МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Колледж экономики, управления и права

Методические указания для выполнения практических и самостоятельных работ по учебной дисциплине

МДК 02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Специальности

Прикладная информатика (по отраслям)

Ростов-на-Дону
2017

Методические указания по учебной дисциплине «МДК 02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности» разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.05 Прикладная информатика (по отраслям), предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных самостоятельных заданий и образцы решения задач, а также список рекомендуемой литературы.

Составитель (автор): Л.А.Шевченко, преподаватель колледжа ЭУП

Рассмотрены на заседании предметной (цикловой) комиссии специальности 09.02.05 Прикладная информатика (по отраслям)

Протокол № 1 от «31» августа 2017 г

Председатель П(Ц)К специальности _____ Л.А.Шевченко
личная подпись

и одобрены решением учебно-методического совета колледжа.

Протокол № 1 от «13» сентября 2017 г

Председатель учебно-методического совета колледжа
_____ С.В.Шинакова
личная подпись

Рекомендованы к практическому применению в образовательном процессе.

Рецензенты:

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методические указания по «МДК02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности» разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.05 Прикладная информатика (по отраслям), предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

В предлагаемом пособии на практическом примере рассмотрены основные особенности разработки программ в режиме управляемого приложения. Помимо описания особенностей работы с такими объектами, как подсистемы, константы, справочники, документы, перечисления, последовательности, нумераторы, регистры накопления, регистры сведений, в пособии уделено внимание организации клиент-серверного взаимодействия. Кроме того, здесь рассматриваются методы конструирования командного интерфейса управляемого приложения и методы работы с управляемыми формами.

Практическая работа№ 1 Формирование информационной базы данных (ИБД)

Цель: научиться конвертации и созданию баз данных, разработанных для 1С:Предприятие 8.1., узнать об особенностях установки и запуска 1С:Предприятие 8.2., данных об управлении информационными базами.

ХОД РАБОТЫ

1. Выполним запуск системы, настройки информационных баз

1.1 Итак, у нас имеется информационная база, которая была рассчитана на работу в 1С:Предприятие 8.1. Она расположена по адресу **C:\Salon**. Подключим эту базу для использования в 1С:Предприятие 8.2.

1.2 Сделаем двойной щелчок по значку 1С Предприятие на Рабочем столе или воспользуемся командой Пуск > Все программы > 1С Предприятие 8.2 > 1С:Предприятие. В появившемся окне интерактивной программы запуска, (см. Рисунок 1.2., уже рассмотренный ранее), нажмем на кнопку **Добавить.**

1.3 Появится окно, которое устроено по принципу пошагового мастера, задающего вопросы пользователю для достижения некоторой цели. Первым вопросом здесь будет выбор между созданием новой информационной базы и добавлением существующей, Рисунок 1.1.

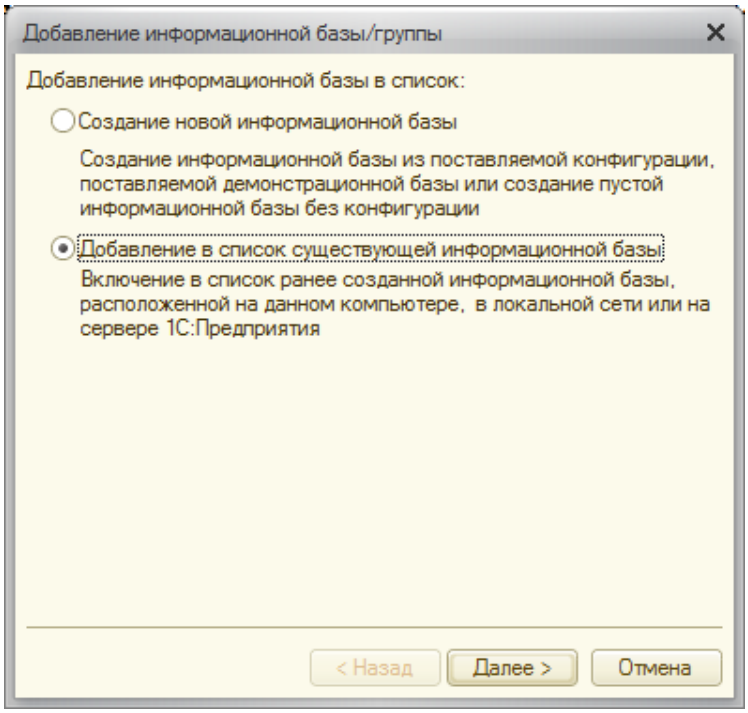


Рисунок 1.1- Добавление новой информационной базы

1.4 Мы собираемся добавить в окно запуска уже существующую информационную базу, выберем соответствующий вариант и нажмем **Далее**. Если нам нужно будет создать новую информационную базу и мы выберем вариант **Создание новой информационной базы**, мы сможем пойти одним из двух путей – либо создать новую пустую базу для разработки собственной конфигурации, либо создать базу из шаблона конфигурации.

1.5 Следующее окно, позволяет задать название и тип расположения базы – введем в поле названия **База салона, 8.1.**, в типе расположения оставим значение по умолчанию –**На данном компьютере или на компьютере в локальной сети**.

1.6 Очередное нажатие на кнопку **Далее** приводит нас к окну, где нужно указать путь к информационной базе. Нас интересует папка, где расположен файл **1Cv8.1CD**, в нашем случае это папка **C:\Salon**.

1.7 Нажав еще раз на кнопку Далее мы переходим к очень важному этапу настройки информационной базы, Рисунок 1.2..

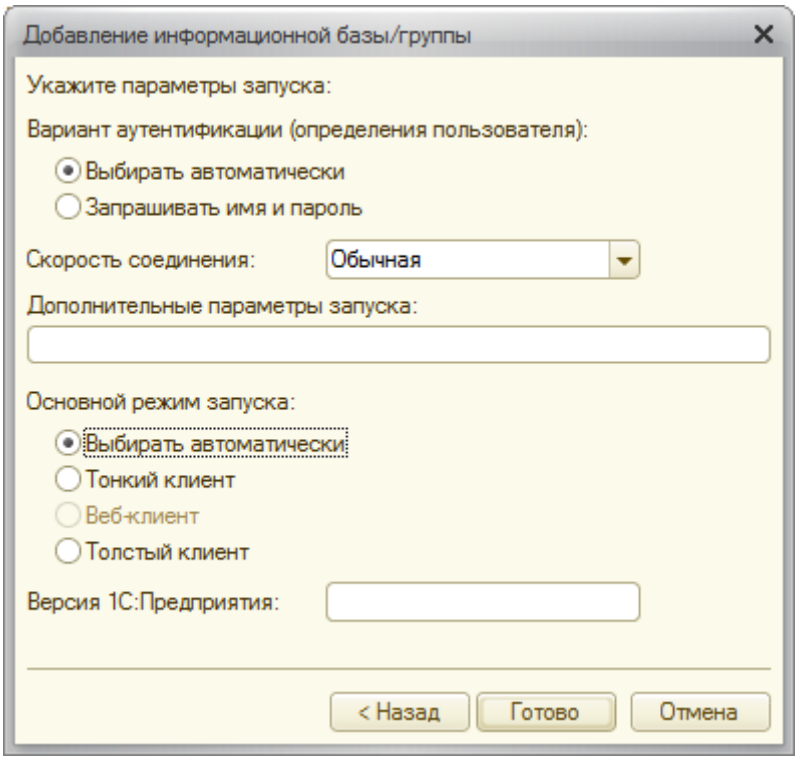


Рисунок 1.2- Настройка параметров запуска информационной базы

Здесь можно, в частности, указать основной режим запуска базы и версию 1С:Предприятия, необходимую для данной базы, введя ее в виде обычного текста в соответствующее поле, задав дополнительные параметры запуска, при необходимости – указав скорость соединения – это актуально для работы в режиме тонкого клиента через Интернет.

Мы оставим здесь параметры, установленные по умолчанию и нажмем на кнопку **Готово**. После этого информационная база появится в списке информационных баз, Рисунок 1.3.

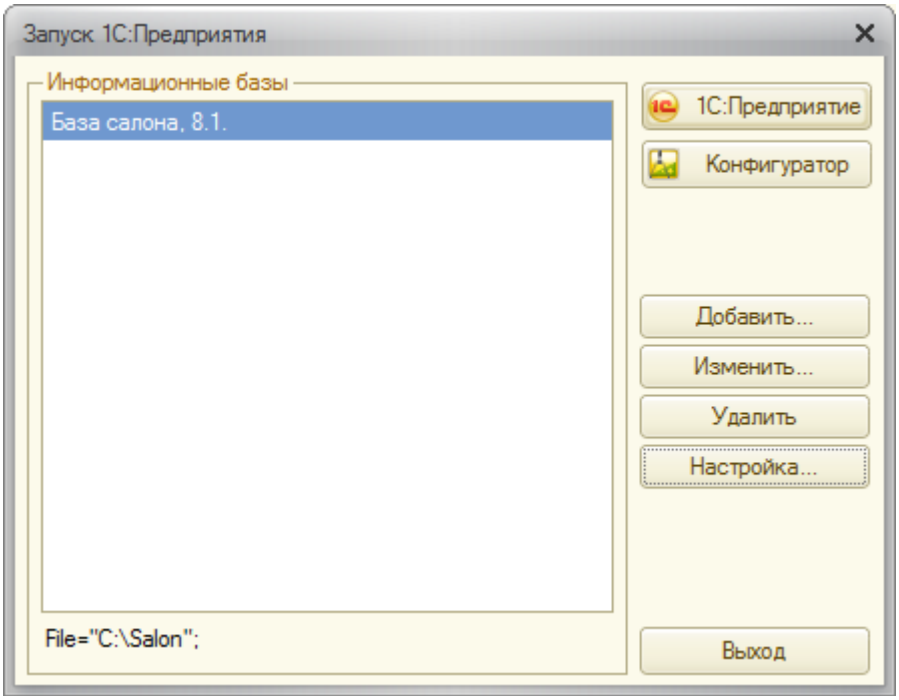


Рисунок 1.3- Информационная база в списке информационных баз

1.8 Нажимаем на кнопку **Конфигуратор**. Нажатие на кнопку **1С:Предприятие** приведет к запуску информационной базы в режиме 1С:Предприятие – в режиме, в котором работают пользователи. Причем, выбор приложения (версии и вида) будет зависеть как от настроек информационной базы, произведенных при ее добавлении или при изменении ее параметров, так и от настроек пользователя, под которым будет осуществлен вход в информационную базу. В одной из следующих лекций мы рассмотрим настройки пользователей, влияющие на выбор приложения для запуска информационной базы. Если мы сразу после добавления в список информационной базы, которая рассчитана на 1С:Предприятие 8.1., попытаемся запустить ее в режиме **1С:Предприятие**, мы получим сообщение об ошибке – база еще не преобразована для использования ее с версией 8.2.

Кнопка **Конфигуратор** предназначена для открытия базы в режиме конфигурирования – основном режиме, которым пользуются разработчики. В нашем случае нажатие на эту кнопку приведет к появлению сообщения о том, что для работы с базой ее нужно преобразовать. Нажмем **Да** в этом окне, после чего появится еще одно окно сообщения, говорящее о несоответствии формата файла формату 1С:Предприятие и предлагающему начать преобразование. Нажмем в этом окне кнопку **Да**. Наша база содержит список пользователей с различными правами на работу с базой – для продолжения нам придется войти в систему под учетной записью с именем **Администратор**, без пароля, Рисунок 1.6.

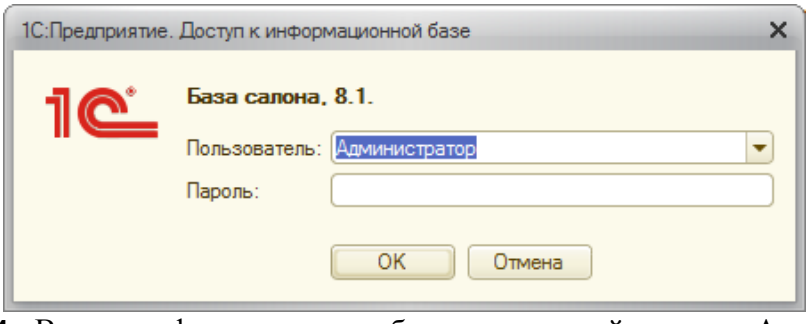


Рисунок 1.4- Вход в информационную базу под учетной записью Администратор

Нажав на **ОК**, получаем очередное окно сообщения с предложением произвести конвертацию информационной базы. Отвечаем утвердительно и через несколько секунд видим сообщение о том, что конвертация информационной базы завершена. Нажатие кнопки **ОК** в этом окне приведет к открытию окна **Конфигуратора**. Сразу же выполним в этом окне команду меню **Конфигурация > Открыть конфигурацию**, эта команда приведет к открытию дерева конфигурации, окно которого расположено в левой части экрана, Рисунок 1.5.

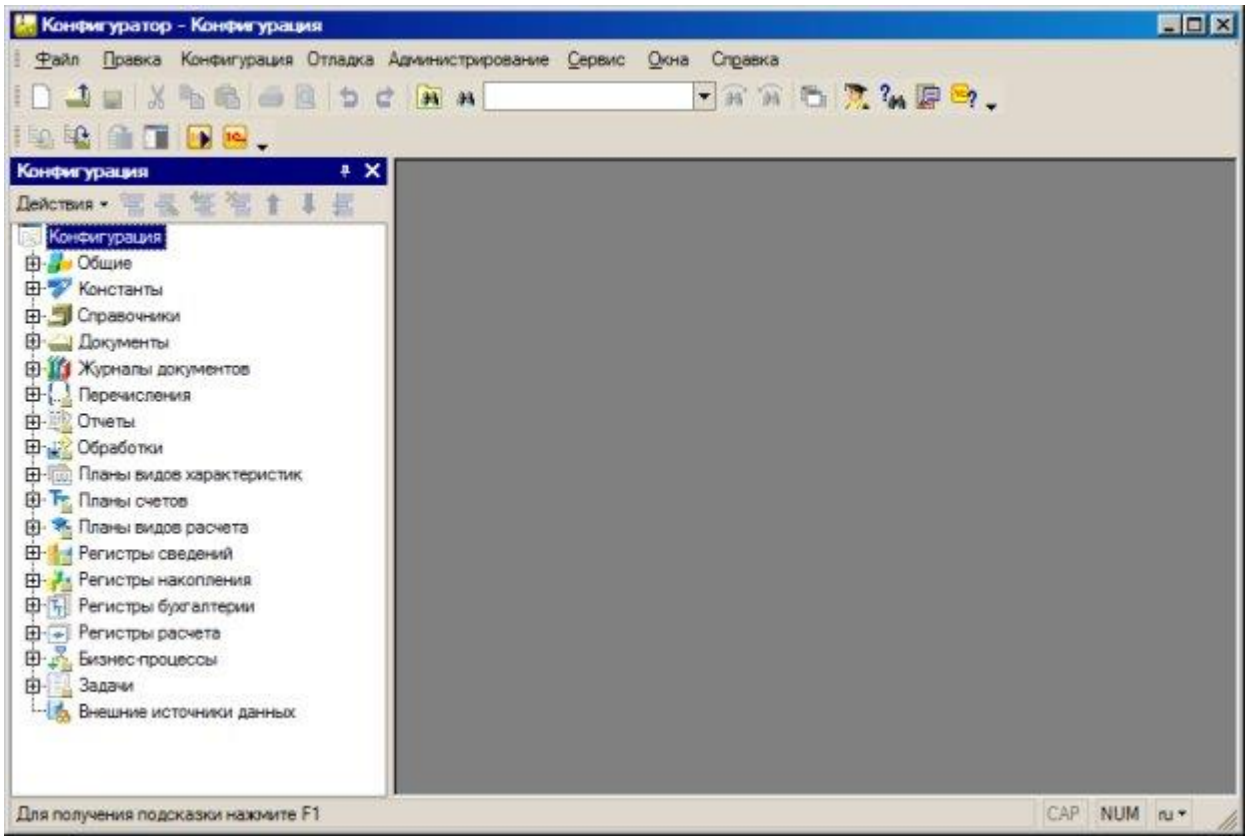


Рисунок 1.5- Информационная база открыта в Конфигураторе

1.9 Нажатием кнопки **1С:Предприятие**, или выполнить в Конфигураторе команду **Сервис > 1С:Предприятие** (так же можно воспользоваться клавиатурным сокращением Ctrl+F5 или нажать кнопку **1С:Предприятие** на панели инструментов **Конфигурация**). На данном этапе работы Конфигуратор нам нужен был лишь для преобразования информационной базы, поэтому мы можем закрыть его окно и запустить нашу информационную базу из окна запуска После запуска информационной базы в режиме **1С:Предприятие** она сохраняет ранее существующую функциональность, то есть, позволяет пользователям продолжать работу. Единственное изменение, которое могут заметить пользователи – это новый значок логотипа 1С:Предприятие в левом верхнем углу окна программы, Рисунок 1.6.

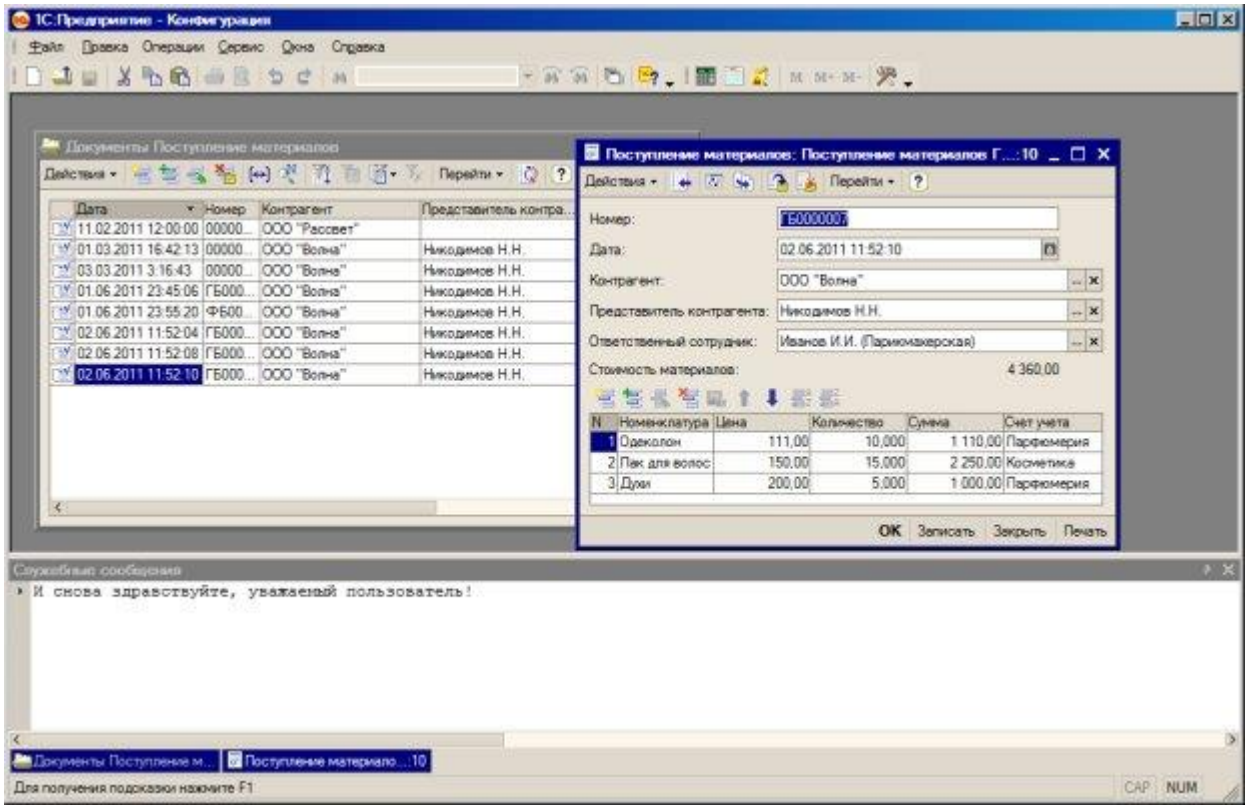


Рисунок 1.6- Сконвертированная информационная база в режиме 1С:Предприятие

Займемся разработкой новой конфигурации для режима "Управляемое приложение"

2. Выполним создание новой информационной базы

2.1 Откроем окно запуска 1С:Предприятие, создадим новую пустую информационную базу. Для этого нажмем на кнопку **Добавить**, в появившемся окне выберем Создание новой информационной базы, в следующем окне, Рисунок 1.7., выберем вариант создания информационной базы без конфигурации.

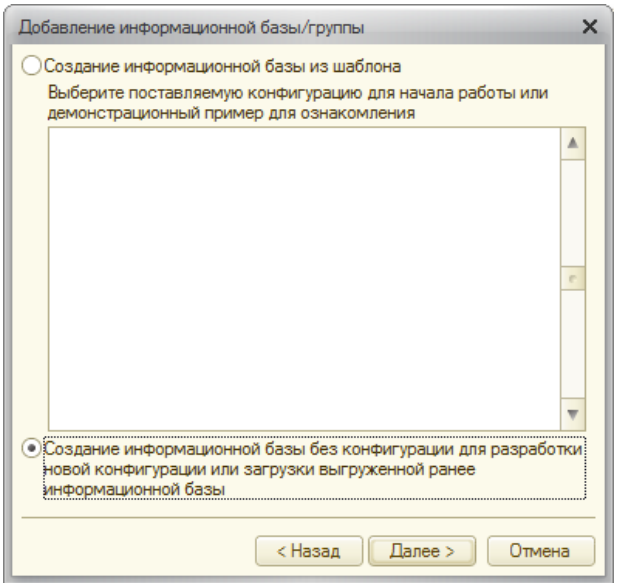


Рисунок 1.7- Создание информационной базы без конфигурации

2.2 Дадим информационной базе имя **Салон, управляемое приложение**, зададим в качестве папки информационной базы **C:\Salon2**, остальные параметры оставим по умолчанию.

2.3 После того, как база будет создана, откроем ее в **Конфигураторе** и, для того, чтобы открыть дерево конфигурации, выполним команду **Конфигурация > Открыть конфигурацию.** Вызовем контекстное меню корневого элемента Конфигурация, выберем в нем пункт **Свойства**, Рисунок 1.8.

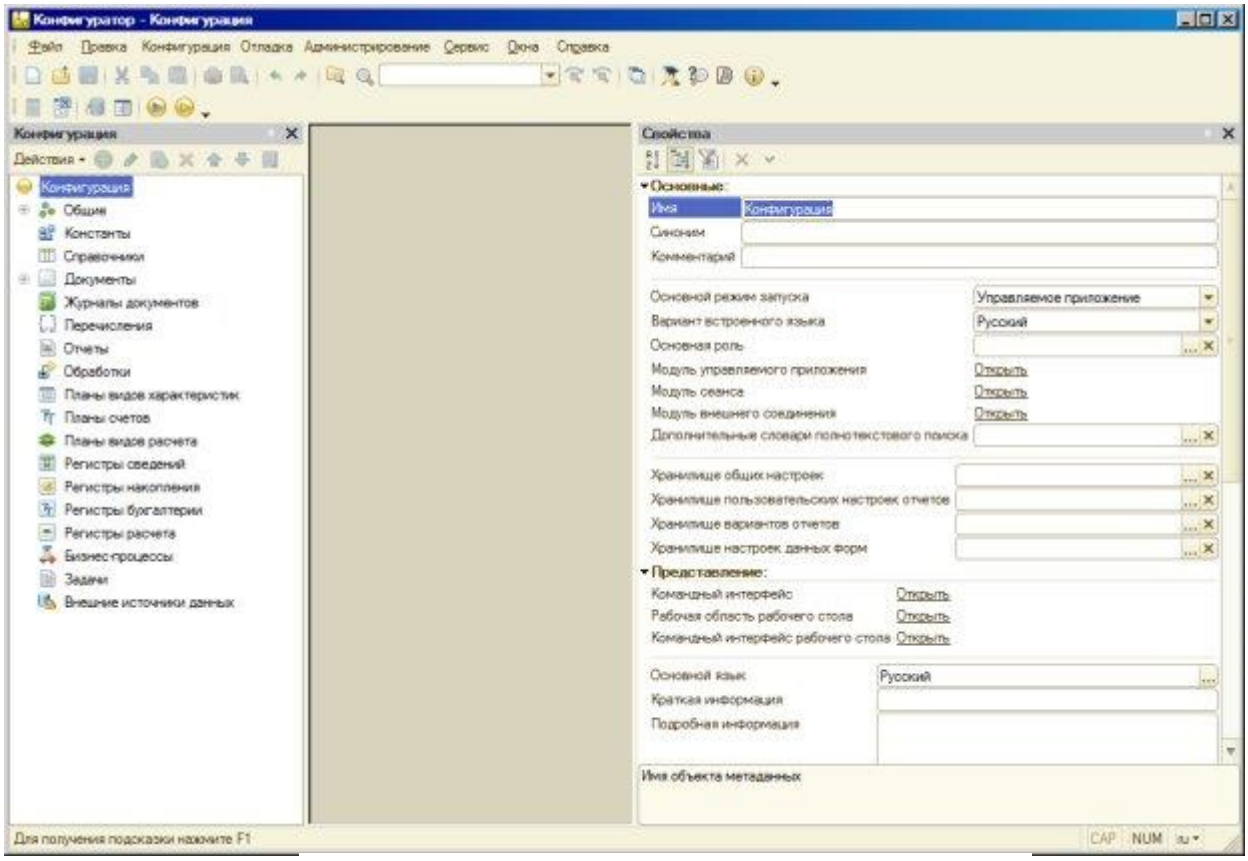


Рисунок 1.8- Свойства новой информационной базы

Обратите внимание на то, что свойство **Основной режим запуска** установлено в значение **Управляемое приложение**, в нижней части окна свойств расположено свойство**Режим совместимости**, которое установлено в значение **Не использовать**. В данном случае оно может принимать значения Версия 8.1. и Версия 8.2.13.

В качестве имени конфигурации введем **СалонКрасоты**, поле **Синоним** будет автоматически заполнено текстом **Салон красоты.** ИБД создана.

2.4 В окне дерева конфигурации представлены различные объекты. Кратко перечислим их. Можно заметить, что изменились изображения интерфейсных элементов в **Конфигураторе.** Все говорит нам о том, что сейчас мы занимаемся разработкой конфигурации в режиме управляемого приложения. Среди нововведений платформы 8.2. можно отметить изменение состава объектов конфигурации. В частности, появились следующие новые объекты:

Общие реквизиты: здесь содержатся реквизиты, которые могут использоваться во многих объектах конфигурации. Например, если вы планируете добавить в документы своей конфигурации одинаковый реквизит, содержащий наименование организации, от имени

которой составлен документ, это вполне логично реализовать с помощью общего реквизита. Кроме того, общие реквизиты используются в механизме разделения данных.

Функциональные опции: их используют для того, чтобы описывать возможности, которые можно включать и отключать в процессе эксплуатации системы. Функциональные опции могут влиять на командный интерфейс, например, скрывая или отображая некоторые группы команд, а так же – на алгоритмы, написанные на встроенном языке.

Параметры функциональных опций: Содержит параметры, влияющие на функциональные опции

Хранилища настроек: Используется для сохранения и загрузки настроек.

Общие команды: Позволяет создавать команды, которые можно использовать в других объектах конфигурации, вызывая их, например, с помощью кнопок на формах.

Группы команд: Позволяет создавать группы для объединения команд

Элементы стиля: Позволяет создавать элементы стиля, такие, как цвет, шрифт, рамка, для организации единообразного оформления других объектов.

Внешние источники данных: эти объекты используются для получения информации из внешних источников и последующего использования ее в системе, в частности, в качестве источников данных для запросов, в качестве типов реквизитов информационной базы и так далее.

.

Практическая работа№ 2 Создание объектов конфигурации справочников, автозаполнение

Цель: изучить интерфейс 1С и научиться настраивать его.

ХОД РАБОТЫ

1. Создадим подсистемы – основу командного интерфейса управляемого приложения

1.1 Чтобы просмотреть подсистемы нужно открыть нашу конфигурацию и посмотреть ветвь дерева конфигурации **Общие > Подсистемы**, Рисунок 2.1. Точно так же можно будет обращаться к конфигурации в дальнейшем.

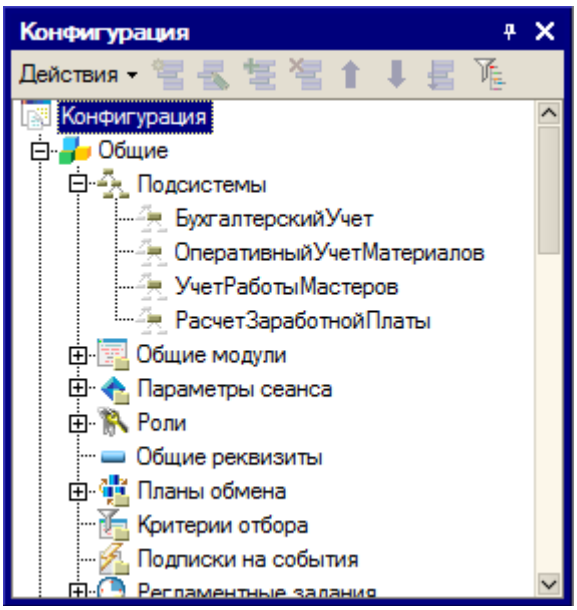


Рисунок 2.1- Подсистемы в информационной базе

Итак, в нашей старой конфигурации должны быть созданы подсистемы:

- «Бухгалтерский учет»;
- «Оперативный учет материалов»;
- «Учет работы мастеров»;
- «Расчет заработной платы».

1.2 Создадим те же подсистемы в новой конфигурации. Для создания новой подсистемы нужно перейти в ветвь дерева конфигурации **Общие > Подсистемы**, после чего либо выбрать команду **Добавить** из контекстного меню ветви **Подсистемы**, либо выделить эту ветвь и нажать клавишу **Ins** на клавиатуре, либо воспользоваться кнопкой **Добавить** из командной панели дерева конфигурации. После этого появится окно редактирования объекта конфигурации, приведенное на Рисунок 2.2.

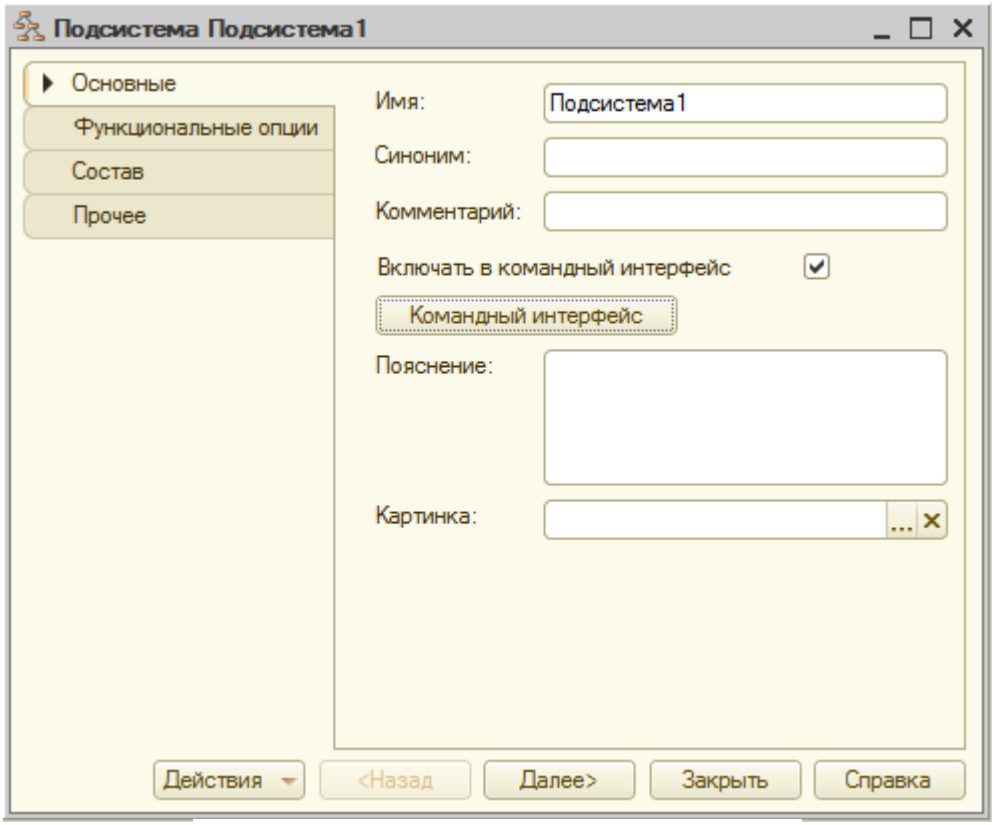


Рисунок 2.2- Окно редактирования объекта

Здесь можно либо перемещаться по вкладкам окна в произвольном порядке, либо, используя кнопку **Далее**, перемещаться по ним последовательно.

1.3 Зададим следующие параметры для нашей новой подсистемы:

Имя: БухгалтерскийУчет

Синоним: Бухгалтерский учет

Синоним генерируется автоматически на основе имени, при необходимости его можно отредактировать вручную.

Поле **Картинка** можно использовать для того, чтобы задать подсистеме заранее созданную картинку. Это позволяет сделать интерфейс пользователя более удобным.

1.4 После того, как подсистема создана, посмотрим, на что будет похожа разрабатываемая конфигурация в режиме 1С:Предприятие. Запустим ее в этом режиме из Конфигуратора, воспользовавшись комбинацией клавиш **Ctrl+F5**, соответствующей командой меню (**Сервис > 1С:Предприятие**), или кнопкой на панели инструментов**Конфигурация**.

То, что мы увидим после запуска конфигурации, разительно отличается от того, что мы привыкли видеть, Рисунок 2.3.

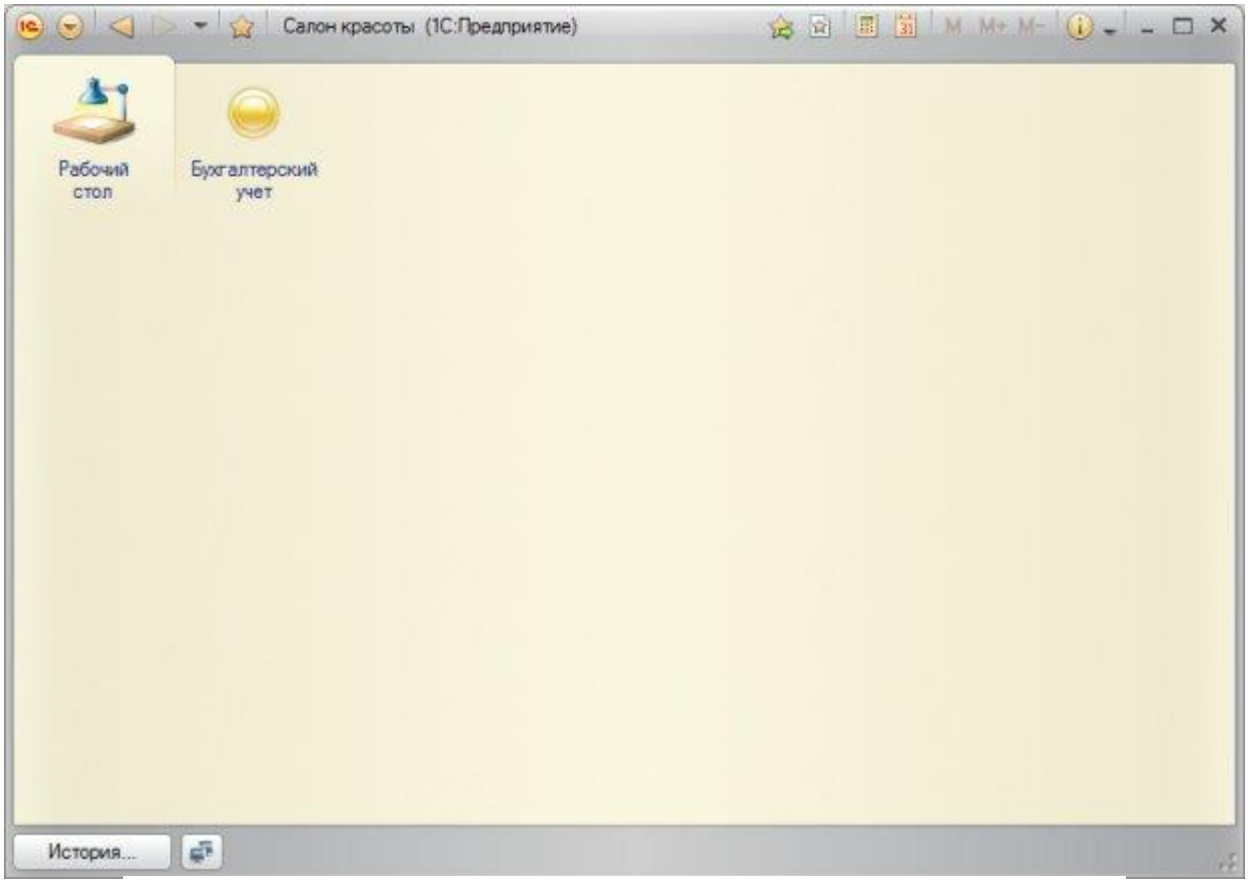


Рисунок 2.3- Разрабатываемая конфигурация в режиме 1С:Предприятие

Рабочий стол нужен для ускорения доступа пользователя к наиболее часто используемым объектам системы. Это – одна из закладок командного интерфейса, которая появляется первой при открытии конфигурации в пользовательском режиме.

Наша подсистема видна в верхней части окна программы, в так называемой панели разделов. Она снабжена стандартным рисунком, назначаемым автоматически, подпись соответствует синониму. Щелчок по вкладке "**Бухгалтерский учет**" приведет нас к командам по работе с объектами конфигурации, которые включены в эту подсистему.

1.5 Обратите ваше внимание на кнопку **Главное меню**. Она открывает меню, содержащее стандартные для Windows-программ команды, Рисунок 2.4.

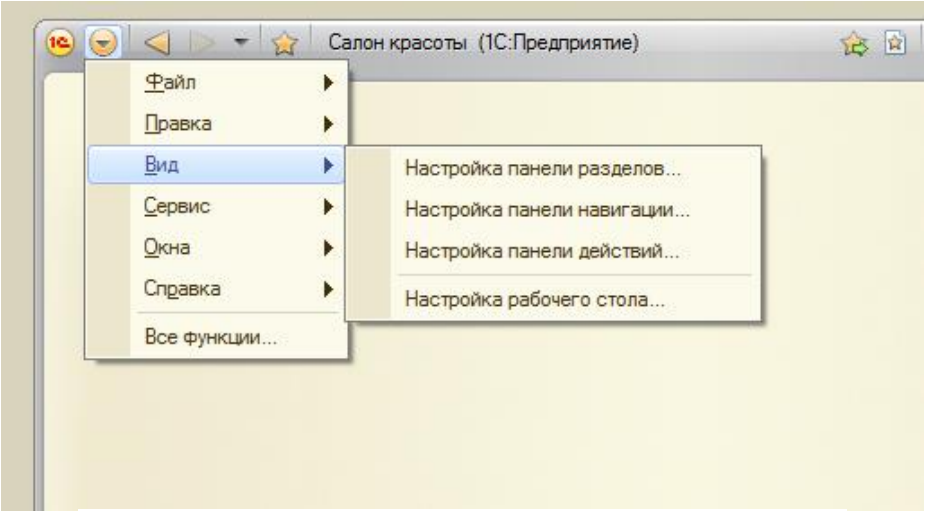


Рисунок 2.4 - Главное меню в режиме 1С:Предприятие

В сравнении с 1С:Предприятие 8.1. в составе разделов этого меню многое поменялось (в особенности это касается разделов **Вид, Сервис**). В частности, обратите внимание на команду **Главное меню > Все функции**. Эта команда, Рисунок 2.5., открывает доступ к дереву объектов конфигурации, позволяет использовать некоторые стандартные команды.

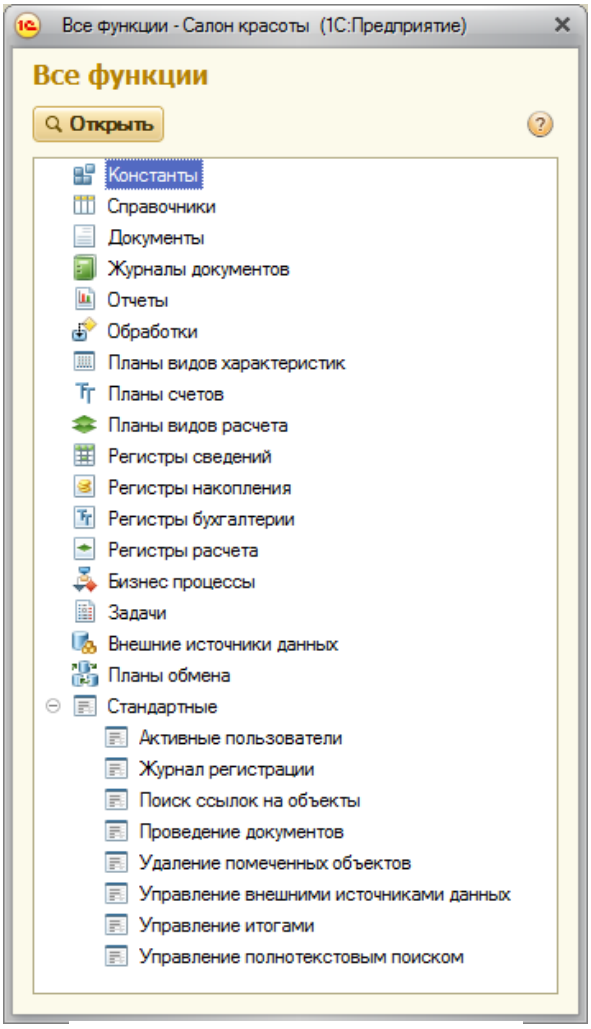


Рисунок 2.5- Окно Все функции

1.6 Вернемся в Конфигуратор, добавим к списку подсистем, которые следует создать, еще одну «Администрирование»

Особенно это окно полезно при разработке и отладке конфигурации – для быстрого поиска необходимых объектов без использования основного пользовательского интерфейса, для выполнения административных функций (таких, как удаление помеченных объектов, просмотр журнала регистрации). В законченной конфигурации есть смысл создать отдельную подсистему, которая будет содержать набор команд для вызова административных функций.

Теперь создадим полный набор подсистем, Рисунок 2.6.

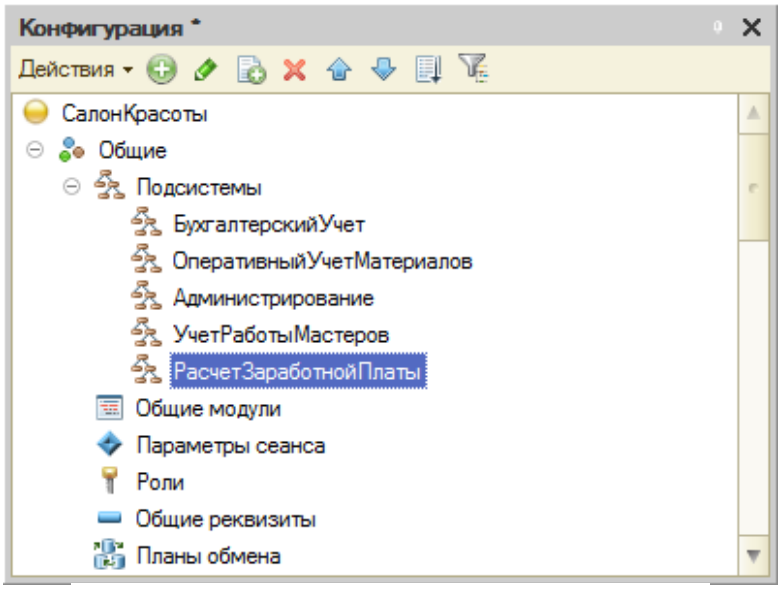


Рисунок 2.6- Набор подсистем конфигурации

Снова откроем конфигурацию в режиме 1С:Предприятие, Рисунок 2.7.

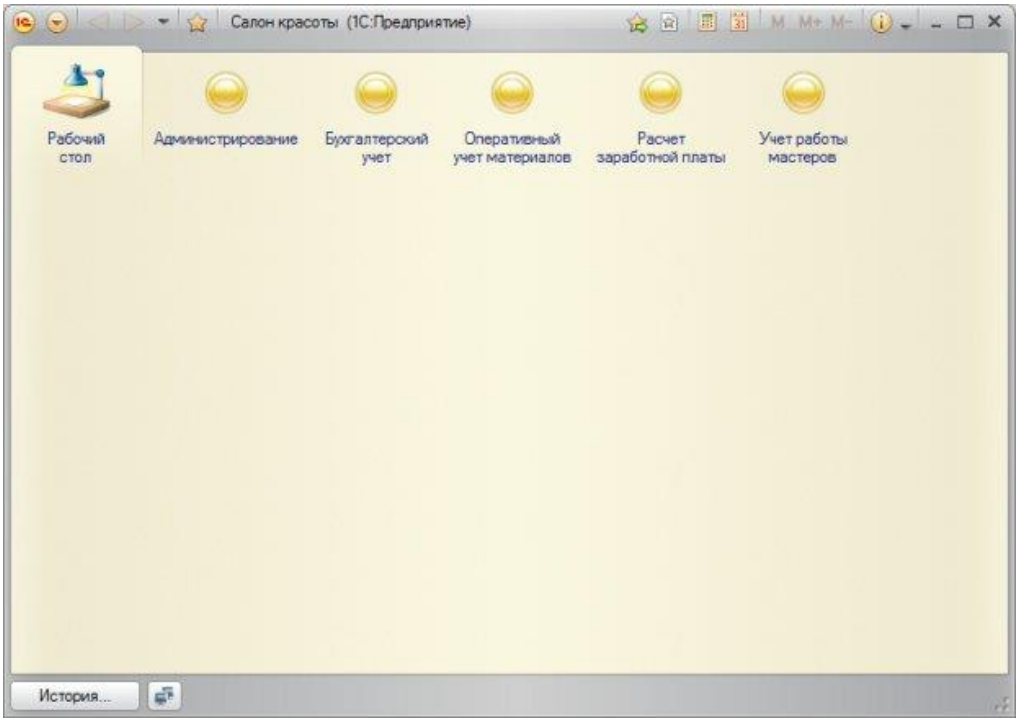


Рисунок 2.7- Панель разделов после добавления подсистем

1.7 Изменим порядок следования подсистем. Логично было бы расположить разделы нашего прикладного решения таким образом, чтобы раздел **Администрирование** оказался в правой части панели. Обычно наиболее часто используемые команды располагают левее и выше других. Можно заметить, что порядок расположения разделов не соответствует порядку расположения объектов **Подсистема** в дереве конфигурации (обратитесь к двум предыдущим рисункам для того, чтобы это увидеть). Для того чтобы изменить порядок следования подсистем в панели разделов нужно воспользоваться командой контекстного меню корневого объекта дерева конфигурации **Открыть командный интерфейс конфигурации**, Рисунок 2.8.

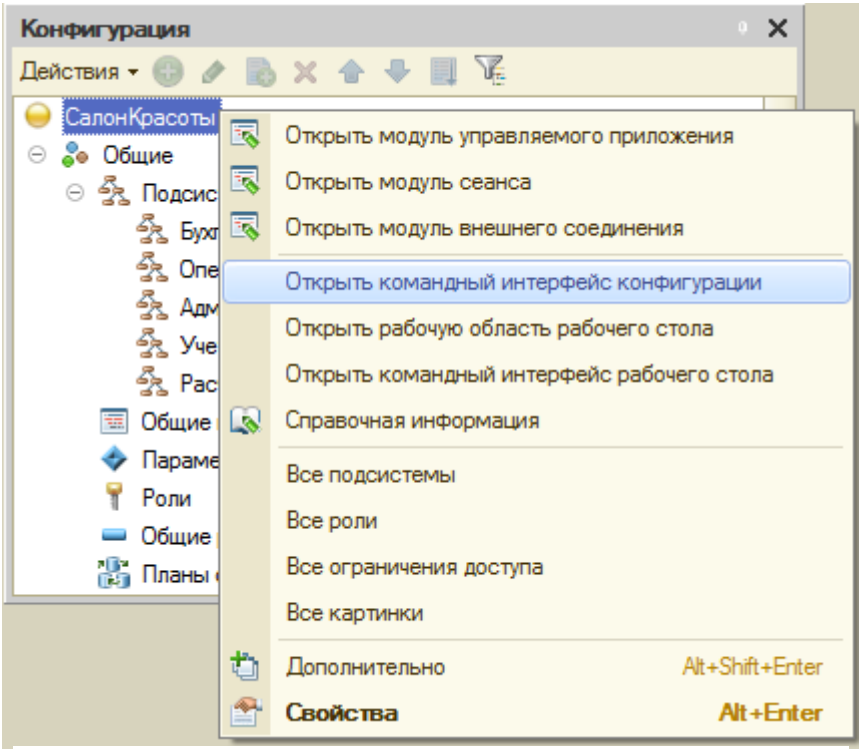


Рисунок 2.8- Открыть командный интерфейс конфигурации

В появившемся окне мы можем управлять порядком следования подсистем на панели разделов и их видимостью. Еще одной полезной возможностью настройки видимости подсистем является видимость по ролям. С помощью этого механизма можно конструировать интерфейсы для отдельных ролей, которые можно назначать пользователям, формируя, таким образом, рабочую среду, которая не содержит ничего лишнего. Настроим порядок следования подсистем с помощью кнопок **Переместить вверх** и **Переместить вниз** так, чтобы они приняли вид, представленный на рисунке 2.9.

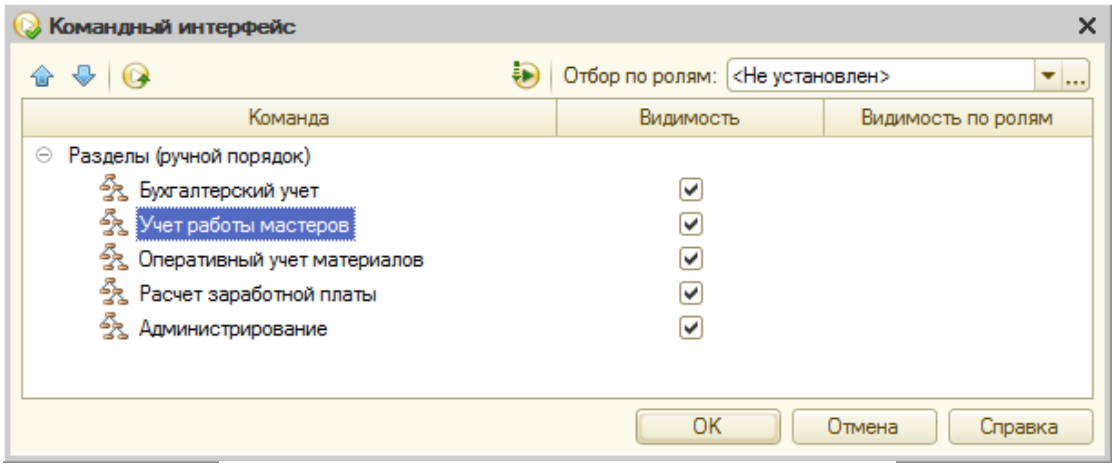


Рисунок 2.9- Настройка командного интерфейса

1.8 После нажатия на кнопку **ОК** и запуска конфигурации в пользовательском режиме, внесенные изменения можно будет наблюдать на панели разделов.

2. Рассмотрим теперь настройку видимости разделов по ролям.

2.1 Для демонстрации настройки видимости подсистем по ролям нам понадобятся два пользователя и две роли. Сначала создадим две роли – **Администратор** и **Сотрудник**. В дереве конфигурации перейдем в ветвь **Общие > Роли**, создадим новую роль, дадим ей имя **Администратор**.

Отметим права на доступ ко всем объектам (можно выполнить команду **Действия > Установить все права**), установим флажок **Устанавливать права для новых объектов**, Рисунок 2.10 Тем самым, при создании новых объектов, права на выполнение различных действий с ними будут добавляться к роли автоматически.

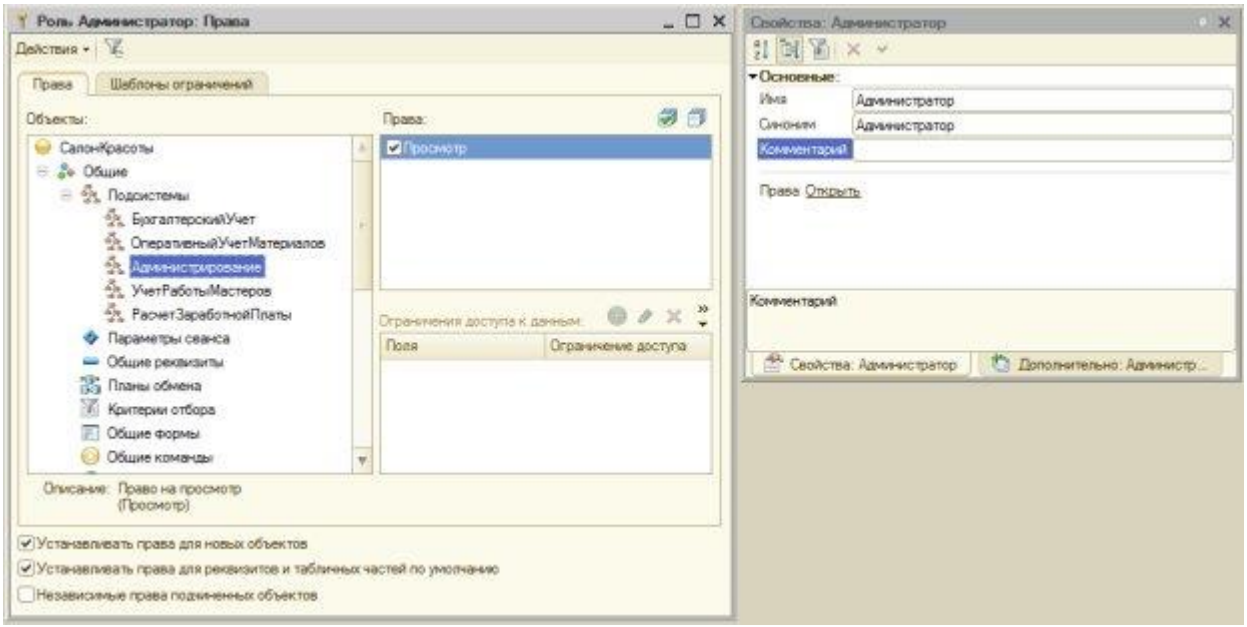


Рисунок 2.10- Настройка роли Администратор

2.2 Вторая роль, которая нас сейчас интересует, отличается от только что созданной названием – назовем ее **Сотрудник,** и тем, что у нее отключено право **Администрирование** у корневого объекта конфигурации. В свое время мы уделим настройке прав доступа к объектам больше

внимания, сейчас сосредоточимся на настройках видимости закладок панели разделов.

Выполним уже знакомую вам команду контекстного меню корневого элемента дерева конфигурации **Открыть командный интерфейс конфигурации**, в окне **Командный интерфейс** снимем флаг напротив раздела **Администрирование** у роли **Сотрудник**, остальные флаги должны быть установлены, Рисунок 2.11.

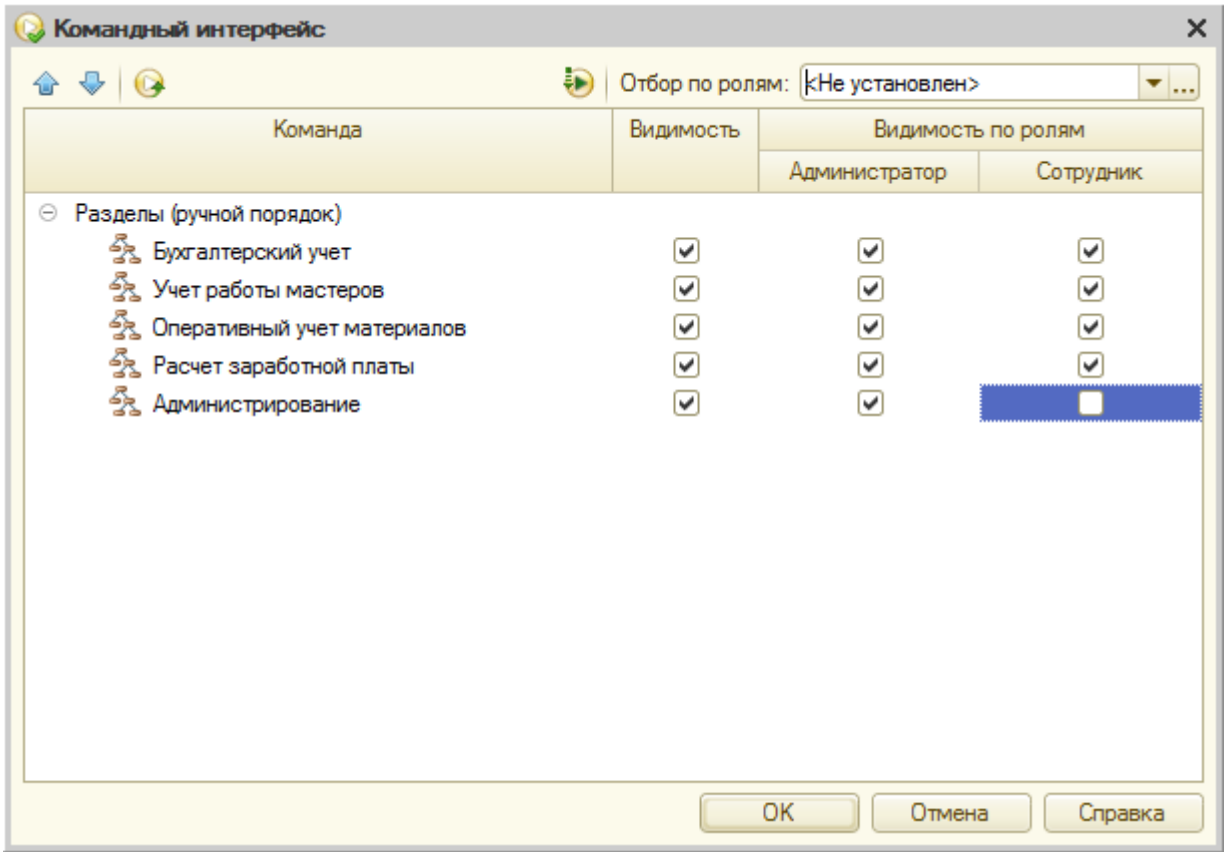


Рисунок 2.11- Настройка видимости разделов по ролям

2.3 Создадим двух пользователей конфигурации. В Конфигураторе выполним команду **Администрирование > Пользователи**, в появившемся окне **Список пользователей** нажмем на кнопку **Добавить**. На вкладке **Основные** окна **Пользователь** дадим первому пользователю имя **Администратор**, реквизит **Полное имя** будет заполнен автоматически.

Остальные параметры оставим в значении по умолчанию, что приведет к тому, что у пользователя будет пустой пароль и он будет отображаться в списке выбора пользователей (Рисунок 2.12.), хотя, в целях безопасности, пользователя с административными правами можно скрыть из списка выбора, да и имя "Администратор" лучше заменить на что-нибудь менее очевидное.

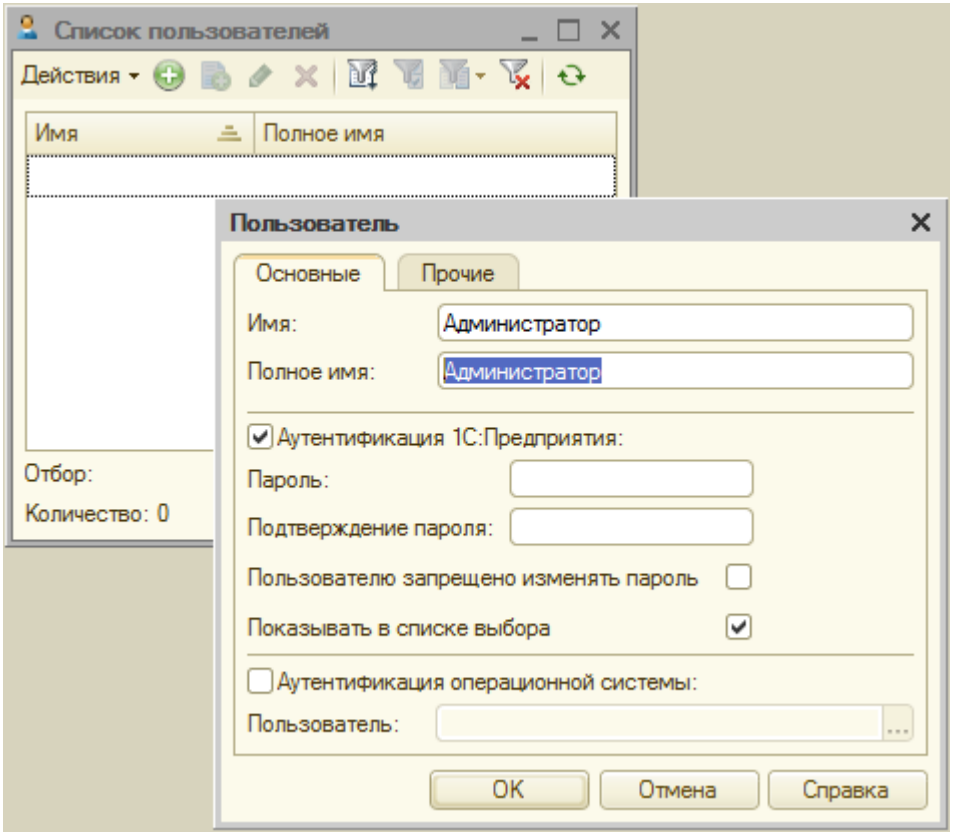


Рисунок 2.12- Создание нового пользователя

Перейдем на вкладку **Прочие** и в списке **Доступные роли** установим роль **Администратор**, Рисунок 2.13.

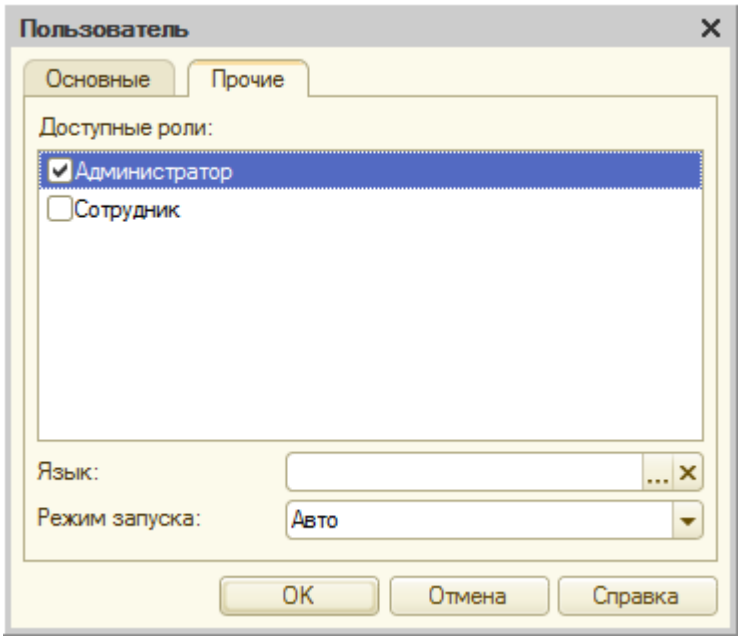


Рисунок 2.13- Настройка доступных ролей

2.4 Добавим в список второго пользователя, дадим ему имя **Директор**, на закладке **Прочие** установим среди доступных ролей роль **Сотрудник**. В итоге окно **Пользователи** примет такой вид, Рисунок 2.14.

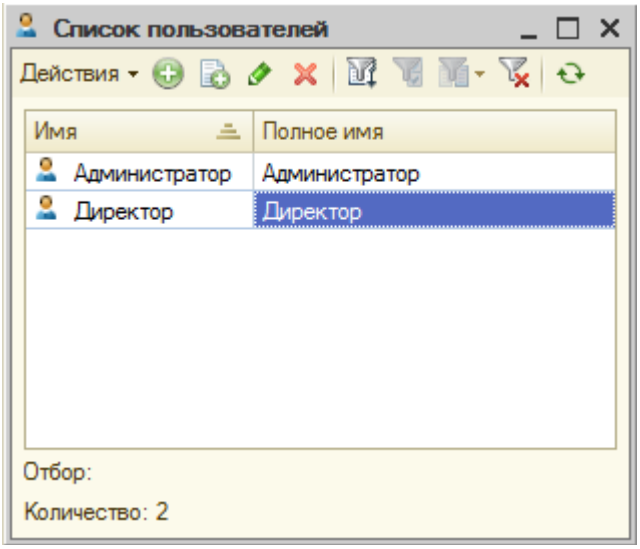


Рисунок 2.14- Список пользователей

2.5 Теперь внесем в нашу конфигурацию еще одно изменение. Добавим в ветвь **Справочники** новый справочник, назовем его **Сотрудники** (реквизит Имя на вкладке **Основные**) и добавим во все подсистемы, Рисунок 2.15.

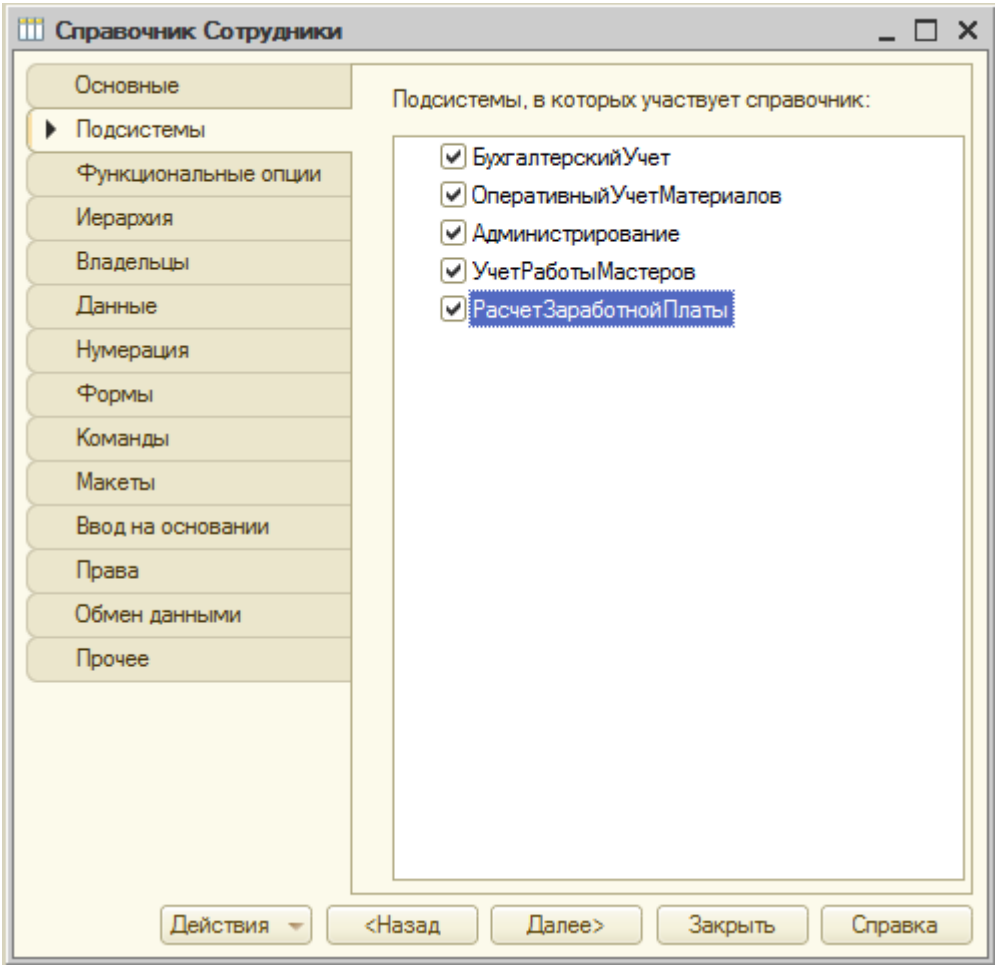


Рисунок 2.15- Новый справочник, добавленный во все подсистемы

2.6 Проверим результаты нашей работы в режиме 1С:Предприятие. После запуска

конфигурации появится окно выбора пользователя, Рисунок 2.16.

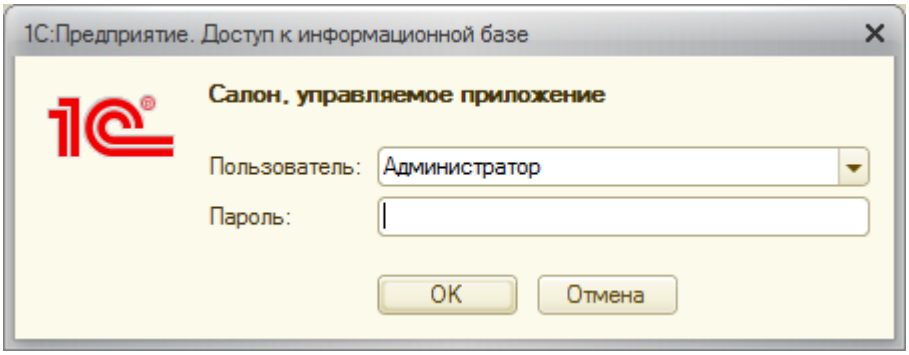


Рисунок 2.16- Окно выбора пользователя

Выбрав в списке пользователей пользователя **Администратора** отметим, что панель разделов в его командном интерфейсе имеет закладки для каждой из подсистем. Выбрав пользователя **Директор**, можно заметить, что его панель разделов не содержит раздела **Администрирование**.

Вывод

В этой работе мы рассмотрели особенности установки 1С:Предприятие 8.2. – теперь на одном компьютере могут существовать различные версии системы, всеми ими можно пользоваться. Так же мы рассмотрели конвертацию информационных баз, разработанных для 1С:Предприятие 8.1., в формат, подходящий для работы в 1С:Предприятие 8.2. Мы ознакомились с процессом создания новой информационной базы, приступили к разработке прикладного решения, создав подсистемы, которые являются основой командного интерфейса, и настроив видимость разделов интерфейса по ролям для различных пользователей, воспользовавшись объектами Роль и Пользователь.

Практическая работа №3 Создание объектов конфигурации, приходных документов

Цель: научиться конструированию справочников, разработке управляемых форм и выполнению различных действий в клиентских методах, в частности, методике вывода сообщений об ошибках в привязке к элементам управления

.

ХОД РАБОТЫ

1. Создадим справочник «Организации»

Создание выполняем через кнопку добавить на дереве конфигурации, когда выделен объект «Справочники». Справочник можно сравнить с картотекой, с неким списком данных, каждая запись которого имеет определенную структуру. В организации – независимо от того, автоматизирован ли в ней учет или нет, присутствует множество таких списков. Это – списки сотрудников, клиентов, товаров.

Справочник **Организации** нужен для хранения списка организаций, по которым планируется вести учет. Справочник, сразу после его создания, имеет некоторые стандартные реквизиты. Это утверждение справедливо и для других объектов конфигурации. Для управления реквизитами объекта служит закладка **Данные** окна редактирования объекта, Рисунок 3.1.

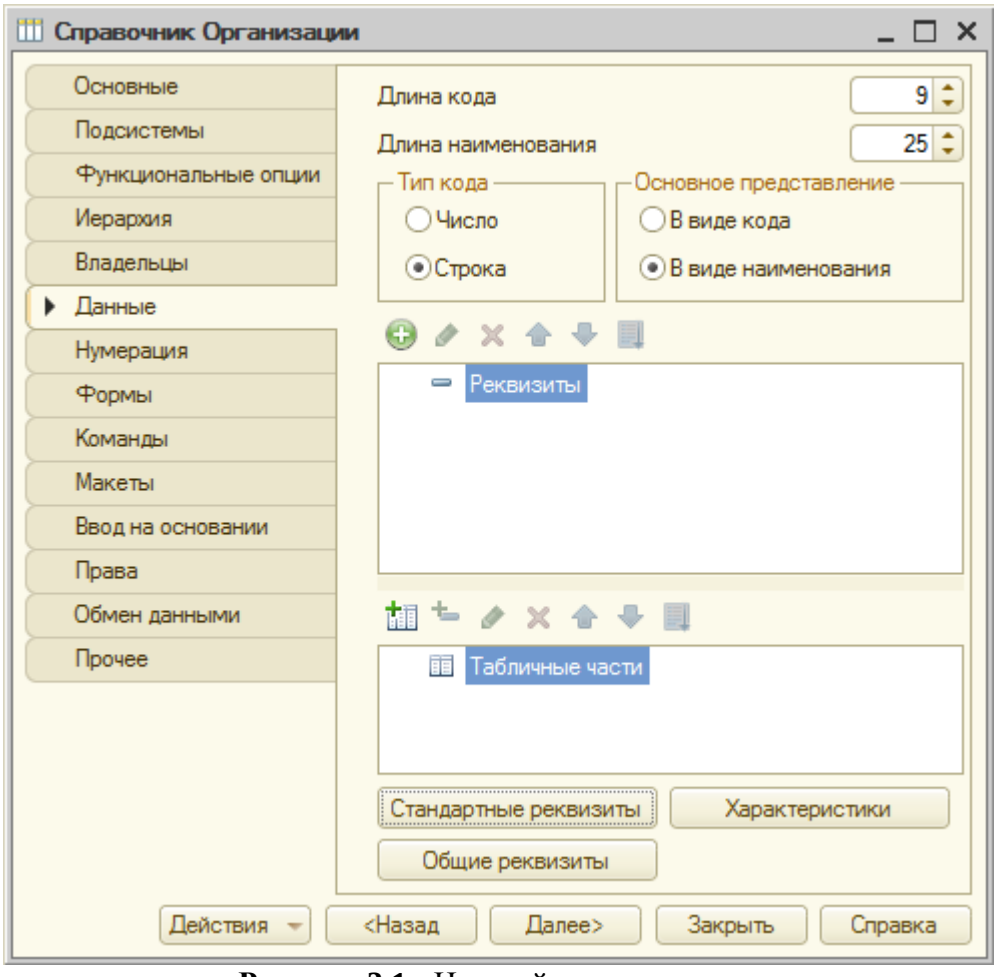


Рисунок 3.1- Настройка справочника

2. Ознакомимся со списком стандартных реквизитов можно, нажав на кнопку **Стандартные** реквизиты – появится окно, содержащее список таких реквизитов, Рисунок 3.2.

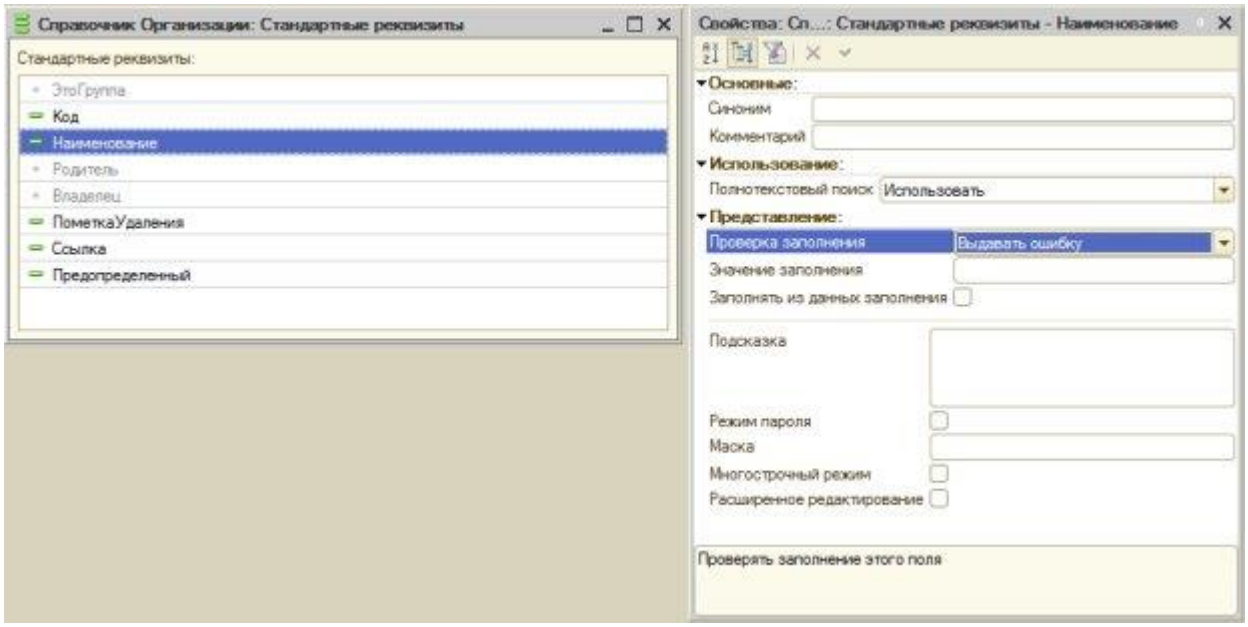


Рисунок 3.2- Стандартные реквизиты справочника и их свойства

Стандартные реквизиты поддерживают настройку некоторых свойств – для доступа к свойствам стандартного реквизита, достаточно выделить его в окне и обратиться к палитре **Свойства.**

3. Добавим реквизит в справочник. Нашему справочнику **Организации** не хватает, для полноты его использования в системе, реквизита, который содержал бы полное наименование организации. Добавим этот реквизит к справочнику – на вкладке **Данные** окна редактирования объекта, нажмем на кнопку **Добавить**, параметры реквизита будут следующими:

Имя: ПолноеНаименование

Тип: Строка, длина – 50.

Проверка заполнения: Выдавать ошибку

Свойство **Проверка заполнения** по умолчанию для новых реквизитов установлено в значение **Не проверять**. Оно позволяет автоматически проверять заполненность поля – если поле не заполнено – система выдаст ошибку. Если нам нужны особые алгоритмы проверки содержимого поля перед записью элемента справочника, мы можем реализовать эти алгоритмы самостоятельно.

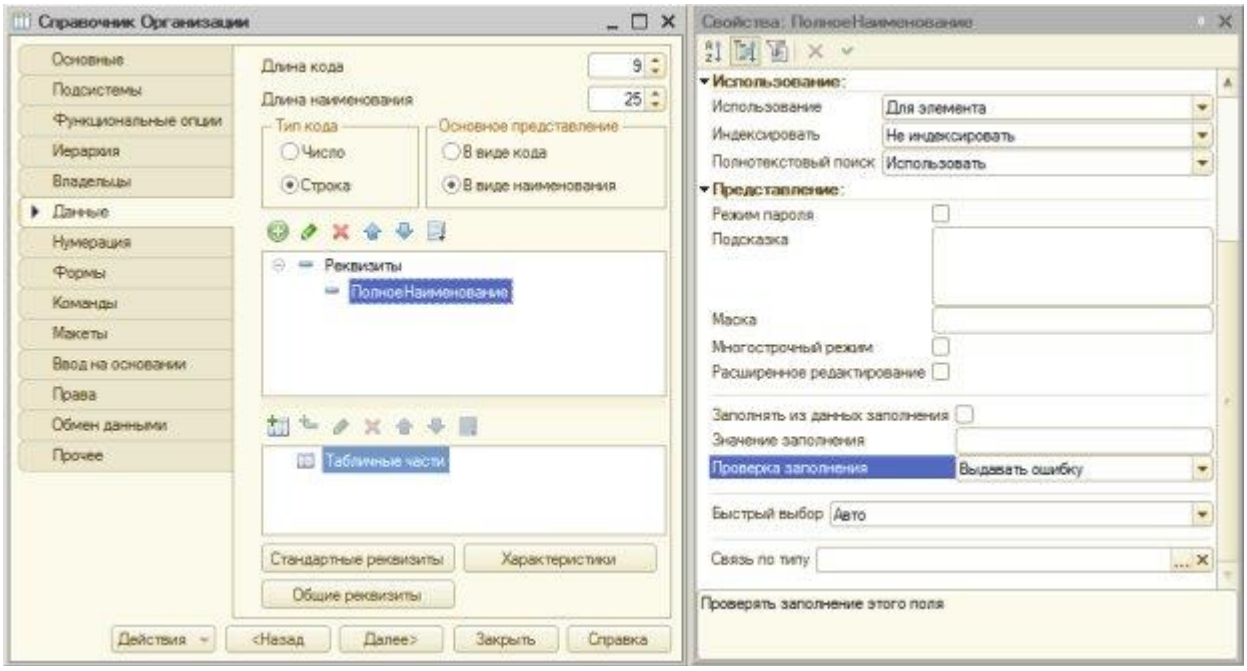


Рисунок 3.3- Настройка нового реквизита справочника

4. Посмотрим на наш справочник в режиме 1С:Предприятие. Создадим новый элемент, дадим ему наименование **Салон красоты**, а полное наименование заполнять не будем, и попытаемся записать элемент, нажав на кнопку **Записать и закрыть**. Элемент не будет записан, мы увидим сообщение об ошибке – в виде сообщения и в виде всплывающей подсказки, Рисунок 3.4.

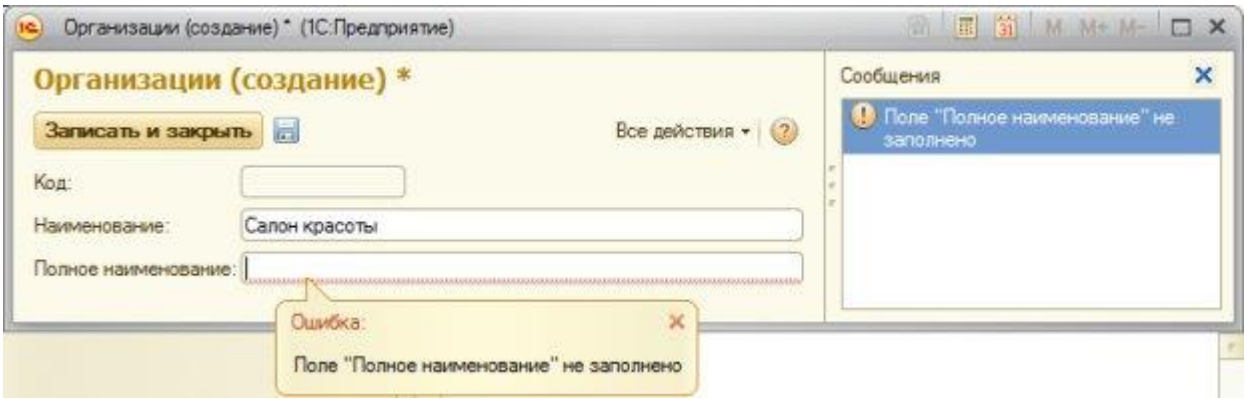


Рисунок 3.4- Сообщение об ошибке при попытке записи элемента справочника

Введем в поле **Полное наименование** текст ООО "Салон красоты" - после этого можно будет записать и закрыть элемент справочника. Он отобразится в списке справочника в рабочей области окна программы, Рисунок 3.5.

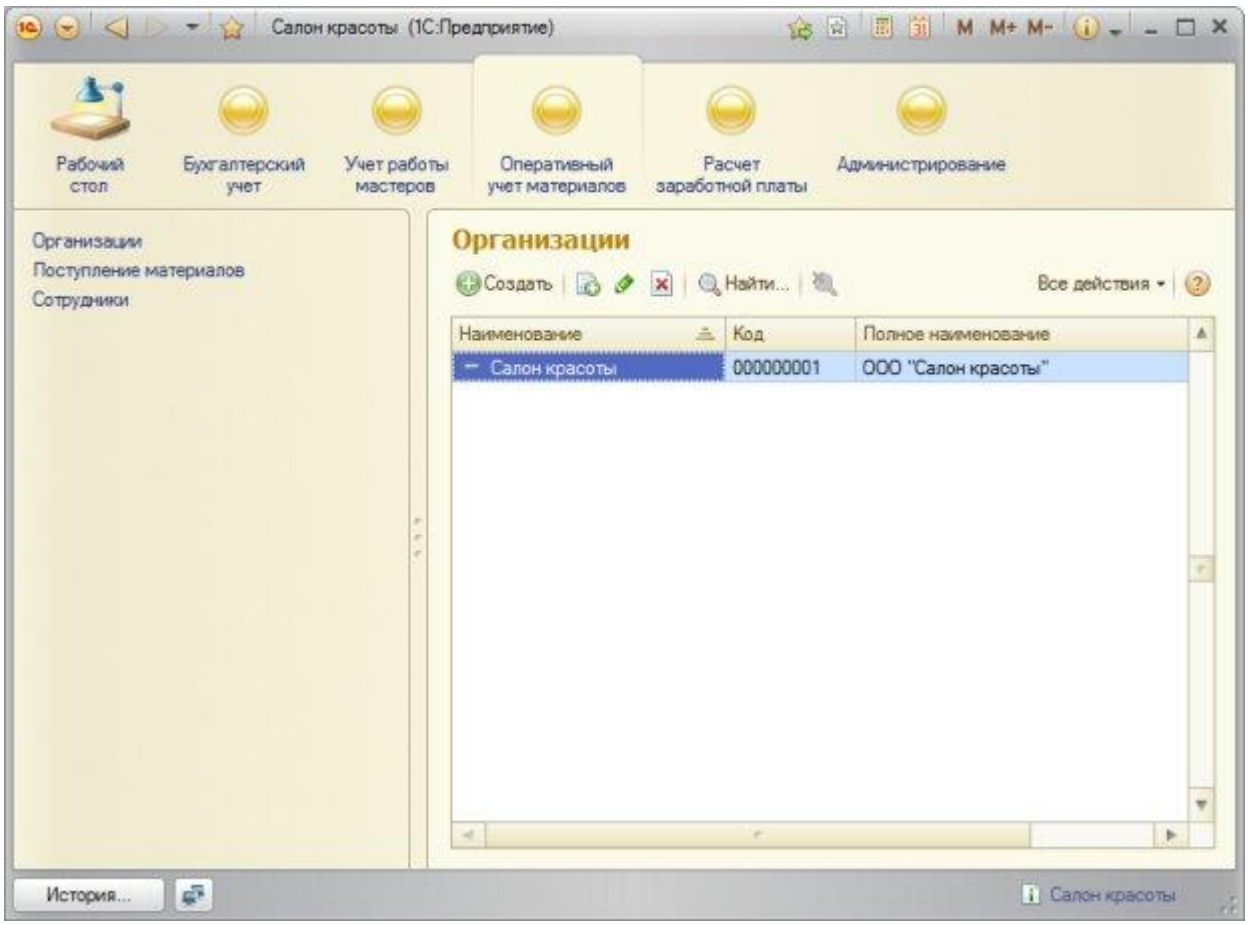


Рисунок 3.5- Новый элемент справочника в списке

В информационной панели, которая расположена в нижней части окна программы, появится ссылка для доступа к только что созданному элементу и будет сообщено о его создании.

