

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по выполнению лабораторных работ
по дисциплине «Автоматизация бухгалтерского учета»
для студентов направления подготовки
09.03.03 «Прикладная информатика»
Направленность (профиль)
«Прикладная информатика в экономике»

Ставрополь 2026

Оглавление

Лабораторная работа №1. Установка и запуск базовой версии конфигурации. Создание ИБ. Разработка подсистем.	3
Лабораторная работа №2. Создание элементов конфигурации: Справочников. Документов.	9
Лабораторная работа №3. Создание элемента конфигурации Регистра накопления.	30
Лабораторная работа № 4. Создание элемента конфигурации Отчеты. Создание Макетов.	36
Лабораторная работа № 5. Создание элемента конфигурации Регистра сведений. Перечисления.	41
Лабораторная работа №6. Проведение документа по нескольким регистрам.	48
Лабораторные работы № 7. Запросы.	54
Лабораторная работа № 8. Планы видов характеристик.	66
Лабораторная работа № 9. Планы видов расчета. Регистры расчета. Пользователи и роли. Рабочий стол.	72
Лабораторная работа № 10. Программирование форм. События формы.	91
Лабораторная работа № 11. Программное выполнение запроса.	105
Лабораторная работа № 12. Экспортируемые процедуры и общие модули.	114
Лабораторная работа № 13. Модуль объекта. События объекта.	117
Лабораторная работа № 14. Создание обработок.	122

Лабораторная работа №1. Установка и запуск базовой версии конфигурации. Создание ИБ. Разработка подсистем.

Цель работы: Научиться создавать новую информационную базу, разрабатывать информационные подсистемы.

Порядок выполнения работы:

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

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

Система 1С:Предприятие имеет различные режимы работы: 1С:Предприятие и Конфигуратор.

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

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

Итак, запустите 1С:Предприятие. В открывшемся диалоге вы увидите список информационных баз, с которыми вы работаете. Если этот список

пуст, система сама предложит вам создать новую базу. Если же в списке информационных баз содержится какая-либо база, например, у вас установлена демонстрационная конфигурация, то для создания новой базы нажмите кнопку *Добавить*. В открывшемся диалоге выберите пункт *Создание* новой информационной базы. Нажмите кнопку *Далее*. На следующем шаге выберите пункт *Создание информационной базы без конфигурации...*

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

Нажмите кнопку *Далее*. На следующем шаге укажите каталог для расположения вашей информационной базы. Нажмите кнопку *Далее*. На следующем шаге нажмите кнопку *Готово*. В диалоге запуска 1С:Предприятия, в списке информационных баз, вы увидите созданную вами новую пустую базу. Запустим 1С:Предприятие в режиме конфигуратора. Перед вами окно конфигуратора. Именно с помощью этого инструмента мы будем создавать нашу конфигурацию.

Большое количество незнакомых пиктограмм на панели инструментов часто смущает начинающего разработчика. Не следует этого бояться – со временем вы сможете свободно ориентироваться среди них. Сначала пользуйтесь пунктами меню, и постепенно ваша работа сама собой переместится на панель инструментов. Выполним первую команду, с которой начинается работа с любой конфигурацией, *Конфигурация – Открыть конфигурацию*. На экране откроется дерево конфигурации. Можно сказать, что дерево конфигурации – основной инструмент, с которым работает разработчик. Дерево конфигурации содержит в себе практически всю информацию о том, из чего состоит конфигурация.

Конфигурация представляет собой *описание*. Она описывает структуру данных, которые пользователь будет использовать в режиме работы 1С:Предприятие. Кроме этого конфигурация описывает всевозможные алгоритмы обработки этих данных, содержит информацию о том, как эти

данные должны будут выглядеть на экране и на принтере, и т.д. В дальнейшем платформа 1С:Предприятия на основании этого описания создаст базу данных, которая будет иметь необходимую структуру, и предоставит пользователю возможность работать с этой базой данных.

Для того чтобы систему 1С:Предприятие можно было быстро и легко настраивать на нужные прикладные задачи, все описание, которое содержит конфигурация, состоит из неких логических единиц, называемых объектами конфигурации. Прежде чем мы приступим к созданию первых объектов конфигурации, следует объяснить некоторые приемы работы с конфигуратором. Для того чтобы открыть и закрыть конфигурацию, следует использовать пункты меню *Конфигурация – Открыть конфигурацию* и *Конфигурация – Закрыть конфигурацию*. После того как конфигурация открыта, ее состав появляется в окне дерева конфигурации. Это окно вы можете закрыть, как любое другое окно Windows, при этом конфигурация останется открытой (т.е. доступной для редактирования). Чтобы снова отобразить на экране окно дерева конфигурации, следует воспользоваться командой меню *Конфигурация – Окно конфигурации*.

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

Первый способ. Необходимо установить курсор на тот вид объектов конфигурации, который вас интересует, и в командной панели окна конфигурации нажать кнопку *Действия – Добавить*.

Второй способ. Вы можете воспользоваться контекстным меню, которое вызывается при нажатии на правую клавишу мыши.

Установите курсор на интересующий вас вид объектов конфигурации и нажмите правую клавишу мыши. В появившемся меню выберите пункт *Добавить*.

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

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

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

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

Зададим имя конфигурации *Лабораторный Практикум*.

Соответствующий ему синоним устанавливается автоматически, но его можно изменить по своему усмотрению. Теперь проверим наши первые изменения в режиме 1С:Предприятие. Для этого выполним пункт меню *Отладка – Начать отладку* или нажмем соответствующую кнопку на панели инструментов конфигуратора. Система сама анализирует наличие изменений в конфигурации и выдает соответствующий вопрос об обновлении конфигурации базы данных. На вопрос конфигуратора ответим *Да*, и на экране появится окно 1С:Предприятия.

В заголовке окна мы видим название нашей конфигурации. Пустое пространство – это рабочая область приложения, которая пока ничем не заполнена. Кроме заголовка конфигурации в окне 1С:Предприятия ничего не появилось. И этого следовало ожидать. Мы еще не создали никаких объектов конфигурации и не создали никаких подсистем, в которых бы эти объекты отображались.

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

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

В простых прикладных решениях можно не использовать подсистемы, но мы рассмотрим общий случай, когда подсистемы используются. Подсистемы позволяют выделить в конфигурации функциональные части, на которые логически разбивается создаваемое прикладное решение. Подсистемы верхнего уровня являются основными элементами интерфейса, так как образуют разделы прикладного решения. Каждый объект конфигурации может быть включен в одну или сразу несколько подсистем, в составе которых он будет отображаться.

Даже в такой небольшой конфигурации, как наша, можно выделить несколько функциональных частей, представляющих собой отдельные предметные области. Так, можно выделить в отдельную подсистему все, что имеет отношение к бухгалтерскому учету. Кроме этого, отдельной предметной областью является расчет зарплаты сотрудников предприятия.

Всю производственную деятельность нашей фирмы ООО «Компьютер-сервис» можно разделить на учет материалов и оказание услуг. А кроме этого, для выполнения специальных административных функций с базой данных нам нужно иметь отдельную подсистему, в которую будет иметь доступ только администратор. Закроем приложение и вернемся в конфигуратор. Чтобы создать новые подсистемы, раскроем ветвь Общие в дереве объектов конфигурации, нажав на + слева от нее. Затем выделим ветвь Подсистемы, вызовем ее контекстное меню и выберем пункт *Добавить* или нажмем соответствующую кнопку в командной панели окна конфигурации. После этого система откроет окно редактирования объекта конфигурации. Оно предназначено специально для сложных объектов конфигурации и позволяет путем выполнения последовательных действий быстро создавать такие объекты.

Зададим имя подсистемы – *Бухгалтерия*. На основании имени платформа автоматически создаст синоним – *Бухгалтерия*.

ЗАДАНИЕ:

1. Добавить в конфигурацию оставшиеся 4 подсистемы: *Учет материалов, Оказание услуг, Расчет зарплаты, Предприятие*.

Запустим 1С:Предприятие в режиме отладки и увидим результат наших изменений. Вид разрабатываемого нами приложения изменился. Сразу под главным меню располагается панель разделов приложения, где и отражены созданные нами подсистемы. Разделы представлены в форме гиперссылок, нажав на которые пользователь может открыть связанные с ними документы, справочники, отчеты и т.п. Сейчас состав разделов пуст, так как мы еще не создали наполняющих их объектов конфигурации. Однако порядок расположения подсистем нас не совсем устраивает. Изменим его.

Закроем приложение и вернемся в конфигуратор. Выделим корень дерева объектов конфигурации *Лабораторный Практикум*, нажатием правой кнопки мыши вызовем контекстное меню и выберем пункт *Открыть командный интерфейс конфигурации*.

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

Расположим сначала подсистемы, отражающие производственную деятельность нашей фирмы: *Учет материалов* и *Оказание услуг*, затем бухгалтерскую деятельность и расчет зарплаты сотрудников: *Бухгалтерия* и *Расчет зарплаты*, а затем подсистему *Предприятие*. Запустим 1С:Предприятие в режиме отладки и увидим, что порядок расположения подсистем в панели разделов приложения изменился так, как мы его задали.

Лабораторная работа №2. Создание элементов конфигурации: Справочников. Документов.

Цель работы: Знакомство с объектом конфигурации Справочник. Научиться создавать Справочники.

Порядок выполнения работы:

Объект конфигурации **Справочник** является прикладным и предназначен для работы со списками данных. Объект конфигурации Справочник используется для того, чтобы на его основе платформа создала в базе данных информационную структуру, в которой будет храниться, например, список сотрудников, перечень товаров, список клиентов или поставщиков.

Справочник состоит из *элементов*. Характерной особенностью объекта конфигурации Справочник является то, что пользователь в процессе работы может самостоятельно добавлять новые элементы в справочник: например, добавить в справочник новых сотрудников, создать новый товар или внести нового клиента.

Каждый элемент справочника, как правило, содержит некоторую дополнительную информацию, которая подробнее описывает этот элемент. Например, все элементы справочника Товары могут содержать дополнительную информацию о производителе, сроке годности и др. Набор такой информации является одинаковым для всех элементов справочника, и для описания такого набора используются реквизиты объекта конфигурации Справочник, которые также, в свою очередь, являются объектами конфигурации. Поскольку эти объекты логически связаны с объектом Справочника, они называются *подчиненными*. Большинство реквизитов объекта конфигурации Справочник разработчик создает самостоятельно, однако у каждого объекта конфигурации Справочник существуют два поля «по умолчанию»: *Код* и *Наименование*.

Кроме этого, каждый элемент справочника может содержать некоторый набор информации, которая одинакова по своей структуре, но различна по количеству, предназначена для разных элементов справочника. Так, например, каждый элемент справочника Сотрудники может содержать информацию о составе семьи сотрудника. Для одного сотрудника это будет только супруга, а у другого семья может состоять из супруги, сына и дочери. Для описания подобной информации могут быть использованы табличные части объекта конфигурации Справочник, являющиеся *подчиненными объектами конфигурации*.

Для удобства использования элементы справочника могут быть сгруппированы пользователем по какому-либо принципу. Например, в справочнике Бытовая техника могут быть созданы следующие группы: Холодильники, Телевизоры, Стиральные машины и т.д. Возможность создания таких групп в справочнике задается свойством *Иерархический* объекта конфигурации Справочник. В этом случае элемент справочника, представляющий собой группу, будет являться родителем для всех элементов и групп, входящих в эту группу. Такой вид иерархии называется *иерархией групп и элементов*.

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

Элементы одного справочника могут быть подчинены элементам или группам другого справочника. Например, справочник *ЕдиницыИзмерения* может быть подчинен справочнику *Товары*. Тогда для каждого элемента справочника *Товары* мы сможем указать единицы измерения, в которых этот товар поступает на склад. В системе 1С:Предприятие это достигается путем

указания списка владельцев справочника для каждого объекта конфигурации *Справочник*.

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

В зависимости от того, какие действия мы хотим выполнять со справочником, нам требуется изображать справочник в «разном виде». Например, для того, чтобы выбрать некоторый элемент справочника, удобнее представить справочник в виде списка, а для того, чтобы изменить какой-то элемент справочника, удобнее представить все реквизиты этого элемента справочника на одной форме. Поэтому объект конфигурации *Справочник* может иметь произвольное количество форм, часть из которых можно назначить в качестве основных.

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

Любая форма может быть описана в конфигураторе. Для создания такого описания существует подчиненный объект конфигурации *Форма*. Как правило, он подчинен одному из прикладных объектов, но может существовать и самостоятельно. На основании описания, содержащегося в объекте конфигурации *Форма*, в нужный момент работы пользователя платформа 1С:Предприятие создаст программный объект *Форма*, с которым и будет работать пользователь.

Теперь создадим несколько справочников, которые будем использовать в нашей базе данных. Так как наше ООО «Компьютер-сервис» оказывает услуги по ремонту компьютерной техники, очевидно, что для ведения учета нам потребуется хранить некоторую списочную информацию. Для начала нам понадобится список сотрудников предприятия, которые будут оказывать услуги. Затем нам будет нужен список клиентов, с которыми работает наше ООО «Компьютер-сервис».

После этого нам понадобится перечень услуг, которые может оказывать наше предприятие, и список материалов, которые могут быть израсходованы.

Кроме этого, нам потребуется список складов, на которых могут находиться материалы ООО «Компьютер-сервис». Сначала создадим справочник, в котором будут храниться наименования наших клиентов. Откроем в конфигураторе нашу учебную конфигурацию, выделим в дереве объектов конфигурации ветвь *Справочники* и нажмем кнопку *Добавить* в командной панели окна конфигурации.

В открывшемся окне редактирования объекта конфигурации зададим имя справочника – *Клиенты*. На основании имени платформа автоматически создаст синоним – *Клиенты*. Напомним, что свойство *Синоним* служит для представления объекта в интерфейсе нашей программы. Также у разработчика есть возможность установки дополнительных свойств, определяющих пользовательское представление объектов. Эти свойства задавать не обязательно. Если они не заданы, то для представления объекта в интерфейсе 1С:Предприятия используется синоним объекта конфигурации *Справочник*.

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

автоматически формируются команды открытия списка справочника и команды создания нового элемента справочника.

Если синоним задан во множественном числе, то для команды открытия списка это вполне подходит – *Клиенты*, то есть посмотреть всех клиентов. Но для команды создания элемента справочника – одного клиента – это неудачный вариант. Для этой команды нужно задать представление в единственном числе – *Клиент*. Представление объекта как раз и используется для того, чтобы описать, как будет выглядеть в интерфейсе команда добавления нового клиента. Также оно будет использовано в заголовке формы клиента (если не указано расширенное представление объекта) и в представлении ссылки на клиента.

Расширенное представление объекта определяет заголовок формы объекта, например формы для создания нового элемента справочника. Если это свойство не задано, то вместо него используется свойство *Представление объекта*. Представление списка определяет название списка объектов и используется в названии стандартной команды, например, команды открытия списка объектов – *Клиенты: открыть*. Представление списка нужно задавать тогда, когда синоним задан в единственном числе.

Например, это часто бывает у документов (Приходная накладная). Тогда в представлении списка нужно указывать название объекта конфигурации во множественном числе (Приходные накладные). Расширенное представление списка определяет заголовок формы списка, например формы списка справочника. Если это свойство не задано, то вместо него используется свойство *Представление списка*.

Зададим два свойства *Представление объекта* – *Клиент* и *Представление списка* – *Клиенты*. Последнее можно было и не задавать, так как синоним справочника совпадает со свойством *Представление списка*. В представлении списка вроде бы подразумевается название *Список клиентов*, но идущие подряд строки *Список сотрудников*, *Список клиентов*, *Список складов* не очень хорошо смотрятся в интерфейсе приложения. Нажмем

кнопку *Далее* и перейдем на закладку *Подсистемы* окна редактирования объекта конфигурации *Справочник*. На этой закладке определяется, в каких подсистемах будет отображаться данный справочник.

В списке подсистем мы видим подсистемы, созданные нами ранее при определении структуры приложения. Логично предположить, что список клиентов должен быть доступен в разделе *Оказание услуг*, так как оказываемые услуги относятся к определенному клиенту. *Бухгалтерская отчетность*, формируемая в разделе *Бухгалтерия*, также может быть представлена в разрезе клиентов. Поэтому отметим в списке подсистемы *Бухгалтерия* и *ОказаниеУслуг*. В окне редактирования объекта конфигурации *Справочник* нажмем на закладку *Данные*.

Здесь для нас представляют интерес длина кода и длина наименования.

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

Длина кода – 9 символов. В результате мы сможем использовать коды от 1 до 999999999 – этого вполне достаточно для нашего небольшого ООО «Компьютер-сервис». Перейдем к длине наименования. 25 символов для нас явно мало, увеличим длину наименования до 50.

Прежде чем запускать 1С:Предприятие, настроим интерфейс приложения, чтобы нам было удобнее вводить новые элементы справочника. Дело в том, что для размещения стандартных команд открытия списков и создания новых объектов конфигурации в интерфейсе 1С:Предприятия существует общий стандартный алгоритм, который мы сейчас объясним на примере справочников. Но это справедливо и для документов, планов счетов и т.п.

Команда для открытия списка справочника, как и команда для создания его новых элементов, добавляется в интерфейс тех разделов (подсистем), в которых будет отображаться справочник. Но команда создания новых элементов по умолчанию невидима в интерфейсе приложения. Это объясняется тем, что возможность просматривать списки справочника нужна, как правило, всегда. А возможность создания новых элементов справочника используется не так часто. Поэтому соответствующую команду следует включать только для тех справочников (объектов конфигурации), создание новых элементов которых является «основной» деятельностью для пользователей в данном разделе прикладного решения.

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

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

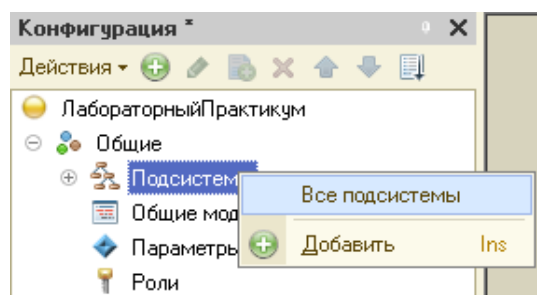


Рисунок 1. Вызов окна редактирования подсистем.

