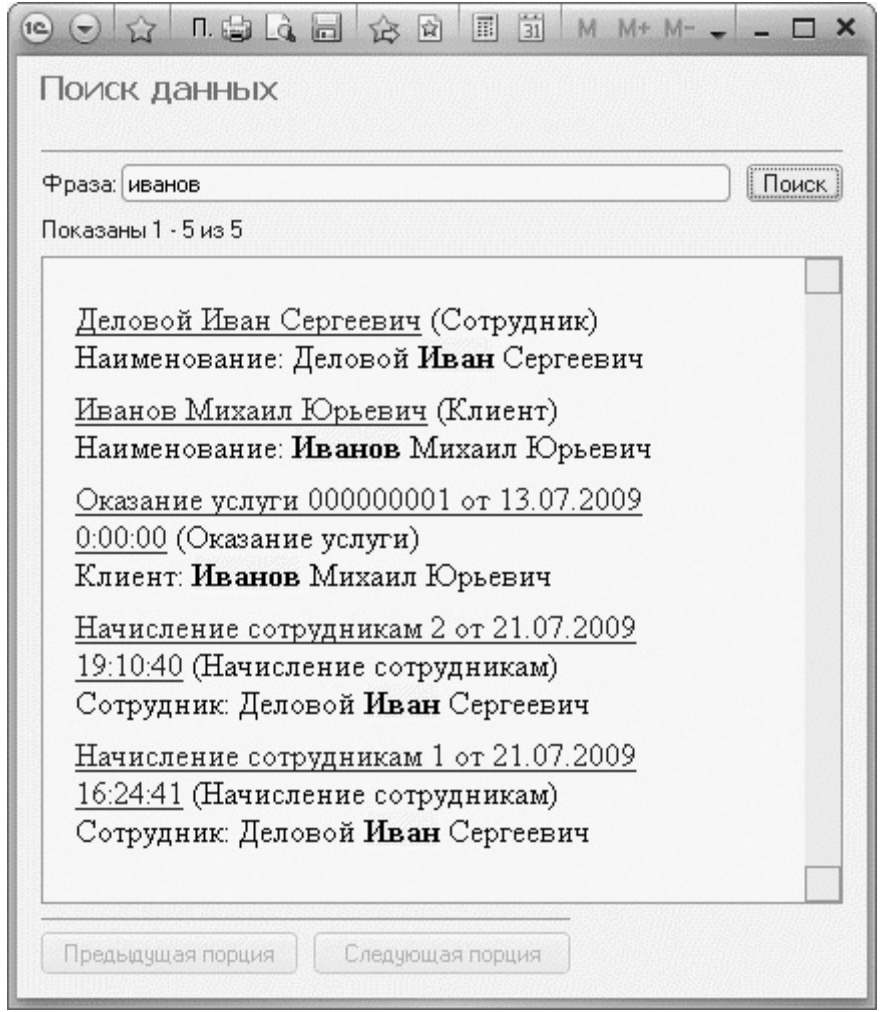


Рис. 19.17. Поиск по выражению «иванов»

Результат поиска содержит 5 элементов, и найденные слова в реквизитах этих элементов выделены желтым фоном.

Обратите внимание, что кроме элемента справочника *Иванов* и документа оказания услуги клиенту *Иванов* база нашла также и другие объекты, которые содержат различные словоформы введенного выражения (в данном случае – *Иван*).

Чтобы выполнить точный поиск по указанному выражению, его необходимо заключить в кавычки (рис. 19.18).

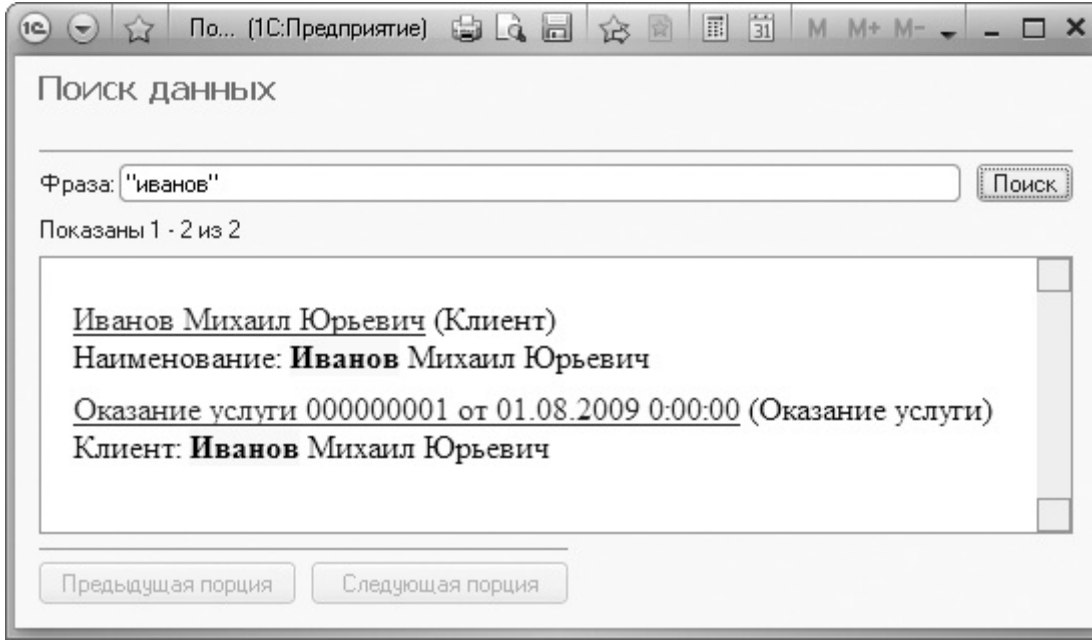


Рис. 19.18. Поиск по выражению «иванов»

Как правило, для получения наилучших результатов полнотекстового поиска рекомендуется использовать в поисковом выражении пару слов.

Например, если требуется узнать, какого числа клиенту *Симонову* заменили трансформатор в телевизоре, можно ввести поисковое выражение *трансформатор Симонов* (рис. 19.19).

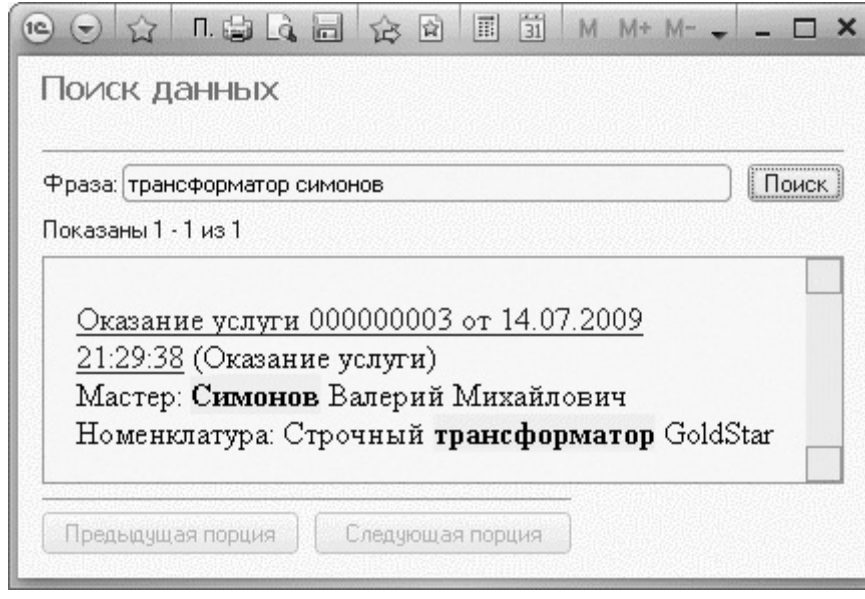


Рис. 19.19. Поиск по выражению «трансформатор Симонов»

При нажатии на гиперссылку система откроет документ *Оказание услуги № 3,* и можно будет просмотреть полный перечень работ, выполненных для этого клиента.

Система индексирует не только данные, содержащиеся в объектах конфигурации, но и имена реквизитов и объектов метаданных.

Поэтому, например, если требуется найти информацию о сотруднике по фамилии Деловой, в поисковом выражении следует указать имя справочника,

который нас интересует: *сотрудник деловой* (рис. 19.20).

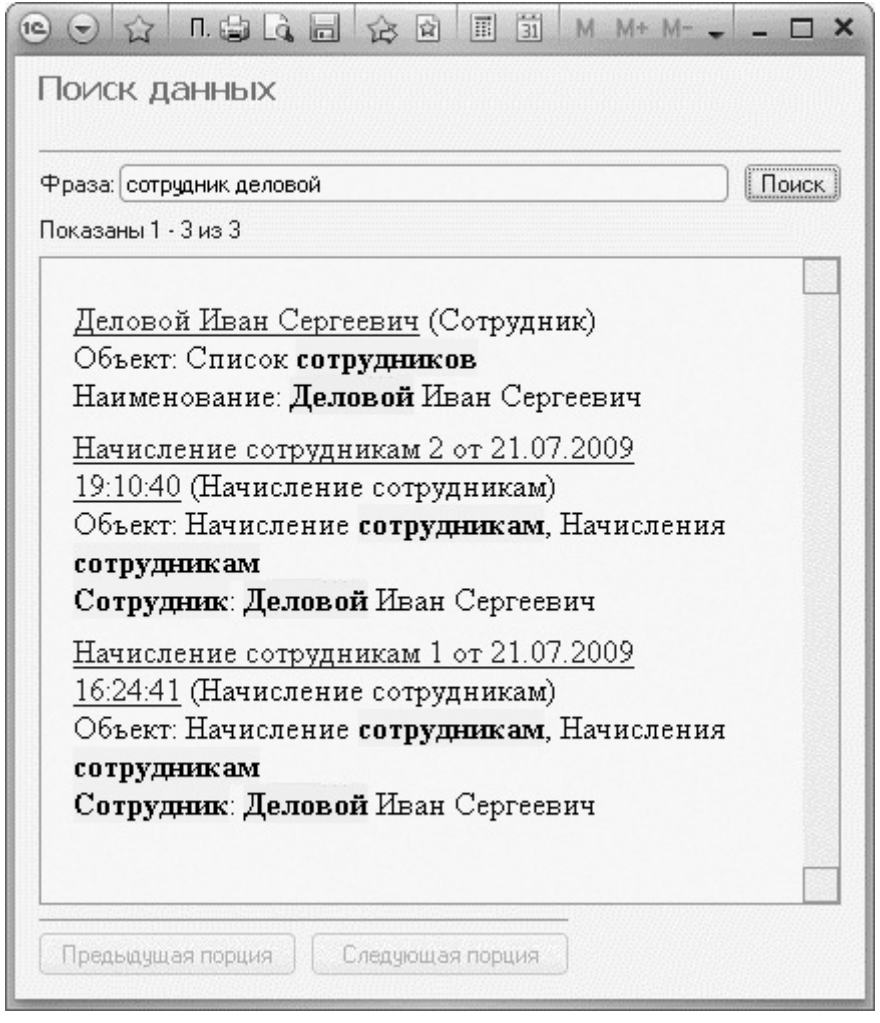


Рис. 19.20. Поиск по выражению «сотрудник Деловой»

Как видите, система нашла также и документы начисления сотрудникам, которые содержат формы слова «сотрудники», но, несмотря на это, искомый нами справочник *Сотрудники* находится первым в списке найденных.

Для составления поисковых выражений можно использовать разнообразные операторы, которые подробно описаны в документации, в книге «1С:Предприятие 8.2. Руководство разработчика», приложение 5 «Поисковые выражения полнотекстового поиска». Их использование не должно вызвать у вас затруднений.

Контрольные вопросы

- *Для чего предназначен полнотекстовый поиск в данных.*
- *Что такое основной полнотекстовый индекс и что такое дополнительный полнотекстовый индекс.*
- *Какова стратегия полнотекстового индексирования информационной базы.*
- *Как создать отчет, выполняющий поиск в данных.*
- *Как составлять простейшие поисковые выражения.*

Занятие 20 (1:00). Выполнение заданий по расписанию

Продолжительность

Ориентировочная продолжительность занятия – 1 час.

ВНИМАНИЕ!

Если вы используете учебную версию платформы «1С:Предприятие 8.2», то пример, приведенный в этом занятии, удастся проверить в работе лишь частично. Для полной проверки примера требуется одновременный запуск двух клиентских сеансов: планировщика заданий и обычного пользовательского сеанса.

Учебная версия позволяет запускать только один пользовательский сеанс. Поэтому их работу можно будет проверить только по отдельности. Сначала запустить планировщик заданий и убедиться, что регламентные задания запускаются. Затем закрыть сеанс планировщика и открыть обычный пользовательский сеанс, чтобы выполнить поиск в данных.

Любая информационная база системы «1С:Предприятие 8» требует периодического выполнения определенного набора регламентных операций.

Например, по мере изменения существующих данных или добавления новых

необходимо выполнять резервное копирование. Тогда, если в результате сбоя информационная база окажется неработоспособной, основную часть данных можно будет восстановить из резервной копии. Поэтому чем чаще выполняется резервное копирование, тем меньшее количество данных придется вводить повторно в случае сбоя.

Другой пример. Для того чтобы использовать полнотекстовый поиск в базе данных, необходимо, чтобы все данные, в которых предполагается выполнять поиск, были проиндексированы. А это значит, что полнотекстовый индекс нужно периодически обновлять. Как часто? Это зависит от интенсивности изменения данных и ввода новых данных. Но очевидно, что делать это нужно с некоторой периодичностью.

Для того чтобы автоматизировать подобные операции, в «1С:Предприятии 8» существует механизм заданий. Этот механизм позволяет создавать задания, каждое из которых представляет собой некоторую последовательность действий, описанных с помощью встроенного языка. Для каждого задания может быть назначено расписание, в соответствии с которым это задание будет автоматически запущено на исполнение.

На этом занятии мы рассмотрим использование механизма заданий на примере автоматизации двух регламентных операций, связанных с полнотекстовым поиском: операции полнотекстового индексирования и операции слияния

индексов.

Мы опишем эти операции средствами встроенного языка, установим расписание для их автоматического выполнения и узнаем, как обеспечить автоматическое выполнение этих заданий по расписанию в случае файлового варианта работы системы «1С:Предприятие 8».

Постановка задачи

На предыдущем занятии мы узнали, что для возможности выполнения полнотекстового поиска обязательно должен существовать полнотекстовый индекс. Полнотекстовый индекс создается один раз, и затем должен периодически обновляться.

На самом деле полнотекстовый индекс состоит из двух индексов: основного и дополнительного. При выполнении полнотекстового поиска поиск осуществляется как в одном, так и в другом индексе. Отличие их заключается в следующем.

Основной индекс спроектирован так, чтобы обеспечивать максимальную скорость поиска при большом объеме данных. Однако оборотной стороной этого является то, что добавление данных в основной индекс осуществляется относительно медленно.

Дополнительный индекс является полной противоположностью основному: добавление данных в дополнительный индекс осуществляется быстро, однако при значительном объеме данных в дополнительном индексе поиск будет выполняться относительно медленно.

Наличие основного и дополнительного индексов предполагает следующую стратегию их использования. Основная масса данных находится в основном индексе, что позволяет выполнять поиск достаточно быстро. Новые данные, измененные или добавленные в систему, добавляются в дополнительный индекс непосредственно в процессе работы системы и пользователей с требуемой периодичностью (например, раз в час или раз в минуту). Такое добавление происходит быстро и оказывает незначительное влияние на производительность системы. Пока объем данных в дополнительном индексе невелик, поиск по нему также выполняется быстро.

В период малой активности пользователей или в период выполнения регламентных действий с информационной базой (например, ночью) выполняется слияние основного и дополнительного индекса (например, раз в сутки). Эта операция может оказывать видимую нагрузку на систему или занимать продолжительное время (в зависимости от накопленных данных). В результате новые данные помещаются в основной индекс, а дополнительный индекс при этом очищается и готов к быстрому приему следующих данных.

Таким образом, чтобы пользователи могли искать во всех данных и не ощущали какого-либо замедления работы системы, дополнительный индекс необходимо обновлять относительно часто (например, раз в час или раз в минуту). В то же время чтобы пользователи выполняли поиск быстро, необходимо, чтобы дополнительный индекс содержал как можно меньше данных, т. е. нужно периодически выполнять слияние основного и дополнительного индексов (например, ночью, в период минимальной активности пользователей).

В результате для автоматизации полнотекстового индексирования нам понадобится два задания. Первое задание будет выполнять индексирование без слияния и запускаться каждую минуту. Второе будет выполнять слияние индексов и запускаться один раз в сутки, ночью.

Приступим к созданию этих заданий.

Что такое регламентное задание

Регламентные задания располагаются в дереве объектов конфигурации, в ветке *Общие*. Каждое регламентное задание содержит два основных свойства: *Имя метода* и *Расписание*.

Свойство *Имя метода* связывает регламентное задание с некоторой процедурой или функцией общего модуля, которая, собственно, и будет

исполняться. Эта процедура должна содержать алгоритм на встроенном языке, описывающий все те операции, которые должны быть выполнены.

Свойство *Расписание* позволяет задать периодичность выполнения этой процедуры.

Кроме перечисленных свойств регламентное задание содержит и другие свойства, например *Интервал повтора при аварийном завершении* и *Количество повторов при аварийном завершении*. Таким образом, если по какой-либо причине выполнение регламентного задания закончится неудачно, система «1С:Предприятие» может автоматически запустить это задание указанное количество раз по прошествии указанного периода времени.

Создание регламентных заданий

В режиме «Конфигуратор»

Сначала создадим первое регламентное задание по обновлению индекса.

Раскроем ветвь *Общие* дерева объектов конфигурации. Выделим строку *Регламентные задания* и добавим новый объект конфигурации Регламентное задание. Зададим его имя – *ОбновлениеИндекса* (рис. 20.1).

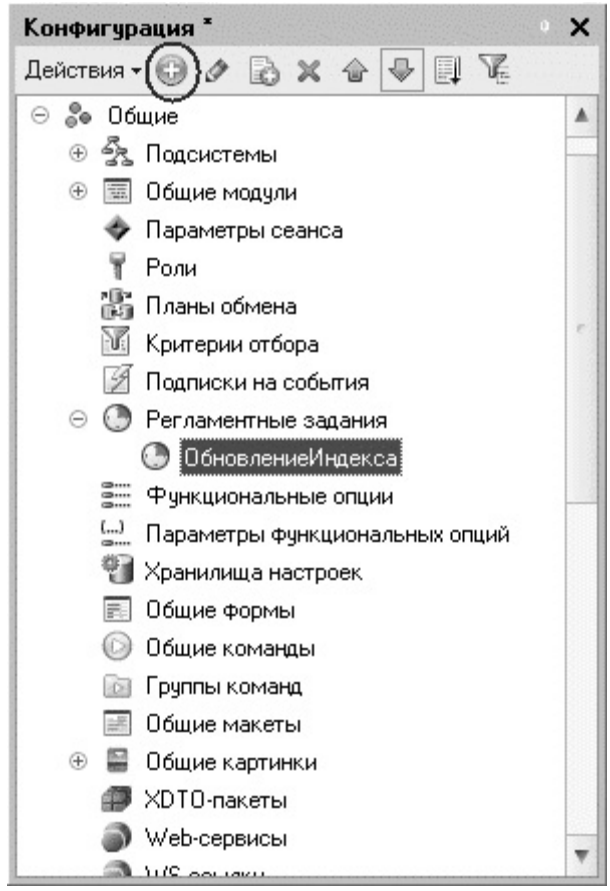


Рис. 20.1. Создание регламентного задания

После этого создадим процедуру, которая и будет выполнять обновление полнотекстового индекса нашей информационной базы.

В качестве такой процедуры может выступать любая процедура или функция неглобального общего модуля, которую можно вызвать на сервере (у общего модуля должно быть установлено свойство *Сервер*).

Добавим в конфигурацию общий модуль с именем *РегламентныеПроцедуры* и установим флажок *Вызов сервера* для видимости его экспортных процедур и функций (рис. 20.2).

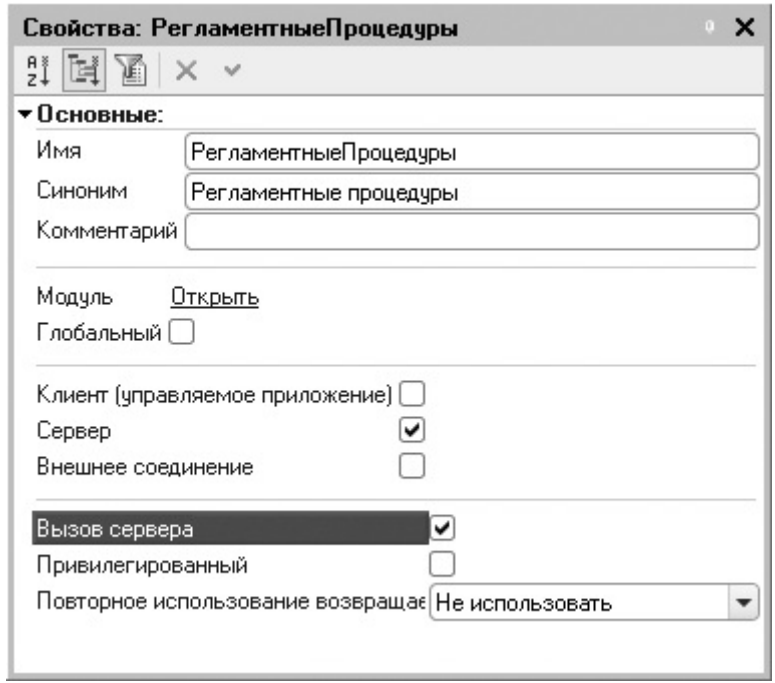


Рис. 20.2. Свойства общего модуля

Вернемся к свойствам регламентного задания *ОбновлениеИндекса*. Нажмем кнопку выбора у поля ввода *Имя метода*.

Система откроет окно выбора общего модуля (рис. 20.3).

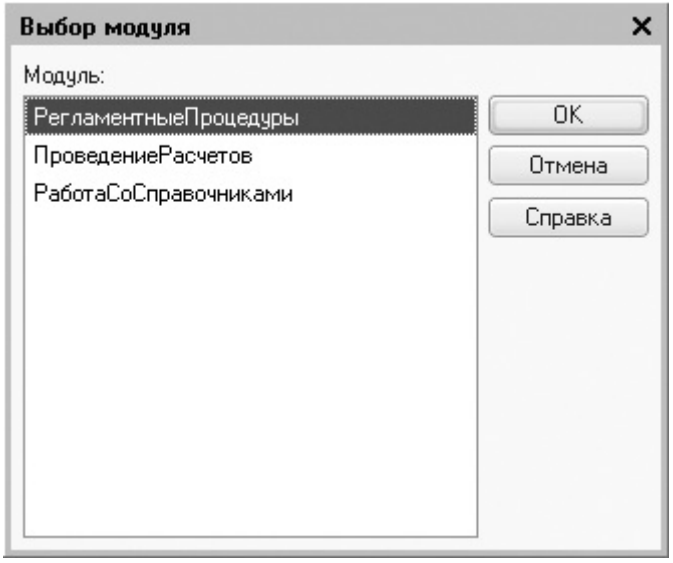


Рис. 20.3. Выбор обработчика события

Выберем модуль *РегламентныеПроцедуры*. В этом модуле будет создан шаблон процедуры *ОбновлениеИндекса()*. Заполним его следующим образом (листинг 20.1).

Листинг 20.1. Процедура обновления индекса

```
Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =
РежимПолнотекстовогоПоиска.Разрешить Тогда

  Если Не ПолнотекстовыйПоиск.ИндексАктуален() Тогда
    ПолнотекстовыйПоиск.ОбновитьИндекс( , Истина);

  КонецЕсли;

КонецЕсли;
```

Сначала в этой процедуре проверяется возможность выполнения операций, связанных с полнотекстовым поиском (ведь они могут быть запрещены, например, интерактивно, см. рис. 19.4.).

Если операции полнотекстового поиска разрешены, проверяется, актуален ли полнотекстовый индекс (если после последнего индексирования данные, подлежащие полнотекстовому индексированию, не изменялись, то индекс будет актуален и повторное индексирование не требуется).

В случае необходимости индексирования вызывается метод *ОбновитьИндекс()* менеджера полнотекстового поиска.

Первый параметр этого метода отвечает за слияние индексов и по умолчанию имеет значение *Ложь*. Это значит, что слияние индексов выполняться не будет.

Второй параметр метода определяет, какое количество данных будет индексироваться: сразу все, которые необходимо проиндексировать, или порциями. Наша задача – выполнить индексирование как можно быстрее, поэтому указываем, что индексирование будет выполняться порциями (значение *Истина*).

Размер одной порции фиксирован – 10 000 объектов. Таким образом, если в данный момент требуется проиндексировать, например, 15 000 объектов, то при вызове этого метода из них будет проиндексировано только 10 000 (первая порция), а оставшиеся объекты будут проиндексированы при следующем вызове этого метода (при следующем запуске нашего регламентного задания).

Перейдем к составлению расписания запуска регламентного задания.

Нажмем на ссылку *Открыть* в свойствах регламентного задания (в строке *Расписание*), и система откроет диалог редактирования расписания (рис. 20.4).

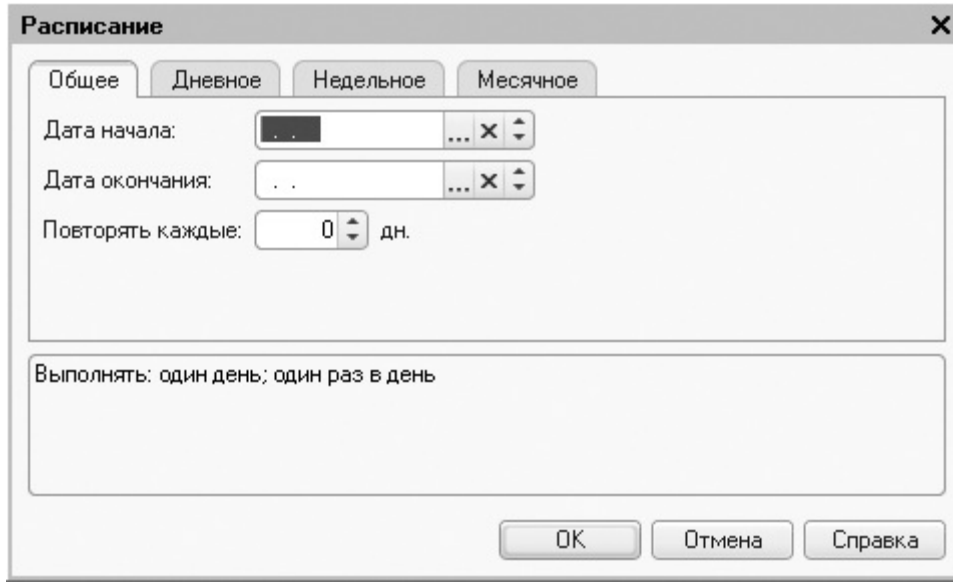


Рис. 20.4. Диалог редактирования расписания

Диалог содержит несколько закладок, которые позволяют задать различные виды расписаний; в нижней части диалога отображается итоговый результат всех установок.

Наша задача – запускать регламентное задание ежедневно, каждую минуту.

Поэтому прежде всего на закладке *Общие* укажем, что запуск задания должен повторяться каждый день (*Повторять каждые: 1 дн.*), рис. 20.5.

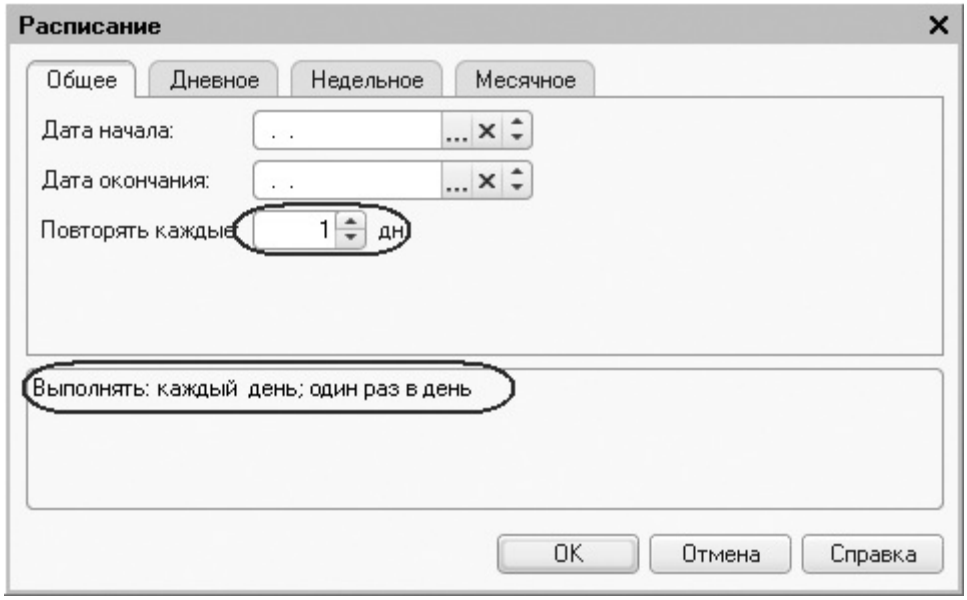


Рис. 20.5. Запуск задания каждый день

Теперь перейдем на закладку *Дневное* и зададим порядок запуска задания в течение дня.

Укажем, что запуск задания должен повторяться каждые 60 секунд (*Повторять через: 60 сек.*), рис. 20.6.

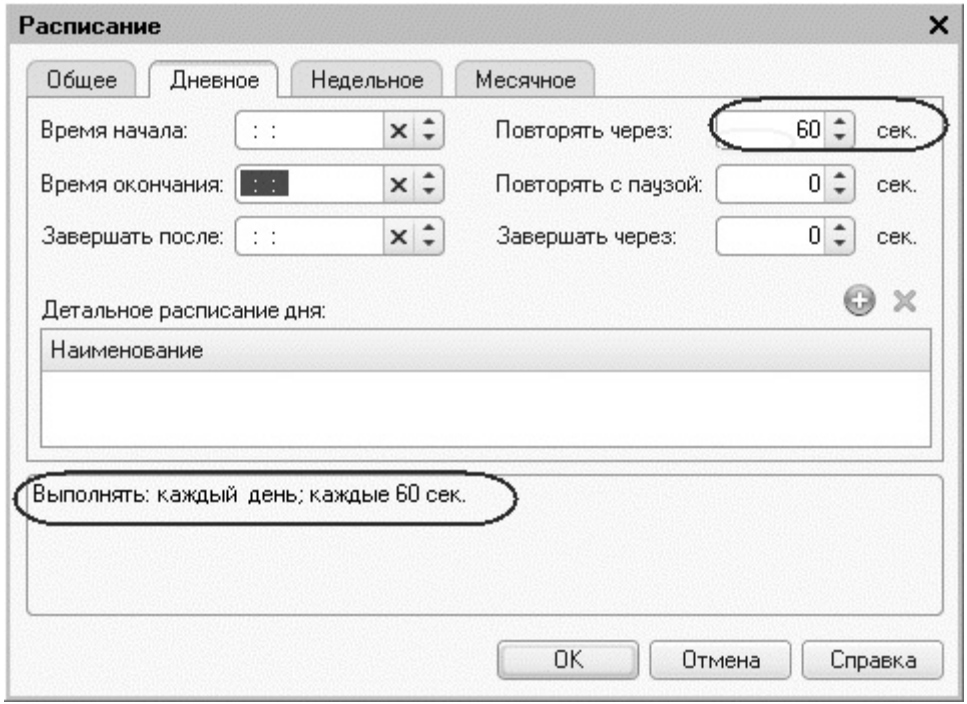


Рис. 20.6. Запуск задания каждую минуту

В нижней части диалога отображено созданное нами расписание запуска: *Выполнять: каждый день; каждые 60 сек*.

Вроде бы мы получили то, что хотели: регламентное задание запускается ежедневно, каждую минуту.

Однако наше ООО «На все руки мастер» не работает круглосуточно, и запуск

этого задания в ночное время будет явно бесполезным – данные в базе данных не изменяются. В то же время вполне возможна ситуация, когда некоторые сотрудники задерживаются после окончания рабочего дня.

Поэтому доработаем расписание следующим образом: укажем *Время начала: 08:00* (рис. 20.7).

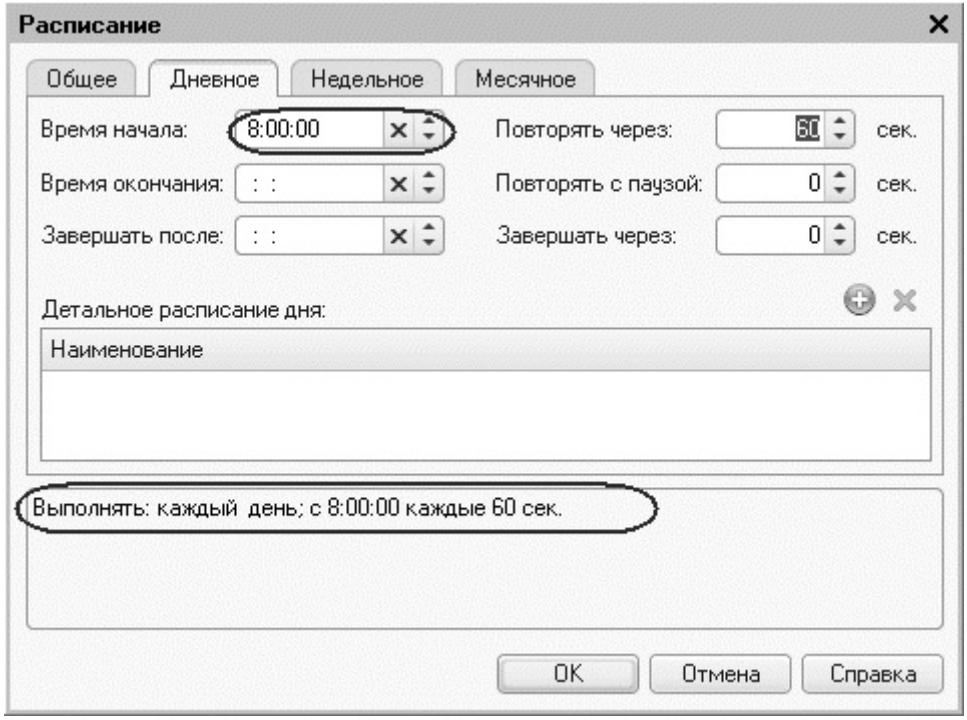


Рис. 20.7. Указание времени начала запуска

В результате запуск задания будет выполняться не круглые сутки, а только с 8 часов утра. Так как время окончания запуска не указано, запуск задания будет прекращен по окончании текущих суток. Таким образом, с 00:00 до 08:00 часов запуск задания выполняться не будет.

На этом создание расписания регламентного задания закончено, нажмем *OK*.

В качестве последнего штриха установим в свойствах регламентного задания флажок *Предопределенное* (рис. 20.8).

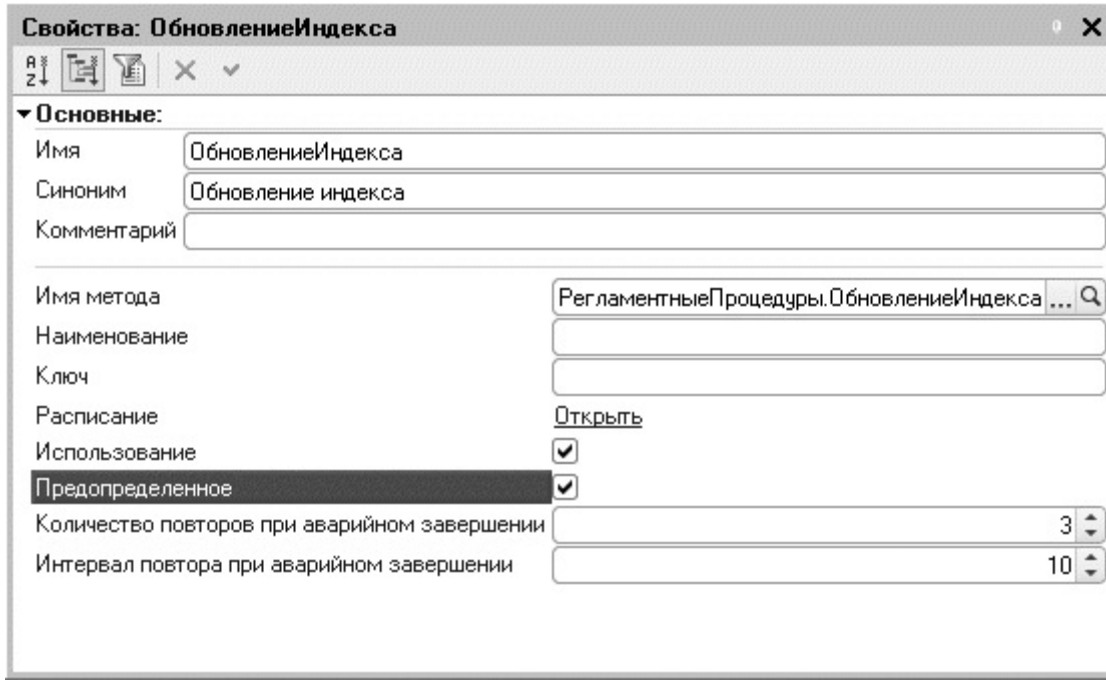


Рис. 20.8. Предопределенное регламентное задание

Установка этого свойства означает, что после запуска системы в режиме *1С:Предприятие* будет создано одно предопределенное регламентное задание. В противном случае такое задание пришлось бы создавать средствами встроенного языка.

На этом создание регламентного задания *Обновление индекса* завершено.

Теперь по аналогии создадим второе регламентное задание – *СлияниеИндексов*. В палитре его свойств нажмем кнопку выбора у поля ввода *Имя метода*. В открывшемся диалоге выберем модуль *РегламентныеПроцедуры*.

В этом модуле будет создан шаблон процедуры *СлияниеИндексов()*. Заполним его следующим образом (листинг 20.2).

Листинг 20.2. Процедура «СлияниеИндексов»

```
Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =
РежимПолнотекстовогоПоиска.Разрешить Тогда

  Если Не ПолнотекстовыйПоиск.ИндексАктуален() Тогда
    ПолнотекстовыйПоиск.ОбновитьИндекс(Истина);

  КонецЕсли;

КонецЕсли;
```

Эта процедура аналогична показанной в листинге 20.1, за исключением того, что при обновлении индекса выполняется слияние индексов (первый параметр *Истина*), и индексирование выполняется целиком, для всех данных (второй параметр *Ложь* по умолчанию), поскольку в данном случае время выполнения индексирования для нас не критично.

В свойствах задания установим также флажок *Предопределенное* и приступим к редактированию расписания. Для этого нажмем на ссылку *Открыть* в свойствах регламентного задания (в строке *Расписание*).

На закладке *Общее* укажем, что задание будет запускаться каждый день (*Повторять каждые: 1 дн.*), а на закладке *Дневное* укажем время начала выполнения задания (*Время начала: 01:00*), рис. 20.9.

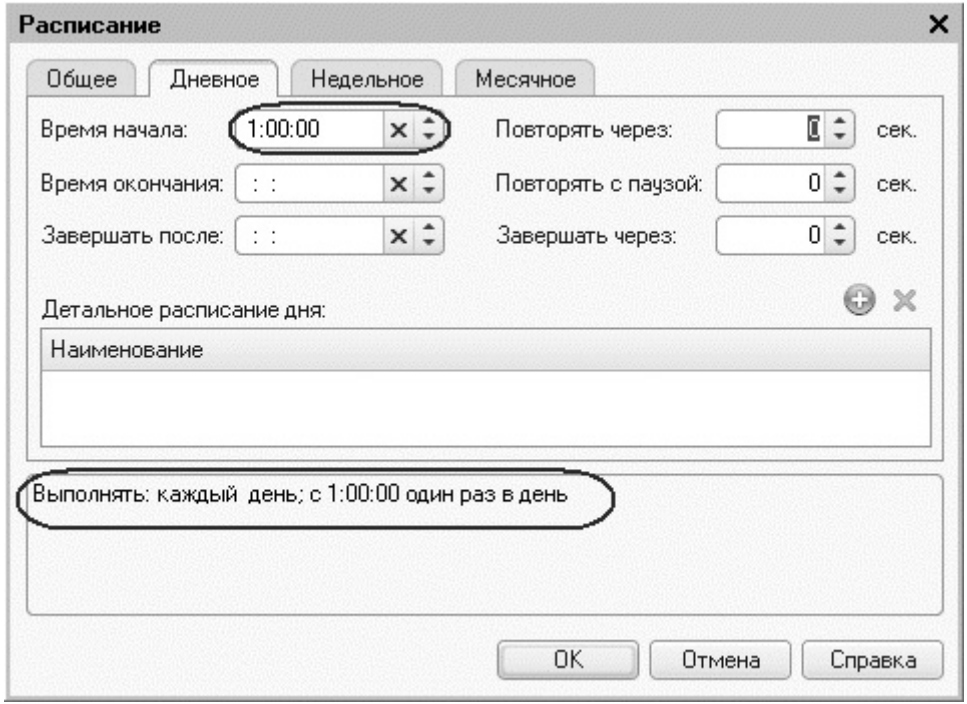


Рис. 20.9. Расписание задания «СлияниеИндексов»

В результате мы получим следующее расписание запуска регламентного задания: *Выполнять*: *каждый день*; *с 1:00 один раз в день*.

На этом создание регламентного задания *СлияниеИндексов* завершено, нажмем *ОК*.

Планировщик заданий

Как мы уже говорили на теоретическом занятии в начале книги, система «1С:Предприятие 8» поддерживает два варианта работы: файловый и клиент-серверный.

Файловый вариант прост в установке и практически не требует никакого администрирования. Именно в нем работает наша демонстрационная информационная база. Однако, как известно, за любую простоту использования нужно чем-то расплачиваться.

Если бы наша информационная база работала в клиент-серверном варианте, то для автоматического запуска и выполнения созданных нами заданий не требовалось бы дополнительных действий. Можно было бы обновить конфигурацию базы данных, и менеджер кластера серверов «1С:Предприятия»

начал бы самостоятельно выполнять задания в соответствии с указанным расписанием.

В нашем случае (файловый вариант работы) такого «промежуточного звена», которое могло бы взять на себя автоматическое выполнение заданий, не существует – есть только клиенты и информационная база.

Поэтому за простоту файлового варианта работы в данном случае придется «заплатить» тем, что для автоматического выполнения заданий необходимо всегда иметь одно работающее клиентское соединение с информационной базой, которое будет заниматься только запуском заданий по расписанию. Назовем такое соединение *планировщик заданий*.

В нашем примере мы создадим простую обработку, которая будет запускать задания по расписанию. Затем соединимся с нашей информационной базой в режиме *1С:Предприятие* и запустим эту обработку. Все, планировщик заданий готов.

Главное – не закрывать обработку и не закрывать это окно «1С:Предприятия», так как именно в нем и будут выполняться регламентные задания.

В режиме «Конфигуратор»

Итак, выделим ветвь *Обработки* в дереве объектов конфигурации и добавим новый объект конфигурации *Обработка* с именем *ПланировщикЗаданий*.

На закладке *Формы* создадим основную форму обработки.

В окне редактора форм на закладке *Реквизиты* добавим реквизит формы *Сообщение* и перетащим его в окно элементов формы.

В открывшейся палитре свойств поля *Сообщение* зададим вид поля – *Поле надписи* и заголовок – *Выполняется обработка заданий. Нажмите закрыть для прекращения процесса обработки* (рис. 20.10).

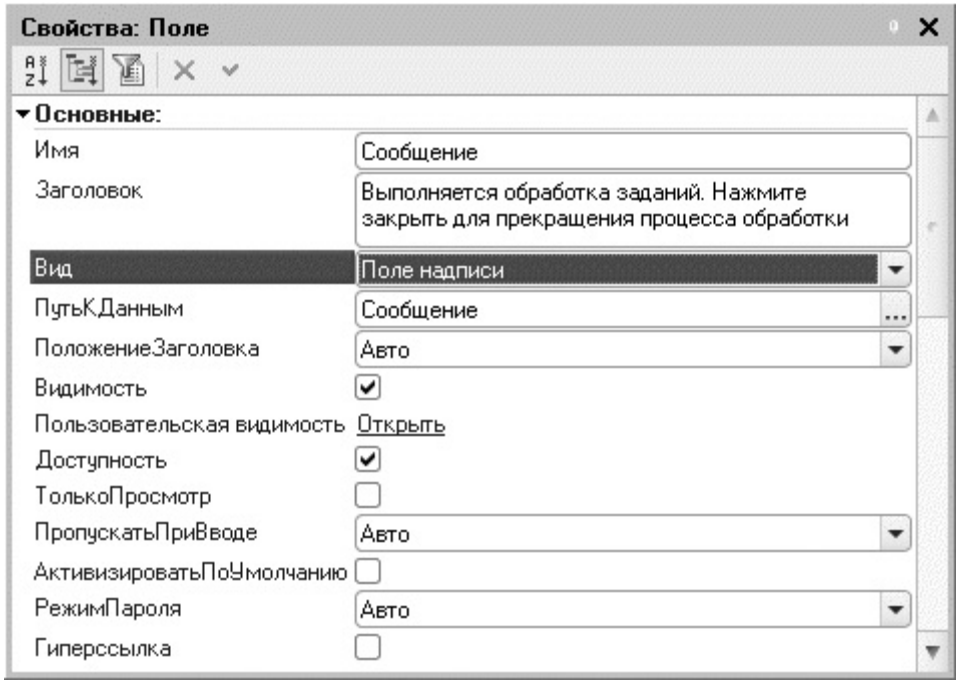


Рис. 20.10. Свойства поля «Сообщение»

В результате форма обработки будет выглядеть следующим образом (рис. 20.11).

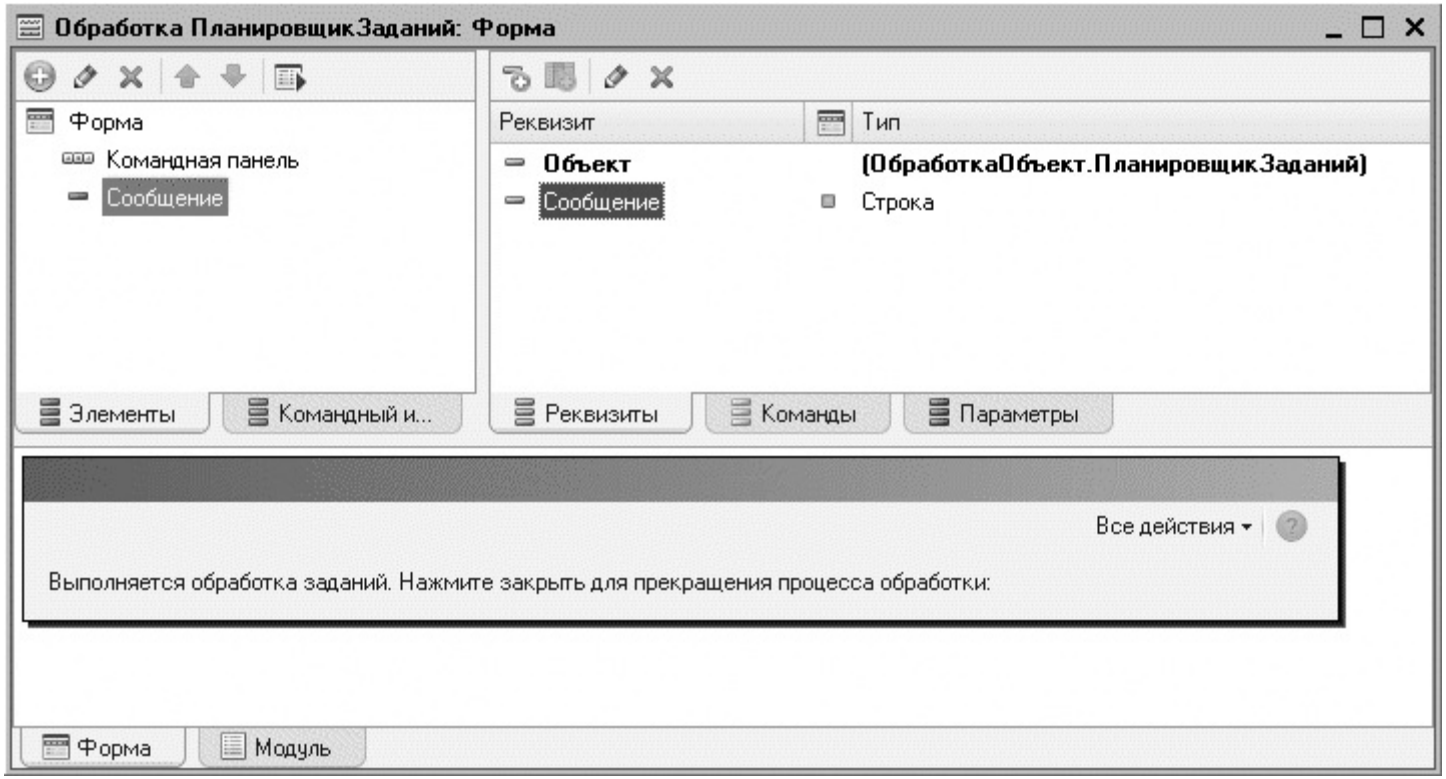


Рис. 20.11. Форма обработки

Откроем события формы и нажмем кнопку открытия у события *ПриОткрытии*. В обработчик этого события формы поместим следующий текст (листинг 20.3).

Листинг 20.3. Обработчик события формы «ПриОткрытии»

```
#Если ТолстыйКлиентУправляемоеПриложение Тогда
```

```
  ПодключитьОбработчикОжидания("ОбработкаЗаданий", 3);

#Иначе
  Предупреждение("Обработка может быть запущена только в толстом клиенте!");

  Закрыть();

#КонецЕсли
```

А также в модуле формы поместим сам обработчик ожидания – процедуру *ОбработкаЗаданий()*, листинг 20.4.

Листинг 20.4. Обработчик ожидания

```
&НаКлиенте
Процедура ОбработкаЗаданий()

  #Если ТолстыйКлиентУправляемоеПриложение Тогда
    ВыполнитьОбработкуЗаданий();

  #КонецЕсли

КонецПроцедуры;
```

Таким образом, при открытии формы выполняется подключение в качестве обработчика ожидания процедуры с именем *ОбработкаЗаданий()*.

Эта процедура будет вызываться каждые три секунды.

В свою очередь, процедура *ОбработкаЗаданий()* выполняет одно-единственное действие – вызывает метод *ВыполнитьОбработкуЗаданий()*. Этот метод проверяет, существуют ли задания, которые в соответствии с их расписанием должны быть выполнены. Если такие задания существуют, он запускает их на выполнение.

В обеих процедурах используются инструкции препроцессора (после символа #) для того, чтобы указать, что фрагмент кода будет присутствовать только в том случае, если запущен толстый клиент в управляемом режиме, так как именно в этом режиме будет запускаться наша обработка *ПланировщикЗаданий*.

Для того чтобы проверить, что запуск заданий действительно происходит, добавим в начало процедуры *ОбновлениеИндекса* (общий модуль *РегламентныеПроцедуры*) следующую строку (листинг 20.5).

Листинг 20.5. Сообщение о запуске регламентного задания

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "Запуск регламентного задания Обновление индекса " +
ТекущаяДата();
Сообщение.Сообщить();
```

В заключение в окне редактирования объекта конфигурации Обработка *ПланировщикЗаданий* на закладке *Подсистемы* мы отметим подсистему *Предприятие*.

То есть обработка будет доступна для пользователя с административными правами и вызываться из панели действий в группе *Сервис* раздела *Предприятие*.

В режиме «1С:Предприятие»

Теперь проверим, как работают наши регламентные задания.

Обновим конфигурацию базы данных, нажав кнопку в панели инструментов *Обновить конфигурацию базы данных (F7)*  , и после этого запустим систему в режиме *«1С:Предприятие»(толстый клиент)*.

Для этого нужно выполнить последовательность действий на вашем компьютере *Старт > Все программы > 1С Предприятие 8.2 > Дополнительно > <номер версии> > 1С Предприятие (толстый клиент)*. В окне запуска откроем в режиме *1С:Предприятие* нашу информационную базу.

В разделе *Предприятие* откроем обработку *Планировщик заданий* и подождем несколько минут.

В результате в окно сообщений будет выведен, например, такой текст (рис. 20.12).

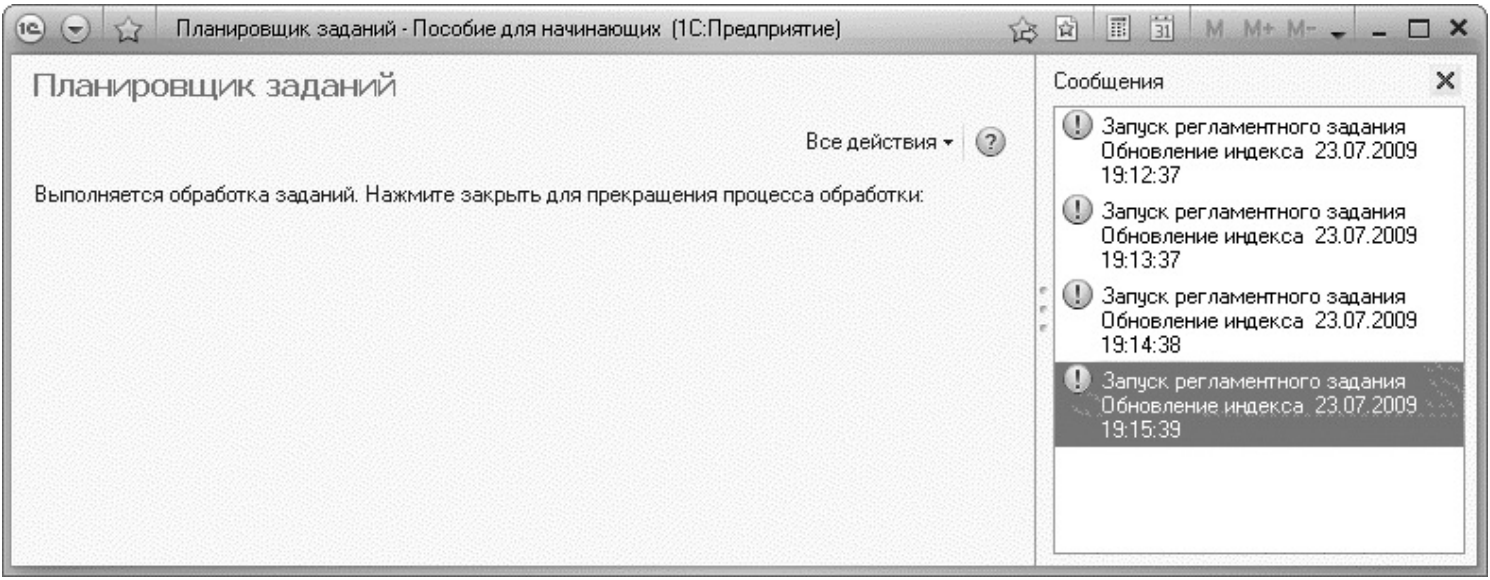


Рис. 20.12. Сообщения о запуске регламентных заданий

Таким образом, мы видим, что задание обновления индекса запускается каждые 60 секунд, как мы и указали в расписании.

Узнай больше!

В соединении, которое занято запуском регламентных заданий, не

рекомендуется выполнять какие-либо другие задачи. Таким образом, для «обычной» работы с нашей информационной базой необходимо еще раз запустить систему в режиме «1С:Предприятие», выбрать нашу информационную базу и создать тем самым второе соединение, в котором уже и выполнять «обычную» работу пользователя информационной базы.

Контрольные вопросы

- *Для чего предназначены регламентные задания.*
- *Как задать расписание для автоматического запуска заданий.*
- *Как обеспечить запуск заданий по расписанию в файловом варианте работы.*

Занятие 21 (0:40). Редактирование движений в форме документа

Продолжительность

Ориентировочная продолжительность занятия – 40 минут.

В нашей информационной базе, как, впрочем, и в любой другой, обязательно следует предусмотреть возможность ввода начальных остатков в регистры. Это необходимо для того, чтобы пользователи могли начать работу с нашей информационной базой не с чистого листа, а с некоторого «исходного состояния», которое было в их прежней системе учета (пусть даже они вели учет на бумаге).

Задача ввода начальных остатков отличается от прочих алгоритмов изменения состояния регистров нашей информационной базы тем, что подразумевает изменение данных непосредственно в регистрах, без использования каких-либо промежуточных алгоритмов (заполнения документов данными, проведения документов, контроля правильности данных, указанных в документах, и пр.).

Рассмотрим пример ввода начальных остатков регистра накопления *ОстаткиМатериалов*.

Для выполнения этой задачи мы создадим документ, в котором будем вручную редактировать его движения по регистру *ОстаткиМатериалов* прямо в форме документа.

В режиме «Конфигуратор»

Создадим новый объект конфигурации *Документ* с именем *ВводНачальныхОстатковНоменклатуры*.

На закладке *Движения* запретим проведение документа (поскольку сами будем формировать записи регистра) и отметим, что движения документа будут находиться в регистре накопления *ОстаткиМатериалов* (рис. 21.1).

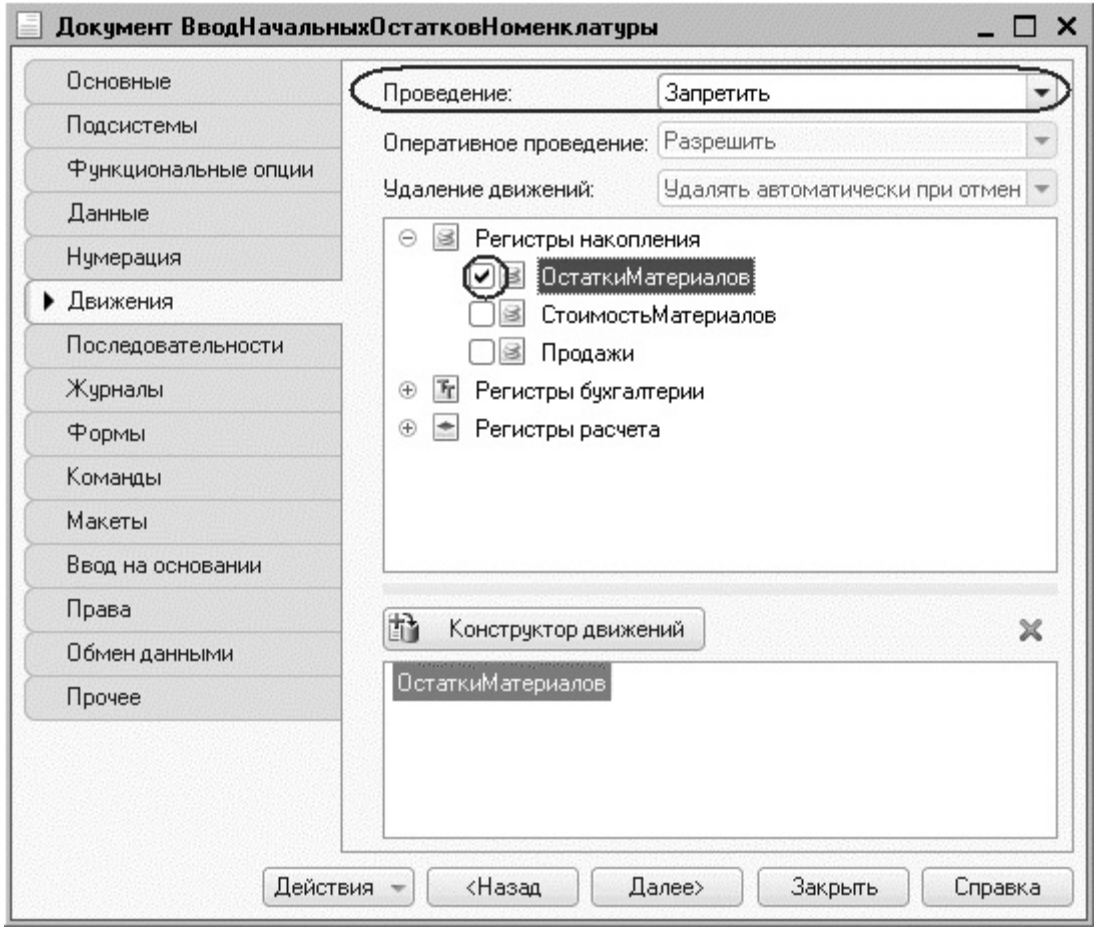


Рис. 21.1. Запрет проведения документа

После этого перейдем на закладку *Формы* и создадим основную форму документа.

В окне редактора форм на закладке *Реквизиты* раскроем основной реквизит формы *Объект*, затем раскроем коллекцию движений *Движения*, найдем строку *ОстаткиМатериалов* и перетащим ее в окно элементов формы. На вопрос системы: «Добавить колонки таблицы?» – ответим утвердительно (рис. 21.2).

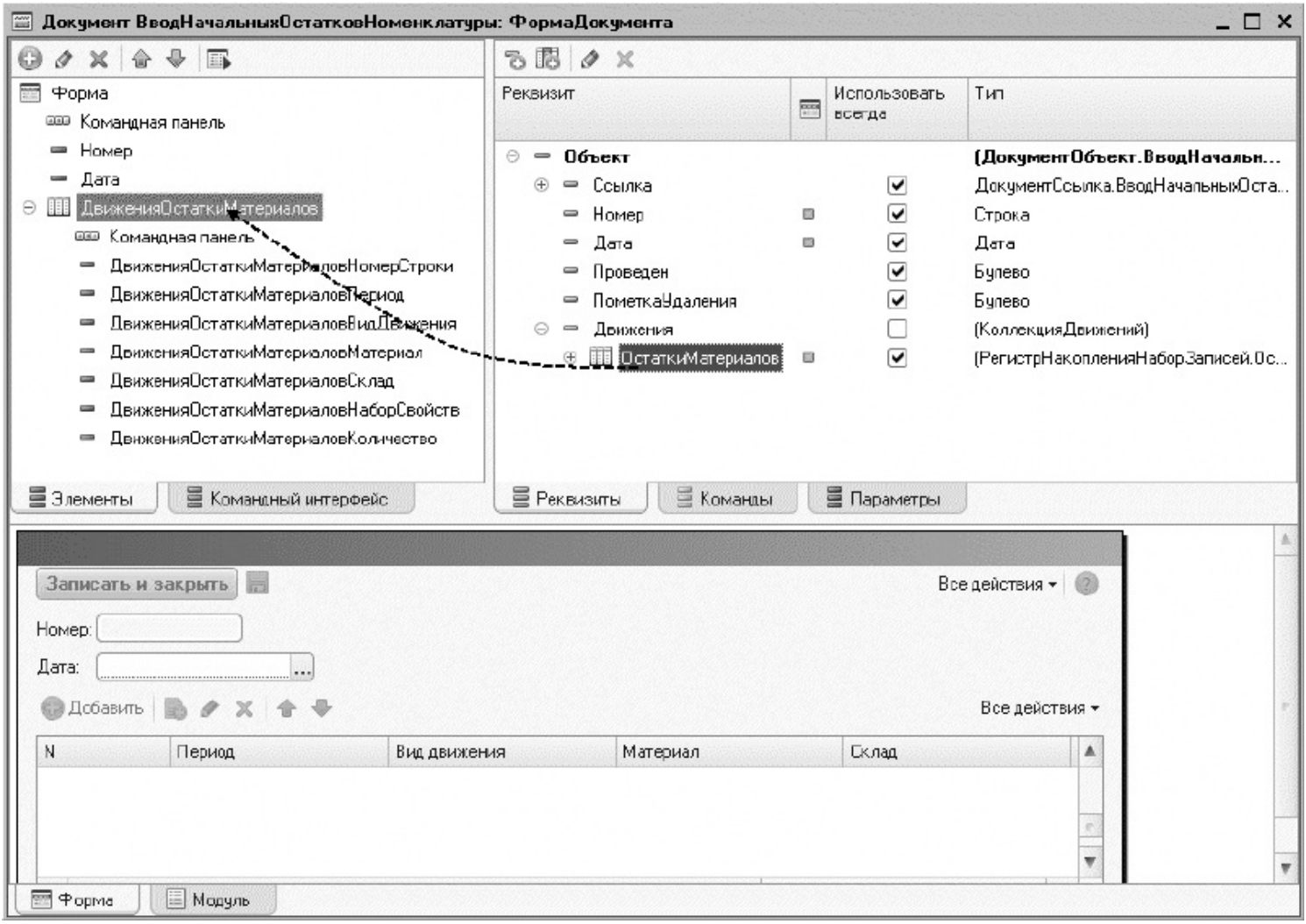


Рис. 21.2. Редактирование формы документа «Ввод начальных остатков номенклатуры»

Обратите внимание, что в палитре свойств этой таблицы в строке *Данные* автоматически будет установлена связь с данными набора записей регистра *Объект.Движения.ОстаткиМатериалов*.

Немного изменим внешний вид формы. В окне элементов формы добавим группу полей с типом группировки *Горизонтальная* и перетащим в нее поля документа *Номер* и *Дата*. А также поменяем местами поля таблицы *ДвиженияОстаткиМатериалов НаборСвойств* и *Склад* (рис. 21.3).

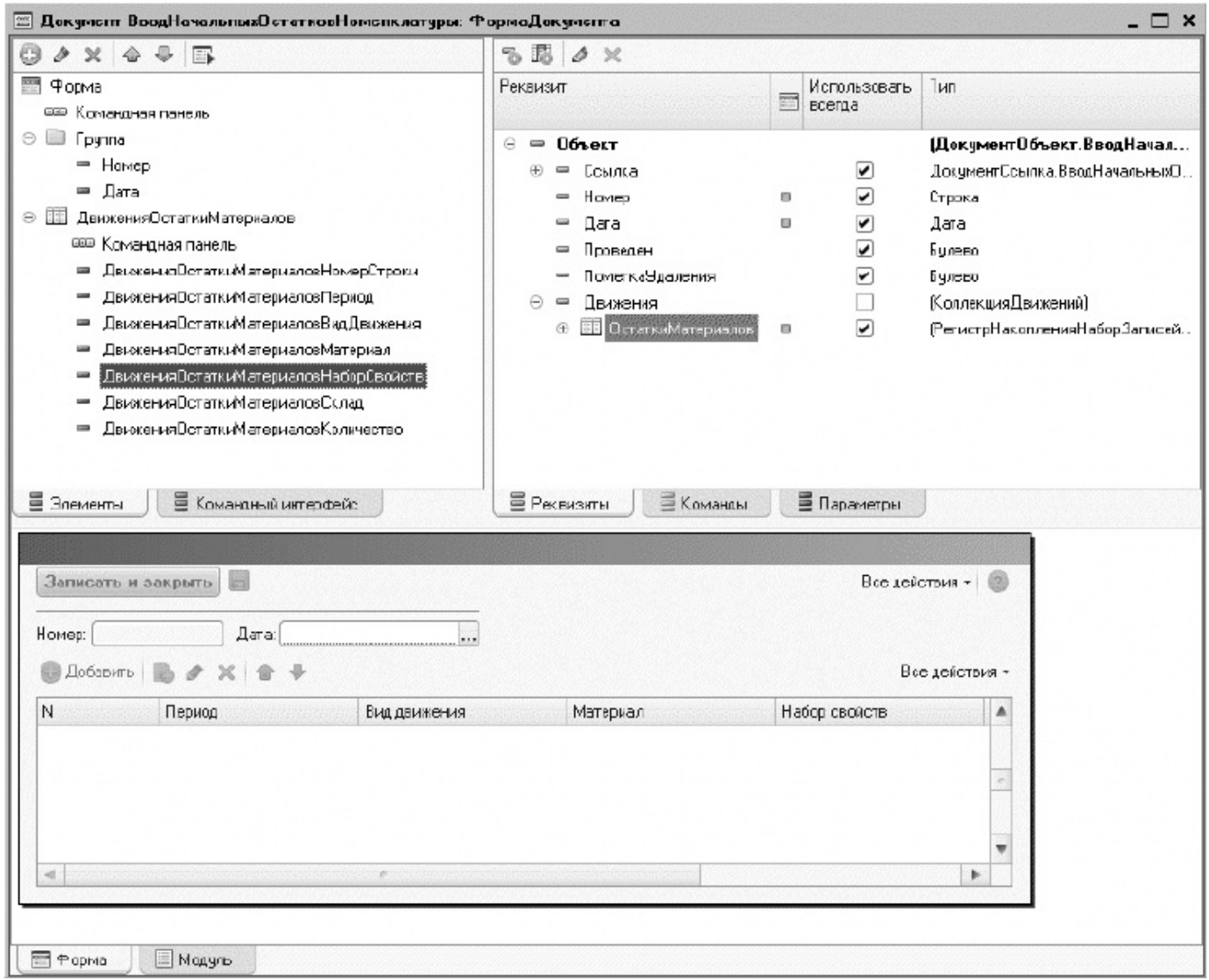

Документ.ВводНачальныхОстатковНоменклатуры. ФормаДокумента
Форма
Командная панель
Группа
Номер
Дата
ДвиженияОстаткиМатериалов
Командная панель
ДвиженияОстаткиМатериаловНомерСтроки
ДвиженияОстаткиМатериаловПериод
ДвиженияОстаткиМатериаловВидДвижения
ДвиженияОстаткиМатериаловМатериал
ДвиженияОстаткиМатериаловНаборСвойств
ДвиженияОстаткиМатериаловСклад
ДвиженияОстаткиМатериаловКоличество
Элементы
Командный интерфейс
Реквизит
Использовать всегда
Тип
Объект
ДокументОбъект.ВводНачал...
Ссылка
ДокументСсылка.ВводНачальныхО...
Номер
Строка
Дата
Дата
Проведен
Булево
ПометкаУдаления
Булево
Движения
(КоллекцияДвижений)
ОстаткиМатериалов
(РегистрНакопленияНаборЗаписей...
Реквизиты
Команды
Параметры
Записать и закрыть
Все действия
Номер:
Дата:
Добавить
Все действия
N
Период
Вид движения
Материал
Набор свойств
Форма
Модуль

Рис. 21.3. Измененная форма документа «Ввод начальных остатков номенклатуры»

В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра *ОстаткиМатериалов*, связанному с документом.

Для этого в левом верхнем окне редактора форм перейдем на закладку *Командный интерфейс*.

В группе *Панель навигации* в подгруппе *Перейти* установим видимость для команды открытия регистра *Остатки материалов* (рис. 21.4).

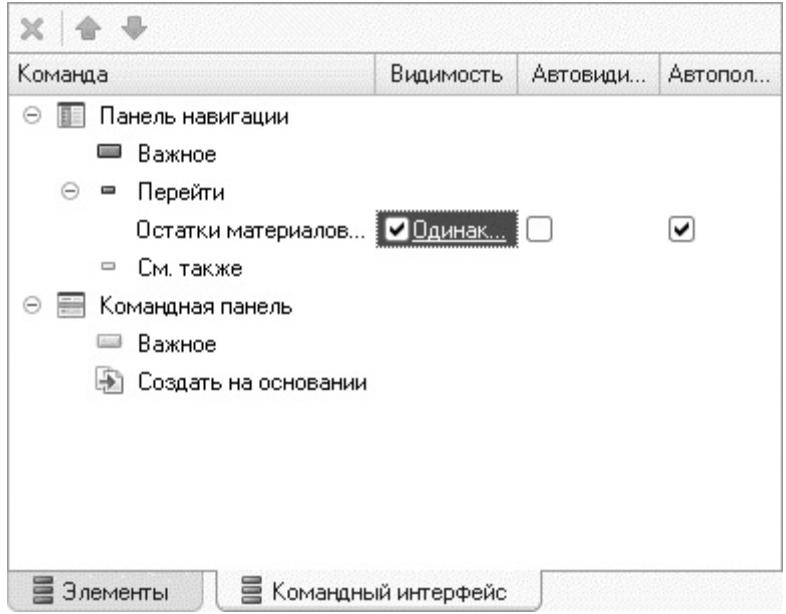


Рис. 21.4. Редактирование командного интерфейса формы

В окне редактирования объекта конфигурации Документ *ВводНачальныхОстатковНоменклатуры* на закладке *Подсистемы* мы укажем его принадлежность к подсистеме *Бухгалтерия*.

В заключение отредактируем командный интерфейс этой подсистемы (*Общие* > *Подсистемы* > *Все подсистемы*).

Выделим в списке подсистем подсистему *Бухгалтерия* и в списке команд этой

подсистемы установим видимость команды *Ввод начальных остатков номенклатуры: создать* в группе *Панель действий.Создать*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим работу нашего документа.

Выполним команду *Ввод начальных остатков номенклатуры* в панели действий раздела *Бухгалтерия*.

Создадим документ для ввода начальных остатков в регистр *ОстаткиМатериалов* и внесем в него следующие данные (рис. 21.5).

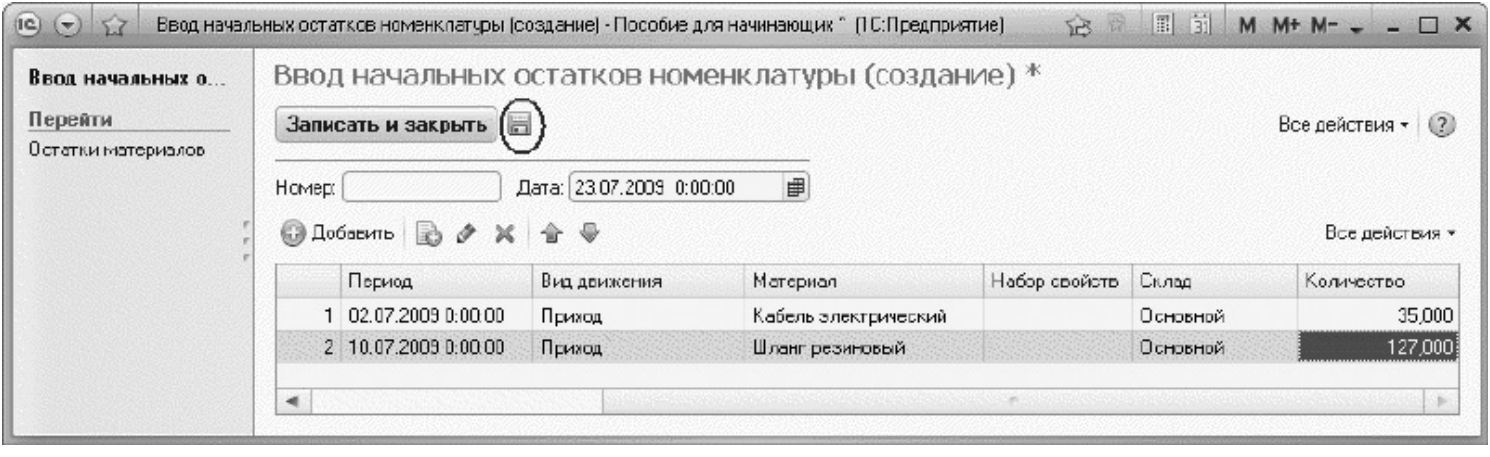


Рис. 21.5. Документ «Ввод начальных остатков номенклатуры № 1»

Обратите внимание на то, что дата документа не совпадает с датами отдельных записей, которые мы создаем в движениях документа.

Нажмем *Записать* (кнопку со значком дискеты) и в панели навигации формы документа перейдем к движениям нашего документа в регистре *ОстаткиМатериалов* (рис. 21.6).

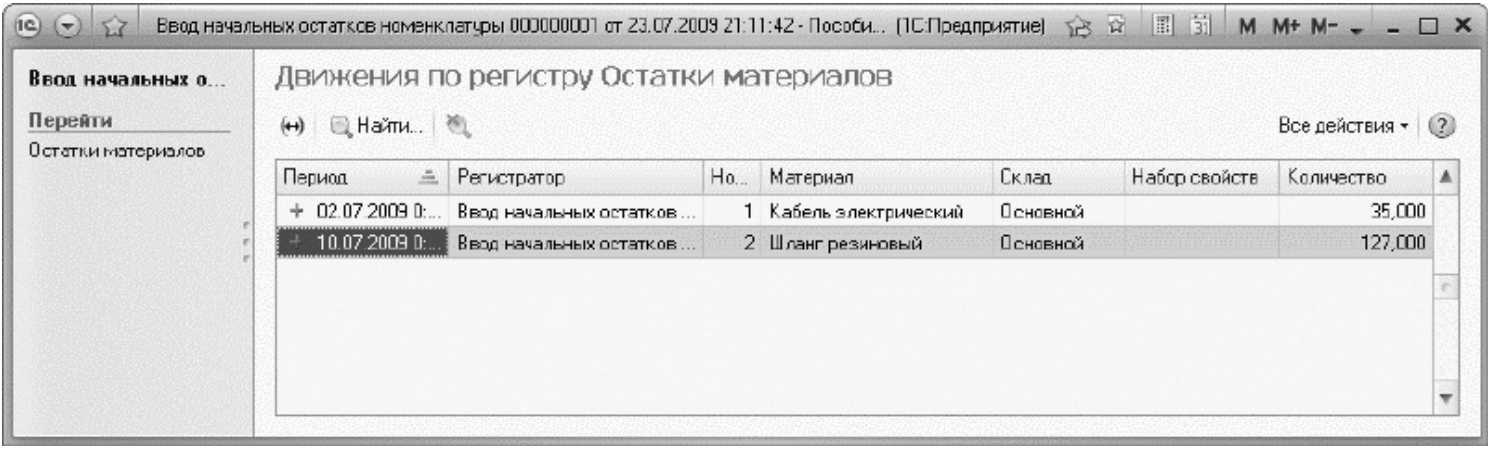


Рис. 21.6. Записи регистра «ОстаткиМатериалов»

Таким образом, мы добились поставленной цели: с одной стороны, задавая дату документа, мы можем фиксировать момент внесения изменений в записи регистра, с другой стороны – для каждой создаваемой нами записи регистра мы

можем указать индивидуальное значение поля *Период*.

Теперь займемся ужесточением требований, предъявляемых к тому, как наш документ формирует записи регистра, и рассмотрим два типичных варианта.

Программное редактирование записей регистра

В режиме «Конфигуратор»

Первое требование, которое мы реализуем, будет заключаться в том, что записи регистра должны формироваться той же датой, что и дата документа. Иначе говоря, синхронизируем дату движений с датой документа.

Для этого создадим для формы документа обработчик события *ПередЗаписью* и добавим в него следующий текст (листинг 21.1).

Листинг 21.1. Обработчик события «ПередЗаписью» формы документа

```
Для Каждого ЗаписьРегистра Из Объект.Движения.ОстаткиМатериалов Цикл
  ЗаписьРегистра.Период = Объект.Дата;

КонецЦикла;
```

В режиме «1С:Предприятие»

Снова запустим «1С:Предприятие» в режиме отладки, откроем наш документ и нажмем *Записать*.

Открыв движения документа в регистре *ОстаткиМатериалов,* увидим, что значение поля *Период* у всех записей стало равно дате документа (рис. 21.7).

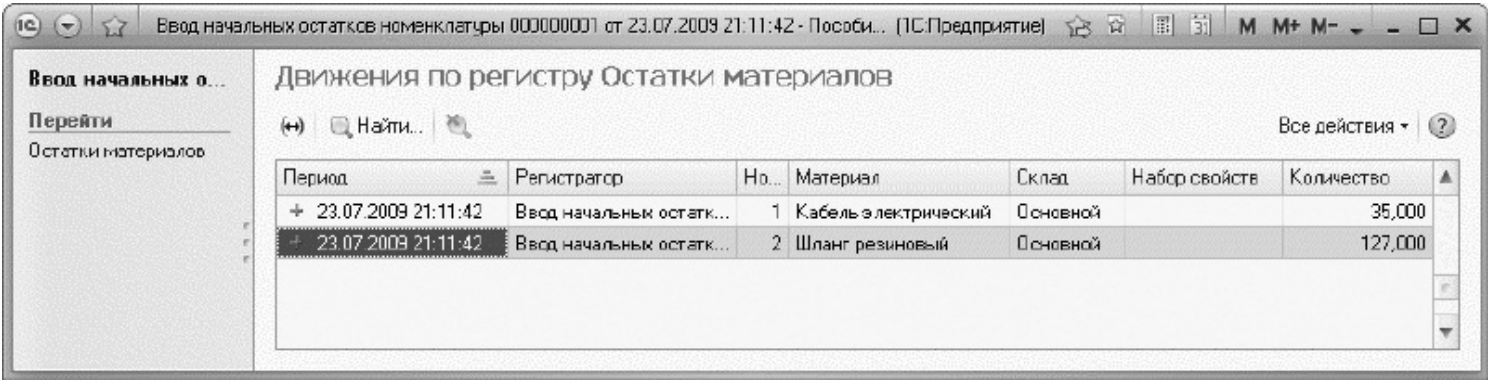


Рис. 21.7. Измененные записи регистра «ОстаткиМатериалов»

Можно сказать, что мы достигли поставленной цели, но лишь в ситуации, когда запись документа выполняется интерактивными средствами.

Если программно вызвать метод *Записать()* у объекта нашего документа, он будет записан без участия формы документа. Это значит, что событие *ПередЗаписью* формы документа вызвано не будет и наш код обработчика не отработает.

Чтобы предусмотреть возможность синхронизации периода движений документа с датой документа и в случае программной записи объекта *Документ*, следует использовать обработчик события *ПередЗаписью* объекта *Документ*, а не формы документа.

Событие *ПередЗаписью* в случае интерактивной записи документа сначала будет вызвано у формы документа, а затем у объекта *Документ* (смотри схему событий в разделе «Последовательность событий при записи документа из формы документа»).

В режиме «Конфигуратор»

Поэтому вернемся в конфигуратор, удалим из модуля формы добавленный нами текст и создадим обработчик события *ПередЗаписью* в модуле объекта *Документ ВводНачальныхОстатковНоменклатуры*.

Для этого на закладке *Прочее* окна редактирования этого объекта конфигурации откроем модуль объекта и внесем в него следующий текст (листинг 21.2).

Листинг 21.2. Обработчик события «ПередЗаписью» модуля объекта

```
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

  // Определить, нужно ли обновлять дату в движениях.
```

```
ОбновитьДатуДвижений = ЭтоНовый() Или
Движения.ОстаткиМатериалов.Модифицированность();
Если Не ОбновитьДатуДвижений Тогда

  // Проверить, что дата изменилась.
  Запрос = Новый Запрос;
  Запрос.УстановитьПараметр("ТекущийДокумент", Ссылка);
  Запрос.Текст =
    "ВЫБРАТЬ
    |        Дата
    |ИЗ
    |        Документ.ВводНачальныхОстатковНоменклатуры
    |ГДЕ
    |        Ссылка = &ТекущийДокумент";

  Выборка = Запрос.Выполнить().Выбрать();
  Выборка.Следующий();
  ОбновитьДатуДвижений = Выборка.Дата > Дата;

КонецЕсли;

// Установить всем новую дату, если нужно.
Если ОбновитьДатуДвижений Тогда

  Если Не Движения.ОстаткиМатериалов.Выбран() И Не
  Движения.ОстаткиМатериалов.Модифицированность() Тогда
    Движения.ОстаткиМатериалов.Прочитать();

  КонецЕсли;

  Для Каждого ЗаписьРегистра Из Движения.ОстаткиМатериалов Цикл
```

```
        ЗаписьРегистра.Период = Дата;

    КонецЦикла;

  КонецЕсли;

