МИИГАиК

Практические работы по 1С «Предприятие 8.2»

Базы данных

Лыгин А.Н.

2011

Оглавление

Практическая работа № 1

Знакомство, создание информационной базы (0:40)

Работы построены следующим образом: в начале дается краткая теория задачи, далее практика, в конце – контрольные вопросы, на которые Вам нужно будет ответить, чтобы защитить работу.

В процессе работы Вы познакомитесь с некоторыми элементами основы системы 1С: Предприятие 8, и возьмете на себя роль разработчика конфигурации, т.е. создадите свою собственную конфигурацию системы и небольшой пример информационной базы.

Конкретно Вы узнаете о конфигураторе, подсистемах, справочниках, документах, регистрах накопления и оборотных регистрах, периодических регистрах, отчетах, макетах, перечислениях и проведении документов по нескольким регистрам.

Первая работа посвящена знакомству с системой 1С: Предприятие 8 и главным инструментом разработчика – **конфигуратором**. Вы узнаете, что такое объект конфигурации, как можно создать новый объект и задать его свойства. В конце Вы создадите пустую информационную базу для разработки нашей учебной конфигурации.

Общие сведения о системе.

1С: Предприятие является универсальной системой автоматизации экономической и организационной деятельности предприятия.

Поскольку такая деятельность может быть довольно разнообразной, система 1С: Предприятие может приспосабливаться к особенностям конкретной области, в которой она применяется. Для обозначения такой способности используется термин *конфигурируемость*, т.е. возможность настройки системы на особенности конкретного предприятия.

Это достигается благодаря тому, что 1С: Предприятие – совокупность различных программных инструментов, с которыми работают разработчики и пользователи. Логически всю систему делят на две части – **конфигурацию** и **платформу**, которая управляет работой конфигурации. Т.е. основа системы и различные её настройки под конкретные условия.

Сама по себе платформа не может выполнить никаких задач автоматизации, т.к. она создана для обеспечения работы какой-либо конфигурации. Конфигурация это синоним прикладного решения.

Существует множество прикладных решений (конфигураций), например: 1С: Управление небольшой фирмой 8, 1С: Бухгалтерия 8, 1С: Предприятие 8. Управление торговлей, 1С: Зарплата и Управление Персоналом 8, 1С: Предприятие 8. Управление производственным предприятием, 1С: Консолидация 8 и др.

Типовое прикладное решение является универсальным и способно удовлетворить потребности разных предприятий, работающих в одной области деятельности. С другой стороны, такая универсальность приведет к тому, что не все возможности прикладного решения на конкретном предприятии будут использоваться, а каких-то возможностей будет недоставать. Вот тут и пригодится возможность конфигурации системы. Платформа содержит средства, позволяющие вносить изменения в используемую конфигурацию и даже создать ее с нуля.

Для обеспечения таких возможностей система 1С: Предприятие имеет два режима работы: **1С: Предприятие** и **Конфигуратор**.

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

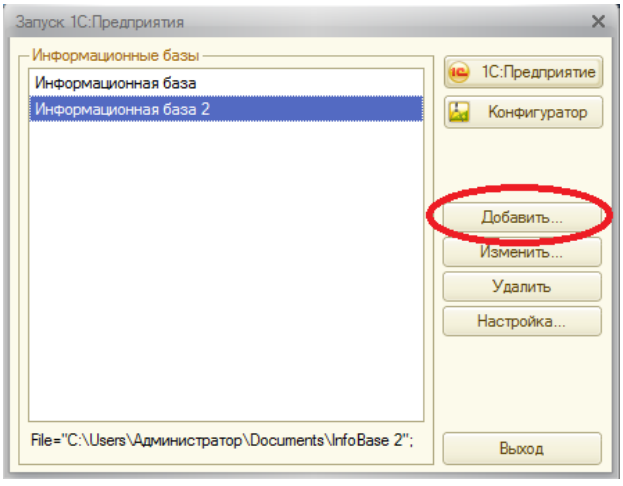
Режим Конфигуратор используется разработчиками и администраторами информационных баз. Именно этот режим предоставляет инструменты для изменения существующей или создания новой конфигурации.

Итак, перейдем к практике.

Создание новой информационной базы.

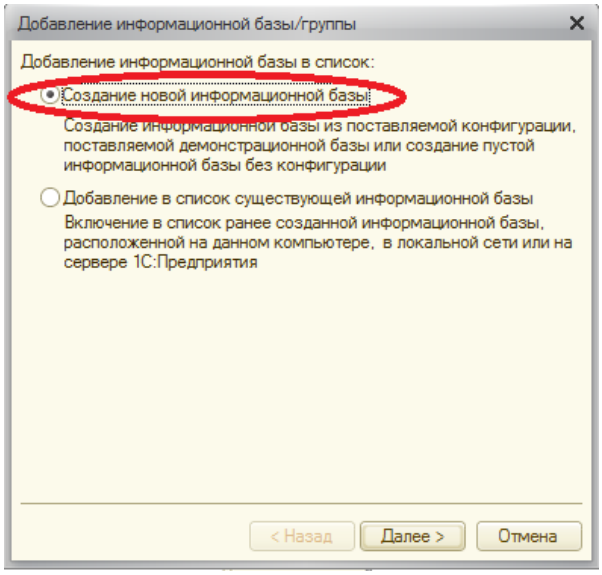
Запустите 1С: Предприятие. В открывшемся диалоге Вы увидите список информационных баз, с которыми Вы работаете. Если список пуст, система предложит Вам создать новую базу.

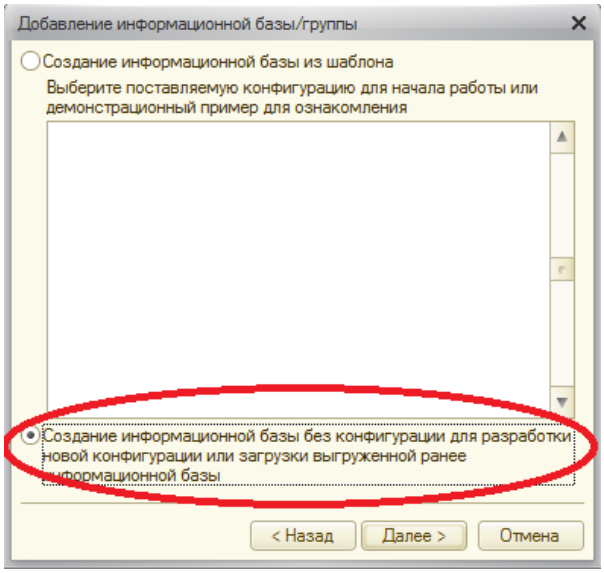
В любом случае, нажмите кнопку **Добавить**



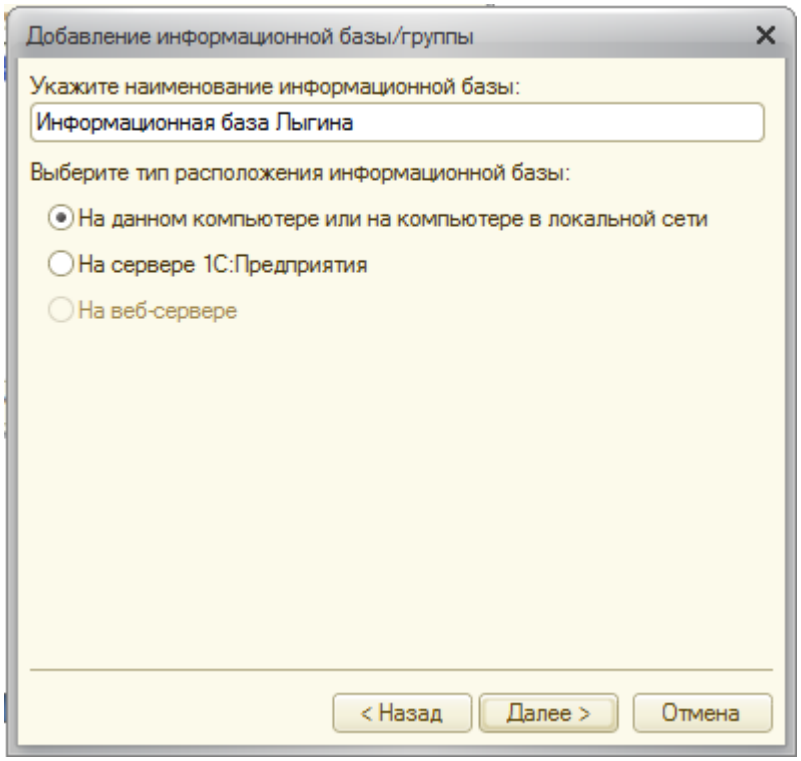
В открывшемся диалоге выберите пункт **Создание новой информационной базы**.

Нажмите **Далее**, выберите пункт **Создание информационной базы без конфигурации**…





Нажмите кнопку **Далее**. Задайте наименование Вашей информационной базы типа «Информационная база ваша_фамилия» и выберите тип ее расположения **На данном компьютере**.

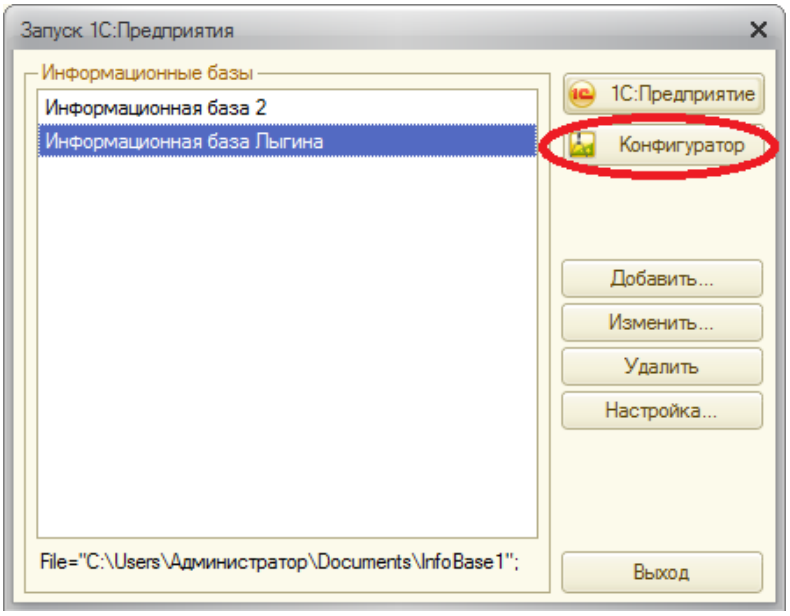


Нажмите кнопку **Далее**. Укажите каталог для расположения Вашей информационной базы.

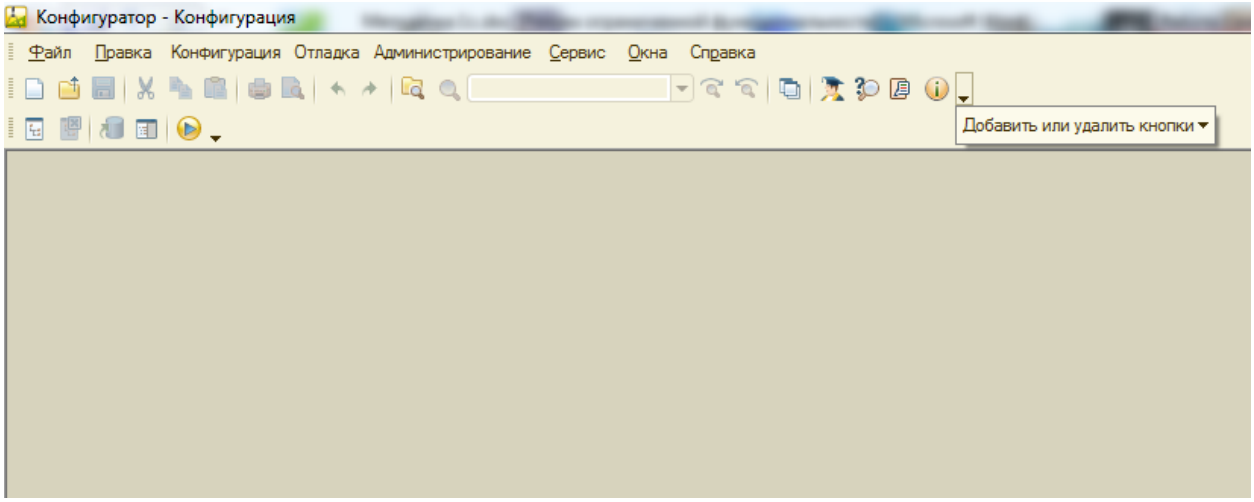
Нажмите **Далее**, ничего не меняйте и нажмите **Готово**.

В диалоге запуска 1С: Предприятия, в списке информационных баз, Вы увидите созданную вами новую пустую базу.

Запустите 1С: Предприятие в режиме Конфигуратора.



Появится пустое окно конфигуратора, возможно с минимальным количеством кнопок. Чтобы включить нужные кнопки, нажмите на стрелочку Добавить или удалить кнопки около кнопки  .



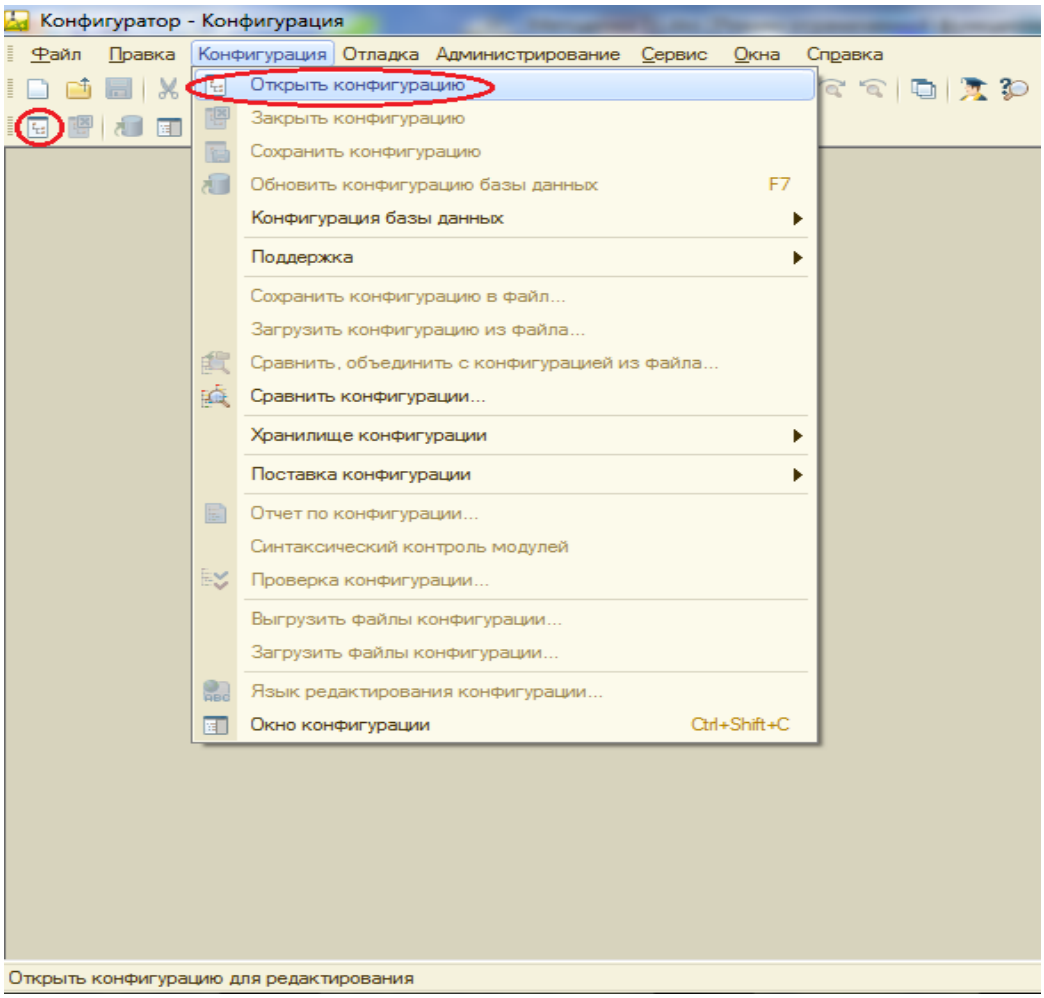
Сразу под заголовком окна находится главное меню конфигуратора, содержащее пункты Файл, Правка, Конфигурация, Администрирование и др. В каждом из них находится много подпунктов, вызов которых обеспечивает выполнение различных действий конфигуратора.

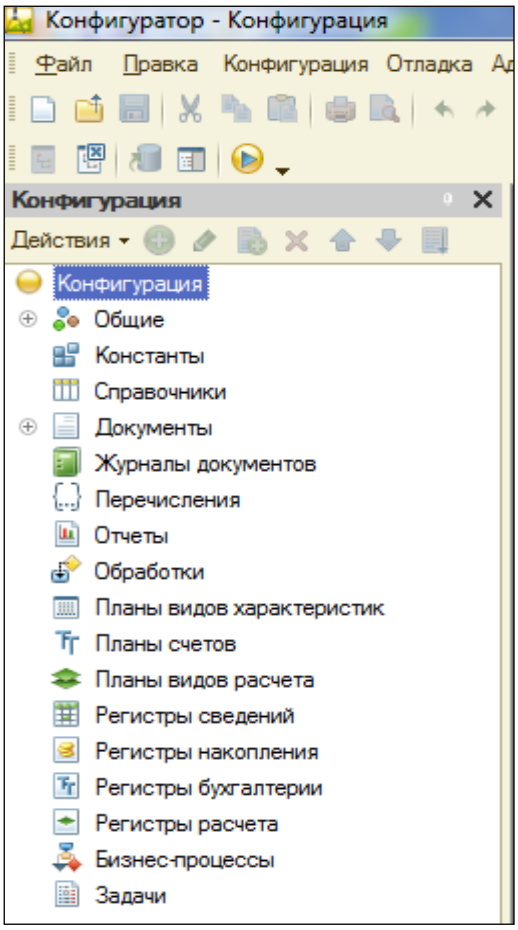
Ниже находится панель инструментов конфигуратора, в которую помещены наиболее часто используемые действия, вызываемые из верхнего меню, в виде кнопок-пиктограмм. Их большое количество может Вас смутить, но со временем вы будете легко в них ориентироваться. Если подвести курсор к какой-нибудь пиктограмме и задержать, появится подсказка о ее назначении.

Дерево объектов конфигурации

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

Конфигурация – Открыть конфигурацию или соответствующей кнопкой.





Откроется *дерево объектов конфигурации*:

Дерево объектов конфигурации – основной инструмент, с которым работает разработчик. Оно содержит в себе почти всю информацию о том, из чего состоит конфигурация.

Всё, из чего состоит конфигурация, сгруппировано и сейчас дерево показывает Вам эти группы.

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

Предприятие. Кроме того, конфигурация описывает алгоритмы обработки данных, содержит информацию о том, как будут выглядеть данные на экране и на принтере и т.д.

В дальнейшем платформа на основании этого описания создаст базу данных, которая будет иметь необходимую структуру, и предоставит пользователю возможность работать с этой базой данных.

Объекты конфигурации – детали конструктора, из которого собирается конфигурация. Объекты одного вида отличаются от объектов другого вида тем, что имеют разные свойства. Объекты могут взаимодействовать друг с другом и мы можем описать такое взаимодействие. Объекты могут взаимодействовать между собой напрямую или через другие объекты, могут быть сложные объекты, состоящие из более простых. Самое важное качество объектов конфигурации – их прикладная направленность. Объекты представляют собой аналоги реальных объектов, которыми оперирует предприятие в ходе своей работы.

Например, на каждом предприятии существуют различные документы, с помощью которых оно фиксирует факты совершения хозяйственных операций. Точно также в конфигурации существуют объекты вида **Документ**. Кроме этого, на предприятии ведется учет сотрудников, товаров и т.д. Для этого в конфигурации есть объекты вида **Справочник**.

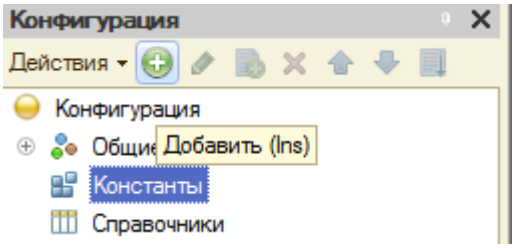
В системе 1С:Предприятие разработчик может использовать только ограниченный набор объектов конфигурации, зашитый в платформе. Он только может добавлять в конфигурацию какой-либо из стандартных объектов, поставляемых системой.

Как добавить объект конфигурации

Любой объект конфигурации добавляется несколькими способами, предварительно выделив нужную группу.

- Установите курсор на ветку объектов конфигурации, где хотите создать объект и нажмите на кнопку **Действия – Добавить**.
- Установите курсор на ветку объектов конфигурации, где хотите создать объект и нажмите правую кнопку мыши – появится контекстное меню, пункт **Добавить,** рядом указана клавиша быстрого вызова – Insert.

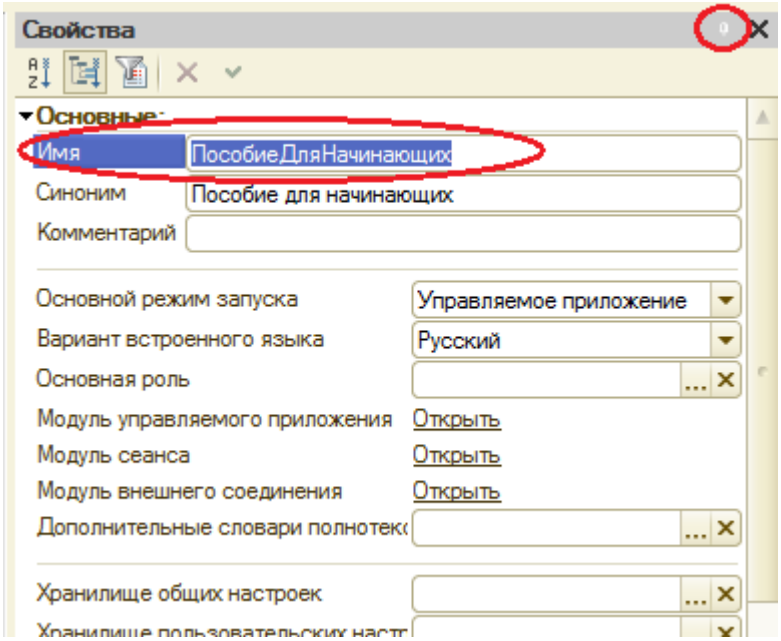
- Установите курсор на ветку объектов конфигурации, где хотите создать объект и нажмите кнопку-пиктограмму **Добавить**



Окно свойств

Окно свойств (палитра свойств) – специальное служебное окно, которое позволяет редактировать все свойства объекта конфигурации и другую связанную с ним информацию.

Зададим имя нашей конфигурации. Выделите в дереве объектов конфигурации корневой элемент Конфигурация и двойным щелчком мыши отройте её окно свойств. Назовите конфигурацию **ПособиеДляНачинающих**.



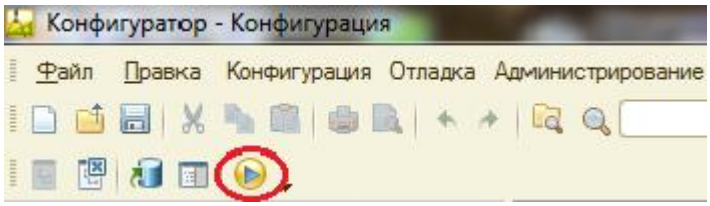
Почему мы именно так задали имя?

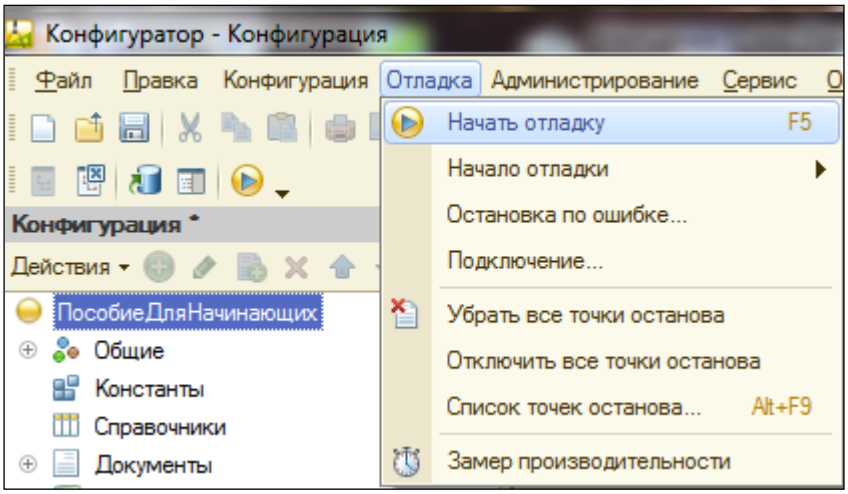
Потому что системе так удобнее различать объекты, когда название не содержит пробелов. Для пользователя будет виден синоним названия, который вполне читаем.

Поэкспериментируйте с настройками отображения окна свойств, используя контекстное меню (правая кнопка мыши) на заголовке окна и белая кнопочка около кнопки закрытия окна.

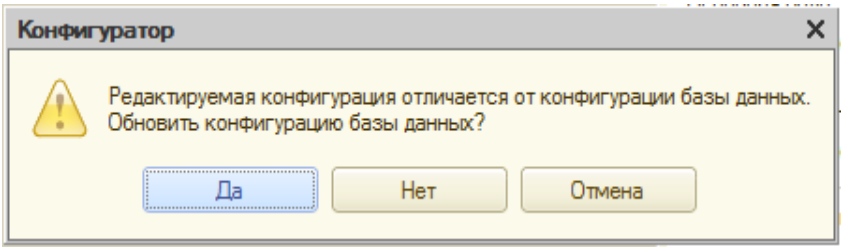
Запуск отладки в режиме 1С: Предприятие

Проверим наши первые изменения в режиме 1С: Предприятие. Для этого выполним пункт меню **Отладка – Начать отладку** или нажмем соответствующую кнопку на панели инструментов (клавиша F5)



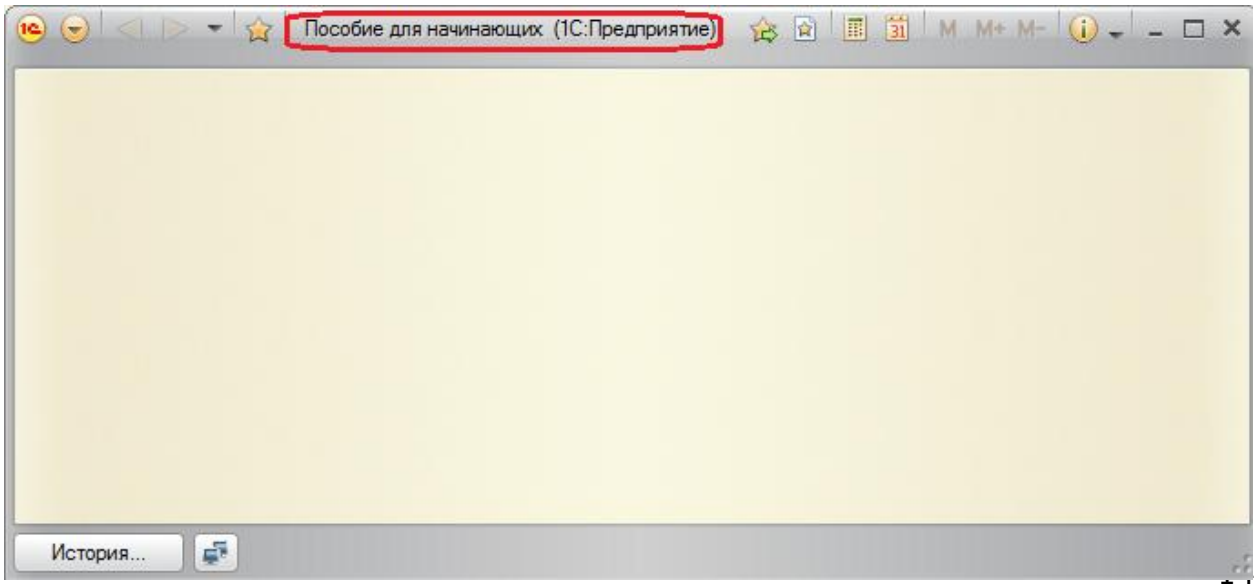


Система проанализирует наличие изменений в конфигурации и выдает вопрос об обновлении конфигурации базы данных.



Ответим на вопрос конфигуратора **Да** и на экране появится окно 1С: Предприятия.

В заголовке окна Вы видите название Вашей конфигурации. Пустое пространство – рабочая область приложения, которая пока ничем не заполнена. Кроме заголовка конфигурации в окне 1С: Предприятия ничего не появилось.



Контрольные вопросы:

- ✓ Что такое конфигурируемость системы 1С: Предприятие.
- ✓ Из каких основных частей состоит система.
- ✓ Что такое платформа и конфигурация.
- ✓ Для чего используются разные режимы запуска системы.
- ✓ Для чего нужно дерево объектов конфигурации.
- ✓ Что такое объекты конфигурации.
- ✓ Как можно добавить новый объект конфигурации.
- ✓ Как запустить 1С: Предприятие в режиме отладки.

Практическая работа № 2

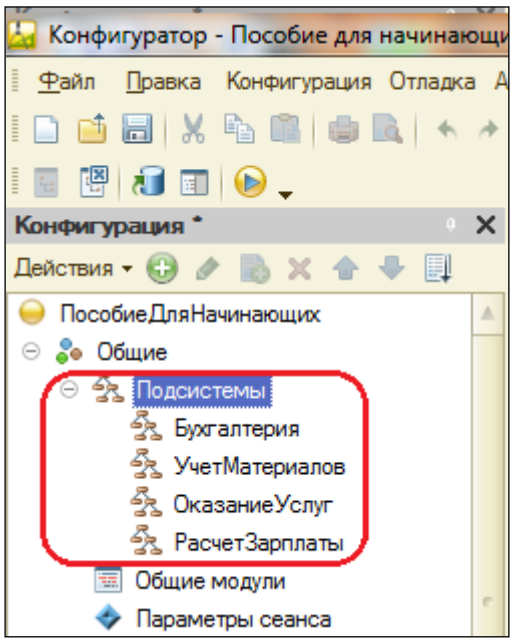
Подсистемы (0:45)

В этой работе Вы познакомитесь с объектом конфигурации Подсистема как основой описания интерфейса 1С: Предприятия 8. Вы создадите несколько подсистем, определяющих логическую структуру прикладного решения, настроите их внешний вид и порядок их следования в интерфейсе 1С: Предприятие.

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

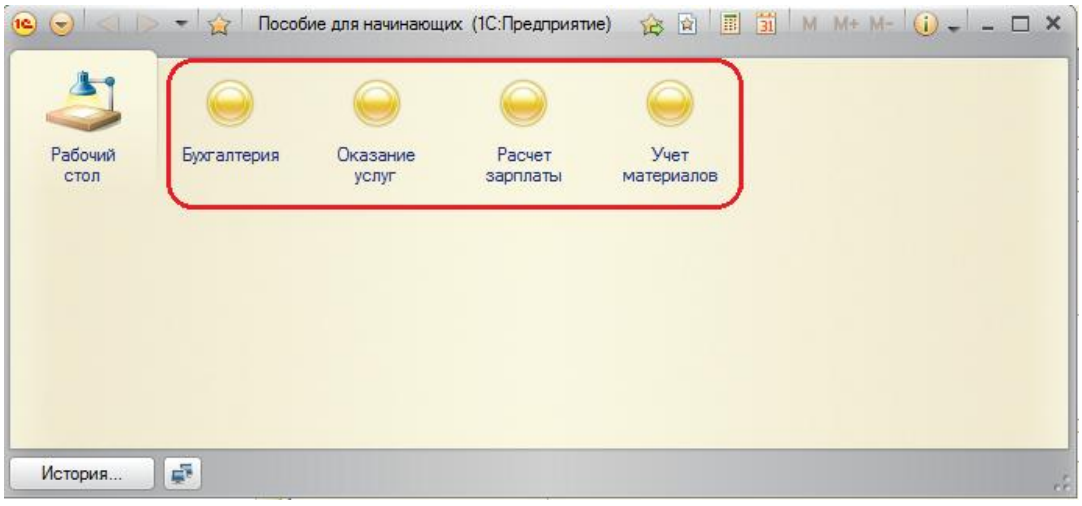
Подсистемы позволяют выделить в конфигурации функциональные части, на которые логически разбивается создаваемое прикладное решение.

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



Сейчас у вас дерево подсистем пусто, на рисунке показан пример какими могут быть подсистемы.

Подсистемы верхнего уровня являются основными элементами интерфейса, т.к. образуют разделы прикладного решения.



Так будет выглядеть окно в режиме 1С: Предприятие с указанными подсистемами.

Каждый объект

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

Таким образом, подсистемы определяют структуру прикладного решения, организуют весь пользовательский интерфейс, позволяют рассортировать различные документы, справочники и отчеты по логически связанными с ними разделами, в которых пользователю будет проще их найти и удобнее с ними работать. При этом каждому конкретному пользователю будут видны лишь те разделы, которые ему нужны в процессе работы.

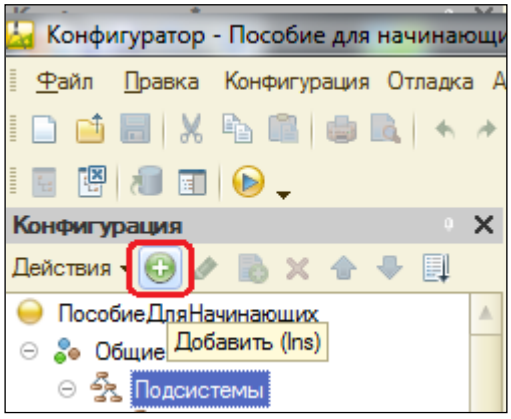
Всю производственную деятельность нашей фирмы можно разделить на учет материалов и оказание услуг. А кроме этого, для выполнения специальных административных функций с базой данных нам нужно иметь отдельную подсистему для администратора.

Добавление подсистемы

Сейчас вы создадите пять новых объектов конфигурации **Подсистема** с именами: **Бухгалтерия, РасчетЗарплаты, УчетМатериалов, ОказаниеУслуг, Предприятие**.

Закройте окно 1С: Предприятие и вернитесь в конфигуратор.

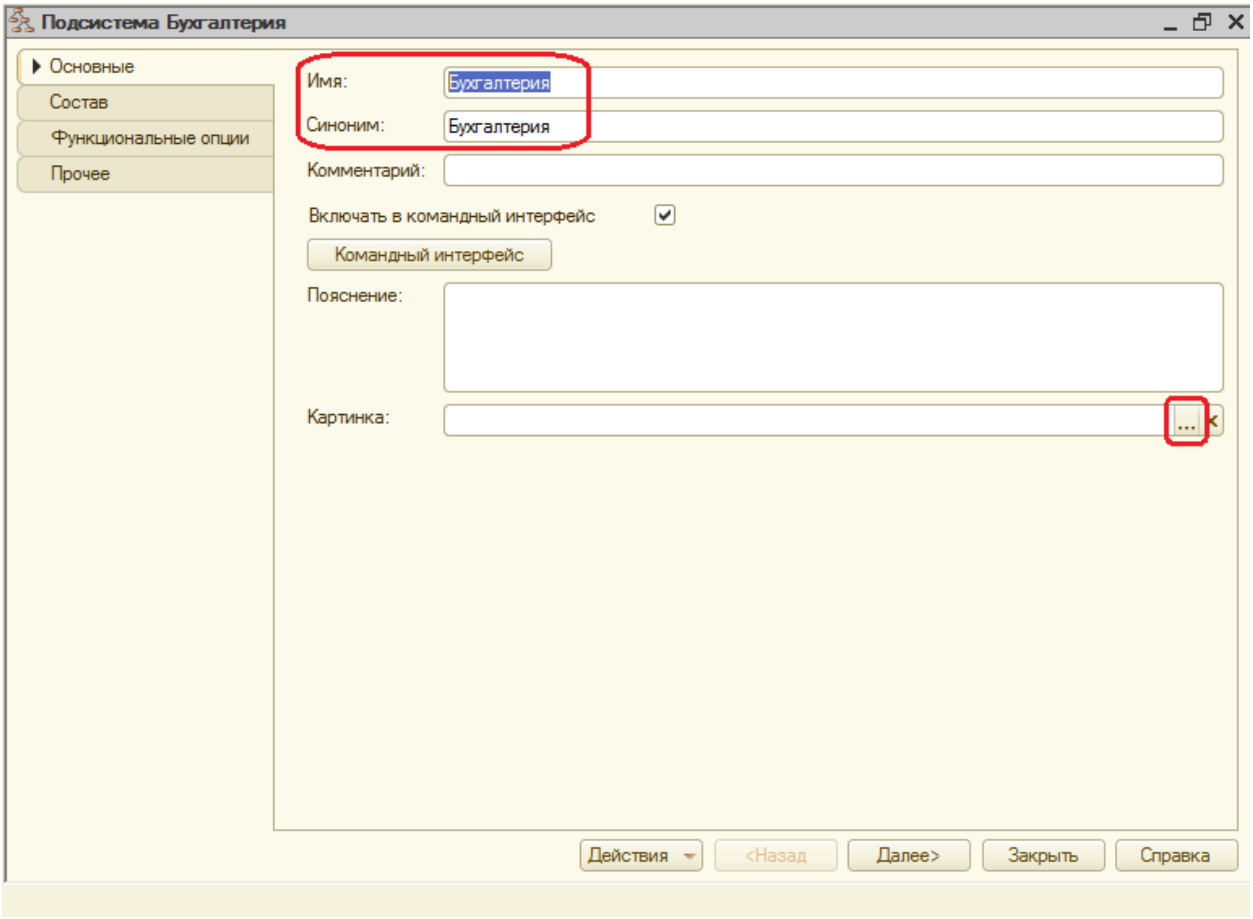
Раскройте ветку Общие в дереве объектов конфигурации, нажав на + слева от нее. Выделите ветку **Подсистемы** и нажмите **Добавить** (или кнопка **Insert** на клавиатуре) или воспользуйтесь контекстным меню.



Откроется *окно редактирования объекта конфигурации*. Оно предназначено для сложных объектов и позволяет быстро создавать такие объекты путем выполнения последовательности действий.

Задайте *имя* подсистемы – **Бухгалтерия**. На основании имени система сама создаст *синоним* – **Бухгалтерия**.

Имя является основным свойством любого объекта конфигурации. Можно использовать имя, присвоенное системой, но лучше использовать своё, более понятное имя. Имя может быть любое, начинающееся с буквы и не содержащее пробела и других специальных символов. Имя объекта является уникальным и служит для обращения к свойствам и методам объекта на встроенном языке.

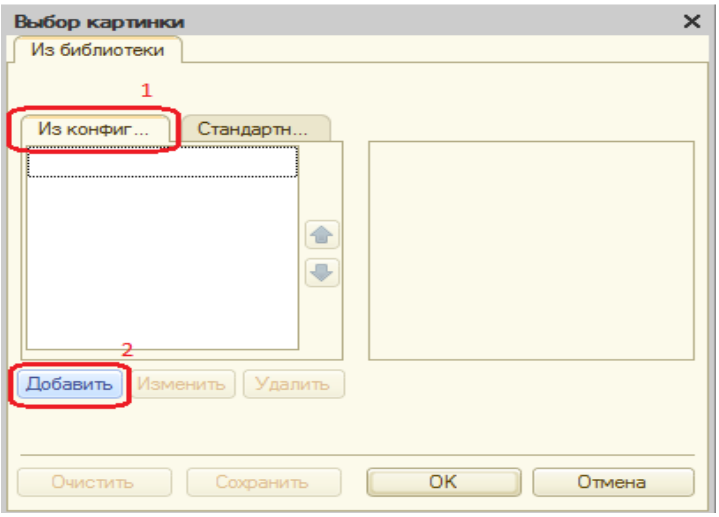


Свойство **Синоним** также есть у любого объекта конфигурации и предназначено для хранения альтернативного наименования объекта, которое будет видно пользователю. На Синоним нет никаких ограничений и его можно задавать в привычно виде.

Картинка подсистемы

Можно задать также картинку (иконку) для отображения подсистемы в окне режима 1С: Предприятие.

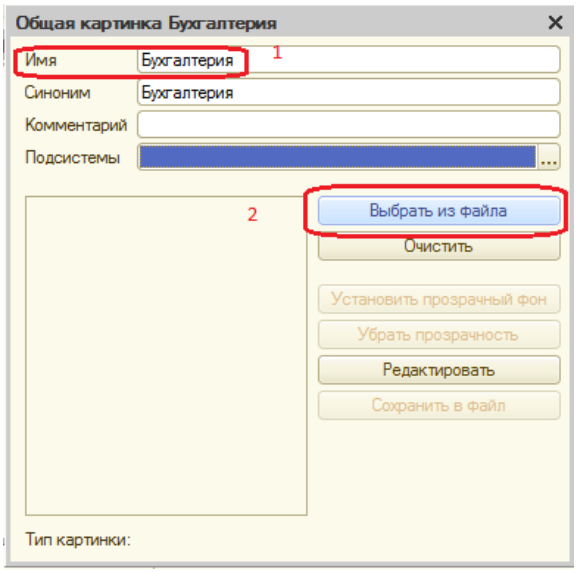
Нажмите кнопку выбора в поле *Картинка*.



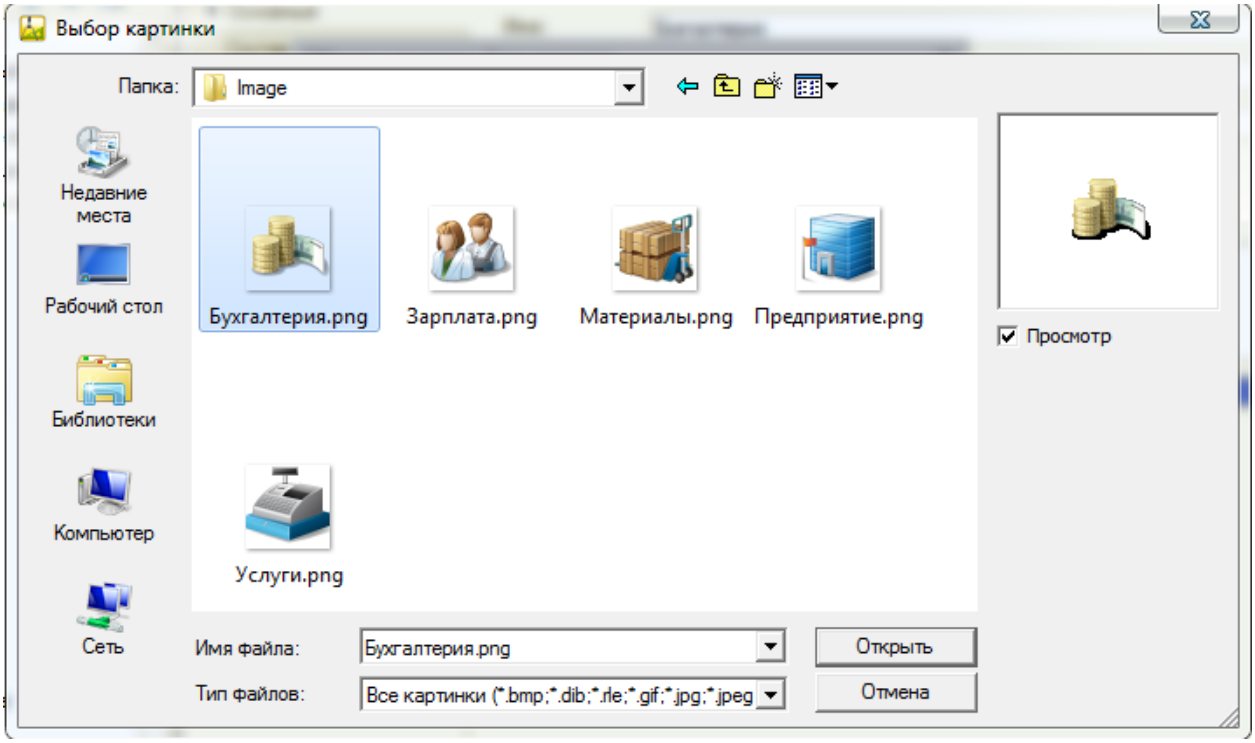
Добавьте картинку в список на закладке **Из конфигурации** – нажмите кнопку **Добавить** как показано на рисунке.

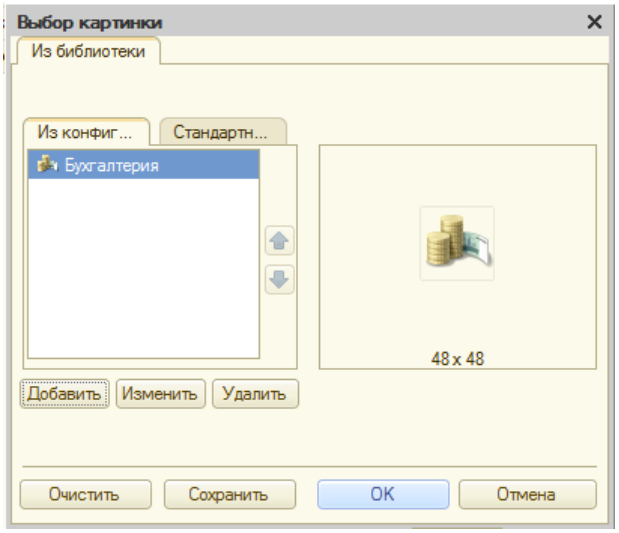
Система создаст объект *Общая картинка* и откроет окно его свойств. Дайте

картинке имя **Бухгалтерия**. Нажмите **Выбрать из файла**, чтобы задать саму картинку.



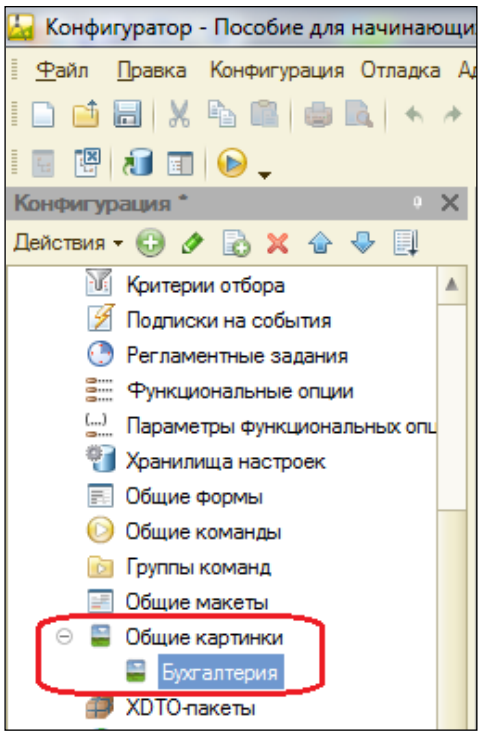
Спросите у преподавателя где находятся картинки и задайте для подсистемы Бухгалтерия одноимённую картинку.





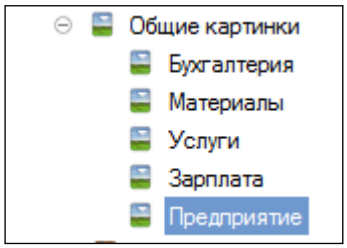
Закройте окно выбора файла картинки. Вы увидите, что картинка появилась в списке на закладке Из конфигурации. Нажмите ОК.

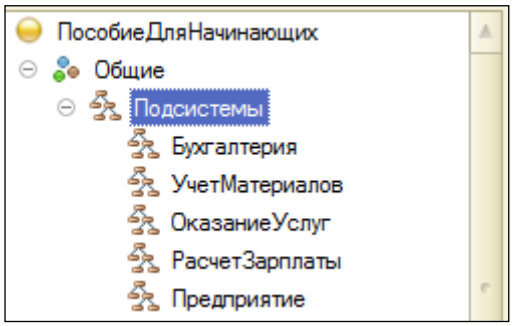
После наших действий в дереве объектов конфигурации в ветке Общие картинки появилась картинка Бухгалтерия, которую мы можем редактировать и использовать в нашей конфигурации.



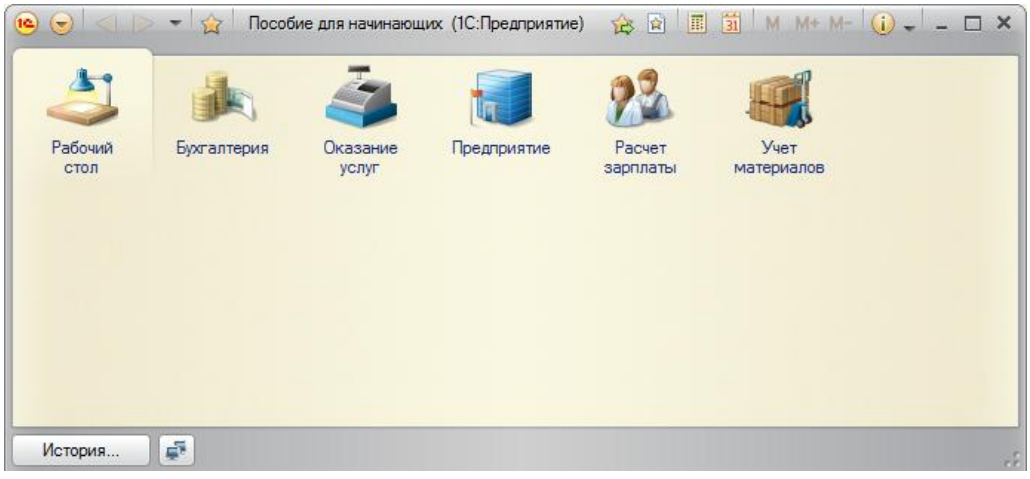
В интерфейсе 1С: Предприятия в качестве названия раздела будет показан *синоним* подсистемы, а над ним будет показана данная картинка.

Аналогично создайте подсистемы с именами **УчетМатериалов**, **ОказаниеУслуг**, **Предприятие**, **РасчетЗарплаты**. Установите для них соответствующие картинки с именами **Материалы, Услуги, Зарплата, Предприятие**. Должно получиться всего 5 подсистем.



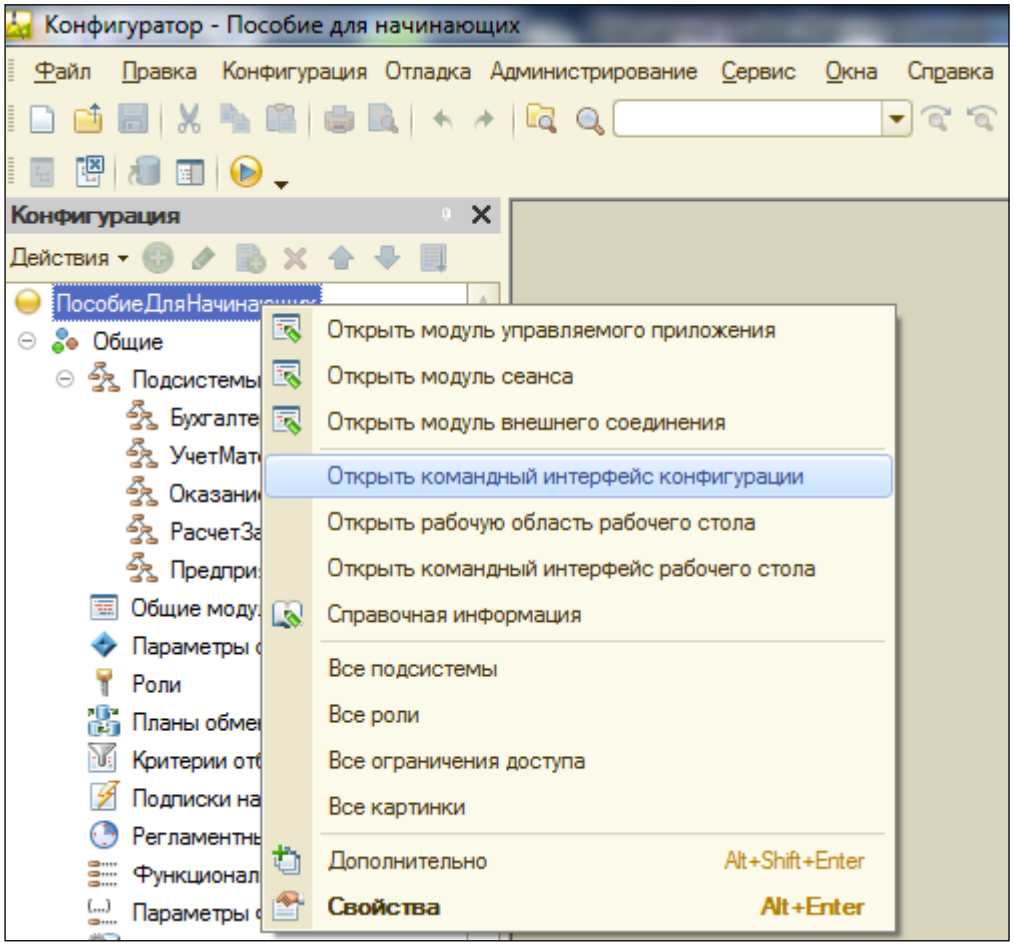


Запустите 1С: Предприятие в режиме отладки кнопкой  Вы увидите изменившееся с прошлого раза окно, где представлены наши подсистемы с установленными Вами картинками.

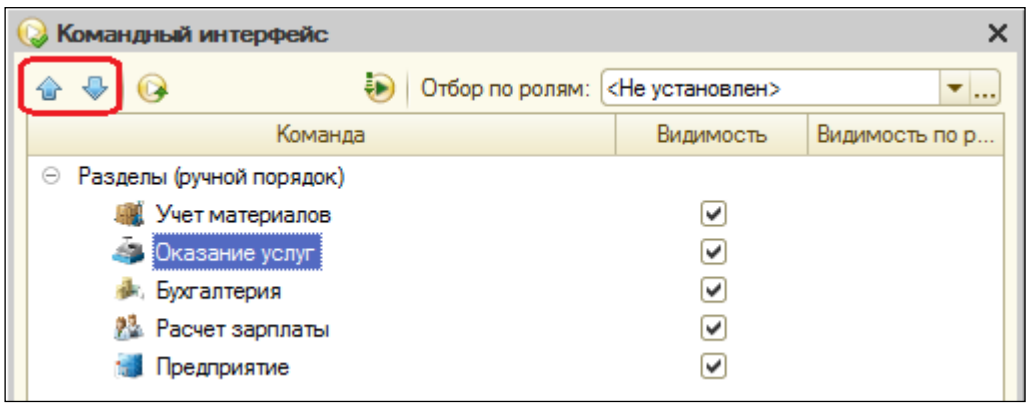


Порядок разделов

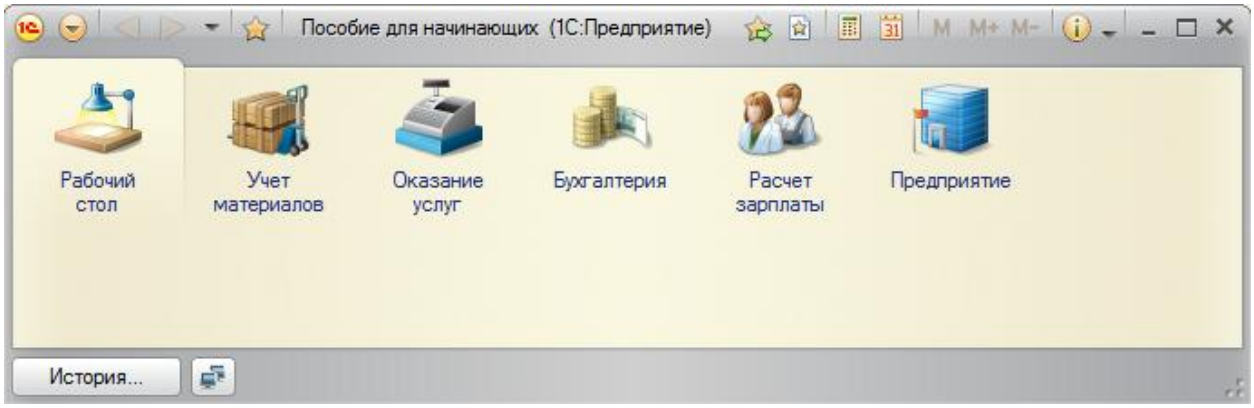
Допустим, порядок расположения подсистем нас не устраивает. Изменим его. Закройте приложение и вернитесь в конфигуратор. Выделите корень дерева объектов конфигурации **ПособиеДляНачинающих**, нажатием правой кнопки мыши вызовите контекстное меню и выберите пункт **Открыть командный интерфейс конфигурации.**



В открывшемся окне Вы увидите список созданных подсистем. С помощью кнопок **Вверх**, **Вниз** измените порядок расположения разделов в этом списке. Расположите сначала подсистемы, отражающие производственную деятельность фирмы: **УчетМатериалов** и **ОказаниеУслуг**, затем бухгалтерскую деятельность и расчет зарплаты сотрудников: **Бухгалтерия** и **РасчетЗарплаты**, а затем подсистему **Предприятие**.



Закройте *командный интерфейс* и запустите 1С: Предприятие в режиме отладки (клавиша F5).



Как видите, порядок отображения подсистем изменился согласно указанному в командном интерфейсе.

После успешного завершения каждой работы сохраняйте конфигурацию: **Администрирование – Выгрузить информационную базу**. Это полезно, если вы запутаетесь в своих действиях и захотите вернуться к работающему варианту. Это можно сделать, выполнив команду **Администрирование – Загрузить информационную базу**.

Контрольные вопросы

- ✓ Для чего используется объект конфигурации Подсистема.
- ✓ Как управлять порядком вывода и отображения подсистем в конфигурации.
- ✓ Как сохранить информационную базу.

Практическая работа № 3

Справочники (2:10)

В этой работе Вы познакомитесь с объектом конфигурации **Справочник**. Научитесь создавать справочники, описывать наиболее важные элементы их структуры и заполнять их данными.

Также Вы познакомитесь с объектом конфигурации **Форма**. Узнаете, какие виды форм существуют у объекта **Справочник** и в каких ситуациях они используются.

Что такое справочник

Объект конфигурации **Справочник** предназначен для работы со списками данных. Как правило, в работе любой фирмы используются списки сотрудников, товаров, клиентов, поставщиков и т.д. Свойства и структура этих списков описываются в объектах Справочник, на основе которых платформа создает в базе данных таблицы для хранения информации из этих справочников.

Справочник состоит из *элементов*. Например, для справочника сотрудники элементом является сотрудник, для справочника товаров – товар. Пользователь в процессе работы может самостоятельно добавлять новые элементы в справочник: новых сотрудников, создать новый товар, внести клиента. В базе данных каждый элемент справочника представляет собой отдельную запись (строчку) в основной таблице, хранящей информацию из этого справочника.

Каждый элемент справочника, как правило, содержит более подробно описывающую этот элемент информацию. Например, все элементы справочника Товары могут содержать дополнительную информацию о производителе, сроке годности и др. Набор такой информации является одинаковым для всех элементов справочника и для описания такого набора используются **реквизиты** объекта конфигурации Справочник, которые также являются объектами конфигурации. Можно это представить наглядно как таблицу: Справочник *Товары* содержит список товаров, имеющих реквизиты (колонки в таблице Товары) *Производитель, Поставщик, Наименование*.

Можно использовать стандартные реквизиты, свойственные выбранному типу Справочника, а можно создать свои, более удобные.

Каждый элемент справочника может содержать некоторый набор информации, одинаковой по структуре, но различной по количеству и предназначена для разных элементов справочника.

Например, каждый элемент справочника *Сотрудники* (каждый сотрудник) может содержать информацию о составе семьи сотрудника. Для одного сотрудника это будет только жена, для другого – жена, сын, дочь.

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

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

Элементы одного справочника могут быть подчинены элементам или группам другого справочника. Например, Справочник *ЕдиницыИзмерения* может быть подчинен справочнику *Товары*, тогда для каждого товара можно будет указать единицы измерения.

Иногда возникают ситуации, когда необходимо, чтобы в справочнике некоторые элементы существовали всегда, независимо от действий пользователя. В таком случае Справочник позволяет описать любое количество таких неизменяемых элементов, называемых **предопределенными элементами справочника**.

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

Формы справочника

Удобно представлять справочник в разном виде – список, дерево, в определенной последовательности и т.п. Для этого используется объект конфигурации **Форма**.

Форма служит для удобного визуального представления и ввода данных.

Система может самостоятельно создать все формы, которые нужны для представления данных Справочника.

«Простой» справочник

После небольшого знакомства с возможностями объекта конфигурации Справочник, создадим несколько таких объектов.

Допустим, наша фирма оказывает услуги по ремонту бытовой техники. Для ведения учёта потребуется хранить список сотрудников предприятия, которые будут оказывать услуги. Также потребуется список клиентов, с которыми работает наша фирма. После потребуется перечень услуг, оказываемых фирмой и список материалов, которые могут быть израсходованы, список складов, где они хранятся.

Создадим сначала справочник, в котором будут храниться данные о клиентах.

Откройте в конфигураторе нашу конфигурацию, выделите в дереве объектов ветку **Справочники**, нажмите **Добавить**. Назовите Справочник – **Клиенты**.

На основании имени система создаст *синоним* – **Клиенты**. Он служит для представления объекта в интерфейсе программы.

Представление объекта определяет название объекта в единственном числе и используется в названии стандартной команды (Например, команды создания клиента Клиент: создать.) Используется также в интерфейсе команды добавления нового клиента, товара и т.п. и в заголовке формы, (если не указано расширенное представление объекта) в ссылке на клиента, товар...

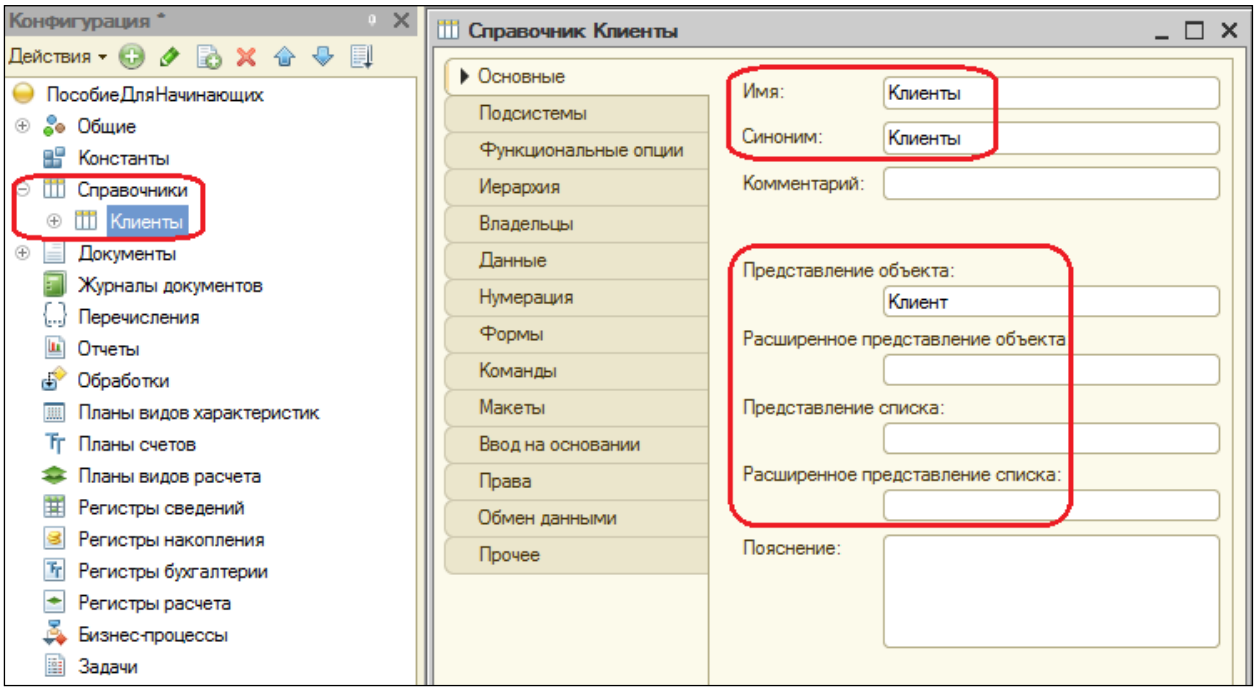
Представление объекта нужно задавать тогда, когда *синоним* задан во множественном числе или когда описывает множество объектов. Т.е. это название одного объекта списка.

Расширенное представление объекта – определяет заголовок формы объекта. Если не задано, используется **Представление объекта**.

Представление списка определяет название списка объектов. Нужно задавать, когда *синоним* задан в единственном числе.

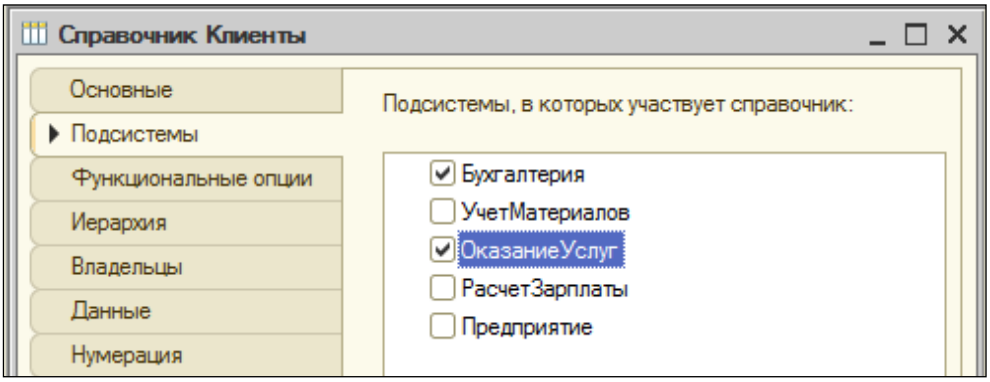
Расширенное представление списка определяет заголовок формы списка. Если не задано, используется **Представление списка**.

Задайте два свойства **Представление объекта** – *Клиент* и **Представление списка** – *Клиенты* (последнее необязательно, т.к. будет использован синоним).



Нажмите кнопку **Далее** и перейдите к закладке **Подсистемы**.

На этой закладке определяется, в каких подсистемах будет отображаться справочник.



Логично, что список клиентов будет доступен в разделе Оказание услуг и Бухгалтерия, поэтому отметьте в списке подсистемы **Бухгалтерия** и **ОказаниеУслуг**.

Нажмите на закладку **Данные**. **Длина кода** нас устраивает, а **длину наименования** установите в 50 символов.

Длина кода – важное свойство справочника. Код используется для идентификации элементов (ключевое поле) справочника и содержит

уникальное для каждого элемента справочника значение. Платформа может сама контролировать уникальность кодов и поддерживать автоматическую нумерацию элементов справочника. Поэтому от длины кода будет зависеть количество элементов, содержащихся в справочнике.

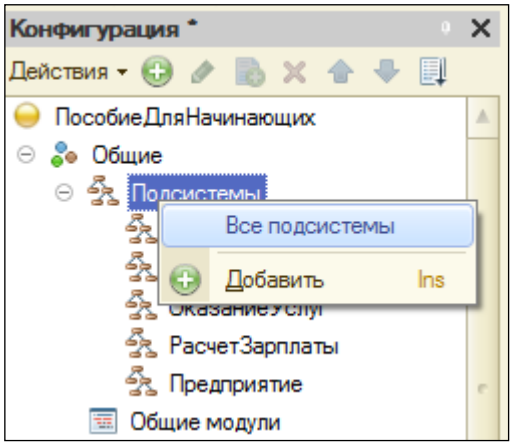
Команда добавления нового элемента

Настроим интерфейс приложения, чтобы было удобнее вводить новые элементы справочника.

Команда для открытия списка справочника, как и команда для создания его новых элементов, добавляется в интерфейс тех подсистем, в которых будет отображаться справочник. Но команда создания новых элементов по умолчанию невидима в интерфейсе приложения. Это объясняется тем, что возможность просматривать списки справочника нужна почти всегда, а возможность создания новых элементов – редко. Поэтому соответствующую команду следует включать только для некоторых справочников.

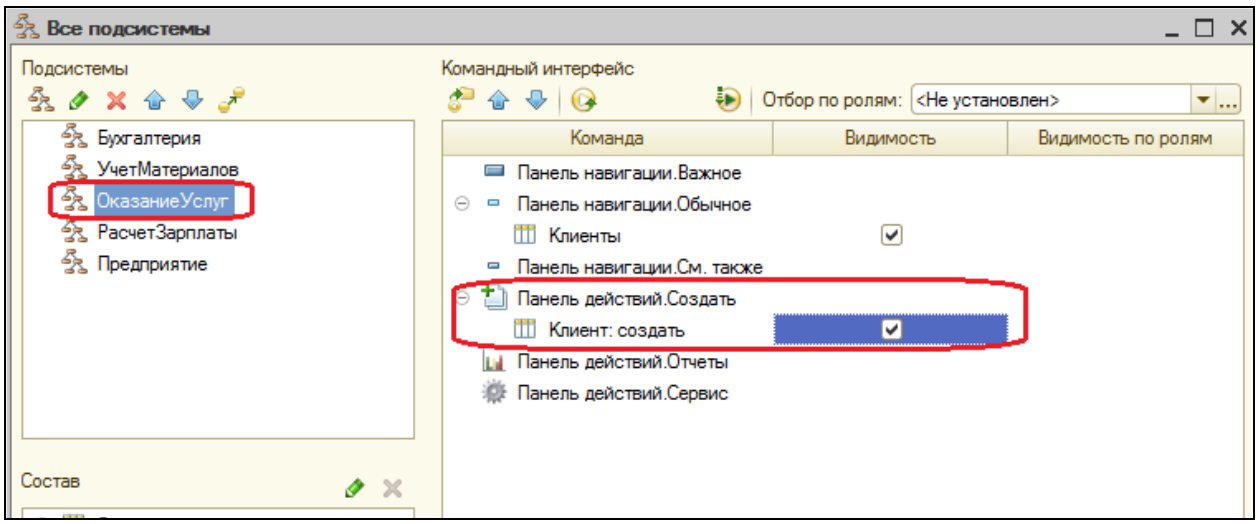
Сделаем доступной в панели действий раздела **ОказаниеУслуг** стандартную команду для создания новых клиентов.

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



В открывшемся окне **Все подсистемы** слева в списке **Подсистемы** выделим **ОказаниеУслуг**, справа в списке Командный интерфейс отразятся все команды выбранной подсистемы.

В группу Панель действий.Создать добавилась команда Клиент:создать для создания нового элемента справочника, но она невидима по умолчанию. Включите видимость этой команды, поставив соответствующую галочку.



Для подсистемы **Бухгалтерия** никаких команд добавлять в панель действий не будем, т.к. она там не нужна.

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

Закройте окно редактирования справочника **Клиенты** и запустите 1С: Предприятие в режиме отладки, всегда отвечая утвердительно на вопрос обновления конфигурации и принятии изменений.

Панель навигации и панель разделов

Если перейти в разделы **Бухгалтерии** и **Оказания услуг**, то слева увидите в вертикальной области появилась **панель навигации**.

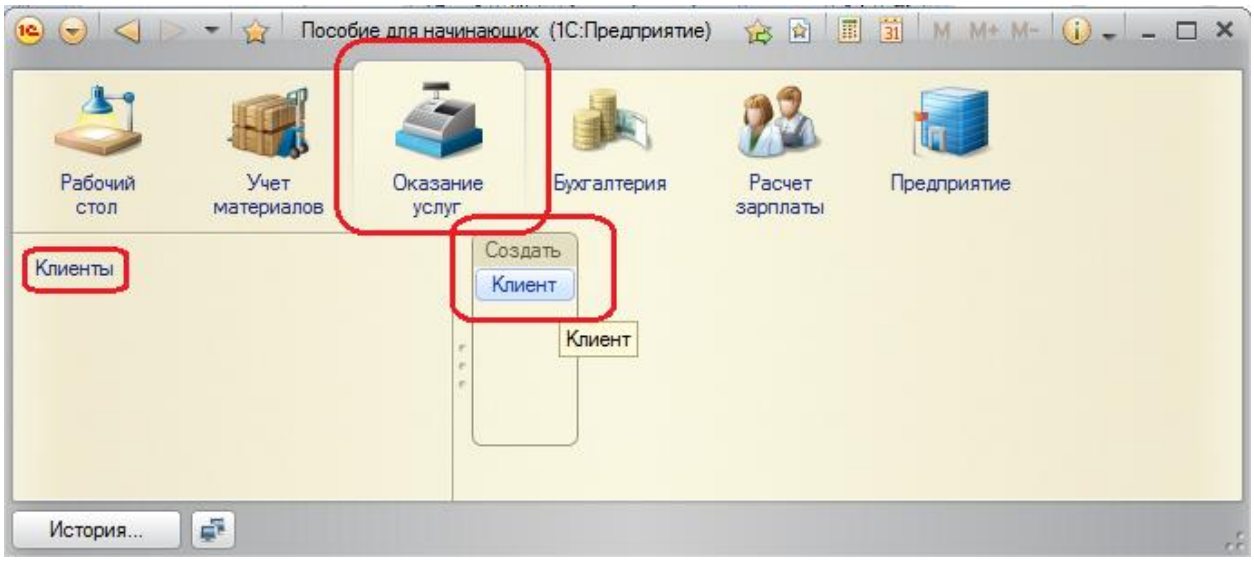
Панель навигации отображает структуру выбранного раздела и предназначена для быстрого перехода к различным спискам в пределах выбранного раздела.

Сейчас она содержит команду для открытия нашего первого списка – **Клиенты**. Её название определяется свойством **Представление списка**, которое мы задали для справочника. Если оно не задано, то используется синоним объекта.

Также в Оказании услуг появилась **панель действий**. Панель действий содержит команды, которые соответствуют текущему разделу.

Сейчас в панели действий раздела **Оказание услуг** в группе **Создать** доступна команда для создания элементов нашего

справочника **Клиенты**, которую мы сделали видимой в интерфейсе этого раздела.

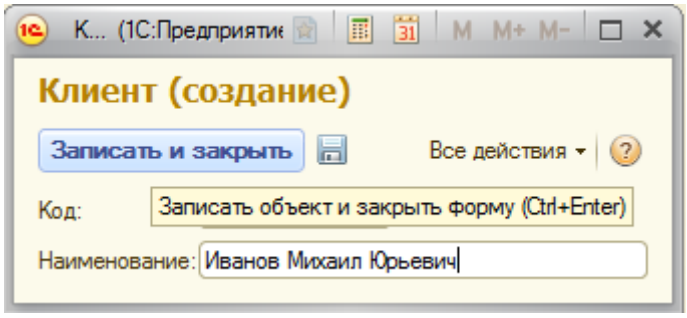


Обратите внимание, что название команды создания нового элемента определяется свойством **Представление объекта**, которое мы задали для этого справочника. Если бы не задали, используется синоним.

Создание элементов справочника

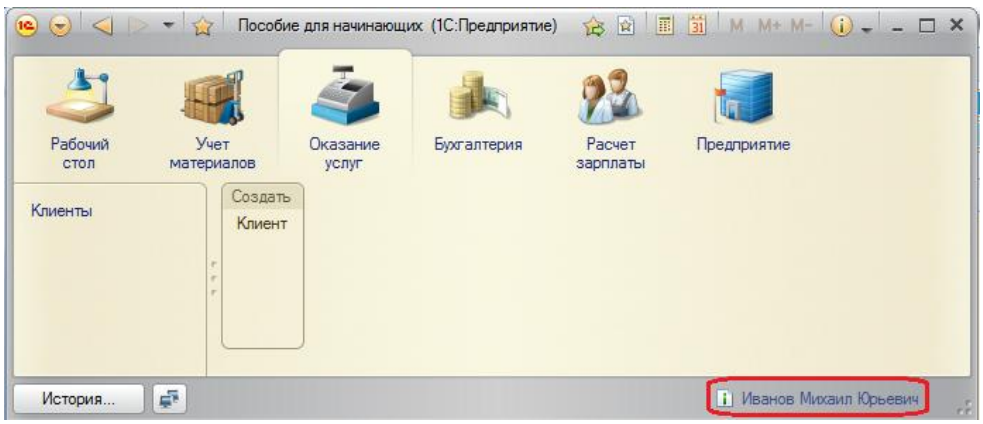
Пока наш справочник пуст, добавим в него несколько элементов.

Для этого выполним команду **Клиент** в *панели действий* раздела **Оказание услуг**. Перед Вами откроется форма для создания элемента справочника – основная форма объекта. Внесите имя нового клиента – **Иванов Михаил Юрьевич**, код создается автоматически.



Нажмите **Записать и закрыть** (не забывайте о сочетаниях клавиш, так быстрее)

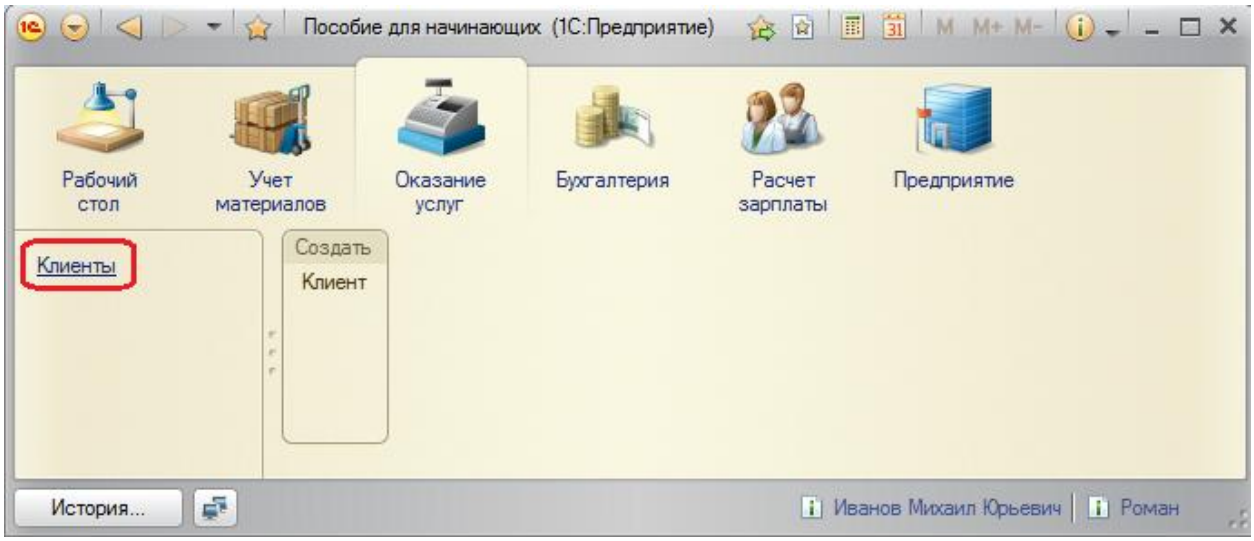
Нажав на ссылку в нижней части окна приложения, можно открыть созданный элемент.



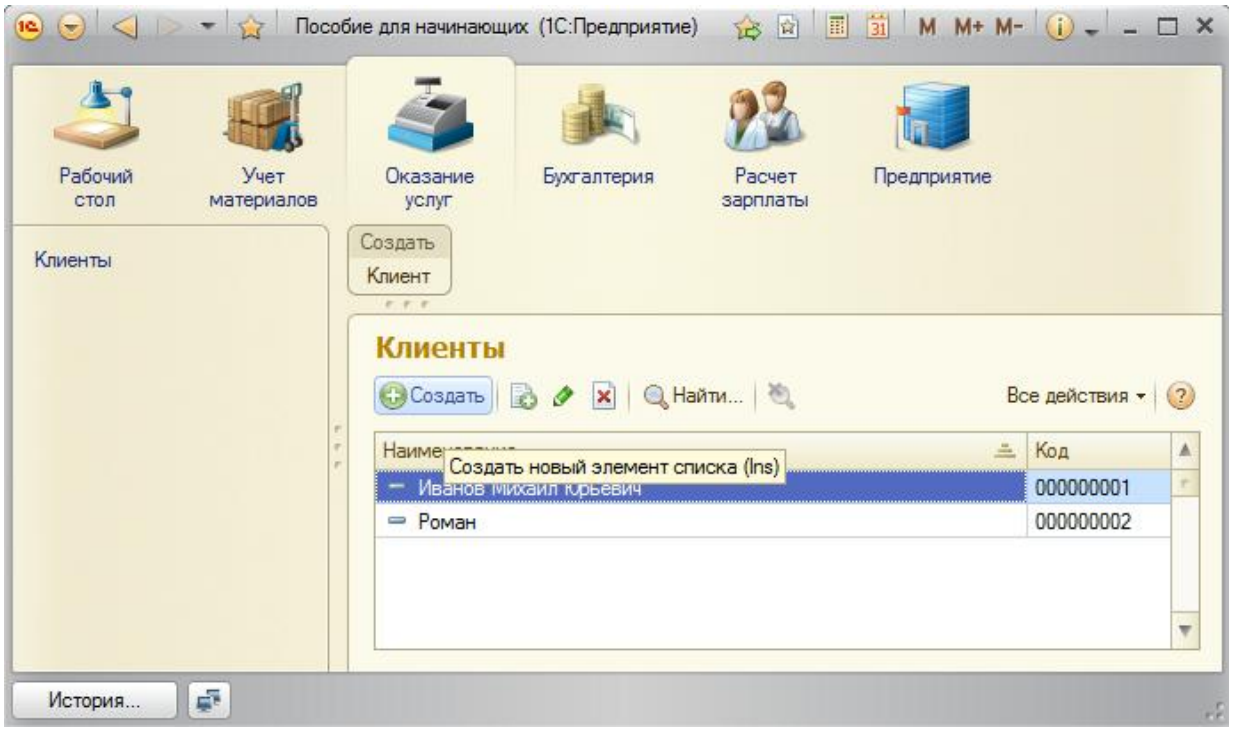
Добавьте таким же образом еще одного клиента с наименованием **Роман**.

Последнего клиента добавьте, пользуясь формой списка клиентов.

Выполните команду **Клиенты** в панели навигации раздела **Оказание услуг**. Справа от панели навигации откроется основная форма списка.



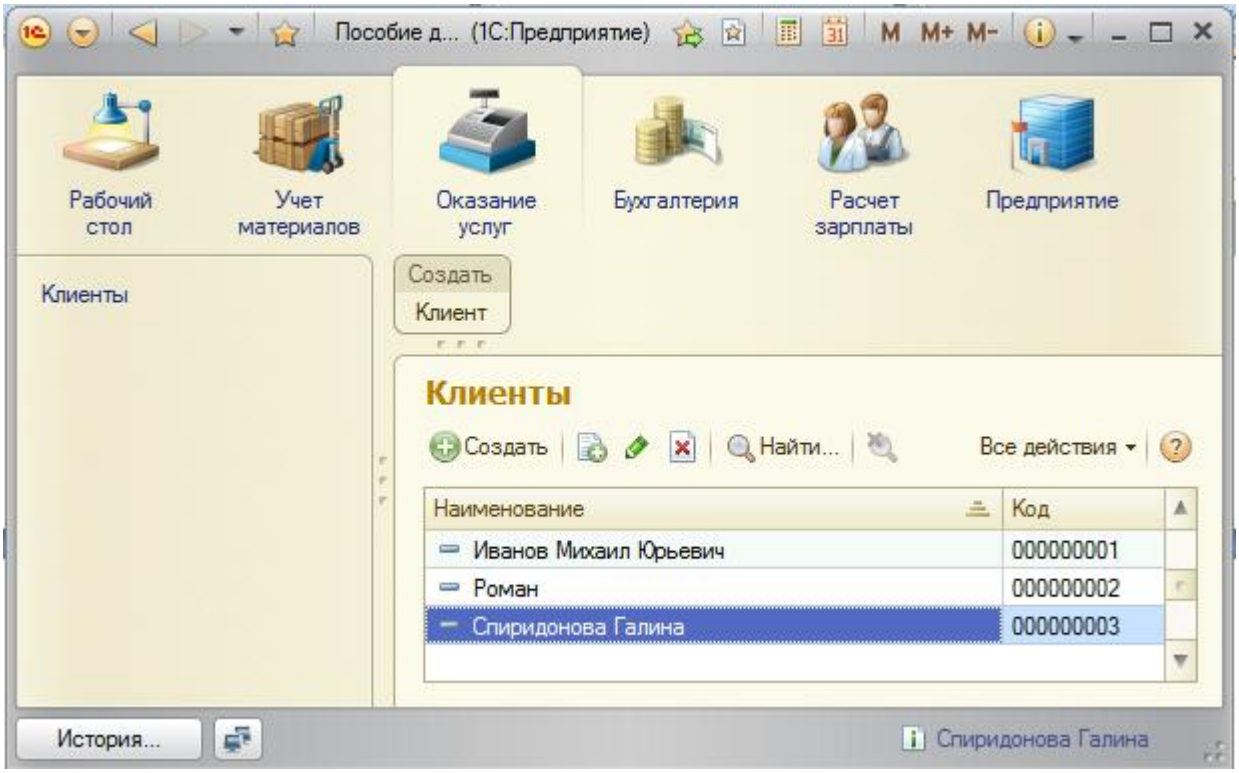
Добавьте новый элемент в справочник кнопкой **Создать.** (Если кнопка не отображается, как и название формы – Клиенты, значит Вы не ввели в свойствах справочника **Клиенты** – **Представление списка** – **Клиенты.** Закройте приложение, исправьте в конфигураторе и запустите отладку заново).



Создайте клиента с наименованием **Спиридонова Галина**.

Обратите внимание, что поле **Наименование** подсвечено красным пунктиром – это значит, что у него стоит свойство обязательного заполнения.

После добавления элементов справочник будет выглядеть следующим образом.



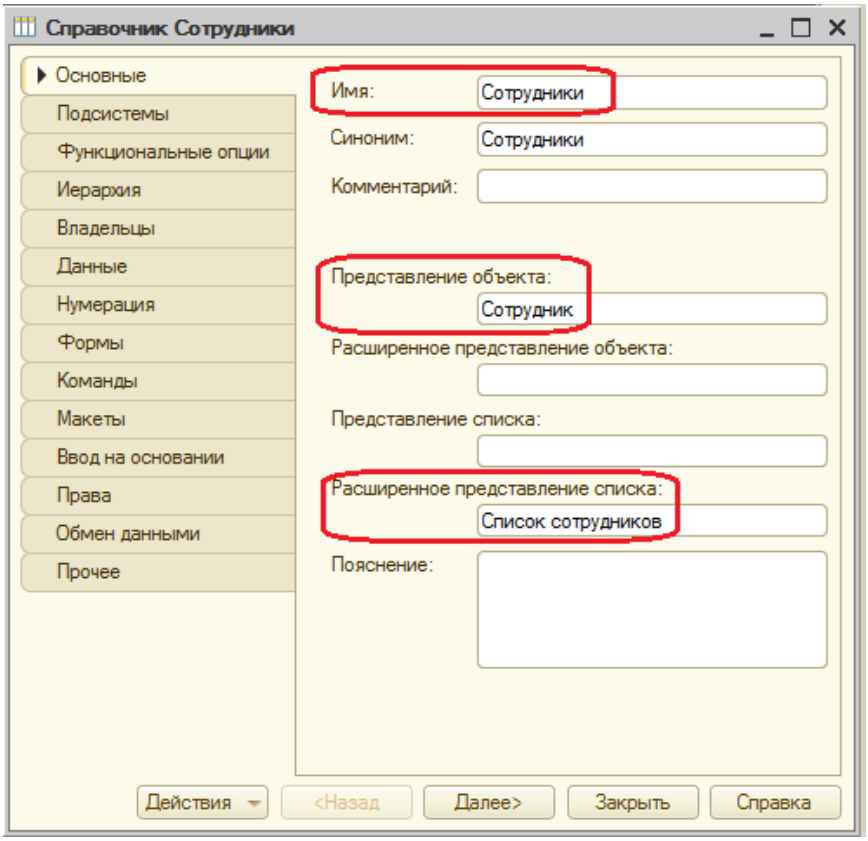
Элементы справочника редактируются двойным нажатием на него.

Справочник с табличной частью

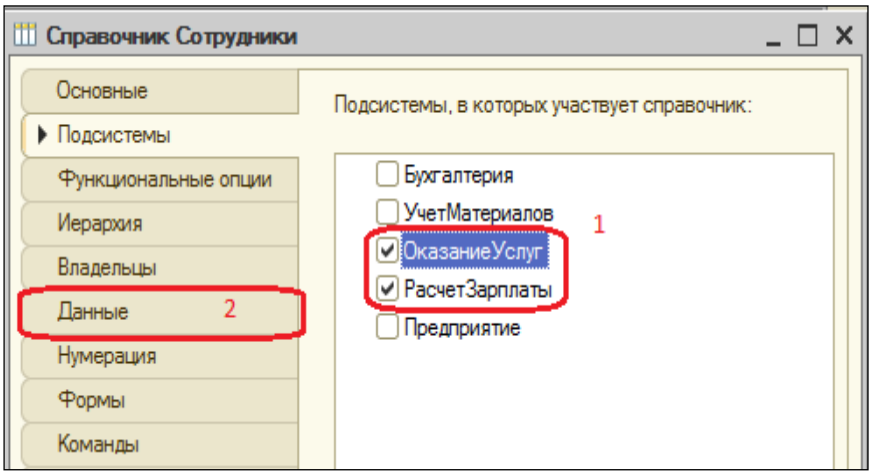
Создадим второй справочник – **Сотрудники**.

В нем будет храниться не только ФИО сотрудника, но и информация о его прошлых работах. Эта информация однородна по структуре, но разная по количеству, поэтому для ее хранения будем использовать табличную часть справочника.

Добавьте новый объект конфигурации Справочник под именем **Сотрудники**. **Представление объекта** – **Сотрудник**. Представление списка не заполняем, а **Расширенное представление списка** – **Список сотрудников**. Нажмите **Далее** на закладку **Подсистемы**, отметьте **ОказаниеУслуг** и **РасчетЗарплаты.**



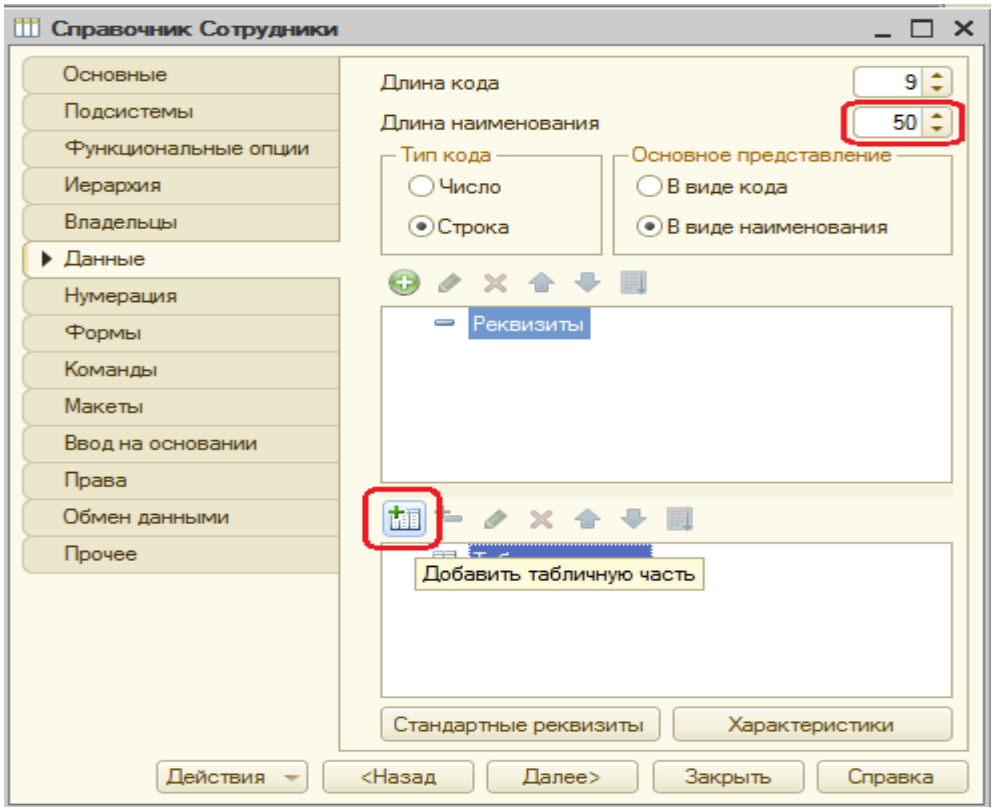
При оказании услуг должен быть указан сотрудник, оказавший эти услуги, и по результатам этой работы мы будем начислять зарплату каждому сотруднику.

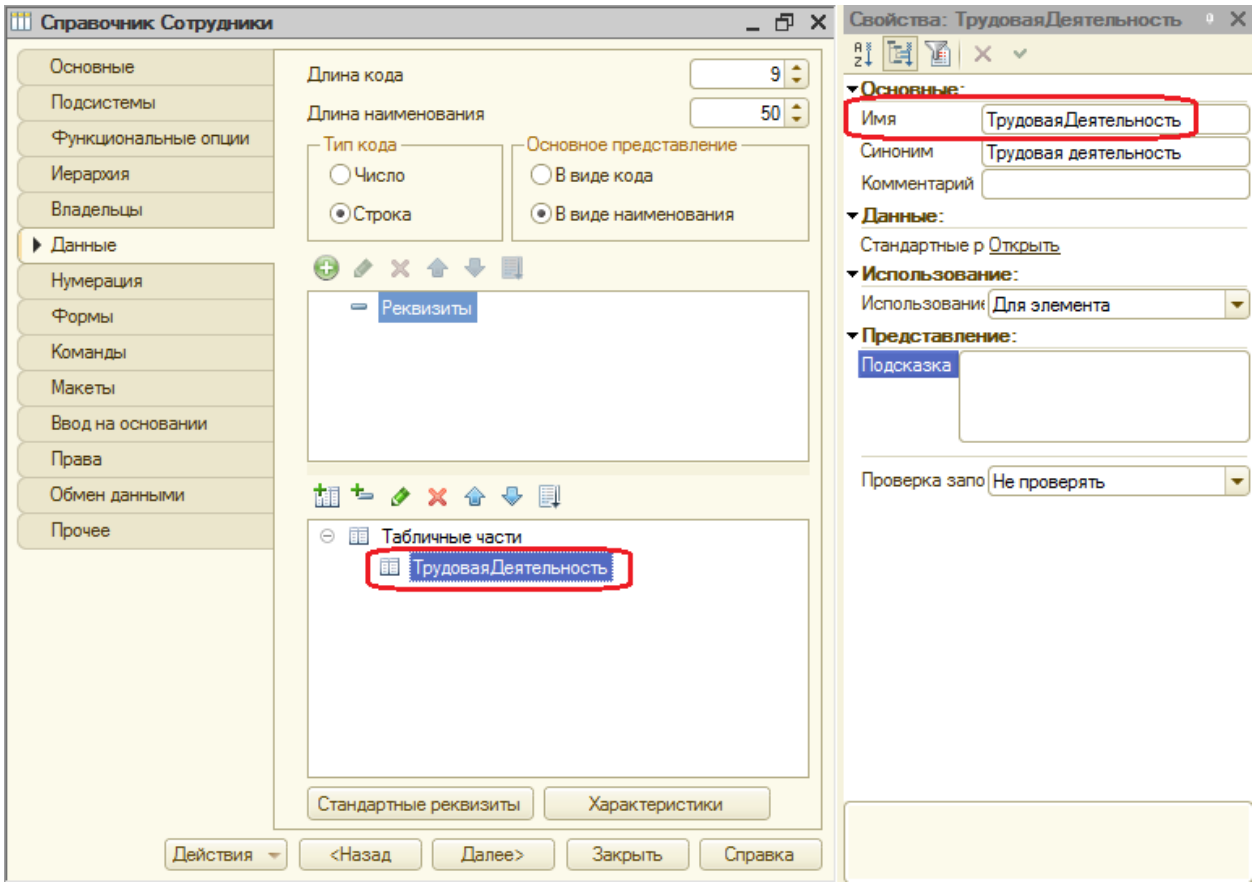


Перейдём на закладку **Данные**. Оставим по умолчанию длину и тип кода, а **длину наименования** справочника зададим равной 50 символов.

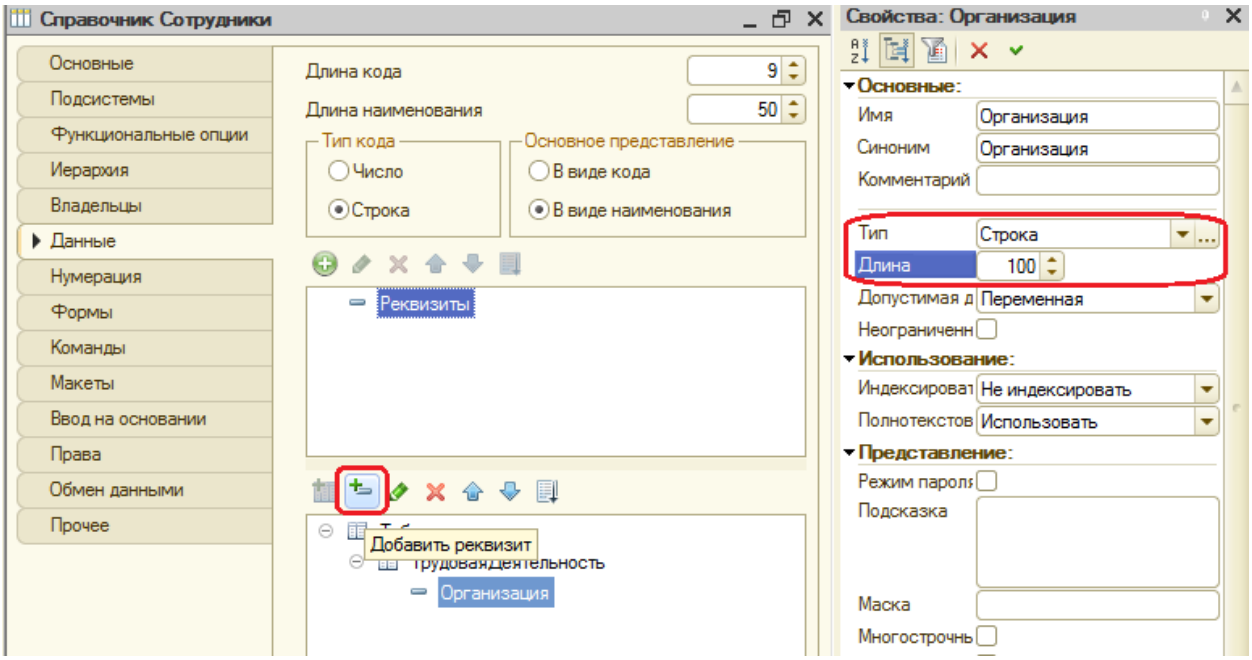
Табличная часть

Добавьте в справочник **табличную часть** с именем **ТрудоваяДеятельность**. Для этого нажмите кнопку **Добавить табличную часть** над списком табличных частей справочника.


Справочник Сотрудники
Основные
Подсистемы
Функциональные опции
Иерархия
Владельцы
Данные
Нумерация
Формы
Команды
Макеты
Ввод на основании
Права
Обмен данными
Прочее
Длина кода
9
Длина наименования
50
Тип кода
Число
Строка
Основное представление
В виде кода
В виде наименования
Реквизиты
Добавить табличную часть
Стандартные реквизиты
Характеристики
Действия
<Назад
Далее>
Закрыть
Справка

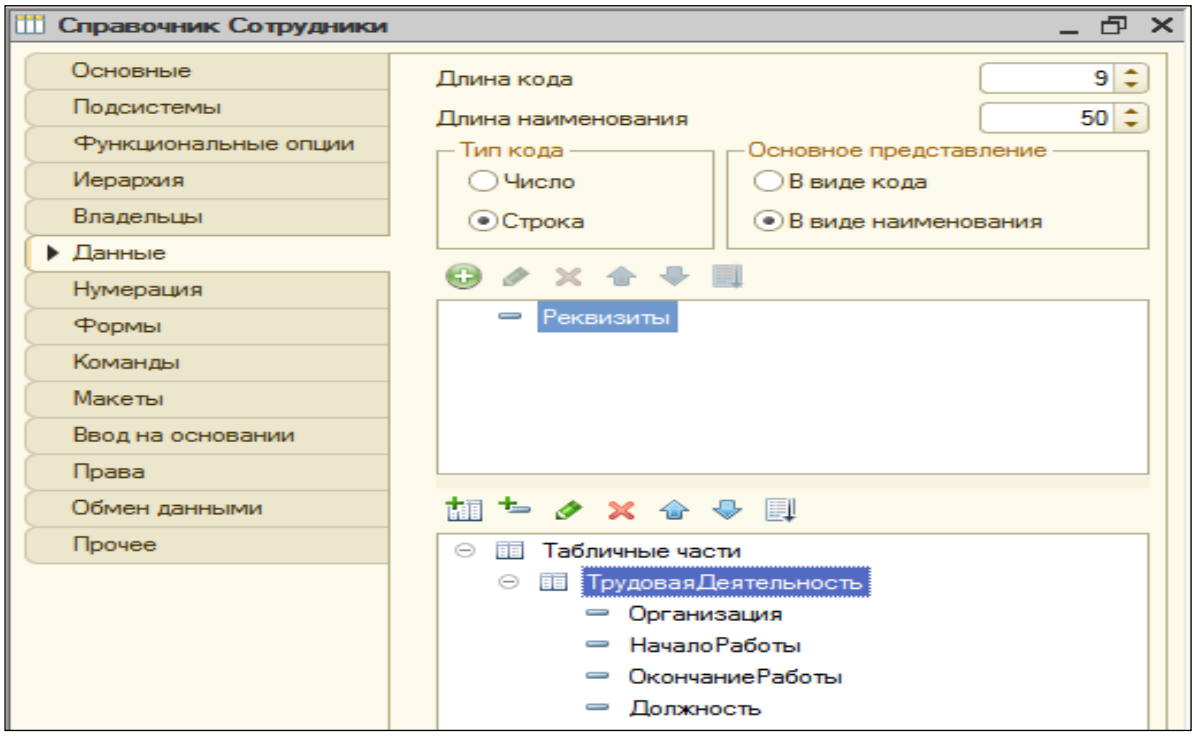

Справочник Сотрудники
Основные
Подсистемы
Функциональные опции
Иерархия
Владельцы
Данные
Нумерация
Формы
Команды
Макеты
Ввод на основании
Права
Обмен данными
Прочее
Длина кода
9
Длина наименования
50
Тип кода
Число
Строка
Основное представление
В виде кода
В виде наименования
Реквизиты
Табличные части
ТрудоваяДеятельность
Стандартные реквизиты
Характеристики
Действия
<Назад
Далее>
Закрыть
Справка
Свойства: ТрудоваяДеятельность
Основные:
Имя
ТрудоваяДеятельность
Синоним
Трудовая деятельность
Комментарий
Данные:
Стандартные р
Открыть
Использование:
Использовани
Для элемента
Представление:
Подсказка
Проверка запо
Не проверять

Создадим **реквизиты** табличной части (колонки) **ТрудоваяДеятельность**. Нажмите кнопку **Добавить реквизит** над списком табличных частей справочника.



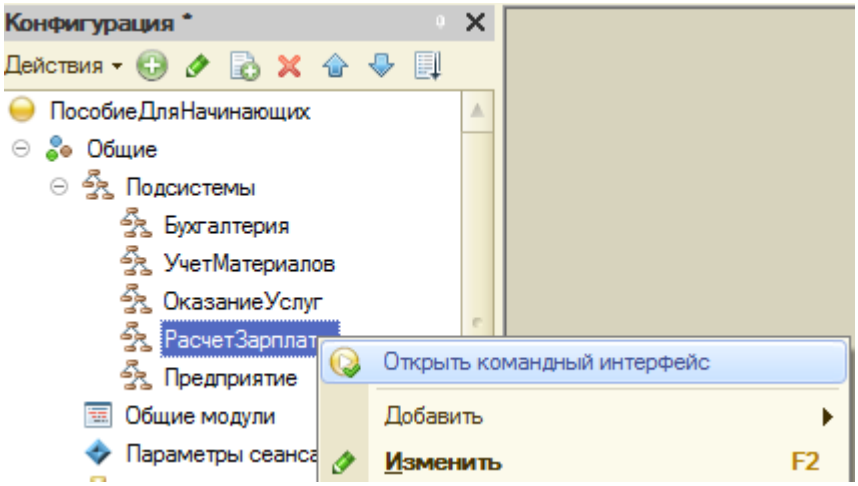
Добавьте следующие реквизиты:

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

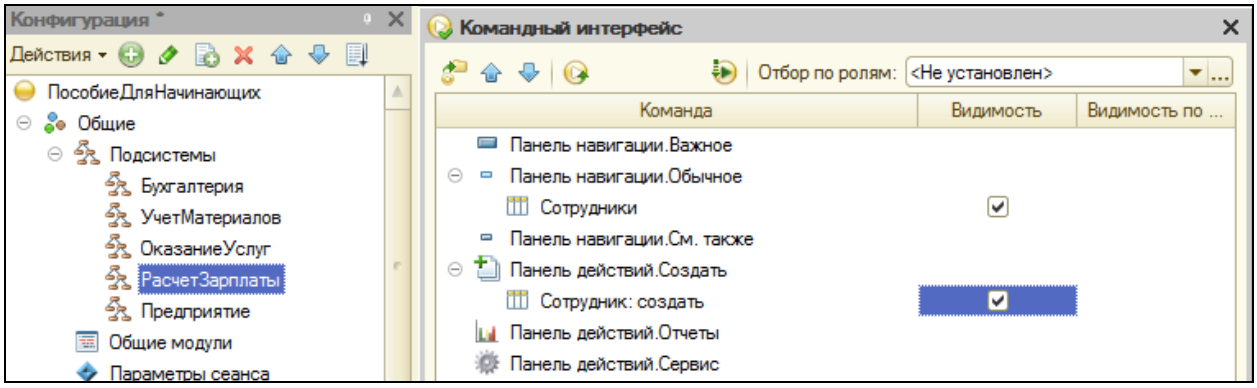


В заключении отредактируем *командный интерфейс*, чтобы было удобнее вводить новые элементы справочника. Сделаем видимой в панели действий подсистемы **РасчетЗарплаты** стандартную команду для создания новых сотрудников.

Для этого выделим подсистему **РасчетЗарплаты**, контекстное меню – **Открыть командный интерфейс**.

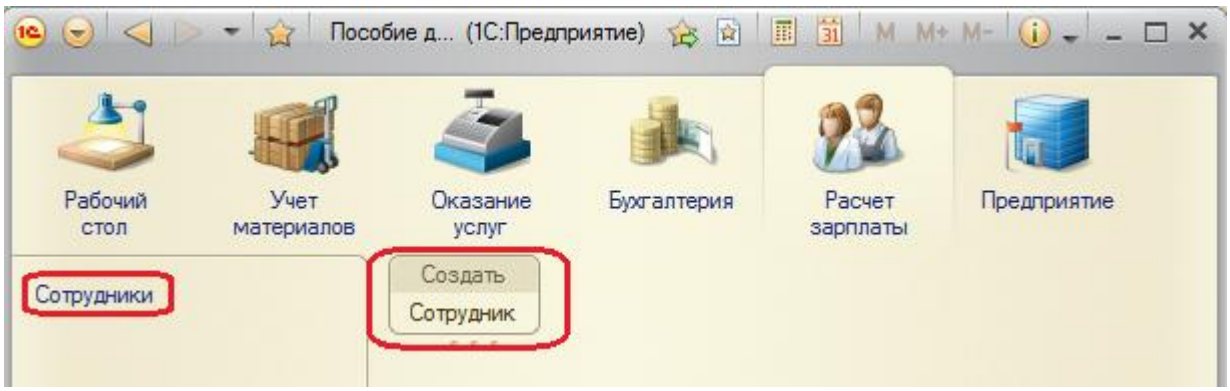


В появившемся окне ставим галочку напротив команды **Сотрудник: создать**, чтобы эта команда была видимой в пользовательском интерфейсе.



На этом создание справочника **Сотрудники** завершено. Закроем окно редактирования справочника и запустим 1С: Предприятие в режиме отладки.

В панели навигации разделов **Оказание услуг** и **Расчет зарплаты** появилась команда **Сотрудники** для открытия списка сотрудников (Название этой команды определяется *синонимом* объекта, т.к. *Представление списка* мы не задавали). А в разделе **Оказание услуг** появилась команда для создания новых сотрудников.



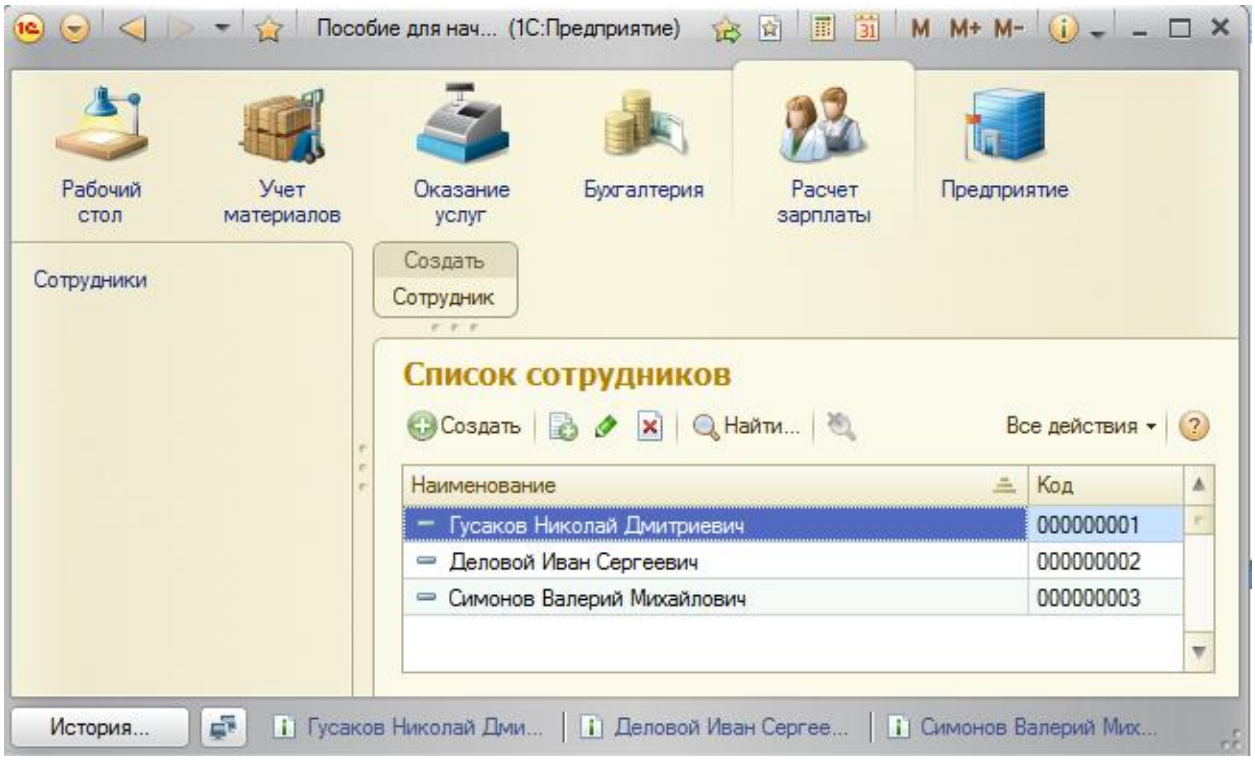
Заполнение табличной части

Выполните команду **Сотрудник**. Перед Вами откроется форма для создания элемента справочника – *основная форма объекта*. Заголовок этой формы определяется свойством справочника **Представление объекта**. Эта форма содержит табличную часть с реквизитами, которые мы описали в конфигураторе для этого справочника. Для создания элемента введите его имя (имя сотрудника), нажмите кнопку **Добавить** и далее вводите его реквизиты. Создайте следующих сотрудников.

- Гусаков Николай Дмитриевич
  - Организация – ЗАО «НТЦ»
  - Начало работы – 01.02.2000
  - Окончание работы – 16.04.2003
  - Должность – Ведущий специалист
- Деловой Иван Сергеевич
  - Организация – ООО «Автоматизация»
  - Начало работы – 22.01.1996
  - Окончание работы – 31.12.2002
  - Должность – Инженер
  - Организация – ЗАО «НПО СпецСвязь»
  - Начало работы – 20.06.1986
  - Окончание работы – 21.01.1995
  - Должность – Начальник производства
- Симонов Валерий Михайлович

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

Чтобы посмотреть список созданных сотрудников, воспользуйтесь командой **Сотрудники** в левой части окна.



Заголовок этой формы определяется свойством **Расширенное представление списка**.

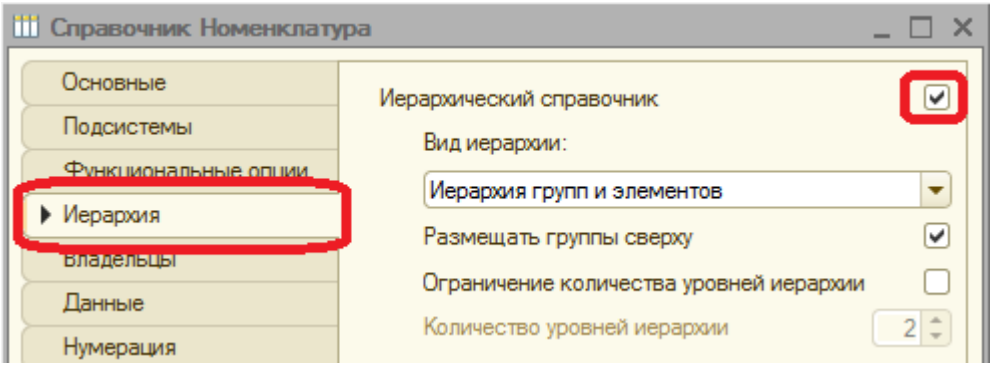
Иерархический справочник

Создадим справочник **Номенклатура**. Он будет содержать информацию об услугах, которые оказывает наша фирма и материалах, которые при этом могут быть использованы. Они будут логически собраны в группы.

В режиме конфигуратора создайте новый объект Справочник и назовите его Номенклатура. Т.к. понятие номенклатура не имеет единственного числа, никаких представлений задавать не будем.

Перейдите на вкладку Подсистемы. Задайте доступность справочника в разделах **Бухгалтерия**, **УчетМатериалов** и **ОказаниеУслуг.**

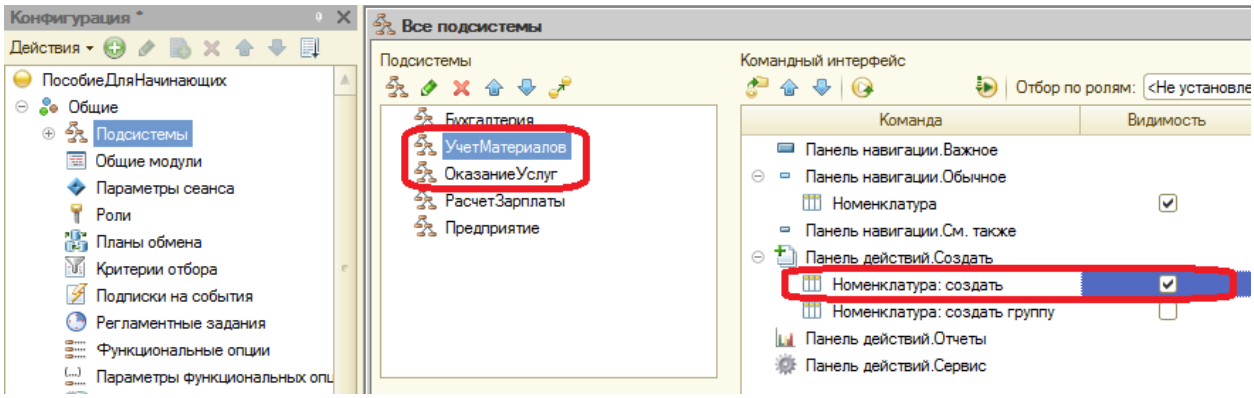
Перейдите на закладку **Иерархия** и установите флажок **Иерархический справочник**.



На закладке **Данные** установите длину наименования 100 символов.

Закройте свойства справочника. Как обычно, настроим интерфейс приложения, включив команду создания новых элементов списка номенклатуры в подсистемах **УчетМатериалов** и **ОказаниеУслуг**.

Для этого выделите ветвь **Подсистемы**, через контекстное меню выберите **Все подсистемы**. В открывшемся окне слева выделите **УчетМатериалов**. Справа отразятся все команды выбранной подсистемы. В группе **Панель действий. Создать** включите видимость у команды **Номенклатура: создать**.



Затем выделите подсистему **ОказаниеУслуг** и проделайте тоже самое.

Теперь заполним справочник Номенклатура. Попутно покажем, как создавать группы и переносить элементы из одной группы в другую.

Закройте окно редактирования справочника и запустите 1С: Предприятие в режиме отладки.

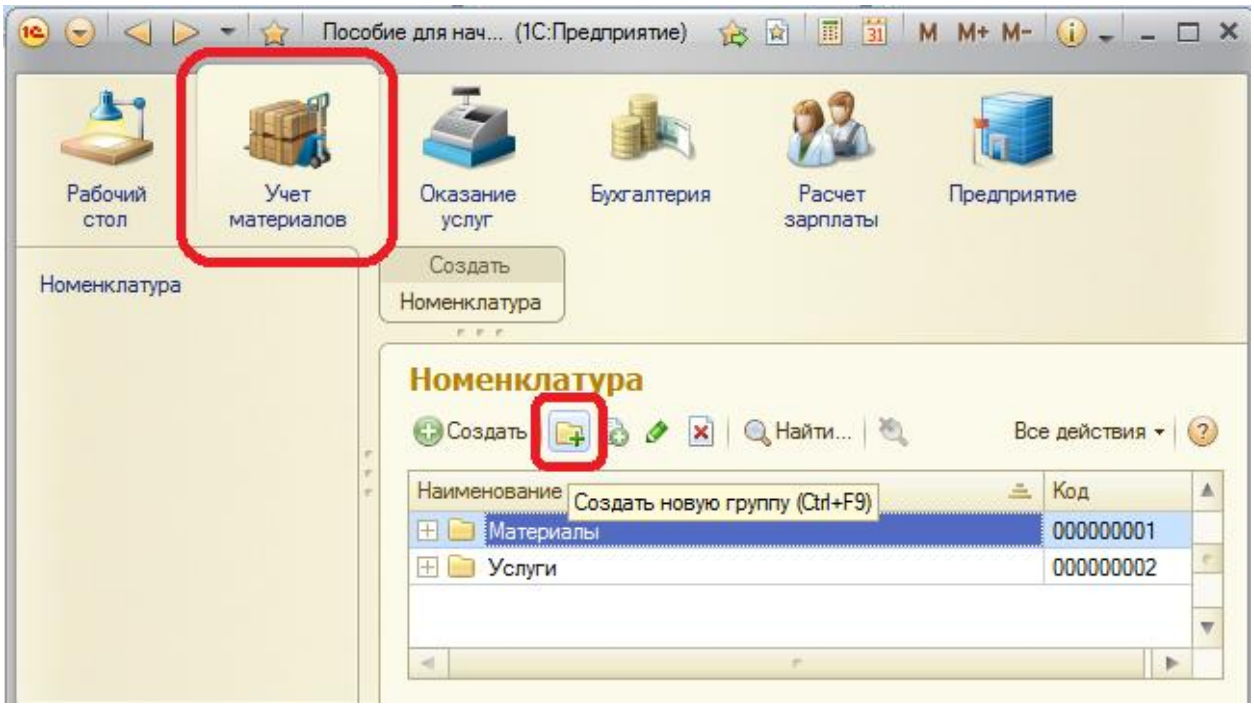
В режиме 1С: Предприятие

В открывшемся окне видно, что в панели навигации разделов **Учет материалов**, **Оказание услуг** и **Бухгалтерия** появилась команда **Номенклатура** (Название этой команды определяется синонимом объекта, т.к. других представлений Вы не задавали).

Выполните команду **Номенклатура** в панели навигации раздела **Учет материалов**. Справа от панели навигации откроется *основная форма списка*.

Создание элементов в иерархическом справочнике

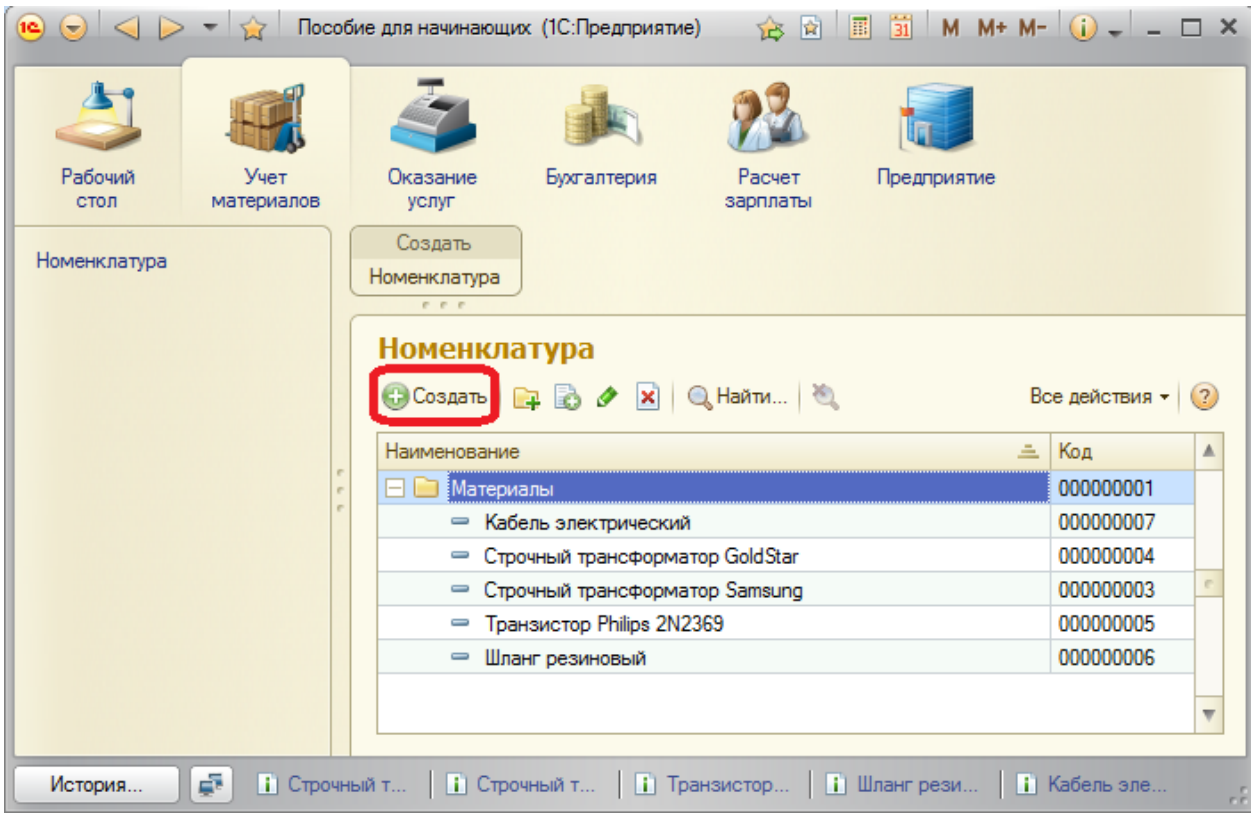
Создайте две группы в корне справочника: **Материалы** и **Услуги** с помощью кнопки **Создать группу** в командной панели формы списка.



Поля **Родитель** и **Код** заполнять не будем.

Затем раскроем группу **Материалы**, нажав на крестик слева от нее. И создадим в ней 5 элементов кнопкой Создать:

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



Если новый элемент добавляется из формы списка в некоторую открытую группу, то система автоматически подставляет в качестве родителя эту группу. В данном случае родитель – **Материалы**.

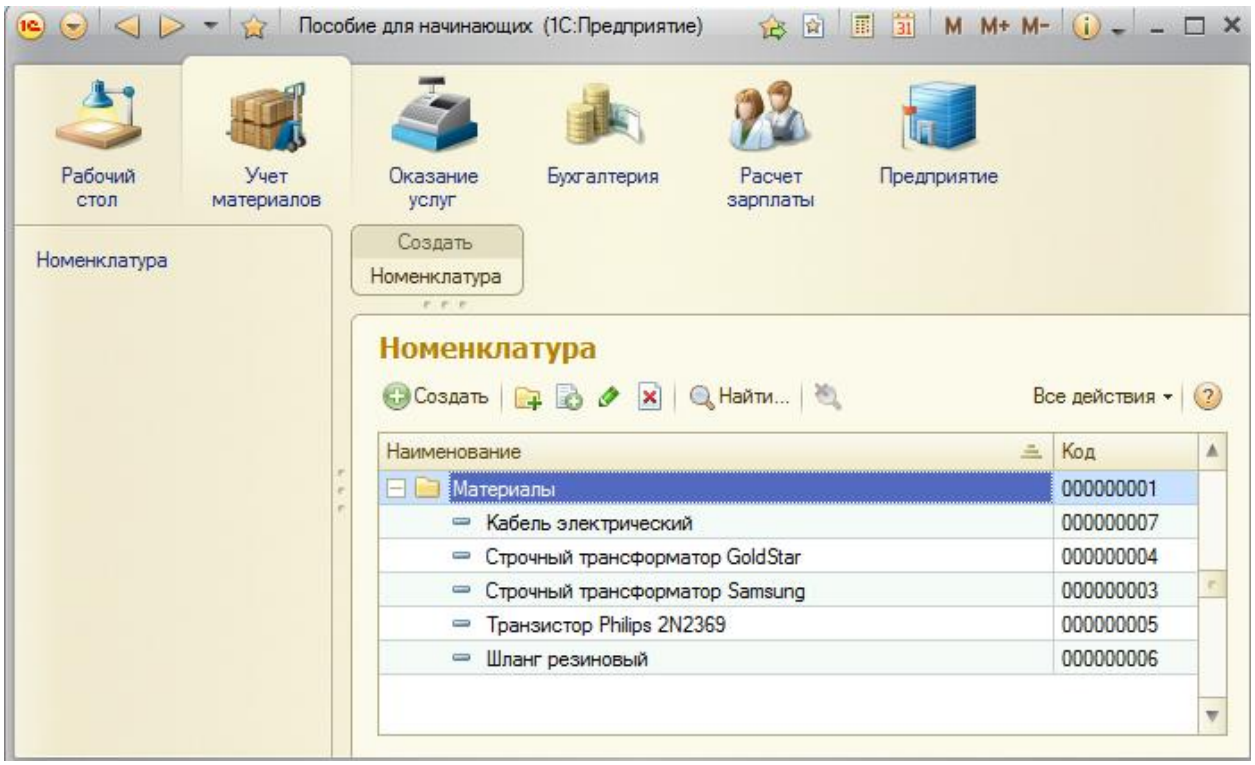
Если элемент добавляется командой **Номенклатура: Создать**, то родителя придется устанавливать вручную.

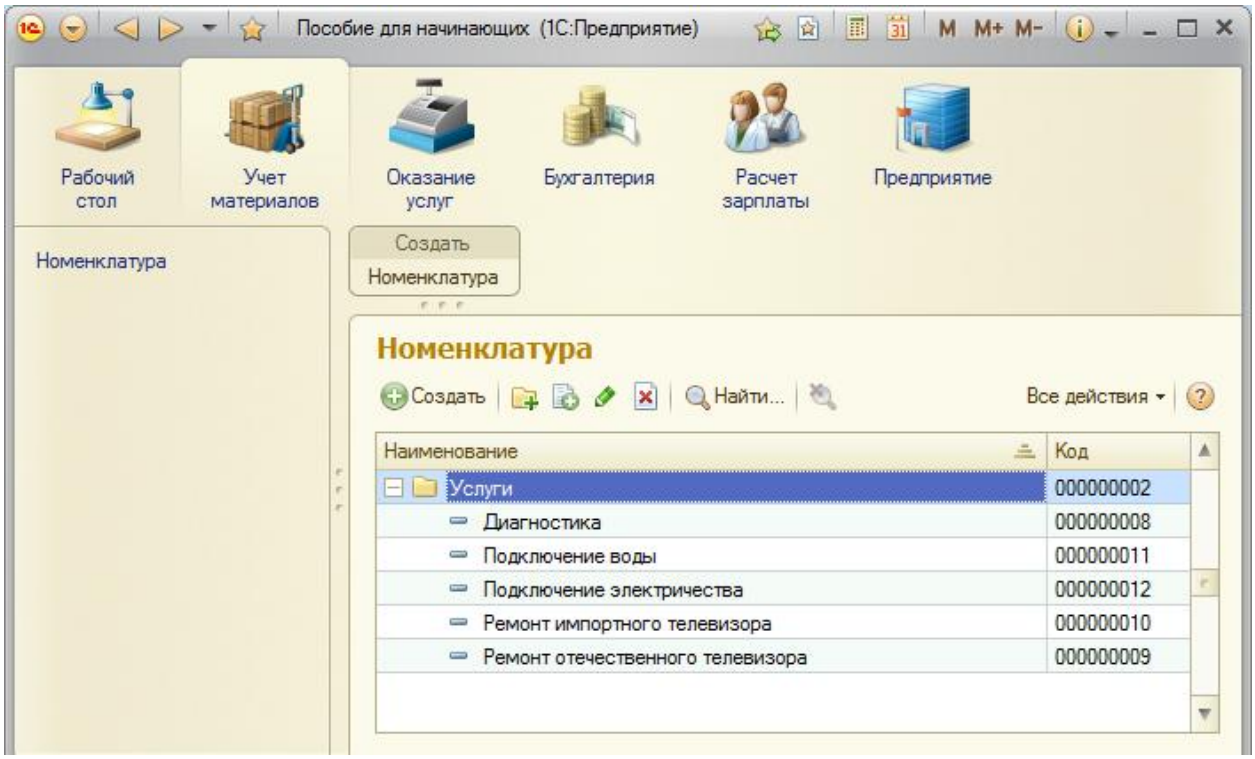
Раскройте группу **Услуги** и создайте в ней услуги по ремонту телевизоров:

- Диагностика
- Ремонт отечественного телевизора
- Ремонт импортного телевизора

И услуги по установке стиральных машин:

- Подключение воды
- Подключение электричества





Перенос элементов в другие группы

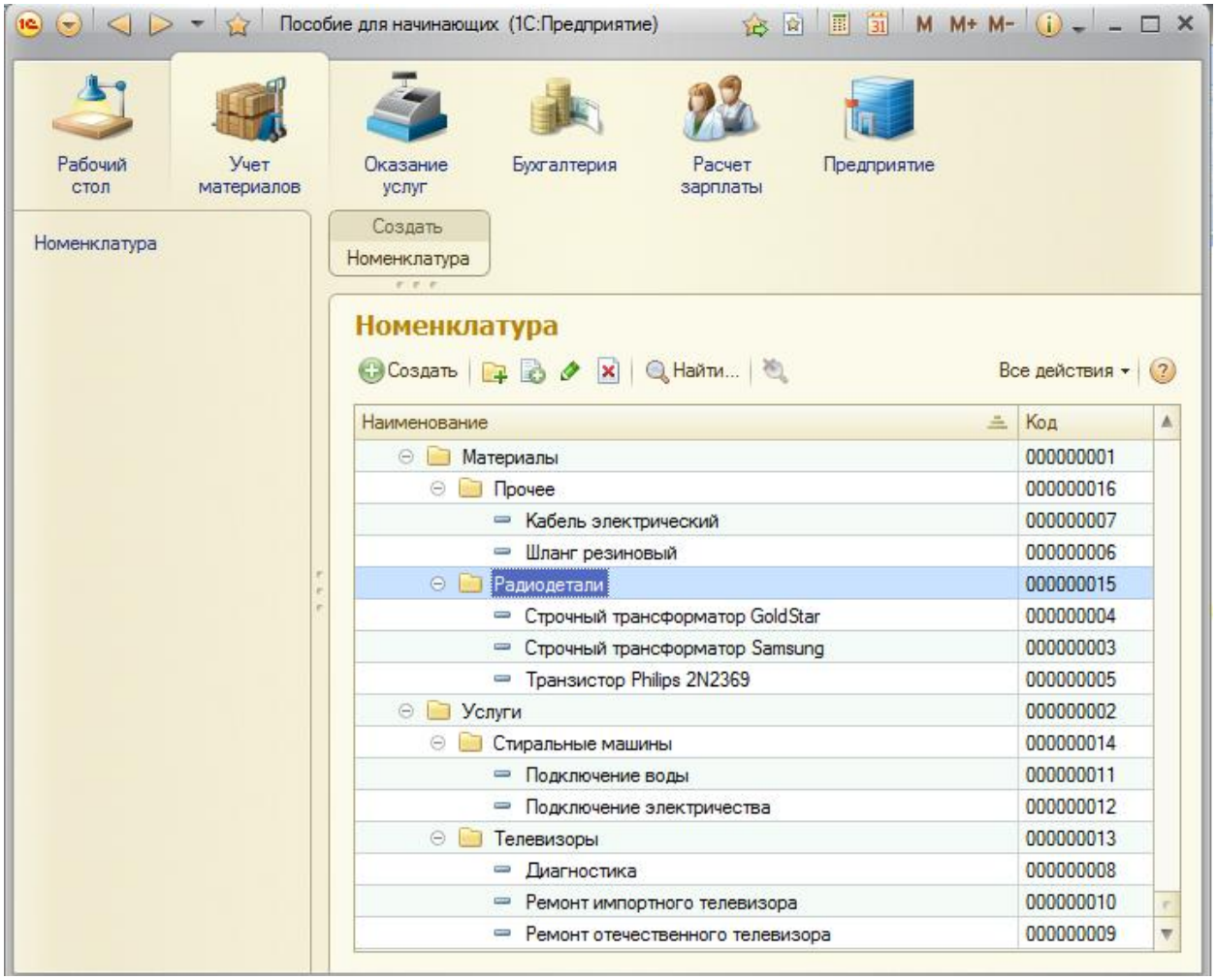
Теперь распределим услуги по двум смысловым группам: услуги по ремонту телевизоров и услуги по установке стиральных машин. Для этого в группе **Услуги** создайте еще две группы: **Телевизоры** и **Стиральные машины**. Есть несколько способов распределения элементов в группы. Первый – ставите курсор на услугу, которую хотите

перетащить в другую группу, кнопка **Все действия – Переместить в группу**. Можно переместить сразу несколько элементов, выделив их с помощью **Ctrl.** Второй способ – выбрать услугу двойным щелчком и установить в поле **Родитель** нужную группу. Третий способ – самый простой – выбрать услуги мышью и ею перетащить в нужную группу.

Переместите услуги **Диагностика**, **Ремонт отечественного телевизора**, **ремонт импортного телевизора** в группу **Телевизоры**, остальное – в группу **Стиральные машины**.

Создайте в группе **Материалы** две группы: **Радиодетали** и **Прочее**. **Кабель электрический** и **Шланг резиновый** поместите в группу **Прочее**. Остальное – в группу **Радиодетали**.

Переключите представление списка в виде дерева – **Все действия – Режим просмотра – Дерево**. Получится такая картина:



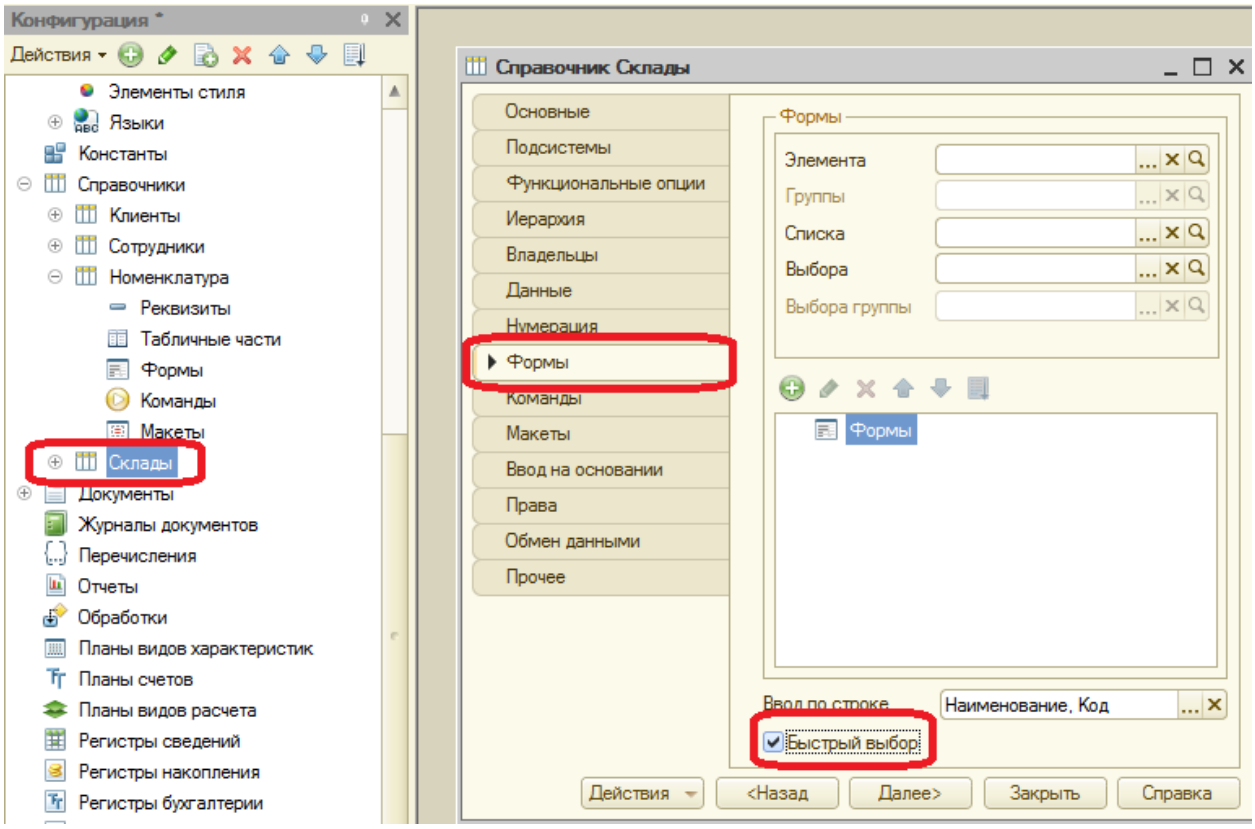
Закройте приложение и вернитесь в конфигуратор.

Справочник с предопределенными элементами

Создадим справочник **Склады**, который будет включать в себя предопределенный элемент – склад **Основной**, на который будут поступать все материалы.

Создайте новый объект конфигурации Справочник с именем Склады.

Представление объекта – **Склад**. На вкладке **Подсистемы** отметьте **ОказаниеУслуг** и **УчетМатериалов**. На вкладке **Формы** установите флажок **Быстрый выбор**.

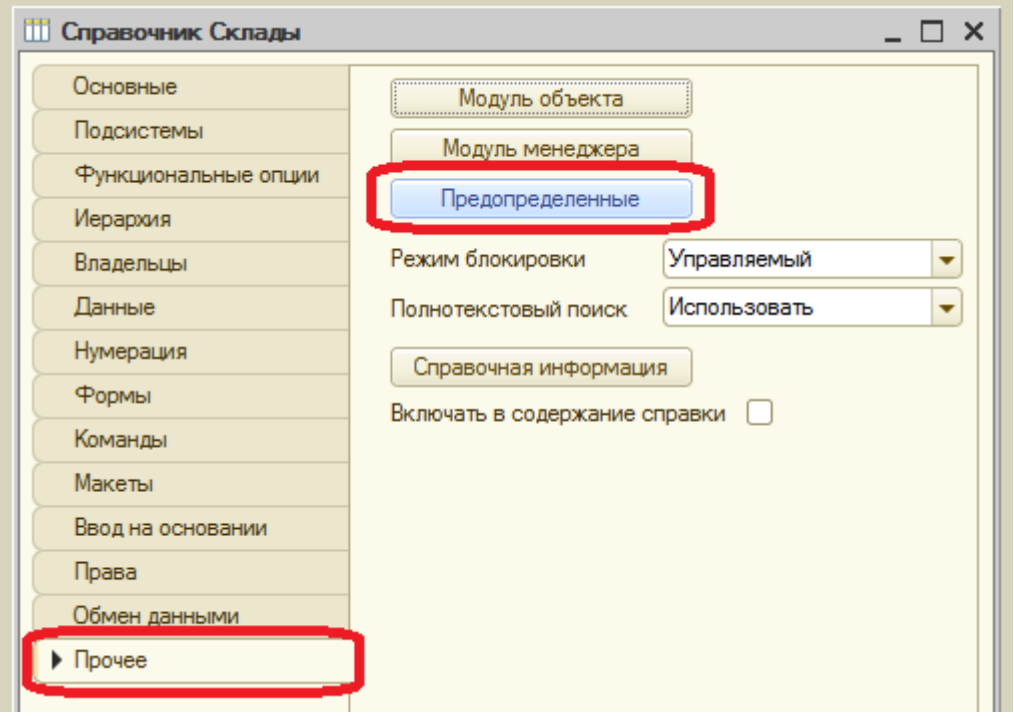


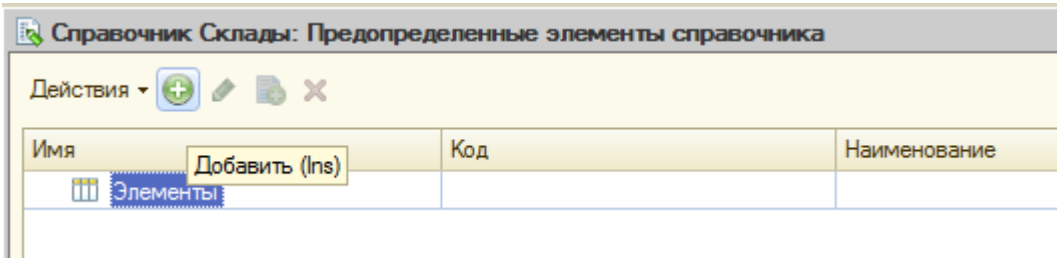
Свойство **Быстрый выбор** позволяет выбирать элементы не из отдельной формы, а из небольшого выпадающего списка, заполненного элементами этого справочника. Актуально, если элементов немного.

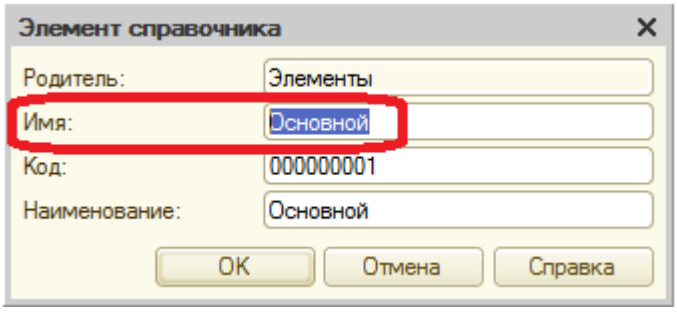
(Для остальных справочников свойство Быстрый выбор мы не устанавливаем, потому что Номенклатура – иерархический справочник и быстрый выбор не имеет для него смысла, а список сотрудников и список клиентов может быть очень большим - будет неудобно прокручивать выпадающий список.)

Предопределенные элементы

Перейдите на закладку **Прочее** и нажмите кнопку **Предопределенные**. Появится список предопределенных элементов справочника. Нажмите кнопку **Добавить** и задайте имя **Основной**.

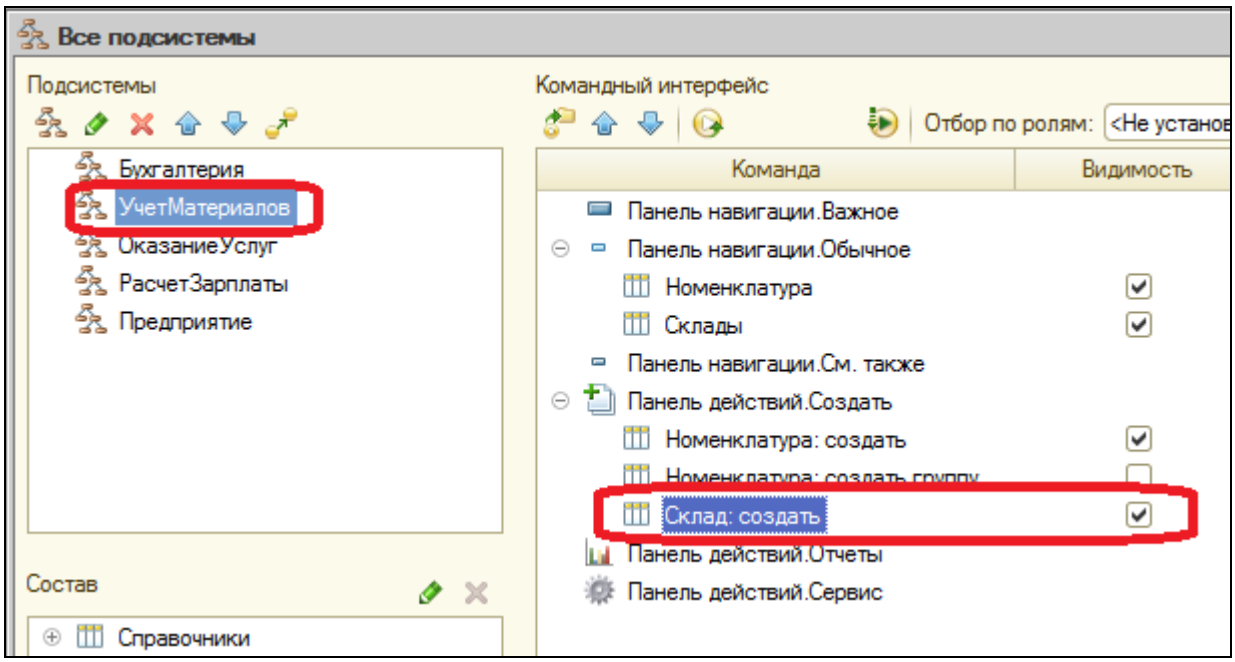






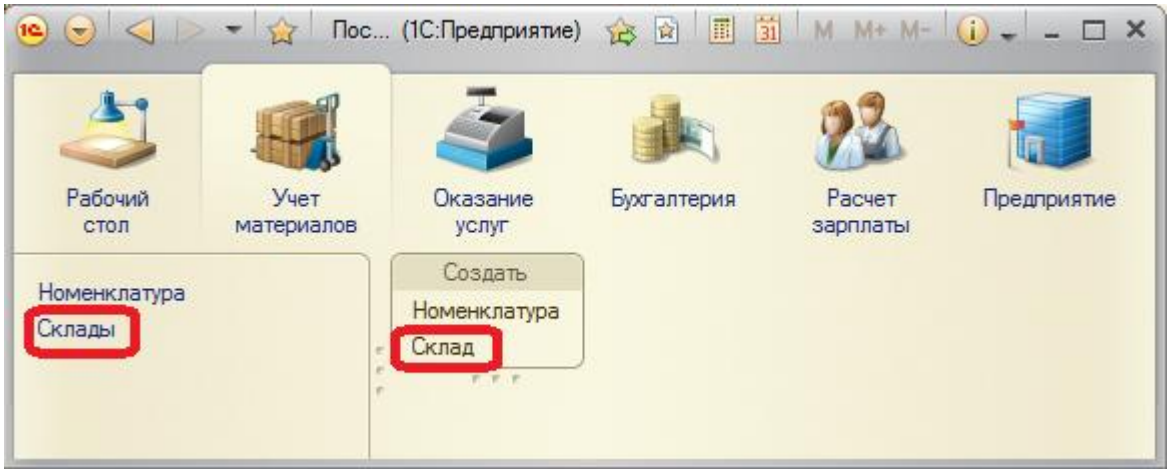
Имя служит для обращения к нему напрямую через встроенный язык. **Наименование** видит и изменяет пользователь, имя – пользователь не видит и изменить не может.

Настройте интерфейс для удобства ввода новых элементов справочника. Выделите **Подсистемы**, контекстное меню – **Все подсистемы**. Выделите в списке слева **УчетМатериалов** и включите видимость команды **Склад:Создать**.

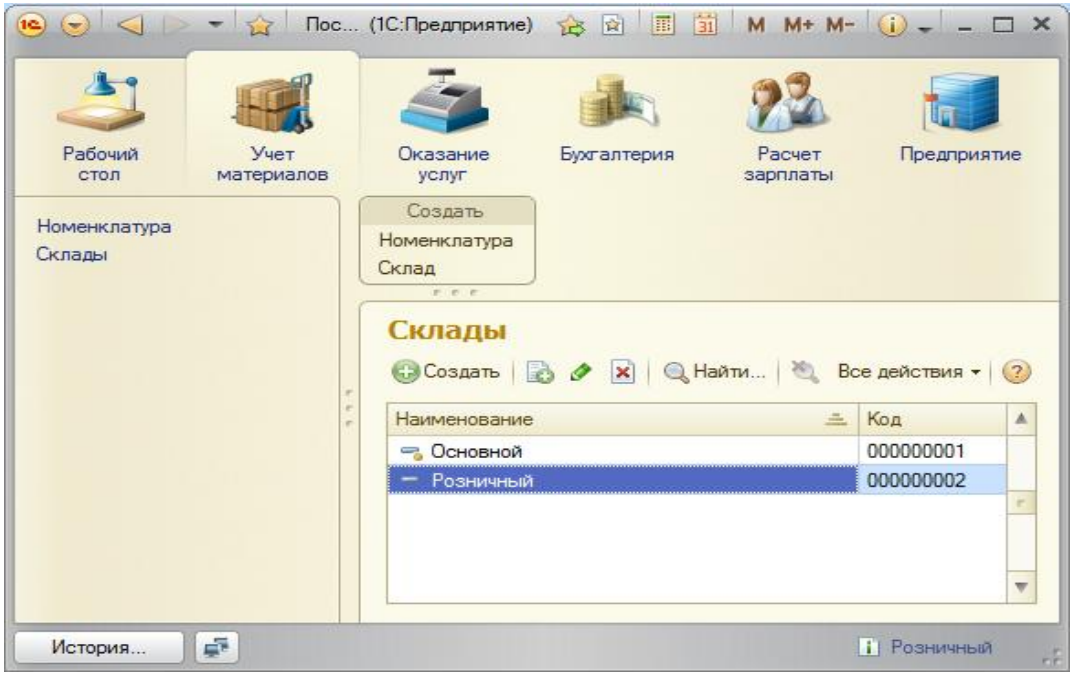


Запустите 1С: Предприятие в режиме отладки.

В открывшемся окне видно, что в панели действий раздела **Учет материалов** появилась команда **Склад** для создания новых складов, а в панели навигации разделов **Учет материалов** и **Оказание услуг** команда **Склады**.



Выполните команду **Склады** в панели навигации раздела **Учет материалов**. Справа от панели навигации откроется основная форма списка. В списке складов уже есть один элемент с наименованием Основной. Это **предопределенный** элемент, который мы недавно задали в конфигураторе, обозначается специальным значком. Добавьте еще один склад, назвав его **Розничный**.



Контрольные вопросы

- ✓ Для чего предназначен объект конфигурации Справочник
- ✓ Каковы характерные особенности Справочника
- ✓ Для чего используются реквизиты и табличные части справочника
- ✓ Зачем нужны иерархические справочники и что такое родитель
- ✓ Что такое предопределенные элементы
- ✓ Чем с точки зрения конфигурации отличаются обычные элементы справочника от предопределенных элементов
- ✓ Как создать объект Справочник и описать его структуру
- ✓ Как добавить новые элементы в Справочник
- ✓ Как создать группу Справочника
- ✓ Как переместить элементы из одной группы Справочника в другую
- ✓ Зачем нужна проверка заполнения у реквизитов справочника
- ✓ Что такое быстрый выбор и как его использовать
- ✓ Как отобразить команды создания нового элемента справочника в интерфейсе подсистем
- ✓ Как редактировать командный интерфейс подсистем

Практическая работа № 4

Документы (1:30)

В этой работе Вы познакомитесь с объектом **Документ**. Узнаете для чего он нужен, его структуру и свойства. Создадите несколько документов и зададите собственные последовательности действий, связанных с работой документа. Также узнаете как создать форму документа и познакомитесь с некоторыми конструкциями встроенного языка.

Объект конфигурации **Документ** предназначен для описания информации о совершенных хозяйственных операциях или о событиях в организации. Свойства и структура реальных документов описываются в объектах **Документ**, на основе которых платформа создает в базе данных таблицы для хранения информации из этих документов.

Документ обладает способностью **проведения**. Факт проведения документа означает, что событие, которое он отражает, повлияло на состояние учёта. До тех пор, пока документ не проведен, состояние учета неизменно, и документ – не более чем черновик, заготовка.

Документ всегда привязан к конкретному времени, что позволяет отражать в базе данных фактическую последовательность событий.

Система 1С: Предприятие умеет отслеживать правильность учёта. Пользователь может сам создавать необходимые документы – приходные и расходные накладные, счета, приказы и т.п.

Каждый документ может содержать дополнительную информацию, которая подробнее его описывает – **реквизиты**. Всегда существуют стандартные реквизиты – **Дата** и **Номер**. Также Документ может содержать **табличную часть** для описания набора данных. Для визуализации документа существует несколько основных **форм**.

Типообразующие объекты конфигурации

Когда мы создавали реквизиты справочников или табличных частей, мы всегда указывали тип значения, которое может принимать этот реквизит. Это были примитивные типы данных – Число, Строка, Дата, Булево. Примитивные типы данных изначально определены в системе и их набор ограничен. В любой конфигурации могут существовать типы данных, определяемые только конкретной конфигурацией, т.е. появляющиеся в результате добавления некоторых объектов.

Например, после создания объекта *Справочник* **Склады**, сразу же появились новые типы данных, связанные с этим справочником. Среди них **СправочникСсылкаСклады**. Если мы укажем какому-либо реквизиту этот тип данных, то сможем хранить в нем ссылку на конкретный объект справочника **Склады**.

Документ «Приходная накладная»

Создадим несколько документов, чтобы иметь возможность фиксировать события, происходящие в нашей фирме.

Наша фирма ремонтирует телевизоры и устанавливает стиральные машины. В обоих случаях требуются некоторые расходуемые в процессе оказания услуг материалы, поэтому двумя событиями в хозяйственной деятельности фирмы будут поступление материалов и оказание услуг.

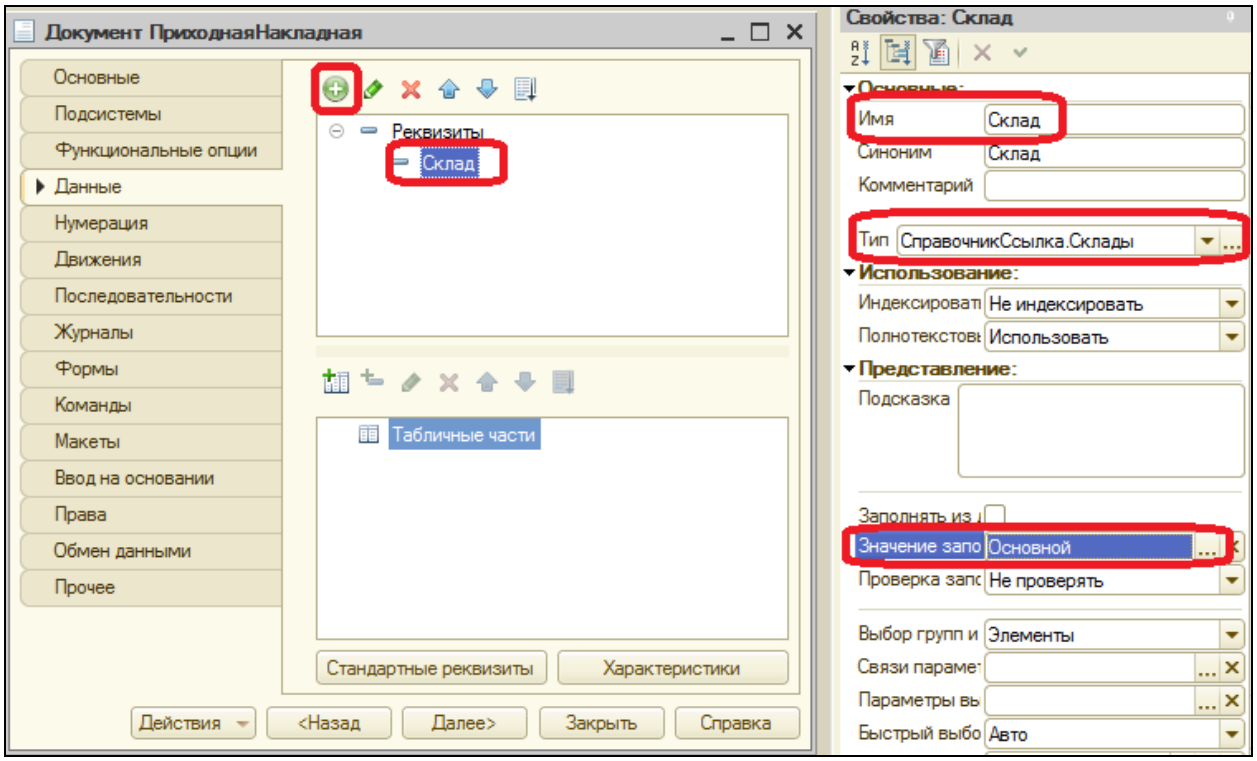
Для отражения этих событий в базе данных создадим два документа: **Приходная накладная** и **Оказание услуги**. Первый будет фиксировать факты поступления в фирму необходимых материалов, а второй – фиксировать оказание услуг и расход материалов, которые используются при оказании этих услуг.

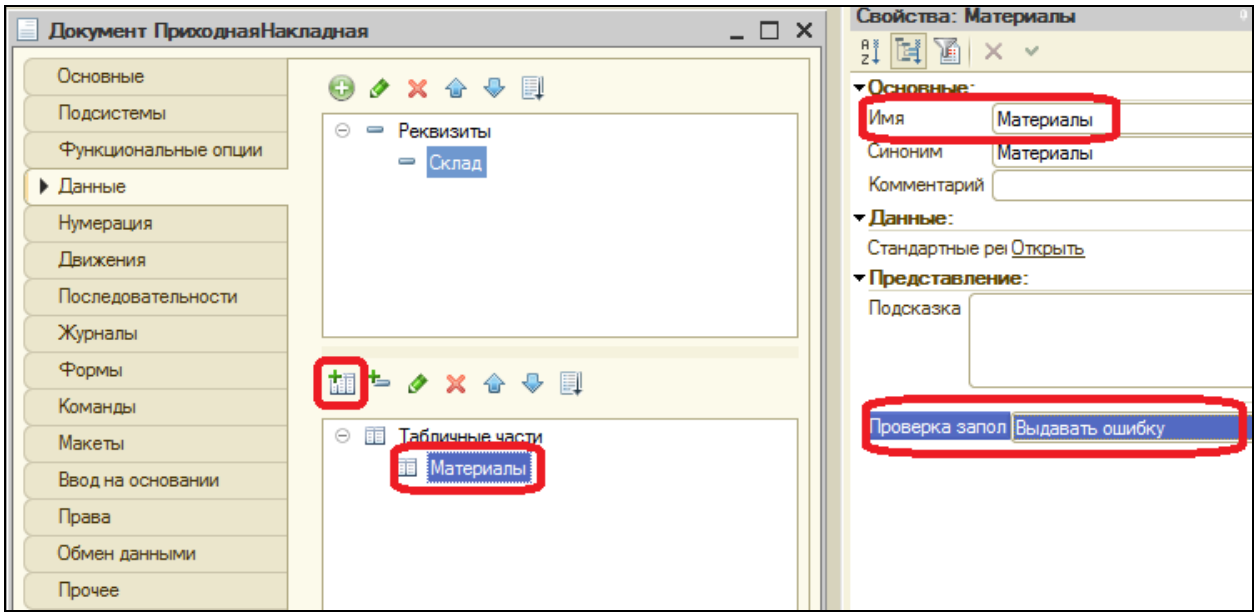
Откройте конфигуратор и добавьте новый объект – **Документ**. Задайте имя – **ПриходнаяНакладная**. **Представление объекта** не задаём, **Представление списка** задайте **Приходные накладные**. Перейдите на вкладку **Подсистемы** и отметьте **УчетМатериалов** и **Бухгалтерия**. Перейдите на вкладку Данные и создайте реквизит документа с именем Склад с помощью кнопки **Добавить.** Выберите для реквизита ссылочный тип данных **СправочникСсылка.Склады**. Этот тип стал доступен в конфигурации после создания справочника **Склады**.

Работа в нашей фирме организована так, что почти всегда поступающие товары идут на основной склад, поэтому установим для созданного реквизита свойство автозаполнения. В свойствах реквизита найдите **Значение заполнения** и выберите в нем **Основной**. Таким образом, при создании нового документа склад будет сразу заполняться значением Основной и пользователю не придется делать это вручную, что удобно.

После этого добавим в документ табличную часть с именем **Материалы**. Для этого нажмите кнопку **Добавить табличную часть** над списком табличных частей документа. Кроме имени табличной части установите свойство **Проверка заполнения** в значение **Выдавать ошибку**. Тем самым Вы задаете условие, что документ Приходная накладная

обязательно должен содержать список приходуемых материалов, иначе будет выдано сообщение об ошибке и документ не будет сохранен.



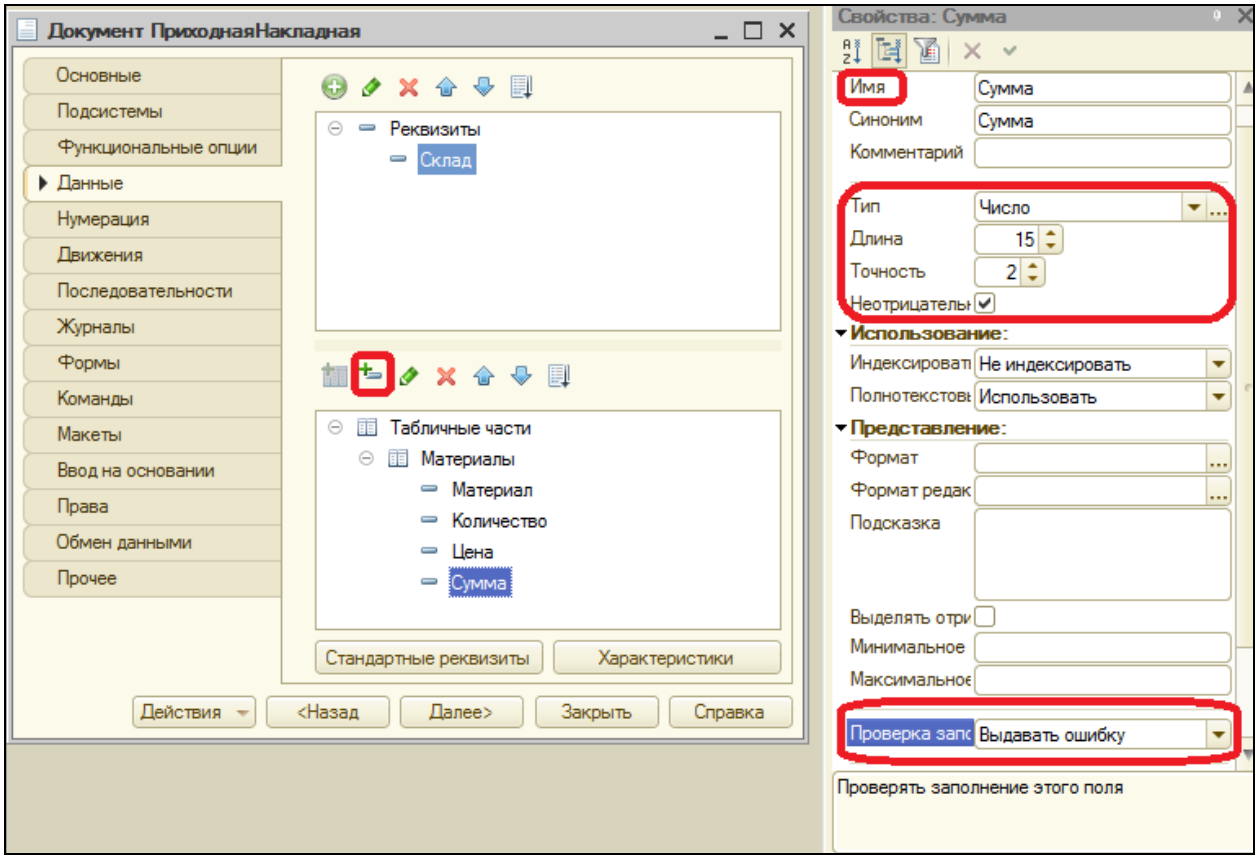


Создайте реквизиты табличной части **Материалы**:

- **Материал**, тип СправочникСсылка.Номенклатура;
- **Количество**, тип **Число**, длина 15, точность 2 знака после запятой, неотрицательное;
- **Цена**, тип **Число**, длина 15, точность 2, неотрицательное;

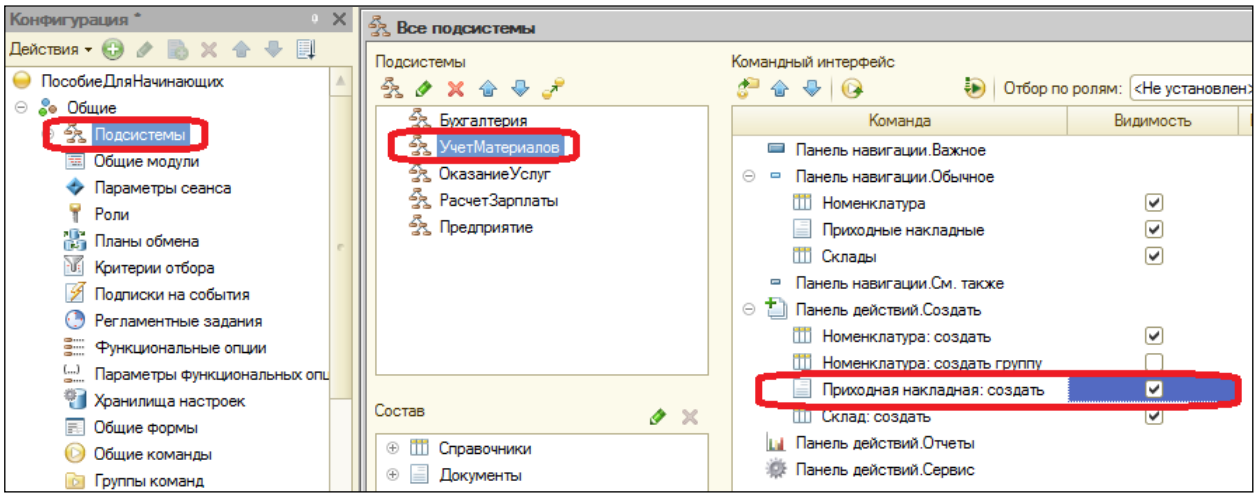
- **Сумма**, тип **Число**, длина 15, точность 2, неотрицательное.

Для каждого реквизита также задайте свойство **Проверка заполнения** в **Выдавать ошибку**. Тем самым при записи документа будет проверяться на заполнение не только табличная часть, но и ее отдельные реквизиты (колонки).



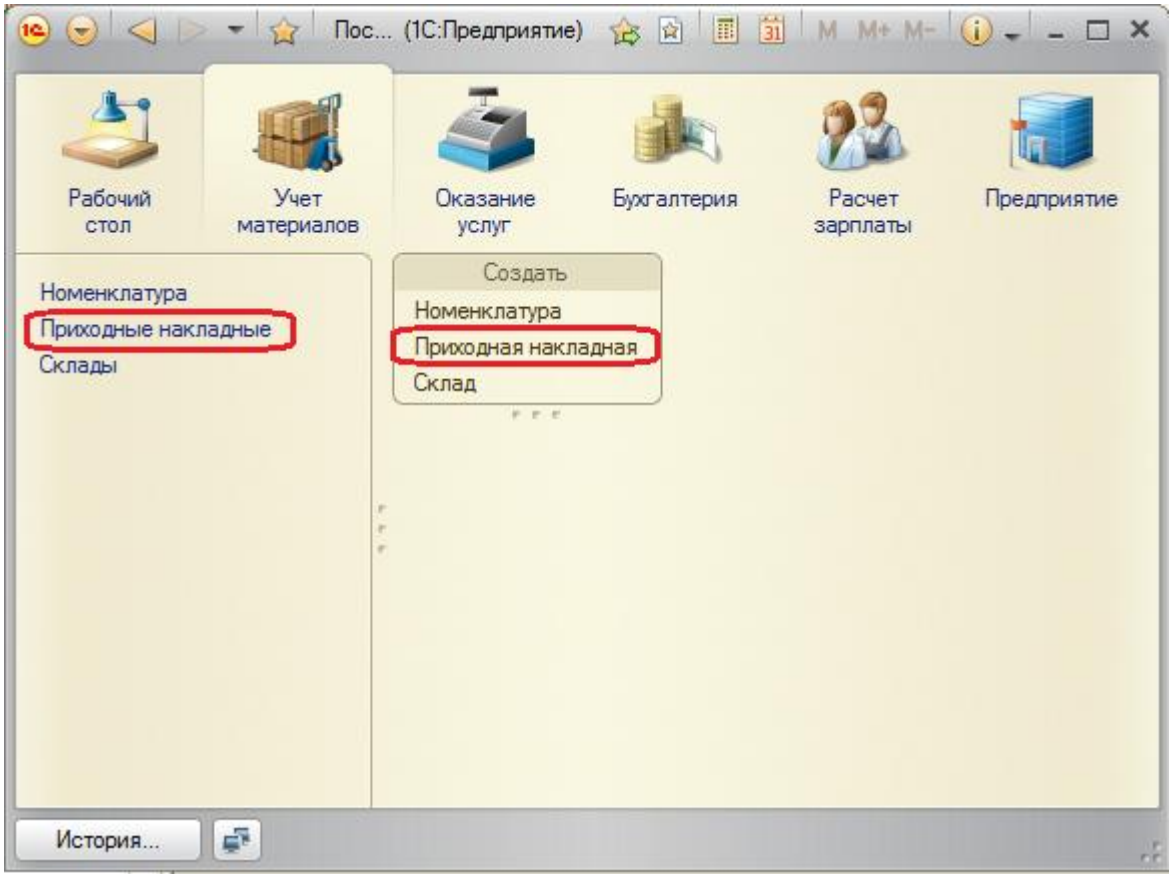
Закройте окно свойств справочника.

Отредактируем командный интерфейс, чтобы в подсистеме **УчетМатериалов** была доступна команда создания новых документов. Выделите ветвь **Подсистемы**, вызовите контекстное меню – **Все подсистемы**. В группе **Панель действий.Создать** включите видимость у команды **Приходная накладная: создать**.



В режиме 1С: Предприятие

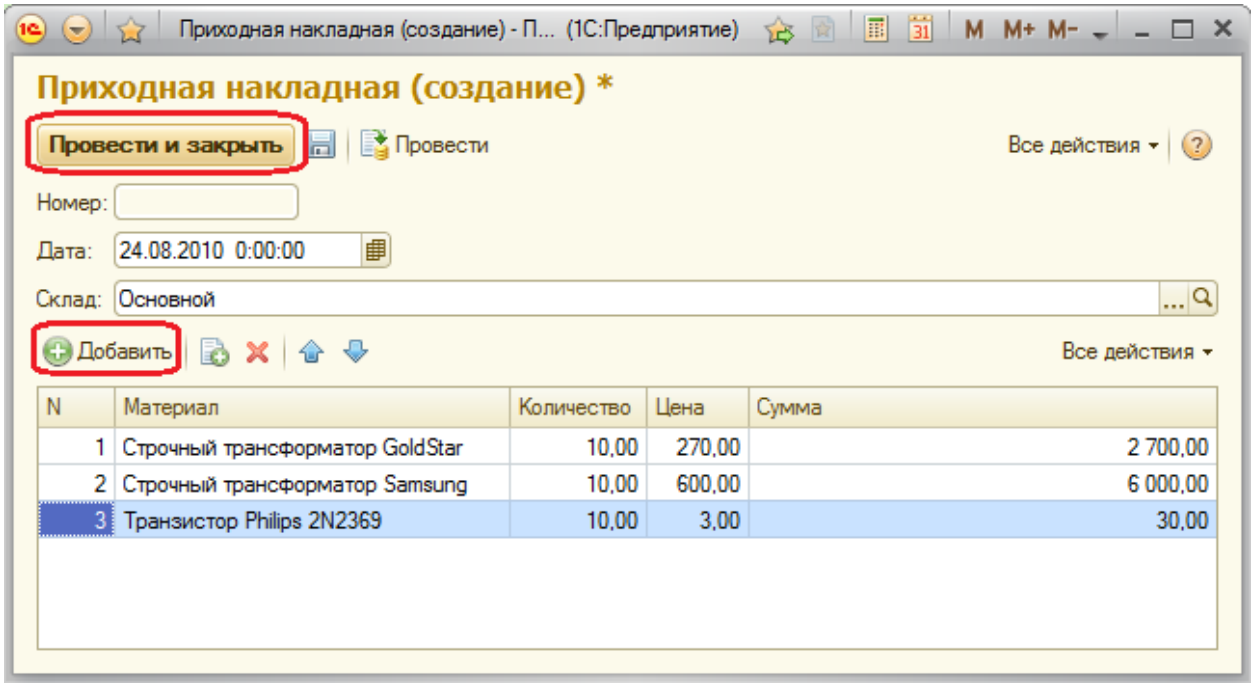
Запустите 1С: Предприятие в режиме отладки. В появившемся окне Вы увидите, что в панели навигации разделов **Бухгалтерия** и **Учет материалов** появилась команда **Приходные накладные** для открытия списка приходных накладных, а также в панели действий раздела **Учет материалов** появилась команда **Приходная накладная** для создания новых документов этого вида.



Пока в нашей базе данных нет ни одного документа Приходная накладная, поэтому выполните команду **Приходная накладная** в панели действий раздела **Учет материалов** и создайте новую приходную накладную. Перед Вами откроется основная форма документа. Заголовок этой формы совпадает с синонимом документа.

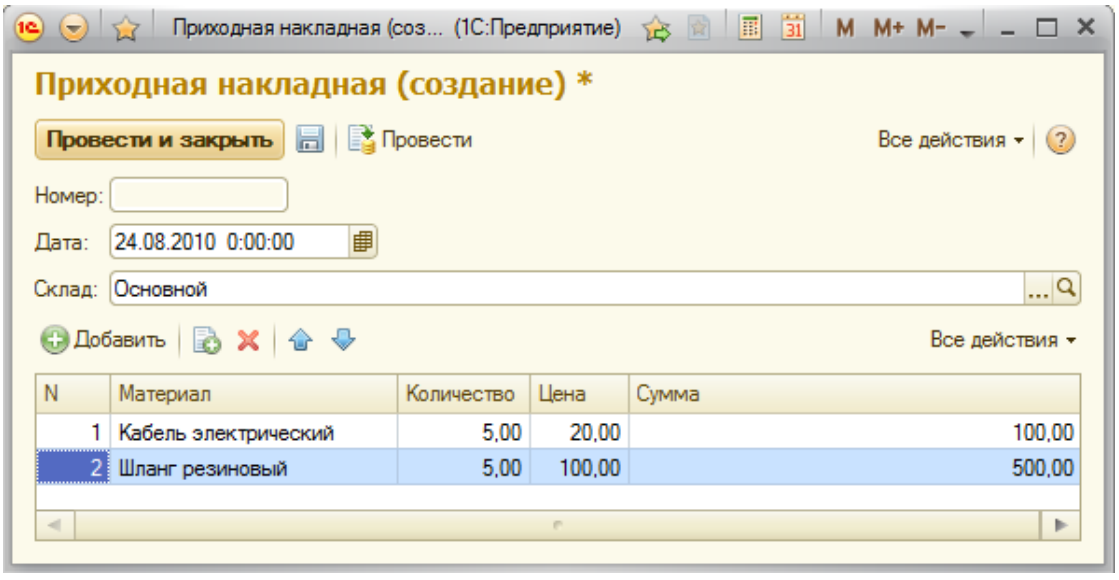
Система автоматически подставит текущую дату создания документа и нулевое время, т.к. документ еще не проведён. Поле **Номер** не заполнено, но система сама его сгенерирует, т.к. свойство **Автонумерация** (на вкладке Нумерация в свойствах документа) включено по умолчанию. Обратите внимание, что поле **Склад** уже заполнено значением Основной, как мы и задавали в свойствах реквизита. Заполните табличную часть приходной накладной материалами для ремонта телевизоров как показано на рисунке. При

нажатии кнопки выбора в поле **Материал** открывается форма для выбора элементов справочника **Номенклатура**, т.к. этот реквизит имеет ссылочный тип данных и ссылается на справочник **Номенклатура**. Также можно начать набирать название материала в поле Материал и система выдаст всплывающий список подходящих под описание материалов. После заполнения нажмите **Провести и закрыть**.

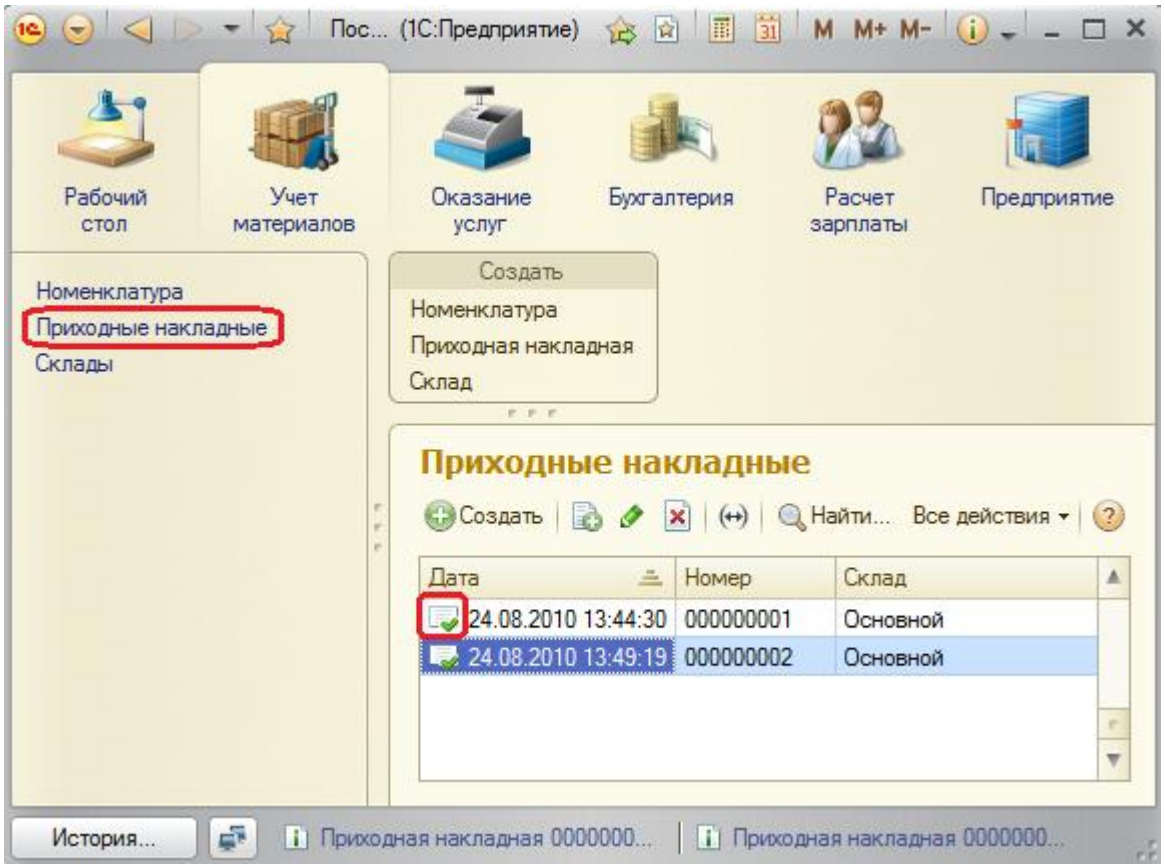


Документ будет сохранен и проведен, ему будет присвоен автоматически сгенерированный номер и текущее время проведения документа.

Аналогичным способом создайте второй документ, который будет приходовать на **Основной склад** материалы для установки стиральных машин.



Посмотрим список созданных документов, выполнив команду **Приходные накладные** в навигационной панели.



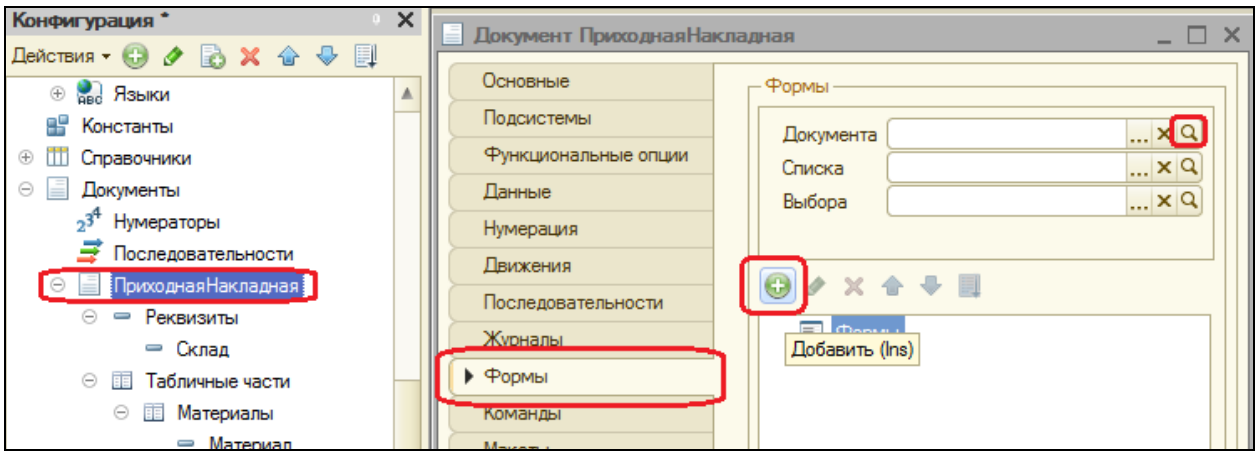
Зеленая галочка около документа означает, что он проведён.

Автоматический пересчет суммы в строках документа

Обратили внимание, что пришлось вручную вводить сумму в каждой строке? Возникает желание, чтобы сумма автоматически вычислялась каждый раз при изменении цены или количества материалов.

Это можно сделать, создав свою форму документа и воспользовавшись возможностями встроенного языка для задания алгоритма.

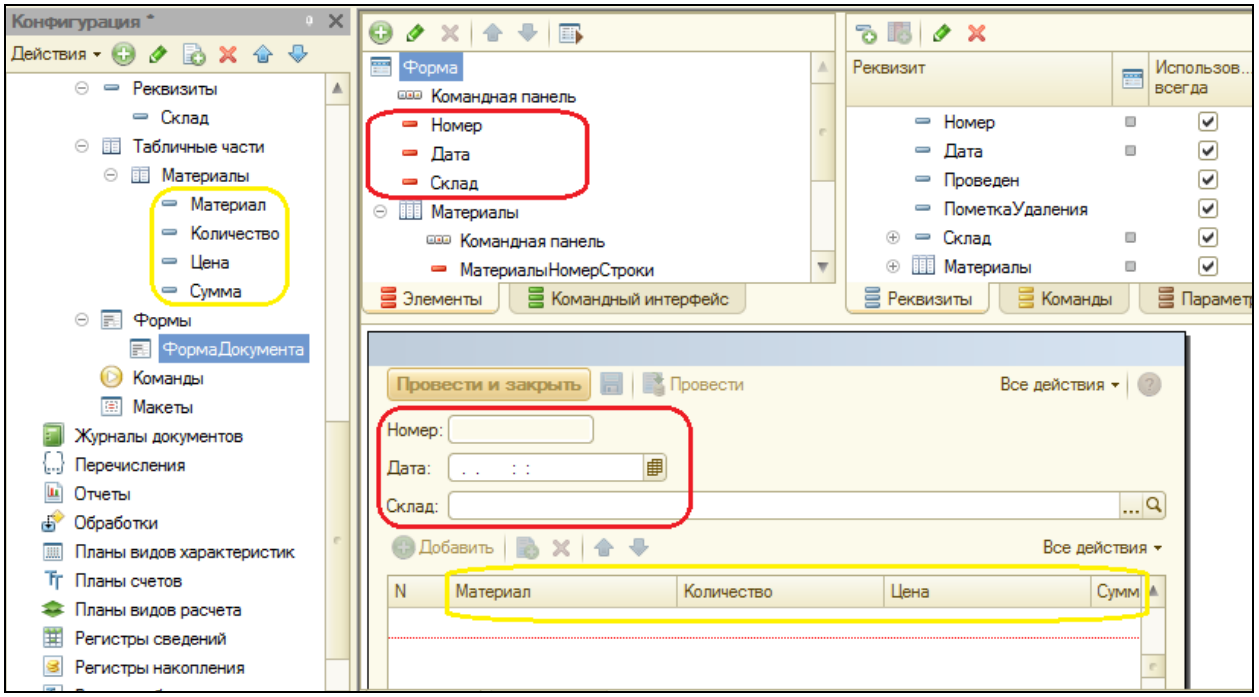
Вернитесь в конфигуратор и откройте окно редактирования документа **ПриходнаяНакладная**, закладка **Формы**. Для создания формы нажмите кнопку **Добавить** или значок лупы около поля **Документы**.



Откроется конструктор форм. Там ничего не меняете и сразу нажмите Готово. Откроется окно редактора форм.

Разработчик не может нарисовать форму, а только может указать из каких элементов будет состоять форма.

Элементы формы в верхнем левом окне редактора форм образует иерархическую структуру. Чем выше в списке элемент, тем выше и левее на форме он будет располагаться (красное). От изменения порядка в дереве объектов изменится порядок следования на форме (желтое).

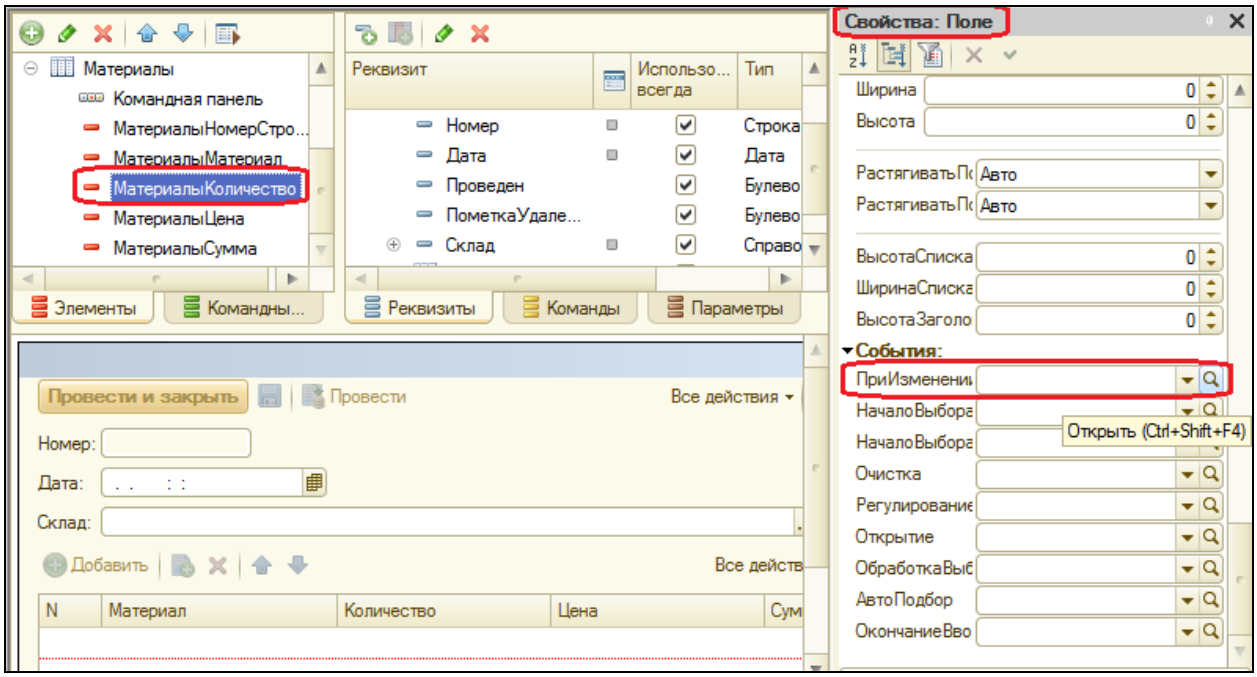


Обработчик события

У системы существуют события, которые связаны с различными моментами её функционирования. В том числе, связанные с функционированием форм и элементов в формах.

Используя встроенный язык, разработчик может внедриться в эти события и описать собственный алгоритм того, что должно происходить при наступлении этого события.

Дважды щелкните на элементе формы **МатериалыКоличество** (или контекстное меню – Свойства). Справа возникнет окно свойств, прокрутите его до конца вниз, найдите поле **ПриИзменении** и нажмите на значок лупы рядом с ним.



Система откроет закладку **Модуль** редактора формы. **Модуль** – это хранилище для текста программы на встроенном языке. В данном случае это модуль формы. В модуль формы в процедуру **МатериалыКоличествоПриИзменении()** добавим следующий текст.

```
СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;

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

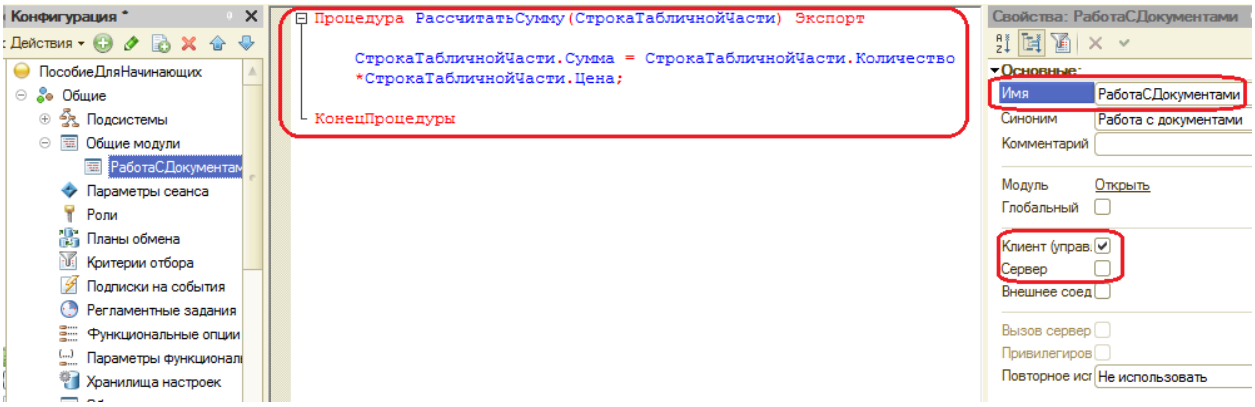
Запустите 1С: Предприятие в режиме отладки, зайдите в любой документ Приходная накладная и измените количество товара – сумма автоматически пересчитывается. Измените количество как было. Закройте приложение и вернитесь в конфигуратор.

Одна процедура для обработки нескольких событий

Теперь при изменении количества в любой строке документа **Приходная накладная**, сумма в этой в этой строке пересчитывается автоматически. Но теперь хотелось бы сделать тоже самое для поля **Цена**. Также подобное автозаполнение может понадобиться нам в

других документах. Поэтому лучше будет поместить расчет суммы в некоторое общедоступное место, чтобы разные документы могли использовать этот алгоритм. Для описания таких мест служат объекты **Общий модуль**, расположенные в ветке **Общие - Общие модули**. Процедуры и функции, содержащиеся в этих модулях, могут быть доступны для любых объектов конфигурации.

Создайте общий модуль РаботаСДокументами, установите в его свойствах флажок Клиент, а флажок Сервер снимите (это означает, что модуль будет скомпилирован в тонком клиенте и в веб-клиенте, а не на сервере).

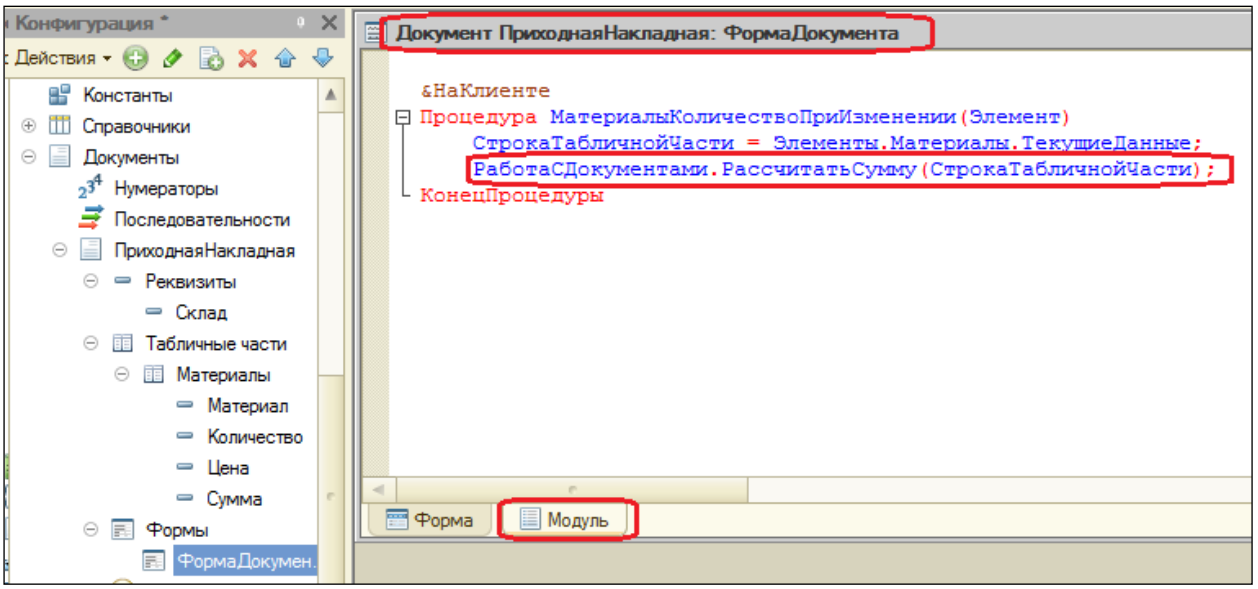


Перенесите в него созданную процедуру расчета суммы. В документе оставим вызов этой процедуры из общего модуля.