В открывшемся окне *Все подсистемы* слева в списке *Подсистемы* выделим подсистему *Оказание Услуг*. Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы.

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

Включим видимость у этой команды.

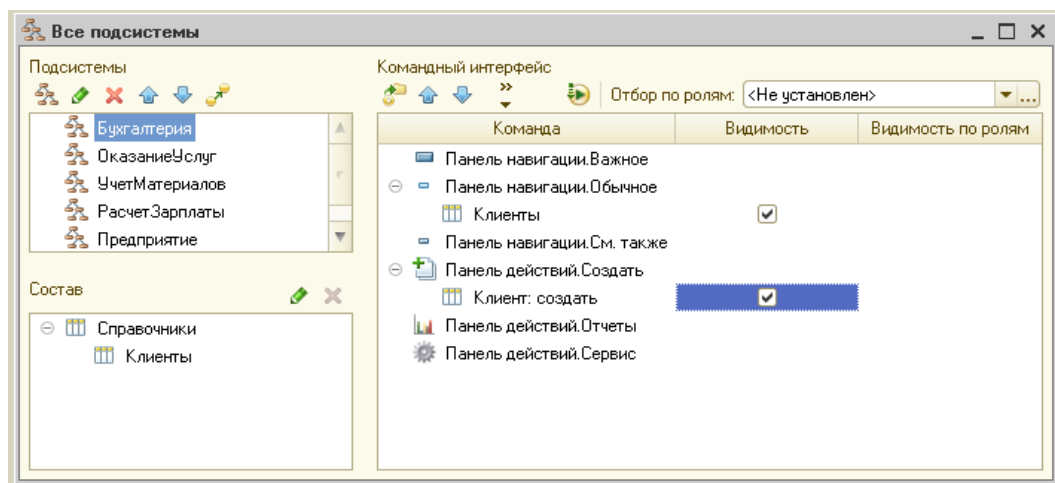


Рисунок 2. Работа с командами подсистемы.

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

Запустим 1С:Предприятие в режиме отладки. Ответим утвердительно на вопрос конфигуратора об обновлении конфигурации и увидим окно, содержащее список изменений в структуре конфигурации, автоматически сгенерированный платформой. В данном случае мы добавили справочник *Клиенты*. Нажмем кнопку *Принять*.

Перед нами откроется окно системы в режиме 1С:Предприятие. Мы видим, что если перейти в раздел *Оказание услуг* или *Бухгалтерия*, то слева в вертикальной области окна появится панель навигации. Сейчас она содержит команду для открытия нашего первого списка – *Клиенты*.

Также в разделе *Оказание услуг* появилась *панель действий*.

Пока наш справочник пуст, поэтому добавим в него несколько элементов. Для этого выполним команду *Клиент* в панели действий раздела *Оказание услуг*. Перед нами откроется форма для создания элемента справочника – основная форма объекта.

Внесем наименование нового клиента – Томашев Илья Викторович. Код вносить не будем, так как он генерируется автоматически. Нажмем *Записать и закрыть*.

При этом в правом нижнем углу появится информационное сообщение о том, какой элемент был создан либо изменен. Кроме этого, нажав на ссылку в

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

ЗАДАНИЕ:

2. Добавить еще двух клиентов – Шевчук Андрей Владимирович и Дворецкий Павел Дмитриевич.
3. Создать справочник «Сотрудники» с табличной частью «ТрудоваяДеятельность», включить его в подсистемы *ОказаниеУслуг* и *РасчетЗарплаты*. Реквизиты табличной части: Организация (строка, длина 100), Начало работы (дата), Окончание работы (дата), Должность (строка, длина 100).
4. Внести информацию о сотрудниках в режиме 1С:Предприятие:
 - а) Савченко Александр Васильевич. Трудовая деятельность: ООО «Савченко и Ко», 01.02.2006 – 31.12.2009, ведущий специалист.
 - б) Шаронов Сергей Сергеевич. Трудовая деятельность: ООО «БИТ-сервис», 01.01.2008 – 31.05.2010, инженер.
 - в) Фоменко Сергей Валерьевич. Трудовая деятельность: ЗАО «Альфа», 01.01.2006 – 31.12.2006, менеджер; ООО «Компьютер-гранд», 01.01.2007 – 01.09.2009, продавец.
5. Создать иерархический справочник «Номенклатура», в котором будет храниться информация об услугах, которые оказывает ООО «Компьютер-сервис» и материалах, которые при этом будут использованы. Сгруппировать *материалы* в одну группу, а *услуги* – в другую. Включить справочник в подсистемы *УчетМатериалов*, *ОказаниеУслуг* и *Бухгалтерия*.
6. В группу *Материалы* добавить элементы: чернила Epson, картридж Canon, транзистор Philips, шлейф IDE, фотобарабан Samsung. В группу *Услуги* добавить: ремонт материнской платы, заправка принтера,

ремонт картриджа.

7. Создать справочник *Склады*, включить в подсистемы *ОказаниеУслуг* и *УчетМатериалов*. Добавить в справочник предопределенный элемент «*Основной склад*» в режиме Конфигуратор.

Документы.

Объект конфигурации *Документ* является прикладным и предназначен для описания информации о совершенных хозяйственных операциях или о событиях, произошедших в жизни организации вообще. На основе объекта конфигурации *Документ* платформа создает в базе данных информационную структуру, в которой будут храниться, например, такие документы, как приходные накладные, или приказы о приеме на работу, или платежные поручения, или счета и т. д.

Характерной особенностью объекта конфигурации *Документ* является то, что в процессе работы пользователь может самостоятельно создавать новые объекты этой структуры – новые документы.

Свойства документа отличается от свойств всех остальных объектов базы данных. *Документ* обладает способностью *проведения*. Факт проведения документа означает, что событие, которое он отражает, повлияло на состояние учета. До тех пор, пока документ не проведен, состояние учета неизменно, и документ – не более чем черновик, заготовка. Как только документ будет проведен – изменения, вносимые документом в учет, могут вступить в силу и состояние учета может быть изменено.

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

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

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

Каждый документ, как правило, содержит информацию, которая подробнее описывает этот документ. Например, каждый документ *ПриходнаяНакладная* может содержать информацию о поставщике товаров, складе, на который приходится товар, и т. д. Набор такой информации является одинаковым для всех документов одного вида, и для описания такого набора используются реквизиты объекта конфигурации *Документ*, являющиеся подчиненными объектами конфигурации. Большинство реквизитов объекта конфигурации *Документ* разработчик создает самостоятельно, однако у каждого объекта конфигурации *Документ* существуют два поля «по умолчанию»: *Дата* и *Номер документа*. Поскольку тип данных *Дата* содержит дату и время с точностью до секунды, этот реквизит и определяет в основном положение документа на оси времени.

Кроме этого, каждый документ содержит, как правило, некоторый набор информации, которая одинакова по своей структуре, но различна по количеству, предназначена для разных документов. Так, например, каждый документ *ПриходнаяНакладная* может содержать список приходуемых товаров. Для описания подобной информации служат табличные части объекта конфигурации *Документ*. Для «визуализации» документа существует несколько основных форм.

Факт проведения документа и необходимость поддержания актуальной последовательности документов на оси событий порождают два различных способа проведения документов: *оперативное* и *неоперативное проведение*.

С оперативным проведением документов связано понятие *оперативной отметки времени*.

Оперативная отметка времени представляет собой значение типа *Дата*, которое формирует система. Оперативная отметка времени создается системой каждый раз при оперативном проведении документа. Ее значение

формируется исходя из текущего времени и последней созданной оперативной отметки.

Если последняя оперативная отметка меньше текущего времени, в качестве новой оперативной отметки принимается текущее время. Если последняя оперативная отметка равна или больше текущего времени, в качестве новой оперативной отметки принимается значение на одну секунду большее, чем старая оперативная отметка времени.

Таким образом, если у объекта конфигурации Документ установлено свойство оперативного проведения, последовательность действий системы будет следующей: при создании нового документа система будет устанавливать ему текущую дату и «нулевое» время. При проведении такого документа (с текущей датой) система установит в качестве даты документа оперативную отметку времени. Если отменить проведение документа и затем провести его снова (не изменяя даты), система установит документу новую оперативную отметку времени.

Если же попытаться перепровести документ, то будет выдан запрос о виде проведения (оперативное или нет). В случае оперативного проведения система установит новую оперативную отметку времени, а при неоперативном проведении время документа будет сохранено прежним.

При попытке проведения (или перепроведения) оперативно проводимого документа с датой, отличающейся от текущей, будет выдано сообщение о том, что оперативное проведение невозможно, и предложено провести документ неоперативно (то есть с сохранением существующей даты и времени документа).

Прежде чем мы приступим к практическому созданию документов, необходимо сделать отступление о том, какие типы данных могут использоваться в системе 1С:Предприятие. В предыдущей главе, когда мы создавали реквизиты справочников или табличных частей, мы всегда указывали тип значения, которое может принимать этот реквизит. Это были

примитивные типы данных: Число, Строка, Дата и Булево. Примитивные типы данных изначально определены в системе и их набор ограничен.

Наряду с такими изначально определенными в любой конфигурации типами могут существовать типы данных, определяемые только конкретной конфигурацией. Такие типы сами образуют объекты конфигурации в момент их создания в конфигураторе.

Например, после того, как мы создали объект конфигурации *Справочник Склады*, сразу же появилось несколько новых типов данных, связанных с этим справочником. Среди них, например, *СправочникСсылка.Склады*. И если теперь мы укажем какому-либо реквизиту этот тип данных, то сможем хранить в нем ссылку на конкретный объект справочника *Склады*.

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

Это небольшое отступление было необходимо потому, что уже при создании первого документа мы столкнемся с использованием типа данных, доступным благодаря объекту конфигурации *Справочник Склады*.

После того, как мы познакомились с объектом конфигурации Документ, создадим несколько таких объектов, чтобы иметь возможность фиксировать события, происходящие в нашем ООО «Компьютер-сервис».

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

Для отражения этих событий в базе данных мы создадим два документа: *Приходная накладная* и *Оказание услуги*. Документ *ПриходнаяНакладная* будет фиксировать факт поступления в нашу организацию необходимых материалов, а документ *ОказаниеУслуги* – фиксировать оказание услуг и расход материалов, которые используются при оказании этих услуг.

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

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

Представление списка, наоборот, зададим во множественном числе как *Приходные накладные*. Нажмем *Далее* и перейдем на закладку *Подсистемы*.

По логике нашей конфигурации список приходных накладных должен быть доступен в разделах *Учет материалов* и *Бухгалтерия*. Перейдем на закладку *Данные* и создадим реквизит документа именем *Склад*. Для этого нажмем кнопку *Добавить* над списком реквизитов документа.

Выберем для реквизита ссылочный тип данных *СправочникСсылка.Склады*. Этот тип стал доступен в конфигурации после создания объекта конфигурации *Справочник Склады*. Теперь покажем, как можно облегчить «жизнь» пользователя при приходовании материалов. Работа в автоматизируемой нами фирме построена таким образом, что, как правило, все поступающие товары приходятся на основной склад.

Поэтому в палитре свойств для созданного нами реквизита *Склад* документа найдем свойство *Значение заполнения*. В качестве значения этого свойства выберем предопределенный элемент справочника *Склады – Основной*.

После этого добавим в документ табличную часть с именем *Материалы*.

Создадим реквизиты табличной части *Материалы*:

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

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

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

Справа в списке *Командный интерфейс* отразятся все команды выбранной подсистемы. В группе *Панель действий. Создать* включим видимость у команды *Приходная накладная: создать*. Запустим 1С:Предприятие в режиме отладки и протестируем получившийся документ.

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

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

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

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

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

Нам осталось только заполнить табличную часть приходной накладной материалами для ремонта телевизоров так, как показано на (рис. 3).

N	Материал	Количество	Цена	Сумма
1	Чернила Epson	10,000	75,00	750,00
2	Картридж Canon	10,000	400,00	4 000,00
3	Фотобарабан Samsung	10,000	580,00	5 800,00

Рисунок 3. Пример заполнения документа.

Нажмем *Провести и закрыть*.

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

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

Нажмем *Провести и закрыть*.

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

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

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

Это совсем не сложно, и для этого нам потребуется сначала создать собственную форму документа, а затем воспользоваться возможностями встроенного языка.

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

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

В этом окне нас интересует закладка *Формы*.

Как мы видим, ни одна из основных форм документа пока не задана. Для того чтобы создать форму документа, нажмем кнопку открытия со значком лупы в поле ввода или кнопку *Добавить* над списком форм. Система вызовет еще один полезный инструмент разработчика – *Конструктор форм*.

Выберем тип формы *Форма документа* и нажмем кнопку *Готово*, согласившись тем самым со всем, что нам предложила система.

Итак, нас интересуют три элемента табличной части: *Материалы*, *Количество*, *МатериалыЦена* и *МатериалыСумма*.

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

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

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

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

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

В модуль формы, в процедуру *МатериалыКоличествоПриИзменении()*, мы и добавим следующий текст:

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

Объясним назначение этих строк.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Добавим объект конфигурации *Общий модуль*.

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

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

Внесем в модуль следующий текст (рис. 4):

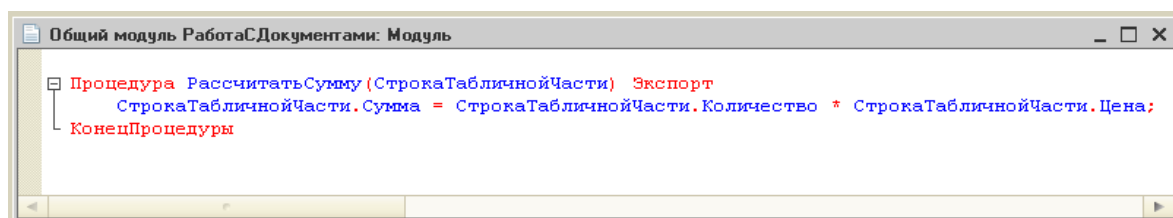


Рисунок 4. Листинг процедуры «Рассчитать сумму».

Прокомментируем этот код. В процедуру *РассчитатьСумму()* мы передаем переменную *СтрокаТабличнойЧасти*, которую мы определили в

обработчике события *ПриИзменении* поля *Количество*. Она содержит данные редактируемой строки табличной части документа *ПриходнаяНакладная*.

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

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

Теперь в модуле нашей формы изменим текст обработчика (рис. 5).

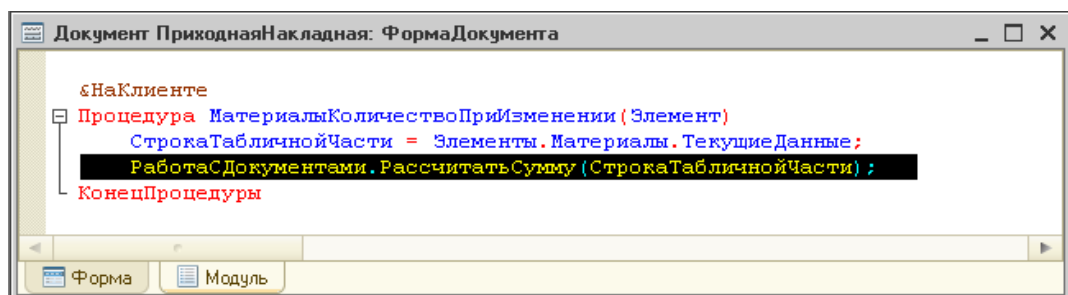


Рисунок 5. Листинг процедуры «Количество при изменении».

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

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

Теперь осталось и для поля *Цена* установить такой же обработчик.

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

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

ЗАДАНИЕ:

1. Создать документ *Оказание услуги* (подсистемы *Оказание услуг* и *Бухгалтерия*). Реквизиты документа: Склад, клиент, мастер, табличная часть *ПереченьНоменклатуры* с реквизитами: номенклатура,

количество, цена, сумма.

2. Создать форму документа *Оказание услуги* и написать в ней процедуры для автоматического расчета суммы аналогично документу *Приходная накладная*.
3. В режиме 1С: Предприятие создать и провести несколько документов по оказанию услуг.

Лабораторная работа №3. Создание элемента конфигурации Регистра накопления.

Цель работы: Знакомство с объектом конфигурации Регистры накопления, его основными свойствами. Научиться создавать объектом конфигурации Регистры накопления.

Порядок выполнения работы:

В конфигурации существуют различные виды модулей. Они могут принадлежать некоторым объектам конфигурации, а могут существовать сами по себе. Текст программы, содержащийся в модулях, будет использоваться платформой в заранее известные моменты работы системы, называемых событиями.

В 1С: Предприятии существуют следующие виды модулей:

- **Модуль управляемого приложения**, выполняется при старте системы в режимах тонкого клиента и веб-клиента.
- **Общие модули**, в них хранятся процедуры и функции, которые вызываются из других модулей системы.
- **Модули объектов** – это, например, модуль элемента справочника или модуль документа. Эти модули вызываются тогда, когда либо программно создается этот объект средствами встроенного языка, либо когда пользователь создает новый элемент интерактивно.
- **Модули формы**. Каждая форма, определенная в конфигурации, имеет свой собственный модуль. Этот модуль выполняется при

создании объекта *УправляемаяФорма* встроенного языка.

- **Модуль сеанса**, автоматически выполняется при старте системы в момент загрузки конфигурации. Предназначен для инициализации параметров сеанса и отработки действий, связанных с сеансом работы.
- **Модуль внешнего соединения**, предназначен для размещения текстов функций и процедур, которые могут вызываться в сессии внешнего соединения.
- **Модули менеджеров**. Для каждого прикладного объекта существует менеджер, предназначенный для управления этим объектом как объектом конфигурации.