Код элементу справочника будет присвоен автоматически.

Справочники в 1С:Предприятие могут содержать предопределенные элементы. К их созданию можно перейти с вкладки **Прочее**, по кнопке **Предопределенные**.

5. Создадим справочник ФизическиеЛица

5.1 Создадим новый справочник, дадим ему имя **ФизическиеЛица**, включим его в состав подсистем **БухгалтерскийУчет**,**УчетРаботыМастеров** и **РасчетЗаработнойПлаты**.

На вкладке Данные создадим следующие реквизиты:

Имя: Фамилия

Тип: Строка, длина 30

Имя: Имя

Тип: Строка, длина 30

Имя: Отчество

Тип: Строка, длина 30

Имя: ДатаРождения

Тип: Дата, состав даты – Дата

Справочник **ФизическиеЛица** предназначен для хранения списка физических лиц и сведений о них. В частности, мы хотели бы хранить данные о самом физическом лице (Фамилия, Имя, Отчество, дата рождения, пол, район проживания), а так же об истории его трудовой деятельности. Для хранения данных о физическом лице хорошо подойдут обычные реквизиты справочника, которыми мы уже занимались выше. А вот для того, чтобы хранить историю трудовой деятельности, нам понадобится другая структура данных, а именно – **табличная часть.**

Табличная часть – это таблица, состав и свойства полей (столбцов) которой мы задаем на этапе разработки. В пользовательском режиме создается необходимое количество строк. В нашем примере количество мест, в которых работало физическое лицо, заранее неизвестно.

Здесь надо отметить, что понятия "Сотрудник" и "Физическое лицо" - это разные вещи. Сотрудник – это тот, кто в настоящий момент работает в организации, и сотрудник обязательно является физическим лицом. А вот физическое лицо, сведения о котором могут храниться в базе данных организации, вполне может не являться сотрудником – например – это может быть кандидат на какую-либо должность, или, наоборот, уволенный сотрудник.

5.2 Следующие реквизиты, которые мы планируем создать – это **Пол** и **РайонПроживания**. Строковые реквизиты, которые мы создавали выше, обычно заполняют вводом данных с клавиатуры. В случае же с указанием пола и района проживания заполнение с клавиатуры непременно приведет к появлению в базе различных наименований для одних и тех же показателей при использовании текстовых полей. Для мужского пола это вполне может быть, при ограничении длины строки одним символом, "М" и "м", для районов так же возможно различное написание. Для обеспечения единообразия при вводе подобных показателей рационально использовать для их хранения отдельные справочники или перечисления. Для хранения наименований пола мы воспользуемся перечислением.

Создадим новое перечисление, дадим ему имя **Пол**, включим в подсистему **РасчетЗаработнойПлаты**. На вкладке **Данные** окна редактирования объекта для перечисления задаются значения перечисления. Зададим два значения – **Мужской** и **Женский**, Рисунок 3.6.

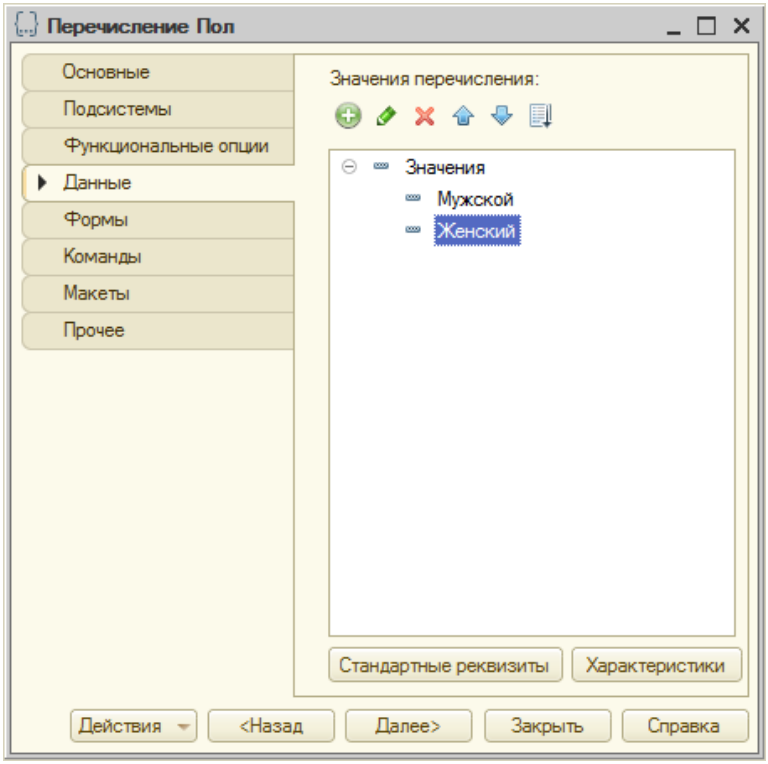


Рисунок 3.6- Создание перечисления Пол

6. Теперь создадим новый справочник – дадим ему имя **Районы**, включим в состав подсистемы **РасчетЗаработнойПлаты**, на вкладке **Данные** изменим длину наименования до**100** символов, этот справочник не будет иметь дополнительных реквизитов, так же мы можем исключить его из состава общего реквизита **Организация**, Рисунок 3.7.

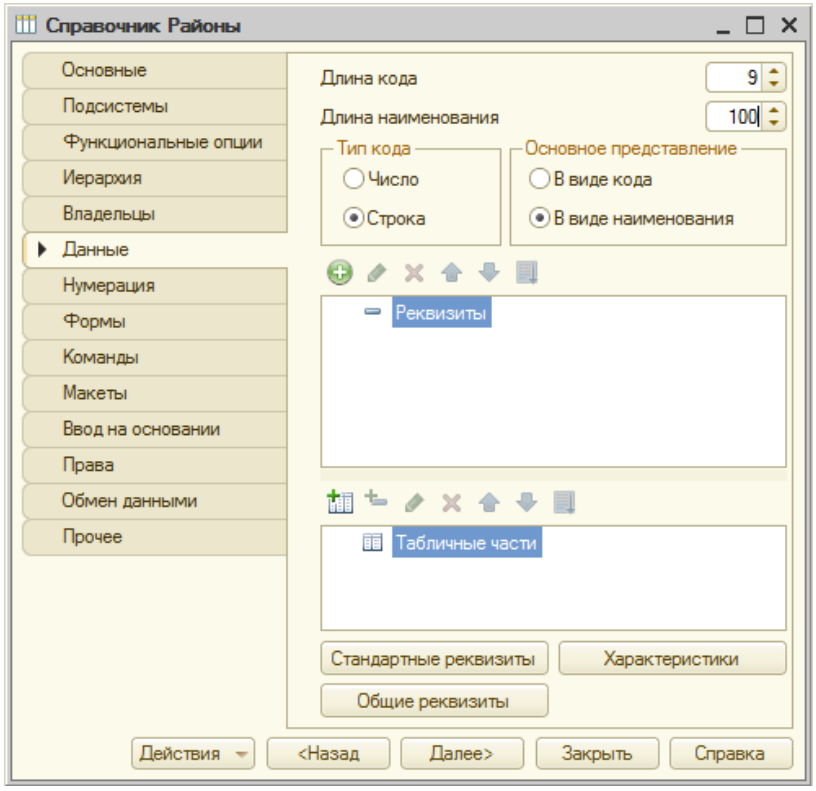


Рисунок 3.7 - Создание справочника Районы

Вернемся к настройке справочника **ФизическиеЛица**. Добавим еще два реквизита:

Имя: Пол

Тип: ПеречислениеСсылка.Пол

Имя: РайонПроживания

Тип: СправочникСсылка.Районы

Теперь займемся табличной частью справочника. При необходимости, справочники могут иметь несколько табличных частей. Сначала нажмем на кнопку **Добавить табличную часть**, зададим имя табличной части **ТрудоваяИстория**. В табличную часть добавим следующие реквизиты (поля), выделив табличную часть и нажав на кнопку**Добавить реквизит**:

Имя: Организация

Тип: Строка, длина 30

Имя: ДатаНачалаРаботы

Тип: Дата, состав даты – Дата

Имя: ДатаОкончанияРаботы

Тип: Дата, состав даты – Дата.

В итоге окно редактирования нашего справочника будет выглядеть так, как показано на Рисунок 3.8.

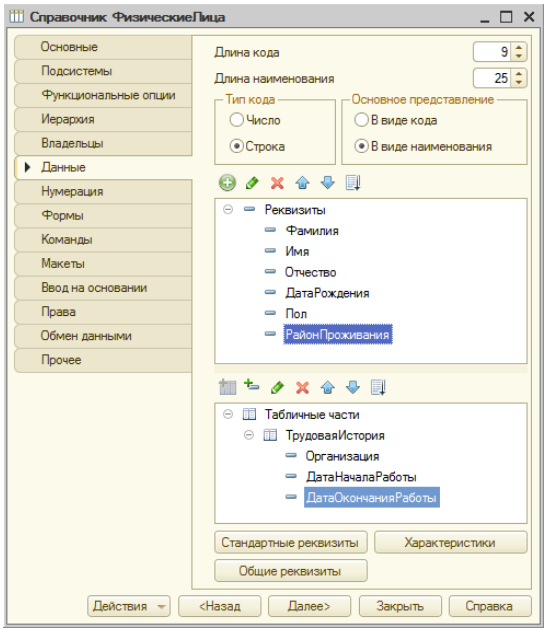


Рисунок 3.8- Состав справочника ФизическиеЛица

7. В предыдущей работе мы создавали общий реквизит **Организация**, который планировалось добавлять ко многим объектам конфигурации. Справочник **ФизическиеЛица** имеет смысл

вести по всем организациям. Как вы уже видели, настроить состав общего реквизита можно в ветви **Общие реквизиты**. Сделать это можно и в окне редактирования объекта, нажав кнопку **Общие реквизиты** на вкладке **Данные**. Нажмем эту вкладку и установим для общего реквизита **Организация** значение **Не использовать**, Рисунок 3.9.

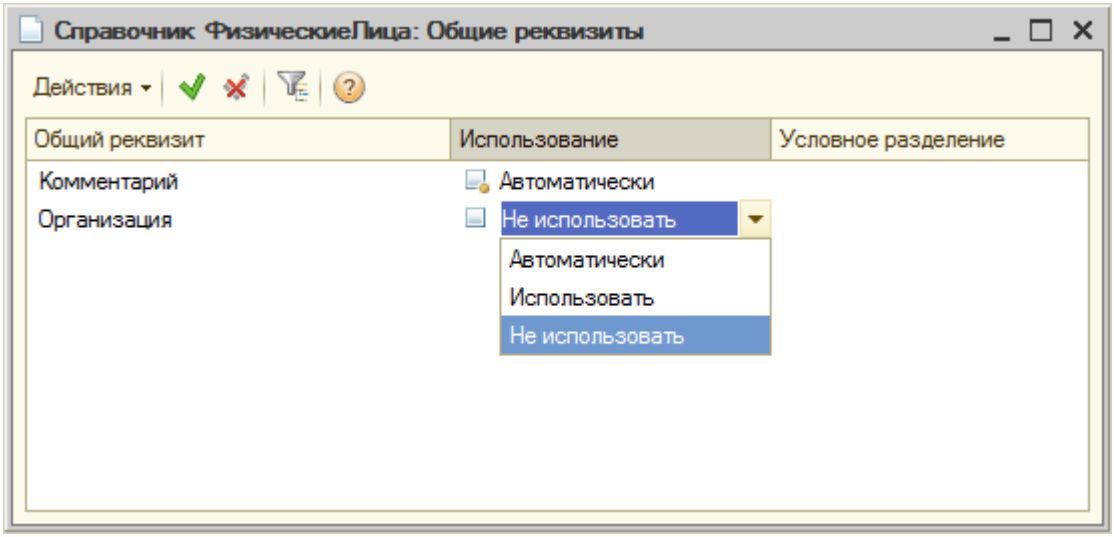


Рисунок 3.9- Настройка общих реквизитов из окна редактирования объекта

8. Просмотрим стандартную форму справочника. Если мы попытаемся открыть справочник в режиме 1С:Предприятие – с ним можно будет работать, так как система автоматически сгенерирует его форму, Рисунок 3.10. – с автоматически созданными формами мы уже встречались ранее. Такие формы подходят в том случае, если мы не планируем каким-либо образом вмешиваться в функционирование формы из Конфигуратора.

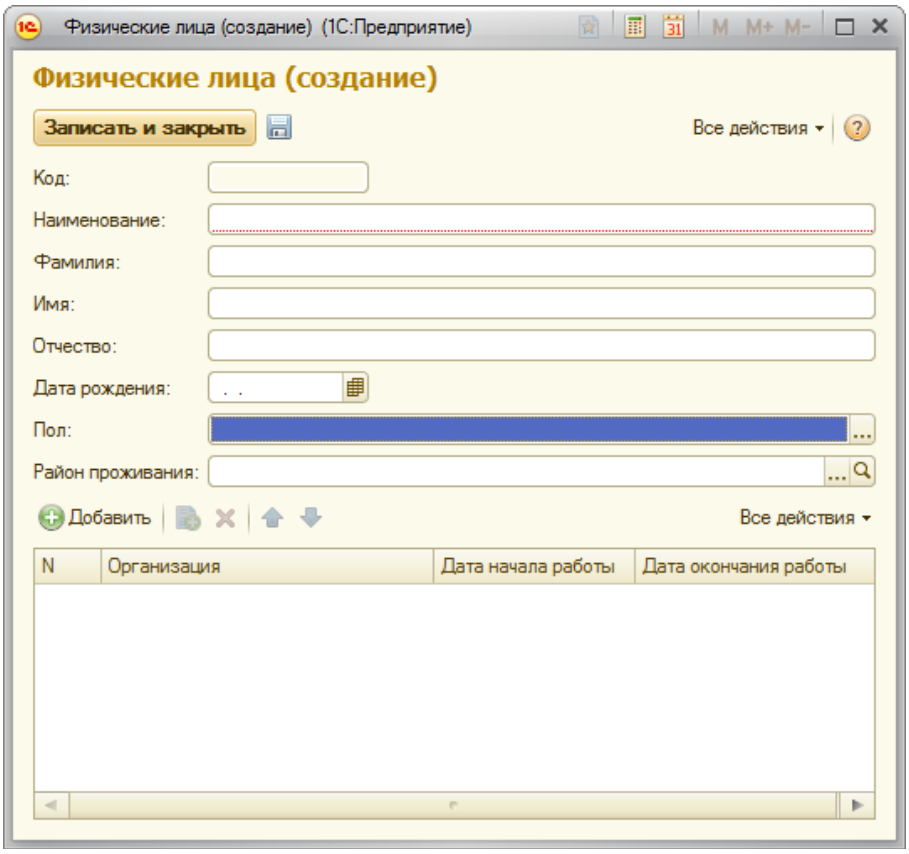


Рисунок 3.10- Форма справочника, сгенерированная автоматически

Если же решаемая нами задача требует каких-то особенных приемов работы с формой объекта, нам понадобится собственная форма. Например, это нам понадобится, если мы хотим автоматически заполнять поле **Наименование** на основе полей **Фамилия**, **Имя** и **Отчество**. А именно, мы хотели бы, чтобы наименование содержало фамилию и инициалы физического лица.

9. Разработаем форму справочника ФизическиеЛица

9.1 Откроем закладку **Формы** окна редактирования справочника **ФизическиеЛица**. Можно отметить, Рисунок 3.11, что ни одной формы не задано – то есть все они создаются системой автоматически. Нам же нужна собственная форма **элемента** справочника.

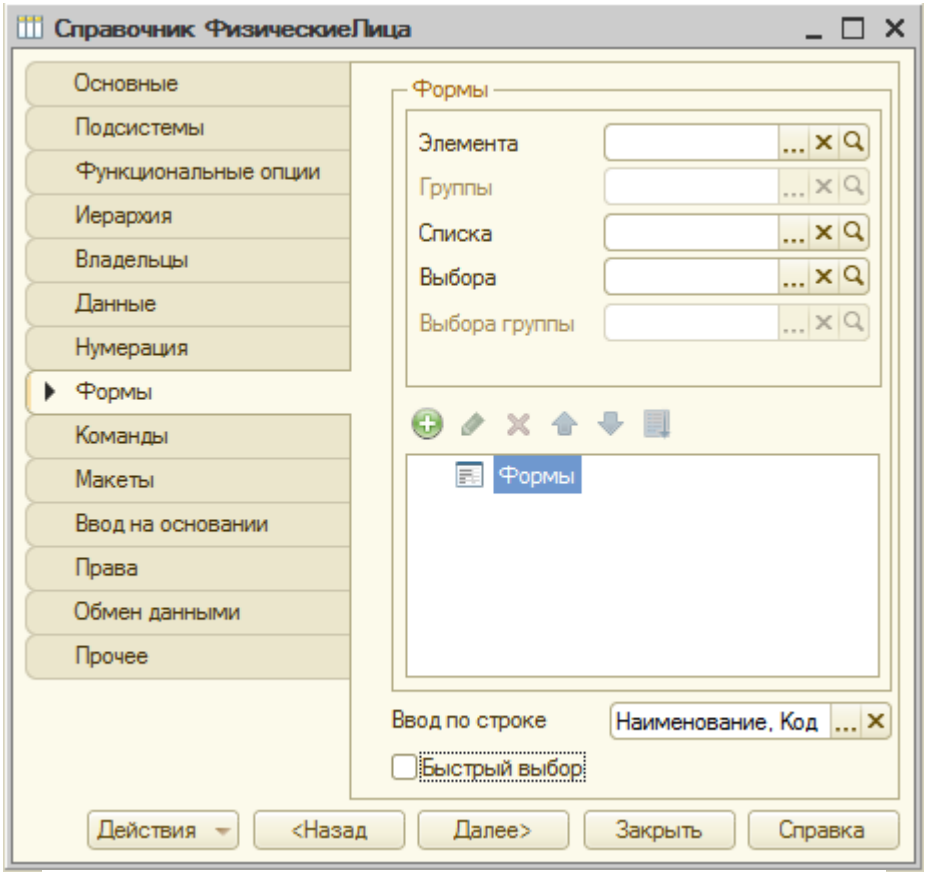


Рисунок 3.11- Вкладка Формы окна редактирования объекта

Нажмем на кнопку с увеличительным стеклом напротив поля **Элемента** в группе **Формы**. Появится окно **Конструктора формы справочника**, в его первом окне оставим все по умолчанию – а именно – нас интересует **Форма элемента справочника**, Рисунок 3.12.

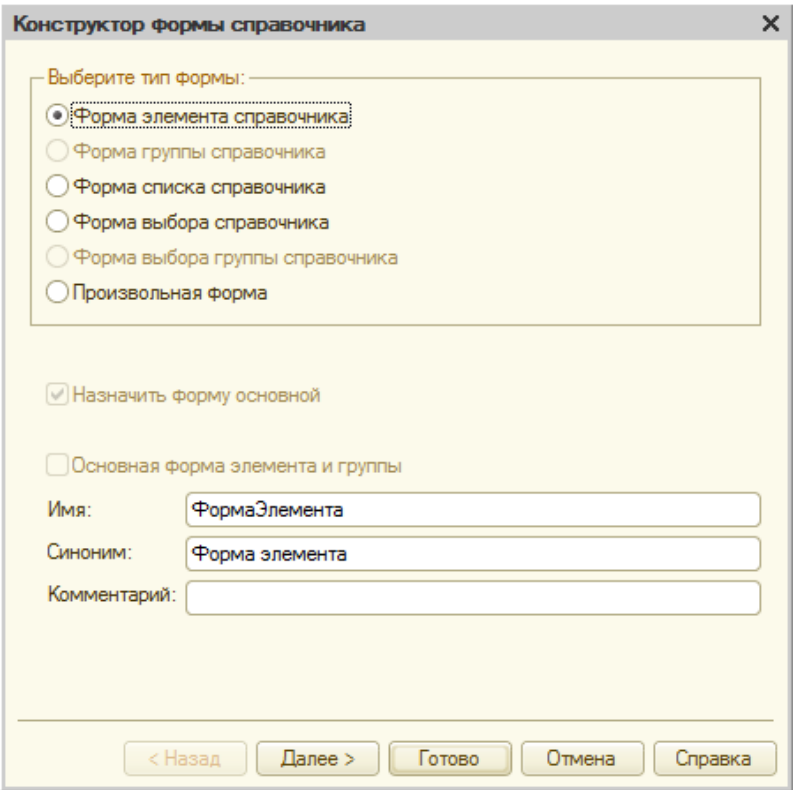


Рисунок 3.12- Первое окно конструктора форм справочника

9.2 В следующем окне, Рисунок 3.13., мы можем указать состав реквизитов для расположения на форме, а так же указать количество колонок, которое нужно для расположения элементов управления на форме. Оставим здесь все так же по умолчанию и нажмем на кнопку **Готово**.

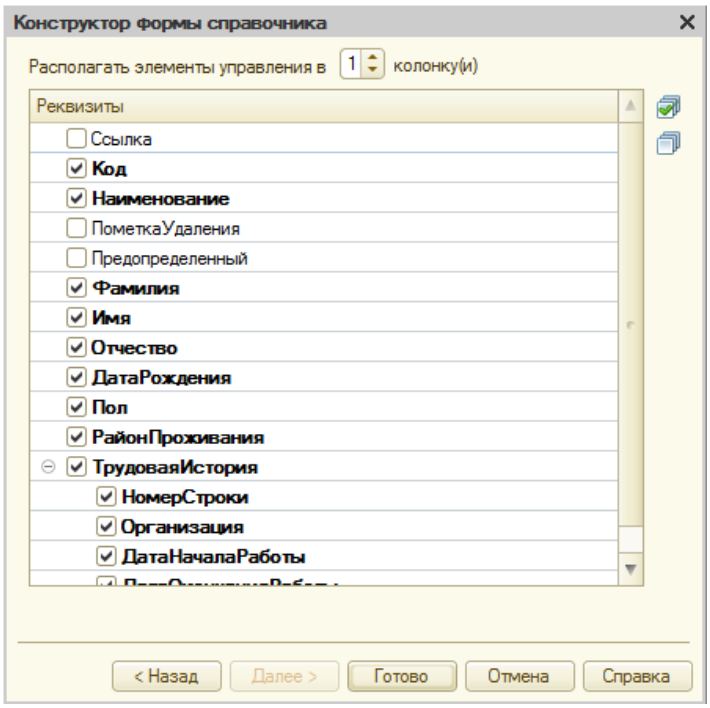


Рисунок 3.13- Второе окно конструктора форм справочника

9.3 После этого нужно открыть окно редактора форм для формы элемента справочника, Рисунок 3.14. Ранее мы уже сталкивались с этим окном, теперь рассмотрим его подробнее.

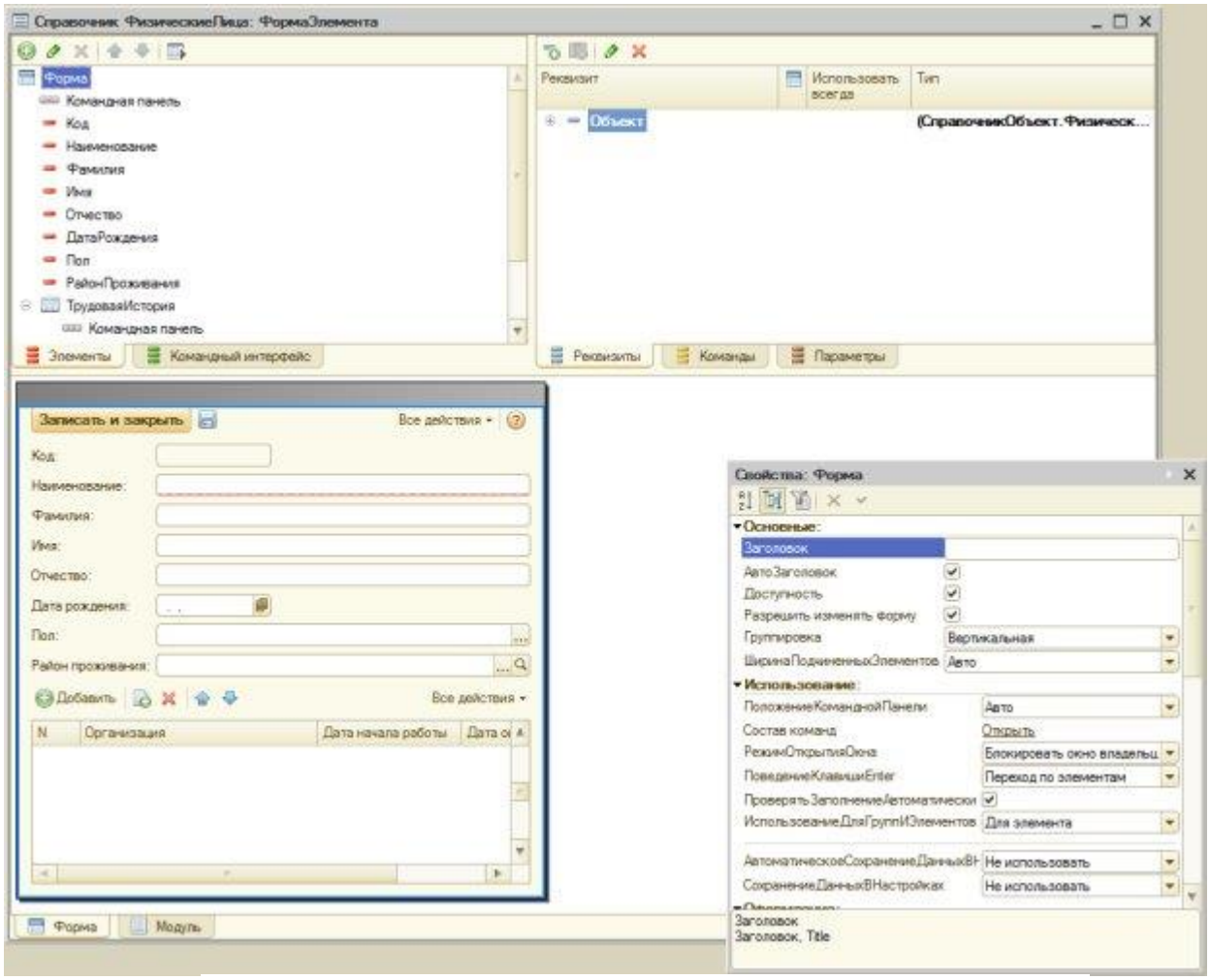


Рисунок 3.14- Окно редактирования формы элемента справочника

На самом деле, это окно объединяет в себе несколько редакторов и окон. В частности, это следующие:

9.4 Изучим окно формы. Редактор элементов формы (закладка **Элементы** в верхней левой части окна) – с его помощью можно контролировать элементы управления, которые будут расположены на форме. Выделив элемент в данном окне, мы можем настраивать его свойства в стандартной палитре свойств. Обратите внимание на кнопку **Проверить,** находящуюся в правой части командной панели закладки **Элементы**. Нажатие на нее приводит к выводу конструируемой формы в интерактивном виде, что позволяет лучше оценить ее внешний вид в пользовательском режиме, но, конечно, не дает возможности работать с данными информационной базы.

Окно просмотра формы (закладка **Форма** в нижней части окна) – здесь представлена форма в том виде, который она примет после настроек. Кроме того, выделяя элементы формы в данном окне, мы, не имея возможности, как это было ранее, произвольно перемещать их, можем вызывать их контекстное меню, Рисунок 3.14., с помощью которого можно перемещать элемент вверх или вниз (то же самое можно делать в окне **Элементы**), открывать окно его

свойств, назначать обработчики событий (их можно назначать и в окне **Свойства**, открытом для данного элемента).

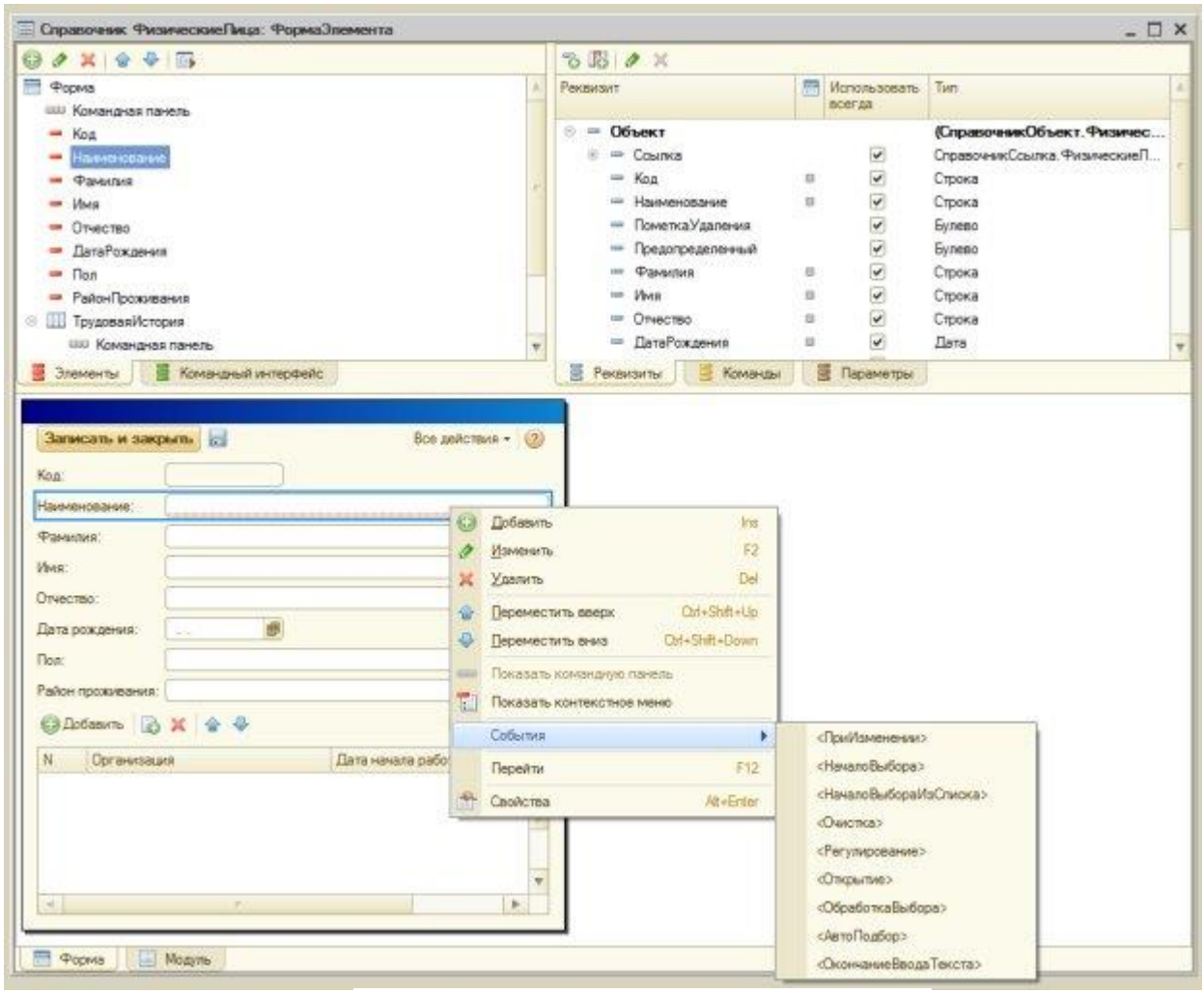


Рисунок 3.14- Работа с элементами формы

Редактор реквизитов представлен вкладкой **Реквизиты** (Рисунок 3.15.). Для того, чтобы добавить реквизит объекта на форму (то есть – создать элемент управления, связанный с данным реквизитом), достаточно перетащить элемент из окна **Реквизиты** в окно **Элементы.** Реквизиты, уже присутствующие на форме, отмечены серым квадратиком.

Редактор команд можно открыть, нажав на вкладку **Команды**. Здесь доступны три дополнительные вкладки. Вкладка **Команды формы** (по умолчанию пустая) содержит команды формы, их можно сравнить с командными кнопками, которые в версии 1С:Предприятие 8.1. можно было размещать на форме. Теперь последовательность действий выглядит так – сначала создать команду формы, потом перетащить ее в окно **Элементы**, настроить свойства, задать обработчики событий. Вкладка **Стандартные команды**(Рисунок 3.16.) содержит стандартный набор команд – в нашем случае – стандартный для формы и табличного поля, размещенного на форме.

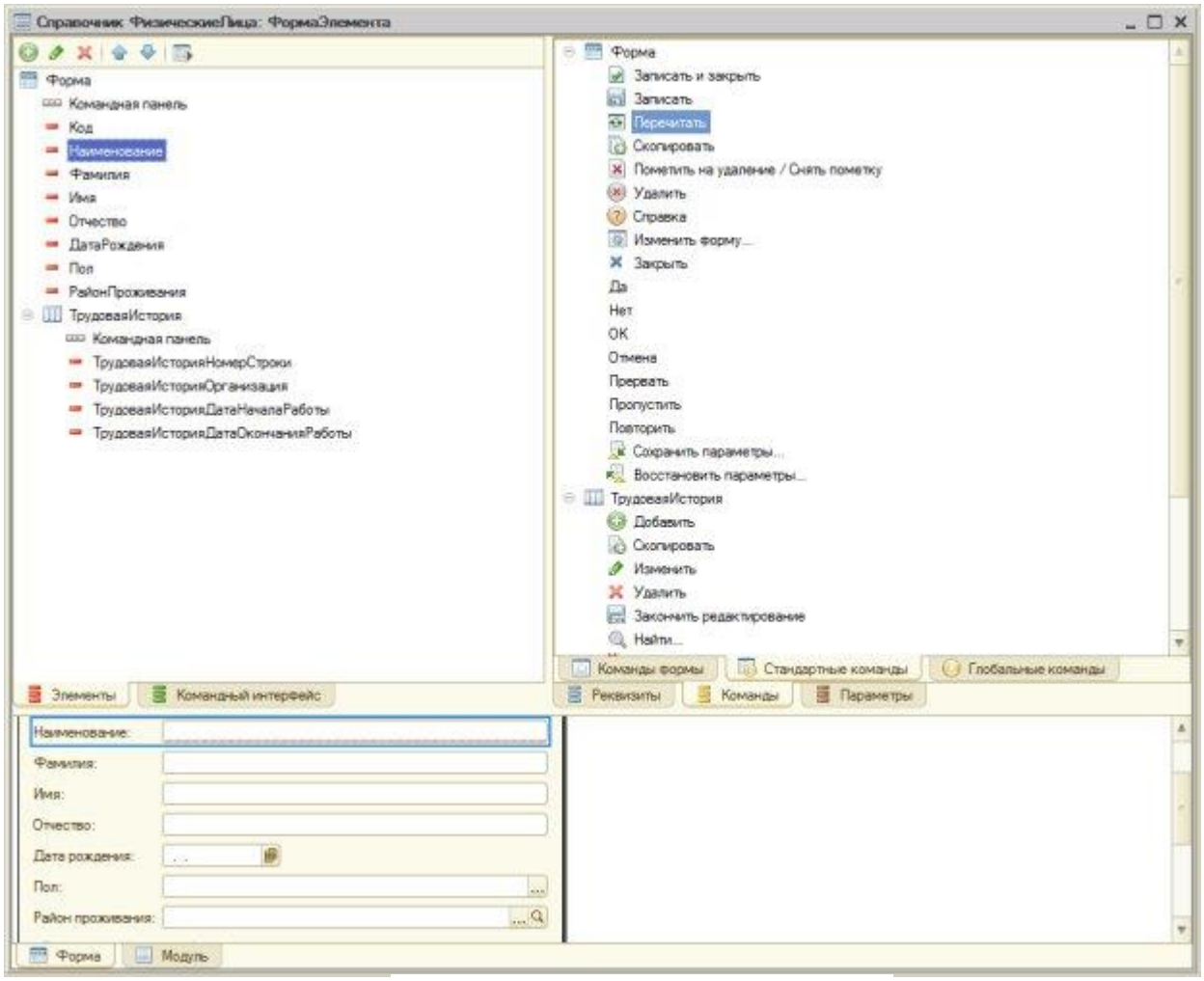


Рисунок 3.16- Стандартные команды

Вкладка **Глобальные команды** содержит набор команд уровня прикладного решения.

Вкладка **Параметры** предоставляет доступ к редактору параметров.

Вкладка **Командный интерфейс** позволяет редактировать командный интерфейс.

10. Реализуем автоматическое заполнение поля **Наименование** на основе полей **Фамилия**, **Имя** и **Отчество**.

Для этого сначала настроим элемент управления, отображающий наименование на форме, таким образом, чтобы его нельзя было редактировать. Выделим элемент управления в панели **Элементы**, откроем окно его свойств и установим свойство **ТолькоПросмотр**, Рисунок 3.17.. Благодаря этому свойству пользователь не сможет отредактировать текст в поле ввода. Похожего эффекта можно достичь и другими способами, например, указав в свойстве **Вид элемента** элемента **Наименование** вместо **Поле ввода –Поле надписи.**

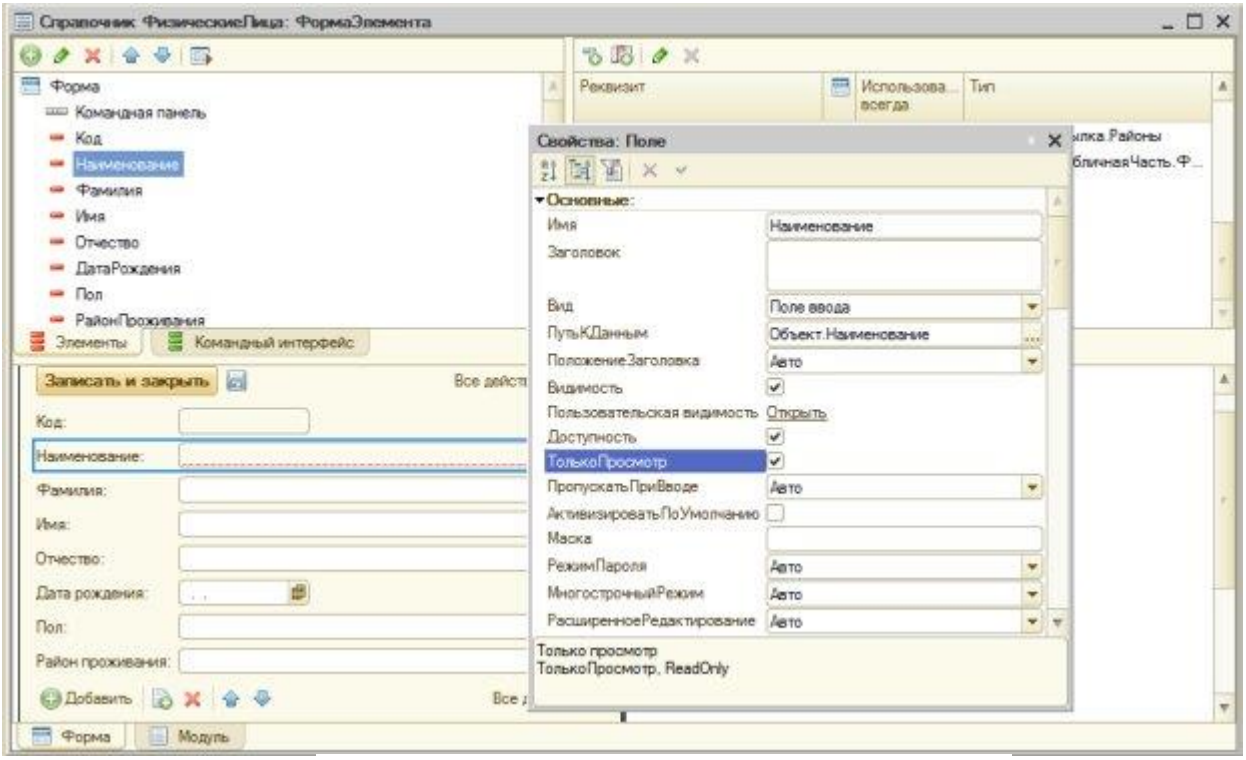


Рисунок 3.17- Настройка элемента Наименование

Для правильного формирования наименования важно, чтобы пользователь ввел данные в поля **Фамилия**, **Имя** и **Отчество**.

Практическая работа №4 Создание объекта конфигурации «Обработка»

Цель: научиться написанию программного кода справочника формирования наименования из других реквизитов с проверкой их заполнения

ХОД РАБОТЫ

1. Применим клиентские методы в модуле формы

Теперь перейдем к написанию кода, в котором будем формировать наименование. Для этого нам нужно понимать, что конфигурации 1С:Предприятие управляются событиями – и сейчас нас интересуют события формы.

1.1 Выделим форму в окне Элементы, откроем окно ее свойств и рассмотрим группу свойств **События**, Рисунок 4.1.. Наименование должно быть сформировано до того, как данные объекта будут записаны. Для достижения нашей цели нам вполне подойдет событие **ПередЗаписью**. Здесь же можно выполнить какие-либо пользовательские проверки полей перед формированием наименования. Хотя, если говорить о производительности решения, лучше подобные проверки производить на сервере, например, с помощью обработчика события **ОбработкаПроверкиЗаполнения**, который создается в модуле объекта.

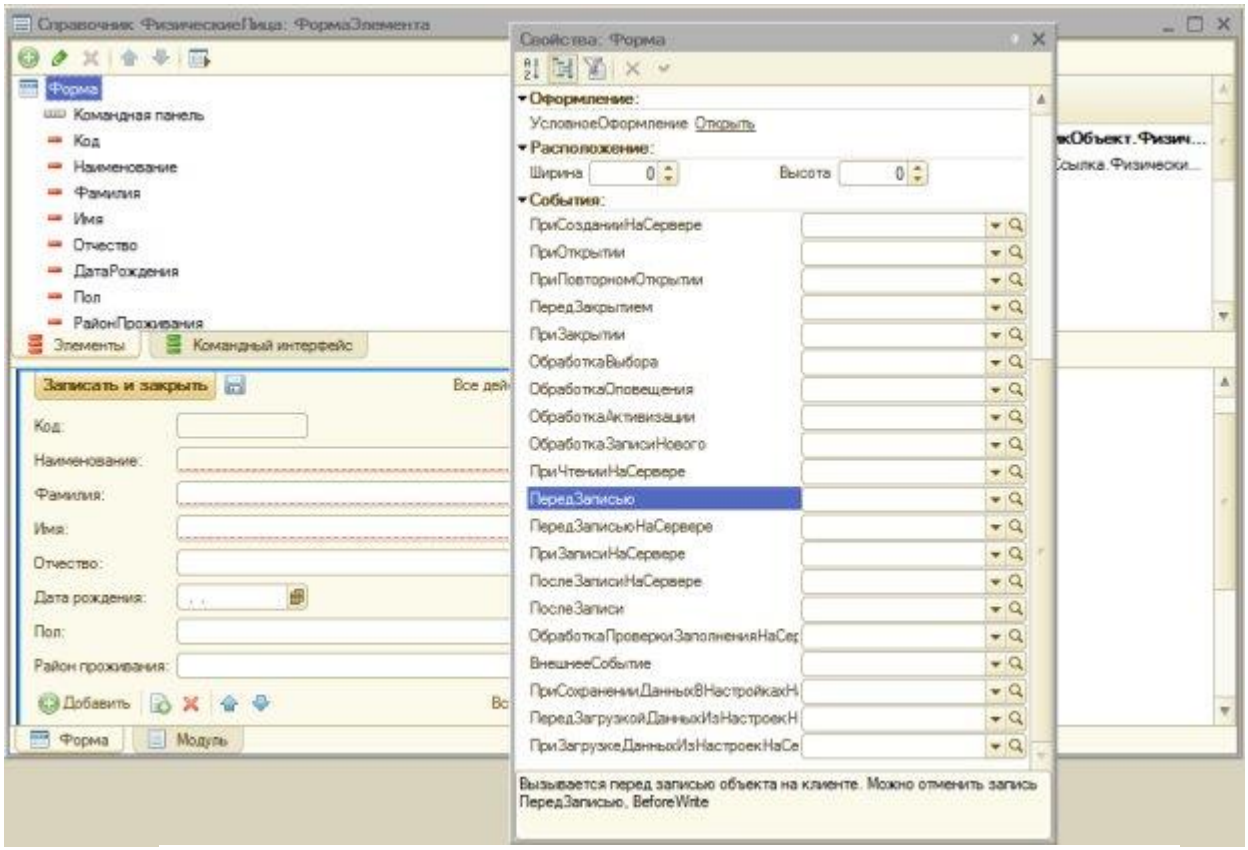


Рисунок 4.1- Выбор события для выполнения запланированных действий

1.2 Нажмем на кнопку с увеличительным стеклом в поле события **ПередЗаписью** – автоматически будет открыт модуль формы и создан пустой обработчик события**ПередЗаписью**. Он имеет следующий вид:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
  // Вставить содержимое обработчика.
КонецПроцедуры>
```

Из директивы компиляции **&НаКлиенте** понятно, что процедура это клиентская, она имеет два параметра – нас сейчас интересует параметр **Отказ** – благодаря этому параметру, а именно, установив его в значение **Истина**, мы можем отказаться от записи объекта в том случае, если выполняется какое-либо условие, препятствующее записи. В нашем случае записи объекта могут воспрепятствовать незаполненные или неправильно заполненные поля **Фамилия**, **Имя** или **Отчество.**

1.3 Проверку на незаполненность реквизита мы можем доверить и системе – для этого можно установить свойство **Проверка заполнения** для нужных реквизитов в значение **Выдавать ошибку**, делается это в списке реквизитов объекта в окне редактирования объекта или в дереве конфигурации, Рисунок 4.2. Не будем включать проверку заполнения, выполним ее и еще некоторые проверки самостоятельно.

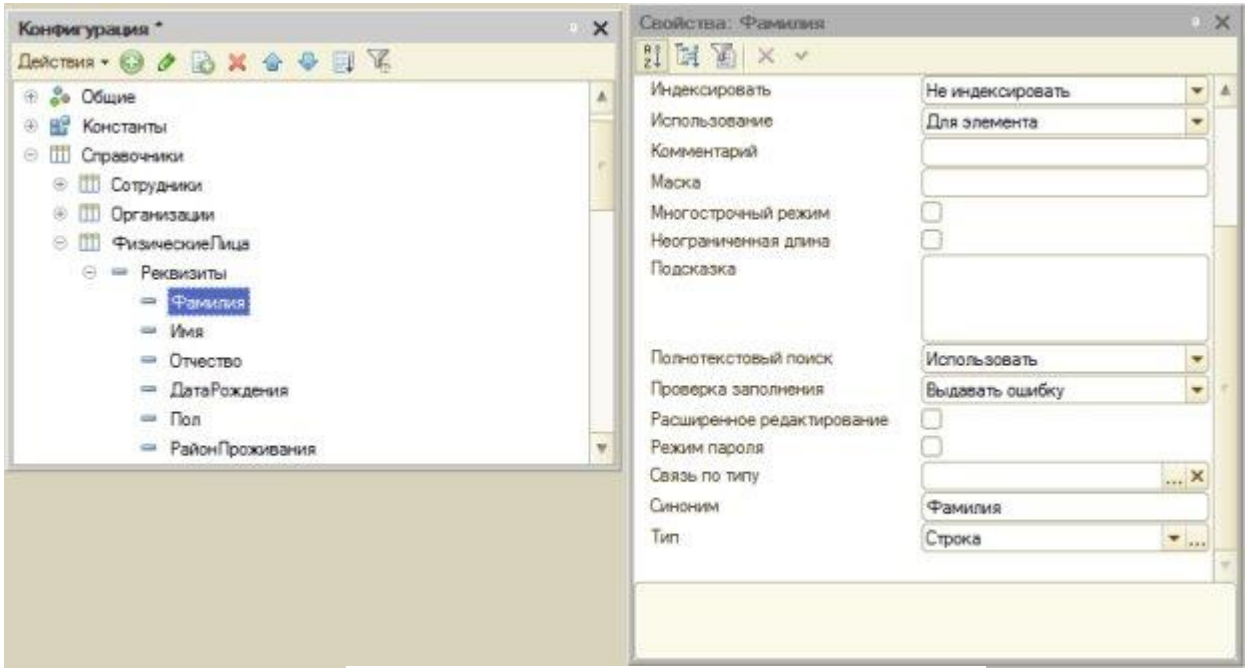


Рисунок 4.2- Настройка проверки заполнения

Параметры и процедуры в системе 1С:Предприятие по умолчанию передаются по ссылке – передав в процедуру некую переменную, мы, на самом деле, передаем ссылку на нее, то есть – при модификации соответствующего этой переменной параметра внутри процедуры, фактически, происходит и модификация переменной. Вернемся к нашей процедуре **ПередЗаписью**.

1.3 В процедуре **ПередЗаписью** мы сначала проверим поля **Фамилия**, **Имя** и **Отчество** на заполненность (возможны и более сложные проверки), после чего, если хотя бы одно поле не заполнено – сообщим об этом пользователю и выйдем из процедуры, если все поля заполнены – сформируем наименование. Вот какой код позволяет реализовать эту задачу:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
 //Переменная для хранения текста сообщения пользователю
 Перем ТекстСообщения;
 //Запишем пустую строку в переменную
 ТекстСообщения="";
  //Если не введена фамилия...
 Если ПустаяСтрока(Объект.Фамилия) Тогда
   //Формируем строку сообщения
   ТекстСообщения=ТекстСообщения+"Не заполнено поле Фамилия;";
 КонецЕсли;
  //Если не введено имя...
 Если ПустаяСтрока(Объект.Имя) Тогда
   ТекстСообщения=ТекстСообщения+" Не заполнено поле Имя;";
 КонецЕсли;
 //Если не введено отчество...
 Если ПустаяСтрока(Объект.Отчество) Тогда
   ТекстСообщения=ТекстСообщения+" Не заполнено поле Отчество;";
 КонецЕсли;
  //Если строка сообщения не пуста, то есть - содержит
```

```
        //сообщения о незаполненных полях
        Если НЕ ПустаяСтрока(ТекстСообщения) Тогда
         //Выводим сообщение
         Сообщить(ТекстСообщения);
         //Отказываемся от записи объекта
         Отказ=Истина;
         //Выходим из процедуры
         Возврат;
        КонецЕсли;
        //Если все поля заполнены, выхода из процедуры не произошло,
        //формируем наименование
        Объект.Наименование=Объект.Фамилия+" "+ ВРег(Лев(Объект.Имя,1))+".
"+ВРег(Лев(Объект.Отчество,1))+".";
      КонецПроцедуры
```

1.4 Строковая функция **Лев** позволяет получить заданное количество символов из строки, начиная с самого левого. Строковая функция **ВРег** переводит символы в верхний регистр – на тот случай, если пользователь случайно ввел имя, фамилию или отчество с маленькой буквы. Конечно, здесь можно предусмотреть еще множество проверок и автоматических корректировок (например, можно исправить первую букву во введенных фамилии, имени и отчестве, если она случайно введена в нижнем регистре), мы ограничимся тем, что сделано сейчас.

В итоге мы получаем следующие сообщения об ошибках при незаполненности полей, Рисунок 4.3.

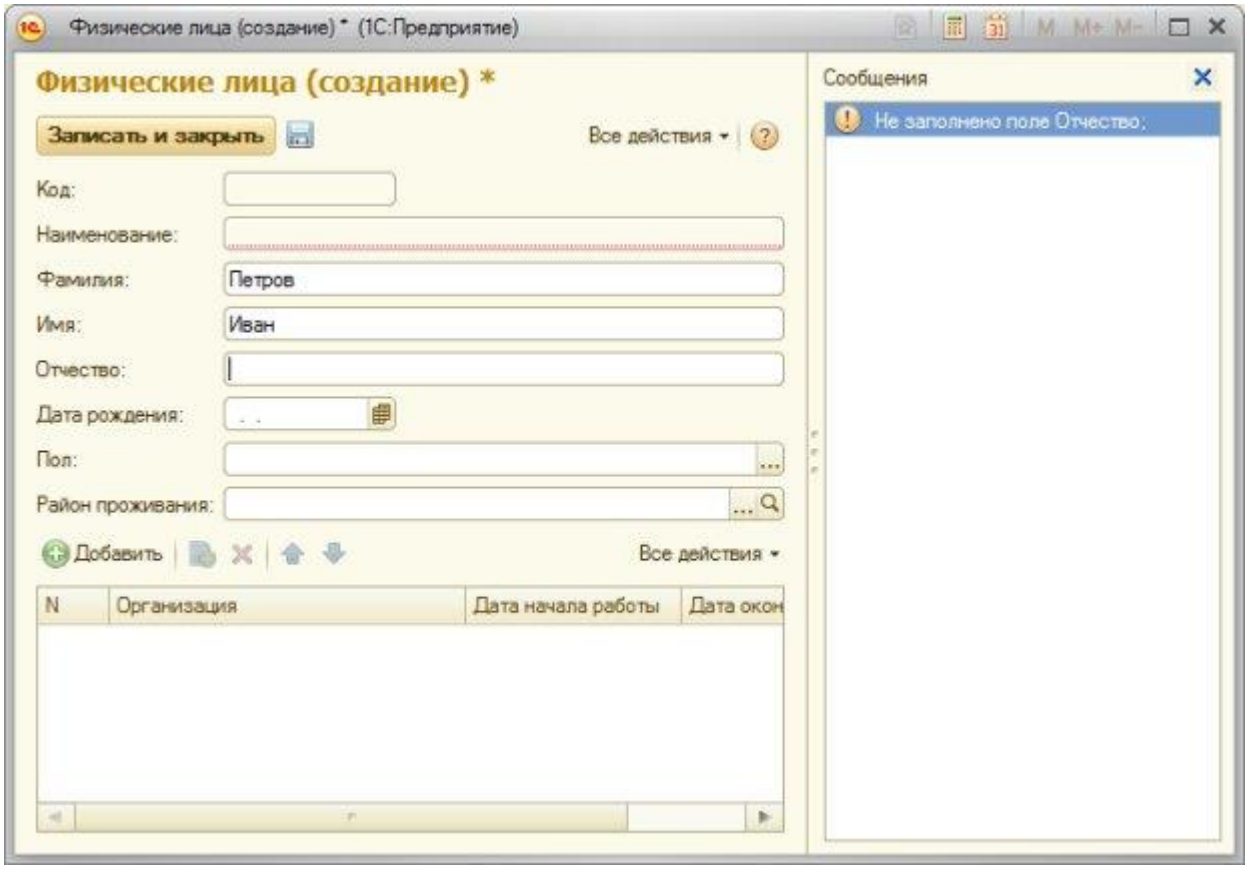


Рисунок 4.3- Сообщение об ошибке

1.5 После успешного выполнения процедуры **ПередЗаписью**, наименование выглядит следующим образом. Мы намеренно ввели отчество с маленькой буквы – как было пояснено выше, наш код готов к такому повороту событий, Рисунок 4.4.

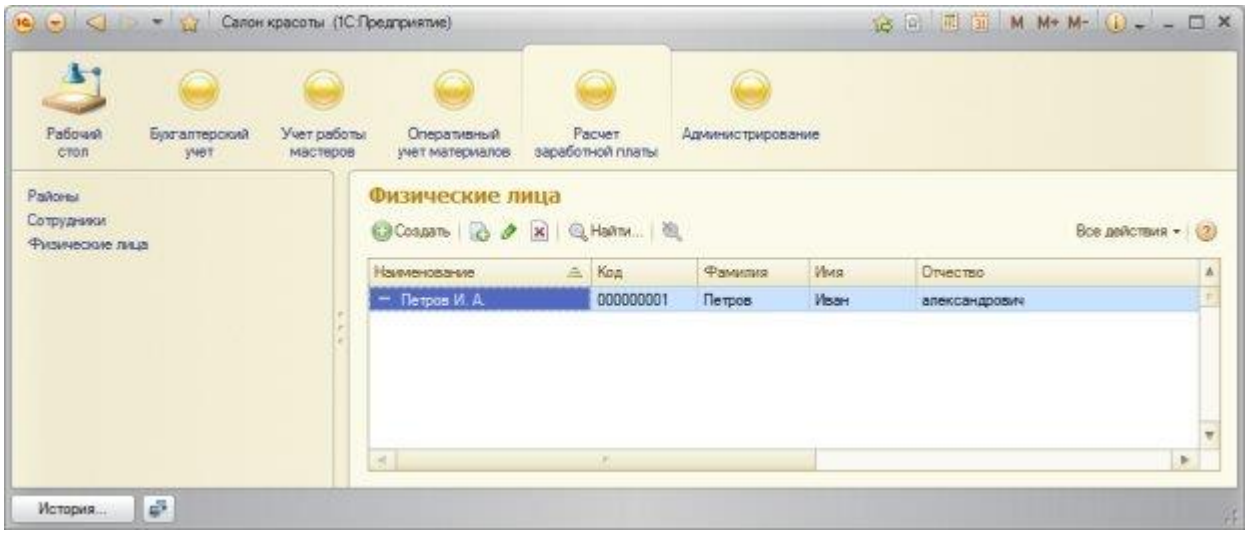


Рисунок 4.4- Новая запись в справочнике Физические лица

1.6 Запрограммируем сообщение пользователю

Обратите внимание на то, что здесь мы пользуемся обычным методом **Сообщить** – мы выводим в окно сообщения одно сообщение, содержащее необходимые сведения. В 1С:Предприятие 8.2. мы можем поступить по-другому – вывести сообщения об ошибках или другие сведения, "привязав" их к полям, которые вызвали ошибки. Для этого можно воспользоваться объектом**СообщениеПользователю.** Он, помимо прочих полезных возможностей, позволяет формировать сообщения и "привязывать" их к реквизитам формы.

Перепишем код таким образом, чтобы сообщения об ошибках (то есть, о незаполненных полях **Фамилия**, **Имя**, или**Отчество**), выявленных в процедуре ПередЗаписью, выводились бы в привязке к соответствующим элементам формы. Вот какой код позволяет этого добиться:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
 //Если не введена фамилия...
 Если ПустаяСтрока(Объект.Фамилия) Тогда
   СообщитьПользователю("Объект.Фамилия", "Заполните поле Фамилия", Отказ);
 КонецЕсли;
 //Если не введено имя...
 Если ПустаяСтрока(Объект.Имя) Тогда
   СообщитьПользователю("Объект.Имя", "Заполните поле Имя", Отказ);
 КонецЕсли;
 //Если не введено отчество...
 Если ПустаяСтрока(Объект.Отчество) Тогда
   СообщитьПользователю("Объект.Отчество", "Заполните поле Отчество", Отказ);
 КонецЕсли;
 //Если флаг Отказ не был установлен - формируем наименование
 Если НЕ Отказ Тогда
   Объект.Наименование=Объект.Фамилия+" "+ ВРег(Лев(Объект.Имя,1))+".
```

```
"+ВРег(Лев(Объект.Отчество,1))+".";
 КонецЕсли;