```
Процедура РассчитатьСумму(СтрокаТабличнойЧасти) Экспорт

        СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество

        *СтрокаТабличнойЧасти.Цена;

КонецПроцедуры
```

Теперь зайдите в модуль формы документа **Приходная накладная** и изменим вторую строчку процедуры на следующую:

```
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
```

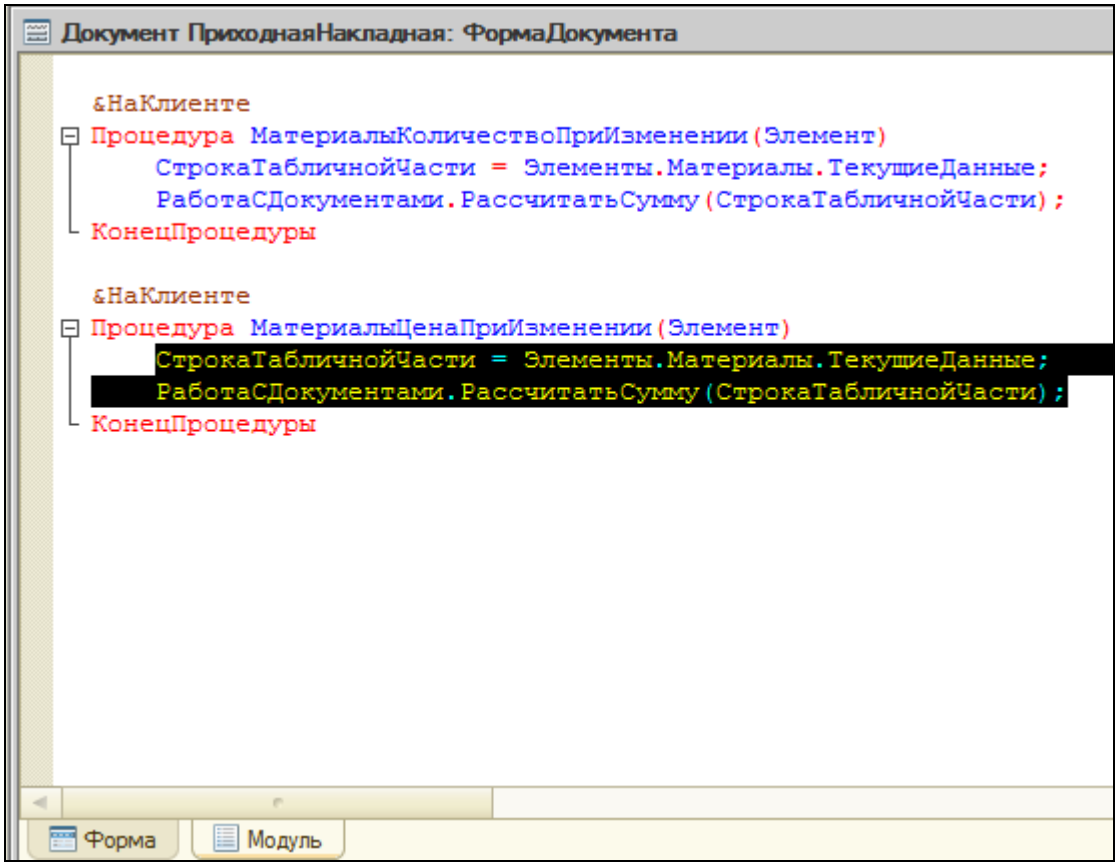


Тем самым мы вызываем процедуру РассчитатьСумму() из общего модуля РаботаСДокументами и передаем ей в качестве параметра текущую строку табличной части.

Осталось для поля **Цена** установить такой же обработчик. Создайте обработчик события **ПриИзменении** для поля табличной части **МатериалыЦена** также, как делали это для поля МатериалыКоличество, повторите в нем вызов процедуры РассчитатьСумму() из общего модуля.

Для этого нажмите внизу вкладку Форма (рядом с Модуль), дважды щелкните в левом верхнем окне на **МатериалыЦена**, прокрутите окно свойств, нажмите на значок лупы в поле **ПриИзменении** и в выделенную область впишите:

```
СтрокаТабличнойЧасти = Элементы.Материалы.ТекущиеДанные;

РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
```



Запустите 1С: Предприятие в режиме отладки и убедитесь, что теперь при изменении как количества, так и цены, сумма автоматически пересчитывается.

Документ «Оказание услуги»

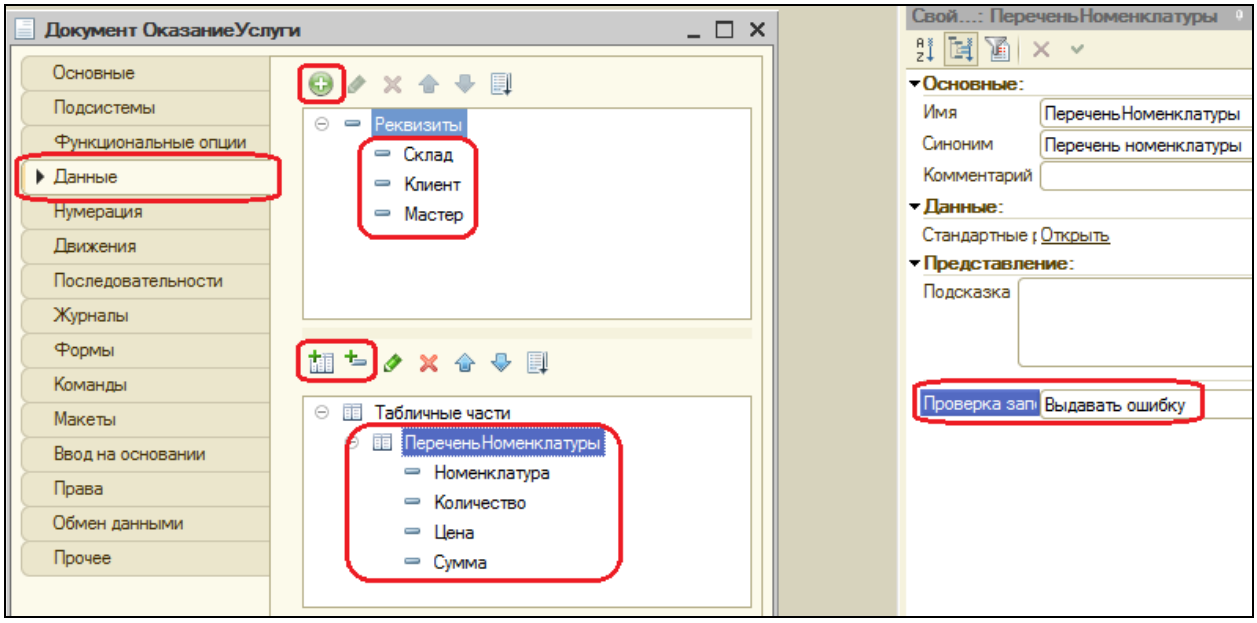
Создайте новый документ с именем **ОказаниеУслуги**. **Представление объекта** не задаем – будет использоваться синоним, **Представление списка** задайте как **Оказание услуг**. На вкладке **Подсистемы** отметьте **ОказаниеУслуг** и **Бухгалтерия**. На вкладке **Данные** создайте реквизиты документа:

- **Склад**, тип **СправочникСсылка.Склады**, **Значение заполнения** – **Основной**;
- **Клиент**, тип **СправочникСсылка.Клиенты**, **Проверка заполнения** – **Выдавать ошибку**;
- **Мастер**, тип **СправочникСсылка.Сотрудники**, **Проверка заполнения** – **Выдавать ошибку**.

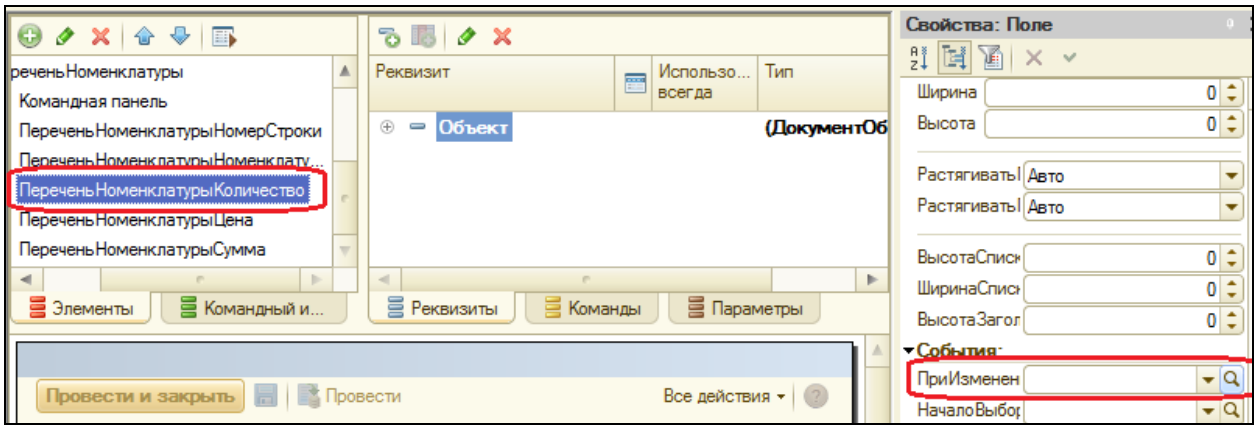
Создайте табличную часть документа с именем **ПереченьНоменклатуры** с реквизитами:

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

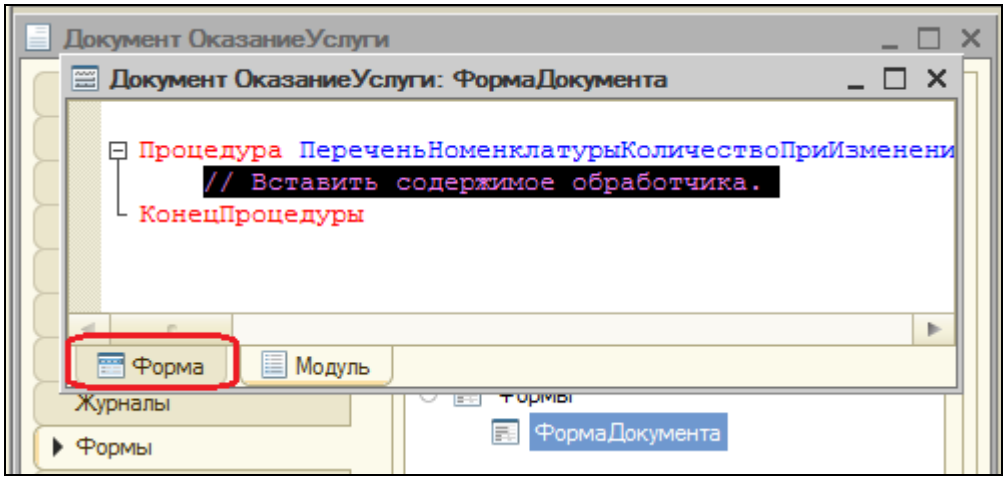
Установите для табличной части в целом и для каждого ее реквизита свойство **Проверка заполнения** в **Выдавать ошибку**.



Перейдите на вкладку **Формы**. Создайте основную форму документа. Для поля **ПереченьНоменклатурыКоличество** создайте обработчик события **ПриИзменении**, в котором будем вызывать процедуру РассчитатьСумму() из общего модуля **РаботаСДокументами**.



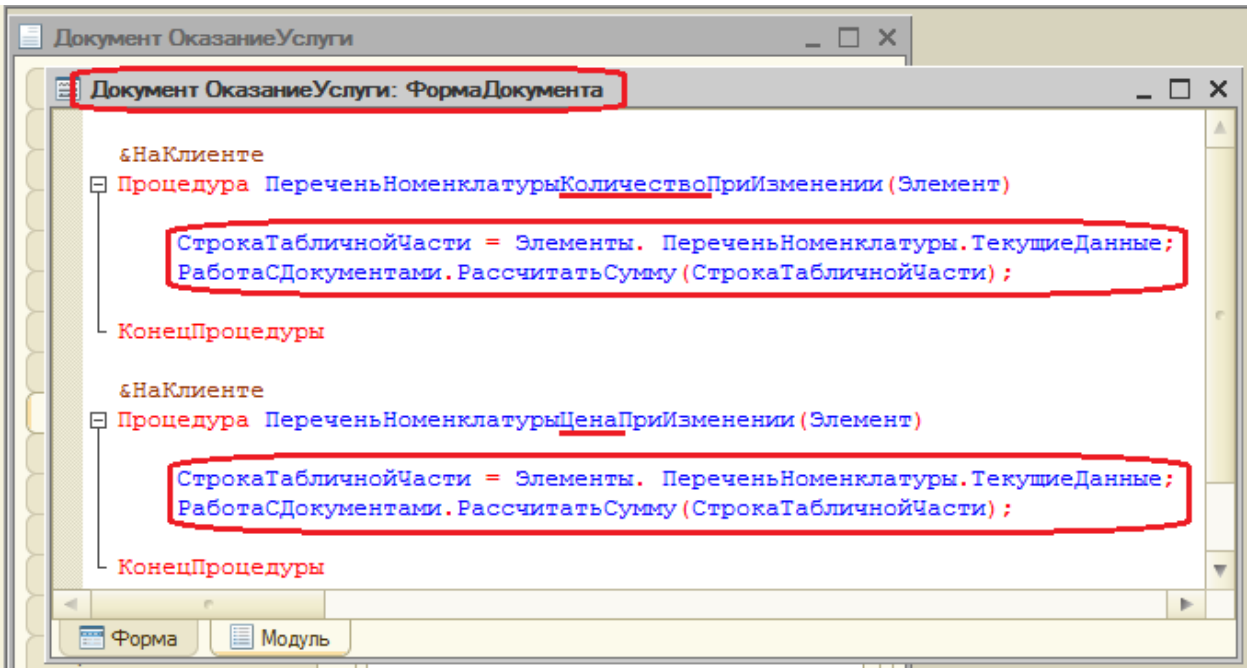
Откроется модуль формы с шаблоном обработчика события **ПереченьНоменклатурыКоличествоПриИзменении**, который пока не заполняйте, а перейдите в окно элементов формы на закладку **Форма** и аналогично создайте обработчик события **ПереченьНоменклатурыЦенаПриИзменении** для поля **ПереченьНоменклатурыЦена**.



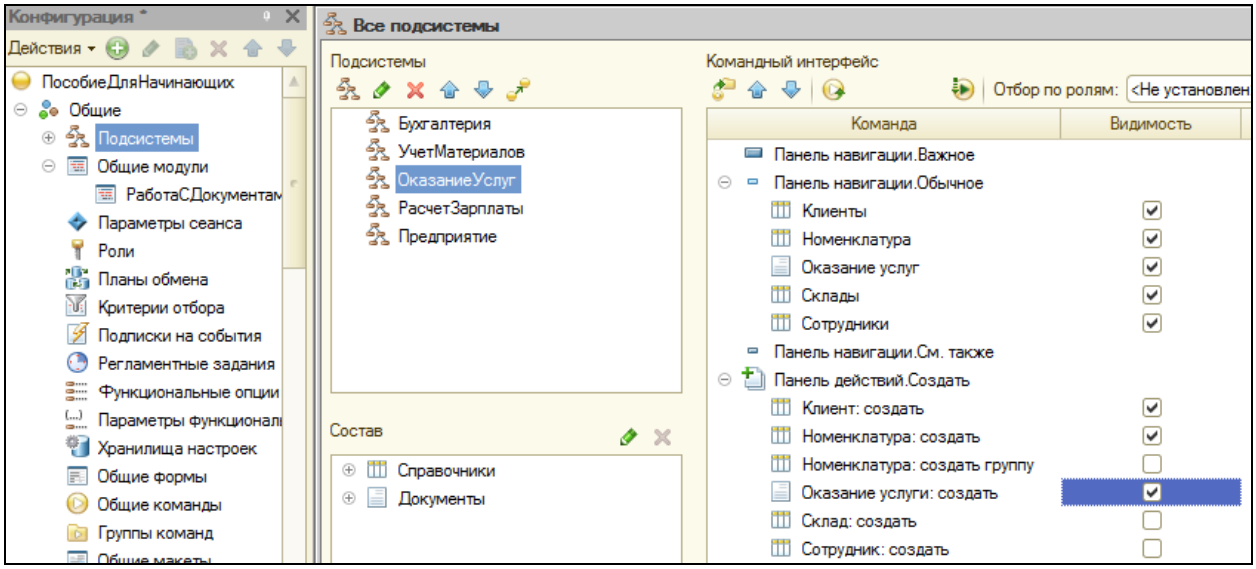
Далее модуль формы документа ОказаниеУслуги заполните следующим образом:

```
&НаКлиенте

Процедура ПереченьНоменклатурыКоличествоПриИзменении(Элемент)

        СтрокаТабличнойЧасти = Элементы. ПереченьНоменклатуры.ТекущиеДанные;

        РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры
```

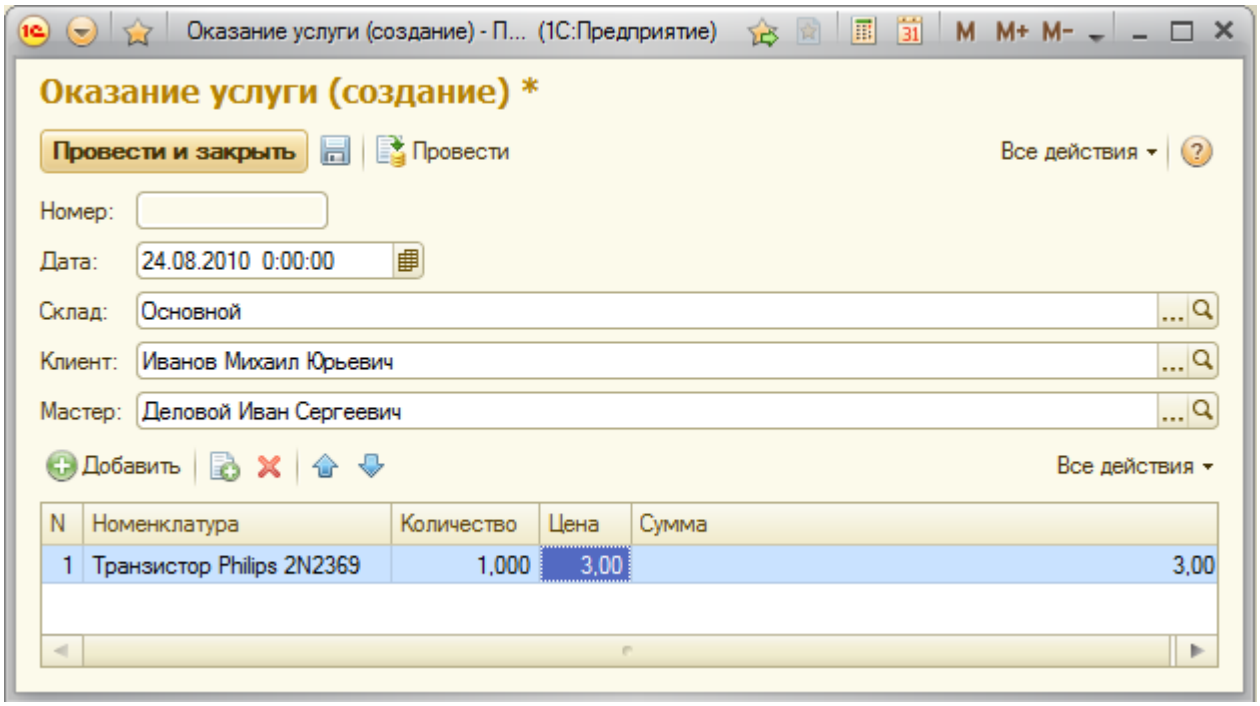
```
&НаКлиенте
Процедура ПеречеьНоменклатурыЦенаПриИзменении(Элемент)
        СтрокаТабличнойЧасти = Элементы. ПеречеьНоменклатуры.ТекущиеДанные;
        РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
КонецПроцедуры
```



Отредактируйте командный интерфейс, чтобы в подсистеме **Оказание услуг** была доступна команда создания новых документов. Запустите 1С: Предприятие в режиме отладки.



В панели действий раздела **Оказание услуг** создайте документ и заполните следующим образом:



Обратите внимание, что склад **Основной** подставляется по умолчанию, а для полей **Мастер** и **Клиент** выполняется проверка заполнения, а также при вводе цены и количества сумма рассчитывается автоматически по нашему алгоритму.

Контрольные вопросы

- ✓ Какими характерными особенностями обладает документ
- ✓ Для чего предназначены реквизиты и табличные части документа
- ✓ Что такое проведение документа
- ✓ Как создать объект Документ и описать его структуру
- ✓ Как создать новый документ и заполнить его данными
- ✓ Как создать собственную форму документа
- ✓ Что такое обработчик события и как его создать
- ✓ Что такое модуль и для чего он нужен
- ✓ Зачем нужны общие модули

Теоретическая вставка. Сервер и клиенты

Система 1С: Предприятие поддерживает два варианта работы: **файловый** и **клиент-серверный**.

Файловый вариант работы с информационной базой рассчитан на персональную работу одного пользователя или небольшого количества пользователей в локальной сети. В этом варианте все данные информационной базы (конфигурация, база данных, административная информация) располагаются в одном файле.

Основное назначение файлового клиента – быстро и легко установить систему и работать с ней. Например, что-то посмотреть или доработать дома или на ноутбуке. В файловом варианте тоже можно вести реальную учетную работу, но при этом нужно понимать, что он не предоставляет всех тех же возможностей по масштабируемости, защите данных, какие имеет клиент-серверный вариант.

Клиент-серверный вариант предназначен для использования в рабочих группах или в масштабе предприятия и реализован на основе трехуровневой архитектуры «клиент-сервер».

Клиент-серверный вариант – основной для работы в многопользовательской среде с большим объемом данных. Он предоставляет все возможности по масштабируемости, администрированию и защите данных. Однако требует значительных усилий по установке и администрированию.

При этом физически серверная и клиентские части системы могут располагаться как на разных компьютерах, так и на одном. Главное, что пользователь не имеет непосредственного доступа к серверу базы данных и это позволяет обеспечивать безопасность данных. А в файловом варианте база данных должна находиться на общем сетевом ресурсе, доступном всем пользователям сети.

Система 1С: Предприятие изначально рассчитана на клиент-серверный вариант работы. Хотя сейчас Вы разрабатываете свою учебную конфигурацию в файловом варианте работы, она будет работать и в клиент-серверном варианте без ваших дополнительных усилий.

Прикладные решения разрабатываются один раз и одинаково работают в обоих вариантах, т.е. переделка конфигурации при переходе с одного варианта на другой не требуется.

Клиент-серверная архитектура разделяет всю работающую систему на три различные части:

- Клиент
- Сервер 1С: Предприятия
- Сервер баз данных.

Клиентское приложение – программа, часть системы 1С: Предприятие. Основное назначение – организация пользовательского интерфейса, отображение данных с возможностью их изменения. Кроме этого, клиент может исполнять код на встроенном языке (т.е. алгоритмы разработчика), но оперирует ограниченным пространством типов языка. Такой подход позволяет клиенту быть очень «легким», не требовать много ресурсов и работать даже в веб-браузере.

Клиент взаимодействует с *сервером 1С: Предприятия* – это тоже программа, основная задача которой – передавать запросы от клиента к *серверу баз данных* и возвращать клиенту результаты запросов. Другая задача сервера – исполнение большинства алгоритмов на встроенном языке, подготовка данных для отображения форм, отчетов и т.д. Т.е. все сложные вычисления выполняются на сервере. При этом на сервере доступно всё пространство типов встроенного языка, за исключением интерфейсных типов.

Сервер баз данных – тоже программа, поставляемая сторонними производителями и не являющаяся частью системы 1С: Предприятие. Её основное назначение – организация и ведение баз данных – структурированных наборов данных.

В настоящее время система 1С: Предприятие может работать со следующими серверами баз данных:

- Microsoft SQL Server
- PostgreSQL
- IBM DB2
- Oracle Database.

Практическая работа № 5

Регистры накопления (0:50)

В этой работе Вы познакомитесь с объектом конфигурации **Регистр накопления**, узнаете, для чего он нужен, его структуру и основные особенности. Создадите один из регистров накопления, отражающий изменение данных в процессе работы созданных Вами ранее документов.

В системе 1С: Предприятие есть несколько объектов конфигурации, которые позволяют создавать в базе данных структуры, предназначенные для накопления информации в удобном для последующего анализа виде. Использование регистров позволяет накапливать в них данные, поставляемые различными документами, легко создавать нужные отчеты или использовать эти данные в алгоритмах работы конфигурации. В системе существуют несколько видов регистров, рассмотрим один из них.

Регистр накопления предназначен для описания структуры накопления данных. Эти данные будут храниться в таблица в виде отдельных записей, каждая из которых имеет одинаковую, заданную в конфигураторе структуру. Отличительная особенность – не предназначен для редактирования пользователем.

Основное назначение регистра накопления – накопление числовой информации в разрезе нескольких измерений, которые описываются разработчиком в свойствах регистра накопления.

Виды числовой информации, накапливаемой регистром накопления, называются *ресурсами*.

Например, регистр накопления может накапливать информацию о количестве и сумме товаров на складах. В этом случае он будет иметь *измерения* **Товар** и **Склад** и *ресурсы* **Количество** и **Сумма**.

Изменение состояние регистра накопления происходит, как правило, при *проведении* документа и заключается в добавлении в него нескольких записей. Каждая запись содержит значения измерений, значения приращений ресурсов, ссылку на документ, который вызвал эти изменения (*регистратор*) и направление приращения (приход или расход). Такой набор записей называется *движениями* регистра накопления. Каждому *движению* регистра накопления всегда должен

соответствовать *регистратор* – объект информационной базы (обычно документ), который произвел эти движения.

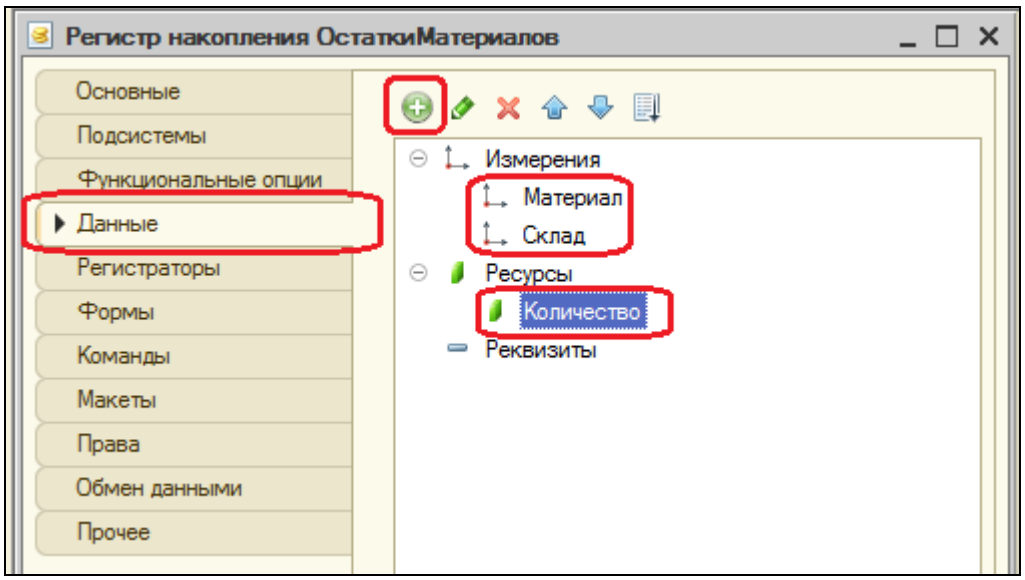
Кроме этого, регистр накопления может хранить дополнительную информацию, описывающую каждое движение – *реквизиты*.

Добавление регистра накопления

Откройте конфигуратор и добавьте новый объект конфигурации **Регистр накопления** с именем **ОстаткиМатериалов**. **Расширенное представление списка** – **Движение по регистру Остатки материалов**. Этот заголовок будет отображаться в окне списка записей регистра. На вкладке **Подсистемы** отметьте **УчетМатериалов**, **ОказаниеУслуг** и **Бухгалтерия**. Перейдите на закладку **Данные** и создайте **измерения** регистра:

- Материал, тип СправочникСсылка.Номенклатура;
- Склад, тип СправочникСсылка.Склады;

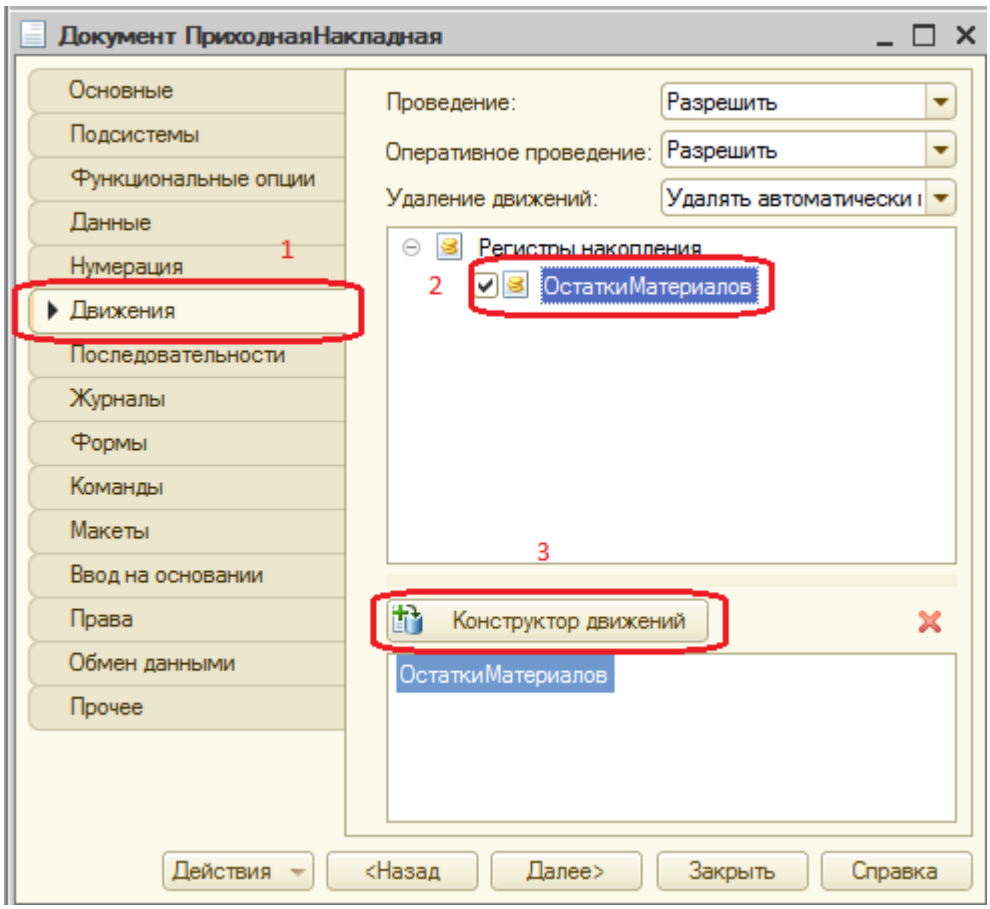
Затем создайте ресурс **Количество** с длиной 15 и точностью 3.

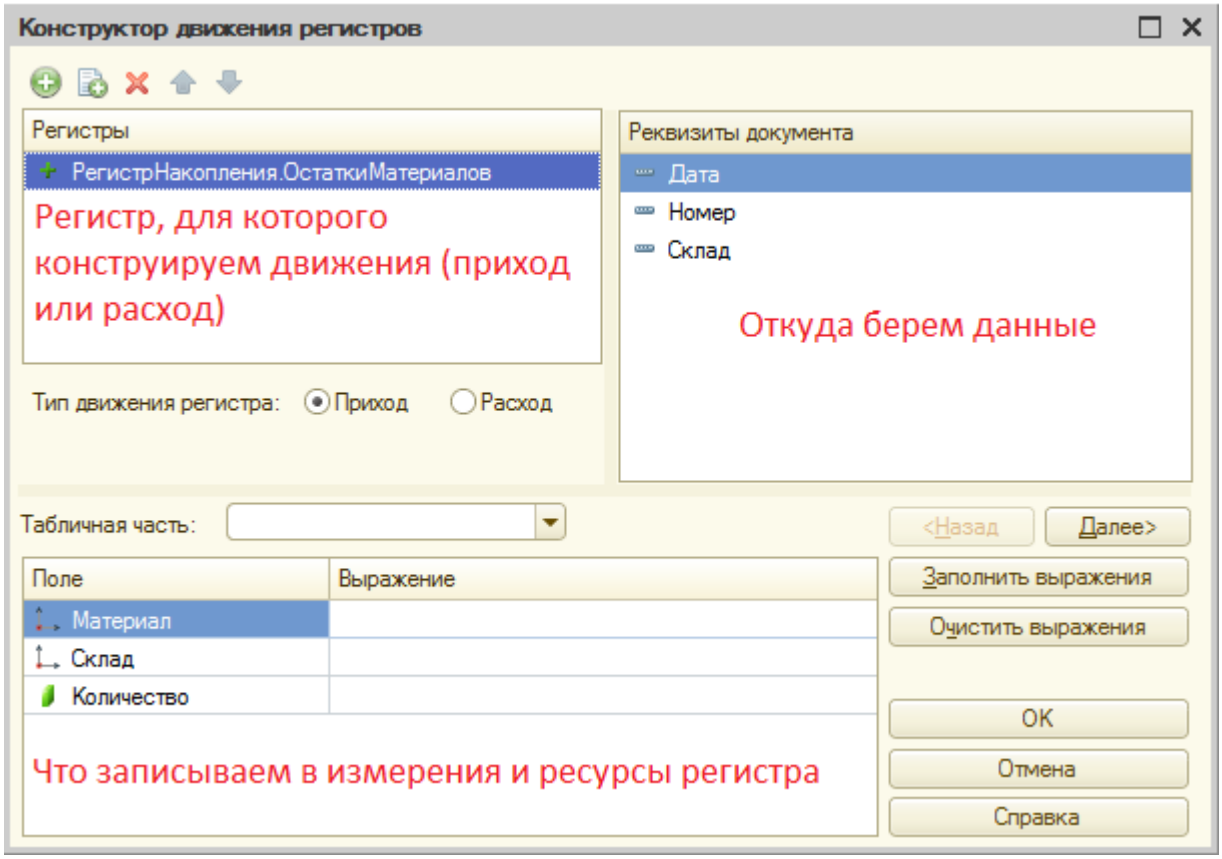


Движения документа

- это записи в регистрах, которые создаются в процессе проведения документа и отражают изменения, производимые документом.

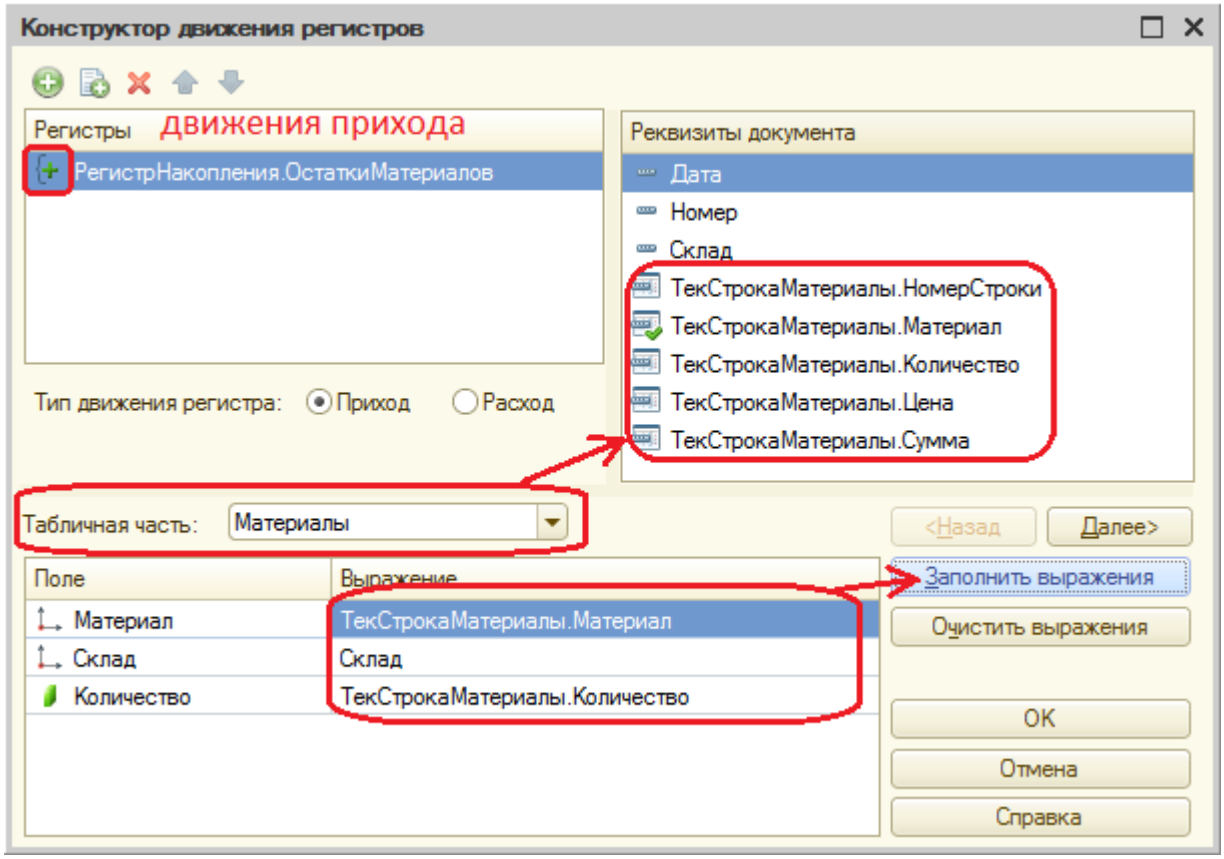
Откройте окно редактирования объекта Документ **ПриходнаяНакладная** на закладке **Движения**. Раскройте список **Регистры накопления** и выделите **ОстаткиМатериалов**. Затем нажмите кнопку **Конструктор движений**.


Документ ПриходнаяНакладная
Основные
Подсистемы
Функциональные опции
Данные
Нумерация
Движения
Последовательности
Журналы
Формы
Команды
Макеты
Ввод на основании
Права
Обмен данными
Прочее
1
Проведение:
Разрешить
Оперативное проведение:
Разрешить
Удаление движений:
Удалять автоматически
Регистры накопления
2
ОстаткиМатериалов
3
Конструктор движений
ОстаткиМатериалов
Действия
<Назад
Далее>
Закрыть
Справка


Конструктор движения регистров
Регистры
РегистрНакопления.ОстаткиМатериалов
Регистр, для которого конструируем движения (приход или расход)
Реквизиты документа
Дата
Номер
Склад
Откуда берем данные
Тип движения регистра:
Приход
Расход
Табличная часть:
<Назад
Далее>
Поле
Выражение
Материал
Склад
Количество
Что записываем в измерения и ресурсы регистра
Заполнить выражения
Очистить выражения
OK
Отмена
Справка

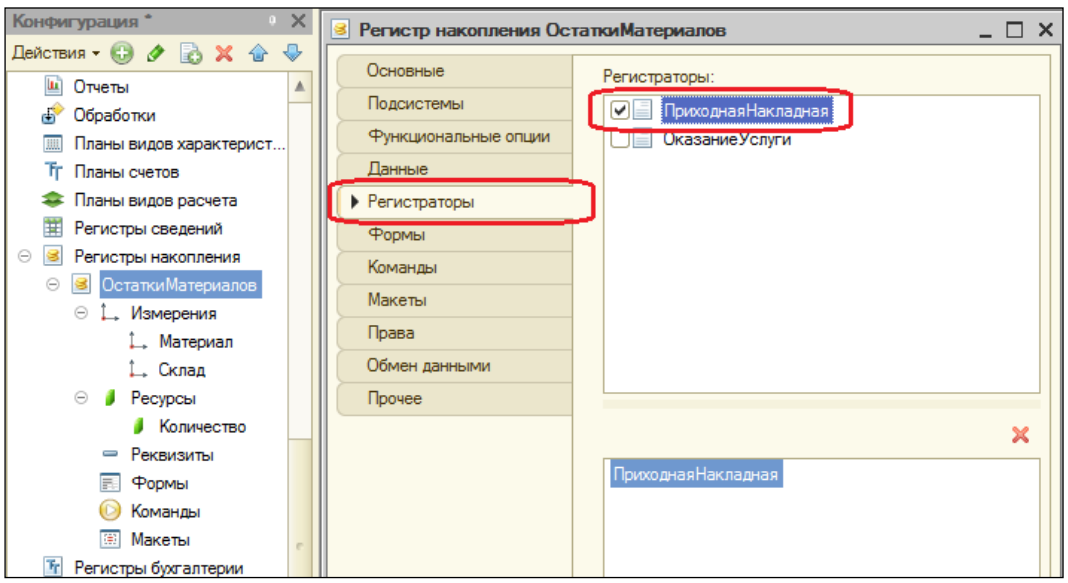
В таблице *Поле – Выражение* должны быть заданы формулы вычисления значений *измерений* и *ресурсов* регистра при записи *движений*.

В поле выбора **Табличная часть** выберем табличную часть документа – **Материалы**. Нажмите кнопку **Заполнить выражения**.



Нажмите ОК. Полюбуйтесь на текст алгоритма, сформированного конструктором в модуле документа ПриходнаяНакладная.

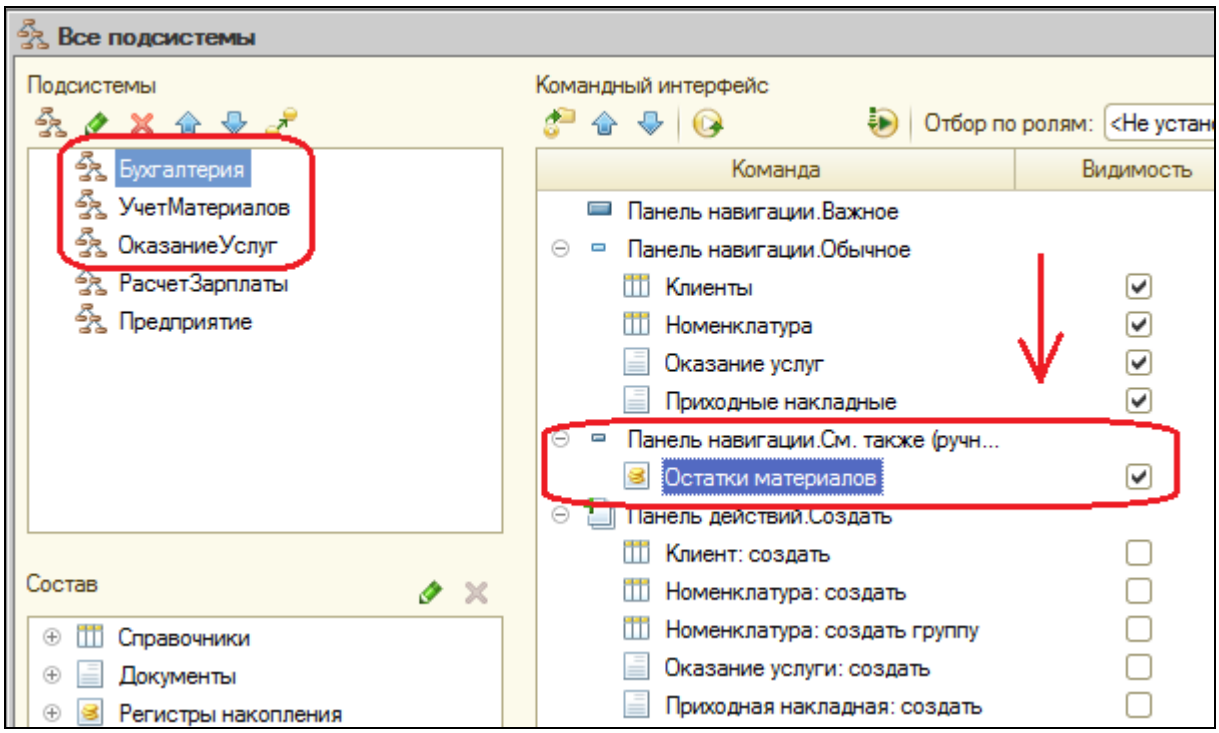
Откройте окно редактирования объекта *Регистр накопления* **ОстаткиМатериалов**, перейдите на закладку **Регистраторы**. В списке документов, созданных в конфигурации, Вы увидите отмеченный документ **ПриходнаяНакладная**, т.к. мы задали в модуле этого документа формирование движений в регистре **ОстаткиМатериалов**.



Отредактируйте командный интерфейс, чтобы в подсистемах **Бухгалтерия**, **ОказаниеУслуг** и **УчетМатериалов** была доступна ссылка для просмотра записей нашего регистра накопления.

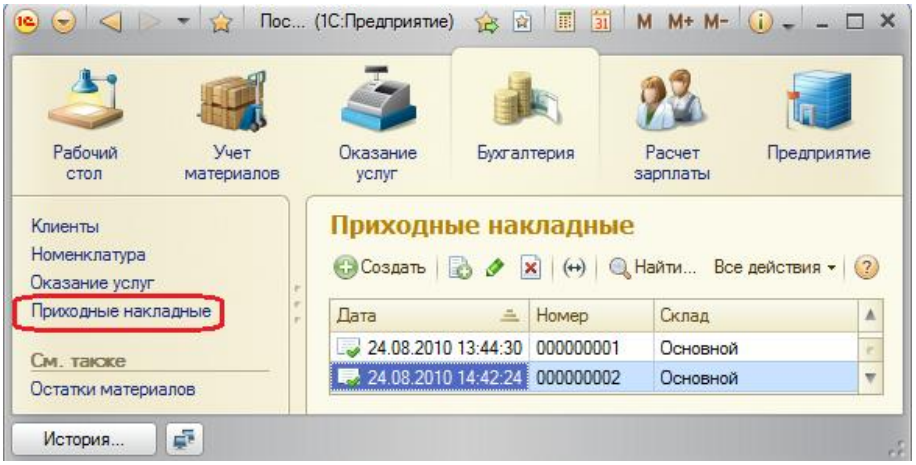
Дело в том, что команды открытия регистров также добавляются в панель навигации подсистем, но по умолчанию невидимы, в отличие от команд открытия справочников и документов.

В группе **Панель навигации.Обычное** включите видимость у команды **Остатки материалов** и перетащите ее в группу **Панель навигации.См также**. Так же сделайте в указанных подсистемах.

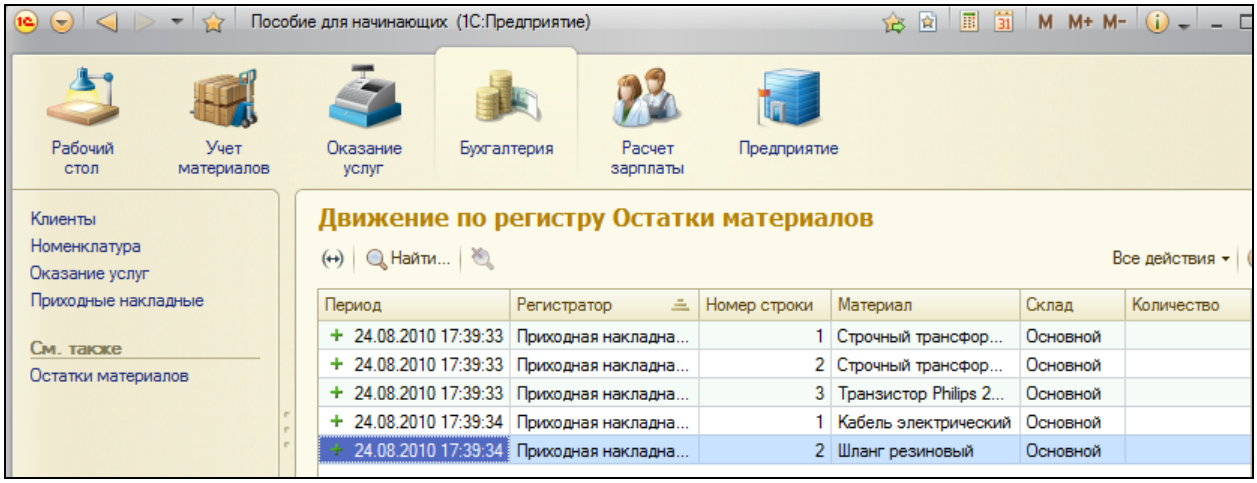


Запустите 1С: Предприятие в режиме отладки. В открывшемся окне видно, что в панели навигации в группе **См. также** разделов **Бухгалтерия**, **Оказание услуг** и **Учет материалов** появилась команда для открытия списка регистра **Остатки материалов**.

Чтобы проследить связь между проведением документа и накоплением информации в регистре, откройте список приходных накладных.



Откройте **Приходную накладную №1** и нажмите **Провести и закрыть**, т.е. перепроведите её. Тоже самое сделайте для второй накладной. Перепровести можно и не открывая документа – выбрать документы, **Все действия – Провести**. У документов в списке изменится время. Откройте регистр **Остатки материалов** через навигационную панель.



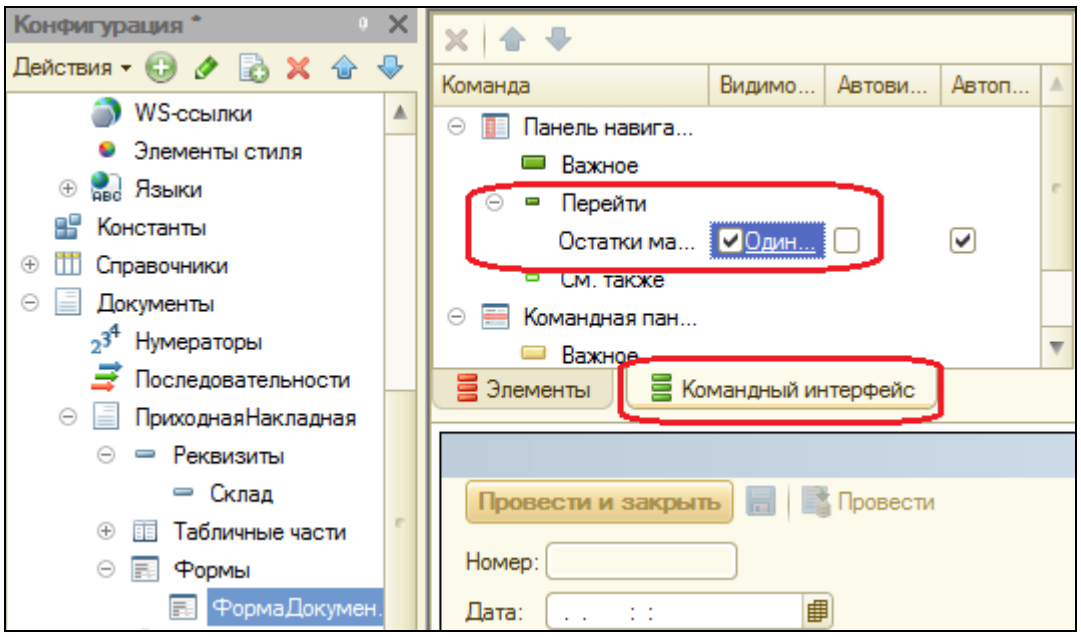
Видите, что при проведении приходных накладных появляются соответствующие записи в регистре накопления Остатки материалов. Добавилось 5 записей – первые три после проведения первого документа из трех строк, последние две после проведения второго из двух строк в табличной части.

Все поля регистра заполнились данными документов так, как Вы задали в обработчике проведения документа **ПриходнаяНакладная**. Пиктограмма со знаком + слева от каждой записи указывает на тип движения – **Приход**.

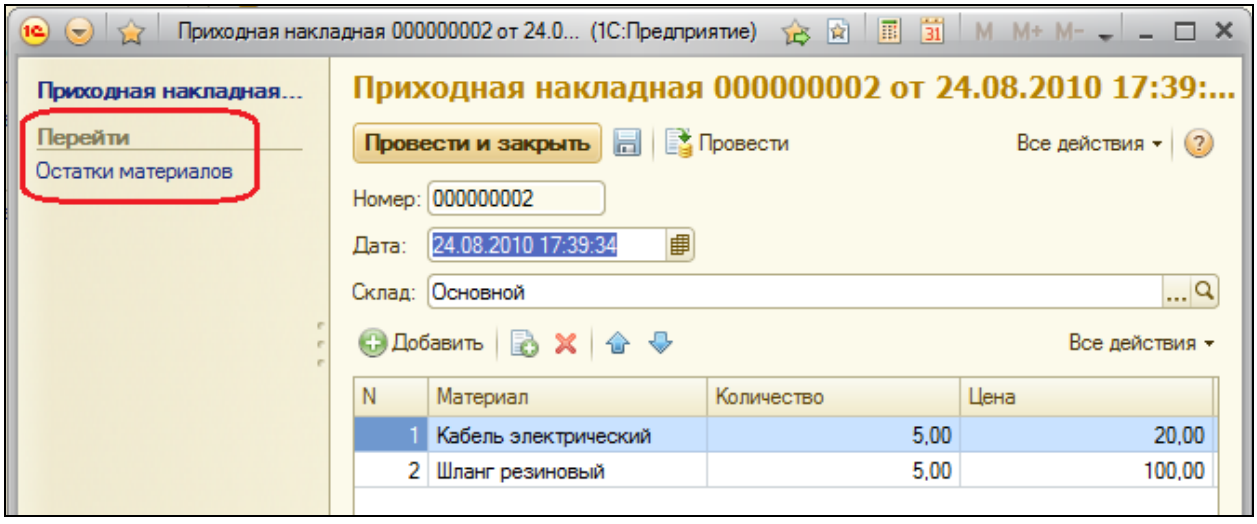
Команда перехода к движениям в форме документа

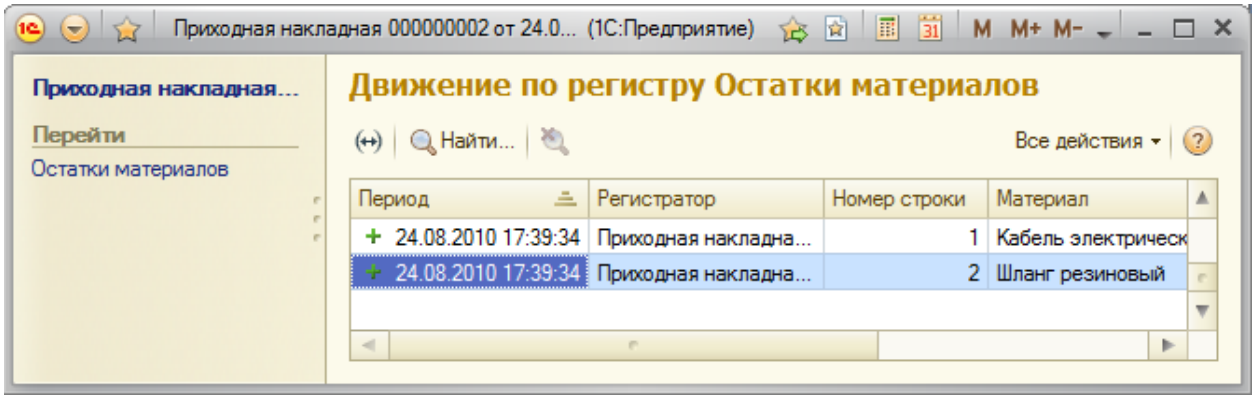
При реальной работе записей в регистре ОстаткиМатериалов будет много и будет трудно понять какие записи относятся к определенному документу. Поэтому наряду с общим списком регистра хотелось бы иметь возможность вызывать из формы документа список регистра, в котором показаны движения, произведенные только этим документом.

Для этого вернитесь в Конфигуратор и откройте форму документа **ПриходнаяНакладная.** На закладке **Командный интерфейс** в разделе **Панель навигации** раскройте группу **Перейти** и установите видимость команды **Остатки материалов (Объект.Ссылка).**



Запустите 1С: Предприятие в режиме отладки, откройте **Приходную накладную №2**. В форме документа появилась панель навигации, в которой можно перейти к списку записей регистра **Остатки материалов**, связанному с документом и обратно к содержимому документа.





Движения документа «Оказание услуги»

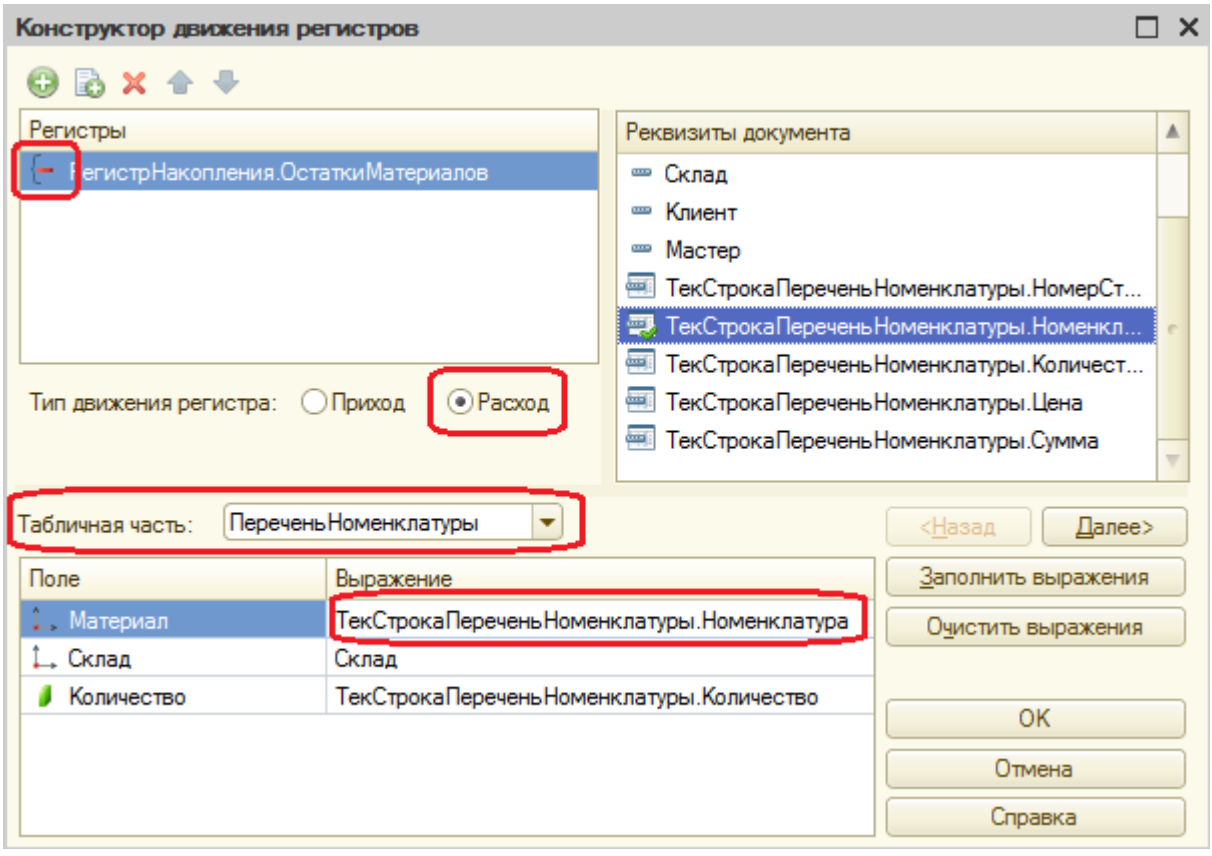
Теперь аналогичным образом создадим движения документа **ОказаниеУслуги**.

Откройте окно редактирования объекта Документ **ОказаниеУслуги**, перейдите на вкладку **Движения**, в списке регистров отметьте **ОстаткиМатериалов**, нажмите кнопку **Конструктор движений**.

Измените тип движения регистра на **Расход**, т.к. документ **ОказаниеУслуги** должен расходовать материалы в процессе оказания услуг. Пиктограмма слева от названия регистра изменится на знак -.

В поле выбора **Табличная часть** выберите **ПереченьНоменклатуры**. Нажмите кнопку **Заполнить выражения**. В нижнем окне сформируется соответствие полей (измерений и ресурсов) регистра и выражений для их расчета, но поле **Материал** не заполнится. Так происходит потому что имя поля табличной части – **Номенклатура** не совпадает с именем измерения регистра – **Материал**. Если оставить как есть, то движение фиксироваться не будет.

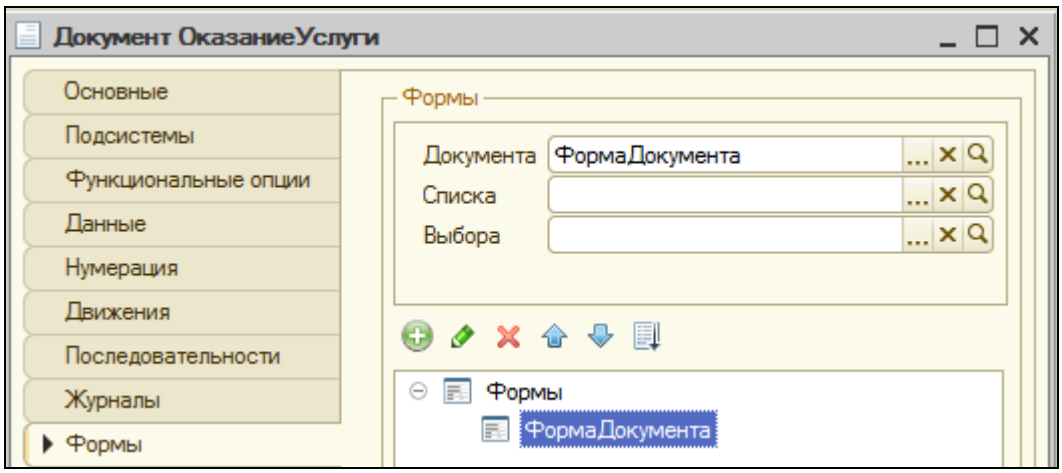
Выделите поле регистра **Материал** и в окне реквизиты документа дважды щелкните по строке **ТекСтрокаПереченьНоменклатуры.Номенклатура**. Таким образом, номенклатура для движений регистра накопления будет выбираться из табличной части документа.



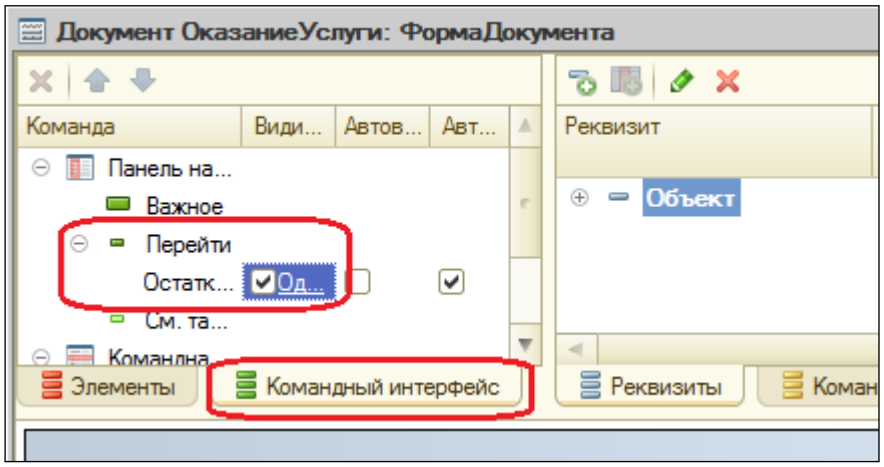
Нажмите ОК. Конструктор создал обработчик события **ОбработкаПроведения** объекта *Документ* **ОказаниеУслуги** и поместил его в модуль объекта.

Отредактируйте командный интерфейс формы документа **ОказаниеУслуги**, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра **Остатки материалов**, связанному с документом.

Закройте окно модуля, перейдите на вкладку формы документа ОказаниеУслуги, дважды щелкните на названии формы.

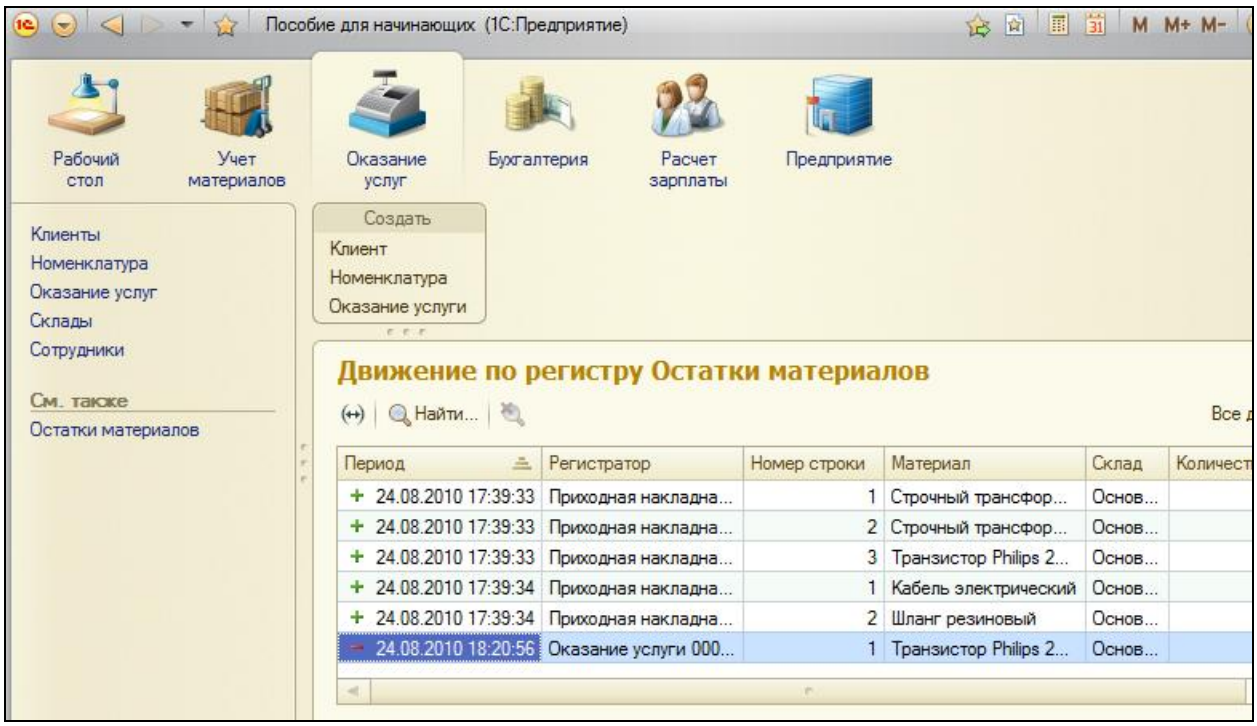


Далее выберите вкладку **Командный интерфейс**, раскройте группу **Панель навигации – Перейти** и установите видимость для команды открытия регистра накопления **Остатки материалов**.

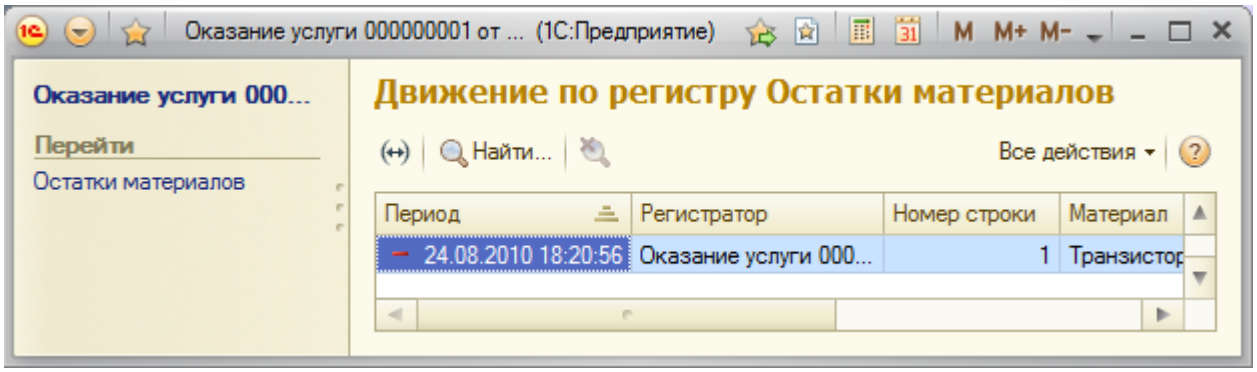


Запустите 1С: Предприятие в режиме отладки. В разделе **Оказание услуг** откройте документ **Оказание услуги №1** и перепроведите его.

Затем нажмите на **Остатки материалов** и откройте список регистра накопления. Вы увидите, что появилась запись, что соответствует количеству строк в табличной части проведенного документа. Пиктограмма  указывает, что произошел расход материала.



Сейчас Вы видите весь список движений регистра. Откройте этот список из формы документа (дважды щелкните на записи, далее из панели навигации Остатки материалов), таким образом, отфильтровав не относящиеся к данному документу записи.



Сформированные таким образом движения этого документа будут не совсем правильными. Дело в том, что в документе Оказание услуги могут содержаться не только расходуемые материалы, но и услуги. Поэтому в регистр Остатки материалов будут попадать записи и о расходуемых услугах, что неправильно. Пока не познакомитесь с перечислениями, оставим это как есть.

Контрольные вопросы

- ✓ Для чего предназначен объект конфигурации Регистр накопления
- ✓ Для чего нужны измерения регистра, ресурсы и реквизиты

- ✓ Что такое движение регистра и что такое регистратор
- ✓ Как создать новый регистр накопления и описать его структуру
- ✓ Как создать движения документа с помощью конструктора движений
- ✓ Как показать команды открытия списка регистра в интерфейсе конфигурации и в интерфейсе формы.

Практическая работа № 6

Простой отчёт (0:25)

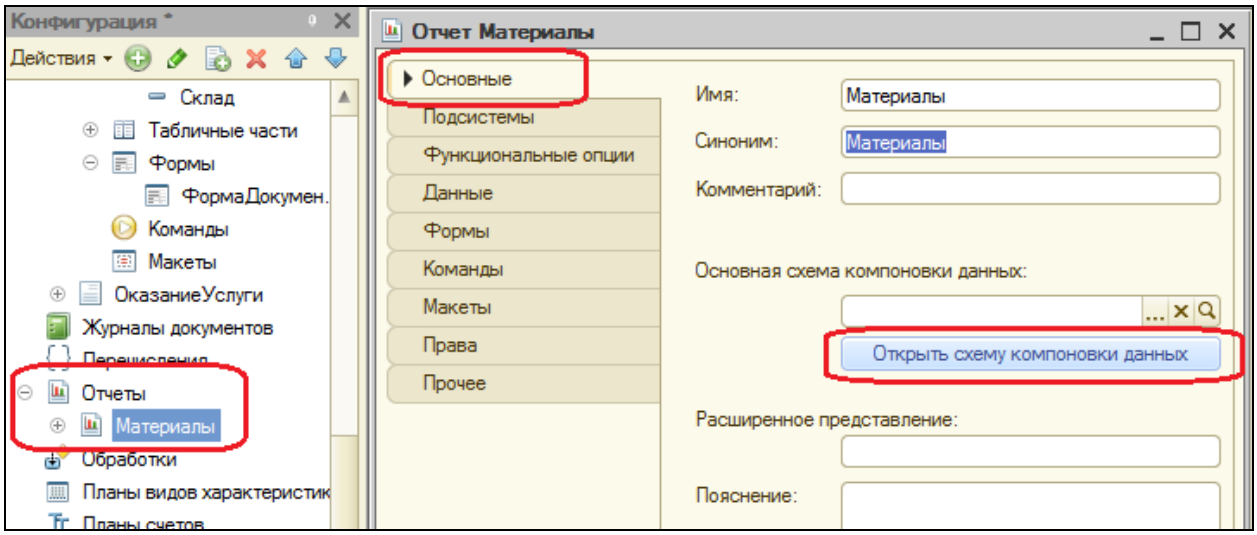
В этой работе Вы познакомитесь с объектом **Отчёт**. Узнаете, для чего он используется, и создадите отчёт, который будет показывать движения и остатки материалов на нашем предприятии.

Объект конфигурации **Отчёт** предназначен для описания алгоритмов, при помощи которых пользователь сможет получать необходимые ему данные. Алгоритм формирования выходных данных описывается при помощи визуальных средств или с использованием встроенного языка. В реальной жизни объектам **Отчёт** соответствуют всевозможные таблицы выходных, сводных данных, диаграммы и т.д.

Добавление отчета

Приступим к созданию отчета, который будет показывать нам приход, расход и остатки материалов.

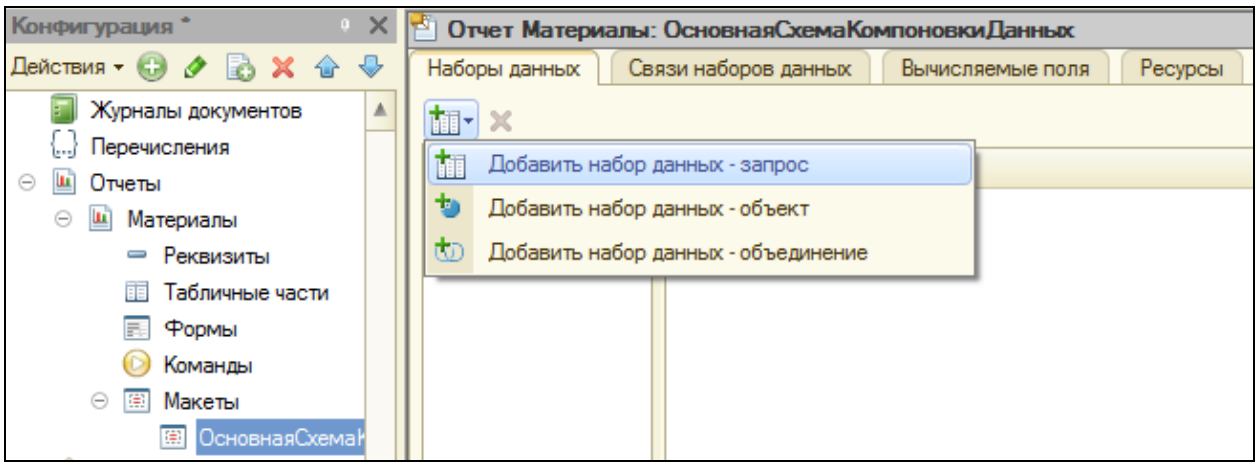
Добавьте новый объект **Отчёт** с именем **Материалы**, нажмите **Открыть схему компоновки данных** – основу для построения любого отчёта.

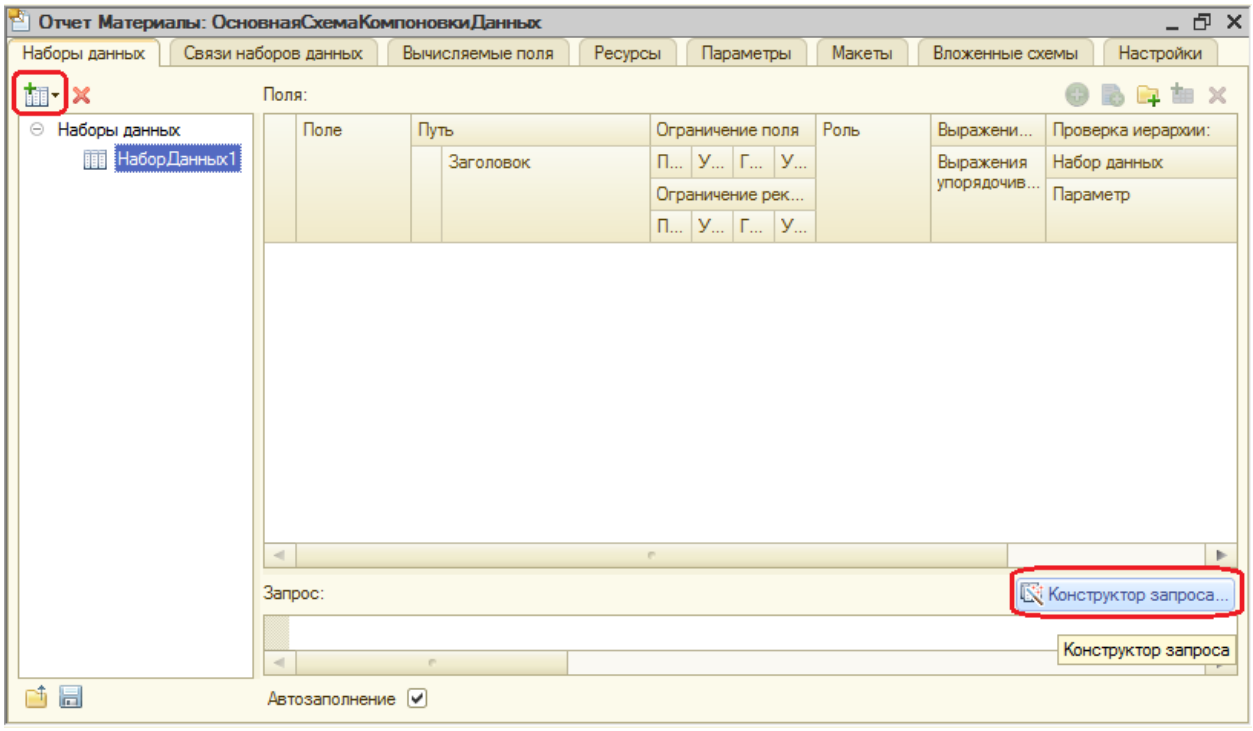


Нажмите **Готово**. Перед Вами откроется конструктор схемы компоновки данных.

Добавьте новый **набор данных – запрос**. Чтобы создать текст запроса, запустите **конструктор запроса** с помощью одноименной кнопки.

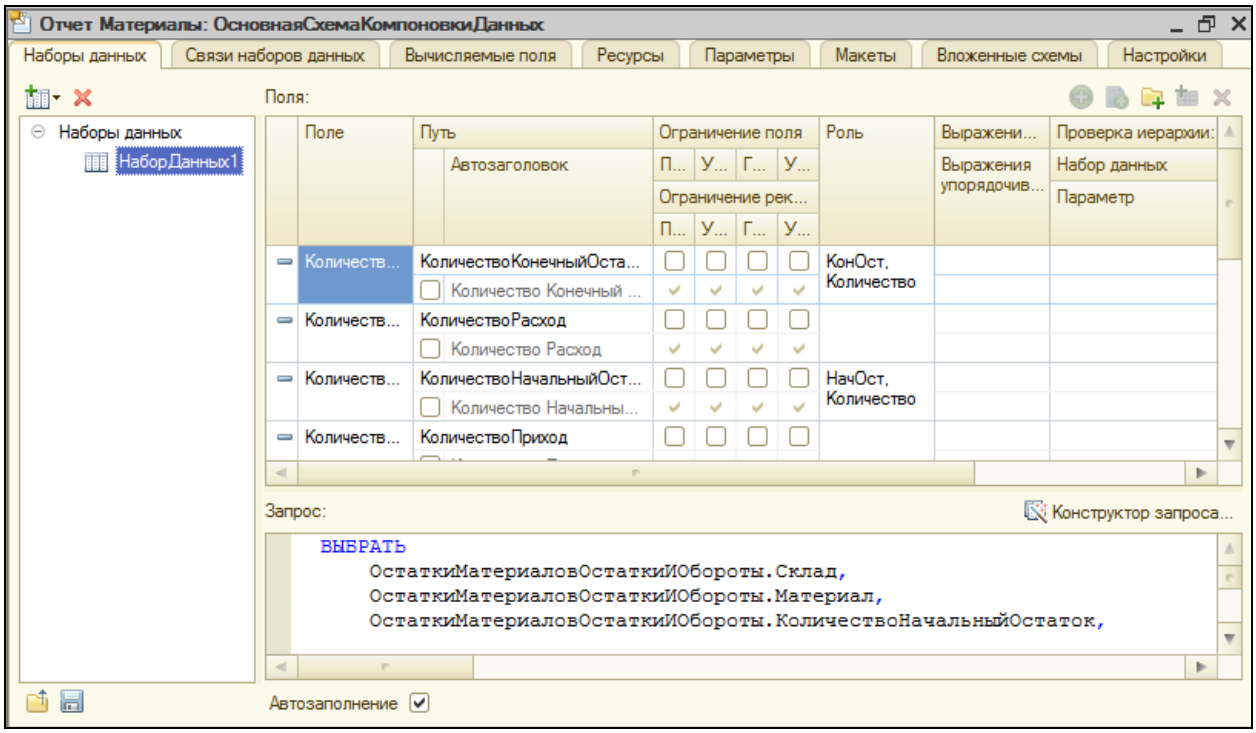
Конструктор запроса – инструмент, позволяющий визуально создать запрос даже не знакомому с языком запросов пользователю.





В списке **База данных** представлены таблицы для создания запроса. На основе их данных можно построить отчёт. Т.к. мы хотим видеть остатки материалов и информацию об их поступлении и расходовании, нас будет интересовать виртуальная таблица **ОстаткиМатериалов.ОстаткиИОбороты**. Раскройте её. Она содержит *измерения* регистра **ОстаткиМатериалов** – Материал, Склад, начальные и конечные остатки, значения прихода и расхода, обороты для всех *ресурсов* регистра. Выберите двойным щелчком из окна **База данных** поля таблицы: **Склад, Материал, КоличествоНачальныйОстаток, КоличествоПриход, КоличествоРасход, КоличествоКонечныйОстаток**. Нажмите ОК. Вы вернулись в конструктор компоновки схемы данных.

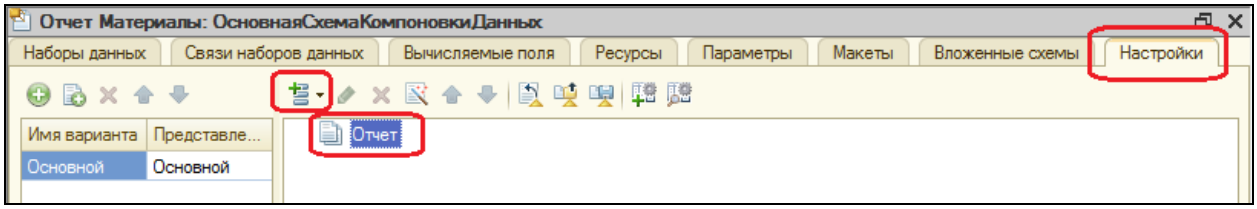
Текст запроса, который был создан с помощью конструктора, платформа поместит в поле Запрос. Это поле представляет собой текстовый редактор, в котором можно отредактировать созданный запрос.



Мы описали, каким образом будут извлекаться данные для отчета, но пока не создали стандартных настроек отображения – ничего не увидим.

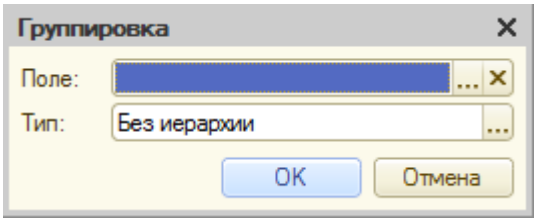
Настройки отчёта

Перейдите на закладку **Настройки**. Выделите элемент **Отчет** и нажмите кнопку **Добавить новую группировку** (или через контекстное меню).



Просто нажмите ОК.

Таким образом мы определим, что в отчете будут выводиться детальные записи из информационной базы – записи, получаемые в результате запроса без итогов.

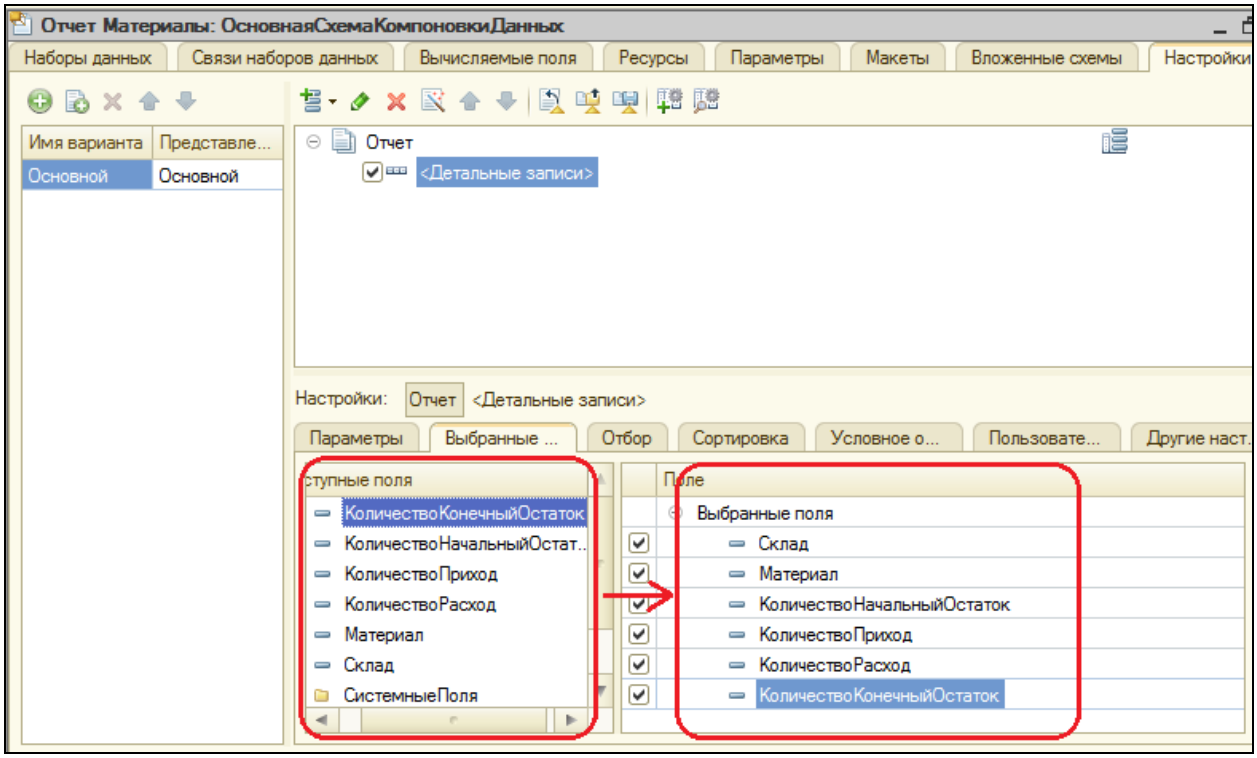


В структуре отчёта появится группировка **Детальные записи**.

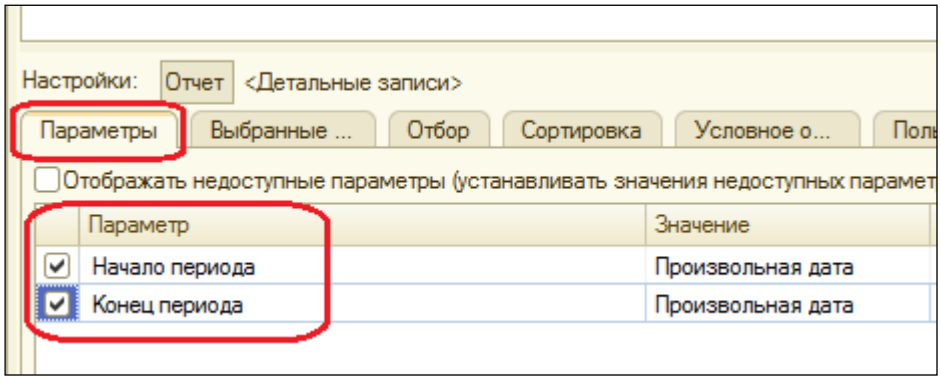
Теперь настроим поля, которые будут выводиться в результат отчёта.

Для этого перейдем в нижнем окне настроек на вкладку **Выбранные поля** и перенесем мышью из списка доступных полей:

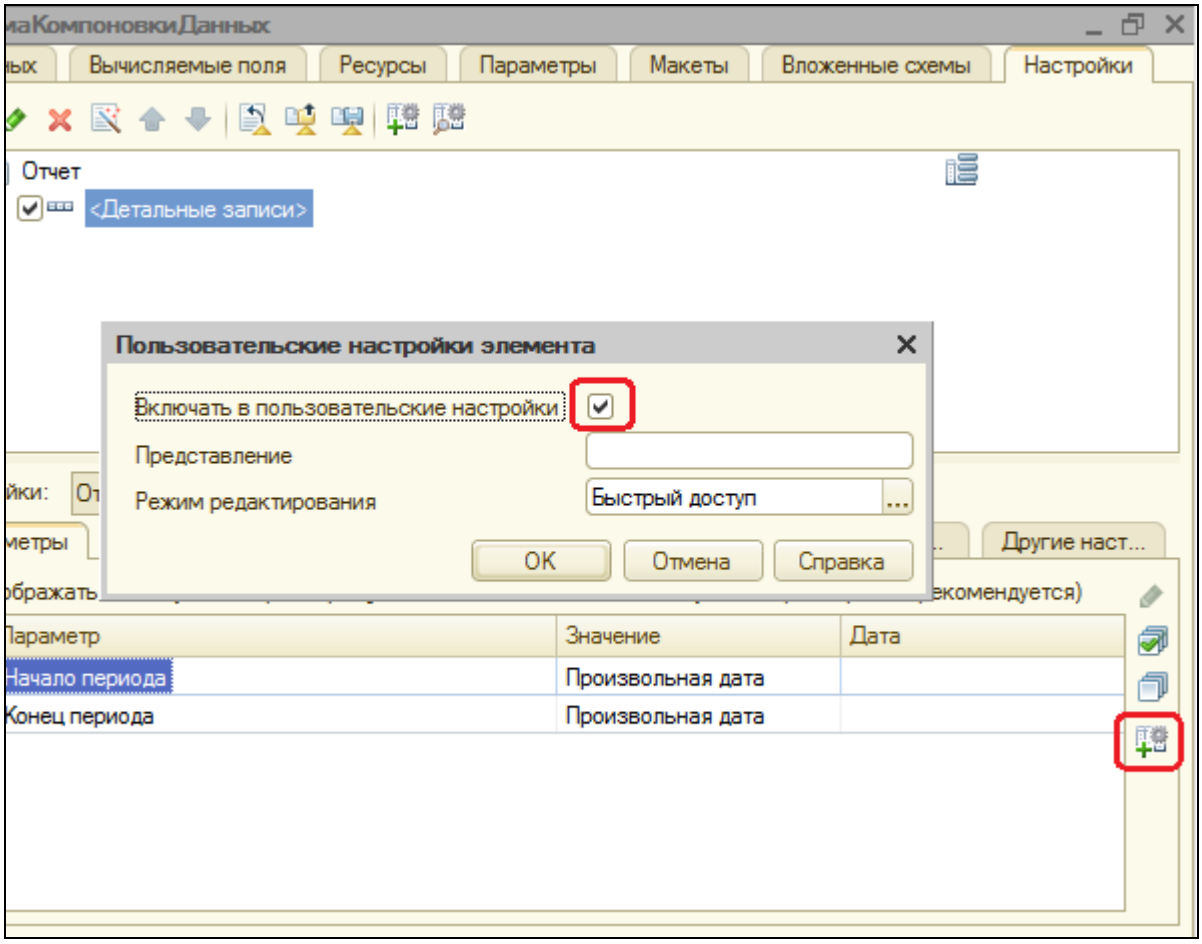
- Склад
- Материал
- КоличествоНачальныйОстаток
- КоличествоПриход
- КоличествоРасход
- КоличествоКонечныйОстаток



Затем перейдите на вкладку **Параметры** и укажите, что параметры отчета **Начало периода** и **Конец периода** будут включены в состав пользовательских настроек и эти настройки будут находиться в форме отчета.



Затем выделите каждый из параметров, нажмите кнопку **Свойства элемента пользовательских настроек** и поставьте флажок **Включать в пользовательские настройки**.

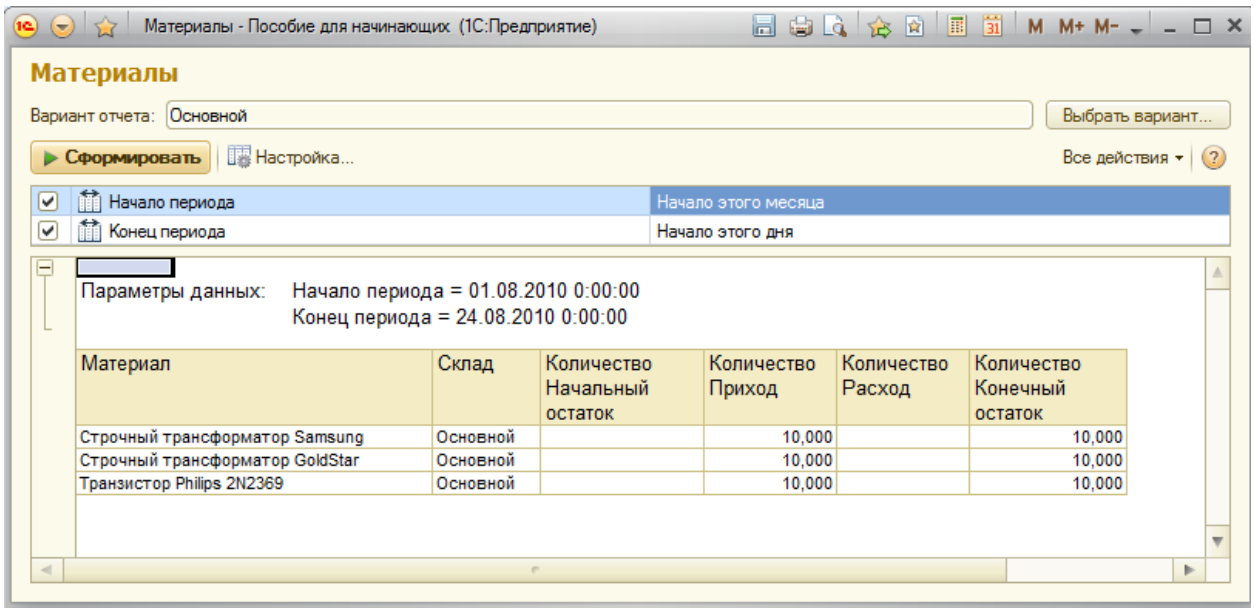


Таким образом, перед формированием отчёта, пользователь сможет задать отчётный период.

Определим, в каких подсистемах будет отображаться наш отчёт. Закройте *конструктор схемы компоновки* данных и перейдите на вкладку **Подсистемы**. Отметим **Бухгалтерия**, **УчетМатериалов**, **ОказаниеУслуг**. Теперь ссылка на наш отчёт попадет в панель действий указанных разделов. Запустите 1С: Предприятие.

Вы увидите, что в панели действий разделов **Бухгалтерия**, **Оказание услуг** и **Учет материалов** появилась новая группа команд для выполнения отчетов и в ней команда для формирования отчета **Материалы**. Выполните её.

Перед Вами открылась сформированная системой форма отчёта. Задайте даты начала (**Начало этого месяца**) и окончания (**Начало этого дня**) отчётного периода и нажмите **Сформировать**.



Если окно отчета пустое, только шапка, выберите другие даты сортировки(раньше текущего дня).

Контрольные вопросы

- ✓ Для чего предназначен объект конфигурации Отчёт
- ✓ Как создать отчёт с помощью конструктора схемы компоновки данных
- ✓ Как отобразить отчёт в разделах прикладного решения

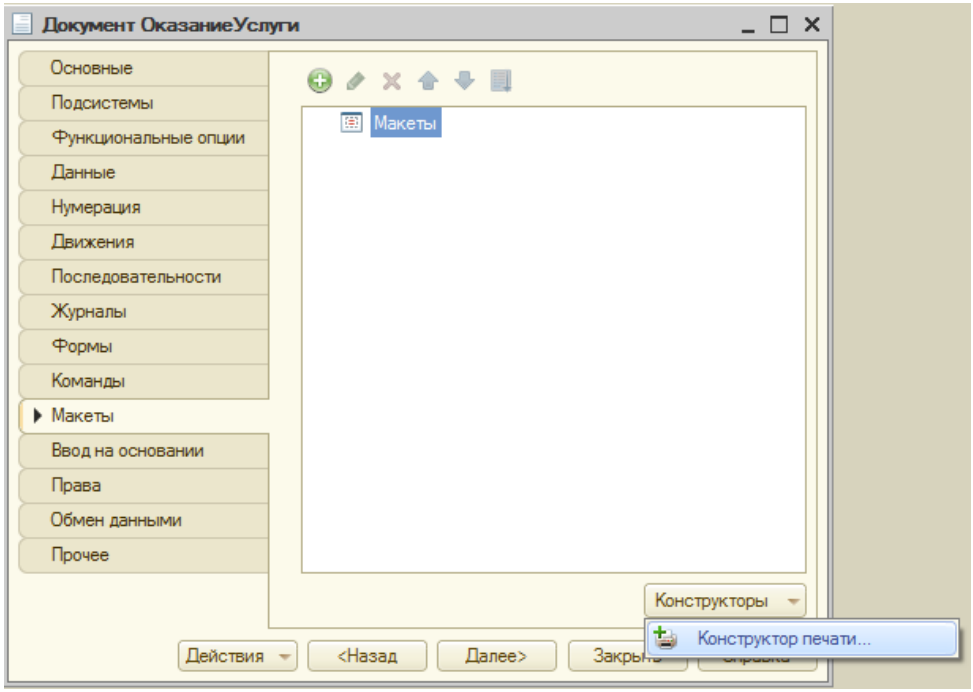
Практическая работа № 7

Макеты. Редактирование макетов и форм (1:10)

В этой практической работе Вы познакомитесь с новым объектом – **Макет**. Узнаете о его назначении и создадите макет документа, на основе которого будет создаваться печатная форма документа.

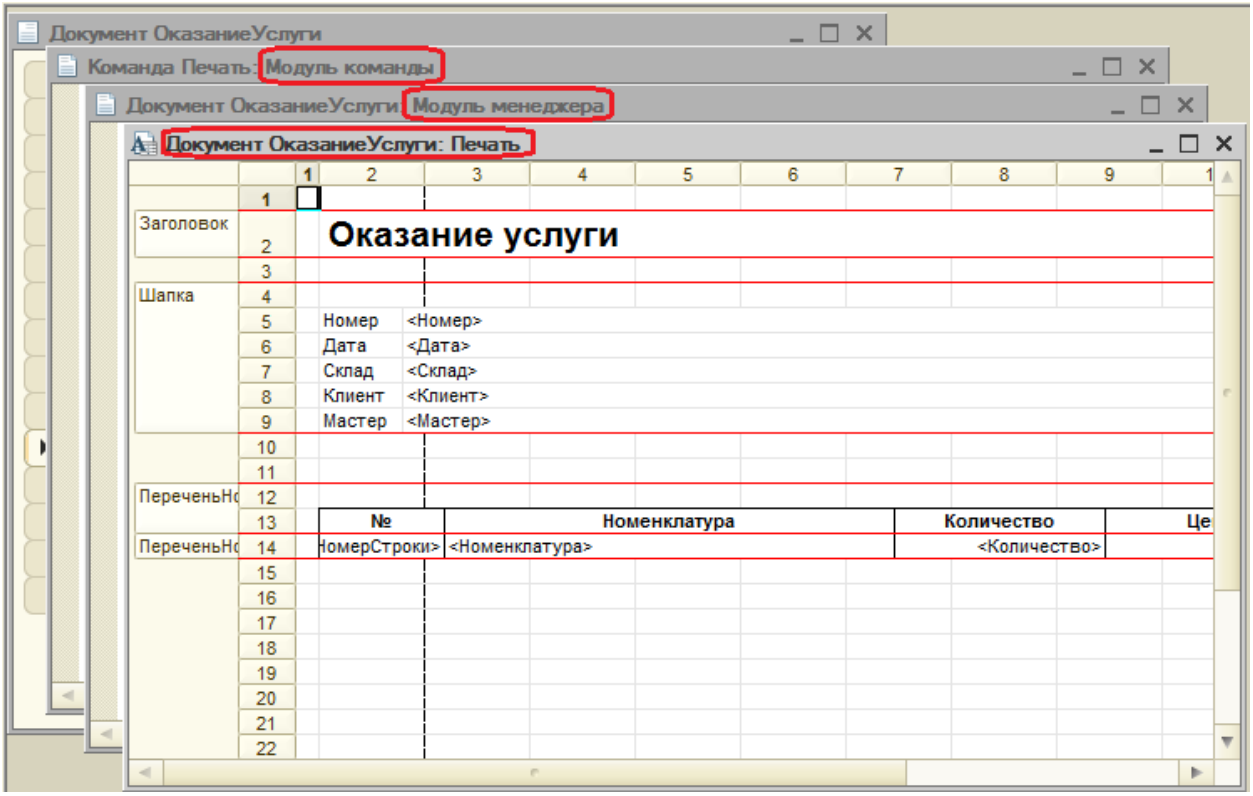
Макет предназначен для хранения различных форм представления данных, различных данных. Может содержать табличный или текстовый документ, двоичные данные, HTML-документ, графическую или географическую схему, схему компоновки данных или макет оформления этой схемы.

Создадим печатную форму документа **ОказаниеУслуги**. Откройте в конфигураторе окно редактирования объекта *Документ* **ОказаниеУслуги**. Перейдите на вкладку **Макеты** и запустите *конструктор печати*.



На первом шаге ничего менять не будем, нажмите **Далее**. На втором шаге перенесите все реквизиты документа из левой части в правую – они будут отображены в шапке печатной формы. Нажмите **Далее**. Также перенесите все реквизиты в табличную часть. На четвертом шаге ничего не переносите – *подвал* (нижняя часть формы) использовать не будем, нажмите **Далее**. Нажмите **ОК**.

Откроется модуль команды Печать, модуль менеджера документа ОказаниеУслуги и макет этого документа.

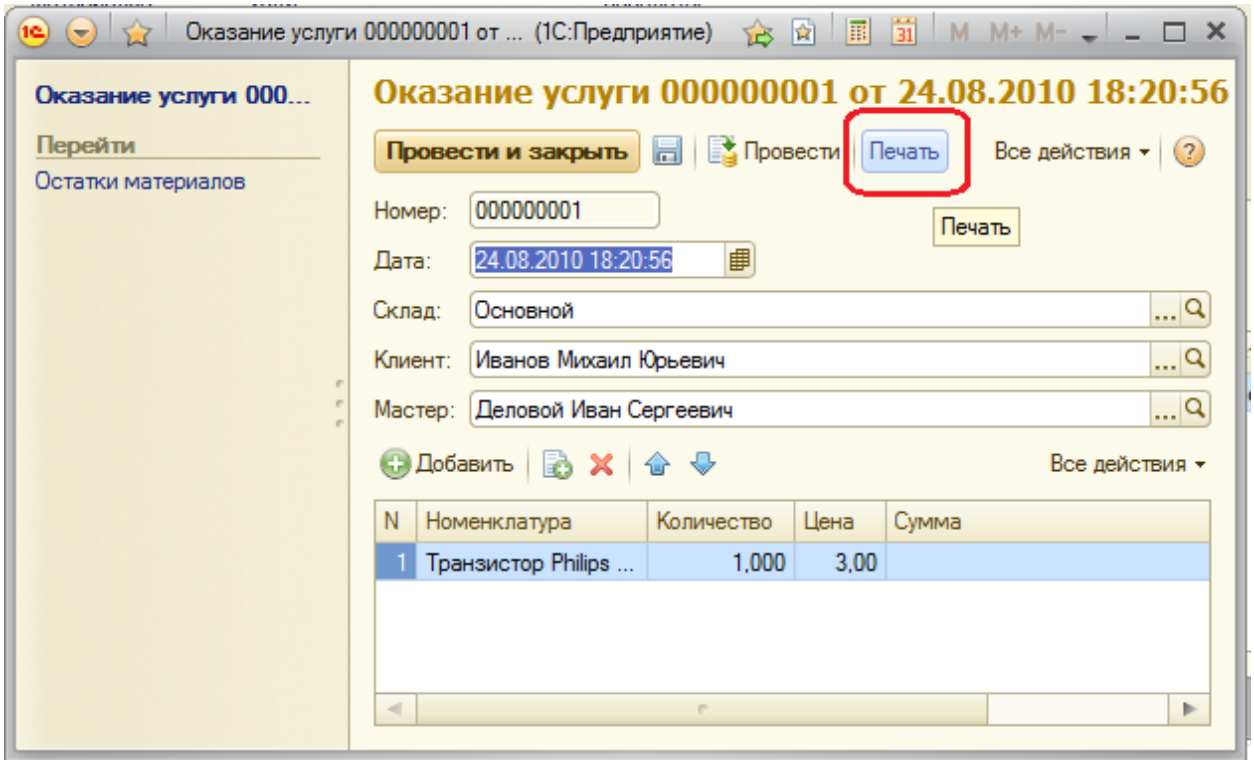


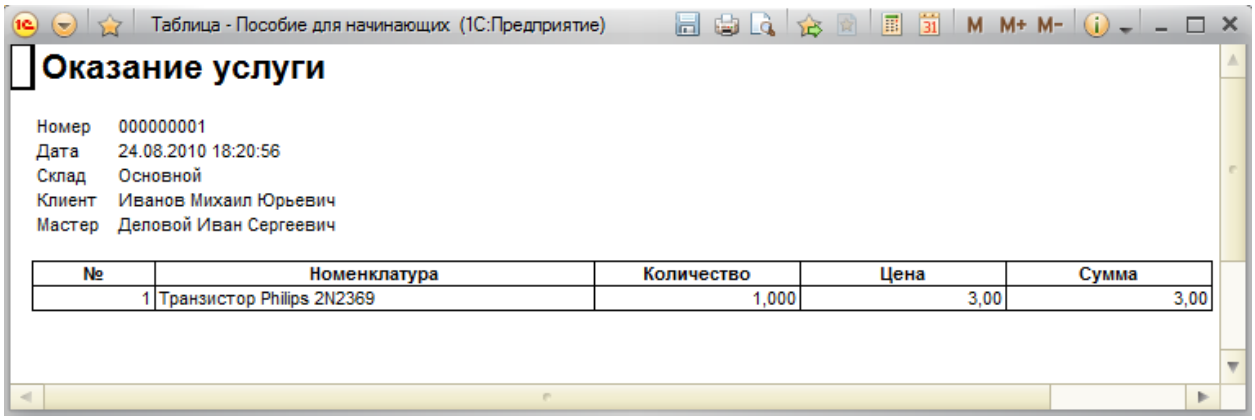
Таким образом, конструктор создал:

- *Макет* печатной формы документа **ОказаниеУслуги** с именем **Печать**
- Команду документа **ОказаниеУслуги** с именем **Печать**. В модуль этой команды помещен обработчик, вызывающий процедуру печати документа, выполняющуюся на сервере. Сама процедура печати помещена в модуль менеджера документа.
- В командную панель формы документа **ОказаниеУслуги** помещена команда **Печать** для формирования печатной формы документа. При этом команда **Печать** принадлежит всему документу, а не одной его форме, поэтому может использоваться в любой его форме.

Запустите 1С: Предприятие в режиме отладки и откройте документ **Оказание услуги №1**. Нажмите на появившуюся кнопку **Печать**.

Откроется печатная форма документа. Единственное, чего в ней не хватает – итоговой суммы документа.





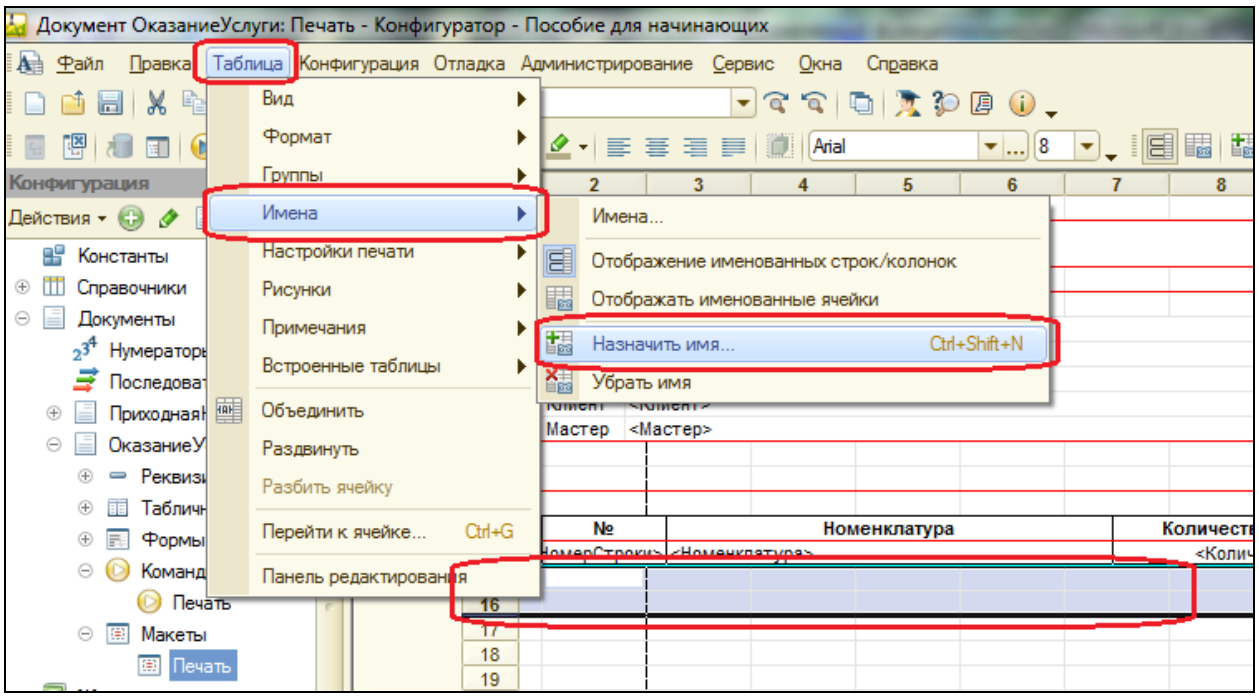
Редактирование макета

Вернитесь в конфигуратор. Добавим итоговую сумму в печатную форму.

Раскройте дерево документа **ОказаниеУслуги** и дважды щелкните на макете **Печать** (Если закрыли перед этим).

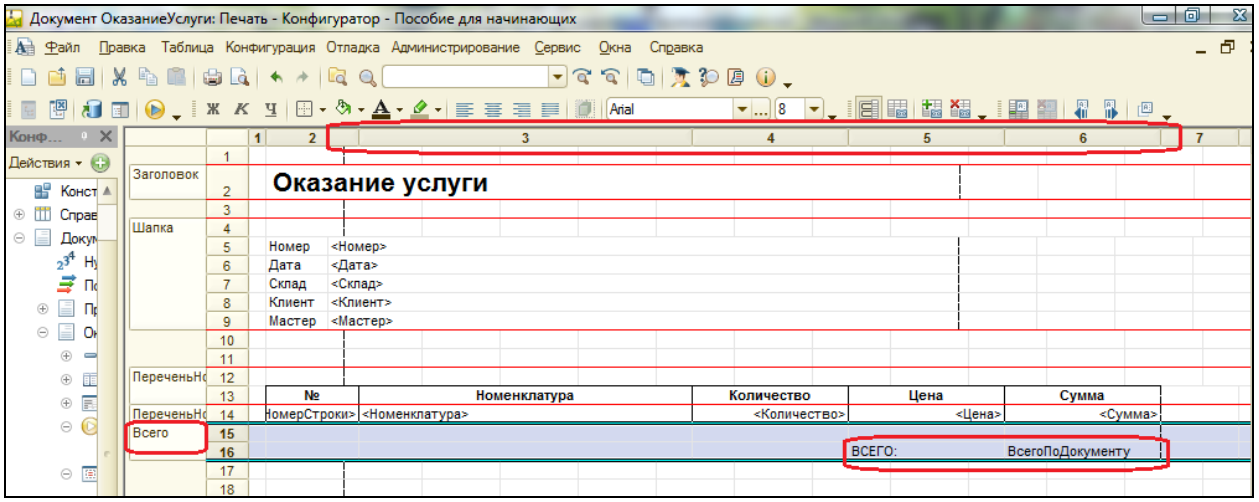
Макет документа состоит из именованных областей, которые в определенном порядке выводятся на печать. Именованные области слева созданы конструктором, но Вы можете сами создавать или удалять области, переименовывать их и т.д.

Добавим новую область для вывода итоговой суммы документа. Выделите мышью две пустые строки под табличной частью документа и выполните **Таблица** – **Имена** – **Назначить имя**. Назовите область **Всего** и нажмите ОК.



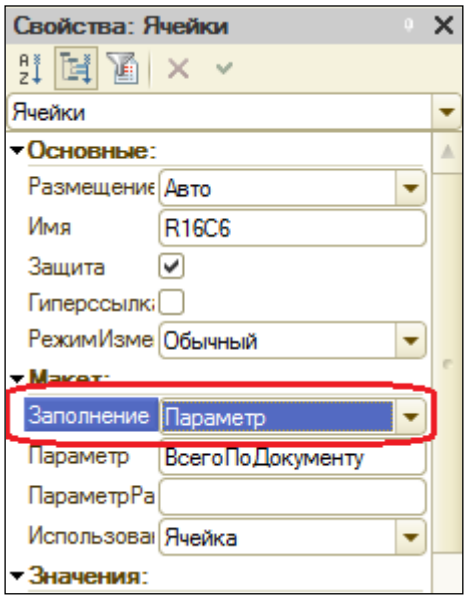
Слева появилась новая область **Всего**. Но ширина области **Всего** не совпадает с остальными, изменим её. Выделите две строчки созданной области **Всего**, потяните мышью в заголовке таблицы за правую границу колонки 2 так, чтобы её ширина совпала с шириной колонки № в шапке документа. Согласитесь, когда платформа предложит создать новый формат строк. Теперь сделайте такое же изменение ширины для колонок 3,4, 5, 6, чтобы они соответствовали колонкам в шапке.

В созданной области во второй строчке колонки **Цена** напишите **ВСЕГО:**, а в колонке Сумма напишите **ВсегоПоДокументу**.



Проконтролировать вид печатной формы можно с помощью кнопки предварительного просмотра  .

Выделите ячейку, где Вы записали **ВсегоПоДокументу**, вызовите контекстное меню – **Свойства**. В свойстве **Заполнение** укажите, что в этой ячейке будет находиться не текст, а параметр. Подробнее об этом выборе.



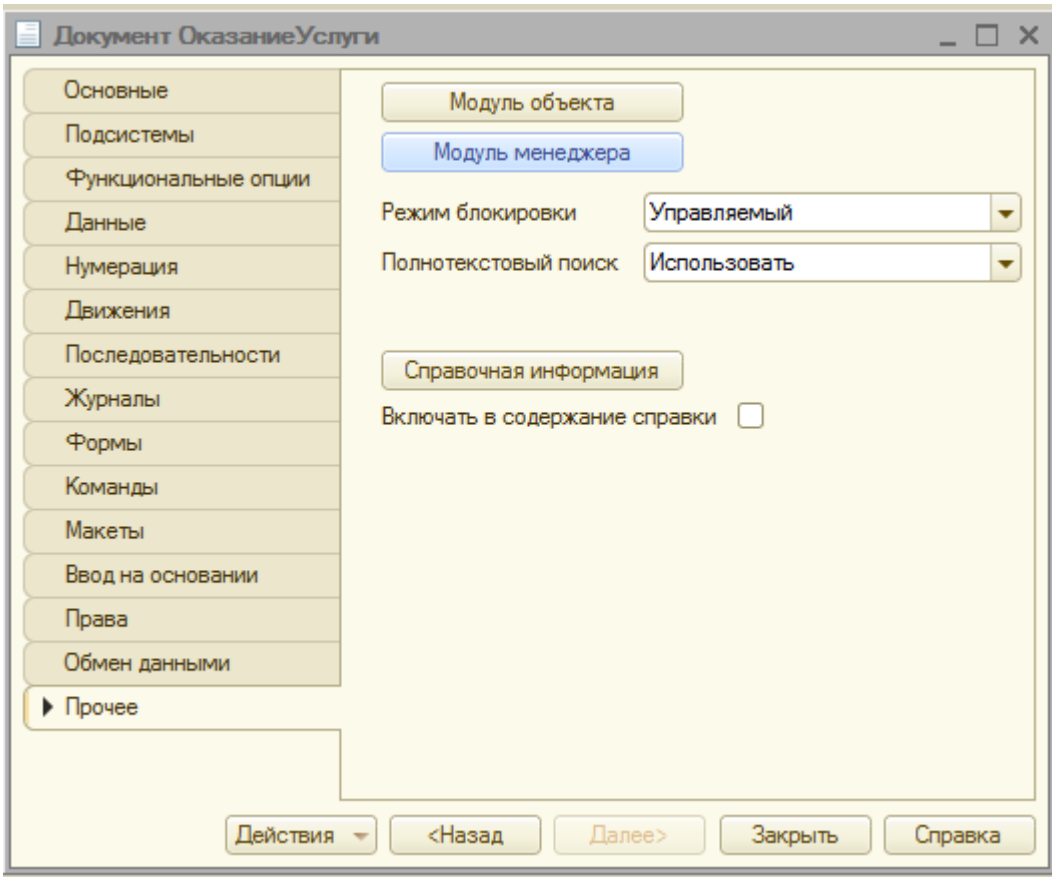
Текст, содержащийся в ячейке, будет показан на экране.

Параметр будет заменен некоторым значение, которое может быть присвоено ему средствами встроенного языка. Текст, содержащийся в ячейке, является именем этого параметра.

Шаблон – текстовая строка, в определенные места которой будут вставлены значения параметров.

Откройте модуль менеджера документа **ОказаниеУслуги**.

Для этого перейдите на закладку **Прочее** окна редактирования объекта **ОказаниеУслуги** и нажмите кнопку **Модуль менеджера** (или закройте макет, менеджер должен быть открыт под ним).

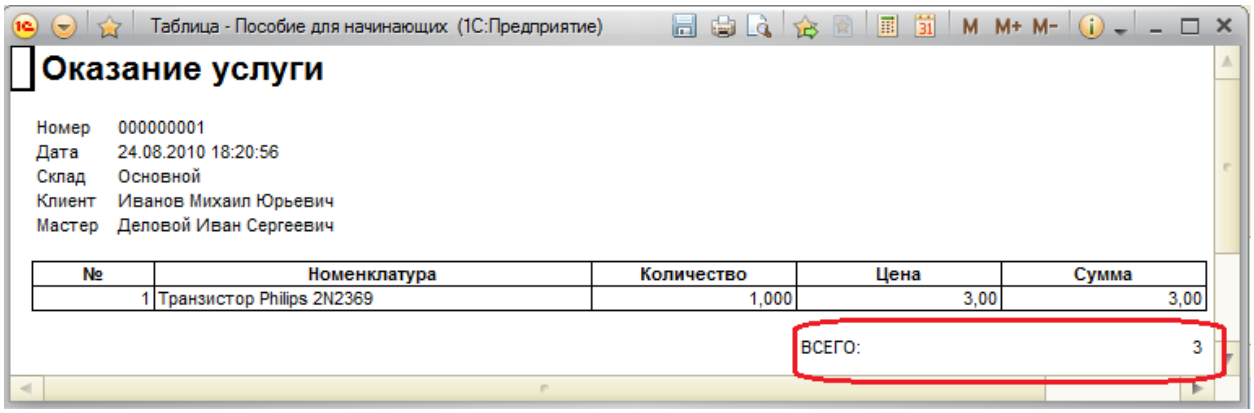


Перед Вами часть

процедуры печати, выделенные **жирным** строчки – <u>новые</u>.

```
ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
	Шапка = Макет.ПолучитьОбласть("Шапка");
	ОбластьПереченьНоменклатурыШапка = Макет.ПолучитьОбласть("ПереченьНоменклатурыШапка");
	ОбластьПереченьНоменклатуры = Макет.ПолучитьОбласть("ПереченьНоменклатуры");
	ОбластьИтог = Макет.ПолучитьОбласть("Всего");
	ТабДок.Очистить();

	ВставлятьРазделительСтраниц = Ложь;
	Пока Выборка.Следующий() Цикл
		Если ВставлятьРазделительСтраниц Тогда
			ТабДок.ВывестиГоризонтальныйРазделительСтраниц();
		КонецЕсли;
		ТабДок.Вывести(ОбластьЗаголовок);
		Шапка.Параметры.Заполнить(Выборка);
		ТабДок.Вывести(Шапка, Выборка.Уровень());
		ТабДок.Вывести(ОбластьПереченьНоменклатурыШапка);
		ВыборкаПереченьНоменклатуры = Выборка.ПереченьНоменклатуры.Выбрать();
		СуммаИтог = 0;
		Пока ВыборкаПереченьНоменклатуры.Следующий() Цикл
	ОбластьПереченьНоменклатуры.Параметры.Заполнить(ВыборкаПереченьНоменклатуры);
			ТабДок.Вывести(ОбластьПереченьНоменклатуры,
ВыборкаПереченьНоменклатуры.Уровень());
			СуммаИтог = СуммаИтог + ВыборкаПереченьНоменклатуры.Сумма;
		КонецЦикла;
		ОбластьИтог.Параметры.ВсегоПоДокументу = СуммаИтог;
		ТабДок.Вывести(ОбластьИтог);
		ВставлятьРазделительСтраниц = Истина;
	КонецЦикла;
```

Запустите 1С: Предприятие и посмотрите форму печати документа **Оказание услуги №1**.

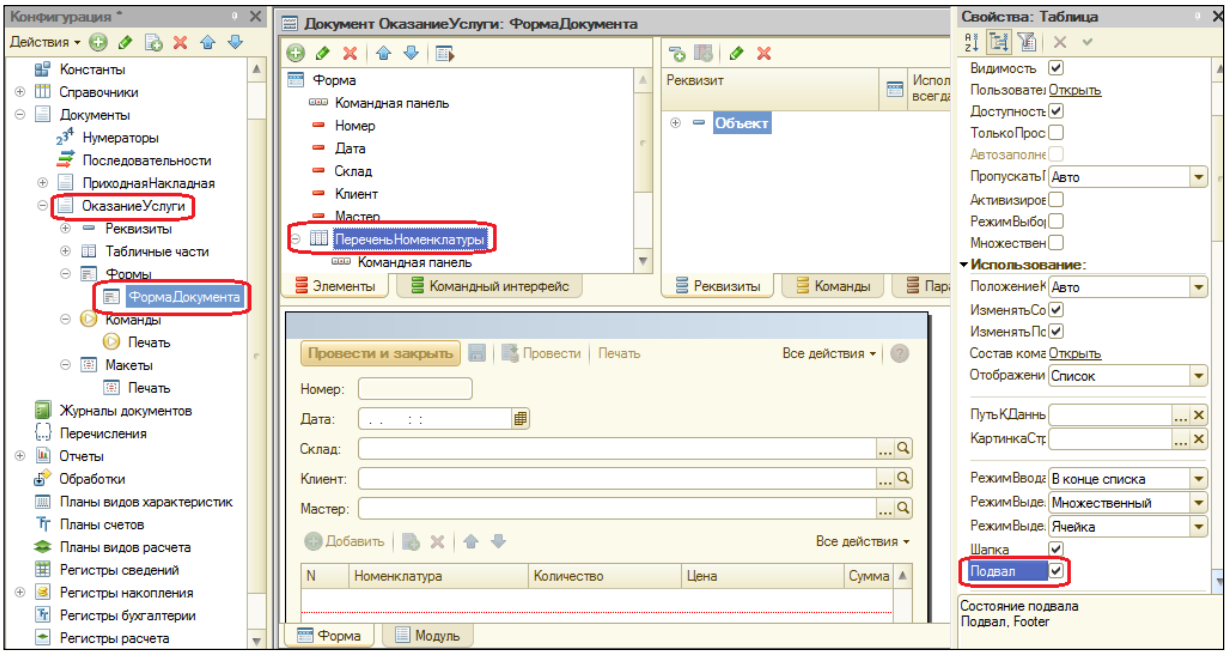


А теперь, чтобы документ ОказаниеУслуги выглядел законченным, добавим итоговую сумму по документу и на экранную форму, чтобы пользователь мог видеть её в процессе заполнения табличной части документа.

Редактирование формы

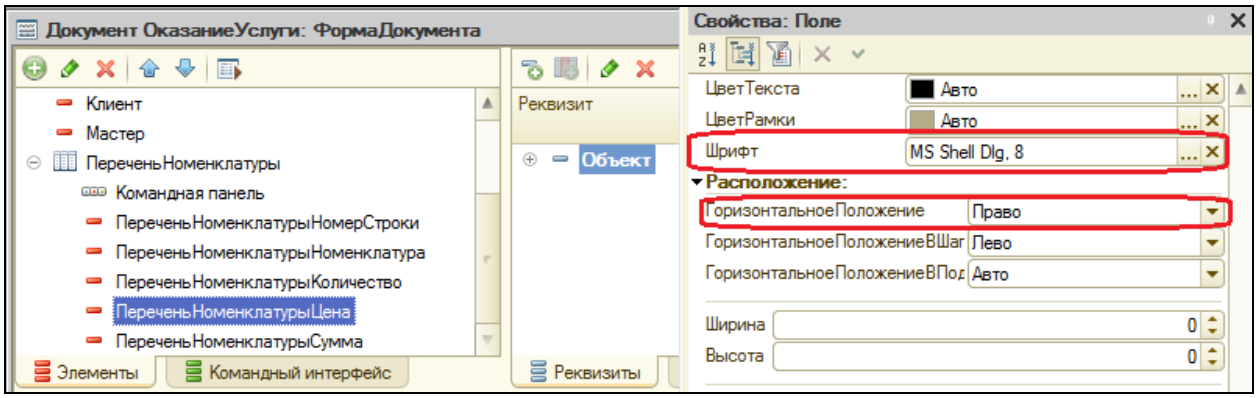
Вернитесь в конфигуратор. Удобно, если в процессе создания документа можно было оперативно, не печатая его, знать итоговую сумму по документу (как в Excel). Для этого внесем небольшие изменения в форму документа **ОказаниеУслуги**.

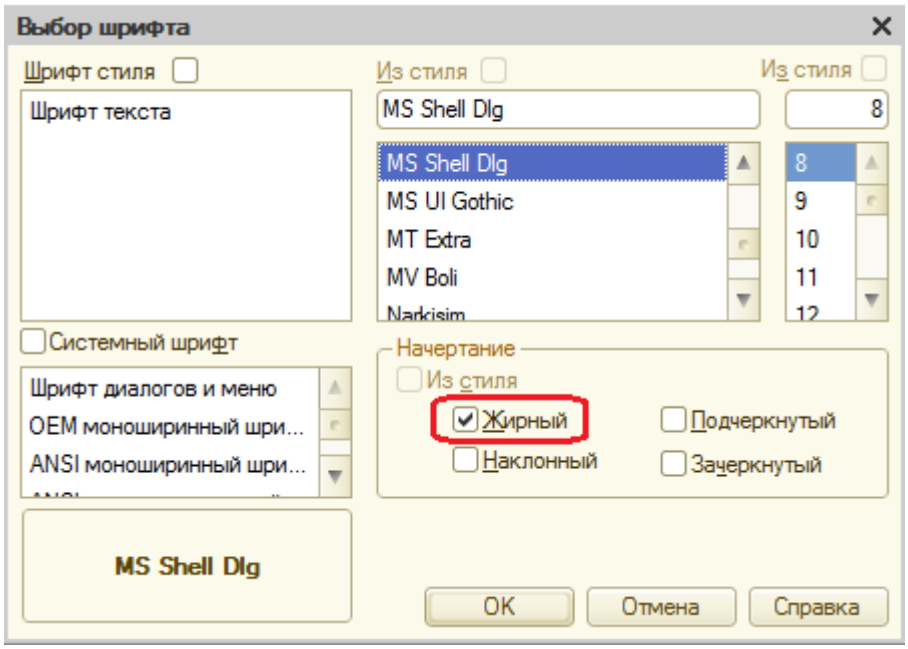
Откройте в дереве объектов форму документа **ОказаниеУслуги**, дважды щелкнув по ней. В левом верхнем окне вызовите свойства пункта **ПереченьНоменклатуры**. Установите свойство **Подвал**, которое определяет наличие подвала (нижней части) у таблицы формы.



Затем откройте свойства элемента формы **ПереченьНоменклатурыЦена** и установите:

- **Текст** подвала – **Всего:**
- **Горизонтальное положение** – **Право**,
- **Шрифт** подвала – **Жирный**.

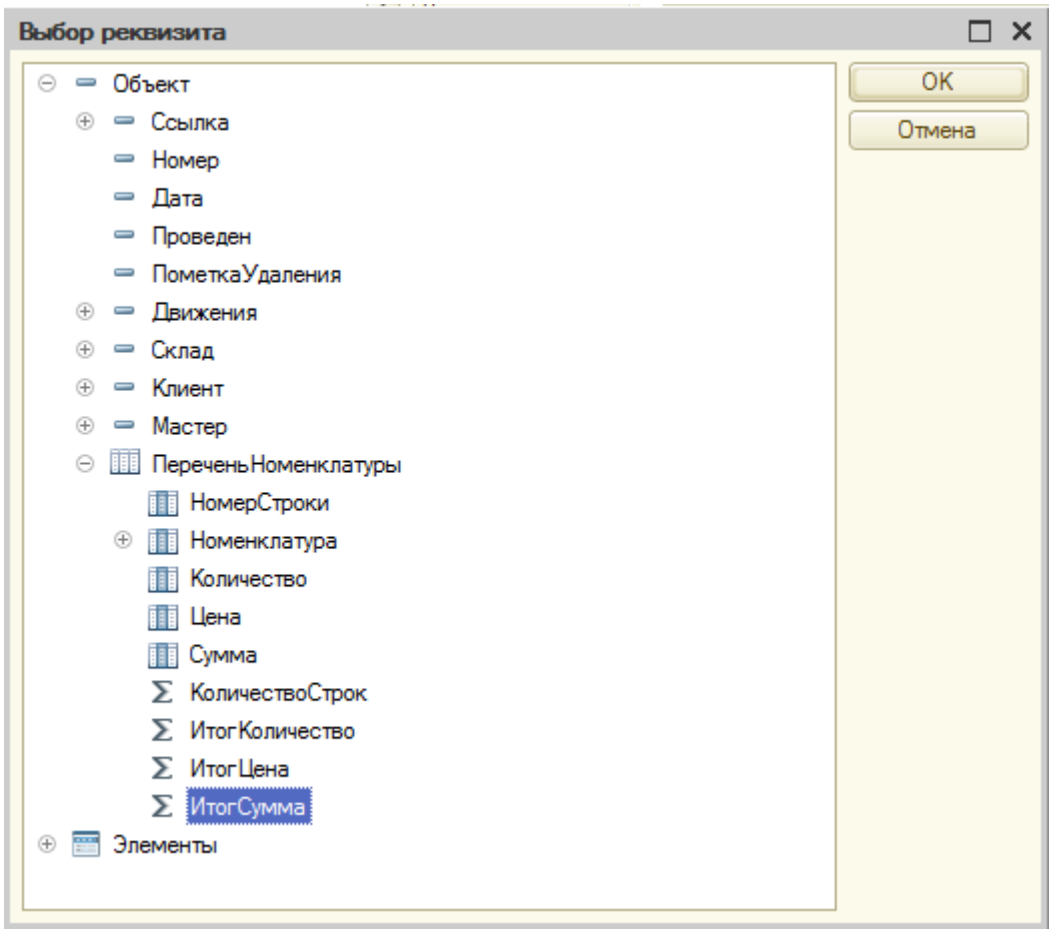


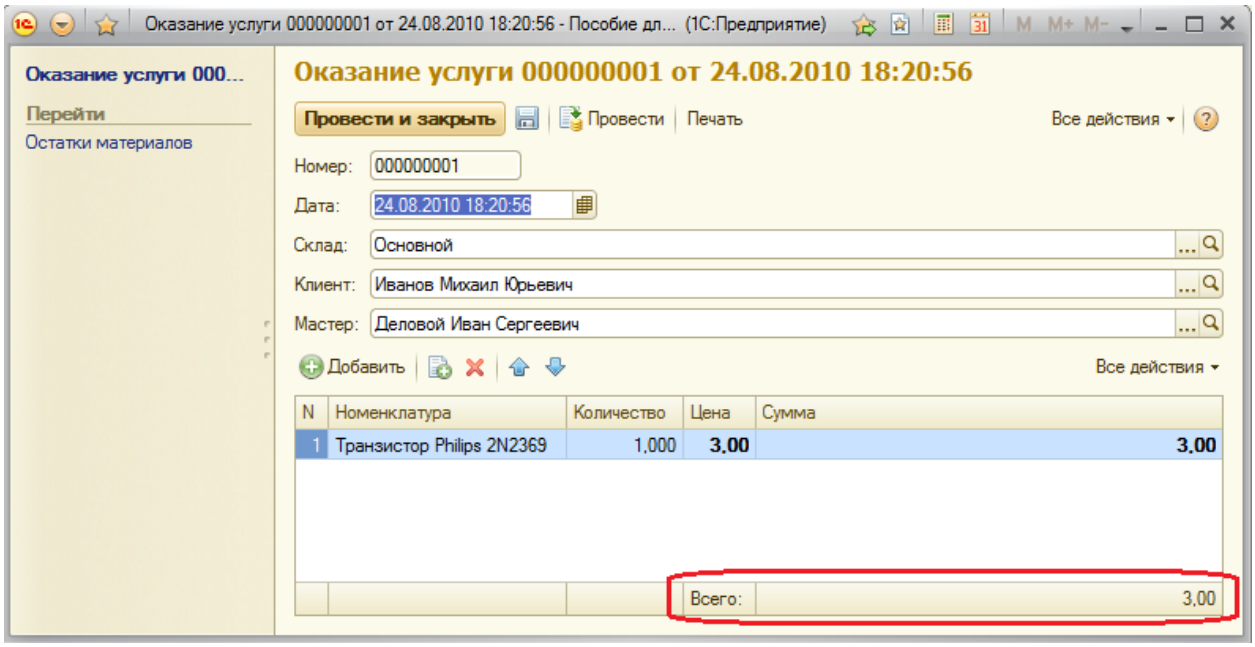


После этого откройте свойства элемента **ПереченьНоменклатурыСумма**, установите: **Горизонтальное положение** – **Право**, **Шрифт** – **жирный**.

Для того, чтобы в подвале колонки **Сумма** отображался итог по ней, нажмите кнопку выбора свойств в поле **ПутьКДаннымПодвала**. Раскройте дерево реквизитов объекта и выберите элемент **ИтогСумма**.

Нажмите ОК и запустите 1С: Предприятие в режиме отладки. Откройте документ **Оказание услуги №1**. Вы увидите, что по колонке Сумма в табличной части документа подсчитывается общий итог документа.





Контрольные вопросы

- ✓ Для чего предназначен объект Макет
- ✓ Что такое конструктор печати
- ✓ Как создать макет с помощью конструктора печати
- ✓ Как изменить макет документа
- ✓ Какая разница в заполнении ячейки табличного документа между текстом, параметром и шаблоном
- ✓ Как изменить внешний вид и поведение элемента формы
- ✓ Как отобразить сумму по колонке таблицы

Практическая работа № 8

Периодические регистры сведений (0:50)

В этой работе Вы познакомитесь с объектом **Регистр сведений**, точнее с одним из его видов – *периодическим регистром сведений*. Вы создадите один периодический регистр сведений и покажете каким образом можно использовать его данные средствами встроенного языка.

В нашей фирме существует перечень услуг, который определяет стоимость каждой услуги. Но она меняется со временем, поэтому может сложиться ситуация, когда нам нужно будет изменить один из ранее проведенных документов Оказание услуги. В этом случае мы не сможем получить правильную стоимость услуги, поскольку в реквизите справочника будет храниться последнее введенное значение. Кроме этого, может потребоваться видеть зависимость прибыли от изменения стоимости оказываемых услуг. И тогда необходимо будет иметь возможность анализировать изменение стоимости услуг во времени.

Поэтому для хранения стоимости услуг мы используем новый для нас объект – **Регистр сведений**. Он предназначен для описания структуры хранения данных в разрезе нескольких измерений.

Принципиальное отличие *Регистра сведений* от *Регистра накопления* в том, что каждое движение *регистра сведений* устанавливает новое значение ресурса, в то время как движение *регистра накопления* изменяет существующее значение ресурса. По этой причине *регистр сведений* может хранить любые данные, а не только числовые как *регистр накопления*.

Еще одна важная особенность *регистра сведений* – способность хранить данные с привязкой ко времени. Благодаря этому *регистр сведений* может хранить историю изменения данных. Такой регистр называют *периодическим регистром сведений*.

Периодичность регистра сведений можно определить одним из значений:

- В пределах секунды
- В пределах дня
- В пределах месяца
- В пределах квартала
- В пределах года

- В пределах регистратора (если установлен режим записи Подчинение регистратору).

Периодический регистр сведений всегда содержит служебное поле **Период**, добавляемое автоматически. Оно имеет тип **Дата** и служит для указания факта принадлежности записи к какому-либо периоду. При записи данных в регистр, платформа всегда приводит значение этого поля к началу того периода, в который он попадает.

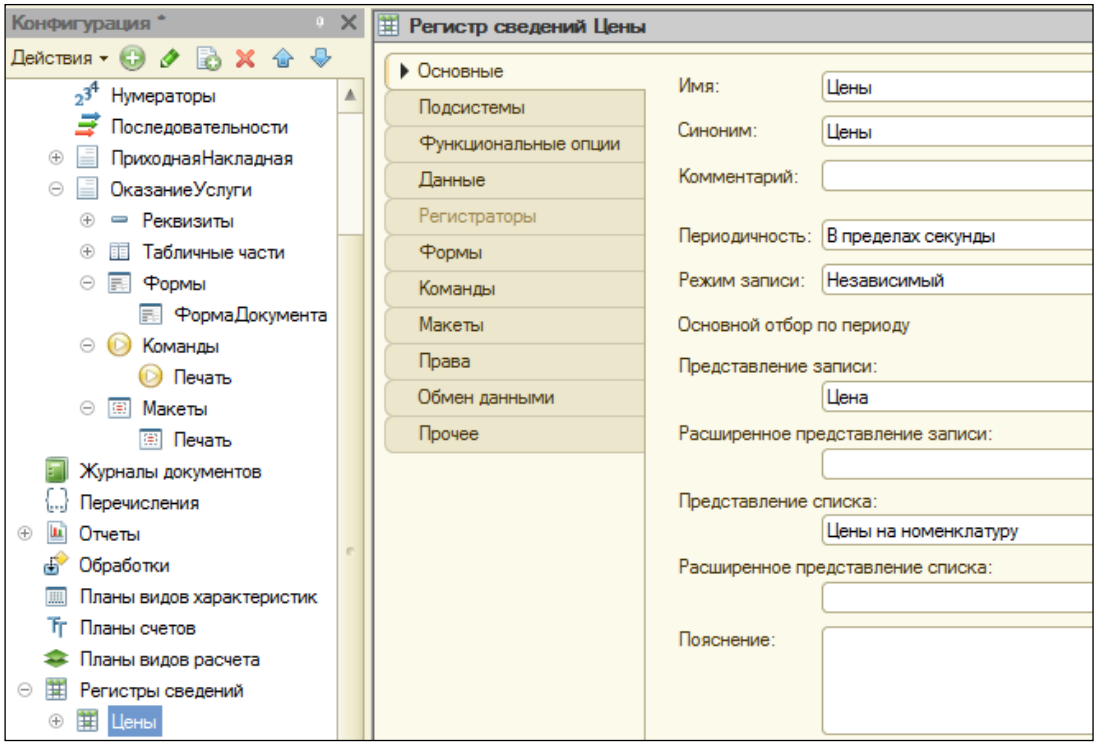
Например, если в регистр сведений с периодичностью в пределах месяца записать данные, в которых период указан как 25.08.2010, то регистр сохранит эти данные со значением периода 01.08.2010.

Ключевым полем в регистре сведений является совокупность значений измерений регистра и периода.

Регистр сведений, не использующий подчинение регистратору, называют *независимым регистром сведений*.

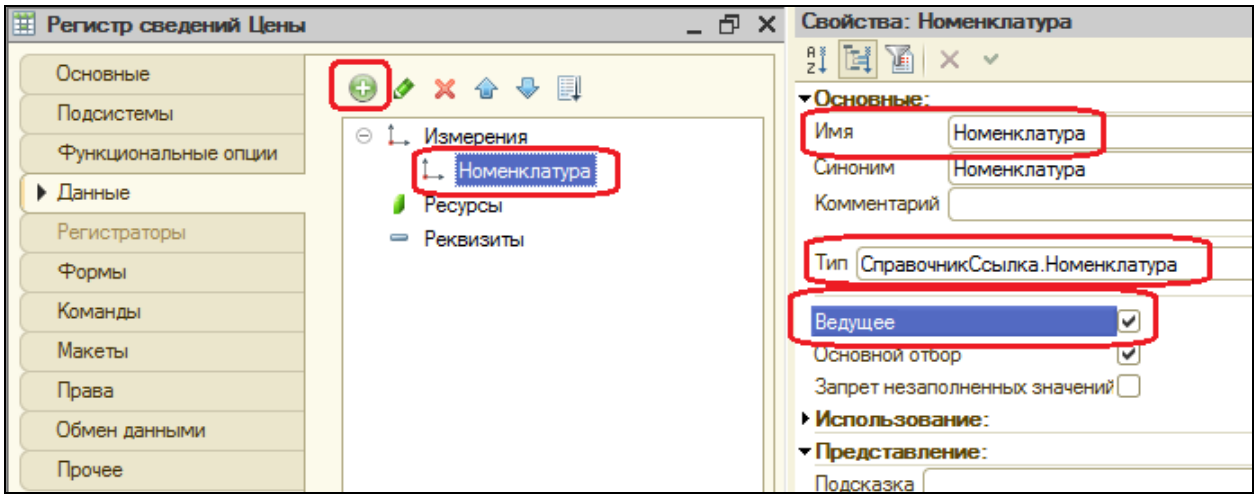
Добавление периодического регистра сведений

Добавьте новый объект конфигурации *Регистр сведений* с именем **Цены**. **Периодичность** – **в пределах секунды**. **Представление записи** – **Цена**, **Представление списка** – **Цены на номенклатуру**. Обратите внимание на свойство **Режим записи**. По умолчанию оно имеет значение **Независимый**, т.е. мы создаем независимый регистр сведений и можем в дальнейшем вводить в него данные без использования регистратора, вручную.

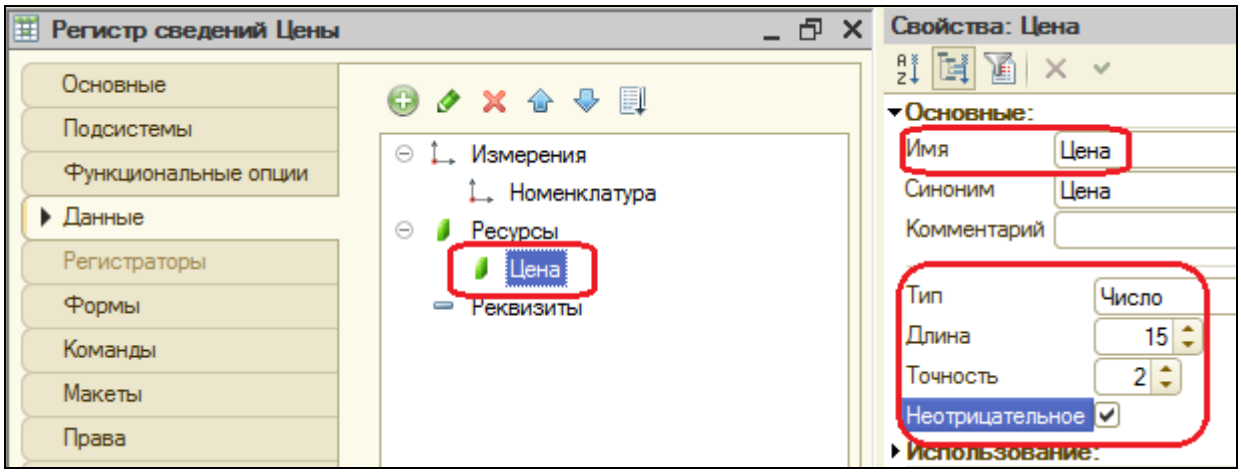


Перейдите на вкладку **Подсистемы**, отметьте **УчетМатериалов, ОказаниеУслуг** и **Бухгалтерия**. Перейдите на вкладку **Данные**. Создайте измерение **Номенклатура** с типом **СправочникСсылка.Номенклатура**.

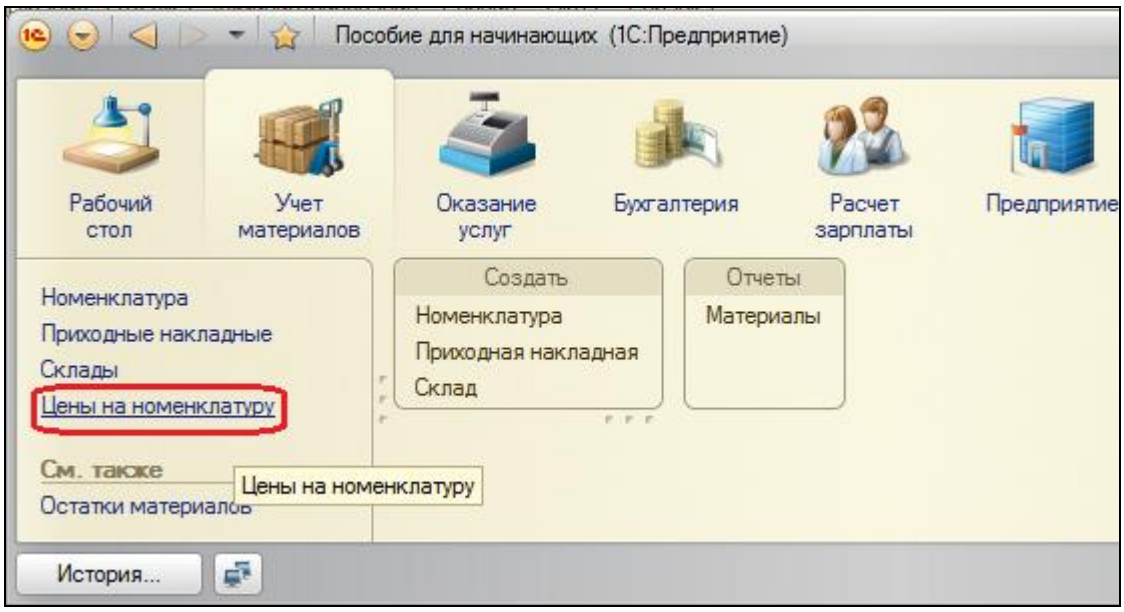
Укажите, что измерение будет **ведущим**. Оно означает, что при удалении объекта, все записи регистра сведений по этому объекту будут автоматически удалены. Также благодаря этому свойству, в панели навигации появится ссылка для перехода к записям этого регистра.



Создайте также ресурс **Цена**, тип **Число**, длина 15, точность 2, неотрицательное.



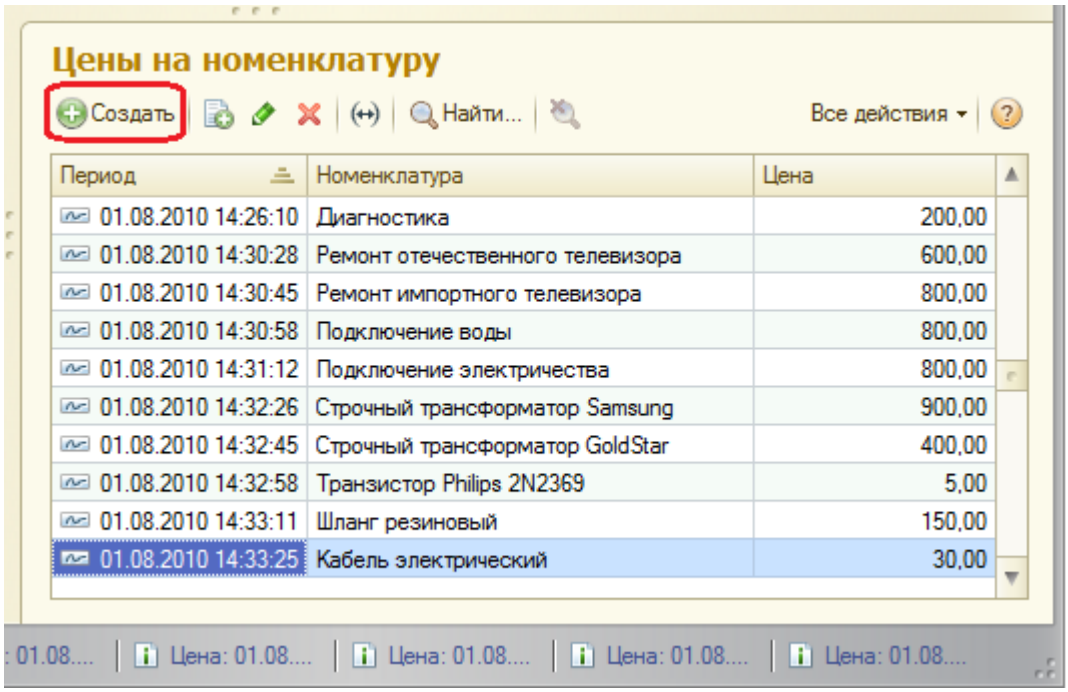
Запустите 1С: Предприятие. Вы увидите, что в разделах **Бухгалтерия**, **Оказание услуг** и **Учет материалов** появилась команда для открытия списка регистра **Цены на номенклатуру**. Эта команда доступна по умолчанию, т.к. в отличие от регистров накопления предполагается изменение данных регистра пользователем.



Создание записей в регистре сведений

Откройте регистр **Цены на номенклатуру** и нажмите кнопку **Создать**.

Заполните регистр ценами всех услуг и материалов, задавая период задним числом, чтобы оно было раньше даты создания документа об оказании услуг (в моем случае дата создания документа – 24.08.2010, поэтому я устанавливаю 01.08.2010).



Теперь мы можем заранее установить новые цены и быть уверены в том, что новые цены вступят в действие не раньше указанного для них времени.

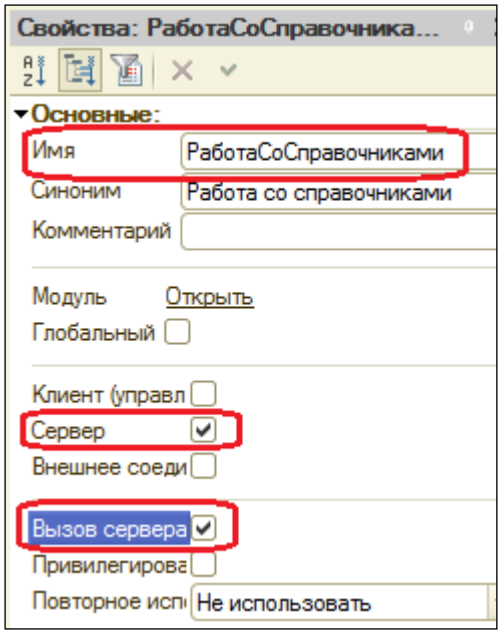
Автоматическая подстановка цены в документ при выборе номенклатуры

Когда мы создаем или изменяем документ **ОказаниеУслуги** и добавляем какую-либо номенклатуру, нам хочется, чтобы одновременно с этим в документ подставлялась бы сразу актуальная цена этой номенклатуры, полученная из регистра сведений и соответствующей дате документа. Для этого нам нужно написать функцию, которая будет возвращать нам актуальную цену, а затем вызывать эту функцию в момент добавления номенклатуры. Поскольку такая возможность может понадобиться и в других документах, поместим функцию в общий модуль.

В конфигураторе, ветка **Общие – Общие** модули добавьте новый модуль с именем **РаботаСоСправочниками**.

Как видите, у модуля по умолчанию установлен флажок **Сервер**. Это означает, что модуль будет компилироваться на сервере.

Установите флажок **Вызов сервера**, чтобы процедуры и функции этого модуля можно было вызывать с клиента.



Поместите в модуль следующий текст:

```
Функция РозничнаяЦена(АктуальнаяДата, ЭлементНоменклатуры) Экспорт

	// Создать вспомогательный объект Отбор

	Отбор = Новый Структура("Номенклатура", ЭлементНоменклатуры);

	// Получить актуальные значения ресурсов регистра

	ЗначенияРесурсов = РегистрыСведений.Цены.ПолучитьПоследнее(АктуальнаяДата, Отбор);

	Возврат ЗначенияРесурсов.Цена;

КонецФункции
```

Далее нам необходимо обеспечить автоматическое заполнение поля **Цена** после того, как пользователь выберет услугу. Причем цена услуги должна определяться исходя из даты создаваемого документа.

Найдите в конфигураторе документ **ОказаниеУслуги** и откройте его форму **ФормаДокумента**. Дважды щелкните на элементе формы **ПереченьНоменклатурыНоменклатура**, найдите событие **ПриИзменении**, которое возникает при изменении значения поля.

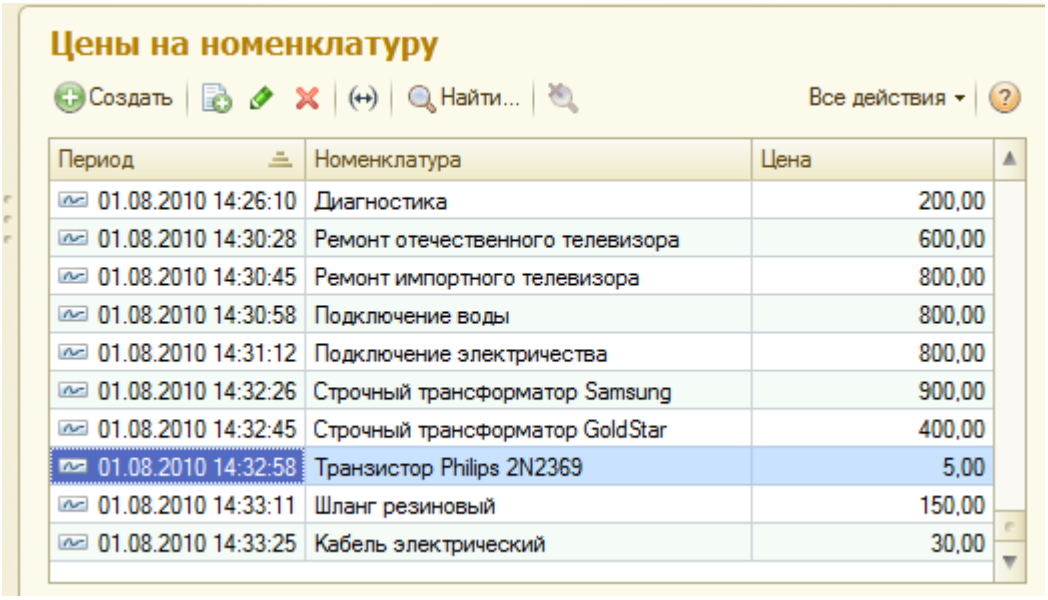
Нажмите на значок лупы. Система создаст шаблон процедуры обработчика этого события в модуле нашей формы и откроет закладку **Модуль** редактора формы. Внесите в него следующий текст:

```
// Получить текущую строку табличной части

СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;

// Установить цену

СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена(Объект.Дата, СтрокаТабличнойЧасти.Номенклатура);

// Пересчитать сумму строки

РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);
```

Должно получиться следующее:

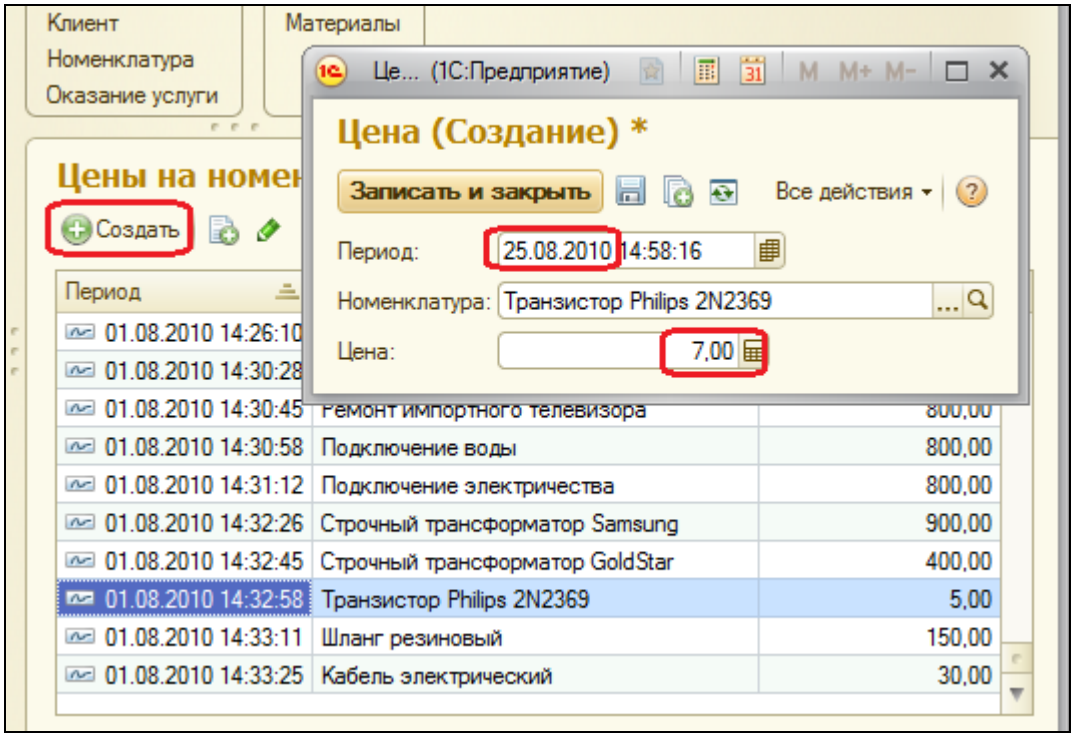
```
&НаКлиенте
Процедура ПеречеьНоменклатурыКоличествоПриИзменении(Элемент) ...

&НаКлиенте
Процедура ПеречеьНоменклатурыЦенаПриИзменении(Элемент) ...

&НаКлиенте
Процедура ПеречеьНоменклатурыНоменклатураПриИзменении(Элемент)
    // Получить текущую строку табличной части
СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;

// Установить цену
СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена(Объект.Дата, СтрокаТабличнойЧасти.Номенклатура);

// Пересчитать сумму строки
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

КонецПроцедуры
```

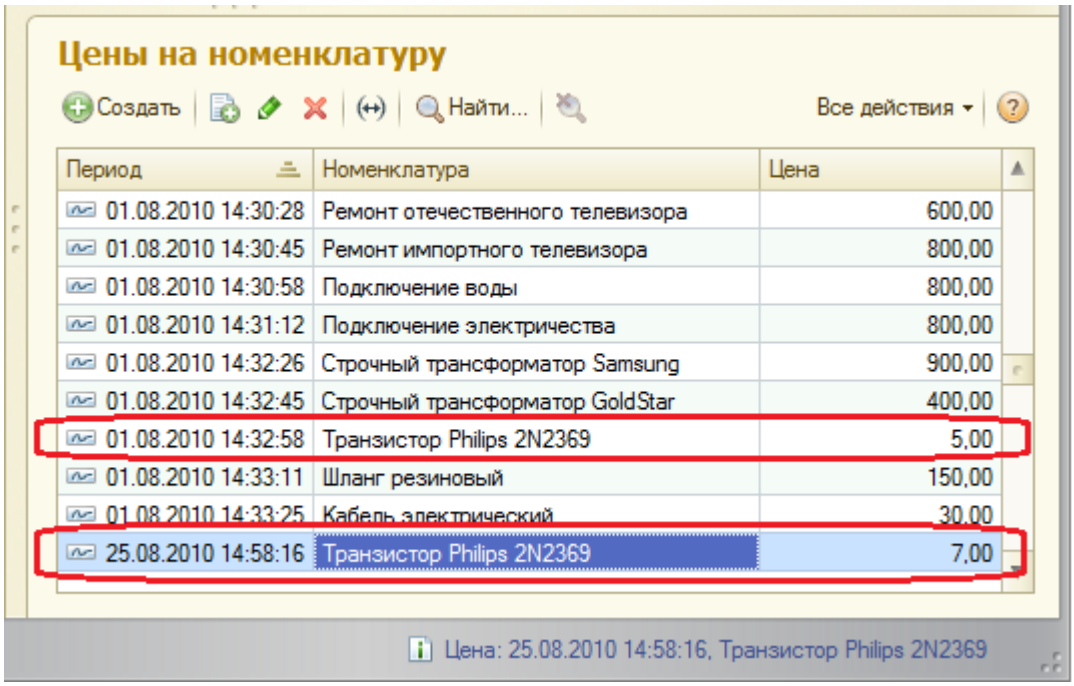
Теперь запустите 1С: Предприятие в режиме отладки и *откройте регистр сведений* **Цены на номенклатуру.**



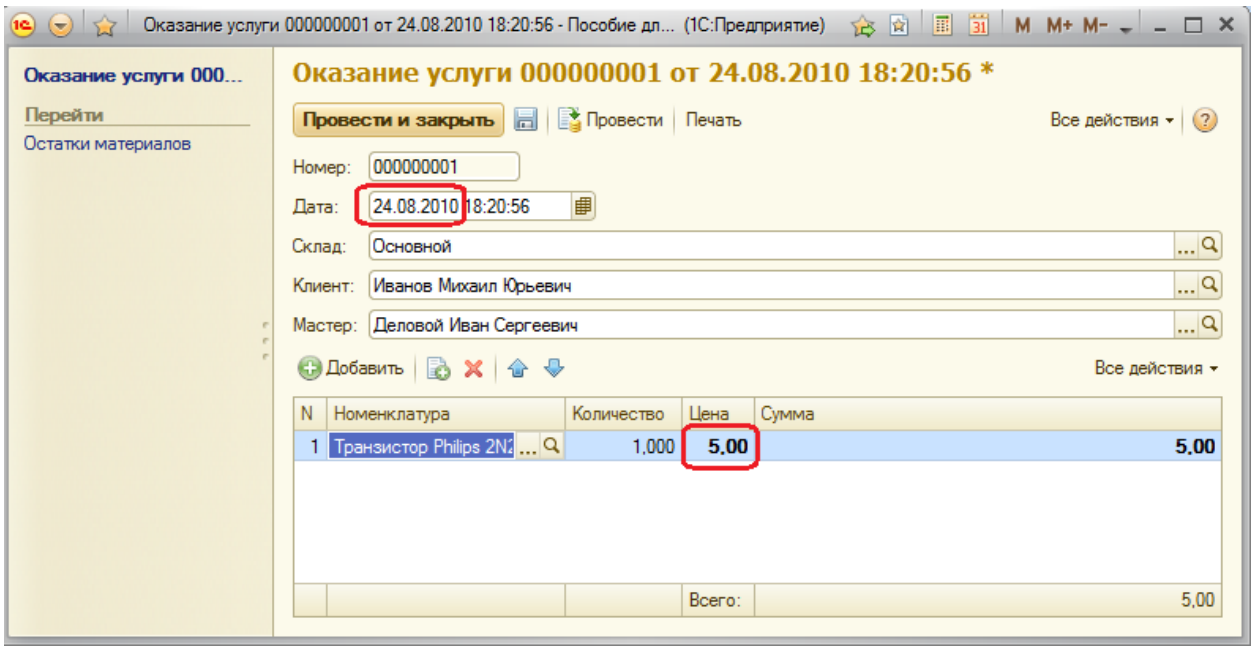
Период	Номенклатура	Цена
01.08.2010 14:26:10	Диагностика	200,00
01.08.2010 14:30:28	Ремонт отечественного телевизора	600,00
01.08.2010 14:30:45	Ремонт импортного телевизора	800,00
01.08.2010 14:30:58	Подключение воды	800,00
01.08.2010 14:31:12	Подключение электричества	800,00
01.08.2010 14:32:26	Строчный трансформатор Samsung	900,00
01.08.2010 14:32:45	Строчный трансформатор GoldStar	400,00
01.08.2010 14:32:58	Транзистор Philips 2N2369	5,00
01.08.2010 14:33:11	Шланг резиновый	150,00
01.08.2010 14:33:25	Кабель электрический	30,00

Для транзистора Philips добавим другим числом новую цену:

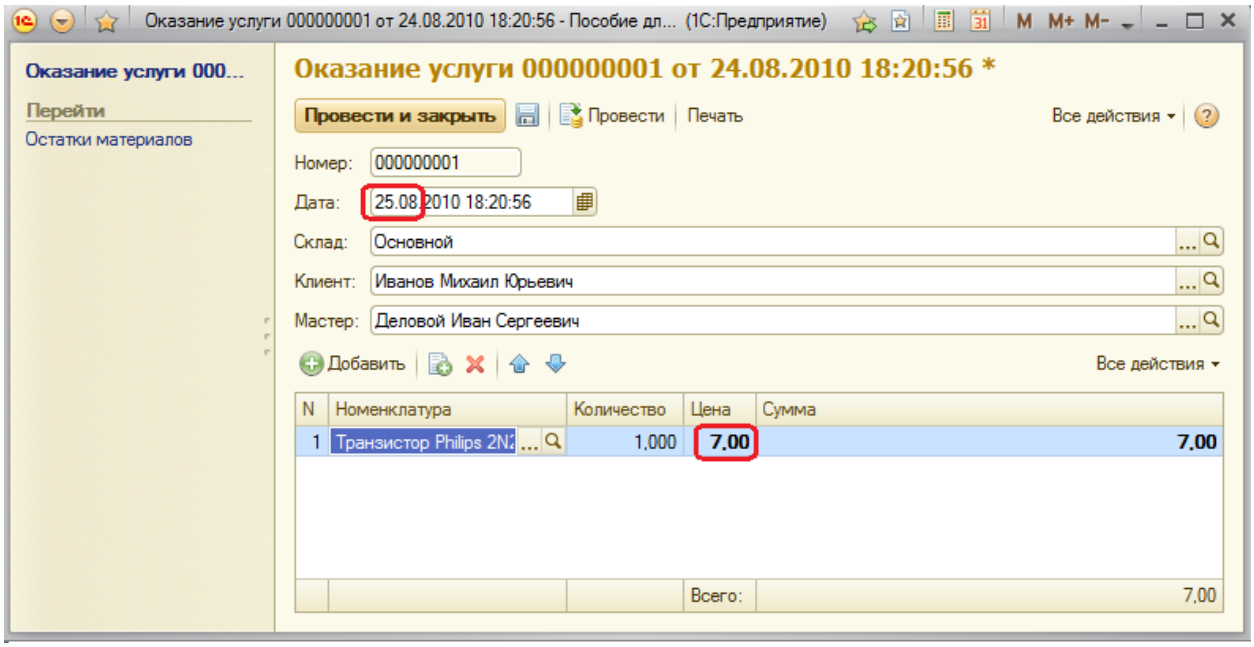




Теперь откройте документ **Оказание услуги №1**, где мы израсходовали один такой транзистор. Оставим дату документа без изменения и повторим выбор транзистора в колонке **Номенклатура** табличной части документа. Автоматически установится значение цены транзистора от 01.08.2010. Это последнее значение цены на дату документа.



Теперь изменим дату документа на текущую (совпадающую с датой установки новой цены) и выберем транзистор заново. Будет установлено новое значение цены, последнее на эту дату.



Таким образом, в документе появляется актуальная на момент создания документа цена услуги.

Контрольные вопросы

- ✓ Для чего предназначен объект Регистр сведений
- ✓ Какими особенностями обладает Регистр сведений
- ✓ В чем главнее отличия регистра сведений от регистра накопления

- ✓ Что такое периодический регистр сведений и что такое независимый регистр сведений
- ✓ Как создать периодический регистр сведений
- ✓ Что такое ведущее измерение регистра

Практическая работа № 9

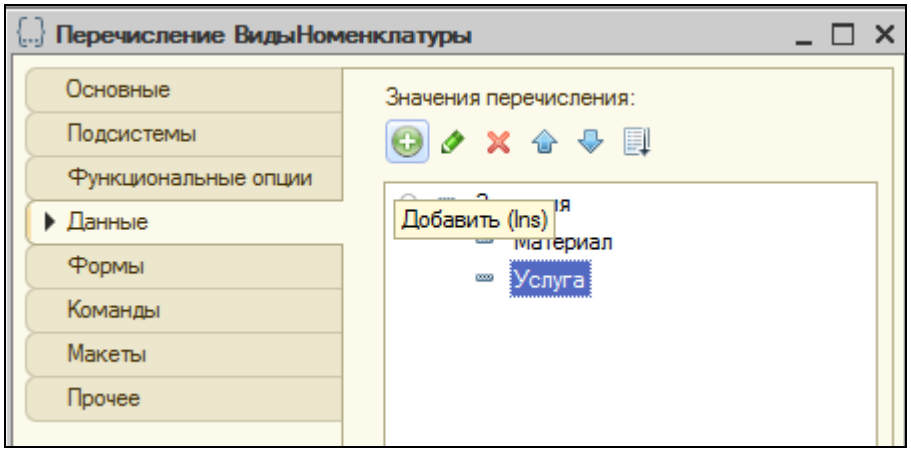
Перечисления (0:30)

В этой работе Вы создадите у справочника **Номенклатура** специальный реквизит, тип значения которого образуется объектом **Перечисление**. Это поможет Вам легко определять, чем является элемент справочника **Номенклатура**: услугой или материалом. Кроме этого, Вы скорректируете процедуру проведения документа **Оказание услуги** и поработаете с *перечислением* средствами встроенного языка.

Объект **Перечисление** предназначен для описания структуры хранения постоянных наборов значений не изменяемых в процессе работы конфигурации. На основе объекта **Перечисление**, платформа создает в базе данных таблицу, в которой хранится набор некоторых постоянных значений.

В реальной жизни этому объекту может соответствовать, например, перечисление вариантов цены – «включая НДС», «без НДС». Набор всех возможных значений, которые содержит *перечисление*, задается при конфигурировании системы, и пользователь не может изменять, удалять или добавлять новые.

Откройте конфигуратор и создайте новый объект *Перечисление* с именем **ВидыНоменклатуры**. На закладке **Данные** добавьте два значения перечисления: **Материал** и **Услуга**.



Привязка номенклатуры к значения перечисления ВидНоменклатуры

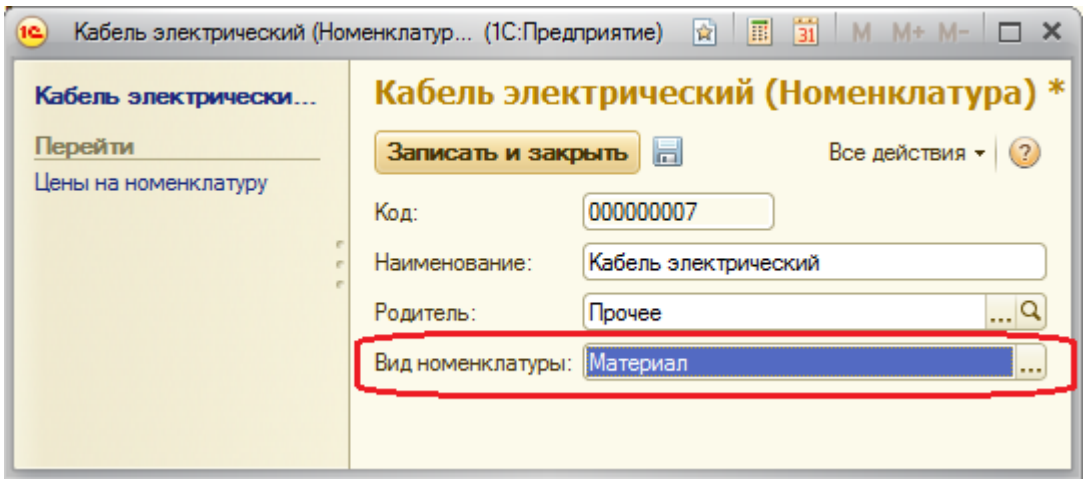
Для привязки номенклатуры к значениям перечисления, сделаем следующее:

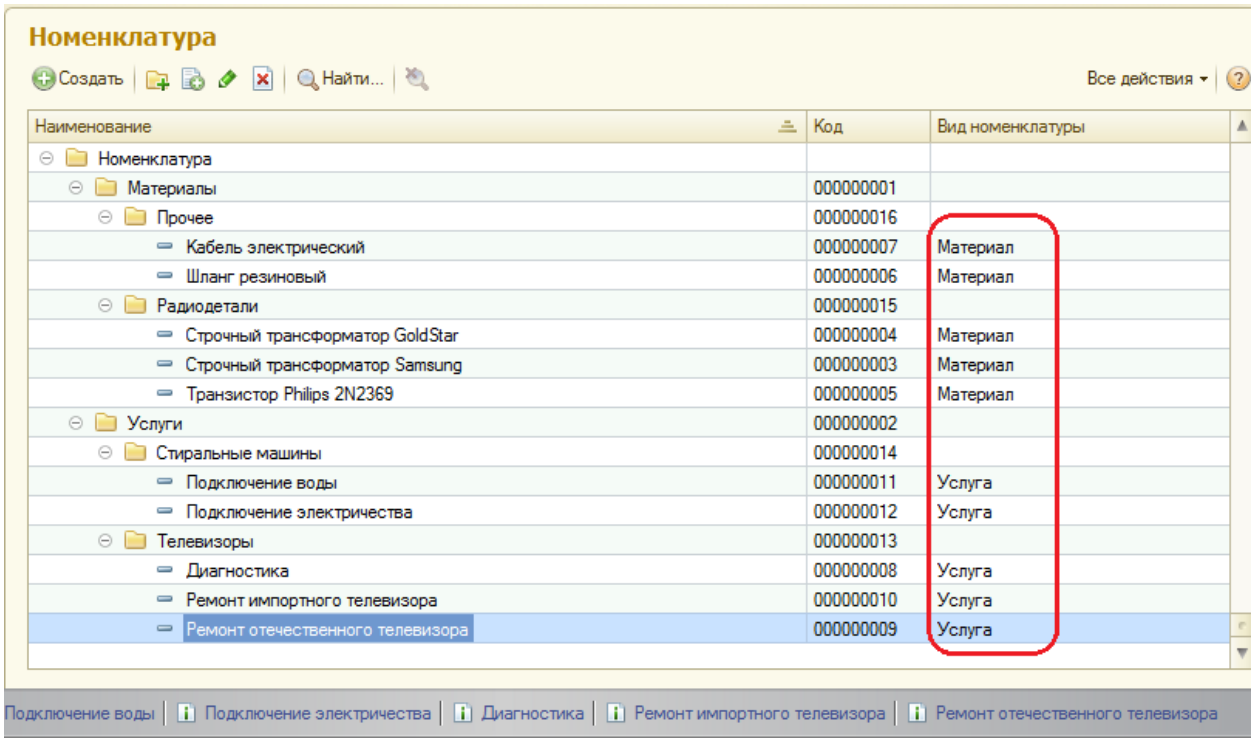
- В режиме конфигуратора создадим у справочника **Номенклатура** реквизит, который будет хранить значение перечисления.

- В режиме 1С: Предприятие проставим нужные значения этого реквизита для всех элементов справочника **Номенклатура**.

Добавьте в справочник **Номенклатура** новый реквизит **ВидНоменклатуры** с типом **ПеречислениеСсылка.ВидыНоменклатуры**.

Запустите 1С: Предприятие в режиме отладки. Зайдите в **Учет материалов – Номенклатура**. Задайте каждому элементу справочника соответствующее значение реквизита **Вид номенклатуры.**





Регистрация расхода только номенклатуры Материал

Вспомните, что в 5й работе, когда создавались движения документа ОказаниеУслуги по регистру накопления ОстаткиМатериалов, мы сказали, что

они не совсем правильные, поскольку в регистр будут попадать не только записи об израсходованных материалах, но и записи об оказанных услугах. Теперь мы доработаем документ таким образом, чтобы в регистре появлялись только записи, относящиеся к расходу материалов.

Для этого мы сначала в Конфигураторе изменим процедуру проведения документа, а потом в режиме 1С: Предприятия заново проведем все документы Оказание услуги, чтобы данные в регистре изменились в соответствии с новым алгоритмом проведения документа.

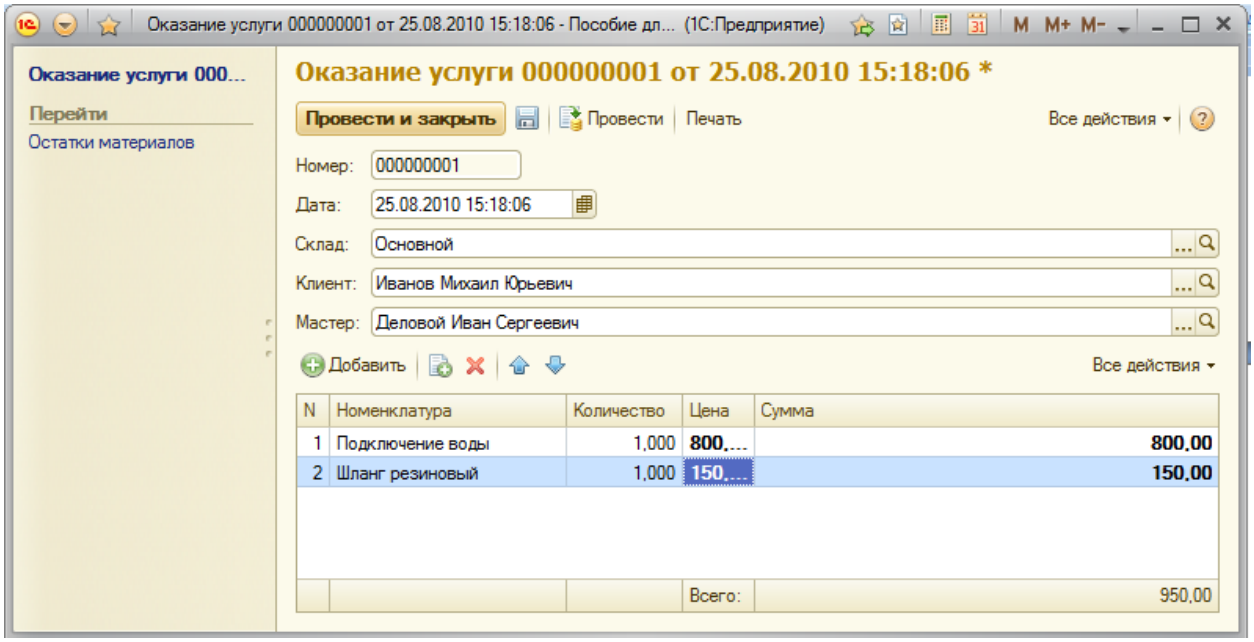
Откройте модуль документа **ОказаниеУслуги** (контекстное меню документа – **Открыть модуль объекта**) и добавьте в обработчик события **ОбработкаПроведения** это условие. **Жирным** выделены новые строки.

```
Процедура ОбработкаПроведения(Отказ, Режим)

	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

	// Данный фрагмент построен конструктором.

	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	Движения.ОстаткиМатериалов.Записывать = Истина;

	Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

		Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал Тогда

			// регистр ОстаткиМатериалов Расход

			Движение = Движения.ОстаткиМатериалов.Добавить();

			Движение.ВидДвижения = ВидДвиженияНакопления.Расход;

			Движение.Период = Дата;

			Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;

			Движение.Склад = Склад;

			Движение.Количество = ТекСтрокаПереченьНоменклатуры.Количество;

		КонецЕсли;

	КонецЦикла;

	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

КонецПроцедуры
```

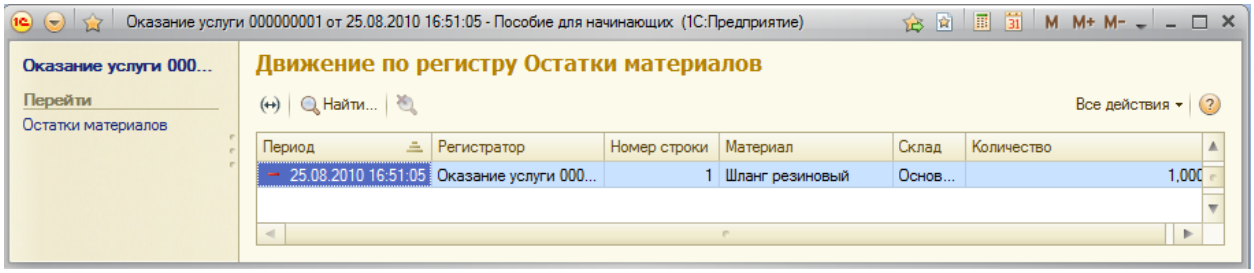
Добавленный текст исключает выполнение операторов цикла для тех строк табличной части документа, в которых номенклатура не является материалом.

Запустите 1С: Предприятие в режиме отладки. Откройте список документов **Оказание услуг**. Откройте документ **Оказание услуги №1** и внесите в него следующие изменения:

- Удалите из табличной части строку, содержащую **Транзистор philips**
- Добавьте услугу – **Подключение воды**
- Добавьте материал – **Шланг резиновый.**



Заметьте, что цены подставляются автоматически из *регистра сведений* **Цены**. Проведите документ кнопкой **Провести**. Перейдите в **Остатки материалов** через панель навигации.



Как видите, в движения по регистру Остатки материалов включаются только строки, содержащие материалы. Запись про услугу Подключение воды в движения не попала.

Контрольные вопросы

- ✓ Для чего предназначен объект Перечисление

- ✓ Как создать новое перечисление
- ✓ Как с помощью перечисления задать принадлежность элементов справочника к той или иной смысловой группе

Практическая работа № 10

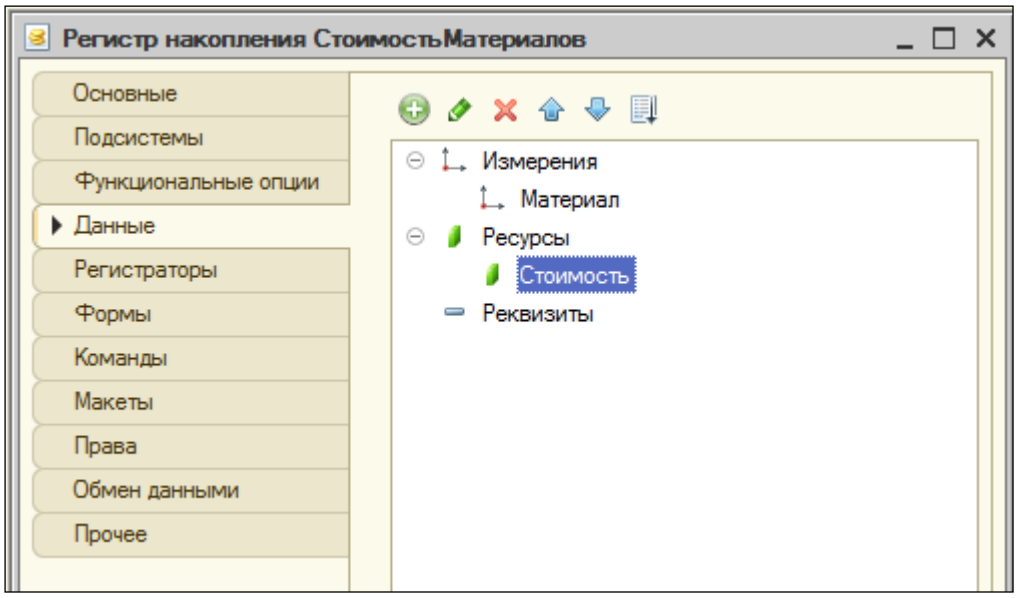
Проведение документа по нескольким регистрам (1:20)

В этой работе Вы создадите еще один *регистр накопления* и измените процедуру *проведения* документов так, чтобы они записывали необходимые данные в несколько регистров.

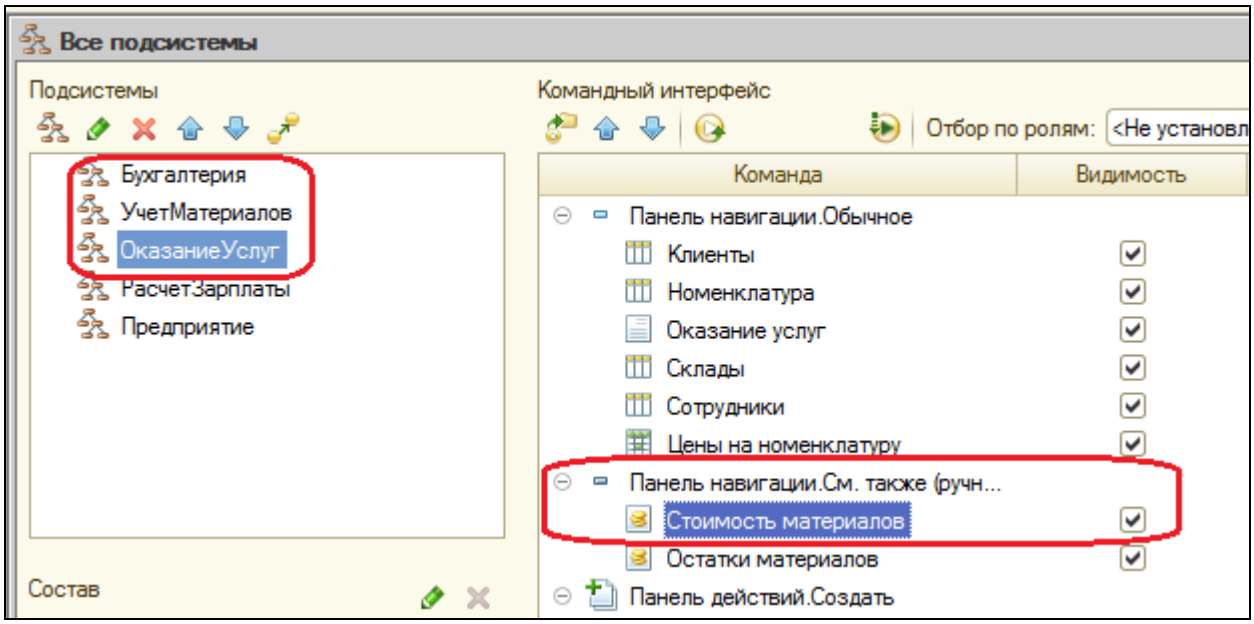
Необходимо знать, какие денежные средства были затрачены на приобретение материалов и каковы материальные запасы нашей фирмы в денежном выражении. Руководство выразило пожелание, чтобы весь суммовой учет материалов велся по средней стоимости. Т.е. при закупке материалов они должны учитываться в ценах приобретения, а при расходе – по средней стоимости, которая рассчитывается исходя из общей суммы закупок данного материала и общего количества этого материала, находящегося в нашей фирме.

Для этих целей будем использовать регистр накопления **СтоимостьМатериалов**. Т.о. документы **ПриходнаяНакладная** и **ОказаниеУслуги** должны будут создавать движения не только в регистре **ОстаткиМатериалов**, но одновременно в регистре **СтоимостьМатериалов**, отражая изменения суммового учета.

Создайте новый объект *Регистр накопления* с именем **СтоимостьМатериалов**. **Расширенное представление списка** задайте как **Движения по регистру Стоимость материалов**. На закладке **Подсистемы** отметьте **Бухгалтерия**, **УчетМатериалов** и **ОказаниеУслуг**. На закладке **Данные** создайте одно измерение – **Материал** с типом **СправочникСсылка.Номенклатура** и один ресурс – **Стоимость** с типом **Число**, длина 15, точность 2. Получится следующее:



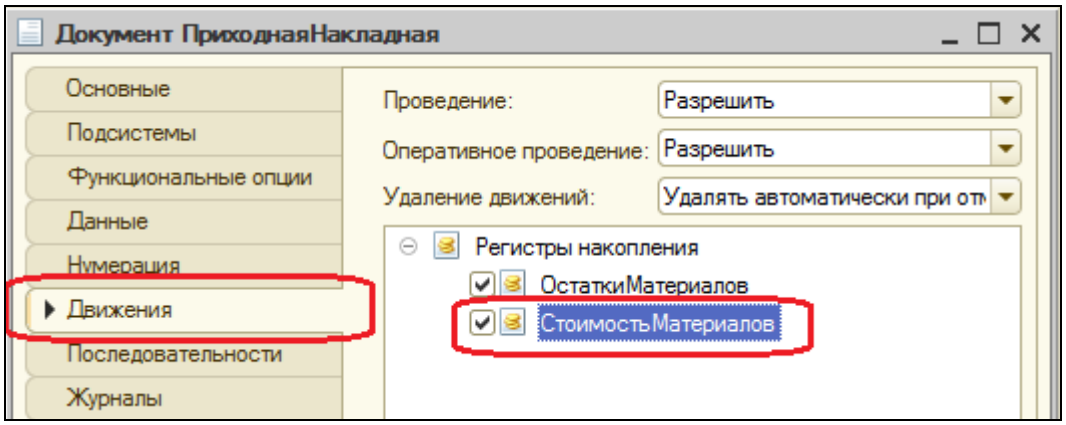
Теперь отредактируйте командный интерфейс, чтобы в подсистемах **Бухгалтерия**, **ОказаниеУслуг** и **УчетМатериалов** была доступна ссылка для просмотра нашего регистра накопления в разделе **Панель навигации.См.также**.



Проведение приходной накладной по двум регистрам

Откройте в конфигураторе окно редактирования объекта *Документ* **Приходная накладная** и перейдите на вкладку **Движения**.

В списке регистров отметьте, что документ будет создавать движения и по регистру **СтоимостьМатериалов**.



На этот раз мы не будем использовать конструктор движений, а внесем изменения прямо в обработчик события **ОбработкаПроведения** документа **ПриходнаяНакладная**.

Дело в том, что с помощью конструктора можно создавать движения одновременно и в нескольких регистрах, но тогда процедура

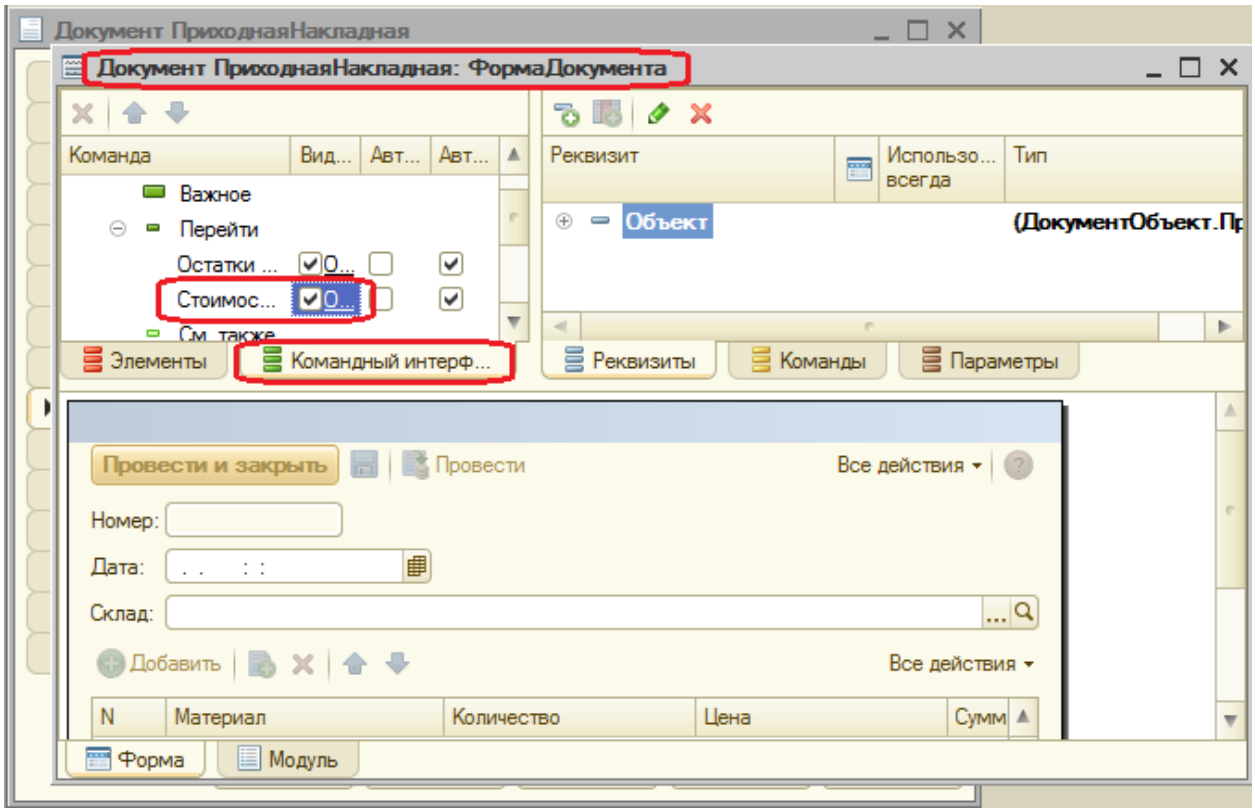
проведения, которую мы написали ранее, затрётся новой процедурой, созданной конструктором.

Перейдите на вкладку **Прочее** и откройте модуль объекта. В самом конце цикла перед строкой **КонецЦикла** добавим строки кода, создающие движение в регистре **СтоимостьМатериалов**.

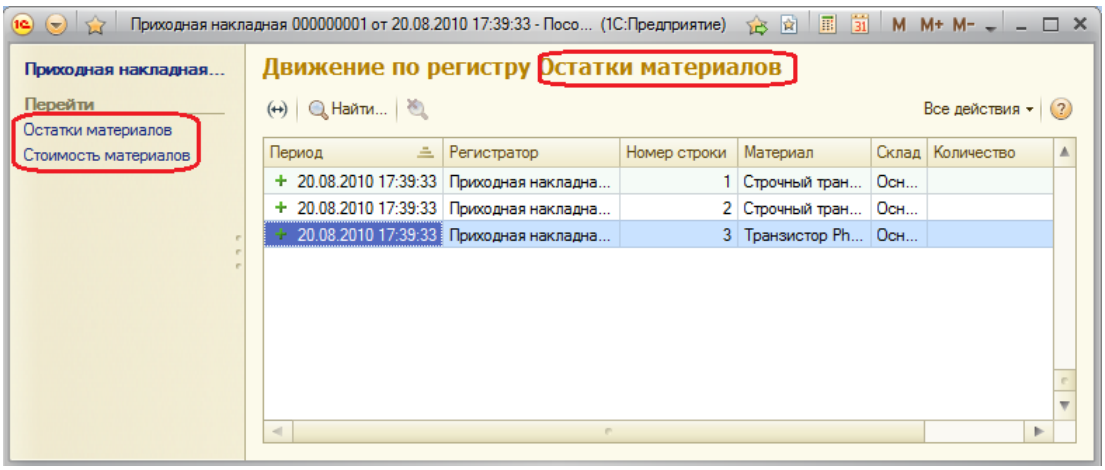
```
Процедура ОбработкаПроведения(Отказ, Режим)

	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

	// Данный фрагмент построен конструктором.

	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр ОстаткиМатериалов Приход

	Движения.ОстаткиМатериалов.Записывать = Истина;

	Движения.СтоимостьМатериалов.Записывать = Истина;

	Для Каждого ТекСтрокаМатериалы Из Материалы Цикл

		Движение = Движения.ОстаткиМатериалов.Добавить();

		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;

		Движение.Период = Дата;

		Движение.Материал = ТекСтрокаМатериалы.Материал;

		Движение.Склад = Склад;

		Движение.Количество = ТекСтрокаМатериалы.Количество;

		// регистр Стоимость Материалов Приход

		Движение = Движения.СтоимостьМатериалов.Добавить();

		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;

		Движение.Период = Дата;

		Движение.Материал = ТекСтрокаМатериалы.Материал;

		Движение.Стоимость = ТекСтрокаМатериалы.Сумма;

	КонецЦикла;

	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

КонецПроцедуры
```

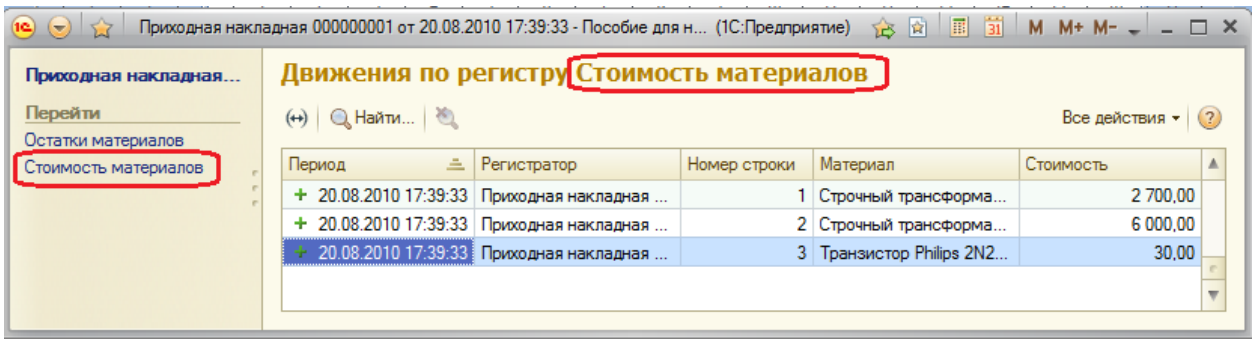
Отредактируйте командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра **СтоимостьМатериалов**, связанному с документом.

Для этого откройте форму документа ПриходнаяНакладная и перейдите на вкладку **Командный интерфейс**. В разделе **Панель навигации** раскройте группу **Перейти** и установите видимость для команды **Стоимость материалов...**



Запустите 1С: Предприятие и перепроведите все документы **Приходная накладная**. Затем откройте первый документ и, выполнив из него переход к регистрам **Остатки материалов** и **Стоимость материалов**, убедитесь, что документ создает желаемые записи в обоих регистрах.





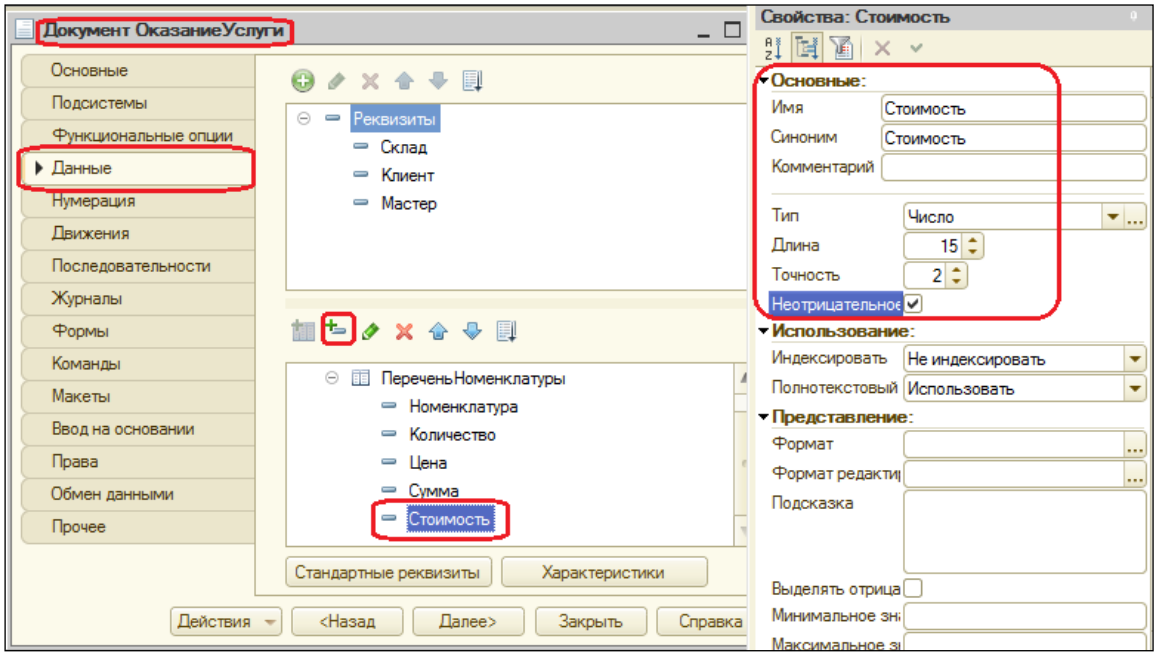
Проведение документа ОказаниеУслуги по двум регистрам

Внесем изменения в процедуру обработки проведения документа **ОказаниеУслуги**. Суть изменения – при списании материалов, израсходованных в процессе оказания услуги, должна быть возможность указывать различную стоимость для одного и того же материала, которая рассчитана руководством исходя из текущих соображений.

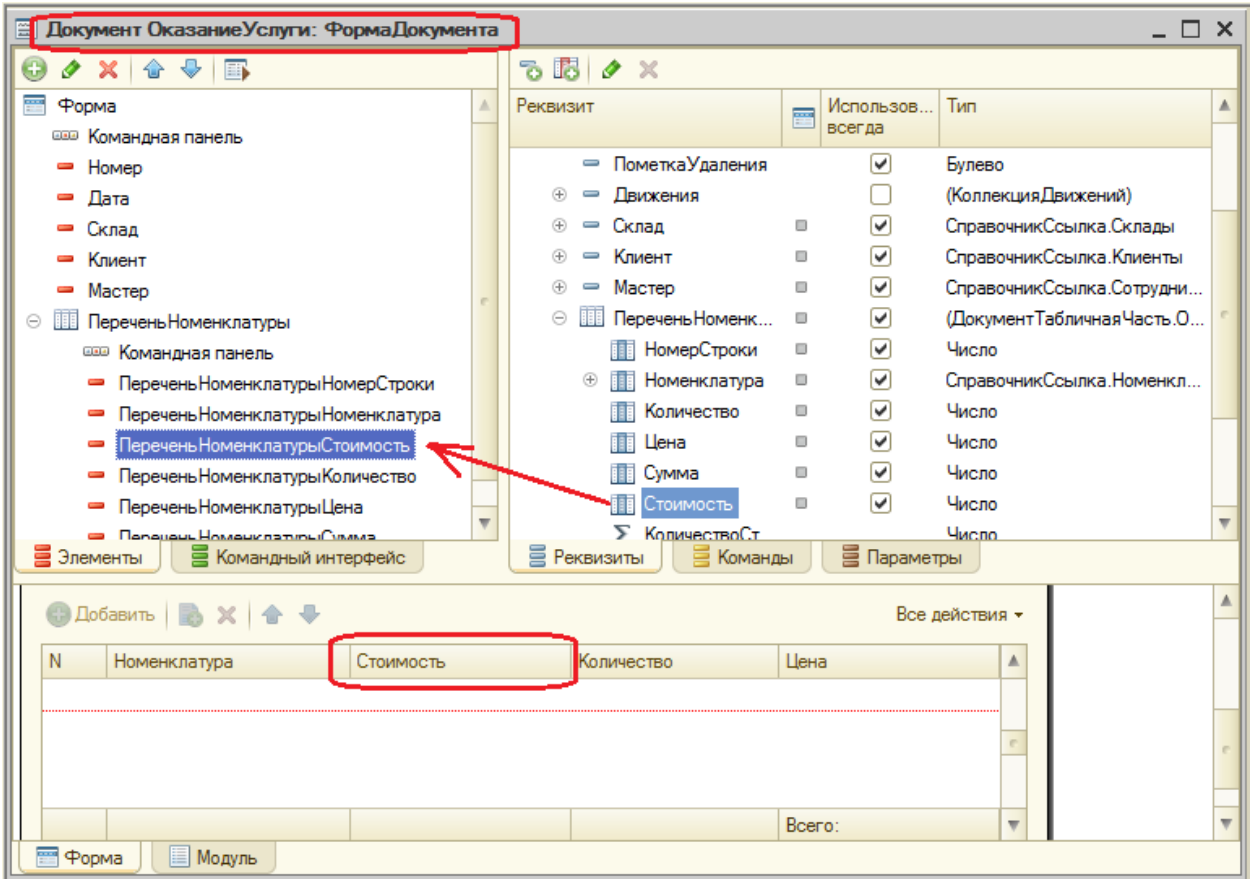
Поскольку в документе отражена только цена номенклатуры, нам понадобится:

1. Добавить в табличную часть документа еще один реквизит, в котором будет указываться стоимость номенклатуры.
2. После этого изменить процедуру проведения документа **ОказаниеУслуги**.
3. В режиме 1С: Предприятие перепровести все эти документы, чтобы отработал новый алгоритм.

Откройте окно редактирования объекта *Документ* **ОказаниеУслуги** и перейдите на вкладку **Данные**. Создайте новый реквизит табличной части документа с именем **Стоимость**, типом **Число**, длиной 15 и точностью 2, неотрицательное.



После этого откройте форму **ФормаДокумента** документа **ОказаниеУслуги** и добавьте в табличную часть **ПереченьНоменклатуры** поле, отображающее новый реквизит **Стоимость**. Для этого в правом верхнем окне редактора форм на закладке **Реквизиты** раскройте реквизит формы **Объект**. Найдите в табличной части реквизит **Стоимость** и мышью перетащите его в окно элементов формы (левый верхний угол) после поля **ПереченьНоменклатурыСтоимость**. Новый реквизит тут же отобразится внизу окна в форме документа.

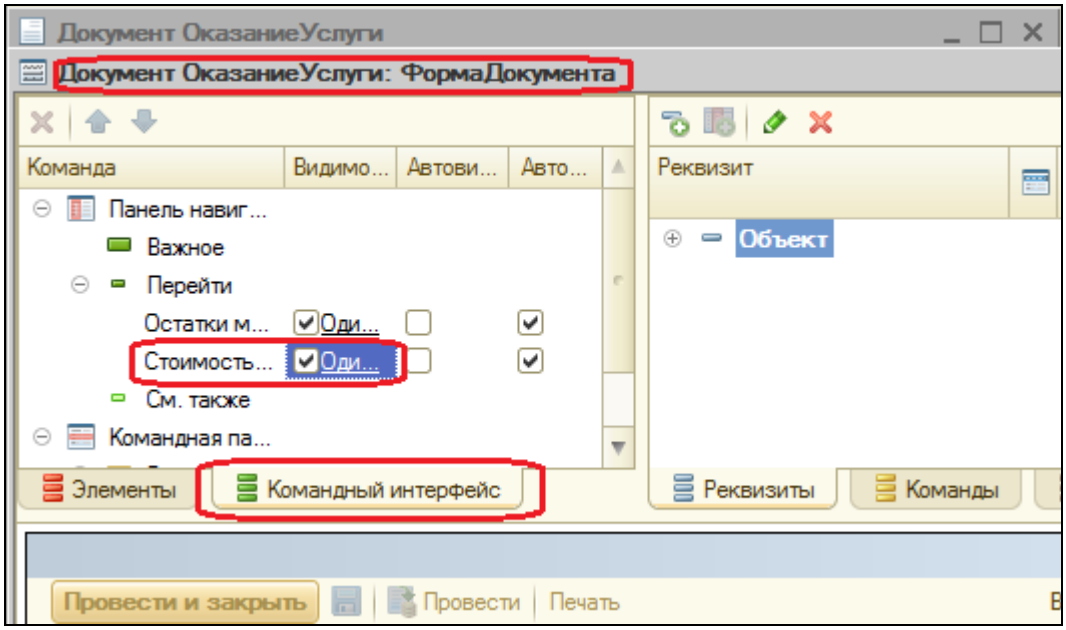


Теперь создадим *движения* документа **ОказаниеУслуги** таким же образом, как делали это для документа **ПриходнаяНакладная**.

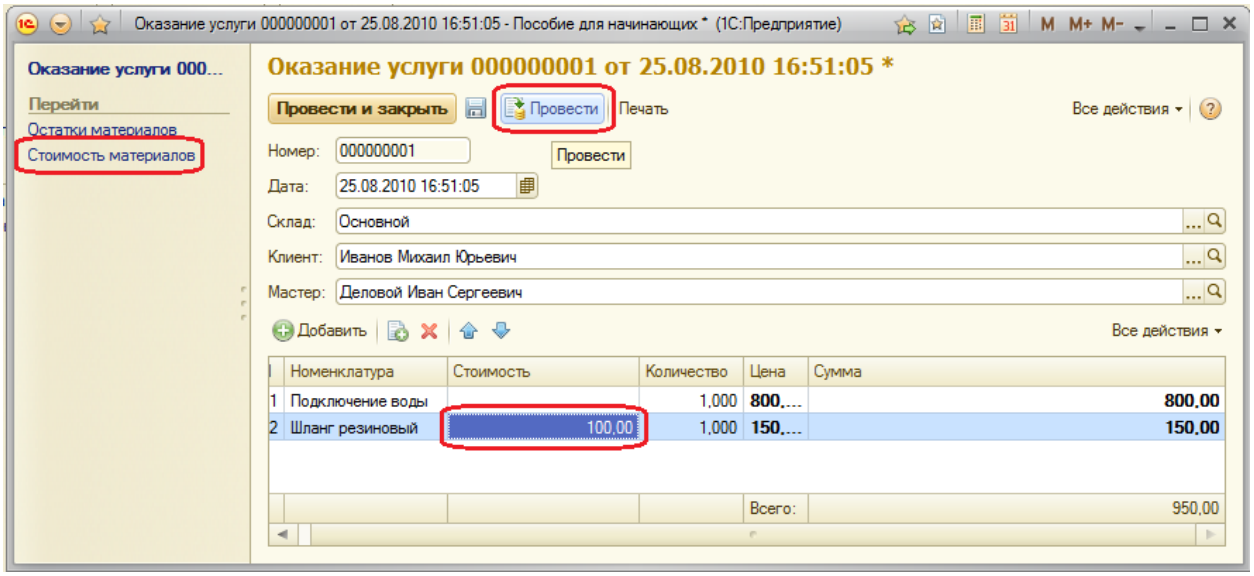
В окне редактирования *Документа* **ОказаниеУслуги** перейдите на вкладку **Движения.** В списке регистров отметьте **СтоимостьМатериалов**. Перейдите на закладку **Прочее** и откройте модуль объекта. Внесите изменения в процедуру обработки проведения, новые строчки выделены жирным:

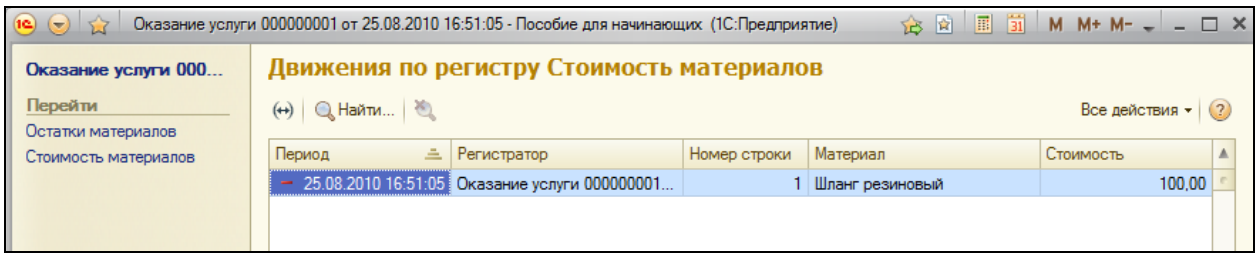
```
Процедура ОбработкаПроведения(Отказ, Режим)

	Движения.ОстаткиМатериалов.Записывать = Истина;

	Движения.СтоимостьМатериалов.Записывать = Истина;

	Для Каждого ТекСтрокаПеречеНоменклатуры Из ПеречеНоменклатуры Цикл

		Если ТекСтрокаПеречеНоменклатуры.Номенклатура.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал Тогда

			// регистр ОстаткиМатериалов Расход

			Движение = Движения.ОстаткиМатериалов.Добавить();

			Движение.ВидДвижения = ВидДвиженияНакопления.Расход;

			Движение.Период = Дата;

			Движение.Материал = ТекСтрокаПеречеНоменклатуры.Номенклатура;

			Движение.Склад = Склад;

			Движение.Количество = ТекСтрокаПеречеНоменклатуры.Количество;

			// регистр СтоимостьМатериалов Расход

			Движение = Движения.СтоимостьМатериалов.Добавить();

			Движение.ВидДвижения = ВидДвиженияНакопления.Расход;

			Движение.Период = Дата;

			Движение.Материал = ТекСтрокаПеречеНоменклатуры.Номенклатура;

			Движение.Стоимость = ТекСтрокаПеречеНоменклатуры.Количество*ТекСтрокаПеречеНоменклатуры.Стоимость;

		КонецЕсли;

	КонецЦикла;

КонецПроцедуры
```

Отредактируйте командный интерфейс формы документа, чтобы из панели навигации можно было попасть в регистр **Стоимость материалов**, связанному с документом.

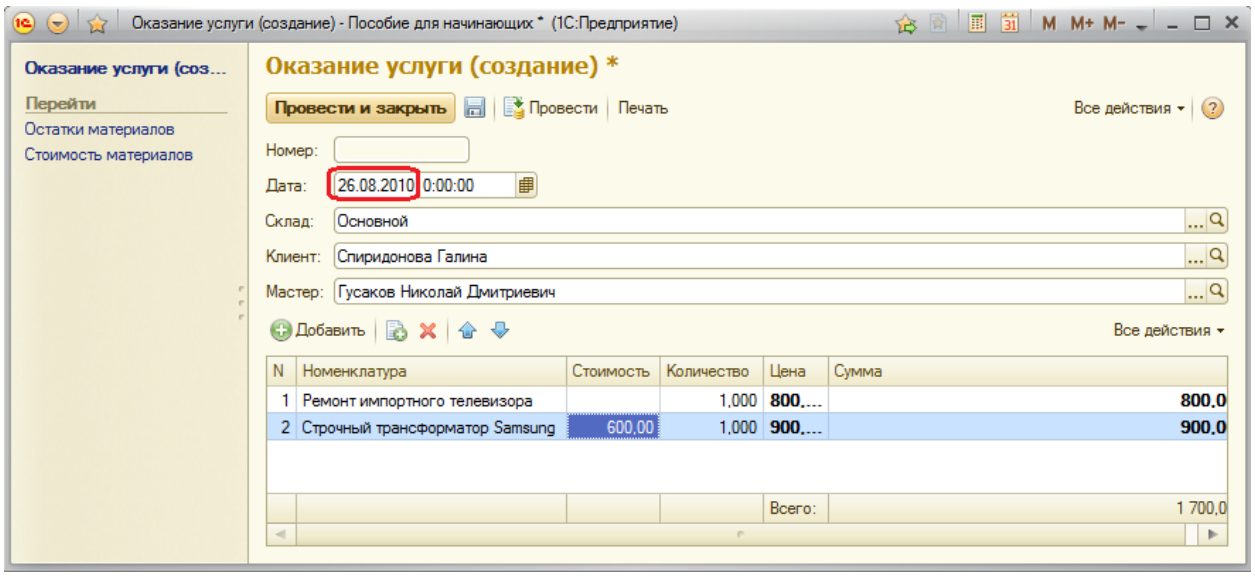


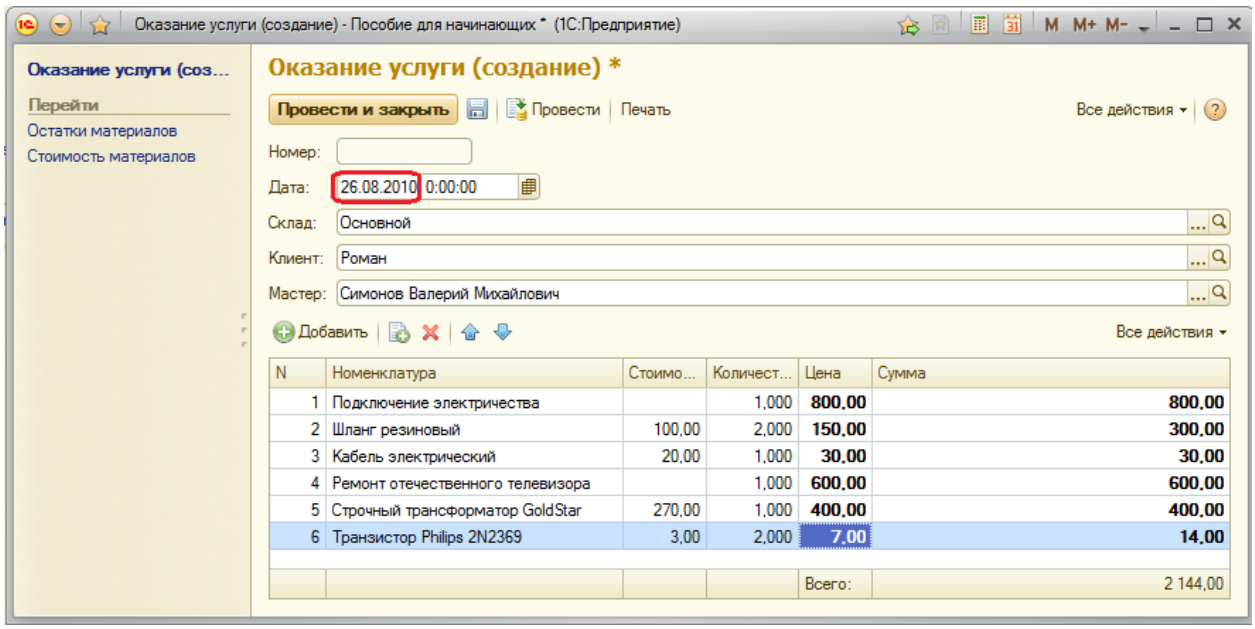
Запустите 1С: Предприятие, откройте документ Оказание услуги № 1 и укажите в нем стоимость резинового шланга – 100. Нажмите **Провести** и перейдите в регистр **Стоимость материалов** через панель навигации.



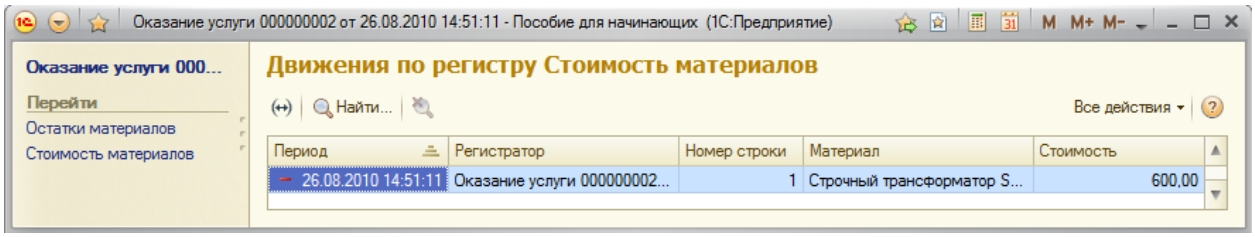


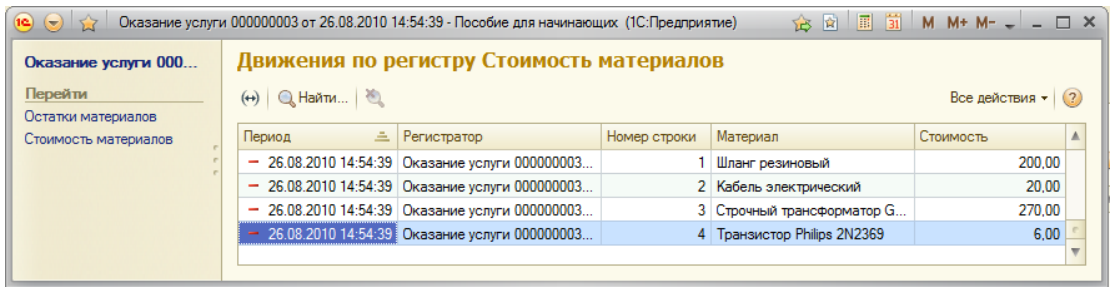
Теперь создайте и проведите еще два документа **ОказаниеУслуги**.





Движения созданных документов по регистру Стоимость материалов:





Контрольные вопросы

- ✓ Для чего может понадобиться проведение документа по нескольким регистрам
- ✓ Как создать движения документа по нескольким регистрам
- ✓ Как создать движения документа без использования конструктора движений
- ✓ Как добавить в форму документа новый реквизит

Практическая работа № 11

Оборотные регистры накопления (0:40)

В этой работе Вы познакомитесь с видом *регистра накопления* – *оборотным регистром накопления*. Узнаете о некоторых важных принципах выбора измерений и реквизитов регистров накопления. Создадите *оборотный регистр накопления* и добавите в один из документов движения еще и по этому регистру.

До сих пор мы создавали в регистрах накопления движения только для строк документа, которые содержат материалы. Услуги, содержащиеся в документе, мы никак не учитывали.

Бессмысленно говорить о том, сколько услуг было и сколько осталось, важна только сумма и количество услуг за промежуток времени. Кроме этого, интересны следующие моменты:

- Какие именно услуги оказаны (для рейтинга услуг)
- Какому именно клиенту оказывались услуги(чтобы, например, предоставить ему скидку от объема ранее оплаченных услуг)
- Какой мастер предоставлял услуги (чтобы начислить ему зарплату)

Очевидно, что существующие регистры накопления не подходят для решения этих задач. Поэтому создадим еще одной хранилище данных – *оборотный регистр* **Продажи**.

Регистры накопления могут быть *регистрами остатков* и *регистрами оборотов*. Существующие в нашей конфигурации регистры накопления **ОстаткиМатериалов** и **СтоимостьМатериалов** являются *регистрами остатков*. Если Вы помните, при создании отчета Материалы в конструкторе запроса мы видели, что для таких регистров система создает три виртуальные таблицы: таблица остатков, оборотов и совокупная таблица остатков и оборотов.

Оборотный регистр очень похож на регистр остатков, только накапливает он обороты, остатки ему безразличны. Поэтому единственная виртуальная таблица будет таблицей оборотов.

По каждому из измерений *регистра накопления* остатков ресурсы обязательно должны изменяться в обе стороны: приход и расход. Не должно существовать таких измерений, по которым осуществляется

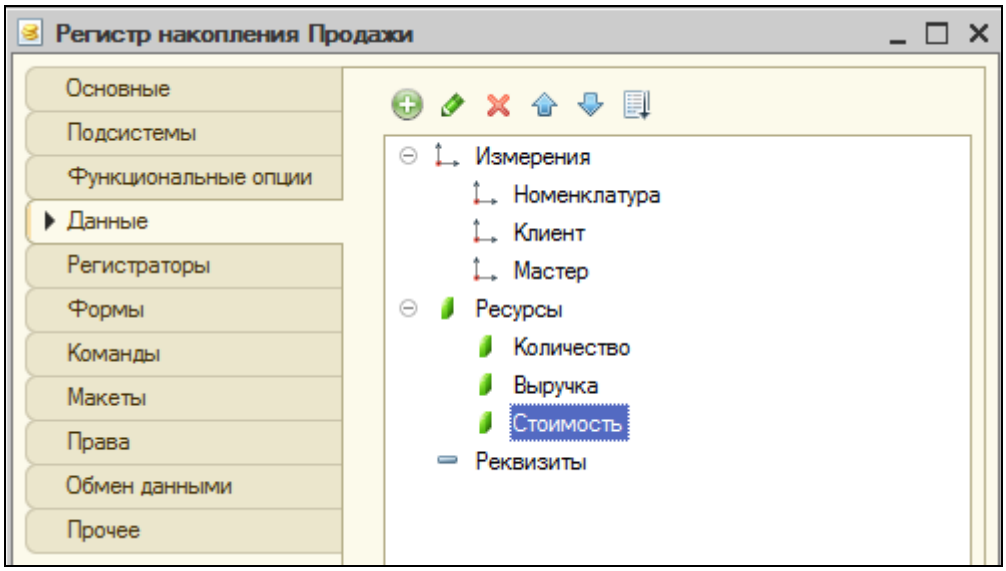
только приход или только расход. Нарушение этого принципа приведет к замедлению работы системы. Для реквизитов же регистра накопления этот принцип не важен. По реквизитам регистра ресурсы могут только приходоваться или только расходоваться.

Создайте новый *регистр накопления* с именем **Продажи**, вид – **Обороты**. **Расширенное представление списка** – **Движение по регистру Продажи**. На вкладке **Подсистемы** отметьте **Бухгалтерия, УчетМатериалов, ОказаниеУслуг**. На вкладке **Данные** создайте *измерения* регистра:

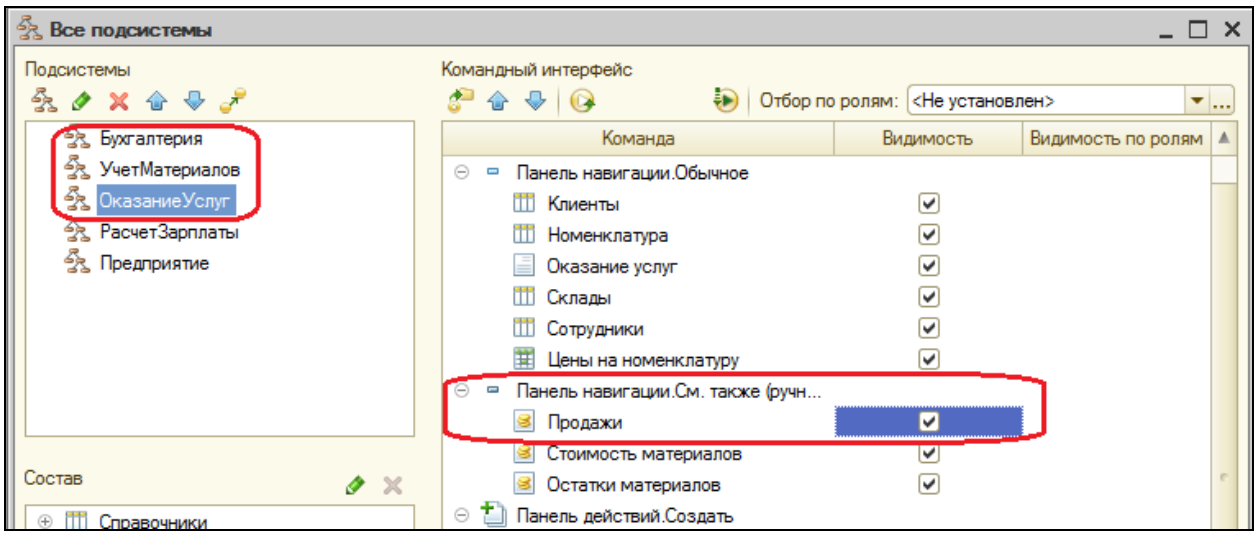
- **Номенклатура**, тип **СправочникСсылка.Номенклатура**;
- **Клиент**, тип **СправочникСсылка.Клиенты**;
- **Мастер**, тип **СправочникСсылка.Сотрудники**;

Создайте три *ресурса*:

- **Количество**, тип **Число**, длина 15, точность 2
- **Выручка**, тип **Число**, длина 15, точность 2;
- **Стоимость**, тип **Число**, длина 15, точность 2.



Отредактируйте *командный интерфейс*, чтобы в подсистемах **Бухгалтерия, ОказаниеУслуг** и **УчетМатериалов** была доступна ссылка для просмотра оборотного регистра накопления **Продажи**. (**Все Подсистемы** – **Продажи** включить и перетащить в **Панель навигации.См.также**).



Проведение документа Оказание услуги по трем регистрам

Сначала изменим процедуру проведения документа **ОказаниеУслуги**, а затем в режиме 1С: Предприятия перепроведем все эти документы, чтобы отработал новый алгоритм проведения.

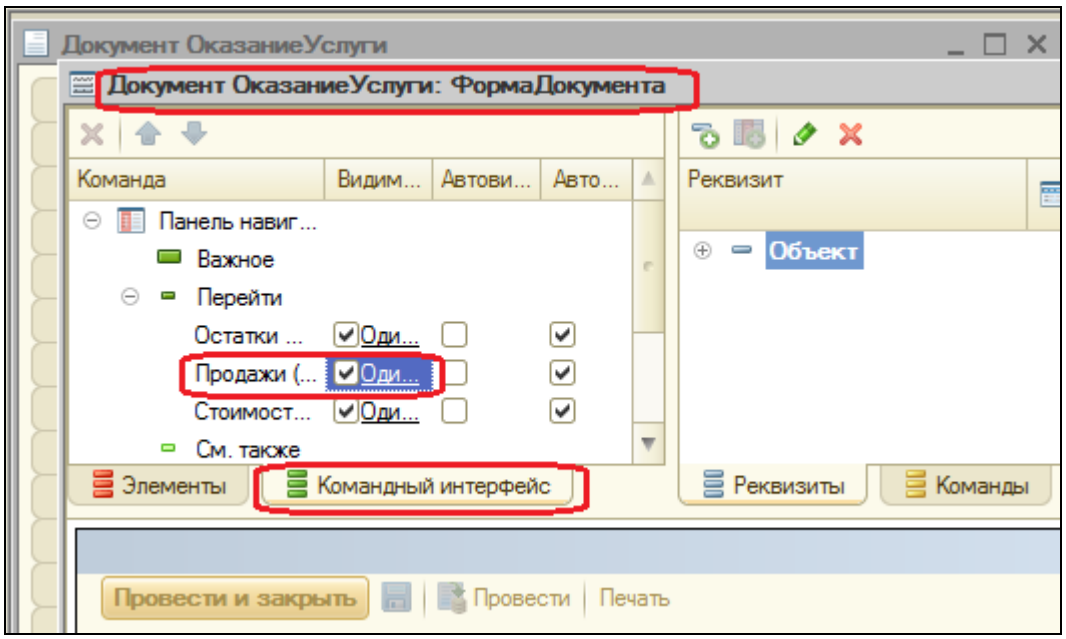
Откройте окно редактирования объекта *Документ* **ОказаниеУслуги** и на вкладке **Движения** укажите регистр **Продажи**. Перейдите на вкладку Прочее и откройте модуль объекта. Изменим процедуру обработки следующим образом:

```
Процедура ОбработкаПроведения(Отказ, Режим)

        Движения.ОстаткиМатериалов.Записывать = Истина;

        Движения.СтоимостьМатериалов.Записывать = Истина;

        Движения.Продажи.Записывать = Истина;

        Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл

                Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
Перечисления.ВидыНоменклатуры.Материал Тогда

                        // регистр ОстаткиМатериалов Расход

                        Движение = Движения.ОстаткиМатериалов.Добавить();

                        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;

                        Движение.Период = Дата;

                        Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;

                        Движение.Склад = Склад;
```

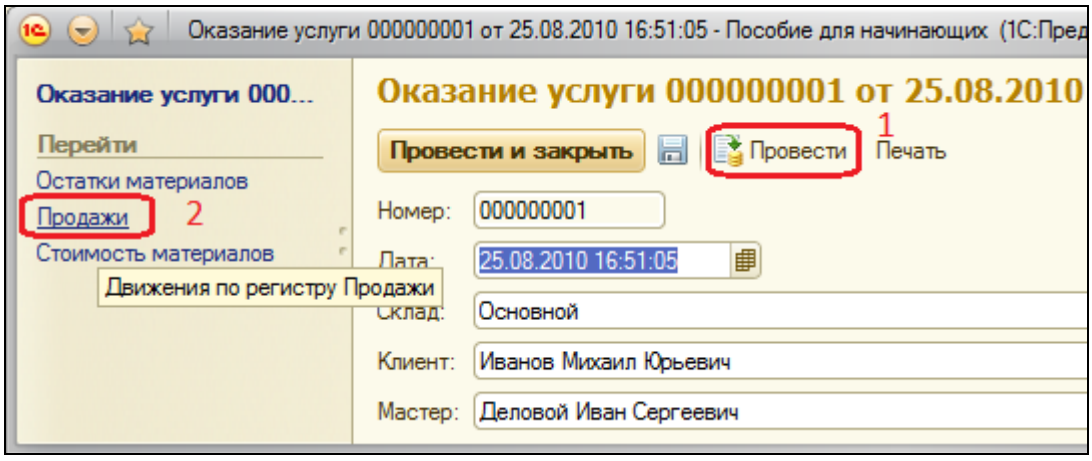
```
                        Движение.Количество = ТекСтрокаПеречеНьНоменклатуры.Количество;

                        // регистр СтоимостьМатериалов Расход

                        Движение = Движения.СтоимостьМатериалов.Добавить();

                        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;

                        Движение.Период = Дата;

                        Движение.Материал = ТекСтрокаПеречеНьНоменклатуры.Номенклатура;

                        Движение.Стоимость =
ТекСтрокаПеречеНьНоменклатуры.Количество*ТекСтрокаПеречеНьНоменклатуры.Стоимость;

                КонецЕсли;

                // Регистр Продажи

                Движение = Движения.Продажи.Добавить();

                Движение.Период = Дата;

                Движение.Номенклатура = ТекСтрокаПеречеНьНоменклатуры.Номенклатура;

                Движение.Клиент = Клиент;

                Движение.Мастер = Мастер;

                Движение.Количество = ТекСтрокаПеречеНьНоменклатуры.Количество;

                Движение.Выручка = ТекСтрокаПеречеНьНоменклатуры.Сумма;

                Движение.Стоимость = ТекСтрокаПеречеНьНоменклатуры.Стоимость *
ТекСтрокаПеречеНьНоменклатуры.Количество;

        КонецЦикла;