КонецПроцедуры
```

Как вы видите, в этом случае обработчик содержит больше кода за счет дополнительных проверок, которые выполняются в результате возможности как интерактивной, так и программной записи объекта.

Поясним содержание обработчика. Если записывается новый документ или были изменены его движения, следует обновить дату движений. В противном случае мы считываем запросом дату документа из базы данных и сравниваем ее с датой, установленной у записываемого объекта. Если даты разные, также следует обновить дату движений.

Перед установкой даты мы проверяем, был ли прочитан набор записей в свойстве *Движения* объекта и изменялся ли он. Если оба этих условия ложны, значит набор записей в свойстве *Движения* объекта пуст, и это состояние не связано с его изменением. В этом случае, чтобы предотвратить ошибочное удаление записей в регистре (перезапись пустым набором записей), мы предварительно читаем движения из регистра в набор записей в свойстве

Движения.

Затем, как и в предыдущем случае, устанавливаем нужную дату для всех записей этого набора. При выполнении записи объекта *Документ* этот набор будет записан в регистр накопления.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и убедимся, что, указав новую дату (например, *01.05.2009*) для нашего документа и записав его, мы получим движения в регистре накопления с новой датой.

В процессе записи нашего документа можно управлять не только периодом записей регистра накопления, но и значениями других полей регистра.

Например, по аналогичному принципу может быть создан документ *Операция*, позволяющий вводить ручные операции в регистр бухгалтерии.

При этом вероятно, что кроме управления периодом записей регистра вам потребуется управлять значением поля *Активность* («включать» и «выключать» проводки документа) и т. д.

Где создавать обработчики событий

В заключение следует сказать, что выбор обработчика, в который будет помещен текст процедуры, зависит от логики работы создаваемого объекта. Если конфигурация не предусматривает программной записи объекта, можно выбрать обработчик модуля формы. Если предполагается и программная модификация объекта, следует выбирать обработчик модуля объекта.

Заметьте, что оба эти способа не исключают модификацию записей регистра через объект *Регистр<…>НаборЗаписей.<имя регистра>*. Поэтому если логика конфигурации подразумевает возможность программной модификации объекта *НаборЗаписей*, код обработки следует помещать в обработчик события набора записей. Все попытки изменить данные регистра будут сведены в конечном счете к записи именно набора записей.

Контрольные вопросы

- *Для чего предназначен документ для ввода начальных остатков и как его создать.*
- *Как программно изменить значение регистра при вводе начальных остатков.*
- *В каких случаях использовать модуль формы, а в каких – модуль объекта для размещения обработчиков событий.*

Занятие 22 (1:00). Список пользователей и их роли

Продолжительность

Ориентировочная продолжительность занятия – 1 час.

После того как созданы все основные объекты конфигурации, можно приступить к определению ролей пользователей. Администрирование списка пользователей «1С:Предприятия» и назначение им ролей в соответствии с их служебными обязанностями – очень важные моменты для организации интерфейса прикладного решения в целом и разграничения прав и действий его отдельных пользователей. Этому вопросу и будет посвящено данное занятие.

До сих пор мы с вами имели полный доступ ко всем разделам приложения и ко всем объектам конфигурации и командам, используемым в этих разделах. Однако при реальной работе пользователей одной из главных возможностей, которую должно обеспечивать прикладное решение, является разграничение прав доступа пользователей к той или иной информации, хранящейся в информационной базе.

Например, руководитель должен, очевидно, иметь доступ ко всей информации, которая содержится в базе данных, а вот кладовщик, напротив, должен иметь

доступ только к информации, касающейся движения товаров на складах, и не иметь возможности просматривать бухгалтерскую или кадровую информацию.

Кроме этого, должна существовать возможность ограничить пользователей в выполнении тех или иных действий с объектами базы данных. Например, кладовщик может создавать и изменять приходные накладные, поскольку он отвечает за учет материалов на предприятии. Мастеру может понадобиться просматривать приходные накладные для того, чтобы знать, какие материалы и когда были получены. Однако мастер не должен иметь возможности вносить какие-либо изменения в приходные накладные.

Что такое роль

Для описания подобных разрешений используются объекты конфигурации *Роль*. С помощью такого объекта разработчик получает возможность описать набор прав на выполнение тех или иных действий над каждым объектом базы данных и над всей конфигурацией в целом.

Как правило, роли создаются отдельно для каждого вида деятельности, и каждому пользователю системы ставится в соответствие одна или несколько ролей.

Если пользователю поставлено в соответствие несколько ролей,

предоставление доступа будет осуществляться по следующему алгоритму:

- если хотя бы в одной роли есть разрешение, то доступ будет открыт;
- если во всех ролях разрешение отсутствует, то доступ будет закрыт.

Создание ролей

В режиме «Конфигуратор»

При создании ролей исходят, как правило, из того, какие полномочия требуются различным группам пользователей на доступ к информации. Для этого мы воспользуемся подсистемами, которые значительно облегчат нашу задачу.

Администратор

Первая роль, которую мы создадим, будет *Администратор*. Она должна включать в себя полные права на работу с данными информационной базы.

Раскроем ветвь *Общие* дерева объектов конфигурации. Выделим строку *Роли* и добавим новый объект конфигурации *Роль*. Зададим его имя – *Администратор* (рис. 22.1).

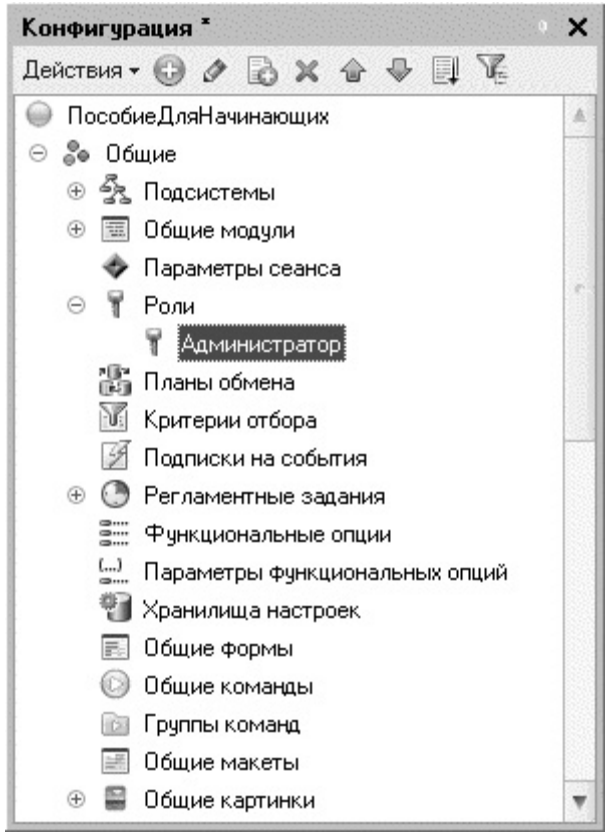


Рис. 22.1. Создание роли

Откроется окно редактирования прав этой роли (рис. 22.2).

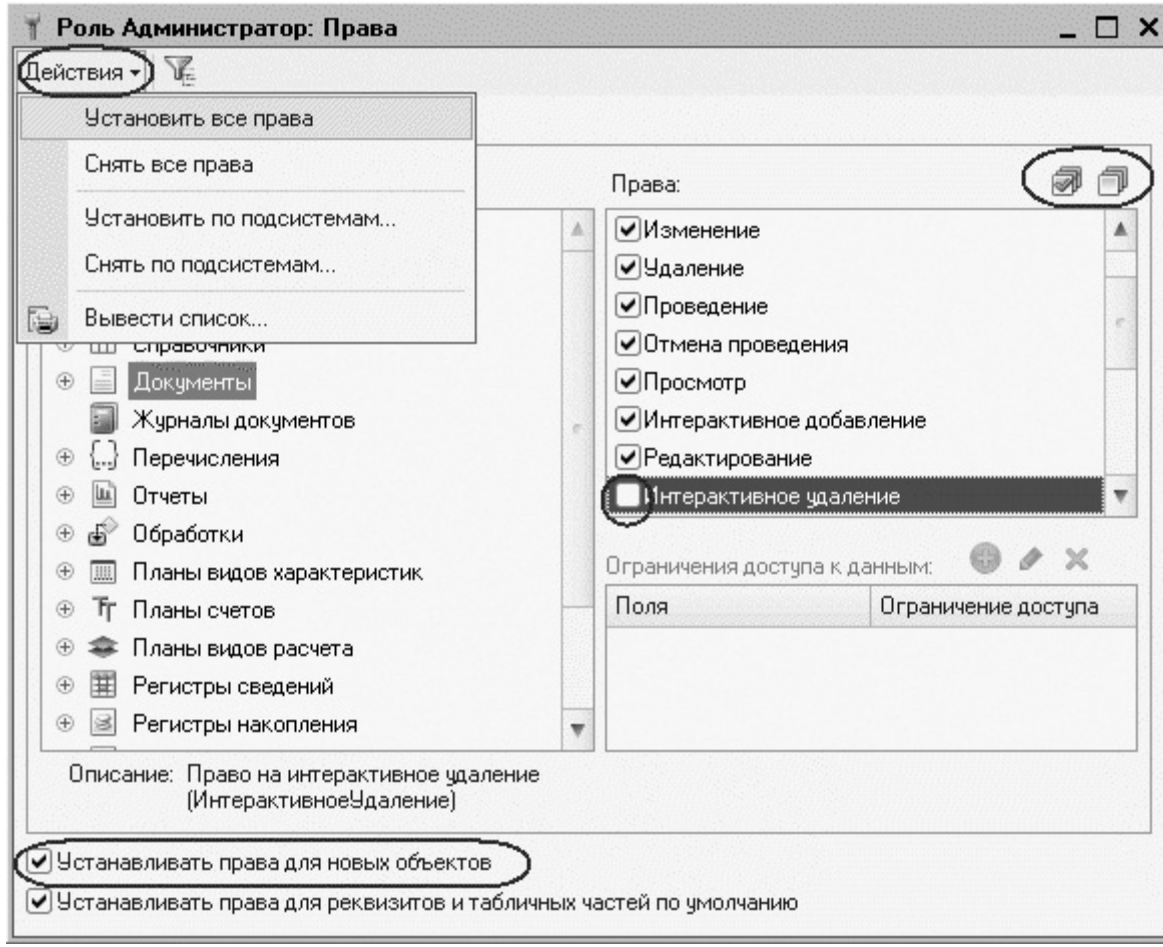


Рис. 22.2. Окно редактирования прав для роли «Администратор»

Слева, в списке объектов, перечислены все объекты и виды объектов

конфигурации, а справа, в окне прав, – доступные права для выбранного объекта или видов объектов конфигурации.

Администратор должен иметь права на все объекты и все виды объектов. Для этого выполним команду *Действия* > *Установить все права* в командной панели окна.

После этого все права для всех объектов будут помечены.

Однако можно поставить или снять отметку для прав конкретного объекта конфигурации, пользуясь кнопками *Отметить все элементы* и *Снять отметку со всех элементов*, расположенными над окном прав.

Теперь единственное, что следует сделать, – снять разрешение на интерактивное удаление для всех объектов. Это необходимо для того, чтобы администратор случайно не мог удалить какой-либо объект базы данных. Для этого пройдемся по всем видам объектов конфигурации (*Справочники*, *Документы* и т. д.) и снимем отметку с команды *Интерактивное удаление* (см. рис. 22.2).

Для того чтобы наш *Администратор* мог работать с объектами, которые мы будем создавать после расстановки прав, зададим для него параметр *Устанавливать права для новых объектов* (см. рис. 22.2).

На этом создание роли *Администратор* закончено.

Директор

Следующей ролью, которую мы создадим, будет роль *Директор*.

Создадим новый объект конфигурации *Роль* с именем *Директор*.

Нас устраивает, что у новой роли нет прав на доступ ко всем объектам, за исключением тех видов объектов конфигурации, для которых не создано ни одного объекта. Для таких видов объектов конфигурации останутся установленными полные права.

Установим право *Вывод* для всей конфигурации (рис. 22.3).

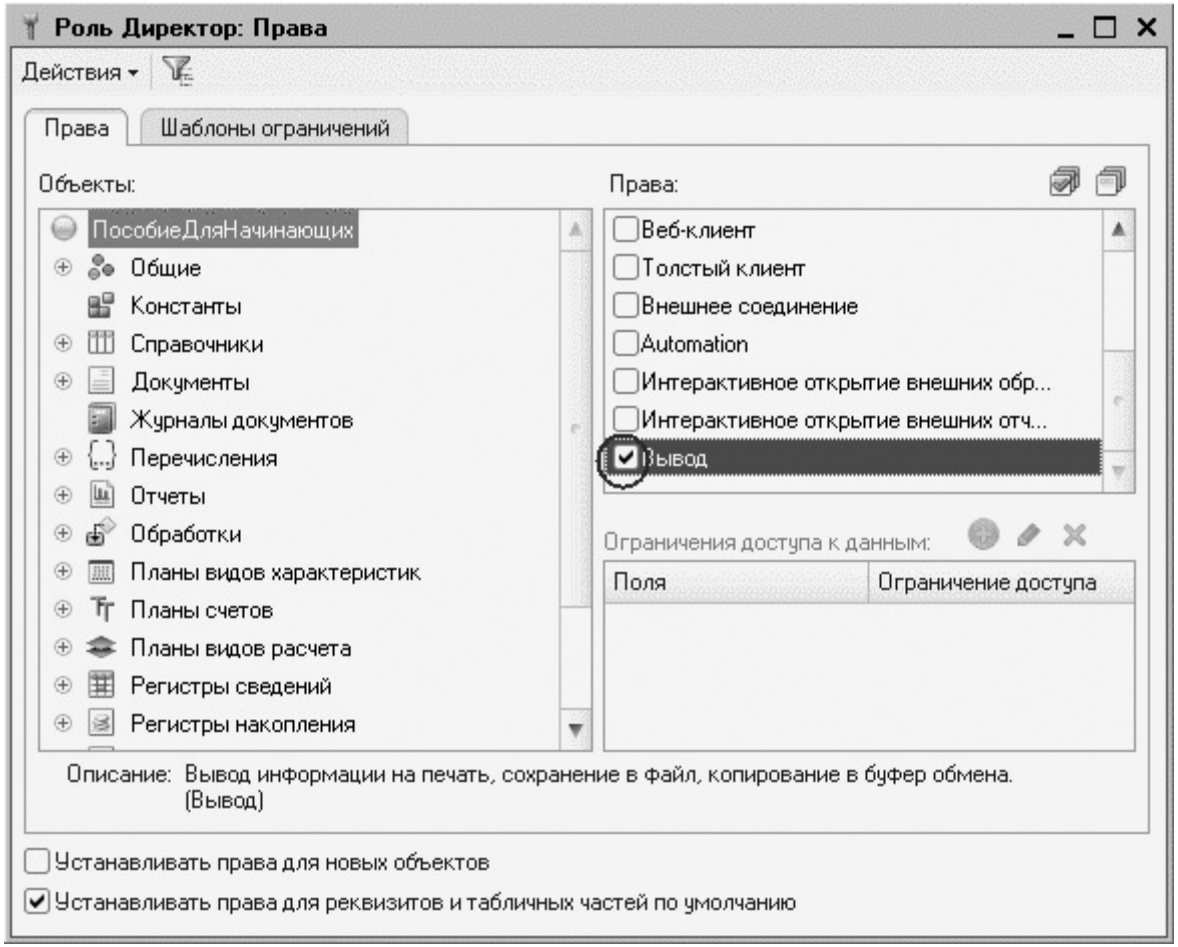


Рис. 22.3. Окно редактирования прав для роли «Директор»

Теперь нам останется лишь пройти по видам объектов конфигурации и

установить для них право *Просмотр* (права *Чтение* и *Использование* при этом установятся автоматически).

Затем раскроем ветвь *Общие*, выделим ветвь *Подсистемы* и отметим право *Просмотр* у всех подсистем, кроме подсистемы *Предприятие*. Тем самым мы предоставим директору возможность просматривать все данные информационной базы, но исключим из его интерфейса все действия, которые по логике нашей конфигурации не относятся к прикладной ее части.

Вторая роль нашей конфигурации готова.

Мастер

Следующая роль, которую мы создадим, будет роль *Мастер*. Снова добавим новый объект конфигурации Роль с именем *Мастер*. Выполним команду *Действия* > *Установить по подсистемам* и выберем подсистемы *УчетМатериалов* и *ОказаниеУслуг*.

В результате будут установлены все права на объекты конфигурации, относящиеся к данным подсистемам.

Если теперь установить фильтр объектов по подсистемам *УчетМатериалов* и *ОказаниеУслуг*, то можно при необходимости внести уточнения в установленные права (рис. 22.4).

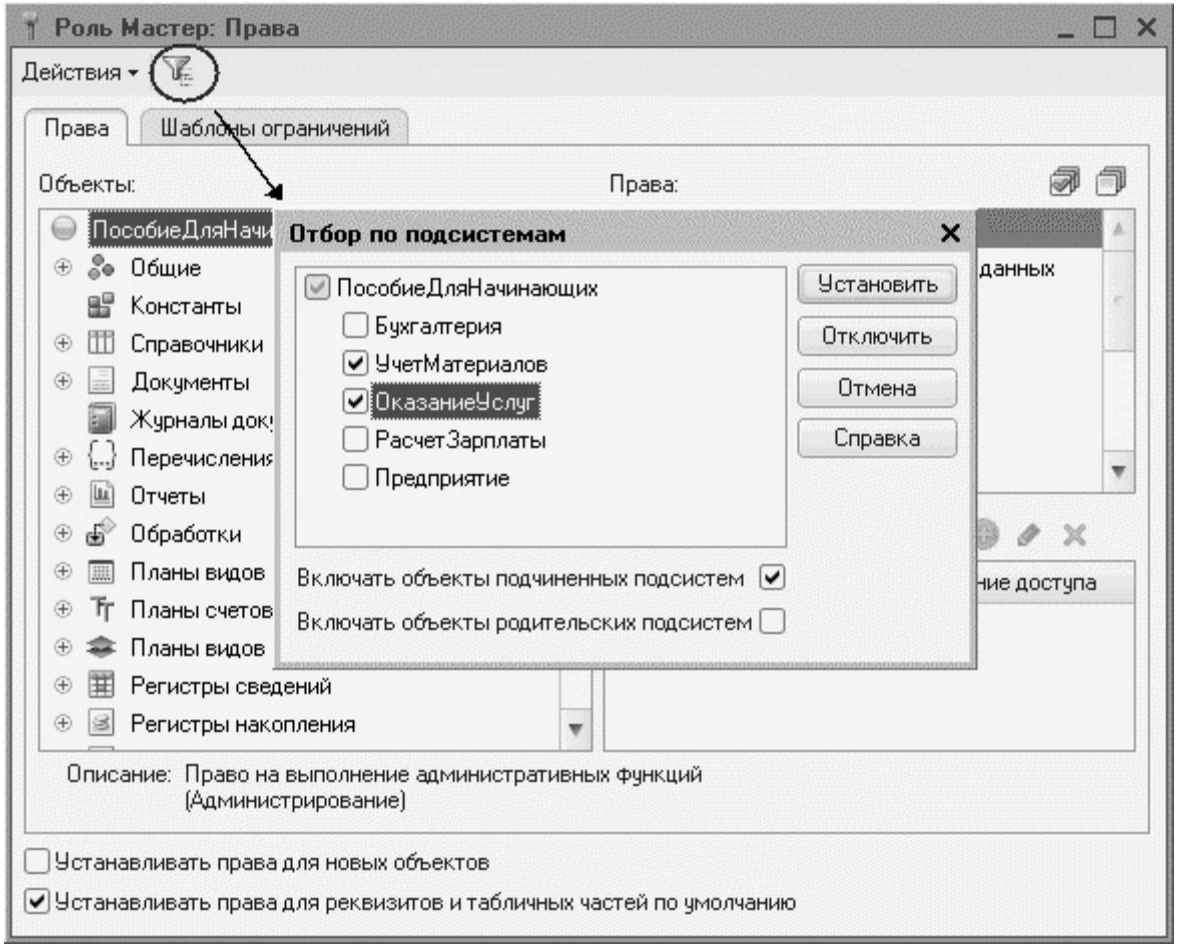


Рис. 22.4. Установка фильтра по подсистеме

В частности, для справочника *Сотрудники* мы запретим права *Добавление*,

Изменение и *Удаление*.

Обратите внимание, что при запрете права *Добавление* исчезла отметка и у права *Интерактивное добавление*, так как оно является «уточнением» права *Добавление*. Точно так же уточненные права запрещаются и при отмене прав на изменение и удаление. Кроме этого, мы снова снимем разрешения на интерактивное удаление для всех объектов базы данных.

Теперь пройдем по всем видам объектов конфигурации и снимем у всех право *Интерактивное удаление*.

Затем снимем фильтр и установим все права, кроме интерактивного удаления для следующих объектов конфигурации:

- справочник *ВариантыНоменклатуры,*
- справочник *ДополнительныеСвойстваНоменклатуры,*
- план видов характеристик *СвойстваНоменклатуры,*
- регистр сведений *ЗначенияСвойствНоменклатуры.*

Эти объекты мы не привязывали ни к каким подсистемам, но они будут нужны для работы с характеристиками номенклатуры.

Роль *Мастер* готова.

Расчетчик

В заключение нам с вами осталось создать две роли: *Бухгалтер* и *Расчетчик*.

Мы разделим права по расчету зарплаты и по ведению бухгалтерского учета.

Дело в том, что в ООО «На все руки мастер» есть бухгалтер и помощник бухгалтера. Помощник бухгалтера занят в основном расчетом зарплаты, но иногда это делает и главный бухгалтер.

Поэтому главному бухгалтеру необходимо будет назначить обе роли, в то время как помощнику – только роль *Расчетчик*.

Создадим новый объект конфигурации Роль с именем *Расчетчик*.

В окне редактирования прав установим их по подсистеме *РасчетЗарплаты* (и не забудем запретить интерактивное удаление). А также установим право *Просмотр* для объекта конфигурации: *Регистр накопления Продажи.*

Роль *Расчетчик* готова.

Бухгалтер

В заключение создадим объект конфигурации *Роль* с именем *Бухгалтер*. В окне редактирования прав установим их по подсистеме *Бухгалтерия*.

После этого отфильтруем список объектов по этой подсистеме и для справочника *Номенклатура* запретим добавление, изменение и удаление.

Также запретим интерактивное удаление для всех объектов.

Затем снимем фильтр и установим все права, кроме интерактивного удаления для следующих объектов конфигурации:

- справочник *Субконто,*
- регистр бухгалтерии *Управленческий.*

А также установим право *Просмотр* для следующих объектов конфигурации:

- справочник *Склады,*
- справочник *ВариантыНоменклатуры,*
- справочник *ДополнительныеСвойстваНоменклатуры,*
- план видов характеристик *СвойстваНоменклатуры,*
- регистр сведений *ЗначенияСвойствНоменклатуры*.

Права на запуск клиентских приложений

В заключение установим права на запуск клиентского приложения для каждой роли.

Для этого воспользуемся другим, более удобным инструментом – редактором *Все роли*. В дереве объектов конфигурации выделим ветку *Роли* и в контекстном меню выполним команду *Все роли*. В списке объектов конфигурации выделим корень, и для всех ролей установим право *Тонкий клиент* (рис. 22.5).

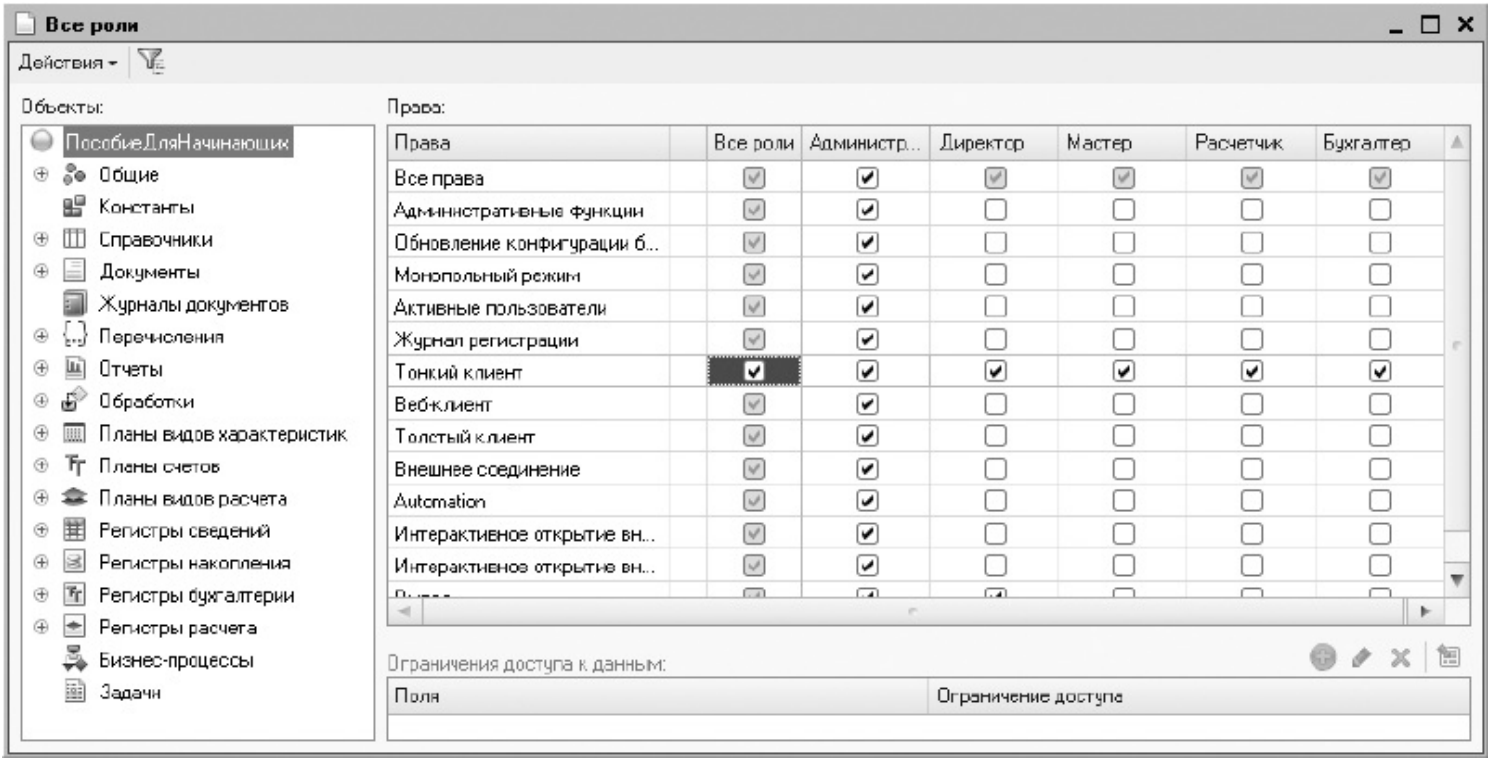


Рис. 22.5. Список прав для роли «Бухгалтер»

Тем самым мы разрешили всем пользователям подключаться к информационной базе только с помощью тонкого клиента. Администратор же имеет возможность подключаться и с помощью других клиентских приложений. Это может понадобиться ему, например, для создания планировщика заданий, о котором шла речь в разделе «Планировщик заданий».

Список прав для каждой роли можно получить, выполнив в окне редактирования прав команду *Действия* > *Вывести список* (рис. 22.6).

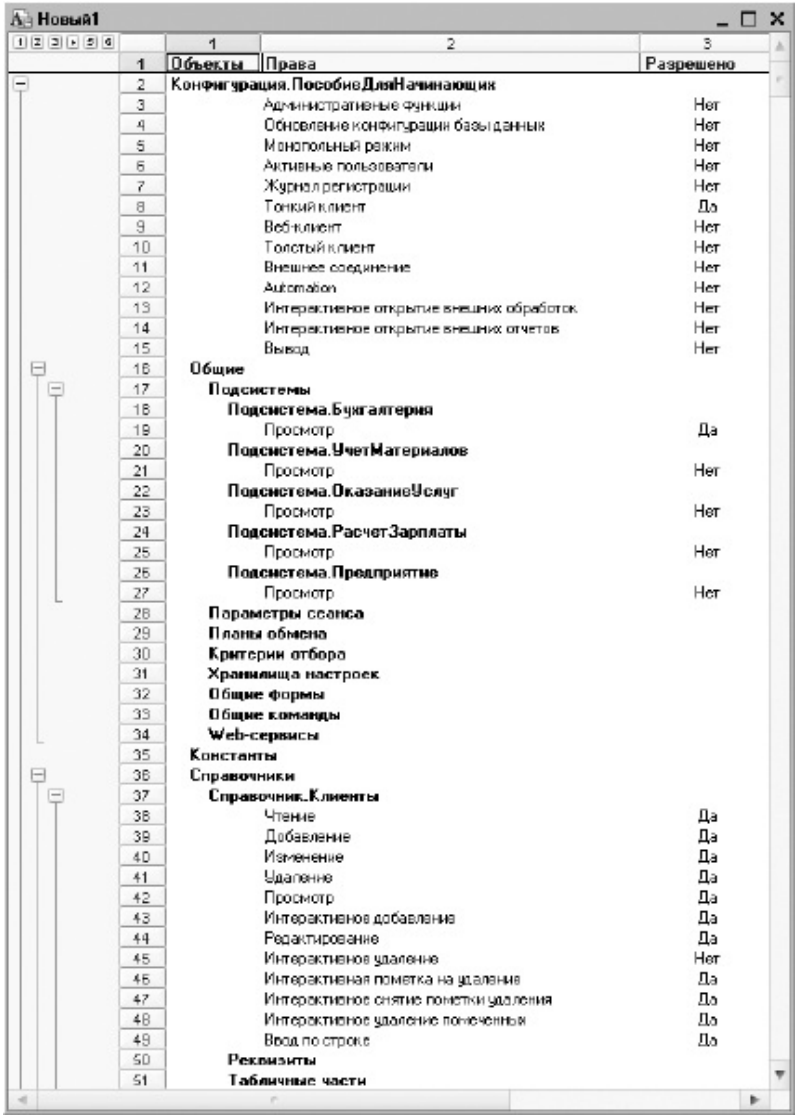

Новый1
1 2 3
1 Объекты Права Разрешено
2 Конфигурация.ПособиеДляНачинающих
3 Административные функции Нет
4 Обновление конфигурации базы данных Нет
5 Монопольный режим Нет
6 Активные пользователи Нет
7 Журнал регистрации Нет
8 Тонкий клиент Да
9 Веб-клиент Нет
10 Толстый клиент Нет
11 Внешнее соединение Нет
12 Automation Нет
13 Интерактивное открытие внешних обработок Нет
14 Интерактивное открытие внешних отчетов Нет
15 Вывод Нет
16 Общие
17 Подсистемы
18 Подсистема.Бухгалтерия
19 Просмотр Да
20 Подсистема.УчетМатериалов
21 Просмотр Нет
22 Подсистема.ОказаниеУслуг
23 Просмотр Нет
24 Подсистема.РасчетЗарплаты
25 Просмотр Нет
26 Подсистема.Предприятие
27 Просмотр Нет
28 Параметры сеанса
29 Планы обмена
30 Критерии отбора
31 Хранилища настроек
32 Общие формы
33 Общие команды
34 Web-сервисы
35 Константы
36 Справочники
37 Справочник.Клиенты
38 Чтение Да
39 Добавление Да
40 Изменение Да
41 Удаление Да
42 Просмотр Да
43 Интерактивное добавление Да
44 Редактирование Да
45 Интерактивное удаление Нет
46 Интерактивная пометка на удаление Да
47 Интерактивное снятие пометки удаления Да
48 Интерактивное удаление помеченных Да
49 Ввод по строке Да
50 Реквизиты
51 Табличные части

Рис. 22.6. Список прав для роли «Бухгалтер»

Аналогичный список, но только для всех ролей, которые есть в конфигурации, можно получить из редактора *Все роли*.

Добавление новых пользователей

Для того чтобы иметь возможность отличать друг от друга пользователей, работающих с информационной базой, в системе «1С:Предприятие 8» существует *список пользователей*.

Можно создавать и удалять пользователей системы, назначать им роли и т. д.

В режиме «Конфигуратор»

Прежде чем мы приступим к созданию пользователей, необходимо выполнить обновление конфигурации базы данных (*Конфигурация* > *Обновить конфигурацию базы данных (F7)* ), поскольку пользователю можно поставить в соответствие только те роли, которые существуют в конфигурации базы данных.

После того как обновление произведено, выполним команду главного меню *Администрирование* > *Пользователи*.

Откроется список пользователей системы. Пока что он пуст, поэтому добавим нового пользователя (*Действия* > *Добавить*) или нажмем кнопку *Добавить* в командной панели окна (рис. 22.7).

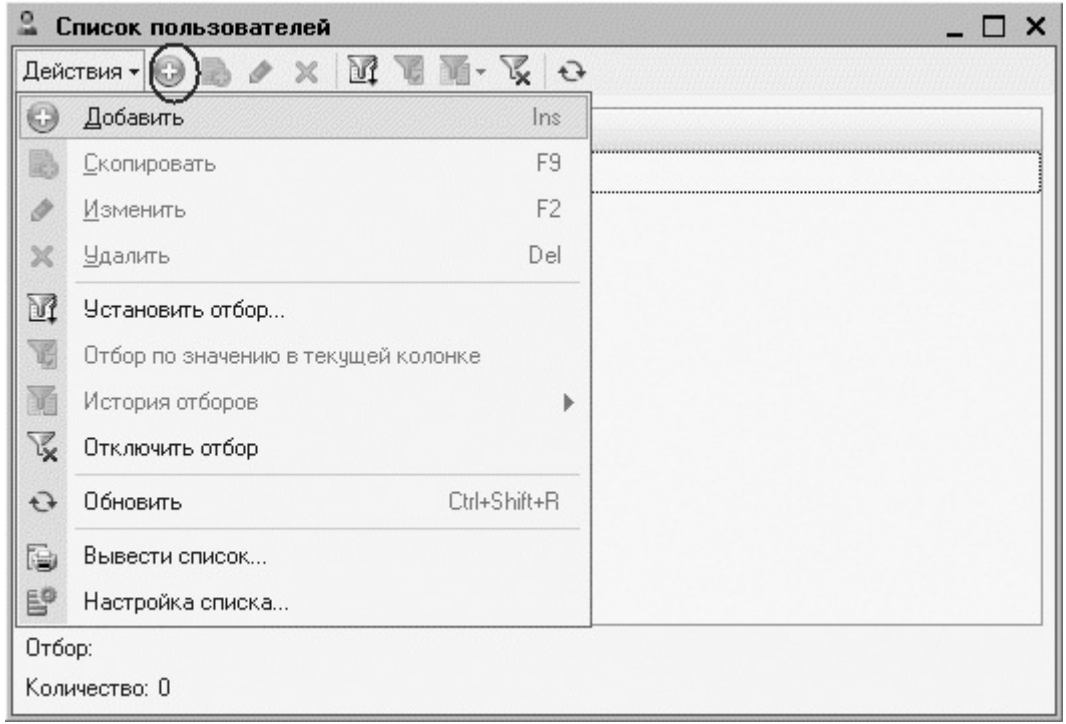


Рис. 22.7. Список пользователей

Откроется окно редактирования пользователя (рис. 22.8).

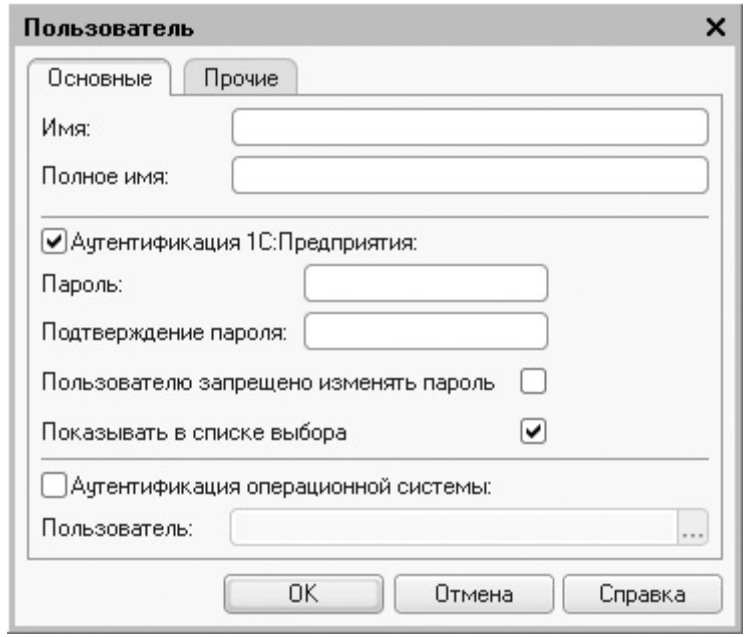


Рис. 22.8. Окно редактирования пользователя

ВНИМАНИЕ!

Если вы используете учебную версию платформы «1С:Предприятие 8.2», то возможность задания паролей пользователей и аутентификация операционной системы будут недоступны. Это ограничения учебной версии.

Имя пользователя – это идентификатор, который будет появляться в окне выбора пользователей при запуске системы в режиме *1С:Предприятие*.

Полное имя – строка, которая может быть использована внутри конфигурации при выводе различной справочной информации. Хорошим стилем администрирования считается указание в качестве полного имени фамилии, имени и отчества пользователя (без сокращений).

Следующие две области окна посвящены способам аутентификации пользователя.

Аутентификация средствами «1С:Предприятия» подразумевает, что после запуска системы пользователю будет предложено выбрать имя одного из пользователей системы и ввести пароль. Если введенный пароль соответствует сохраненному в системе для этого идентификатора пользователя, система открывается с правами, которые указаны для этого пользователя. При этом он сможет поменять пароль, если флажок *Пользователю запрещено изменять пароль* не установлен.

Аутентификация операционной системы подразумевает, что при запуске системы «1С:Предприятие 8» от пользователя не требуется никакой дополнительной информации. Система «1С:Предприятие 8» определяет, под каким пользователем запущена операционная система и затем обращается к своему списку пользователей. Если она находит в нем пользователя, которому поставлен в соответствие текущий пользователь операционной системы, информационная база открывается с правами, указанными для этого

пользователя.

Приступим к созданию пользователей.

Зададим имя пользователя *Администратор*, полное имя тоже *Администратор*. Перейдем на закладку *Прочие*. Отметим роль *Администратор* и язык конфигурации выберем *Русский* (рис. 22.9).

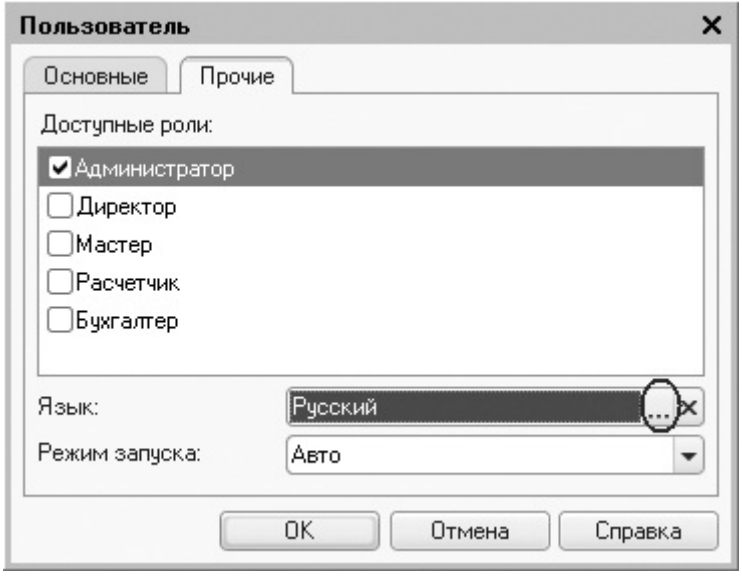


Рис. 22.9. Закладка «Прочие» окна редактирования пользователя

Нажмем *ОК*.

После этого создадим остальных пользователей системы (рис. 22.10). Для всех них мы будем использовать аутентификацию «1С:Предприятия» и русский язык.

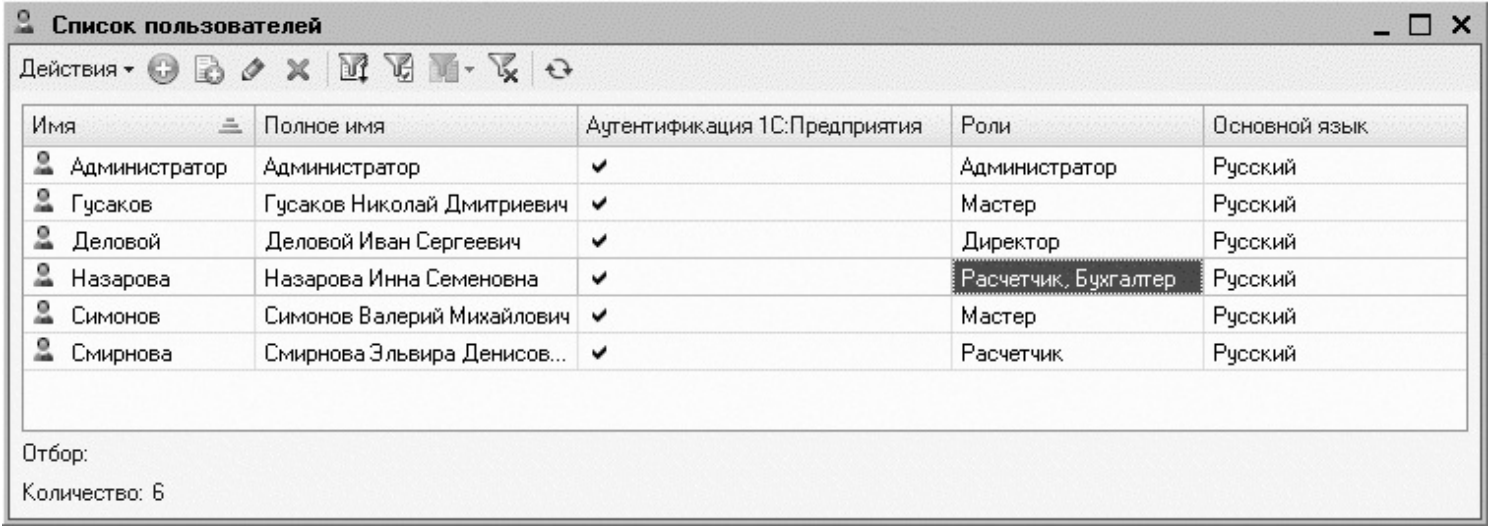


Рис. 22.10. Список пользователей системы

ПРИМЕЧАНИЕ

Если некоторые колонки, например ***Роли****, не видны в списке пользователей, можно выполнить настройку списка, выполнив команду* ***Действия*** *>* ***Настройка списка…****, и добавить нужные колонки.*

Обратите внимание, что главному бухгалтеру Назаровой поставлены в соответствие две роли: *Расчетчик* и *Бухгалтер*, поскольку она должна иметь

возможность не только вести бухгалтерский учет, но и рассчитывать зарплату.

Список пользователей, зарегистрированных в системе, можно получить, выполнив команду *Действия > Вывести список…*

Ограничение доступа к данным на уровне записей и полей базы данных

В завершение занятия мы покажем, как можно ограничить доступ к данным более точно в зависимости от самих данных, которые хранятся в информационной базе.

Для этого в системе «1С:Предприятие 8» используется *механизм ограничения доступа на уровне записей и полей базы данных*. Этот механизм позволяет для четырех основных прав (чтение, добавление, изменение и удаление) уточнить, какие же именно данные информационной базы будут доступны пользователю.

Такое уточнение записывается на специальном языке, являющемся подмножеством языка запросов.

Далее, на примере документа *Начисления сотрудникам*, мы рассмотрим небольшой пример, когда мастерам нужно дать возможность просмотреть

начисленную им зарплату, но руководство запрещает им доступ к информации о начисленной премии.

Другими словами, мастерам нужно запретить просмотр тех документов *Начисления сотрудникам*, в которых есть записи о начислении премии.

В режиме «Конфигуратор»

Для решения этой задачи сначала установим для роли *Мастер* право *Просмотр* для документа *НачисленияСотрудникам*.

Поскольку этот документ принадлежит подсистеме *РасчетЗарплаты*, дадим право на просмотр этой подсистемы. Также дадим права на просмотр справочника *ВидыГрафиковРаботы* и плана видов расчета *Основные начисления*, т. к. ссылки на эти объекты используются в документе *НачисленияСотрудникам*. Вернемся к редактированию прав роли *Мастер*.

Как мы видим, при установке права *Просмотр* право *Чтение* документа *НачисленияСотрудникам* установилось автоматически. Выделим его. В правой нижней части экрана находится поле *Ограничение доступа к данным*. Нажмем кнопку *Добавить* (рис. 22.11).

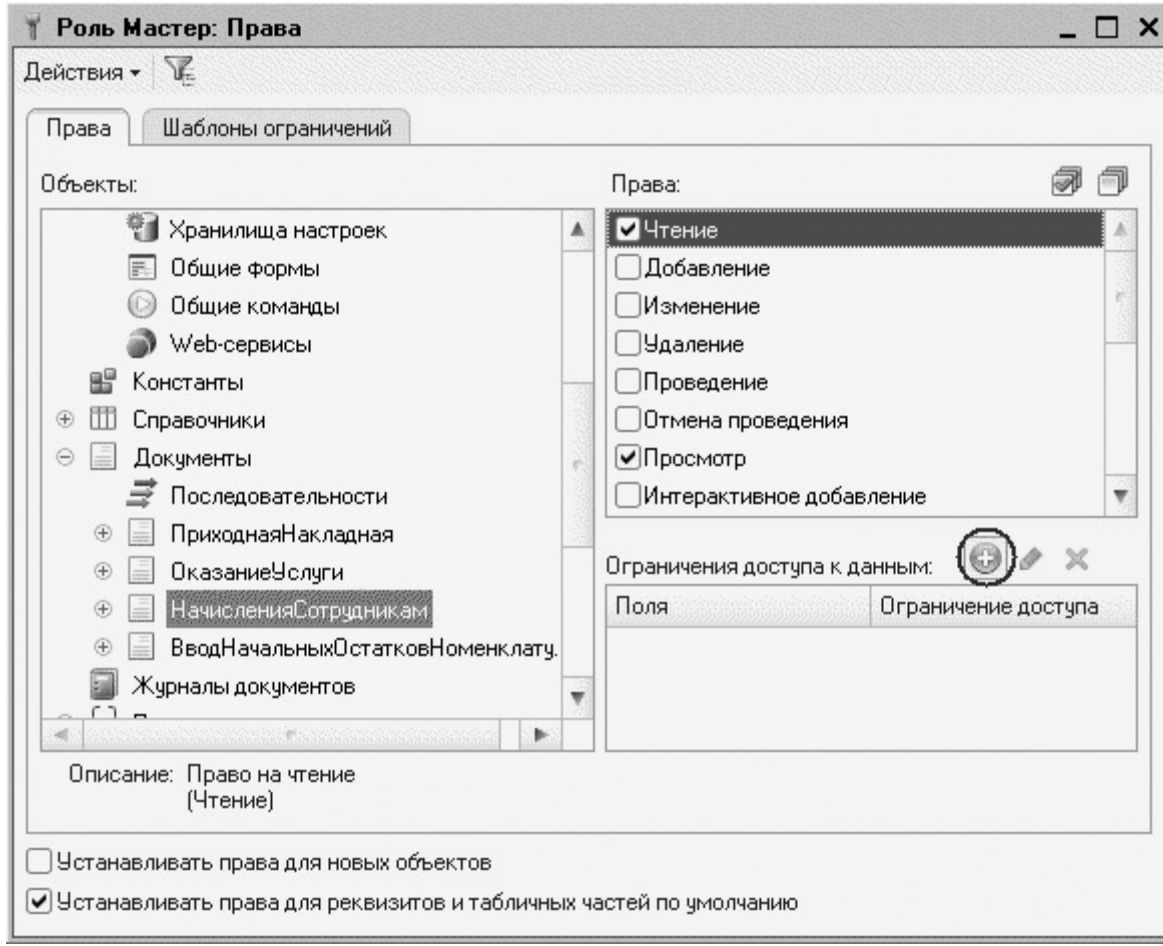


Рис. 22.11. Добавление ограничения доступа для роли «Мастер»

Мы хотим запретить доступ ко всем полям документа *Начисление*

сотрудникам.

Поэтому мы не будем выбирать поля, а нажмем кнопку выбора в поле *Ограничение доступа* (рис. 22.12).

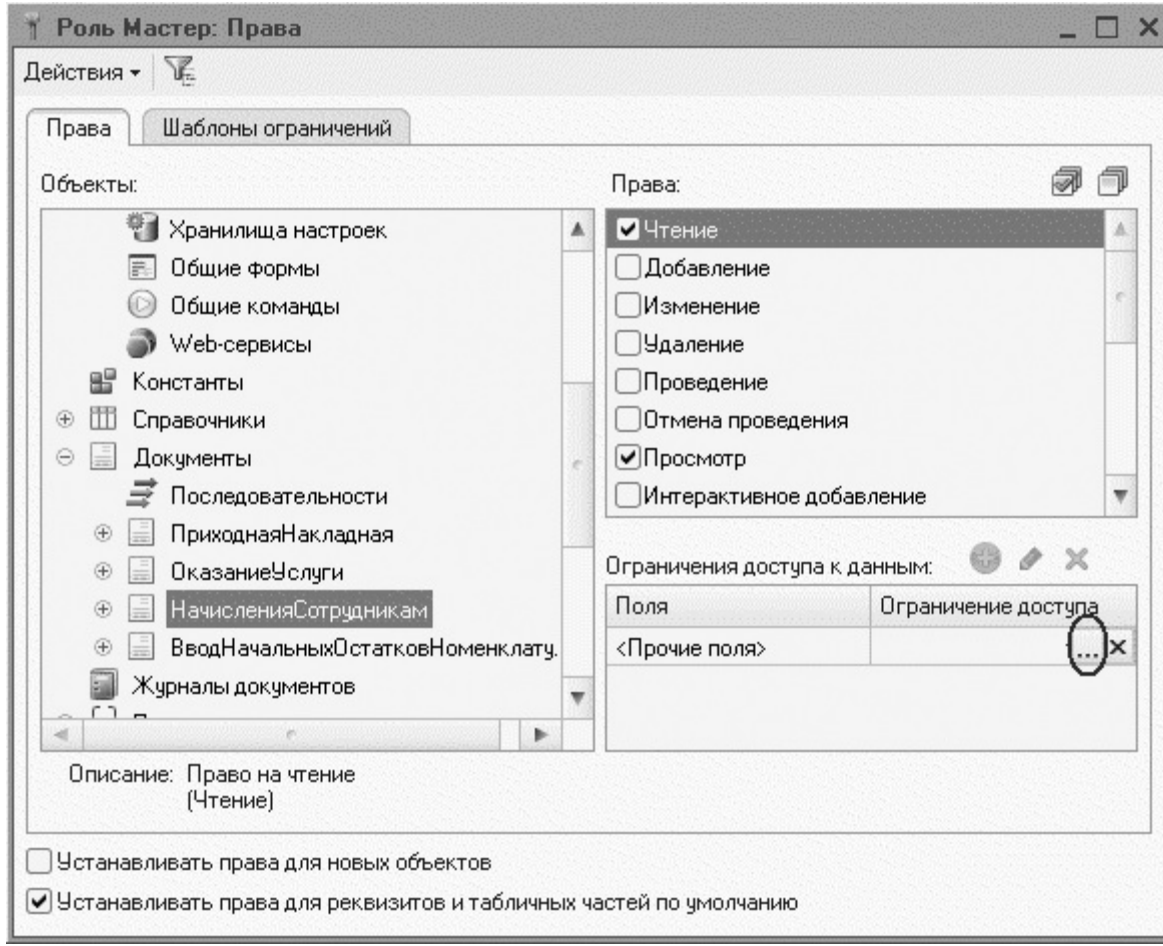


Рис. 22.12. Вызов редактора ограничений доступа

Откроется окно *Ограничение доступа*, в котором можно задать текст на

специальном языке, который является подмножеством языка запроса.

Для облегчения работы мы воспользуемся конструктором запроса. Нажмем кнопку *Конструктор запроса*.

Откроется конструктор ограничений доступа к данным. Он похож на конструктор запросов (см. рис. 22.13).

Таблица *НачисленияСотрудникам* автоматически попала на закладку *Таблицы и поля*, а конструктор открылся на закладке *Условия*.

Перенесем в список условий поле *ВидРасчета* табличной части *Начисления*, установим флажок *Произвольное* и заполним правую часть условия, как показано в листинге 22.1 (рис. 22.13).

Листинг 22.1. Ограничение доступа к данным

```
НачисленияСотрудникам.Начисления.ВидРасчета
ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
```

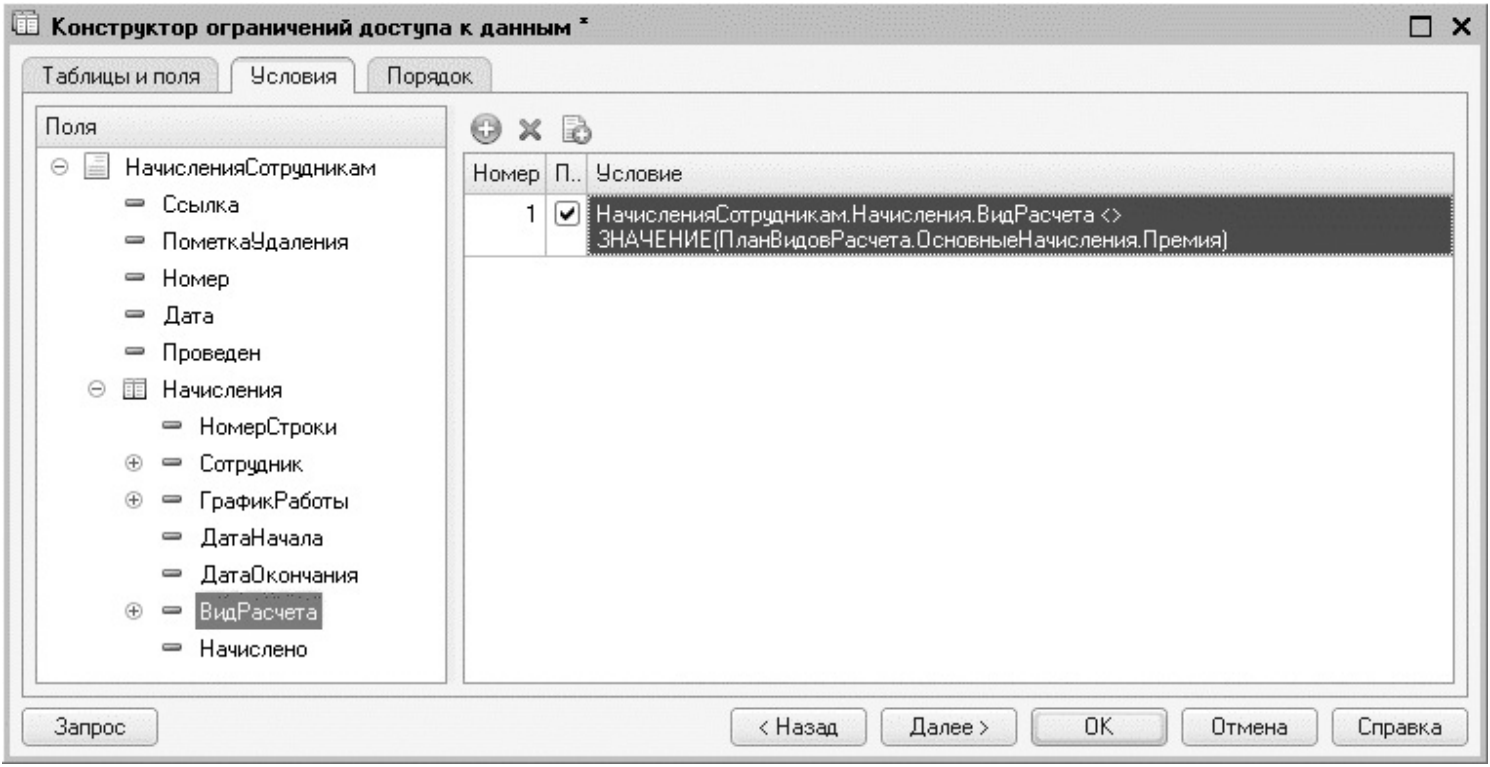


Рис. 22.13. Конструктор ограничений доступа к данным

Нажмем *ОК* (рис. 22.14).

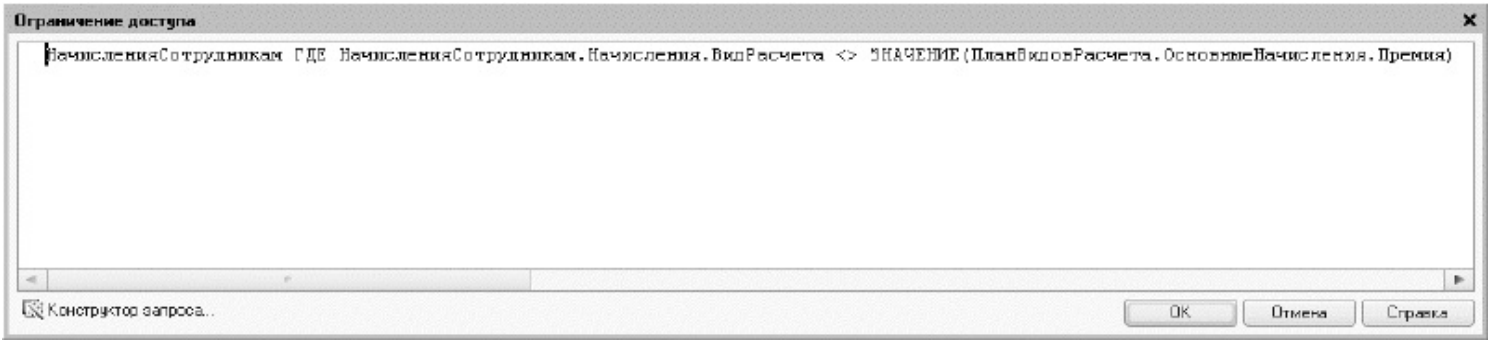


Рис. 22.14. Текст ограничения доступа к данным

Текущий пользователь имеет право прочитать или изменить некоторый объект базы данных только в том случае, если ограничение доступа предоставляет ему такое право. То есть когда условие ограничения истинно.

В нашем случае пользователь сможет прочитать документ *Начисления сотрудникам НачисленияСотрудникам ГДЕ…* только в том случае, если в его табличной части *Начисления … ГДЕ НачисленияСотрудникам.Начисления …* есть виды расчета *… ГДЕ НачисленияСотрудникам.Начисления.ВидРасчета …,* не являющиеся видом расчета *Премия …* <> *ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)*

Нажмем *ОК*.

Окно ограничений доступа к данным для роли *Мастер* будет выглядеть

следующим образом (рис. 22.15).

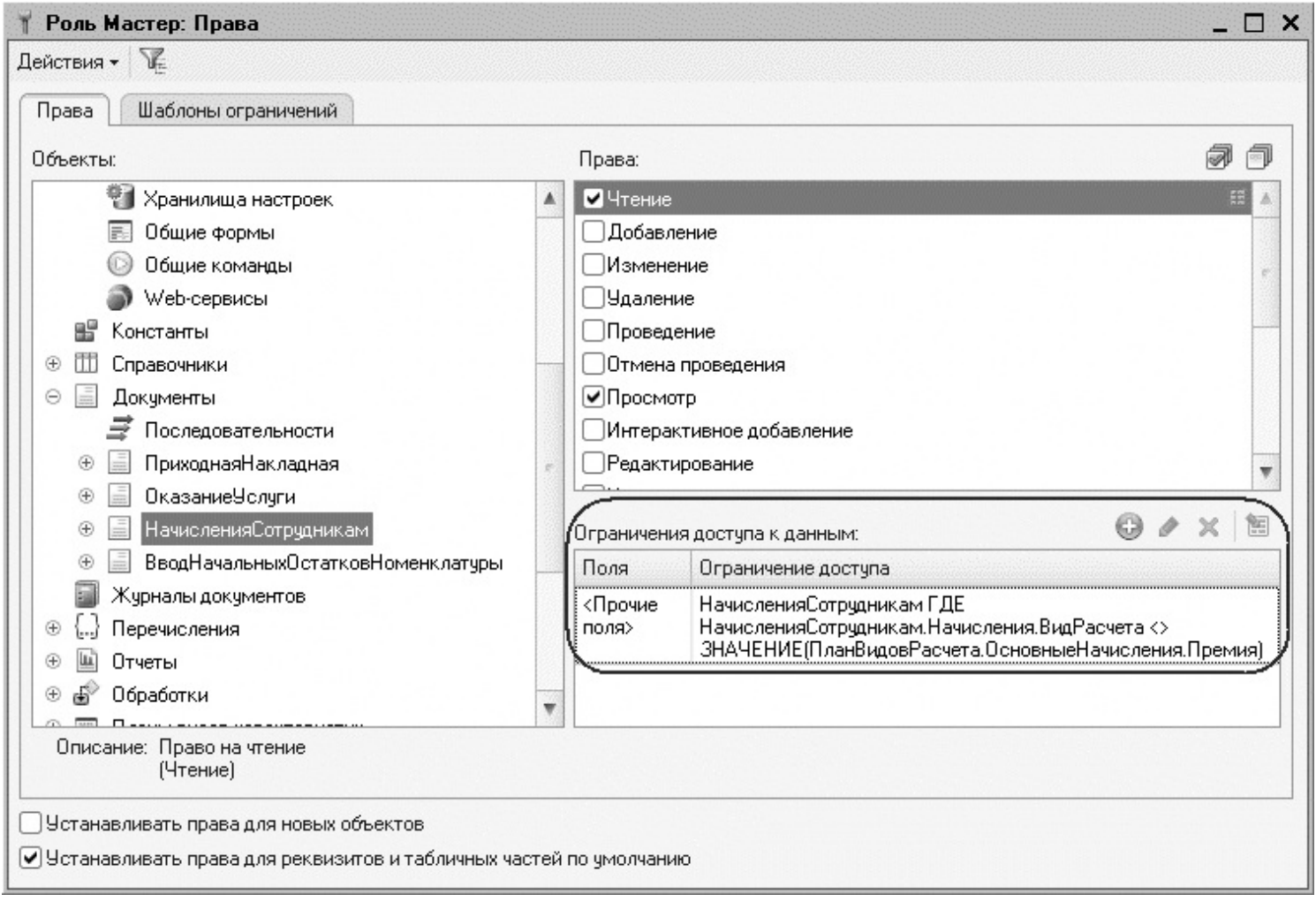


Рис. 22.15. Ограничение доступа для роли «Мастер»

В режиме «1С:Предприятие»

Обновим информационную базу, нажав *F7*, и запустим «1С:Предприятие» для пользователя с ролью *Мастер*, например, для пользователя *Гусаков*.

В разделе *Расчет зарплаты* откроем список документов *НачислениеСотрудникам* (рис. 22.16).

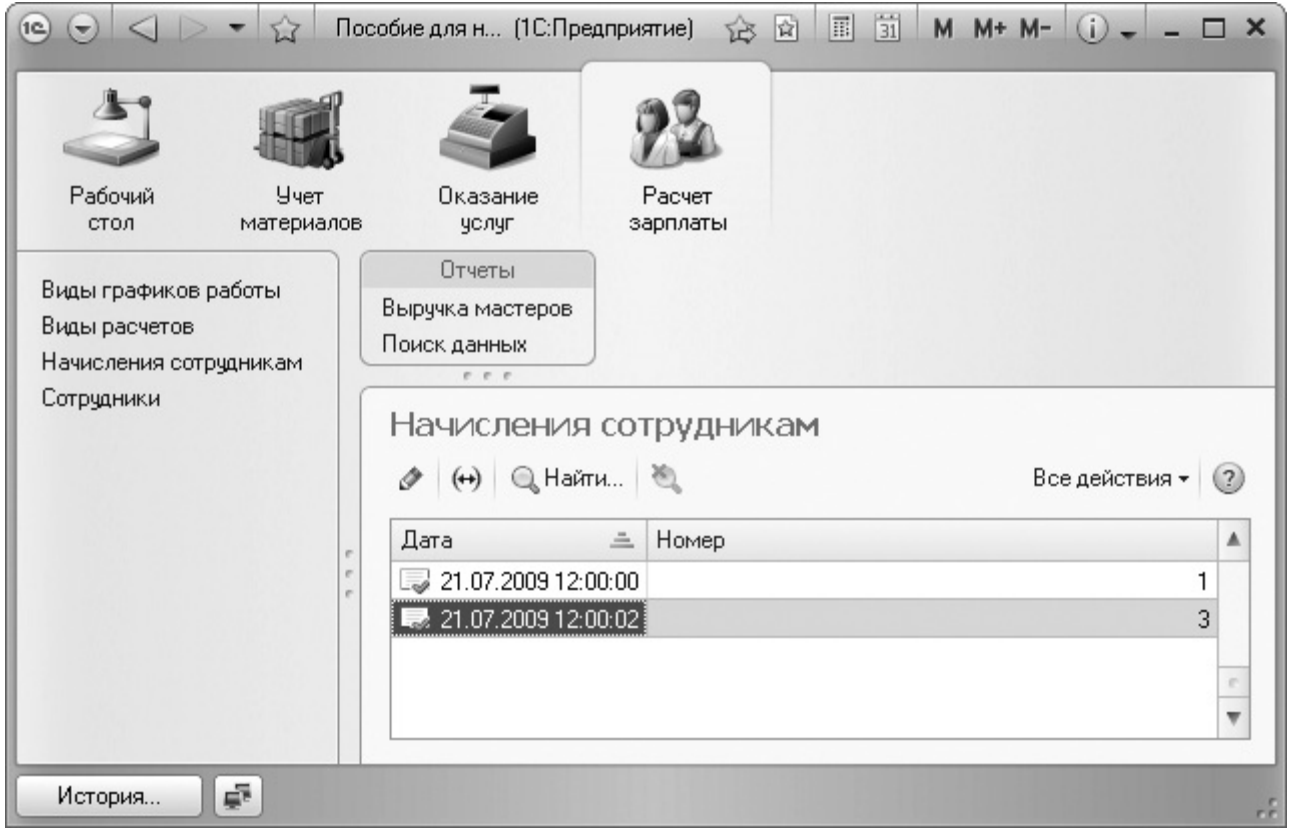


Рис. 22.16. Список документов «Начисления сотрудникам»

Как мы видим, в списке показаны только документы № 1 и № 3, так в документе № 2 начисляется премия.

Немного усложним задачу. Мы все так же не хотим, чтобы мастер видел

начисленные премии, но в то же время не хотим скрывать от него факт существования такого документа.

Другими словами, в списке документов мастер должен его видеть, но не должен иметь возможности открыть его.

В режиме «Конфигуратор»

Вернемся в конфигуратор и посмотрим на наше ограничение.

Мы не задавали никаких полей, поэтому ограничение применяется ко всем полям документа (см. рис. 22.15).

Поэтому сейчас мы безусловно разрешим читать те поля документа, которые необходимы для отображения документа в списке.

Тем самым мы разрешим документу отображаться в списке. Но поскольку существующее условие на прочие поля мы удалять не будем, то открыть документ, как и раньше, можно будет только в том случае, если в его табличной части есть виды расчета, отличные от *Премия*.

Добавим к ограничениям доступа еще одно условие.

В списке полей выберем поля:

- *Ссылка*,
- *ПометкаУдаления,*
- *Номер*,
- *Дата*,
- *Проведен.*

В ограничении доступа напишем *ГДЕ ИСТИНА* (рис. 22.17).

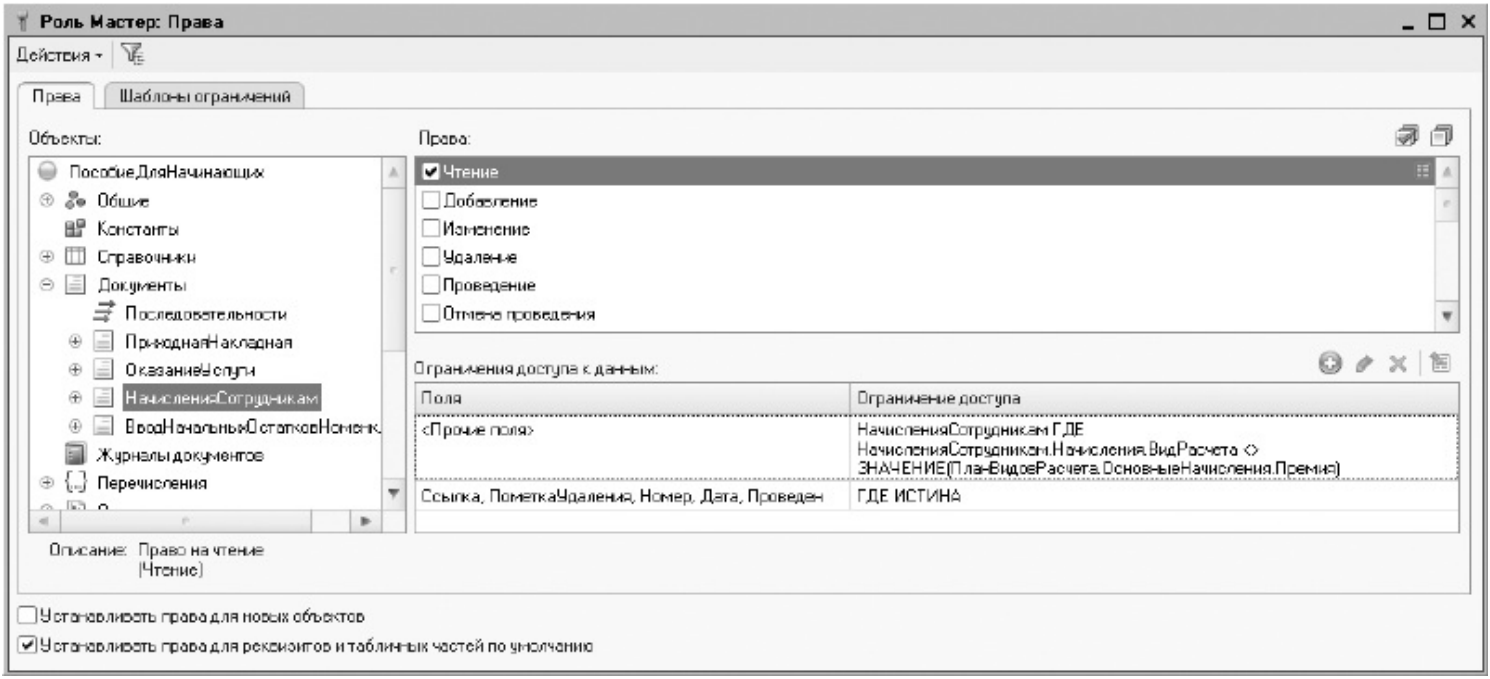


Рис. 22.17. Установка ограничений доступа к данным для роли «Мастер»

Закроем окно редактирования прав.

В режиме «1С:Предприятие»

Обновим информационную базу, нажав *F7*, и запустим «1С:Предприятие» для пользователя с ролью *Мастер*, например, для пользователя *Гусаков*.

В разделе *Расчет зарплаты* откроем список документов

НачисленияСотрудникам (рис. 22.18).

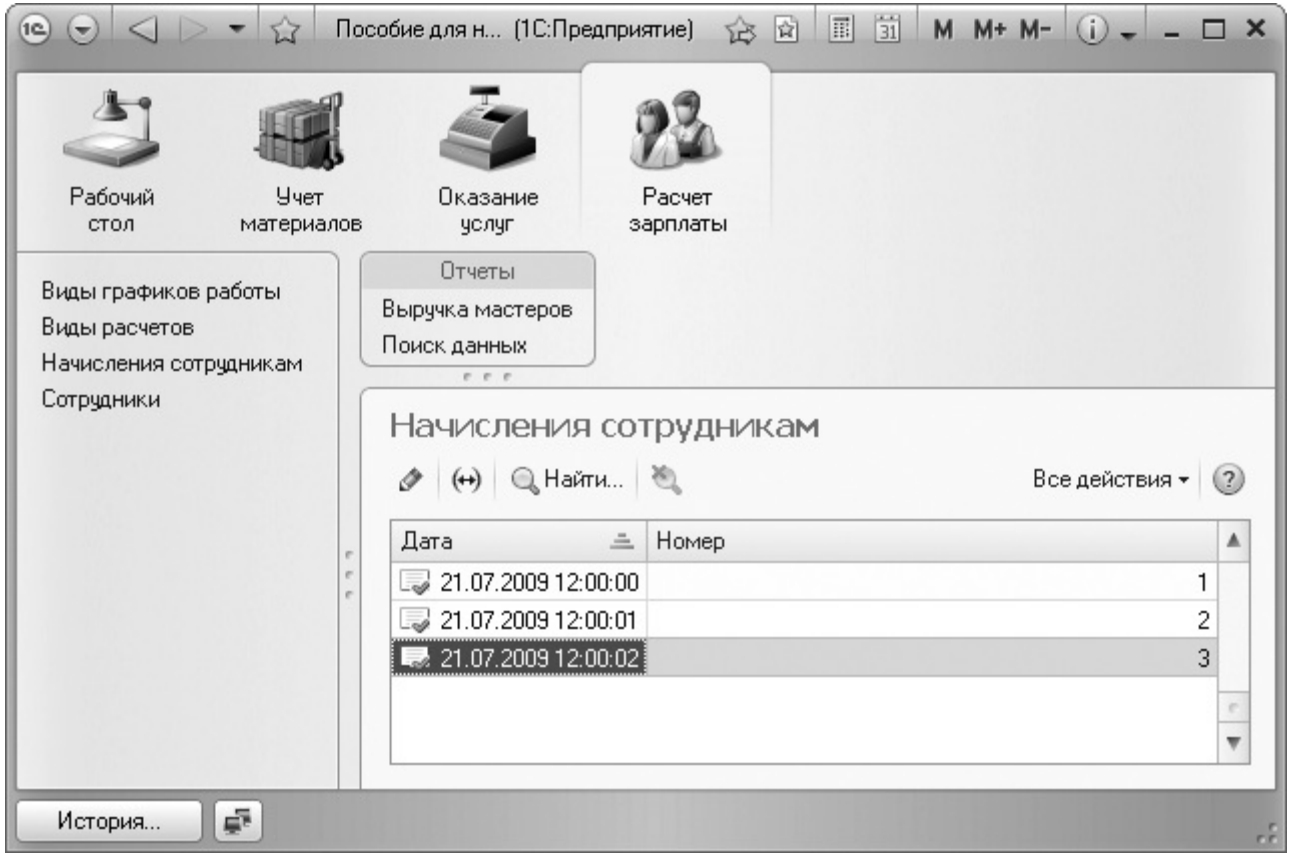


Рис. 22.18. Установка ограничений доступа к данным для роли «Мастер»

В списке документов мы увидим все документы начислений. Документы № 1 и

№ 3 мы сможем открыть и просмотреть, но при попытке открыть документ № 2 мы получим сообщение о нарушении прав доступа (рис. 22.19).

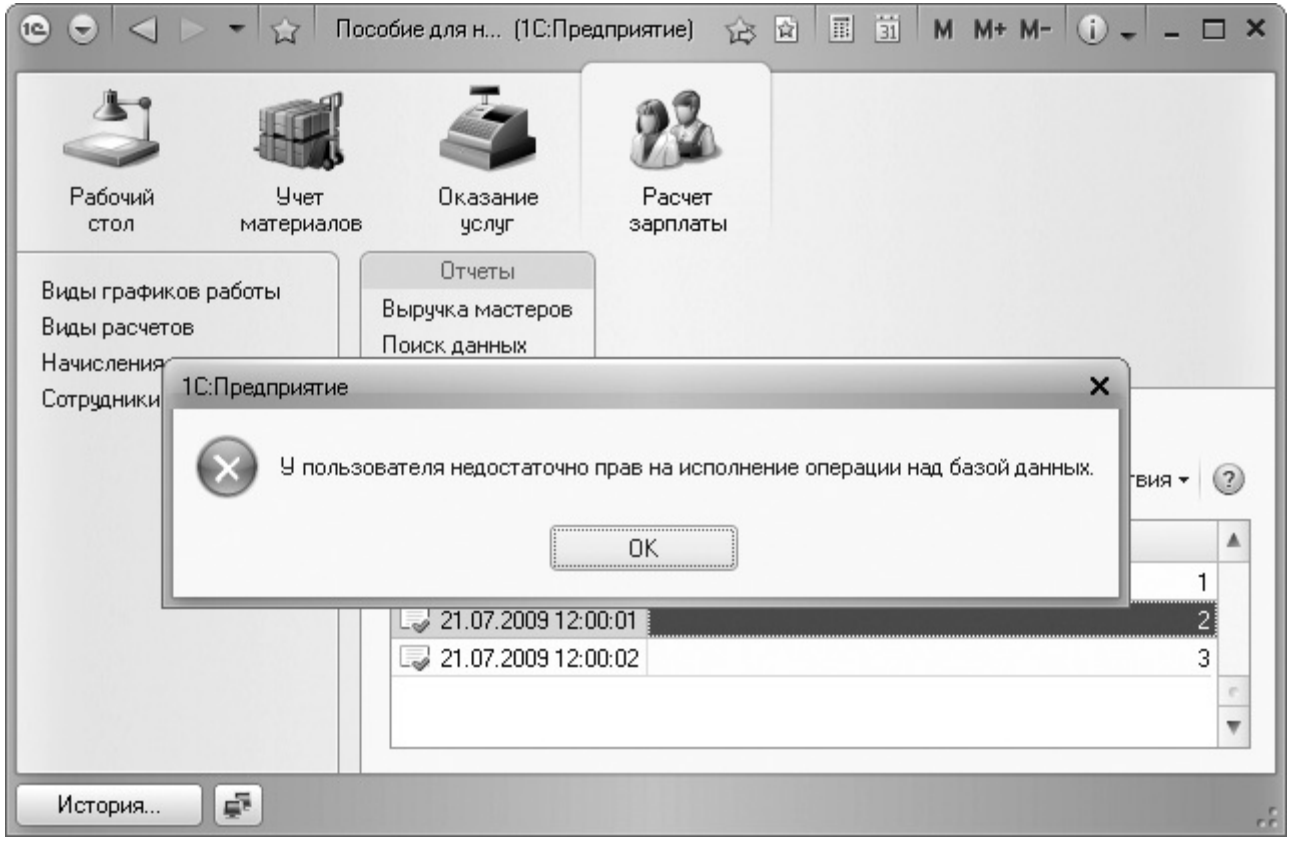


Рис. 22.19. Установка ограничений доступа к данным для роли «Мастер»

То есть мы добились того, чего хотели.

Теперь обратим внимание на следующий момент.

Все хорошо, пока в документе № 2 содержатся записи только о расчете премии. Но вспомним как формулируется наше ограничение доступа: пользователь сможет прочитать документ *Начисления сотрудникам* только в том случае, если в его табличной части *Начисления* есть виды расчета, не являющиеся видом расчета *Премия*.

Это значит, что если в этом документе окажутся виды расчета, отличные от *Премия*, мастер сможет его открыть и просмотреть.

Убедимся в этом.

Запустим «1С:Предприятие» от имени пользователя *Администратор*.

В разделе *Расчет зарплаты* откроем список документов *НачислениеСотрудникам*. Откроем документ № 2 и скопируем любую его строку. В новой строке изменим вид расчета на *Оклад*. Проведем и закроем документ. Завершим сеанс работы.

Теперь запустим «1С:Предприятие» от имени пользователя *Гусаков*. Точно так же в разделе *Расчет зарплаты* откроем список документов *НачислениеСотрудникам*.

Откроем документ № 2. Документ откроется, и мы увидим все его строки.

В режиме «Конфигуратор»

Вернемся в конфигуратор.

Для того чтобы документ невозможно было просмотреть и в этой ситуации, нам нужно будет изменить существующее условие ограничения доступа.

Новое условие будет более сложным, поэтому заодно мы продемонстрируем использование шаблонов в ограничениях доступа.

Итак, откроем роль *Мастер* и перейдем на закладку *Шаблоны ограничений* (рис. 22.20).

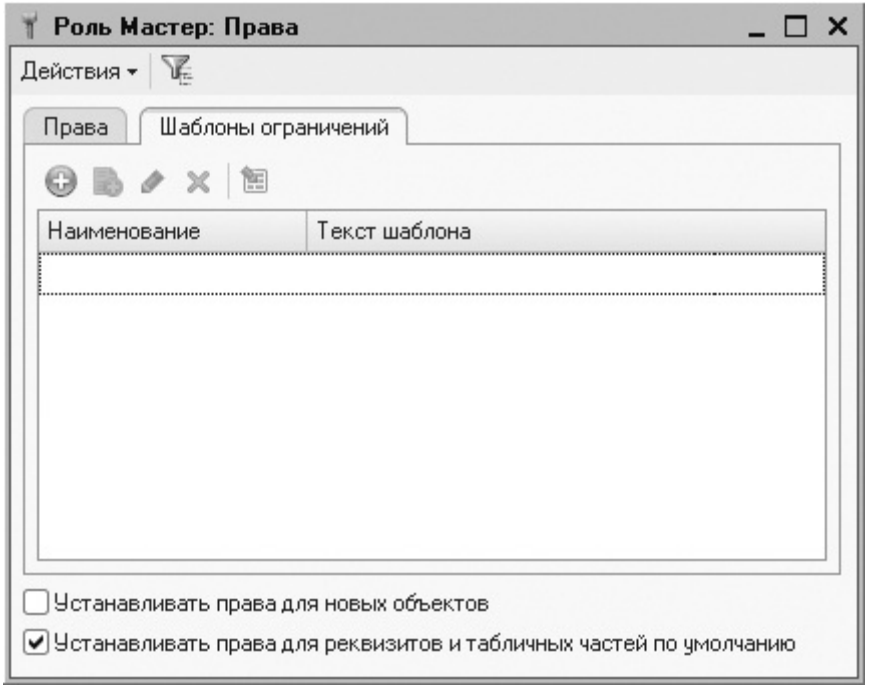


Рис. 22.20. Установка ограничений доступа к данным для роли «Мастер»

Здесь добавим новый шаблон, назовем его *ЕстьПремия*.

Текст шаблона будет выглядеть следующим образом:

Листинг 22.2. Ограничение доступа к данным

```
ВЫБРАТЬ
  1
```

```
ИЗ
  Документ.НачисленияСотрудникам.Начисления
ГДЕ
  Документ.НачисленияСотрудникам.Начисления.ВидРасчета =
  ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
    И Документ.НачисленияСотрудникам.Начисления.Ссылка = #Параметр(1).Ссылка
```

По сути это запрос к табличной части документа *НачисленияСотрудникам*, который либо не вернет нам ничего, либо вернет одну запись с одним полем, в котором будет значение 1.

Такую запись он вернет нам в том случае, если в табличной части документа есть вид расчета *Премия*.

Второе условие в этом запросе нужно нам для того, чтобы указать, табличная часть какого именно документа нас интересует. В этом условии используется возможность указания параметров в шаблоне.

Листинг 22.3. Ограничение доступа к данным

```
И Документ.НачисленияСотрудникам.Начисления.Ссылка = #Параметр(1).Ссылка
```

Вместо *#Параметр(1)* будет подставлена та строка, которую мы укажем при вызове этого шаблона в условии ограничения доступа.

Теперь вернемся на закладку *Права*.

В имеющемся ограничении прав доступа для прочих полей заменим старый текст новым (листинг 22.4).

Листинг 22.4. Ограничение доступа к данным

```
ДокНачисления ГДЕ НЕ 1 В (#ЕстьПремия("ДокНачисления"))
```

Здесь с помощью конструкции *#ЕстьПремия("ДокНачисления")* мы обращаемся к нашему шаблону. Текст шаблона просто механически будет подставлен в это место, причем строка *ДокНачисления* заменит собой первый параметр шаблона (*#Параметр(1)*).

Как мы уже говорили, если в табличной части есть начисление *Премия*, запрос в шаблоне вернет единственную запись со значением 1.

Поэтому это условие (см. листинг 22.4) разрешит нам прочитать *ДокНачисления* тогда, когда запрос из шаблона не возвращает 1:

```
ГДЕ НЕ 1 В (#ЕстьПремия("ДокНачисления"))
```

То есть тогда, когда в табличной части нет начисления *Премия*.

Можно было бы записать это условие ограничения и без использования шаблонов.

Но, во-первых, такая запись была бы менее читаемой (листинг 22.5), а во-вторых, использование шаблонов позволяет выделить и не дублировать части условий ограничений, которые могут использоваться в разных условиях.

Листинг 22.5. Ограничение доступа к данным