Каждый модуль связан с остальной частью конфигурации. Эта связь называется *контекстом модуля*. **Контекст модуля** определяет набор доступных во время выполнения модуля объектов, переменных, процедур и функций.

Контекст модуля образуется:

- Локальным контекстом самого модуля формы;
- Реквизитами формы, которой «принадлежит» модуль;
- Свойствами и методами объекта *УправляемаяФорма* встроенного языка;
- Свойствами и методами расширения формы, определяемого типом того объекта, данные которого содержатся в основном реквизите формы;
- Глобальным контекстом, в том числе не глобальными общими модулями и экспортируемыми процедурами и функциями глобальных общих модулей;
- Экспортируемыми переменными, процедурами и функциями модуля управляемого приложения.

Итак, мы подошли к одному из главных моментов разработки любой конфигурации – созданию механизма учета накопления данных.

Казалось бы, все необходимое мы с вами уже создали: у нас есть что расходовать и приходить (справочники), и у нас есть чем расходовать и приходить (документы). Осталось только построить несколько отчетов, и автоматизация ООО «Компьютер-сервис» будет закончена.

Однако это не так.

Во-первых, путем анализа документов можно, конечно, получить требуемые нам выходные данные, но представьте, что завтра ООО «Компьютер-сервис» решит немного изменить свои бизнес-процессы, и нам потребуется ввести в конфигурацию еще один документ (или несколько документов!).

Например, сейчас мы полагаем, что товары поступают в ООО и затем расходуются. Руководство захотело усилить материальный контроль и решило приходить товары на основной склад организации и затем выдавать их материально ответственным лицам. В этом случае нам придется добавить в конфигурацию еще один документ, который будет фиксировать перемещение материалов между основным складом и материально ответственными лицами. И очевидно, нам придется переработать все отчеты, которые были нами созданы к этому моменту с тем, чтобы они учитывали изменения, вносимые новым документом. А представьте, если в нашей конфигурации не два, а двадцать документов.

Во-вторых, отчеты, анализирующие документы, будут работать довольно медленно, что будет вызывать раздражение пользователей и недовольство руководителей.

Поэтому в системе 1С:Предприятие есть несколько объектов конфигурации, которые позволяют создавать в базе данных структуры, предназначенные для накопления информации в удобном для последующего анализа виде.

Использование таких «хранилищ» данных позволяет нам, с одной стороны, накапливать в них данные, поставляемые различными документами (или другими объектами базы данных), а с другой стороны, легко создавать

нужные нам отчеты или использовать эти данные в алгоритмах работы конфигурации.

В конфигурации существует несколько объектов, называемых *регистрами*, для описания подобных «хранилищ».

Объект конфигурации **Регистр накопления** является прикладным и предназначен для описания структуры накопления данных. На основе объекта конфигурации *Регистр накопления* платформа создает в базе данных информационную структуру, в которой будут накапливаться данные, «поставляемые» различными объектами базы данных. Эти данные будут храниться в регистре в виде отдельных записей, каждая из которых имеет одинаковую, заданную в конфигураторе структуру.

Отличительной особенностью регистра накопления является то, что он не предназначен для интерактивного редактирования пользователем. Разработчик может при необходимости предоставить пользователю возможность редактировать регистр накопления, но предназначение регистра накопления заключается в том, чтобы его модификация производилась на основе алгоритмов работы других объектов базы данных, а не в результате непосредственных действий пользователя.

Основным назначением регистра накопления является накопление числовой информации в разрезе нескольких измерений, которые описываются разработчиком в соответствующем объекте конфигурации *Регистр накопления* и являются подчиненными объектами конфигурации.

Виды числовой информации, накапливаемой регистром накопления, называются *ресурсами* и также являются подчиненными объектами и описываются в конфигураторе.

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

Изменение состояния регистра накопления происходит, как правило, при проведении документа, и заключается в том, что в регистр добавляется

некоторое количество записей. Каждая запись содержит значения измерений, значения приращений ресурсов, ссылку на документ, который вызвал эти изменения (регистратор) и «направление» приращения (приход или расход). Такой набор записей называется движениями регистра накопления. Каждому движению регистра накопления всегда должен соответствовать регистратор.

Кроме того, регистр накопления может хранить дополнительную информацию, описывающую каждое движение. Набор такой дополнительной информации задается разработчиком при помощи реквизитов объекта конфигурации *Регистр накопления*.

Итак, откроем в конфигураторе нашу учебную конфигурацию и добавим новый объект конфигурации *Регистр накопления* с именем *ОстаткиМатериалов*, который будет принадлежать подсистемам *Бухгалтерия*, *Учет материалов*, *Оказание услуг*. Выделим закладку *Данные* и создадим измерения регистра:

Материал, тип СправочникСсылка.Номенклатура;

Склад, тип СправочникСсылка.Склады;

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

Далее сформируем движения регистра накопления *ОстаткиМатериалов* в процессе проведения двух созданных нами документов *ПриходнаяНакладная* и *ОказаниеУслуг*.

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

Откроем окно редактирования объекта конфигурации *Документ ПриходнаяНакладная*, перейдем на закладку *Движения*, раскроем список *Регистры накопления* и отметим регистр накопления *ОстаткиМатериалов*.

Запустим конструктор движений. В списке *Регистры* перечислены регистры, в которых документ может создавать движения. В списке *Реквизиты документа* должны находиться исходные данные для создания движений – реквизиты документа *ПриходнаяНакладная*. В таблице *Поле-*

Выражение должны быть заданы формулы, по которым будут вычисляться значения измерений и ресурсов регистра при записи движений.

В поле выбора *Табличная часть* выберем табличную часть нашего документа – *Материалы*. Нажмем кнопку *Заполнить выражения*. В нижнем окне сформируется соответствие полей регистра и выражений для их расчета.

Нажмем *ОК* и посмотрим, какой текст сформировал конструктор в модуле документа *Приходная накладная*.

```
Процедура ОбработкаПроведения(Отказ, Режим)
  {{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
  // Данный фрагмент построен конструктором.
  // При повторном использовании конструктора, внесенные вручную
  изменения будут утеряны!!!
  Движения.ОстаткиМатериалов.Записывать = Истина;
  Для Каждого ТекСтрокаМатериалы Из Материалы Цикл
    // регистр ОстаткиМатериалов Приход
    Движение = Движения.ОстаткиМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.Склад = Склад;
    Движение.Количество = ТекСтрокаМатериалы.Количество;
  КонецЦикла;
  }}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Конструктор создал обработчик события *ОбработкаПроведения* объекта конфигурации *ПриходнаяНакладная*, поместил его в модуль объекта и открыл текст модуля.

ЗАДАНИЕ:

1. В режиме 1С:Предприятия перепровести все приходные накладные;
2. В режиме *Конфигуратор* создать движения для документа *Оказание услуги по регистру Остатки материалов*;
3. В режиме 1С:Предприятия перепровести все документы по оказанию услуг.

Лабораторная работа № 4. Создание элемента конфигурации

Отчеты. Создание Макетов.

Цель работы: Знакомство с объектом конфигурации *Отчеты*, его основными свойствами. Научиться создавать объектом конфигурации *Отчеты*.

Порядок выполнения работы:

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

Теперь у нас все готово для того, чтобы можно было получать выходные данные. Поэтому приступим к созданию отчета, который будет показывать нам приход, расход и остатки материалов.

Создадим новый объект конфигурации *Отчет* и назовем его *Материалы*. Откроем окно редактирования объекта конфигурации *Отчет* *Материалы* и создадим схему компоновки данных. Для этого нажмем кнопку *Открыть схему компоновки данных*.

Так как у отчета, который мы создаем, еще не существует схемы компоновки данных, платформа предложит создать новую схему. Схема компоновки данных с точки зрения конфигурации является макетом, поэтому будет открыт конструктор макета, предлагающий выбрать единственный тип макета – *Схема компоновки данных*. Нажмем кнопку *Готово*.

Платформа создаст новый макет, содержащий схему компоновки данных, и сразу же откроет конструктор схемы компоновки данных.

Конструктор обладает большим количеством возможностей для визуального проектирования отчетов, но мы сейчас воспользуемся только самыми простыми его возможностями и определим те данные, которые

хотим видеть в результате работы нашего отчета. Добавим новый набор данных – *запрос*.

Для того чтобы создать текст запроса, запустим конструктор запроса – кнопка *Конструктор запроса*.

В списке *База данных* представлены таблицы запросов. На основе их данных мы имеем возможность построить отчет. Если раскрыть ветку *РегистрыНакопления*, то мы увидим, что кроме таблицы регистра *ОстаткиМатериалов* в этой ветке присутствуют еще несколько виртуальных таблиц, которые формирует система.

Поскольку мы хотим видеть как остатки материалов, так и информацию об их поступлении и расходовании, нас будет интересовать виртуальная таблица *ОстаткиМатериалов.ОстаткиИОбороты*. Раскроем ее.

Как вы видите, эта таблица содержит материал, склад и кроме этого начальные и конечные остатки, а также значения прихода, расхода и оборотов для всех ресурсов регистра *ОстаткиМатериалов*.

Начнем выбирать поля таблицы в нужном нам порядке двойным щелчком мыши. Сначала выберем *Склад* и *Материал*. Затем выберем *Количество НачальныйОстаток*, *КоличествоПриход*, *КоличествоРасход* и в заключение *КоличествоКонечныйОстаток*.

В результате окно Поля должно быть заполнено следующим образом (рис. 6):

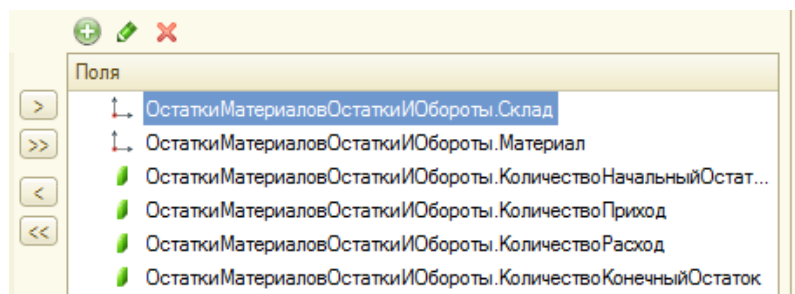


Рисунок 6. Фрагмент конструктора запросов.

Нажмем ОК и вернемся в конструктор схемы компоновки данных.

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

того, можно снова вызвать конструктор запроса и отредактировать запрос при помощи него.

Обратим внимание на список полей системы компоновки данных, который платформа заполнила в верхней части конструктора. В нем отображаются поля, которые доступны у текущего набора данных. В нашем случае система 1С:Предприятие заполнила данный список автоматически, из текста запроса, и нет необходимости в его ручной настройке.

Перейдем на закладку *Настройки* и добавим группировку (контекстное меню – *Новая группировка*). При этом не станем указывать поле группировки, а просто нажмем *ОК*. В отчет будут выводиться детальные записи из информационной базы – записи, получаемые в результате выполнения запроса без итогов.

Теперь настроим поля, которые будут выводиться в результат отчета.

Для этого перейдем на закладку *Выбранные поля* и перенесем мышью из списка доступных полей:

- Склад;
- Материал;
- КоличествоНачальныйОстаток;
- КоличествоПриход;
- КоличествоРасход;
- КоличествоКонечныйОстаток.

Затем перейдем на закладку *Параметры* и укажем, что параметры отчета *Дата начала* и *Дата окончания* будут включены в состав пользовательских настроек и эти настройки будут находиться непосредственно в форме отчета, то есть будут «быстрыми» настройками (рис. 7).

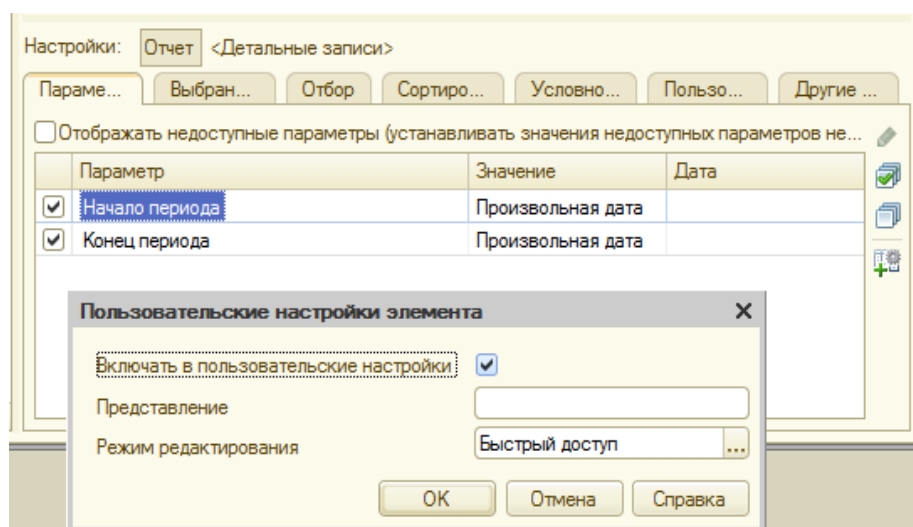


Рисунок 7. Настройка отображения параметров.

В заключение определим, что данный отчет будет отображаться в подсистемах *Учет материалов*, *Оказание услуг* и *Бухгалтерия*.

Далее познакомимся с объектом конфигурации *Макет*.

Объект конфигурации **Макет** предназначен для хранения различных форм представления данных, которые могут потребоваться каким-либо объектам конфигурации или всему прикладному решению в целом. Макет может содержать табличный или текстовый документ, двоичные данные, HTML-документ или Active Document, графическую или географическую схему, схему компоновки данных или макет оформления схемы компоновки данных. Макеты могут существовать как сами по себе (общие макеты), так и быть подчинены какому-либо объекту конфигурации.

Одно из предназначений макета, подчиненного объекту конфигурации и содержащего табличный документ, – создание печатной формы этого объекта.

Создание печатной формы заключается в конструировании ее составных частей – *именованных областей*, из которых затем «собирается» готовая печатная форма. Порядок заполнения областей данными и порядок вывода их в итоговую форму описывается при помощи встроенного языка.

Печатная форма может включать в себя различные графические объекты: картинки, OLE-объекты, диаграммы и т. д.

Помимо создания макета «вручную», конфигуратор предоставляет разработчику возможность воспользоваться специальным инструментом – *конструктором печати*, который берет на себя большинство рутинной работы по созданию макета.

Наша цель будет заключаться в создании печатной формы документа *Оказание услуги*.

Откроем в конфигураторе окно редактирования объекта конфигурации *Документ ОказаниеУслуги*. Перейдем на закладку *Макеты* и запустим *конструктор печати*.

На первом шаге укажем, что новая процедура, которая будет создана конструктором для формирования печатной формы документа, будет располагаться в модуле формы.

На втором шаге нажатием двойной стрелки определим, что все реквизиты нашего документа будут отображены в шапке печатной формы.

На третьем шаге точно так же определим, что все реквизиты табличной части документа будут отображены в печатной форме.

На четвертом шаге конструктор предложит сформировать нам подвал (нижнюю часть) печатной формы. Мы не станем ничего указывать (подвал в данном случае использовать не будем) и перейдем к пятому шагу.

Здесь ничего менять не будем. Тем самым согласимся с тем, что команда для вызова процедуры формирования печатной формы будет помещена в командную панель формы, в раздел *Важное*.

Нажмем *ОК*. В конфигураторе откроется модуль команды *Печать*, модуль менеджера документа *ОказаниеУслуги* и макет этого документа.

Заметим, что разработчик может создать макет печатной формы с нуля и для его вывода создать соответствующую команду и кнопку в форме документа, но в данном случае всю работу сделал конструктор печати.

Следует сказать о том, что каждая ячейка редактируемого нами табличного документа может содержать либо текст, либо некоторый параметр, либо шаблон.

Текст, содержащийся в ячейке, будет показан на экране.

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

Шаблон представляет собой текстовую строку, в определенные места которой будут вставлены значения параметров.

ЗАДАНИЕ:

1. В режиме 1С: Предприятие проверить работоспособность отчета *Остатки материалов* и процедуры *Печать* в форме документа *Оказание услуги*.
2. Отредактировать в режиме Конфигуратор печатную форму документа *Оказание услуги* таким образом, чтобы в подвале табличной части был показан итог по колонке *Сумма*.

Лабораторная работа № 5. Создание элемента конфигурации

Регистры сведений. Перечисления.

Цель работы: Знакомство с объектом конфигурации Регистры сведений. Перечисления, их основными свойствами. Научиться создавать объекты конфигурации Регистры сведений. Перечисления.

Порядок выполнения работы:

Начнем мы с того, что обратим ваше внимание на документ *ОказаниеУслуги*. Как вы помните, в этом документе мы выбираем услугу, которая оказывается, и затем указываем цену.

Очевидно, что в ООО «Компьютер-сервис» существует перечень услуг, который определяет стоимость каждой услуги. Казалось бы, стоимость услуги является неотъемлемым свойством самой услуги, и поэтому ее следует добавить в качестве реквизита справочника *Номенклатура*.

Однако стоимость услуг имеет особенность меняться со временем. И может сложиться такая ситуация, когда нам потребуется внести изменения или уточнения в один из ранее проведенных документов *ОказаниеУслуги*. В

этом случае мы не сможем получить правильную стоимость услуги, поскольку в реквизите справочника будет храниться последнее введенное значение.

Кроме того, не исключено, что руководство ООО «Компьютер-сервис» пожелает видеть, как зависит прибыль предприятия от изменения стоимости оказываемых услуг. В этом случае просто необходимо будет иметь возможность анализировать изменение стоимости услуг во времени.

Поэтому для хранения стоимости услуг мы используем новый пока еще для нас объект – регистр сведений.

Объект конфигурации **Регистр сведений** является прикладным и предназначен для описания структуры хранения данных в разрезе нескольких измерений. На основе объекта конфигурации *Регистр сведений* платформа создает в базе данных информационную структуру, в которой может храниться произвольная информация, «привязанная» к набору измерений.

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

Следующей важной особенностью регистра сведений является его способность (при необходимости) хранить данные с привязкой ко времени. Благодаря этому регистр сведений может хранить не только актуальные значения данных, но и историю их изменения во времени. Регистр сведений, использующий привязку ко времени, называют обычно *периодическим регистром сведений*.

Периодичность регистра сведений можно определить одним из следующих значений:

- в пределах секунды;
- в пределах дня;

- в пределах месяца;
- в пределах квартала;
- в пределах года;
- в пределах регистратора (если установлен режим записи – Подчинение регистратору).

Периодический регистр сведений всегда содержит служебное поле *Период*, добавляемое системой автоматически. Оно имеет тип *Дата* и служит для указания факта принадлежности записи к какому-либо периоду. При записи данных в регистр платформа всегда приводит значение этого поля к началу периода, в который он попадает. Например, если в регистр сведений с периодичностью в пределах месяца записать данные, в которых период указан как 08.14.2013, то регистр сохранит эти данные со значением периода, равным 01.04.2013.

Как и для других регистров, система контролирует уникальность записей для регистра сведений. Однако если для прочих регистров уникальным идентификатором записи является регистратор и номер строки, то для регистра сведений применяется другой принцип формирования ключевого значения.

Ключом записи, однозначно идентифицирующим запись, является в данном случае совокупность значений измерений регистра и периода (в случае, если регистр сведений периодический). *Регистр сведений* не может содержать несколько записей с одинаковыми ключами.

Если продолжать сравнение с регистром накопления, то можно сказать, что регистр сведений предоставляет больше свободы в редактировании хранимых данных. Наряду с возможностью использования в режиме подчинения регистратору (когда записи регистра сведений «привязаны» к документу-регистратору) регистр сведений может применяться и в независимом режиме, в котором пользователю предоставляется полная свобода интерактивной работы с данными регистра. Регистр сведений, не

использующий подчинение регистратору, называют *независимым регистром сведений*.

Приступим к созданию периодического регистра сведений, который будет хранить развернутые во времени розничные цены материалов и стоимости услуг, оказываемых нашим ООО «Компьютер-Сервис».

Откроем конфигуратор и создадим новый объект конфигурации *Регистр сведений*. Назовем его *Цены*. Установим периодичность этого регистра в пределах секунды.

Обратите внимание на свойство *Режим записи*. По умолчанию оно имеет значение – *Независимый*, то есть мы создаем независимый регистр сведений и сможем в дальнейшем вводить в него данные без использования регистратора, «вручную».

Определим, что наш регистр будет доступен в подсистемах *Бухгалтерия, Учет материалов, Оказание услуг*.

Перейдем на закладку *Данные* и создадим измерение *регистра Номенклатура* с типом *СправочникСсылка.Номенклатура*. Укажем, что это измерение будет ведущим.

Свойство *Ведущее* имеет смысл использовать лишь тогда, когда измерение имеет тип ссылки на объект базы данных. *Установка свойства Ведущее* будет говорить о том, что запись регистра сведений представляет интерес, пока существует этот объект. При удалении объекта все записи регистра сведений по этому объекту тоже будут автоматически удалены.

После этого создадим новый ресурс *Цена*, тип *Число*, длина 15, точность 2, неотрицательное.

Теперь запустим 1С:Предприятие в режиме отладки и посмотрим, как работает наш периодический регистр сведений *Цены*.

Зададим стоимость услуг и материалов следующим образом (рис. 8):

	13.07.2010 16:36:00	Чернила Epson	75,00
	13.07.2010 16:36:36	Картридж Canon	400,00
	13.07.2010 16:38:09	Фотобарабан Samsung	580,00
	13.07.2010 16:38:48	Шлейф IDE	20,00
	13.07.2010 16:39:08	Транзистор Philips	30,00
	13.07.2010 16:41:34	Ремонт материнской платы	450,00
	13.07.2010 16:42:07	Ремонт картриджа	800,00
	13.07.2010 16:42:58	Заправка принтера	170,00

Рисунок 8. Образец заполнения регистра сведений.

Итак, мы с вами имеем очень полезную возможность в нашей программе – возможность установки цен на услуги и материалы. Поскольку цены хранятся с привязкой к дате, мы можем заранее установить новые цены и быть уверенными в том, что новые цены вступят в действие не раньше указанной для них даты.

Теперь посмотрим, как можно использовать заданные нами цены в документе *ОказаниеУслуги*.

Наша задача заключается в следующем. *Цена номенклатуры* у нас хранится теперь в отдельном регистре сведений. Когда мы создаем или изменяем документ *ОказаниеУслуги* и добавляем в табличную часть какую-либо номенклатуру, нам хочется, чтобы одновременно с этим в документ подставлялась бы сразу и актуальная цена этой номенклатуры, полученная из регистра сведений и соответствующая дате документа.

Сначала мы создадим функцию, которая будет возвращать нам актуальную розничную цену номенклатуры.

Откроем конфигуратор, в ветке *Общие – Общие модули* создадим новый объект конфигурации Модуль и назовем его *РаботаСоСправочниками*.

Установим флажок *Вызов сервера* для того, чтобы экспортные процедуры и функции этого модуля можно было вызывать с клиента.

Поместим в модуле следующий текст:

```

Функция РозничнаяЦена(АктуальнаяДата, ЭлементНоменклатуры) Экспорт
    Отбор = Новый Структура("Номенклатура",ЭлементНоменклатуры);
    ЗначенияРесурсов = РегистрыСведений.Цены.ПолучитьПоследнее
(АктуальнаяДата, Отбор);
    Возврат ЗначенияРесурсов.Цена;
КонецФункции

```

Для получения розничной цены мы будем передавать в функцию два параметра:

- *АктуальнаяДата* – параметр типа Дата определяет точку на оси времени, в которой нас интересует значение розничной цены.
- *ЭлементНоменклатуры* – ссылка на элемент справочника *Номенклатура*, для которого мы хотим получить розничную цену.

В теле процедуры мы сначала создаем вспомогательный объект *Отбор*. С его помощью определяем, что нас будут интересовать записи регистра, в которых измерение *Номенклатура* равно переданной в процедуру ссылке на элемент справочника.

Во второй строке мы обращаемся к менеджеру регистра сведений *Цены (РегистрыСведений.Цены)* и выполняем метод *ПолучитьПоследнее()*, который возвращает нам значения ресурсов наиболее поздней записи регистра, соответствующей передаваемой дате (*АктуальнаяДата*) и значениям измерений регистра (*Отбор*).

Значения ресурсов возвращаются в объекте *Структура*, поэтому в следующей строке мы получаем искомую нами розничную цену, просто указав имя нужного нам ресурса регистра через точку (*ЗначенияРесурсов.Цена*).

Теперь эту функцию нужно вызвать в некоторый момент работы документа.

Задача, которая перед нами стоит, заключается в следующем. При создании документа *ОказаниеУслуги* нам необходимо обеспечить автоматическое заполнение поля *Цена* после того, как пользователь выберет услугу. *Причем* цена услуги должна определяться исходя из даты создаваемого документа.

Найдем в конфигураторе документ *ОказаниеУслуги* и откроем его форму *ФормаДокумента*. Откроем свойства поля ввода, расположенного в колонке *Номенклатура*, и внизу списка найдем событие *При изменении*. Нажмем на

кнопку с лупой и в открывшейся заготовке обработчика события напишем следующий текст:

```
Процедура НоменклатураПереченьНоменклатурыПриИзменении()  
СтрокаТабличнойЧасти = Элементы.ПереченьНоменклатуры.ТекущиеДанные;  
СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена  
(Объект.Дата, СтрокаТабличнойЧасти.Номенклатура);  
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);  
КонецПроцедуры
```

Прокомментируем содержимое обработчика.

Первая строка обработчика вам уже знакома: мы получаем текущую строку табличной части документа, так как она нам понадобится в дальнейшем.

Во второй мы устанавливаем полученную цену в документе, вызывая нашу процедуру *РозничнаяЦена*. Первым параметром мы передаем дату документа, на которую необходимо получить цену. Вторым параметром мы передаем ссылку, которую отображает элемент управления формой, вызвавший это событие (*СтрокаТабличнойЧасти.Номенклатура*), то есть ссылку на элемент справочника *Номенклатура*.

В заключение мы вызываем нашу процедуру *РассчитатьСумму* из общего модуля *РаботаСДокументами*, чтобы она пересчитала итоговую сумму в строке нашего документа.

До сих пор мы с вами не обращали внимания на то, что у нас нет никакого признака, по которому мы могли бы сказать, чем является конкретный элемент справочника *Номенклатура*: материалом или услугой. То, что все элементы справочника разложены у нас по некоторым группам, не может являться надежным критерием оценки: группы можно удалить, переименовать, сгруппировать элементы по другим принципам...

Поэтому нам требуется некоторый признак, позволяющий однозначно определять принадлежность элемента справочника к материалам или услугам, независимо от изменения иерархической структуры справочника.

Объект конфигурации **Перечисление** является прикладным и предназначен для описания структуры хранения постоянных наборов

значений, не изменяемых в процессе работы конфигурации. На основе объекта конфигурации *Перечисление* платформа создает в базе данных информационную структуру, в которой может храниться набор некоторых постоянных значений.

Набор всех возможных значений, которые содержит перечисление, задается при конфигурировании системы, и пользователь не может изменять их, удалять или добавлять новые.

Из этого следует важная особенность перечисления: значения перечисления не «обезличены» для конфигурации, на них могут опираться алгоритмы работы программы.

Откроем конфигуратор и создадим сначала новый объект конфигурации *Перечисление* с именем *ВидыНоменклатуры*.

На закладке *Данные* добавим два значения перечисления: *Материал* и *Услуга*. Затем добавим в справочник *Номенклатура* новый реквизит *ВидНоменклатуры* с типом *ПеречислениеСсылка.ВидыНоменклатуры*.

ЗАДАНИЕ:

1. Проверить работу регистра сведений *Цены* в режиме 1С: Предприятие.
2. В режиме Конфигуратор внести изменения в модуль документа *Оказание услуги* таким образом, чтобы происходило списание только той номенклатуры, которая является материалом.

Лабораторная работа №6. Проведение документа по нескольким регистрам.

Цель работы: Знакомство с методикой проведения документов по нескольким регистрам.

Порядок выполнения работы:

До сих пор мы с вами учитывали только количественное движение материалов в ООО «Компьютер-сервис». Для этих целей мы создали регистр накопления *ОстаткиМатериалов*. Однако, как вы, наверное, догадываетесь,

одного только количественного учета совершенно недостаточно для нужд нашего ООО.

Очевидно, что необходимо также знать, какие денежные средства были затрачены на приобретение тех или иных материалов, и каковы материальные запасы ООО «Компьютер-сервис» в денежном выражении.

После того, как мы начали автоматизировать наше предприятие, руководство ООО «Компьютер-сервис» высказало пожелание, чтобы весь суммовой учет материалов велся бы теперь по средней стоимости. То есть при закупке материалов они должны учитываться в ценах приобретения, а при расходе – по средней стоимости, которая рассчитывается исходя из общей суммы закупок данного материала и общего количества этого материала, находящегося в ООО.

Поскольку подобная информация имеет совершенно другую структуру, нежели количественный учет, для хранения данных об общей стоимости тех или иных материалов мы будем использовать еще один регистр накопления *СтоимостьМатериалов*.

Таким образом, документы *ПриходнаяНакладная* и *ОказаниеУслуги* должны будут создавать движения не только в регистре *ОстаткиМатериалов*, но одновременно и в регистре *СтоимостьМатериалов*, отражая изменения суммового учета.

Регистр *СтоимостьМатериалов* совсем не сложен, поэтому мы не будем подробно останавливаться на его создании. Этот регистр будет иметь всего одно измерение – *Материал* с типом *СправочникСсылка.Номенклатура* и один ресурс – *Стоимость* с длиной 15 и точностью 2.

Добавим наш регистр в подсистемы *Бухгалтерия*, *УчетМатериалов*, *ОказаниеУслуг*.

Теперь мы можем приступить к внесению изменений в процедуры проведения документов.

Начнем с самого простого – документа *ПриходнаяНакладная*.

Откроем в конфигураторе окно редактирования объекта конфигурации *Документ ПриходнаяНакладная* и перейдем на закладку *Движения*. В списке регистров отметим, что документ будет создавать теперь движения и по регистру *СтоимостьМатериалов*.

Внесем необходимые дополнения прямо в обработчик события *ОбработкаПроведения* документа *ПриходнаяНакладная*, не используя конструктор движений.

Перейдем на закладку прочее и откроем модуль объекта. Откроем процедуру обработчика события *Обработка проведения*. Добавим строки кода, создающие движения в регистре *СтоимостьМатериалов*. В результате процедура обработка проведения будет выглядеть следующим образом:

```
Процедура ОбработкаПроведения(Отказ, Режим)
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;

    Движение = Движения.СтоимостьМатериалов.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Материал = ТекСтрокаМатериалы.Материал;
    Движение.Стоимость = ТекСтрокаМатериалы.Сумма;

КонецЦикла;
КонецПроцедуры
```

ЗАДАНИЕ:

1. Перепровести все документы по приходованию материалов.
2. Создать в документе *ОказаниеУслуги* реквизит *Стоимость* для отражения себестоимости материалов.
3. Создать движения для документа *ОказаниеУслуги* по регистру *СтоимостьМатериалов*.
4. Перепровести все документы по оказанию услуг с учетом нового

реквизита.

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

Дело в том, что при учете услуг важны совершенно другие критерии, нежели при учете материалов. Прежде всего, бессмысленно говорить о том, сколько услуг было и сколько их осталось, важна только сумма и количество услуг, которые были оказаны за определенный промежуток времени. Кроме этого интересны следующие моменты:

- какие именно услуги были оказаны (чтобы составить рейтинг услуг);
- какому именно клиенту оказывались услуги (чтобы предоставить ему скидку от объема оплаченных ранее услуг, например);
- какой мастер предоставлял услуги (чтобы начислить ему заработную плату).

Очевидно, что существующие регистры накопления совершенно не подходят для решения таких задач. Поэтому мы создадим еще одно «хранилище» данных, которое будет использоваться в нашей программе, – *оборотный регистр накопления Продажи*.

Регистры накопления могут быть регистрами остатков и регистрами оборотов.

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

Оборотный регистр накопления очень похож на знакомый уже нам регистр остатков, для которого понятие «остаток» не имеет смысла.

Оборотный регистр накапливает только обороты, остатки ему безразличны. Поэтому единственной виртуальной таблицей, которую будет создавать система для такого регистра, будет таблица оборотов.

В остальном оборотный регистр ничем не отличается от регистра остатков.

Следует сказать об одной особенности конструирования регистров накопления, напрямую связанной с возможностью получения остатков.

При создании оборотного регистра накопления нет особой сложности в определении того, какие именно параметры должны являться измерениями регистра – мы можем назначить в качестве его измерений любые нужные нам параметры.

Совсем иная ситуация в случае регистра накопления, поддерживающего накопление остатков. Для него выбор измерений должен выполняться исходя из того, что движения регистра могут быть осуществлены в две стороны: приход и расход. Таким образом, в качестве измерений нужно выбирать те параметры, по которым движения точно будут осуществляться как в одну, так и в другую сторону.

Например, если ведется учет материалов в разрезах номенклатуры и склада, очевидно, что и номенклатура, и склад могут быть измерениями, поскольку как приход, так и расход материалов всегда будут осуществляться с указанием конкретной номенклатуры и конкретного склада. Если же в этой ситуации появляется желание отразить учет материалов еще и в разрезе поставщика, то здесь уже нужно исходить из конкретной схемы учета, принятой на предприятии.

Скорее всего, при поступлении материалов поставщик будет указан, а вот при расходе материалов, с большой долей вероятности, поставщик указываться не будет, так как в большинстве случаев это совершенно лишняя информация. Значит, поставщика следует добавить как реквизит регистра накопления.

Если же при расходе материалов поставщик будет указываться наверняка, имеет смысл добавить поставщика в измерения регистра.

Иными словами, по каждому из измерений регистра накопления остатков изменение ресурсов обязательно должно осуществляться в обе стороны: приход и расход.

Для реквизитов регистра этот принцип неважен. По реквизитам регистра, ресурсы могут только приходоваться или только расходоваться.

Нарушение этого принципа построения регистров накопления будет вести к непроизводительному использованию ресурсов системы и как следствие замедлению работы и падению производительности.

Теперь, когда мы знаем «практически все» о регистрах накопления, откроем конфигуратор и создадим новый объект конфигурации *Регистр накопления*. Назовем его Продажи и определим вид регистра – *Обороты*.

Укажем, что этот регистр будет отображаться в подсистемах *Бухгалтерия, ОказаниеУслуг, УчетМатериалов*.

На закладке *Данные* создадим измерения регистра:

- Номенклатура, тип *СправочникСсылка.Номенклатура*,
- Клиент, тип *СправочникСсылка.Клиенты*,
- Мастер, тип *СправочникСсылка.Сотрудники*.
- У регистра будет три ресурса:
- Количество, тип *Число*, длина 15, точность 3,
- Выручка, тип *Число*, длина 15, точность 2,
- Себестоимость, тип *Число*, длина 15, точность 2.

Откроем окно редактирования объекта конфигурации *Документ ОказаниеУслуги* и на закладке *Движения* укажем, что этот документ будет создавать движения по регистру *Продажи*.

Далее откроем в конфигураторе модуль объекта конфигурации *Документ ОказаниеУслуги* и найдем в нем процедуру обработчика события *ОбработкаПроведения*.

В конце цикла сразу после строки *КонецЕсли* и перед строкой *КонецЦикла* добавим строки кода, создающие движения регистра *Продажи*, производимые документом *ОказаниеУслуги*:

```
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.Себестоимость =
ТекСтрокаПереченьНоменклатуры.Стоимость *
ТекСтрокаПереченьНоменклатуры.Количество;
```

Перед началом цикла установим свойство *Записывать* набора записей движений по этому регистру в значение *Истина*:

```
Движения.Продажи.Записывать = Истина;
```

Запустим 1С:Предприятие в режиме отладки и перепроведем все документы *ОказаниеУслуги*. Теперь у нас есть практически вся необходимая информация для анализа деятельности ООО «Компьютер-сервис».

Лабораторные работы № 7. Запросы.

Цель работы: Знакомство с объектом конфигурации Отчёты, его основными свойствами. Научиться создавать объекты конфигурации Отчеты.

Порядок выполнения работы:

Настало время, чтобы познакомиться с одним мощным инструментом платформы 1С:Предприятие – *системой компоновки данных*.

В этой главе мы рассмотрим построение нескольких отчетов, которые будут использоваться в нашей конфигурации, и на их примере объясним основные возможности системы компоновки данных.

Любой отчет, как правило, подразумевает получение сложной выборки данных, сгруппированных и отсортированных определенным образом. Система компоновки данных представляет собой мощный и гибкий механизм, позволяющий выполнить все необходимые действия – от

получения данных из различных источников, до представления этих данных в виде, удобном для пользователя.

Чаще всего исходные данные, необходимые для отчета, находятся в базе данных. Для того, чтобы указать системе компоновки данных, какая информация и откуда должна быть получена, используется язык запросов системы 1С:Предприятие 8.

На этапе разработки отчета можно задать настройки отчета по умолчанию для того, чтобы пользователь мог сразу же запустить отчет на выполнение.

В то же время пользователь может самостоятельно изменить настройки отчета и выполнить его. При этом система компоновки данных сгенерирует другой запрос, и другим образом представит конечные данные – в соответствии с новыми настройками, заданными пользователем.

Для работы с запросами используется объект встроенного языка – *Запрос*. Он позволяет получать информацию, хранящуюся в полях базы данных, в виде выборки, сформированной по заданным правилам.

Исходную информацию *запрос* получает из набора *таблиц*. Эти таблицы представляют данные реальных таблиц базы данных в удобном для анализа виде. Их можно разделить на две большие группы: *реальные* и *виртуальные*.

Реальные таблицы, в свою очередь, могут быть *объектными* (ссылочными) или *не объектными* (не ссылочными).

Отличительной особенностью *реальных таблиц* является то, что они содержат данные какой-либо одной реальной таблицы, хранящейся в базе данных. Например, реальной является таблица *Справочник.Клиенты*, соответствующая справочнику Клиенты, или таблица *РегистрНакопления.ОстаткиМатериалов*, соответствующая регистру накопления *Остатки Материалов*.

Виртуальные таблицы формируются в основном из данных нескольких таблиц базы данных. Например, виртуальной является таблица *РегистрНакопления.ОстаткиМатериалов.ОстаткиИОбороты*,

формируемая из нескольких таблиц регистра накопления *ОстаткиМатериалов*.

Иногда виртуальные таблицы могут формироваться и из одной реальной таблицы (например, виртуальная таблица *Цены.СрезПоследних* формируется на основе таблицы *регистра сведений Цены*). Однако общим для всех виртуальных таблиц является то, что им можно задать ряд параметров, определяющих какие данные, будут включены в эти виртуальные таблицы. Набор таких параметров может быть различным для разных виртуальных таблиц, и определяется данными, хранящимися в исходных таблицах базы данных.

Алгоритм, по которому данные будут выбраны из исходных таблиц запроса, описывается в тексте запроса на специальном языке – *языке запросов*. Текст запроса состоит из нескольких частей:

- описание запроса;
- объединение запросов;
- упорядочивание результатов;
- автоупорядочивание;
- описание итогов.

Обязательной частью запроса является только первая – *описание запроса*. Все остальные присутствуют по необходимости.

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

Объединение запросов определяет, как будут объединены результаты выполнения нескольких запросов.

Упорядочивание результатов определяет условия упорядочивания строк результата запроса.

Автоупорядочивание позволяет включить режим автоматического упорядочивания строк результата запроса.

Описание итогов определяет, какие итоги необходимо рассчитывать в запросе и каким образом группировать результат.

Следует заметить, что в случае, когда язык запросов используется для описания источников данных в системе компоновки данных, секция описания итогов языка запросов не используется. Это связано с тем, что система компоновки данных самостоятельно рассчитывает итоги на основании тех настроек, которые сделаны разработчиком или пользователем.

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

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

Разработчик создает схему компоновки данных, в которой описывает текст запроса, наборы данных, связи между ними, доступные поля, параметры получения данных и задает первоначальные настройки компоновки – структуру отчета, макет оформления данных и др.

Алгоритм взаимодействия выглядит следующим образом: разработчик создает схему компоновки данных и настройки по умолчанию. В общем случае на основе одной схемы компоновки данных может быть создано большое количество различных отчетов. *Настройки компоновки данных* – создаваемые разработчиком или изменяемые пользователем – определяют, какой именно отчет будет получен в конкретном случае.

На основе схемы компоновки и имеющихся настроек компоновщик макета создает макет компоновки данных. Это этап подготовки к исполнению отчета. Макет компоновки данных является уже готовым заданием для выполнения процессором компоновки. Макет компоновки содержит необходимые запросы, макеты областей отчета и др.

Процессор компоновки данных выбирает данные из информационной базы согласно макету компоновки, агрегирует и оформляет эти данные. Результат компоновки обрабатывается процессором вывода, и в итоге пользователь получает результирующий табличный документ.

Итак, создадим отчет *Реестр документов Оказание услуги*, используя систему компоновки данных. Этот отчет будет выводить список существующих в базе данных документов *ОказаниеУслуги* в порядке их дат и номеров.

Создадим в конфигураторе новый объект конфигурации *Отчет*. Присвоим ему имя *РеестрДокументовОказаниеУслуги*. Установим свойство *Расширенное представление* как *Список оказанных услуг*.

На закладке *Основные* нажмем кнопку *Открыть схему компоновки данных*. В открывшемся диалоговом окне конструктора макета нажмем *Готово*. В конструкторе схемы компоновки данных создадим *Набор данных – запрос* и запустим конструктор запроса.

В качестве источника данных для запроса выберем объектную (ссылочную) таблицу документов *ОказаниеУслуги*. Из этой таблицы выберем следующие поля:

- Склад;
- Мастер;
- Клиент;
- Ссылка.

Перейдем на закладку *Объединения/Псевдонимы* и укажем, что поле *Ссылка* будет иметь псевдоним *Документ*. После этого перейдем на закладку *Порядок* и укажем, что результат запроса должен быть сначала упорядочен по значению поля *Документ*. Нажмем *ОК* и посмотрим, какой запрос сформировал конструктор запроса:

ВЫБРАТЬ

ОказаниеУслуги.Склад,
ОказаниеУслуги.Мастер,
ОказаниеУслуги.Клиент,
ОказаниеУслуги.Ссылка КАК Документ

ИЗ

Документ.ОказаниеУслуги КАК ОказаниеУслуги

УПОРЯДОЧИТЬ ПО

Документ

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

После ключевого слова *ИЗ* указываются источники данных – исходные таблицы запроса, содержимое которых обрабатывается в запросе. В данном случае это объектная (ссылочная) таблица *Документ.ОказаниеУслуги*. После ключевого слова *КАК* указывается псевдоним источника данных. В нашем случае это *ОказаниеУслуги*. В дальнейшем к этому источнику данных можно будет обращаться в тексте запроса, используя псевдоним.

Поля выборки также могут иметь псевдонимы, по которым в дальнейшем в тексте запроса можно обращаться к этому полю. В нашем случае это псевдоним *Документ*.

После части описания запроса в нашем примере следует часть упорядочивания результатов.

Предложение «*УПОРЯДОЧИТЬ ПО*» позволяет сортировать строки в результате запроса. После этого ключевого предложения располагается выражение упорядочивания, которое в общем случае представляет собой перечисление полей (выражений) и порядка вывода. В нашем случае упорядочивание будет выполняться сначала по полю *Дата*, а потом по полю *ОказаниеУслуги.Ссылка*. В обоих случаях порядок сортировки будет по возрастанию (настроен по умолчанию). На этом закончим изучение текста запроса и перейдем к настройке схемы компоновки данных.

Перейдем на закладку *Настройки* и добавим новую группировку в структуру отчета.

В окне выбора поля группировки просто нажмем *ОК* (тем самым мы указываем, что в группировке будут выводиться детальные записи из информационной базы) и на закладке *Выбранные поля* зададим поля, которые будут выводиться в отчет:

- Документ;
- Склад;
- Мастер;
- Клиент.

На этом создание отчета закончено. В заключение определим, в каких подсистемах будет отображаться наш отчет. Закроем конструктор схемы компоновки данных и в окне редактирования отчета перейдем на закладку *Подсистемы*. Отметим в списке подсистему *Оказание услуг*.

Теперь запустим 1С:Предприятие в режиме отладки и откроем отчет *Реестр документов Оказание услуги*. Нажмем *Сформировать* и посмотрим на результат работы нашего отчета. Мы видим, что отчет содержит реестр документов *Оказание услуги*. Причем двойным щелчком мыши на поле документ мы можем открыть исходный документ, а также выполнить другие действия «расшифровки», которые представляет нам система компоновки данных.

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

Далее создадим *отчет Рейтинг* услуг, который будет содержать информацию о том, выполнение каких услуг принесло ООО «Компьютер-сервис» наибольшую прибыль в указанном периоде. На примере *отчета Рейтинг* услуг мы проиллюстрируем, как отбирать данные в некотором периоде, как задавать параметры запроса и как использовать в запросе данные из нескольких таблиц и включать в результат запроса все данные одного из источников.

Создадим новый объект конфигурации Отчет. Назовем его *РейтингУслуг*. На закладке *Основные* нажмем кнопку *Открыть схему компоновки данных*. В открывшемся диалоговом окне конструктора макета нажмем *Готово*.

В конструкторе схемы компоновки данных создадим *Набор данных* – запрос и запустим конструктор запроса.

Выберем *объектную (ссылочную) таблицу* справочника *Номенклатура* и *виртуальную таблицу* регистра накопления *Продажи.Обороты*. Для того чтобы исключить неоднозначность имен в запросе, переименуем таблицу *Номенклатура* в *СпрНоменклатура* (контекстное меню правой кнопки мыши – *Переименовать таблицу*).

Затем выберем из таблиц поля *СпрНоменклатура.Ссылка* и *ПродажиОбороты.ВыручкаОборот*.

Перейдем на закладку *Связи* и увидим, что конструктор уже создал связь между двумя выбранными таблицами – значение изменения *регистра Номенклатура* должно быть равно ссылке на элемент *справочника Номенклатура*.

Единственное, что нам останется сделать, это сбросить *флаг Все* у таблицы регистра и установить его у таблицы справочника.

Установка *флага Все* у таблицы справочника будет означать, что из справочника будут выбраны все элементы и этим элементам будет поставлено в соответствие значение оборота выручки из регистра. Таким образом, в результате запроса будут присутствовать все услуги, и для некоторых из них будут указаны обороты выручки. Для тех услуг, которые не оказывались в выбранном периоде, не будет указано ничего.

Перейдем на закладку *Условия* и зададим условия выбора элементов из справочника *Номенклатура*. При задании условий выбора мы будем использовать параметры запроса. Первым условием должно быть то, что выбранный элемент не является группой (для этого следует переключиться в режим *Произвольное условие* – установить флаг *Произвольное*). Затем ввести в поле условие следующий текст:

СпрНоменклатура.ЭтоГруппа = ЛОЖЬ

Вторым условием должно быть то, что выбранный элемент является услугой.

спрНоменклатура.Услуга = ИСТИНА

Перейдем на закладку *Объединения/Псевдонимы* и укажем, что представление элемента справочника будет иметь псевдоним *Услуга*, а поле регистра будет иметь псевдоним *Выручка*.

Перейдем на закладку *Порядок* и укажем, что результат запроса должен быть отсортирован по убыванию значения поля *Выручка*. Создание запроса закончено, нажмем кнопку *ОК*. 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.Услуга = ИСТИНА
УПОРЯДОЧИТЬ ПО
Выручка УБЫВ

В нашем отчете мы хотим видеть итоговые значения выручки для каждой услуги. Для этого нам нужно определить поля ресурсов отчета. Перейдем на закладку *Ресурсы* и нажмем кнопку *>>*, чтобы конструктор выбрал все доступные ресурсы, по которым можно вычислять итоги. В нашем случае – это единственный ресурс *Выручка*.

Перейдем на закладку *Параметры*. На этой закладке мы увидим два параметра: *НачалоПериода*, *КонецПериода* и *ВидНоменклатуры*.

Система компоновки данных самостоятельно анализирует текст запроса и предоставляет возможность настроить те параметры, которые с большой долей вероятности могут быть использованы в отчете. Такими параметрами являются *НачалоПериода* и *КонецПериода*. Это первые два параметра виртуальной таблицы *РегистрНакопления.Продажи.Обороты*, которую мы использовали в запросе, в левом соединении.

Первым параметром передается начало периода расчета итогов, вторым – конец периода. В результате исходная таблица будет содержать только обороты, рассчитанные в переданном периоде.

Для параметра *НачалоПериода* зададим заголовок, который будет отображаться пользователю – *Дата начала*.

Здесь всегда следует помнить, что если мы передаем в качестве этих параметров дату (а в нашем случае так и будет), то дата содержит и время с точностью до секунды.