КонецПроцедуры
```

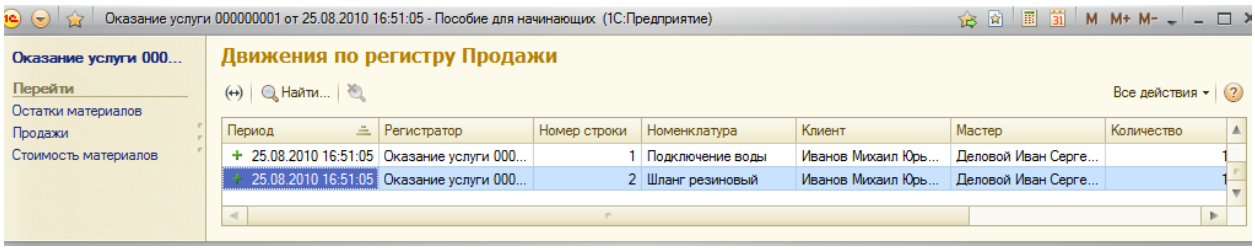
Обратите внимание, что у оборотного регистра отсутствует свойство **ВидДвижения**, т.к. отражение вида движения (приход или расход) имеет смысл только при учете остатков.

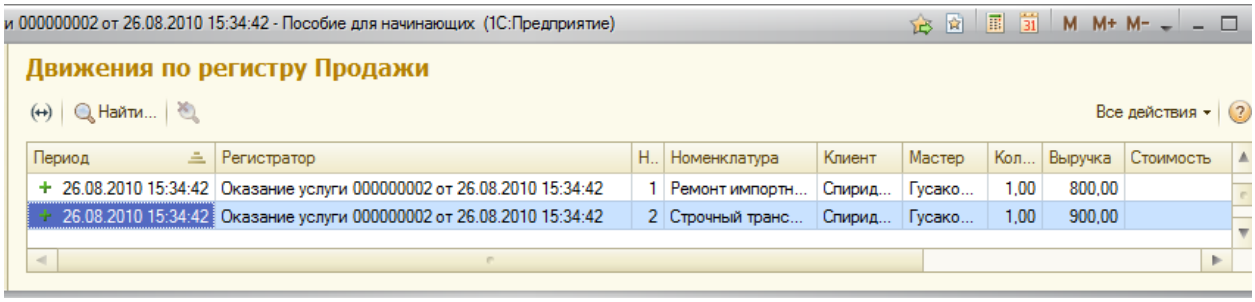
Отредактируйте командный интерфейс формы документа **ОказаниеУслуги**, включив видимость команды открытия регистра **Продажи**.

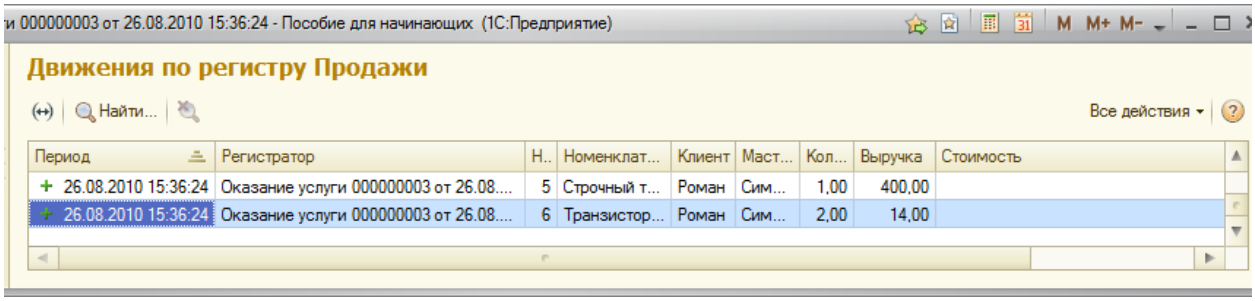


Запустите 1С: Предприятие, перепроведите каждый из документов **Оказание услуги**, после переходя из каждого в регистр **Продажи**.









Контрольные вопросы

- ✓ Что такое оборотный регистр накопления
- ✓ В чем отличие между регистром накопления остатков и оборотным регистром накопления
- ✓ Как выбирать реквизиты и измерения при создании регистров накопления
- ✓ Как создать оборотный регистр накопления

Практическая работа № 12

Отчёты (4:30)

В этой работе мы познакомимся с *системой компоновки данных*. Для этого создадим шесть отчетов.

Чаще всего данные, необходимые для отчета, находятся в базе данных. Для указания системе компоновки данных, какая информация и откуда должна быть получена, используется язык запросов.

В начале этой работы мы познакомимся с общими сведениями о языке запросов системы 1С: Предприятие и о системе компоновки данных.

Теория

Способы доступа к данным

Система поддерживает два способа доступа к данным, хранящимся в БД:

- **Объектный (для чтения и записи)**,
- **Табличный (для чтения)**.

Объектный способ реализован через использование объектов встроенного языка.

Важная особенность этого способа – при обращении к объекту встроенного языка, мы обращаемся к совокупности данных в БД как к единому целому.

Например, объект **ДокументОбъект.ОказаниеУслуги** будет содержать значения всех реквизитов документа **ОказаниеУслуги** и всех его табличных частей.

Объектный способ обеспечивает сохранение целостности объектов, их кэширование, вызов соответствующих обработчиков событий и т.д.

Табличный способ доступа к данным реализован с помощью запросов к БД на *языке запросов*.

В данном способе разработчик получает возможность оперировать отдельными полями таблиц.

Этот способ предназначен для отбора, группировки, сортировки, объединения выборок, расчета итогов и т.д. Т.е. оптимизирован для

обработки больших объемов информации в БД и получения данных по заданным критериям.

Работа с запросами

Для работы с запросами используется объект встроенного языка **Запрос.** Он позволяет получать информацию, хранящуюся в полях БД в виде выборки.

Исходную информацию запрос получает из набора таблиц. Все таблицы, которыми оперирует язык запросов, можно разделить на группы.

Реальные таблицы содержат данные одной реальной таблицы в БД. *Виртуальные таблицы* формируются в основном из данных нескольких таблиц БД. Общим для виртуальных таблиц является то, что им можно задать ряд параметров, определяющих, какие данные будут в них включены.

Объектные (ссылочные) таблицы представляют информацию ссылочных типов данных (справочники, документы, планы видов характеристик и т.д.). А *необъектные (нессылочные)* – все остальные типы данных (константы, регистры и т.д.).

Отличительной особенностью *объектных таблиц* является то, что они включают в себя поле Ссылка, содержащее ссылку на текущую запись. Для таких таблиц возможно получение пользовательского представления объекта.

Язык запросов

Текст запроса может состоять из нескольких частей:

- Описание запроса,
- Объединение результатов,
- Упорядочивание результатов
- Автоупорядочивание,
- Описание итогов.

Обязательная часть запроса – первая. Остальные присутствуют по необходимости.

Описание запроса определяет источники данных, поля выборки, группировки и т.д.

Мы не будем писать запросы «руками». Для большинства отчетов с помощью *системы компоновки данных*, запрос можно создать при помощи конструктора запросов.

Поэтому наша задача в этой работе – научиться читать и понимать тексты этих запросов, чтобы в дальнейшем иметь возможность изменять их.

Система компоновки данных

Она предназначена для создания произвольных отчетов в системе 1С: Предприятие и состоит из нескольких основных частей.

Исходные данные для компоновки отчета содержит в себе *схема компоновки данных*. Это наборы данных и методы работы с ними.

Отчет системы компоновки имеет сложную иерархическую структуру и может состоять из различных элементов (группировки, таблицы, диаграммы).

Пользователь может изменить существующую структуру отчета или создать новую, может настроить необходимый ему отбор, оформление элементов структуры, получить расшифровку по каждому элементу и т.д.

Последовательность работы системы компоновки можно представить так:

Разработчик создает *схему компоновки данных* и *настройки по умолчанию*. На основе этого *компоновщик макета* создает *макет*. *Процессор компоновки* выбирает данные из информационной базы согласно макету, собирает и оформляет эти данные. *Результат компоновки* обрабатывается *процессором вывода* и пользователь получает результирующий табличный документ.

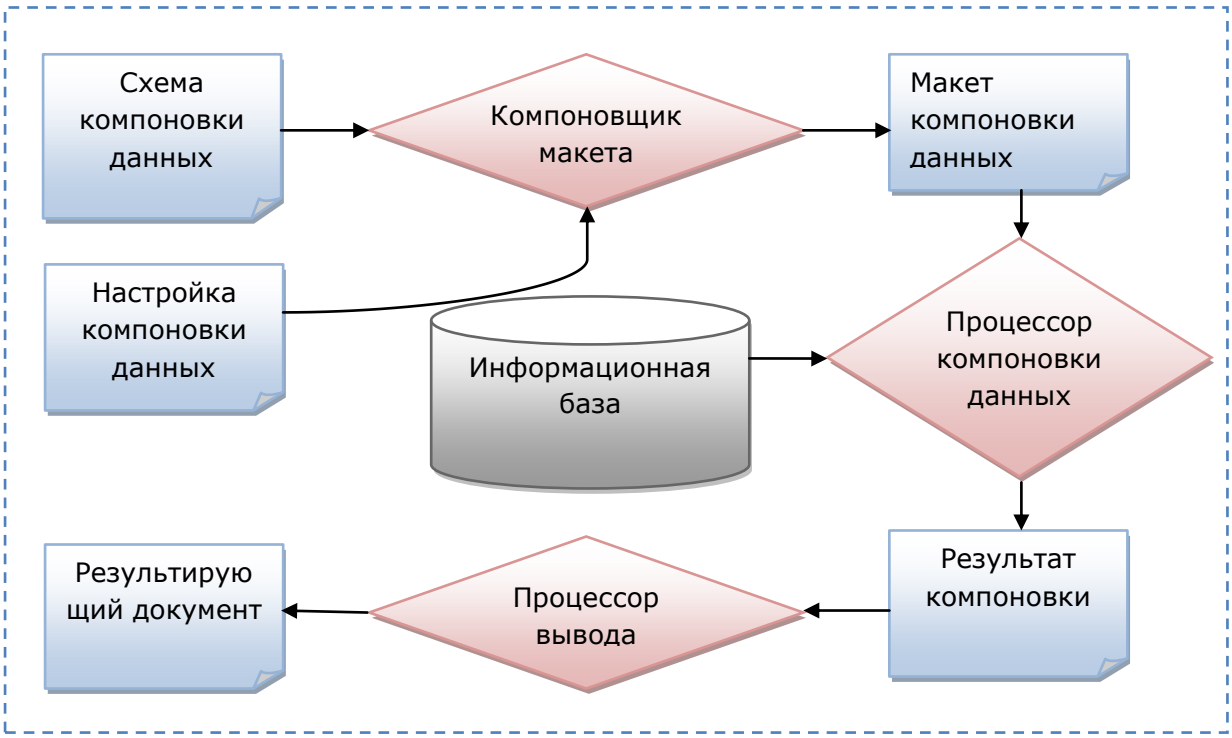


Рис. Схема работы системы компоновки данных.

Выбор данных из одной таблицы

Создадим отчёт **Реестр документов оказание услуги**, используя систему компоновки данных.

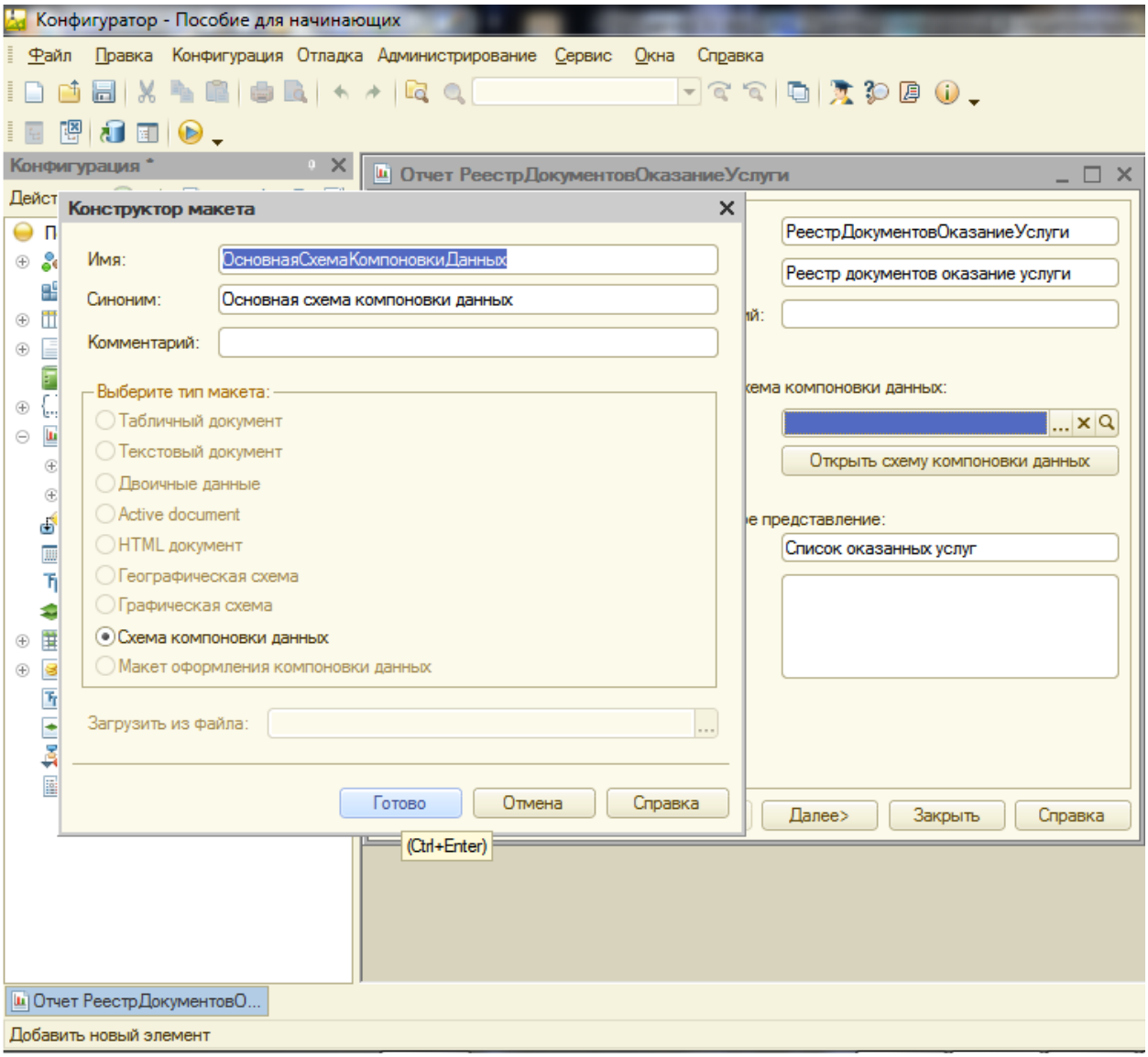
На примере этого отчета мы покажем, как выбрать данные из одной таблицы и как вывести их в определенном порядке, а также познакомимся, как использовать расшифровку в готовом отчете. Этот отчёт будет выводить список существующих в базе данных документов **ОказаниеУслуги** в порядке их дат и номеров.

Добавим в Конфигураторе объект конфигурации Отчёт. Повторим первые шаги по созданию отчета, описанные в работе №6.

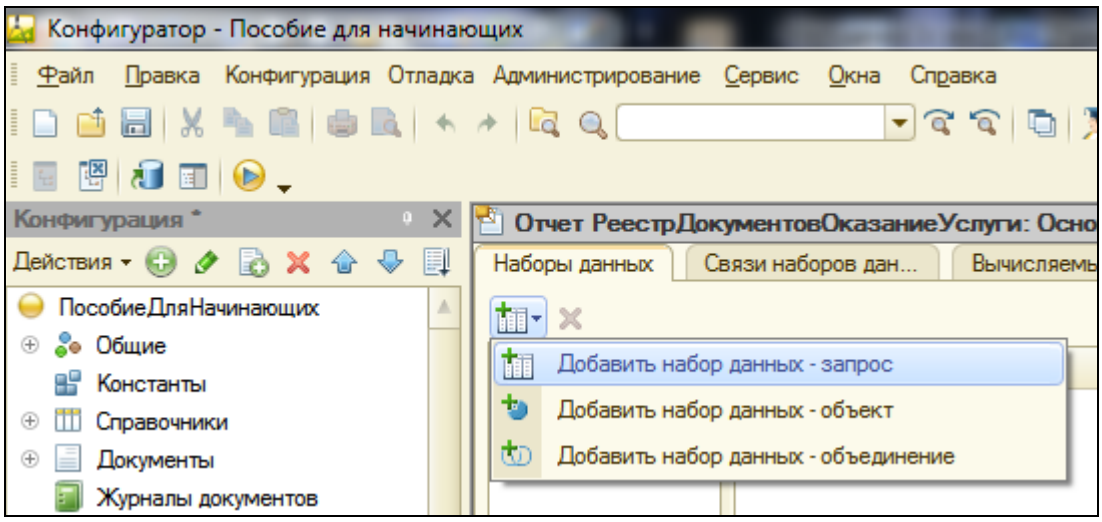
На закладке Основные зададим имя отчета – **РеестрДокументовОказаниеУслуги**.

Установите свойство **Расширенное представление** как **Список оказанных услуг** для представления отчета в интерфейсе программы.

Создайте схему компоновки данных для отчета. Для этого нажмите кнопку **Открыть схему компоновки данных** или кнопку со значком лупы.



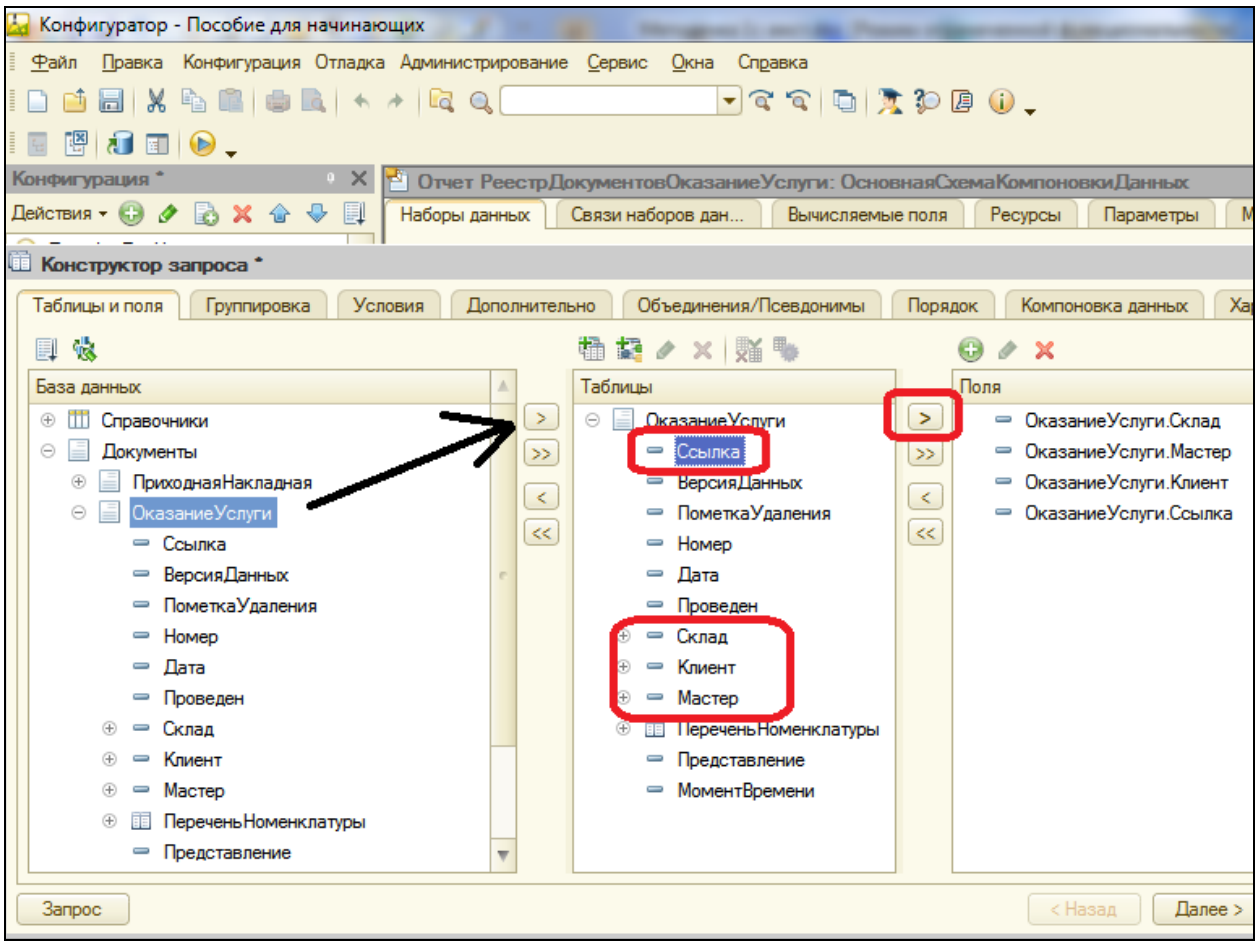
В открывшемся окне нажмите **Готово**. В конструкторе схемы компоновки создайте **Набор данных – запрос**.



Запрос для набора данных

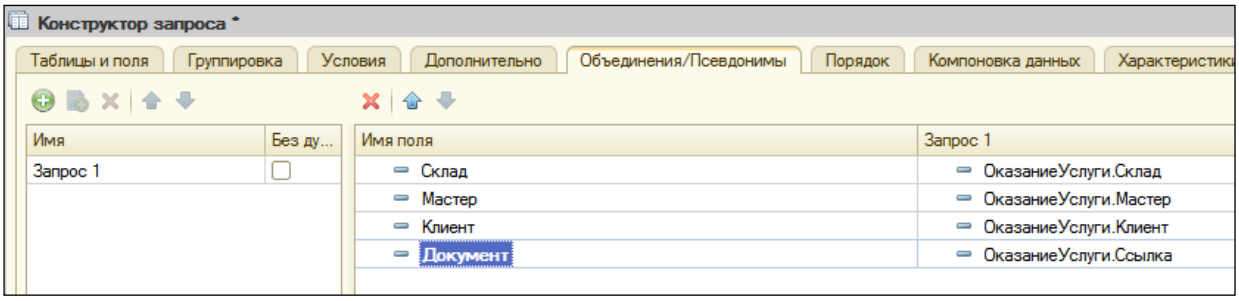
Нажмите кнопку Конструктор запроса. В качестве источника данных для запроса выберите объектную (ссылочную) таблицу документа ОказаниеУслуги. Из этой таблицы выберите следующие поля:

- Склад
- Мастер
- Клиент
- Ссылка

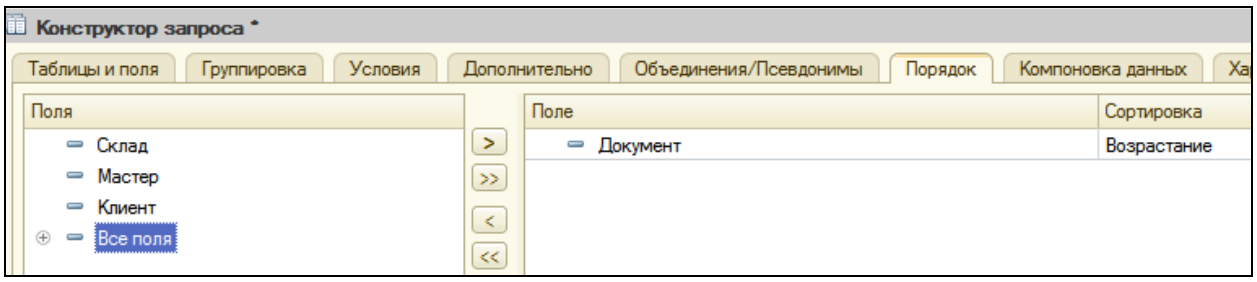


Перейдите на закладку **Объединения/Псевдонимы** и укажите, что поле **Ссылка** будет иметь псевдоним **Документ**.

Совет: имена полей лучше изменять в запросе, т.к. в этом случае в схему компоновки данных они перенесутся сразу в три колонки: Поле, Путь и Заголовок, и не нужно будет лишний раз их менять.



Перейдите в закладку **Порядок** и укажите, что результат запроса должен быть упорядочен по значению поля **Документ**.



Анализ текста запроса

Нажмите ОК и посмотрите, какой запрос сформировал конструктор:

```
ВЫБРАТЬ
	ОказаниеУслуги.Склад,
	ОказаниеУслуги.Мастер,
	ОказаниеУслуги.Клиент,
	ОказаниеУслуги.Ссылка КАК Документ
ИЗ
	Документ.ОказаниеУслуги КАК ОказаниеУслуги


УПОРЯДОЧИТЬ ПО
	Документ
```

Если Вы знакомы с языком SQL, то увидите, что это его аналог на русском.

Описание запроса начинается с обязательного ключевого слова **ВЫБРАТЬ**. Затем следует список полей выборки. В нем описываются поля, которые должны содержаться в результате запроса. Этот список может содержать как сами поля, так и выражения, вычисляемые на основе значений полей.

После ключевого слова **ИЗ** указываются *источники данных* – исходные таблицы запроса, содержимое которых обрабатывается в запросе.

После ключевого слова **КАК** указывается псевдоним источника данных, чтобы в дальнейшем обращаться к этому источнику через псевдоним.

Предложение УПОРЯДОЧИТЬ ПО позволяет сортировать строки в результате запроса. После этого оператора располагается выражение упорядочивания, которое представляет собой перечисление полей (выражений) и порядка вывода.

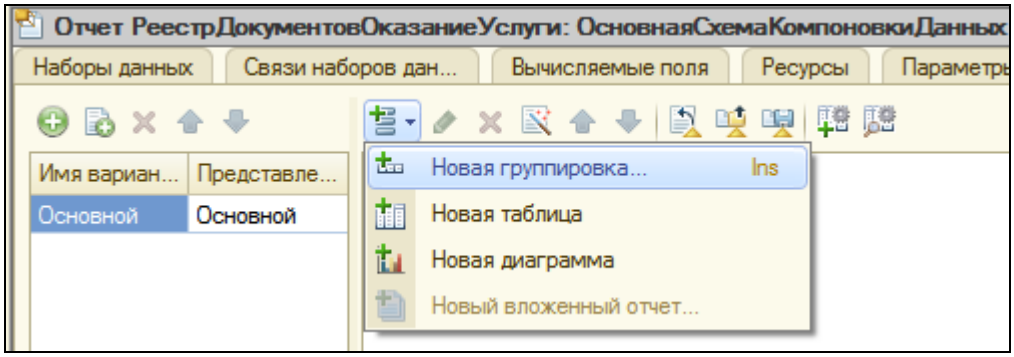
Настройки

Перейдите на закладку **Настройки** и создайте стандартные настройки, определяющие вывод информации в отчете.

Иерархическая структура отчета может содержать в различных сочетания три основных элемента:

- *Группировка* – для вывода информации в виде обычного линейного отчета.
- *Таблица* – для вывода информации в виде таблицы.
- *Диаграмма* – для вывода информации в виде диаграммы.

Для добавления нового элемента, в нашем случае группировки, выделите в дереве структуры отчета корневой элемент **Отчет** и вызовите его контекстное меню. Можно также нажать кнопку **Добавить**, расположенную в командной панели окна или нажать клавишу Insert.

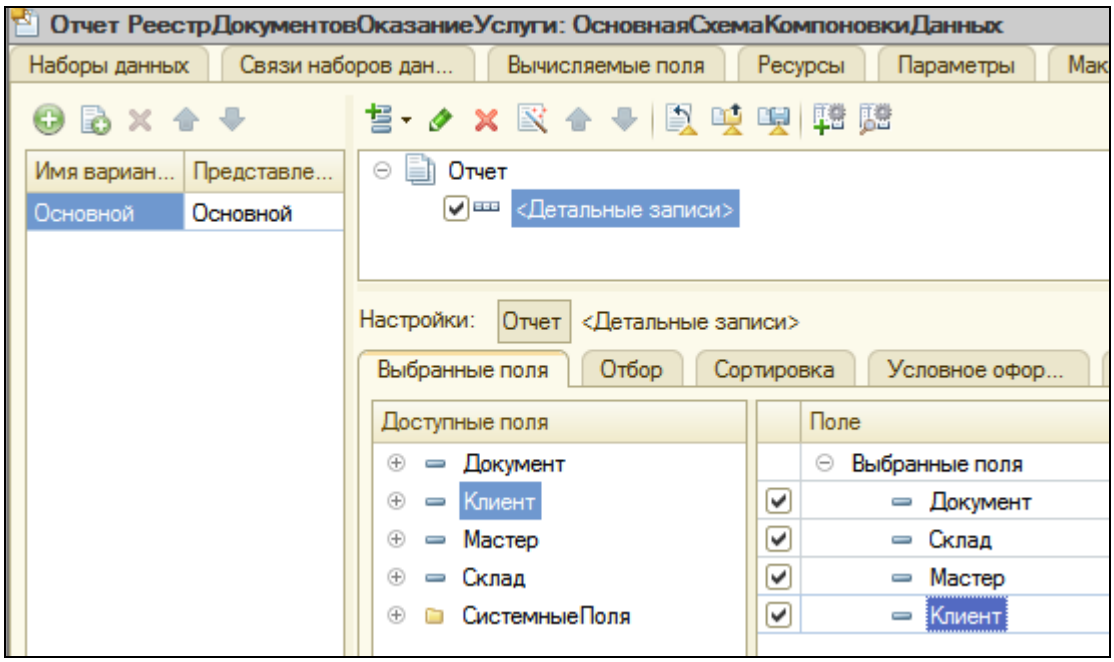


В окне выбора поля группировки просто нажмите ОК (указываем, что будут выводиться детальные записи из информационной базы).

В структуре отчета появится группировка **Детальные записи**.

На закладке **Выбранные поля** перенесите мышью из списка доступных полей те, которые будут выводиться в отчет:

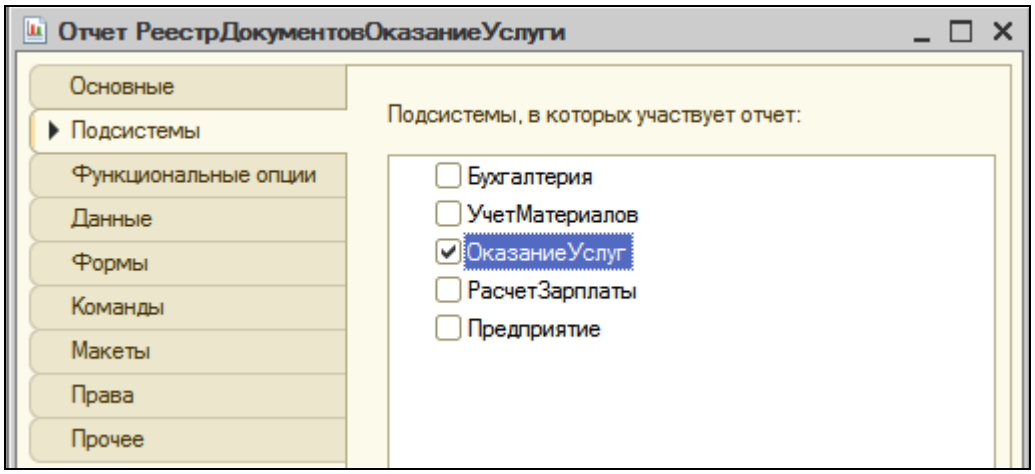
- Документ
- Склад
- Мастер
- Клиент



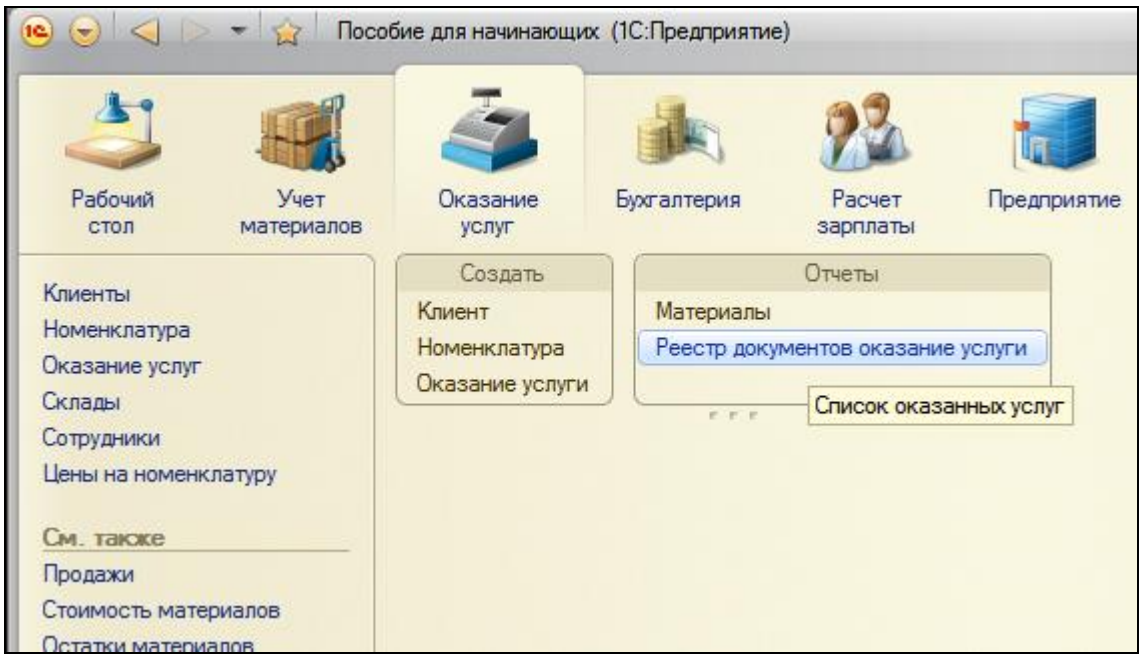
На этом создание отчета закончено. В заключение определим, в каких подсистемах будет отображаться наш отчет.

Закройте конструктор схемы компоновки данных и в окне редактирования объекта конфигурации Отчет **РеестрДокументовОказаниеУслгуи** перейдите на закладку **Подсистемы**.

Отметьте в списке подсистему **ОказаниеУслуг**. Так наш отчет попадет в панель действий этой подсистемы.



Запустите 1С: Предприятие в режиме отладки и посмотрите, как работает отчет.

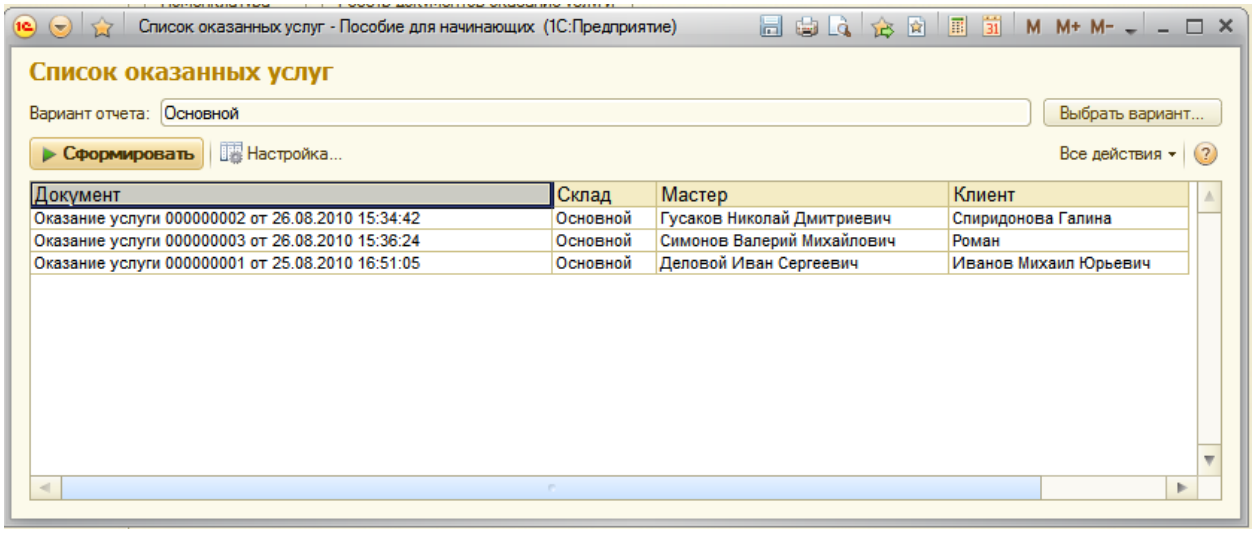


В открывшемся окне мы видим, что в панели действий раздела **Оказание услуг** в группе команд для выполнения отчетов появилась команда для формирования отчета **Реестр документов оказание услуги**. Причем если подвести к ней мышь, то появится всплывающая подсказка **Список оказанных услуг**, которая определяется свойством **Расширенное представление**, заданное нами для отчета.

Выполните эту команду и перед Вами откроется форма отчета, автоматически сформированная системой.

Заметьте, что заголовок этой формы называется **Список оказанных услуг** и определяется свойством **Расширенное представление**.

Нажмите кнопку **Сформировать**.



Отчет содержит реестр документов Оказание услуги. Двойным щелчком на поле **Документ** мы можем открыть исходный документ, а также выполнить другие действия расшифровки, которые предоставляет нам система компоновки данных.

Таким образом, на примере этого отчета мы показали, как использовать конструктор схемы компоновки данных и познакомились с некоторыми основными конструкциями языка запросов.

Выбор данных из двух таблиц

Отчет **Рейтинг услуг** будет содержать информацию о том, выполнение каких услуг принесло нашей фирме наибольшую прибыль в указанном периоде.

На примере этого отчета будет показано, 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 вы узнаете, как работать с параметрами системы компоновки данных, как использовать стандартные даты и быстрые пользовательские настройки отчетов.

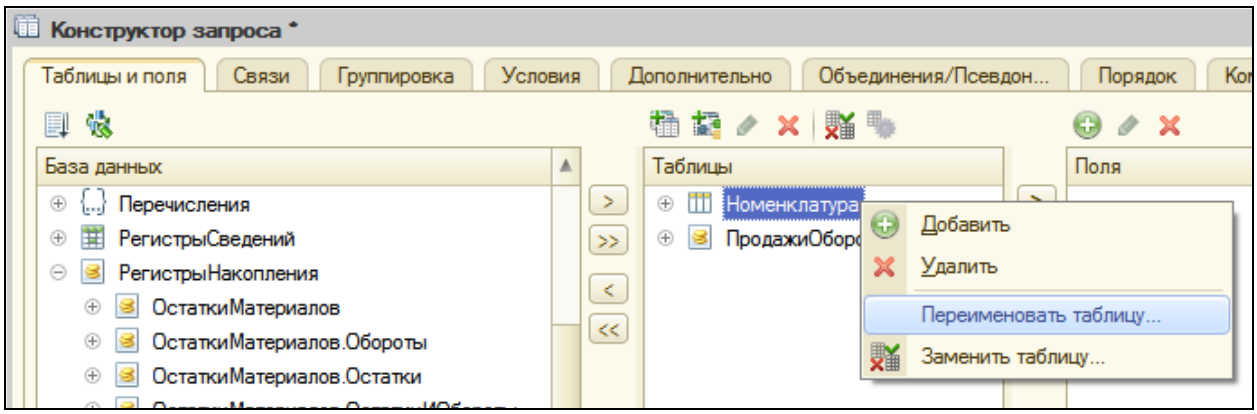
Закройте режим отладки. Добавьте новый объект конфигурации **Отчет**. Назовите его **РейтингУслуг** и запустите конструктор схемы компоновки данных. Добавьте новый **Набор данных – запрос** и вызовите **конструктор запроса**.

Запрос для набора данных

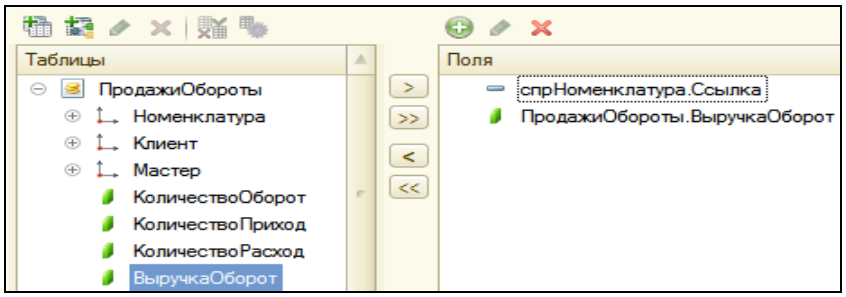
Левое соединение двух таблиц

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

Чтобы исключить неоднозначность имен в запросе, переименуйте таблицу **Номенклатура** в **спрНоменклатура**. Для этого выделите ее в списке **Таблицы**, в контекстном меню выберите **Переименовать таблицу**.

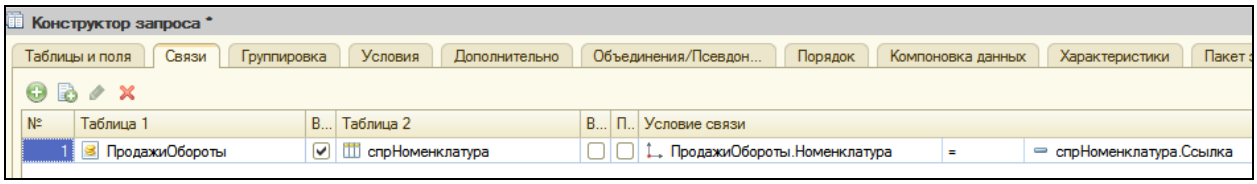


В список полей перенесите поля **СпрНоменклатура.Ссылка** и **ПродажиОбороты.ВыручкаОборот** из этих таблиц.

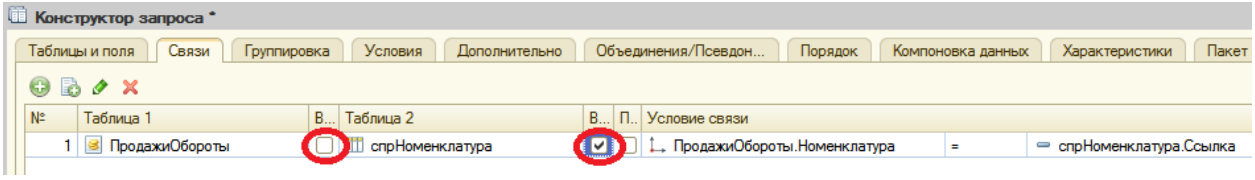


Перейдите на закладку **Связи**. Т.к. в запросе участвует несколько таблиц, необходимо определить связи между ними.

По умолчанию уже будет создана связь по полю **Номенклатура**. Т.е. значение измерения **Номенклатура** регистра **Продажи** должно быть равно ссылке на элемент справочника **Номенклатура**.



Нужно снять флажок **Все** у таблицы **ПродажиОбороты** и установить его там же у таблицы **спрНоменклатура**. Тем самым мы задаем тип связи **Левое соединение**, т.е. в результат запроса будут включены все записи справочника **Номенклатура** и те записи регистра **Продажи**, которые удовлетворяют условию связи по полю **Номенклатура**.



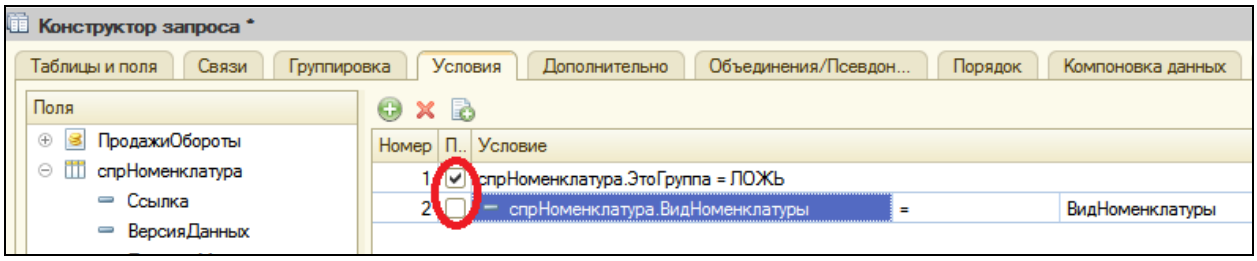
Условие отбора записей

Перейдите на закладку **Условия** и установите отбор, чтобы группы справочника **Номенклатура** не попадали в отчет. Для этого раскройте таблицу **спрНоменклатура**, перетащите мышью поле **ЭтоГруппа** в список условий, установите флажок **Произвольное** и напишите в поле **Условие** следующий текст:

```
спрНоменклатура.ЭтоГруппа = ЛОЖЬ
```

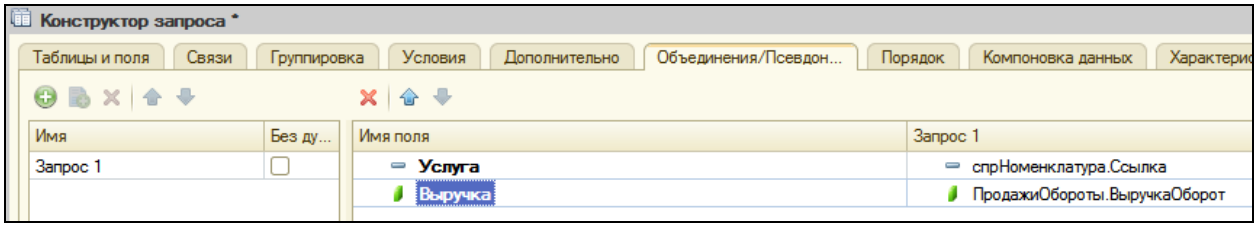
Тем самым мы указали, что из БД нужно выбрать только те записи справочника **Номенклатура**, которые не являются группами.

Создайте второе условие, что выбранный элемент является услугой. Это – **Простое условие**. Для его создания перетащите поле **ВидНоменклатуры** в список условий. Сформируется условие, когда вид номенклатуры должен быть равен значению параметра **ВидНоменклатуры**. В дальнейшем перед выполнением запроса мы передадим в параметр **ВидНоменклатуры** значение перечисления – **Услуга**.



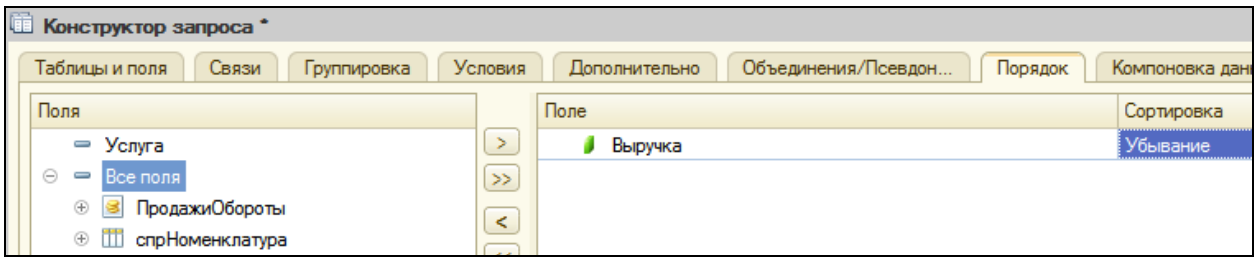
Псевдонимы полей

Перейдите на закладку **Объединения/Псевдонимы** и укажите, что поле **Ссылка** будет иметь псевдоним **Услуга**, а поле регистра будет иметь псевдоним **Выручка**.



Порядок записей

Перейдите на закладку **Порядок** и укажите, что результат запроса должен быть отсортирован по убыванию значения поля **Выручка**.



Создание запроса закончено, нажмите ОК и вернитесь в конструктор схемы компоновки данных.

Анализ текста запроса

Текст запроса, сформированный платформой, примет вид:

```
ВЫБРАТЬ
	спрНоменклатура.Ссылка КАК Услуга,
	ПродажиОбороты.ВыручкаОборот КАК Выручка
ИЗ
	Справочник.Номенклатура КАК спрНоменклатура
		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
		ПО ПродажиОбороты.Номенклатура = спрНоменклатура.Ссылка
ГДЕ
	спрНоменклатура.ЭтоГруппа = ЛОЖЬ
	И спрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры


УПОРЯДОЧИТЬ ПО
	Выручка УБЫВ
```

Сначала идет часть описания запроса и в ней есть новые для нас конструкции.

При описании источников запроса (после ИЗ) использована возможность определения нескольких источников запроса. В данном случае выбираются записи из двух источников: **спрНоменклатура** и **ПродажиОбороты**, причем ключевым предложением ЛЕВОЕ СОЕДИНЕНИЕ…ПО описан способ, которым будут соединены 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.

В части описания запроса есть еще одна новая конструкция – задание условий отбора данных из исходных таблиц.

```
ГДЕ
        спрНоменклатура.ЭтоГруппа = ЛОЖЬ
        И спрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры
```

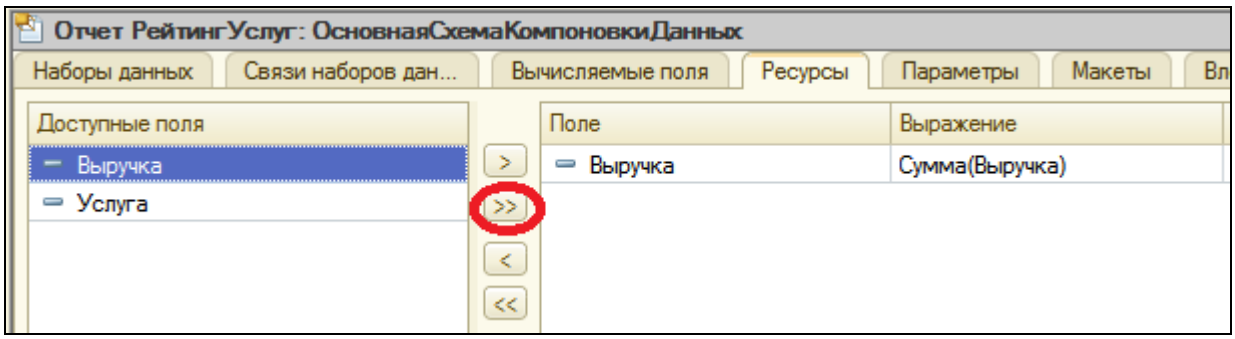
Условию отбора всегда предшествует оператор ГДЕ. После него описывается условие. Обратите внимание, что поля исходных таблиц, на которые накладывается условие, могут и не входить в список выборки (как в нашем случае). Кроме того, в нашем условии использован параметр запроса ВидНоменклатуры (перед именем параметра указывается символ & - амперсанд). & - Используется в языке запросов для указания системе, что далее будет использовано имя внешнего параметра. Имя задается в соотвествии с правилами формирования идентификаторов. Внешние параметры используются в тех случаях, когда в момент формирования системой текста запроса в режиме исполнения этот параметр может быть заменен конкретным значением. Для определения значения параметра необходимо использовать метод объекта встроенного языка "Запрос" - УстановитьПараметр().

Ресурсы

В нашем отчете мы хотим видеть итоговые значения выручки для каждой услуги. Для этого нам нужно определить поля ресурсов отчета.

Ресурсы в системе компоновки – поля, значения которых рассчитываются на основании детальных записей, входящих в группировку. По сути, ресурсы являются групповыми или общими итогами отчета.

Итоговые данные формируются на закладке **Ресурсы**. Перейдите туда и выберите все доступные ресурсы, по которым можно вычислять итоги. В нашем случае это единственный ресурс **Выручка**. По умолчанию рассчитывается сумма этого поля, что нам и нужно.



Параметры

Параметры отчета задают условия отбора записей в отчет. Перейдите на закладку **Параметры**.

Вы увидите еще два параметра, которые не задавали – начало периода и конец периода расчета итогов. Они задаются с точностью до секунды.

Допустим, заранее известно, что пользователю не нужно будет задавать период с точностью до секунды.

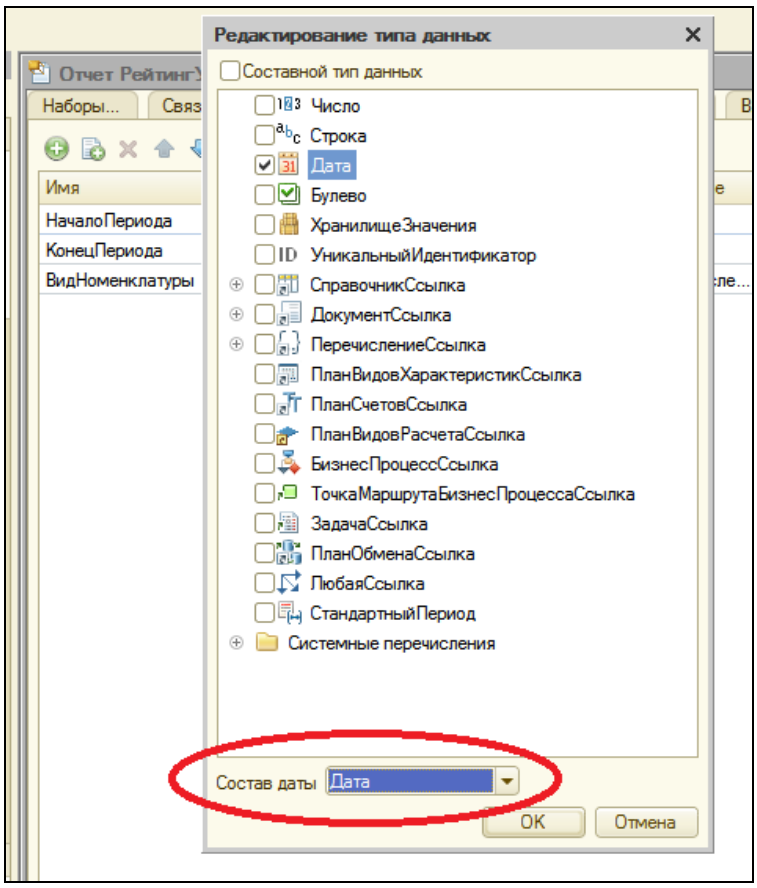
Во-первых, для этого нужно избавить пользователя от необходимости указывать время при вводе даты периода, за который формируется отчет.

Изменим существующее описание типа для параметра **НачалоПериода**.

Вернитесь на закладку **Параметры** схемы компоновки данных и дважды щелкните на ячейке **Тип** параметра **НачалоПериода**. Нажмите кнопку выбора и в нижней части окна редактирования типа данных установим **Состав даты** в значение **Дата**.

Во-вторых, по умолчанию время в дате установлено 00:00:00, поэтому если пользователь задаст период отчета с 01.07.2011 по 14.07.2011, итоги регистра будут рассчитаны с начала дня 01.07 00:00:00 по начало дня 14.07 00:00:00. Т.о. данные за 14-е число, отличные от начала дня, в расчет не войдут, что сильно удивит пользователя.

Чтобы исключить эту ситуацию, добавим еще один параметр, в который пользователь будет вводить дату окончания. А значение параметра КонецПериода будем рассчитывать автоматически так, чтобы оно указывало на конец дня даты, введенной пользователем.



Для параметра **КонецПериода** установите флажок Ограничение доступности.

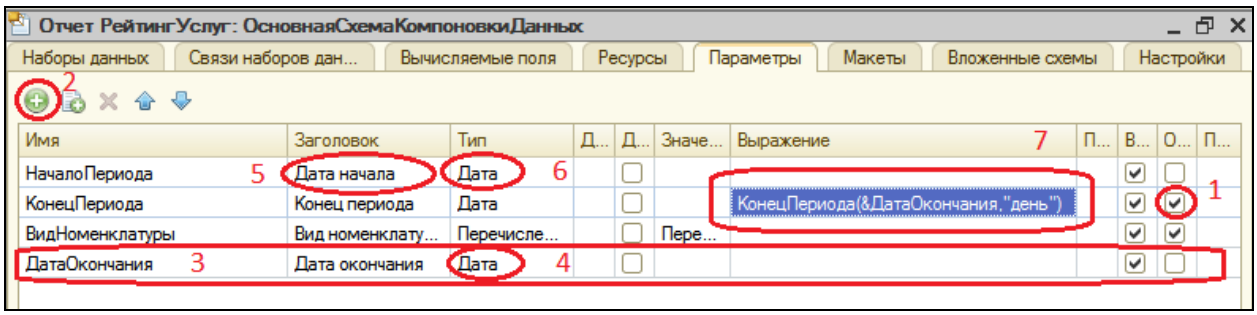
Если этот флажок не установлен, то параметр будет доступен для настройки пользователем. Если же установить, то пользователь не увидит этот параметр.

Нажмите кнопку **Добавить** – параметр с именем **ДатаОкончания**. Автоматически сформируется заголовок **Дата окончания**. Оставим его без изменений. Тип значения – **Дата**. При этом, как и для параметра **НачалоПериода**, укажите состав даты – **Дата**. Для параметра **НачалоПериода** задайте заголовок, который будет отображаться пользователю – **Дата начала**.

Перейдите к параметру **КонецПериода**. Зададим формулу вычисления значения этого параметра через язык выражений системы компоновки. В нем есть функция **КонецПериода()**, которая позволяет получить дату, соответствующую концу какого-либо периода.

В ячейке Выражение напишите следующее выражение:

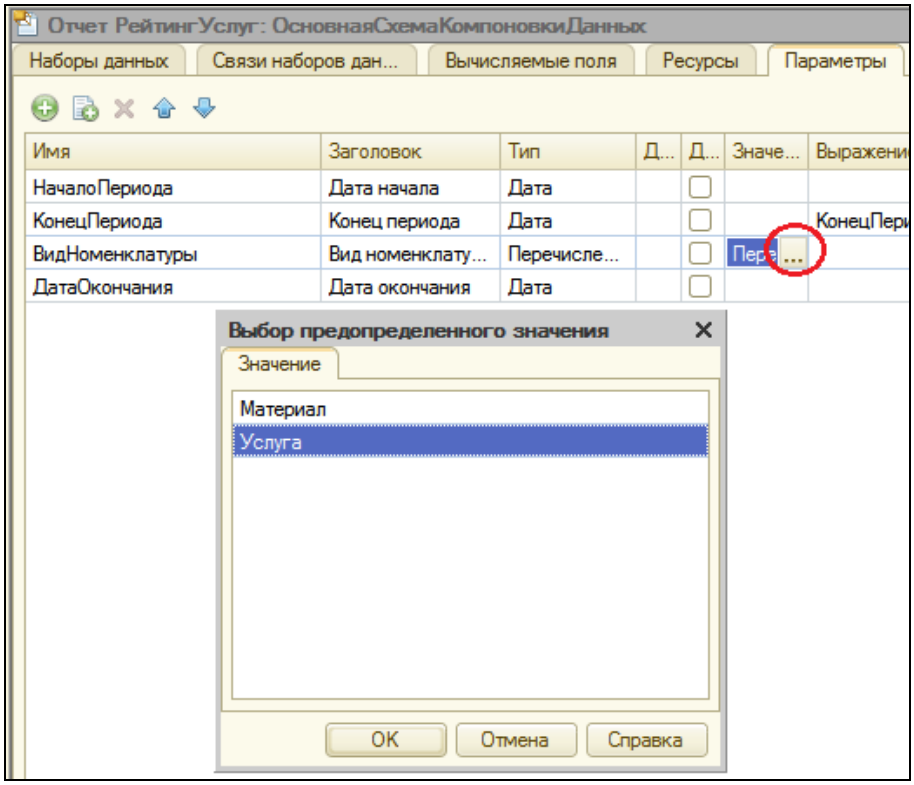
```
КонецПериода(&ДатаОкончания,"день")
```

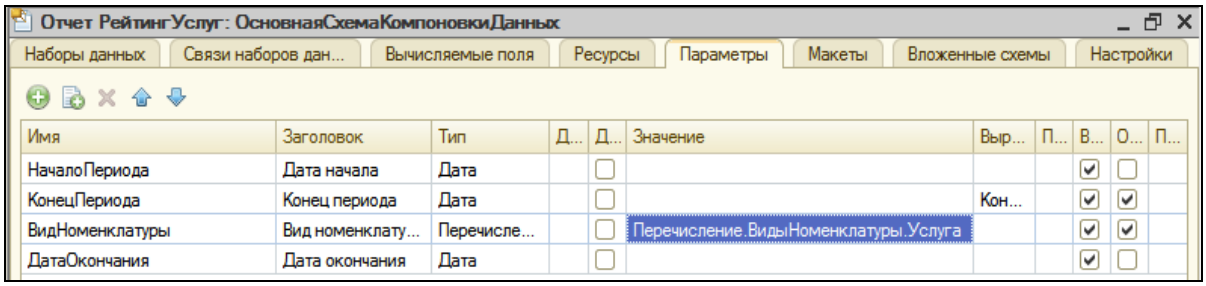


Настроим параметр **ВидНоменклатуры**.

Поскольку отчет должен отображать выручку, полученную только от реализации услуг, значение параметра **ВидНоменклатуры** пользователь изменять не должен. Оно должно быть задано непосредственно в схеме компоновки как **Перечисление.ВидыНоменклатуры.Услуга**.

Воспользуйтесь кнопкой выбора значения и выберите из списка **Услуга**.

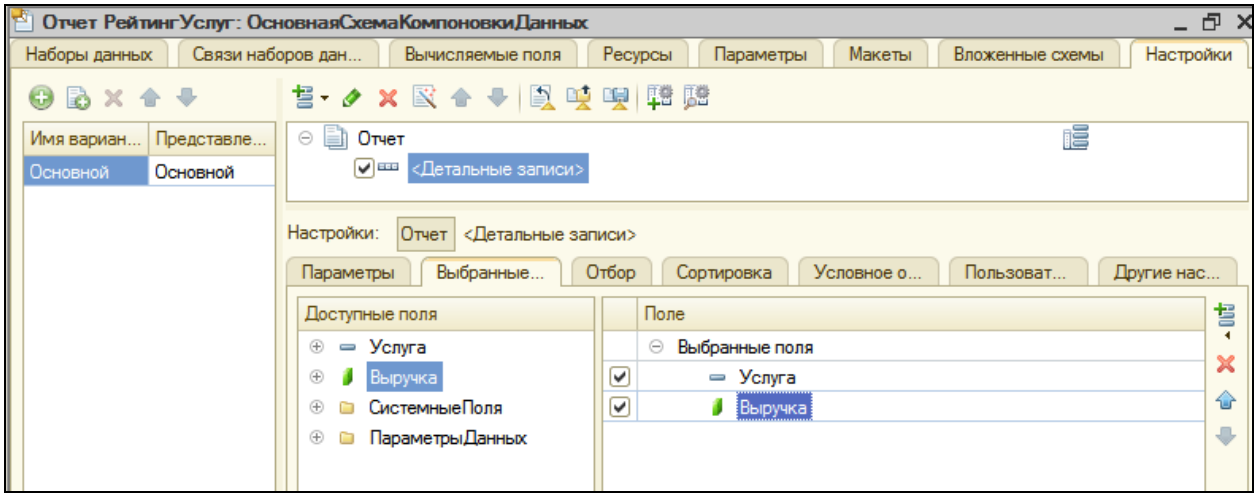




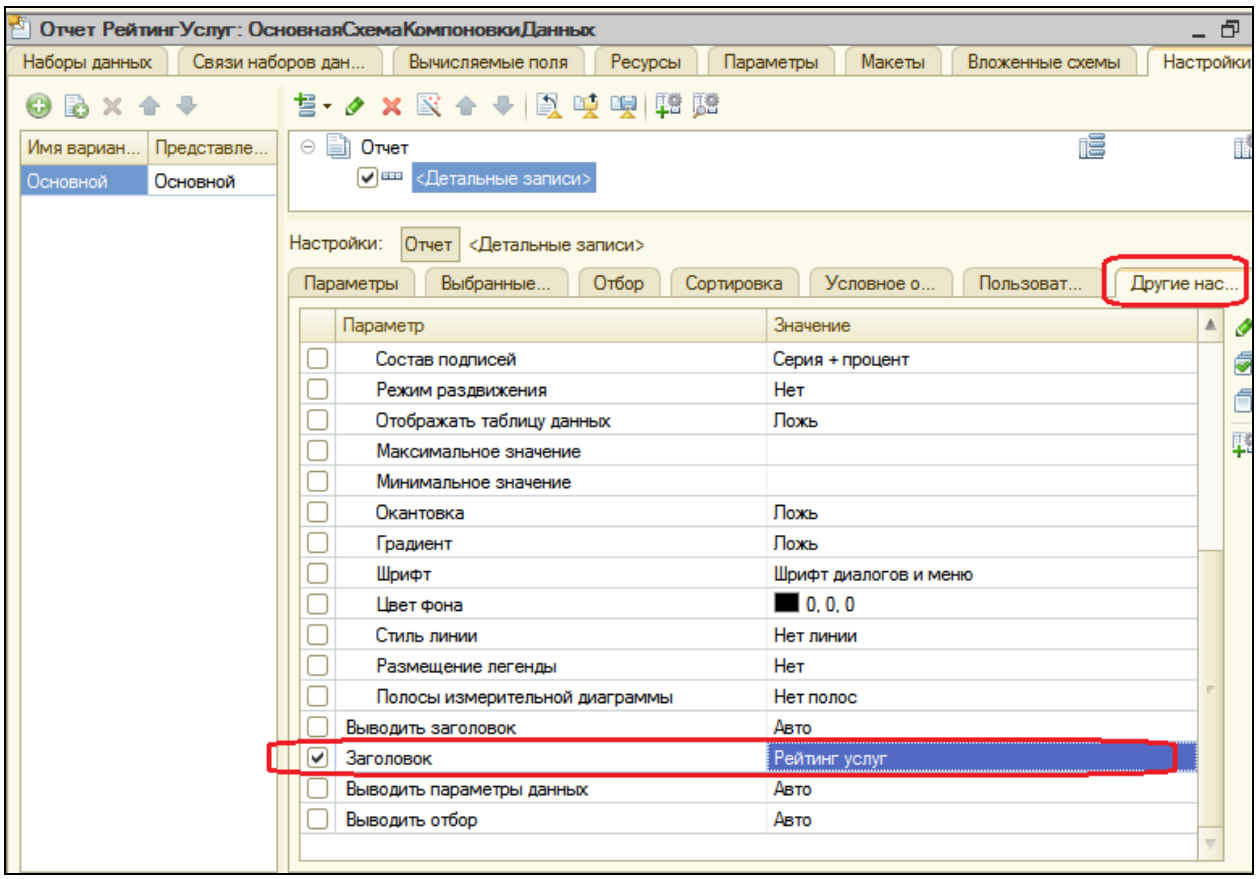
Настройки

Перейдем к формированию структуры отчета. На вкладке **Настройки** добавьте группировку и не указывайте поле группировки.

На закладке **Выбранные поля** укажите поля **Услуга** и **Выручка**.



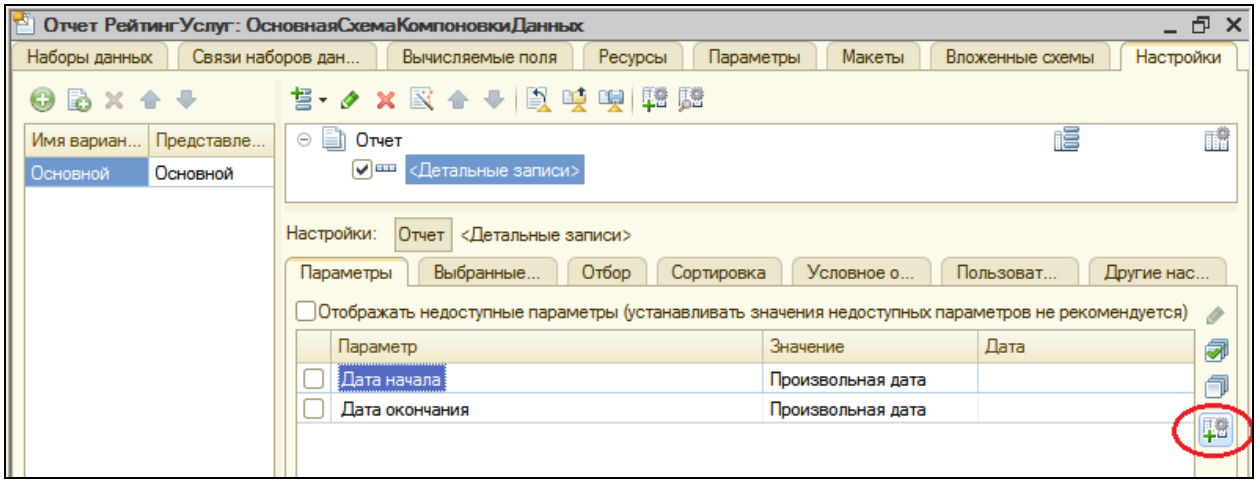
Затем перейдите на вкладку **Другие настройки** и задайте заголовок отчета – **Рейтинг услуг**.



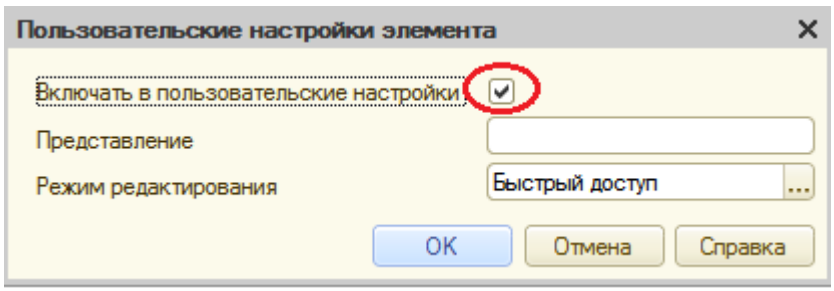
Быстрые пользовательские настройки

В заключение мы должны предоставить пользователю возможность задавать отчетный период перед формированием отчета. Т.е. параметры **ДатаНачала** и **ДатаОкончания** должны быть включены в состав пользовательских настроек, отображаемых в форме отчета.

На вкладке **Параметры** выделите по очереди каждый из параметров и нажмите кнопку **Свойства элемента пользовательских настроек** в правом нижнем углу окна настроек.

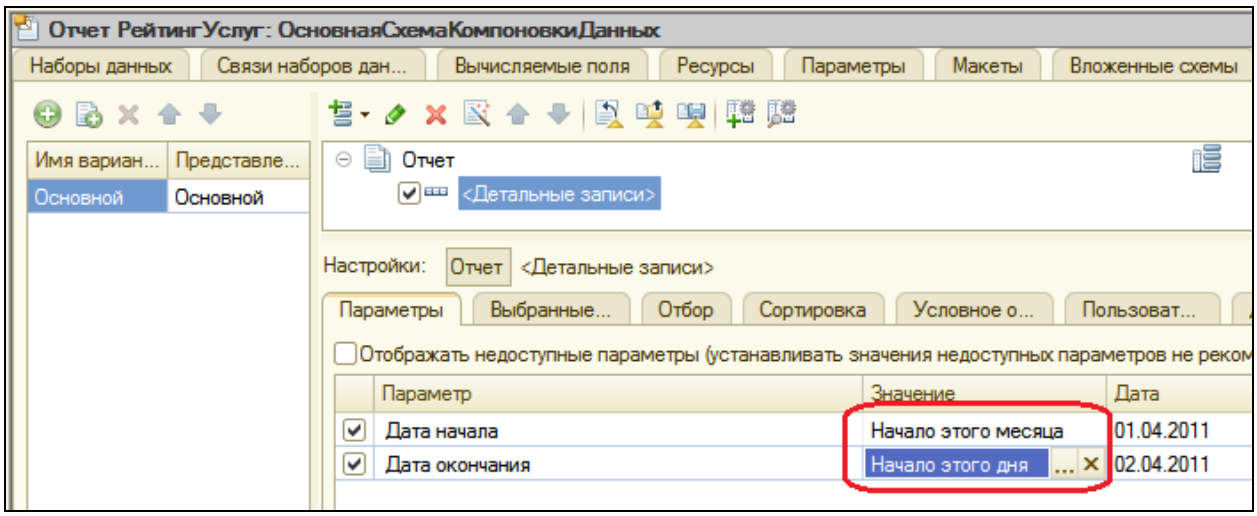


Установите флажок **Включать в пользовательские настройки** и оставьте предложенное по умолчанию значение **Быстрый доступ**.

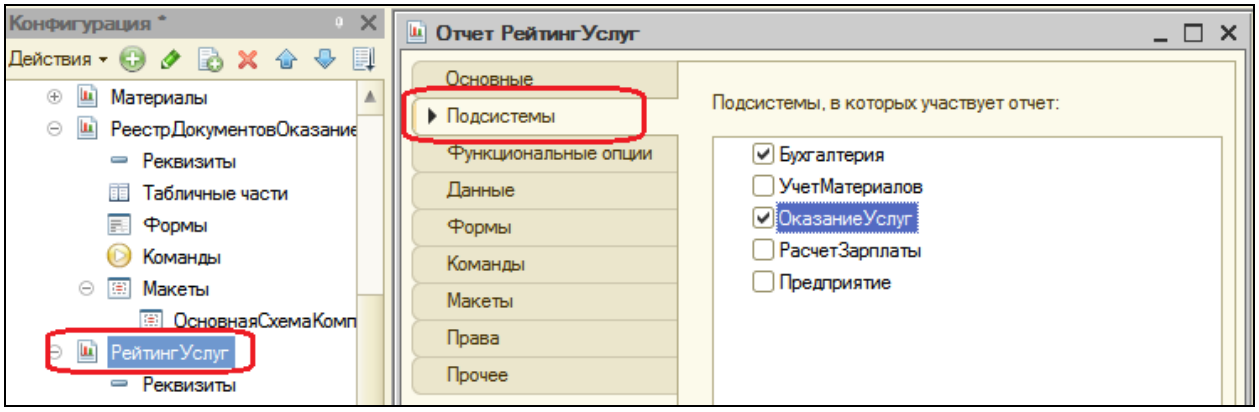


При этом пользователь будет иметь быстрый доступ к настройке даты начала и окончания прямо на форме отчета.

Для улучшения интерфейса пользователя, зададим для параметров **Дата Начала** и **Дата окончания** в качестве начальных значений соответственно **Начало этого месяца** и **Начало этого дня**. Т.о., при выполнении отчета, даты начала и окончания отчетного периода будут динамически меняться и показывать период с начала текущего месяца по сегодняшнее число и пользователю, возможно, не придется менять их вручную.



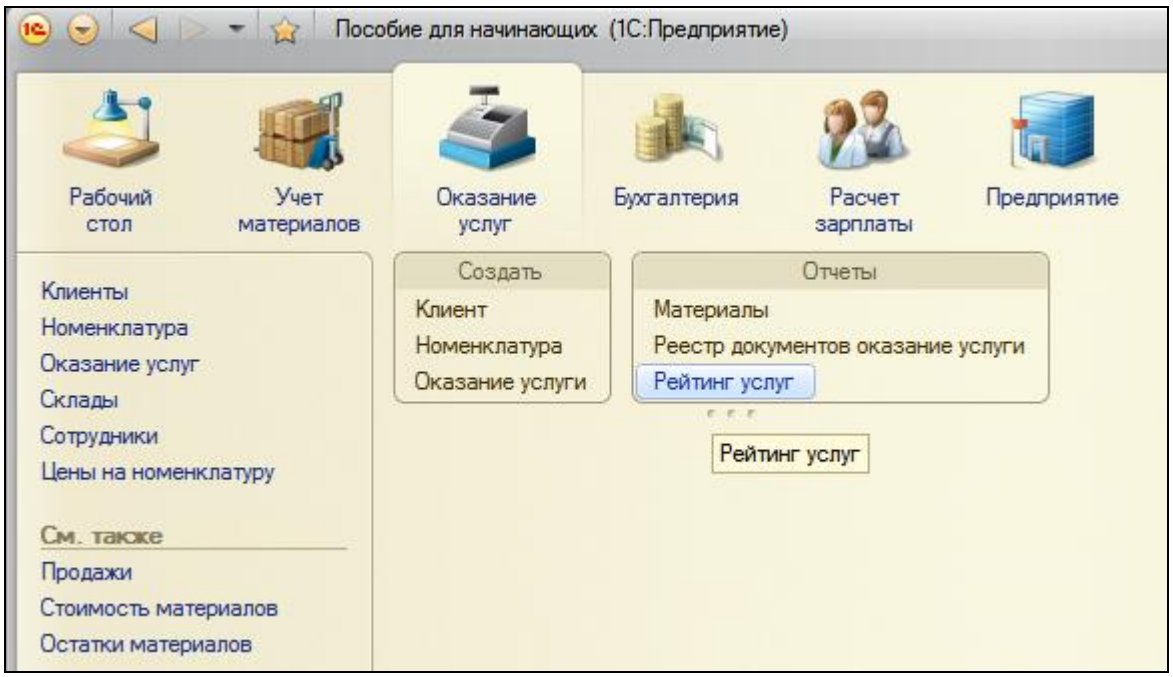
Определим подсистемы для отображения отчета. Закройте конструктор схемы компоновки и в окне редактирования объекта Отчет **РейтингУслуг** перейдите на вкладку **Подсистемы**, отметьте **ОказаниеУслуг** и **Бухгалтерия**.

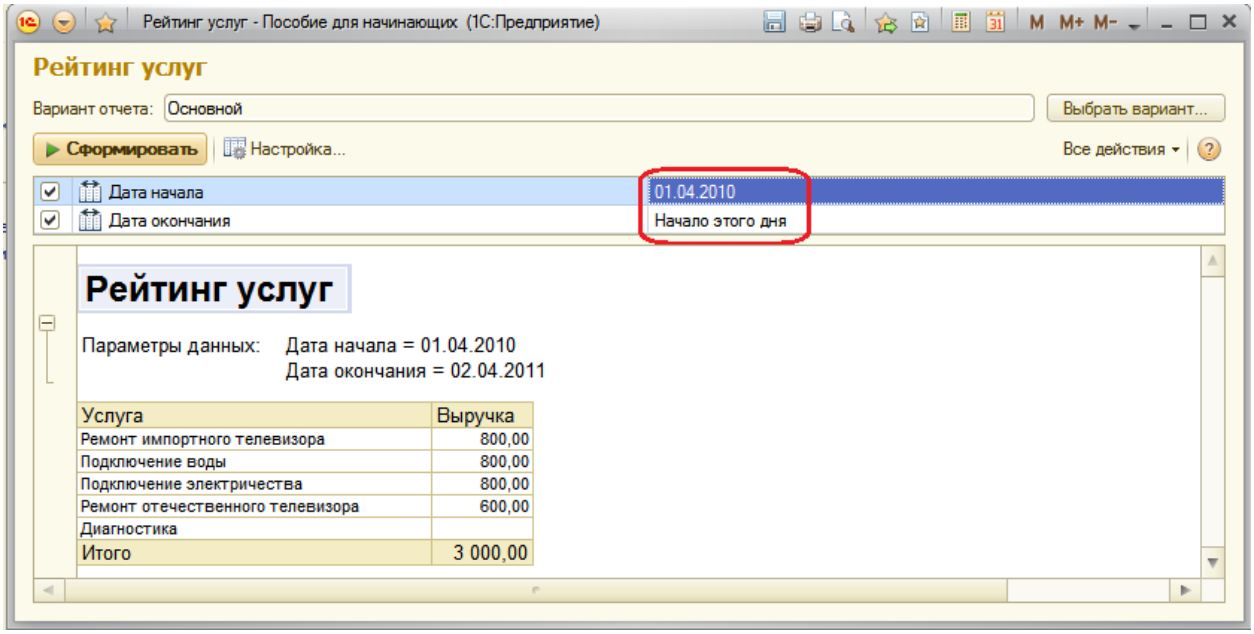


Таким образом, ссылка на наш отчет автоматически попадет в панель действий этих подсистем.

Запустите 1С: Предприятие в режиме отладки, вкладка **Оказание услуг**. Нажмите на **Рейтинг услуг**, **Сформировать**.

Учтите, что диапазон дат для отображения выручки должен попадать в диапазон заполнения документов в предыдущих работах.





Настройки в конфигураторе и в режиме 1С: Предприятие

Теперь на примере этого отчета покажем создание и использование других настроек отчета - **Условное оформление** и **Отбор**. В процессе мы будем периодически переходить из Конфигуратора в Предприятие и обратно.

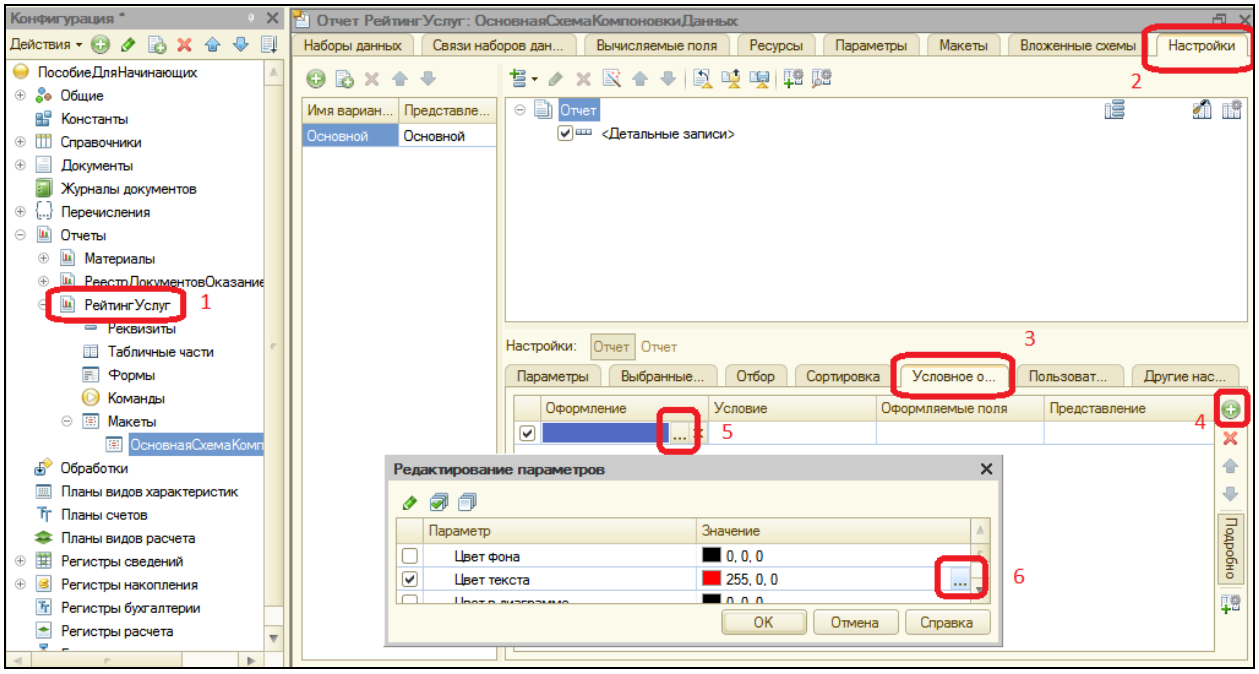
Условное оформление

В отчете **Рейтинг услуг** удобно было бы выделять цветом записи отчета, содержащие услуги с наименьшей или с наибольшей выручкой или другим условием.

В режиме Конфигуратор

В конфигураторе откройте схему компоновки данных на закладке **Настройки**.

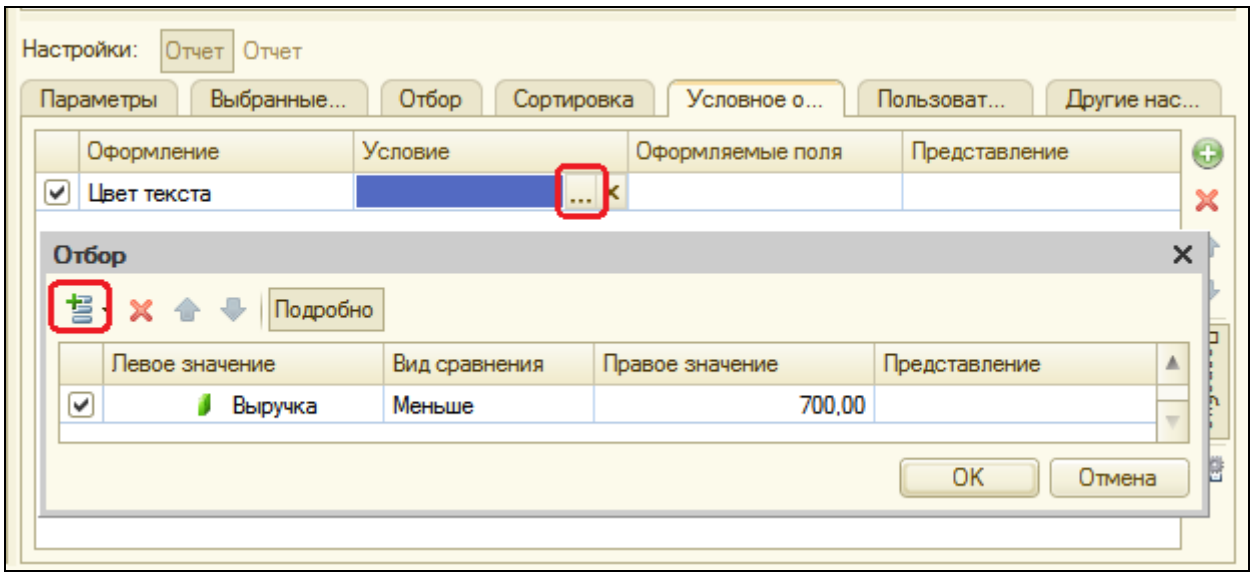
В нижней части окна перейдите на закладку **Условное оформление** и нажмите кнопку **Добавить** в верхнем углу окна настроек. Укажите **Оформление** – красный цвет текста.



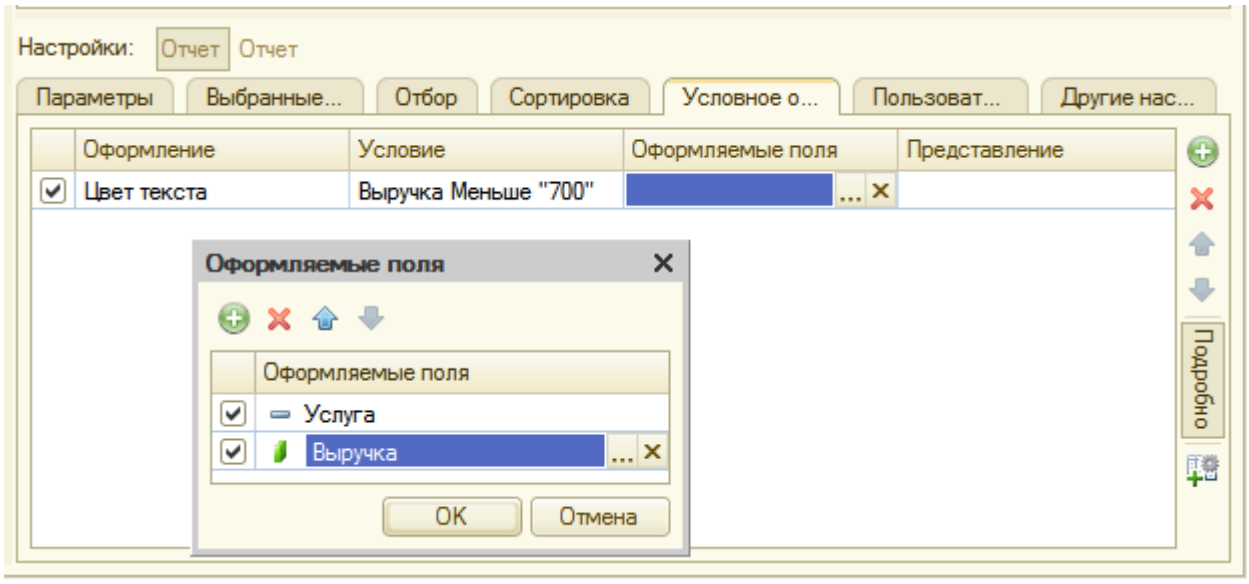
Затем укажем **Условие**, при наступлении которого будет применяться оформление (красный текст).

Нажмите кнопку выбора  в поле Условие и в появившемся окне добавьте **Новый элемент** отбора. Каждый элемент задает одно условие. **Условий** может быть несколько. Нажмите кнопку **Добавить** и укажите в графе **Левое значение** – поле **Выручка**, в графе **Вид сравнения** – **Меньше**, в **Правое значение** – **700.** Нажмите ОК.

Т.е. когда в поле **Выручка** окажется значение меньше 700, что-то будет выделено красным цветом текста.



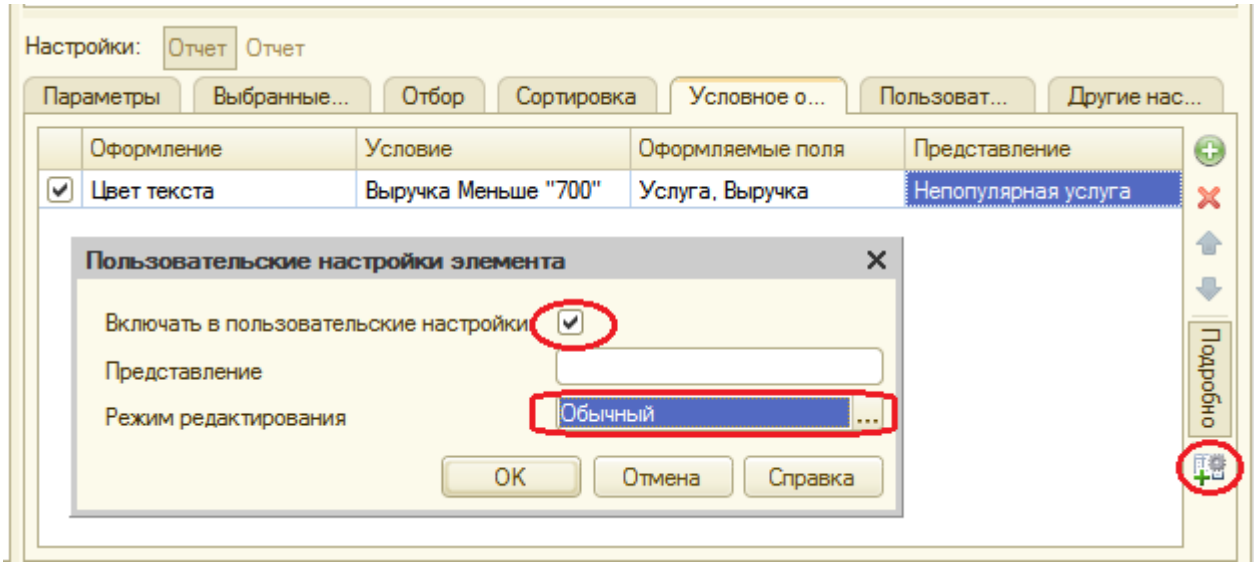
Теперь укажем это «что-то», т.е. зададим список оформляемых полей. Если мы хотим выделять всю строку отчета, то можно оставить этот список пустым. Или же нажать кнопку выбора … в поле **Оформляемые поля** и в появившемся окне **Добавить** поля **Услуга** и **Выручка**.



В нашем случае этого можно было и не делать, т.к. **Услуга** и **Выручка** и есть все поля отчета. Нажмите ОК.

Зададим **Представление** условного оформления как **Непопулярная услуга**. Это то, что увидит пользователь в своих настройках. Т.е. вместо пугающей строки «Выручка меньше 700» он увидит осмысленное выражение, которое задано в поле **Представление**.

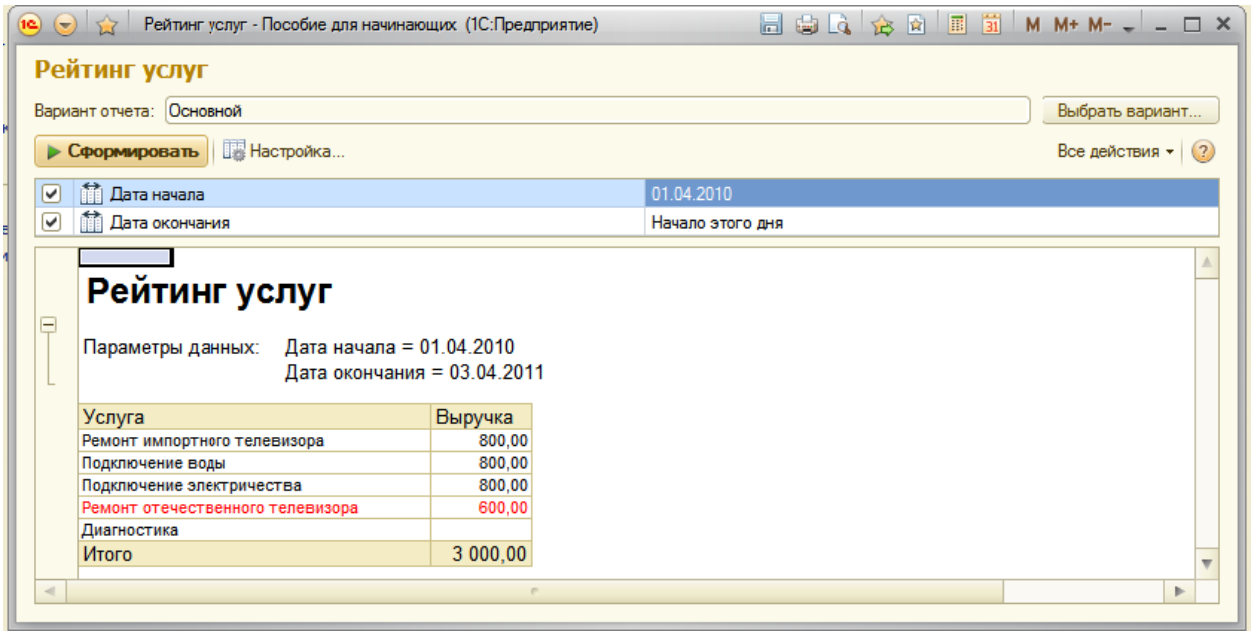
Мы задали условное оформление отчета, по которому все услуги с выручкой менее 700 руб. будут считаться непопулярными и выделяться красным цветом. Теперь добавьте это условие в пользовательские настройки.



Тем самым Вы включили созданную настройку условного оформления в обычные пользовательские настройки. Эти настройки расположены не в форме отчета, а вызываются нажатием кнопки **Настройка** и появляются в отдельном окне, т.к. эти настройки используются реже, чем настройки отчетного периода.

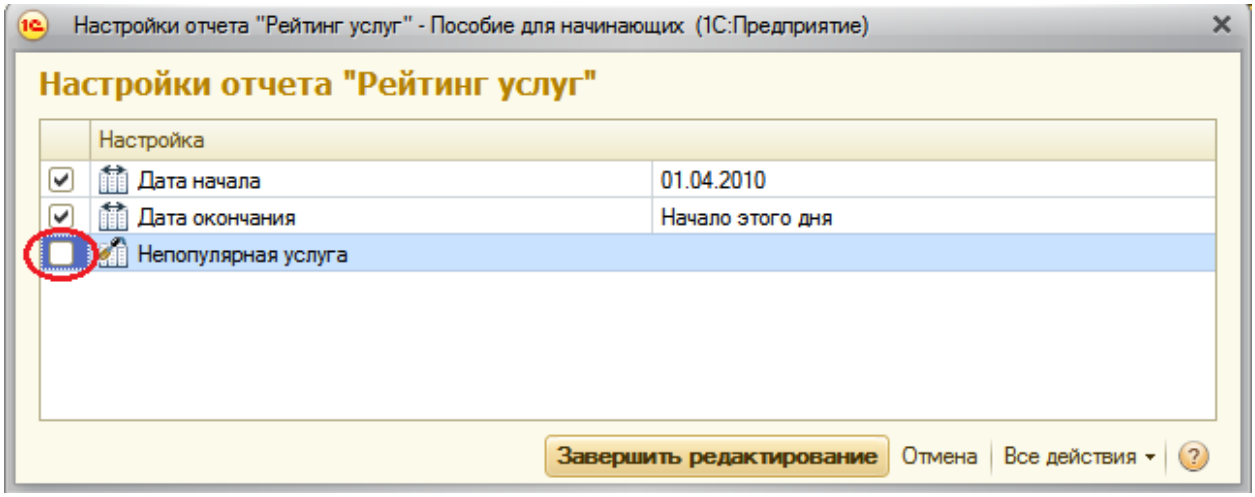
В режиме 1С: Предприятие

Перейдите в режим отладки. Вызовите отчет. Задайте **Дату окончания** как **Начало этого дня** и нажмите **Сформировать**.

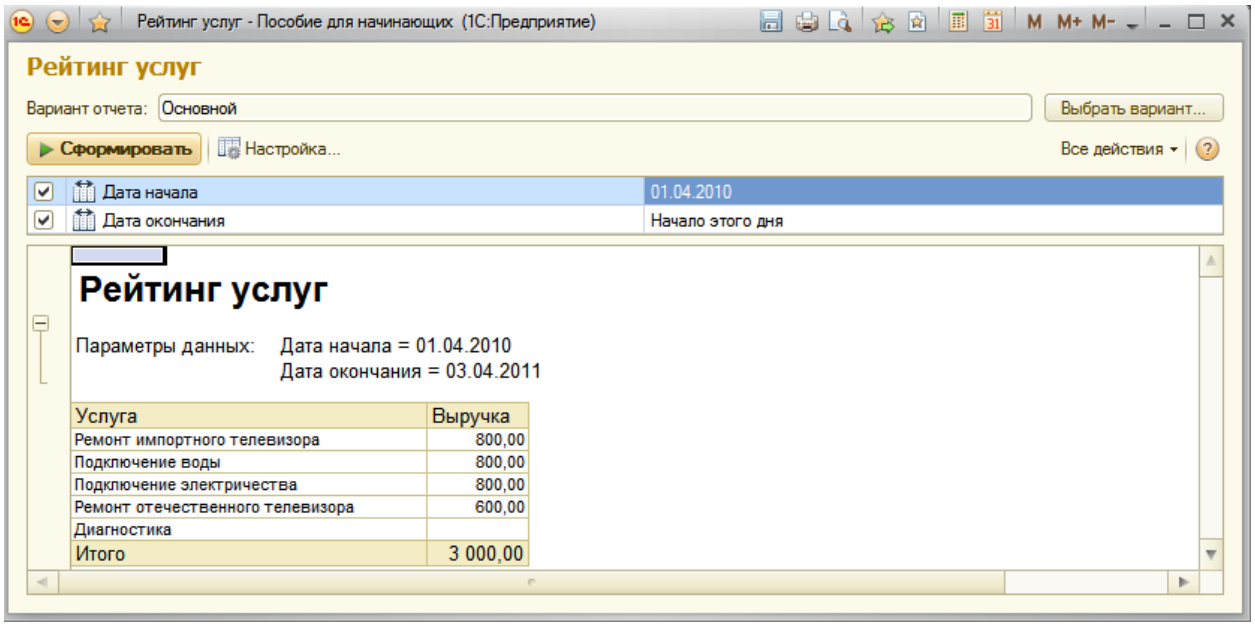


Как видите, сумма услуг менее 700 руб. выделена красным. Нажмите кнопку **Настройка**. Появится окно пользовательских настроек отчета, содержащее параметры отчетного периода и настройку условного оформления **Непопулярная услуга**.

Снимите флажок использования этой настройки и нажмите **Завершить редактирование**, снова сформируйте отчет.



Выделение цветом исчезнет.



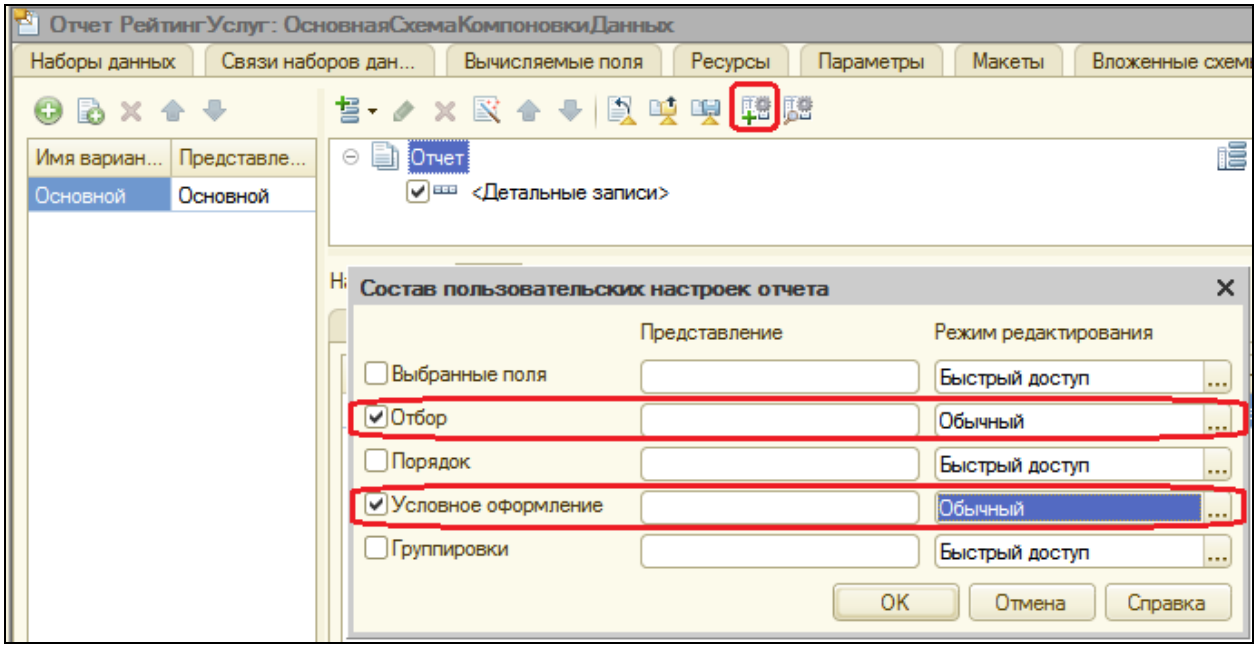
Данная настройка условного оформления задана жестко и пользователь может лишь включить или выключить ее. Но для более продвинутых пользователей мы можем предоставить более широкую свободу в использовании настроек, т.е. возможность самостоятельно задавать настройки отчета: отбор, порядок, оформление и прочие. Рассмотрим их в следующем примере.

Пользовательские настройки

В режиме Конфигуратор

Вернитесь в конфигуратор.

На закладке **Настройки** схемы компоновки нажмите кнопку **Свойства элемента пользовательских настроек**  , расположенную сверху в командной панели окна настроек. Установите галочку использования для настроек **Отбор** и **Условное оформление** и свойство **Режим редактирования** – **Обычный**.

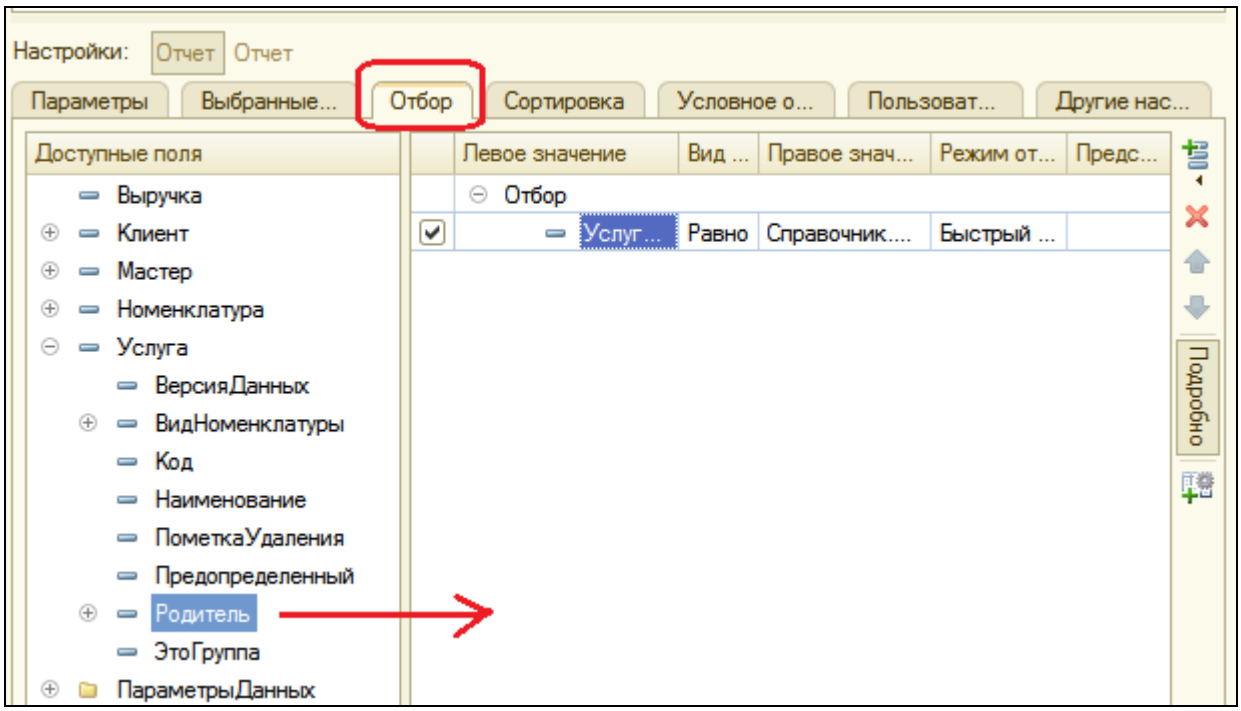


Таким образом, мы включили настройки отбора и оформления в состав пользовательских настроек и предоставили пользователю возможность задавать их в отдельном окне, вызываемом кнопкой **Настройка**.

Отбор

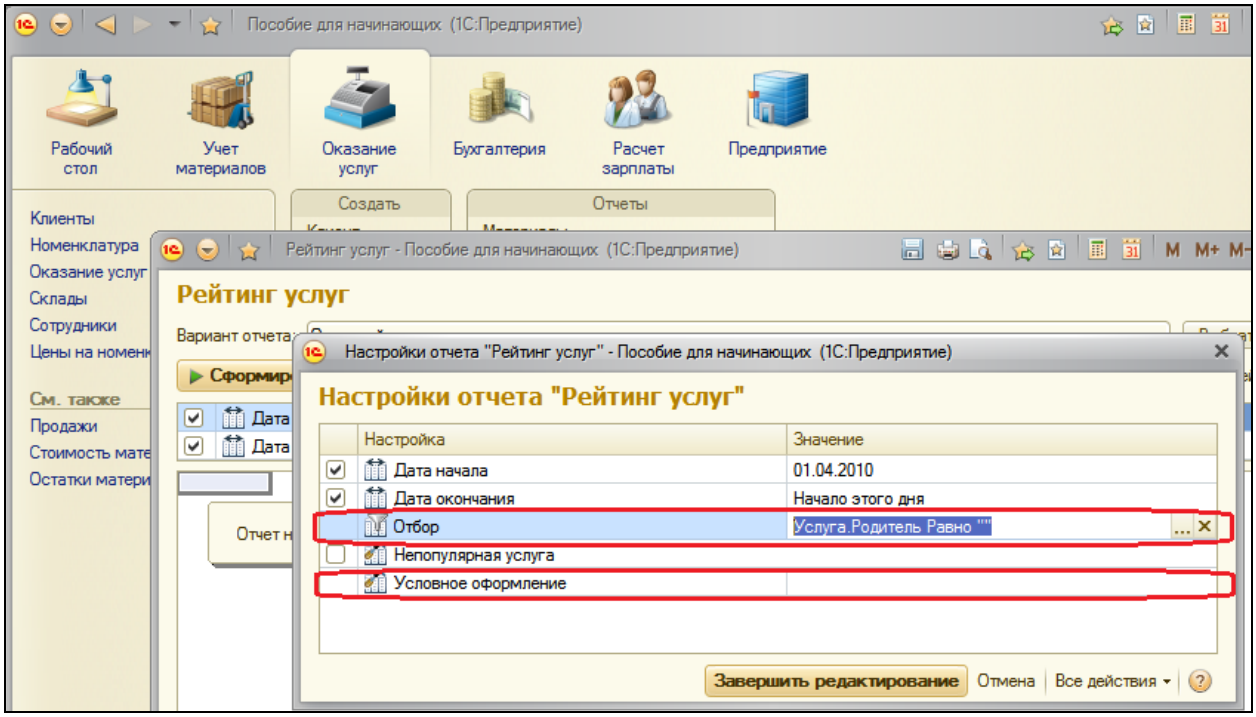
В режиме Конфигуратор

Теперь создадим настройку отбора в отчете. Для этого в нижней части окна настроек перейдем на закладку Отбор. Слева увидим список доступных полей отчета. Раскройте поле **Услуга** и перенесите поле **Родитель** в список условий отбора в правой части окна. Таким образом, мы создали возможность отбора по группам услуг, которые пользователь может задать в режиме 1С: Предприятие.



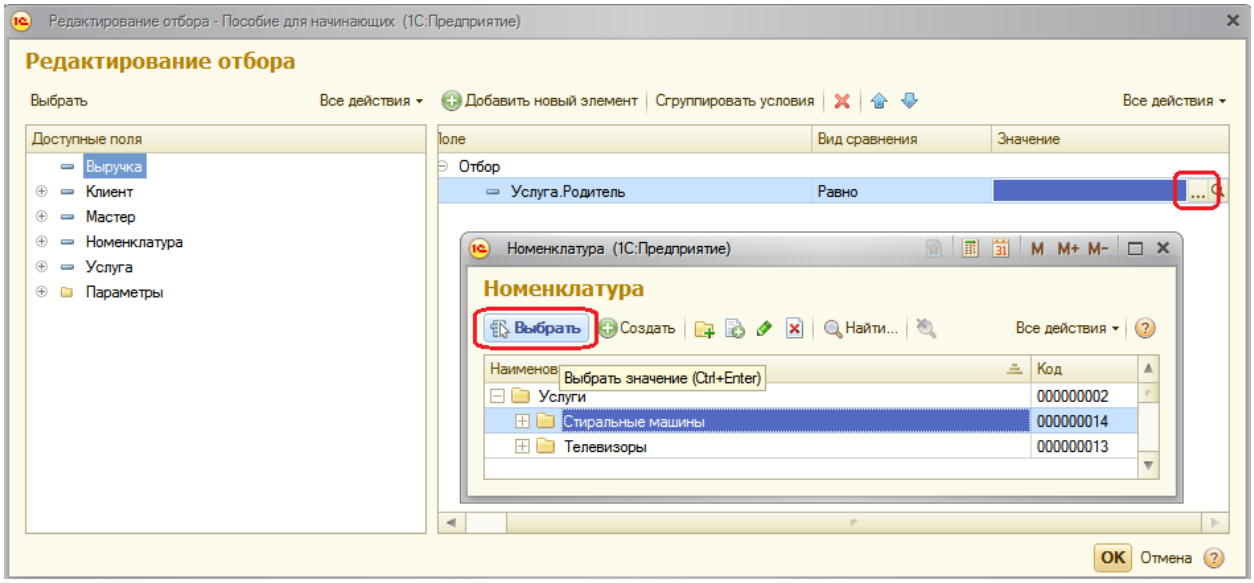
В режиме 1С: Предприятие

Откройте отчет в режиме 1С: Предприятие и нажмите кнопку **Настройка**. Появились настройки **Отбор** и **Условное форматирование**.



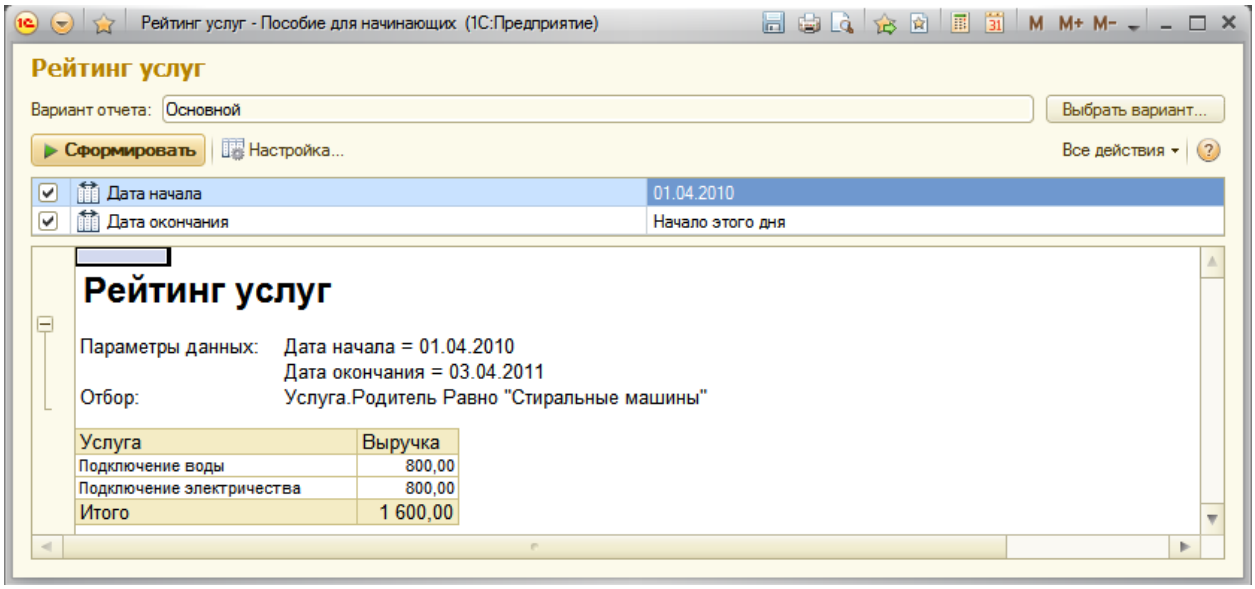
Сейчас мы зададим отбор в отчете так, чтобы в него попадали только услуги, относящиеся к установке стиральных машин. Для этого нажмите кнопку выбора ... в окне пользовательских настроек в строке **Отбор**.

В открывшемся окне **Редактирование отбора** мы видим созданное нами ранее в конфигураторе условие отбора. Нажмите … в строке **Значение**, раскройте группу **Услуги**, выберите группу **Стиральные машины** из справочника **Номенклатура**.



Нажмите ОК. Т.о. мы задали отбор по услугам, родителем которых является группа **Стиральные машины** справочника **Номенклатура**.

Нажмите **Завершить редактирование** и сформируйте отчет.

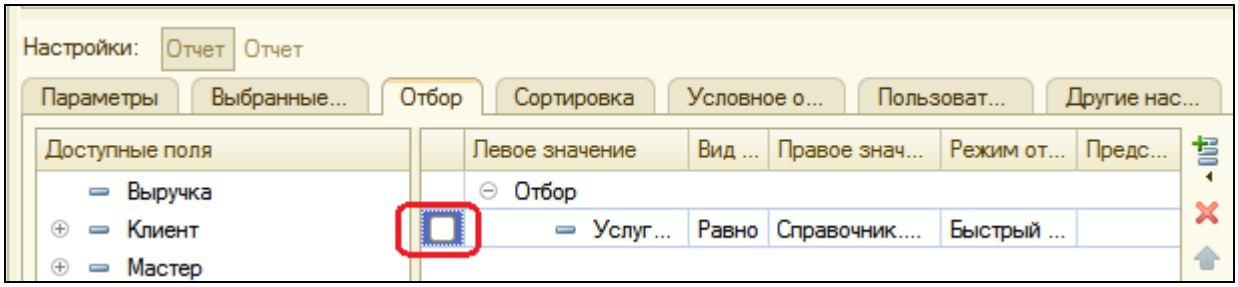


Видно, что в отчет включены только услуги по установке стиральных машин и в заголовке отчета отражена информация об отборе.

Вызвав окно настроек, мы можем очистить настройку отбора, нажав кнопку очистки  или создать ее по другому критерию, нажав кнопку выбора  в строке **Отбор**.

Таким образом, пользователь может, при наличии определенной квалификации, задавать многие настройки по своему желанию. Если же нет такого желания или способностей, лучше задавать эти настройки жестко, а пользователю останется включать или выключать их использование.

Вернитесь в Конфигуратор и снимите галочку использования у настройки отбора – это нам понадобится в дальнейшем.



Вывод данных по всем дням в выбранном порядке

Следующий отчет будет называться **Выручка мастеров** и будет содержать информацию о том, какая выручка была получена благодаря работе каждого из мастеров, с детализацией по всем дням в выбранном периоде и разворотом по клиентам, обслуженным в каждый из дней.

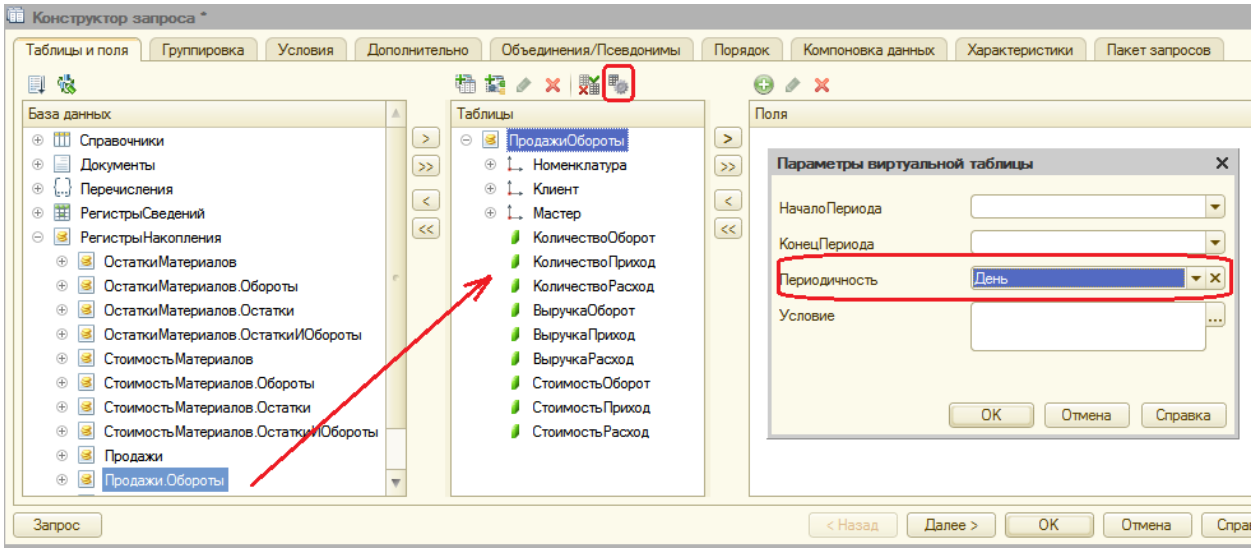
На примере этого отчета мы покажем, как строить многоуровневые группировки в запросе и как обходить все даты в выбранном периоде. Также покажем настройку отдельных элементов структуры отчета, научимся выводить данные в диаграмму и создавать несколько вариантов отчета в конфигураторе.

В конфигураторе добавьте новый отчет **ВыручкаМастеров** и запустите конструктор схемы компоновки данных. Добавьте новый **Набор данных – запрос** и вызовите конструктор запроса. Источник данных для запроса – виртуальная таблица регистра накопления Продажи.Обороты.

Запрос для набора данных

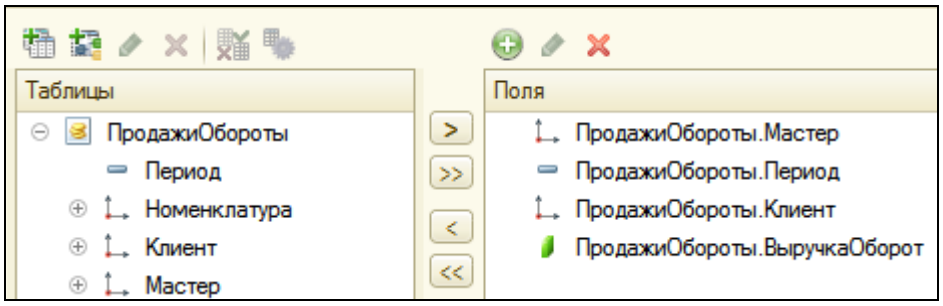
Параметры виртуальной таблицы

Задайте один из параметров этой виртуальной таблицы – **Периодичность**. Для этого перейдите в поле **Таблицы**, выделите таблицу и нажмите **Параметры виртуальной таблицы**. Задайте значение **Периодичность** – **День**. Нажмите ОК.

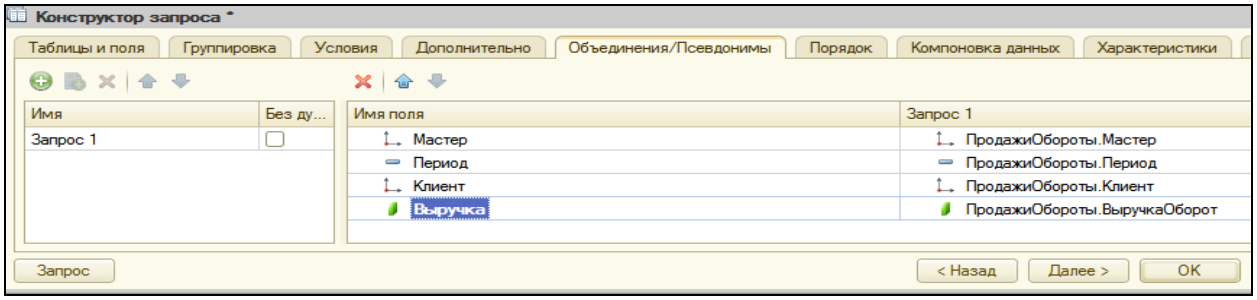


После этого выберите из таблицы поля:

- ПродажиОбороты.Мастер
- ПродажиОбороты.Период
- ПродажиОбороты.Клиент
- ПродажиОбороты.ВыручкаОборот



Перейдите на закладку **Объединения/Псевдонимы** и задайте псевдоним **Выручка** для поля **ПродажиОбороты.ВыручкаОборот**.



Анализ текста запроса

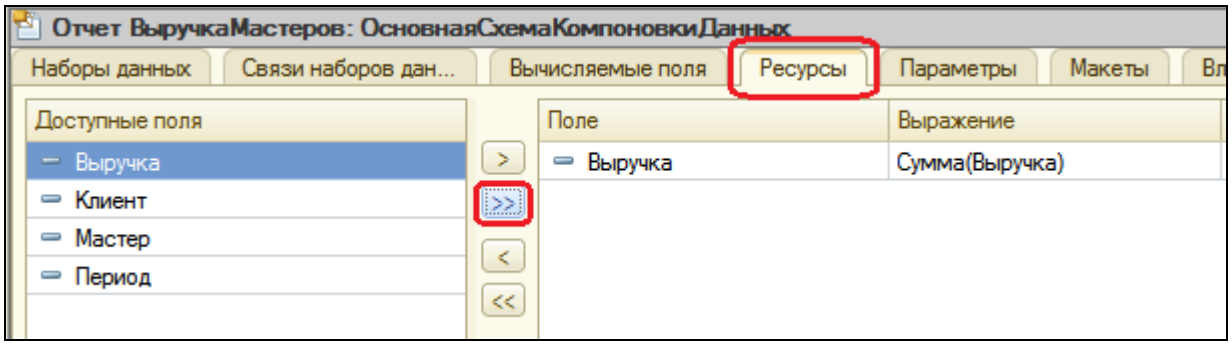
Нажмите ОК и проанализируйте текст запроса, сформированный конструктором.

```
ВЫБРАТЬ
	ПродажиОбороты.Мастер,
	ПродажиОбороты.Период,
	ПродажиОбороты.Клиент,
	ПродажиОбороты.ВыручкаОборот КАК Выручка
ИЗ
	РегистрНакопления.Продажи.Обороты(, , День, ) КАК ПродажиОбороты
```

В части описания запроса обратите внимание, что у источника данных задана периодичность выбираемых данных – День. Именно поэтому у нас появляется возможность описать среди выбранных полей поле **Период**.

Ресурсы

Перейдите на закладку **Ресурсы** схемы компоновки и перенесите единственный ресурс **Выручка**.



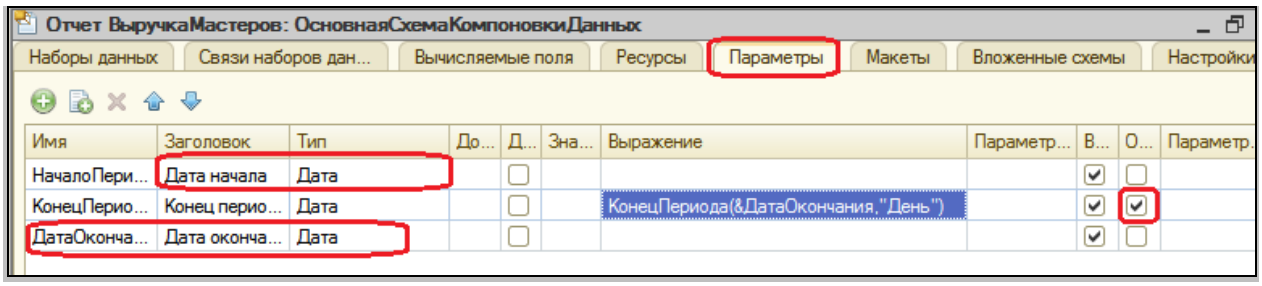
Параметры

Перейдите на закладку **Параметры** и для параметра **НачалоПериода** задайте заголовок **Дата** начала, тип и состав даты – **Дата**.

Добавьте еще один параметр – **ДатаОкончания**, тип **Дата**, состав даты - **Дата**.

Для параметра **КонецПериода** задайте выражение и в поле **Ограничение доступности** установите флажок.

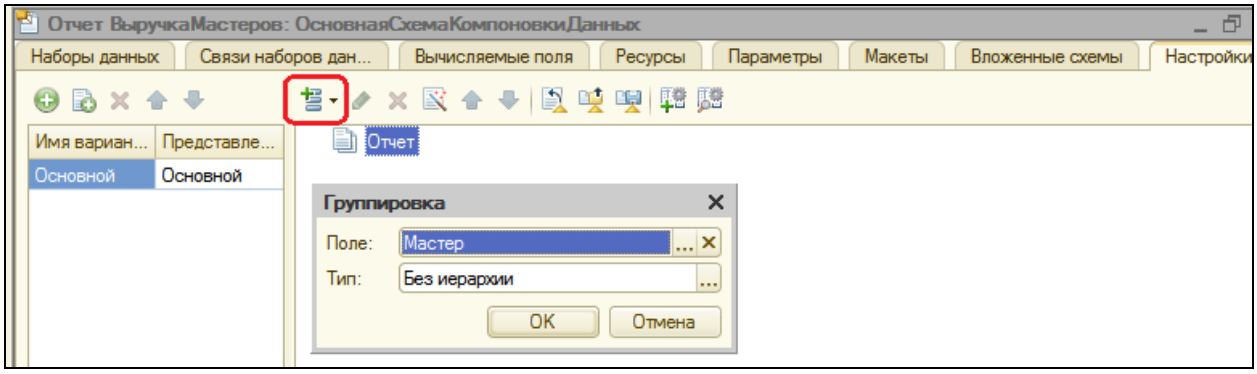
```
КонецПериода(&ДатаОкончания,"День")
```



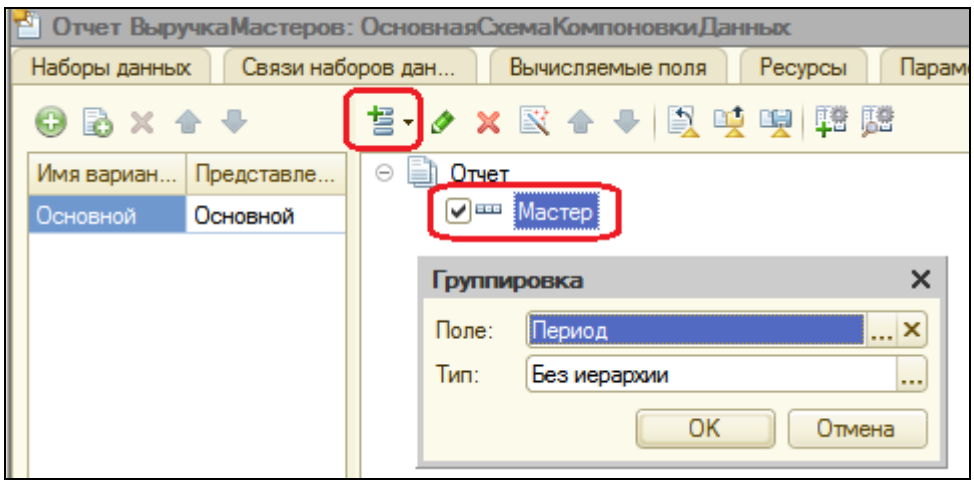
Настройки

Теперь создадим структуру отчета. На закладке **Настройки** последовательно создайте две вложенные группировки:

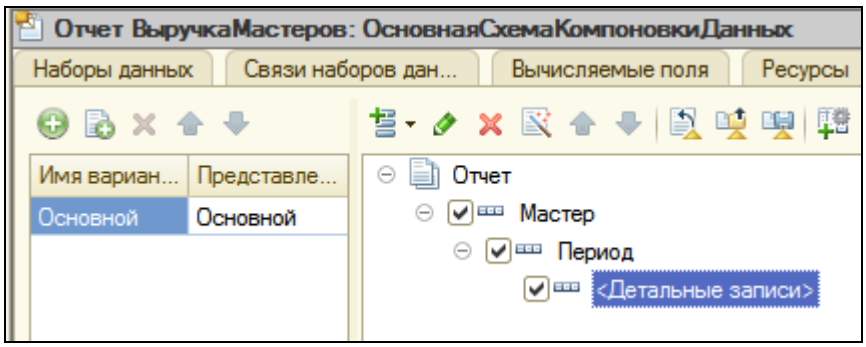
- Верхнего уровня – по полю **Мастер**
- Вложенная в нее – по полю **Период**



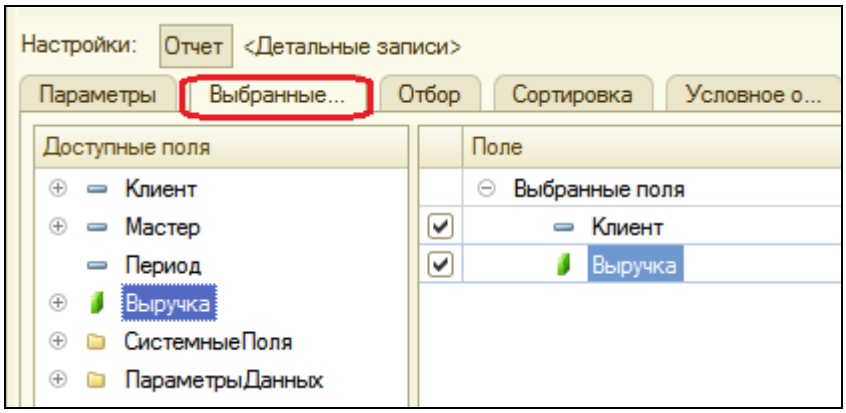
Затем добавьте в группировку **Мастер** вложенную группировку по полю **Период**. Для этого выделите группировку **Мастер** и нажмите **Добавить**.



Затем добавим еще одну группировку, вложенную в группировку по полю **Период**, - **Детальные записи** (без указания поля группировки).



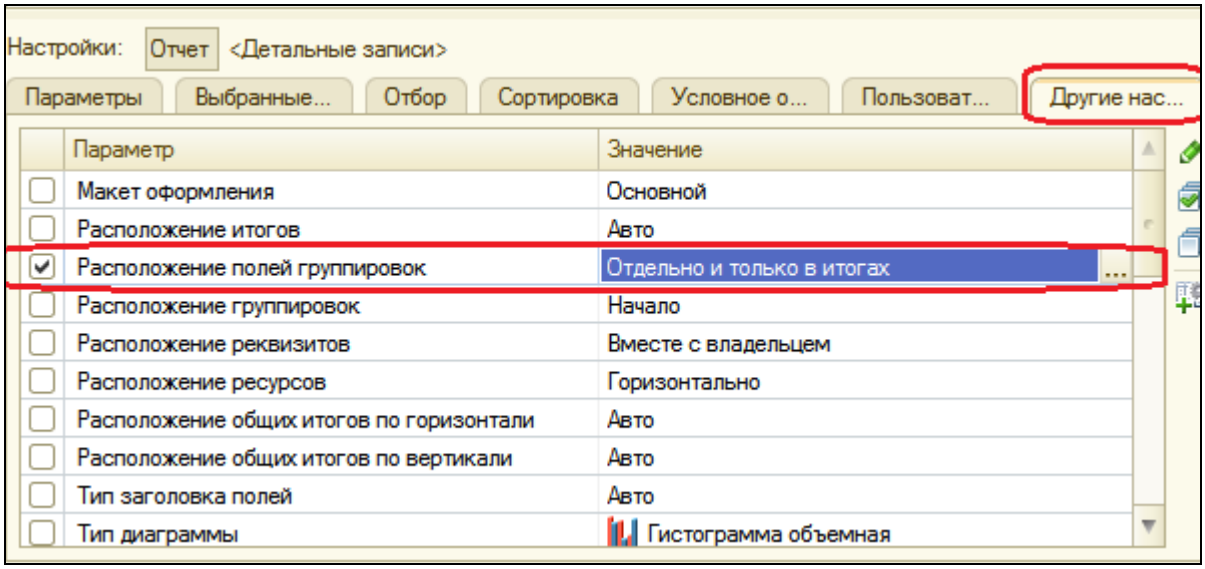
После перейдите на вкладку **Выбранные поля** и добавьте в список выбранных полей **Клиент** и **Выручка**.



Поля **Мастер** и **Период** не задаем, т.к. по ним идет группировка, и они будут выведены автоматически.

Перейдите на закладку **Другие настройки** и измените следующие параметры:

1. **Расположение полей группировок** – **Отдельно и только в итогах**.



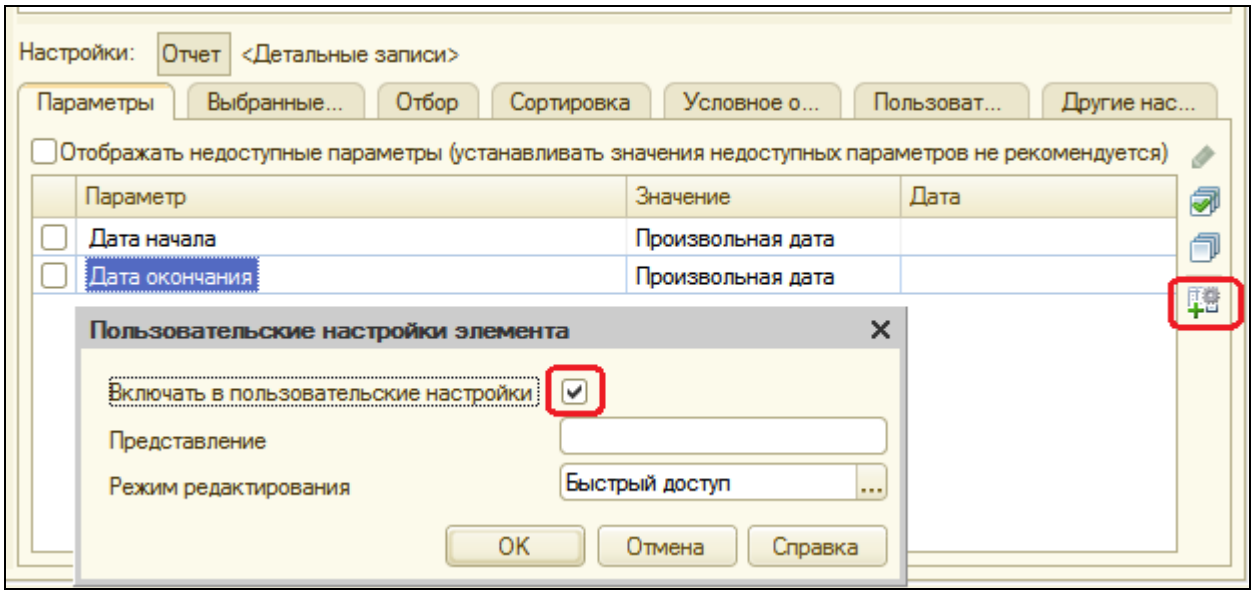
Установка этого свойства означает, что каждая группировка будет располагаться в отдельной области отчета слева направо и ее наименование будет выводиться только в данной группировке.

2. **Расположение общих итогов по вертикали** – **Начало**.

Установка этого свойства означает, что общие итоги будут отображаться в начале перед строками группировки.

3. **Заголовок** – **Выручка мастеров**.

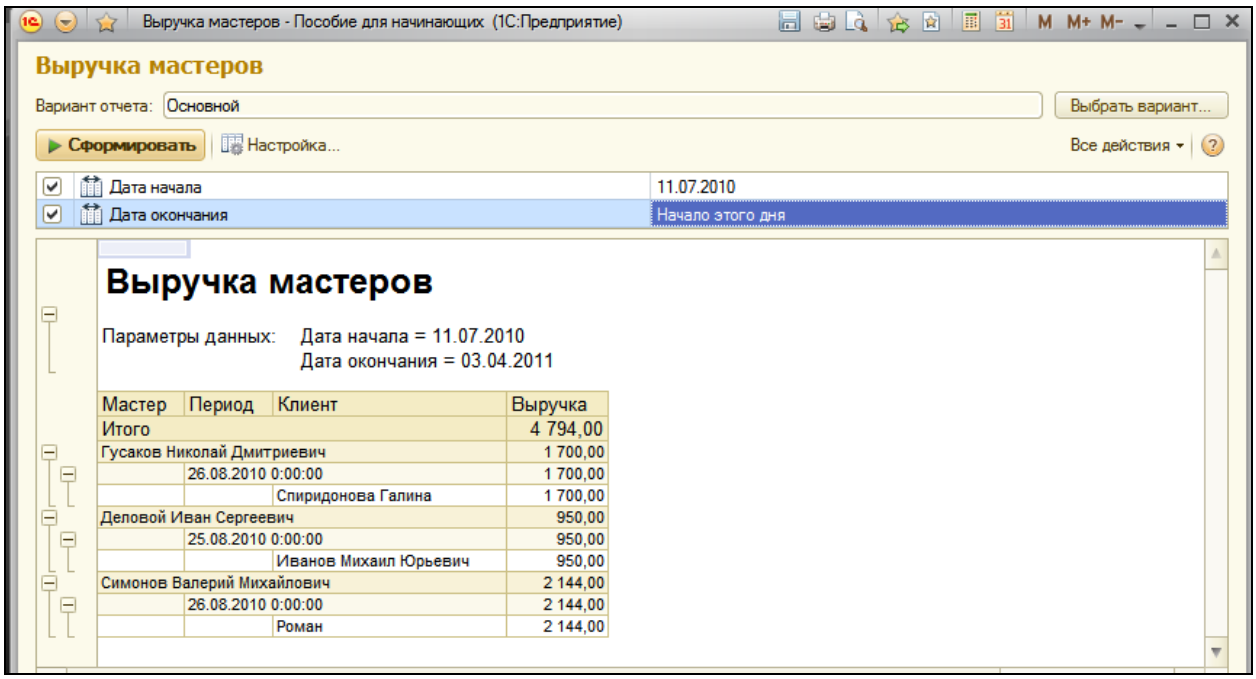
Затем укажите, что параметры **Дата начала** и **Дата окончания** будут включены в состав пользовательских настроек и эти настройки будут находиться в форме (быстрые). Т.о. пользователь перед формированием запроса сможет указать отчетный период.



Определим, в каких подсистемах будет отображаться наш отчет. Закройте конструктор и перейдите на закладку **Подсистемы**, отметьте **ОказаниеУслуг** и **РасчетЗарплаты**.

В режиме 1С: Предприятие

Запустите отладку. В панели действий разделов **Оказание услуг** и **Расчет зарплаты** в группе команд для выполнения отчетов появилась команда для формирования отчета **Выручка мастеров**. Выполните этот отчет, задав отчетный период.



Вывод всех дат в выбранном периоде

Этот отчет должен показывать данные с детализацией по всем дням в выбранном периоде. У нас же отображаются только те дни, для которых существуют ненулевые записи в таблице регистра накопления **Продажи**.

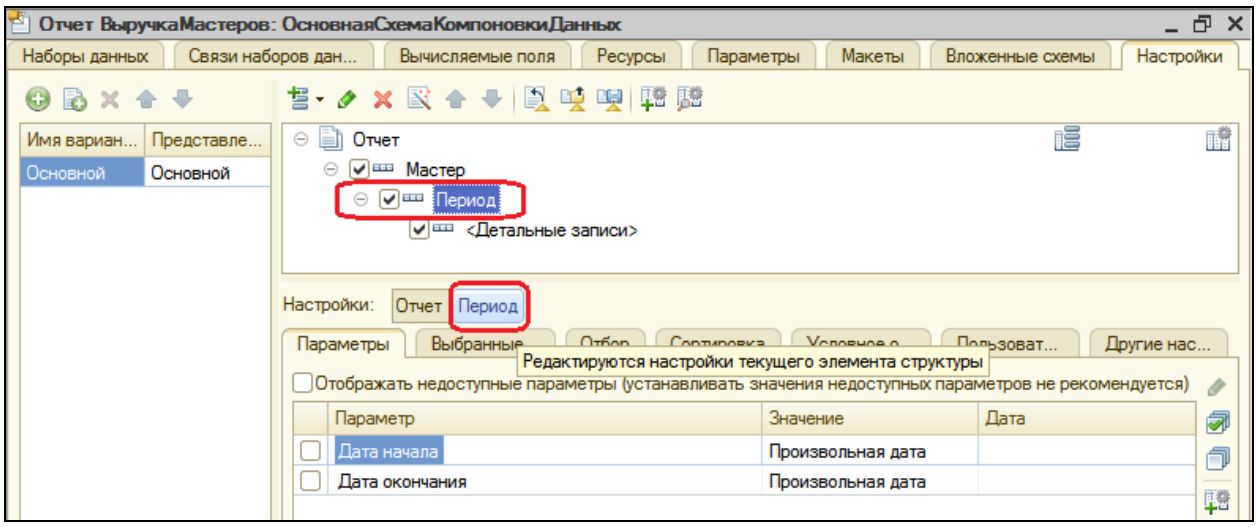
Для детализации данных в отчете система компоновки позволяет указывать для группировок *дополнение* периодов с заданной периодичностью в указанном интервале. Поэтому сейчас мы изменим настройки отчета, чтобы в отчет попадала каждая дата из периода.

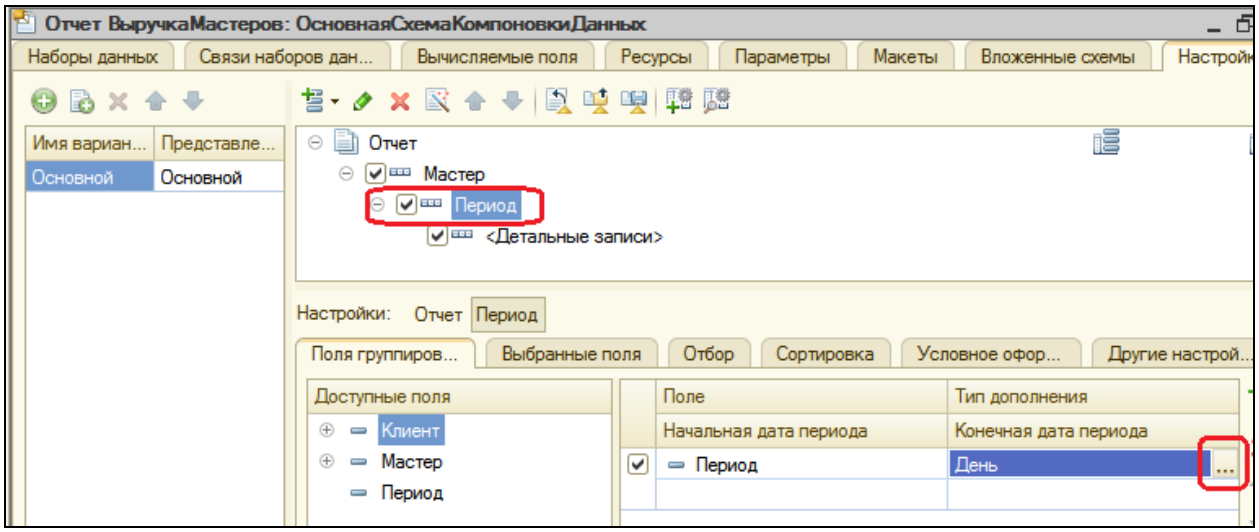
В режиме Конфигуратор

Вернитесь в конфигуратор и откройте схему компоновки на закладке **Настройки**. До сих пор мы выполняли настройку всего отчета в целом. Но можно настраивать и отдельные его элементы.

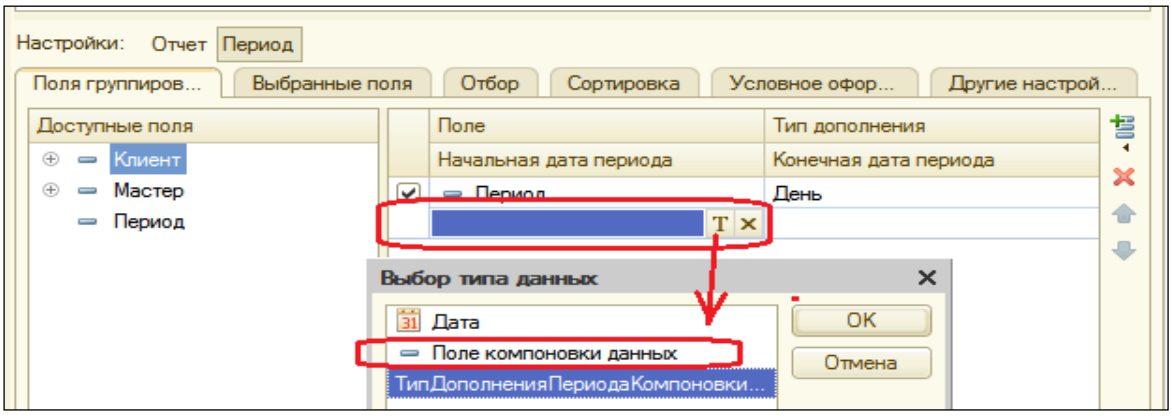
Выделите группировку **Период** в верхней части окна и нажмите кнопку **Период** в командной панели (рядом с Отчет).

Перейдите на закладку **Поля группировки**. Для поля **Период** установите **Тип дополнения** – **День**. Тем самым укажем, что для этой группировки существующие записи с ненулевым значением ресурса будут дополняться записями для каждого из дней.

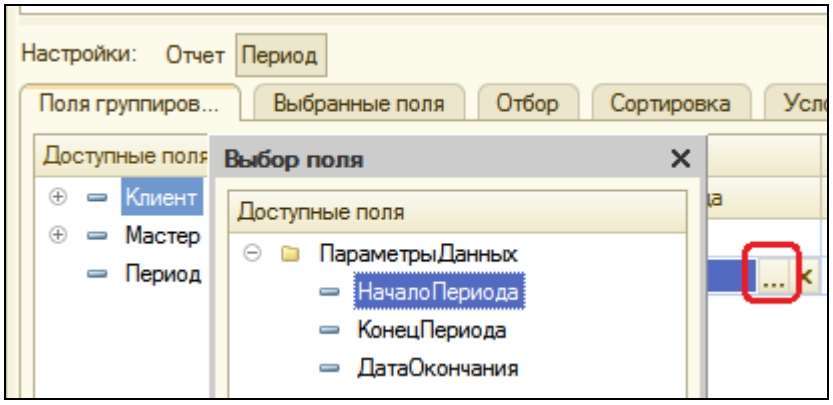




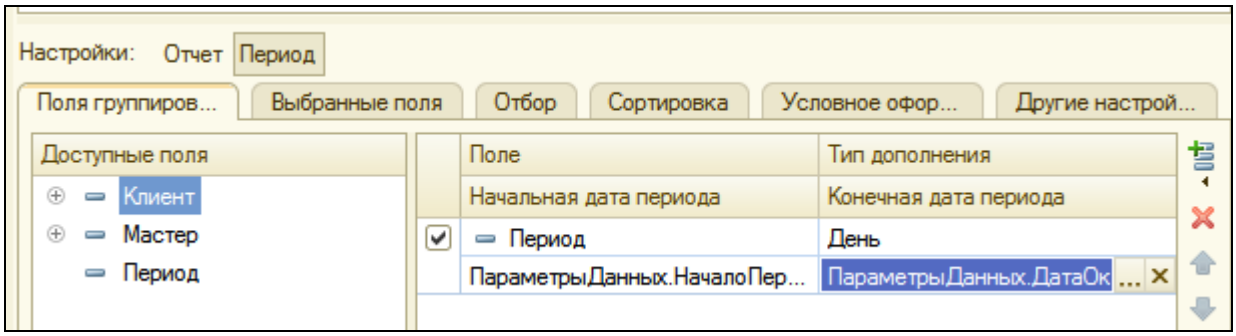
После этого нужно указать, в каком именно периоде будет выполняться такое дополнение. Но нам нужно, чтобы пользователь мог задать произвольную дату. Для этого зайдите в режим редактирования поля **Начальная дата периода**, дважды щелкнув на нем, и нажмите кнопку очистки ✕. Затем нажмите кнопку выбора данных T, выберите тип данных **Поле компоновки данных**.



Нажмите ОК. Теперь нажмите кнопку выбора [...] и отметьте параметр **НачалоПериода**. Нажмите ОК.

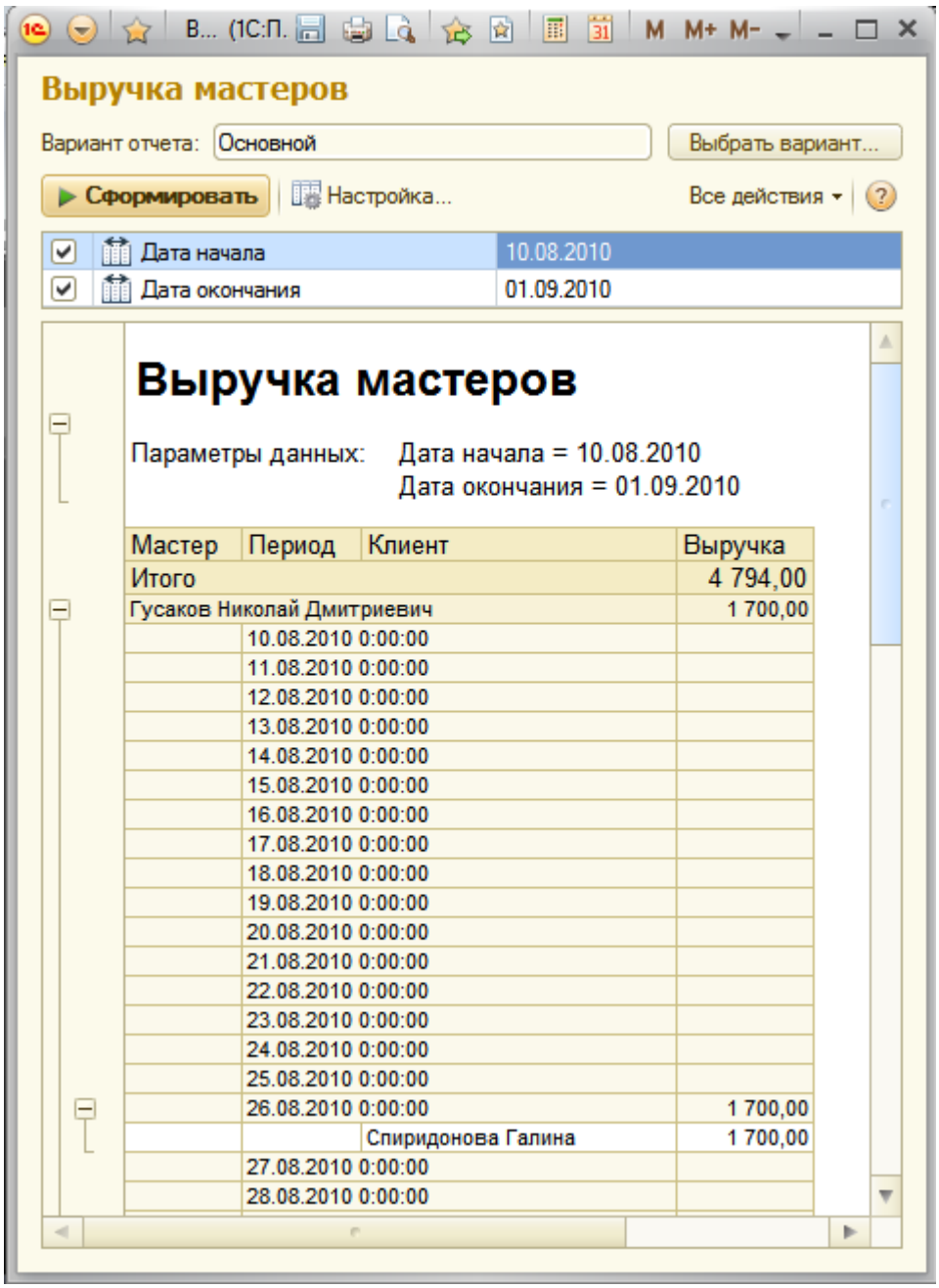


Для второго периода аналогично укажите, что дата окончания периода будет получена из параметра **ДатаОкончания**.



В режиме 1С: Предприятия

Запустите режим отладки и сделайте отчет Выручка мастеров за период выполнения ваших работ.



Новый вариант отчета

Для анализа работы мастеров за определенный период может понадобиться представить информацию в более наглядном виде. Например, в виде диаграммы, отражающей вклад каждого мастера в общую выручку предприятия. Поэтому мы создадим другой вариант отчета **ВыручкаМастеров** в виде диаграммы.

Диаграмма является совокупностью точек, серий и значений серий в точке. В качестве *точек* используются моменты или объекты, для которых мы получаем значения характеристик, а в качестве *серий* – характеристики, значения которых нас интересуют. На пересечении

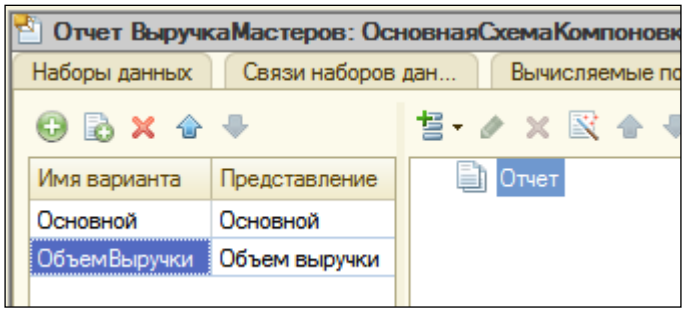
серии и точки находится *значение* диаграммы. Например, диаграмма продаж видов номенклатуры по месяцам будет состоять из *точек* – месяцев, *серий* – видов номенклатуры и *значений* – оборотов продаж.

В режиме Конфигуратор

Вернитесь в Конфигуратор и откройте схему компоновки на закладке **Настройки**. В левой части окна находится список вариантов отчета.

При создании настроек отчета в первый раз система компоновки по умолчанию создает **Основной** вариант настроек.

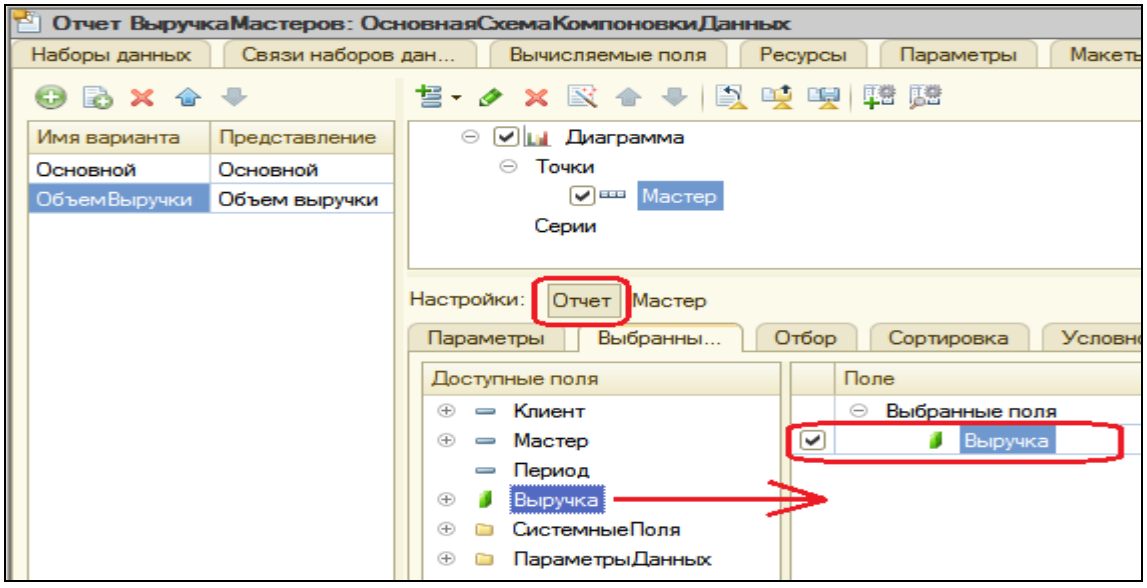
Чтобы добавить новый вариант, нажмите кнопку **Добавить** над этим списком. Задайте имя варианта – **ОбъемВыручки**.



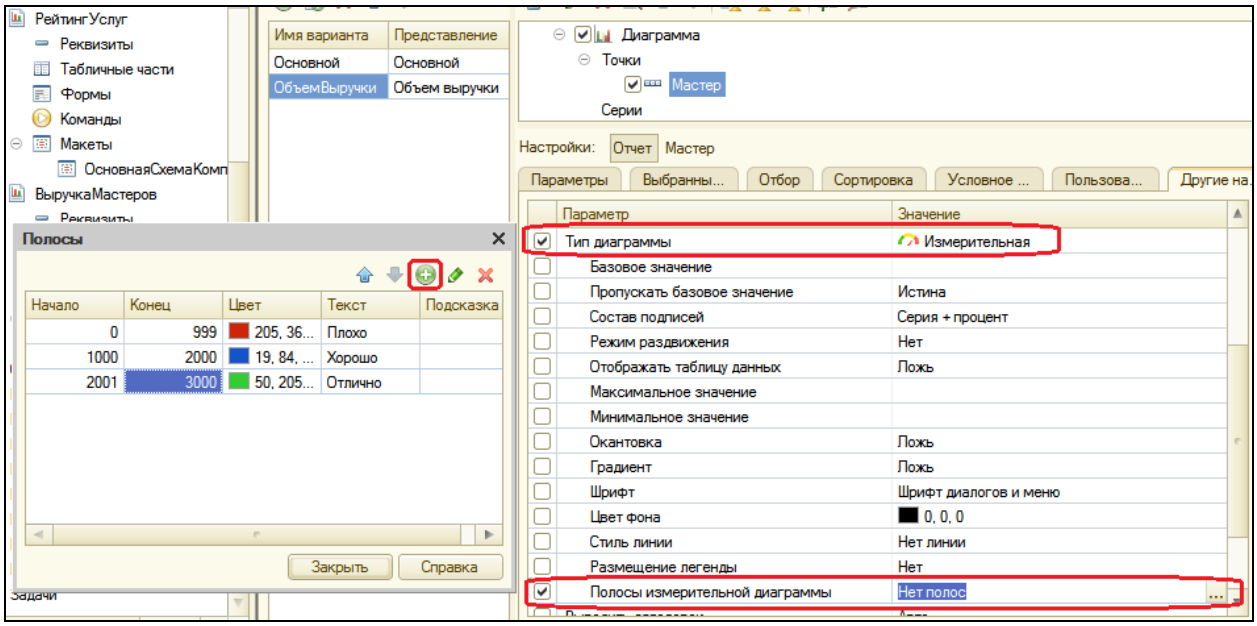
Если у отчета несколько вариантов, то мы видим и можем изменять настройки только выделенного варианта.

Добавим в структуру отчета диаграмму с помощью контекстного меню элемента **Отчет**. Затем выделите ветку **Точки** и добавьте в нее группировку по полю **Мастер**. **Серии** оставим без изменений.

В значения диаграммы всегда выводится один из ресурсов отчета. У нас всего один ресурс – **Выручка** (поле ресурса помечено особой пиктограммой). Поэтому перейдите на закладку **Выбранные поля**, нажмите кнопку **Отчет** и выберите поле **Выручка** для вывода в отчет.



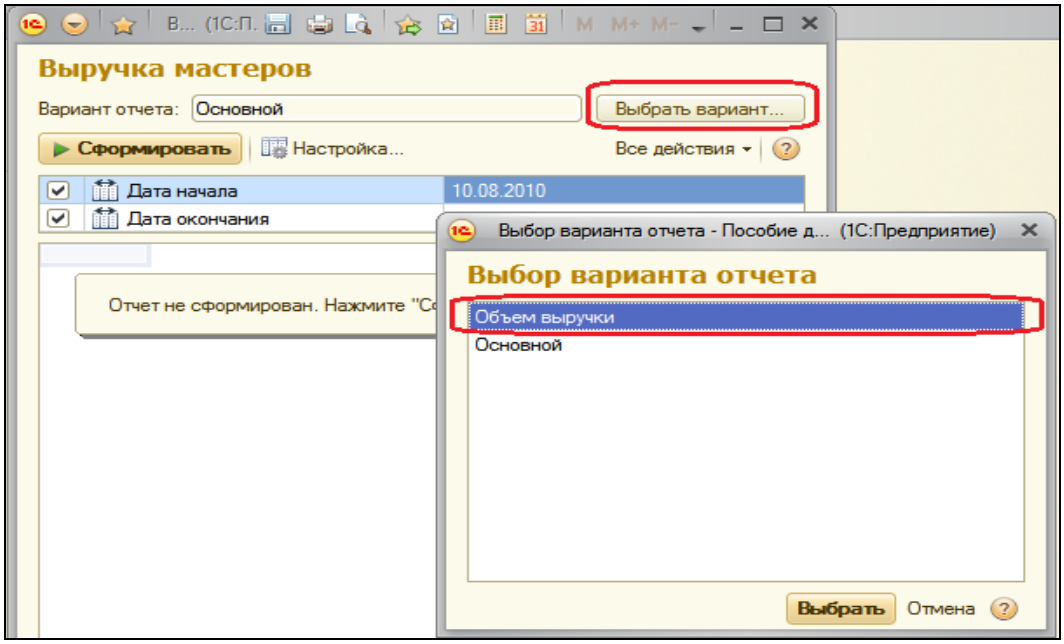
На закладке **Другие настройки** выберите тип диаграммы – **Измерительная**. Прокрутив список свойств диаграммы, задайте ее полосы – **Плохо**, **Хорошо**, **Отлично**.



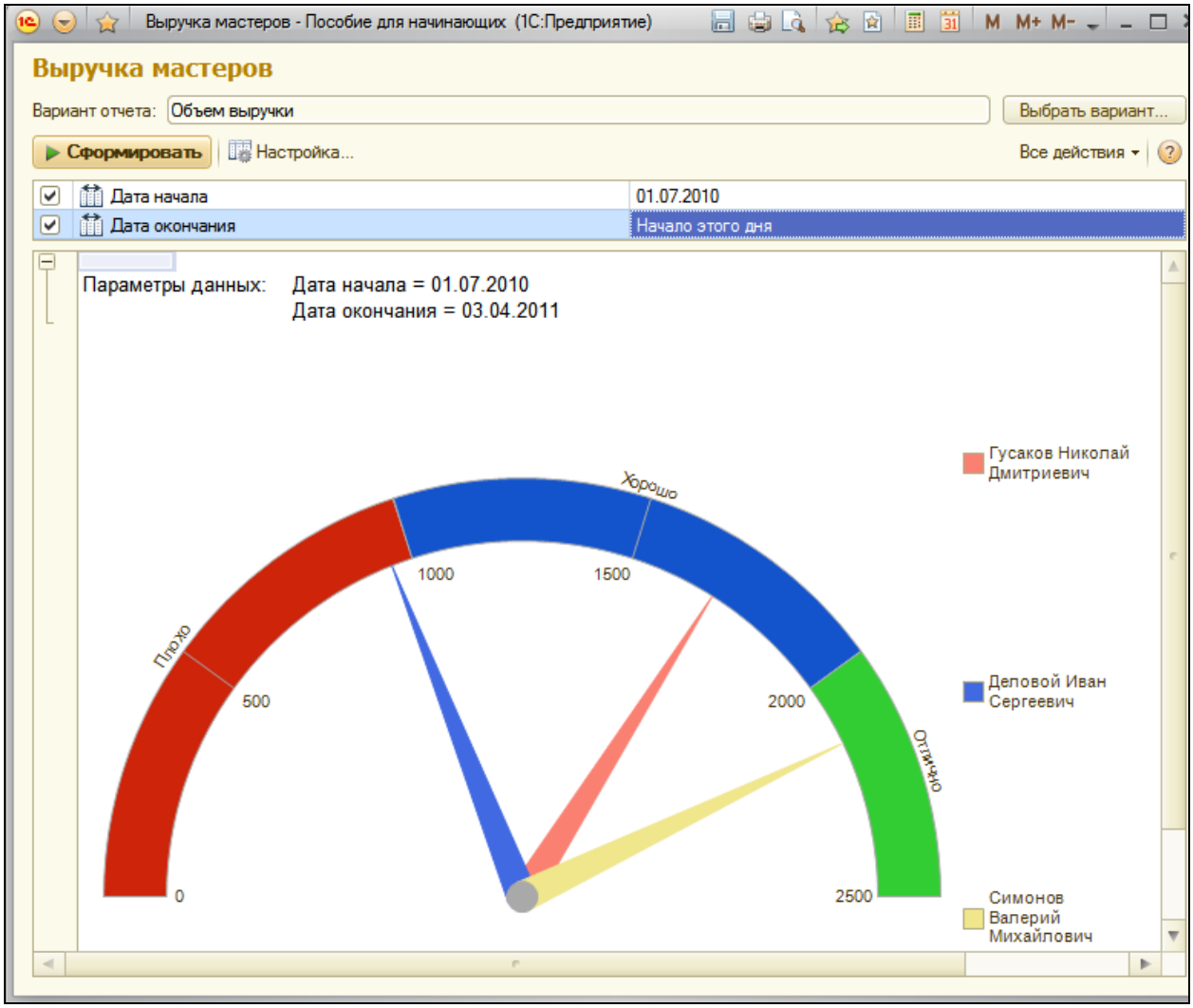
Включите параметры **Дата начала** и **Дата окончания** в состав быстрых пользовательских настроек (на вкладке **Параметры**). Учтите, что состав пользовательских настроек для каждого варианта отчета – свой.

В режиме 1С: Предприятие

Запустите отладку и выполните **Выручка мастеров** в панели действий **Расчет зарплаты**. Нажмите кнопку **Выбрать вариант**, выделите **Объем выручки** и нажмите кнопку **Выбрать**.



Задайте отчетный период и сформируйте отчет.



Если же понадобится просмотреть данные о работе какого-либо мастера с разбивкой по дням и клиентам, достаточно выбрать **Основной** вариант отчета и переформировать его.

Т.о. на примере отчета **Выручка мастеров** мы показали создание и использование различных вариантов отчета в целях наилучшего представления информации о работе мастеров.

Получение актуальных значений из периодического регистра сведений

Следующий отчет – **Перечень услуг** – будет содержать информацию о том, какие услуги и по какой цене оказывает наша фирма.

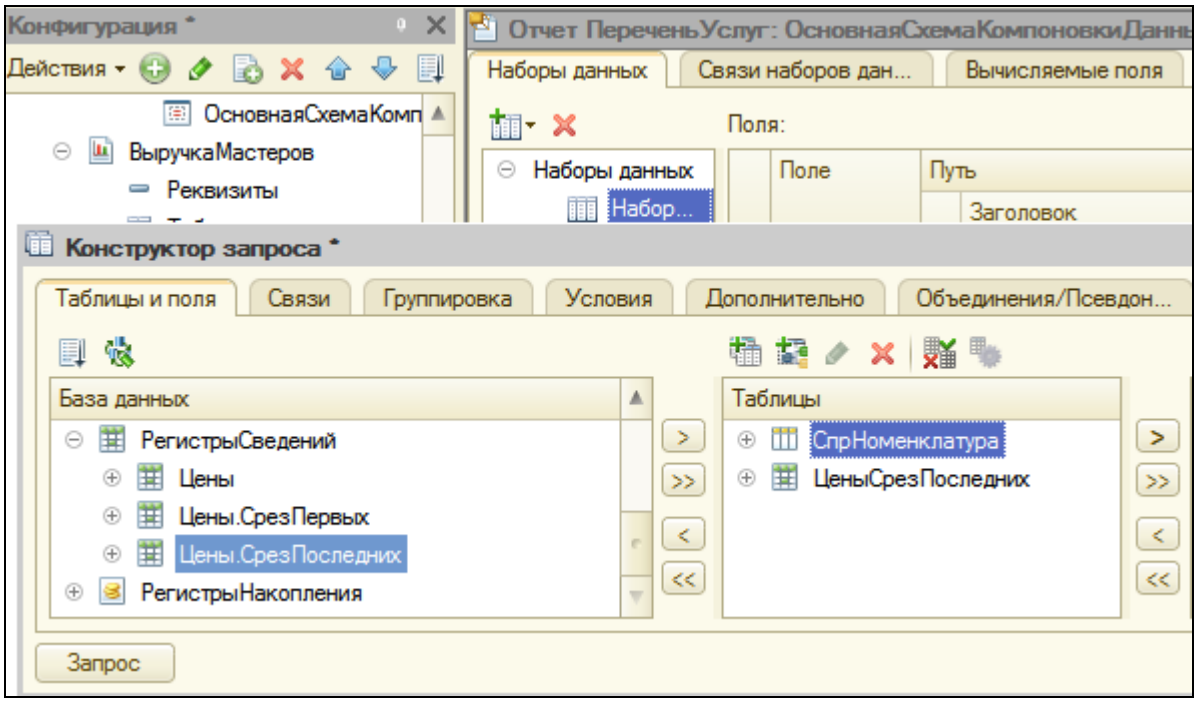
На его примере мы познакомимся с возможностью получения последних значений из периодического регистра сведений и с возможностью вывода иерархических справочников.

В режиме Конфигуратор

Добавьте новый отчет **ПереченьУслуг** и запустите конструктор схемы компоновки данных. Добавьте новый **Набор данных – запрос** и вызовите конструктор запроса.

Запрос для набора данных

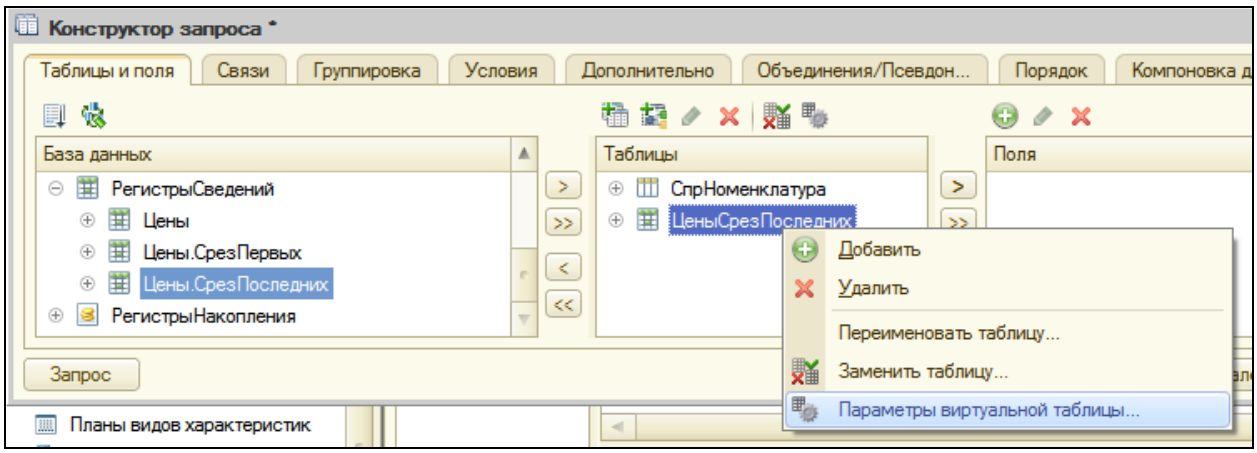
В качестве источника данных для запроса выберете объектную (ссылочную) таблицу справочника **Номенклатура** и виртуальную таблицу регистра сведений **Цены.СрезПоследних**. Переименуйте таблицу Номенклатура в СпрНоменклатура.

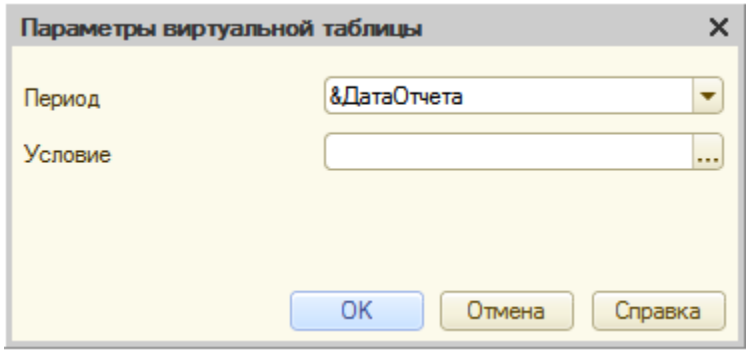


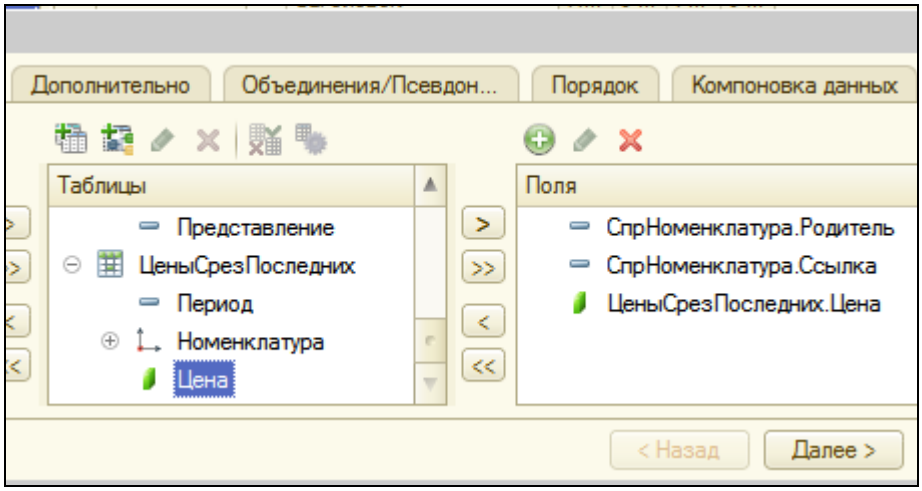
Параметры виртуальной таблицы

Вызовите диалог ввода параметров виртуальной таблицы **ЦеныСрезПоследних** и укажите, что период будет передан в параметре **ДатаОтчета**. Затем выберете следующие поля из таблиц:

- СпрНоменклатура.Родитель
- СпрНоменклатура.Ссылка
- ЦеныСрезПоследних.Цена

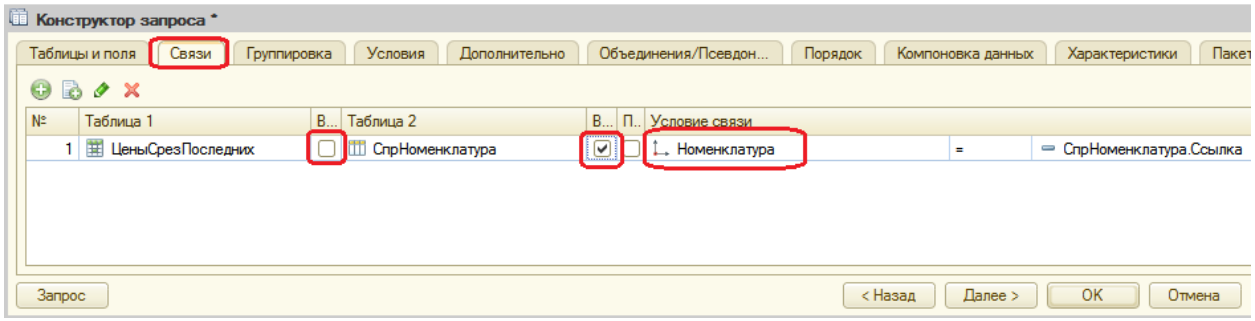




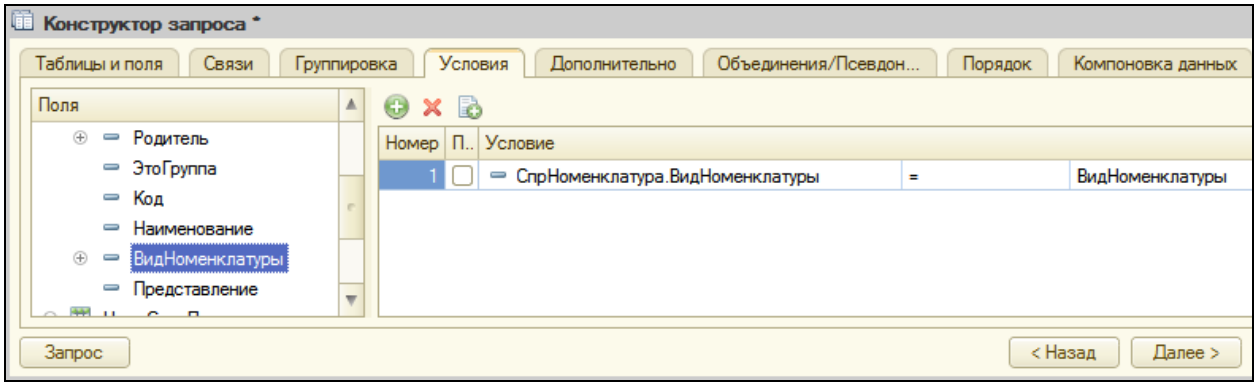


Левое соединение таблиц

Перейдите на закладку **Связи** и укажите в поле **Условие связи**, что значение измерения **Номенклатура** регистра сведений должно быть равно ссылке на элемент справочника **Номенклатура**. А также снимите флажок **Все** у таблицы регистра и установите его у таблицы справочника, тем самым установив вид связи как *левое соединение* для таблицы справочника.

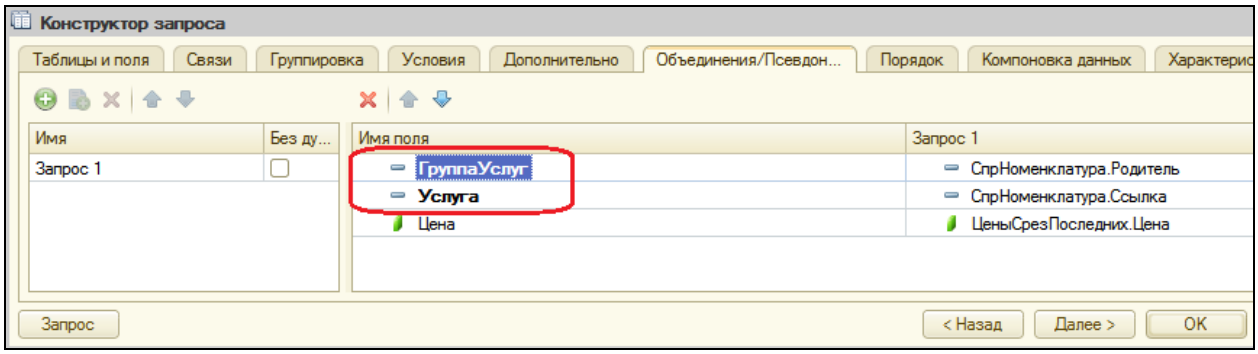


На закладке **Условия** задайте условие выбора элементов справочника **Номенклатура** – выбираемые элементы должны соответствовать виду номенклатуры, переданному в параметре запроса **Вид Номенклатуры**.



Псевдонимы полей

На закладке **Объединения/Псевдонимы** укажите, что поле **Родитель** будет иметь псевдоним **ГруппаУслуг**, а поле **Ссылка** – **Услуга**. Нажмите ОК.



Анализ текста запроса

Рассмотрим текст запроса, сформированного конструктором.

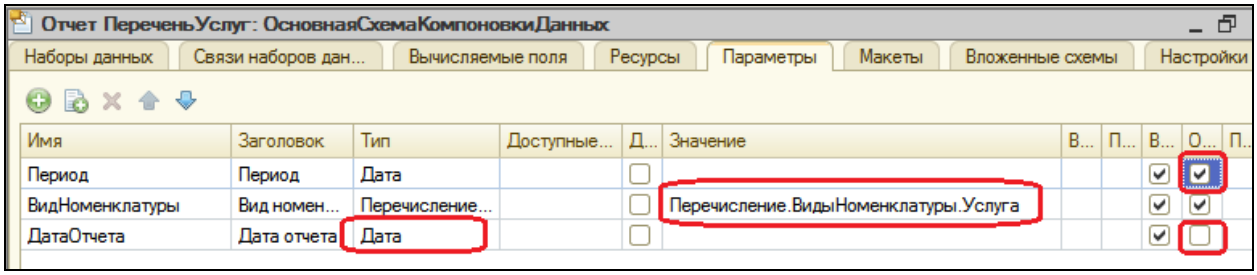
```
ВЫБРАТЬ
	СпрНоменклатура.Родитель КАК ГруппаУслуг,
	СпрНоменклатура.Ссылка КАК Услуга,
	ЦеныСрезПоследних.Цена
ИЗ
	Справочник.Номенклатура КАК СпрНоменклатура
		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних(&ДатаОтчета,
) КАК ЦеныСрезПоследних
		ПО ЦеныСрезПоследних.Номенклатура = СпрНоменклатура.Ссылка
ГДЕ
	СпрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры
```

Перейдем к редактированию схемы компоновки данных. На закладке **Ресурсы** выберем единственный доступный ресурс – **Цена**.

Параметры

На закладке **Параметры** задайте значения параметра **ВидНоменклатуры** как **Перечисление.ВидыНоменклатуры.Услуга**. Снимите ограничение доступности для параметра **ДатаОтчета**. В поле **Тип** этого параметра задайте состав даты – **Дата**.

Для параметра **Период**, наоборот, установите ограничение доступности.

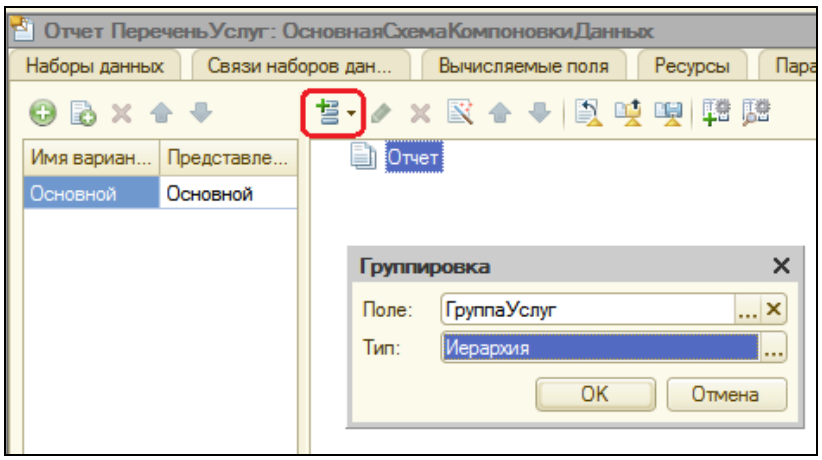


Настройки

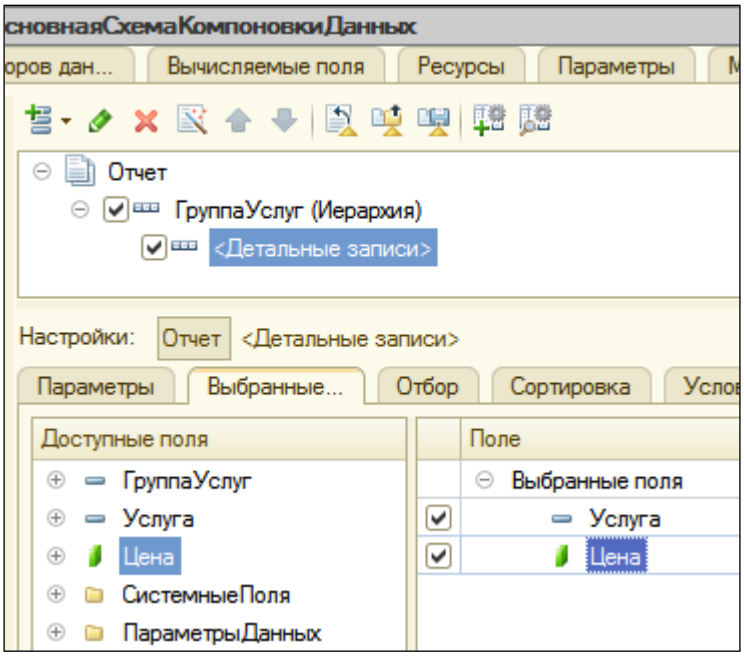
Перейдите на закладку **Настройки** и создайте группировку по полю **ГруппаУслуг** с типом группировки **Иерархия**.

Существуют следующие типы иерархии для группировки отчета:

- Без иерархии – в группировке выводятся только неиерархические записи.
- Иерархия – выводятся неиерархические и иерархические записи.
- Только иерархия – выводятся только иерархические (родительские) записи.



Внутри этой группировки создайте еще одну без указания группового поля. Она будет содержать детальные записи отчета.

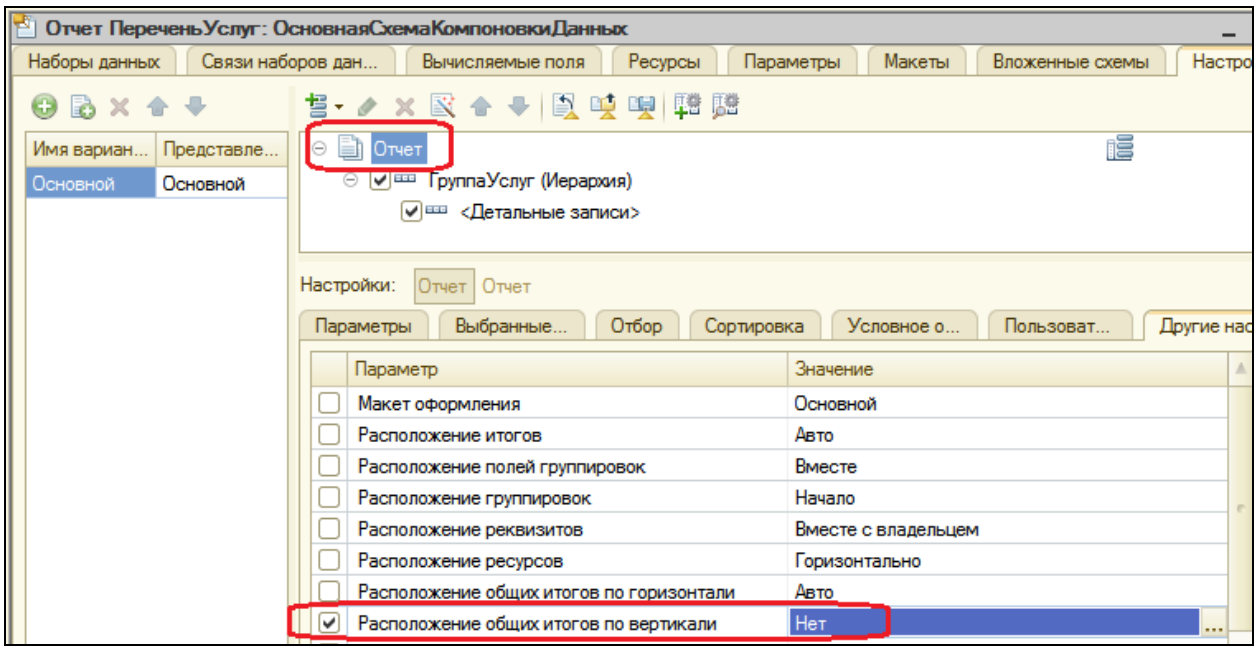


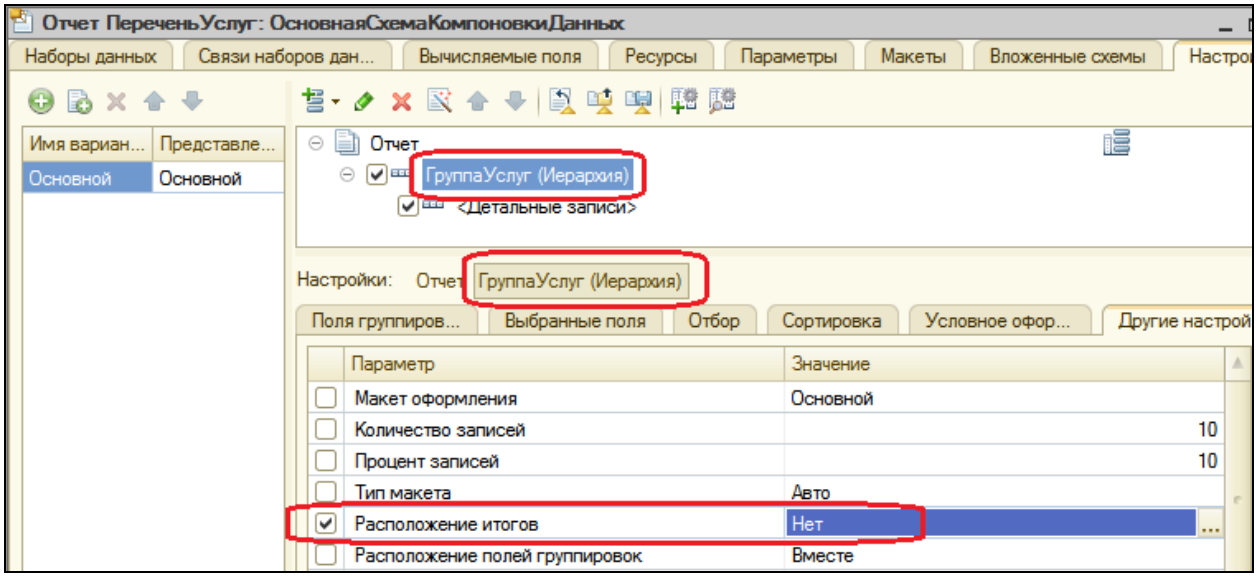
Перейдите на закладку внизу **Выбранные поля** и укажите поля **Услуга** и **Цена**.

Теперь перейдем на закладку Другие настройки и определим внешний вид отчета.

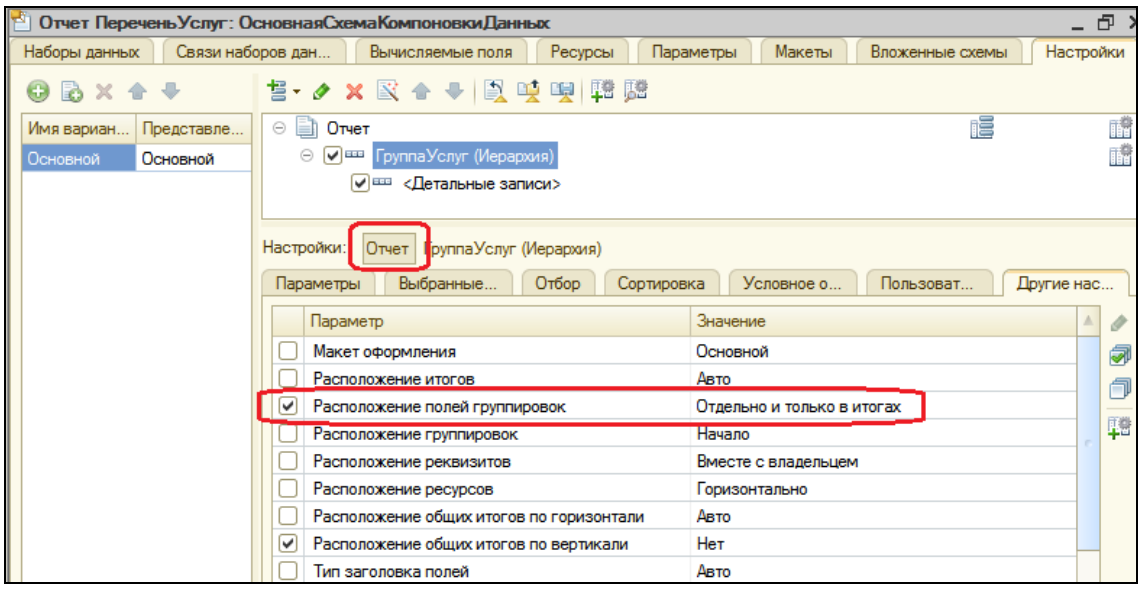
Т.к. наш отчет будет представлять собой просто список оказываемых услуг, в котором интересны цены на конкретные услуги, выводить значения ресурса **Цена** для каждой из группировок и для всего отчета не имеет смысла. Чтобы запретить вывод общих итогов в отчете, установите параметр Отчета **Расположение общих итогов по вертикали** в значение **Нет**.

Затем перейдите к настройкам конкретной группировки – **ГруппаУслуг**. Для параметра **РасположениеИтогов** этой группировки укажите **Нет**.





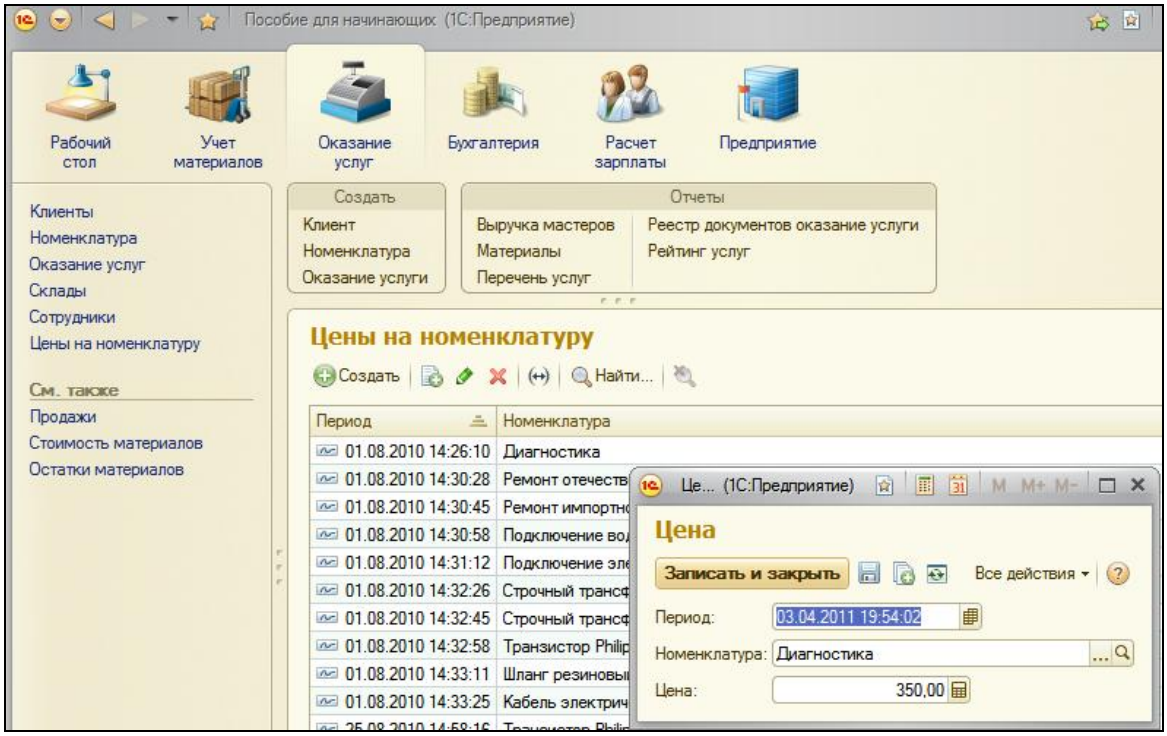
Вернитесь к настройкам отчета в целом. Для параметра **Расположение полей группировок** укажите значение **Отдельно и только в итогах** (так наш отчет будет лучше читаться). Задайте заголовок отчета – **Перечень услуг**.



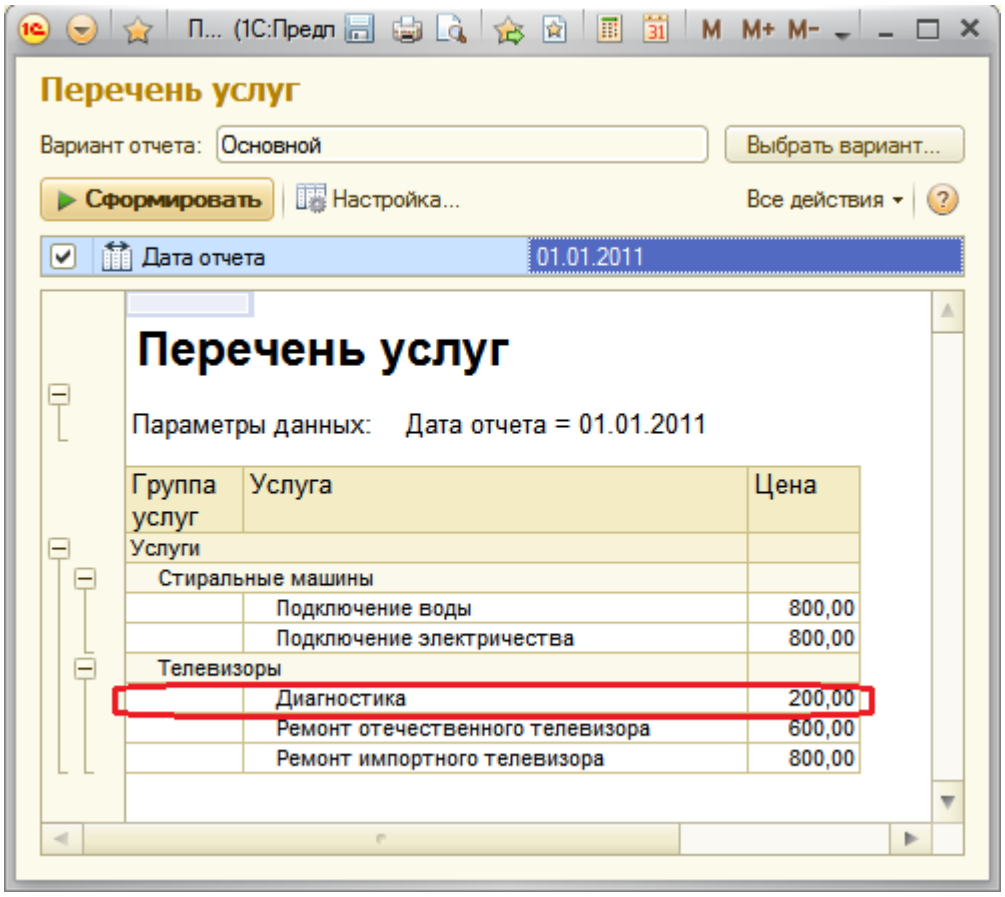
Включите параметр **Дата отчета** в состав пользовательских быстрых настроек (вкладка Параметры в нижнем окне). Также определите, в каких подсистемах будет отображаться наш отчет – **Оказание услуг** и **Бухгалтерия**.

В режиме 1С: Предприятие

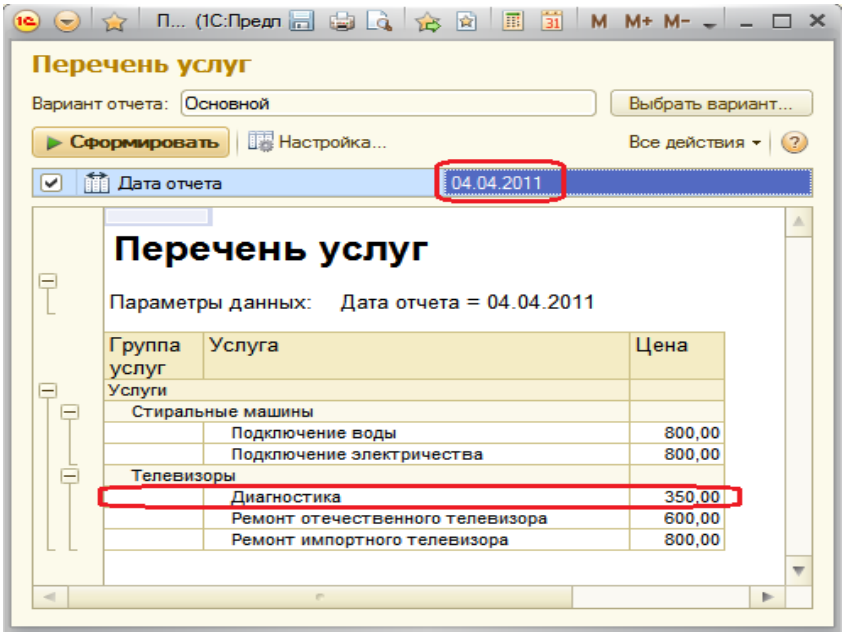
Запустите отладку и откройте периодический регистр **Цены на номенклатуру**. Добавьте в него еще одно значение для услуги **Диагностика**: новая цена услуги на текущее число – **350**. Это позволит нам протестировать отчет.



Теперь выполним отчет **Перечень услуг** по состоянию на пару дней раньше текущей даты (у меня это 01.04.2011).



Наш отчет правильно отражает цену услуги Диагностика на прошлые числа – 200р. Еще раз сформируйте отчет, но уже на текущую дату (или на день позже).



Как видите, показана новая цена услуги Диагностика – 350 р.

На примере этого отчета было показано, как система компоновки данных получает последние значения из периодического регистра сведений и как вывести группировки по иерархии справочника.

Использование вычисляемого поля в отчете

Следующий отчет – **Рейтинг клиентов** – будет показывать в графическом виде доход от оказания услуг каждому из клиентов за все время работы фирмы. На его примере мы продемонстрируем возможность использования вычисляемого поля и вывод результата в виде круговой диаграммы и гистограммы.

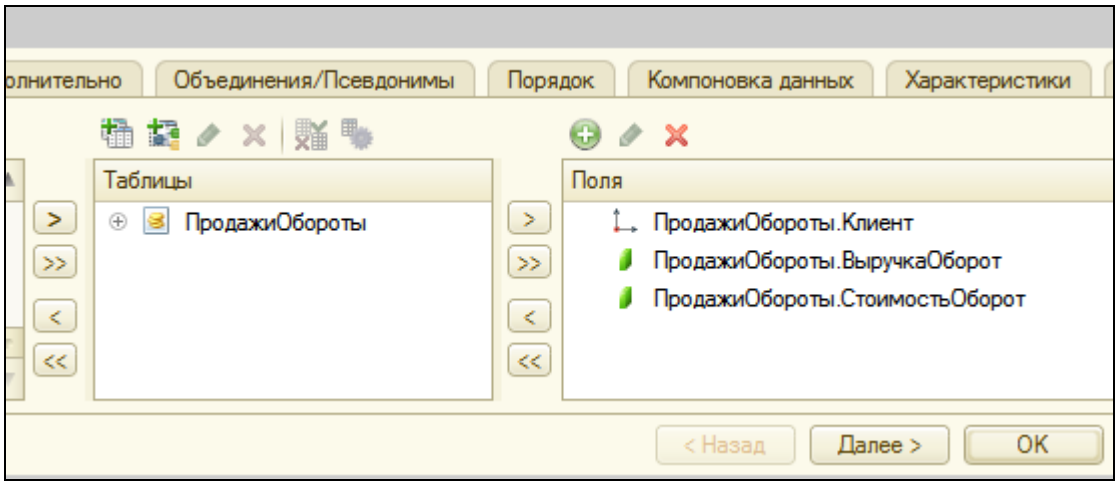
В режиме Конфигуратор

Добавьте новый отчет **РейтингКлиентов** и запустите конструктор схемы компоновки. Создайте новый Набор данных – запрос и вызовите конструктор запроса.

Запрос для набора данных

В качестве источника данных для запроса выберем виртуальную таблицу регистра накопления **Продажи.Обороты**. Затем выберем из нее следующие поля:

- ПродажиОбороты.Клиент
- ПродажиОбороты.ВыручкаОборот
- ПродажиОбороты.СтоимостьОборот



На закладке **Объединения/Псевдонимы** укажите, что поле **ВыручкаОборот** будет иметь псевдоним Выручка, а поле **СтоимостьОборот** – **Стоимость**. Нажмите ОК.

Ничего нового и непонятного в запросе нет, поэтому рассматривать его не будем. Перейдем к редактированию схемы компоновки данных.

Вычисляемые поля

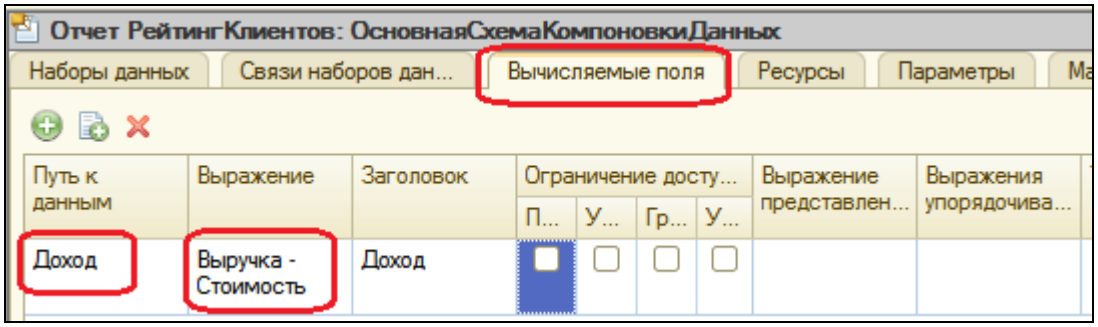
Теперь нам нужно создать дополнительное поле для отображения дохода от оказания услуг в разрезе клиентов. Для этого в системе компоновки есть возможность определения *вычисляемого поля*.

Вычисляемое поле – дополнительное поле схемы компоновки, значения которых будут вычисляться по некоторой формуле.

Перейдите на закладку **Вычисляемые поля** и нажмите **Добавить**.

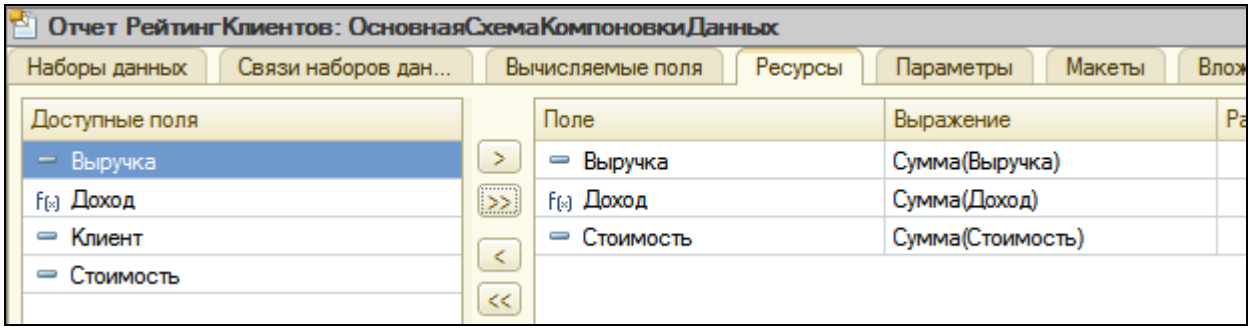
Задайте имя в колонке **Путь к данным** – **Доход**. В колонку **Выражение** введите следующее выражение:

```
Выручка – Стоимость
```



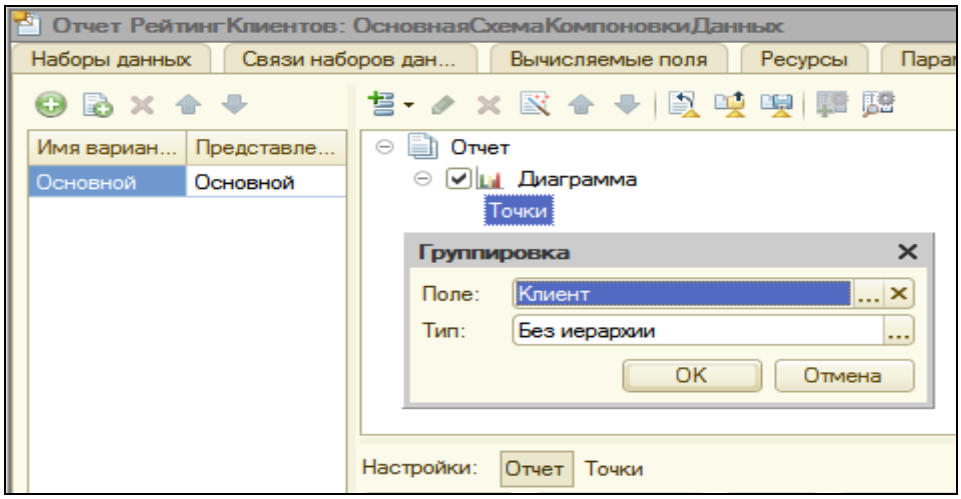
Ресурсы

На закладке **Ресурсы** выберите все доступные ресурсы отчета.

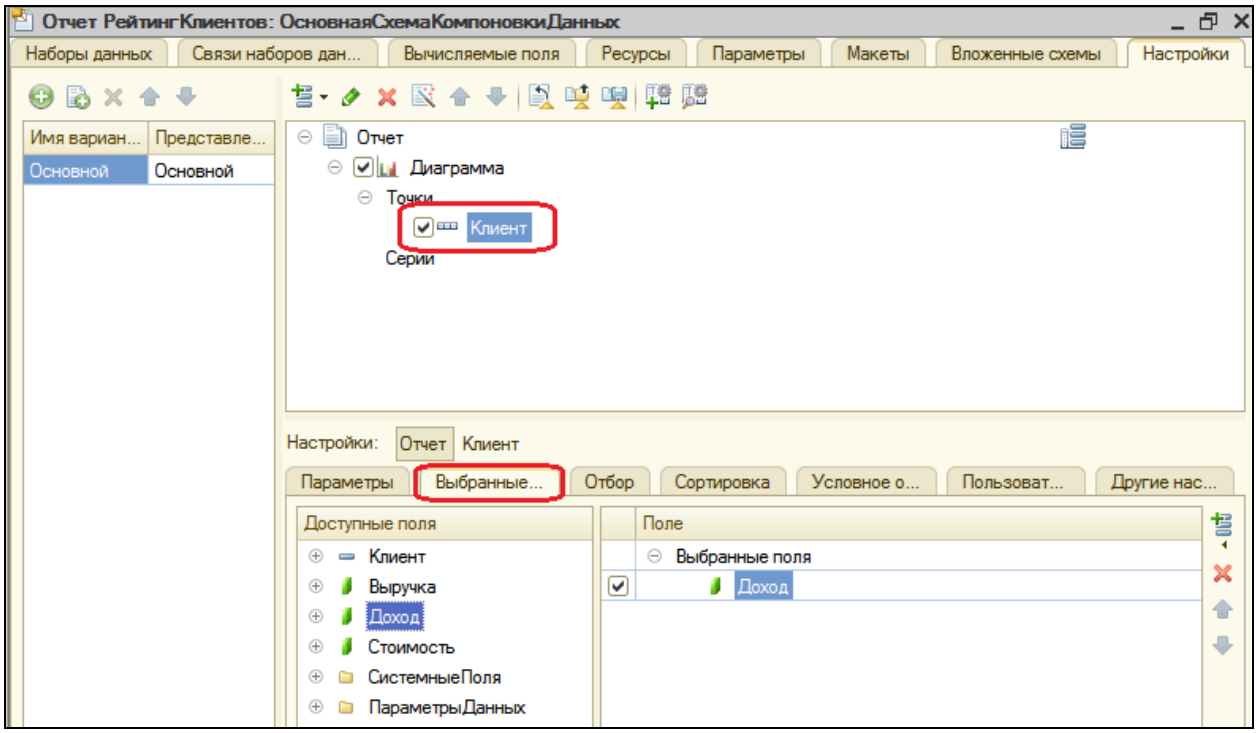


Настройки

На закладке **Настройки** добавьте в структуру отчета диаграмму. Затем выделите ветку **Точки** и добавьте в нее группировку по полю **Клиент**. В значения диаграммы всегда выводится один из ресурсов отчета.



Перейдите на закладку **Выбранные поля** и выберете поле **Доход** для вывода в отчет.

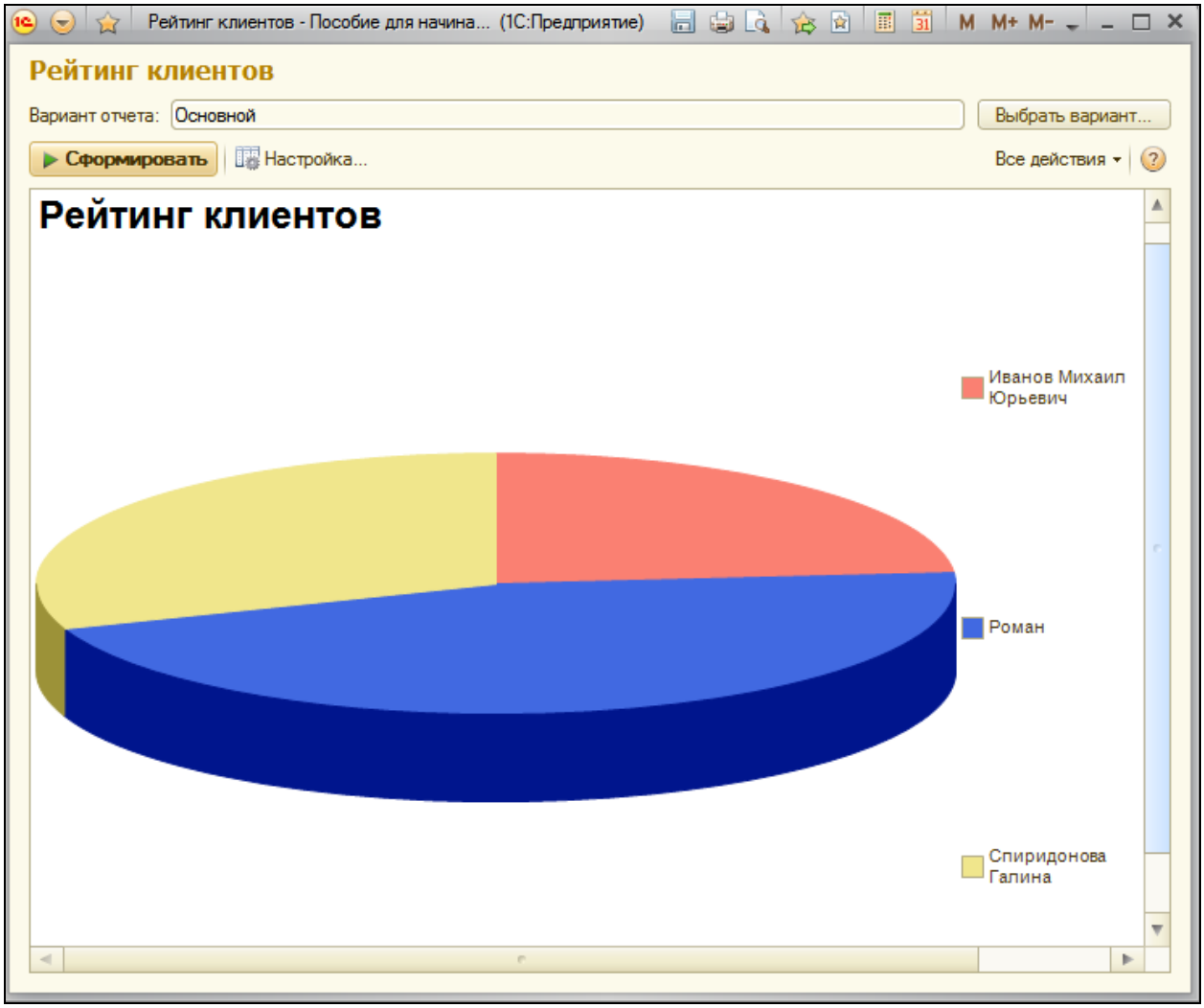


На закладке **Другие настройки** выберете тип диаграммы – **Круговая объемная** и задайте заголовок отчета – **Рейтинг клиентов**.

Отметьте в списке подсистем **Оказание услуг** и **Бухгалтерия.**

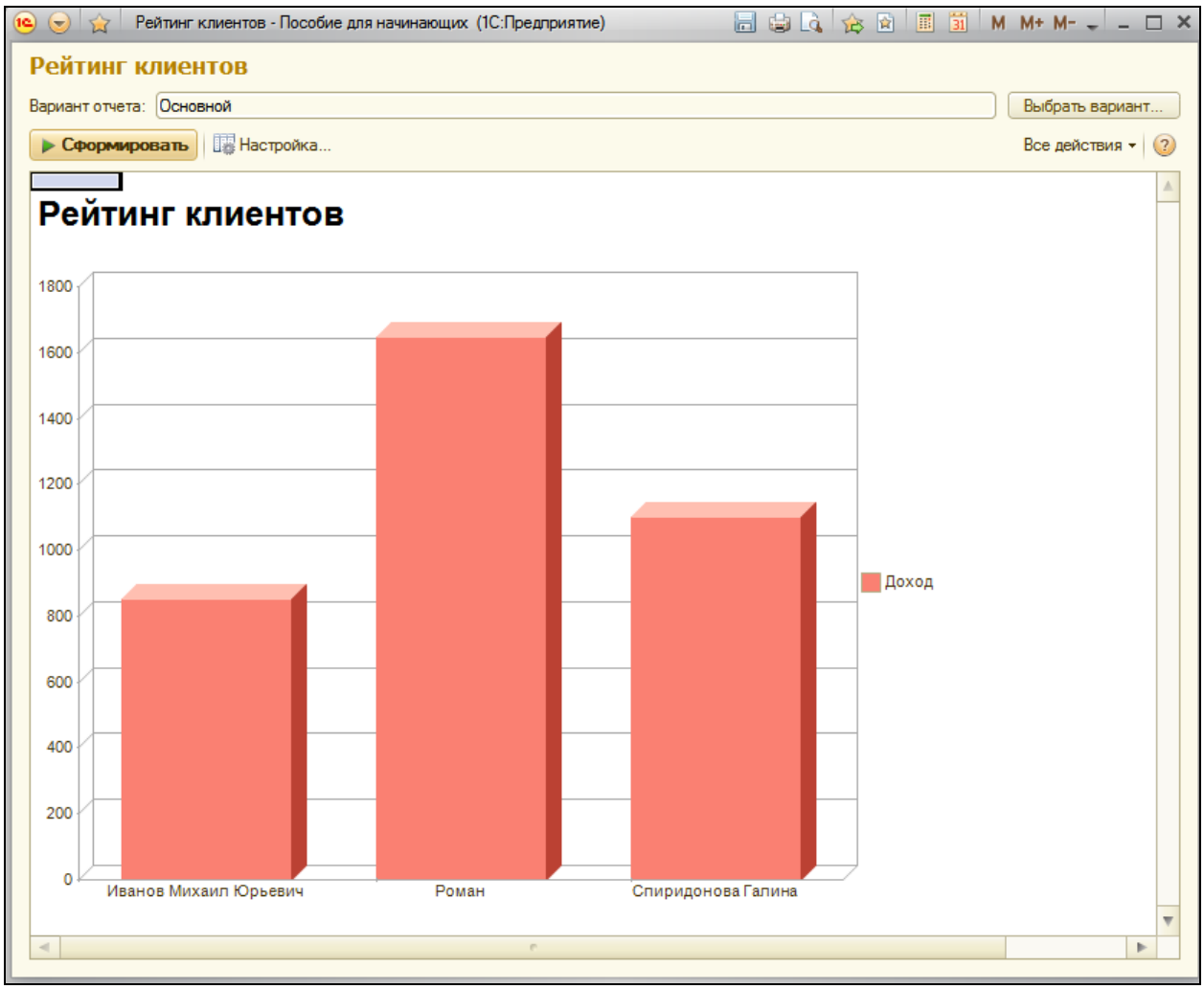
В режиме 1С: Предприятие

Запустите режим отладки и выполните **Рейтинг клиентов** в панели действий **Бухгалтерии**. Нажмите **Сформировать**.



Вернитесь в конфигуратор и измените тип диаграммы на **Гистограмма объемная**. Заново сформируйте отчет.

Таким образом, мы продемонстрировали, как можно использовать различные виды диаграмм для визуализации данных отчета.



Вывод данных в таблицу

На примере создания универсального отчета, мы продемонстрируем вывод данных в таблицу.

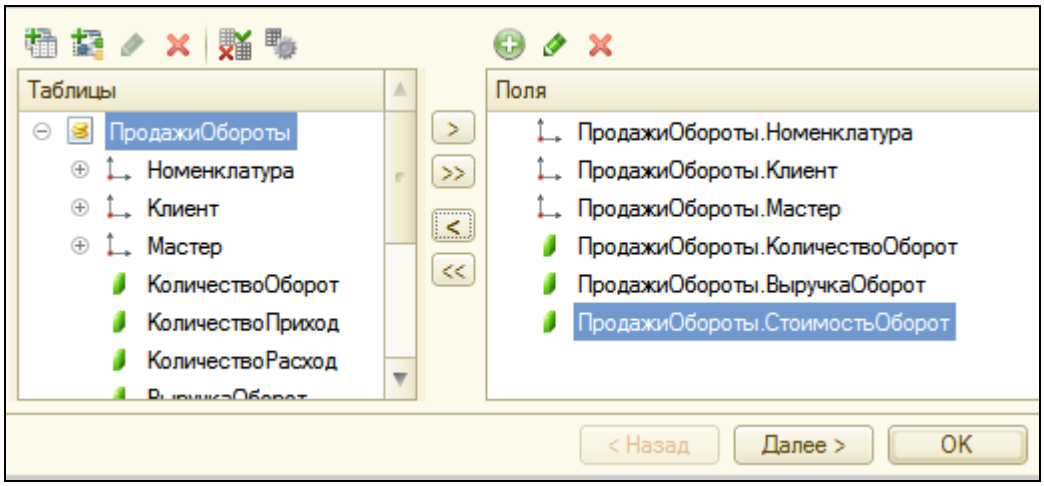
Мы покажем как сделать отчет максимально универсальным, чтобы позволить пользователю в режиме 1С: Предприятие, не обращаясь к полным настройкам отчета (не выполняя Все действия – Изменить вариант), изменять его структуру и внешний вид. Например, поменять местами строки и колонки таблицы или изменить данные, выводящиеся в ячейках таблицы.

В режиме Конфигуратор

Добавим новый отчет **Универсальный** и запустим конструктор компоновки данных. Создайте новый **Набор данных – запрос** и зайдите в конструктор запроса.

Запрос для набора данных

В качестве источника данных для запроса выберем виртуальную таблицу регистра накопления **Продажи.Обороты**. Затем выберем из нее указанные поля.



Анализ текста запроса

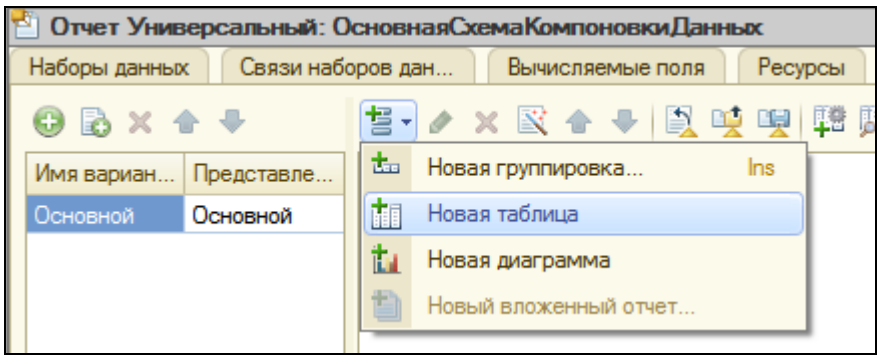
```
ВЫБРАТЬ
	ПродажиОбороты.Номенклатура,
	ПродажиОбороты.Клиент,
	ПродажиОбороты.Мастер,
	ПродажиОбороты.КоличествоОборот,
	ПродажиОбороты.ВыручкаОборот,
	ПродажиОбороты.СтоимостьОборот
ИЗ
	РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
```

Ресурсы

На закладке Ресурсы выберете все доступные ресурсы отчета.

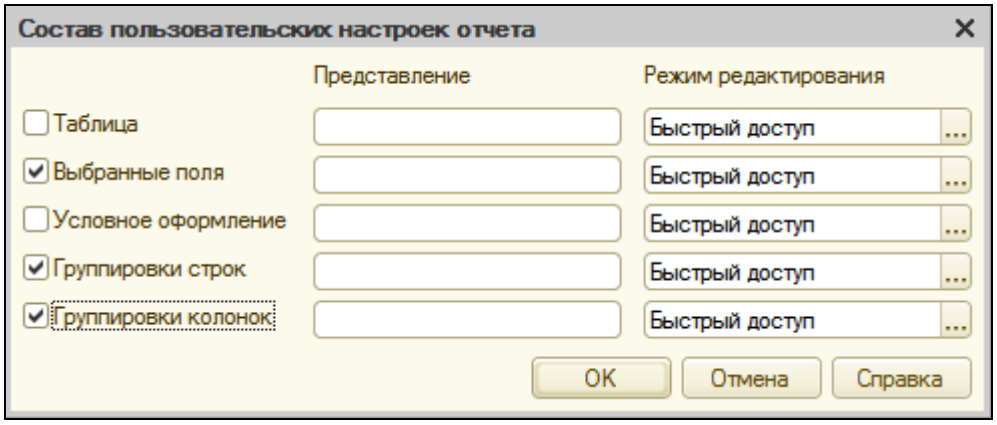
Настройки

На закладке **Настройки** добавим в структуру отчета таблицу. Для этого нажмите **Добавить** в командной панели окна настроек.



Мы не будем задавать строки и колонки этой таблицы и список выбранных полей, потому что хотим предоставить полную свободу пользователю в этих действиях. Для этого выделим в структуре элементов отчета **Таблица** и нажмем кнопку **Свойства элемента пользовательских настроек**  .

Установите галочки для **Выбранные поля**, **Группировки строк** и **Группировки колонок**. Оставим их свойства по умолчанию.



Так мы предоставили пользователю возможность самостоятельно определять состав выбранных полей, группировок строк и колонок таблицы непосредственно в отчетной форме перед формированием отчета.

Определим, что отчет будет отображаться в подсистеме **Оказание услуг**.

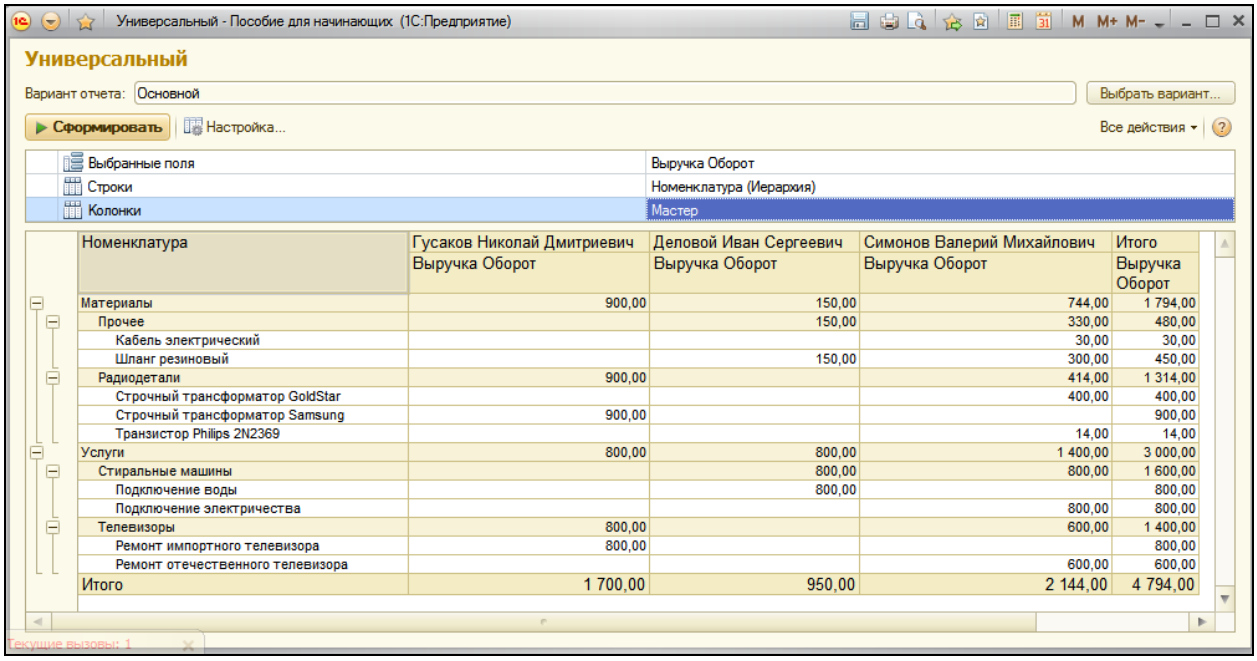
В режиме 1С: Предприятие

Запустите отладку и выполните команду **Универсальный** в панели действий **Оказание услуг**.

Нажмите кнопку выбора в строке **Выбранные поля** и выберете поле **ВыручкаОборот**.

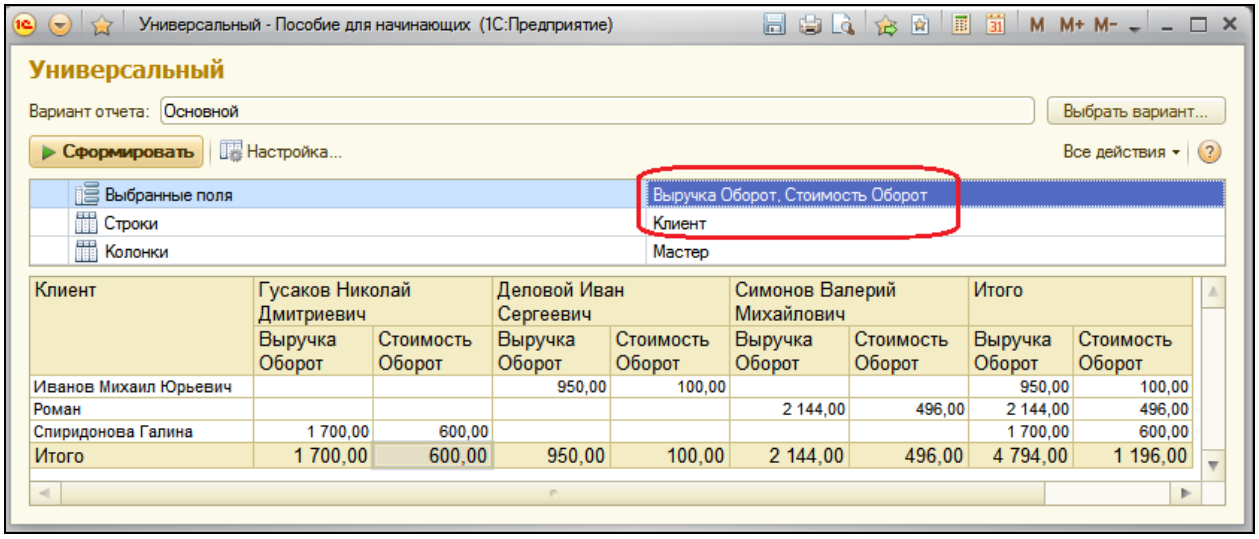
В поле **Строки** добавьте в строки таблицы группировку по полю **Номенклатура** с типом **Иерархия**.

В строке **Колонки** добавьте в колонки таблицы группировку по полю **Мастер**. Нажмите **Сформировать**.



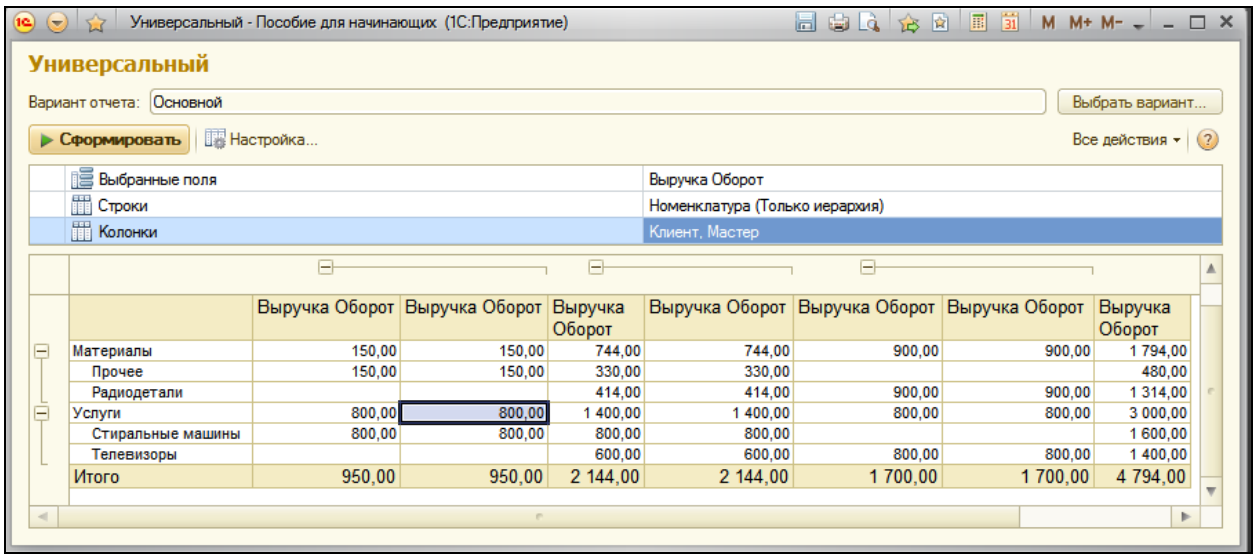
Номенклатура	Гусаков Николай Дмитриевич Выручка Оборот	Деловой Иван Сергеевич Выручка Оборот	Симонов Валерий Михайлович Выручка Оборот	Итого Выручка Оборот
Материалы	900,00	150,00	744,00	1 794,00
Прочее		150,00	330,00	480,00
Кабель электрический			30,00	30,00
Шланг резиновый		150,00	300,00	450,00
Радиодетали	900,00		414,00	1 314,00
Строчный трансформатор GoldStar			400,00	400,00
Строчный трансформатор Samsung	900,00			900,00
Транзистор Philips 2N2369			14,00	14,00
Услуги	800,00	800,00	1 400,00	3 000,00
Стиральные машины		800,00	800,00	1 600,00
Подключение воды		800,00		800,00
Подключение электричества			800,00	800,00
Телевизоры	800,00		600,00	1 400,00
Ремонт импортного телевизора	800,00			800,00
Ремонт отечественного телевизора			600,00	600,00
Итого	1 700,00	950,00	2 144,00	4 794,00

Теперь добавим в список выбранных полей **СтоимостьОборот**. И в строки таблицы вместо группировки по полю **Номенклатура** поместите группировку по полю **Клиент**.



Клиент	Гусаков Николай Дмитриевич		Деловой Иван Сергеевич		Симонов Валерий Михайлович		Итого	
	Выручка Оборот	Стоимость Оборот	Выручка Оборот	Стоимость Оборот	Выручка Оборот	Стоимость Оборот	Выручка Оборот	Стоимость Оборот
Иванов Михаил Юрьевич			950,00	100,00			950,00	100,00
Роман					2 144,00	496,00	2 144,00	496,00
Спиридонова Галина	1 700,00	600,00					1 700,00	600,00
Итого	1 700,00	600,00	950,00	100,00	2 144,00	496,00	4 794,00	1 196,00

Теперь исключите из списка **выбранных полей СтоимостьОборот**. В **строках** таблицы заменим прежнюю группировку по **Клиент** на группировку по полю **Номенклатура** с типом **Только Иерархия**. В **колонки** таблицы добавим группировку по полю **Клиент** и поместим ее первой в списке группировок.



	Выручка Оборот	Выручка Оборот	Выручка Оборот	Выручка Оборот	Выручка Оборот	Выручка Оборот	Выручка Оборот
Материалы	150,00	150,00	744,00	744,00	900,00	900,00	1 794,00
Прочее	150,00	150,00	330,00	330,00			480,00
Радиодетали			414,00	414,00	900,00	900,00	1 314,00
Услуги	800,00	800,00	1 400,00	1 400,00	800,00	800,00	3 000,00
Стиральные машины	800,00	800,00	800,00	800,00			1 600,00
Телевизоры			600,00	600,00	800,00	800,00	1 400,00
Итого	950,00	950,00	2 144,00	2 144,00	1 700,00	1 700,00	4 794,00

Таким образом, используя этот отчет, мы предоставили пользователю альтернативную возможность самостоятельно формировать отчет по регистру **Продажи**.

Контрольные вопросы

- ✓ Для чего предназначена система компоновки данных.
- ✓ Для чего предназначены настройки компоновки данных.
- ✓ В чем отличие между реальными и виртуальными таблицами.
- ✓ Что является источником данных запроса.
- ✓ Что такое псевдонимы в языке запросов.
- ✓ Что такое параметры запроса.
- ✓ Что такое параметры виртуальной таблицы.
- ✓ Что такое левое соединение.
- ✓ Как использовать конструктор запроса.
- ✓ Как выбрать данные в некотором периоде для отчета.
- ✓ Как упорядочить данные в отчете.
- ✓ Как использовать в отчете данные нескольких таблиц.
- ✓ Как использовать группировки в структуре отчета.
- ✓ Как получить последние значения регистра сведений.
- ✓ Как вывести в отчет иерархические данные.
- ✓ Как управлять выводом итогов по группировкам и общих итогов.

- ✓ Как создать отчет с диаграммой.
- ✓ Как использовать параметры в системе компоновки данных.
- ✓ Что такое ресурсы в системе компоновки данных.
- ✓ Что такое вычисляемые поля в системе компоновки данных.
- ✓ Как создать пользовательские настройки отчета
- ✓ В чем отличие быстрых настроек от остальных пользовательских.
- ✓ Как определить состав пользовательских настроек отчета.
- ✓ Как вывести данные в виде таблицы.
- ✓ Как сделать отчет универсальным.

Практическая работа № 13

Оптимизация проведения документа «Оказание услуги» (3:20)

После изучения предыдущего занятия вы уже достаточно знакомы с языком запросов и можете приступить к одному из самых важных занятий – к оптимизации документа **ОказаниеУслуги**, в частности – к полному изменению обработчика события **ОбработкаПроведения**.

Зачем это нужно? Этому есть три причины.

Во-первых, в обращении к событию **ОбработкаПроведения** используется обращение через точку, что может сильно замедлить скорость проведения при больших объемах табличной части документа.

Во-вторых, руководство нашей фирмы решило прекратить ручной ввод стоимости расходуемых материалов и перейти на автоматический расчет «по среднему».

В-третьих, при проведении документа необходимо контролировать остатки расходуемых товаров на складе. Если товаров не хватает – выдавать предупреждение и не проводить документ.

Поэтому в нашей работе три цели:

1. Повышение скорости выполнения процедуры;
2. Автоматическое определение стоимости расходуемых материалов при проведении документа;
3. Разделение алгоритма проведения документа на оперативный и неоперативный режимы и контроль остатков в оперативном проведении документа.

Если алгоритм проведения документа использует только те данные, которые присутствуют в реквизитах документа (и его табличных частях), вполне достаточно использовать конструктор движений документа.

Если же в алгоритме проведения необходимо анализировать дополнительные реквизиты объектов, ссылки на которые содержатся в документе, а также использовать результаты расчета итогов регистров, следует использовать запросы для более быстрой выборки.

Механизм запросов лучше «читает» информационную базу и может за один раз выбрать только нужные данные. Поэтому в типовых решениях

вы не увидите использование объекта встроенного языка **СправочникВыборка. <имя>**. Вместо этого используются запросы к БД.

Повышение скорости проведения

Первое, чем мы займемся в этой работе – избавимся от «вредной» конструкции **ТекСтрокаПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры.**

В режиме Конфигуратор

Откройте модуль документа **ОказаниеУслуги**. Из процедуры обработки проведения видно, что все данные, необходимые для проведения документа, мы получаем из самого документа и только для определения типа номенклатуры (товар или услуга) – читая данные всего объекта Номенклатура.

Обращение к объекту **Номенклатура**:

```
Если ТекСтрокаПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры
```

Это не единственные данные, которые содержатся не в самом документе и которые будут нужны для его правильного проведения.

Для оптимизации поступим следующим образом: все данные, связанные с номенклатурой, которая содержится в табличной части документа, мы будем получать с помощью запроса к БД. А данные, связанные с самим документом (дата, склад..) будем по-прежнему получать из документа. Такой подход позволит читать только нужные данные и максимально ускорить проведение документа.

Запросом мы будем получать:

- Номенклатуру
- Количество
- Сумму
- Стоимость

Из документа возьмем:

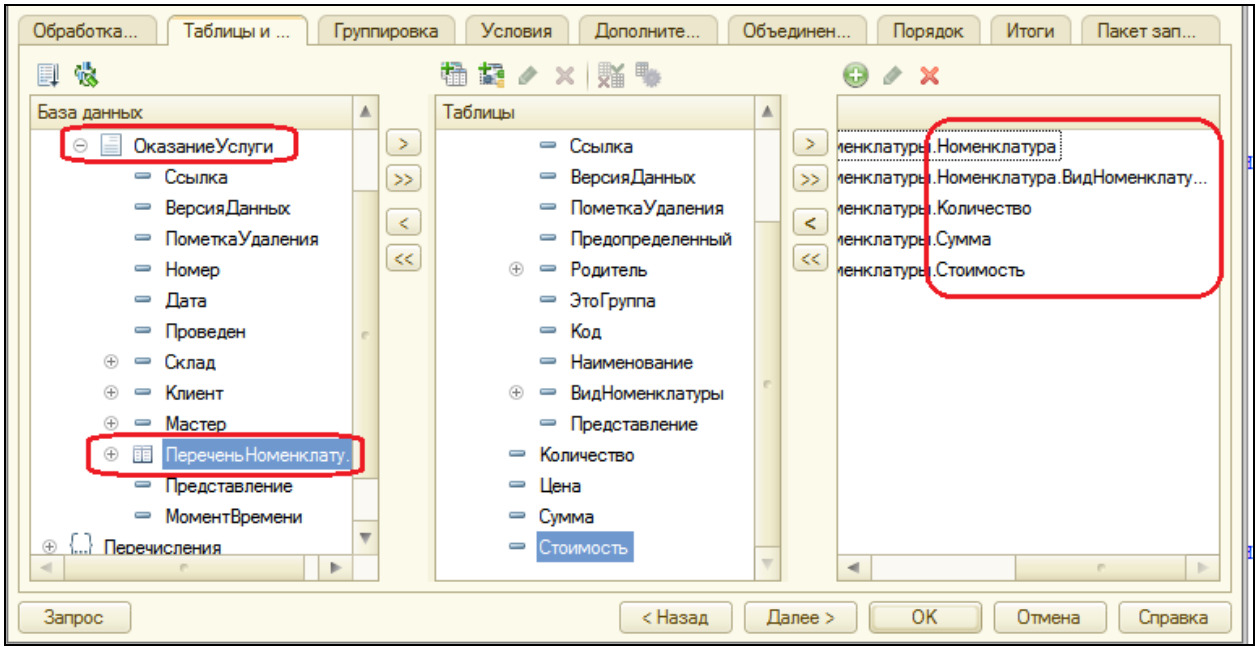
- Дата

- Клиент
- Мастер
- Склад

Установите курсор перед циклом Если… и из контекстного меню выберите **Конструктор запроса с обработкой результата**. Подтвердите создание нового запроса.

В окне конструктора перейдите на вкладку **Таблицы и поля** и выберите таблицу **ОказаниеУслугиПеречеьНоменклатуры –** это табличная часть документа **ОказаниеУслуги**.

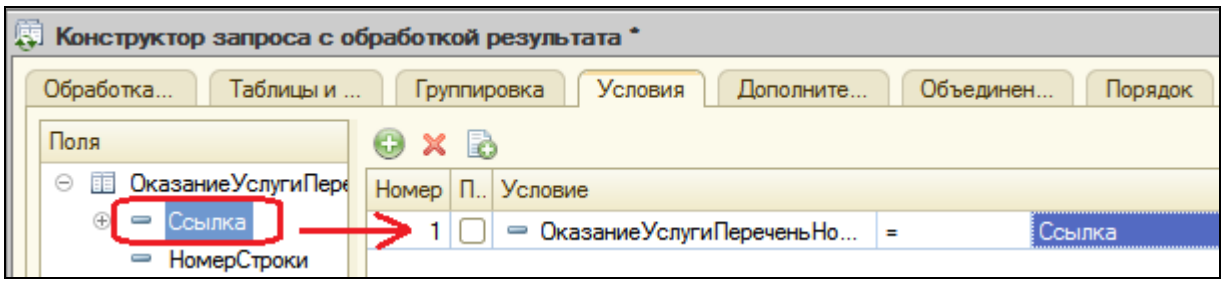
Из этой таблицы нам нужны поля – **Номенклатура**, **Номенклатура.ВидНоменклатуры**, **Количество**, **Сумма** и **Стоимость**.



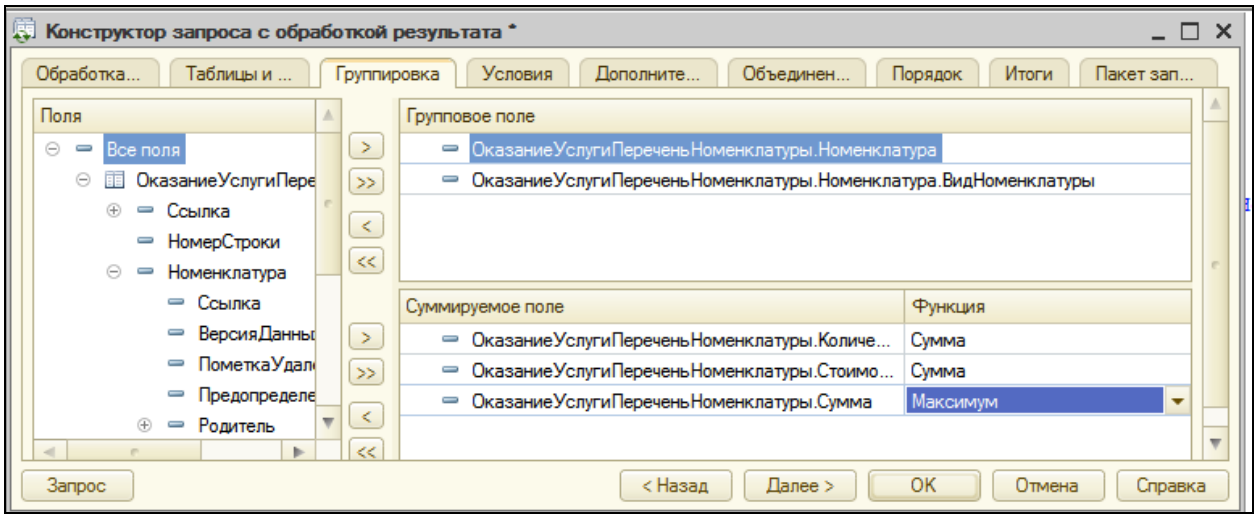
Но нам нужны не все записи этой таблицы, а только те, которые относятся к нашему документу. Поэтому перейдите на вкладку **Условия** и задайте условие отбора из таблицы только строк проводимого документа. Для этого перетащите поле **Ссылка** в список условий запроса:

В обработчике появится условие:

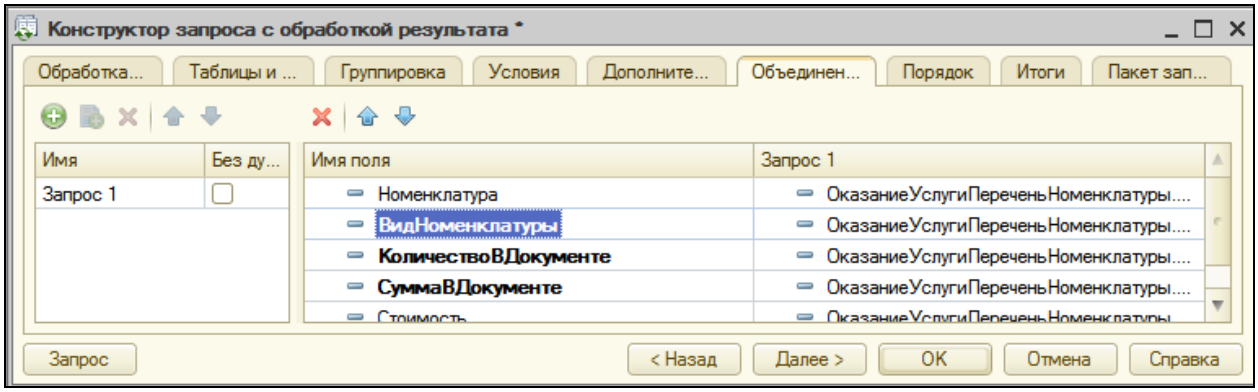
```
ОказаниеУслугиПеречеьНоменклатуры.Ссылка=&Ссылка
```



Следует учесть, что в табличной части документа одна и та же номенклатура может встречаться несколько раз. Поэтому на закладке **Группировка** сгруппируйте записи по полю **Номенклатура** и **НоменклатураВидНоменклатур**ы, а рассчитывать будем сумму значений для полей **Количество** и **Сумма**. Также в состав суммируемых полей включим поле **Стоимость**. По нему будем рассчитывать, например, функцию **Максимум**.



На закладке **Объединения/Псевдонимы** задайте псевдонимы для полей **Количество** и **Сумма** – **КоличествоВДокументе** и **СуммаВДокументе**, а для поля **НоменклатураВидНоменклатуры** - **ВидНоменклатуры**.



Нажмите ОК и посмотрите какой текст запроса сформирован.