```
ДокНачисления ГДЕ НЕ 1 В (
  ВЫБРАТЬ
    1
  ИЗ
    Документ.НачисленияСотрудникам.Начисления
  ГДЕ
    Документ.НачисленияСотрудникам.Начисления.ВидРасчета =
    ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
      И Документ.НачисленияСотрудникам.Начисления.Ссылка =
      ДокНачисления.Ссылка)
```

Закроем окно редактирования прав. Проверим, как это работает.

В режиме «1С:Предприятие»

Обновим информационную базу, нажав *F7*, и запустим «1С:Предприятие» от имени пользователя *Гусаков*.

В разделе *Расчет зарплаты* откроем список документов *НачислениеСотрудникам*.

Как вы помните, в документе № 2 есть строки и с видом расчета *Премия*, и с видом расчета *Оклад*. Раньше этот документ у нас открывался.

Попробуем открыть его теперь.

Мы получим сообщение о нарушении прав доступа, что нам и требовалось (см. рис. 22.19).

В режиме «Конфигуратор»

Поскольку пример с ограничением прав доступа на уровне записей и полей базы данных мы делали скорее в демонстрационных целях, вернемся к исходному состоянию конфигурации.

Снимем для роли *Мастер* право *Чтение* для документа *НачислениеСотрудникам*. Снимем право *Просмотр* для подсистемы *РасчетЗарплаты*. Снимем право *Чтение* для справочника

ВидыГрафиковРаботы и для плана видов расчета *Основные начисления*. Запустим «1С:Предприятие» от имени пользователя *Администратор*. В разделе *Расчет зарплаты* откроем список документов *НачисленияСотрудникам*. Откроем документ № 2 и удалим последнюю строку (которую мы добавляли). Проведем и закроем документ.

Контрольные вопросы

- *Для чего предназначен объект конфигурации «Роль».*
- *Как создать роль, используя подсистемы конфигурации.*
- *Как создать список пользователей системы и определить их права.*
- *Чем аутентификация средствами «1С:Предприятия» отличается от аутентификации операционной системы*.

Занятие 23 (1:10). Рабочий стол и настройка командного интерфейса

Продолжительность

Ориентировочная продолжительность занятия – 1 час 10 минут.

На этом занятии мы придадим нашей конфигурации «товарный» вид, то есть усовершенствуем командный интерфейс приложения, настроим рабочий стол и видимость команд по ролям для созданных нами пользователей.

Это сделает интерфейс приложения более завершенным и, главное, более удобным для пользователя.

На самом деле от внешнего вида и организации интерфейса приложения напрямую зависит, насколько разработанное нами прикладное решение понравится пользователю.

Командный интерфейс разделов

До сих пор мы практически не занимались организацией командного интерфейса разделов (подсистем), так как занимались другими, более важными, с точки зрения разработчика, вопросами.

Теперь пришло время заняться этим очень существенным для пользователя моментом, то есть осмысленно рассортировать команды, разложить их по группам в зависимости от приоритета и частоты использования.

В режиме «Конфигуратор»

Итак, для начала зададим синоним для отчета *ПоискДанных* как *Поиск в данных*. Так будет более понятно для пользователя (рис. 23.1).

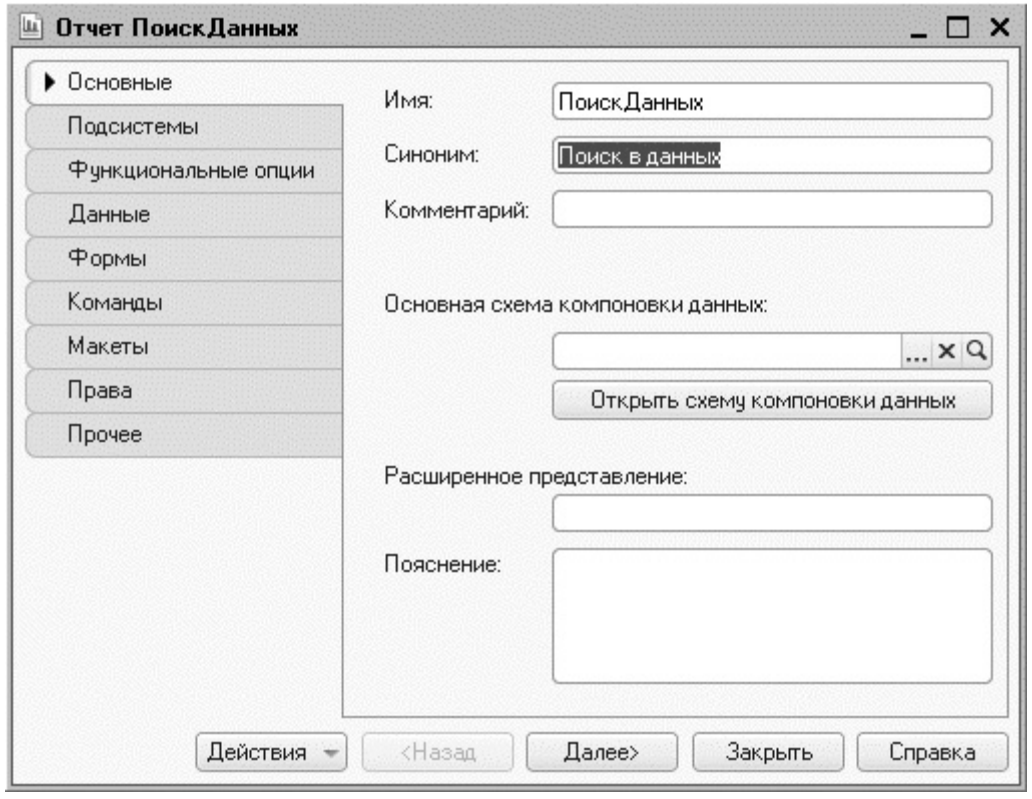


Рис. 23.1. Синоним отчета «ПоискДанных»

Затем вызовем редактор командного интерфейса подсистем *Все подсистемы* (*Общие > Подсистемы > Все подсистемы*), см. рис. 23.2.

Выделим в списке подсистем подсистему *УчетМатериалов* и в окне *Командный интерфейс* произведем следующие изменения:

- С помощью мыши переместим команду *Приходные накладные* из группы *Панель навигации.Обычное* в группу команд *Панель навигации.Важное*.
- В группе *Панель навигации.Обычное* зададим следующий порядок расположения команд:
  - *Номенклатура,*
  - *Цены на Номенклатуру,*
  - *Склады.*
- В группе *Панель навигации.См. также* уберем видимость у команды *Продажи* и зададим следующий порядок расположения команд:
  - *Остатки материалов,*
  - *Стоимость материалов.*
- В группе *Панель действий.Создать* зададим следующий порядок расположения видимых команд (порядок расположения невидимых команд нам не важен):
  - *Приходная накладная: создать,*
  - *Номенклатура: создать,*
  - *Склад: создать.*
- В группе *Панель действий.Отчеты* зададим следующий порядок расположения команд:
  - *Материалы,*
  - *Остатки материалов по свойствам.*

- Переместим команду *Поиск в данных* из группы *Панель действий.Отчеты* в группу команд *Панель действий.Сервис*.

В результате окно *Все подсистемы* примет вид (рис. 23.2).

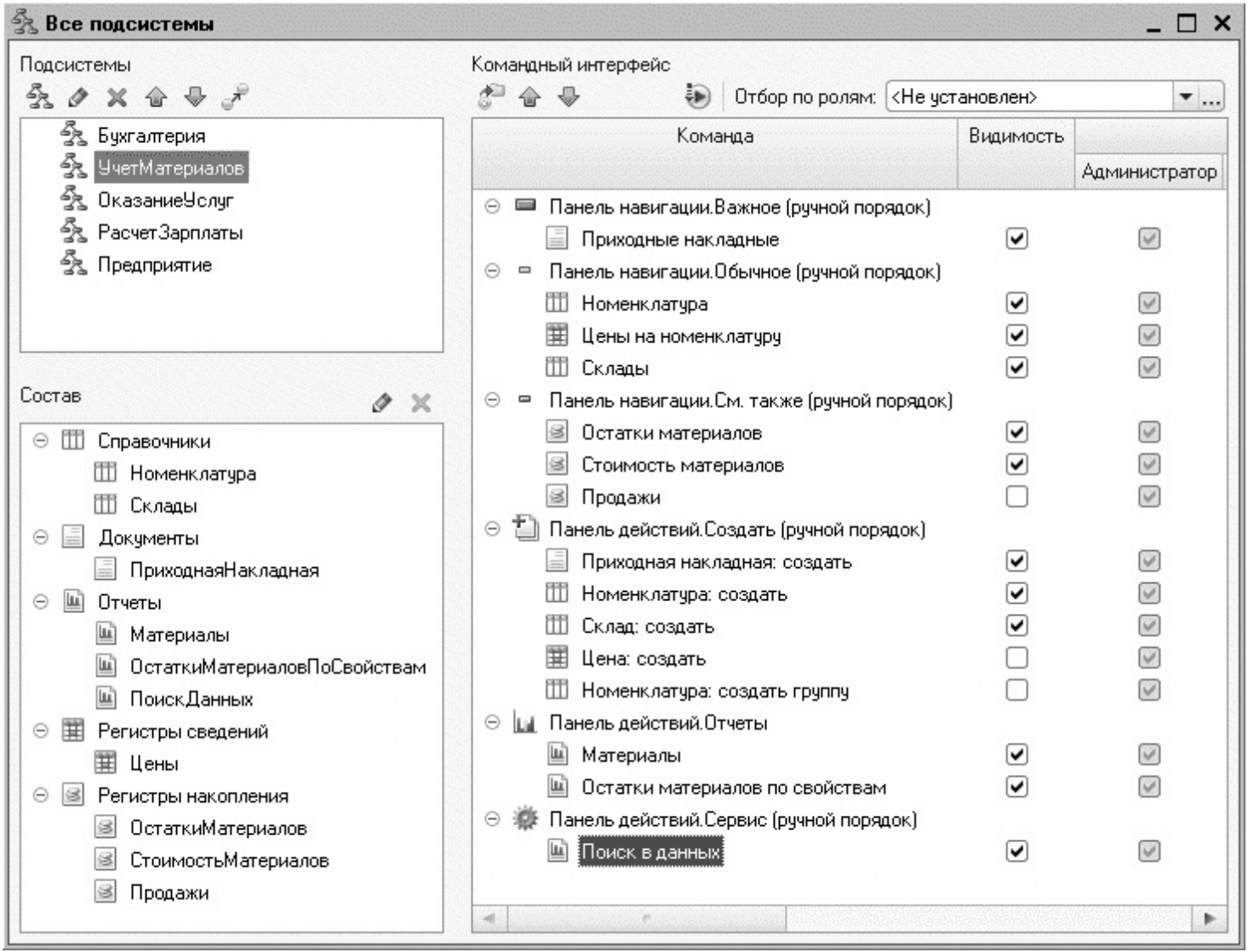

Все подсистемы
Подсистемы
Бухгалтерия
УчетМатериалов
ОказаниеУслуг
РасчетЗарплаты
Предприятие
Состав
Справочники
Номенклатура
Склады
Документы
ПриходнаяНакладная
Отчеты
Материалы
ОстаткиМатериаловПоСвойствам
ПоискДанных
Регистры сведений
Цены
Регистры накопления
ОстаткиМатериалов
СтоимостьМатериалов
Продажи
Командный интерфейс
Отбор по ролям:
<Не установлен>
Команда
Видимость
Администратор
Панель навигации.Важное (ручной порядок)
Приходные накладные
Панель навигации.Обычное (ручной порядок)
Номенклатура
Цены на номенклатуру
Склады
Панель навигации.См. также (ручной порядок)
Остатки материалов
Стоимость материалов
Продажи
Панель действий.Создать (ручной порядок)
Приходная накладная: создать
Номенклатура: создать
Склад: создать
Цена: создать
Номенклатура: создать группу
Панель действий.Отчеты
Материалы
Остатки материалов по свойствам
Панель действий.Сервис (ручной порядок)
Поиск в данных

Рис. 23.2. Командный интерфейс подсистемы «УчетМатериалов»

Поясним наши действия.

В разделе *Учет материалов* наиболее часто пользователю может понадобиться создавать приходные накладные и просматривать их список. Поэтому мы поместили команду для просмотра списка приходных накладных (*Приходные накладные*) в группу *Важное* панели навигации, а команду для создания приходных накладных (*Приходная накладная*) поместили первой в панели действий подсистемы *Учет материалов*.

Довольно часто пользователю может понадобиться создавать новую номенклатуру и просматривать список номенклатуры. Поэтому мы поместили команду для просмотра списка номенклатуры (*Номенклатура*) в группу *Обычное* панели навигации, а команду для создания номенклатуры (*Номенклатуры*) поместили второй в панели действий подсистемы *Учет материалов*. То же самое, но с меньшим приоритетом, относится к справочнику складов.

В группе *См. также* панели навигации раздела мы поместили команды *Остатки материалов* и *Стоимость материалов* для просмотра записей соответствующих регистров накопления и расположили их в порядке частоты использования. А команду для просмотра записей регистра *Продажи* мы

вообще убрали, так как в разделе *Учет материалов* вряд ли она может понадобиться.

Отчеты в панели действий раздела мы расположили в порядке их приоритета. А команду для открытия отчета *Поиск в данных* мы перенесли из группы *Отчеты* в группу *Сервис* панели действий, так как на самом деле поиск данных – это скорее сервисная операция, чем классический отчет.

Руководствуясь подобными соображениями, отредактируем командный интерфейс остальных подсистем.

Подсистема *ОказаниеУслуг* будет иметь следующий командный интерфейс (рис. 23.3):

- Группа *Панель навигации.Важное*:
  - *Оказание услуг.*
- Группа *Панель навигации.Обычное*:
  - *Клиенты,*
  - *Номенклатура,*
  - *Цены на Номенклатуру,*
  - *Сотрудники,*
  - *Склады.*

- Группа *Панель навигации.См. также*:
  - *Остатки материалов,*
  - *Стоимость материалов,*
  - *Продажи.*
- Группа *Панель действий.Создать* (только видимые команды):
  - *Оказание услуги: создать,*
  - *Клиент: создать,*
  - *Номенклатура: создать.*
- Группа *Панель действий.Отчеты*:
  - *Реестр документов Оказание услуги,*
  - *Перечень услуг,*
  - *Рейтинг услуг,*
  - *Рейтинг клиентов,*
  - *Выручка мастеров,*
  - *Материалы,*
  - *Универсальный.*
- Группа *Панель действий.Сервис*:
  - *Поиск в данных.*

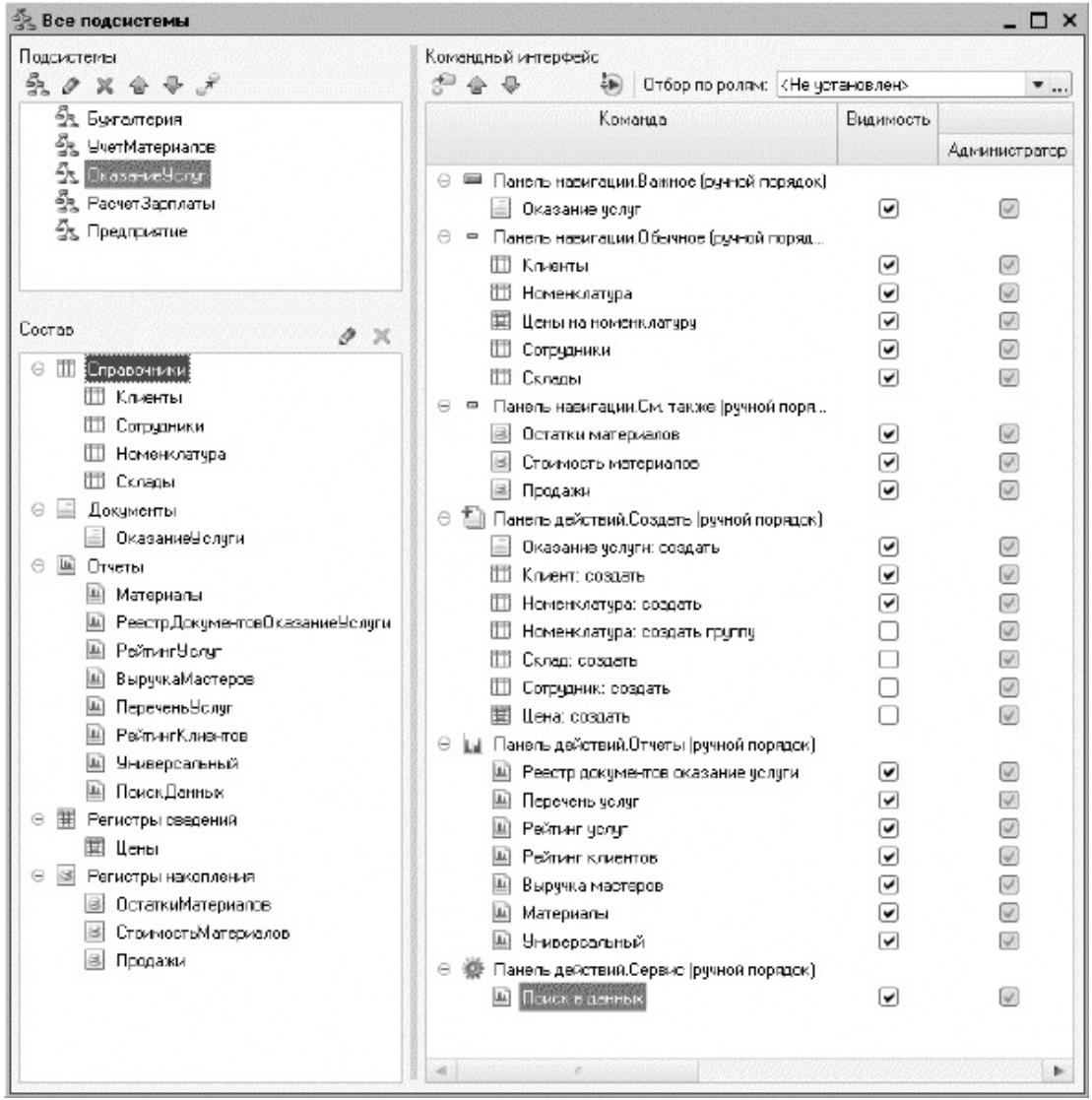

Все подсистемы
Подсистемы
Бухгалтерия
УчетМатериалов
ОказаниеУслуг
РасчетЗарплаты
Предприятие
Состав
Справочники
Клиенты
Сотрудники
Номенклатура
Склады
Документы
ОказаниеУслуги
Отчеты
Материалы
РеестрДокументовОказаниеУслуги
РейтингУслуг
ВыручкаМастеров
ПеречеьУслуг
РейтингКлиентов
Универсальный
ПоискДанных
Регистры сведений
Цены
Регистры накопления
ОстаткиМатериалов
СтоимостьМатериалов
Продажи
Командный интерфейс
Отбор по ролям:
<Не установлен>
Команда
Видимость
Администратор
Панель навигации.Важное (ручной порядок)
Оказание услуг
Панель навигации.Обычное (ручной поряд...
Клиенты
Номенклатура
Цены на номенклатуру
Сотрудники
Склады
Панель навигации.См. также (ручной поря...
Остатки материалов
Стоимость материалов
Продажи
Панель действий.Создать (ручной порядок)
Оказание услуги: создать
Клиент: создать
Номенклатура: создать
Номенклатура: создать группу
Склад: создать
Сотрудник: создать
Цена: создать
Панель действий.Отчеты (ручной порядок)
Реестр документов оказание услуги
Перечень услуг
Рейтинг услуг
Рейтинг клиентов
Выручка мастеров
Материалы
Универсальный
Панель действий.Сервис (ручной порядок)
Поиск в данных

Рис. 23.3. Командный интерфейс подсистемы «ОказаниеУслуг»

Подсистема *Бухгалтерия* будет иметь следующий командный интерфейс:

- Группа *Панель навигации.Важное*:
  - *Приходные накладные,*
  - *Оказание услуг.*
- Группа *Панель навигации.Обычное*:
  - *Клиенты,*
  - *Номенклатура,*
  - *Цены на Номенклатуру.*
- Группа *Панель навигации.См. также*:
  - *Остатки материалов,*
  - *Стоимость материалов,*
  - *Продажи,*
  - *Ввод начальных остатков номенклатуры,*
  - *Основной план счетов,*
  - *Виды субконто.*
- Группа *Панель действий.Создать* (только видимые команды):
  - *Ввод начальных остатков номенклатуры: создать.*
- Группа *Панель действий.Отчеты*:

    - *Оборотно-сальдовая ведомость,*
    - *Начисления сотрудникам,*
    - *Перечень услуг,*
    - *Рейтинг услуг,*
    - *Рейтинг клиентов,*
    - *Материалы,*
    - *Остатки материалов по свойствам.*
- Группа *Панель действий.Сервис*:
    - *Поиск в данных.*

Подсистема *РасчетЗарплаты* будет иметь следующий командный интерфейс:

- Группа *Панель навигации.Важное*:
    - *Начисления сотрудникам.*
- Группа *Панель навигации.Обычное*:
    - *Сотрудники,*
    - *Графики работы.*
- Группа *Панель навигации.См. также*:
    - *Начисления,*
    - *Виды расчетов,*
    - *Виды графиков работы.*

- Группа *Панель действий.Создать* (только видимые команды):
  - *Начисления сотрудникам: создать,*
  - *Сотрудник: создать.*
- Группа *Панель действий.Отчеты*:
  - *Начисления сотрудникам,*
  - *Выручка мастеров,*
  - *Диаграмма начислений.*
- Группа *Панель действий.Сервис*:
  - *Поиск в данных.*

Подсистема *Предприятие* будет иметь следующий командный интерфейс:

- Группа *Панель действий.Сервис*:
  - *Планировщик заданий,*
  - *Поиск в данных.*

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

Поскольку мы создали пользователей нашей конфигурации и присвоили им роли, то теперь перед запуском приложения нужно указать пользователя.

Выберем пользователя *Администратор*, пароль для пользователей указывать не нужно, так как мы его не задавали.

Посмотрим, как красиво выглядит теперь интерфейс приложения, например в разделах *Бухгалтерия* и *Расчет зарплаты* (рис. 23.4, 23.5).

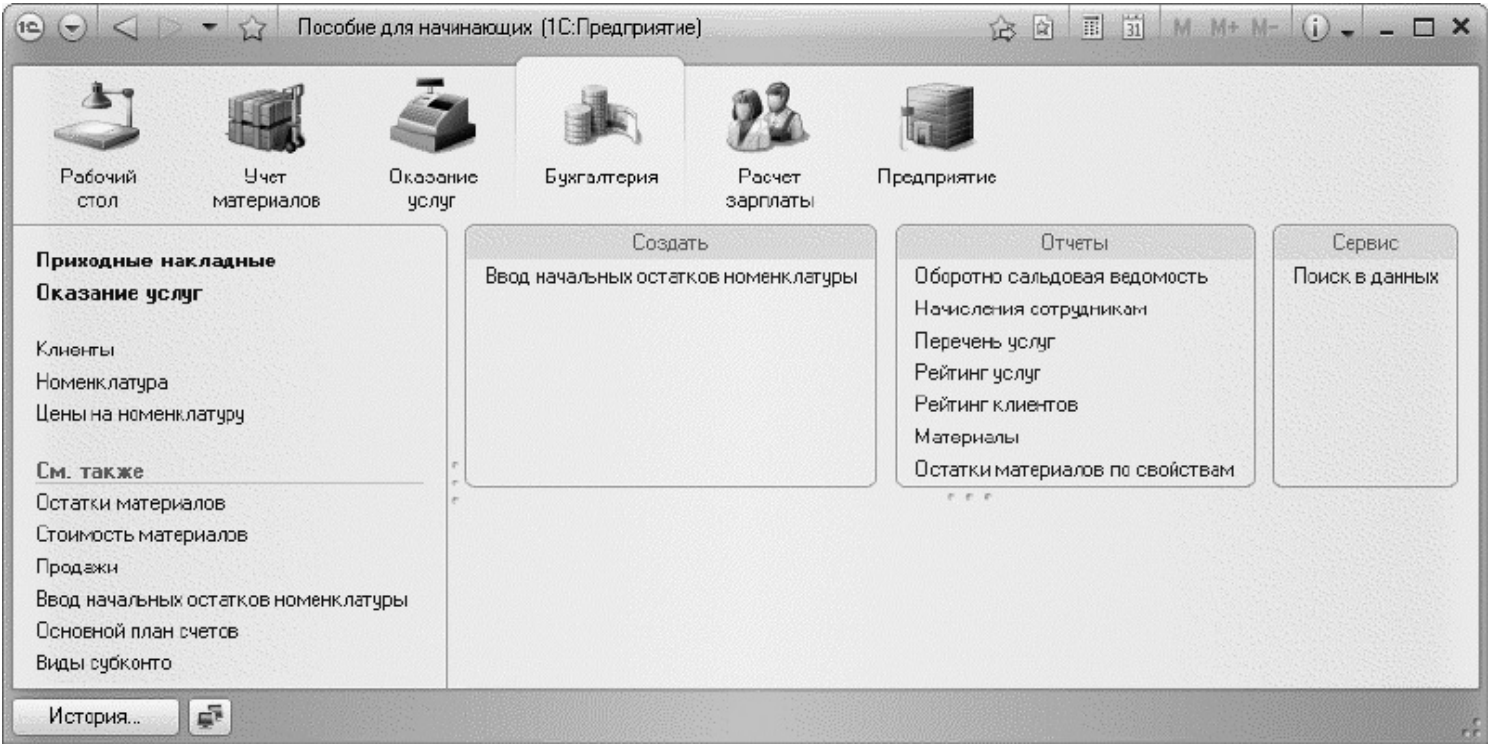


Рис. 23.4. Командный интерфейс раздела «Бухгалтерия»

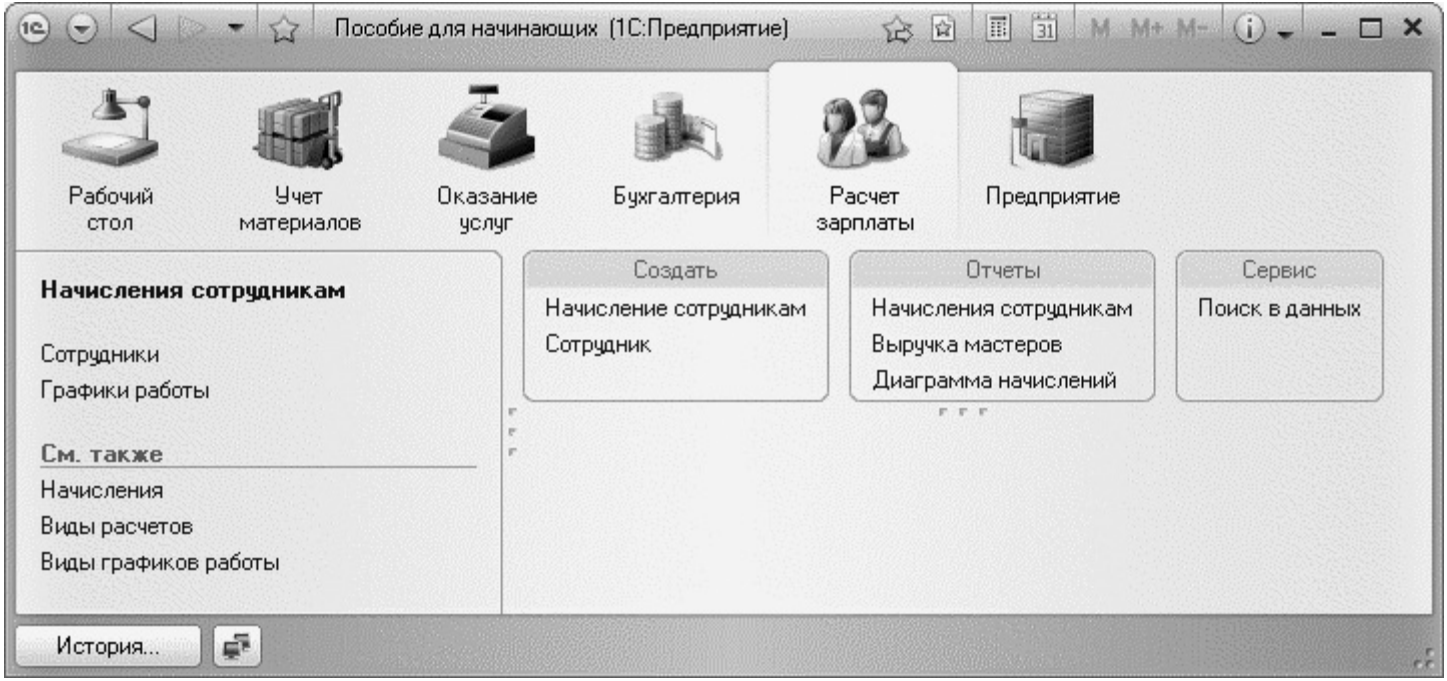


Рис. 23.5. Командный интерфейс раздела «Расчет зарплаты»

Как мы видим, все группы команд зрительно отделены друг от друга, а наиболее важные команды в панели навигации разделов выделены жирным шрифтом.

Итак, мы настроили командный интерфейс разделов, согласно нашим представлениям о работе прикладного решения.

Однако если у пользователя – другие предпочтения, то в режиме *1С:Предприятие* он может настраивать командный интерфейс по своему усмотрению, выполнив команду главного меню *Сервис > Настройка интерфейса > …*

Рабочий стол

Рабочий стол предназначен для размещения наиболее часто используемых пользователем документов, отчетов, справочников и т. п. Поэтому нужно поместить на рабочий стол пользователя те формы документов, отчетов и пр., работа с которыми входит в его ежедневные должностные обязанности.

Например, для кладовщика было бы удобно иметь под руками список номенклатуры и список приходных накладных, для менеджера – список клиентов и документов оказания услуг и т. д.

При запуске «1С:Предприятия» раздел *Рабочий стол* становится активным по умолчанию и нужные формы сразу открываются в рабочей области приложения (рис. 23.6).

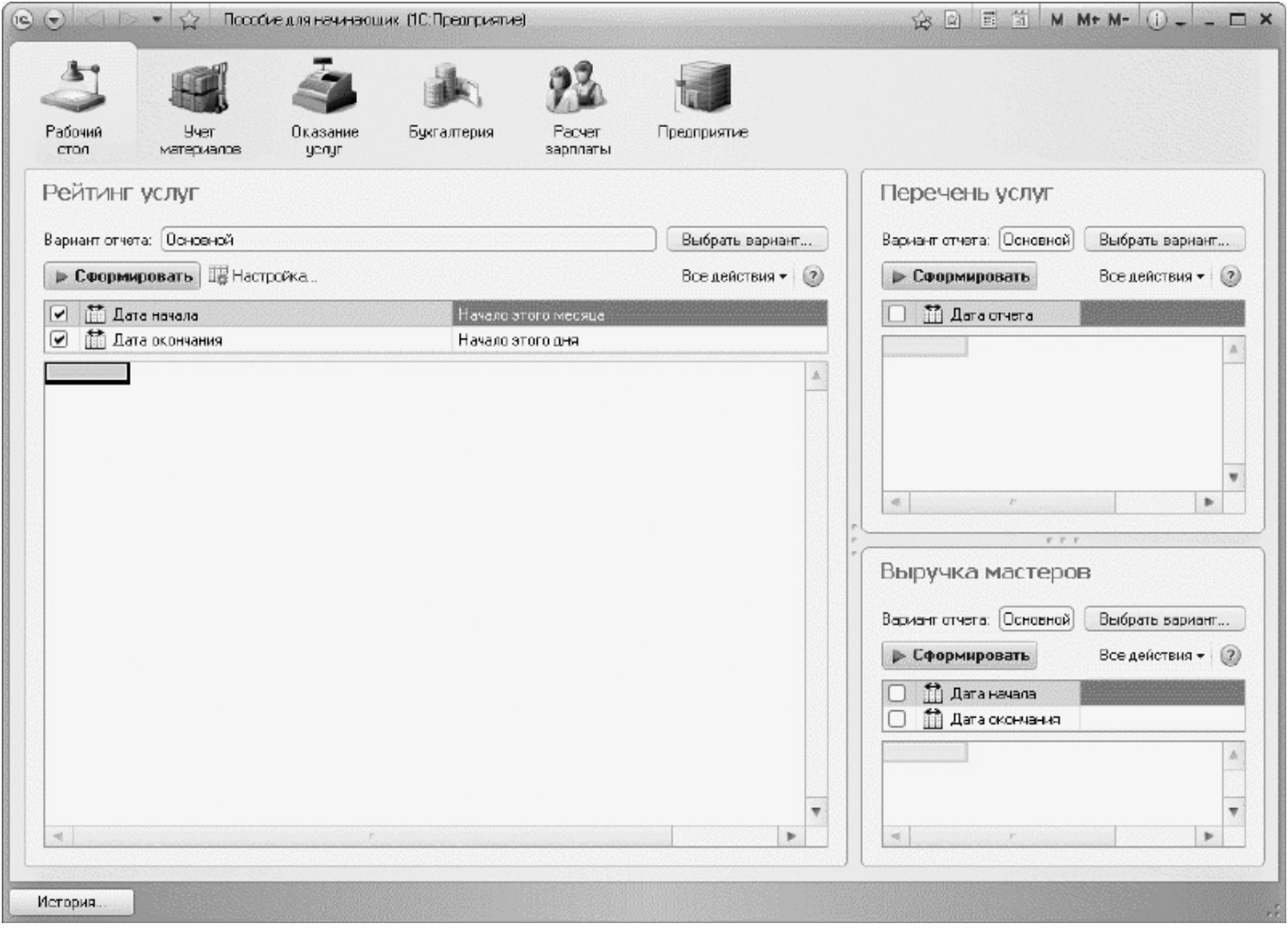


Рис. 23.6. Рабочий стол для пользователя с ролью «Директор»

Таким образом, пользователю не нужно ходить по разделам, выполнять команды в панели действий или в панели навигации, а можно сразу начинать работать, предварительно включив компьютер и запустив «1С:Предприятие 8».

Однако не стоит перегружать рабочий стол различными формами, иначе вместо удобства пользователь будет ощущать дискомфорт.

В режиме «Конфигуратор»

Итак, начнем настройку рабочего стола.

Выделим корень дерева объектов конфигурации *ПособиеДляНачинающих*, вызовем его контекстное меню и выберем пункт *Открыть рабочую область рабочего стола* (рис. 23.7).

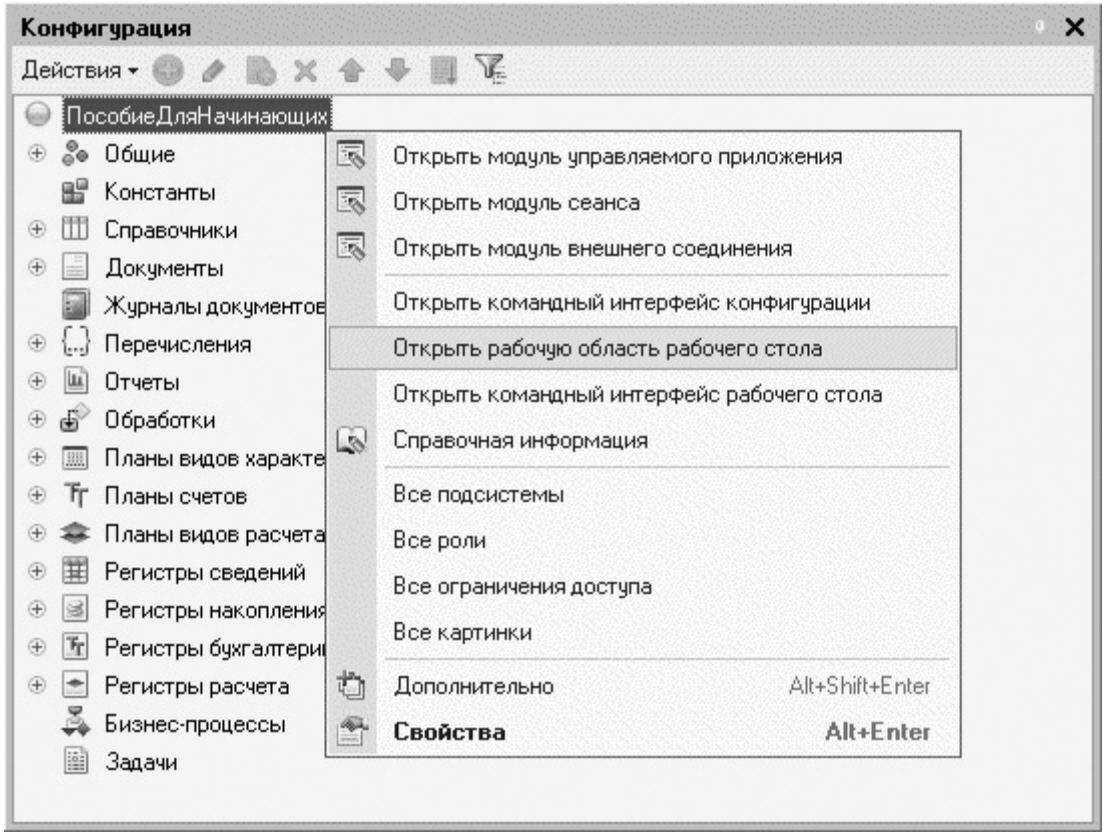


Рис. 23.7. Вызов диалога настройки рабочего стола

Откроется окно настройки рабочего стола (рис. 23.8). Сначала вверху окна выберем шаблон рабочего стола *Две колонки разной ширины (2:1)*.

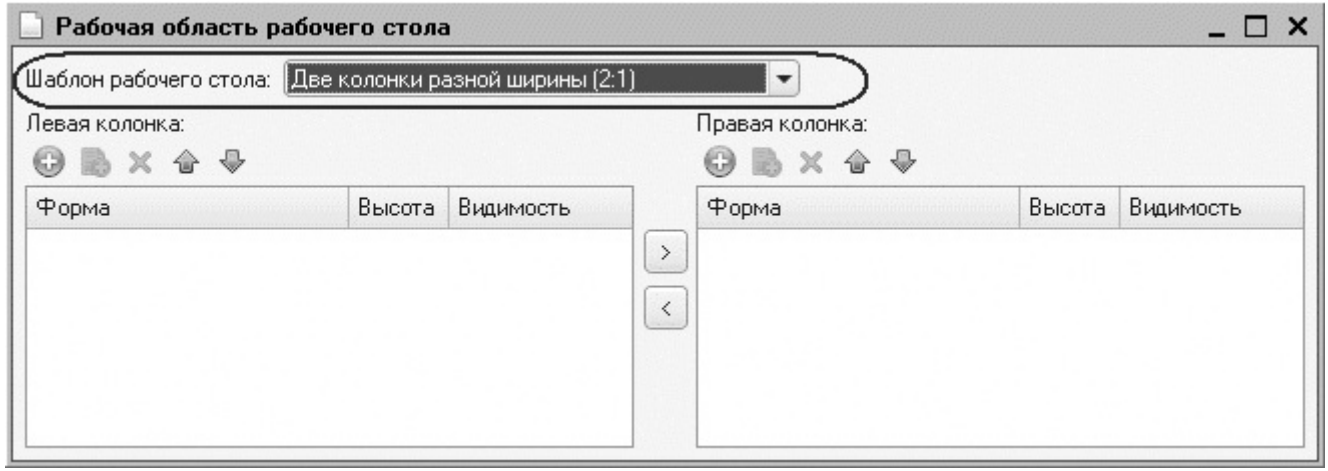


Рис. 23.8. Окно настройки рабочего стола

Это значит, что формы на рабочем столе будут располагаться в две колонки, при этом левая колонка будет в два раза шире правой.

Можно выбрать другой шаблон, при котором колонки будут одинаковой ширины, или будет всего одна колонка. Но кажется, что предпочтительнее первый вариант *(2:1)*, так как в этом случае взгляд пользователя сразу будет падать на наиболее приоритетные для работы формы, которые мы расположим в левой колонке.

Формы в каждой колонке будут располагаться друг под другом. Оптимально, если в левой колонке будет располагаться одна, максимум две формы, а в

правой – две-три формы для каждого пользователя.

Следует иметь в виду, что автоматически сгенерированные системой формы нельзя располагать на рабочем столе. Поэтому прежде чем добавлять форму на рабочий стол, нужно создать ее в явном виде в конфигурации. Чтобы не путаться, будем создавать нужные формы прямо по ходу.

Итак, начнем настройку рабочего стола для роли *Мастер*.

Наше ООО «На все руки мастер» – фирма по оказанию услуг, и мастера имеют к этому непосредственное отношение. Поэтому логично, если для мастеров в левой колонке рабочего стола будет располагаться список документов об оказании услуг, а в правой колонке – список приходных накладных и список клиентов.

Перечисленные формы списка отсутствуют в конфигурации, поэтому создадим формы списка для объектов конфигурации:

- справочник *Клиенты*,
- документ *ПриходнаяНакладная,*
- документ *ОказаниеУслуги*.

Теперь перейдем в окно настройки рабочего стола и над списком форм левой

колонки нажмем *Добавить*.

Выберем форму списка документа *ОказаниеУслуги* (рис. 23.9).

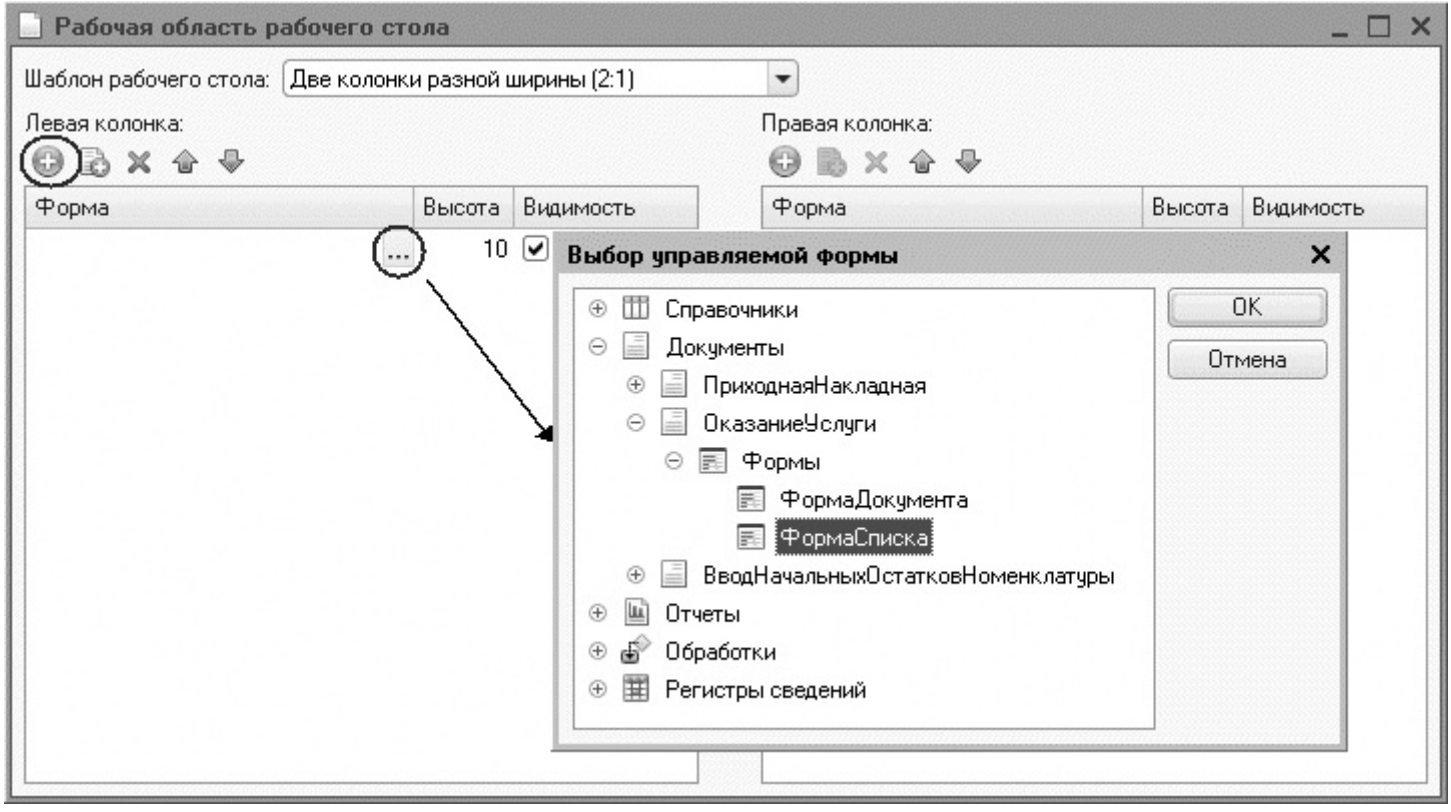


Рис. 23.9. Настройка рабочего стола для пользователя с ролью «Мастер»

Аналогичным образом в правую колонку добавим формы списка документа *ПриходнаяНакладная* и справочника *Клиенты*.

Теперь для каждой из трех форм нажмем ссылку в колонке *Видимость* и установим видимость этих форм только для роли *Мастер* (рис. 23.10).

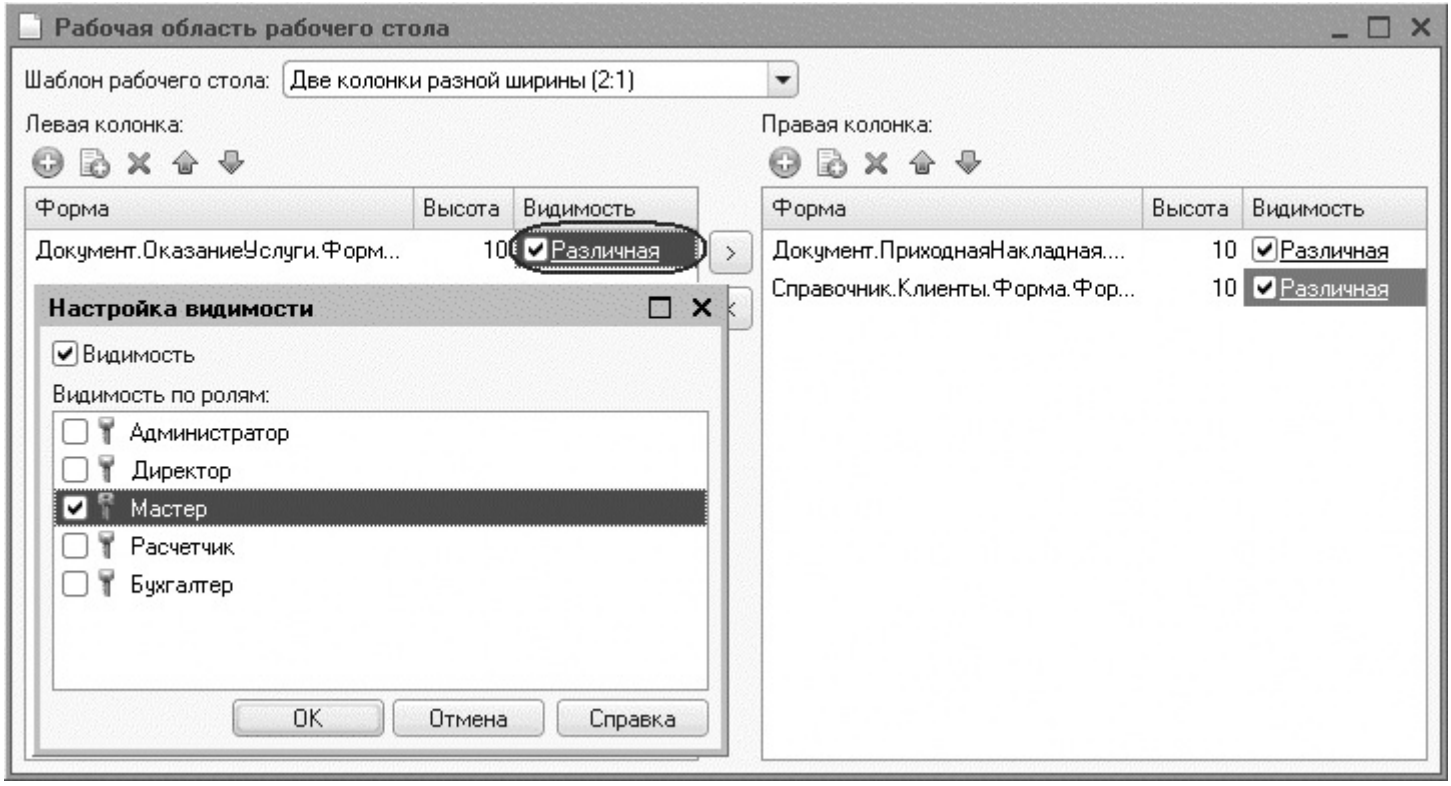


Рис. 23.10. Настройка рабочего стола для пользователя с ролью «Мастер»

Возможно, эти формы понадобятся на рабочем столе для пользователей с другими ролями, но мы сейчас, чтобы не запутаться, ограничимся только ролью *Мастер*.

Теперь настроим рабочий стол для роли *Бухгалтер*.

Предположим, бухгалтер наиболее часто будет пользоваться оборотно-сальдовой ведомостью и отчетом о начислениях сотрудникам. Расположим эти отчеты в левой колонке рабочего стола, а правую оставим пустой.

Создадим форму отчета для отчета *ОборотноСальдоваяВедомость*, а для отчета *НачисленияСотрудникам* форму отчета мы уже создали ранее при разработке нашей конфигурации. Затем перейдем в окно настройки рабочего стола и добавим эти формы в левую колонку и уставим видимость этих форм только для роли *Бухгалтер*. Аналогичным образом настроим рабочий стол для роли *Расчетчик*.

По роду деятельности расчетчик в основном пользуется документами и отчетами о начислениях сотрудникам. Расположим отчет о начислениях сотрудникам в левой колонке рабочего стола, а в правой – список документов о начислениях.

Для этого создадим форму списка документа *НачисленияСотрудникам*.

Теперь перейдем в окно настройки рабочего стола, добавим форму списка в правую колонку и установим видимость этой формы только для роли *Расчетчик*.

Отчет *НачисленияСотрудникам* мы уже добавили в левую колонку рабочего стола ранее, при настройке роли *Бухгалтер*. Поэтому осталось только установить видимость этого отчета также и для роли *Расчетчик* (рис. 23.11).

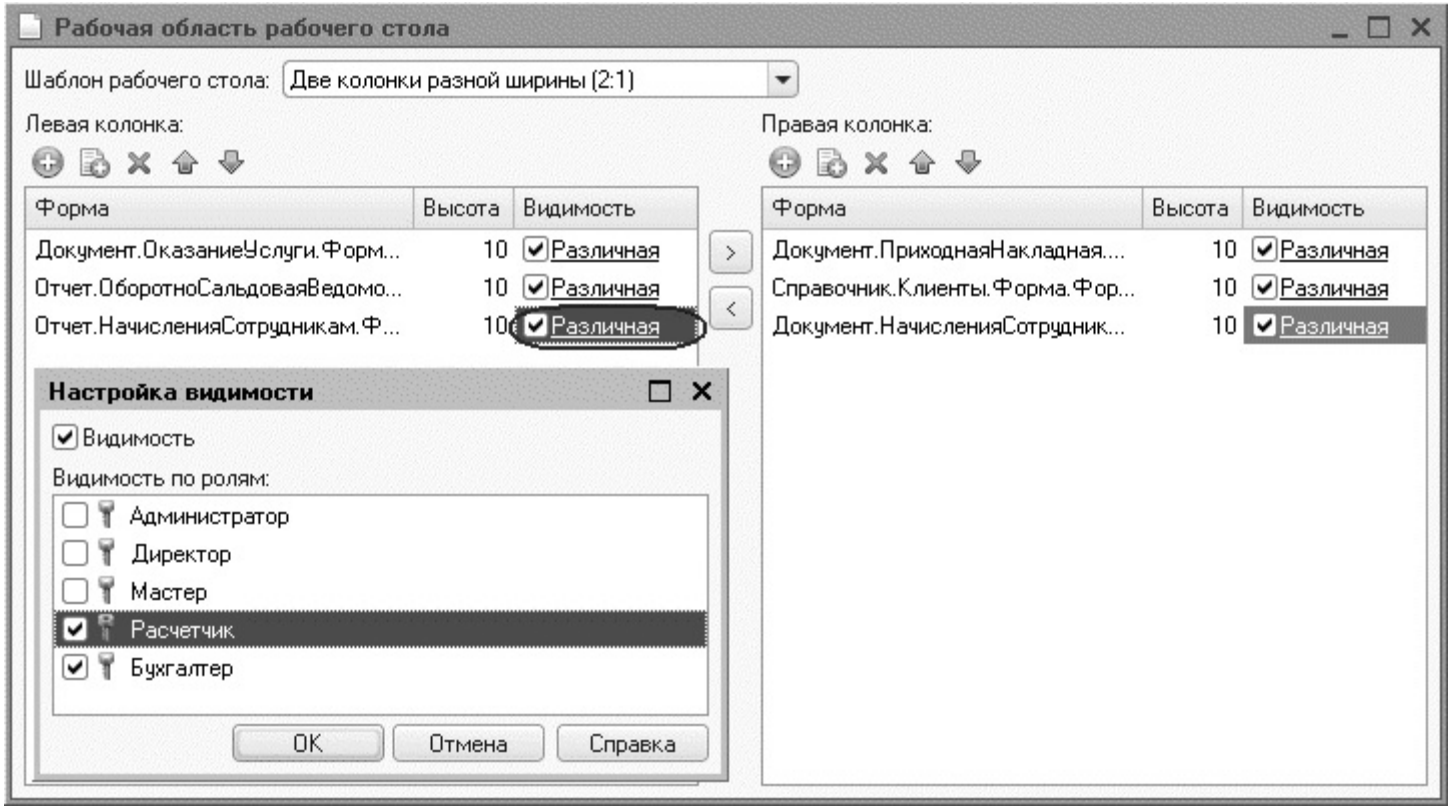


Рис. 23.11. Настройка рабочего стола для пользователя с ролью «Расчетчик»

Затем настроим рабочий стол для роли *Директор*.

Мы предполагаем, что эта роль будет назначена пользователю, осуществляющему руководящие функции. Ему не нужно вводить никаких

документов, да у него и нет на это прав.

Но ему понадобится регулярно просматривать отчеты о деятельности фирмы, чтобы принимать руководящие решения.

Поэтому расположим на его рабочем столе в левой колонке отчет *Рейтинг услуг*, а в правой – отчеты *Перечень услуг* и *Выручка мастеров*. Создадим формы отчета для этих отчетов. Затем перейдем в окно настройки рабочего стола, добавим эти формы в соответствующие колонки и установим видимость этих форм только для роли *Директор*.

И в заключение настроим рабочий стол для роли *Администратор*.

По роду деятельности администратор отвечает за состояние информационной базы «1С:Предприятия» и может иметь доступ ко всем объектам конфигурации. Но для администратора не должно быть проблемой найти и выполнить любую команду нашей конфигурации. Поэтому лучше узнать у администратора, что ему хочется иметь под рукой.

Для примера расположим на его рабочем столе в левой колонке отчет для поиска данных, а правую – оставим пустой.

Форму для отчета *ПоискДанных* мы уже создали ранее при разработке нашей

конфигурации. Теперь перейдем в окно настройки рабочего стола, добавим форму в левую колонку и установим видимость этой формы только для роли *Администратор*.

В результате окно настройки рабочего стола должно принять следующий вид (рис. 23.12).

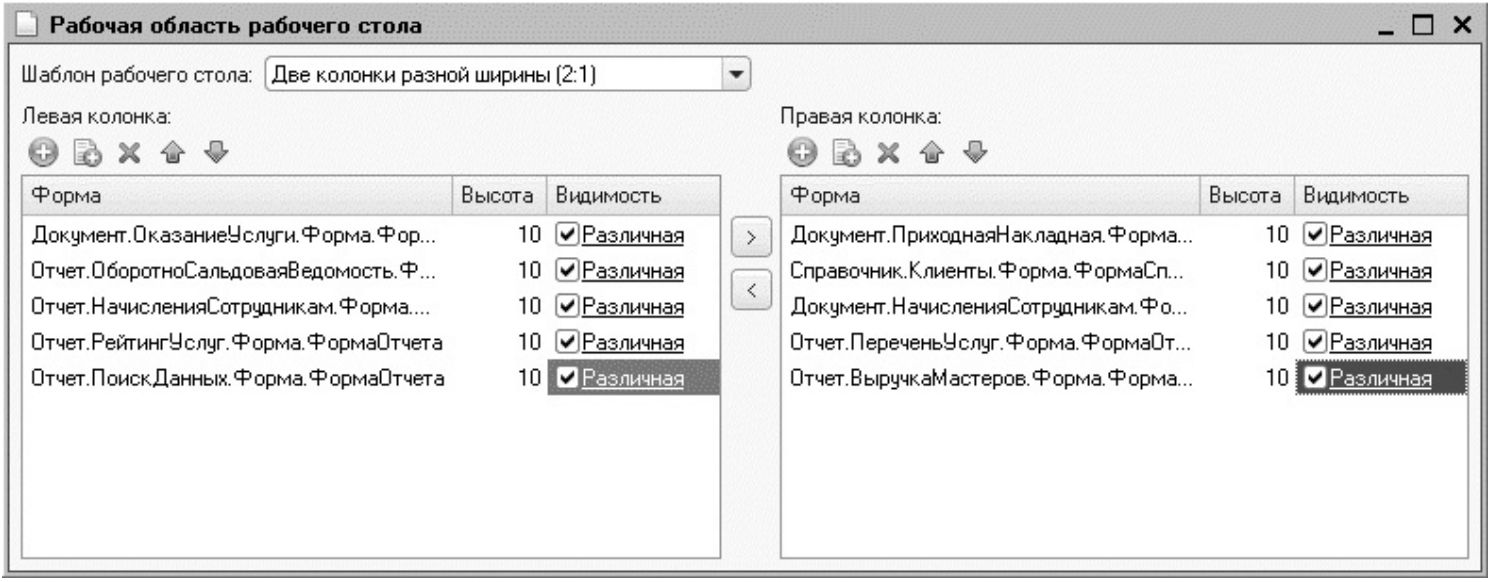


Рис. 23.12. Окно настройки рабочего стола

В режиме «1С:Предприятие»

Теперь, если мы зайдем в режиме *1С:Предприятие*, указав пользователя

Назарова (с ролями *Бухгалтер* и *Расчетчик*), мы увидим такой рабочий стол (рис. 23.13).

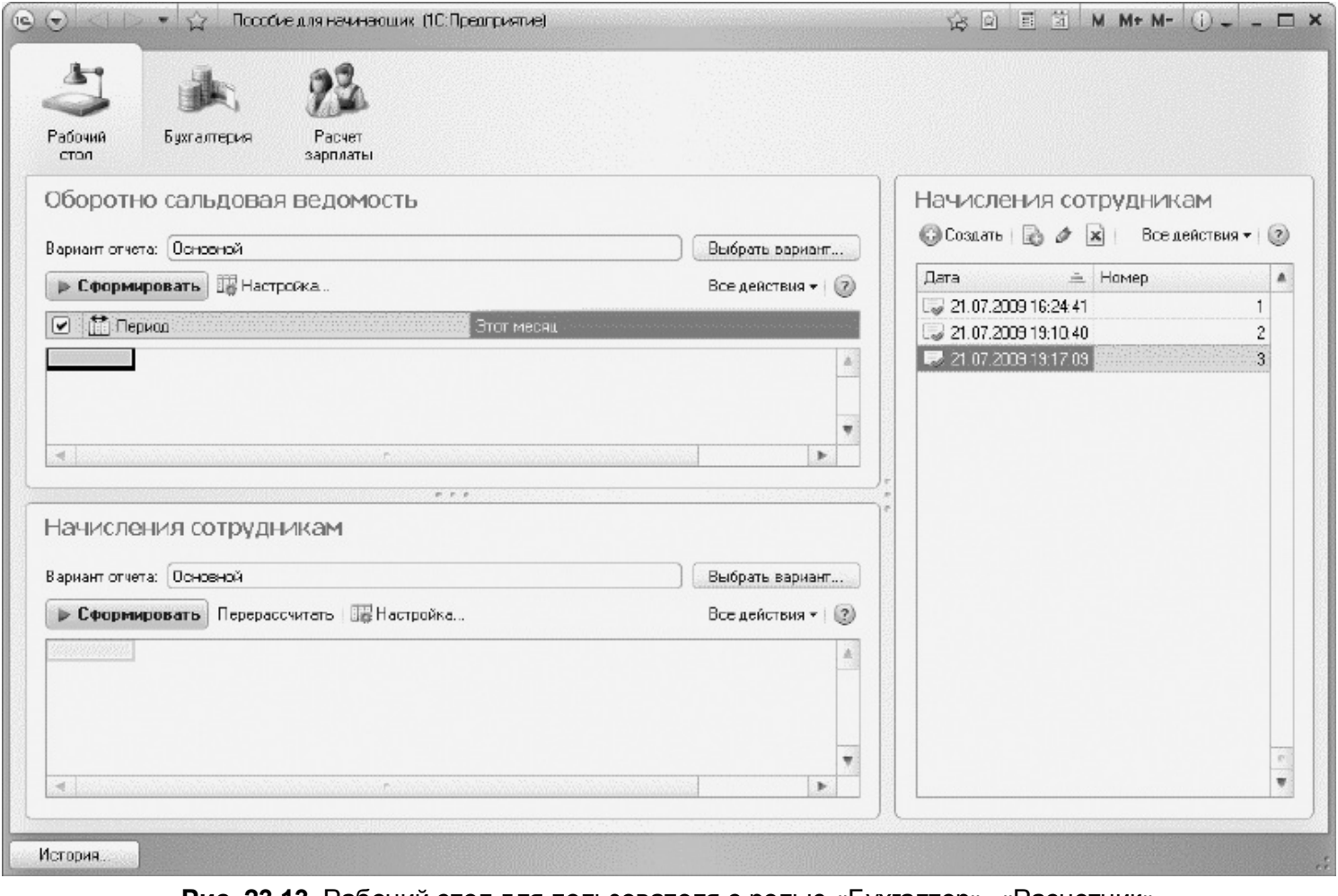


Рис. 23.13. Рабочий стол для пользователя с ролью «Бухгалтер», «Расчетчик»

ВНИМАНИЕ!

Чтобы зайти в режим 1С:Предприятия под различными пользователями, нужно выполнить обновление конфигурации и запустить «1С:Предприятие» еще раз, выбрав режим 1С:Предприятие (тонкий клиент) и нужного пользователя. Если вы, как обычно, запустите «1С:Предприятие» в режиме отладки, то есть из конфигуратора, то система будет считать, что вы заходите в программу под тем же пользователем, что и в конфигуратор, то есть Администратор.

Обратите внимание, что на рабочем столе находится также список документов о начислениях сотрудникам, так как пользователь *Назарова* имеет две роли – *Бухгалтер* и *Расчетчик*. А также этому пользователю доступны разделы *Бухгалтерия* и *Расчет зарплаты* в соответствии с правами, которые мы установили для этих ролей на предыдущем занятии.

А для пользователя *Гусаков* (с ролью *Мастер*) рабочий стол будет выглядеть следующим образом (рис. 23.14).

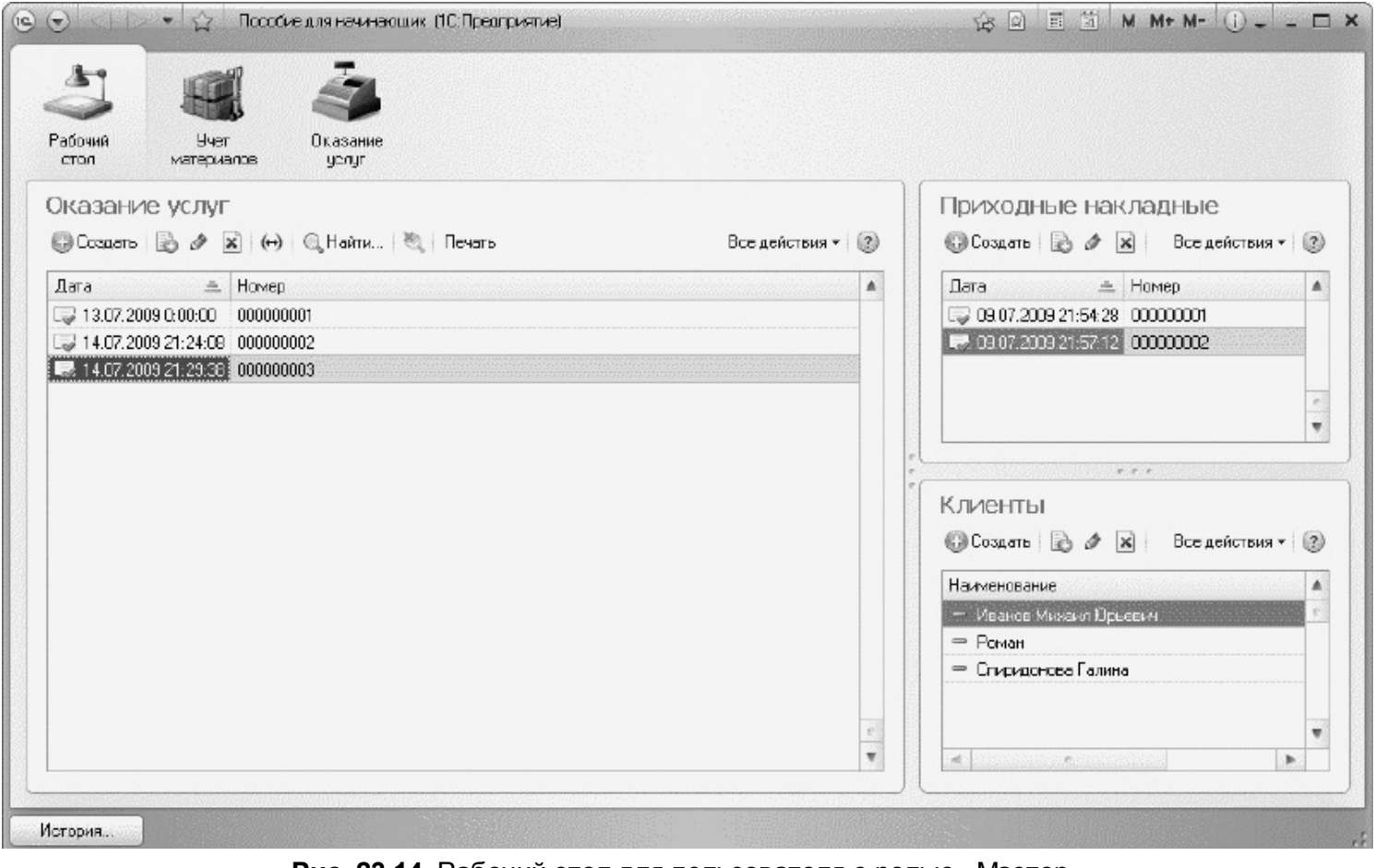


Рис. 23.14. Рабочий стол для пользователя с ролью «Мастер»

Этому пользователю доступны разделы *Учет материалов* и *Оказание услуг* в соответствии с правами, которые мы установили для этой роли на предыдущем

занятии.

Можно также настроить панель навигации и панель действий рабочего стола, выполнив команду *Открыть командный интерфейс рабочего стола* (см. рис. 24.7), но вряд ли в этом есть необходимость, так как все нужные команды есть в панели действий и в панели навигации разделов, доступных пользователю в соответствии с его ролью.

И наконец, в процессе работы «1С:Предприятия 8» пользователь может настраивать рабочий стол по своему усмотрению, выполнив команду главного меню *Сервис* > *Настройка интерфейса* > *Рабочий стол*.

Но следует иметь в виду, что пользователь может размещать на своем рабочем столе только те формы, которые разработчик создал в конфигурации.

Видимость команд по ролям

В режиме «Конфигуратор»

По мере создания новых объектов конфигурации мы видели, что система автоматически добавляла команды по умолчанию для работы с этими объектами (вызов списков справочников, документов, открытие отчетов, обработок и т. д.). Эти команды становились доступны для всех пользователей

в тех подсистемах, к которым относится данный объект.

Используя команду *Общие > Подсистемы > Все подсистемы,* мы не раз редактировали командный интерфейс подсистем – меняли порядок, включали видимость команд и т. д. После появления ролей в конфигурации появилась возможность настройки видимости этих команд по ролям.

Ролевое редактирование видимости команд по умолчанию – это средство, позволяющее настроить начальную насыщенность глобального командного интерфейса в первую очередь для пользователей с широкими правами доступа.

Действительно, если пользователю, например, закрыт доступ в подсистему *Бухгалтерия*, то он не увидит и всех команд этой подсистемы. Или если пользователь не имеет права создавать какие-либо документы, то он не увидит и соответствующей команды.

Но для администратора системы, с его широкими правами доступа, число команд оказалось слишком большим. Предположим, что командой открытия регистров накопления из панели навигации разделов он будет пользоваться крайне редко.

Для того чтобы облегчить его командный интерфейс, опять перейдем к

редактированию глобального командного интерфейса (*Общие* > *Подсистемы* > *Все подсистемы*) и для соответствующих команд каждой подсистемы снимем видимость для роли *Администратор* (рис. 23.15).

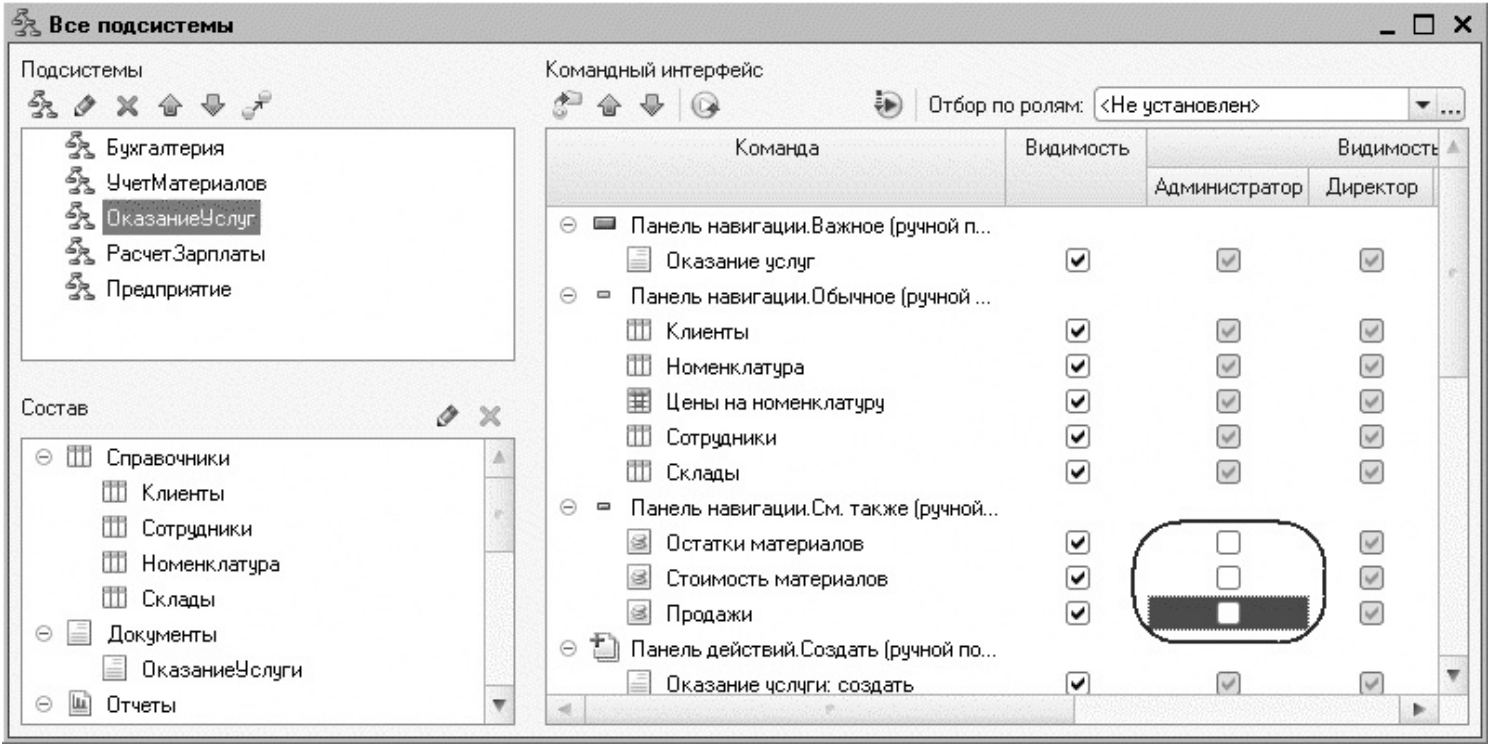


Рис. 23.15. Настройка видимости команд подсистем для роли «Администратор» в интерфейсе подсистем

Также откроем формы документов *ОказаниеУслуги* и *ПриходнаяНакладная* и на закладке *Командный интерфейс* в панели навигации снимем видимость у

команд перехода к регистру бухгалтерии *Управленческий* для роли *Администратор* (рис. 23.16).

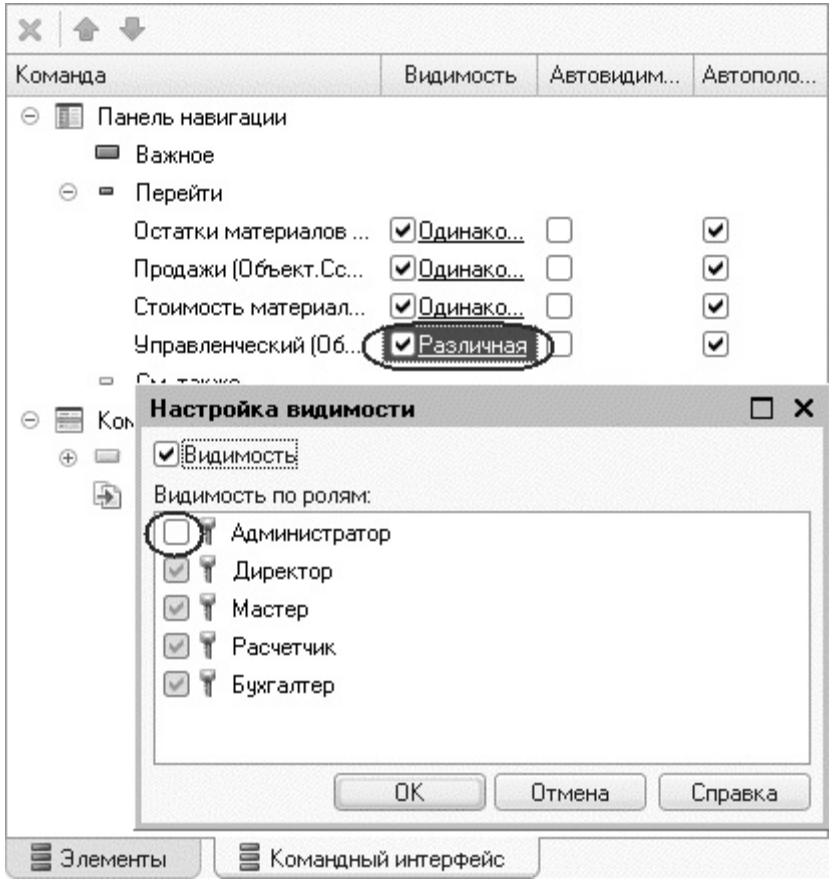


Рис. 23.16. Настройка видимости команд подсистем для роли «Администратор» в форме документа

В режиме «1С:Предприятие»

Запустите «1С:Предприятие» в режиме отладки и убедитесь, что команды открытия регистров накопления не видны для администратора, хотя он имеет к ним доступ. Например, через главное меню: *Все функции – Регистры накопления*.

В реальной конфигурации, конечно, роли пользователей и наполнение их рабочих столов будут другими. Это зависит от специфики работы предприятия и пожеланий заказчика.

На последних двух занятиях мы показали только принцип организации интерфейса прикладного решения по ролям и ограничения доступа к отдельным командам и разделам в целом в зависимости от прав пользователя.

Контрольные вопросы

- *Что такое рабочий стол.*
- *Как настроить рабочий стол для различных пользователей.*
- *Как настроить видимость команд по ролям.*

Занятие 24 (6:10). Обмен данными

Продолжительность

Ориентировочная продолжительность занятия – 6 часов 10 минут.

На этом занятии мы познакомимся с механизмами обмена данными, которые содержит система «1С:Предприятие 8», и добавим в нашу конфигурацию возможность обмена данными с удаленными филиалами и отделениями.

Общие сведения об обмене данными

Механизмы обмена данными «1С:Предприятия 8» позволяют организовывать обмен информацией, хранимой в базе данных, с другими программными системами.

В качестве таких систем могут выступать как другие информационные базы «1С:Предприятия 8» (имеющие аналогичную или отличающуюся конфигурацию), так и программные системы, не основанные на «1С:Предприятии 8».

Такая гибкость обмена данными достигается за счет того, что средства обмена данными «1С:Предприятия 8» могут использоваться в различных комбинациях. Кроме этого, формат обмена данными основан на языке XML, являющимся на сегодняшний день общепринятым средством представления данных.

К механизмам обмена данными могут быть отнесены:

- *Планы обмена*,
- *XML-сериализация,*
- *Средства чтения и записи документов XML*.

В общем случае схема взаимодействия этих трех составляющих может быть представлена следующим образом (рис. 24.1).

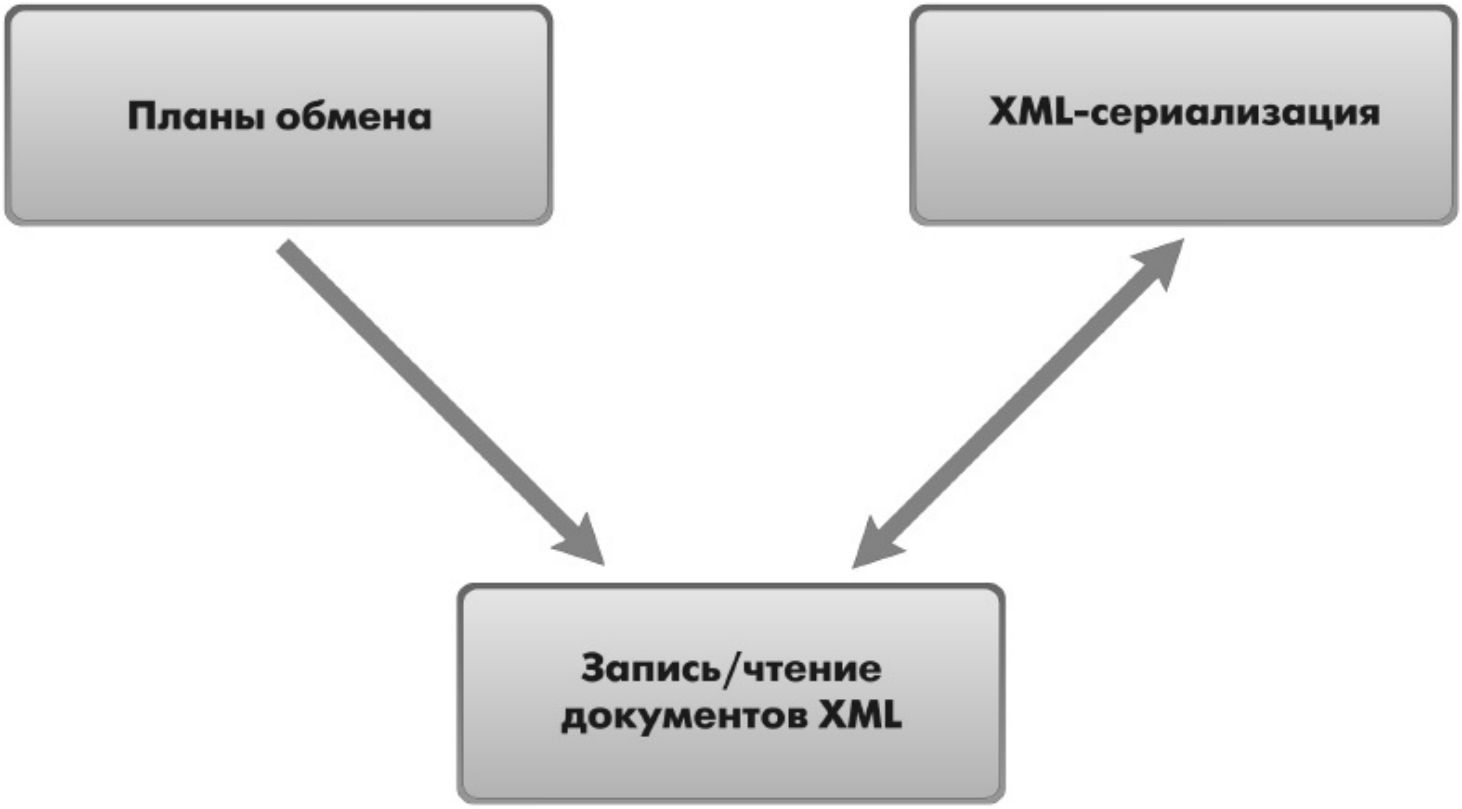


Рис. 24.1. Схема взаимодействия механизмов обмена данными

При помощи *планов обмена* мы получаем информацию о том, какие элементы данных были изменены и в какой узел обмена их необходимо передать.

Это возможно благодаря тому, что планы обмена содержат механизм

регистрации изменений. Информация об измененных данных переносится с помощью сообщений, инфраструктура которых также поддерживается планами обмена.

XML-сериализация позволяет преобразовать объект «1С:Предприятия 8» в последовательность данных, представленных в формате XML. Кроме этого, XML-сериализация выполняет и обратное преобразование – преобразует последовательность данных формата XML в объект «1С:Предприятия 8», при условии что имеется соответствующий тип «1С:Предприятия 8».

Запись и чтение документов XML обеспечивают запись/чтение документов формата XML из встроенного языка.

При реализации алгоритма обмена данными перечисленные механизмы могут быть использованы как все вместе, так и в различной комбинации. В каждом конкретном случае разработчик решает эту задачу самостоятельно.

В примере, приведенном в этой книге, мы используем все эти три механизма. Поэтому прежде чем перейти непосредственно к написанию кода, познакомимся с каждым из них подробнее.

Что такое план обмена

Для того чтобы существовала возможность обмена какими-либо данными с кем-либо, необходимо некоторым образом идентифицировать тех, с кем мы будем обмениваться, и для каждого из них описать перечень обмена. Обе эти задачи позволяет решать объект конфигурации *План обмена*.

Подобно тому, как элементами данных справочника являются элементы справочника, элементами данных плана обмена являются узлы плана обмена.

Каждый узел идентифицирует участника обмена по данному плану обмена. Кроме этого, в каждом плане обмена всегда существует один предопределенный узел, идентифицирующий данную информационную базу.

В одной конфигурации может существовать несколько планов обмена. Каждый план обмена определяет набор данных, которым будет производиться обмен в рамках данного плана, и сам механизм этого обмена.

Наличие нескольких планов обмена может потребоваться, если с разными узлами ведется обмен разным составом данных, или когда схема организации обмена с одними узлами отличается от схемы организации обмена с другими узлами.

Узнай больше!

О структуре объектов встроенного языка, предназначенных для работы с планом обмена, можно прочитать в разделе «Краткий справочник разработчика. Планы обмена»*.*

В обмене данными могут участвовать:

- объекты базы данных: элементы справочников, документы и т. д.,
- необъектные данные: наборы записей регистров, последовательностей, константы,
- специальный объект встроенного языка – *УдалениеОбъекта.*

Для упрощения изложения в дальнейшем будем называть эти элементы информационных структур *объектами обмена*.

Разработчик имеет возможность определить состав каждого плана обмена, указав объекты конфигурации, данные которых должны участвовать в обмене по данному плану.

При описании состава данных плана обмена разработчик имеет возможность указать для каждого типа объектов признак *Авторегистрация*. Этот признак определяет, каким образом план обмена будет отслеживать изменения данных.

Возможность отслеживать изменения данных реализована в плане обмена за

счет использования *механизма регистрации изменений*.

Работа этого механизма базируется на том, что каждый из объектов обмена имеет свойство *ОбменДанными*, с помощью которого можно указать, для каких узлов необходимо производить регистрацию изменений этого объекта. Любые изменения объекта обмена сводятся в конечном итоге к записи или удалению объекта обмена. Механизм регистрации изменений анализирует события записи и удаления объектов обмена и на основании параметров обмена данными, содержащихся в каждом из объектов обмена, формирует записи регистрации изменений. Следует отметить, что свойство *ОбменДанными* не хранится в базе данных, а используется только во время записи объекта обмена.

Так вот, признак *Авторегистрация*, устанавливаемый при указании состава данных плана обмена, позволяет указать, что параметры обмена данными будут формироваться каждый раз самим механизмом регистрации изменений на основании информации, содержащейся в плане обмена.

После автоматического заполнения параметров обмена разработчик все же имеет возможность внести изменения в сформированные таким образом параметры. Для этого следует использовать обработчики событий объектов, участвующих в обмене, – *ПередЗаписью* и *ПередУдалением*, в которых можно модифицировать список узлов-получателей (то есть тех узлов, для которых регистрируются изменения).

Кроме этого, существует возможность отключить авторегистрацию изменений, и тогда параметры обмена данными нужно будет формировать полностью средствами встроенного языка. Гипотетически это можно делать в любом фрагменте кода, но для того, чтобы конфигурация была легко читаема, рекомендуется использовать все те же обработчики событий *ПередЗаписью* и *ПередУдалением*. В этом случае код формирования параметров обмена данными будет сосредоточен в логически понятных точках, а не разбросан по всей конфигурации.

Итак, как мы теперь знаем, при записи и удалении объектов обмена план обмена формирует *записи регистрации изменений*. Записи регистрации изменений хранятся в таблицах регистрации изменений, причем для каждого объекта обмена ведется своя таблица.

При изменении объекта обмена в таблице регистрации изменений создается столько записей, сколько узлов-получателей указано в параметрах обмена данными у объекта обмена. Каждая запись при этом будет хранить ссылку на свой узел-получатель. Таблицы регистрации изменений создаются лишь в том случае, если соответствующий объект метаданных указан в составе хотя бы одного плана обмена.

Кроме ссылки на узел обмена, для которого регистрируются изменения, каждая запись таблицы регистрации изменений хранит также номер *сообщения*, в

котором изменение было передано в первый раз в этот узел. До тех пор, пока сообщение не будет передано в первый раз, это поле хранит *Null*.

Сообщение с точки зрения плана обмена – это единица обмена информацией. Поэтому одной из важнейших составляющих плана обмена, помимо службы регистрации изменений, является *инфраструктура сообщений*.

Поскольку сообщения передаются в рамках плана обмена от одного узла к другому, каждое сообщение точно ассоциировано с планом обмена, имеет уникальный номер и одного отправителя и получателя. За нумерацию сообщений отвечает инфраструктура сообщений. Благодаря этому записи регистрации изменений и имеют возможность хранить номера сообщений, в которых эти изменения были переданы первый раз.

Инфраструктура сообщений позволяет также получать подтверждения от узла-получателя о приеме сообщений. Такое подтверждение содержится в каждом сообщении, приходящем от узла-получателя в виде номера последнего принятого сообщения.

Впоследствии, проанализировав номер последнего принятого сообщения и номера сообщений, содержащиеся в записях регистрации изменений, разработчик может удалить записи регистрации изменений, прием которых подтвержден получателем.

XML-сериализация

Термином *XML-сериализация* обозначается механизм, позволяющий представить объект «1С:Предприятия» в виде последовательности данных в формате XML. Кроме этого, XML-сериализация позволяет выполнить и обратное преобразование – представить последовательность данных формата XML в виде объекта «1С:Предприятия 8», если существует подходящий тип данных.

Дело в том, что объект обмена, являющийся в системе «1С:Предприятие 8» единым целым, на самом деле представляет собой совокупность данных различных типов, определенным образом связанных между собой.

Например, элемент справочника кроме кода и наименования может содержать некоторое количество реквизитов различного типа и некоторое количество табличных частей, содержащих, в свою очередь, некоторое количество реквизитов различного типа.

В результате XML-сериализации вся эта совокупность данных представляется в виде последовательности соответствующих данных формата XML.

Вследствие обратного преобразования производится «сборка» объекта при условии, что существует подходящий тип данных «1С:Предприятия 8».

Запись/чтение документов XML

В отличие от XML-сериализации, механизмы *записи/чтения документов XML* позволяют работать с данными формата XML на базовом уровне, без привязки к объектам «1С:Предприятия 8».

В частности, они позволяют открывать файлы XML для чтения, читать данные из файлов, создавать новые файлы XML и записывать в них данные.

Универсальный механизм обмена данными

Итак, наше ООО «На все руки мастер» открыло свой филиал в городе N и установило в нем такую же конфигурацию для учета работы филиала.

В результате возникла необходимость наладить обмен данными между этими двумя базами таким образом, чтобы каждая из них отражала полную информацию о материалах и услугах, в то время как бухгалтерский учет и расчет зарплаты велись бы в каждой базе отдельно.

Для этого мы создадим план обмена, опишем состав данных, которые будут включены в обмен, и сделаем несколько процедур, позволяющих нам формировать на жестком диске файлы обмена и соответственно загружать полученные файлы обмена с жесткого диска.

Для упрощения примера мы не будем программировать какой-либо автоматический обмен файлами между двумя базами и запуск процедуры обмена будем осуществлять вручную.

Прежде чем мы начнем непосредственно программировать алгоритм обмена, следует сказать о некоторых доработках, которые нам придется предварительно внести в нашу базу.

Эти доработки будут связаны с тем, что до сих пор мы работали только в одной базе и использовали уникальность номеров кодов справочников и номеров документов.

Теперь, когда создание новых элементов справочников и новых документов будет происходить в двух базах одновременно и независимо друг от друга, нам снова необходимо обеспечить уникальность номеров кодов элементов справочников и номеров документов теперь уже «в пространстве» двух баз.

Если мы этого не сделаем, то не исключено, что в каждой из баз будут созданы, например, новые документы с одинаковыми номерами, и при обмене данными возникнет конфликт, поскольку система будет пытаться записать в базу документ с номером, который уже используется другим документом.

Для исключения подобных ситуаций в каждой базе к номерам документов и

кодам справочников мы будем добавлять уникальный префикс, однозначно идентифицирующий базу данных.

Тогда даже если номера новых документов в двух базах совпадут, они все равно будут отличаться префиксом и конфликта не возникнет.

Для хранения префикса номеров мы используем объект конфигурации *Константа*.

Константа для обмена данных

В режиме «Конфигуратор»
ВНИМАНИЕ!

> *Поскольку на предыдущем занятии мы создали в конфигурации список пользователей, теперь при входе в конфигуратор система спросит имя пользователя. Нужно указать пользователя – Администратор, так как он имеет полный доступ ко всем объектам конфигурации. Пароль задавать не нужно, так как мы его не устанавливали.*

Объект конфигурации *Константа* предназначен для создания в базе данных таблиц, в которых будет храниться информация, не изменяющаяся во времени или изменяющаяся очень редко.

Каждый объект конфигурации *Константа* описывает таблицу для хранения одного значения.

Теперь приступим к созданию константы, в которой мы будем хранить значение префикса номеров.

Откроем конфигуратор и добавим новый объект конфигурации *Константа* с именем *ПрефиксНумерации*.

Определим тип значения константы – *Строка* с фиксированной длиной *2* символа.

Доработка объектов конфигурации, участвующих в обмене

В режиме «Конфигуратор»

Первое, что нам следует сделать, – внести изменения в модули всех объектов, участвующих в обмене (в нашем случае это будут документы, справочники и планы видов характеристик).

Эти изменения будут заключаться в том, что теперь при формировании номера документа и кода справочника или плана видов характеристик будет использоваться значение константы *ПрефиксНумерации* для обеспечения уникальности номеров и кодов в каждой из наших баз.

Функцию формирования префикса номера мы вынесем в общий модуль, поскольку не исключена возможность того, что в будущем алгоритм формирования префикса документов может быть изменен.

Добавим общий модуль *Обмен*.

В модуль поместим следующую функцию (листинг 24.1).

Листинг 24.1. Функция формирования префикса номера

```
Функция ПолучитьПрефиксНомера() Экспорт

  Возврат Константы.ПрефиксНумерации.Получить();

КонецФункции
```

Как вы видите, эта функция просто возвращает значение константы *ПрефиксНумерации*.

Теперь доработаем справочник *Клиенты*.

Выделим этот объект в дереве объектов конфигурации, вызовем контекстное меню и откроем модуль объекта.

Добавим в него обработчик события *ПриУстановкеНовогоКода* (листинг 24.2).

Листинг 24.2. Обработчик события «ПриУстановкеНовогоКода»

```
Процедура ПриУстановкеНовогоКода(СтандартнаяОбработка, Префикс)

  Префикс = Обмен.ПолучитьПрефиксНомера();

КонецПроцедуры
```

Событие *ПриУстановкеНовогоКода* возникает в момент, когда выполняется установка нового кода элемента справочника. Обратите внимание, что мы пишем этот код не в модуле формы, а в модуле объекта, поскольку это событие возникает не для формы, а для объекта в целом.

Вторым параметром вызова обработчика передается префикс, который будет заполнен в данной процедуре и использован системой для генерации кода.

В обработчике события мы вызываем функцию общего модуля. Поскольку модуль неглобальный, то обращаемся к ней по имени модуля и имени функции (*Обмен.ПолучитьПрефиксНомера*). В этой процедуре мы устанавливаем префикс равным значению константы *ПрефиксНумерации*.

Такие же обработчики нужно будет добавить во все справочники и планы видов

характеристик, участвующие в обмене.

В нашем случае это:

- справочники:
- *Сотрудники,*
- *Склады,*
- *Номенклатура,*
- *ВариантыНоменклатуры,*
- *ДополнительныеСвойстваНоменклатуры,*

- план видов характеристик: *СвойстваНоменклатуры*.

После этого у всех этих объектов и у справочника *Клиенты* нужно (в палитре свойств объекта конфигурации) будет увеличить длину кода до *11* символов.

Теперь займемся доработкой документов.

В модуль документа *ПриходнаяНакладная* добавим обработчик события *ПриУстановкеНовогоНомера* (листинг 24.3).

Листинг 24.3. Обработчик события «ПриУстановкеНовогоНомера»

```
Процедура ПриУстановкеНовогоНомера(СтандартнаяОбработка, Префикс)

  Префикс = Обмен.ПолучитьПрефиксНомера();

КонецПроцедуры
```

Такие же обработчики нужно будет добавить во все документы, участвующие в обмене.

В нашем случае это единственный документ – *ОказаниеУслуги.*

После этого для обоих документов нужно (в палитре свойств объекта конфигурации) увеличить длину номера до *11* символов.

На этом подготовительная работа с существующими объектами конфигурации завершена, и мы можем перейти к созданию процедур обмена данными.

Добавление плана обмена

В режиме «Конфигуратор»

Теперь займемся созданием центра любого алгоритма обмена данными, вокруг которого группируются прочие механизмы, – плана обмена.

Раскроем ветвь *Общие* дерева объектов конфигурации и добавим новый объект конфигурации *ПланОбмена* с именем *Филиалы*, представление объекта – *Филиал*.

На закладке *Данные* создадим реквизит плана обмена *Главный*, имеющий тип *Булево* (рис. 24.2).

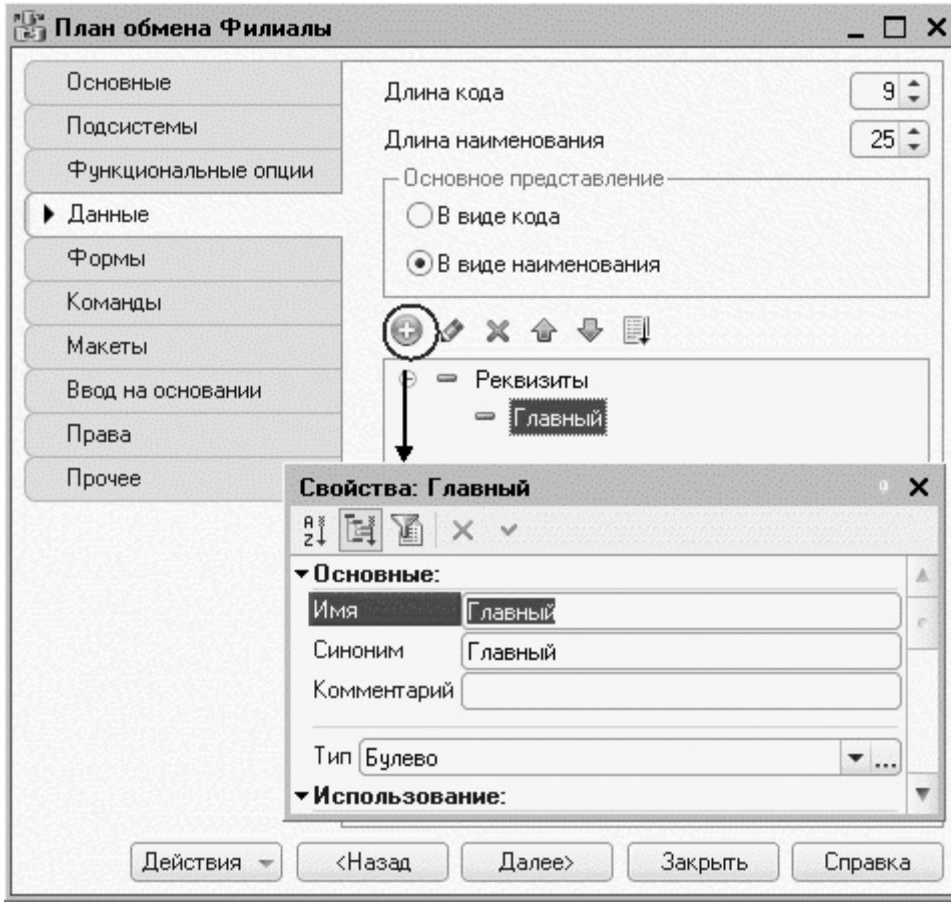


Рис. 24.2. Реквизит плана обмена

Этот реквизит понадобится нам для того, чтобы разрешать *коллизии* при обмене данными. Под коллизией понимается ситуация, когда один и тот же

объект обмена данными был изменен одновременно в двух узлах.

В этом случае мы будем анализировать значение реквизита *Главный* и принимать изменения только в том случае, если они сделаны в главном узле. В случае коллизии изменения, произведенные не в главном узле, мы будем отвергать.

Теперь определим состав объектов, участвующих в обмене. Для этого на закладке *Основные* нажмем кнопку *Состав*. Включим в обмен все объекты, не относящиеся к ведению бухгалтерии и расчету зарплаты.

Обратите внимание, что константа *ПрефиксНумерации* не участвует в обмене, поскольку ее значение должно быть уникальным для каждой базы, участвующей в обмене.

Состав данных обмена должен выглядеть следующим образом (рис. 24.3).

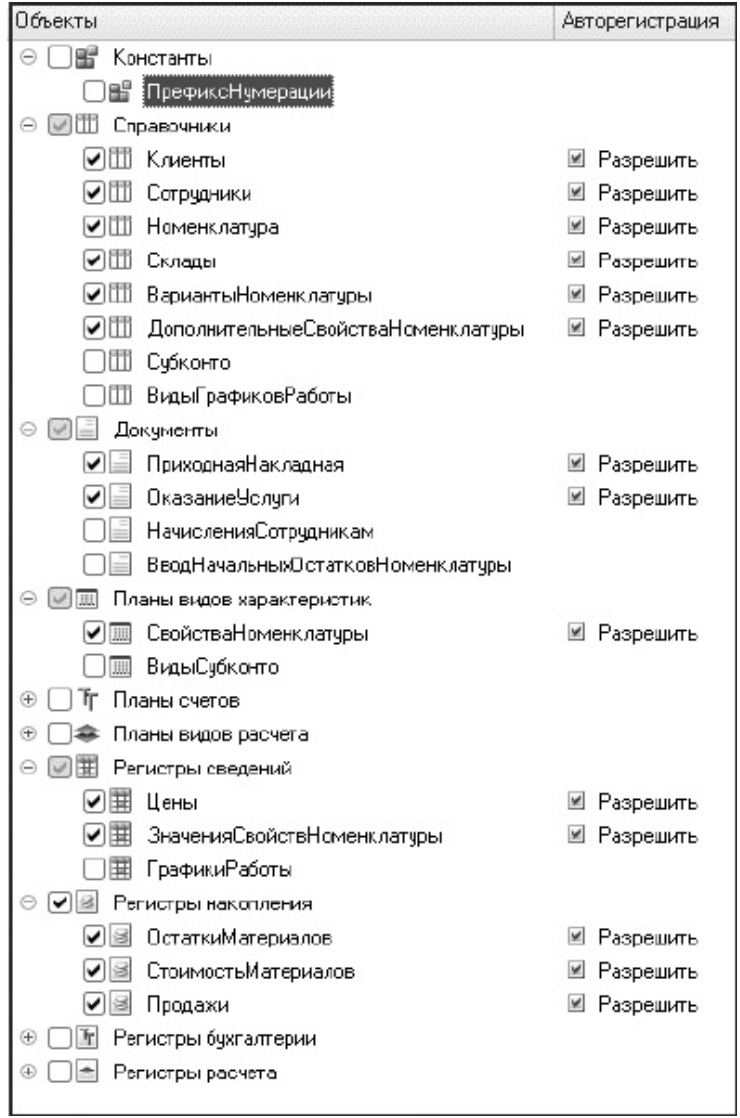

Объекты
Авторегистрация
Константы
ПрефиксНумерации
Справочники
Клиенты
Разрешить
Сотрудники
Разрешить
Номенклатура
Разрешить
Склады
Разрешить
ВариантыНоменклатуры
Разрешить
ДополнительныеСвойстваНоменклатуры
Разрешить
Субконто
ВидыГрафиковРаботы
Документы
ПриходнаяНакладная
Разрешить
ОказаниеУслуги
Разрешить
НачисленияСотрудникам
ВводНачальныхОстатковНоменклатуры
Планы видов характеристик
СвойстваНоменклатуры
Разрешить
ВидыСубконто
Планы счетов
Планы видов расчета
Регистры сведений
Цены
Разрешить
ЗначенияСвойствНоменклатуры
Разрешить
ГрафикиРаботы
Регистры накопления
ОстаткиМатериалов
Разрешить
СтоимостьМатериалов
Разрешить
Продажи
Разрешить
Регистры бухгалтерии
Регистры расчета

Рис. 24.3. Состав данных обмена

Теперь на закладке *Формы* окна редактирования объекта конфигурации нажмем кнопку открытия и с помощью конструктора создадим основную форму узла (рис. 24.4).

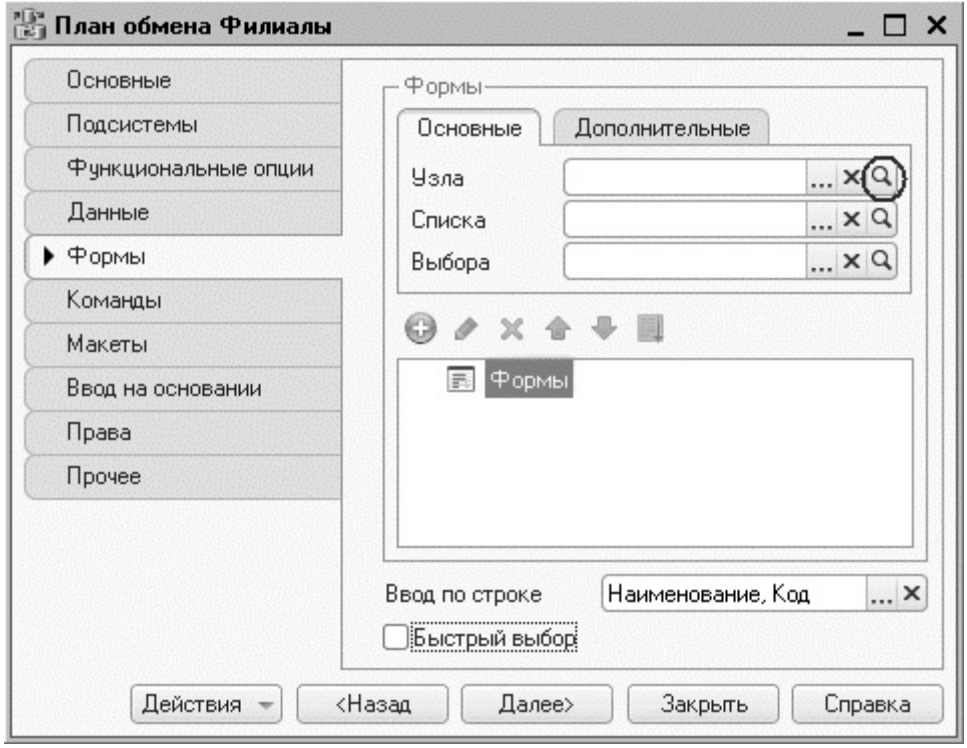


Рис. 24.4. Создание формы узла обмена

В окне элементов формы выделим корневой элемент *Форма*, вызовем его контекстное меню и создадим обработчик события формы *ПриСозданииНаСервере*.

Этот обработчик понадобится нам для того, чтобы запретить установку реквизита *Главный* для предопределенного узла, соответствующего данной информационной базе (листинг 24.4).

Листинг 24.4. Обработчик события формы «ПриСозданииНаСервере»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

  Если Объект.Ссылка = ПланыОбмена.Филиалы.ЭтотУзел() Тогда
    Элементы.Главный.Доступность = Ложь;

  КонецЕсли;

КонецПроцедуры
```

В этой процедуре мы используем метод менеджера плана обмена *ЭтотУзел()*, который возвращает ссылку на узел плана обмена, соответствующий данной информационной базе.

Затем создадим основную форму списка плана обмена, чтобы описать в ней

некоторые действия по регистрации нового узла обмена.

Суть этих действий будет заключаться в том, что при регистрации нового узла обмена мы должны будем сформировать для него все необходимые записи регистрации изменений для всех объектов конфигурации, входящих в данный план обмена. Это будет своего рода начальная синхронизация узла обмена всеми данными обмена.

Для этого на закладке *Команды* создадим команду *ЗарегистрироватьИзменения*.

В открывшейся палитре свойств нажмем кнопку открытия в строке *Действие*.

Шаблон обработчика события выполнения этой команды заполним следующим образом (листинг 24.5).

Листинг 24.5. Обработчик выполнения команды «ЗарегистрироватьИзменения»

```
&НаКлиенте
Процедура ЗарегистрироватьИзменения(Команда)

  РегистрацияИзмененийНаСервере(Элементы.Список.ТекущаяСтрока);

КонецПроцедуры
```

В этом обработчике мы вызываем процедуру *РегистрацияИзмененийНаСервере()*, которую мы напишем в дальнейшем. Она будет выполняться на сервере.

В параметре *Элементы.Список.ТекущаяСтрока* мы передаем в нее ссылку на объект *ПланОбмена.Филиалы*, используя свойство *ТекущаяСтрока* для таблицы *Список* (источником данных этой таблицы является динамический список узлов плана обмена *Филиалы)*.

Саму процедуру *РегистрацияИзмененийНаСервере()* мы предварим директивой компиляции *&НаСервереБезКонтекста*, так как процедура работает быстрее, если при ее вызове не передается контекст всей формы (листинг 24.6).

Листинг 24.6. Процедура «РегистрацияИзмененийНаСервере»

```
&НаСервереБезКонтекста
Процедура РегистрацияИзмененийНаСервере(Узел)

  // Регистрация изменений всех данных для узла.
  ПланыОбмена.ЗарегистрироватьИзменения(Узел);

КонецПроцедуры
```

В этой процедуре мы обращаемся к механизму регистрации изменений,

вызывая метод менеджера планов обмена – *ЗарегистрироватьИзменения()*.

В этот метод передается ссылка на текущий узел плана обмена *Филиалы*.

В результате выполнения этой процедуры в информационной базе будут созданы записи регистрации изменений, предназначенные для пересылки в созданный нами узел, для всех объектов обмена, указанных в составе данного плана обмена.

В заключение перейдем на закладку *Форма* и перетащим команду *ЗарегистрироватьИзменения* из окна команд в окно элементов формы, в командную панель формы.

В результате форма списка примет вид (рис. 24.5).

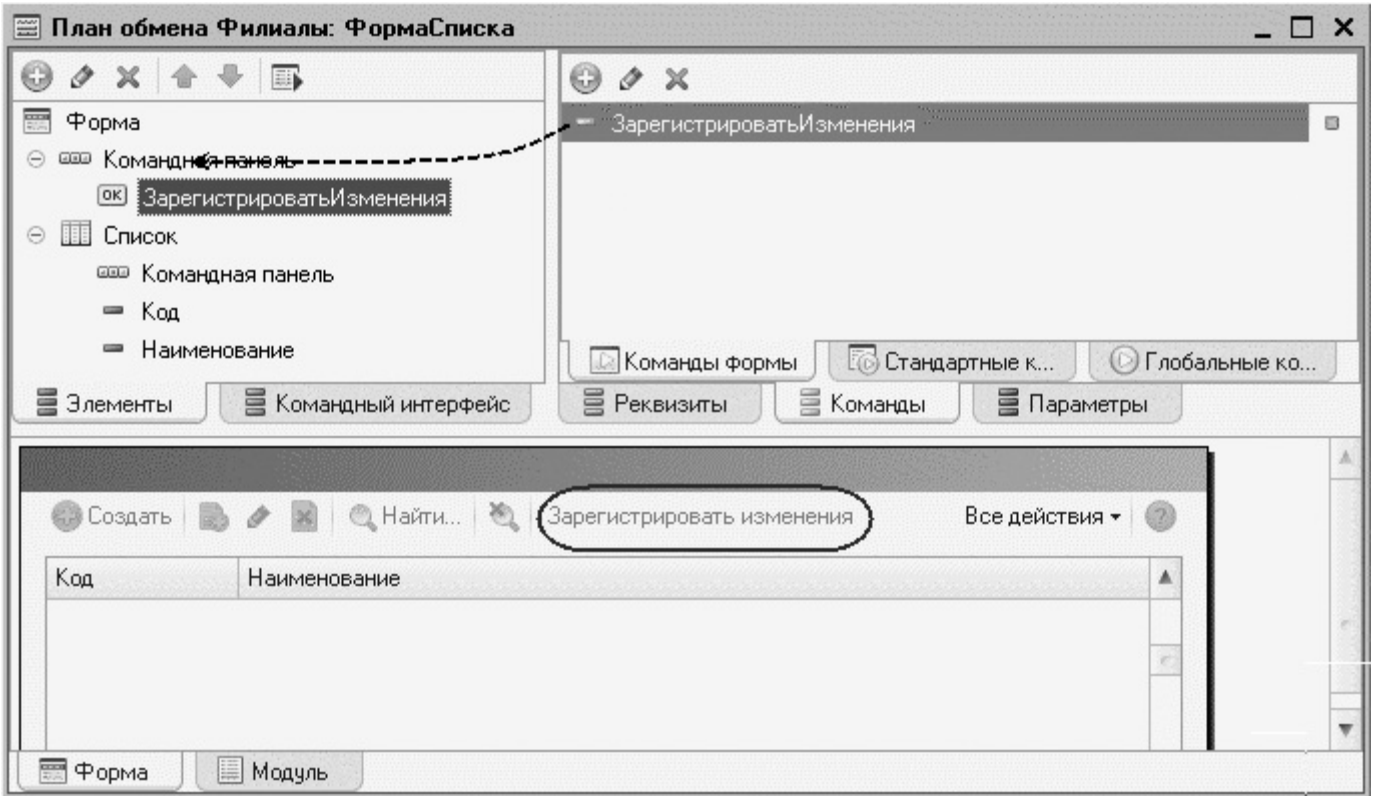


Рис. 24.5. Форма списка плана обмена

Причем кнопка *ЗарегистрироватьИзменения* должна быть доступна только в случае, если текущий узел не является предопределенным для данной информационной базы, иначе регистрация изменений невозможна.

Чтобы обеспечить такое поведение кнопки, создадим в модуле формы списка

функцию, выполняющуюся на сервере и возвращающую истину, если переданный в функцию узел является предопределенным (листинг 24.7).

Листинг 24.7. Функция «ПредопределенныйУзел()»

```
&НаСервереБезКонтекста
Функция ПредопределенныйУзел(Узел)

  Возврат Узел = ПланыОбмена.Филиалы.ЭтотУзел();

КонецФункции
```

Затем в окне элементов формы выделим элемент *Список*, вызовем его палитру свойств и создадим обработчик события *ПриАктивизацииСтроки*.

Заполним обработчик следующим образом (листинг 24.8).

Листинг 24.8. Обработчик события «ПриАктивизацииСтроки()» элемента формы «Список»

```
&НаКлиенте
Процедура СписокПриАктивизацииСтроки(Элемент)

  Если ПредопределенныйУзел(Элемент.ТекущаяСтрока) Тогда
    Элементы.ЗарегистрироватьИзменения.Доступность = Ложь;

  Иначе
    Элементы.ЗарегистрироватьИзменения.Доступность = Истина;
```

```
    КонецЕсли;

КонецПроцедуры
```

В этой процедуре доступность кнопки *ЗарегистрироватьИзменения* определяется в зависимости от значения функции *ПредопределенныйУзел()*, в которую передается ссылка на текущий узел (*Элемент.ТекущаяСтрока*).

На этом создание плана обмена завершено, и мы можем перейти непосредственно к созданию процедур обмена данными.

Процедуры обмена данными

В режиме «Конфигуратор»

Для инициализации обмена данными мы используем обработку.

Добавим новый объект конфигурации *Обработка* с именем *ОбменДанными*. На закладке *Формы* создадим основную форму обработки. В окне редактора форм на закладке *Команды* создадим команду формы *ВыполнитьОбмен*. В строке *Действие* нажмем кнопку открытия и создадим обработчик выполнения этой команды – вызов процедуры *ОбменСФилиалами(),* листинг 24.9.

Листинг 24.9. Обработчик команды «ВыполнитьОбмен»

```
&НаКлиенте
Процедура ВыполнитьОбмен(Команда)

  ОбменСФилиалами();

КонецПроцедуры
```

Затем в модуле формы создадим процедуру *ОбменСФилиалами*, исполняющуюся на сервере (листинг 24.10).

Листинг 24.10. Создание процедуры «ОбменСФилиалами»

```
&НаСервереБезКонтекста
Процедура ОбменСФилиалами() Экспорт

  ВыборкаУзлов = ПланыОбмена.Филиалы.Выбрать();

  Пока ВыборкаУзлов.Следующий() Цикл

    // Произвести обмен данными со всеми узлами, кроме текущего (ЭтотУзел).
    Если ВыборкаУзлов.Ссылка > ПланыОбмена.Филиалы.ЭтотУзел() Тогда
      УзелОбъект = ВыборкаУзлов.ПолучитьОбъект();

      // Получить сообщение.
      УзелОбъект.ПрочитатьСообщениеСИзменениями();

      // Сформировать сообщение.
      УзелОбъект.ЗаписатьСообщениеСИзменениями();
```

```
    КонецЕсли;
  КонецЦикла;

КонецПроцедуры
```

Алгоритм работы этой процедуры заключается в следующем: в цикле мы перебираем узлы, которые содержатся в плане обмена *Филиалы*, и для всех узлов, кроме себя самого, производим сначала чтение сообщений, поступивших из других узлов обмена (процедуру *ПрочитатьСообщенияСИзменениями* мы создадим позднее).

Затем мы формируем для них сообщения, предназначенные для передачи и содержащие измененные данные для этого узла (процедура *ЗаписатьСообщениеСИзменениями* также будет создана нами позднее).

В заключение перетащим команду *ВыполнитьОбмен* из окна команд в окно элементов формы. В результате форма обработки будет выглядеть следующим образом (рис. 24.6).