КонецПроцедуры

&НаКлиенте
//Процедура, формирующая и выводящая сообщение с переданными ей параметрами
Процедура СообщитьПользователю(ПутьКРеквизиту, Текст, Отказ)
 Сообщение=Новый СообщениеПользователю;
 Сообщение.Поле=ПутьКРеквизиту;
 Сообщение.Текст=Текст;
 Сообщение.Сообщить();
 Отказ=Истина;
КонецПроцедуры
```

1.7 Поясним приведенный код. Для начала, мы создали новую клиентскую процедуру **СообщитьПользователю**. Эта процедура принимает три параметра. Первый – **ПутьКРеквизиту** содержит строковый путь к полю, к которому должно быть привязано сообщение. Второй – **Текст** – содержит текст сообщения. Третий – **Отказ** – используется для установки в значение **Истина** параметра **Отказ** процедуры **ПередЗаписью** в том случае, если процедура **СообщитьПользователю** будет вызвана хотя бы один раз. А хотя бы однократный ее вызов означает, что одно из полей не заполнено, то есть наименование сформировать невозможно, соответственно, записать объект так же не получится.

Когда процедура вызывается, мы сначала создаем новый объект типа **СообщениеПользователю**. Затем его свойство **Поле** устанавливаем в значение параметра**ПутьКРеквизиту**. Этот параметр должен быть строковым и имеет, в нашем случае вид "Объект.Фамилия", "Объект.Имя", "Объект.Отчество" - это позволяет правильно "привязать" сообщение к полям формы. Свойство **Текст** объекта **СообщениеПользователю** содержит текст для вывода.

Мы, кроме того, полностью переработали процедуру **ПередЗаписью**. А именно, если проверка на заполнение поля указывает на то, что поле пустое, вызывается процедура**СообщитьПользователю**. По окончании проверок мы проверяем, установлен ли параметр **Отказ** в значение **Истина** – если не установлен – ни одна из проверок не завершилась обнаружением пустого поля и мы можем формировать наименование. Если установлен – наименование мы не формируем – и процедура заканчивает работу, а записи объекта, естественно, не происходит – пользователь видит лишь сообщения об ошибках.

Если было сформировано несколько сообщений типа **СообщениеПользователю** – пользователь видит одно окно сообщения около поля, но это окно снабжено кнопками для перемещения вперед и назад – щелчок по кнопке приводит к "переходу" сообщения от одного поля с ошибкой к другому Рисунок 4.5, 4.6.

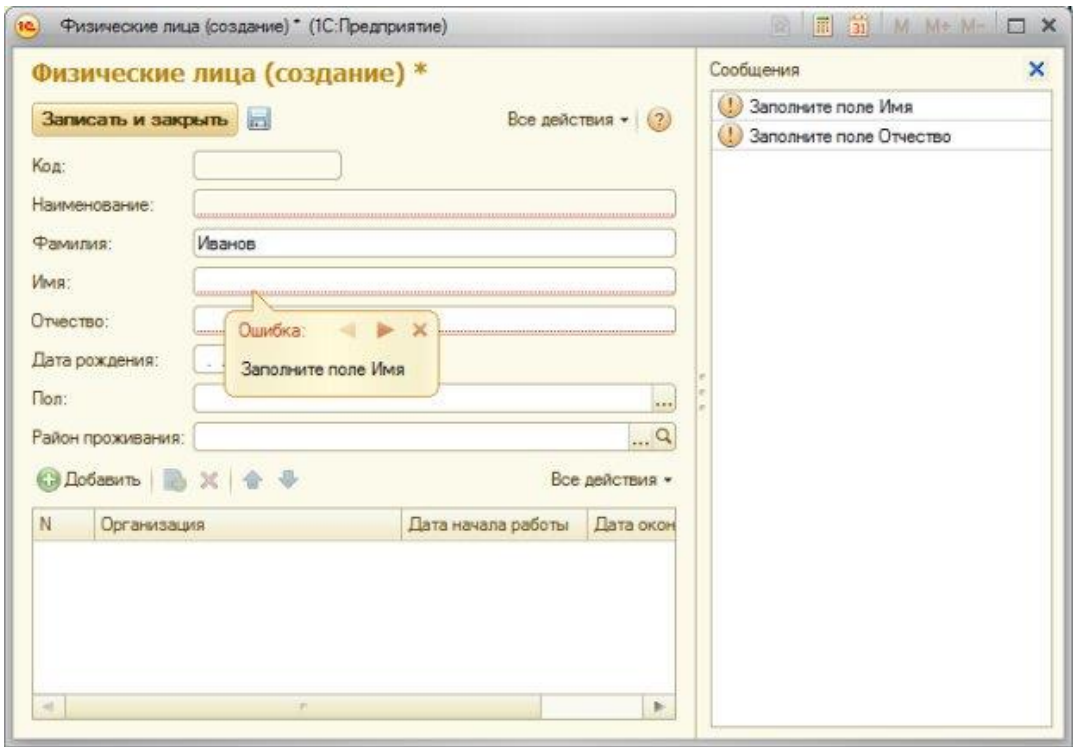


Рисунок 4.5- Сообщение об ошибке, привязанное к полю Имя

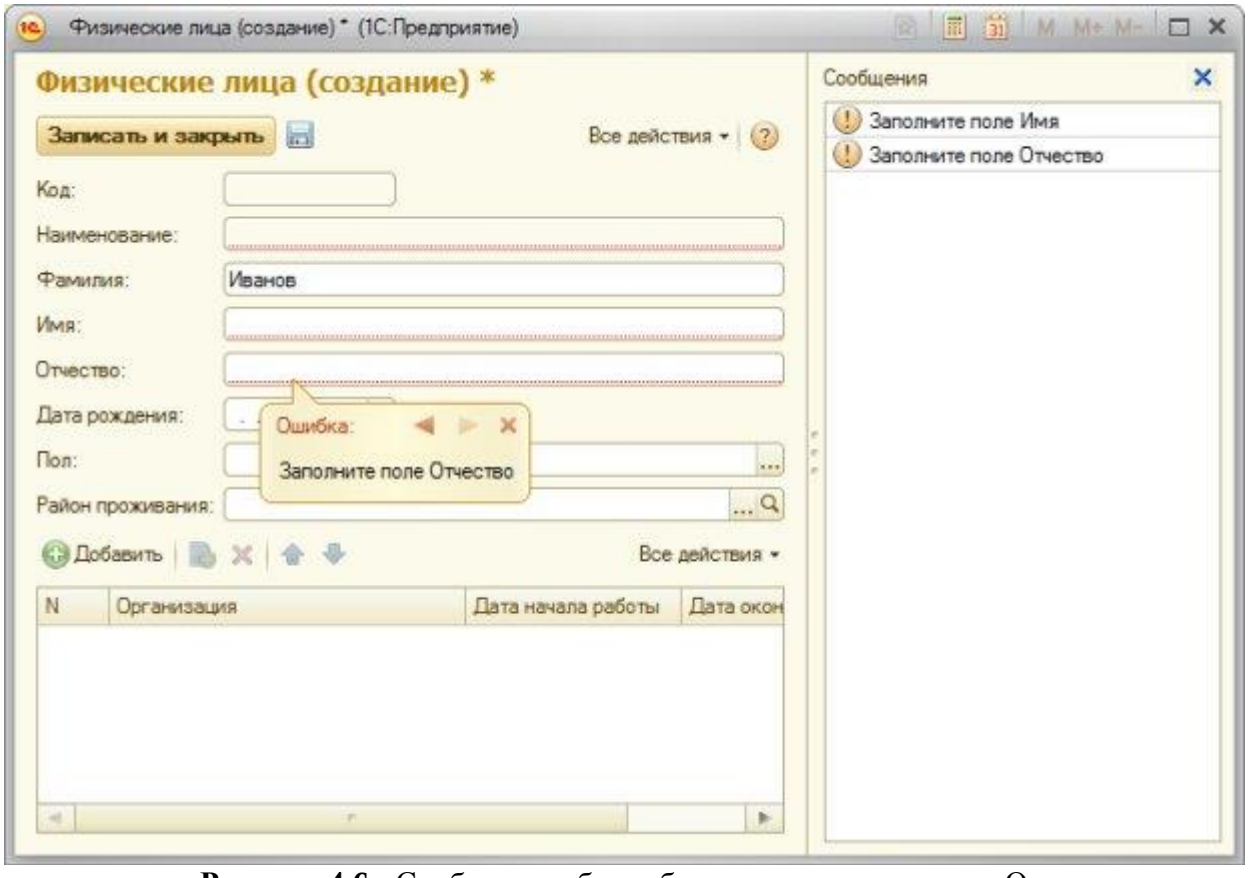


Рисунок 4.6- Сообщение об ошибке, привязанное к полю Отчество

1.8 Доведем до логического завершения пример со справочником **ФизическиеЛица**. Для этого заполним справочник Районы и введем в информационную базу сведения о следующих физических лицах:

Фамилия	Имя	Отчество	Дата рождения	Пол	Район
Иванов	Иван	Иванович	27.02.1984	Мужской	Ленинский
Петров	Петр	Петрович	12.06.1985	Мужской	Ленинский
Васильев	Павел	Петрович	17.05.1985	Мужской	Ленинск й
Расчетчиков	Александр	Иванович	12.03.1980	Мужской	Октябрьский
Александров	Александр	Александрович	17.09.1970	Мужской	Октябрьский
Бухгалтерова	Василиса	Владимиров	11.08.1976	Женский	Советский

Обратите внимание на то, что справочник **ФизическиеЛица** – это пример справочника, с которым пользователям нашей информационной базы придется работать достаточно часто. В данный момент для того, чтобы создать новый элемент справочника, нам нужно выполнить несколько действий – перейти в раздел **Расчет заработной платы**, щелкнуть по ссылке, открывающей список справочника, после чего нажать на кнопку **Создать новый элемент списка**. Для того, чтобы сократить количество действий, необходимых для выполнения часто используемых операций, мы можем соответствующим образом настроить интерфейс нашего прикладного решения, в частности, поработать с панелью действий соответствующего раздела и с **Рабочим столом**.

2. Выполним настройку командного интерфейса для ускорения доступа к справочнику

2.1 Добавим команду создания нового элемента справочника **ФизическиеЛица** в панель действий раздела **Расчет заработной платы**. Для этого откроем окно Все подсистемы командой контекстного меню ветви Подсистемы дерева конфигурации и установим флаг **Видимость** напротив команды **Физические лица: Создать** в области **Панель действий. Создать**, Рисунок 4.7.

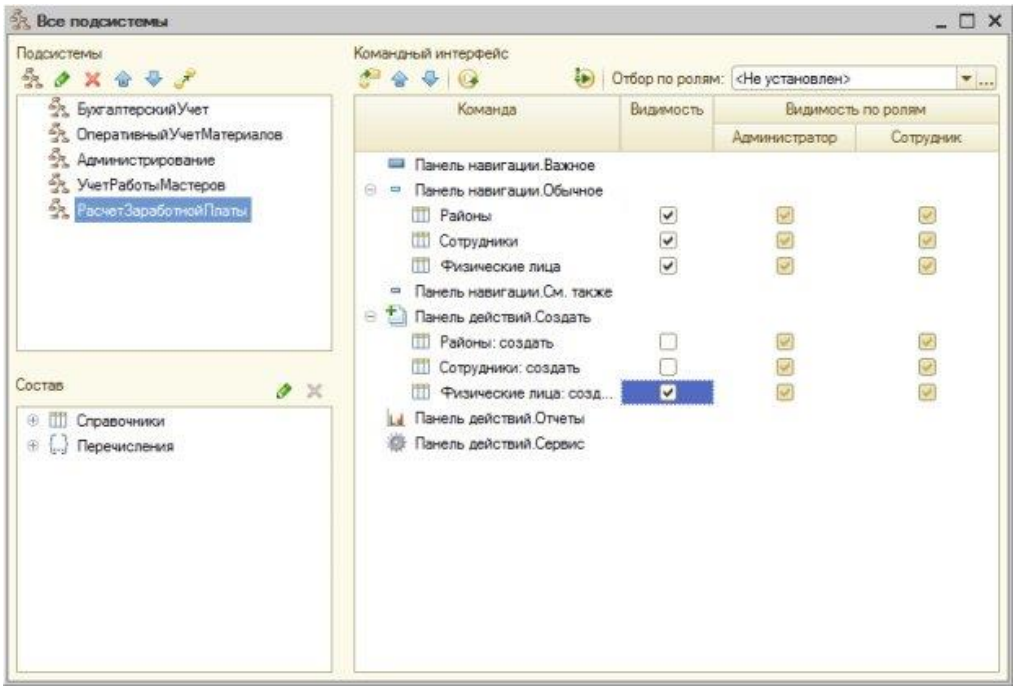


Рисунок 4.7- Настройка панели действий раздела Расчет заработной платы

2.2 Мы можем включить команду добавления нового физического лица в командный интерфейс **Рабочего стола**.

Для этого выполним команду контекстного меню корневого элемента конфигурации **Открыть командный интерфейс рабочего стола**

Выделим в поле **Доступные команды** команду **Физические лица: создать**, в поле состава командного интерфейса – **команду Панель действий.Создать** и нажмем на кнопку со значком ">" (Добавить команду на рабочий стол), которая находится между полями, после чего установим флаг Видимость для добавленной команды, рисунок 4.8.

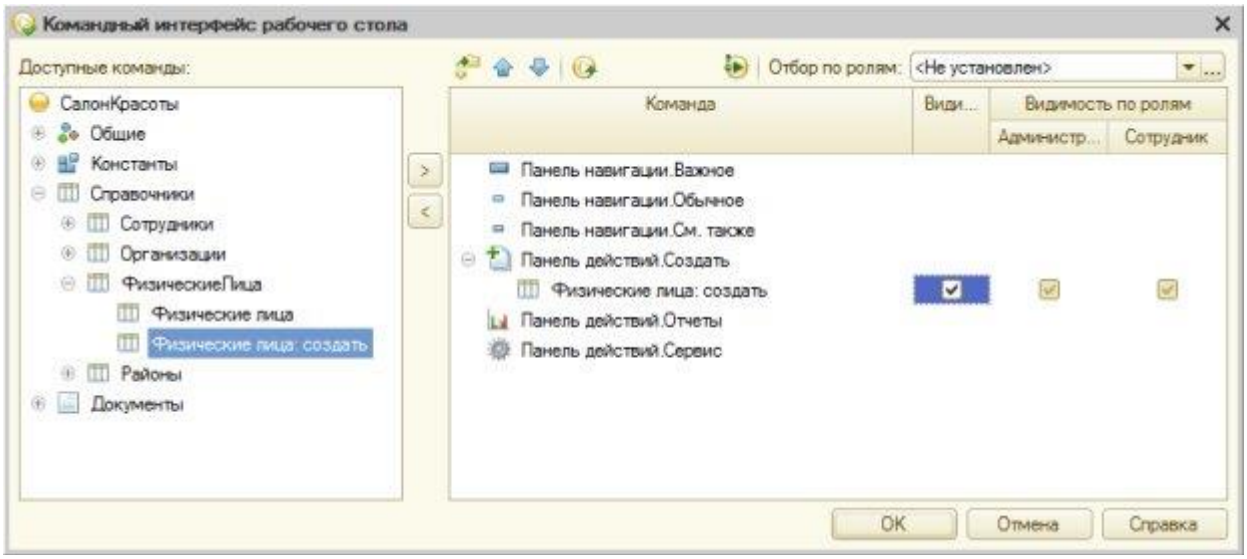


Рисунок 4.8- Настройка панели действий Рабочего стола

Теперь,(Рисунок 4.9.), команда для быстрого создания элементов справочника **ФизическиеЛица** добавлена в панель действий **Рабочего стола**, аналогичная команда появилась в разделе **Расчет заработной платы**.

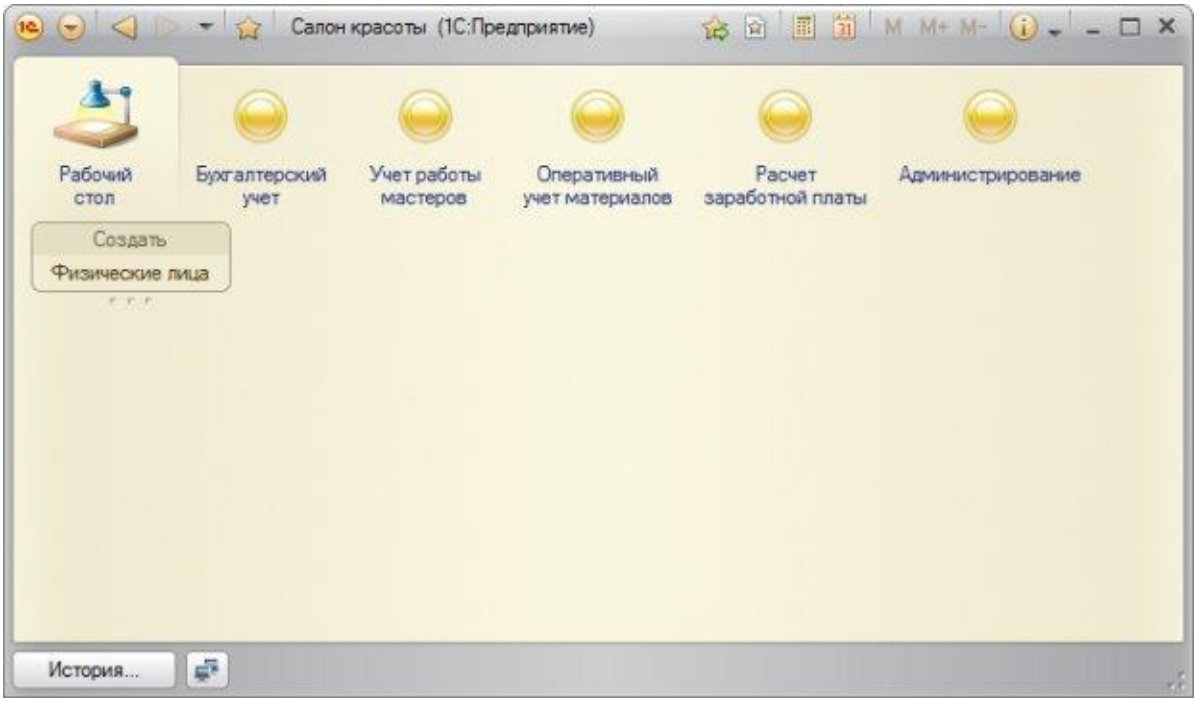


Рисунок 4.9- Новая команда в панели действий рабочего стола

Вывод

В этой работе мы создали справочники Организации и ФизическиеЛица. Для справочника ФизическиеЛица мы реализовали программное заполнение реквизита на основе других реквизитов, познакомились с объектом СообщениеПользователю, который позволяет выводить сообщения в привязке к элементам управления. Так же мы рассмотрели основные составные части редактора управляемых форм и настроили командный интерфейс для ускорения доступа пользователя к часто используемой функциональности справочника ФизическиеЛица.

Практическая работа №5 Создание объекта конфигурации «План видов характеристик»

Цель: научиться работе с константами, основам клиент-серверного программирования в 1С:Предприятие 8.2, а так же использованию общих реквизитов.

ХОД РАБОТЫ

1. Создадим новую константу (Рисунок 5.1.), заполним ее параметры следующим образом:

Имя: ТекстСообщения

Тип: Строка

Длина: 50

2. Включим константу в состав подсистемы **УчетРаботыМастеров**. Предполагается, что данная константа будет использоваться для показа сообщения пользователям, входящим в систему.

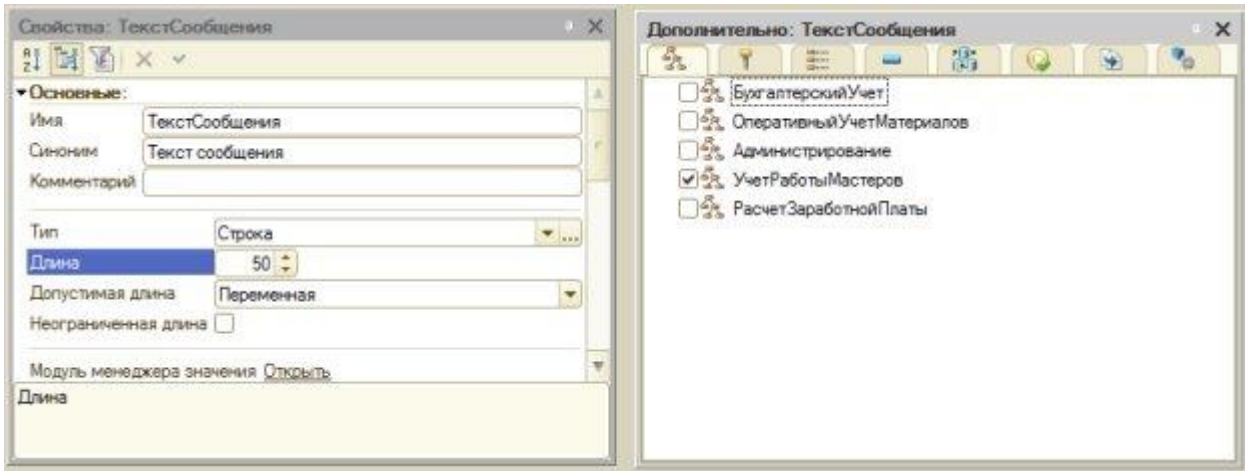


Рисунок 5.1- Настройка параметров новой константы

3. Посмотрим, как включение константы в подсистему **УчетРаботыМастеров**, отразится на интерфейсе нашего приложения в режиме 1С:Предприятие. Видно, Рисунок 5.2., что в разделе **Учет работы мастеров**, под панелью разделов, появилась еще одна панель. Она называется **панелью действий**. В панель действий автоматически включаются команды, разбитые на группы – **Сервис, Создать, Отчеты**. Группы в панели действий можно создавать и самостоятельно. В нашем случае в панели действий видна группа **Сервис**, содержащая команду для работы с только что созданной константой.

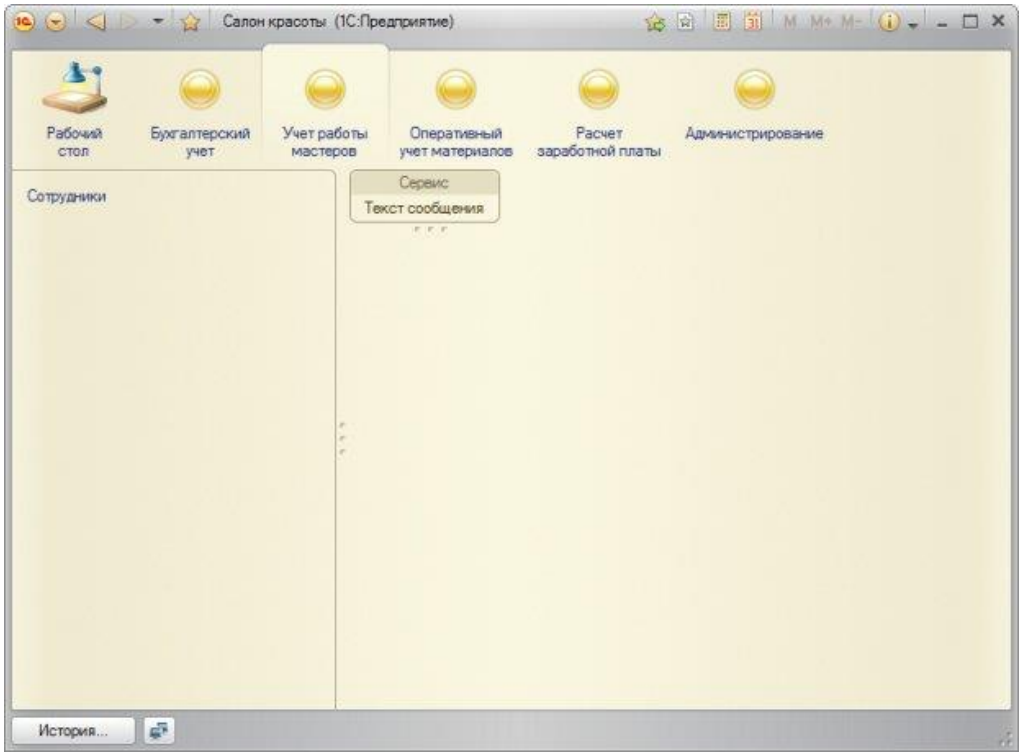


Рисунок 5.2- Константа в панели действий в разделе Учет работы мастеров

В левой части окна программы можно видеть еще одну панель – она называется **панелью навигации**. Сейчас она отображает ссылку для доступа к справочнику **Сотрудники**, который мы создавали в предыдущей лекции. Свободная часть окна – это рабочая область, в которой, например, открываются списки справочников.

4. Щелкнем по команде **Текст сообщения** в панели действий. Отобразится окно, которое позволяет нам редактировать константу **ТекстСообщения**. Введем в поле **Текст сообщения** строку "Здравствуйте, уважаемый пользователь!", Рисунок 5.3. и нажмем на кнопку **Записать и закрыть.**

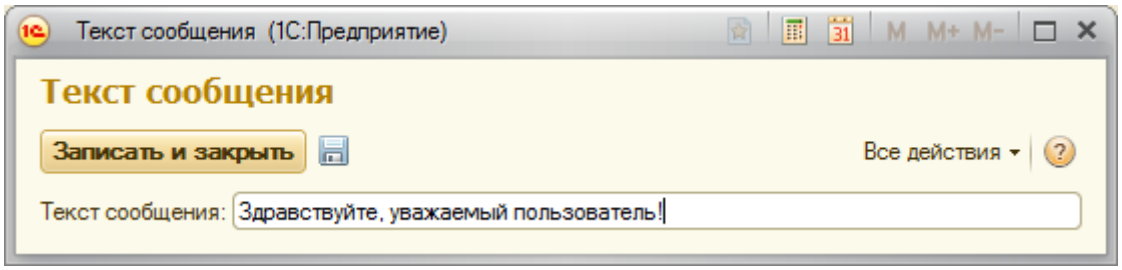


Рисунок 5.3. Форма редактирования константы Текст сообщения

Если мы не хотим сохранять внесенные изменения, можно просто закрыть окно с помощью стандартной кнопки **Закрыть**, для записи изменений без закрытия формы служит кнопка **Записать объект**.

Для того, чтобы воспользоваться дополнительными возможностями по работе с формой, можно использовать меню **Все действия**, Рисунок 5.4.

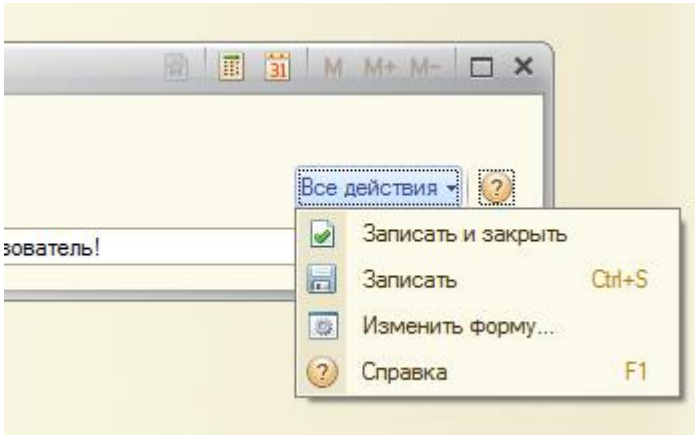


Рисунок 5.4- Меню Все действия

Форма, которую мы видим, сформирована автоматически. Однако, в режиме 1С:Предприятие мы можем вносить в нее некоторые изменения. Выполним команду **Изменить форму**, появится окно Настройка формы, Рисунок 5.5.

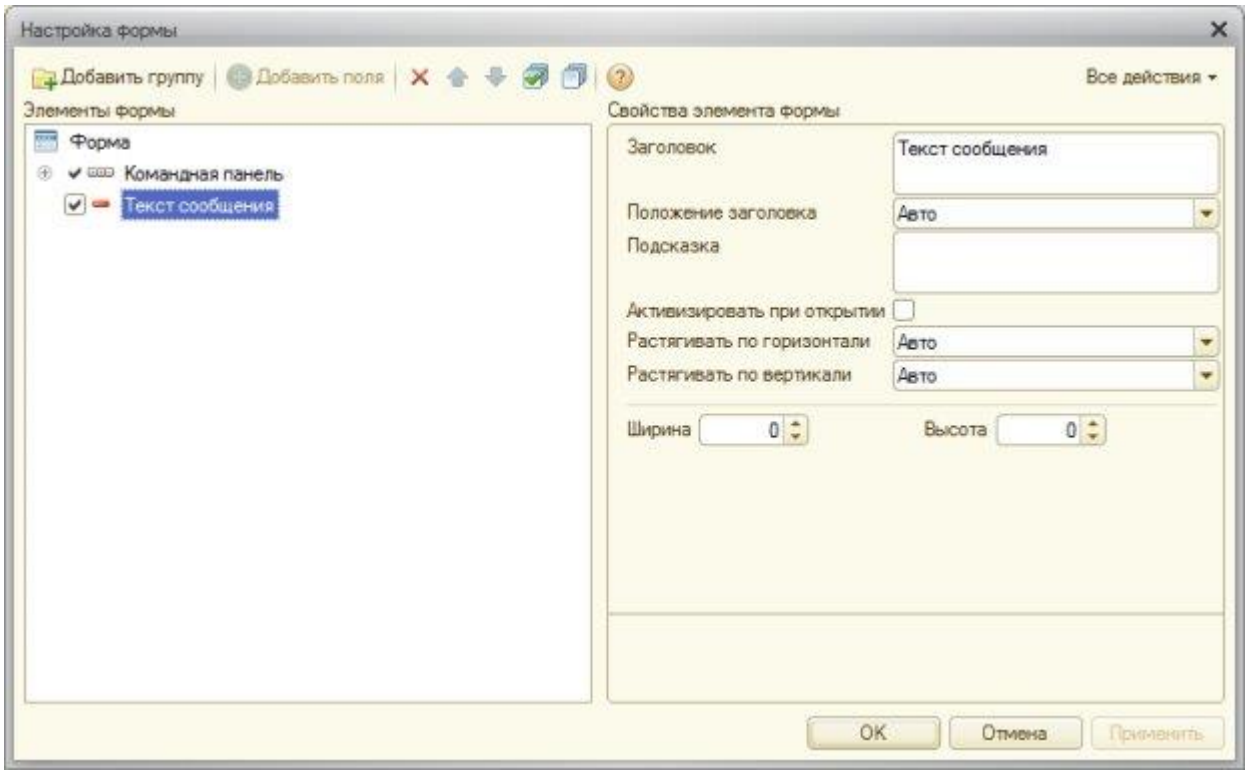


Рисунок 5.5- Окно Настройка формы

Нужно учитывать, что пользователь сможет настраивать внешний вид форм в том случае, если для него установлено право **Сохранение данных пользователя**. Это право можно настраивать, как и другие права, в роли пользователя, Рисунок 5.6. В нашем случае оно установлено.

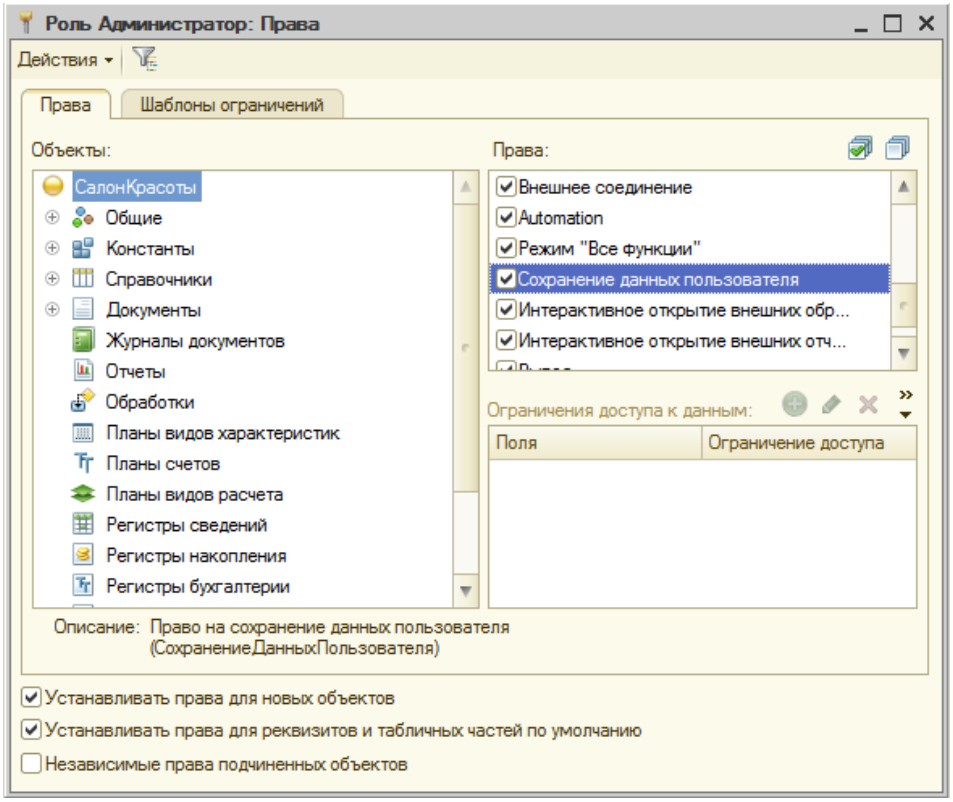


Рисунок 5.6- Право Сохранение данных пользователя

В нашем случае (Рисунок 5.5.) в группе **Элементы формы** выделен элемент **Текст сообщения**, в группе **Свойства элемента формы** мы можем настраивать его свойства.

5. Изменим свойство **Заголовок**, вместо "Текст сообщения" введем "Текст сообщения для пользователей", в итоге форма будет выглядеть так, как показано на Рисунок 5.7.

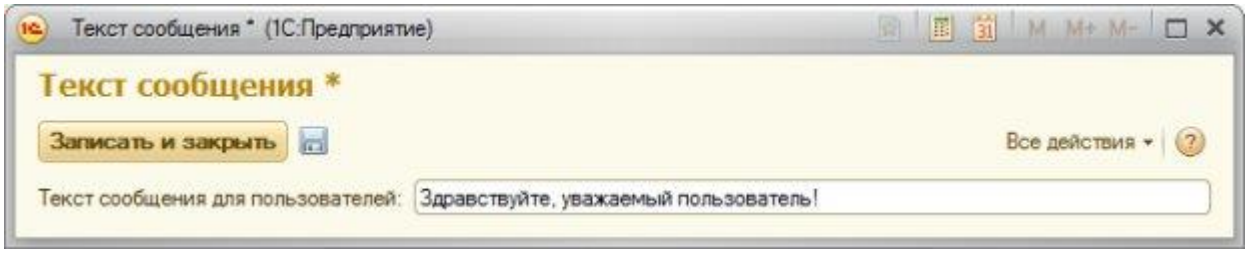


Рисунок 5.7- Отредактированный заголовок объекта

6. Перейдем в режим конфигурирования, создадим еще одну константу (она пригодится нам позже):

Имя: ПрефиксНомера

Тип: Строка

Длина: 2

Включим эту константу в подсистему **Администрирование**. В режиме 1С:Предприятие доступ к этой константе будет организован в группе **Сервис** панели действий раздела**Администрирование**. Кроме того, мы можем организовать доступ к константам из других мест нашего приложения. Мы можем самостоятельно включить команду для вызова

формы просмотра и редактирования константы, отредактировав командный интерфейс, можем так же создать специальную форму, называемую **формой констант**.

7. Создадим форму констант

7.1 Для создания формы констант нужно вызвать контекстное меню ветви **Константы** дерева конфигурации и выбрать в нем команду **Создать форму констант**. В появившемся окне **Конструктор общих форм**, Рисунок 5.8., нужно оставить тип формы в значении **Форма констант**, при необходимости заполнить другие поля и нажать на кнопку Далее.

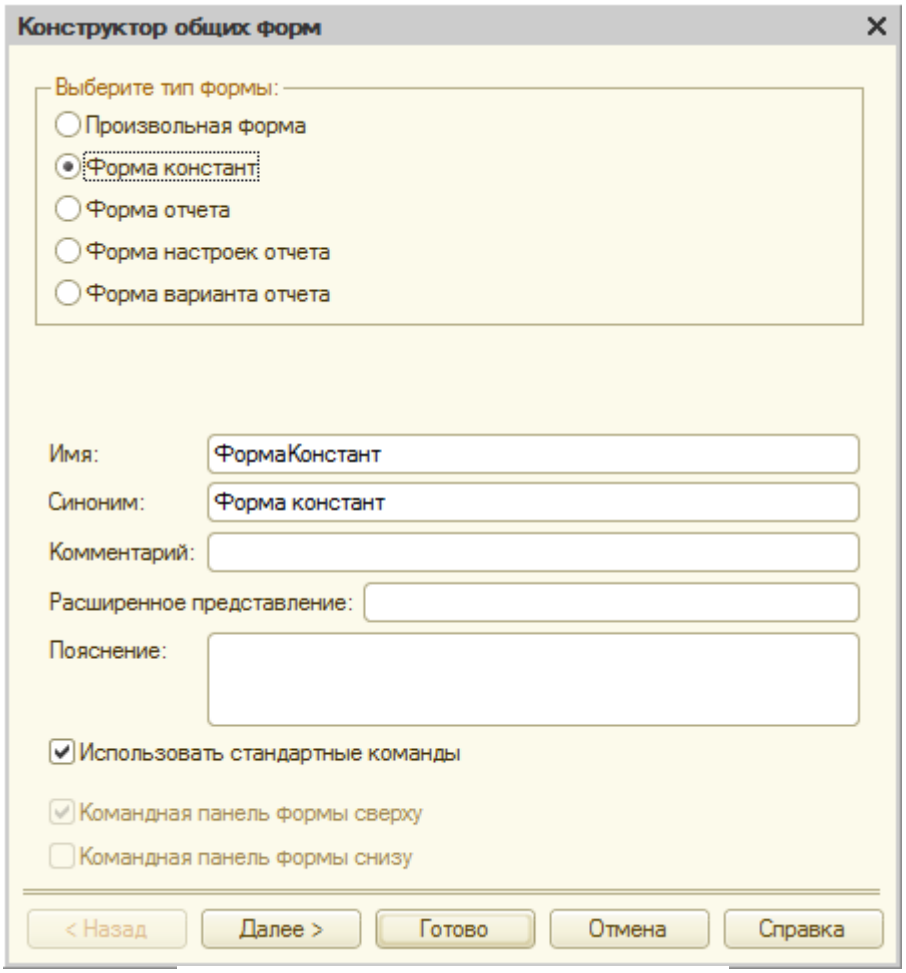


Рисунок 5.8- Конструктор общих форм

7.2 В появившемся окне, Рисунок 5.9. можно настроить состав формы констант, в нашем случае нас устраивает то, что в нее включены обе созданные в конфигурации константы, поэтому нажмем на кнопку **Готово**.

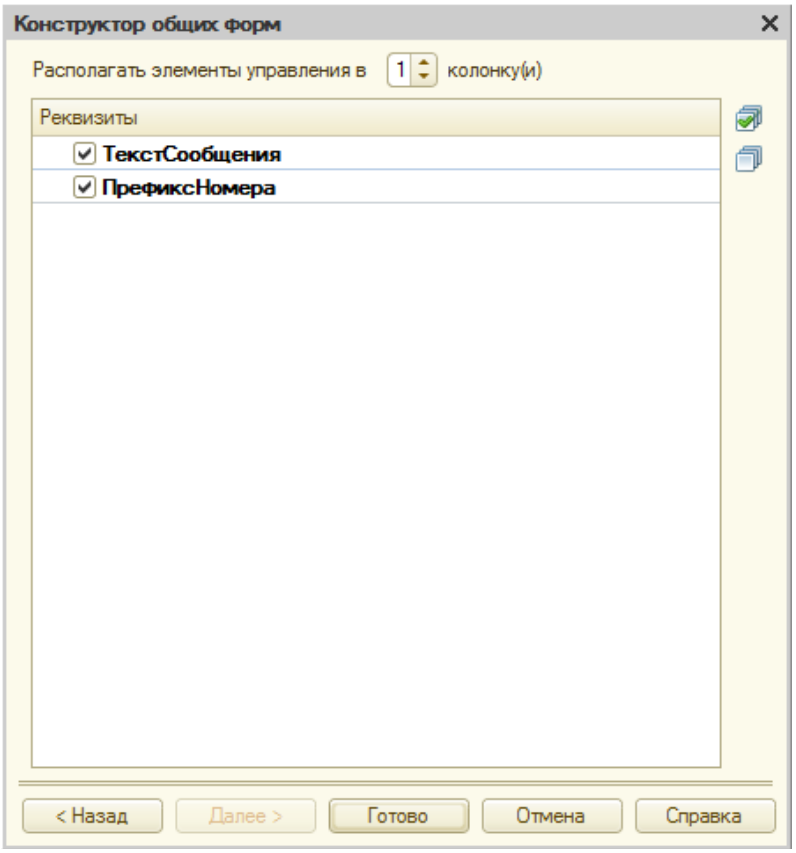


Рисунок 5.9- Конструктор общих форм, состав формы констант

7.3 В ветви **Общие формы** появится новая форма с именем **ФормаКонстант**, будет открыто окно редактирования формы, Рисунок 5.10.

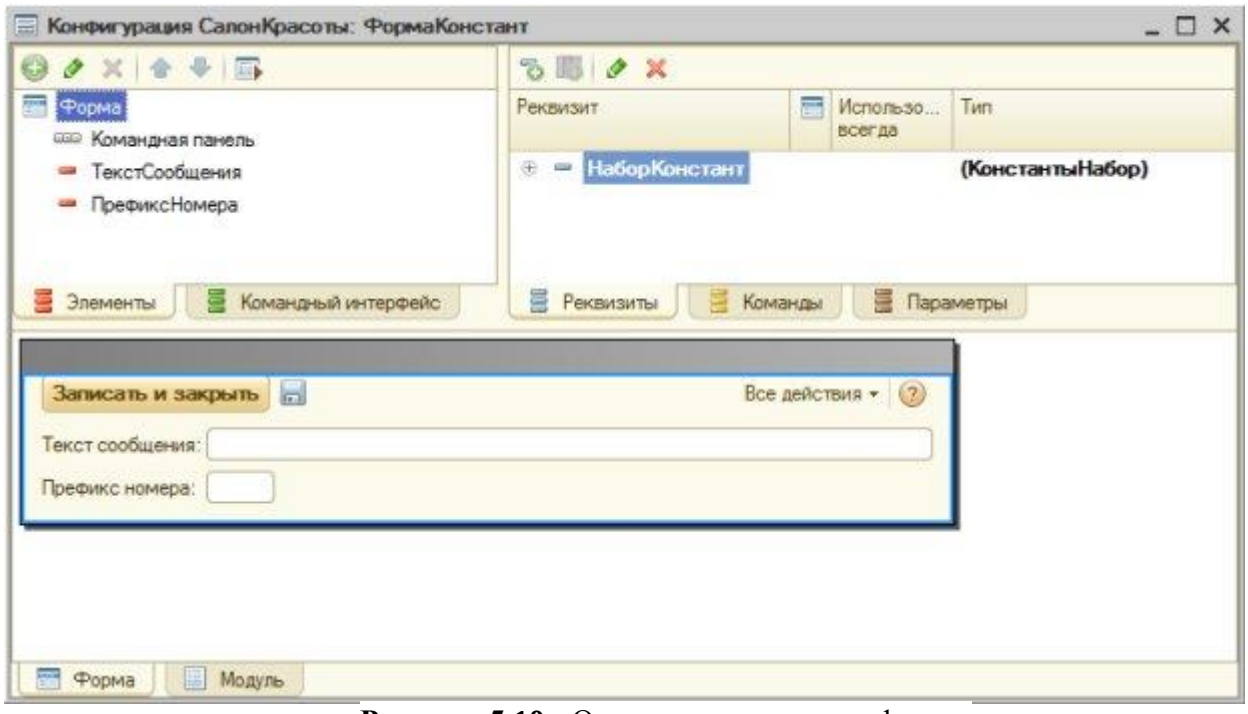


Рисунок 5.10- Окно редактирования формы

7.4 Форму констант так же нужно включить в одну из подсистем. Включим ее в подсистему **Администрирование**, посмотрим, что у нас получилось, Рисунок 5.11.

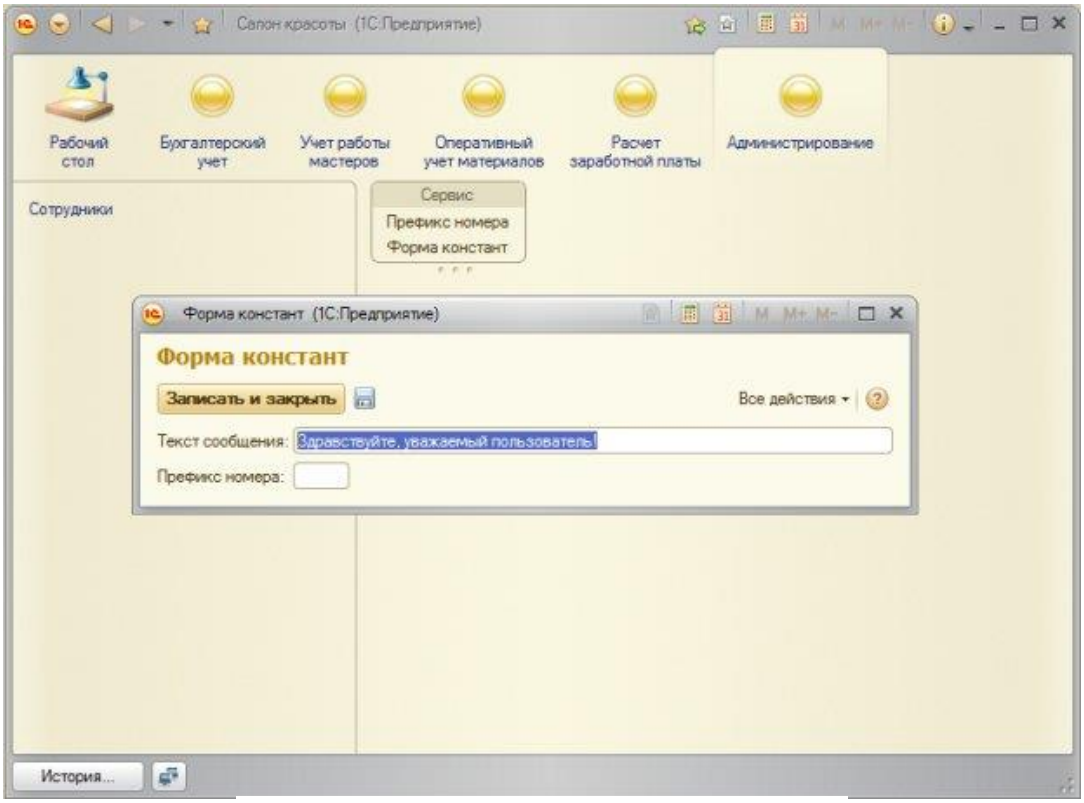


Рисунок 5.11- Окно редактирования формы

7.5 Мы видим, что форма констант доступна в группе **Сервис** панели действий раздела **Администрирование**. В текущей ситуации наличие в той же группе команды вызова окна константы **Префикс номера** может показаться избыточным. Для того чтобы убрать эту команду из панели действий, нам понадобится отредактировать командный интерфейс. Для этого мы можем выполнить команду **Открыть командный интерфейс** подсистемы **Администрирование** и в появившемся окне, Рисунок 5.12., снять флаг **Видимость** для команды **Префикс номера** группы **Сервис** панели действий.

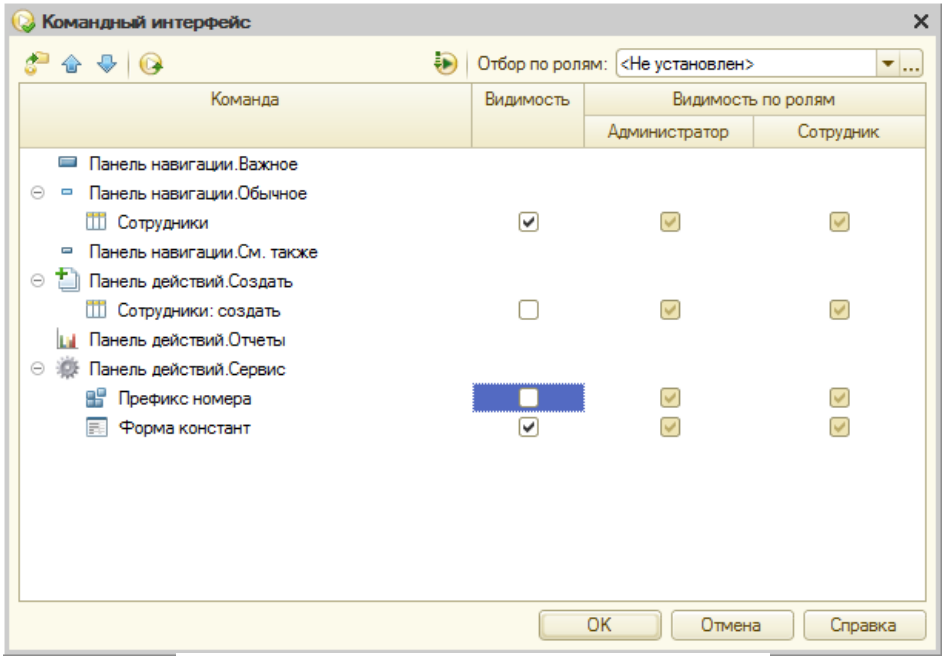


Рисунок 5.12- Настройка панели действий

Теперь при запуске в режиме 1С:Предприятие ненужная команда отображаться не будет.

7.6 Выше мы создавали константу **Текст сообщения**, предполагая выводить заданный в ней текст в качестве сообщения для пользователей, входящих в систему. Реализуем эту функциональность. Для этого нам понадобится написать код в модуле управляемого приложения. Для того, чтобы открыть этот модуль, нужно воспользоваться командой **Открыть модуль управляемого приложения** корневого элемента конфигурации. Для этого модуля предусмотрено несколько стандартных обработчиков событий, которые можно найти в панели инструментов **Модуль**, Рисунок 5.13. Нас интересует обработчик **ПриНачалеРаботыСистемы**.

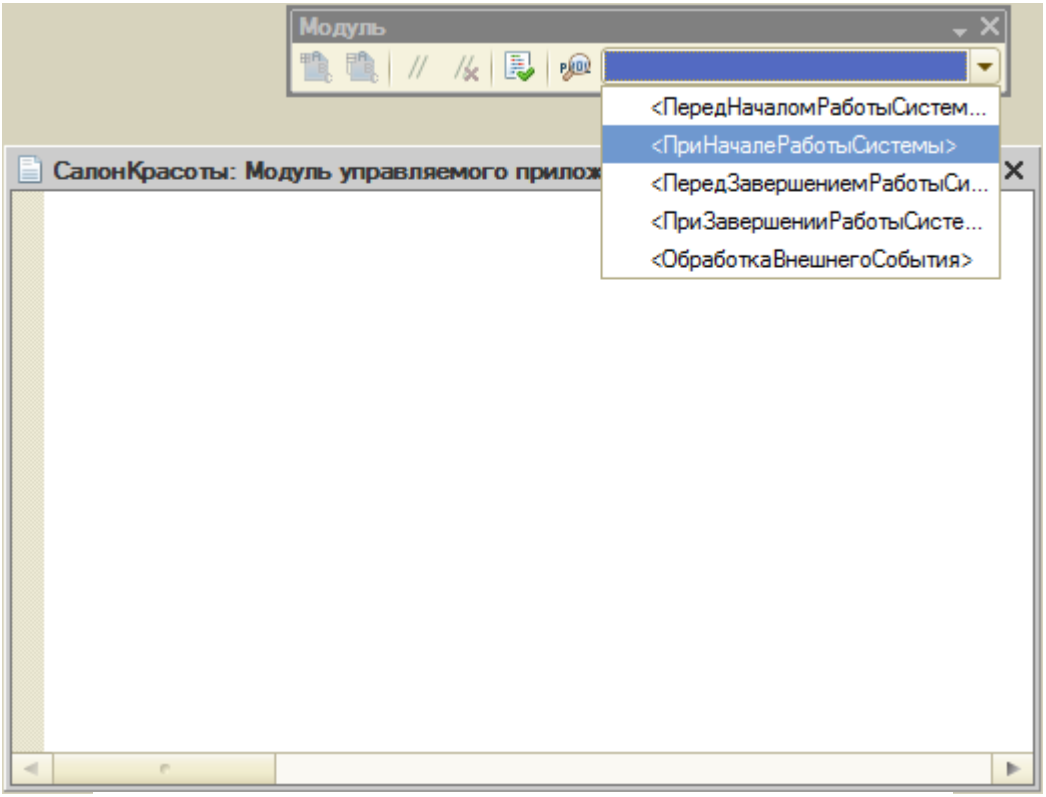


Рисунок 5.13- Выбор обработчика ПриНачалеРаботыСистемы

7.7 Создадим новый общий модуль (в ветви **Общие модули** дерева конфигурации), назовем его **СерверныеФункции**. Проследим за тем, чтобы в его свойствах были установлены флаги **Сервер** и **Вызов сервера**, Рисунок 5.14.

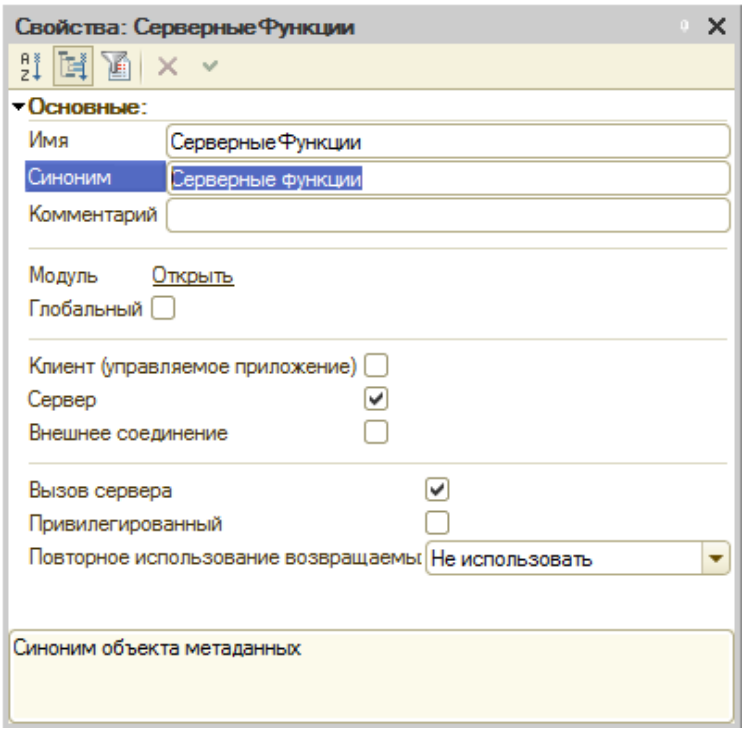


Рисунок 5.14- Общий модуль СерверныеФункции, свойства

7.8 Откроем редактор кода для кода модуля (например, двойным щелчком по модулю в дереве конфигурации) и введем следующий код, Рисунок 5.15.:

```
//Экспортная функция для вызова из других модулей
Функция ПолучитьКонстанту() Экспорт
   //Возвращаем полученное значение константы
   Возврат(Константы.ТекстСообщения.Получить());
КонецФункции
```

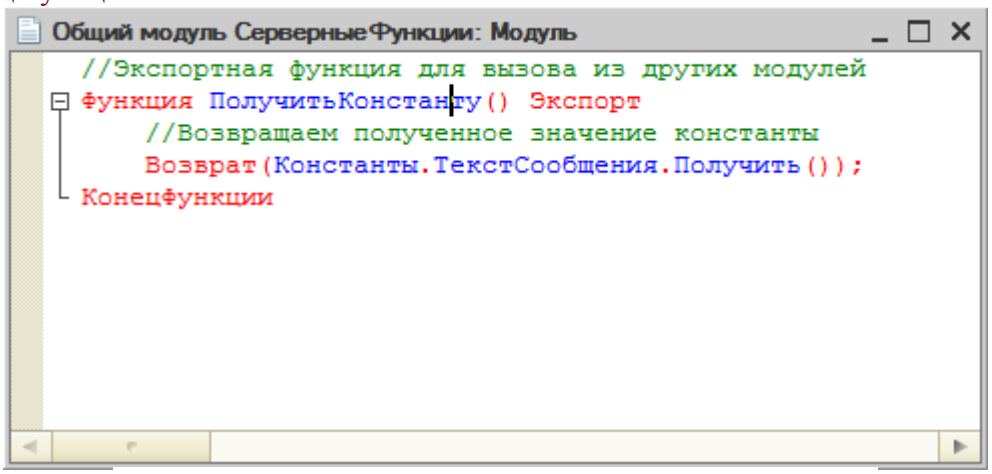


Рисунок 5.15- Общий модуль СерверныеФункции, код

7.9 Теперь нам нужно вызвать эту функцию в подходящем месте кода обработчика события **ПриНачалеРаботыСистемы** в модуле управляемого приложения. Например, это можно сделать так:

```
Процедура ПриНачалеРаботыСистемы()
   //Выводим сообщение пользователю
   Сообщить(СерверныеФункции.ПолучитьКонстанту());
```

КонецПроцедуры

В результате при входе в систему мы получим сообщение следующего вида, Рисунок 5.16.

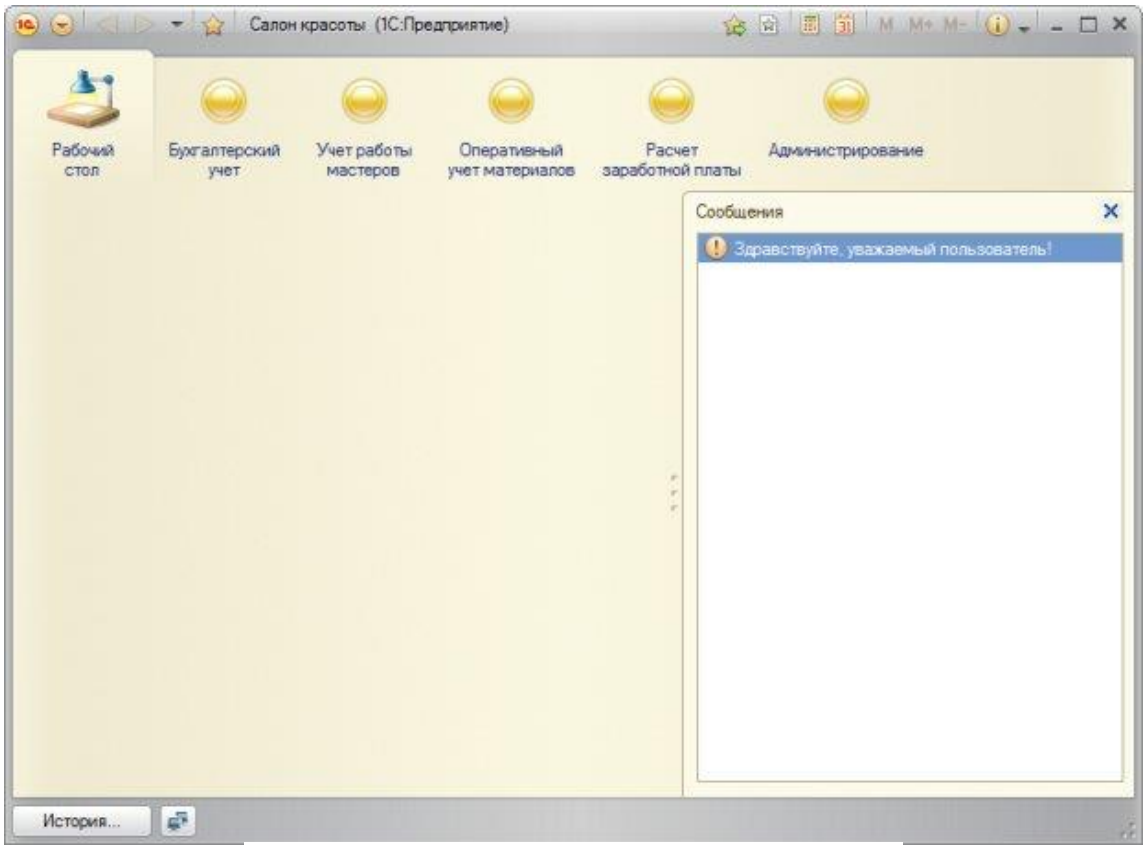


Рисунок 5.16- Вывод сообщения пользователю

Обратите внимание на то, что сообщение выводится в область **Сообщения** основного рабочего окна. Если сообщение вызвано из модуля какого-либо отдельного окна, например, из модуля формы констант, которая создана ранее, то, по умолчанию, сообщение будет выведено в этом окне.

7.10 Откроем окно редактирования формы констант (**Общие формы** > **ФормаКонстант**), перейдем на вкладку **Модуль**, на панели инструментов **Модуль** выберем стандартный обработчик события **ПриОткрытии**, отредактируем тело обработчика, чтобы оно приняло следующий вид, Рисунок 5.17.:

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
   Сообщить("Вы открыли форму констант!");
КонецПроцедуры
```

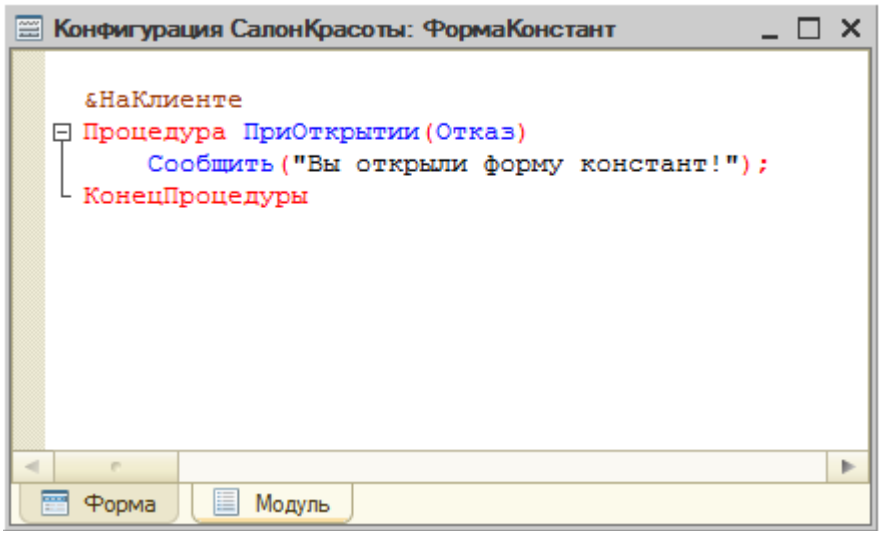


Рисунок 5.17- Вывод сообщения пользователю из модуля формы констант

Благодаря этому коду при открытии формы констант будет появляться следующее сообщение, Рисунок 5.18.

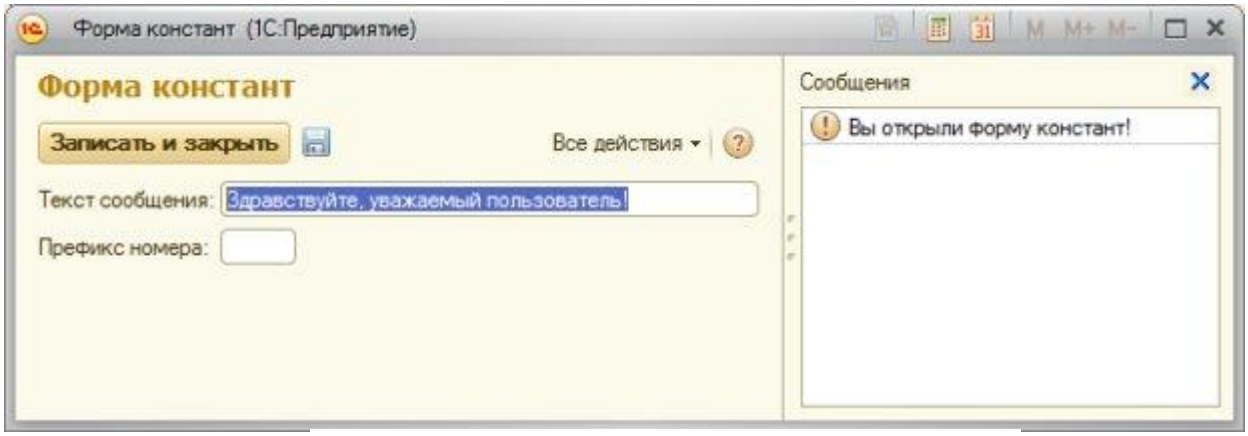


Рисунок 5.18- Вывод сообщения в форму констант

8. Попытаемся в нашем модуле формы вывести в окно сообщения значение константы. Для этого мы можем добавить в модуль функцию, возвращающую значение константы, которая должна выполняться в контексте сервера. Например, это можно сделать одним из следующих способов – ниже приведена дополненная процедура ПриОткрытии и еще пара процедур, заданных в коде модуля формы:

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
   Сообщить("Вы открыли форму констант!");
   Сообщить(ПолучитьКонстанту()+" - из функции модуля формы без директивы");
   Сообщить(СерверныеФункции.ПолучитьКонстанту()+" - из общего модуля");
   Сообщить(ПолучитьКонстантуНаСервере()+" - из функции модуля формы с директивой &НаСервере");
 КонецПроцедуры

//По умолчанию функция считается серверной
 Функция ПолучитьКонстанту()
```

```
    //Возвращаем полученное значение константы
    Возврат(Константы.ТекстСообщения.Получить());
КонецФункции

//Директива компиляции задана явно
&НаСервере
 Функция ПолучитьКонстантуНаСервере()
    //Возвращаем полученное значение константы
    Возврат(Константы.ТекстСообщения.Получить());
КонецФункции
```

Здесь мы создали пару функций – одну назвали **ПолучитьКонстанту()**, при ее описании директиву компиляции мы не указывали. Вторую назвали**ПолучитьКонстантуНаСервере()** – при ее описании была указана директива **&НаСервере**. Мы вызвали эти функции для вывода сообщения в клиентской процедуре**ПриОткрытии()**. У нас уже есть серверная функция в общем модуле **СерверныеФункции** – здесь показан пример ее использования, в подобном случае, возникшем при реальной разработке, если действия, которые выполняются в серверной функции модуля формы, совпадают с действиями функции, описанной в общем модуле, можно и даже нужно пользоваться функцией общего модуля.

На рисунке 5.19. вы можете видеть вывод сообщений, выполненный вышеприведенным кодом.

Рисунок 5.19- Вывод сообщения в форму констант, разные варианты работы с серверными данными

9. Создадим общие реквизиты.

9.1 В нашем учебном примере мы собираемся вести в базе данных учет по нескольким организациям. Для этого нам понадобится, чтобы все объекты конфигурации, для которых уместен данный реквизит, содержали бы реквизит **Организация**, который содержит ссылку на организацию. Например, каждый документ будет оформляться от лица определенной организации, каждый элемент справочника будет относиться к той или иной организации, и так далее. Для того, чтобы не усложнять наши примеры, мы не будем в дальнейших лекциях курса развивать тему многофирменного учета в одной базе данных. Однако, в любом случае, общие реквизиты позволяют снизить трудоемкость разработки.

9.2 Второй реквизит, который предназначен для документов, будет использоваться для ввода комментариев к документу.

9.3 Прежде чем продолжать работу над общими реквизитами, создадим следующие объекты конфигурации, не настраивая их дополнительных свойств – справочник с именем**Организация**, и документ с именем **ПоступлениеМатериалов**. Включим их в подсистему **ОперативныйУчетМатериалов**.

Создадим новый **общий реквизит** со следующими параметрами, Рисунок 5.20.:

Имя: Комментарий

Тип: Строка, длина 50

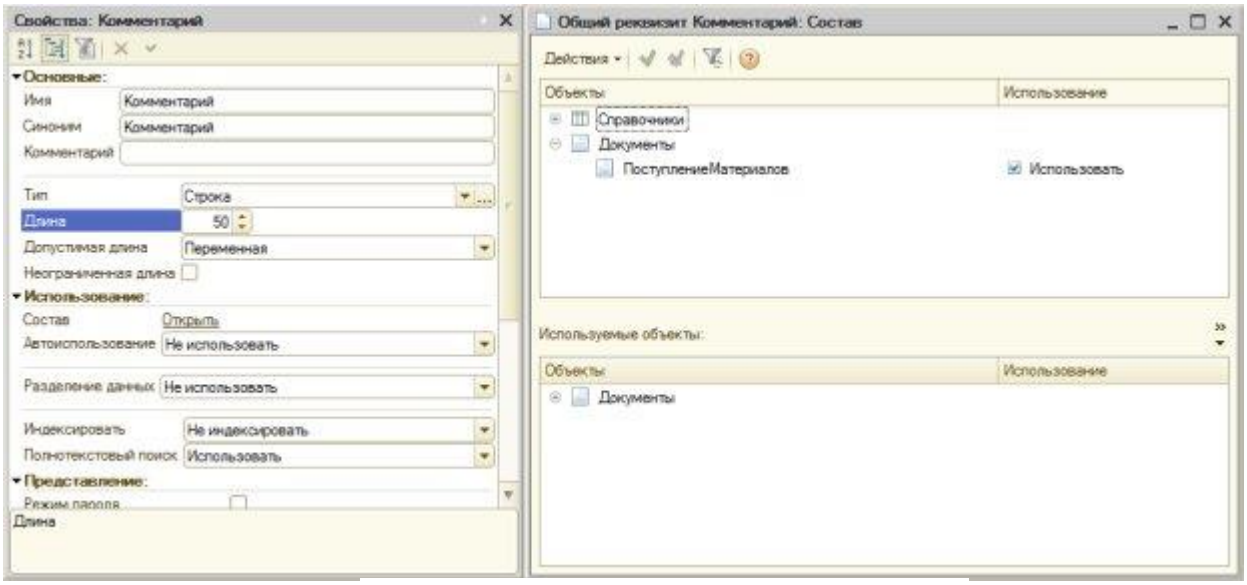


Рисунок 5.20- Настройка общего реквизита

9.4 Обратите внимание на параметр **Автоиспользование**. В данном случае мы оставляем его в значении по умолчанию – **Не использовать**. То есть – состав общего реквизита мы будем настраивать вручную. Этот общий реквизит мы планируем добавить ко всем документам, поэтому найдем свойство **Состав**, нажмем на ссылку **Открыть**, в появившемся окне выберем вариант **Использовать** для документа **ПоступлениеМатериалов**. При создании других документов мы сможем самостоятельно включать их в состав общего реквизита. Быстро проверить состав используемых объектов общего реквизита можно в нижней части окна настройки состава.

9.5 Создадим **второй** общий реквизит:

Имя: Организация

Тип: СправочникСсылка.Организации

Автоиспользование: Использовать

Этот реквизит мы планируем добавить ко всем объектам, допускающим использование общих реквизитов, за исключением справочника **Организации** и некоторых других. Перейдем в окно настройки состава общего реквизита и установим свойство **Использование** у справочника **Организации** в значение **Не использовать**, Рисунок 5.21.

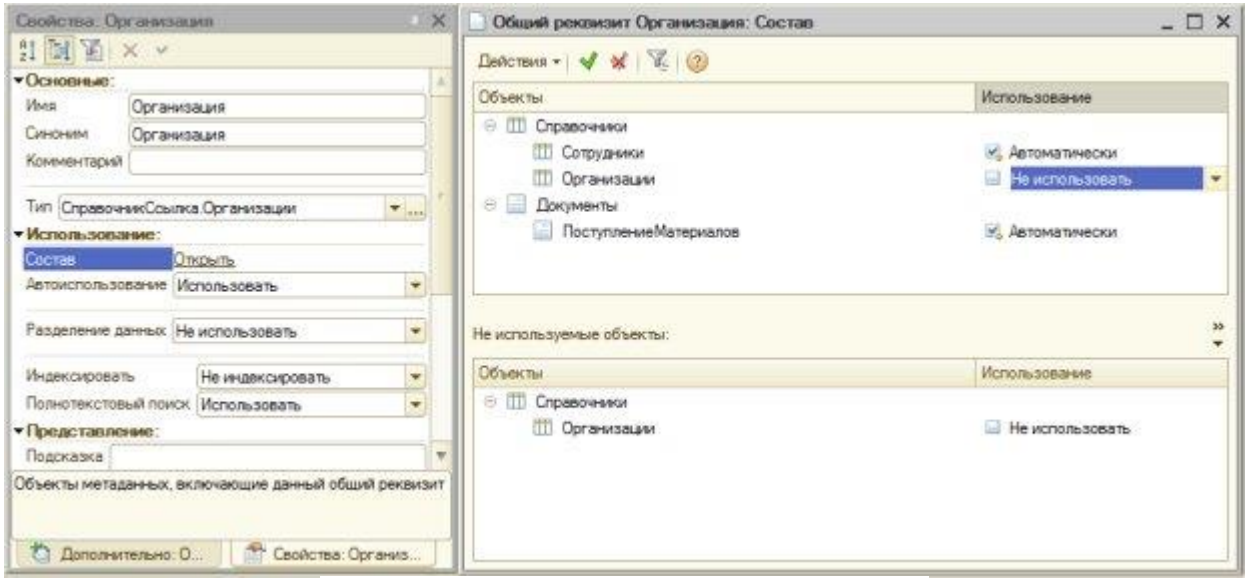


Рисунок 5.21- Настройка общего реквизита

9.6 Откроем нашу конфигурацию в режиме 1С:Предприятие и посмотрим, как выглядит документ **ПоступлениеМатериалов** и справочники **Организации** и **Сотрудники**.

9.7 Для начала перейдем на вкладку **Оперативный учет материалов**. Обратите внимание на то, что в панель навигации раздела были автоматически добавлены ссылки для доступа к только что созданному справочнику **Организации** и к документу **Поступление материалов**. Щелкнем по ссылке **Организации**. В рабочей области окна появится список справочника. На данный момент он пуст, так как мы пока не заполняли справочник организациями, по которым будет вестись учет в базе. Щелкнем по кнопке **Создать**, которая расположена на командной панели списка – появится отдельное окно для заполнения свойств элемента справочника, Рисунок 5.22. Можно отметить, что помимо стандартных реквизитов (**Наименование**, **Код**) данный справочник не содержит ничего другого – это неудивительно, мы исключили его из состава общего реквизита **Организация**.

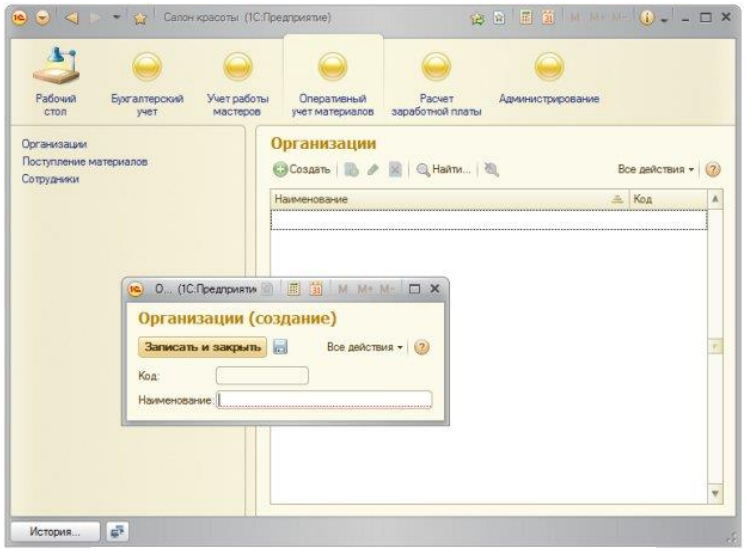


Рисунок 5.22- Справочник Организации

9.8 Теперь откроем список справочника **Сотрудники** и нажмем на кнопку **Добавить**. Общий реквизит **Организация** у данного справочника присутствует, Рисунок 5.23.

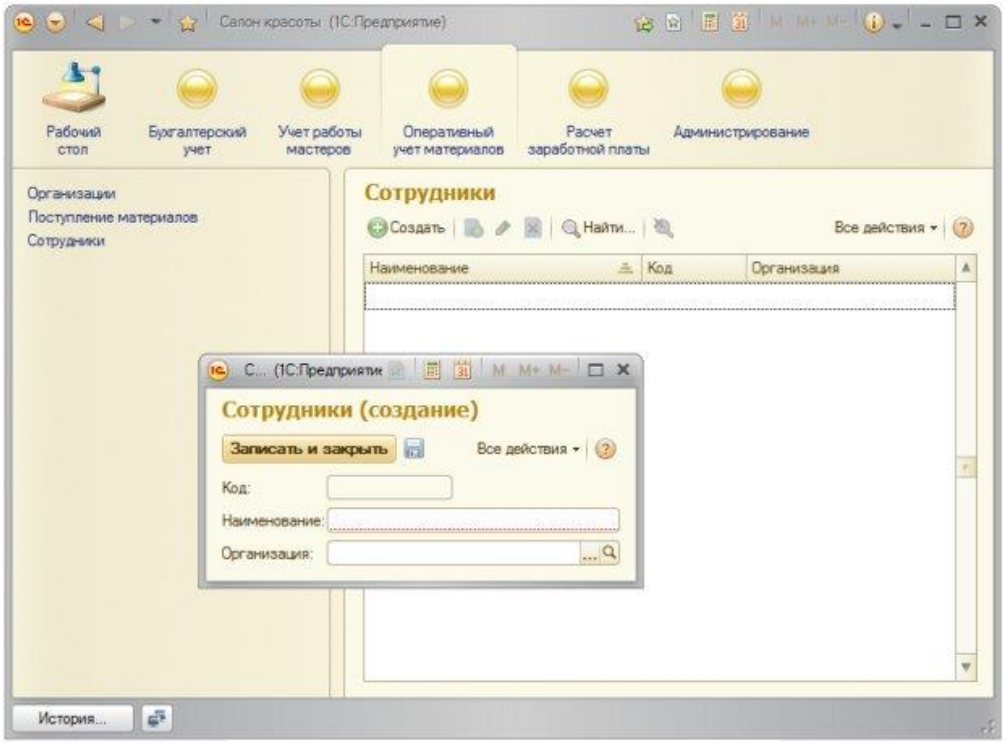


Рисунок 5.23- Справочник Сотрудники

9.9 Откроем теперь окно создания документа **ПоступлениеМатериалов**. Здесь мы видим два общих реквизита – **Комментарий** и **Организация**, Рисунок 5.24.

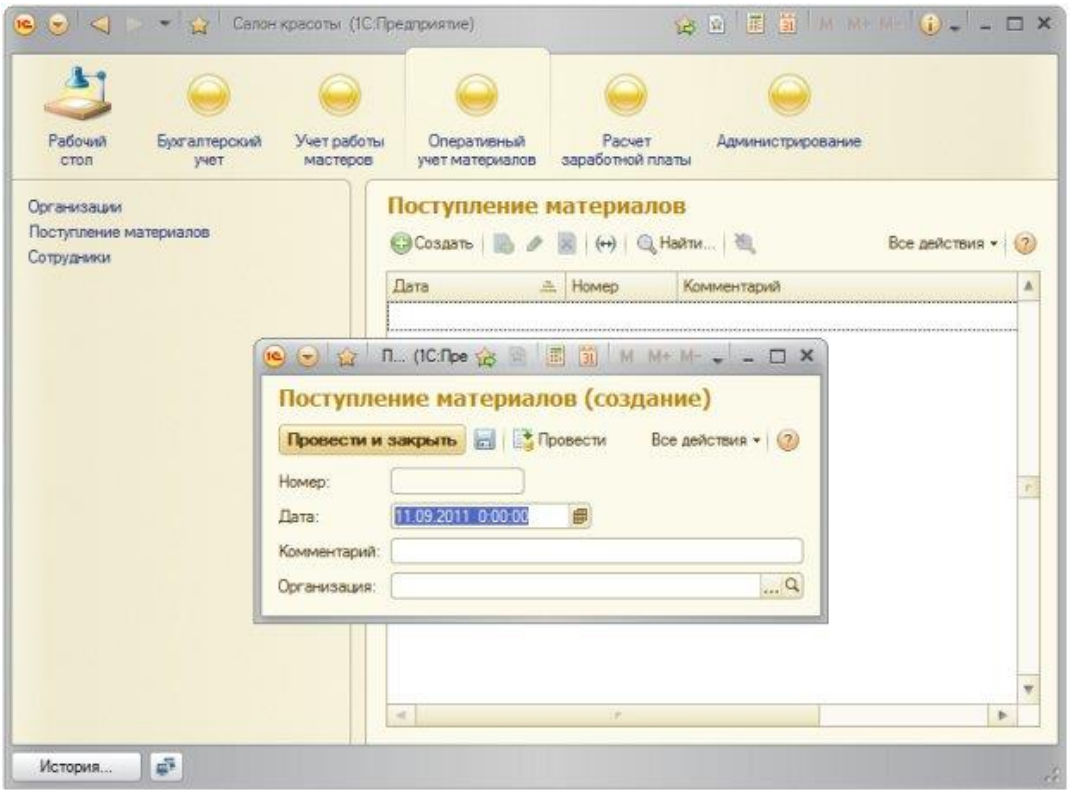


Рисунок 5.24- Документ ПоступлениеМатериалов

Вывод

В этой работе мы научились создавать константы и программно работать с ними. Так же здесь мы начали обсуждение вопросов клиент-серверного программирования, в частности, использовали директивы компиляции **&НаСервере, &НаКлиенте**. Мы познакомились с использованием экспортных методов общих модулей, с модулем управляемого приложения, с модулем формы, рассмотрели использование общих реквизитов.

Практическая работа №6 Создание журналов документов

Создание журналов документов

Цель: научиться разработке иерархических и подчиненных справочников, реализации дополнительных программных механизмов.

ХОД РАБОТЫ:

1. Выполним работу с иерархическими справочниками.

Создадим новый справочник **Единицы измерения**, зададим следующие его параметры:

Имя: ЕдиницыИзмерения

Длина наименования: 100 символов

Подсистемы: БухгалтерскийУчет, ОперативныйУчетМатериалов

Это будет очень простой справочник, стандартный реквизит которого **Наименование** будет использоваться для хранения информации о наименовании единицы измерения.

2. Теперь создадим очередной справочник – **Номенклатура**. Зададим следующие параметры:

Имя: Номенклатура

Подсистемы: БухгалтерскийУчет, ОперативныйУчетМатериалов

На вкладке окна редактирования объекта **Иерархия**, Рисунок 6.1., установим следующие параметры:

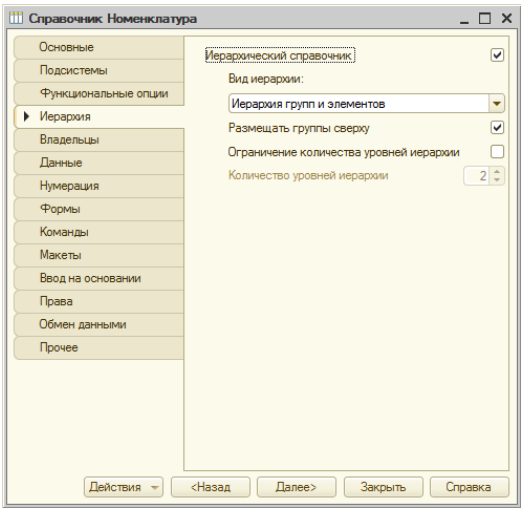


Рисунок 6.1- Настройка иерархического справочника

Иерархический справочник: Установлено

Вид иерархии: Иерархия групп и элементов.

Этот параметр может принимать значение **Иерархия элементов**. В нашем случае справочник сможет содержать отдельные элементы, собранные, в зависимости от их вида, в группы. Эту структуру можно сравнить с папками и файлами в файловой системе компьютера. Группы – это папки, отдельные элементы – это файлы.

3. На вкладке **Данные**, Рисунок 6.2., добавим следующие реквизиты:

ЕдиницаИзмерения: Тип СправочникСсылка.ЕдиницыИзмерения.

Услуга: Тип Булево, **Использование**: Для группы и элемента. Эта установка позволит задавать данный реквизит и для элементов и для групп.

Заполнять из данных заполнения: Истина

Отдельные группы нашего справочника планируется использовать для хранения исключительно услуг, и подобная установка (в частности, истинность параметра **Заполнять из данных заполнения**) позволит нам реализовать автоматический механизм заполнения данного реквизита для элементов, входящих в группы.

Длина наименования: 100

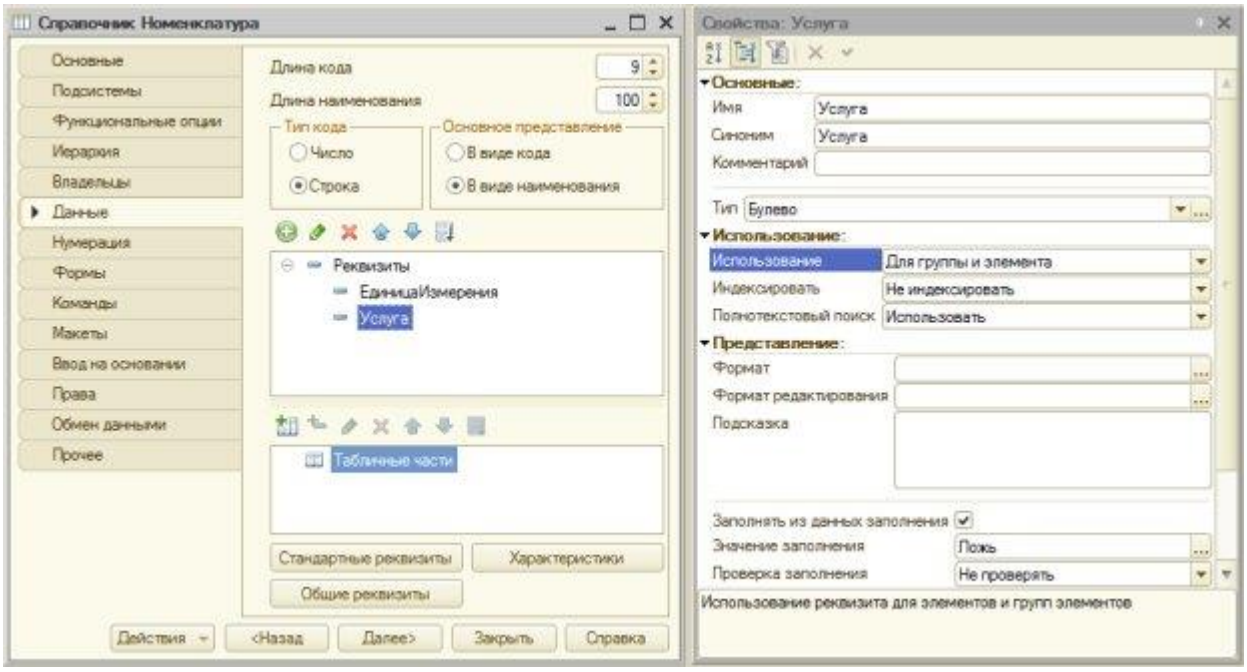


Рисунок 6.2- Состав реквизитов справочника Номенклатура

Таким образом, при создании элемента справочника мы будем задавать название элемента в стандартном реквизите **Наименование**, указывать единицу измерения, а так же, для услуг, устанавливать флаг **Услуга**, причем, установка этого флага для группы будет означать, что в ней хранятся списки услуг, а для элемента – то, что он является услугой.

4. Реализуем функцию автоматического заполнения реквизита **Услуга** для элементов, входящих в группы. Нам нужно, чтобы элемент, создаваемый в группе с установленным флагом **Услуга,** при его создании, автоматически бы получал установленный флаг **Услуга**, соответственно, если данный флаг у группы не установлен, у элемента он так же не должен быть установлен. При этом нам нужно предусмотреть ситуацию, когда элемент создается вне группы – на верхнем уровне справочника **Номенклатура**.

Для решения этой задачи мы можем воспользоваться обработчиком события **ОбработкаЗаполнения**, его процедура располагается в модуле объекта.

5. Перейдем в модуль объекта (кнопка **Модуль объекта** на закладке **Прочие** окна редактирования объекта), из списка процедур и выберем **ОбработкаЗаполнения**, Рисунок 6.3.

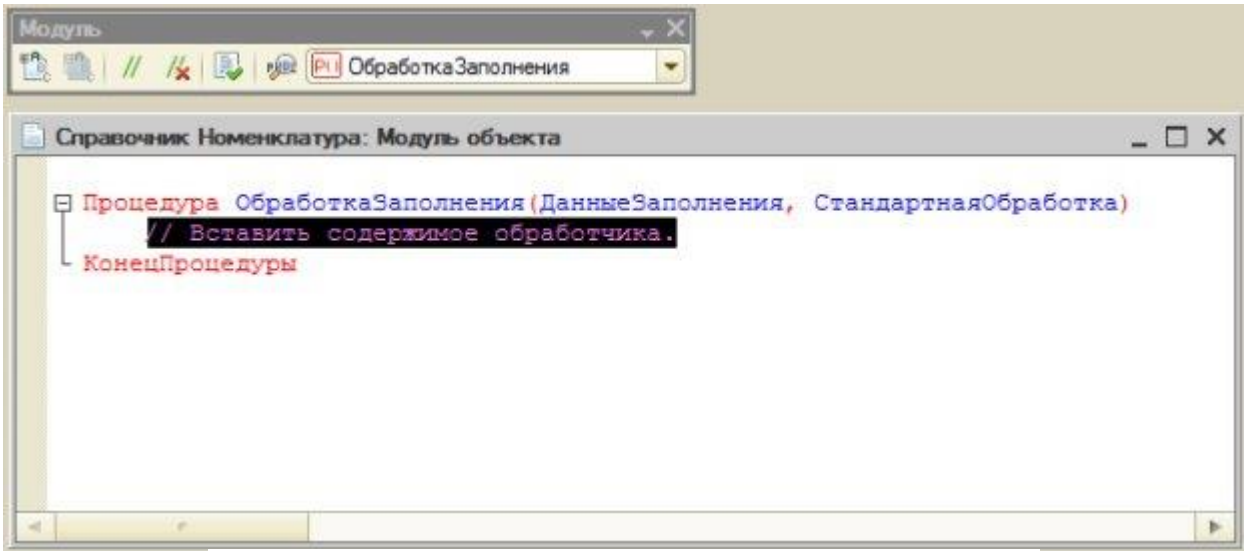


Рисунок 6.3- Процедура обработки заполнения справочника

6. В режиме 1С:Предприятие откроем справочник **Номенклатура**, создадим две группы – **Товары** – флаг **Услуги** в этой группе не устанавливаем, и **Услуги** – флаг установлен,Рисунок 6.4**.**

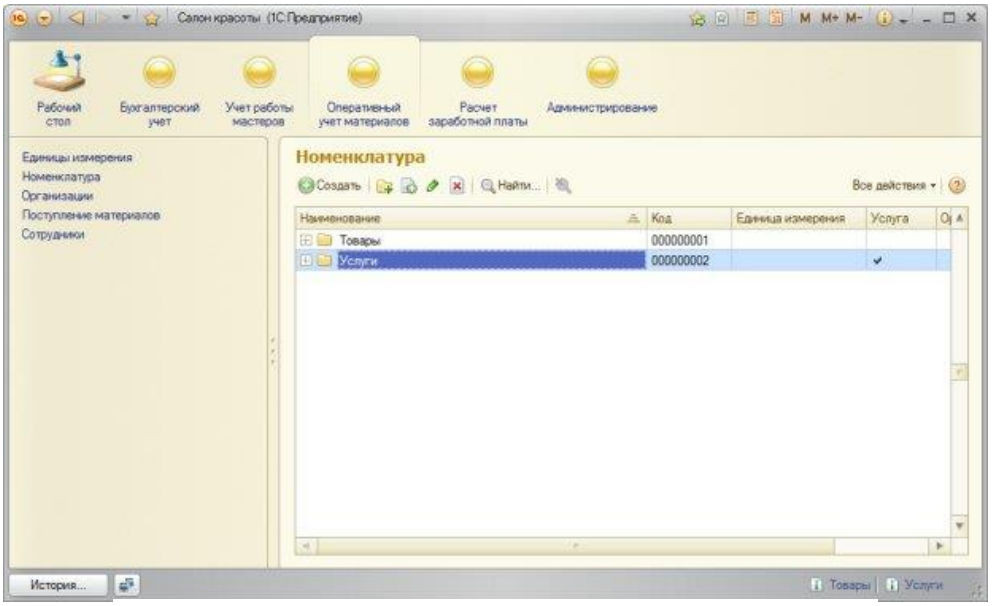


Рисунок 6.4- Две группы в справочнике Номенклатура

Процедура ОбработкаЗаполнения предусматривает автоматический механизм заполнения реквизитов на основе переданной структуры. Так как стандартный механизм нас вполне устраивает, мы можем поступить по-другому. А именно, для установки свойства **Услуга** нам нужно лишь дополнить структуру необходимой записью. Сделать это можно с помощью стандартных операций по работе со структурой. А именно, следующим образом:

```
ДанныеЗаполнения.Вставить("Услуга", ДанныеЗаполнения.Родитель.Услуга);
```

В итоге у нас получается такой код, Рисунок 6.7.

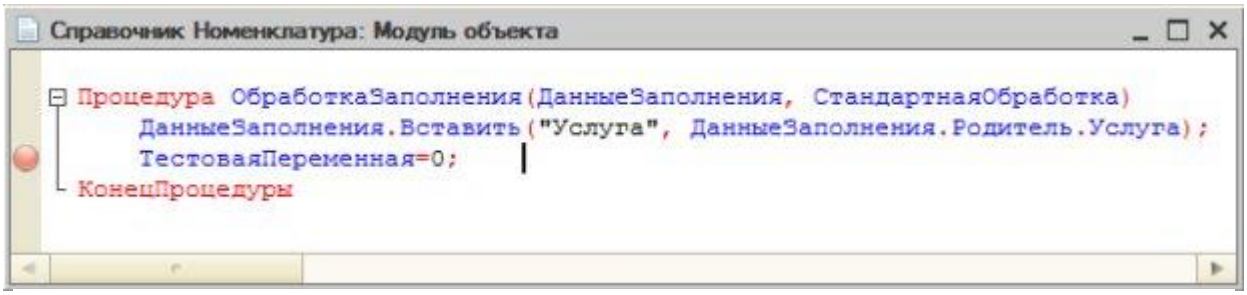


Рисунок 6.7. Заполнение реквизита Услуга на основании параметров элемента родителя

7. Опробуем решение в пользовательском режиме, можно заметить, что, во-первых, структура **ДанныеЗаполнения** действительно теперь содержит ключ **Услуга** со значением **Истина**, а так же то, что элементы, создаваемые в группе с установленным флагом **Услуга**, имеют данный реквизит в установленном положении, Рисунок 6.8.

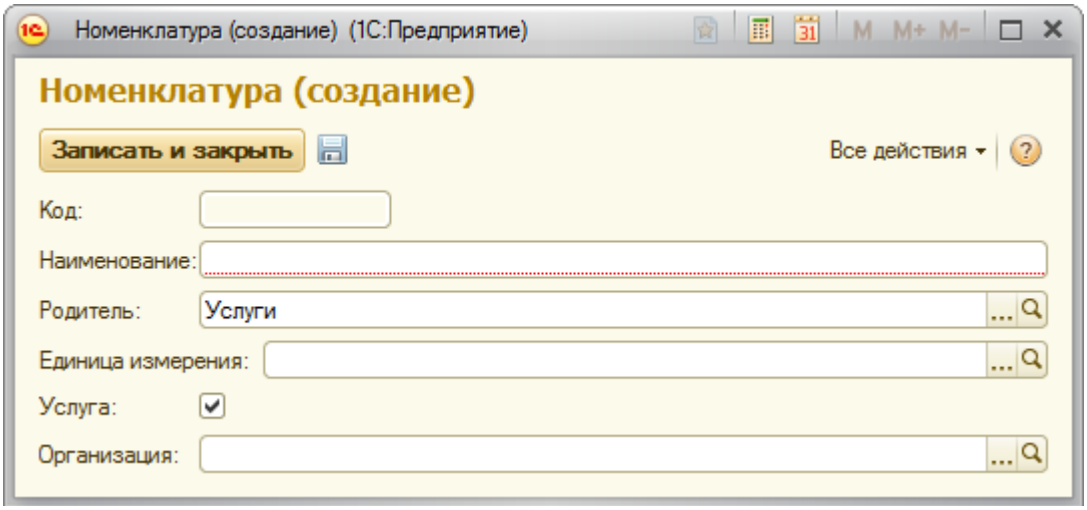


Рисунок 6.8- Результат заполнения реквизита Услуга на основании параметров элемента родителя

8. Вышеприведенные рассуждения приводят нас к следующему коду:

```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)
Если ДанныеЗаполнения<>Неопределено Тогда
 Если ДанныеЗаполнения.Свойство("Родитель") Тогда
   ДанныеЗаполнения.Вставить("Услуга", ДанныеЗаполнения.Родитель.Услуга);
 КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

В данной редакции обработчика события **ОбработкаЗаполнения** все работает верно.

Чисто теоретически (предположим, при изменении кем-либо нашего кода) возможна ситуация, когда **ДанныеЗаполнения** будут являться структурой и в этой структуре, в то же время, не будет свойства **Родитель**. Поэтому наряду с проверкой на неопределенность значения мы оставляем и проверку на наличие свойства **Родитель.**

.

9. Выполним работу с подчиненными справочниками

9.1 Создадим новый справочник, назовем его **Контрагенты**.

Добавим его в подсистемы **БухгалтерскийУчет** и **ОперативныйУчетМатериалов**.

Справочник будет иерархическим, с иерархией групп и элементов.

В состав реквизитов справочника добавим следующие (Рисунок 6.9.):

Имя: ПолноеНаименование, тип – Строка, длина 100.

Имя: КонтактныеСведения, тип – Строка, длина 100

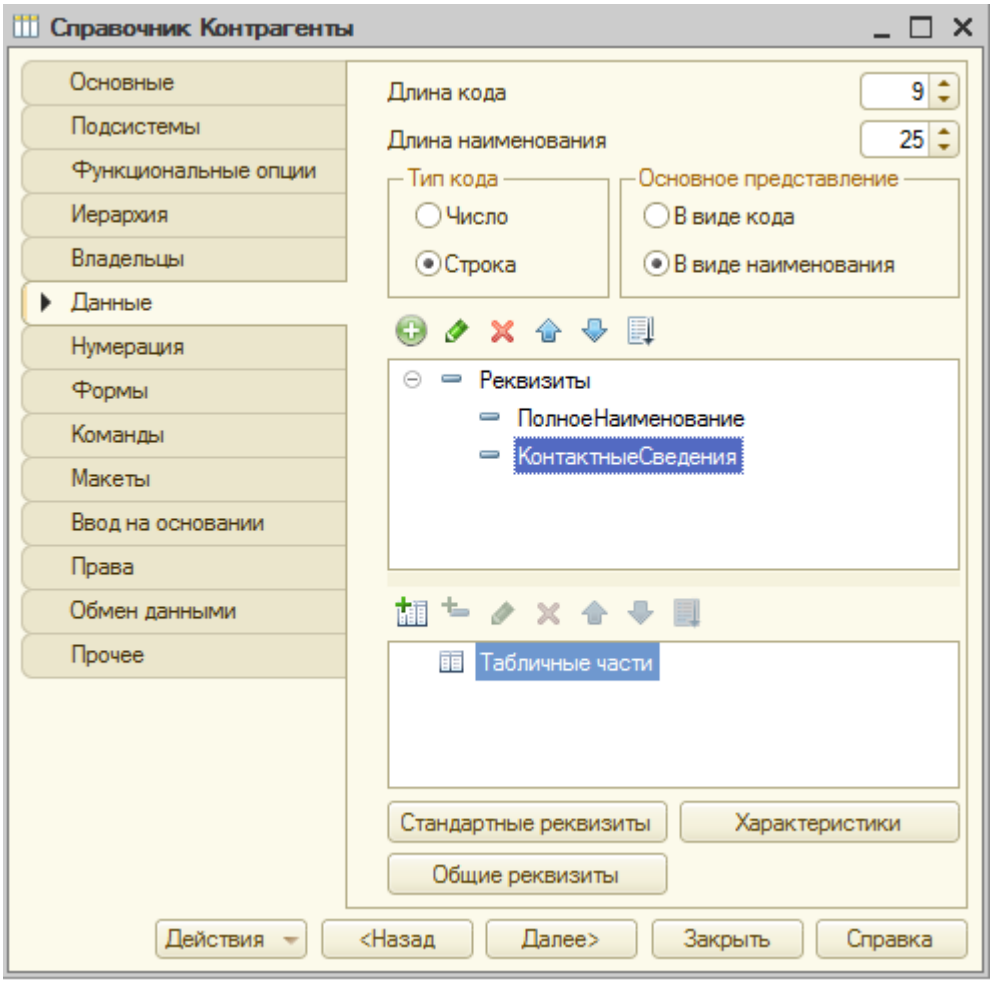


Рисунок 6.9- Справочник Контрагенты

9.2 Создадим еще один справочник. Назовем его **ПредставителиКонтрагентов**. Главная черта этого справочника – то, что он подчинен справочнику **Контрагенты**. Для настройки подчинения используется вкладка окна настройки объекта конфигурации **Владельцы**. Здесь мы должны добавить в **Список владельцев справочника** справочники-владельцы, в нашем случае – справочник **Контрагенты**. После того, как владелец добавлен в этот список, мы можем настроить для него параметр **Использование подчинения**. Здесь возможны три варианта:

Мы укажем в параметре **Использование подчинения** вариант **Элементам**, Рисунок 6.10.

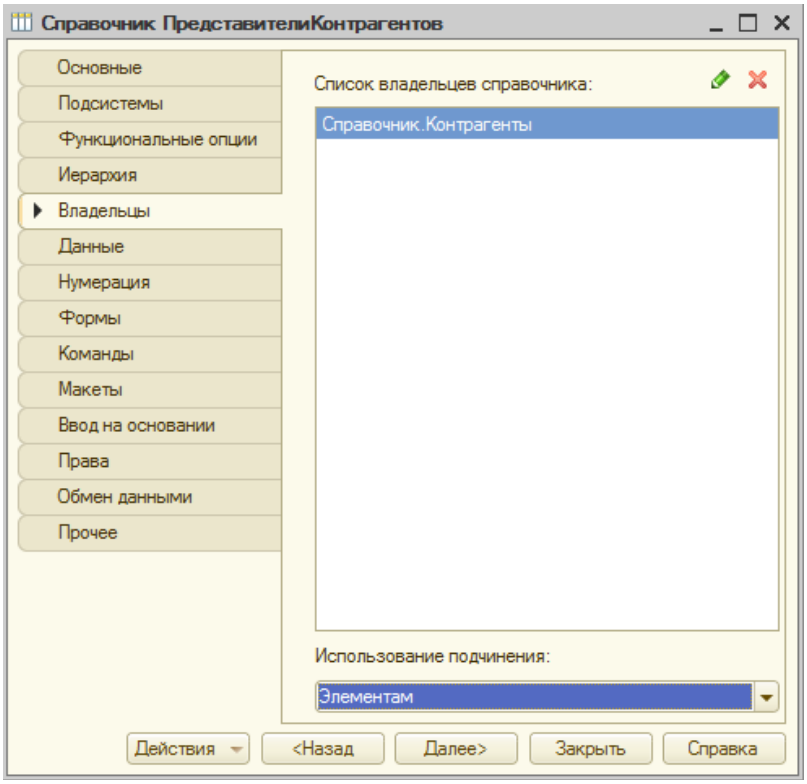


Рисунок 6.10- Настройка подчинения

Добавим справочник в состав подсистем **БухгалтерскийУчет** и **ОперативныйУчетМатериалов**.

В состав реквизитов справочника добавим следующие:

Имя: ФИО, тип – Строка, длина 100

Имя: КонтактныеСведения, тип – Строка, длина 100.

Имя: ПредставительРаботает, тип – Булево.

9.3 Посмотрим теперь, как выглядит работа с созданными справочниками в режиме **1С:Предприятие**. Особенность здесь заключается в том, что, открывая карточку контрагента, в левой ее части мы видим область **Перейти**, где можно найти ссылку для перехода в справочник **ПредставителиКонтрагентов**, Рисунок 6.11.

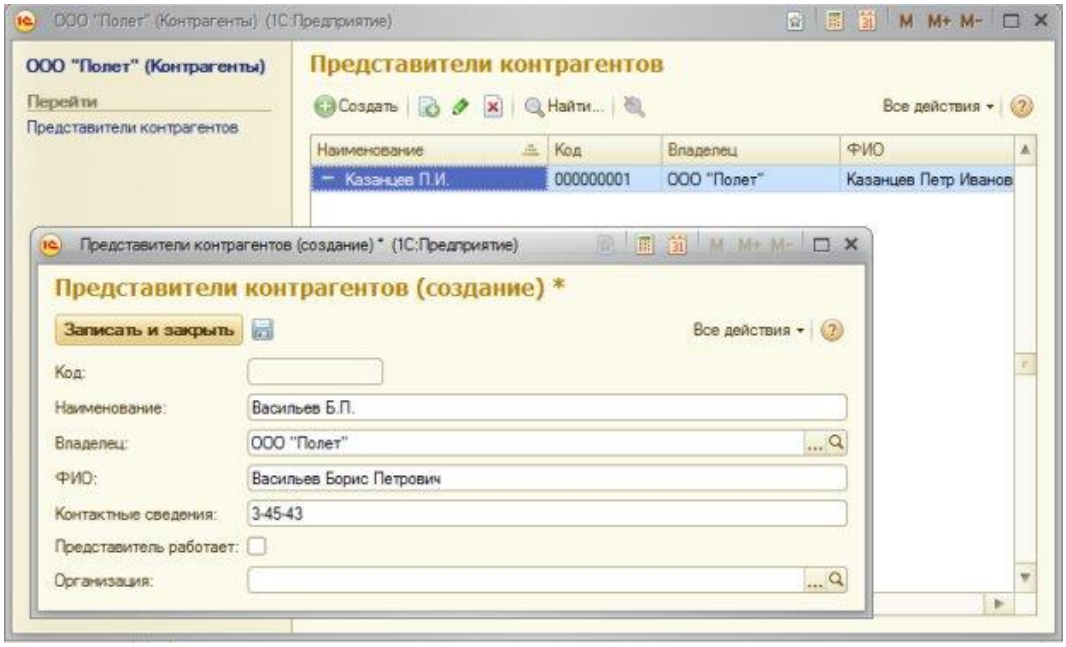


Рисунок 6.11- Форма элемента справочника Контрагенты

При переходе в этот справочник мы будем видеть в открывшемся окне лишь тех представителей, которые относятся к контрагенту, с которым мы в данный момент работаем. При создании новой записи о представителе он автоматически будет "привязываться" к тому контрагенту (поле владелец будет заполнено должным образом), из формы элемента которого мы перешли в справочник **ПредставителиКонтрагентов**. В форме списка справочника будет отображаться ссылка для перехода к форме элемента справочника-владельца, Рисунок 6.12.

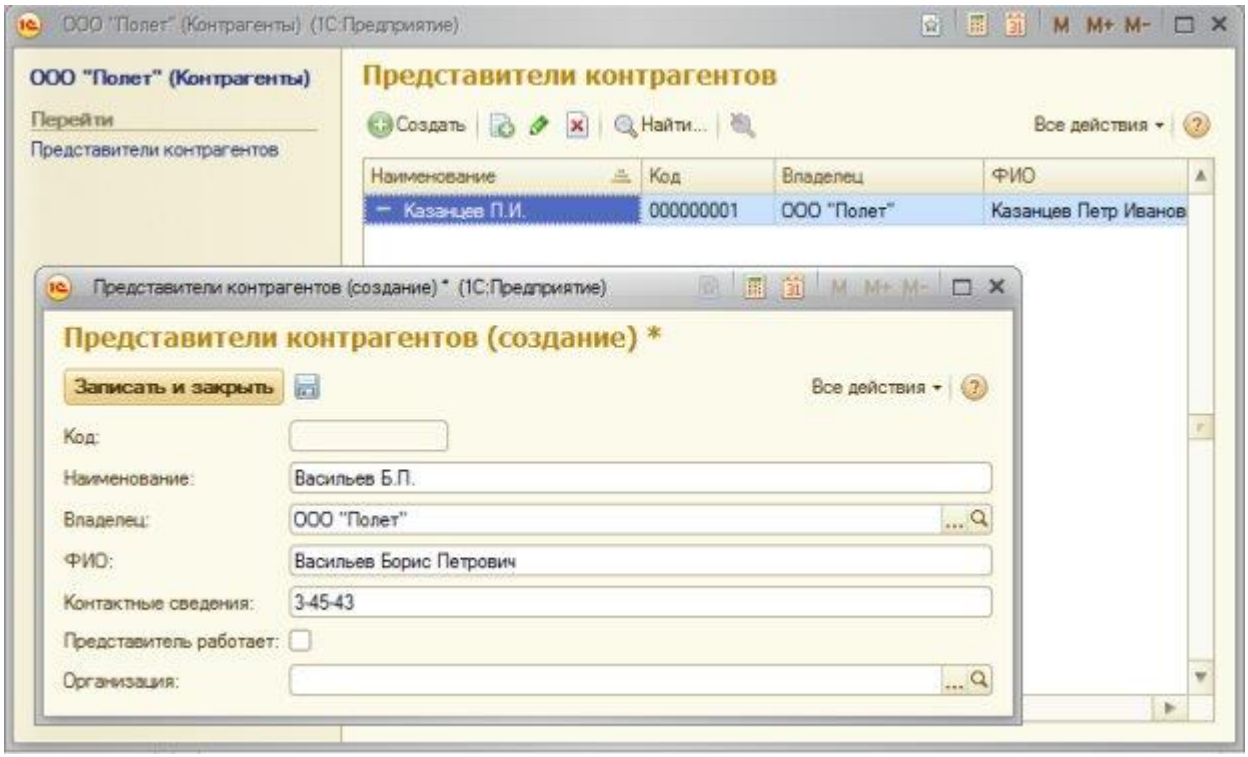


Рисунок 6.12- Формы списка и элемента справочника ПредставителиКонтрагентов

Мы можем создавать элементы справочника **ПредставителиКонтрагентов** и непосредственно перейдя в него, тогда нам придется самостоятельно указывать его владельца –

элемент справочника **Контрагенты**. При переходе в подчиненный справочник не из формы элемента справочника-владельца, мы можем просматривать все его элементы, Рисунок 6.13.

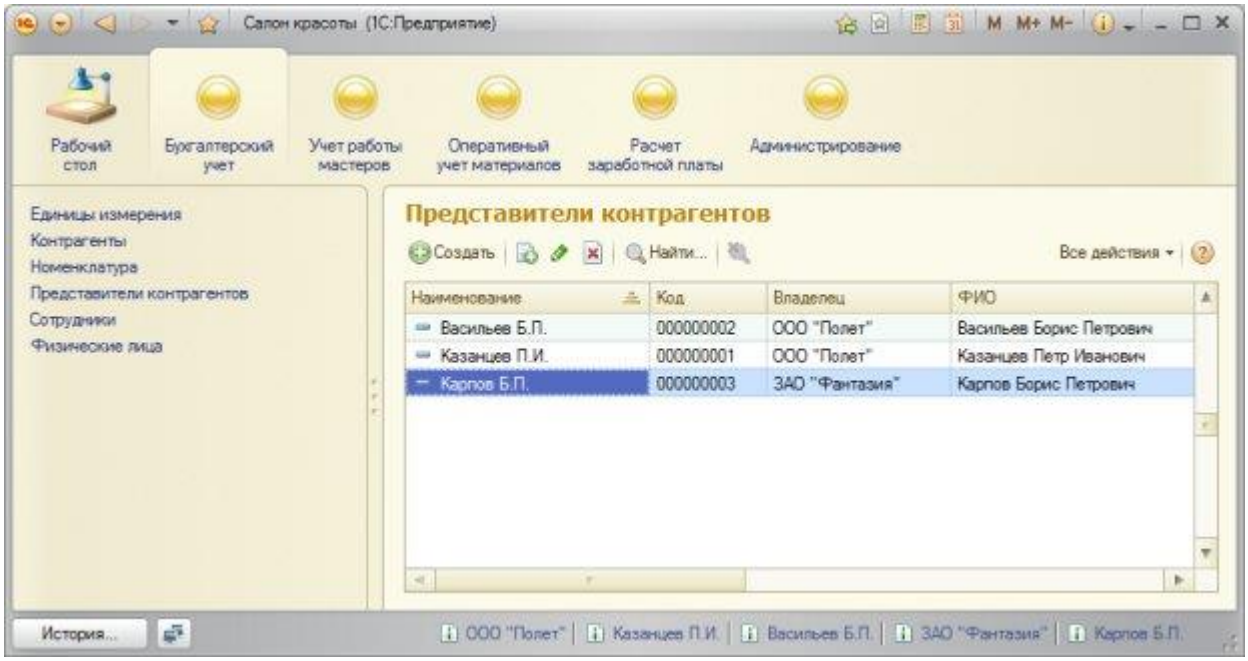


Рисунок 6.13- Просмотр формы списка справочника ПредставителиКонтрагентов

Перейдем в окно редактирования объекта конфигурации справочника **Контрагенты**, перейдем на его закладку **Формы**, создадим новую форму списка. При работе с конструктором форм можно заметить, что на закладке управления реквизитами присутствуют два элемента – **Дерево** и **Список**. Список мы с вами уже видели, а элемент**Дерево** характерен для иерархических справочников, он позволяет облегчить навигацию по большим справочникам, выводя их иерархическую структуру в дополнение к списку. Установим флаг в поле **Дерево**, из списка реквизитов, отображаемых в дереве элементов, выберем **Наименование**, Рисунок 6.14.