```
//{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
		|
	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры	КАК ВидНоменклатуры,
		|	СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Количество)	КАК КоличествоВДокументе,
		|	МАКСИМУМ(ОказаниеУслугиПеречеНьНоменклатуры.Сумма)	КАК СуммаВДокументе,
		|	СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Стоимость)	КАК Стоимость
		|ИЗ
		|	Документ.ОказаниеУслуги.ПеречеНьНоменклатуры	КАК ОказаниеУслугиПеречеНьНоменклатуры
		|ГДЕ
		|	ОказаниеУслугиПеречеНьНоменклатуры.Ссылка = &Ссылка
		|
		|СГРУППИРОВАТЬ ПО
		|	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
		|
	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры";

	Запрос.УстановитьПараметр("Ссылка", Ссылка);

	Результат = Запрос.Выполнить();

	ВыборкаДетальныеЗаписи = Результат.Выбрать();

	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
		// Вставить обработку выборки ВыборкаДетальныеЗаписи
	КонецЦикла;

	//}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
```

Проанализируем текст. Для работы с запросами используется объект встроенного языка **Запрос**. Вначале создается новый объект **Запрос** и помещается в переменную **Запрос**. Затем в свойство **Текст** объекта **Запрос** помещается сам текст запроса (**Запрос.Текст**=…). После этого устанавливается значение параметра запроса **&Ссылка** как ссылка на тот документ, в модуле которого мы сейчас находимся.

```
Запрос.УстановитьПараметр("Ссылка", Ссылка);
```

Затем запросы выполняется (**Запрос.Выполнить()**), получается объект **РезультатЗапроса**, и выполняется его метод **Выбрать()**, который формирует выборку записей из результата запроса.

Получается объект **ВыборкаИзРезультатаЗапроса**, который помещается в переменную **ВыборкаДетальныеЗаписи**.

Далее, используя метод этого объекта **Следующий()** (**ВыборкаДетальныеЗАписи.Следующий()**), мы будем в цикле обходить выборку записей запроса. Выполняя метод выборки запроса **ВыборкаДетальныеЗаписи.Следующий()**, мы на каждом шаге цикла позиционируем указатель на следующую запись выборки, пока не будет достигнут конец выборки.

Чтобы в цикле получить значение какого-либо поля выборки из результата запроса, мы будем обращаться к полям запроса через точку от переменной **ВыборкаДетальныеЗаписи**, которая содержит текущую строку выборки запроса. Например, так: **ВыборкаДетальныеЗаписи.Номенклатура**.

Теперь осталось перенести существовавшие в этом модуле ранее строки, описывающие движения регистров, внутрь цикла обхода результата запроса.

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                // Вставить обработку выборки ВыборкаДетальныеЗаписи
        КонецЦикла;
```

Новый текст в листинге будет выделен **жирным** текстом для удобства восприятия.

Сначала вместо комментария «// Вставить обработку выборки ВыборкаДетальныеЗаписи» перенесите условие проверки и весь код, формирующий движения по регистрам **ОстаткиМатериалов** и **СтоимостьМатериалов**.

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                    Если
ТекСтрокаПеречеНоменклатуры.Номенклатура.ВидНоменклатуры =
Перечисления.ВидыНоменклатуры.Материал Тогда

                    // регистр ОстаткиМатериалов Расход
                    Движение = Движения.ОстаткиМатериалов.Добавить();
                    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                    Движение.Период = Дата;
                    Движение.Материал =
ТекСтрокаПеречеНоменклатуры.Номенклатура;
                    Движение.Склад = Склад;
                    Движение.Количество =
ТекСтрокаПеречеНоменклатуры.Количество;

                    // регистр СтоимостьМатериалов Расход
                    Движение = Движения.СтоимостьМатериалов.Добавить();
                    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                    Движение.Период = Дата;
```

```
                    Движение.Материал = ТекСтрокаПереченьНоменклатуры.Номенклатура;
                    Движение.Стоимость = ТекСтрокаПереченьНоменклатуры.Количество*ТекСтрокаПереченьНоменклатуры.Стоимость;
                КонецЕсли;

        КонецЦикла;
```

В условии замените ТекСтрокаПереченьНоменклатуры.Номенклатура на ВыборкаДетальныеЗаписи, так как вид номенклатуры мы теперь получаем из запроса. В движениях также заменим ТекСтрокаПереченьНоменклатуры на ВыборкаДетальныеЗаписи. Для поля **Количество** мы задали псевдоним в запросе, поэтому заменим его на **КоличествоВДокументе**.

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                    Если ВыборкаДетальныеЗаписи.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал Тогда

                    // регистр ОстаткиМатериалов Расход
                    Движение = Движения.ОстаткиМатериалов.Добавить();
                    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                    Движение.Период = Дата;
                    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
                    Движение.Склад = Склад;
                    Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

                    // регистр СтоимостьМатериалов Расход
                    Движение = Движения.СтоимостьМатериалов.Добавить();
                    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                    Движение.Период = Дата;
                    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
                    Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе*ВыборкаДетальныеЗаписи.Стоимость;
                КонецЕсли;

        КонецЦикла;
```

Теперь перенесем формирование движений по регистру **Продажи**:

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
                    Если ВыборкаДетальныеЗаписи.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал Тогда

                    // регистр ОстаткиМатериалов Расход
                    Движение = Движения.ОстаткиМатериалов.Добавить();
                    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                    Движение.Период = Дата;
                    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
                    Движение.Склад = Склад;
```

```
                    Движение.Количество                                    =
ВыборкаДетальныеЗаписи.КоличествоВДокументе;

                    // регистр СтоимостьМатериалов Расход
                    Движение = Движения.СтоимостьМатериалов.Добавить();
                    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                    Движение.Период = Дата;
                    Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
                    Движение.Стоимость                                     =
ВыборкаДетальныеЗаписи.КоличествоВДокументе*ВыборкаДетальныеЗаписи.Стоимость
;
            КонецЕсли;
      // Регистр Продажи
            Движение = Движения.Продажи.Добавить();
            Движение.Период = Дата;
            Движение.Номенклатура                                          =
ТекСтрокаПеречеНоменклатуры.Номенклатура;
            Движение.Клиент = Клиент;
            Движение.Мастер = Мастер;
            Движение.Количество                                            =
ТекСтрокаПеречеНоменклатуры.Количество;
            Движение.Выручка = ТекСтрокаПеречеНоменклатуры.Сумма;
            Движение.Стоимость                                             =
ТекСтрокаПеречеНоменклатуры.Стоимость                                      *
ТекСтрокаПеречеНоменклатуры.Количество;

КонецЦикла;
```

Здесь произведем аналогичные замены. ТекСтрокаПеречеНоменклатуры заменим на ВыборкаДетальныеЗаписи. А также поля Сумма и Количество заменим на их псевдонимы СуммаВДокументе и КоличествоВДокументе.

```
// Регистр Продажи
            Движение = Движения.Продажи.Добавить();
            Движение.Период = Дата;
            Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
            Движение.Клиент = Клиент;
            Движение.Мастер = Мастер;
            Движение.Количество                                            =
ВыборкаДетальныеЗаписи.КоличествоВДокументе;
            Движение.Выручка = ВыборкаДетальныеЗаписи.СуммаВДокументе;
            Движение.Стоимость    =    ВыборкаДетальныеЗаписи.Стоимость    *
ВыборкаДетальныеЗаписи.КоличествоВДокументе;

      КонецЦикла;
```

Оставшийся цикл обхода табличной части можно удалить:

```
Для Каждого ТекСтрокаПеречеНоменклатуры Из ПеречеНоменклатуры Цикл

                    КонецЦикла;
```

Процедура проведения примет следующий вид:

```
Процедура ОбработкаПроведения(Отказ, Режим)

	Движения.ОстаткиМатериалов.Записывать = Истина;
	Движения.СтоимостьМатериалов.Записывать = Истина;
	Движения.Продажи.Записывать = Истина;

		//{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
		|	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры КАК
ВидНоменклатуры,
		|	СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Количество) КАК
КоличествоВДокументе,
		|	СУММА(ОказаниеУслугиПеречеНьНоменклатуры.Сумма) КАК СуммаВДокументе,
		|	МАКСИМУМ(ОказаниеУслугиПеречеНьНоменклатуры.Стоимость) КАК Стоимость
		|ИЗ
		|	Документ.ОказаниеУслуги.ПеречеНьНоменклатуры КАК
ОказаниеУслугиПеречеНьНоменклатуры
		|ГДЕ
		|	ОказаниеУслугиПеречеНьНоменклатуры.Ссылка = &Ссылка
		|
		|СГРУППИРОВАТЬ ПО
		|	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
		|	ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры";

	Запрос.УстановитьПараметр("Ссылка", Ссылка);

	Результат = Запрос.Выполнить();

	ВыборкаДетальныеЗаписи = Результат.Выбрать();

	Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
			Если ВыборкаДетальныеЗаписи.ВидНоменклатуры =
Перечисления.ВидыНоменклатуры.Материал Тогда

			// регистр ОстаткиМатериалов Расход
			Движение = Движения.ОстаткиМатериалов.Добавить();
			Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
			Движение.Период = Дата;
			Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
			Движение.Склад = Склад;
			Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

			// регистр СтоимостьМатериалов Расход
			Движение = Движения.СтоимостьМатериалов.Добавить();
			Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
			Движение.Период = Дата;
			Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
			Движение.Стоимость =
ВыборкаДетальныеЗаписи.КоличествоВДокументе*ВыборкаДетальныеЗаписи.Стоимость;
		КонецЕсли;
	// Регистр Продажи
		Движение = Движения.Продажи.Добавить();
		Движение.Период = Дата;
		Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
```

```
                Движение.Клиент = Клиент;
                Движение.Мастер = Мастер;
                Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;
                Движение.Выручка = ВыборкаДетальныеЗаписи.СуммаВДокументе;
                Движение.Стоимость = ВыборкаДетальныеЗаписи.Стоимость * ВыборкаДетальныеЗаписи.КоличествоВДокументе;

        КонецЦикла;


КонецПроцедуры
```

Теперь запустите 1С: Предприятие в режиме отладки и перепроведите документы **ОказаниеУслуги**, проверьте, что ничего не изменилось.

Т.о. мы выполнили первый пункт нашего плана – избавились в процедуре проведения от считывания всех данных объекта **Номенклатура**, оптимизировав выполнение процедуры проведения.

Автоматический расчет стоимости

Приступим ко второму этапу плана. До сих пор стоимость расходуемых материалов мы вписывали в документ **Оказание услуги** вручную, при его создании.

Теперь же будем определять стоимость номенклатуры «по среднему»: для каждой номенклатуры делить ее общую, суммарную стоимость на имеющееся у нас количество номенклатуры, т.о. получая среднюю стоимость единицы номенклатуры.

Чтобы выполнить такой расчет, понадобятся дополнительные данные, которых сейчас нет. Для каждой номенклатуры из табличной части нам понадобятся:

- Ее стоимость, хранящаяся в регистре **СтоимостьМатериалов**;
- Общее ее количество на всех складах, хранящееся в регистре **ОстаткиМатериалов**.

Поэтому нам нужно будет доработать запрос, чтобы он получал из БД и эти данные тоже. Т.о. нам хотелось бы, чтобы запрос возвращал следующие поля для каждой номенклатуры, которая есть в документе:

Табличная часть документа				**Регистр Стоимость материалов**	**Регистр Остатки материалов**
Номенклатура	Кол-во в документе	Сумма в документе	Вид номенклатуры	Стоимость	Кол-во на всех складах

Первые четыре поля мы уже получаем из табличной части самого документа, а последние два нужно будет получить из других таблиц.

Это значит, что наш запрос должен содержать два левых соединения таблицы документа с другими таблицами: одно – с таблицей **РегистрНакопления.СтоимостьМатериалов.Остатки**; другое – с таблицей **РегистрНакопления.ОстаткиМатериалов.Остатки**.

Важная деталь: в предложенной схеме виртуальные таблицы будут возвращать стоимость и остатки абсолютно для всей номенклатуры, когда нас интересует только та, которая указана в нашем документе.

Для маленькой базы это не важно, но для реальной БД, где 15 000 наименований и только 5 используются в документе – это непозволительная расточительность вычислительных ресурсов.

Поэтому во все виртуальные таблицы, которые мы будем использовать, нужно добавить условие отбора только номенклатуры из табличной части нашего документа. В этом случае стоимость и остатки будут рассчитаны не для всей номенклатуры вообще, а только для нужной нам.

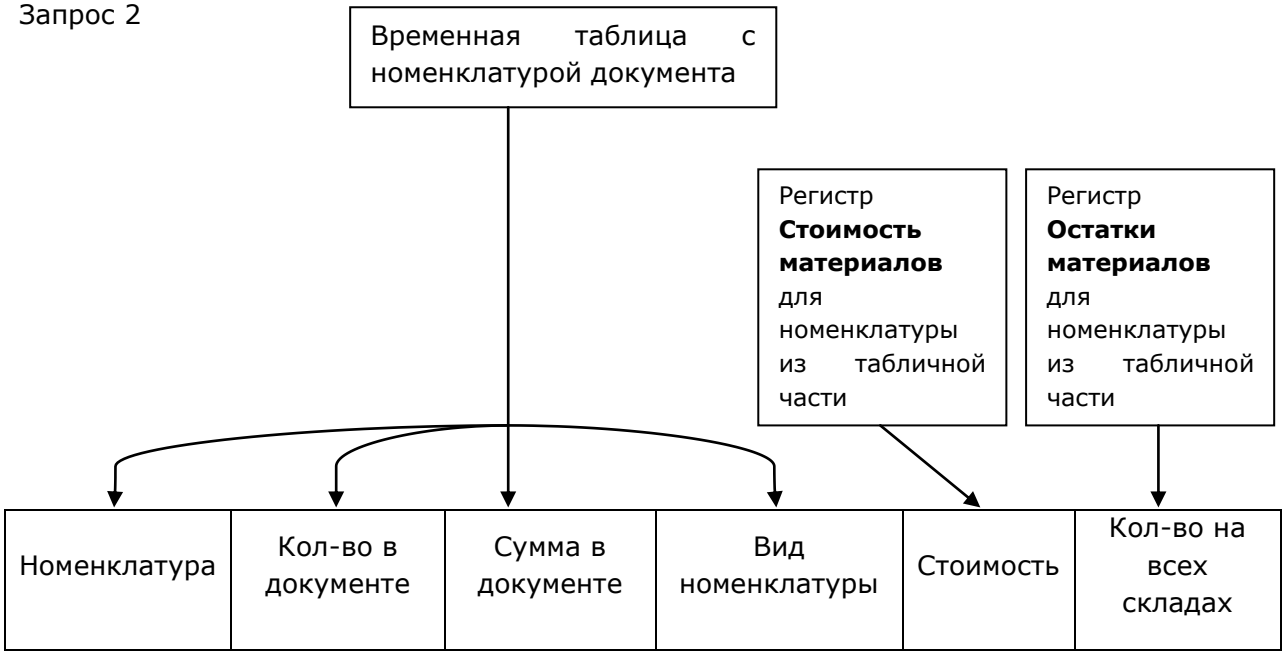
Чтобы не получать список номенклатуры три раза (для документа и в каждой виртуальной таблице заново), мы можем сформировать его заранее и затем уже использовать в нужных нам условиях запроса.

Выполнить эту задачу нам помогут *временные таблицы* – программные объекты, которые разработчик может создать и заполнить данными, а запросы могут использовать данные временных таблиц для своих нужд.

Таким образом, схема нашего запроса приобретает следующий вид:

Запрос 1

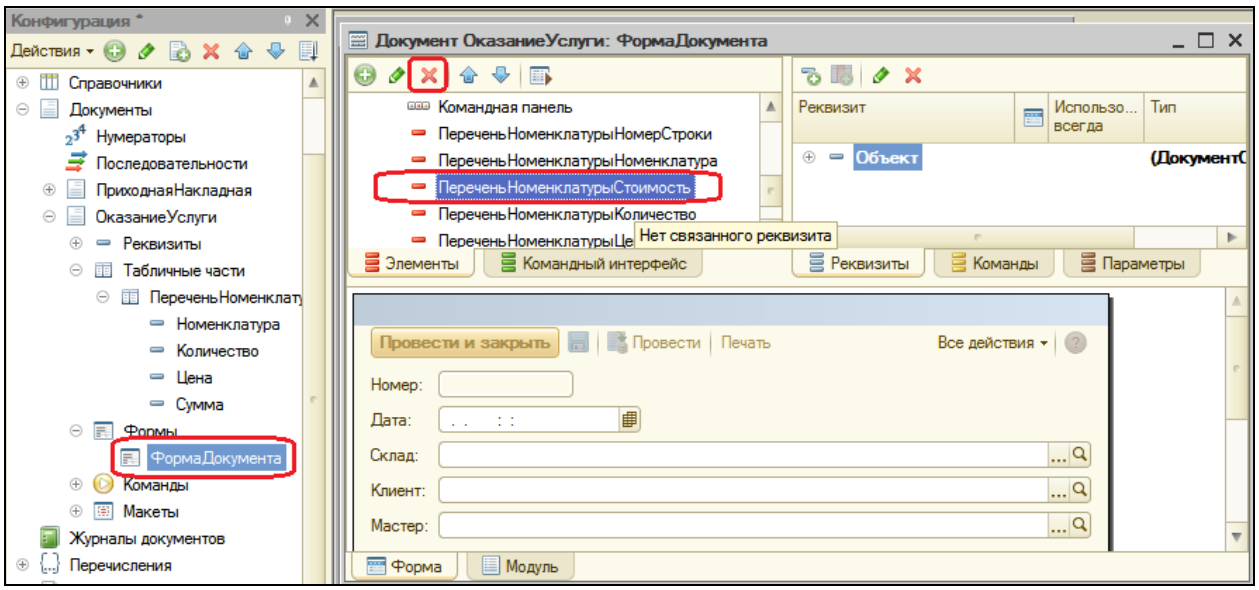
Запрос 2



В режиме Конфигуратор

Первое – мы удалим реквизит табличной части **Стоимость** документа **ОказаниеУслуги**, который нам больше не понадобится. Для этого откройте окно редактирования объекта Документа **ОказаниеУслуги**, перейдите на закладку **Данные**, раскройте список реквизитов табличной части документа, выделите реквизит **Стоимость** и нажмите кнопку **Удалить** в командной панели. Хотя проще это сделать из окна конфигурации.

Также удалите поле **Стоимость** из таблицы **ПереченьНоменклатуры**, расположенной в форме.



Временную таблицу мы сформируем с помощью того запроса, который у нас уже написан. Откройте модуль документа **ОказаниеУслуги**.

В процедуре **ОбработкаПроведения()** перед созданием запроса создайте менеджер временных таблиц и укажите, что этот запрос будет использовать созданный менеджер:

```
Процедура ОбработкаПроведения(Отказ, Режим)

	Движения.ОстаткиМатериалов.Записывать = Истина;
	Движения.СтоимостьМатериалов.Записывать = Истина;
	Движения.Продажи.Записывать = Истина;

	//Создать менеджер временных таблиц.
	МенеджерВТ = Новый МенеджерВременныхТаблиц;

	Запрос = Новый Запрос;
	//Укажем, какой менеджер временных таблиц использует этот запрос
	Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

	Запрос.Текст =
		"ВЫБРАТЬ
		|	ОказаниеУслугиПеречеНоменклатуры.Номенклатура,
```

Теперь изменим запрос, чтобы он создавал временную таблицу, которая будет храниться в нашем менеджере временных таблиц **МенеджерВТ**.

Чтобы конструктор запроса смог открыть наш запрос, удалите из него строку (поля Стоимость у нас больше нет) и запятую в строке выше этой:

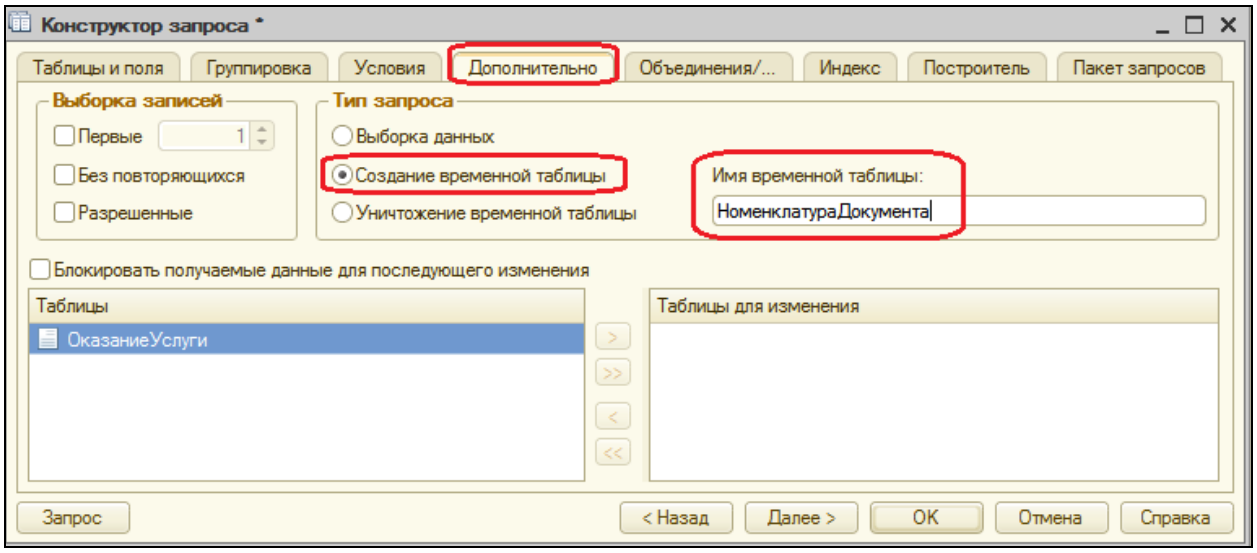
```
МАКСИМУМ(ОказаниеУслугиПеречеНоменклатуры.Стоимость) КАК Стоимость
```

Получится следующее:

```
Запрос.Текст =
                "ВЫБРАТЬ
                |       ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
                |
        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры      КАК
ВидНоменклатуры,
                |       СУММА(ОказаниеУслугиПереченьНоменклатуры.Количество)     КАК
КоличествоВДокументе,
                |       СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма)          КАК
СуммаВДокументе
                |
                |ИЗ
```

Установите курсор внутрь текста запроса (например, на слове ВЫБРАТЬ) и выполните команду контекстного меню **Конструктор запроса**.

Чтобы результат запроса поместить во временную таблицу, перейдите на закладку **Дополнительно** и отметьте пункт **Создание временной таблицы**. Задайте ей имя – **НоменклатураДокумента**. Нажмите ОК.



Конструктор сформировал текст запроса с одной новой строкой:

```
        "ВЫБРАТЬ
                |       ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
                |
        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры      КАК
ВидНоменклатуры,
                |       СУММА(ОказаниеУслугиПереченьНоменклатуры.Количество)     КАК
КоличествоВДокументе,
                |       СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма)          КАК
СуммаВДокументе
                |ПОМЕСТИТЬ НоменклатураДокумента
                |ИЗ
                |       Документ.ОказаниеУслуги.ПереченьНоменклатуры            КАК
ОказаниеУслугиПереченьНоменклатуры
```

```
            |ГДЕ
            |       ОказаниеУслугиПеречеНьНоменклатуры.Ссылка = &Ссылка
            |
            |СГРУППИРОВАТЬ ПО
            |       ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура,
            |
      ОказаниеУслугиПеречеНьНоменклатуры.Номенклатура.ВидНоменклатуры";
```

Это означает, что результат запроса будет сохранен во временной таблице **НоменклатураДокумента**. Это был Запрос 1 на нашей схеме.

Теперь если мы для другого запроса укажем этот же самый менеджер временных таблиц МенеджерВТ, то в другом запросе мы сможем обратиться к данным этой временной таблицы.

Займемся вторым запросом.

Установите курсор на следующую строку после оператора `Результат = Запрос.Выполнить();` (именно здесь выполняется создание временной таблицы) и напишите заготовку будущего запроса:

```
Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос2.Текст = "";
```

Мы создали новый объект Запрос и назначили ему тот же самый менеджер временных таблиц, чтобы иметь возможность обращаться к созданной нами ранее временной таблице.

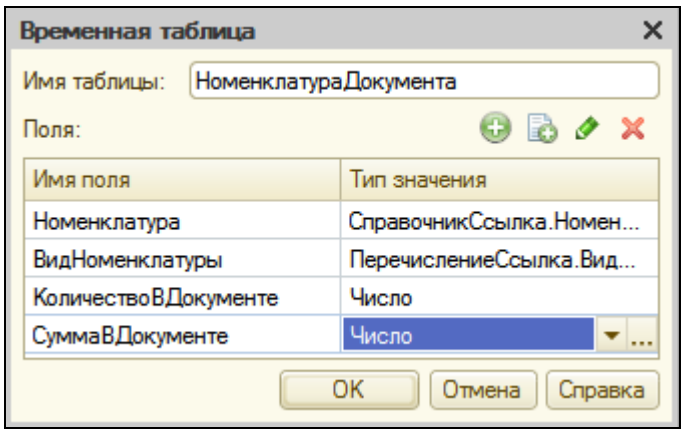
Установите курсор внутрь кавычек и выполните контекстную команду **Конструктор запроса**. Согласитесь на создание нового запроса.

Создадим описание временной таблицы в этом запросе. Для этого над списком Таблицы нажмите **Создать описание временной таблицы**  .

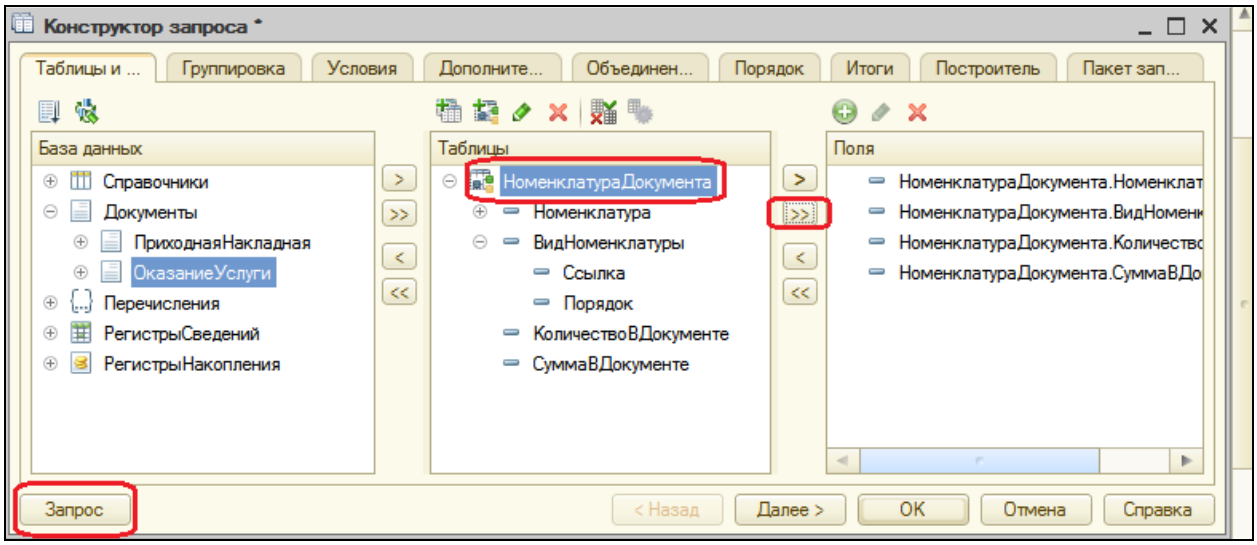
Введите имя нашей временной таблицы **НоменклатураДокумента** и добавьте поля:

- Номенклатура, тип СправочникСсылка.Номенклатура;
- ВидНоменклатуры, тип ПеречислениеСсылка.ВидыНоменклатуры;
- КоличествоВДокументе, тип Число, 15, 3;
- СуммаВДокументе, тип Число, 15, 2.

Нажмите ОК.



Выберите из **НоменклатураДокумента** все поля и нажмите кнопку **Запрос**.

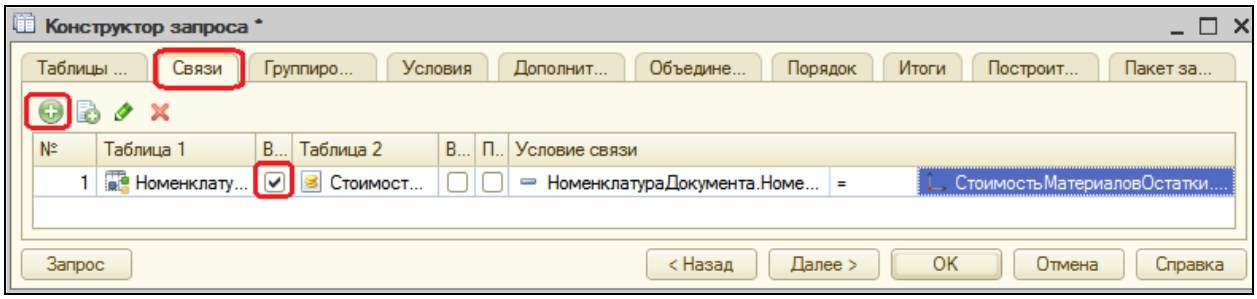



```
ВЫБРАТЬ
	НоменклатураДокумента.Номенклатура,
	НоменклатураДокумента.ВидНоменклатуры,
	НоменклатураДокумента.КоличествоВДокументе,
	НоменклатураДокумента.СуммаВДокументе
ИЗ
	НоменклатураДокумента КАК НоменклатураДокумента
```

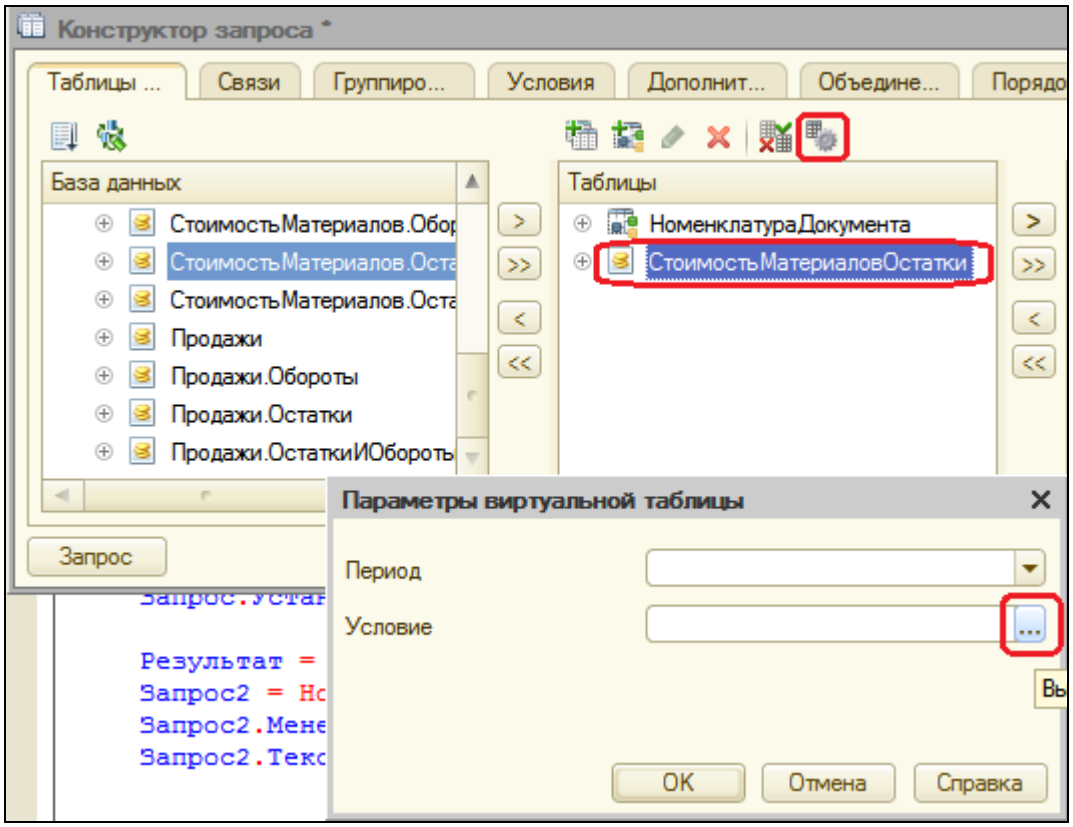
Итак, мы создали первую часть второго запроса – выбрали информацию из временной таблицы. Теперь будем соединять эту конструкцию левыми соединениями с таблицами остатков.

Добавим в список таблиц запроса виртуальную таблицу **РегистрНакопления.СтоимостьМатериалов.Остатки**. Из нее выберем поле **СтоимостьОстаток**. Перейдем на закладку **Связи** и зададим связь между таблицами.

Из временной таблицы будем выбирать все записи, и поле **Номенклатура** временной таблицы должно быть равно полю **Материал** таблицы остатков.



Ограничим виртуальную таблицу только нужной номенклатурой. Для этого вернемся на вкладку таблицы и поля, выделите в списке таблицу **СтоимостьМатериаловОстатки** и нажмите кнопку  **Параметры виртуальной таблицы**.



Задайте условие:

```
Материал В (ВЫБРАТЬ НоменклатураДокумента.Номенклатура ИЗ НоменклатураДокумента)
```

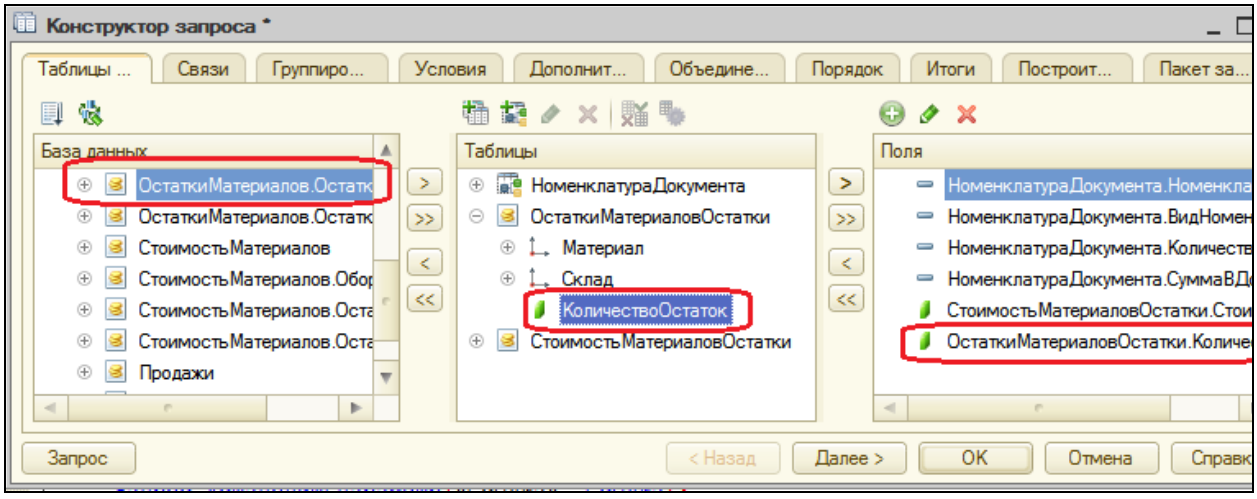
Т.е. материал должен быть среди номенклатуры, выбранной из временной таблицы. Нажмите ОК.

Теперь нажмите кнопку **Запрос** и посмотрите на текст, сформированный конструктором.

```
ВЫБРАТЬ
	НоменклатураДокумента.Номенклатура,
	НоменклатураДокумента.ВидНоменклатуры,
	НоменклатураДокумента.КоличествоВДокументе,
	НоменклатураДокумента.СуммаВДокументе,
	СтоимостьМатериаловОстатки.СтоимостьОстаток
ИЗ
	НоменклатураДокумента КАК НоменклатураДокумента
		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СтоимостьМатериалов.Остатки(
				,
				Материал В
					(ВЫБРАТЬ
						НоменклатураДокумента.Номенклатура
					ИЗ
						НоменклатураДокумента)) КАК
СтоимостьМатериаловОстатки
		ПО НоменклатураДокумента.Номенклатура =
СтоимостьМатериаловОстатки.Материал
```

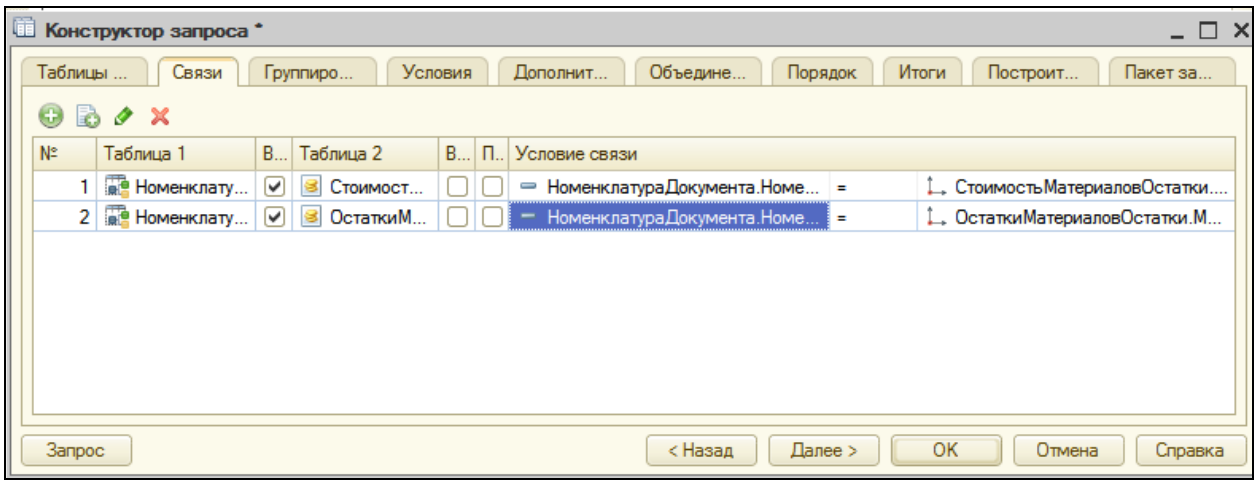
Тем самым мы добавили к выбранным ранее полям стоимость номенклатуры.

Теперь добавим виртуальную таблицу остатков регистра **ОстаткиМатериалов.Остатки**, из которой выберем поле **Количество.Остаток**.



Перейдите на закладку **Связи** и задайте связь между таблицами.

Из временной таблицы будем выбирать все записи, поле **Номенклатура** временной таблицы должно быть равно полю **Материал** таблицы остатков.



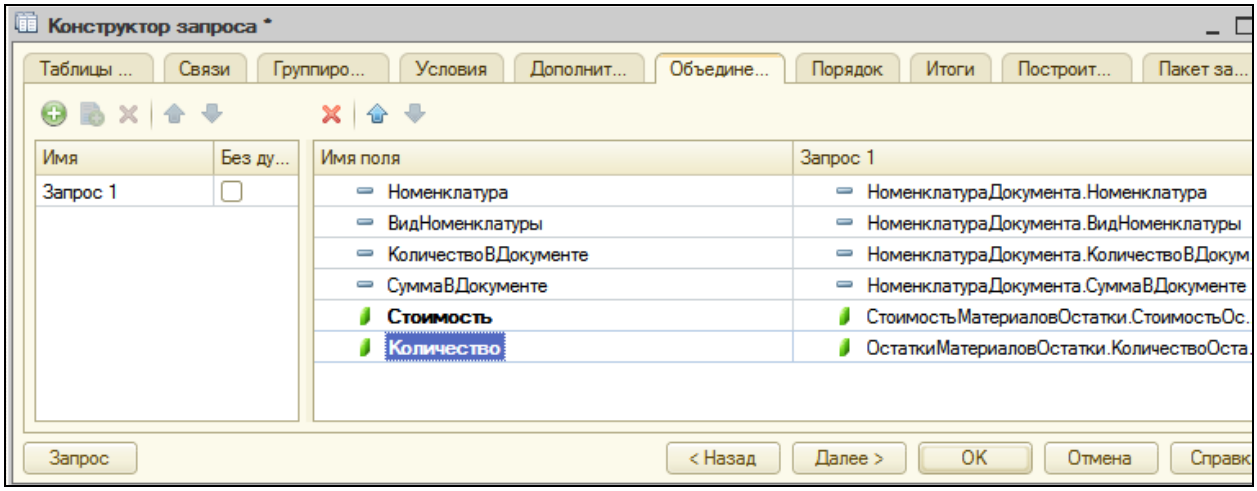
Также зададим параметры виртуальной таблицы **ОстаткиМатериаловОстатки**. В параметр Условие введите:

```
Материал В (ВЫБРАТЬ НоменклатураДокумента.Номенклатура ИЗ НоменклатураДокумента)
```

Тем самым мы добавили к выбранным ранее полям остатки номенклатуры на всех складах.

В заключение перейдем на закладку **Объединения/Псевдонимы** и зададим следующие псевдонимы полей:

- **СтоимостьОстаток** – **Стоимость**;
- **КоличествоОстаток** – **Количество**.



В результате получим следующий текст запроса:

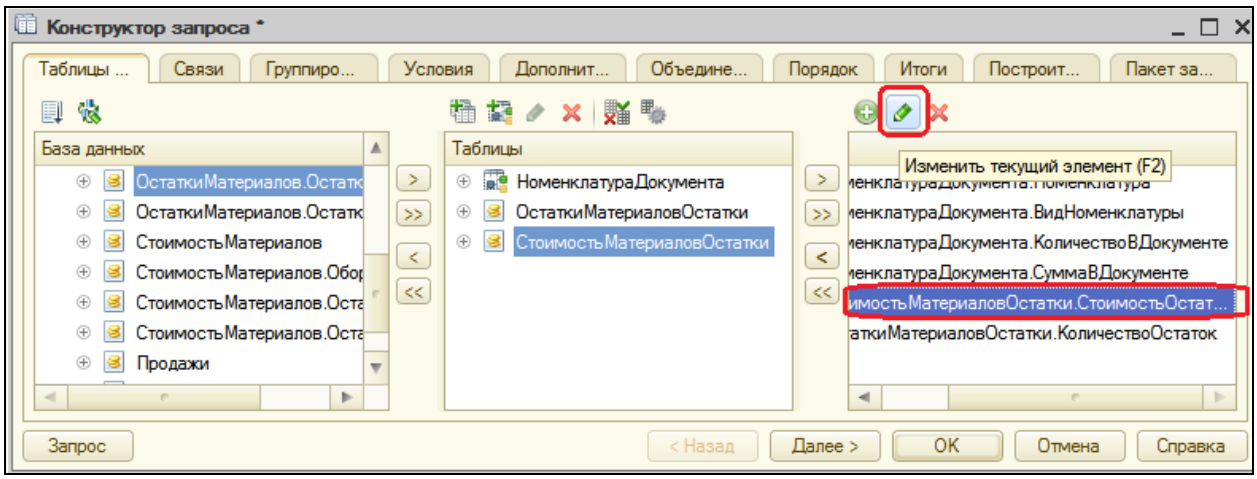
```
ВЫБРАТЬ
	НоменклатураДокумента.Номенклатура,
	НоменклатураДокумента.ВидНоменклатуры,
	НоменклатураДокумента.КоличествоВДокументе,
	НоменклатураДокумента.СуммаВДокументе,
	СтоимостьМатериаловОстатки.СтоимостьОстаток КАК Стоимость,
	ОстаткиМатериаловОстатки.КоличествоОстаток КАК Количество
ИЗ
	НоменклатураДокумента КАК НоменклатураДокумента
		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СтоимостьМатериалов.Остатки(
				,
				Материал В
					(ВЫБРАТЬ
						НоменклатураДокумента.Номенклатура
					ИЗ
						НоменклатураДокумента))		КАК
СтоимостьМатериаловОстатки
		ПО		НоменклатураДокумента.Номенклатура		=
СтоимостьМатериаловОстатки.Материал
		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиМатериалов.Остатки(
				,
				Материал В
					(ВЫБРАТЬ
						НоменклатураДокумента.Номенклатура
					ИЗ
						НоменклатураДокумента))		КАК
ОстаткиМатериаловОстатки
		ПО		НоменклатураДокумента.Номенклатура		=
ОстаткиМатериаловОстатки.Материал
```

Нужно предусмотреть тот случай, когда номенклатура в справочнике есть, но у нее нет ни остатков, ни стоимости. Это может быть, например, когда номенклатуру создали в справочнике, но она еще не поступала в нашу фирму.

В такой ситуации левые соединения с виртуальными таблицами не вернут ничего. На языке запросов это значит, что в полях **Стоимость** и **Количество** будут значения **NULL**.

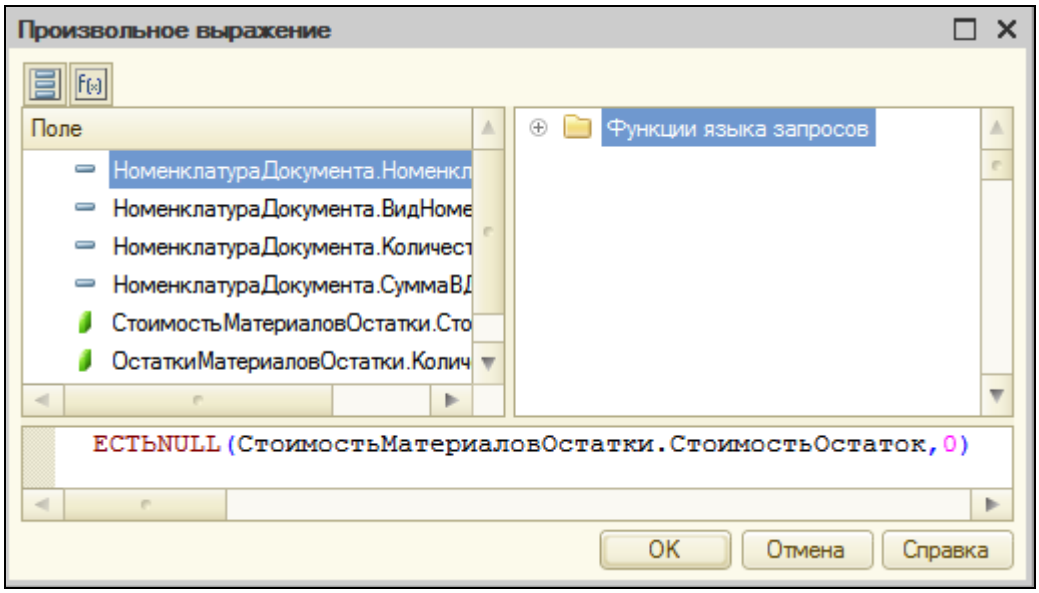
Избавимся в самом запросе от этих значений. Для этого мы применим функцию **ЕСТЬNULL()** к полям **Стоимость** и **Количество**. Если значение этого поля будет **NULL**, функция вернет **0**. В остальных случаях функция вернет само значение этого поля.

Перейдите на закладку **Таблицы и поля**, выделите **СтоимостьМатериаловОстатки.СтоимостьОстаток** и нажмите кнопку  **Изменить текущий элемент**.



Отредактируем значение поля:

```
ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток,0)
```



Аналогично с другим полем – **ОстаткиМатериаловОстатки.КоличествоОстаток**.

```
ЕСТЬNULL(ОстаткиМатериаловОстатки.КоличествоОстаток,0)
```

Нажмите ОК – текст запроса будет вставлен в модуль. Останется всего лишь дописать после него оператор выполнения запроса:

```
	Запрос2 = Новый Запрос;
	Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
	Запрос2.Текст = "ВЫБРАТЬ
			|	НоменклатураДокумента.Номенклатура,
			|	НоменклатураДокумента.ВидНоменклатуры,
			|	НоменклатураДокумента.КоличествоВДокументе,
			|	НоменклатураДокумента.СуммаВДокументе,
			|	ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток, 0) КАК
Стоимость,
```

```
                |  ЕСТЬNULL(ОстаткиМатериаловОстатки.КоличествоОстаток, 0) КАК
Количество
                |ИЗ
                |  НоменклатураДокумента КАК НоменклатураДокумента
                |        ЛЕВОЕ                                          СОЕДИНЕНИЕ
РегистрНакопления.СтоимостьМатериалов.Остатки(
                |                ,
                |                Материал В
                |                        (ВЫБРАТЬ
                |
      НоменклатураДокумента.Номенклатура
                |                        ИЗ
                |                                НоменклатураДокумента))     КАК
СтоимостьМатериаловОстатки
                |        ПО        НоменклатураДокумента.Номенклатура          =
СтоимостьМатериаловОстатки.Материал
                |        ЛЕВОЕ                                          СОЕДИНЕНИЕ
РегистрНакопления.ОстаткиМатериалов.Остатки(
                |                ,
                |                Материал В
                |                        (ВЫБРАТЬ
                |
      НоменклатураДокумента.Номенклатура
                |                        ИЗ
                |                                НоменклатураДокумента))     КАК
ОстаткиМатериаловОстатки
                |        ПО        НоменклатураДокумента.Номенклатура          =
ОстаткиМатериаловОстатки.Материал";
      Результат = Запрос2.Выполнить();

      ВыборкаДетальныеЗаписи = Результат.Выбрать();
```

Теперь разберемся с записью движений. Все операторы, которые были написаны нами ранее, будут работать без изменений.

Единственное, что потребуется изменить – способ получения стоимости. Раньше мы просто брали ее из документа, теперь же нам нужно ее рассчитать на основании данных запроса. Стоимость материала равна частному от деления всей стоимости, полученной запросом (**Стоимость**) на общее количество материала на всех складах (**Количество**).

Но, как сказано ранее, **Количество** может быть равно 0, а на ноль делить нельзя. Поэтому сразу после начала цикла обхода результата запроса рассчитаем стоимость для текущей номенклатуры.

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
            Если ВыборкаДетальныеЗаписи.Количество = 0 Тогда
                  СтоимостьМатериала = 0;
            Иначе
                  СтоимостьМатериала                                          =
ВыборкаДетальныеЗаписи.Стоимость/ВыборкаДетальныеЗаписи.Количество;
            КонецЕсли;
```

```
    Если ВыборкаДетальныеЗаписи.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал Тогда
```

Теперь заменим расчет стоимости в движениях регистров **СтоимостьМатериалов** и **Продажи**.

```
    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
        Если ВыборкаДетальныеЗаписи.Количество = 0 Тогда
            СтоимостьМатериала = 0;
        Иначе
            СтоимостьМатериала = ВыборкаДетальныеЗаписи.Стоимость/ВыборкаДетальныеЗаписи.Количество;
        КонецЕсли;
            Если ВыборкаДетальныеЗаписи.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал Тогда

            // регистр ОстаткиМатериалов Расход
            Движение = Движения.ОстаткиМатериалов.Добавить();
            Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
            Движение.Период = Дата;
            Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
            Движение.Склад = Склад;
            Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

            // регистр СтоимостьМатериалов Расход
            Движение = Движения.СтоимостьМатериалов.Добавить();
            Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
            Движение.Период = Дата;
            Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
            Движение.Стоимость = ВыборкаДетальныеЗаписи.КоличествоВДокументе*СтоимостьМатериала;
        КонецЕсли;
    // Регистр Продажи
        Движение = Движения.Продажи.Добавить();
        Движение.Период = Дата;
        Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
        Движение.Клиент = Клиент;
        Движение.Мастер = Мастер;
        Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;
        Движение.Выручка = ВыборкаДетальныеЗаписи.СуммаВДокументе;
        Движение.Стоимость = СтоимостьМатериала * ВыборкаДетальныеЗаписи.КоличествоВДокументе;

    КонецЦикла;
```

Теперь, если проводить этот документ в первый раз, результат получится правильный. Но, если документ был проведен ранее и мы заново решим его провести, получим неправильный результат. Дело в том, что когда мы находимся в обработчике проведения и этот документ был проведен ранее, то в БД существуют движения этого документа.

Таким образом, читая из базы стоимость и остатки материалов, мы прочитаем их с учетом движений документа. А это неправильно.

Чтобы в обработчике проведения документа прочитать данные без учета предыдущих движений, которые мог выполнять документ, нужно перед чтением записать пустые наборы записей в те регистры, из которых мы собираемся читать. В нашем случае это **СтоимостьМатериалов** и **ОстаткиМатериалов**.

Перед выполнением второго запроса добавим строки:

```
                | ПО НоменклатураДокумента.Номенклатура = ОстаткиМатериаловОстатки.Материал";

            //Запишем пустые наборы записей, чтобы читать остатки без учета данных в документе.
            Движения.СтоимостьМатериалов.Записать();
            Движения.ОстаткиМатериалов.Записать();

        Результат = Запрос2.Выполнить();
```

Запустите режим отладки и перепроведите все документы **Оказание услуги**, проверьте правильность занесения данных в регистры.

Оперативное и неоперативное проведение документов

Оперативное проведение документов пользователями выполняется в режиме реального времени, т.е. отображает изменения, факты, свершающиеся в настоящее время. Оперативное проведение особенно актуально при многопользовательской работе. Поэтому при этом способе проведения следует осуществлять максимум проверок, способных исключить ошибки при вводе данных пользователями.

Например, при оперативном проведении следует выполнять контроль остатков на складе списываемой номенклатуры, чтобы исключить одновременную продажу одного товара несколькими продавцами.

Оперативность или неоперативность проведения документа определяется по его дате. Если дата проводимого документа совпадает с текущей, то система будет проводить такой документ в оперативном режиме, не задавая вопросов, и в обработке проведения об этом можно узнать, чтобы выстроить определенный алгоритм проведения.

Если дата документа меньше текущей, то такой документ система будет проводить в неоперативном режиме. При этом перед проведением она напомнит, что оперативно она его провести не может (вдруг пользователь ошибся). И если пользователь подтвердит, что хочет провести его неоперативно, то система проведет документ неоперативно.

Неоперативное проведение подразумевает отражение в базе данных фактов, которые свершились в прошлом или которые точно будут совершены в будущем. Поэтому задача неоперативного проведения документов – просто отразить в информационной базе данные о совершенных операциях.

При неоперативном проведении не имеет смысла производить целый ряд проверок, в частности – контроль остатков. Подразумевается, что если в процессе неоперативного проведения были допущены ошибки, то анализ полученного состояния БД является отдельной задачей, не относящейся к неоперативному проведению и выполняющейся не в момент проведения документа.

Если логика учета подразумевает, что какой-то документ должен проводиться будущей датой, для такого документа механизм оперативного проведения должен быть отключен в метаданных (на закладке **Движения** окна редактирования объекта конфигурации).

Контроль остатков

Общая методика контроля остатков при проведении документа заключается в записи движения документа, а затем чтения из базы остатков. Если появились отрицательные остатки, значит такой документ проводить нельзя. Нужно сообщить пользователю каких материалов не хватает и отменить проведение документа. Нам осталось проконтролировать это.

В режиме Конфигуратор

Сделаем заготовку. После цикла обхода результата запроса и перед концом процедуры напишем:

```
КонецЦикла;

        Движения.Записать();
        Если Режим = РежимПроведенияДокумента.Оперативный Тогда
                //проверить отрицательные остатки.
```

```
        КонецЕсли;


КонецПроцедуры
```

Сначала мы записываем движения в регистры. Затем определяем режим проведения документа. При выполнении процедуры **ОбработкаПроведения()** вторым параметром (**Режим**) в нее передается режим проведения документа и значение этой переменной сравнивается со значением системного перечисления **РежимПроведенияДокумента**. В случае оперативного проведения мы будем выполнять контроль остатков.

Теперь сделаем заготовку запроса для проверки отрицательных остатков. Используем тот же менеджер виртуальных таблиц.

```
Если Режим = РежимПроведенияДокумента.Оперативный Тогда
                //проверить отрицательные остатки.
                Запрос3 = Новый Запрос;
                Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
                Запрос3.Текст = "";
        КонецЕсли;
```

Установите курсор внутрь кавычек и вызовите конструктор запроса. Подтвердите создание нового запроса.

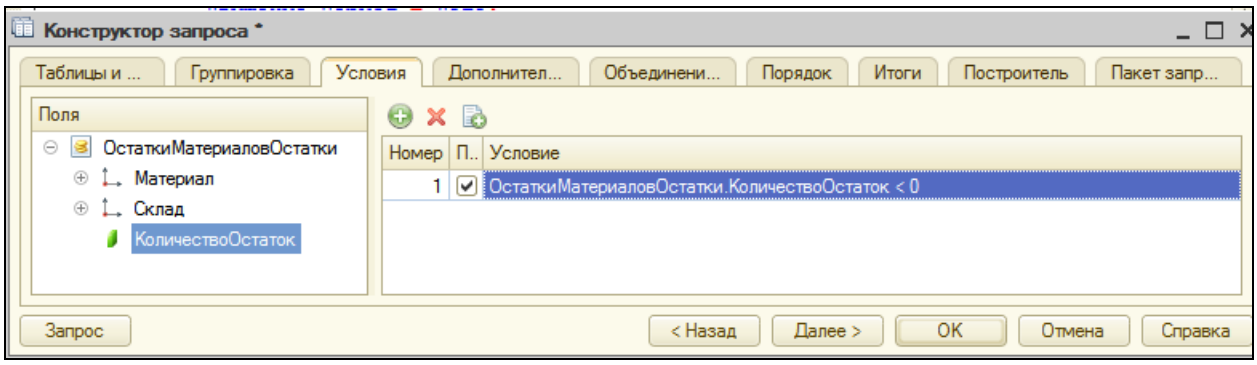
Выберите таблицу **ОстаткиМатериалов.Остатки** и из нее два поля: **Материал** и **КоличествоОстаток**. Задайте параметры этой таблице. В параметре **Условие** напишем:

```
Материал В (ВЫБРАТЬ НоменклатураДокумента.Номенклатура ИЗ НоменклатураДокумента) И Склад = &Склад
```

Т.е. мы получаем итоги только для той номенклатуры, которая содержится в нашей временной таблице и только по складу, который указан в документе.

Затем на вкладке **Условия** перенесем в список условий поле **ОстаткиМатериаловОстатки.КоличествоОстаток**, поставим флажок **Произвольное** и укажем, что нас интересуют только отрицательные остатки:

```
ОстаткиМатериаловОстатки.КоличествоОстаток < 0
```



Нажмите ОК.

Теперь осталось только установить параметр запроса, обойти результат запроса и вывести сообщения об отрицательных остатках.

```
//проверить отрицательные остатки.
		Запрос3 = Новый Запрос;
		Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
		Запрос3.Текст = "ВЫБРАТЬ
				|	ОстаткиМатериаловОстатки.Материал,
				|	ОстаткиМатериаловОстатки.КоличествоОстаток
				|ИЗ
				|	РегистрНакопления.ОстаткиМатериалов.Остатки(
				|			,
				|			Материал В
				|				(ВЫБРАТЬ
				|
	НоменклатураДокумента.Номенклатура
				|				ИЗ
				|					НоменклатураДокумента)
				|			И	Склад	=	&Склад)	КАК
ОстаткиМатериаловОстатки
				|ГДЕ
				|	ОстаткиМатериаловОстатки.КоличествоОстаток < 0";
		Запрос3.УстановитьПараметр("Склад", Склад);
		Результат = Запрос3.Выполнить();
		ВыборкаДетальныеЗаписи = Результат.Выбрать();

		Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
			Сообщение = Новый СообщениеПользователю();
			Сообщение.Текст	=	"Не	хватает"	+	Строка(-
ВыборкаДетальныеЗаписи.КоличествоОстаток)	+	"единиц	материала"""	+
ВыборкаДетальныеЗаписи.Материал + """";
			Сообщение.Сообщить();
			Отказ = Истина;
		КонецЦикла;
	КонецЕсли;


КонецПроцедуры
```

При выполнении проверки в запрос в параметре **Склад** передается склад, указанный в документе. Затем выполняется запрос для получения

отрицательных остатков номенклатуры, содержащейся во временной таблице и на складе, указанном в параметре **Склад**.

После этого выборка записей запроса обходится в цикле, и если есть такие записи, они выводятся в сообщениях пользователю.

При этом параметру **Отказ** присваивается значение **Истина**, т.е. документ не проводится, начатая транзакция отменяется, состояние данных, измененных в процессе проведения, возвращается в исходное, как до начала проведения документа.

Блокировка данных, которые читаются и изменяются при проведении

Сейчас схема нашей процедуры такова:

1. Выполняем первый запрос с именем **Запрос**. В результате мы формируем временную таблицу из перечня номенклатуры документа.
2. Выполняем второй запрос с именем **Запрос2**. В результате мы читаем стоимость и остатки для номенклатуры, содержащейся в табличной части документа.
3. Записываем движения регистров (**Движения.Записать()**)
4. Выполняем третий запрос **Запрос3**. Тем самым мы проверяем наличие отрицательных остатков.

Обратите внимание, что, начиная с выполнения второго запроса и до конца процедуры, нам необходимо обеспечить неизменность стоимости и остатков номенклатуры, с которой мы работаем, и запретить другим транзакциям даже читать эти данные. Сама система заблокирует изменение этих данных, но лишь начиная с момента записи движения.

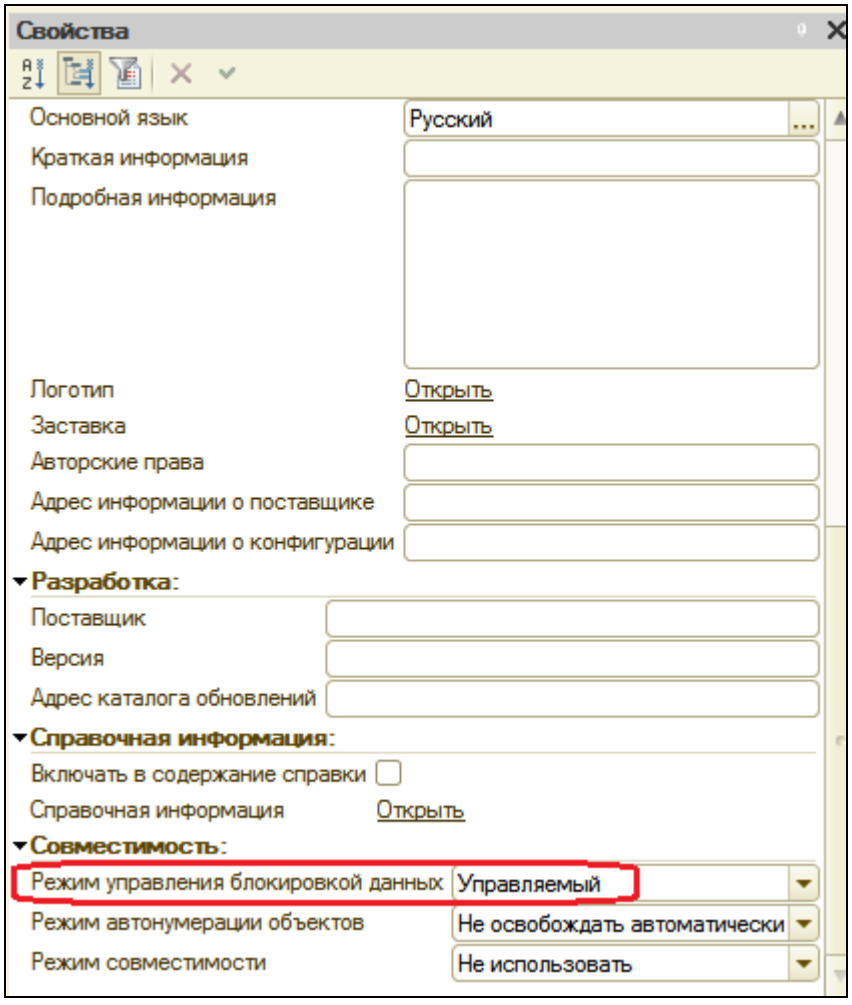
Однако может возникнуть следующая ситуация. Выполняя второй запрос, мы прочитали, что есть 2 шт. некоторого материала. И другая транзакция (другой пользователь), которая собирается списывать материалы, тоже прочитала, что есть 2 шт. этого материала. После этого мы записали движения, система заблокировала эти данные. Другая транзакция ждет, когда мы освободим данные. Мы провели документ, списали 2 шт. материала и освободили данные. Другая транзакция пытается тоже списать 2 шт. материала, но его уже нет!

Аналогичная ситуация может возникнуть и между пунктами 3 и 4, в результате чего контроль остатков будет работать неверно.

Поэтому необходимо заблокировать остатки от чтения другими транзакциями еще до выполнения второго запроса. Т.е. прежде чем читать что-то, что мы собираемся изменять, нужно запретить чтение этих данных другими транзакциями до окончания введения изменений нами.

В режиме Конфигуратор

Как это сделать? Давайте посмотрим на свойство нашей конфигурации **Режим управления блокировкой данных**. Он установлен в значение **Управляемый**.



Это значит, что нам нужно использовать управляемые блокировки, которые устанавливаются средствами встроенного языка.

Необходимо заблокировать те данные, которые мы собираемся читать и впоследствии изменять. Для этого у наборов записей регистров есть

свойство **БлокироватьДляИзменения**. Вставим этот код перед записью пустых наборов записей.

```
ПО НоменклатураДокумента.Номенклатура = ОстаткиМатериаловОстатки.Материал";

                                    //Установим необходимость блокировки данных в регистрах СтоимостьМатериалов и ОстаткиМатериалов.

        Движения.СтоимостьМатериалов.БлокироватьДляИзменения = Истина;

        Движения.ОстаткиМатериалов.БлокироватьДляИзменения = Истина;

                                    //Запишем пустые наборы записей, чтобы читать остатки без учета данных в документе.
                                    Движения.СтоимостьМатериалов.Записать();
                                    Движения.ОстаткиМатериалов.Записать();

        Результат = Запрос2.Выполнить();
```

Управляемая блокировка будет установлена в момент записи этих наборов записей, т.е. как раз перед выполнением второго запроса. Что и требовалось.

В режиме 1С: Предприятие

Запустите режим отладки и проверьте работу нового обработчика события **ОбработкаПроведения**, перепроведя все документы Оказание услуги.

Контрольные вопросы

- ✓ Почему для доступа к массивам данных информационной базы предпочтительнее использовать запросы.
- ✓ Чем отличается оперативное проведение документов от неоперативного.
- ✓ Почему при неоперативном проведении документов не нужно контролировать остатки.
- ✓ Что такое временные таблицы и зачем их использовать.
- ✓ Что такое менеджер запросов.
- ✓ Как программно блокировать данные.

Практическая работа № 14

План видов характеристик (2:50)

В этой работе мы познакомимся с новым объектом конфигурации – План видов характеристик - и узнаем, как можно использовать этот объект для расширения возможностей нашей конфигурации.

Задача работы: создать механизм, позволяющий пользователю произвольно описывать материалы и вести учет в разрезе всех описаний, которые могут быть заданы пользователем.

Что такое план видов характеристик

Объект конфигурации план видов характеристик предназначен для описания структуры хранения информации о характеристиках, создаваемых пользователем.

Он напоминает справочник, но с более узкой специализацией – хранит только виды характеристик объекта БД.

План видов характеристик состоит из видов характеристик, описываемых наименованием и типом значения.

Разработчик и пользователь могут задать в нем любое необходимое количество видов характеристик.

Для задания набора возможных типов значений, которые могут принимать виды характеристик, существует свойство **Тип значения характеристик**.

Если пользователю будет недостаточно определенных в данной конфигурации типов данных, он может воспользоваться специальным вспомогательным справочником, который разработчик создаст заблаговременно и укажет в качестве свойства объекта **План видов характеристик** – **Дополнительные значения характеристик**.

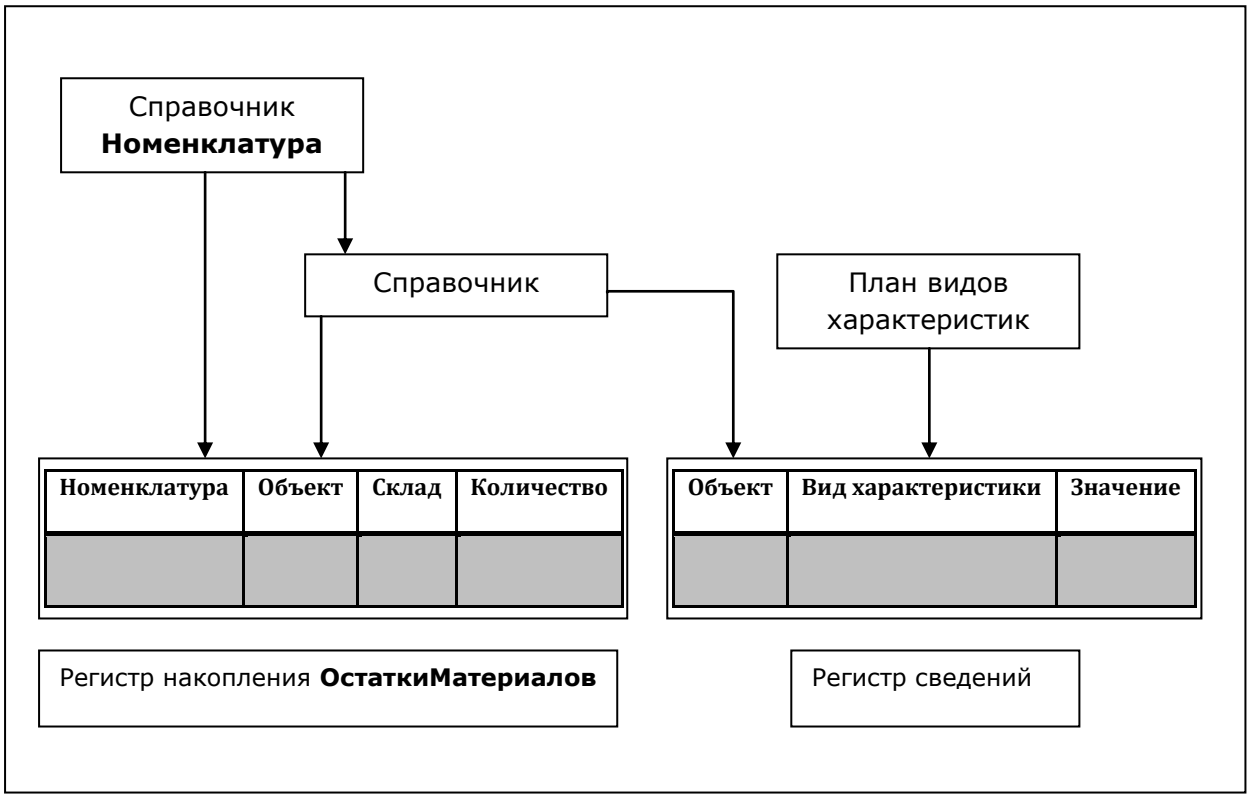
Логическая связь объектов

Для реализации этого примера нам понадобятся три новых объекта конфигурации.

Прежде всего, **План видов характеристик**, который будет хранить виды характеристик, описывающих материалы.

Второе, нам понадобится специальный справочник, подчиненный справочнику **Номенклатура**. Элементы этого справочника будут идентифицировать партии материалов с некоторым фиксированным набором значений характеристик.

Третье – регистр сведений, в котором и будет храниться соответствие конкретных значений характеристик некоторому варианту материала.



Создание новых объектов конфигурации

В режиме Конфигуратор

Нам понадобится создать несколько новых объектов:

- Справочник **ВариантыНоменклатуры**, чтобы описывать партии материалов.
- Справочник **ДополнительныеСвойстваНоменклатуры**, чтобы задавать значения видов характеристик, для которых нет подходящих типов в конфигурации.

- План видов характеристик **СвойстваНоменклатуры**, чтобы создавать виды характеристик.
- Регистр сведений **ЗначенияСвойствНоменклатуры**, чтобы хранить значения видов характеристик для различных партий материалов.

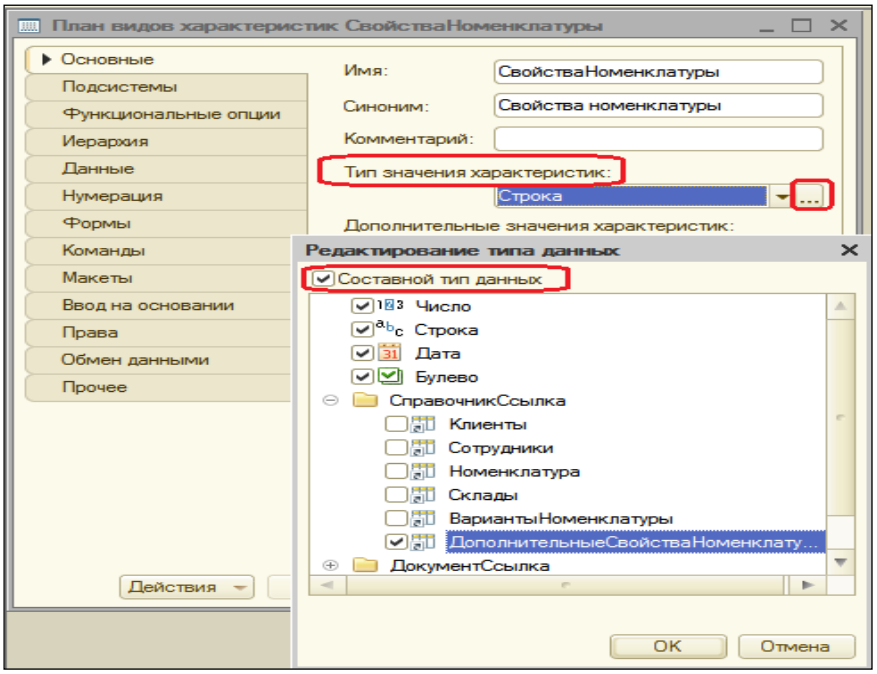
Сначала создадим справочник с именем **ВариантыНоменклатуры** и укажем, что он будет подчинен справочнику **Номенклатура**. Для этого добавим справочник **Номенклатура** на закладке **Владельцы** в список владельцев справочника **ВариантыНоменклатуры**.

Затем создадим еще один справочник с именем **ДополнительныеСвойстваНоменклатуры**.

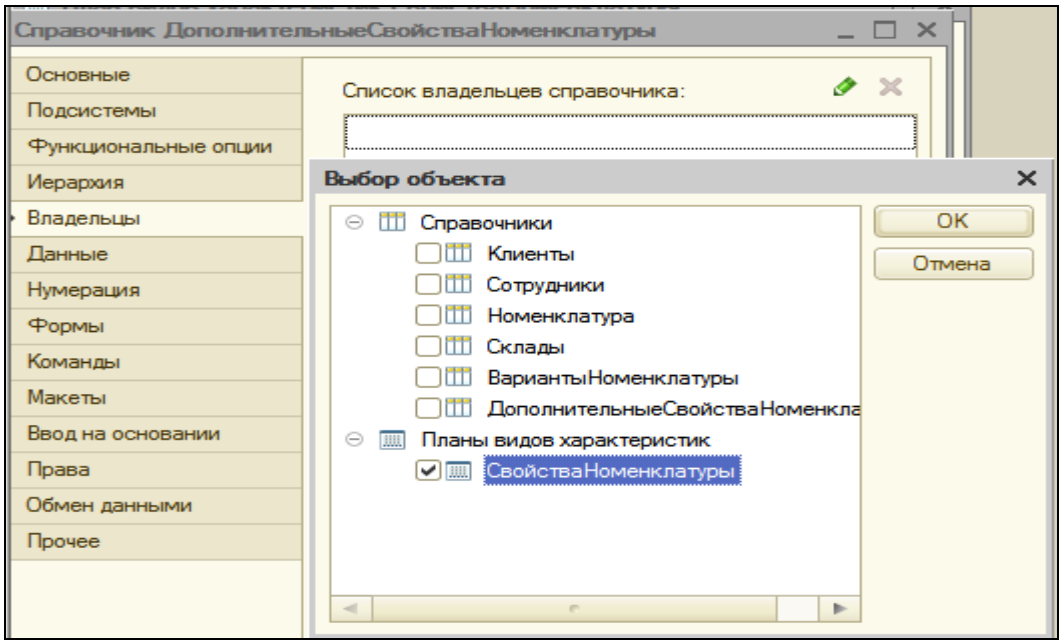
После этого создадим **План видов характеристик** с именем **СвойстваНоменклатуры**. Установим его свойство **Тип значения характеристик**.

Для этого нажмите кнопку выбора … и задайте составной тип данных:

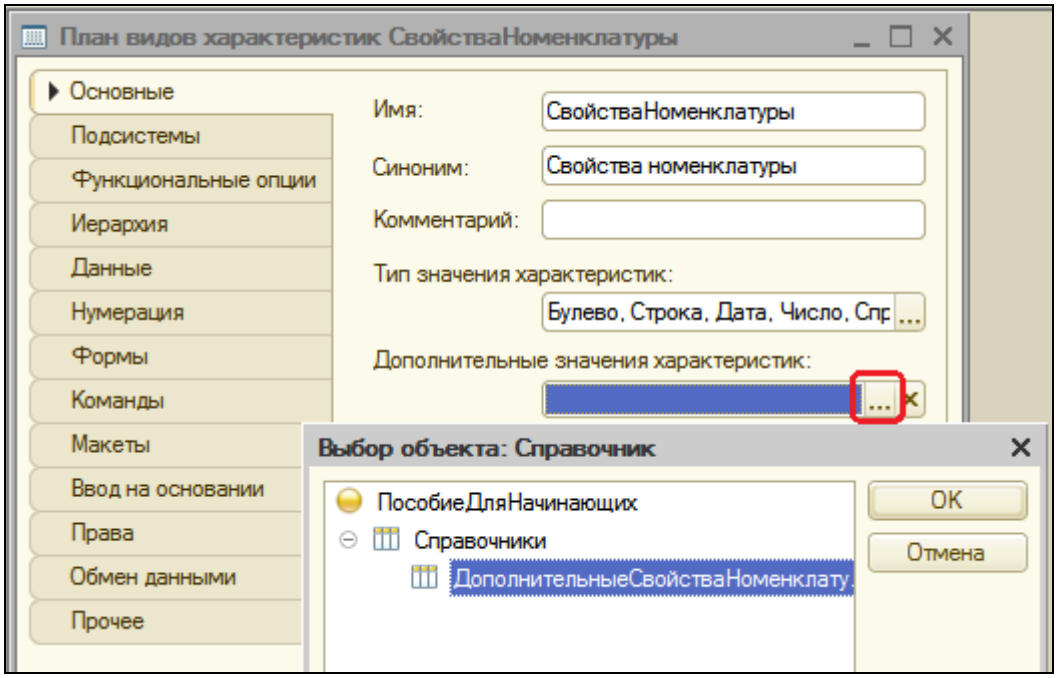
- Число, длина 15, точность 3;
- Строка, длина 25;
- Дата;
- Булево;
- СправочникСсылка.ДополнительныеСвойстваНоменклатуры.



Затем справочнику **ДополнительныеСвойстваНоменклатуры** укажем владельца – план видов характеристик **СвойстваНоменклатуры**.



После определим, что дополнительные значения характеристик плана видов характеристик будут располагаться в справочнике **ДополнительныеСвойстваНоменклатуры**.

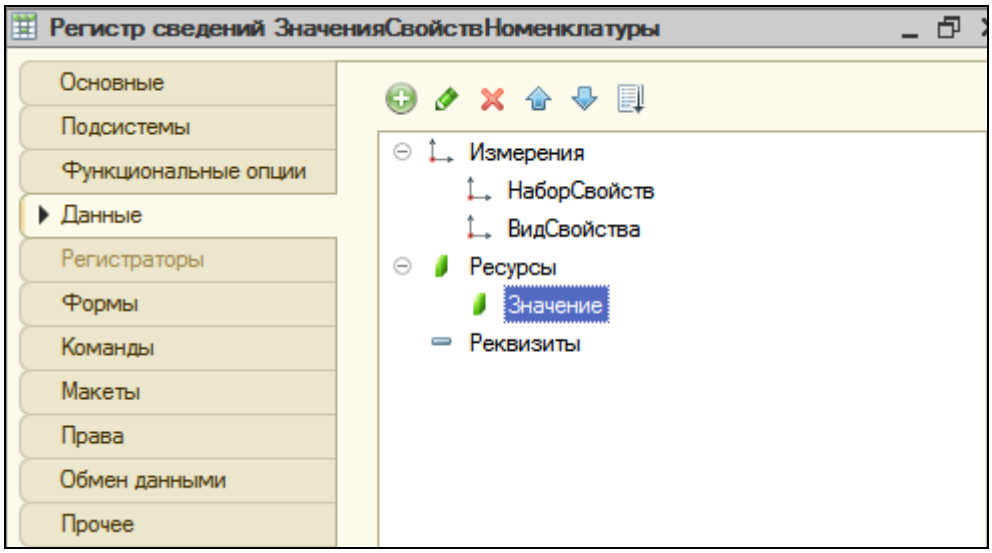


В заключение создадим объект Регистр сведений с именем **ЗначенияСвойствНоменклатуры**.

На закладке **Данные** создадим измерения регистра:

- **НаборСвойств**, **Ведущее**, тип **СправочникСсылка.ВариантыНоменклатуры**;
- **ВидСвойства**, тип **ПланВидовХарактеристикСсылка.СвойстваНоменклатуры**.

Затем создадим ресурс регистра: **Значение**, тип **Характеристика.СвойстваНоменклатуры**.

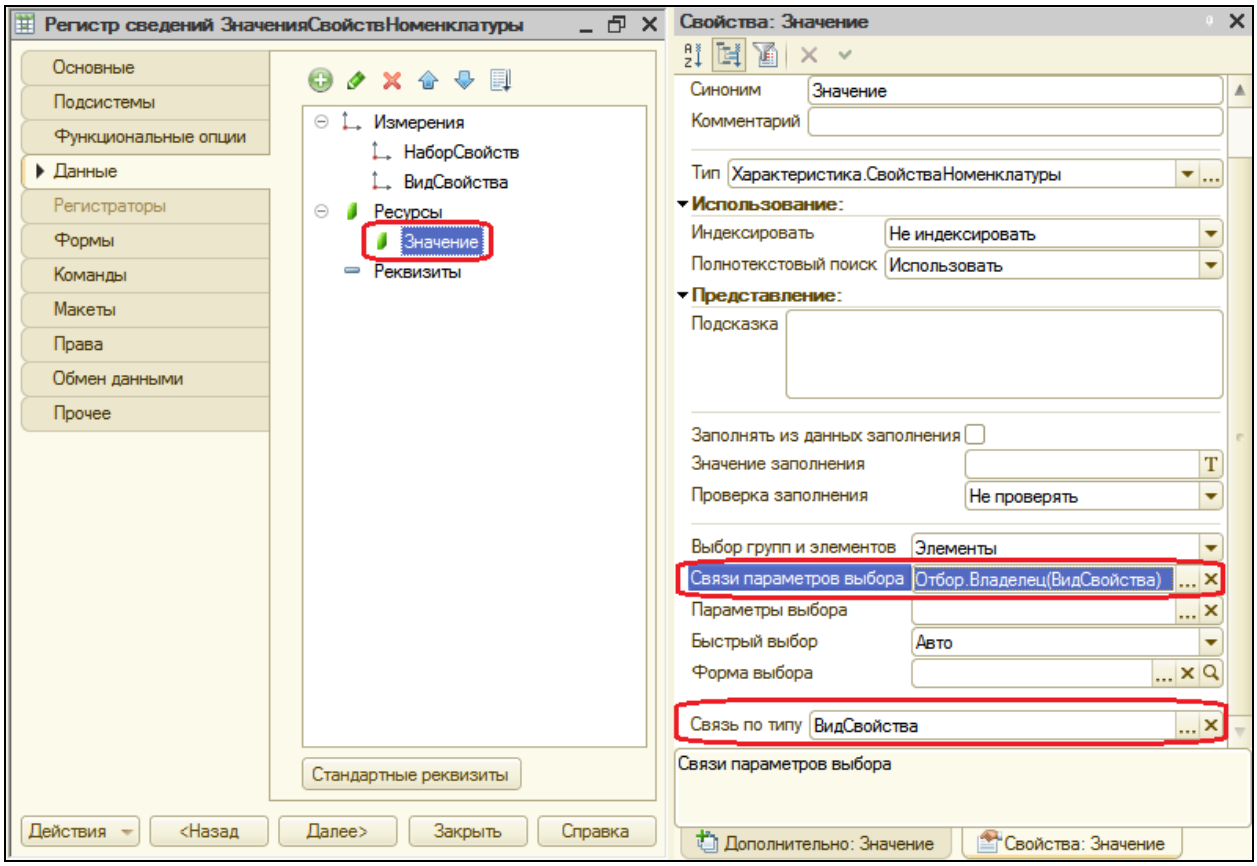


Обратите внимание, что мы имеем возможность определить тип значения ресурса регистра как Характеристика.<имя>. Это определение представляет собой составной тип данных, как он задан в типе значения соответствующего плана видов характеристик. Т.е. ресурс регистра может иметь значение любого типа из тех, которые описаны в типе значения плана видов характеристик.

Кроме этого, зададим в свойстве **Связь** по типу этого ресурса измерение регистра **ВидСвойства**. Связь по типу будет обеспечивать нам соответствие типа значений, вводимых в это поле, и типа характеристики, выбранной в поле **Вид свойства**.

А также заполним еще одно свойство – **Связи параметров выбора**. Для этого нажмите кнопку выбора … у этого свойства и перенесите из списка доступных реквизитов в список параметров измерение регистра **ВидСвойства**.

Установка свойства **Связи параметров выбора** обеспечит нам то, что при выборе значений, содержащихся в справочнике **Дополнительные свойства номенклатуры**, для выбора будут предлагаться только те значения, которые относятся к выбранной характеристике.



Доработка объектов конфигурации

Справочник Номенклатура

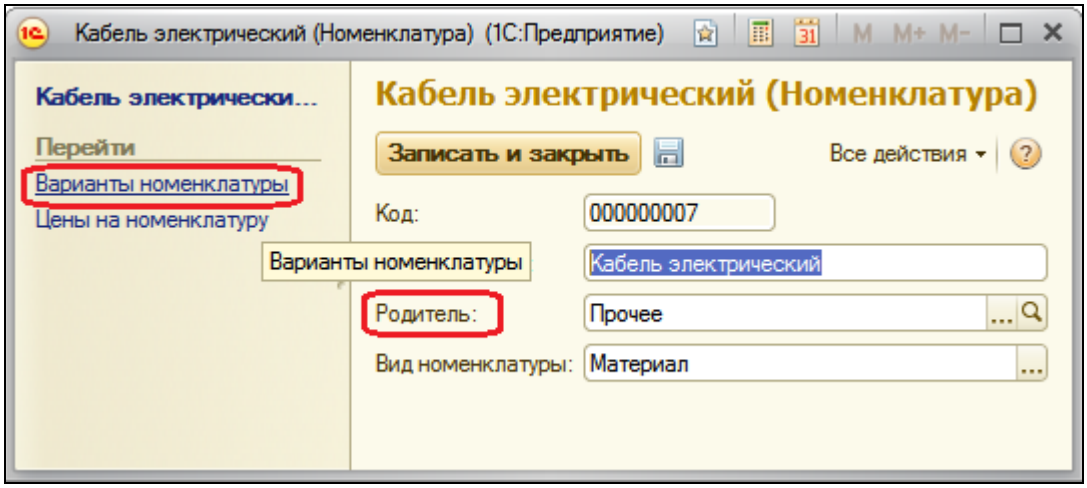
В режиме 1С:Предприятие

Запустите режим отладки и посмотрите, как взаимодействуют логически связанные объекты конфигурации Справочник **Номенклатура**, Справочник **ВариантыНоменклатуры**, План видов характеристик **СвойстваНоменклатуры** и Регистр сведений **ЗначенияСвойствНоменклатуры**.

Мы не указывали для этих объектов подсистем, к которым они относятся, потому что отображение этих объектов вне их логической связи не имеет особого смысла. Т.к. мы задали подчинения, то нужные объекты автоматически попадут в панели навигации форм своих владельцев.

Поэтому проигнорируйте системное сообщение об отсутствии привязки объектов к подсистемам.

В разделе **Учет материалов** откройте справочник **Номенклатура** и его элемент **Кабель электрический** из группы **Материалы - Прочее**.



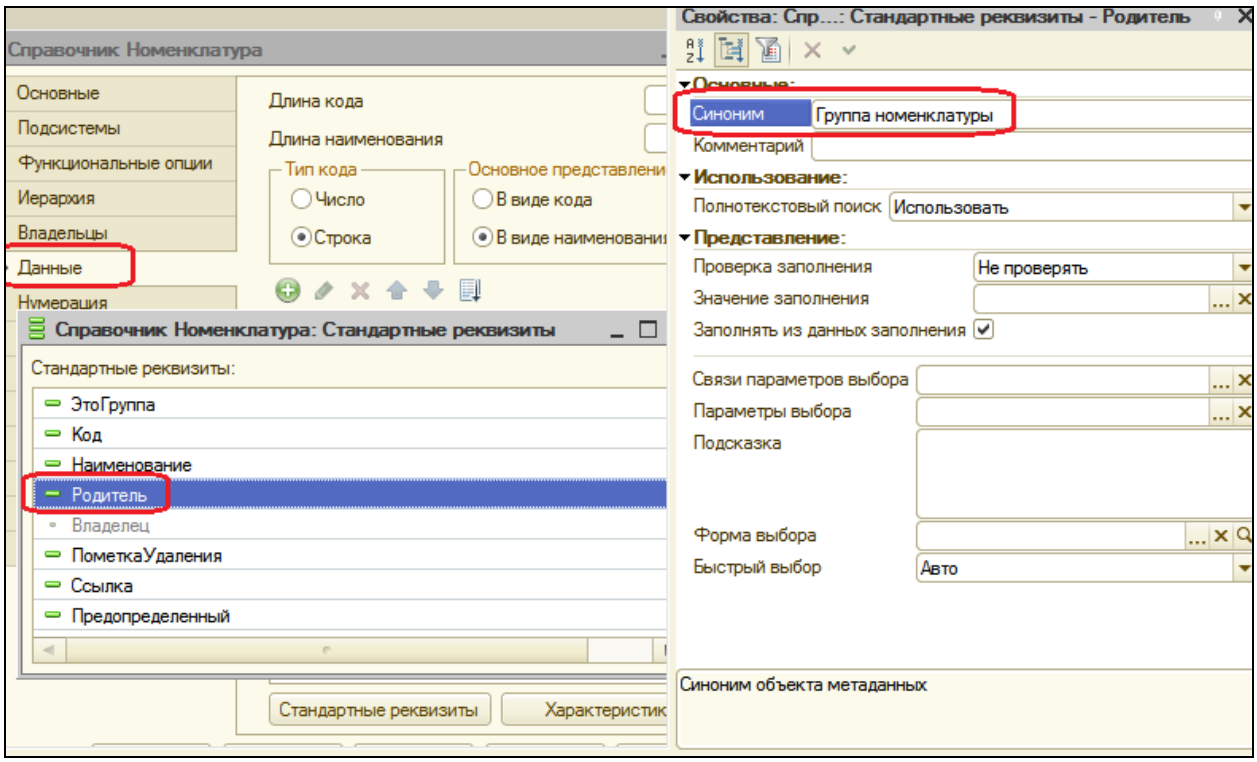
Поскольку справочник **Номенклатура** является владельцем справочника **ВариантыНоменклатуры**, мы видим в панели навигации ссылку для перехода к подчиненному списку. Это значит, что при открытии этого списка мы будем видеть только наборы свойств, относящиеся к редактируемому элементу справочника **Номенклатура**.

Это система сделала за нас, и нас это устраивает. Но название стандартного реквизита **Родитель** не совсем понятно. Более естественное название – **Группа номенклатуры**. Поскольку в интерфейсе отражаются синонимы объектов, нам нужно изменить синоним стандартного реквизита справочника, который по умолчанию совпадает с его именем **Родитель**.

В режиме Конфигуратор

Для этого вернемся в конфигуратор и откроем окно редактирования Справочника **Номенклатура**.

На закладке **Данные** нажмите кнопку **Стандартные реквизиты**, дважды щелкните на реквизите **Родитель** и в открывшемся окне задайте **Синоним** реквизита **Группа номенклатуры**.

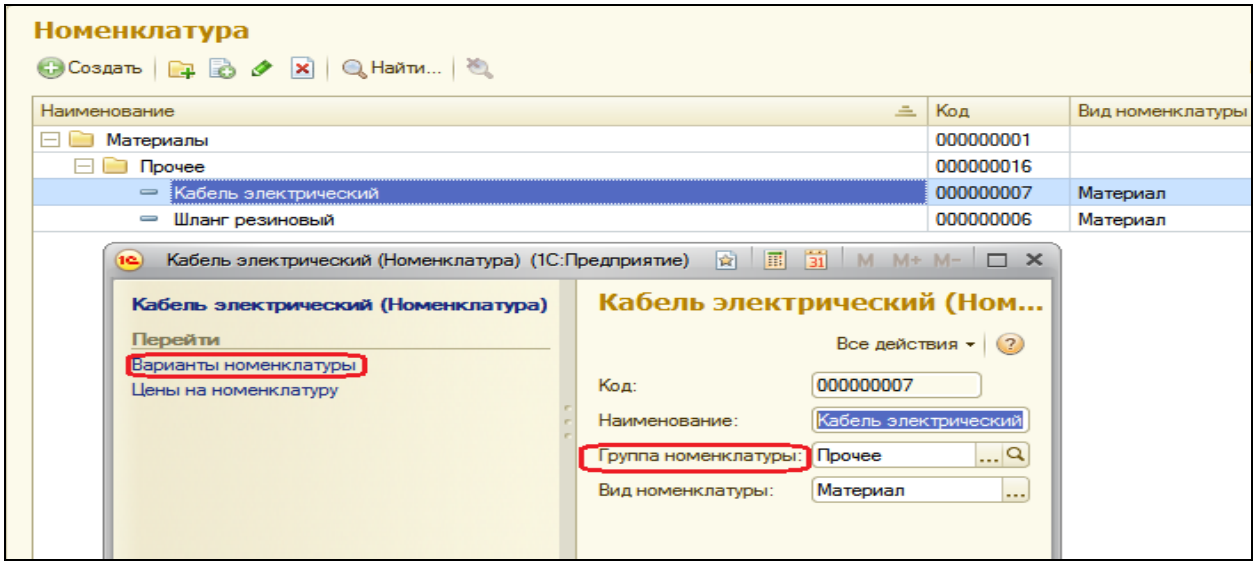


Мы изменили синоним реквизита объекта конфигурации, а не реквизита формы. В данном случае форма элемента справочника **Номенклатура** вообще сгенерирована системой автоматически.

Теперь во всех видах форм данный реквизит будет иметь установленный синоним, если только разработчик не захочет его изменить при создании своей собственной формы.

В режиме 1С:Предприятие

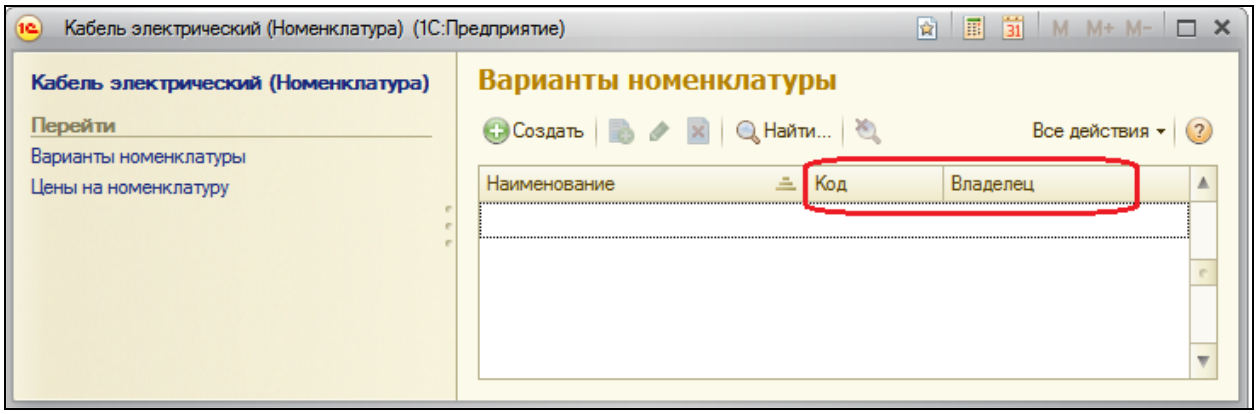
Откройте форму того же элемента номенклатуры и вместо названия мы увидим **Группа номенклатуры**.



Справочник Варианты номенклатуры

В режиме 1С:Предприятие

Теперь зададим набор свойств для элемента номенклатуры **Кабель электрический**. Для этого выполним команду **Варианты номенклатуры** для перехода к списку, где будут храниться наборы свойств элементов номенклатуры.



Открывшаяся форма списка вариантов номенклатуры не совсем нас устраивает – поля **Код** и **Владелец** явно лишние. **Код** нового варианта номенклатуры генерируется автоматически и ничего пользователю не говорит. **Владелец** варианта номенклатуры отражен в левом верхнем углу панели навигации формы и тоже в списке не имеет смысла.

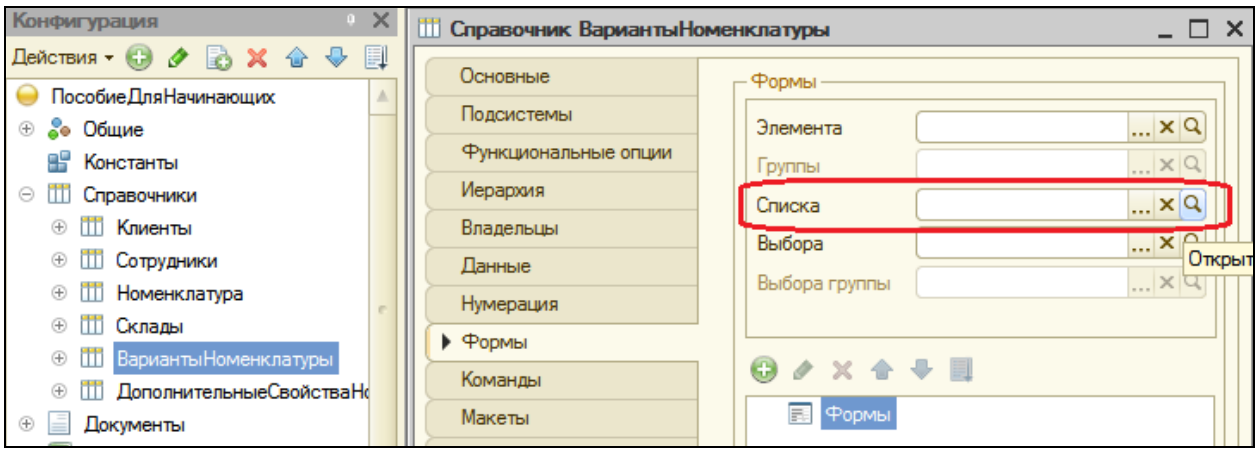
Чтобы сделать эти колонки невидимыми, нам нужно создать форму списка справочника **ВариантыНоменклатуры** и при ее создании проанализировать, откуда она открывается (это можно понять по значению параметра формы **Отбор**).

Если установлен отбор по владельцу (т.е. она открывается из списка номенклатуры), то мы будем в ней скрывать колонки **Код** и **Владелец**. Если же форма открывается другими способами, то эти колонки могут понадобиться, поэтому просто удалить их из формы было бы неправильно.

Поскольку форма создается на сервере, делать это нужно в обработчике события формы **ПриСозданииНаСервере**.

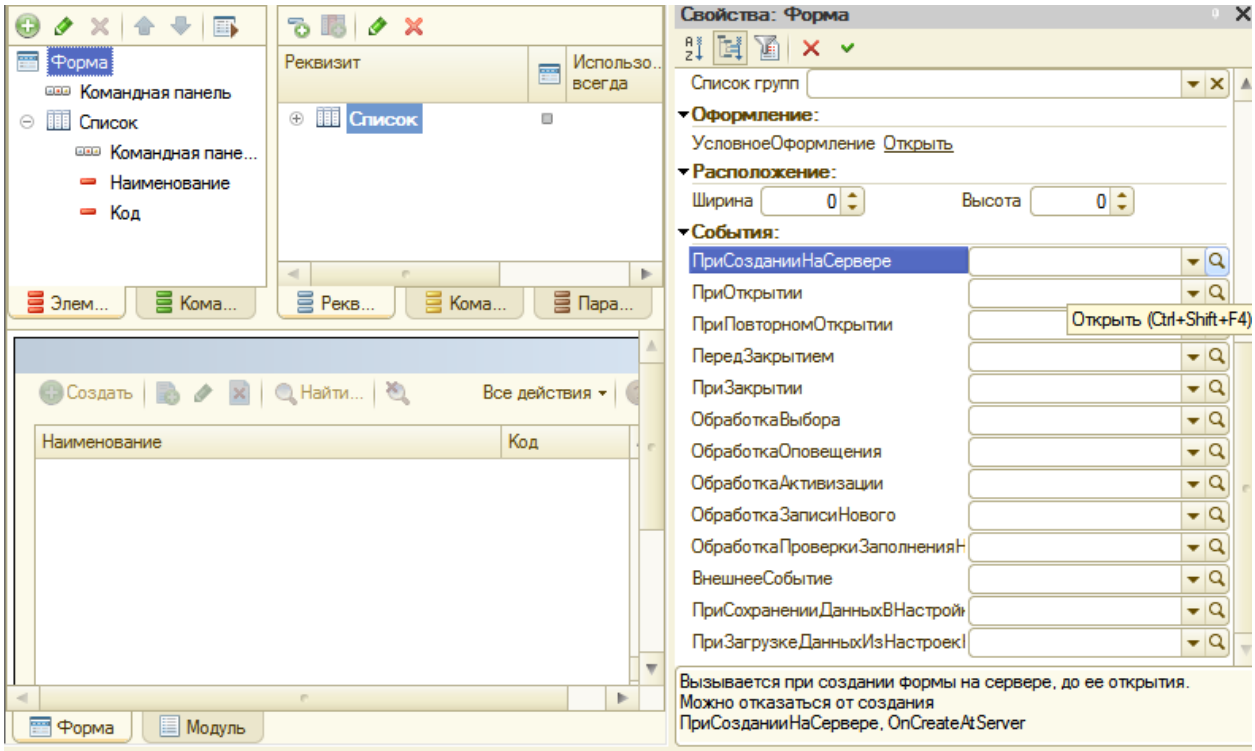
В режиме Конфигуратор

Для создания формы откройте окно редактирования объекта Справочник **ВариантыНоменклатуры**, перейдите на вкладку Формы, нажмите кнопку открытия  и создайте основную форму списка.



Форма, созданная конструктором, не содержит поля **Владелец**. Поэтому наша задача даже упрощается: нам нужно будет скрыть только поле **Код**.

В открывшемся окне редактора форм вверху слева расположено окно элементов формы. Выделите в нем элемент **Форма** (нам нужно событие формы в целом) и двойным щелчком откройте свойства этого элемента. Прокрутите список вниз, найдите событие **ПриСозданииНаСервере** и нажмите кнопку открытия  .



В модуле формы будет создан обработчик события формы **ПриСозданииНаСервере**, в который мы внесем следующий текст:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
        Если Параметры.Отбор.Свойство("Владелец") Тогда
                Элементы.Код.Видимость = Ложь;
        КонецЕсли;
КонецПроцедуры
```

Прокомментируем этот код.

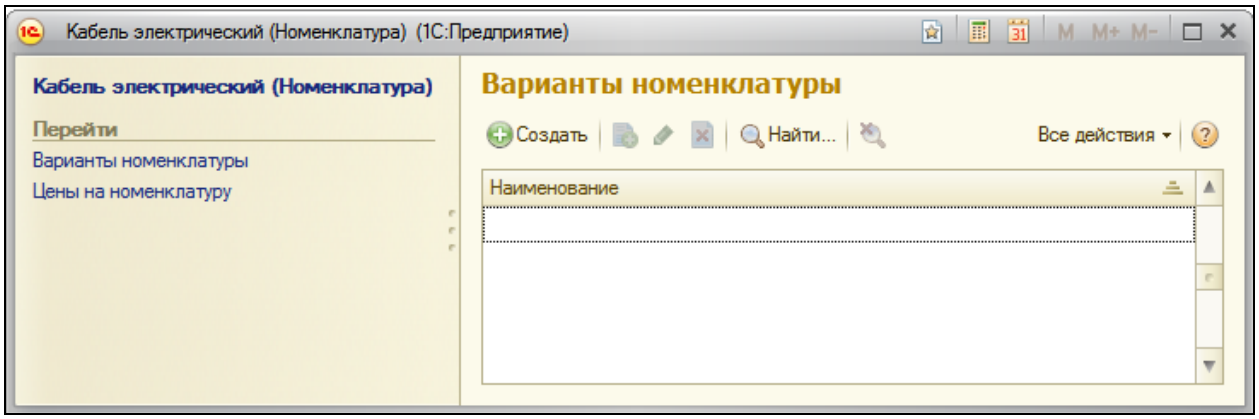
Параметры – это свойство управляемой формы, в модуле которой мы находимся. Используя это свойство, мы получаем объект, который содержит коллекцию параметров формы.

К элементу этой коллекции **Отбор** мы обращаемся по имени.

Используя метод **Свойство** структуры элементов отбора, мы определяем, установлен ли отбор по полю **Владелец**. Если такой отбор установлен, мы устанавливаем видимость поля **Код** в значение **Ложь**, т.е. скрываем это поле. Здесь **Элементы** – это свойство управляемой формы, которое позволяет получить доступ ко всем элементам формы.

В режиме 1С:Предприятие

Форма списка вариантов номенклатуры будет иметь вид:



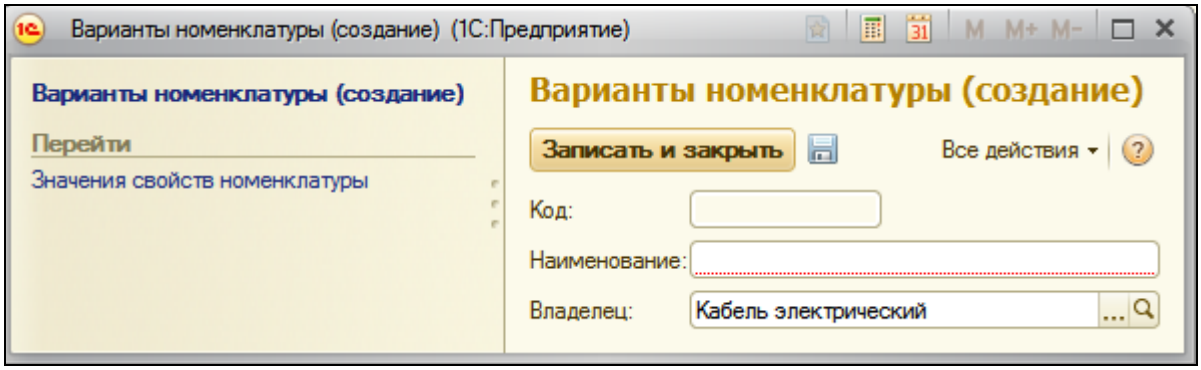
Как видите, мы добились желаемого результата: было три колонки, теперь только одна – Наименование.

Нажмите кнопку **Создать**, чтобы создать новый набор свойств для элемента номенклатуры. Появится сгенерированная форма элемента справочника **ВариантыНоменклатуры**. В ней есть недостатки:

- Заголовок формы должен быть задан в единственном числе,
- Лишние поля Код и Владелец,

- Команду перехода к подчиненной информации нужно переименовать в более понятную.

Вернемся в конфигуратор и исправим это.



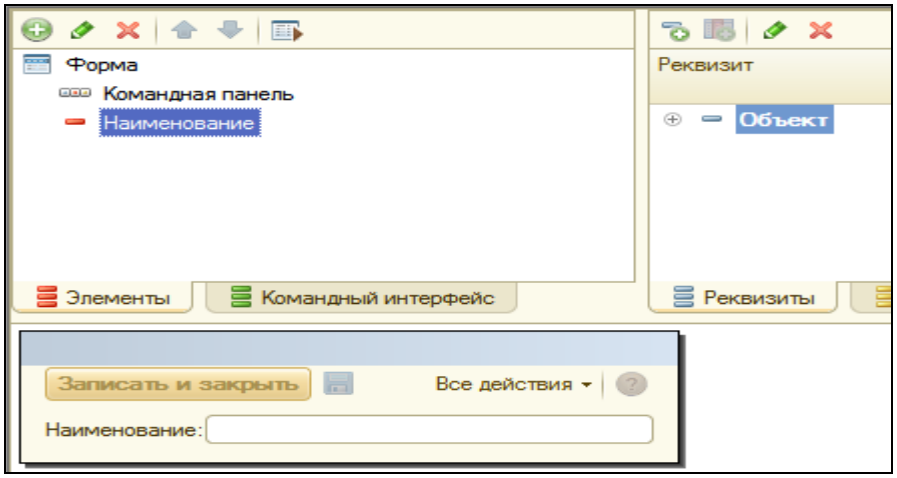
В режиме Конфигуратор

Переименуем заголовок формы, чтобы было понятно, что мы создаем в данный момент один элемент номенклатуры.

Для этого в окне редактирования справочника **ВариантыНоменклатуры** на закладке **Основные** задайте **Представление объекта** в единственном числе как **Вариант номенклатуры**. Это свойство будет использоваться в интерфейсе как заголовок формы элемента справочника.

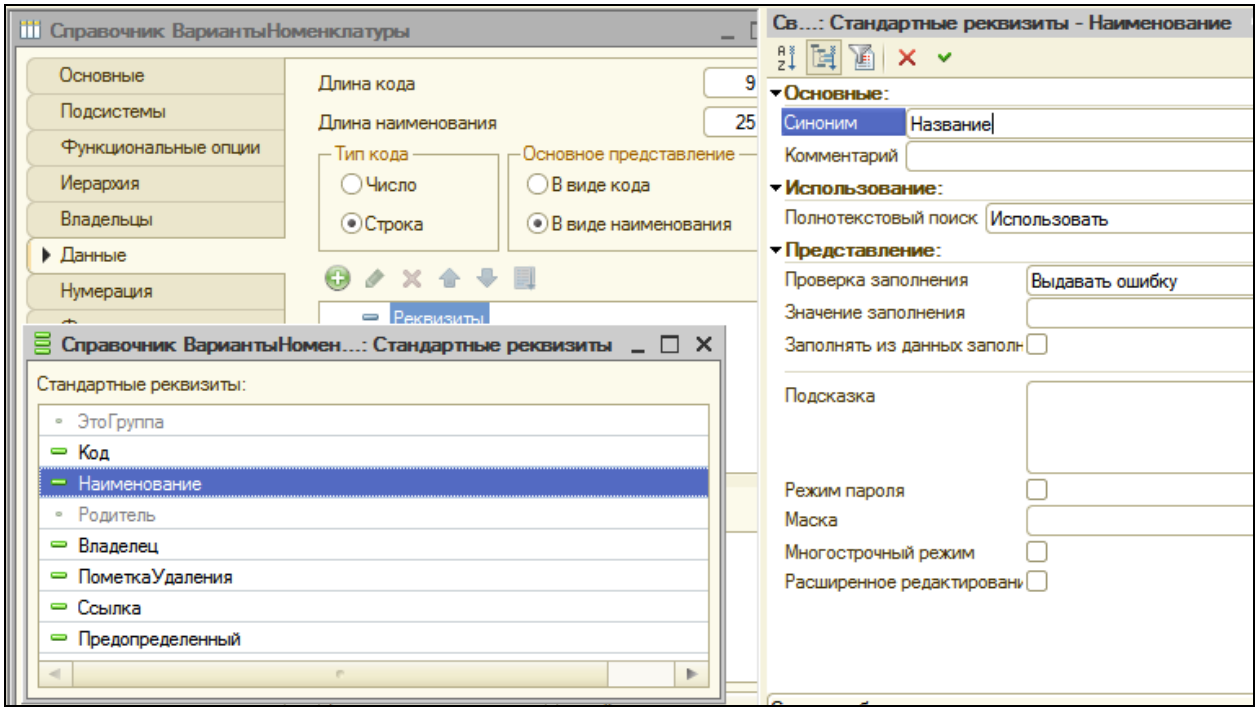
Нужно убрать поля **Код** и **Владелец** из этой формы. Для этого в окне редактирования справочника **ВариантыНоменклатуры** перейдите на закладку **Формы**, нажмите кнопку открытия и создайте основную форму элемента.

В окне структуры элементов формы выделите поочередно эти элементы и удалите их из формы.



В результате в форме элемента будет отображен только один реквизит справочника – **Наименование**. Его представление мы тоже немного поправим.

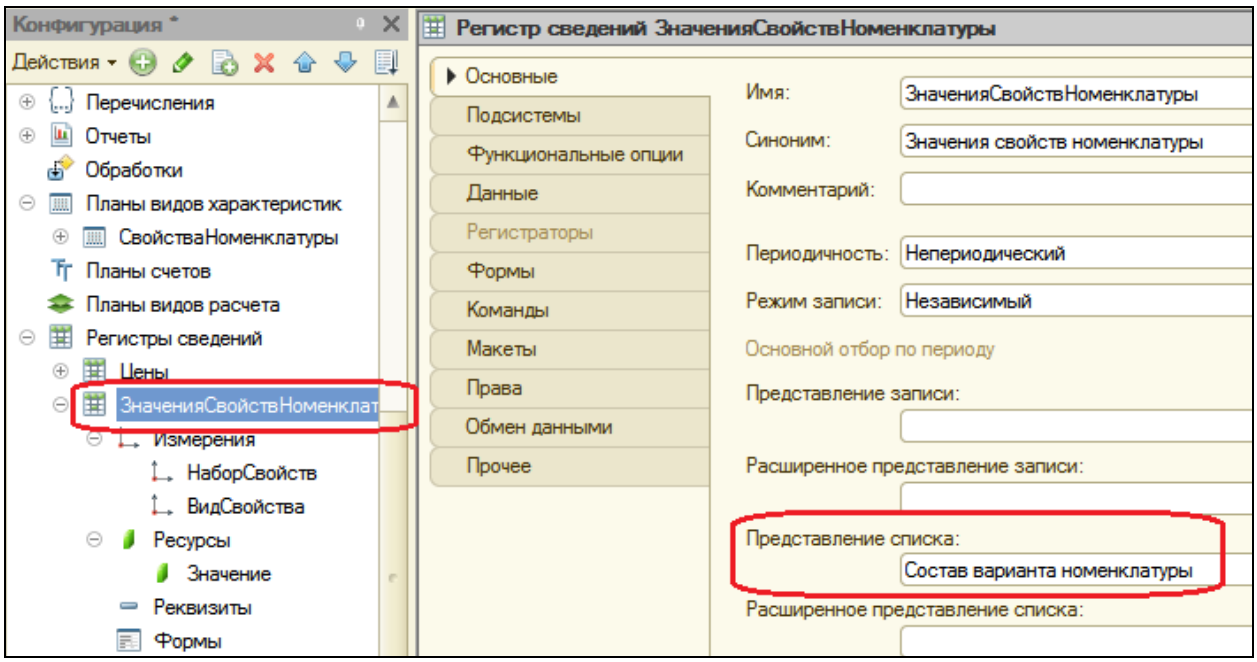
На закладке **Данные** в окне редактирования справочника нажмите кнопку **Стандартные реквизиты**, в списке реквизитов дважды щелкните на реквизите **Наименование** и в свойствах задайте **Синоним** – **Название**.



Заголовок формы **Вариант номенклатуры** и подчиненная ему информация – **Значения свойств номенклатуры** не «вяжутся» друг с другом. Это записи одноименного регистра, к которым можно перейти из формы элемента.

Поэтому в окне редактирования объекта конфигурации Регистр сведений **ЗначенияСвойствНоменклатуры** на закладке **Основные** задайте **Представление списка** как **Состав варианта номенклатуры**.

Это свойство будет использоваться в интерфейсе 1С:Предприятия как заголовок формы списка регистра.



В режиме 1С:Предприятие

В разделе **Учет материалов** откройте справочник **Номенклатура** и его элемент **Кабель электрический**.

В форме элемента выполните команду **Варианты номенклатуры** для перехода к списку наборов свойств данного элемента. Пока этот список пуст. Нажмите кнопку **Создать**. Теперь в открывшейся форме варианта номенклатуры нас всё устраивает.

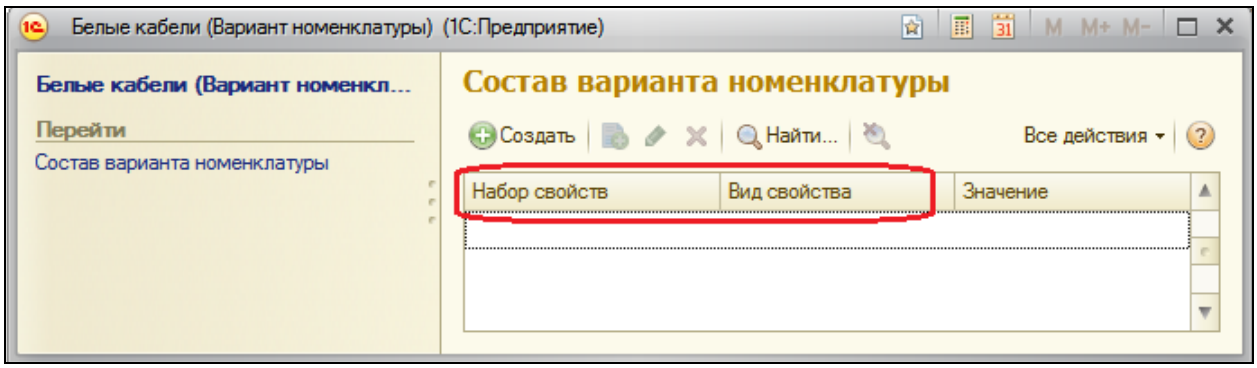
Регистр Значения свойств номенклатуры

В режиме 1С:Предприятие

Создадим вариант номенклатуры **Белые кабели**.

Выполните команду **Состав варианта номенклатуры** для перехода к составу редактируемого варианта номенклатуры. Если вариант номенклатуры еще не записан, то появится вопрос о записи данных, на который мы ответим утвердительно.

После этого откроется форма списка регистра **Значения свойств номенклатуры**, которая генерируется по умолчанию.



В этой форме нас также не всё устраивает:

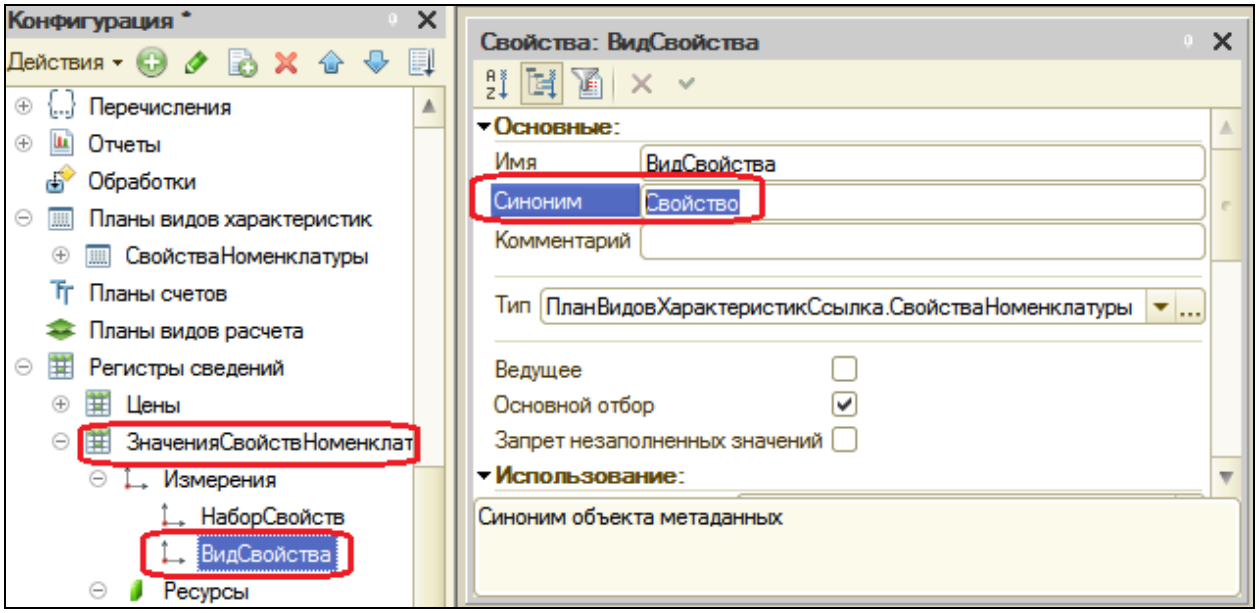
- Заголовок колонки ВидСвойства лучше переименовать,
- Лишняя колонка НаборСвойств.

Вернемся в конфигуратор и устраним эти недостатки.

В режиме Конфигуратор

Название колонки **Вид свойства** лучше переименовать в **Свойство**.

Для этого в окне редактирования регистра сведений **ЗначенияСвойствНоменклатуры** на закладке **Данные** откройте свойства измерения **ВидСвойства** и задайте его **Синоним** как **Свойство**.



Поскольку регистр имеет ведущее измерение **НаборСвойств** типа **СправочникСсылка.ВариантыНоменклатуры**, поле **Набор свойств** –

лишнее, т.к. владелец данного набора свойств отражен в левом верхнем углу панели навигации формы.

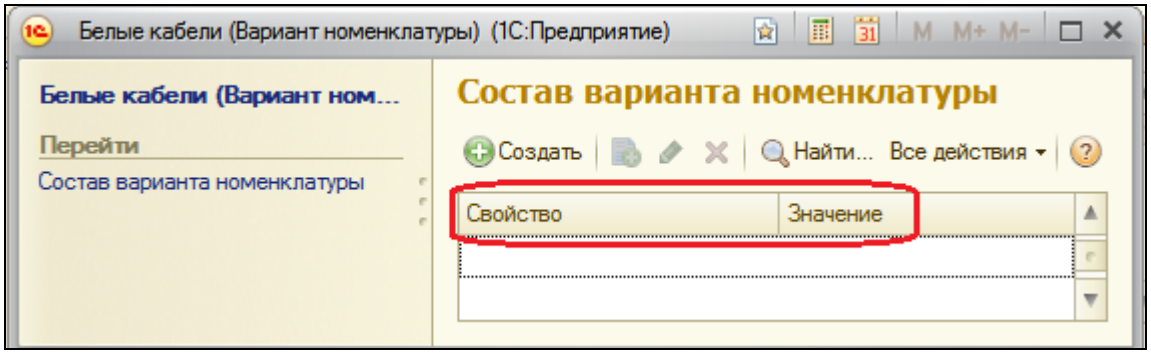
Поэтому создадим обработчик события **ПриСозданииНаСервере** формы списка регистра и в нем сделаем колонку **НаборСвойств** невидимой в случае открытия формы с отбором по этому полю, т.е. если форма списка регистра открыта из формы элемента справочника Варианты номенклатуры.

Откройте окно редактирования регистра сведений **ЗначенияСвойствНоменклатуры**, перейдите на закладку **Формы**, создайте основную форму списка. Затем создайте для формы обработчик события формы **ПриСозданииНаСервере** со следующим текстом:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
	Если Параметры.Отбор.Свойство("НаборСвойств") Тогда
		Элементы.НаборСвойств.Видимость = Ложь;
	КонецЕсли;
КонецПроцедуры
```

В режиме 1С:Предприятие

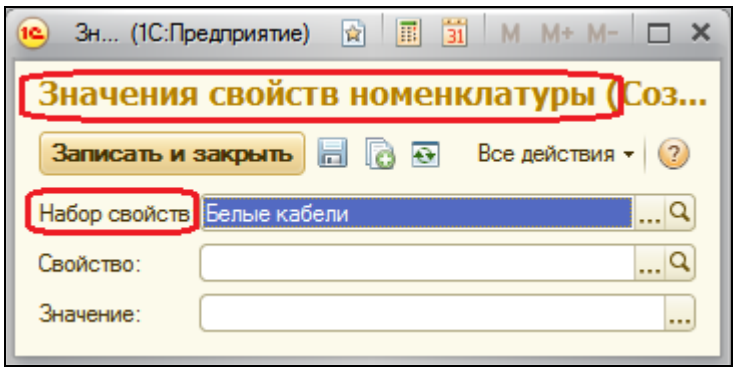
В результате форма списка регистра **Состав варианта номенклатуры** примет вид:



Теперь, если нажать **Создать**, чтобы ввести новую запись, откроется форма записи регистра **ЗначенияСвойствНоменклатуры**.

Эта форма сгенерирована автоматически и не лишена недостатков:

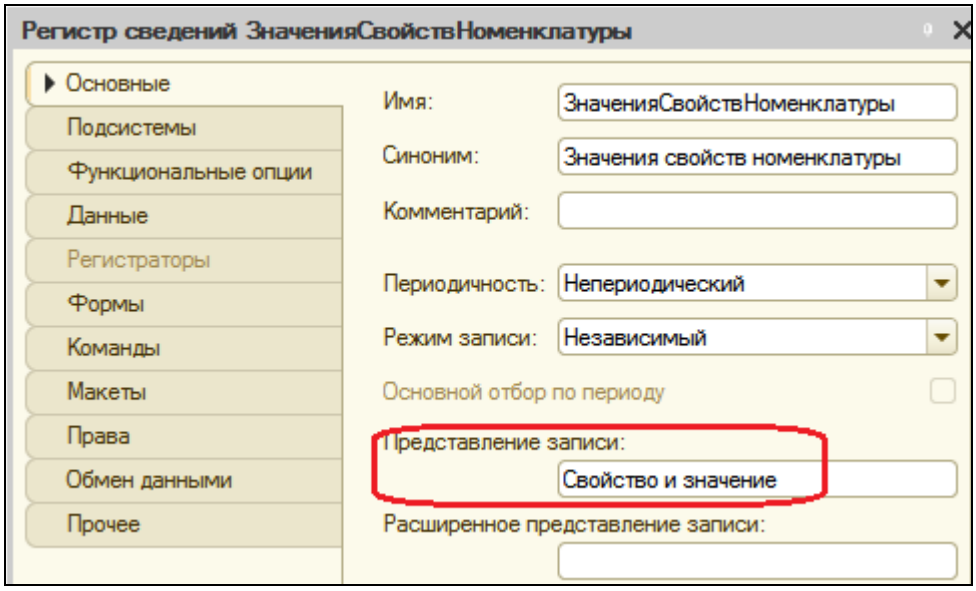
- Заголовок формы должен быть задан в единственном числе,
- Лишняя колонка **Набор свойств.**



В режиме конфигуратор

Во-первых, нужно переименовать заголовок формы, чтобы было понятно, что мы создаем в данный момент одно свойство и его значение в составе варианта номенклатуры.

Для этого в окне редактирования регистра сведений **ЗначенияСвойствНоменклатуры** на закладке **Основные** задайте **Представление** записи как **Свойство и значение**.



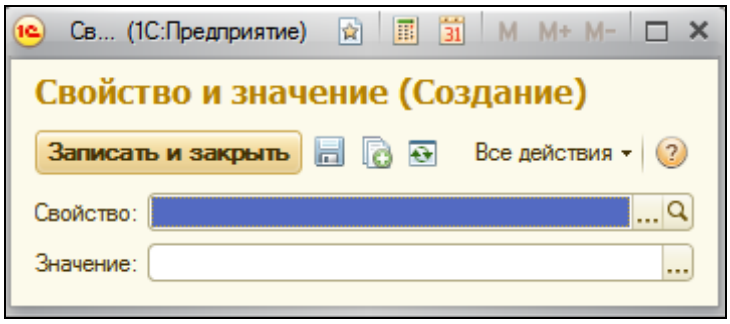
Это свойство будет использоваться в интерфейсе как заголовок формы записи регистра.

Во-вторых, нужно убрать поле **НаборСвойств** из этой формы. Для этого в окне редактирования регистра сведений **ЗначенияСвойствНоменклатуры** перейдите на закладку **Формы** и создайте основную форму записи.

В окне структуры элементов формы выделите поле **НаборСвойств** и удалите его.

В режиме 1С:Предприятие

Форма записи регистра **ЗначенияСвойствНоменклатуры** примет вид:



Создание характеристик номенклатуры

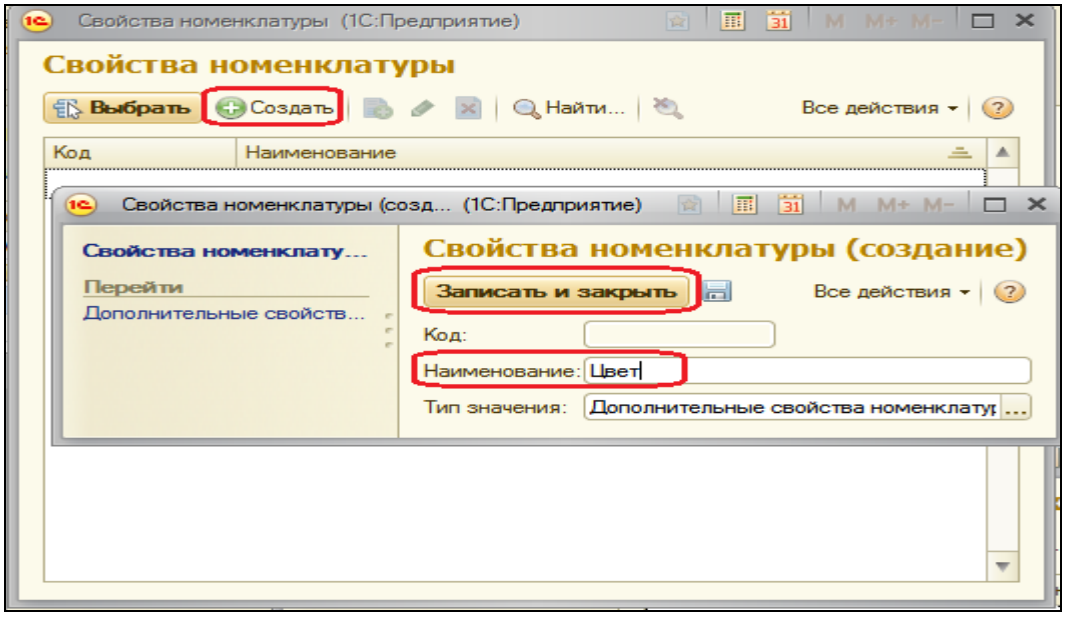
В режиме 1С:Предприятие

Теперь создадим различные варианты номенклатуры в режиме 1С:Предприятие.

В разделе **Учет материалов** откройте справочник **Номенклатура** и его элемент **Кабель электрический**.

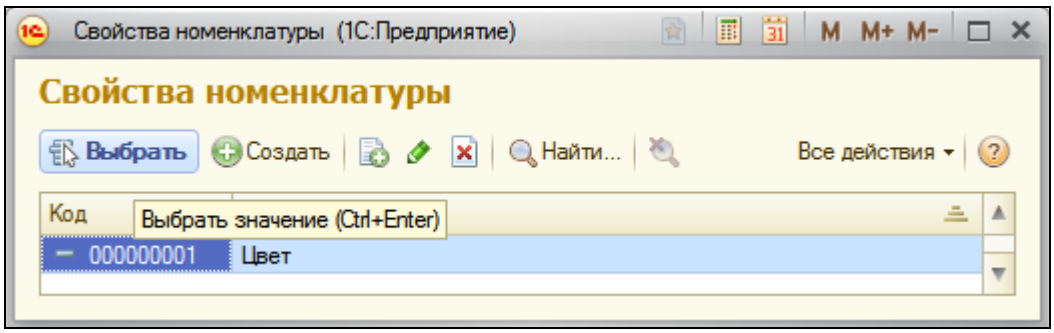
В форме элемента номенклатуры выполните команду **Варианты номенклатуры** для перехода к списку наборов свойств данного элемента номенклатуры. В форме списка вариантов номенклатуры откройте набор свойств **Белые кабели**, который мы создали ранее.

В форме варианта номенклатуры нажмите **Состав варианта номенклатуры**. Этот список пока пуст. Нажмите кнопку **Создать**. В открывшейся форме создайте свойство **Цвет** со значением **Белый**.



Обратите внимание, что в форме элемента плана видов характеристик и в форме элемента справочника дополнительных характеристик номенклатуры также есть лишнее поле **Код**. Можете самостоятельно исправить эти недостатки аналогично предыдущим для справочника **ВариантыНоменклатуры**.

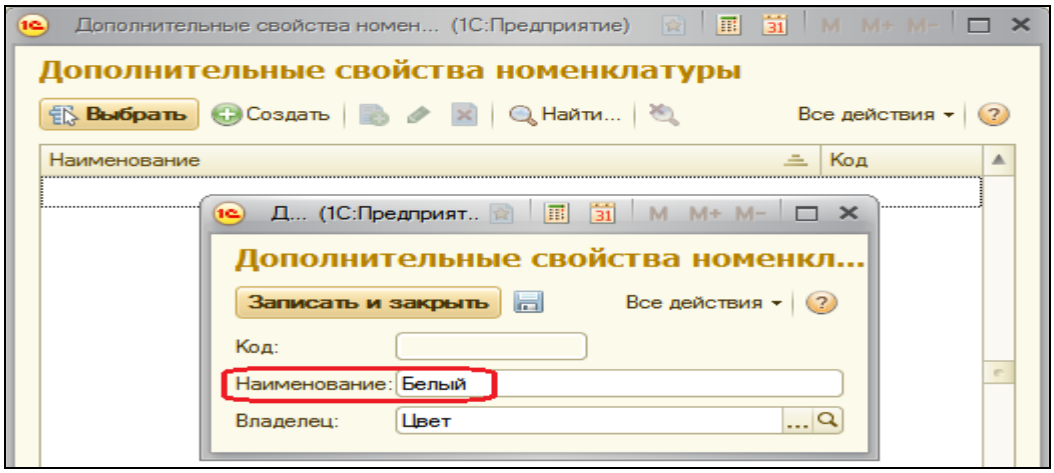
В окне выбора плана видов характеристик появится созданная нами характеристика. Нажмите кнопку **Выбрать**. В результате мы вернемся в форму записи состава варианта номенклатуры с заголовком **Свойство и значение**.



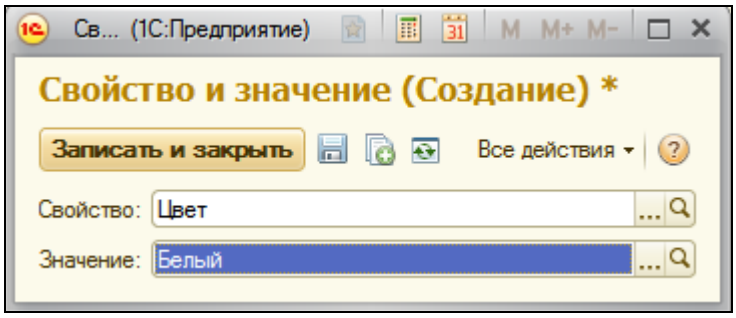
Нажмите кнопку выбора  в поле Значение.

Ресурс **Значение** регистра **ЗначенияСвойствНоменклатуры** имеет тип **Характеристика.СвойстваНоменклатуры**. Это составной тип данных, который описан в свойстве **Тип** значения характеристик плана видов характеристик **СвойстваНоменклатуры**. Т.к. для характеристики **Цвет** мы задали тип значения **СправочникСсылка.ДополнительныеСвойстваНоменклатуры**, то перед нами появится форма выбора этого справочника. Список свойств пока пуст. Нажмите кнопку **Создать**.

Введите тип значения **Белый**, в поле **Владелец** оставим имеющееся значение – **Цвет**.

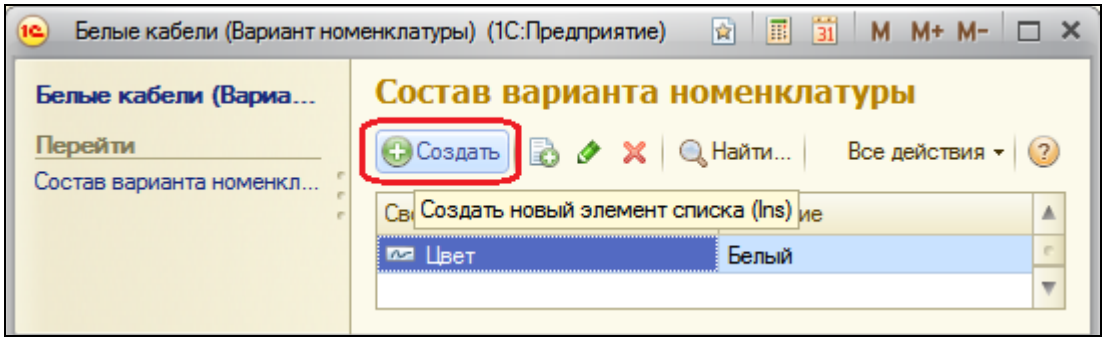


Нажмите **Записать и закрыть**. Нажмите кнопку **Выбрать**.

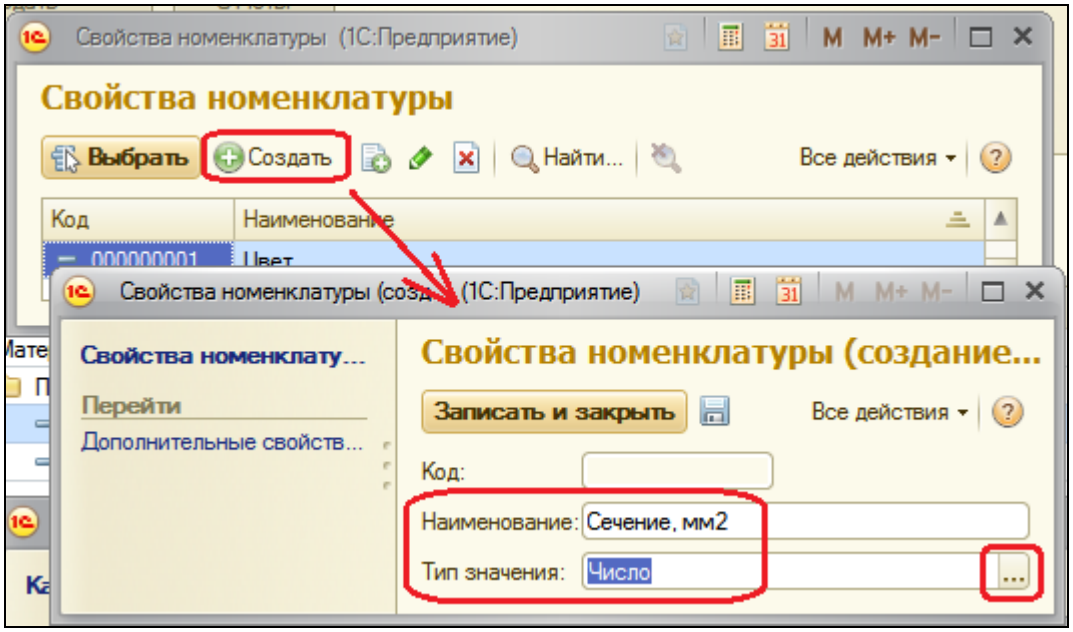


Нажмите **Записать и закрыть**. Мы вернулись в форму списка состава варианта номенклатуры.

Создадим еще одно свойство – **Сечение, мм2** – в составе варианта номенклатуры **Белые кабели**. Для этого повторим только что выполненные действия. Нажмите кнопку **Создать**.

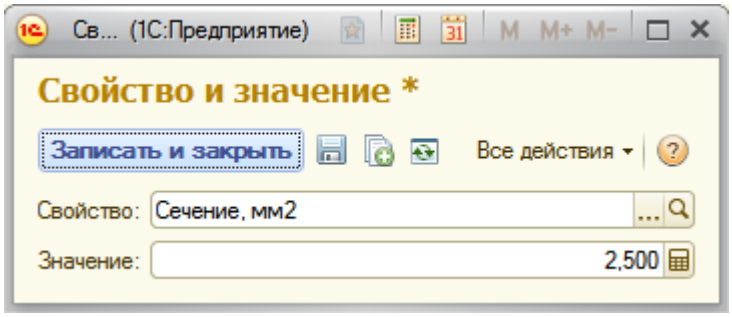


В открывшейся форме нажмите кнопку выбора в поле **Свойство**. Далее кнопку **Создать**.

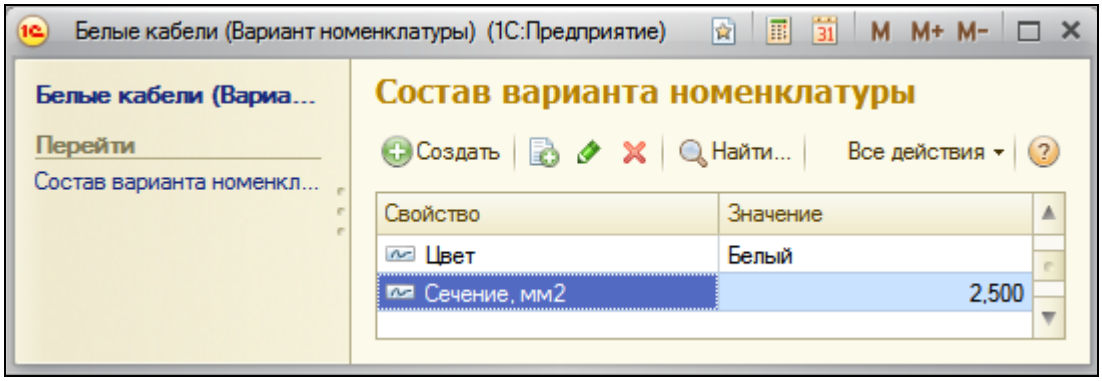


Введите наименование – **Сечение, мм2** и выберите тип **Число**, длина 15, точность 3.

Нажмите **Записать и закрыть**. Нажмите **Выбрать**. Введите в поле **Значение** число **2,5**.



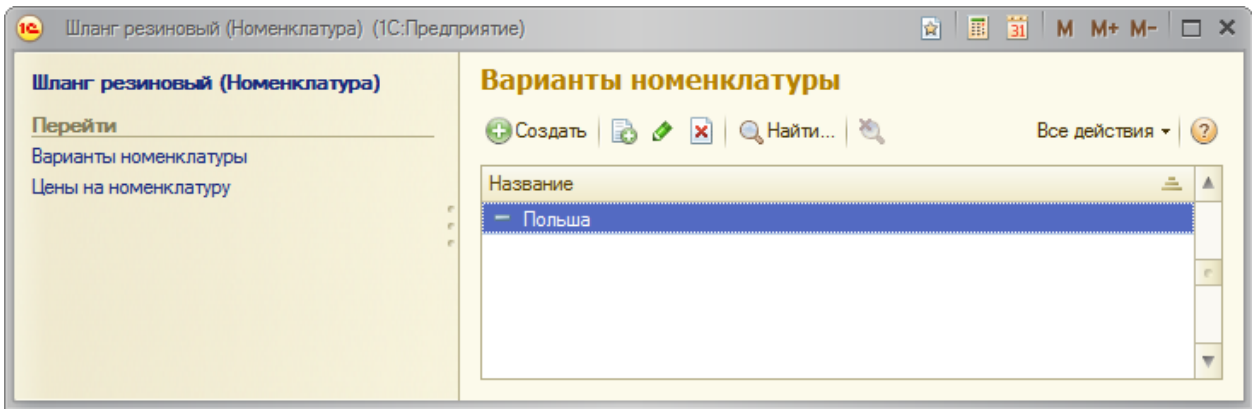
Нажмите **Записать и закрыть**.

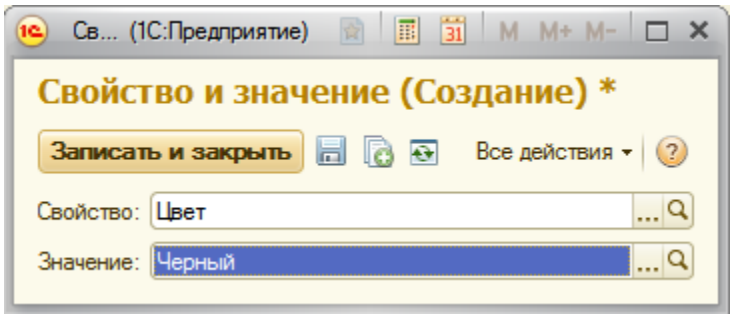


Теперь аналогично создадим набор свойств для элемента справочника Номенклатура – Шланг резиновый. Этот набор будет называться Польша и состоять из свойств:

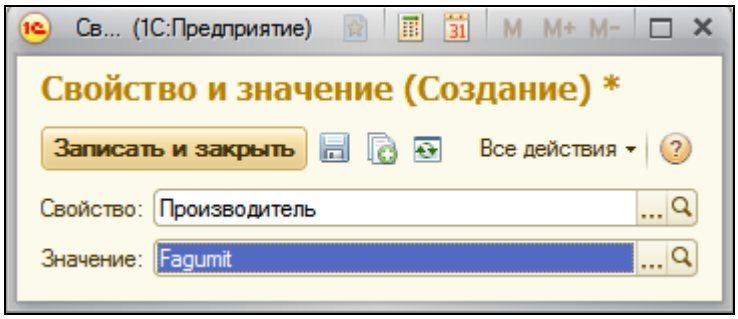
- Цвет – Черный;
- Производитель – Fagumit.

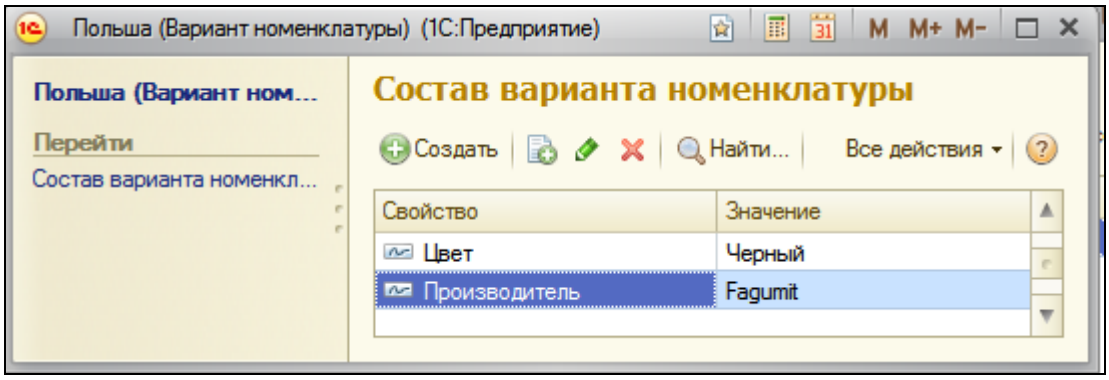
При создании свойства **Цвет** выберем его из уже имеющихся свойств в плане видов характеристик. Значение этой характеристики (**Черный**) сначала добавим в справочник дополнительных свойств номенклатуры и затем выберем из него.



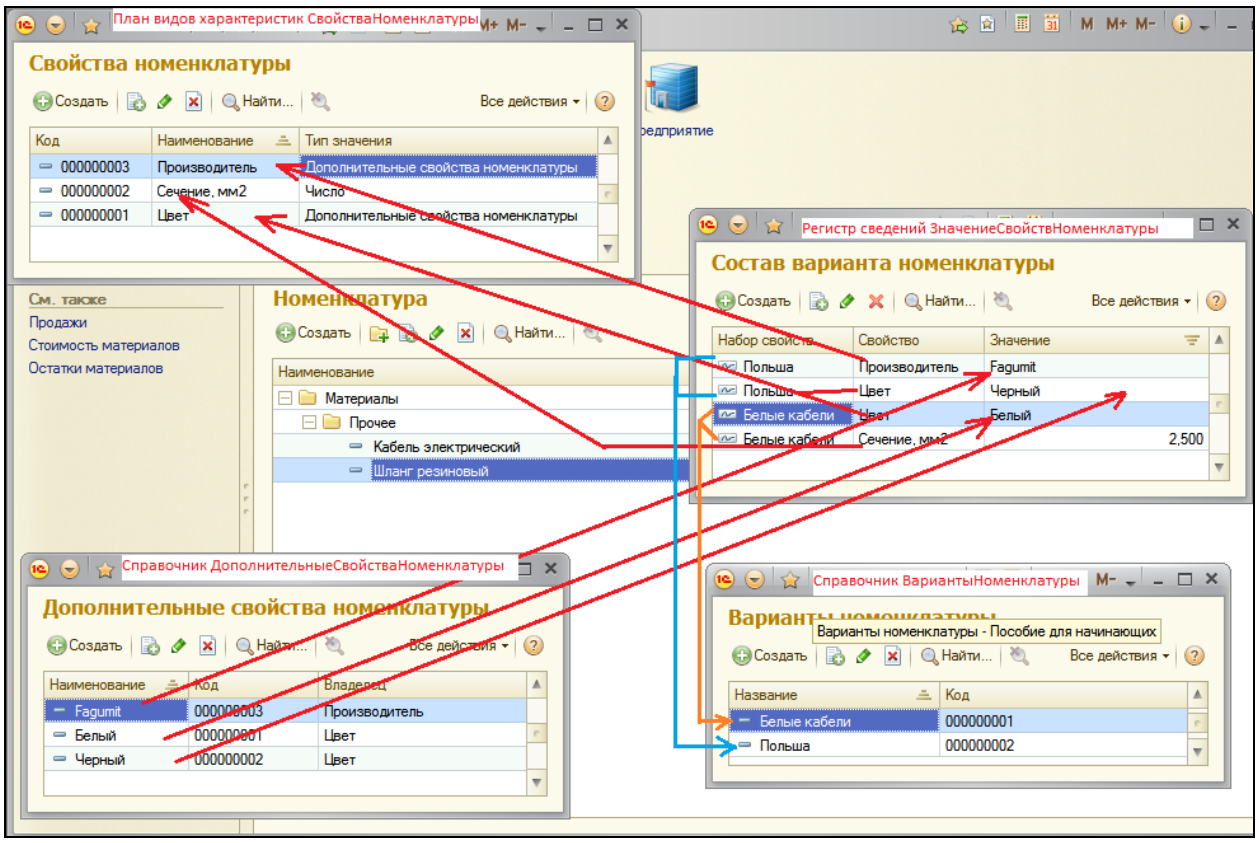


При создании свойства **Производитель** с типом значения **Дополнительные свойства номенклатуры** сначала добавим это свойство в план видов характеристик (тип значения – **Дополнительные свойства номенклатуры**), а затем выберем из него. Значение этой характеристики (**Fagumit**), сначала добавим в справочник дополнительных свойств номенклатуры и затем выберем из него.





Теперь посмотрим на созданное нами с точки зрения разработчика.



Доработка учетных механизмов

Мы добавили возможность указывать произвольные характеристики для номенклатуры и создали несколько таких характеристик – вариантов номенклатуры. Теперь хотелось бы иметь возможность еще и учитывать номенклатуру в разрезе этих характеристик, а именно:

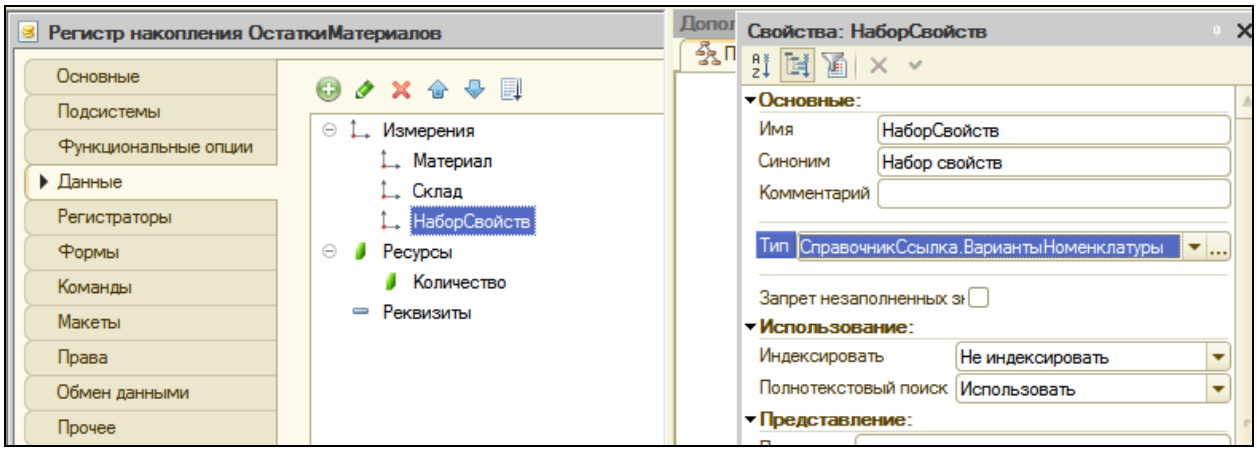
- Приходовать товар, указывая характеристики;
- Расходовать товар, указывая характеристики;
- Получать отчеты не просто по номенклатуре, а по номенклатуре с определенными характеристиками.

Для этого требуется доработать регистры и создать новый отчет.

Регистр Остатки материалов

В режиме Конфигуратор

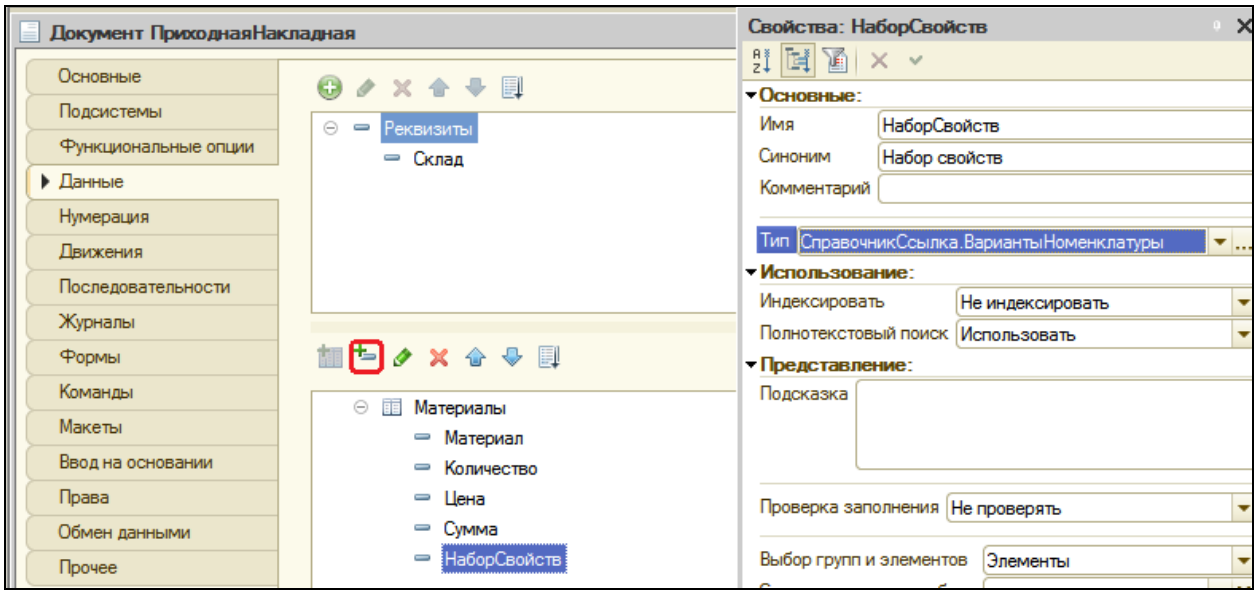
Изменим структуру регистра накопления **ОстаткиМатериалов**. Откройте окно редактирования регистра накопления и на закладке **Данные** добавим новое измерение **НаборСвойств** с типом **Справочник.Ссылка.ВариантыНоменклатуры**.



Документ Приходная накладная

Теперь нам нужно доработать документ **ПриходнаяНакладная**, чтобы при приходовании материалов можно было указать набор свойств и чтобы этот набор записывался в регистры при проведении документа.

Для этого откройте окно редактирования документа **ПриходнаяНакладная** и на вкладке **Данные** добавим в табличную часть документа новый реквизит **НаборСвойств** с типом **СправочникСсылка.ВариантыНоменклатуры**.



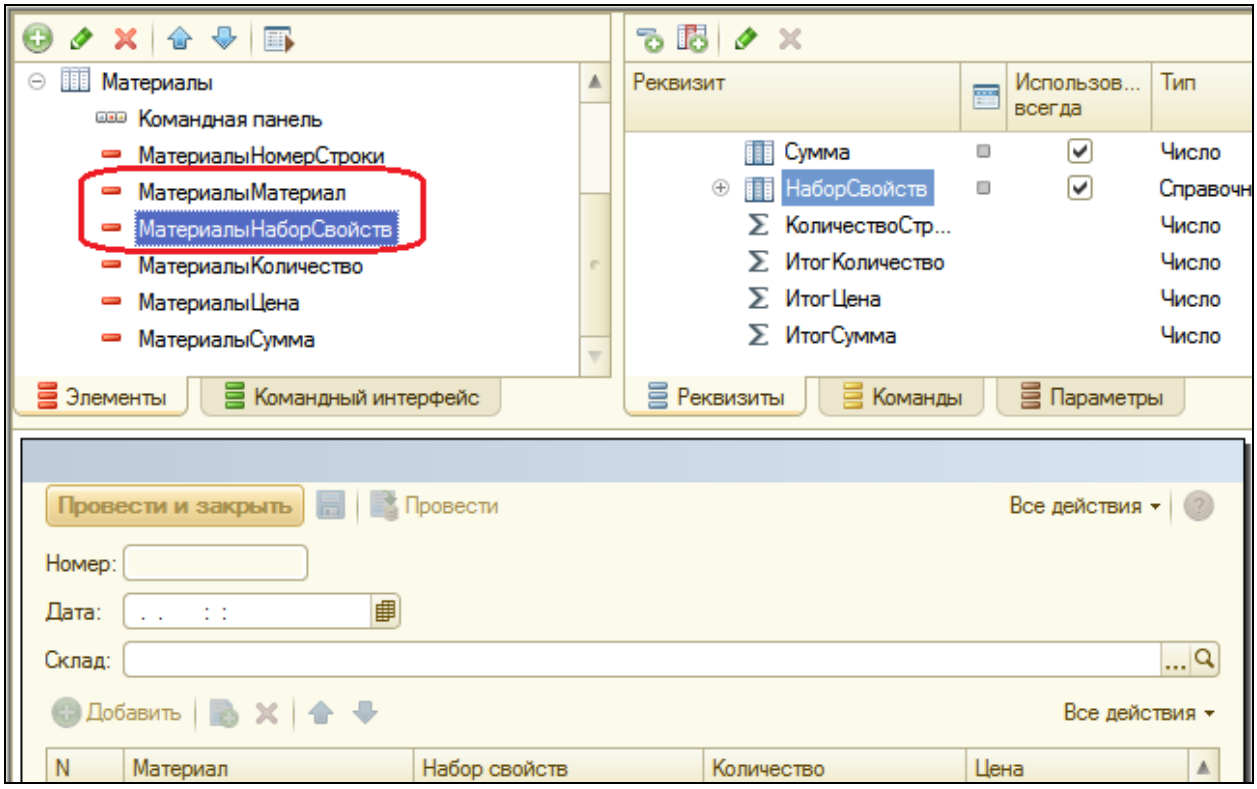
У этого реквизита необходимо заполнить свойство **Связи параметров выбора**, чтобы после выбора номенклатуры в этом свойстве выбирать только среди тех наборов свойств, которые относятся к данной номенклатуре. Найдите это свойство и нажмите кнопку выбора.

Перенесите из списка реквизит **Материалы.Материал**. Тем самым мы задали, что при выборе в поле **НаборСвойств** будет всегда открываться список элементов справочника **Варианты номенклатуры**, подчиненных материалу, выбранному в колонке **Материал**.

После этого расположим этот реквизит в табличной части формы документа. Для этого перейдите на вкладку **Формы** и двойным щелчком на строке **ФормаДокумента** откройте форму.

В правом верхнем окне на закладке **Реквизиты** раскройте реквизит формы **Объект**.

Найдите в табличной части реквизит **НаборСвойств** и перетащите его в окно элементов формы, расположенное слева в верхней части редактора форм, в таблицу **Материалы** и расположите его в структуре элементов формы после поля **Материал**.



В заключение в окне редактирования документа **ПриходнаяНакладная** на закладке **Прочее** откроем модуль объекта. Откройте процедуру обработчика события **ОбработкаПроведения** и

добавьте к формируемым движениям присвоение значения измерению **НаборСвойств** регистра **ОстаткиМатериалов**.

```
…
// регистр ОстаткиМатериалов Приход
        Движения.ОстаткиМатериалов.Записывать = Истина;
        Движения.СтоимостьМатериалов.Записывать = Истина;
        Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
                Движение = Движения.ОстаткиМатериалов.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Материал = ТекСтрокаМатериалы.Материал;
                Движение.НаборСвойств = ТекСтрокаМатериалы.НаборСвойств;
                Движение.Склад = Склад;
                Движение.Количество = ТекСтрокаМатериалы.Количество;
…
```

Документ Оказание услуги

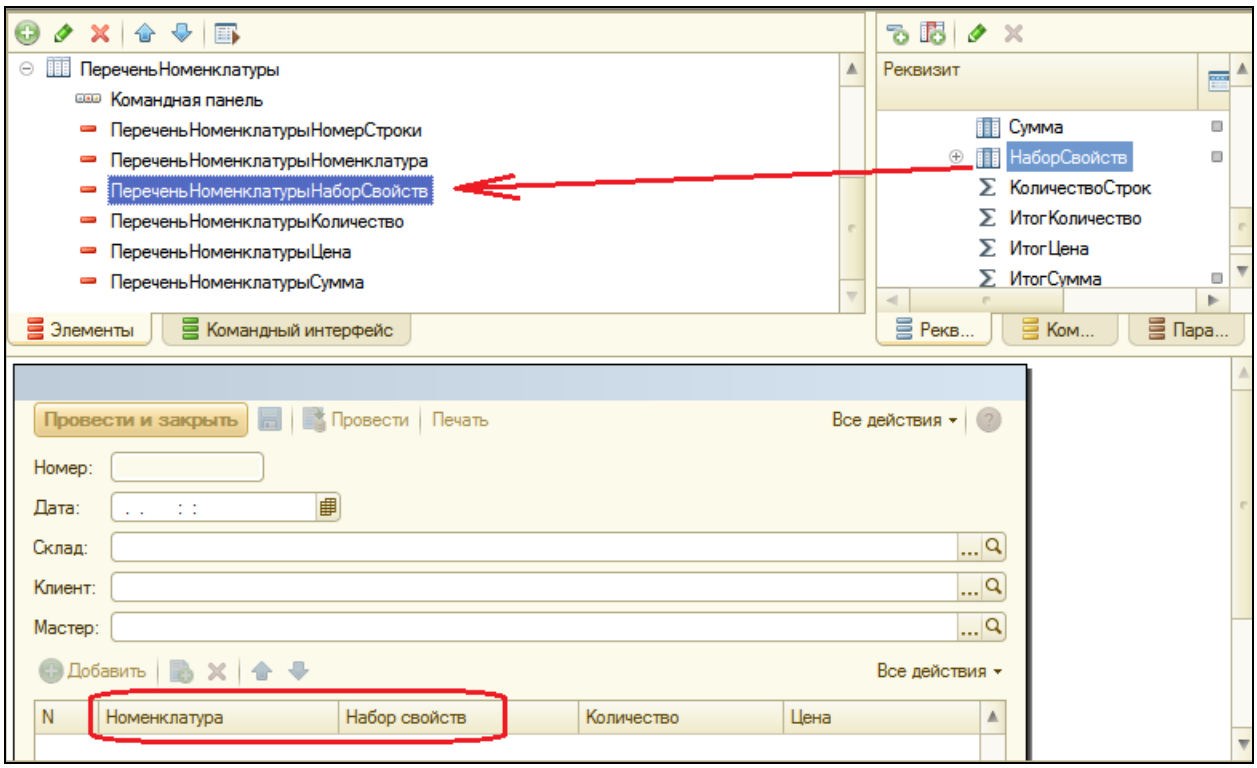
Теперь аналогично доработаем документ **ОказаниеУслуги**.

Для того чтобы при расходовании материалов пользователь мог указывать набор свойств для каждого расходуемого материала, откроем окно редактирования документа **ОказаниеУслуги**, на закладке **Данные** добавим в табличную часть документа новый реквизит **НаборСвойств** с типом **СправочникСсылка.ВариантыНоменклатуры**.

У этого реквизита заполним свойство **Связи параметров выбора**. Перенесем из списка реквизитов в список параметров **ПеречеьНоменклатуры.Номенклатура**. Тем самым мы задали, что при выборе в поле **НаборСвойств** будет всегда открываться список элементов справочника **Варианты номенклатуры**, подчиненных материалу, выбранному в колонке **Номенклатура**. Расположим этот реквизит в табличной части формы документа после поля **Номенклатура**.

В окне редактирования документа на закладке **Прочее** откройте модуль объекта. Откройте процедуру обработчика события **ОбработкаПроведения** и добавьте к формируемым движениям присвоение значения измерению **НаборСвойств** регистра **ОстаткиМатериалов**:

```
// регистр ОстаткиМатериалов Расход
…
                Движение = Движения.ОстаткиМатериалов.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                Движение.Период = Дата;
                Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
                Движение.НаборСвойств = ВыборкаДетальныеЗаписи.НаборСвойств;
                Движение.Склад = Склад;
```



Поскольку в предыдущей работе мы оптимизировали процедуру проведения документа и получали все данные документа с помощью запроса, то в текст запроса нужно также добавить строки для получения нового реквизита документа.

```
Запрос.Текст =
                "ВЫБРАТЬ
                |        ОказаниеУслугиПеречеьНоменклатуры.Номенклатура,
                |        ОказаниеУслугиПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры
КАК ВидНоменклатуры,
                |        ОказаниеУслугиПеречеьНоменклатуры.НаборСвойств,
                |        СУММА(ОказаниеУслугиПеречеьНоменклатуры.Количество) КАК
КоличествоВДокументе,
                |        СУММА(ОказаниеУслугиПеречеьНоменклатуры.Сумма) КАК
СуммаВДокументе
                |ПОМЕСТИТЬ НоменклатураДокумента
                |ИЗ
                |        Документ.ОказаниеУслуги.ПеречеьНоменклатуры КАК
ОказаниеУслугиПеречеьНоменклатуры
                |ГДЕ
                |        ОказаниеУслугиПеречеьНоменклатуры.Ссылка = &Ссылка
                |
                |СГРУППИРОВАТЬ ПО
                |        ОказаниеУслугиПеречеьНоменклатуры.Номенклатура,
                |
        ОказаниеУслугиПеречеьНоменклатуры.Номенклатура.ВидНоменклатуры,
                |        ОказаниеУслугиПеречеьНоменклатуры.НаборСвойств";

                Запрос.УстановитьПараметр("Ссылка", Ссылка);

        Результат = Запрос.Выполнить();
        Запрос2 = Новый Запрос;
        Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
        Запрос2.Текст = "ВЫБРАТЬ
```

```
                    |    НоменклатураДокумента.Номенклатура,
                    |    НоменклатураДокумента.ВидНоменклатуры,
                                |    НоменклатураДокумента.НаборСвойств,
                    |    НоменклатураДокумента.КоличествоВДокументе,
                    |    НоменклатураДокумента.СуммаВДокументе,
                    |    ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток, 0) КАК
Стоимость,
                    |    ЕСТЬNULL(ОстаткиМатериаловОстатки.КоличествоОстаток, 0) КАК
Количество
                    |ИЗ
                    |    НоменклатураДокумента КАК НоменклатураДокумента
```

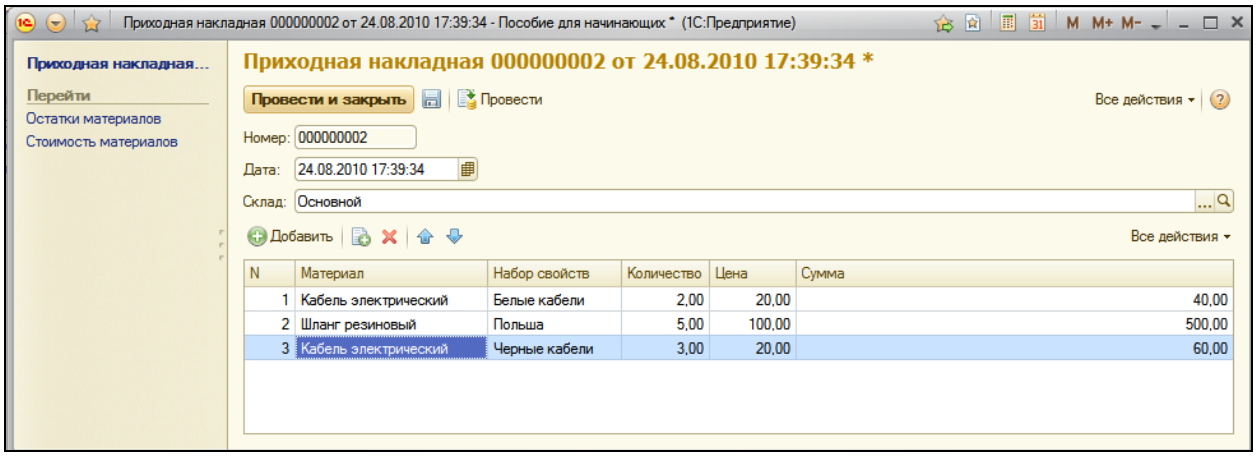
Приход/расход номенклатуры с учетом характеристик

В режиме 1С:Предприятие

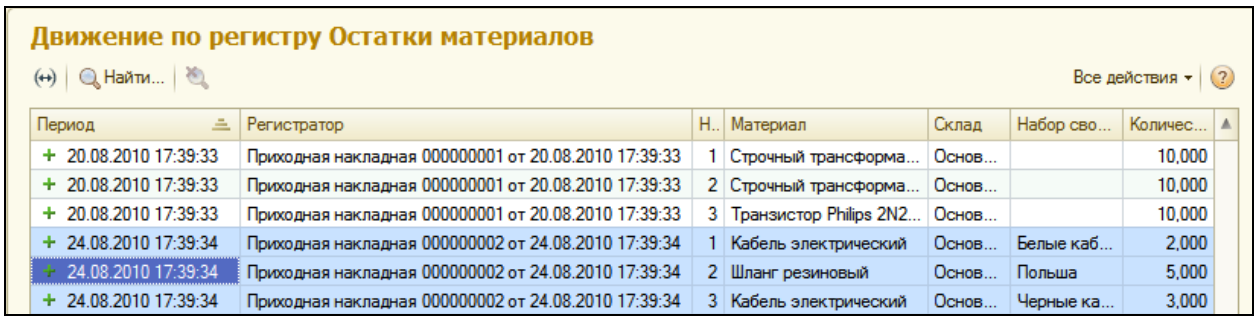
Запустим режим отладки и укажем наборы свойств при приходовании материалов.

Откройте документ **Приходная накладная №2** и укажите, что был закуплен белый электрический кабель в количестве 2 шт. и польский резиновый шланг 5 шт (см рисунок).

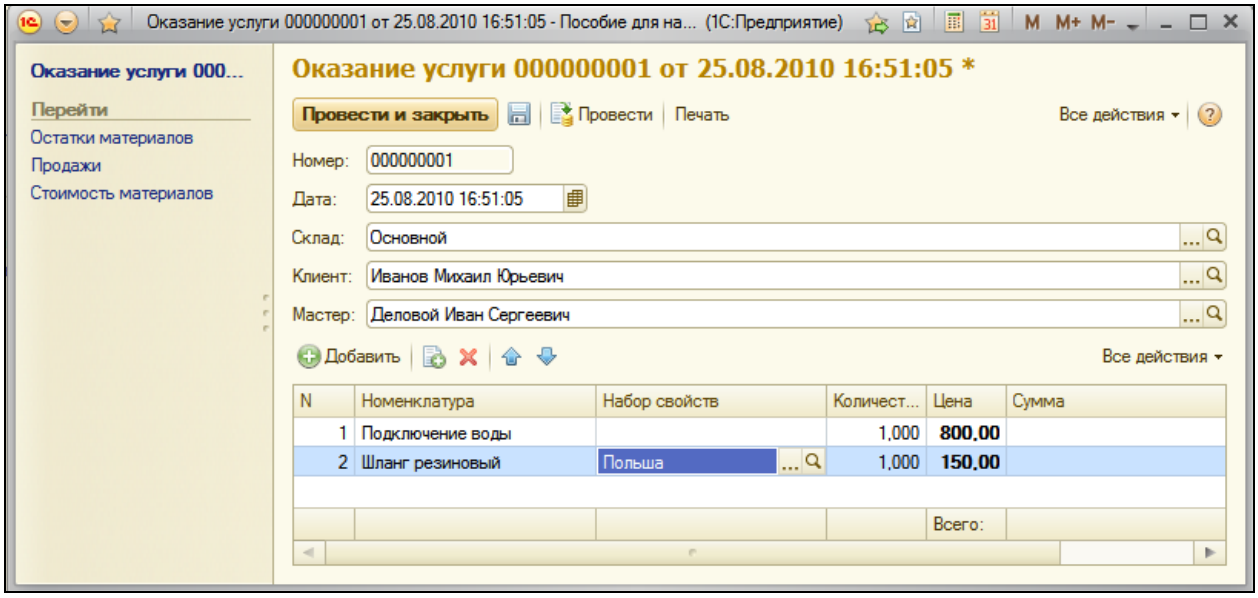
Затем скопируйте первую строку документа и укажите, что был закуплен еще и черный электрический кабель 3 шт. (В процессе ввода придется создать еще один набор свойств для электрического кабеля – **Черные кабели**, у которого **Цвет** – **Черный** и **Сечение** – **2,5**).



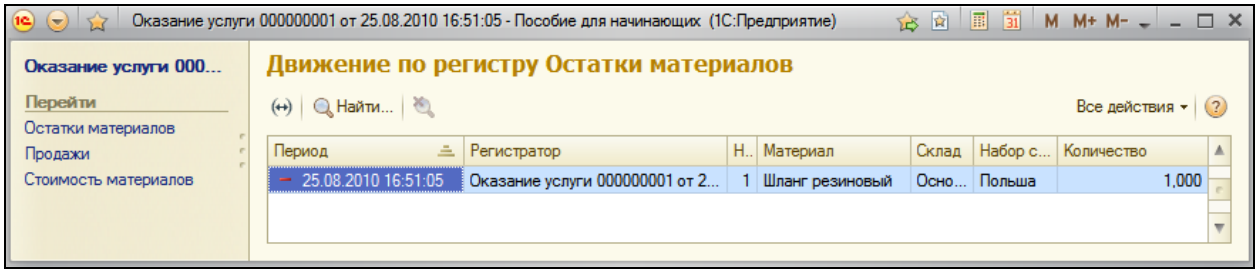
Проведите документ и перейдите в регистр **Остатки материалов** по ссылке слева.



Теперь откройте **Оказание услуги №1** и укажите, что был израсходован польский резиновый шланг.



Проведите документ и перейдите в **Остатки материалов**.



Отчет, использующий характеристики

Для полного завершения картины мы создадим отчет, который будет показывать нам наличие материалов с теми или иными свойствами. При создании этого отчета мы используем возможности системы компоновки данных для работы с характеристиками.

Набором данных для системы компоновки данных будет простой запрос к регистру ОстаткиМатериалов, к нему мы опишем как выглядит механизм характеристик. На основе этих описаний система компоновки сама сформирует достаточный интерфейс.

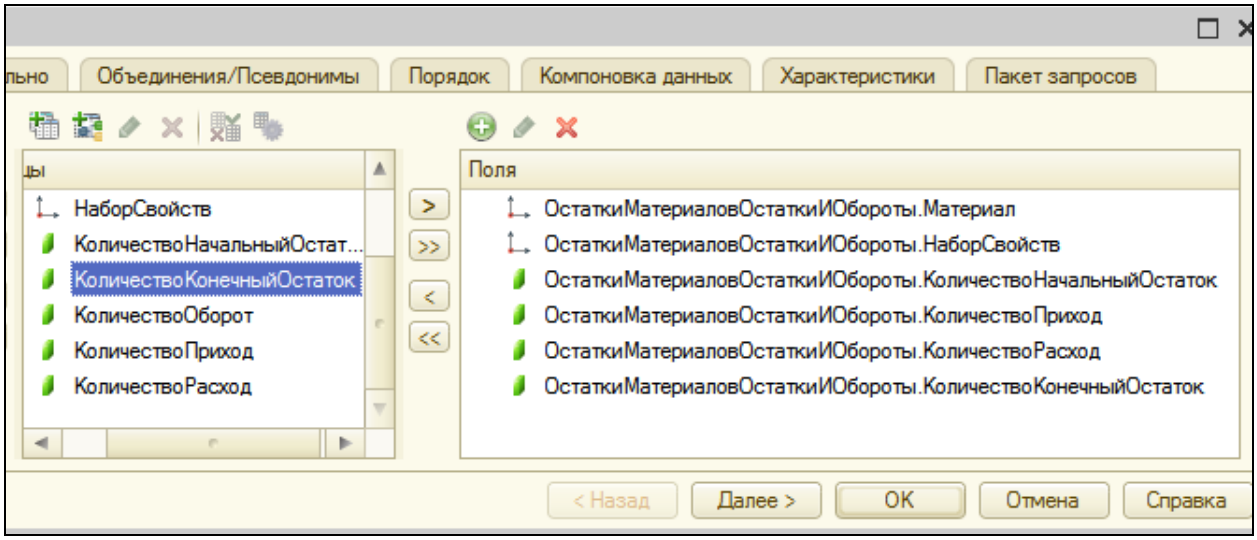
В режиме Конфигуратор

Добавьте новый отчет **ОстаткиМатериаловПоСвойствам** и запустите конструктор схемы компоновки данных. Добавьте новый **Набор данных – запрос** и вызовите конструктор запроса.

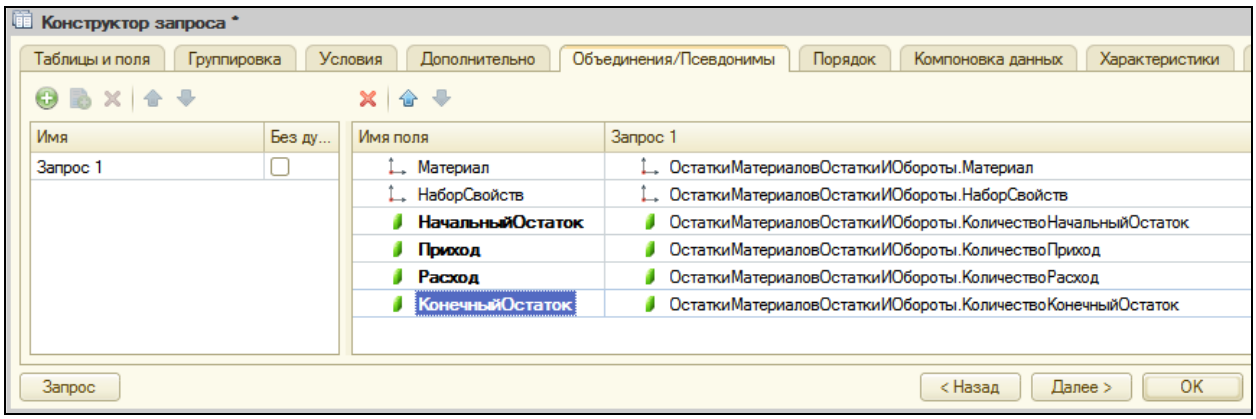
Запрос для набора данных

Источником данных для запроса будет виртуальная таблица регистра накопления **ОстаткиМатериалов.ОстаткиИОбороты.** Из нее выберем поля:

1. Материал,
2. НаборСвойств,
3. КоличествоНачальныйОстаток,
4. КоличествоПриход,
5. КоличествоРасход,
6. КоличествоКонечныйОстаток.



После этого на вкладке **Объединения/Псевдонимы** задайте псевдонимы числовых полей, убрав из них слово **Количество**:



Характеристики

Теперь приступим к описанию характеристик. Для этого перейдем на закладку **Характеристики** и нажмем кнопку **Добавить**.

Первое поле – Тип. Выбираем **СправочникСсылка.ВариантыНоменклатуры**.

Следующее – описание того, откуда система компоновки будет получать список характеристик. Для этого нужно указать источник списка характеристик и описать назначение конкретных полей этого источника.

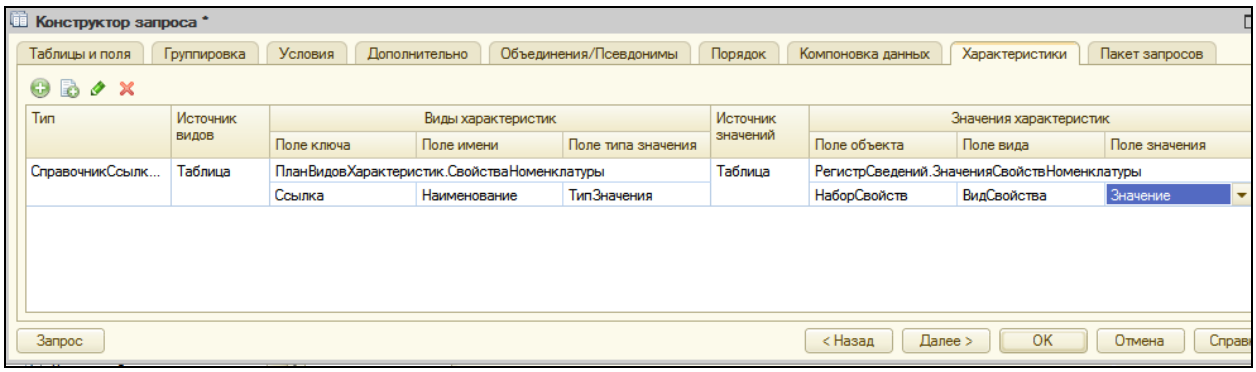
В качестве источника укажем **Таблица**, в поле Виды характеристик выберем **ПланВидовХарактеристик.СвойстваНоменклатуры**.

Далее следует описание назначения полей источника списка характеристик. Это стандартные реквизиты плана видов характеристик **СвойстваНоменклатуры**.

В **Поле ключа** выберем **Ссылка**, в **Поле имени** – **Наименование**, в **Поле типа значения** – **ТипЗначения**.

Опишем источник значений характеристик. В нашем случае это регистр сведений **ЗначенияСвойствНоменклатуры**, поэтому в поле источник значений **– Таблица,** в поле **Значение характеристик** – **РегистрСведений.ЗначенияСвойствНоменклатуры**.

В **Поле объекта** – **НаборСвойств**, **Поле вида** – **ВидСвойства**, **Поле значения** – ресурс регистра **Значение**.



Анализ текста запроса

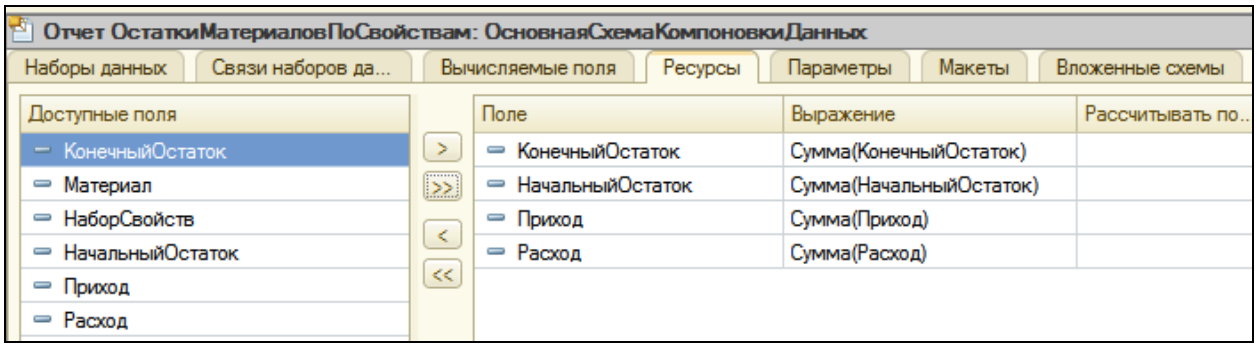
```
ВЫБРАТЬ
	ОстаткиМатериаловОстаткиИОбороты.Материал,
	ОстаткиМатериаловОстаткиИОбороты.НаборСвойств,
	ОстаткиМатериаловОстаткиИОбороты.КоличествоНачальныйОстаток КАК
НачальныйОстаток,
	ОстаткиМатериаловОстаткиИОбороты.КоличествоПриход КАК Приход,
	ОстаткиМатериаловОстаткиИОбороты.КоличествоРасход КАК Расход,
	ОстаткиМатериаловОстаткиИОбороты.КоличествоКонечныйОстаток КАК
КонечныйОстаток
ИЗ
	РегистрНакопления.ОстаткиМатериалов.ОстаткиИОбороты КАК
ОстаткиМатериаловОстаткиИОбороты
{ХАРАКТЕРИСТИКИ
	ТИП(Справочник.ВариантыНоменклатуры)
	ВИДЫХАРАКТЕРИСТИК ПланВидовХарактеристик.СвойстваНоменклатуры
	ПОЛЕКЛЮЧА Ссылка
	ПОЛЕИМЕНИ Наименование
	ПОЛЕТИПАЗНАЧЕНИЯ ТипЗначения
	ЗНАЧЕНИЯХАРАКТЕРИСТИК РегистрСведений.ЗначенияСвойствНоменклатуры
	ПОЛЕОБЪЕКТА НаборСвойств
	ПОЛЕВИДА ВидСвойства
	ПОЛЕЗНАЧЕНИЯ Значение }
```

Секция ХАРАКТЕРИСТИКИ описывает для системы компоновки те характеристики, которые будут использованы в данном отчете. Текст этой секции заключен в фигурные скобки. Это означает, что он не является частью запроса, а представляет собой конструкцию для системы компоновки.

Ресурсы

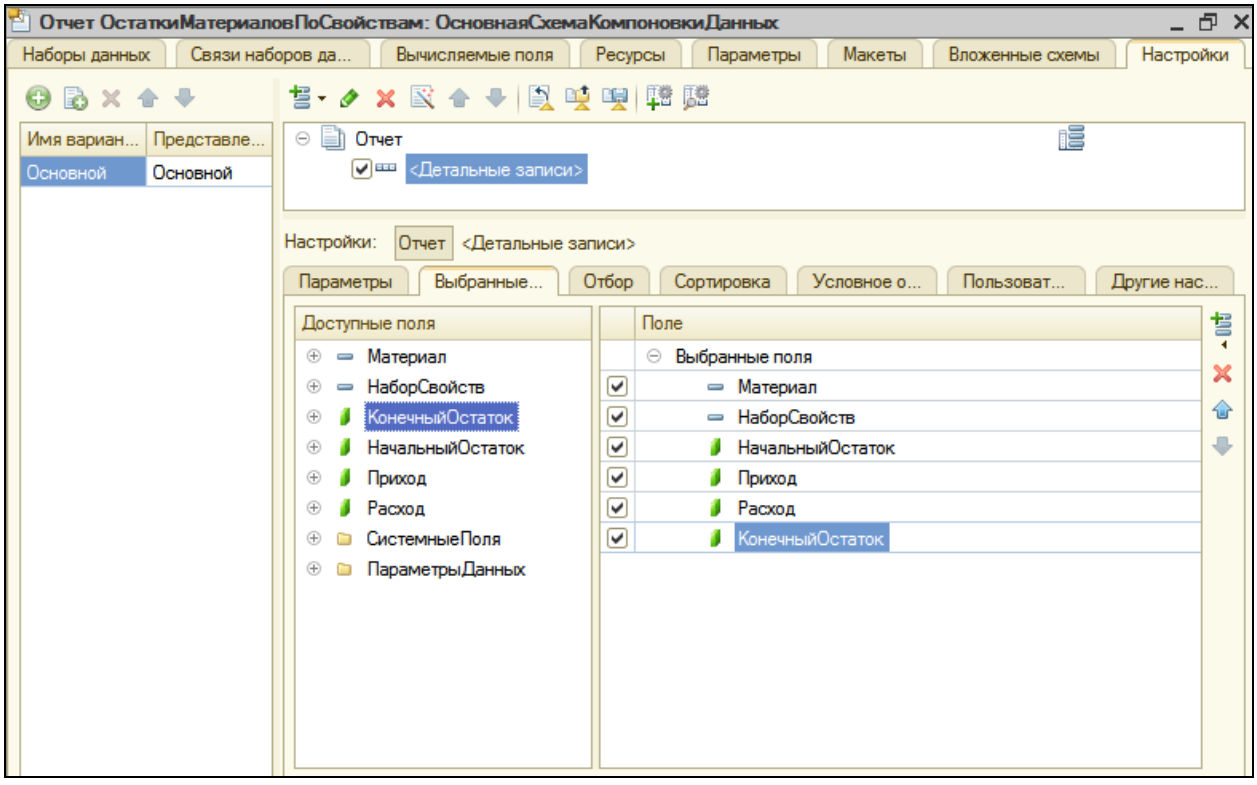
Приступим к редактированию схемы компоновки данных.