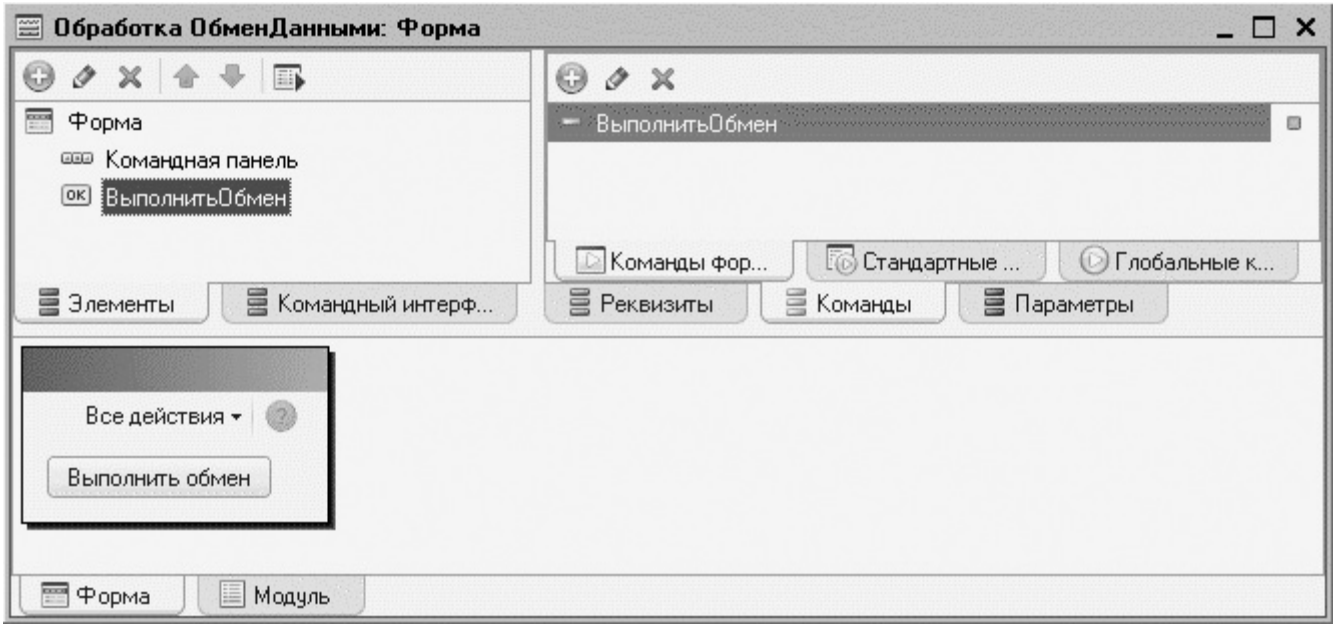


Рис. 24.6. Форма обработки

Процедура записи данных

Сами процедуры записи и чтения данных обмена мы разместим в модуле объекта План обмена *Филиалы*.

В окне редактирования этого объекта конфигурации перейдем на закладку *Прочее* и откроем модуль объекта. Сначала создадим процедуру, которая используется нами при обмене данными, – *ЗаписатьСообщениеСИзменениями*. Создавать ее мы будет постепенно.

Сначала мы сформируем имя файла, который будет содержать данные для обмена, и сообщим пользователю о начале и окончании выгрузки данных в узел (листинг 24.11).

Листинг 24.11. Формирование имени файла в процедуре записи данных

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ------
  ------";
  Сообщение.Сообщить();
  Каталог = КаталогВременныхФайлов();

  // Сформировать имя временного файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
  СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) +
  ".xml";
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец выгрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

Для упрощения примера мы будем обмениваться сообщениями через каталог временных файлов. Имена сообщений стандартизованы и имеют вид *MessageКодУзлаОтправителя_КодУзлаПолучателя.xml*.

После этого обратимся к механизмам записи/чтения XML-документов и создадим новый объект – *ЗаписьXML*. С помощью него откроем новый XML-файл для записи, запишем в него объявление XML. В конце процедуры завершим запись XML и закроем файл (листинг 24.12).

Листинг 24.12. Создание объекта записи XML в процедуре записи данных

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ------
  ------";
  Сообщение.Сообщить();
  Каталог = КаталогВременныхФайлов();

  // Сформировать имя временного файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
  СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) +
  ".xml";

  // Создать объект записи XML
  // *** ЗаписьXML-документов.
  ЗаписьXML = Новый ЗаписьXML;
  ЗаписьXML.ОткрытьФайл(ИмяФайла);
  ЗаписьXML.ЗаписатьОбъявлениеXML();
  ЗаписьXML.Закрыть();

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец выгрузки ------------";
```

```
  Сообщение.Сообщить();

КонецПроцедуры
```

Теперь мы обратимся к механизмам инфраструктуры сообщений и создадим новый объект *ЗаписьСообщенияОбмена*, метод которого *НачатьЗапись()* позволяет кроме всего прочего создать очередной номер сообщения и записать заголовок сообщения в XML. В конце процедуры мы опять же закончим запись сообщения (листинг 24.13).

Листинг 24.13. Создание очередного номера сообщения и запись заголовка сообщения в XML

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ------
  ------";
  Сообщение.Сообщить();
  Каталог = КаталогВременныхФайлов();

  // Сформировать имя временного файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
  СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) +
  ".xml";

  // Создать объект записи XML.
  // *** ЗаписьXML-документов.
  ЗаписьXML = Новый ЗаписьXML;
```

```
    ЗаписьXML.ОткрытьФайл(ИмяФайла);
    ЗаписьXML.ЗаписатьОбъявлениеXML();

    // *** Инфраструктура сообщений.
    ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
    ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;
    Сообщение.Сообщить();
    ЗаписьСообщения.ЗакончитьЗапись();

    ЗаписьXML.Закрыть();
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "-------- Конец выгрузки ------------";
    Сообщение.Сообщить();

КонецПроцедуры
```

Поскольку мы находимся в модуле объекта, то мы используем стандартный реквизит *Ссылка* в качестве ссылки на объект *План обмена Филиалы*.

После этого, чтобы получить данные, которые необходимо сохранить в этом файле, мы обратимся к механизму регистрации изменений и получим выборку из записей регистрации изменений, предназначенных данному узлу. При формировании выборки мы передаем вторым параметром номер сообщения (листинг 24.14).

Листинг 24.14. Получение выборки из записей регистрации изменений, предназначенных данному узлу

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ------
  ------";
  Сообщение.Сообщить();
  Каталог = КаталогВременныхФайлов();

  // Сформировать имя временного файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
  СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) +
  ".xml";

  // Создать объект записи XML
  // *** ЗаписьXML-документов.
  ЗаписьXML = Новый ЗаписьXML;
  ЗаписьXML.ОткрытьФайл(ИмяФайла);
  ЗаписьXML.ЗаписатьОбъявлениеXML();

  // *** Инфраструктура сообщений.
  ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
  ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;
  Сообщение.Сообщить();

  // Получить выборку измененных данных
  // *** Механизм регистрации изменений.
  ВыборкаИзменений =
```

```
  ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель,ЗаписьСообщения.НомерСо

  ЗаписьСообщения.ЗакончитьЗапись();
  ЗаписьXML.Закрыть();
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец выгрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

Теперь осталось только перебрать выборку записей в цикле и сериализовать их в открытый XML-файл (листинг 24.15).

Листинг 24.15. Перебор выборки записей и сериализация их в открытый XML-файл

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ------
  ------";
  Сообщение.Сообщить();
  Каталог = КаталогВременныхФайлов();

  // Сформировать имя временного файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
  СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) +
  ".xml";

  // Создать объект записи XML
```

```
  // *** ЗаписьXML-документов.
  ЗаписьXML = Новый ЗаписьXML;
  ЗаписьXML.ОткрытьФайл(ИмяФайла);
  ЗаписьXML.ЗаписатьОбъявлениеXML();

  // *** Инфраструктура сообщений.
  ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
  ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;
  Сообщение.Сообщить();

  // Получить выборку измененных данных
  // *** Механизм регистрации изменений.
  ВыборкаИзменений =
  ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель,ЗаписьСообщения.НомерСо
  Пока ВыборкаИзменений.Следующий() Цикл
    // Записать данные в сообщение *** XML-сериализация.
    ЗаписатьXML(ЗаписьXML, ВыборкаИзменений.Получить());

  КонецЦикла;

  ЗаписьСообщения.ЗакончитьЗапись();
  ЗаписьXML.Закрыть();
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец выгрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

На этом создание процедуры записи данных обмена закончено.

Процедура чтения данных

Порядок создания процедуры чтения данных обмена будет таким же, как и ранее: сначала мы сформируем имя файла, содержащего данные обмена (листинг 24.16).

Листинг 24.16. Формирование имени файла, содержащего данные обмена

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
  Если Не Файл.Существует() Тогда

    Возврат;

  КонецЕсли;

  УдалитьФайлы(ИмяФайла);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец загрузки ----------";
  Сообщение.Сообщить();
```

```
КонецПроцедуры
```

Мы формируем имя файла, которое надеемся найти в этом каталоге, а затем, создав новый объект *Файл* с таким именем, проверяем, существует ли он. Если такого файла нет, мы завершаем работу процедуры. Если же такой файл найден, нужно будет удалить его после того, как все данные, содержащиеся в нем, будут обработаны.

Теперь добавим в процедуру команды чтения найденного файла с данными обмена (листинг 24.17).

Листинг 24.17. Добавление чтения найденного файла с данными обмена

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
  Если Не Файл.Существует() Тогда

    Возврат;
```

```
КонецЕсли;

// *** Чтение документов XML
// Попытаться открыть файл.
ЧтениеXML = Новый ЧтениеXML;
Попытка
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

Исключение
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "Невозможно открыть файл обмена данными.";
  Сообщение.Сообщить();

  Возврат;

КонецПопытки;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

ЧтениеXML.Закрыть();
УдалитьФайлы(ИмяФайла);
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Конец загрузки ------------";
Сообщение.Сообщить();
```

```
КонецПроцедуры
```

Именно в этот момент мы обращаемся к механизмам записи/чтения документов XML, которые работают с ними на базовом уровне.

Для этого мы создаем новый объект *ЧтениеXML*, с помощью которого открываем найденный файл для чтения. В случае успеха мы выводим сообщение о начале загрузки данных из файла. В конце процедуры мы также прекращаем чтение XML-данных из файла методом *Закрыть()*.

Полученные таким образом данные должны являться некоторым сообщением обмена данными. Для того чтобы представить их в терминах сообщений, мы добавим в процедуру следующий код (листинг 24.18).

Листинг 24.18. Добавление чтения заголовка XML-сообщения

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
```

```
Если Не Файл.Существует() Тогда

  Возврат;

КонецЕсли;

// *** Чтение документов XML
// Попытаться открыть файл.
ЧтениеXML = Новый ЧтениеXML;
Попытка
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

Исключение
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "Невозможно открыть файл обмена данными.";
  Сообщение.Сообщить();

  Возврат;

КонецПопытки;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

// Загрузить из найденного файла
```

```
    // *** Инфраструктура сообщений.
    ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

    // Читать заголовок сообщения обмена данными - файла XML.
    ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

    ЧтениеСообщения.ЗакончитьЧтение();
    ЧтениеXML.Закрыть();
    УдалитьФайлы(ИмяФайла);
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "-------- Конец загрузки ------------";
    Сообщение.Сообщить();

КонецПроцедуры
```

Здесь мы обращаемся к механизмам инфраструктуры сообщений планов обмена и создаем объект *ЧтениеСообщенияОбмена*. Используя метод этого объекта *НачатьЧтение()*, мы считываем заголовок XML-сообщения, в котором содержится в том числе информация об отправителе сообщения. После того как все сообщение будет нами обработано, мы заканчиваем чтение.

Теперь, когда мы представили данные обмена в виде сообщения и получили его заголовок, можно произвести одну проверку, перед тем как начать собственно обрабатывать данные (листинг 24.19).

Листинг 24.19. Добавление проверки сообщения

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
  Если Не Файл.Существует() Тогда

    Возврат;

  КонецЕсли;

  // *** Чтение документов XML
  // Попытаться открыть файл.
  ЧтениеXML = Новый ЧтениеXML;
  Попытка
    ЧтениеXML.ОткрытьФайл(ИмяФайла);

  Исключение
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "Невозможно открыть файл обмена данными.";
    Сообщение.Сообщить();

    Возврат;

  КонецПопытки;
```

```
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
  --";
  Сообщение.Сообщить();
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = " - Считывается файл " + ИмяФайла;
  Сообщение.Сообщить();

  // Загрузить из найденного файла
  // *** Инфраструктура сообщений.
  ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

  // Читать заголовок сообщения обмена данными - файла XML.
  ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

  // Сообщение предназначено не для этого узла.
  Если ЧтениеСообщения.Отправитель > Ссылка Тогда
    ВызватьИсключение "Неверный узел";

  КонецЕсли;

  ЧтениеСообщения.ЗакончитьЧтение();
  ЧтениеXML.Закрыть();
  УдалитьФайлы(ИмяФайла);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец загрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

Мы проверяем, является ли отправитель сообщения тем узлом, для которого мы в данном вызове этой процедуры производим обмен данными.

Если все в порядке, то перед тем как начать чтение данных, следует удалить все записи регистрации изменений, которые были сделаны для этого узла и соответствовали номерам сообщений меньше или равным указанному в обрабатываемом нами сообщении как номер принятого. Это делается затем, чтобы исключить дублирование данных, которые уже были ранее посланы этому узлу и им обработаны (листинг 24.20).

Листинг 24.20. Удаление записей регистрации изменений для узла отправителя

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
  Если Не Файл.Существует() Тогда

    Возврат;

  КонецЕсли;
```

```
// *** Чтение документов XML
// Попытаться открыть файл.
ЧтениеXML = Новый ЧтениеXML;
Попытка
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

Исключение
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "Невозможно открыть файл обмена данными.";
  Сообщение.Сообщить();

  Возврат;

КонецПопытки;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

// Загрузить из найденного файла
// *** Инфраструктура сообщений.
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными - файла XML.
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Сообщение предназначено не для этого узла.
```

```
  Если ЧтениеСообщения.Отправитель > Ссылка Тогда
    ВызватьИсключение "Неверный узел";

  КонецЕсли;

  // Удаляем регистрацию изменений для узла отправителя сообщения.
  // *** Служба регистрации изменений.
  ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
  ЧтениеСообщения.НомерПринятого);

  ЧтениеСообщения.ЗакончитьЧтение();
  ЧтениеXML.Закрыть();
  УдалитьФайлы(ИмяФайла);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец загрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

Обратите внимание, что здесь мы обращаемся к службе регистрации изменений и используем метод *УдалитьРегистрациюИзменений()* для выполнения описанных действий.

Теперь, наконец, мы можем приступить к чтению непосредственно самих данных, содержащихся в сообщении (листинг 24.21).

Листинг 24.21. Чтение данных из сообщения

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
  Если Не Файл.Существует() Тогда

    Возврат;

  КонецЕсли;

  // *** Чтение документов XML
  // Попытаться открыть файл.
  ЧтениеXML = Новый ЧтениеXML;
  Попытка
    ЧтениеXML.ОткрытьФайл(ИмяФайла);

  Исключение
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "Невозможно открыть файл обмена данными.";
    Сообщение.Сообщить();

    Возврат;

  КонецПопытки;
```

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

// Загрузить из найденного файла
// *** Инфраструктура сообщений.
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными - файла XML.
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Сообщение предназначено не для этого узла.
Если ЧтениеСообщения.Отправитель > Ссылка Тогда
  ВызватьИсключение "Неверный узел";

КонецЕсли;

// Удаляем регистрацию изменений для узла отправителя сообщения.
// *** Служба регистрации изменений
ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
ЧтениеСообщения.НомерПринятого);

// Читаем данные из сообщения *** XML-сериализация.
Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

КонецЦикла;
```

```
  ЧтениеСообщения.ЗакончитьЧтение();
  ЧтениеXML.Закрыть();
  УдалитьФайлы(ИмяФайла);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец загрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

Чтение данных выполняется в цикле, причем мы снова обращаемся к механизмам XML-сериализации и методом глобального контекста *ВозможностьЧтенияXML()* получаем очередной тип данных XML из объекта *ЧтениеXML* и определяем, имеется ли соответствующий тип «1С:Предприятия». В случае успеха выполнение цикла продолжается.

И первое, что нам нужно сделать, – представить данные XML в виде некоторого значения, имеющего тип «1С:Предприятия». Для этого мы используем метод глобального контекста *ПрочитатьXML(),* листинг 24.22.

Листинг 24.22. Представление данных XML в виде значения, имеющего тип

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
```

```
ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
".xml";
Файл = Новый Файл(ИмяФайла);
Если Не Файл.Существует() Тогда

  Возврат;

КонецЕсли;

// *** Чтение документов XML
// Попытаться открыть файл.
ЧтениеXML = Новый ЧтениеXML;
Попытка
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

Исключение
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "Невозможно открыть файл обмена данными.";
  Сообщение.Сообщить();

  Возврат;

КонецПопытки;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
```

```
Сообщение.Сообщить();

// Загрузить из найденного файла
// *** Инфраструктура сообщений.
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными - файла XML.
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Сообщение предназначено не для этого узла.
Если ЧтениеСообщения.Отправитель > Ссылка Тогда
  ВызватьИсключение "Неверный узел";

КонецЕсли;

// Удаляем регистрацию изменений для узла отправителя сообщения.
// *** Служба регистрации изменений.
ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
ЧтениеСообщения.НомерПринятого);

// Читаем данные из сообщения *** XML-сериализация.
Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

  // Читаем очередное значение.
  Данные = ПрочитатьXML(ЧтениеXML);

КонецЦикла;

ЧтениеСообщения.ЗакончитьЧтение();
ЧтениеXML.Закрыть();
```

```
  УдалитьФайлы(ИмяФайла);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец загрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

В результате выполнения этого метода переменная *Данные* будет содержать объект «1С:Предприятия», соответствующий данным XML.

Теперь, после того как объект «1С:Предприятия» получен, следует разрешить возможную коллизию (листинг 24.23).

Листинг 24.23. Разрешение возможных коллизий

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

  Каталог = КаталогВременныхФайлов();

  // Сформировать имя файла.
  ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
  СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
  ".xml";
  Файл = Новый Файл(ИмяФайла);
  Если Не Файл.Существует() Тогда

    Возврат;
```

```
КонецЕсли;

// *** Чтение документов XML
// Попытаться открыть файл.
ЧтениеXML = Новый ЧтениеXML;
Попытка
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

Исключение
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "Невозможно открыть файл обмена данными.";
  Сообщение.Сообщить();

  Возврат;

КонецПопытки;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

// Загрузить из найденного файла
// *** Инфраструктура сообщений.
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными - файла XML.
```

```
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Сообщение предназначено не для этого узла.
Если ЧтениеСообщения.Отправитель > Ссылка Тогда
  ВызватьИсключение "Неверный узел";

КонецЕсли;

// Удаляем регистрацию изменений для узла отправителя сообщения.
// *** Служба регистрации изменений.
ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
ЧтениеСообщения.НомерПринятого);

// Читаем данные из сообщения *** XML-сериализация.
Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

  // Читаем очередное значение.
  Данные = ПрочитатьXML(ЧтениеXML);

  // Не переносим изменение, полученное в главный из неглавного, если есть
  регистрация изменения.
  Если Не ЧтениеСообщения.Отправитель.Главный И
    ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправитель,
    Данные) Тогда
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = " - Изменения отклонены";
    Сообщение.Сообщить();

    Продолжить;

  КонецЕсли;
```

```
    КонецЦикла;

    ЧтениеСообщения.ЗакончитьЧтение();
    ЧтениеXML.Закрыть();
    УдалитьФайлы(ИмяФайла);
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "-------- Конец загрузки ------------";
    Сообщение.Сообщить();

КонецПроцедуры
```

Возможная коллизия разрешается следующим образом: мы проверяем, является ли узел-отправитель главным узлом и есть ли записи об изменении этого объекта для данного узла в нашей базе данных. Если объект изменялся в нашей базе и отправитель не является главным узлом, мы отклоняем запись полученного объекта. Во всех остальных случаях мы принимаем изменения полученного объекта.

Теперь единственное, что нам осталось сделать, – записать полученные данные (листинг 24.24).

Листинг 24.24. Запись полученных данных

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт
```

```
Каталог = КаталогВременныхФайлов();

// Сформировать имя файла.
ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") + "Message" +
СокрЛП(Ссылка.Код) + "_" + СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) +
".xml";
Файл = Новый Файл(ИмяФайла);
Если Не Файл.Существует() Тогда

  Возврат;

КонецЕсли;

// *** Чтение документов XML
// Попытаться открыть файл.
ЧтениеXML = Новый ЧтениеXML;
Попытка
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

Исключение
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "Невозможно открыть файл обмена данными.";
  Сообщение.Сообщить();

  Возврат;

КонецПопытки;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------
--";
```

```
Сообщение.Сообщить();
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = " - Считывается файл " + ИмяФайла;
Сообщение.Сообщить();

// Загрузить из найденного файла
// *** Инфраструктура сообщений.
ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

// Читать заголовок сообщения обмена данными - файла XML.
ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

// Сообщение предназначено не для этого узла.
Если ЧтениеСообщения.Отправитель > Ссылка Тогда
  ВызватьИсключение "Неверный узел";

КонецЕсли;

// Удаляем регистрацию изменений для узла отправителя сообщения
// *** Служба регистрации изменений.
ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,
ЧтениеСообщения.НомерПринятого);

// Читаем данные из сообщения *** XML-сериализация.
Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

  // Читаем очередное значение.
  Данные = ПрочитатьXML(ЧтениеXML);

  // Не переносим изменение, полученное в главный из неглавного, если есть
```

```
    регистрация изменения.
    Если Не ЧтениеСообщения.Отправитель.Главный И
      ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправитель,
      Данные) Тогда
      Сообщение = Новый СообщениеПользователю;
      Сообщение.Текст = " - Изменения отклонены";
      Сообщение.Сообщить();

      Продолжить;

    КонецЕсли;

    // Записать полученные данные.
    Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
    Данные.ОбменДанными.Загрузка = Истина;
    Данные.Записать();

  КонецЦикла;

  ЧтениеСообщения.ЗакончитьЧтение();
  ЧтениеXML.Закрыть();
  УдалитьФайлы(ИмяФайла);
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "-------- Конец загрузки ------------";
  Сообщение.Сообщить();

КонецПроцедуры
```

Перед записью полученного объекта мы устанавливаем у него в параметрах

обмена данными узел отправителя для того, чтобы система при записи этого объекта в нашей базе данных не формировала записи регистрации изменений этого объекта для того узла, от которого мы его только что получили.

Кроме этого, в параметрах обмена данными мы устанавливаем свойство *Загрузка*, информирующее систему о том, что запись объекта будет происходить в режиме обновления данных, полученных в результате обмена. Такое указание позволяет системе упростить процедуру записи объекта, отказавшись от ряда стандартных проверок и исключив изменения связанных данных, которые выполняются при обычной записи.

На этом создание процедуры получения и обработки данных обмена закончено.

Проверка работы обмена данными

В режиме «Конфигуратор»

Чтобы иметь возможность редактировать константу *ПрефиксНумерации,* раскроем ветвь *Общие* дерева объектов конфигурации, выделим ветвь *Общие формы* и с помощью конструктора форм создадим форму констант с именем *ОбщиеНастройки*.

Откроем окно свойств *Дополнительно* для этой формы (контекстное меню – пункт *Дополнительно*) и укажем принадлежность этой формы к подсистеме

Предприятие (рис. 24.7).

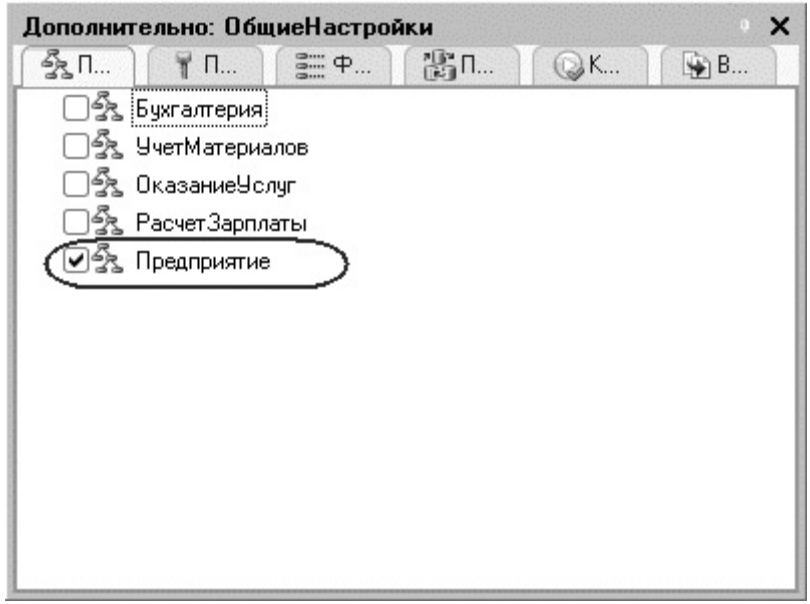


Рис. 24.7. Окно «Дополнительно» формы констант «ОбщиеНастройки»

В окне редактирования объекта конфигурации *План обмена Филиалы* и объекта Обработка *ОбменДанными* на закладке *Подсистемы* также укажем их принадлежность к подсистеме *Предприятие*.

Таким образом, доступ к командам открытия плана обмена, обработки, а также формы констант будет иметь только *Администратор*, так как подсистема *Предприятие* будет доступна только для роли *Администратор*.

А также в окне редактора командного интерфейса подсистем *Все подсистемы* включим видимость у команды *Филиал: создать* в группе панели действий *Создать* подсистемы *Предприятие* и установим следующий порядок следования команд в группе панели действий *Сервис*:

- *Поиск в данных,*
- *Общие настройки,*
- *Обмен данными,*
- *Планировщик заданий.*

И в заключение создадим новый каталог, в котором будет размещаться база нашего филиала.

Обновим конфигурацию базы данных (*F7*). Затем сохраним в созданный каталог нашу конфигурацию, выполнив команду главного меню *Конфигурация* > *Сохранить конфигурацию в файл…*

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и установим необходимые значения в нашей центральной базе.

Прежде всего, зададим значение константы *Префикс нумерации – ЦБ*. Для

этого выполним команду *Общие настройки* в панели действий раздела *Предприятие* (рис. 24.8).

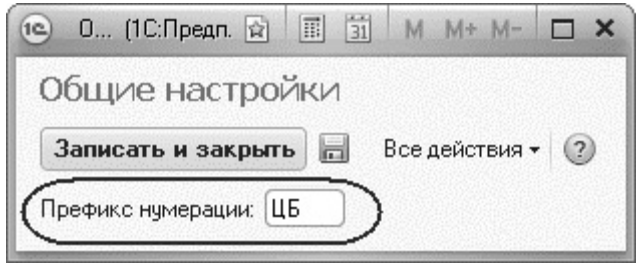


Рис. 24.8. Значение константы «Префикс нумерации»

Нажмем *Записать и закрыть*. После этого откроем план обмена *Филиалы* и зададим параметры узла по умолчанию, то есть параметры нашей базы.

Для этого выполним команду *Филиалы* в панели навигации раздела *Предприятие*. В списке планов обмена уже присутствует одна запись. Откроем и отредактируем ее.

Код базы будет *ЦБ*, а наименование – *Центральная база.*

Не забудьте, что именно код идентифицирует узлы обмена в различных базах, поэтому в базе филиала мы будем создавать узлы с такими же кодами (рис. 24.9).

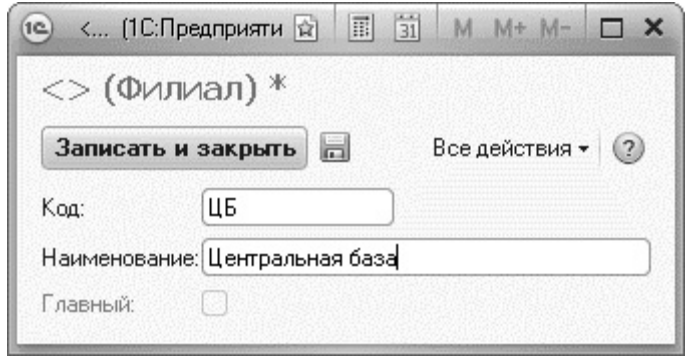


Рис. 24.9. Создание узла плана обмена

Нажмем *Записать и закрыть*.

Затем нажмем кнопку *Создать* или воспользуемся командой *Филиал* в панели действий.

Создадим новый узел, который будет соответствовать базе филиала, присвоим ему код *Фил* и наименование *Филиал* (рис. 24.10).

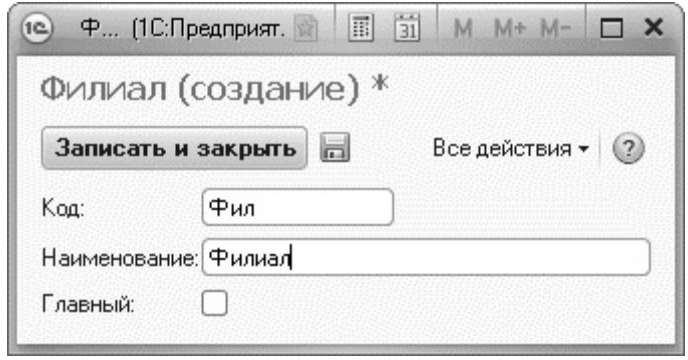


Рис. 24.10. Создание узла плана обмена

Обратите внимание, что предопределенный узел нашей информационной базы (*Центральная база*) выделен в списке узлов обмена специальной пиктограммой. Кнопка *Зарегистрировать изменения* недоступна для этого узла.

Выделим в списке новый узел *Филиал* и нажмем кнопку *Зарегистрировать изменения*.

Теперь вызовем обработку *ОбменДанными* и нажмем *Выполнить обмен*.

В окне сообщений появится следующий текст (рис. 24.11).

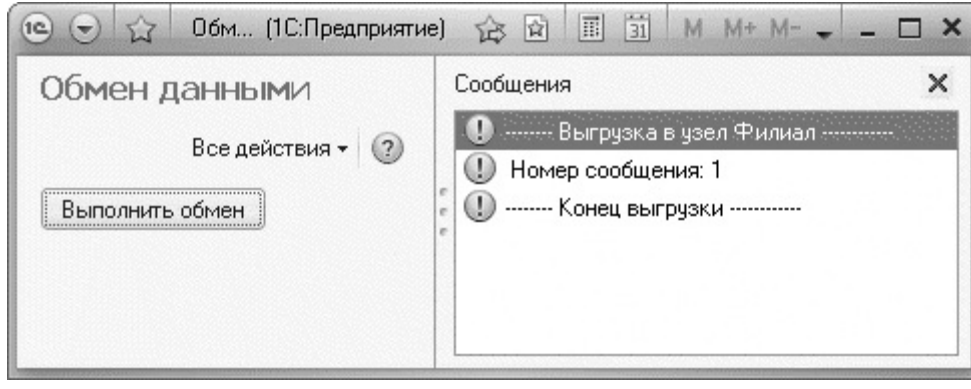


Рис. 24.11. Окно служебных сообщений

Таким образом, в результате обмена данными центральная база сформировала файл обмена, содержащий изменения всех данных, которыми она обменивается с филиалом.

Запуск базы филиала

Настало время перейти к базе филиала.

Запустим «1С:Предприятие» и добавим в список баз новую базу с пустой конфигурацией, которая будет расположена в созданном нами каталоге базы филиала. Для этого в окне запуска «1С:Предприятия» нажмем кнопку *Добавить* и выберем *Создание новой информационной базы*. Нажмем *Далее*.

В следующем окне выберем *Создание информационной базы без*

конфигурации для … загрузки выгруженной ранее информационной базы.

Нажмем *Далее*, затем укажем наименование информационной базы, например, *база Филиала*.

Нажмем *Далее*, затем укажем каталог информационной базы, где находится сохраненная конфигурация, например, *D:\1C_8.2\filial*, нажмем *Далее* и затем *Готово*.

В режиме «Конфигуратор»

Откроем созданную нами конфигурацию *база Филиала* в режиме *Конфигуратор*. Выполним команду главного меню *Конфигурация* > *Открыть конфигурацию*. Мы видим, что список объектов конфигурации пуст.

Теперь загрузим конфигурацию из файла (*Конфигурация* > *Загрузить конфигурацию из файла…*).

В окне выбора файла выберем каталог и имя файла, где находится сохраненная конфигурация, например, *D:\1C_8.2\filial\1Cv8.cf*. На вопрос системы об обновлении конфигурации ответим утвердительно и в окне изменений структуры конфигурации нажмем *Принять*.

Теперь все объекты конфигурации перенесены из нашей центральной базы.

Выполним команду главного меню *Администрирование* > *Пользователи* и создадим в конфигурации филиала одного пользователя – *Администратор* с ролью *Администратор*. Дело в том, что пользователей в каждой информационной базе нужно создавать заново.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

Первым делом зададим значение константы *ПрефиксНумерации* – *ФЛ* (рис. 24.12).

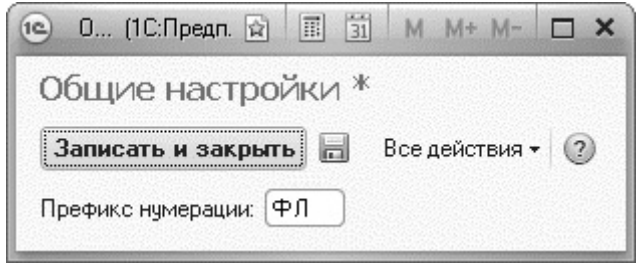


Рис. 24.12. Редактирование константы

Затем откроем план обмена *Филиал* и опишем предопределенный узел (узел текущей информационной базы). Для этого выполним команду *Филиалы* в панели навигации раздела *Предприятие*.

В списке планов обмена уже присутствует одна запись. Откроем и

отредактируем ее. Зададим код *Фил* и наименование *Филиал* (рис. 24.13).

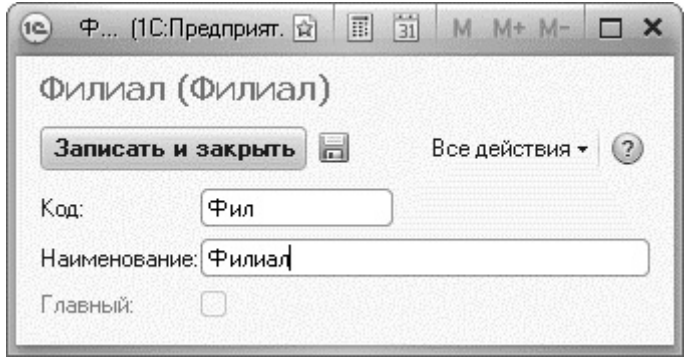


Рис. 24.13. Создание узла плана обмена

После этого создадим новый узел плана обмена с кодом *ЦБ*, наименованием *Центральная база* и признаком *Главный* (рис. 24.14).

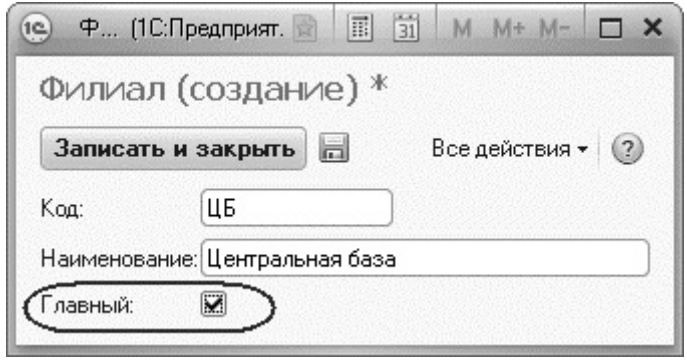


Рис. 24.14. Создание узла плана обмена

Выделим в списке узлов обмена новый узел *Центральная база* и нажмем кнопку *Зарегистрировать изменения*.

Теперь для большей наглядности откроем список справочника *Клиенты*. Сейчас в нем нет ни одного элемента.

Запустим обработку *ОбменДанными* и нажмем *Выполнить обмен*. Справочник будет заполнен элементами, а в окне сообщений появится текст (рис. 24.15).

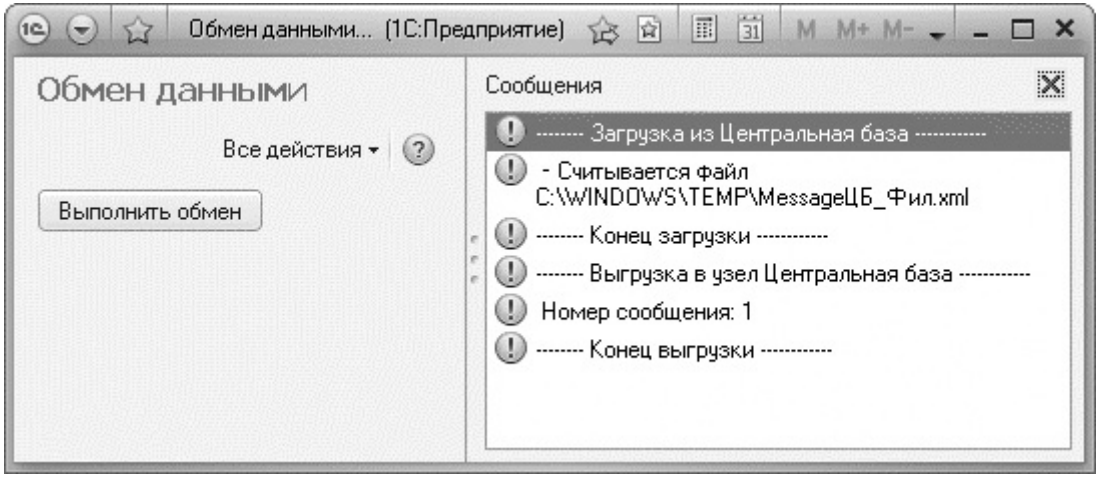


Рис. 24.15. Сообщения плана обмена

Теперь проверим, как будет происходить обмен в другую сторону.

Создадим в справочнике *Клиенты* нового клиента с произвольным наименованием.

Обратите внимание, что нумерация кода нового клиента начинается с единицы и имеет префикс *ФЛ*.

После этого снова нажмем *Выполнить обмен* в открытой форме обработки *ОбменДанными*. Затем перейдем в центральную базу, также выполним обмен данными и убедимся, что клиент, созданный в базе филиала, перенесен в центральную базу.

Механизм распределенных информационных баз

Механизм распределенных информационных баз является развитием универсального механизма обмена данными.

ВНИМАНИЕ!

> *Если вы используете учебную версию платформы «1С:Предприятие 8.2», то воспроизвести этот пример не удастся, так как учебная версия не поддерживает работу с распределенными информационными базами.*

Он реализует привычную по прежним версиям «1С:Предприятия» модель распределенной информационной базы, которая подразумевает наличие

идентичных конфигураций во всех узлах, имеет древовидную структуру и позволяет выполнять обмен как измененными данными, так и изменениями, внесенными в конфигурацию.

Механизм распределенных информационных баз реализуется планами обмена. Для этого объект конфигурации *План обмена* содержит свойство *Распределенная информационная база*.

Если это свойство установлено, для данного плана обмена включается механизм распределенных информационных баз и разработчик получает возможность создать распределенную базу исключительно интерактивными средствами, без написания кода.

Такая возможность не исключает программного управления обменом, которое также доступно при работе с распределенными информационными базами.

В ходе создания примера мы рассмотрим оба варианта организации обмена в распределенных информационных базах.

Основные сведения

Как мы уже говорили выше, распределенная информационная база должна иметь четко определенную древовидную структуру. Количество уровней в

такой структуре не ограничено, главное – между двумя связанными узлами всегда должно быть определено отношение «главный – подчиненный» (рис. 24.16).

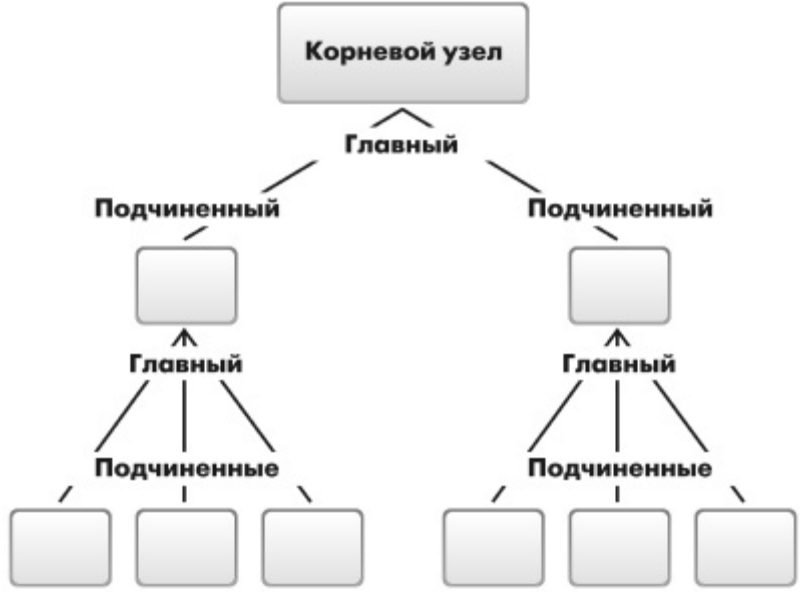


Рис. 24.16. Структура распределенной информационной базы

Таким образом, любой узел этой структуры может иметь произвольное количество подчиненных узлов (в том числе и ни одного). Кроме этого, все узлы, кроме одного, должны иметь по одному главному узлу, и один узел не будет иметь главного узла – это корневой узел. Такое жесткое задание структуры узлов необходимо для определения порядка миграции изменений

данных и изменений конфигурации.

Конфигурация может быть изменена только в узле, не имеющем главного узла (то есть в корневом). Изменения данных могут выполняться в любом узле.

Изменения конфигурации будут передаваться от главного к подчиненным узлам. Изменения данных могут передаваться между любыми связанными узлами.

Разрешение коллизий также будет производиться исходя из отношения «главный – подчиненный». Если изменения выполнены одновременно и в главном, и в подчиненном узле, при обмене данными будут приняты только изменения главного узла, а изменения подчиненного отвергнуты.

Для любого подчиненного узла возможно создание *начального образа* – информационной базы, созданной на основании конфигурации и данных главного узла в соответствии с правилами, определяемыми планом обмена. Процедура создания начального образа узла может выполняться неоднократно, при этом удаляются все записи изменений в базе главного узла для подчиненного узла. Сразу после создания начальный образ готов к обмену с главным узлом.

Создание начального образа является рекомендуемым способом создания

подчиненного узла в распределенной информационной базе.

Постановка задачи

В качестве примера, на котором мы проиллюстрируем использование механизма распределенных информационных баз, будет создание нескольких отделений нашего ООО «На все руки мастер».

В отличие от филиалов, которые расположены в других городах, являются отдельными юридическими лицами и довольно самостоятельны в плане организации учета своей деятельности, отделения нашего предприятия расположены в этом же городе, никакой юридической самостоятельностью не обладают и ведут учет в точности так, как это организовано в главном офисе.

Поэтому все они используют ту же конфигурацию, что и главный офис, причем если главный офис вносит какие-либо изменения в свою конфигурацию, они должны быть своевременно внесены и в конфигурации отделений.

Для реализации такой схемы работы распределенная информационная база подойдет как нельзя лучше, и сначала мы организуем обмен с отделениями, используя исключительно интерактивные средства.

Интерактивный обмен

В режиме «Конфигуратор»

Для построения распределенной информационной базы нам понадобится создать еще один объект конфигурации *План обмена*, который мы назовем *Отделения*, представление объекта – *Отделение*.

Для этого плана обмена мы установим свойство *Распределенная информационная база* (рис. 24.17).

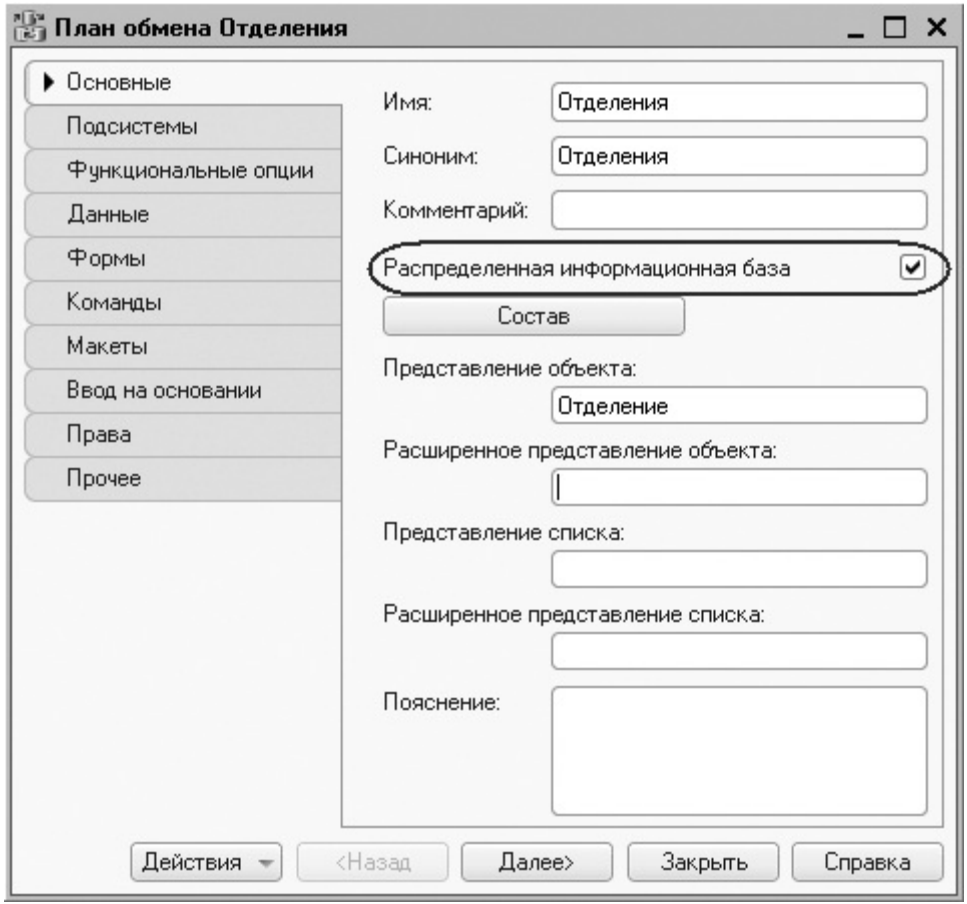


Рис. 24.17. Установка свойства «Распределенная информационная база»

На закладке *Формы* установим флажок *Быстрый выбор*, чтобы иметь возможность выбора узлов плана обмена из выпадающего списка. Затем

определим состав объектов, участвующих в обмене.

На закладке *Основные* нажмем кнопку *Состав* и определим тот же состав данных для обмена, что и в плане обмена *Филиалы*. Включим в обмен все объекты, не относящиеся к ведению бухгалтерии и расчету зарплаты.

В результате состав данных обмена должен выглядеть следующим образом (рис. 24.18).

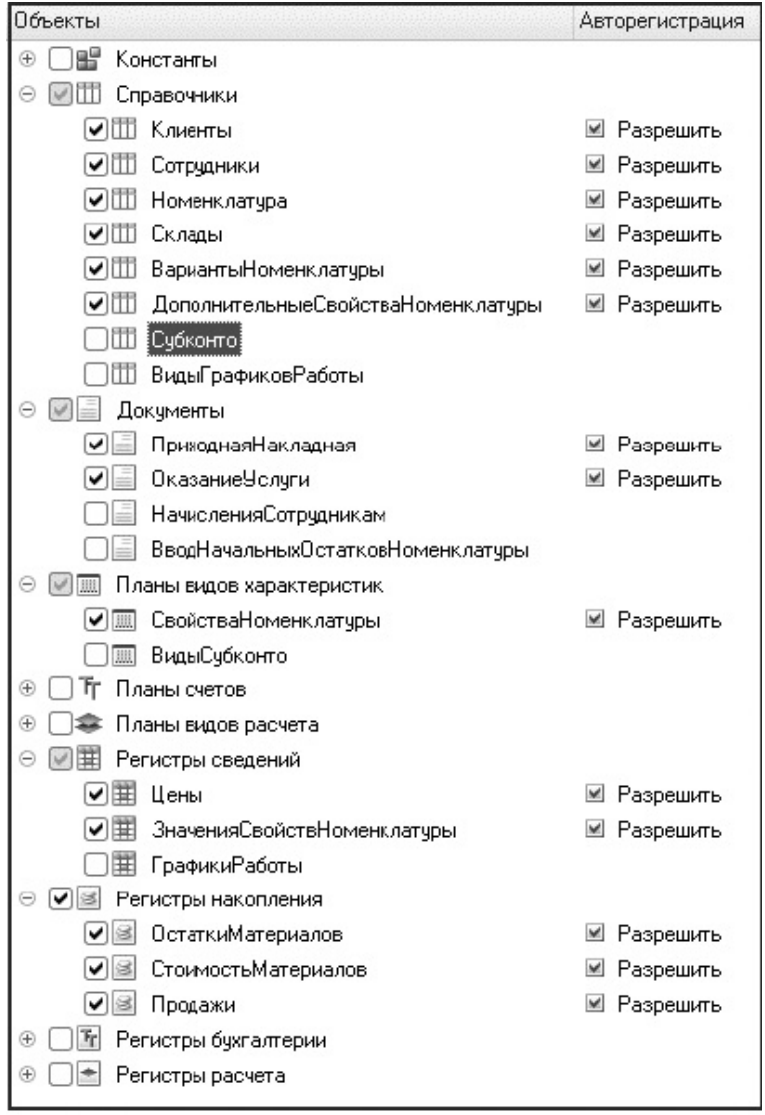

Объекты
Авторегистрация
Константы
Справочники
Клиенты
Разрешить
Сотрудники
Разрешить
Номенклатура
Разрешить
Склады
Разрешить
ВариантыНоменклатуры
Разрешить
ДополнительныеСвойстваНоменклатуры
Разрешить
Субконто
ВидыГрафиковРаботы
Документы
ПриходнаяНакладная
Разрешить
ОказаниеУслуги
Разрешить
НачисленияСотрудникам
ВводНачальныхОстатковНоменклатуры
Планы видов характеристик
СвойстваНоменклатуры
Разрешить
ВидыСубконто
Планы счетов
Планы видов расчета
Регистры сведений
Цены
Разрешить
ЗначенияСвойствНоменклатуры
Разрешить
ГрафикиРаботы
Регистры накопления
ОстаткиМатериалов
Разрешить
СтоимостьМатериалов
Разрешить
Продажи
Разрешить
Регистры бухгалтерии
Регистры расчета

Рис. 24.18. Состав данных обмена

На закладке *Подсистемы* укажем принадлежность плана обмена к подсистеме *Предприятие*. Таким образом, команда открытия плана обмена *Отделения* будет доступна только для роли *Администратор*.

А также в окне редактора командного интерфейса подсистем *Все подсистемы* включим видимость у команды *Отделение: создать* в группе панели действий *Создать* подсистемы *Предприятие*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки. Откроем план обмена *Отделения* и зададим параметры центрального узла (предопределенный элемент плана обмена). Для этого выполним команду *Отделения* в панели навигации раздела *Предприятие*.

В списке планов обмена уже присутствует одна запись. Это предопределенный узел нашей информационной базы. Откроем и отредактируем ее.

Внесем код *ЦБ* и наименование *Центральная база*. После этого создадим новый узел с кодом *Отд* и наименованием *Отделение*.

Для созданного нами узла доступны три кнопки в командной панели формы

плана обмена: *Записать изменения*, *Прочитать изменения* и *Создать начальный образ* (рис. 24.19).

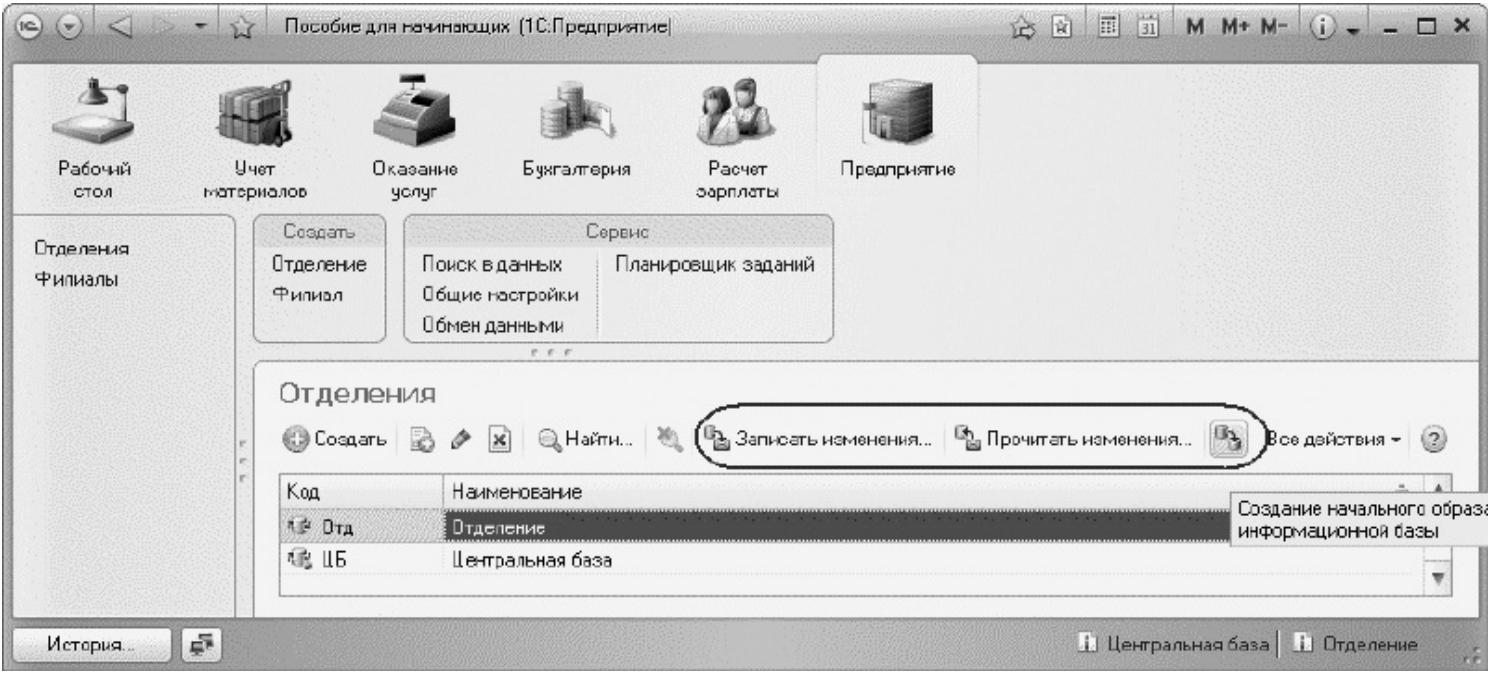


Рис. 24.19. Команды работы с распределенной информационной базой

Не откладывая, воспользуемся одной из них, чтобы создать начальный образ информационной базы нашего отделения. Для этого нам потребуется сначала создать на диске новый каталог, в котором будет располагаться база отделения.

После этого выделим в списке узлов обмена узел *Отделение,* выполним команду *Создать начальный образ* и укажем, что информационная база будет расположена на данном компьютере (рис. 24.20).

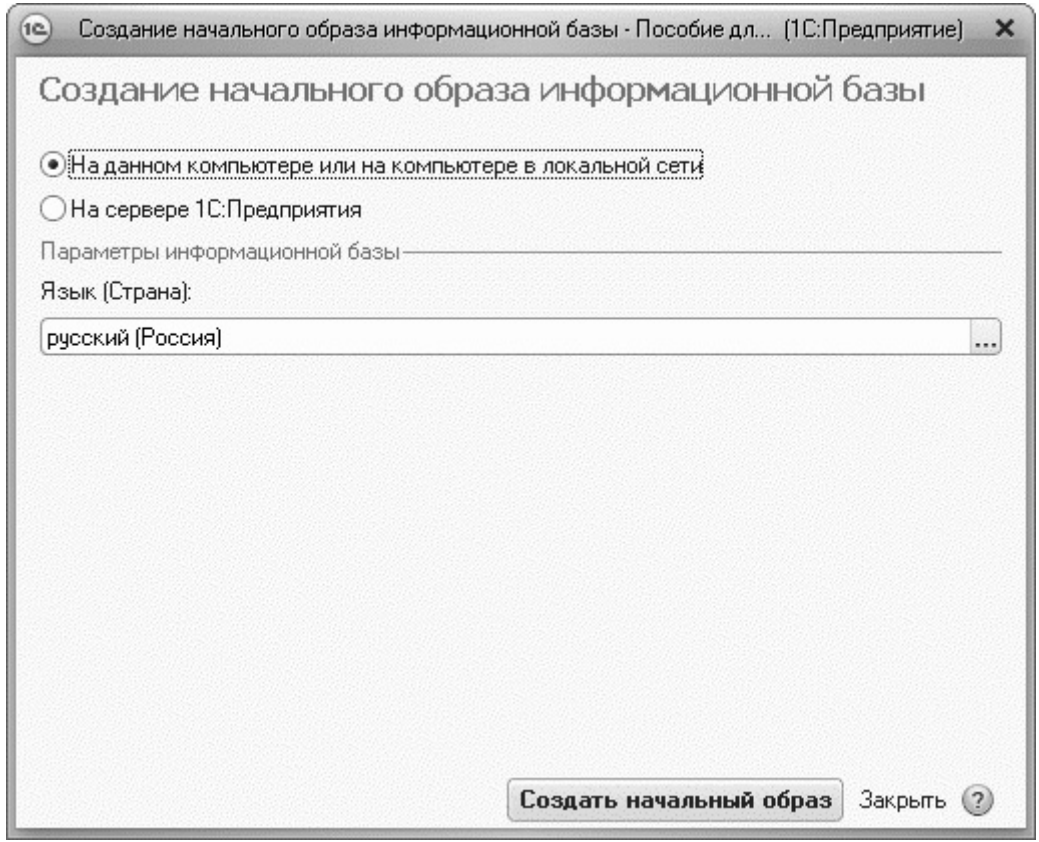


Рис. 24.20. Создание начального образа информационной базы

Нажмем *Создать начальный образ*. На следующем шаге укажем каталог информационной базы и нажмем *Готово*.

Система создаст в указанном каталоге начальный образ информационной базы нашего отделения.

Запуск базы отделения

Теперь перейдем к базе отделения.

Запустим «1С:Предприятие» и добавим в список баз созданную нами базу, расположенную в каталоге, в который мы поместили начальный образ информационной базы нашего отделения.

Для этого в окне запуска «1С:Предприятия» нажмем кнопку *Добавить* и выберем *Добавление существующей информационной базы*.

Нажмем *Далее*, затем укажем наименование информационной базы, например, *база Отделения*. Нажмем *Далее*, затем укажем каталог информационной базы, например, *D:\1C_8.2\Отделение* и нажмем *Готово*.

В режиме «Конфигуратор»

Откроем созданную нами конфигурацию *база Отделения* в режиме *Конфигуратор*.

Выполним команду главного меню *Конфигурация* > *Открыть конфигурацию*. Мы видим, что конфигурация нашего отделения стала защищенной от изменений средствами управления распределенной информационной базой (рис. 24.21).

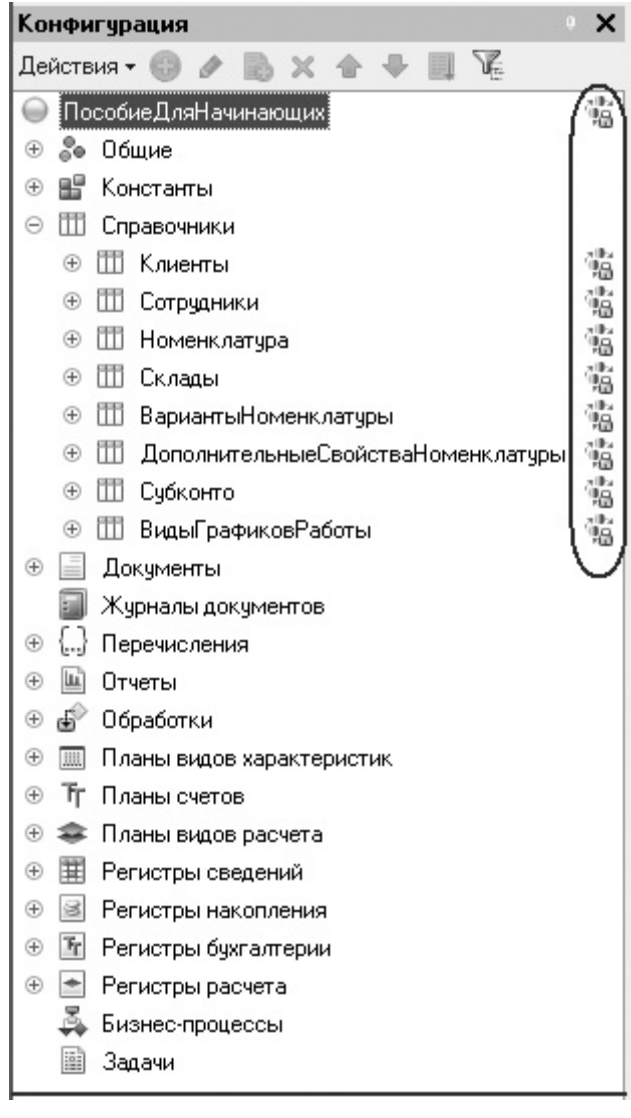

Конфигурация
Действия
ПособиеДляНачинающих
Общие
Константы
Справочники
Клиенты
Сотрудники
Номенклатура
Склады
ВариантыНоменклатуры
ДополнительныеСвойстваНоменклатуры
Субконто
ВидыГрафиковРаботы
Документы
Журналы документов
Перечисления
Отчеты
Обработки
Планы видов характеристик
Планы счетов
Планы видов расчета
Регистры сведений
Регистры накопления
Регистры бухгалтерии
Регистры расчета
Бизнес-процессы
Задачи

Рис. 24.21. Конфигурация подчиненного узла защищена от изменений средствами управления распределенной информационной базой

Выполним команду главного меню *Администрирование* > *Пользователи* и создадим в конфигурации филиала одного пользователя – *Администратор* с ролью *Администратор*.

В режиме «1С:Предприятие»

Запустим базу отделения в режиме отладки и откроем план обмена *Отделения* (рис. 24.22).

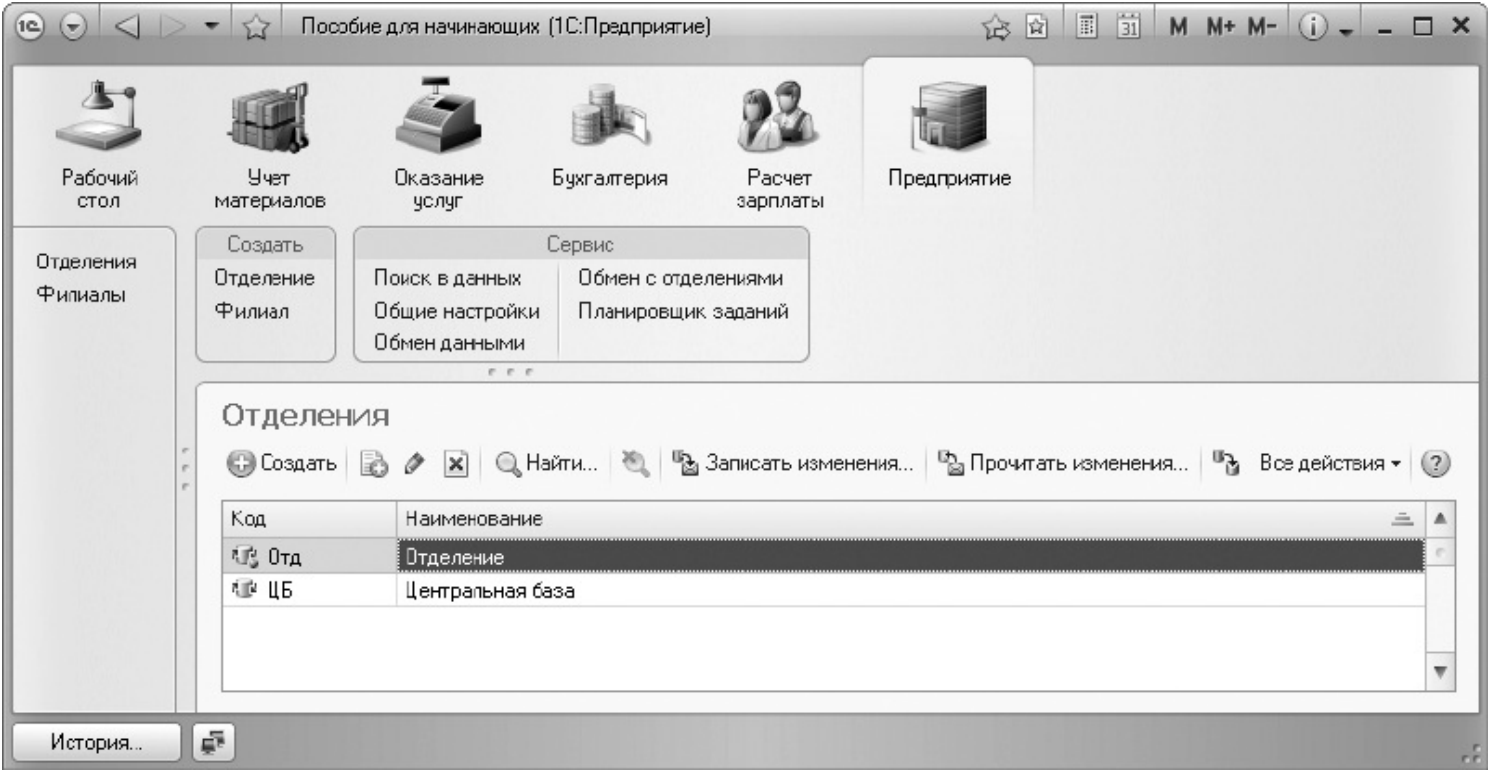


Рис. 24.22. Узлы плана обмена «Отделения»

Обратите внимание, что в базе подчиненного узла сам подчиненный узел является предопределенным узлом плана обмена, а узел центральной базы отмечен красной пиктограммой, указывающей на то, что он является главным для информационной базы отделения. Кроме этого, для узла центральной базы доступны только команды *Записать изменения* и *Прочитать изменения*.

Теперь проверим работу обмена данными.

Откроем список констант и зададим значение константы *ПрефиксНумерации* – *ОТ*. После этого откроем справочник клиентов и добавим в него нового клиента. Затем выполним запись изменений для центральной базы (указав имя файла сообщения). Перейдем в центральную базу и выполним чтение изменений для центральной базы (выбрав имя файла сообщения). Убедимся, что новый клиент, созданный в базе отделения, присутствует и в центральной базе.

Теперь посмотрим, как будут переноситься изменения конфигурации между главным и подчиненным узлами. В конфигураторе центральной базы создадим новую константу с именем *НоваяКонстанта*. Выполним обновление конфигурации базы данных и запустим «1С:Предприятие» в режиме отладки. Откроем план обмена *Отделения* и выполним запись изменений для подчиненного узла *Отделение*. После этого закроем конфигуратор информационной базы отделения, запустим эту базу в режиме *1С:Предприятие* и выполним чтение изменений в базе подчиненного узла.

По окончании чтения система выдаст следующее сообщение (рис. 24.23).

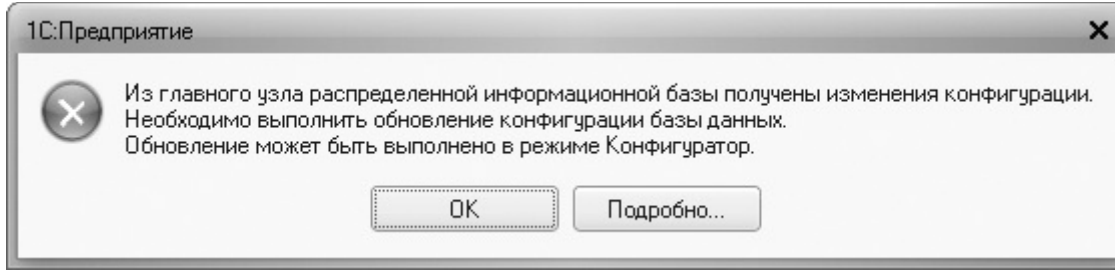


Рис. 24.23. Системное сообщение

Откроем конфигуратор базы отделения и увидим, что в основной конфигурации появилась новая константа *НоваяКонстанта*, то есть изменения, внесенные в конфигурацию центральной базы, были автоматически перенесены в конфигурацию подчиненного узла.

Теперь единственное, что остается сделать, – это выполнить обновление конфигурации базы данных в подчиненном узле.

Следует сказать несколько слов о порядке принятия изменений, когда в одном сообщении получены как изменения конфигурации, так и изменения данных.

В этом случае сначала будет изменена основная конфигурация и выдано сообщение о необходимости выполнения сохранения конфигурации базы данных. После объединения конфигураций следует выполнить повторное получение данных, при котором будут приняты уже изменения данных, содержащиеся в сообщении. Такой порядок принятия изменений не зависит от

того, относятся измененные данные к существующим объектам конфигурации или к новым.

В заключение удалим объект *НоваяКонстанта* из дерева объектов нашей конфигурации.

Программный обмен

Все описанные выше действия по обмену данными в распределенной информационной базе можно выполнить программно.

Мы создадим обработку, которая будет программно выполнять для выбранного узла все те действия, которые были рассмотрены в предыдущем разделе.

В режиме «Конфигуратор»

В конфигураторе центральной базы создадим новый объект конфигурации *Обработка* с именем *ОбменСОтделениями*.

На закладке *Формы* создадим основную форму обработки. В окне редактора форм на закладке *Реквизиты* добавим реквизит формы *ПолеВводаОтделение* с типом *ПланОбменаСсылка.Отделения* и перетащим его в окно элементов формы.

В открывшейся палитре свойств этого поля зададим заголовок – *Отделение*, вид поля (*Поле ввода)* оставим по умолчанию (рис. 24.24).

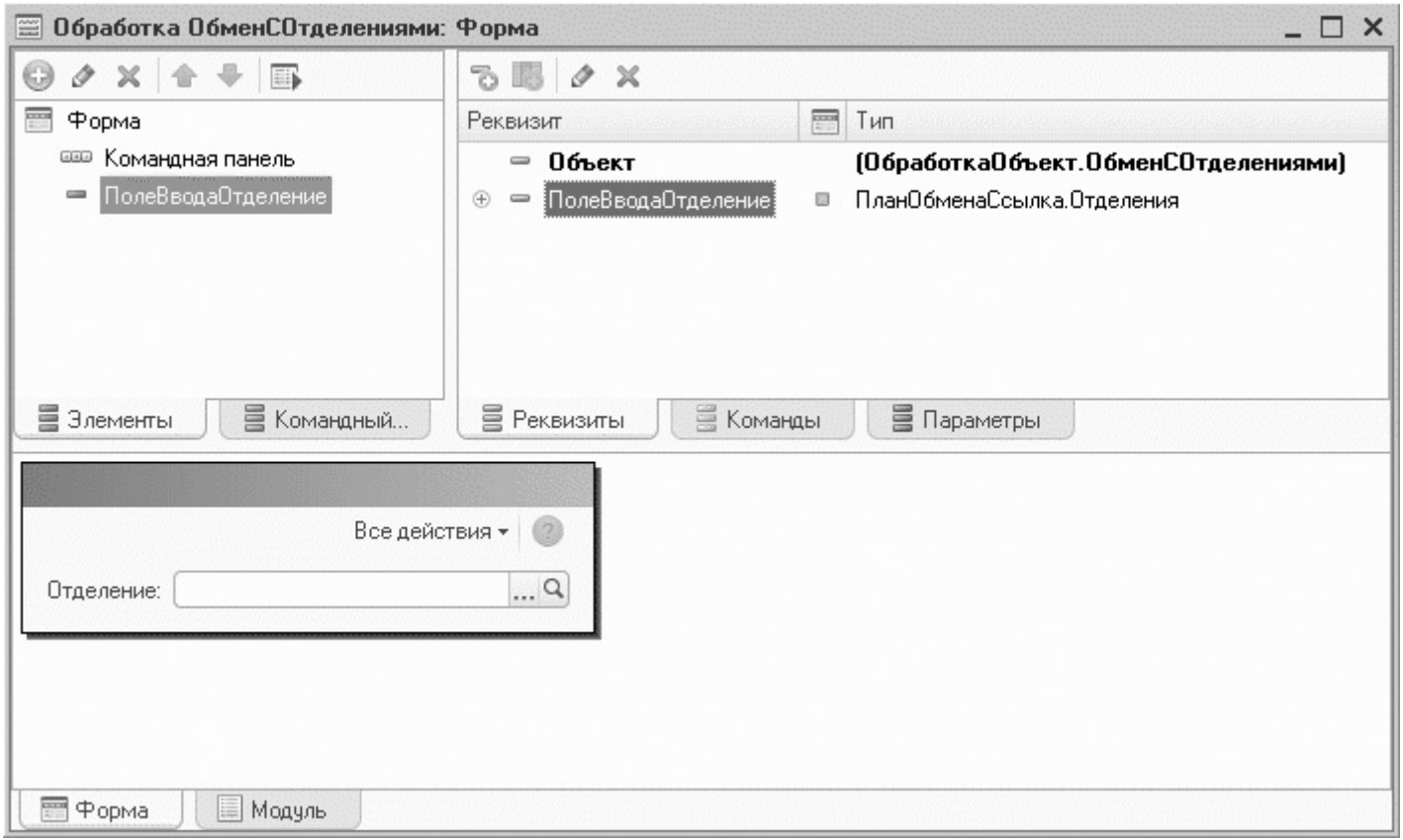


Рис. 24.24. Редактирование формы обработки

Затем на закладке *Команды* поочередно создадим команды *СоздатьНачальныйОбраз, ЗаписатьИзменения* и *ПрочитатьИзменения*. Нажмем кнопку открытия в строке *Действие* для каждой команды.

Шаблоны обработчиков событий пока заполнять не будем, а перейдем на закладку *Форма* и поочередно перетащим эти команды в окно элементов формы.

В результате форма обработки примет вид (рис. 24.25).

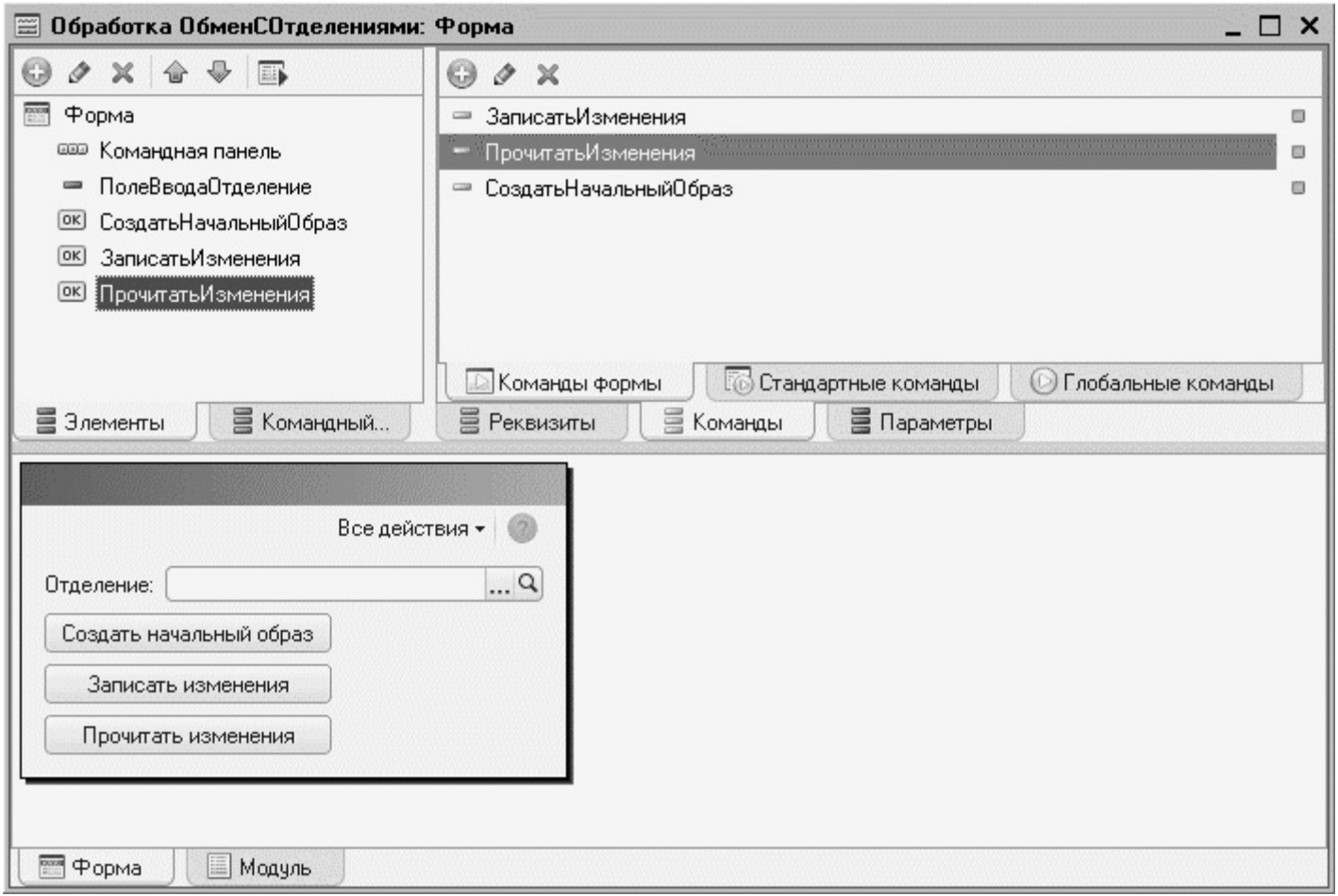


Рис. 24.25. Форма обработки

Откроем палитру свойств кнопки *СоздатьНачальныйОбраз* и снимем флажок у

свойства *Доступность*.

Таким образом, при открытии обработки кнопка будет недоступной, пока не выбран узел плана обмена в поле *ПолеВводаОтделение*. Эта кнопка также будет недоступной в случае выбора предопределенного узла нашей информационной базы, то есть создание начального образа невозможно, если выбранный узел является предопределенным.

Чтобы обеспечить такое поведение кнопки, создадим в модуле формы обработки функцию, выполняющуюся на сервере и возвращающую истину, если переданный в функцию узел является предопределенным (листинг 24.25).

Листинг 24.25. Функция «ПредопределенныйУзел()»

```
&НаСервереБезКонтекста
Функция ПредопределенныйУзел(Узел)

  Возврат Узел = ПланыОбмена.Отделения.ЭтотУзел();

КонецФункции
```

Затем в окне элементов формы выделим элемент *ПолеВводаОтделение*, вызовем его палитру свойств и создадим обработчик события *ОбработкаВыбора*.

Заполним обработчик следующим образом (листинг 24.26).

Листинг 24.26. Процедура «ПолеВводаОтделениеОбработкаВыбора()»

```
&НаКлиенте
Процедура ПолеВводаОтделениеОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)

  Если ПредопределенныйУзел(ВыбранноеЗначение)Тогда
    Элементы.СоздатьНачальныйОбраз.Доступность = Ложь;

  Иначе
    Элементы.СоздатьНачальныйОбраз.Доступность = Истина;

  КонецЕсли;

КонецПроцедуры
```

В этой процедуре доступность кнопки *СоздатьНачальныйОбраз* определяется в зависимости от значения функции *ПредопределенныйУзел()*, в которую передается ссылка на выбранный узел (*ВыбранноеЗначение*).

Теперь заполним обработчик команды *СоздатьНачальныйОбраз* следующим образом (листинг 24.27).

Листинг 24.27. Обработчик нажатия кнопки «Создать начальный образ»

```
&НаКлиенте
Процедура СоздатьНачальныйОбраз()

  Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.ВыборКаталога);
  Диалог.Заголовок = "Укажите каталог информационной базы:";
  Если Диалог.Выбрать() Тогда
    СоздатьНачальныйОбразНаСервере(ПолеВводаОтделение, Диалог.Каталог);
    Предупреждение("Создание начального образа узла завершено.");

  КонецЕсли;

КонецПроцедуры
```

В начале процедуры мы вызываем диалог выбора каталога, в который будет помещен образ информационной базы, и затем вызываем процедуру *СоздатьНачальныйОбразНаСервере()*, исполняющуюся на сервере, в которой вызывается метод *СоздатьНачальныйОбраз()* объекта *ПланыОбменаМенеджер*.

Именно этот метод и позволяет нам создать образ подчиненного узла распределенной информационной базы. В первом параметре метода передается ссылка на узел (реквизит формы *ПолеВводаОтделение*), для которого мы хотим создать начальный образ, а во втором – строка соединения, указывающая информационную базу (листинг 24.28).

Листинг 24.28. Процедура «СоздатьНачальныйОбразНаСервере»

```
&НаСервереБезКонтекста
Процедура СоздатьНачальныйОбразНаСервере(Узел, КаталогСоединения)

  ПланыОбмена.СоздатьНачальныйОбраз(Узел, "File =" + КаталогСоединения);

КонецПроцедуры
```

Теперь создадим обработчик команды *Записать изменения*.

Текст обработчика будет выглядеть следующим образом (листинг 24.29).

Листинг 24.29. Обработчик нажатия кнопки «Записать изменения»

```
&НаКлиенте
Процедура ЗаписатьИзменения()

  Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
  Диалог.Заголовок = "Укажите файл обмена:";
  Если Диалог.Выбрать() Тогда
    ЗаписатьИзмененияНаСервере(ПолеВводаОтделение , Диалог.ПолноеИмяФайла);
    Предупреждение("Запись изменений завершена.");

  КонецЕсли;

КонецПроцедуры
```

В начале процедуры мы вызываем диалог ввода имени файла, в который будут записаны изменения, и затем вызываем процедуру *ЗаписатьИзмененияНаСервере()*, исполняющуюся на сервере. В первом параметре метода передается ссылка на узел (реквизит формы *ПолеВводаОтделение*), для которого будет производиться запись изменений.

В этой процедуре мы создаем объект *ЗаписьXML* для работы с этим файлом.

Затем создаем объект *ЗаписьСообщенияОбмена*, с помощью которого будем делать сообщение обмена. В методе *НачатьЗапись()*, во втором параметре, мы указываем, для какого узла обмена будет создаваться это сообщение.

После этого мы выполняем метод *ЗаписатьИзменения()* объекта *ПланыОбменаМенеджер*, который и записывает изменения, предназначенные для передачи в выбранный узел, в указанное сообщение обмена.

В заключение мы, как обычно, заканчиваем запись сообщения обмена и закрываем файл (листинг 24.30).

Листинг 24.30. Процедура «ЗаписатьИзмененияНаСервере»

```
&НаСервереБезКонтекста
Процедура ЗаписатьИзмененияНаСервере(Узел, ИмяФайла)
```

```
    // Создать и проинициализировать объект ЗаписьXML.
    ЗаписьXML = Новый ЗаписьXML;
    ЗаписьXML.ОткрытьФайл(ИмяФайла);

    // Создать объект ЗаписьСообщенияОбмена и начать запись сообщения.
    ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
    ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

    // Записать содержимое тела сообщения обмена данными распределенной ИБ.
    ПланыОбмена.ЗаписатьИзменения(ЗаписьСообщения);

    // Закончить запись сообщения и запись XML.
    ЗаписьСообщения.ЗакончитьЗапись();
    ЗаписьXML.Закрыть();

КонецПроцедуры
```

Узнай больше!

Следует отметить, что метод «ЗаписатьИзменения()» позволяет задать максимальное число элементов данных, которые помещаются в сообщение в рамках одной транзакции базы данных. По умолчанию все данные помещаются в сообщение в рамках одной транзакции.

Такой режим является рекомендуемым, так как гарантирует согласованность данных, помещаемых в сообщение.

Но при создании сообщения в многопользовательском режиме могут быть конфликты блокировок между транзакцией, в которой данные помещаются в сообщение, и транзакциями, выполняемыми другими пользователями. Для снижения вероятности возникновения таких конфликтов можно задать значение этого параметра, отличное от значения по умолчанию. Чем меньше значение параметра, тем меньше вероятность конфликта блокировок, но выше вероятность помещения в сообщение несогласованных данных.

Учитывая все вышесказанное, идеальным вариантом является выполнение обмена данными в монопольном режиме. Однако такой вариант не всегда приемлем в силу специфики организации работы конкретных информационных баз.

И последним мы создадим обработчик команды *Прочитать изменения*.

Текст обработчика будет выглядеть следующим образом (листинг 24.31).

Листинг 24.31. Обработчик нажатия кнопки «Прочитать изменения»

```
&НаКлиенте
Процедура ПрочитатьИзменения()

  Диалог = Новый ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
  Диалог.Заголовок = "Укажите файл обмена:";
```

```
  Если Диалог.Выбрать() Тогда
    ПрочитатьИзмененияНаСервере(Диалог.ПолноеИмяФайла);
    Предупреждение("Чтение изменений завершено.");

  КонецЕсли;

КонецПроцедуры
```

В начале процедуры мы снова вызываем диалог ввода имени файла, который будет прочитан, и затем вызываем процедуру *ПрочитатьИзмененияНаСервере()*, исполняющуюся на сервере.

В этой процедуре мы создаем объект *ЧтениеXML* для работы с этим файлом. Затем создаем объект *ЧтениеСообщенияОбмена* для чтения сообщения, содержащегося в указанном файле.

Затем методом *ПрочитатьИзменения()* объекта *ПланыОбменаМенеджер* мы читаем полученное сообщение.

В заключение процедуры мы завершаем чтение сообщения обмена и закрываем файл (листинг 24.32).

Листинг 24.32. Процедура «ПрочитатьИзмененияНаСервере»

```
&НаСервереБезКонтекста
```

```
Процедура ПрочитатьИзмененияНаСервере(ИмяФайла)

  // Создать и проинициализировать объект ЧтениеXML.
  ЧтениеXML = Новый ЧтениеXML;
  ЧтениеXML.ОткрытьФайл(ИмяФайла);

  // Создать объект ЧтениеСообщенияОбмена и начать чтение сообщения.
  ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
  ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

  // Прочитать содержимое тела сообщения.
  ПланыОбмена.ПрочитатьИзменения(ЧтениеСообщения);

  // Закончить чтение сообщения и чтение XML.
  ЧтениеСообщения.ЗакончитьЧтение();
  ЧтениеXML.Закрыть();

КонецПроцедуры
```

В заключение на закладке *Подсистемы* укажем принадлежность обработки *ОбменСОтделениями* к подсистеме *Предприятие*.

А также в окне редактирования командного интерфейса этой подсистемы (*Все подсистемы*) установим следующий порядок следования команд в группе панели действий *Сервис*:

- *Поиск в данных,*

- *Общие настройки,*
- *Обмен данными,*
- *Обмен с отделениями,*
- *Планировщик заданий.*

Проверить работу нашей обработки можно на примере, аналогичном приведенному в разделе универсального обмена данными.

Следует лишь сделать несколько заключительных замечаний.

При использовании механизма распределенных информационных баз становятся доступными четыре события объекта встроенного языка *ПланОбменаОбъект.<имя>*, которые позволяют управлять отправкой и приемом данных на уровне отдельных элементов данных:

- *ПриОтправкеДанныхГлавному(),*
- *ПриОтправкеДанныхПодчиненному(),*
- *ПриПолученииДанныхОтГлавного(),*
- *ПриПолученииДанныхОтПодчиненного().*

Эти события будут вызываться для каждого элемента данных, включаемого в сообщение.

Работу этих событий можно увидеть, добавив в модуль объекта *План обмена Отделения* следующий текст (листинг 24.33).

Листинг 24.33. Просмотр работы событий объекта «ПланОбменаОбъект»

```
Процедура ПриОтправкеДанныхГлавному(ЭлементДанных, ОтправкаЭлемента)

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "ПриОтправкеДанныхГлавному " + ЭлементДанных;
  Сообщение.Сообщить();

КонецПроцедуры

Процедура ПриОтправкеДанныхПодчиненному(ЭлементДанных, ОтправкаЭлемента)

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "ПриОтправкеДанныхПодчиненному " + ЭлементДанных;
  Сообщение.Сообщить();

КонецПроцедуры

Процедура ПриПолученииДанныхОтГлавного(ЭлементДанных, ПолучениеЭлемента,
ОтправкаНазад)

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "ПриПолученииДанныхОтГлавного " + ЭлементДанных;
  Сообщение.Сообщить();

КонецПроцедуры
```

```
Процедура ПриПолученииДанныхОтПодчиненного(ЭлементДанных, ПолучениеЭлемента,
ОтправкаНазад)

  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = "ПриПолученииДанныхОтПодчиненного " + ЭлементДанных;
  Сообщение.Сообщить();

КонецПроцедуры
```

В первом параметре всех перечисленных событий находится тот элемент данных, для которого вызвано это событие.

Параметр *ОтправкаЭлемента* позволяет управлять тем, какая информация будет помещена в сообщение. Он может принимать три значения:

- *Авто* – значение по умолчанию. Указывает на то, что элемент данных будет помещен в сообщение;
- *Удалить* – в сообщение будет помещено значение, предназначенное для удаления этого элемента данных;
- *Игнорировать* – в сообщение не будет помещено ничего, связанного с этим элементом данных.

Параметр *ПолучениеЭлемента* позволяет указать, будет ли прочитанный элемент данных записан в базу данных или нет. Параметр также может

принимать три значения:

- *Авто* – значение по умолчанию. Если элемент данных получен от главного узла, он будет записан всегда. Если элемент данных получен от подчиненного узла, он будет записан, только если не зарегистрированы изменения для этого элемента данных;
- *Принять* – полученный элемент данных будет записан всегда;
- *Игнорировать* – проигнорировать получение элемента данных и ничего не записывать.

Также в событиях получения данных существует третий параметр – *ОтправкаНазад*, имеющий тип *Булево*.

Этот параметр позволяет выполнять принудительную регистрацию изменений для полученного элемента данных в базе-получателе.

Такая необходимость может возникнуть, например, когда при приеме данных от узла-отправителя обнаружено, что полученные данные противоречивы (например, в узле-отправителе была допущена ошибка при изменении данных).

Тогда мы можем проигнорировать присланные изменения и, установив флажок *ОтправкаНазад*, вызвать принудительную регистрацию изменений полученного элемента данных в нашей базе для узла-отправителя.

В результате последующего обмена состояние этого элемента данных в узле-отправителе будет установлено таким же, как и в нашей базе.

Изменение структуры узлов

В заключение следует сказать о том, что механизм распределенных информационных баз содержит программное средство реконфигурирования структуры узлов распределенной базы.

Для этого следует использовать метод *УстановитьГлавныйУзел()* объекта *ПланыОбменаМенеджер*.

В параметре этого метода передается ссылка на узел плана обмена распределенной информационной базы, который устанавливается главным для текущей базы. Также в этом параметре может быть передано значение *Неопределено*, и это приведет к тому, что у текущей информационной базы будет отсутствовать главный узел.

Допустим, необходимо переместить один из подчиненных узлов в корень дерева (рис. 24.26).

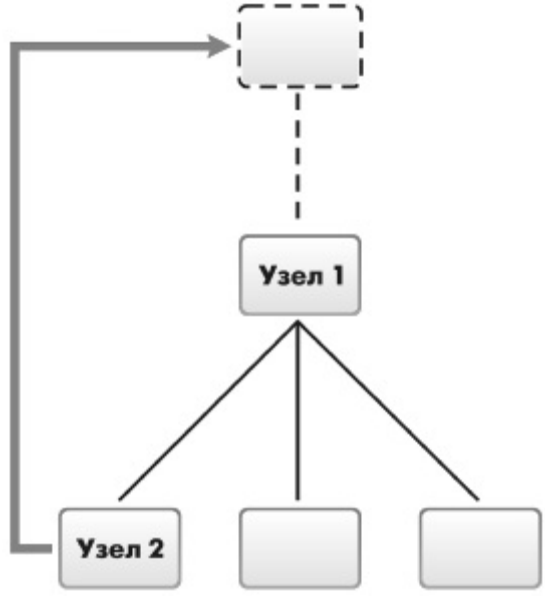


Рис. 24.26. Реконфигурирование структуры узлов

Для этого следует выполнить следующие действия (листинг 24.34).

Листинг 24.34. Перемещение Узла2 в корень дерева

```
// В информационной базе Узла2.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Неопределено);

// В информационной базе Узла1.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Узел2);
```

При этом будут удалены все записи регистрации изменений конфигурации Узла1, относящиеся к Узлу2, т. к. передача изменений конфигурации будет возможна теперь только от Узла2 к Узлу1. Записи регистрации изменения данных удалены не будут, так как передача изменений данных будет по-прежнему возможна между этими узлами.

Таким же образом, используя значение параметра метода *Неопределено*, мы можем отключать от дерева отдельную информационную базу или целое поддерево (рис. 24.27, листинг 24.35).

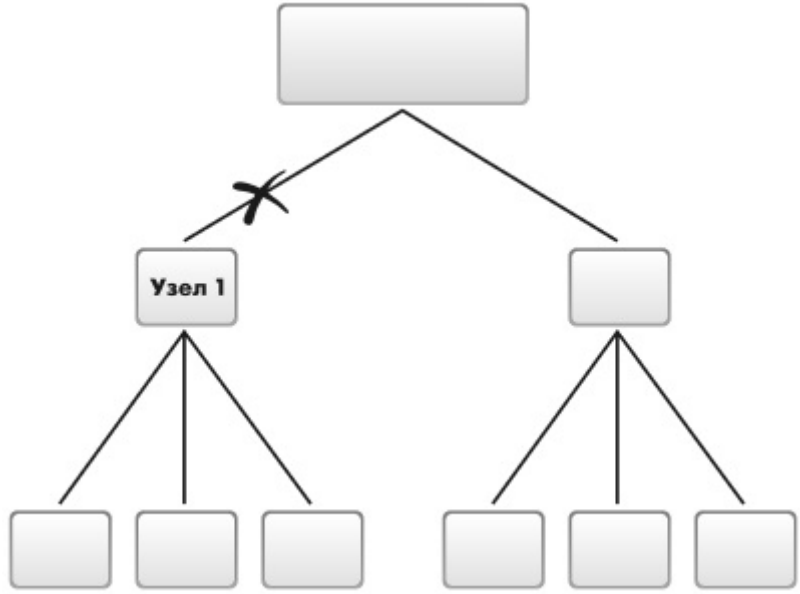


Рис. 24.27. Отключение поддерева от распределенной информационной базы

Листинг 24.35. Перемещение Узла2 в корень дерева

```
// В информационной базе Узла1.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Неопределено);
```

Кроме этого, мы можем создавать распределенную информационную базу из отдельных информационных баз с идентичной конфигурацией (рис. 24.28, листинг 24.36).