Допустим, заранее известно, что пользователя не будут интересовать результаты работы отчета в периодах, указанных с точностью до секунды. В этом случае следует учесть две особенности.

Во-первых, пользователя надо избавить от необходимости указывать время при вводе даты. Для этого мы изменим существующее описание типа для параметра *НачалоПериода*. Дважды щелкнем в ячейке *Доступные типы*, соответствующей параметру *НачалоПериода*, нажмем кнопку выбора (...) и в нижней части окна редактирования типа данных установим *Состав даты в значение Дата*.

Нажмем *ОК*.

Вторая особенность заключается в том, что по умолчанию время в дате установлено 00:00:00. Поэтому, если пользователь задаст период отчета с 01.03.2004 по 31.03.2004, итоги регистра будут рассчитаны с начала дня 01.03.2004 00:00:00 по начало дня 31.03.2004 00:00:00. Таким образом, данные за 31 число, отличные от начала дня, в расчет не войдут, что сильно удивит пользователя.

Для того чтобы исключить эту ситуацию, мы добавим еще один параметр, в который пользователь будет вводить дату окончания. *Значение* параметра *КонецПериода* будем рассчитывать автоматически таким образом, чтобы оно указывало на конец дня даты, введенной пользователем. Поэтому для параметра *КонецПериода* установим флаг *Ограничение доступности*.

С помощью кнопки командной панели добавим новый параметр с именем *ДатаОкончания*.

Для этого параметра платформа автоматически сформирует заголовок – *Дата окончания*. Оставим его без изменений. Зададим тип значения параметра – *Дата*. При этом, как и для параметра *НачалоПериода*, укажем состав даты – *Дата*.

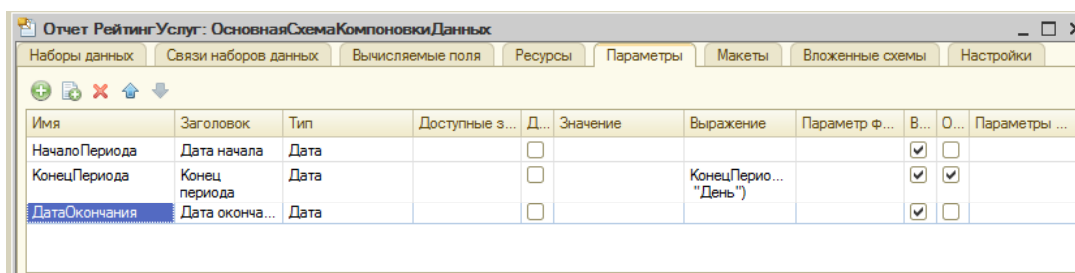


Рисунок 9. Настройка параметров отчета.

Обратите внимание, что по умолчанию добавленный нами параметр доступен для пользователя (ограничение доступности в последней колонке снято). Нас это вполне устраивает.

Перейдем к параметру *КонецПериода*. Для него система установила флаг ограничения доступности. Это нас также устраивает, поскольку значение этого параметра мы собираемся вычислять на основании значения, установленного пользователем для параметра *ДатаОкончания*.

Чтобы задать формулу, по которой будет вычисляться значение параметра *КонецПериода*, воспользуемся языком выражений системы компоновки данных. Он содержит функцию *КонецПериода()*, которая позволяет получить дату, соответствующую концу какого-либо периода, например, указанного дня.

В ячейке *Выражение* зададим для параметра *КонецПериода* следующее выражение:

КонецПериода(&ДатаОкончания, "День")

Перейдем к формированию структуры отчета. На закладке *Настройки* добавим группировку и снова не укажем поле группировки. На закладке *Выбранные поля* укажем поля *Услуга* и *Выручка*.

Затем перейдем на закладку *Другие* настройки и зададим заголовок отчета – *РейтингУслуг*.

В заключение мы должны предоставить пользователю возможность задавать отчетный период перед формированием отчета. То есть параметры *Дата начала* и *Дата окончания* должны быть включены в состав пользовательских настроек и находиться непосредственно в форме отчета.

Для этого на закладке *Параметры* выделим по очереди каждый из параметров и нажмем кнопку *Свойства элемента* пользовательских настроек. Установим флажок *Включать* в пользовательские настройки и оставим предложенное по умолчанию для свойства *Режим редактирования* значение *Быстрый доступ*.

Кроме того, чтобы улучшить интерфейс пользователя, зададим для параметров *Дата начала* и *Дата окончания* в качестве начальных значений соответственно Начало этого месяца и Начало этого дня.

Закроем конструктор схемы компоновки данных и окне редактирования отчета перейдем на закладку *Подсистемы*. Отметим в списке подсистемы *Оказание услуг* и *Бухгалтерия*. Запустим 1С: Предприятие в режиме отладки и убедимся в работоспособности отчета.

ЗАДАНИЕ:

1. Разработать отчет *Выручка мастеров*, который будет содержать информацию о том, какая выручка была получена ООО «Компьютер-сервис» благодаря работе каждого из мастеров в выбранном периоде и с разворотом по клиентам.
2. Разработать отчет *Перечень услуг*, который будет содержать информацию о том, какие услуги и по какой цене оказывает ООО «Компьютер-сервис».
3. Разработать отчет *Рейтинг клиентов*, который будет показывать в графическом виде, каков доход от оказания услуг каждому из клиентов за все время работы ООО «Компьютер-сервис».

Лабораторная работа № 8. Планы видов характеристик.

Цель работы: Знакомство с объектом конфигурации Планы видов характеристик, его основными свойствами. Научиться создавать и работать с объектом конфигурации Планы видов характеристик.

Порядок выполнения работы:

Задача, которую мы перед собой поставим, будет заключаться в следующем: мы создадим механизм, позволяющий пользователю произвольным образом описывать материалы и, что самое главное, вести учет в разрезе всех тех описаний, которые могут быть заданы пользователем.

Описывать материалы пользователь сможет следующим образом: для каждого материала будет возможность создать некоторые (произвольные) характеристики этого материала (например, цвет, производитель и пр.). Затем, при поступлении материалов, можно будет задать конкретные значения интересующих характеристик (например, при поступлении DVD-приводов можно будет указать, что они белого цвета и их интерфейс – SATA, а при поступлении лазерных принтеров указать, что они черного цвета и произведены на фирме «Canon»).

В дальнейшем всегда можно будет получить информацию о том, сколько и каких материалов есть у нас, скажем, белого цвета, или сколько было израсходовано черных картриджей.

Поскольку заранее не известно, какими именно характеристиками пользователь захочет описать тот или иной материал, мы должны предоставить ему некоторый механизм, позволяющий создавать любые характеристики и, что самое важное, указывать, какой тип значения должен быть у этих характеристик. Тогда при задании значений определенной характеристики пользователь сможет выбирать значения строго в соответствии с указанным типом.

Такую возможность описания характеристик как раз и обеспечивает объект конфигурации *План видов характеристик*, с которым мы сейчас познакомимся.

Объект конфигурации **План видов характеристик** является прикладным и предназначен для описания структуры хранения информации о характеристиках, создаваемых пользователем. На основе объекта конфигурации План видов характеристик платформа создает в базе данных информационную структуру. В ней может храниться информация существующих видах характеристик и типе значения характеристики каждого вида.

В сущности, план видов характеристик очень напоминает справочник, однако имеет более узкую «специализацию»: хранит, по сути, информацию только о том, какими видами характеристик может описываться какой-либо объект базы данных. Эта информация состоит из названия вида характеристики и ее типа. Разработчик и, что самое важное, пользователь могут задать в нем любое необходимое им количество видов характеристик.

Для того чтобы разработчик мог задать некий «набор» возможных типов значений, которые могут принимать виды характеристик, у объекта конфигурации План видов характеристик существует свойство Тип значения характеристик. Это свойство определяет составной тип данных, куда входят все типы, которые могут понадобиться при указании типа значения характеристики.

Кроме этого может случиться так, что пользователю станет недостаточно тех типов данных, которые существуют в конкретной системе (например, он захочет вести учет в разрезе цвета товаров, а справочник Цвет в конфигурации отсутствует). В этом случае он сможет воспользоваться неким вспомогательным справочником, который разработчик создаст заблаговременно и укажет в качестве свойства объекта конфигурации *План видов характеристик* – дополнительные значения характеристик.

Тогда пользователь, создав новый вид характеристики Цвет, сможет дать необходимые значения в справочнике дополнительных значений характеристик. Примечательно, что этот справочник является подчиненным плану видов характеристик. Таким образом, если затем пользователь пожелает создать новый вид характеристик Запах и его значения, он будет создавать их в том же самом справочнике дополнительных характеристик, и они не будут «смешиваться» со значениями цвета.

План видов характеристик не имеет внутренних predetermined механизмов привязки вида характеристики к тому объекту, который он должен описывать. Он лишь предоставляет возможность разработчику и пользователю описать некий набор характеристик и задать их тип. Каким образом хранить соответствие конкретного вида характеристик или значения характеристик конкретному объекту базы данных, решает сам разработчик в зависимости от создаваемого прикладного решения.

С точки зрения реализации пример, который мы будем рассматривать далее, является сложнее остальных, поэтому сначала мы объясним логическую связь между объектами, которые будут использоваться в этом примере.

Для реализации этого примера нам понадобятся три новых объекта конфигурации. Прежде всего, это *План видов характеристик*. Он будет хранить виды характеристик, которыми в принципе можно описывать материалы. Затем это *Справочник*, подчиненный справочнику Номенклатура, элементы которого будут идентифицировать партии материалов с некоторым фиксированным набором значений характеристик. И третий объект – это *Регистр сведений*, в котором собственно и будет храниться соответствие конкретных значений характеристик некоторому варианту материала.

В результате использования подобной логической структуры объектов мы получим возможность описывать каждую поступающую партию материала любым количеством видов характеристик, поскольку это соответствие будет храниться в регистре сведений. И вместе с тем получим

возможность вести учет в разрезе видов характеристик, добавив в регистры накопления еще одно измерение для хранения ссылки на элемент справочника, подчиненного справочнику *Номенклатура*.

Теперь для того, чтобы узнать остатки материалов, обладающих некоторым значением характеристики, достаточно будет выбрать из регистра сведений все элементы подчиненного справочника, обладающие таким значением характеристики, и затем по ним и их владельцам получить остатки регистра накопления.

Как мы уже говорили, нам понадобится создать несколько новых объектов конфигурации:

- Справочник *ВариантыНоменклатуры*, чтобы описывать партии материалов;
- Справочник *ДополнительныеСвойстваНоменклатуры*, чтобы задавать значения видов характеристик, для которых нет подходящих типов в конфигурации;
- План видов характеристик *СвойстваНоменклатуры*, чтобы создавать виды характеристик;
- Регистр сведений *ЗначенияСвойствНоменклатуры*, чтобы хранить значения видов характеристик для различных «партий» материалов.

Создадим объект конфигурации Справочник с именем *ВариантыНоменклатуры* и укажем, что он будет подчинен справочнику *Номенклатура*.

Затем создадим еще один объект конфигурации Справочник с именем *ДополнительныеСвойстваНоменклатуры*.

После этого создадим объект конфигурации План видов характеристик с именем *СвойстваНоменклатуры*. Тип значения характеристик установим следующим:

- Число, длина 15, точность 3;
- Строка, длина 25;

- Дата;
- Булево;
- *СправочникСсылка.ДополнительныеСвойстваНоменклатуры*.

Справочнику *ДополнительныеСвойстваНоменклатуры* укажем владельца – план видов характеристик *СвойстваНоменклатуры*. После этого определим, что дополнительные значения характеристик плана видов характеристик будут располагаться в справочнике *ДополнительныеСвойстваНоменклатуры*.

В заключение создадим объект конфигурации *Регистр сведений* с именем *ЗначенияСвойствНоменклатуры*.

Измерения регистра:

- *НаборСвойств*, ведущее, тип *СправочникСсылка.ВариантыНоменклатуры*;
- *ВидСвойства*, тип *ПланВидовХарактеристикСсылка.СвойстваНоменклатуры*.
- *Ресурс регистра*:
- *Значение*, тип *Характеристика.СвойстваНоменклатуры*.

Обратите внимание, что мы имеем возможность определить тип значения ресурса регистра как *Характеристика.<имя>*. По сути, это определение представляет собой составной тип данных, как он задан в типе значения соответствующего плана видов характеристик. То есть ресурс регистра может иметь значение любого типа из тех, которые описаны в типе значения плана видов характеристик.

Кроме этого, зададим в свойстве *Связь* по типу этого ресурса измерение регистра *ВидСвойства*. Связь по типу будет обеспечивать нам соответствие типа значений, вводимых в это поле, и типа характеристики, выбранной в поле *Вид свойства*.

Заполним еще одно свойство – *Связи параметров выбора*. Для этого нажмем кнопку выбора (...) у этого свойства и перенесем из списка доступных реквизитов в список параметров измерение регистра

ВидСвойства. Установка этого свойства обеспечит нам то, что при выборе значений, содержащихся в справочнике *Дополнительные свойства номенклатуры*, для выбора будут предлагаться только те значения, которые относятся к выбранной характеристике, а не все, которые есть в этом справочнике.

Далее отредактируем справочник *Номенклатура*. На закладке *Данные* нажмем кнопку *Стандартные реквизиты*, в списке этих реквизитов дважды щелкнем на реквизите *Родитель* и в открывшейся палитре свойств зададим *Синоним реквизита – Группа номенклатуры*.

Далее для создания формы списка справочника *Номенклатура* откроем окно редактирования этого справочника и перейдем на закладку *Формы*, нажмем на кнопку открытия и создадим основную форму списка. В открывшемся окне конструктора нажмем *Готово*.

В открывшемся окне прокрутим вниз список свойств формы, найдем событие *ПриСозданииНаСервере* и нажмем кнопку открытия.

В модуле формы будет создан обработчик события формы *ПриСозданииНаСервере*, в который мы внесем следующий текст:

```
Если Параметры.Отбор.Свойство("Владелец") Тогда  
Элементы.Код.Видимость = Ложь;  
КонецЕсли;
```

Это позволит сделать колонку *Код* невидимой, что улучшит восприятие информации. Для этих же целей в окне редактирования регистра сведений *ЗначенияСвойствНоменклатуры* на закладке *Основные* зададим *Представление списка* как *Состав варианта номенклатуры*, а *Представление записи* – *Свойство и значение*.

ЗАДАНИЕ:

1. Создать различные варианты номенклатуры в режиме 1С: Предприятие.
2. Доработать документы *Приходная накладная* и *Оказание услуги* таким образом, чтобы в них отражались данные о свойствах номенклатуры.
3. Доработать регистр остатки материалов таким образом, чтобы можно

было получать данные в разрезе свойств номенклатуры.

4. Создать отчет *Остатки материалов* по свойствам, который будет показывать остатки материалов с теми или иными свойствами.

Лабораторная работа № 9. Планы видов расчета. Регистры расчета. Пользователи и роли. Рабочий стол.

На этом занятии мы рассмотрим, какие возможности для автоматизации сложных периодических расчетов предоставляет система 1С:Предприятие.

Такие расчеты используются прежде всего при расчете заработной платы. Поэтому дальнейшее их рассмотрение мы будем строить на примере расчета заработной платы сотрудников, которые работают в нашем ООО «Компьютер-сервис».

В общем случае сумма заработной платы сотрудника складывается из множества частей (например, оплата по окладу, премии, штрафы, оплаты по больничному листу, разовые выплаты и т. д.). Каждая из этих частей рассчитывается по некоторому алгоритму, присущему только этой части. Например, сумма штрафа может определяться просто фиксированной суммой, сумма премии может рассчитываться как процент от оклада, а сумма оплаты по окладу рассчитывается исходя из количества рабочих дней в месяце и количества дней, отработанных сотрудником. Поэтому для обозначения каждой такой части мы будем использовать термин вид расчета.

Алгоритм каждого вида расчета опирается в общем случае на две категории параметров: период, за который нужно получить конечные данные, и набор некоторых исходных данных, используемых при расчете.

Как правило, в реальной жизни различные виды расчета существуют не сами по себе, а оказывают некоторое влияние на другие виды расчета. Исходя из того, что вид расчета опирается на две различные категории параметров, такое влияние тоже имеет двойственный характер.

Во-первых, это может быть влияние на исходные данные, используемые при расчете. В качестве примера можно привести начисление премии в виде

процента от оплаты по окладу. При изменении оплаты по окладу размер премии тоже должен быть пересчитан, исходя из новой суммы начисленного оклада. Другими словами, сумма начисленного оклада является базой для расчета премии. Причем, поскольку оклад рассчитывается за некоторый период, при расчете премии нам интересно знать не значение оклада вообще, а сумму, начисленную в том периоде, который влияет на расчет премии. Такой период мы будем называть базовым, а подобную зависимость между видами расчета – зависимостью по базовому периоду.

В качестве примера рассмотрим расчет премии за апрель. Премия должна начисляться в размере 10% от суммы, начисленной в качестве оплаты по окладу. Следовательно, необходимо проанализировать все записи о начислениях оплаты по окладу, которые попадают в интересующий нас базовый период, а именно апрель. Допустим, общая сумма таких начислений составила 8000 рублей – в этом случае премия должна быть начислена в размере 800 рублей.

Во-вторых, это влияние может быть не на исходные данные, а на сам период, за который производится расчет. В качестве примера можно привести расчет оплаты по окладу и невыход на работу. Предположим, мы начислили сотруднику оплату по окладу за март месяц. В этом случае период действия такого расчета будет с 01.03.2004 по 31.03.2004. После этого мы получили информацию от руководителя отдела, что, оказывается, сотрудник отсутствовал на работе с 1 по 10 марта по неизвестной причине. В этом случае нам нужно будет произвести расчет Невыход (в котором можно рассчитать какие-то удержания с сотрудника). Но кроме этого необходимо будет пересчитать и оклад сотрудника, исходя из того, что фактический период действия расчета Оклад стал теперь с 11.03.2004 по 31.03.2004. Такое влияние мы будем называть вытеснением по периоду действия. В результате если за весь месяц работы сотруднику должно было быть начислено 9300 рублей, то теперь, за фактический период работы, начисление составит 6300 рублей.

Таким образом, исходя из двух видов взаимного влияния расчетов, можно сказать, что в общем случае с каждым видом расчета будет связано три периода: период действия, фактический период и базовый период.

Период действия является «запрашиваемым». То есть, указывая период действия, мы говорим: «Мы хотели бы, чтобы результат действовал в этом периоде».

Фактический период – это то, что получилось после анализа всех периодов действия расчетов, которые вытесняют наш по периоду действия.

Базовый период – это период, в котором мы анализируем результаты других расчетов, влияющих на наш по базовому периоду.

Как видите, взаимное влияние между видами расчетов может быть довольно разнообразным и, что самое сложное, многоуровневым. То есть один вид расчета может влиять на другой, который, в свою очередь, влияет на третий, и т. д.

Очевидно, что в этой ситуации требуется некий универсальный механизм, позволяющий описать каждый из видов расчетов (его алгоритм, влияние на другие виды расчетов, зависимость от других видов расчетов), обеспечить хранение данных, полученных в результате этих расчетов, и контроль необходимости перерасчета результатов зависимых расчетов в случае изменения результатов «первичных» расчетов.

В системе 1С:Предприятие такой универсальный механизм реализован при помощи планов видов расчета и регистров расчета. И первым объектом конфигурации, с которым мы начнем знакомиться, будет План видов расчета.

Объект конфигурации План видов расчета является прикладным и предназначен для описания структуры хранения информации о возможных видах расчетов. На основе объекта конфигурации План видов расчета платформа создает в базе данных информационную структуру, в которой может храниться информация о том, какие существуют виды расчета и каковы взаимосвязи между ними.

Отличительной особенностью плана видов расчета является то, что пользователь в процессе работы может добавлять новые виды расчета. Такая возможность делает механизм периодических расчетов более гибким и позволяет пользователю создавать собственные виды расчета, помимо тех, которые заданы разработчиком как предопределенные.

Объект конфигурации План видов расчета имеет свойство Использует период действия. С его помощью определяется, будут ли в этом плане находиться виды расчета, которые могут быть вытеснены по периоду действия. Если это свойство установлено, то разработчик получает возможность указать для каждого вида расчета те виды, которые вытесняют его по периоду действия.

Следующим важным свойством объекта конфигурации План видов расчета является Зависимость от базы. Оно определяет, будут ли в этом плане находиться зависимые по базовому периоду виды расчета. Если это свойство установлено, появляется возможность указать, в каком плане видов расчета будут находиться базовые виды расчета и, кроме этого, как будет определяться эта зависимость.

Существует возможность указать один из двух видов зависимости от базы: Зависимость по периоду действия и Зависимость по периоду регистрации.

Еще одной важной особенностью плана видов расчета является возможность создания предопределенных видов расчета и описания их взаимного влияния. При этом в общем случае разработчик имеет возможность указать три категории видов расчета, влияющих на предопределенный вид расчета:

Базовые – их результаты должны быть использованы при перерасчете этого вида расчета;

Вытесняющие –вытесняют этот вид расчета по периоду действия;

Ведущие –изменение их результатов должно приводить к необходимости перерасчета этого вида расчета.

Здравый смысл подсказывает, что все базовые виды расчета должны быть включены и в категорию ведущих. Кроме того, ведущие виды расчета могут содержать и некоторые другие виды, косвенно влияющие на данный вид расчета. Например, мы имеем три вида расчета: невыход, оклад и премия. Невыход вытесняет оклад по периоду действия, а премия зависит от оклада по базовому периоду.

В этом случае для премии следует указать базовым видом расчета оклад, а ведущими – оклад и невыход, поскольку изменение результата расчета невыхода приведет к изменению результата оклада, что, в свою очередь, должно привести к изменению результата премии.



Приступим теперь к созданию плана видов расчета Основные начисления, который будет использоваться в нашей конфигурации.

Откроем конфигуратор и создадим новый объект конфигурации План видов расчета. Зададим его имя – ОсновныеНачисления.

На закладке Подсистемы укажем, что план видов расчета будет отображаться в подсистеме Расчет зарплаты.

Затем перейдем на закладку Расчет и укажем, что он будет использовать период действия и зависеть от базы по периоду действия.

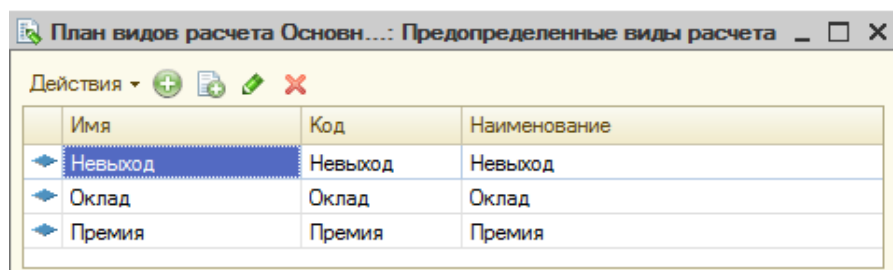
В качестве базового плана видов расчета укажем его самого, поскольку все наши виды расчетов будут храниться в единственном плане видов расчета.

Перейдем на закладку Прочее и зададим predetermined виды расчета. Как и в случае с бухгалтерией, расчеты в нашем ООО «Компьютер-Сервис» будут «скромные», поэтому мы создадим всего три элемента:

Невыход – с именем и наименованием Невыход и кодом Невыход;

Оклад – с именем, кодом и наименованием Оклад и вытесняющим его видом расчета Невыход;

Премия – с именем, кодом и наименованием Премия, с базовым видом расчета Оклад и ведущими видами расчета Невыход и Оклад.



Имя	Код	Наименование
Невыход	Невыход	Невыход
Оклад	Оклад	Оклад
Премия	Премия	Премия

Теперь мы перейдем к рассмотрению второго объекта, используемого при реализации механизмов сложных периодических расчетов, – регистра расчета.

Объект конфигурации Регистр расчета является прикладным и предназначен для описания структуры накопления данных, являющихся результатами расчетов. На основе объекта конфигурации Регистр расчета платформа создает в базе данных информационную структуру, в которой будут накапливаться данные, формируемые различными объектами базы данных.

Отличительной особенностью регистра расчета является то, что он не предназначен для интерактивного редактирования пользователем. Разработчик может при необходимости предоставить пользователю возможность редактировать регистр расчета, но предназначение регистра расчета заключается в том, чтобы его модификация производилась на основе алгоритмов работы объектов базы данных, а не в результате непосредственных действий пользователя.

Как и другие регистры, регистр расчета имеет ресурсы, в которых хранит числовые данные; имеет измерения, в разрезе которых можно получать значения ресурсов регистра; имеет реквизиты, которые характеризуют каждую запись регистра расчета.

Отличительными же особенностями регистра расчета является его периодичность, возможность использования механизмов вытеснения по периоду действия и зависимости по базовому периоду, а также связь с планом видов расчета. Рассмотрим все эти особенности по порядку.

Периодичность регистра расчета может быть определена одним из следующих значений:

День;

Месяц;

Квартал;

Год.

Периодичность регистра расчета определяет промежуток времени, к которому будет относиться каждая запись регистра. Если указана периодичность День, то каждая запись регистра будет относиться к какому-либо дню; если периодичность Месяц, то к какому-либо месяцу и т. д. Для указания факта принадлежности записи к какому-либо периоду регистр имеет служебный реквизит ПериодРегистрации типа Дата. При записи данных в регистр платформа всегда приводит значение этого реквизита к началу того периода, в который он попадает.

Например, если в регистр расчета с периодичностью месяц записать данные, где ПериодРегистрации задан как 08.14.2004, то регистр сохранит эти данные со значением поля ПериодРегистрации 01.04.2004.



Если в этой же ситуации периодичность регистра будет год, сохраненное значение периода регистрации будет 01.01.2004.

The screenshot displays a software window titled "Документ" (Document). Inside, there is a "Дата:" (Date) field with the value "08.04.2004". Below it is a "Модуль документа" (Document module) section containing the text "Движение.ПериодРегистрации = Дата;" and "Движения. Записать (!);". An orange arrow points from the "Дата:" field to a table below. The table is titled "Таблица регистра расчета (периодичность регистра – «Год»)" (Calculation register table (register periodicity – «Year»)). The table has four columns: "Регистратор" (Registrar), "Период регистрации" (Registration period), "Номер строки" (Row number), and an empty column. The first three rows are highlighted in orange and contain the date "01.01.2004 00:00:00". The first row has "1" in the "Номер строки" column, and the third row has "N".

Регистратор	Период регистрации	Номер строки	
	01.01.2004 00:00:00	1	
	01.01.2004 00:00:00		
	01.01.2004 00:00:00	N	

Следующей важной особенностью регистра расчета является возможность использования механизма вытеснения одних записей другими по периоду действия. При этом для каждой записи регистр расчета формирует фактический период действия, который является в общем случае совокупностью нескольких периодов, расположенных внутри периода действия.

Другим механизмом, который поддерживает регистр расчета, является зависимость записей по базовому периоду. Этот механизм позволяет основывать расчет зависимых (вторичных) записей регистра на данных, полученных в результате расчета первичных записей. Регистр расчета может поддерживать два вида зависимости от базы: зависимость по периоду действия и зависимость по периоду регистрации.

Зависимость по периоду действия означает, что при анализе базовых записей будут выбираться те, для которых найдено пересечение их фактического периода действия и указанного базового периода.

Зависимость по периоду регистрации означает, что при анализе базовых записей будут выбираться те, которые попадают в указанный базовый период значением своего поля Период регистрации.

Заключительной важной особенностью регистра расчета является его связь с планом видов расчета. Именно на основе этой связи работают механизмы вытеснения по периоду действия и зависимости по базовому периоду, поскольку в плане видов расчета описано взаимное влияние видов расчета друг на друга.

У регистра расчета могут существовать подчиненные объекты Перерасчет. Они предназначены для регистрации фактов появления в регистре записей, влияющих на результат расчета уже существующих записей регистра. Объект конфигурации Перерасчет может иметь несколько измерений, каждое из которых устанавливает связь между измерениями данного регистра расчета и влияющих регистров расчета. В частном случае это может быть один и тот же регистр.

В информационной структуре, созданной в базе данных на основе объекта конфигурации Перерасчет, платформа хранит информацию о том, какие записи регистра подлежат перерасчету. Таблицы перерасчета заполняются автоматически как на основании записей регистров расчета, затронутых ведущими видами расчета, так и на основании записей регистра расчета, для которых изменился фактический период действия. Исходя из этой информации, разработчик может принимать решение о необходимости перерасчета записей регистра.

Последним замечанием, которое следует сделать, говоря о регистре расчета, является возможность установки связи регистра расчета с графиком времени. Такой график времени должен представлять собой регистр сведений (непериодический, с обязательным измерением типа Дата и ресурсом типа Число), в котором содержится временная схема исходных данных, участвующих в расчетах. Измерениями этого графика могут быть, например, график работы (ссылка на справочник) и дата, а ресурсом – количество рабочих часов в этой дате. В этом случае можно будет связать запись регистра расчета с каким-либо конкретным графиком работы (указав в качестве реквизита записи ссылку на справочник график работы) и в

дальнейшем средствами встроенного языка получать информацию о количестве рабочих часов в периоде действия, фактическом периоде действия или периоде регистрации этой записи.

Прежде чем мы начнем создавать объект конфигурации Регистр расчета Начисления, нам потребуется создать два дополнительных объекта конфигурации – Регистр сведений ГрафикиРаботы и Справочник ВидыГрафиковРаботы. Справочник понадобится нам для того, чтобы хранить информацию о том, какие графики работы существуют в ООО «Компьютер-сервис», а регистр сведений – для указания того, какие дни в месяце являются рабочими, поскольку сумма оплаты по окладу будет рассчитываться исходя из того, сколько дней отработал сотрудник в расчетном месяце.

Откроем конфигуратор и создадим новый объект конфигурации Справочник с именем ВидыГрафиковРаботы.

В этом справочнике у нас будет два predetermined графика работы – ГрафикАдминистрации и ГрафикМастеров.

Запустим 1С:Предприятие в режиме отладки и заполним регистр данными о рабочих днях сентября графика мастеров. Чтобы проще выполнить эту довольно однообразную работу, воспользуйтесь возможностью добавления элементов в справочник копированием (Действия - Скопировать).

Теперь все готово для создания регистра расчета.

Создадим новый объект конфигурации Регистр расчета с именем Начисления. В качестве плана видов расчета, используемого регистром, выберем ОсновныеНачисления. Установим, что регистр будет использовать период действия, график будет задаваться в регистре сведений ГрафикиРаботы, значение графика будет находиться в ресурсе Значение, а дата графика – в измерении Дата.

Укажем, что регистр расчета будет использовать базовый период и периодичность регистра будет Месяц.

Затем перейдем на закладку Данные и создадим:

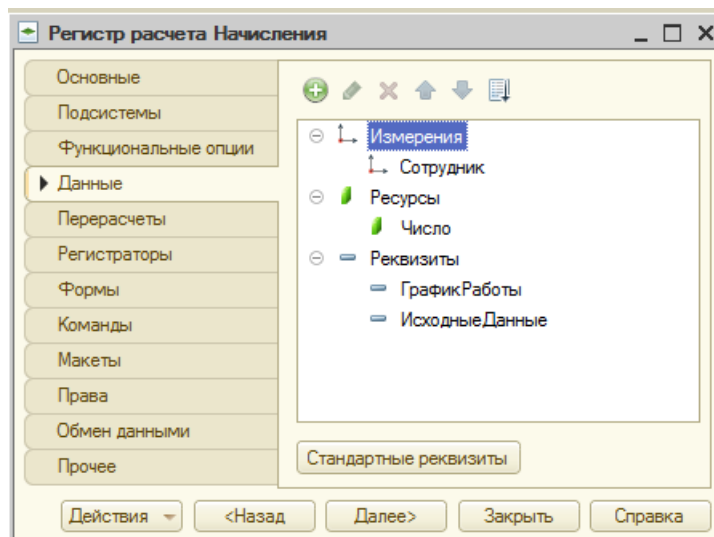
измерение Сотрудник, тип СправочникСсылка.Сотрудники, базовое;

ресурс Результат, тип Число, длина 15, точность 2;

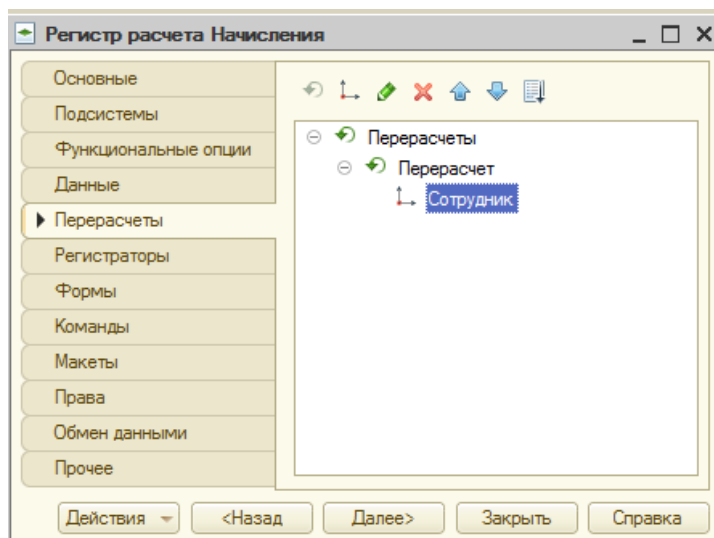
реквизит ГрафикРаботы, тип СправочникСсылка.ВидыГрафиковРаботы, связь с графиком по измерению ГрафикРаботы;

реквизит ИсходныеДанные, тип Число, длина 15, точность 2.