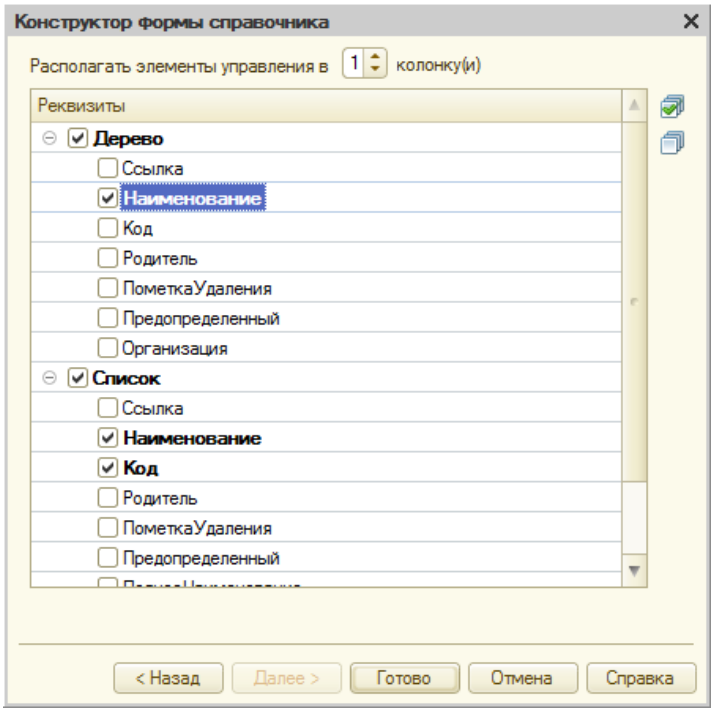


Рисунок 6.14- Конструктор формы справочника Контрагенты

Вот как будет выглядеть форма списка справочника в режиме 1С:Предприятие, рисунок 6.15

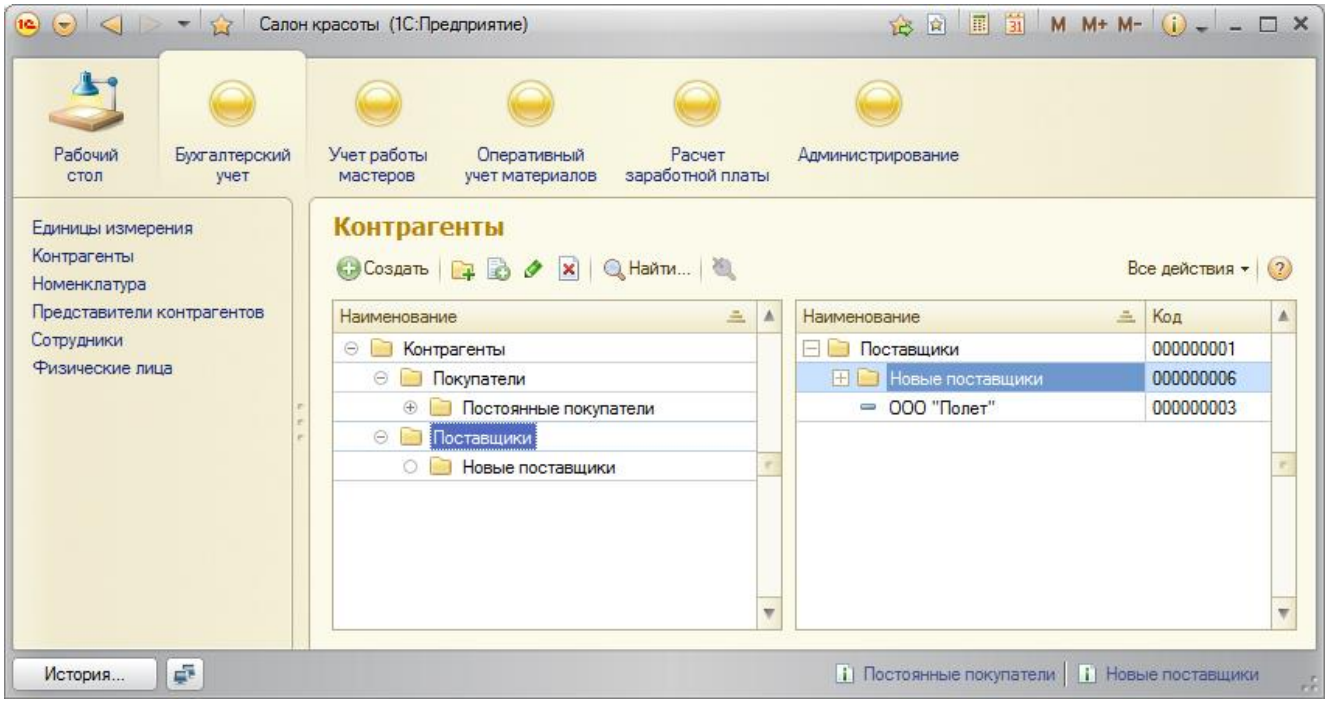


Рисунок 6.15. Форма справочника со списком и деревом элементов

Практическая работа №7 Создание процедур пользователя

Цель: научиться разработке подчиненных справочников

ХОД РАБОТЫ:

1. Расширим справочник **Контрагенты**, добавим в состав его реквизитов еще один – назовем его **ОсновноеКонтактноеЛицо**, тип –**СправочникСсылка.ПредставителиКонтрагентов**. Смысл этого поля заключается в хранении ссылки на представителя контрагента, который является "основным" для данного контрагента. Если нужно связаться с контрагентом, можно открыть его карточку и тут же увидеть, какой представитель является основным. Создадим форму элемента справочника **Контрагенты** и посмотрим на нее, попытавшись установить новое поле – **Основное контактное лицо**, Рисунок 7.1.

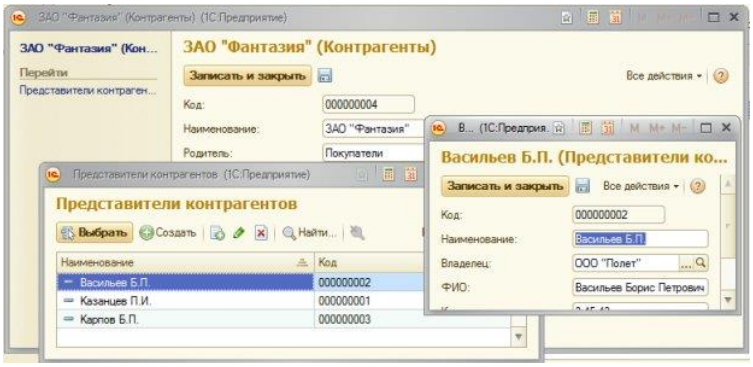


Рисунок 7.1- Попытка заполнения реквизита Основное контактное лицо

2. Видно, что при попытке подбора элемента в данное поле нам показывают не только те элементы справочника **ПредставителиКонтрагентов**, владельцем которых является редактируемый элемент, но и все остальные. Так работать неудобно – это значит, что нам нужно настроить фильтрацию выводимых элементов. Для того, чтобы это сделать, удобнее всего будет воспользоваться свойством **Связи параметров выбора** реквизита **ОсновноеКонтактноеЛицо**. Для открытия палитры свойств реквизита мы можем сделать двойной щелчок по реквизиту в окне редактирования объекта конфигурации, в дереве конфигурации, или воспользоваться командой контекстного меню Свойства.

3. В открывшейся палитре свойств найдем свойство **Связи параметров выбора** и нажмем на кнопку с тремя точками около этого поля. Появится окно **Связи параметров выбора**, Рисунок 7.2.

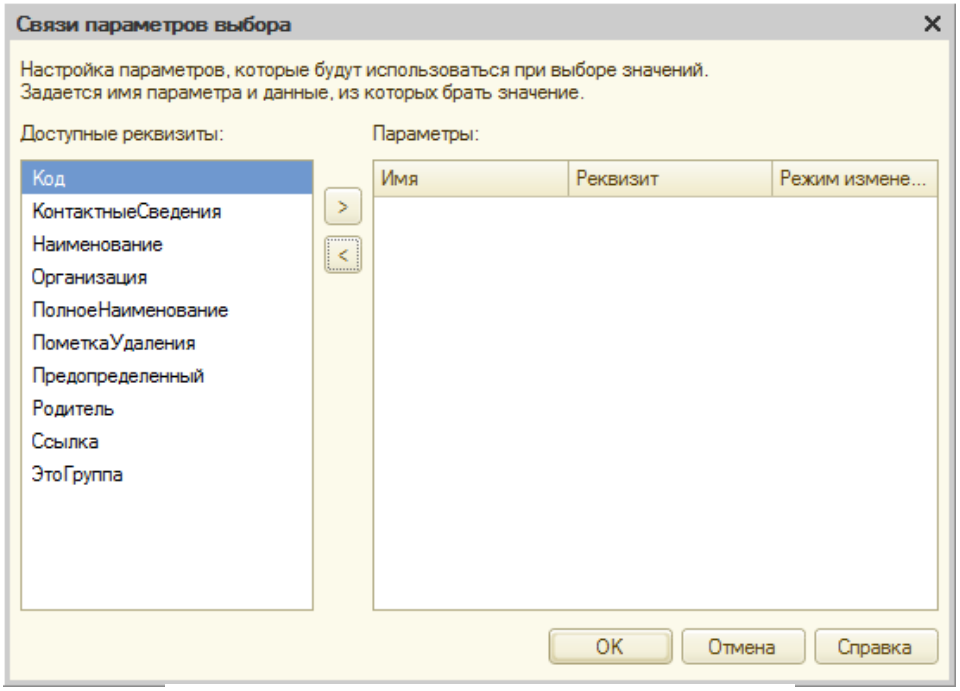


Рисунок 7.2- Окно Связи параметров выбора

4. В левой части окна можно видеть доступные реквизиты (это реквизиты открытого элемента справочника **Контрагенты**), в правом – параметры, влияющие на отбор элементов в появляющемся окне выбора элементов при заполнении поля представителя контрагента. Выделим реквизит **Ссылка** и нажмем на кнопку **Добавить выбранный реквизит в параметры выбора** (она находится между полями). Окно примет следующий вид, Рисунок 7.3.

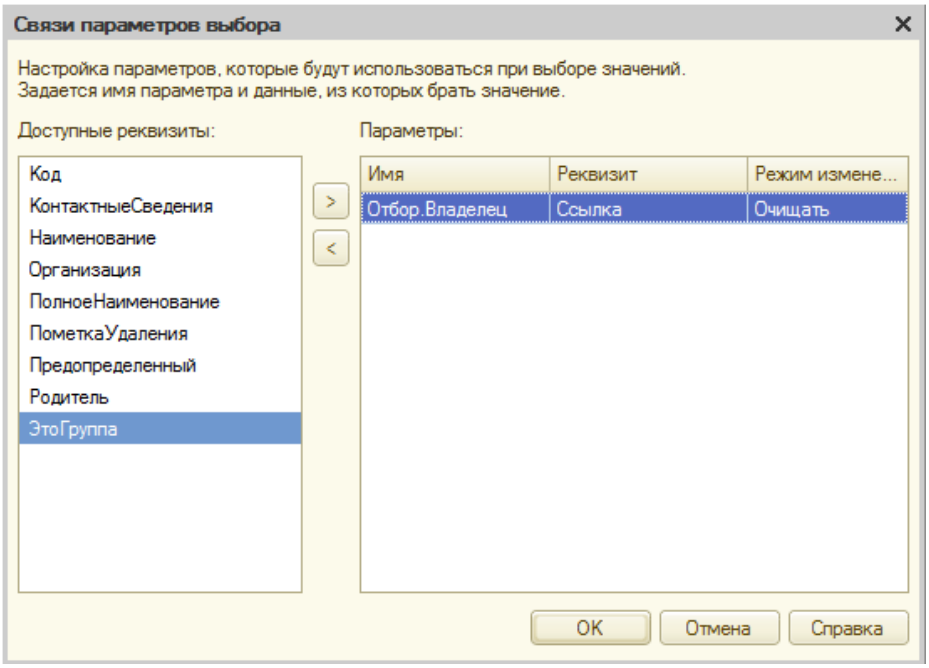


Рисунок 7.3- Окно Связи параметров выбора с настроенным параметром

5. В данном случае в строке области **Параметры** отображается как раз то, что нам нужно – нам нужно, чтобы отбор в раскрывающемся списке происходил по владельцу, а именно – по текущему открытому элементу справочника **Контрагенты**, на который и указывает реквизит **Ссылка**. В поле имя можно выбрать другие варианты отбора, Рисунок 7.4.

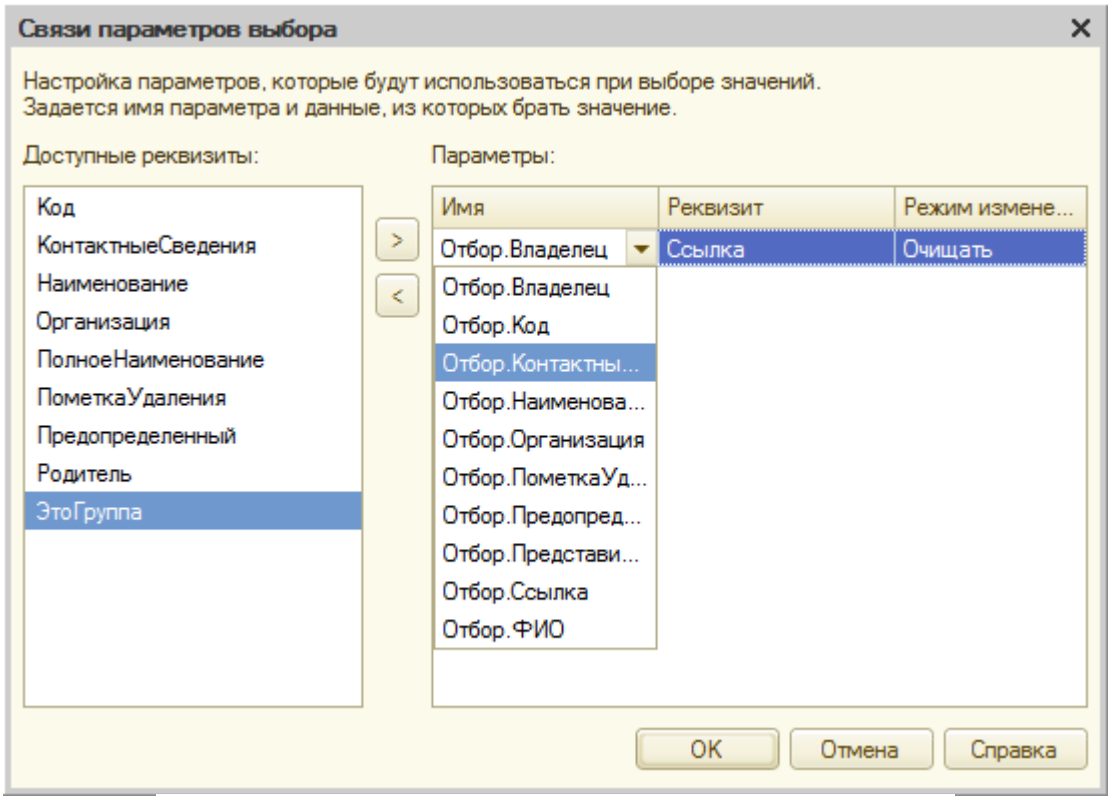


Рисунок 7.4- Настройки в окне Связи параметров выбора

6. После того, как эта настройка выполнена, мы можем нажать **ОК** в окне **Связи параметров выбора** и проверить функциональность решения – при заполнении поля**ОсновноеКонтактноеЛицо** список выбора ограничивается подчиненными элементами.

7. Добавим в справочник **Контрагенты** еще один реквизит – **ТелефонКонтактногоЛица**. Зададим тип – **Строка**, длина – **100**. Этот реквизит соответствует реквизиту**КонтактныеСведения** справочника **ПредставителиКонтрагентов**. Он нужен нам исключительно для удобства – для того, чтобы, когда в форме контрагента указано основное контактное лицо, пользователю не пришлось бы, для поиска телефона контактного лица, заглядывать в его карточку.

8. После добавления реквизита в справочник **Контрагенты**, запустим режим 1С:Предприятие и откроем форму одного из элементов этого справочника. Если присмотреться к этой форме на данном этапе работы, окажется, что реквизита **ТелефонКонтактногоЛица** на ней не наблюдается. Все дело в том, что, создав собственную форму элемента для справочника, мы отказываемся от автоматического механизма создания форм, который, если бы не наша, самостоятельно созданная ранее форма, автоматически построил бы форму с новым реквизитом.

9. Добавим элемент управления для реквизита **ТелефонКонтактногоЛица** на форму. Откроем форму элемента справочника Контрагенты для редактирования и перетащим реквизит **ТелефонКонтактногоЛица** с вкладки **Реквизиты** на вкладку **Элементы**, Рисунок 7.5.

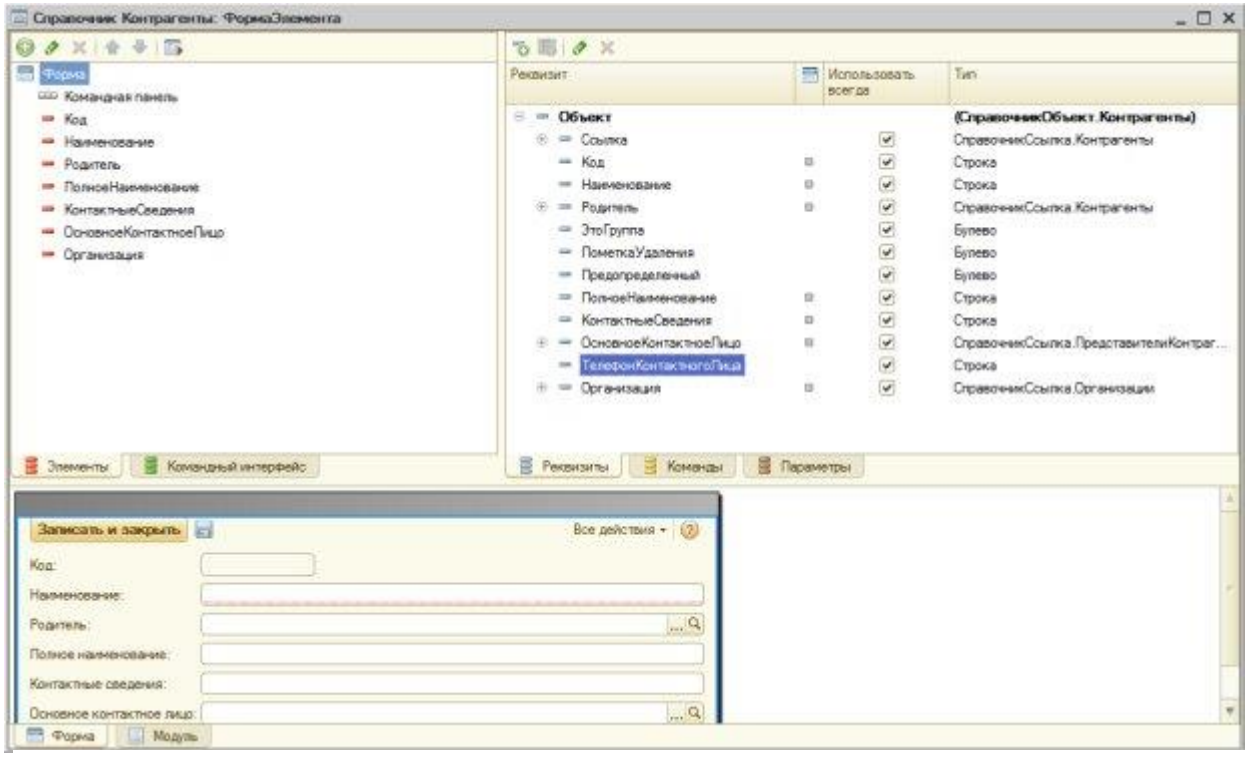


Рисунок 7.6- Реквизит ТелефонКонтактногоЛица нужно переместить со вкладки Реквизиты на вкладку Элементы

10. В справочнике **ПредставителиКонтрагентов** есть реквизит, который указывает на то, что представитель контрагента работает в организации-контрагенте – это реквизит логического типа **ПредставительРаботает**. Нам нужно реализовать следующий функционал. Если пользователь, заполняя карточку элемента справочника **Контрагенты**выбирает в качестве

реквизита **ОсновноеКонтактноеЛицо** сотрудника, у которого флаг **ПредставительРаботает** не установлен – мы предупреждаем пользователя об этом, выводя сообщение.

Для этого нам понадобится перехватить событие изменения поля **ОсновноеКонтактноеЛицо**, после чего проверить, установлен ли у выбранного контактного лица флаг**ПредставительРаботает**, и если такой флаг не установлен – вывести сообщение пользователю.

11. Из контекстного меню элемента формы **ОсновноеКонтактноеЛицо** выберем событие **ПриИзменении**, откроется редактор кода, в котором уже будет создана пустая клиентская процедура для перехвата этого события. К этому моменту реквизит **ОсновноеКонтактноеЛицо** уже будет содержать выбранного представителя контрагента. Нам понадобится серверная процедура, которая обратится к реквизиту этого представителя **ПредставительРаботает** и вернет нам его значение. После того, как мы получим с сервера сведения о том, работает ли представитель, мы примем решение – выводить ли пользователю сообщение или нет.

12. Все это реализовано с помощью нижеприведенного кода:

```
&НаКлиенте
Процедура ОсновноеКонтактноеЛицоПриИзменении(Элемент)
Если НЕ ПроверитьЗаполнениеРеквизита() Тогда
 Сообщить("Выбранное контактное лицо, "+Объект.ОсновноеКонтактноеЛицо+", не работает у контрагента.");
КонецЕсли;
КонецПроцедуры

&НаСервере
Функция ПроверитьЗаполнениеРеквизита()
 Возврат (Объект.ОсновноеКонтактноеЛицо.ПредставительРаботает);
КонецФункции
```

13. Кроме того, зададим автоматический механизм переноса в реквизит **ТелефонКонтактногоЛица** сведений из элемента справочника **ПредставителиКонтрагентов**, который указан в поле **ОсновноеКонтактноеЛицо**. В частности, дополним обработчик события для поля **ОсновноеКонтактноеЛицо ПриИзменении():**

14. Подготовим серверную процедуру, которая будет работать с реквизитами объекта, она будет иметь следующий вид:

```
&НаСервере
Процедура УстановитьНомерПредставителя()

Объект.ТелефонКонтактногоЛица=Объект.ОсновноеКонтактноеЛицо.КонтактныеСведения;
КонецПроцедуры
```

15. Вызовем эту процедуру в уже существующем обработчике **ПриИзменении**, код модуля приобретет вид, показанный на Рисунок 7.7.

Кроме того, в окне его свойств отредактируем свойство **Вид** – выберем его значение **Поле надписи**. Пользователь не будет ничего в это поле вводить самостоятельно, поэтому поле надписи нас вполне устроит.

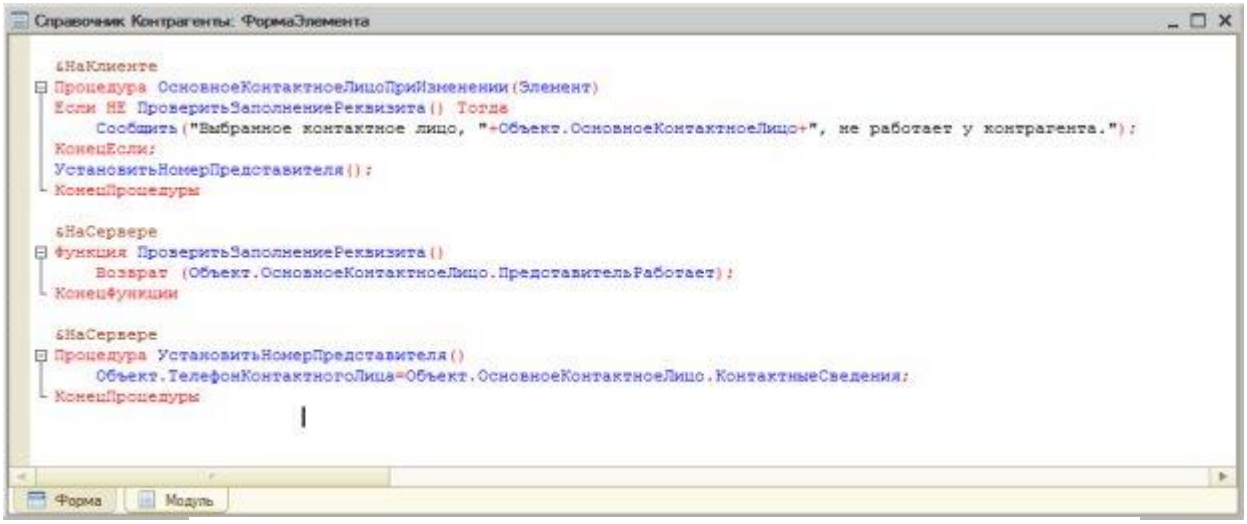


Рисунок 7.7- Код модуля формы элемента справочника Контрагенты

16. При выборе "неподходящего" представителя окно элемента справочника Контрагенты примет следующий вид, Рисунок 7.8.

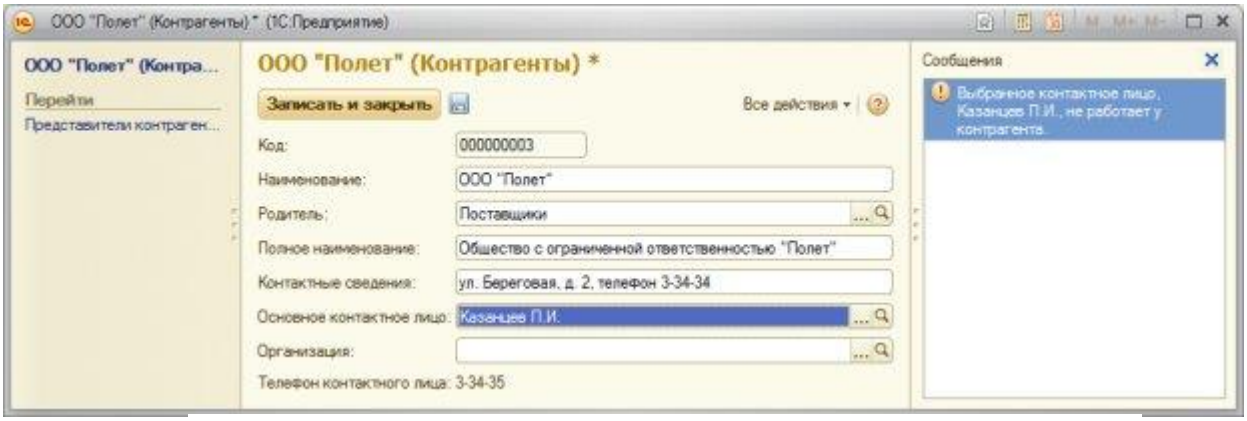


Рисунок 7.8- Сообщение о выборе неподходящего контактного лица

17. Объявим процедуру **ОбработкаПроверкиЗаполнения ()**, открыв модуль объекта (закладка **Прочее** окна редактирования свойств объекта, кнопка **Модуль объекта**) выбором из списка **Процедуры и функции панели инструментов** модуль процедуры ОбработкаПроверкиЗаполнения().

Эта процедура работает на сервере, мы можем напрямую обращаться к реквизитам объекта.

```
Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)
 Если СтрДлина(ПолноеНаименование) < 5 И НЕ ЭтотОбъект.ЭтоГруппа Тогда
   Отказ=Истина;
   Сообщить("Полное наименование организации должно быть не короче 5-ти символов");
 КонецЕсли;
КонецПроцедуры
```

Передаваемый в процедуру параметр **Отказ** можно установить в значение **Истина** для того, чтобы показать, что проверка не пройдена. **ПроверяемыеРеквизиты** – это массив, он содержит

реквизиты для автоматической проверки, в частности, там находятся те реквизиты, для которых включена автоматическая проверка заполнения. По умолчанию это – реквизит **Наименование.** В этом можно убедиться, просмотрев содержимое переменной в отладчике.

Практическая работа №8 Использование функций диалога с пользователем

Цель: изучить подробности объектной модели справочников, программной работе со справочниками, а так же – созданию обработок и простых отчетов.

ХОД РАБОТЫ:

1. Начнем с создания обработки, которая выводит имена всех справочников, имеющихся в системе. Для этого добавим новую обработку в ветви **Обработки** дерева конфигурации. Назовем ее **РаботаСоСправочниками**, Рисунок 8.1.

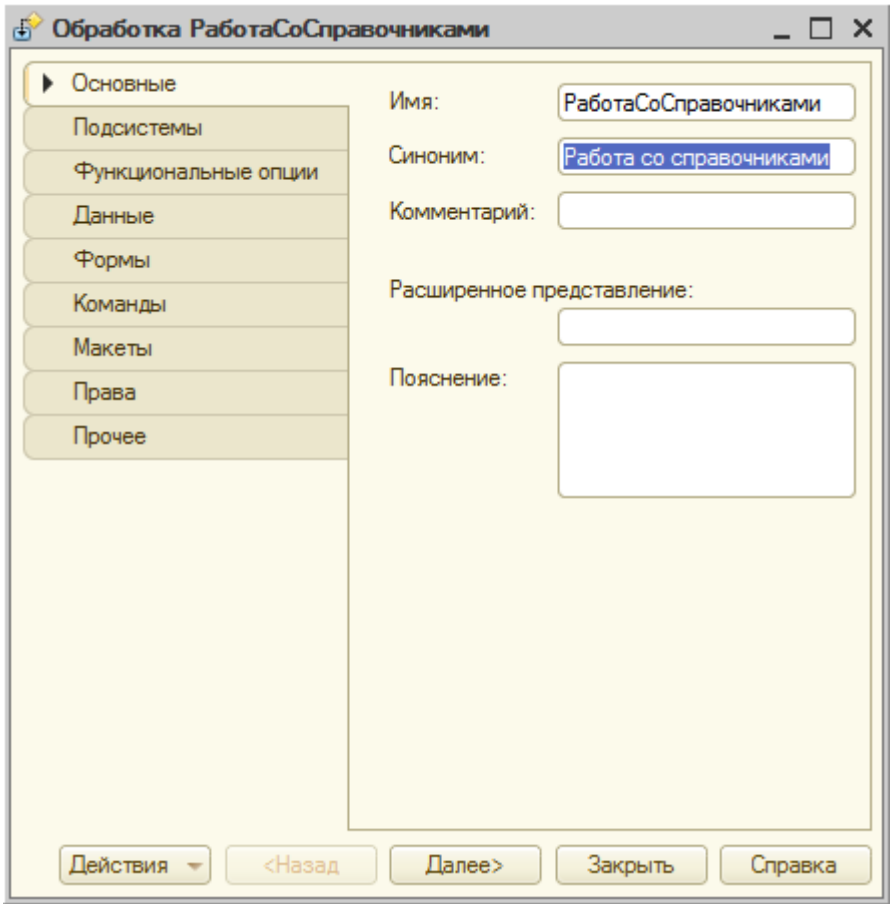


Рисунок 8.1- Создание обработки

2. Включим новую обработку в состав подсистемы **Администрирование** на закладке **Подсистемы**. Перейдем на закладку **Формы** и создадим форму обработки. Наша обработка не имеет реквизитов – сразу после запуска конструктора формы обработки, мы можем нажать на кнопку **Готово** и увидим пустую форму обработки, Рисунок 8.2.

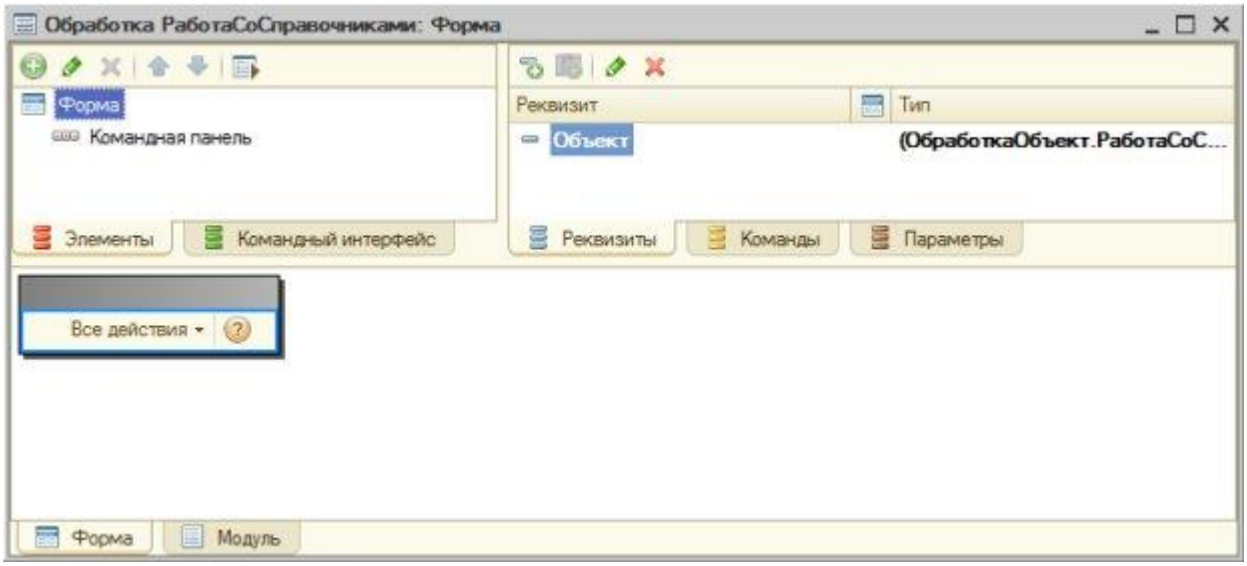


Рисунок 8.2- Форма обработки

3. Перейдем на вкладку **Команды** в окне редактора форм. После этого нам будут доступны еще несколько вкладок, нас интересует первая из них – **Команды формы**, Рисунок 8.3.

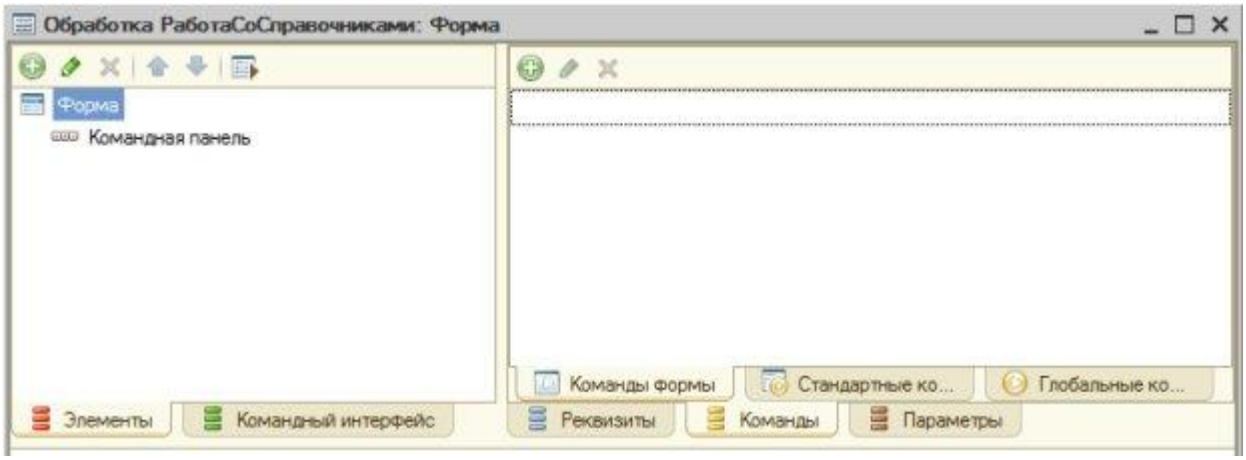


Рисунок 8.3- Переход к командам формы обработки

4. Список команд формы пуст – нам нужно создать собственную команду. Нажмем на кнопку **Добавить** в верхней части панели **Команды формы**, назовем ее**ВывестиСписокСправочников**, в окне свойств команды нажмем на кнопку с увеличительным стеклом в поле свойства **Действие** – в модуле формы будет создана процедура для этой команды, Рисунок 8.4.

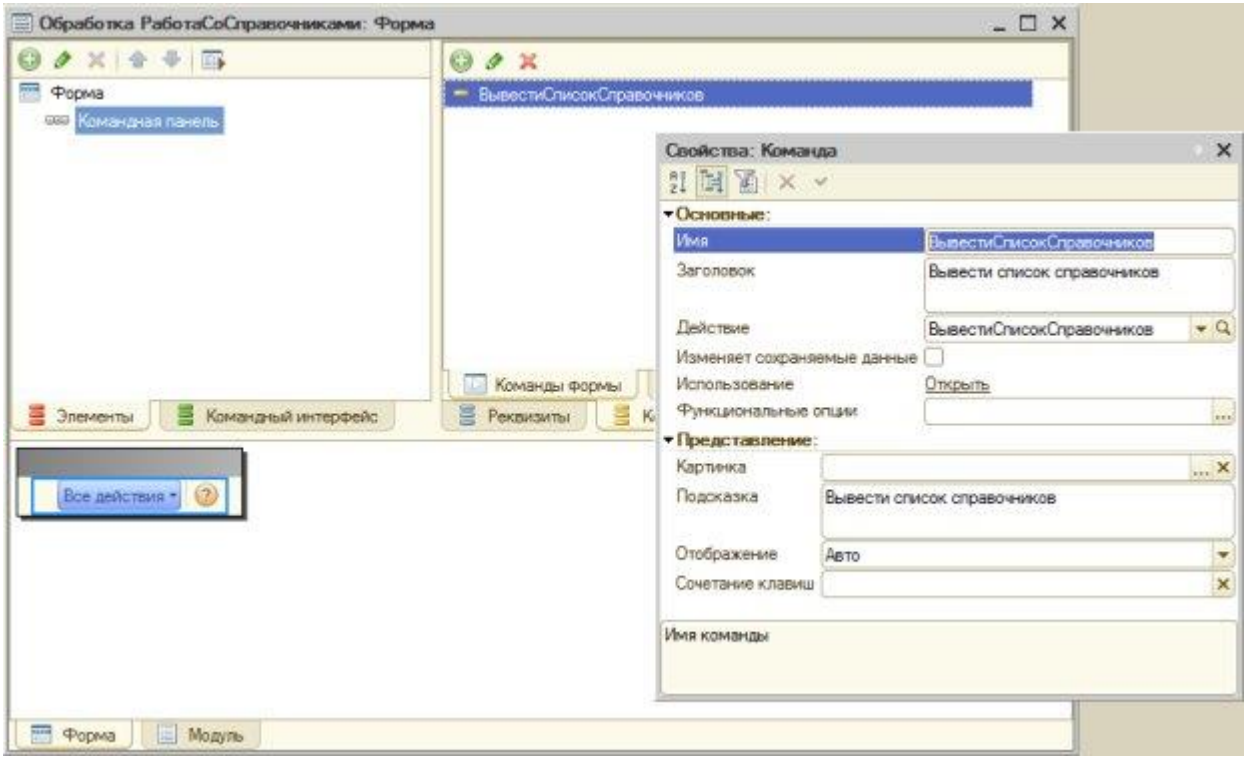


Рисунок 8.4- Настройка команды

В модуль формы был добавлен такой код:

```
&НаКлиенте
Процедура ВывестиСписокСправочников(Команда)
  // Вставить содержимое обработчика.
КонецПроцедуры
```

То, что мы добавили в обработку команду, еще не означает автоматическое добавление на форму команды, например, кнопки, нажатие которой приведет к выполнению команды. Добавить такую кнопку на форму можно несколькими способами. Во-первых, мы можем просто перетащить команду из панели **Команды формы** на панель**Элементы** – на форме появится кнопка **Вывести список справочников**, а напротив команды – серый квадратик, говорящий о присутствии элемента управления, связанного с командой, на форме.

Во-вторых, в список элементов формы можно добавить кнопку (кнопка **Добавить** в командной панели закладки **Элементы**) и задать свойства кнопки, в частности, в свойстве **ИмяКоманды** выбрать нужную команду. После добавления кнопки и настройки ее связи с командой, редактор форм приобрел вид, показанный на Рисунок 8.5.

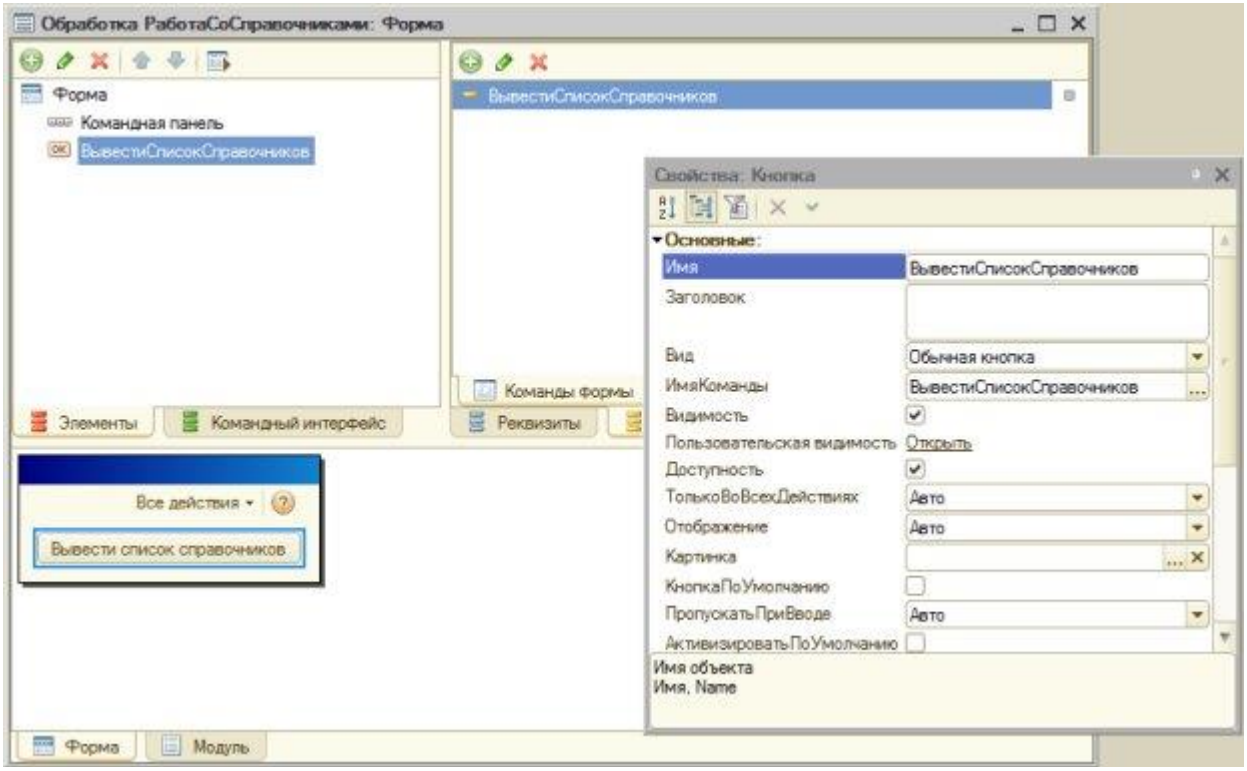


Рисунок 8.5- Настройка кнопки

5. Теперь приступим к редактированию кода. Код команды выполняется на клиенте, нам же нужно работать с базой данных, то есть – объявить серверную процедуру или функцию. В итоге у нас получился следующий код:

```
&НаКлиенте
Процедура ВывестиСписокСправочников(Команда)
  ВывестиИменаСправочников();
КонецПроцедуры

Процедура ВывестиИменаСправочников()
  Для каждого Справочник из Метаданные.Справочники Цикл
      Сообщить (Справочник.Имя);
  КонецЦикла;
КонецПроцедуры
```

Мы получаем имена справочников и выводим их в окно сообщений, Рисунок 8.6.

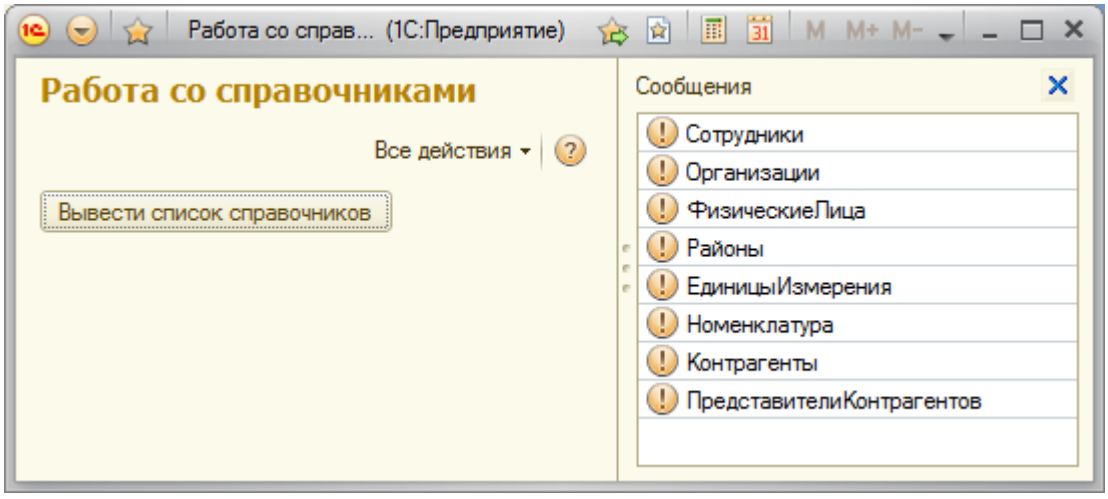


Рисунок 8.6- Вывод списка справочников

6. Теперь нужно программно создать элемент справочника с заданными параметрами. На верхнем уровне типов данных, которые имеют отношение к справочникам, находится объект Справочники, имеющий тип СправочникиМенеджер. С его помощью можно обращаться к отдельным справочникам, через их объектыСправочникМенеджер. При работе с объектом типа СправочникиМенеджер используется свойство глобального контекста Справочники.
Обращение к объектам СправочникМенеджер возможно по имени справочника, заданному в конфигурации. Мы собираемся программно создать элемент с наименованием, которое задаст пользователь в форме обработки. Для этого добавим в список команд формы новую – назовем ее СоздатьЭлементСправочника, создадим ее процедуру, добавим ее на форму. Добавим новый реквизит в список реквизитов, назовем его НаименованиеЭлемента, зададим тип – Строка, длина 25, так же переместим реквизит в область Элементы – там он будет представлен в виде текстового поля, Рисунок 8.7.

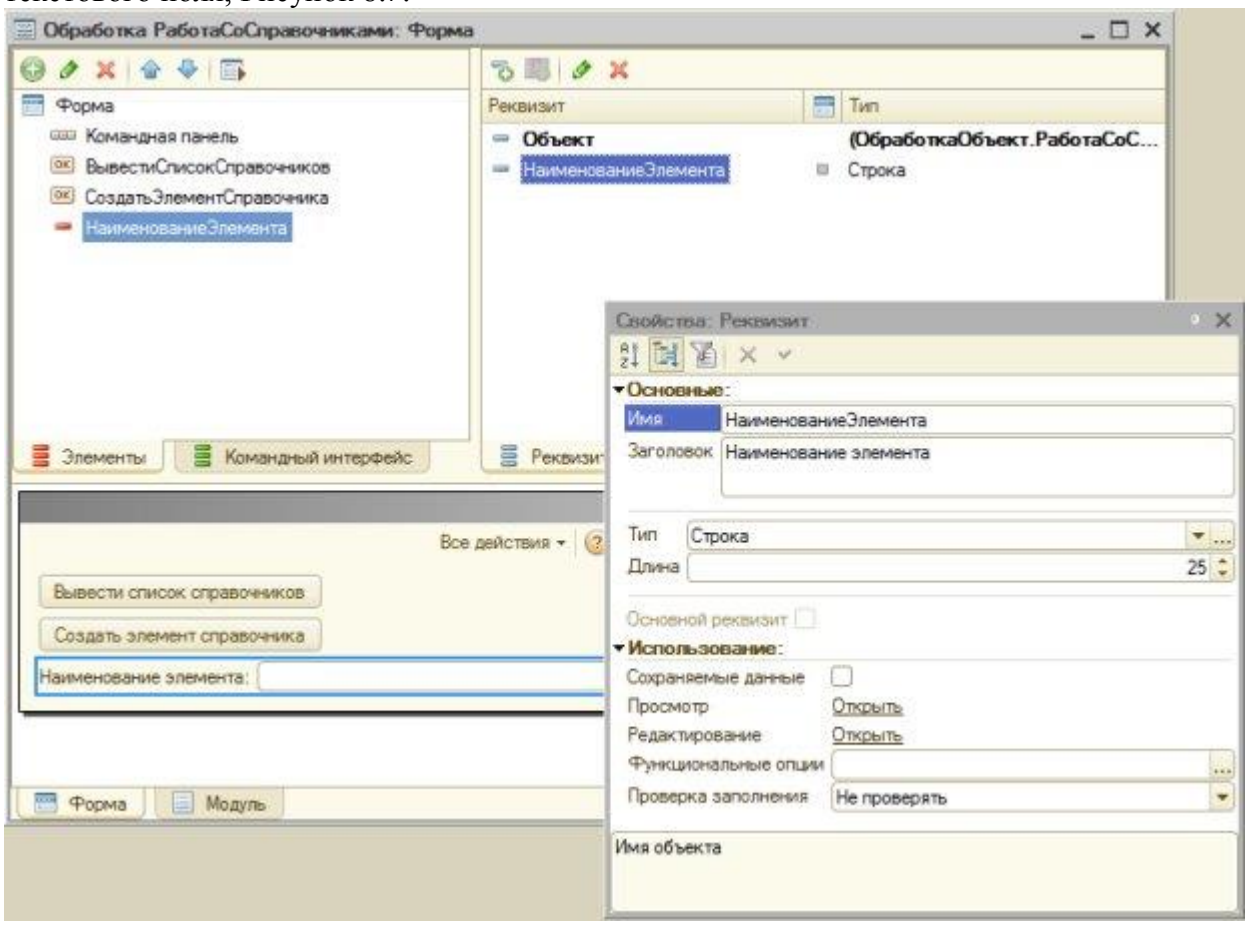


Рисунок 8.7- Настройка нового реквизита формы

7. Добавим еще один реквизит – назовем его ИмяСправочника, тип Строка, длина – 100. Сюда пользователь будет вводить имя справочника, в котором он хочет создать новый элемент. На нашей форме теперь имеются три логически связанных элемента. Удобно объединить их в одну группу, чтобы пользователь сразу мог понять, что они работают вместе. Для этого можно сгруппировать элементы. В командной панели вкладки Элементы нажмем на кнопку Добавить, появится окно – Тип элемента (Рисунок 8.8.), среди списка элементов, представленных в котором, можно найти несколько видов групп.

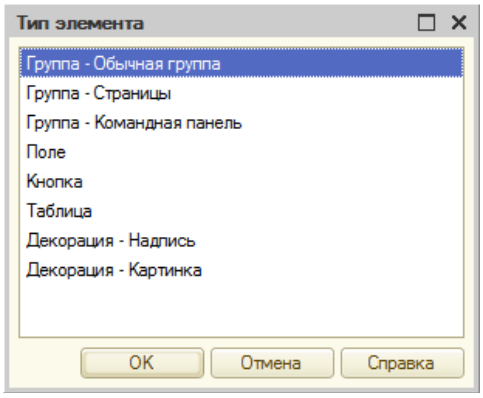


Рисунок 8.8- Добавление новой группы на форму

8. Добавим на форму новую группу, назовем ее СозданиеЭлементаСправочника, перетащим в нее элементы управления, относящиеся к этой группе. Результат реорганизации элементов показан на Рисунок 8.9.

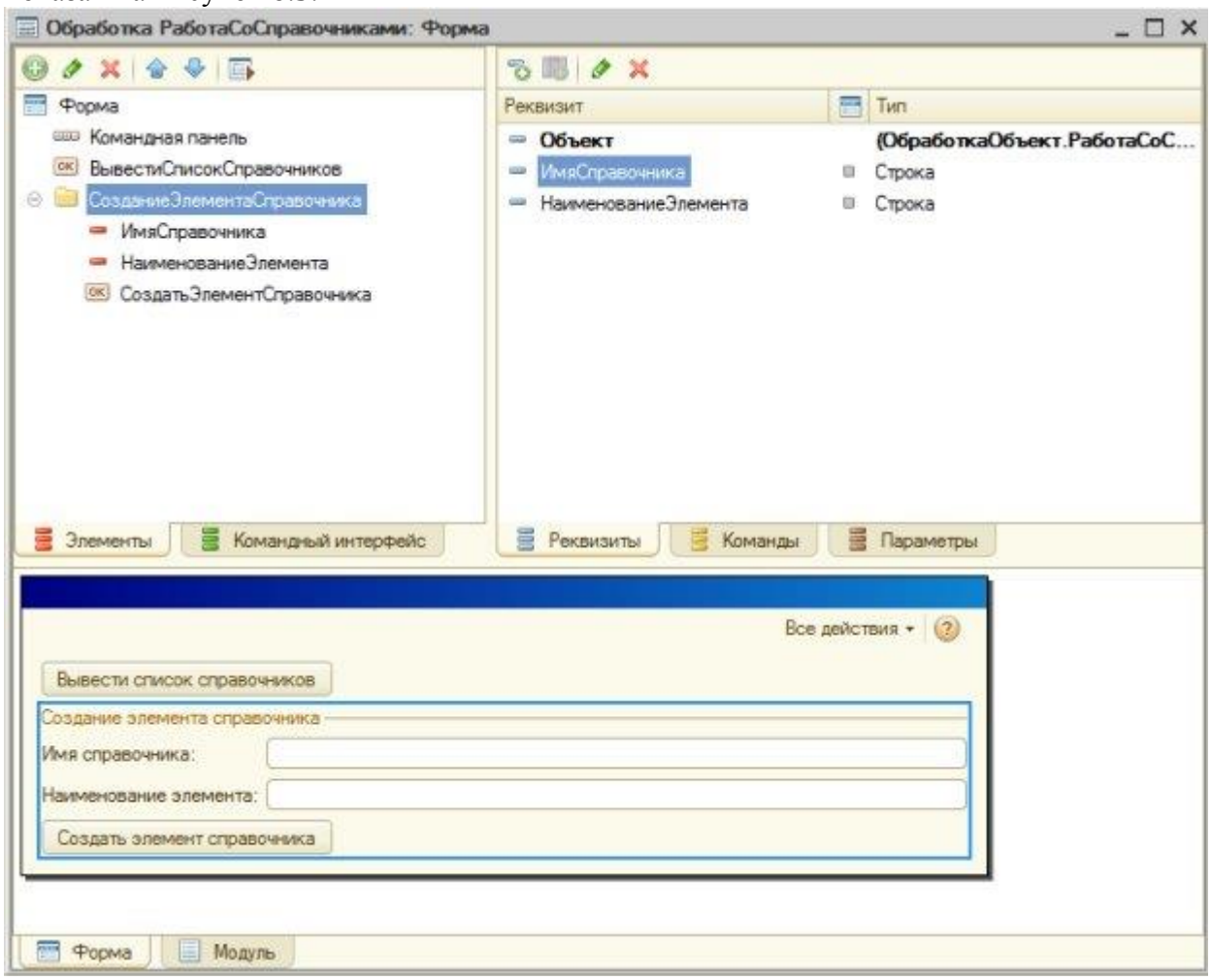


Рисунок 8.9- Добавление новой группы на форму

9. Теперь займемся кодом. Нам, в дополнение к клиентской процедуре команды СоздатьЭлементСправочника, понадобится серверная процедура или функция, которая и занимается созданием элемента. Обратиться к объекту СправочникМенеджер для конкретного справочника можно различными способами. Предположим, мы заранее знаем, с каким справочником нам нужно работать (например, это – справочник Номенклатура). Для того, чтобы вызвать метод этого справочника СоздатьЭлемент, нам понадобится такая конструкция:

НовыйЭлемент=Справочники.Номенклатура.СоздатьЭлемент();

10. После того, как мы получили переменную типа СправочникОбъект, мы можем настроить необходимые свойства конкретного элемента справочника (в нашем случае – наименование) и записать элемент. Вот, как выглядит результирующий код:

```
&НаКлиенте
Процедура СоздатьЭлементСправочника(Команда)
  КодНовогоЭлемента=СоздатьЭлементСправочникаНаСервере();
  Сообщить("В справочнике "+ИмяСправочника+" создан элемент
   "+НаименованиеЭлемента + " с автоматически присвоенным кодом:
     "+КодНовогоЭлемента);
КонецПроцедуры

Функция СоздатьЭлементСправочникаНаСервере()
  НовыйЭлемент = Справочники[ИмяСправочника].СоздатьЭлемент();
  НовыйЭлемент.Наименование=НаименованиеЭлемента;
  НовыйЭлемент.Записать();
  Возврат (НовыйЭлемент.Код);
КонецФункции
```

Вот, каковы результаты работы этого кода, Рисунок 8.10.

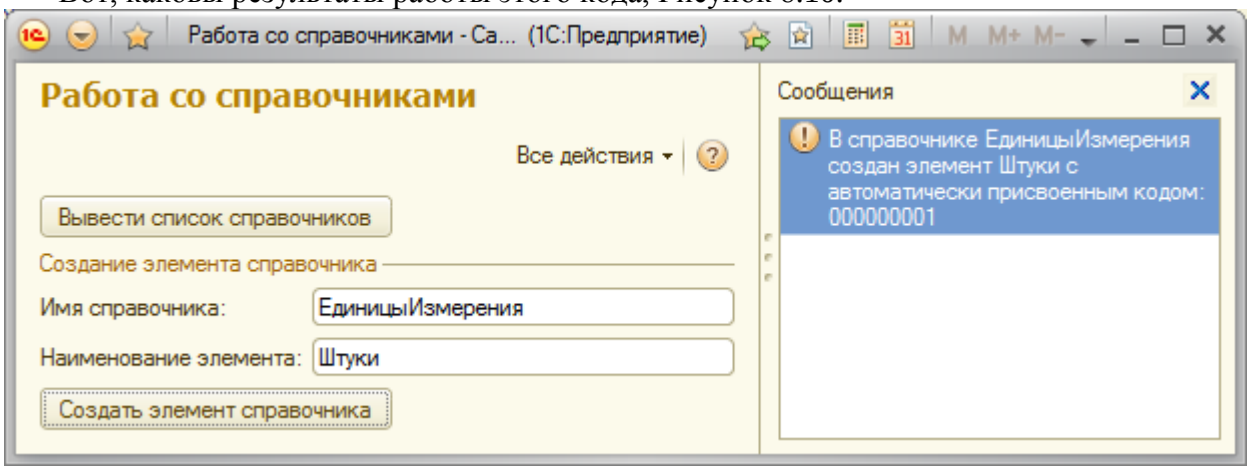


Рисунок 8.10- Создание нового элемента справочника

11. После создания процедуры, связанной с этой командой и серверной процедуры, выполняющей работу с базой. У нас получился такой код:

```
&НаКлиенте
Процедура ПометитьНаУдалениеВсеЭлементыСправочника(Команда)
  ПометитьНаУдаление();
КонецПроцедуры

Процедура ПометитьНаУдаление()
  СчетчикПомеченных = 0;
  Выборка = Справочники[ИмяСправочника].Выбрать();
  Пока Выборка.Следующий() Цикл
    Элемент=Выборка.ПолучитьОбъект();
    Если НЕ Элемент.ЭтоГруппа Тогда
      Элемент.УстановитьПометкуУдаления(Истина);
      СчетчикПомеченных=СчетчикПомеченных+1;
    КонецЕсли;
  КонецЦикла;
```

Сообщить("В справочнике "+ИмяСправочника+" помечено на удаление"+СчетчикПомеченных+" элементов");
КонецПроцедуры
.
12. Нужно в заданном справочнике нужно найти элемент с заданным наименованием (или сообщить, что элемента с таким наименованием в справочнике нет), изменить регистр символов в наименовании таким образом, чтобы все буквы были прописными, и сообщить пользователю его код с указанием старого и нового наименования.
Обычным образом добавим в форму обработки новую команду, для указания имени справочника и наименования искомого элемента используем те же реквизитыИмяСправочника и НаименованиеЭлемента, реорганизуем элементы управления на форме, Рисунок 8.11.

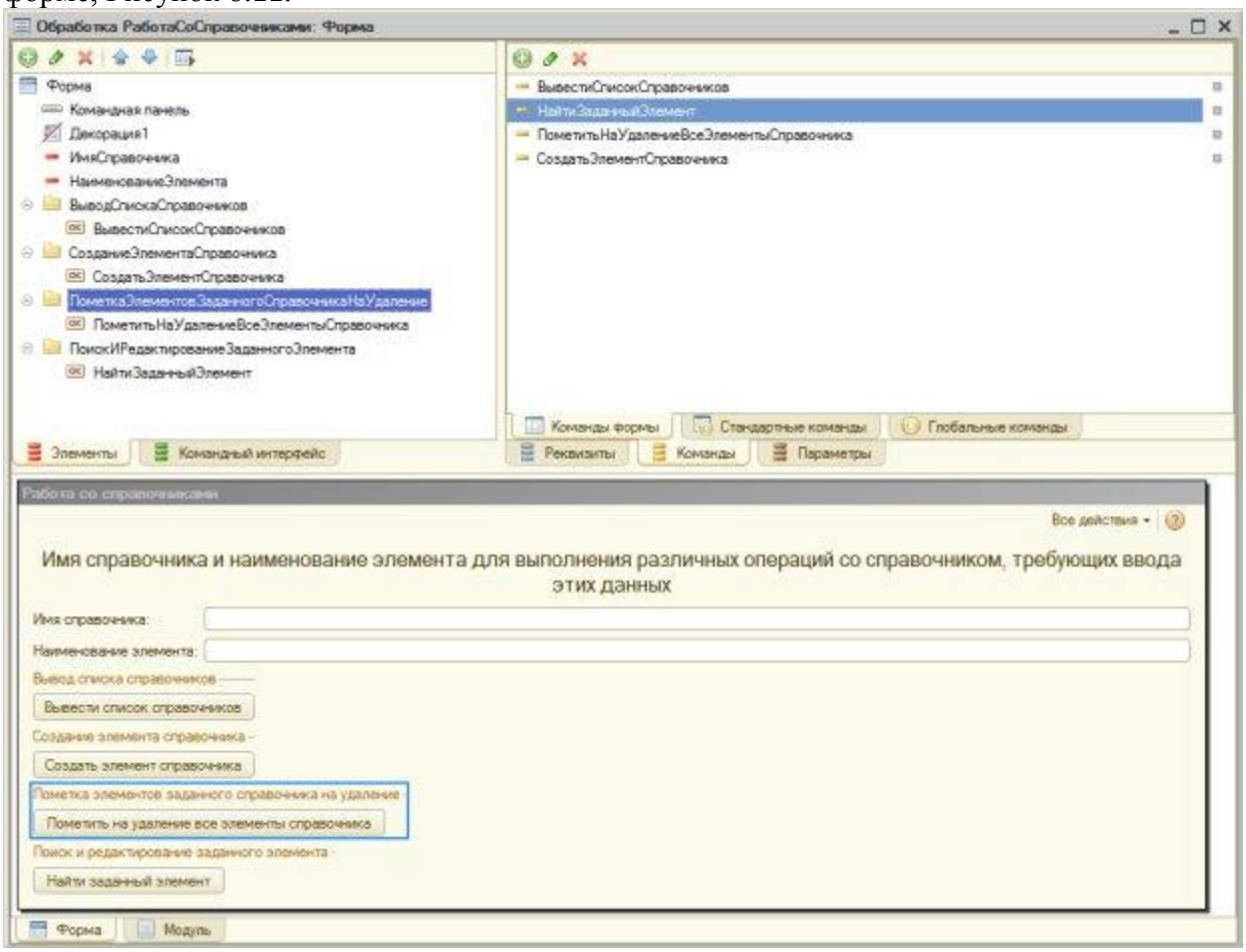


Рисунок 8.11- Переработанная форма

Поиск, редактирование заданного элемента и вывод необходимых сообщений реализуется с помощью следующего кода:

```
&НаКлиенте
Процедура НайтиЗаданныйЭлемент(Команда)
  НайтиЗаданныйЭлементНаСервере();
КонецПроцедуры

Процедура НайтиЗаданныйЭлементНаСервере()

ССылкаНаЭлемент=Справочники[ИмяСправочника].НайтиПоНаименованию(НаименованиеЭлемента);
```

```
        Если СсылкаНаЭлемент=Справочники[ИмяСправочника].ПустаяСсылка() Тогда
          Сообщить ("В справочнике "+ИмяСправочника+" нет элемента
"+НаименованиеЭлемента);
        Иначе
          Элемент=СсылкаНаЭлемент.ПолучитьОбъект();
          СтароеНаименование=Элемент.Наименование;
          Элемент.Наименование=ВРег(Элемент.Наименование);
          Элемент.Записать();
          Сообщить("Элемент справочника "+ИмяСправочника+"
            с кодом "+Элемент.Код+" найден, наименование изменено
              с "+СтароеНаименование+" на "+Элемент.Наименование);
        КонецЕсли;
      КонецПроцедуры
```

Вот как выглядит работа этого кода, Рисунок 8.12.

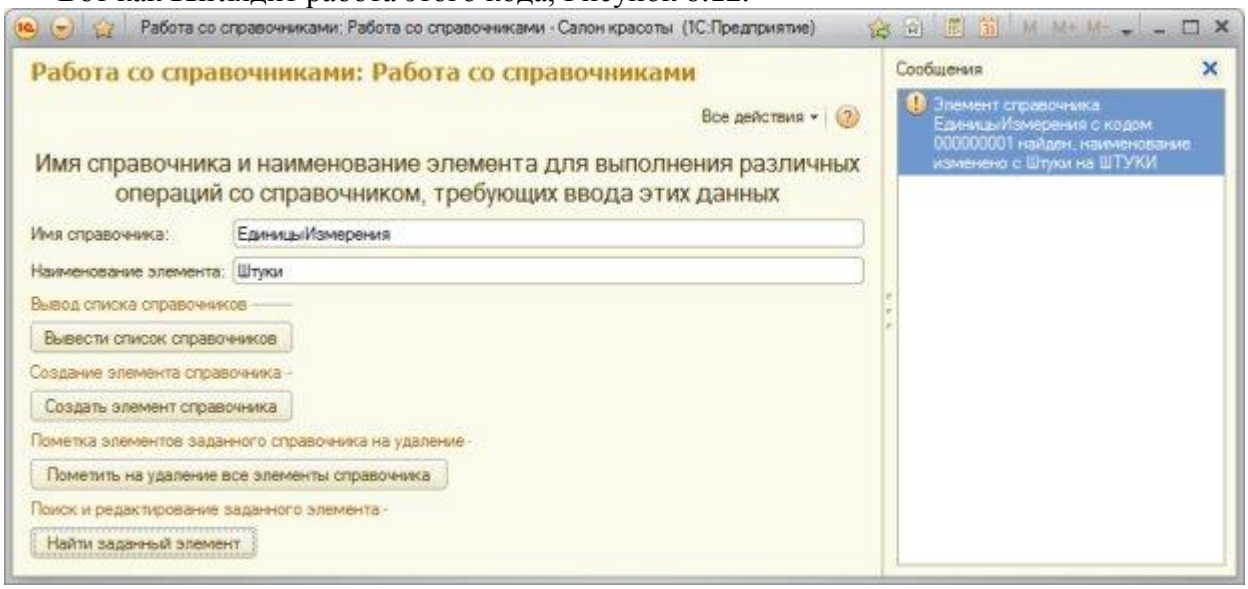


Рисунок 8.12. Результат работы кода по поиску и редактированию элемента справочника

Практическая работа №9 Применение функций обработки строковых данных

Цель: научиться разработке простых отчетов

ХОД РАБОТЫ:

1. Создадим в ветви дерева конфигурации **Отчеты** новый отчет, дадим ему имя **СписокКонтрагентов**, Рисунок 9.1.

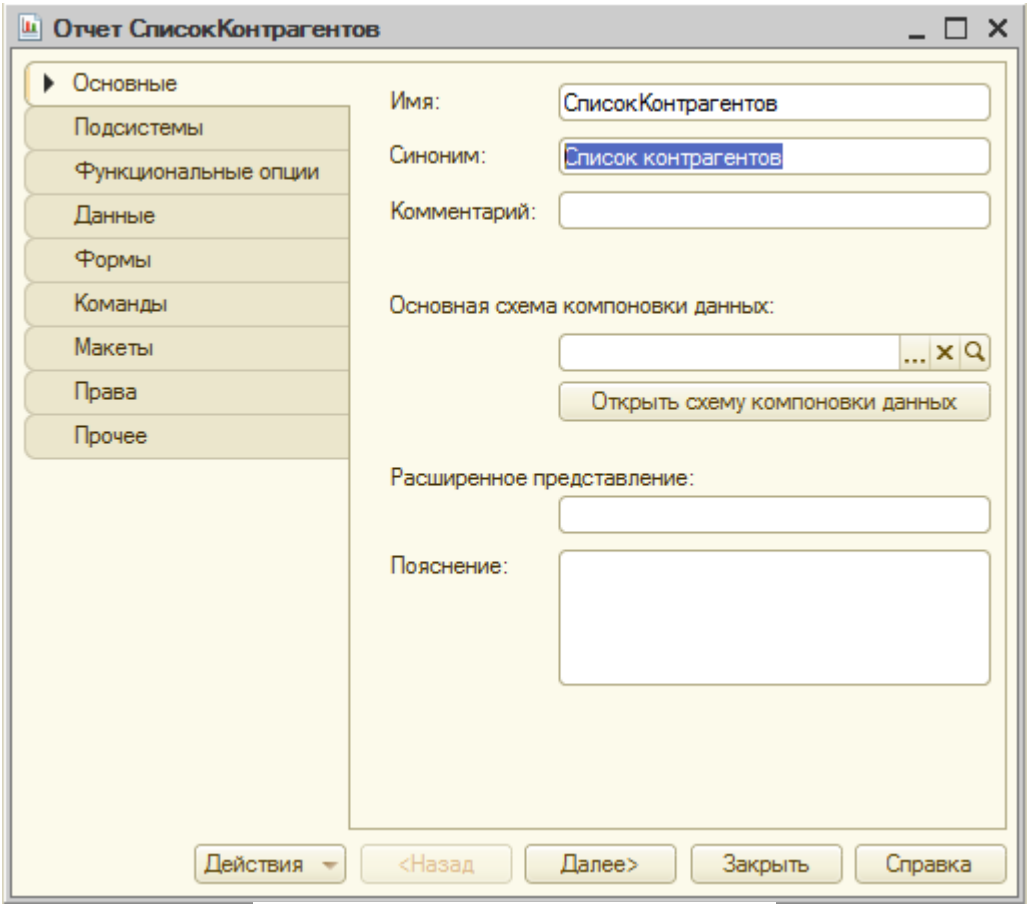


Рисунок 9.1- Создание нового отчета

2. Первым этапом работы над отчетом станет создание макета отчета. Макет позволяет заранее определить и оформить "блоки", из которых будет построен отчет.

3. Перейдем на закладку формы редактирования объекта **Макеты** и нажмем на кнопку **Добавить**. Появится окно конструктора макета, где нам предложат задать его имя (оставим имя по умолчанию – **Макет**), и тип макета – нас устроит **Табличный документ**, Рисунок 9.2.

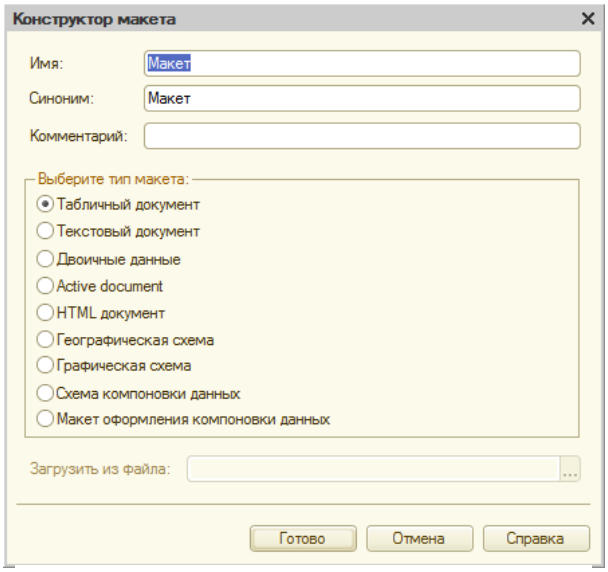


Рисунок 9.2- Создание макета для отчета

4. После нажатия на кнопку **Готово**, мы видим табличный редактор, Рисунок 9.3., очень напоминающий Microsoft Excel. Работая с ним, мы можем пользоваться стандартной палитрой свойств, а так же – панелями инструментов, в частности – **Форматирование**, **Табличный документ**, **Имена**. Наша задача сейчас – создать и отформатировать области, которые позже будут использованы для формирования готового отчета.

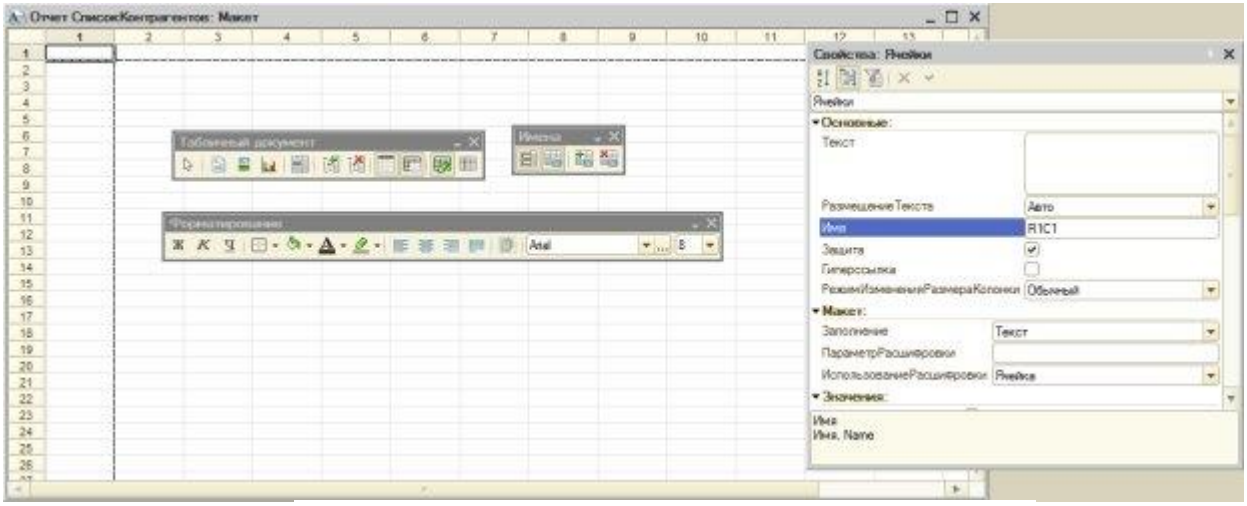


Рисунок 9.3. Средства редактирования макета отчета

5. На Рисунок 9.4 показан готовый макет.

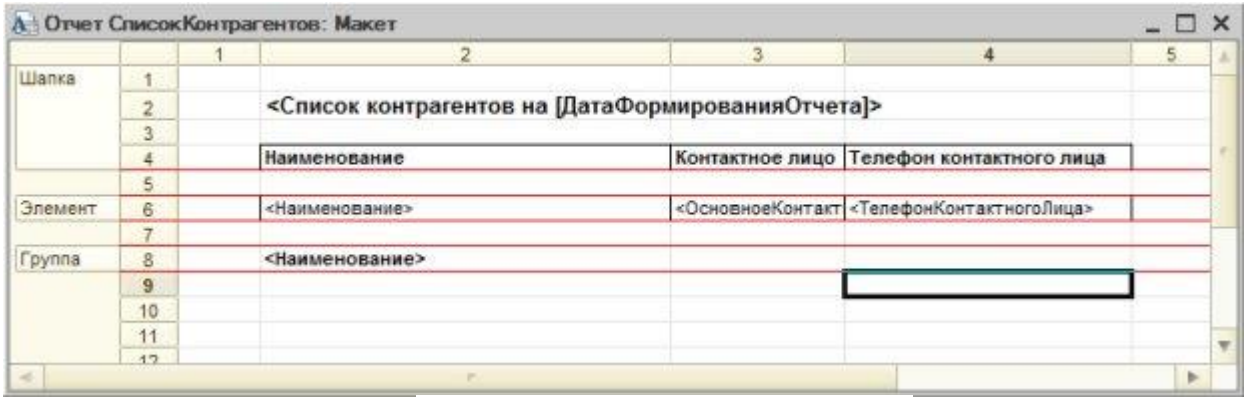


Рисунок 9.4- Готовый макет отчета

6. Ячейка 2,2 заполнена следующим образом: в нее сначала введен текст **"Список контрагентов на [ДатаФормированияОтчета]"**, после чего вызвано окно свойств этой ячейки, в которых, в свойстве **Заполнение** выбрано **Шаблон**, Рисунок 9.5.

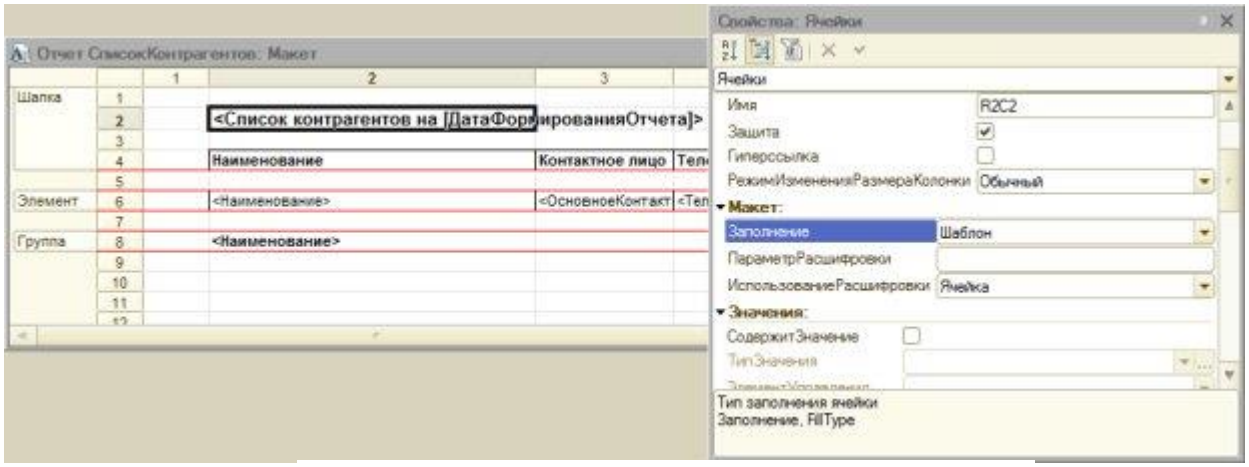


Рисунок 9.5. Настройка ячейки, содержащей шаблон

7. Параметр **ДатаФормированияОтчета** мы установим в текущую дату программно при формировании отчета.

8. Ячейки с 4,2 по 4,4 содержат обычный текст – он будет выводиться в качестве шапки таблицы.

9. И заголовок отчета и шапка таблицы объединены в область с именем **Шапка**. Для задания имени области достаточно выделить нужные ячейки (выделять нужно по заголовкам строк) и отредактировать в палитре свойств параметр **Имя выделенного диапазона**, или воспользоваться кнопкой **Назначить имя** панели инструментов **Имена.**

10. Область **Элемент** содержит три параметра – **Наименование**, **ОсновноеКонтактноеЛицо** и **ТелефонКонтактногоЛица**. После ввода в каждую из ячейку имен параметров, нужно выделить их (все вместе или по одной) и в окне свойств в поле **Заполнение** указать **Параметр.** К тексту в ячейках будут автоматически добавлены угловые скобки (<>), что позволяет визуально определить наличие в ячейке параметра.

11. Область **Группа** содержит лишь параметр **Наименование.**

Обратите внимание на имена параметров – они соответствуют именам реквизитов справочника, которыми мы собираемся их заполнять.

Ячейки в шаблоне можно форматировать – задавать их границы, оформление текста, выравнивание и т.д.

12. Теперь приступим к созданию формы отчета. Перейдем на вкладку **Формы** окна редактирования объекта, добавим новую форму отчета, оставим все настройки в состоянии по умолчанию и нажмем **Готово**. Добавим, на вкладке **Реквизиты** редактора форм новый реквизит, назовем его **ТабличныйДокумент**, выберем для него тип**ТабличныйДокумент**. Перетащим созданный реквизит в поле **Элементы**. В состав команд формы добавим новую команду, зададим ей имя **СформироватьОтчет** и так же переместим в поле **Элементы**. В итоге у нас получится форма, выглядящая так, как показано на Рисунок 9.6.

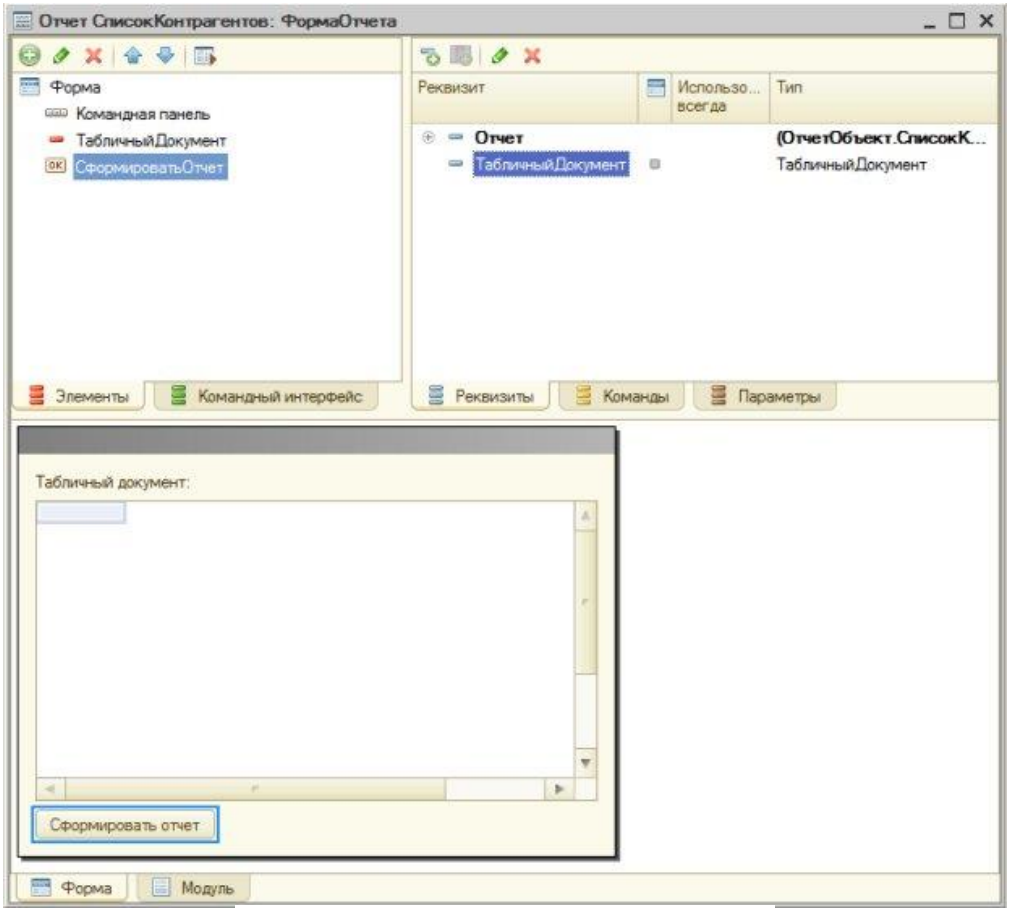


Рисунок 9.6- Настройка формы отчета

13. При реализации метода в виде процедуры, нам придется передать в него в качестве параметра наш реквизит **ТабличныйДокумент**. По умолчанию параметры передаются по ссылке, то есть, работать процедура будет непосредственно с нашим реквизитом.

14. При реализации метода в виде функции мы можем ничего не передавать в него, сформировать внутри функции табличный документ и вернуть уже заполненный документ в точку вызова, присвоив его нашему реквизиту **ТабличныйДокумент**.