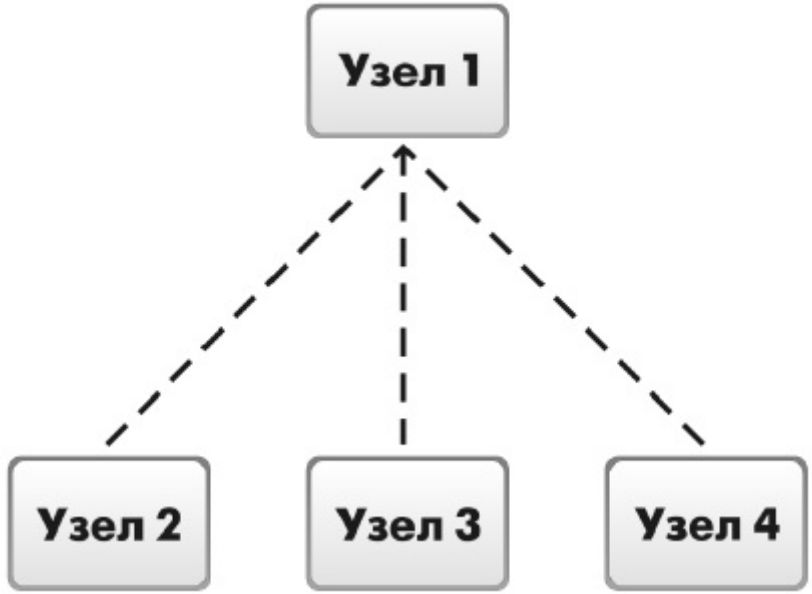


Рис. 24.28. Информационная база из отдельных информационных баз с идентичной конфигурацией

Листинг 24.36. Создание распределенной информационной базы из баз с идентичной конфигурацией

```
// В информационных базах Узла2, Узла3 и Узла4.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Узел1);
```

Контрольные вопросы

- *Какие средства входят в состав механизма универсального обмена данными.*
- *Для чего предназначен объект конфигурации «План обмена».*
- *Каковы основные составляющие плана обмена.*
- *Что такое узлы плана обмена.*
- *Что такое состав плана обмена и для каких элементов данных возможен обмен данными.*
- *Что такое авторегистрация.*
- *Для чего предназначен механизм регистрации изменений.*
- *Как работает инфраструктура сообщений.*
- *Каково назначение XML-сериализации.*
- *Для чего используется запись/чтение документов XML.*
- *Как создать план обмена.*

- *Как настроить конфигурацию для обмена данными.*
- *Как реализовать обмен данными в общем виде.*
- *Как реализовать обмен данными в распределенной информационной базе.*
- *Как программно управлять обменом данными в распределенной информационной базе.*
- *Как изменить структуру дерева распределенной информационной базы.*

Занятие 25 (0:30). Функциональные опции

Продолжительность

Ориентировочная продолжительность занятия – 30 минут.

Вот мы и создали с вами небольшое прикладное решение, которое позволило автоматизировать работу нашей ремонтной фирмы ООО «На все руки мастер». Теперь настало время для одного чудесного превращения.

Дело в том, что наше прикладное решение настолько понравилось сотрудникам ООО «На все руки мастер», что они рассказали о нем своим соседям – косметическому салону «Королева красоты». Сотрудники салона посмотрели, как работает наше прикладное решение, и обратились к нам с просьбой автоматизировать и их салон тоже.

И мы, конечно же, с радостью согласились по одной простой причине: мы с вами создали универсальную конфигурацию, подходящую для автоматизации практически любой деятельности, связанной с оказанием услуг.

Все, что нам осталось теперь сделать, чтобы наша конфигурация смогла работать в косметическом салоне, – просто создать новую информационную базу с нашей конфигурацией и заполнить ее новыми данными – сотрудниками,

новой номенклатурой и т. п. Все механизмы учета, которые мы с вами создавали, не были привязаны к какой-либо специфике конкретного предприятия, а потому могут с успехом использоваться на любом другом предприятии, имеющем аналогичную область деятельности.

Таким образом, даже если в косметическом салоне пожелают иметь дополнительную функциональность, нам потребуется всего лишь дописать несколько модулей к нашей конфигурации, что гораздо эффективнее, чем создавать заново индивидуальное решение только для данного предприятия.

В то же время в косметическом салоне может не потребоваться какая-то функциональность, которая уже есть в нашей конфигурации.

Что делать? Удалять ненужные объекты конфигурации, программный код?

Это может оказаться очень трудоемкой операцией. В реальных конфигурациях может присутствовать большое количество объектов конфигурации. Связи между ними могут быть очень сложными.

Поэтому в «1С:Предприятии 8» существует механизм функциональных опций, который позволяет включать/выключать при внедрении целые блоки функциональности, не изменяя при этом саму конфигурацию.

Функциональные опции позволяют разработчику выделить некоторую часть функциональности прикладного решения, которую можно оперативно включать или выключать на этапе внедрения и/или в процессе работы системы.

Опции «Бухгалтерский учет» и «Расчет зарплаты»

Предположим, что косметический салон по каким-то причинам не ведет бухгалтерский учет и расчет заработной платы. Для отключения соответствующей функциональности мы создадим функциональные опции *БухгалтерскийУчет* и *РасчетЗарплаты*, установим их для соответствующих объектов конфигурации и отключим их в режиме *1С:Предприятие*.

Таким образом, при совершенно одинаковой конфигурации в прикладном решении косметического салона не будет видно никаких действий, связанных с расчетом зарплаты и ведением бухучета, как будто их и нет вовсе.

В режиме «Конфигуратор»

Поскольку значения функциональных опций обязательно должны где-то храниться, добавим сначала константы *БухгалтерскийУчет* и *РасчетЗарплаты* с типом *Булево*, в которых будут храниться значения функциональных опций (рис. 25.1).

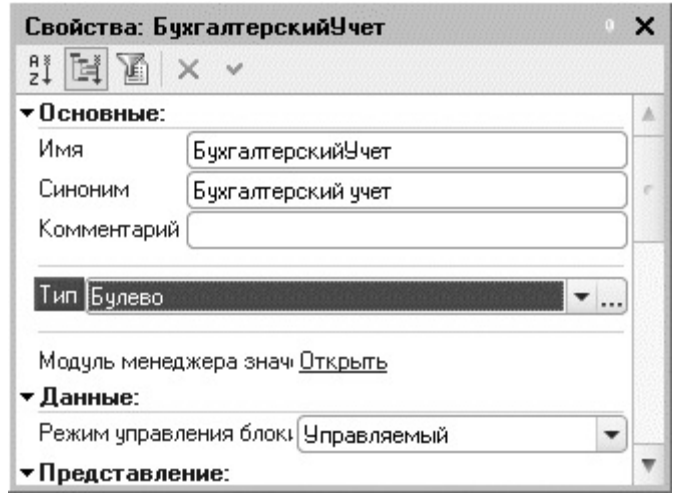


Рис. 25.1. Создание константы

Если значение константы *Истина,* значит, функциональная опция включена. Если значение *Ложь,* функциональная опция выключена.

Затем раскроем ветвь *Общие*, выделим ветвь *Функциональные опции* и создадим функциональные опции *БухгалтерскийУчет* и *РасчетЗарплаты*, указав в свойстве *Хранение* соответствующие константы (рис. 25.2).

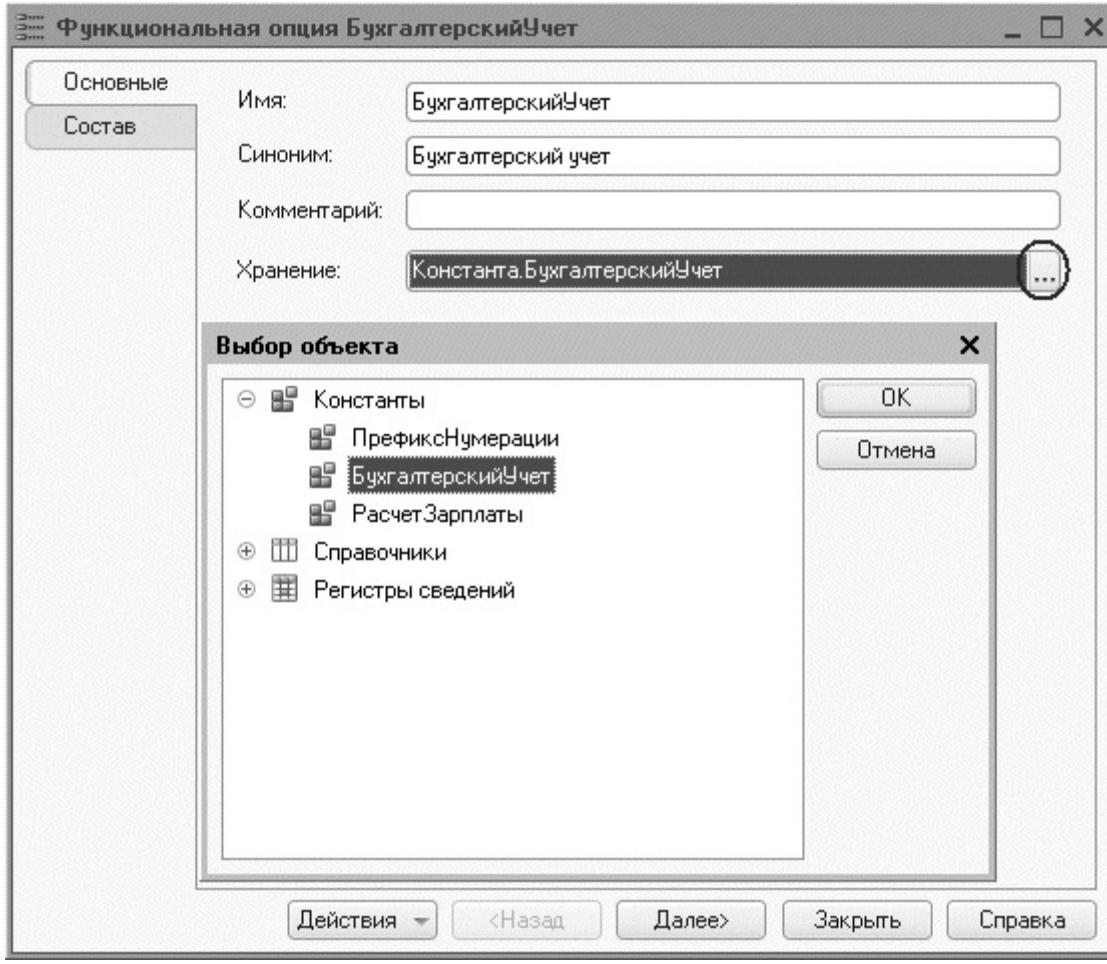


Рис. 25.2. Создание функциональной опции

Теперь нам нужно привязать объекты конфигурации к функциональным опциям.

К ведению бухгалтерского учета в нашей конфигурации относятся следующие объекты:

- справочник *Субконто,*
- документ *ВводНачальныхОстатковНоменклатуры,*
- отчет *ОборотноСальдоваяВедомость,*
- план видов характеристик *ВидыСубконто,*
- план счетов *Основной,*
- регистр бухгалтерии *Управленческий.*

На закладке *Состав* отметим эти объекты для функциональной опции *БухгалтерскийУчет* (рис. 25.3).

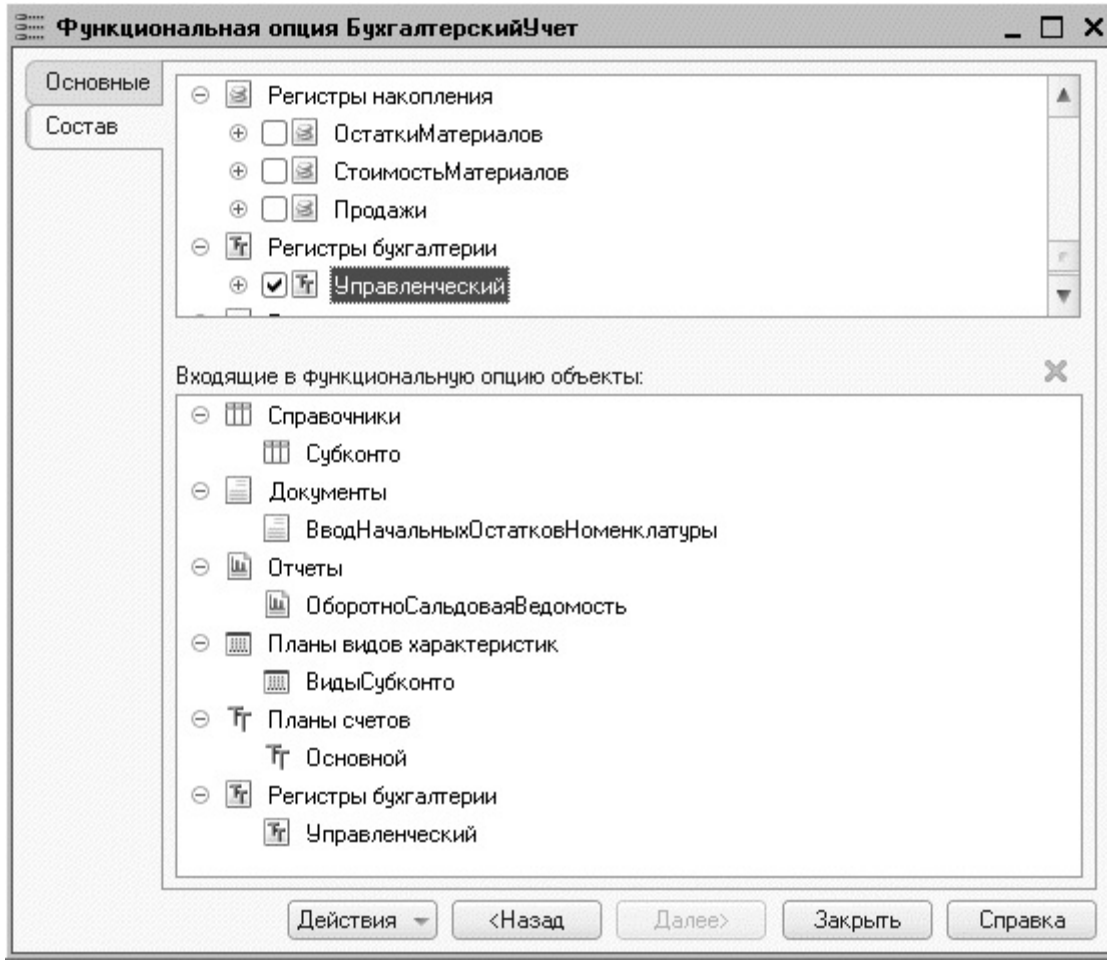


Рис. 25.3. Состав объектов функциональной опции «Бухгалтерский учет»

К расчету заработной платы в нашей конфигурации относятся следующие

объекты:

- справочник *ВидыГрафиковРаботы,*
- документ *НачислениеСотрудникам,*
- отчет *НачислениеСотрудникам,*
- отчет *Перерасчет,*
- отчет *ДиаграммаНачислений,*
- план видов расчета *ОсновныеНачисления,*
- регистр сведений *ГрафикиРаботы,*
- регистр расчета *Начисления.*

На закладке *Состав* отметим эти объекты для функциональной опции *РасчетЗарплаты*.

Теперь, если мы откроем окно редактирования объекта конфигурации Справочник *Субконто* или любого другого объекта конфигурации, входящего в состав функциональной опции *БухгалтерскийУчет*, то эта опция будет включена на закладке *Функциональные опции* окна редактирования этого объекта (рис. 25.4).

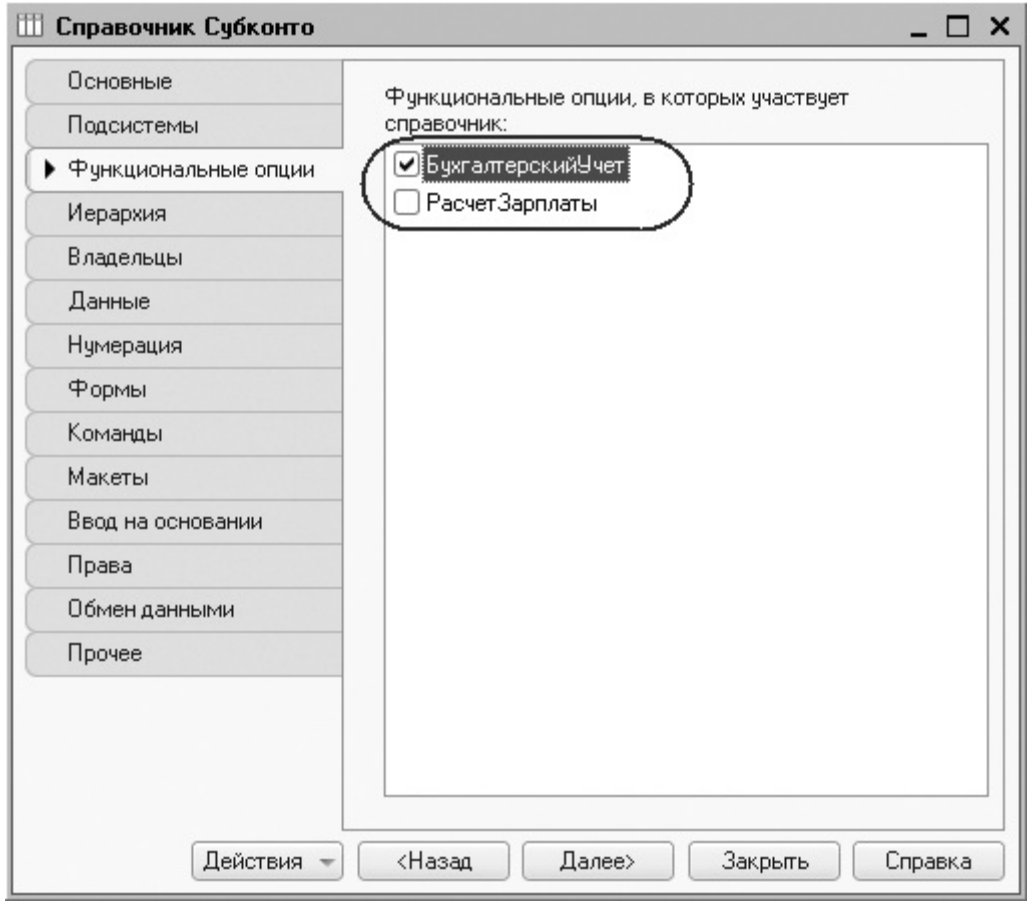


Рис. 25.4. Значение функциональных опций для объектов бухгалтерского учета

Таким образом, отображение объектов конфигурации в интерфейсе приложения зависит от того, включена связанная с ними функциональная опция (значение

соответствующей константы – *Истина*) или нет.

У объектов, относящихся к расчету зарплаты, мы увидим включенной функциональную опцию *РасчетЗарплаты* на закладке *Функциональные опции* окна редактирования этих объектов.

Для остальных объектов конфигурации на закладке *Функциональные опции* ничего не отмечено. Если функциональная опция для объекта выключена, это значит, что данный объект не зависит от значения функциональной опции и отображается всегда (рис. 25.5).

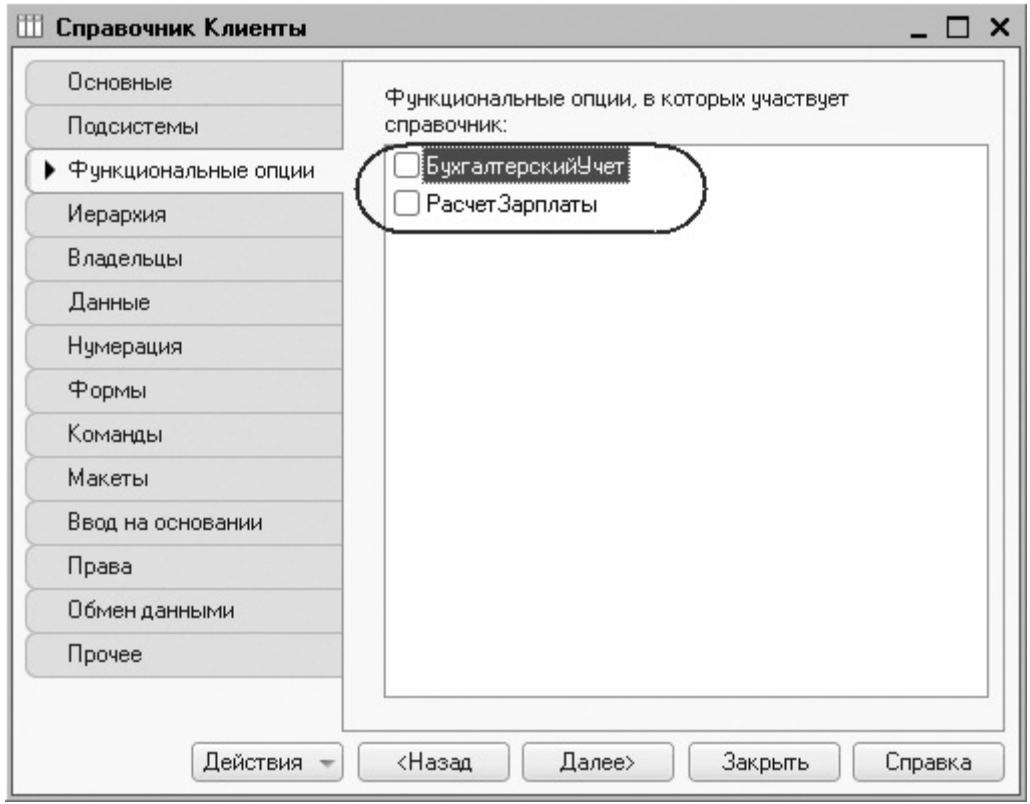


Рис. 25.5. Установка функциональной опции для независимых объектов

ПРИМЕЧАНИЕ

Если включить в состав функциональной опции какую-либо подсистему, то мы вообще не увидим соответствующего раздела в «1С:Предприятии», пока данная функциональная опция отключена.

После этого раскроем ветвь *Общие формы* и откроем общую форму констант. Эту форму с именем *ОбщиеНастройки* мы создали на предыдущем занятии, и она уже содержит константу *ПрефиксНумерации*.

Теперь нам нужно добавить в нее новые константы, чтобы затем в пользовательском режиме открывать форму констант и изменять значение функциональных опций.

На закладке *Реквизиты* этой формы раскроем основной реквизит *НаборКонстант* и перетащим константы *БухгалтерскийУчет* и *РасчетЗарплаты* в окно элементов формы (рис. 25.6).

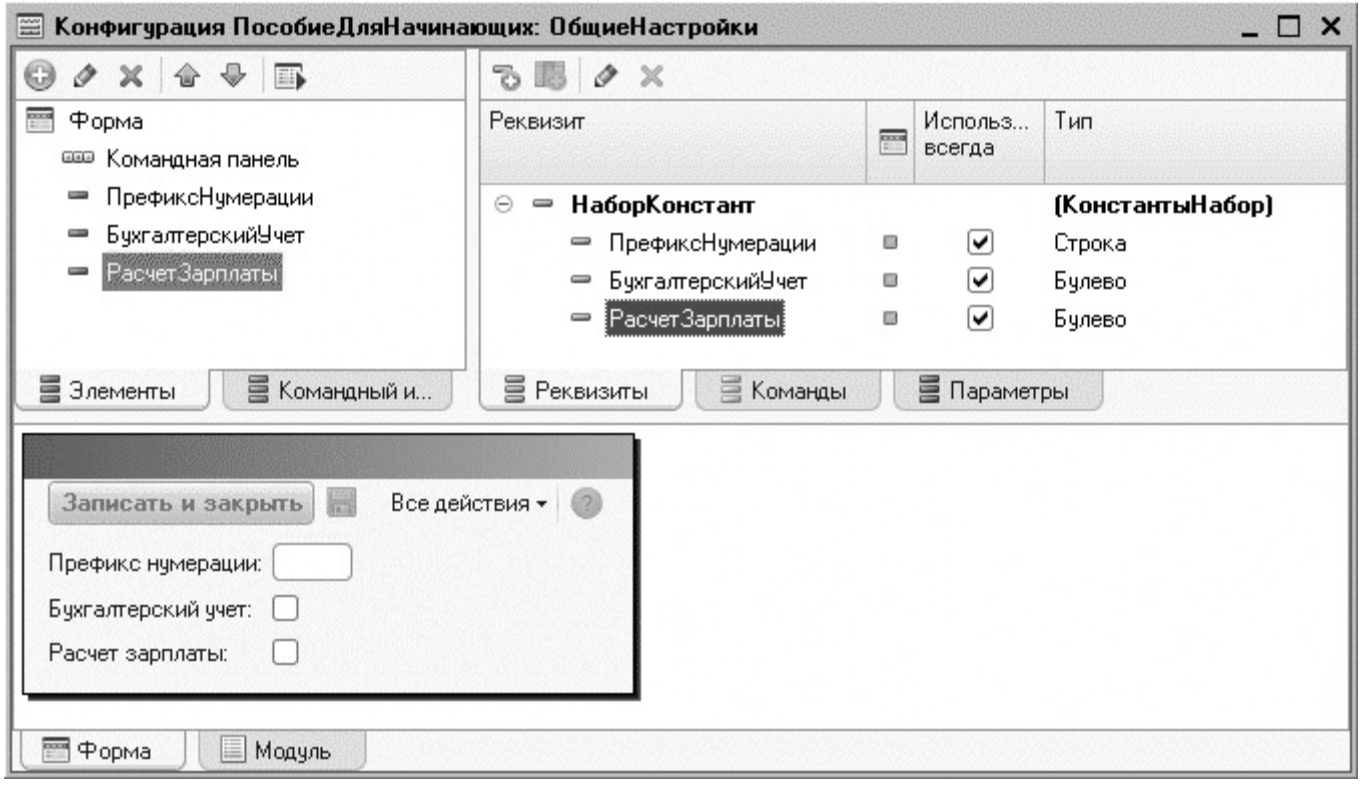


Рис. 25.6. Редактирование общей формы констант

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим работу функциональных опций. В панели действий раздела *Предприятие* выполним команду *Общие настройки*.

В открывшейся форме констант мы видим, что обе константы имеют значение *Ложь* (рис. 25.7).

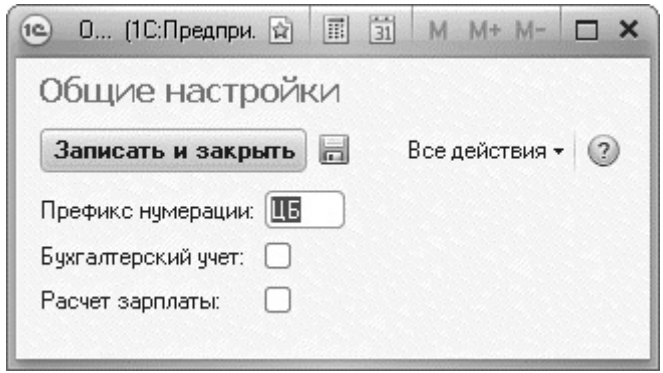


Рис. 25.7. Форма констант

Это значит, что соответствующие функциональные опции отключены.

И действительно, в разделах *Бухгалтерия* и *Расчет зарплаты* мы не видим команд для ведения бухучета и расчета заработной платы (рис. 25.8).

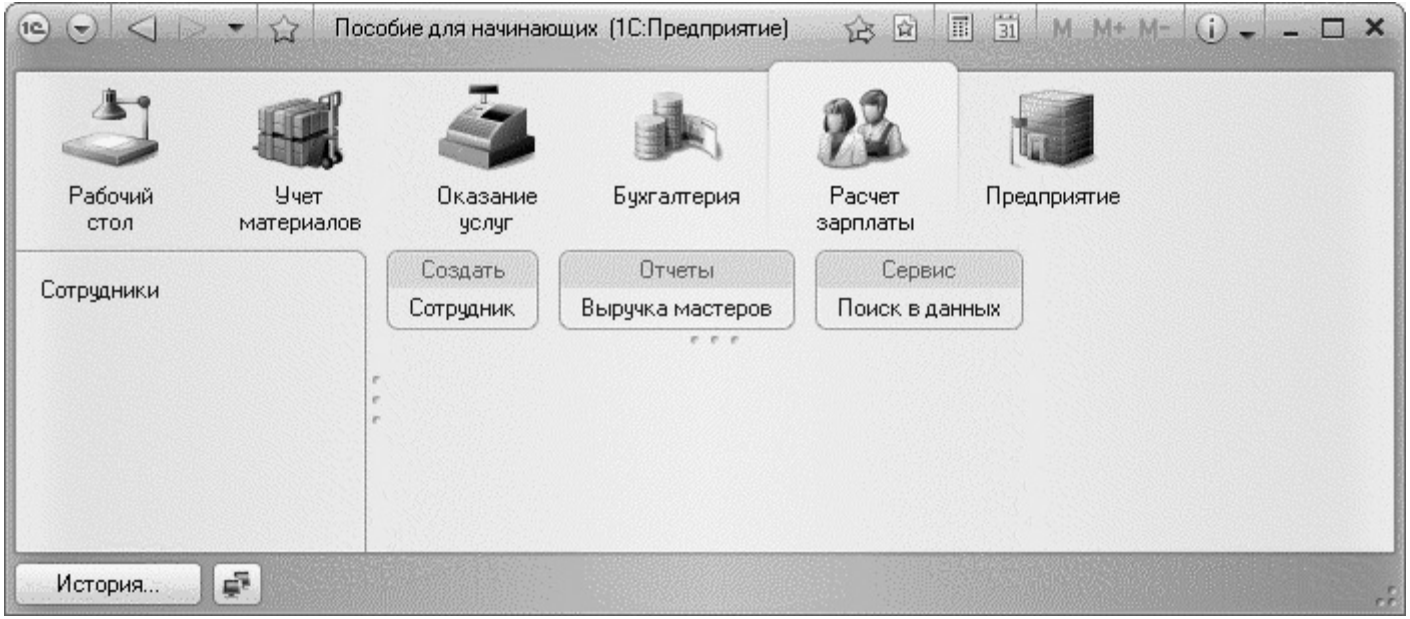


Рис. 25.8. Интерфейс раздела «Расчет зарплаты»

Теперь если через некоторое время руководство косметического салона пожелает вести расчет зарплаты, то администратор включит соответствующую опцию *Расчет зарплаты,* и все!

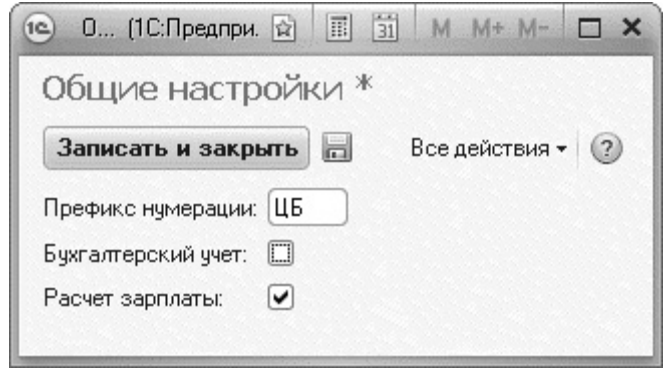


Рис. 25.9. Изменение значения функциональной опции для расчета зарплаты

Нужно только сохранить измененное значение констант и перезапустить «1С:Предприятие», чтобы платформа «отрисовала» новый интерфейс.

В результате раздел *Расчет зарплаты* будет выглядеть следующим образом (рис. 25.10).

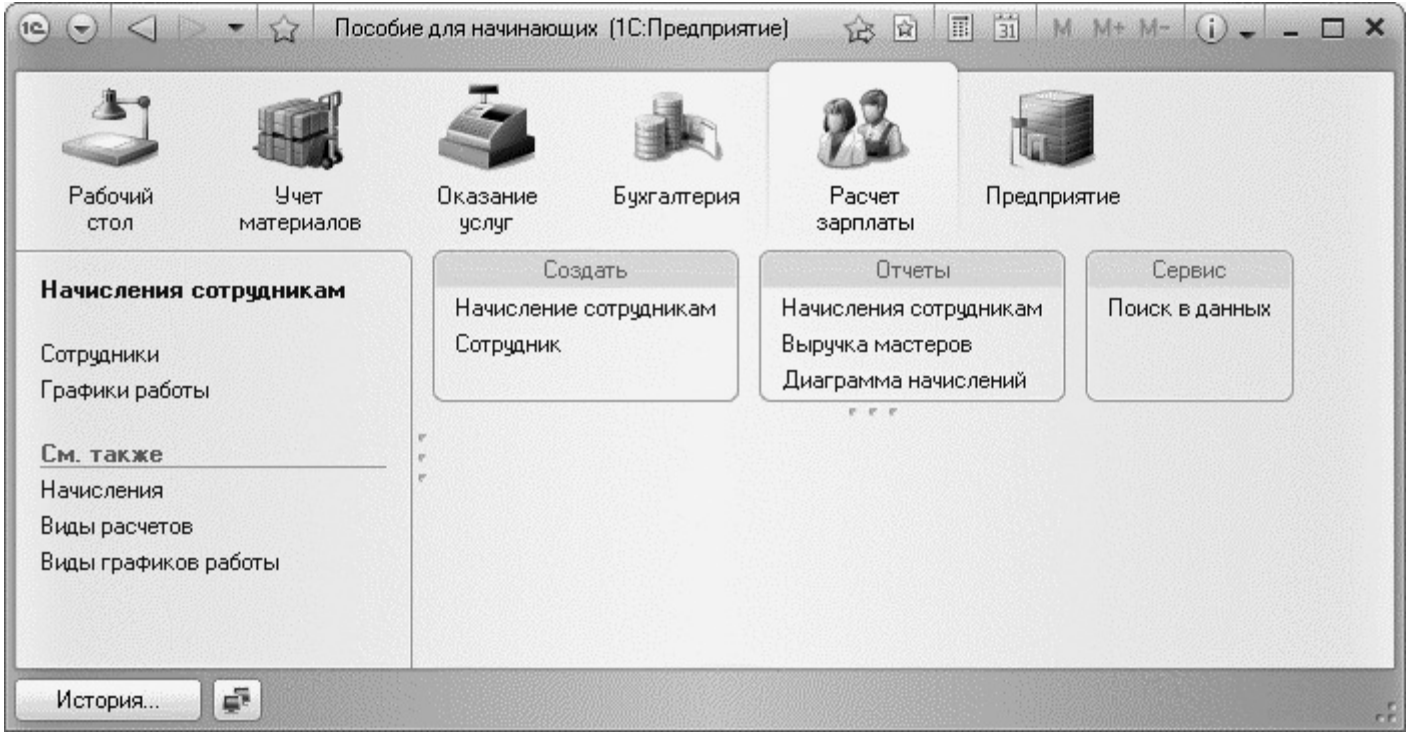


Рис. 25.10. Интерфейс раздела «Расчет зарплаты»

А если мы включим и вторую функциональную опцию *Бухгалтерский учет*, то мы восстановим интерфейс прикладного решения, разработанный нами для фирмы ООО «На все руки мастер».

Вот так быстро и легко происходит настройка прикладного решения под требования заказчика.

Опция «Учет клиентов»

Нужно отметить, что функциональные опции могут влиять не только на командный интерфейс приложения, но и на внешний вид форм, используемых в прикладном решении. Кроме этого, включение/выключение функциональности можно выполнять и без перезапуска клиентского приложения. А если к этому прибавить возможности работы с функциональными опциями во встроенном языке, то становится понятным, что механизм функциональных опций может сделать процесс внедрения и настройки прикладного решения у заказчика простым и понятным даже для неопытного пользователя.

Рассмотрим еще один пример.

«Поименный» учет клиентов при оказании услуг востребован далеко не всегда. Зачастую важен лишь сам факт оказания услуги, при этом «личность» клиента не имеет значения.

Поэтому предусмотрим в нашей конфигурации возможность отключить ведение списка клиентов и избавимся от необходимости указывать клиента каждый раз при оказании услуги.

Также доработаем существующие функциональные опции, включив в них и подсистемы *Бухгалтерия* и *РасчетЗарплаты,* чтобы наше решение

выглядело «законченным». Раз бухгалтерия не нужна, значит ее нет нигде.

В режиме «Конфигуратор»

Откроем состав функциональной опции *БухгалтерскийУчет* и добавим в него подсистему *Бухгалтерия*.

Аналогичным образом добавим в состав функциональной опции *РасчетЗарплаты* подсистему *РасчетЗарплаты* (рис. 25.11).

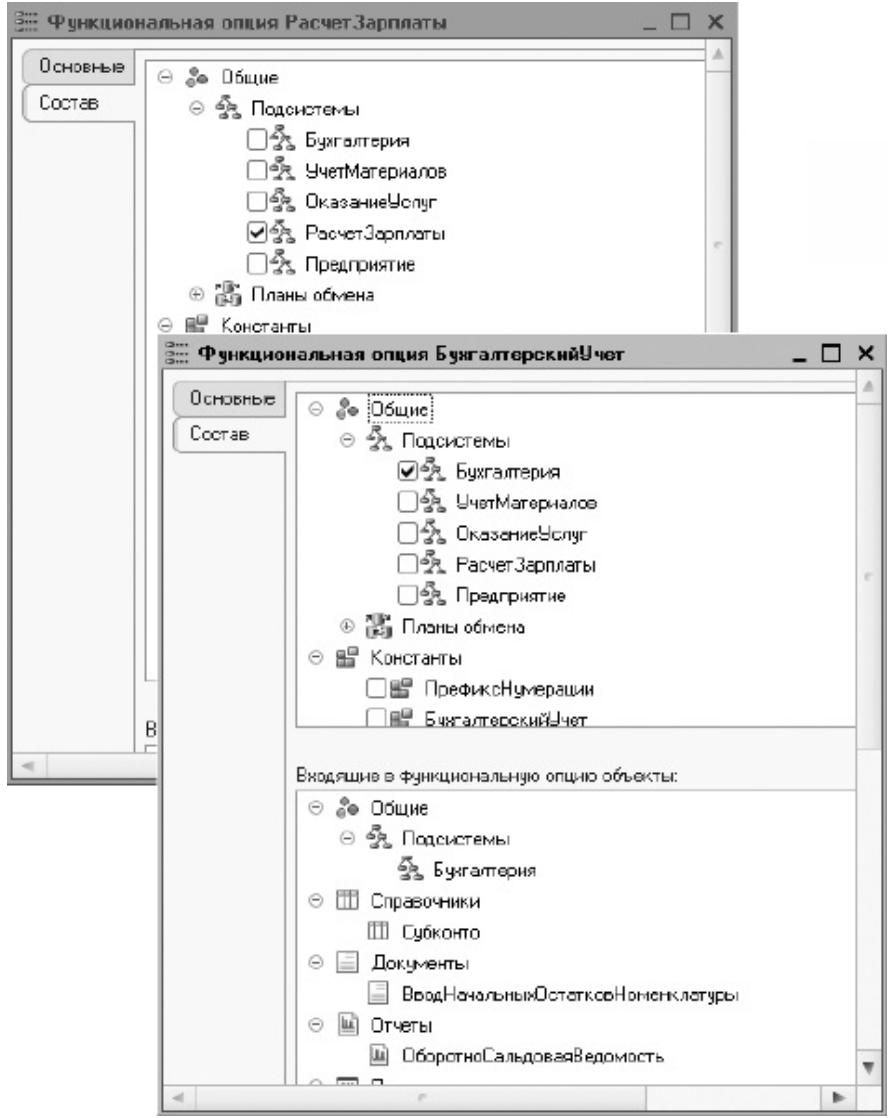

Функциональная опция РасчетЗарплаты
Основные
Состав
Общие
Подсистемы
Бухгалтерия
УчетМатериалов
ОказаниеУслуг
РасчетЗарплаты
Предприятие
Планы обмена
Константы
Функциональная опция БухгалтерскийУчет
Основные
Состав
Общие
Подсистемы
Бухгалтерия
УчетМатериалов
ОказаниеУслуг
РасчетЗарплаты
Предприятие
Планы обмена
Константы
ПрефиксНумерации
БухгалтерскийУчет
Входящие в функциональную опцию объекты:
Общие
Подсистемы
Бухгалтерия
Справочники
Субконто
Документы
ВводНачальныхОстатковНоменклатуры
Отчеты
ОборотноСальдоваяВедомость

Рис. 25.11. Состав функциональных опции «РасчетЗарплаты» и «БухгалтерскийУчет»

Теперь займемся созданием новой функциональной опции.

Для хранения этой опции добавим константу с именем *УчетКлиентов*. Она будет иметь тип *Булево* (рис. 25.12).

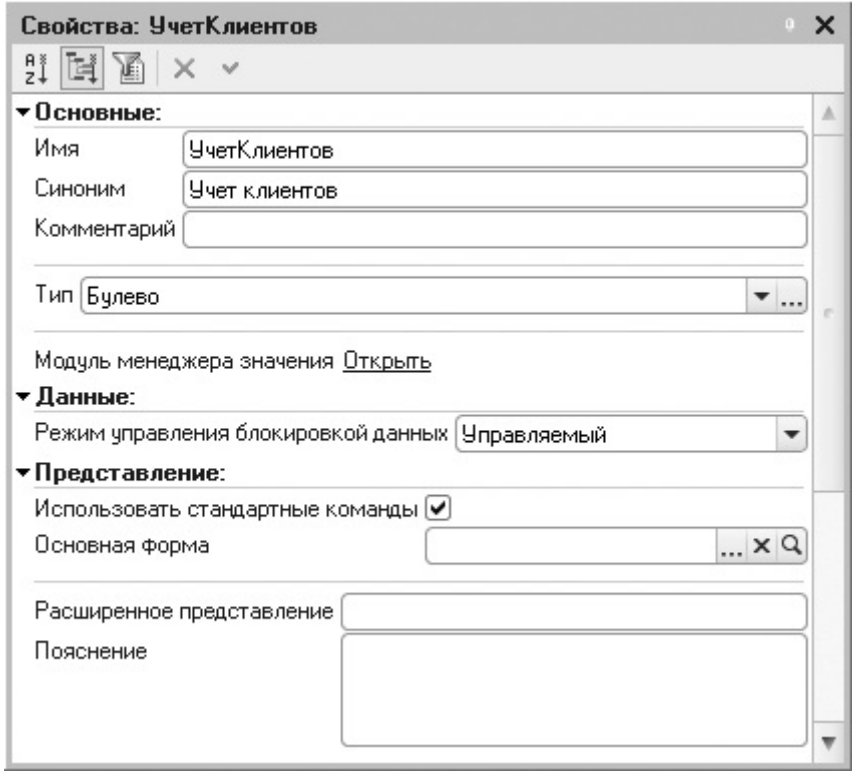


Рис. 25.12. Константа «УчетКлиентов»

Добавим функциональную опцию *УчетКлиентов* и укажем, что ее значение будет храниться в константе *УчетКлиентов* (рис. 25.13).

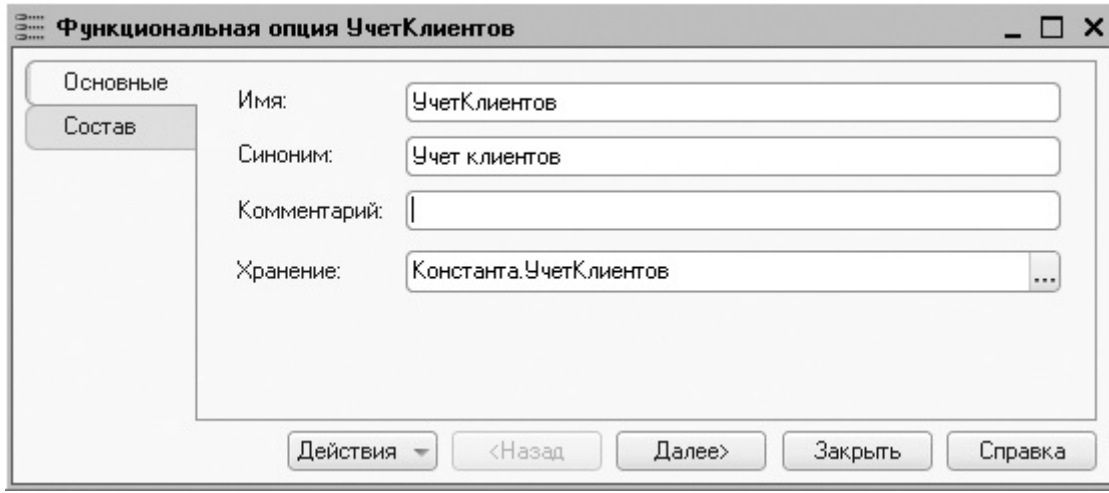


Рис. 25.13. Функциональная опция «УчетКлиентов»

Теперь на закладке *Состав* укажем, какие объекты будут входить в эту функциональную опцию.

Прежде всего – справочник *Клиенты*. Затем – реквизит *Клиент* документа *ОказаниеУслуги*. И в заключение – измерение *Клиент* регистра накопления *Продажи* (рис. 25.14).

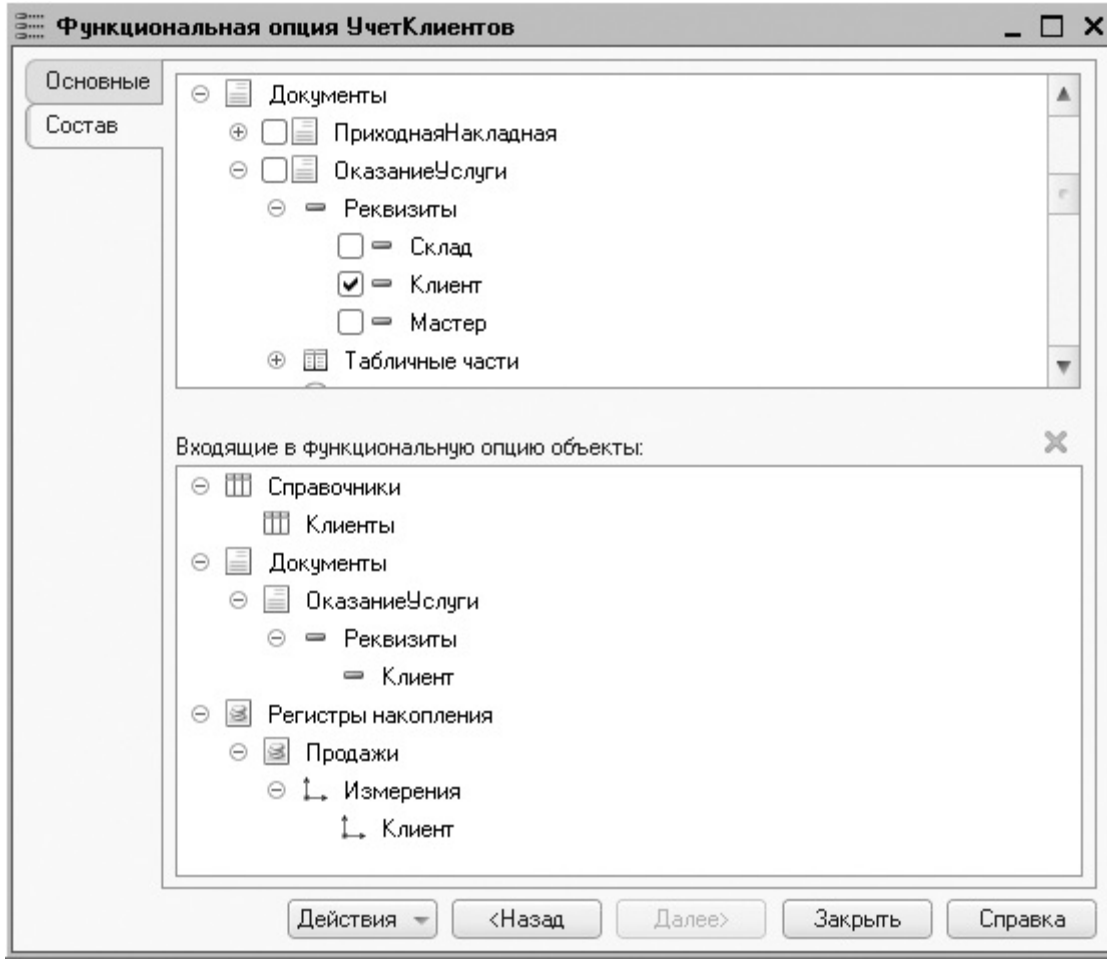


Рис. 25.14. Состав функциональной опции «УчетКлиентов»

Теперь доработаем общую форму *ОбщиеНастройки*, с помощью которой мы

устанавливаем значения функциональных опций.

Прежде всего перенесем в состав элементов формы нашу новую константу *УчетКлиентов* (рис. 25.15).

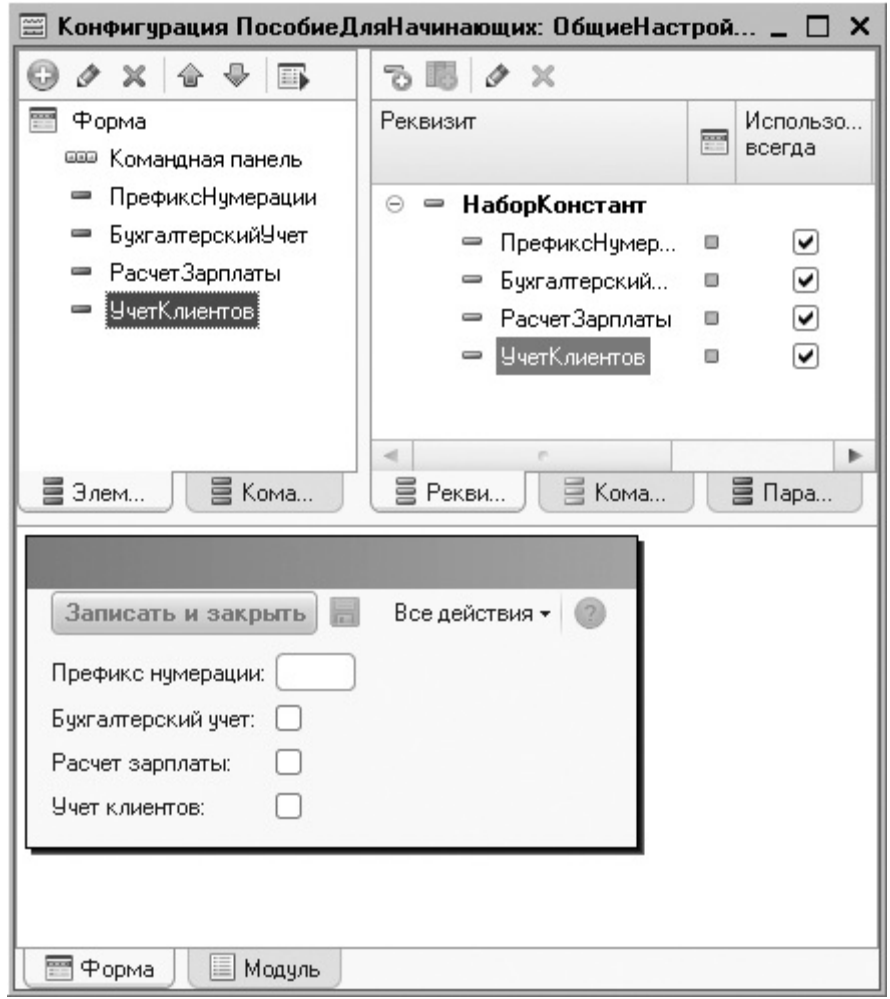


Рис. 25.15. Форма «ОбщиеНастройки»

После этого обеспечим автоматическую перерисовку интерфейса прикладного решения после установки новых значений функциональных опций.

Для этого в дереве элементов формы выделим корень (*Форма*), в палитре свойств найдем событие *ПослеЗаписи* формы и нажмем кнопку *Открыть* в поле ввода этого события.

В открывшемся модуле формы, в обработчике события формы *После записи*, напишем единственную строку (листинг 25.1).

Листинг 25.1. Обработчик события «ПослеЗаписи» формы

```
&НаКлиенте
Процедура ПослеЗаписи(ПараметрыЗаписи)

  ОбновитьИнтерфейс();

КонецПроцедуры
```

ОбновитьИнтерфейс() – это метод глобального контекста, который обновляет командный интерфейс, рабочий стол и открытые формы с учетом текущих значений функциональных опций и их параметров.

Для того чтобы проверка работы функциональных опций стала для нас

максимально удобной и легкой, сделаем так, чтобы открытие этой общей формы не блокировало основное окно программы.

Сейчас, открыв форму *ОбщиеНастройки* в режиме *1С:Предприятие*, мы не сможем переключить что-либо в основном окне программы, т. к. оно заблокировано до тех пор, пока эта форма открыта.

Так происходит потому, что по умолчанию конструктор форм установил для этой формы свойство *РежимОткрытияОкна* в значение *Блокировать окно владельца*.

Поэтому перейдем за закладку *Форма*, в дереве элементов формы выделим корневой элемент, в палитре свойств найдем свойство *РежимОткрытияОкна* и установим его в значение *Независимый* (рис. 25.16).

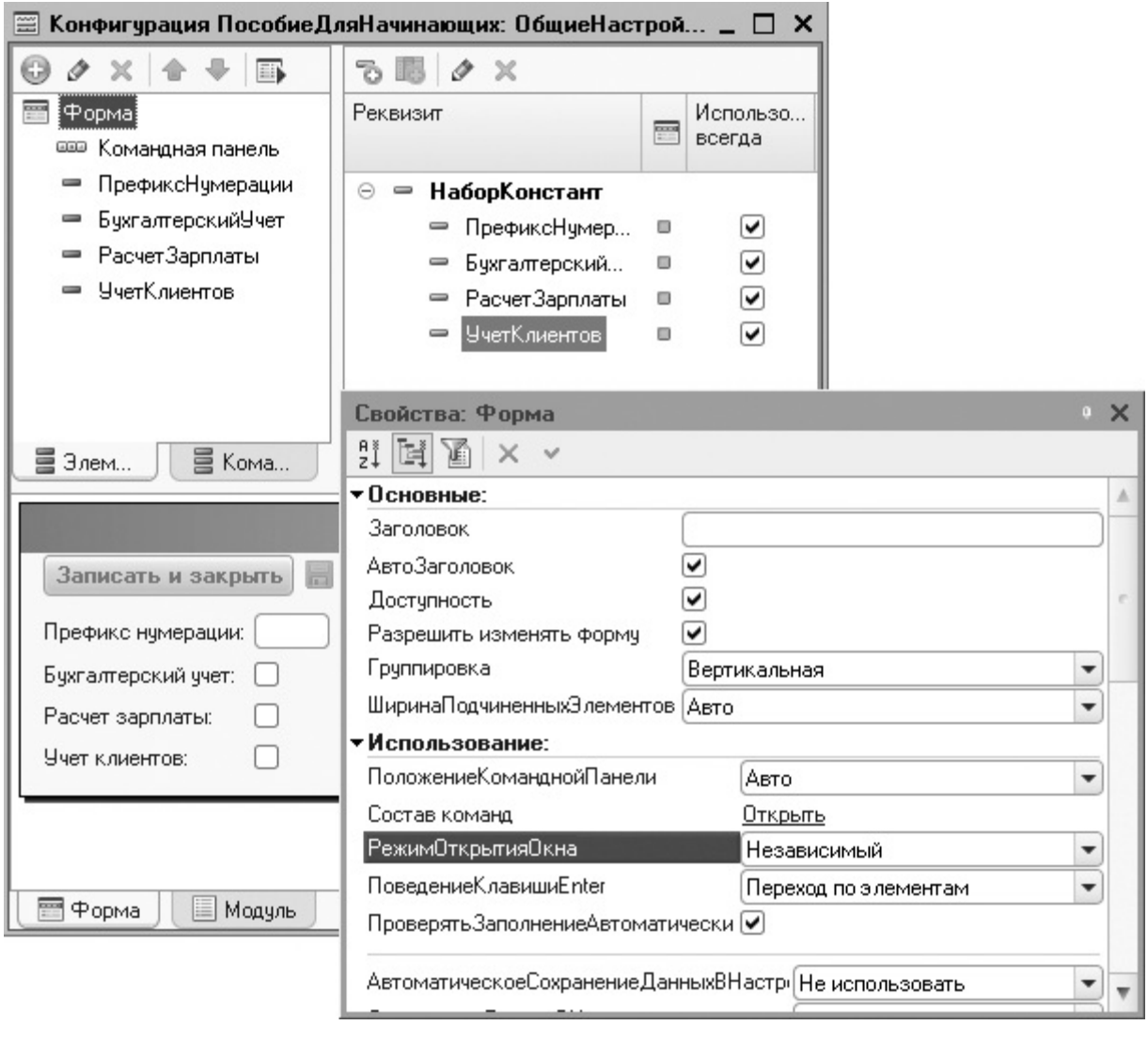

Конфигурация ПособиеДляНачинающих: ОбщиеНастрой...
Форма
Командная панель
ПрефиксНумерации
БухгалтерскийУчет
РасчетЗарплаты
УчетКлиентов
Элем...
Кома...
Реквизит
Использо... всегда
НаборКонстант
ПрефиксНумер...
Бухгалтерский...
РасчетЗарплаты
УчетКлиентов
Записать и закрыть
Префикс нумерации:
Бухгалтерский учет:
Расчет зарплаты:
Учет клиентов:
Форма
Модуль
Свойства: Форма
Основные:
Заголовок
АвтоЗаголовок
Доступность
Разрешить изменять форму
Группировка
Вертикальная
ШиринаПодчиненныхЭлементов
Авто
Использование:
ПоложениеКомандной Панели
Авто
Состав команд
Открыть
РежимОткрытияОкна
Независимый
ПоведениеКлавишиEnter
Переход по элементам
ПроверятьЗаполнениеАвтоматически
АвтоматическоеСохранениеДанныхВНастр
Не использовать

Рис. 25.16. Свойство «РежимОткрытияОкна»

В режиме «1С:Предприятие»

Запустим систему в режиме *1С:Предприятие*. В разделе *Предприятие* выполним команду *Общие настройки*. В основном окне перейдем в раздел *Оказание услуг* и выполним команду *Оказание услуг*.

Откроем любой документ *Оказание услуги*, например *№ 3* (рис. 25.17).

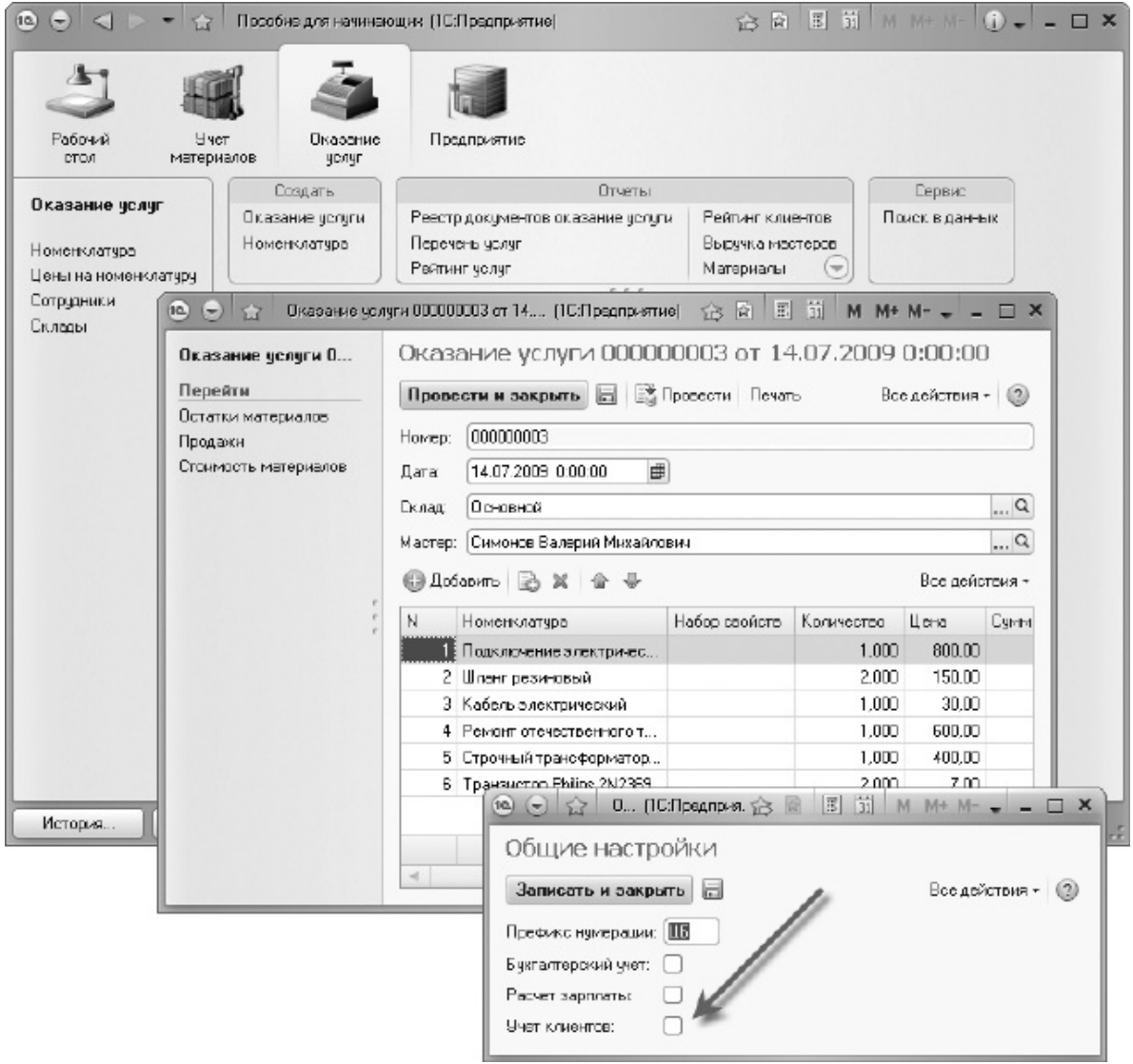

Пособие для начинающих (1С:Предприятие)
Рабочий стол
Учет материалов
Оказание услуг
Предприятие
Оказание услуг
Номенклатура
Цены на номенклатуру
Сотрудники
Склады
Создать
Оказание услуги
Номенклатура
Отчеты
Реестр документов оказание услуги
Перечень услуг
Рейтинг услуг
Рейтинг клиентов
Выручка мастеров
Материалы
Сервис
Поиск в данных
История...
Оказание услуги 000000003 от 14... (1С:Предприятие)
Оказание услуги 0...
Перейти
Остатки материалов
Продажи
Стоимость материалов
Оказание услуги 000000003 от 14.07.2009 0:00:00
Провести и закрыть
Провести
Печать
Все действия
Номер: 000000003
Дата: 14.07.2009 0:00:00
Склад: Основной
Мастер: Симонов Валерий Михайлович
Добавить
Все действия
N
Номенклатура
Набор свойств
Количество
Цена
Сумма
1 Подключение электричес... 1,000 800,00
2 Шланг резиновый 2,000 150,00
3 Кабель электрический 1,000 30,00
4 Ремонт отечественного т... 1,000 600,00
5 Строчный трансформатор... 1,000 400,00
6 Транзистор Philips 2N2369 2,000 7,00
0... (1С:Предприя...
Общие настройки
Записать и закрыть
Все действия
Префикс нумерации: ЦБ
Бухгалтерский учет:
Расчет зарплаты:
Учет клиентов:

Рис. 25.17. Функциональная опция «УчетКлиентов» отключена

Функциональная опция *Учет клиентов* отключена.

Поэтому в документе нет поля *Клиент*, в панели навигации раздела *Оказание услуг* нет команды *Клиенты*, а в ее панели действий нет команды создания нового клиента – *Клиент*.

В форме *Общие настройки* установим флажок *Учет клиентов* и нажмем пиктограмму *Записать* (рядом с кнопкой *Записать и закрыть*).

Интерфейс прикладного решения изменится (рис. 25.18).

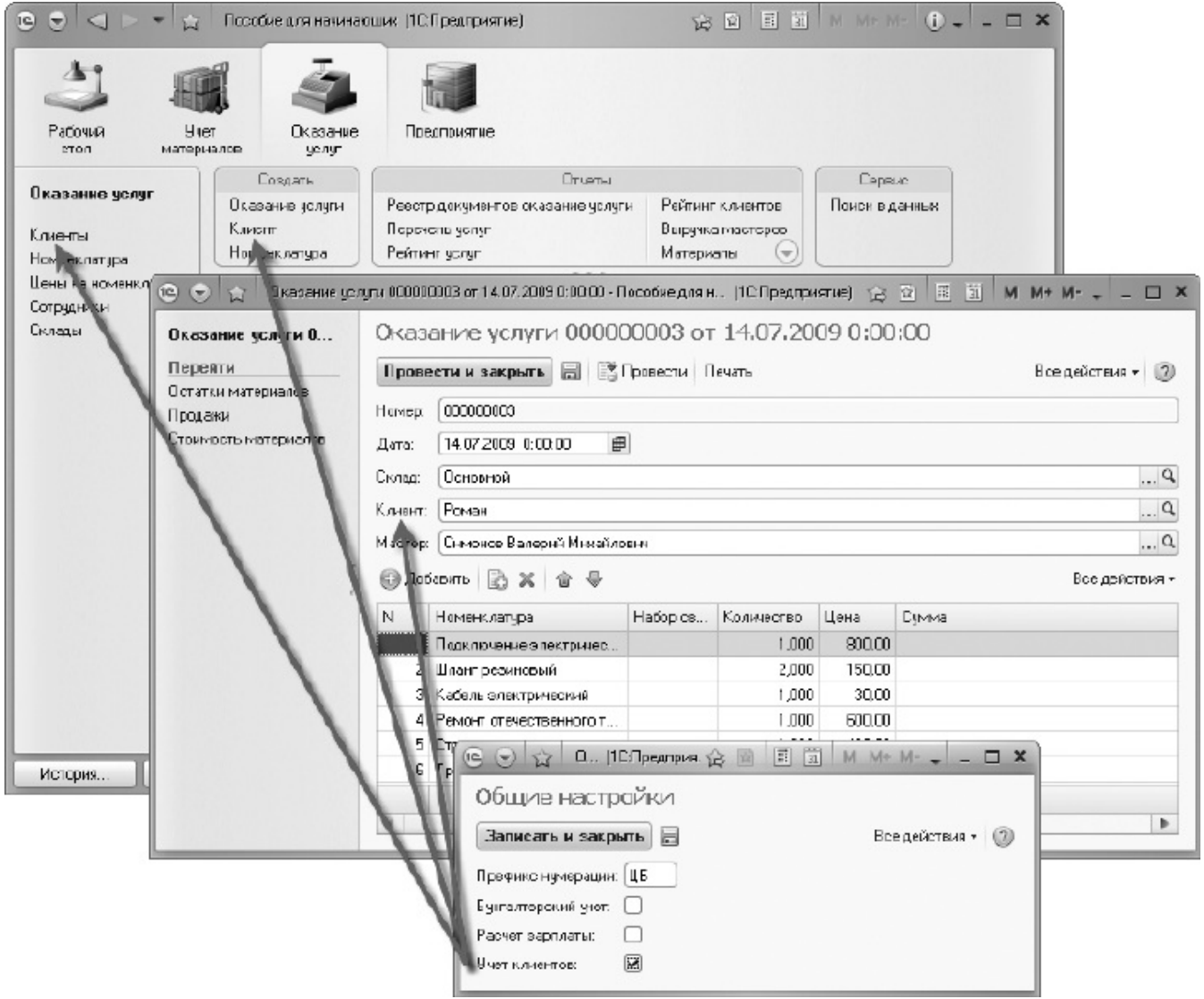

Пособие для начинающих (1С:Предприятие)
Рабочий стол
Учет материалов
Оказание услуг
Предприятие
Оказание услуг
Клиенты
Создать
Оказание услуги
Клиент
Отчеты
Реестр документов оказание услуги
Перечень услуг
Рейтинг услуг
Рейтинг клиентов
Выручка мастеров
Материалы
Сервис
Поиск в данных
История...
Оказание услуги 000000003 от 14.07.2009 0:00:00 - Пособие для н... (1С:Предприятие)
Оказание услуги 0...
Перейти
Остатки материалов
Продажи
Стоимость материалов
Оказание услуги 000000003 от 14.07.2009 0:00:00
Провести и закрыть
Провести
Печать
Все действия
Номер: 000000003
Дата: 14.07.2009 0:00:00
Склад: Основной
Клиент: Роман
Мастер: Симонов Валерий Михайлович
Добавить
Все действия
N
Номенклатура
Набор св...
Количество
Цена
Сумма
1 Подключение электричес... 1,000 800,00
2 Шланг резиновый 2,000 150,00
3 Кабель электрический 1,000 30,00
4 Ремонт отечественного т... 1,000 600,00
Общие настройки
Записать и закрыть
Все действия
Префикс нумерации: ЦБ
Бухгалтерский учет:
Расчет зарплаты:
Учет клиентов:

Рис. 25.18. Функциональная опция «УчетКлиентов» включена

Функциональная опция *Учет клиентов* включена.

В документе появилось поле *Клиент*, в панели навигации и панели действий раздела *Оказание услуг* появились команды работы со справочником *Клиенты*.

Аналогичным образом вы можете самостоятельно переключить различные функциональные опции и посмотреть, как при этом меняется интерфейс прикладного решения.

На этом мы фактически завершили разработку нашей конфигурации.

Следующие два занятия будут посвящены отдельным приемам разработки, которые часто используются в «1С:Предприятии 8».

Некоторые примеры мы будем выполнять в нескольких различных вариантах, поэтому какой из этих вариантов использовать в имеющейся конфигурации – это уже дело вашего личного вкуса.

Контрольные вопросы

- *Что такое функциональные опции и зачем они нужны.*
- *Как с помощью функциональных опций изменять интерфейс прикладного решения.*

Занятие 26 (1:00). Подборы и ввод на основании

Продолжительность

Ориентировочная продолжительность занятия – 1 час.

Существует ряд приемов использования объектов конфигурации, которые нельзя отнести только к какому-то одному виду объектов.

О таких приемах (подбор, ввод на основании) и пойдет речь на этом занятии.

Организация подборов

Задача организации подбора заключается, как правило, в заполнении табличной части документа информацией, которую выбирает пользователь в списке какого-либо объекта.

Для иллюстрации механизма подбора информации в форме мы будем использовать задачу подбора элементов справочника в табличную часть документа как наиболее распространенную.

Поскольку механизм подбора реализован на уровне форм, то в других случаях просто будут задействованы иные прикладные объекты. Сама механика подбора не изменится.

Для организации подбора в форму документа следует открыть форму справочника как подчиненную форме документа в целом либо одному из элементов формы. Способ получения формы справочника может быть любым, так же как и сама форма справочника, которая будет использована. Важно лишь то, что эта форма должна быть открыта как подчиненная.

Результат подбора будет доступен в обработчике события *ОбработкаВыбора* формы документа или элемента формы (в зависимости от того, чему мы подчиним форму справочника при открытии).

Событие *ОбработкаВыбора* в форме документа будет вызвано в двух случаях:

- когда в форме справочника будет выполнен интерактивный выбор;
- когда в форме справочника будет вызван метод *ОповеститьОВыборе()*.

Различные способы подбора мы проиллюстрируем на примере подбора элементов справочника *Номенклатура* в документ *ПриходнаяНакладная*.

Одиночный подбор

При одиночном подборе форма справочника будет закрываться сразу после

выбора элемента. Для выбора следующего элемента необходимо будет снова инициировать подбор.

В режиме «Конфигуратор»
Откроем форму документа *ПриходнаяНакладная.*

На закладке *Команды* создадим команду *Подбор* и в открывшейся палитре свойств нажмем кнопку открытия в строке *Действие*.

Шаблон обработчика события выполнения этой команды заполнять пока не будем, а перейдем на закладку *Форма* и перетащим эту команду в окно элементов формы, в командную панель таблицы *Материалы* (рис. 26.1).

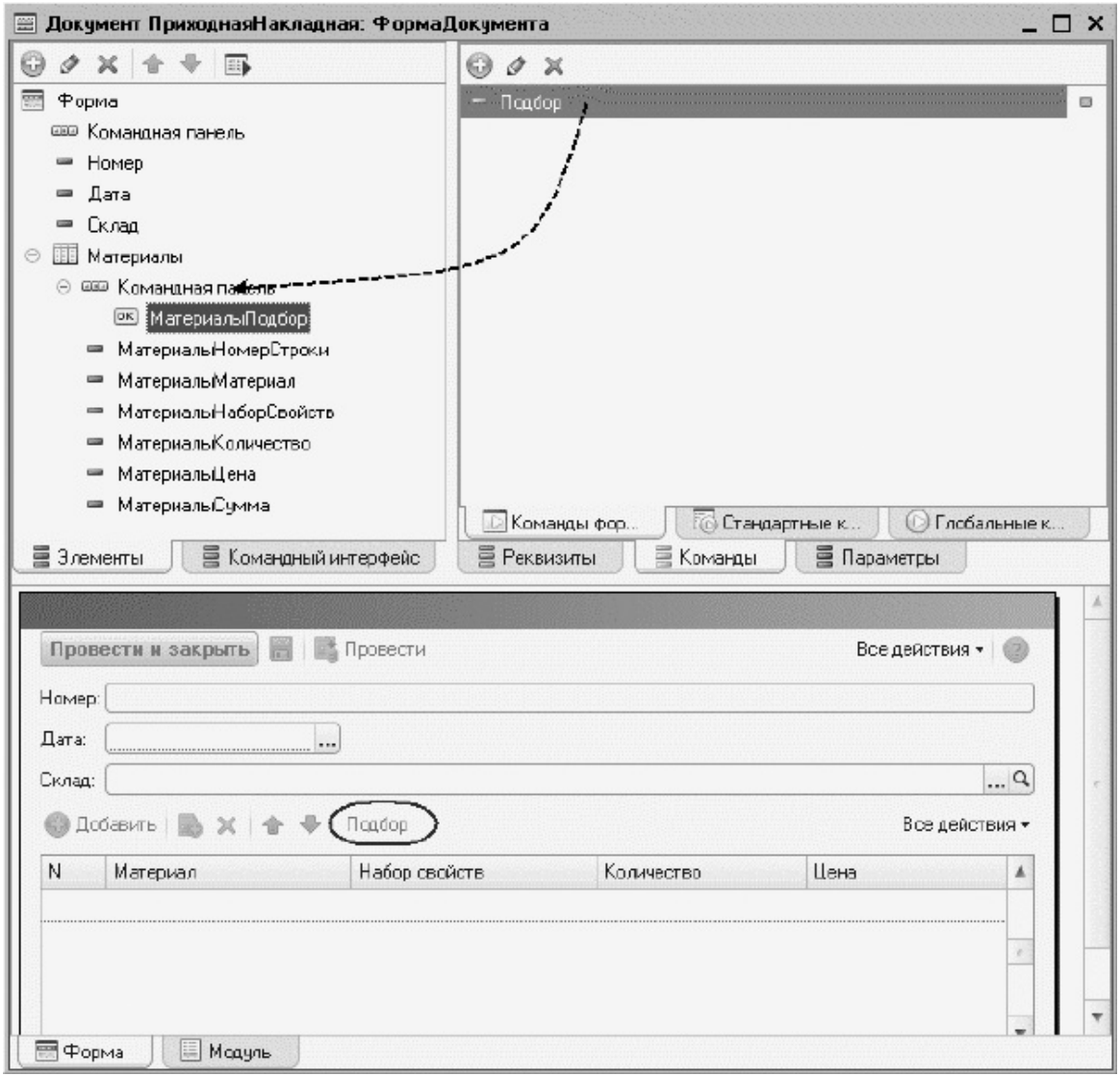

Документ ПриходнаяНакладная: ФормаДокумента
Форма
Командная панель
Номер
Дата
Склад
Материалы
Командная панель
МатериалыПодбор
МатериалыНомерСтроки
МатериалыМатериал
МатериалыНаборСвойств
МатериалыКоличество
МатериалыЦена
МатериалыСумма
Подбор
Команды фор...
Стандартные к...
Глобальные к...
Элементы
Командный интерфейс
Реквизиты
Команды
Параметры
Провести и закрыть
Провести
Все действия
Номер:
Дата:
Склад:
Добавить
Подбор
Все действия
N
Материал
Набор свойств
Количество
Цена
Форма
Модуль

Рис. 26.1. Создание кнопки «Подбор»

В форме документа, в обработчик события нажатия кнопки *Подбор* добавим следующий текст (листинг 26.1).

Листинг 26.1. Обработчик нажатия кнопки «Подбор»

```
&НаКлиенте
Процедура Подбор()

  ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", , Элементы.Материалы);

КонецПроцедуры
```

В этой процедуре мы открываем форму выбора для справочника *Номенклатура*, указывая, что она подчинена таблице *Материалы* формы документа *ПриходнаяНакладная* (*Элементы.Материалы*).

При выборе из формы выбора справочника выбранное значение будет передано в обработчик события *ОбработкаВыбора* таблицы формы *Материалы*, так как она является владельцем открытой формы выбора.

Поэтому откроем палитру свойств таблицы *Материалы* и создадим обработчик события *ОбработкаВыбора* (рис. 26.2), листинг 26.2.

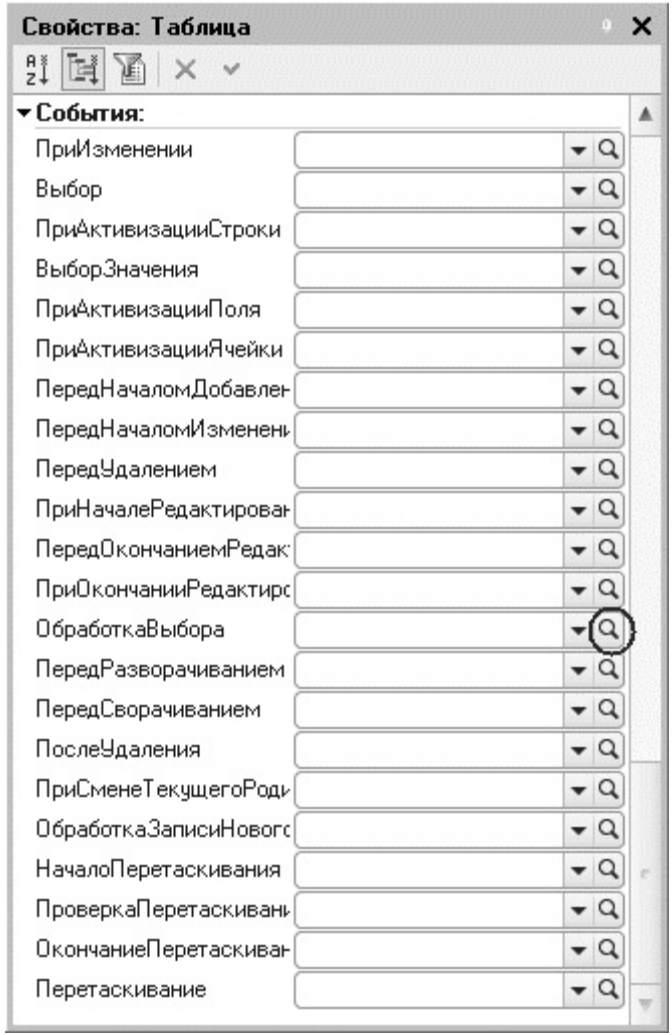


Рис. 26.2. Создание обработчика события «ОбработкаВыбора» для таблицы «Материалы»

Листинг 26.2. Обработчик события «ОбработкаВыбора» таблицы «Материалы»

```
&НаКлиенте
Процедура МатериалыОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)

  Элементы.Материалы.ДобавитьСтроку();
  Элементы.Материалы.ТекущиеДанные.Материал = ВыбранноеЗначение;

КонецПроцедуры
```

В этой процедуре мы добавляем новую строку в таблицу *Материалы* и присваиваем колонке *Материал* в новой строке выбранное в форме выбора справочника значение. Это значение передается в обработчик события в параметре *ВыбранноеЗначение*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

Перейдем в раздел *Учет материалов* и создадим новую приходную накладную командой *Приходная накладная*.

В командной панели списка нажмем *Подбор* и двойным щелчком мыши подберем в накладную один материал.

Множественный подбор

В режиме «Конфигуратор»

При множественном подборе форма справочника будет открыта до тех пор, пока пользователь не закроет ее интерактивно или не будет вызван метод формы *Закрыть()*.

В форме документа *ПриходнаяНакладная*, в обработчике события нажатия кнопки *Подбор* заменим прежний текст новым (листинг 26.3).

Листинг 26.3. Обработчик нажатия кнопки «Подбор»

```
&НаКлиенте
Процедура Подбор()

  ПараметрыФормы = Новый Структура("ЗакрыватьПриВыборе", Ложь);
  ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ПараметрыФормы,
  Элементы.Материалы);

КонецПроцедуры
```

При открытии формы мы используем ее *параметры*.

Параметры формы нужны для того, чтобы открыть форму в некотором нужном нам состоянии. Параметры формы представляют собой структуру. Каждый

элемент этой структуры описывает один параметр формы. Ключ элемента – это имя параметра формы.

Такую структуру мы передаем в метод *ОткрытьФорму()* вторым параметром (переменная *ПараметрыФормы*).

Предварительно мы эту структуру формируем. В ней у нас всего один элемент с ключом *ЗакрыватьПриВыборе*.

Таким образом, передавая эту структуру в метод *ОткрытьФорму(),* мы устанавливаем параметр открываемой формы *ЗакрыватьПриВыборе* в значение *Ложь*.

Это значит, что созданная форма выбора после двойного щелчка мыши на номенклатуре закрываться не будет.

В режиме «1С:Предприятие»
Запустим «1С:Предприятие» в режиме отладки.

Перейдем в раздел *Учет материалов* и создадим новую приходную накладную командой *Приходная накладная*.

В командной панели списка нажмем *Подбор.* Двойным щелчком мыши подберем

в накладную один материал, затем другой материал. Перейдем в группу *Услуги* и подберем несколько услуг.

Когда все желаемые товары и услуги будут выбраны, закроем окно с формой выбора.

Подбор с использованием множественного выбора

В режиме «Конфигуратор»

Еще одним способом организации подбора является возможность выделения в списке сразу нескольких строк.

Режим множественного выделения в списке устанавливается, как правило, во всех формах списков по умолчанию.

Однако возможность выбрать сразу несколько элементов из списка по умолчанию, как правило, отключена.

Поэтому для того, чтобы в форме списка справочника *Номенклатура* можно было не только отметить, но и выбрать сразу несколько элементов, мы снова воспользуемся одним из параметров расширения формы динамического списка – *МножественныйВыбор*.

В форме документа *ПриходнаяНакладная* заменим текст обработчика события нажатия кнопки *Подбор* следующим (листинг 26.4).

Листинг 26.4. Обработчик нажатия кнопки «Подбор»

```
&НаКлиенте
Процедура Подбор()

  ПараметрыФормы = Новый Структура("МножественныйВыбор", Истина);
  ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ПараметрыФормы,
  Элементы.Материалы);

КонецПроцедуры
```

При множественном выборе форма выбора будет возвращать уже не один элемент, а массив элементов.

Поэтому в обработчик события *ОбработкаВыбора* добавим обход массива переданных элементов (листинг 26.5).

Листинг 26.5. Обработчик события «ОбработкаВыбора» таблицы «Материалы»

```
&НаКлиенте
Процедура МатериалыОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)

  Для Каждого ВыбранныйЭлемент Из ВыбранноеЗначение Цикл
```

```
    НоваяСтрока = Объект.Материалы.Добавить();
    НоваяСтрока.Материал = ВыбранныйЭлемент;

  КонецЦикла;

КонецПроцедуры
```

В режиме «1С:Предприятие»
Запустим «1С:Предприятие» в режиме отладки.

Перейдем в раздел *Учет материалов* и создадим новую приходную накладную командой *Приходная накладная*.

В командной панели списка нажмем *Подбор.* Для удобства в открывшейся форме выбора справочника *Номенклатура* установим режим просмотра в виде списка – *Все функции – Режим просмотра – Список*.

Удерживая клавишу *Ctrl,* выделим в списке несколько товаров и несколько услуг.

Нажмем кнопку *Выбрать*.

Отмеченные элементы появятся в табличной части документа.

Множественный подбор с использованием множественного выбора

В режиме «Конфигуратор»

Последний способ подбора, который мы посмотрим, будет сочетать в себе оба рассмотренных ранее способа. Мы будем отмечать сразу несколько элементов справочника и подбирать их в документ без закрытия формы выбора. Затем снова отмечать несколько элементов справочника и подбирать их в документ.

Для этого нам будет необходимо при открытии формы выбора установить оба параметра: *ЗакрыватьПриВыборе* и *МножественныйВыбор*.

Процедура нажатия клавиши *Подбор* будет выглядеть следующим образом (листинг 26.6).

Листинг 26.6. Обработчик нажатия кнопки «Подбор»

```
&НаКлиенте
Процедура Подбор()

  ПараметрыФормы = Новый Структура("ЗакрыватьПриВыборе, МножественныйВыбор",
  Ложь, Истина);
  ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ПараметрыФормы,
  Элементы.Материалы);

КонецПроцедуры
```

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки.

Перейдем в раздел *Учет материалов* и создадим новую приходную накладную командой *Приходная накладная*.

В командной панели списка нажмем *Подбор.*

Откроем группу *Услуги – Телевизоры*, выделим в ней все услуги и нажмем кнопку *Выбрать*.

Откроем группу *Материалы – Прочее*, выделим в ней все материалы и нажмем кнопку *Выбрать*.

Закроем окно с формой выбора справочника *Номенклатура*.

Использование метода «Оповестить о выборе()»

Метод формы *ОповеститьОВыборе()* используется в тех случаях, когда алгоритм формирования данных подбора сложен, и кроме собственно выбора элемента справочника от пользователя требуется указание некоторой дополнительной информации. В этом случае метод *ОповеститьОВыборе()* вызывается тогда, когда вся необходимая информация подбора

сформирована.

Метод *ОповеститьОВыборе()* посылает оповещение владельцу формы о выполнении выбора или подбора, передает ему выбранное значение и закрывает форму, если она открыта не в режиме множественного выбора.

Также метод *ОповеститьОВыборе()* может использоваться в тех случаях, когда требуется передать в форму документа не только выбранный элемент справочника (или массив элементов), а некоторую произвольную структуру данных.

Ввод на основании

Механизм ввода на основании может быть использован для ввода новых объектов различного типа (документы, справочники, планы видов характеристик и т. д.). Мы рассмотрим этот механизм на примере ввода новых документов как наиболее распространенном.

Для каждого объекта конфигурации *Документ* можно разрешить его ввод на основании других объектов базы данных и возможность являться основанием для других объектов.

Действия по заполнению реквизитов при вводе на основании должны быть

описаны в модуле объекта *Документ*, в обработчике события *ОбработкаЗаполнения*.

Это можно сделать вручную или с использованием конструктора ввода на основании, который позволяет визуальными средствами конструировать текст обработчика.

Рассмотрим пример, когда документ *ОказаниеУслуги* будет вводиться на основании элемента справочника *Клиенты*.

Команда ввода на основании

В режиме «Конфигуратор»

Откроем окно редактирования объекта конфигурации *Документ ОказаниеУслуги* и добавим новый реквизит документа – *ОбъектОснование* с типом *СправочникСсылка.Клиенты*.

Создание такого реквизита не является обязательной частью механизма ввода на основании и понадобится нам только для того, чтобы впоследствии построить цепочку зависимых документов.

Перейдем на закладку *Ввод на основании* и определим состав документов, на основании которых может вводиться документ *ОказаниеУслуги* и основанием

для которых он может являться.

Нажмем кнопку *Редактировать элемент списка* над списком *Вводится на основании* и выберем справочник *Клиенты* (рис. 26.3).

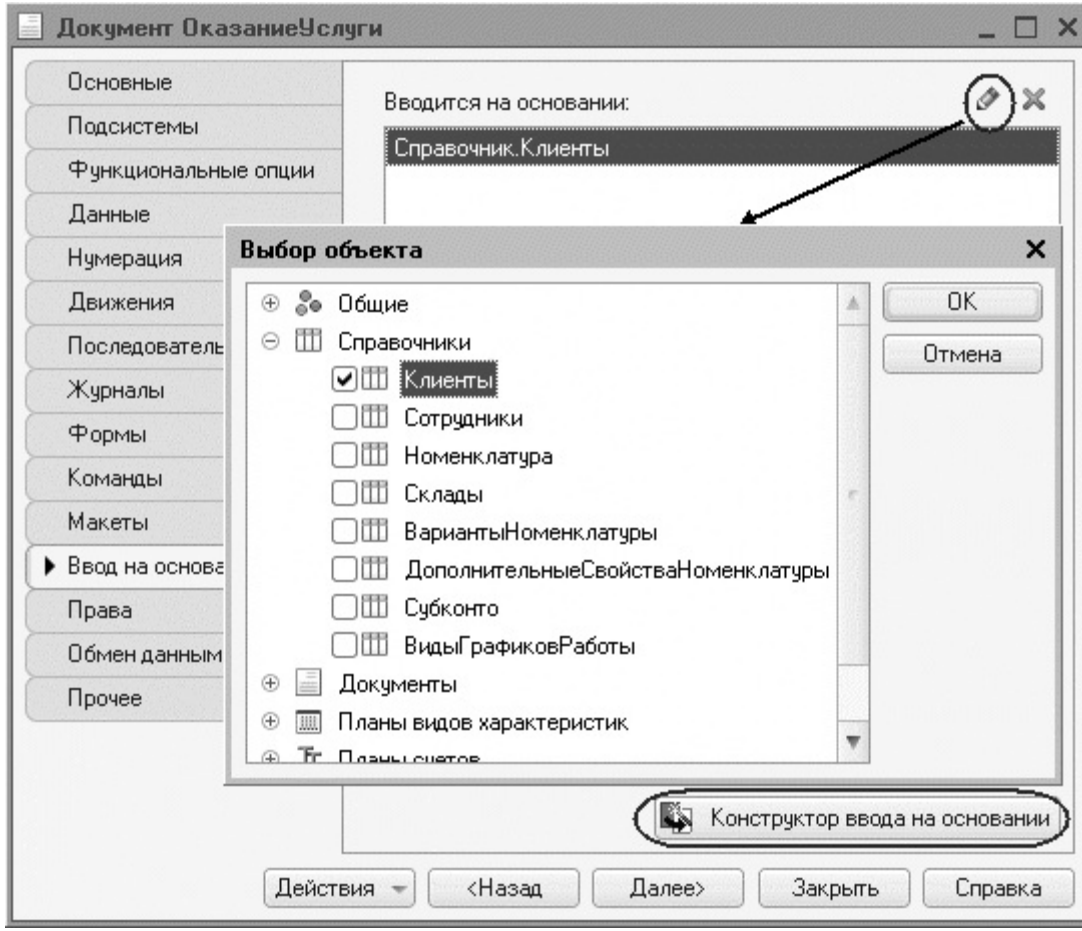


Рис. 26.3. Определение состава объектов, на основании которых вводится документ

Затем вызовем конструктор ввода на основании и зададим значения реквизитов документа, создаваемого на основании. Для этого воспользуемся

кнопкой *Заполнить выражения* (рис. 26.4).

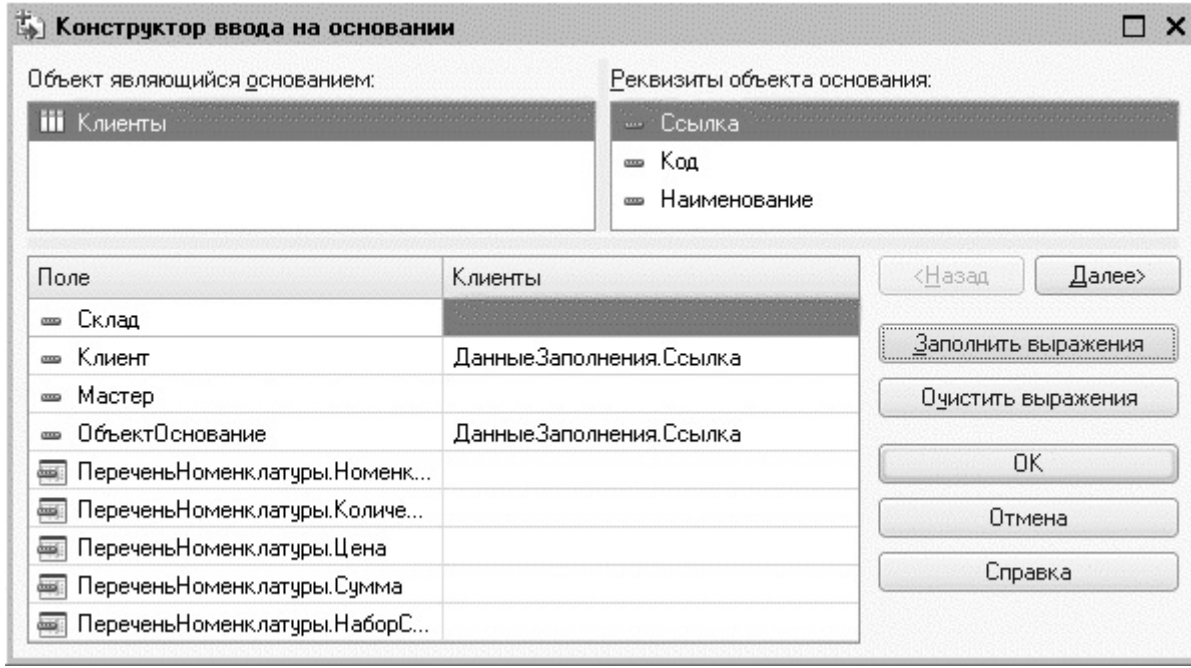


Рис. 26.4. Заполнение значений реквизитов документа, создаваемого на основании

Обратите внимание, что для заполнения реквизита *ОбъектОснование* конструктор предложил использовать значение *ДанныеЗаполнения.Ссылка*. В данном случае такая запись будет избыточной, поскольку в качестве основания будет передана ссылка на элемент справочника.

Однако в общем случае событие *ОбработкаЗаполнения* возникает при создании нового объекта на основании некоторого переданного значения. Совсем не обязательно, что это значение будет иметь тип ссылки.

Согласимся со всем, что предложил конструктор, и нажмем *OK*.

В модуле документа будет сформирован текст обработчика события *ОбработкаЗаполнения* (листинг 26.7).

Листинг 26.7 Обработчик события «ОбработкаЗаполнения»

```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)

  //{{__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
  // Данный фрагмент построен конструктором.
  // При повторном использовании конструктора внесенные вручную изменения будут
  утеряны!!!
  Если ТипЗнч(ДанныеЗаполнения) = Тип("СправочникСсылка.Клиенты") Тогда

    // Заполнение шапки
    Клиент = ДанныеЗаполнения.Ссылка;
    ОбъектОснование = ДанныеЗаполнения.Ссылка;

  КонецЕсли;
  //}}__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ

КонецПроцедуры
```

Как видите, для каждого типа объекта-основания формируется своя ветка условия *Если…*, в которой происходит заполнение реквизитов нового документа.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим работу ввода на основании.

Откроем список клиентов. Обратите внимание, что в командной панели формы списка справочника *Клиенты* появилась команда *Создать на основании*.

Выделив нужного клиента и выполнив команду *Создать на основании* > *Оказание услуги*, создадим новый документ *Оказание услуги*, где в качестве клиента будет выбран выделенный в списке справочника клиент (рис. 26.5).