На закладке **Ресурсы** выберем все доступные ресурсы.



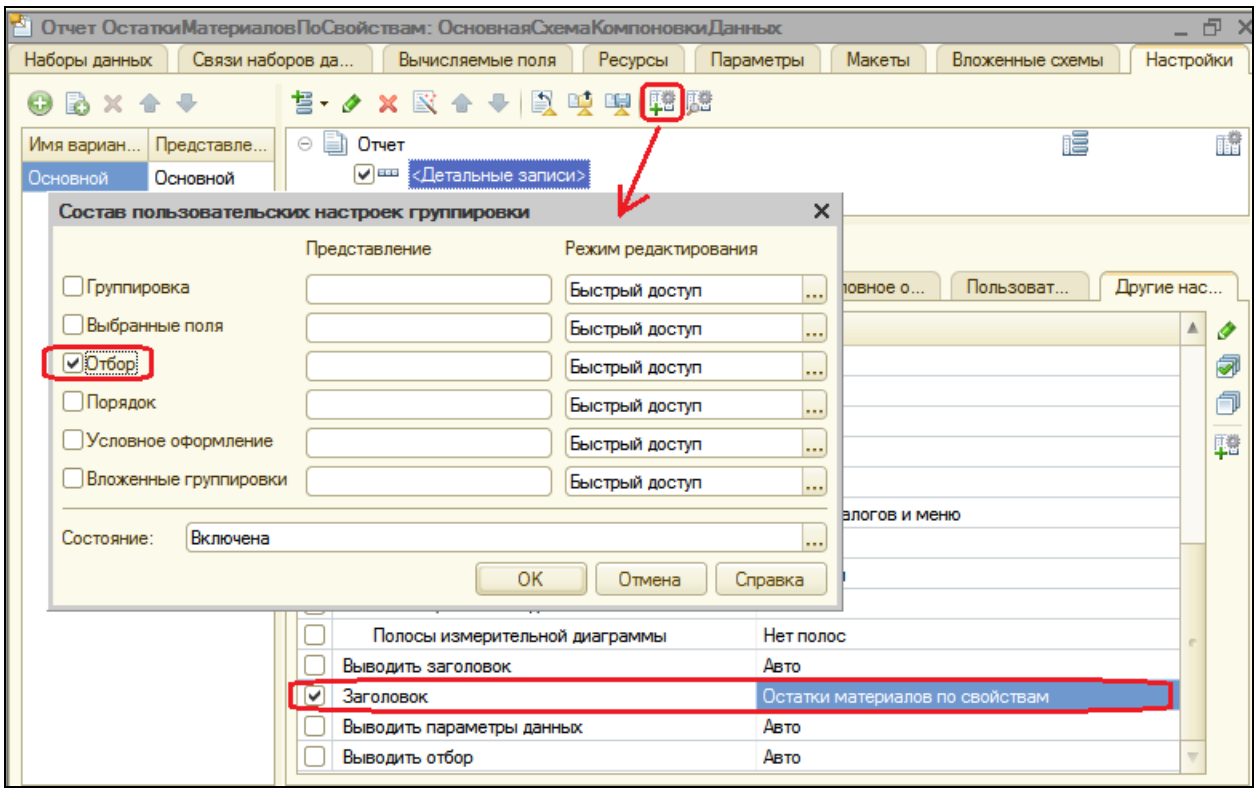
Настройки

Перейдем на закладку **Настройки**. Создадим структуру отчета – добавим группировку **Детальные записи**. Затем на закладке **Выбранные поля** выберем поля: **Материал, НаборСвойств, НачальныйОстаток, Приход, Расход, КонечныйОстаток**.



Затем перейдем на закладку **Другие настройки** и зададим заголовок отчета – **Остатки материалов по свойствам**.

Чтобы иметь возможность протестировать наш отчет, включим настройку Отбор в состав быстрых пользовательских настроек.



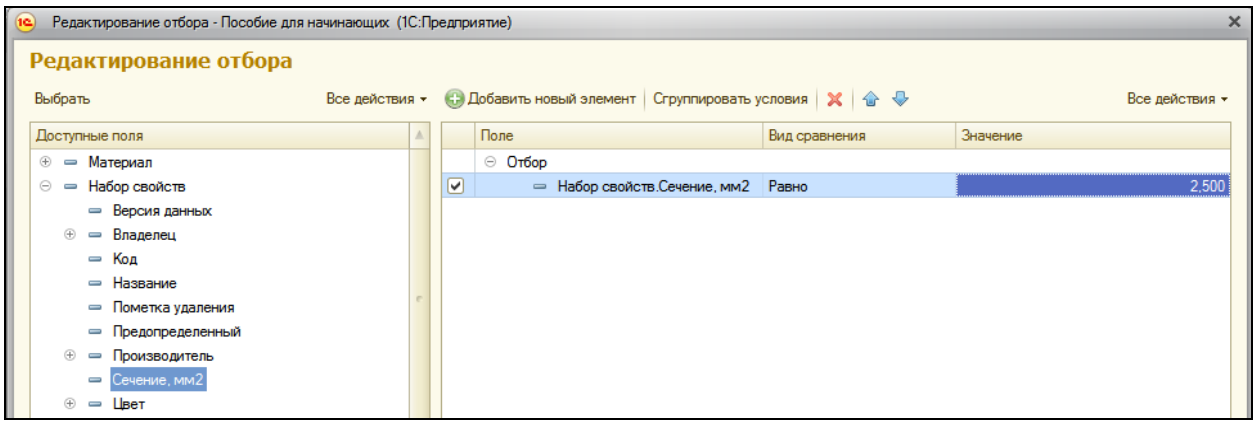
В заключение определим подсистемы, где будет отображаться наш отчет. На вкладке **Подсистемы** отметьте **УчетМатериалов** и **Бухгалтерия**. На этом создание отчета завершено, перейдите в режим отладки.

В режиме 1С:Предприятие

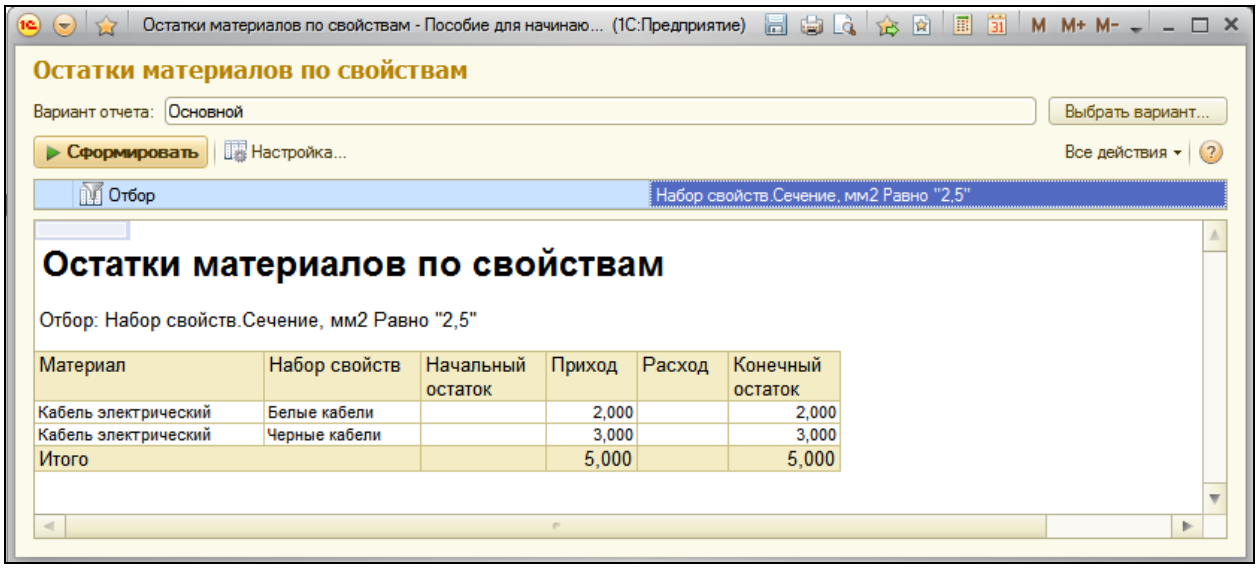
В разделе **Учет материалов** откройте отчет **Остатки материалов по свойствам**.

Сначала посмотрим, какие у нас есть материалы с сечением 2,5 мм^2.

Для этого в поле настройки **Отбор** нажмите кнопку выбора ..., раскройте поле **Набор свойств**. Выберите поле **Сечение, мм2** и задайте для него условие равенства **2,5**. Нажмите ОК.

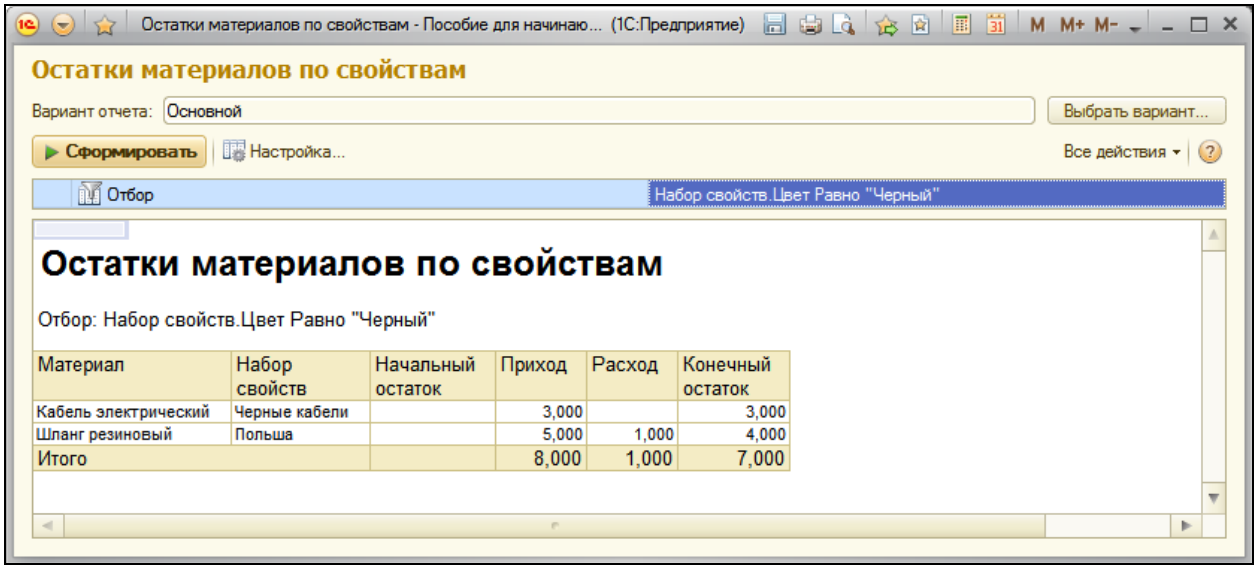


В окне отчета нажмите **Сформировать** и получим результат.

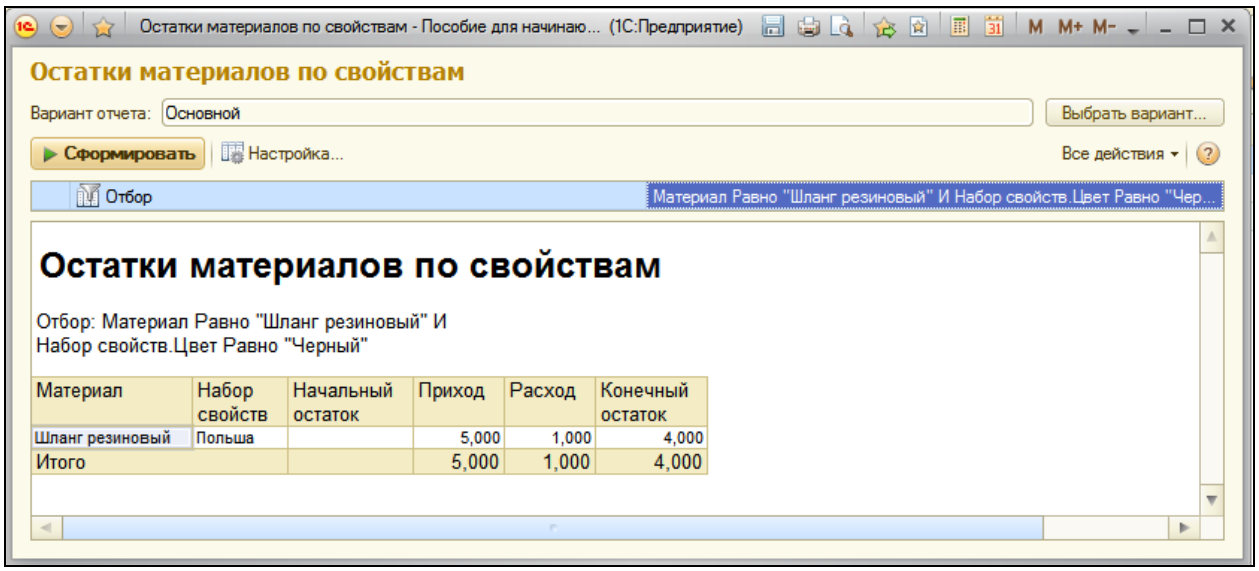


Затем посмотрим, какие у нас есть материалы черного цвета. Для этого в поле настройки Отбор еще раз нажмем кнопку выбора и удалим прежний отбор кнопкой удалить  над списком условий отбора.

Затем выберем из списка полей **Цвет**, в поле **Значение** нажмите кнопку выбора и выберем **Черный**. Нажмите ОК и **Сформировать**.



В заключение посмотрим, сколько у нас резиновых шлангов черного цвета. Создайте новый отбор по полю **Материал**, значение – **Шланг резиновый** и тут же **Цвет** значение **Черный**. Сформируйте отчет.



Следует заметить, что пример, рассмотренный нами в этой работе, не является законченным решением для данной конфигурации. Для полноценного ее использования необходимо внести соответствующие изменения в остальные регистры, документы и некоторые отчеты.

Контрольные вопросы

- ✓ Для чего предназначен объект конфигурации План видов характеристик.
- ✓ В чем принципиальное отличие Плана видов характеристик от Справочника.
- ✓ Что такое тип значения характеристик.
- ✓ Как создать план видов характеристик.
- ✓ Как, используя план видов характеристик, организовать учет по переменному количеству характеристик.
- ✓ Что такое связь по параметрам выбора.
- ✓ Как задать синоним стандартного реквизита.
- ✓ Как изменить заголовок формы.
- ✓ Как скрывать элементы формы с подчиненной информацией при ее создании.
- ✓ Как использовать характеристики при выполнении отчета.

Практическая работа № 15

Бухгалтерский учёт (1:50)

В этой работе мы покажем возможность ведения бухучета средствами 1С: Предприятия. Для организации бухучета мы используем уже знакомый нам план видов характеристик и два новых объекта – *План счетов* и *Регистр бухгалтерии*.

Регистр бухгалтерии будет использоваться для накопления данных о совершенных хозяйственных операциях.

С помощью плана счетов мы будем описывать счета, в разрезе которых ведется учет, а план видов характеристик будет служить для описания объектов аналитического учета, в разрезе которых должен вестись учет на счетах.

План счетов в данной работе будет очень сильно упрощен. Он содержит всего несколько условных счетов, которые позволят познакомиться с основными методами организации бухучета средствами 1С: Предприятия.

Бухучет подразумевает ведение аналитического учета на большинстве счетов. Для обозначения разрезов аналитического учета мы будем использовать термин *виды субконто*. Т.е. на каждом счете учет может вестись в разрезе нескольких видов субконто. А для обозначения конкретных объектов аналитического учета мы будем использовать термин *субконто*.

Для ведения аналитического учета в системе 1С:Предприятие применяется механизм субконто. Субконто называется любой объект аналитического учета: основные средства, нематериальные активы, малоценные и быстроизнашивающиеся предметы, материалы, организации, подотчетные лица, договоры, бюджеты

Видом субконто, в свою очередь, называется множество однотипных объектов аналитического учета. Например, список покупателей и заказчиков в системе 1С:Предприятие будет называться "видом субконто "Организации", а любая организация из этого списка будет именоваться "субконто".

Для организации видов субконто в системе 1С:Предприятие используются объекты метаданных "Виды субконто". Конфигуратор позволяет организовывать любое необходимое число видов субконто.

Субконто — термин, обозначающий аналитический признак («разрез») счета бухгалтерского учёта. Низший иерархический элемент в структуре бухгалтерских счетов. Используется в бухгалтерских программах. От subcount = субсчет. Например в программе 1С — аналитический признак бухгалтерского счета. По назначению схож с субсчетом.

Субконто - аналитический показатель, позволяющий расшифровать счет по заранее неизвестному или постоянно меняющемуся набору строк - справочнику.

Например, на 41 счете (**Товары**) учет ведется обычно в разрезе **Номенклатуры** и **Складов**, которые являются видами субконто. А вот конкретная номенклатура **Паста шоколадная** и конкретный склад **Основной**, указанные для некоторой проводки по 41 счету, – это субконто.

Добавление Плана видов характеристик

В режиме Конфигуратор

Приступим к созданию плана видов характеристик, который будет содержать описания разрезов аналитического учета – видов субконто.

Откройте конфигуратор и добавьте новый объект **План видов характеристик** с именем **ВидыСубконто**.

На закладке **Подсистемы** укажите, что план счетов будет отображаться в подсистеме **Бухгалтерия**.

Поскольку нам понадобится некий вспомогательный справочник, в котором пользователи будут осуществлять «свободное творчество» по созданию значений новых объектов аналитического учета, добавим объект конфигурации **Справочник** с именем **Субконто**.

Затем на закладке **Владельцы** укажем, что этот справочник будет подчинен плану видов характеристик **ВидыСубконто**. (Добавить – ВидыСубконто). Закройте свойства справочника и вернитесь к плану видов характеристик.

На закладке Основные установите свойство **Тип значения характеристик**. Нажмите кнопку выбора и задайте составной тип данных:

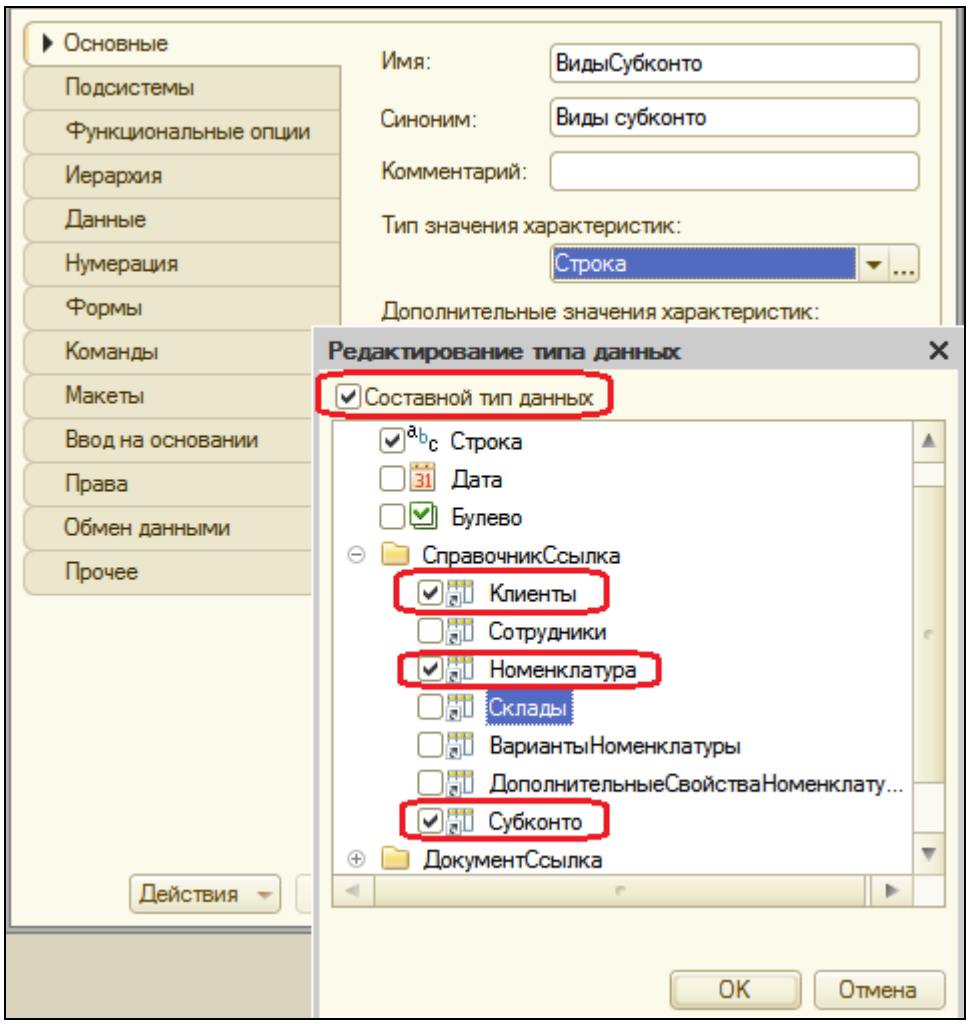
- СправочникСсылка.Клиенты,
- СправочникСсылка.Номенклатура,
- СправочникСсылка.Субконто.

Бухгалтерия нашей фирмы ведет учет движения денежных средств только в разрезе материалов и клиентов, но не исключено, что в

дальнейшем понадобится дополнительная аналитика, поэтому мы и используем справочник Субконто.

Обратите внимание, что тот справочник, который будет использован в качестве дополнительных значений характеристик, тоже должен входить в составной тип данных типа значений характеристик, иначе конфигуратор выдаст ошибку.

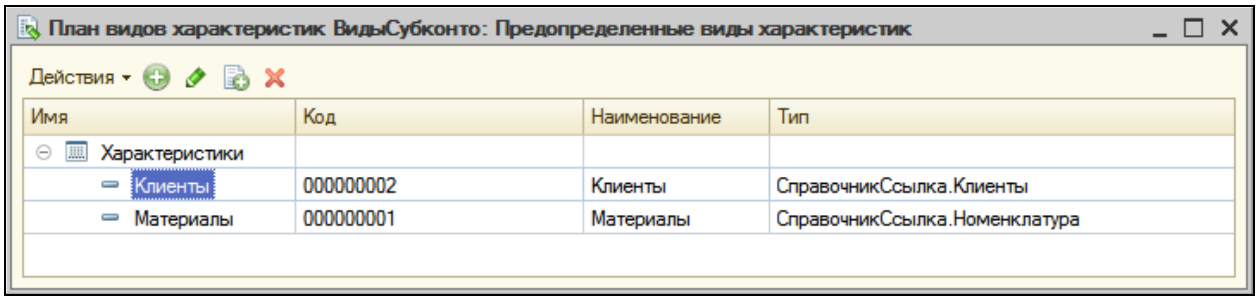
Затем укажем, что дополнительные значения характеристик будут находиться в справочнике Субконто.



После этого перейдите на вкладку **Прочее**, нажмите кнопку **Предопределенные**, введите значения плана видов характеристик по умолчанию:

Нажимая кнопку **Добавить**, создайте предопределенный вид субконто **Материалы** с кодом 000000001 и типом **СправочникСсылка.Номенклатура**.

Затем создайте вид субконто **Клиенты** с кодом 000000002 и типом **СправочникСсылка.Клиенты**. На этом создание видов субконто завершено.



Что такое План счетов

План счетов предназначен для описания структуры хранения информации о совокупности синтетических счетов предприятия, которые созданы для группировки данных его хозяйственной деятельности.

На основе плана счетов платформа создает в базе данных таблицы, в которых будет храниться информация о том, какие счета и каким образом будет использовать предприятие. Это может быть система бухгалтерских счетов, установленная государством, план управленческих счетов или произвольный набор счетов, используемых для анализа тех или иных видов деятельности предприятия.

План счетов в системе 1С:Предприятие поддерживает иерархию субсчетов: к каждому счету первого уровня может быть открыто несколько субсчетов, которые в свою очередь могут иметь свои субсчета и т.д.

По любому счету или субсчету может вестись аналитический учет в разрезе субконто, описанных в плане видов характеристик. Связь между планов счетов и планом видов характеристик задается разработчиком на этапе конфигурирования.

Для описания используемых субконто система создает в плане счетов специальную табличную часть ВидыСубконто, которая не видна в конфигураторе (но доступна средствами встроенного языка).

Для каждого счета есть возможность задать несколько видов учета (например, количественный и валютный). Виды учета задаются при помощи подчиненных объектов конфигурации *признак учета*.

Также можно определить несколько видов учета субконто (суммовой, валютный, количественный). Виды учета субконто задаются при помощи подчиненных объектов *признак учета субконто*.

Помимо этого, каждый счет может иметь набор свойств, которые задаются в качестве реквизитов объекта План счетов. Они позволяют определять уникальные свойства элементов плана счетов (например, реквизит ЗапретитьИспользоватьВПроводках).

Добавление плана счетов

В режиме Конфигуратор

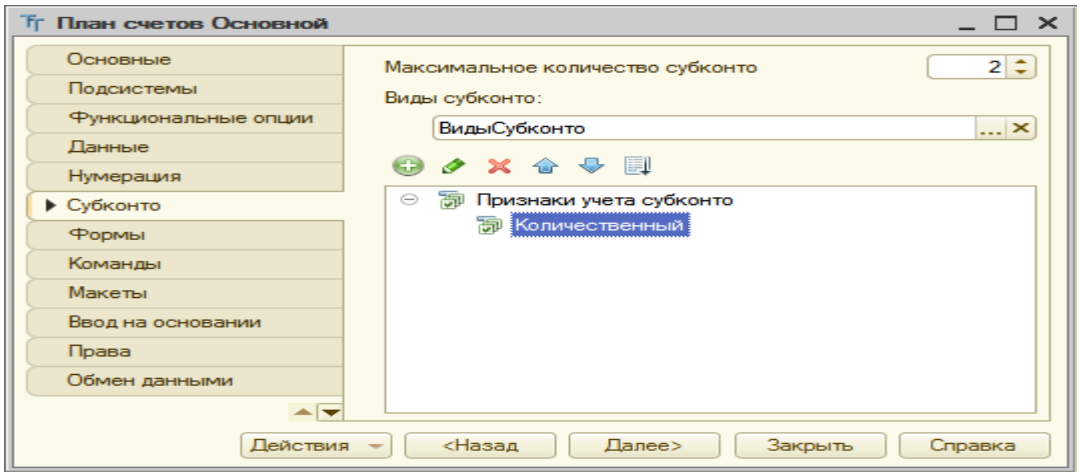
Приступим к созданию плана счетов. Он будет содержать всего четыре счета:

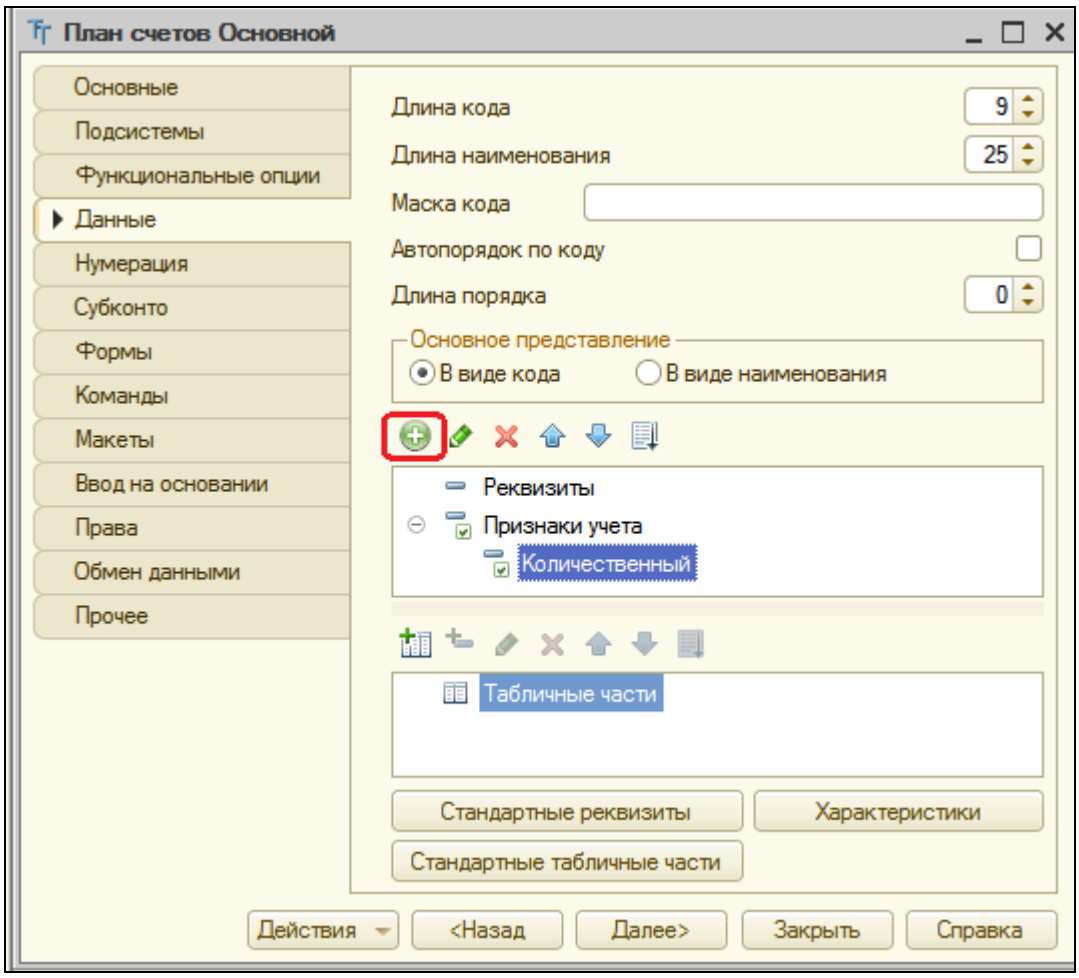
- Товары,
- РасчетыСПоставщиками,
- Капитал,
- Дебиторская задолженность.

Добавим новый объект **План счетов** с именем **Основной**. Свойство представления списка задайте как **Основной план счетов**. На закладке **Подсистемы** укажите **Бухгалтерия**.

На закладке **Данные** выделите группу реквизитов **Признаки учета** и создайте признак учета **Количественный**.

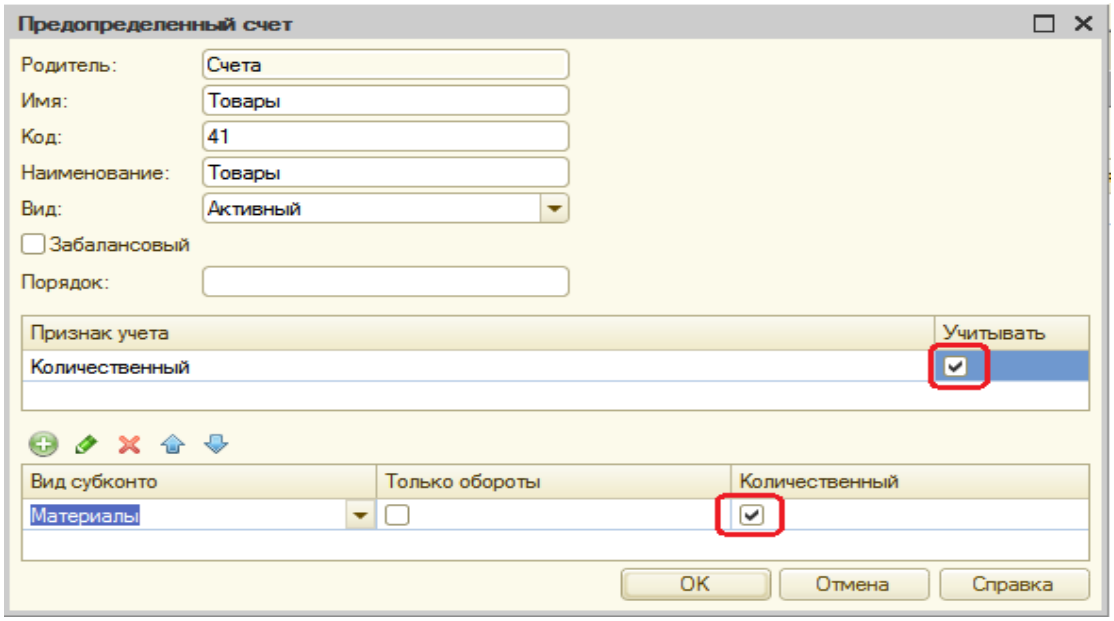
Перейдите на вкладку **Субконто** и укажите, что виды субконто для этого плана счетов будут находиться в плане видов характеристик **ВидыСубконто**. Максимальное количество субконто установите равным двум. Также создадим признак учета субконто **Количественный**.



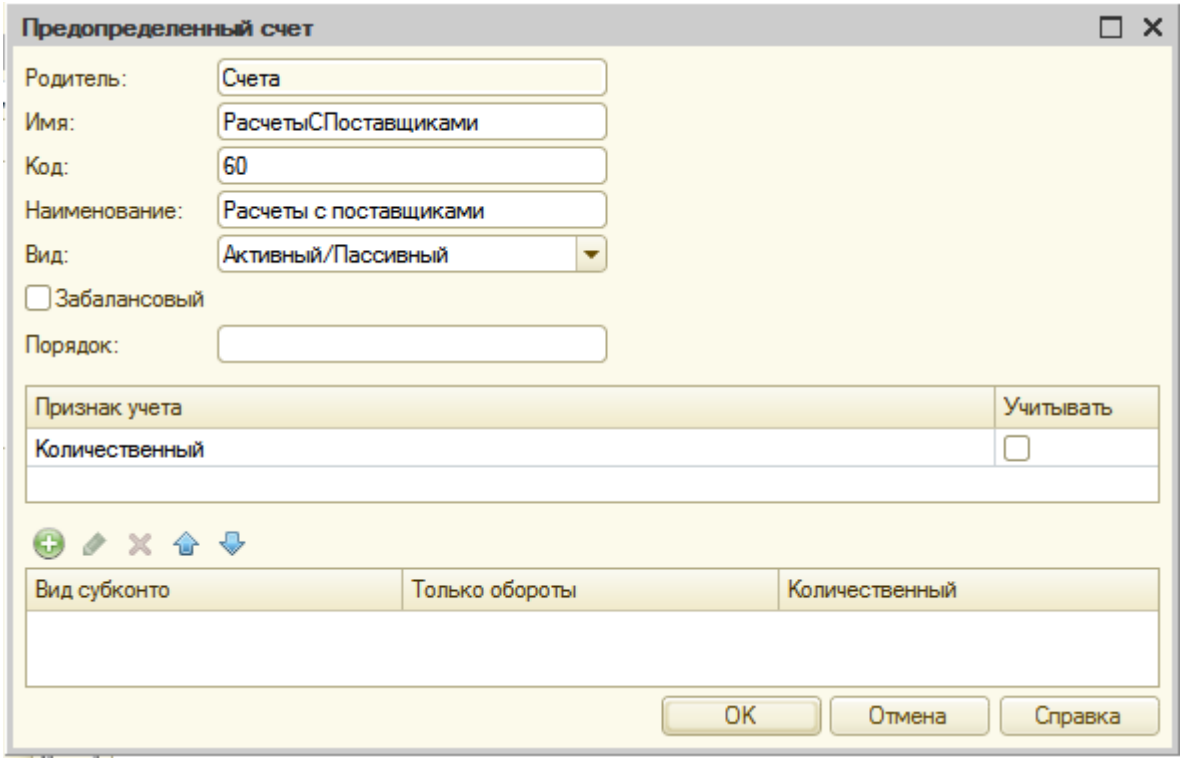


Затем откройте вкладку **Прочее**, нажмите кнопку **Предопределенные** и создайте четыре предопределенных счета:

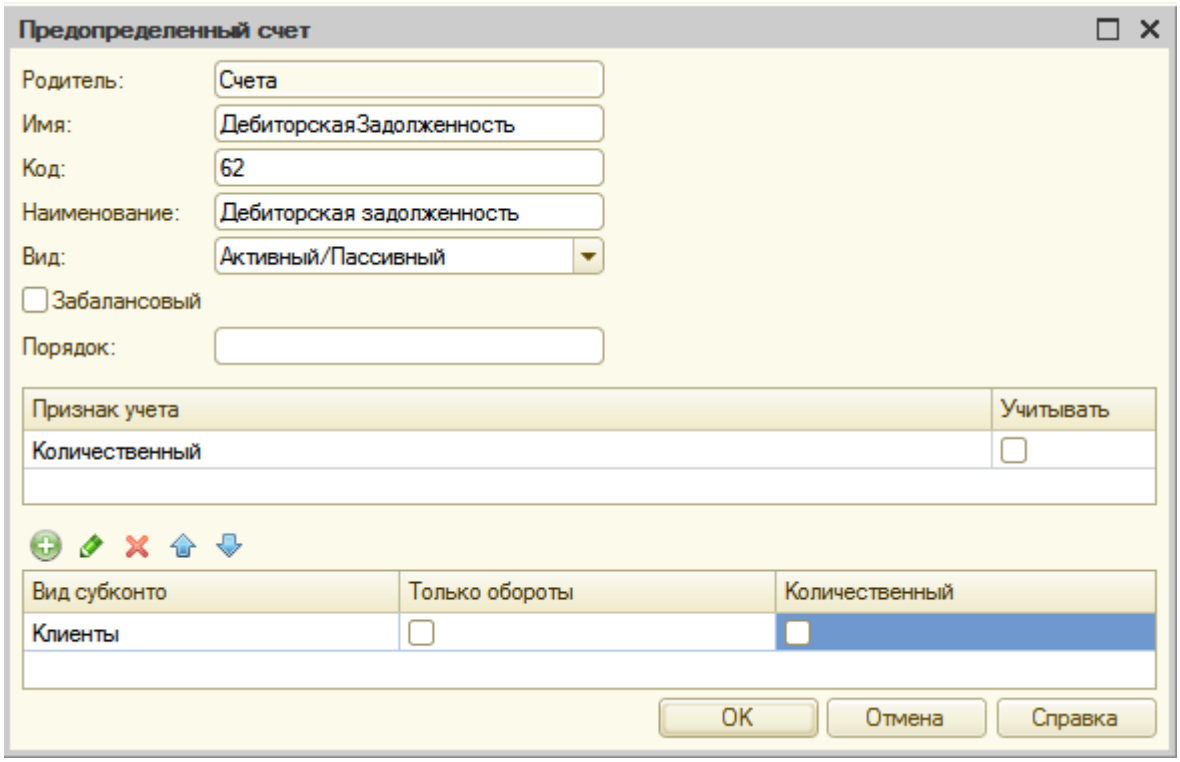
- **Товары**, код **41**, **активный**, с видом субконто **Материалы** (количественным учетом в разрезе материалов);



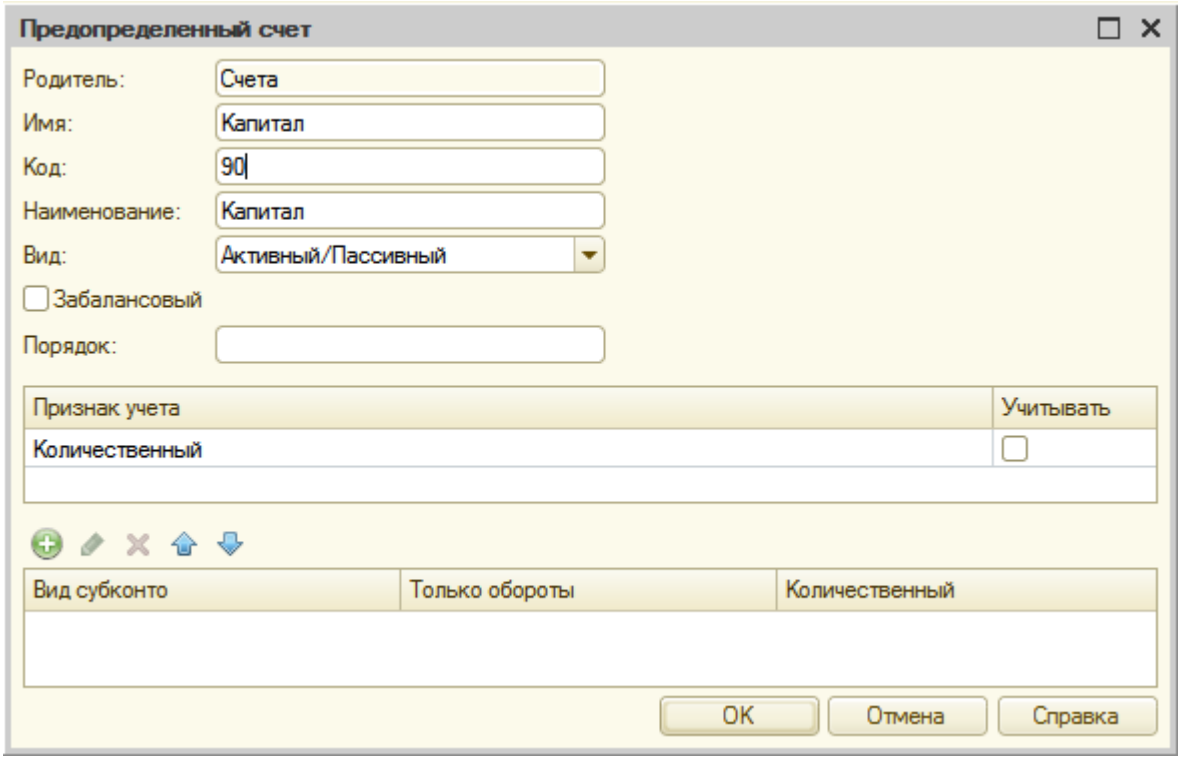
- РасчетыСПоставщиками, код 60, активный/пассивный;



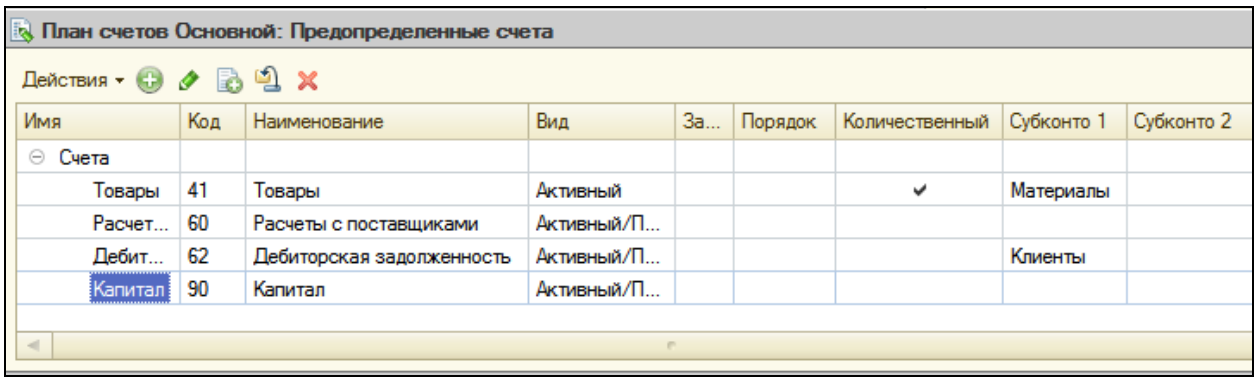
- **ДебиторскаяЗадолженность**, код **62**, **активный/пассивный**, в разрезе клиентов (субконто **Клиенты**);



- **Капитал**, код **90**, **активный/пассивный**.



В результате наш план счетов будет выглядеть следующим образом:



Теперь мы можем перейти к знакомству с *Регистром бухгалтерии*.

Что такое Регистр бухгалтерии

Объект *Регистр бухгалтерии* предназначен для описания структуры накопления данных, учет которых ведется исходя из некоторого плана счетов.

На его основе платформа создает в базе данных таблицу, в которой будут накапливаться данные о хозяйственных операциях, отображаемых в бухучете.

По виду регистр бухгалтерии напоминает регистр накопления – он также имеет ресурсы, может иметь измерения и реквизиты.

Измерения позволяют разделить ведение учета (например, используя измерение Организация, можно вести учет в разрезе нескольких юридических лиц).

Реквизиты служат признаком, по которому одни записи регистра можно отделить от других (например, в качестве реквизита может использоваться номер журнала, что позволит отбирать проводки, имеющие одинаковый смысл).

Значительное отличие от регистра накопления в том, что регистр бухгалтерии имеет жесткую связь с используемым планом счетов. Поэтому каждая запись регистра бухгалтерии содержит дополнительные поля, определяемые настройкой используемого плана счетов. А также возможность поддержки механизма двойной записи, при которой каждая запись регистра содержит обязательные поля для указания счета дебета и счета кредита.

Добавление регистра бухгалтерии

В режиме Конфигуратор

Создайте новый объект **Регистр бухгалтерии** с именем **Управленческий**.

Свойство **Расширенное представление списка** задайте как **Движения в регистре Управленческий**.

Укажите, что с ним будет связан план счетов **Основной**.

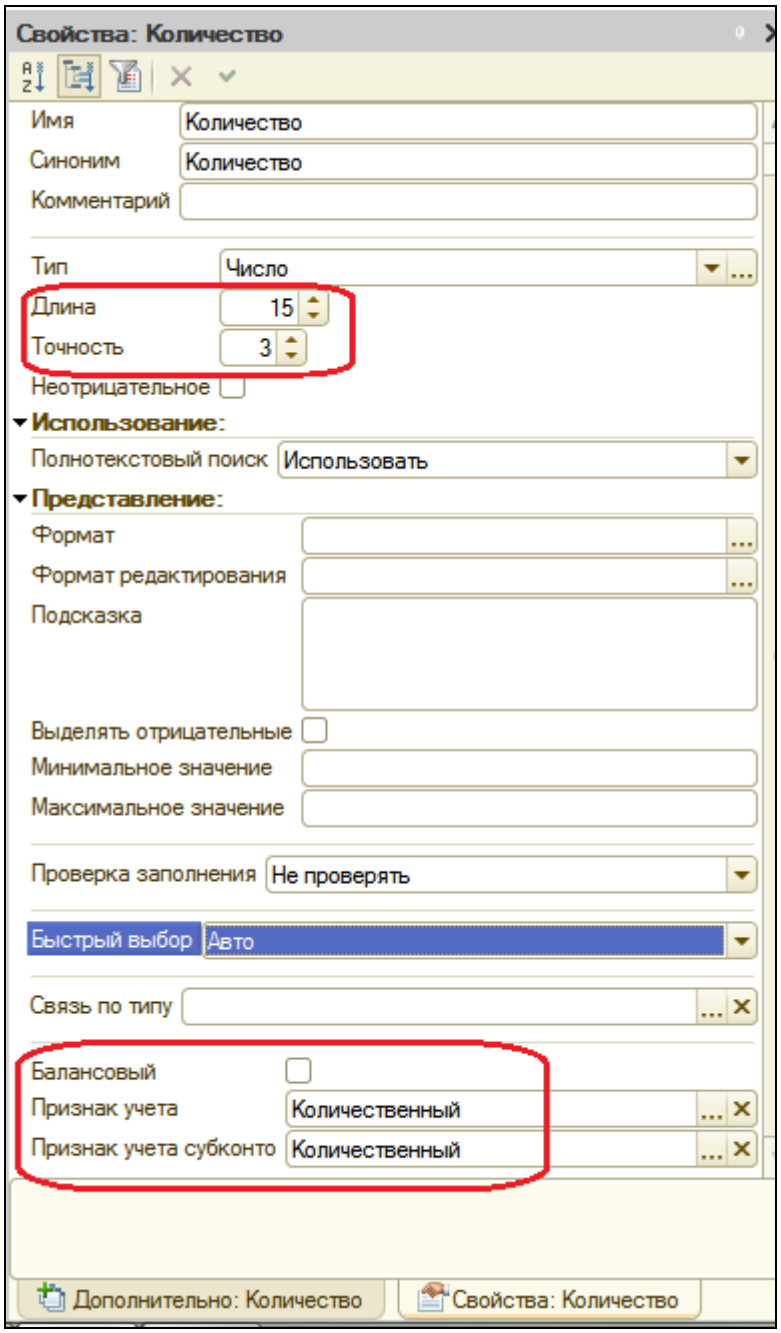
Установите флажок **Корреспонденция**. Он говорит о том, что создаваемый регистр поддерживает корреспонденции, т.е. каждая запись регистра имеет дебетовую и кредитовую часть, что позволит получать информацию не только об остатках и оборотах по счетам, но и о корреспонденциях между счетами.

Регистры, не поддерживающие корреспонденцию, применяются, когда не нужно использовать принцип двойной записи, регламентированный в бухучете и контролировать баланс хозяйственных средств и их источников.

Перейдите на закладку Данные, создайте два ресурса:

- Сумма, длина 15, точность 2, Балансовый;

- Количество, длина 15, точность 3, Небалансовый, признак учета – Количественный, признак учета субконто – Количественный.



На этом создание регистра бухгалтерии завершено.

Доработка приходной накладной

Сначала мы доработаем оба наших документа (**ПриходнаяНакладная** и **ОказаниеУслуги**) так, чтобы они поставляли

данные не только для регистров накопления, но и для регистра бухгалтерии.

Затем мы создадим бухгалтерский отчет **Оборотно-сальдовая** ведомость, который будет показывать нам состояние товародвижения в фирме, основываясь на данных регистра бухгалтерии.

Сначала изменим процедуру проведения документа **ПриходнаяНакладная**, а затем в режиме отладки перепроведем все эти документы.

В режиме Конфигуратор

В окне редактирования документа **ПриходнаяНакладная** перейдите на вкладку **Движения**.

В списке регистров отметим, что документ будет создавать движения и по регистру бухгалтерии **Управленческий**.

Перейдите на закладку **Прочее** и откройте модуль объекта. Откройте процедуру обработчика события **ОбработкаПроведения**. Перед началом цикла установите свойство Записывать набора записей регистра Управленческий в значение Истина для записи изменений регистра в БД. В самом конце цикла перед строкой **КонецЦикла** добавим строки, создающие движения регистра **Управленческий**.

```
// регистр ОстаткиМатериалов Приход
	Движения.ОстаткиМатериалов.Записывать = Истина;
	Движения.СтоимостьМатериалов.Записывать = Истина;
	Движения.Управленческий.Записывать = Истина;

	Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
		Движение = Движения.ОстаткиМатериалов.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Материал = ТекСтрокаМатериалы.Материал;
		Движение.НаборСвойств = ТекСтрокаМатериалы.НаборСвойств;
		Движение.Склад = Склад;
		Движение.Количество = ТекСтрокаМатериалы.Количество;
		// регистр Стоимость Материалов Приход
		Движение = Движения.СтоимостьМатериалов.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Материал = ТекСтрокаМатериалы.Материал;
		Движение.Стоимость = ТекСтрокаМатериалы.Сумма;
		//регистр Управленческий
		Движение = Движения.Управленческий.Добавить();
		Движение.СчетДт = ПланыСчетов.Основной.Товары;
```

```
            Движение.СчетКт =
ПланыСчетов.Основной.РасчетыСПоставщиками;
            Движение.Период = Дата;
            Движение.Сумма = ТекСтрокаМатериалы.Сумма;
            Движение.КоличествоДт = ТекСтрокаМатериалы.Количество;

      Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Материалы] = ТекСтрокаМатериалы.Материал;



      КонецЦикла;
```

Как видите, движения для регистра бухгалтерии формируются так же, как и для регистра накопления. Но платформа добавила к созданным нами реквизитам регистра еще ряд полей, которые явились следствием использования плана счетов **Основной**. Прежде всего это поля **СчетДт** и **СчетКт**. В этих полях указываются счета, дебет и кредит которых затрагивает данная проводка.

Кроме этого, для измерений и ресурсов регистра, связанных с признаками учета, платформа создает пару полей для хранения значения каждого ресурса отдельно по дебету и отдельно по кредиту проводки – **КоличествоДт** и **КоличествоКт**. А также для счетов, по которым ведется учет в разрезе субконто, платформа создает коллекции **СубконтоДт** и **СубконтоКт**.

К счету мы обращаемся с помощью свойства глобального контекста **ПланыСчетов**. Оно предоставляет доступ ко всем планам счетов конфигурации.

Через точку от него мы указываем имя нужного нам плана счетов – **Основной**. А далее через точку – имя предопределенного счета в этом плане счетов – **Товары**. Этот счет и три других мы создали в конфигураторе.

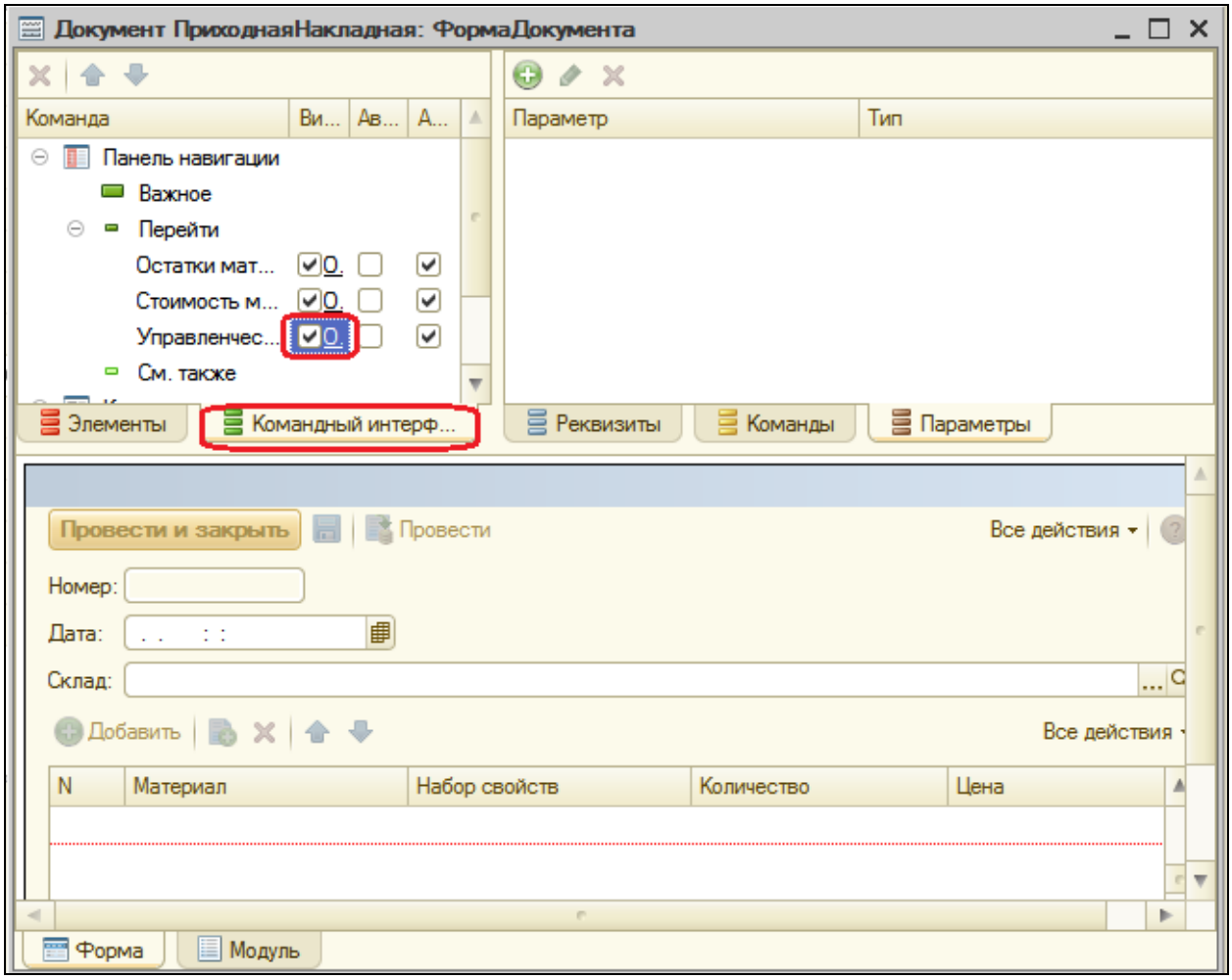
Т.к. количественный учет ведется только для счета **Товары** (41), то поле регистра **КоличествоДт** заполняется количеством товара из табличной части документа. Поле регистра **КоличествоКт** не заполняется, т.к. по счету кредита проводки (**РасчетыСПоставщиками**) количественный учет не ведется.

Для каждой записи движения регистра бухгалтерии платформа хранит две коллекции значений: субконто дебета и субконто кредита.
Обратиться к элементу коллекции можно. Указав в квадратных скобках

ссылку на соответствующий вид субконто. Другой способ – в явном виде указать имя предопределенного вида субконто через точку от коллекции субконто дебета. Т.е. запись
`Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Материалы]`
равносильна записи `Движение.СубконтоДт.Материалы`.

Коллекция регистра **СубконтоКт** не заполняется, т.к. по счету кредита проводки учет в разрезе субконто не ведется.

В заключение отредактируйте командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра **Управленческий**, связанному с документом. Для этого откройте форму документа **ПриходнаяНакладная**, в левом верхнем окне перейдите на закладку **командный интерфейс**. В разделе **Панель навигации** раскройте группу **Перейти** и установите видимость для команды открытия регистра бухгалтерии **Управленческий**.

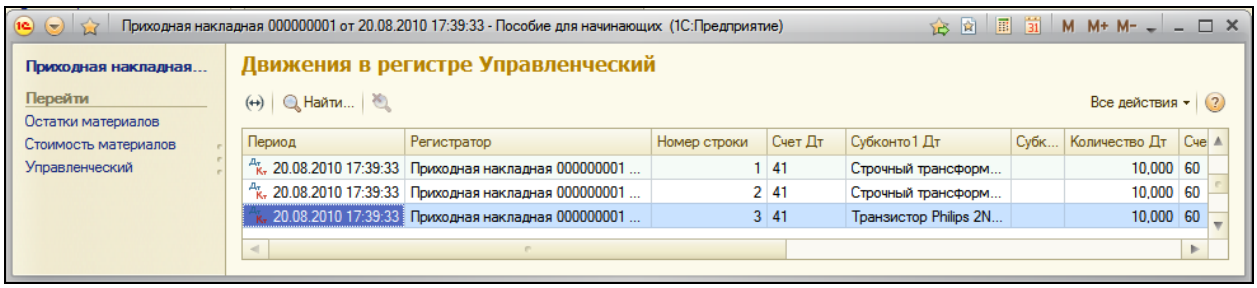


В режиме 1С:Предприятие

Запустите режим отладки, проигнорировав сообщение, что регистр бухгалтерии **Управленческий** и справочник **Субконто** не включены ни в одну подсистему.

Откройте документ **Приходная накладная №1** и нажмите **Провести**.

Выполните переход к регистру **Управленческий** и посмотрите, какие движения сформировал документ в регистре бухгалтерии.



Обратите внимание: поскольку в счете 60 (**РасчетыСПоставщиками**) отсутствует аналитика и ведется только учет суммы, в записях движений регистра **СубконтоКт1**, **СубконтоКт2**, **КоличествоКт** не указаны.

После этого перепроведите аналогично документа **Приходная накладная №2** и убедитесь, что он тоже формирует правильные проводки.

Доработка документа Оказание услуги

Сначала мы изменим процедуру проведения документа **ОказаниеУслуги**, а затем перепроведем все эти документы.

В режиме Конфигуратор

В отличие от документа **ПриходнаяНакладная**, который создавал всего одну бухгалтерскую проводку, документ **ОказаниеУслуги** будет создавать уже две.

Напомним, что бухгалтерия нашей фирмы сильно упрощена, поэтому услуги, которые оказывает организация, для нее как бы не существуют.

Поэтому документ **ОказаниеУслуги** будет формировать движения по регистру бухгалтерии только в той части, которая касается расходования материалов.

В окне редактирования документа **ОказаниеУслуги** перейдите на закладку **Движения**. В списке регистров отметим, что документ будет создавать теперь движения и по регистру **Управленческий**. Перейдите на закладку **Прочее** и откройте модуль объекта. Процедуру обработчика события **ОбработкаПроведения**.

Поскольку нас интересует только движение материалов, для внесения дополнений подойдет тело условия Если…, в котором мы формировали движения по регистрам **ОстаткиМатериалов** и **СтоимостьМатериалов**.

Добавим в конец условия, перед строкой КонецЕсли движения по регистру Управленческий:

```
        // регистр СтоимостьМатериалов Расход
                        Движение = Движения.СтоимостьМатериалов.Добавить();
                        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
                        Движение.Период = Дата;
                        Движение.Материал = ВыборкаДетальныеЗаписи.Номенклатура;
                        Движение.Стоимость =
ВыборкаДетальныеЗаписи.КоличествоВДокументе*СтоимостьМатериала;

                        //регистр Управленческий
                        //первая проводка: Д 62 (ДебиторскаяЗадолженность) - К
90 (Капитал) Розничная сумма
                        Движение = Движения.Управленческий.Добавить();
                        Движение.СчетДт =
ПланыСчетов.Основной.ДебиторскаяЗадолженность;
                        Движение.СчетКт = ПланыСчетов.Основной.Капитал;
                        Движение.Период = Дата;
                        Движение.Сумма =
ВыборкаДетальныеЗаписи.СуммаВДокументе;

        Движение.СубконтоДт[ПланыВидовХарактеристик.ВидыСубконто.Клиент
ы] = Клиент;
                        //вторая проводка: Д 90 (Капитал) - К 41 (Товары) -
себестоимость
                        Движение = Движения.Управленческий.Добавить();
                        Движение.СчетДт = ПланыСчетов.Основной.Капитал;
                        Движение.СчетКт = ПланыСчетов.Основной.Товары;
                        Движение.Период = Дата;
                        Движение.Сумма =
СтоимостьМатериала*ВыборкаДетальныеЗаписи.КоличествоВДокументе;
                        Движение.КоличествоКт =
ВыборкаДетальныеЗаписи.КоличествоВДокументе;

        Движение.СубконтоКт[ПланыВидовХарактеристик.ВидыСубконто.Матери
алы]=ВыборкаДетальныеЗаписи.Номенклатура;

                КонецЕсли;
      // Регистр Продажи
```

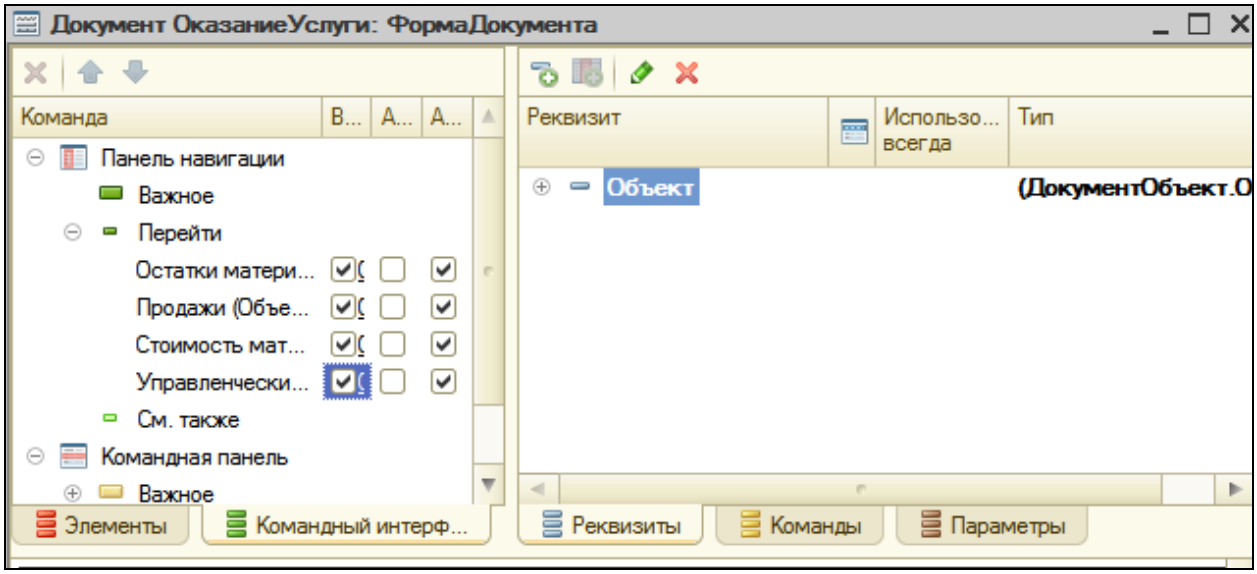
В самом начале процедуры установите свойство **Записывать** регистра бухгалтерии в значение **Истина** для записи изменений регистров в БД.

```
Процедура ОбработкаПроведения(Отказ, Режим)

	Движения.ОстаткиМатериалов.Записывать = Истина;
	Движения.СтоимостьМатериалов.Записывать = Истина;
	Движения.Продажи.Записывать = Истина;
	Движения.Управленческий.Записывать = Истина;
```

В первой проводке мы указываем розничную сумму материала из документа и субконто дебета, поскольку на счете Дебиторская задолженность ведется учет в разрезе клиентов.

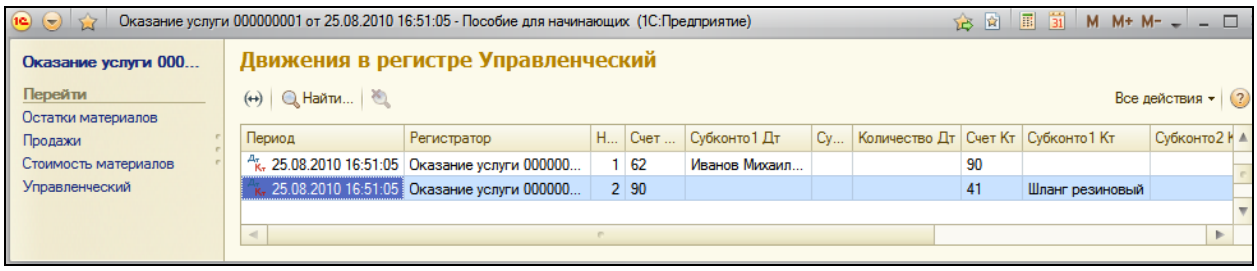
Во второй проводке мы указываем стоимость материала, количество и субконто кредита, поскольку на счете Товары ведется количественный учет в разрезе материалов.

В заключение аналогично предыдущему случаю отредактируйте командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра Управленческий, связанному с документом.



В режиме 1С:Предприятие

Запустите режим отладки, откройте документ **Оказание услуги №1** и нажмите **Провести**. Перейдите к регистру **Управленческий** и посмотрите, какие движения сформировал документ. Перепроведите все остальные документы оказания услуг.



Оборотно-сальдовая ведомость

Теперь нам только осталось создать отчет для бухгалтерии предприятия и наше знакомство с использованием регистра бухгалтерии будет закончено.

В режиме Конфигуратор

Добавьте новый объект **Отчет** с именем **ОборотноСальдоваяВедомость**. Создайте новую схему компоновки данных и добавьте **Набор данных – запрос**. Откройте конструктор запроса.

Запрос для набора данных

Бухгалтерский отчет **Оборотно-сальдовая ведомость** представляет собой таблицу, в строках которой перечислены все имеющиеся в плане счетов счета, а в колонках – начальное сальдо, оборот и конечное сальдо по дебету и кредиту каждого счета.

Сальдо в бухгалтерском учёте — остаток по бухгалтерскому счёту, разность между суммой записей по дебету и кредиту счетов.

- Дебетовое сальдо (дебет больше кредита) отражает состояние данного вида хозяйственных средств на определённую дату и показывается в активе баланса.
- Кредитовое сальдо (кредит больше дебета) отражает состояние источников хозяйственных средств и показывается в пассиве.

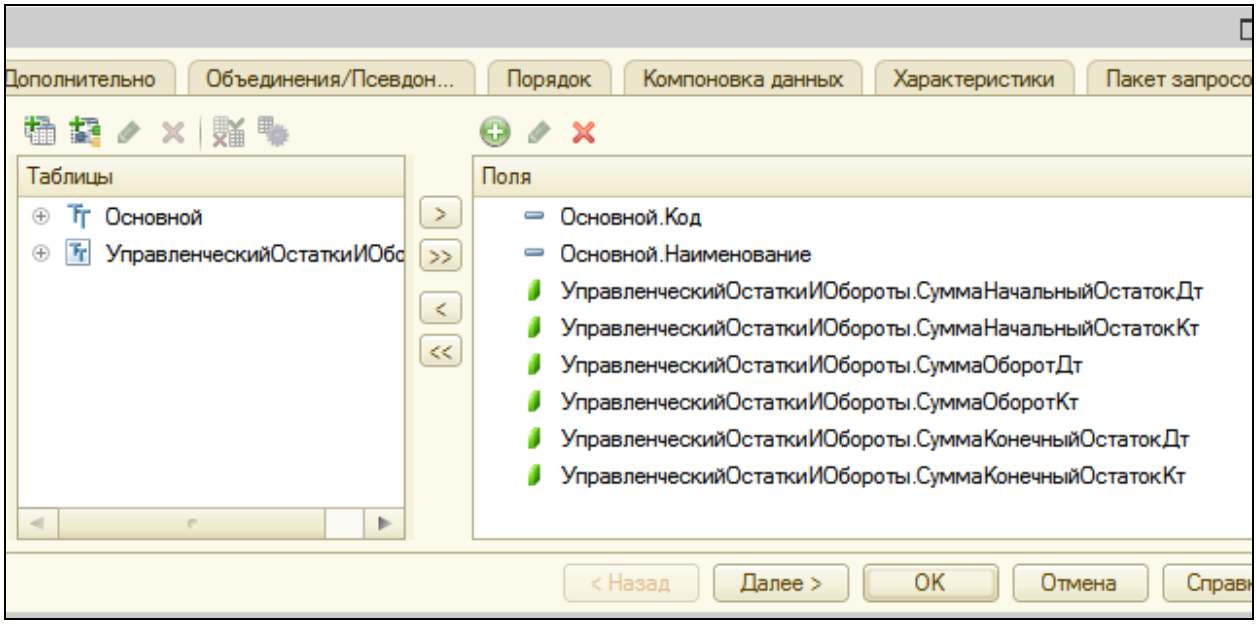
Если счёт не имеет остатка (сальдо равно нулю), то такой счёт называется закрытым. В бухгалтерском учёте некоторые счета могут одновременно иметь и дебетовое, и кредитовое сальдо.

Поэтому для построения такого отчета понадобятся две исходные таблицы:

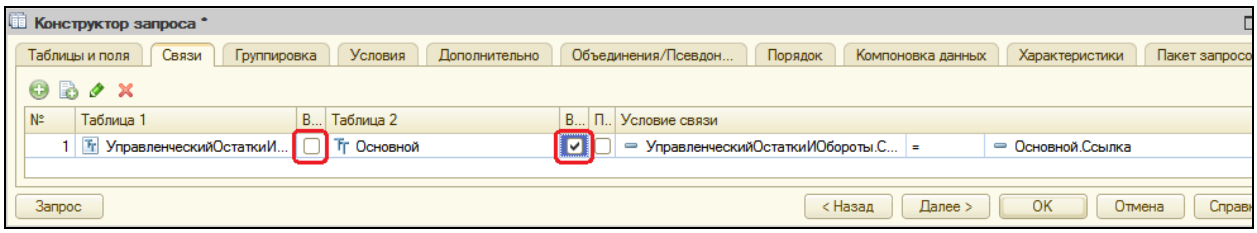
- Объектная (ссылочная) таблица плана счетов Основной;
- Виртуальная таблица регистра бухгалтерии **Управленческий.ОстаткиИОбороты**.

Из таблицы **Основной** выберем поля **Код** и **Наименование**, из таблицы **Управленческий.ОстаткиИОбороты** поля:

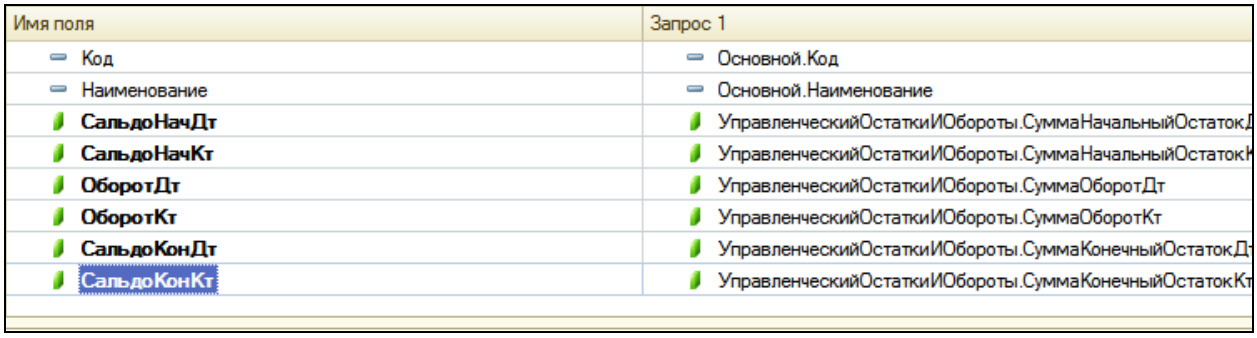
- **СуммаНачальныйОстатокДт**,
- **СуммаНачальныйОстатокКт**,
- **СуммаОборотДт**,
- **СуммаОборотКт**,
- **СуммаКонечныйОстатокДт**,
- **СуммаКонечныйОстатокКт**.



Перейдите на закладку **Связи** и укажите, что из таблицы **Основной** мы будем выбирать все записи, а из таблицы регистра – только те, которые соответствуют условию связи.



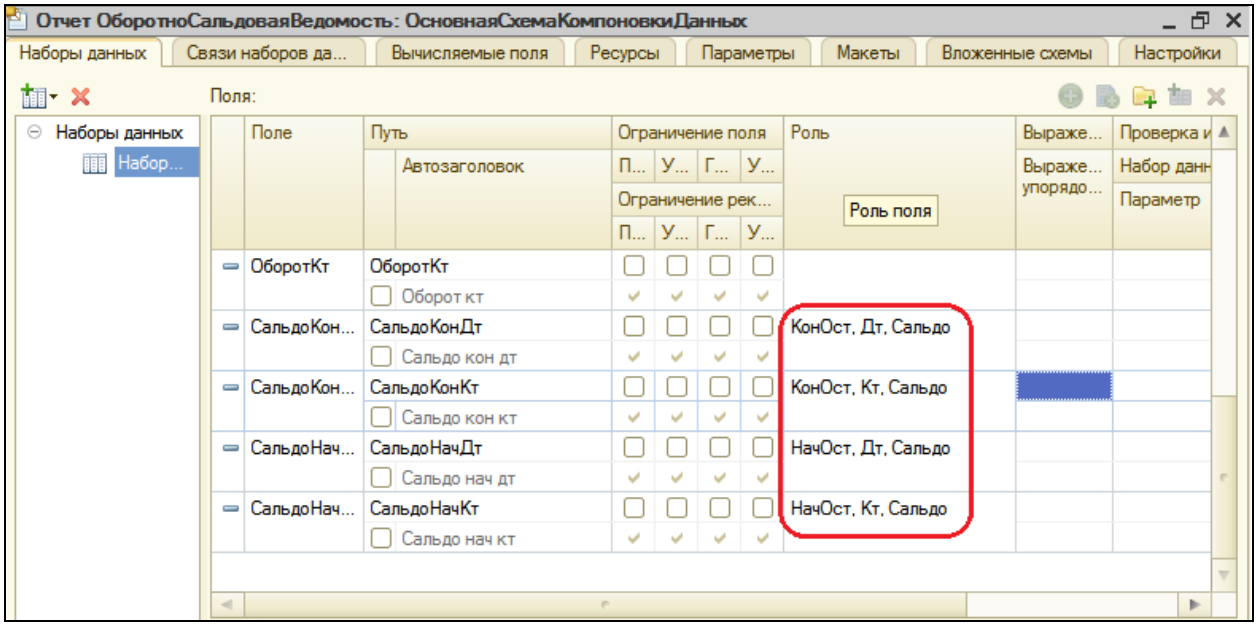
Затем на вкладке **Объединения/Псевдонимы** задайте псевдонимы полей регистра: **СальдоНачДт, СальдоНачКт, ОборотДт, ОборотКт, СальдоКонДт, СальдоКонКт**.



На вкладке **Порядок** укажите, что результат запроса должен быть отсортирован по возрастанию поля **Код**. Нажмите ОК.

Роли полей остатка

Теперь, чтобы схема компоновки могла отобразить общие итоги по полям бухгалтерских остатков, внесем небольшие изменения в роли, которые она автоматически определила для полей остатка.



Для этих полей система определила бухгалтерский тип – **Дебет** и **Кредит**.

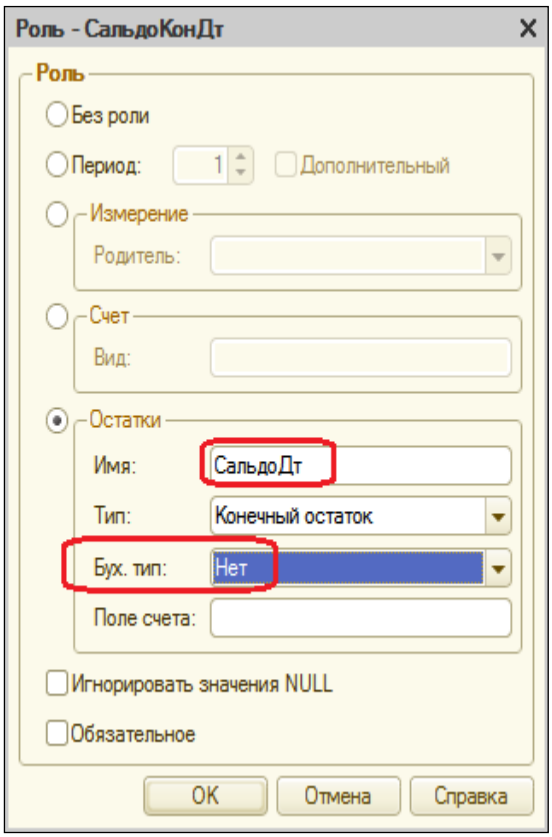
Поэтому, когда в нашей оборотно-сальдовой ведомости будет рассчитываться общий итог по этим полям, мы получим значение 0, т.к. сумма по дебету будет равна сумме по кредиту, только с обратным знаком.

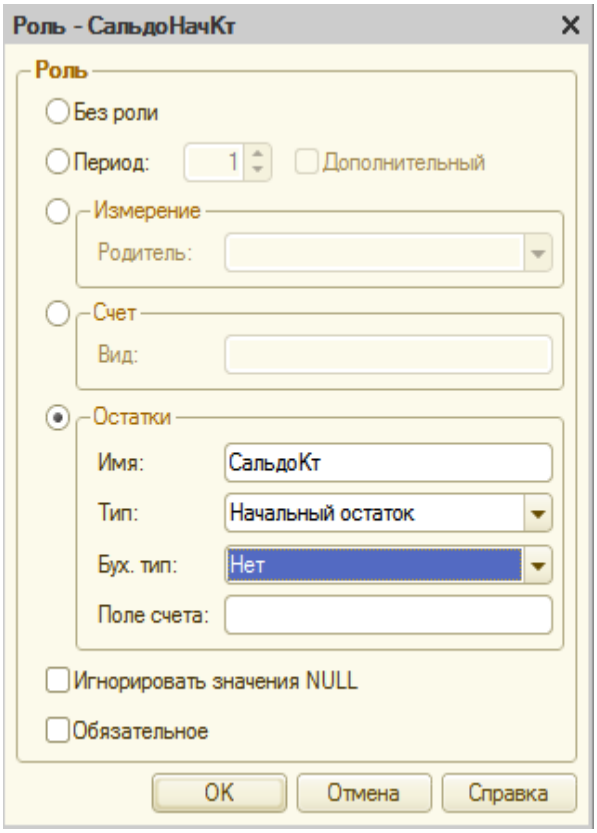
Для того, чтобы избежать такой ситуации, в ролях этих полей мы уберем указание бухгалтерского типа и изменим имена групп полей. В этом случае система компоновки будет воспринимать эти поля как обычные поля остатков.

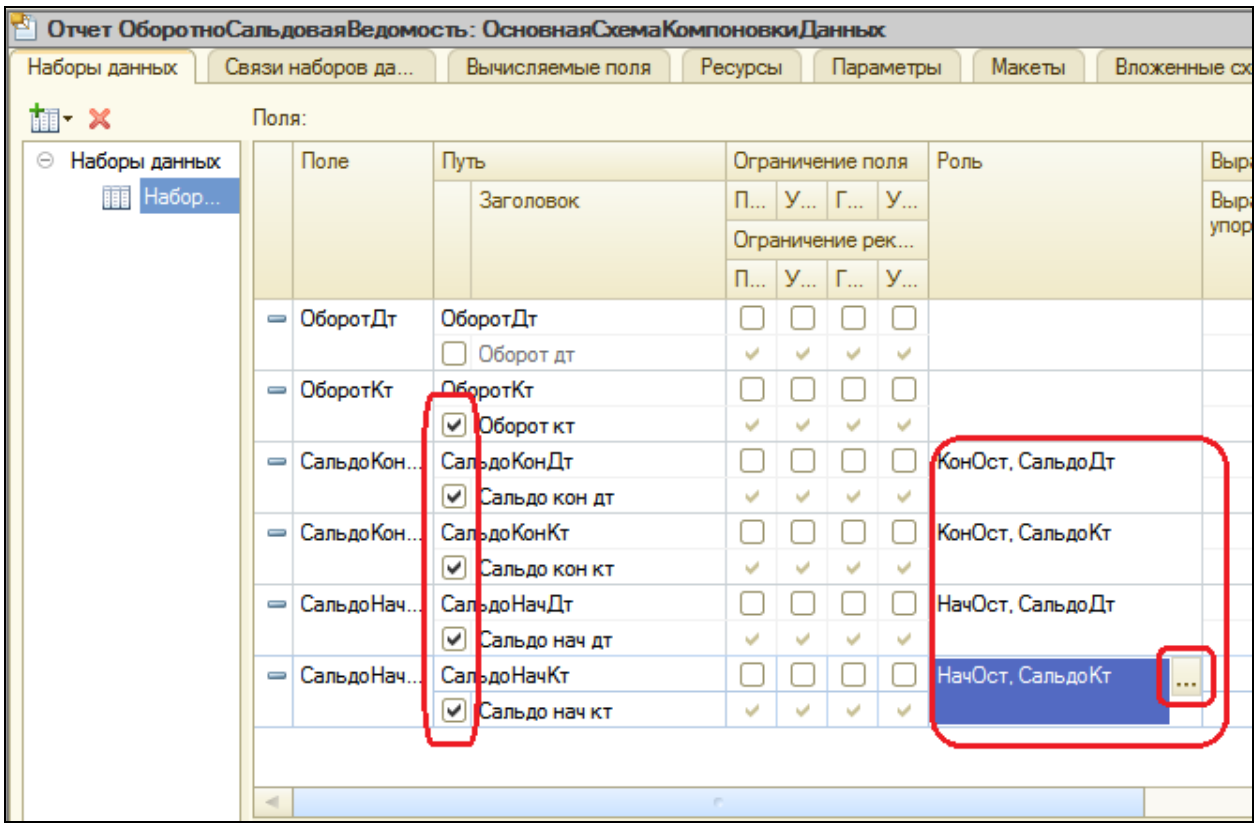
Дважды щелкните на записи в поле **Роль** для ее редактирования и нажмите кнопку выбора.

Для полей **СальдоНачДт** и **СальдоКонДт** задайте имя – **СальдоДт**. Для полей **СальдоНачКт** и **СальдоКонКт** имя **СальдоКт**.

Для всех четырех полей установите **Бухгалтерский тип** в значение **Нет**.

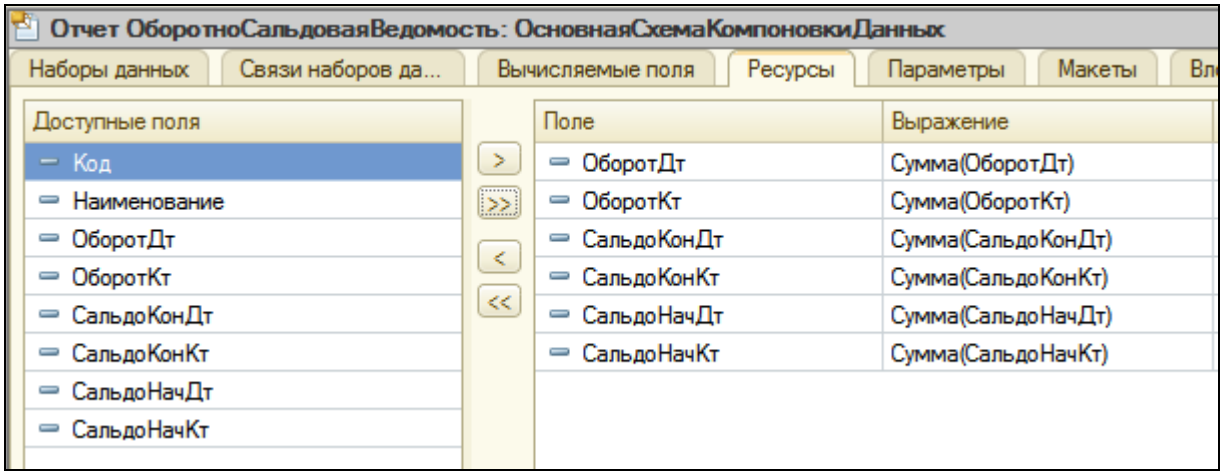






Ресурсы

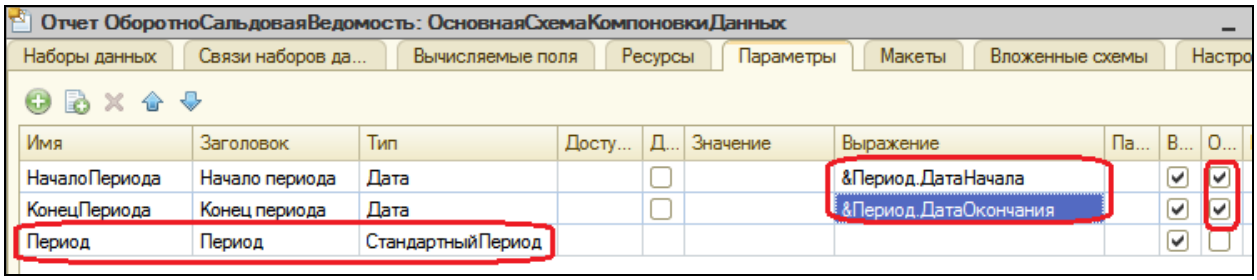
Перейдите на закладку **Ресурсы** и добавьте все доступные ресурсы кнопкой >>.



Параметры

Бух. отчеты формируются, как правило, для определенного периода: месяц, квартал, год. Поэтому на примере нашего отчета продемонстрируем использование стандартного периода для указания периода отчета.

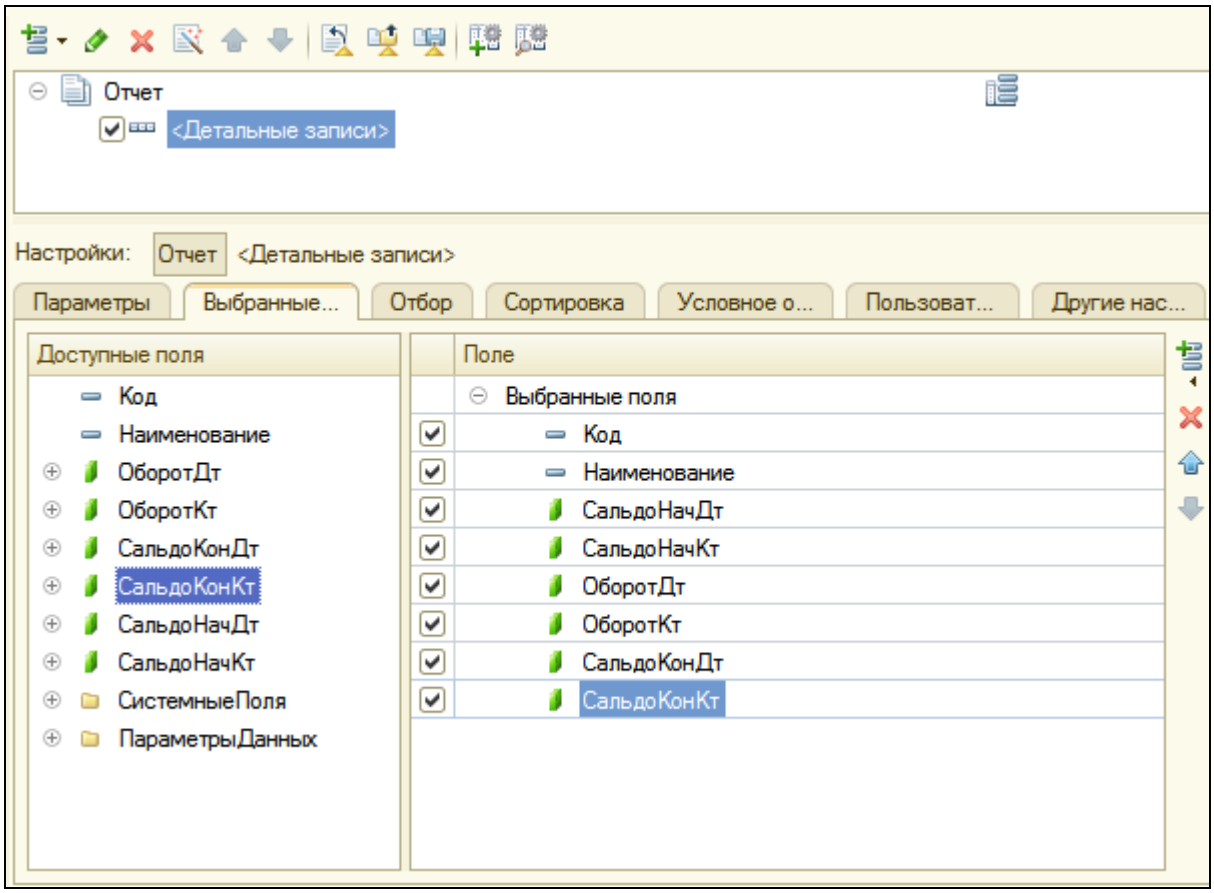
На закладке **Параметры** добавим параметр с именем **Период** типа **СтандартныйПериод**, а для параметров **НачалоПериода** и **КонецПериода** укажем **Выражение** &Период.ДатаНачала и &Период.ДатаОкончания соответственно для расчета и запретим их редактирование пользователем.



Настройки

Перейдите на закладку **Настройки** и создайте структуру отчета.

Добавим группировку, содержащую детальные записи. Затем на закладке **Выбранные поля** выберем все поля для вывода в отчет и разместим их в следующем порядке:



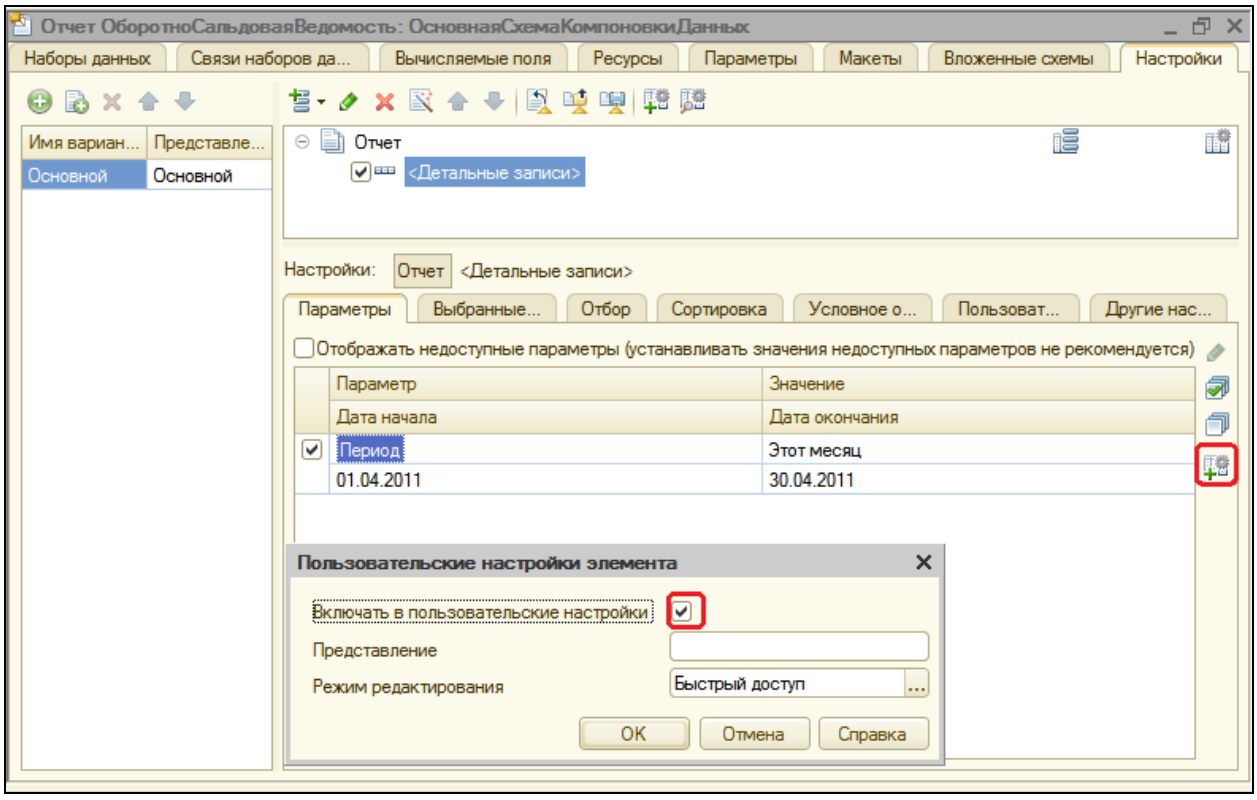
На закладке **Другие настройки** укажем заголовок отчета – **Оборотно-сальдовая ведомость**.

Для параметра **Расположение общих итогов по вертикали** укажем значение **Начало и конец**.

Затем на закладке **Параметры** выберем для параметра **Период** значение из списка стандартных периодов – **Этот месяц**.

Тем самым мы обеспечим указание текущего месяца при открытии формы отчета с динамическим изменением даты.

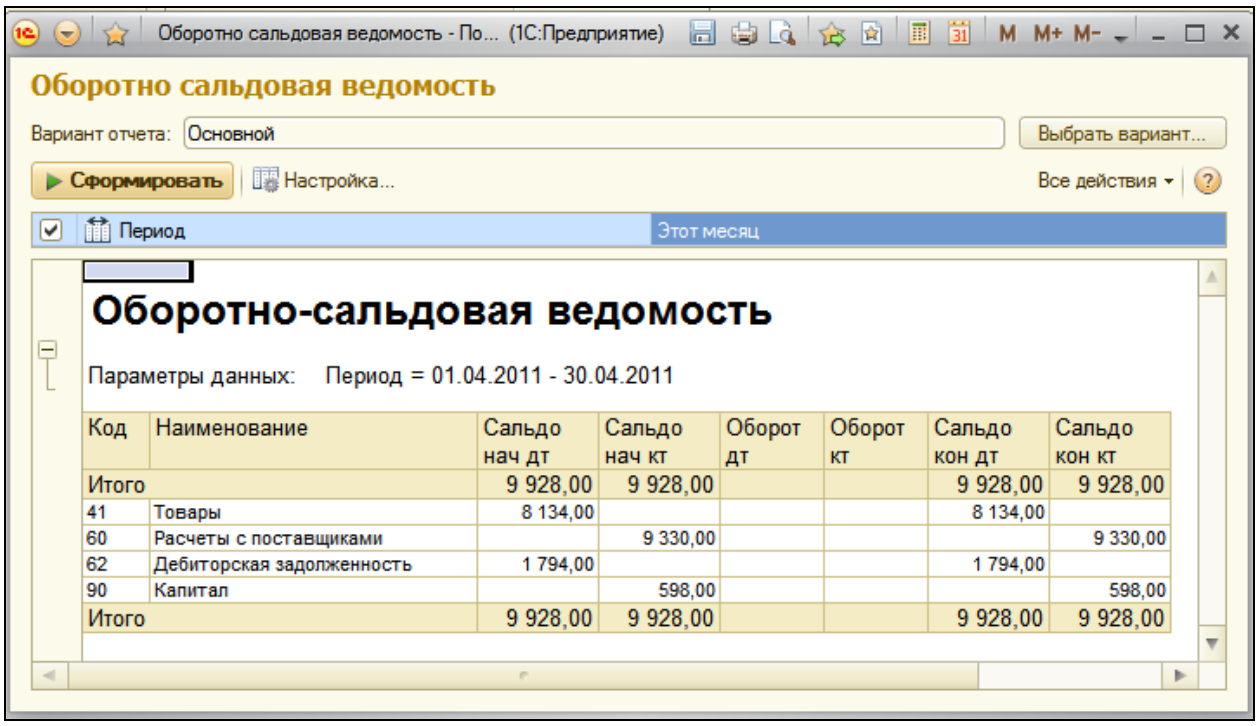
Нажмите кнопку **Свойства элемента пользовательских настроек**, укажите, что параметр **Период** будет включен в состав быстрых пользовательских настроек.



В заключение определим, что отчет будет отображаться в подсистеме **Бухгалтерия**.

В режиме 1С:Предприятие

Запустите отладку, откройте в разделе **Бухгалтерия** отчет и нажмите **Сформировать**.



Код	Наименование	Сальдо нач дт	Сальдо нач кт	Оборот дт	Оборот кт	Сальдо кон дт	Сальдо кон кт
Итого		9 928,00	9 928,00			9 928,00	9 928,00
41	Товары	8 134,00				8 134,00	
60	Расчеты с поставщиками		9 330,00				9 330,00
62	Дебиторская задолженность	1 794,00				1 794,00	
90	Капитал		598,00				598,00
Итого		9 928,00	9 928,00			9 928,00	9 928,00

Мы видим, что стандартный период отчета задан по умолчанию – Этот месяц.

Контрольные вопросы

- ✓ Как использовать план видов характеристик для организации ведения бухучета.
- ✓ Что такое субконто.
- ✓ Для чего предназначен объект План счетов.
- ✓ Как создать план счетов.
- ✓ Для чего предназначен Регистр бухгалтерии.
- ✓ Как создать регистр бухгалтерии и настроить параметры учета.
- ✓ Как создать движения документа по регистру бухгалтерии средствами встроенного языка.
- ✓ Как получить данные из регистра бухгалтерии запросом.
- ✓ Как создать отчет на основании данных из регистра бухгалтерии с помощью системы компоновки.
- ✓ Как задать роли и тип бухгалтерского остатка полям в схеме компоновки данных.
- ✓ Как задать стандартный период для выполнения отчета.

Практическая работа № 16

План видов расчета, регистр расчета (1:00)

В этой работе мы познакомимся с объектами конфигурации *План видов расчета* и *Регистр расчета* и узнаем об основных понятиях, используемых при создании сложных периодических расчетов. В конце работы мы создадим план видов расчета и регистр расчета, на основе которых в следующей работе продемонстрируем работу механизмов периодических расчетов.

Такие расчеты используются, прежде всего, при расчете зарплаты. Поэтому дальнейшее их рассмотрение мы будем строить на примере расчета зарплаты сотрудников нашей фирмы.

Сумма зарплаты складывается из множества частей (оплата по окладу, премии, штрафы, больничные, разовые выплаты и т.д.), которые могут зависеть друг от друга. Назовем такие части *вид расчета*.

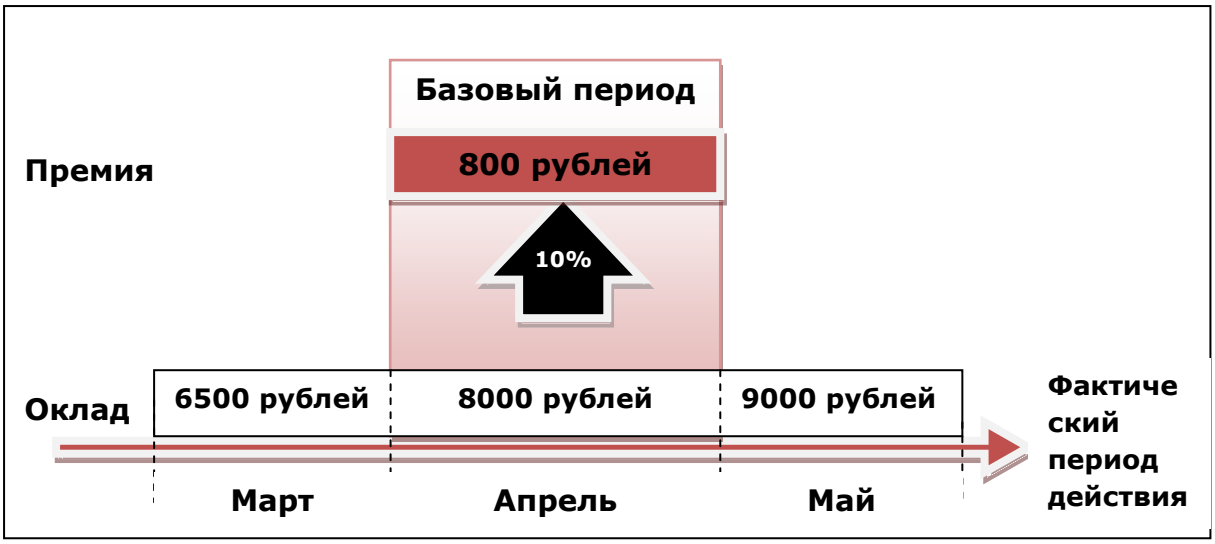
Алгоритм каждого расчета опирается на две категории параметров: период, за который нужно получить конечные данные, и набор некоторых исходных данных, используемых при расчете.

В реальной жизни различные виды расчета оказывают некоторое влияние на другие виды расчета.

- Это влияние может быть на исходные данные, используемые при расчете.

В качестве примера можно привести начисление премии в виде процента от оплаты по окладу. При изменении оклада размер премии тоже должен быть пересчитан. Т.е. сумма начисленного оклада является базой для расчета премии.

Причем оклад рассчитывается для некоторого периода, при расчете премии нам интересно знать не значение оклада вообще, а сумму за определенный период, влияющий на расчет премии. Такой период мы будем называть *базовым*, а подобную зависимость между видами расчета – *зависимостью по базовому периоду*.

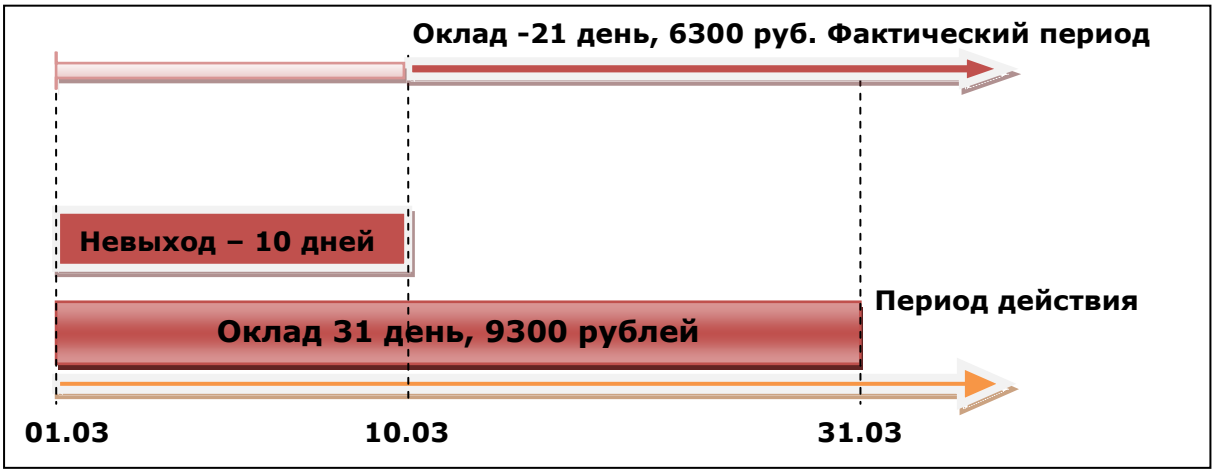


- Это влияние может быть не на исходные данные, а на сам период, за который производится расчет.

В качестве примера можно привести расчет оплаты по окладу и невыход на работу. Предположим, мы начислили сотруднику оплату по окладу за март. В этом случае период действия расчета будет с 01.03.2010 по 31.03.2010. После этого мы получили информацию от руководителя отдела, что сотрудник отсутствовал с 1 по 10 марта по неизвестной причине. В этом случае нам нужно будет произвести расчет **Невыход**, в котором можно рассчитать какие-то удержания с сотрудника. Но кроме этого необходимо будет пересчитать и оклад сотрудника исходя из фактического периода действия – с 11.03 по 31.03.

Такое влияние будем называть *вытеснение по периоду действия*.

В результате если за полный месяц работы сотруднику должно быть начислено 9300 рублей, то теперь за фактический период работы, начисление составит 6300 рублей.



Таким образом, с каждым видом расчета будет связано три периода:

- Период действия является «запрашиваемым». Т.е. указывая период действия, мы говорим: «Мы хотели бы, чтобы результат действовал в этом периоде».
- Фактический период – это то, что получилось из периода действия после анализа всех периодов действия расчетов, которые вытесняют наш по периоду действия.
- Базовый период – в котором мы анализируем результаты других расчетов, влияющих на наш по базовому периоду (календарный месяц).

План видов расчета

Объект *План видов расчета* предназначен для описания структуры хранения информации о возможных видах расчетов. На основе *Плана видов расчета* платформа создает в базе данных таблицу, в которой будет храниться информация о том, какие существуют виды расчета и какие взаимосвязи между ними.

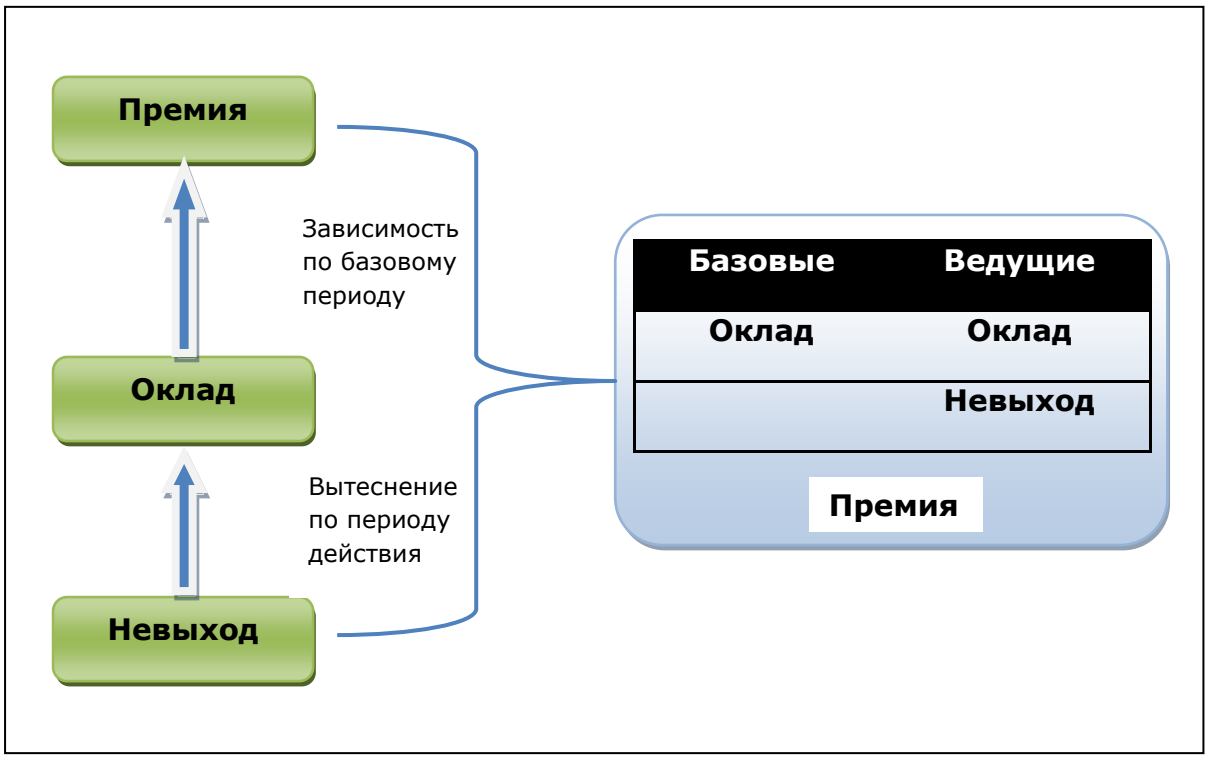
Отличительной особенностью плана видов расчета является то, что пользователь в процессе работы может добавлять новые виды расчета.

План видов расчета имеет свойства:

- **Использует период действия** – определяет, будут ли в этом плане находиться виды расчета, которые могут быть вытеснены по периоду действия.
- **Зависимость от базы** – определяет, будут ли в этом плане находиться зависимые по базовому периоду виды расчета.

Еще одной важной особенностью плана видов расчета является возможность создания предопределенных видов расчета и описания их взаимного влияния. При этом разработчик имеет возможность указать три категории видов расчета, влияющих на предопределенный вид расчета:

- *Базовые* – их результаты должны быть использованы при перерасчете этого вида расчета.
- *Вытесняющие* – вытесняют этот вид расчета по периоду действия.
- *Ведущие* – изменение их результатов должно приводить к необходимости перерасчета этого вида расчета.



Добавление плана видов расчета

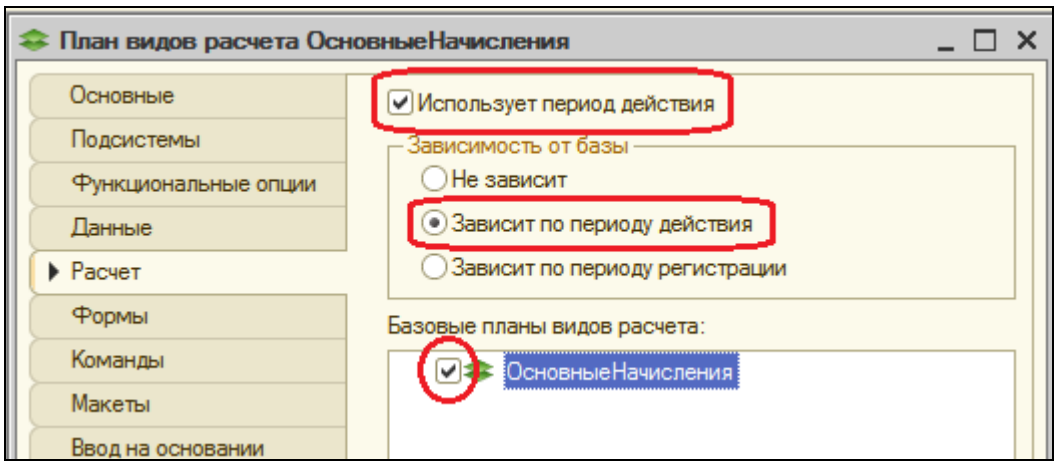
В режиме Конфигуратор

Создайте новый объект конфигурации План видов расчета с именем **ОсновныеНачисления**. Представление списка задайте как **Виды расчетов**.

На закладке **Подсистемы** укажите, что план видов расчета будет отображаться в подсистеме **РасчетЗарплаты**.

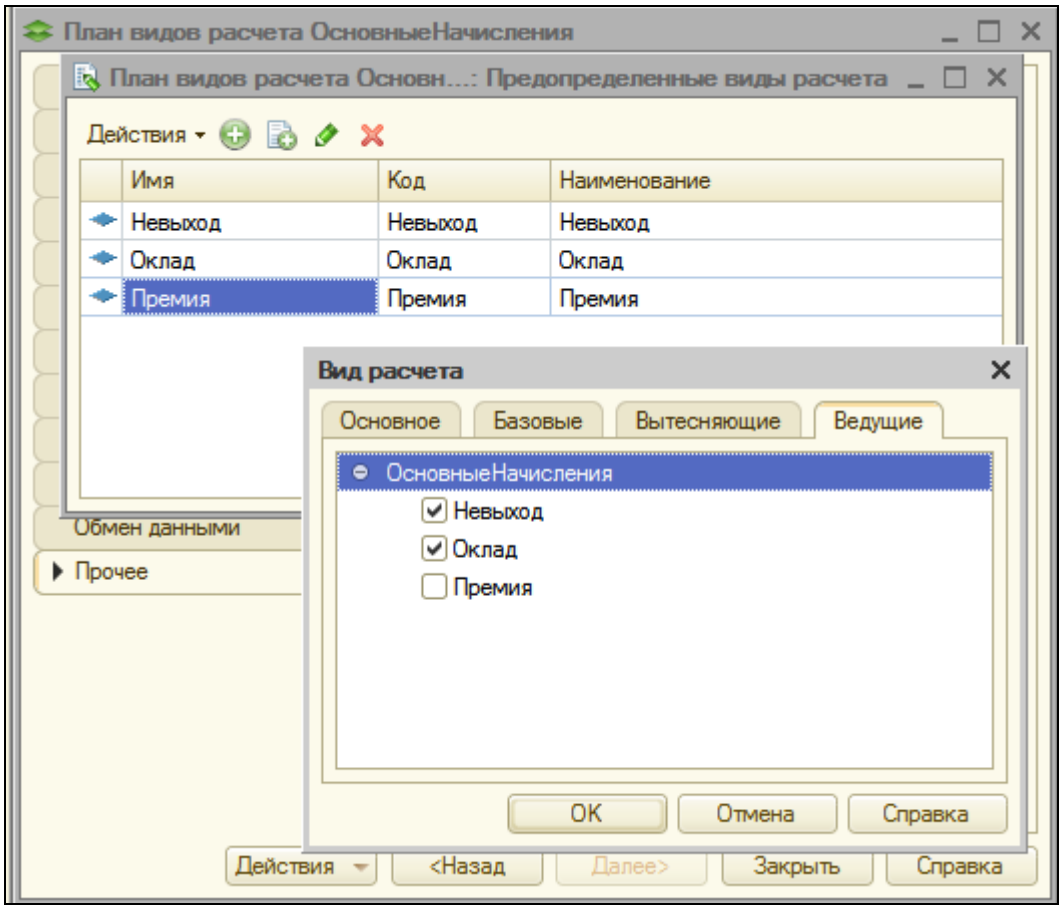
На закладке **Расчет** укажите, что он будет использовать период действия и зависеть от базы по периоду действия.

В качестве базового плана видов расчета укажем его самого, поскольку все наши виды расчетов будут храниться в единственном плане видов расчета.



На закладке Прочее зададим предопределенные виды расчета:

- **Невыход** – с именем, кодом и наименованием **Невыход**;
- **Оклад** – с именем, кодом и наименованием **Оклад** и вытесняющим его видом расчета **Невыход**;
- **Премия** – с именем, кодом и наименованием **Премия**, с базовым видом расчета **Оклад** и ведущими видами расчета **Невыход** и **Оклад**.



Что такое Регистр расчета

Объект конфигурации *Регистр расчета* предназначен для описания структуры накопления данных, являющихся результатами расчетов. На основе регистра расчета платформа создает в базе данных таблицы, в которых будут накапливаться данные, формируемые различными объектами базы данных.

Отличительные особенности регистра расчета:

- **Он не предназначен для интерактивного редактирования пользователем.** Разработчик может при необходимости предоставить пользователю возможность редактирования, но изначально регистр расчета для этого не предназначен.
- **Периодичность.** Определяет промежуток времени, к которому будет относиться каждая запись регистра.
- **Возможность использования механизмов вытеснения по периоду действия**. При этом для каждой записи регистр формирует фактический период действия, который является совокупностью нескольких периодов внутри периода действия.
- **Возможность использования механизмов вытеснения по зависимости по базовому периоду**. Этот механизм позволяет основывать расчет зависимых (вторичных) записей регистра на данных, полученных в результате расчета первичных записей.
    - **Зависимость по периоду действия.** При анализе базовых записей будут выбираться те, в которых найдено пересечение фактического периода действия и указанного базового периода.
    - **Зависимость по периоду регистрации.** При анализе базовых записей будут выбираться те, которые попадают в указанный базовый период значением своего поля Период регистрации.
- **Связь с планом видов расчета.** На основе этой связи работают механизмы вытеснения по периоду действия и зависимости по базовому периоду, поскольку в плане видов расчета описано взаимное влияние видов расчета друг на друга.

У регистра расчета могут существовать подчиненные объекты **Перерасчет**. Они предназначены для регистрации фактов появления в регистре записей, влияющих на результат расчета уже существующих записей регистра. **Перерасчет** может иметь несколько измерений, каждое из которых устанавливает связь между измерениями данного регистра расчета и влияющих регистров расчета. В частном случае это может быть один и тот же регистр.

Таблицы перерасчета заполняются автоматически на основании записей регистров расчета, затронутых ведущими видами расчета и на основании записей регистра расчета, для которых изменился фактический период действия.

Регистр расчета может быть связан с графиком времени. Такой график времени должен представлять собой регистр сведений (непериодический, с обязательным измерением типа Дата и ресурсом типа Число), в котором содержится временная схема исходных данных, участвующих в расчетах. Измерениями этого графика могут быть, например, график работы (ссылка на справочник) и дата, а ресурсом – количество рабочих часов в этой дате. В этом случае можно будет связать запись регистра расчета с каким-либо конкретным графиком работы (указав в качестве реквизита записи ссылку на справочник ВидыГрафиковРаботы) и в дальнейшем средствами встроенного языка получать информацию о количестве рабочих часов в периоде действия, фактическом периоде действия или периоде регистрации этой записи.

Добавление регистра расчета

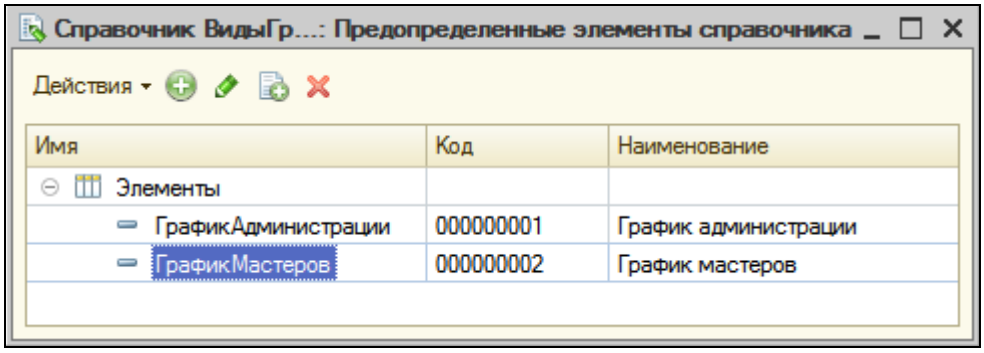
Прежде чем мы начнем создавать объект конфигурации Регистр расчета **Начисления**, нам потребуется создать два дополнительных объекта конфигурации:

- Регистр сведений **ГрафикиРаботы**,
- Справочник **ВидыГрафиковРаботы**.

Справочник понадобится нам для хранения информации о том, какие графики работы существуют в нашей фирме, а регистр сведений – для указания того, какие дни в месяце являются рабочими, поскольку сумма оплаты по окладу будет рассчитываться исходя из отработанных дней в месяце сотрудником.

В режиме Конфигуратор

Создайте новый объект Справочник с именем **ВидыГрафиковРаботы**. На закладке **Подсистемы** укажите, что справочник будет отображаться в подсистеме **РасчетЗарплаты**. На закладке Прочее создайте для справочника два предопределенных графика работы – **ГрафикАдминистрации** и **ГрафикМастеров**.



После этого создадим объект конфигурации Регистр сведений с именем **ГрафикиРаботы**. Этот регистр будет иметь два измерения:

- **ГрафикРаботы**, тип **СправочникСсылка.ВидыГрафиковРаботы**;
- **Дата**, тип **Дата**.

Затем создадим единственный ресурс регистра – **Значение**, тип **Число**, **длина 1**.

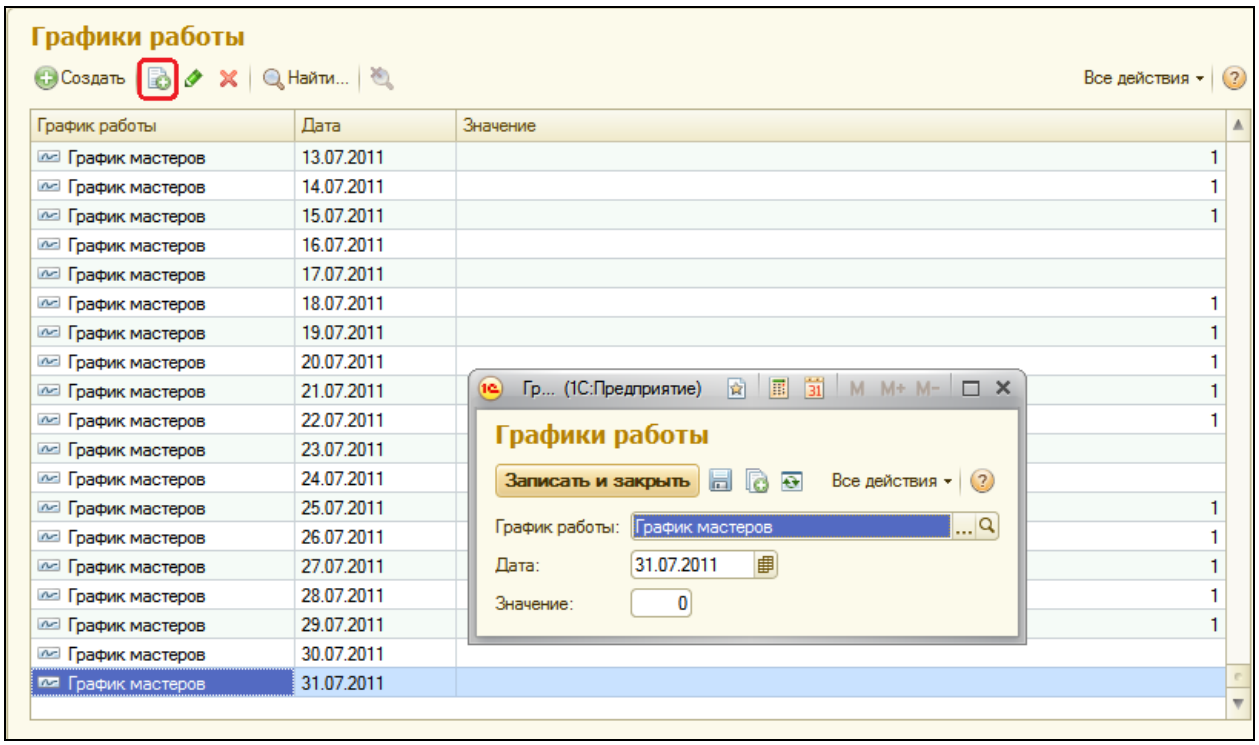
На закладке **Подсистемы** укажем **РасчетЗарплаты**.

Теперь заполним регистр сведений **ГрафикиРаботы** данными о рабочих днях июля графика мастеров.

(Если произошла ошибка Frame.dll, настройте родительское окно в режим Соединяемое, Свободное.)

В режиме 1С:Предприятие

Запустите режим отладки и в разделе **Расчет зарплаты** выполните команду **Графики работы**. Поочередно создайте 31 запись в регистре для июля. Для облегчения работы воспользуйтесь кнопкой **Создать новый элемент копированием текущего**. В качестве измерения **ГрафикРаботы** нашего регистра выберем предопределенный элемент **ГрафикМастеров** справочника **ВидыГрафиковРаботы**. В качестве ресурса **Значение** у рабочих дней проставим 1, выходных – 0.



Теперь все готово для создания регистра расчета.

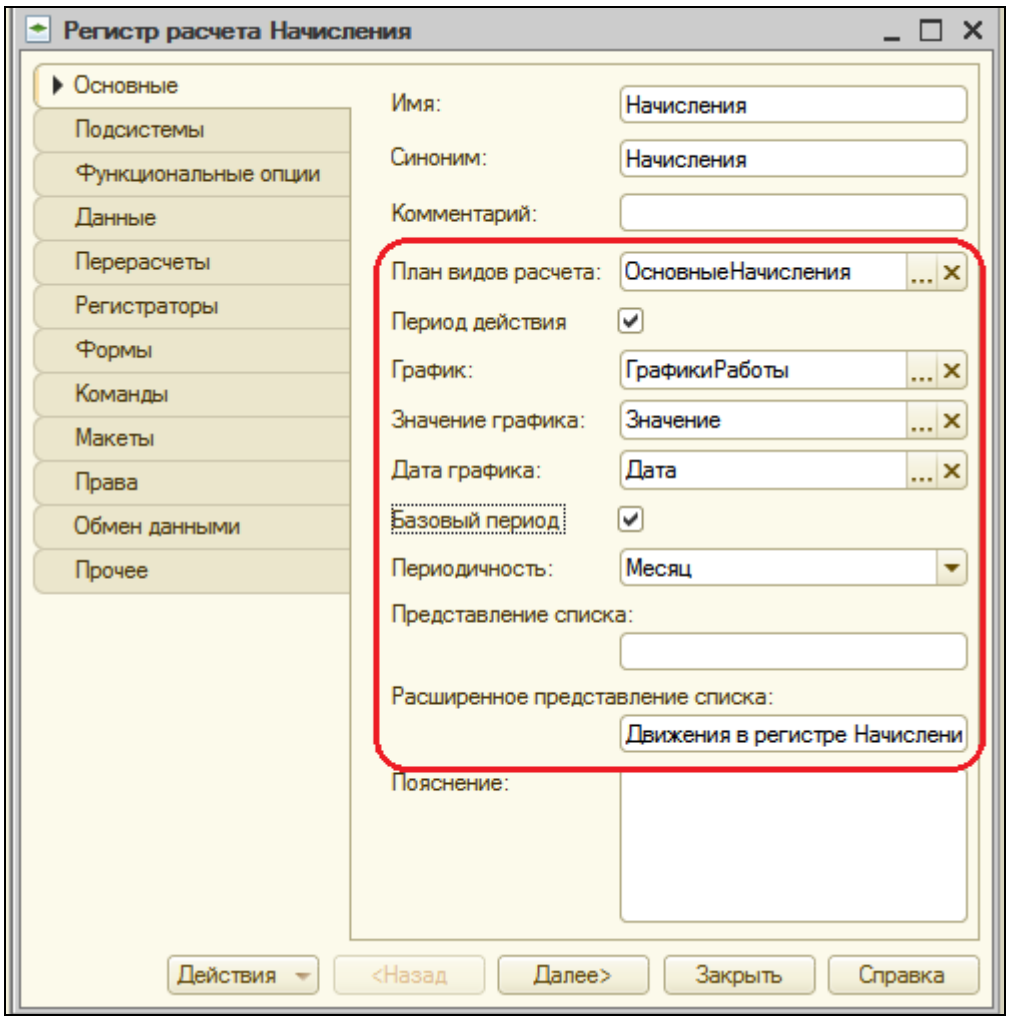
В режиме Конфигуратор

Добавим новый объект конфигурации *Регистр расчета* с именем **Начисления**. Зададим **Расширенное представление списка** как **Движения в регистре Начисления**.

В качестве плана видов расчета, используемого регистром, выберем **ОсновныеНачисления**.

Установим, что регистр будет использовать период действия, график будет задаваться в регистре сведений **ГрафикиРаботы**, значение графика будет находиться в ресурсе **Значение**, а дата графика – в измерении **Дата**. Укажем, что регистр будет использовать базовый период и периодичностью регистра будет **Месяц**.

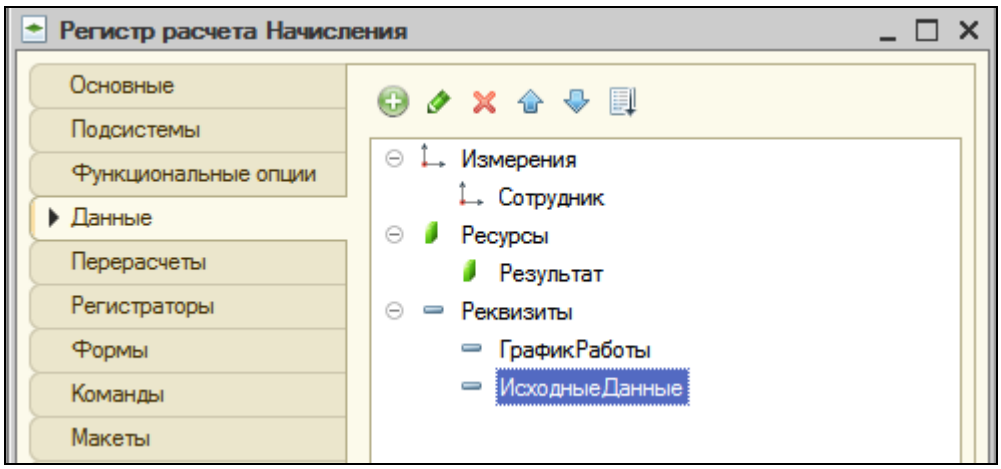
На закладке **Подсистемы** укажем **РасчетЗарплаты**.



На закладке **Данные** создадим:

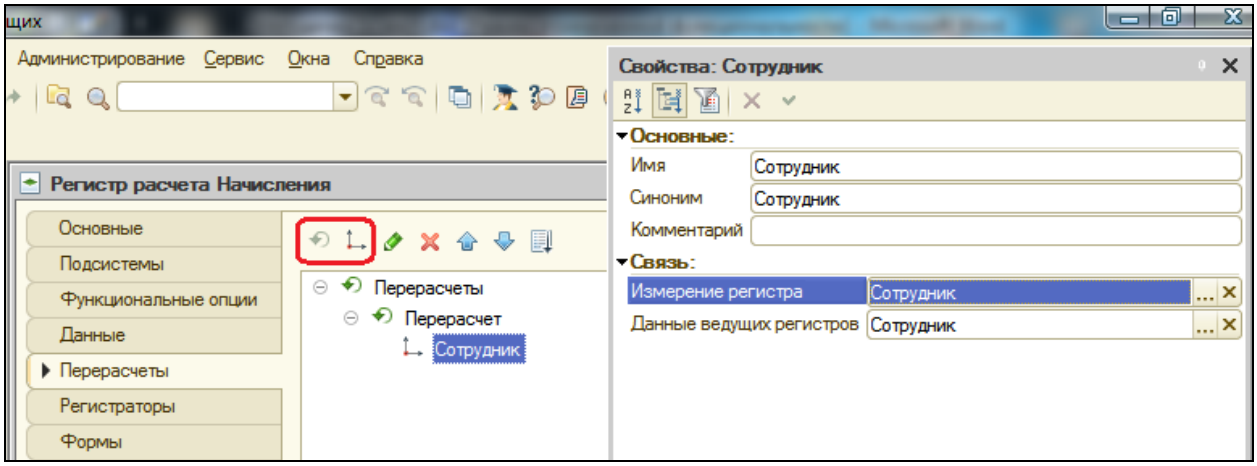
- Измерение **Сотрудник**, тип **СправочникСсылка.Сотрудники**, **Базово**е;
- Ресурс **Результат**, тип **Число**, длина **15**, точность **2**;
- Реквизит **ГрафикРаботы**, тип **СправочникСсылка.ВидыГрафиковРаботы**, в разделе свойств **Данные** зададим свойство **Связь с графиком** по измерению **ГрафикРаботы**;
- Реквизит **ИсходныеДанные**, тип **Число**, длина **15**, точность **2**.

Реквизит **ГрафикРаботы** будем использовать, чтобы связать запись регистра с используемым графиком работы, а реквизит **ИсходныеДанные** – чтобы хранить в нем данные для расчета или перерасчета.



Перейдем на закладку **Перерасчеты**. Создадим объект конфигурации **Перерасчет** с таким же именем. У него будет измерение Сотрудник, для которого в разделе Связь укажем:

- Измерение регистра – **Сотрудник**;
- Данные ведущих регистров – выберем то же самое измерение **Сотрудник** регистра расчета **Начисления**.



В заключение отредактируем командный интерфейс, чтобы в подсистеме **РасчетЗарплаты** была доступна команда для просмотра записей регистра расчета.

Для этого в дереве объектов конфигурации выберите подсистему **РасчетЗарплаты**, вызовите контекстное меню и выберите **Открыть командный интерфейс**. В группе **Панель навигации.Обычное** включите видимость команды Начисления.

На этом создание объекта Регистр расчета **Начисления** завершено.

Контрольные вопросы

- ✓ Как использовать план видов характеристик для организации.
- ✓ Что такое вид расчета, база.
- ✓ Какая разница между базовым периодом, периодом действия и фактическим периодом.
- ✓ Что такое зависимость по базовому периоду.
- ✓ Что такое вытеснение по периоду действия.
- ✓ Для чего предназначен объект конфигурации План видов расчета.
- ✓ Каковы основные свойства плана видов расчета.
- ✓ Какая разница между базовыми, вытесняющими и ведущими видами расчетов.
- ✓ Как создать план видов расчета.
- ✓ Что такое объект конфигурации Регистр расчета.
- ✓ Каковы отличительные особенности регистра расчета.
- ✓ Что такое перерасчет.
- ✓ Как создать регистр расчета.

Практическая работа № 17

Использование регистра расчета

В этой работе мы создадим документ, с помощью которого будут выполняться различные виды начислений, посмотрим, как и когда платформа формирует записи перерасчета, увидим, как работают механизмы вытеснения по периоду действия и зависимости по базовому периоду.

Кроме этого, мы создадим отчет, показывающий начисления сотрудникам нашей фирмы, и сделаем так, чтобы данные расчетов можно было поддерживать в актуальном состоянии. Всё это нам нужно для системы расчета зарплаты.

В заключение мы познакомимся с новым элементом формы – *Диаграмма Ганта* и с его помощью наглядно покажем работу некоторых механизмов расчета.

Добавление документа о начислениях

Откройте конфигуратор и добавьте новый объект Документ с именем **НачисленияСотрудникам** и представлением **Начисление сотрудникам**. На закладке **Нумерация** установим:

- **Тип номера** – **Число**;
- **Длина номера** – **5**.

На закладке **Подсистемы** укажем, что документ будет отображаться в подсистеме **РасчетЗарплаты**.

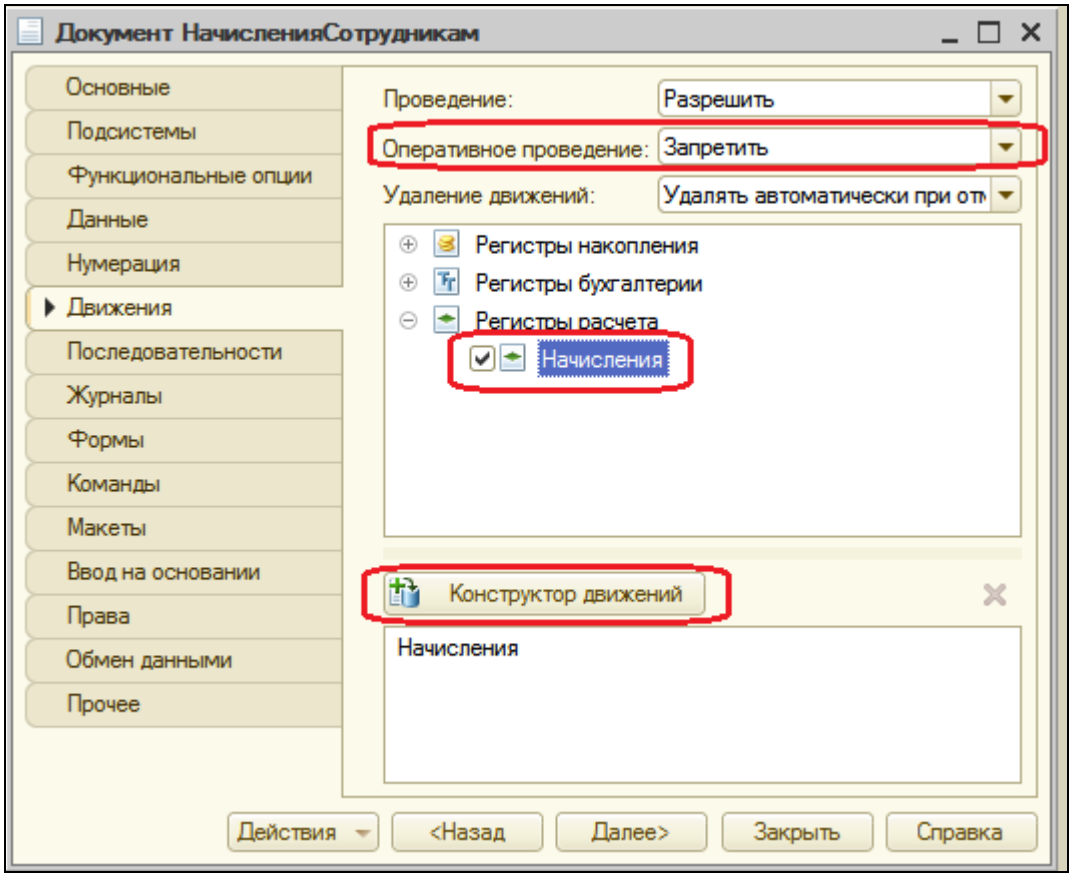
На закладке **Данные** укажем, что этот документ будет иметь табличную часть **Начисления**, содержащую следующие реквизиты:

- Сотрудник, тип СправочникСсылка.Сотрудник;
- ГрафикРаботы, тип СправочникСсылка.ВидыГрафиковРаботы;
- ДатаНачала, тип Дата;
- ДатаОкончания, тип Дата;
- ВидРасчета, тип ПланВидовРасчетаСсылка.ОсновныеНачисления;
- Начислено, Число, длина 15, точность 2.

Реквизиты **ДатаНачала** и **ДатаОкончания** понадобятся нам для задания периода, в котором должна действовать запись расчета.

На закладке **Движения** запретим оперативное проведение документа.

Отметим, что документ будет создавать движения по регистру расчета **Начисления**, и запустим конструктор движений.



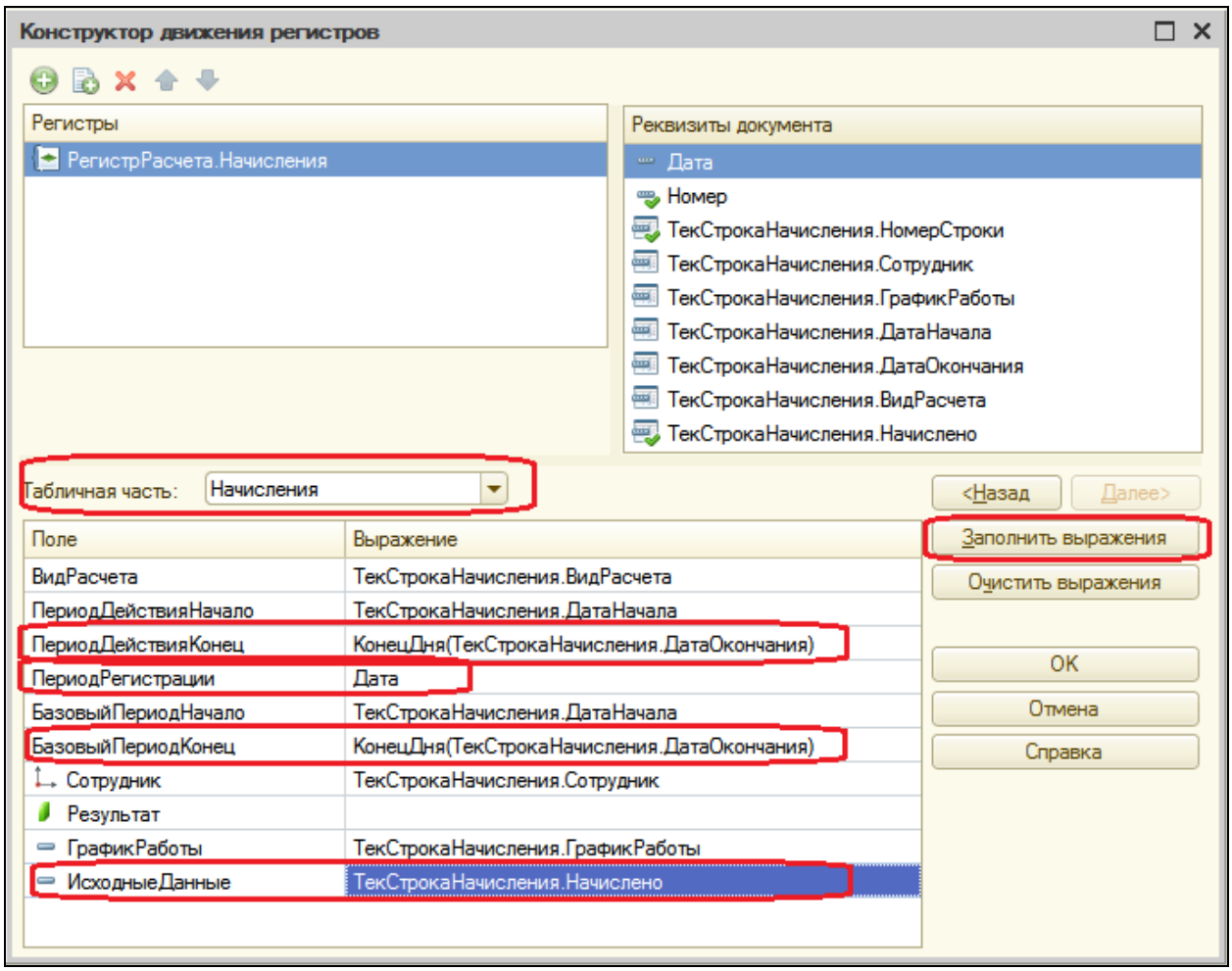
В окне конструктора выберем табличную часть **Начисления** и нажмите **Заполнить выражения**.

Для реквизитов **ПериодДействияКонец** и **БазовыйПериодКонец** укажем выражение **КонецДня(ТекСтрокаНачисления.ДатаОкончания).**

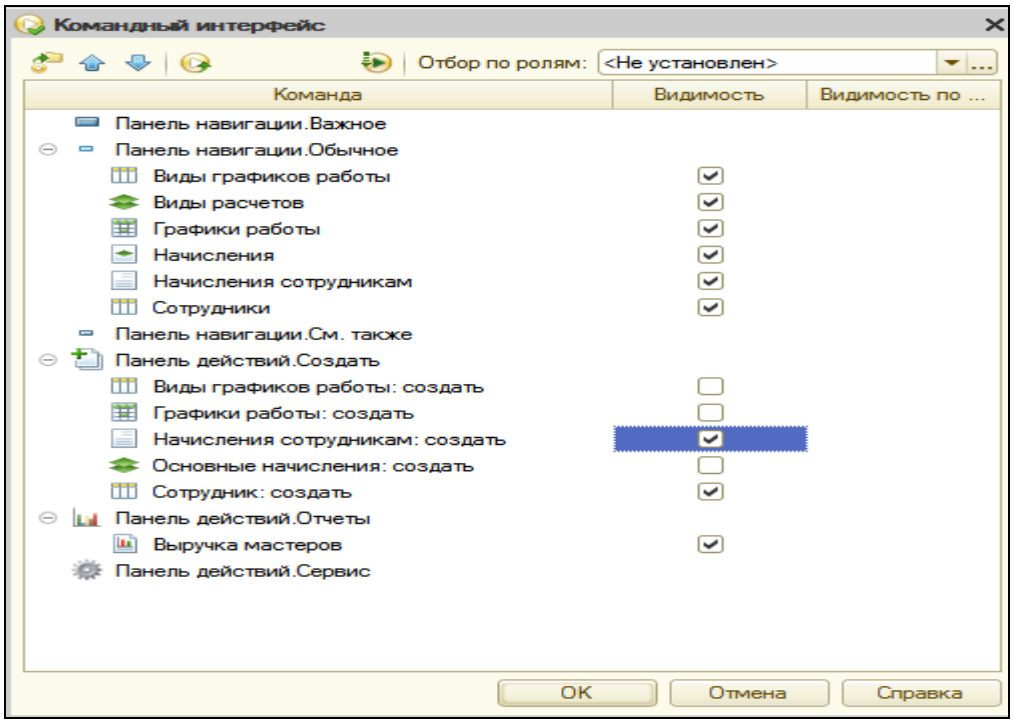
Для поля **ПериодРегистрации** укажем выражение **Дата**.

Реквизиту **ИсходныеДанные** поставим в соответствие реквизит табличной части **Начислено - ТекСтрокаНачисления.Начислено**, а для ресурса **Результат** оставим пустое выражение, т.к. мы будем его потом рассчитывать.

Нажмите **ОК** и посмотрим текст обработчика, созданный конструктором.



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

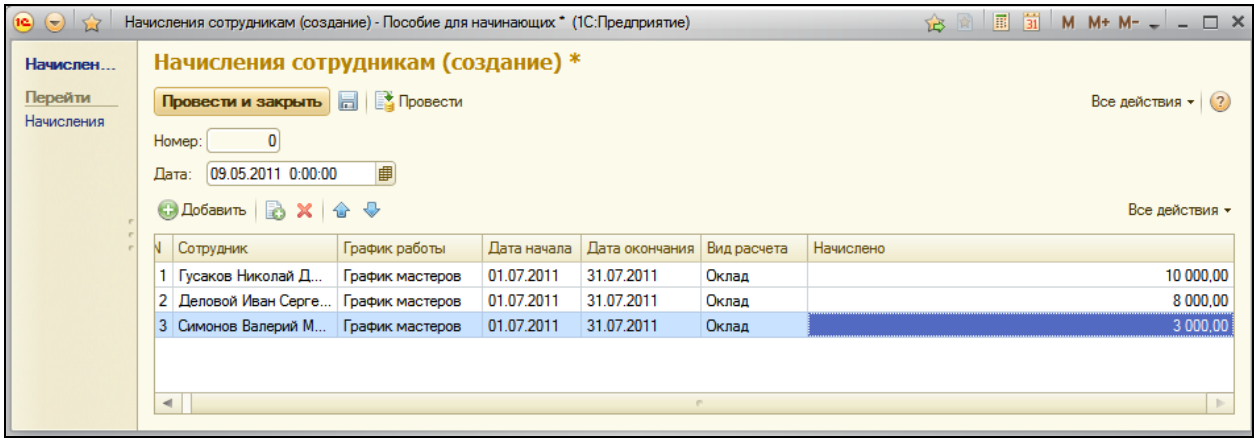


В режиме 1С:Предприятие

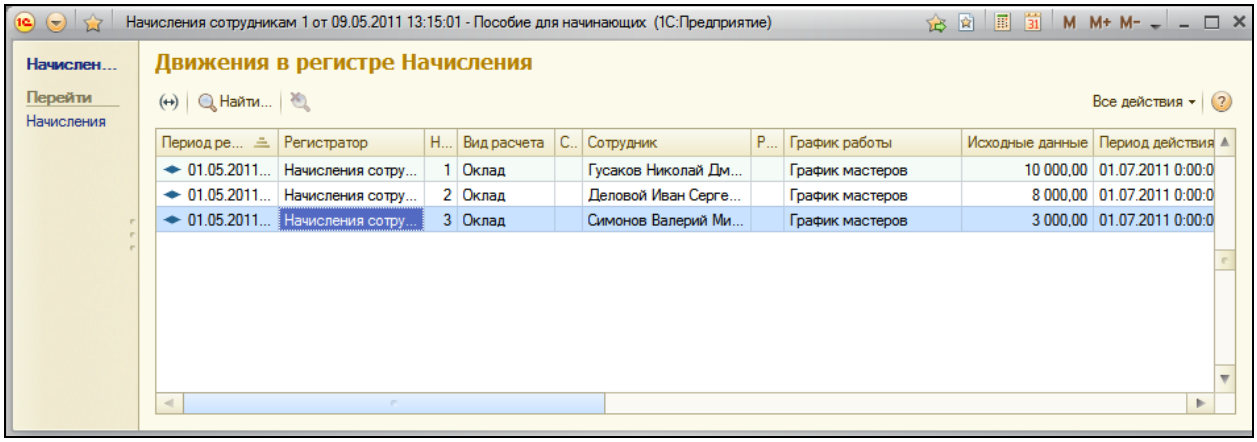
Запустим режим отладки и посмотрим, как работает наш документ.

В панели действий раздела **Расчет зарплаты** выполним команду **Начисление сотрудникам** и начислим оклад за июль всем сотрудникам фирмы.

Период начисления должен совпадать с периодом, введенным в регистр сведений Графики работы.



Проведем документ и посмотрим, какие движения он сформировал в регистре **Начисления**.



Обратите внимание, что платформа привела период регистрации каждой записи к началу периода регистра расчета.

Кроме этого, в каждой записи мы сохранили в реквизите **ИсходныеДанные** размер оклада сотрудника, введенный в документе, чтобы в дальнейшем рассчитать сумму оплаты по окладу.

Для дальнейшего изучения работы регистра расчета нам понадобится служебный отчет, с помощью которого мы сможем посмотреть содержимое записей перерасчета.

Иллюстрация механизмов вытеснения и зависимости от базы

Отчет по перерасчетам

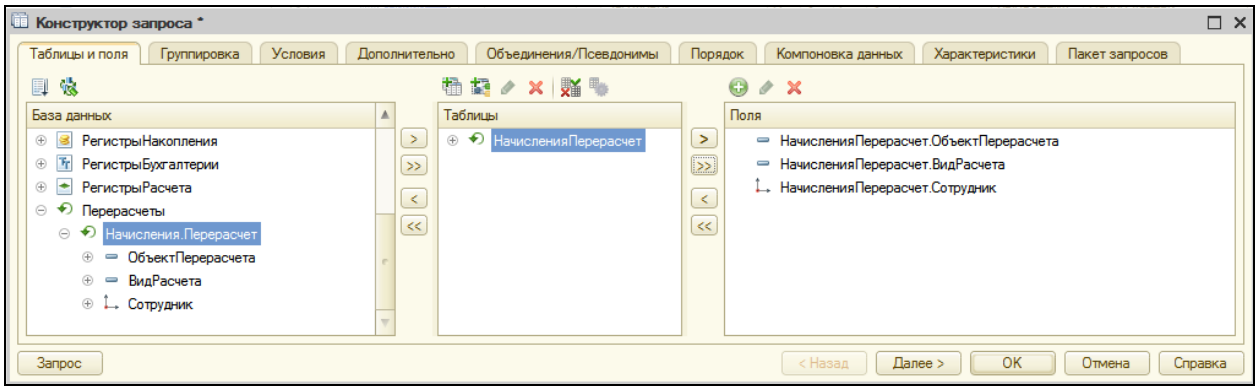
В режиме Конфигуратор

Создадим новый объект **Отчет** с именем **Перерасчет**. Создадим новую схему компоновки данных, добавим источник данных – запрос и откроем конструктор запроса.

В списке **База данных** раскроем ветвь **Перерасчеты** и из виртуальной таблицы перерасчета **Начисления.Перерасчет** выберем все поля:

- **ОбъектПерерасчета**,
- **ВидРасчета**,
- **Сотрудник**.

На этом создание запроса закончено, нажмите ОК.



Перейдем на закладку **Настройки** и добавим группировку детальных записей. На закладке **Выбранные поля** выберем для вывода в отчет поля **ОбъектПерерасчета**, **ВидРасчета** и **Сотрудник**.

На этом создание схемы компоновки закончено, закройте ее.

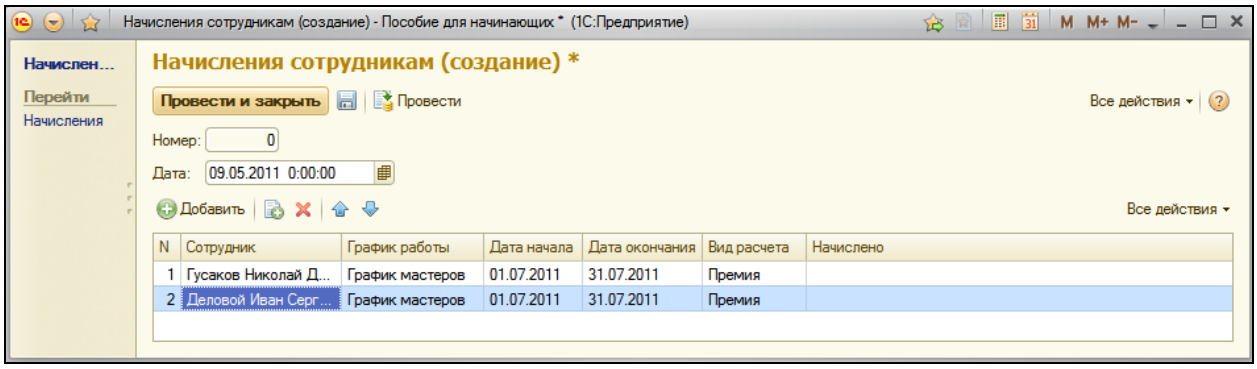
В окне редактирования отчета **Перерасчет** на закладке **Подсистемы** укажите **РасчетЗарплаты**.

Зависимость по базовому периоду

В режиме 1С:Предприятие

Если сейчас мы выполним отчет в режиме 1С:Предприятие, то мы увидим, что ни один перерасчет еще не выполнялся.

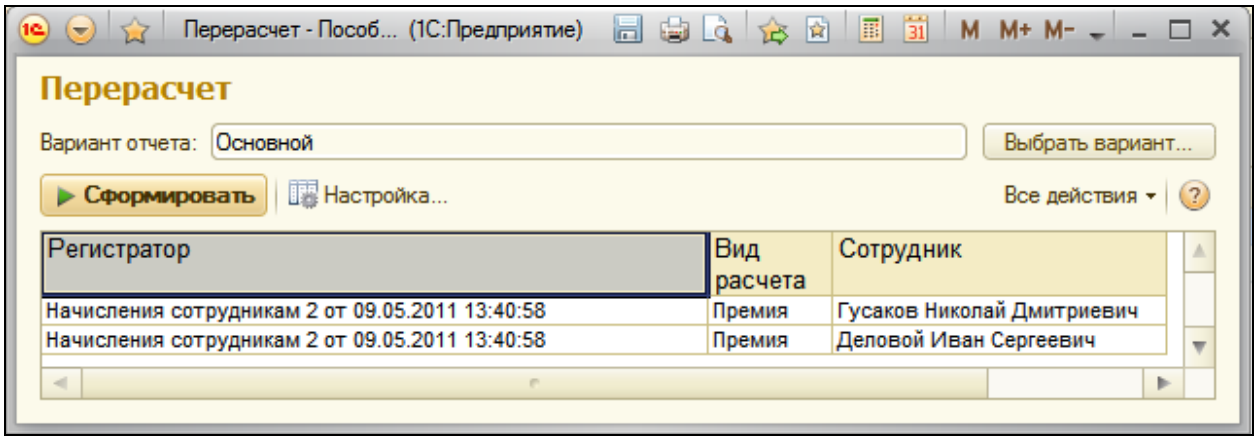
Поэтому создадим новый документ **Начисление сотрудникам №2**, в котором начислим премию за июль Гусакову и Деловому.



Этим документом мы зафиксируем факт, что сотрудникам Гусакову и Деловому нужно начислить премию по итогам работы за июль. Поскольку размер премии нам неизвестен (он будет рассчитываться), поле Начислено мы оставили пустым. Нажмите **Провести и закрыть**.

Теперь откроем документ **Начисление сотрудникам №1** (**Начисления сотрудникам** в навигационной панели) и изменим оклад Гусакова с 10000 на 7000. Нажмите **Провести и закрыть**.

Сформируйте отчет **Перерасчет**.



Как видите, отчет содержит какие-то данные. В самом деле, вид расчета **Премия** зависит у нас по базовому периоду от вида расчета **Оклад**. Как только мы изменили существовавшие в регистре записи по виду расчета **Оклад**, платформа сразу же сформировала набор записей перерасчета, которые должны быть рассчитаны заново, т.к. изменилась их база.

В перерасчет попали записи как про Делового, так и про Гусакова, потому что платформа не отслеживает конкретные изменения, а

отслеживает факт изменения набора записей регистра расчета в результате проведения документа. Поэтому она включает информацию обо всех записях регистра, значение ресурсов которых может измениться в результате перепроведения документа, создавшего базовые записи регистра.

Перепроведем документ **Начисления сотрудникам №2** и сформируем отчет **Перерасчет**.

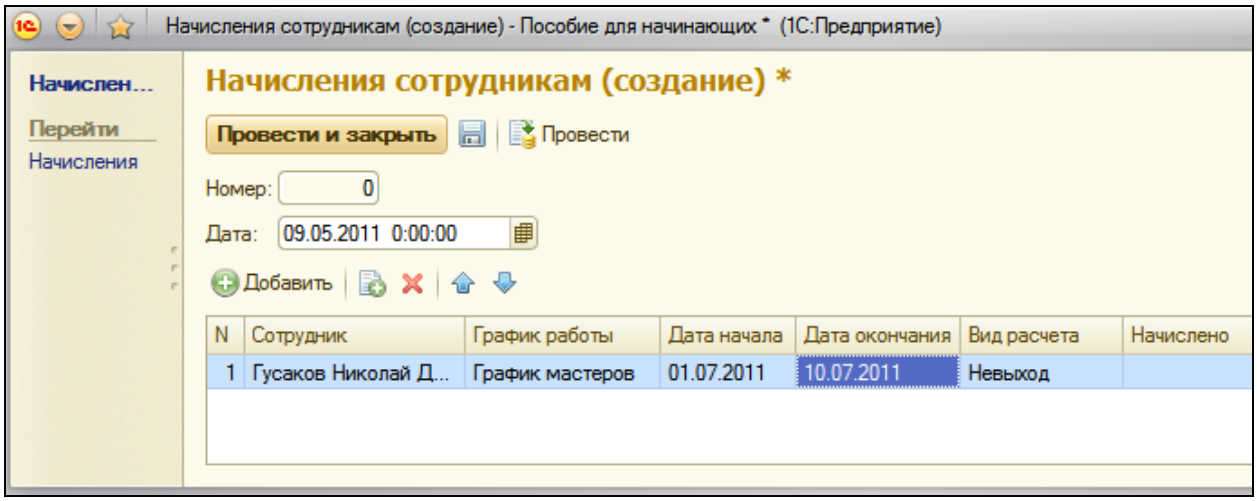
Он снова не содержит никаких данных – система отметила, что мы пересчитали зависимые записи, и очистила таблицу перерасчета.

На этом примере мы познакомились с работой механизма поддержки зависимости по базовому периоду у регистра расчета.

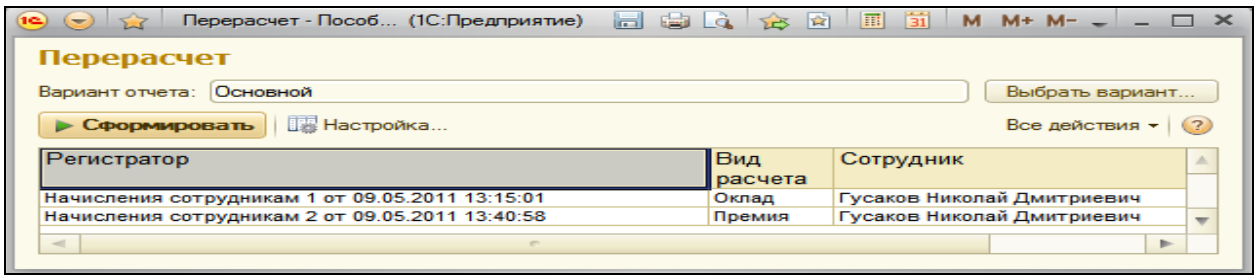
Вытеснение по периоду действия

В режиме 1С:Предприятие

Теперь посмотрим, как работает механизм вытеснения по периоду действия. Для этого нам понадобится создать документ **Начисления сотрудникам №3.**



Этим документом мы зафиксируем факт невыхода на работу Гусакова с 1 по 10 июля. В этом случае потребуется пересчитать его оплату по окладу и как следствие начисленную премию. Нажмите **Провести и закрыть** и сформируйте отчет **Перерасчет**.



Как видите, в перерасчет попала запись о начислении оклада Гусакову. Это явилось результатом работы механизма вытеснения по периоду действия, ведь вид расчета **Невыход** вытесняет у нас вид расчета **Оклад**.

Обратите внимание, что в перерасчет попала и запись о начислении премии Гусакову. При создании предопределенных видов расчета мы указали, что результат вида расчета **Премия** будет зависеть от изменения результата вида расчета **Невыход**. Платформа отследила эту зависимость.

Перепроведем документы **Начисления сотрудникам №1** и **№2** и убедимся, что таблица перерасчета очистилась, сформировав отчет.

Процедура расчета записей регистра расчета

В режиме Конфигуратор

До сих пор мы просто заносили в регистр расчета **Начисления** записи о том, что необходимо выполнить какой-либо вид расчета, но каким именно образом получать эти результаты, мы не говорили.

Теперь настало время описать алгоритмы формирования различных видов расчета. Поскольку эти алгоритмы нам нужно будет использовать не только в документе **Начисления сотрудникам**, удобнее всего будет разместить их в отдельном общем модуле.

Откроем в конфигураторе текст обработчика проведения документа **НачислениеСотрудникам** и добавим в него после завершения создания движений в регистре **Начисления** вызов процедуры **РассчитатьНачисления()** из общего модуля **ПроведениеРасчетов**.

```
Процедура ОбработкаПроведения(Отказ, Режим)
…
КонецЦикла;

        //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ

        //Записываем движения регистров.
        Движения.Начисления.Записать();
```

```
        //Получим список всех сотрудников, содержащихся в документе.
                Запрос = Новый Запрос(
                "ВЫБРАТЬ РАЗЛИЧНЫЕ
        |       НачисленияСотрудникамНачисления.Сотрудник
        |ИЗ
        |   Документ.НачисленияСотрудникам.Начисления
        |       КАК НачисленияСотрудникамНачисления
        |ГДЕ
        |       НачисленияСотрудникамНачисления.Ссылка =
&ТекущийДокумент");

        Запрос.УстановитьПараметр("ТекущийДокумент", Ссылка);

        //Сформируем список сотрудников.
        ТаблЗнач = Запрос.Выполнить().Выгрузить();
        МассивСотрудников = ТаблЗнач.ВыгрузитьКолонку("Сотрудник");

        //Вызов процедуры РассчитатьНачисления из общего модуля.
        ПроведениеРасчетов.РассчитатьНачисления(Движения.Начисления,
                ПланыВидовРасчета.ОсновныеНачисления.Оклад,
МассивСотрудников);
        Движения.Начисления.Записать(, Истина);
        ПроведениеРасчетов.РассчитатьНачисления(Движения.Начисления,
                ПланыВидовРасчета.ОсновныеНачисления.Премия,
МассивСотрудников);
        Движения.Начисления.Записать(, Истина);
КонецПроцедуры
```

Обратите внимание: при проведении документа мы сначала записываем движения, сформированные документом, в регистр (`Движения.Начисления.Записать()`), а затем передаем этот набор записей регистра в процедуру расчета **РассчитатьНачисления()**, которую мы создадим в общем модуле **ПроведениеРасчетов**.

Эту процедуру мы вызываем сначала для расчета первичных записей (**Оклад**), а затем для расчета вторичных (**Премия**).

После того, как ресурсы будут рассчитаны, мы перезаписываем набор записей регистра без формирования записей перерасчета (второй параметр в методе **Записать()** – **Истина**).

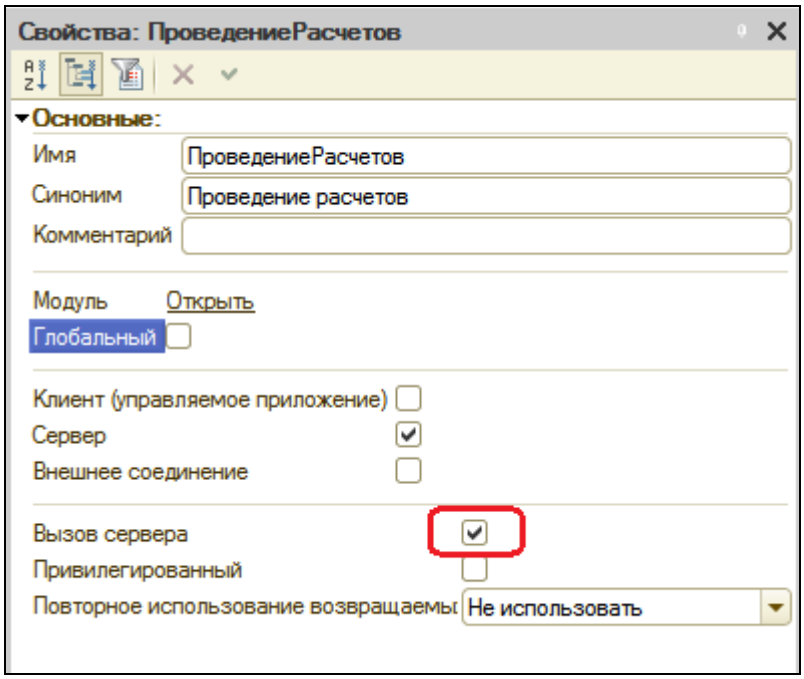
Перед вызовом процедуры из общего модуля мы с помощью запроса формируем массив сотрудников, перечисленных в документе, чтобы передать его в вызываемую процедуру.

Для параметра запроса **ТекущийДокумент** устанавливаем значение стандартного реквизита документа – **Ссылка**. Используя метод запроса **Запрос.Выполнить().Выгрузить()**, выгружаем результат запроса в таблицу значений (переменную **ТаблЗнач**). Затем формируем массив

МассивСотрудников, содержащий колонку **Сотрудник** из этой таблицы значений.

Теперь создадим в ветке **Общие** новый общий модуль **ПроведениеРасчетов**.

Установим флажок **Вызов сервера** для видимости его экспортных процедур и функций.



Добавим в него следующий текст процедуры **РассчитатьНачисления**:

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
СписокСотрудников) Экспорт

	Регистратор=НаборЗаписейРегистра.Отбор.Регистратор.Значение;

	// Рассчитать первичные записи
	Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад
Тогда
Запрос = Новый Запрос;
Запрос.Текст =
	"ВЫБРАТЬ
	| НачисленияДанныеГрафика.ЗначениеПериодДействия КАК Норма,
	| НачисленияДанныеГрафика.ЗначениеФактическийПериодДействия КАК Факт,
	| НачисленияДанныеГрафика.НомерСтроки КАК НомерСтроки
	|ИЗ
	| РегистрРасчета.Начисления.ДанныеГрафика(Регистратор = &Регистратор И
	| ВидРасчета = &ВидРасчета И Сотрудник В (&СписокСотрудников))
	| КАК НачисленияДанныеГрафика";

Запрос.УстановитьПараметр("Регистратор", Регистратор);
Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);
Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);
```

```
ВыборкаРезультата = Запрос.Выполнить().Выбрать();
        // Рассчитать вторичные записи
        ИначеЕсли ТребуемыйВидРасчета =
ПланыВидовРасчета.ОсновныеНачисления.Премия Тогда
        КонецЕсли;

КонецПроцедуры
```

В этом запросе мы выбираем из виртуальной таблицы данных графика регистра расчета значение графика для периода действия и для фактического периода действия. При задании параметров виртуальной таблицы мы ограничиваем выборку регистратором, нужным нам видом расчета и списком сотрудников, по которым нужно получить значения графика.

Теперь добавим обход переданного в процедуру набора записей и расчет записей, для которых получены значения графика.

```
        Запрос.УстановитьПараметр("Регистратор", Регистратор);
        Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);
        Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);

        ВыборкаРезультата = Запрос.Выполнить().Выбрать();

        Для Каждого ЗаписьРегистра Из НаборЗаписейРегистра Цикл
                СтруктураНомер = Новый Структура("НомерСтроки");
                СтруктураНомер.НомерСтроки = ЗаписьРегистра.НомерСтроки;
                ВыборкаРезультата.Сбросить();
                Если ВыборкаРезультата.НайтиСледующий(СтруктураНомер) Тогда
                        Если ВыборкаРезультата.Норма = 0 Тогда
                                Сообщение = Новый СообщениеПользователю;
                                Сообщение.Текст = "Вид расчета: Оклад – Нет рабочих
дней в заданном периоде";
                                Сообщение.Сообщить();
                                ЗаписьРегистра.Результат = 0;
                        Иначе

                                // Рассчитать оклад по фактическому периоду и
исходным данным
                                ЗаписьРегистра.Результат =
(ЗаписьРегистра.ИсходныеДанные  /ВыборкаРезультата.Норма) *
ВыборкаРезультата.Факт;
                                Сообщение = Новый СообщениеПользователю;
                                Сообщение.Текст = "Выполнен расчет" +
ЗаписьРегистра.Регистратор + " – "  + ЗаписьРегистра.ВидРасчета + " – "
                                 + ЗаписьРегистра.Сотрудник;
                                Сообщение.Сообщить();
                        КонецЕсли;
                КонецЕсли;
        КонецЦикла;
```

```
            // Рассчитать вторичные записи
            ИначеЕсли ТребуемыйВидРасчета =
ПланыВидовРасчета.ОсновныеНачисления.Премия Тогда
            КонецЕсли;

        КонецПроцедуры
```

Для каждой записи из набора записей регистра расчета мы получаем номер строки, идентифицирующий начисление для конкретного сотрудника, и по этому номеру ищем соответствующую запись в выборке из результатов запроса.

Если в результате запроса есть запись с таким номером строки, мы рассчитываем результат записи регистра расчета. Т.е. мы получаем начисление по окладу для каждого сотрудника как результат от деления начисленной суммы (поле регистра **ИсходныеДанные**) на количество рабочих дней в месяце (**Норма**) и умножения на фактически отработанные рабочие дни (**Факт**).

Добавим текст запроса во вторую ветку условия с той лишь разницей, что теперь мы будем получать значения базы, используя виртуальную таблицу регистра расчета **РегистрРасчета.Начисления.БазаНачисления**.

```
// Рассчитать вторичные записи
ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
Тогда

        Запрос = Новый Запрос;
Запрос.Текст =
        "ВЫБРАТЬ
        | НачисленияБазаНачисления.РезультатБаза КАК База,
        | НачисленияБазаНачисления.НомерСтроки КАК НомерСтроки
        |ИЗ
        | РегистрРасчета.Начисления.БазаНачисления(&Измерения Основного,
        | &ИзмеренияБазового, , Регистратор = &Регистратор И ВидРасчета =
&ВидРасчета И
        | Сотрудник В (&СписокСотрудников)) КАК НачисленияБазаНачисления";

Измер = Новый Массив(1);
Измер[0] = "Сотрудник";

Запрос.УстановитьПараметр("ИзмеренияОсновного", Измер);
Запрос.УстановитьПараметр("ИзмеренияБазового", Измер);
Запрос.УстановитьПараметр("Регистратор", Регистратор);
Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);
Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);
```

```
ВыборкаРезультата = Запрос.Выполнить().Выбрать();
	КонецЕсли;

КонецПроцедуры
```

В параметрах виртуальной таблицы запроса мы кроме регистратора, вида расчета и списка сотрудников задаем еще измерения основного и базового регистров. В нашем случае это будет один и тот же регистр **Начисления**, а нужное нам измерение – **Сотрудник**.

В заключение осталось добавить во второе условие обход набора записей регистра расчета и вычисление результата вторичных записей.

```
// Рассчитать вторичные записи
ИначеЕсли ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Премия
Тогда
…
ВыборкаРезультата = Запрос.Выполнить().Выбрать();

Для Каждого ЗаписьРегистра Из НаборЗаписейРегистра Цикл
	СтруктураНомер = Новый Структура("НомерСтроки");
	СтруктураНомер.НомерСтроки = ЗаписьРегистра.НомерСтроки;
	ВыборкаРезультата.Сбросить();
	Если ВыборкаРезультата.НайтиСледующий(СтруктураНомер)  Тогда
		ЗаписьРегистра.Результат = ВыборкаРезультата.База * (10 / 100);
		Сообщение = Новый СообщениеПользователю;
		Сообщение.Текст = "Выполнен расчет" +
ЗаписьРегистра.Регистратор + " – "  + ЗаписьРегистра.ВидРасчета + " – " +
ЗаписьРегистра.Сотрудник;
		Сообщение.Сообщить();
	КонецЕсли;
КонецЦикла;

	КонецЕсли;

КонецПроцедуры
```

Сумму начисленной премии мы рассчитываем как 10% от рассчитанной оплаты по окладу.

Общий вид на данный момент общего модуля:

```
Процедура РассчитатьНачисления(НаборЗаписейРегистра, ТребуемыйВидРасчета,
СписокСотрудников) Экспорт



	Регистратор=НаборЗаписейРегистра.Отбор.Регистратор.Значение;



	// Рассчитать первичные записи
```

```
        Если ТребуемыйВидРасчета = ПланыВидовРасчета.ОсновныеНачисления.Оклад
Тогда
                Запрос = Новый Запрос;
                Запрос.Текст =
                        "ВЫБРАТЬ
                        | НачисленияДанныеГрафика.ЗначениеПериодДействия КАК Норма,
                        |
НачисленияДанныеГрафика.ЗначениеФактическийПериодДействия КАК Факт,
                        | НачисленияДанныеГрафика.НомерСтроки КАК НомерСтроки
                        |ИЗ
                        | РегистрРасчета.Начисления.ДанныеГрафика(Регистратор =
&Регистратор И
                        | ВидРасчета = &ВидРасчета И Сотрудник В (&СписокСотрудников))

                        | КАК НачисленияДанныеГрафика";


                Запрос.УстановитьПараметр("Регистратор", Регистратор);
                Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);
                Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);

                ВыборкаРезультата = Запрос.Выполнить().Выбрать();

                Для Каждого ЗаписьРегистра Из НаборЗаписейРегистра Цикл
                        СтруктураНомер = Новый Структура("НомерСтроки");
                        СтруктураНомер.НомерСтроки = ЗаписьРегистра.НомерСтроки;
                        ВыборкаРезультата.Сбросить();
                        Если ВыборкаРезультата.НайтиСледующий(СтруктураНомер) Тогда
                                Если ВыборкаРезультата.Норма = 0 Тогда
                                        Сообщение = Новый СообщениеПользователю;
                                        Сообщение.Текст = "Вид расчета: Оклад – Нет
рабочих дней в заданном периоде";
                                        Сообщение.Сообщить();
                                        ЗаписьРегистра.Результат = 0;
                                Иначе
```

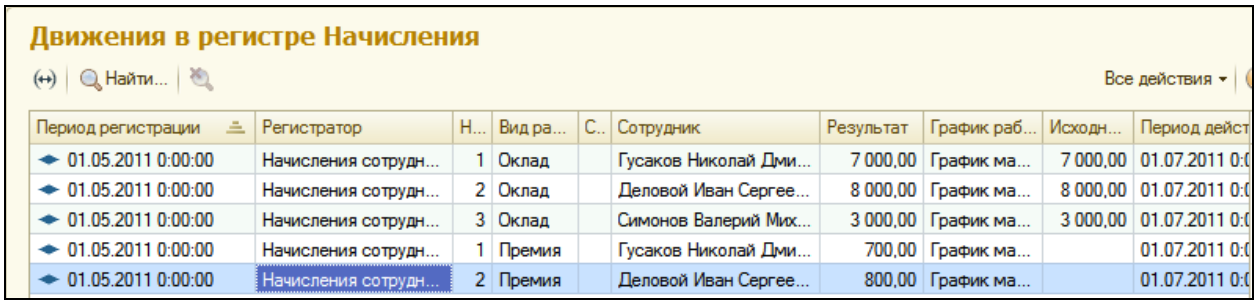
```
// Рассчитать оклад по фактическому периоду и исходным данным

                                    ЗаписьРегистра.Результат =
(ЗаписьРегистра.ИсходныеДанные  /ВыборкаРезультата.Норма) *
ВыборкаРезультата.Факт;

                                    Сообщение = Новый СообщениеПользователю;

                                    Сообщение.Текст = "Выполнен расчет" +
ЗаписьРегистра.Регистратор + " – "  + ЗаписьРегистра.ВидРасчета + " – "

                                     + ЗаписьРегистра.Сотрудник;

                                    Сообщение.Сообщить();

                                КонецЕсли;

                            КонецЕсли;

                        КонецЦикла;

        // Рассчитать вторичные записи

        ИначеЕсли ТребуемыйВидРасчета =
ПланыВидовРасчета.ОсновныеНачисления.Премия Тогда

        Запрос = Новый Запрос;

        Запрос.Текст =

                "ВЫБРАТЬ

                | НачисленияБазаНачисления.РезультатБаза КАК База,

                | НачисленияБазаНачисления.НомерСтроки КАК НомерСтроки

                |ИЗ

                | РегистрРасчета.Начисления.БазаНачисления(&ИзмерениеОсновного,

                | &ИзмерениеБазового, , Регистратор = &Регистратор И ВидРасчета =
&ВидРасчета И

                | Сотрудник В (&СписокСотрудников)) КАК НачисленияБазаНачисления";


        Измер = Новый Массив(1);

        Измер[0] = "Сотрудник";


        Запрос.УстановитьПараметр("ИзмерениеОсновного", Измер);

        Запрос.УстановитьПараметр("ИзмерениеБазового", Измер);

        Запрос.УстановитьПараметр("Регистратор", Регистратор);

        Запрос.УстановитьПараметр("ВидРасчета", ТребуемыйВидРасчета);

        Запрос.УстановитьПараметр("СписокСотрудников", СписокСотрудников);
```

```
        ВыборкаРезультата = Запрос.Выполнить().Выбрать();


        Для Каждого ЗаписьРегистра Из НаборЗаписейРегистра Цикл
        СтруктураНомер = Новый Структура("НомерСтроки");
        СтруктураНомер.НомерСтроки = ЗаписьРегистра.НомерСтроки;
        ВыборкаРезультата.Сбросить();
        Если ВыборкаРезультата.НайтиСледующий(СтруктураНомер)  Тогда
                ЗаписьРегистра.Результат = ВыборкаРезультата.База * (10 / 100);
                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст = "Выполнен расчет" + ЗаписьРегистра.Регистратор + "
– "  + ЗаписьРегистра.ВидРасчета + " – " + ЗаписьРегистра.Сотрудник;
                Сообщение.Сообщить();
        КонецЕсли;
КонецЦикла;
        КонецЕсли;
КонецПроцедуры
```

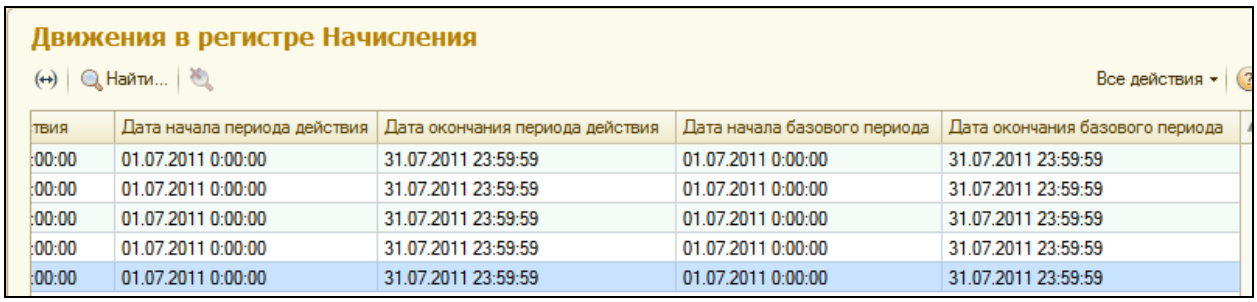
В режиме 1С:Предприятие

Запустите режим отладки и проверим правильность работы процедуры расчета.

Отменим проведение документа **Начисления сотрудникам №3** (**Все действия – Отмена проведения**) и перепроведем документы **Начисления сотрудникам №1** и **№2**. Регистр расчета **Начисления** должен выглядеть так:



Период регистрации	Регистратор	Н...	Вид ра...	С..	Сотрудник	Результат	График раб...	Исходн...	Период дейст
01.05.2011 0:00:00	Начисления сотрудн...	1	Оклад		Гусаков Николай Дми...	7 000,00	График ма...	7 000,00	01.07.2011 0:(
01.05.2011 0:00:00	Начисления сотрудн...	2	Оклад		Деловой Иван Сергее...	8 000,00	График ма...	8 000,00	01.07.2011 0:(
01.05.2011 0:00:00	Начисления сотрудн...	3	Оклад		Симонов Валерий Мих...	3 000,00	График ма...	3 000,00	01.07.2011 0:(
01.05.2011 0:00:00	Начисления сотрудн...	1	Премия		Гусаков Николай Дми...	700,00	График ма...		01.07.2011 0:(
01.05.2011 0:00:00	Начисления сотрудн...	2	Премия		Деловой Иван Сергее...	800,00	График ма...		01.07.2011 0:(

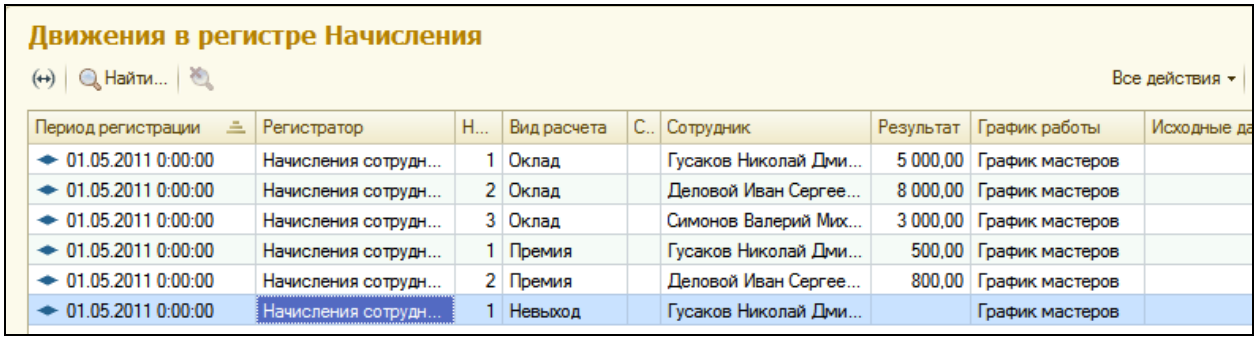


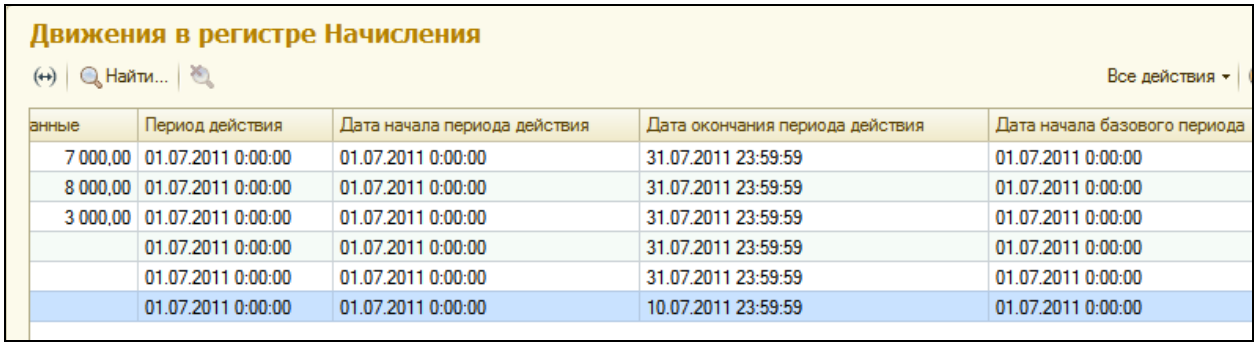
Мы видим, что всем сотрудникам произведены начисления по окладу (поле **Результат**) за полный месяц в соответствии с исходными данными (поле **Исходные данные**).

Сотрудникам Гусакову и Деловому начислена премия 10% от оклада.

Проведем документы **Начисление сотрудникам №3**, затем **№1** и **№2**. При этом отчет **Перерасчет** должен быть пуст.

Состояние регистра изменится следующим образом (только для июля 2011):





В результате невыхода на работу Гусакова сумма оплаты по окладу будет уменьшена и соответственно уменьшится премия.

Отчет о начислениях сотрудникам

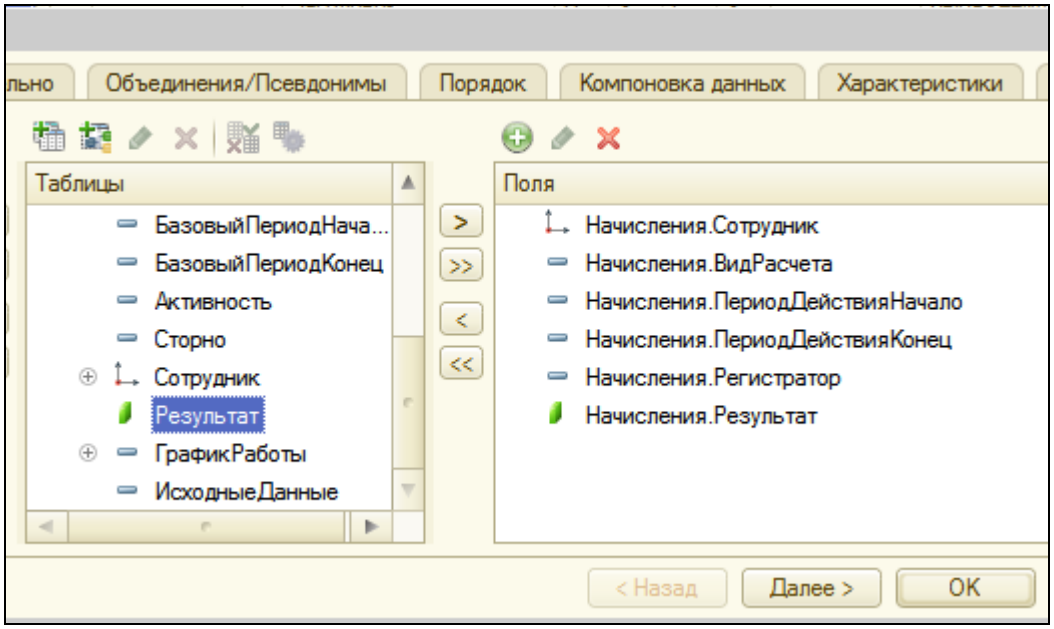
Теперь мы посмотрим, каким образом можно использовать данные, хранящиеся в регистре расчета, для получения в отчете итоговой информации о начислениях сотрудникам.

В режиме Конфигуратор

Создадим новый объект **Отчет** с именем **НачисленияСотрудникам**. Создадим основную схему компоновки данных, добавим новый **Набор данных – запрос**, откроем конструктор запроса.

Выберем таблицу регистра расчета **Начисления**. Из нее выберем поля:

- Сотрудник,
- ВидРасчета,
- ПериодДействияНачало,
- ПериодДействияКонец,
- Регистратор,
- Результат.



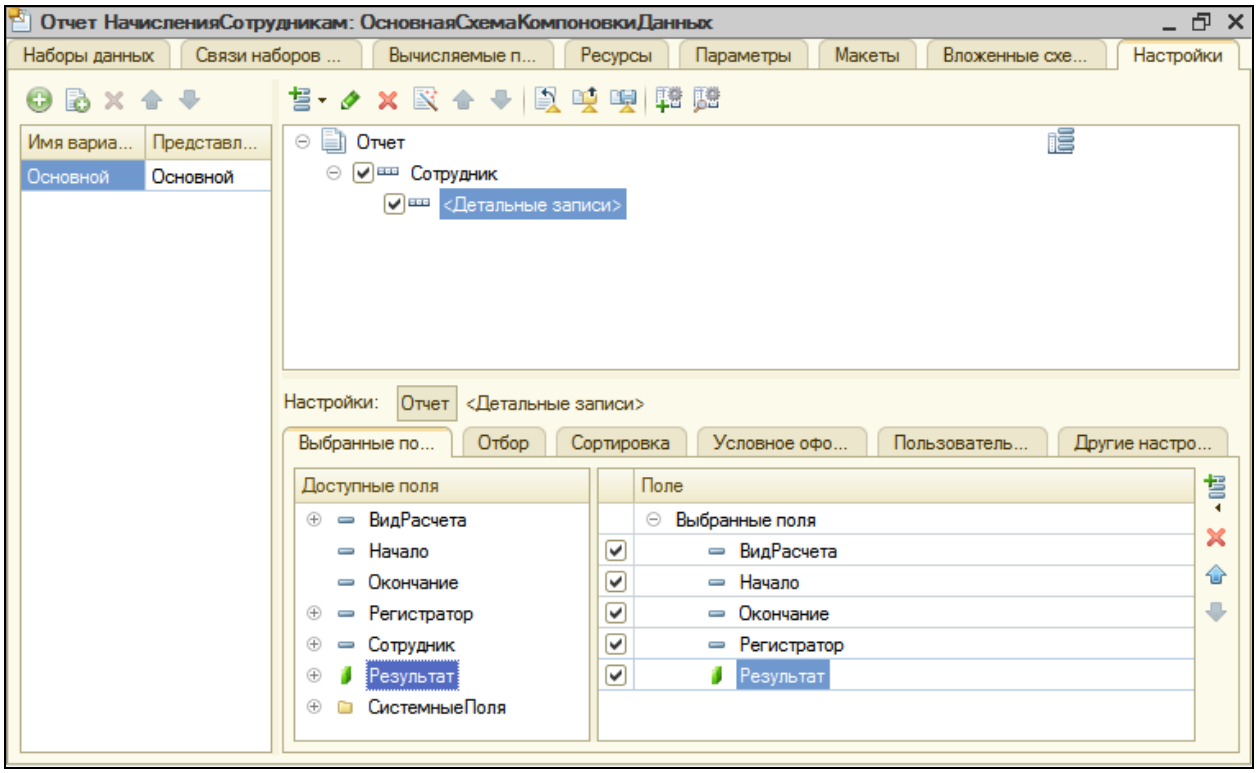
На закладке **Объединения/Псевдонимы** определим псевдонимы полей **ПериодДействияНачало** и **ПериодДействияКонец** как **Начало** и **Окончание** соответственно.

На этом создание запроса закончено, нажмите ОК.

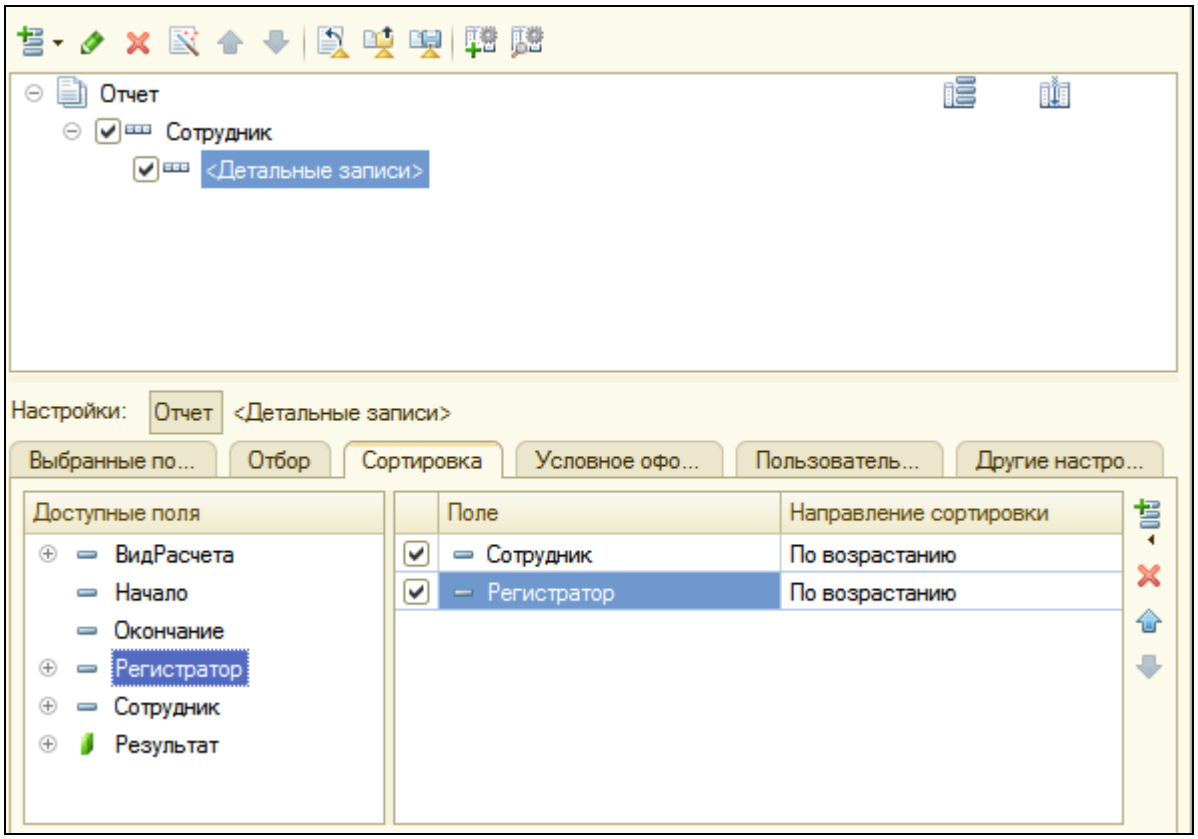
Перейдите на закладку **Ресурсы** и укажите, что должна быть рассчитана сумма по полю результат, дважды щелкнув на нем.

На закладке **Настройки** добавим группировку по полю **Сотрудник** и в ней – подчиненную группировку детальных записей.

В качестве полей, выводимых в отчет, выберем **ВидРасчета**, **Начало**, **Окончание**, **Регистратор** и **Результат**.



На закладке **Сортировка** укажем, что она должна выполняться по возрастанию значения поля **Сотрудник** и **Регистратор**.

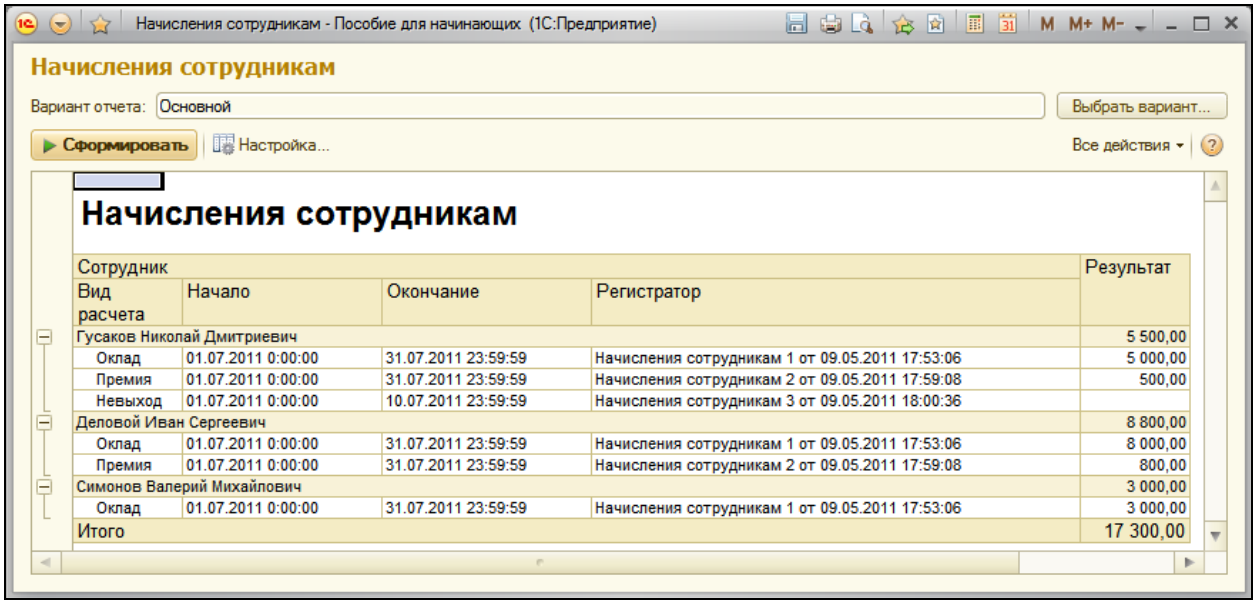


На закладке **Другие настройки** зададим заголовок отчета – **Начисления сотрудникам**. Закройте схему компоновки.

В окне редактирования отчета на закладке **Подсистемы** укажите **РасчетЗарплаты** и **Бухгалтерия**.

В режиме 1С:Предприятие

Запустите режим отладки. В командной панели раздела **Расчет зарплаты** выполните команду создания отчета **Начисления сотрудникам** и сформируйте его.



Начисления сотрудникам

Сотрудник / Вид расчета	Начало	Окончание	Регистратор	Результат
Гусаков Николай Дмитриевич				5 500,00
Оклад	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 1 от 09.05.2011 17:53:06	5 000,00
Премия	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 2 от 09.05.2011 17:59:08	500,00
Невыход	01.07.2011 0:00:00	10.07.2011 23:59:59	Начисления сотрудникам 3 от 09.05.2011 18:00:36	
Деловой Иван Сергеевич				8 800,00
Оклад	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 1 от 09.05.2011 17:53:06	8 000,00
Премия	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 2 от 09.05.2011 17:59:08	800,00
Симонов Валерий Михайлович				3 000,00
Оклад	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 1 от 09.05.2011 17:53:06	3 000,00
Итого				17 300,00

Перерасчет

В нашем алгоритме осталось одно узкое место – контроль актуальности данных в регистре расчета.

До сих пор мы использовали служебный отчет **Перерасчет** для определения актуальности данных (актуальны, если отчет пуст).

Поскольку единственным способом получения итоговой информации о начислениях сотрудникам в нашей конфигурации является отчет **НачисленияСотрудникам**, для вызова этой процедуры мы создадим основную форму этого отчета и добавим в командную панель формы кнопку **Перерассчитать**, по которой будет выполняться перерасчет данных регистра.

В режиме Конфигуратор

В окне редактирования отчета **НачисленияСотрудникам** перейдем на закладку **Формы**, нажмем кнопку открытия  и создадим основную форму отчета.

В правом верхнем окне редактора форм перейдем на закладку **Команды** и на закладке **Команды формы** создадим команду формы **Перерассчитать**.

Теперь нужно установить **Действие** для этой команды. Для этого нажмите кнопку открытия  в строке **Действия**.

В модуле формы будет создан шаблон процедуры **Перерассчитать()**, в которую мы поместим вызов процедуры **ПерерассчитатьНачисления()** из общего модуля **ПроведениеРасчетов**.

```
&НаКлиенте
Процедура Перерассчитать(Команда)
	ПроведениеРасчетов.ПерерассчитатьНачисления(ПредопределенноеЗначение
	("ПланВидовРасчета.ОсновныеНачисления.Оклад"));
ПроведениеРасчетов.ПерерассчитатьНачисления(ПредопределенноеЗначение
	("ПланВидовРасчета.ОсновныеНачисления.Премия"));
КонецПроцедуры
```

Саму процедуру перерасчета поместим в общем модуле **ПроведениеРасчетов** после процедуры **РассчитатьНачисления()**.

```
Процедура ПерерассчитатьНачисления(ТребуемыйВидРасчета) Экспорт

	// Здесь следует выбрать из набора записей перерасчета записи в следующей
последовательности:
	// записи документа1 для сотрудников из списка,
	// записи документа2 для сотрудников из списка, и т. д.
	Запрос = Новый Запрос(
	"ВЫБРАТЬ
	| НачисленияПерерасчет.ОбъектПерерасчета,
	| НачисленияПерерасчет.Сотрудник
	|ИЗ
	| РегистрРасчета.Начисления.Перерасчет КАК НачисленияПерерасчет
	|
	|ГДЕ
	| НачисленияПерерасчет.ВидРасчета = &ТребуемыйВидРасчета
	|
	|ИТОГИ ПО
	| НачисленияПерерасчет.ОбъектПерерасчета");

	Запрос.УстановитьПараметр("ТребуемыйВидРасчета", ТребуемыйВидРасчета);
	СписокСотрудников = Новый СписокЗначений;

	// Перебрать группировку по регистратору.
```

```
	ВыборкаПоРегистратору =
Запрос.Выполнить().Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
	Пока ВыборкаПоРегистратору.Следующий() Цикл
		Регистратор = ВыборкаПоРегистратору.ОбъектПерерасчета;

		// Перебрать группировку по сотрудникам для выбранного регистратора
		// и создать список сотрудников.
		ВыборкаПоСотрудникам = ВыборкаПоРегистратору.Выбрать();
		СписокСотрудников.Очистить();

		Пока ВыборкаПоСотрудникам.Следующий() Цикл
			СписокСотрудников.Добавить(ВыборкаПоСотрудникам.Сотрудник);
		КонецЦикла;

		// Получить набор записей регистра расчета для выбранного
регистратора.
		НаборЗаписей = РегистрыРасчета.Начисления.СоздатьНаборЗаписей();
		НаборЗаписей.Отбор.Регистратор.Значение = Регистратор;
		НаборЗаписей.Прочитать();

		РассчитатьНачисления(НаборЗаписей, ТребуемыйВидРасчета,
СписокСотрудников);
		НаборЗаписей.Записать( , Истина);

		// Очистить перерассчитанные записи в перерасчете.
		НаборЗаписейПерерасчета =
РегистрыРасчета.Начисления.Перерасчеты.Перерасчет.СоздатьНаборЗаписей();
		НаборЗаписейПерерасчета.Отбор.ОбъектПерерасчета.Значение =
Регистратор;
		НаборЗаписейПерерасчета.Записать();
	КонецЦикла;

КонецПроцедуры
```

В самом начале процедуры мы запросом выбираем данные о записях перерасчетов, содержащие переданный вид расчета и сгруппированные по объекту перерасчета.

Далее, при обходе результата запроса, мы формируем для каждого объекта перерасчета список сотрудников, читаем соответствующие записи регистра расчета и вызываем процедуру **РассчитатьНачисления**, которая использовалась нами при расчете записей документа **НачисленияСотрудникам**.

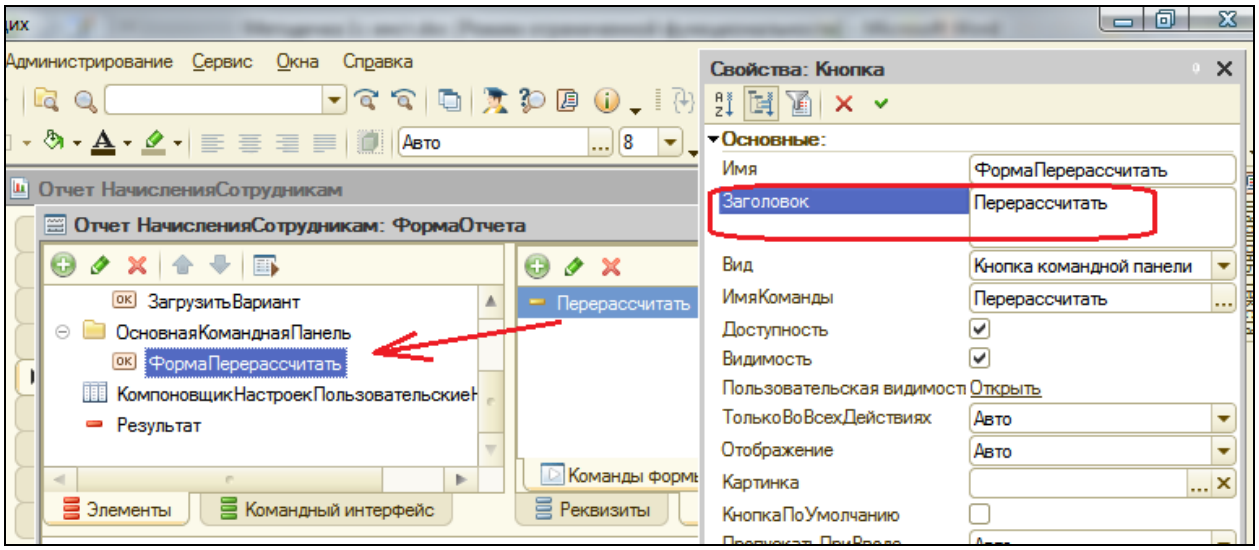
После выполнения расчета записей, мы записываем набор записей без формирования записей перерасчета и очищаем записи перерасчета по тому объекту, который только что обработали.

Вернемся в форму отчета **НачислениеСотрудникам**. Мы указали для команды **Перерассчитать** действие, т.е. процедуру ее выполнения.

Но чтобы воспользоваться этой командой, нужно создать в форме кнопку и связать ее с этой командой (в строке **Команда**).

Проще всего это сделать перетаскиванием команды из окна **Команды формы** в окно элементов формы.

Перетащите мышью команду **Перерассчитать** в группу элементов формы **ОсновнаяКомандныйПанель**. Задайте заголовок кнопки - **Перерассчитать**.

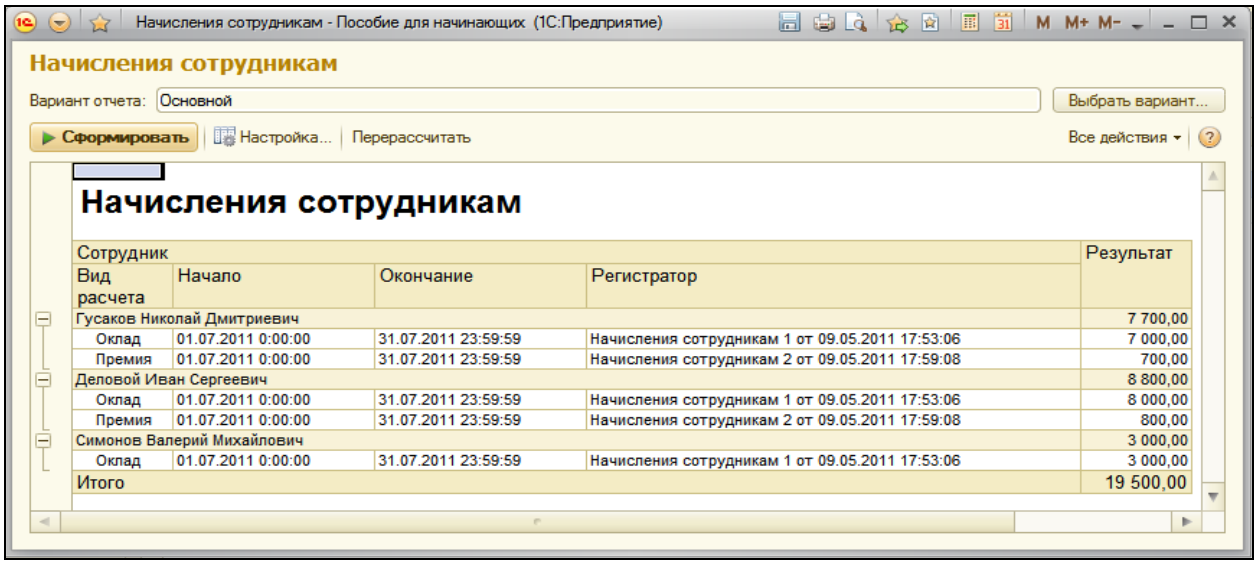


В режиме 1С:Предприятие

Запустим отладку и проверим как выполняется перерасчет.

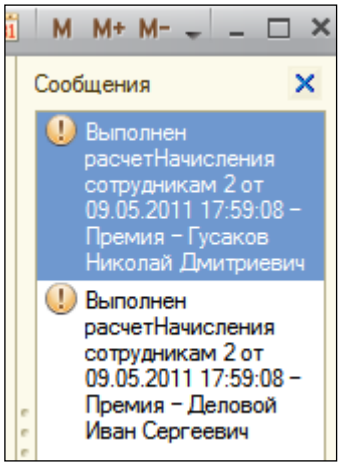
Отменим проведение всех документов **Начисления сотрудникам** и проведем документ **Начисления сотрудникам №1** и затем **№2**.

Сформируем отчет **Начисления сотрудникам**.

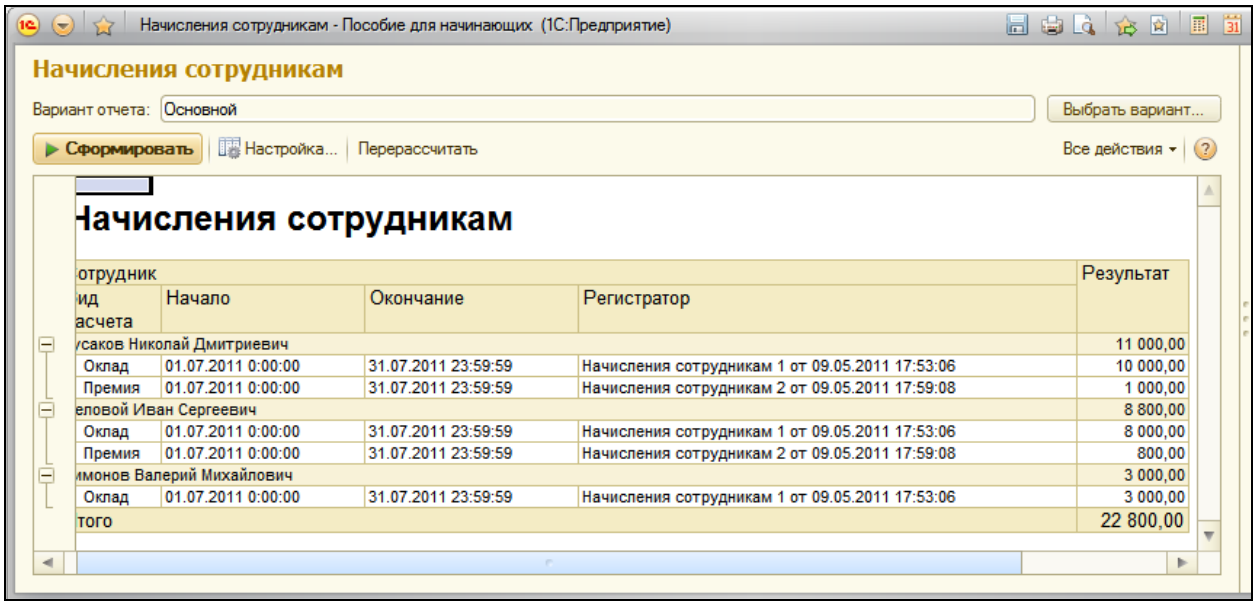


Теперь откроем документ **Начисления сотрудникам №1**, изменим оклад Гусакова на 10000 и проведем документ.

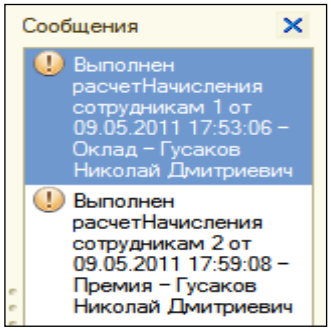
В отчете Начисления сотрудникам нажмите **Перерассчитать**. Будет выполнен перерасчет премии Гусакову и Деловому.



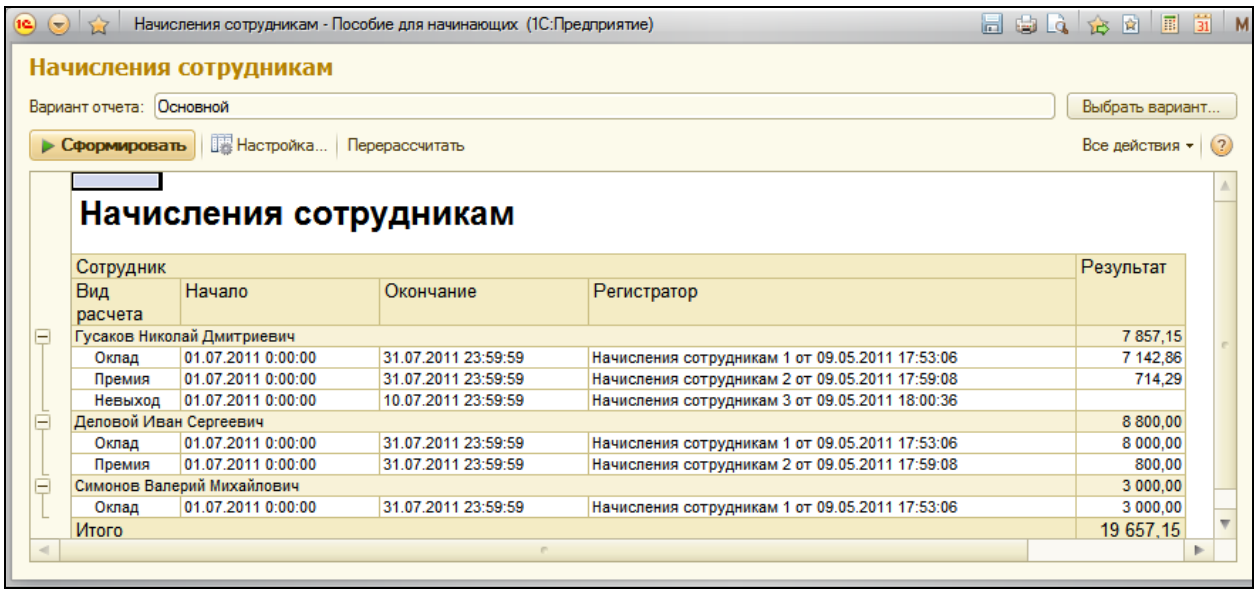
Чтобы увидеть в отчете актуальные данные, нажмите **Сформировать**. Результат работы будет содержать новые значения премии для Гусакова.



Наконец, проведем документ **Начисления сотрудникам №3** и нажмем **Перерассчитать** в отчете. Снова будет произведен перерасчет оклада и премии Гусакова.



Нажмите **Сформировать**. Данные отчета будут содержать актуальные значения начисления оклада и премии.



Сотрудник / Вид расчета	Начало	Окончание	Регистратор	Результат
Гусаков Николай Дмитриевич				7 857,15
Оклад	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 1 от 09.05.2011 17:53:06	7 142,86
Премия	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 2 от 09.05.2011 17:59:08	714,29
Невыход	01.07.2011 0:00:00	10.07.2011 23:59:59	Начисления сотрудникам 3 от 09.05.2011 18:00:36	
Деловой Иван Сергеевич				8 800,00
Оклад	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 1 от 09.05.2011 17:53:06	8 000,00
Премия	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 2 от 09.05.2011 17:59:08	800,00
Симонов Валерий Михайлович				3 000,00
Оклад	01.07.2011 0:00:00	31.07.2011 23:59:59	Начисления сотрудникам 1 от 09.05.2011 17:53:06	3 000,00
Итого				19 657,15

Диаграмма Ганта

В конце этой работы создадим отчет, который в графическом виде будет показывать фактический период действия записей расчета.

Помимо наглядной демонстрации работы механизма вытеснения записей по периоду действия этот отчет позволит познакомиться с элементом формы, позволяющим создавать *диаграммы Ганта*.

Диаграмма Ганта представляет собой диаграмму интервалов на шкале времени и отражает использование объектами (точками) ресурсов (серий).

Эта диаграмма будет отображать для каждого сотрудника фактический период действия записи по каждому из видов расчета, имеющих место для этого сотрудника.

В нашем случае точками диаграммы являются сотрудники, а сериями – виды расчетов. Т.о. для каждого сотрудника существует некоторое значение диаграммы по каждой из серий, т.е. по каждому виду расчета.

Все интервалы всех значений диаграммы располагаются с привязкой к единой оси времени, что дает возможность видеть их взаимное расположение.

В качестве исходных данных для построения диаграммы, мы возьмем данные регистра расчета **Начисления**. Каждая запись этого регистра уже содержит все необходимое для построения диаграммы: сотрудника,

вид расчета, начало и конец интервала. Остается средствами встроенного языка разместить все это в диаграмме.

В режиме Конфигуратор

Создайте новый отчет **ДиаграммаНачислений**. Мы не будем создавать схему компоновки, а создадим основную форму отчета и обеспечим формирование и настройку диаграммы Ганта с помощью кода.

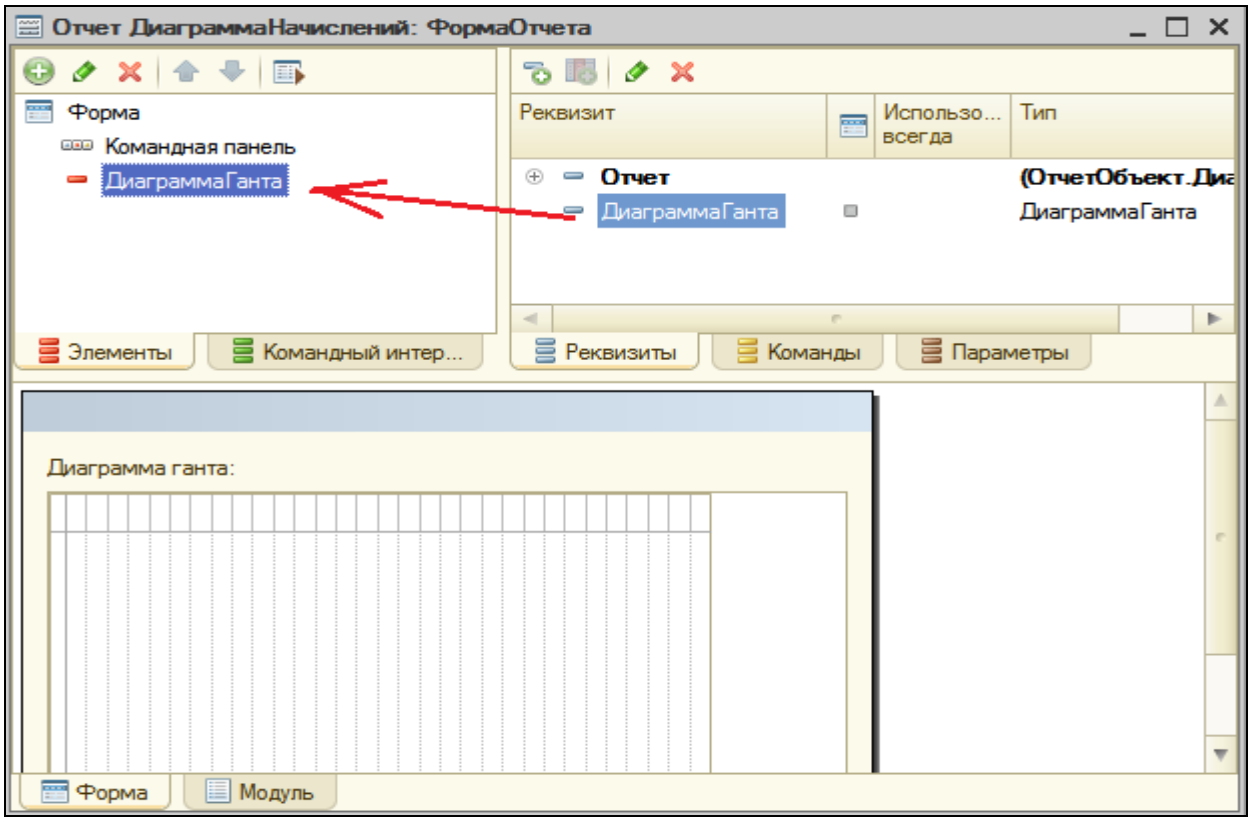
Для этого в окне редактирования отчета перейдите на закладку Формы, нажмите кнопку открытия и создайте основную форму отчета.

В правом верхнем углу редактора форм на закладке **Реквизиты** находятся реквизиты формы. Мы видим здесь основной реквизит формы **Отчет**, который был создан автоматически.

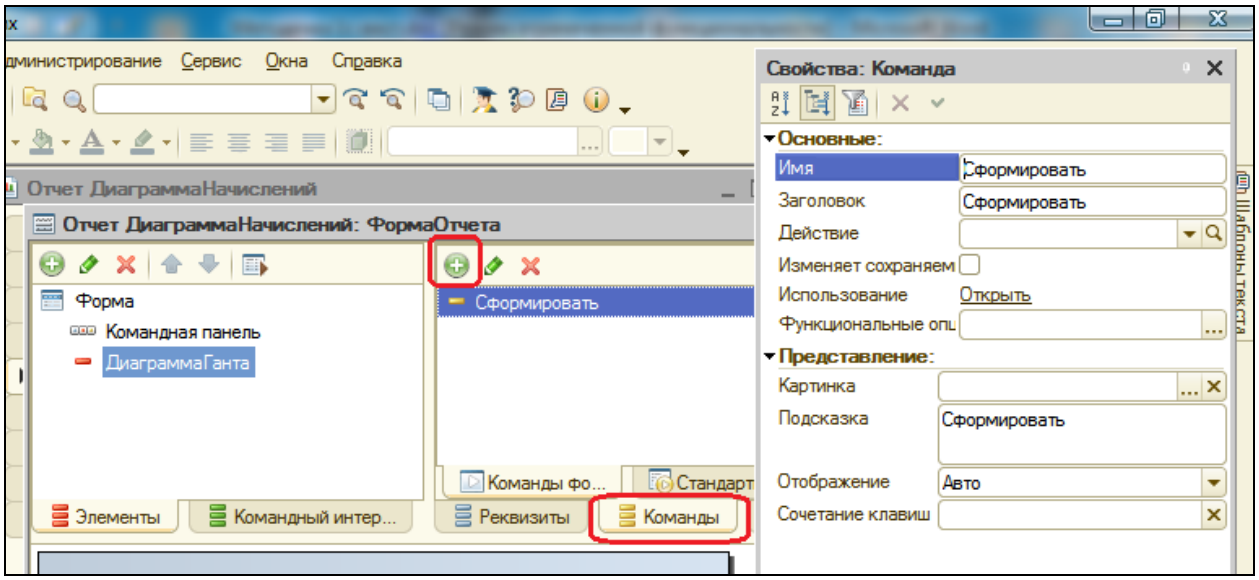
Нажмите кнопку **Добавить** и добавим новый реквизит формы. Назовем его **ДиаграммаГанта** с типом **ДиаграммаГанта**.

Теперь перетащим новый реквизит в окно элементов формы, которое пока пусто.

В окне элементов формы будет создано новое поле для отображения диаграммы Ганта, а в нижнем окне просмотра формы мы сразу увидим поле диаграммы.



На закладке **Команды** создадим команду формы **Сформировать**.



Теперь нужно установить **Действие** для этой команды. Для этого нажмите кнопку открытия в строке **Действие**.

В модуле формы будет создан шаблон процедуры Сформировать(), в которую мы поместим вызов процедуры СформироватьНаСервере() и в качестве параметра передадим в нее ссылку на реквизит формы ДиаграммаГанта.