15. Реализуем метод в виде функции. Готовый код формирования отчета (Рисунок 9.7) будет выглядеть следующим образом:

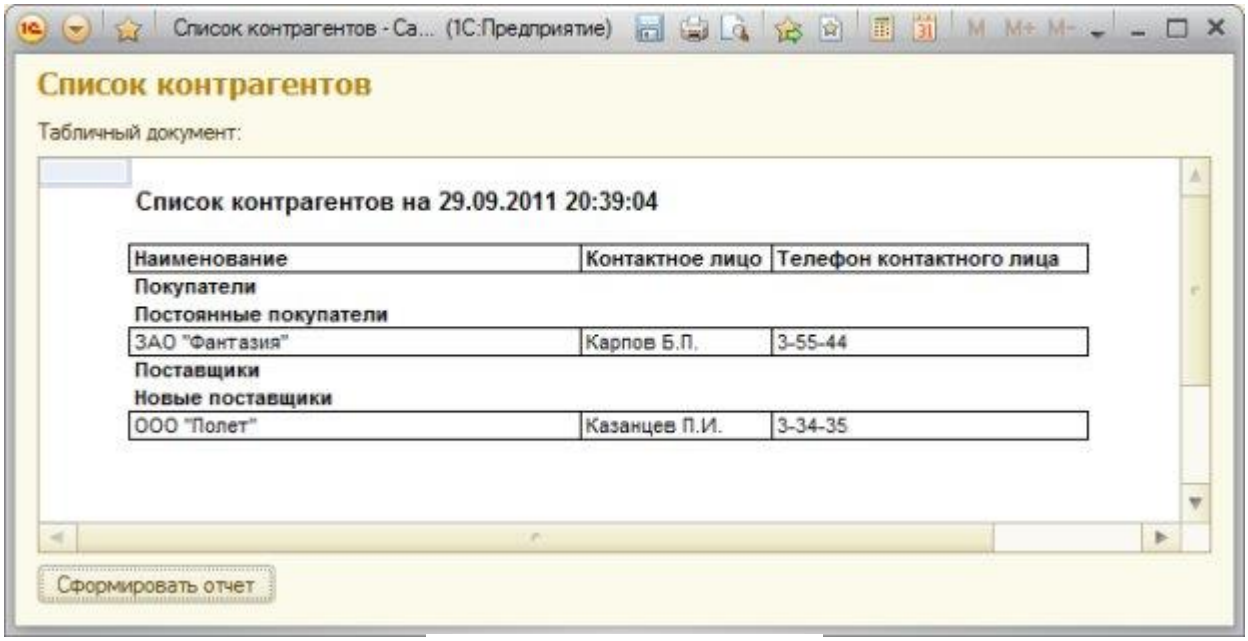


Рисунок 9.7- Готовый отчет

```
&НаКлиенте
Процедура СформироватьОтчет(Команда)
  ТабличныйДокумент=СформироватьОтчетНаСервере();
КонецПроцедуры

&НаСервереБезКонтекста
Функция СформироватьОтчетНаСервере()
  ТабличныйДокумент=Новый ТабличныйДокумент();
  Макет=Отчеты.СписокКонтрагентов.ПолучитьМакет("Макет");
  Шапка=Макет.ПолучитьОбласть("Шапка");
  Элемент=Макет.ПолучитьОбласть("Элемент");
  Группа=Макет.ПолучитьОбласть("Группа");
  Шапка.Параметры.ДатаФормированияОтчета=ТекущаяДата();
  ТабличныйДокумент.Вывести(Шапка);
  Выборка=Справочники.Контрагенты.ВыбратьИерархически();
  Пока Выборка.Следующий() Цикл
    Если Выборка.ЭтоГруппа Тогда
      Область=Группа;
    Иначе
      Область=Элемент;
    КонецЕсли;
    Область.Параметры.Заполнить(Выборка);
    ТабличныйДокумент.Вывести(Область);
  КонецЦикла;
  Возврат (ТабличныйДокумент);
КонецФункции
```

16. Для формирования простейших отчетов, пользователь может воспользоваться стандартной функциональностью, присутствующей в 1С:Предприятие 8. Для этого, открыв, например, список справочника, он может выполнить команду **Все действия > Вывести список**. Появится окно **Вывести список**, Рисунок 9.8, где в поле **Выводить в** можно выбрать либо **Табличный документ** (его обычно и используют), либо – **Текстовый документ.**

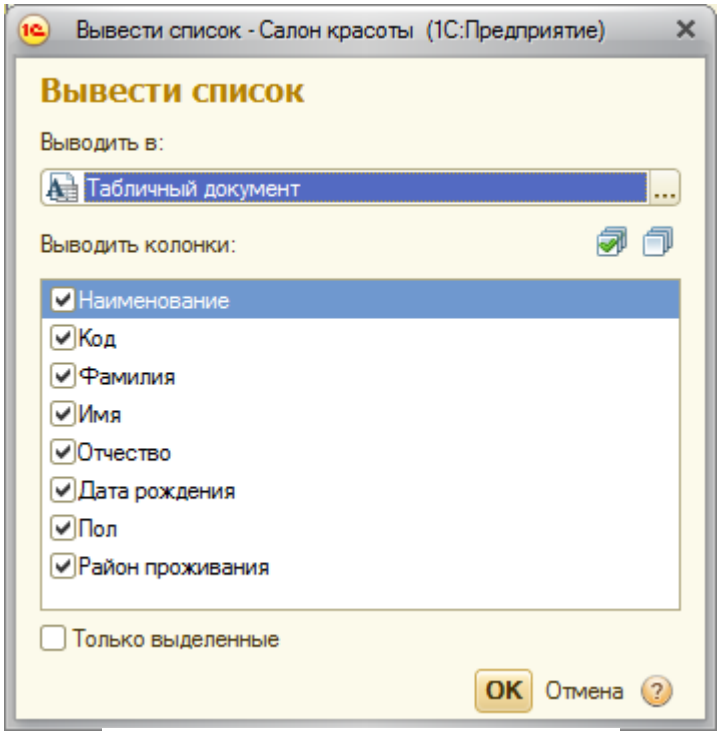


Рисунок 9.8. Окно Настройка списка

17. В поле **Выводить колонки** можно настроить состав выводимых в документ колонок (в нашем случае команда выполнена для справочника **ФизическиеЛица**). После нажатия на **ОК** выбранные данные оформляются в виде табличного документа, а с помощью команды **Файл > Сохранить как**, Рисунок 9.9., этот документ можно сохранить в нужном формате для дальнейшей обработки в других приложениях.

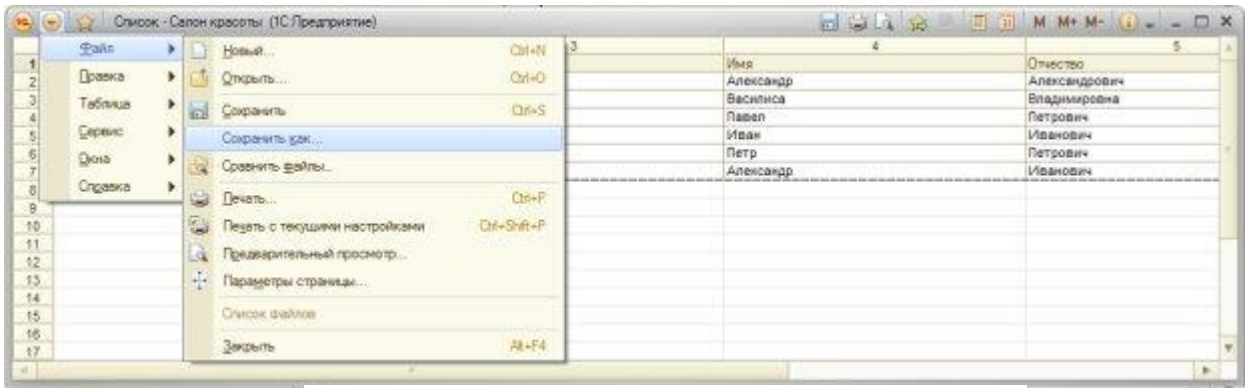


Рисунок 9.9-. Вывод данных в табличный документ

Практическая работа №10 Применение математических функций

Цель: научиться созданию регистров накопления

ХОД РАБОТЫ:

1. Добавим в нашу конфигурацию еще один справочник. Дадим ему имя **Подразделения**, добавим в состав подсистем **БухгалтерскийУчет**, **УчетРаботыМастеров** и **РасчетЗаработнойПлаты**. Увеличим длину наименования на закладке **Данные** до 100 символов. Сделаем справочник иерархическим – на закладке **Иерархия** установим флаг **Иерархический справочник**, параметр **Вид иерархии** установим в значение **Иерархия элементов**, Рисунок 10.1.

Иерархия элементов вполне логична для справочника **Подразделения**, так как одни подразделения могу включать в себя другие, и, при этом, вполне самостоятельны, их можно выбирать при заполнении, например, реквизитов других справочников, в то время, как при иерархии групп и элементов, группы играют лишь вспомогательную роль для организации информации внутри справочника.

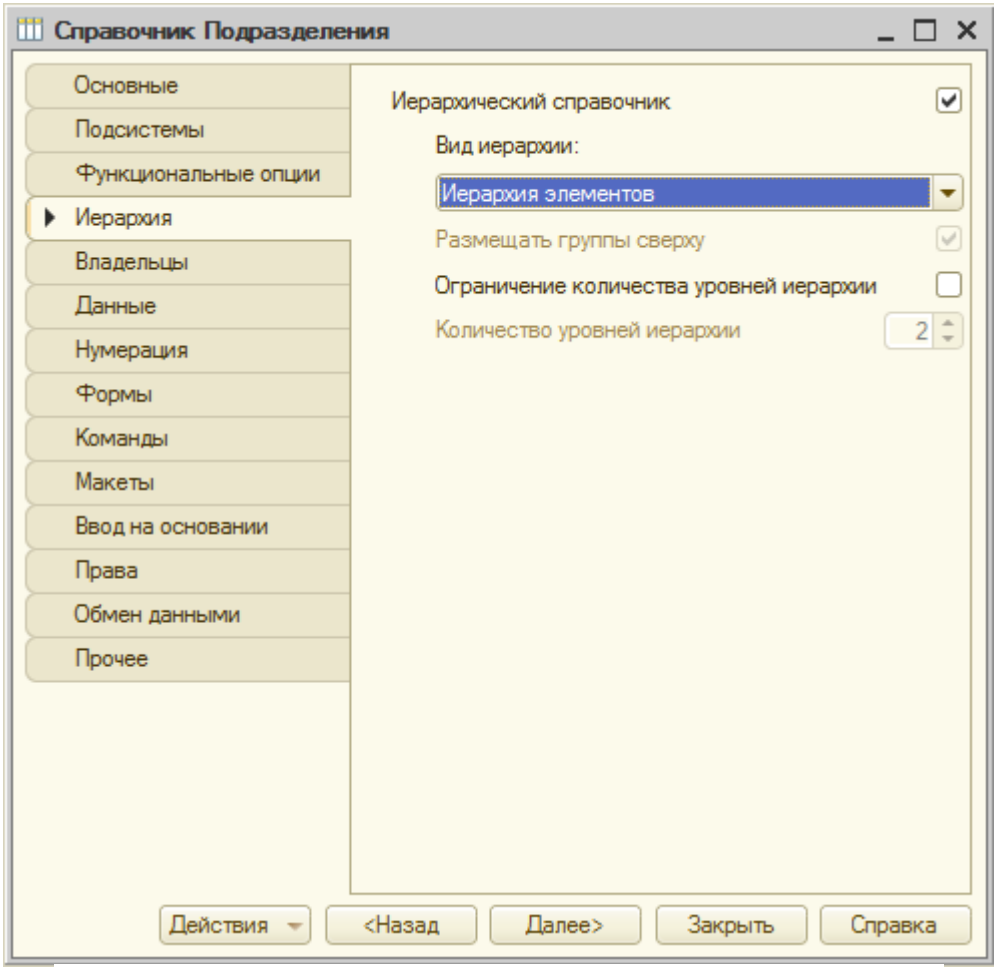


Рисунок 10.1- Настройка иерархии справочника Подразделения

2. Кроме того, в справочник **Подразделения** мы добавим несколько предопределенных элементов. Эти элементы справочника задаются в **Конфигураторе**, пользователь обладает лишь ограниченными возможностями по управлению ими, в частности, не может их удалить. Такие элементы обычно создают для того, чтобы ими можно было удобно и надежно оперировать в программном коде, не опасаясь того, что пользователь удалит их.

Для этого перейдем на вкладку окна редактирования объекта **Прочее** и нажмем на вкладку **Предопределенные**. В окне ввода предопределенных элементов справочника введем следующие (Рисунок 10.2.):

- Администрация
- Бухгалтерия
- Парикмахерская

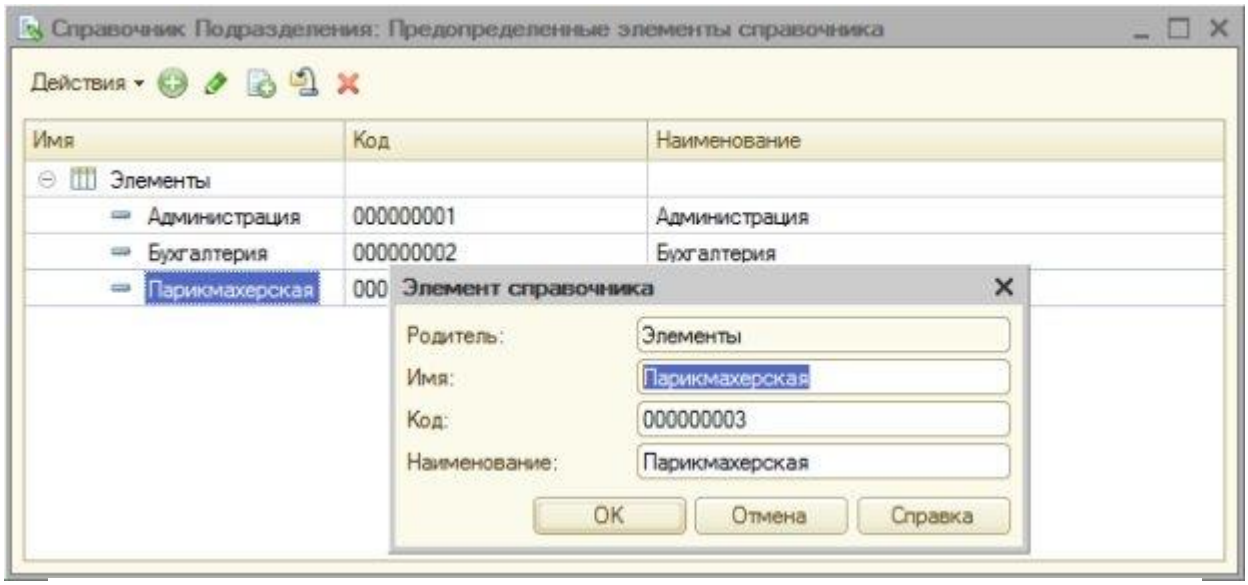


Рисунок 10.2- Создание предопределенных элементов справочника Подразделения

3. Доработаем справочник **Сотрудники**. Снабдим его следующими реквизитами, Рисунок 10.3.:

Имя: ФизическоеЛицо, **Тип**: СправочникСсылка.ФизическиеЛица

Имя: Подразделение, **Тип**: СправочникСсылка.Подразделения

Имя: Расчетчик, **Тип**: Булево

Имя: Пользователь, **Тип**: Строка, длина 50.

Увеличим длину **наименования** до 50 символов.

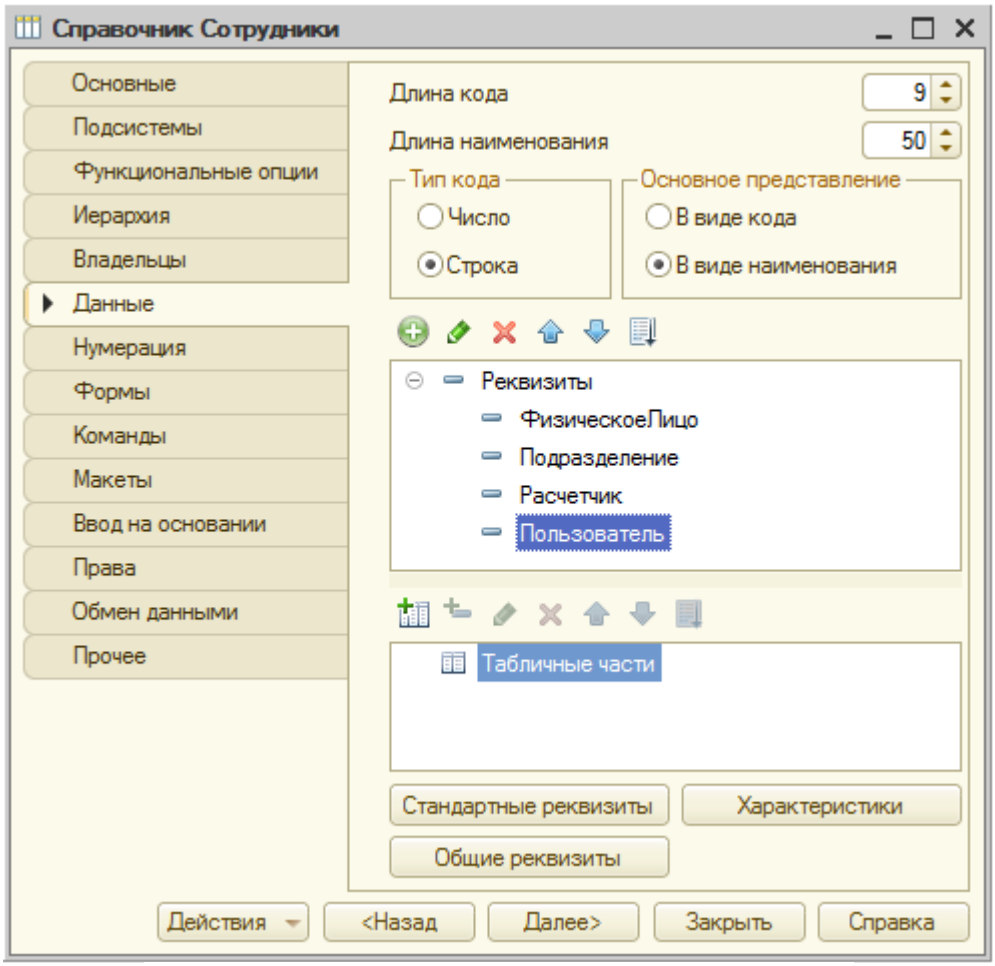


Рисунок 10.3. Реквизиты справочника Сотрудники

4. Мы хотели бы, чтобы наименование сотрудника в данном справочнике формировалось бы автоматически и состояло бы из ФИО физического лица и подразделения, в котором работает сотрудник. Создадим форму элемента справочника и, для элемента формы **Наименование**, снимем флаг **Доступность**, Рисунок 10.4.

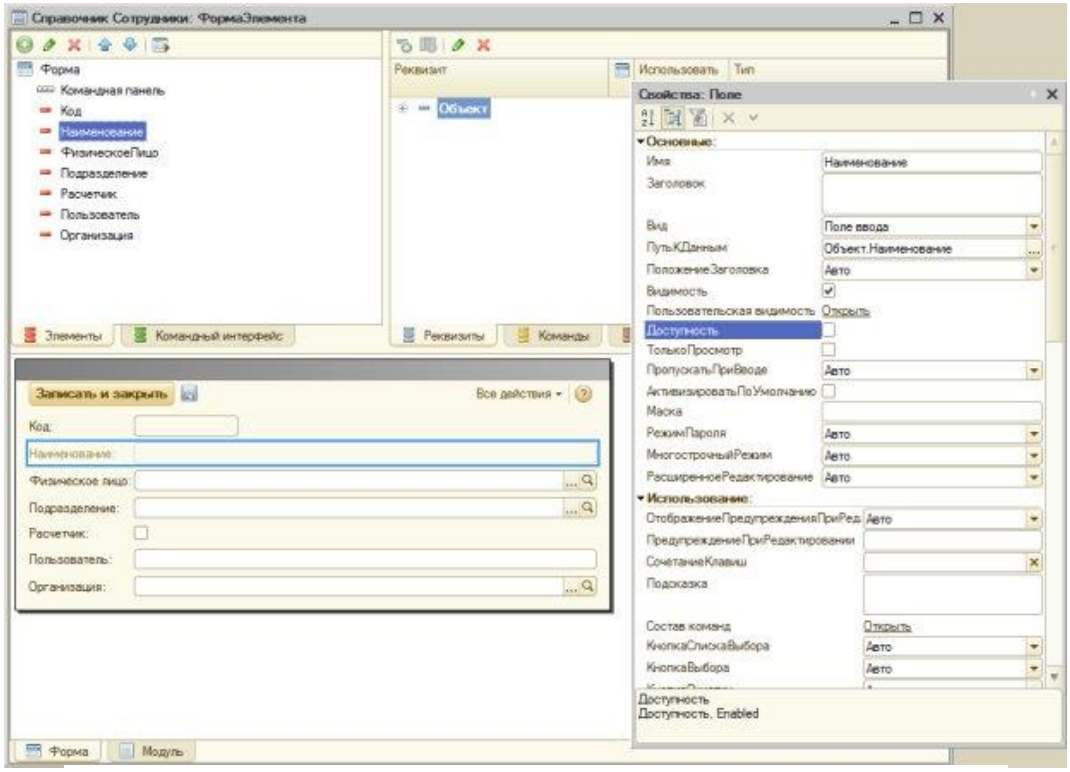


Рисунок 10.4. Настройка формы элемента справочника Сотрудники

5. Теперь подумаем над тем, как автоматически заполнить поле **Наименование** на основе данных полей **Физическое лицо** и **Подразделение**. Сделать это можно различными способами, мы реализуем следующую функциональность: перехватим события изменения полей **Физическое лицо** и **Подразделение** и вызовем в обработчике каждого из этих событий процедуру, заполняющую поле **Наименование**. Так пользователь, заполняющий элемент справочника, сможет сразу же увидеть результаты формирования наименования.

Нашей задаче отвечает следующий код:

```
&НаКлиенте
Процедура ФизическоеЛицоПриИзменении(Элемент)
  СформироватьНаименование();
КонецПроцедуры

&НаКлиенте
Процедура ПодразделениеПриИзменении(Элемент)
  СформироватьНаименование();
КонецПроцедуры

Процедура СформироватьНаименование()
  Объект.Наименование=Объект.ФизическоеЛицо.Наименование +" ("+
  Объект.Подразделение.Наименование+") ";
КонецПроцедуры
```

Результаты работы созданного нами механизма показаны на Рисунок 10.5.

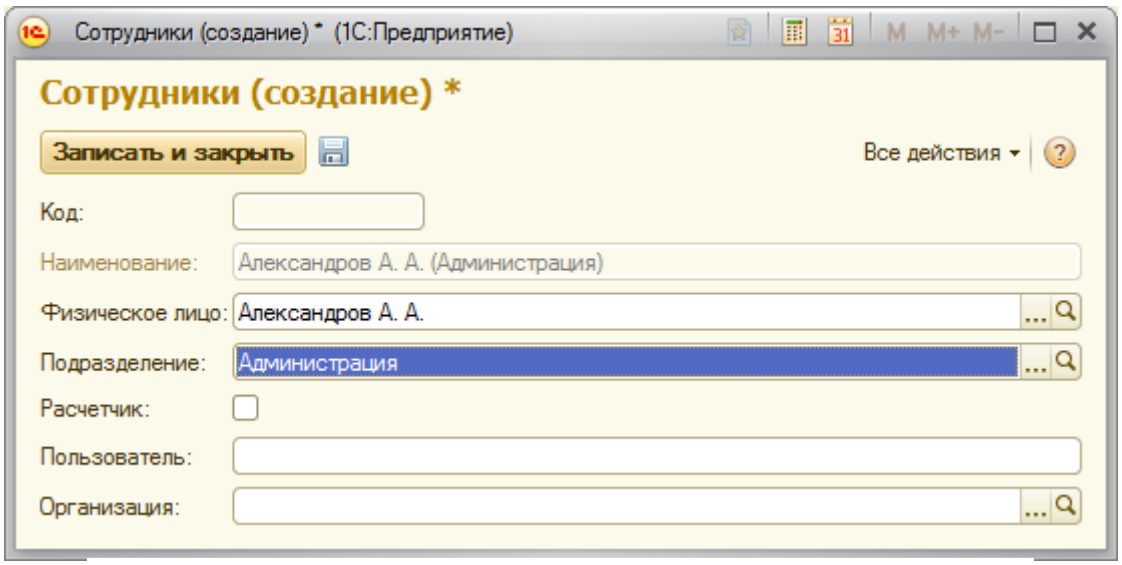


Рисунок 10.5- Настройка формы элемента справочника Сотрудники

В этой работе мы познакомились с созданием обработок и простых отчетов. Так же мы подробно рассмотрели объектную модель 1С:Предприятие 8.2., предназначенную для работы со справочниками и создали справочник с иерархией элементов.

Практическая работа №11 Построение выражений

Цель: научиться разработке документов, работе с регистрами накопления и построению отчетов с использованием системы компоновки данных (СКД).

ХОД РАБОТЫ:

1. Для описания документов в дереве конфигурации имеется отдельная ветвь – **Документы**. В одной из предыдущих лекций мы создали один документ – **ПоступлениеМатериалов**. Сейчас мы займемся работой с ним. Для начала определимся с целью использования этого документа. Мы планируем с его помощью отражать в системе поступление материалов. Исходя из этих целей, нам понадобятся следующие реквизиты документа (Рисунок 11.1.), которые мы зададим на вкладке **Данные** окна редактирования объекта:

Имя: Контрагент, Тип: СправочникСсылка.Контрагенты

Имя: ОтветственныйСотрудник: Тип: СправочникСсылка.Сотрудники

Добавим в состав табличных частей нашего документа новую табличную часть с именем **Материалы** и следующими реквизитами:

Имя: Номенклатура, Тип: СправочникСсылка.Номенклатура

Имя: Цена, Тип: Число, длина 10, точность 2

Имя: Количество, Тип: Число, длина 10, точность 3

Имя: Сумма, Тип: Число, длина 10, точность 2

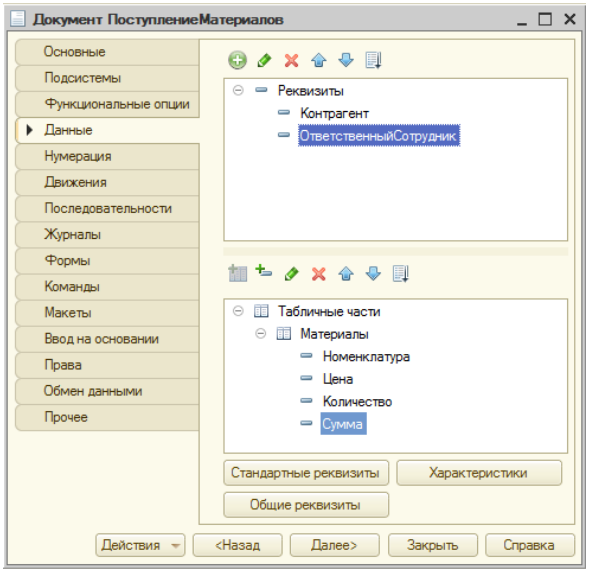


Рисунок 11.1- Настройка состава реквизитов документа

2. На закладке **Нумерация**, Рисунок 11.2., можно задать параметры нумерации документов.

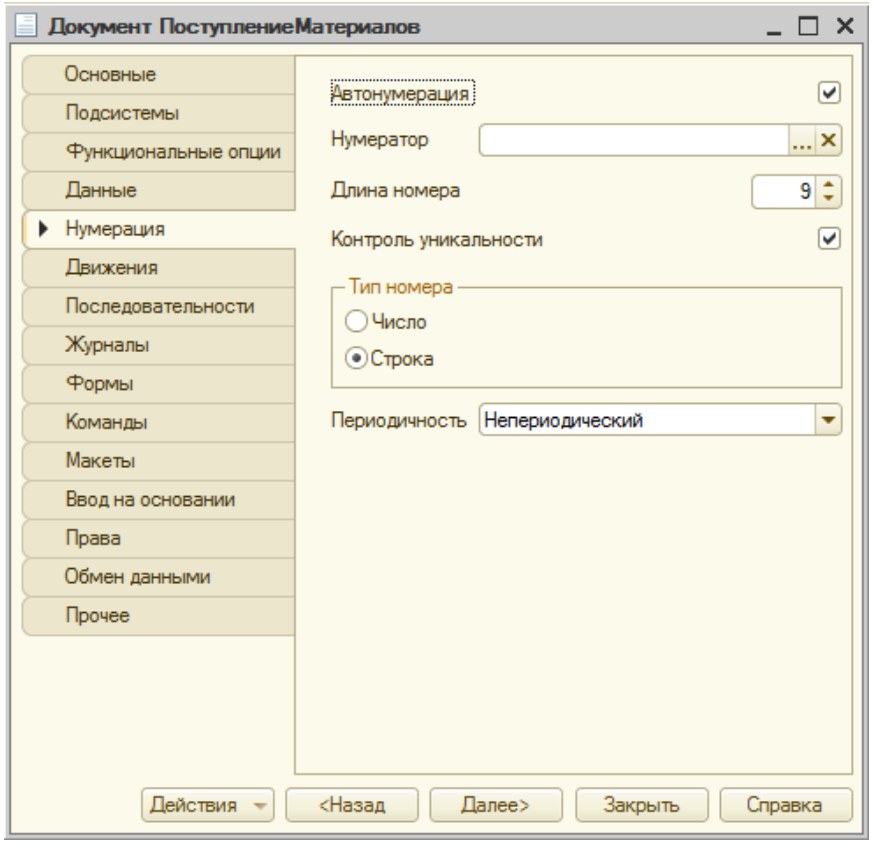


Рисунок 11.2- Настройка параметров нумерации документа

В данном случае документы будут нумероваться автоматически с контролем уникальности номеров.

3. Закладка **Движения**, Рисунок 11.3., позволяет управлять проведением документа.

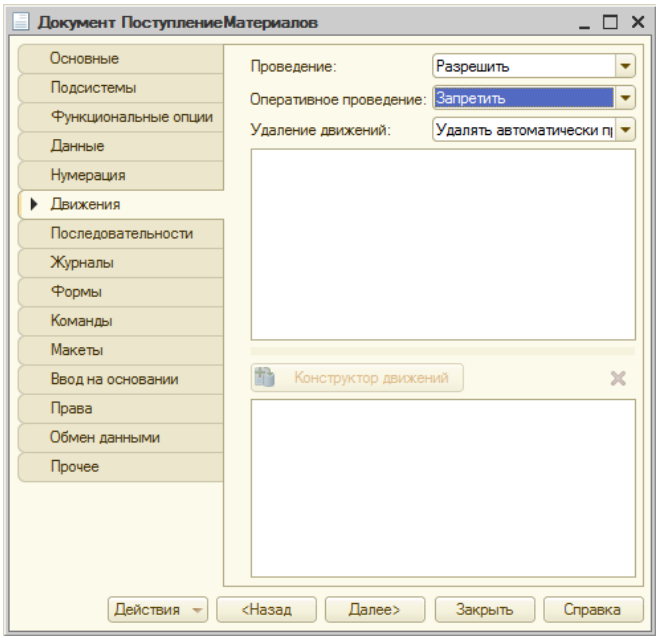


Рисунок 11.3- Настройка параметров проведения документа

4. Если проведение документа запрещено – то пользователь сможет лишь сохранить документ в базе данных. Других воздействий на информационную базу такой документ не произведет. Например, такое поведение может быть характерно для документов, вроде выписанных счетов, которые сами по себе воздействия на учет не производят, но их важно хранить в системе для того, чтобы "помнить" о том, какие счета выписаны, важно иметь возможность формировать их печатные формы.

Но то, что счет выписан, еще не гарантирует то, что счет будет оплачен, то, что товары, указанные в выписанном счете будут действительно отгружены покупателю. Если продолжить пример с выписанным счетом и перейти на вкладку **Ввод на основании**, Рисунок 11.4., то окажется, что эта вкладка позволяет настроить ввод одного документа на основании другого.

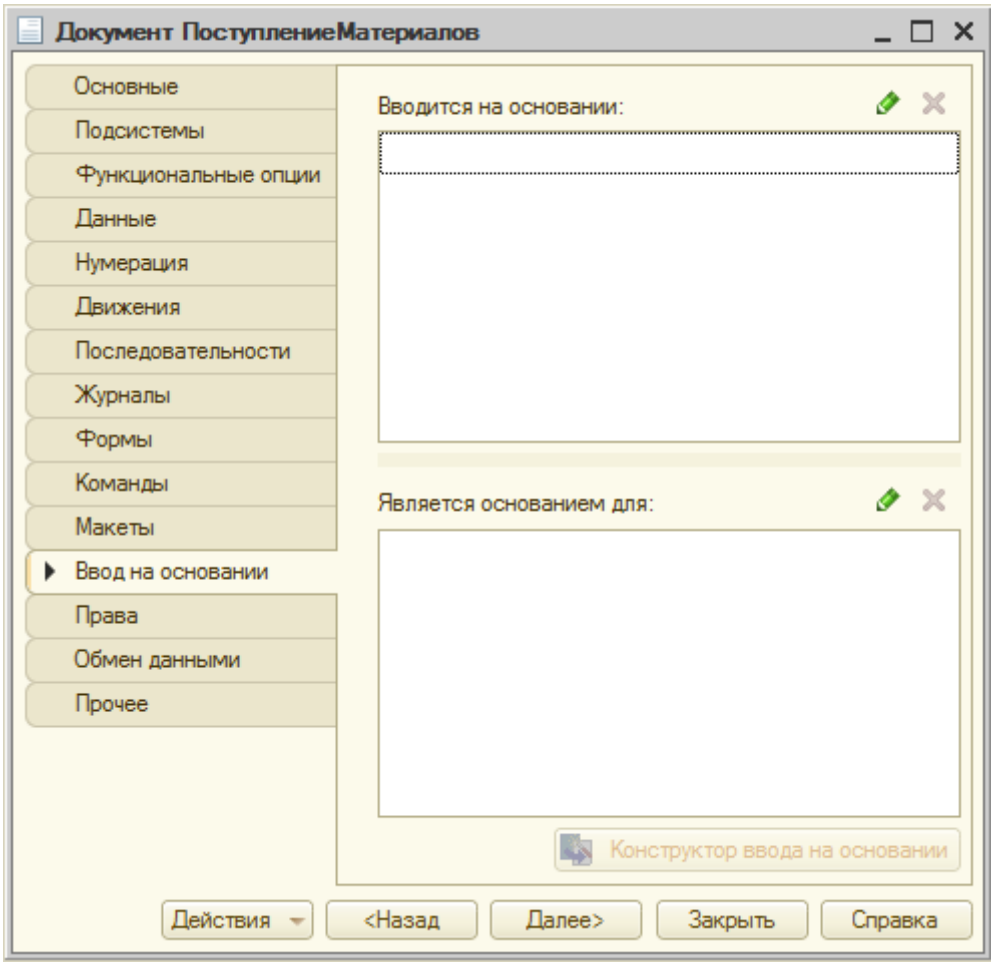


Рисунок 11.4. Настройка параметров ввода на основании

5. Если, например, мы имеем дело с документом наподобие "**Отгрузка материалов**", окажется, что такой документ вполне логично будет вводить на основании документа "**Счет**" - после оплаты этого счета и фактической отгрузки материалов. Документ отгрузки, в отличие от счета, фиксирует уже свершившийся факт хозяйственной жизни, который должен оказать воздействие на состояние информационной базы. Такой документ должен проводиться – то есть – делать записи в соответствующие регистры.

6. На данном этапе мы можем запустить систему, попытаться поработать с документом, используя автоматически сгенерированную форму, и посмотреть, все ли в данной форме нас устраивает.

7. Прежде чем продолжать работу с документом **ПоступлениеМатериалов**, приведите данные справочника **Номенклатура** в вашей информационной базе к виду, показанному в таблице 11.1.

Таблица 11.1. Данные справочника Номенклатура

Наименование	Единица измерения	Услуга	Группа
Парикмахерские услуги		Да	Да
Завивка	Час	Да	
Стрижка	Час	Да	
Парфюмерия		Нет	Да
Духи	Штука	Нет	
Одеколон	Штука	Нет	
Прочие материалы		Нет	Да
УФ-гель	Упаковка	Нет	
Спецодежда		Нет	Да
Одежда для парикмахера	Штука	Нет	
Уход за волосами		Нет	Да
Бальзам для волос	Штука	Нет	
Лак для волос	Упаковка	Нет	

8. На рисунке 11.5. вы можете видеть форму документа после ввода в нее некоторых данных.

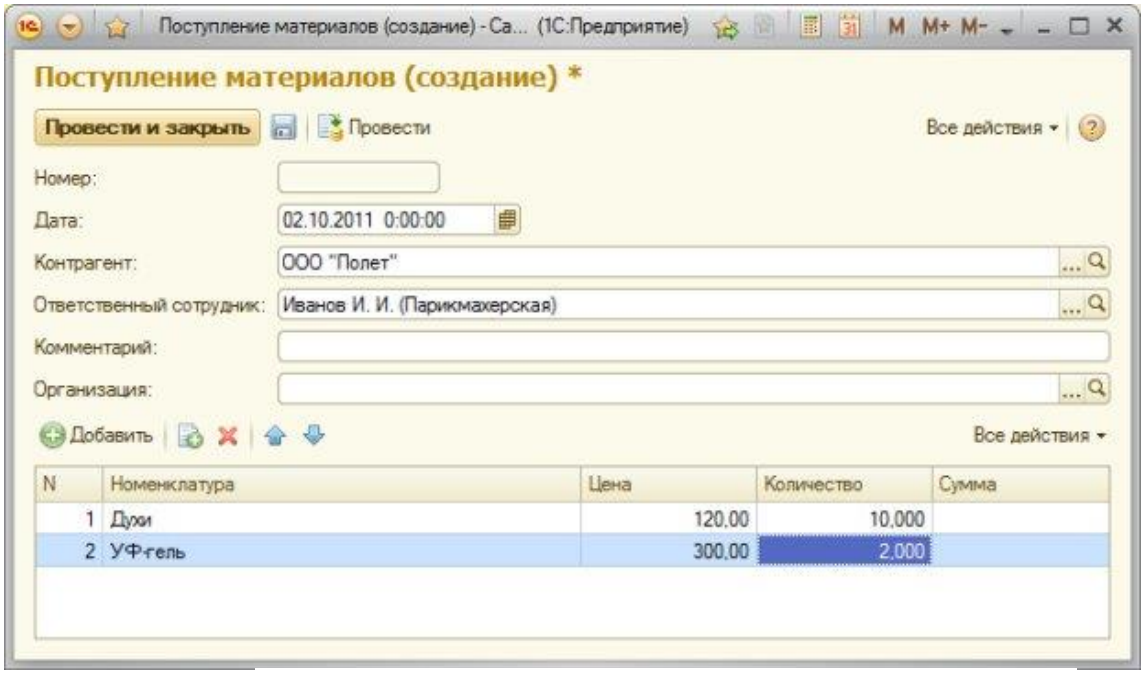


Рисунок 11.5. Заполнение документа Поступление товаров

9. Для того, чтобы автоматически заполнить поле сумма по каждой из строк табличной части, редактируемой пользователем, очевидно, что рассчитывать сумму имеет смысл либо после заполнения поля **Цена**, либо – после заполнения поля **Количество**, перехватив какие-либо события, имеющие отношение к редактируемой табличной части.

В нашем случае это должны быть события, генерируемые при изменении полей **Цена** или **Количество** при вводе данных в определенной строке. Для того, чтобы назначить обработчики подобных событий для определенных элементов табличной части, можно поступить так же, как мы поступали, назначая обработчики событий для любых других элементов формы (Рисунок 11.6.). Для начала, конечно же, нам нужно будет создать собственную форму документа, делается это на закладке **Формы** окна редактирования объекта. С параметрами, предложенными конструктором форм по умолчанию, можно согласиться.

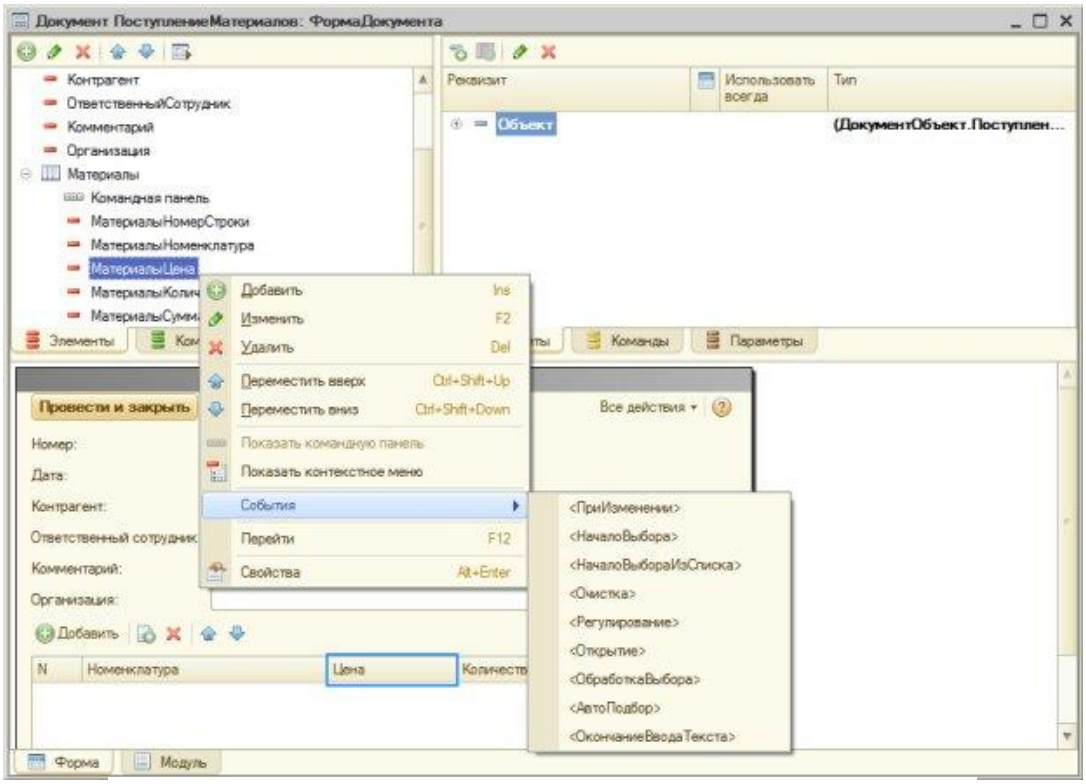


Рисунок 11.6- Назначение обработчика полю табличной части

10. Назначим обработчики событий **ПриИзменении** для полей **МатериалыЦена** и **МатериалыКоличество**.

Теперь нам нужно реализовать следующее: при работе в определенной строке таблицы, при вводе в нее данных, получить эту строку, и, при изменении цены или количества номенклатуры рассчитать сумму.

У табличных полей есть свойство **ТекущиеДанные**, которое, как раз, позволяет обращаться к текущей редактируемой строке. Данные редактируются на клиенте, поэтому мы вполне можем обойтись здесь без вызова серверных процедур, выполнив все необходимые действия на клиенте. Если вы хотите побольше узнать о том, что можно сделать с табличным полем из кода, как и в других случаях, помочь вам в этом могут инструменты отладки. Вот как, например, выглядит свойство **ТекущиеДанные** при срабатывании точки останова в коде модуля нашей формы при отладке кода, который будет представлен ниже, Рисунок 11.7.

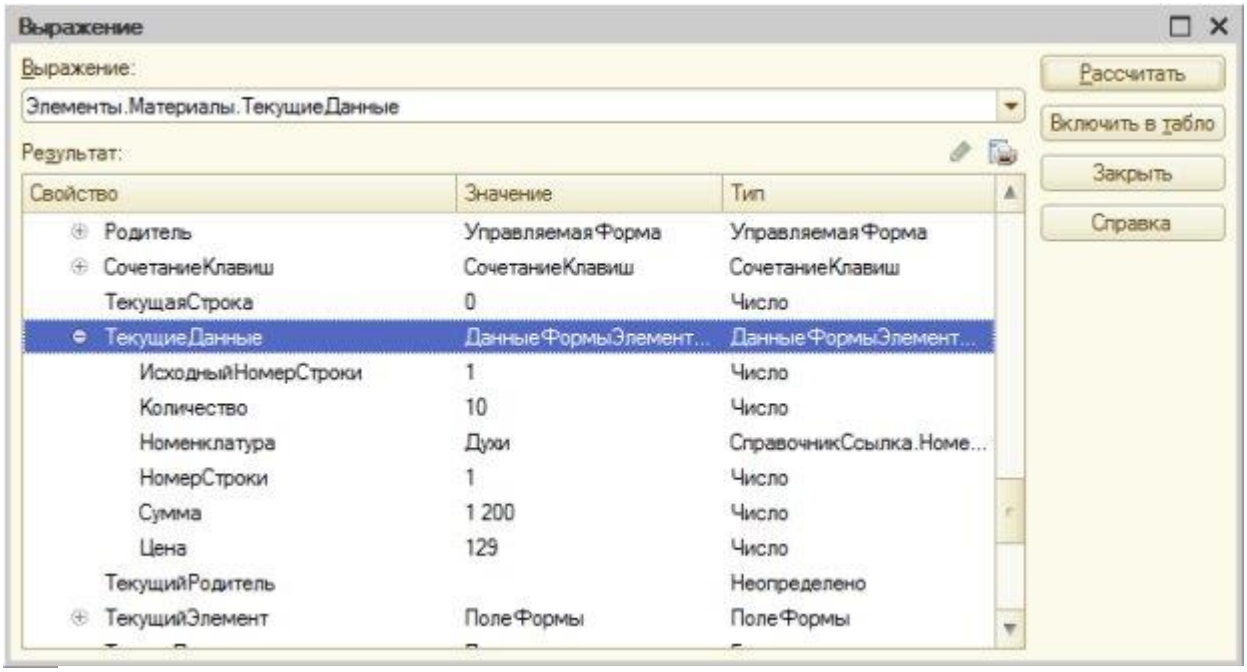


Рисунок 11.7- Просмотр свойства ТекущиеДанные в окне Выражение при отладке кода

Итак, наша задача может быть решена следующим образом:

```
&НаКлиенте
Процедура МатериалыЦенаПриИзменении(Элемент)
 РассчитатьСумму();
КонецПроцедуры

&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)
 РассчитатьСумму();
КонецПроцедуры

&НаКлиенте
Процедура РассчитатьСумму()
 ТекущаяСтрока=Элементы.Материалы.ТекущиеДанные;
 ТекущаяСтрока.Сумма=ТекущаяСтрока.Количество*ТекущаяСтрока.Цена;
КонецПроцедуры
```

Из пары обработчиков событий **ПриИзменении** вызывается клиентская процедура **РассчитатьСумму()**. Здесь мы получаем данные текущей строки через свойство **ТекущиеДанные** и вычисляем поле **Сумма**, перемножая данные в полях **Количество** и **Цена**.

При необходимости, мы можем редактировать поле **Сумма** независимо от значений полей **Цена** и **Количество**.

11. Вторая задача из тех, которые мы поставили себе выше, заключается в выводе на форму итоговых сведений по табличному полю. Ее можно реализовать различными способами, но лучше всего воспользоваться стандартными итоговыми показателями табличного поля, которые можно найти в составе табличного поля на закладке **Реквизиты** редактора форм, Рисунок 11.8.

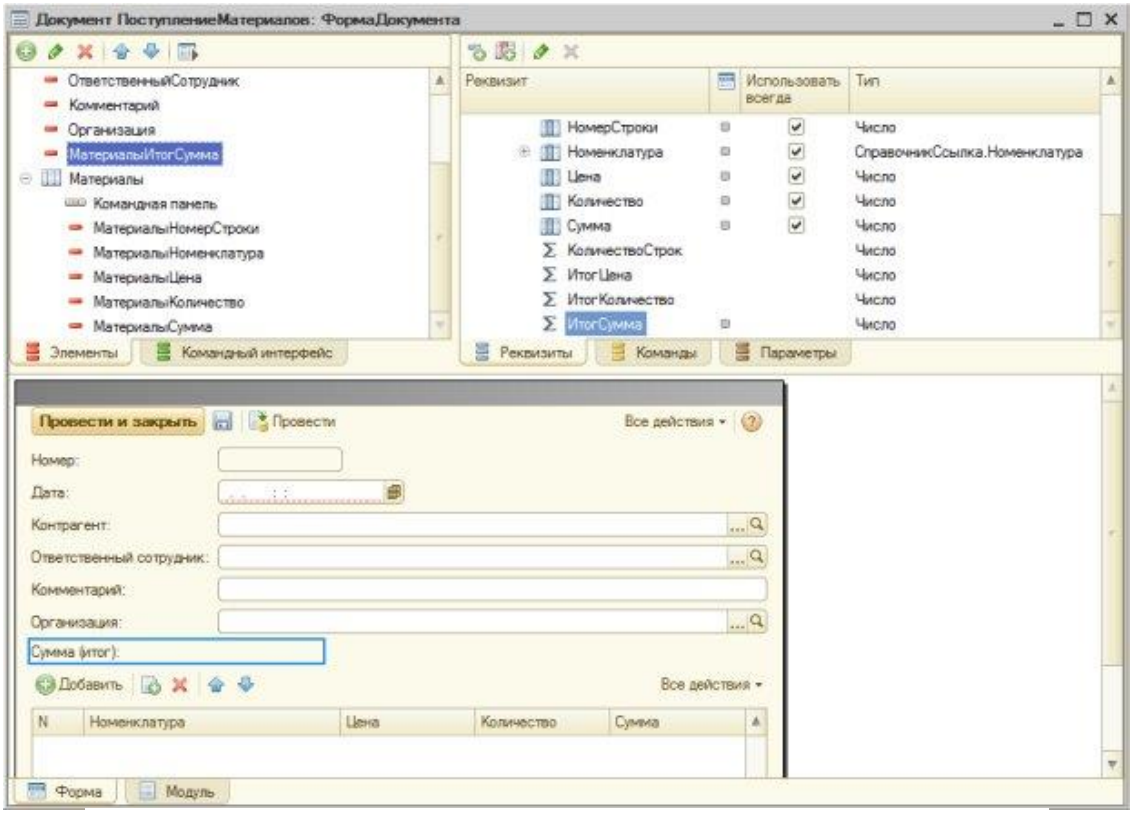


Рисунок 11.8. Вывод итогового показателя для поля Сумма на форму

Этот реквизит – **ИтогСумма** – нужно перетащить на вкладку **Элементы**. Он будет отображаться на форме, изменяясь при изменениях суммы в строках табличной части, Рисунок 11.9.

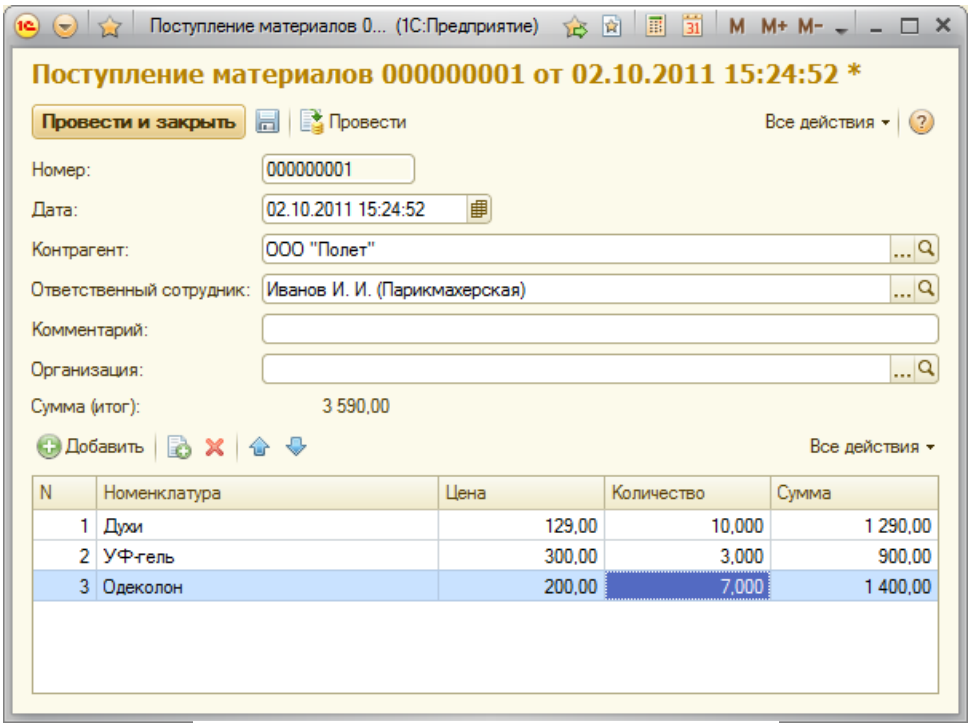


Рисунок 11.9- Форма после модификации

12. Сейчас займемся еще одним документом, который, являясь, по составу реквизитов и по особенностям устройства формы, очень похожим на документ **ПоступлениеМатериалов**,

выполняет противоположную ему функцию – а именно – отвечает за списание материалов. В нашей системе материалы выбывают при передаче их в производство. Мы вполне можем создать новый документ копированием предыдущего и изменением некоторых его реквизитов. Так и поступим. Скопируем документ и приведем состав его реквизитов к показанному на Рисунок 11.10.

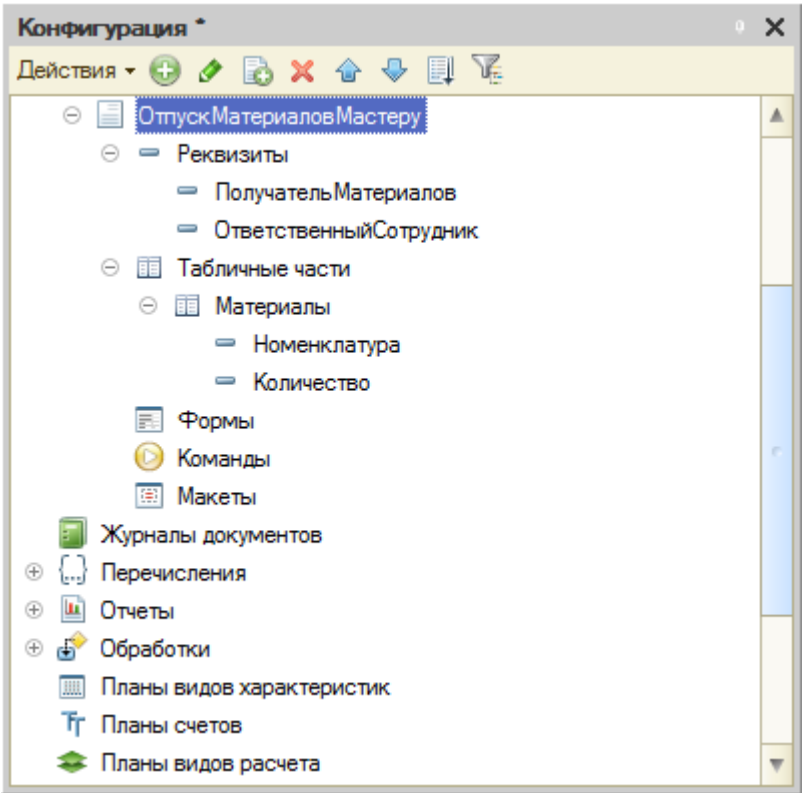


Рисунок 11.10. Создание документа ОтпускМатериаловМастеру

Включим данный документ в состав подсистемы **ОперативныйУчетМатериалов**, вместо реквизита **Контрагент** у него будет реквизит **ПолучательМатериалов** с типом**СправочникСсылка.Сотрудники**.

13. В табличной части документа мы используем лишь два реквизита – это **Номенклатура** и **Количество**. Показатели стоимости списываемой номенклатуры мы будем рассчитывать автоматически. При работе с этим документом нас вполне устроит форма, генерируемая автоматически.

Форма нашего нового документа будет выглядеть так, как показано на Рисунок 11.11.

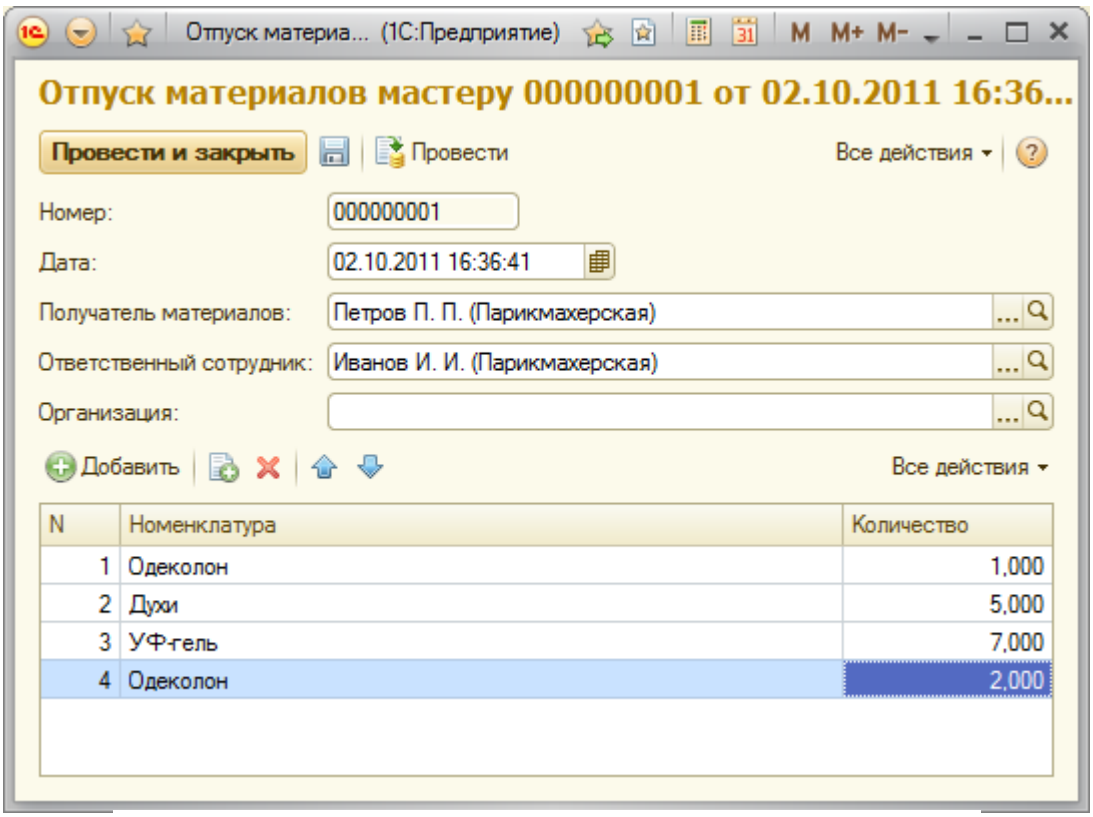


Рисунок 11.11- Документ ОтпускМатериаловМастеру в работе

Практическая работа № 12 Применение процедур и функций сеанса работы

Цель: научиться программированию регистров накопления и процедур работы с формой обработки

ХОД РАБОТЫ:

1. При планировании состава регистра накопления нужно понять, какие именно данные мы собираемся в нем хранить, после чего "разложить" эти данные по измерениям, ресурсам и реквизитам регистра.

Итак, нам нужно хранить следующие данные:

- Номенклатурная позиция
- Ответственный сотрудник, на котором числится данная позиция
- Количество номенклатуры
- Стоимость номенклатуры
- Данные о мастере, которому переданы материалы для использования.

2. Создадим новый регистр накопления, назовем его **ОстаткиМатериалов**, параметр **Вид регистра** оставим в значении **Остатки**, Рисунок 12.1.

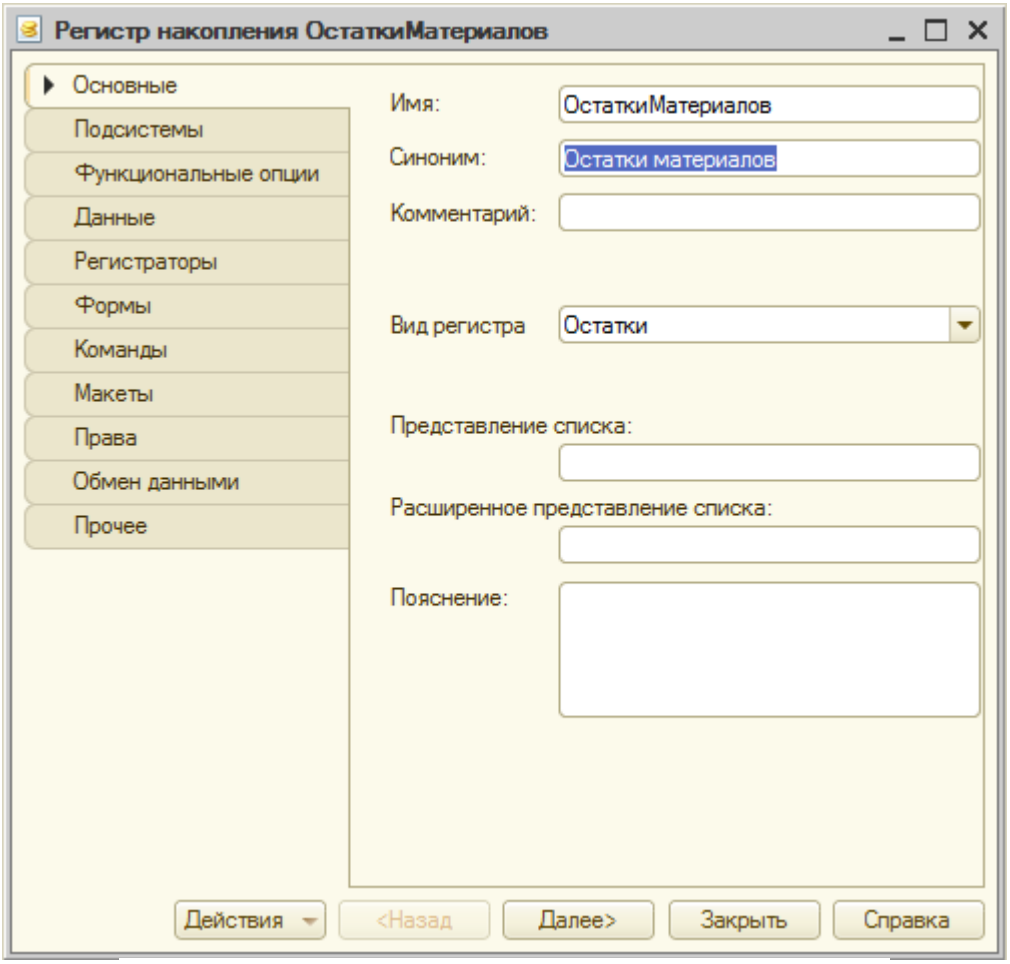


Рисунок 12.1- Регистр накопления ОстаткиМатериалов

3. Включим регистр накопления в состав подсистемы **ОперативныйУчетМатериалов**.

4. На вкладке **Данные** создадим следующие измерения, ресурсы и реквизиты:

Измерения:

Имя: Номенклатура, Тип: СправочникСсылка.Номенклатура, Запрет незаполненных значений – установлено.

Имя: ОтветственныйСотрудник, Тип: СправочникСсылка.Сотрудники, Запрет незаполненных значений – установлено.

Ресурсы

Имя: Количество, Тип: число, длина 10, точность 3

Имя: Сумма, Тип: число, длина 10, точность 2

Реквизиты:

Имя: ПолучательМатериалов, Тип: СправочникСсылка.Сотрудники

Обратите внимание на имена этих реквизитов, на их типы, а так же – на стандартные реквизиты регистра (Рисунок 12.2.) – эти данные пригодятся нам при работе над процедурой проведения документа.

Исключим из состава реквизитов регистра общий реквизит **Организация**. Сейчас в нем нет необходимости. Для организации хранения данных в регистре в разрезе различных организаций нам понадобилось бы новое измерение – Организация, благодаря наличию которого мы смогли бы работать с материалами различных организаций.

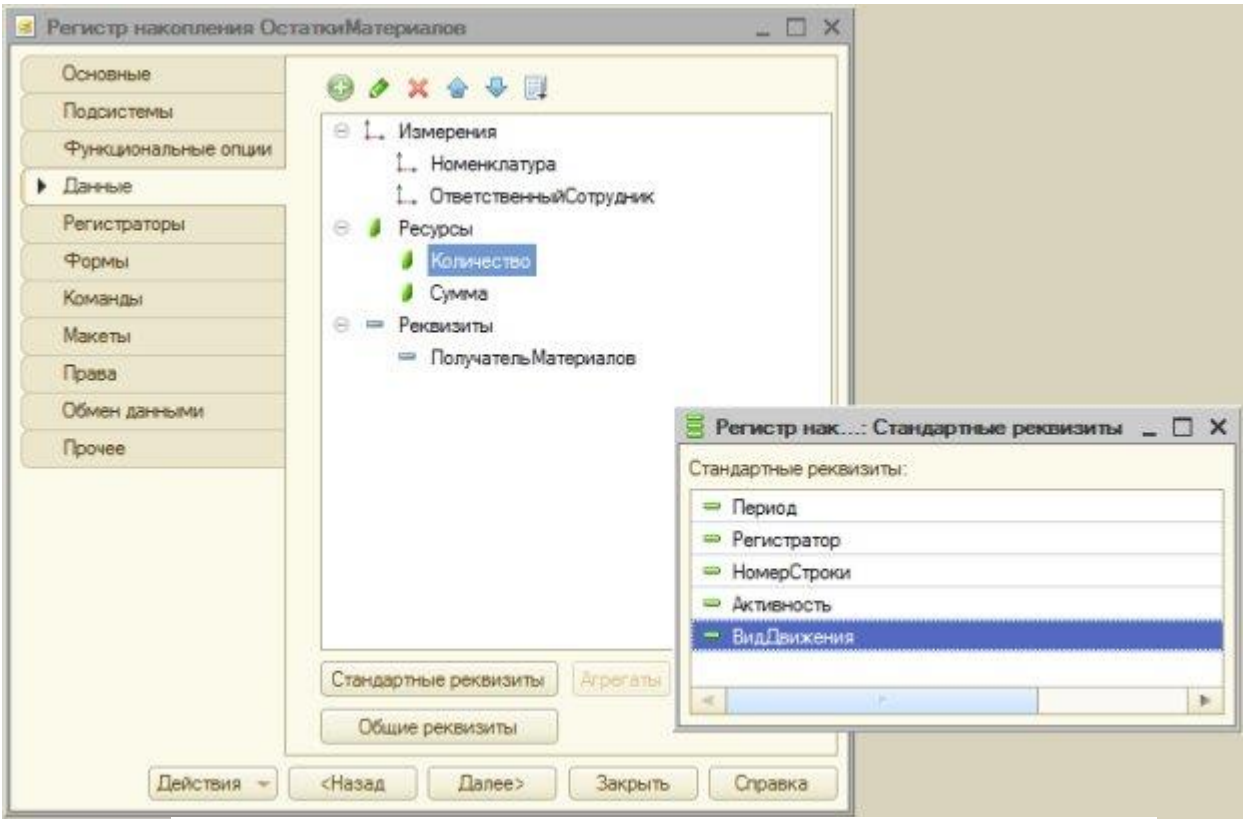


Рисунок 12.2- Регистр накопления ОстаткиМатериалов, состав данных

5. Перейдем на вкладку **Регистраторы** окна редактирования объекта и выберем в качестве документов-регистраторов документы – **ПоступлениеМатериалов** и **ОтпускМатериаловМастеру**.

На данном этапе настройка регистра накопления окончена, перейдем к настройкам документов. Начнем с документа **ПоступлениеМатериалов.**

6. Откроем окно редактирования документа **ПоступлениеМатериалов**, перейдем на вкладку **Движения** (Рисунок 12.3.) и нажмем на кнопку **Конструктор движений**

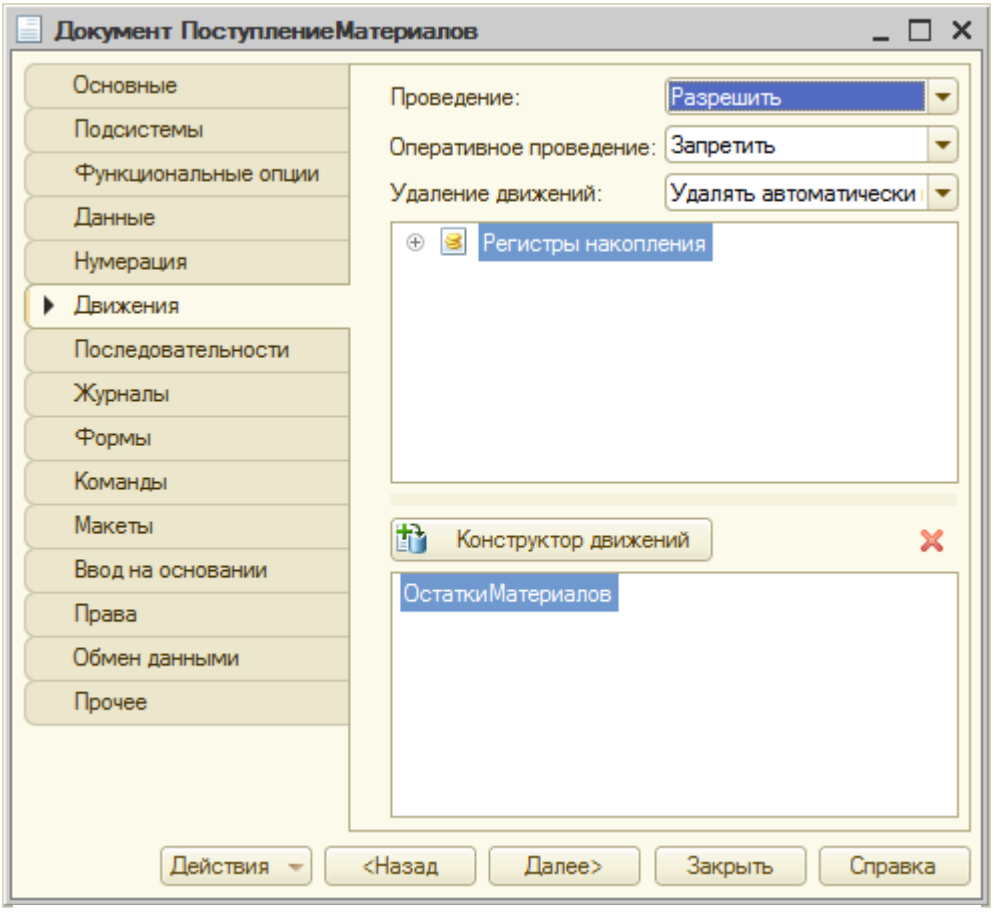


Рисунок 12.3. Документ ПоступлениеМатериалов, вкладка Движения

7. В конструкторе, выберем тип движения регистра – **Приход**, в поле **Табличная часть** укажем табличную часть документа **Материалы**, нажмем на кнопку **Заполнить выражения**. Автоматический механизм установления соответствия между данными документа и регистра не всегда работает правильно (в том случае, если не может однозначно определить соответствия, или тогда, когда соответствие, определенное им по его логике, отличается от желаемого), поэтому проверим правильность установленных соответствий. В итоге окно **Конструктора движения регистра** должно выглядеть так, как показано на Рисунок 12.4.

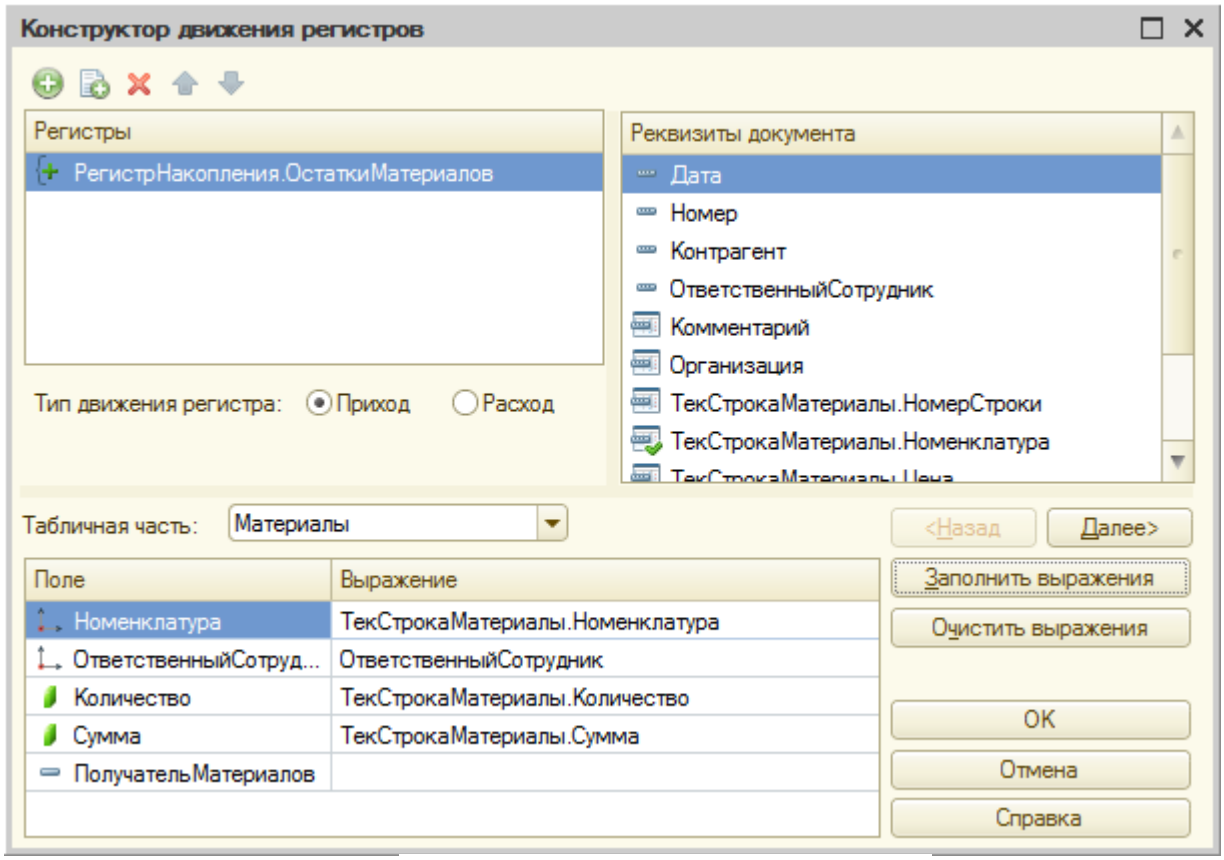


Рисунок 12.4- Конструктор движений

После нажатия на кнопку ОК, в модуле объекта документа будет сформирована такая процедура обработки проведения (так она выглядит после удаления комментариев о том, что код построен конструктором движений):

```
Процедура ОбработкаПроведения(Отказ, Режим)
 // регистр ОстаткиМатериалов Приход
 Движения.ОстаткиМатериалов.Записывать = Истина;
 Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
   Движение = Движения.ОстаткиМатериалов.Добавить();
   Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
   Движение.Период = Дата;
   Движение.Номенклатура = ТекСтрокаМатериалы.Номенклатура;
   Движение.ОтветственныйСотрудник = ОтветственныйСотрудник;
   Движение.Количество = ТекСтрокаМатериалы.Количество;
   Движение.Сумма = ТекСтрокаМатериалы.Сумма;
 КонецЦикла;
КонецПроцедуры
```

8. Запустим конфигурацию в режиме 1С:Предприятие, проверим работу механизма на практике. Для этого перепроведем существующие документы **ПоступлениеМатериалов**, можем ввести и новые документы этого вида. При проведении или перепроведении данные из документов попадают в регистр накопления **ОстаткиМатериалов**, Рисунок 12.5.

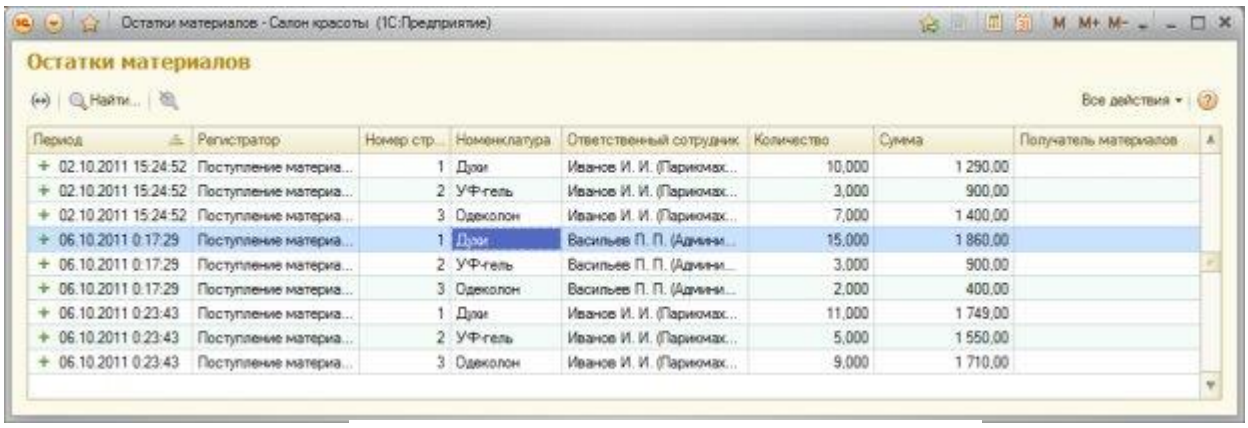


Рисунок 12.5. Данные в регистре накопления

Обратите внимание на то, что регистры накопления, как объекты, которые не предназначены изначально для "ручной" работы пользователя, не выводятся в командном интерфейсе даже при указании подсистем, в которые они входят. Нам, при разработке, понадобится просматривать регистры.

Для того, чтобы открыть окно регистра можно либо воспользоваться командой **Главное меню > Все функции** и в появившемся окне **Все функции** найти нужный регистр, либо открывать его с помощью команды в интерфейсе, предварительно самостоятельно добавив эту команду в нужный раздел интерфейса.

При записи данных о приходе материалов нам, в нашем случае, нет нужды в каких-либо дополнительных проверках вводимых данных, поэтому нас вполне устроит стандартная процедура проведения.

Практическая работа №13 Использование процедур и функций работы с ИБД

Цель: научиться программированию клиентских и серверных процедур

ХОД РАБОТЫ:

1. Добавим в дереве конфигурации новый отчет, назовем его **ОстаткиМатериалов**. Включим его в состав подсистемы**ОперативныйУчетМатериалов**.

2. На закладке **Основные** нажмем на кнопку с увеличительным стеклом в поле **Основная схема компоновки данных**. Появится окно конструктора макета, где мы можем задать имя (нас устроит имя по умолчанию – **ОсновнаяСхемаКомпоновкиДанных**), тип макета ограничен единственным – **Схема компоновки данных**. Нажмем в этом окне **Готово** и попадем в окно **конструктора СКД**. Здесь нам, в первую очередь, нужно добавить новый **источник данных**, в нашем случае это будет **Запрос**, Рисунок 13.1.

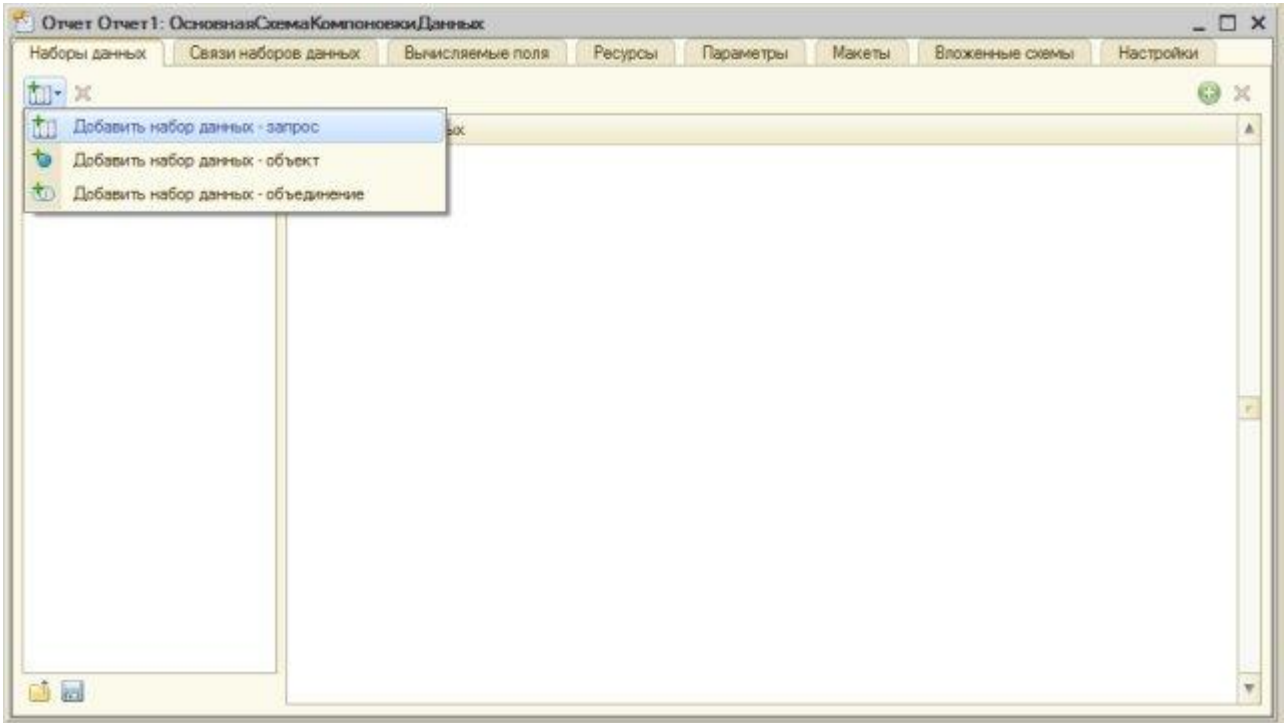


Рисунок 13.1- Добавление нового набора данных – запроса

3. Когда набор данных, названный **НаборДанных1**, будет добавлен, мы можем нажать на кнопку **КонструкторЗапроса**, находящуюся над полем **Запрос** в нижней части окна. Это приведет к открытию окна конструктора запроса.

4. Из виртуальной таблицы регистра накопления **ОстаткиМатериалов** выберем следующие поля, Рисунок 13.2.

- Номенклатура
- ОтвественныйСотрудник
- КоличествоОстаток
- СуммаОстаток

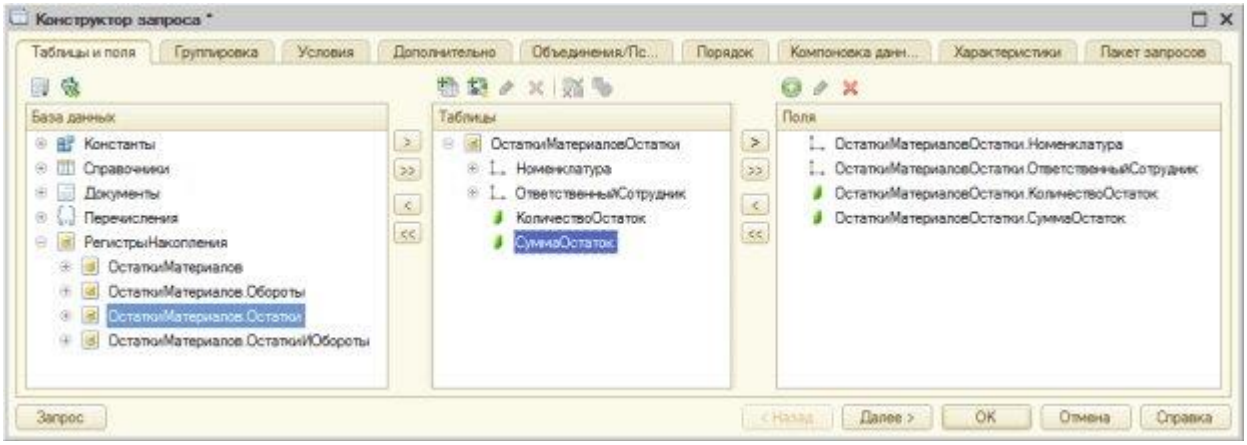


Рисунок 13.2- Создание запроса

5. На этом работа с конструктором запроса завершена – остальные настройки мы будем делать в конструкторе СКД. Благодаря установленному по умолчанию флагу **Автозаполнение**, на вкладке **Наборы данных** после создания запроса мы можем видеть заполненный список

полей, Рисунок 13.3. – с этими полями мы сможем работать при создании отчета.

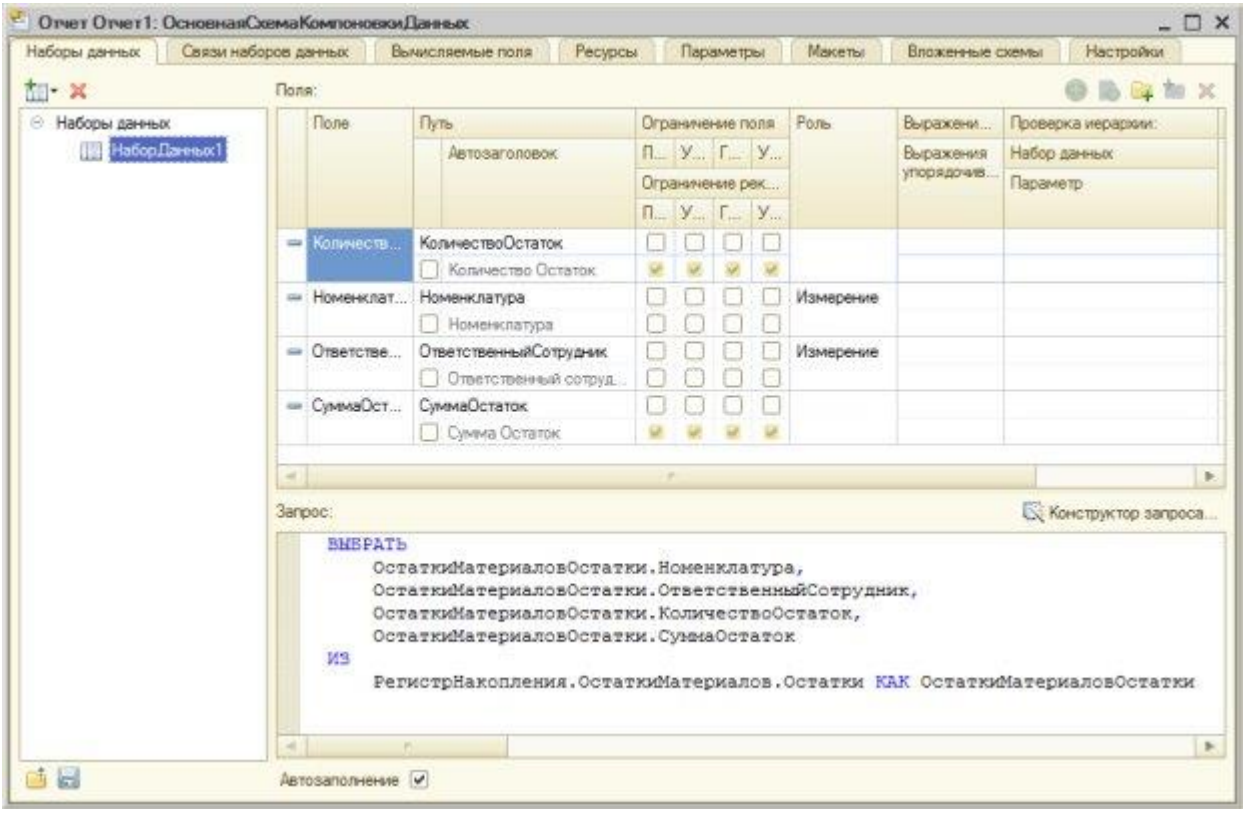


Рисунок 13.3- Автозаполнение списка полей на закладке Наборы данных

6. Переместимся в окне редактора СКД на вкладку **Ресурсы**, из списка **Доступные поля** перенесем в список, находящийся в правой части окна, поля, по которым можно вычислять итоги. В нашем случае это поля **КоличествоОстаток** и **СуммаОстаток**. По умолчанию этим полям в поле **Выражение** будет назначена агрегатная функция**Сумма**, нас устроит такое положение дел, Рисунок 13.4.