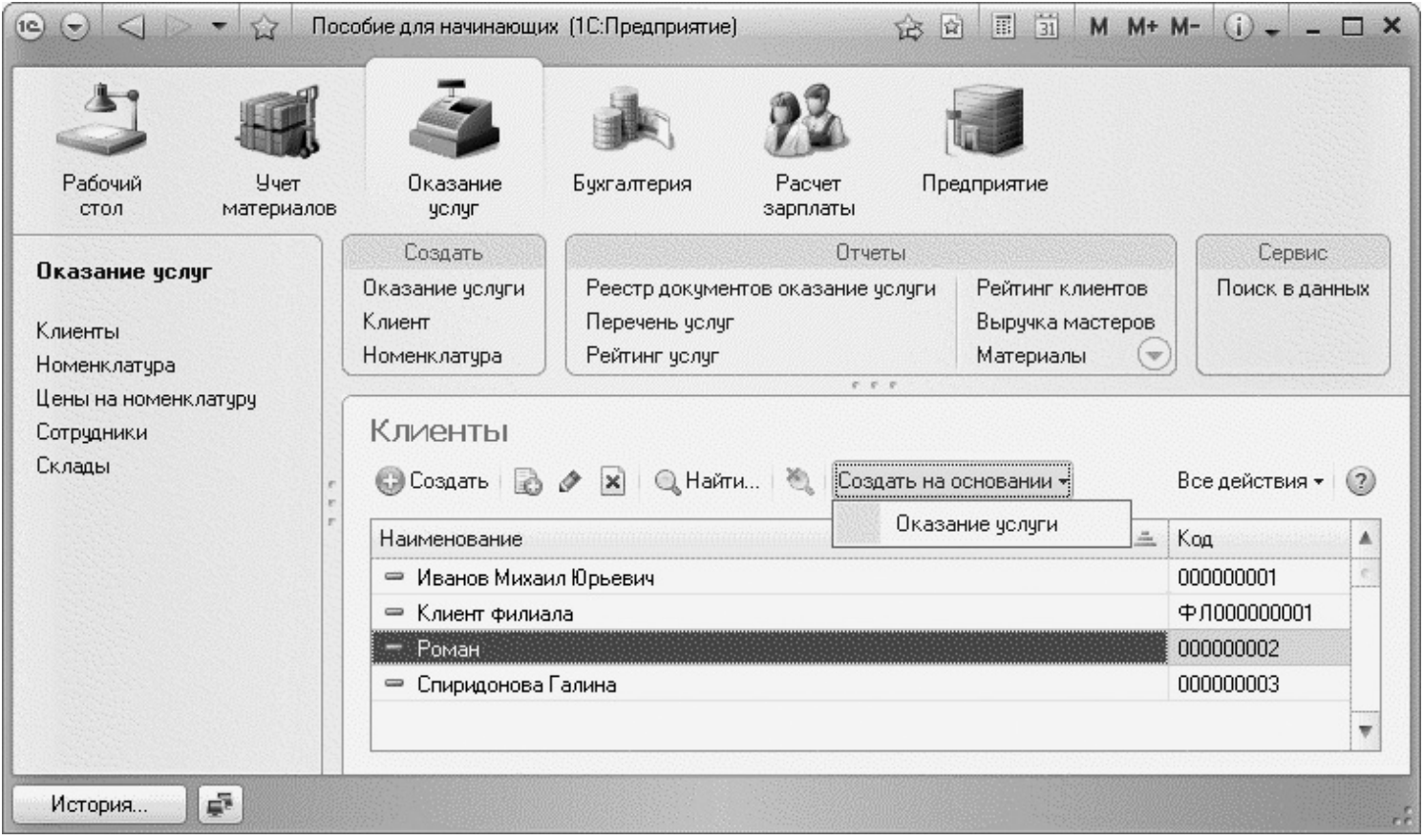


Рис. 26.5. Создание документа «Оказание услуги» на основании клиента

Введите самостоятельно еще несколько документов на основании какого-либо клиента.

Объекты, введенные на основании

Наряду с тем, что платформа содержит механизмы, позволяющие создавать одни объекты на основании других, каких-либо специальных механизмов для анализа цепочек связанных объектов в платформе нет.

Для решения подобной задачи мы дадим некоторые рекомендации, которые могут быть положены в основу конкретного решения.

Для построения цепочек связанных объектов необходимо у каждого объекта, который будет вводиться на основании, создать служебный реквизит для хранения ссылки на объект-основание. Затем следует создать объект конфигурации *КритерийОтбора*, который будет использоваться для установки отбора по требуемому значению служебного реквизита.

В дальнейшем для получения всех объектов, введенных на основании, достаточно будет установить нужное значение отбора в критерии отбора.

Критерий отбора

Объект конфигурации *КритерийОтбора* предназначен для задания правил, по которым может выполняться отбор объектов.

Этот объект используется в случае поиска различной информации, когда,

например, требуется отобрать все документы, в которых используется (в реквизитах и в табличных частях) определенный контрагент.

При этом можно учитывать также и другие условия отбора информации (например, поиск ведется только среди проведенных документов или в определенном интервале дат).

Получение объектов, введенных на основании

Поскольку задача получения всех объектов, введенных на основании какого-либо другого объекта, чаще всего возникает при анализе документов, мы рассмотрим применение описанной выше методики на примере получения списка документов, введенных на основании элемента справочника *Клиенты*.

В режиме «Конфигуратор»

Раскроем ветвь *Общие* и создадим новый объект конфигурации КритерийОтбора с именем *ОказаниеУслуги.*

На закладке *Данные* выберем тип используемого критерия – *СправочникСсылка.Клиенты*.

На закладке *Состав* в качестве объектов, входящих в критерий, выберем реквизит *ОбъектОснование* документа *ОказаниеУслуги* (рис. 26.6).

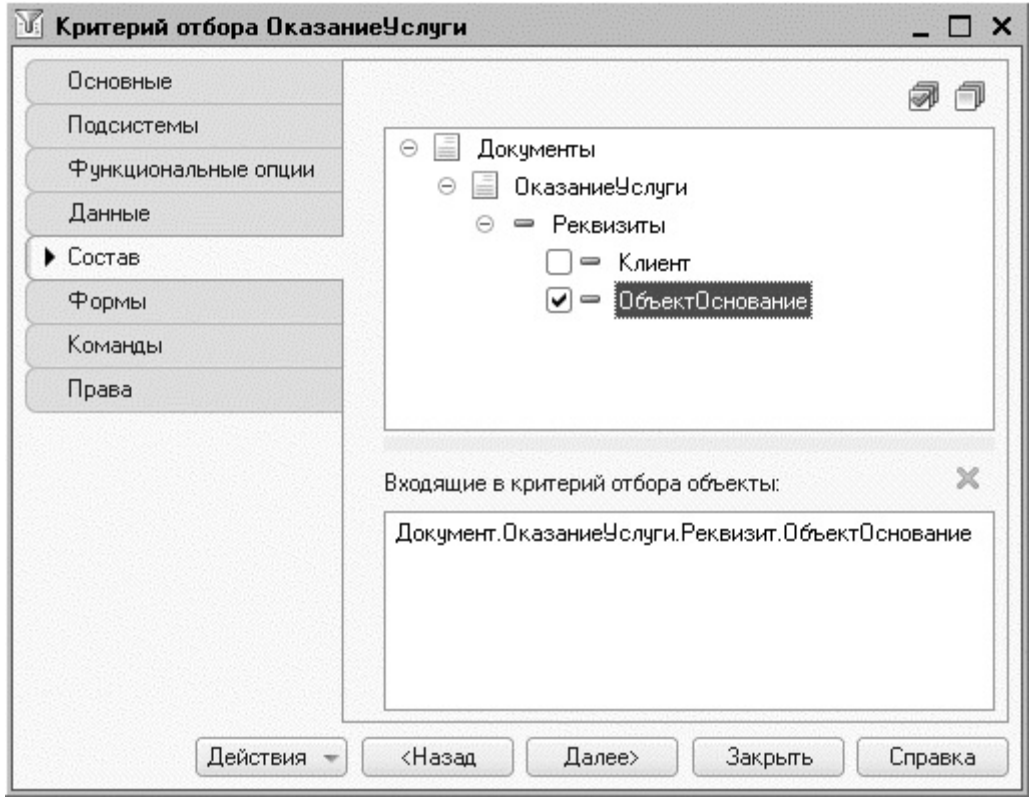


Рис. 26.6. Состав критерия отбора «ОказаниеУслуги»

После этого в панели навигации формы элемента справочника *Клиенты* в группе *Перейти* появится команда для открытия критерия отбора.

Создадим эту форму и на закладке *Командный интерфейс* установим

видимость команды *Оказание услуги* (рис. 26.7).

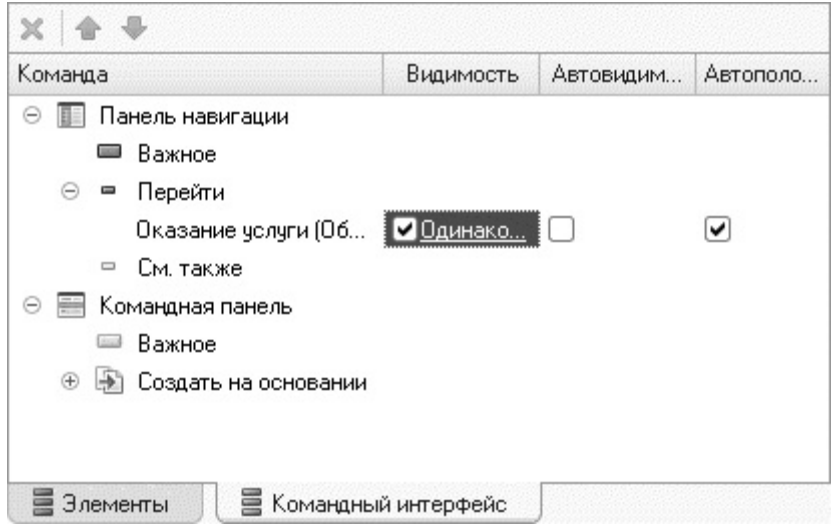


Рис. 26.7. Команда открытия критерия отбора из формы элемента справочника «Клиенты»

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и проверим работу критерия отбора.

В форме списка клиентов выделим клиента *Роман*, на основании которого мы создавали документы *Оказание услуги*.

Откроем форму этого клиента. В панели навигации появилась команда

Оказание услуги для открытия формы списка созданного нами критерия отбора с установленным отбором по открытому элементу справочника *Клиенты*.

Выполним эту команду (рис. 26.8).

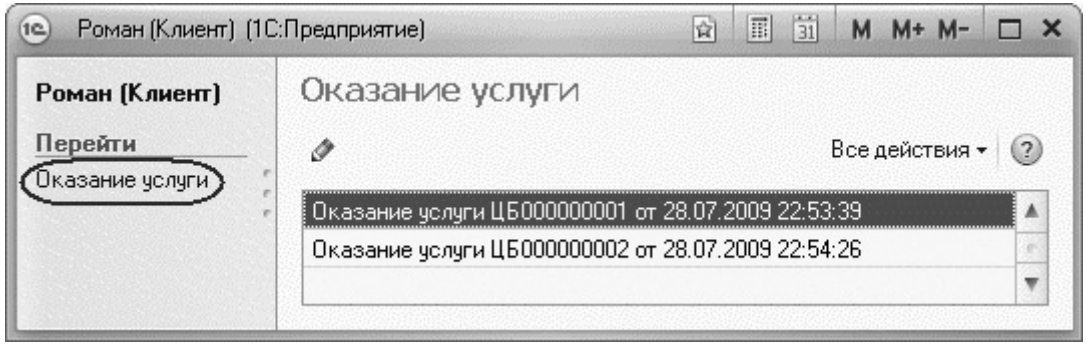


Рис. 26.8. Открытие списка критерия отбора «ОказаниеУслуги» с отбором по клиенту

Мы видим в этом списке документы *Оказание услуги*, созданные на основании клиента *Роман*. К содержимому документа можно перейти, нажав соответствующую ссылку в списке документов.

Контрольные вопросы

- *Что такое подбор.*
- *Как организовать различные виды подбора в табличную часть формы документа.*

- *Что такое ввод на основании.*
- *Как организовать ввод одних объектов конфигурации на основании других.*
- *Как с помощью критерия отбора вывести список объектов, введенных на основании текущего объекта.*

Занятие 27 (2:10). Приемы разработки форм

Продолжительность

Ориентировочная продолжительность занятия – 2 часа 10 минут.

На этом занятии мы рассмотрим несколько типичных приемов работы с формами объектов.

Начнем занятие с рассказа о том, каким образом данные отображаются в формах, и о тех типах данных, которые при этом используются.

Данные и элементы формы

Важной особенностью платформы «1С:Предприятие 8» является механизм представления данных в формах. Ключевым моментом здесь является то, что принадлежность формы к тому или иному объекту конфигурации никоим образом не определяет состав данных, которые форма будет отображать.

Например, можно создать общую форму, которая не будет подчинена ни одному из объектов конфигурации, но которая, в зависимости от содержимого, будет либо отображать список справочника, либо позволять редактировать документ и т. п. Однако такую форму уже нельзя будет назначить основной для выполнения определенных действий.

Форма сама по себе и ее элементы обособлены от объектов конфигурации. Для того чтобы форма отображала какие-либо данные, необходимо задать связь самой формы и ее элементов с данными. Если связь элементов формы с данными не задана, то элементы вообще не будут отображены в форме (кроме элементов оформления формы).

При использовании конструктора форм конфигуратор создает такие связи автоматически. Если разработчик создает форму вручную, он может определить эти связи путем задания свойств формы и ее элементов (рис. 27.1).

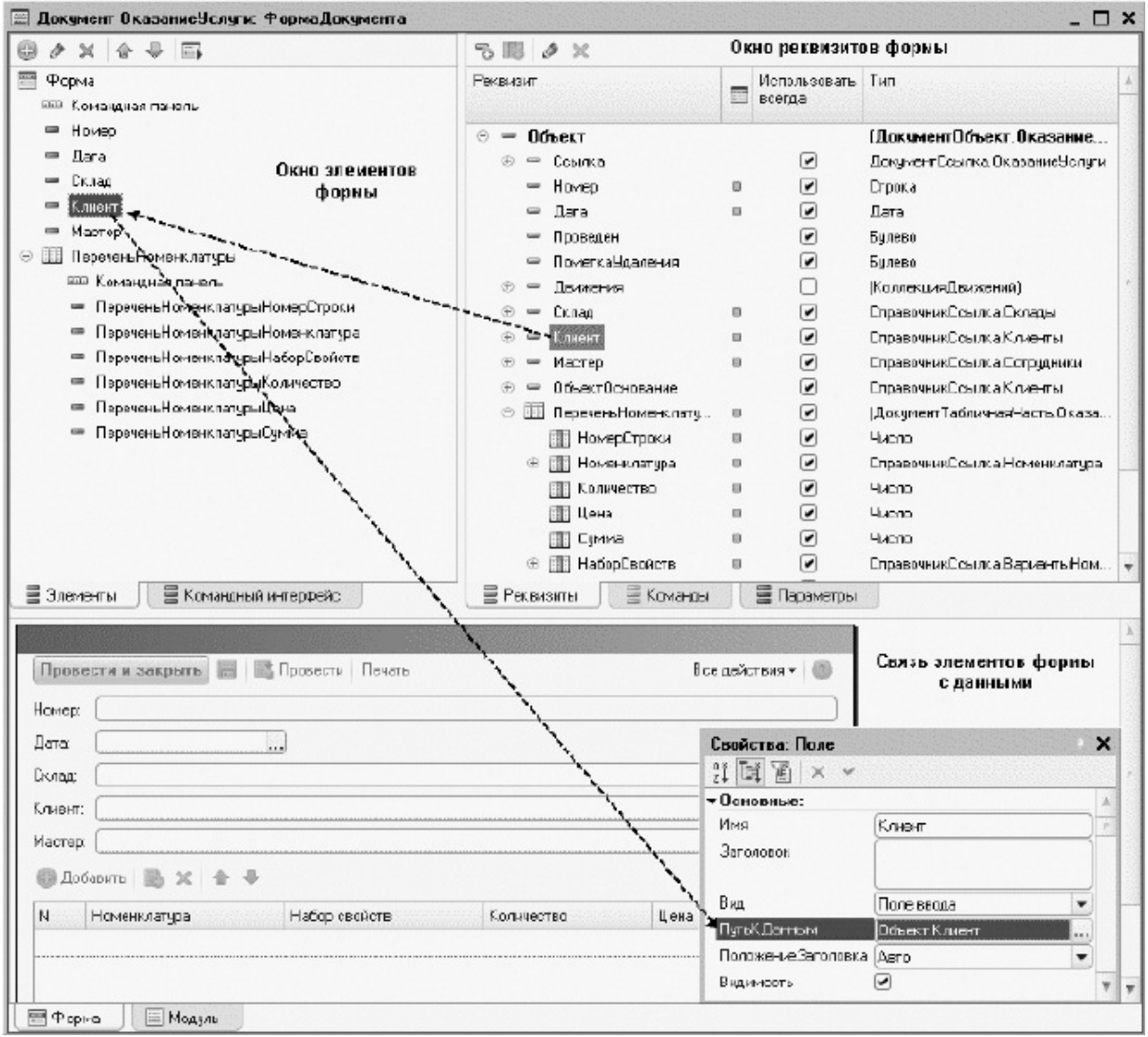

Документ ОказаниеУслуг: ФормаДокумента
Форма
Командная панель
Номер
Дата
Склад
Клиент
Мастер
ПереченьНоменклатуры
Командная панель
ПереченьНоменклатурыНомерСтроки
ПереченьНоменклатурыНоменклатура
ПереченьНоменклатурыНаборСвойств
ПереченьНоменклатурыКоличество
ПереченьНоменклатурыЦена
ПереченьНоменклатурыСумма
Окно элементов формы
Элементы
Командный интерфейс
Окно реквизитов формы
Реквизит
Использовать всегда
Тип
Объект
ДокументОбъект.ОказаниеУ...
Ссылка
ДокументСсылка.ОказаниеУслуги
Номер
Строка
Дата
Дата
Проведен
Булево
ПометкаУдаления
Булево
Движения
КоллекцияДвижений
Склад
СправочникСсылка.Склады
Клиент
СправочникСсылка.Клиенты
Мастер
СправочникСсылка.Сотрудники
ОбъектОснование
СправочникСсылка.Клиенты
ПереченьНоменклату...
ДокументТабличнаяЧасть.Оказа...
НомерСтроки
Число
Номенклатура
СправочникСсылка.Номенклатура
Количество
Число
Цена
Число
Сумма
Число
НаборСвойств
СправочникСсылка.ВариантыНом...
Реквизиты
Команды
Параметры
Связь элементов формы с данными
Провести и закрыть
Провести
Печать
Все действия
Номер:
Дата:
Склад:
Клиент:
Мастер:
Добавить
N
Номенклатура
Набор свойств
Количество
Цена
Свойства: Поле
Основные:
Имя
Клиент
Заголовок:
Вид
Поле ввода
ПутьКДанным
Объект.Клиент
ПоложениеЗаголовка
Авто
Видимость
Форма
Модуль

Рис. 27.1. Связь элементов формы с данными

Связь элементов формы с данными, которые они должны отображать, задается в свойстве *ПутьКДанным*.

Связь формы и ее элементов с данными осуществляется при помощи реквизитов формы. Список существующих реквизитов формы доступен на закладке *Реквизиты* окна редактора формы.

Среди всех реквизитов формы, как правило, существует один основной реквизит (он выделен жирным шрифтом). Он определяет источник данных для формы в целом. От типа значения основного реквизита формы зависит не только то, какие данные будут отображены в элементах формы, но и поведение самой формы.

Например, если основному реквизиту формы указать тип значения *ДокументОбъект.ПриходнаяНакладная*, то при закрытии формы программа будет запрашивать подтверждение записи и проведения документа. Если же основному реквизиту формы указать тип значения *СправочникОбъект.Клиенты*, то подобного подтверждения при закрытии формы возникать не будет.

Похожее влияние источники данных оказывают и на элементы формы.

Например, состав колонок таблицы, источником данных которой является реквизит формы с типом значения *ДинамическийСписок*, будет различным в зависимости от того, какой объект используется в качестве основной таблицы этого динамического списка (например, *РегистрНакопления.ОстаткиНоменклатуры* или *Справочник.Номенклатура*).

То же самое справедливо и для элемента формы *Командная панель*. При установленном свойстве командной панели *Автозаполнение* смена источника данных (а точнее говоря, источника действий) будет приводить к изменению состава команд, которые отображает командная панель.

Возможность связать форму и ее элементы с различными данными является причиной того, что у формы и ее элементов существует несколько *расширений*.

Расширение представляет собой набор дополнительных свойств, методов и событий, появляющихся у формы или у элемента формы. Наличие того или иного расширения определяется либо типом данных, которые отображает форма/элемент, либо расположением элемента формы в других ее элементах.

Чтобы подробнее познакомиться с этим механизмом, создадим основную форму списка справочника *Номенклатура*. При этом в конструкторе формы мы не будем сразу нажимать кнопку *Готово*, как делали раньше.

Нажмем кнопку *Далее* > и кроме полей *Наименование* и *Код* включим в состав таблицы *Список* еще одно поле – *ВидНоменклатуры*. И затем уже нажмем *Готово* (рис. 27.2).

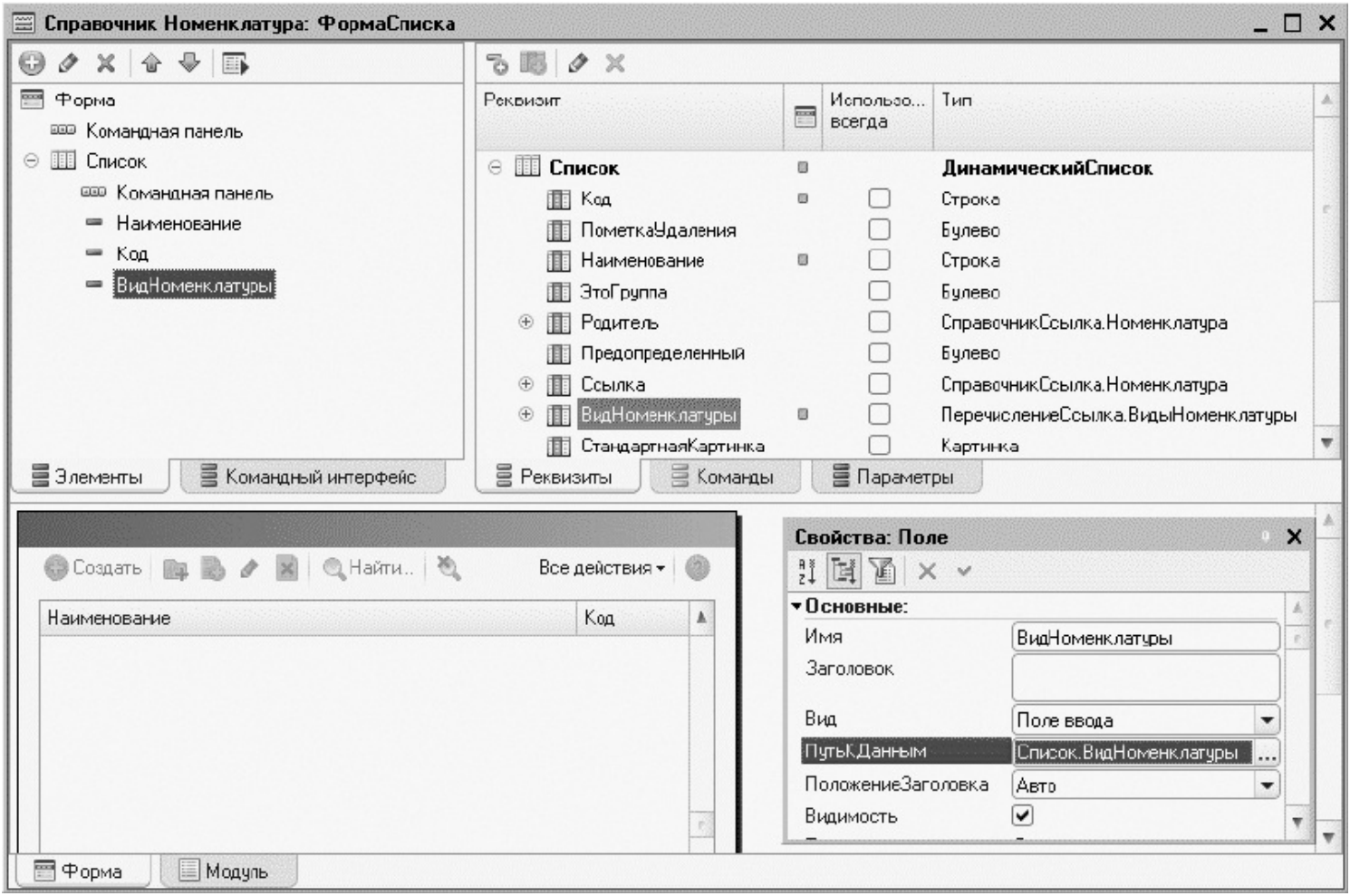


Рис. 27.2. Форма списка справочника «Номенклатура»

Итак, с механизмом расширений мы будем знакомиться на примере поля

ВидНоменклатуры, расположенного в таблице *Список* формы списка справочника *Номенклатура*.

Сама форма отображает данные объекта *ДинамическийСписок* (основной реквизит формы *Список* имеет тип *ДинамическийСписок*, рис. 27.3).

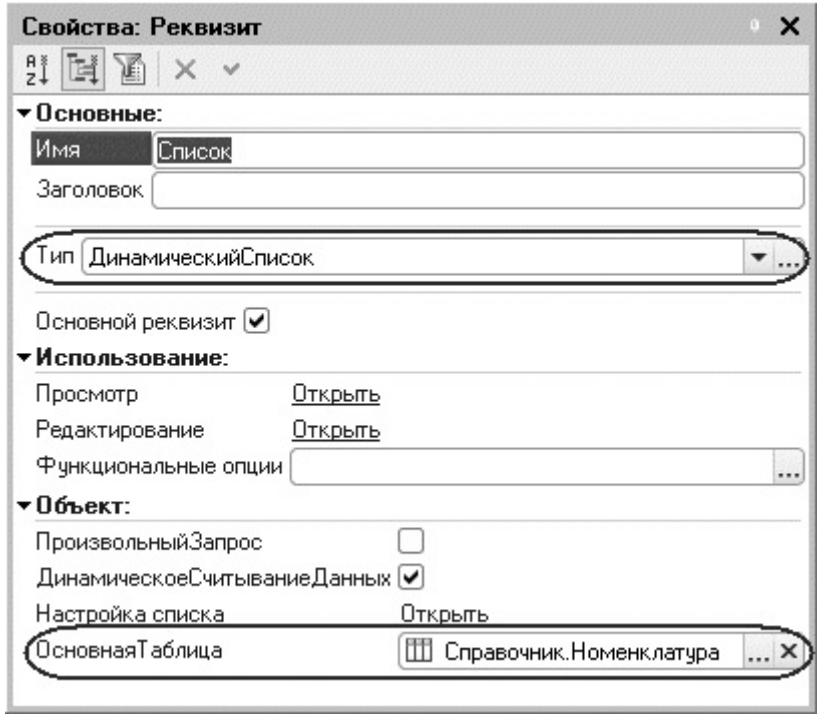


Рис. 27.3. Основной реквизит формы списка

Поэтому к свойствам, методам и событиям объекта встроенного языка *УправляемаяФорма* добавляется *Расширение динамического списка* (рис. 27.4).

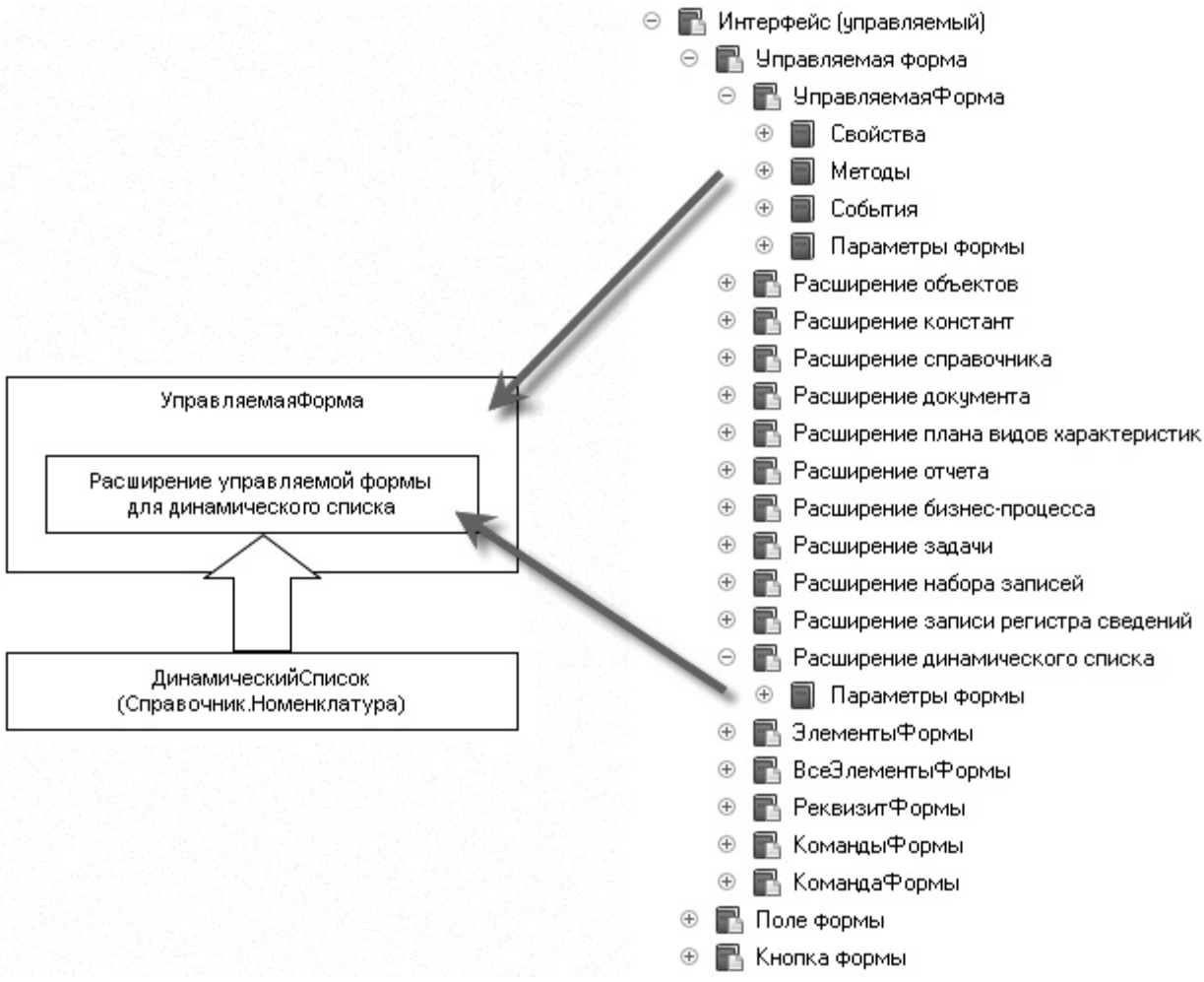


Рис. 27.4. Контекст формы дополняется контекстом расширения

В результате этого у формы появляются такие параметры, как *ТекущаяСтрока, Отбор* и т. п.

Теперь посмотрим на таблицу *Список*.

Поскольку в таблице отображается динамический список, то к свойствам, методам и событиям объекта встроенного языка *ТаблицаФормы* добавляется *Расширение динамического списка* (рис. 27.5).

Рис. 27.5. Контекст таблицы дополняется контекстом расширения

В результате у таблицы *Список* появляются такие свойства, как *АвтоОбновление*, *ОтображатьКорень* и т. д.

И в заключение посмотрим на поле *ВидНоменклатуры*.

Это поле связано с реквизитом типа *ПеречислениеСсылка.ВидыНоменклатуры* и является полем ввода.

Поэтому к свойствам, методам и событиям объекта встроенного языка *ПолеФормы* добавляется *Расширение поля ввода* (рис. 27.6).

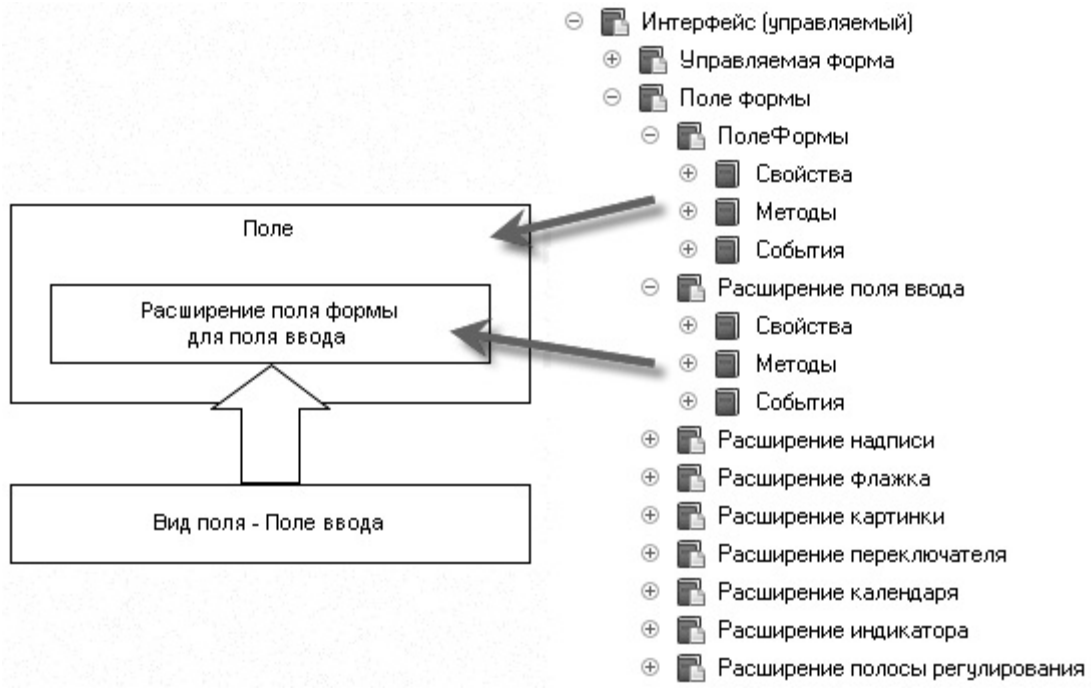


Рис. 27.6. Связь формы и ее элементов с данными

В результате у поля *ВидНоменклатуры* появляются такие свойства, как

БыстрыйВыбор, *ВыделенныйТекст* и т. д.

Типы данных формы

В управляемой форме можно выделить следующие категории типов, с которыми она работает:

- Типы встроенного языка, предназначенные для использования как в управляемых формах, так и вне их. Например, *Число, СправочникСсылка.<имя>, ГрафическаяСхема, ТабличныйДокумент* и т. д.
- Типы встроенного языка, предназначенные исключительно для того, чтобы представить в форме данные прикладных объектов (справочников, документов и т. д.). Это такие типы, как *ДанныеФормыСтруктура*, *ДанныеФормыКоллекция* и другие.
- Отдельно следует упомянуть тип *ДинамическийСписок*, который используется в управляемых формах для отображения списков прикладных объектов.

Все типы прикладных объектов (такие как *СправочникОбъект* и т. д.) не существуют на стороне тонкого и веб-клиентов, они существуют только на сервере. Однако данные этих объектов нужно отображать в управляемых формах.

Поэтому для представления в форме данных этих прикладных типов введены специальные типы данных, предназначенные для работы именно в управляемых формах. Используются следующие типы данных:

- *ДанныеФормыСтруктура* – содержит набор свойств произвольного типа. Свойствами могут быть другие структуры, коллекции или структуры с коллекциями. Таким типом представляется, например, в форме *СправочникОбъект*.
- *ДанныеФормыКоллекция* – это список типизированных значений, похожий на массив. Доступ к элементу коллекции осуществляется по индексу или по идентификатору. Доступ по идентификатору может отсутствовать в некоторых случаях. Это обусловлено типом прикладного объекта, который представлен этой коллекцией. Идентификатором может быть любое целое число. Таким типом представляется, например, в форме табличная часть.
- *ДанныеФормыСтруктураСКоллекцией* – это объект, который представлен в виде структуры и коллекции одновременно. С ним можно обращаться как с любой из этих сущностей. Таким типом представляется, например, в форме набор записей.
- *ДанныеФормыДерево* – объект предназначен для хранения иерархических данных.

Прикладной объект представлен либо одним, либо несколькими элементами

данных формы. В общем виде иерархия и состав данных формы зависят от сложности и взаимосвязи прикладных объектов управляемой формы.

Например, документ, содержащий табличную часть, будет представлен объектом типа *ДанныеФормыСтруктура* (собственно документ), которому подчинен объект типа *ДанныеФормыКоллекция* (табличная часть документа).

ВНИМАНИЕ!

Во время разработки конфигурации важно помнить, что прикладные объекты доступны только на сервере, в то время как объектами данных форм можно пользоваться и на сервере, и на клиенте.

Фактически можно сказать, что данные формы – это унифицированное представление данных различных прикладных объектов, с которыми форма работает единообразно и которые присутствуют и на сервере, и на клиенте.

В редакторе формы (у реквизитов формы) вместо имен этих типов обычно отображаются те прикладные типы, данные которых содержит реквизит.

Например, если реквизит *Объект* содержит данные элемента справочника товары, то в колонке *Тип* отображается ненастоящий тип этого реквизита формы – *ДанныеФормы Структура*, а тип прикладного объекта, данные которого содержатся в этом реквизите – *СправочникОбъект.Клиенты.* Причем

чтобы было понятно, что это «ненастоящий тип» реквизита, тип прикладного объекта показывается в круглых скобках.

Таким образом форма содержит некоторую «проекцию» данных прикладных объектов в виде своих собственных типов данных и автоматически выполняет преобразование между ними при необходимости.

Однако в случае если разработчик конфигурации реализует свой алгоритм обработки данных, то преобразование данных (из специализированных типов в прикладные и обратно) он должен выполнять самостоятельно.

Для конвертирования прикладных объектов в данные формы и обратно существует набор глобальных методов:

- *ЗначениеВДанныеФормы()* – преобразует объект прикладного типа в данные формы;
- *ДанныеФормыВЗначение()* – преобразует данные формы в объект прикладного типа;

Аналогичные методы, предназначенные для конвертирования значений реквизитов формы в прикладные объекты и обратно, существуют и у самой управляемой формы:

- *ЗначениеВРеквизитФормы()* – преобразует объект прикладного типа в реквизит управляемой формы;
- *РеквизитФормыВЗначение()* – преобразует реквизит управляемой формы в значение прикладного типа.

Методы, работающие с прикладными объектами, доступны только в серверных процедурах формы.

При выполнении стандартных действий формы с основным реквизитом (открытие формы, выполнение стандартной команды *Записать* и т. д.) преобразование выполняется автоматически.

Приведем пример преобразования данных, которое может потребоваться в собственных алгоритмах.

Например, у нас есть особенная форма, в которой в качестве одного из реквизитов (*ТоварДляМодификации)* используются данные элемента справочника *Товары*. При создании формы на сервере мы по некоторому алгоритму определяем, какой именно это товар, и читаем его данные в реквизит формы. При этом используется преобразование данных *ЗначениеВДанныеФормы(),* листинг 27.1.

Листинг 27.1. Пример преобразования данных прикладных объектов в данные формы

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

  ОбъектТовар =
  Справочники.Товары.НайтиПоНаименованию("Кофейник").ПолучитьОбъект();
  ЗначениеВДанныеФормы(ОбъектТовар, ТоварДляМодификации);

КонецПроцедуры

&НаКлиенте
Процедура Записать()

  ЗаписатьНаСервере();

КонецПроцедуры

&НаСервере
Процедура ЗаписатьНаСервере()

  ОбъектТовар = ДанныеФормыВЗначение(ТоварДляМодификации,
  Тип("СправочникОбъект.Товары"));
  ОбъектТовар.Записать();

КонецПроцедуры
```

В некоторый момент работы формы мы решаем, что измененные данные нашего товара необходимо записать в базу данных, и тогда выполняем обратное преобразование данных формы в прикладной объект

(*ДанныеФормыВЗначение()*) и записываем его.

Как мы уже упомянули, у формы также есть методы, позволяющие преобразовать прикладные данные в реквизит формы и наоборот.

Использование данных методов обычно удобнее, так как они имеют, например, информацию о типе реквизита формы. Кроме этого, метод *РеквизитФормыВЗначение()* выполняет установку соответствия данных формы и объекта, которая используется при формировании сообщений.

Приведем пример использования этих методов. В серверной процедуре формы мы получаем прикладной объект из реквизита формы и выполняем метод этого прикладного объекта *Пересчитать()*. Затем данные объекта, измененные в результате пересчета, преобразуем обратно в реквизит формы (листинг 27.2).

Листинг 27.2. Пример преобразования данных прикладных объектов в данные формы

```
&НаСервере
Процедура ПересчитатьНаСервере()

  // Преобразует реквизит Объект в прикладной объект.
  Документ = РеквизитФормыВЗначение("Объект");

  // Выполняет пересчет методом, определенным в модуле документа.
  Документ.Пересчитать();
```

```
    // Преобразует прикладной объект обратно в реквизит.
    ЗначениеВРеквизитФормы(Документ, "Объект");

КонецПроцедуры
```

Связанные списки

При создании прикладных решений часто возникает необходимость из какой-либо формы прикладного объекта перейти к информации, логически связанной с этим объектом.

Это может быть, например, список подчиненного справочника; регистры, в которых объект производит движения; регистры, где измерение с типом этого объекта указано как ведущее; критерии отбора, в которые входит этот тип; объекты, которые можно ввести на основании этого типа, и т. д.

Одним из распространенных примеров использования связанных списков является необходимость перейти из формы документа к списку движений, которые произвел этот документ в каком-либо регистре.

Все перечисленные ситуации платформа контролирует автоматически. На основании информации, содержащейся в объектах конфигурации, платформа автоматически создает в формах объектов команды для перехода к связанной

информации. Некоторые такие команды она сразу же делает видимыми, и они появляются в форме. Некоторые команды она только создает, но по умолчанию не включает их видимость.

Поэтому чаще всего, если в форме вы не видите команды перехода к связанной информации, которая должна быть, просто нужно включить ее видимость в интерфейсе формы.

Рассмотрим это на примере.

В режиме «Конфигуратор»

Для примера откроем форму документа *ОказаниеУслуги*. В левом верхнем окне перейдем на закладку *Командный интерфейс*.

В разделе *Панель навигации* в группе *Перейти* уже находится набор таких команд. Это команды перехода к записям регистров, для которых этот документ является регистратором.

Мы можем установить общую видимость для этих команд, а можем более точно настроить видимость этих команд для каждой отдельной роли, которая есть в нашей конфигурации (рис. 27.7).

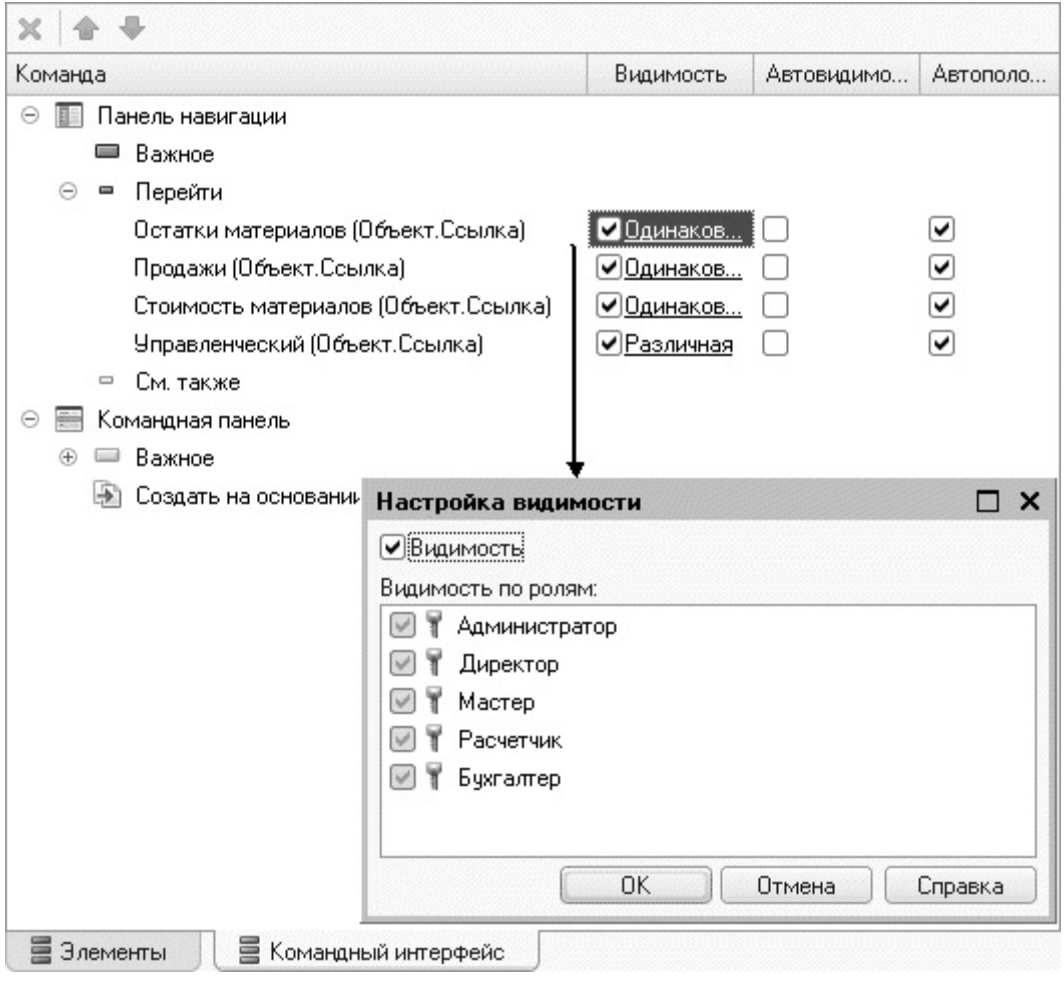


Рис. 27.7. Командный интерфейс формы

В режиме «1С:Предприятие»

В режиме *1С:Предприятие* откроем один из документов *Оказание услуги* (рис. 27.8).

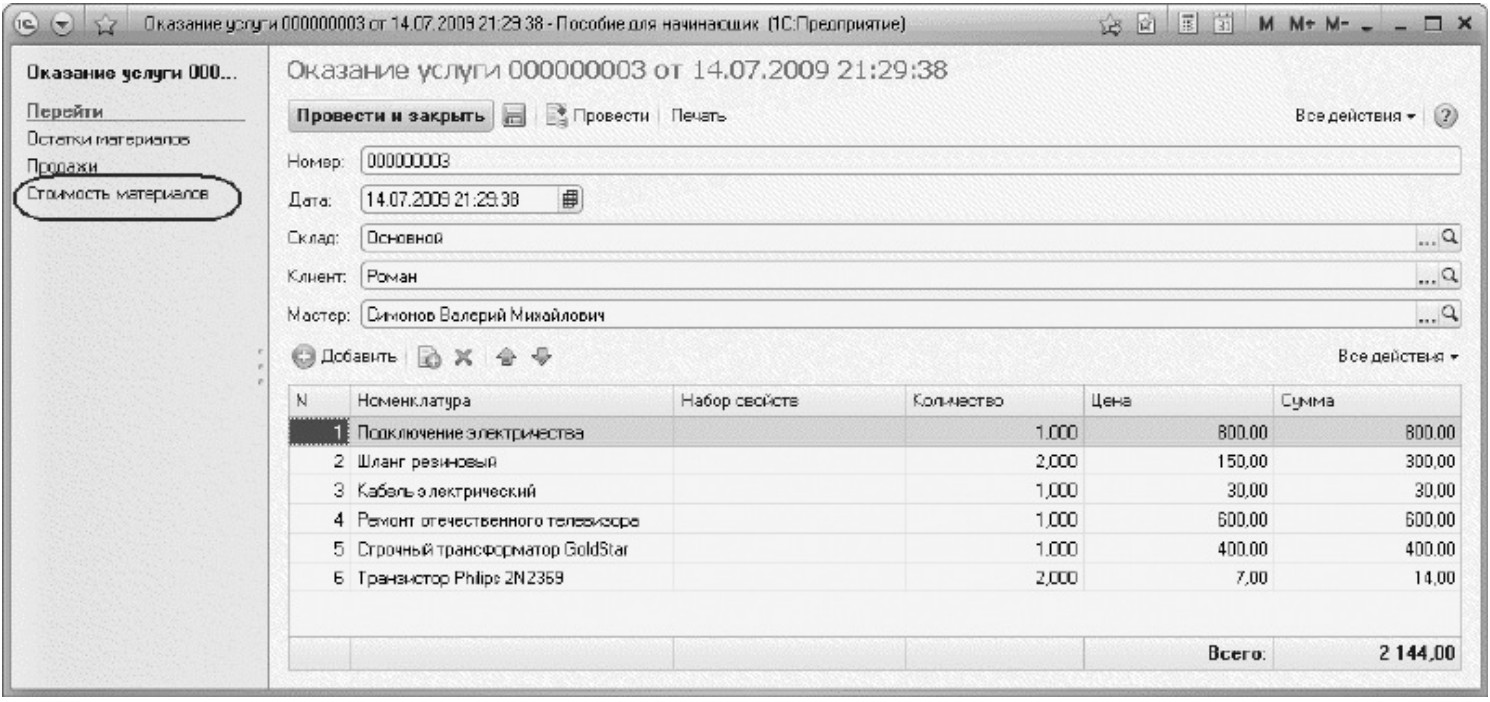


Рис. 27.8. Документ «Оказание услуги № 3»

В панели навигации формы мы видим команды перехода к списку записей регистров, связанных с открытым документом.

Например, выполним команду *Стоимость материалов* и перейдем к

движениям, которые произвел в этом регистре наш документ (рис. 27.9).

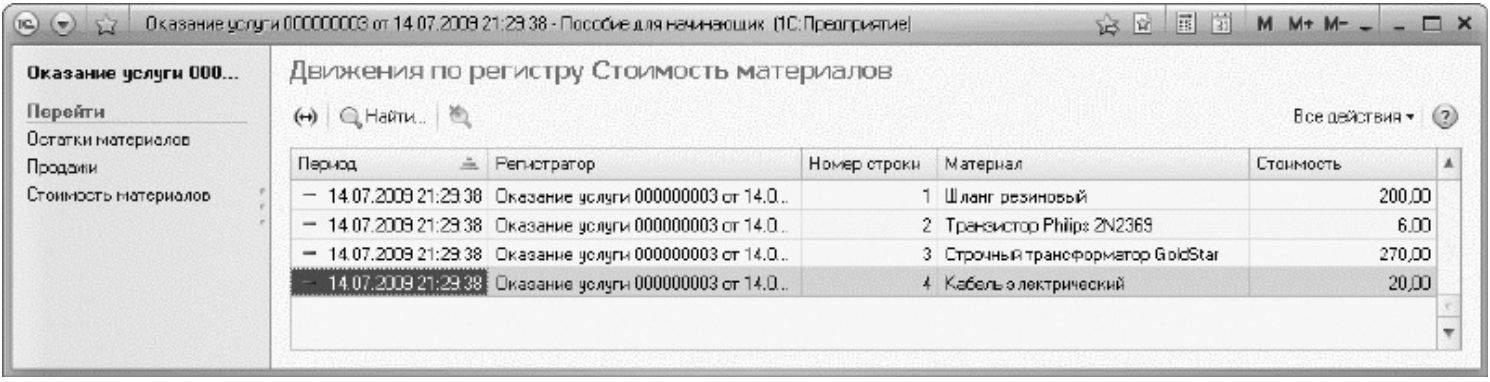


Рис. 27.9. Движения документа «Оказание услуги № 3» по регистру «Стоимость материалов»

Оформление строк в форме списка

Одним из полезных свойств формы списка является возможность настройки оформления его строк. Для иллюстрации этой возможности мы воспользуемся формой списка справочника *Номенклатура* и придадим ей нестандартный вид.

В режиме «Конфигуратор»

Откроем в конфигураторе форму списка справочника *Номенклатура* и создадим обработчик события формы *ПриСозданииНаСервере*.

Внесем в него следующий текст (листинг 27.3).

Листинг 27.3. Обработчик события формы «При открытии»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

  СписокСправочника = Элементы.Список;

  // Задать режим отображения справочника.
  СписокСправочника.Отображение = ОтображениеТаблицы.Список;

  // Скрыть линии сетки.
  СписокСправочника.ВертикальныеЛинии = Ложь;
  СписокСправочника.ГоризонтальныеЛинии = Ложь;

КонецПроцедуры
```

В этой процедуре, выполняющейся при создании формы на сервере, мы сначала представляем список в виде обычного, а не иерархического списка. Это сделано для большей наглядности, чтобы материалы отображались вперемешку с услугами.

Затем мы скрываем линии, разделяющие колонки и строки таблицы списка.

Теперь настроим условное оформление строк списка. Для этого вызовем палитру свойств основного реквизита формы *Список*.

В строке *Настройка списка* нажмем *Открыть* (рис. 27.10).

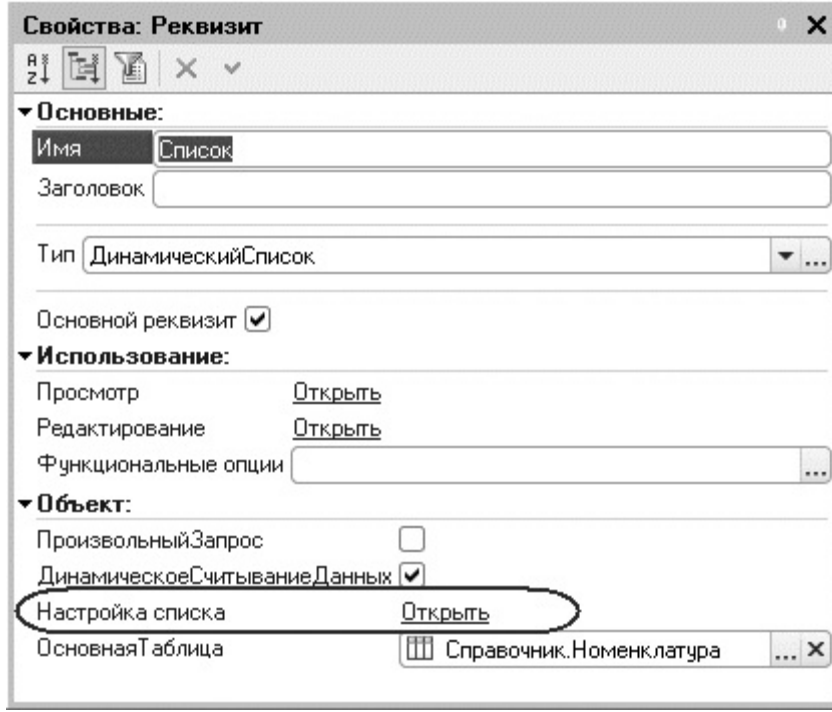


Рис. 27.10. Вызов настройки динамического списка

Поскольку этот реквизит имеет тип *ДинамическийСписок*, который построен на основе системы компоновки данных, то мы можем настроить для него *Отбор, Порядок, Группировку* и *УсловноеОформление*, аналогично тому, как это делалось в отчетах.

В открывшемся окне настройки динамического списка перейдем на закладку *Условное оформление* и нажмем кнопку *Добавить* в командной панели окна.

Сначала укажем *Оформление* для выделения полей.

Нажмем кнопку выбора в поле *Оформление* и установим сиреневый цвет фона (рис. 27.11).

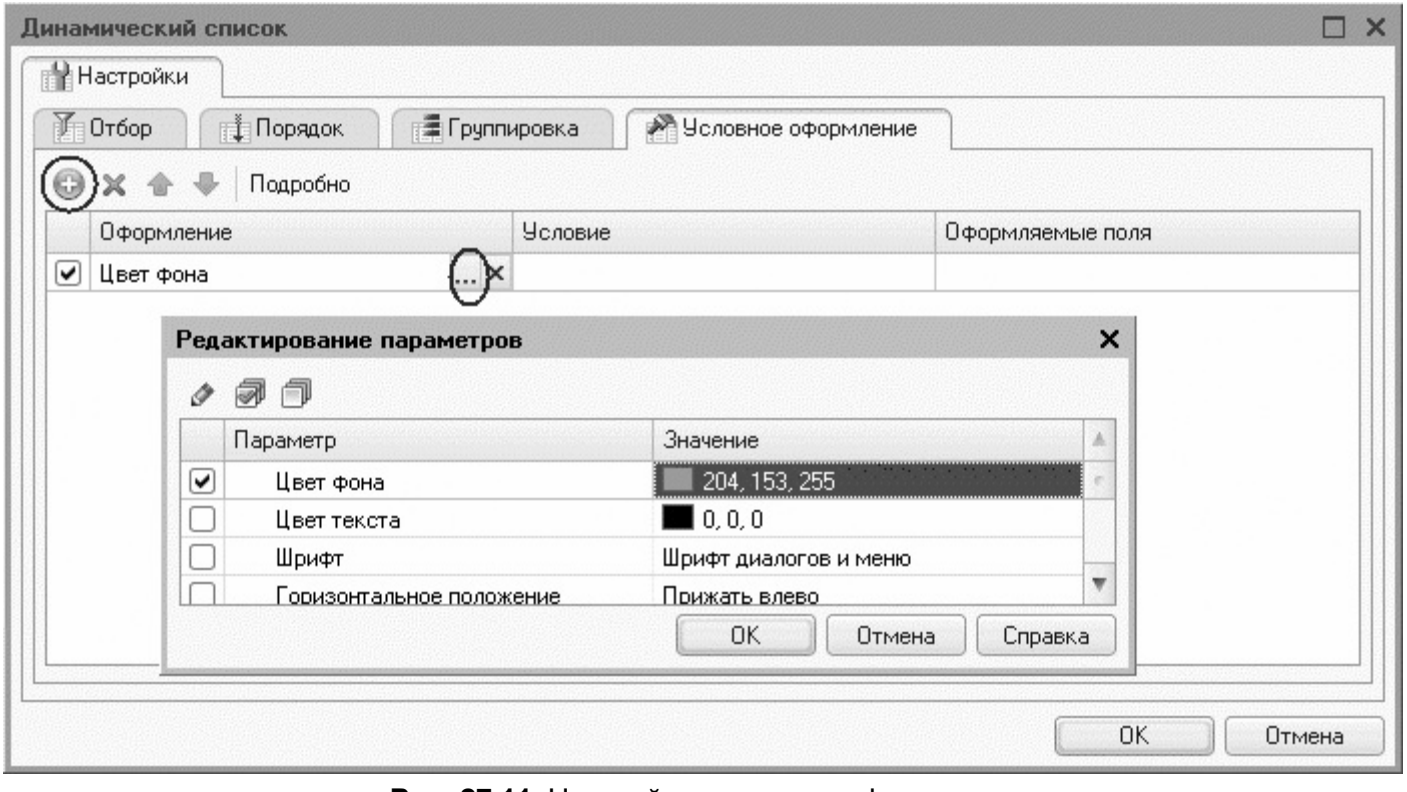


Рис. 27.11. Настройка условного оформления

Нажмем *ОК*. Затем укажем условие, при наступлении которого будет применяться оформление, то есть когда строки списка будут сиреневыми.

Нажмем кнопку выбора в поле *Условие* и в появившемся окне добавим *Новый элемент* отбора (рис. 27.12). Для этого нажмем кнопку *Добавить* и укажем в

графе *Левое значение* – поле *ВидНоменклатуры*, в графе *Вид сравнения* – *Равно*, а в графе *Правое значение* выберем *Перечисление.ВидыНоменклатры.Услуга*.

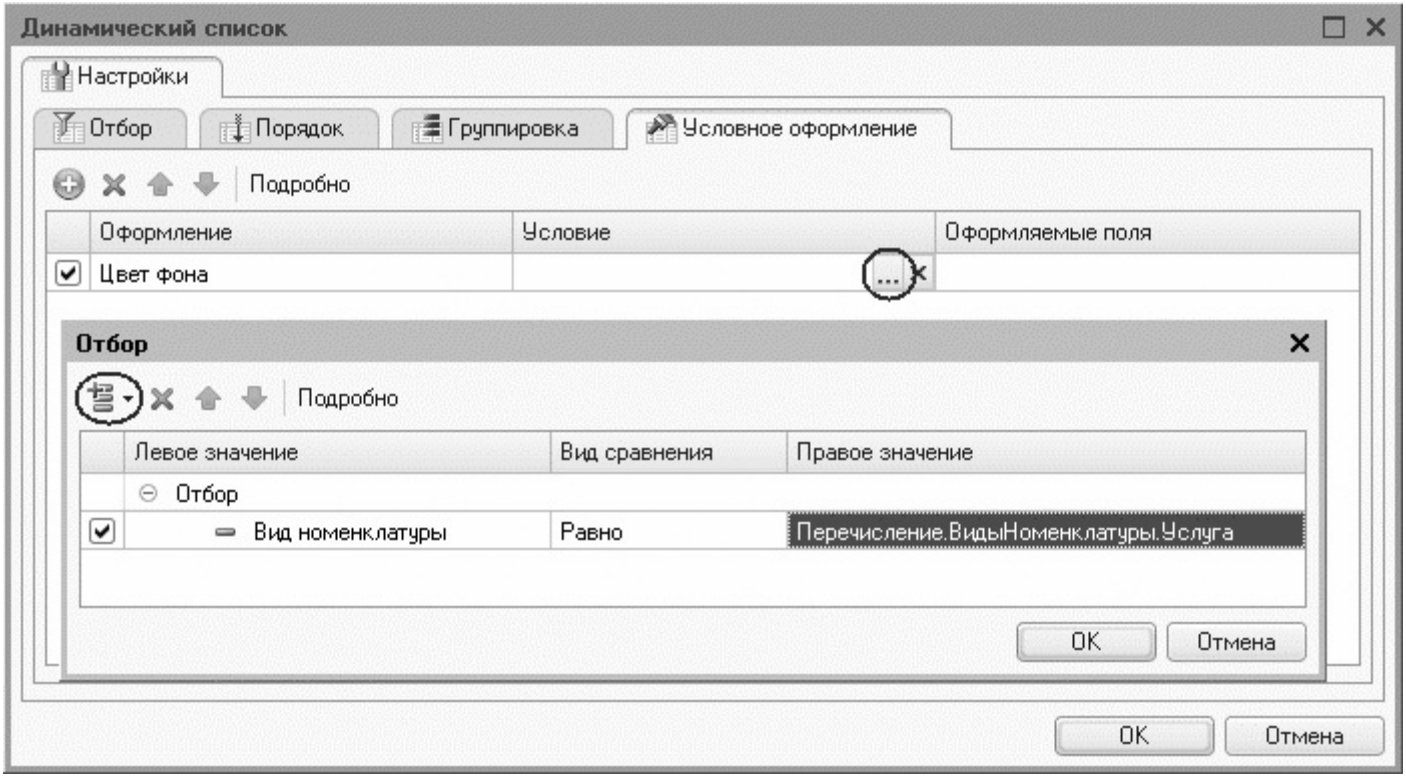


Рис. 27.12. Настройка условного оформления

Нажмем *ОК*.

Таким образом, мы установили, что когда в списке номенклатуры будут отражаться услуги, эти строки будут выделены сиреневым фоном.

Поскольку мы хотим выделить полностью строки, а не отдельные поля списка, то список оформляемых полей можно оставить пустым. Нажмем *ОК*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем список номенклатуры.

Мы видим, что список имеет вид обычного неиерархического списка, услуги выделены сиреневым цветом, а также отсутствуют разделительные линии строк и колонок списка (рис. 27.13).

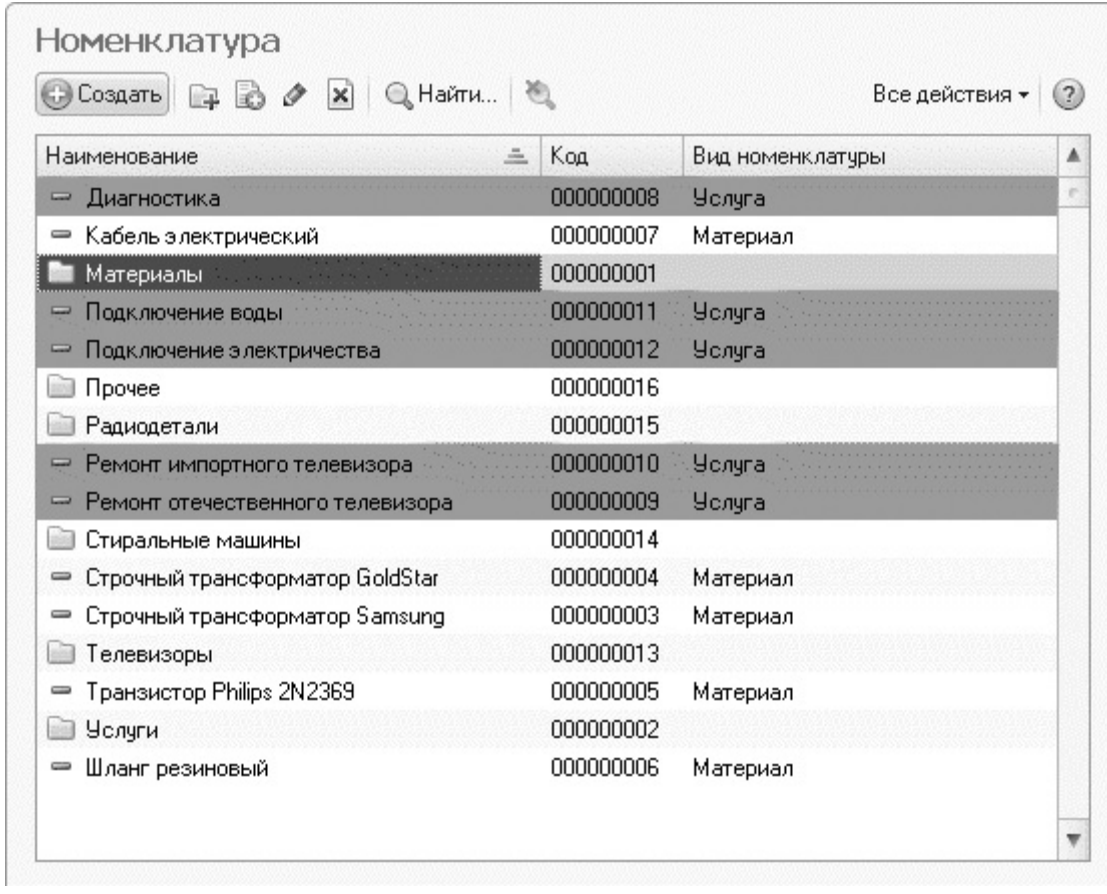


Рис. 27.13. Список номенклатуры с заданным оформлением

Пользователь может изменить заданное оформление списка, выполнив команду *Все действия > Настроить список…* Откроется окно настройки динамического списка, аналогичное окну в конфигураторе, где он может задать

свое условное оформление списка и/или другие настройки.

Аналогичным образом можно настраивать условное оформление и для списков, источником которых является не динамический список, а другие типы данных. Например условное оформление табличной части документа.

Но выполняется это уже с помощью условного оформления самой формы. То есть в дереве элементов формы нужно выделить корневой элемент и в палитре свойств открыть ссылку *УсловноеОформление*.

Создавая элементы условного оформления формы, нужно помнить, что, в отличие от системы компоновки данных и от условного оформления динамических списков, в форме обязательно нужно указать те поля, которые должны быть оформлены.

Вычисляемые колонки в списках

Необходимость вывода произвольных данных в колонках списка возникает, когда вместе с элементом списка нужно отобразить некоторую вычисляемую информацию.

Мы рассмотрим эту ситуацию на примере отображения актуальной цены в списке справочника *Номенклатура*.

Эти данные мы можем получить из таблицы регистра сведений *Цены.СрезПоследних*. Следовательно, поле *Цена* из этой таблицы нам нужно добавить в динамический список *Список*, который является основным реквизитом формы списка номенклатуры и служит источником данных для таблицы списка.

В режиме «Конфигуратор»

Откроем в конфигураторе форму списка справочника *Номенклатура* и вызовем палитру свойств основного реквизита формы *Список*.

До сих пор в свойствах динамического списка была указана *Основная таблица – Справочник.Номенклатура* (см. рис. 27.10), и список формировался путем запроса к этой таблице.

Теперь нам нужна еще связанная информация из таблицы регистра сведений *Цены.СрезПоследних*.

Поэтому установим флажок *ПроизвольныйЗапрос* и в строке *Настройка списка* нажмем *Открыть* (рис. 27.14).

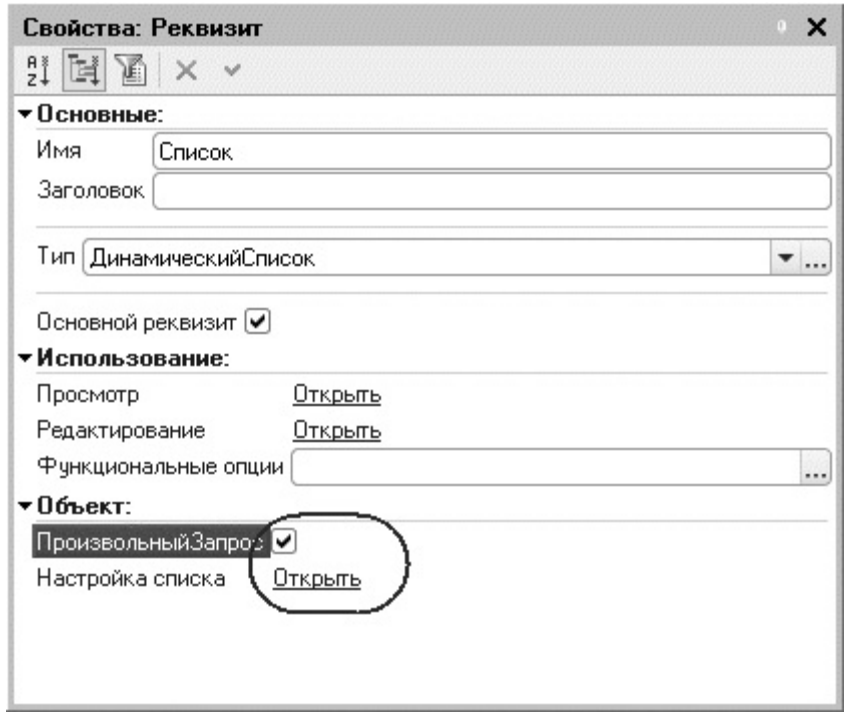


Рис. 27.14. Вызов настройки динамического списка

Откроется окно настройки динамического списка. На закладке *Запрос* мы видим запрос, в котором выбираются все поля из таблицы *Справочник.Номенклатура*.

Изменим его. Для этого нажмем кнопку *Конструктор запроса* (рис. 27.15).

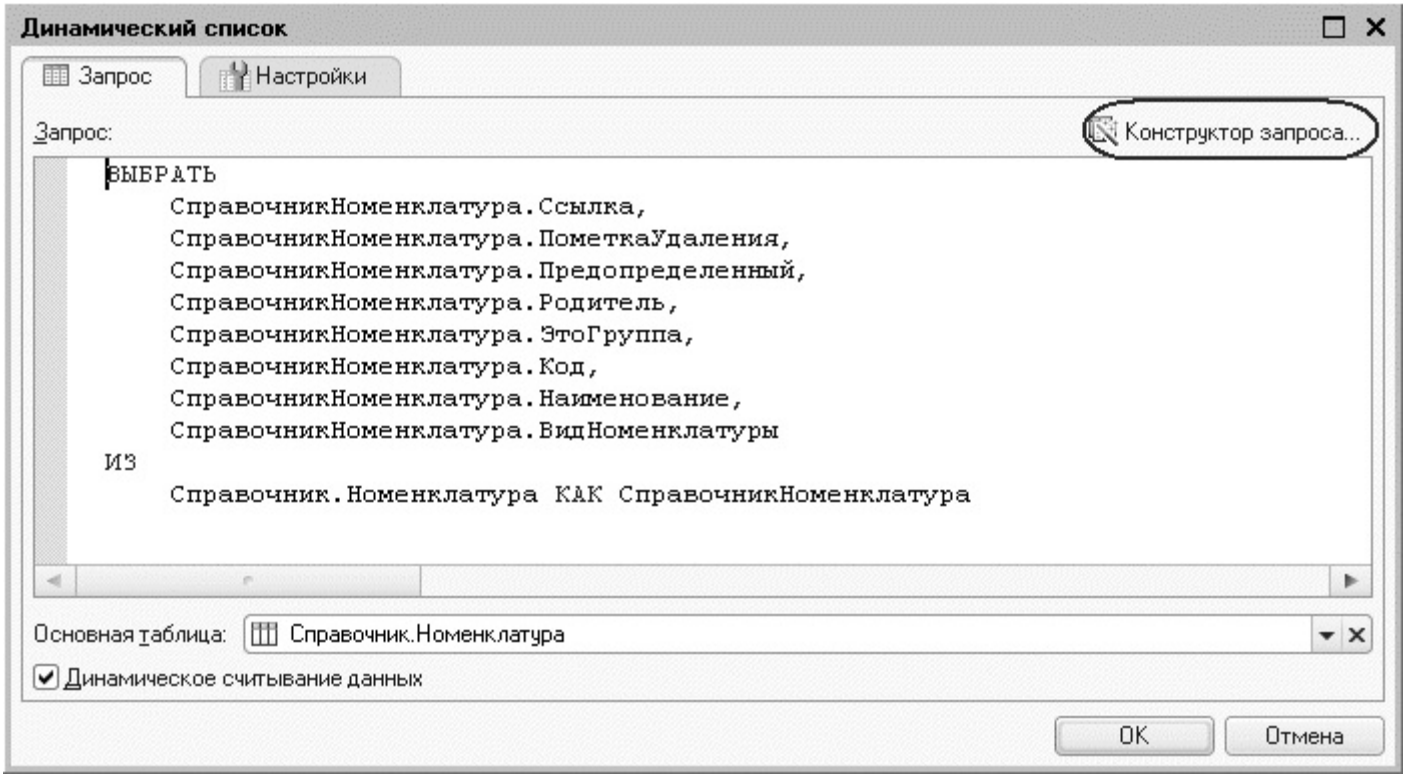


Рис. 27.15. Создание произвольного запроса для динамического списка

Добавим в список таблиц *Цены.СрезПоследних* и выберем из нее поле *Цена* (рис. 27.16).

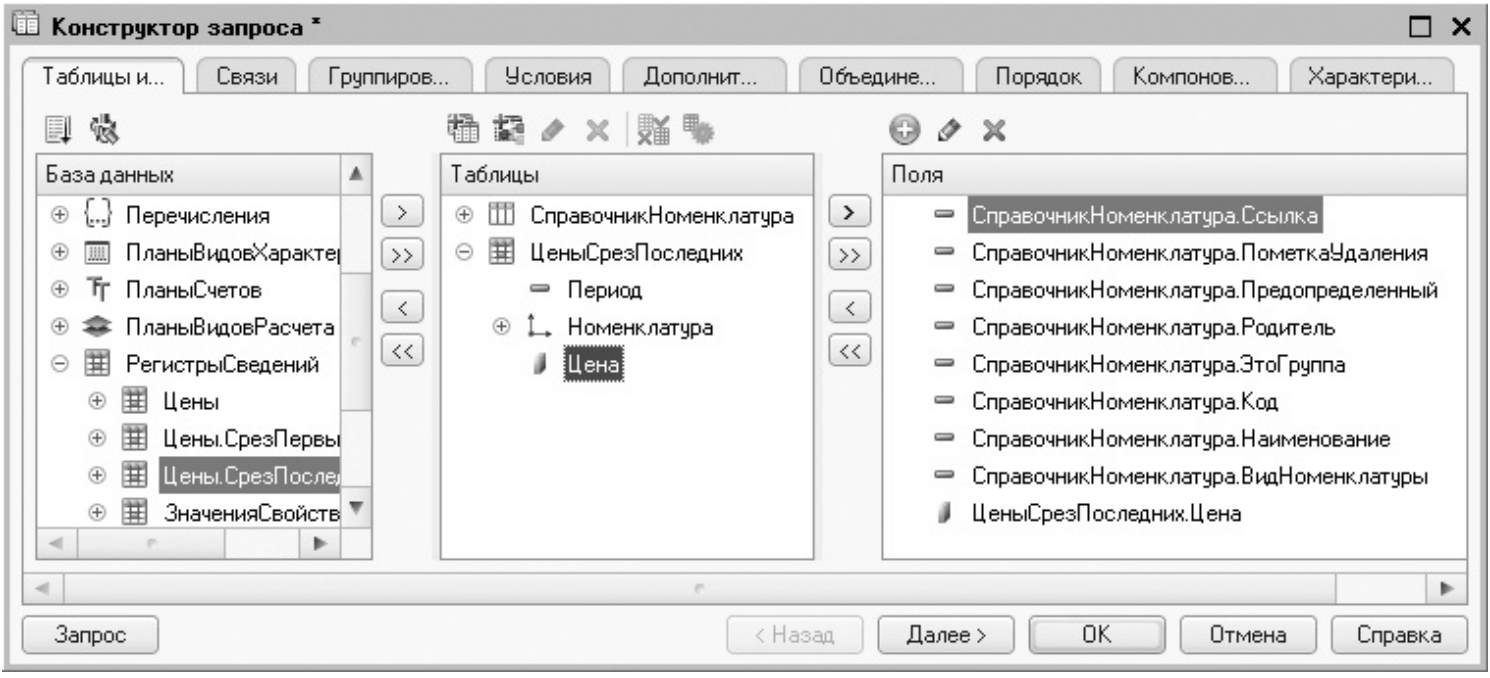


Рис. 27.16. Добавление второй таблицы

На закладке *Связи* отредактируем связь между таблицами, созданную по умолчанию.

Установим флажок *Все* у таблицы *Справочник.Номенклатура* и снимем его у таблицы *Цены.СрезПоследних* (рис. 27.17).

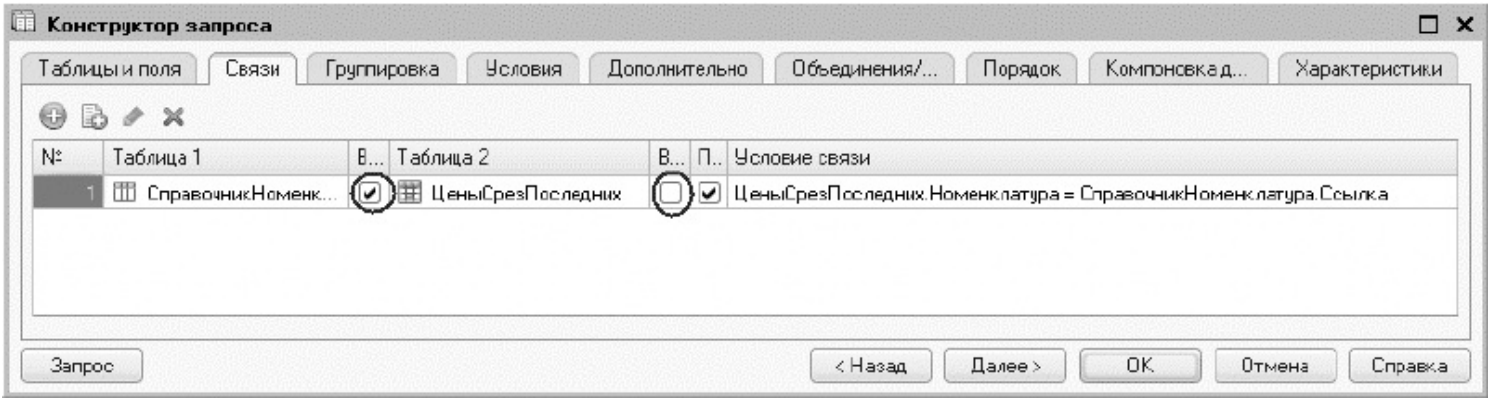


Рис. 27.17. Установка связи между таблицами

Тем самым мы задаем, что в списке номенклатуры будут отражены все позиции, даже те, по которым не установлены цены.

Похожие действия мы уже выполняли на занятии 13 «Отчеты» для отчета «Перечень услуг».

Создание запроса закончено, нажмем *ОК*. Текст запроса нам уже знаком и понятен, поэтому не будем на нем останавливаться.

Теперь колонка *Цена*, содержащая актуальную цену, будет отображаться в списке номенклатуры, когда мы разместим ее в форме списка.

В окне настройки динамического списка перейдем на закладку *Настройки* и

зададим условное оформление этой колонки так, чтобы низкие цены выделялись цветом.

Для этого перейдем на закладку *Условное оформление*. Там мы видим созданное нами ранее условное оформление для строк списка. Нажмем кнопку *Добавить* в командной панели окна.

Сначала укажем *Оформление* для выделения полей. Нажмем кнопку выбора в поле *Оформление* и установим синий цвет текста.

Затем укажем условие, при наступлении которого будет применяться оформление, то есть когда текст в колонке *Цена* будет синим.

Нажмем кнопку выбора в поле *Условие* и в появившемся окне добавим *Новый элемент* отбора (рис. 27.12). Для этого нажмем кнопку *Добавить* и укажем в графе *Левое значение* – поле *Цена*, в графе *Вид сравнения* – *Меньше*, а в графе *Правое значение* выберем *500*.

Затем укажем список оформляемых полей. Нажмем кнопку выбора в поле *Оформляемые поля*, затем нажмем *Добавить* и выберем поле *Цена* (рис. 27.18).

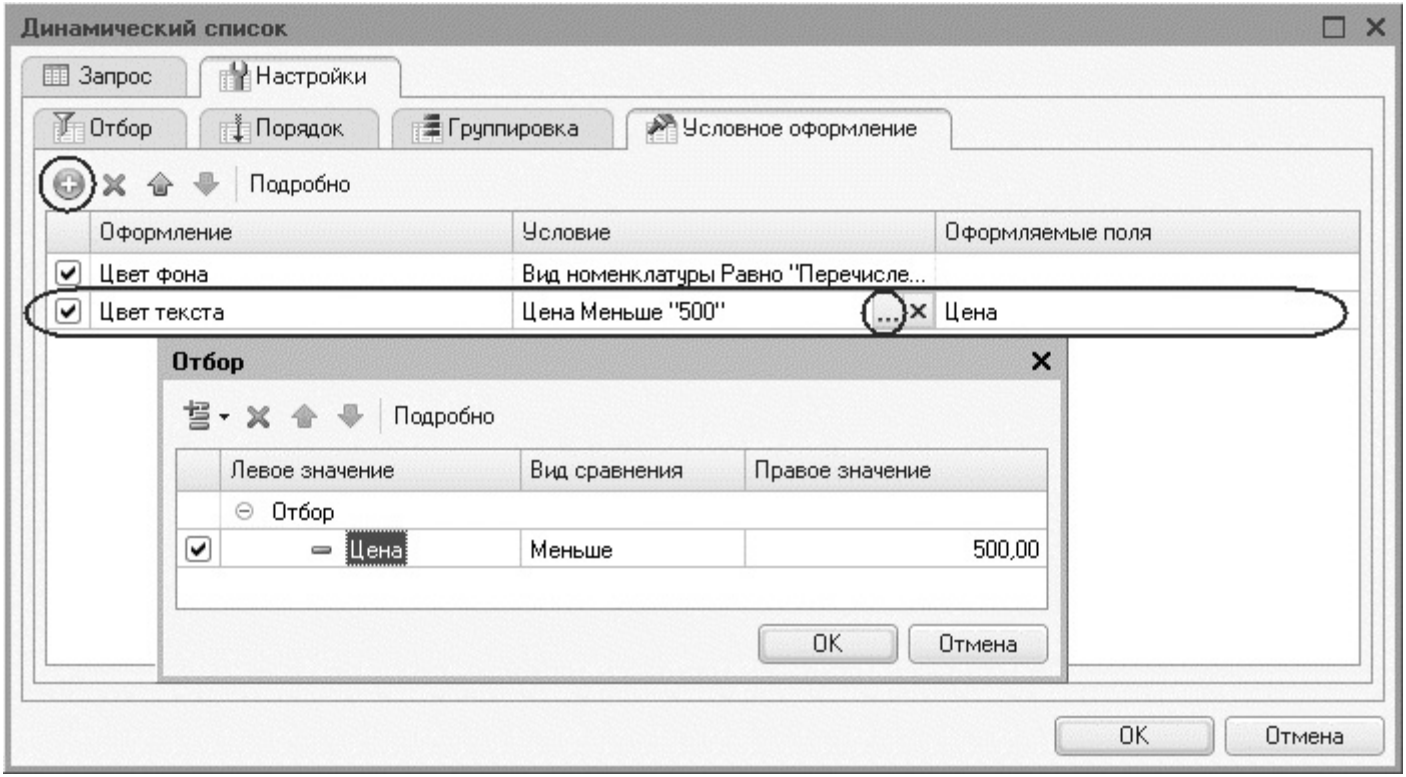


Рис. 27.18. Настройка условного оформления динамического списка

Нажмем *ОК*. Теперь нам осталось только перетащить поле *Цена* из окна реквизитов в окно реквизитов формы (рис. 27.19).

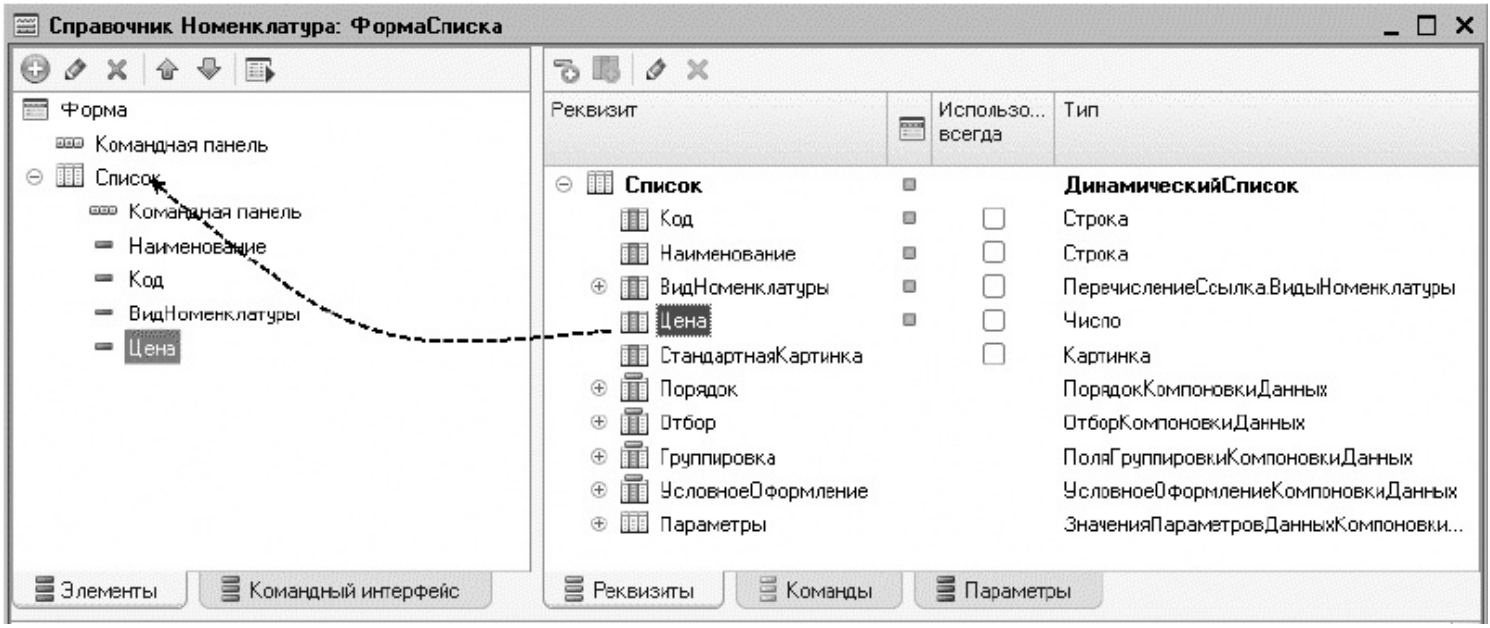


Рис. 27.19. Добавление колонки в форму списка

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем список номенклатуры. Мы видим, что вместе с номенклатурой выводится ее актуальная цена, причем цены на номенклатуру меньше *500* выделены синим цветом (рис. 27.20).

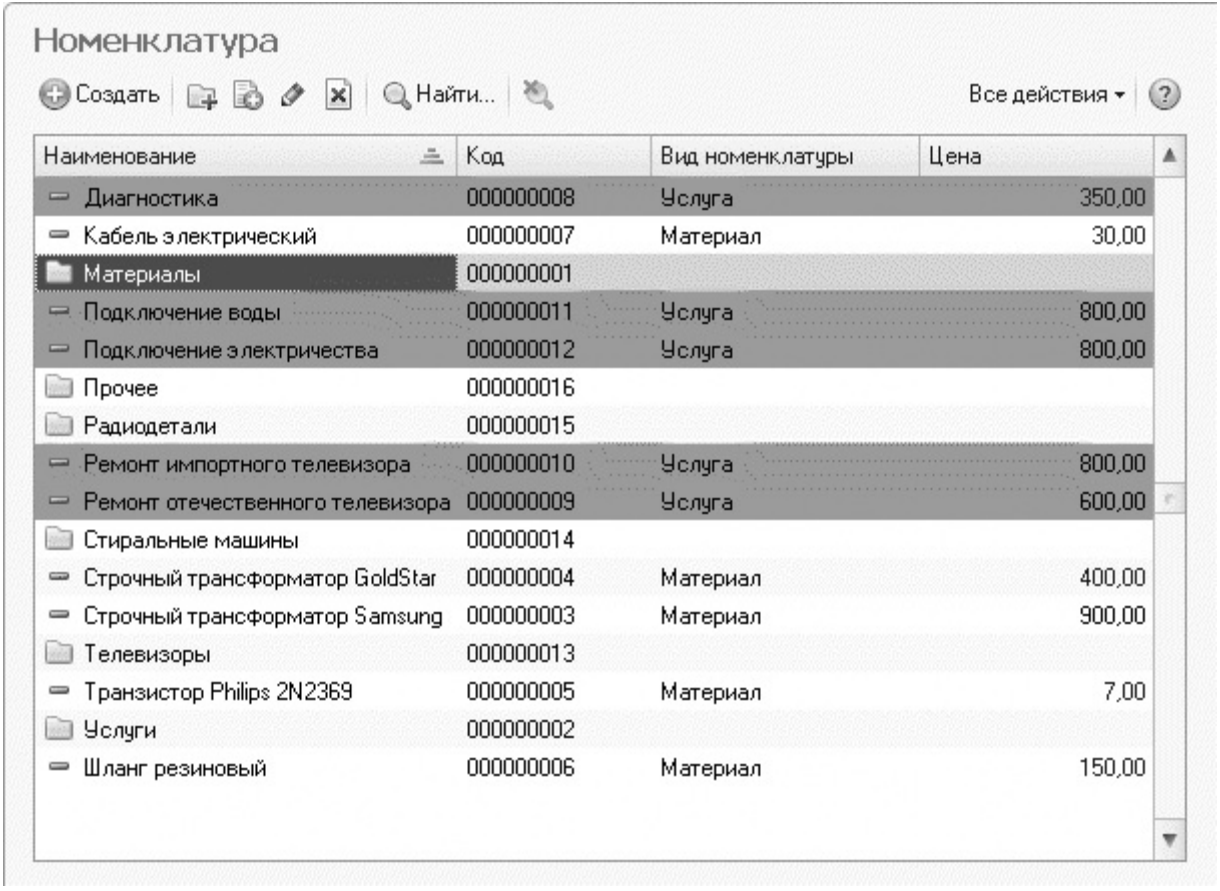


Рис. 27.20. Список номенклатуры с заданным оформлением

Список выбора для поля ввода

Одним из полезных и удобных механизмов поля ввода является использование

списка выбора для этих полей.

Поля ввода, имеющие ссылочный тип, например, *Склад*, *Сотрудник*, по умолчанию имеют кнопку выбора. При нажатии этой кнопки открывается форма выбора объекта, на которую ссылается данное поле.

Но для полей других типов тоже бывает нужно выполнять выбор из нескольких предопределенных значений.

Рассмотрим совсем простой пример, когда пользователю нужно внести адреса клиентов, начинающиеся с названия города.

В режиме «Конфигуратор»

Сначала нужно добавить поле *Адрес* в форму элемента справочника *Клиенты*.

Для этого откроем в конфигураторе окно редактирования объекта конфигурации Справочник *Клиенты* и на закладке *Данные* добавим реквизит *Адрес* с типом *Строка*, длиной *25*. Затем откроем форму элемента справочника *Клиенты*. Раскроем основной реквизит формы *Объект* и перетащим поле *Адрес* из окна реквизитов в окно реквизитов формы. В открывшейся палитре свойств этого поля установим свойство *РежимВыбораИзСписка*. Затем нажмем кнопку выбора в строке *СписокВыбора* и, нажимая кнопку *Добавить*, создадим

значения, например: *Москва, Королев, Монино* (рис. 27.21).