```
&НаКлиенте
Процедура Сформировать(Команда)
	СформироватьНаСервере(ДиаграммаГанта)
КонецПроцедуры
```

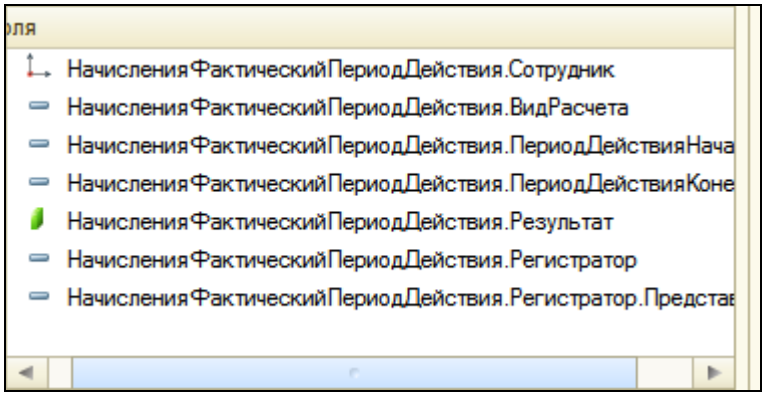
Процедуру **СформироватьНаСервере()** мы поместим также в модуле формы и предварим ее директивой исполнения **&НаСервереБезКонтекста**. В нее мы вставим заготовку запроса.

```
&НаСервереБезКонтекста
Процедура СформироватьНаСервере(Диаграмма)

	Запрос = Новый Запрос;
	Запрос.Текст = ;

КонецПроцедуры
```

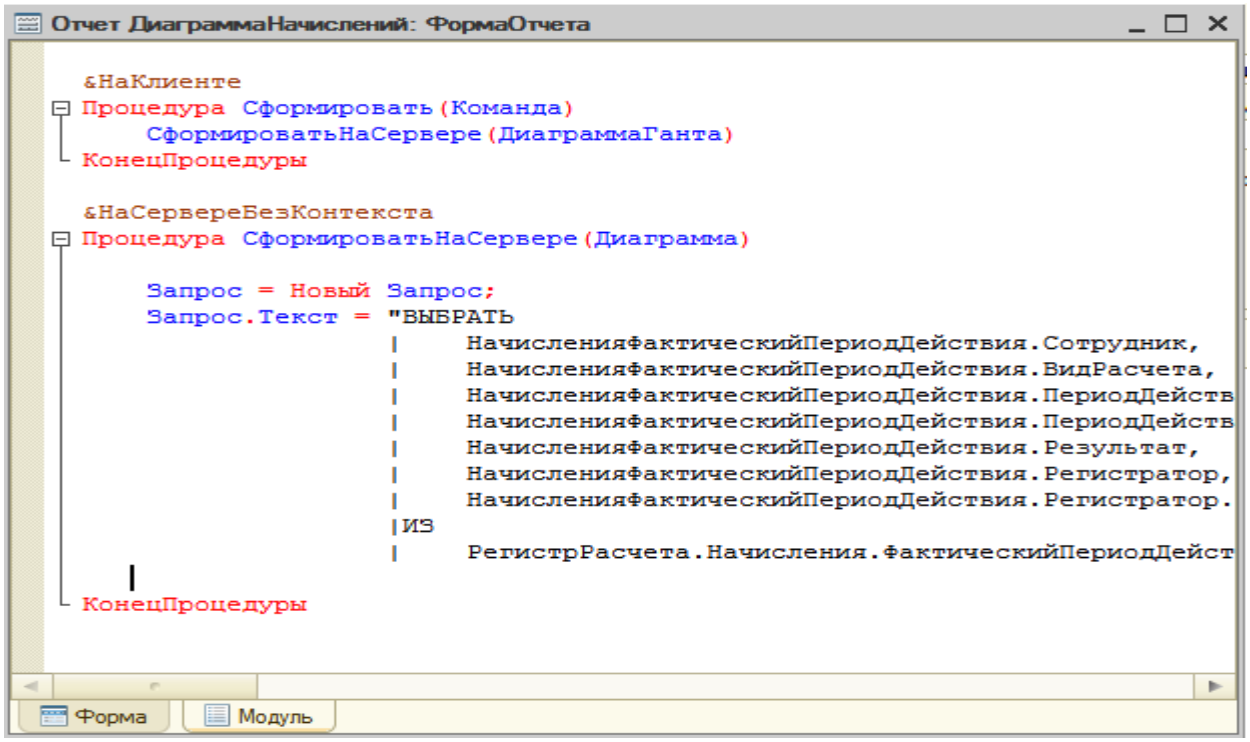
Установите курсор после равно перед точкой с запятой, вызовите контекстное меню, откройте конструктор запроса и создайте новый запрос.

Выберем виртуальную таблицу регистра расчета Начисления, **ФактическийПериодДействия**. Выберем из нее следующие поля:

- **Сотрудни**к,
- **ВидРасчета**,
- **ПериодДействияНачало**,
- **ПериодДействияКонец**,
- **Результат,**
- **Регистратор,**
- **Регистратор.Представление.**



Так будет выглядеть созданный запрос:



Нажмите ОК и после текста запроса добавим в процедуру текст:

```
                    | РегистрРасчета.Начисления.ФактическийПериодДействия КАК
НачисленияФактическийПериодДействия";

        ВыборкаРезультата = Запрос.Выполнить().Выбрать();

// Запретить обновление диаграммы
Диаграмма.Обновление = Ложь;

Диаграмма.Очистить();
Диаграмма.ОтображатьЗаголовок = Ложь;

// Заполнить диаграмму
Пока ВыборкаРезультата.Следующий() цикл
        // Получить серию, точку и значение для них
        ТекущаяСерия =
Диаграмма.УстановитьСерию(ВыборкаРезультата.ВидРасчета);
        ТекущаяТочка =
Диаграмма.УстановитьТочку(ВыборкаРезультата.Сотрудник);
        ТекущееЗначение = Диаграмма.ПолучитьЗначение(ТекущаяТочка,
ТекущаяСерия);

        // Создать нужные нам интервалы в значении
        ТекущийИнтервал = ТекущееЗначение.Добавить();
        ТекущийИнтервал.Начало = ВыборкаРезультата.ПериодДействияНачало;
        ТекущийИнтервал.Конец = ВыборкаРезультата.ПериодДействияКонец;
        ТекущийИнтервал.Текст =
ВыборкаРезультата.РегистраторПредставление;
        ТекущийИнтервал.Расшифровка = ВыборкаРезультата.Регистратор;
КонецЦикла;

// Раскрасить серии своими цветами
Для Каждого Серия из Диаграмма.Серии Цикл
        Если Серия.Значение = ПланыВидовРасчета.ОсновныеНачисления.Оклад
Тогда
                Серия.Цвет = WEBЦвета.Желтый;
        ИначеЕсли Серия.Значение =
ПланыВидовРасчета.ОсновныеНачисления.Премия Тогда
                Серия.Цвет = WEBЦвета.Зеленый;
        ИначеЕсли Серия.Значение =
ПланыВидовРасчета.ОсновныеНачисления.Невыход Тогда
                Серия.Цвет = WEBЦвета.Красный;
        КонецЕсли;
КонецЦикла;

// Разрешить обновление диаграммы
Диаграмма.Обновление = Истина;

КонецПроцедуры
```

Сначала мы запрещаем обновление диаграммы на то время, пока мы будем заполнять ее данными. Это нужно, чтобы в процессе заполнения не выполнялись пересчеты при каждом изменении данных диаграммы.

После окончания заполнения диаграммы мы разрешим обновление, и все пересчеты будут выполнены один раз.

Затем в цикле по выборке запроса мы заполняем диаграмму.

Сначала, используя методы **УстановитьСерию()** и **УстановитьТочку()**, мы получаем либо существующие, либо новые точку и серию. Точки и серии у нас сотрудник и вид расчета из запроса.

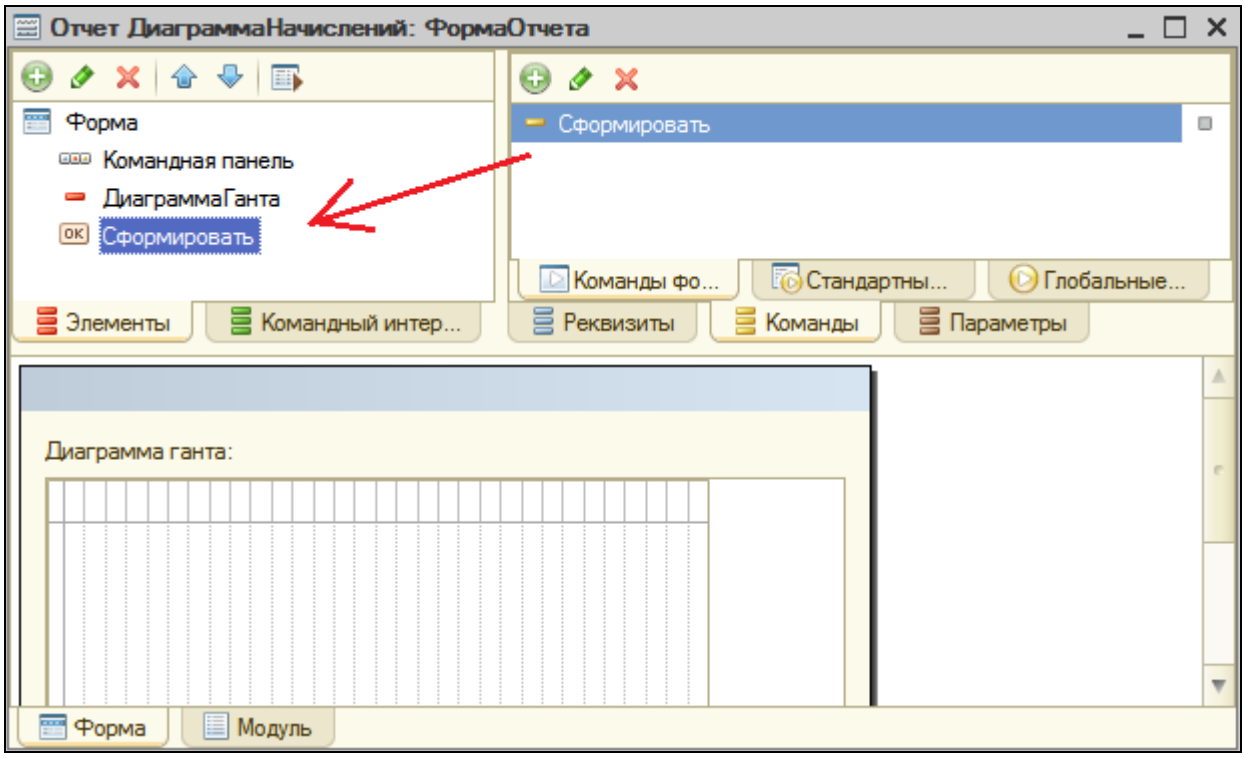
После получения точек и серий, с помощью метода ПолучитьЗначение(), мы получаем соответствующее им значение диаграммы.

Затем мы добавляем в значение диаграммы новый интервал, задаем его начало и конец, текст для всплывающей подсказки, расшифровку интервала, которая будет выполняться при двойном щелчке мыши на интервале.

После этого мы раскрашиваем серии своими цветами.

Вернемся в форму и добавим для нее кнопку для выполнения команды **Сформировать**.

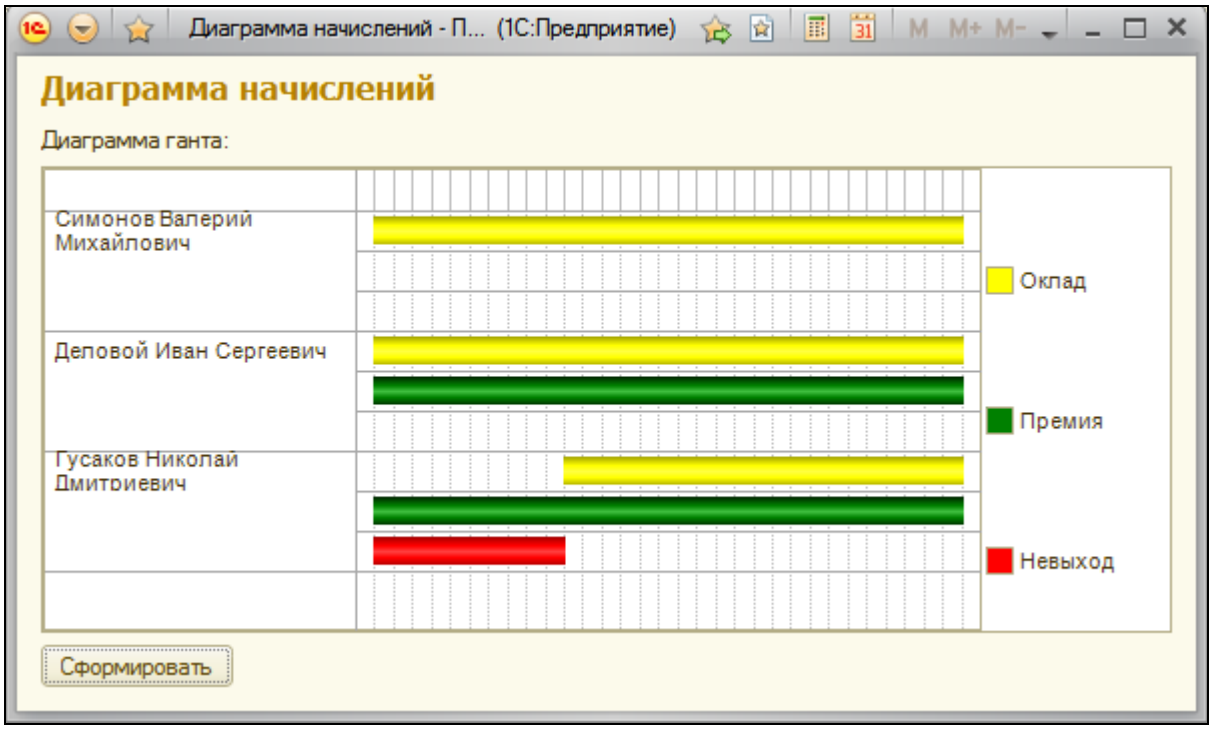
Для этого перетащим мышью команду **Сформировать** из окна **Команды формы** в окно элементов формы.



В заключение в окне редактирования отчета **ДиаграммаНачислений** укажем, что отчет будет вызываться из подсистемы **РасчетЗарплаты**.

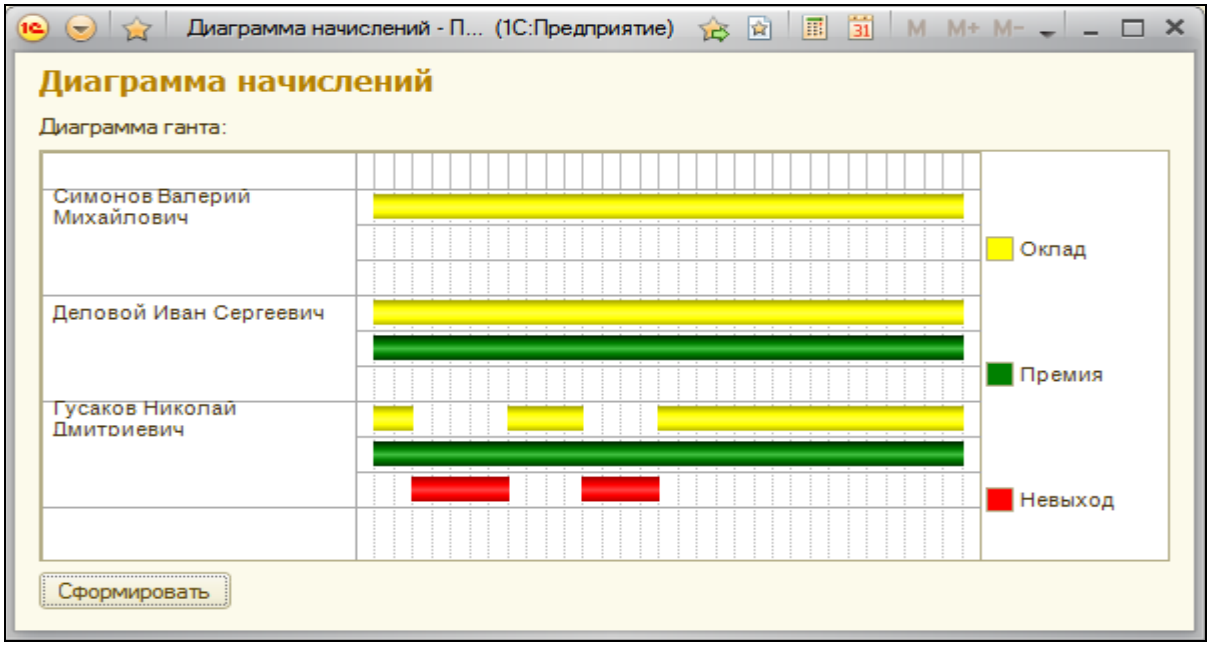
В режиме 1С:Предприятие

Запустите режим отладки и посмотрите на результат работы отчета.



А теперь посмотрим, как выглядит механизм вытеснения по периоду действия в работе.

Откройте документ **Начисления сотрудникам №3** и вместо одного прогула с 1 по 10 число, зададим Гусакову два прогула: с 3 по 7 число и с 12 по 15 число. Проведем документ и снова нажмем **Сформировать** в нашем отчете.



Контрольные вопросы

- ✓ Как запросом получить записи перерасчета.
- ✓ Как работает перерасчет.
- ✓ Как рассчитать записи регистра расчета.
- ✓ Как запросом получить данные графика и базы.
- ✓ Как выполнить перерасчет отдельных записей регистра расчета.
- ✓ Как устроена и для чего используется диаграмма Ганта.
- ✓ Как заполнить диаграмму Ганта данными.

Практическая работа № 18

Поиск в базе данных (1:30)

В этой работе мы познакомимся с механизмом полнотекстового поиска в данных и разработаете отчет, который позволит выполнять поиск в БД на основе полнотекстового индекса. На примере этого отчета вы научитесь разрабатывать форму «с нуля», наполняя ее реквизитами и командами.

В большой информационной базе просто необходим поиск. Поэтому система содержит механизм полнотекстового поиска в данных. Преимущества этого механизма в том, что он позволяет искать данные, вводя поисковый запрос в простой и естественной форме, например: «телефон абдулова». При этом можно использовать специальные операторы, вроде тех, что применяются при поиске в интернете.

Полнотекстовый поиск очень удобен, когда мы не знаем точно, где находятся нужные данные (например, в каком справочнике), и не знаем точно, что нужно искать (не помним точное название номенклатуры).

Кроме этих возможностей, он позволяет находить данные там, где другие методы поиска крайне трудоемки или требуют создания специальных алгоритмов и обработок.

Общие сведения о механизме полнотекстового поиска в данных

Механизм полнотекстового поиска 1С:Предприятия 8 основан на использовании двух составляющих:

- Полнотекстового индекса,
- Средств выполнения полнотекстового поиска

Для выполнения полнотекстового поиска обязательно должен существовать полнотекстовый индекс. Он создается один раз и затем должен периодически обновляться.

Поиск осуществляется по данным, которые содержатся в полнотекстовом индексе. Т.о., если ведется интенсивная работа с БД, то полнотекстовый индекс следует обновлять как можно чаще. Если же объем изменяемых или новых данных невелик, то обновление полнотекстового индекса

можно выполнять реже, например раз в сутки, в период наименьшей загруженности системы.

Создание и обновление индекса может выполняться как интерактивно, в режиме 1С:Предприятие, так и программно, средствами встроенного языка. В этой работе мы рассмотрим возможности интерактивного индексирования, а в следующей – как можно обновлять полнотекстовый индекс в автоматическом режиме.

В процессе работы информационной базы система отслеживает факт изменения данных в тех объектах конфигурации, которые могут участвовать в полнотекстовом поиске (планы счетов, планы видов расчета, планы обмена, справочники, документы, планы видов характеристик, регистры сведений, накопления, бухгалтерии, расчета).

Впоследствии при создании или обновлении полнотекстового индекса система анализирует данные, содержащиеся в реквизитах этих объектов, и включает эти данные в полнотекстовый индекс. При этом анализироваться могут не все реквизиты, а только которые имеют тип **Строка**, **Число**, **Дата**, **ХранилищеЗначения** или ссылочный тип.

Собственно сам полнотекстовый поиск выполняется средствами встроенного языка и в соответствии с правами пользователя. Если какая-то информация недоступна данному пользователю, он не сможет получить ее и при помощи полнотекстового поиска.

Приступим к первой части необходимых действий – созданию полнотекстового индекса. Второй частью будет создание отчета, который будет собственно выполнять полнотекстовый поиск, используя созданный нами индекс.

Полнотекстовый индекс

В режиме Конфигуратор

Каждый объект конфигурации, данные которого могут участвовать в полнотекстовом индексировании, имеет свойство **ПолнотекстовыйПоиск**. По умолчанию при создании нового объекта это свойство установлено в значение **Использовать**.

Кроме объектов конфигурации свойство ПолнотекстовыйПоиск существует и у реквизитов этих объектов. Т.о. мы имеем возможность указывать конкретные реквизиты, данные которых должны участвовать в полнотекстовом индексировании. По умолчанию оно также включено.

Для знакомства откройте свойства справочника **Номенклатура**. Найдите свойство **Полнотекстовый поиск** и убедитесь, что оно включено.

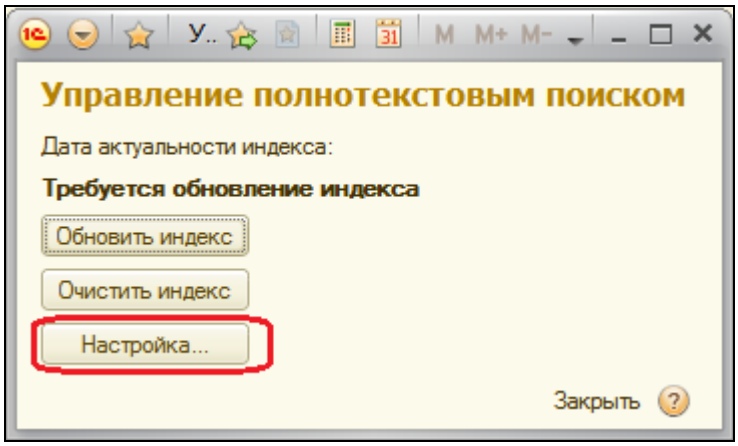
Теперь откройте свойства реквизита **ВидНоменклатуры** справочника **Номенклатура** и также убедитесь, что свойство включено.

Т.о. по умолчанию в нашей конфигурации полнотекстовый индекс используется для всех возможных реквизитов всех объектов конфигурации.

Перейдем в режим 1С:Предприятие.

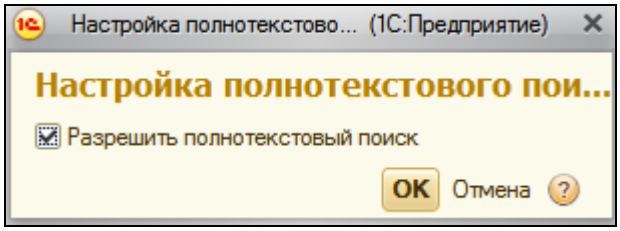
В режиме 1С:Предприятие

Выполним команду меню **Все функции – Стандартные - Управление полнотекстовым поиском.**



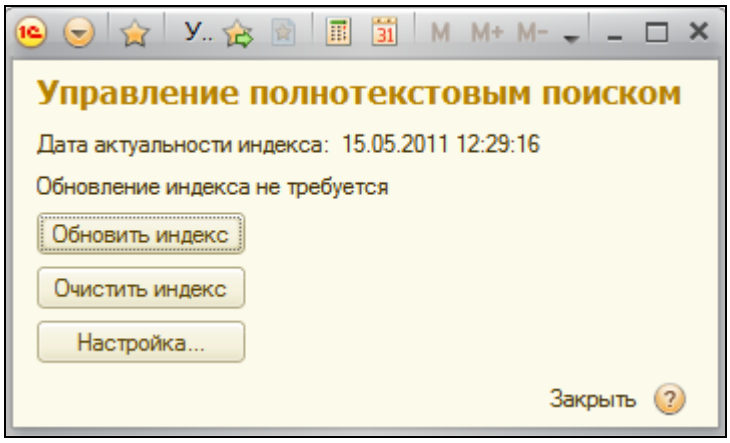
Это окно позволяет создавать и обновлять полнотекстовый индекс **интерактивно**. Кроме этого, позволяет разрешать или запрещать вообще все операции, связанные с полнотекстовым поиском: обновление, очистка полнотекстового индекса, полнотекстовый поиск.

Чтобы узнать, разрешены ли сейчас операции полнотекстового поиска, нажмите кнопку **Настройка...** Система откроет окно настройки полнотекстового поиска.



По умолчанию полнотекстовый поиск разрешен. Закройте это окно.

Система сообщает нам, что требуется обновление полнотекстового индекса, потому что в нашем случае индекс вообще отсутствует. Для создания или обновления индекса, нажмите кнопку **Обновить индекс**.



При больших размерах информационной базы создание и обновление полнотекстового индекса может занимать несколько минут.

Мы создали полнотекстовый индекс для нашей информационной базы.

Теперь перейдем к созданию отчета, который позволит выполнять полнотекстовый поиск в базе данных.

Отчет для поиска данных

В режиме Конфигуратор

Добавим в конфигурацию новый отчет с именем **ПоискДанных**.

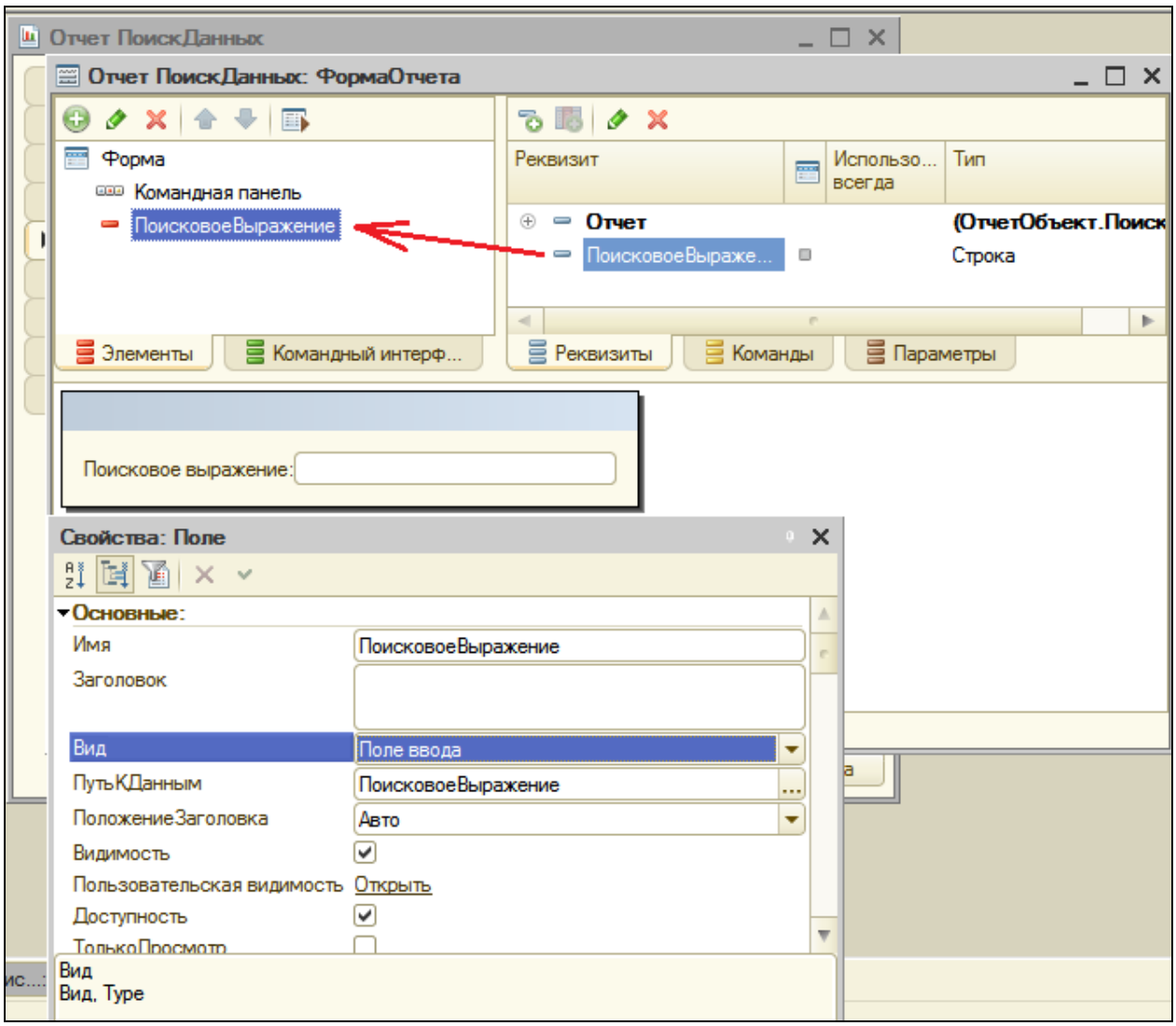
Перейдем на закладку **Формы**, нажмем кнопку открытия, создадим основную форму отчета и займемся ее редактированием.

Как мы уже говорили, элементы формы обязательно должны быть связаны с данными, иначе они не будут отображены. Поэтому сначала создадим соответствующие реквизиты и команды формы и затем перетащим их в окно элементов форм.

На закладке **Реквизиты** создадим реквизит **ПоисковоеВыражение** и перетащим его в окно элементов формы.

В открывшемся окне свойств поля **ПоисковоеВыражение** зададим его заголовок **Фраза**.

В поле **Вид** автоматически подставилось значение **Поле ввода** – это нам и нужно.

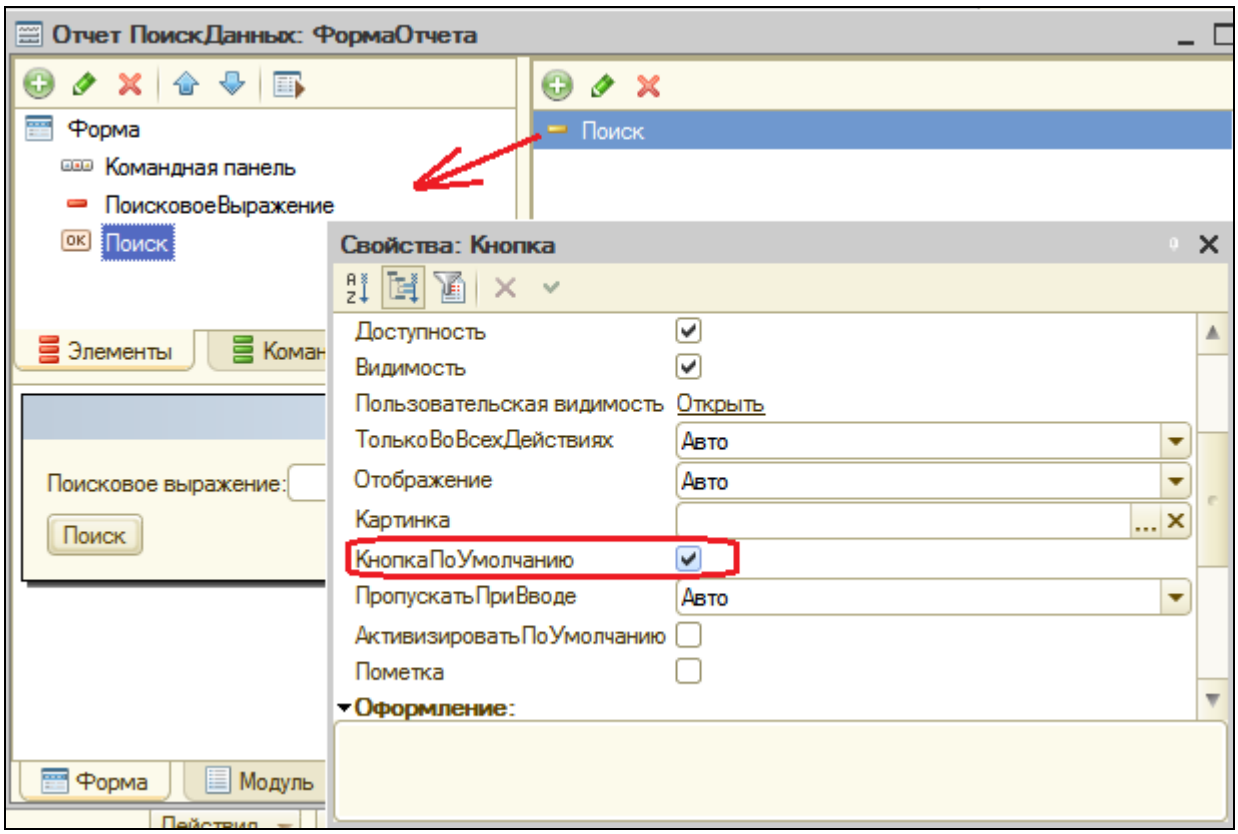


В поле ввода **ПоисковоеВыражение** мы будем вводить фразу для поиска в базе данных.

Затем на закладке **Команды** создадим команду **Поиск** и нажмем кнопку открытия в строке **Действие**.

Шаблон обработчика команды, открывшийся в модуле формы, пока заполнять не будем, а перейдем на закладку **Форма** и перетащим эту команду в окно элементов формы.

В открывшемся окне свойств кнопки **Поиск** поставим флажок **КнопкаПоУмолчанию**.

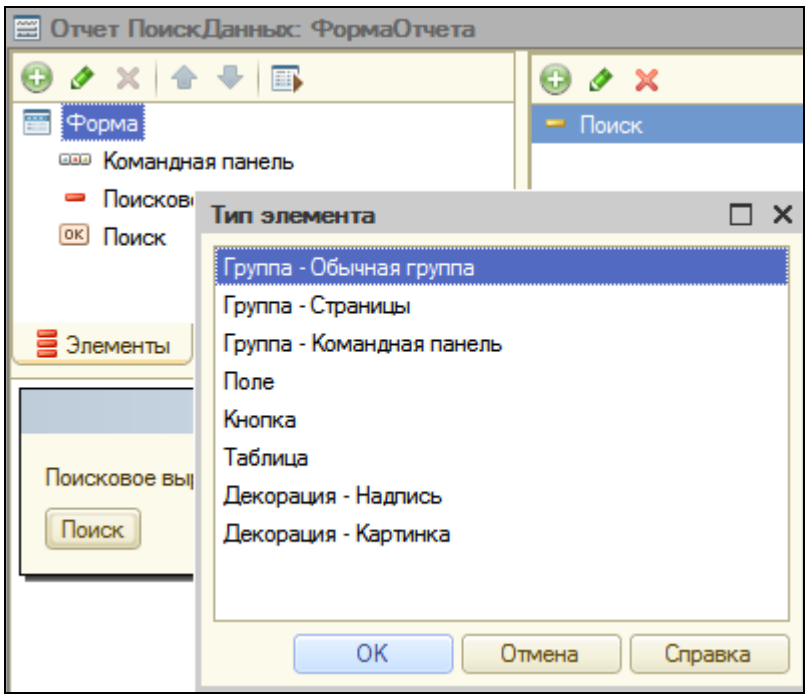


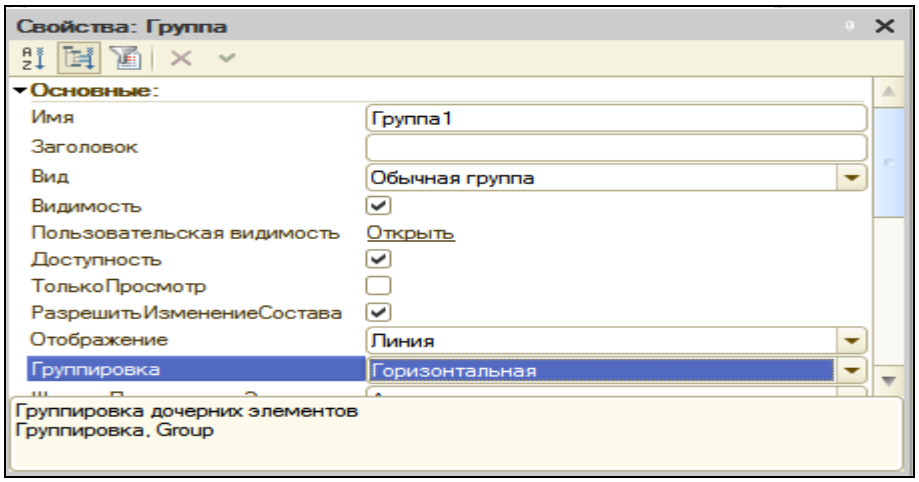
Однако мы видим, что все добавляемые элементы формы располагаются вертикально – друг под другом. Это потому что в свойствах формы установлена группировка элементов формы по умолчанию – **Вертикальная**. Нас это устраивает, но некоторые элементы формы, в частности поле **ПоисковоеВыражение** и кнопку **Найти**, хотелось бы расположить горизонтально – рядом друг с другом.

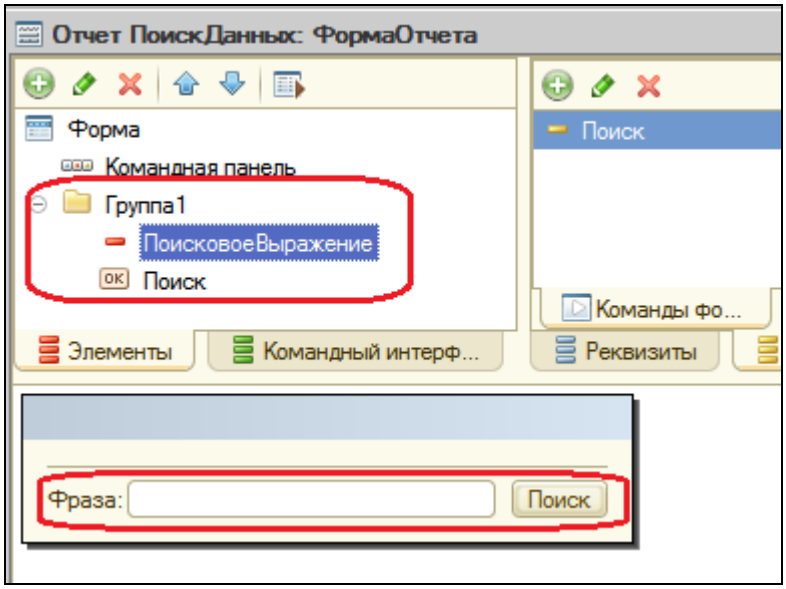
Для этого нужно добавить в форму группу и определить в ней тип группировки элементов **Горизонтальная**.

Выделим строку **Форма** в дереве элементов формы, нажмите кнопку **Добавить** в командной панели и выберем тип элемента **Группа – Обычная группа.**

В открывшемся окне свойств группы зададим тип группировки **Горизонтальная**. Затем мышью перетащим в эту группу элементы **ПоисковоеВыражение** и **Поиск**. Теперь мы добились желаемого расположения элементов.


Отчет ПоискДанных: ФормаОтчета
Форма
Командная панель
Поиск
Элементы
Поиск
Тип элемента
Группа - Обычная группа
Группа - Страницы
Группа - Командная панель
Поле
Кнопка
Таблица
Декорация - Надпись
Декорация - Картинка
OK
Отмена
Справка
Поисковое вы
Поиск

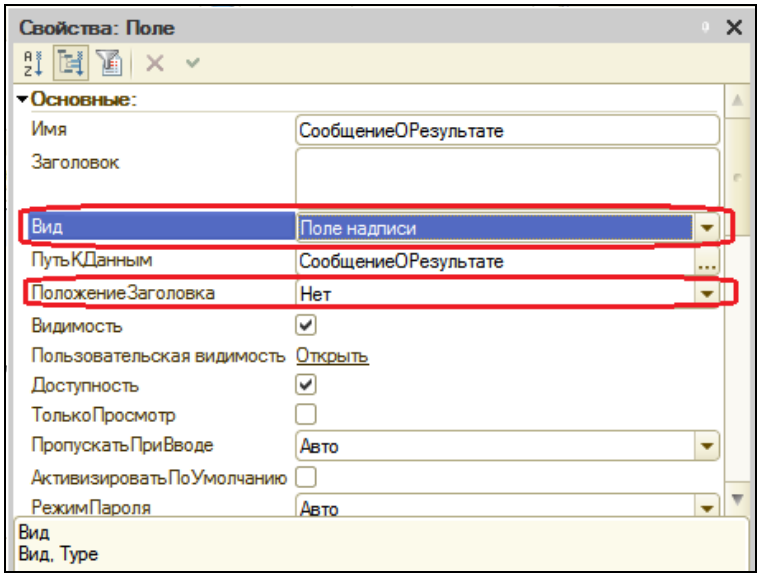

Свойства: Группа
Основные:
Имя
Группа1
Заголовок
Вид
Обычная группа
Видимость
Пользовательская видимость
Открыть
Доступность
ТолькоПросмотр
РазрешитьИзменениеСостава
Отображение
Линия
Группировка
Горизонтальная
Группировка дочерних элементов
Группировка, Group


Отчет ПоискДанных: ФормаОтчета
Форма
Командная панель
Группа1
ПоисковоеВыражение
Поиск
Поиск
Команды фо...
Элементы
Командный интерф...
Реквизиты
Фраза:
Поиск

Добавим в форму реквизит **СообщениеОРезультате** и перетащим его в окно элементов формы в группу 1. Появится окно свойств поля. В нем зададим **ПоложениеЗаголовка** в значение **Нет**.

В поле **Вид** установим значение **Поле надписи**.

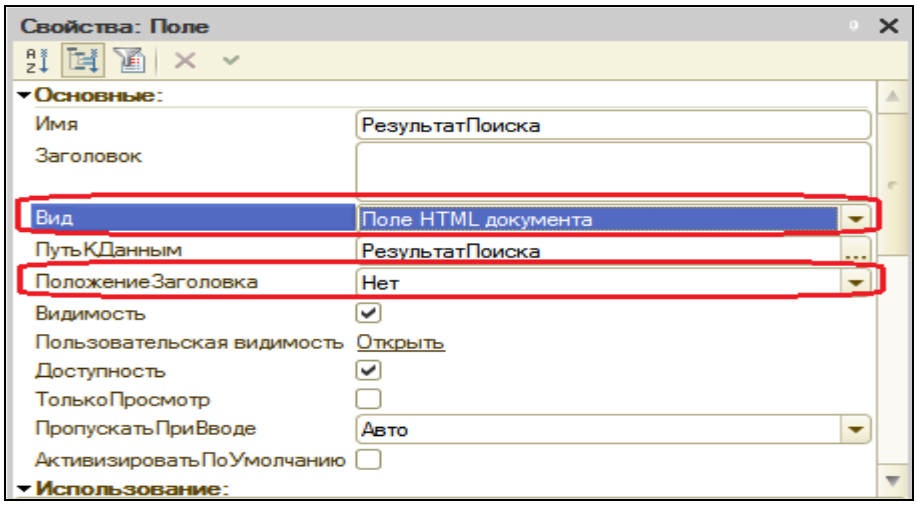
В поле надписи **СообщениеОРезультате** мы будем выводить сообщение о результате поиска.



Добавим в форму реквизит **РезультатПоиска** и перетащим его в окно элементов формы в **Группу1**.

В свойствах поля зададим **ПоложениеЗаголовка** в значение **Нет**. В поле **Вид** установим значение **Поле HTML документа**.

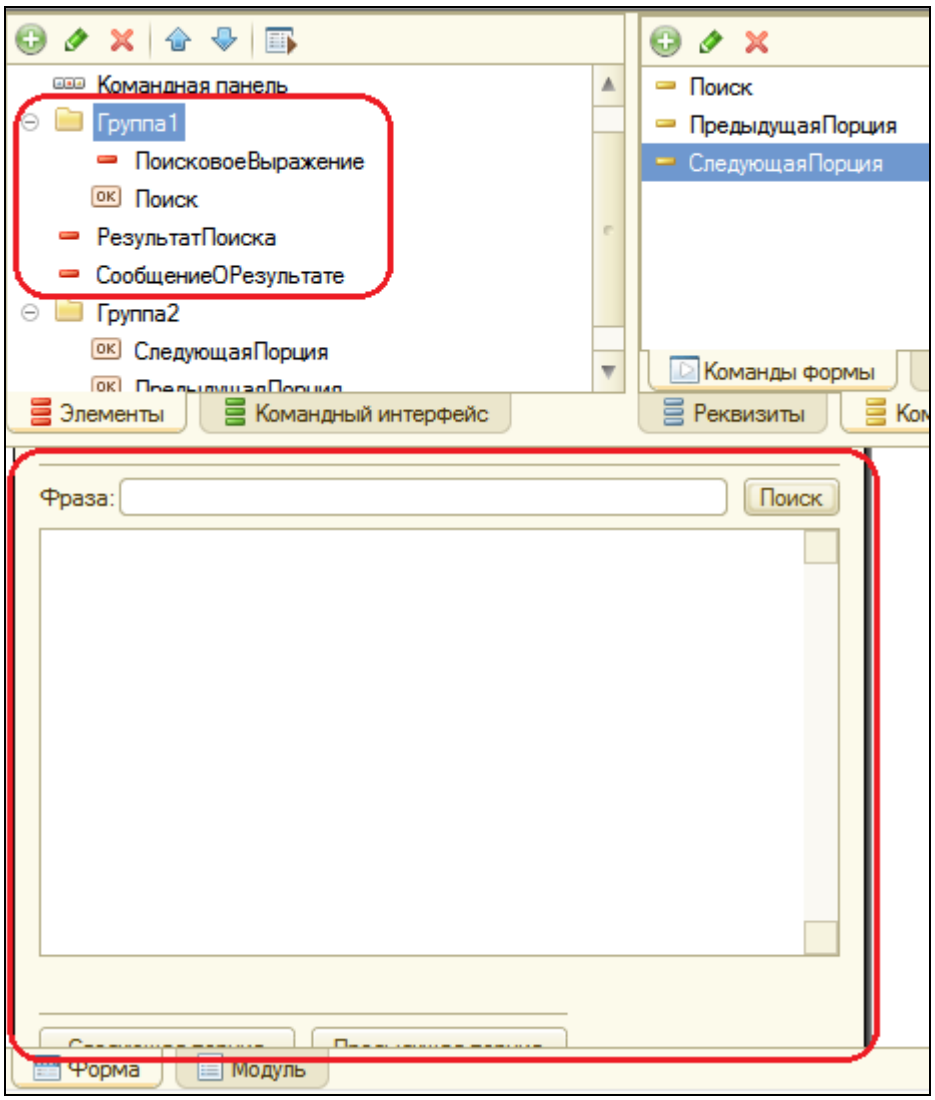
В поле HTML документа **РезультатПоиска** мы будем выводить найденные элементы поиска.



На закладке **Команды** поочередно создадим команды **ПредыдущаяПорция** и **СледующаяПорция.**

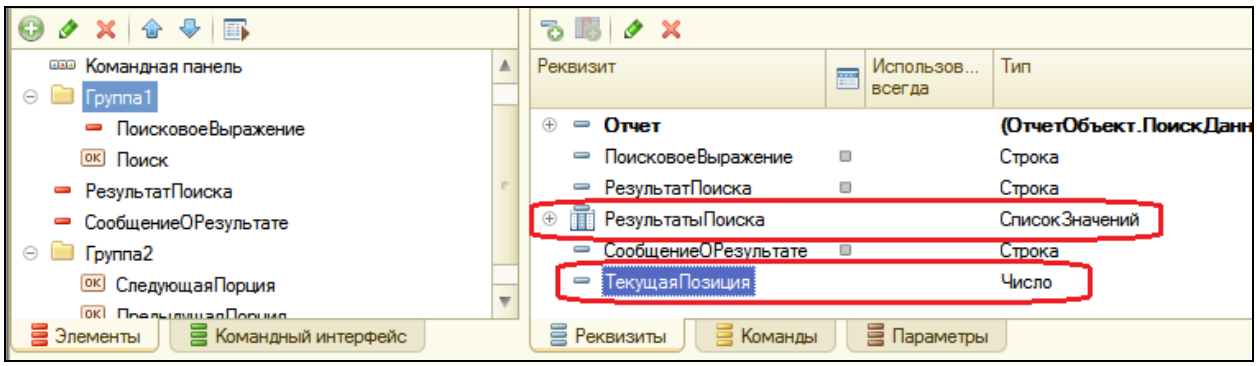
Нажмем кнопку открытия в строке **Действие** для каждой команды.

Шаблоны обработчиков событий пока заполнять не будем, а перейдем на закладку **Форма**, выделим корень дерева элементов и добавим новую группу. Зададим для этой группы тип группировки **Горизонтальная**. Затем перетащим наши команды в эту группу. Перетащите **Группу1** в командную панель, чтобы форма приняла вид:



Затем добавим в форму реквизит **РезультатыПоиска** типа **СписокЗначений** для хранения найденных элементов поиска. А также добавим в форму реквизит **ТекущаяПозиция** типа **Число** для хранения текущей позиции списка.

Эти реквизиты играют вспомогательную роль и в форму их перетаскивать не нужно.



Теперь реализуем работу формы с помощью кода на встроенном языке.

Для обработчиков событий нажатия кнопок **Поиск**, **Предыдущая порция** и **Следующая порция** напишем код, который позволит нам выполнять поиск в соответствии с направлением поиска (искать сначала, искать вперед или назад). Это делается на вкладке **Команды** в свойствах команды.

```
&НаКлиенте
Процедура Поиск()
        Искать(0);
КонецПроцедуры

&НаКлиенте
Процедура ПредыдущаяПорция()
        Искать(-1);
КонецПроцедуры

&НаКлиенте
Процедура СледующаяПорция()
        Искать(1);
КонецПроцедуры
```

Все эти обработчики вызывают процедуру **Искать()**. В ней проверяется, задано ли выражение для поиска, и вызывается собственно процедура полнотекстового поиска, выполняющаяся на сервере **ИскатьСервер()**, в которую передается направление поиска.

После этих процедур вставим следующий текст процедуры поиска на клиенте:

```
&НаКлиенте
// Процедура поиска, получение и отображение результата
Процедура Искать(Направление)
        Если ПустаяСтрока(ПоисковоеВыражение) Тогда
                Предупреждение("Не задана строка поиска.");
                Возврат;
        КонецЕсли;
```

```
        ИскатьСервер(Направление);
КонецПроцедуры
```

И процедуру поиска на сервере:

```
&НаСервере
Процедура ИскатьСервер(Направление) Экспорт
        СписокПоиска = ПолнотекстовыйПоиск.СоздатьСписок();
        СписокПоиска.СтрокаПоиска = ПоисковоеВыражение;

        Если Направление = 0 Тогда
                СписокПоиска.ПерваяЧасть();
        ИначеЕсли Направление = -1 Тогда
                СписокПоиска.ПредыдущаяЧасть(ТекущаяПозиция);
        ИначеЕсли Направление = 1 Тогда
                СписокПоиска.СледующаяЧасть(ТекущаяПозиция);
        КонецЕсли;

        РезультатыПоиска.Очистить();
        Для Каждого Результат Из СписокПоиска Цикл
                РезультатыПоиска.Добавить(Результат.Значение);
        КонецЦикла;

        РезультатПоиска =
СписокПоиска.ПолучитьОтображение(ВидОтображенияПолнотекстовогоПоиска.HTMLТек
ст);
        ТекущаяПозиция = СписокПоиска.НачальнаяПозиция();
        ПолноеКоличество = СписокПоиска.ПолноеКоличество();

        Если СписокПоиска.Количество() <> 0 Тогда
                СообщениеОРезультате = "Показаны " +        Строка(ТекущаяПозиция +
1) + " - " +
                        Строка(ТекущаяПозиция + СписокПоиска.Количество()) + " из " +
Строка(ПолноеКоличество);
                Элементы.СледующаяПорция.Доступность = (ПолноеКоличество -
ТекущаяПозиция) > СписокПоиска.Количество();
                Элементы.ПредыдущаяПорция.Доступность = (ТекущаяПозиция > 0);
        Иначе
                СообщениеОРезультате = "Не найдено";
                Элементы.СледующаяПорция.Доступность = Ложь;
                Элементы.ПредыдущаяПорция.Доступность = Ложь;
        КонецЕсли;
КонецПроцедуры
```

Сначала в этой процедуре мы создаем список поиска, используя метод **СоздатьСписок()** объекта **ПолнотекстовыйПоиск**, и сохраняем его в переменной **СписокПоиска**.

Затем устанавливаем поисковое выражение, введенное пользователем в качестве строки поиска. Затем в зависимости от направления поиска выполняем метод **ПерваяЧасть()**, **ПредыдущаяЧасть()** или **СледующаяЧасть()**, который собственно запускает полнотекстовый

поиск и возвращает соответственно первую порцию результатов, либо предыдущую порцию, либо следующую порцию в зависимости от текущей позиции поиска. По умолчанию порция содержит 20 элементов. Синонимом порции является страница с результатами поиска.

Затем мы очищаем список значений **РезультатыПоиска** и заполняем его найденными элементами.

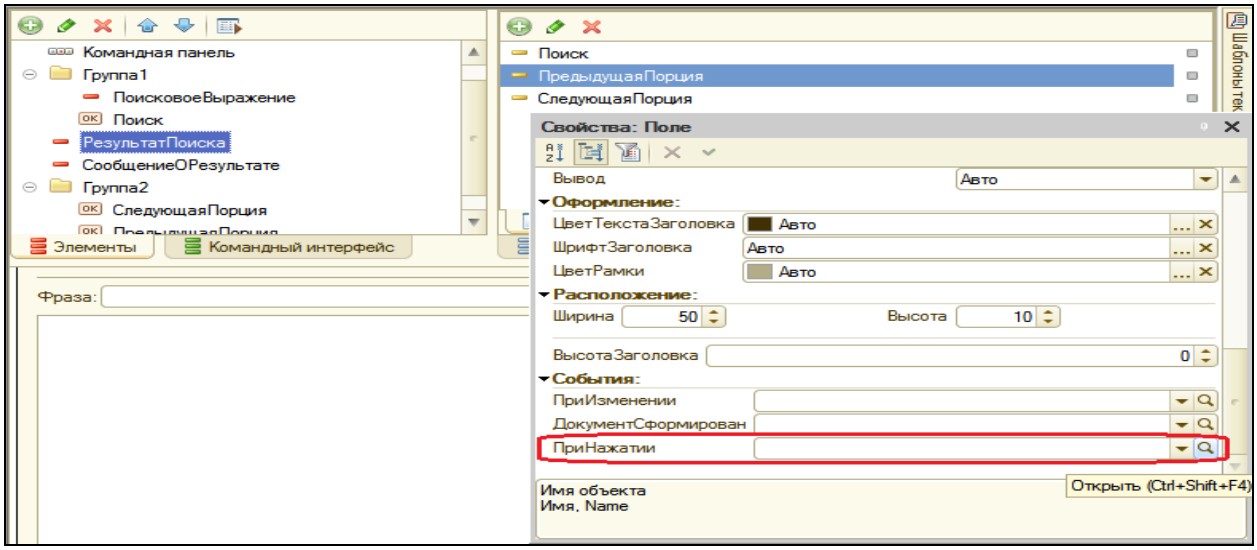
Получаем результат полнотекстового поиска в виде HTML-текста и сохраняем этот текст в реквизите **РезультатПоиска**, имеющим тип HTML-домента.

После этого мы анализируем количество элементов в списке поиска. Если он не содержит ни одного элемента, то мы выводим в форму соответствующее сообщение. В противном случае мы формируем сообщение о том, какие элементы отображены и сколько всего элементов найдено. В зависимости от того, какая порция полученных результатов отображена, мы устанавливаем доступность кнопок Предыдущая порция и Следующая порция.

Заключительным штрихом будет создание обработчика события **ПриНажатии** поля HTML-документа **РезультатПоиска**, расположенного в форме.

Результат поиска содержит гиперссылки на номера элементов списка поиска. Нам бы хотелось, чтобы при нажатии на ссылку система открывала форму того объекта, который содержится в этом элементе списка.

Для этого создадим обработчик события **ПриНажатии** поля HTML-документа **РезультатПоиска:**

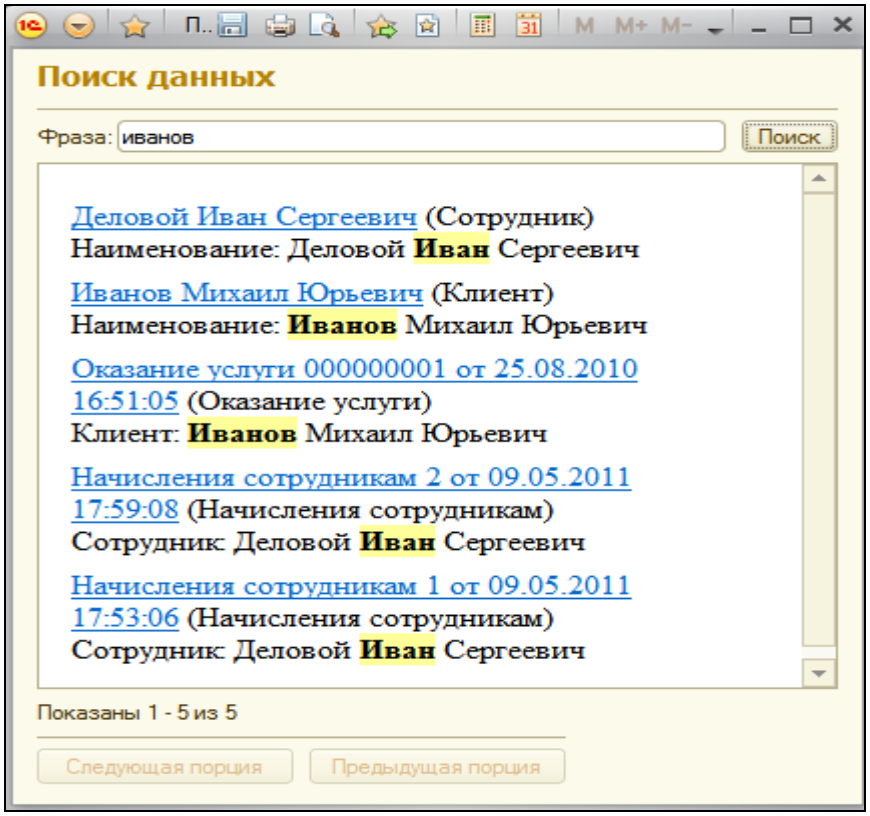


```
&НаКлиенте
Процедура РезультатПоискаПриНажатии(Элемент, ДанныеСобытия,
СтандартнаяОбработка)
        ЭлементHTML = ДанныеСобытия.Event.srcElement;
Если (ЭлементHTML.id = "FullTextSearchListItem") Тогда

        // Получить имя файла (номер строки списка поиска), содержащегося в
гиперссылке
        НомерВСписке = Число(ЭлементHTML.nameProp);

        // Получить строку списка поиска по номеру
        ВыбраннаяСтрока = РезультатыПоиска[НомерВСписке].Значение;

        // Открыть форму найденного объекта
        ОткрытьЗначение(ВыбраннаяСтрока);
        СтандартнаяОбработка = Ложь;
КонецЕсли;

КонецПроцедуры
```

В окне редактирования отчета **ПоискДанных** на закладке **Подсистемы** отметим все подсистемы, чтобы все пользователи в соответствии с их правами могли пользоваться поиском данных.

В режиме 1С:Предприятие

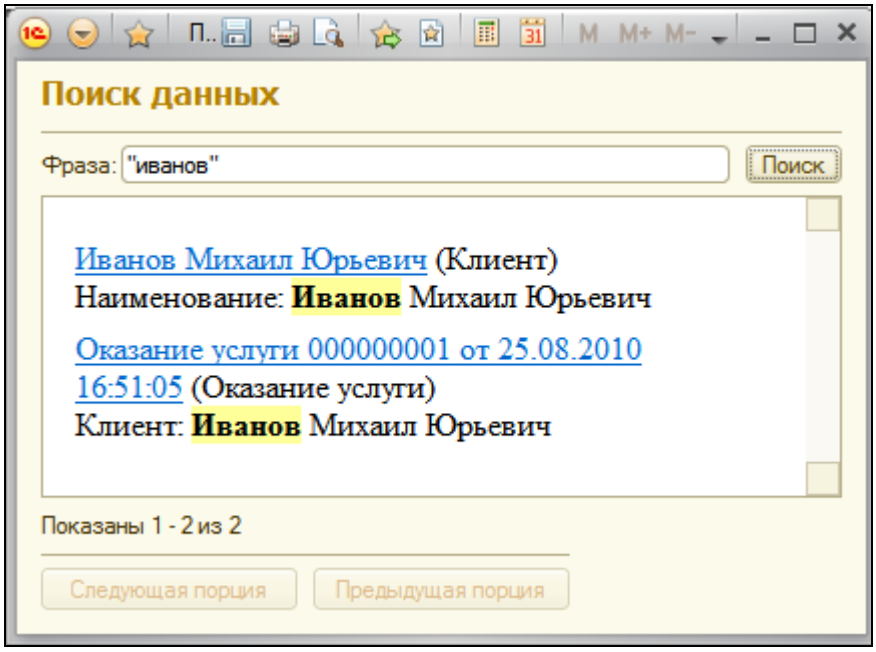
Для начала попробуем найти данные, связанные с Ивановым. Введем **иванов** и нажмем ctrl+enter или кнопку Поиск.



Результат поиска содержит 5 элементов и найденные слова в реквизитах этих документов выделены желтым фоном.

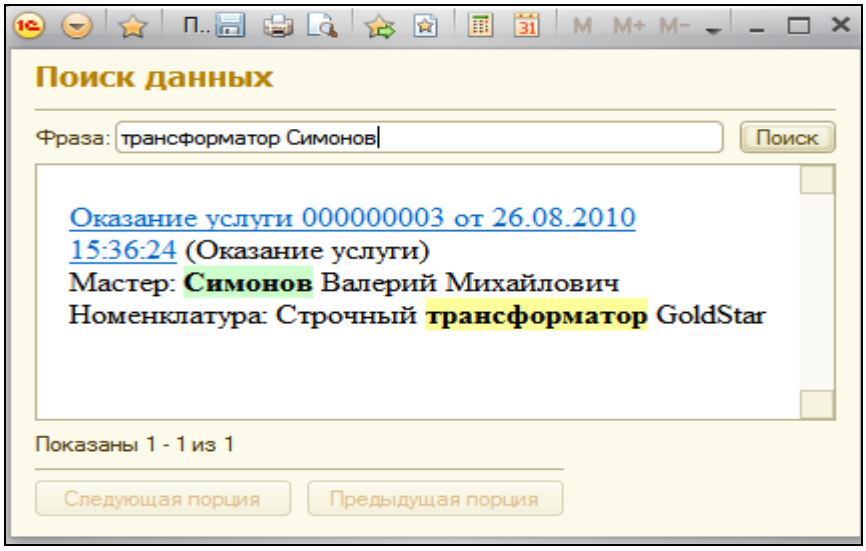
Обратите внимание, что система нашла все словоформы введенного выражения (Иван).

Чтобы выполнить точный поиск по указанному выражению, его необходимо заключить в кавычки (также и в интернете).



Для получения наилучших результатов полнотекстового поиска рекомендуется использовать в поисковом выражении пару слов.

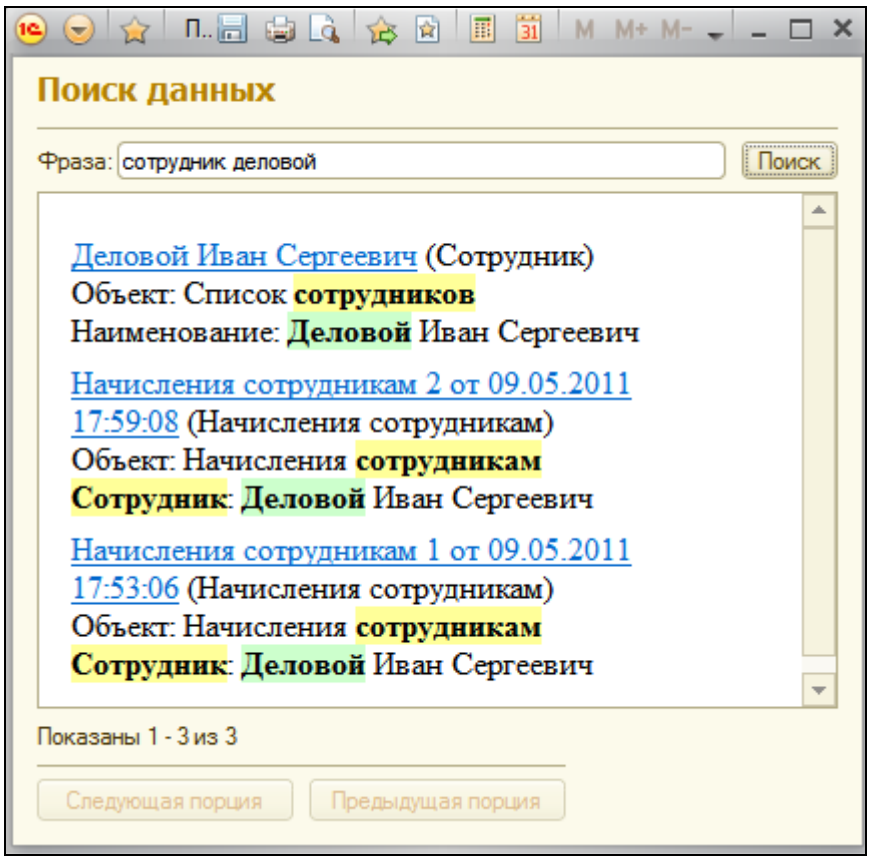
Например, если требуется узнать, какого числа клиенту Симонову заменили трансформатор в телевизоре, можно ввести поисковое выражение **трансформатор Симонов**.



При нажатии на гиперссылку система откроет документ Оказание услуги №3 и можно будет просмотреть полный перечень работ, выполненных для этого клиента.

Система индексирует не только данные, содержащиеся в объектах конфигурации, но и имена реквизитов и объектов метаданных.

Поэтому, например, если требуется найти информацию о сотруднике по фамилии Деловой, в поисковом выражении следует указать имя справочника, который нас интересует: **сотрудник деловой**.



Как видите, система нашла также и документы начисления сотрудникам, которые содержат формы слова «сотрудники», но, несмотря на это, искомый нами справочник Сотрудники находится первым в списке найденных.

Контрольные вопросы

- ✓ Для чего предназначен полнотекстовый поиск в данных.
- ✓ Какова стратегия полнотекстового индексирования инфобазы.
- ✓ Как создать отчет, выполняющий поиск в данных.
- ✓ Как составлять простейшие поисковые выражения.

Практическая работа № 19

Выполнение заданий по расписанию (1:00)

Внимание!

Если вы используете учебную версию программы, то пример, приведенный в этой работе, удастся проверить в работе лишь частично. Для полной проверки примера требуется одновременный запуск двух клиентских сеансов: планировщика заданий и обычного пользовательского сеанса.

Учебная версия позволяет запускать только один пользовательский сеанс. Поэтому их работу можно будет проверить только по отдельности. Сначала запустить планировщик заданий и убедиться, что регламентные задания запускаются. Затем закрыть сеанс планировщика и открыть обычный пользовательский сеанс, чтобы выполнить поиск в данных.

Любая информационная база системы 1С:Предприятие требует периодического выполнения определенного набора регламентных операций.

Например, по мере изменения существующих данных или добавления новых необходимо выполнять резервное копирование. Тогда, если в результате сбоя инфобаза окажется неработоспособной, основную часть данных можно будет восстановить из резервной копии. Поэтому чем чаще выполняется резервное копирование, тем меньшее количество данных придется вводить повторно в случае сбоя.

Другой пример – полнотекстовый поиск данных. Для него нужно индексировать данные по мере их обновления, т.е. с некоторой периодичностью.

Для автоматизации подобных операций существует механизм заданий. Он позволяет создавать задания, каждое из которых представляет собой некоторую последовательность действий, описанных с помощью встроенного языка. Для каждого задания может быть назначено расписание, в соответствии с которым это задание будет автоматически запущено на исполнение.

В этой работе мы рассмотрим использование механизма заданий на примере автоматизации двух регламентных операций, связанных с полнотекстовым поиском: операции полнотекстового индексирования и операции слияния индексов.

Мы опишем эти операции средствами встроенного языка, установим расписание для их автоматического выполнения и узнаем, как обеспечить автоматическое выполнение этих заданий по расписанию в случае файлового варианта работы системы.

Постановка задачи

В предыдущей работе мы узнали, что для возможностей выполнения полнотекстового поиска обязательно должен существовать полнотекстовый индекс. Он создается один раз и затем должен периодически обновляться.

На самом деле полнотекстовый индекс состоит из двух индексов: основного и дополнительного.

Поиск осуществляется по обоим индексам. Но отличие в следующем.

Основной индекс сделан так, чтобы обеспечивать максимальную скорость поиска при большом объеме данных, но добавление в него данных осуществляется медленно.

Дополнительный индекс противоположен основному: добавление данных в него осуществляется быстро, но при значительном объеме данных поиск будет выполняться медленно.

Стратегия использования индексов такова. Основная масса данных находится в основном индексе, что позволяет выполнить быстрый поиск. Новые данные, измененные или добавленные в систему, добавляются в дополнительный индекс непосредственно во время работы системы и пользователей с требуемой периодичностью (раз в час, раз в минуту). Такое добавление происходит быстро и не оказывает влияния на производительность системы. Пока объем данных в дополнительном индексе невелик, поиск по нему также выполняется быстро.

В период малой активности пользователей или в период выполнения регламентных действий с инфобазой выполняется слияние основного и дополнительного индексов (раз в сутки). В результате новые данные помещаются в основной индекс, а дополнительный индекс очищается и готов к быстрому приему следующих данных.

В результате для автоматизации полнотекстового индексирования нам понадобится два задания. Первое будет выполнять индексирование без слияния и запускаться каждую минуту. Второе будет выполнять слияние индексов и запускаться раз в сутки ночью.

Что такое регламентное задание

Регламентные задания располагаются в дереве объектов конфигурации, в ветке **Общие.** Каждое регламентное задание содержит два основных свойства: **Имя метода** и **Расписание**.

Свойство **Имя метода** связывает регламентное задание с некоторой процедурой или функцией общего модуля, которая и будет исполняться. Эта процедура должна содержать алгоритм на встроенном языке, описывающий все операции, которые должны быть выполнены.

Свойство **Расписание** позволяет задать периодичность выполнения этой процедуры.

Кроме перечисленных свойств регламентное задание содержит другие свойства, например **Интервал повтора при аварийном завершении** и **Количество повторов при аварийном завершении**.

Если по какой-то причине выполнение регламентного задания закончится неудачно, система может автоматически запустить это задание указанное количество раз по прошествии указанного периода времени.

Создание регламентных заданий

В режиме Конфигуратор

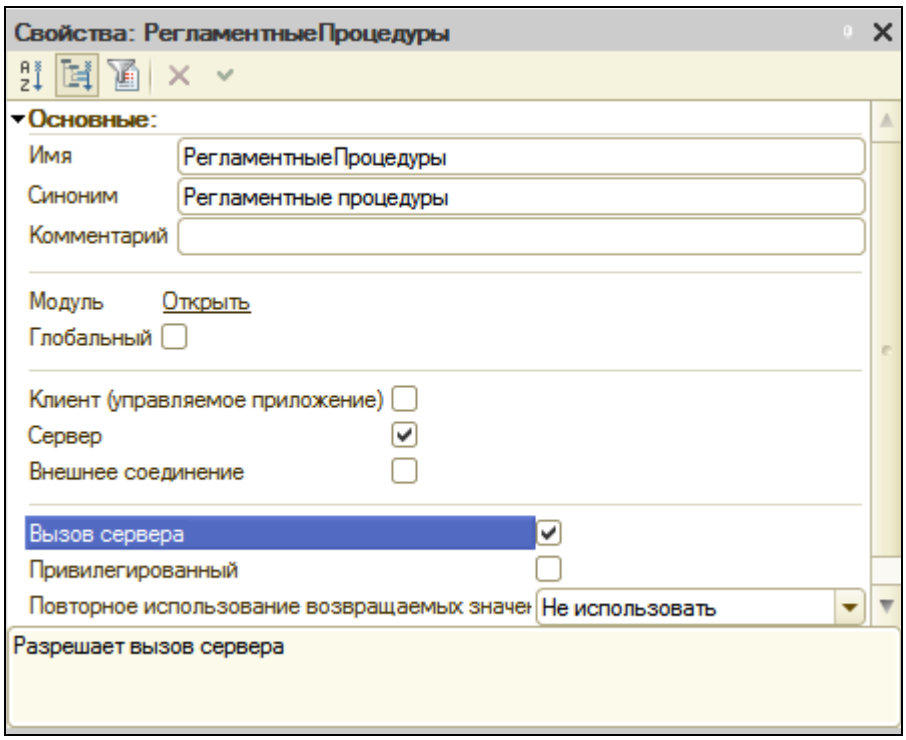
Сначала создадим первое регламентное задание по обновления индекса.

Раскроем ветвь **Общие** дерева объектов конфигурации. Выделим строку **Регламентные задания** и добавим новый объект **Регламентное задание** с именем **ОбновлениеИндекса**.

После этого создадим процедуру, которая и будет выполнять обновление полнотекстового индекса нашей информационной базы.

В качестве такой процедуры может выступать любая процедура или функция неглобального общего модуля, которую можно вызвать на сервере (у общего модуля должно быть установлено свойство **Сервер**).

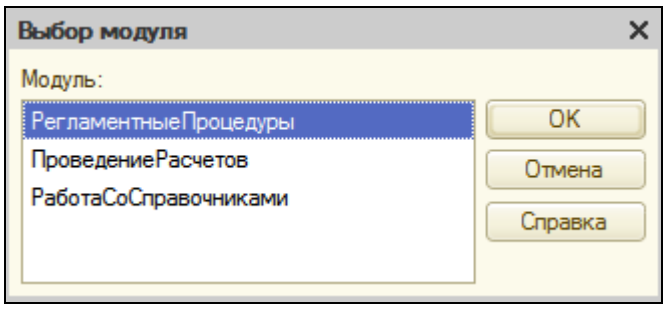
Добавим в конфигурацию общий модуль с именем **РегламентныеПроцедуры** и установим флажок **Вызов сервера** для видимости его экспортных процедур и функций.



Вернемся к свойствам регламентного задания **ОбновлениеИндекса**.

Нажмем кнопку выбора  у поля ввода **Имя метода**. Откроется окно выбора общего модуля. Выберем модуль **РегламентныеПроцедуры**.

В этом модуле будет создан шаблон процедуры **ОбновлениеИндекса().**



Заполним его следующим образом:

```
Процедура ОбновлениеИндекса() Экспорт
        Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =
РежимПолнотекстовогоПоиска.Разрешить Тогда
        Если Не ПолнотекстовыйПоиск.ИндексАктуален() Тогда
                ПолнотекстовыйПоиск.ОбновитьИндекс( , Истина);
        КонецЕсли;
КонецЕсли;
КонецПроцедуры
```

Сначала в этой процедуре проверяется возможность выполнения операций, связанных с полнотекстовым поиском (ведь они могут быть запрещены, например, интерактивно).

Если они разрешены, то проверяется, актуален ли полнотекстовый поиск (если после последнего индексирования данные, подлежащие полнотекстовому индексированию, не изменялись, то индекс будет актуален и повторное индексирование не требуется.

В случае необходимости индексирования вызывается метод **ОбновитьИндекс()** менеджера полнотекстового поиска.

Первый параметр этого метода отвечает за слияние индексов и по умолчанию имеет параметр **Ложь**, что значит, слияние выполняться не будет.

Второй параметр метода определяет, какое количество данных будет индексироваться: сразу все или порциями. Наша задача – выполнить индексирование как можно быстрее, поэтому указываем индексирование порциями (значение **Истина**).

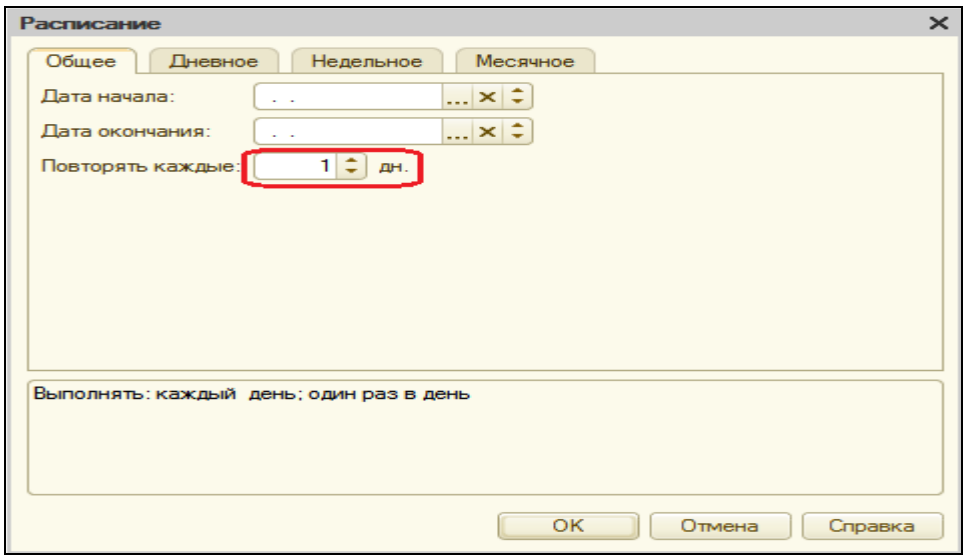
Размер одной порции фиксирован – 10 000 объектов. Т.о., если в данный момент требуется проиндексировать 15 000 объектов, то при вызове этого метода из них будет проиндексировано 10 000 (первая порция), а оставшиеся объекты – при следующем вызове этого метода (при следующем запуске регламентного задания).

Перейдем к составлению расписания запуска регламентного задания.

Нажмите ссылку **Открыть** в свойствах регламентного задания в строке Расписание. Откроется диалог редактирования расписания. Он содержит несколько вкладок, которые позволяют задать различные виды расписаний, в нижней части диалоги отображается итоговый результат всех установок.

Наша задача – запускать регламентное задание ежедневно, каждую минуту.

Поэтому на закладке **Общие** укажем, что запуск задания должен повторяться каждый день (Повторять каждые: 1 дн.).

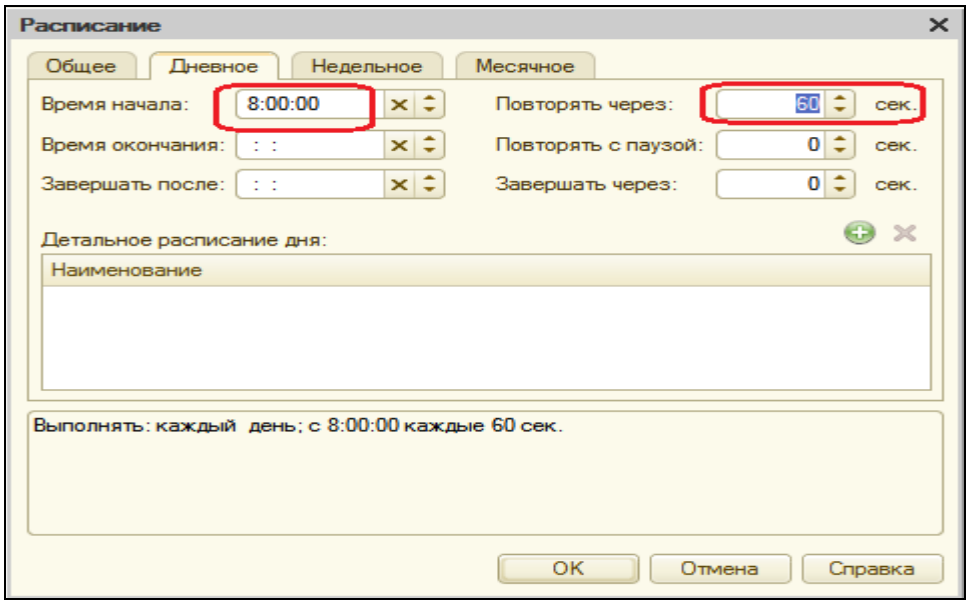


Теперь перейдем на закладку **Дневное** и зададим порядок запуска задания в течение дня.

Укажем, что запуск задания должен повторяться каждые 60 секунд (Повторять через: 60 сек.).

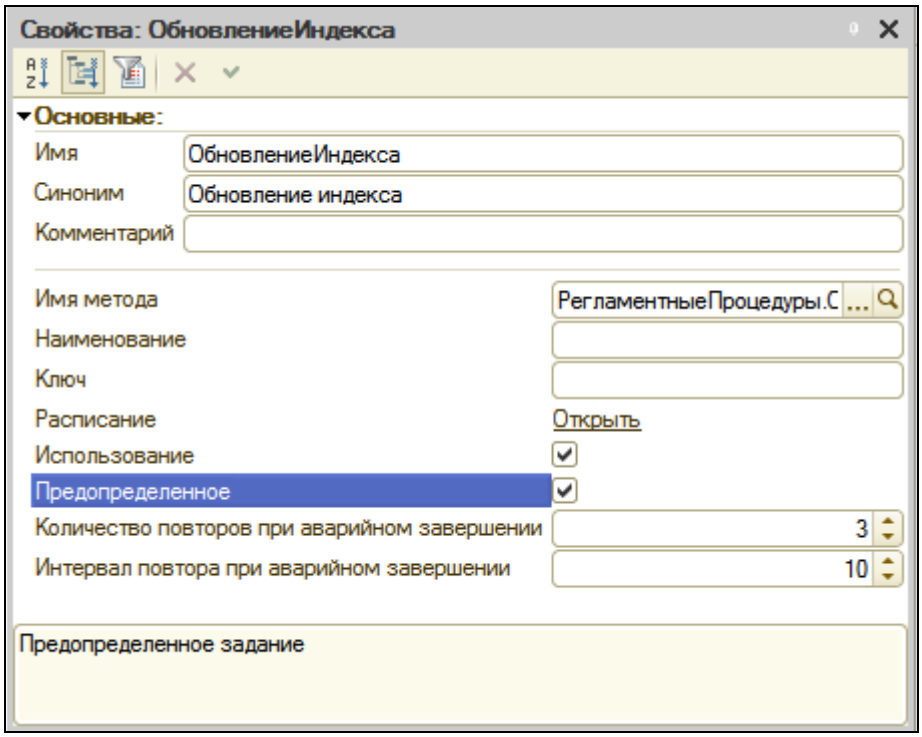
Вроде бы мы получили что хотели, но наша фирма не работает круглосуточно, и запуск этого задания в ночное время будет явно бесполезным - данные в базе не изменяются. В то же время вполне возможно, что некоторые сотрудники задерживаются после окончания рабочего дня.

Поэтому доработаем расписание: укажем Время начала: 08:00.



Нажмите ОК.

В качестве последнего штриха установим в свойствах регламентного задания флажок **Предопределенное**.



Установка этого свойства означает, что после запуска системы в режиме 1С:Предприятие будет создано одно предопределенное регламентное задание. В противном случае такое задание пришлось бы создавать средствами встроенного языка.

На этом создание регламентного задания **Обновление индекса** завершено.

Теперь по аналогии создадим второе регламентное задание – **СлияниеИндексов**.

В свойствах нажмем кнопку выбора  у поля ввода **Имя метода**. В открывшемся диалоге выберем модуль **РегламентныеПроцедуры**.

В этом модуле будет создан шаблон процедуры **СлияниеИндексов()**. Заполним его следующим образом:

```
Процедура СлияниеИндексов() Экспорт
        Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =
РежимПолнотекстовогоПоиска.Разрешить Тогда
        Если Не ПолнотекстовыйПоиск.ИндексАктуален() Тогда
                ПолнотекстовыйПоиск.ОбновитьИндекс(Истина);
        КонецЕсли;
КонецЕсли;

КонецПроцедуры
```

В свойствах задания установим флажок **Предопределенное** и приступим к редактированию расписания.

На закладке Общее укажем, что задание будет запускаться каждый день, а на закладке **Дневное** укажем время начала выполнения задания – **01:00**.

В результате мы получим следующее расписание запуска регламентного задания: **Выполнять: каждый день; с 1:00:00 один раз в день.**

На этом создание регламентного задания **СлияниеИндексов** закончено.

Планировщик заданий

1С:Предприятие поддерживает два варианта работы: файловый и клиент-серверный.

Файловый вариант прост в установке и практически не требует никакого администрирования. Именно в нем работает наша демонстрационная база. Однако, за любую простоту использования нужно чем-то расплачиваться.

Если бы наша информационная база работала в клиент-серверном варианте, то для автоматического запуска и выполнения созданных нами заданий не требовалось бы дополнительных действий. Можно было бы обновить конфигурацию базы данных, и менеджер кластера серверов 1С:Предприятия начал бы самостоятельно выполнять задания в соответствии с указанным расписанием.

В нашем случае (файловый вариант работы) такого «промежуточного звена», которое могло бы взять на себя автоматическое выполнение заданий, не существует – есть только клиенты и информационная база.

Поэтому за простоту файлового варианта придется заплатить тем, что для автоматического выполнения заданий необходимо всегда иметь одно работающее клиентское соединение с информационной базой, которое будет заниматься только запуском заданий по расписанию. Назовем такое соединение *планировщик заданий*.

В нашем примере мы создадим простую обработку, которая будет запускать задания по расписанию. Затем соединимся с нашей информационной базой в режиме 1С:Предприятие и запустим эту обработку.

Главное – не закрывать обработку и не закрывать это окно 1С:Предприятия, т.к. именно в нем будут выполняться регламентные задания.

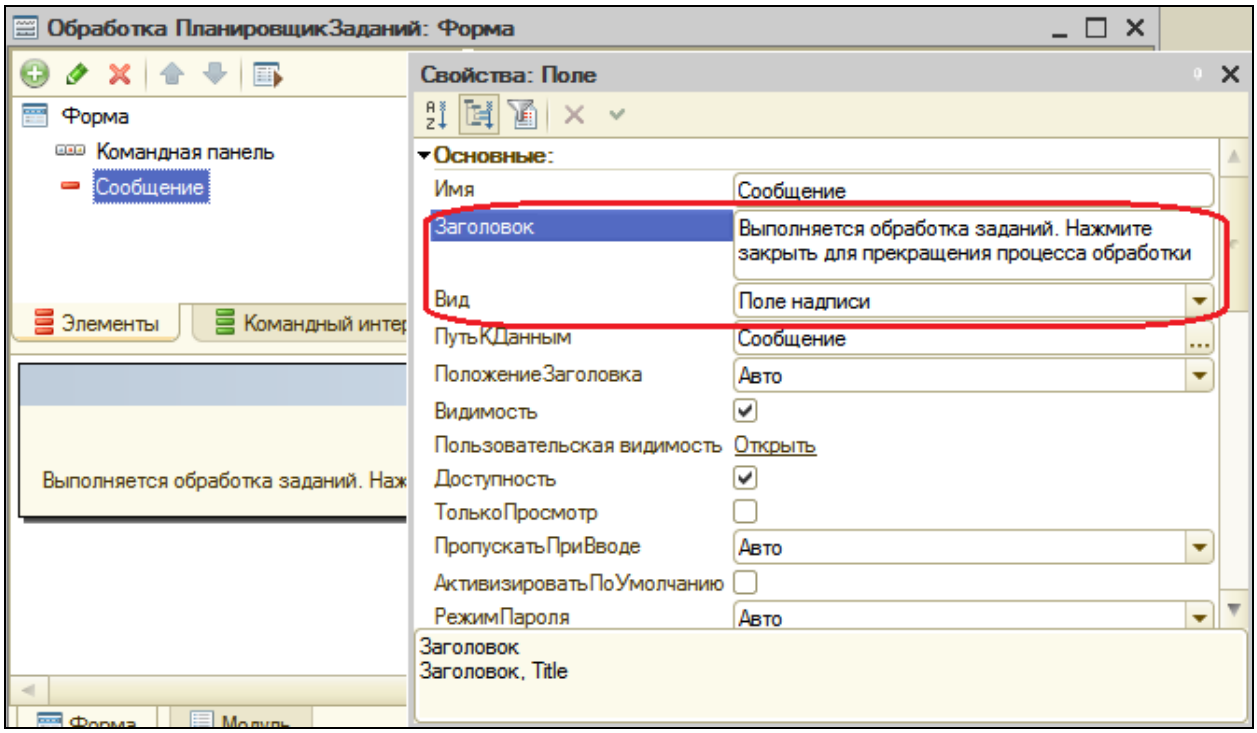
В режиме Конфигуратор

Выделим ветвь **Обработки** в дереве объектов конфигурации и добавим новый объект *Обработка* с именем **ПланировщикЗаданий**.

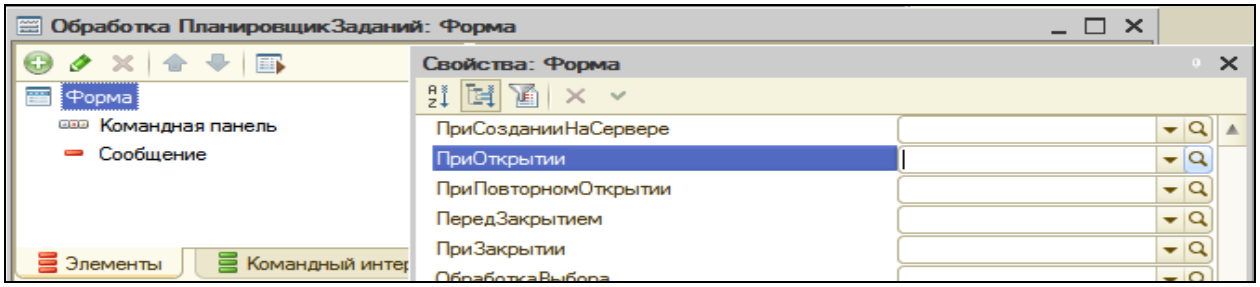
На закладке **Формы** создадим основную форму обработки.

В окне редактора форм на закладке **Реквизиты** добавим реквизит формы **Сообщение** и перетащим его в окно элементов формы.

В открывшемся окне свойств поля **Сообщение** зададим вид поля – **Поле надписи** и заголовок – **Выполняется обработка заданий**. **Нажмите закрыть для прекращения процесса обработки.**



Откроем события формы и нажмем кнопку открытия у события **ПриОткрытии**. В обработчик этого события поместим следующий текст:



```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
	#Если ТолстыйКлиентУправляемоеПриложение Тогда
	ПодключитьОбработчикОжидания("ОбработкаЗаданий", 3);
#Иначе
	Предупреждение("Обработка может быть запущена только в толстом клиенте!");
	Закрыть();
#КонецЕсли
КонецПроцедуры
```

А также в модуле формы поместим сам обработчик ожидания – процедуру **ОбработкаЗаданий**().

```
&НаКлиенте
Процедура ОбработкаЗаданий()

	#Если ТолстыйКлиентУправляемоеПриложение Тогда
		ВыполнитьОбработкуЗаданий();
	#КонецЕсли

КонецПроцедуры
```

Таким образом, при открытии формы выполняется подключение в качестве обработчика ожидания процедуры с именем **ОбработкаЗаданий()**.

Эта процедура будет вызываться каждые три секунды.

В свою очередь, процедура **ОбработкаЗаданий()** выполняет одно-единственное действие – вызывает метод **ВыполнитьОбработкуЗаданий()**, который проверяет, существуют ли задания, которые в соответствии с их расписанием, должны быть выполнены. Если такие задания существуют, он запускает их на выполнение.

В обеих процедурах используются инструкции препроцессора (после символа #) для того, чтобы указать, что фрагмент кода будет присутствовать только в том случае, если запущен толстый клиент в управляемом режиме, т.к. именно в этом режиме будет запускаться наша обработка **ПланировщикЗаданий**.

Для того, чтобы проверить, что запуск действительно происходит, добавим в начало процедуры **ОбновлениеИндекса** (общий модуль **РегламентныеПроцедуры**) следующую строку:

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "Запуск регламентного задания Обновление индекса " +
ТекущаяДата();
Сообщение.Сообщить();
```

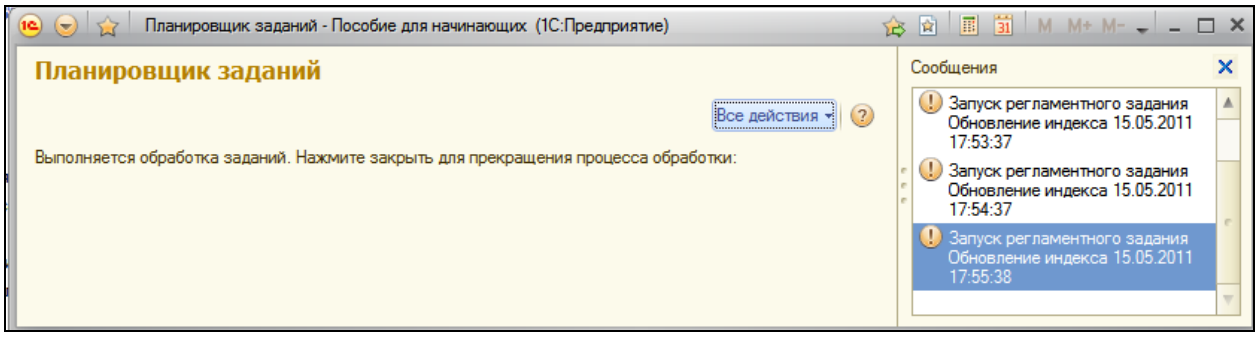
В заключение в окне редактирования объекта *Обработка* **ПланировщикЗаданий** отметьте подсистему **Предприятие**.

В режиме 1С:Предприятие

Обновим конфигурацию базы данных, нажав кнопку в Конфигураторе в панели инструментов **Обновить конфигурацию базы данных (F7)** и после этого запустим систему в режиме 1С:Предприятие (толстый клиент). Для этого зайдите в **Пуск - Все программы-1С Предприятие 8.2 – Дополнительно - <номер версии> - 1С Предприятие (толстый клиент).** В окне запуска откроем в режиме 1С:Предприятие нашу информационную базу.

В разделе **Предприятие** откроем обработку **Планировщик заданий** и подождем несколько минут.

В результате в окно сообщений будет выведен, например, такой текст:



Таким образом, мы видим, что задание обновления индекса запускается каждые 60 секунд, как мы и указали в расписании.

В соединении, которое занято запуском регламентных заданий, не рекомендуется выполнять какие-либо другие задачи. Для обычной работы с нашей информационной базой необходимо еще раз запустить систему в режиме 1С:Предприятие, выбрать нашу информационную базу и создать тем самым второе соединение, в котором уже и выполнять обычную работу пользователя.

Контрольные вопросы

- ✓ Для чего предназначены регламентные задания.
- ✓ Как задать расписание для автоматического запуска заданий.
- ✓ Как обеспечить запуск заданий по расписанию в файловом варианте работы.
- ✓ Что такое основной и дополнительный полнотекстовые индексы.

Практическая работа № 20

Редактирование движений в форме документа (00:40)

В нашей информационной базе, как и в любой другой, следует предусмотреть возможность ввода начальных остатков в регистры. Это необходимо для того, чтобы пользователи могли начать работу с нашей информационной базой не с чистого листа, а с некоторого исходного состояния, которое было в их прежней системе учета (на бумаге например).

Задача ввода начальных остатков отличается от прочих алгоритмов изменения состояния регистров тем, что подразумевает изменение данных непосредственно в регистрах, без использования промежуточных алгоритмов (заполнения документов данными, проведения документов, контроля правильности данных, указанных в документах и т.д.).

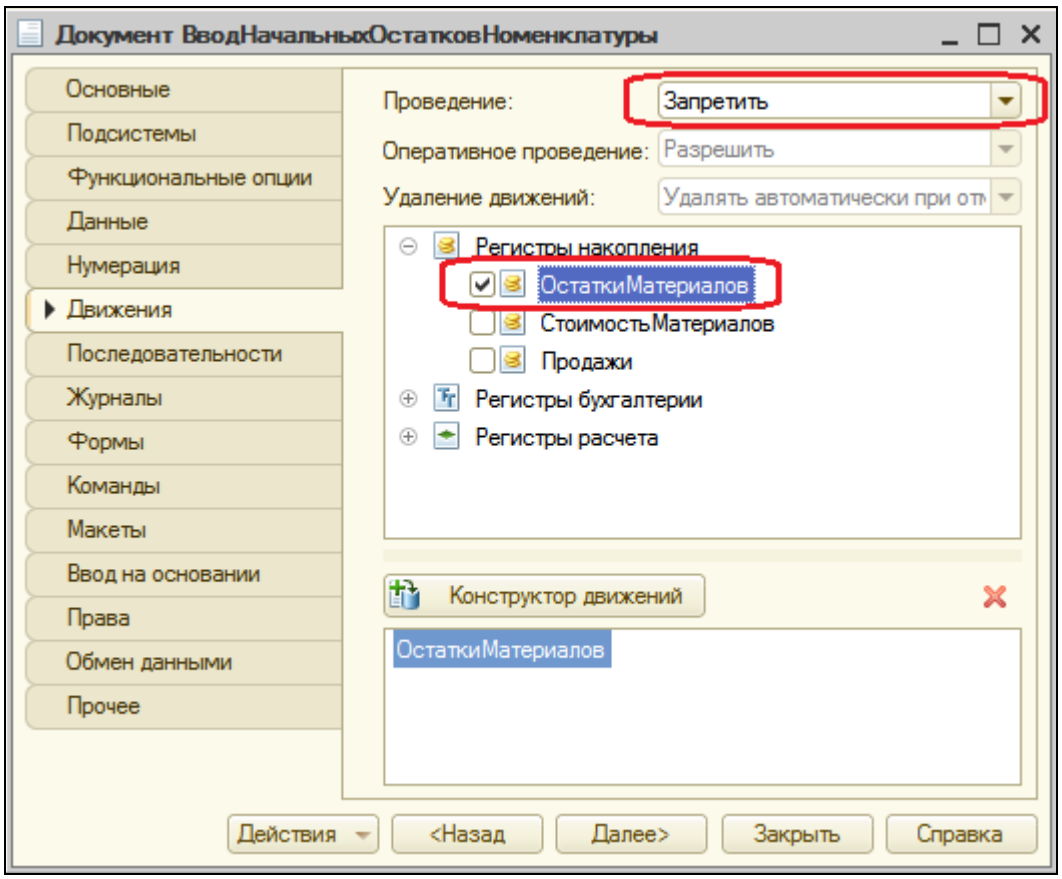
Рассмотрим пример ввода начальных остатков регистра накопления **ОстаткиМатериалов**.

Для выполнения этой задачи мы создадим документ, в котором будем вручную редактировать его движения по регистру **ОстаткиМатериалов** прямо в форме документа.

В режиме Конфигуратор

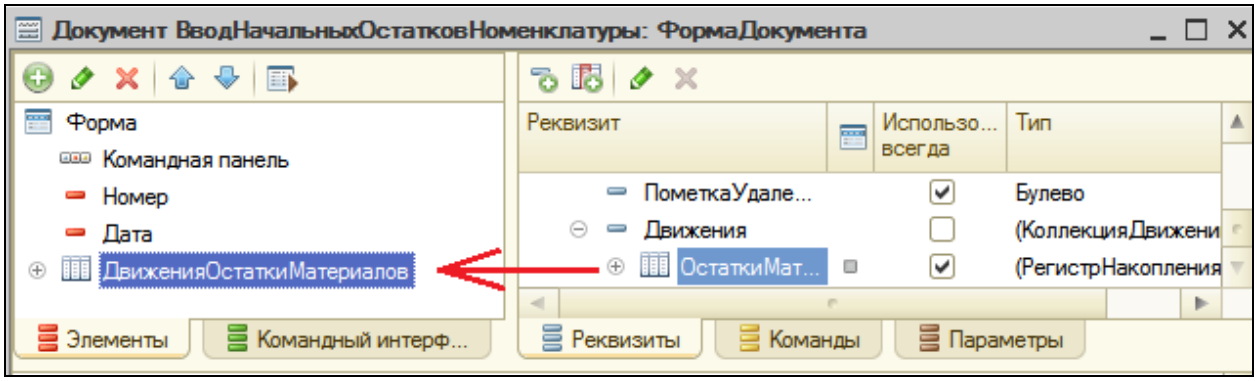
Создадим новый объект документ с именем **ВводНачальныхОстатковНоменклатуры**.

На закладке **Движения** запретим проведение документа (поскольку сами будем формировать записи регистра) и отметим, что движения документа будут находиться в регистре накопления **ОстаткиМатериалов**.



После этого перейдем на закладку **Формы** и создадим основную форму документа.

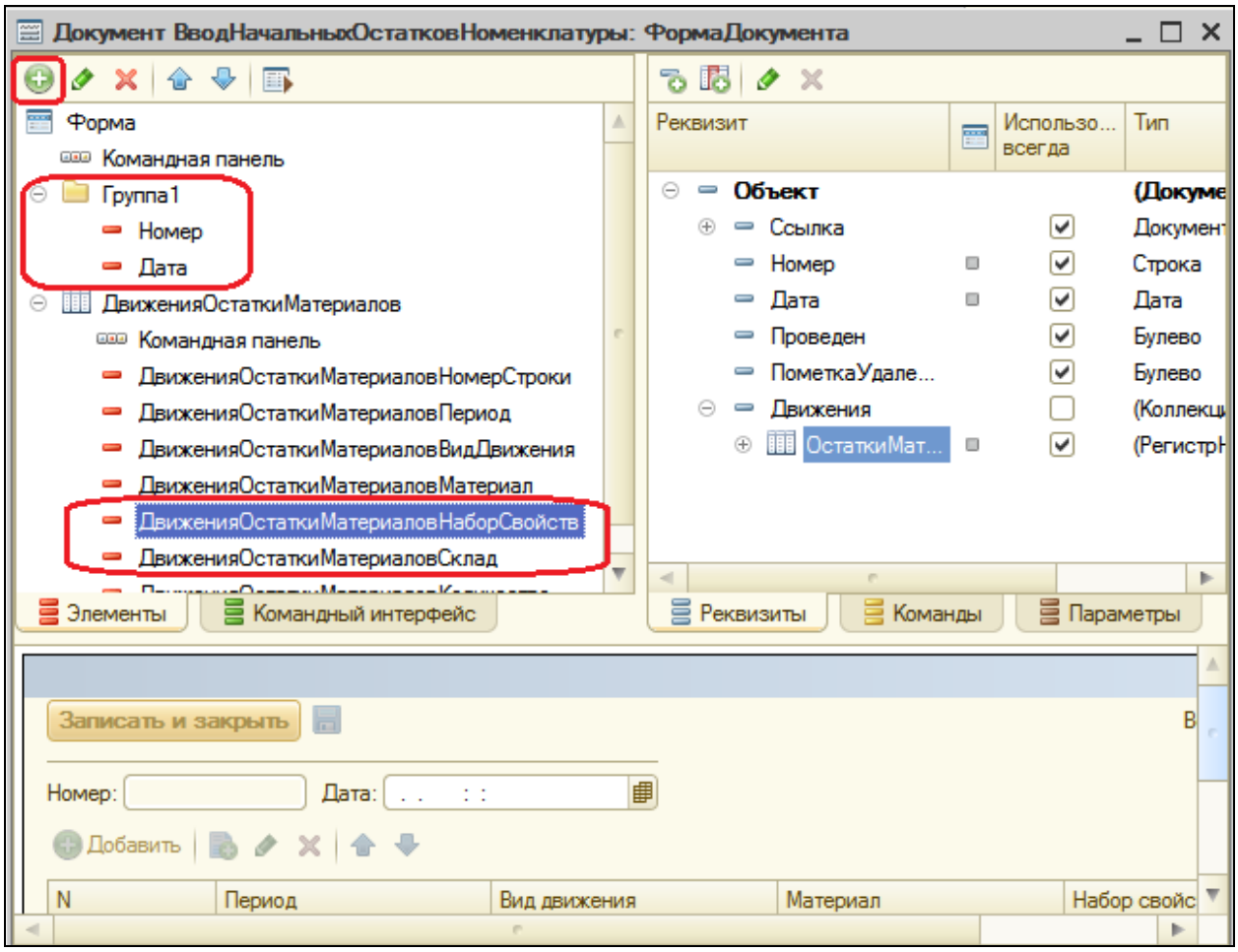
В окне редактора форм на закладке **Реквизиты** раскроем основной реквизит формы **Объект**, затем раскроем **Движения**, найдем строку **ОстаткиМатериалов** и перетащим ее в окно элементов формы. На вопрос системы «добавить колонки таблицы?» ответим «да».



Немного изменим внешний вид формы.

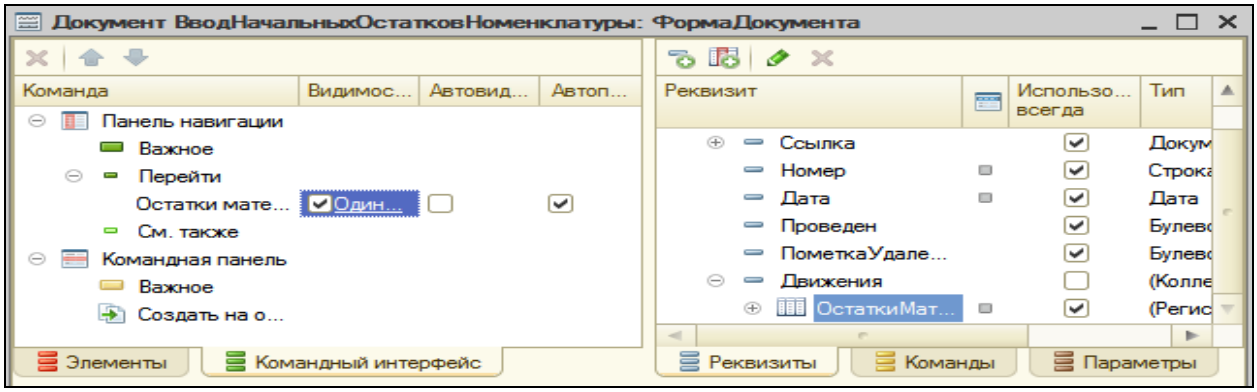
В окне элементов формы добавим группу полей с типом группировки **Горизонтальная** и перетащим в нее поля документа **Номер** и **Дата**. А

также поменяем местами поля таблицы **ДвиженияОстаткиМатериаловНаборСвойств** и **Склад**.



В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра **ОстаткиМатериалов**, связанному с документом. Для этого в левом верхнем окне редактора форм перейдем на вкладку **Командный интерфейс**.

В группе **Панель навигации** в подгруппе **Перейти** установим видимость для команды открытия регистра **Остатки материалов**.



В окне редактирования документа **ВводНачальныхОстатковНоменклатуры** установим принадлежность к подсистеме **Бухгалтерия**.

В заключение отредактируем командный интерфейс этой подсистемы – **Общие – Подсистемы – Все подсистемы.**

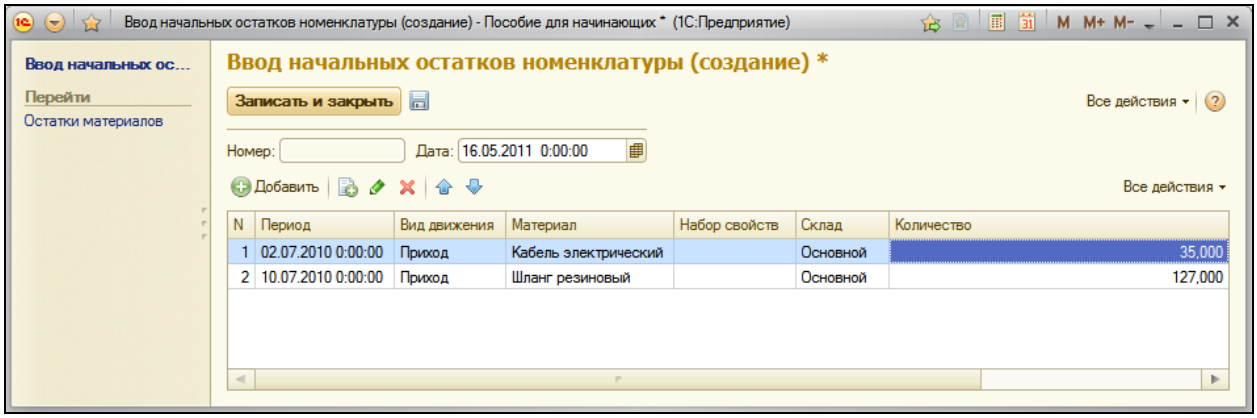
Выделим в списке подсистем **Бухгалтерию** и в списке команд установим видимость команды **Ввод начальных остатков номенклатуры: создать** в группе **Панель действий.Создать**.

В режиме 1С:Предприятие

Запустим режим отладки и проверим работу нашего документа.

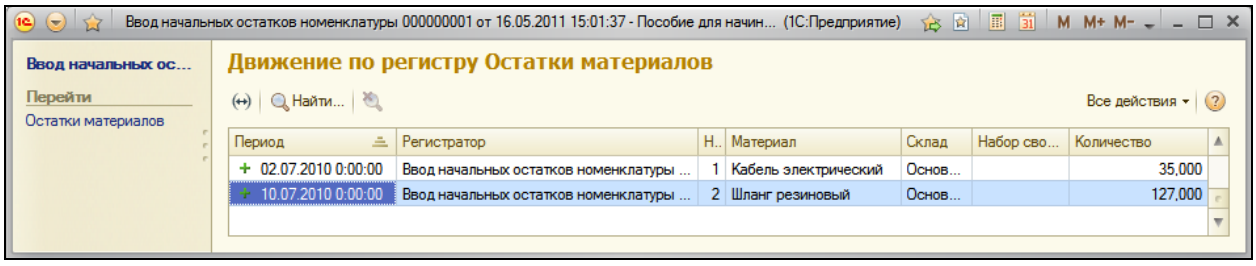
Выполним команду **Ввод начальных остатков номенклатуры** в панели действий раздела **Бухгалтерия**.

Создадим документ для ввода начальных остатков в регистр **ОстаткиМатериалов** и внесем в него следующие данные.



Обратите внимание, что дата документа не совпадает с датами отдельных записей, которые мы создаем в движениях документа.

Нажмем **Записать** и в панели навигации перейдем к движениям нашего документа в регистре **ОстаткиМатериалов**.



Таким образом, мы добились поставленной цели: с одной стороны, задавая дату документа, мы можем фиксировать момент внесения изменений в записи регистра, с другой стороны – для каждой создаваемой нами записи регистра мы можем указать индивидуальное значение поля **Период.**

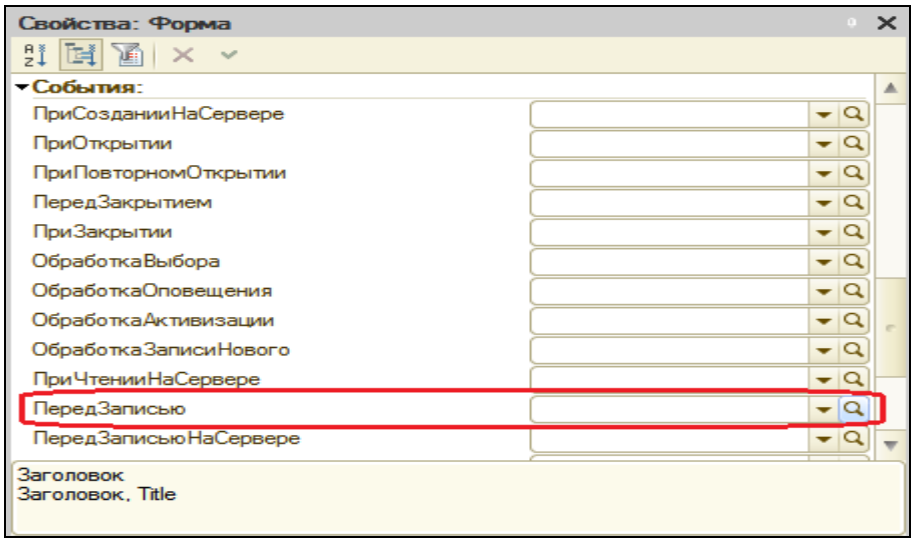
Теперь займемся ужесточением требований к тому, как наш документ формирует записи регистра, и рассмотрим два типичных варианта.

Программное редактирование записей регистра

В режиме Конфигуратор

Первое требование будет заключаться в том, что записи регистра должны формироваться той же датой, что и дата документа. Иначе говоря, синхронизируем дату движений с датой документа.

Для этого создадим для формы документа обработчик события **ПередЗаписью** и добавим в него следующий текст.

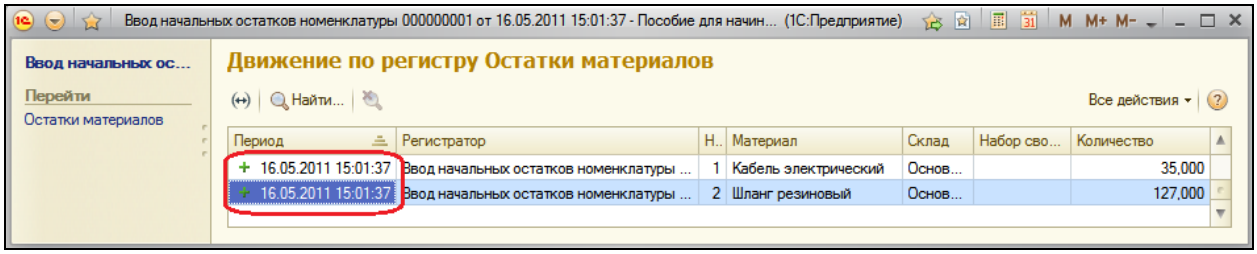



```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
        Для Каждого ЗаписьРегистра Из Объект.Движения.ОстаткиМатериалов
Цикл
        ЗаписьРегистра.Период = Объект.Дата;
КонецЦикла;

КонецПроцедуры
```

В режиме 1С:Предприятие

Запустим отладку, откроем наш документ и нажмем **Записать**.

Открыв движения документа в регистре **ОстаткиМатериалов**, увидим, что значение поля **Период** у всех записей стало равно дате документа.



Можно сказать, что мы достигли поставленной цели, но лишь для интерактивной записи документа. Если программно вызвать метод **Записать()** у объекта нашего документа, он будет записан без участия формы документа. Это значит, что событие **ПередЗаписью** формы документа вызвано не будет и наш код обработчика не сработает.

Чтобы предусмотреть возможность синхронизации периода движений документа с датой документа и в случае программной записи объекта **Документ**, следует использовать обработчик события **ПередЗаписью** объекта **Документ**, а не формы документа. Событие **ПередЗаписью** в случае интерактивной записи документа сначала будет вызвано у формы документа, а затем у объекта **Документ**.

В режиме Конфигуратор

Вернемся в конфигуратор и удалим из модуля формы добавленный нами текст и создадим обработчик события **ПередЗаписью** в модуле документа **ВводНачальныхОстатковНоменклатуры**.

Для этого откроем на закладке **Прочее** окна редактирования этого объекта модуль объекта и внесем в него следующий текст.

```
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

	// Определить, нужно ли обновлять дату в движениях
	ОбновитьДатуДвижений = ЭтоНовый() Или
Движения.ОстаткиМатериалов.Модифицированность();
	Если Не ОбновитьДатуДвижений Тогда
		// Проверить, что дата изменилась
		Запрос = Новый Запрос;
		Запрос.УстановитьПараметр("ТекущийДокумент", Ссылка);
		Запрос.Текст =
		"ВЫБРАТЬ
		| Дата
		|ИЗ
```

```
            | Документ.ВводНачальныхОстатковНоменклатуры
            |ГДЕ Ссылка = &ТекущийДокумент";

            Выборка = Запрос.Выполнить().Выбрать();
            Выборка.Следующий();
            ОбновитьДатуДвижений = Выборка.Дата <> Дата;
        КонецЕсли;

        // Установить всем новую дату, если нужно
        Если ОбновитьДатуДвижений Тогда
            Если Не Движения.ОстаткиМатериалов.Выбран() И
            Не Движения.ОстаткиМатериалов.Модифицированность() Тогда
                Движения.ОстаткиМатериалов.Прочитать();
            КонецЕсли;
            Для Каждого ЗаписьРегистра Из Движения.ОстаткиМатериалов Цикл
                ЗаписьРегистра.Период = Дата;
            КонецЦикла;
        КонецЕсли;

КонецПроцедуры
```

Как вы видите, в этом случае обработчик содержит больше кода за счет дополнительных проверок, которые выполняются в результате возможности как интерактивной, так и программной записи объекта.

Поясним содержание обработчика. Если записывается новый документ или были изменены его движения, следует обновить дату движений. В противном случае мы считваем запросом дату документа из базы и сравниваем ее с датой, установленной у записываемого объекта. Если даты разные, также следует обновить дату движений.

Перед установкой даты мы проверяем, был ли прочитан набор записей в свойстве **Движения** объекта и изменился ли он. Если оба этих условия ложны, значит набор записей в свойстве **Движения** объекта пуст, и это состояние не связано с его изменением. В этом случае, чтобы предотвратить ошибочное удаление записей в регистре (перезаписать пустым набором записей), мы предварительно читаем движения из регистра в набор записей в свойстве **Движения**.

Затем, как и в предыдущем случае, устанавливаем нужную дату для всех записей этого набора. При выполнении записи объекта документ этот набор будет записан в регистр накопления.

В режиме 1С:Предприятие

Запустим режим отладки и убедимся, что указав новую дату для нашего документа и записав его, мы получим движения в регистре накопления с новой датой.

В процессе записи нашего документа можно управлять не только периодом записей регистра накопления, но и значениями других полей регистра.

Например, по аналогичному принципу может быть создан документ **Операция**, позволяющий вводить ручные операции в регистр бухгалтерии.

При этом вероятно, что кроме управления периодом записей регистра вам потребуется управлять значением поля **Активность** (включать и выключать проводки документа) и т.д.

Где создавать обработчики событий

Выбор обработчика, в который будет помещен текст процедуры, зависит от логики работы создаваемого объекта. Если конфигурация не предусматривает прогаммной записи объекта, можно выбрать обработчик модуля формы. Если предполагается и программная модификация объекта, следует выбирать обработчик модуля объекта.

Заметьте, что оба этих способа не исключают модификацию записей регистра через объект **Регистр<…>НаборЗаписей.<имя регистра>.** Поэтому если логика конфигурации подразумевает возможность программной модификации объекта НаборЗаписей, код обработки следует помещать в обработчик события набора записей. Все попытки изменить данные регистра будут сведены к записи именно набора записей.

Контрольные вопросы

- ✓ Для чего предназначен документ для ввода начальных остатков и как его создать.
- ✓ Как программно изменить значение регистра при вводе начальных остатков.
- ✓ В каких случаях использовать модуль формы, а в каких – модуль объекта для размещения обработчиков событий.

Практическая работа № 21

Список пользователей и их роли (1:00)

Что такое Роль

После создания всех основных объектов конфигурации, можно приступить к определению ролей пользователей.

Очевидно, что не всем пользователям нужно разрешать просматривать и изменять всю базу данных. Кладовщику нельзя просматривать и изменять данные о сотрудниках и бухгалтерии, например.

Для описания подобных разрешений используются объекты конфигурации *Роль*.

Как правило, роли создаются отдельно для каждого вида деятельности и каждому пользователю системы ставится в соответствие одна или несколько ролей. Если пользователю поставлено в соответствие несколько ролей, то предоставление доступа будет осуществляться по следующему алгоритму:

- Если хотя бы в одной роли есть разрешение, то доступ будет открыт;
- Если во всех ролях разрешение отсутствует, то доступ будет закрыт.

Создание ролей

В режиме Конфигуратор

При создании ролей исходят, как правило, из того, какие полномочия требуются различным группам пользователей на доступ к информации. Для этого мы воспользуемся подсистемами, которые значительно облегчат нашу задачу.

Администратор

Первая роль, которую мы создадим, будет **Администратор**. Она должна включать в себя полные права на работу с данными информационной базы.

Раскроем ветвь **Общие** дерева объектов конфигурации. Добавим новый объект **Роль** с именем **Администратор**.

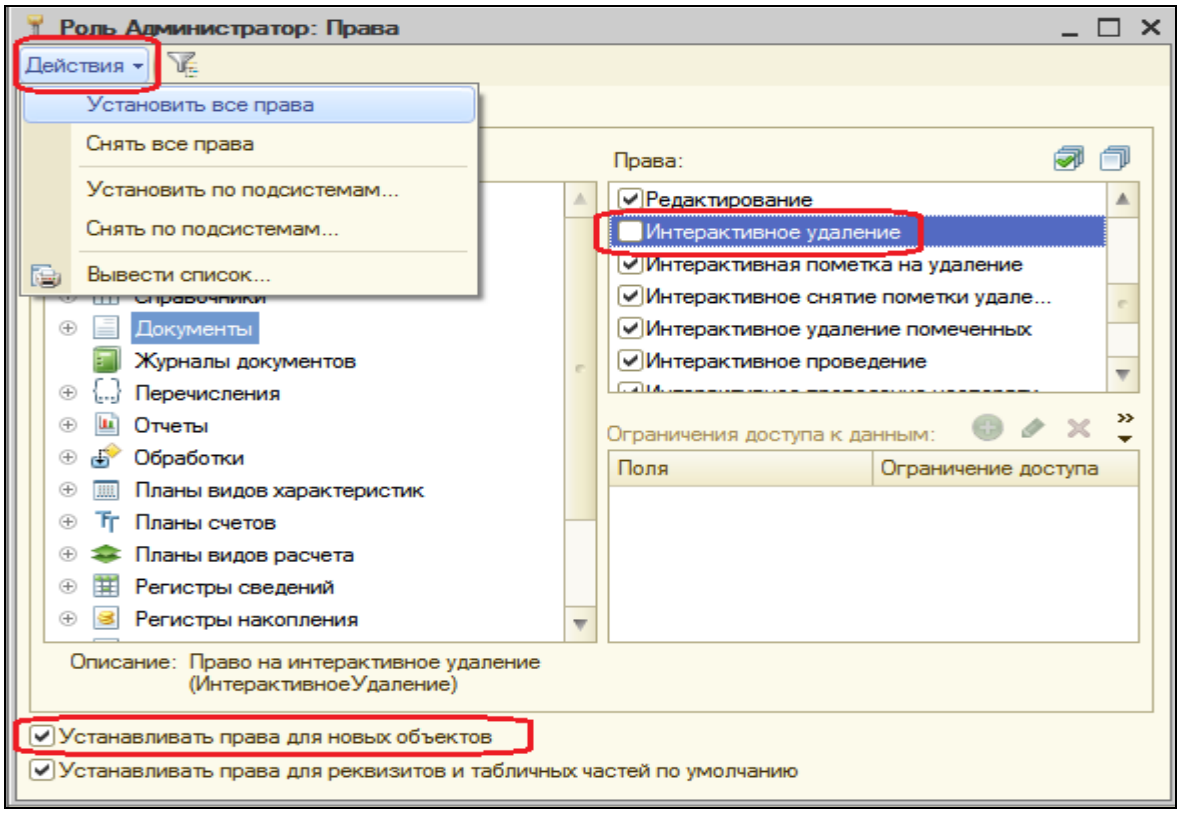
Откроется окно редактирования этой роли. Слева, в списке объектов, перечислены все объекты и виды объектов конфигурации, а справа, в окне прав, - доступные права для выбранного объекта или видов объектов конфигурации.

Администратор должен иметь права на все объекты и все виды объектов. Для этого выполним команду **Действия – Установить все права** в командной панели окна. После этого все права для всех объектов будут помечены.

Снимите разрешение на интерактивное удаление для всех объектов. Это нужно, чтобы администратор случайно не мог удалить какой-либо объект базы данных.

Для этого пройдемся по всем видам объектов конфигурации (Справочники, Документы и т.д.) и снимем отметку с команды **Интерактивное удаление**.

Для того, чтобы наш **Администратор** мог работать с объектами, которые мы будем создавать после расстановки прав, зададим для него параметр **Устанавливать права для новых объектов**. На этом создание роли Администратор закончено.

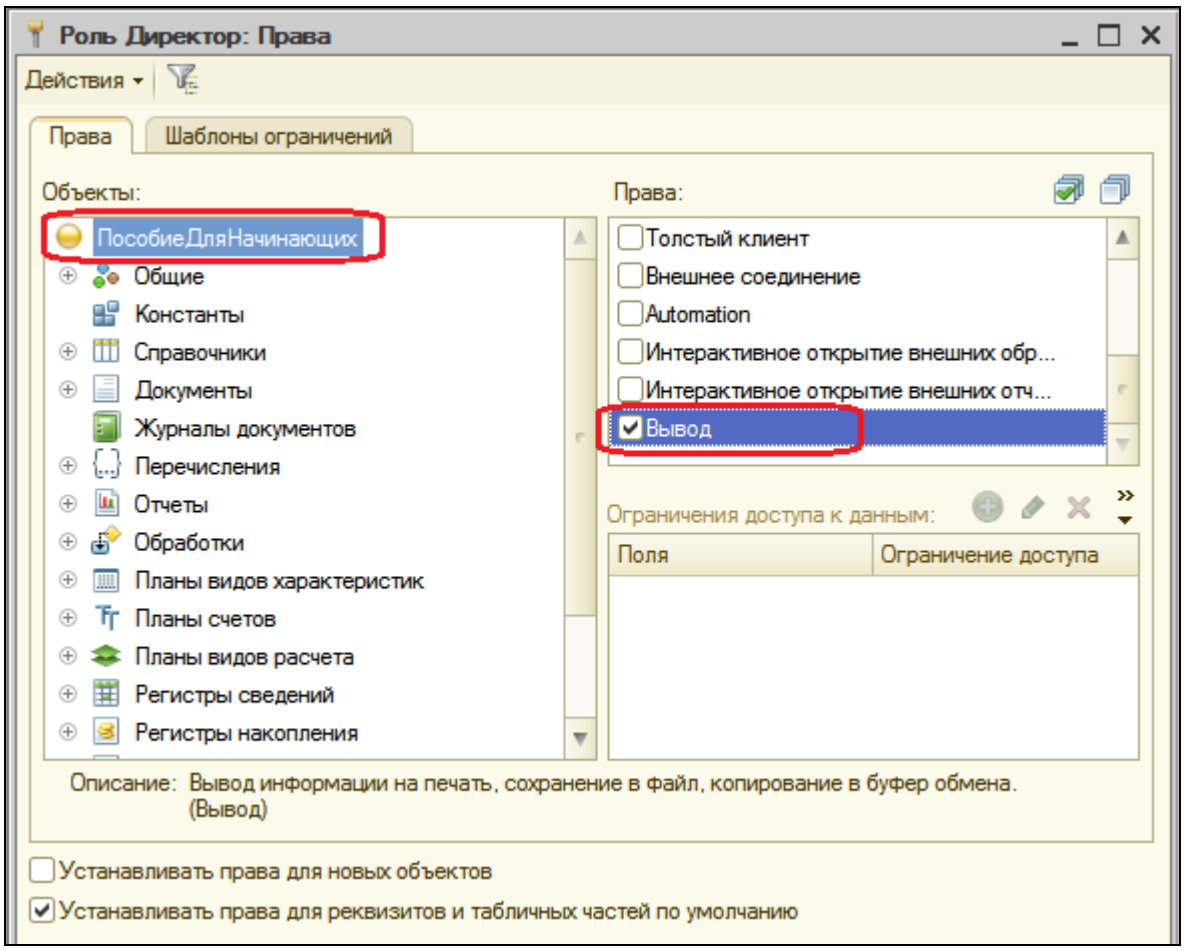


Директор

Создадим новый объект конфигурации **Роль** с именем **Директор**.

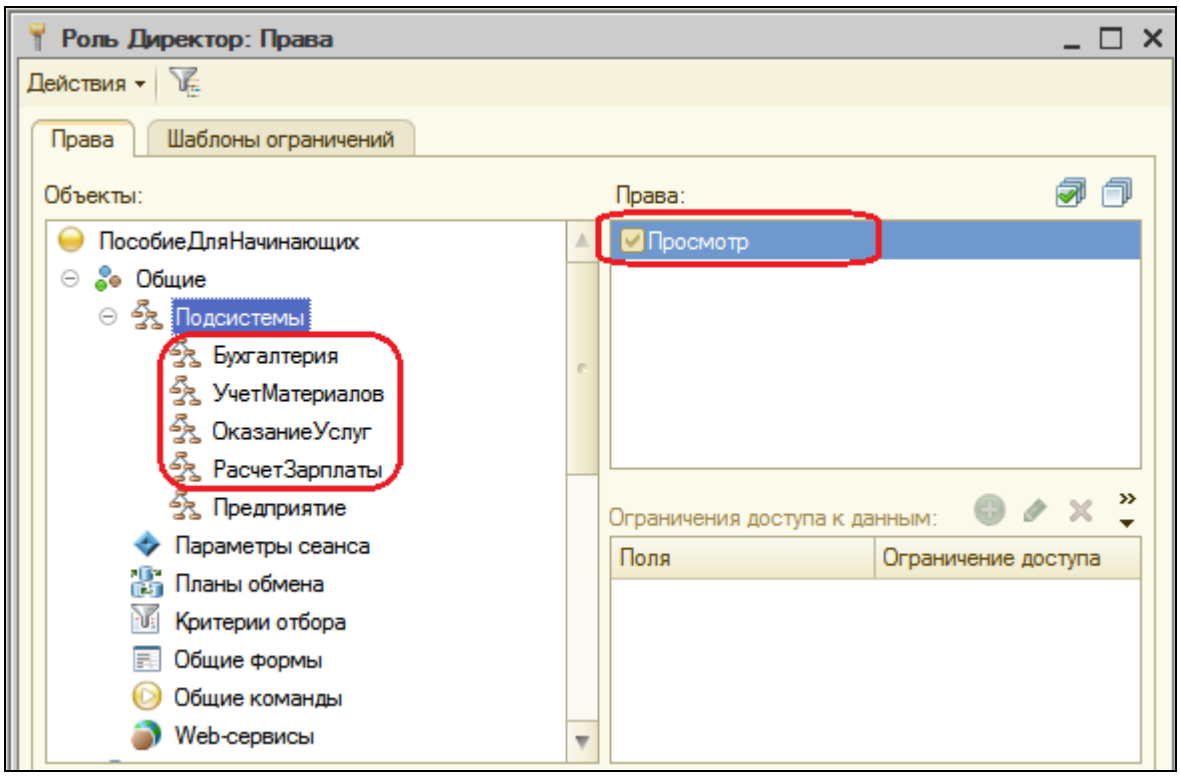
Нас устраивает, что у новой роли нет прав на доступ ко всем объектам, за исключением тех видов объектов, для которых не создано ни одного объекта. Для таких видов останутся установленными полные права.

Установим право **Вывод** для всей конфигурации.



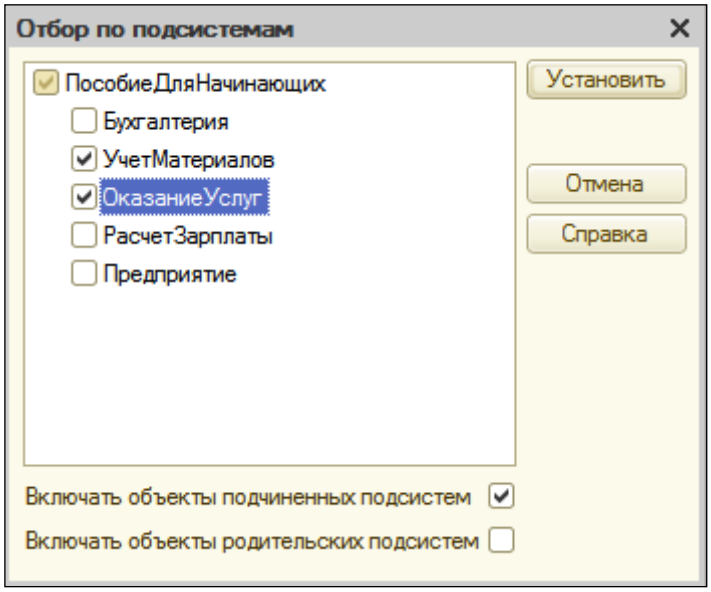
Теперь остается пройти по всем видам объектов конфигурации и установить для них право **Просмотр** (права **Чтение** и **Использование** при этом установятся автоматически).

Затем раскроем ветвь **Общие**, выделим ветвь **Подсистемы** и отметим право **Просмотр** у всех подсистем, кроме подсистемы **Предприятие**. Тем самым мы предоставим директору возможность просматривать все данные информационной базы, но исключим из его интерфейса все действия, которые по логике нашей конфигурации не относятся к прикладной ее части.



Мастер

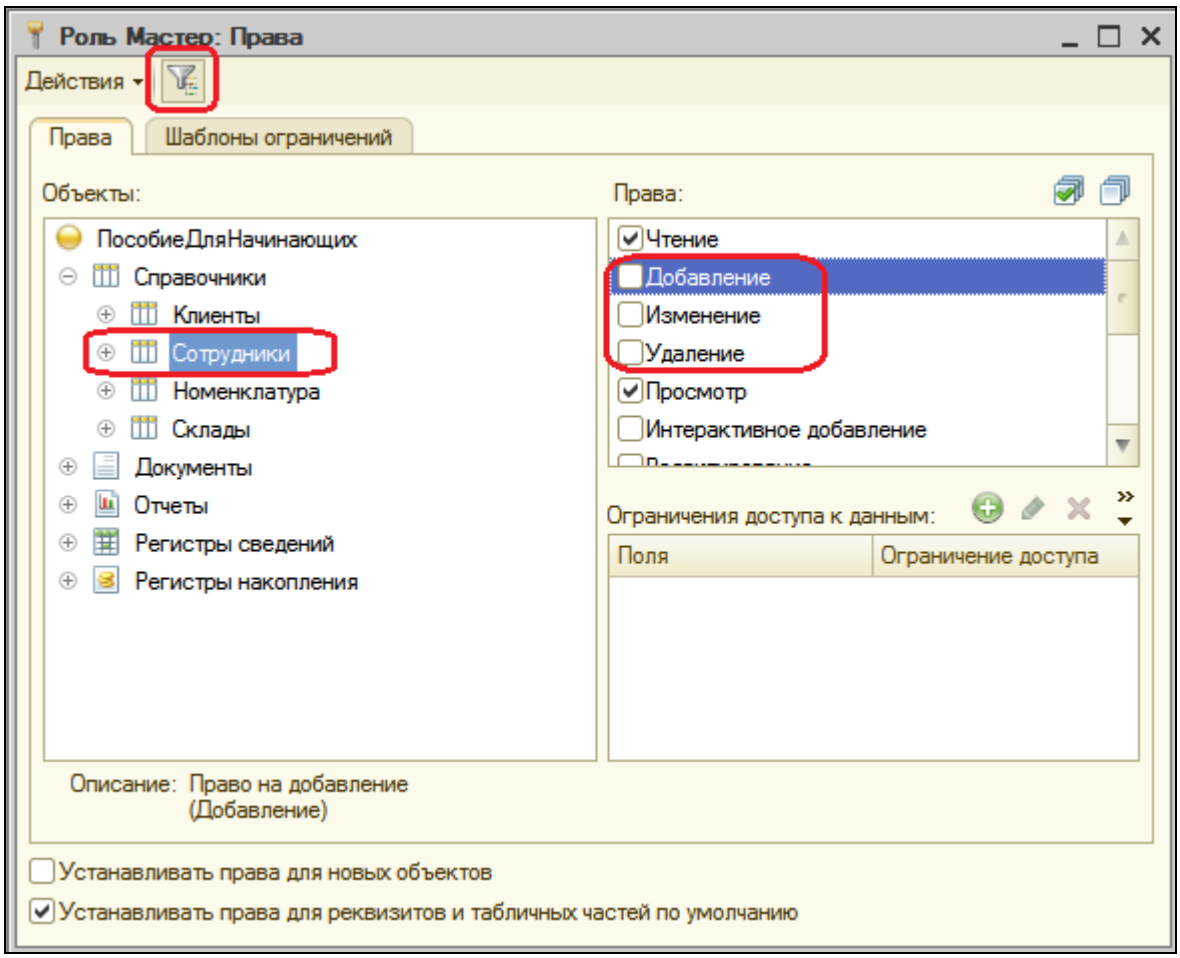
Снова добавим роль с именем **Мастер**. Выполним команду **Действия – Установить по подсистемам** и выберем подсистемы **УчетМатериалов** и **ОказаниеУслуг**.



В результате будут установлены все права на объекты конфигурации, относящиеся к данным подсистемам.

Если теперь установить фильтр объектов по подсистемам **УчетМатериалов** и **ОказаниеУслуг**, то можно внести уточнения в установленные права.

В частности для справочника **Сотрудники** мы запретим права **Добавление**, **Изменение**, **Удаление**.



Обратите внимание, что при запрете права **Добавление** исчезла отметка и у права **Интерактивное добавление**, т.к. оно является уточнением права **Добавление**. Точно также уточненные права запрещаются и при отмене прав на изменение и удаление.

Теперь пройдем по всем видам объектов и снимем у всех право **Интерактивное удаление.**

Затем снимем фильтр и установим все права, кроме интерактивного удаления для следующих объектов:

- Справочник **ВариантыНоменклатуры**,
- Справочник **ДополнительныеСвойстваНоменклатуры**,
- План видов характеристик **СвойстваНоменклатуры**,
- Регистр сведений З**наченияСвойствНоменклатуры**.

Эти объекты мы не привязывали ни к каким подсистемам, но они будут нужны для работы с характеристиками номенклатуры.

Расчетчик

В заключение нам осталось создать две роли: **Бухгалтер** и **Расчетчик**.

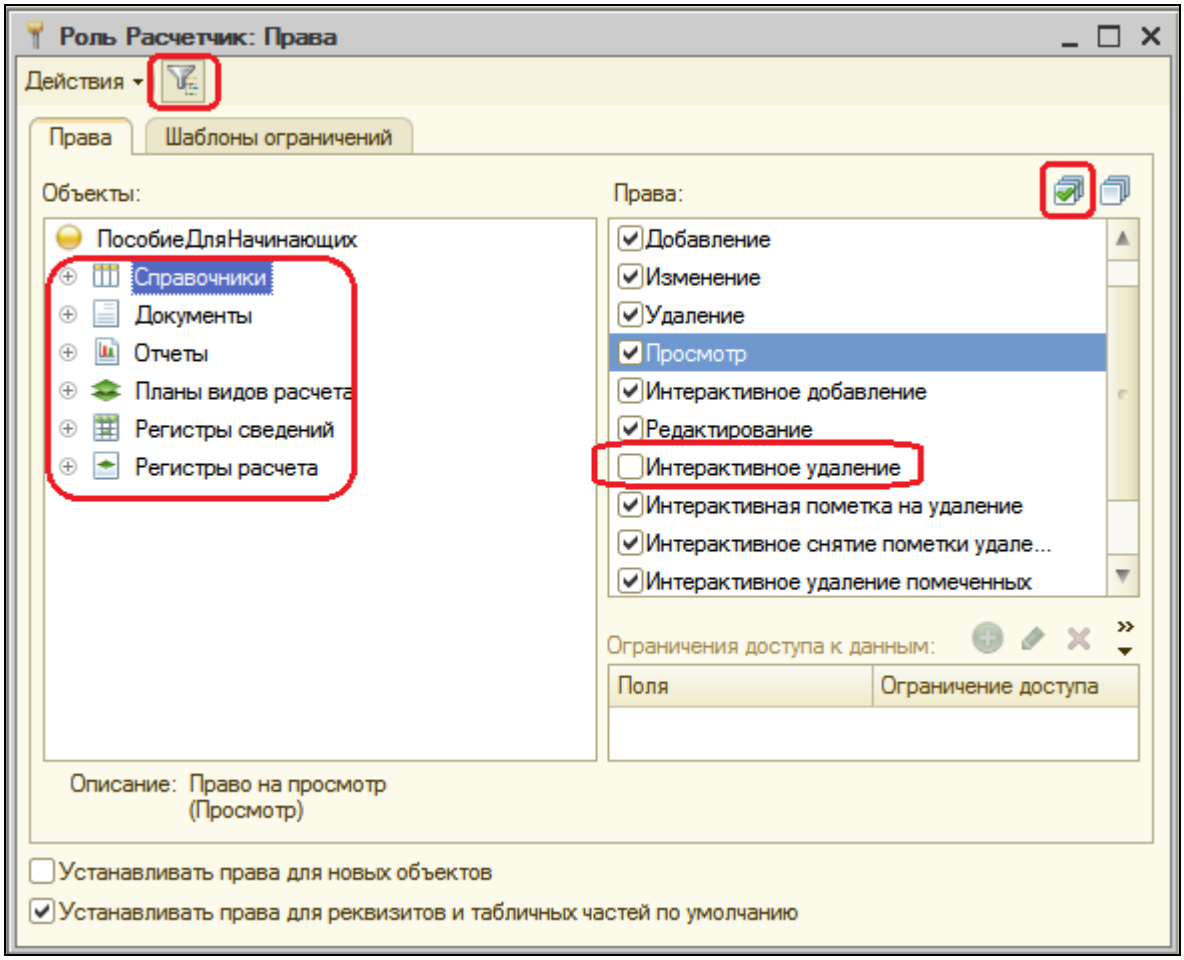
Мы разделим права по расчету зарплаты и по ведению бухгалтерского учета.

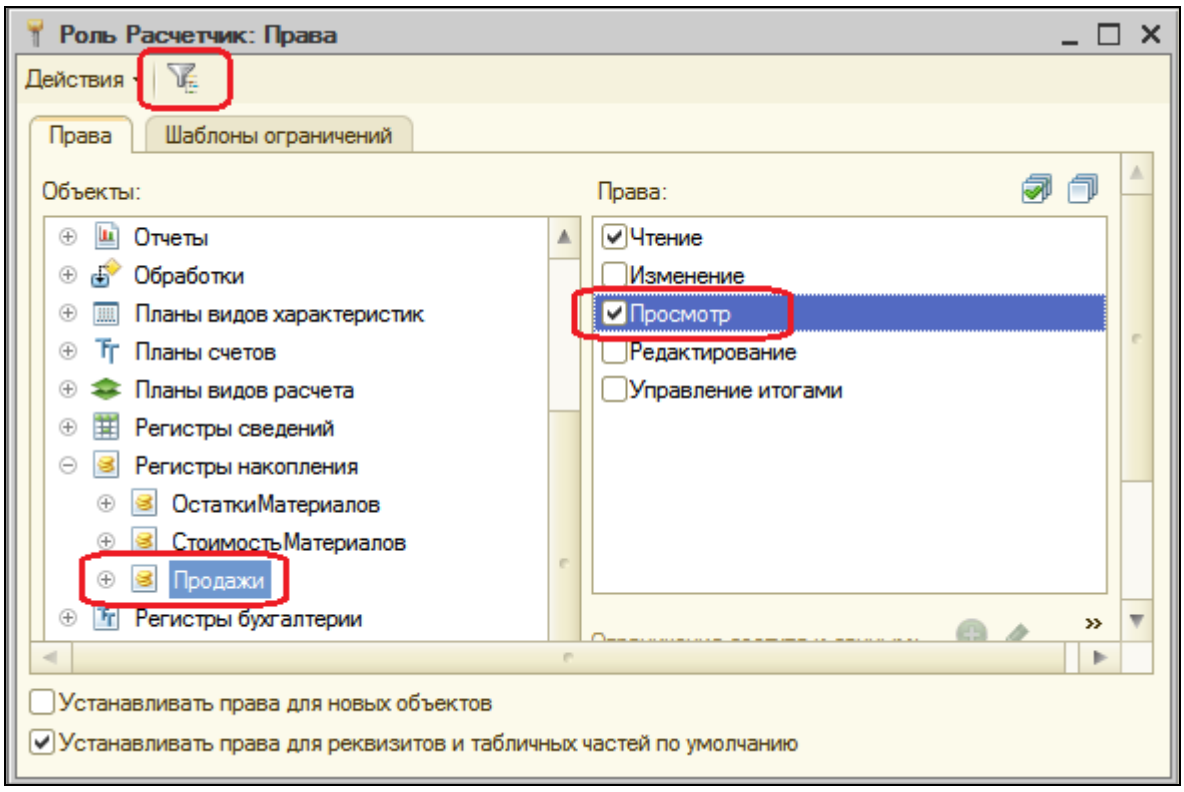
Дело в том, что в нашей фирме есть бухгалтер и его помощник. Помощник бухгалтера будет занят в основном расчетом зарплаты, но иногда это делает и главный бухгалтер.

Поэтому главному бухгалтеру необходимо будет назначить обе роли, а помощнику – только роль **Расчетчик**.

Создадим новую роль **Расчетчик**. В окне редактирования прав установим все права по подсистеме **РасчетЗарплаты,** пользуясь фильтром (и не забудем запретить интерактивное удаление).

А также установим право **Просмотр** для регистра накопления **Продажи,** отключив фильтр.

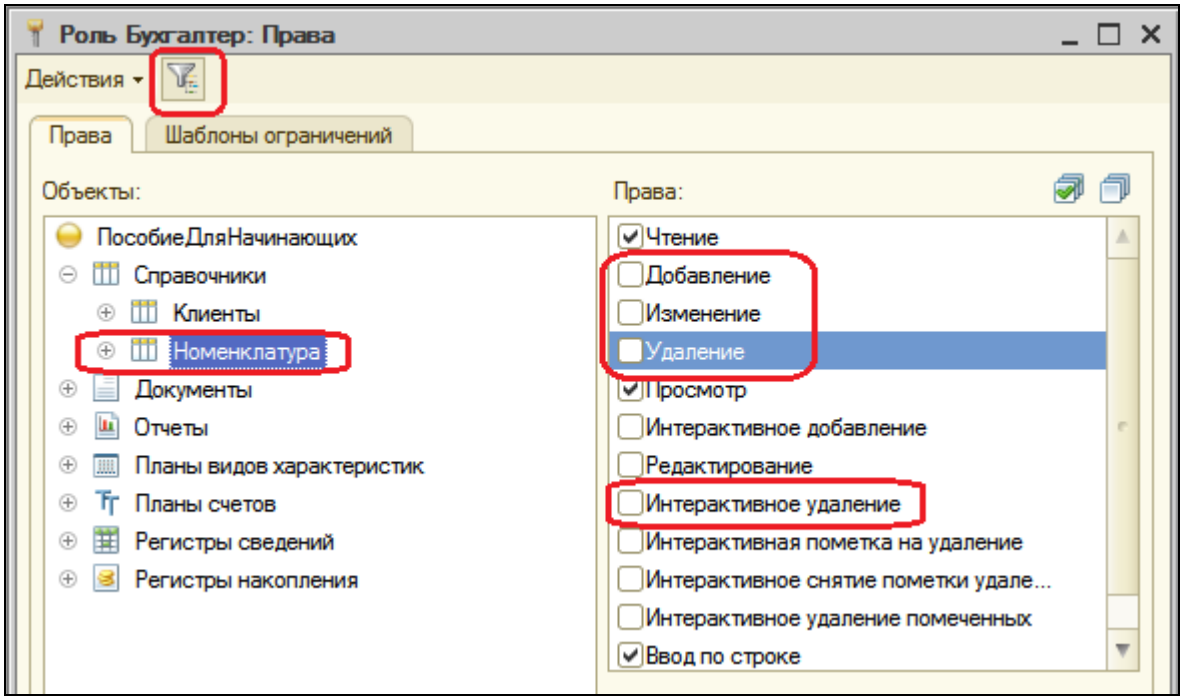




Бухгалтер

Создадим роль с именем **Бухгалтер**. В окне редактирования прав установим их по подсистеме **Бухгалтерия**. После этого отфильтруем список объектов по этой подсистеме и для справочника **Номенклатура** запретим добавление, изменение и удаление.

Также запретим **интерактивное удаление** для всех объектов.



Затем снимем фильтр и установим все права, кроме интерактивного удаления для следующих объектов:

- Справочник **Субконто**,
- Регистр бухгалтерии **Управленческий**.

Установим право **Просмотр** для следующих объектов:

- Справочник **Склады**,
- Справочник **ВариантыНоменклатуры**,
- Справочник **ДополнительныеСвойстваНоменклатуры**,
- План видов характеристик **СвойстваНоменклатуры**,
- Регистр сведений **ЗначенияСвойствНоменклатуры**.

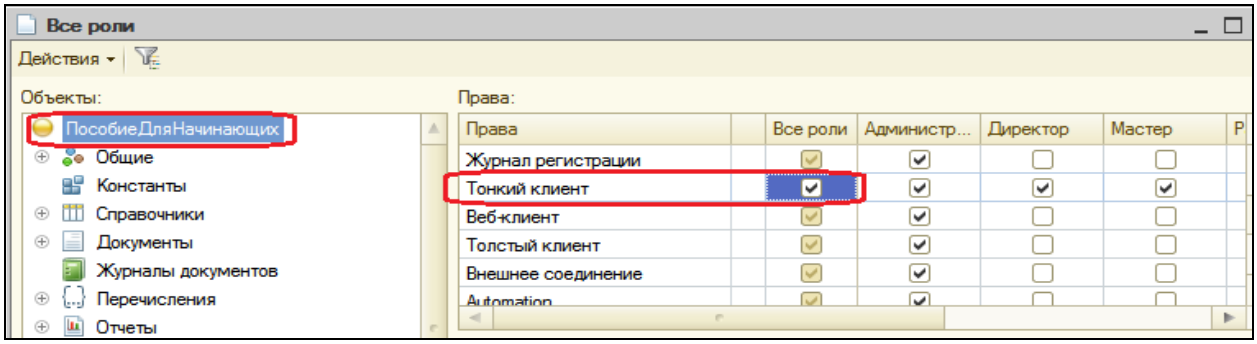
Права на запуск клиентских приложений

В заключение установим права на запуск клиентского приложения для каждой роли. Для этого воспользуемся другим, более удобным инструментом – редактором **Все роли**.

В дереве объектов конфигурации выделим ветку **Роли** и в контекстном меню выполним команду **Все роли**.

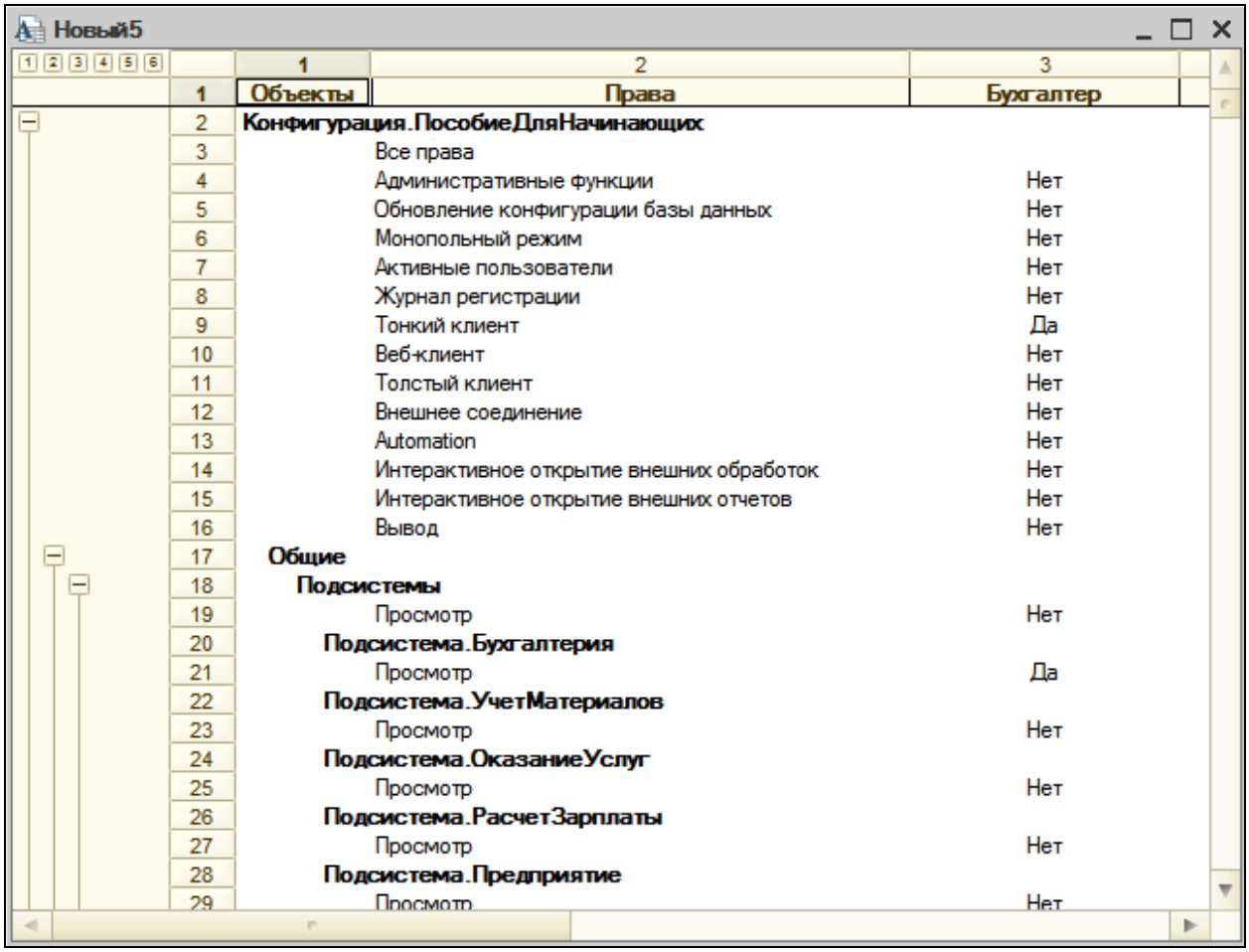
В списке объектов конфигурации выделим корень и для всех ролей установим право **Тонкий клиент**.

Тем самым мы разрешили всем пользователям подключаться к информационной базе с помощью тонкого клиента. Администратор же имеет возможность подключаться и с помощью других клиентских приложений. Это может понадобиться ему, например, для создания планировщика заданий, о котором шла речь в разделе «Планировщик заданий».



Список прав для каждой роли можно получить, выполнив в окне редактирования прав команду **Действия – Вывести список**. Аналогичный список, но только для всех ролей, которые есть в конфигурации, можно получить из редактора **Все роли**.

Это пример списка прав для роли **Бухгалтер**.



	1	2	3
1	Объекты	Права	Бухгалтер
2	**Конфигурация.ПособиеДляНачинающих**		
3		Все права	
4		Административные функции	Нет
5		Обновление конфигурации базы данных	Нет
6		Монопольный режим	Нет
7		Активные пользователи	Нет
8		Журнал регистрации	Нет
9		Тонкий клиент	Да
10		Веб-клиент	Нет
11		Толстый клиент	Нет
12		Внешнее соединение	Нет
13		Automation	Нет
14		Интерактивное открытие внешних обработок	Нет
15		Интерактивное открытие внешних отчетов	Нет
16		Вывод	Нет
17	**Общие**		
18	**Подсистемы**		
19		Просмотр	Нет
20	**Подсистема.Бухгалтерия**		
21		Просмотр	Да
22	**Подсистема.УчетМатериалов**		
23		Просмотр	Нет
24	**Подсистема.ОказаниеУслуг**		
25		Просмотр	Нет
26	**Подсистема.РасчетЗарплаты**		
27		Просмотр	Нет
28	**Подсистема.Предприятие**		
29		Просмотр	Нет

Добавление новых пользователей

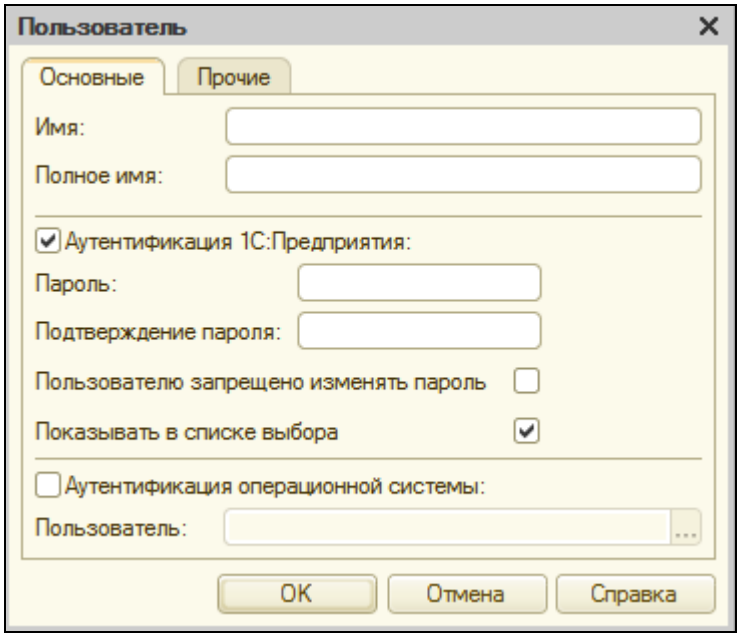
В режиме Конфигуратор

Прежде чем мы приступим к созданию пользователей, необходимо выполнить обновление конфигурации базы данных (**Конфигурация – обновить конфигурацию базы данных (F7)** ) поскольку пользователю можно поставить в соответствие только те роли, которые существуют в конфигурации.

После обновления, выполним команду главного меню **Администрирование – Пользователи**. Откроется список пользователей системы. Пока что он пуст, поэтому добавим нового пользователя (**Действия – Добавить**) или нажмем кнопку **Добавить**.

Внимание!

Если вы используете учебную версию платформы, то возможность задания паролей пользователей и аутентификация операционной системы будут недоступны. Это ограничения учебной версии.



Имя пользователя – это идентификатор, который будет появляться в окне выбора пользователей при запуске системы в режиме 1С:Предприятие.

Полное имя – строка, которая может быть использована внутри конфигурации при выводе различной справочной информации. Хорошим стилем администрирования считается указание в качестве полного имени фамилии, имени и отчества пользователя (без сокращений).

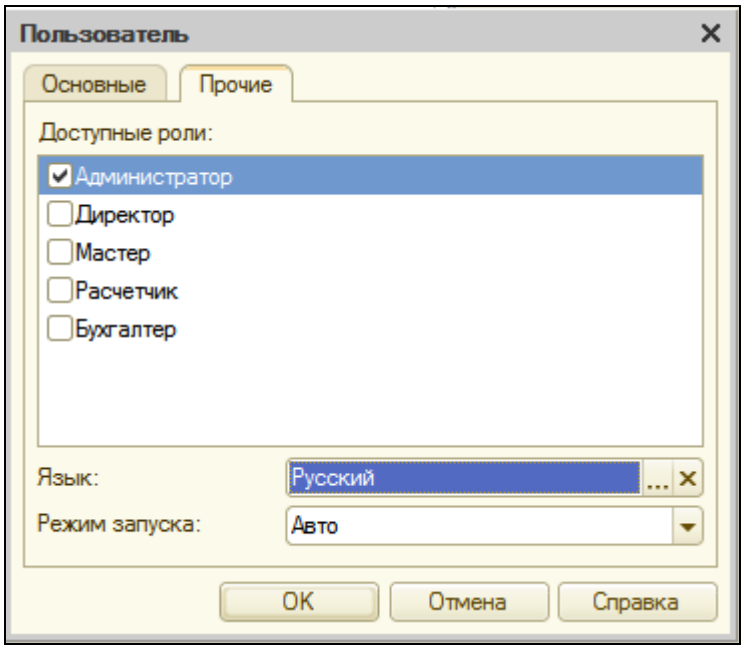
Аутентификация средствами 1С:Предприятия подразумевает, что после запуска системы пользователю будет предложено выбрать имя пользователя и ввести пароль. Если введенный пароль соответствует сохраненному в системе для этого идентификатора пользователя, система открывается с правами этого пользователя. При этом он может поменять пароль, если флажок **Пользователю запрещено изменять пароль** не установлен.

Аутентификация операционной системы подразумевает, что при запуске системы 1С:Предприятие 8 от пользователя не требуется никакой дополнительной информации. Система определяет, под каким пользователем запущена операционная система и затем обращается к своему списку пользователей. Если она находит в нем пользователя с

именем текущего пользователя операционной системы, то информационная база открывается с правами этого пользователя.

Приступим к созданию пользователей. Зададим имя пользователя **Администратор**, полное имя тоже **Администратор**. Перейдем на закладку **Прочие**.

Отметим роль **Администратор** и язык конфигурации выберем **Русский**.

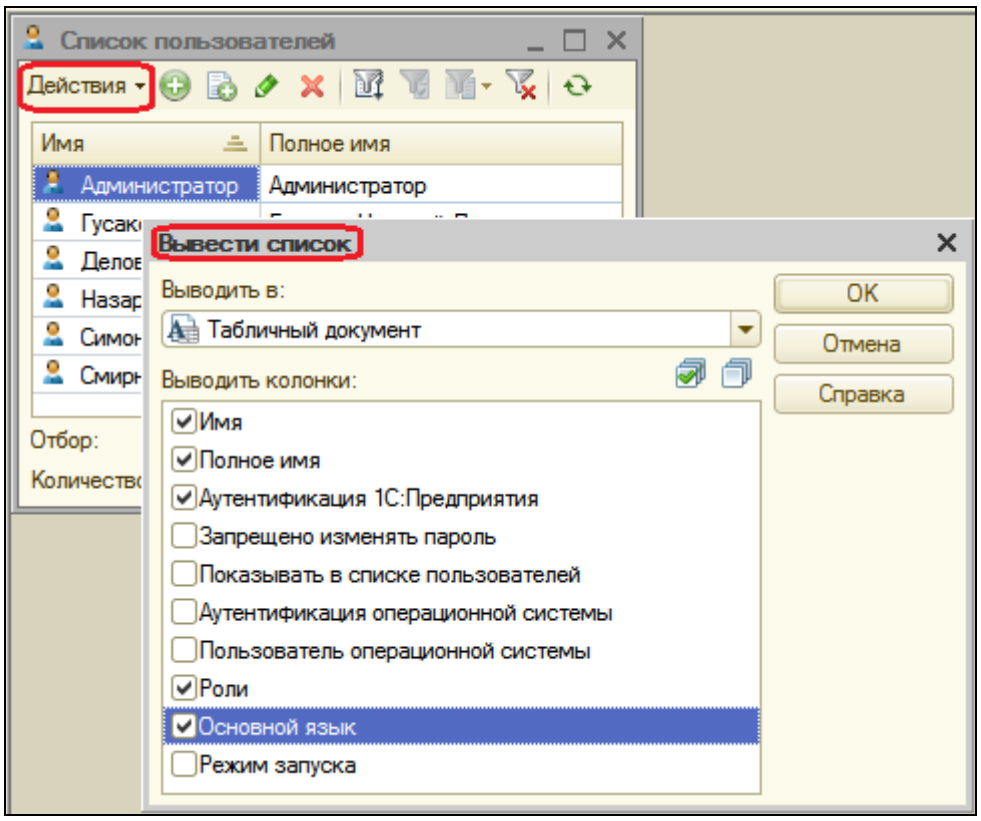


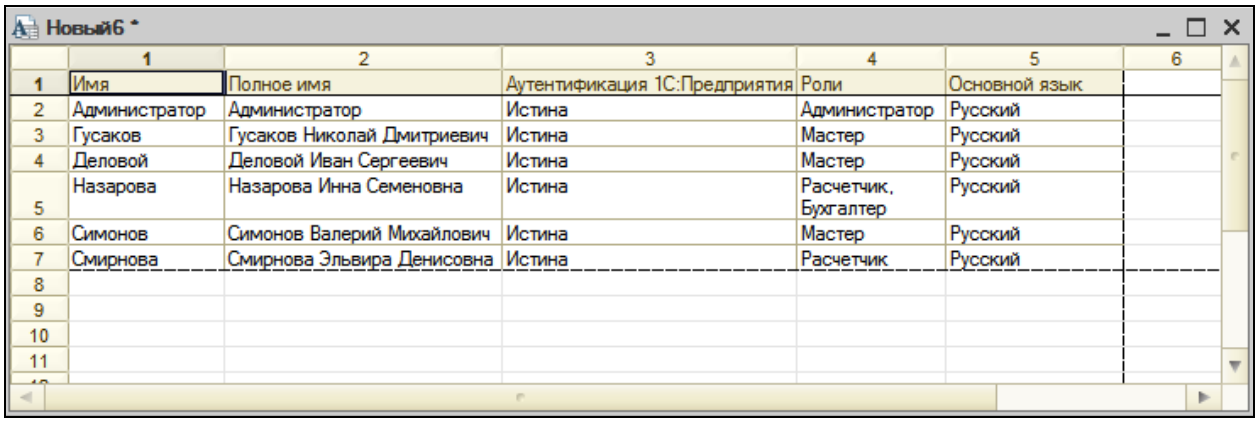
Нажмите ОК.

После этого создадим остальных пользователей системы согласно рисунку. Для всех них мы будем использовать аутентификацию 1С:Предприятия и русский язык.

Если некоторые колонки, например, **Роли**, не видны в списке пользователей, можно выполнить настройку списка **Действия – Настройка списка** и добавить нужные колонки (или при выводе списка сразу поставить нужные галочки).

Список пользователей можно получить, выполнив команду **Действия – Вывести список**.





Новый6 *

	1	2	3	4	5	6
1	Имя	Полное имя	Аутентификация 1С:Предприятия	Роли	Основной язык	
2	Администратор	Администратор	Истина	Администратор	Русский	
3	Гусаков	Гусаков Николай Дмитриевич	Истина	Мастер	Русский	
4	Деловой	Деловой Иван Сергеевич	Истина	Мастер	Русский	
5	Назарова	Назарова Инна Семеновна	Истина	Расчетчик, Бухгалтер	Русский	
6	Симонов	Симонов Валерий Михайлович	Истина	Мастер	Русский	
7	Смирнова	Смирнова Эльвира Денисовна	Истина	Расчетчик	Русский	
8						
9						
10						
11						

Ограничение доступа к данным на уровне записей и полей базы данных

Для более точного ограничения доступа к БД в системе 1С:Предприятие 8 используется *механизм ограничения доступа на уровне записей и полей базы данных*. Этот механизм позволяет для четырех основных прав (чтение, добавление, изменение, удаление) уточнить, какие же именно данные информационной базы будут доступны пользователю.

Такое уточнение записывается на специальном языке, являющемся подмножеством языка запросов.

Далее, на примере документа **Начисления сотрудникам**, мы рассмотрим небольшой пример, когда мастерам нужно дать возможность просмотреть начисленную им зарплату, но руководство запрещает им доступ к информации о начисленной премии.

Другими словами, мастерам нужно запретить просмотр тех документов **Начисления сотрудникам**, в которых есть записи о начислении премии.

В режиме Конфигуратор

Для решения этой задачи сначала установим для роли **Мастер** право **Просмотр** для документа **НачисленияСотрудникам**.

Поскольку этот документ принадлежит подсистеме **РасчетЗарплаты**, дадим право на просмотр этой подсистемы.

Также дадим права на просмотр справочника **ВидыГрафиковРаботы** и плана видов расчета **Основные начисления**, т.к. ссылки на эти объекты используются в документах **НачисленияСотрудникам**.

Вернемся к редактированию прав роли **Мастер**.

Как мы видим, при установке права **Просмотр** право **Чтение** документа **НачисленияСотрудникам** установилось автоматически. Выделим его.

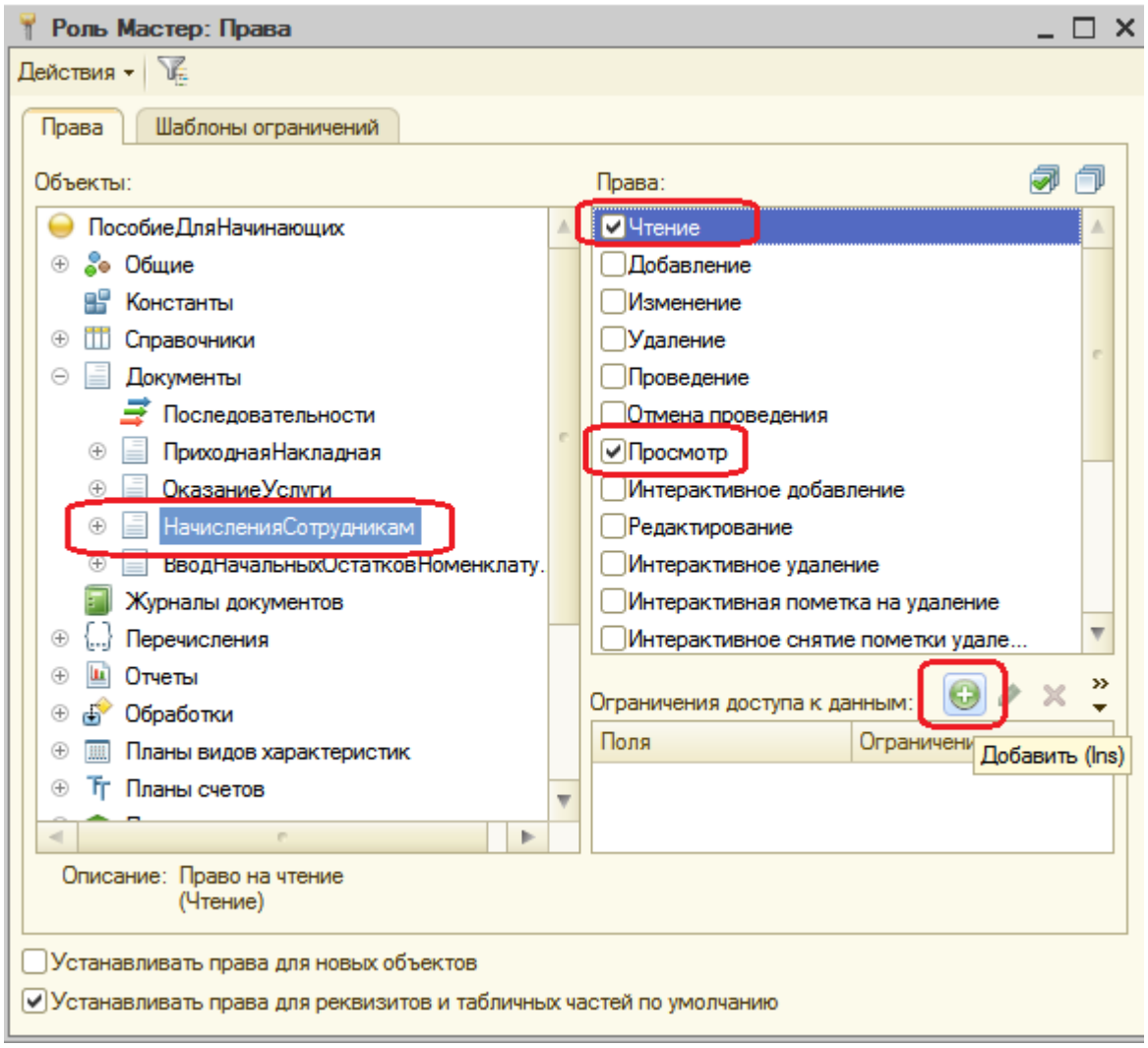
В правой нижней части экрана находится поле **Ограничение доступа к данным**. Нажмем кнопку **Добавить**.

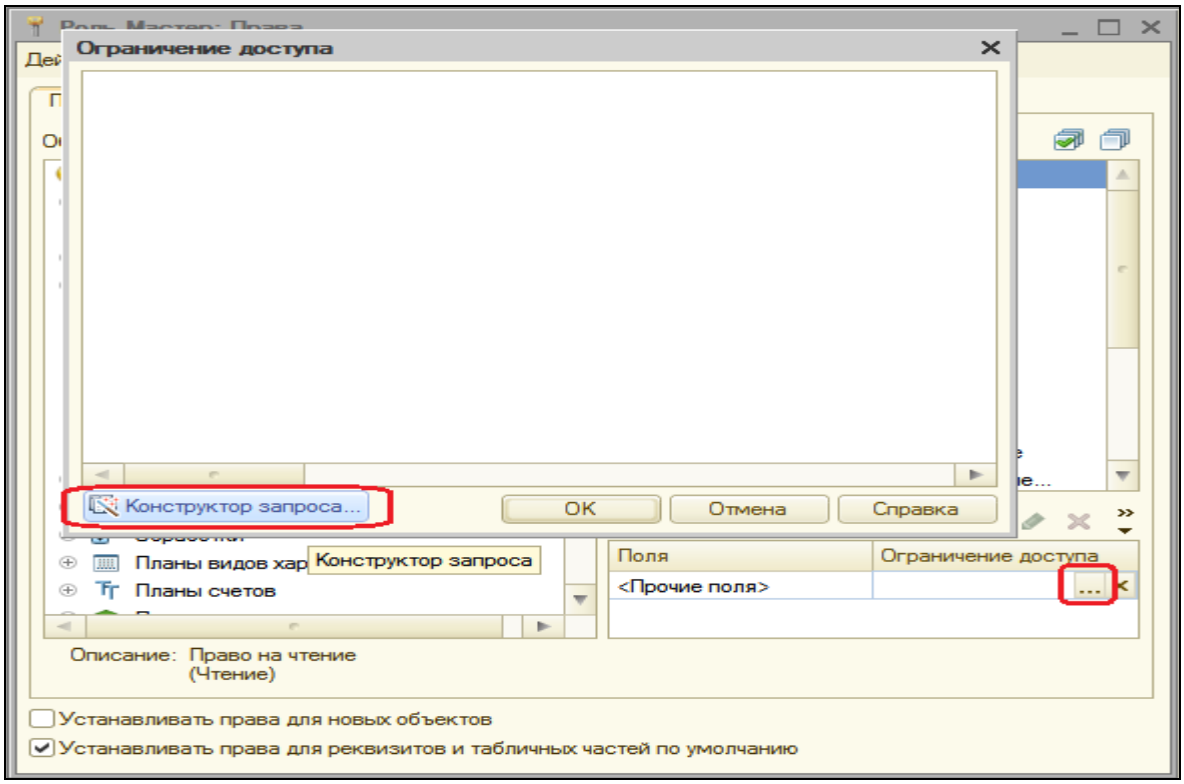
Мы хотим запретить доступ ко всем полям документа **Начисление сотрудникам**.

Поэтому мы не будем выбирать поля, а нажмем кнопку выбора в поле **Ограничение доступа**.

Откроется окно **Ограничение доступа**, в котором можно задать текст на специальном языке. Для облегчения работы воспользуемся конструктором запросов. Нажмем кнопку **Конструктор запроса**.

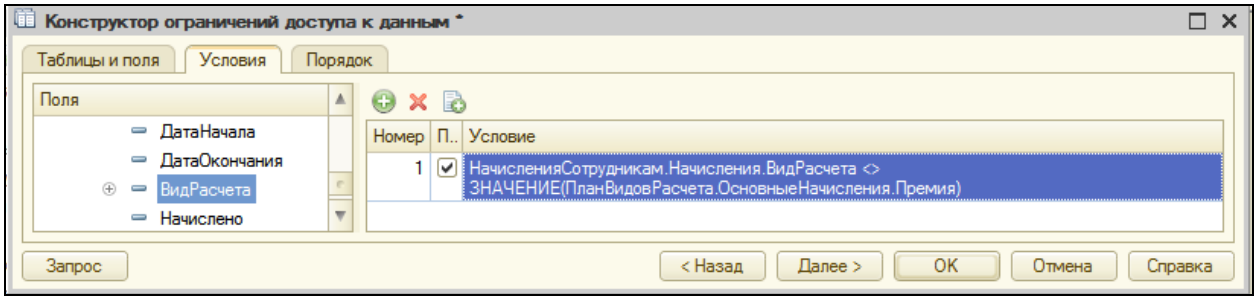
Таблица **НачисленияСотрудникам** автоматически попала на закладку **Таблицы и поля**, а конструктор открылся на вкладке **Условия**.


Роль Мастер: Права
Действия
Права
Шаблоны ограничений
Объекты:
ПособиеДляНачинающих
Общие
Константы
Справочники
Документы
Последовательности
ПриходнаяНакладная
ОказаниеУслуги
НачисленияСотрудникам
ВводНачальныхОстатковНоменклату...
Журналы документов
Перечисления
Отчеты
Обработки
Планы видов характеристик
Планы счетов
Права:
Чтение
Добавление
Изменение
Удаление
Проведение
Отмена проведения
Просмотр
Интерактивное добавление
Редактирование
Интерактивное удаление
Интерактивная пометка на удаление
Интерактивное снятие пометки удале...
Ограничения доступа к данным:
Поля
Ограничени
Добавить (Ins)
Описание: Право на чтение
(Чтение)
Устанавливать права для новых объектов
Устанавливать права для реквизитов и табличных частей по умолчанию

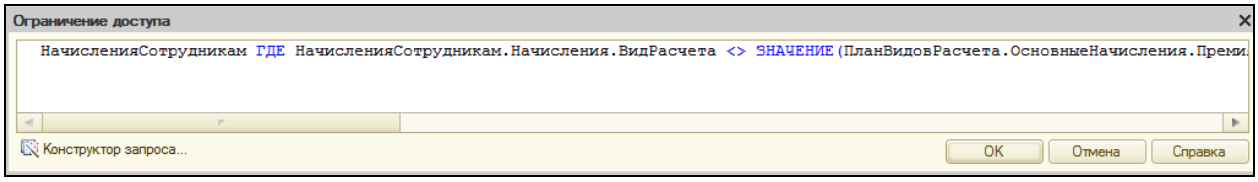

Ограничение доступа
Конструктор запроса...
OK
Отмена
Справка
Конструктор запроса
Планы видов хар
Планы счетов
Поля
Ограничение доступа
<Прочие поля>
Описание: Право на чтение
(Чтение)
Устанавливать права для новых объектов
Устанавливать права для реквизитов и табличных частей по умолчанию

Перенесем в список условий поле **ВидРасчета** табличной части **Начисления**, поставим флажок **Произвольное** и заполним поле условия:

```
НачисленияСотрудникам.Начисления.ВидРасчета <>
ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
```

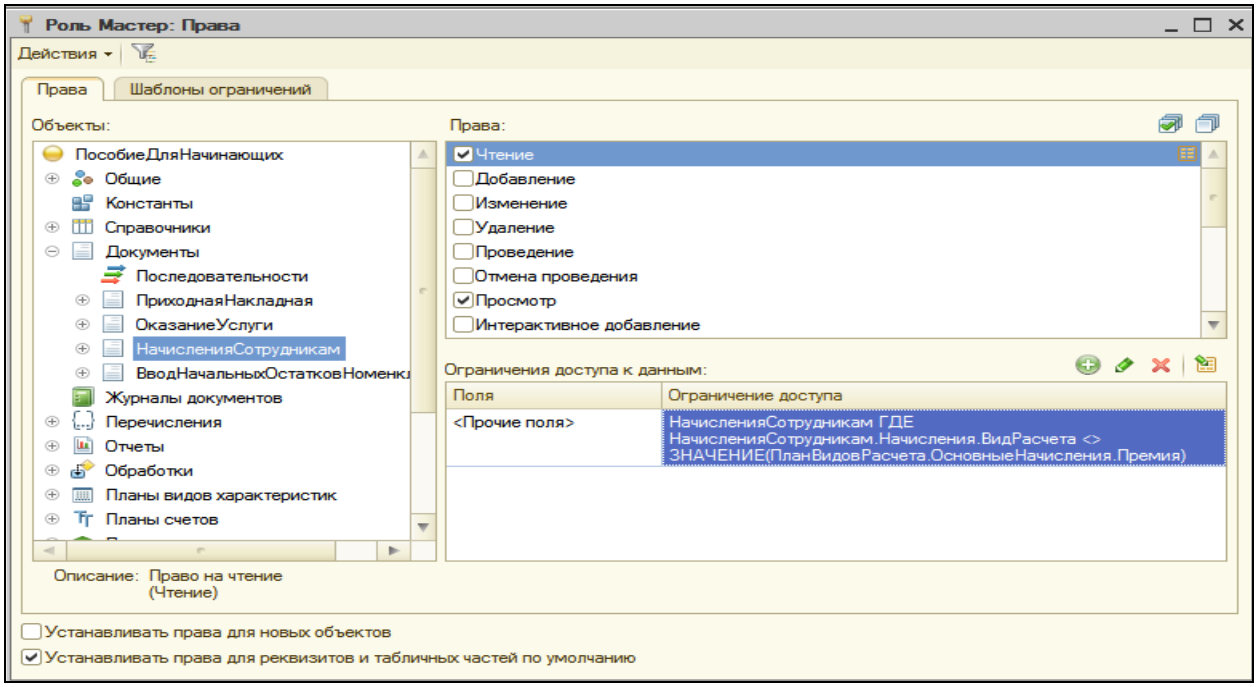


Нажмите ОК.



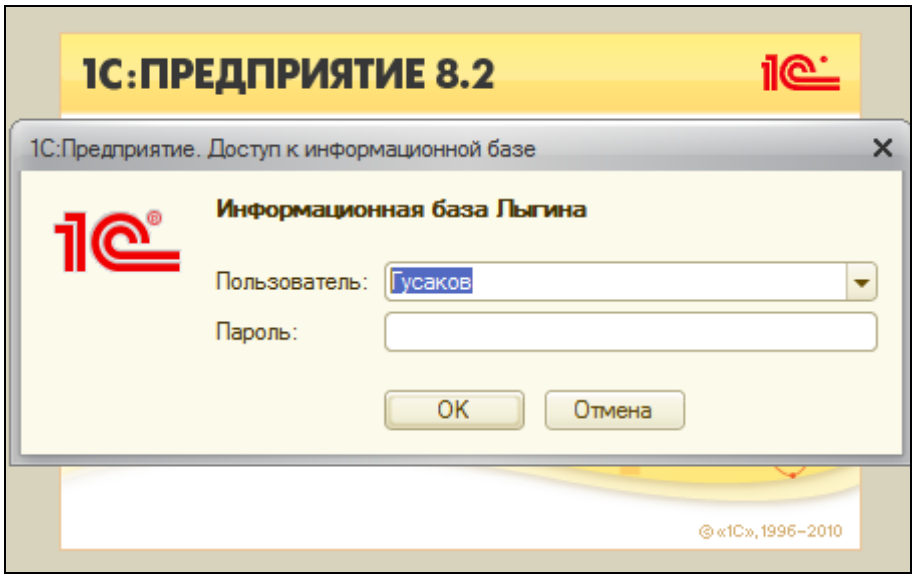
Текущий пользователь имеет право прочитать или изменить некоторый объект базы данных только в том случае, если ограничение доступа предоставляет ему такое право. Т.е. когда условие ограничения истинно.

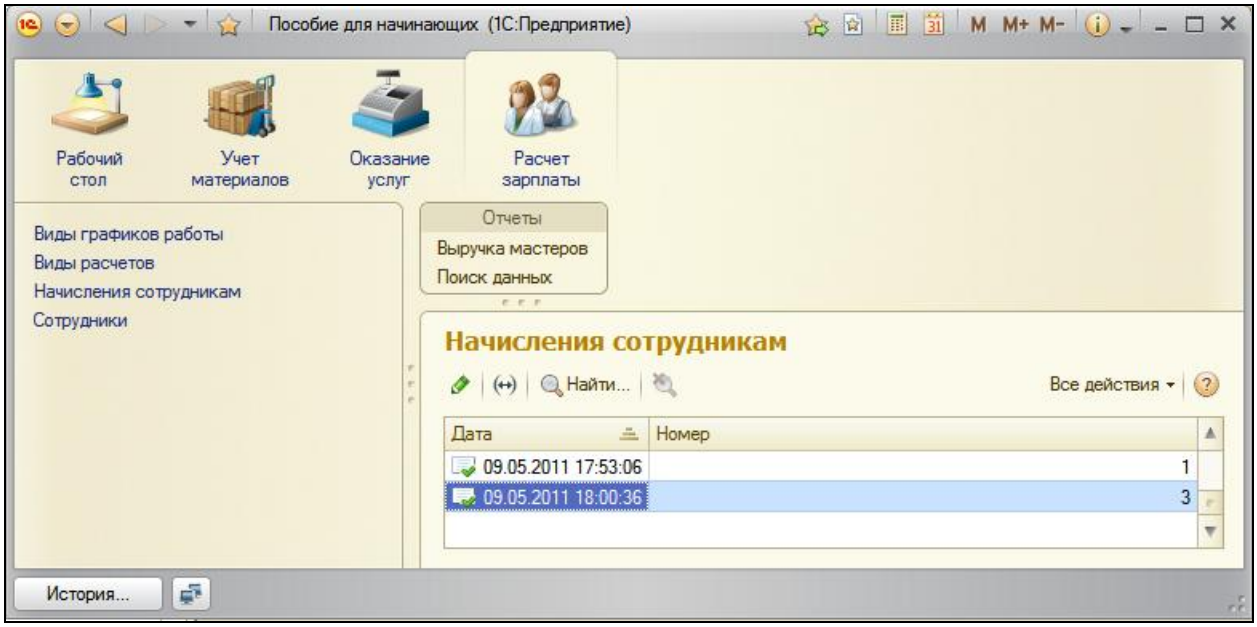
Окно ограничений доступа к данным для роли **Мастер** будет выглядеть так:



В режиме 1С:Предприятие

Обновим информационную базу и запустим 1С:Предприятие для пользователя с ролью **Мастер**, например, для пользователя **Гусаков**. В разделе **Расчет зарплаты** откроем список документов **НачисленияСотрудникам**.





Как мы видим, показаны только три подсистемы и только документы №1 и №3, т.к. в документе №2 начисляется премия.

Немного усложним задачу. Мы не хотим, чтобы мастер видел начисленные премии, но и не хотим скрывать от него факт существования такого документа.

Другими словами, в списке документов мастер должен его видеть, но не должен иметь возможность открыть его.

В режиме Конфигуратор

Вернемся в конфигуратор и посмотрим на наше ограничение.

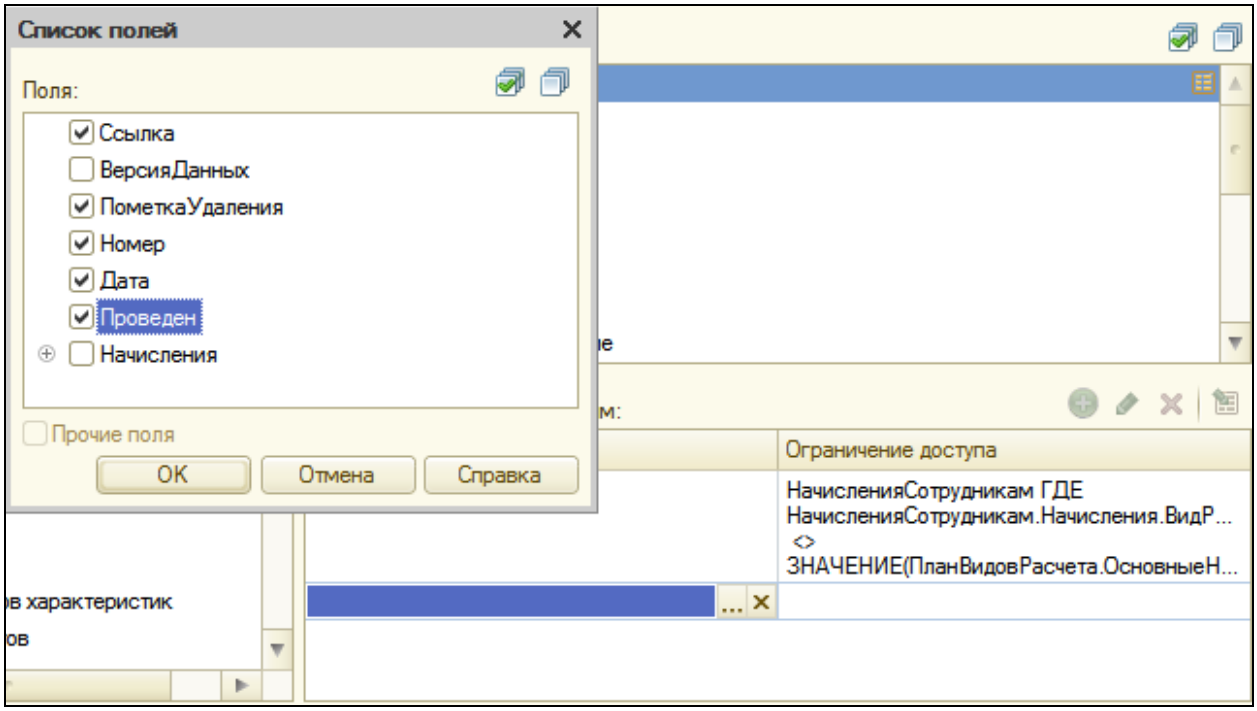
Мы не задавали никаких полей, поэтому ограничение применяется ко всем полям документа. Поэтому сейчас разрешим читать те поля, которые необходимы для отображения документа в списке.

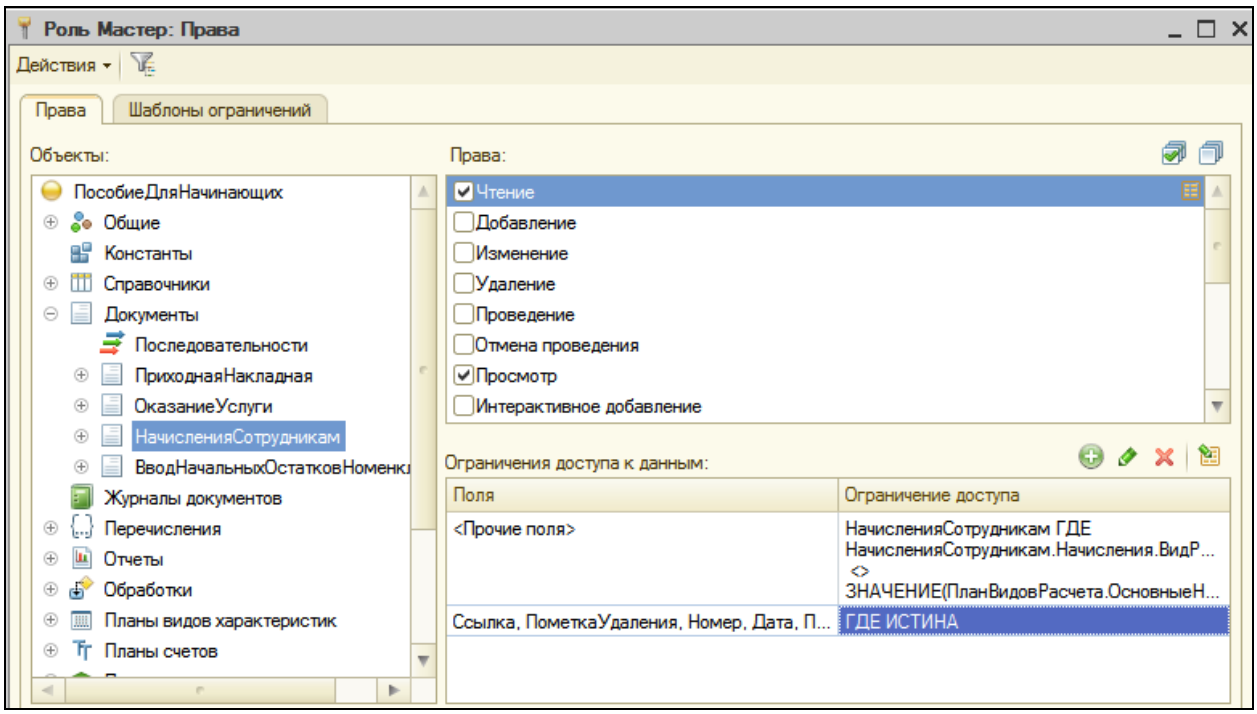
Но т.к. существующее условие на прочие поля мы удалять не будем, то открыть документ, как и раньше, можно будет только в том случае, если в его табличной части есть виды расчета, отличные от вида **Премия**.

Добавим к ограничениям доступа еще одно условие. В списке полей выберем поля:

- **Ссылка**,
- **ПометкаУдаления**,
- **Номер**,
- **Дата**,
- **Проведен**.

В ограничении доступа напишем ГДЕ ИСТИНА.



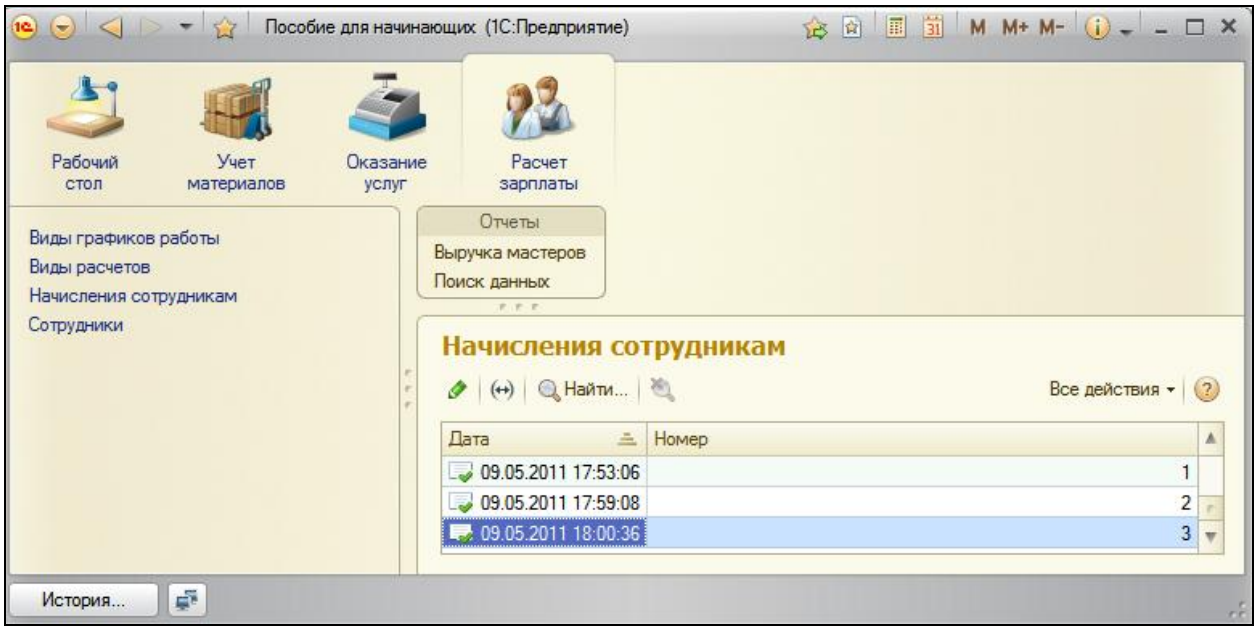


Закроем окно редактирования прав.

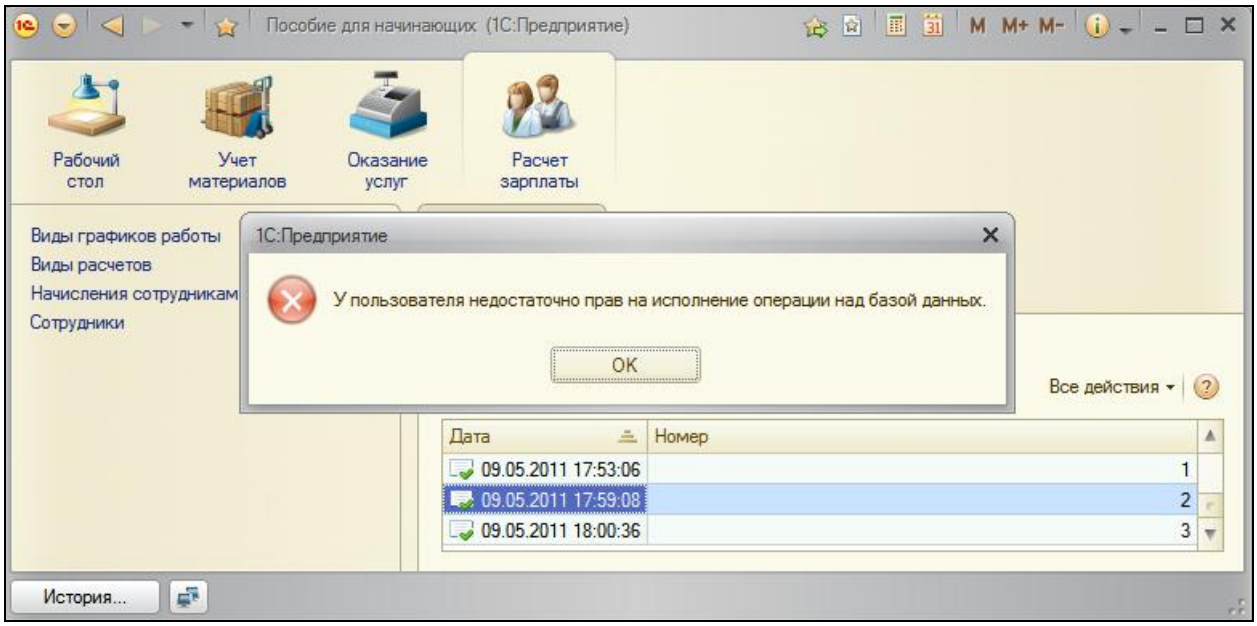
В режиме 1С:Предприятие

Обновим информационную базу и запустим отладку для пользователя с ролью **Мастер Гусаков**.

В разделе **Расчет зарплаты** откроем список документов **НачисленияСотрудникам**.



В списке документов мы увидим все документы начислений. Документы №1 и №3 мы сможем открыть и просмотреть, но при попытке открыть документ №2 мы получим сообщение о нарушении прав доступа.



Т.е. мы добились того, чего хотели.

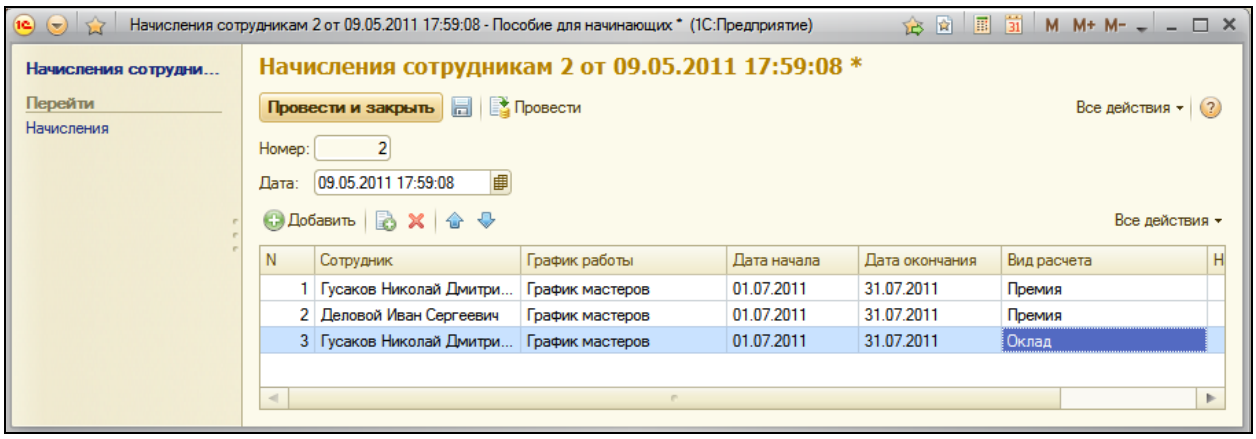
Все хорошо, пока в документе №2 содержатся записи только о расчете премии. Но вспомним, как регулируется наше ограничение доступа: пользователь сможет прочитать документ **Начисления сотрудникам** только в случае, если в его табличной части **Начисления** есть виды расчета, не являющиеся видом расчета **Премия**.

Это значит, что если в этом документе окажутся виды расчета, отличные от **Премия**, мастер сможет его открыть и просмотреть.

Убедимся в этом. Запустим 1С:Предприятие от имени пользователя **Администратор**.

В разделе **Расчет зарплаты** откроем список документов **Начисления сотрудникам**.

Откроем документ №2 и скопируем любую его строку. В новой строке изменим вид расчета на **Оклад**. Проведем и закроем документ. Завершим сеанс работы.



Теперь запустим 1С:Предприятие от имени пользователя **Гусаков**. Откроем список документов **НачисленияСотрудникам**. Откроем **Документ №2**. Документ откроется, и мы увидим все его строки.

В режиме Конфигуратор

Вернемся в конфигуратор.

Для того чтобы документ невозможно было просмотреть и в этой ситуации, нам нужно будет изменить существующее условие ограничения доступа.

Новое условие будет более сложным, поэтому мы заодно продемонстрируем использование шаблонов в ограничениях доступа.

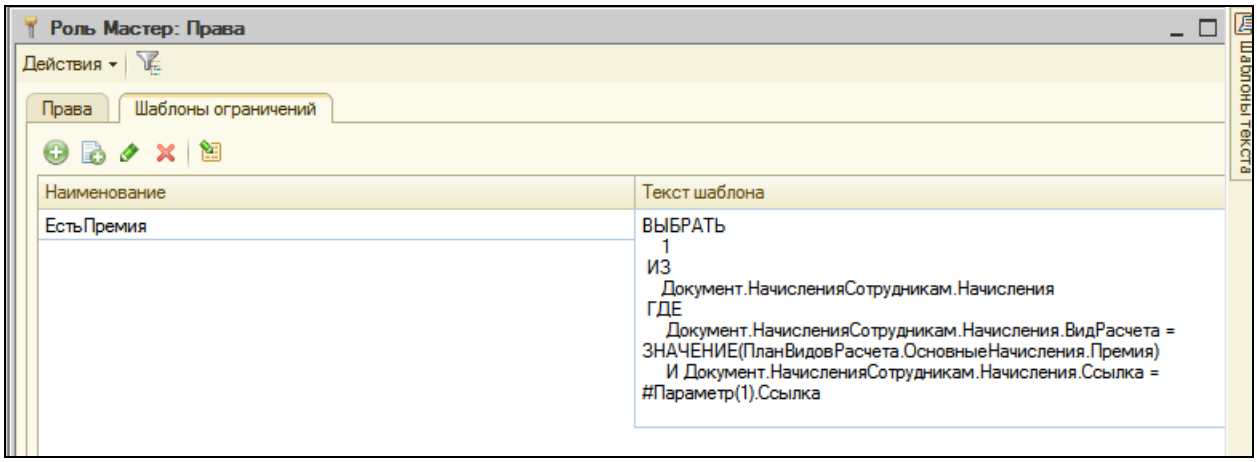
Откроем роль **Мастер** и перейдем на вкладку **Шаблоны ограничений**.

Здесь добавим новый шаблон с именем **ЕстьПремия**. Текст шаблона будет выглядеть так:

```
ВЫБРАТЬ
	1
ИЗ
	Документ.НачисленияСотрудникам.Начисления
ГДЕ
	Документ.НачисленияСотрудникам.Начисления.ВидРасчета =
ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
	И Документ.НачисленияСотрудникам.Начисления.Ссылка =
#Параметр(1).Ссылка
```

По сути это запрос к табличной части документа **НачисленияСотрудникам**, который либо не вернет нам ничего, либо вернет одну запись с одним полем со значением 1. Такую запись он

вернет нам в случае, если в табличной части документа есть вид расчета **Премия**.

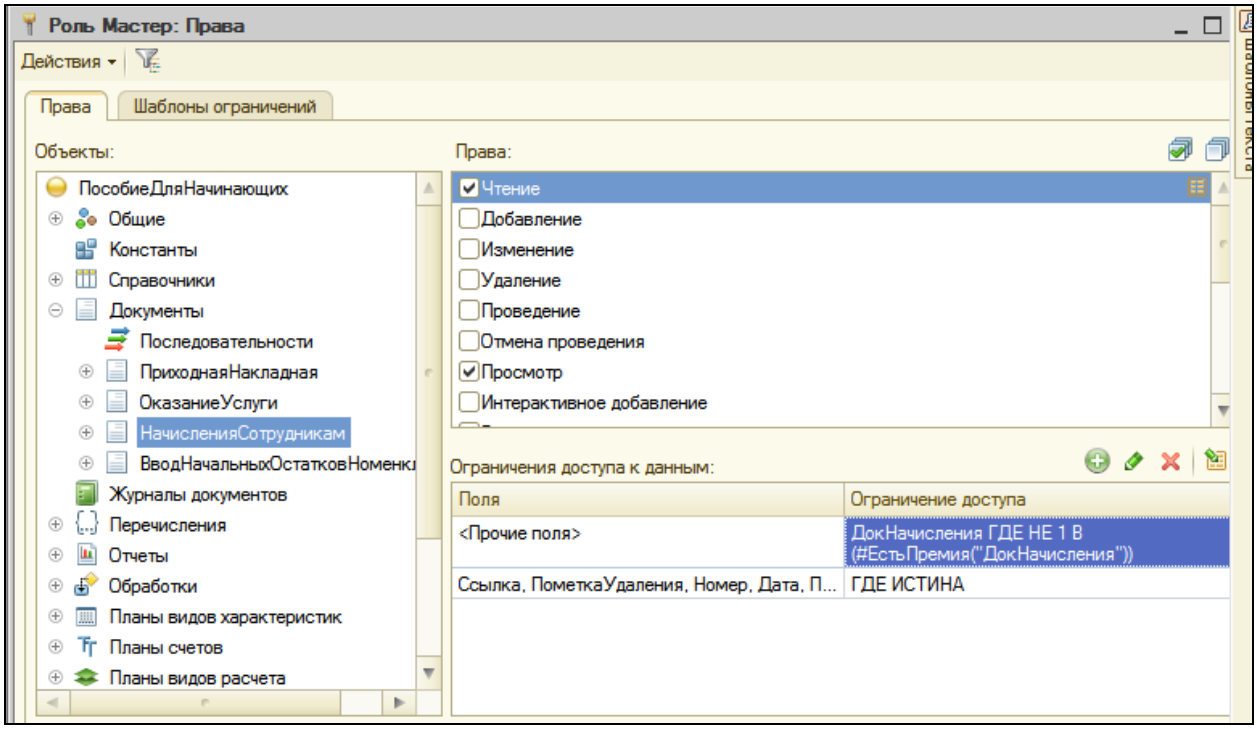


Второе условие (`И Документ.НачисленияСотрудникам.Начисления.Ссылка = #Параметр(1).Ссылка`) в этом запросе нужно нам для того, чтобы указать, табличная часть какого именно документа нас интересует. В этом условии используется возможность указания параметров в шаблоне.

Вместо `#Параметр(1).` будет подставлена та строка, которую мы укажем при вызове этого шаблона в условии ограничения доступа.

Теперь вернемся на закладку **Права**. В имеющемся ограничении прав для прочих полей заменим старый текст новым:

```
ДокНачисления ГДЕ НЕ 1 В (#ЕстьПремия("ДокНачисления"))
```



Здесь с помощью конструкции #ЕстьПремия("ДокНачисления") мы обращаемся к нашему шаблону. Текст шаблона просто будет подставлен в это место, причем строка ДокНачисления заменит собой первый параметр шаблона (#Параметр(1)).

Поэтому это условие разрешит нам прочитать ДокНачисления тогда, когда запрос из шаблона не возвращает 1.: ГДЕ НЕ 1 В (#ЕстьПремия("ДокНачисления"))

Т.е. тогда, когда в табличной части нет начисления **Премия**.

Можно было бы записать это условие и без использования шаблонов. Но, во-первых, такая запись была бы менее читаемой (листинг ниже), а во-вторых, использование шаблонов позволяет выделить и не дублировать части условий ограничений, которые могут использоваться в разных условиях.

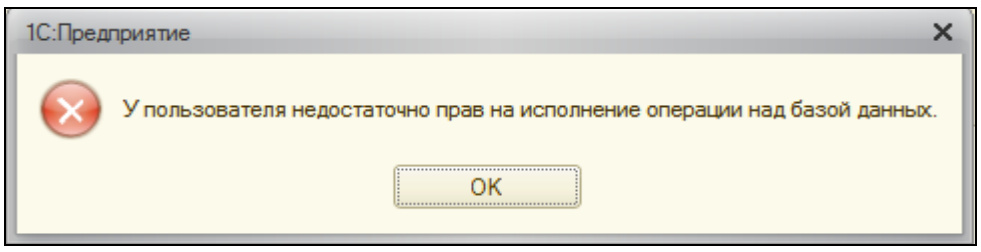
```
ДокНачисления ГДЕ НЕ 1 В (
	ВЫБРАТЬ
		1
	 ИЗ
		 Документ.НачисленияСотрудникам.Начисления
	ГДЕ
		Документ.НачисленияСотрудникам.Начисления.ВидРасчета =
ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
		И Документ.НачисленияСотрудникам.Начисления.Ссылка =
ДокНачисления.Ссылка)
```

Закроем окно редактирования прав и проверим как это работает.

В режиме 1С:Предприятие

Обновим конфигурацию и запустим Предприятие от имени **Гусакова**.

В разделе **Расчет зарплаты** откроем список документов **Начисления сотрудникам**. Как вы помните, в документе №2 есть строки и с видом расчета **Премия**, и с видом расчета **Оклад**. Раньше документ открывался. Попробуем открыть его теперь. Мы получим сообщение о нарушении прав доступа, что нам и требовалось.



В режиме Конфигуратор

Поскольку пример с ограничением прав доступа на уровне записей и полей базы данных мы делали скорее в демонстрационных целях, вернемся к исходному состоянию конфигурации.

Снимем для роли **Мастер** право **Чтение** для документа **НачисленияСотрудникам**.

Право **Просмотр** для подсистемы **РасчетЗарплаты**.

Право **Чтение** для справочника **ВидыГрафиковРаботы** и для плана видов расчета **Основные начисления**.

Запустим 1С:Предприятие от имени **Администратора**.

В разделе расчет зарплаты откроем список документов Начисления сотрудникам, откроем документ №2 и удалим последнюю строку, которую мы добавляли. Проведем и закроем документ.

Контрольные вопросы

- ✓ Для чего предназначен объект конфигурации Роль.
- ✓ Как создать роль, используя подсистемы конфигурации.
- ✓ Как создать список пользователей системы и определить их права.
- ✓ Чем аутентификация средствами 1С:Предприятия отличается от аутентификации операционной системы.

Практическая работа № 22

Рабочий стол и настройка командного интерфейса (1:10)

В этой работе мы придадим нашей конфигурации «товарный» вид, т.е. усовершенствуем командный интерфейс приложения, настроим рабочий стол и видимость команд по ролям для созданных нами пользователей.

Это сделает интерфейс более завершенным и более удобным для пользователя, что очень важно для работы с приложением.

Командный интерфейс разделов

Пришло время заняться организацией командного интерфейса разделов (подсистем), т.е. осмысленно отсортировать команды, разложить их по группам в зависимости от приоритета и частоты использования.

В режиме Конфигуратор

Для начала зададим синоним для отчета **ПоискДанных** как **Поиск в данных**. Так будет более понятно для пользователя.