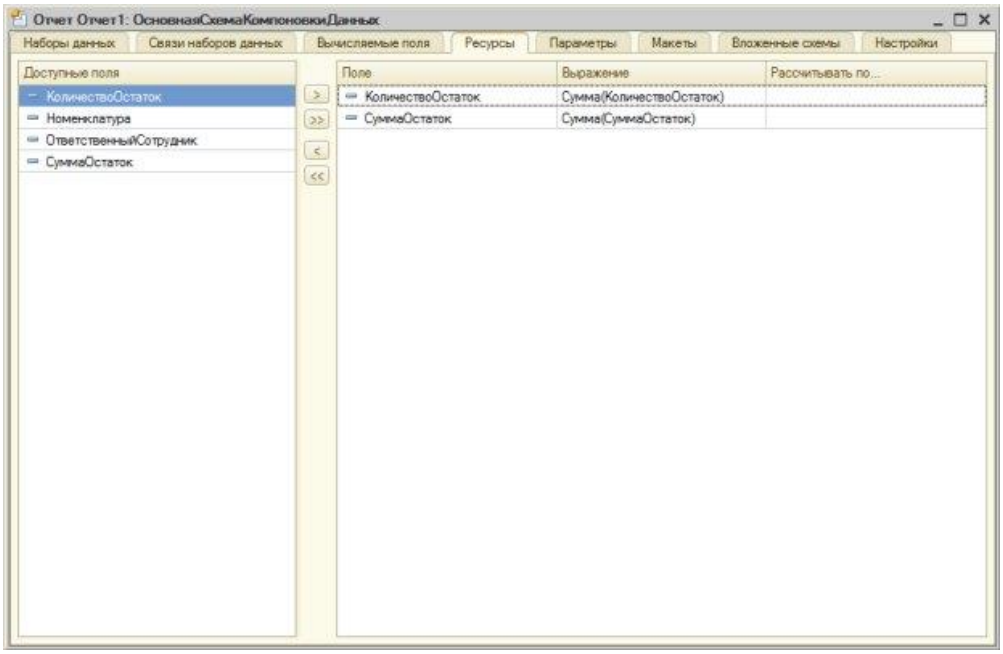


Рисунок 13.4. Настройка состава ресурсов отчета

7. Теперь займемся настройкой внешнего вида отчета.

Перейдем на закладку **Настройки**, вызовем кнопкой с соответствующим названием **Конструктор настроек** и выберем на его первой странице тип отчета – **таблицу**. Нажмем на кнопку **Далее** и в следующем окне выберем поля, которые будут отображаться в отчете в следующем порядке (Рисунок 13.5.):

- Номенклатура
- ОтветственыйСотрудник
- КоличествоОстаток
- СуммаОстаток

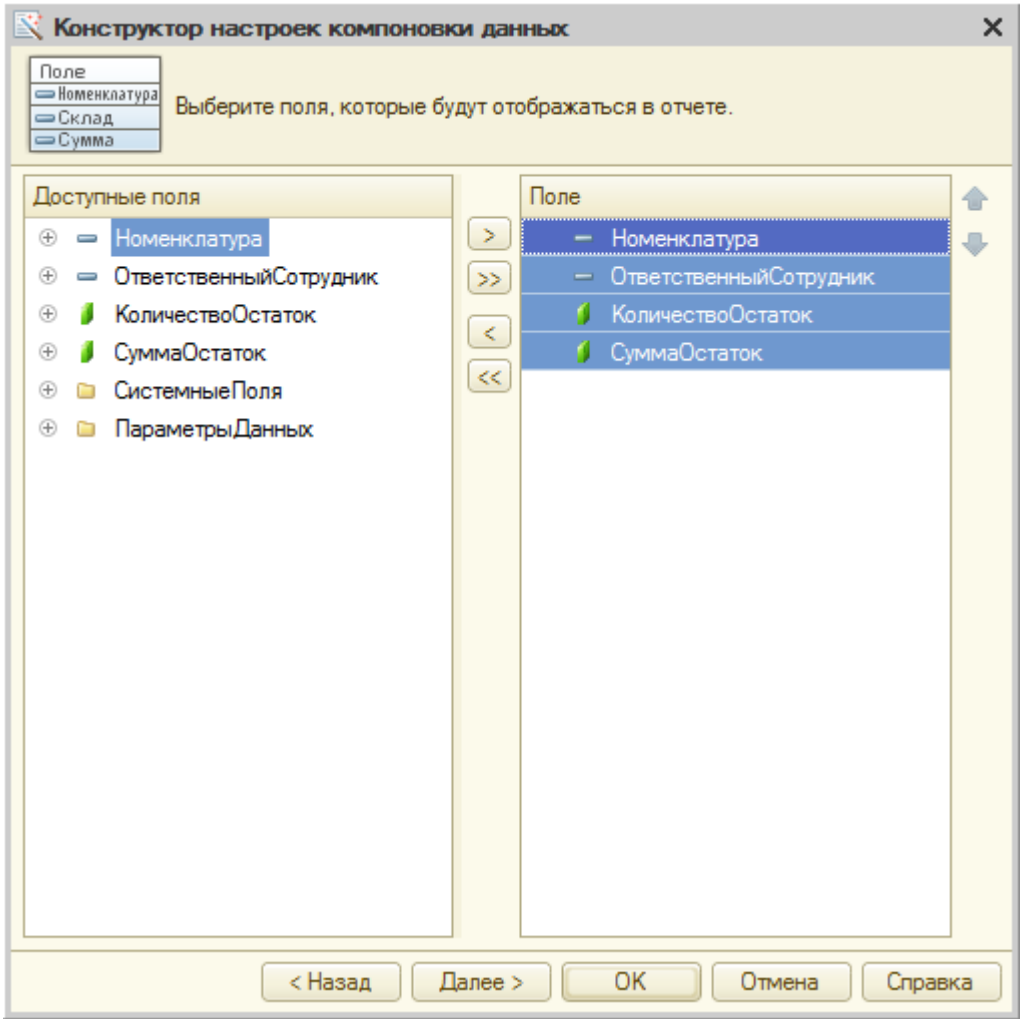


Рисунок 13.5- Выбор полей, которые будут отображаться в отчете

8. Нажмем кнопку **Далее**, в следующем окне конструктора, служащим для настройки группировки таблиц, в группу **Строки** добавим поле **Номенклатура**, в поле **Колонки** – **ОтветственныйСотрудник**. Тип группировки оставим в состоянии **Без иерархии**.

9. В следующем окне конструктора, который позволяет задать упорядочение отчета, зададим упорядочивание по полю **Номенклатура**, по возрастанию, Рисунок 13.6.

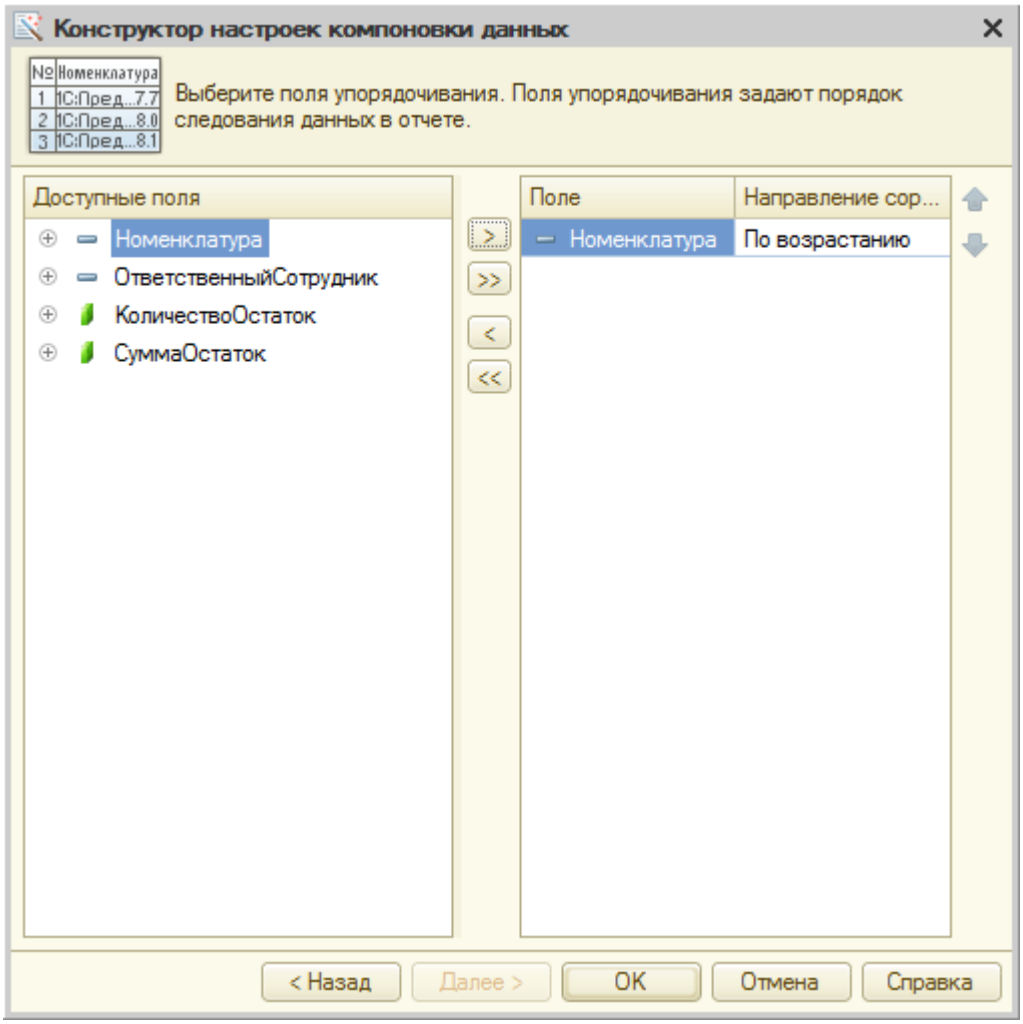


Рисунок 13.6- Настройка упорядочивания отчета

10. Нажмем **ОК**, в отчет будет добавлена новая таблица. В нижней части формы конструктора СКД, на закладке **Параметры**, выделим параметр **Период** и нажмем на кнопку**Свойства элемента пользовательских настроек**. В появившемся окне установим флаг **Включать в пользовательские настройки**, режим редактирования оставим в значении **Быстрый доступ**, Рисунок 13.7. Это позволит нам вывести данный параметр в форму отчета, позволит пользователю выбирать нужный период перед построением отчета

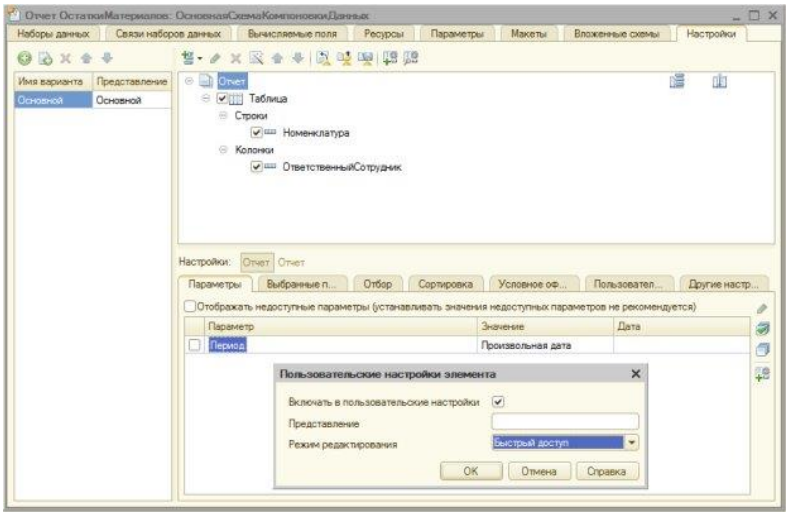
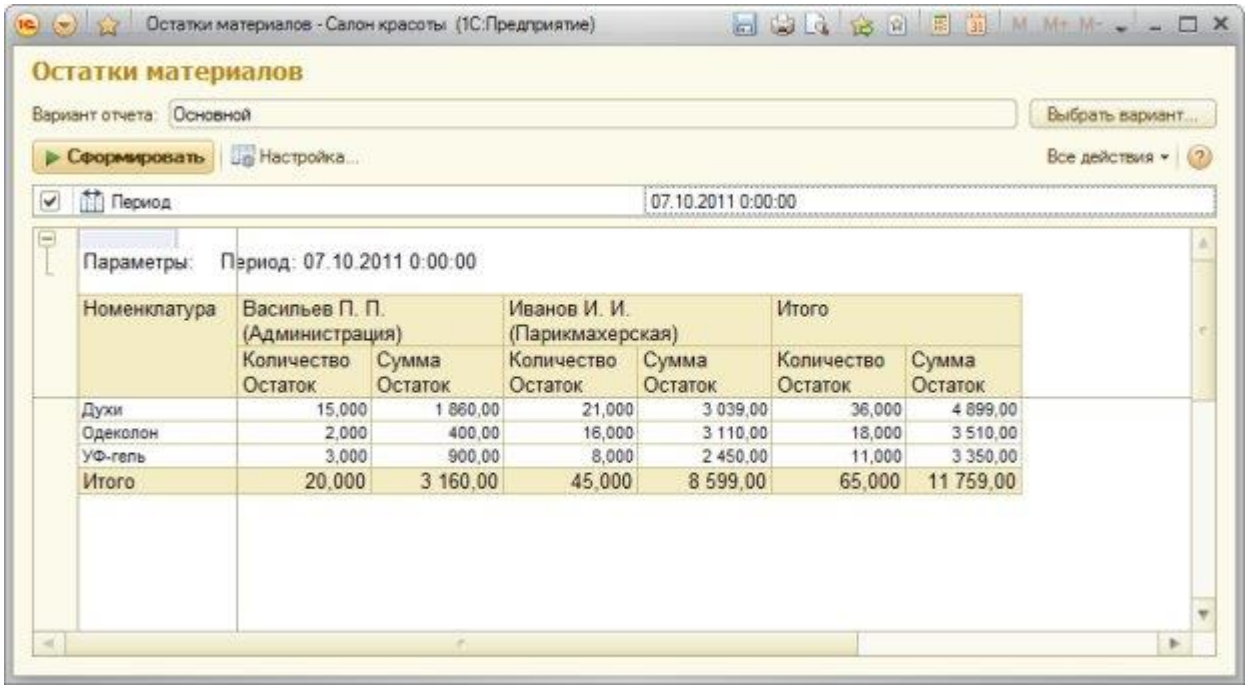


Рисунок 13.7- Настройка вывода параметра

11. Запустим систему в режиме 1С:Предприятие и построим отчет, Рисунок 13.8.



Номенклатура	Васильев П. П. (Администрация)		Иванов И. И. (Парикмахерская)		Итого	
	Количество Остаток	Сумма Остаток	Количество Остаток	Сумма Остаток	Количество Остаток	Сумма Остаток
Духи	15,000	1 860,00	21,000	3 039,00	36,000	4 899,00
Одеколон	2,000	400,00	16,000	3 110,00	18,000	3 510,00
УФ-гель	3,000	900,00	8,000	2 450,00	11,000	3 350,00
Итого	20,000	3 160,00	45,000	8 599,00	65,000	11 759,00

Рисунок 13.8- Готовый отчет по количественным и суммовым остаткам материалов

12. Перед построением отчета, если мы хотим задать параметр **Период**, установим флаг перед этим параметром и выберем нужную дату.

.

Практическая работа №14 Создание общих форм, серверных и клиентских процедур, процедуры запуска в начале работы системы

Цель: научиться конструированию процедуры проведения расходного документа, особое внимание уделено созданию сложных запросов с помощью консоли запросов..

ХОД РАБОТЫ:

1.Займемся проведением документа, отвечающего за списание материалов. Это – документ **ОтпускМатериаловМастеру**.

Наши две задачи:

Во-первых, мы хотим, чтобы система не позволяла списать больше материалов, чем числится за конкретным ответственным лицом. Это означает, что перед формированием движений мы должны сверить данные, введенные в табличную часть документа с данными по остаткам материалов, хранящимися в нашей базе, и, в том случае, если материалов нам не хватит – отказаться проводить документ и сообщить пользователю об ошибке.

Во-вторых, списывая материалы, мы должны придерживаться какой-либо политики оценки. Наиболее простая и широко используемая политика – это списание материалов по средней стоимости.

Определившись с нашими двумя основными задачами – реализации списания материалов по средней стоимости и контроля остатков, приступим к работе над процедурой для проведения нашего документа.

2. Перейдем в модуль объекта документа **ОтпускМатериаловМастеру**, с помощью панели инструментов **Модуль** создадим процедуру **ОбработкаПроведения**. Данные из табличной части мы будем получать с помощью запроса – в дальнейшем мы будем развивать этот запрос для получения необходимых сведений об остатках номенклатуры.

3. При создании запроса очень удобно пользоваться консолью запросов, которая позволяет в режиме 1С:Предприятие тут же проверять результаты, возвращаемые запросом. Подобные обработки можно найти на дисках ИТС, на различных Интернет-ресурсах.

4. Итак, обработку консоли запросов следует открыть командой **Главное меню > Файл > Открыть**. Начнем конструировать запрос, выбрав все поля из таблицы документа **ОтпускМатериаловМастеру**, Рисунок 14.1.

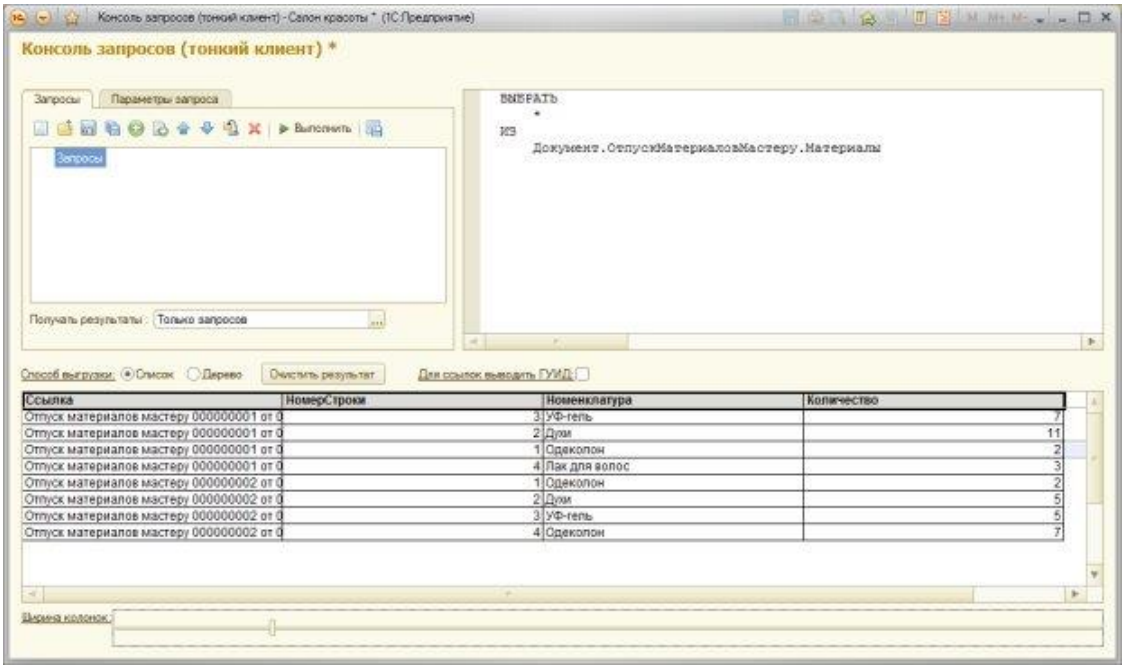


Рисунок 14.1- Конструирование запроса с помощью консоли запросов

5 В начале наш запрос имеет такой вид:

```
ВЫБРАТЬ
  *
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы
```

Мы выбрали все поля из таблицы, не вводя никаких ограничений. Мы собираемся получать данные из таблицы документа, проведением которого мы занимаемся. В запросе же мы получили данные по всем документам. Ограничим наш запрос по документу. Модифицируем запрос в консоли таким образом:

```
ВЫБРАТЬ
  *
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы
ГДЕ
  Ссылка=&Ссылка
```

6. Для того, чтобы задать параметр запроса в консоли запросов, перейдем на вкладку **Параметры запроса**, нажмем на кнопку **Заполнить**, после чего в поле **Значение параметра** выберем нужное его значение, в нашем случае – это будет один из документов **ОтпускМатериаловМастеру**, Рисунок 14.2.

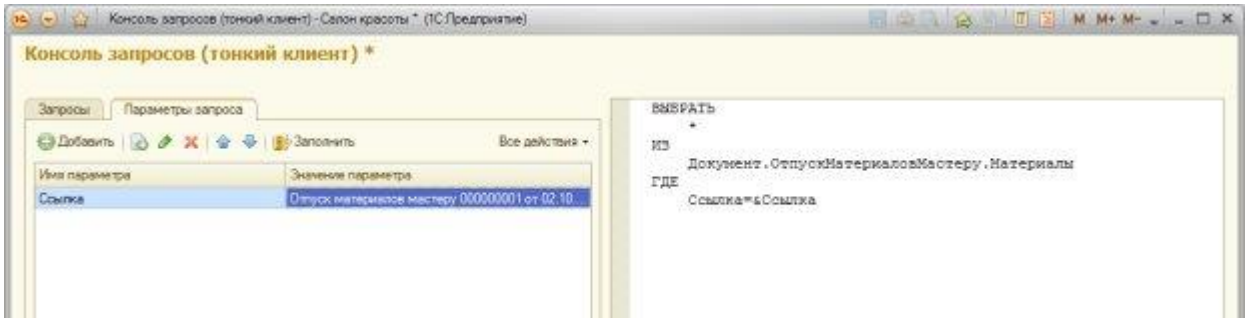


Рисунок 14.2- Настройка параметра запроса

7. Выполнив этот запрос, получим таблицу **Материалы** из указанного документа. Теперь подумаем над тем, какие именно данные нас интересуют. Нам нужны, во-первых, сведения о номенклатуре (поле **Номенклатура**), во-вторых – о количестве номенклатуры, которую мы хотим списать (поле **Количество**). Модифицируем запрос следующим образом:

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
  ДокМ.Количество
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
ГДЕ
  Ссылка=&Ссылка
```

Этот запрос даст нам такой результат:

Номенклатура	Количество
Одеколон	2
Духи	11
УФ-гель	7
Лак для волос	3
Одеколон	3

8. В документе мы намеренно смоделировали ситуацию, в которой пользователь, заполняя его, два раза ввел одну и ту же номенклатурную позицию. Сгруппируем теперь результаты запроса по полю **Номенклатура** – придем к такому тексту запроса:

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
  СУММА (ДокМ.Количество) КАК Количество
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
ГДЕ
  Ссылка=&Ссылка
СГРУППИРОВАТЬ ПО ДокМ.Номенклатура
```

Здесь мы применили функцию **СУММА** к количественному показателю и с помощью выражения **СГРУППИРОВАТЬ ПО** сгруппировали результаты по полю **Номенклатура**. Это привело к такому результату:

Номенклатура	Количество
Духи	11

Одеколон	5
УФ-гель	7
Лак для волос	3

Теперь все данные, которые нужны нам для проведения документа, мы получили.

9. Следующим этапом работы над запросом будет добавление в него команд для выбора нужных данных из регистра накопления **ОстаткиМатериалов**. Мы приходим к такому запросу:

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
  СУММА (ДокМ.Количество) КАК Количество,
  МАКСИМУМ (ОстМ.КоличествоОстаток) КАК КоличествоОстатков,
  МАКСИМУМ (ОстМ.СуммаОстаток) КАК СуммаОстатков
ИЗ
  Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
  ЛЕВОЕ СОЕДИНЕНИЕ
  РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени) Как ОстМ
  ПО
    ДокМ.Номенклатура = ОстМ.Номенклатура
ГДЕ
  Ссылка=&Ссылка
СГРУППИРОВАТЬ ПО ДокМ.Номенклатура
```

Здесь мы соединили таблицу регистра остатков с полученной таблицей документа по полю **Номенклатура**. К числовым полям, полученным из регистра, мы применили функцию **МАКСИМУМ** – иначе запрос будет выполняться неверно. В частности, если бы выше мы не выполнили группировку результатов запроса по полю **Номенклатура**, то в результатах запроса мы получили бы несколько полей с одной и той же номенклатурой, к каждому из которых было бы присоединено одно и то же поле из таблицы регистра.

Вышеописанный запрос привел к такому результату:

Номенклатура	Количество	КоличествоОстатков	СуммаОстатков
Духи	11	36	4 899
Одеколон	5	18	3 510
УФ-гель	7	11	3 350
Лак для волос	3	NULL	NULL

Здесь нас не устраивают два момента. Во-первых, в полях **КоличествоОстатков** и **СуммаОстатков** показаны данные по всем ответственным лицам – а нам нужно знать данные лишь по тому ответственному, с которого мы материалы списываем. Во-вторых, по номенклатурной позиции, по которой данных в регистре **ОстаткиМатериалов** не имеется, в полях находится значение **NULL**. Для того, чтобы попытка работать с этим значением не привела в будущем к возникновению ошибок, обработаем поля, полученные из регистра, функцией **ЕСТЬNULL.**

10. Модифицируем запрос в соответствии с последними соображениями.

```
ВЫБРАТЬ
  ДокМ.Номенклатура,
```

```
    СУММА (ДокМ.Количество) КАК Количество,
    МАКСИМУМ (ЕСТЬNULL(ОстМ.КоличествоОстаток, 0)) КАК КоличествоОстатков,
    МАКСИМУМ (ЕСТЬNULL(ОстМ.СуммаОстаток, 0)) КАК СуммаОстатков
ИЗ
    Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
    ЛЕВОЕ СОЕДИНЕНИЕ
    РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени ,
ОтветственныйСотрудник = &ОтвСотр) Как ОстМ
    ПО
      ДокМ.Номенклатура = ОстМ.Номенклатура
ГДЕ
    Ссылка=&Ссылка
СГРУППИРОВАТЬ ПО ДокМ.Номенклатура
```

Здесь мы добавили отбор из регистра только записей, относящихся к заданному ответственному сотруднику (**ОтветственныйСотрудник = &ОтвСотр**) и применили к показателям количества и суммы, полученным из регистра, функцию **ЕСТЬNULL**. Если в поле находится **NULL**, мы заменяем это значение нулем.

Результат запроса теперь выглядит так:

Номенклатура	Количество	КоличествоОстатков	СуммаОстатков
Духи	11	15	1 860
Одеколон	5	2	400
УФ-гель	7	3	900
Лак для волос	3	0	0

11. Скопируем полученный текст запроса в буфер обмена и перейдем в Конфигуратор. В процедуре ОбработкаПроведения документаОтпускМатериаловМастеру. Процедура пока пуста, щелкнем в ней правой кнопкой мыши и вызовем из контекстного меню командуКонструктор запроса с обработкой результата. В ответ на вопрос конструктора о создании нового запроса, ответим утвердительно, после чего, в окне конструктора нажмем на кнопку Запрос и вставим в пустое поле для текста запроса полученный текст запроса (предварительно нажав на кнопку Редактировать запрос в окне Запрос), Рисунок 14.3.

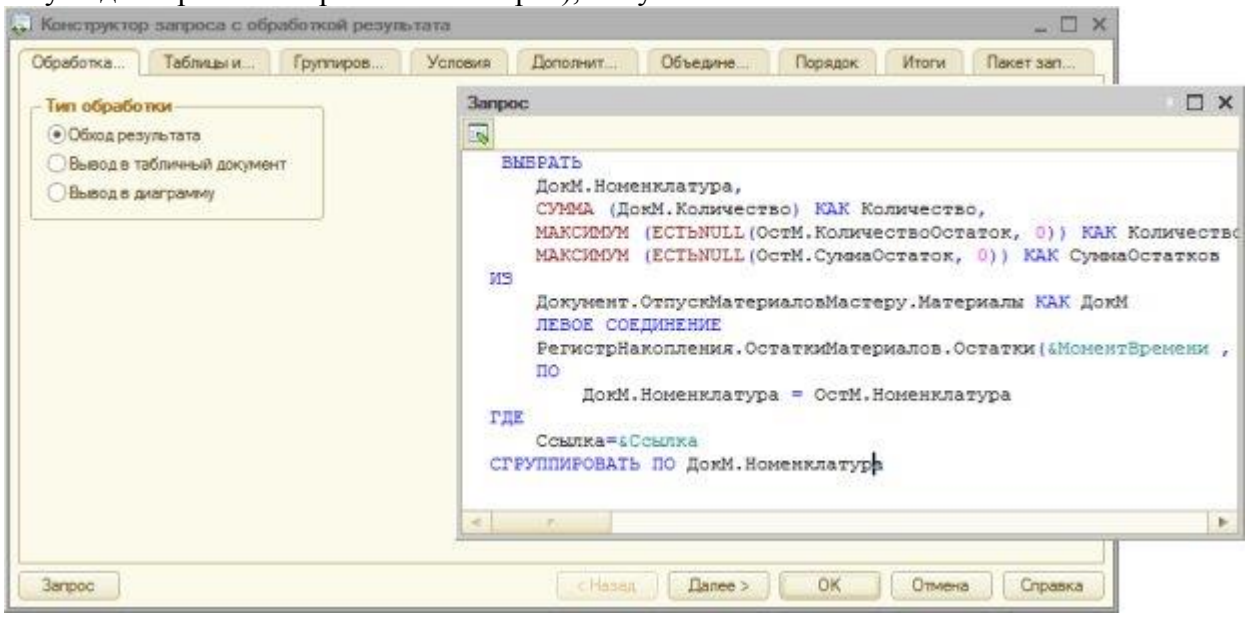


Рисунок 14.3. Добавление сформированного текста запроса в конструктор

12. На закладке Обработка окна Конструктор запроса оставим переключатель Тип обработки в положении Обход результата. После закрытия окна Запрос конструктор автоматически разберет запрос, "разложит" по закладкам своего окна, при необходимости, его можно будет редактировать, пользуясь инструментами, расположенными на этих закладках. Нас запрос устраивает – поэтому мы можем нажимать в окне конструктора ОК и переходить к дальнейшему редактированию кода, Рисунок 14.4.

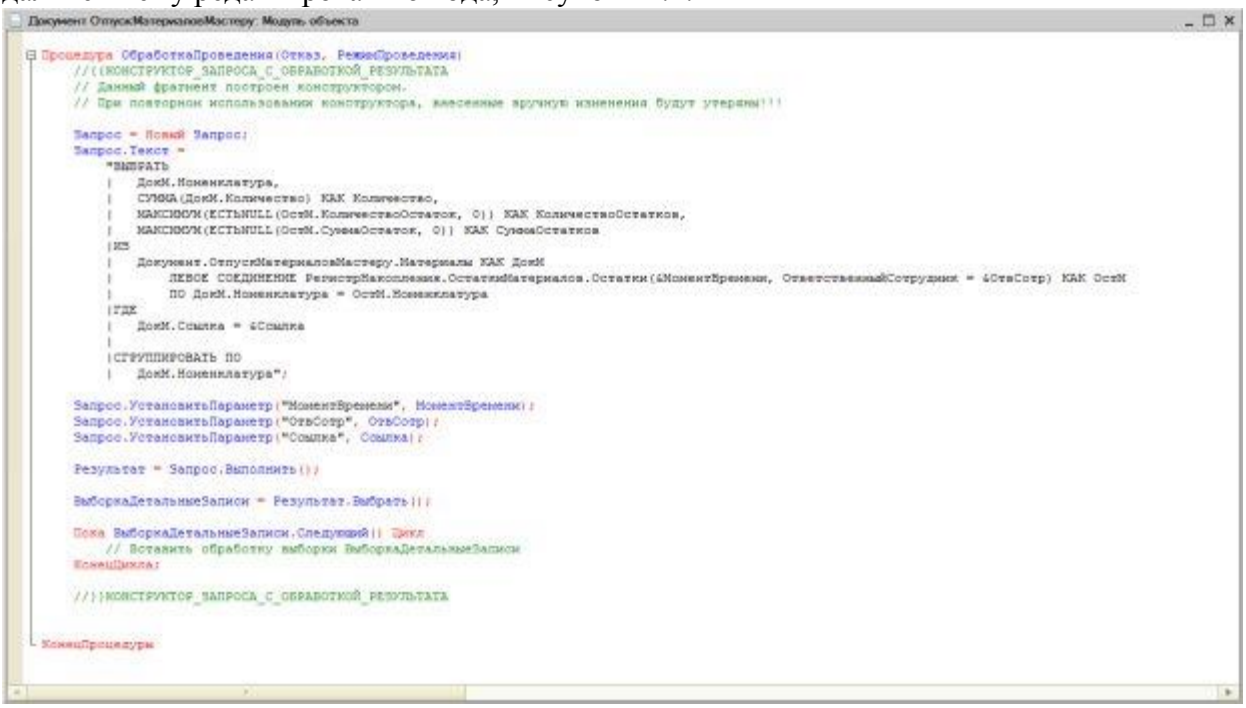


Рисунок 14.4. Добавление сформированного текста запроса в конструктор

13. Здесь нас, в первую очередь, не устраивает автоматическое заполнение параметров запроса

Заменим код:

```
Запрос.УстановитьПараметр("МоментВремени", МоментВремени);
Запрос.УстановитьПараметр("ОтвСотр", ОтвСотр);
```

На код:

```
Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
Запрос.УстановитьПараметр("ОтвСотр", ОтветственныйСотрудник);
```

Здесь мы, во-первых, вызвали метод МоментВремени(), возвращающий момент времени для нашего документа (то есть – для того, в модуле объекта которого мы сейчас работаем). Во-вторых, мы обратились к реквизиту документа ОтветственныйСотрудник для установки параметра ОтвСотр.

14. Проверим работу созданного механизма, запустив систему в режиме отладки и установив в коде модуля точку останова после получения выборки из результатов запроса.

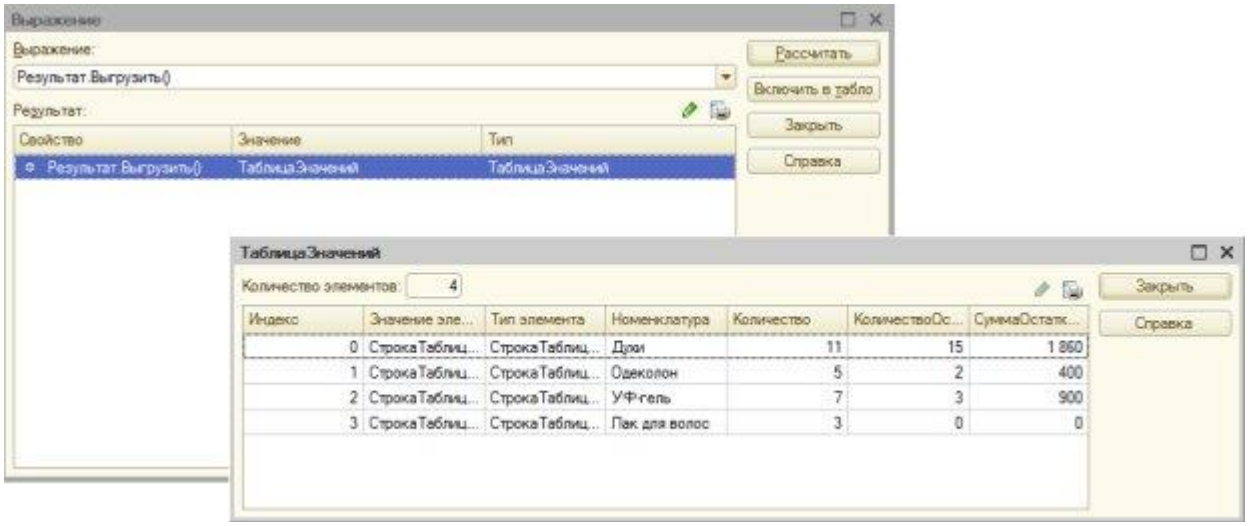


Рисунок 14.5. Анализ результата выполнения запроса в коде процедуры проведения документа

15. Здесь мы выполнили метод Выгрузить() для результата выполнения запроса (переменная Результат), получили таблицу значений, которую можно проанализировать. Результат нас устраивает, поэтому мы принимаемся за дальнейшую работу над процедурой. В итоге у нас получился следующий код:

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    | ДокМ.Номенклатура,
    | СУММА(ДокМ.Количество) КАК Количество,
    | МАКСИМУМ(ЕСТЬNULL(ОстМ.КоличествоОстаток, 0)) КАК КоличествоОстатков,
    | МАКСИМУМ(ЕСТЬNULL(ОстМ.СуммаОстаток, 0)) КАК СуммаОстатков
    |ИЗ
    | Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
    |                                                ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени, ОтветственныйСотрудник
= &ОтвСотр) КАК ОстМ
    |   ПО ДокМ.Номенклатура = ОстМ.Номенклатура
    |ГДЕ
    | ДокМ.Ссылка = &Ссылка
    |
    |СГРУППИРОВАТЬ ПО
    | ДокМ.Номенклатура";

  Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
  Запрос.УстановитьПараметр("ОтвСотр", ОтветственныйСотрудник);
  Запрос.УстановитьПараметр("Ссылка", Ссылка);

  Результат = Запрос.Выполнить();

  ВыборкаДЗ = Результат.Выбрать();

  Движения.ОстаткиМатериалов.Записывать=Истина;

  Пока ВыборкаДЗ.Следующий() Цикл
```

```
            Если ВыборкаДЗ.Количество>ВыборкаДЗ.КоличествоОстатков Тогда
              Сообщить("Недостаточное количество товара "+ВыборкаДЗ.Номенклатура
              +", необходимо "+ВыборкаДЗ.Количество+", в наличии "
              +ВыборкаДЗ.КоличествоОстатков);
              Отказ=Истина;
              Движения.ОстаткиМатериалов.Записывать=Ложь;
            КонецЕсли;

            Если Отказ Тогда
              Продолжить;
            КонецЕсли;

            Движение=Движения.ОстаткиМатериалов.Добавить();
            Движение.ВидДвижения=ВидДвиженияНакопления.Расход;
            Движение.Период=Дата;
            Движение.Номенклатура=ВыборкаДЗ.Номенклатура;
            Движение.Количество=ВыборкаДЗ.Количество;

Движение.Сумма=ВыборкаДЗ.Количество*ВыборкаДЗ.СуммаОстатков/ВыборкаДЗ.Количество
Остатков;
            Движение.ОтветственныйСотрудник=ОтветственныйСотрудник;
            Движение.ПолучательМатериалов=ПолучательМатериалов;
          КонецЦикла;

        КонецПроцедуры
```

16. Проверим результаты работы нашего кода в режиме 1С:Предприятие. Если документ верно реагирует на попытку списания материалов, количество которых превышает имеющееся, и если анализ состава регистра накопления после проведения показывает, что списано нужное количество материалов и их стоимость определена верно – можно считать, что мы справились с поставленной задачей.

17. Мы собираемся построить отчет, который выводил бы сведения о начальном и конечном остатке материалов за определенный временной интервал, а так же – сведения о приходе и расходе материалов за этот период.

Создадим новый отчет, назовем его Материалы, включим в подсистему ОперативныйУчетМатериалов, добавим основную схему компоновки данных, создадим новый набор данных – Запрос. В конструкторе запроса выберем из виртуальной таблицы регистра накопления ОстаткиМатериалов следующие поля, Рисунок 14.6.:

- Номенклатура
- ОтветственныйСотрудник
- КоличествоНачальныйОстаток
- КоличестоПриход
- КоличествоРасход
- КоличествоКонечныйОстаток

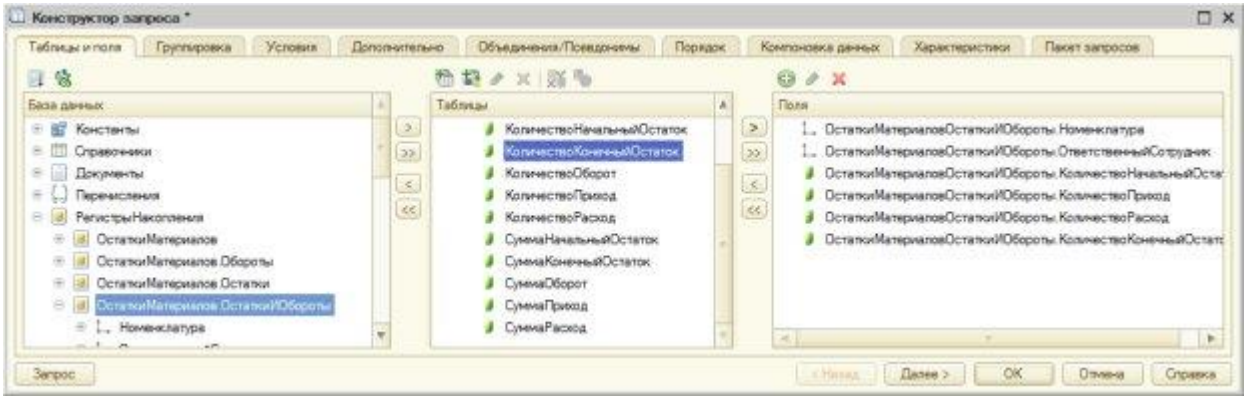


Рисунок 14.6. Настройка запроса для отчета

18. Нажмем ОК в окне конструктора запроса, перейдем на закладку Ресурсы окна редактора СКД, добавим все количественные поля в состав ресурсов, Рисунок 14.7.

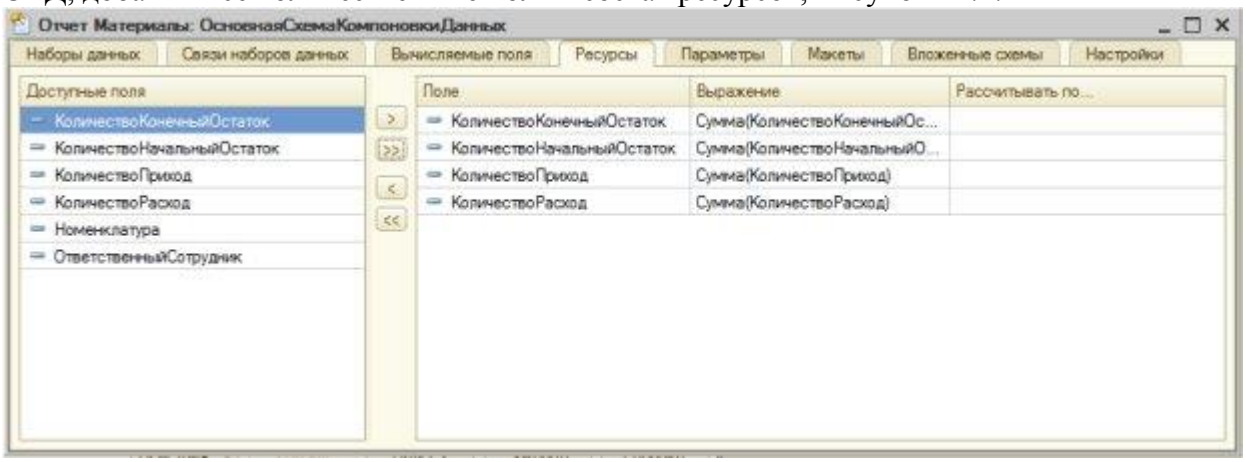


Рисунок 14.7. Настройка состава ресурсов

19.На закладке Настройки воспользуемся конструктором настроек. Выберем табличный тип отчета, нажмем Далее, в окне настройки состава и порядка следования полей, которые будут отображаться в отчете, расположим поля следующим образом:

- Номенклатура
- ОтветственныйСотрудник
- КоличествоНачальныйОстаток
- КоличестоПриход
- КоличествоРасход
- КоличествоКонечныйОстаток

20. На следующем этапе укажем, что группировка строк будет осуществляться по полю Номенклатура, колонок – по полю ОтветственныйСотрудник.
21. На этапе настройки упорядочения укажем упорядочение по возрастанию по полю Номенклатура.
На этом настройка таблицы завершена.
22. На верхнем уровне настроек отчета укажем, что параметры Начало периода и Конец периода следует включать в пользовательские настройки.
23. Отчет готов, нам осталось лишь проверить его работу в режиме 1С:Предприятие, Рисунок 14.8.

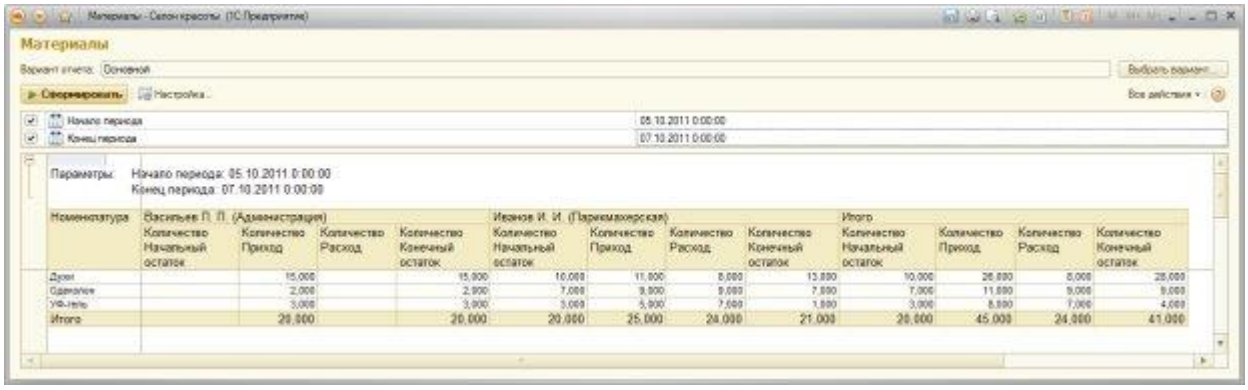


Рисунок 14.8. Готовый отчет

24. В нашей конфигурации есть пара документов, относящихся к одной сфере деятельности – к учету материалов. Выше мы упоминали об объекте Журнал документов. Познакомимся с этим объектом поближе.

25. Добавим в конфигурацию новый журнал документов, назовем его ДокументыУчетаМатериалов. Включим журнал в подсистему ОперативныйУчетМатериалов.

На вкладке Данные добавим в состав документов, регистрируемых в журнале, документы ПоступлениеМатериалов и ОтпускМатериаловМастеру. Добавим в журнал графу с именем ОтветственныйСотрудник, заполним свойство Ссылки для этой графы, указав реквизиты ОтветственныйСотрудник из включенных в журнал документов, Рисунок 14.9.

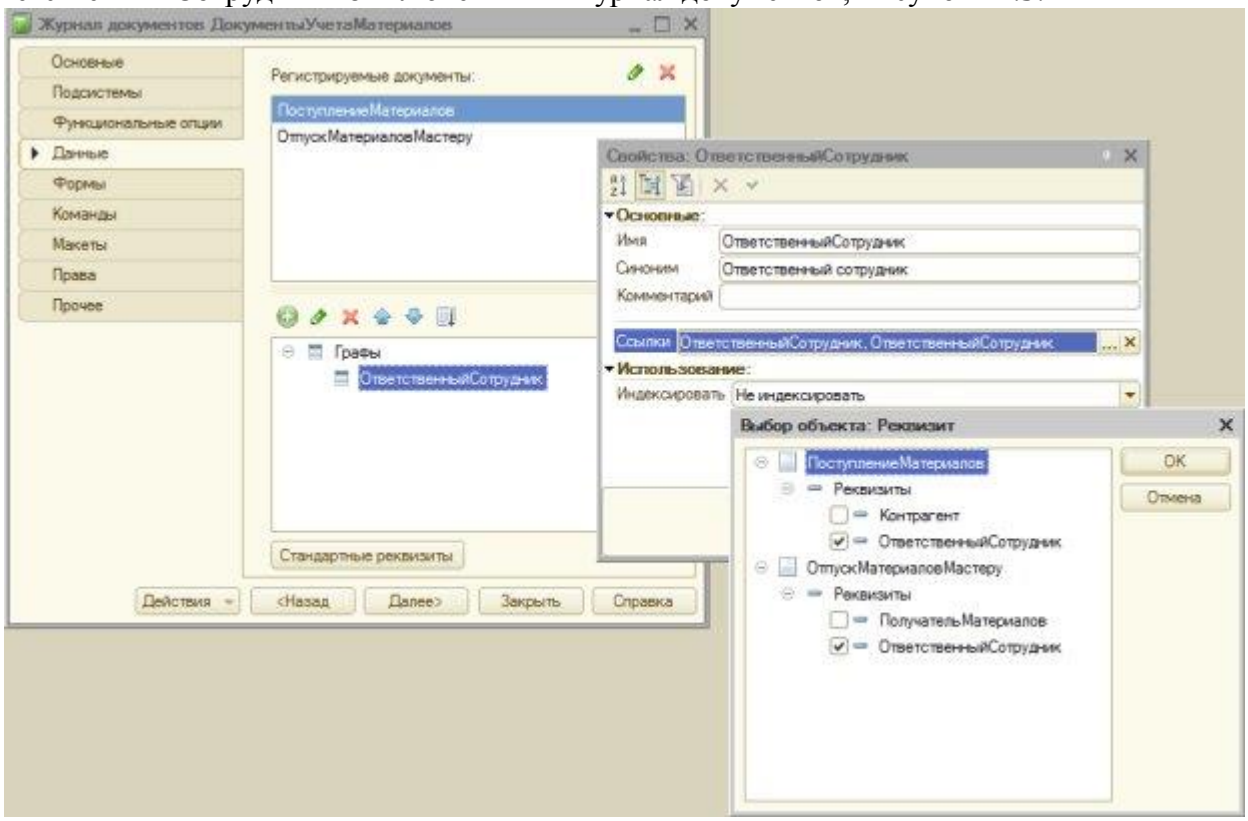


Рисунок 14.9. Настройка журнала документов

26. В режиме 1С:Предприятие наш журнал позволит просматривать список документов разных типов, включенных в него, Рисунок 14.10.

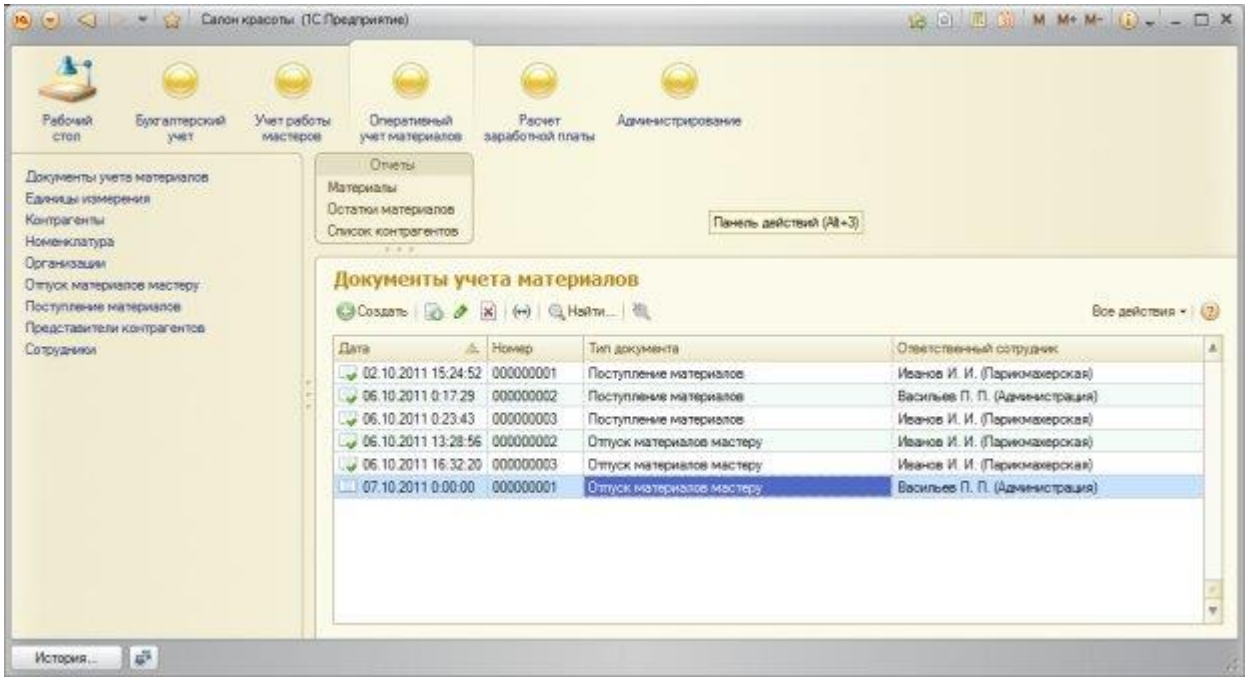


Рисунок 14.10. Журнал документов в режиме 1С:Предприятие

27.Создадим новую обработку, назовем ее РаботаСДокументами. Включим в подсистему Администрирование.

28. Добавим в обработку команду с именем ВывестиСписокВидовДокументов, зададим обработчик для этой команды, выведем ее на форму обработки.

29. Сейчас мы воспользуемся свойством глобального контекста Метаданные для того, чтобы вывести пользователю список синонимов существующих в конфигурации документов. Для подобных действий нам понадобится серверная процедура, которую мы вызовем из клиентской процедуры обработчика ранее созданной команды. Выполнить запланированное можно с помощью следующего кода:

```
&НаКлиенте
Процедура ВывестиСписокВидовДокументов(Команда)
  ВывестиСинонимыДокументов();
КонецПроцедуры

Процедура ВывестиСинонимыДокументов()
  Для каждого Документ из Метаданные.Документы Цикл
    Сообщить(Документ.Синоним);
  КонецЦикла;
КонецПроцедуры
```

Результат выполнения показан на Рисунок 14.11.

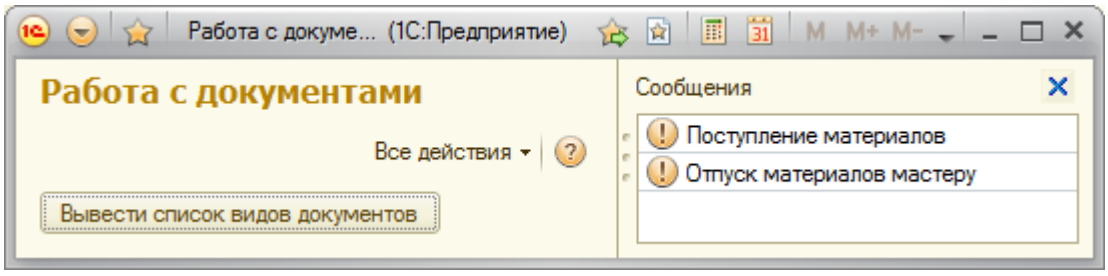


Рисунок 14.11. Вывод списка синонимов документов

30. Добавим в нашу обработку новую команду – СоздатьДокументПоступлениеМатериалов. Так же добавим новый реквизит – ПроводитьДокумент, поместим его на форму, Рисунок 14.12. Мы зададим все данные, в том числе – и тип документа для создания – в коде.

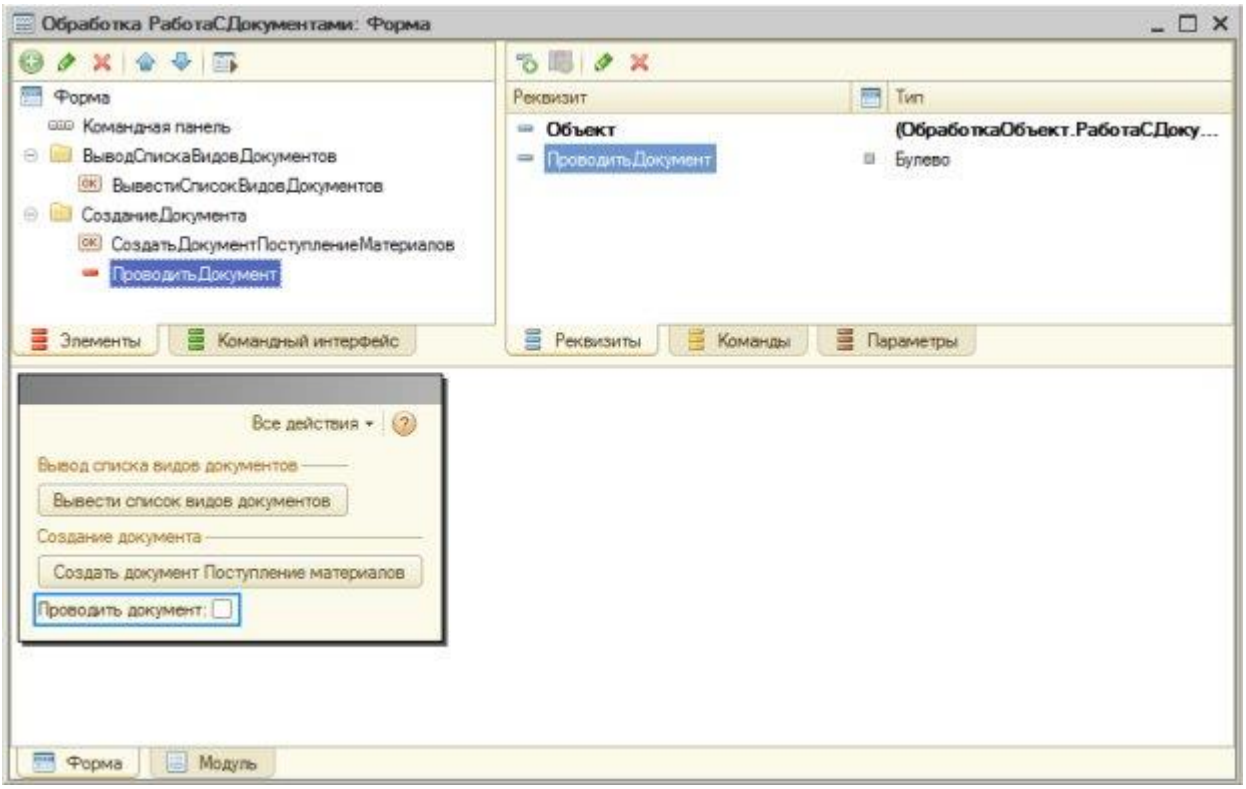


Рисунок 14.12. Модификация формы обработки

Решить поставленную задачу можно с помощью следующего кода:

```
&НаКлиенте
Процедура СоздатьДокументПоступлениеМатериалов(Команда)
  //Настраиваем режим записи нового документа
  РежимЗаписи=РежимЗаписиДокумента.Запись;
  Если ПроводитьДокумент Тогда
    РежимЗаписи=РежимЗаписиДокумента.Проведение;
  КонецЕсли;
  //В функции будет создан новый документ
  //Она возвратит ссылку на него
  Документ=СоздатьДокумент(РежимЗаписи);
  //Открываем форму документа
  ОткрытьЗначение(Документ);
КонецПроцедуры

Функция СоздатьДокумент(РежимЗаписи)
  //Создаем новый документ
  Документ = Документы.ПоступлениеМатериалов.СоздатьДокумент();
  //Заполняем его реквизиты
  Документ.Дата=ТекущаяДата();

Документ.ОтветственныйСотрудник=Справочники.Сотрудники.НайтиПоКоду("000000003");

Документ.Контрагент=Справочники.Контрагенты.НайтиПоРеквизиту("КонтактныеСведения",
```

"ул. Береговая, д. 2, телефон 3-34-34");
Документ.Комментарий="Документ создан автоматически";
//Заполняем строку табличной части
НоваяСтрокаТЧ=Документ.Материалы.Добавить();

НоваяСтрокаТЧ.Номенклатура=Справочники.Номенклатура.НайтиПоНаименованию("Духи");
НоваяСтрокаТЧ.Количество=10;
НоваяСтрокаТЧ.Цена=200;
НоваяСтрокаТЧ.Сумма=10*200;
//Записываем документ
Документ.Записать(РежимЗаписи);
//Возвращаем ссылку на документ
Возврат(Документ.Ссылка);
КонецФункции

Вот, как выглядит документ, созданный программно с помощью нашего кода, Рисунок 14.13.

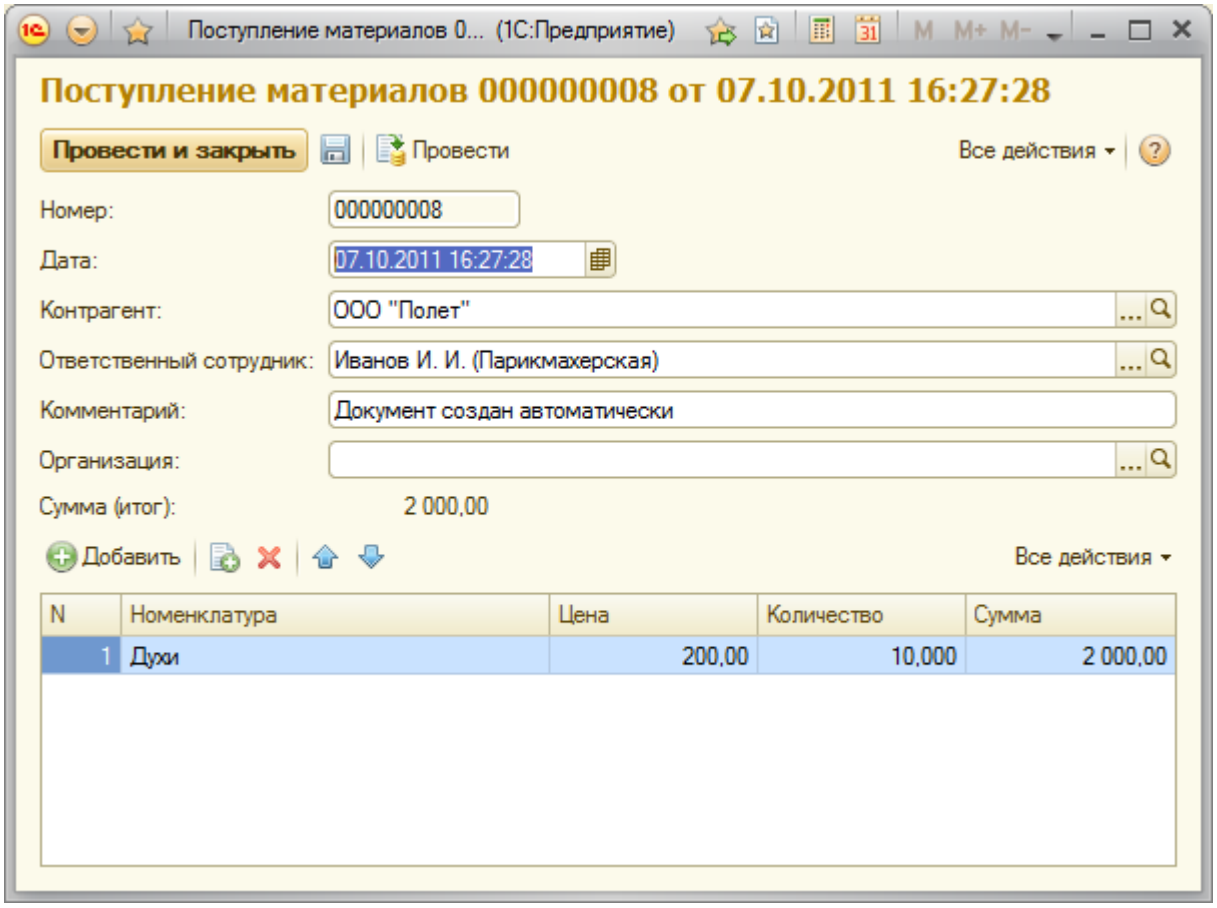


Рисунок 14.13. Документ, созданный автоматически

31. Решим теперь следующую задачу. Нужно пометить на удаление все документы типа ПоступлениемМатериалов, которые созданы автоматически – их реквизитКомментарий содержит текст "Документ создан автоматически".
32. Добавим в форму обработки новую команду, назовем ее ПометитьНаУдаление. Поставленную задачу можно реализовать с помощью следующего кода:

&НаКлиенте
Процедура ПометитьНаУдаление(Команда)
ПометитьДокументыНаУдаление();
Предупреждение("Были помечены на удаление документы поступления материалов");

```
КонецПроцедуры

Процедура ПометитьДокументыНаУдаление()
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    | ПоступлениеМатериалов.Ссылка
    |ИЗ
    | Документ.ПоступлениеМатериалов КАК ПоступлениеМатериалов
    |ГДЕ
    | ПоступлениеМатериалов.Комментарий = &Комментарий";

  Запрос.УстановитьПараметр("Комментарий", "Документ создан автоматически");

  Результат = Запрос.Выполнить();

  ВыборкаДетальныеЗаписи = Результат.Выбрать();

  Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
    Документ=ВыборкаДетальныеЗаписи.Ссылка.ПолучитьОбъект();
    Документ.УстановитьПометкуУдаления(Истина);
  КонецЦикла;
КонецПроцедуры
```

Здесь мы, в серверной процедуре ПометитьДокументыНаУдаление(), получаем с помощью запроса список ссылок на документы, реквизит Комментарий которых равен нужному нам значению. После этого в цикле обхода выборки запроса переходим от ссылки на объект к объекту (тип ДокументОбъект) и устанавливаем у объектов пометки удаления.

При завершении серверной процедуры, мы, на клиенте, показываем пользователю окно сообщения, Рисунок 14.14.

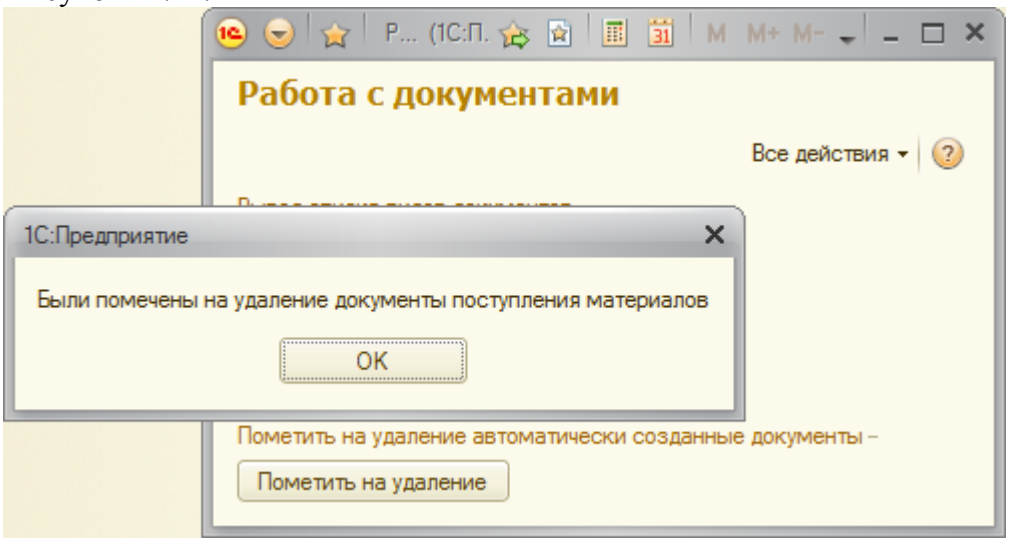


Рисунок 14.14. Сообщение пользователю о пометке документов на удаление

33. Нашей следующей задачей будет вывод пользователю списка документов за заданный пользователем период. Добавим в форму обработки команду ВыводСпискаДокументовЗаПериод и два реквизита – ДатаНачала и ДатаОкончаният – тип Дата, состав даты – Дата и время. Дата документа содержит сведения о дате и времени создания документа, поэтому для выбора периода, в который должны попасть искомые документы, нам понадобятся значения даты с датой и временем.

Решение задачи может выглядеть так:

```
&НаКлиенте
Процедура ВыводСпискаДокументовЗаПериод(Команда)
  Сообщить("Обнаружены следующие документы за период с "+ДатаНачала+" по "+ДатаОкончания);
  ВыводСписка();
КонецПроцедуры

Процедура ВыводСписка()
  Выборка=Документы.ПоступлениеМатериалов.Выбрать(ДатаНачала, ДатаОкончания);
  Пока Выборка.Следующий() Цикл
    Сообщить(Выборка.Ссылка);
  КонецЦикла
КонецПроцедуры
```

Здесь мы пользуемся методом Выбрать с параметрами, устанавливающими дату начала и дату окончания для выборки документов. Полученную выборку перебираем в цикле и сообщаем пользователю о найденных документах, Рисунок 14.15.

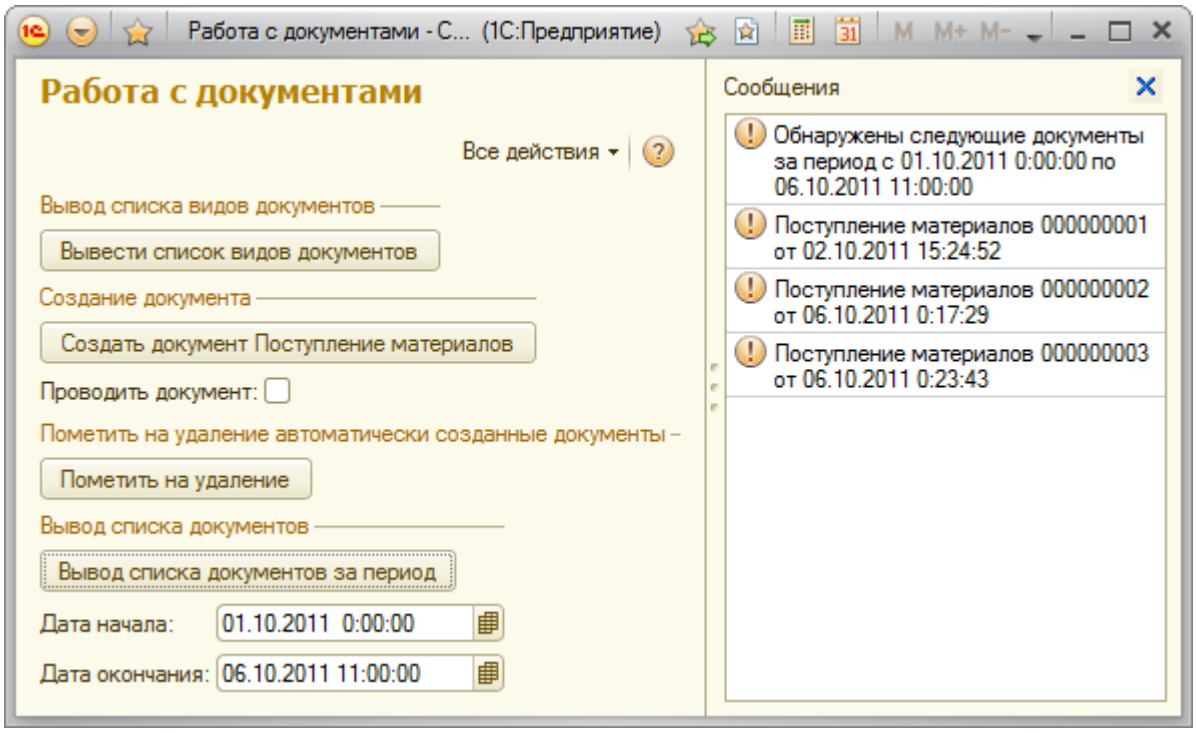


Рисунок 14.15. Вывод списка документов, принадлежащих периоду, заданному пользователем

Практическая работа №15 Создание и программирование формы справочника

Цель: изучить особенности работы с оборотными регистрами накопления, кроме того, научиться использованию агрегатов, последовательностей, нумераторов и регистров сведений.

ХОД РАБОТЫ:

1. Создадим документ **РеализацияМатериалов**. Добавим его в состав подсистемы **ОперативныйУчетМатериалов**. Запретим **оперативное проведение** документа.
2. В состав данных документа включим следующие реквизиты:

   Имя: Покупатель: Тип: СправочникСсылка.Контрагенты

   Имя: ОтветственныйСотрудник: Тип: СправочникСсылка.Сотрудники

   Добавим в документ табличную часть **Материалы** со следующими реквизитами:

   Имя: Номенклатура, Тип: СправочникСсылка.Номенклатура

   Имя: Остаток, Тип: Число, Длина 10, точность 3

   Имя: Себестоимость, Тип: Число, длина 10, точность 2

   Имя: Количество, Тип: Число, длина 10, точность 3

   Имя: ЦенаПродажи, Тип: Число, длина 10, точность 2

   Имя: Выручка, тип: Число, длина 10, точность 2

3. Теперь займемся формой документа. Нам хотелось бы реализовать следующую функциональность – при заполненной данными о номенклатуре табличной части, по нажатию на кнопку **РассчитатьСебестоимостьИОстатки**, заполнять сведения о себестоимости материалов и об их остатках по выбранному ответственному сотруднику, исходя из данных, хранящихся в регистре накопления **ОстаткиМатериалов**.
4. Создадим форму документа, добавим команду формы **РассчитатьСебестоимостьИОстатки** и переместим ее на командную панель табличного поля, Рисунок 17.1.

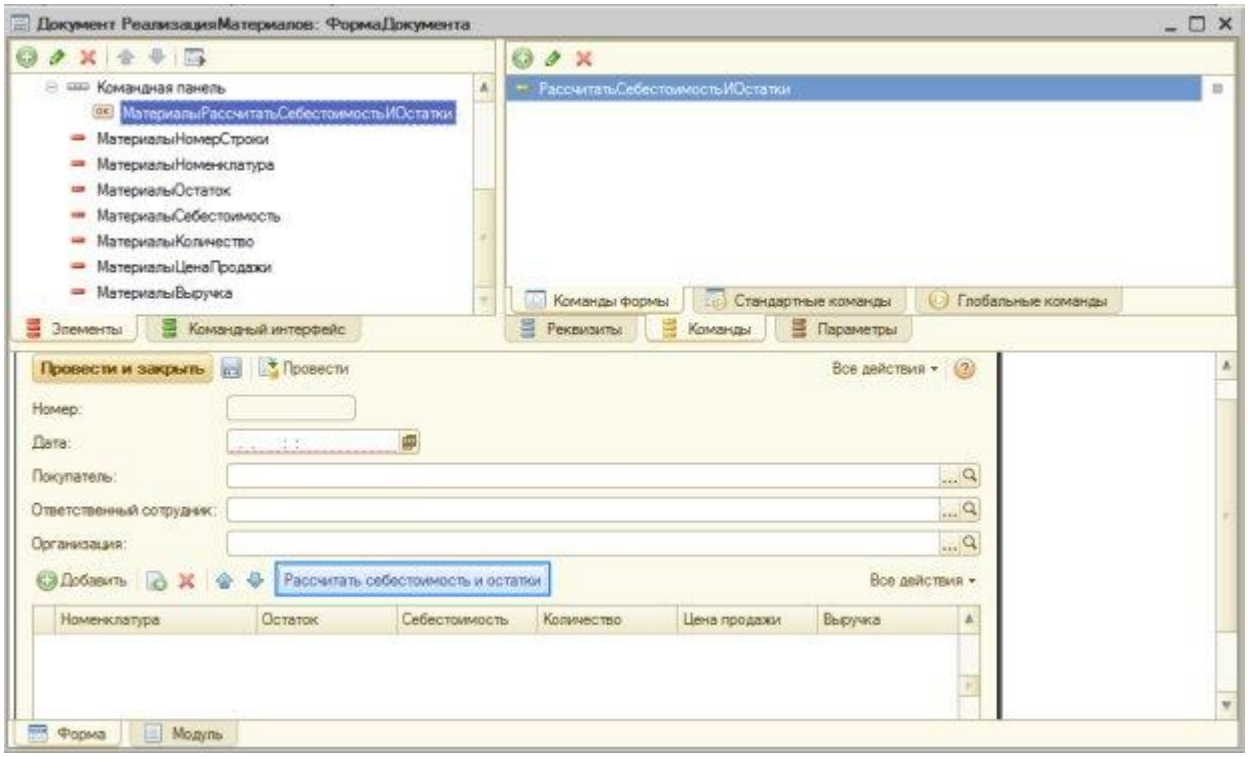


Рисунок 17.1- Конструирование формы документа РеализацияМатериалов

5. В полях **Остаток** и **Себестоимость** пользователь должен увидеть текущие сведения об остатках и себестоимости выбранной номенклатуры, которая числится за выбранным в реквизите документа ответственным лицом. Эти поля должны быть недоступны для редактирования, так как играют лишь вспомогательную, информационную функцию. Для этого у полей **МатериалыОстаток** и **МатериалыСебестоимость** установим свойство **ТолькоПросмотр**, Рисунок 17.2

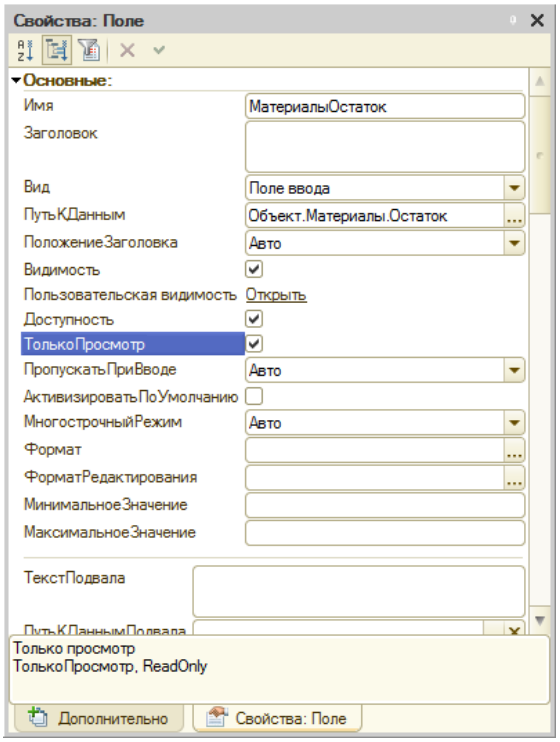


Рисунок 17.2- Запрет редактирования поля пользователем

6. Теперь займемся кодом, отвечающим за получение данных по остаткам и себестоимости материалов, которые находятся в табличной части. Действия по получению данных и по заполнению ими табличного поля мы должны выполнять на сервере, для этого создадим серверную процедуру **РассчитатьНаСервере()** и вызовем ее из процедуры обработчика **РассчитатьСебестоимостьИОстатки()**.

7. Для того, чтобы получить нужные данные, нам понадобится запрос, который выбирает данные из табличной части и из регистра накопления **ОстаткиМатериалов**, сгруппировав их по номенклатуре. Воспользуемся уже знакомой вам консолью запросов для того, чтобы построить подобный запрос. Здесь следует вспомнить, что, конструируя процедуру проведения документа **ОтпускМатериаловМастеру**, мы строили подобный запрос. Поэтому сейчас мы можем просто этот запрос доработать. Дорабатывать его, однако, удобно с помощью консоли запросов. Перенесем текст запроса из кода метода **ОбработкаПроведения** документа **ОтпускМатериаловМастеру**, добавим в поле текста запроса консоли запросов в режиме 1С:Предприятие и займемся редактированием текста.

Мы начинаем с такого текста запроса:

```
ВЫБРАТЬ
    ДокМ.Номенклатура,
    СУММА(ДокМ.Количество) КАК Количество,
    МАКСИМУМ(ЕСТЬNULL(ОстМ.КоличествоОстаток, 0)) КАК КоличествоОстатков,
    МАКСИМУМ(ЕСТЬNULL(ОстМ.СуммаОстаток, 0)) КАК СуммаОстатков
ИЗ
    Документ.ОтпускМатериаловМастеру.Материалы КАК ДокМ
        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени, ОтветственныйСотрудник = &ОтвСотр) КАК ОстМ
        ПО ДокМ.Номенклатура = ОстМ.Номенклатура
ГДЕ
    ДокМ.Ссылка = &Ссылка

СГРУППИРОВАТЬ ПО
    ДокМ.Номенклатура
```

8. В этом запросе нам понадобится откорректировать обращение к таблице документа, который содержит нужные данные. Предполагая, что пользователь может ввести одну и ту же номенклатурную позицию несколькими строками, возможно, собираясь особым образом задать цену продажи для одной и той же позиции, мы не будем группировать запрос по номенклатуре. Мы получим из таблицы все данные, которые мог ввести пользователь – позже мы используем эти данные для заполнения табличного поля.

9. Остальные данные нас устраивают. У нас получился такой запрос, Рисунок 17.3.

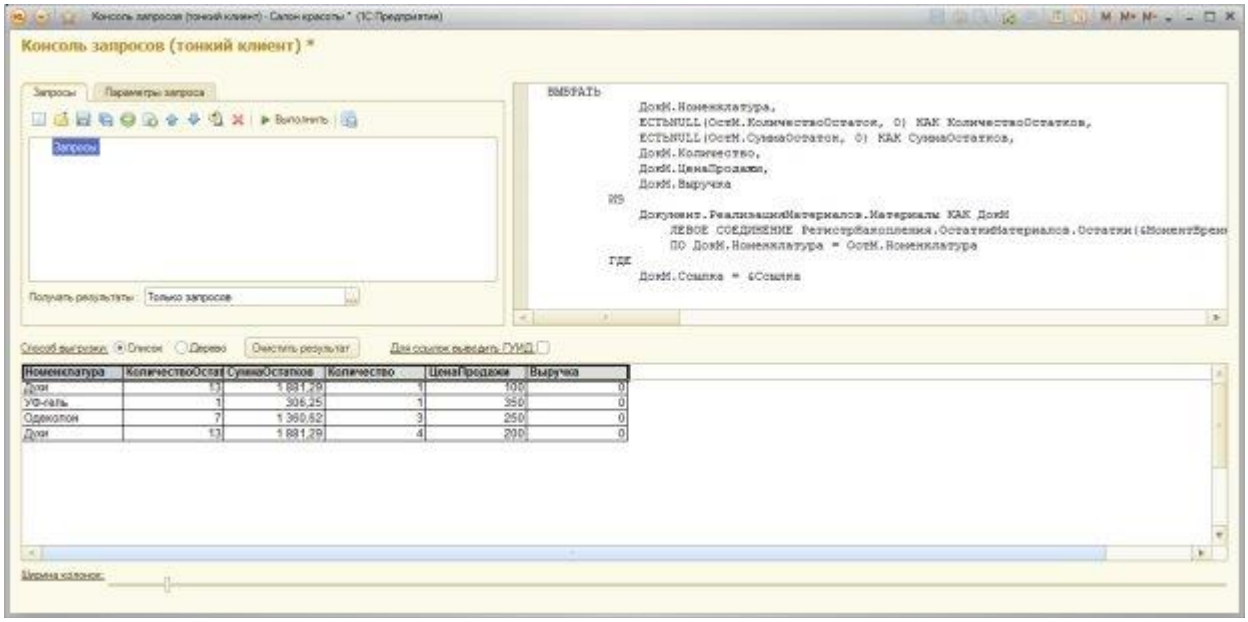


Рисунок 17.3. Запрос для получения данных для заполнения табличного поля

10. Займемся теперь кодом модуля **РассчитатьНаСервере()**. Воспользуемся конструктором запросов с обработкой результатов для того, чтобы перенести сформированный в консоли текст запроса в код модуля.

В итоге задачу заполнения табличной части данными мы решили следующим образом:

```
&НаКлиенте
Процедура РассчитатьСебестоимостьИОстатки(Команда)
  Режим=РежимДиалогаВопрос.ДаНет;
  Ответ=Вопрос("Для продолжения нужно записать документ. Сделать это?",Режим,0);
  Если Ответ=КодВозвратаДиалога.Да Тогда
    Записать();
    РассчитатьНаСервере();
    Предупреждение("Табличная часть заполнена");
  Иначе
    Предупреждение("Табличная часть не заполнена");
  КонецЕсли;
КонецПроцедуры

&НаСервере
Процедура РассчитатьНаСервере()
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    | ДокМ.Номенклатура,
    | ЕСТЬNULL(ОстМ.КоличествоОстаток, 0) КАК КоличествоОстатков,
    | ЕСТЬNULL(ОстМ.СуммаОстаток, 0) КАК СуммаОстатков,
    | ДокМ.Количество,
    | ДокМ.ЦенаПродажи,
    | ДокМ.Выручка
      |ИЗ
    | Документ.РеализацияМатериалов.Материалы КАК ДокМ
```

```
        |                                                    ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени, ОтветственныйСотрудник
= &ОтвСотр) КАК ОстМ
        |   ПО ДокМ.Номенклатура = ОстМ.Номенклатура
        |ГДЕ
        | ДокМ.Ссылка = &Ссылка";

      Запрос.УстановитьПараметр("МоментВремени", Объект.Ссылка.МоментВремени());
      Запрос.УстановитьПараметр("ОтвСотр", Объект.ОтветственныйСотрудник);
      Запрос.УстановитьПараметр("Ссылка", Объект.Ссылка);

      Результат = Запрос.Выполнить();

      ВыборкаДетальныеЗаписи = Результат.Выбрать();

      Объект.Материалы.Очистить();

      Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
        СтрокаТЧ=Объект.Материалы.Добавить();
        СтрокаТЧ.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
        Если НЕ  ВыборкаДетальныеЗаписи.КоличествоОстатков=0 Тогда
          СтрокаТЧ.Себестоимость=ВыборкаДетальныеЗаписи.СуммаОстатков/
          ВыборкаДетальныеЗаписи.КоличествоОстатков;
        КонецЕсли;
        СтрокаТЧ.Остаток=ВыборкаДетальныеЗаписи.КоличествоОстатков;
        СтрокаТЧ.Количество=ВыборкаДетальныеЗаписи.Количество;
        СтрокаТЧ.ЦенаПродажи=ВыборкаДетальныеЗаписи.ЦенаПродажи;
        СтрокаТЧ.Выручка=ВыборкаДетальныеЗаписи.Выручка;
      КонецЦикла;
    КонецПроцедуры

    &НаКлиенте
    Процедура РассчитатьСумму()
      ТекущаяСтрока=Элементы.Материалы.ТекущиеДанные;
      ТекущаяСтрока.Выручка=ТекущаяСтрока.Количество*ТекущаяСтрока.ЦенаПродажи;
    КонецПроцедуры

    &НаКлиенте
    Процедура МатериалыКоличествоПриИзменении(Элемент)
      РассчитатьСумму();
    КонецПроцедуры

    &НаКлиенте
    Процедура МатериалыЦенаПродажиПриИзменении(Элемент)
      РассчитатьСумму();
    КонецПроцедуры
```

В методе, работающем на клиенте, мы, сразу после запуска, сообщаем пользователю о том, что для правильной работы системы нужно сначала записать документ. Если он отвечает утвердительно – записываем документ и вызываем серверную процедуру. Для задавания подобных вопросов в виде диалоговых окон с кнопками-вариантами вопроса (Рисунок 17.4.), используется метод **Вопрос()**.

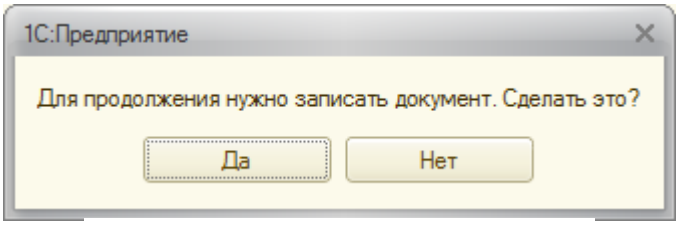


Рисунок 17.4- Вопрос пользователю

В серверной процедуре мы выбираем с помощью запроса данные из табличной части нашего документа и из регистра **ОстаткиМатериалов**. Когда данные получены, мы просто очищаем табличную часть и заполняем ее снова, теперь уже с использованием новых данных.