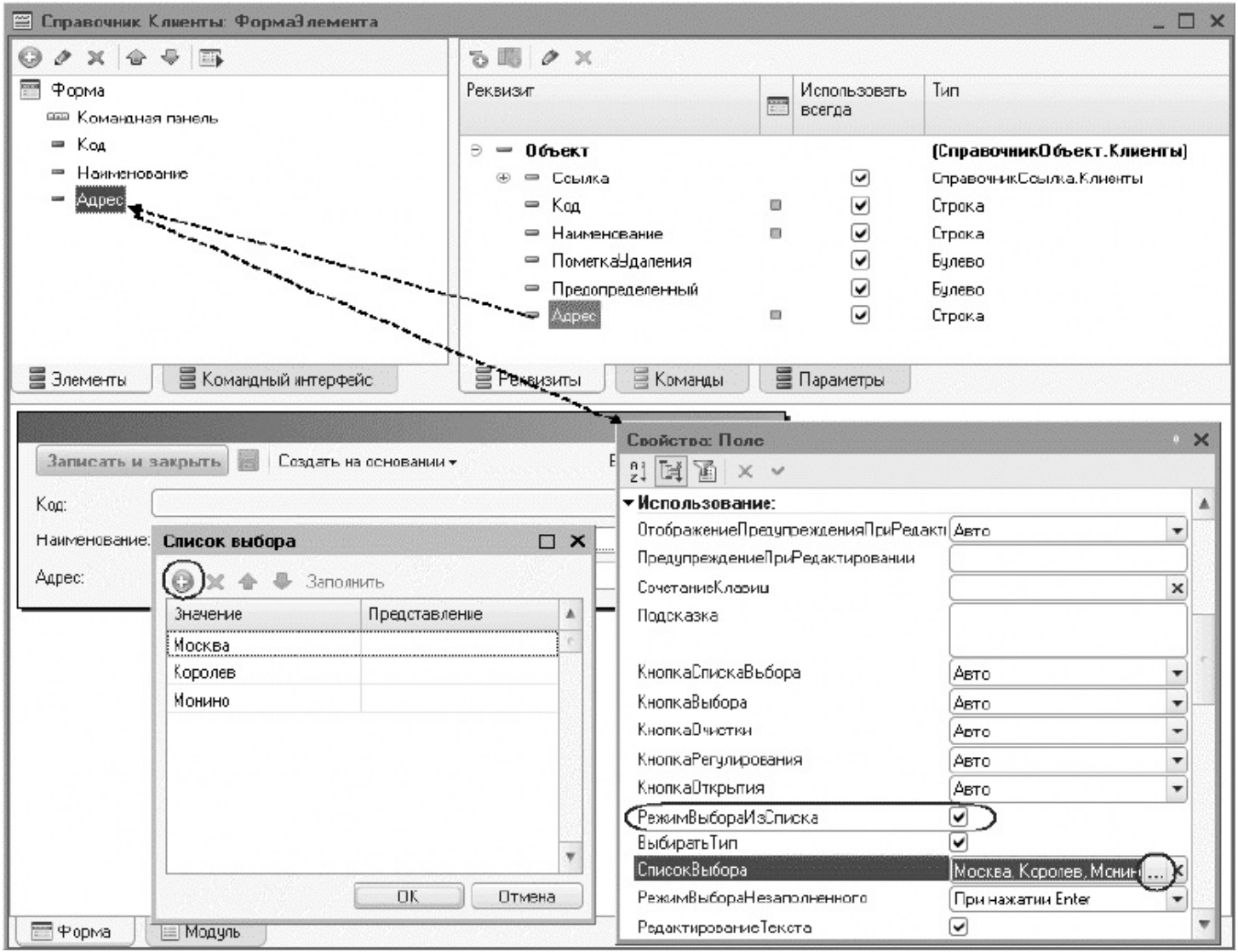

Справочник Клиенты: ФормаЭлемента
Форма
Командная панель
Код
Наименование
Адрес
Реквизит
Использовать всегда
Тип
Объект
(СправочникОбъект.Клиенты)
Ссылка
СправочникСсылка.Клиенты
Код
Строка
Наименование
Строка
ПометкаУдаления
Булево
Предопределенный
Булево
Адрес
Строка
Элементы
Командный интерфейс
Реквизиты
Команды
Параметры
Записать и закрыть
Создать на основании
Код:
Наименование:
Адрес:
Список выбора
Заполнить
Значение
Представление
Москва
Королев
Монино
OK
Отмена
Форма
Модуль
Свойство: Пол
Использование:
ОтображениеПредупрежденияПриРедакти
Авто
ПредупреждениеПриРедактировании
СочетаниеКлавиш
Подсказка
КнопкаСпискаВыбора
Авто
КнопкаВыбора
Авто
КнопкаОчистки
Авто
КнопкаРегулирования
Авто
КнопкаОткрытия
Авто
РежимВыбораИзСписка
ВыбиратьТип
СписокВыбора
Москва, Королев, Монин
РежимВыбораНезаполненного
При нажатии Enter
РедактированиеТекста

Рис. 27.21. Создание списка выбора для поля ввода

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем форму редактирования клиента. В поле *Адрес* наберем «м», и система предложит на выбор два подходящих названия (рис. 27.22).

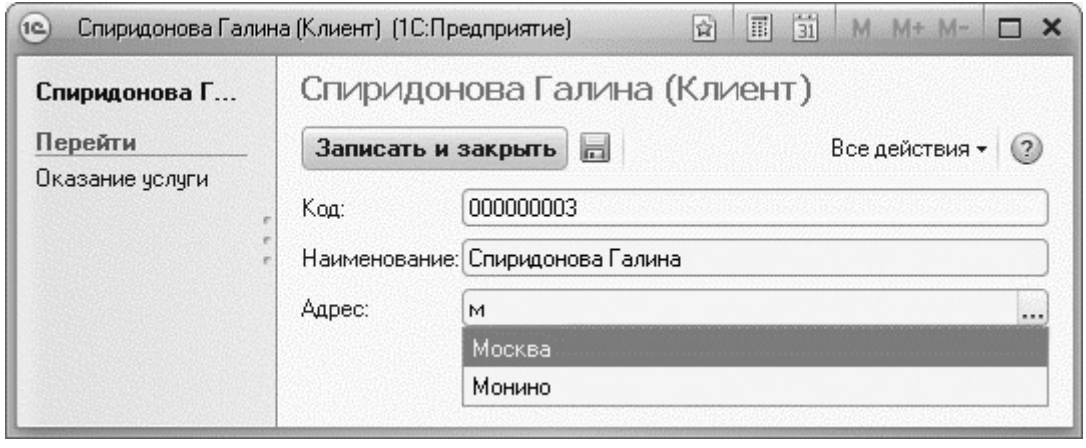


Рис. 27.22. Свойство «Проверка заполнения» реквизита «НаборСвойств»

Аналогичные действия, которые мы выполнили в конфигураторе, можно выполнить и во встроенном языке. То есть в зависимости от некоторого алгоритма можно формировать список выбора для поля ввода динамически.

Более того, разработчик из встроенного языка может изменять тот список

выбора, который формируется платформой автоматически, например в автогенерируемых формах. Для этого используется обработчик события *Обработка получения данных выбора*, который располагается в модуле менеджера объекта. Например, в модуле менеджера справочника, когда в поле ввода подбирается один из элементов этого справочника.

Форма выбора для поля, содержащего ссылочный реквизит

В процессе работы прикладного решения довольно распространенной является ситуация, когда данные вводятся в поля ссылочных реквизитов, то есть реквизитов, ссылающихся на какие-либо объекты конфигурации.

Например, в документе *Оказание услуги* мы заполняем поля ссылочного типа *Клиент* (тип *СправочникСсылка.Клиенты*), *Мастер* (тип *СправочникСсылка.Сотрудники*) и др.

При нажатии кнопки выбора в этих полях по умолчанию открывается основная форма выбора соответствующих объектов конфигурации (справочника *Клиенты*, справочника *Сотрудники* и др.).

Но иногда бывает нужно открывать свою специальную форму для выбора ссылочного реквизита. Для этого в свойстве *Форма выбора* этого реквизита

нужно указать эту специальную форму, и тогда она будет открываться для выбора данного реквизита в любой форме, где он находится.

В режиме «Конфигуратор»

Для примера рассмотрим документ *ОказаниеУслуги*. Допустим, что в поле *Мастер* нам нужно открывать не стандартную форму выбора справочника *Сотрудники*, а специально разработанную для этого форму.

Сначала эту форму нужно создать.

Откроем окно редактирования этого объекта конфигурации и перейдем на закладку *Формы*.

Мы видим, что у этого объекта вообще еще нет ни одной формы. Это значит, что все формы справочника в режиме *1С:Предприятие*, в том числе и форма выбора, генерировались системой автоматически.

Нажмем кнопку *Добавить* (рис. 27.23).

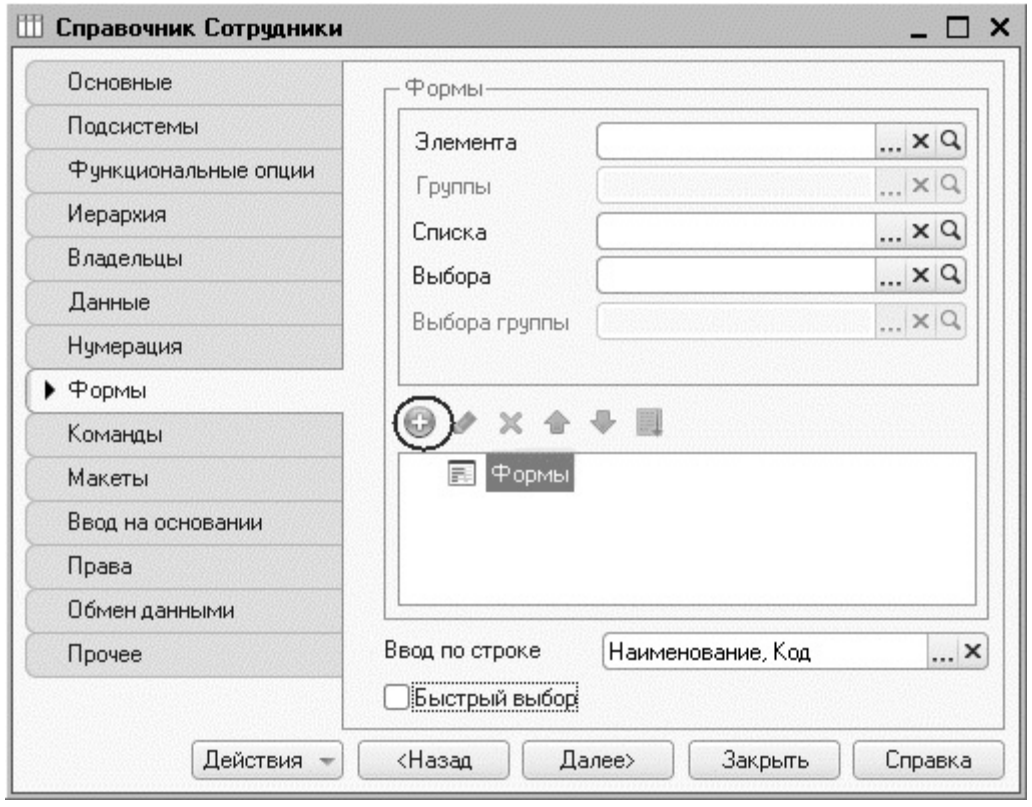


Рис. 27.23. Создание формы справочника «Сотрудники»

В открывшемся конструкторе форм выберем тип формы – *Произвольная форма* и зададим ее имя *ФормаДляВыбора* (рис. 27.24).

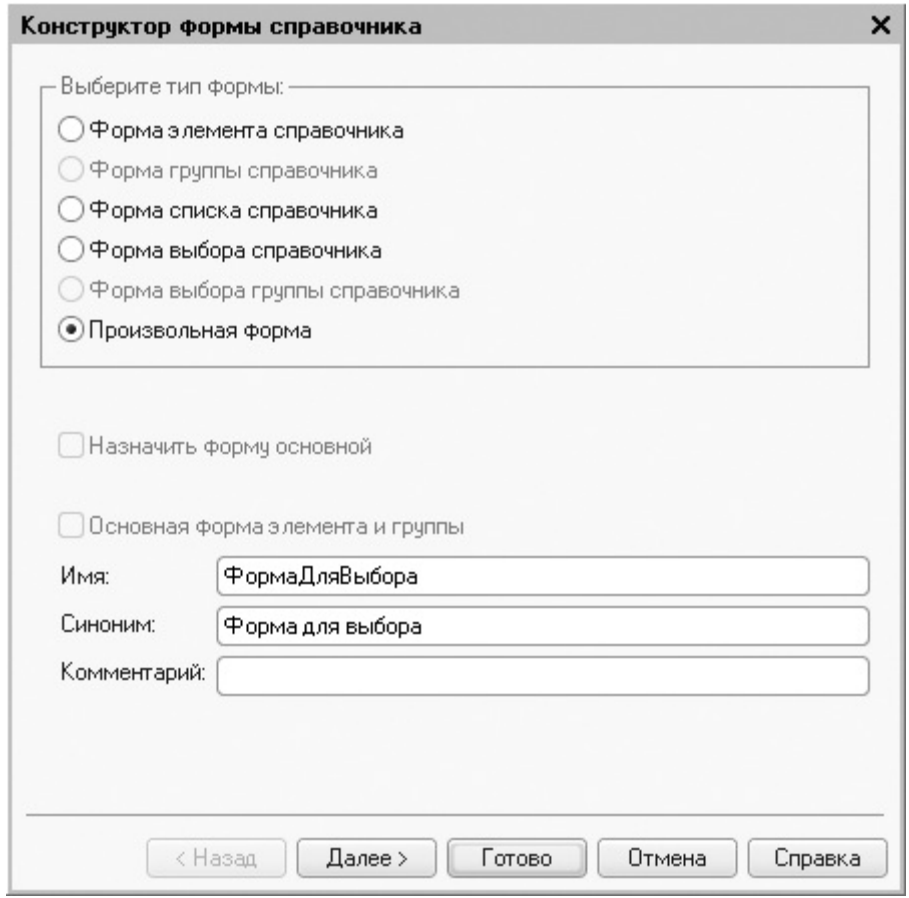


Рис. 27.24. Создание произвольной формы справочника «Сотрудники»

Нажмем *Готово*.

В открывшемся редакторе форм мы видим, что в форме нет ни одного элемента, и у формы нет данных, так как форма – произвольная, и мы можем наполнять ее данными и элементами по своему усмотрению.

На закладке *Реквизиты* создадим основной реквизит формы с именем *Список* и типом *ДинамическийСписок*. В качестве основной таблицы выберем *Справочник.Сотрудники* (рис. 27.25).

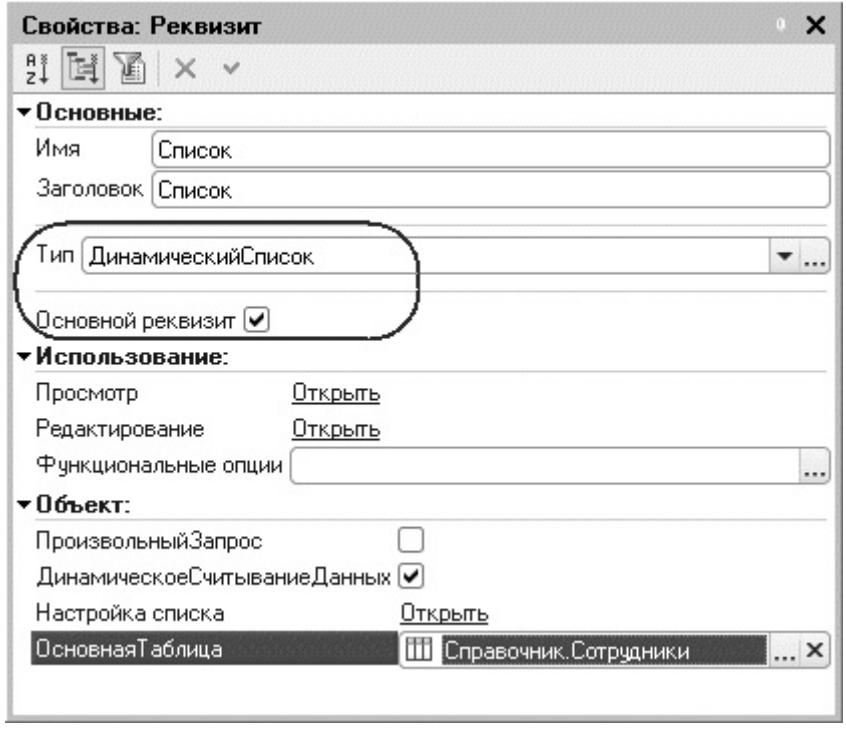


Рис. 27.25. Создание основного реквизита формы

Затем перетащим этот реквизит в окно элементов формы. Согласимся с предложением «Добавить колонки таблицы *Список*?». В форме появится таблица *Список*, отображающая список сотрудников.

Теперь немного изменим внешний вид формы.

В окне элементов формы раскроем таблицу *Список* и удалим поле *Код*. Затем удалим командную панель формы, так как таблица по умолчанию имеет свою командную панель. Для этого выделим корневой элемент *Форма*, откроем его палитру свойств и установим свойство *ПоложениеКомандной Панели* в значение *Нет* (рис. 27.26).

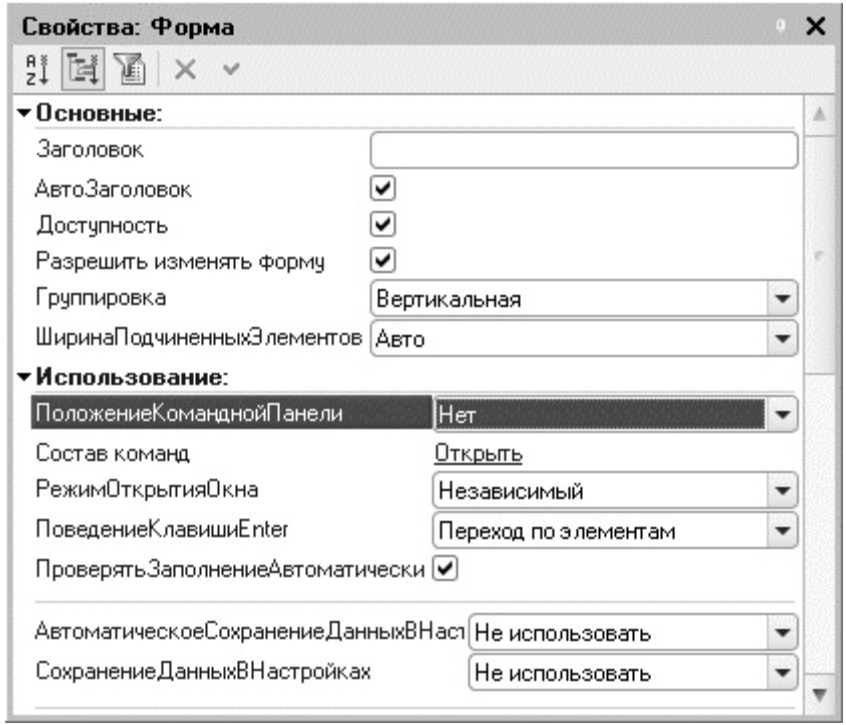


Рис. 27.26. Установка свойств формы

Чтобы из таблицы, содержащей список сотрудников, можно было сделать выбор, откроем палитру свойств таблицы *Список* и установим флажок *РежимВыбора* (рис. 27.27).

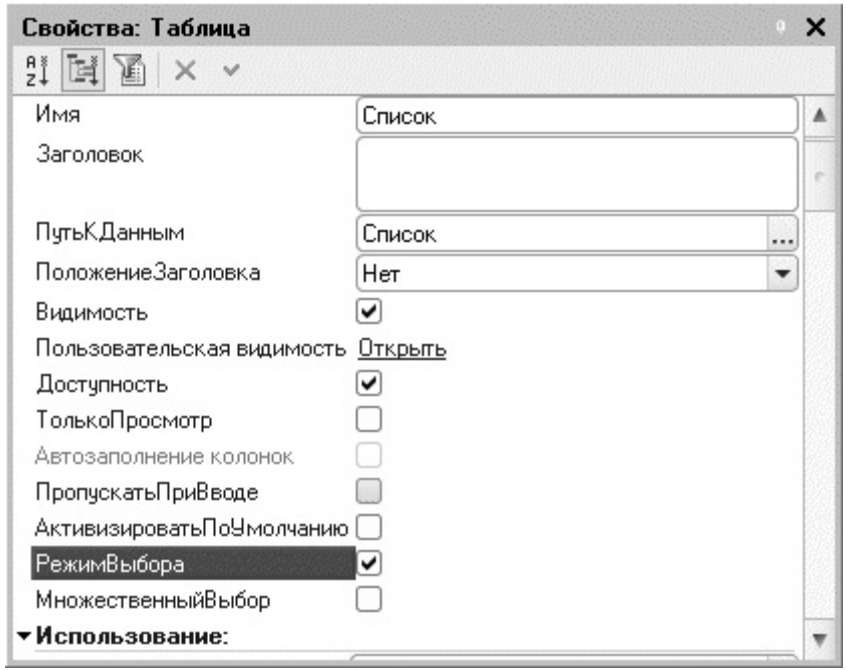


Рис. 27.27. Свойства таблицы

И в заключение зададим заголовок формы *Выбор сотрудников*, а флажок *АвтоЗаголовок* снимем, чтобы не отражался заголовок, заданный нами в расширенном представлении списка для справочника *Сотрудники* (рис. 27.28).

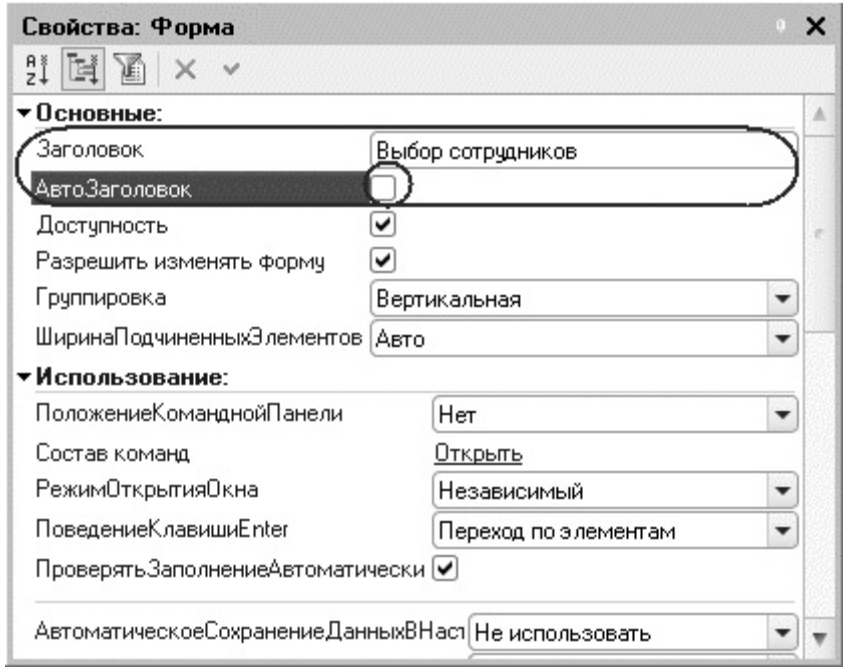


Рис. 27.28. Установка свойств формы

В результате мы создали форму, очень похожую на форму выбора. Мы убрали из списка поле *Код* и изменили заголовок формы, чтобы наша форма внешне в чем-то отличалась от формы выбора.

В действительности произвольная форма, конечно, будет отличаться своей функциональностью, иначе зачем ее назначать в качестве формы для выбора? Но это сейчас не входит в нашу задачу. Мы хотим только показать принцип

использования произвольной формы в качестве формы выбора.

Теперь откроем окно редактирования объекта конфигурации *Документ ОказаниеУслуги*.

Вызовем палитру свойств реквизита *Мастер* и нажмем кнопку выбора в поле *Форма выбора*. Откроется список форм, созданных в конфигурации для объекта, на который ссылается данный реквизит. В данном случае для справочника *Сотрудники*, на который ссылается реквизит *Мастер*, создана одна форма – *ФормаДляВыбора*. Выберем ее (рис. 27.29).

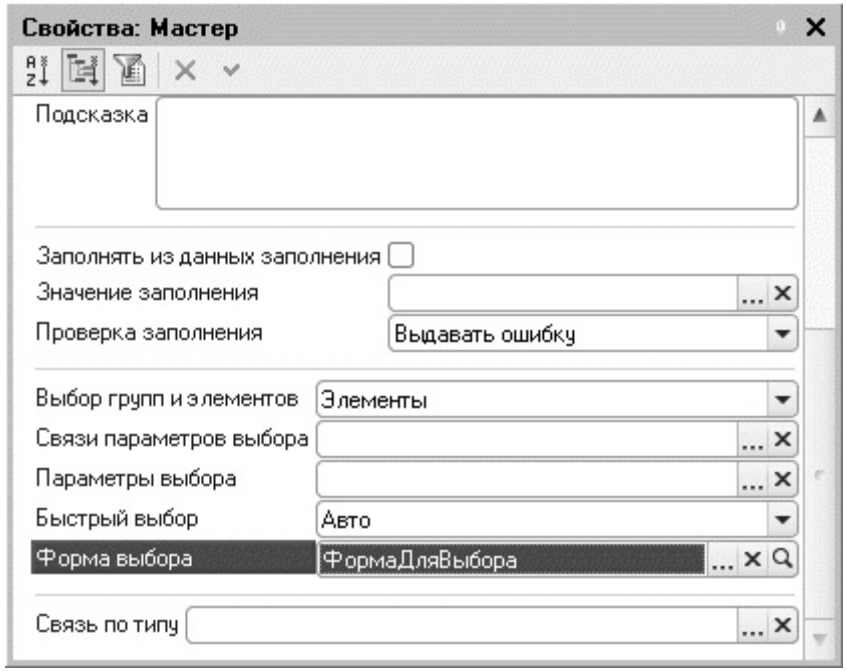


Рис. 27.29. Установка формы выбора для ссылочного реквизита

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем один из документов *Оказание услуги*. Нажмем кнопку выбора в поле *Мастер*.

Откроется созданная нами произвольная форма с заголовком *Выбор сотрудников* (рис. 27.30).

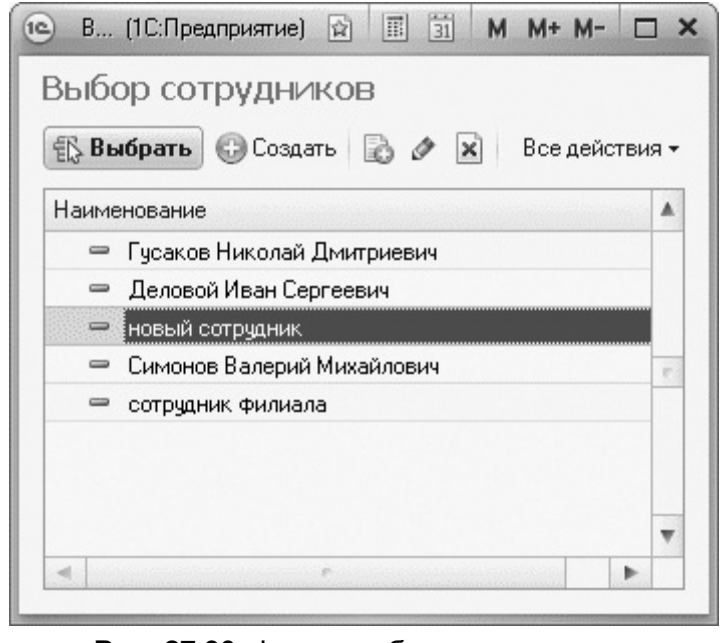


Рис. 27.30. Форма выбора сотрудников

Таким образом, поскольку мы задали свойство *Форма выбора* на уровне реквизита *Мастер*, а не на уровне отдельной формы, в любой форме документа *ОказаниеУслуги*, где используется данный реквизит, при выборе в поле *Мастер* будет открываться наша специальная форма.

Если очистить свойство *Форма выбора* для реквизита *Мастер* в документе *ОказаниеУслуги*, то в режиме *1С:Предприятие* при выборе в поле *Мастер* будет открываться основная форма выбора справочника *Сотрудники,*

автоматически сгенерированная системой.

Проверка заполнения реквизитов

Для реквизитов объектов конфигурации существует возможность как автоматической, так и программной проверки их заполнения. Причем делается это не на уровне форм, а на уровне свойств реквизитов, или в модуле объекта, к которому относится данный реквизит. Таким образом, проверка заполнения реквизита будет производиться во всех формах, где используется этот реквизит.

Автоматическая проверка заполнения

В режиме «Конфигуратор»

Для примера рассмотрим объект конфигурации *Документ ОказаниеУслуги*. Допустим, нам нужно контролировать заполнение реквизита *НаборСвойств* табличной части этого документа.

Откроем окно редактирования объекта конфигурации Документ *ОказаниеУслуги*. Вызовем палитру свойств реквизита табличной части *НаборСвойств* и установим свойство *Проверка заполнения* в значение *Выдавать ошибку*.

Тем самым при записи документа этот реквизит будет проверяться на заполнение. Иначе будет выдано сообщение об ошибке, и документ не будет сохранен (рис. 27.31).

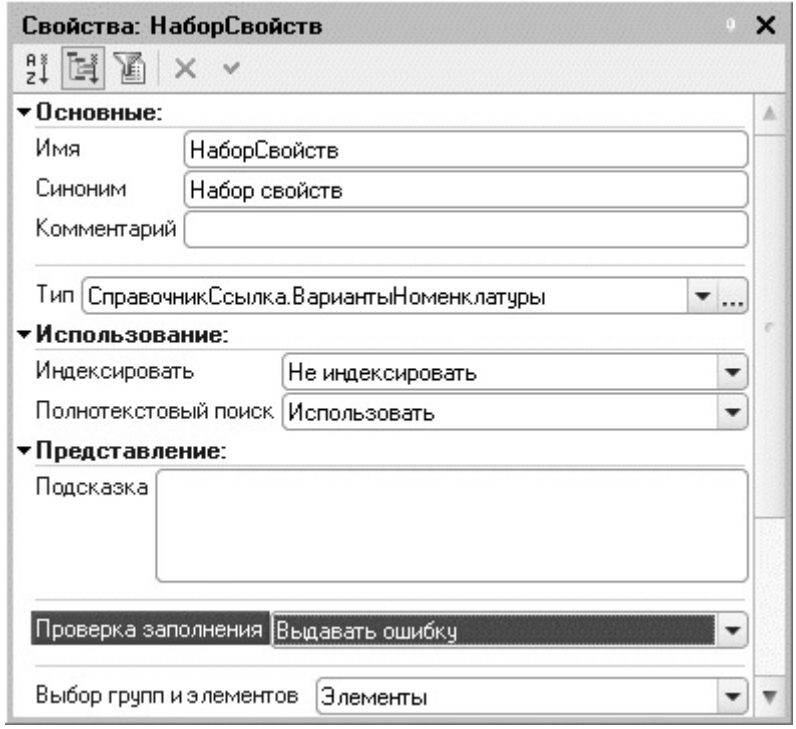


Рис. 27.31. Свойство «Проверка» заполнения реквизита «НаборСвойств»

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем документ *Оказание*

услуги № 4. В табличной части этого документа содержится одна строка с услугой *Диагностика*, для которой колонка *Набор свойств* не заполнена. При проведении этого документа будет выдано сообщение об ошибке, и изменения документа не будут сохранены (рис. 27.32).

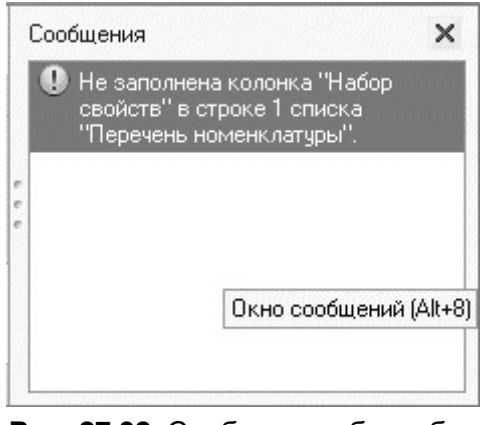


Рис. 27.32. Сообщение об ошибке

Программная проверка заполнения

В режиме «Конфигуратор»

Иногда бывает нужно самостоятельно производить проверку заполнения реквизита, в соответствии с программной логикой. В этом случае, если для реквизита установлено свойство *Проверка заполнения* в значение *Выдавать ошибку*, нужно удалить этот реквизит из массива проверяемых реквизитов и выполнить проверку программным путем. Или же, если для реквизита не

установлена проверка заполнения, можно программно добавить реквизит в массив проверяемых реквизитов.

Программная проверка заполнения объектов конфигурации выполняется в обработчике *ОбработкаПроверкиЗаполнения()*, который нужно поместить в модуле объекта. Этот обработчик вызывается автоматически при сохранении любой формы. Программную проверку объектов интерактивного ввода нужно делать именно в этом обработчике, а не при записи объекта.

Откроем модуль документа *ОказаниеУслуги* и поместим в нем следующую процедуру (листинг 27.4).

Листинг 27.4. Обработчик события «ОбработкаПроверкиЗаполнения»

```
Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)

  ПроверяемыеРеквизиты.Удалить(ПроверяемыеРеквизиты.Найти("ПереченьНоменклатуры.Н
  Индекс = 0;

  Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл
    Индекс = Индекс + 1;

    Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
    Перечисления.ВидыНоменклатуры.Материал Тогда

      Если Не ЗначениеЗаполнено(ТекСтрокаПереченьНоменклатуры.НаборСвойств)
      Тогда
```

```
        Сообщение = Новый СообщениеПользователю();
        Сообщение.Текст = "В строке " + Индекс + " списка Перечень номенклатуры
        не заполнена колонка Набор свойств";
        Сообщение.Поле = "ПереченьНоменклатуры[" + Строка(Индекс - 1) +
        "].НаборСвойств";
        Сообщение.УстановитьДанные(ЭтотОбъект);
        Сообщение.Сообщить();

        Отказ = Истина;

      КонецЕсли;

    КонецЕсли;

  КонецЦикла;

КонецПроцедуры
```

Сначала мы удаляем реквизит табличной части *НаборСвойств* из массива *ПроверяемыеРеквизиты*. Этот массив передается в обработчик и содержит массив проверяемых реквизитов, которым мы установили свойство *Проверка заполнения* в значение *Выдавать ошибку* во время разработки конфигурации.

Затем мы в цикле обходим строки табличной части документа и формируем сообщения об ошибке только в том случае, если номенклатура в табличной части является материалом и для нее не заполнена колонка *НаборСвойств*.

Параметр *Отказ* мы устанавливаем в значение *Истина*. Это значит, что документ не будет сохранен, если найден хоть один незаполненный реквизит *НаборСвойств* для номенклатуры, являющейся материалом. Если этот параметр закомментировать, сообщения об ошибке будут выдаваться, но документ будет сохранен.

Для упрощения примера мы опять здесь используем обращение *ПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры*, хотя оптимальнее использовать запрос. Об этом подробно рассказывалось на занятии №14, поэтому мы не будем здесь рассматривать этот вопрос.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем документ *Оказание услуги* № 4. В табличную часть этого документа добавим еще одну строку, содержащую какой-либо материал, и попробуем записать документ, нажав кнопку со значком дискеты. Для второй строки табличной части будет выдано сообщение об ошибке (рис. 27.33).

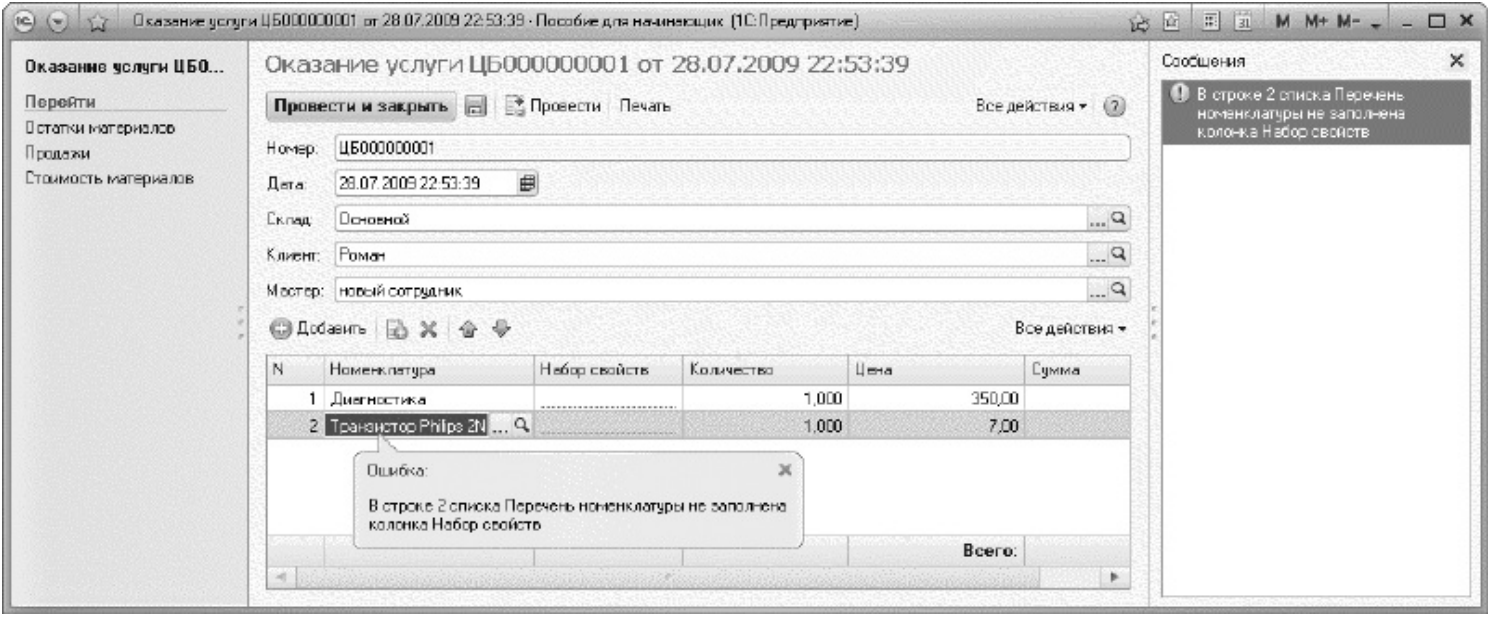


Рис. 27.33. Сообщение об ошибке при записи документа

Можно также программно добавить реквизит в массив проверяемых реквизитов (листинг 27.5).

Листинг 27.5. Добавление реквизита в массив проверяемых реквизитов

```
ПроверяемыеРеквизиты.Добавить("ПеречеьНоменклатуры.НаборСвойств");
```

Использование параметризированных команд

Использование параметризированных команд в формах объектов позволяет при выполнении команды передать в обработчик команды какой-либо параметр, например значение ссылочного реквизита. И затем использовать его, например, открыть с этим параметром форму отчета.

Для примера создадим команду открытия отчета, показывающего остатки материалов по складу, указанному в документе. Отчет будет вызываться из формы документа *ОказаниеУслуги*, в него будет передаваться значение реквизита документа *Склад*, и при открытии отчета в настройки отчета будет добавляться отбор по параметру *Склад*.

В режиме «Конфигуратор»

Сначала создадим параметризированную команду объекта конфигурации *Отчет Материалы*.

Для этого откроем окно редактирования этого объекта. На закладке *Команды* нажмем кнопку *Добавить* и создадим команду *ОстаткиПоСкладу*.

В открывшейся палитре свойств зададим *Тип параметра команды – СправочникСсылка.Склады*. В свойстве *Группа* укажем *Командная панель формы.Важное* (рис. 27.34).

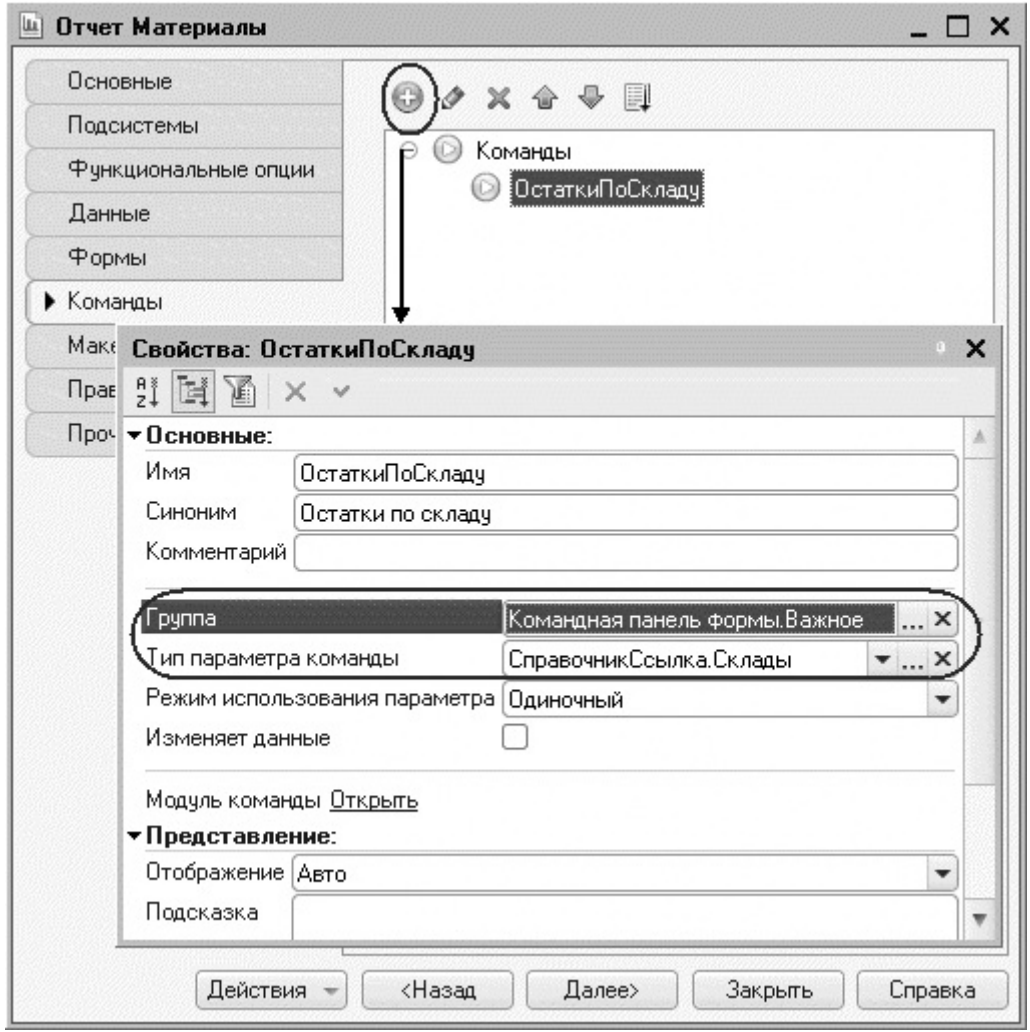


Рис. 27.34. Создание параметризированной команды

Таким образом, мы создали команду с типом параметра *СправочникСсылка.Склады*, и теперь во всех формах, имеющих реквизит такого типа, эта команда автоматически будет добавлена в список параметризированных команд, доступных в форме.

В открывшемся модуле команды заполним обработчик для ее выполнения следующим образом (листинг 27.6).

Листинг 27.6. Обработчик выполнения команды

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

  ПараметрыФормы = Новый Структура("Отбор, КлючНазначенияИспользования,
  СформироватьПриОткрытии", Новый Структура("Склад", ПараметрКоманды),
  "ОстаткиПоСкладу", Истина);
  ОткрытьФорму("Отчет.Материалы.Форма", ПараметрыФормы,
  ПараметрыВыполненияКоманды.Источник, ПараметрыВыполненияКоманды.Уникальность,
  ПараметрыВыполненияКоманды.Окно);

КонецПроцедуры
```

Прокомментируем этот код.

В процедуру *ОбработкаКоманды()* передается *ПараметрКоманды*,

содержащий значение типа *СправочникСсылка.Склады*. Затем создается структура параметров формы (*ПараметрыФормы*): *Отбор, КлючНазначенияИспользования, СформироватьПриОткрытии*.

В параметр *Отбор* добавляется структура, содержащая элемент *Склад* со значением, содержащимся в параметре команды (*ПараметрКоманды*), параметр *КлючНазначенияИспользования* – *«ОстаткиПоСкладу»* определяет назначение использования формы, а параметру *СформироватьПриОткрытии* присваивается значение *Истина*, чтобы отчет формировался сразу после открытия.

Затем структура параметров формы передается в глобальный метод *ОткрытьФорму()*, и форма, указанная в первом параметре метода, открывается с параметром *Склад*.

Теперь разместим нашу команду в форме документа.

Как мы уже говорили, в формах документов *ПриходнаяНакладная* и *ОказаниеУслуги* содержится команда *ОстаткиПоСкладу*, так как они имеют реквизит *Склад* типа *СправочникСсылка.Склады*.

Откроем форму документа *ОказаниеУслуги*.

На закладке *Команды* перейдем в раздел *Глобальные команды*. Здесь мы видим список глобальных параметризированных команд, доступных в форме. В группе *Параметризированные* раскроем строку *Объект* и перетащим нашу команду *Отчет.Материалы.Команда.ОстаткиПоСкладу(Объект.Склад)* в командную панель элементов формы.

В скобках у команды указано значение реквизита *Склад (Объект.Склад)*, которое будет передаваться в команду при ее выполнении (рис. 27.35).

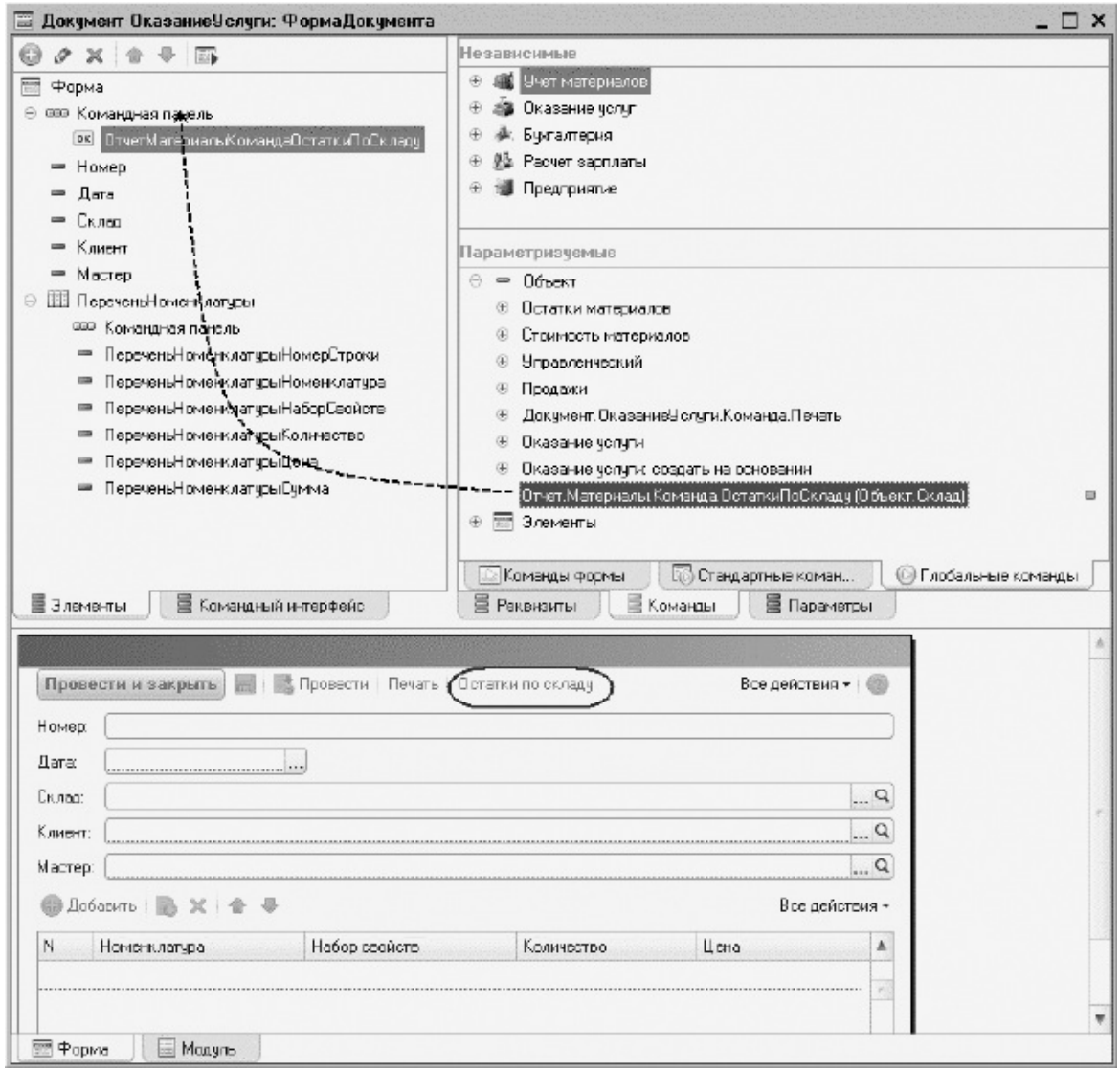

Документ ОказаниеУслуги: ФормаДокумента
Форма
Командная панель
ОтчетМатериалыКомандаОстаткиПоСкладу
Номер
Дата
Склад
Клиент
Мастер
ПеречеьНоменклатуры
Командная панель
ПеречеьНоменклатурыНомерСтроки
ПеречеьНоменклатурыНоменклатура
ПеречеьНоменклатурыНаборСвойств
ПеречеьНоменклатурыКолличество
ПеречеьНоменклатурыЦена
ПеречеьНоменклатурыСумма
Независимые
Учет материалов
Оказание услуг
Бухгалтерия
Расчет зарплаты
Предприятие
Параметризуемые
Объект
Остатки материалов
Стоимость материалов
Управленческий
Продажи
Документ.ОказаниеУслуги.Команда.Печать
Оказание услуги
Оказание услуги: создать на основании
Отчет.Материалы.Команда.ОстаткиПоСкладу (Объект.Склад)
Элементы
Команды формы
Стандартные коман...
Глобальные команды
Элементы
Командный интерфейс
Реквизиты
Команды
Параметры
Провести и закрыть
Провести
Печать
Остатки по складу
Все действия
Номер:
Дата:
Склад:
Клиент:
Мастер:
Добавить
Все действия
N
Номенклатура
Набор свойств
Количество
Цена
Форма
Модуль

Рис. 27.35. Размещение параметризированной команды в форме

Заметьте, что мы не создавали форму отчета и не добавляли в настройки отчета отбор по параметру формы *Склад*. Система сделает это сама при выполнении обработчика команды *ОстаткиПоСкладу*.

В режиме «1С:Предприятие»

Запустим «1С:Предприятие» в режиме отладки и откроем документ *Оказание услуги* № 4.

Нажмем кнопку *Остатки по складу* и вызовем отчет *Материалы*. Форма отчета генерируется системой автоматически. Отчет будет выполнен сразу при открытии формы с отбором по складу *Основной*, указанному в форме документа (рис. 27.36).

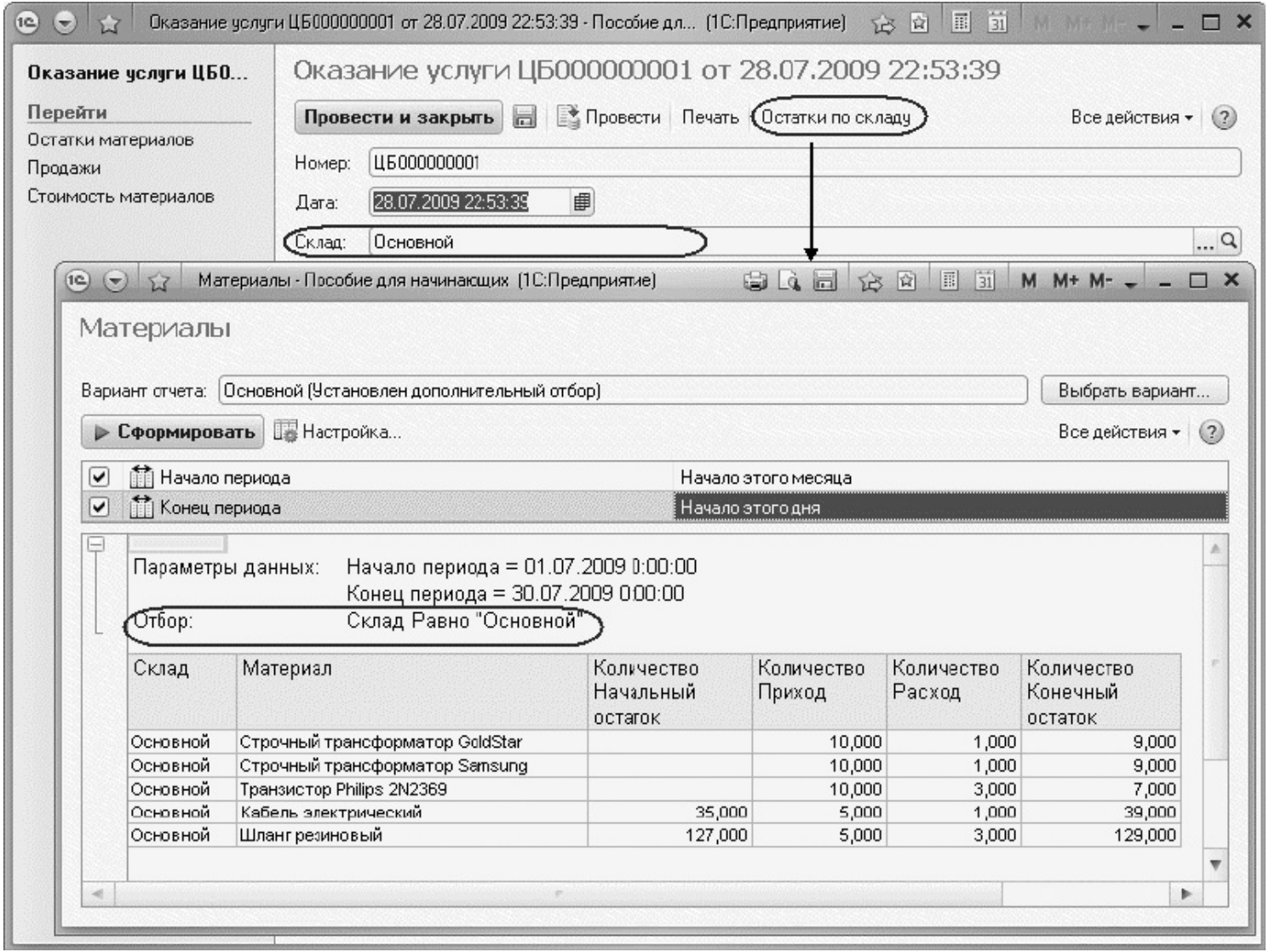

Оказание услуги ЦБ000000001 от 28.07.2009 22:53:39 - Пособие дл... (1С:Предприятие)
Оказание услуги ЦБ...
Перейти
Остатки материалов
Продажи
Стоимость материалов
Оказание услуги ЦБ000000001 от 28.07.2009 22:53:39
Провести и закрыть
Провести
Печать
Остатки по складу
Все действия
Номер: ЦБ000000001
Дата: 28.07.2009 22:53:39
Склад: Основной
Материалы - Пособие для начинающих (1С:Предприятие)
Материалы
Вариант отчета: Основной (Установлен дополнительный отбор)
Выбрать вариант...
Сформировать
Настройка...
Все действия
Начало периода
Начало этого месяца
Конец периода
Начало этого дня
Параметры данных: Начало периода = 01.07.2009 0:00:00
Конец периода = 30.07.2009 0:00:00
Отбор: Склад Равно "Основной"
Склад
Материал
Количество Начальный остаток
Количество Приход
Количество Расход
Количество Конечный остаток
Основной Строчный трансформатор GoldStar 10,000 1,000 9,000
Основной Строчный трансформатор Samsung 10,000 1,000 9,000
Основной Транзистор Philips 2N2369 10,000 3,000 7,000
Основной Кабель электрический 35,000 5,000 1,000 39,000
Основной Шланг резиновый 127,000 5,000 3,000 129,000

Рис. 27.36. Отчет «Материалы» с отбором по складу, указанном в документе

Закроем отчет. Если теперь в форме документа мы изменим склад на *Розничный* и вызовем отчет нажатием кнопки *Остатки по складу* и выполним его, то он будет сформирован с отбором складу *Розничный*.

Таким образом, мы создали для пользователя очень удобную возможность – открывать отчет, показывающий остатки материалов по складу, прямо из формы документа с отбором по указанному в документе складу.

Контрольные вопросы

- *Как связаны данные и элементы формы.*
- *Что такое основной реквизит формы.*
- *Что такое расширения формы и ее элементов.*
- *Какие существуют типы данных у формы.*
- *Как выполнить преобразование данных прикладных объектов в данные формы.*
- *Что такое связанная информация и как к ней перейти из формы.*
- *Как настроить условное оформление строк формы списка.*
- *Как установить форму выбора для ссылочного реквизита.*
- *Как установить автоматическую и программную проверку заполнения*

реквизитов.

- *Что такое параметризированная команда.*
- *Как использовать параметризированные команды в формах.*

Краткий справочник разработчика

Объекты встроенного языка для работы с прикладными данными

Для обеспечения доступа к данным, хранящимся в базе данных, встроенный язык содержит набор унифицированных объектов. Их можно разделить на несколько видов, в зависимости от их назначения.

Менеджер информационных структур одного вида. Это такие объекты, как:

- *СправочникиМенеджер*,
- *ДокументыМенеджер*,
- *ОтчетыМенеджер*,
- *ПланыСчетовМенеджер*,
- и т. д.

Каждый из них является коллекцией значений, содержащей менеджеры всех информационных структур этого вида, существующих в базе данных.

Например, менеджер справочников, – *СправочникиМенеджер,* – это коллекция

значений, содержащая объекты *СправочникМенеджер.<имя>*.

Каждый из них предназначен для доступа к отдельным менеджерам информационных структур.

Менеджер конкретной информационной структуры. Это такие объекты, как:

- *СправочникМенеджер.Клиенты*,
- *СправочникМенеджер.Номенклатура*,
- *ДокументМенеджер.ПриходнаяНакладная*,
- *ДокументМенеджер.ОказаниеУслуги*,
- и т. д.

Каждый из этих объектов предоставляет средства для работы с конкретной информационной структурой. Например менеджер документа *ПриходнаяНакладная – ДокументМенеджер.ПриходнаяНакладная –* позволяет находить конкретные документы *Приходная накладная*, создавать объекты этих документов и т. д.

Объект – это такие объекты, как:

- *СправочникОбъект.Клиенты*,

- *СправочникОбъект.Номенклатура*,
- *ДокументОбъект.ПриходнаяНакладная*,
- *ДокументОбъект.ОказаниеУслуги*,
- и т. д.

С помощью объектов этого вида возможно чтение, изменение, запись и удаление данных информационной структуры. Они предоставляют доступ к объекту информационной структуры и позволяют изменять информацию в базе данных. Применяются для тех информационных структур, на объекты которых могут существовать ссылки (справочники – *СправочникОбъект.<имя>*, документы – *ДокументОбъект.<имя>* и т. д.).

Набор записей – это такие объекты, как:

- *РегистрСведенийНаборЗаписей.Цены*,
- *РегистрНакопленияНаборЗаписей.ОстаткиМатериалов*,
- *РегистрБухгалтерииНаборЗаписей.Управленческий*,
- и т. д.

С помощью объектов этого вида также возможно чтение, изменение, запись и удаление данных информационной структуры. Предоставляют доступ к объекту информационной структуры и позволяют изменять информацию в базе данных.

Применяются для тех информационных структур, ссылки на объекты которых в принципе не могут использоваться в базе данных (регистры – *РегистрНакопленияНаборЗаписей.<имя>*, перерасчеты – *ПерерасчетНаборЗаписей.<имя>* и т. д.).

Ссылка – это такие объекты, как:

- *СправочникСсылка.Клиенты*,
- *СправочникСсылка.Номенклатура*,
- *ДокументСсылка.ПриходнаяНакладная*,
- *ДокументСсылка.ОказаниеУслуги*,
- и т. д.

Объекты этого вида служат для указания ссылки на объект базы данных и кроме этого предоставляют некоторую информацию об этом объекте (например, документ – *ДокументСсылка.<имя>*).

Выборка – это такие объекты, как:

- *СправочникВыборка.Клиенты*,
- *ДокументВыборка.ПриходнаяНакладная*,
- *РегистрСведенийВыборка.Цены*,

- *РегистрНакопленияВыборка.ОстаткиМатериалов*,
- и т. д.

Объекты этого вида представляют собой набор данных, содержащий данные объектов одной информационной структуры, отобранных по определенному критерию. Обход выборки выполняется методом *Следующий()*, и считывание данных из базы данных происходит динамически, по мере продвижения по выборке. Получение ссылки на объект возможно при помощи свойства *Ссылка*, а получение объекта – методом *ПолучитьОбъект()* (справочник – *СправочникВыборка.<имя>*).

Манипулирование данными объектов

Несмотря на большое разнообразие объектов встроенного языка, предназначенных для работы с данными, хранящимися в базе данных, лишь некоторые из них позволяют изменять данные, хранящиеся в базе данных. Такие объекты мы назовем *объектами манипулирования данными*.

Каждый тип объектов манипулирования данными имеет в конфигурации соответствующий модуль. Он называется либо *модулем объекта*, либо *модулем набора записей*, в зависимости от принадлежности к тому или иному объекту конфигурации. Для констант этот модуль называется *модулем менеджера значений*.

Так вот, модуль объекта манипулирования данными будет всегда выполняться при создании объекта манипулирования данными. Кроме этого, он будет всегда выполняться и при интерактивном обращении пользователя к самой структуре данных, поскольку оно будет вызывать создание соответствующего объекта манипулирования данными. Например, при открытии формы элемента справочника будет создаваться объект *СправочникОбъект.<имя>*.

В модуле объекта манипулирования данными кроме всего прочего могут быть описаны процедуры с ключевым словом *Экспорт*, что подразумевает вызов этих процедур как методов соответствующего объекта манипулирования данными. Здесь важно не путать объект манипулирования данными с другими объектами, позволяющими получить доступ к данным этой информационной структуры.

Например, если мы для объекта конфигурации *Справочник Клиенты* опишем в модуле объекта процедуру (листинг 28.1), то в дальнейшем сможем вызывать ее как метод объекта *СправочникОбъект.Клиенты* (листинг 28.2).

Листинг 28.1. Процедура «Проверка()» в модуле справочника

```
Процедура Проверка() Экспорт
…
КонецПроцедуры;
```

Листинг 28.2. Вызов процедуры как метода объекта «Справочник»

```
Клиент = Справочники.Клиенты.НайтиПоКоду(1).ПолучитьОбъект();
Клиент.Проверка();
```

Однако следующий код будет приводить к ошибке, так как объект *СправочникСсылка.Клиенты* не имеет метода *Проверка* (листинг 28.3).

Листинг 28.3. Вызов процедуры «Проверка» приведет ошибке

```
Клиент = Справочники.Клиенты.НайтиПоКоду(1);
Клиент.Проверка();
```

В следующей таблице представлен перечень объектов, позволяющих манипулировать данными. Как всегда, не бывает правил без исключений, и существует два таких исключения.

Таблица 4.1. Работа с данными объектов

Объект конфигурации	База данных – структура манипулирования данными	Встроенный язык – структу манипулирования данны
		КонстантаМенеджерЗначения.<

Константа	Константа	(КонстантаМенеджер.<Имя>, КонстантыНабор)
Справочник	Элемент справочника	СправочникОбъект.<Имя>
Документ	Документ	ДокументОбъект.<Имя>
Последовательность	Набор записей последовательности	ПоследовательностьНаборЗапи <Имя>
ПланВидовХарактеристик	Вид характеристики	ПланВидовХарактеристикОбъе <Имя>
ПланСчетов	Счет	ПланСчетовОбъект.<Имя>
ПланВидовРасчета	Вид расчета	ПланВидовРасчетаОбъект.<Им
РегистрСведений	Набор записей регистра сведений	РегистрСведенийНаборЗаписей <Имя> (РегистрСведенийМенеджерЗа <Имя>)
РегистрНакопления	Набор записей регистра накопления	РегистрНакопленияНаборЗапис <Имя>
	Набор записей	РегистрБухгалтерииНаборЗапис

РегистрБухгалтерии	регистра бухгалтерии	<Имя>
РегистрРасчета	Набор записей регистра расчета	РегистрРасчетаНаборЗаписей.<Имя>

Во-первых, для констант указаны три объекта манипулирования данными – *КонстантаМенеджерЗначения.<имя>*, *КонстантаМенеджер.<имя>* и *КонстантыНабор*. На самом деле манипулирование данными константы осуществляется при помощи объекта *КонстантаМенеджерЗначения.<имя>*.

Два других объекта – *КонстантаМенеджер.<имя>* и *КонстантыНабор* – также позволяют изменять значения констант, хранящиеся в базе данных, однако при выполнении своих методов *Установить()* и *Записать()* они вызывают создание объекта *КонстантаМенеджерЗначения.<имя>*, который и выполняет непосредственное изменение данных.

При выполнении метода *Установить()* объекта *КонстантаМенеджер.<имя>* будет вызван модуль менеджера значения и обработчики событий *ПриЗаписи()* и *ПередЗаписью()* для изменяемой константы. При выполнении метода *Записать()* объекта *КонстантыНабор* модуль менеджера значения и соответствующие обработчики будут вызваны для каждой константы, входящей в набор.

Во-вторых, для регистра сведений указаны два объекта манипулирования данными. В чистом виде манипулирование данными регистра сведений осуществляется при помощи объекта *РегистрСведенийНаборЗаписей.<имя>*.

Однако существует возможность манипулирования записями регистра сведений и при помощи объекта *РегистрСведенийМенеджерЗаписи.<имя>*. Но объект *РегистрСведенийМенеджерЗаписи.<имя>* работает с данными регистра не напрямую, а через объект *РегистрСведенийНаборЗаписей.<имя>*. Таким образом, модуль набора записей, а также обработчики событий *ПередЗаписью()* и *ПриЗаписи()* набора записей будут отрабатывать и при манипулировании объектом *РегистрСведенийМенеджерЗаписи.<имя>*. Однако процедуры и функции, описанные в модуле набора записей с ключевым словом *Экспорт*, не будут доступны как методы объекта *РегистрСведенийМенеджерЗаписи.<имя>*.

Константы

Объекты встроенного языка для работы с константами

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с константами (рис. 28.1).

Рис. 28.1. Объекты встроенного языка для работы с константами

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными. Метод объекта, от которого идет стрелка, приводится в листинге под соответствующей цифрой, а объект, к которому идет стрелка, – это тип возвращаемого метода.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

КонстантыНабор – предоставляет возможность проведения операций чтения и записи сразу для группы констант, в частном случае – для всех констант. Также используется в форме констант для хранения, записи и считывания констант.

КонстантаМенеджерЗначения.<имя> – используется для доступа к константе. Любая запись константы (интерактивно в форме, объекты *КонстантыНабор* и *КонстантаМенеджер.<имя>*) создает объект этого типа и производит запись с его помощью, что обеспечивает вызов модуля и обработчиков событий этого объекта.

Ниже приведены примеры использования объектов встроенного языка для работы с константами (листинг 28.4).

Листинг 28.4. Примеры использования объектов

```
1.      // Глобальный констекст
  // Константы

// Пример: установить значение константы.
Константы.Бухгалтер.Установить("Сидоров Петр Иванович");

2.      // объект КонстантыМенеджер
  // .
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;
```

```
// Пример: прочитать значение константы.
Результат = Константы.ПрефиксНумерации.Получить();
Сообщить("Значение константы ПрефиксНумерации = "+ Результат);

// Пример: установить значение константы ПрефиксНумерации равным ЦБ.
Константы["ПрефиксНумерации"].Установить("ЦБ");
Сообщить("Новое значение = " + Константы["ПрефиксНумерации"].Получить());

// Пример: очистить значения всех констант.
Для Каждого ОчереднаяКонстанта Из Константы Цикл
  ОчереднаяКонстанта.Установить(Неопределено);
КонецЦикла;

3.      // объект КонстантыМенеджер
  // СоздатьНабор()
// Пример: установить новые значения нескольких констант.
Набор = Константы.СоздатьНабор("Руководитель, Бухгалтер");
Набор.Руководитель = "Николаев Денис Павлович";
Набор.Бухгалтер = "Николаева Людмила Сергеевна";
Набор.Записать();

4.      // объект КонстантаМенеджер.
  // СоздатьМенеджерЗначения()
// Пример: вывести значения всех констант, существующих в конфигурации.
Для Каждого ОчереднаяКонстанта Из Константы Цикл
  ИмяКонст = ОчереднаяКонстанта.СоздатьМенеджерЗначения().Метаданные().Имя;
  ЗначениеКонст = ОчереднаяКонстанта.Получить();
  Сообщить("Константа "+ ИмяКонст +" = "+ ЗначениеКонст);
КонецЦикла;
```

Последовательность событий при записи констант из формы констант (записать и закрыть)

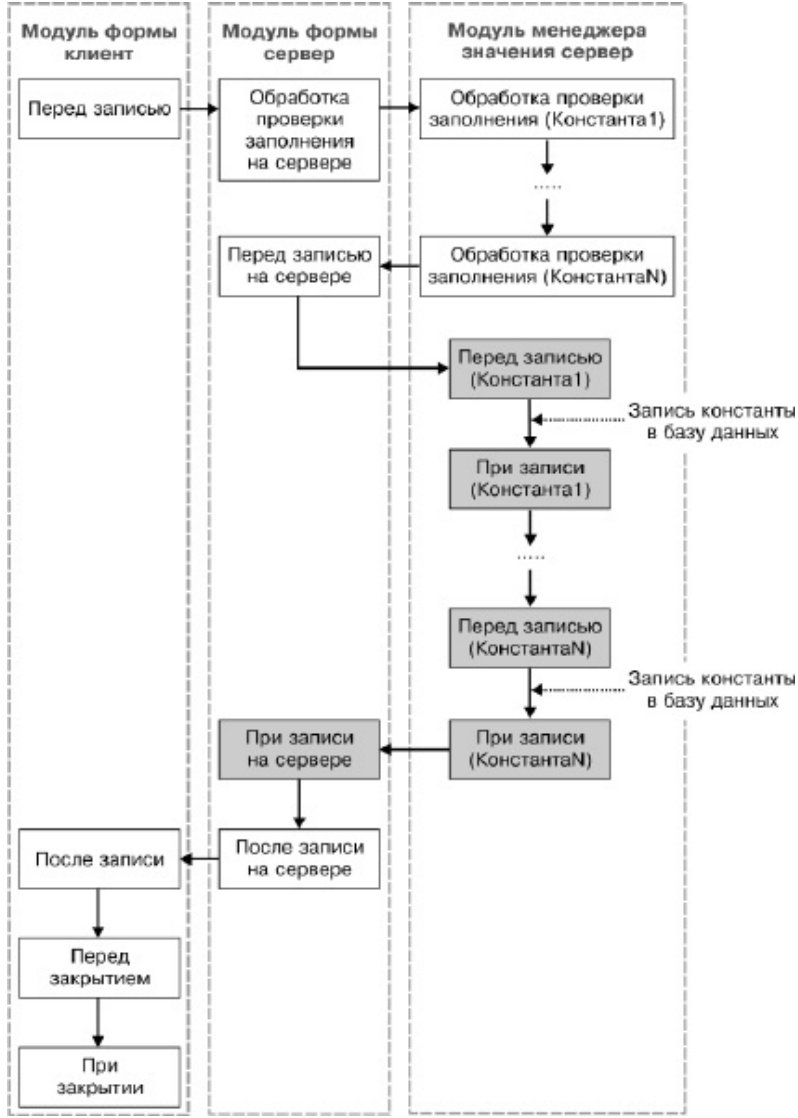

Модуль формы клиент
Модуль формы сервер
Модуль менеджера значения сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения (Константа1)
.....
Обработка проверки заполнения (КонстантаN)
Перед записью на сервере
Перед записью (Константа1)
Запись константы в базу данных
При записи (Константа1)
.....
Перед записью (КонстантаN)
Запись константы в базу данных
При записи (КонстантаN)
При записи на сервере
После записи на сервере
После записи
Перед закрытием
При закрытии

Рис. 28.2. Последовательность событий при записи констант из формы констант

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Работа с формой констант осуществляется при помощи объекта *КонстантыНабор*, который, в свою очередь, использует объект *КонстантаМенеджерЗначения.<имя>*.

Особенности внутренней реализации объекта *КонстантыНабор* таковы, что при записи набора констант обработчики события *ОбработкаПроверкиЗаполнения()*, *ПередЗаписью()* и *ПриЗаписи()* модуля менеджера значения константы будут вызваны для каждой константы, входящей в записываемый набор.

Справочники

Объекты встроенного языка для работы со справочниками

На следующей схеме изображено взаимодействие объектов встроенного языка для работы со справочниками (рис. 28.3).

Рис. 28.3. Объекты встроенного языка для работы со справочниками

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными. Метод объекта, от которого идет стрелка, приводится в листинге под соответствующей цифрой (например, под цифрой 3 приводится метод НайтиПоКоду() *объекта* СправочникМенеджер.<имя>*), а объект, к которому идет стрелка, – это тип возвращаемого метода (например,* СправочникСсылка.<имя>*).*

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

Ниже приведены примеры использования объектов встроенного языка для работы со справочниками (листинг 28.5).

Листинг 28.5. Примеры использования объектов

```
1.      // Глобальный контекст
  // Справочники

// Пример: вывести все типы ссылок на элементы справочников, существующие в
конфигурации.
Массив = Справочники.ТипВсеСсылки().Типы();
Для Каждого ОчереднойТип из Массив Цикл
  Сообщить(ОчереднойТип);
КонецЦикла;

2.      // объект СправочникиМенеджер
  // .
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;

// Пример: создать новую группу справочника "Номенклатура".
НоваяГруппа = Справочники.Номенклатура.СоздатьГруппу();
НоваяГруппа.Наименование = "Моя новая группа";
НоваяГруппа.Записать();

// Пример: получить ссылку на справочник "Номенклатура".
```

```
Справочники["Номенклатура"].ПолучитьСсылку();

3.      // объект СправочникМенеджер.
   // НайтиПоКоду()
   // НайтиПоНаименованию()
   // НайтиПоРеквизиту()
   // ПустаяСсылка()
   // ПолучитьСсылку()
   // .

// Пример: проверить, помечен ли на удаление элемент справочника "Номенклатура"
с кодом 13.
Если Справочники.Номенклатура.НайтиПоКоду(13).ПометкаУдаления Тогда
   Сообщить("Элемент с кодом 13 помечен на удаление");
КонецЕсли;

// Пример: является ли элемент справочника "Номенклатура" с наименованием
"Услуги" группой.
Если Справочники.Номенклатура.НайтиПоНаименованию("Услуги", Истина).ЭтоГруппа
Тогда
   Сообщить("Элемент Услуги является группой");
КонецЕсли;

// Пример: проверить, что для всех элементов задан вид номенклатуры.
ПустаяСсылкаПеречисления = Перечисления.ВидыНоменклатуры.ПустаяСсылка();
Если Не Справочники.Номенклатура.НайтиПоРеквизиту("ВидНоменклатуры",
ПустаяСсылкаПеречисления).Пустая() Тогда
   Сообщить("Есть элементы для которых не задан вид номенклатуры");
КонецЕсли;

// Пример: передать пустую ссылку в параметр метода.
```

```
Выборка =
Справочники.Номенклатура.Выбрать(Справочники.Номенклатура.ПустаяСсылка());

4.      // объект СправочникМенеджер.
  // Выбрать()
  // ВыбратьИерархически()

// Пример: вывести список элементов, расположенных в корне справочника.
Выборка =
Справочники.Номенклатура.Выбрать(Справочники.Номенклатура.ПустаяСсылка());
Пока Выборка.Следующий() Цикл
  Если Не Выборка.ЭтоГруппа тогда Сообщить(Выборка);
  КонецЕсли;
КонецЦикла;

// Пример: удалить все элементы иерархического справочника.
Выборка = Справочники.Номенклатура.ВыбратьИерархически();
Пока Выборка.Следующий() Цикл
  Выборка.Удалить();
КонецЦикла;

5.      // объект СправочникМенеджер.
  // СоздатьГруппу()
  // СоздатьЭлемент()

// Пример: создать новый элемент справочника "Сотрудники".
НовыйЭлемент = Справочники.Сотрудники.СоздатьЭлемент();
НовыйЭлемент.Наименование = "Смирнов Андрей Анатольевич";
// Заполнить табличную часть "ТрудоваяДеятельность".
НоваяСтрокаТабличнойЧасти = НовыйЭлемент.ТрудоваяДеятельность.Добавить();
```

```
НоваяСтрокаТабличнойЧасти.Организация = "ООО НТЦ";
НоваяСтрокаТабличнойЧасти.НачалоРаботы = Дата(2003,05,01);
НоваяСтрокаТабличнойЧасти.ОкончаниеРаботы = Дата(2003,12,31);
НоваяСтрокаТабличнойЧасти.Должность = "Программист";
НовыйЭлемент.Записать();

6.      // объект СправочникОбъект., СправочникСсылка.
  // Владелец
  // Родитель
  // Ссылка

// Пример: запретить изменение подчиненных элементов, если у
// владельца установлено соответствующее свойство
// "ИзмененияЗапрещены" в модуле формы элемента справочника.
Процедура ПередЗаписью(Отказ)
  Если Владелец.ИзмененияЗапрещены Тогда Отказ = Истина;
  КонецЕсли;
КонецПроцедуры

7.      // объект СправочникСсылка.
  // ПолучитьОбъект()
  //СправочникОбъект.
  // Скопировать()

// Пример: изменить наименование элемента справочника.
Элемент = Справочники.Номенклатура.НайтиПоКоду(10).ПолучитьОбъект();
Элемент.Наименование = "Мое новое наименование";
Элемент.Записать();

// Пример: заполнить справочник тестовыми данными.
Элемент = Справочники.Номенклатура.СоздатьЭлемент();
```

```
Элемент.Наименование = "Тестовый элемент";
Элемент.Записать();
Для ш = 1 по 1000 Цикл
  НовыйЭлемент = Элемент.Скопировать();
  НовыйЭлемент.Записать();
КонецЦикла;

8.      // объект СправочникВыборка.
  // Ссылка

// Пример: заполнить табличную часть документа
// "ПриходнаяНакладная" всеми элементами из указанной группы справочника
"Номенклатура".
Выборка = Справочники.Номенклатура.ВыбратьИерархически(ПолеВвода1);
Пока Выборка.Следующий() Цикл
  ССылкаНаНоменклатуру = Выборка.Ссылка;
  Если ССылкаНаНоменклатуру.ЭтоГруппа Тогда Продолжить;
  КонецЕсли;
  НоваяСтрока = Материалы.Добавить();
  НоваяСтрока.Материал = ССылкаНаНоменклатуру;
КонецЦикла;

9.      // объект СправочникВыборка.
  // ПолучитьОбъект()

// Пример: пометить все элементы неиерархического справочника на удаление.
Выборка = Справочники.Клиенты.Выбрать();
Пока Выборка.Следующий() Цикл
  Выборка.ПолучитьОбъект().УстановитьПометкуУдаления(Истина);
КонецЦикла;
```

Последовательность событий при записи элемента справочника из формы элемента (записать и закрыть)

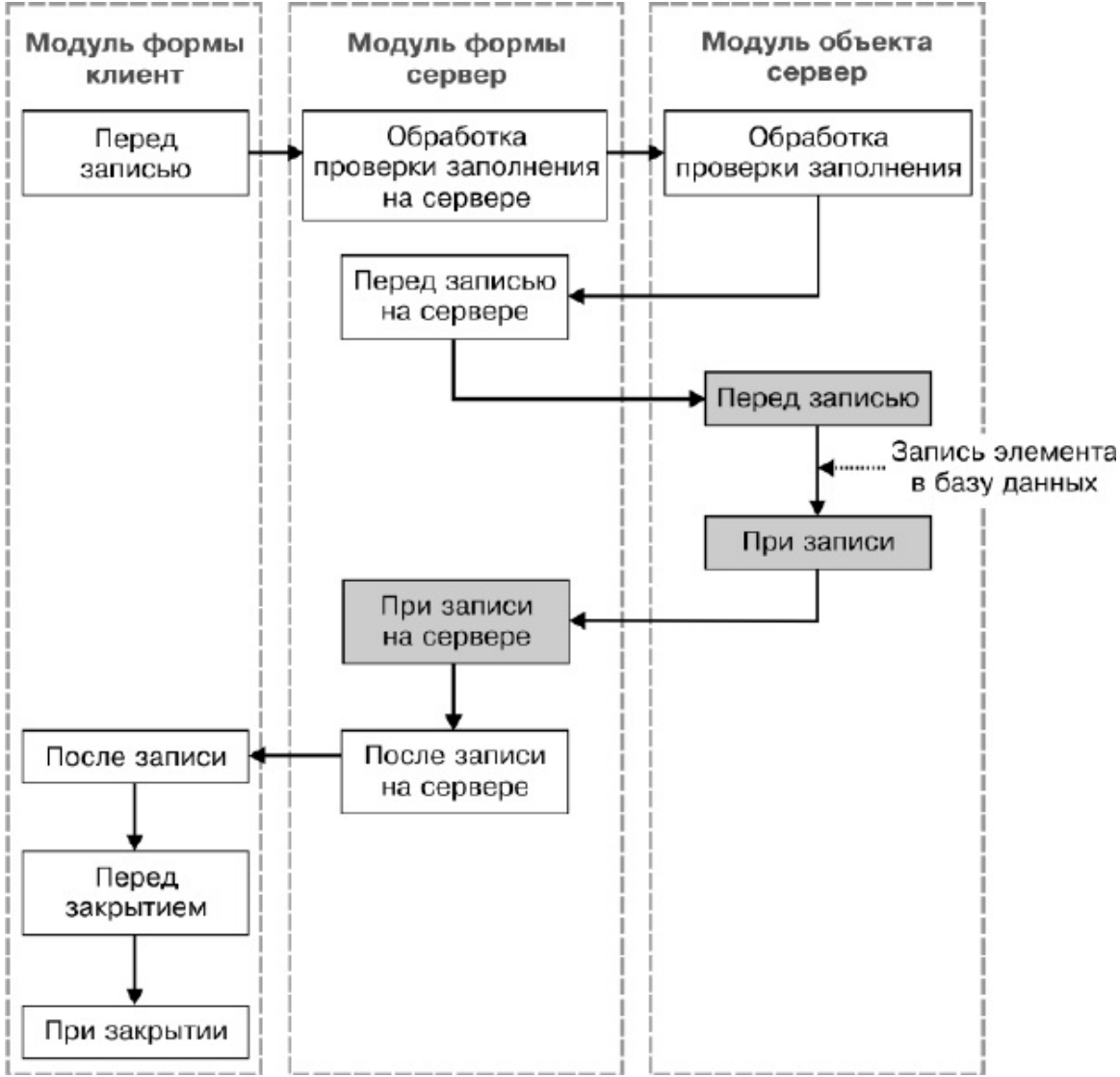

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись элемента в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.4. Последовательность событий при записи элемента справочника из формы элемента

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Документы

Объекты встроенного языка для работы с документами

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с документами (рис. 28.5).

Рис. 28.5. Объекты встроенного языка для работы с документами

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными. Метод объекта, от которого идет стрелка, приводится в листинге под соответствующей цифрой, а объект, к которому идет стрелка, – это тип возвращаемого метода.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

Ниже приведены примеры использования объектов встроенного языка для работы с документами (листинг 28.6).

Листинг 28.6. Примеры использования объектов

```
1.      // Глобальный констекст
  // Документы

// Пример: вывести все типы ссылок на элементы справочников, существующие в
конфигурации.
Массив = Документы.ТипВсеСсылки().Типы();
Для Каждого ОчереднойТип из Массив Цикл
  Сообщить(ОчереднойТип);
КонецЦикла;

2.      //объект ДокументыМенеджер
  // .
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;

// Пример: получить макет для печати документа "Оказание услуги".
Макет = Документы["ОказаниеУслуги"].ПолучитьМакет("Печать");

// Пример: получить ссылку на каждый из документов, существующих в
конфигурации.
Для Каждого ОчереднойДокумент Из Документы Цикл
  Ссылка = ОчереднойДокумент.ПолучитьСсылку();
  …
КонецЦикла;
```

```
3.      // объект ДокументМенеджер.
  // НайтиПоНомеру()
  // НайтиПоРеквизиту()
  // ПустаяСсылка()

// Пример: Проверить, проведен ли документ ПриходнаяНакладная с номером 3.
Если Документы.ПриходнаяНакладная.НайтиПоНомеру(3).Проведен Тогда
  Сообщить("Документ с номером 3 проведен");
КонецЕсли;

// Пример: Проверить, что во всех документах ПриходнаяНакладная заполнен
реквизит Склад.
ПустаяСсылкаСклада = Справочники.Склады.ПустаяСсылка();
Если Не Документы.ПриходнаяНакладная.НайтиПоРеквизиту("Склад",
ПустаяСсылкаСклада).Пустая() Тогда
  Сообщить("Есть документы, у которых не заполнен реквизит Склад");
КонецЕсли;

4.      // объект ДокументМенеджер.
  // Выбрать()

// Пример: Выбрать все документы ПриходнаяНакладная за текущий месяц.
Выборка = Документы.ПриходнаяНакладная.Выбрать(НачалоМесяца(ТекущаяДата()),
КонецМесяца(ТекущаяДата()));
Пока Выборка.Следующий() Цикл
  Сообщить(Выборка);
КонецЦикла;

5.      // объект ДокументМенеджер.
  // СоздатьДокумент()
```

```
// Пример: Создать новый документ ПриходнаяНакладная.
НовыйДокумент = Документы.ПриходнаяНакладная.СоздатьДокумент();
НовыйДокумент.Дата = ТекущаяДата();
НовыйДокумент.Склад = Справочники.Склады.Основной;
// Заполнить табличную часть Материалы
НоваяСтрокаТабличнойЧасти = НовыйДокумент.Материалы.Добавить();
НоваяСтрокаТабличнойЧасти.Материал = Справочники.Номенклатура.НайтиПоКоду(6);
НоваяСтрокаТабличнойЧасти.Количество = 10;
НоваяСтрокаТабличнойЧасти.Цена = 22,5;
НоваяСтрокаТабличнойЧасти.Сумма = 225;
НовыйДокумент.Записать();

6.      // объект ДокументОбъект., объект ДокументСсылка.
    // Ссылка

// Пример: в модуле объекта вызвать процедуру проверки заполнения реквизитов
документа.
Если Не ПроверитьЗаполнениеРеквизитов(ЭтотОбъект.Ссылка) Тогда
  Сообщить("Реквизиты документа не заполнены!");
КонецЕсли;

7.      // объект ДокументСсылка., объект ДокументОбъект.
  // ПолучитьОбъект()
  // Скопировать()

// Пример: пометить документ на удаление.
НенужныйДокумент = Документы.ОказаниеУслуги.НайтиПоНомеру(13).ПолучитьОбъект();
НенужныйДокумент.УстановитьПометкуУдаления(Истина);

8.      // объект ДокументВыборка.
```

```
  // Ссылка

// Сформировать список ссылок на все документы "ПриходнаяНакладная" за текущий
месяц.
СписокНакладных = Новый СписокЗначений;
Выборка = Документы.ПриходнаяНакладная.Выбрать(НачалоМесяца(ТекущаяДата()),
КонецМесяца(ТекущаяДата()));
Пока Выборка.Следующий() Цикл
  СписокНакладных.Добавить(Выборка.Ссылка);
КонецЦикла;

9.      // объект ДокументВыборка.
    // ПолучитьОбъект()

// Пример: удалить все документы "ПриходнаяНакладная".
Выборка = Документы.ПриходнаяНакладная.Выбрать();
Пока Выборка.Следующий() Цикл
  Выборка.ПолучитьОбъект().Удалить();
КонецЦикла;
```

Последовательность событий при записи документа из формы документа

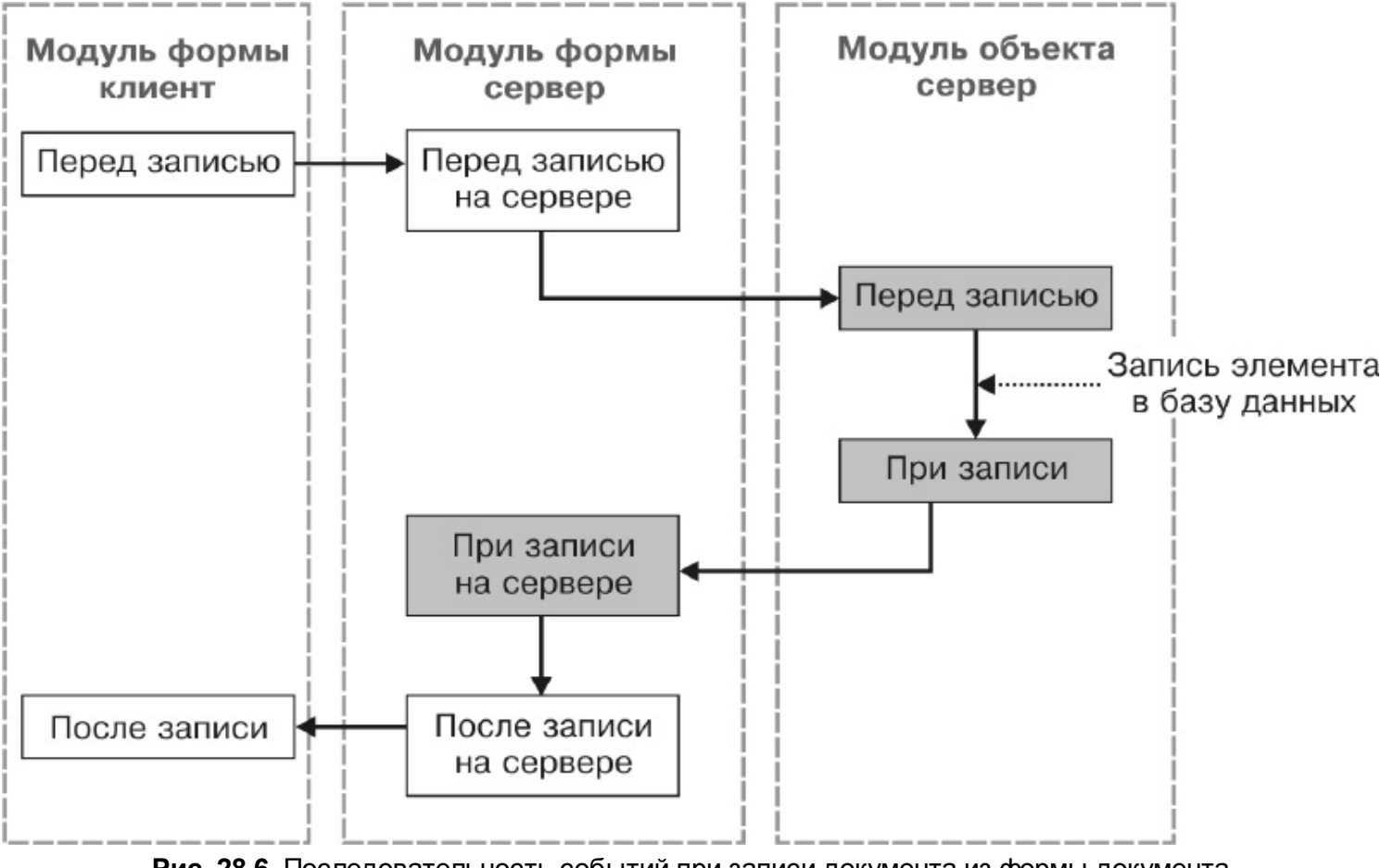


Рис. 28.6. Последовательность событий при записи документа из формы документа

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Последовательность событий при проведении документа из формы документа (провести и закрыть)

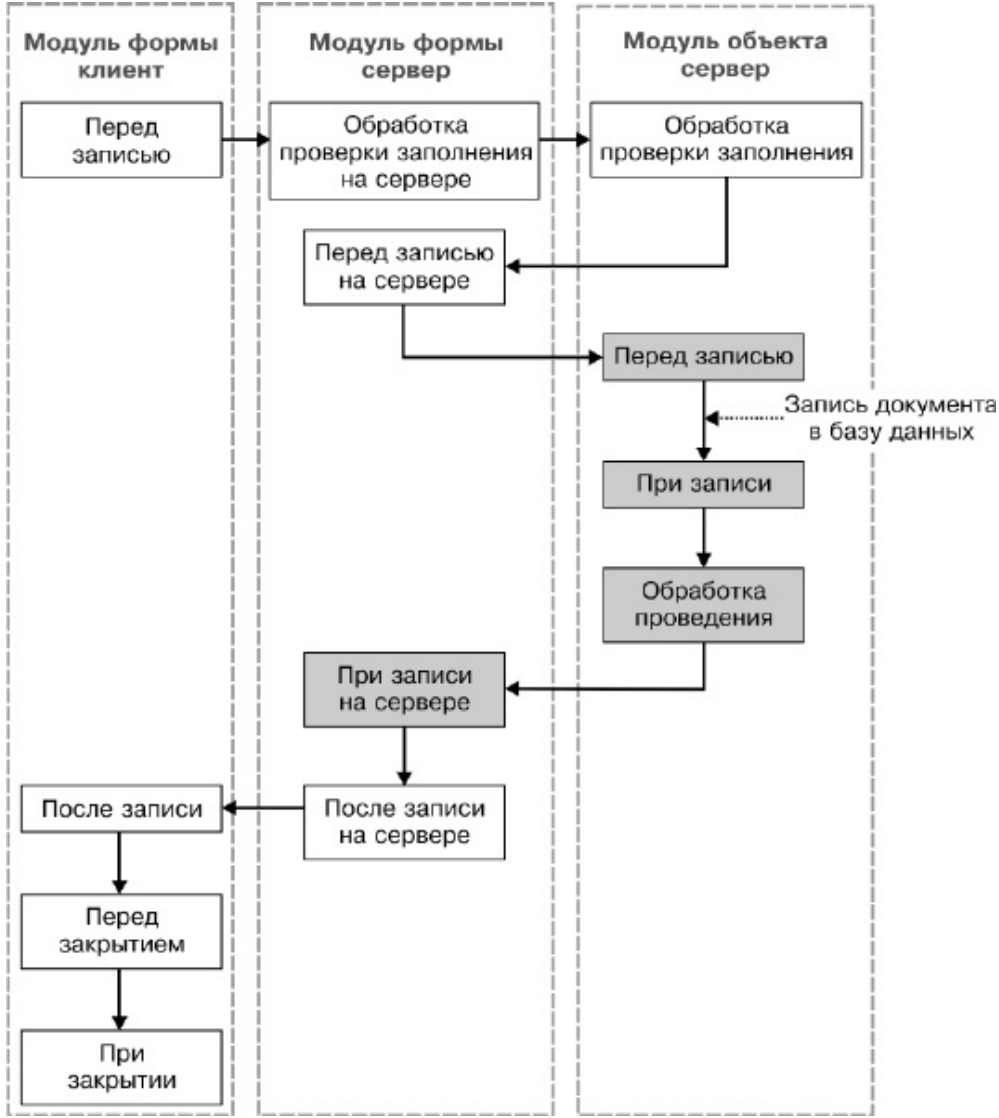

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись документа в базу данных
При записи
Обработка проведения
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.7. Последовательность событий при проведении документа из формы документа

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Последовательность событий при отмене проведения документа из формы документа

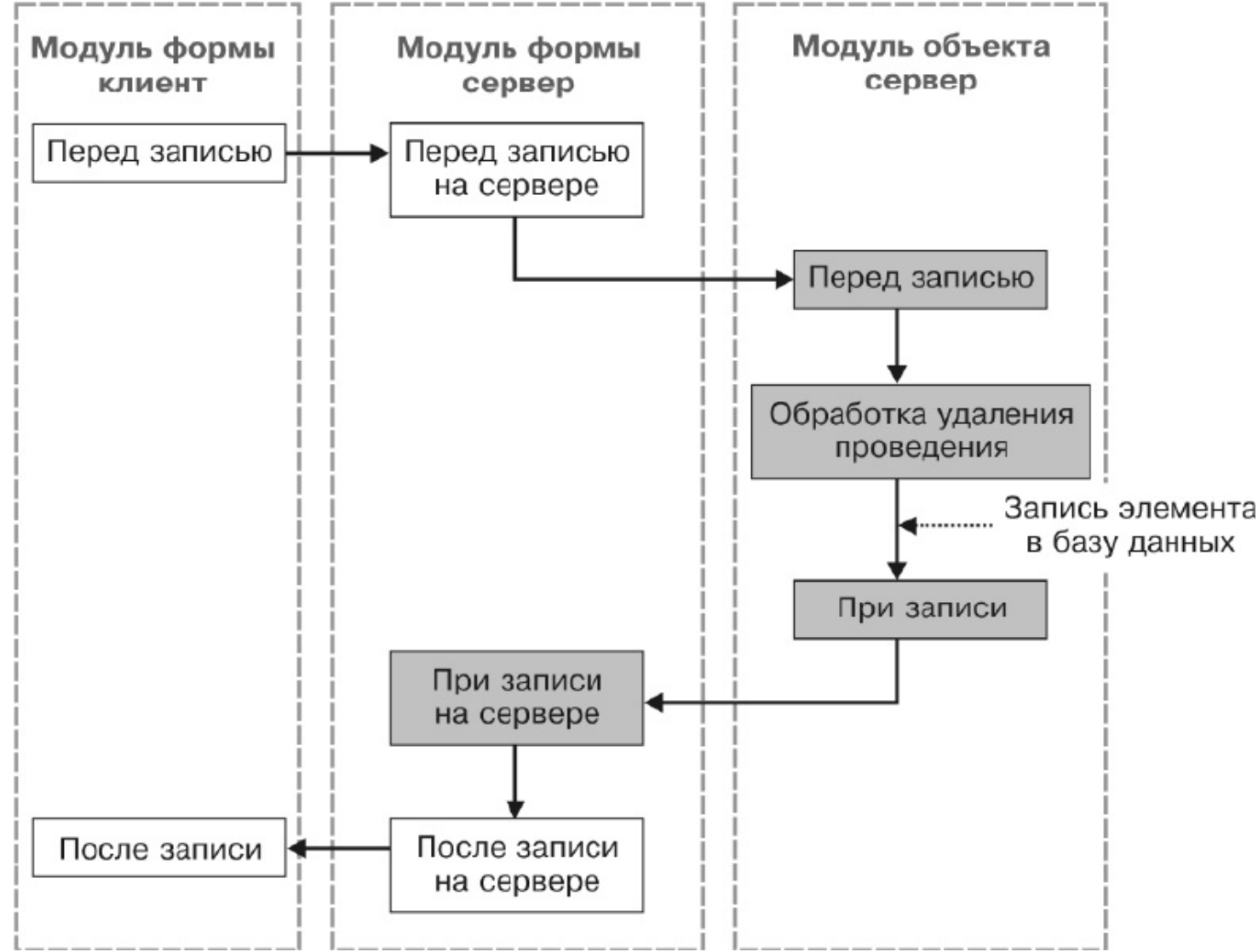

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Перед записью на сервере
Перед записью
Обработка удаления проведения
Запись элемента в базу данных
При записи
При записи на сервере
После записи
После записи на сервере

Рис. 28.8. Последовательность событий при отмене проведения документа из формы документа

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Перечисления

Объекты встроенного языка для работы с перечислениями

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с перечислениями (рис. 28.9).

Рис. 28.9. Объекты встроенного языка для работы с перечислениями

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными. Метод объекта, от которого идет стрелка, приводится в листинге под соответствующей цифрой, а объект, к которому идет стрелка, – это тип возвращаемого метода.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

Ниже приведены примеры использования объектов встроенного языка для работы с перечислениями (листинг 28.7).

Листинг 28.7. Примеры использования объектов