Затем вызовем редактор командного интерфейса подсистем **Все подсистемы** (**Общие – Подсистемы – Все подсистемы**).

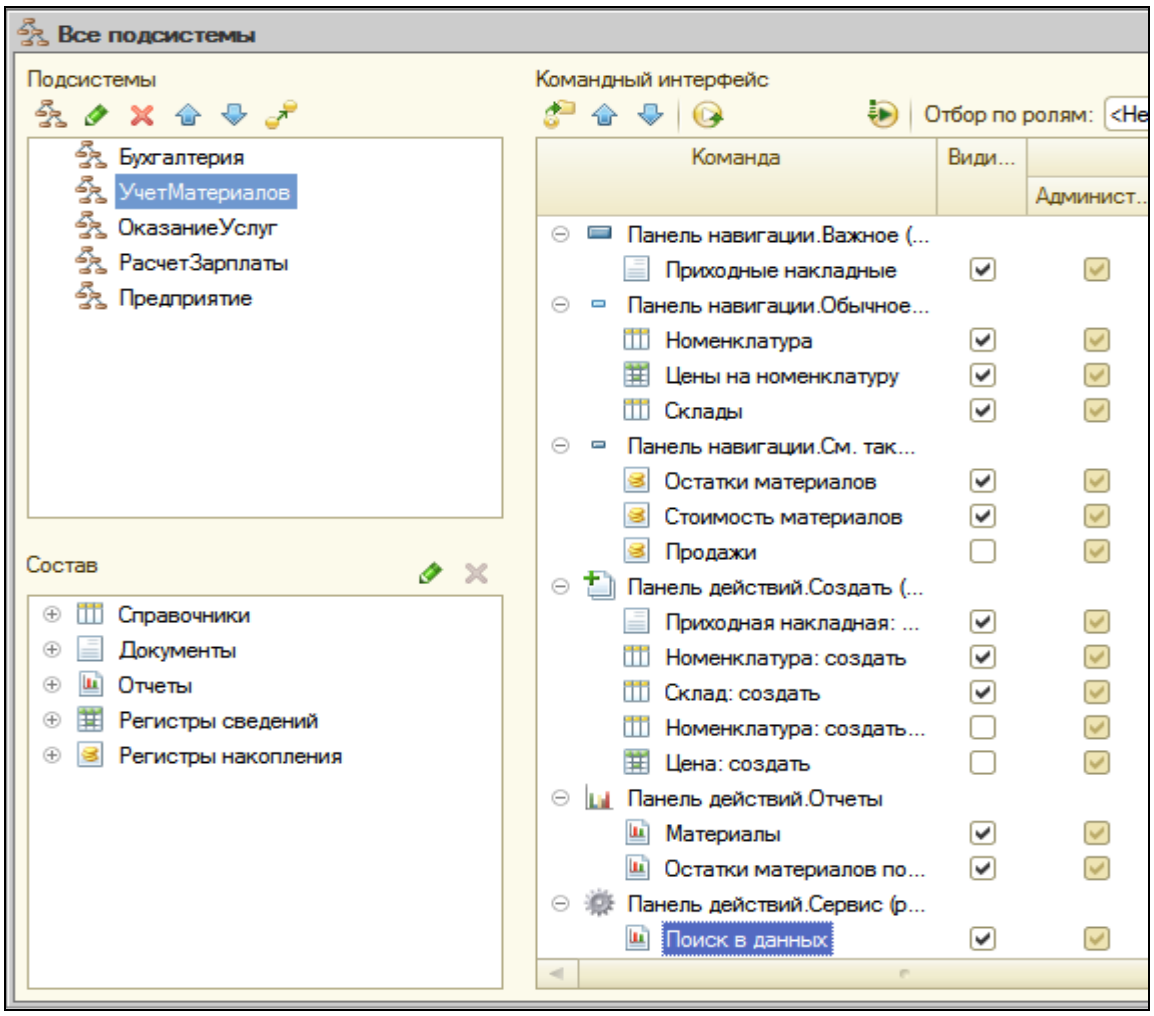
Выделим в списке подсистем **УчетМатериалов** и в окне командный интерфейс произведем следующие изменения.

- С помощью мыши переместим команду **Приходные накладные** из группы **Панель навигации.Обычное** в группу команд **Панель навигации.Важное.**
- В группе **Панель навигации.Обычное** зададим следующий порядок расположения команд:
    - **Номенклатура**,
    - **Цены на Номенклатуру**,
    - **Склады**.
- В группе **Панель навигации.См. также** уберем видимость у команды **Продажи** и зададим следующий порядок:
    - **Остатки материалов**,

- **Стоимость материалов**.

- В группе **Панель действий.Создать** зададим порядок расположения видимых команд (порядок невидимых команд нам не важен):
  - **Приходная накладная: создать**,
  - **Номенклатура: создать**,
  - **Склад: создать**.
- В группе **Панель действий.Отчеты** зададим порядок:
  - **Материалы**,
  - **Остатки материалов по свойствам**.
- Переместим команду **Поиск в данных** из группы **Панель действий.Отчеты** в группу команд **Панель действий.Сервис**.

В результате окно **Подсистемы** примет вид:



Поясним наши действия.

В разделе **Учет материалов** наиболее часто пользователю может понадобиться создавать приходные накладные и просматривать их список. Поэтому мы поместили команду для просмотра списка приходных накладных (**Приходные накладные**) в группу **Важное** панели навигации, а команду для создания приходных накладных поместили первой в панели действий подсистемы **Учет материалов**.

Довольно часто пользователю может понадобиться создавать новую номенклатуру и просматривать список номенклатуры. Поэтому мы поместили команду для просмотра списка номенклатуры (Номенклатура) в группу **Обычное** панели навигации, а команду для создания номенклатуры поместили второй в панели действий подсистемы Учет материалов. Тоже самое, но с меньшим приоритетом, относится к справочнику складов.

В группе **См.также** панели навигации раздела мы поместили команды **Остатки материалов** и **Стоимость материалов** для просмотра записей соответствующих регистров накопления и расположили их в порядке частоты использования. А команду для просмотра записей регистра **Продажи** вообще убрали, т.к. в разделе **Учет материалов** вряд ли она может понадобиться.

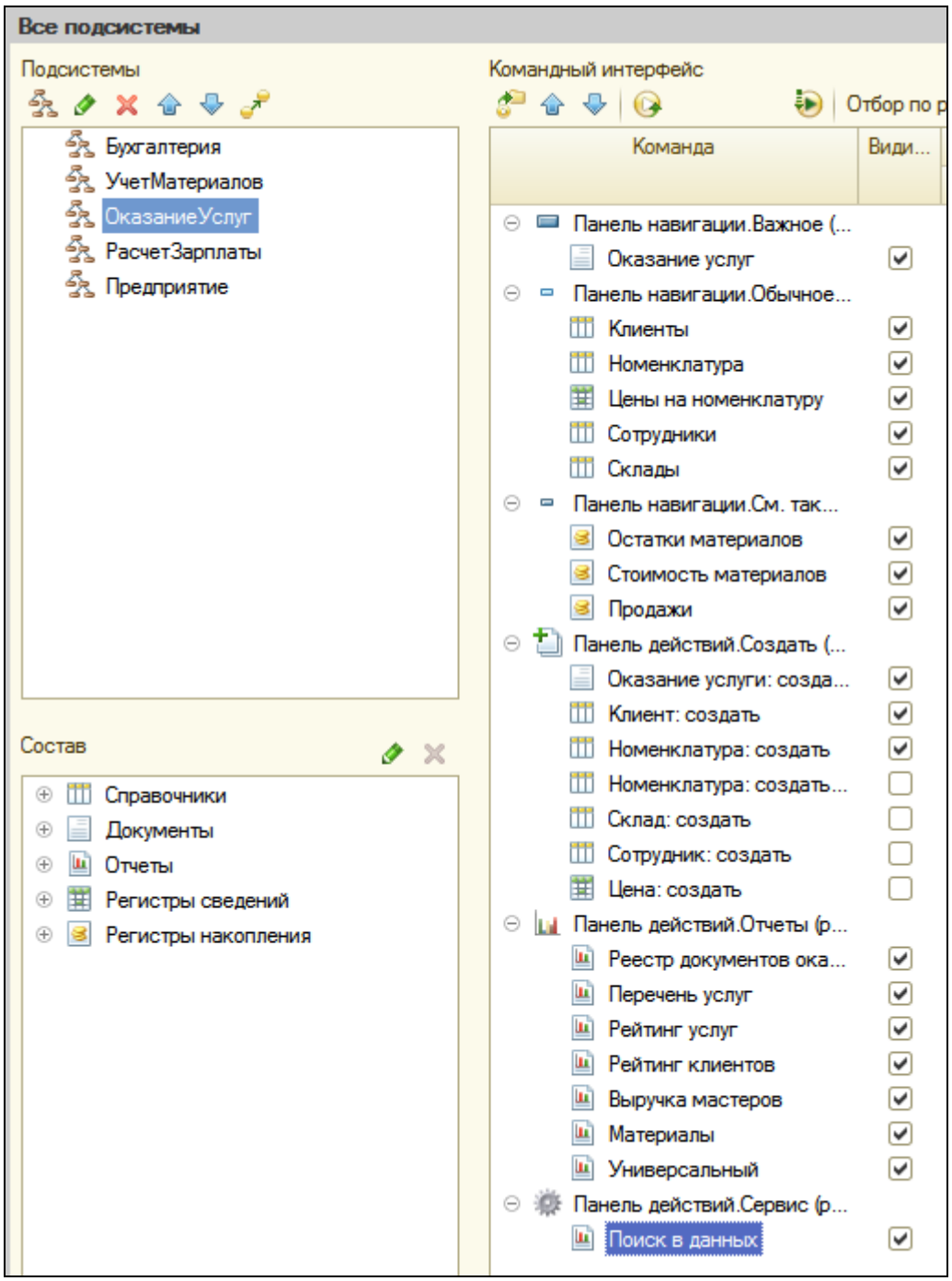
Отчеты в панели действий раздела мы расположили в порядке их приоритета. А команду для открытия отчета **Поиск в данных** мы перенесли из группы **Отчеты** в группу **Сервис** панели действий, т.к. на самом деле поиск данных – это скорее сервисная операция, чем классический отчет.

Руководствуясь подобными соображениями, отредактируем командный интерфейс остальных подсистем.

Подсистема **ОказаниеУслуг** будет иметь следующий интерфейс:

- Группа **Панель навигации.Важное**:
    - **Оказание услуг**.
- Группа **Панель навигации.Обычное**:
    - **Клиенты**,
    - **Номенклатура**,
    - **Цены на Номенклатуру**,

    - **Сотрудники**,
    - **Склады**.
- Группа **Панель навигации.См. также**:
    - **Остатки материалов**,
    - **Стоимость материалов**,
    - **Продажи**.
- **Панель действий.Создать** (только видимые команды):
    - **Оказание услуги: создать**,
    - **Клиент: создать**,
    - **Номенклатура: создать**.
- **Панель действий.Отчеты**:
    - **Реестр документов Оказание услуги**,
    - **Перечень услуг**,
    - **Рейтинг услуг**,
    - **Рейтинг клиентов**,
    - **Выручка мастеров**,
    - **Материалы**
    - **Универсальный**.
- **Панель действий.Сервис**:
    - **Поиск в данных**.



Подсистема **Бухгалтерия** будет иметь следующий командный интерфейс:

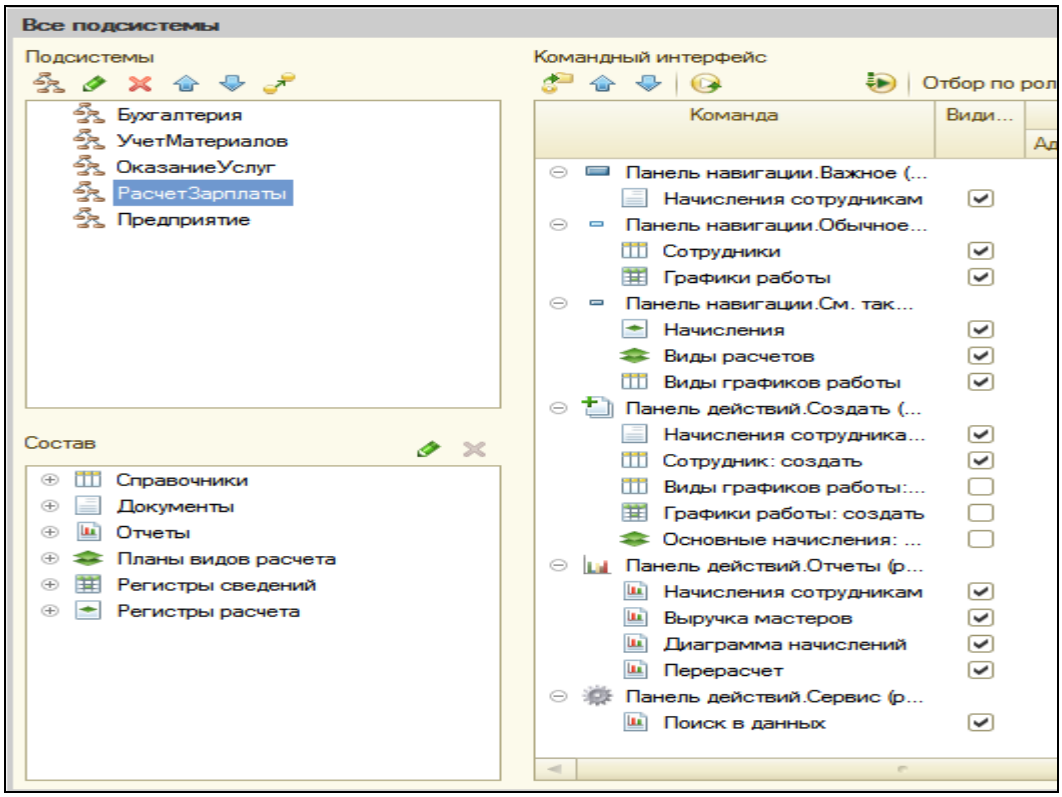
- **Панель навигации.Важное**:
  - **Приходные накладные**,
  - **Оказание услуг**.
- **Панель навигации.Обычное**:
  - **Клиенты**,

        - **Номенклатура**,
        - **Цены на номенклатуру**.
    - **Панель навигации.См. также**:
        - **Остатки материалов**,
        - **Стоимость материалов**,
        - **Продажи**,
        - **Ввод начальных остатков номенклатуры**,
        - **Основной план счетов**,
        - **Виды субконто**
    - **Панель действий.Создать** (только видимые):
        - **Ввод начальных остатков номенклатуры: создать**.
    - **Панель действий.Отчеты**:
        - **Оборотно-сальдовая ведомость**,
        - **Начисления сотрудникам**,
        - **Перечень услуг**,
        - **Рейтинг услуг**,
        - **Рейтинг клиентов**,
        - **Материалы**,
        - **Остатки материалов по свойствам**.
    - **Панель действий.Сервис**:
        - **Поиск в данных**.

Подсистема **РасчетЗарплаты:**

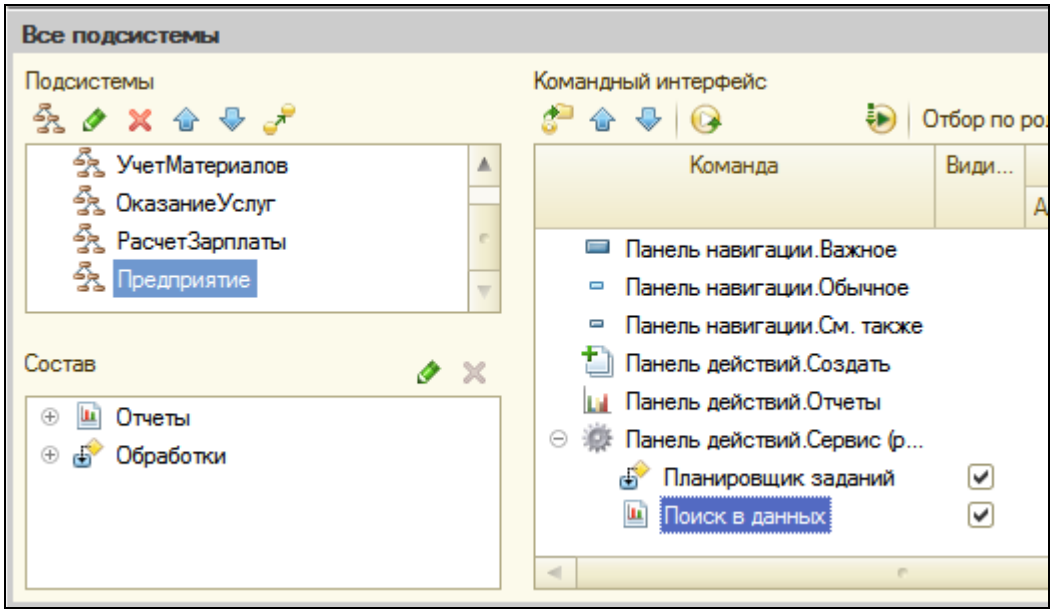
- **Панель навигации.Важное**:
    - **Начисления сотрудникам**.
- **Панель навигации.Обычное**:
    - **Сотрудники**,

    - **Графики работы**.
- **Панель навигации.См. также**:
    - **Начисления**,
    - **Виды расчетов**,
    - **Виды графиков работы**.
- **Панель действий.Создать** (только видимые):
    - **Начисления сотрудникам: создать**,
    - **Сотрудник: создать.**
- **Панель действий.Отчеты**:
    - **Начисления сотрудникам**,
    - **Выручка мастеров**,
    - **Диаграмма начислений,**
    - **Перерасчет.**
- **Панель действий.Сервис**:
    - **Поиск в данных**.



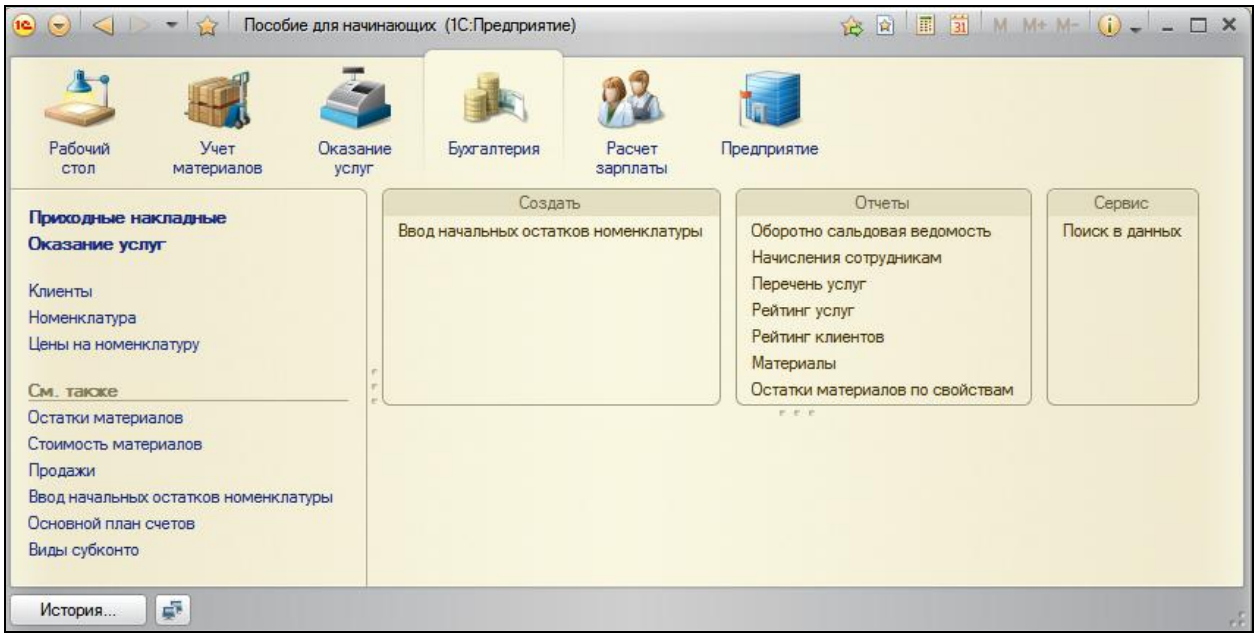
Подсистема **Предприятие:**

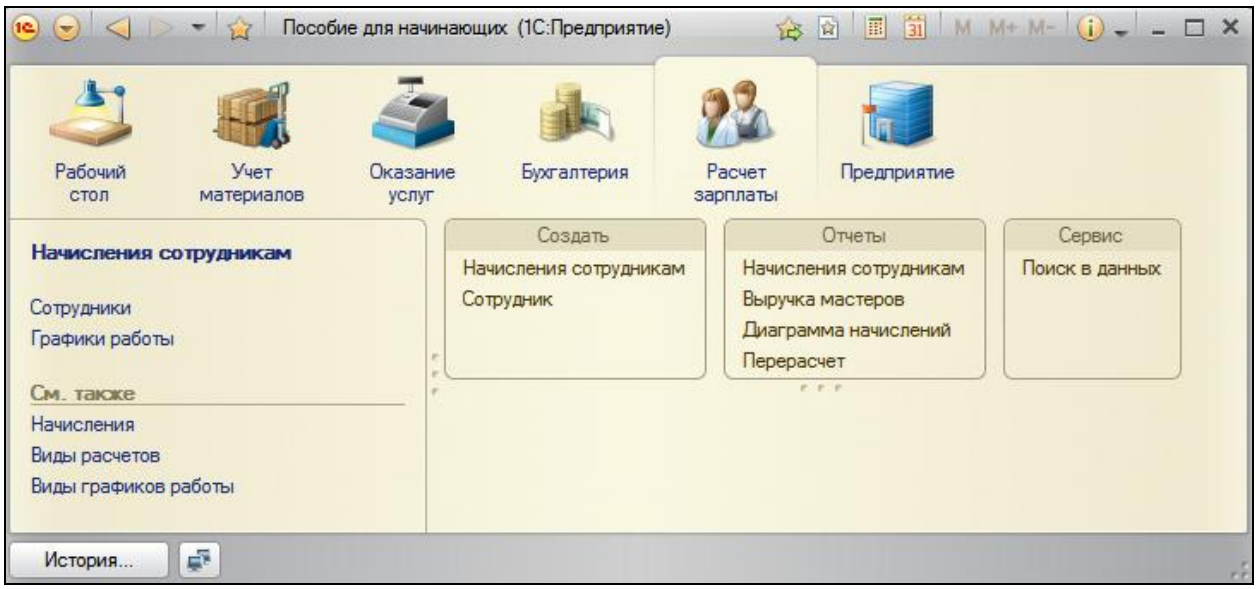
- **Панель действий.Сервис:**
  - **Планировщик заданий,**
  - **Поиск в данных.**



В режиме 1С:Предприятие

Запустим режим отладки под Администратором. Так будет выглядеть интерфейс разделов **Бухгалтерия** и **Расчет зарплаты**:





Как мы видим, все группы команд зрительно отделены друг от друга, а наиболее важные команды выделены жирным шрифтом.

Однако, если у пользователя есть другие предпочтения в настройке интерфейса, то в режиме 1С:Предприятие он может сам настроить интерфейс, выполнив команду главного меню **Сервис – Настройка интерфейса**.

Рабочий стол

Рабочий стол предназначен для размещения наиболее часто используемых пользователем документов, отчетов, справочников и т.п. Поэтому нужно поместить на рабочий стол те формы документов, отчетов и т.п. работа с которыми входит в его ежедневные обязанности.

Например, для кладовщика было бы удобно иметь под руками список номенклатуры и список прикладных накладных, для менеджера – список клиентов и документов оказания услуг.

При запуске 1С:Предприятия раздел **Рабочий стол** становится активным по умолчанию и нужные формы сразу открываются в рабочей области приложения.

В режиме Конфигуратор

Начнем настройку рабочего стола.

Выделим корень дерева объектов конфигурации, вызовем контекстное меню и выберем пункт **Открыть рабочую область рабочего стола**.

Откроется окно настройки рабочего стола. Сначала вверху окна выберем шаблон рабочего стола **Две колонки разной ширины (2:1)**.

Это значит, что формы на рабочем столе будут располагаться в две колонки, при этом левая будет в два раза шире правой.

Можно выбрать другой шаблон, при котором колонки будут одинаковой ширины, или будет всего одна колонка. Но кажется, что предпочтительнее первый вариант, т.к. в этом случае взгляд пользователя сразу будет падать на наиболее приоритетные для работы формы, которые мы расположим в левой колонке.

Формы в каждой колонке будут располагаться друг под другом. Оптимально, если в левой колонке будет располагаться одна, максимум две формы, а в правой – две-три формы для каждого пользователя.

Следует иметь ввиду, что автоматически созданные системой формы нельзя располагать на рабочем столе. Поэтому нужно создать форму в явном виде в конфигурации. Чтобы не путаться, будем создавать формы прямо по ходу.

Начнем настройку рабочего стола для роли Мастер.

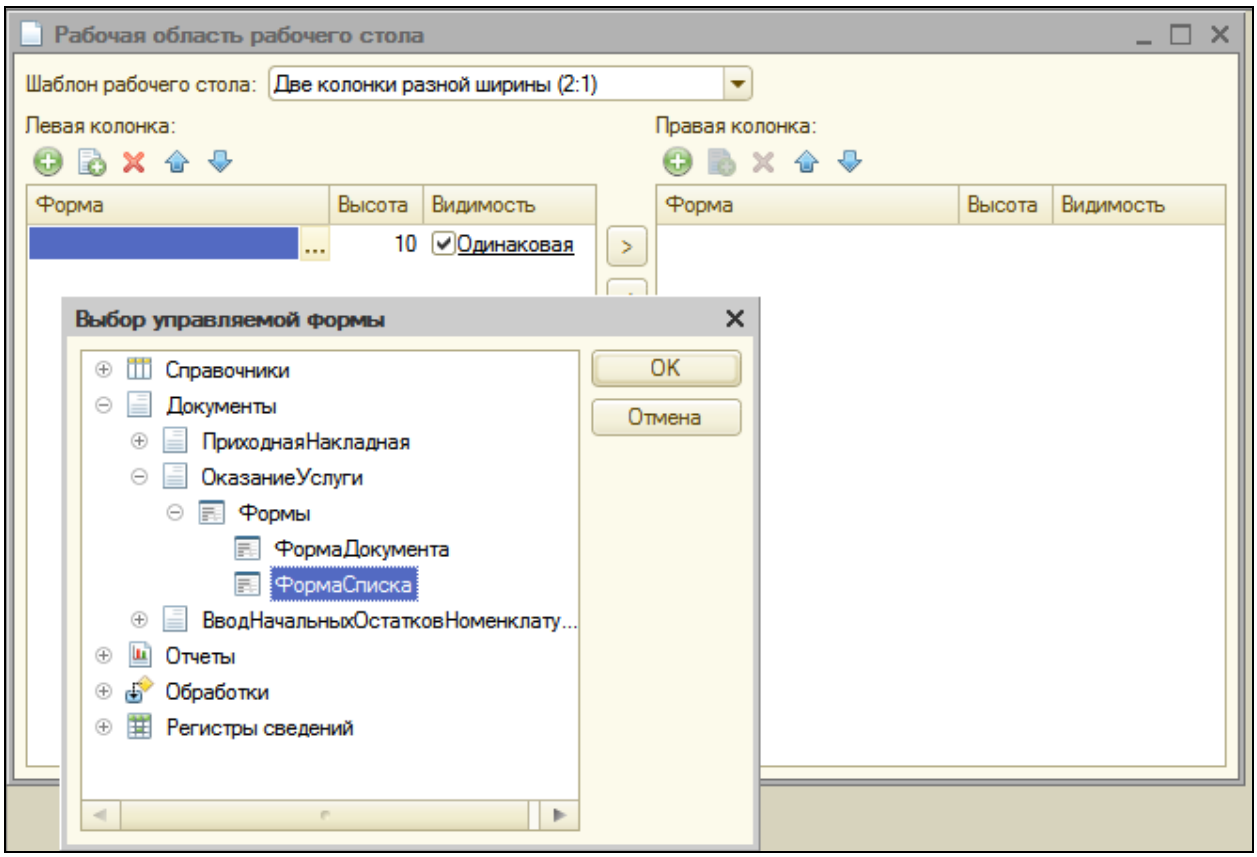
Наша фирма специализируется на оказании услуг и мастера имеют к этому непосредственное отношение. Поэтому логично, если для них в левой колонке рабочего стола будет располагаться список документов об оказании услуг, а в правой – список приходных накладных и список клиентов.

Перечисленные формы списка отсутствуют в конфигурации, поэтому создадим формы списка для объектов конфигурации:

- Справочник **Клиенты**,
- Документ **ПриходнаяНакладная**,
- Документ **ОказаниеУслуги**.

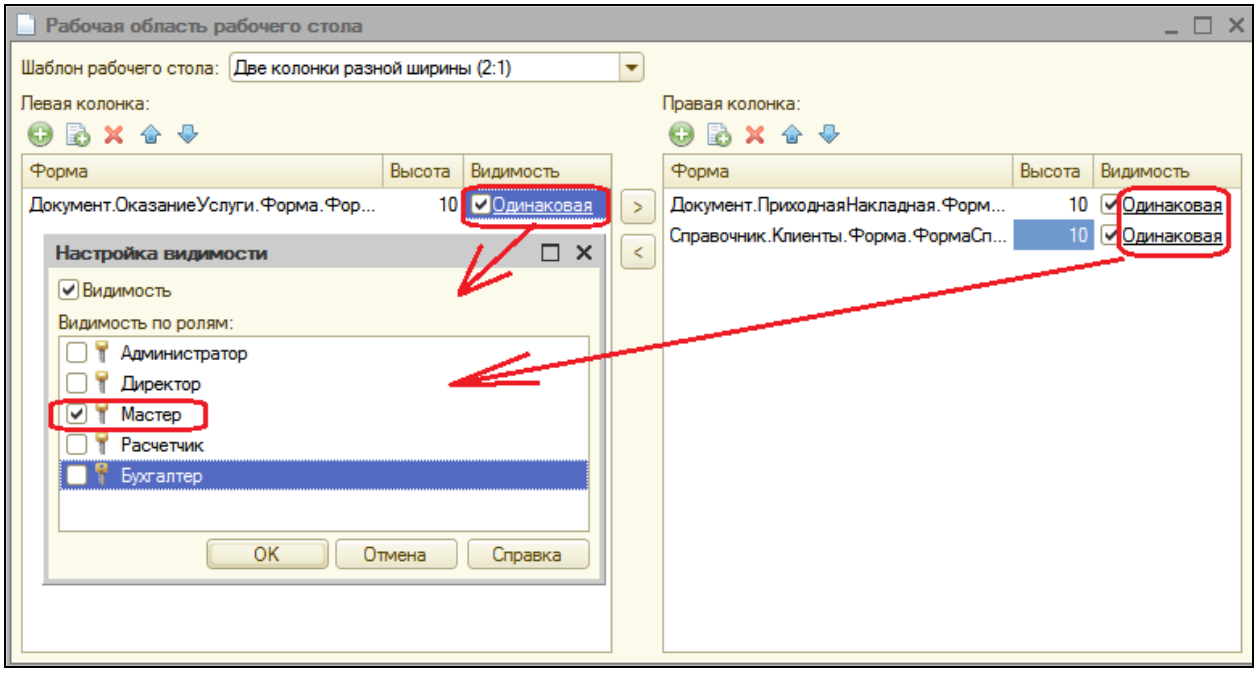
После этого перейдем в окно настройки рабочего стола и над списком форм левой колонки нажмем **Добавить**.

Выберем форму списка документа **ОказаниеУслуги**.



Аналогичным образом добавим в правую колонку формы списка документа **ПриходнаяНакладная** и справочника **Клиенты**.

Теперь для каждой из трех форм нажмем ссылку в колонке **Видимость** и установим видимость этих форм для роли **Мастер**.



Возможно, эти формы понадобятся на рабочем столе для пользователей с другими ролями, но мы сейчас, чтобы не запутаться, ограничимся только ролью **Мастер**.

Теперь настроим рабочий стол для роли **Бухгалтер**.

Предположим, что бухгалтер наиболее часто будет пользоваться оборотно-сальдовой ведомостью и отчетом о начислениях сотрудникам. Расположим эти отчеты в левой колонке рабочего стола, а правую оставим пустой.

Создадим форму отчета для отчета **ОборотноСальдоваяВедомость,** а для отчета **НачисленияСотрудникам** форму отчета мы уже создали ранее.

Затем перейдем в окно настройки рабочего стола, добавим эти формы в левую колонку и установим видимость этих форм только для роли **Бухгалтер**.

Аналогично настроим рабочий стол для роли **Расчетчик**.

По роду деятельности расчетчик в основном пользуется документами и отчетами о начислениях сотрудникам. Расположим отчет о начислениях сотрудникам в левой колонке, а в правой – список документов о начислениях.

Для этого создадим форму списка документа **НачисленияСотрудникам**.

Теперь перейдем в окно настройки рабочего стола, добавим форму списка документа НачисленияСотрудникам в правую колонку и установим видимость этой формы только для Расчетчика.

Отчет **НачисленияСотрудникам** мы уже добавили для роли **Бухгалтер**, поэтому установим также его видимость для роли **Расчетчик**.

Затем настроим рабочий стол для роли **Директор**.

Мы предполагаем, что эта роль будет назначена пользователю с руководящими функциями. Ему не нужно вводить никаких документов, да и у него и нет на это прав.

Но ему понадобится регулярно просматривать отчеты о деятельности фирмы, чтобы принимать решения.

Поэтому расположим на его столе в левой колонке отчет **Рейтинг услуг**, а в правой – отчеты **Перечень услуг** и **Выручка мастеров**.

Создадим формы отчета для этих отчетов. Затем перейдем в окно настройки рабочего стола и добавим эти формы, установим видимость только для роли **Директор**.

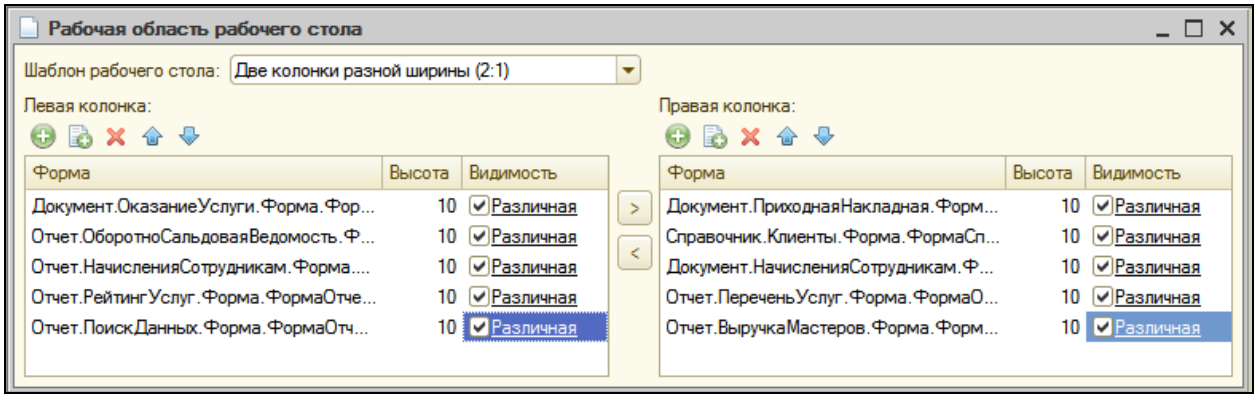
Рабочий стол для роли **Администратор**.

По роду деятельности администратор отвечает за состояние информационной базы и может иметь доступ ко всем объектам конфигурации. Но для администратора не должно быть проблемой найти и выполнить любую команду нашей конфигурации. Поэтому лучше узнать у администратора, что ему хочется иметь под рукой.

Для примера расположим на его рабочем столе в левой колонке отчет для поиска данных, а правую оставим пустой.

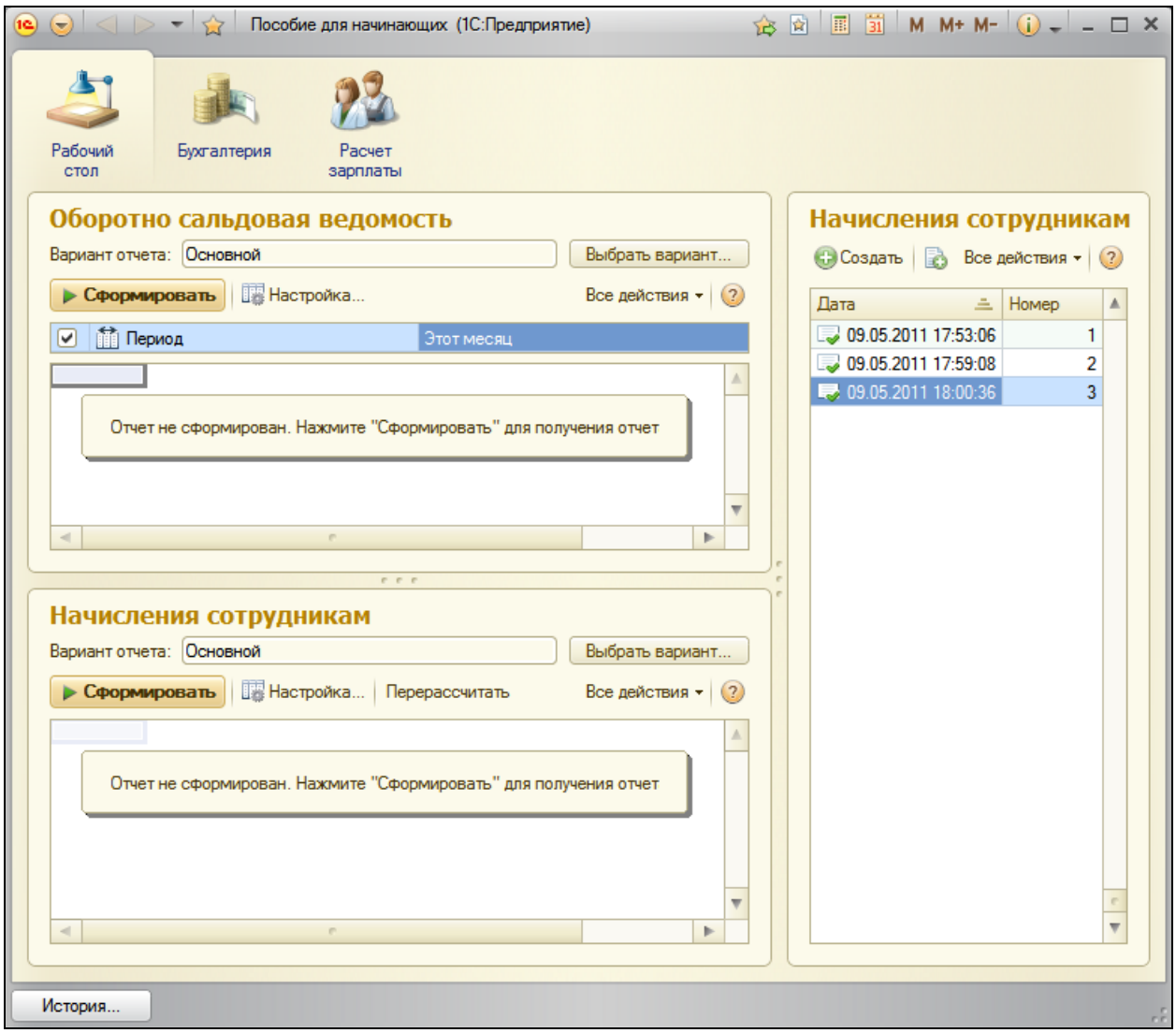
Форму для отчета **ПоискДанных** мы уже создали ранее. Теперь перейдем в окно настройки рабочего стола, добавим форму в левую колонку и установим видимость для роли **Администратор**.

В результате окно настройки рабочего стола должно принять вид:

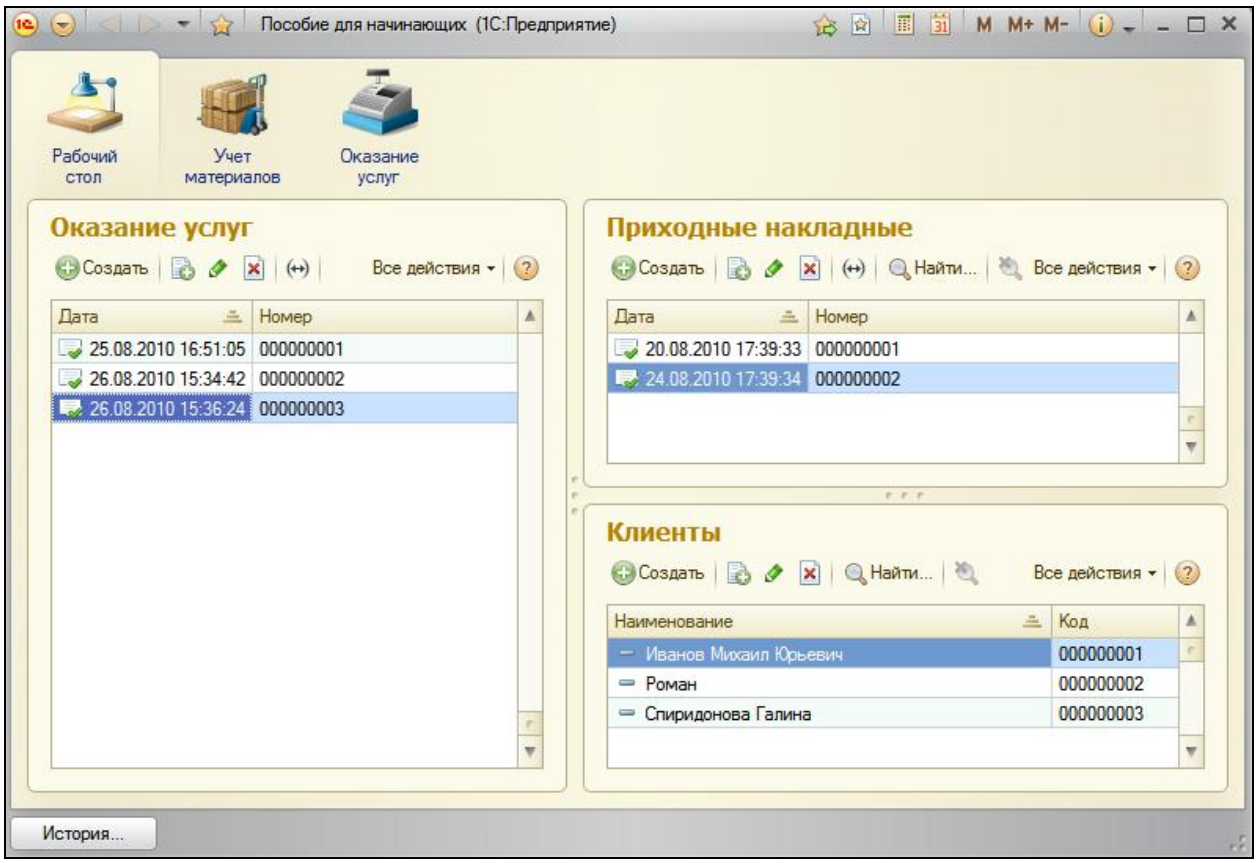


В режиме 1С:Предприятие

Зайдем в режим отладки под пользователем **Назарова** (**Бухгалтер** и **Расчетчик**) – потребуется создать отдельный сеанс через **Пуск – 1С**…(**Тонкий клиент**), увидим такой рабочий стол:



А для пользователя **Гусаков** (с ролью **Мастер**) рабочий стол будет таким:



Можно также настроить панель навигации и панель действий рабочего стола, выполнив команду **Открыть командный интерфейс рабочего стола**, но вряд ли в этом есть необходимость, т.к. все нужные команды есть в панели действий и в панели навигации разделов, доступных пользователю.

Наконец, в процессе работы 1С:Предприятия 8 пользователь может настраивать рабочий стол по своему усмотрению, выполнив команду главного меню **Сервис – Настройка интерфейса – Рабочий стол**.

Но следует помнить, что пользователь может размещать на своем рабочем столе только формы, заданные разработчиком в конфигурации.

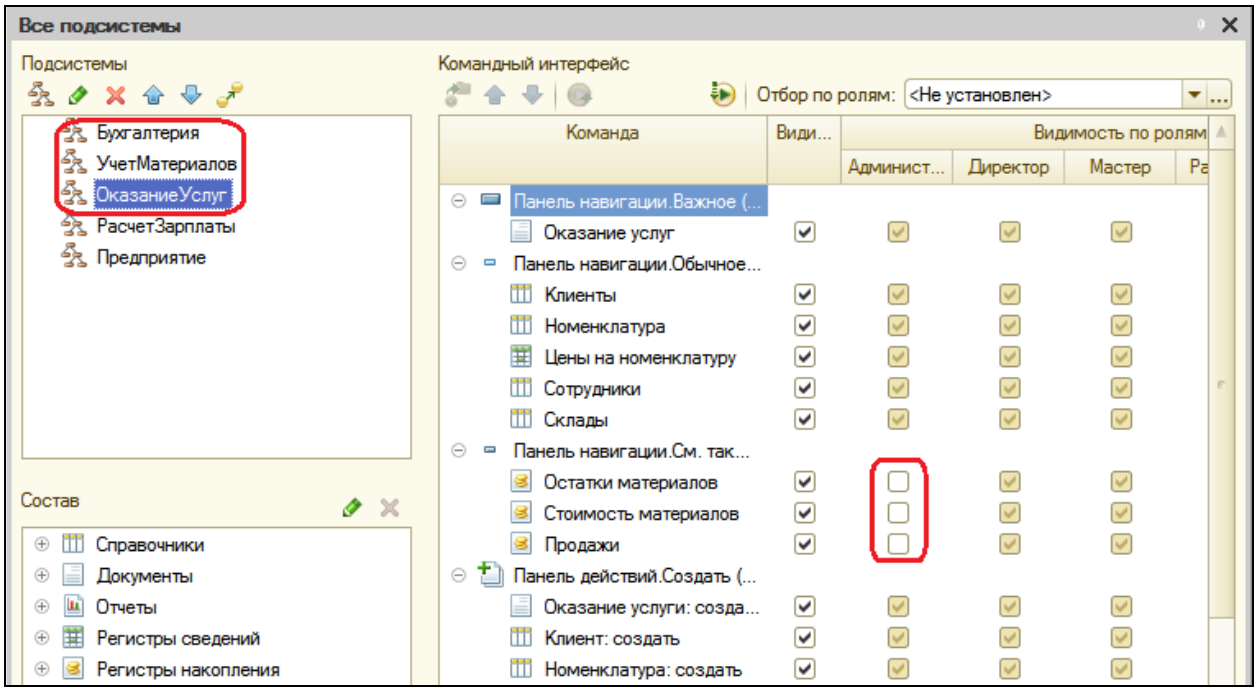
Видимость команд по ролям

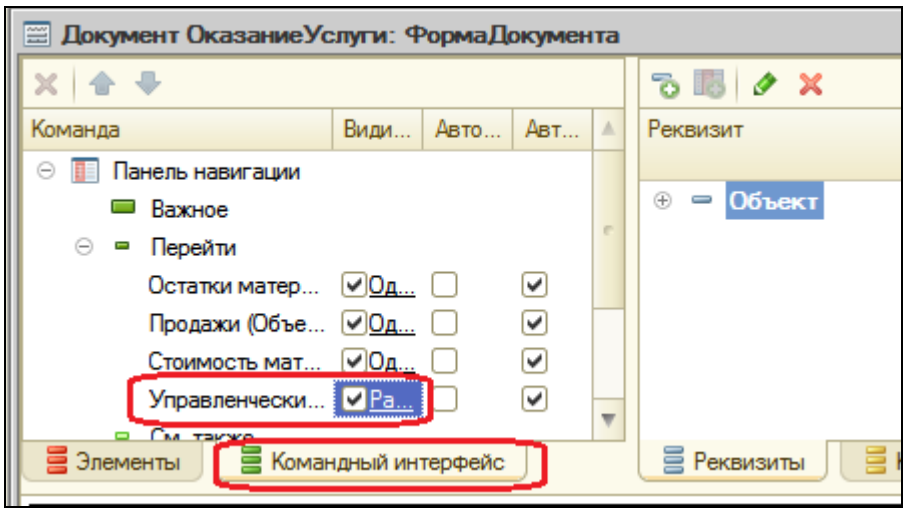
Используя команду Общие – Подсистемы – Все подсистемы, мы не раз редактировали командный интерфейс подсистем – меняли порядок, включали видимость команд и т.д. после появления ролей в конфигурации появилась возможность настройки видимости этих команд по ролям.

Ролевое редактирование видимости команд по умолчанию – это средство, позволяющее настроить начальную «насыщенность» глобального командного интерфейса в первую очередь для пользователей с широкими правами доступа.

Для администратора системы, с его широкими правами доступа, число команд оказалось слишком большим. Предположим, что командой открытия регистров накопления из панели навигации разделов он будет пользоваться крайне редко.

Для облегчения его командного интерфейса, опять перейдем к редактированию глобального командного интерфейса (**Общие – Подсистемы – Все подсистемы**) и для соответствующих команд каждой подсистемы снимем видимость для роли **Администратор**. А также откроем формы документов **ОказаниеУслуги** и **ПриходнаяНакладная** и на закладке **Командный интерфейс** в панели навигации снимем видимость у команд перехода к регистру бухгалтерии **Управленческий** для роли **Администратор**.





В режиме 1С:Предприятие

Запустите режим отладки и убедитесь, что команды открытия регистров накопления не видны для администратора, хотя он имеет к ним доступ. Например, через главное меню: Все функции – Регистры накопления.

В реальной конфигурации, конечно, роли пользователей и наполнение их рабочих столов будут другими. Это зависит от специфики работы предприятия и пожеланий заказчика.

В последних двух работах мы показали только принцип организации интерфейса прикладного решения по ролям и ограничения доступа к отдельным командам и разделам в целом в зависимости от прав пользователя.

Контрольные вопросы

- ✓ Что такое рабочий стол
- ✓ Как настроить рабочий стол для различных пользователей
- ✓ Как настроить видимость команд по ролям.

Практическая работа № 23

Обмен данными (6:10)

Общие сведения об обмене данными

В этой работе мы познакомимся с механизмами обмена данными, которые содержит система 1С:Предприятие 8, и добавим в нашу конфигурацию возможность обмена данными с удаленными филиалами и отделениями.

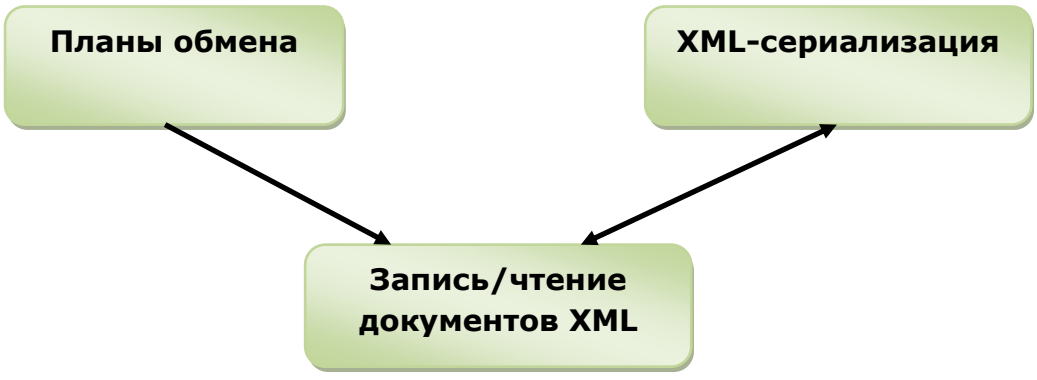
Механизмы обмена данными позволяют организовывать обмен информацией, хранимой в базе данных, с другими программными системами, основанными на 1С:Предприятие 8 или другими.

Такая гибкость обмена достигается за счет использования средств обмена данными в различных комбинациях. Кроме того, формат обмена основан на языке XML, являющимся на сегодняшний день общепринятым средством представления данных.

К механизмам обмена данными могут быть отнесены:

- Планы обмена,
- XML-сериализация,
- Средства чтения и записи документов XML.

Схема взаимодействия этих трех составляющих может быть такой:



При помощи *планов обмена* мы получаем информацию о том, какие элементы данных были изменены и в какой узел обмена их необходимо передать.

Это возможно благодаря тому, что планы обмена содержат механизм регистрации изменений. Информация об измененных данных переносится с помощью сообщений, инфраструктура которых также поддерживается планами обмена.

XML-сериализация позволяет преобразовать объект 1С:Предприятия 8 в последовательность данных, представленных в формате XML. Кроме этого, она выполняет и обратное преобразование – преобразует последовательность данных формата XML в объект 1С:Предприятия 8, при условии что имеется соответствующий тип данных.

Запись и чтение документов XML обеспечивает запись/чтение документов формата XML из встроенного языка.

При реализации алгоритма обмена данными перечисленные механизмы могут быть использованы как все вместе, так и в различной комбинации. В каждом конкретном случае разработчик решает эту задачу самостоятельно.

Что такое план обмена

Чтобы существовала возможность обмена какими-либо данными с кем-либо, необходимо некоторым образом идентифицировать тех, с кем мы будем обмениваться, и для каждого из них описать перечень обмена. Обе эти задачи позволяет решать объект конфигурации *План обмена*.

Элементами данных плана обмена являются узлы плана обмена.

Каждый узел идентифицирует участника обмена по данному плану обмена. В каждом плане обмена всегда существует один предопределенный узел, идентифицирующий данную информационную базу.

В одной конфигурации может существовать несколько планов обмена. Каждый план обмена определяет набор данных, которым будет производиться обмен в рамках данного плана, и сам механизм обмена.

Наличие нескольких планов может потребоваться, если с разными узлами ведется обмен разным составом данных, или когда схема организации обмена с одними узлами отличается от схемы организации обмена с другими узлами.

В обмене данными могут участвовать:

- Объекты базы данных: элементы справочников, документы и т.д.
- Необъектные данные: наборы записей регистров, последовательностей, константы.
- Специальный объект встроенного языка – **УдалениеОбъекта**.

Для упрощения изложения в дальнейшем будем называть эти элементы информационных структур *объектами обмена*.

Разработчик имеет возможность определить состав каждого плана обмена, указав объекты конфигурации, данные которых должны участвовать в обмене по данному плану и указать для каждого типа объектов признак **Авторегистрация**. Этот признак определяет, каким образом план обмена будет отслеживать изменения данных.

Возможность отслеживать изменения данных реализована в плане обмена за счет использования *механизма регистрации изменений*.

Работа этого механизма базируется на том, что каждый из объектов обмена имеет свойство ОбменДанными, с помощью которого можно указать, для каких узлов необходимо производить регистрацию изменений этого объекта. Любые изменения объекта обмена сводятся к записи или удалению объекта обмена. Механизм регистрации изменений анализирует события записи и удаления объектов обмена и на основании параметров обмена данными, содержащихся в каждом из объектов обмена, формирует записи регистрации изменений. Свойство **ОбменДанными** не хранится в БД, а используется только во время записи объекта обмена.

Признак **Авторегистрация**, устанавливаемый при указании состава данных плана обмена, позволяет указать, что параметры обмена данными будут формироваться каждый раз самим механизмом регистрации изменений на основании информации, содержащейся в плане обмена.

После автоматического заполнения параметров обмена разработчик все же имеет возможность внести изменения в сформированные таким образом параметры. Для этого следует использовать обработчики событий объектов, участвующих в обмене, - **ПередЗаписью** и **ПередУдалением**, в которых можно модифицировать список узлов-получателей (т.е. тех узлов, для которых регистрируются изменения).

Как мы теперь знаем, при записи и удалении объектов обмена план обмена формирует *записи регистрации изменений*. Они хранятся в

таблицах регистрации изменений, причем для каждого объекта обмена своя таблица.

При изменении объекта обмена в таблице регистрации изменений создается столько записей (строчек), сколько узлов-получателей указано в параметрах обмена данными у объекта обмена. Каждая запись при этом будет хранить ссылку на свой узел-получатель. Таблицы регистрации изменений создаются лишь в том случае, если соответствующий объект метаданных указан в составе хотя бы одного плана обмена.

Кроме ссылки на узел обмена, каждая запись таблицы регистрации хранит номер *сообщения*, в котором изменение было передано в первый раз в этот узел. До тех пор, пока сообщение не будет передано в первый раз, это поле хранит Null.

Сообщение с точки зрения плана обмена – это единица обмена информацией. Поэтому одной из важнейших составляющих плана обмена, помимо службы регистрации изменений, является *инфраструктура сообщений*.

Поскольку сообщения передаются в рамках плана обмена от одного узла к другому, каждое сообщение точно ассоциировано с планом обмена, имеет уникальный номер и одного отправителя и получателя. За нумерацию сообщений отвечает инфраструктура сообщений. Благодаря этому записи регистрации изменений и имеют возможность хранить номера сообщений, в которых эти изменения были переданы в первый раз.

Инфраструктура сообщений позволяет также получать подтверждения от узла-получателя о приеме сообщений. Такое подтверждение содержится в каждом сообщении, приходящем от узла-получателя в виде номера последнего принятого сообщения.

Впоследствии, проанализировав номер последнего принятого сообщения и номера сообщений, содержащиеся в записях регистрации изменении, разработчик может удалить записи регистрации изменений, прием которых подтвержден получателем.

XML-сериализация

Это механизм, позволяющий представить объект 1С:Предприятия в виде последовательности данных в формате XML. Кроме этого позволяет выполнить и обратное преобразование.

Дело в том, что объект обмена, являющийся в системе единым целым, на самом деле представляет собой совокупность данных различных типов, определенным образом связанных между собой.

Например, элемент справочника, кроме кода и наименования, может содержать некоторое количество реквизитов различного типа и некоторое количество табличных частей, содержащих в свою очередь, некоторые количество реквизитов различного типа.

Запись/чтение документов XML

В отличие от XML-сериализации, механизмы *записи/чтения документов XML* позволяют работать с данными формата XML на базовом уровне, без привязок к объектам 1С:Предприятия.

В частности, они позволяют открывать файлы XML для чтения, читать данные из файлов, создавать новые файлы и записывать в них данные.

Универсальный механизм обмена данными

Наша фирма открыла свой филиал в городе N и установило в нем такую же конфигурацию для учета работы филиала.

В результате возникла необходимость наладить обмен данными между этими двумя базами т.о., чтобы каждая из них отражала полную информацию о материалах и услугах, в то время как бухучет и расчет зарплаты велись бы в каждой базе отдельно.

Для этого мы создадим план обмена, опишем состав данных, которые будут включены в обмен, и сделаем несколько процедур, позволяющих нам формировать на жестком диске файлы обмена и соответственно загружать полученные файлы обмена с жесткого диска.

Для упрощения примера мы не будем программировать какой-либо автоматический обмен файлами между двумя базами, и запуск процедуры обмена будем осуществлять вручную.

Сначала нам нужно внести некоторые доработки. Они будут связаны с тем, что мы работали только в одной базе и использовали уникальность номеров кодов справочников и номеров документов. Теперь, когда создание новых элементов справочников и новых документов будет происходить в двух базах одновременно и независимо, нам снова требуется обеспечить их уникальность. Для этого в каждой базе к

номерам документов и кодам справочников мы будем добавлять уникальный префикс, однозначно идентифицирующий базу данных.

Для хранения префикса номеров мы используем объект конфигурации *Константа*.

Константа для обмена данными

Объект конфигурации *Константа* предназначен для создания в базе данных таблиц, в которых будет храниться информация, не изменяющаяся во времени или изменяющаяся очень редко.

Каждый такой объект описывает таблицу для хранения одного значения.

Приступим к созданию константы, в которой мы будем хранить значение префикса номеров.

Доработка объектов конфигурации, участвующих в обмене

В режиме Конфигуратор

Откроем конфигуратор под **Администратором** и добавим новый объект конфигурации *Константа* с именем **ПрефиксНумерации**. Определим типа значения константы – **Строка** с фиксированной длиной 2 символа.

Первое, что следует сделать – внести изменения в модули всех объектов, участвующих в обмене (в нашем случае это справочники, документы и планы видов характеристик).

Эти изменения будут заключаться в том, что теперь при формировании номера документа и кода справочника или плана видов характеристик будет использоваться значение константы **ПрефиксНумерации** для обеспечения уникальности номеров и кодов в каждой из наших баз.

Функцию формирования префикса мы вынесем в общий модуль, поскольку не исключена возможность. Что в будущем алгоритм формирования префикса может быть изменен.

Добавим общий модуль **Обмен**. В него поместим функцию:

```
Функция ПолучитьПрефиксНомера() Экспорт

    Возврат Константы.ПрефиксНумерации.Получить();

КонецФункции
```

Как вы видите, эта функция просто возвращает значение константы **ПрефиксНумерации**.

Теперь доработаем справочник **Клиенты**.

Откроем модуль объекта и добавим в него обработчик события **ПриУстановкеНовогоКода**:

```
Процедура ПриУстановкеНовогоКода(СтандартнаяОбработка, Префикс)

	Префикс = Обмен.ПолучитьПрефиксНомера();

КонецПроцедуры
```

Событие **ПриУстановкеНовогоКода** возникает в момент, когда выполняется установка нового кода элемента справочника. Обратите внимание, что мы пишем этот код не в модуле формы, а в модуле объекта, поскольку это событие возникает не для формы, а для объекта в целом.

Вторым параметром вызова обработчика передается префикс, который будет заполнен в данной процедуре и использован системой для генерации кода.

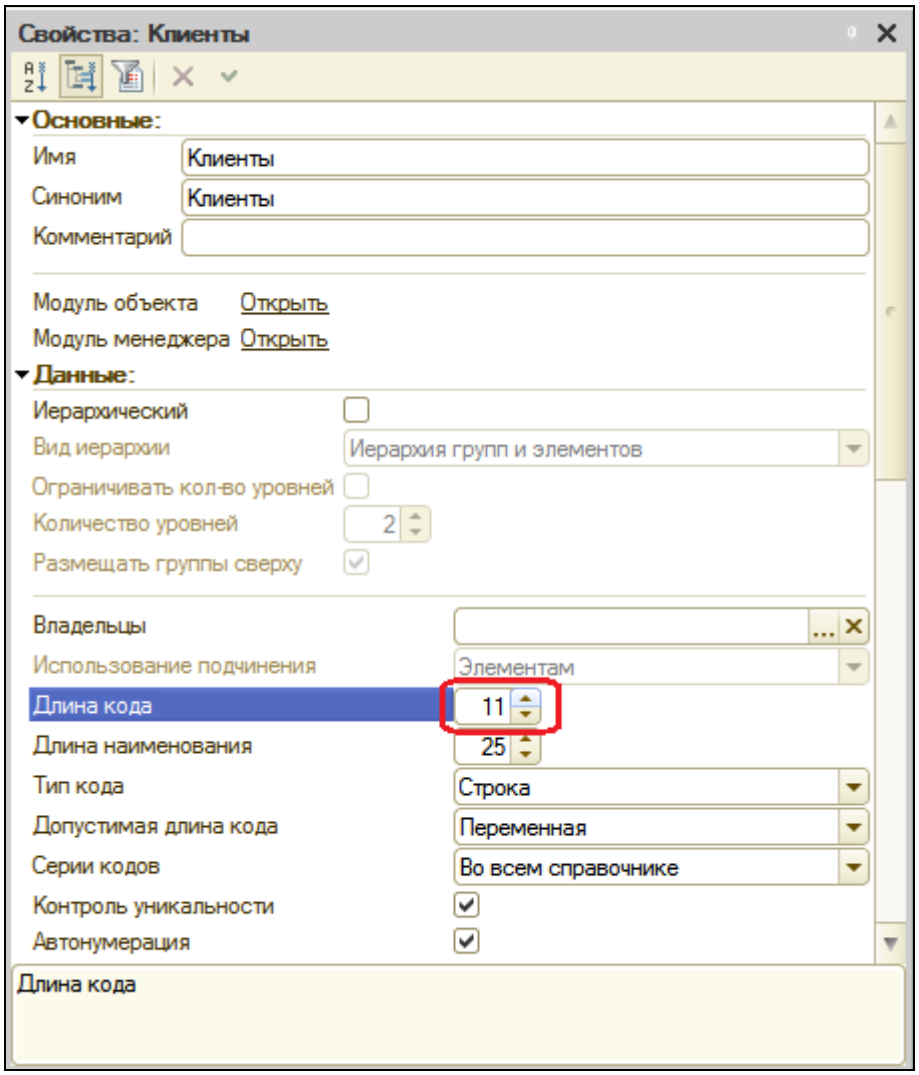
В обработчике события мы вызываем функцию общего модуля. Поскольку модуль неглобальный, то обращаемся к ней по имени модуля и имени функции (Обмен.ПолучитьПрефиксНомера). В этой процедуре мы устанавливаем префикс равным значению константы **ПрефиксНумерации**.

Такие же обработчики нужно будет добавить во все справочники и планы видов характеристик, участвующие в обмене.

В нашем случае это:

- Справочники:
  - **Сотрудники**,
  - **Склады**,
  - **Номенклатура**,
  - **ВариантыНоменклатуры**,
  - **ДополнительныеСвойстваНоменклатуры**,
- План видов характеристик: **СвойстваНоменклатуры**.

После этого у всех этих объектов и у справочника **Клиенты** нужно в свойствах увеличить длину кода до 11 символов.



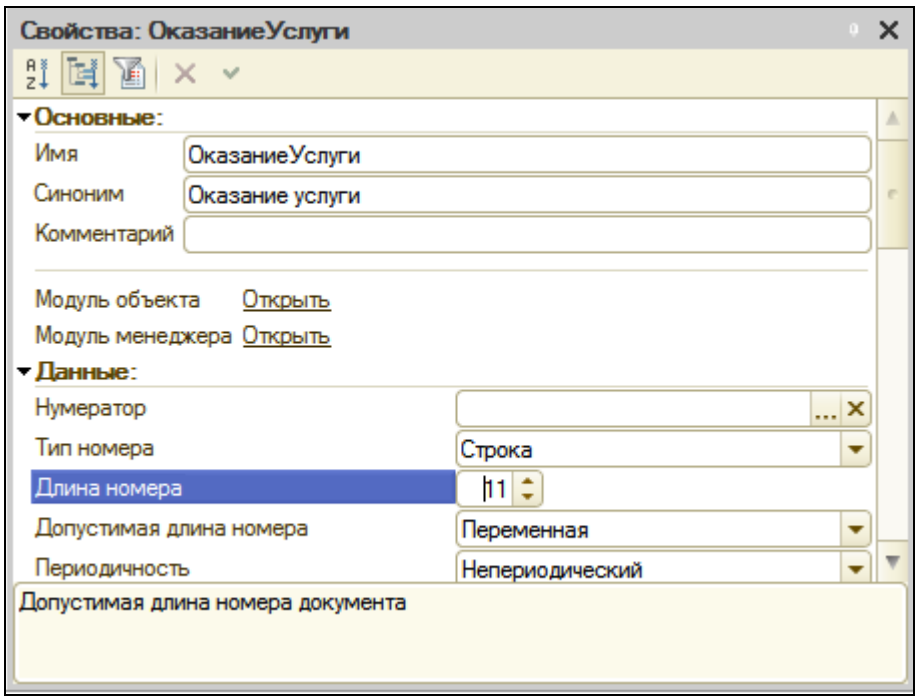
Теперь займемся доработкой документов.

В модуль документа **ПриходнаяНакладная** добавим обработчик события **ПриУстановкеНовогоНомера**:

```
Процедура ПриУстановкеНовогоНомера(СтандартнаяОбработка, Префикс)

	Префикс = Обмен.ПолучитьПрефиксНомера();

КонецПроцедуры
```

Такие же обработчики нужно добавить во все документы, участвующие в обмене. В нашем случае это документ **ОказаниеУслуги**.

После этого в обоих документах в свойствах увеличить длину номера до 11 символов.

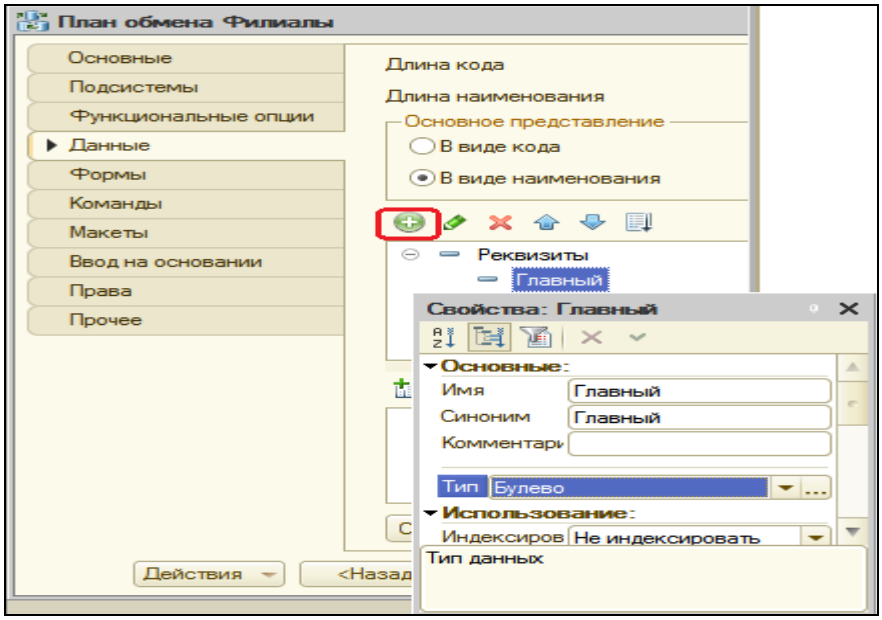


При этом подготовительная работа с существующими объектами конфигурации завершена, и мы можем перейти к созданию процедур обмена данными.

Добавление плана обмена

Раскроем ветвь Общие дерева объектов конфигурации и добавим новый **ПланОбмена** с именем **Филиалы**, представление объекта – **Филиал**.

На закладке **Данные** создадим реквизит плана обмена **Главный** с типом **Булево**.



Этот реквизит понадобится, чтобы разрешать коллизии при обмене данными. Под коллизией понимается ситуация, когда один и тот же объект обмена данными был изменен одновременно в двух узлах.

В этом случае мы будем анализировать значение реквизита **Главный** и принимать изменения только в случае, если они сделаны в главном узле.

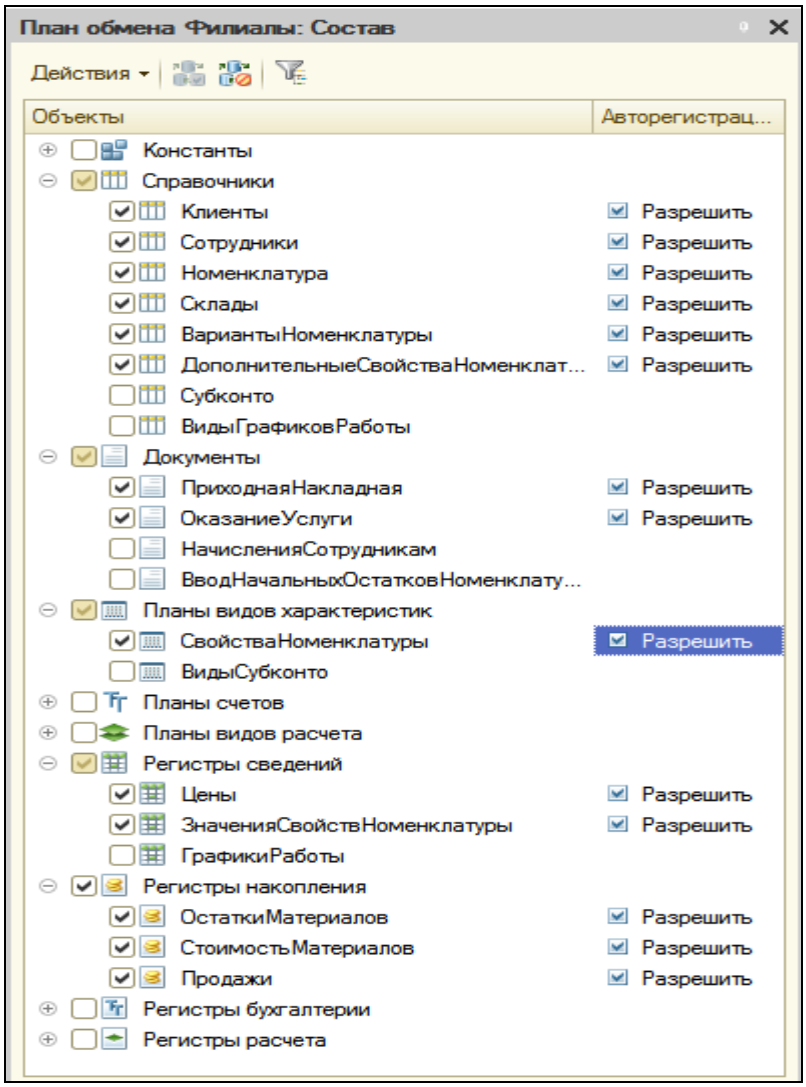
Теперь определим состав объектов, участвующих в обмене.

Для этого на закладке **Основные** нажмем кнопку **Состав**.

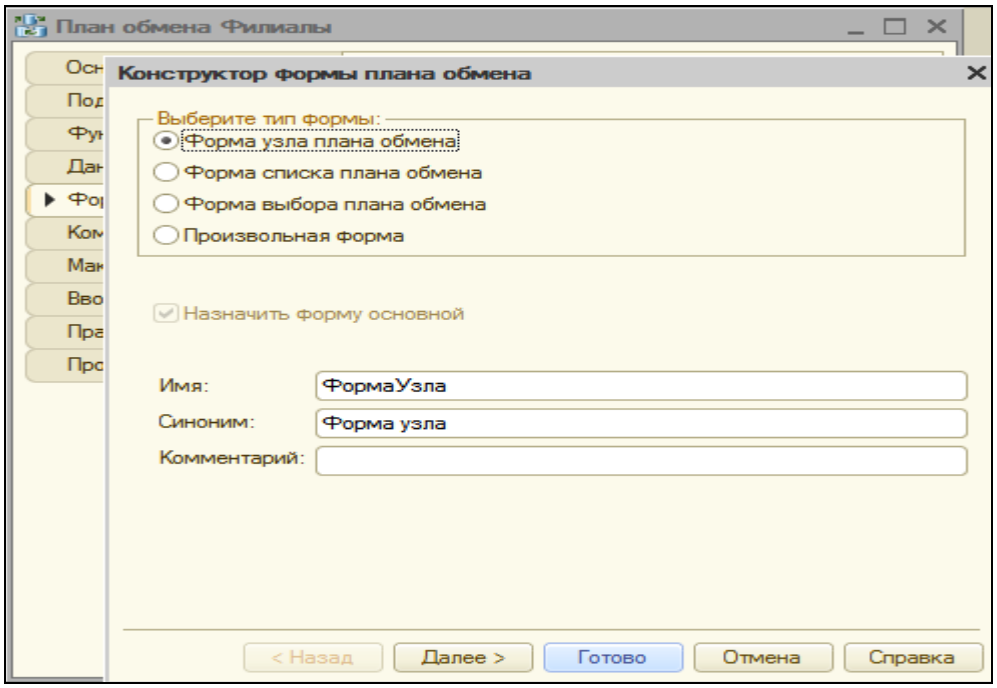
Включим в обмен все объекты, не относящиеся к ведению бухучета и расчету зарплаты.

Обратите внимание, что константа **ПрефиксНумерации** не участвует в обмене, поскольку ее значение должно быть уникальным для каждой базы, участвующей в обмене.

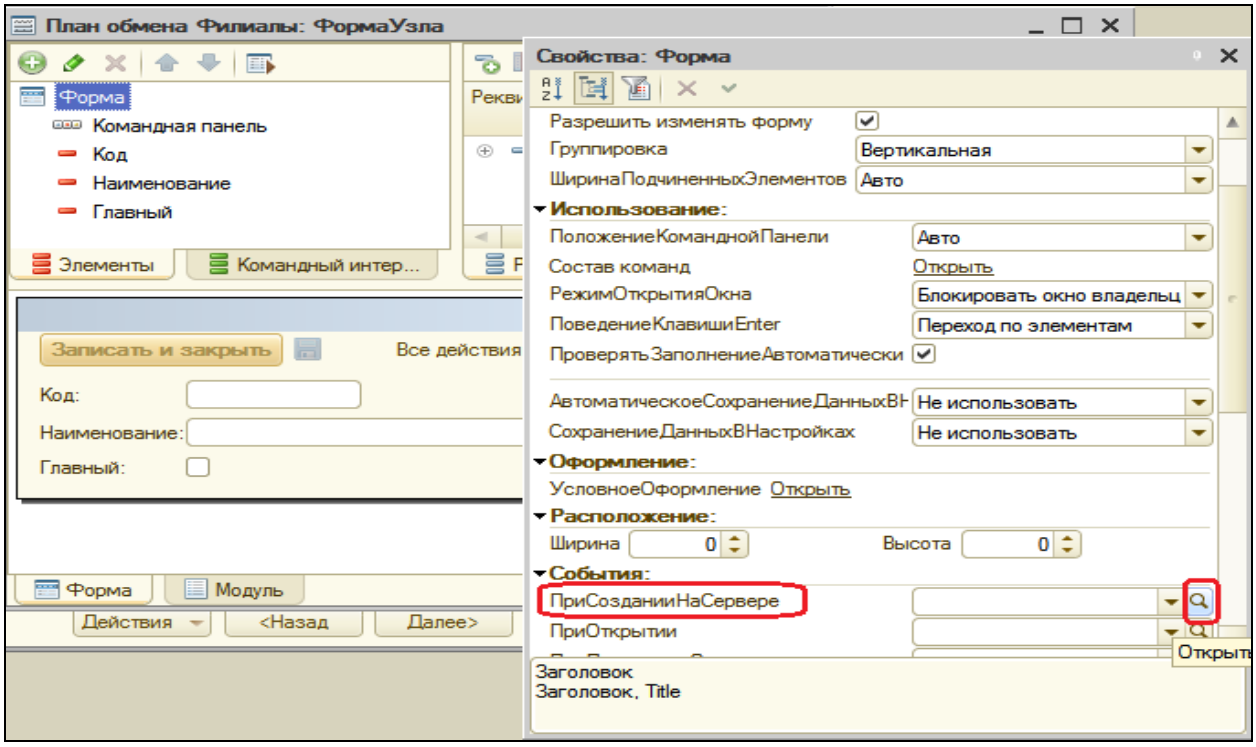
Состав данных обмена должен выглядеть следующим образом:



Теперь на закладке **Формы** нажмем кнопку открытия и с помощью конструктора создадим основную форму узла.



В окне элементов формы выделим корневой элемент **Форма**, вызовем контекстное меню - **Свойства** и создадим обработчик события формы **ПриСозданииНаСервере**. Он понадобится для того, чтобы запретить установку реквизита **Главный** для предопределенного узла, соответствующего данной информационной базе.



```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
        Если Объект.Ссылка = ПланыОбмена.Филиалы.ЭтотУзел() Тогда
        Элементы.Главный.Доступность = Ложь;
КонецЕсли;
КонецПроцедуры
```

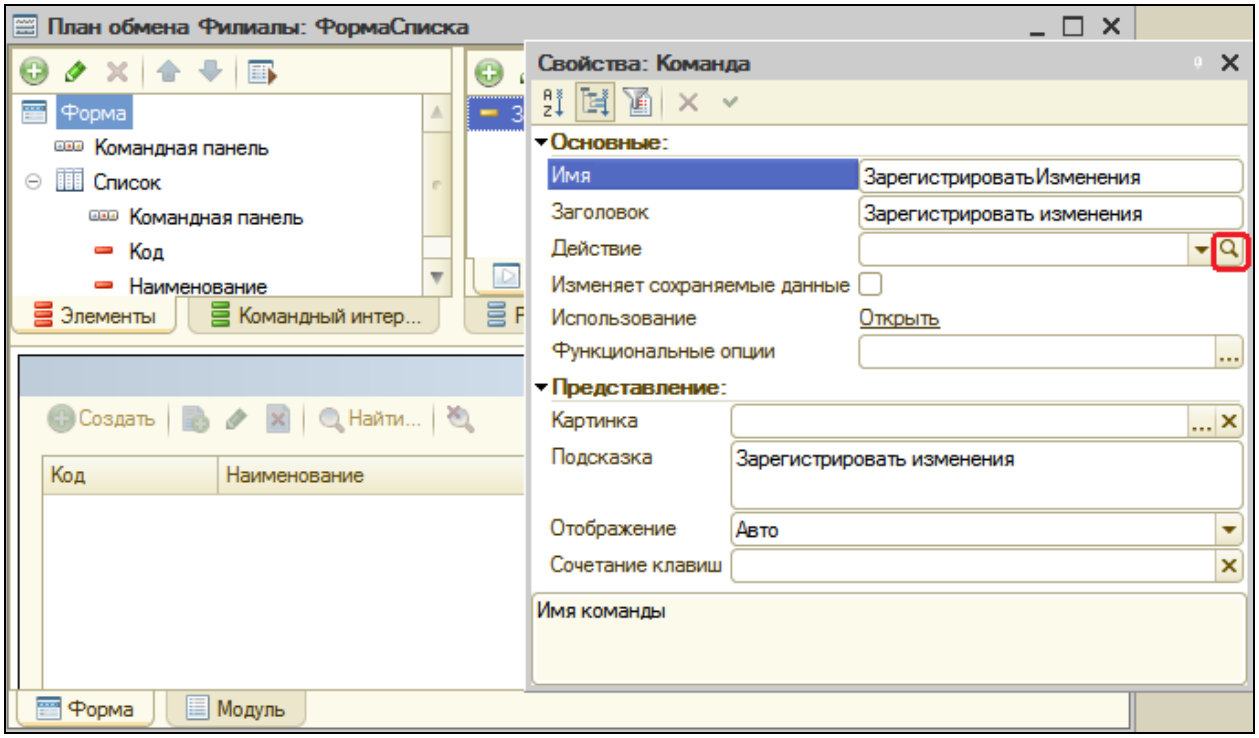
В этой процедуре мы используем метод менеджера плана обмена **ЭтотУзел()**, который возвращает ссылку на узел плана обмена, соответствующий данной информационной базе.

Затем создадим основную форму списка плана обмена, чтобы описать в ней некоторые действия по регистрации нового узла обмена.

Суть этих действий будет заключаться в том, что при регистрации нового узла обмена мы должны будем сформировать для него все необходимые записи регистрации изменений для всех объектов конфигурации, входящих в данный план обмена. Это будет своего рода начальная синхронизация узла обмена всеми данными обмена.

Для этого на закладке **Команды** создадим команду **ЗарегистрироватьИзменения**.

В окне свойств нажмем кнопку открытия  в строке **Действие**.



Иногда при нажатии на кнопку открытия ничего не происходит. В этом случае закройте окно свойств и откройте его заново – кнопка должна работать.

Шаблон обработчика события выполнения этой команды заполним следующим образом:

```
&НаКлиенте
Процедура ЗарегистрироватьИзменения(Команда)
        РегистрацияИзмененийНаСервере(Элементы.Список.ТекущаяСтрока);
КонецПроцедуры
```

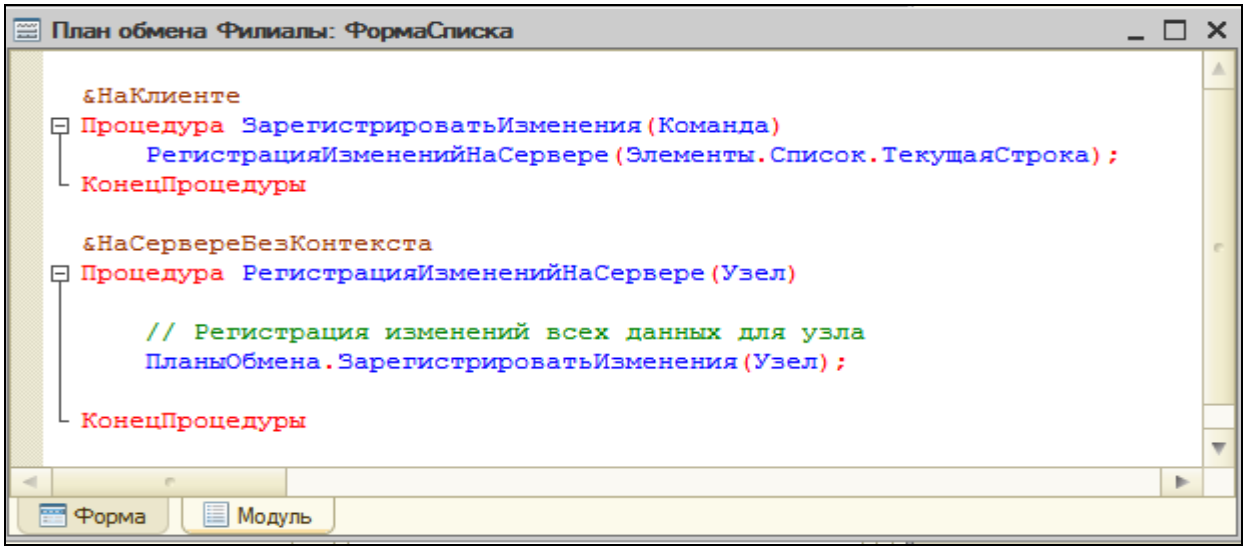
В этом обработчике мы вызываем процедуру **РегистрацияИзмененийНаСервере(),** которую мы напишем в дальнейшем. Она будет выполняться на сервере.

В параметре **Элементы.Список.ТекущаяСтрока** мы передаем в нее ссылку на объект **ПланОбмена.Филиалы**, используя свойство **ТекущаяСтрока** для таблицы **Список** (источником данных для этой таблицы является динамический список узлов плана обмена **Филиалы**).

Саму процедуру **РегистрацияИзмененийНаСервере()** мы предварим директивой компиляции **&НаСервереБезКонтекста**, т.к. процедура работает быстрее, если при ее вызове не передается контекст всей формы. Запишем ее ниже предыдущей процедуры в модуле формы списка:

```
&НаСервереБезКонтекста
Процедура РегистрацияИзмененийНаСервере(Узел)

        // Регистрация изменений всех данных для узла
        ПланыОбмена.ЗарегистрироватьИзменения(Узел);

КонецПроцедуры
```

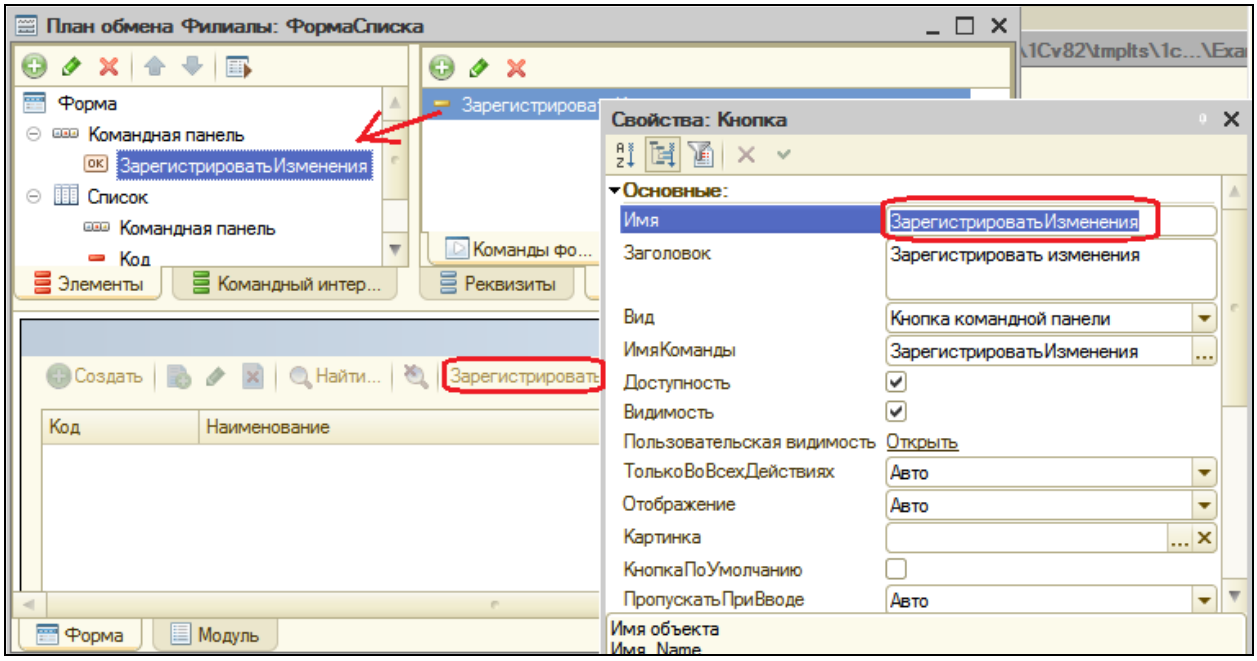
Модуль формы списка плана обмена **Филиалы** будет выглядеть так:



В этой процедуры мы обращаемся к механизму регистрации изменений, вызывая метод менеджера планов обмена – **ЗарегистрироватьИзменения()**. В этот метод передается ссылка на текущий узел плана обмена **Филиалы**.

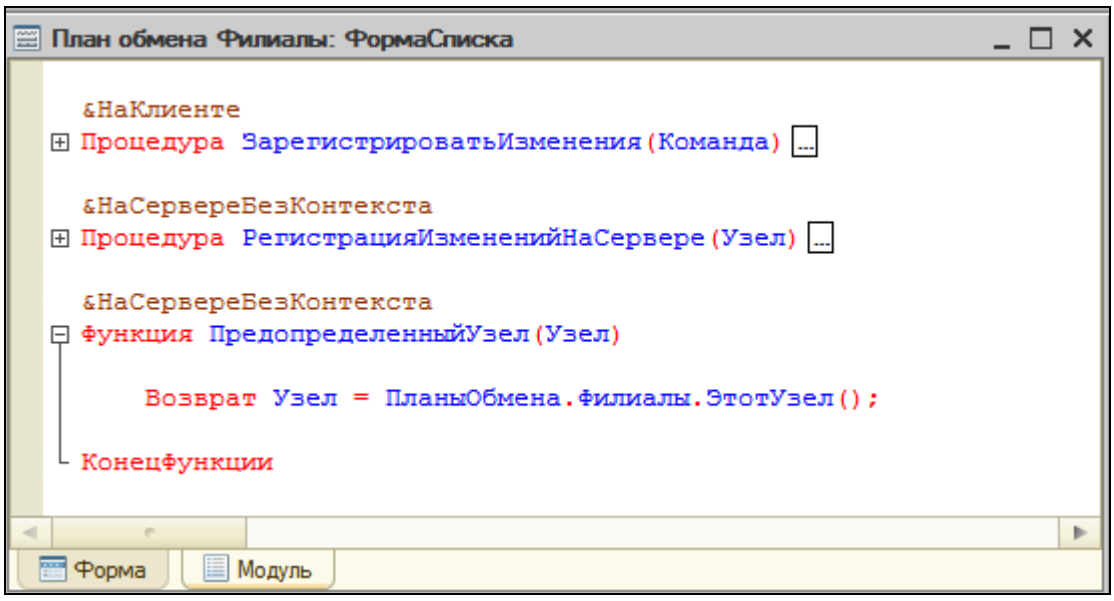
В результате выполнения этой процедуры в информационной базе будут созданы записи регистрации изменений, предназначенные для пересылки в созданный нами узел, для всех объектов обмена, связанных в составе данного плана обмена.

Перейдем на закладку **Форма** и перетащим команду **ЗарегистрироватьИзменения** из окна команд в окно элементов формы в командную панель. Причем кнопка Зарегистрировать изменения должна быть доступна только в случае, если текущий узел является предопределенным для данной информационной базы, иначе регистрация изменений невозможна.

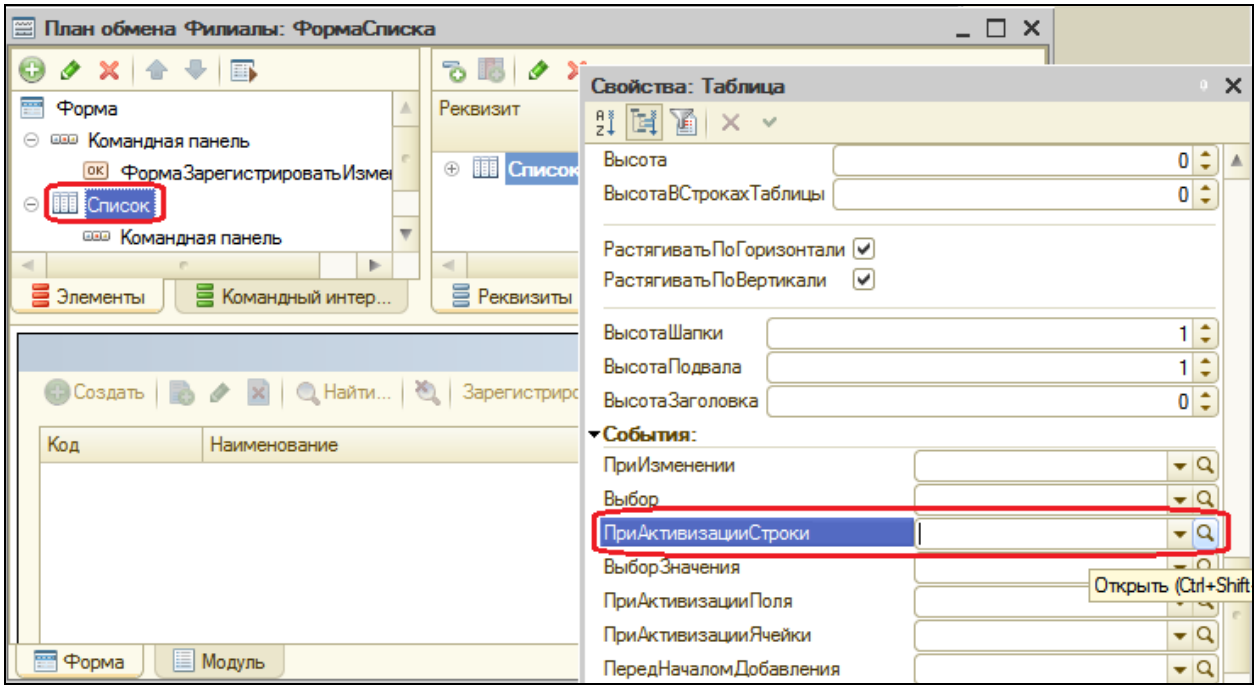


Чтобы обеспечить такое поведение кнопки, создадим в модуле формы списка функцию, выполняющуюся на сервере и возвращающую истину, если переданный в функцию узел является предопределенным.

```
&НаСервереБезКонтекста
Функция ПредопределенныйУзел(Узел)

	Возврат Узел = ПланыОбмена.Филиалы.ЭтотУзел();

КонецФункции
```



Затем в окне элементов формы выделим элемент **Список**, вызовем его свойства и создадим обработчик события **ПриАктивизацииСтроки**:




```
Процедура СписокПриАктивизацииСтроки(Элемент)

        Если ПредопределенныйУзел(Элемент.ТекущаяСтрока) Тогда
        Элементы.ЗарегистрироватьИзменения.Доступность = Ложь;
Иначе
        Элементы.ЗарегистрироватьИзменения.Доступность = Истина;
КонецЕсли;

КонецПроцедуры
```

В этой процедуре доступность кнопки **ЗарегистрироватьИзменения** определяется в зависимости от значения функции **ПредопределенныйУзел()**, в которую передается ссылка на текущий узел (**Элемент.ТекущаяСтрока**).

На этом создание плана обмена завершено, и мы можем перейти непосредственно к созданию процедур обмена данными.

Процедуры обмена данными

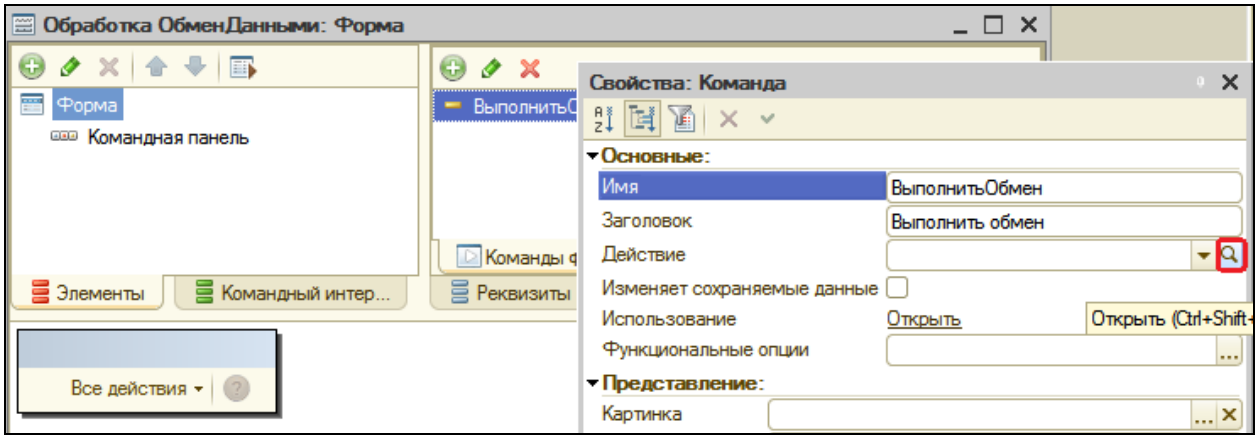
Для начала обмена данными мы используем обработку.

Если вы не хотите детально разбираться в коде, пролистайте до конца раздела, там будет процедура полностью

Добавим новый объект **Обработка** с именем **ОбменДанными**.

На закладке **Формы** создадим основную форму обработки.

В окне редактора форм на закладке **Команды** создадим команду формы **ВыполнитьОбмен**. В строке **Действие** нажмем кнопку открытия и создадим обработчик выполнения этой команды – вызов процедуры **ОбменСФилиалами()**.

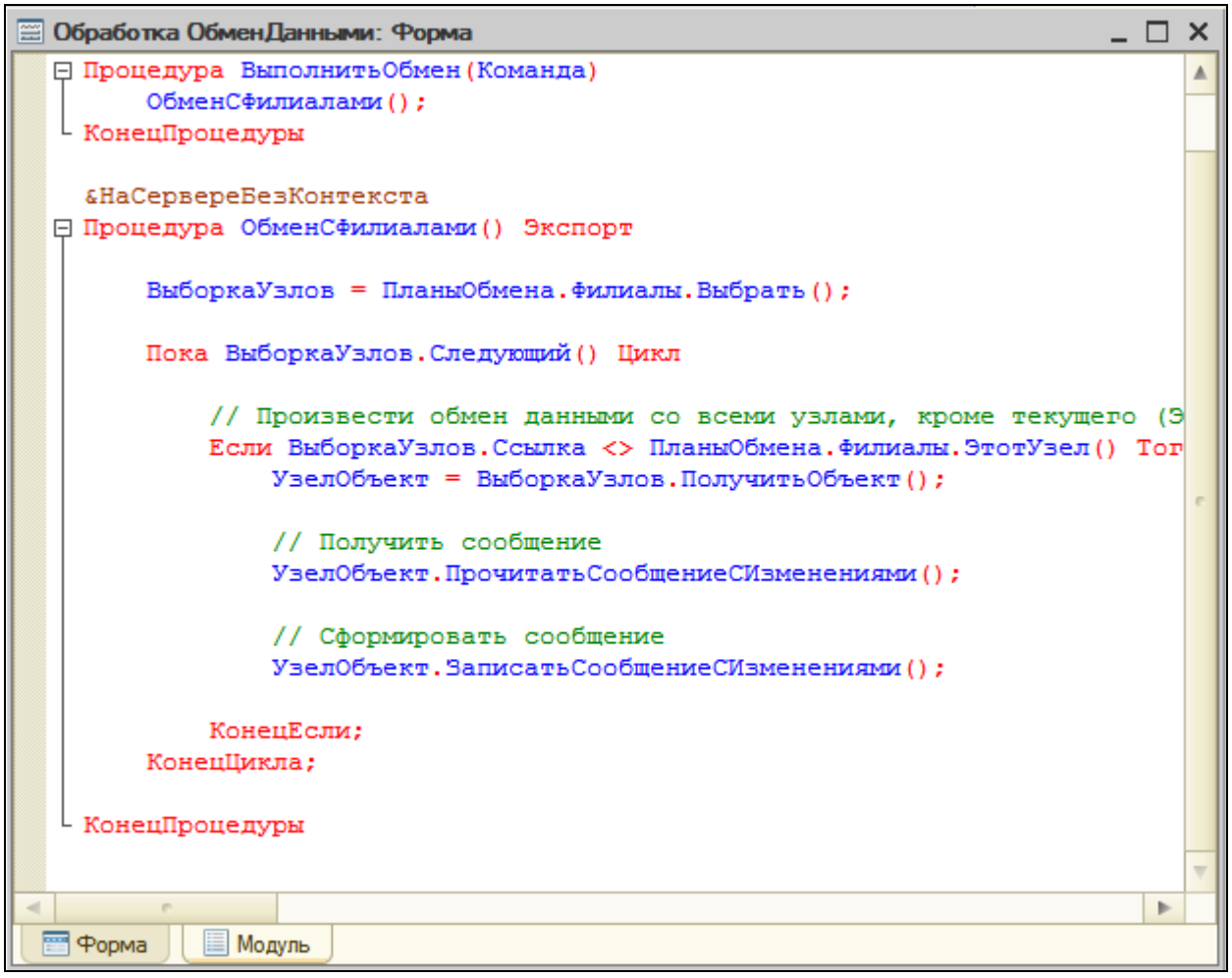



```
&НаКлиенте
Процедура ВыполнитьОбмен(Команда)
        ОбменСФилиалами();
КонецПроцедуры
```

Затем в модуле формы создадим процедуру **ОбменСФилиалами**, выполняющуюся на сервере:

```
&НаСервереБезКонтекста
Процедура ОбменСФилиалами() Экспорт

        ВыборкаУзлов = ПланыОбмена.Филиалы.Выбрать();
```

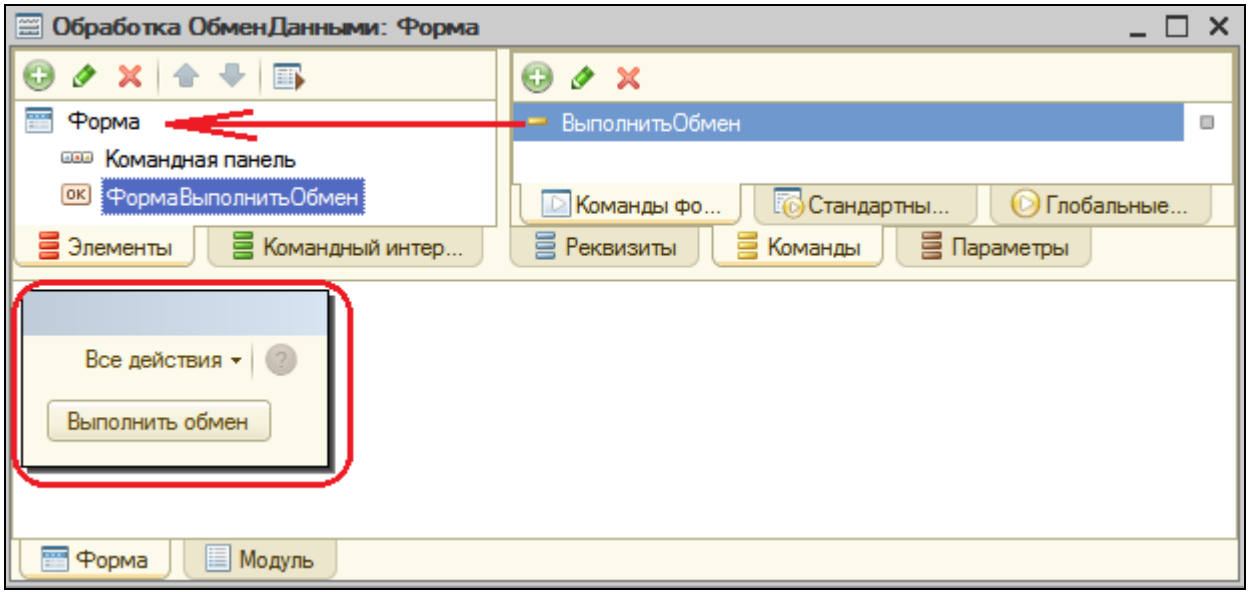
```
        Пока ВыборкаУзлов.Следующий() Цикл

                // Произвести обмен данными со всеми узлами, кроме текущего (ЭтотУзел)
                Если ВыборкаУзлов.Ссылка <> ПланыОбмена.Филиалы.ЭтотУзел() Тогда
                        УзелОбъект = ВыборкаУзлов.ПолучитьОбъект();

                        // Получить сообщение
                        УзелОбъект.ПрочитатьСообщениеСИзменениями();

                        // Сформировать сообщение
                        УзелОбъект.ЗаписатьСообщениеСИзменениями();

                КонецЕсли;
        КонецЦикла;

КонецПроцедуры
```



Алгоритм процедуры в следующем: в цикле мы перебираем узлы, которые содержатся в плане обмена **Филиалы**, и для всех узлов, кроме себя самого, производим сначала чтение сообщений, поступивших из других узлов (процедуру **ПрочитатьСообщенияСИзменениями** мы создадим позднее).

Затем мы формируем для них сообщения, предназначенные для передачи и содержащие измененные данные для этого узла (процедура **ЗаписатьСообщенияСИзменениями** также будет создана позднее).

В заключение перетащим команду **ВыполнитьОбмен** из окна команд в окно элементов формы (перетащить на **Форма**, а на командную панель).



Процедура записи данных

Сами процедуры записи и чтения данных обмена мы разместим в модуле объекта План обмена **Филиалы**.

В окне редактирования этого объекта перейдем на закладку **Прочее** и откроем модуль объекта.

Сначала создадим процедуру, которая используется нами при обмене данными – **ЗаписатьСообщениеСИзменениями**. Создавать ее будем постепенно.

Сначала сформируем имя файла, который будет содержать данные для обмена, и сообщим пользователю о начале и окончании выгрузки данных в узел.

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ---------
---";
	Сообщение.Сообщить();
	Каталог = КаталогВременныхФайлов();

	// Сформировать имя временного файла
```

```
        ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
СокрЛП(ПланыОбмена.
                        Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец выгрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Для упрощения примера мы будем обмениваться сообщениями через каталог временных файлов. Имена сообщений стандартизованы и имеют вид **MessageКодУЗлаОтправителя_КодУзлаПолучателя.xml**.

После этого обратимся к механизмам записи/чтения XML-документов и создадим новый объект – **ЗаписьXML**. С его помощью откроем новый XML-файл для записи, запишем в него объявление XML. В конце процедуры завершим запись и закроем файл:

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ---------
---";
        Сообщение.Сообщить();
        Каталог = КаталогВременныхФайлов();

        // Сформировать имя временного файла
        ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
СокрЛП(ПланыОбмена.
                        Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";

        // Создать объект записи XML
        // *** ЗаписьXML-документов
        ЗаписьXML = Новый ЗаписьXML;
        ЗаписьXML.ОткрытьФайл(ИмяФайла);
        ЗаписьXML.ЗаписатьОбъявлениеXML();
        ЗаписьXML.Закрыть();

        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец выгрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Теперь мы обратимся к механизмам инфраструктуры сообщений и создадим новый объект **ЗаписьСообщенияОбмена**, метод которого **НачатьЗапись()** позволяет кроме всего прочего создать очередной номер сообщения и записать заголовок сообщения в XML. В конце процедуры мы опять закончим запись сообщения.

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ---------
---";
	Сообщение.Сообщить();
	Каталог = КаталогВременныхФайлов();

	// Сформировать имя временного файла
	ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
СокрЛП(ПланыОбмена.
		Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";

	// Создать объект записи XML
	// *** ЗаписьXML-документов
	ЗаписьXML = Новый ЗаписьXML;
	ЗаписьXML.ОткрытьФайл(ИмяФайла);
	ЗаписьXML.ЗаписатьОбъявлениеXML();

	// *** Инфраструктура сообщений
	ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
	ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = " Номер сообщения: " +
ЗаписьСообщения.НомерСообщения;
	Сообщение.Сообщить();
	ЗаписьСообщения.ЗакончитьЗапись();

	ЗаписьXML.Закрыть();

	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = "-------- Конец выгрузки ------------";
	Сообщение.Сообщить();

КонецПроцедуры
```

Поскольку мы находимся в модуле объекта, то мы используем стандартный реквизит **Ссылка** в качестве ссылки на объект План обмена **Филиалы**.

После этого, чтобы получить данные, которые необходимо сохранить в этом файле, мы обратимся к механизму регистрации изменений и получим выборку из записей регистрации изменений, предназначенных данному узлу. При формировании выборки мы передаем вторым параметром номер сообщения.

```
	// *** Инфраструктура сообщений
	ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
	ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;
	Сообщение.Сообщить();
```

```
        // Получить выборку измененных данных
        // *** Механизм регистрации изменений
        ВыборкаИзменений =
ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель
                                        ,ЗаписьСообщения.НомерСообщения);

        ЗаписьСообщения.ЗакончитьЗапись();

        ЗаписьXML.Закрыть();

        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец выгрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Теперь осталось только перебрать выборку записей в цикле и сериализовать их в открытый XML-файл.

```
        // Получить выборку измененных данных
        // *** Механизм регистрации изменений
        ВыборкаИзменений =
ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель
                                        ,ЗаписьСообщения.НомерСообщения);
                Пока ВыборкаИзменений.Следующий() Цикл
                // Записать данные в сообщение *** XML-сериализация
                ЗаписатьXML(ЗаписьXML, ВыборкаИзменений.Получить());
        КонецЦикла;

        ЗаписьСообщения.ЗакончитьЗапись();

        ЗаписьXML.Закрыть();

        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец выгрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

На этом создание процедуры записи данных обмена закончено. Полностью она выглядит так:

```
Процедура ЗаписатьСообщениеСИзменениями() Экспорт

        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Выгрузка в узел " + Строка(ЭтотОбъект) + " ---------
---";
        Сообщение.Сообщить();
        Каталог = КаталогВременныхФайлов();

        // Сформировать имя временного файла
        ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\","", "\") + "Message" +
СокрЛП(ПланыОбмена.
                        Филиалы.ЭтотУзел().Код) + "_" + СокрЛП(Ссылка.Код) + ".xml";
```

```
        // Создать объект записи XML
        // *** ЗаписьXML-документов
        ЗаписьXML = Новый ЗаписьXML;
        ЗаписьXML.ОткрытьФайл(ИмяФайла);
        ЗаписьXML.ЗаписатьОбъявлениеXML();

        // *** Инфраструктура сообщений
        ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
        ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Ссылка);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = " Номер сообщения: " + ЗаписьСообщения.НомерСообщения;
        Сообщение.Сообщить();

        // Получить выборку измененных данных
        // *** Механизм регистрации изменений
        ВыборкаИзменений =
ПланыОбмена.ВыбратьИзменения(ЗаписьСообщения.Получатель
                                        ,ЗаписьСообщения.НомерСообщения);
                Пока ВыборкаИзменений.Следующий() Цикл
                // Записать данные в сообщение *** XML-сериализация
                ЗаписатьXML(ЗаписьXML, ВыборкаИзменений.Получить());
        КонецЦикла;

        ЗаписьСообщения.ЗакончитьЗапись();

        ЗаписьXML.Закрыть();

        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец выгрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Процедура чтения данных

Порядок создания процедуры чтения данных будет таким же, как и ранее. Если вы не хотите детально разбираться в коде, пролистайте до конца раздела, там будет процедура полностью.

Сначала сформируем имя файла, содержащего данные обмена.

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

        Каталог = КаталогВременныхФайлов();

        // Сформировать имя файла
        ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") +
         "Message" + СокрЛП(Ссылка.Код) + "_" +
СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + ".xml";
        Файл = Новый Файл(ИмяФайла);
        Если Не Файл.Существует() Тогда
                Возврат;
        КонецЕсли;
```

```
        УдалитьФайлы(ИмяФайла);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец загрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Мы формируем имя файла, которое надеемся найти в этом каталоге, а затем, создав новый объект **Файл** с таким именем проверяем, существует ли он. Если такого файла нет, мы завершаем работу процедуры. Если же он найден, нужно будет удалить его после того, как все данные, содержащиеся в нем, будут обработаны.

Теперь добавим в процедуру команды чтения найденного файла с данными обмена.

```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

        Каталог = КаталогВременныхФайлов();

        // Сформировать имя файла
        ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") +
         "Message" + СокрЛП(Ссылка.Код) + "_" +
СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + ".xml";
        Файл = Новый Файл(ИмяФайла);
        Если Не Файл.Существует() Тогда
                Возврат;
        КонецЕсли;
        // *** Чтение документов XML
        // Попытаться открыть файл
        ЧтениеXML = Новый ЧтениеXML;
        Попытка
                ЧтениеXML.ОткрытьФайл(ИмяФайла);
        Исключение
                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст = "Невозможно открыть файл обмена данными.";
                Сообщение.Сообщить();
      Возврат;
        КонецПопытки;
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ------
------";
        Сообщение.Сообщить();
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = " – Считывается файл " + ИмяФайла;
        Сообщение.Сообщить();

        ЧтениеXML.Закрыть();

        УдалитьФайлы(ИмяФайла);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец загрузки ------------";
        Сообщение.Сообщить();
```

```
КонецПроцедуры
```

Именно в этот момент мы обращаемся к механизмам записи/чтения документов XML, которые работают с ними на базовом уровне.

Для этого мы создаем новый объект **ЧтениеXML**, с помощью которого открываем найденный файл для чтения. В случае успеха мы выводим сообщение о начале загрузки данных из файла. В конце процедуры мы также прекращаем чтение XML-данных из файла методом **Закрыть()**.

Полученные таким образом данные должны являться некоторым сообщением обмена данными. Для того чтобы предоставить их в терминах сообщений, мы добавим в процедуру следующий код:

```
// *** Чтение документов XML
        // Попытаться открыть файл
        ЧтениеXML = Новый ЧтениеXML;
        Попытка
                ЧтениеXML.ОткрытьФайл(ИмяФайла);
        Исключение
                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст = "Невозможно открыть файл обмена данными.";
                Сообщение.Сообщить();
      Возврат;
        КонецПопытки;
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ------------";
        Сообщение.Сообщить();
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = " – Считывается файл " + ИмяФайла;
        Сообщение.Сообщить();

        // Загрузить из найденного файла
        // *** Инфраструктура сообщений
        ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

        // Читать заголовок сообщения обмена данными – файла XML
        ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

        ЧтениеСообщения.ЗакончитьЧтение();

        ЧтениеXML.Закрыть();

        УдалитьФайлы(ИмяФайла);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец загрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Здесь мы обращаемся к механизмам инфраструктуры сообщений планов обмена и создаем объект **ЧтениеСообщенияОбмена**. Используя метод

этого объекта **НачатьЧтение()**, мы считываем заголовок XML-сообщения, в котором содержится в том числе информация об отправителе сообщения. После того, как всё сообщение будет нами обработано, мы заканчиваем чтение.

Теперь, когда мы представили данные обмена в виде сообщения и получили его заголовок, можно произвести одну проверку перед тем, как начать собственно обрабатывать данные.

```
// Загрузить из найденного файла
        // *** Инфраструктура сообщений
        ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

        // Читать заголовок сообщения обмена данными – файла XML
        ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

        // Сообщение предназначено не для этого узла
        Если ЧтениеСообщения.Отправитель <> Ссылка Тогда
ВызватьИсключение "Неверный узел";
        КонецЕсли;


        ЧтениеСообщения.ЗакончитьЧтение();

        ЧтениеXML.Закрыть();

        УдалитьФайлы(ИмяФайла);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец загрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Мы проверяем, является ли отправитель сообщения тем узлом, для которого мы в данном вызове этой процедуры производим обмен данными.

Если все в порядке, то перед тем как начать чтение данных, следует удалить все записи регистрации изменений, которые были сделаны для этого узла и соответствовали номерам сообщений как номер принятого. Это делается затем, чтобы исключить дублирование данных, которые уже были ранее посланы этому узлу и им обработаны.

```
// Сообщение предназначено не для этого узла
        Если ЧтениеСообщения.Отправитель <> Ссылка Тогда ВызватьИсключение
"Неверный узел";
        КонецЕсли;
    // Удаляем регистрацию изменений для узла отправителя сообщения.
        // *** Служба регистрации изменений
        ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправ
итель,ЧтениеСообщения.НомерПринятого);
```

```
        ЧтениеСообщения.ЗакончитьЧтение();
```

Здесь мы обращаемся к службе регистрации изменений и используем метод **УдалитьРегистрациюИзменений()** для выполнения описанных действий.

Теперь можем приступить к непосредственно чтению самих данных, содержащихся в сообщении.

```
        // Сообщение предназначено не для этого узла
        Если ЧтениеСообщения.Отправитель <> Ссылка Тогда ВызватьИсключение
"Неверный узел";
        КонецЕсли;
    // Удаляем регистрацию изменений для узла отправителя сообщения.
        // *** Служба регистрации изменений
        ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,Чт
енияСообщения.НомерПринятого);
                // Читаем данные из сообщения *** XML-сериализация
        Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл

        КонецЦикла;


        ЧтениеСообщения.ЗакончитьЧтение();

        ЧтениеXML.Закрыть();
```

Чтение данных выполняется в цикле, причем мы снова обращаемся к механизмам XML-сериализации и методом глобального контекста **ВозможностьЧтенияXML()** получаем очередной тип данных XML из объекта **ЧтениеXML** и определяем, имеется ли соответствующий тип 1С:Предприятия. В случае успеха выполнение цикла продолжается.

Первое, что нам нужно сделать – представить данные XML в виде некоторого значения, имеющего тип 1С:Предприятия. Для этого мы используем метод глобального контекста **ПрочитатьXML()**.

```
        // Читаем данные из сообщения *** XML-сериализация
        Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
    // Читаем очередное значение
        Данные = ПрочитатьXML(ЧтениеXML);

        КонецЦикла;
```

В результате выполнения этого метода переменная Данные будет содержать объект 1С:Предприятия, соответствующий данным XML.

Теперь следует разрешить возможную коллизию.

```
        // Читаем данные из сообщения *** XML-сериализация
        Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
    // Читаем очередное значение
        Данные = ПрочитатьXML(ЧтениеXML);

        // Не переносим изменение, полученное в главный из неглавного, если
есть регистрация изменения
                Если Не ЧтениеСообщения.Отправитель.Главный И

        ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправит
ель, Данные) Тогда
                        Сообщение = Новый СообщениеПользователю;
                        Сообщение.Текст = " – Изменения отклонены";
                        Сообщение.Сообщить();
          Продолжить;
                КонецЕсли;


        КонецЦикла;
```

Возможная коллизия решается следующим образом: мы проверяем, является ли узел-отправитель главным узлом и есть ли записи об изменении этого объекта для данного узла в нашей БД. Если объект изменялся в нашей базе и отправитель не является главным узлом, мы отклоняем запись полученного объекта. Во всех остальных случаях мы принимаем изменения полученного объекта.

Теперь единственное, что нам осталось сделать – записать полученные данные.

```
        // Не переносим изменение, полученное в главный из неглавного, если есть
регистрация изменения
                Если Не ЧтениеСообщения.Отправитель.Главный И

        ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправитель,
Данные) Тогда
                        Сообщение = Новый СообщениеПользователю;
                        Сообщение.Текст = " – Изменения отклонены";
                        Сообщение.Сообщить();
          Продолжить;
                КонецЕсли;

// Записать полученные данные
                Данные.ОбменДанными.Отправитель =
ЧтениеСообщения.Отправитель;
                Данные.ОбменДанными.Загрузка = Истина;
                Данные.Записать();

        КонецЦикла;
```

Перед записью полученного объекта мы устанавливаем у него в параметрах обмена данными узел отправителя для того, чтобы система

при записи этого объекта в нашей базе данных не формировала записи регистрации изменений этого объекта для того узла, от которого мы его только что получили.

Кроме этого, в параметрах обмена мы устанавливаем свойство Загрузка, информирующее систему о том, что запись объекта будет происходить в режиме обновления данных, полученных в результате обмена. Такое указание позволяет системе упростить процедуру записи объекта, отказавшись от ряда стандартных проверок и исключив изменения связанных данных, которые выполняются при обычной записи.

На этом создание процедуры получения и обработки данных обмена закончено. Полностью процедура будет выглядеть так:

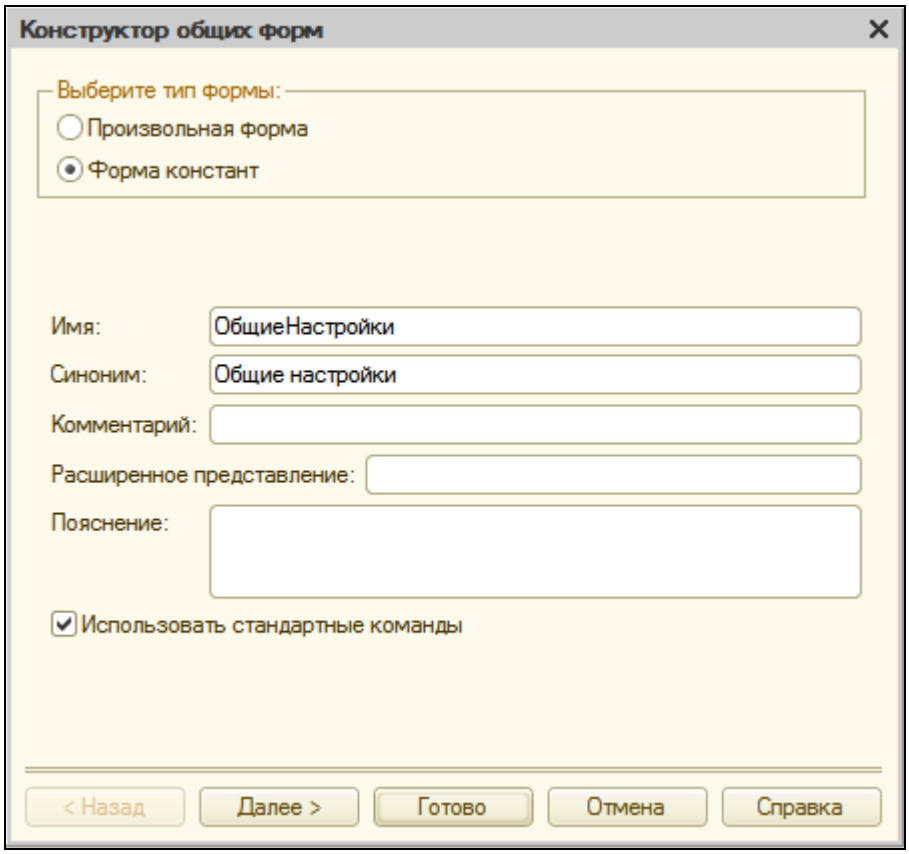
```
Процедура ПрочитатьСообщениеСИзменениями() Экспорт

	Каталог = КаталогВременныхФайлов();

	// Сформировать имя файла
	ИмяФайла = Каталог + ?(Прав(Каталог, 1) = "\", "", "\") +
	 "Message" + СокрЛП(Ссылка.Код) + "_" +
СокрЛП(ПланыОбмена.Филиалы.ЭтотУзел().Код) + ".xml";
	Файл = Новый Файл(ИмяФайла);
	Если Не Файл.Существует() Тогда
		Возврат;
	КонецЕсли;
	// *** Чтение документов XML
	// Попытаться открыть файл
	ЧтениеXML = Новый ЧтениеXML;
	Попытка
		ЧтениеXML.ОткрытьФайл(ИмяФайла);
	Исключение
		Сообщение = Новый СообщениеПользователю;
		Сообщение.Текст = "Невозможно открыть файл обмена данными.";
		Сообщение.Сообщить();
      Возврат;
	КонецПопытки;
	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = "-------- Загрузка из " + Строка(ЭтотОбъект) + " ----------";
	Сообщение.Сообщить();
	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = " – Считывается файл " + ИмяФайла;
	Сообщение.Сообщить();

	// Загрузить из найденного файла
	// *** Инфраструктура сообщений
	ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();

	// Читать заголовок сообщения обмена данными – файла XML
	ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

	// Сообщение предназначено не для этого узла
```

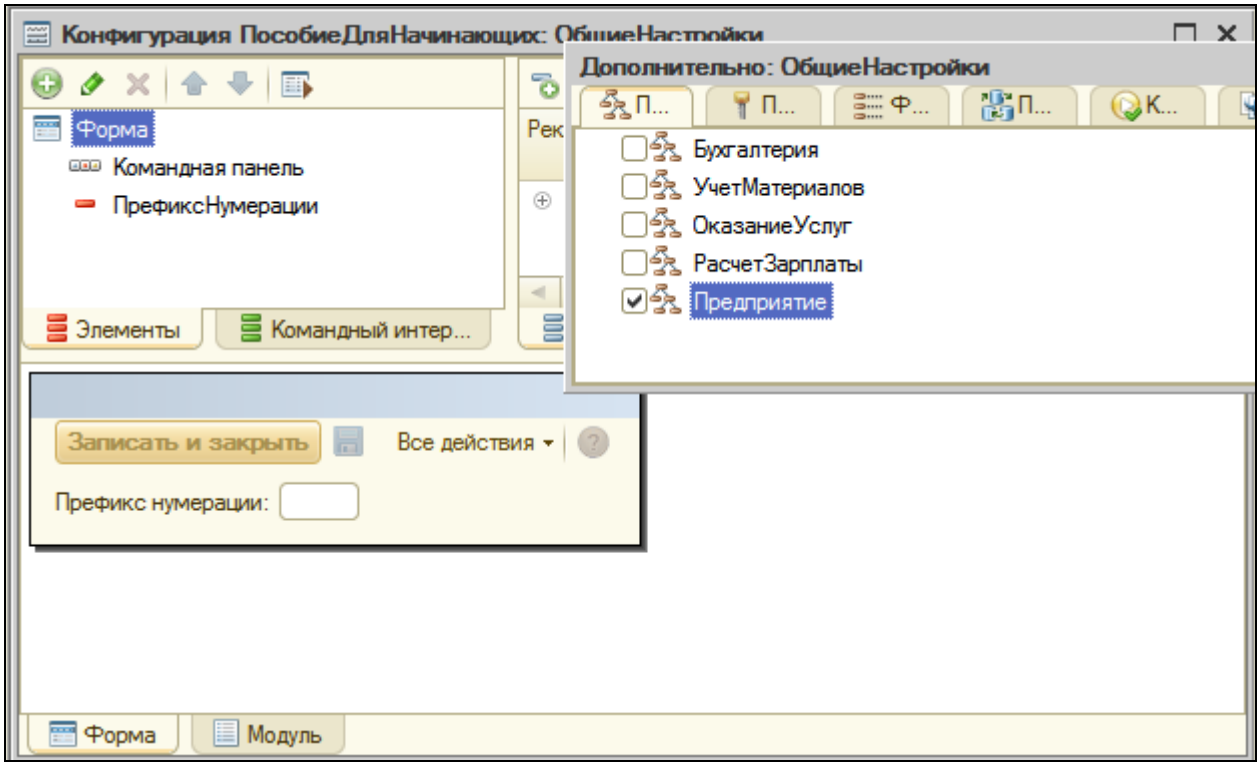
```
        Если ЧтениеСообщения.Отправитель <> Ссылка Тогда ВызватьИсключение
"Неверный узел";
        КонецЕсли;
    // Удаляем регистрацию изменений для узла отправителя сообщения.
        // *** Служба регистрации изменений
        ПланыОбмена.УдалитьРегистрациюИзменений(ЧтениеСообщения.Отправитель,Чт
ениеСообщения.НомерПринятого);
         // Читаем данные из сообщения *** XML-сериализация
        Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
     // Читаем очередное значение
        Данные = ПрочитатьXML(ЧтениеXML);

        // Не переносим изменение, полученное в главный из неглавного, если есть
регистрация изменения
                Если Не ЧтениеСообщения.Отправитель.Главный И

        ПланыОбмена.ИзменениеЗарегистрировано(ЧтениеСообщения.Отправитель,
Данные) Тогда
                        Сообщение = Новый СообщениеПользователю;
                        Сообщение.Текст = " – Изменения отклонены";
                        Сообщение.Сообщить();
          Продолжить;
                КонецЕсли;

// Записать полученные данные
                Данные.ОбменДанными.Отправитель = ЧтениеСообщения.Отправитель;
                Данные.ОбменДанными.Загрузка = Истина;
                Данные.Записать();

        КонецЦикла;


        ЧтениеСообщения.ЗакончитьЧтение();

        ЧтениеXML.Закрыть();

        УдалитьФайлы(ИмяФайла);
        Сообщение = Новый СообщениеПользователю;
        Сообщение.Текст = "-------- Конец загрузки ------------";
        Сообщение.Сообщить();

КонецПроцедуры
```

Проверка работы обмена данными

Чтобы иметь возможность редактировать константу **ПрефиксНумерации**, раскроем ветвь **Общие**, выделим ветвь Общие формы и с помощью конструктора форм создадим форму констант с именем **ОбщиеНастройки**.

Откроем окно свойств **Дополнительно** для этой формы (контекстное меню – **Дополнительно**) и укажем принадлежность этой формы к подсистеме **Предприятие**.



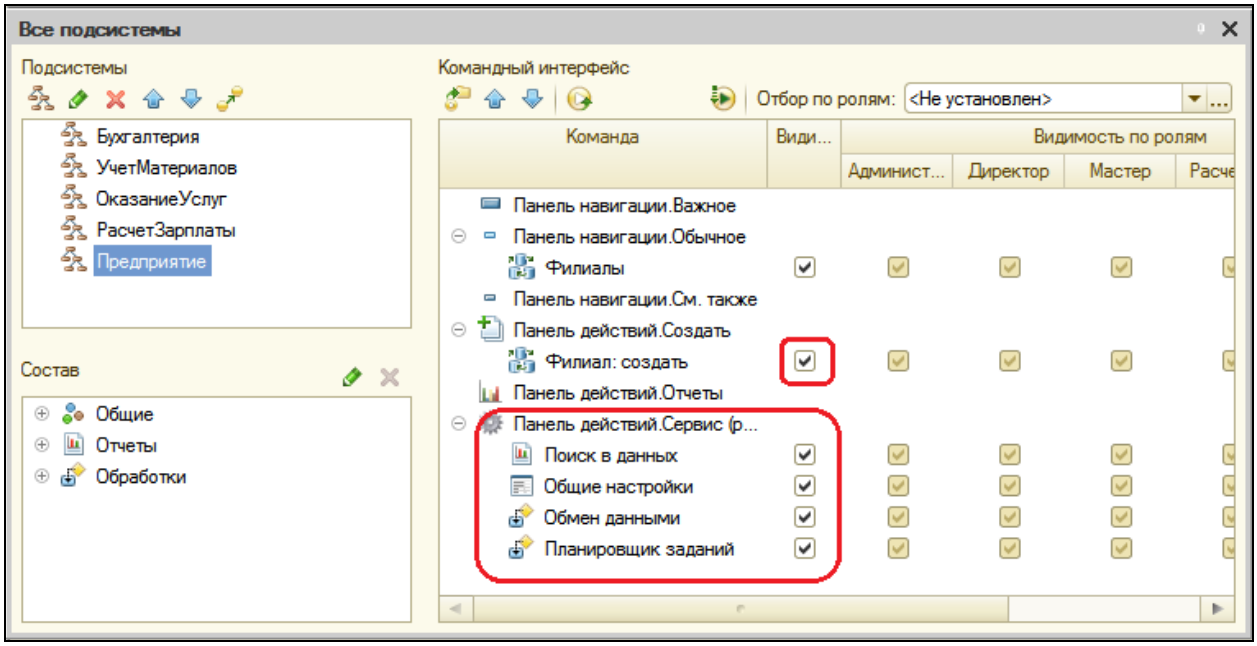


В окне редактирования плана обмена **Филиалы** и обработки **ОбменДанными** на закладке **Подсистемы** также укажем их принадлежность к подсистеме **Предприятие**.

Т.о. доступ к командам открытия плана обмена, обработки, а также формы констант будет иметь только Администратор, т.к. подсистема Предприятие будет доступна только для роли **Администратор**.

В окне редактора командного интерфейса подсистем **Все подсистемы** включим видимость у команды **Филиал: создать** в группе панели действий **Создать** подсистемы **Предприятие** и установим следующий порядок следования команд в группе панели действий **Сервис**:

- **Поиск в данных**,
- **Общие настройки**,
- **Обмен данными**,
- **Планировщик заданий**.



И в заключение создадим новый каталог на жестком диске, в котором будет размещаться база нашего филиала.

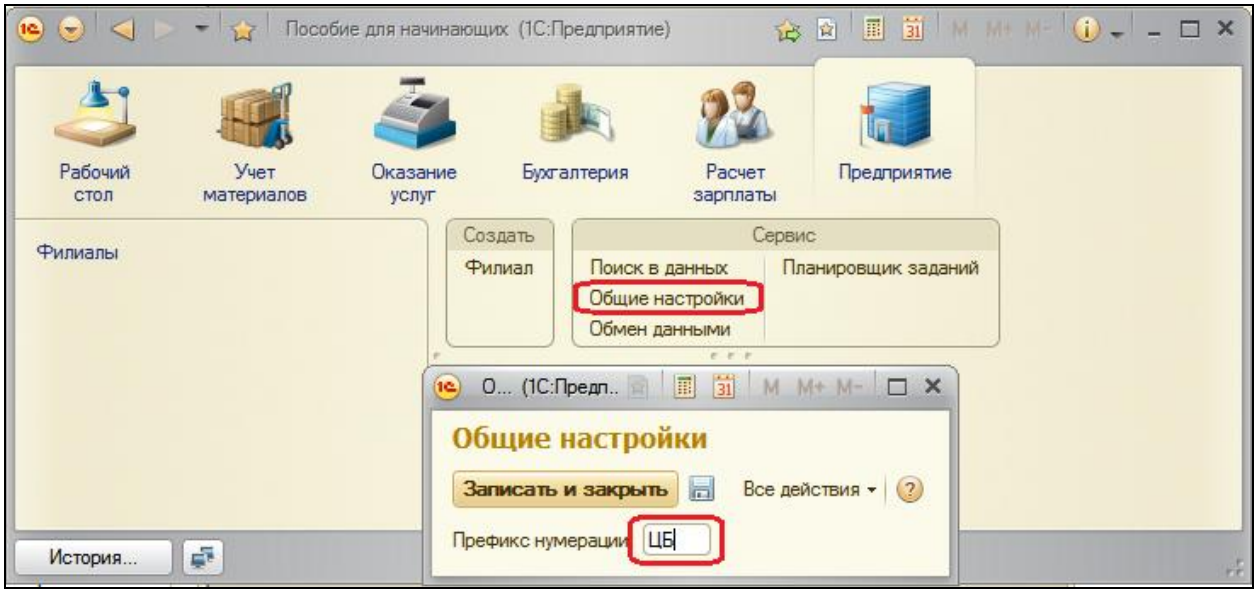
Обновим конфигурацию базы данных (F7). Затем сохраним в созданный каталог нашу конфигурацию, выполнив команду главного меню **Конфигурация – Сохранить конфигурацию в файл...**

В режиме 1С:Предприятие

Запустим режим отладки и установим необходимые значения в нашей центральной базе.

Прежде всего, зададим значение константы **Префикс нумерации – ЦБ**.

Для этого выполним команду **Общие настройки** в панели действий раздела **Предприятие**. Нажмем **Записать и закрыть**.

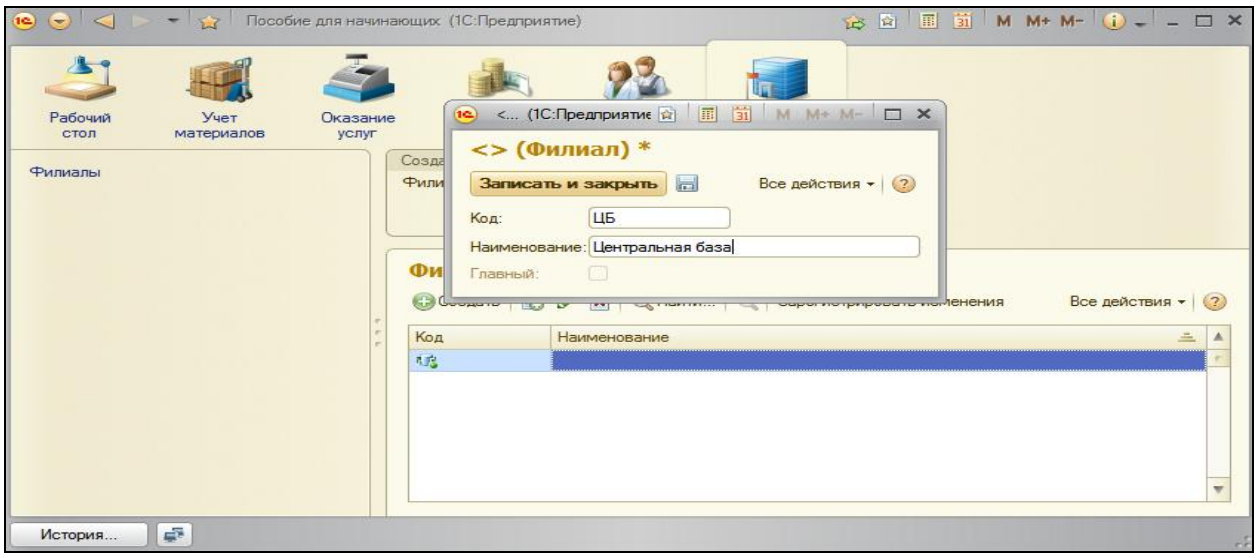


После этого откроем план обмена **Филиалы** и зададим параметры узла по умолчанию, т.е. параметры нашей базы.

Для этого выполним команду **Филиалы** в панели навигации раздела **Предприятие**. В списке планов обмена уже присутствует одна запись. Откроем и отредактируем ее.

Код базы будет **ЦБ**, а наименование – **Центральная база**.

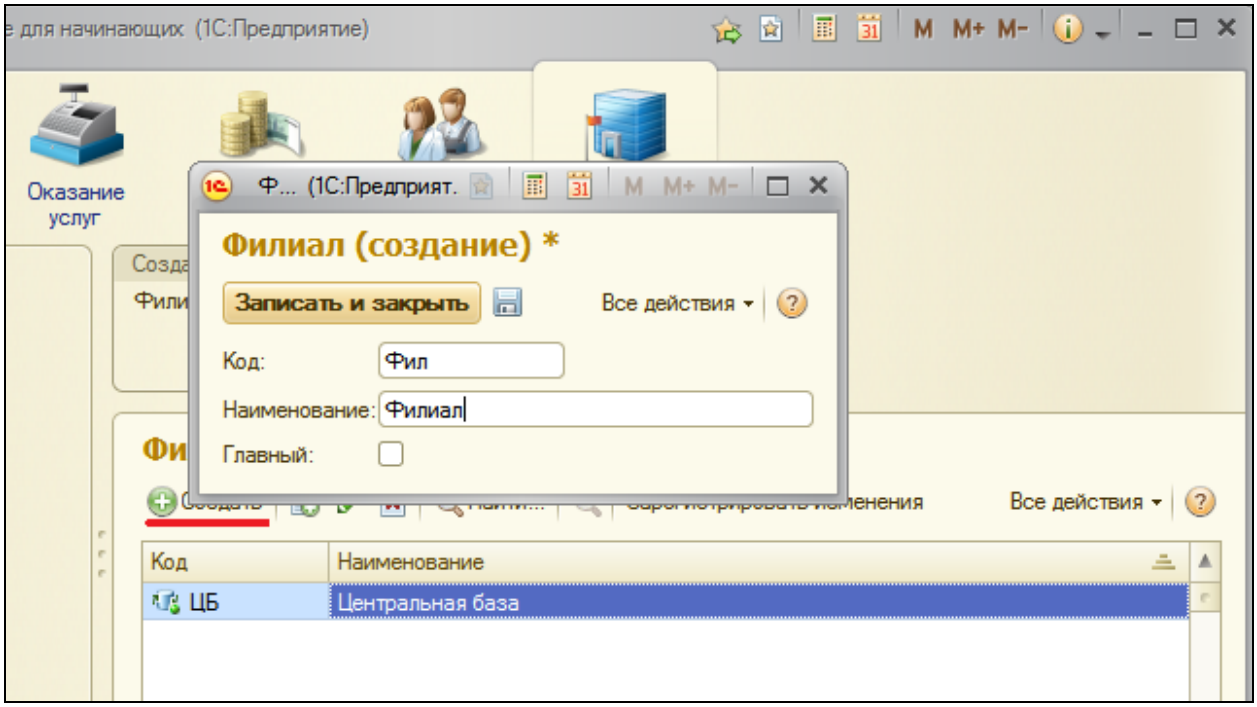
Не забудьте, что именно код идентифицирует узлы обмена в различных базах, поэтому в базе филиала мы будем создавать узлы с такими же кодами.



Нажмите **Записать и закрыть**.

Затем нажмем кнопку **Создать** или команду **Филиал** в панели действий.

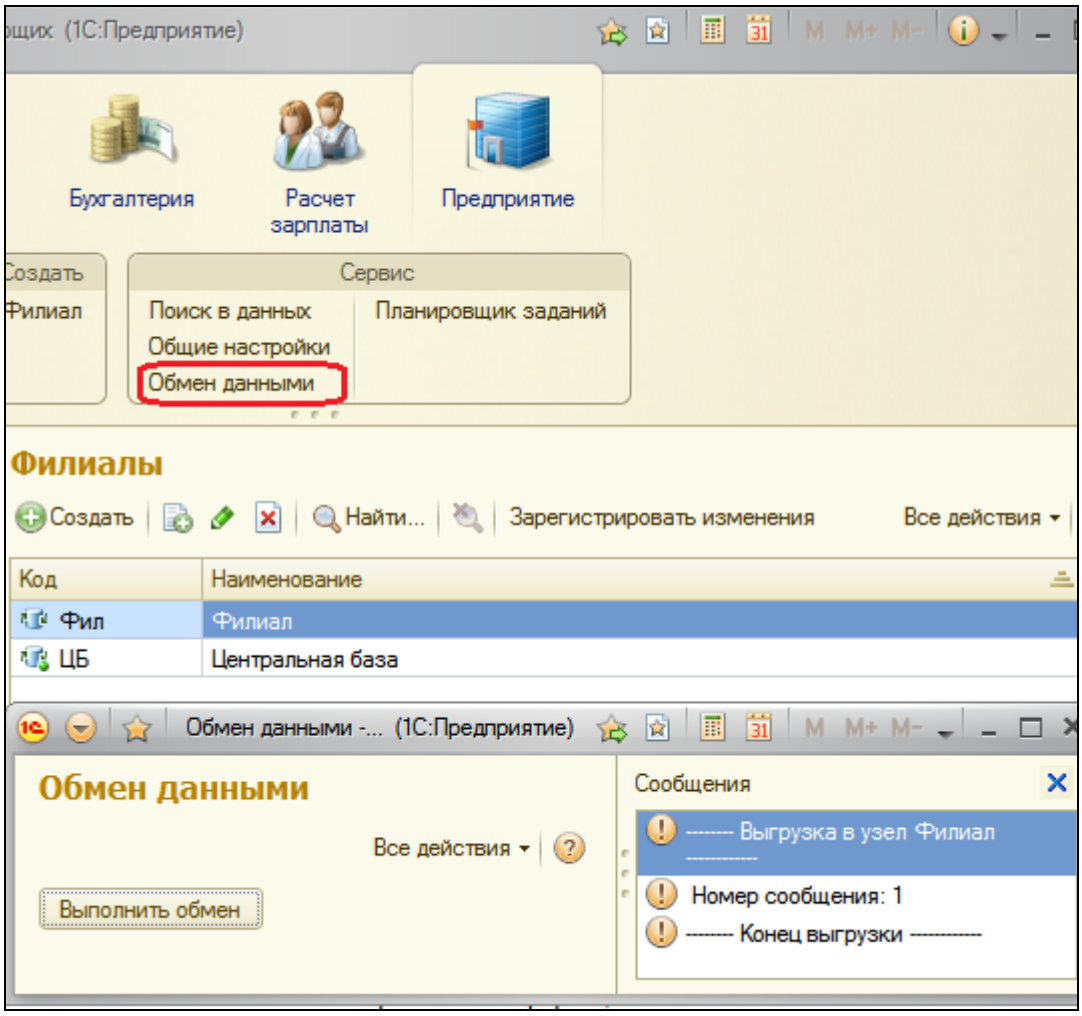
Создадим новый узел, который будет соответствовать базе филиала, присвоим ему код **Фил** и наименование **Филиал**.



Обратите внимание, что предопределенный узел нашей информационной базы (**Центральная база**) выделен в списке узлов обмена специальной пиктограммой. Кнопка **Зарегистрировать изменения** недоступна для этого узла.

Выделим в списке новый узел **Филиал** и нажмем кнопку **Зарегистрировать изменения**.

Теперь вызовем обработку **ОбменДанными** и нажмем **Выполнить обмен**. В окне сообщения появится следующий текст.



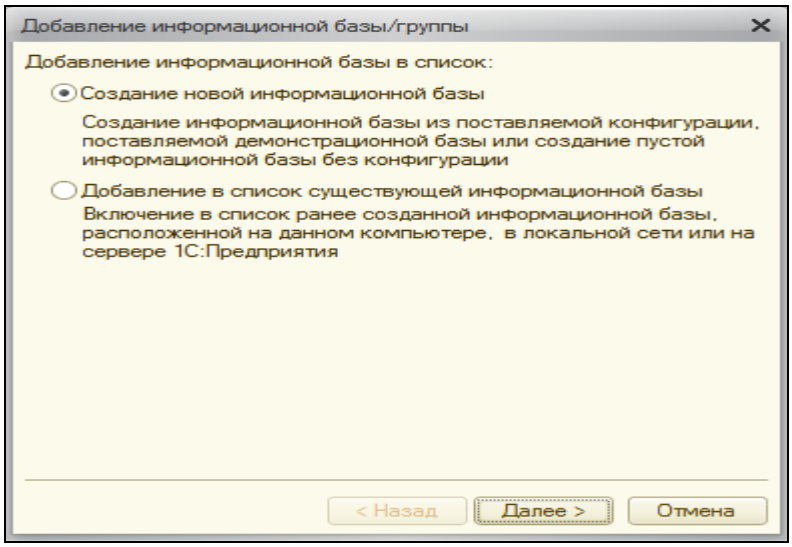
Таким образом, в результате обмена данными центральная база сформировала файл обмена, содержащий изменения всех данных, которыми она обменивается с филиалом.

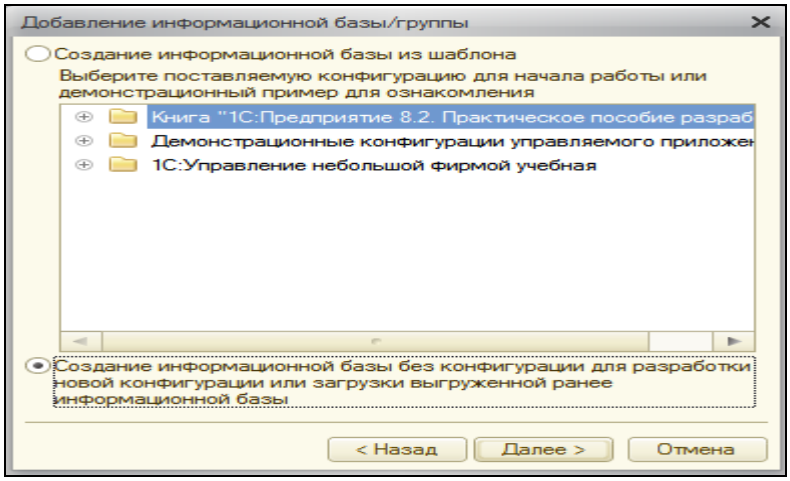
Запуск базы филиала

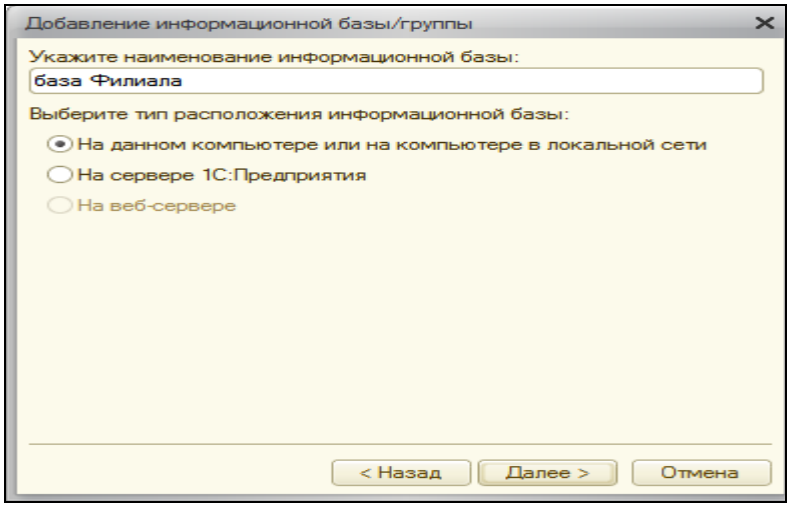
Настало время перейти к базе филиала.

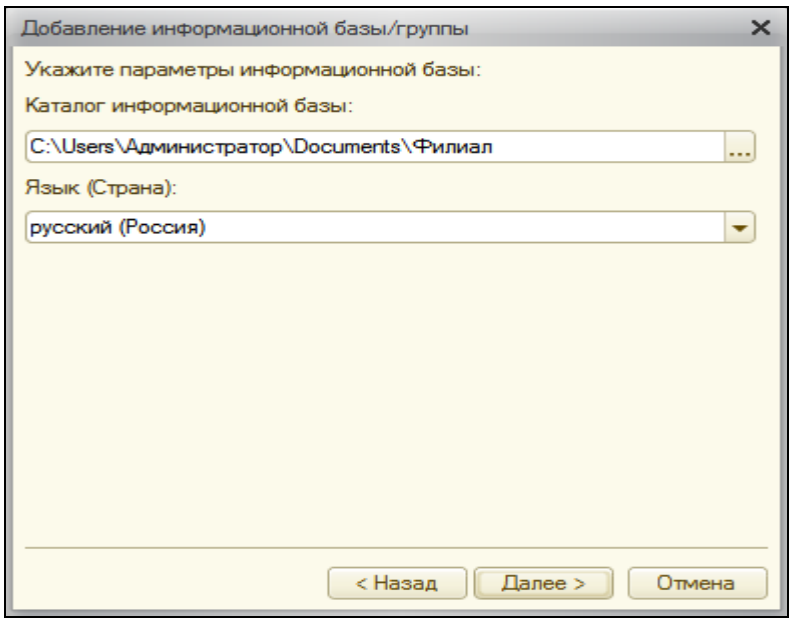
Запустим 1С:Предприятие и добавим в список баз новую базу с пустой конфигурацией, которая будет расположена в созданном нами каталоге базы филиала. Для этого в окне запуска 1С:Предприятия нажмем кнопку **Добавить** и выберем **Создание новой информационной базы**. Нажмем **Далее**. Выберем **Создание информационной базы без конфигурации для... загрузки выгруженной ранее информационной базы**. Нажмем **Далее** и укажем наименование информационной базы, например, **база Филиала**. Нажмем **Далее** и укажем каталог информационной базы, где находится сохраненная

конфигурация, например C:\Users\Администратор\Documents\Филиал. Нажмем **Далее** и **Готово**.









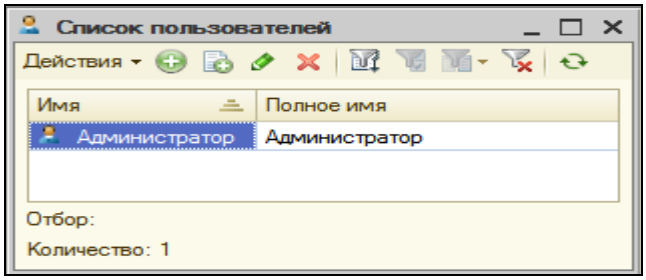
В режиме Конфигуратор

Откроем созданную нами конфигурацию **база Филиала** в режиме Конфигуратор. Выполним команду главного меню **Конфигурация – Открыть конфигурацию**. Мы видим, что список объектов конфигурации пуст.

Теперь загрузим конфигурацию из файла (**Конфигурация – загрузить конфигурацию из файла…**).

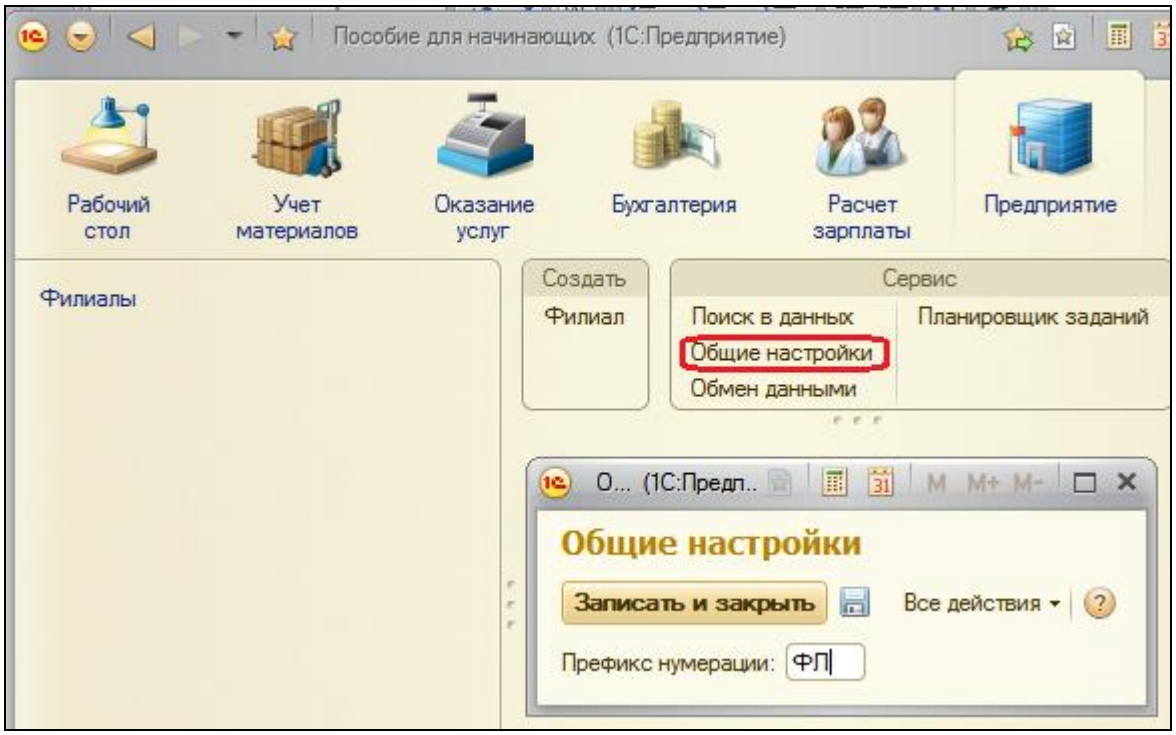
В окне выбора файла выберем каталог и имя файла, где находится сохраненная конфигурация. На вопрос обновления конфигурации ответим утвердительно и в окне изменений структуры конфигурации нажмем **Принять**.

Теперь все объекты конфигурации перенесены из нашей центральной базы. Выполним команду главного меню **Администрирование – Пользователи** и создадим в конфигурации филиала одного пользователя – **Администратор** с одноименной ролью. Дело в том, что пользователей для каждой информационной базы нужно создавать заново.



В режиме 1С:Предприятие

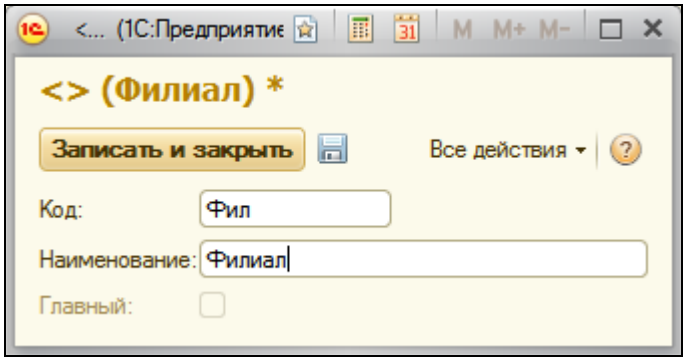
Запустим режим отладки под администратором. Первым делом зададим значение константы **ПрефиксНумерации** – **ФЛ**.



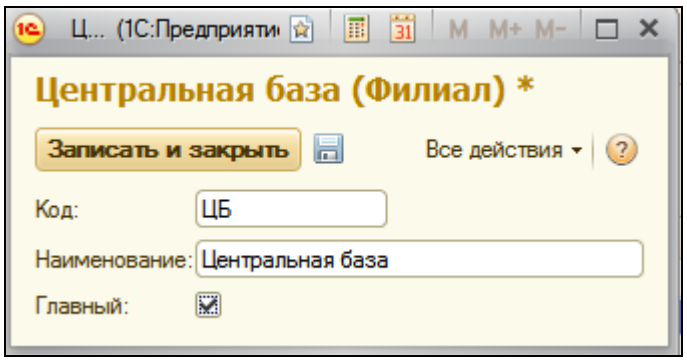
Затем откроем план обмена **Филиал** и опишем предопределенный узел (узел текущей информационной базы).

Для этого выполним команду **Филиалы** в панели навигации раздела **Предприятие**.

В списке планов обмена уже присутствует запись. Откроем и отредактируем ее. Зададим код **Фил** и наименование **Филиал**.



После этого создадим новый узел плана обмена с кодом **ЦБ**, наименованием **Центральная база** и признаком **Главный**.

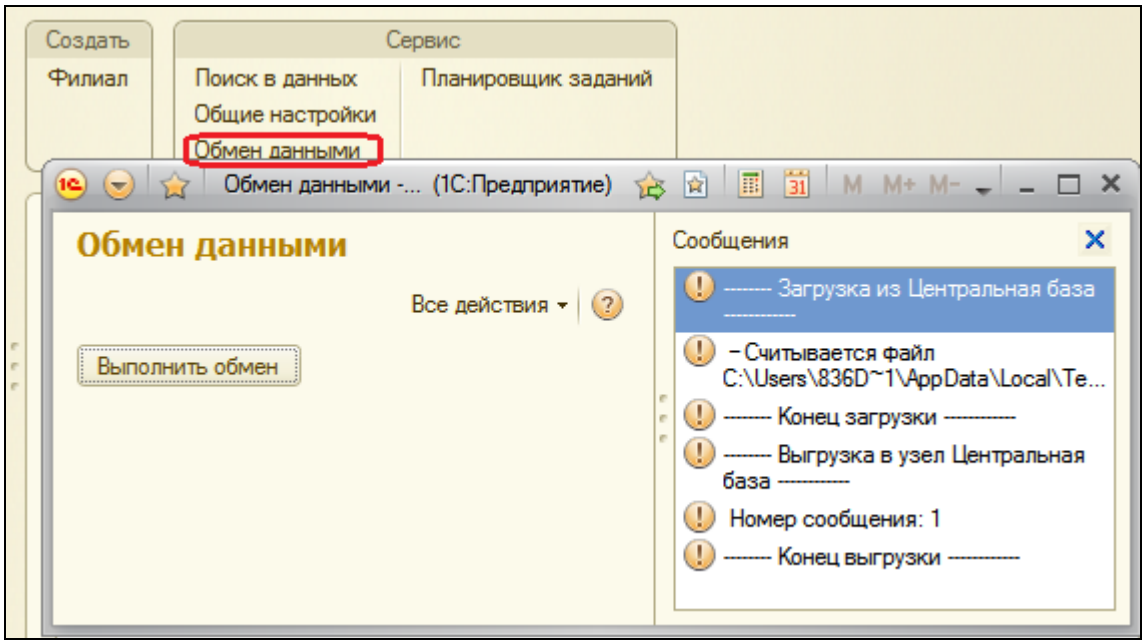


Выделим в списке узлов обмена новый узел **Центральная база** и нажмем кнопку **Зарегистрировать изменения**.

Теперь для большей наглядности откроем список справочника **Клиенты**. Сейчас в нем нет ни одного элемента.

Запустим обработку **Обмен данными** и нажмем **Выполнить обмен**.

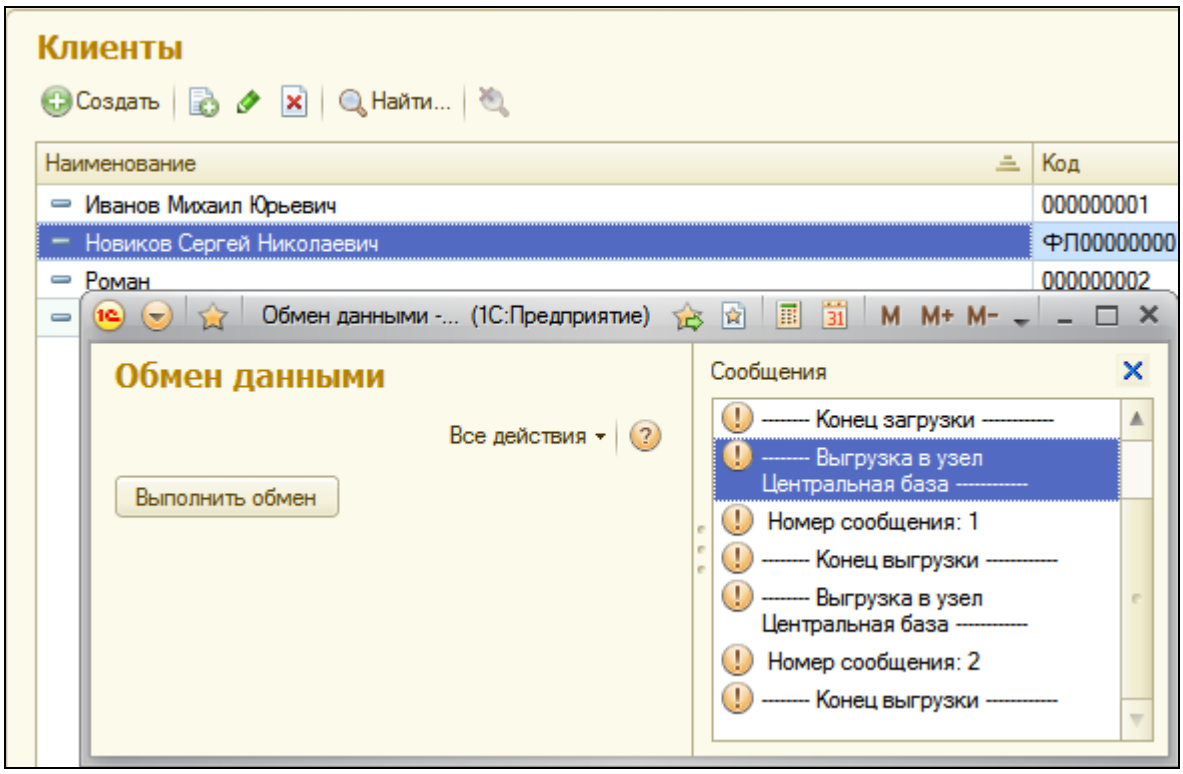
Справочник будет заполнен элементами, а в коне сообщений появится текст.



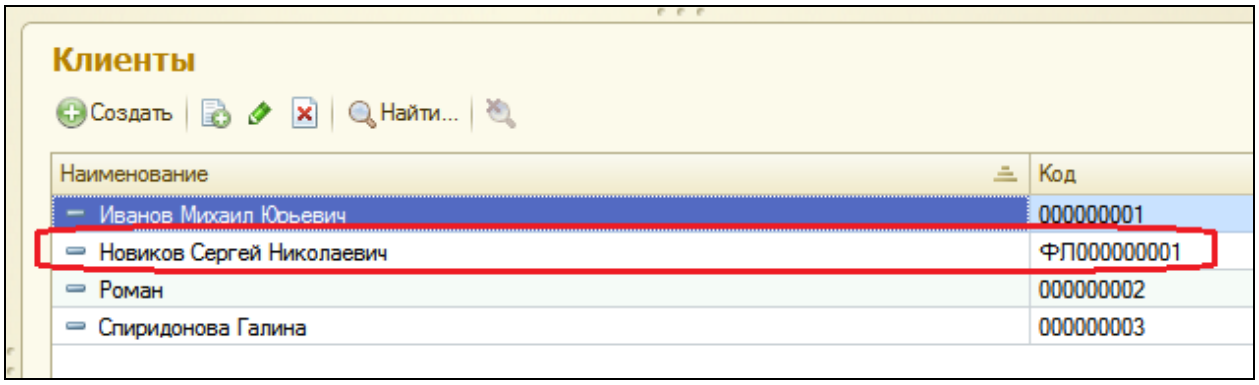
Теперь проверим, как будет происходить обмен в другую сторону.

Создадим в справочнике **Клиенты** нового клиента с произвольным наименованием. Обратите внимание, что нумерация кода нового клиента начинается с единицы и имеет префикс ФЛ.

После этого снова нажмем **Выполнить обмен** в открытой форме обработки **ОбменДанными.**



Затем перейдем в центральную базу, также выполним обмен данными и убедимся, что клиент, созданный в базе филиала, перенесен в центральную базу.



Механизм распределенных информационных баз

Является развитием универсального механизма обмена данными.

Он реализует модель распределенной информационной базы, которая подразумевает наличие идентичных конфигураций во всех узлах, имеет древовидную структуру и позволяет выполнять обмен как измененными данными, так и изменениями, внесенными в конфигурацию.

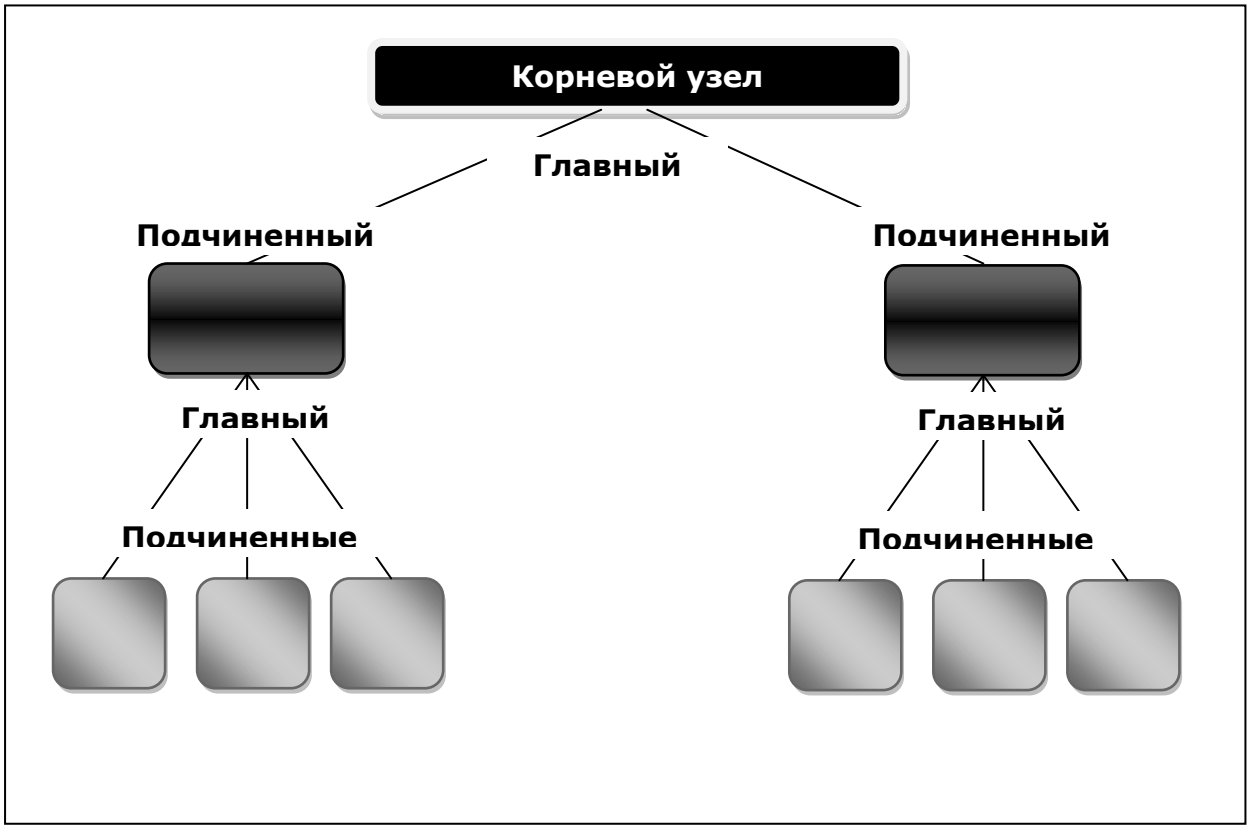
Механизм распределенных информационных баз реализуется планами обмена. Для этого план обмена содержит свойство **Распределенная информационная база**.

Если это свойство установлено, для данного плана обмена включается механизм распределенных информационных баз и разработчик получает возможность создать распределенную базу исключительно интерактивными средствами, без написания кода.

Такая возможность не исключает программного управления обменом, которое также доступно при работе с распределенными информационными базами. В ходе создания примера мы рассмотрим оба варианта организации обмена в распределенных информационных базах.

Основные сведения

Распределенная база должна иметь четко определенную древовидную структуру. Количество уровней в такой структуре не ограничено, главное – между двумя связанными узлами всегда должно быть определено отношение «главный-подчиненный».



Все узлы, кроме одного, должны иметь по одному главному узлу, и один узел не будет иметь главного узла – это корневой узел. Такое жесткое задание структуры узлов необходимо для определения порядка миграции изменений данных и изменений конфигурации.

Конфигурация может быть изменена только в узле, не имеющем главного узла (в корневом). Изменения данных могут выполняться в любом узле.

Изменения конфигурации будут передаваться от главного к подчиненным узлам. Изменения данных могут передаваться между любыми связанными узлами.

Разрешение коллизий также будет производиться исходя из отношения «главный-подчиненный». Если изменения выполнены одновременно и в главном, и в подчиненном узле, при обмене данными будут приняты только изменения главного узла, а изменения подчиненного отвергнуты.

Для любого подчиненного узла возможно создание *начального образа* – информационной базы, созданной на основании конфигурации и данных главного узла в соответствии с правилами, определяемыми планом обмена. Процедура создания начального образа узла может выполняться неоднократно, при этом удаляются все записи изменений в базе главного узла для подчиненного узла. Сразу после создания начальный образ готов к обмену с главным узлом.

Постановка задачи

В качестве примера мы покажем использование механизма распределенных баз в создании нескольких отделений нашей фирмы.

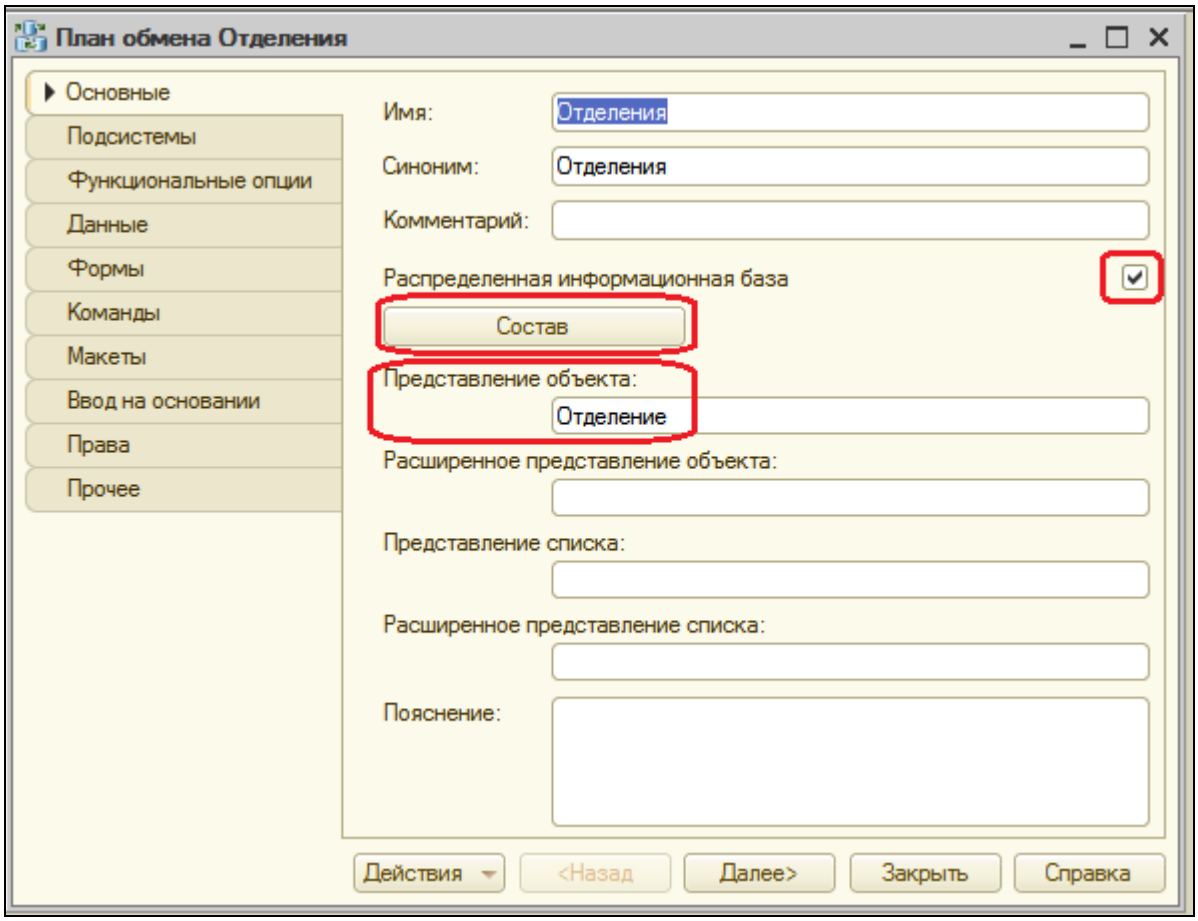
В отличие от филиалов, которые расположены в других городах, являются отдельными юридическими лицами и довольно самостоятельны в плане организации учета деятельности, отделения нашей фирмы расположены в этом же городе, никакой юридической самостоятельностью не обладают и ведут учет в точности так, как это организовано в главном офисе.

Поэтому все они используют ту же конфигурацию, что и главный офис, причем если главный офис вносит какие-либо изменения в свою конфигурацию, они должны быть своевременно внесены в конфигурацию отделения.

Интерактивный обмен

Для построения распределенной информационной базы нам понадобится создать еще один план обмена с именем **Отделения**, представлением объекта – **Отделение**.

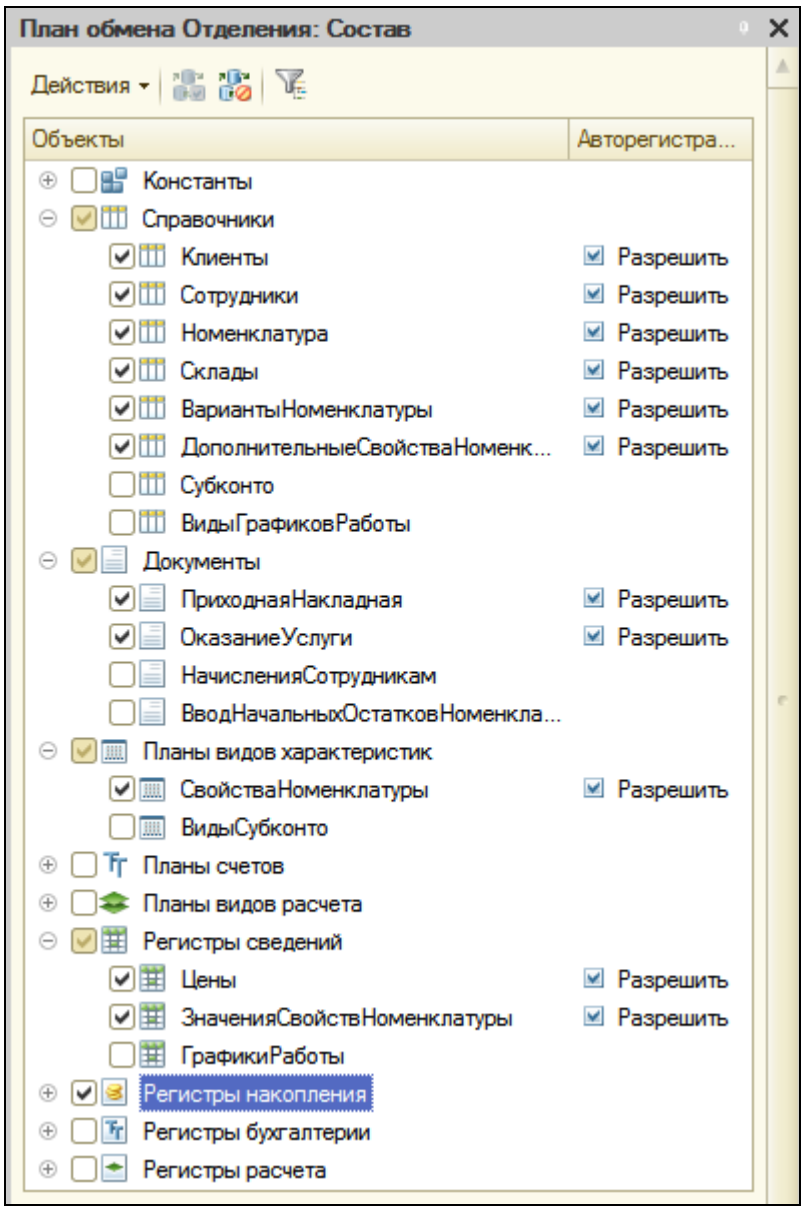
Для этого плана установим свойство **Распределенная информационная база**.



Нажмем кнопку **Состав** и определим тот же состав данных для обмена, что и в плане обмена **Филиалы**. Включим в обмен все объекты, не относящиеся к ведению бухгалтерии и расчету зарплаты.

На закладке **Формы** включим флажок **Быстрый выбор**, чтобы иметь возможность выбора узлов плана обмена из выпадающего списка.

На закладке **Подсистемы** укажем принадлежность плана обмена к подсистеме **Предприятие**. Т.о. команда открытия плана обмена будет доступна только для роли **Администратор**.



А также в окне редактора командного интерфейса подсистем **Все подсистемы** включим видимость у команды **Отделение: создать**.

В режиме 1С:Предприятие

Запустим режим отладки. Откроем план обмена Отделения и зададим параметры центрального узла (предопределенный элемент плана обмена).

Для этого выполним команду **Отделения** в панели навигации раздела **Предприятие**.

В списке планов обмена, как и раньше, присутствует одна запись. Это предопределенный узел нашей информационной базы. Откроем и отредактируем ее.

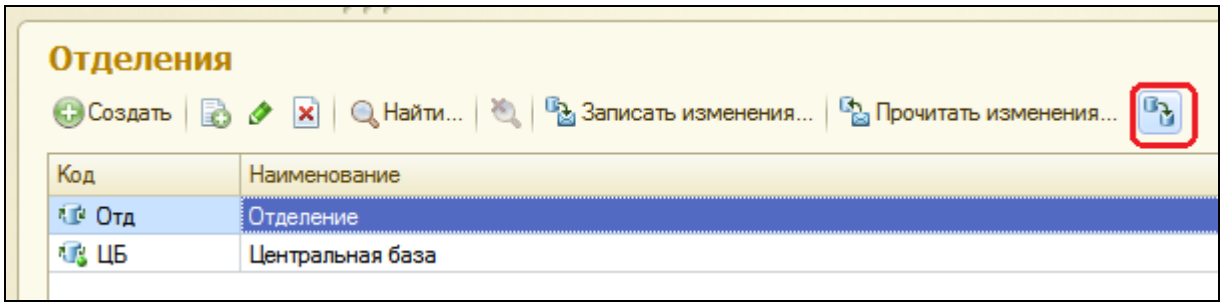
Внесем код **ЦБ** и наименование **Центральная база**.

После этого создадим новый узел с кодом **Отд** и наименованием **Отделение**.

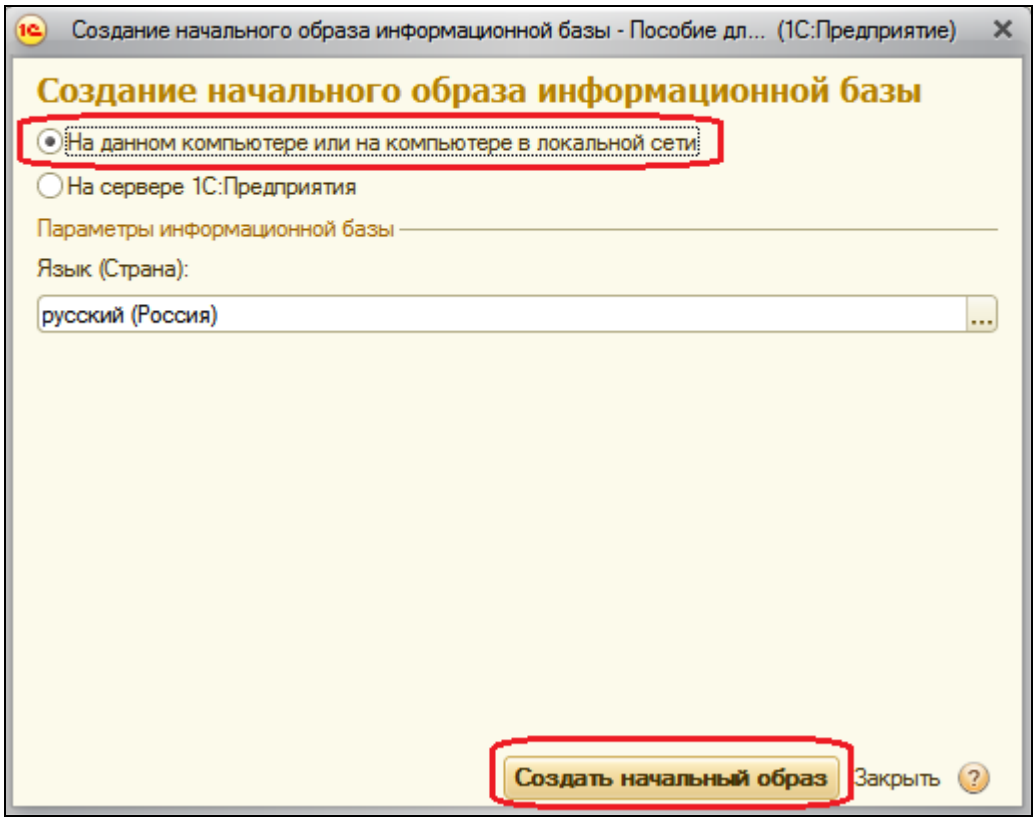
Для созданного узла доступны три кнопки в командной панели формы плана обмена: **Записать изменения**, **Прочитать изменения** и **Создать начальный образ**.

Создайте на диске новый каталог, в котором будет располагаться база отделения (Отделение).

Воспользуемся кнопкой, чтобы создать начальный образ информационной базы нашего отделения.



Укажите, что информационная база будет расположена на данном компьютере. Нажмите **Создать начальный образ.**



На следующем шаге укажите каталог информационной базы и нажмите **Готово**.

Система создаст в указанном каталоге начальный образ информационной базы нашего отделения.

Запуск базы отделения

Теперь перейдем к базе отделения. Запустим 1С:Предприятие и добавим в список баз созданную нами базу, расположенную в каталоге, в который мы поместили начальный образ информационной базы нашего отделения.

Для этого в окне запуска 1С:Предприятия нажмем кнопку **Добавить** и выберем **Добавление существующей информационной базы**.

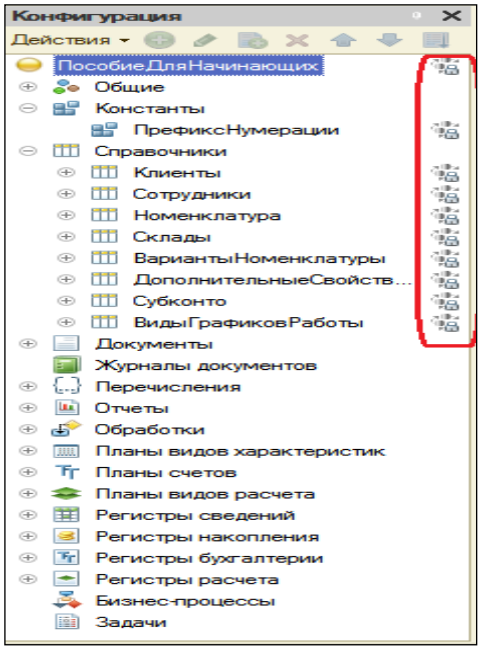
Нажмем **Далее**, затем укажем наименование информационной базы (**база Отделения**).

Нажмем **Далее**, укажем каталог информационной базы отделения, нажмем **Готово**.

В режиме Конфигуратор

Откроем созданную нами конфигурацию **база Отделения** в режиме Конфигуратор.

Откройте конфигурацию. Мы видим, что конфигурация нашего отделения стала защищенной от изменений средствами управления распределенной информационной базой.



Выполним команду главного меню **Администрирование – Пользователи** и создадим нового пользователя **Администратор** с одноименной ролью.

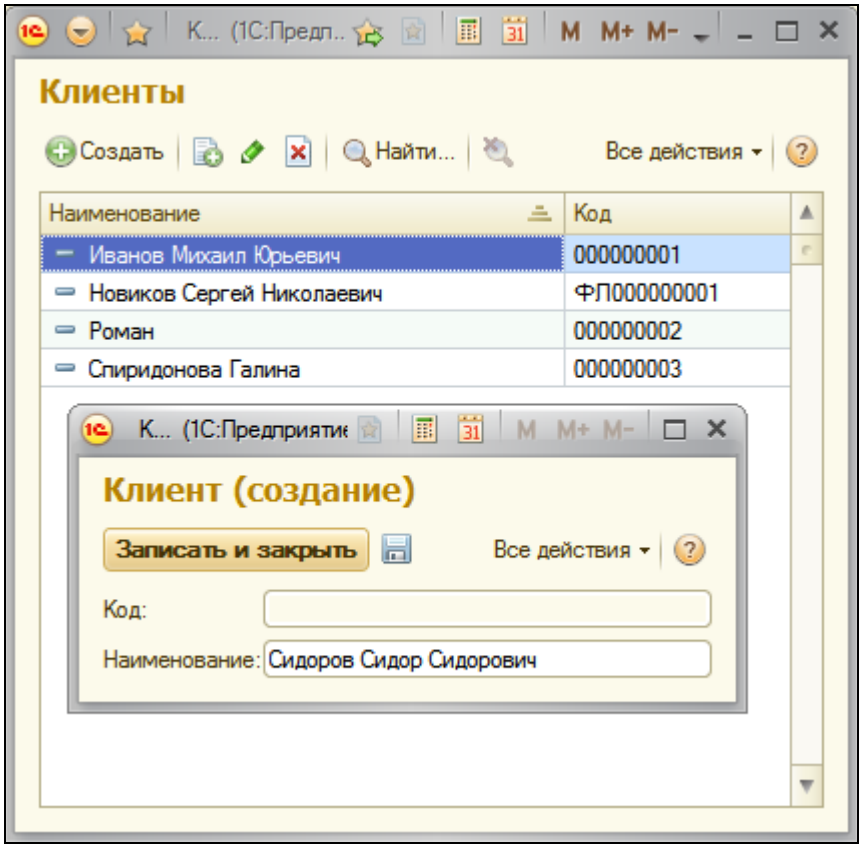
В режиме 1С:Предприятие

Запустим базу отделения в режиме отладки и откроем план обмена **Отделения**.

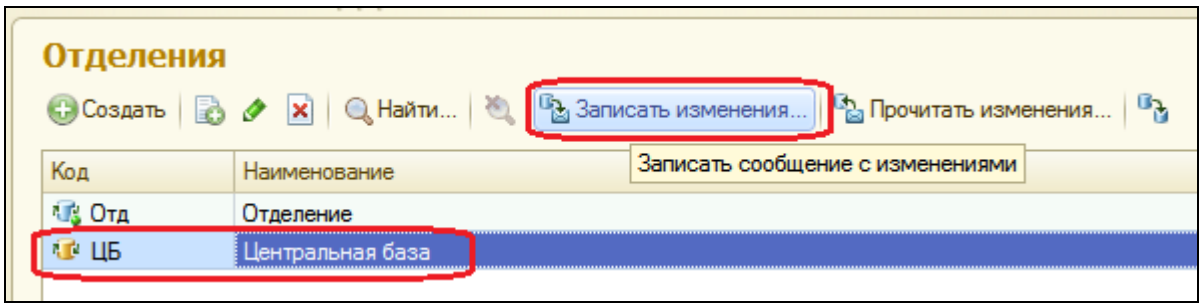
Теперь проверим работу обмена данными.

Откроем список констант (**Все функции – Константы**) и зададим значение константы **ПрефиксНумерации** – **ОТ**.

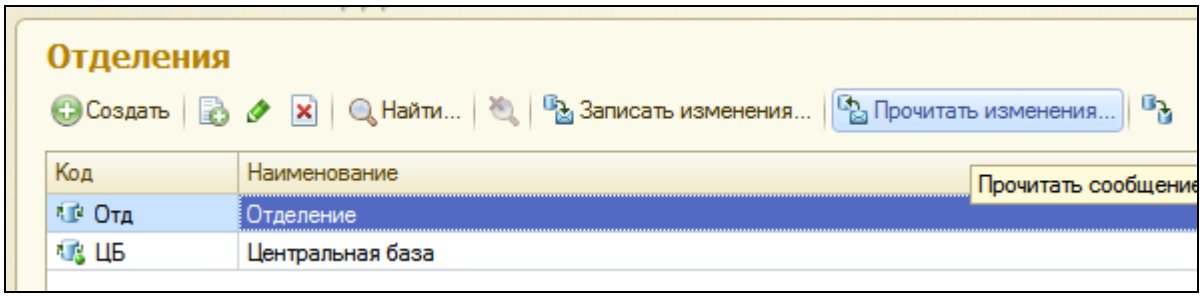
После этого откроем справочник клиентов и добавим в него нового клиента.

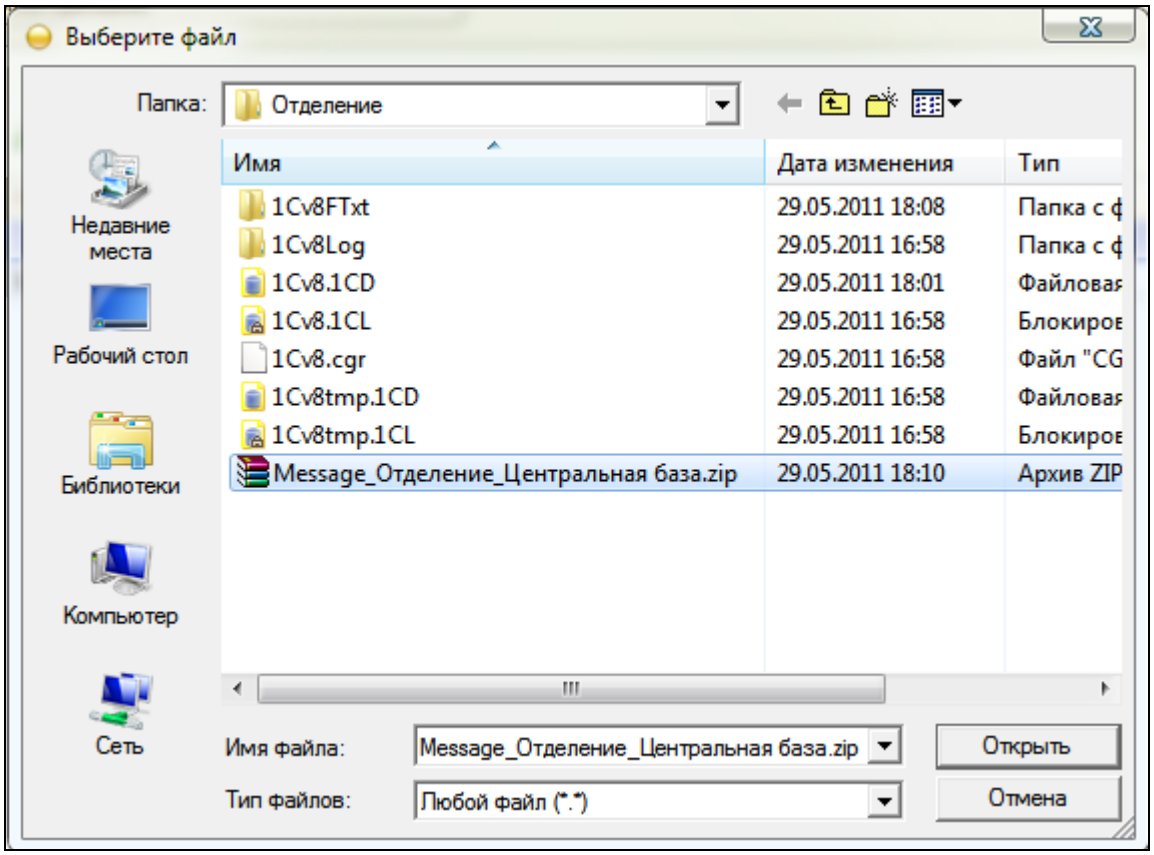


Затем выполним запись изменений для центральной базы (указав имя файла сообщения).

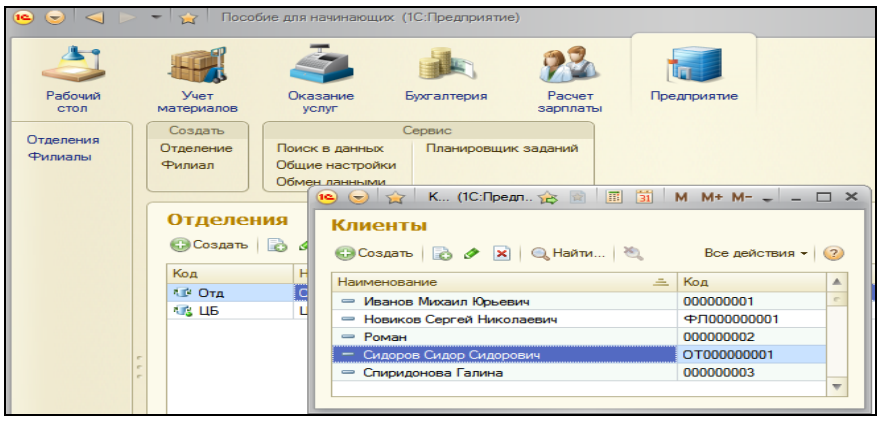


Перейдем в центральную базу и выполним чтение изменений для центральной базы (выбрав имя файла сообщения).





Убедимся, что новый клиент, созданный в базе отделения, присутствует и в центральной базе.

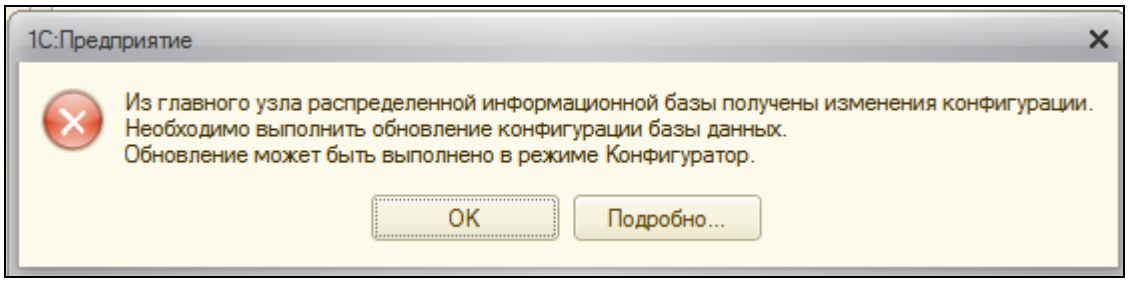


Теперь посмотрим, как будут переноситься изменения конфигурации между главным и подчиненными узлами.

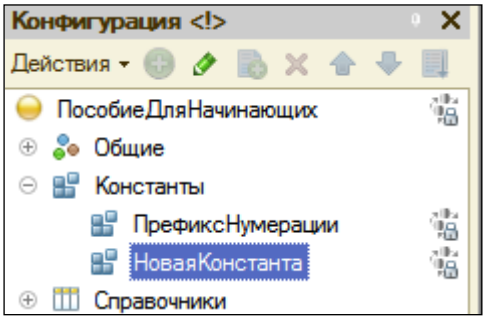
В конфигураторе центральной базы создадим новую константу с именем **НоваяКонстанта**.

Выполним обновление конфигурации и запустим режим отладки (проигнорируем сообщение о возможной ошибке про подсистемы).

Откроем план обмена **Отделения** и выполним запись изменений для подчиненного узла **Отделение**. После этого закроем конфигуратор информационной базы отделения, запустим эту базу в режиме 1С:Предприятие и выполним чтение изменений в базе подчиненного узла (**Центральная база**). По окончании чтения система выдаст следующее сообщение:



Нажмем ОК. Откроем конфигуратор базы отделения и увидим, что в основной конфигурации появилась новая константа **НоваяКонстанта**, т.е. изменения, внесенные в конфигурацию центральной базы, были автоматически перенесены в конфигурацию подчиненного узла.



Остается выполнить обновление конфигурации базы данных в подчиненном узле.

О порядке принятия изменений, когда в одном сообщении получены как изменения конфигурации, так и изменения данных.

В этом случае сначала будет изменена основная конфигурация и выдано сообщение о необходимости выполнения сохранения конфигурации. После объединения конфигураций следует выполнить повторное получение данных, при котором будут приняты уже изменения данных, содержащиеся в сообщении. Такой порядок принятия изменений не зависит от того, относятся измененные данные к существующим объектам конфигурации или к новым.

В заключение удалим объект **НоваяКонстанта** из дерева объектов нашей главной конфигурации (все равно не сможете удалить из подчиненной).

Программный обмен

Все описанные выше действия по обмену данными в распределенной информационной базе можно выполнить программно.

Мы создадим обработку, которая будет программно выполнять для выбранного узла все те действия, которые были рассмотрены в предыдущем разделе.

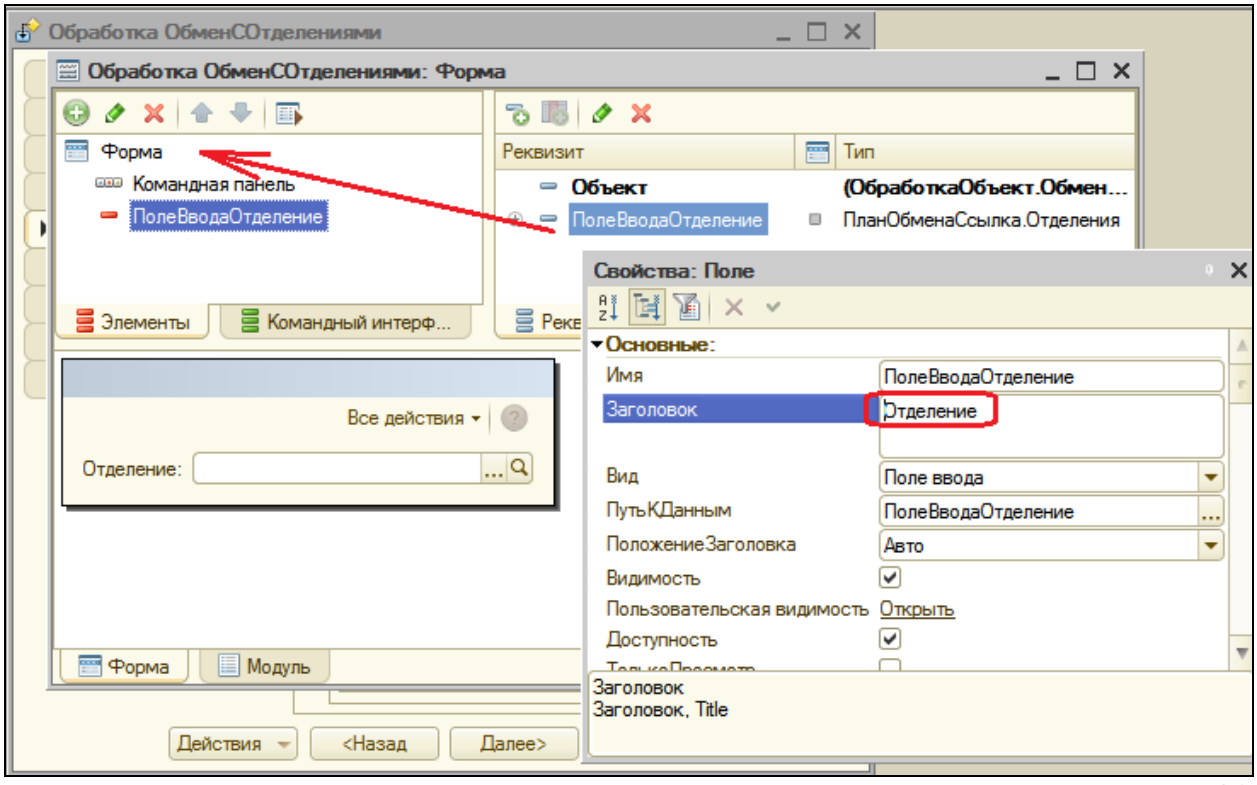
В режиме Конфигуратор

В конфигураторе центральной базы создадим новую обработку с именем **ОбменСОтделениями**.

На закладке **Формы** создадим основную форму обработки.

В окне редактора форм на закладке **Реквизиты** добавим реквизит формы **ПолеВводаОтделение** с типом **ПланОбменаСсылка.Отделения** и перетащим его в окно элементов формы.

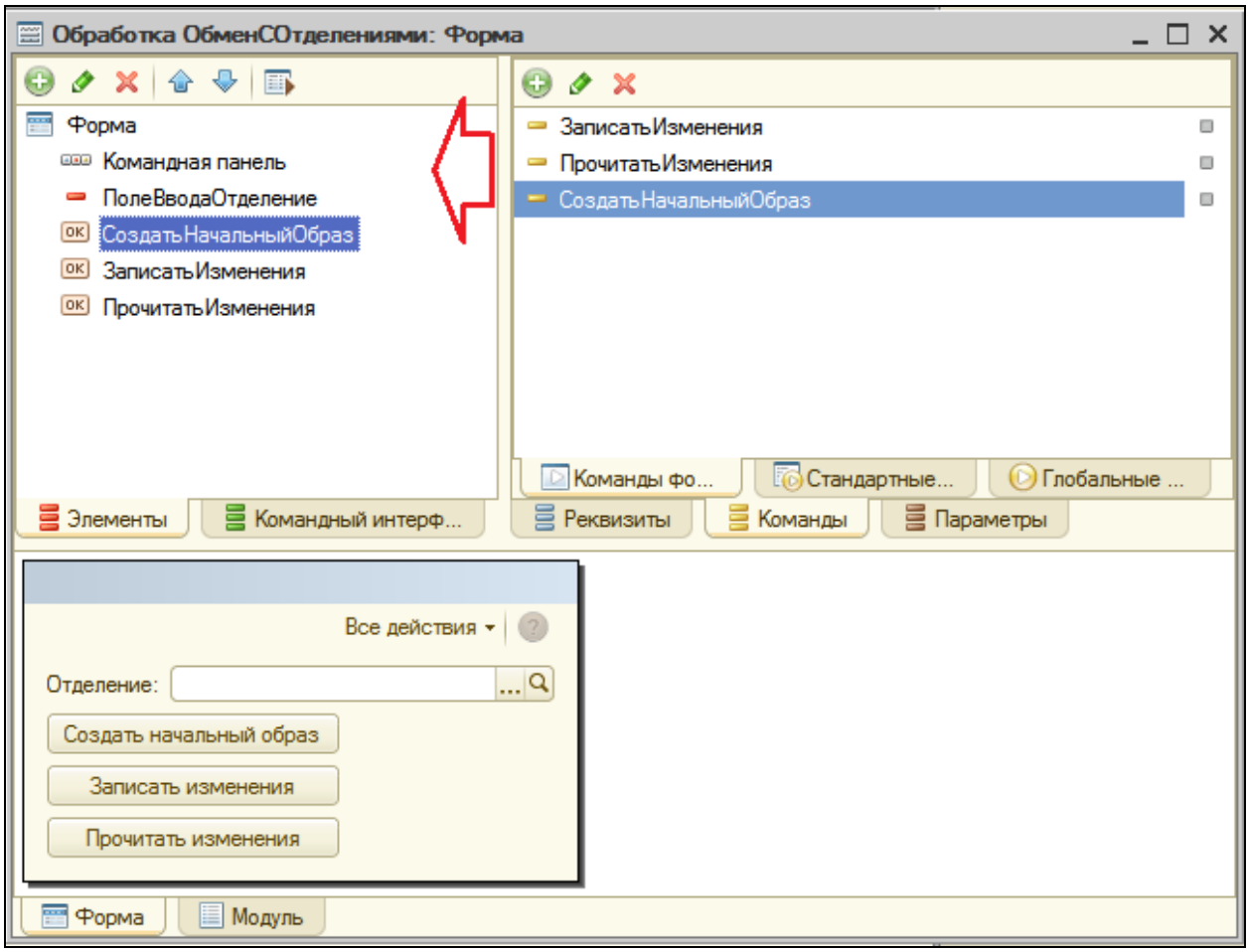
В открывшемся окне свойств этого поля зададим заголовок – **Отделение**.



На закладке **Команды** поочередно создадим команды **СоздатьНачальныйОбраз**, **ЗаписатьИзменения** и **ПрочитатьИзменения**.

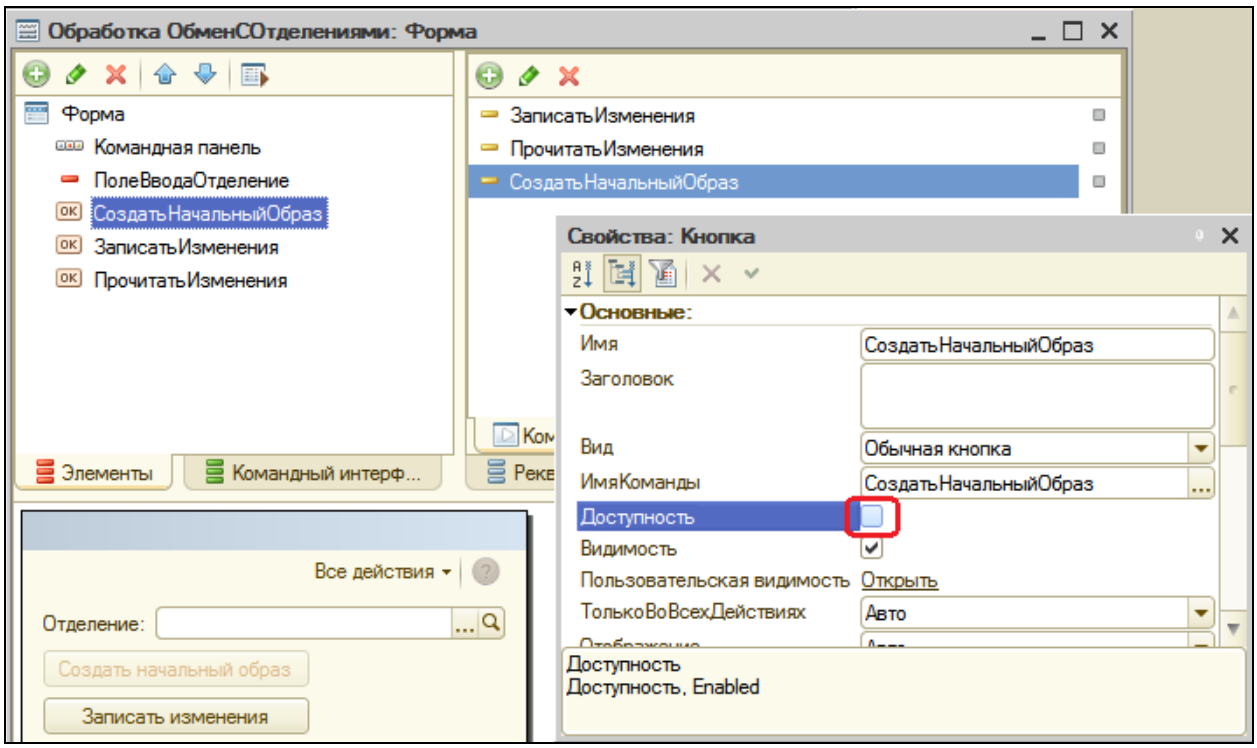
Нажмем кнопку открытия в строке **Действие** для каждой команды.

Шаблоны обработчиков для каждой команды пока заполнять не будем, а перейдем на закладку **Форма** и поочередно перетащим эти команды в окно элементов формы.



Откроем свойства кнопки **СоздатьНачальныйОбраз** и снимем флажок у свойства **Доступность**.

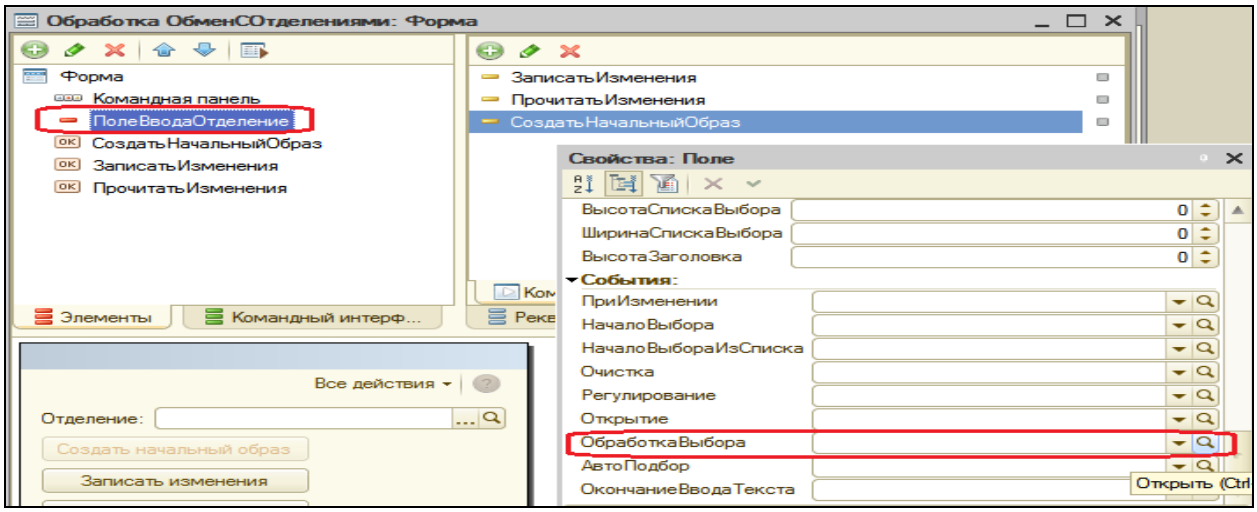
Т.о., при открытии обработки кнопка будет недоступной, пока не выбран узел плана обмена в поле **ПолеВводаОтделение**. Эта кнопка также будет недоступна в случае выбора предопределенного узла базы, т.е. создание начального образа невозможно, если выбранный узел является предопределенным.



Чтобы обеспечить такое поведение кнопки, создадим в модуле формы обработки функцию, выполняющуюся на сервере и возвращающую истину, если переданный в функцию узел является предопределенным.

```
&НаСервереБезКонтекста
Функция ПредопределенныйУзел(Узел)

	Возврат Узел = ПланыОбмена.Отделения.ЭтотУзел();

КонецФункции
```

Затем в окне элементов формы выделим элемент **ПолеВводаОтделение**, вызовем его свойства и создадим обработчик события **ОбработкаВыбора**.



Заполним обработчик:

```
&НаКлиенте
Процедура ПолеВводаОтделениеОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
            Если ПредопределенныйУзел(ВыбранноеЗначение)Тогда
            Элементы.СоздатьНачальныйОбраз.Доступность = Ложь;
      Иначе
            Элементы.СоздатьНачальныйОбраз.Доступность = Истина;
      КонецЕсли;

КонецПроцедуры
```

В этой процедуре доступность кнопки **СоздатьНачальныйОбраз** определяется в зависимости от значения функции **ПредопределенныйУзел()**, в которую передается ссылка на выбранный узел (**ВыбранноеЗначение**).

Теперь заполним обработчик команды **СоздатьНачальныйОбраз**:

```
&НаКлиенте
Процедура СоздатьНачальныйОбраз(Команда)
      Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.ВыборКаталога);
      Диалог.Заголовок = "Укажите каталог информационной базы:";
      Если Диалог.Выбрать() Тогда
            СоздатьНачальныйОбразНаСервере(ПолеВводаОтделение,
Диалог.Каталог);
            Предупреждение("Создание начального образа узла завершено.");
      КонецЕсли;

КонецПроцедуры
```

В начале процедуры мы вызываем диалог выбора каталога, в который будет помещен образ информационной базы, и затем вызываем процедуру **СоздатьНачальныйОбразНаСервере()**, исполняющуюся на сервере, в которой вызывается метод **СоздатьНачальныйОбраз()** объекта **ПланыОбменаМенеджер**.

Именно этот метод и позволяет нам создать образ подчиненного узла распределенной базы. В первом параметре метода передается ссылка на узел (реквизит формы ПолеВводаОтделение), для которого мы хотим создать начальный образ, а во втором – строка соединения, указывающая информационную базу. Поместите эту процедуру после только что написанной:

```
&НаСервереБезКонтекста
Процедура СоздатьНачальныйОбразНаСервере(Узел, КаталогСоединения)

	ПланыОбмена.СоздатьНачальныйОбраз(Узел, "File =" + КаталогСоединения);

КонецПроцедуры
```

Теперь создадим обработчик команды **Записать изменения**.

```
&НаКлиенте
Процедура ЗаписатьИзменения(Команда)
		Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
	Диалог.Заголовок = "Укажите файл обмена:";
	Если Диалог.Выбрать() Тогда
		ЗаписатьИзмененияНаСервере(ПолеВводаОтделение,
Диалог.ПолноеИмяФайла);
		Предупреждение("Запись изменений завершена.");
	КонецЕсли;

КонецПроцедуры
```

В начале процедуры мы вызываем диалог ввода имени файла, в который будут записаны изменения, и затем вызываем процедуру **ЗаписатьИзмененияНаСервере()**, исполняющуюся на сервере. В первом параметре метода передается ссылка на узел (реквизит формы **ПолеВводаОтделение**), для которого будет производиться запись изменений.

В этой процедуре мы создаем объект **ЗаписьXML** для работы с этим файлом.

Затем создаем объект **ЗаписьСообщенияОбмена**, с помощью которого будем делать сообщение обмена. В метода **НачатьЗапись()**, во втором параметре мы указываем, для какого узла обмена будет создаваться это сообщение.

После этого мы выполняем метод **ЗаписатьИзменения()** объекта **ПланыОбменаМенеджер**, который и записывает изменения, предназначенные для передачи в выбранный узел, в указанное сообщение обмена.

В заключение заканчиваем запись сообщения обмена и закрываем файл:

```
&НаСервереБезКонтекста
Процедура ЗаписатьИзмененияНаСервере(Узел, ИмяФайла)

	// Создать и проинициализировать объект ЗаписьXML
	ЗаписьXML = Новый ЗаписьXML;
```

```
        ЗаписьXML.ОткрытьФайл(ИмяФайла);

        // Создать объект ЗаписьСообщенияОбмена и начать запись сообщения
        ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
        ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

        // Записать содержимое тела сообщения обмена данными распределенной ИБ
        ПланыОбмена.ЗаписатьИзменения(ЗаписьСообщения);

        // Закончить запись сообщения и запись XML
        ЗаписьСообщения.ЗакончитьЗапись();
        ЗаписьXML.Закрыть();

КонецПроцедуры
```

Последним мы создадим обработчик команды **Прочитать изменения**.

```
&НаКлиенте
Процедура ПрочитатьИзменения(Команда)
            Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
        Диалог.Заголовок = "Укажите файл обмена:";
        Если Диалог.Выбрать() Тогда
            ПрочитатьИзмененияНаСервере(Диалог.ПолноеИмяФайла);
            Предупреждение("Чтение изменений завершено.");
        КонецЕсли;

КонецПроцедуры
```

В начале процедуры мы снова вызываем диалог ввода имени файла, который будет прочитан, и затем вызываем процедуру **ПрочитатьИзмененияНаСервере()**, исполняющуюся на сервере.