В итоге, после того, как пользователь заполнил табличную часть документа и нажал на кнопку **Рассчитать себестоимость и остатки**, он получит примерно следующее,Рисунок 17.5.

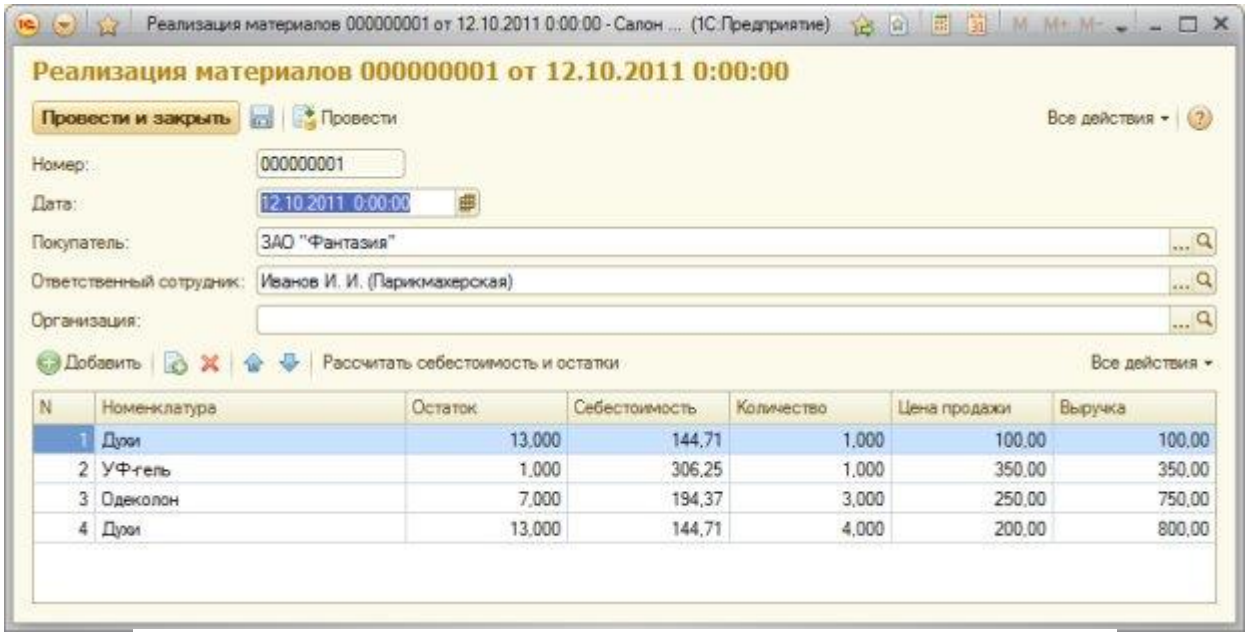


Рисунок 17.5- Результат автоматического заполнения табличной части

Поля **Остаток** и **Себестоимость** в данном случае играют лишь вспомогательную роль, позволяя пользователю сразу, при заполнении документа, понять, каково состояние дел с остатками материалов.

11. Создадим новый регистр накопления, назовем его **Продажи**. Вид регистра установим в значение **Обороты**, Рисунок 17.6. Включим его в состав подсистемы **ОперативныйУчетМатериалов**.

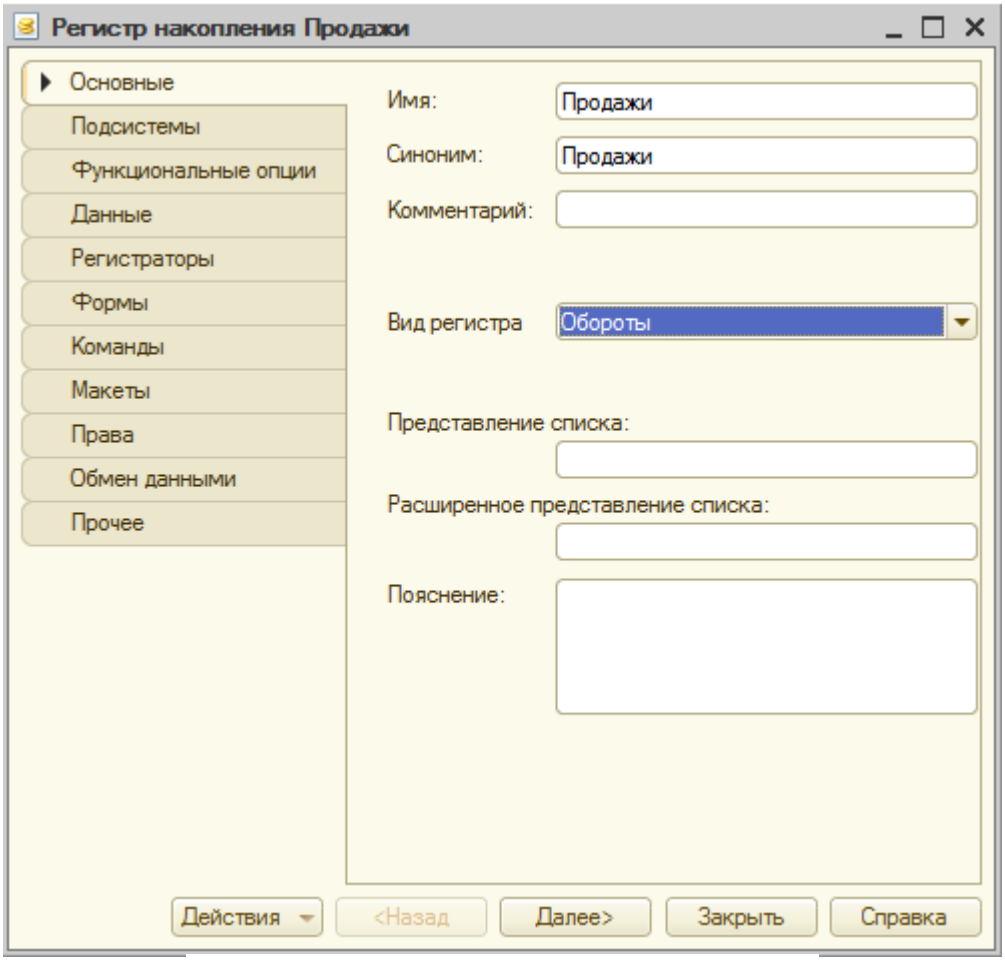


Рисунок 17.6. Создание регистра накопления

12. В состав данных регистра, Рисунок 17.7., внесем следующие:

Измерения:

Имя: Контрагент, тип: СправочникСсылка.Контрагенты

Имя: ОтветственныйСотрудник, тип: СправочникСсылка.Сотрудники

Имя: Номенклатура, тип: СправочникСсылка.Номенклатура

Ресурсы:

Имя: Себестоимость, тип: Число, длина 10, точность 2

Имя: Количество, тип: Число, длина 10, точность 3

Имя: Выручка, тип: Число, длина 10, точность 2

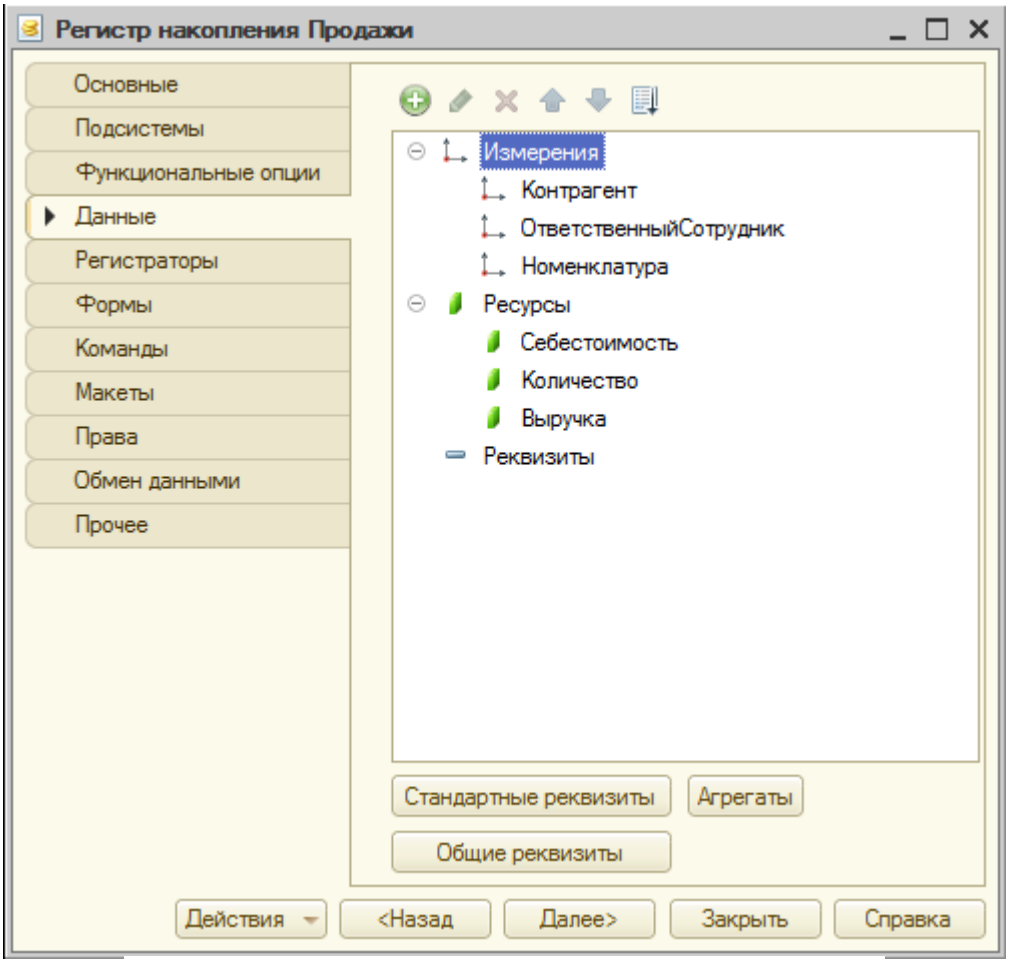


Рисунок 17.7- Настройка данных регистра накопления

13. В качестве регистратора для данного регистра выберем документ **РеализацияМатериалов**.

14. Теперь займемся проведением этого документа. Он должен формировать движения по двум регистрам – по регистру **ОстаткиМатериалов**, и по регистру **Продажи**.

15. Добавим в модуль объекта документа процедуру **ОбработкаПроведения**. При конструировании этой процедуры мы можем воспользоваться уже отработанными при проведении документа **ОтпускМатериаловМастеру** механизмами.

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

Запрос = Новый Запрос;
Запрос.Текст =
 "ВЫБРАТЬ
 | ДокМ.Номенклатура,
 | СУММА(ДокМ.Количество) КАК Количество,
 | СУММА(ДокМ.Выручка) КАК Выручка,
 | МАКСИМУМ(ЕСТЬNULL(ОстМ.КоличествоОстаток, 0)) КАК КоличествоОстатков,
 | МАКСИМУМ(ЕСТЬNULL(ОстМ.СуммаОстаток, 0)) КАК СуммаОстатков
 |ИЗ
 | Документ.РеализацияМатериалов.Материалы КАК ДокМ
```

```
            |                                                    ЛЕВОЕ                СОЕДИНЕНИЕ
РегистрНакопления.ОстаткиМатериалов.Остатки(&МоментВремени, ОтветственныйСотрудник
= &ОтвСотр) КАК ОстМ
            |    ПО ДокМ.Номенклатура = ОстМ.Номенклатура
            |ГДЕ
            | ДокМ.Ссылка = &Ссылка
            |
            |СГРУППИРОВАТЬ ПО
            | ДокМ.Номенклатура";

          Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
          Запрос.УстановитьПараметр("ОтвСотр", ОтветственныйСотрудник);
          Запрос.УстановитьПараметр("Ссылка", Ссылка);

          Результат = Запрос.Выполнить();
          ВыборкаДЗ = Результат.Выбрать();
          Движения.ОстаткиМатериалов.Записывать=Истина;
          Движения.Продажи.Записывать=Истина;

          Пока ВыборкаДЗ.Следующий() Цикл
            Если ВыборкаДЗ.Количество>ВыборкаДЗ.КоличествоОстатков Тогда
              Сообщить("Недостаточное количество товара "+ВыборкаДЗ.Номенклатура
              +", необходимо "+ВыборкаДЗ.Количество+", в наличии "
              +ВыборкаДЗ.КоличествоОстатков);
              Отказ=Истина;
              Движения.ОстаткиМатериалов.Записывать=Ложь;
              Движения.Продажи.Записывать=Ложь;
            КонецЕсли;

            Если Отказ Тогда
              Продолжить;
            КонецЕсли;

            Движение=Движения.ОстаткиМатериалов.Добавить();
            Движение.ВидДвижения=ВидДвиженияНакопления.Расход;
            Движение.Период=Дата;
            Движение.Номенклатура=ВыборкаДЗ.Номенклатура;
            Движение.Количество=ВыборкаДЗ.Количество;

Движение.Сумма=ВыборкаДЗ.Количество*ВыборкаДЗ.СуммаОстатков/ВыборкаДЗ.Количество
Остатков;
            Движение.ОтветственныйСотрудник=ОтветственныйСотрудник;

            Движение=Движения.Продажи.Добавить();
            Движение.Период=Дата;
            Движение.Номенклатура=ВыборкаДЗ.Номенклатура;
            Движение.Количество=ВыборкаДЗ.Количество;

Движение.Себестоимость=ВыборкаДЗ.Количество*ВыборкаДЗ.СуммаОстатков/ВыборкаДЗ.Ко
личествоОстатков;
            Движение.Выручка=ВыборкаДЗ.Выручка;
```

```
    Движение.ОтветственныйСотрудник=ОтветственныйСотрудник;
    Движение.Контрагент=Покупатель;
  КонецЦикла;
КонецПроцедуры
```

Так, здесь мы, во-первых, проверим достаточность материалов для списания – ранее заполненная табличная часть, служит лишь подсказкой пользователю, к тому же, он может, в процессе работы, получать сведения об остатках и себестоимости единицы, а может и не получать, оставляя соответствующие поля табличной части пустыми, поэтому при проведении документа мы получим с помощью запроса нужные данные из базы.

Дальше все идет по уже знакомому вам плану – мы сверяем количество материалов, которое пользователь хочет продать с количеством остатков и принимаем решение либо о продолжении работы, либо – об отмене процедуры проведения документа.

После того, как у нас есть хранилище данных о продажах, мы построим соответствующий отчет. На базе тех данных, которые у нас есть, можно построить различные отчеты – все зависит от того, какие именно данные нас интересуют. Предположим, мы заинтересованы в сведениях о продажах по контрагентам. Нас интересует количественный показатель продажи номенклатурной позиции и прибыль от продажи.

Практическая работа №16 Работа с группой справочников

Цель: научиться формированию отчетов на основе запроса

ХОД РАБОТЫ:

1. Создадим новый отчет, дадим ему имя **ПродажиПоКонтрагентам**, добавим его в подсистему **ОперативныйУчетМатериалов**.

2. Откроем основную схему компоновки данных отчета. Добавим новый набор данных – **Запрос**. Создадим с помощью конструктора запроса запрос следующего содержания, Рисунок 16.1.

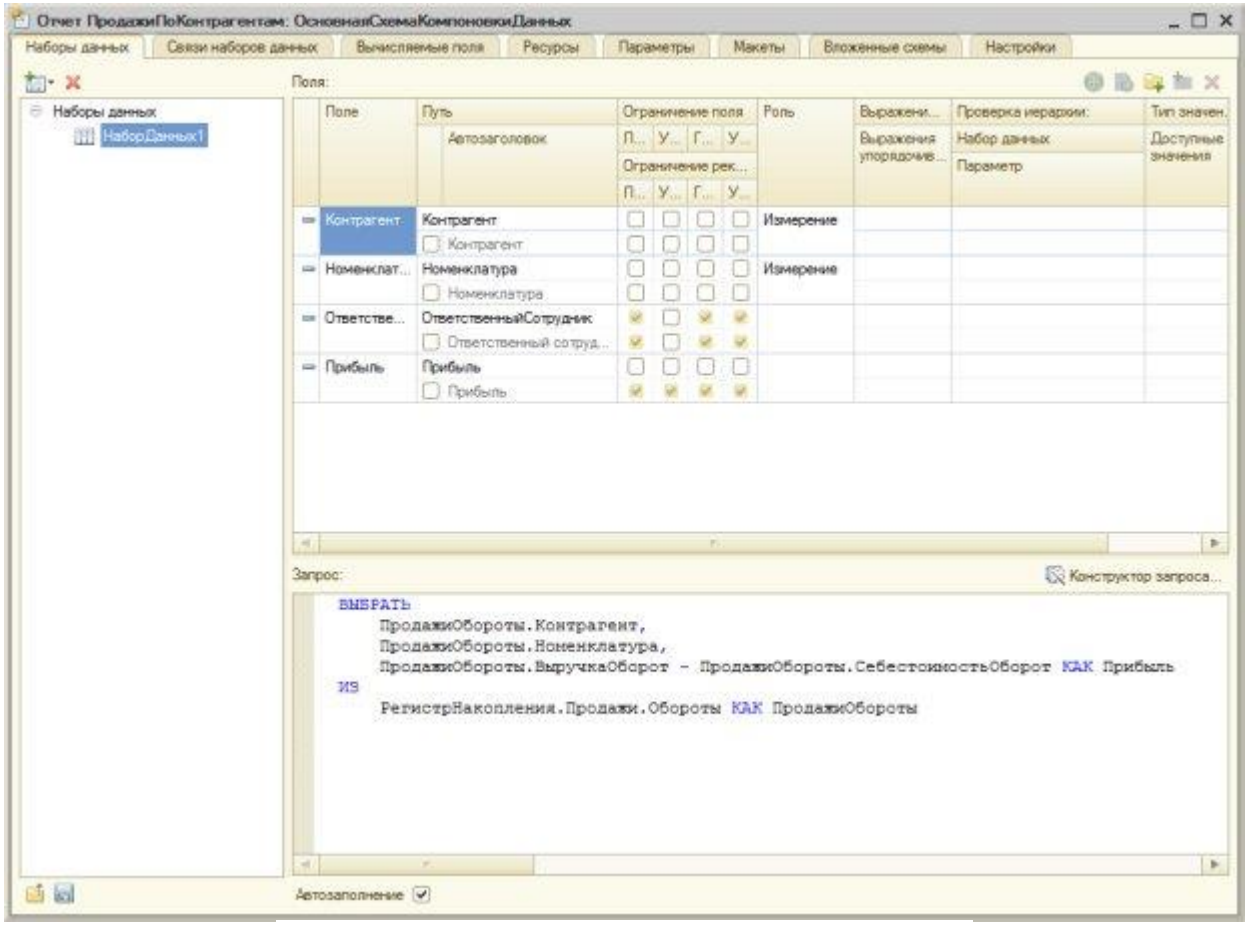


Рисунок 16.1- Набор данных в конструкторе СКД

3. Мы выбираем из виртуальной таблицы **ПродажиОбороты** данные о контрагенте, о количестве проданных материалов и получаем показатель прибыли

4. На закладке **Ресурсы** мы указываем, что поле **Прибыль** является ресурсом, применяем к нему функцию **Сумма**, Рисунок 16.2.

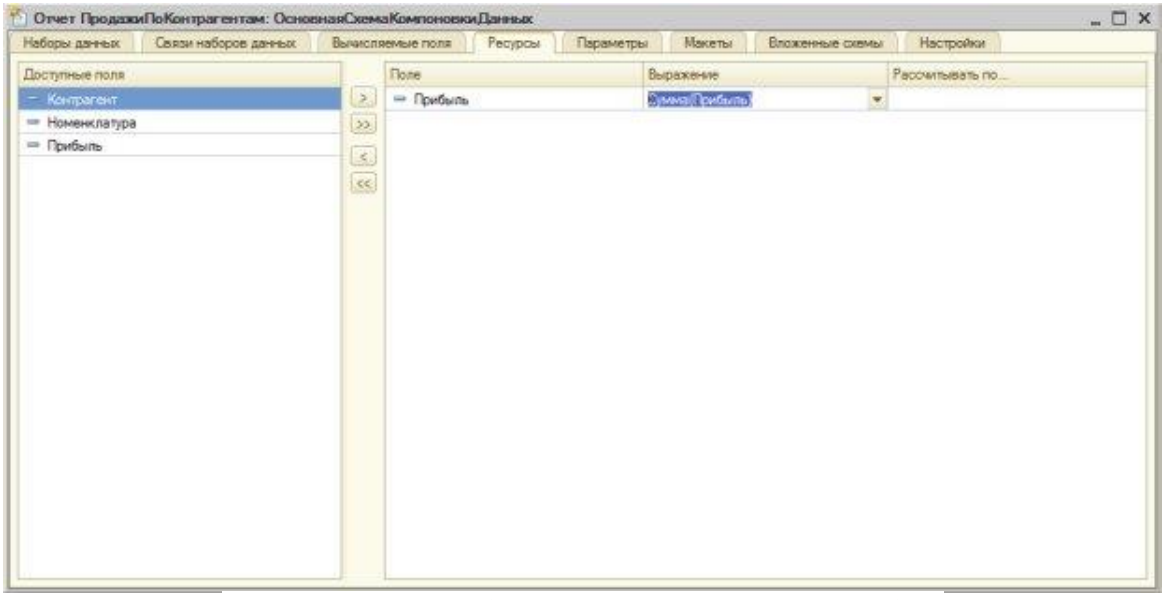


Рисунок 16.2- Выбор ресурсов в конструкторе СКД

5. На вкладке **Настройки** мы включим отображение параметров **Начало** периода и **Конец** периода в пользовательском интерфейсе и, вызывав конструктор настроек компоновки данных, выберем тип отчета – **Список**, в качестве полей, которые будут отображаться в отчете, выберем **Контрагент**, **Номенклатура** и **Прибыль**, Рисунок 16.3.

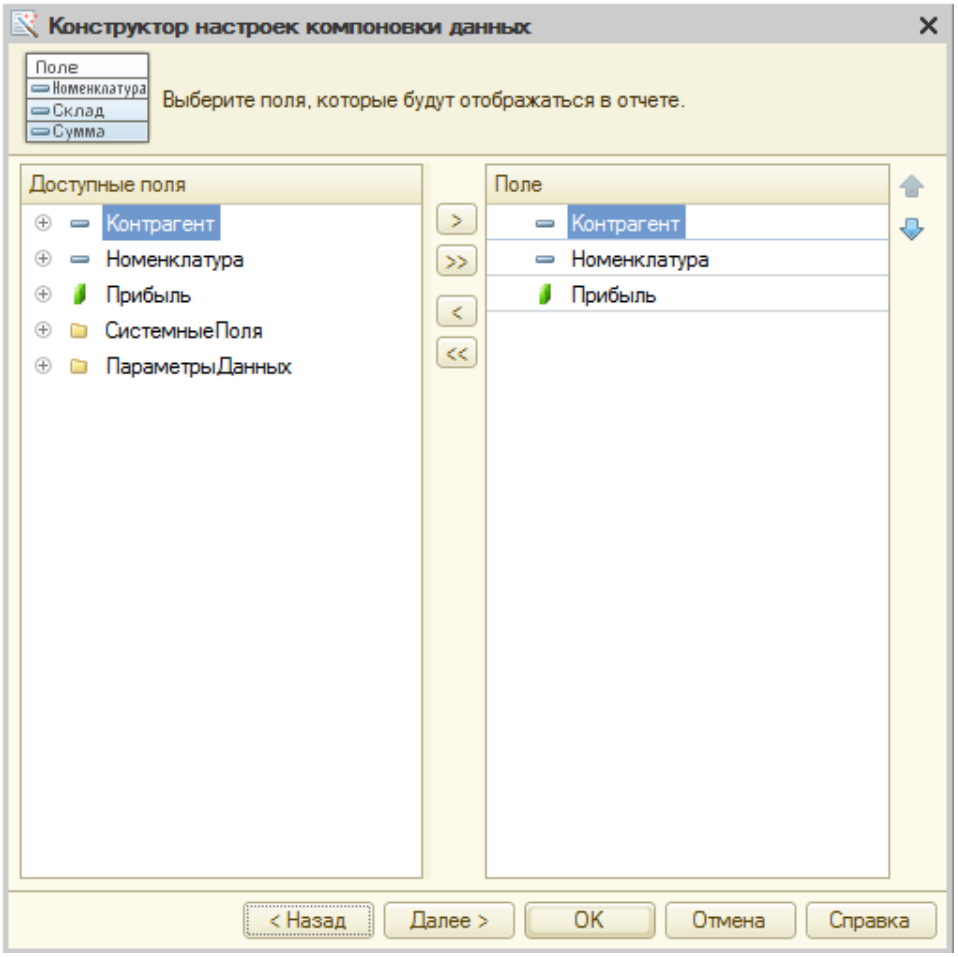


Рисунок 16.3- Выбор полей отчета при представлении отчета в виде списка

6. В следующем окне в качестве полей, по которым будет производиться группировка данных, выберем **Контрагент** и **Номенклатура**.

7. В качестве поля для упорядочивания, в следующем окне конструктора, выберем поле **Контрагент**, упорядочивать будем по возрастанию. После этого нажмем **ОК** и вкладка**Настройки** окна конструктора СКД приобретет такой вид, Рисунок 16.4.

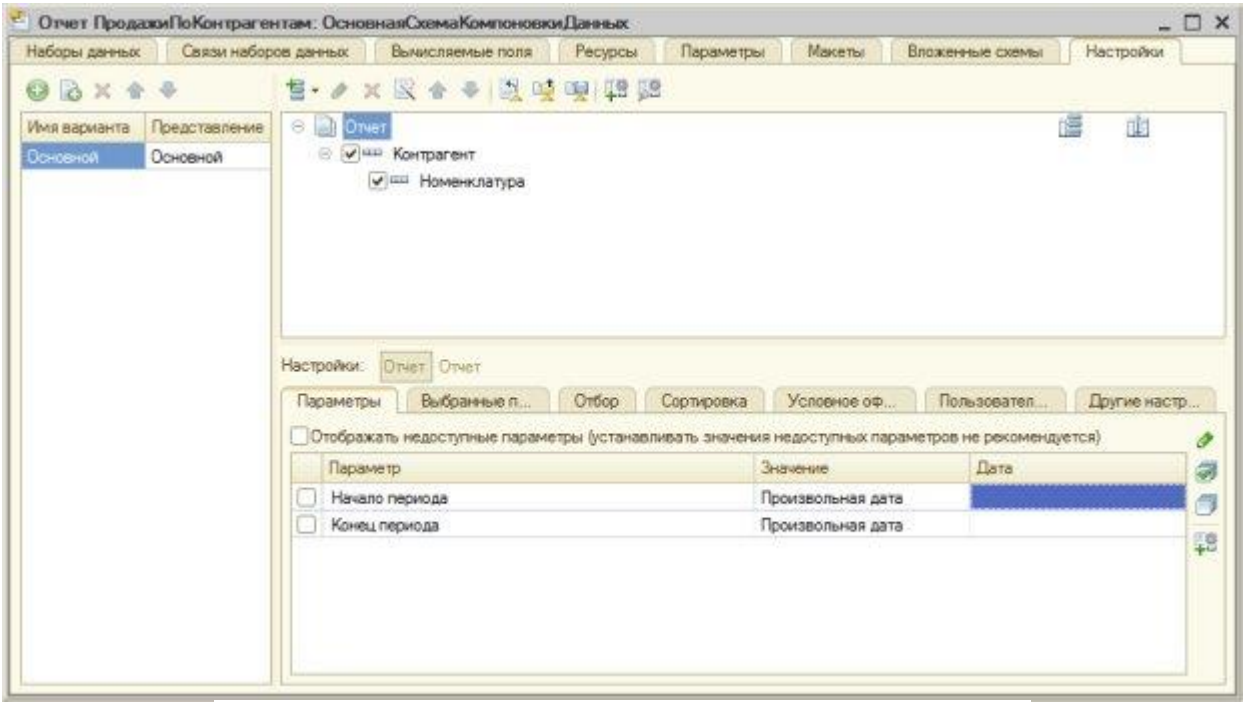


Рисунок 16.4- Вкладка Настройки окна конструктора СКД

На рисунке 16.5. представлены результаты работы отчета в пользовательском режиме.

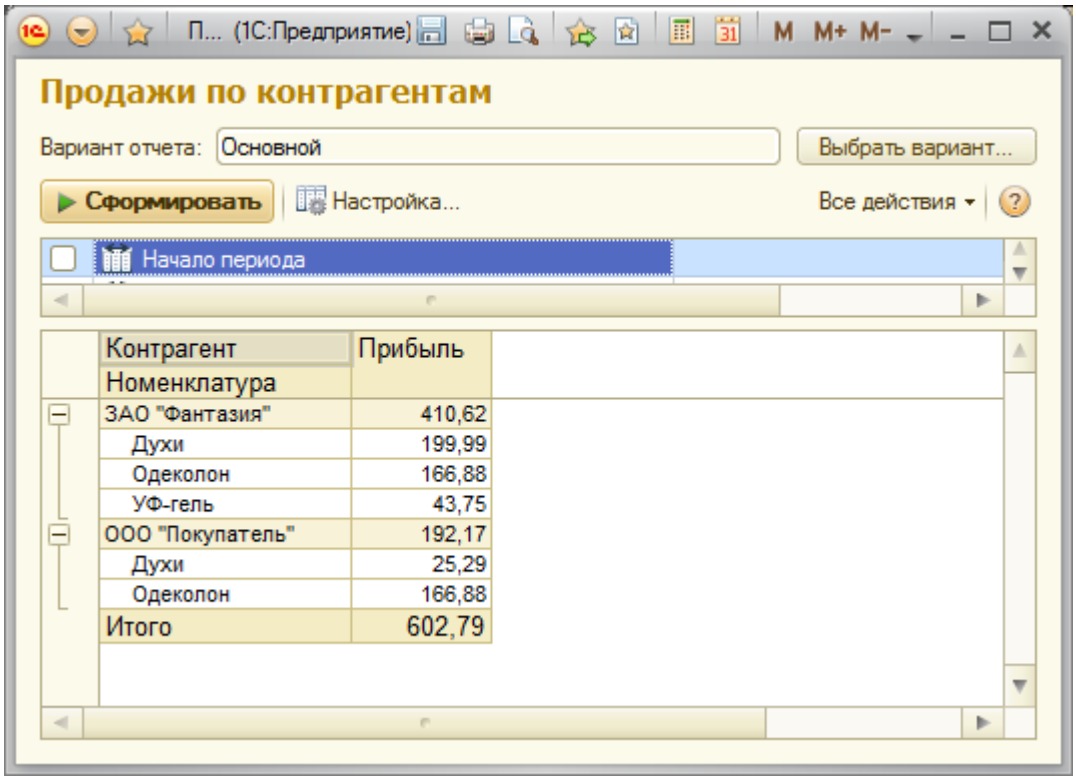


Рисунок 16.5- Отчет Продажи по контрагентам

Практическая работа №17 Программирование простого отчета. Создание макета

Цель: научиться программированию документа с расчетом себестоимости товаров по регистру

ХОД РАБОТЫ:

1. Доработаем форму документа **РеализацияМатериалов**. Документ проводится по двум регистрам накопления – по регистру **Продажи** и по регистру **ОстаткиМатериалов**. Удобно было бы иметь возможность прямо из документа перейти в окно регистра и просмотреть его содержимое после проведения документа. Для того, чтобы реализовать это, нам понадобится закладка **Глобальные команды** закладки **Команды** окна редактора форм. Все, что нужно для размещения на командной панели формы дополнительных кнопок, ведущих к регистрам – перетащить нужную команду на панель, Рисунок 17.1.

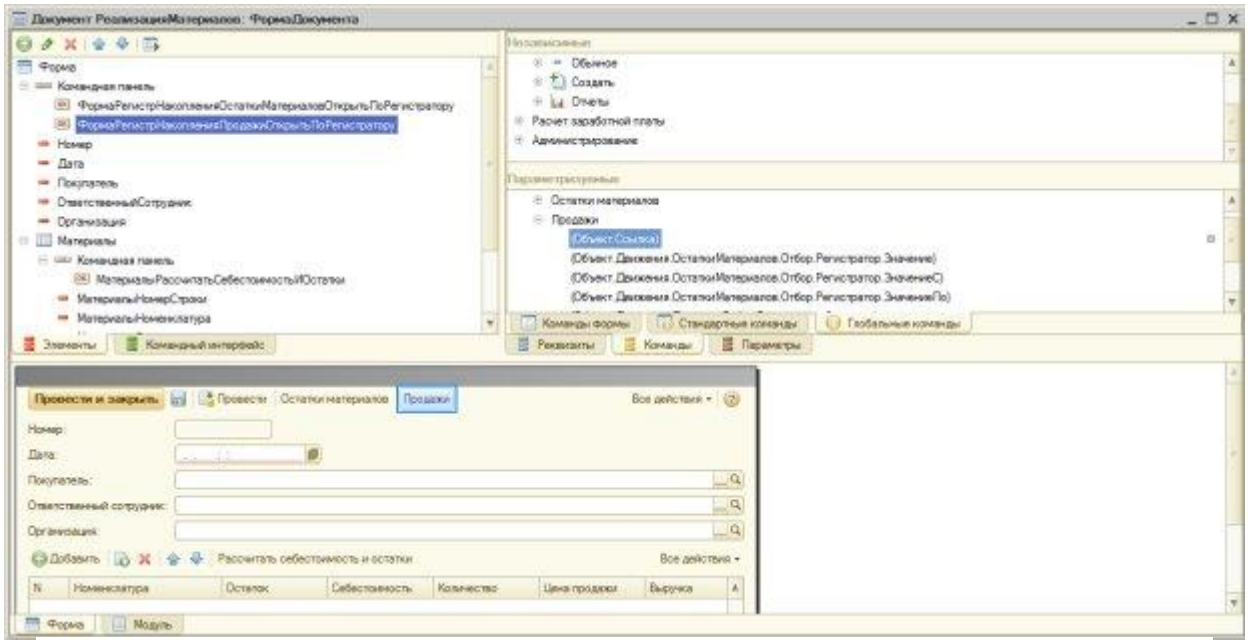


Рисунок 17.1- Кнопки перехода в регистры накопления на командной панели формы

2. Для создания агрегатов используется конструктор агрегатов, который можно вызвать, нажав на кнопку **Агрегаты**, расположенную на закладке **Данные** окна редактирования объекта для оборотного регистра накопления, Рисунок 17.2

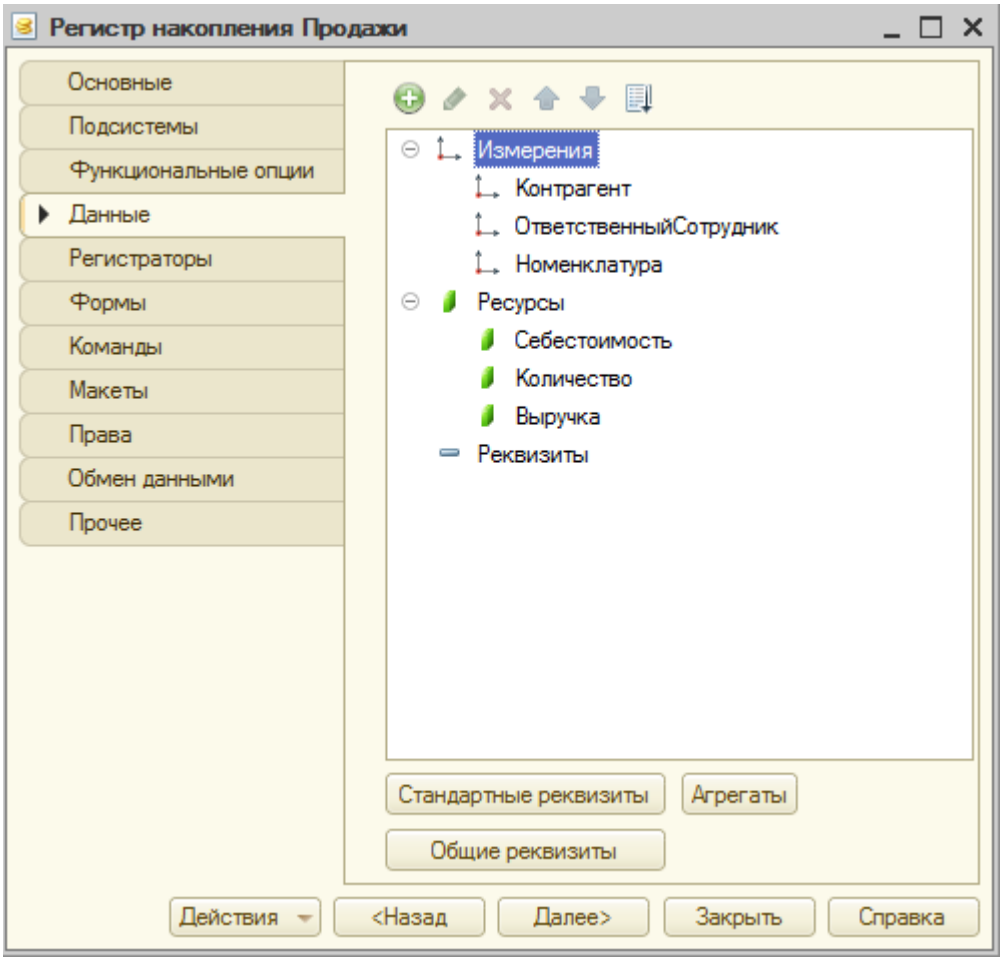


Рисунок 17.2- Кнопка Агрегаты в окне редактирования объекта

3. По нажатию на эту кнопку можно открыть окно **Конструктора агрегатов** (Рисунок 17.3.), где можно добавлять новые агрегаты, настраивать состав измерений регистра, входящих в агрегаты, а так же их свойства. При установленной периодичности **Авто** система сама подбирает подходящую периодичность для агрегата. Мы установили такую периодичность для агрегата, в который входят сведения по измерениям **Контрагент** и **Номенклатура**.

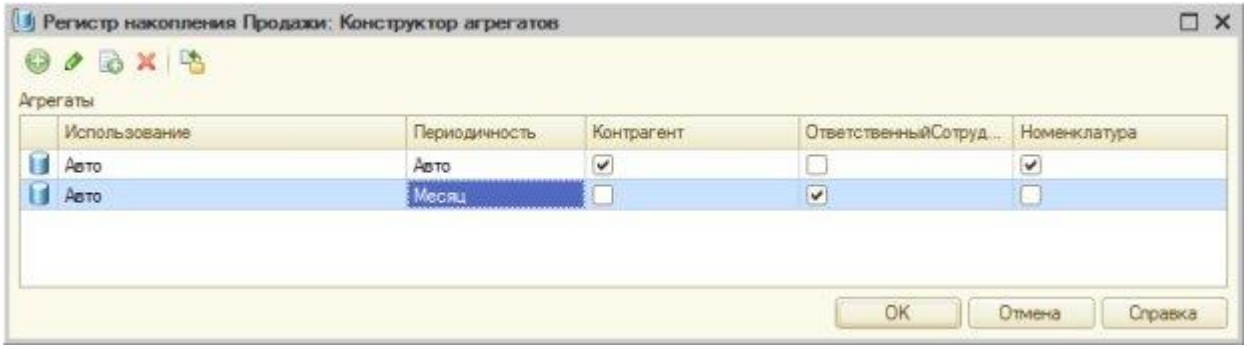


Рисунок 17.3- Конструктор агрегатов

4. По измерению **ОтветственныйСотрудник** мы собираемся получать данные ежемесячно, поэтому у агрегата, содержащего указанное измерение, мы установим периодичность **Месяц**.

В нашей учебной базе, содержащей небольшие объемы данных, мы вряд ли сможем оценить полезность агрегатов, однако, при работе с большими базами данных они способны

повысить производительность отчетов.

5. В нашей конфигурации есть документы, последовательность ввода которых способна влиять на данные, хранимые в системе. Это три документа – **ПоступлениеМатериалов**,**ОтпускМатериаловМастеру** и **РеализацияМатериалов**..

6. Для контроля за правильной последовательностью проведения документов в системе имеется объект, который так и называется – **Последовательность**.

Последовательность позволяет автоматически контролировать хронологическую последовательность проведения документов, включенных в нее, а так же – для документов, влияющих на состояние отслеживаемых последовательностью регистров.

7.Создадим новую последовательность (ветвь дерева конфигурации **Последовательности** находится в ветви **Документы**), назовем ее **СебестоимостьМатериалов**, Рисунок 17.4.

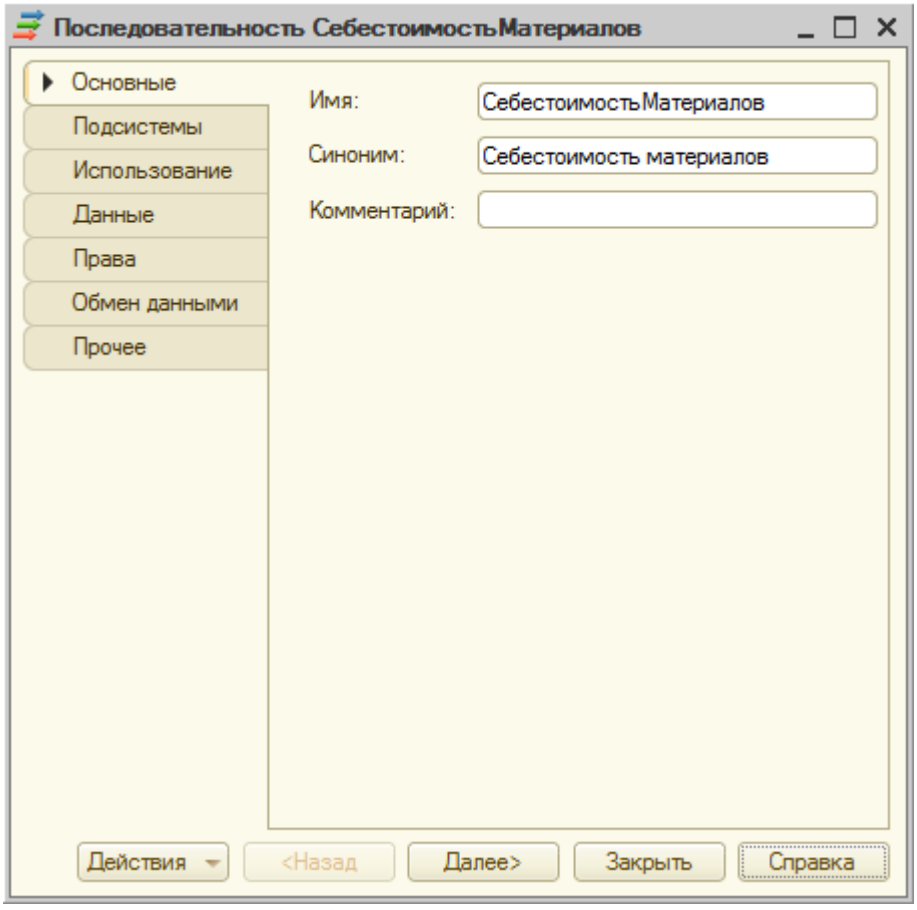


Рисунок 17.4- Создание новой последовательности

8.Включим последовательность **СебестоимостьМатериалов** в подсистему **ОперативныйУчетМатериалов**.

9. На закладке **Использование**, Рисунок 17.5., мы оставим параметр **Перемещение границы при проведении** в значении **Перемещать**, в поле **Входящие документы** внесем документы **ОтпускМатериаловМастеру** и **РеализацияМатериалов**, в поле **Движения,**

влияющие на последовательность, внесем регистр накопления**ОстаткиМатериалов**.

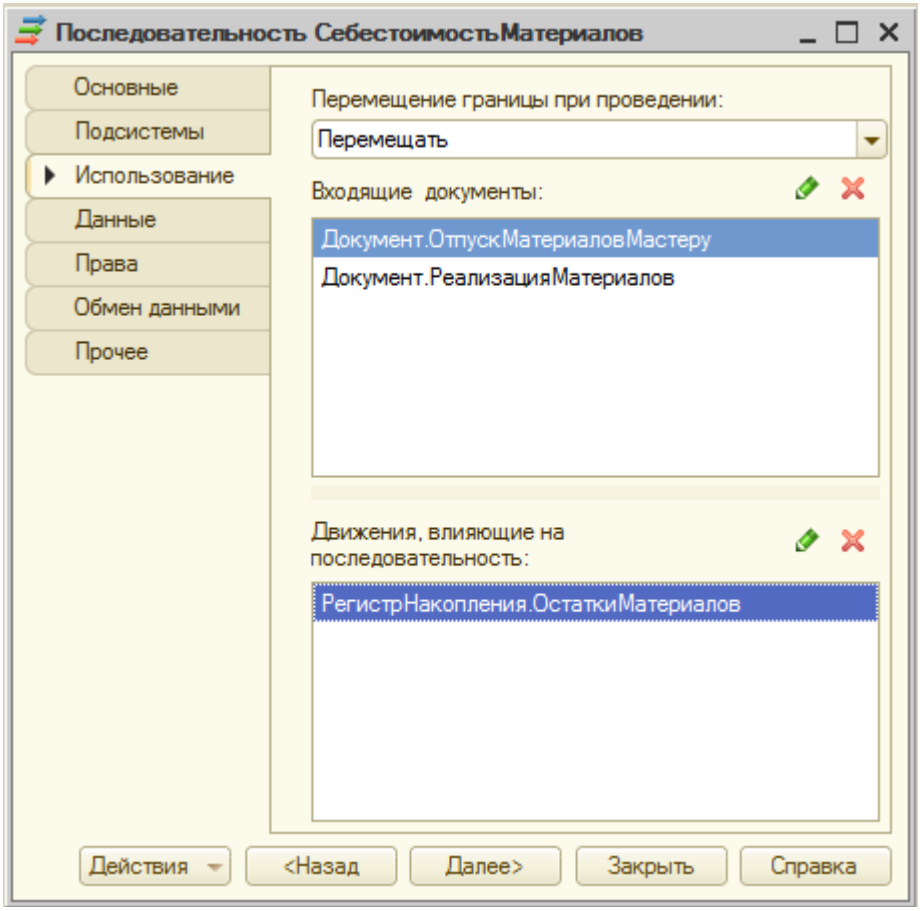


Рисунок 17.5- Настройка последовательности

10.Сущность действия последовательности заключается в следующем. Она отслеживает документы, влияющие на нее (и документы, влияющие на состояние регистра, в нашем случае это документ **ПоступлениеМатериалов**) и, в зависимости от даты проведения документов перемещает границу либо вперед (при нормальной хронологической последовательности ввода документов), либо – назад – в том случае, если документ введен (проведен) не в нормальном порядке.

11.Граница последовательности указывает нам на документ, документы, введенные после которого могут отражать некорректные данные. Последовательность можно восстанавливать – при восстановлении последовательности перепроводятся документы, входящие в нее. В нашем случае это документы, которыми мы списываем материалы. Документы поступления материалов влияют на последовательность, но они вводят в систему исходные данные для расчета себестоимости, поэтому внеочередное проведение документа поступления материалов на другие документы поступления не влияет, но вполне может повлиять на документы списания материалов.

12.При восстановлении последовательности производится автоматическое проведение нужных документов и граница последовательности устанавливается на последний из них. Для управления последовательностями в пользовательском режиме можно выполнить команду **Главное меню > Все функции > Стандартные > Проведение документов**, и там, на закладке Последовательности, Рисунок 17.6., посмотреть состояние последовательностей, и,

при необходимости, восстановить их.

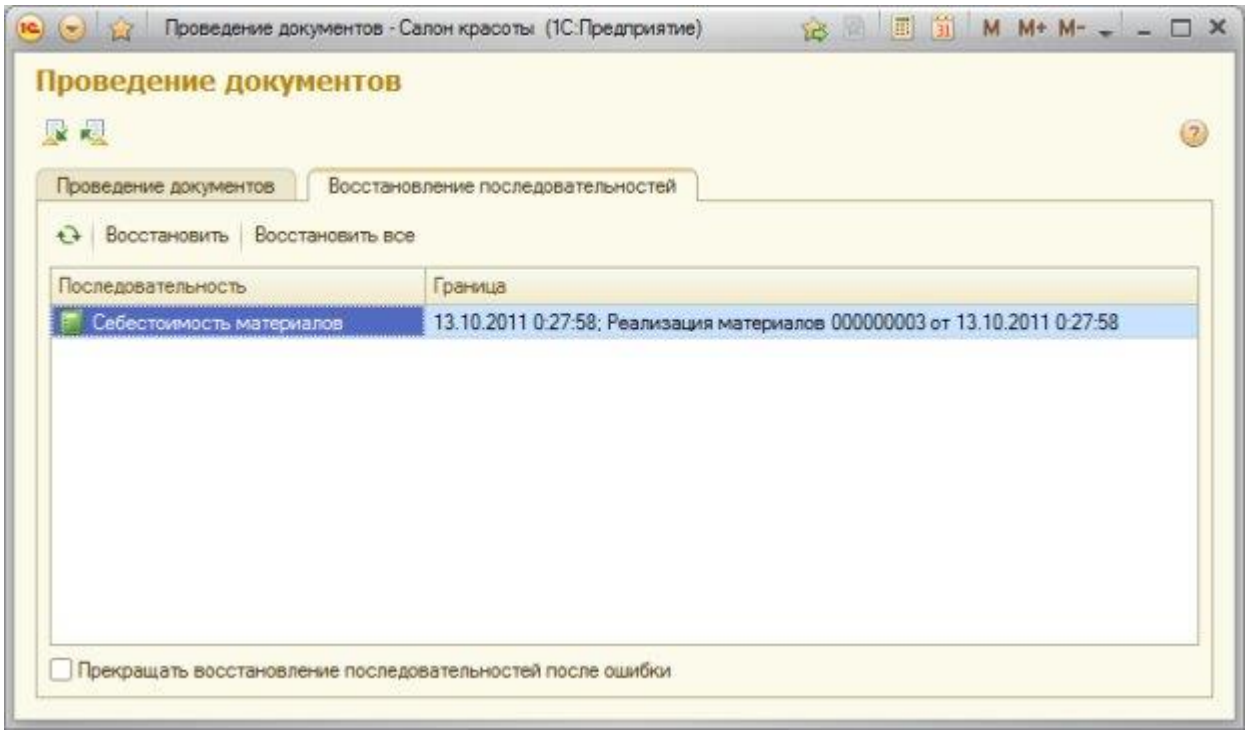


Рисунок 17.6- Работа с последовательностями

13. Здесь мы видим, что граница последовательности установлена на документ **Реализация материалов** от 13.10.2011. Введем документ **ПоступлениеМатериалов**, например, от 10.10.2011. Это приведет к такому состоянию последовательности, Рисунок 17.7.

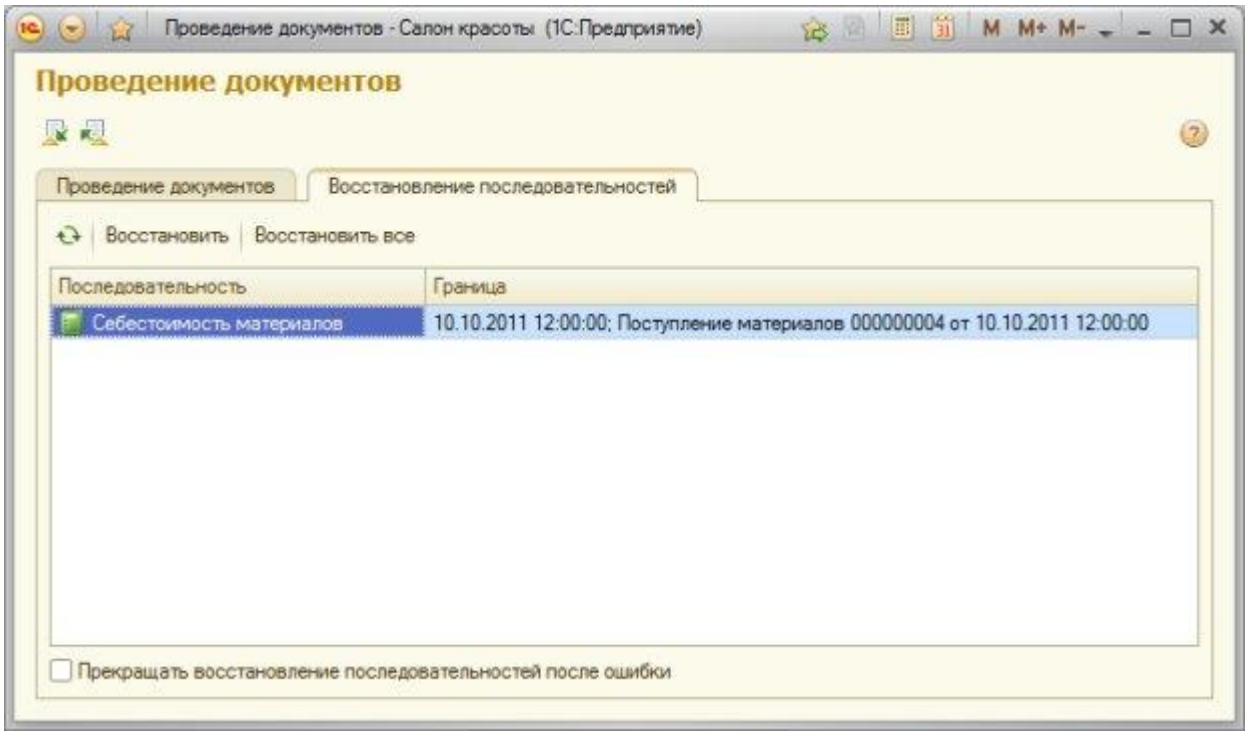


Рисунок 17.7. Перемещение границы последовательности при вводе документа в прошлом периоде

14. Восстановление последовательности приведет к перепроведению документов, входящих в

нее и снова установит ее границу на документ Реализация материалов от 13.10.2011.

15. Еще один полезный объект, который можно найти в ветви **Документы** дерева конфигурации – это **нумераторы**. Нумератор позволяет задавать единые правила нумерации для различных документов. Создадим новый нумератор, назовем его **НумераторРасходныхДокументов**, Рисунок 17.8.

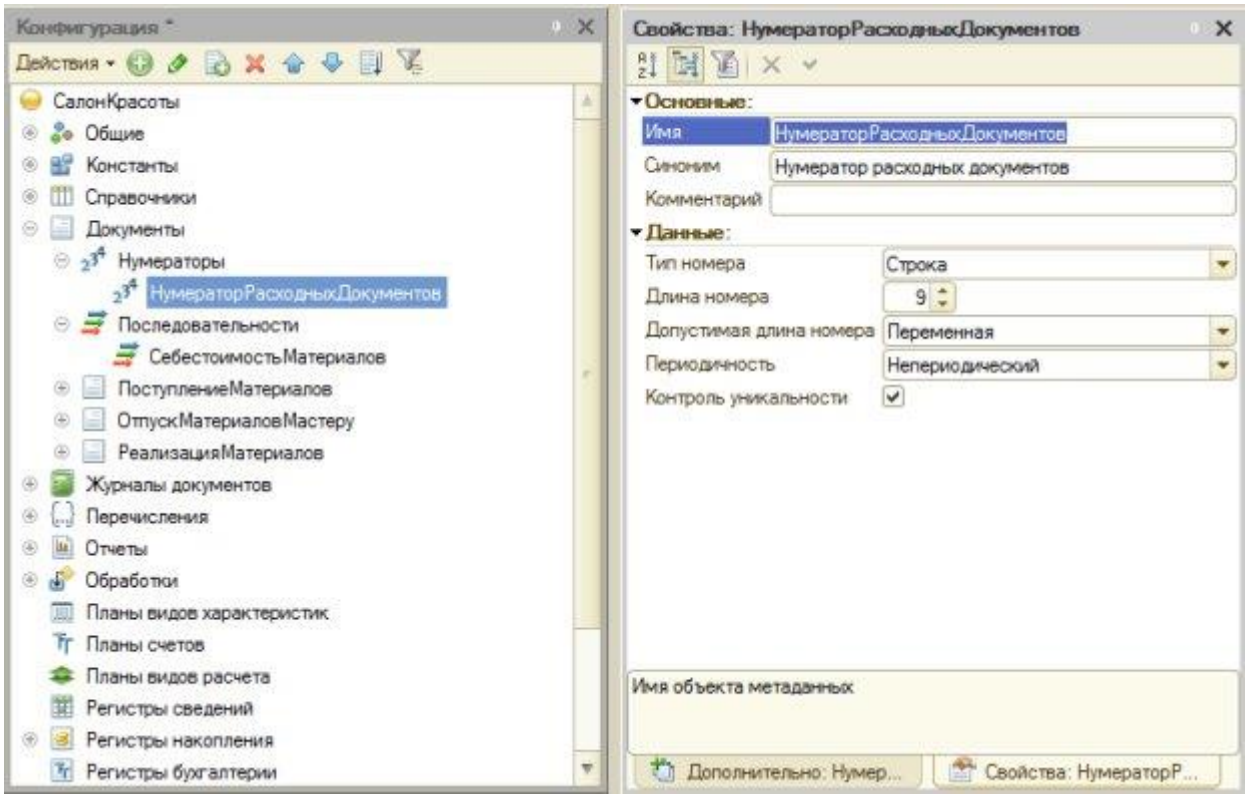


Рисунок 17.8- Создание нового нумератора

16.Редактирование свойств нумератора осуществляется в окне свойств объекта, для него не предусмотрено окна редактирования объекта.

Благодаря нумератору документы разных видов, которым он назначен (делается это на вкладке **Нумерация** окна настройки свойств документа), Рисунок 17.9. приобретают сквозную нумерацию.

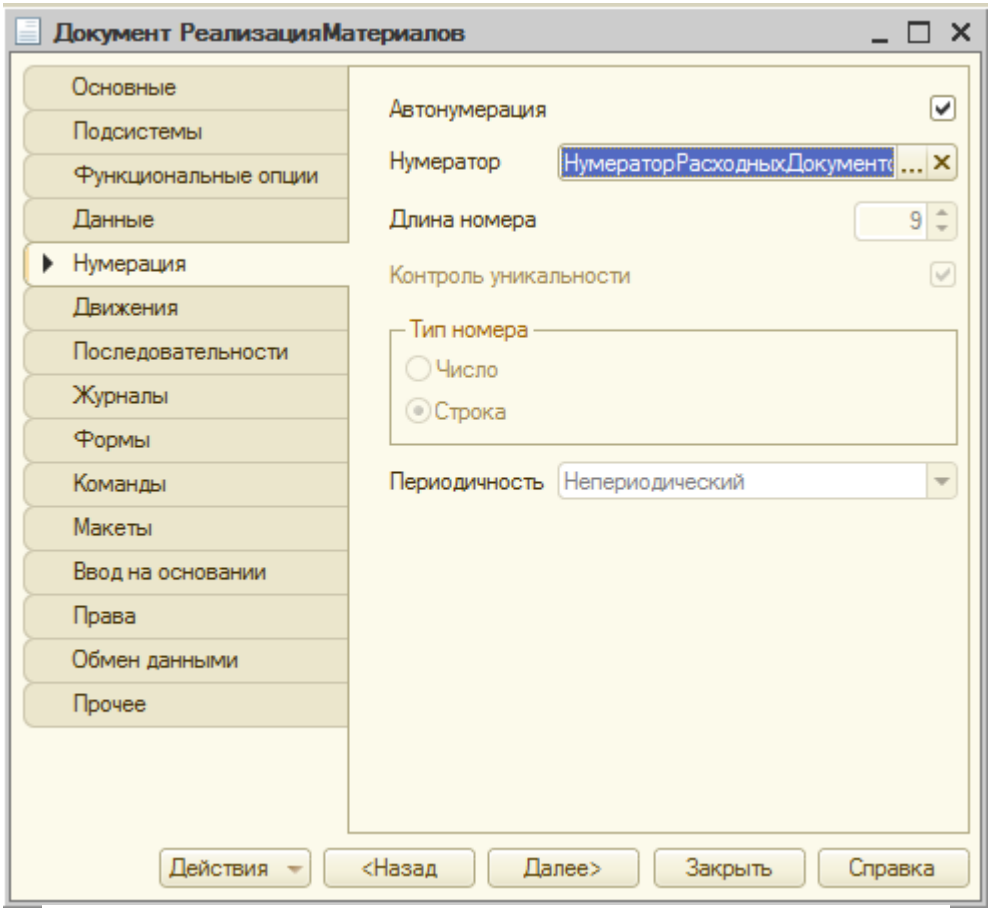


Рисунок 17.9- Настройка использования нумератора в документе

То есть, если, например, мы создали документ РеализацияМатериалов и он приобрел номер 000000001, то, если следующим документом с тем же нумератором будет документ ОтпускМатериаловМастеру, ему автоматически будет присвоен номер 000000002.

Практическая работа №18 Программирование автозаполнения реквизита справочника из других справочников

Цель: научиться созданию отчетов по различным регистрам и условиям

ХОД РАБОТЫ:

1. Рассмотрим использование периодического независимого регистра сведений на примере с курсами валют. Для начала создадим новый справочник. Назовем его **Валюты**, включим в подсистему **БухгалтерскийУчет**, других настроек для данного справочника задавать не будем. Нас вполне устроит возможность задать наименование валюты с помощью стандартного реквизита.

2. Создадим новый регистр сведений **КурсыВалют**, установим его периодичность – В пределах дня, Рисунок 18.1.

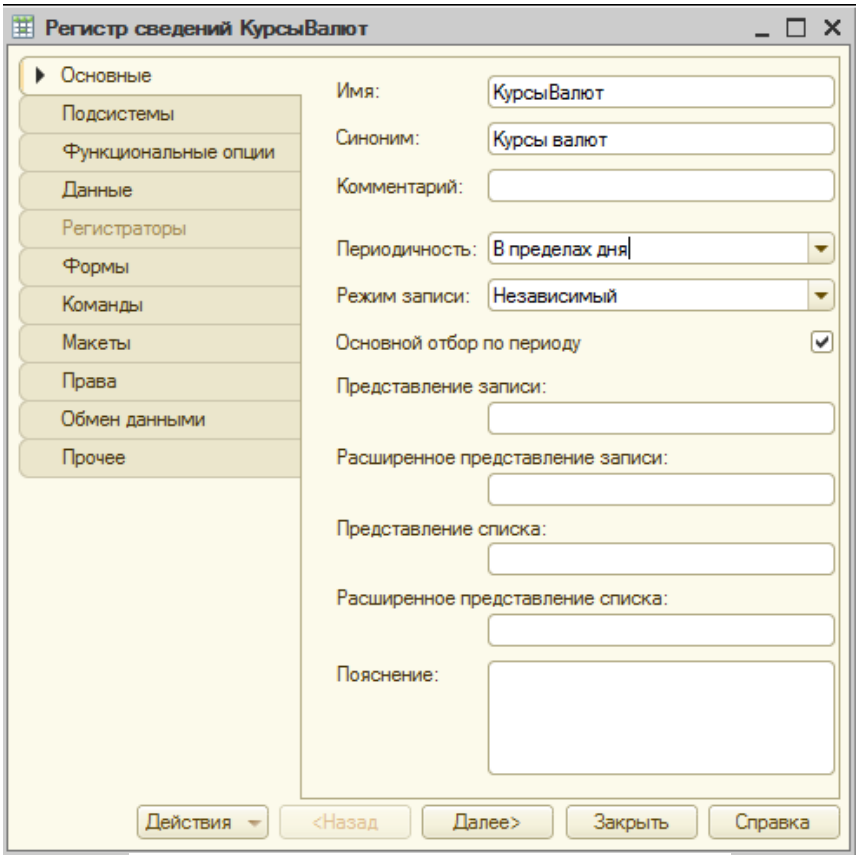


Рисунок 18.1- Создание регистра сведений

3. На закладке **Подсистемы** включим регистр в подсистему **БухгалтерскийУчет**.

4. На закладке **Данные** (Рисунок 18.2) добавим следующее:

Измерения:

Имя: Валюта, тип СправочникСсылка.Валюты. Флаги Ведущее, Основной отбор и Запрет незаполненных значений установлены

Ресурсы

Имя: Курс, тип Число, длина 10, точность 4

Имя: Кратность, тип Число, длина 10, точность 0

Исключим регистр из состава общего реквизита **Организация.**

5. В ресурсе **Курс** мы будем хранить курс, который может выражаться числом с точностью до 4-х знаков после запятой.

6. В ресурсе **Кратность** мы будем хранить кратность валюты по отношению к рублю. Например, для американского доллара курс может быть 34.3234, а кратность 1 – то есть, за один доллар дают 34.3234 рубля.

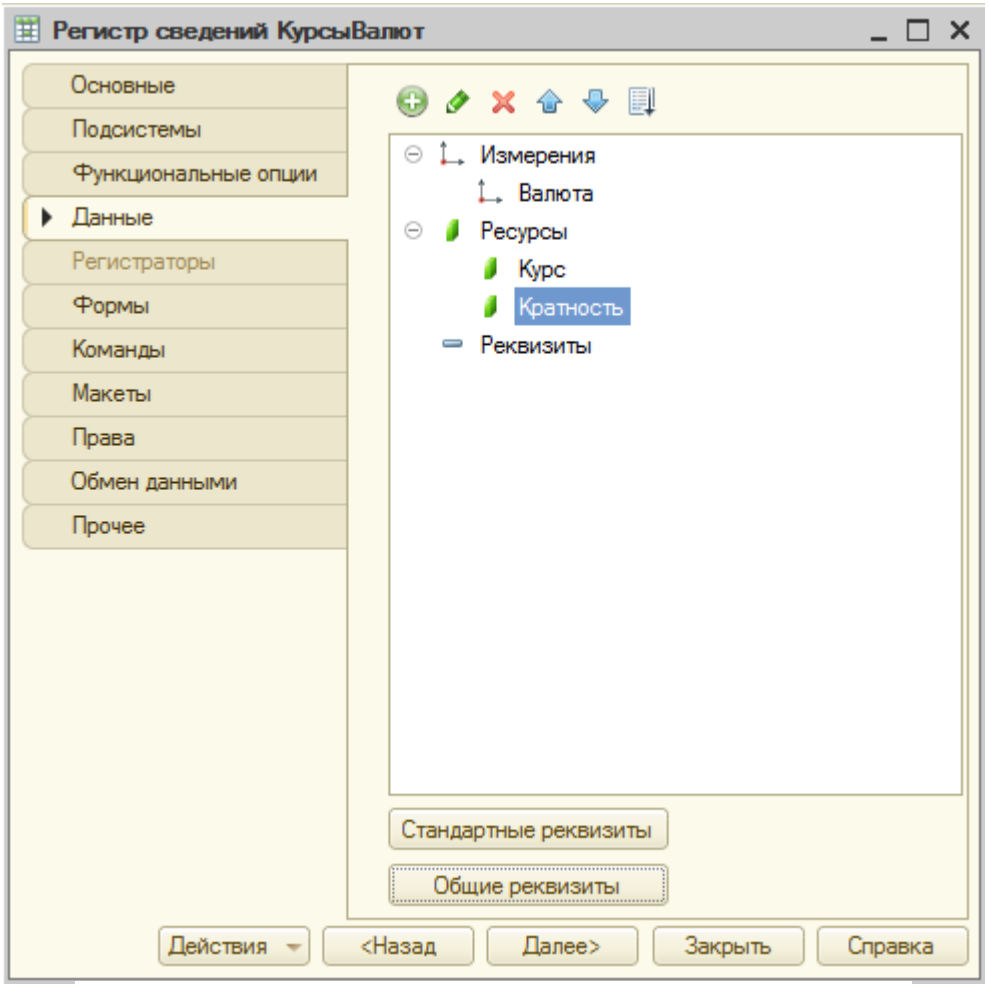


Рисунок 18.2- Настройка состава данных регистра сведений

7. Воспользуемся созданной группой объектов в пользовательском режиме. Здесь мы можем вручную вводить данные по валютам, Рисунок 18.3.

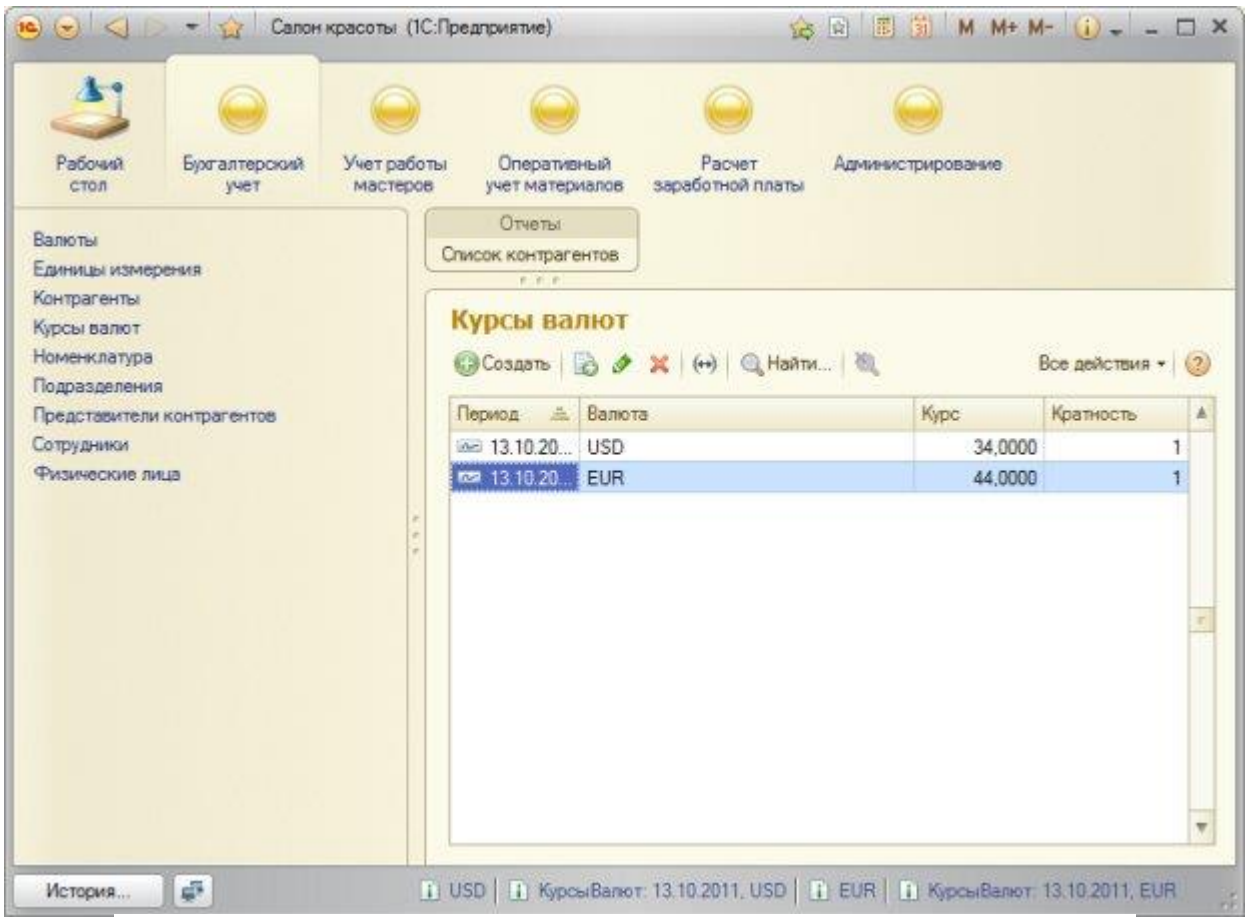


Рисунок 18.3- Заполнение регистра сведений в пользовательском режиме

Теперь в нашей конфигурации есть сведения о курсах валют.

9. По регистру сведений можно создать отчет через СКД. Для этого добавим новый объект отчетов «КурсыВалют»: Включим объект в подсистему «Справочники» (рисунок 18.4):

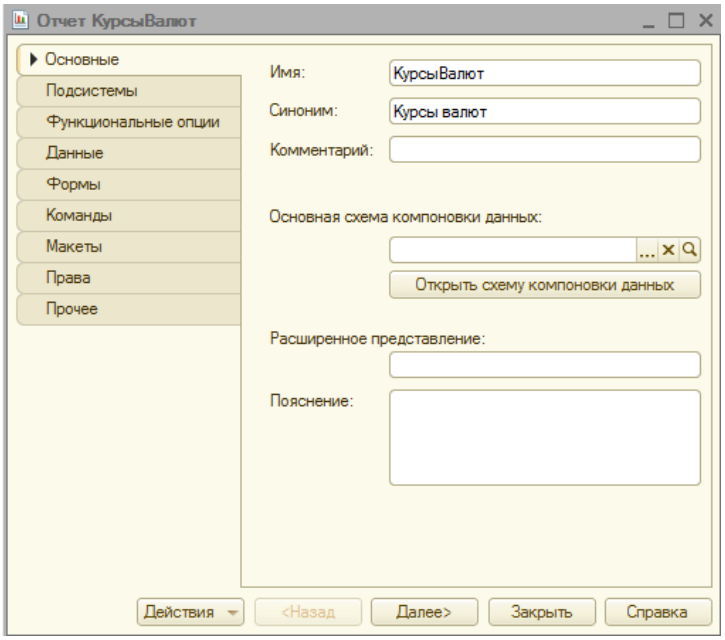


Рисунок 18.4 – Создание отчета КурсыВалют

10. Создадим новую СКД отчета и укажем в ней объекты запроса набора данных:

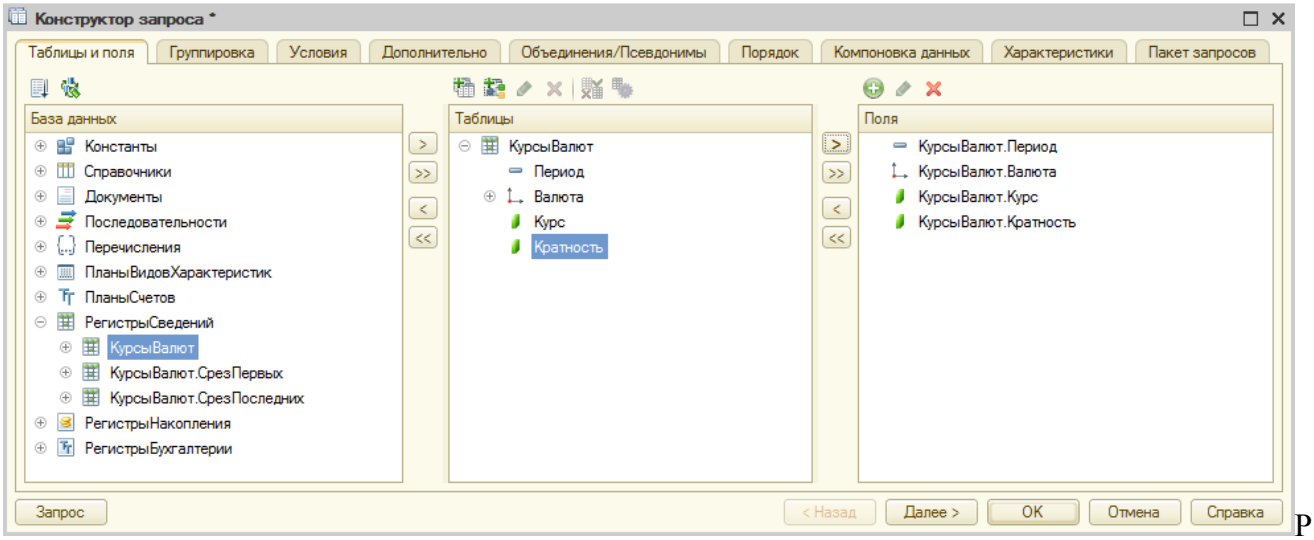


Рисунок 18.5 – Набор данных запроса отчета КурсыВалют

11. Выолним настройки отчета (рисунок 18.6):

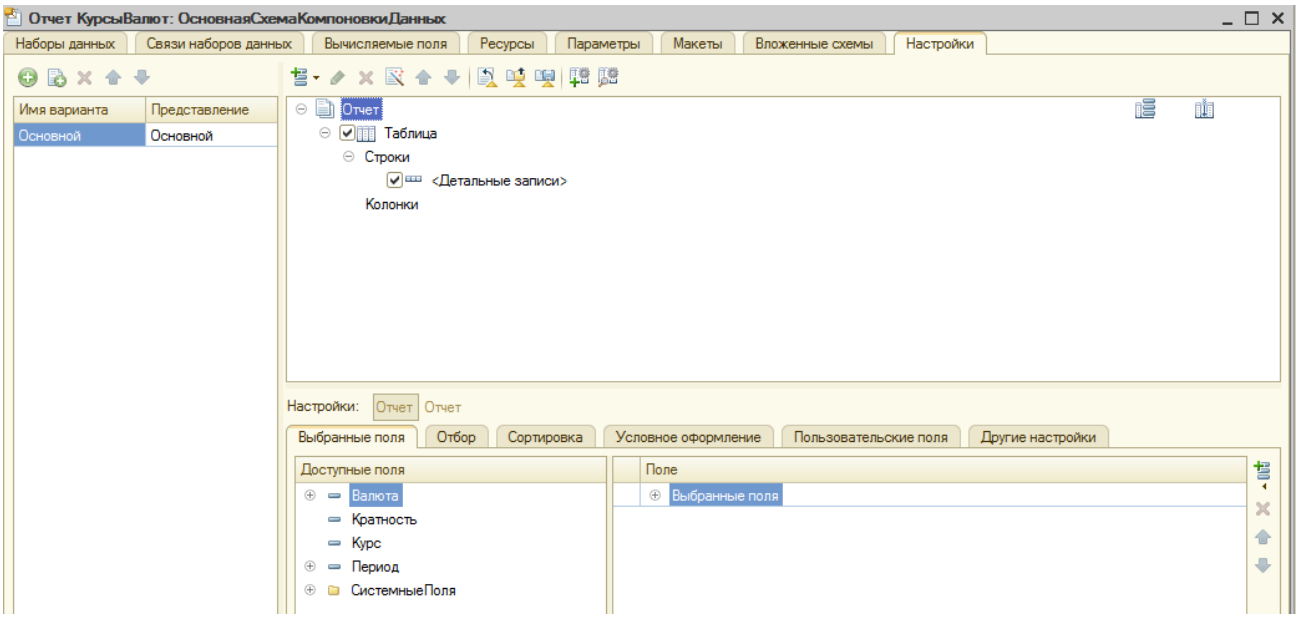


Рисунок 18.6 – Настройки отчета КурсыВалют

12. В пользовательском режиме проверим работу отчета (рисунок 18.7):

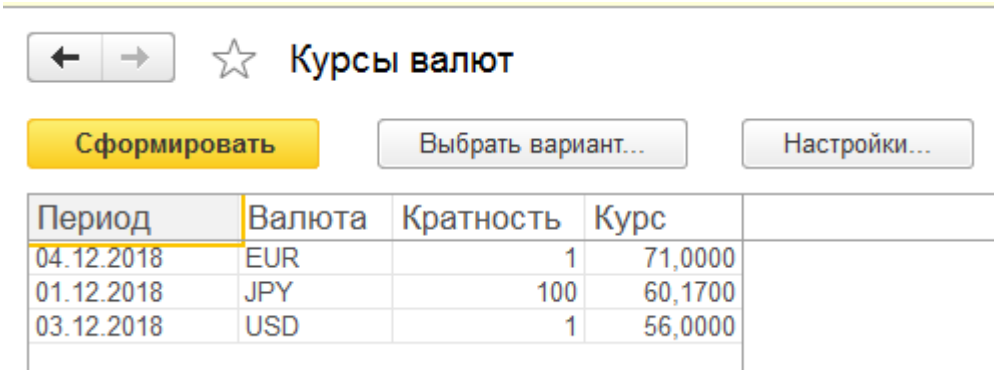


Рисунок 18.7 – Вид отчета КурсыВалют в режиме пользователя

Практическая работа №19 Программирование документов конфигурации

Цель: научиться использовать логические операторы в запросе

ХОД РАБОТЫ

1. Решим следующую задачу. Нам хотелось бы вывести курс валюты на текущую дату в списке справочника **Валюты**.

2. Для начала создадим форму списка справочника **Валюты**.

3. Обратите внимание на то, что реквизит **Список** имеет тип **ДинамическийСписок**. Это означает, что мы можем вмешаться в создание этого списка, так как он строится на основе некоего запроса, генерируемого, в данном случае, системой автоматически. Для того, чтобы самостоятельно отредактировать запрос, который лежит в основе динамического списка, нам нужно в окне его свойств (Рисунок 19.1.) установить флаг **ПроизвольныйЗапрос**.

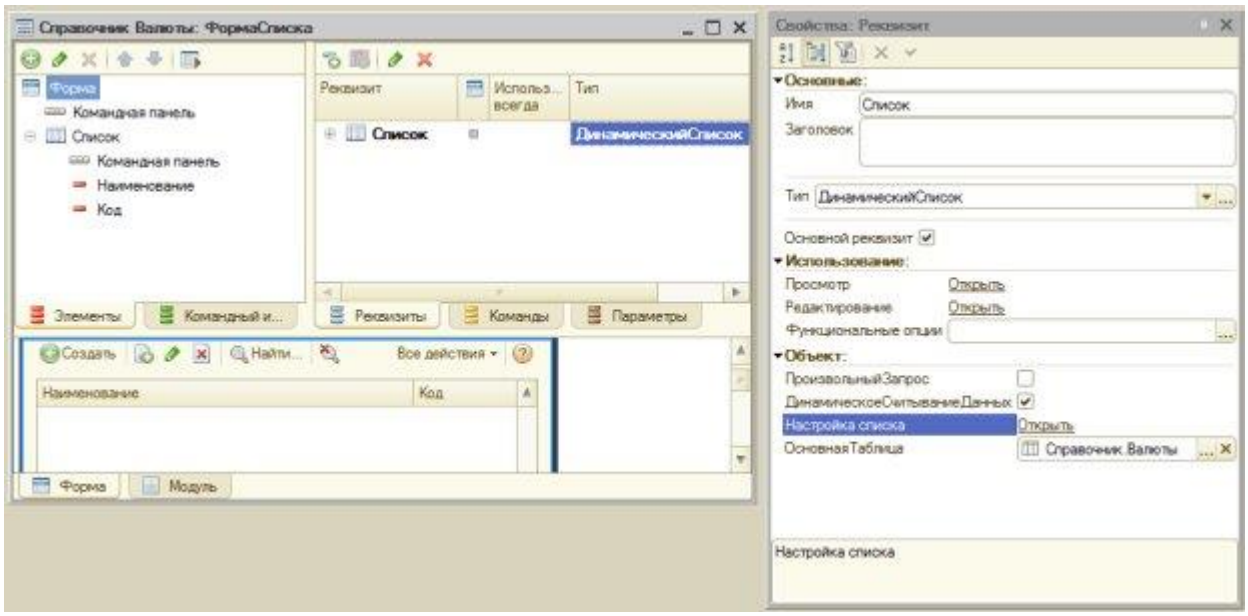


Рисунок 19.1- Редактирование свойств динамического списка

4. После установки этого флага мы можем нажать на ссылку **Открыть** у поля **Настройка**

списка и увидеть, какой именно запрос система построила для создания этого списка, Рисунок 19.2

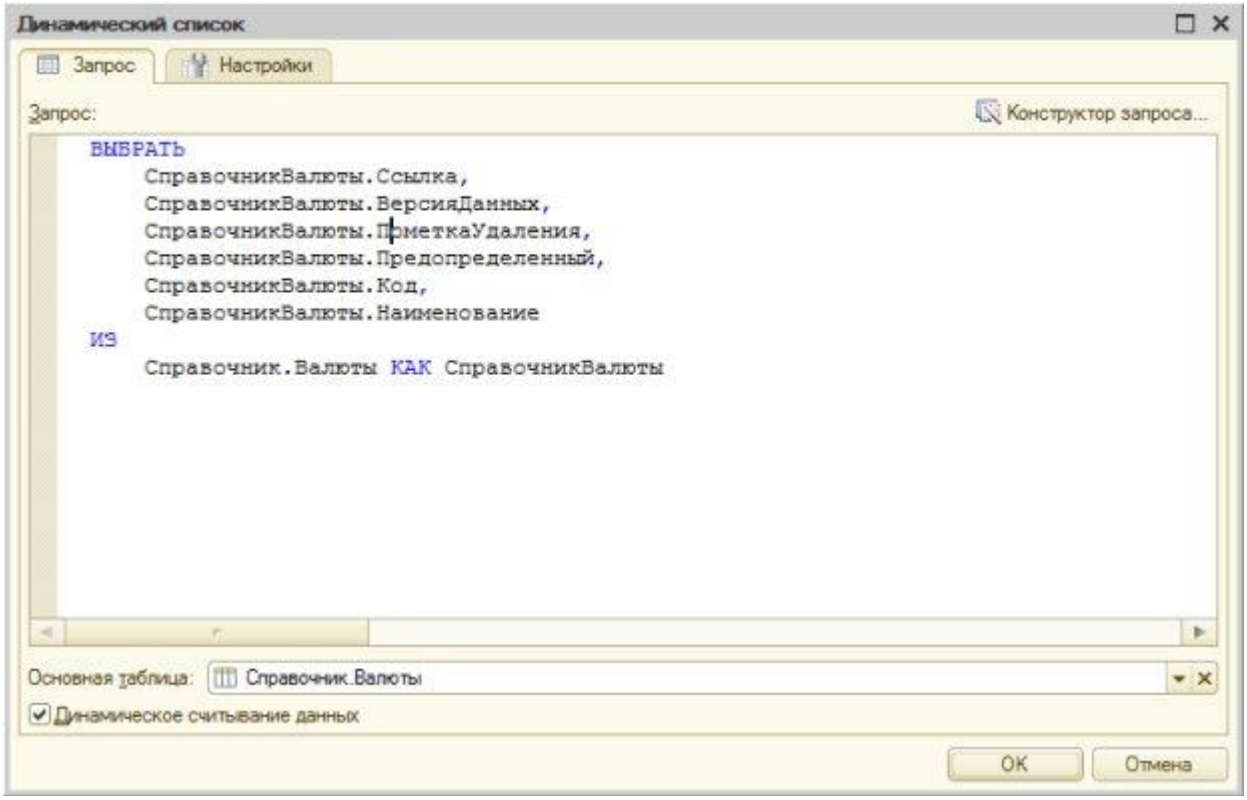


Рисунок 19.2- Запрос по умолчанию для заполнения динамического списка

5. Дополним запрос таким образом, чтобы он выводил в дополнение к запрошенным полям еще и наиболее свежее значение курса. Запрос можно ввести вручную в поле Запрос, воспользоваться конструктором запроса, доступным из этого же поля или, как мы уже делали, сначала отладить запрос в консоли запросов, а потом добавить его в поле **Запрос**. То, что мы хотим, можно сделать с помощью следующего запроса:

```
ВЫБРАТЬ
  СправочникВалюты.Ссылка,
  СправочникВалюты.ВерсияДанных,
  СправочникВалюты.ПометкаУдаления,
  СправочникВалюты.Предопределенный,
  СправочникВалюты.Код,
  СправочникВалюты.Наименование,
  КурсыВалютСрезПоследних.Курс
ИЗ
  Справочник.Валюты КАК СправочникВалюты
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.КурсыВалют.СрезПоследних КАК КурсыВалютСрезПоследних
    ПО СправочникВалюты.Ссылка = КурсыВалютСрезПоследних.Валюта
```

Здесь мы получаем из виртуальной таблицы **КурсыВалютСрезПоследних** наиболее поздние значения курса, не прибегая к специальной настройке ее параметров.

6. После показанной модификации запроса, формирующего динамический список и размещения в форме списка справочника нового поля **Курс**, которое будет доступно в списке реквизитов динамического списка, форма списка приобретет вид, показанный на Рисунок 19.3.

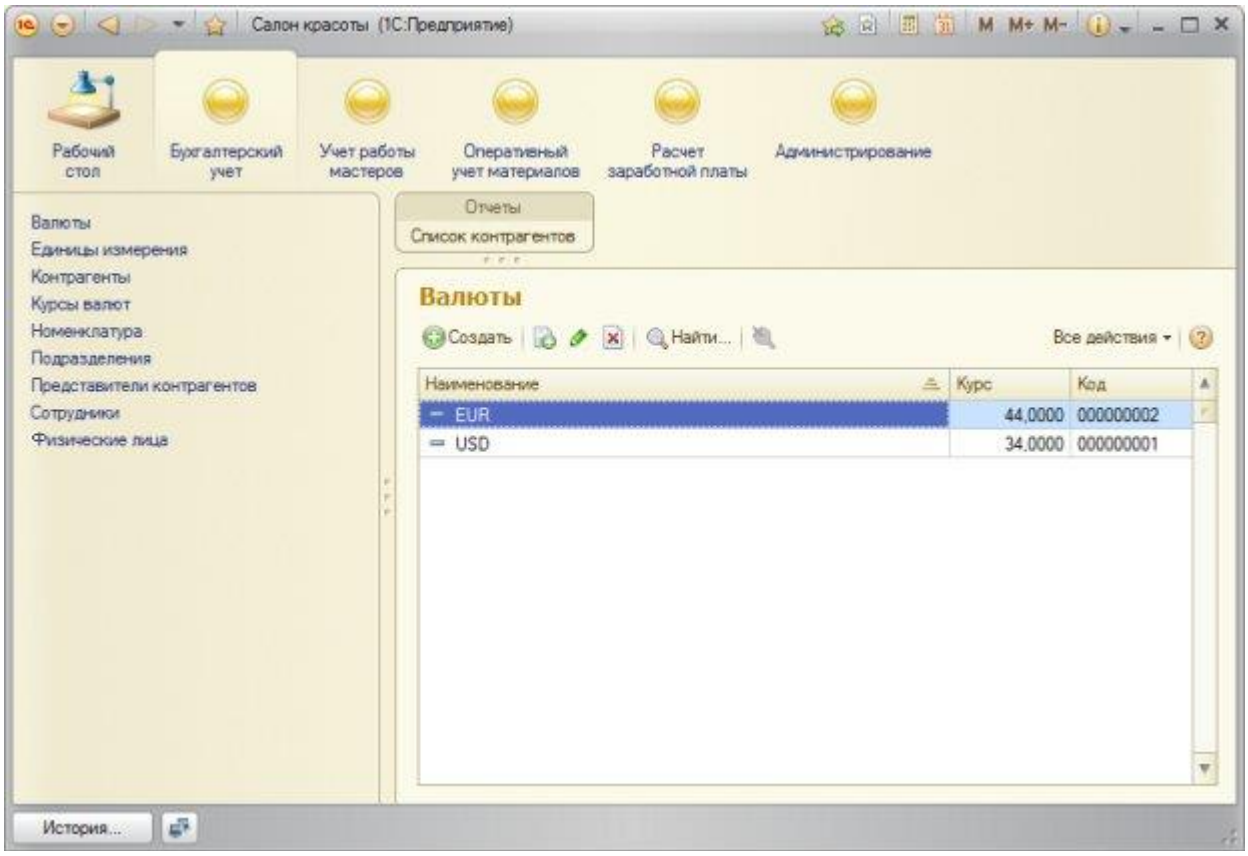


Рисунок 19.3- Измененная форма списка справочника

.

Практическая работа №20 Программная работа с документами. Функциональные опции

Цель: научиться выполнять соединение таблиц в запросе

ХОД РАБОТЫ:

1. Отчет Рейтинг услуг будет содержать информацию о том, выполнение каких услуг принесло ООО «На все руки мастер» наибольшую прибыль в указанном периоде (рисунок 20.1).

Рисунок 20.1- Результат отчета

2. 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.

3. В режиме «Конфигуратор» добавим новый объект конфигурации Отчет.

Назовем его **РейтингУслуг** и запустим конструктор схемы компоновки данных.

4. Добавим новый Набор данных – запрос и вызовем конструктор запроса.
Запрос для набора данных -Левое соединение двух таблиц

5. В качестве источника данных для запроса выберем объектную (ссылочную) таблицу **Номенклатура** и виртуальную таблицу регистра накопления **Продажи.Обороты**.

6. 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 (рисунок 20.2).

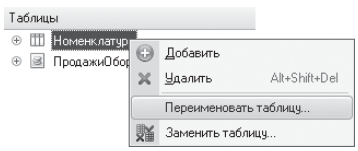


Рисунок 20.2- Переименование таблицы в запросе

7. В список полей перенесем поля СпрНоменклатура.Ссылка и ПродажиОбороты.ВыручкаОборот из этих таблиц (рисунок 20.2).

8. Так как в запросе теперь участвуют несколько таблиц, требуется определить связь между ними.

9. 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 (рисунок 20.3).

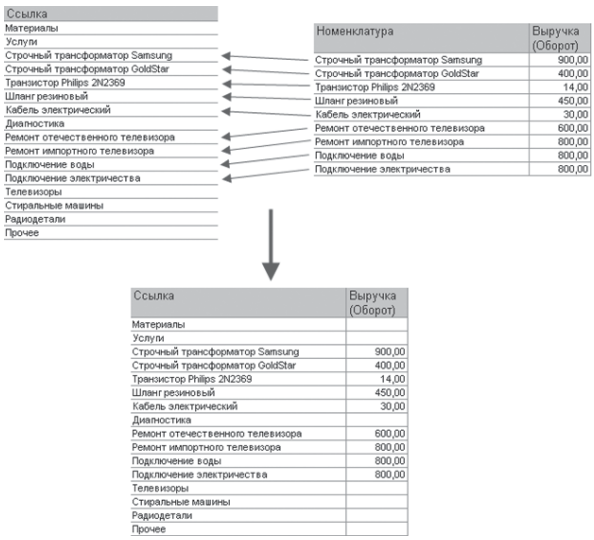


Рисунок 20.3- Связь записей таблиц в запросе

1

10. В результате описанных выше действий закладка Связи будет иметь следующий вид (рисунок 20.4).

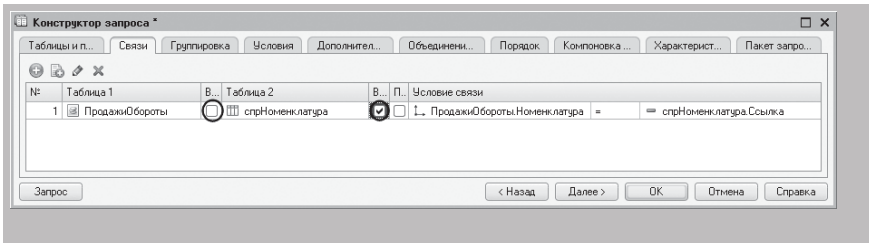


Рисунок 20.4- Определение связи между таблицами

11. Выполним анализ текста запроса

Текст запроса, сформированный платформой, примет определенный вид (рисунок 20.5).