```
1.      // Глобальный контекст
  // Перечисления

// Пример: получить значение перечисления по индексу.
Перечисления.ВидыНоменклатуры.Получить(0);

2.      // объект ПеречислениеМенеджер
  // .
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;

// Пример: получить количество значений перечисления
Перечисления. ["ВидыНоменклатуры"].Количество();

3.      // объект ПеречислениеМенеджер.
  // .
  // []
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;
  // ПустаяСсылка()

// Пример: получить пустую ссылку на значение перечисления.
…
ПустаяСсылкаПеречисления = Перечисления.ВидыНоменклатуры.ПустаяСсылка();
Если ТекущаяНоменклатура.ВидНоменклатуры = ПустаяСсылкаПеречисления Тогда
```

```
    // Предложить заполнение вида номенклатуры.
    …
КонецЕсли;
…
```

Планы видов характеристик

Объекты встроенного языка для работы с планами видов характеристик

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с планами видов характеристик (рис. 28.10).

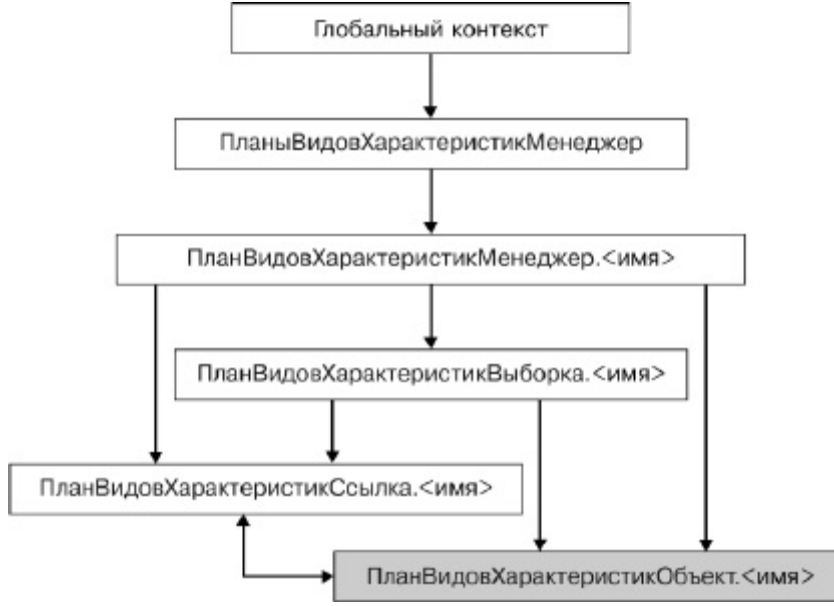


Рис. 28.10. Объекты встроенного языка для работы с планами видов характеристик

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

Свойства и методы взаимодействия перечисленных объектов в большинстве

своем аналогичны свойствам и методам объектов, предназначенных для работы со справочниками (см. раздел «Объекты встроенного языка для работы со справочниками»).

Последовательность событий при записи вида характеристики из формы элемента (записать и закрыть)

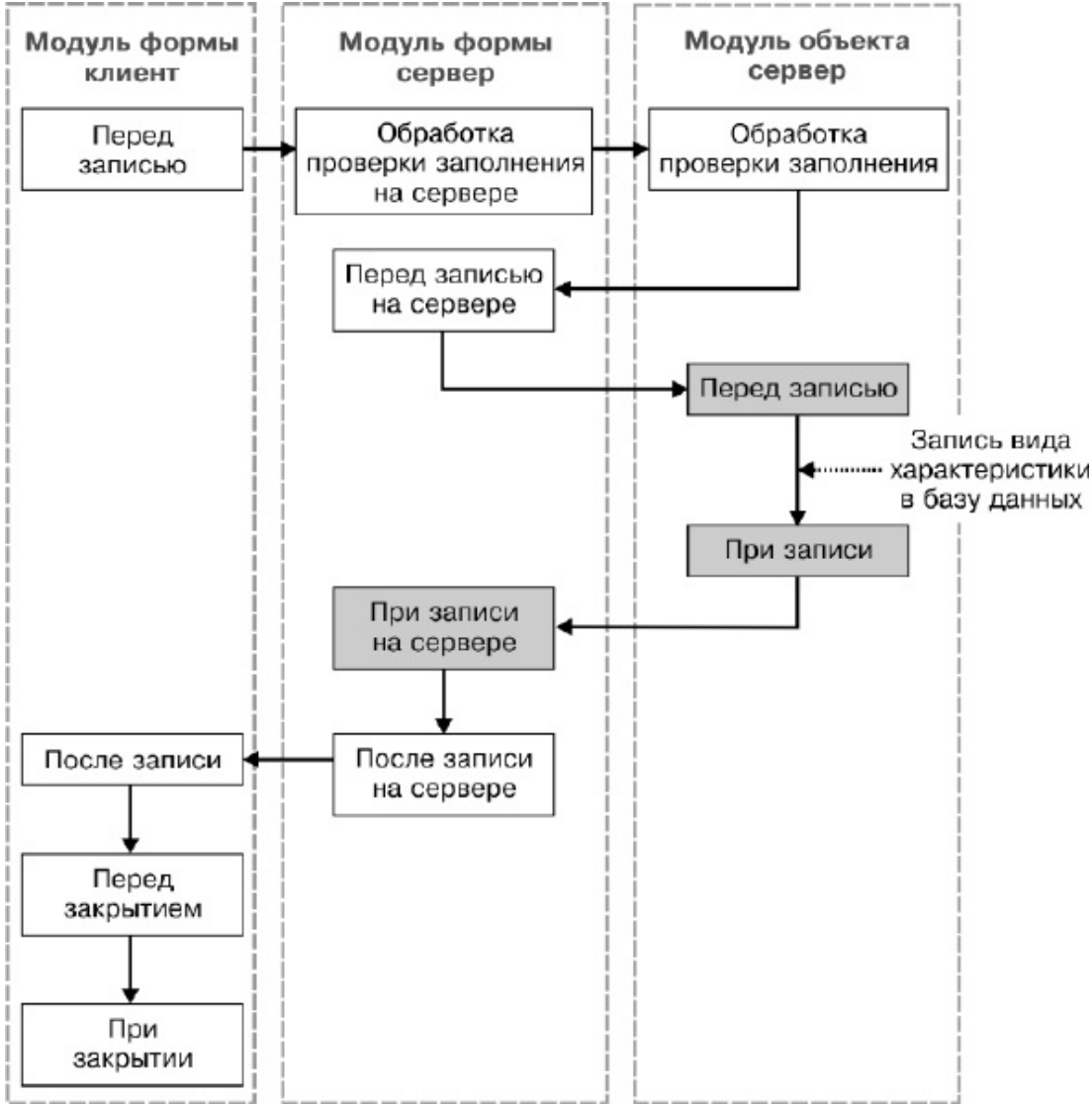

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись вида характеристики в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.11. Последовательность событий при записи вида характеристики из формы элемента

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Планы счетов

Объекты встроенного языка для работы с планами счетов

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с планами счетов (рис. 28.12).

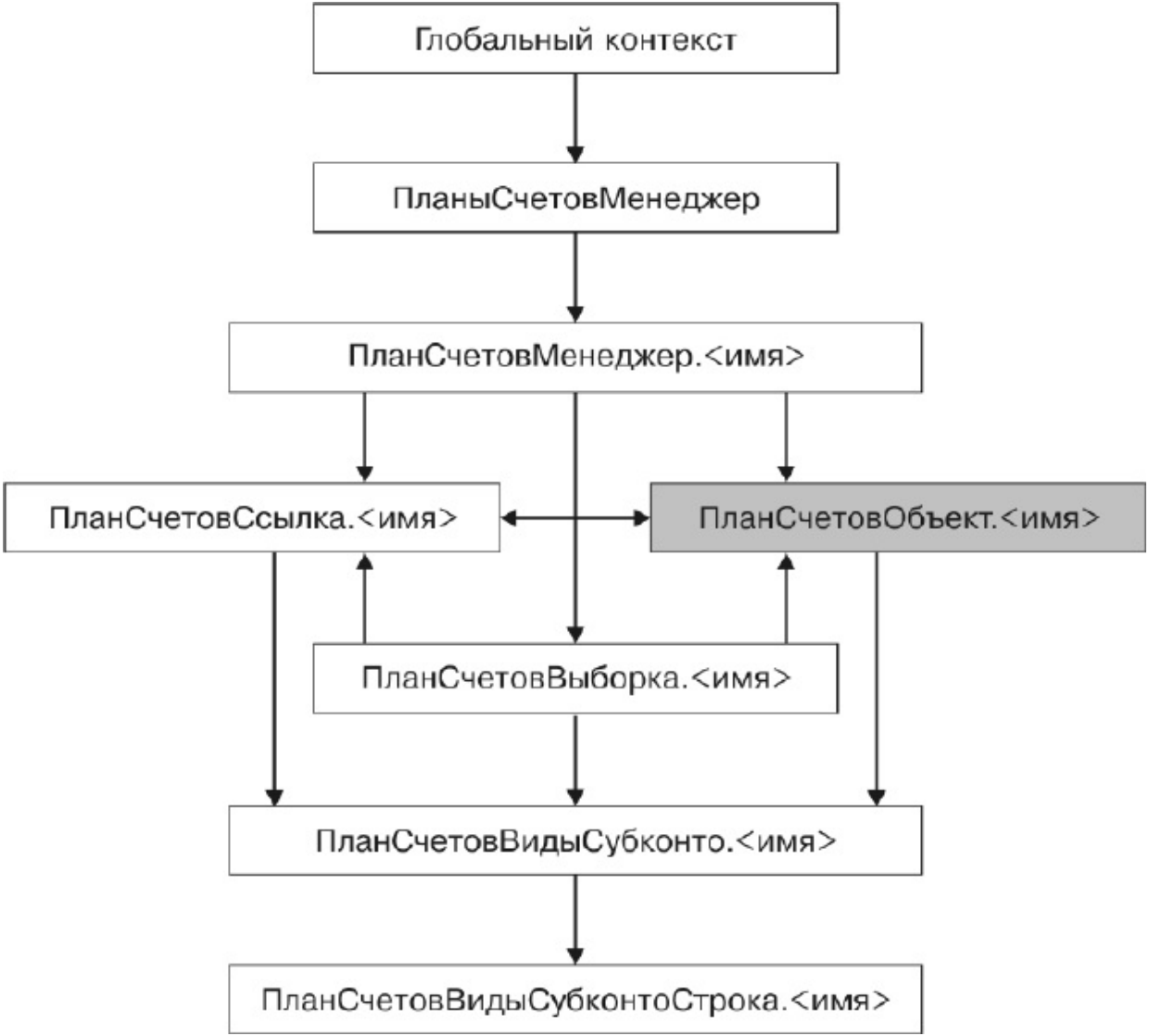

Глобальный контекст
ПланыСчетовМенеджер
ПланСчетовМенеджер.<имя>
ПланСчетовСсылка.<имя>
ПланСчетовОбъект.<имя>
ПланСчетовВыборка.<имя>
ПланСчетовВидыСубконто.<имя>
ПланСчетовВидыСубконтоСтрока.<имя>

Рис. 28.12. Объекты встроенного языка для работы с планами счетов

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

ПланСчетовВидыСубконто.<имя>. Используется для доступа к методам специальной табличной части счета *ВидыСубконто* в целом.

ПланСчетовВидыСубконтоСтрока.<имя>. Строка специальной табличной части счета *ВидыСубконто*.

Свойства и методы взаимодействия перечисленных объектов в большинстве своем аналогичны свойствам и методам объектов, предназначенных для работы со справочниками (см. раздел «Объекты встроенного языка для работы со справочниками»).

Последовательность событий при записи счета из формы счета (записать и закрыть)

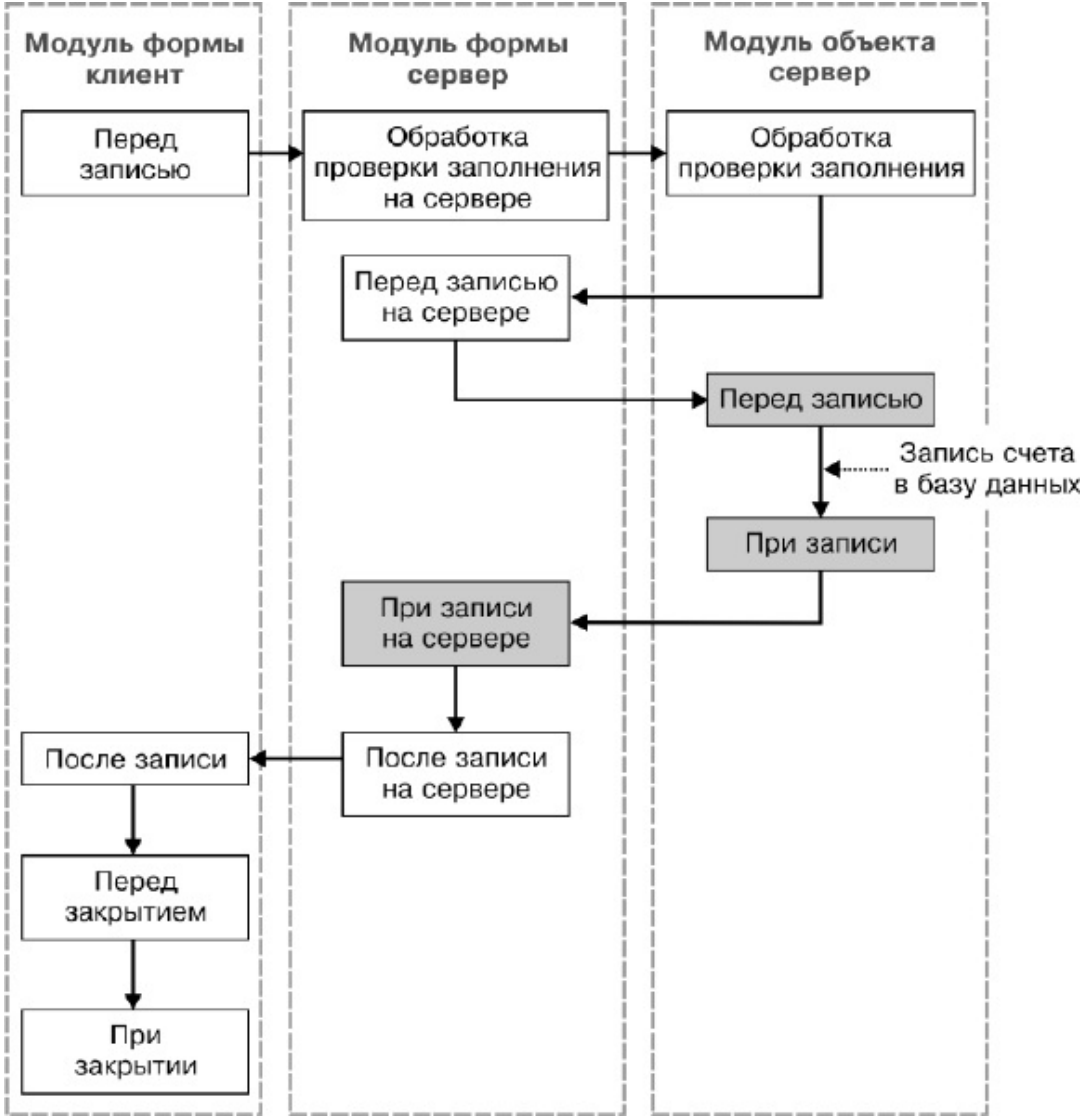

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись счета в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.13. Последовательность событий при записи счета из формы счета

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Планы видов расчета

Объекты встроенного языка для работы с планом видов расчета

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с планами видов расчета (рис. 28.14).

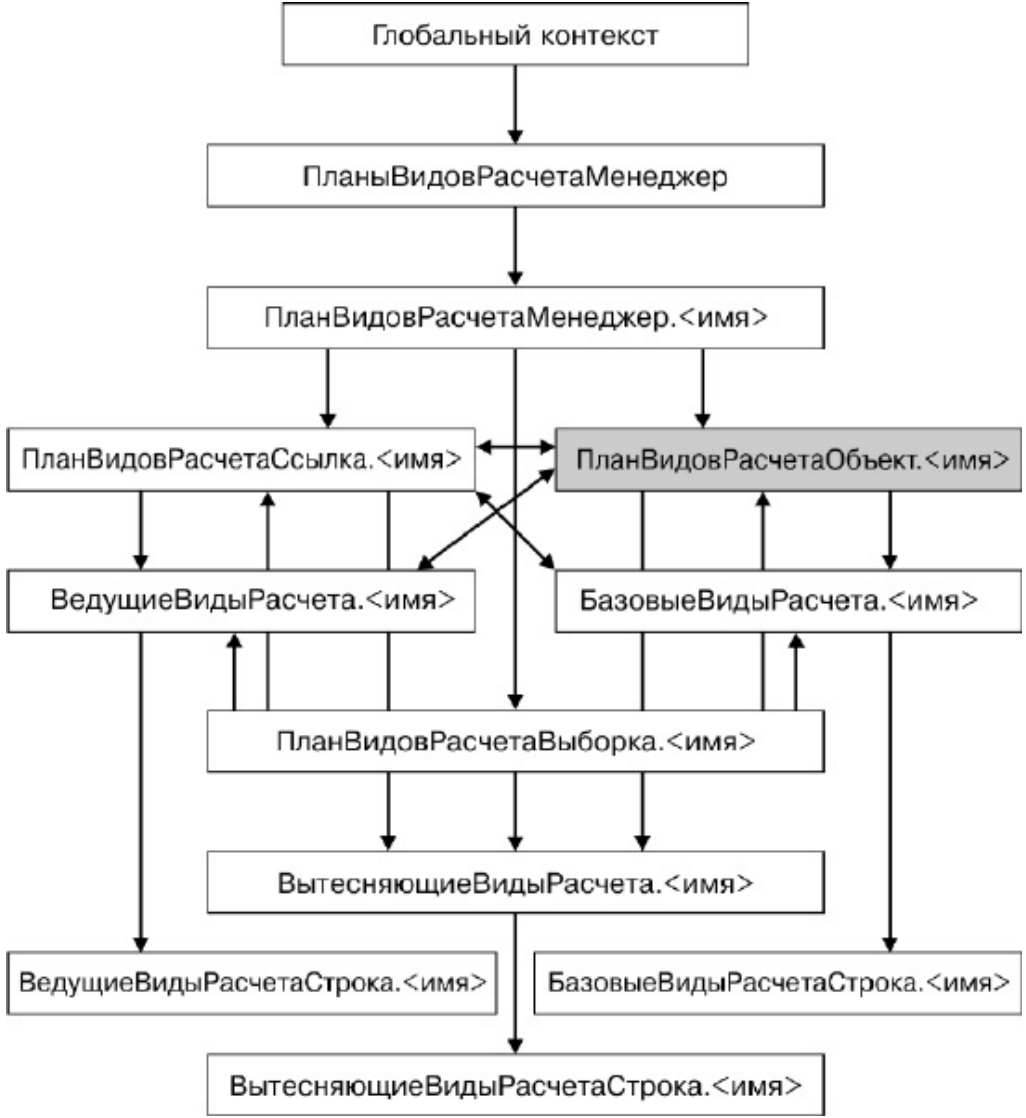

Глобальный контекст
ПланыВидовРасчетаМенеджер
ПланВидовРасчетаМенеджер.<имя>
ПланВидовРасчетаСсылка.<имя>
ПланВидовРасчетаОбъект.<имя>
ВедущиеВидыРасчета.<имя>
БазовыеВидыРасчета.<имя>
ПланВидовРасчетаВыборка.<имя>
ВытесняющиеВидыРасчета.<имя>
ВедущиеВидыРасчетаСтрока.<имя>
БазовыеВидыРасчетаСтрока.<имя>
ВытесняющиеВидыРасчетаСтрока.<имя>

Рис. 28.14. Объекты встроенного языка для работы с планами видов расчета

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

ВытесняющиеВидыРасчета.<имя>. Предопределенная табличная часть вида расчета – список вытесняющих видов расчета. Такая табличная часть определена только для планов видов расчета с признаком *ИспользуетПериодДействия*. Имеет единственную колонку – *ВидРасчета* типа *ПланВидовРасчетаСсылка.<имя>*.

ВытесняющиеВидыРасчетаСтрока.<имя>. Строка предопределенной таблицы вытесняющих видов расчета.

ВедущиеВидыРасчета.<имя>. Предопределенная табличная часть вида расчета – список ведущих видов расчета. Имеет единственную колонку – *ВидРасчета* типа *ПланВидовРасчетаСсылка.<имя>*.

ВедущиеВидыРасчетаСтрока.<имя>. Строка предопределенной таблицы ведущих видов расчета.

БазовыеВидыРасчета.<имя>. Предопределенная табличная часть вида расчета – список ведущих видов расчета. Такая табличная часть (свойство *БазовыеВидыРасчета*) определена только для планов видов расчета со свойством *ЗависимостьОтБазы*, не равным значению *Не зависит*. Имеет единственную колонку – *Вид Расчета* типа *ПланВидовРасчетаСсылка.<имя>*.

БазовыеВидыРасчетаСтрока.<имя>. Строка предопределенной таблицы базовых видов расчета.

Свойства и методы взаимодействия перечисленных объектов в большинстве своем аналогичны у объектов, предназначенных для работы со справочниками (см. раздел «Объекты встроенного языка для работы со справочниками»).

Последовательность событий при записи вида расчета из формы вида расчета (записать и закрыть)

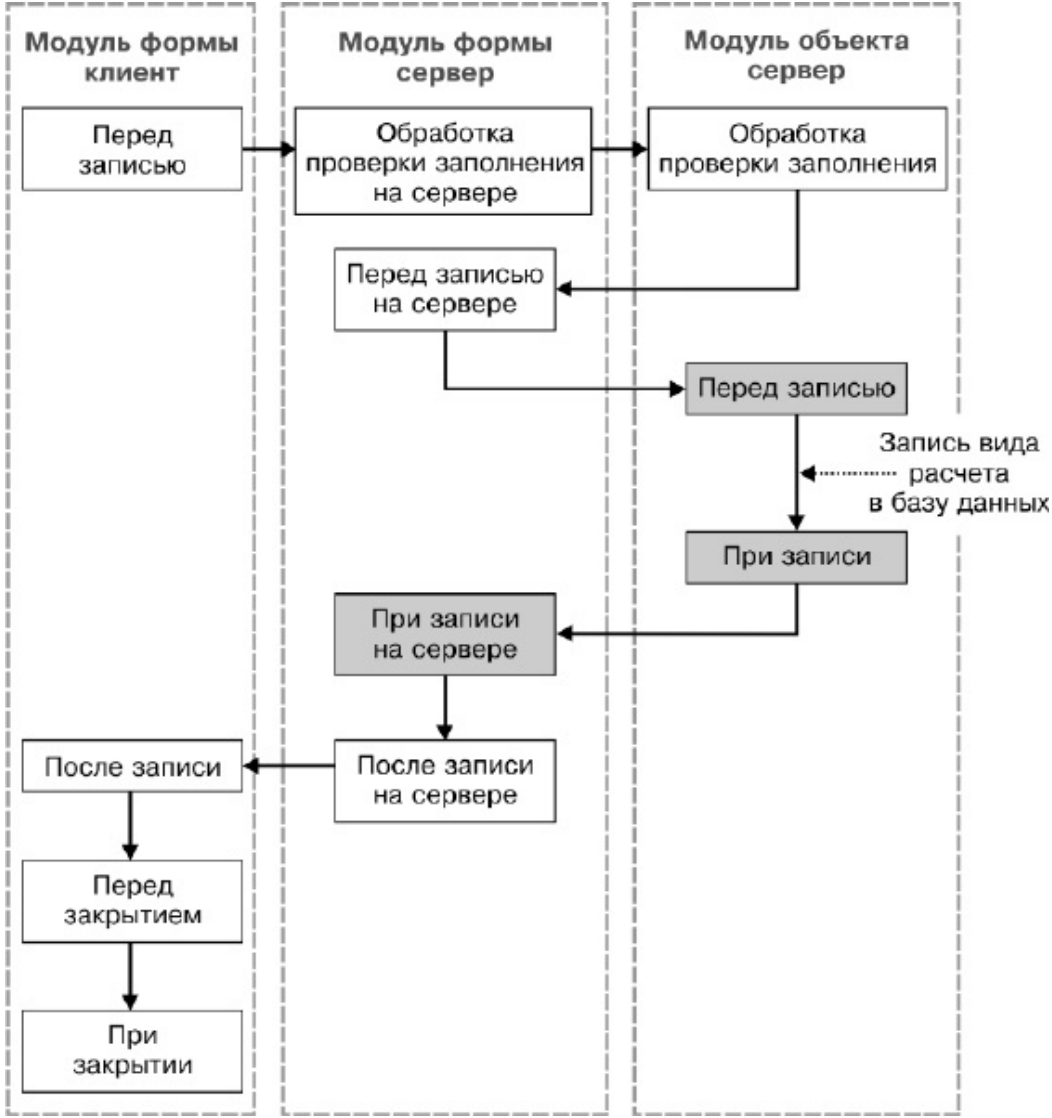

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись вида расчета в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.15. Последовательность событий при записи вида расчета из формы вида расчета

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Регистры сведений

Объекты встроенного языка для работы с регистрами сведений

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с регистрами сведений (рис. 28.16).

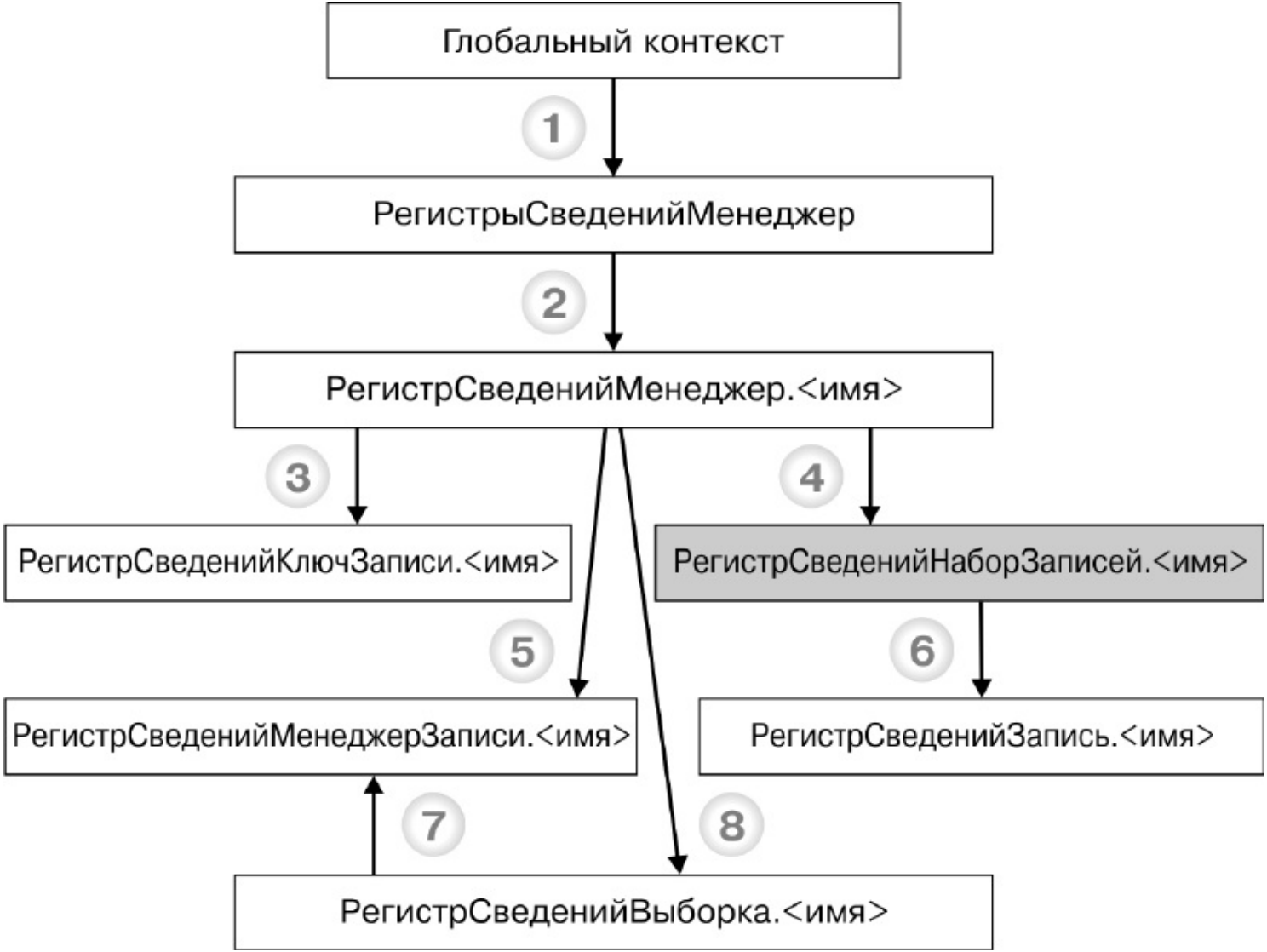

Глобальный контекст
1
РегистрыСведенийМенеджер
2
РегистрСведенийМенеджер.<имя>
3
4
РегистрСведенийКлючЗаписи.<имя>
РегистрСведенийНаборЗаписей.<имя>
5
6
РегистрСведенийМенеджерЗаписи.<имя>
РегистрСведенийЗапись.<имя>
7
8
РегистрСведенийВыборка.<имя>

Рис. 28.16. Объекты встроенного языка для работы с регистрами сведений

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными. Метод объекта, от которого идет стрелка, приводится в листинге под соответствующей цифрой, а объект, к которому идет стрелка, – это тип возвращаемого метода.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

РегистрСведенийМенеджерЗаписи.<имя>. Позволяет читать, записывать и удалять отдельную запись регистра сведений. Используется только для регистров сведений, не изменяемых регистраторами, т. е. для которых в конфигураторе установлен режим записи *Независимый*.

РегистрСведенийЗапись.<имя>. Предоставляет доступ к записи регистра сведений. Объект не создается непосредственно, а предоставляется другими объектами, связанными с регистром сведений. Например, данный объект представляет записи регистра в наборе записей.

РегистрСведенийКлючЗаписи.<имя>. Представляет собой набор значений,

однозначно идентифицирующих запись регистра. Объект используется в тех случаях, когда необходимо сослаться на определенную запись. Например, он выступает в качестве значения свойства *ТекущаяСтрока* табличного поля, отображающего список записей регистра.

Ниже приведены примеры использования объектов встроенного языка для работы с регистрами сведений (листинг 28.8).

Листинг 28.8. Примеры использования объектов

```
1.      // Глобальный контекст
  // РегистрыСведений

// Пример: получить текущую цену из периодического регистра сведений "Цены".
Элемент = Справочники.Номенклатура.НайтиПоКоду(4);
Отбор = Новый Структура("Номенклатура", Элемент);
ЗначенияРесурсов = РегистрыСведений.Цены.ПолучитьПоследнее(ТекущаяДата(),
Отбор);
Цена = ЗначенияРесурсов.Цена;

2.      // объект РегистрыСведенийМенеджер
  // .
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;

// Пример: Получить начальную цену из периодического регистра сведений Цены.
ИмяРегистра = "Цены";
Услуга = Справочники.Номенклатура.НайтиПоНаименованию("Диагностика");
```

```
Отбор = Новый Структура;
Отбор.Вставить("Номенклатура", Услуга );
Цена = РегистрыСведений[ИмяРегистра].ПолучитьПервое(ТекущаяДата(), Отбор).Цена;

3.      // объект РегистрСведенийМенеджер.
  // СоздатьКлючЗаписи()

// Пример: активизировать требуемую строку списка регистра сведений.
СтруктураКлючевыхПолей = Новый Структура;
СтруктураКлючевыхПолей.Вставить("Период", Дата("20040331000000"));
СтруктураКлючевыхПолей.Вставить("Номенклатура",
Справочники.Номенклатура.НайтиПоКоду("0000006"));
Элементы.Материалы.ТекущаяСтрока =
РегистрыСведений.Цены.СоздатьКлючЗаписи(СтруктураКлючевыхПолей);

4.      // объект РегистрСведенийМенеджер.
  // СоздатьНаборЗаписей()

// Пример: показать номенклатуру, цена на которую была установлена в заданную
дату и время.
Набор = РегистрыСведений.Цены.СоздатьНаборЗаписей();
Набор.Отбор.Период.Установить(ЗаданнаяДата, Истина);
Набор.Прочитать();
Для Каждого ОчереднаяЗапись Из Набор Цикл
  Сообщить("Номенклатура = "+ ОчереднаяЗапись.Номенклатура +", цена = "+
  ОчереднаяЗапись.Цена);
КонецЦикла;

5.      // объект РегистрСведенийМенеджер.
  // СоздатьМенеджерЗаписи()
```

```
// Пример: добавить новое значение цены в регистр "Цены".
Запись = РегистрыСведений.Цены.СоздатьМенеджерЗаписи();
Запись.Период = ТекущаяДата();
Запись.Номенклатура = Справочники.Номенклатура.НайтиПоКоду("0000005");
Запись.Цена = 568;
Запись.Записать();

6.      // объект РегистрСведенийНаборЗаписей.
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;

// Пример: показать номенклатуру, цена на которую была установлена в заданную
дату и время.
Набор = РегистрыСведений.Цены.СоздатьНаборЗаписей();
Набор.Отбор.Период.Установить(ЗаданнаяДата, Истина);
Набор.Прочитать();
Для Каждого ОчереднаяЗапись Из Набор Цикл
  Сообщить("Номенклатура = "+ ОчереднаяЗапись.Номенклатура + ", цена = " +
  ОчереднаяЗапись.Цена);
КонецЦикла;

7.      // объект РегистрСведенийВыборка.
  // ПолучитьМенеджерЗаписи()

// Пример: удалить все записи регистра сведений за текущий месяц.
Выборка = РегистрыСведений.Цены.Выбрать(НачалоМесяца(ТекущаяДата()),
КонецМесяца(ТекущаяДата()));
Пока Выборка.Следующий() цикл Выборка.
  ПолучитьМенеджерЗаписи().Удалить();
КонецЦикла;
```

```
8.      // объект РегистрСведенийМенеджер.
  // Выбрать()
  // ВыбратьПоРегистратору()

// Пример: показать изменение цен на элемент номенклатуры в течение года.
Отбор = Новый Структура("Номенклатура",
Справочники.Номенклатура.НайтиПоКоду("0000005"));
Выборка = РегистрыСведений.Цены.Выбрать(НачалоГода(ТекущаяДата()),
ТекущаяДата(), Отбор);
Пока Выборка.Следующий() цикл
  Сообщить("Дата = " + Выборка.Период + ", цена = " + Выборка.Цена);
КонецЦикла;
```

Последовательность событий при сохранении данных из формы записи регистра сведений (записать и закрыть)

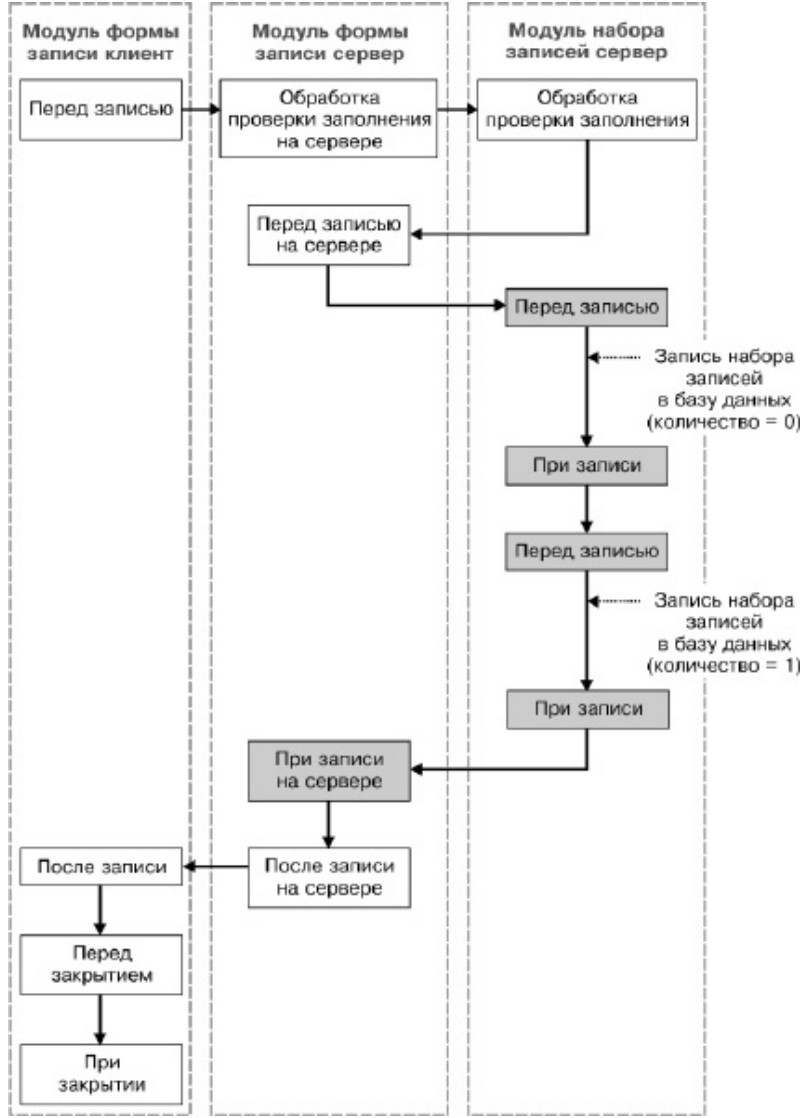

Модуль формы записи клиент
Модуль формы записи сервер
Модуль набора записей сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись набора записей в базу данных (количество = 0)
При записи
Перед записью
Запись набора записей в базу данных (количество = 1)
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.17. Последовательность событий при сохранении данных из формы записи регистра сведений

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Работа с формой записи регистра сведений осуществляется при помощи объекта *РегистрСведенийМенеджерЗаписи.<имя>*, который, в свою очередь, использует объект *РегистрСведенийНаборЗаписей.<имя>*.

Особенности внутренней реализации объекта *РегистрСведенийМенеджерЗаписи.<имя>* таковы, что в случае сохранения существующей записи регистра сведений обработчики события *Перед Записью()* и *ПриЗаписи()* модуля набора записей будут вызваны дважды: сначала для старого набора записей (с количеством записей 0) и затем для нового (с количеством записей 1).

Последовательность событий при сохранении данных из формы набора записей регистра сведений (записать и закрыть)

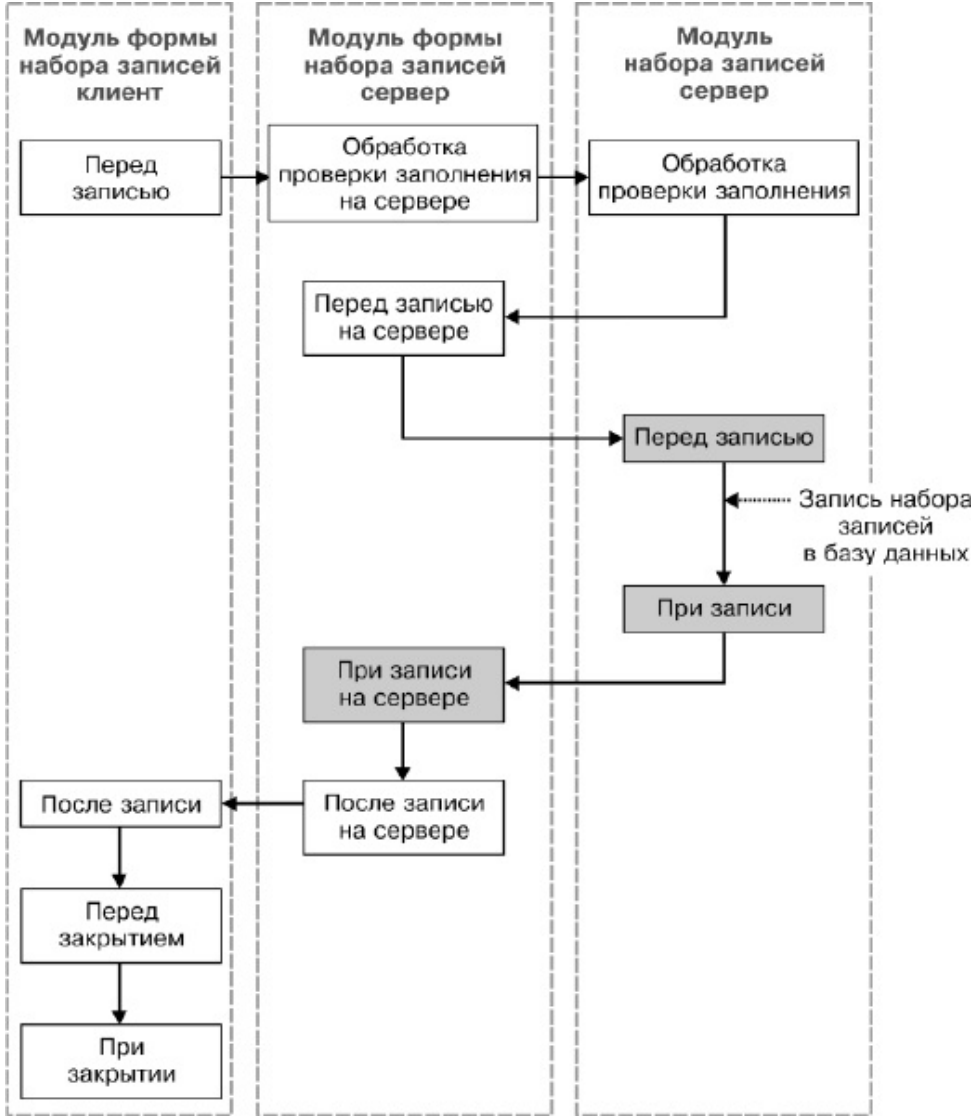

Модуль формы набора записей клиент
Модуль формы набора записей сервер
Модуль набора записей сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись набора записей в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.18. Последовательность событий при сохранении данных из формы набора записей регистра сведений

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Регистры накопления

Объекты встроенного языка для работы с регистрами накопления

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с регистрами накопления (рис. 28.19).

Рис. 28.19. Объекты встроенного языка для работы с регистрами накопления

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными. Метод объекта, от которого идет стрелка, приводится в листинге под соответствующей цифрой, а объект, к которому идет стрелка, – это тип возвращаемого метода.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

РегистрНакопленияЗапись.<имя>. Используется для доступа к записи регистра накопления. Объект не создается непосредственно, а предоставляется другими объектами, отвечающими за регистр накопления. Например, данный объект представляет записи регистра в наборе записей.

РегистрНакопленияКлючЗаписи.<имя>. Представляет собой набор значений, однозначно идентифицирующих запись регистра. Объект используется в тех случаях, когда необходимо сослаться на определенную запись. Например, он выступает в качестве значения свойства *ТекущаяСтрока* табличного поля, отображающего список записей регистра.

Ниже приведены примеры использования объектов встроенного языка для работы с регистрами сведений (листинг 28.9).

Листинг 28.9. Примеры использования объектов

```
1.      // Глобальный контекст
  // РегистрыНакопления

// Пример: выполнить полный пересчет итогов регистра "ОстаткиМатериалов".
РегистрыНакопления.ОстаткиМатериалов.ПересчитатьИтоги();

2.      // объект РегистрыНакопленияМенеджер
  // .
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;
```

```
// Пример: рассчитать итоги регистра "ОстаткиМатериалов" на указанную дату.
ИмяРегистра = ОстаткиМатериалов;
РегистрыНакопления[ИмяРегистра].УстановитьПериодРассчитанныхИтогов(УказаннаяДата)

3.      // объект РегистрНакопленияМенеджер.
  // СоздатьКлючЗаписи()

// Пример: активизировать требуемую строку списка регистра накопления.
СтруктураКлючевыхПолей = Новый Структура;
СтруктураКлючевыхПолей.Вставить("Регистратор",
Документы.ПриходнаяНакладная.НайтиПоНомеру("0000002"));
СтруктураКлючевыхПолей.Вставить("НомерСтроки", 2);
Элементы.Материалы.ТекущаяСтрока =
РегистрыНакопления.ОстаткиМатериалов.СоздатьКлючЗаписи(СтруктураКлючевыхПолей);

4.      // объект РегистрНакопленияМенеджер.
  // СоздатьНаборЗаписей()

// Пример: получить движения документа.
НужныйДокумент = Документы.ПриходнаяНакладная.НайтиПоНомеру(4);
Движения = РегистрыНакопления.ОстаткиМатериалов.СоздатьНаборЗаписей();
Движения.Отбор.Регистратор.Значение = НужныйДокумент;
Движения.Прочитать();

5.      // объект РегистрНакопленияМенеджер.
  // Выбрать()
  // ВыбратьПоРегистратору()

// Пример: выбрать все записи регистра "ОстаткиМатериалов" за текущий месяц.
Выборка =
```

```
РегистрыНакопления.ОстаткиМатериалов.Выбрать(НачалоМесяца(ТекущаяДата()),
КонецМесяца(ТекущаяДата()));

6.      // объект РегистрНакопленияНаборЗаписей.
  // []
  // Для Каждого … Из … Цикл … КонецЦикла;

// Пример: получить движения документа.
НужныйДокумент = Документы.ПриходнаяНакладная.НайтиПоНомеру(4);
Движения = РегистрыНакопления.ОстаткиМатериалов.СоздатьНаборЗаписей();
Движения.Отбор.Регистратор.Значение = НужныйДокумент;
Движения.Прочитать();
Для Каждого ОчередноеДвижение Из Движения Цикл
  // Алгоритм обработки движений
    …
КонецЦикла;
```

Последовательность событий при сохранении набора записей регистра накопления из формы набора записей

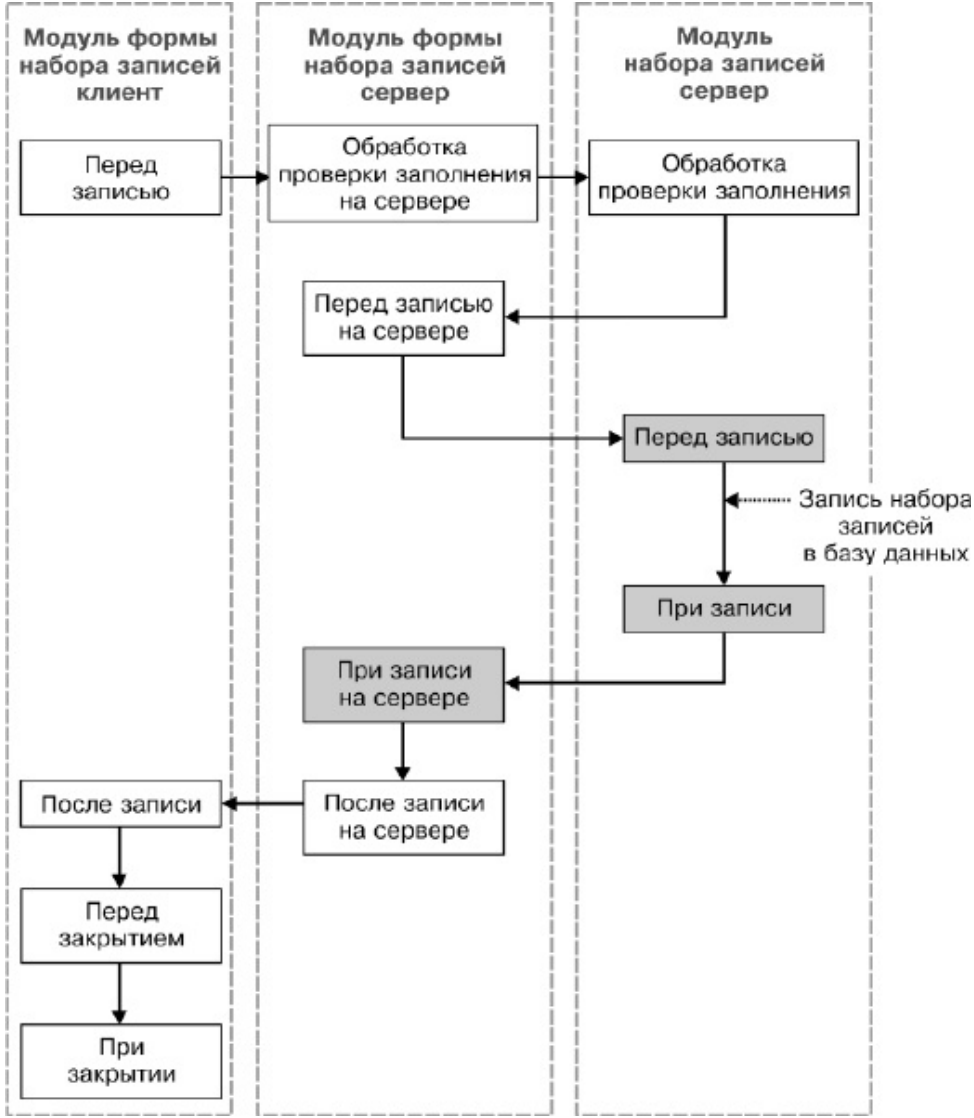

Модуль формы набора записей клиент
Модуль формы набора записей сервер
Модуль набора записей сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись набора записей в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.20. Последовательность событий при сохранении набора записей регистра накопления из формы набора записей

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Регистры бухгалтерии

Объекты встроенного языка для работы с регистрами бухгалтерии

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с регистрами бухгалтерии (рис. 28.21).

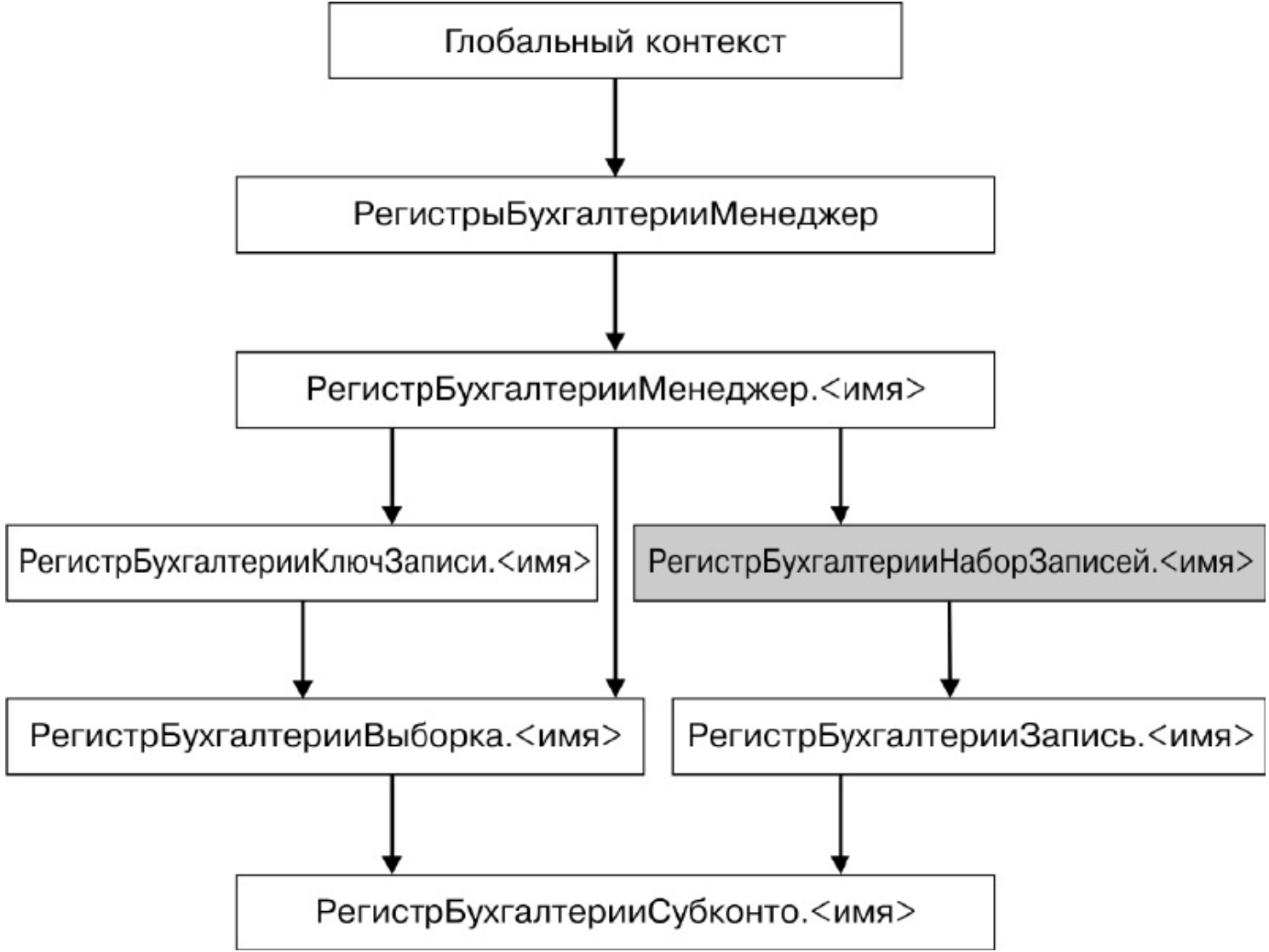

Глобальный контекст
РегистрыБухгалтерииМенеджер
РегистрБухгалтерииМенеджер.<имя>
РегистрБухгалтерииКлючЗаписи.<имя>
РегистрБухгалтерииНаборЗаписей.<имя>
РегистрБухгалтерииВыборка.<имя>
РегистрБухгалтерииЗапись.<имя>
РегистрБухгалтерииСубконто.<имя>

Рис. 28.21. Объекты встроенного языка для работы с регистрами бухгалтерии

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

РегистрБухгалтерииЗапись.<имя>. Используется для доступа к записи регистра бухгалтерии. Объект не создается непосредственно, а предоставляется другими объектами, отвечающими за регистр бухгалтерии. Например, данный объект представляет записи регистра в наборе записей.

РегистрБухгалтерииСубконто.<имя>. Коллекция значений субконто записи регистра бухгалтерии. Установка и получение значения конкретного субконто осуществляются через оператор *[]*, в качестве параметра которому передается вид субконто, или через имя предопределенного субконто.

РегистрБухгалтерииКлючЗаписи.<имя>. Набор значений, однозначно идентифицирующий запись регистра. Объект используется в тех случаях, когда необходимо сослаться на определенную запись. Например, он выступает в

качестве значения свойства *ТекущаяСтрока* табличного поля, отображающего список записей регистра.

Свойства и методы взаимодействия перечисленных объектов в большинстве своем аналогичны у объектов, предназначенных для работы с регистрами накопления (см. раздел «Объекты встроенного языка для работы с регистрами накопления»).

Последовательность событий при сохранении набора записей регистра бухгалтерии из формы

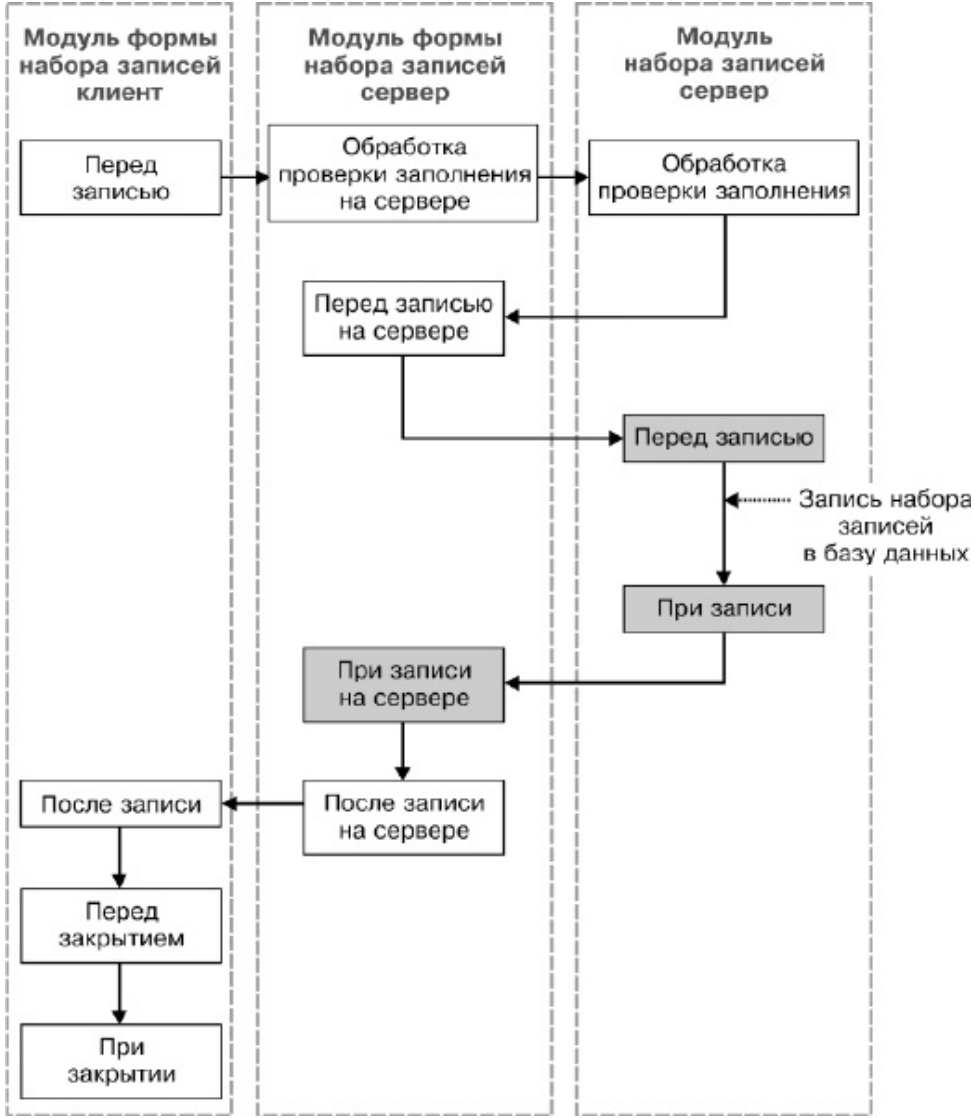

Модуль формы набора записей клиент
Модуль формы набора записей сервер
Модуль набора записей сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись набора записей в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.22. Последовательность событий при сохранении набора записей регистра бухгалтерии из формы

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Регистры расчета

Объекты встроенного языка для работы с регистрами расчетов

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с регистрами расчета (рис. 28.23).

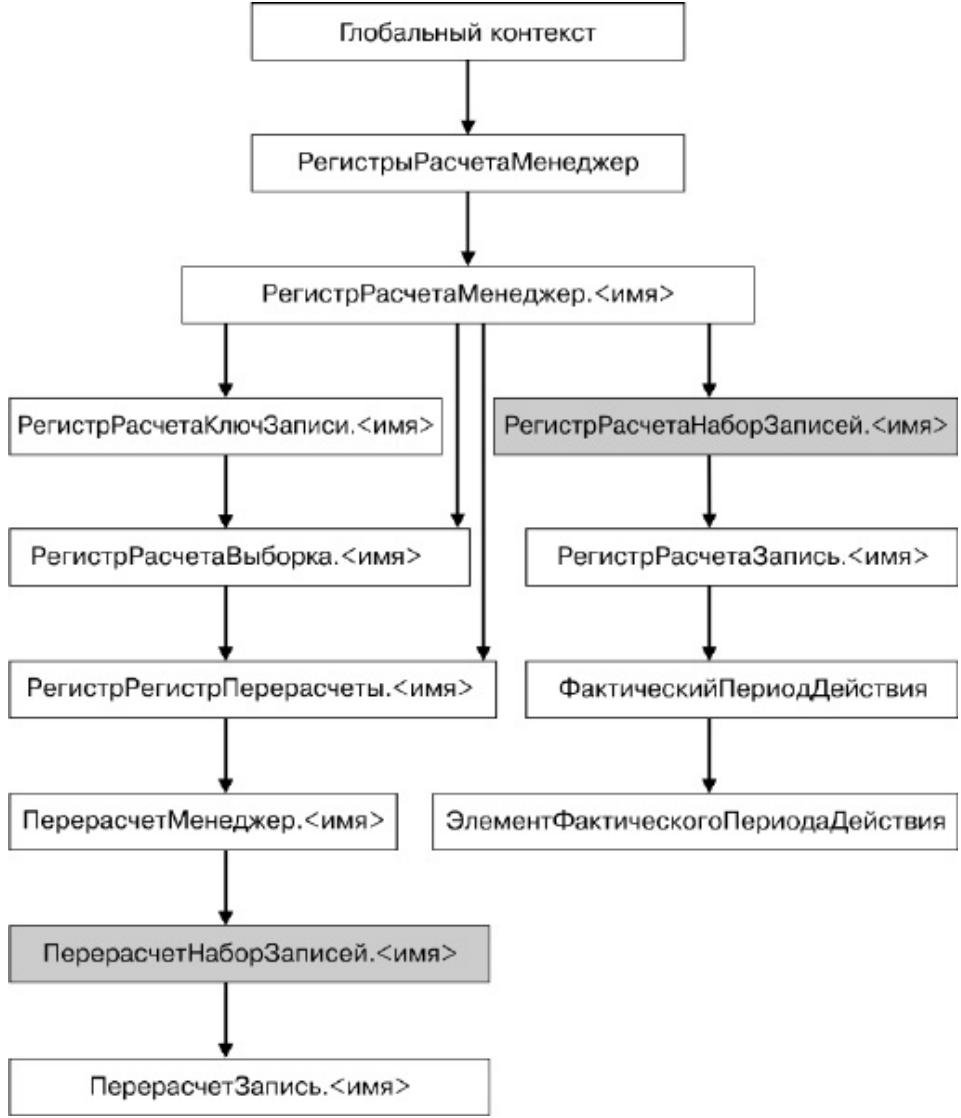

Глобальный контекст
РегистрыРасчетаМенеджер
РегистрРасчетаМенеджер.<имя>
РегистрРасчетаКлючЗаписи.<имя>
РегистрРасчетаНаборЗаписей.<имя>
РегистрРасчетаВыборка.<имя>
РегистрРасчетаЗапись.<имя>
РегистрРегистрПерерасчеты.<имя>
ФактическийПериодДействия
ПерерасчетМенеджер.<имя>
ЭлементФактическогоПериодаДействия
ПерерасчетНаборЗаписей.<имя>
ПерерасчетЗапись.<имя>

Рис. 28.23. Объекты встроенного языка для работы с регистрами расчетов

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

РегистрРасчетаЗапись.<имя>. Используется для доступа к записи регистра расчета. Объект не создается непосредственно, а предоставляется другими объектами, отвечающими за регистр расчета. Например, данный объект представляет записи регистра в наборе записей.

РегистрРасчетаКлючЗаписи.<имя>. Представляет собой набор значений, однозначно идентифицирующих запись регистра. Объект используется в тех случаях, когда необходимо сослаться на определенную запись. Например, он выступает в качестве значения свойства *ТекущаяСтрока* табличного поля, отображающего список записей регистра.

РегистрРасчетаПерерасчеты.<имя>. Менеджер всех менеджеров перерасчетов регистра расчетов.

ПерерасчетМенеджер.<имя>. Менеджер перерасчета служит для получения набора записей перерасчета.

ПерерасчетНаборЗаписей.<имя>. Набор записей перерасчета.

ПерерасчетЗапись.<имя>. Объект используется для доступа к записи перерасчета.

ФактическийПериодДействия. Массив значений типа *ЭлементФактическогоПериодаДействия*.

ЭлементФактическогоПериодаДействия. Элемент фактического периода действия.

Свойства и методы взаимодействия перечисленных объектов в большинстве своем аналогичны у объектов, предназначенных для работы с регистрами накопления (см. раздел «Объекты встроенного языка для работы с регистрами накопления»).

Последовательность событий при сохранении набора записей регистра расчета из формы

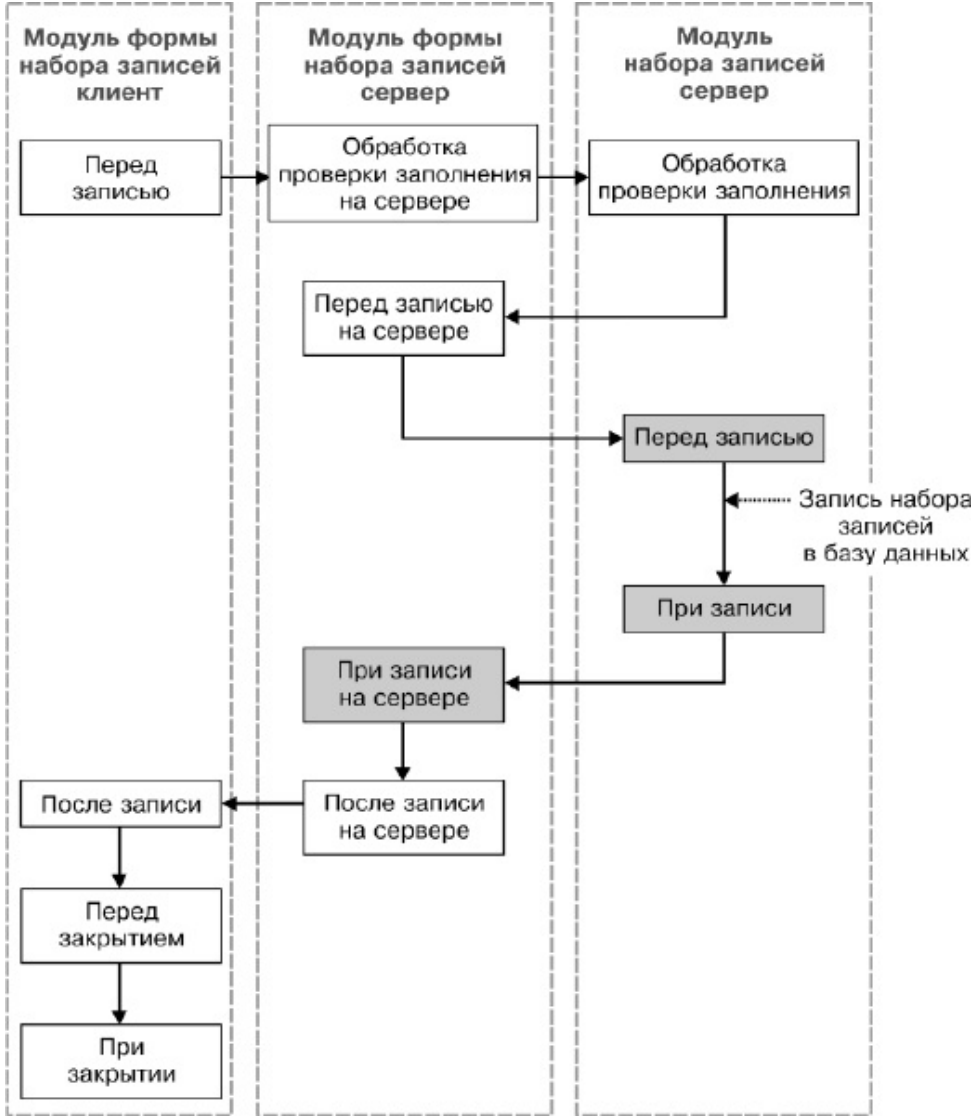

Модуль формы набора записей клиент
Модуль формы набора записей сервер
Модуль набора записей сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись набора записей в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.24. Последовательность событий при сохранении набора записей регистра расчета из формы

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Планы обмена

Объекты встроенного языка для работы с планами обмена

На следующей схеме изображено взаимодействие объектов встроенного языка для работы с планами обмена (рис. 28.25).

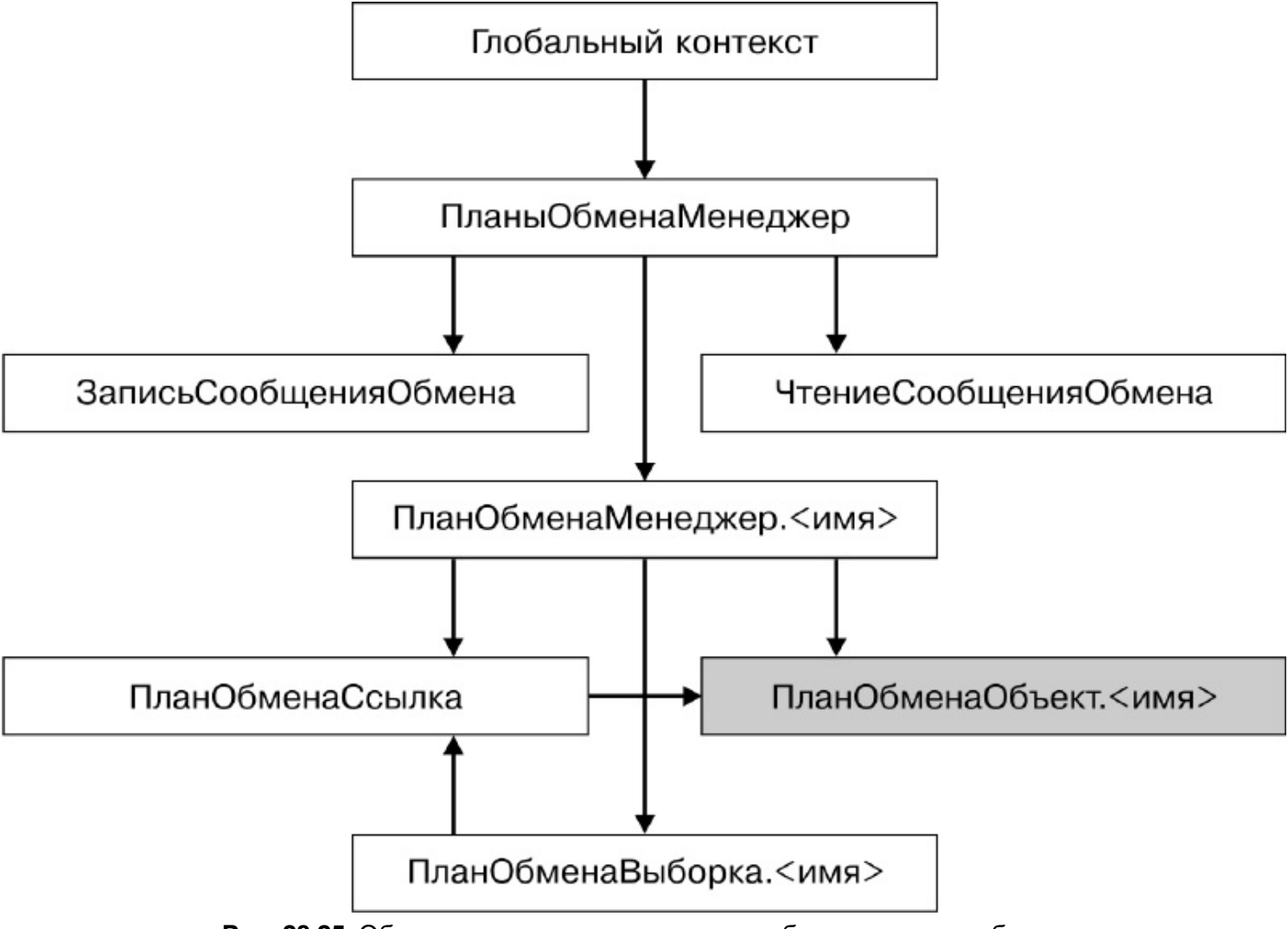


Рис. 28.25. Объекты встроенного языка для работы с планами обмена

ПРИМЕЧАНИЕ

Заливкой выделен объект манипулирования данными.

Узнай больше!

Про основные виды объектов встроенного языка можно прочитать в главе «Объекты встроенного языка для работы с прикладными данными».

ЗаписьСообщенияОбмена – объект предназначен для организации записи сообщения обмена данными.

ЧтениеСообщенияОбмена – объект предназначен для приема сообщений обмена данными. При начале чтения он осуществляет проверку правильности задания заголовка сообщения и отвергает неправильные сообщения. При завершении чтения данный объект модифицирует значение реквизита *НомерПринятого* соответствующего узла плана обмена в соответствии с номером принятого сообщения.

Свойства и методы взаимодействия перечисленных объектов в большинстве своем аналогичны у объектов, предназначенных для работы со справочниками (см. раздел «Объекты встроенного языка для работы со справочниками»).

Последовательность событий при записи узла плана обмена из

формы узла (записать и закрыть)

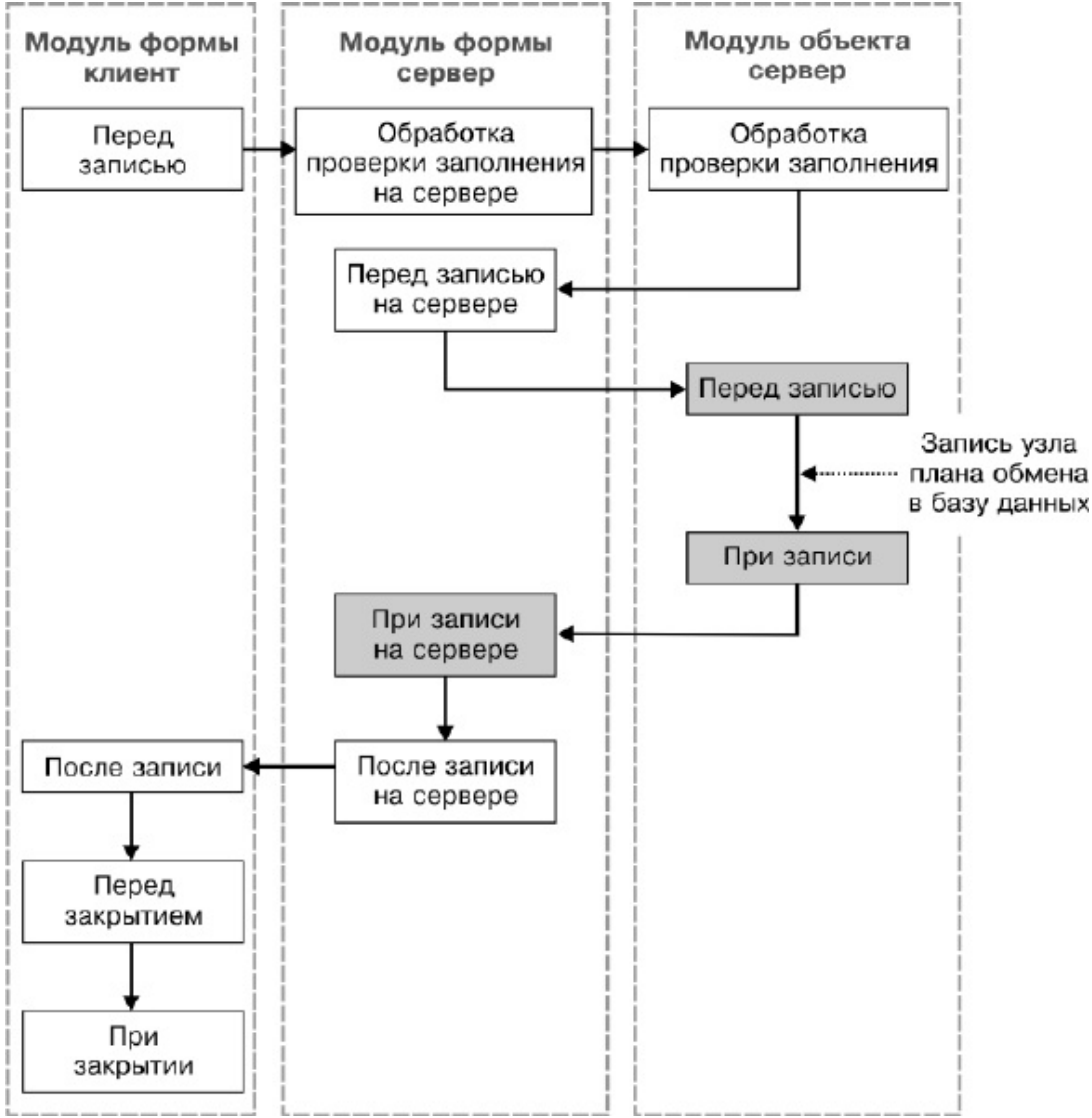

Модуль формы клиент
Модуль формы сервер
Модуль объекта сервер
Перед записью
Обработка проверки заполнения на сервере
Обработка проверки заполнения
Перед записью на сервере
Перед записью
Запись узла плана обмена в базу данных
При записи
При записи на сервере
После записи
После записи на сервере
Перед закрытием
При закрытии

Рис. 28.26. Последовательность событий при записи узла плана обмена из формы узла

ПРИМЕЧАНИЕ

Заливкой выделены события, выполняющиеся в транзакции записи.

Глоссарий

Базовый вид расчета – вид расчета, результат которого будет использован при перерасчете данного вида расчета. Например, вид расчета *Оклад* является базовым для вида расчета *Премия*.

Базовый период расчета – период, в котором анализируются результаты других расчетов, влияющих на данный вид расчета по базовому периоду.

Быстрая пользовательская настройка – настройка, расположенная в форме отчета, которая требуется пользователю постоянно.

Ведущий вид расчета – вид расчета, изменение результатов которого приводит к необходимости перерасчета данного вида расчета. Например, виды расчета *Невыход* и *Оклад* являются ведущими для вида расчета *Премия*.

Вид расчета – алгоритм, по которому рассчитывается некоторая часть заработной платы, например *Оклад*, *Премия* и т. п.

Виды субконто – разрезы для ведения аналитического учета на счетах бухгалтерского учета.

Виртуальные таблицы – таблицы, формируемые платформой с помощью запросов из данных реальных таблиц базы данных.

Владелец – объект конфигурации, которому подчинен другой объект конфигурации. Например, справочник *Товары* является владельцем справочника *Единицы измерения*.

Временные таблицы – это программные объекты, которые разработчик может создать и заполнить данными, а запросы могут использовать данные временных таблиц для своих нужд.

Встроенный язык является важной частью технологической платформы «1С:Предприятие 8», поскольку позволяет разработчику описывать собственные алгоритмы функционирования прикладного решения.

Вытеснение по периоду действия – зависимость, которую оказывают вытесняющие виды расчета на период действия данного вида расчета.

Вытесняющий вид расчета – вид расчета, который вытесняет данный вид расчета по периоду действия. Например, вид расчета *Невыход* является вытесняющим для вида расчета *Оклад*.

Вычисляемые поля представляют собой дополнительные поля схемы

компоновки данных, значения которых будут вычисляться по некоторой формуле.

Группировка – элемент структуры отчета, служащий для вывода информации в виде обычного линейного отчета.

Движения документа – это записи в регистрах, которые создаются в процессе проведения документа и отражают изменения, производимые этим документом.

Движения регистра – набор записей, отражающий изменение состояния регистра. В каждой записи содержатся значения измерений, значения приращений ресурсов и т. п.

Дерево объектов конфигурации – иерархическая структура всех объектов конфигурации.

Детальные записи отчета – записи, получаемые в результате выполнения запроса без итогов.

Диаграмма – элемент структуры отчета, служащий для вывода информации в виде диаграммы.

Диаграмма Ганта представляет собой диаграмму интервалов на шкале

времени и отражает использование объектами (точками) ресурсов (серий).

Документ – объект конфигурации, предназначенный для описания информации о совершенных хозяйственных операциях или о событиях, произошедших в жизни организации вообще.

Дополнение периода макета компоновки данных позволяет указывать для группировок дополнение периодов с заданной периодичностью в указанном интервале, которое служит для детализации данных в отчете.

Зависимость по базовому периоду – зависимость, которую оказывают базовые виды расчета на базовый период действия данного вида расчета.

Запись XML – объект встроенного языка, обеспечивающий запись документов формата XML из встроенного языка.

Иерархия групп и элементов – вид подчинения в иерархическом справочнике, когда элемент или группа элементов справочника подчинены другой группе элементов этого справочника.

Измерения регистра – объекты конфигурации, в разрезе которых накапливается информации в регистре. Например, *Материал*, *Склад* и т. п.

Иерархия элементов – вид подчинения в иерархическом справочнике, когда один элемент подчинен другому.

Имя объекта конфигурации – уникальное наименование объекта, которое служит для обращения к свойствам и методам объекта на встроенном языке.

Информационная панель – панель в нижней части окна приложения, которая отображает информацию о последних действиях, выполненных в системе.

Клиент-серверная архитектура разделяет всю работающую систему на три различные части, определенным образом взаимодействующие между собой – Клиент, Сервер «1С:Предприятия» и Сервер баз данных.

Клиент – это пользовательская часть приложения, которую видит и с которой работает пользователь.

Ключ записи регистра сведений – совокупность значений измерений регистра и периода (в случае, если регистр сведений периодический). Регистр сведений не может содержать несколько записей с одинаковыми ключами.

Конструктор запроса – инструмент, созданный для помощи разработчику, который позволяет визуально конструировать запрос.

Конструктор форм – инструмент разработчика, построенный по принципу мастеров, для создания форм объектов конфигурации.

Конструктор печати – инструмент разработчика, построенный по принципу мастеров, для создания макетов печатных форм объектов конфигурации.

Контекст модуля определяет набор доступных во время выполнения модуля объектов, переменных, процедур и функций.

Конфигурация – совокупность созданных разработчиком объектов, их свойств, методов и алгоритмов поведения, отражающих хозяйственную деятельность предприятия. Конфигурация разрабатывается в режиме *Конфигуратор*.

Конфигурация базы данных – конфигурация, с которой работают пользователи.

Конфигурируемость системы «1С:Предприятие» – возможность настройки системы на особенности конкретного предприятия и класса решаемых задач.

Макет – объект конфигурации, предназначенный для хранения различных форм представления данных или вспомогательных данных, которые использует некоторый объект конфигурации или вся конфигурация в целом.

Модуль – «хранилище» для текста программы на встроенном языке.

Момент времени представляет собой совокупность даты, времени и ссылки на объект базы данных. Он позволяет однозначно идентифицировать любой объект ссылочного типа базы данных на оси событий, но имеет смысл в основном только для документов.

Настройки схемы компоновки данных определяют иерархическую структуру отчета (группировка, таблица, диаграмма) и его внешний вид (список полей отчета, сортировку, отбор, условное оформление записей отчета и т. п.)

Независимый регистр сведений – регистр сведений, не использующий подчинение регистратору.

Неоперативное проведение документов подразумевает отражение в базе данных фактов, которые свершились в прошлом или которые точно будут совершены в будущем. Задача неоперативного проведения документов – просто отразить в информационной базе данные о совершенных операциях.

Обработчики событий – процедуры на встроенном языке, выполняющиеся в момент наступления событий объектов конфигурации.

Объекты обмена – объекты конфигурации, данные которых должны

участвовать в обмене конкретного плана обмена.

Объектный способ доступа к данным реализован посредством использования объектов встроенного языка. Например, объект *ДокументОбъект.ОказаниеУслуги* будет содержать значения всех реквизитов документа *Оказание услуги* и всех его табличных частей.

Объекты конфигурации – логические единицы, «блоки», из которых состоит конфигурация.

Окно редактирования свойств объекта конфигурации предназначено для сложных объектов конфигурации и позволяет путем выполнения последовательных действий быстро создавать такие объекты.

Оперативная отметка времени создается системой каждый раз при оперативном проведении документа. Ее значение формируется исходя из текущего времени и последней созданной оперативной отметки.

Оперативное проведение документов пользователями выполняется в режиме «реального времени», то есть отображает изменения, факты, свершающиеся в настоящее время.

Основная конфигурация (или просто *Конфигурация*) – конфигурация,

предназначенная для разработчика. Она редактируется в конфигураторе.

Отладчик – вспомогательный инструмент, облегчающий разработку и отладку программных модулей системы 1С:Предприятие.

Палитра свойств – это специальное служебное окно, которое позволяет редактировать все свойства объекта конфигурации и другую связанную с ним информацию.

Панель действий – панель в верхней части окна приложения, которая содержит команды, соответствующие текущему разделу, выбранному в панели разделов. Эти команды объединены в стандартные группы: *Создать, Отчеты, Сервис* и группы, созданные разработчиком.

Панель навигации – панель в левой стороне окна приложения, которая отображает структуру выбранного раздела. Как правило, панель навигации предназначена для быстрого перехода к различным спискам в пределах выбранного раздела программы.

Панель разделов – панель в верхней части окна приложения, которая отражает функциональную структуру приложения и позволяет быстро переключаться между его частями.

Перерасчет – подчиненный регистру расчета объект конфигурации для регистрации фактов появления в регистре записей, влияющих на результат расчета уже существующих записей регистра.

Перечисление – объект конфигурации, предназначенный для описания структуры хранения постоянных наборов значений, не изменяемых в процессе работы конфигурации.

Период действия расчета – период, задаваемый пользователем, в котором действует результат расчета.

Периодический регистр сведений – регистр сведений, использующий привязку ко времени.

План видов характеристик – объект конфигурации, предназначенный для описания структуры хранения информации о характеристиках, создаваемых пользователем.

План счетов – объект конфигурации, предназначенный для описания структуры хранения информации о совокупности синтетических счетов предприятия, которые созданы для группировки данных о его хозяйственной деятельности.

План обмена – объект конфигурации, предназначенный для описания

участников обмена (узлов обмена) и объектов конфигурации, данные которых участвуют в обмене (объектов обмена).

Планировщик заданий – соединение, которое запускает задания по расписанию.

Платформа – базисная часть системы «1С:Предприятие», которая обеспечивает работу конфигурации и позволяет вносить в нее изменения или создавать собственную конфигурацию.

Подсистемы – объекты конфигурации, позволяющие выделить в конфигурации функциональные части, на которые логически разбивается создаваемое прикладное решение.

Подчиненные объекты конфигурации – объекты конфигурации, которые логически связаны и подчинены другому объекту конфигурации. Например, *Реквизиты*, *Формы* и т. п.

Пользовательская настройка – настройка, которую пользователь задает перед формированием отчета.

Предопределенные элементы – элементы объекта конфигурации, созданные разработчиком и не зависящие от действий пользователя.

Признак учета – подчиненный плану счетов объект конфигурации для хранения вида учета (например, количественный и валютный) на конкретном счете.

Признак учета субконто – подчиненный плану счетов объект конфигурации для хранения вида учета субконто (например, суммовой, количественный или валютный) на конкретном счете.

Прикладное решение – содержит всю функциональность, необходимую для работы предприятия. Это часть системы, видимая для конечного пользователя.

Примитивные типы данных – это *Число*, *Строка*, *Дата* и *Булево*. Примитивные типы данных изначально определены в системе, и их набор ограничен.

Проведение документа означает, что событие, которое он отражает, повлияло на состояние учета.

Рабочая область приложения – рабочее окно приложения, в котором отражаются формы списков и др. команды навигации.

Рабочий стол – раздел приложения, предназначенный для размещения наиболее часто используемых пользователем документов, отчетов, справочников и т. п.

Регистр бухгалтерии – объект конфигурации, предназначенный для описания структуры накопления бухгалтерских данных, учет которых ведется исходя из некоторого плана счетов.

Регистр накопления – объект конфигурации, предназначенный для описания структуры накопления данных.

Регистр оборотов – регистр накопления, накапливающий обороты.

Регистр расчета – объект конфигурации, предназначенный для описания структуры накопления данных, являющихся результатами сложных периодических расчетов.

Регистр сведений – объект конфигурации, предназначенный для описания структуры хранения данных в разрезе нескольких измерений.

Регистратор регистра – объект конфигурации, который может производить движения в регистре.

Редактор форм – объединяет несколько взаимосвязанных между собой окон для редактирования данных и элементов формы, команд формы, модуля формы и т. д.

Режим «1С:Предприятие» служит для работы пользователей системы. В этом режиме пользователи вносят данные, обрабатывают их и получают выходные результаты.

Режим Конфигуратор используется разработчиками для модификации существующей или создания новой конфигурации.

Реквизиты объекта конфигурации – свойства, характеризующие объект конфигурации, созданные разработчиком. Например, *Артикул*, *Производитель* и т. п.

Реквизиты регистра – набор свойств регистра для хранения дополнительной информации.

Ресурсы регистра – виды информации, накапливаемой регистром. Например, *Количество*, *Сумма* и т. п.

Ресурсы схемы компоновки данных – поля, значения которых рассчитываются на основании детальных записей, входящих в группировку. По сути, ресурсы являются групповыми или общими итогами отчета.

Родитель – элемент или группа элементов справочника, в зависимости от вида иерархии, которому подчинены другие элементы этого справочника.

Роль – объект конфигурации, предназначенный для описания прав пользователей на выполнение различных действий с той или иной информацией, хранящейся в информационной базе.

Сервер «1С:Предприятия» – это часть системы 1С:Предприятие, передающая запросы от клиентского приложения к серверу баз данных и возвращающая обратно клиенту результаты этих запросов. На сервере выполняется большинство алгоритмов на встроенном языке, подготовка данных для отображения форм, отчетов и т. д.

Сервер баз данных – это программа, поставляемая сторонними производителями. Ее основное назначение – это организация и ведение баз данных.

Синоним объекта конфигурации предназначен для хранения «альтернативного» наименования объекта конфигурации, которое будет использовано в элементах интерфейса программы.

Синтакс-помощник – инструмент, созданный для помощи разработчику, который содержит описание всех программных объектов, которые использует система, их методов, свойств, событий и пр.

Система компоновки данных – мощный и гибкий механизм для построения

отчетов, позволяющий выполнить все необходимые действия – от получения данных из различных источников до представления этих данных в виде, удобном для пользователя.

События – различные ситуации, которые возникают в процессе работы прикладного решения. События связаны с конкретными объектами конфигурации. Например, событие *ПриОткрытии* объекта конфигурации *Форма* возникает при открытии формы.

Справочник – объект конфигурации, предназначенный для работы со списками данных.

Стандартные реквизиты – свойства объекта конфигурации, автоматически созданные платформой. Например, *Код*, *Наименование* и т. п.

Субконто – конкретные объекты для ведения аналитического учета на счетах бухгалтерского учета.

Схема компоновки данных – основа для построения отчета, содержащая исходные данные для компоновки отчета.

Таблица – элемент структуры отчета, служащий для вывода информации в виде таблицы.

Табличная часть – набор информации, которая одинакова по своей структуре, но различна по количеству, и предназначена для разных элементов объекта конфигурации. Например, список мест работы в справочнике *Сотрудники*.

Табличный доступ к данным реализован посредством использования запросов к базе данных. В этой технике разработчик получает возможность оперировать отдельными полями таблиц базы данных, в которых хранятся те или иные данные.

Типообразующие объекты – объекты конфигурации, которые могут образовывать новые типы данных.

Точки останова позволяют прерывать выполнение программы во время отладки в тех местах, где они установлены.

Транзакция – это неделимая последовательность манипулирования данными, переводящая базу данных из одного целостного состояния в другое. Если по каким-то причинам одно из действий транзакции невыполнимо, база данных возвращается в то состояние, которое было до начала транзакции.

Фактический период расчета – период, получившийся из периода действия вида расчета после анализа всех периодов действия расчетов, вытесняющих данный вид расчета по периоду действия.

Функциональные опции позволяют разработчику выделить некоторую часть функциональности прикладного решения, которую можно оперативно включать или выключать на этапе внедрения и/или в процессе работы системы.

Чтение XML – объект встроенного языка, обеспечивающий чтение документов формата XML из встроенного языка.

Язык запросов – специальный язык. На нем описывается алгоритм, по которому данные будут выбраны из таблиц запроса базы данных. Этот алгоритм помещается в текст запроса.

XML-сериализация преобразует объект «1С:Предприятия» в последовательность данных, представленных в формате XML, и выполняет и обратное преобразование – преобразует последовательность данных формата XML в объект «1С:Предприятия», при условии что имеется соответствующий тип «1С:Предприятия».