В этой процедуре создается объект **ЧтениеXML** для работы с этим файлом. Затем создаем объект **ЧтениеСообщенияОбмена** для чтения сообщения, содержащегося в указанном файле.

Затем методом **ПрочитатьИзменения(**) объекта **ПланыОбменаМенеджер** мы читаем полученное сообщение.

В заключение процедуры мы завершаем чтение сообщения обмена и закрываем файл:

```
&НаСервереБезКонтекста
Процедура ПрочитатьИзмененияНаСервере(ИмяФайла)

        // Создать и проинициализировать объект ЧтениеXML
        ЧтениеXML = Новый ЧтениеXML;
        ЧтениеXML.ОткрытьФайл(ИмяФайла);

        // Создать объект ЧтениеСообщенияОбмена и начать чтение сообщения
        ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
```

```
        ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

        // Прочитать содержимое тела сообщения
        ПланыОбмена.ПрочитатьИзменения(ЧтениеСообщения);

        // Закончить чтение сообщения и чтение XML
        ЧтениеСообщения.ЗакончитьЧтение();
        ЧтениеXML.Закрыть();

КонецПроцедуры
```

Полностью обработка **ОбменСОтделениями** в модуле формы будет такой:

```
&НаКлиенте
Процедура ЗаписатьИзменения(Команда)
                Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Сохранение);
        Диалог.Заголовок = "Укажите файл обмена:";
        Если Диалог.Выбрать() Тогда
                ЗаписатьИзменениянаСервере(ПолеВводаОтделение,
Диалог.ПолноеИмяФайла);
                Предупреждение("Запись изменений завершена.");
        КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ЗаписатьИзменениянаСервере(Узел, ИмяФайла)

        // Создать и проинициализировать объект ЗаписьXML
        ЗаписьXML = Новый ЗаписьXML;
        ЗаписьXML.ОткрытьФайл(ИмяФайла);

        // Создать объект ЗаписьСообщенияОбмена и начать запись сообщения
        ЗаписьСообщения = ПланыОбмена.СоздатьЗаписьСообщения();
        ЗаписьСообщения.НачатьЗапись(ЗаписьXML, Узел);

        // Записать содержимое тела сообщения обмена данными распределенной ИБ
        ПланыОбмена.ЗаписатьИзменения(ЗаписьСообщения);

        // Закончить запись сообщения и запись XML
        ЗаписьСообщения.ЗакончитьЗапись();
        ЗаписьXML.Закрыть();

КонецПроцедуры


&НаКлиенте
Процедура ПрочитатьИзменения(Команда)
                Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.Открытие);
        Диалог.Заголовок = "Укажите файл обмена:";
        Если Диалог.Выбрать() Тогда
                ПрочитатьИзменениянаСервере(Диалог.ПолноеИмяФайла);
```

```
            Предупреждение("Чтение изменений завершено.");
        КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ПрочитатьИзмененияНаСервере(ИмяФайла)

        // Создать и проинициализировать объект ЧтениеXML
        ЧтениеXML = Новый ЧтениеXML;
        ЧтениеXML.ОткрытьФайл(ИмяФайла);

        // Создать объект ЧтениеСообщенияОбмена и начать чтение сообщения
        ЧтениеСообщения = ПланыОбмена.СоздатьЧтениеСообщения();
        ЧтениеСообщения.НачатьЧтение(ЧтениеXML);

        // Прочитать содержимое тела сообщения
        ПланыОбмена.ПрочитатьИзменения(ЧтениеСообщения);

        // Закончить чтение сообщения и чтение XML
        ЧтениеСообщения.ЗакончитьЧтение();
        ЧтениеXML.Закрыть();

КонецПроцедуры


&НаКлиенте
Процедура СоздатьНачальныйОбраз(Команда)

        Диалог = Новый
ДиалогВыбораФайла(РежимДиалогаВыбораФайла.ВыборКаталога);
        Диалог.Заголовок = "Укажите каталог информационной базы:";
        Если Диалог.Выбрать() Тогда
            СоздатьНачальныйОбразНаСервере(ПолеВводаОтделение,
Диалог.Каталог);
            Предупреждение("Создание начального образа узла завершено.");
        КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Процедура СоздатьНачальныйОбразНаСервере(Узел, КаталогСоединения)

        ПланыОбмена.СоздатьНачальныйОбраз(Узел, "File =" + КаталогСоединения);

КонецПроцедуры


&НаСервереБезКонтекста
Функция ПредопределенныйУзел(Узел)

        Возврат Узел = ПланыОбмена.Отделения.ЭтотУзел();

КонецФункции

&НаКлиенте
```

```
Процедура ПолеВводаОтделениеОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
                Если ПредопределенныйУзел(ВыбранноеЗначение)Тогда
                Элементы.СоздатьНачальныйОбраз.Доступность = Ложь;
        Иначе
                Элементы.СоздатьНачальныйОбраз.Доступность = Истина;
        КонецЕсли;

КонецПроцедуры
```

В заключение на закладке **Подсистемы** укажем принадлежность обработки **ОбменСОтделениями** к подсистеме **Предприятие**.

В окне редактирования командного интерфейса этой подсистемы (**Все подсистемы**) установим следующий порядок следования команд в группе панели действий **Сервис**:

- Поиск в данных,
- Общие настройки,
- Обмен данными,
- Обмен с отделениями,
- Планировщик заданий.

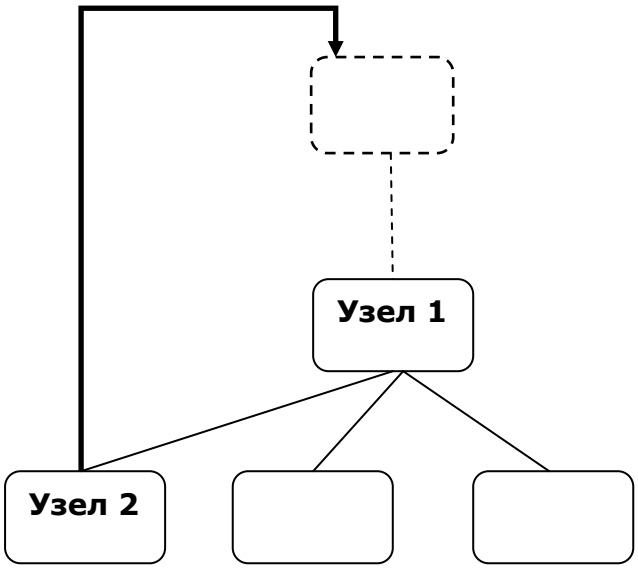
Проверить нашу обработку можно на примере, аналогичном приведенному в разделе универсального обмена данными (но не обязательно).

Изменение структуры узлов

В заключение следует сказать о том, что механизм распределенных информационных баз содержит программное средство реконфигурирования структуры узлов распределенной базы.

Для этого следует использовать метод **УстановитьГлавныйУзел()** объекта **ПланыОбменаМенеджер**. В параметре этого метода передается ссылка на узел плана обмена распределенной базы, который устанавливается главным для текущей базы. Также в этом параметре может быть передано значение **Неопределено**, и это приведет к тому, что у текущей информационной базы будет отсутствовать главный узел.

Допустим, необходимо переместить один из подчиненных узлов в корень дерева.



Для этого следует выполнить следующие действия в плане обмена:

```
// В информационной базе Узла2.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Неопределено);

// В информационной базе Узла1.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Узел2);
```

При этом будут удалены все записи регистрации изменений конфигурации Узла1, относящиеся к Узлу2, т.к. передача изменений конфигурации будет возможна теперь только от Узла2 к Узлу1. Записи регистрации изменения данных удалены не будут, т.к. передача изменений данных по-прежнему возможна между этими узлами.

Таким же образом, используя значение параметра метода Неопределено, мы можем отключать от дерева отдельную информационную базу или целое поддерево

```
// В информационной базе Узла1.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Неопределено);
```

Кроме этого, мы можем создавать распределенную базу из отдельных информационных баз с идентичной конфигурацией.

```
// В информационных базах Узла2, Узла3 и Узла4.
ПланыОбменаМенеджер.УстановитьГлавныйУзел(Узел1);
```

Контрольные вопросы

- ✓ Какие средства входят в состав механизма универсального обмена данными.

- ✓ Для чего предназначен объект конфигурации План обмена.
- ✓ Каковы основные составляющие плана обмена.
- ✓ Что такое узлы плана обмена.
- ✓ Что такое состав плана обмена и для каких элементов данных возможен обмен данными.
- ✓ Что такое авторегистрация.
- ✓ Для чего предназначен механизм регистрации изменений.
- ✓ Как работает инфраструктура сообщений.
- ✓ Каково назначение XML-сериализации.
- ✓ Для чего используется запись/чтение документов XML.
- ✓ Как создать план обмена.
- ✓ Как настроить конфигурацию для обмена данными.
- ✓ Как реализовать обмен данными в общем виде.

Практическая работа № 24

Функциональные опции (0:30)

Мы создали с вами небольшое прикладное решение, которое позволило автоматизировать работу нашей ремонтной фирмы. Наше прикладное решение настолько понравилось сотрудникам нашей фирмы, что они рассказали о нем своим соседям – косметическому салону. Сотрудники салона посмотрели на работу нашего решения и попросили автоматизировать и их салон тоже.

Мы создали универсальную конфигурацию, подходящую для автоматизации практически любой деятельности, связанной с оказанием услуг, поэтому решение подходит и косметическому салону.

Все, что нам осталось сделать, чтобы наша конфигурация смогла работать в косметическом салоне – создать новую информационную базу и заполнить ее новыми данными. В случае необходимости, довольно легко добавить новые модули. Но если потребуется обратное решение – что-то будет ненужно, то простым удалением объектов не обойтись – связи между объектами конфигурации бывают очень сложными и запутанными.

Поэтому в 1С:Предприятии 8 существует механизм функциональных опций, который позволяет включать и выключать при внедрении целые блоки функциональности, не изменяя при этом саму конфигурацию.

Функциональные опции позволяют разработчику выделить некоторую часть функциональности прикладного решения, которую можно оперативно включать или выключать на этапе внедрения и/или в процессе работы системы.

Опции «Бухгалтерский учет» и «Расчет зарплаты»

Предположим, что косметический салон по каким-то причинам не ведет бухучет и расчет зарплаты. Для отключения соответствующей функциональности мы создадим функциональные опции **БухгалтерскийУчет** и **РасчетЗарплаты**, установим их для соответствующих объектов конфигурации и отключим их в режиме 1С:Предприятие.

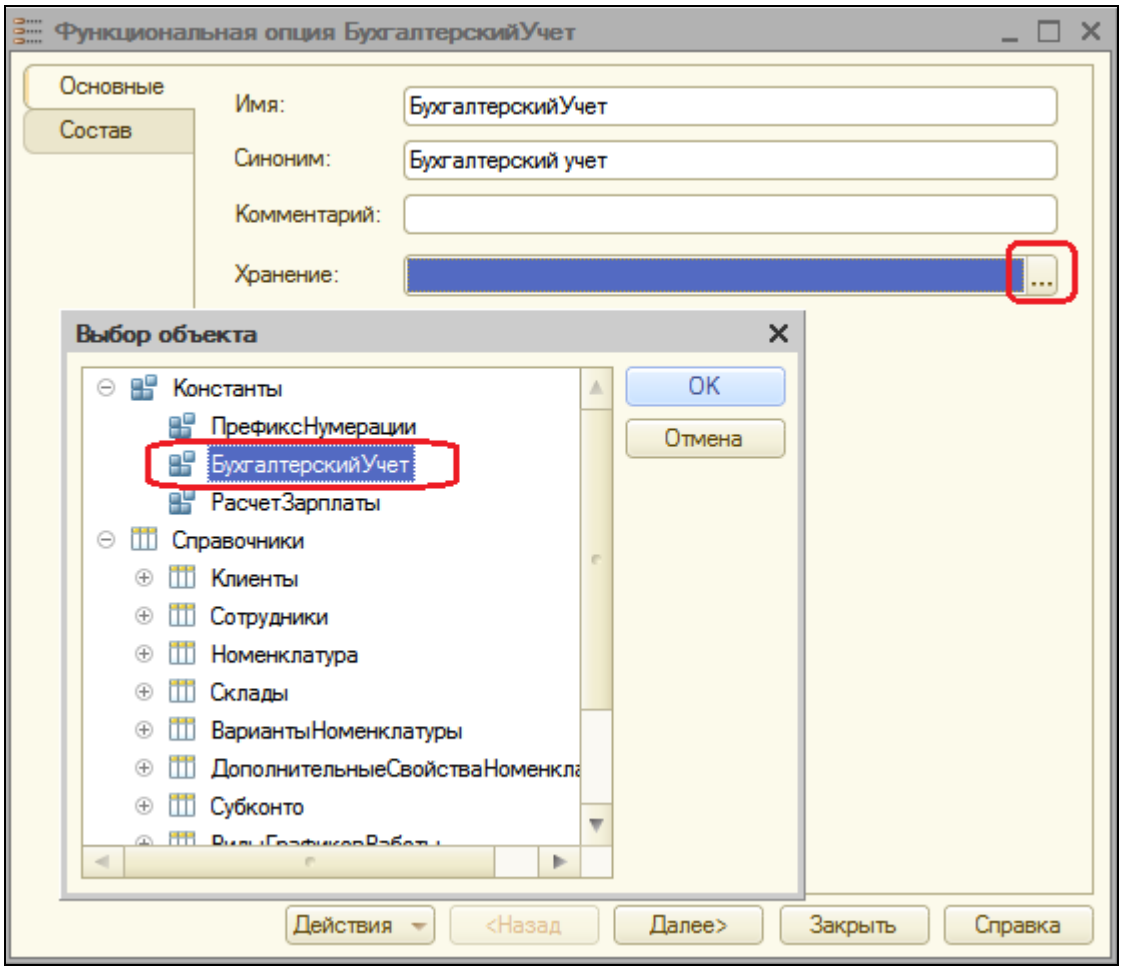
Т.о., при совершенно одинаковой конфигурации в прикладном решении косметического салона не будет видно никаких действий, связанных с расчетом зарплаты и ведением бухучета, будто их и нет вовсе.

В режиме Конфигуратор

Поскольку значения функциональных опций должны где-то храниться, добавим сначала константы **БухгалтерскийУчет** и **РасчетЗарплаты** с типом **Булево**, в которых будут храниться значения функциональных опций.

Если значение константы **Истина**, значит, функциональная опция включена. Если значение **Ложь** – выключена.

Затем раскроем ветвь **Общие**, выделим ветвь **Функциональные опции** и создадим функциональные опции **БухгалтерскийУчет** и **РасчетЗарплаты**, указав в свойстве **Хранение** соответствующие константы.

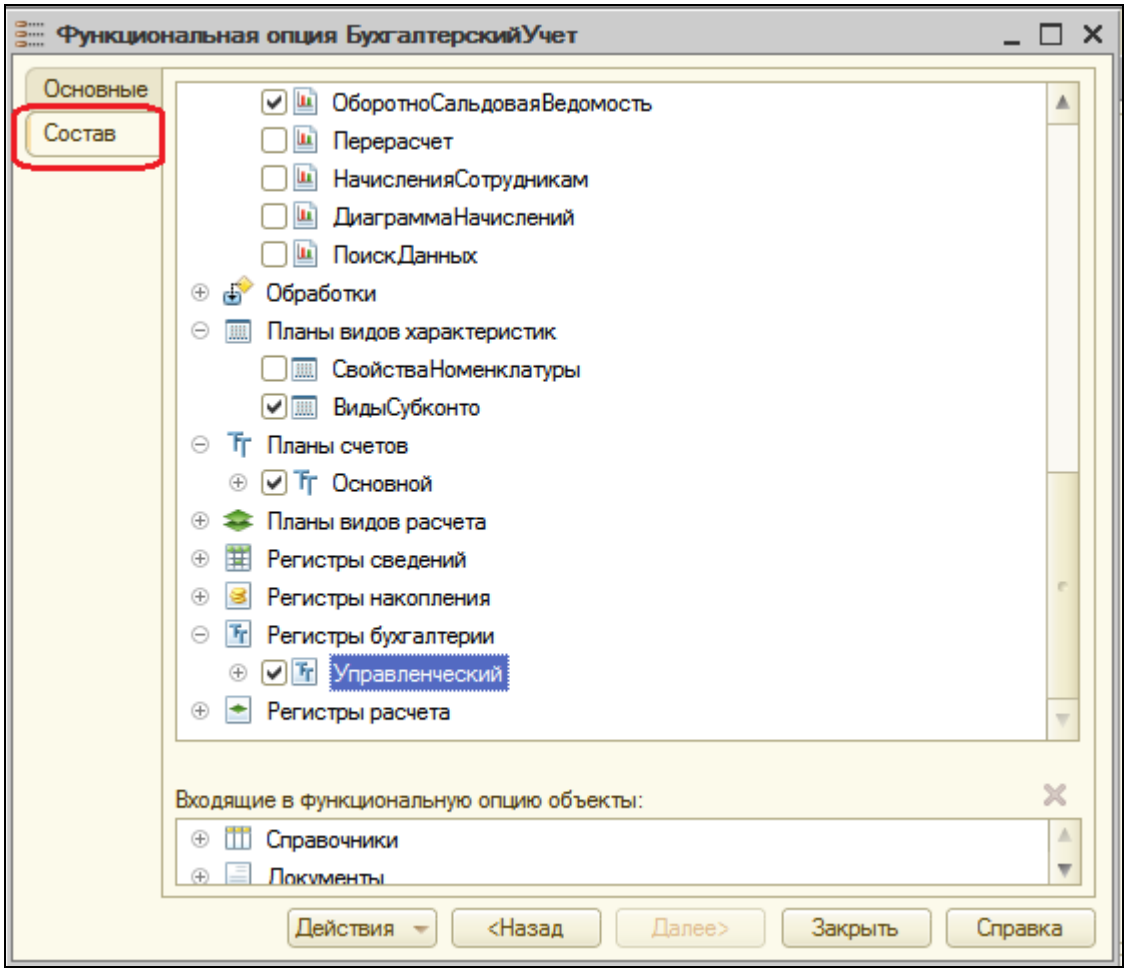


Теперь нам нужно привязать объекты конфигурации к функциональным опциям.

К ведению бухучета в нашей конфигурации относятся объекты:

- Справочник **Субконто**,
- Документ **ВводНачальныхОстатковНоменклатуры**,
- Отчет **ОборотноСальдоваяВедомость**,
- План видов характеристик **ВидыСубконто**,
- План счетов **Основной**,
- Регистр бухгалтерии **Управленческий**.

На закладке **Состав** отметим эти объекты для функциональной опции **БухгалтерскийУчет**.

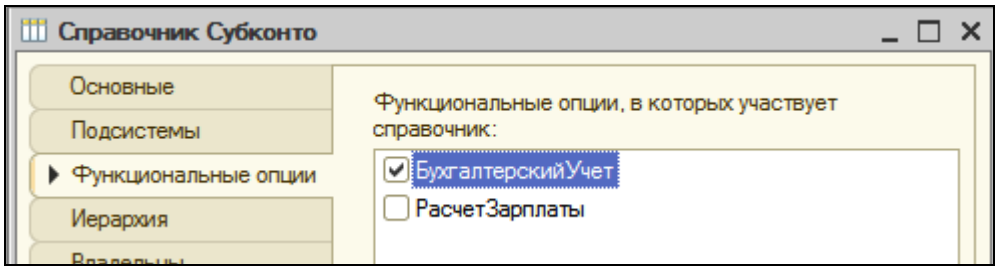


К расчету зарплаты в нашей конфигурации относятся:

- Справочник **ВидыГрафиковРаботы**,
- Документ **НачислениеСотрудникам**,
- Отчет **НачислениеСотрудникам**,
- Отчет **Перерасчет**,
- Отчет **ДиаграммаНачислений**,
- План видов расчета **ОсновныеНачисления**,
- Регистр сведений **ГрафикиРаботы**,
- Регистр расчета **Начисления**.

На закладке **Состав** отметим эти объекты для функциональной опции **РасчетЗарплаты.**

Теперь, если мы откроем окно редактирования объекта конфигурации Справочник **Субконто** или любого другого объекта конфигурации, входящего в состав функциональной опции **БухгалтерскийУчет**, то эта опция будет включена на закладке **Функциональные опции** окна редактирования этого объекта.



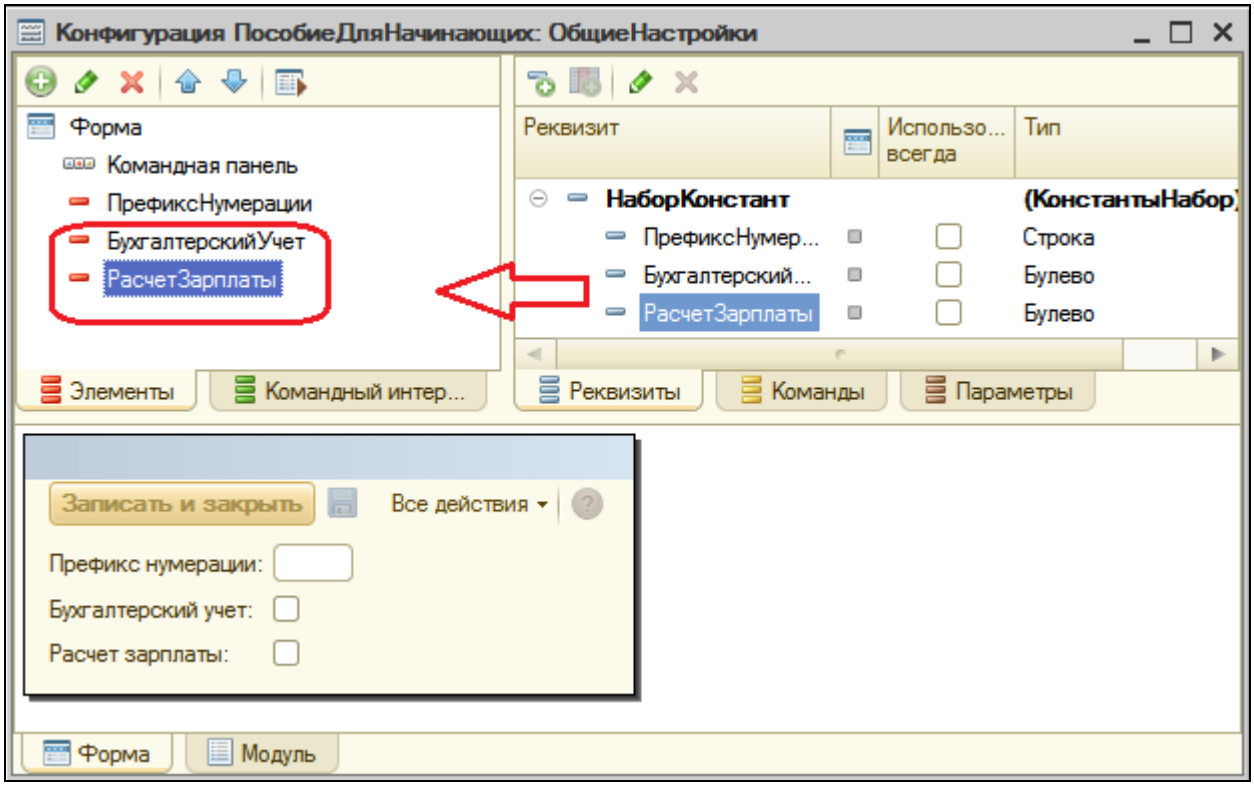
У объектов, относящихся к расчету зарплаты, мы увидим включенной функциональную опцию **РасчетЗарплаты** на закладке **Функциональные опции**.

Если включить в состав функциональной опции какую-нибудь подсистему, то мы вообще не увидим соответствующего раздела в 1С:Предприятии, пока данная функциональная опция отключена.

После этого раскроем ветвь **Общие формы** и откроем общую форму констант. Эту форму с именем **ОбщиеНастройки** мы создали в предыдущей работе, и она уже содержит константу **ПрефиксНумерации**.

Теперь нам нужно добавить в нее новые константы, чтобы затем в пользовательском режиме открывать форму констант и изменять значение функциональных опций.

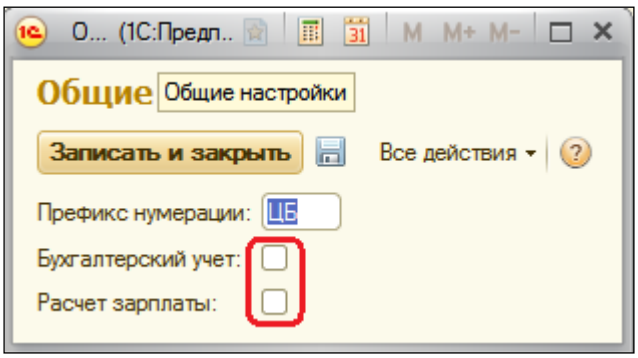
На закладке **Реквизиты** этой формы раскроем основной реквизит **НаборКонстант** и перетащим константы **БухгалтерскийУчет** и **РасчетЗарплаты** в окно элементов формы.



В режиме 1С:Предприятие

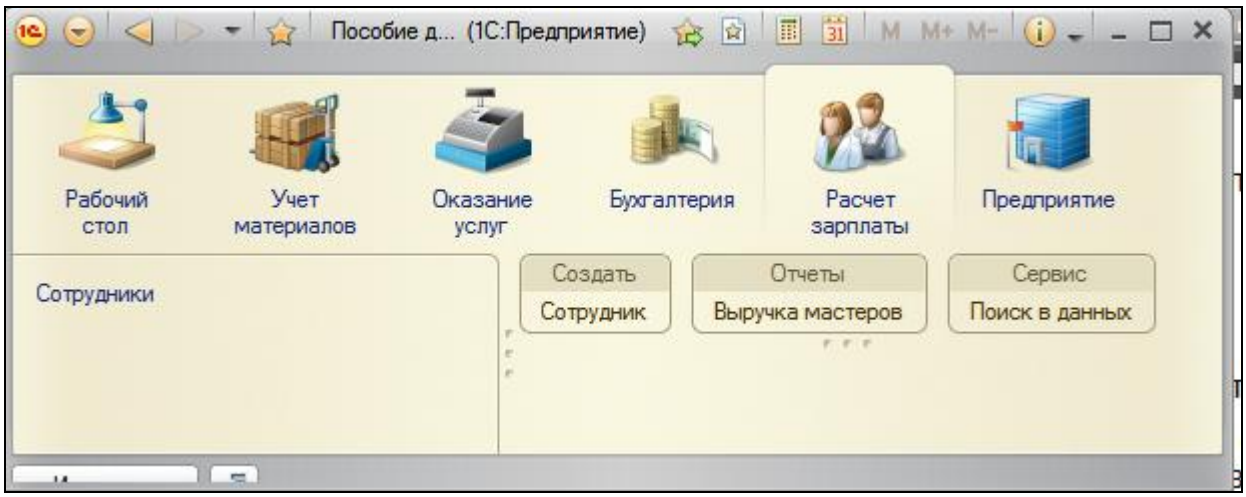
Запустим режим отладки. В панели действий раздела **Предприятие** выполним команду **Общие настройки**.

В открывшейся форме констант мы видим, что обе константы имеют значение **Ложь**.

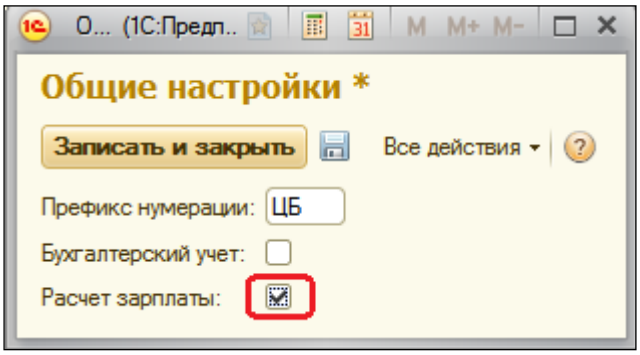


Это значит, что соответствующие функциональные опции отключены.

И действительно, в разделах Бухгалтерия и Расчет зарплаты мы не видим команд для ведения бухучета и расчета зарплаты.

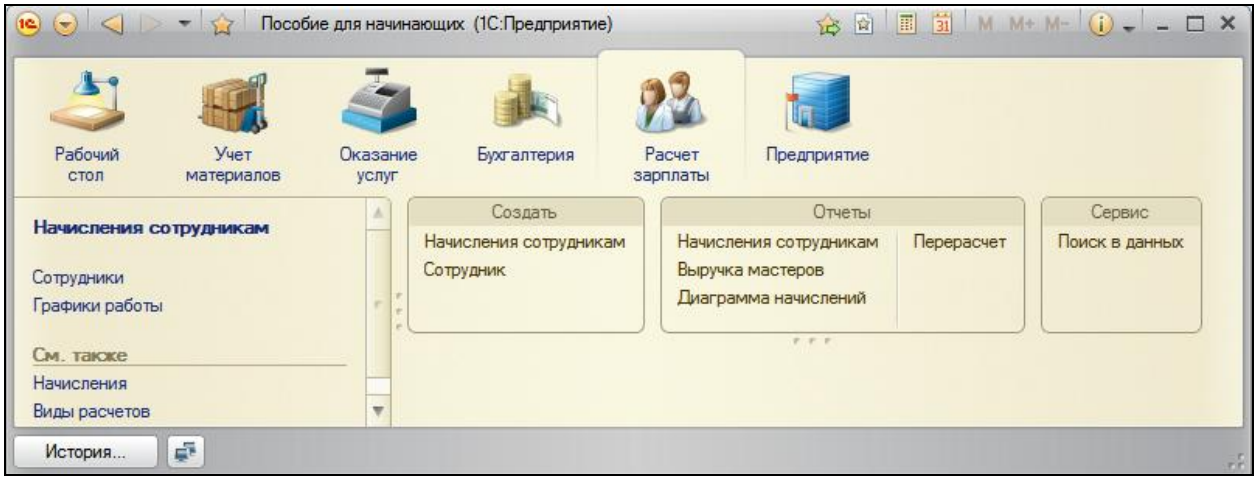


Теперь, если руководство косметического салона пожелает вести расчет зарплаты, то администратор включит соответствующую опцию **Расчет зарплаты**, и всё!



Нужно только сохранить измененное значение констант и перезапустить 1С:Предприятие, чтобы платформа отрисовала новый интерфейс.

В результате раздел **Расчет зарплаты** будет выглядеть так:



А если мы включим и вторую функциональную опцию **Бухгалтерский учет**, то мы восстановим интерфейс прикладного решения нашей фирмы.

Вот так быстро и легко происходит настройка прикладного решения под требования заказчика.

Опция «Учет клиентов»

Функциональные опции могут влиять не только на командный интерфейс приложения, но и на внешний вид форм, используемых в прикладном решении. Кроме этого, включение/выключение функциональности можно выполнять и без перезапуска клиентского приложения. А если к этому прибавить возможности работы с функциональными опциями во встроенном языке, то становится понятным, что механизм функциональных опций может сделать процесс внедрения и настройки прикладного решения у заказчика простым и понятным даже для неопытного пользователя.

Рассмотрим еще один пример.

«Поименный» учет клиентов при оказании услуг востребован далеко не всегда. Зачастую важен лишь сам факт оказания услуги, при этом личность клиента не имеет значения.

Поэтому предусмотрим в нашей конфигурации возможность отключить ведение списка клиентов и избавимся от необходимости указывать клиента каждый раз при оказании услуги.

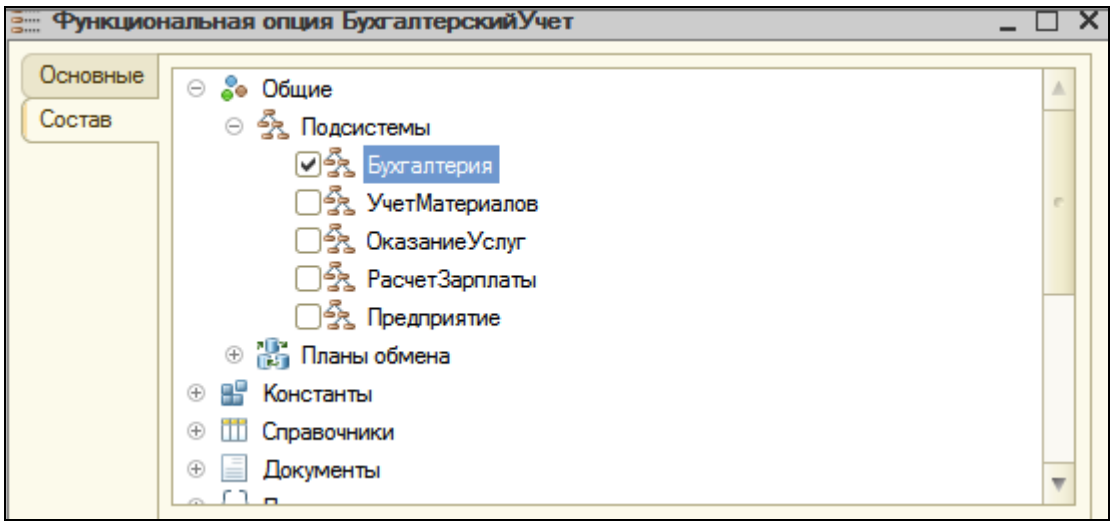
Также доработаем существующие функциональные опции, включив в них и подсистемы **Бухгалтерия** и **РасчетЗарплаты**, чтобы наше решение выглядело законченным.

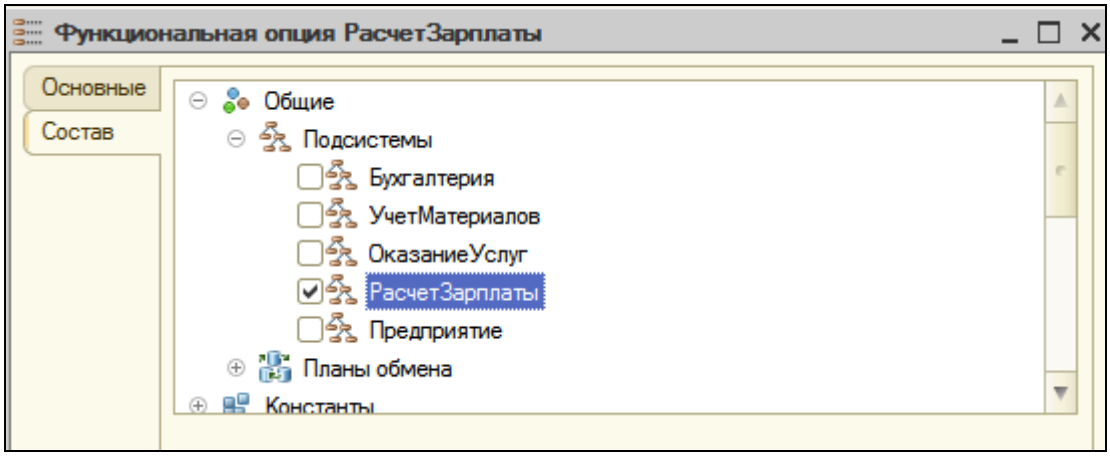
В режиме Конфигуратор

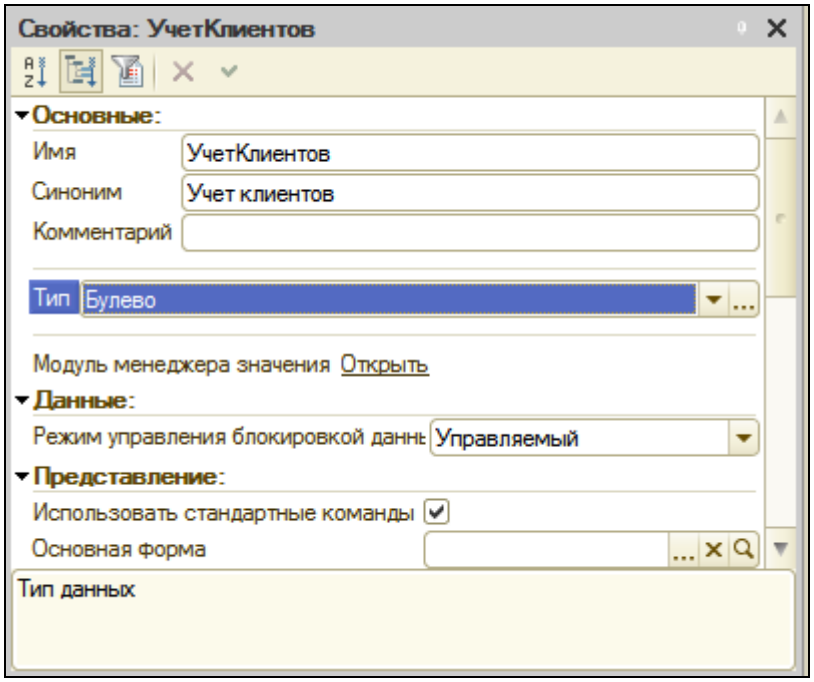
Откроем состав функциональной опции **БухгалтерскийУчет** и добавим в него подсистему **Бухгалтерия**.

Аналогично добавим в состав функциональной опции **РасчетЗарплаты** подсистему **РасчетЗарплаты**.

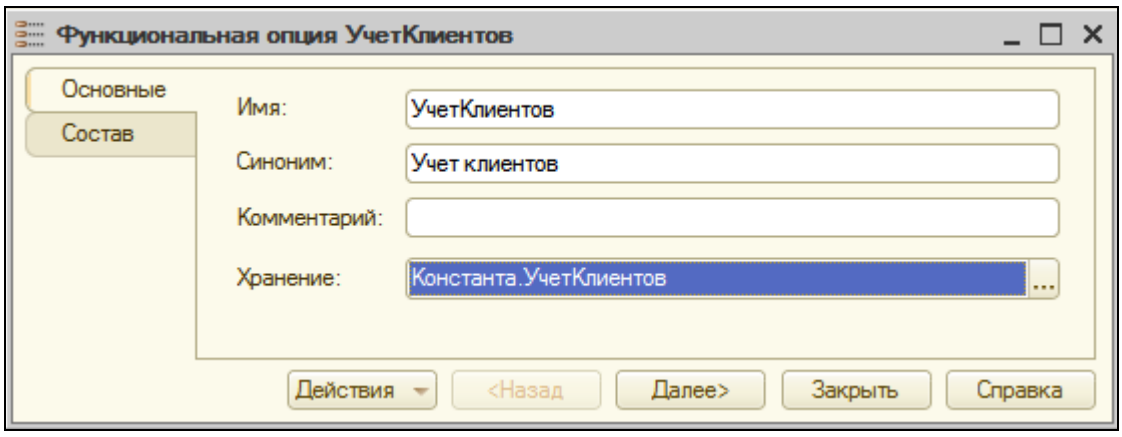
Для хранения этой опции добавим константу с именем **УчетКлиентов** с типом **Булево**.







Добавим функциональную опцию УчетКлиентов и укажем, что ее значение будет храниться в одноименной константе.

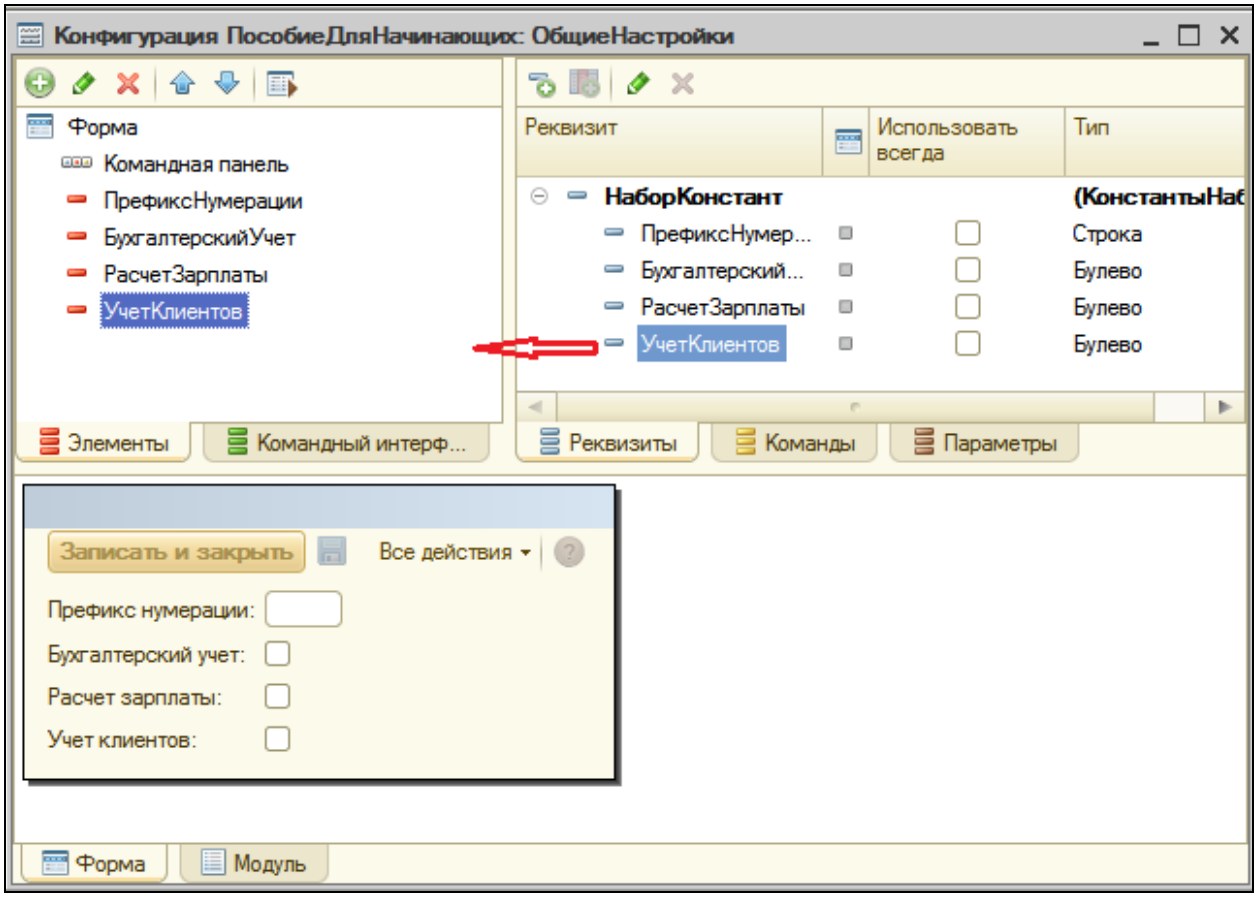


Теперь на закладке **Состав** укажем, что в эту опцию будут входить:

- Справочник **Клиенты**,
- Реквизит **Клиент** документа **ОказаниеУслуги**,
- Измерение **Клиент** регистра накопления **Продажи**.

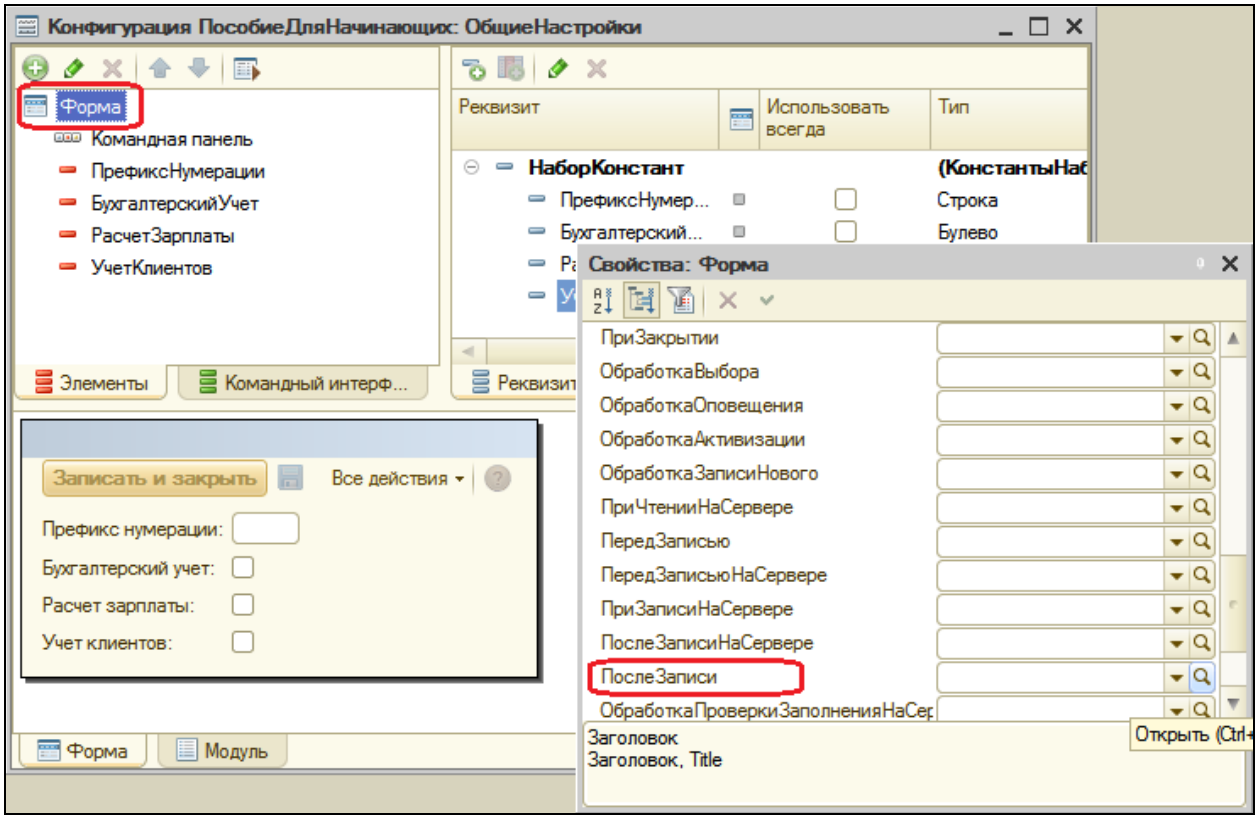
Теперь доработаем общую форму **ОбщиеНастройки**, с помощью которой мы устанавливаем значения функциональных опций.

Перенесем в состав элементов формы нашу новую константу **УчетКлиентов**.



После этого обеспечим автоматическую перерисовку интерфейса прикладного решения после установки новых значений функциональных опций.

Для этого в дереве элементов формы выделим корень **Форма**, в свойствах найдем событие **ПослеЗаписи** формы и нажмем кнопку **Открыть** в поле ввода этого события.



В открывшемся модуле формы, в обработчике события формы **После записи**, напишем единственную строку:

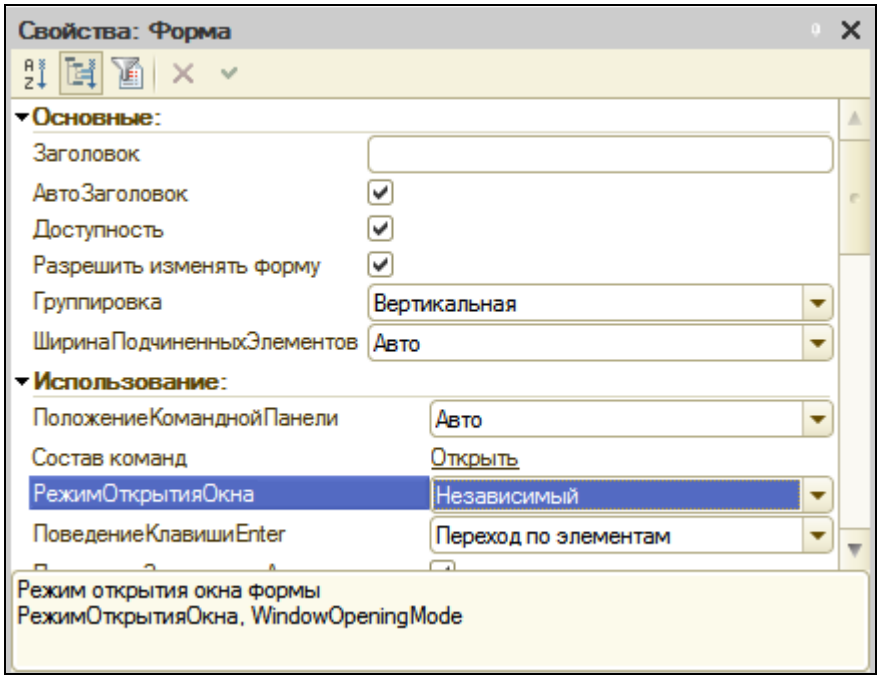
```
&НаКлиенте
Процедура ПослеЗаписи(ПараметрыЗаписи)
        ОбновитьИнтерфейс();
КонецПроцедуры
```

ОбновитьИнтерфейс() – это метод глобального контекста, который обновляет командный интерфейс, рабочий стол и открытые формы с учетом текущих значений функциональных опций и их параметров.

Для удобной проверки работы функциональных опций, сделаем так, чтобы открытие этой общей формы не блокировало основное окно программы.

Так происходит, потому что по умолчанию конструктор форм установил для этой формы свойство **РежимОткрытияОкна** в значение **Блокировать окно владельца**.

Поэтому перейдем на закладку **Форма**, в дереве элементов формы выделим корневой элемент, в палитре свойств найдем свойство **РежимОткрытияОкна** и установим его в значение **Независимый**.



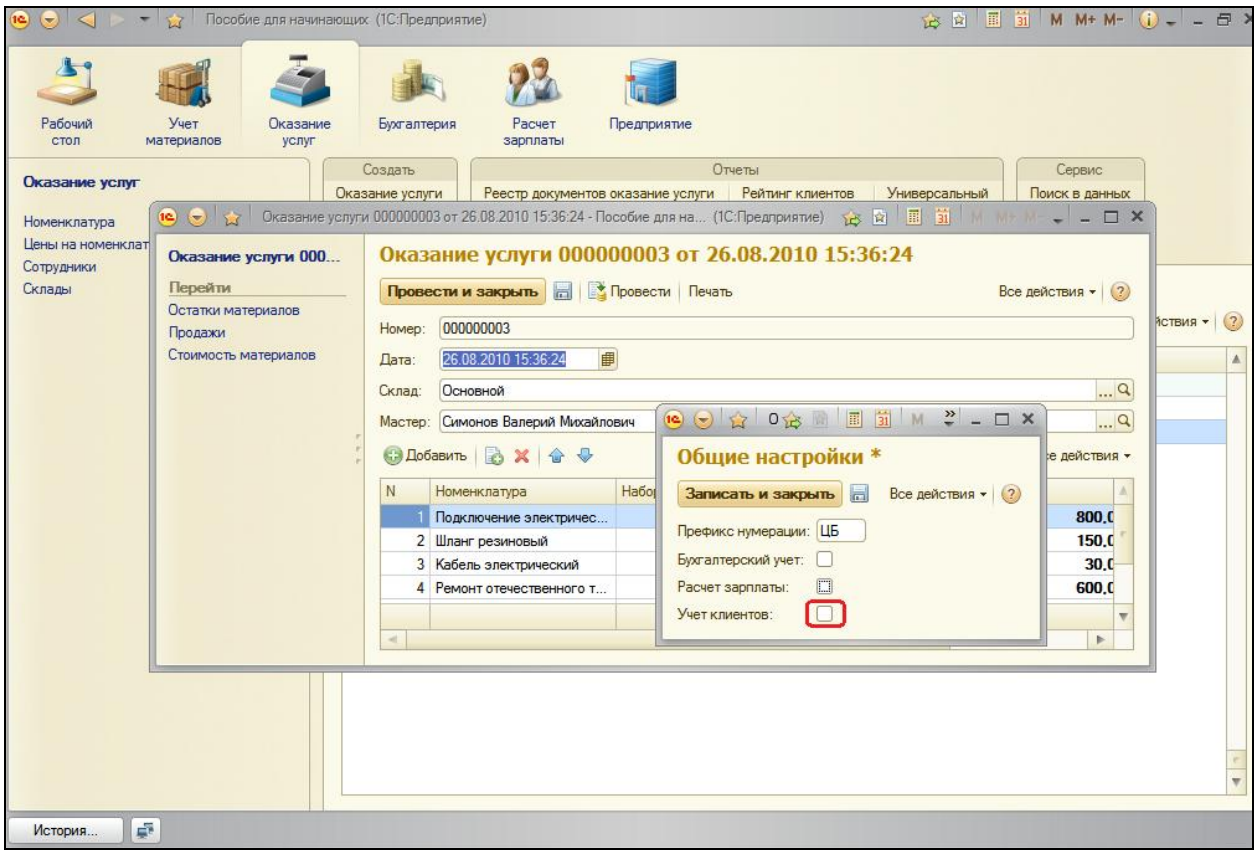
В режиме 1С:Предприятие

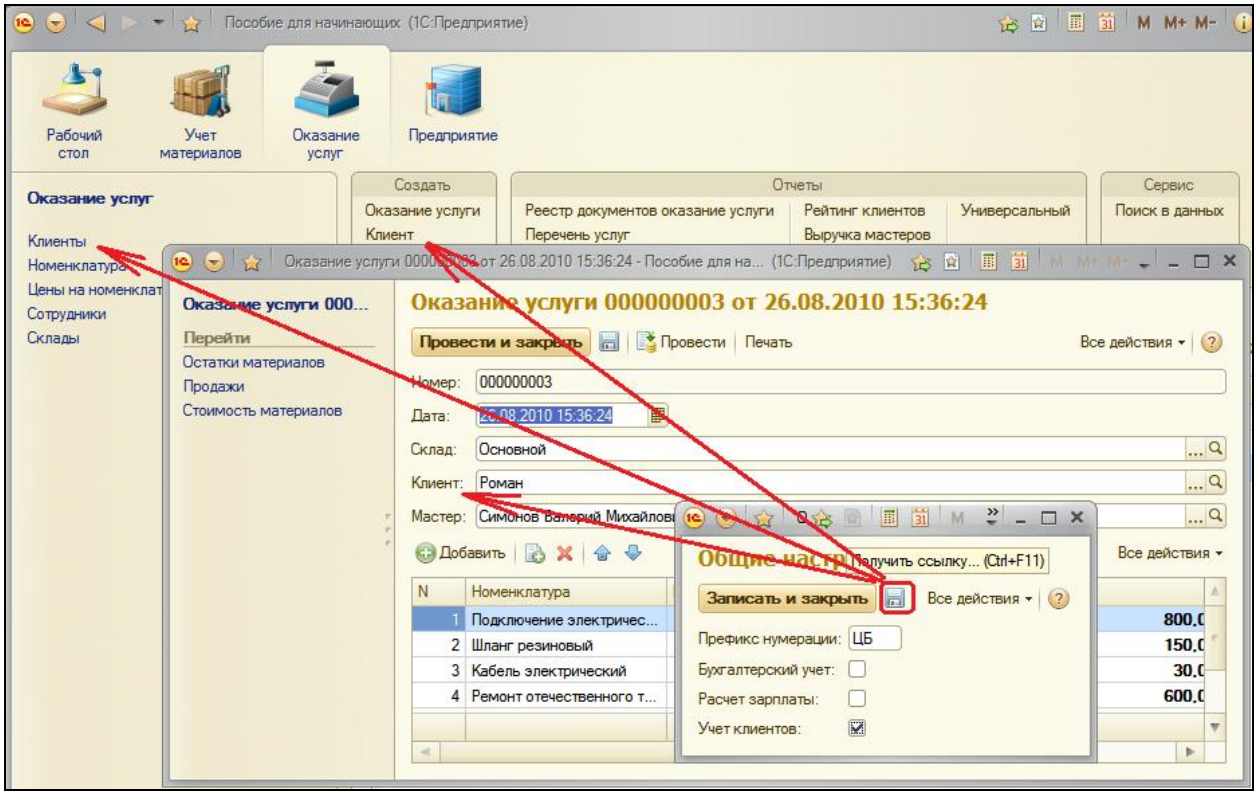
Запустим режим отладки. В разделе **Предприятие** выполним команду **Общие настройки**. В основном окне перейдем в раздел **Оказание услуг** и выполним команду **Оказание услуг**. Откроем любой документ **Оказание услуги**.

Функциональная опция **Учет клиентов** отключена. Поэтому в документе нет поля **Клиент**, в панели навигации раздела Оказание услуг нет команды **Клиенты**, а в ее панели действий нет команды создания нового клиента – **Клиент**.

В форме Общие настройки поставим флажок **Учет клиентов** и нажмем кнопку **Записать**.

Интерфейс прикладного решения изменится. Функциональная опция **Учет клиентов** включена. В документе появилось поле **Клиент**, в панели навигации и панели действий раздела **Оказание услуг** появились команды работы со справочником **Клиенты**.





Аналогичным образом вы можете самостоятельно переключить различные функциональные опции и посмотреть, как при этом меняется интерфейс прикладного решения.

На этом мы фактически завершили разработку нашей конфигурации.

Следующие две работы будут посвящены отдельным приемам разработки, которые часто используются в 1С:Предприятии 8.

Некоторые примеры мы будем выполнять в нескольких различных вариантах, поэтому какой из этих вариантов использовать в имеющейся конфигурации – дело вашего вкуса.

Контрольные вопросы

- ✓ Что такое функциональные опции и зачем они нужны.
- ✓ Как с помощью функциональных опций изменять интерфейс прикладного решения.

Практическая работа № 25

Подборы и ввод на основании (1:00)

Существует ряд приемов использования объектов, которые нельзя отнести только к одному виду объектов.

О таких приемах – подбор, ввод на основании – и пойдет речь в этой работе.

Организация подборов

Задача организации подбора заключается, как правило, в заполнении табличной части документа информацией, которую выбирает пользователь в списке какого-либо объекта.

Для иллюстрации механизма подбора информации в форме мы будем использовать задачу подбора элементов справочника в табличную часть документа как наиболее распространенную.

Поскольку механизм подбора реализован на уровне форм, то в других случаях просто будут задействованы иные прикладные объекты. Сама механика подбора не изменится.

Для организации подбора в форму документа следует открыть форму справочника как подчиненную форме документа в целом либо одному из элементов формы. Способ получения формы справочника может быть любым, так же как и сама форма справочника, которая будет использована. Важно лишь то, что эта форма должна быть открыта как подчиненная.

Результат подбора будет доступен в обработчике события **ОбработкаВыбора** формы документа или элемента формы (в зависимости от того, чему мы подчиним форму справочника при открытии).

Событие **ОбработкаВыбора** в форме документа будет вызвано в двух случаях:

- Когда в форме справочника будет выполнен интерактивный выбор,
- Когда в форме справочника будет вызван метод **ОповеститьОВыборе()**.

Различные способы подбора мы покажем на примере подбора элементов справочника **Номенклатура** в документ **ПриходнаяНакладная**.

Одиночный подбор

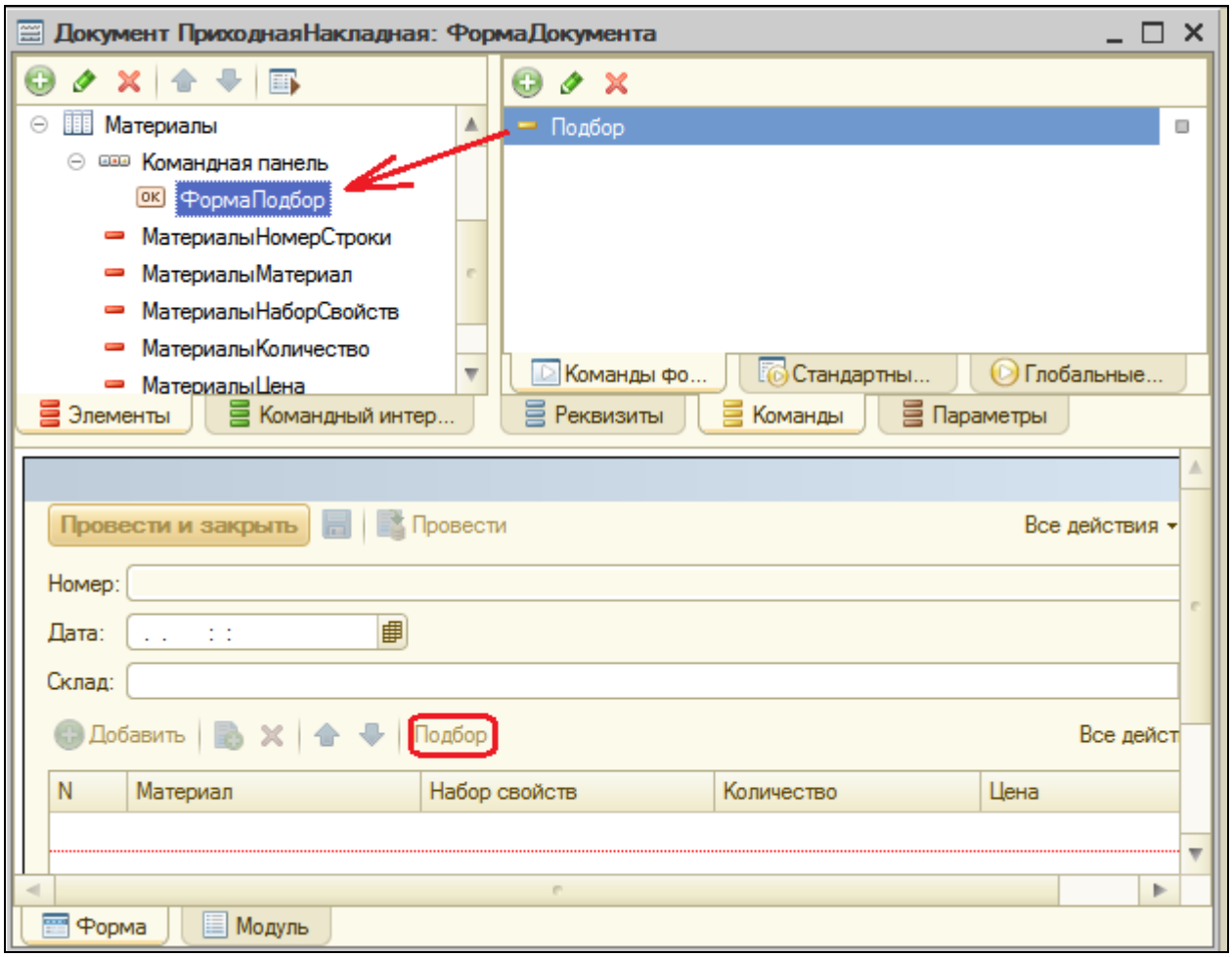
При одиночном подборе форма справочника будет закрываться сразу после выбора элемента. Для выбора следующего элемента необходимо будет снова инициировать подбор.

В режиме Конфигуратор

Откроем форму документа **ПриходнаяНакладная**.

На закладке **Команды** создадим команду **Подбор** и в открывшемся окне свойств нажмем кнопку открытия в строке **Действие**.

Шаблон обработчика события выполнения этой команды заполнять пока не будем, а перейдем на закладку **Форма** и перетащим эту команду в окно элементов формы, в командную панель таблицы **Материалы**.



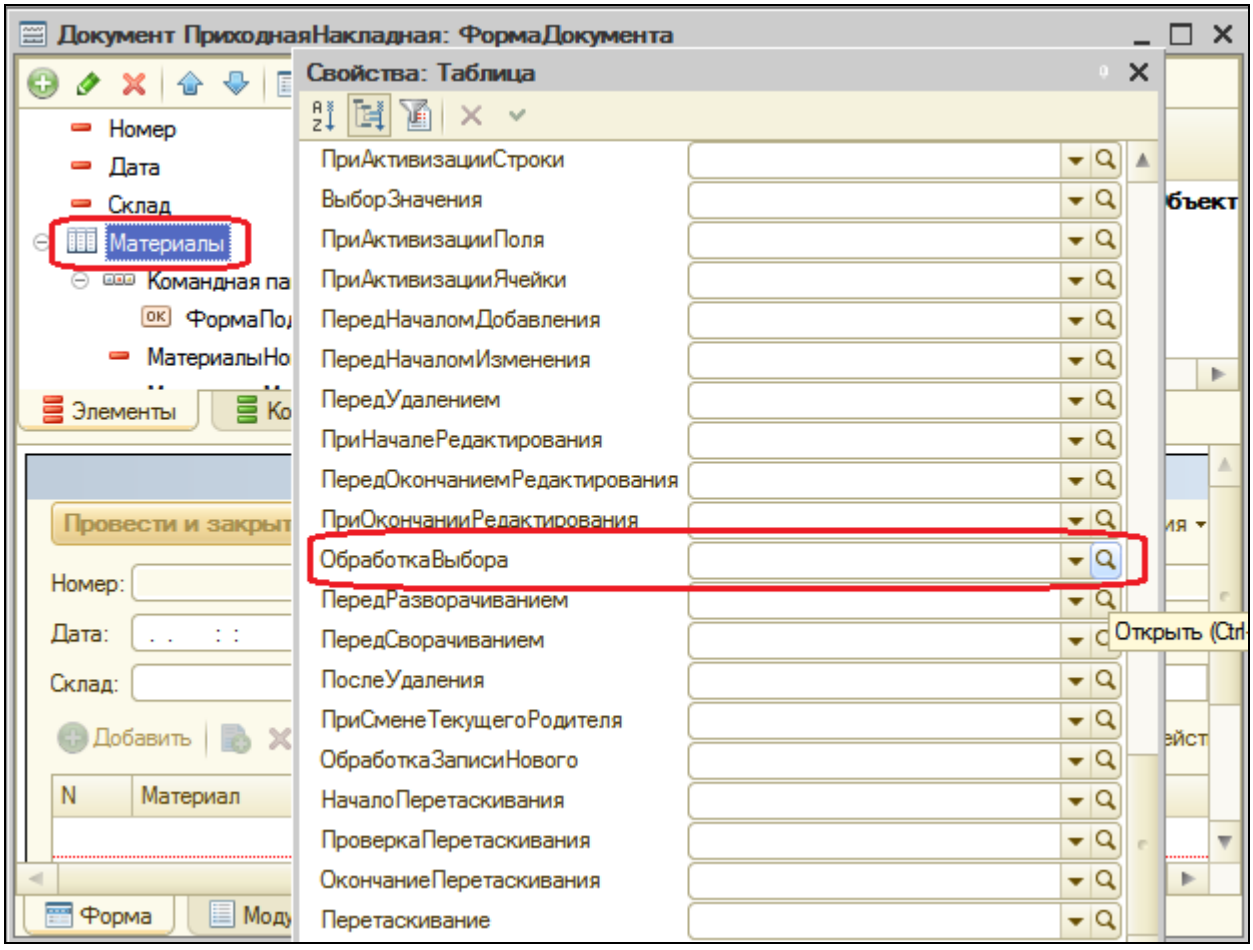
В форме документа, в обработчик события нажатия кнопки **Подбор**, добавим текст:

```
Процедура Подбор(Команда)
	ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ,
Элементы.Материалы);
КонецПроцедуры
```

В этой процедуре мы открываем форму выбора для справочника **Номенклатура**, указывая, что она подчинена таблице **Материалы** формы документа **ПриходнаяНакладная** (**Элементы.Материалы**).

При выборе из формы выбора справочника значение будет передано в обработчик события **ОбработкаВыбора** таблицы формы **Материалы**, т.к. она является владельцем открытой формы выбора.

Поэтому откроем свойства таблицы **Материалы** в дереве элементов формы и создадим обработчик события **ОбработкаВыбора**.



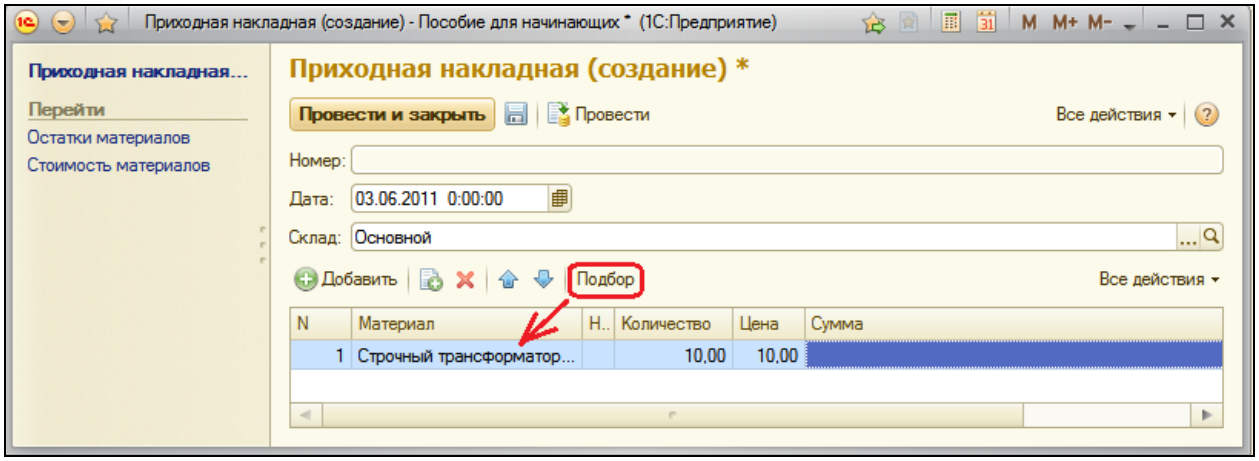

```
&НаКлиенте
Процедура МатериалыОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
	НоваяСтрока = Объект.Материалы.Добавить();
	НоваяСтрока.Материал = ВыбранноеЗначение;
КонецПроцедуры
```

В этой процедуре мы добавляем новую строку в таблицу **Материалы** и присваиваем колонке **Материал** в новой строке выбранное в форме выбора справочника значение. Это значение передается в обработчик события в параметре **ВыбранноеЗначение**.

В режиме 1С:Предприятие

Запустим режим отладки, перейдем в раздел **Учет материалов** и создадим новую приходную накладную командой **Приходная накладная**.

В командной панели списка нажмем **Подбор** и двойным щелчком подберем в накладную один материал.



Множественный подбор

В режиме Конфигуратор

При множественном подборе форма справочника будет открыта до тех пор, пока пользователь не закроет ее или не будет вызван метод формы **Закрыть()**.

В форме документа **ПриходнаяНакладная**, в обработчике события нажатия кнопки **Подбор** заменим прежний текст новым:

```
&НаКлиенте
Процедура Подбор(Команда)
    ПараметрыФормы = Новый Структура("ЗакрыватьПриВыборе", Ложь);
ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ПараметрыФормы,
Элементы.Материалы);
КонецПроцедуры
```

При открытии формы мы используем ее *параметры*.

Параметры формы нужны для того, чтобы открыть форму в некотором нужном нам состоянии. Параметры формы представляют собой структуру. Каждый элемент этой структуры описывает один параметр формы. Ключ элемента – это имя параметра формы.

Такую структуру мы передаем в метод **ОткрытьФорму()** вторым параметром (переменная **ПараметрыФормы**).

Предварительно мы эту структуру формируем. В ней у нас всего один элемент с ключом **ЗакрыватьПриВыборе().**

Таким образом, передавая эту структуру в метод **ОткрытьФорму()**, мы устанавливаем параметр открываемой формы **ЗакрыватьПриВыборе** в значение **Ложь**.

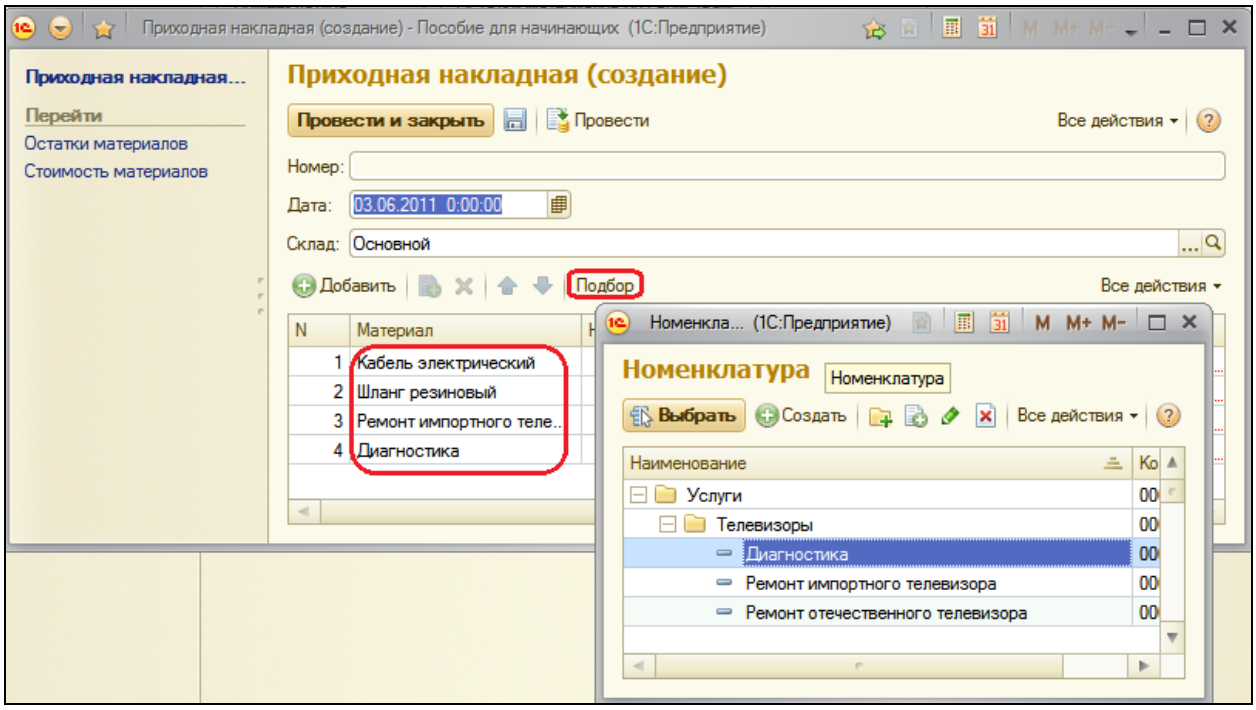
Это значит, что созданная форма выбора после двойного щелчка на номенклатуре закрываться не будет.

В режиме 1С:Предприятие

Запустим режим отладки. Перейдем в раздел **Учет материалов** и создадим новую приходную накладную.

В командной панели списка нажмем **Подбор** и двойным щелчком мыши подберем в накладную несколько материалов и несколько услуг.

Когда все выберем, закром окно с формой выбора.



Подбор с использованием множественного выбора

В режиме Конфигуратор

Еще одним способом организации подбора является возможность выделения в списке сразу нескольких строк.

Режим множественного выделения в списке устанавливается, как правило, во всех формах списков по умолчанию. Однако возможность выбрать сразу несколько элементов из писка по умолчанию, как правило, отключена.

Поэтому для того, чтобы в форме списка справочника **Номенклатура** можно было не только отметить, но и выбрать сразу несколько элементов, мы снова воспользуемся одним из параметров расширения формы динамического списка – **МножественныйВыбор**.

В форме документа **ПриходнаяНакладная** заменим текст обработчика события нажатия кнопки **Подбор**:

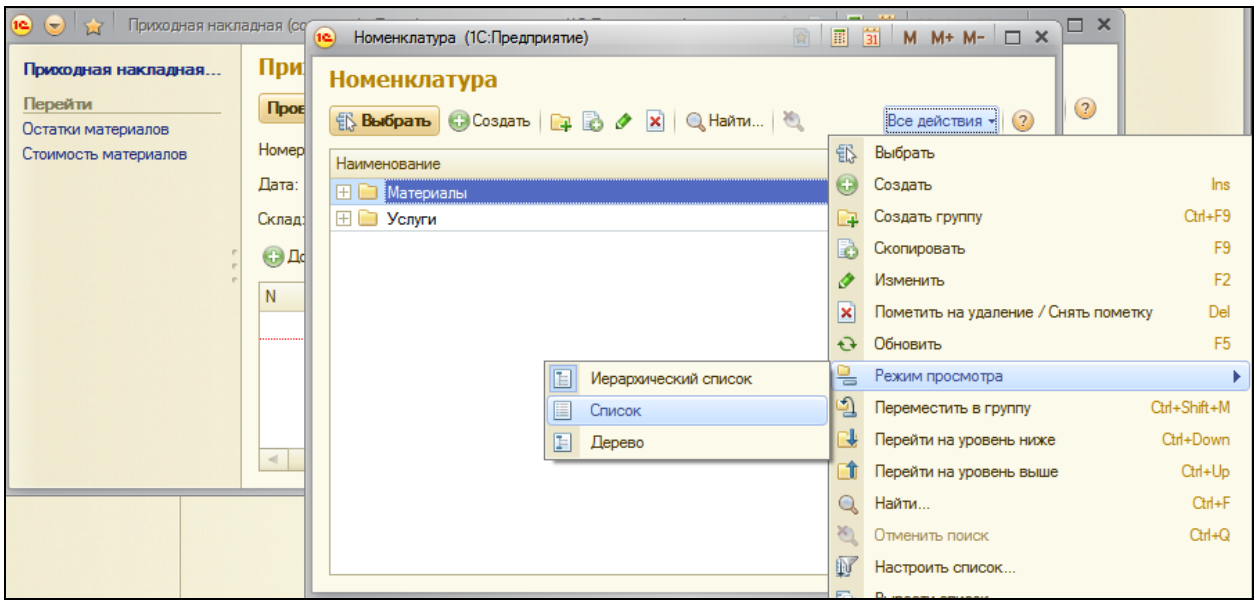
```
&НаКлиенте
Процедура Подбор(Команда)
        ПараметрыФормы = Новый Структура("МножественныйВыбор", Истина);
        ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора",
ПараметрыФормы, Элементы.Материалы);

КонецПроцедуры
```

При множественном выборе форма будет возвращать уже не один элемент, а массив элементов. Поэтому в обработчик события **ОбработкаВыбора** добавим обход массива переданных элементов и удалим предыдущие строки.

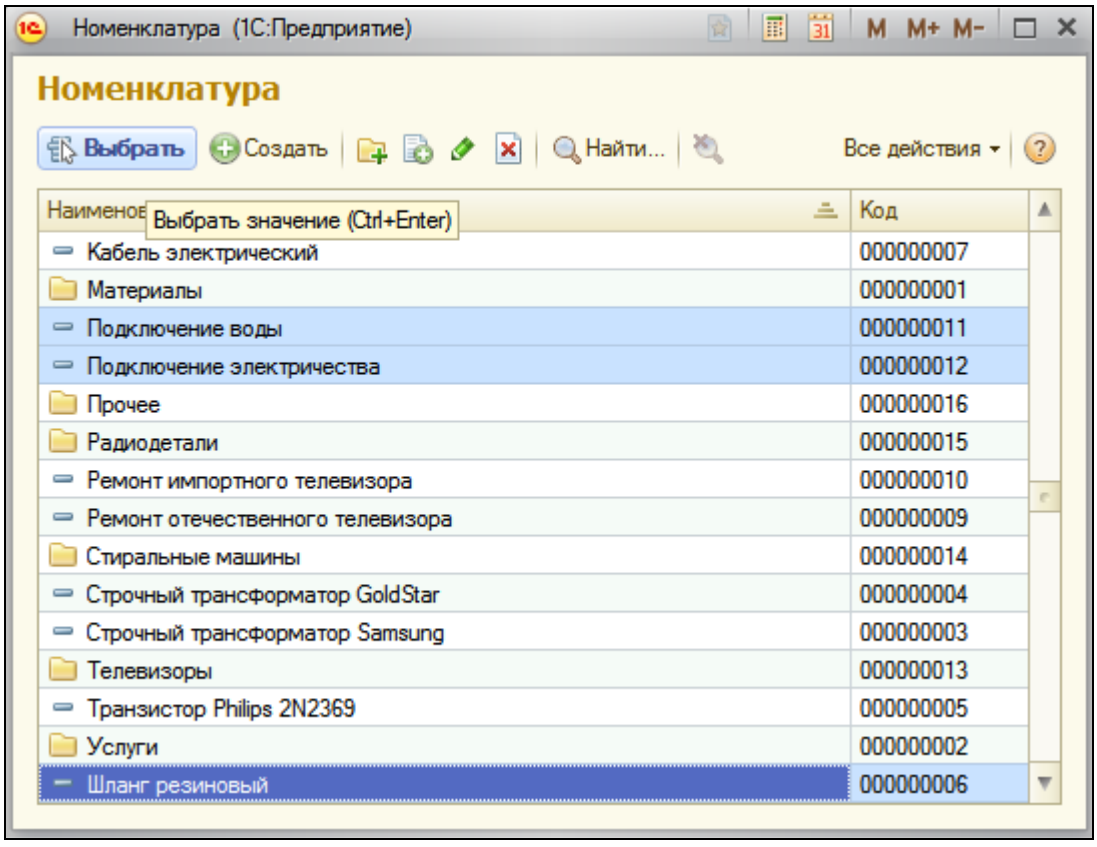
```
Процедура МатериалыОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
            Для Каждого ВыбранныйЭлемент Из ВыбранноеЗначение Цикл
      НоваяСтрока = Объект.Материалы.Добавить();
            НоваяСтрока.Материал = ВыбранныйЭлемент;
        КонецЦикла;
КонецПроцедуры
```

В режиме 1С:Предприятие

Запустим отладку, в разделе **Учет материалов** создадим новую приходную накладную и нажмем **Подбор**. Для удобства в открывшейся форме выбора справочника Номенклатура установим режим просмотра в виде списка – **Все действия - Режим просмотра – Список**.



Удерживая **Ctrl**, выделим в списке несколько товаров и несколько услуг и нажмем кнопку **Выбрать**. Отмеченные элементы появятся в табличной части документа.



Множественный подбор с использованием множественного выбора

В режиме Конфигуратор

Последний способ подбора будет сочетать в себе оба рассмотренных ранее способа. Мы будем отмечать сразу несколько элементов справочника и подбирать их в документ без закрытия формы выбора. Затем снова отмечать несколько элементов справочника и подбирать их в документ.

Для этого нам будет необходимо при открытии формы выбора установить оба параметра: **ЗакрыватьПриВыборе** и **МножественныйВыбор**.

Процедура нажатия клавиши **Подбор** будет выглядеть так:

```
&НаКлиенте
Процедура Подбор(Команда)
        ПараметрыФормы = Новый Структура("ЗакрыватьПриВыборе, МножественныйВыбор", Ложь, Истина);
        ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ПараметрыФормы, Элементы.Материалы);

КонецПроцедуры
```

В режиме 1С:Предприятие

Запустите отладку. Создайте новую приходную накладную и нажмите кнопку **Подбор**.

Откройте группу **Услуги – Телевизоры**, выделите в ней все услуги и нажмите кнопку **Выбрать**.

Откройте группу **Материалы – Прочее**, выделите в ней все материалы и нажмите **Выбрать**. Закройте окно формы выбора.

Использование метода «Оповестить о выборе()»

Этот метод используется, когда алгоритм формирования данных подбора сложен, и кроме собственно выбора элементов справочника и от пользователя требуется указание некоторой дополнительной информации. В этом случае метод вызывается, когда необходимая информация подбора сформирована.

Метод **ОповеститьОВыборе()** посылает оповещение владельцу формы о выполнении выбора или подбора, передает ему выбранное значение и закрывает форму, если она открыта не в режиме множественного выбора.

Также метод может использоваться, когда требуется передать в форму документа не только выбранный элемент справочника (или массив элементов), а некоторую произвольную структуру данных.

Ввод на основании

Механизм ввода на основании может быть использован для ввода новых объектов различного типа (документа, справочники, планы видов характеристик и т.д.). Мы рассмотрим этот механизм на примере ввода новых документов как наиболее распространенном.

Для каждого объекта конфигурации Документ можно разрешить его ввод на основании других объектов базы данных и возможность являться основанием для других объектов.

Действия по заполнению реквизитов при вводе на основании должны быть описаны в модуле объекта Документ, в обработчике события **ОбработкаЗаполнения**.

Это можно сделать вручную или с помощью конструктора ввода на основании, который позволяет визуальными средствами конструировать текст обработчика.

Рассмотрим пример, когда документ **ОказаниеУслуги** буде вводиться на основании справочника **Клиенты**.

Команда ввода на основании

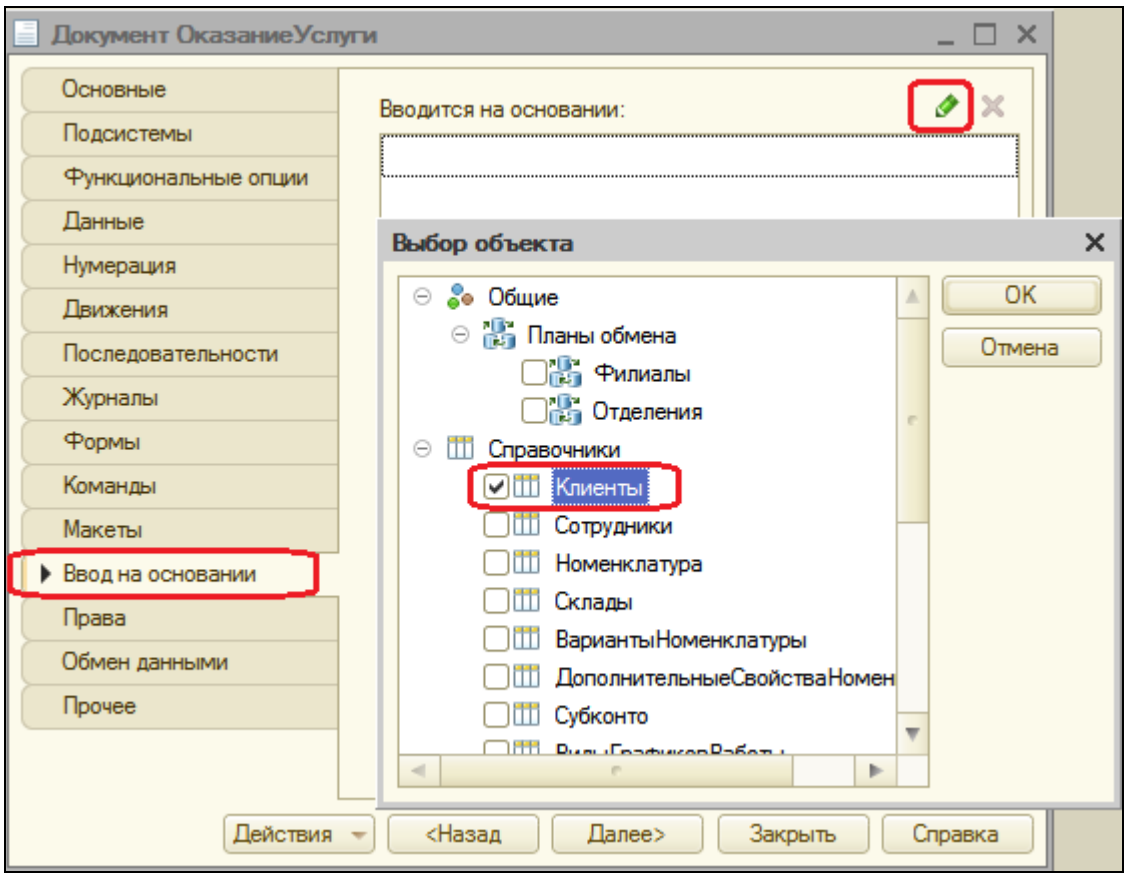
В режиме Конфигуратор

Откроем окно редактирования документа **ОказаниеУслуги** и добавим новый реквизит документа – **ОбъектОснование** с типом **СправочникСсылка.Клиенты**.

Создание такого реквизита не является обязательной частью механизма ввода на основании и понадобится нам только для того, чтобы впоследствии построить цепочку зависимых документов.

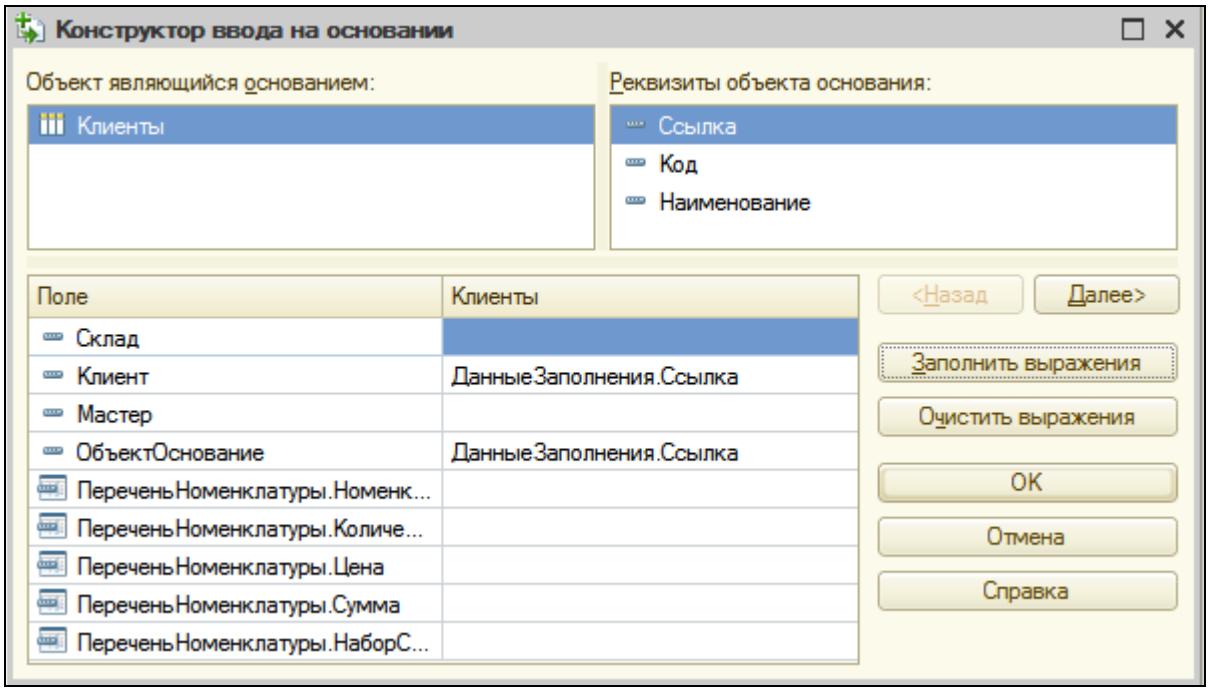
Перейдем на закладку **Ввод на основании** в окне редактирования документа и определим состав документов, на основании которых может вводиться документ **ОказаниеУслуги** и основанием для которых он может являться.

Нажмем кнопку **Редактировать элемент списка** над списком **Вводится на основании** и выберем справочник **Клиенты**.



Затем вызовем конструктор ввода на основании и зададим значения реквизитов документа, создаваемого на основании.

Для этого воспользуемся кнопкой **Заполнить выражения**. Нажмем ОК.



В модуле документа будет сформирован текст обработчика события **ОбработкаЗаполнения**.

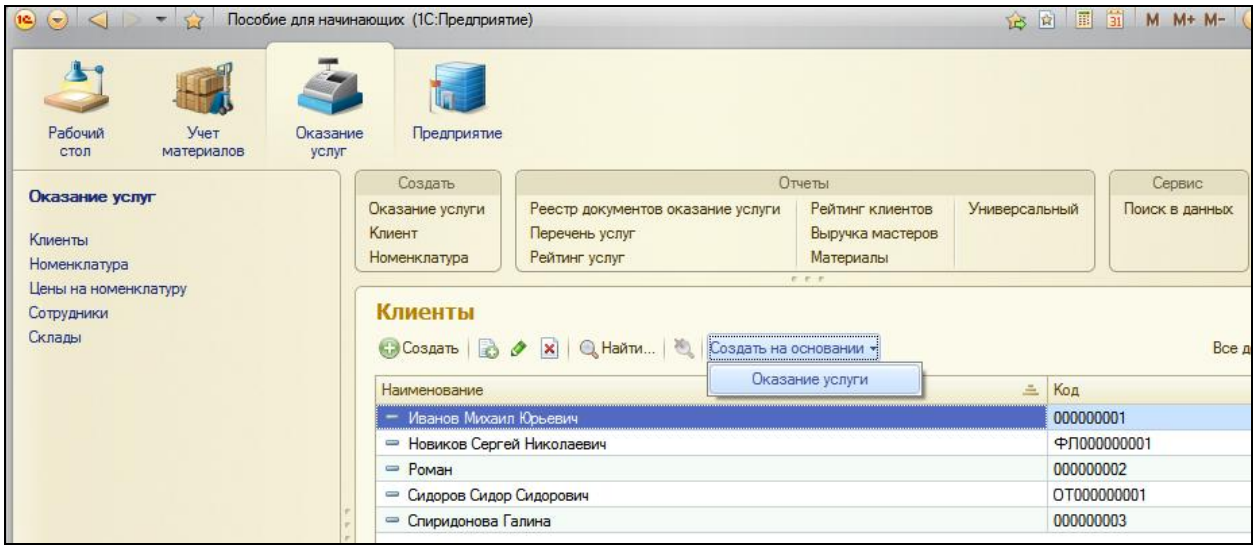
```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)
	//{{__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
	Если ТипЗнч(ДанныеЗаполнения) = Тип("СправочникСсылка.Клиенты") Тогда
		// Заполнение шапки
		Клиент = ДанныеЗаполнения.Ссылка;
		ОбъектОснование = ДанныеЗаполнения.Ссылка;
	КонецЕсли;
	//}}__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
КонецПроцедуры
```

Как видите, для каждого типа объекта-основания формируется своя ветка условия Если…, в которой происходит заполнение реквизитов нового документа.

В режиме 1С:Предприятие

Запустим режим отладки, откроем список клиентов. В командной панели формы списка справочника **Клиенты** появилась команда **Создать на основании**.

Выделив нужного клиента и выполнив команду **Создать на основании – Оказание услуги**, создадим новый документ **Оказание услуги**, где в качестве клиента будет выбран выделенный в списке справочника клиент.



Введите самостоятельно еще несколько документов на основании какого-либо клиента.

Объекты, введенные на основании

Хотя платформа содержит механизмы создания одних объектов на основании других, она не содержит специальных механизмов для анализа цепочек связанных объектов.

Для решения подобной задачи мы дадим некоторые рекомендации, которые могут быть положены в основу конкретного решения.

Для построения цепочек связанных объектов необходимо у каждого объекта, который будет вводиться на основании, создать служебный реквизит для хранения ссылки на объект-основание. Затем следует создать объект конфигурации *КритерийОтбора*, который будет использоваться для установки отбора по требуемому значению служебного реквизита.

В дальнейшем для получения всех объектов, введенных на основании, достаточно будет установить нужное значение отбора в критерии отбора.

Критерий отбора

Объект конфигурации КритерийОтбора предназначен для задания правил, по которым может выполняться отбор объектов.

Этот объект используется в случае поиска различной информации, когда, например, требуется отобрать все документы, в которых используется (в реквизитах и в табличных частях) определенный контрагент.

При этом можно учитывать также и другие условия отбора информации (например, поиск ведется только среди проведенных документов или в определенном интервале дат).

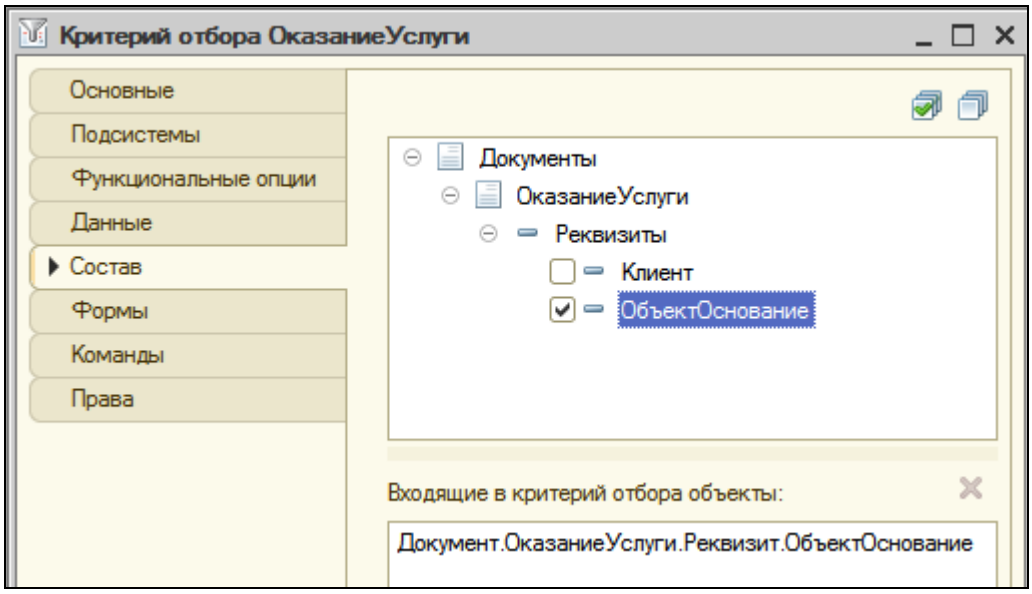
Получение объектов, введенных на основании

Поскольку задача получения всех объектов, введенных на основании какого-либо другого объекта, чаще всего возникает при анализе документов, мы рассмотрим применение описанной выше методики на примере получения списка документов, введенных на основании элемента справочника **Клиенты**.

Раскроем ветвь **Общие** и создадим новый объект **КритерийОтбора** с именем **ОказаниеУслуги**.

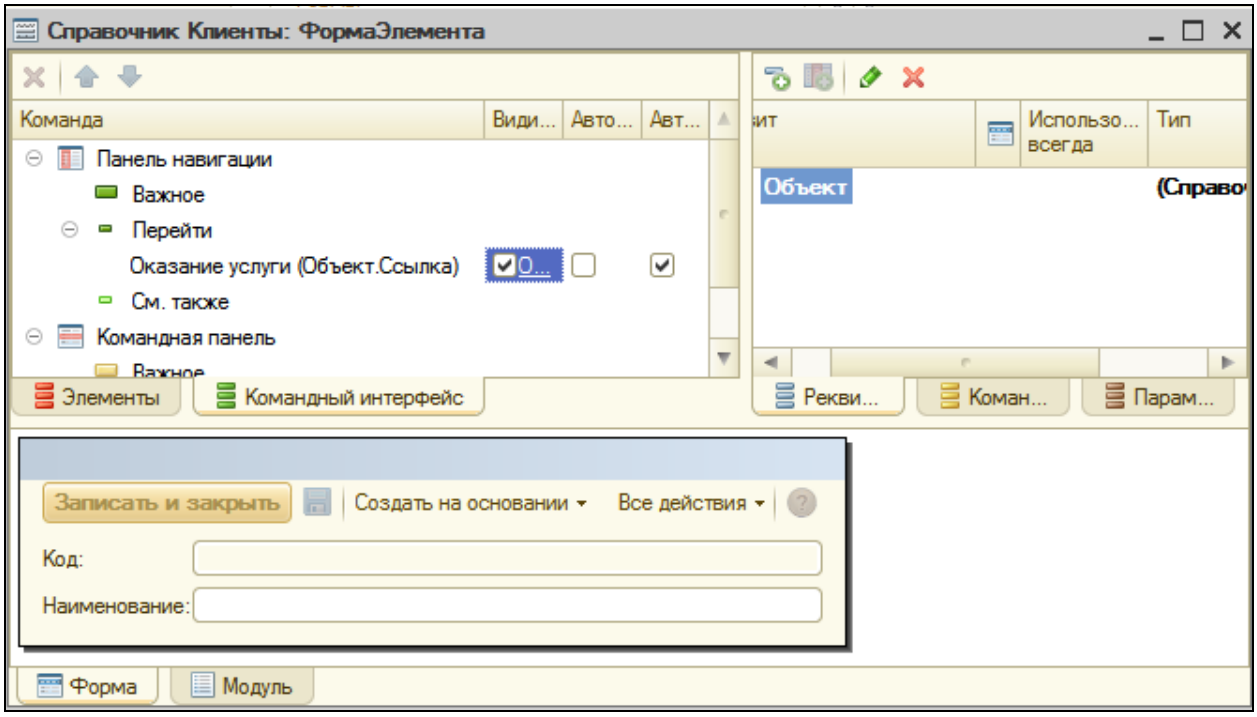
На закладке **Данные** выберем тип используемого критерия – **СправочникСсылка.Клиенты**.

На закладке **Состав** в качестве объектов, входящих в критерий, выберем реквизит **ОбъектОснование** документа **ОказаниеУслуги**.



После этого в панели навигации формы элемента справочника **Клиенты**, в группе **Перейти**, появится команда для открытия критерия отбора.

Создадим эту форму: в Справочнике **Клиенты – создать форму элемента** и на закладке **Командный интерфейс** установим видимость команды **Оказание услуги**.

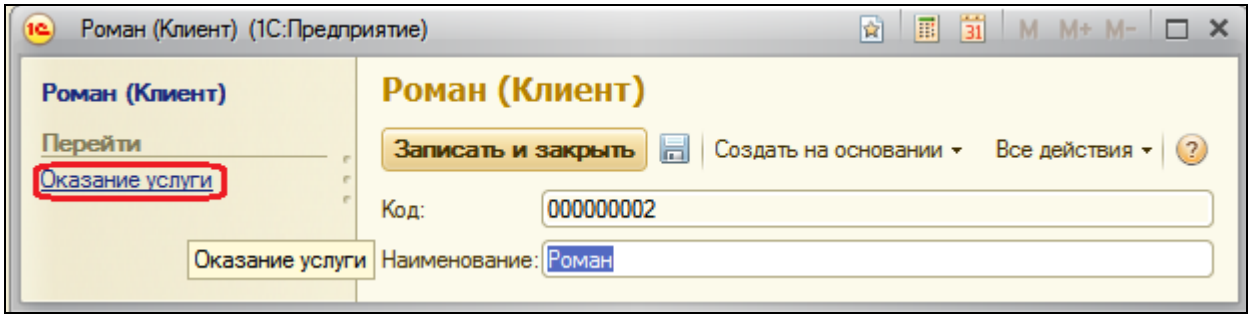


В режиме 1С:Предприятие

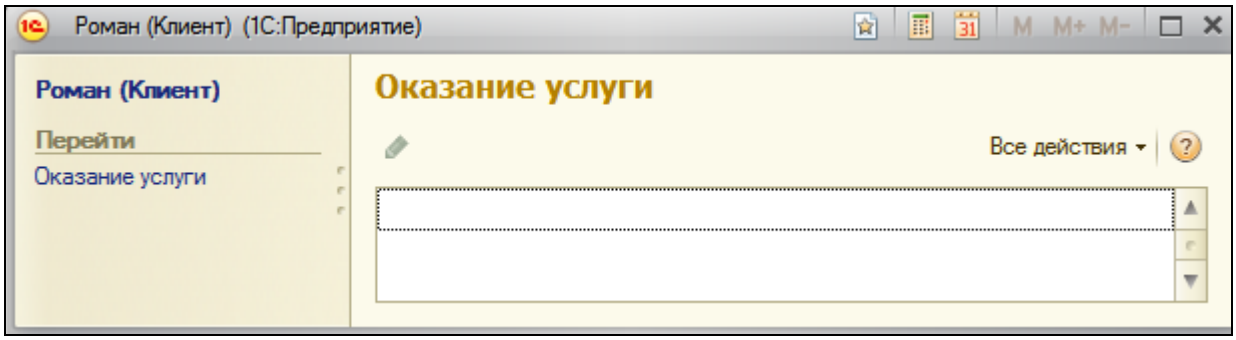
Запустим режим отладки и проверим работу критерия отбора.

В форме списка клиентов выделим клиента **Роман**, на основании которого мы создавали документы **Оказание услуги**.

Откроем форму этого клиента. В панели навигации появилась команда **Оказание услуги** для открытия формы списка созданного нами критерия отбора с установленным отбором по открытому элементу справочника **Клиенты**. Выполним эту команду.



Мы видим в этом списке документы **Оказание услуги**, созданные на основании клиента **Роман**. К содержимому документа можно перейти, нажав соответствующую ссылку в списке документов. (У меня список пуст, потому что я не проводил документы с его участием, может у вас будет по-другому).



Контрольные вопросы

- ✓ Что такое подбор.
- ✓ Как организовать различные виды подбора в табличную часть формы документа.
- ✓ Что такое ввод на основании
- ✓ Как организовать ввод одних объектов на основании других.
- ✓ Как с помощью критерия отбора вывести список объектов введенных на основании текущего объекта.

Практическая работа № 26

Приемы разработки форм (2:10)

В этой работе мы рассмотрим несколько типичных приемов работы с формами объектов. Начнем с рассказа о том, каким образом данные отображаются в формах, и о тех типах данных, которые при этом используются.

Данные и элементы формы

Важной особенностью платформы 1С:Предприятие 8 является механизм представления данных в формах. Ключевым моментом является то, что принадлежность формы к тому или иному объекту никоим образом не определяет состав данных, которые форма будет отображать.

Например, можно создать общую форму, которая не будет подчинена ни одному из объектов, но которая, в зависимости от содержимого, будет либо отображать список справочника, либо позволять редактировать документ и т.п. Однако такую форму уже нельзя будет назначить основной для выполнения определенных действий.

Форма сама по себе и ее элементы обособлены от объектов конфигурации. Для отображения данных в форме, необходимо задать связь самой формы и ее элементов с данными. Если связь не задана, то элементы вообще не будут отображены в форме (кроме элементов оформления).

При использовании конструктора форм конфигуратор создает такие связи автоматически. Если разработчик создает форму вручную, он может определить эти связи путем задания свойств формы и ее элементов.

Связь элементов формы с данными, которые они должны отображать, задается в свойстве **ПутьКДанным**.

Связь формы и ее элементов с данными осуществляется при помощи реквизитов формы. Список существующих реквизитов формы доступен на закладке **Реквизиты** окна редактора формы.

Среди всех реквизитов формы, как правило, существует один основной реквизит (он выделен жирным шрифтом). Он определяет источник данных для формы в целом. От типа значения основного реквизита

зависит не только то, какие данные будут отображены в элементах формы, но и поведение самой формы.

Например, если основному реквизиту формы указать тип значения **Документ.Объект.ПриходнаяНакладная**, то при закрытии формы программа будет запрашивать подтверждение записи и проведения документа.

Если же основному реквизиту формы указать тип значения **СправочникОбъект.Клиенты**, то подобного подтверждения не будет.

Похожее влияние источники данных оказывают и на элементы формы.

Например, состав колонок таблицы, источником данных которой является реквизит формы с типом значения **ДинамическийСписок**, будет различным, в зависимости от того, какой объект используется в качестве основной таблицы этого динамического списка (например, **РегистрНакопления.ОстаткиНоменклатуры** или **Справочник.Номенклатура**.

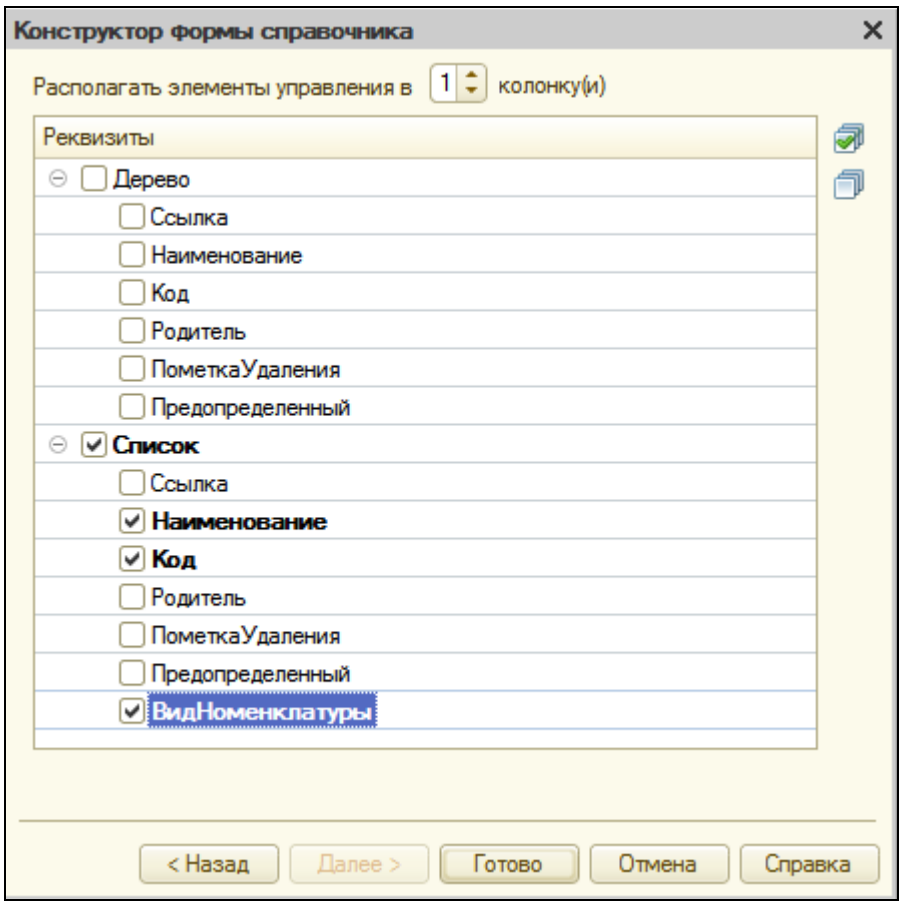
То же самое справедливо и для элемента формы **Командная панель**. При установленном свойстве командной панели **Автозаполнение** смена источника данных (источника действий) будет приводить к изменению состава команд, которые отображает командная панель.

Возможность связать форму и ее элементы с различными данными является причиной того, что у формы и ее элементов существует несколько *расширений*.

Расширение представляет собой набор дополнительных свойств, методов и событий, появляющихся у формы или ее элемента. Наличие расширения определяется типом данных, которые отображает форма/элемент, либо расположением элемента формы в других ее элементах.

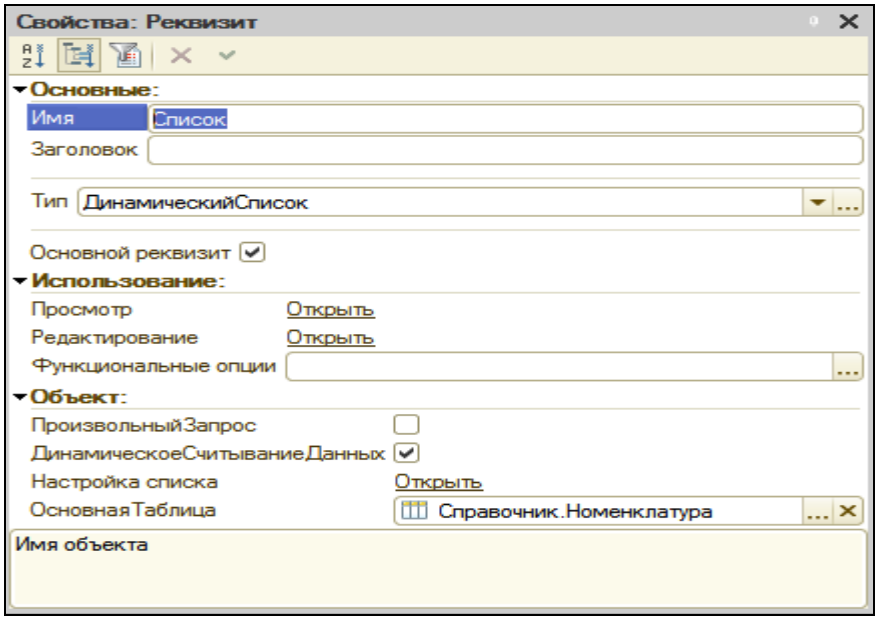
Чтобы подробнее познакомиться с этим механизмом, создадим основную форму списка справочника **Номенклатура**. При этом в конструкторе формы мы не будем нажимать сразу кнопку Готово, как делали раньше.

Нажмем кнопку **Далее** и кроме полей **Наименование** и **Код** включим в состав таблицы Список еще одно поле – **ВидНоменклатуры**. И затем нажмем **Готово**.



С механизмом расширений мы будем знакомиться на примере поля **ВидНоменклатуры**, расположенного в таблице **Список** формы списка справочника **Номенклатура**.

Сама форма отображает данные объекта **ДинамическийСписок** (основной реквизит формы – **Список** – имеет тип **ДинамическийСписок**).



Поэтому к свойствам, методам и событиям объекта встроенного языка **УправляемаяФорма** добавляется **Расширение управляемой формы для динамического списка**. В результате этого у формы появляются такие параметры, как **ТекущаяСтрока**, **Отбор** и т.п.

Теперь посмотрим на таблицу **Список**.

Поскольку в таблице отображается динамический список, то к свойствам, методам и событиям объекта встроенного языка **ТаблицаФормы** добавляется **Расширение таблицы формы для динамического списка**.

В результате у таблицы **Список** появляются такие свойства, как **АвтоОбновление**, **ОтображатьКорень** и т.д.

И в заключение посмотрим на поле **ВидНоменклатуры**.

Это поле связано с реквизитом типа **ПеречислениеСсылка.ВидыНоменклатуры** и является полем ввода.

Поэтому к свойствам, методам и событиям объекта встроенного языка **ПолеФормы** добавляется **Расширение поля формы для поля ввода**.

Типы данных формы

В управляемой форме можно выделить следующие категории типов, с которыми она работает:

- Типы встроенного языка, предназначенные для использования как в управляемых формах, так и вне их. Например, **Число**, **СправочникСсылка.<имя>**, **ГрафическаяСхема**, **ТабличныйДокумент** и т.д.
- Типы встроенного языка, предназначенные исключительно для того. Чтобы представить в форме данные прикладных объектов (справочников, документов и т.д.). Это такие типы, как **ДанныеФормыСтруктура**, **ДанныеФормыКоллекция** и другие.
- Тип **Динамический список**, который используется в управляемых формах для отображения списков прикладных объектов.

Все типы прикладных объектов не существуют на стороне тонкого и веб-клиентов, они существуют только на сервере. Однако данные этих объектов нужно отображать в управляемых формах.

Поэтому для представления в форме данных этих прикладных типов введены специальные типы данных, предназначенные для работы именно в управляемых формах. Используются следующие типы данных:

- **ДанныеФормыСтруктура** – содержит набор свойств произвольного типа. Свойствами могут быть другие структуры, коллекции или структуры с коллекциями. Таким типом представляется, например, в форме **СправочникОбъект**.
- **ДанныеФормыКоллекция** – это список типизированных значений, похожий на массив. Доступ к элементу коллекции осуществляется по индексу или идентификатору. Доступ по идентификатору может отсутствовать в некоторых случаях. Это обусловлено типом прикладного объекта, который представлен этой коллекцией. Идентификатором может быть любое целое число. Таким типом представляется, например, в форме табличная часть.
- **ДанныеФормыСтруктураСКоллекцией** – это объект, который представлен в виде структуры и коллекции одновременно. С ним можно обращаться как с любой из этих сущностей. Таким типом представляется, например, в форме набор записей.
- **ДанныеФормыДерево** – объект предназначен для хранения иерархических данных.

Фактически можно сказать, что данные формы – это унифицированное представление данных различных прикладных объектов, с которыми форма работает единообразно, и которые присутствуют и на сервере, и на клиенте. В редакторе формы (у реквизитов) вместо имен этих типов обычно отображаются те прикладные типы, данные которых содержат реквизит.

Например, если реквизит **Объект** содержит данные элемента справочника **Товары**, то в колонке **Тип** отображается не настоящий тип этого реквизита формы – **ДанныеФормыСтруктура**, а тип прикладного объекта, данные которого содержатся в этом реквизите – **СправочникОбъект.Клиенты**. Причем чтобы было понятно, что это «ненастоящий» тип реквизита, тип прикладного объекта показывается в круглых скобках.

Т.о. форма содержит некоторую проекцию данных прикладных объектов в виде своих собственных типов данных и автоматически выполняет преобразование между ними при необходимости.

Однако в случае если разработчик конфигурации реализует свой алгоритм обработки данных, то преобразование данных (из специализированных типов в прикладные и обратно) он должен выполнять самостоятельно.

Для конвертирования прикладных объектов в данные формы и обратно существует набор глобальных методов:

- **ЗначениеВДанныеФормы()** – преобразует объект прикладного типа в данные формы.
- **ДанныеФормыВЗначение()** – преобразует данные формы в объект прикладного типа.

Аналогичные методы, предназначенные для конвертирования значений реквизитов формы в прикладные объекты и обратно, существуют и у самой управляемой формы:

- **ЗначениеВРеквизитФормы()** – преобразует объект прикладного типа в реквизит управляемой формы.
- **РеквизитФормыВЗначение()** – преобразует реквизит в значение прикладного типа.

Методы, работающие с прикладными объектами, доступны только в серверных процедурах формы.

При выполнении стандартных действий формы с основным реквизитом (открытие формы, выполнение команды **Записать** и т.д.) преобразование выполняется автоматически.

Связанные списки

При создании прикладных решений часто возникает необходимость из какой-либо формы прикладного объекта перейти к информации, логически связанной с этим объектом.

Это может быть, например, список подчиненного справочника; регистры, в которых объект производит движения; регистры, где измерение с типом этого объекта указано как ведущее; критерии отбора, в которые входит этот тип; объекты, которые можно ввести на основании этого типа и т.д.

Одним из распространенных примеров использования, связанных списков является необходимость перейти из формы документа к списку движений, которые произвел документ в каком-либо регистре.

Все перечисленные ситуации платформа контролирует автоматически. На основании информации, содержащейся в объектах конфигурации, платформа автоматически создает в формах команды для перехода к связанной информации. Некоторые такие команды она сразу же делает видимыми, и они появляются в форме. Некоторые команды она только создает, но по умолчанию не включает их видимость.

Поэтому чаще всего, если в форме вы не видите команды перехода к связанной информации, которая должна быть, просто нужно включить ее видимость в интерфейсе формы.

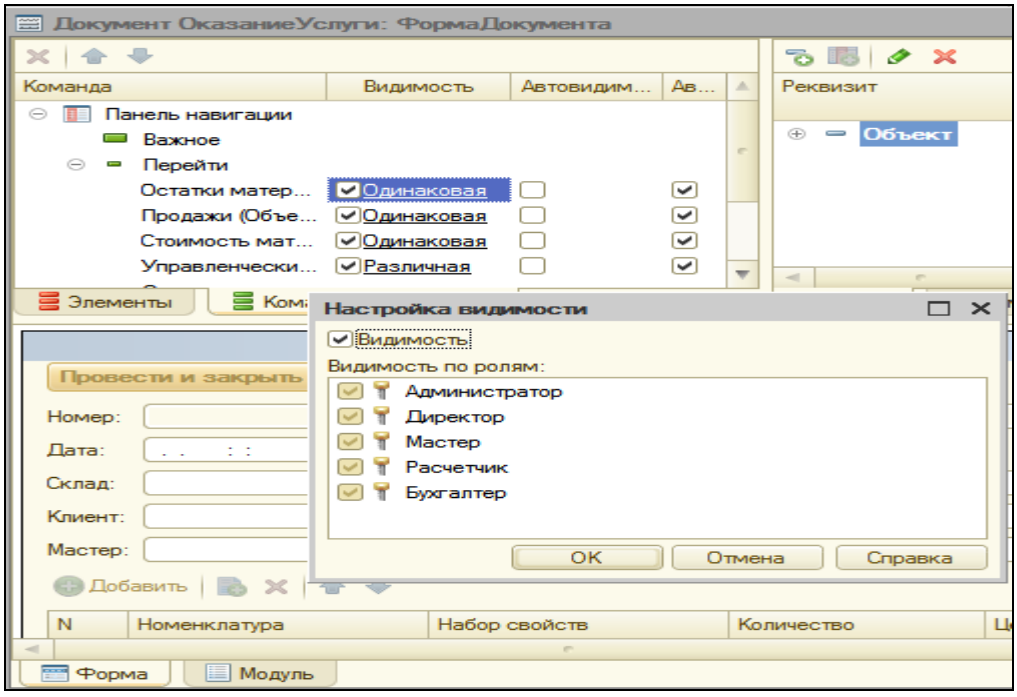
В режиме Конфигуратор

Для примера откроем форму документа **ОказаниеУслуги**.

В левом верхнем окне перейдем на закладку **Командный интерфейс**.

В разделе **Панель навигации**, в группе **Перейти**, уже находится набор таких команд. Это команды перехода к записям регистров, для которых этот документ является регистратором.

Мы можем установить общую видимость для этих команд, а можем более точно настроить видимость для каждой отдельной роли, которая есть в нашей конфигурации.

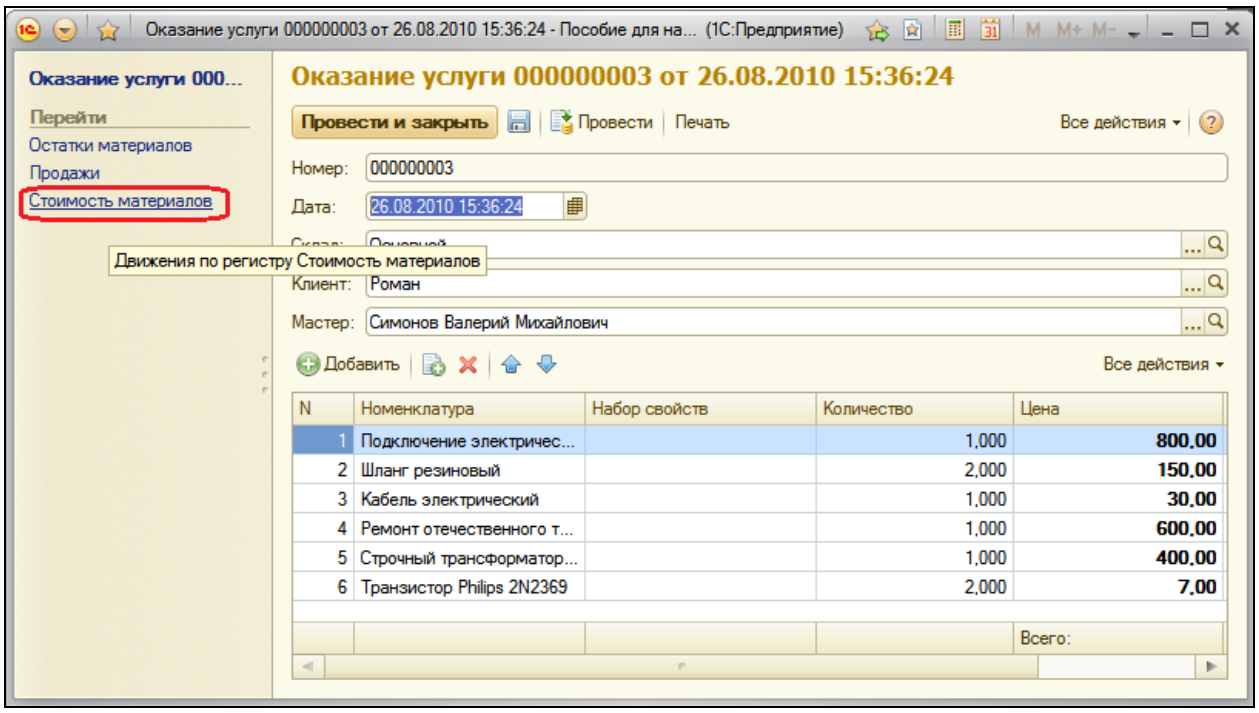


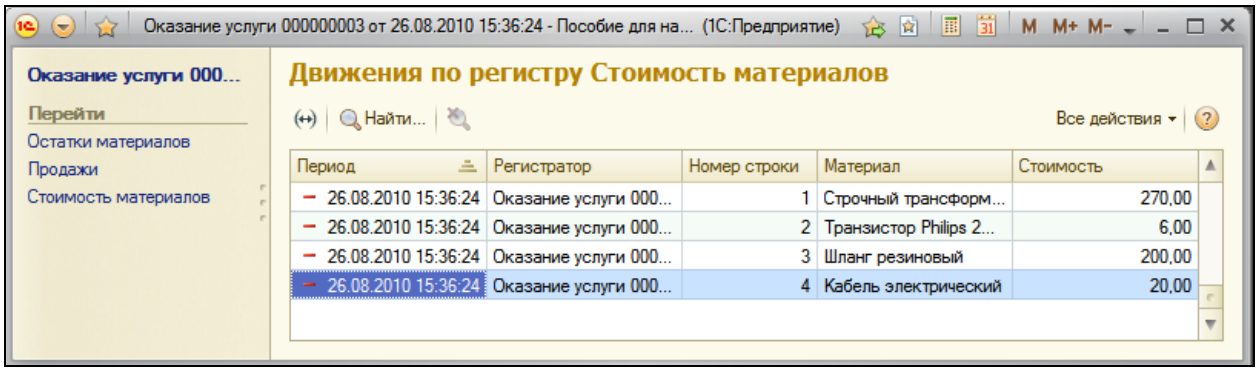
В режиме 1С:Предприятие

В режиме отладки откроем один из документов **Оказание услуги**.

В панели навигации формы мы видим команды перехода к списку записей регистров, связанных с открытым документом.

Например, выполним команду **Стоимость материалов** и перейдем к движениям, которые произвел в этом регистре наш документ.





Оформление строк в форме списка

Одним из полезных свойств формы списка является возможность настройки оформления его строк. Для иллюстрации этой возможности мы воспользуемся формой списка справочника Номенклатура и придадим ей нестандартный вид.

В режиме Конфигуратор

Откроем в конфигураторе форму списка справочника **Номенклатур**а и создадим обработчик события формы **ПриСозданииНаСервере**.

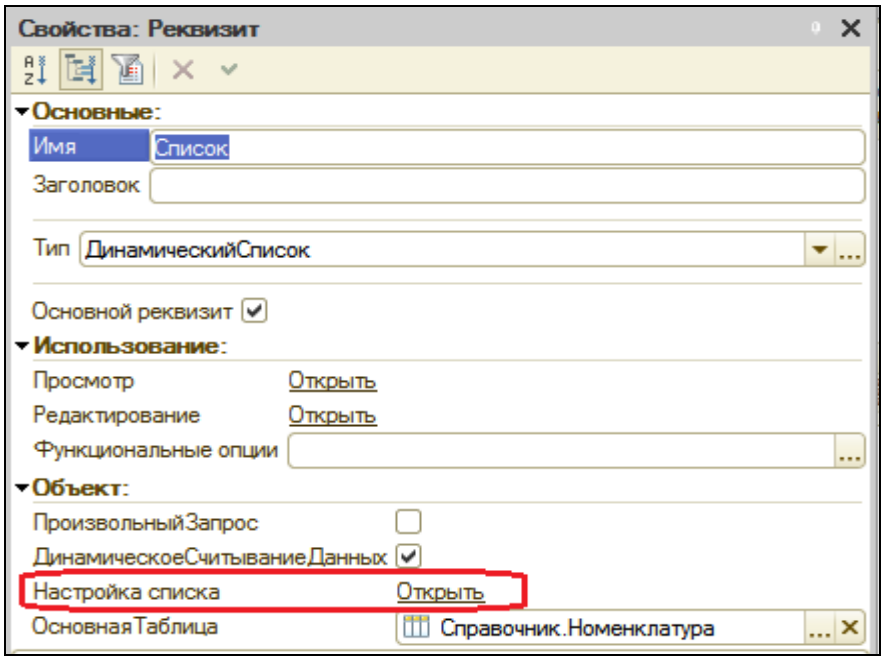
```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

	СписокСправочника = Элементы.Список;

	// Задать режим отображения справочника
	СписокСправочника.Отображение = ОтображениеТаблицы.Список;

	// Скрыть линии сетки
	СписокСправочника.ВертикальныеЛинии = Ложь;
	СписокСправочника.ГоризонтальныеЛинии = Ложь;

КонецПроцедуры
```

В этой процедуре мы сначала представляем список в виде обычного, а не иерархического списка. Это сделано для большей наглядности, чтобы материалы отображались вперемешку с услугами.

Затем мы скрываем линии, разделяющие колонки и строки таблицы списка.

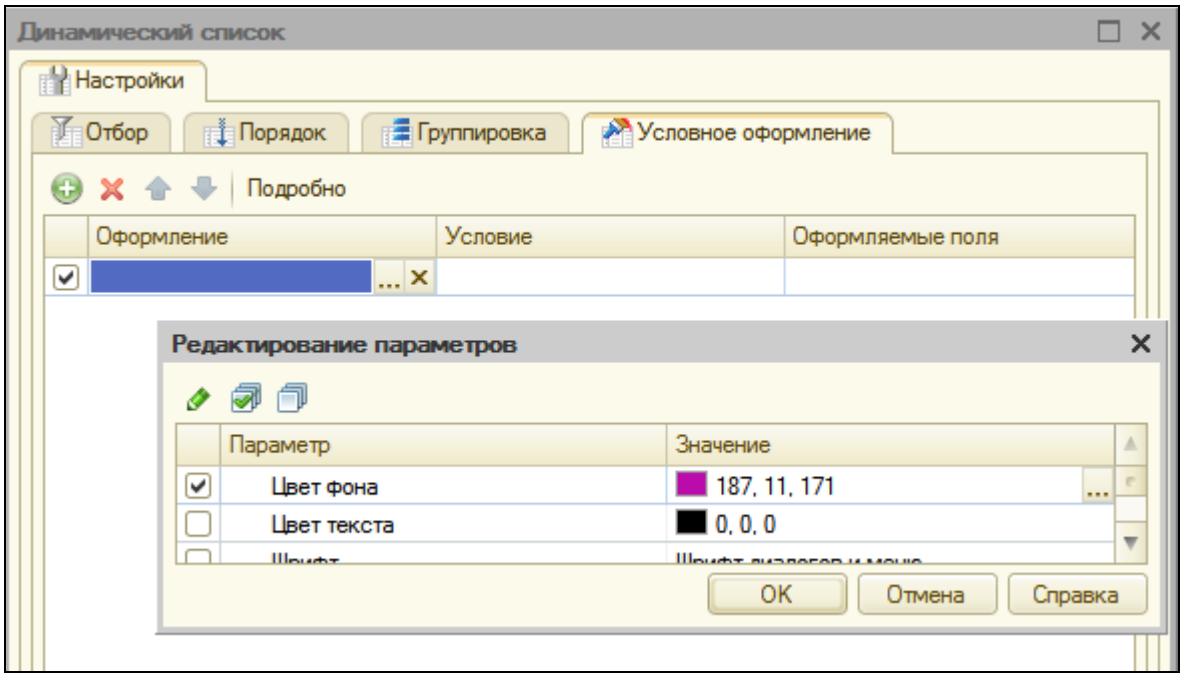
Теперь настроим условное оформление строк списка. Для этого вызовем окно свойств основного реквизита формы **Список**.

В строке **Настройка списка** нажмем **Открыть**.



Поскольку реквизит имеет тип **ДинамическийСписок**, который построен на основе системы компоновки данных, то мы можем настроить для него **Отбор**, **Порядок**, **Группировку** и **УсловноеОформление**, аналогично тому, как это делалось в отчетах.

В открывшемся окне настройки динамического списка перейдем на закладку **Условное оформление** и нажмем кнопку **Добавить** в командной панели окна. Нажмем кнопку выбора в поле **Оформление** и установим сиреневый цвет фона. Нажмем ОК.

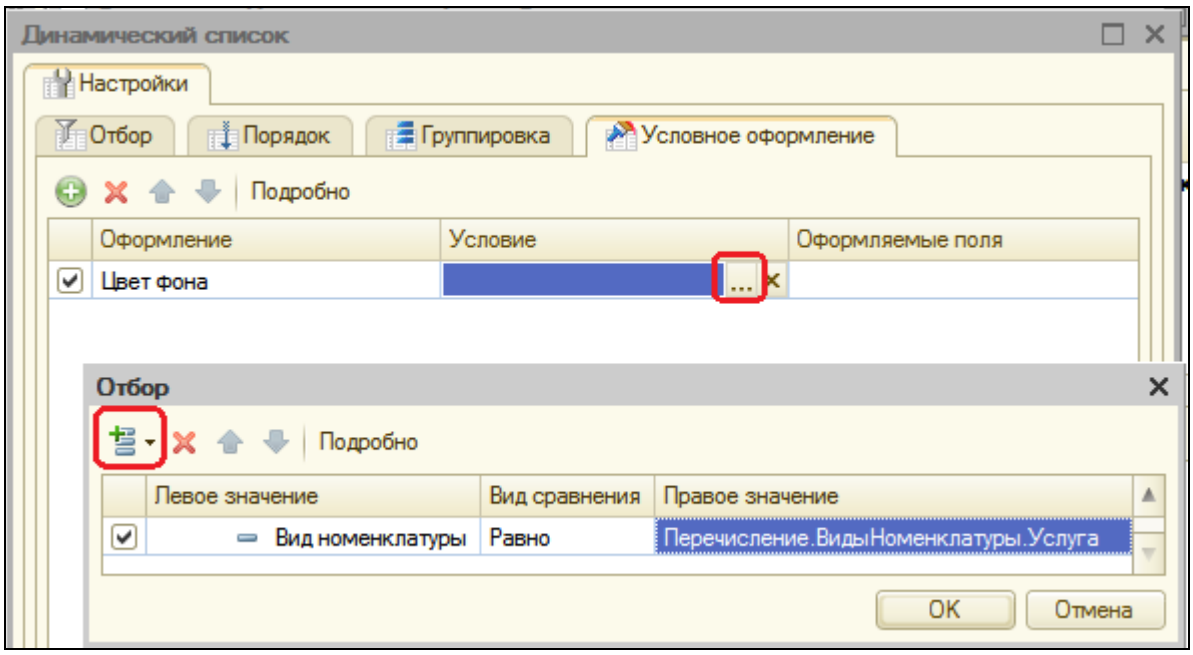


Затем укажем условие, при наступлении которого будет применяться оформление, то есть когда строки списка будут сиреневыми.

Нажмем кнопку выбора в поле **Условие** и в появившемся окне добавим **Новый элемент** отбора. Для этого нажмем кнопку **Добавить** и укажем в графе **Левое значение** – поле **ВидНоменклатуры**, в графе **Вид сравнения** – **Равно**, а в графе **Правое значение** выберем **Перечисление.ВидыНоменклатуры.Услуга**.

Таким образом, мы установили, что когда в списке номенклатуры будут отражаться услуги, эти строки будут выделены сиреневым фоном.

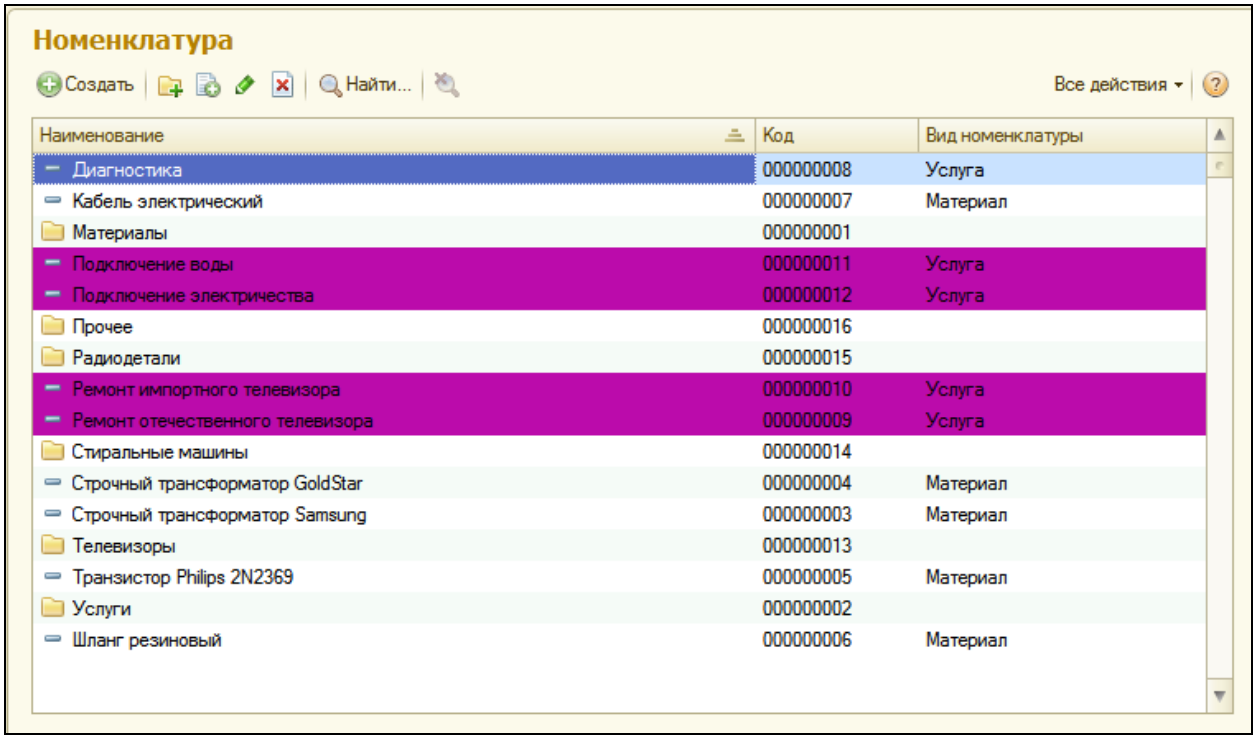
Поскольку мы хотим выделить полностью строки, а не отдельные поля списка, то список оформляемых полей можно оставить пустым. Нажмем ОК.



В режиме 1С:Предприятие

Запустим режим отладки и откроем список номенклатуры.

Мы видимо, что список имеет вид обычного неиерархического списка, услуги выделены сиреневым цветом, а также отсутствуют разделительные линии строк и колонок списка.



Пользователь может изменить заданное оформление списка, выполнив команду Все действия – Настроить список… Откроется окно настройки

динамического списка, аналогичное окну в конфигураторе, где он может задать свое условное оформление списка и другие настройки.

Аналогично можно настраивать условное оформление и для списков, источником которых является не динамический список, а другие типы данных. Например, условное оформление табличной части документа.

Но выполняется это уже с помощью условного оформления самой формы. Т.е. в дереве элементов формы нужно выделить корневой элемент и в палитре свойств открыть ссылку **УсловноеОформление**.

Создавая элементы условного оформления формы, нужно помнить, что в отличие от оформления динамических списков, в форме обязательно нужно указать те поля, которые должны быть оформлены.

Вычисляемые колонки в списках

Необходимость вывода произвольных данных в колонках списка возникает, когда вместе с элементом списка нужно отобразить некоторую вычисляемую информацию.

Мы рассмотрим эту ситуацию на примере отображения актуальной цены в списке справочника **Номенклатура**.

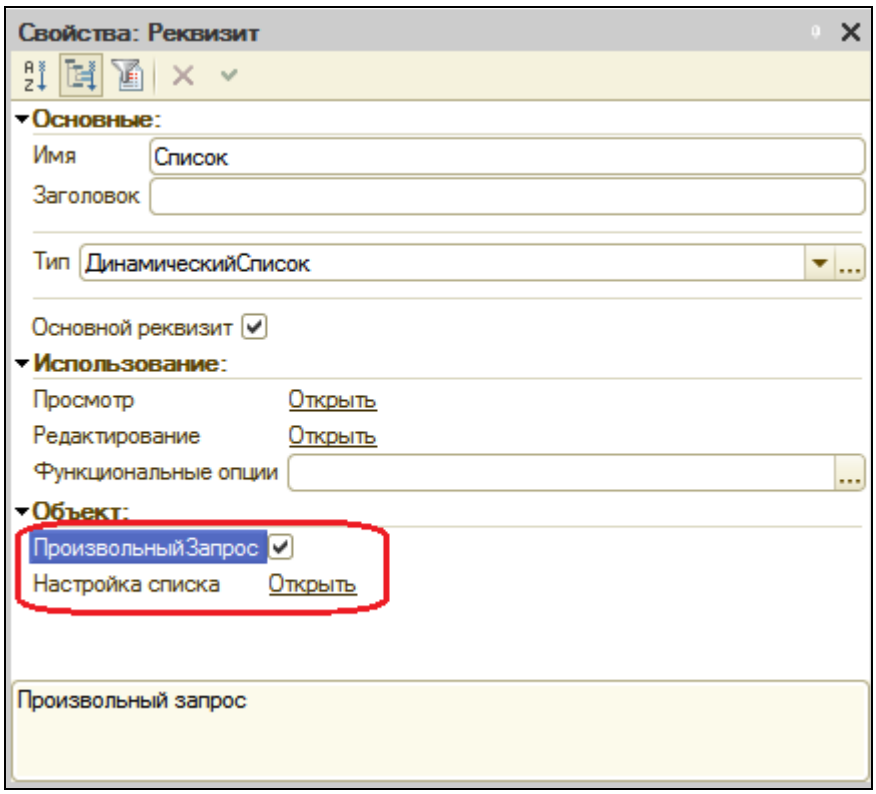
Эти данные мы можем получить из таблицы регистра сведений **Цены.СрезПоследних**. Следовательно, поле Цена из этой таблицы нам нужно добавить в динамический список **Список**, который является основным реквизитом формы списка номенклатуры и служит источником данных для таблицы списка.

В режиме Конфигуратор

Откроем в конфигураторе форму списка справочника **Номенклатура** и вызовем свойства основного реквизита формы **Список**.

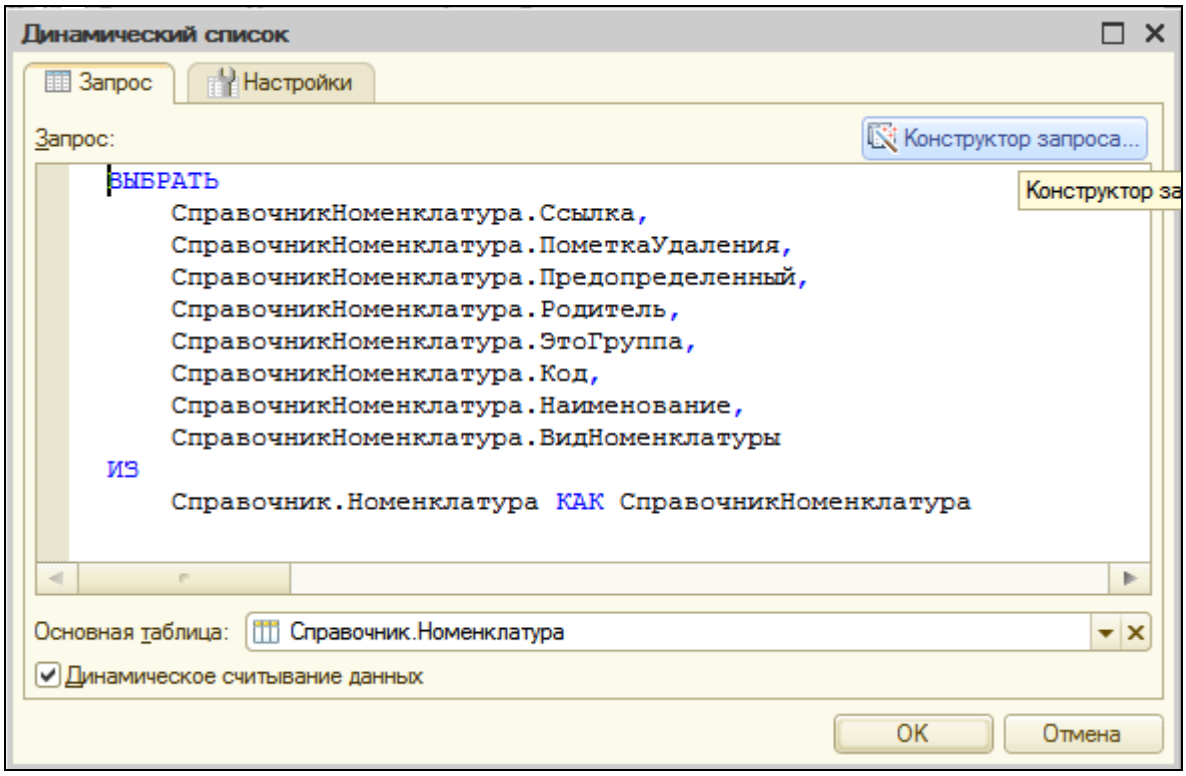
До сих пор в свойствах динамического списка была указана **Основная таблица** – **Справочник.Номенклатура**, и список формировался путем запроса к этой таблице.

Нам нужна еще связанная информация из таблицы регистра сведений **Цены.СрезПоследних**. Поэтому установим флажок **ПроизвольныйЗапрос** и в строке **Настройка списка** нажмем **Открыть**.

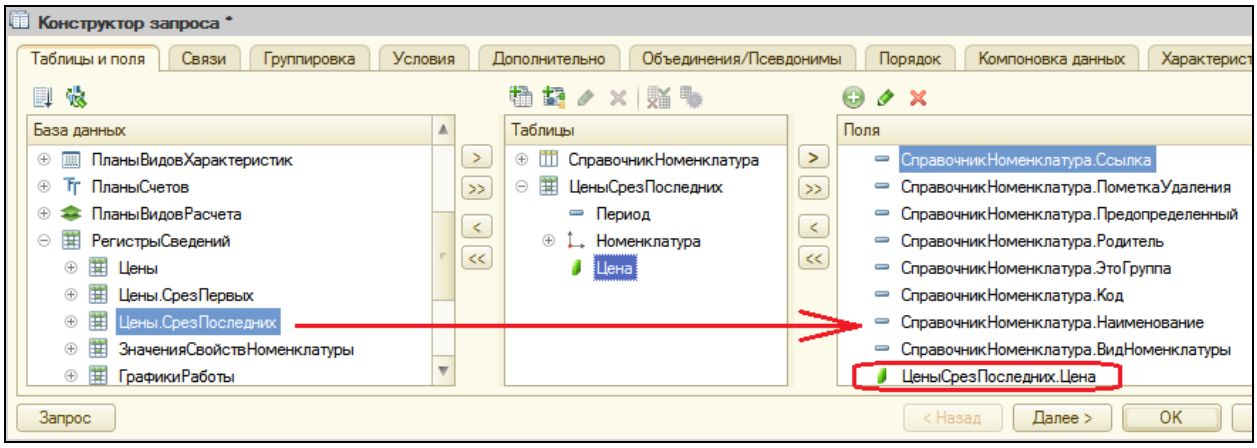


Откроется окно настройки динамического списка. На закладке **Запрос** мы видим запрос, в котором выбираются все поля из таблицы **Справочник.Номенклатура**.

Изменим его. Для этого нажмем **Конструктор запроса**.

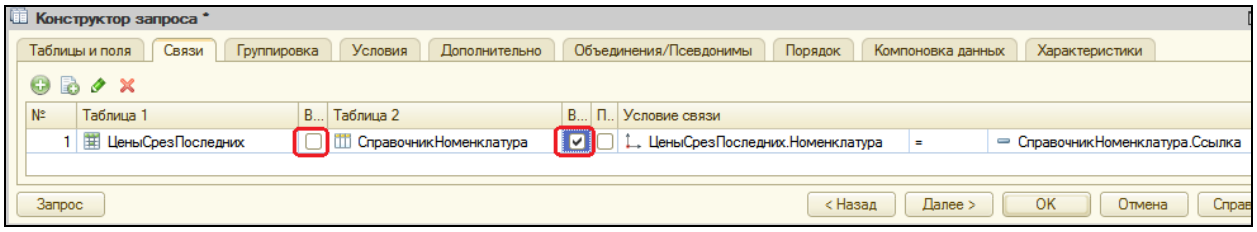


Добавим в список таблиц **Цены.СрезПоследних** и выберем из нее поле **Цена**.



На закладке **Связи** отредактируем связь между таблицами, созданную по умолчанию.

Поставим флажок **Все** у таблицы **Справочник.Номенклатура** и снимем его у таблицы **Цены.СрезПоследних**.



Тем самым мы задаем, что в списке номенклатуры будут отражены все позиции, даже те, по которым не установлены цены.

Создание запроса окончено, нажмем ОК. Текст запроса нам уже знаком и понятен, поэтому не будет на нем останавливаться.

Теперь колонка **Цена**, содержащая актуальную цену, будет отображаться в списке номенклатуры, когда мы поместим ее в форме списка.

В окне настройки динамического списка перейдем на закладку **Настройки** и зададим оформление этой колонки так, чтобы низкие цены выделялись цветом.

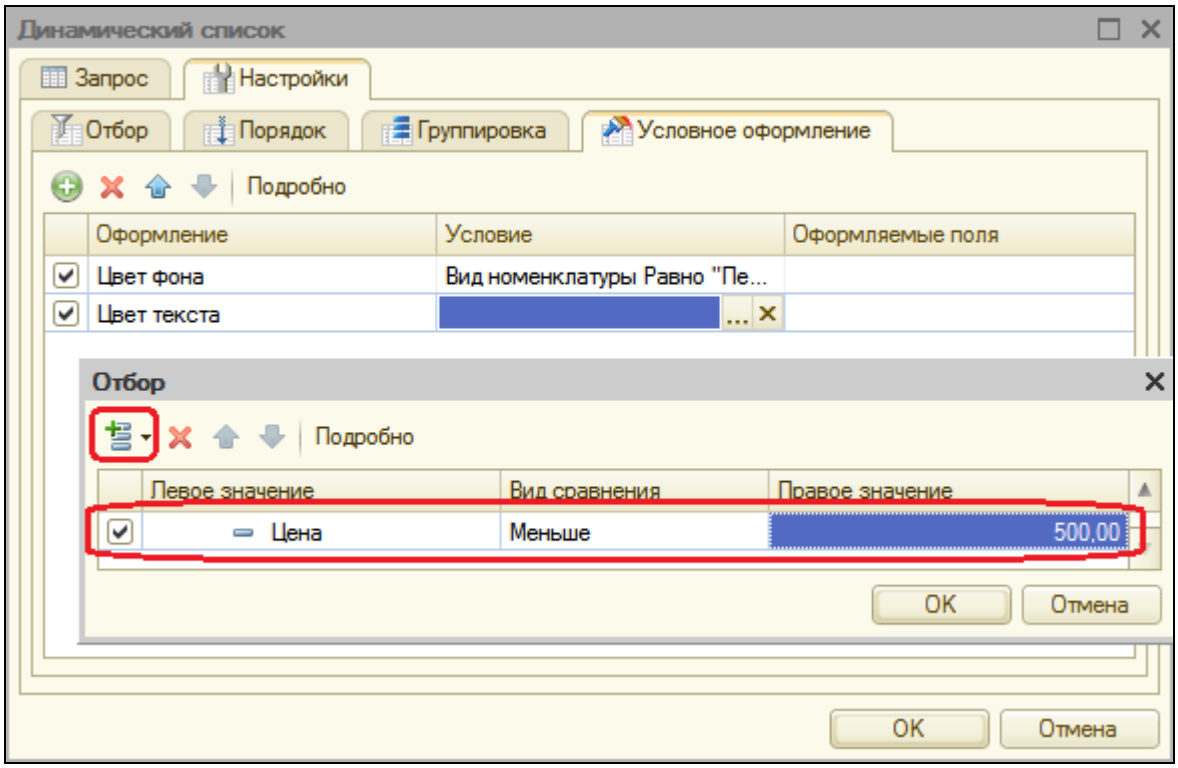
Для этого перейдем на закладку Условное оформление. Там мы видим созданное нами ранее условное оформление для строк списка. Нажмем кнопку **Добавить** в командной панели окна.

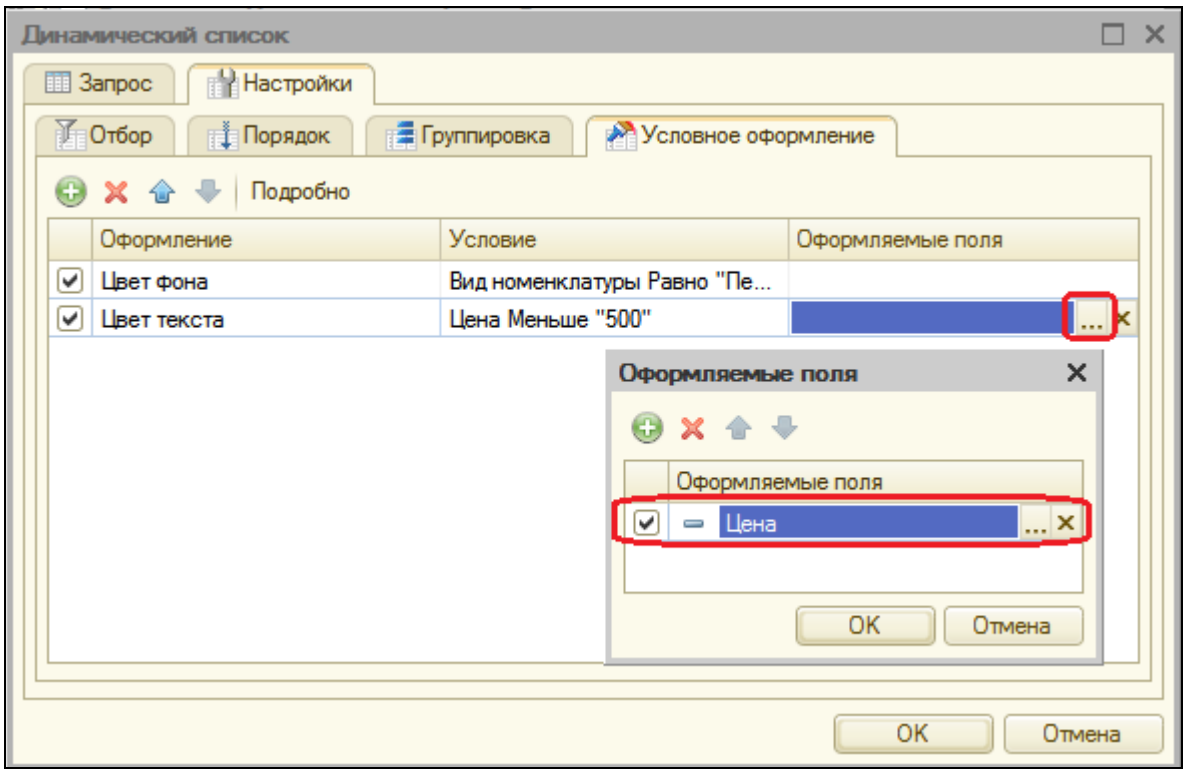
Сначала укажем **Оформление** для выделения полей. Нажмем кнопку выбора в поле **Оформление** и установим синий цвет текста.

Затем укажем условие, при наступлении которого будет применяться оформление, т.е. когда текст в колонке **Цена** будет синим.

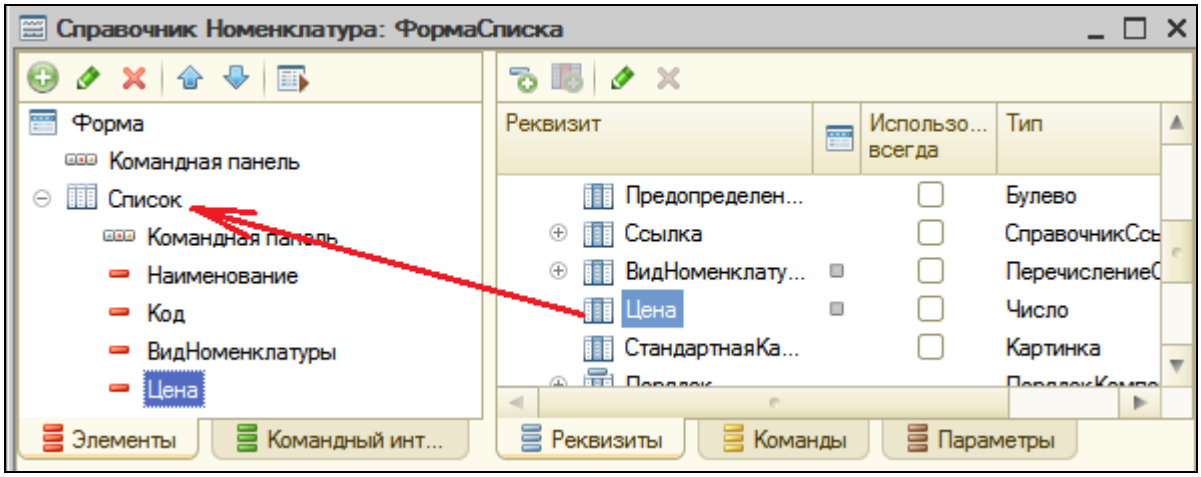
Нажмем кнопку выбора в поле **Условие** и добавим **Новый элемент** отбора. Для этого нажмем кнопку **Добавить** и укажем в графе **Левое значение** – поле **Цена**, **Вид сравнения** – **Меньше**, **Правое значение** – **500**.

Затем укажем список оформляемых полей. Нажмем кнопку выбора в поле **Оформляемые поля**, затем **Добавить** и выберем поле **Цена**.



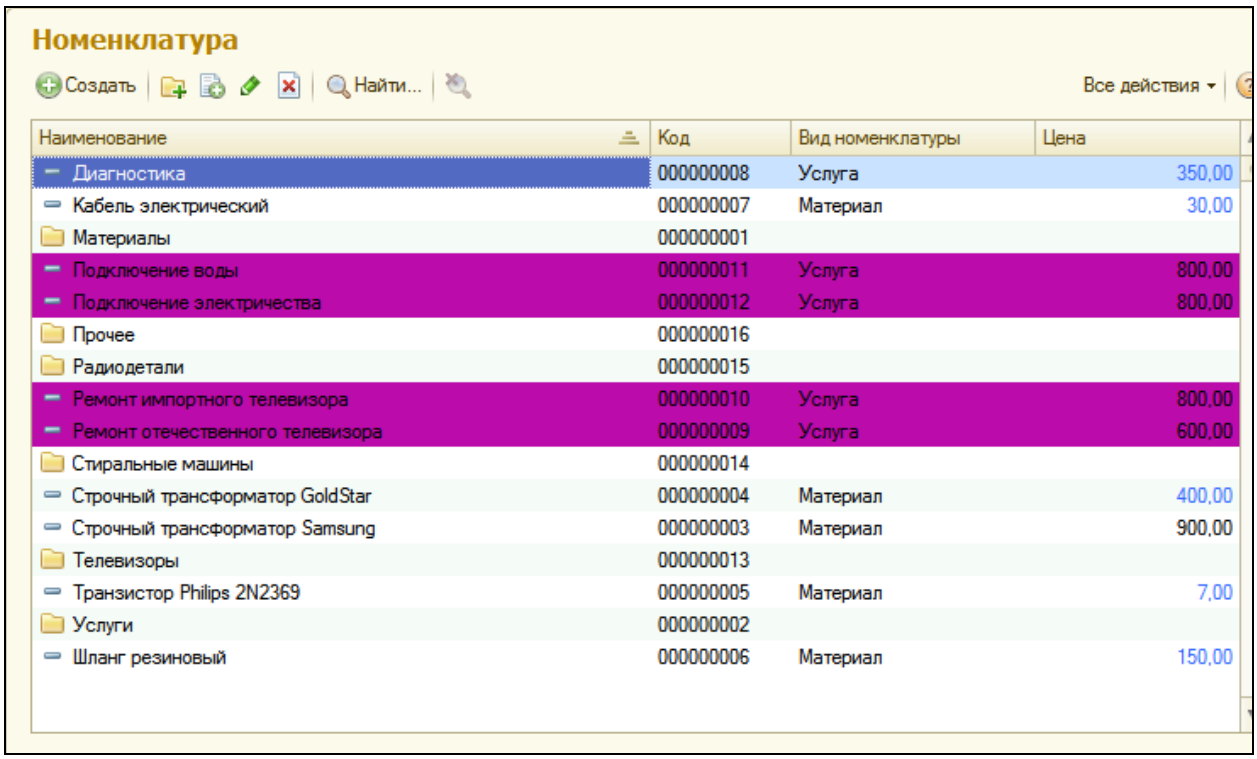


Нажмем ОК. Теперь перетащим поле **Цена** из окна реквизитов в окно реквизитов формы.



В режиме 1С:Предприятие

Запустим режим отладки и откроем список номенклатуры. Мы видим, что вместе с номенклатурой выводится ее актуальная цена, причем цены меньше 500 выделены синим цветом.



Список выбора для поля ввода

Одним из полезных и удобных механизмов поля ввода является использование списка выбора для этих полей.

Поля ввода, имеющие ссылочный тип, например **Склад**, **Сотрудник**, по умолчанию имеют кнопку выбора. При нажатии этой кнопки открывается форма выбора объекта, на которую ссылается данное поле.

Но для полей других типов тоже бывает нужно выполнять выбор из нескольких предопределенных значений.

Рассмотрим совсем простой пример, когда пользователю нужно внести адреса клиентов, начинающиеся с названия города.

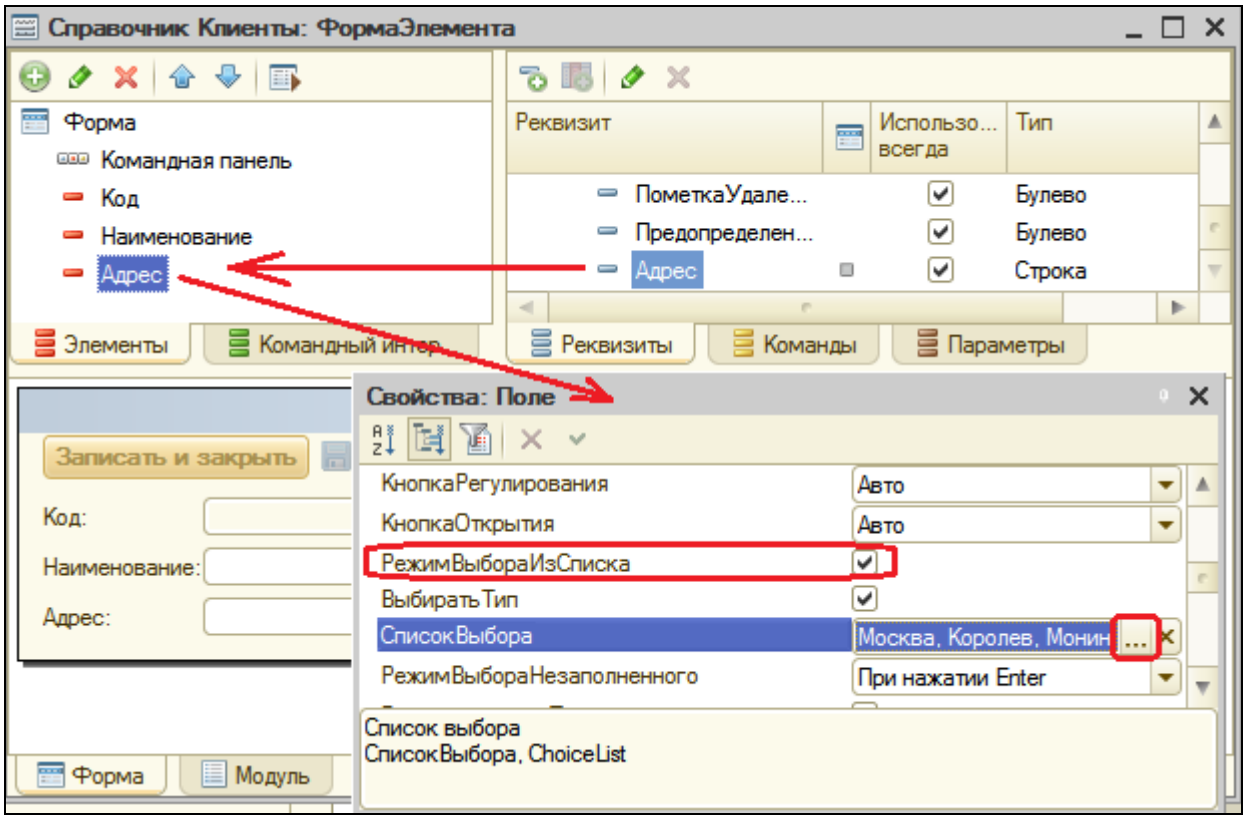
В режиме Конфигуратор

Сначала нужно добавить поле **Адрес** в форму элемента справочника **Клиенты**.

Для этого откроем в конфигураторе окно редактирования справочника **Клиенты** и на закладке данные добавим реквизит **Адрес** с типом **Строка** длиной **25**.

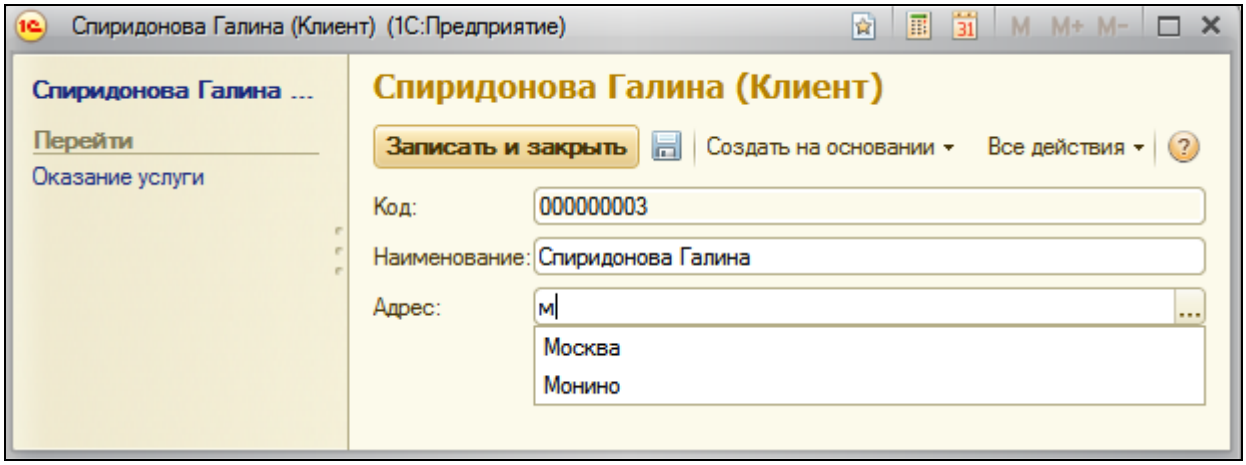
Затем откроем форму элемента справочника **Клиенты**.

Раскроем основной реквизит формы **Объект** и перетащим поле **Адрес** из окна реквизитов в окно элементов формы.



В режиме 1С:Предприятие

Запустим режим отладки и откроем форму редактирования клиента. В поле **Адрес** наберем «м» и система предложит на выбор два подходящих названия.



Форма выбора для поля, содержащего ссылочный реквизит

В процессе работы прикладного решения довольно распространенной является ситуация, когда данные вводятся в поля ссылочных реквизитов, т.е. реквизитов, ссылающихся на какие-либо объекты конфигурации.

Например, в документе **Оказание услуги** мы заполняем поля ссылочного типа **Клиент** (тип **СправочникСсылка.Клиенты**), **Мастер** (тип **СправочникСсылка.Сотрудники**) и др.

При нажатии кнопки выбора в этих полях по умолчанию открывается основная форма выбора соответствующих объектов (справочника **Клиенты**, **Сотрудники** и др.).

Но иногда бывает нужно открывать свою специальную форму для выбора ссылочного реквизита. Для этого в свойстве **Форма** выбора этого реквизита нужно указать эту специальную форму, и тогда она будет открываться для выбора данного реквизита в любой форме, где он находится.

В режиме Конфигуратор

Для примера рассмотрим документ **ОказаниеУслуги**. Допустим, что в поле **Мастер** нам нужно открывать не стандартную форму выбора справочника **Сотрудники**, а специально разработанную для этого форму.

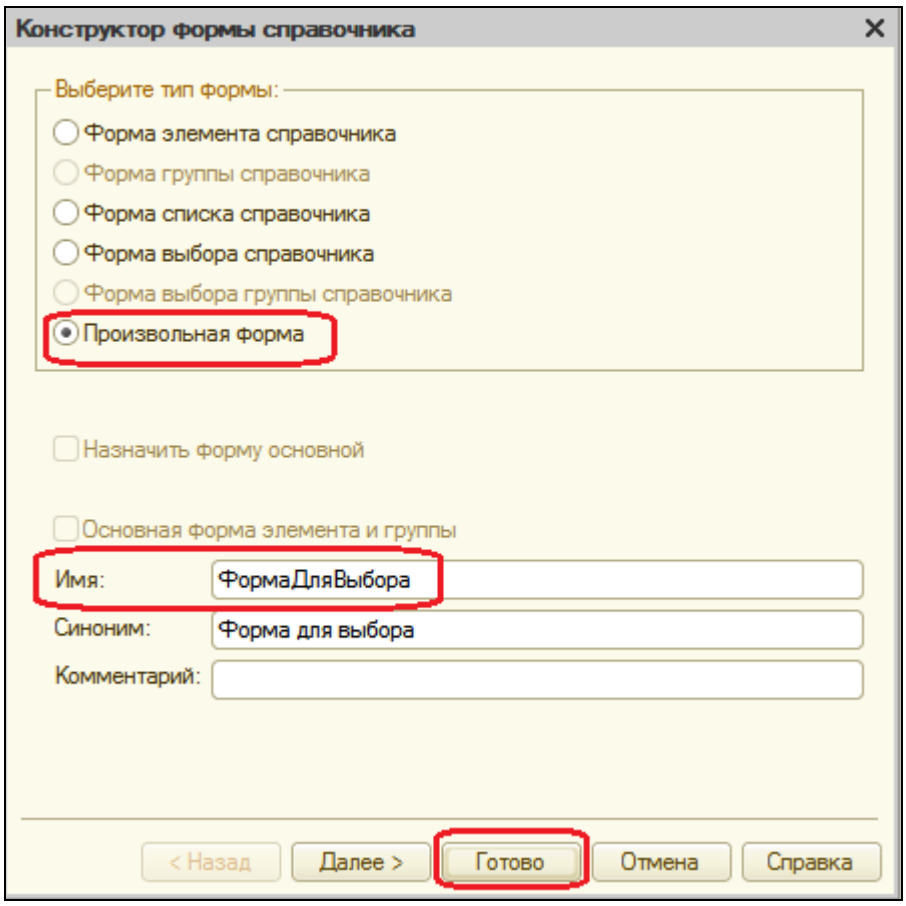
Сначала нужно эту форму создать.

Откроем окно редактирования справочника **Сотрудники** и перейдем на закладку **Формы**.

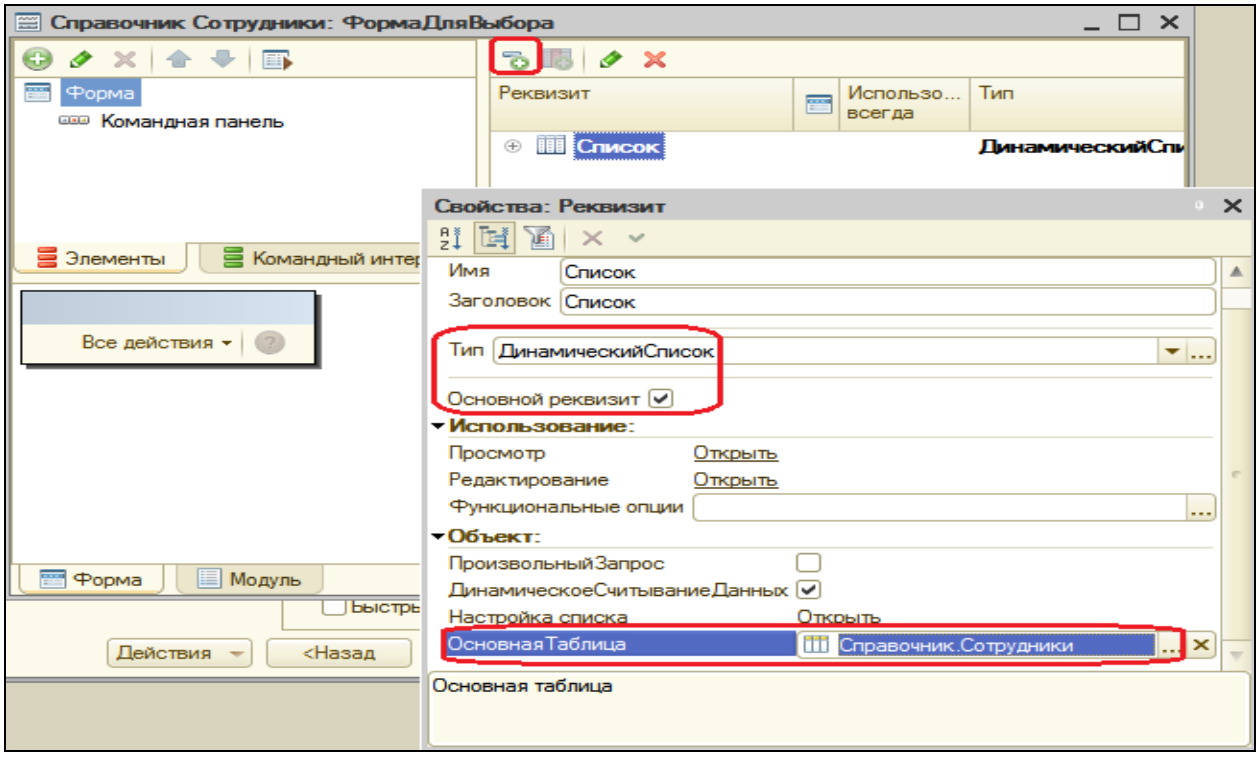
Мы видим, что у этого объекта нет ни одной формы. Это значит, что все формы справочника в режиме 1С:Предприятие, в том числе и форма выбора, генерировалась системой автоматически.

Нажмем кнопку **Добавить**. В открывшемся конструкторе форм выберем тип формы – **Произвольная форма** и зададим ее имя **ФормаДляВыбора**. Нажмем Готово.

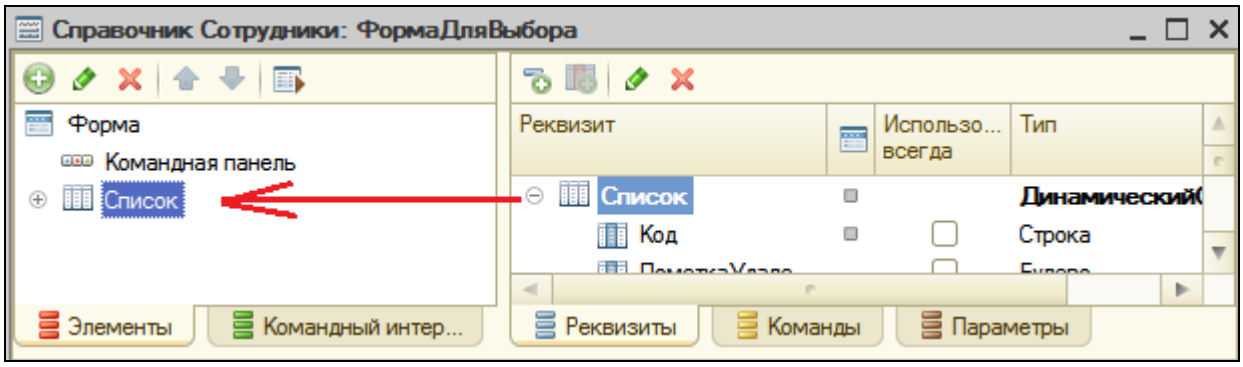
В открывшемся редакторе форм мы видим, что в форме нет ни одного элемента, и у формы нет данных, т.к. форма – произвольная, и мы можем наполнять ее данными и элементами по своему усмотрению.



На закладке **Реквизиты** создадим основной реквизит формы с именем **Список** и типом **ДинамическийСписок**. В качестве основной таблицы выберем **Справочник.Сотрудники**.

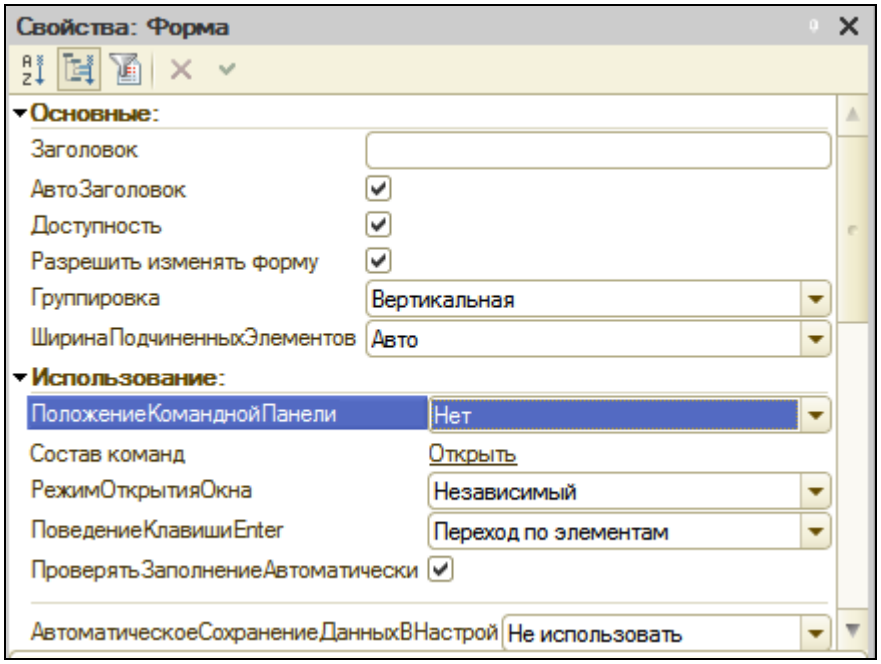


Затем перетащим этот реквизит в окно элементов формы. Согласимся с предложением «Добавить колонки таблицы **Список**?». В форме появится таблица **Список**, отображающая список сотрудников.

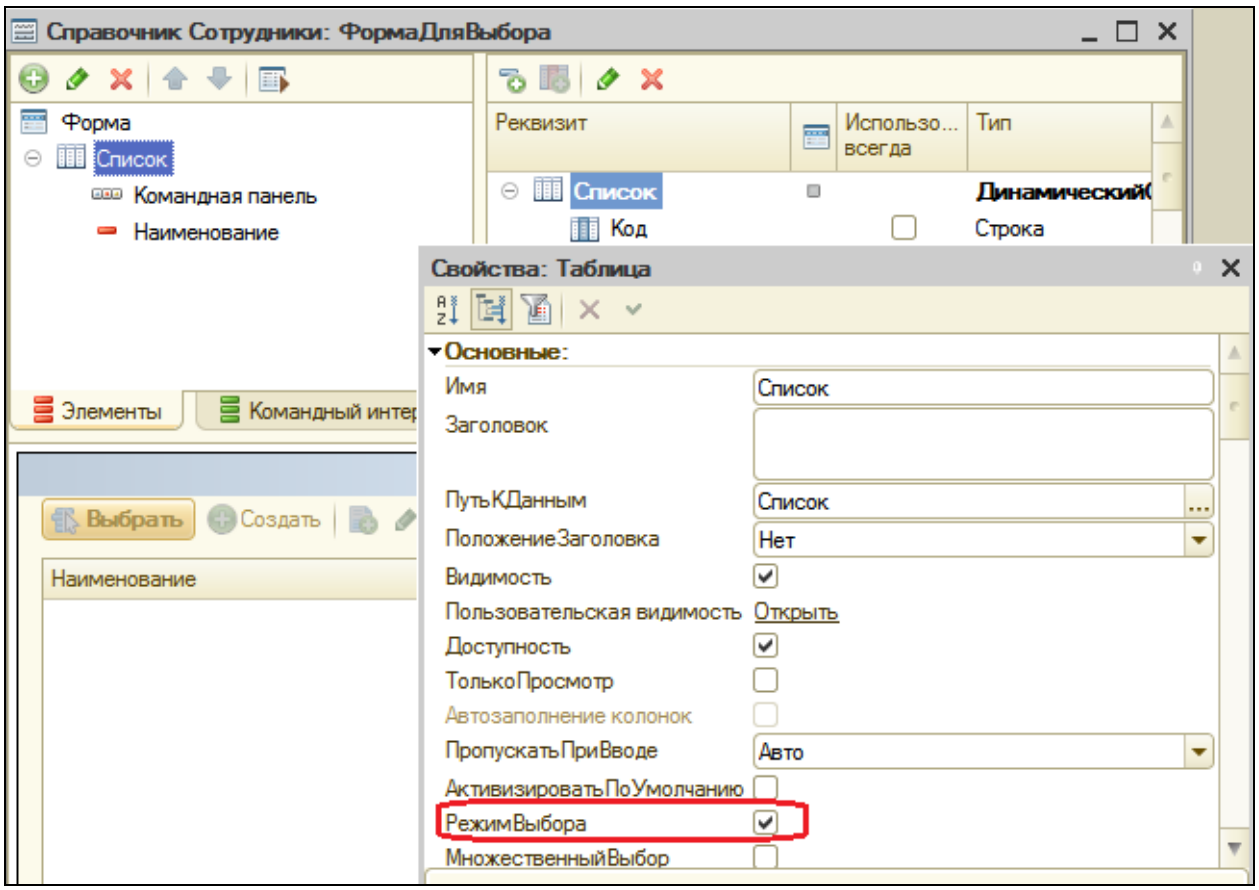


Теперь немного изменим внешний вид формы.

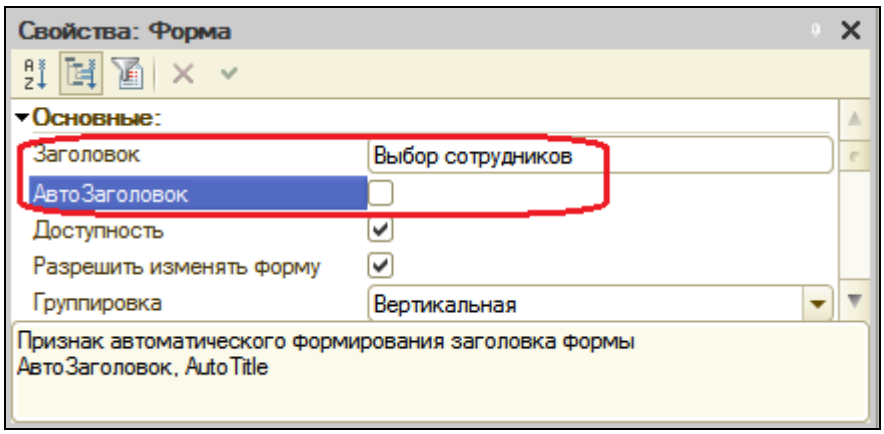
В окне элементов формы раскроем таблицу **Список** и удалим поле **Код**. Затем удалим командную панель формы, т.к. таблица по умолчанию имеет свою командную панель. Для этого выделим корневой элемент Форма, откроем его палитру свойств и установим свойство **ПоложениеКомандногоПанели** в значение **Нет**.



Чтобы из таблицы, содержащей список сотрудников, можно было сделать выбор, откроем палитру свойств таблицы **Список** и установим флажок **РежимВыбора**.



И в заключение зададим заголовок формы **Выбор сотрудников**, а флажок **АвтоЗаголовок** снимем, чтобы не отражался заголовок, заданный нами в расширенном представлении списка для справочника **Сотрудники**.



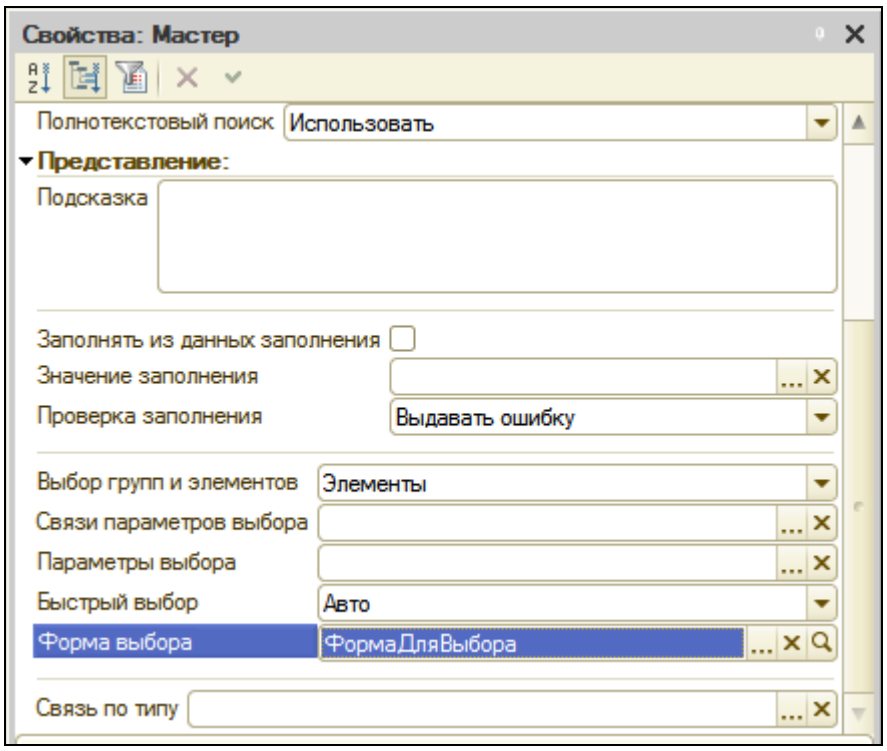
В результате мы создали форму, очень похожую на форму выбора. Мы убрали из списка поле **Код** и изменили заголовок формы, чтобы наша форма внешне в чем-то отличалась от формы выбора.

В действительности произвольная форма, конечно, будет отличаться своей функциональностью, иначе зачем ее назначать в качестве формы для выбора? Но это сейчас не входит в нашу задачу. Мы хотим только

показать принцип использования произвольной формы в качестве формы выбора.

Теперь откроем окно редактирования документа **ОказаниеУслуги**.

Вызовем окно свойств реквизита **Мастер** и нажмем кнопку выбора в поле **Форма выбора**. Откроется список форм, созданных в конфигурации для объекта, на который ссылается данный реквизит. В данном случае для справочника **Сотрудники**, на который ссылается реквизит **Мастер**, создана одна форма – **ФормаДляВыбора**. Выберем ее.

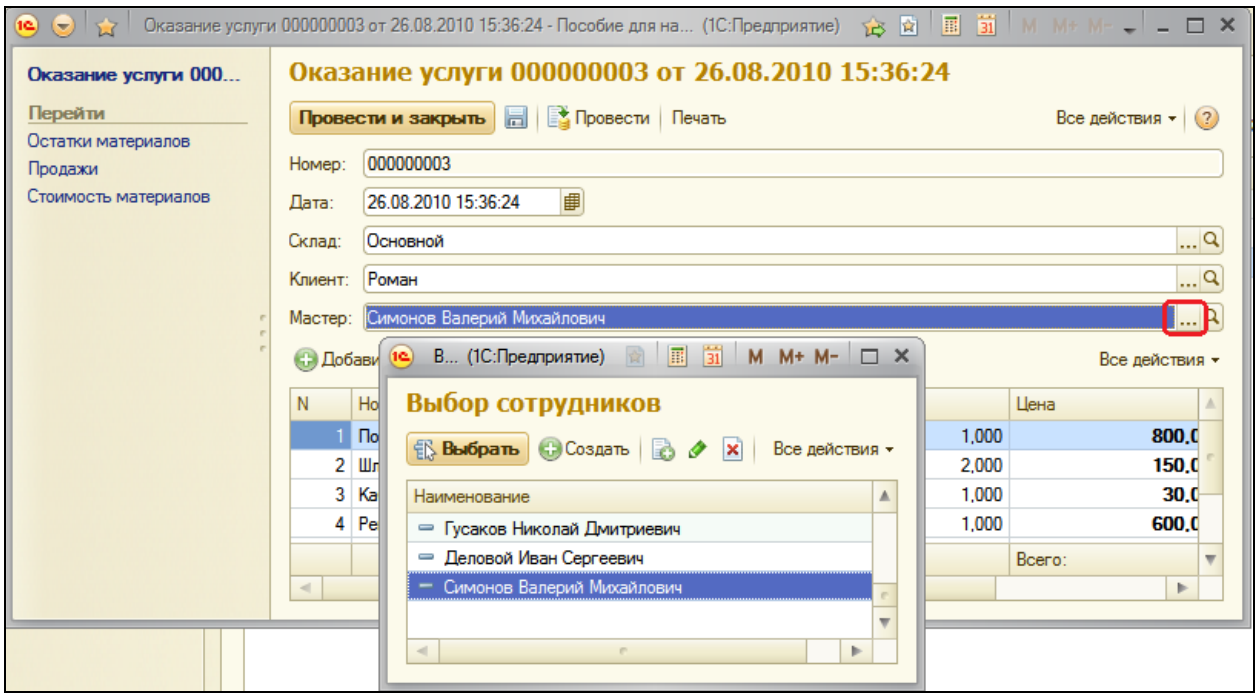


В режиме 1С:Предприятие

Запустим режим отладки и откроем один из документов **Оказание услуги**.

Нажмем кнопку выбора в поле **Мастер**. Откроется созданная нами произвольная форма с заголовком **Выбор сотрудников**.

Поскольку мы задали свойство **Форма** выбора на уровне реквизита **Мастер**, а не на уровне отдельной формы, в любой форме документа **ОказаниеУслуги**, где испльзуется данный реквизит, при выборе в поле **Мастер** будет открываться наша специальная форма.



Если очистить свойство **Форма** выбора для реквизита **Мастер** в документе **ОказаниеУслуги**, то в режиме 1С:Предприятие при выборе в поле **Мастер** будет открываться основная форма выбора справочника **Сотрудники**, автоматически сгенерированная системой.

Проверка заполнения реквизитов

Для реквизитов объектов конфигурации существует возможность как автоматической, так и программной проверки их заполнения. Причем делается это не на уровне форм, а на уровне свойств реквизитов, или в модуле объекта, к которому относится данный реквизит. Т.о., проверка заполнения реквизита будет производиться во всех формах, где используется этот реквизит.

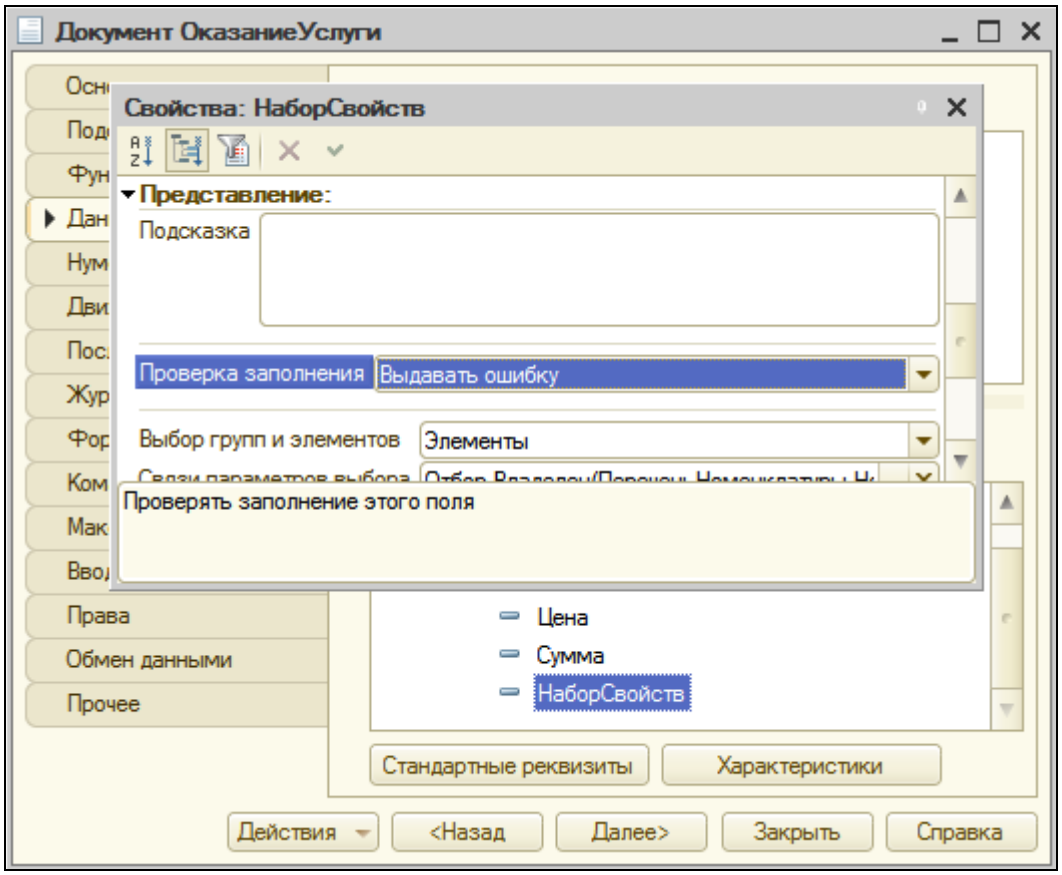
Автоматическая проверка заполнения

В режиме Конфигуратор

Для примера рассмотрим документ **ОказаниеУслуги**. Допустим, нам нужно контролировать заполнение реквизита **НаборСвойств** табличной части этого документа.

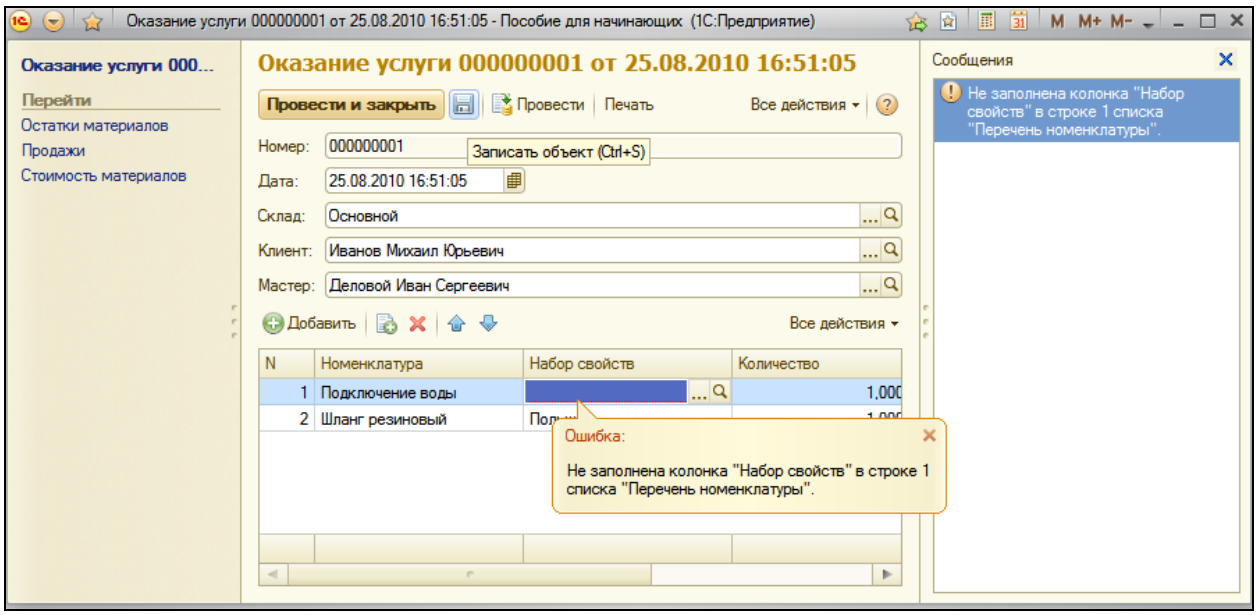
Откроем окно редактирования документа и вызовем свойства реквизита табличной части **НаборСвойств**, установим свойство **Проверка заполнения** в значение **Выдавать ошибку**.

Тем самым при записи документа этот реквизит будет проверяться на заполнение. Иначе будет выдано сообщение об ошибке, и документ не будет сохранен.



В режиме 1С:Предприятие

Запустим режим отладки и откроем документ **Оказание услуги №1**. В табличной части этого документа содержится строка с услугой **Подключение воды**, для которой колонка **Набор свойств** не заполнена. Если нажать кнопку записи, то появится сообщение об ошибке.



Программная проверка заполнения

В режиме Конфигуратор

Иногда бывает нужно самостоятельно производить проверку заполнения реквизита, в соответствии с программной логикой. В этом случае, если для реквизита установлено свойство **Проверка заполнения** в значение **Выдавать ошибку**, нужно удалить этот реквизит из массива проверяемых реквизитов и выполнить проверку программным путем. Или же, если для реквизита не установлена проверка заполнения, можно программно добавить реквизит в массив проверяемых.

Программная проверка заполнения объектов конфигурации выполняется в обработчике **ОбработкаПроверкиЗаполнения()**, который нужно поместить в модуле объекта. Этот обработчик вызывается автоматически при сохранении любой формы. Программную проверку объектов интерактивного ввода нужно делать именно в этом обработчике, а не при записи объекта.

Откроем модуль документа **ОказаниеУслуги** и поместим в нем следующую процедуру.

```
Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)

	ПроверяемыеРеквизиты.Удалить(ПроверяемыеРеквизиты.Найти("ПереченьНоменклатуры.НаборСвойств"));
	Индекс = 0;

	Для Каждого ТекСтрокаПереченьНоменклатуры Из ПереченьНоменклатуры Цикл
		Индекс = Индекс + 1;
```

```
            Если ТекСтрокаПереченьНоменклатуры.Номенклатура.ВидНоменклатуры =
Перечисления.ВидыНоменклатуры.Материал Тогда

                    Если Не
ЗначениеЗаполнено(ТекСтрокаПереченьНоменклатуры.НаборСвойств) Тогда
                        Сообщение = Новый СообщениеПользователю();
                        Сообщение.Текст = "В строке " + Индекс + " списка
Перечень номенклатуры не заполнена колонка Набор свойств";
                        Сообщение.Поле =  "ПереченьНоменклатуры[" +
Строка(Индекс - 1) + "].НаборСвойств";
                        Сообщение.УстановитьДанные(ЭтотОбъект);
                        Сообщение.Сообщить();
                        Отказ = Истина;
                    КонецЕсли;

            КонецЕсли;

        КонецЦикла;

КонецПроцедуры
```

Сначала мы удаляем реквизит табличной части **НаборСвойств** из массива **ПроверяемыеРеквизиты**. Этот массив передается в обработчик и содержит массив проверяемых реквизитов, которым мы установили свойство **Проверка заполнения** в значение **Выдавать ошибку**.

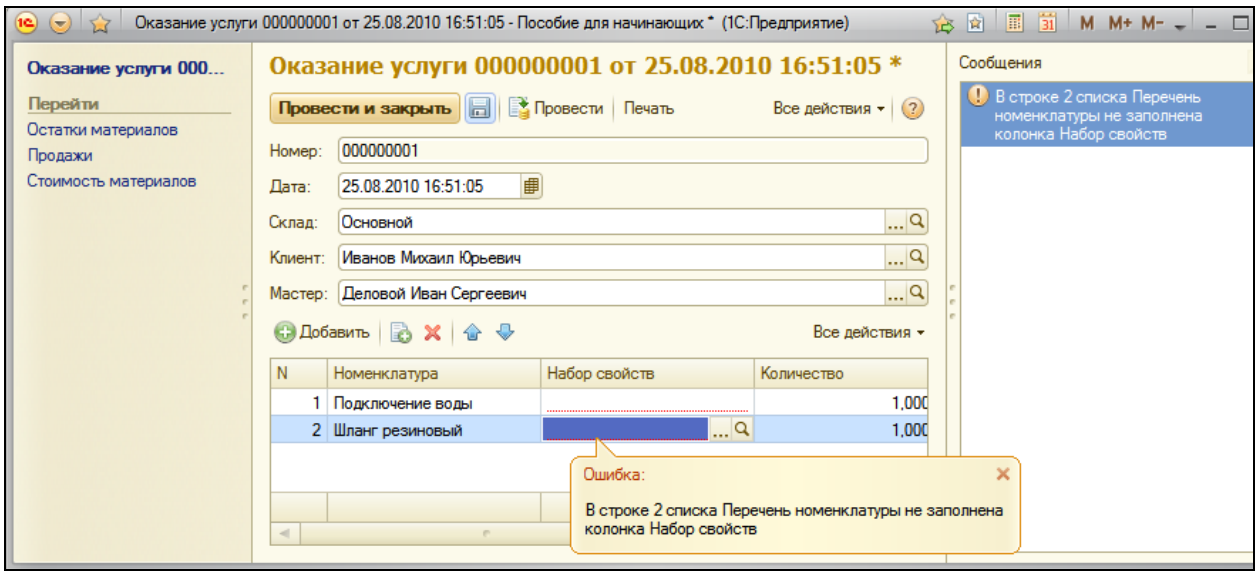
Затем мы в цикле обходим строки табличной части документа и формируем сообщение об ошибке только в том случае, если номенклатура в табличной части является материалом и для нее не заполнена колонка **НаборСвойств**.

Параметр **Отказ** мы устанавливаем в значение **Истина**. Это значит, что документ не будет сохранен, если найден хотя бы один незаполненный реквизит **НаборСвойств** для номенклатуры, являющейся материалом. Если этот параметр закомментировать, сообщения об ошибке будут выдаваться, но документ будет сохранен.

Для упрощения примера мы опять здесь используем обращение **ПереченьНоменклатуры.Номенклатура.ВидНоменклатуры**, хотя более оптимально использовать запрос.

В режиме 1С:Предприятие

Запустим режим отладки и откроем документ **Оказание услуги №1**. В табличной части этого документа удалим **Набор свойств** для резинового шланга и попробуем записать документ. Для второй строки табличной части будет выдано сообщение об ошибке.



Можно также программно добавить реквизит в массив проверяемых реквизитов.

```
ПроверяемыеРеквизиты.Добавить("ПеречеьНоменклатуры.НаборСвойств");
```

Использование параметризированных команд

Использование параметризированных команд в формах объектов позволяет при выполнеии команды передать в обработчик команды какой-либо параметр, например значение ссылочного реквизита. И затем использовать его, например открыть с этим параметром форму отчета.

Для примера создадим команду открытия отчета, показывающего остатки материалов по складу, указанному в документе. Отчет будет вызываться из формы документа **ОказаниеУслуги**, в него будет передаваться значение реквизита документа **Склад**, и при открытии отчета в настройки отчета будет добавляться отбор по параметру **Склад**.

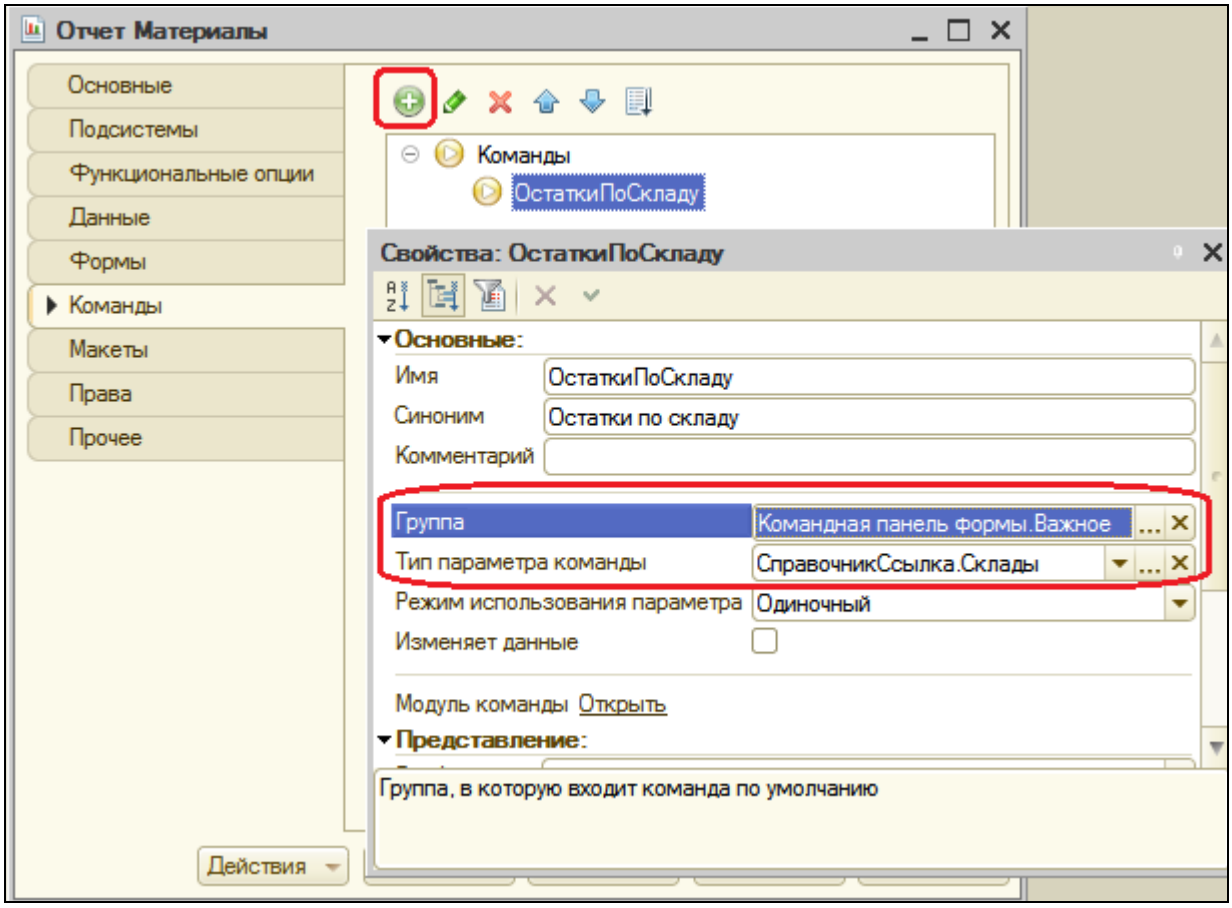
В режиме Конфигуратор

Сначала создадим параметризированную команду отчета **Материалы**.

Для этого откроем окно редактирования этого объекта. На закладке **Команды** нажмем кнопку **Добавить** и создадим команду **ОстаткиПоСкладу**.

В открывшемся окне свойств зададим **Тип параметра** команды – **СправочникСсылка.Склады**. В свойстве **Группа** укажем **Командная панель формы.Важное**.

Мы создали команду с типом параметра **СправочникСсылка.Склады**, и теперь во всех формах, имеющих реквизит такого типа, эта команда будет автоматически добавлена в список параметризированных команд, доступных в форме.



В открывшемся модуле команды заполним обработчик для ее выполнения следующим образом.

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

ПараметрыФормы = Новый
Структура("Отбор,КлючНазначенияИспользования,СформироватьПриОткрытии
", Новый Структура("Склад", ПараметрКоманды), "ОстаткиПоСкладу", Истина);
ОткрытьФорму("Отчет.Материалы.Форма", ПараметрыФормы,
ПараметрыВыполненияКоманды.Источник,
ПараметрыВыполненияКоманды.Уникальность,
ПараметрыВыполненияКоманды.Окно);

КонецПроцедуры
```

В процедуру **ОбработкаКоманды()** передается **ПараметрКоманды**, содержащий значение типа **СправочникСсылка.Склады**. Затем создается структура параметров формы (**ПараметрыФормы**): **Отбор**, **КлючНазначенияИспользования**, **СформироватьПриОткрытии**.

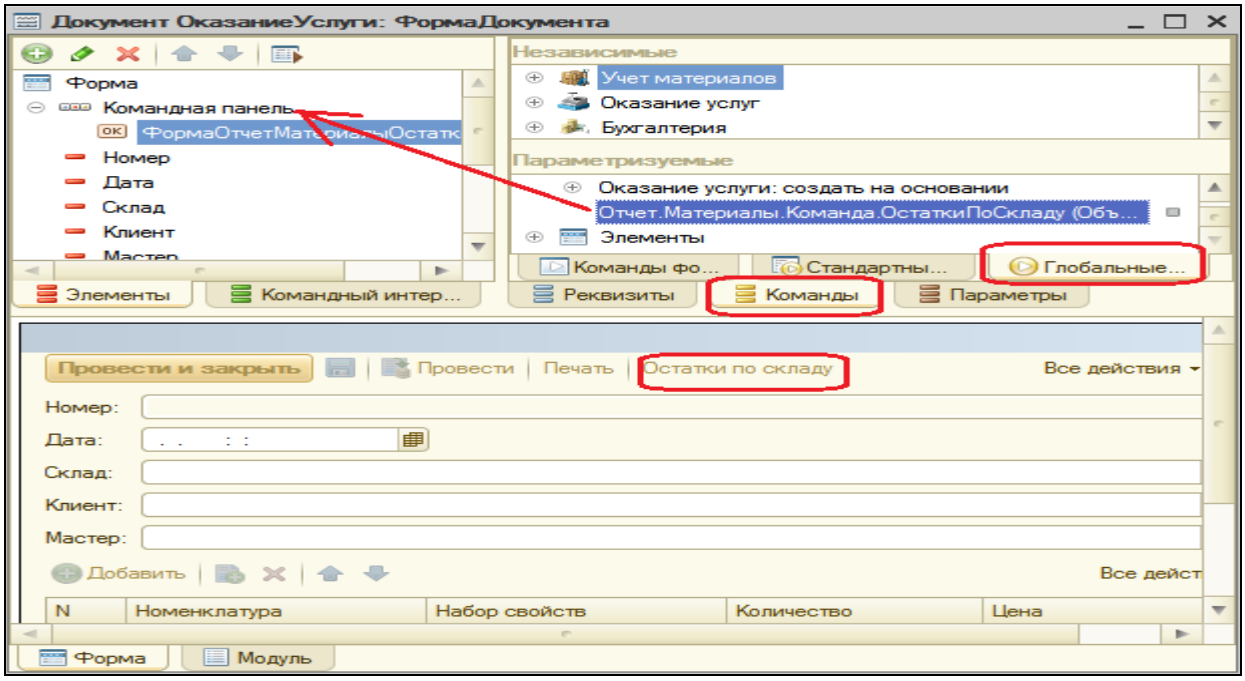
В параметр **Отбор** добавляется структура, содержащая элемент **Склад** со значением, содержащимся в параметре команды, параметр **КлючНазначенияИспользования** – **«Остатки по складу»** определяет назначение использования формы, а параметру **СформироватьПриОткрытии** присваивается значение **Истина**, чтобы отчет формировался сразу после открытия.

Затем структура параметров формы передается в глобальный методы **ОткрытьФорму()**, и форма, указанная в первом параметре метода, открывается с параметром **Склад**.

Теперь поместим нашу команду в форме документа.

Откроем форму документа **ОказаниеУслуги**. На закладке **Команды** перейдем в раздел **Глобальные команды**. Здесь мы видим список глобальных параметризированных команд, доступных в форме. В группе **Параметризуемые** раскроем строку **Объект** и перетащим нашу команду **Отчет.Материалы.Команда.ОстаткиПоСкладу(Объект.Склад)** в командную панель элементов формы.

В скобках у команды указано значение реквизита **Склад**, которое будет передаваться в команду при ее выполнении.

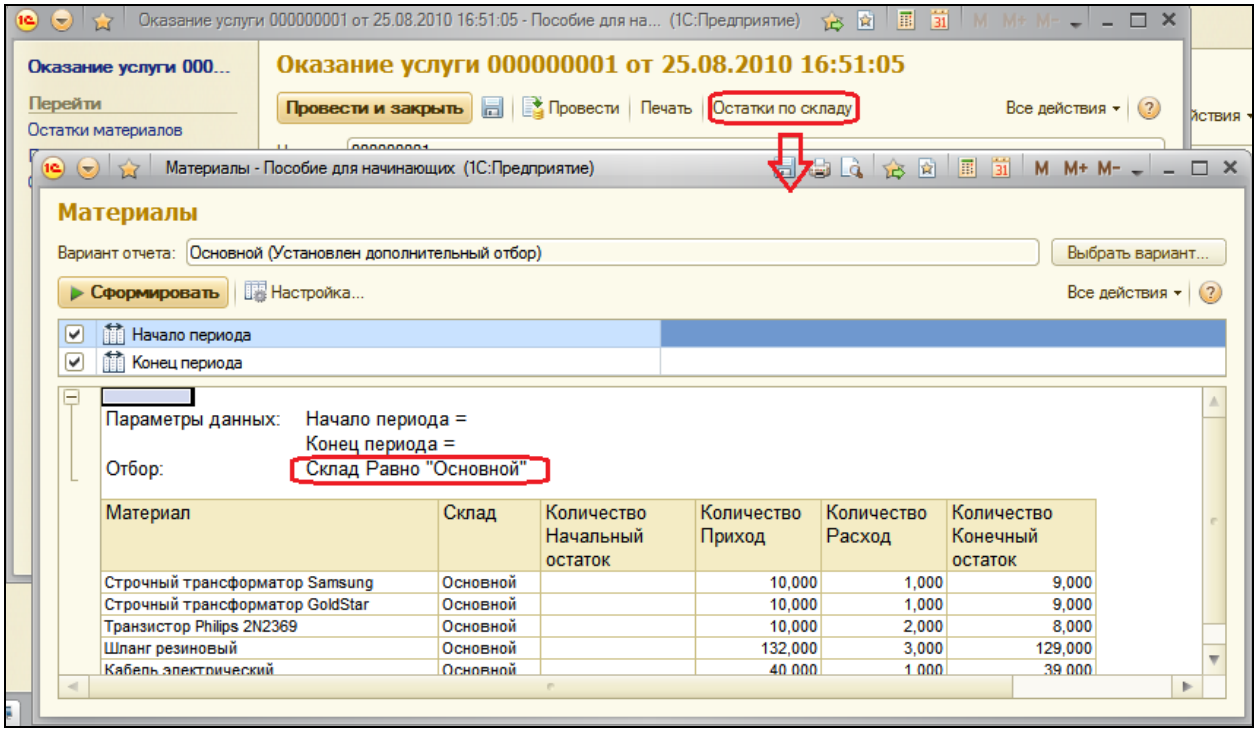


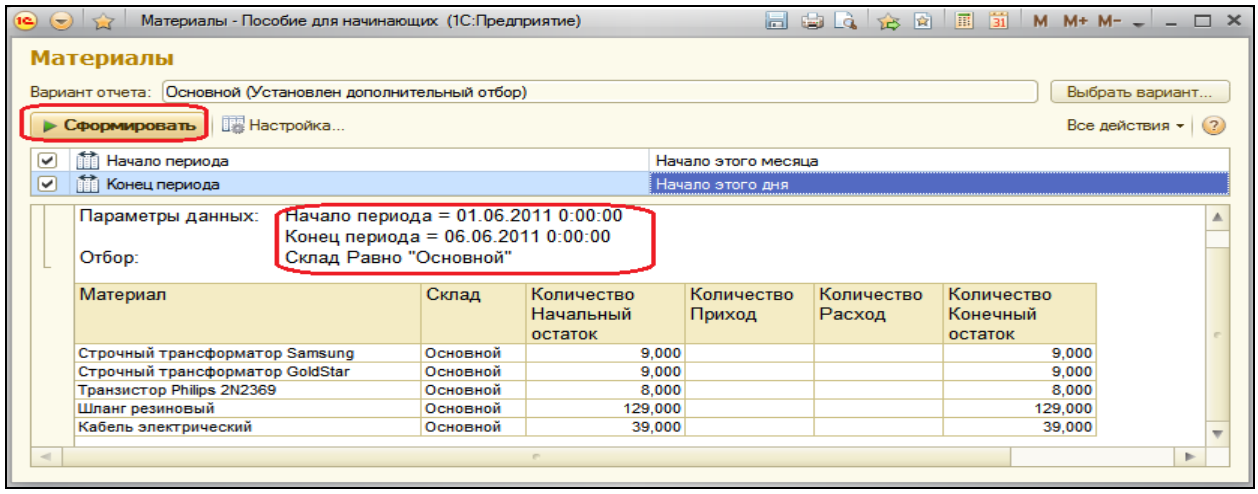
Заметьте, что мы не создавали форму отчета и не добавляли в настройки отчета отбор по параметру формы **Склад**. Система сделает это сама при выполнении обработчика команды **ОстаткиПоСкладу**.

В режиме 1С:Предприятие

Запустим режим отладки и откроем документ **Оказание услуги №1**.

Нажмем кнопку **Остатки по складу** и вызовем отчет **Материалы**. Форма отчета генерируется системой автоматически. Отчет будет выполнен сразу при открытии формы с отбором по складу **Основной**, указанному в форме документа. Дальше можно указать интервал дат и нажать **Сформировать**.





Закроем отчет. Если теперь в форме документа мы изменим склад на **Розничный** и вызовем отчет нажатием кнопки **Остатки по складу** и выполним его, то он будет сформирован с отбором по складу **Розничный**.

Таким образом, мы создали для пользователя очень удобную возможность – открывать отчет, показывающий остатки материалов по складу, прямо из формы документа с отбором по указанному в документе складу.

Контрольные вопросы

- ✓ Как связаны данные и элемент формы.
- ✓ Что такое основной реквизит формы.
- ✓ Какие существуют типы данных у формы.
- ✓ Как выполнить преобразование данных прикладных объектов в данные формы.
- ✓ Как настроить условное оформление строк формы списка.
- ✓ Что такое параметризированная команда и как ее использовать в формах.

Используемая литература:

1. М.Г. Радченко, Е.Ю. Хрусталева «1С:Предприятие 8.2 Практическое пособие разработчика. Примеры и типовые приемы» М.:ООО «1С-Паблишинг», 2009.-874с.