```
ВЫБРАТЬ
    СпрНоменклатура.Ссылка
    КАК Услуга,
    ПродажиОбороты.ВыручкаОб
    орот 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
```

Рисунок 20.5- Текст запроса, сформированный алвтформой

12. Сначала, как обычно, идет часть описания запроса, и в ней есть новые для нас конструкции.

13. При описании источников запроса (после ключевого слова ИЗ) использована возможность определения нескольких источников запроса (рисунок 20.6).

```
ИЗ

    Справочник.Номенклатура КАК СпрНоменклатура
        ЛЕВОЕ СОЕДИНЕНИЕ
        РегистрНакопления.Продажи.Обороты  КАК
        ПродажиОбороты ПО
        ПродажиОбороты.Номенклатура =
        СпрНоменклатура.Ссылка
```

Рисунок 20.6- Определение нескольких источников запроса

14. 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.

Задания для самостоятельного выполнения :

1. Составьте программный модуль, осуществляющий ввод и вывод периода расчета бухгалтерских итогов, а также определяющей какая из введенных дат более ранняя, о чем выдайте сообщение пользователю. В случае отмены окна ввода выведите сообщение «Данные не введены».

Доработать модуль: изначально пользователю выдаются даты начала и конца текущего квартала**.**

2. Составьте программный модуль, осуществляющий ввод ФИО сотрудника организации, его оклада и даты приема.
   Осуществите расчет оклада +30% премии и стажа работы сотрудника на текущую дату.

3. Составьте программный модуль, осуществляющий поиск и вывод заданной пользователем Организации в одноименном справочнике и выводящей сообщение «Организация не найдена», если такой организации в справочнике нет.

4. Составить программный модуль, который запрашивает от пользователя его Имя и телефоны. Из телефонов сформируйте список значений, затем определите количество в списке обозначений «+7» и замените их на символ «8». Пользователю выдайте имя и новую нумерацию телефонов в системном окне.

Практическая работа №21 Работа с конструктором запросов

Цель: научиться применению текстовых функций

ХОД РАБОТЫ:

Наберите приведенные ниже функции в модуле обработки и просмотрите их работу.

1. Функция СтрДлина (StrLen)

Синтаксис:
СтрДлина(<Строка>)
Параметры:
<Строка> (обязательный)
Тип: Строка.
Исходная строка.
Возвращаемое значение:
Тип: Число.
Длина строки.
Описание:
Получает количество символов в строке.
Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).
Пример:

```
ДлинаНаименования = СтрДлина(Товар.Наименование);
```

2. Функция СокрЛ (TrimL)

Синтаксис:
СокрЛ(<Строка>)
Параметры:
<Строка> (обязательный)
Тип: Строка.
Исходная строка.
Возвращаемое значение:
Тип: Строка.
Строка, полученная в результате отсечения пробелов .
Описание:
Отсекает незначащие символы, стоящие слева от первого значащего символа в строке.
Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).
Примечание:
К незначащим символам относятся символы "пробел", "неразрывный пробел" (НПП), "табуляция" (горизонтальная Таб и вертикальная ВТаб), "возврат каретки" (ВК), "перевод строки" (ПС), "перевод формы (страницы)" (ПФ).
Пример:

```
НаименованиеДляПечати = СокрЛ(Товар.Наименование);
```

3. Функция СокрП (TrimR)

Синтаксис:
СокрП(<Строка>)

Параметры:
<Строка> (обязательный)
Тип: Строка.
Исходная строка.

Возвращаемое значение:
Тип: Строка.
Строка, полученная в результате отсечения пробелов .

Описание:
Отсекает незначащие символы, стоящие справа от последнего значащего символа в строке.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Примечание:
К незначащим символам относятся символы "пробел", "неразрывный пробел" (НПП), "табуляция" (горизонтальная Таб и вертикальная ВТаб), "возврат каретки" (ВК), "перевод строки" (ПС), "перевод формы (страницы)" (ПФ).

Пример:

```
НаименованиеДляПечати = СокрП(Товар.Наименование);
```

4. СокрЛП (TrimAll)

Синтаксис:
СокрЛП(<Строка>)

Параметры:
<Строка> (обязательный)
Тип: Строка.
Исходная строка.

Возвращаемое значение:
Тип: Строка.
Строка, полученная в результате отсечения пробелов .

Описание:
Отсекает незначащие символы, стоящие слева от первого значащего символа в строке, и пробелы, стоящие справа от последнего значащего символа в строке.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Примечание:
К незначащим символам относятся символы "пробел", "неразрывный пробел" (НПП), "табуляция" (горизонтальная Таб и вертикальная ВТаб), "возврат каретки" (ВК), "перевод строки" (ПС), "перевод формы (страницы)" (ПФ).

Пример:

```
НаименованиеДляПечати = СокрЛП(Товар.Наименование);
```

5. Функция Сред (Mid)

Синтаксис:
Сред(<Строка>, <НачальныйНомер>, <ЧислоСимволов>)

Параметры:

<Строка> (обязательный)
Тип: Строка.
Исходная строка.

<НачальныйНомер> (обязательный)
Тип: Число.
Начальный номер символа, с которого начинается выборка. Нумерация символов в строке начинается с 1. Если указано значение, меньшее или равное нулю, то параметр принимает значение 1.

<ЧислоСимволов> (необязательный)
Тип: Число.
Количество выбираемых символов. Если параметр не указан, то выбираются символы до конца строки.

Возвращаемое значение:
Тип: Строка.
Строка выбранных символов.

Описание:
Выбирает строку символов, начиная с символа <НачальныйНомер>, общим количеством <ЧислоСимволов>.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
// Пусть номер автомобиля в справочнике основных средств задается
// в виде x999xx99, где последние две цифры - код региона.
// Получим цифровую часть номера.
ЦифрыНомера = Сред(ОсновноеСредство.ГосНомер, 2, 3);
```

6. Функция ВРег (Upper)

Синтаксис:
ВРег(<Строка>)

Параметры:

<Строка> (обязательный)
Тип: Строка.
Исходная строка.

Возвращаемое значение:
Тип: Строка.
Строка, полученная в результате преобразования.

Описание:
Преобразует все символы строки в верхний регистр.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
ГосНомер = ВРег(ОсновноеСредство.ГосНомер);
```

7. **Функция Символ (Char)**

Синтаксис:
Символ(<КодСимвола>)

Параметры:
<КодСимвола> (обязательный)
Тип: Число.
Код получаемого символа. Код задается в соответствии с кодировкой Unicode.

Возвращаемое значение:
Тип: Строка.
Результирующий символ.

Описание:
Преобразует код символа в строку, содержащую символ.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Буква_Я = Символ(1103);
```

8. **Функция КодСимвола (CharCode)**

Синтаксис:
КодСимвола(<Строка>, <НомерСимвола>)

Параметры:
<Строка> (обязательный)
Тип: Строка.
Исходная строка.
<НомерСимвола> (необязательный)
Тип: Число.
Номер символа в строке, код которого необходимо получить. Нумерация символов в строке начинается с 1.
Значение по умолчанию: 1.

Возвращаемое значение:
Тип: Число.
Код переданного символа. Код возвращается в соответствии с кодировкой Unicode.

Описание:
Получает код символа, расположенного в переданной строке в позиции с указанным номером.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
КодПервогоСимволаФамилии = КодСимвола(Сотрудник.Фамилия);
```

Практическая работа №22 Разработка регистров накопления

Цель: научиться применению функций даты

ХОД РАБОТЫ:

Наберите приведенные ниже функции в модуле обработки и просмотрите их работу.

1. Функция Год (Year)

Синтаксис:
Год(<Дата>)

Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.

Возвращаемое значение:
Тип: Число.
Год в указанной дате.

Описание:
Определяет год в указанной дате.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Сообщить(Год("20031120") + "; Размер=" + СтрДлина(Год("20031120")));
// Результат: "2 003; Размер=5"
// Пробел в номере года вставляется при установках по умолчанию
// (подразумевается, что выбран язык "Русский" и не установлен
// признак группировки в региональных установках)
//
// Чтобы вывести номер года в виде ГГГГ, следует использовать
// метод Формат с указанием форматной строки "ЧГ=0":
Сообщить(Формат(Год("20031120"),"ЧГ=0");
// Результат: "2003"
```

2. Функция Месяц (Month)

Синтаксис:
Месяц(<Дата>)

Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.

Возвращаемое значение:
Тип: Число.
Месяц в указанной дате.

Описание:

Определяет месяц в указанной дате.

Доступность:

Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
// см. пример для метода День
```

3. Функция День (Day)

Синтаксис:

День(<Дата>)

Параметры:

<Дата> (обязательный)

Тип: Дата.

Исходная дата.

Возвращаемое значение:

Тип: Число.

День в указанной дате.

Описание:

Определяет календарный день в указанной дате.

Доступность:

Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Процедура ПолучитьСтажЗаПериод(Знач ДатаОкончанияПериода,
            Знач ДатаНачалаПериода,
            Лет=0, Месяцев=0, Дней=0) Экспорт
    Если ДатаОкончанияПериода >= ДатаНачалаПериода тогда
        Лет1 = Год(ДатаОкончанияПериода);
        Месяцев1 = Месяц(ДатаОкончанияПериода);
        Если НачалоДня(ДатаОкончанияПериода) <> НачалоДня(КонецМесяц
а(ДатаОкончанияПериода)) Тогда
            Дней1 = День(ДатаОкончанияПериода);
        Иначе
            Дней1 = 30;//В каждом месяце ровно 30 дней
        КонецЕсли;

        Лет2 = Год(ДатаНачалаПериода);
        Месяцев2 = Месяц(ДатаНачалаПериода);
        Если НачалоДня(ДатаНачалаПериода) <> НачалоДня(КонецМесяца(Д
ataНачалаПериода)) Тогда
            Дней2 = День(ДатаНачалаПериода);
        Иначе
            Дней2 = 30;//В каждом месяце ровно 30 дней
        КонецЕсли;

        Лет = Лет1 - Лет2;
        Месяцев = Месяцев1 - Месяцев2;
        Дней = Дней1 - Дней2 + 1; //1 день на увольнение
```

```
    Иначе
        Лет = 0;
        Месяцев = 0;
        Дней = 0;
    КонецЕсли;
КонецПроцедуры //ПолучитьСтажЗаПериод
```

4 Функция Час (Hour)

Синтаксис:
Час(<Дата>)

Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.

Возвращаемое значение:
Тип: Число.
Час в указанной дате.

Описание:
Определяет час в указанной дате.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Сообщить(Час(ТекущаяДата()));
```

5. Функция НачалоГода (BegOfYear)

Синтаксис:
НачалоГода(<Дата>)

Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.

Возвращаемое значение:
Тип: Дата.
Дата и время начала года.

Описание:
Определяет дату и время начала года для указанной даты.

Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Сообщить(НачалоГода(ТекущаяДата()));
```

6. Функция НачалоКвартала (BegOfQuarter)

Синтаксис:
Функция НачалоКвартала(<Дата>)

Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.
Возвращаемое значение:
Тип: Дата.
Дата и время начала квартала.
Описание:
Определяет дату и время начала квартала для указанной даты.
Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).
Пример:

```
Сообщить(НачалоКвартала(ТекущаяДата()));
```

7. Функция КонецГода (EndOfYear) // Квартала, месяц, недели, дня

Синтаксис:
КонецГода(<Дата>)
Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.
Возвращаемое значение:
Тип: Дата.
Дата и время конца года.
Описание:
Определяет дату и время конца года для указанной даты.
Доступность:
Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).
Пример:

```
Сообщить(КонецГода(ТекущаяДата()));
```

8. Функция НеделяГода (WeekOfYear)

Синтаксис:
НеделяГода(<Дата>)
Параметры:
<Дата> (обязательный)
Тип: Дата.
Исходная дата.
Возвращаемое значение:
Тип: Число.
Номер недели в году.
Описание:
Определяет номер недели в году для указанной даты.
Доступность:

Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Сообщить(НеделяГода(ТекущаяДата()));
```

9. Функция ДобавитьМесяц (AddMonth)

Синтаксис:

ДобавитьМесяц(<Дата>, <ЧислоМесяцев>)

Параметры:

<Дата> (обязательный)

Тип: Дата.

Исходная дата.

<ЧислоМесяцев> (обязательный)

Тип: Число.

Количество месяцев, которое необходимо добавить к исходной дате. Если принимает отрицательное значение, то число месяцев вычитается.

Возвращаемое значение:

Тип: Дата.

Дата, полученная в результате добавления.

Описание:

Добавляет (или вычитает) к указанной дате заданное число месяцев.

Доступность:

Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
ДатаРождения = СтрокаТаблицы.КонтактноеЛицо.ДатаРождения;
ДеньРожденияБудущегоГода = ДобавитьМесяц(
    (ДобавитьМесяц(НачалоДня(ТекущаяДата()), -(Год(ДатаРождения) -
 1) * 12)),
    (Год(ТекущаяДата()) * 12));
```

10. Функция ТекущаяДата (CurrentDate)

Синтаксис:

ТекущаяДата()

Возвращаемое значение:

Тип: Дата.

Текущая (системная) дата.

Описание:

Определяет текущую (системную) дату на компьютере.

Доступность:

Тонкий клиент, веб-клиент, сервер, толстый клиент, внешнее соединение, мобильное приложение(клиент), мобильное приложение(сервер).

Пример:

```
Если ДатаНачала = '00010101000000' Тогда
    ДатаНачала = НачалоМесяца(ТекущаяДата());
КонецЕсли;
Если ДатаОкончания = '00010101000000' Тогда
    ДатаОкончания = КонецМесяца(ТекущаяДата());
КонецЕсли;
```

Практическая работа №23 Формирование отчета с помощью СКД

Цель: научиться использованию встроенных функций языка

ХОД РАБОТЫ:

Наберите приведенные ниже функции в модуле обработки и просмотрите их работу.

Функции для вызова диалога ввода данных:

1. ВвестиЗначение (InputValue)

Синтаксис:

ВвестиЗначение(<Значение>, <Подсказка>, <Тип>)

Параметры:

<Значение> (обязательный)

Тип: Произвольный.

Имя доступной в модуле переменной. В эту переменную будет помещено введенное значение. Если параметр <Тип> не задан или имеет значение Неопределено, тип данной переменной используется в качестве типа вводимого в диалоге данного. Начальное значение переменной будет использовано в качестве начального значения в диалоге.

<Подсказка> (необязательный)

Тип: Строка.

Текст заголовка окна диалога ввода значения. Может использоваться в качестве подсказки пользователю.

Значение по умолчанию: Пустая строка.

<Тип> (необязательный)

Тип: Тип; ОписаниеТипов.

Тип вводимого значения. Если параметр не указан, используется тип параметра <Значение>.

Возвращаемое значение:

Тип: Булево.

Истина - в диалоге было введено значение; Ложь - пользователь отказался от ввода значения.

Описание:

Вызывает диалог для ввода значения заданного типа. Если тип переменной не определен и <Тип> представляет собой составной тип данных, то в поле ввода появляется кнопка выбора типа.

Доступность:

Тонкий клиент, веб-клиент, толстый клиент, мобильное приложение(клиент).

Примечание:

Если для конфигурации свойство РежимИспользованияМодальности установлено в НеИспользовать, следует использовать метод ПоказатьВводЗначения.

Пример:

```
Перем ВыбЗнач;
Массив = Новый Массив;
Массив.Добавить(Тип("Число"));
Массив.Добавить(Тип("Строка"));
```

```
Массив.Добавить(Тип("Дата"));
КЧ = Новый КвалификаторыЧисла(12,2);
КС = Новый КвалификаторыСтроки(20);
КД = Новый КвалификаторыДаты(ЧастиДаты.Дата);
ОписаниеТипов = Новый ОписаниеТипов(Массив, КЧ, КС, КД);
Если ВвестиЗначение(ВыбЗнач, "Введите значение", ОписаниеТипов) Тогда
    // обработка введенного значения
    Сообщить("Введенное значение: "+ВыбЗнач);
КонецЕсли;
```

2. ВвестиСтроку (InputString)

Синтаксис:

ВвестиСтроку(<Строка>, <Подсказка>, <Длина>, <Многострочность>)

Параметры:

<Строка> (обязательный)

Тип: Строка.

Доступная в модуле переменная. В эту переменную будет помещена введенная в диалоге строка. Начальное значение переменной будет использовано в качестве начального значения в диалоге.

<Подсказка> (необязательный)

Тип: Строка.

Текст заголовка окна диалога ввода строки. Может использоваться в качестве подсказки пользователю.

Значение по умолчанию: Пустая строка.

<Длина> (необязательный)

Тип: Число.

Длина вводимой строки. Если параметр не указан, то строка неограниченной длины.

Значение по умолчанию: 0.

<Многострочность> (необязательный)

Тип: Булево.

Определяет режим ввода многострочного текста: Истина - ввод многострочного текста с разделителями строк; Ложь - ввод простой строки.

Значение по умолчанию: Ложь.

Возвращаемое значение:

Тип: Булево.

Истина - строка введена; Ложь - пользователь отказался от ввода строки.

Описание:

Вызывает диалог для ввода строки.

Доступность:

Тонкий клиент, веб-клиент, толстый клиент, мобильное приложение(клиент).

Примечание:

Если для конфигурации свойство РежимИспользованияМодальности установлено в НеИспользовать, следует использовать метод ПоказатьВводСтроки.

Пример:

```
Текст = "";
Подсказка = "Введите текст напоминания";
Если ВвестиСтроку(Текст, Подсказка, 0, Истина) Тогда
```

```
    // запомнить текст напоминания
КонецЕсли;
```

3. ВвестиЧисло (InputNumber)

Синтаксис:

ВвестиЧисло(<Число>, <Подсказка>, <Длина>, <Точность>)

Параметры:

<Число> (обязательный)

Тип: Число.

Имя доступной в модуле переменной. В эту переменную будет помещено введенное число. Начальное значение переменной будет использовано в качестве начального значения в диалоге.

<Подсказка> (необязательный)

Тип: Строка.

Текст заголовка окна диалога ввода числа. Может использоваться в качестве подсказки пользователю.

Значение по умолчанию: Пустая строка.

<Длина> (необязательный)

Тип: Число.

Длина вводимого числа включая дробную часть (символы разделителей не учитываются).

Значение по умолчанию: 0.

<Точность> (необязательный)

Тип: Число.

Количество знаков в дробной части вводимого числа.

Значение по умолчанию: 0.

Возвращаемое значение:

Тип: Булево.

Истина - число введено; Ложь - пользователь отказался от ввода.

Описание:

Вызывает диалог для ввода числа.

Доступность:

Тонкий клиент, веб-клиент, толстый клиент, мобильное приложение(клиент).

Примечание:

Если для конфигурации свойство РежимИспользованияМодальности установлено в НеИспользовать, следует использовать метод ПоказатьВводЧисла.

Пример:

```
Количество = 1;
Если ВвестиЧисло(Количество, "Введите количество", 10, 2) Тогда
    // обработка введенного количества
КонецЕсли;
```

4. ВвестиДату (InputDate)

Синтаксис:

ВвестиДату(<Дата>, <Подсказка>, <ЧастьДаты>)

Параметры:

<Дата> (обязательный)

Тип: Дата.
Имя доступной в модуле переменной. В эту переменную будет помещено введенное значение даты. Начальное значение переменной будет использовано в качестве начального значения в диалоге.
<Подсказка> (необязательный)
Тип: Строка.
Текст заголовка окна диалога ввода даты. Может использоваться в качестве подсказки пользователю.
Значение по умолчанию: Пустая строка.
<ЧастьДаты> (необязательный)
Тип: ЧастиДаты.
Вводимая в диалоге часть (или части) даты.
Значение по умолчанию: ДатаВремя.
Возвращаемое значение:
Тип: Булево.
Истина - дата введена; Ложь - пользователь отказался от ввода даты.
Описание:
Вызывает диалог для ввода даты.
Доступность:
Тонкий клиент, веб-клиент, толстый клиент, мобильное приложение(клиент).
Примечание:
Если для конфигурации свойство РежимИспользованияМодальности установлено в НеИспользовать, следует использовать метод ПоказатьВводДаты.
Пример:

```
ДатаНапоминания = РабочаяДата;
Подсказка = "Введите дату и время";
ЧастьДаты = ЧастиДаты.ДатаВремя;
Если ВвестиДату(ДатаНапоминания, Подсказка, ЧастьДаты) Тогда
    // запомнить дату напоминания
КонецЕсли;
```

Практическая работа №24 Разработка средств ведения бухгалтерского учета

Цель: научиться созданию процедур приходно-расходных операций

ХОД РАБОТЫ:

I. Конфигуратор. Создание печатной формы справочника Курсы.

1) Откройте справочник Курсы (ДЩ на названии в дереве);
2) Откройте закладку Макеты, выберите кнопку внизу Конструкторы→ Конструкторы печати.
3) Включите режим создания процедуры: в модуле формы→кн. Далее
4) Выберите реквизиты шапки: все, кроме поля Код →кн. Далее
5) Включите режим: новая кнопка: Печать→кн. ОК
6) Появится макет печатной формы

7) Просмотрите модуль печатной формы: закладка Формы→ ФормаСписка→ закл внизу Модуль

II. Отладка. Формирование печатной формы справочника.

1) П.м. Операции→ Справочник→ Курсы→ кн. Печать
2) Если ширина столбцов требует корректировки, откорректируйте ее в конфигураторе.

III. Конфигуратор. Внесение изменений в печатную форму справочника Курсы (выбор коротких курсов- продолжительностью 2 недели).

1) Откройте макет Печать справочника Курсы.
2) Уберите значение часов и минут в формате поля ДатаНачала: ЩП→Свойства→Значения→Формат→ Дата→Формат даты: dd.MM.yyyy
3) Создайте флажок отбора на форме: закладка Формы→ ФормаСписка, выберите меню: Форма→Вставить элемент управления→ Флажок
4) Введите имя флажка: КороткиеКурсы → ОК
5) Закладка Данные: Реквизиты- Продолжительность установите режим Индексировать.
6) Откройте модуль печатной формы: закладка Формы→ ФормаСписка→ закл внизу Модуль
7) После строки **ТабДок.Вывести(Область)**; вместо строки **Выборка = Справочники.Курсы.Выбрать();**
8) Добавьте операторы:

```
Если (ЭлементыФормы.КороткиеКурсы.Значение=Истина) Тогда
        СтруктураОтбора = Новый Структура;
        СтруктураОтбора.Вставить("Продолжительность", 2);
        Выборка = Справочники.Курсы.Выбрать(,,СтруктураОтбора);
Иначе
        Выборка = Справочники.Курсы.Выбрать();
КонецЕсли;
```

IV. Отладка. Проверка внесенных изменений.

1) П.м. Операции→ Справочник→ Курсы
2) Включите флажок «Коротки курсы», затем кн. Печать
3) Должны быть выведены только курсы с продолжительностью 2.

V. Конфигуратор. Создадим еще один справочник – Клиенты

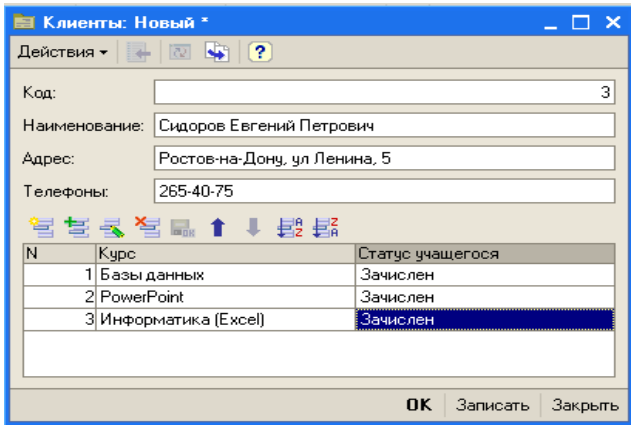
1) Выбрать на дереве объект Справочники, Нажать кл Insert
2) Закладка Основные- ввести имя: Клиенты.
3) Закладка Данные: Реквизиты- Адрес, Телефоны (Тип данных – строка длинной не более 50).

Табличная часть: ИзучаемыеКурсы, Реквизиты внутри табличной части: Курс (Тип данных – СправочникСсылка.Курсы), СтатусУчащегося (Тип данных – ПеречислениеСсылка.Статус).

4) Закладка Формы: Во фрейме Редактирование выбрать режим «в диалоге»
5) Закладка Формы: в строке Основные формы .. **Элемента** щелкнуть по кнопке 
6) Все реквизиты оставить по умолчанию – кн Далее
7) Снова подтвердить все реквизиты по умолчанию – кн Готово

VI. Отладка. Заполнение справочника Клиенты через форму.

1) П.м. Операции→ Справочник→ Клиенты→Insert
2) Заполните данные о **трех** клиентах, например:



VII. Конфигуратор. Создание печатного варианта элемента формы справочника Клиенты.

1) Откройте справочник Клиенты;
2) Откройте закладку Макеты, выберите кнопку внизу Конструкторы→ Конструкторы печати.
3) Включите режим создания процедуры: в модуле формы: ФормаЭлемента →кн. Далее
4) Выберите реквизиты шапки: Наименование, Адрес, Телефоны→кн. Далее
5) Выберите реквизиты табличной части: Курс, СтатусУчащегося→кн. Далее
6) Реквизиты подвала не указываем→кн. Далее
7) Новая кнопка: Печать→ ОК
8) Появится макет печатной формы.
9) Увеличьте ширину столбцов для значений полей <Наименование> и Курс, чтобы названия были хорошо видны.
10) Просмотрите модуль печатной формы: закладка Формы→ ФормаЭлемента→ закл внизу Модуль
11) Обратите внимание на новую конструкцию цикла «Для Каждого». Цикл продолжается до окончания перебора всех элементов.

VIII. Отладка. Формирование печатной формы для элемента справочника Клиенты.

1) П.м. Операции→ Справочник→ Клиенты
2) Откройте клиента Сидорова Е.П.
3) Щелкните на кн. Печать
4) Просмотрите печатную форму элемента.

Практическая работа №25 Разработка средств ведения бухгалтерского учета

Цель: научиться разработке в конфигурации плана счетов бухгалтерского учета

ХОД РАБОТЫ:

1. Для реализации подсистемы бухгалтерского учета нам понадобятся следующие объекты 1С:Предприятие 8:

1. *План видов характеристик*. Его мы будем использовать для хранения видов аналитики (*субконто*), которые должны присутствовать у наших счетов.

2. План счетов. Это основа бухгалтерской подсистемы. План счетов хранит описания счетов, на которых будет вестись учет. В конфигурациях может присутствовать неограниченное количество планов счетов, однако, обычно количество планов счетов в одной конфигурации не превышает 1-2. План счетов можно сравнить со справочником особого назначения, который предназначен для хранения информации о счетах бухгалтерского учета.
3. *Регистр бухгалтерии*. Он связан с планом счетов и применяется для хранения бухгалтерских записей. *Регистр бухгалтерии* можно сравнить с журналом, в котором ведутся бухгалтерские записи.

Создавая бухгалтерскую подсистему конфигурации, сначала создадим *план видов характеристик* – на него нужно будет сослаться при создании плана счетов, затем – *план счетов* – без указания плана счетов мы не сможем создать *регистр бухгалтерии*.

2. Создадим новый *план видов характеристик*, назовем его **ВидыСубконто**, рис. 1.1

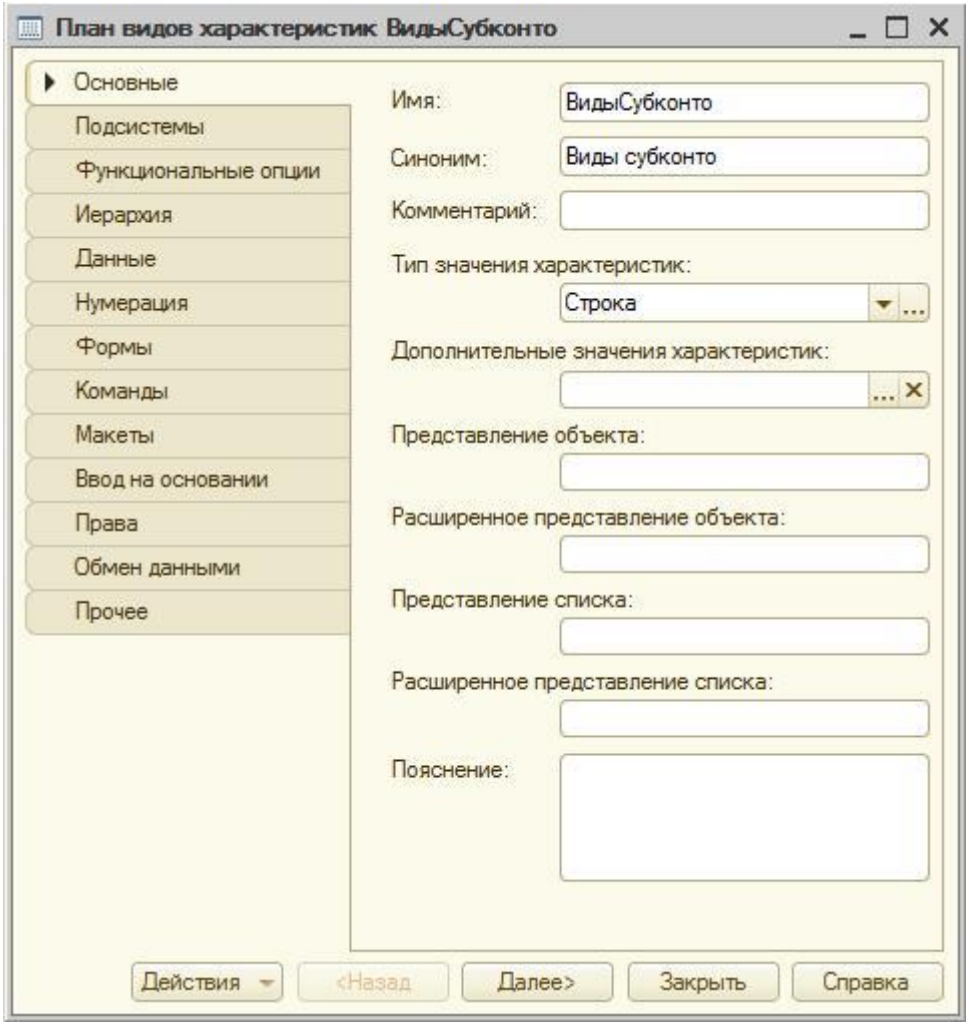


Рисунок 25.1. План видов характеристик ВидыСубконто

План видов характеристик добавляет в систему новый *тип данных*, который, *по* сути, является составным типом данных. В этот составной *тип данных* входят обычно справочники, элементы

которых, в итоге, используются в аналитическом учете. Значения характеристик могут поставлять не только справочники – кроме того, это могут, например, документы и перечисления.

Добавим созданный *план видов характеристик* в состав подсистемы **БухгалтерскийУчет**.

При настройке *плана видов характеристик* особую важность имеют его свойства **Тип значения характеристик** и **Дополнительные значения характеристик**. Именно они определяют набор типов данных, объединенных планом видов характеристик.

Для правильной настройки этих свойств, прежде чем продолжать, создадим новый справочник – назовем его **Субконто**.

Добавим справочник в состав подсистемы **БухгалтерскийУчет**.

4. Выберем, на вкладке **Владельцы** окна настройки справочника, *план видов характеристик* **ВидыСубконто** в качестве владельца, установим *параметр* **Использование подчинения** в *значение* **Элементам**, рисунок 25.2.

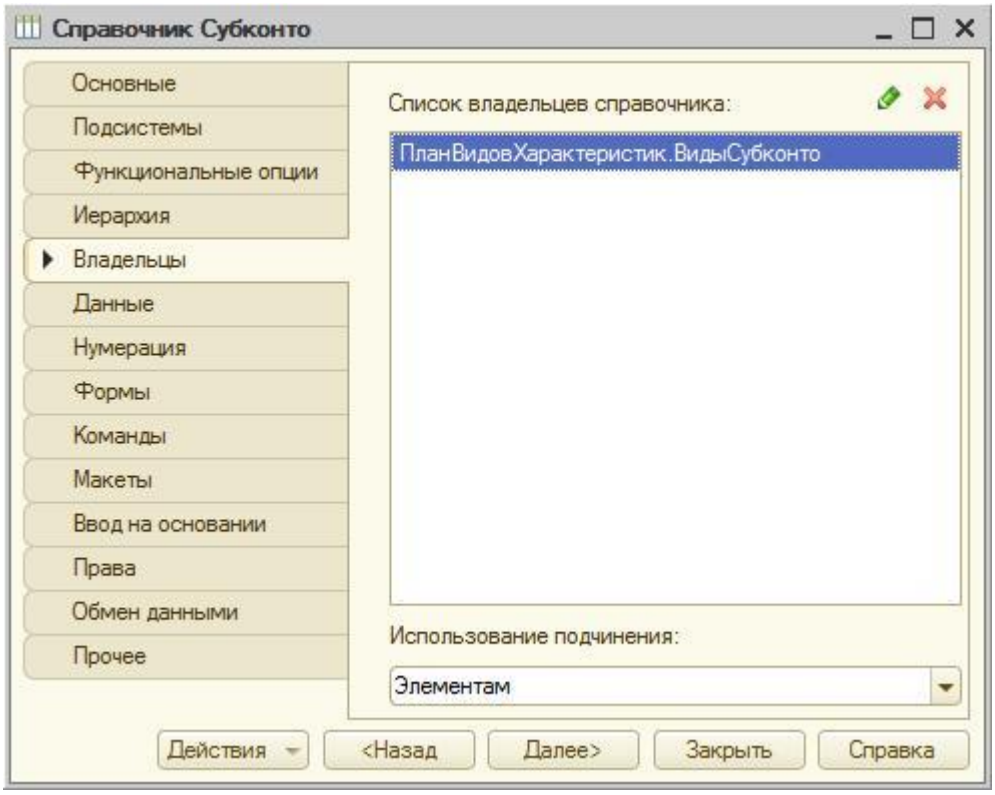


Рисунок 25.2. Справочник Субконто

Мы не будем настраивать другие значения справочника, хотя, при необходимости, это можно сделать. Он нужен нам для того, чтобы не ограничивать пользователя конфигурации значениями *субконто*, которые он может задать, пользуясь существующими справочниками, указанными в плане видов характеристик. Фактически, это позволит пользователю самостоятельно задавать необходимые ему *аналитические разрезы*, не прибегая к конфигурированию системы и настройке *плана видов характеристик*.

Перейдем в *план видов характеристик*, на закладке **Основные** откроем его свойство **Тип значения характеристик**, рисунок 25.3.

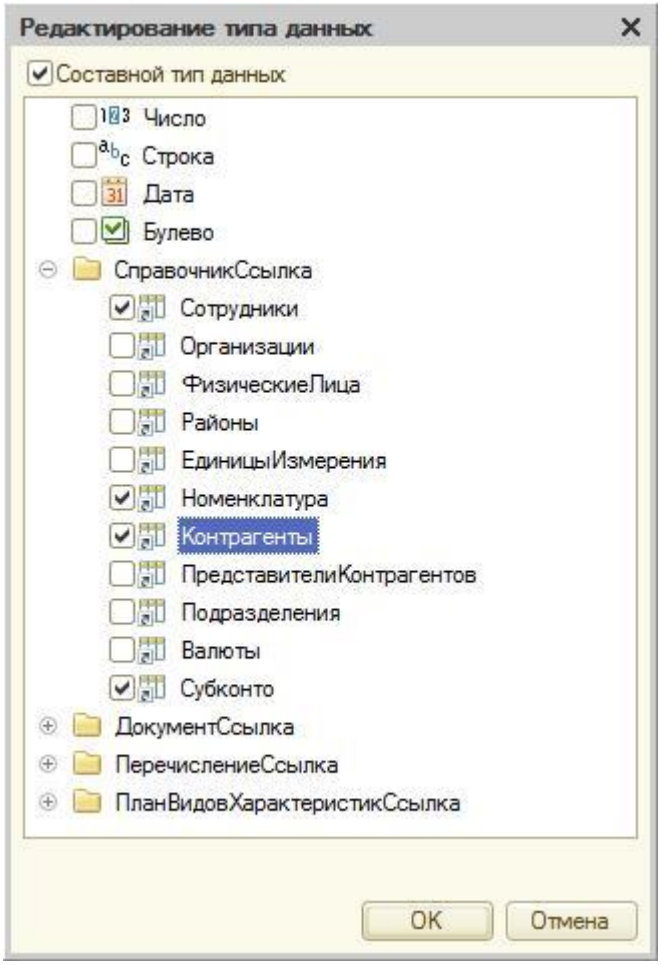


Рисунок 25.3. Настройка типа значения характеристик

4. Установим флажок **Составной тип данных**, снимем флажок **Строка** (не рекомендуется использовать в планах видов характеристик *простые типы данных*), установим флажки напротив тех справочников, которые содержат нужные виды аналитики – **Контрагенты, Номенклатура, Сотрудники**. Отметим, так же, справочник **Субконто**.

5. Откроем свойство *плана видов характеристик* **Дополнительные значения характеристик**. Здесь нужно выбрать справочник, подчиненный *плану видов характеристик*. В нашем случае, рисунок 25.4, это лишь один справочник –**Субконто**.

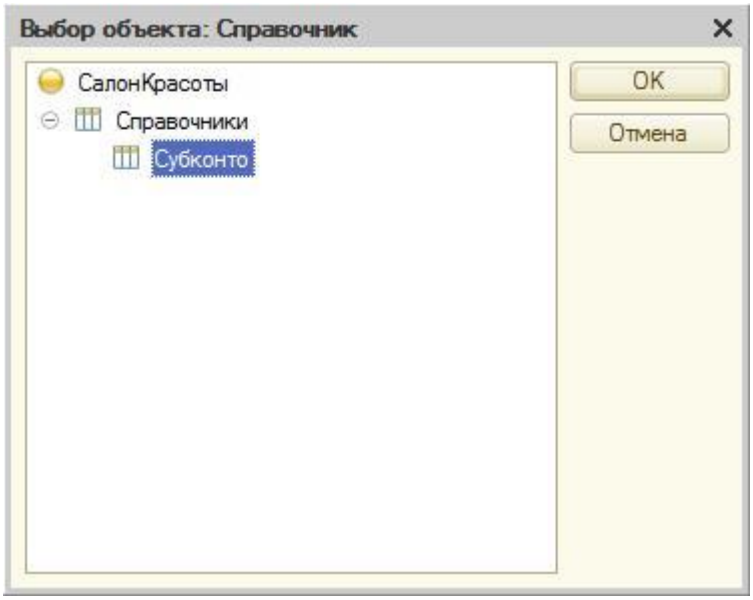


Рисунок 25.4. Настройка дополнительных значений характеристик.

Теперь откроем предопределенные элементы *плана видов характеристик* (вкладка **Прочее**, кнопка **Предопределенные**) и создадим следующие предопределенные значения, таблица 1.2.

Таблица 1.2. Предопределенные элементы*плана видов характеристик* ВидыСубконто

Имя	**Тип**
Номенклатура	СправочникСсылка.Номенклатура
Контрагенты	СправочникСсылка.Контрагенты
Сотрудники	СправочникСсылка.Сотрудники

На рисунок 25.5 вы может видеть *список* созданных предопределенных элементов и окно свойств одного из предопределенных элементов.

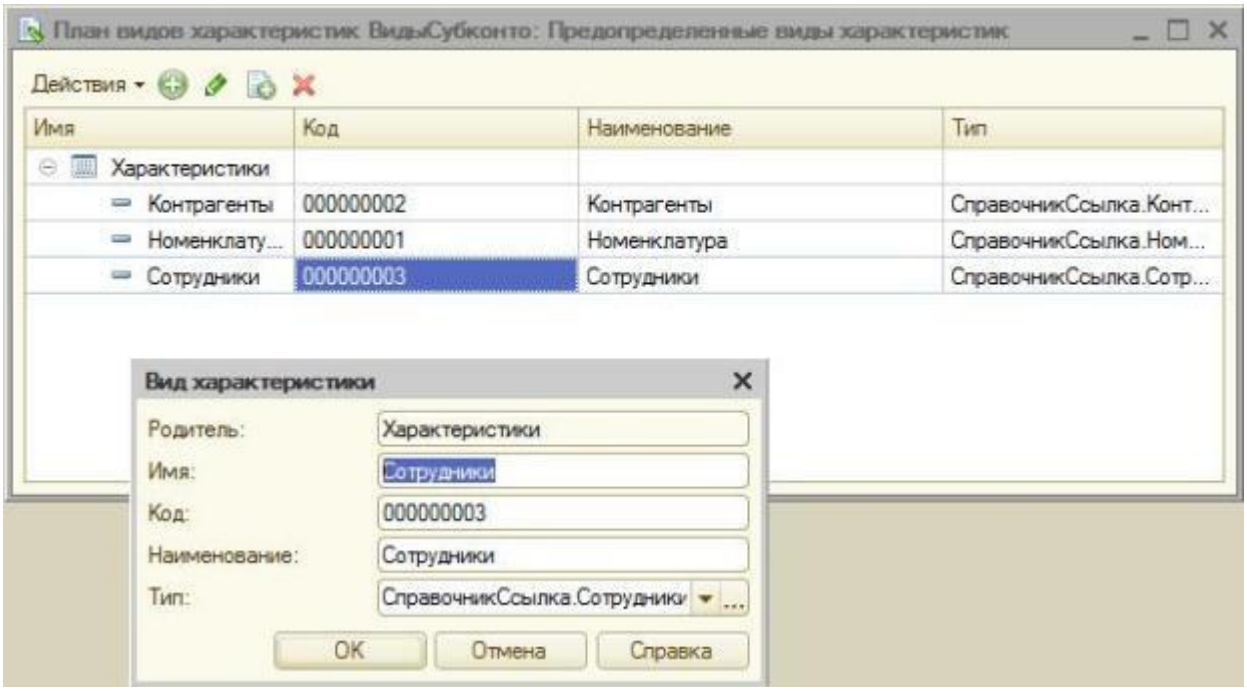


Рисунок 25.5. Настройка предопределенных значений характеристик

6. Теперь приступим к созданию плана счетов.

План счетов

Создадим новый *план счетов*, назовем его **Хозрасчетный**. Добавим *план счетов* в подсистему **БухгалтерскийУчет**. Перейдем на вкладку **Данные**, зададим следующие параметры (рис. 1.6):

Длина кода: 5
Длина наименования: 50
Маска кода: @@.@@
Автопорядок по коду: установлено
Длина порядка: 5
Основное представление: В виде наименования
Добавим признаки учета Количественный и Валютный

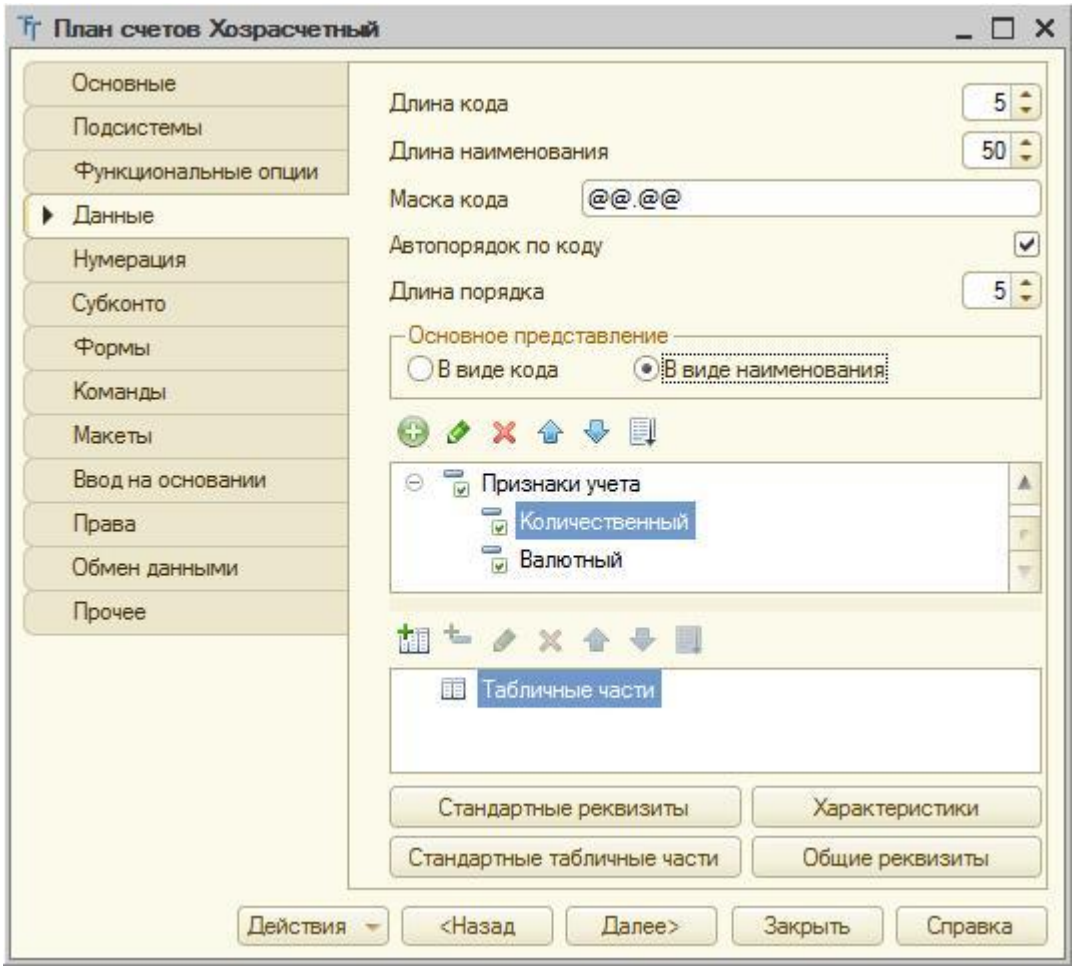


Рисунок 25.6. Настройка плана счетов

Код счета – это цифровой код счета, принятый в учетной практике. **Наименование** – это буквенное наименование счета.

Маска кода счета задает структуру кода счета и возможность использования в коде различных символов. В нашем случае *маска* имеет вид @@.@@. Символ "@" подразумевает использование в коде счета любых алфавитно-цифровых символов. *План счетов* с такой маской предусматривает два уровня вложенности счетов, и *по* два символа в кодах счета и субсчета (например, 10.01 и т.д.). Существуют и другие специальные символы, с помощью которых можно задавать маски кодов счетов. Среди них следующие:

? – подразумевает использование вместо себя любой цифры.

! – подразумевает преобразование любого введенного символа в верхний *регистр*.

- подразумевает ввод любой цифры, пробела, знаков "+" и "-"

N – подразумевает любые буквенные и цифровые символы

U – подразумевает любые буквенные и цифровые символы, которые преобразуются в верхний *регистр*.

X – подразумевает произвольный символ.

Свойство **Автопорядок по коду** оказывает комплексное воздействие на сортировку счетов, в частности, оно позволяет в правильном, с точки зрения пользователя, порядке, формировать отсортированную последовательность субсчетов счета.

Поле **Порядок** не может быть меньше длины кода счета, задаваемой маской кода. Именно *по* этому полю, которое добавляется в *план счетов*, и осуществляется *сортировка* элементов. Ключевая разница между, например, счетом 01.01 и порядком этого счета, заключается в том, что в порядке счета нули заменяются пробелами, что позволяет корректно сортировать счета.

В качестве основного представления счета мы выберем **Наименование**.

Признак учета **Количественный** позволяет вести на счете количественный учет. Это актуально, например, для учета материалов. Сам *по* себе признак "работать" не будет – позже мы используем его при настройке счетов, при проектировании *регистра бухгалтерии*, а еще позже – будем ориентироваться на него при формировании движений *по* счетам. Название признака роли не играет – роль играет его использование в системе.

Признаки учета *субконто* после связи их с ресурсами *регистра бухгалтерии* позволяют управлять хранением остатков и оборотов *по* этим ресурсам для используемых на нем *субконто*. В нашем случае нужно, чтобы учет товарно-материальных ценностей на счете велся в разрезе ответственных лиц лишь в количественном показателе, а в целом *по* предприятию, в разрезе номенклатурных позиций – и в количественном и в суммовом варианте. Это позволит нам, во-первых – правильно формировать себестоимость ценностей в целом *по* организации, а, во-вторых – технически верно списывать материальные ценности с ответственных лиц.

Признак учета **Валютный** позволит нам вести на счете валютный учет, то есть – отражать на счете показатели не только в валюте баланса, но и в других валютах. В случае с этим признаком, так же, важно его последующее использование, а не название или факт наличия в плане счетов.

7.Перейдем на вкладку **Субконто**, в *поле* **Виды субконто** выберем *план видов характеристик*, содержащий виды *субконто* – **ВидыСубконто** (именно сейчас мы связали ранее созданный *план видов характеристик* с планом счетов), создадим признак учета *субконто* **Суммовой**, рис. 1.7.

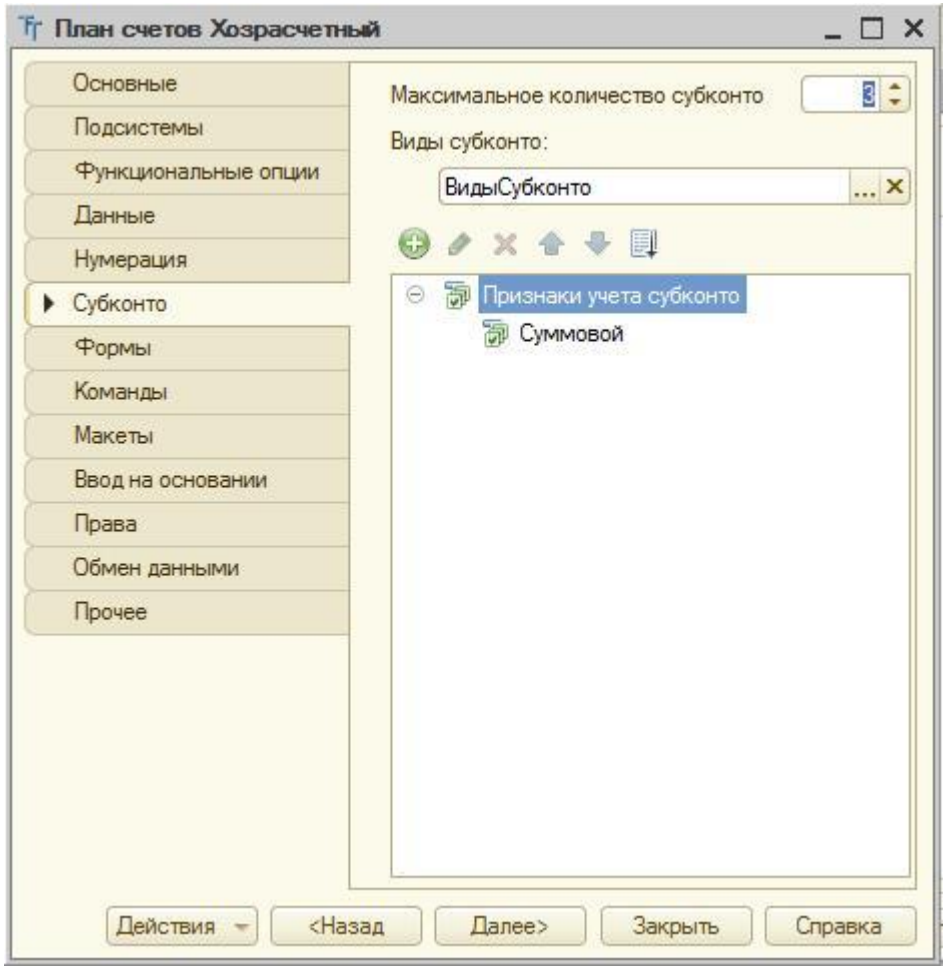


Рисунок 25.7. Настройка плана счетов, виды субконто

Признак учета **Суммовой** позволит включать и отключать суммовой учет *по субконто*.

8.Перейдем на вкладку **Прочее**, нажмем на кнопку **Предопределенные** и введем сведения о предопределенных счетах, состав которых мы обсуждали выше. В таблица 1.3 приведены свойства счетов, которые нужно задать. Если признак учета не упомянут в колонке **Установленные признаки учета** или в колонке **Субконто, признаки учета субконто** – этот признак устанавливать не нужно. Имена счетов можно получить из приведенных наименований *по* правилам именования объектов в 1С:Предприятие 8 (то есть, наименование **Касса в валюте** преобразуется в имя **КассаВВалюте**).

Таблица 1.3. Настройка предопределенных счетов

Родитель	**Наименование**	**Код**	**Вид**	**Установленные признаки учета**	***Субконто*, признаки учета *субконто***
Счета	Материалы	10	Активный		

Материалы	Материалы для производства	10.1	Активный	Количественный	Номенклатура, суммовой установлен; Сотрудники, суммовой установлен
Материалы	Прочие материалы	10.11	Активный	Количественный	Номенклатура, суммовой установлен; Сотрудники
Счета	Касса	50	Активный		
Касса	Касса в рублях	50.1	Активный		
Касса	Касса в валюте	50.2	Активный	Валютный	
Счета	Поставщики	60	Пассивный		
Поставщики	Расчеты с поставщиками в рублях	60.1	Пассивный		Контрагенты, суммовой установлен
Поставщики	Расчеты с поставщиками в валюте	60.2	Пассивный	Валютный	Контрагенты, суммовой установлен
Счета	Покупатели	62			
Покупатели	Расчеты с покупателями в рублях	62.1	Активный		Контрагенты, суммовой установлен
Покупатели	Расчеты с покупателями в валюте	62.2	Активный	Валютный	Контрагенты, суммовой установлен
Счета	Персонал	70	Пассивный		Сотрудники, суммовой установлен
Счета	Капитал	80	Пассивный		

На рисунок 25.8 вы можете видеть окно с предопределенными счетами.

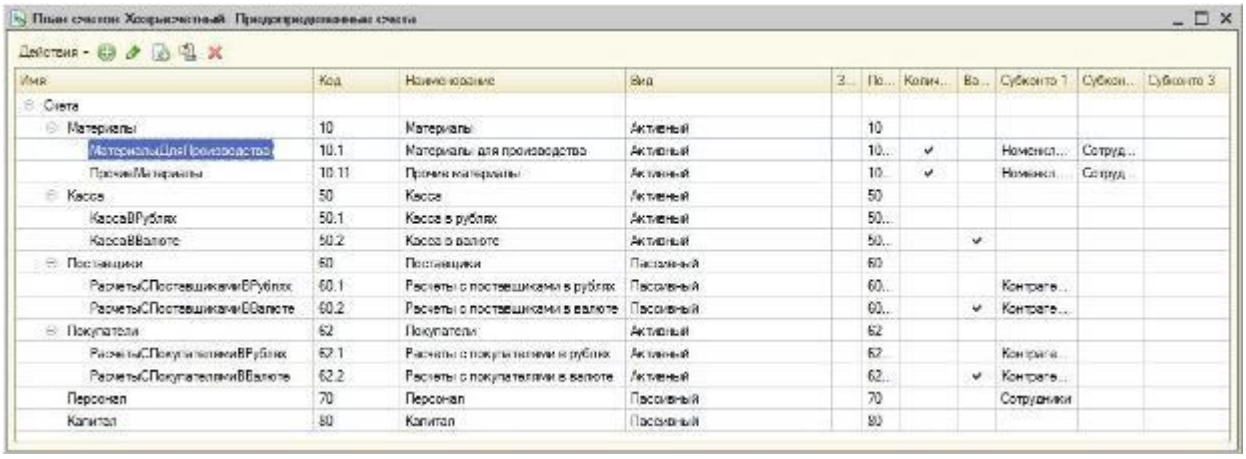


Рисунок 25.8. Предопределенные счета в плане счетов

На рисунке 25.9 вы можете видеть окно настройки одного из предопределенных счетов.

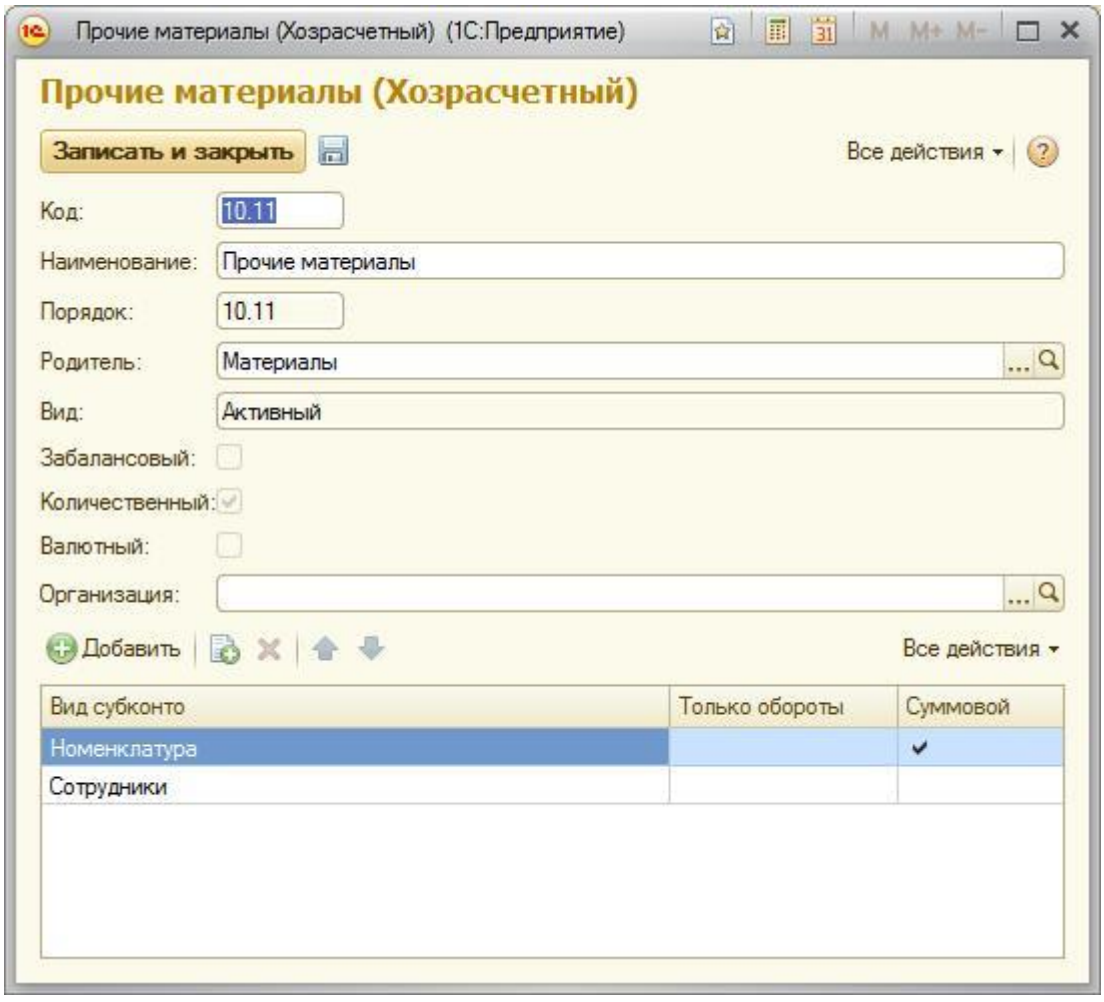


Рисунок 25.9. Предопределенный элемент плана счетов

9.Для создания субсчетов к счету нужно выделить в списке предопределенных счетов нужный счет и создать новый *субсчет*. Если *субсчет* случайно создан не для того счета, или, например, новый счет случайно создан как*субсчет* одного из существующих счетов, его можно

переместить в подчинение другого счета (или сделать самостоятельным), воспользовавшись кнопкой панели инструментов **Переместить элемент в другую группу**.

В *завершение работы* над планом счетов создадим форму счета, при создании формы отметим, в *дополнение* к уже отмеченным реквизитам, *реквизит* **Порядок**, рис. 1.10.

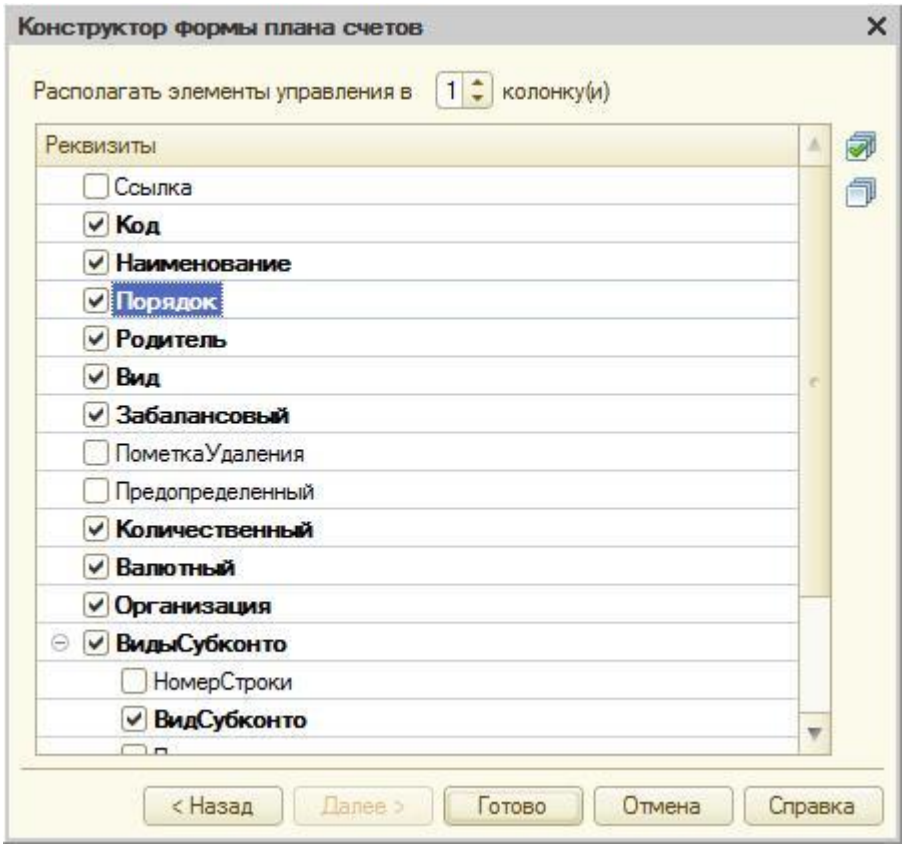


Рисунок 25.10. Включение отображения реквизита Порядок на форме счета

По умолчанию *реквизит* **Порядок** не входит в число реквизитов, которые в конструкторе формы плана счетов предлагается разместить на форме счета. Заполнить *поле* **Порядок** можно как вручную, так и программно. Мы, зная правила формирования поля **Порядок**, можем сделать это и вручную, но *поле*

Порядок не случайно не выводится на форму *по* умолчанию – желательно реализовать его заполнение автоматически, например, перед записью нового счета, который создал *пользователь*.

Сделать это можно, воспользовавшись процедурой ПередЗаписью() в модуле объекта. У объекта типа ПланСчетовОбъект существует метод ПолучитьПорядокКода(). Этот метод, в том случае, если при указании маски счета использованы символы "@" и ".", позволяет получить порядок кода, построенный в соответствии с этой маской. Метод возвращает порядок в виде строки, эту строку можно присвоить реквизиту **Порядок**.

10.Для автоматического формирования порядка мы можем воспользоваться таким кодом в модуле объекта плана счетов **Хозрасчетный**:

Процедура ПередЗаписью(Отказ)

```
  Порядок=ПолучитьПорядокКода();
КонецПроцедуры
```

11. Реализовав автоматическое формирование порядка, запретим изменение поля **Порядок** пользователем, но оставим его на форме и снабдим подсказкой. Для этого в редакторе форм вызовем палитру свойств для поля **Порядок**, установим флаг **ТолькоПросмотр** и в свойство **Подсказка** введем текст: **"Порядок будет сформирован автоматически перед записью счета"**. Вот как, в итоге, будет выглядеть форма счета в пользовательском режиме (рисунок 25.11).

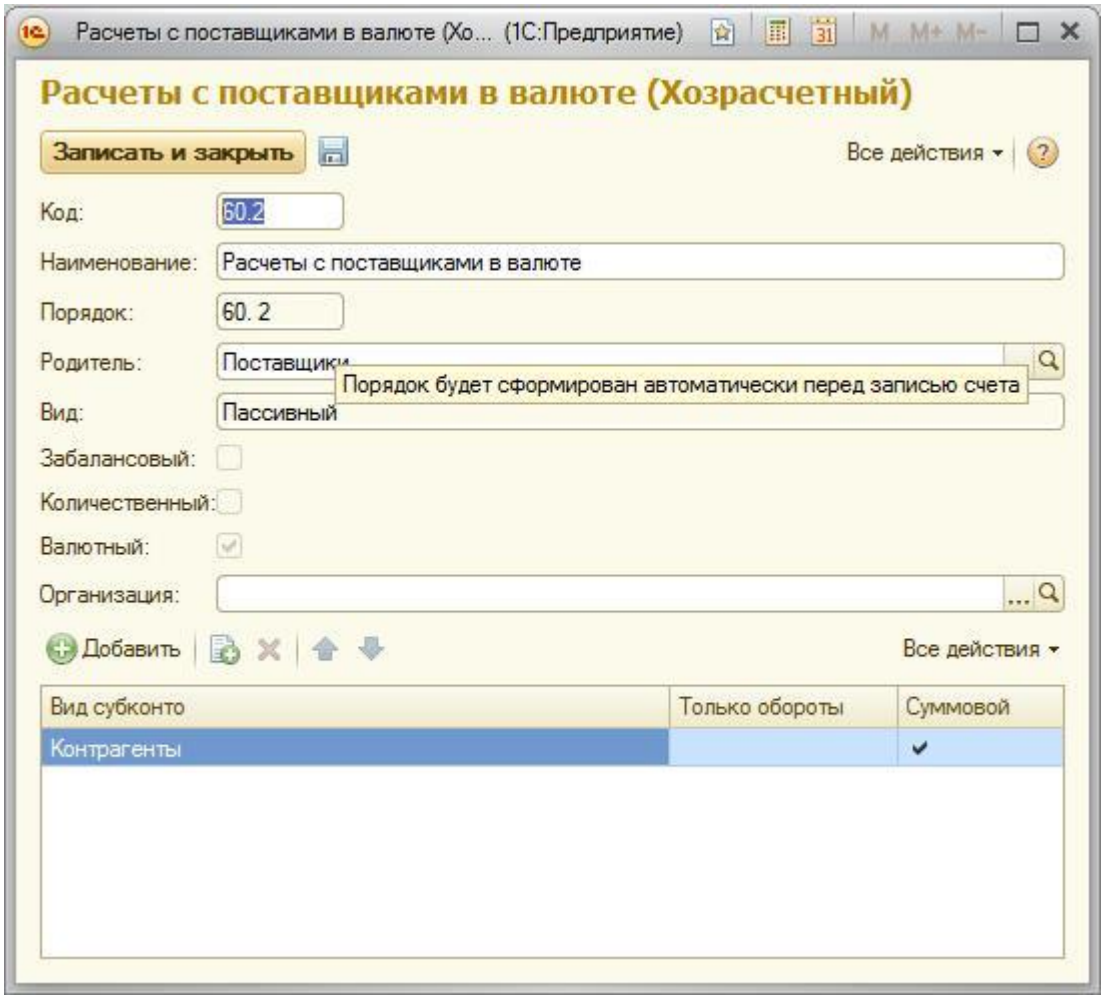


Рисунок 25.11. Подсказка, выводимая при наведении указателя мыши на поле Порядок

После того, как *план счетов* настроен, мы можем переходить к созданию *регистра бухгалтерии*.

12. Создадим *регистр бухгалтерии* **Хозрасчетный**. В качестве плана счетов на вкладке **Основные**, рисунок 25.12, укажем ранее созданный *план счетов* **Хозрасчетный**. Установим флаг **Корреспонденция** – это будет означать, что*регистр бухгалтерии* поддерживает корреспонденцию счетов, то есть – ведение учета методом двойной записи.

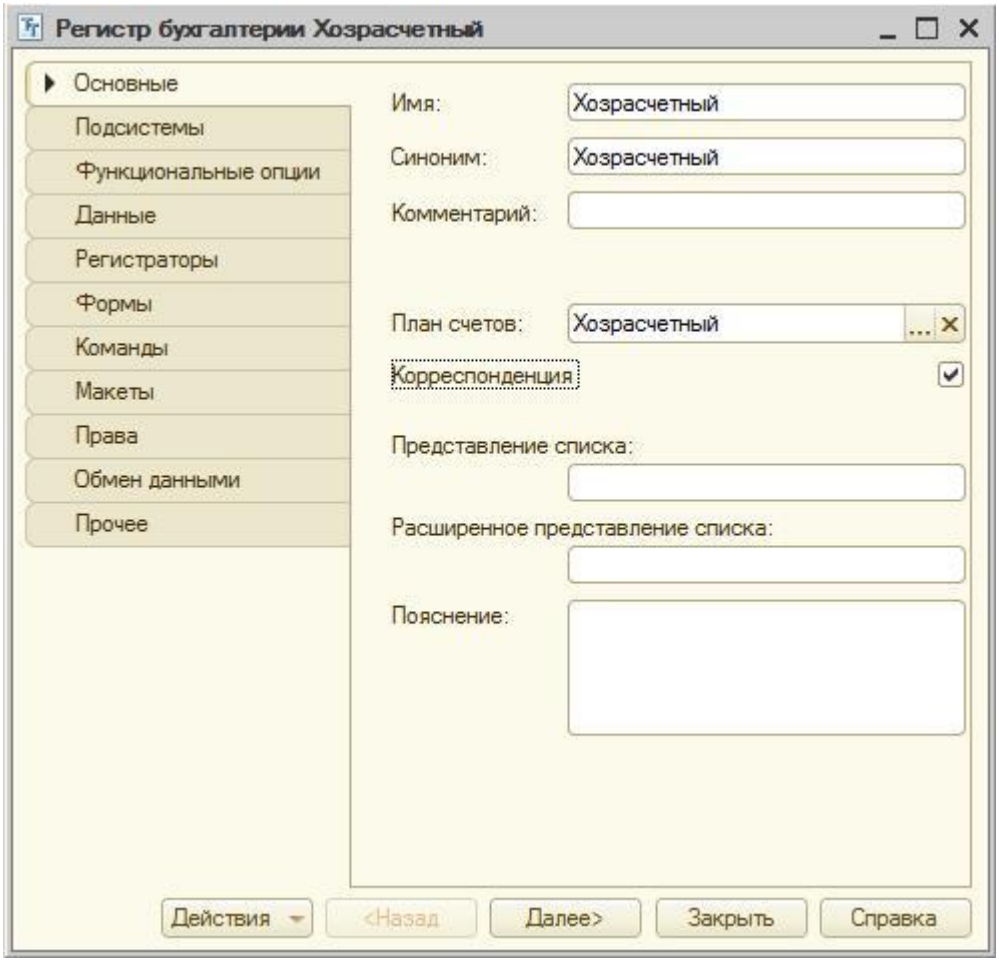


Рисунок 25.12. Настройка регистра бухгалтерии Хозрасчетный

13. Включим *регистр бухгалтерии* в состав подсистемы **БухгалтерскийУчет**. Исключим *регистр* из состава общего реквизита **Организация**.

Самый простой *регистр бухгалтерии* может содержать лишь один *ресурс*, который содержит суммовой показатель в валюте учета. В нашем же случае предполагается хранение в регистре не только суммовых показателей, но и количественных, и валютных показателей в валюте, это требует соответствующей настройки регистра.

14.Перейдем на вкладку **Данные** *регистра бухгалтерии*.

Создадим измерение Валюта.
Тип: СправочникСсылка.Валюты.
Признак учета: Валютный.
Признак Балансовый не установлен.
Флаг Запрет незаполненных значений установлен.

Благодаря наличию данного измерения мы сможем организовать в регистре бухгалтерии валютный учет на счетах с установленным признаком учета **Валютный**. То, что признак **Балансовый** не установлен, позволит нам, при формировании бухгалтерских записей, указывать валютную сумму лишь для того счета из двух корреспондирующих, на котором нужно ведение валютного учета. Например, при поступлении материалов от поставщика, взаиморасчеты с которым ведутся в валюте, на счет учета материалов попадут данные в

рублевой оценке, а на счет учета расчетов с поставщиками мы поместим сведения и в валюте регламентированного учета, и в валюте взаиморасчетов с контрагентом. Установка флага **Запрет незаполненных значений** позволяет нам контролировать заполнение данного измерения в тех случаях, когда на счете, который участвует в проводке, ведется валютный учет.

15. Создадим следующие ресурсы: (рисунок 25.13)

Сумма.
Тип – число,
длина – 10,
точность – 2.
Признак Балансовый установлен.
Признак учета субконто – Суммовой.

Количество.
Тип – число,
длина –10,
точность – 3.
Признак Балансовый не установлен.
Признак учета – Количественный.

ВалютнаяСумма.
Тип – Число,
длина – 10,
точность – 2.
Признак Балансовый не установлен.
Признак учета – Валютный.

Создадим реквизит Содержание.
Тип – строка,
длина – 50,
Допустимая длина – Переменная

В случае с ресурсами регистра мы руководствуемся рассуждениями, аналогичными тем, которыми руководствовались при создании измерения **Валюта**. Балансовые ресурсы – это, в нашем случае, лишь **Сумма**, подразумевают попадание и на счет дебета и на счет кредита одной и той же суммы для соблюдения баланса. Небалансовые позволяют указывать данные для одних счетов и не указывать для других (в общем случае – указывать различные данные). Если мы получаем материалы от поставщика, мы, на счете учета материалов, должны хранить сведения не только о сумме поступления, но и о количестве материалов. А на счете учета расчетов с поставщиком количество поставленных им материалов отражать не нужно – нас интересует лишь сумма нашего долга перед поставщиком. Аналогична ситуация с ресурсом **ВалютнаяСумма.**

Реквизит, в сущности, ни к чему нас не обязывает – он играет вспомогательную роль.

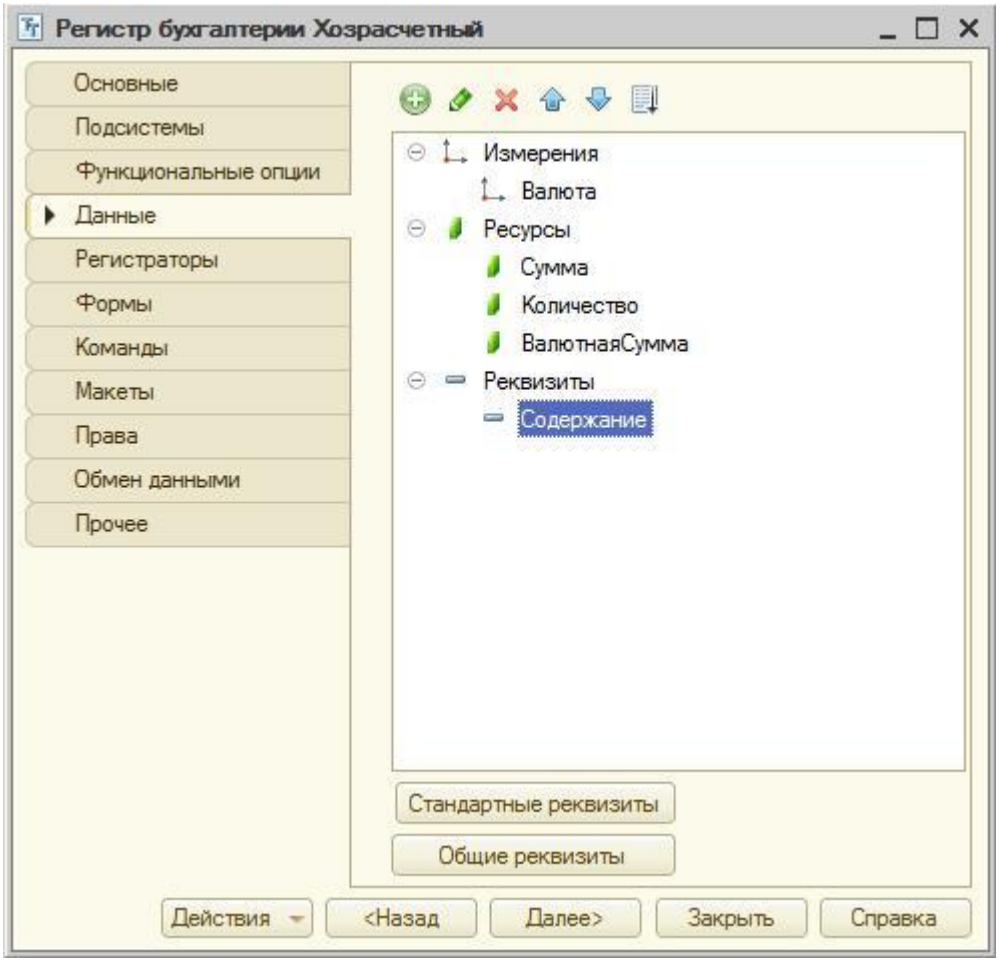


Рисунок 25.13. Состав измерений, ресурсов и реквизитов регистра бухгалтерии.

Вывод

В этой работе мы создали основные компоненты бухгалтерской подсистемы нашей конфигурации. Рассмотрели использование таких объектов, как *Планы видов характеристик*, Планы счетов, Регистры бухгалтерии при формировании бухгалтерской подсистемы. Мы уделили внимание настройке признаков учета и признаков учета *субконто*, подготовили бухгалтерскую подсистему для ведения количественного и валютного учета.

Список использованных источников

Основные источники:

Радченко М.Г., Хрусталева Е.Ю. 1С: Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы, 1С-Паблишинг, 2016.

Кашаев С.М. 1С:Предприятие 8. СПб -БХВ-Петербург, 2016, 336с.

Дополнительные источники:

Документация-описание встроенного языка 1С к ППП 1С:Предприятие 8.3-2018

Михайлов С.Е. 1С Программирование как дважды два. Самоучитель. СПб.: Тритон, 2016 – 173с.

Интернет-ресурсы:

1. Меркулова Т.А. Разработка управляемого приложения на платформе 1С:Предприятие 8// Ульяновский государственный технический университет URL: http://venec.ulstu.ru/lib/disk/2013/Merkulova_up.pdf (дата обращения: 28.05.2018)
2. Заика А. Программирование в «1С:Предприятие 8.2» //Национальный открытый университет. Интуит. URL:http://www.intuit.ru/department/pl/dev1c82up/1/5.html (дата обращения: 28.05.2018)