Реквизит ГрафикРаботы мы будем использовать для того, чтобы связать запись регистра с используемым графиком работы, а реквизит ИсходныеДанные – чтобы хранить в нем данные, которые могут понадобиться при расчете или перерасчете (в нашем примере это будет расчет оклада).



Теперь перейдем на закладку Перерасчеты. Создадим объект конфигурации Перерасчет, который так и назовем – Перерасчет. У него будет единственное измерение – Сотрудник, для которого в установке связи мы укажем измерение регистра Сотрудник и в качестве данных ведущих регистров выберем то же самое измерение Сотрудник регистра расчета Начисления.



На этом создание объекта конфигурации Регистр расчета Начисления завершено.

ЗАДАНИЕ:

1. Создать документ Начисления сотрудникам, отражающий факт начисления заработной платы, со следующими реквизитами: Сотрудник, График работы, Дата начала, Дата окончания, Вид расчета, Начислено (руб.).

2. В модуле документа Начисления сотрудникам описать механизмы, отражающие вытеснение по базовому периоду и по периоду действия различных видов расчета.
3. В режиме отладки ввести несколько документов по начислению заработной платы.
4. Создать отчет о начислениях сотрудникам.

После того, как созданы все основные объекты конфигурации, можно приступить к определению ролей пользователей.

До сих пор мы с вами имели полный доступ ко всем разделам приложения и ко всем объектам конфигурации и командам, используемым в этих разделах.

Однако при реальной работе пользователей одной из главных возможностей, которую должно обеспечивать прикладное решение, является разграничение прав доступа пользователей к той или иной информации, хранящейся в информационной базе.

Например, руководитель должен, очевидно, иметь доступ ко всей информации, которая содержится в базе данных, а вот кладовщик – напротив, должен иметь доступ только к информации, касающейся движения товаров на складах и не иметь возможности просматривать бухгалтерскую или кадровую информацию.

Кроме этого, должна существовать возможность ограничить пользователей в выполнении тех или иных действий с объектами базы данных. Например, кладовщик может создавать и изменять приходные накладные, поскольку он отвечает за учет материалов на предприятии. Мастеру может понадобиться просматривать приходные накладные для того, чтобы знать, какие материалы и когда были получены. Однако мастер не должен иметь возможности вносить какие-либо изменения в приходные накладные.

Для описания подобных разрешений используются объекты конфигурации Роль. С помощью такого объекта разработчик получает

возможность описать набор прав на выполнение тех или иных действий над каждым объектом базы данных и над всей конфигурацией в целом.

Как правило, роли создаются отдельно для каждого вида деятельности, и каждому пользователю системы ставится в соответствие одна или несколько ролей.

Если пользователю поставлено в соответствие несколько ролей, предоставление доступа будет осуществляться по следующему алгоритму:

если хотя бы в одной роли есть разрешение, то доступ будет открыт;

если во всех ролях разрешение отсутствует, то доступ будет закрыт.

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.

Первая роль, которую мы создадим, будет Администратор. Она должна включать в себя полные права на работу с данными информационной базы.

Раскроем ветвь Общие дерева объектов конфигурации. Выделим строку Роли и создадим новый объект конфигурации Роль с именем Администратор. Откроется окно редактирования прав.

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. Теперь единственное, что следует сделать, снять разрешение на интерактивное удаление для всех объектов. Это необходимо для того, чтобы администратор случайно не мог удалить какой-либо объект базы данных. Для того чтобы наш Администратор мог работать с объектами, которые мы будем создавать после расстановки прав, зададим для него параметр Устанавливать права для новых объектов.

Следующей ролью, которую мы создадим, будет роль Директор. Создадим новый объект конфигурации Роль с именем Директор. Нам

достаточно того, что у новой роли нет прав на доступ ко всем объектам, за исключением тех видов объектов конфигурации, для которых не создано ни одного объекта. Для таких видов объектов конфигурации останутся установленными полные права.

Установим право Вывод. Теперь нам останется лишь пройти по видам объектов конфигурации и установить для них права Чтение, Просмотр и Использование. Затем раскроем ветвь Общие, выделим ветвь Подсистемы и отметим право Просмотр у всех подсистем, кроме подсистемы Предприятие. Вторая роль нашей конфигурации готова.

Следующая роль, которую мы создадим, будет роль Мастер. Снова создадим новый объект конфигурации Роль с именем Мастер. Выполним команду Действия - Установить по подсистемам и выберем подсистемы УчетМатериалов и ОказаниеУслуг. В результате будут установлены все права на объекты конфигурации, относящиеся к данным подсистемам.

Если теперь установить фильтр объектов по подсистемам УчетМатериалов и ОказаниеУслуг, то можно при необходимости внести уточнения в установленные права.

В частности, для справочника Сотрудники мы запретим добавление, изменение и удаление. Обратите внимание, что при запрете права Добавление исчезла отметка и у права Интерактивное добавление, т. к. оно является «уточнением» права Добавление. Точно так же «уточненные» права запрещаются и при отмене прав на изменение и удаление.

Кроме этого мы снова снимем разрешения на интерактивное удаление для всех объектов базы данных.

Затем снимем фильтр и установим все права, кроме интерактивного удаления для следующих объектов конфигурации:

- Справочник Варианты номенклатуры.
- Справочник Дополнительные свойства номенклатуры.
- План видов характеристик Свойства номенклатуры
- Регистр сведений Значения свойств номенклатуры.

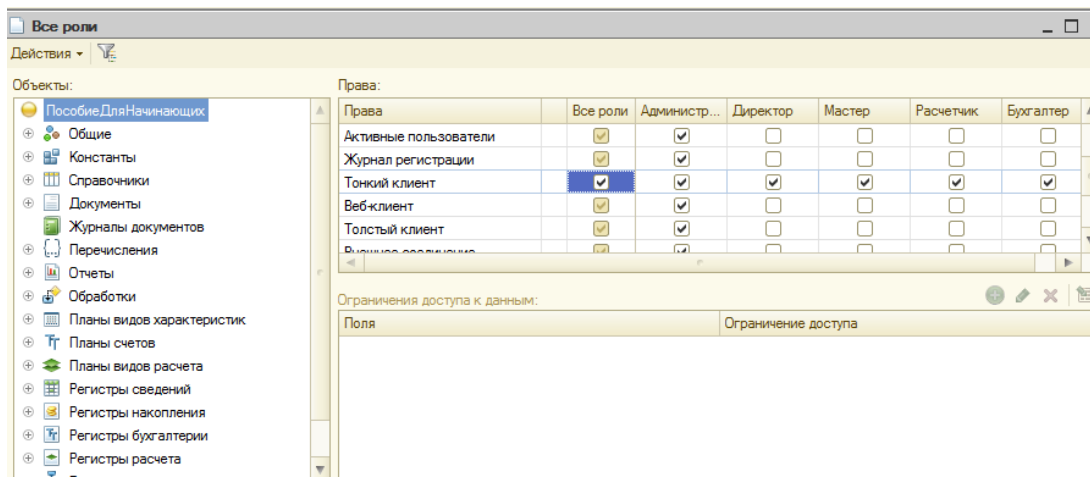
– Роль Мастер готова.

ЗАДАНИЕ:

1) Создать роль Расчетчик для сотрудника, который будет заниматься расчетом заработной платы.

2) Создать роль Бухгалтер для бухгалтера предприятия.

В заключение установим права на запуск клиентского приложения для каждой роли. Для этого воспользуемся 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.



Для того, чтобы иметь возможность отличать друг от друга пользователей, работающих с информационной базой, в системе 1С:Предприятие существует режим ведения списка пользователей.

В этом режиме администратор базы имеет возможность создавать и удалять пользователей системы, назначать им интерфейсы и роли и т. д.

Прежде чем мы приступим к созданию списка пользователей, необходимо выполнить обновление конфигурации базы данных, поскольку пользователю можно поставить в соответствие только тот интерфейс и те роли, которые существуют в конфигурации базы данных.

После того как обновление произведено, выполним команду Администрирование - Пользователи. Откроется список пользователей системы.

Пока что он пуст, поэтому добавим нового пользователя (Действия - Добавить).

Имя пользователя – это идентификатор, который будет появляться в окне выбора пользователей при запуске системы в режиме 1С:Предприятие.

Полное имя – строка, которая может быть использована внутри конфигурации при выводе различной справочной информации. Хорошим стилем администрирования считается указание в качестве полного имени фамилии, имени и отчества пользователя (без сокращений).

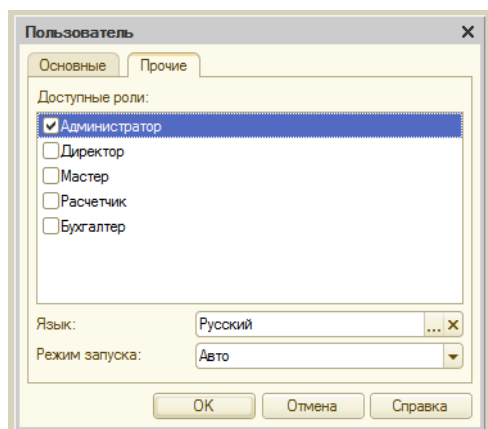
Следующие две области окна посвящены способам аутентификации пользователя.

Аутентификация средствами 1С:Предприятия подразумевает, что после запуска системы пользователю будет предложено выбрать имя одного из пользователей системы и ввести пароль. Если введенный пароль соответствует сохраненному в системе для этого идентификатора пользователя, система открывается с правами и интерфейсом, которые указаны для этого пользователя. При этом он сможет поменять пароль, если флаг Пользователю запрещено изменять пароль не установлен.

Аутентификация Операционной системы подразумевает, что при запуске системы 1С:Предприятие от пользователя не требуется никакой дополнительной информации. Система 1С:Предприятие определяет, под каким пользователем запущена операционная система, и затем обращается к своему списку пользователей. Если она находит в нем пользователя, которому поставлен в соответствие текущий пользователь ОС, система открывается с правами и интерфейсом, указанными для этого пользователя.

Зададим имя пользователя Администратор, полное имя тоже Администратор. Перейдем на закладку Прочие.

Отметим роль Администратор, основным интерфейсом укажем Администратор и язык конфигурации выберем Русский.



После этого создадим остальных пользователей системы. Для всех них мы будем использовать аутентификацию 1С: Предприятия и русский язык.

1	2	3	4	5
	Имя	Аутентификация 1С:Предприятия	Роли	Основной язык
2	Администратор	Истина	Администратор	Русский
3	Гусаков	Истина	Мастер	Русский
4	Деловой	Истина	Директор	Русский
5	Назарова	Истина	Расчетчик, Бухгалтер	Русский
6	Симонов	Истина	Мастер	Русский
7	Смирнова	Истина	Расчетчик	Русский

Рабочий стол предназначен для наиболее часто используемых пользователем документов, отчетов, справочников и т.п. Поэтому нужно поместить на рабочий стол пользователя те формы документов, отчетов и пр., работа с которыми входит в его ежедневные должностные обязанности.

Например, для кладовщика было бы удобно иметь под руками список номенклатуры и список приходных накладных, для менеджера – список клиентов и документов оказания услуг.

При запуске 1С: Предприятия раздел Рабочий стол становится активным по умолчанию и нужные формы сразу открываются в рабочей области приложения.

Таким образом, пользователю не нужно ходить по разделам, выполнять команды в панели действий или в панели навигации, а можно сразу начинать работать, предварительно включив компьютер и запустив 1С: предприятие.

Однако не стоит перегружать рабочий стол различными формами, иначе вместо удобства пользователь будет ощущать дискомфорт.

Итак, начнем настройку рабочего стола. Выделим корень дерева объектов конфигурации ПособиеДляНачинающих, вызовем его 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.

Это значит, что формы на рабочем столе будут располагаться в две колонки, при этом левая колонка будет в два раза шире правой.

Следует иметь в виду, что автоматически сгенерированные системой формы нельзя располагать на рабочем столе. Поэтому, прежде, чем добавлять форму на рабочий стол, нужно создать ее в явном виде в конфигурации.

Итак, начнем настройку рабочего стола для роли Мастер. Логично, если для мастеров в левой колонке рабочего стола будет располагаться список документов об оказании услуг, а в правой колонке – список приходных накладных и список клиентов.

Перечисленные формы списка отсутствуют в конфигурации, поэтому создадим формы списка для объектов конфигурации: Справочник Клиенты, Документ Приходная накладная, Документ Оказание услуги.

Теперь перейдем в окно настройки рабочего стола и над списком форм левой колонки нажмем Добавить.

Выберем форму списка документа Оказание услуги.

Аналогичным образом в правую колонку добавим формы списка документа Приходная накладная и справочника Клиенты.

Теперь для каждой из трех форм нажмем ссылку в колонке Видимость и установим видимость этих форм только для роли Мастер.

ЗАДАНИЕ:

1. Настроить рабочий стол для остальных ролей.
2. Проверить работу рабочего стола для разных пользователей в режиме отладки.

Лабораторная работа № 10. Программирование форм. События формы.

Цель работы: Знакомство с управляемой и обычной формой документа. Создание в системе управляемой и обычной формы документа.

Порядок выполнения работы:

До того, как мы начнем писать программный код, давайте научимся делать это быстро и без ошибок.

Синтакс-помощник

Синтакс-помощник открывается из меню "Справка" (рисунок 1) и содержит все книги по встроенному языку 1С: Предприятия. Синтакс-помощник позволяет выполнять поиск и быстрый поиск по индексу и при наборе с клавиатуры Ctrl+F1 на интересующем вас методе. Причем, при поиске будет учитываться контекст, которым вызывается поиск – на сервере или на клиенте должен использоваться код.

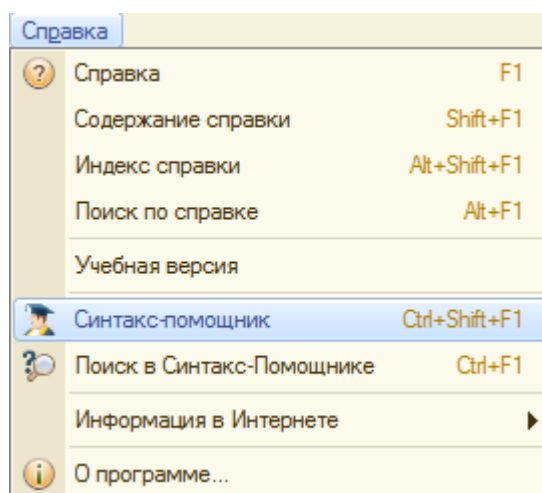


Рисунок 1 – Синтакс-помощник

В подавляющем большинстве случаев его будет достаточно для получения справки по встроенному языку, и в литературу заглядывать не придется.

Шаблоны текста

Значительная часть конструкций языка пишется всегда одинаково, и писать ее с клавиатуры не хочется. В конфигураторе есть возможность

автозамены строк, которой можно и нужно воспользоваться. С ее помощью вы не только будете писать быстрее, самое главное - вы будете допускать меньше ошибок.

Чтобы шаблоны текста заработали, их, во-первых, надо включить. "Главное меню конфигуратора / Сервис / Параметры" (рисунок 2).

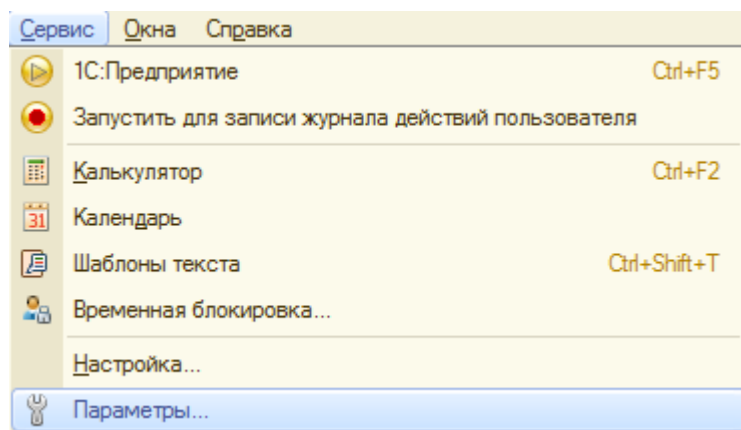


Рисунок 2 – Параметры конфигурации

Сами шаблоны можно увидеть в конфигураторе "Главное меню / Сервис / Шаблоны текста" (рисунок 3).

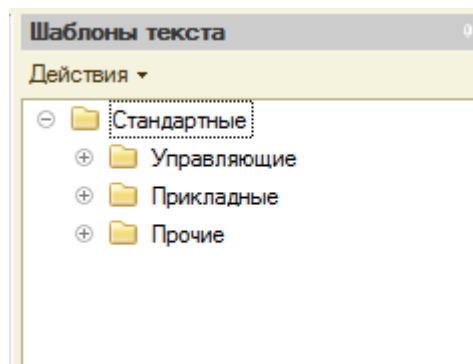


Рисунок 3 – Шаблон текста

Если они там есть, значит, какой-то файл шаблонов подключен. По умолчанию это стандартные шаблоны, поставляемые вместе с платформой в виде файла с расширением *.st.

Хранятся шаблоны во внешнем файле. Файлов шаблонов может быть несколько, какие из них программа подключит - можно настроить (в окне шаблонов "Действия / Настройка шаблонов"). Сами шаблоны редактируются в отдельном окне.

Так, например, описание шаблона *"Если"*, которое приведено на картинке, говорит о том, что если мы наберем слово *"Если"* (Автоматически заменять строку) в тексте модуля и нажмем *"Enter"* или *"пробел"*, то шаблон задаст нам вопрос с текстом *"Условие"* и допишет остальную часть кода сам.

Контекстная подсказка

Конфигуратор позволяет не держать в голове все тома встроенного языка с описанием свойств и методов объектов. Достаточно включить в параметрах конфигулятора контекстную подсказку. Подсказка может вызываться при наборе точки, при наборе *"="*, когда значение свойства объекта или имеет тип *<Булево>*, или системное перечисление, и при нажатии сочетания клавиш *"Ctrl + Пробел"*. Последний вариант позволяет, набрав только часть имени, дописать остальное одним нажатием сочетания клавиш.

Синтаксический контроль

Для проверки модуля на наличие синтаксических ошибок целесообразно использовать синтаксический контроль модулей, который вызывается по кнопке *Синтаксический контроль* с панели инструментов, или меню *Текст*. Можно установить свойство *"Проверять автоматически"* в параметрах конфигурации.

При установленном флаге программа сама будет запускать синтаксический контроль при закрытии каждого модуля. В большой конфигурации первый запуск синтаксического контроля модуля может занять несколько секунд. Все последующие запуски, в том же сеансе работы в конфигураторе, будут выполняться моментально.

Форматирование модуля и другие полезные свойства

Обратите внимание на панель инструментов, расположенную по умолчанию внизу слева под текстом модуля. Эти же команды можно вызвать из меню *"Текст"*. Кнопки этой панели позволяют запомнить то место в модуле, над которым идет работа (управление закладками). Кнопка форматирования позволит придать модулю опрятный и, главное, *"читабельный"* вид -

выровняет модуль "лесенкой". Кнопка перехода к строке позволит найти нужную строку по ее номеру.

Переход к определению процедур и функций

Отдельно следует обратить внимание на функцию перехода к определению функции/процедуры. В процессе работы код конфигурации будет становиться все больше и начнет изобиловать вызовами процедур функций.

Если вызываемая процедура/функция написана в другом (например, общем) модуле конфигурации или в этом же, но где-то далеко и сразу не видна, поможет клавиша "F12". Кстати, чтобы перейти обратно, нужно нажать сочетание клавиш "Ctrl + Минус в левой части клавиатуры".

ЗАДАНИЕ:

Теперь, когда мы, кажется, уже знаем достаточно теории, пора переходить к практике. Начнем с простой задачи. Разработаем калькулятор. Во-первых, что это такое с точки зрения конфигуратора? Это форма. Причем сама по себе форма, своя собственная. В том смысле, что нормальная форма, будь то форма справочника или документа, привязана через основной реквизит формы к объекту, данные которого в этой форме просматриваются-редактируются. А здесь форма ни к чему не должна быть привязана. Такие формы создаются в ветке *"Общие/Общие формы"* конфигурации.

Добавим новую общую форму по имени "Калькулятор".

Можно включить эту форму в подсистему «Общие» и/или разместить на рабочей области начальной страницы, как мы поступили со справочником «Валюты».

Реквизиты формы

Наш калькулятор будет выполнять суперсложную задачу: складывать два числа, которые будет вводить пользователь в две ячейки и показывать результат в третьей. Для этого нам для начала нужно добавить в реквизиты формы новые реквизиты "Число1", "Число2", "Число3", типа (кто бы мог подумать) <Число>.

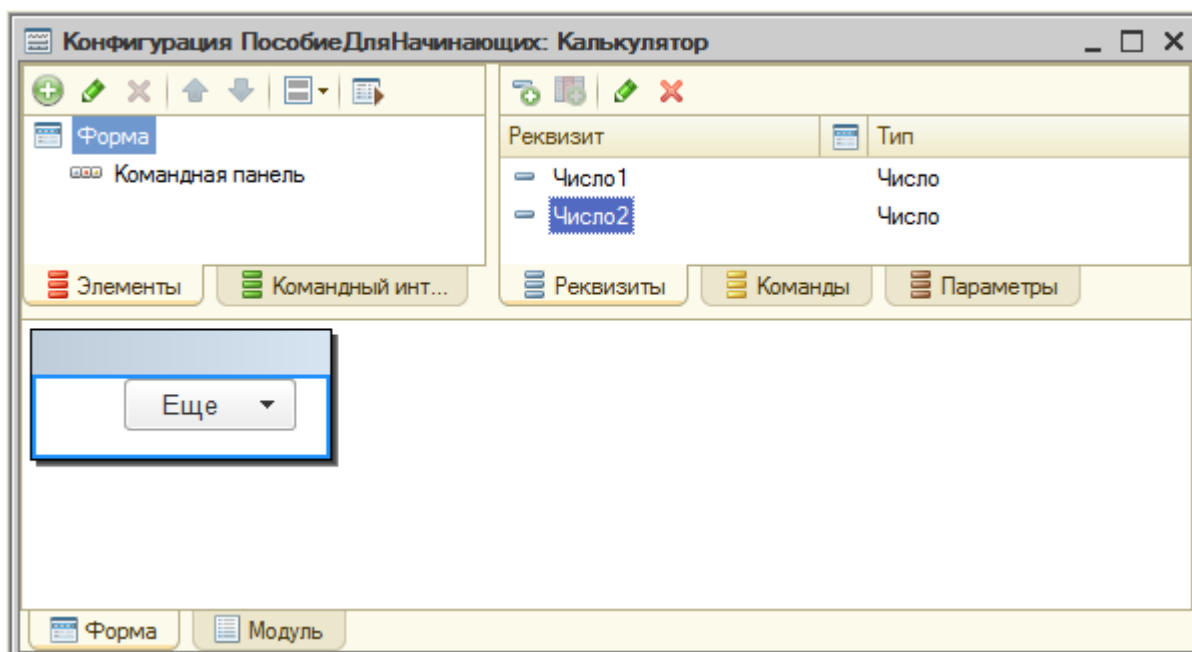


Рисунок 4 – Новые реквизиты формы

Команды формы

Теперь нужна кнопка, при нажатии на которую будет выполняться какое-то арифметическое действие над этими числами. Любая кнопка (на панели, в форме, гиперссылка) - это команда. В нашем случае - команда формы.

Команда создается в правом окне и переносится в левое, чтобы создать кнопку в диалоге. Что будет делать команда, описывается в ее свойствах, в окошке *"Действие"*. Если там уже что-то написано, то это имя процедуры, которую она выполняет, и по кнопке *"Открыть"* (лупа) можно перейти к ней в модуле и прочесть. Если ничего нет (у нас), то по кнопке *"Открыть"* будет создана процедура, содержимое которой будет выполнено, когда кнопка будет нажата пользователем. Перед созданием программа задаст вопрос, какую директиву (конструкцию) создать в модуле для исполнения нажатия на эту кнопку.

Выберем первый вариант (создать на клиенте, на сервер такую суперсложную задачу выносить не будем). Результат - в модуле будет добавлена процедура, нам останется лишь написать формулу внутри (листинг).

Листинг. Процедура "Сложить"

& НаКлиенте

Процедура Сложить (Команда)

Число3=Число1+Число2;

КонецПроцедуры

ЗАДАНИЕ.

Добавьте в "Калькулятор" кнопки "Вычесть" и "Умножить"

А как добавить кнопку "Разделить"? Думаете также? Не совсем. Делить на ноль нельзя. А раз так, необходимо добавить проверку - условный оператор (лист.). Написать его можно с помощью шаблона ("Если", Пробел).

Листинг. Процедура " Разделить"

& НаКлиенте

Процедура Разделить (Команда)

Число3=Число1/Число2;

КонецЕсли;

КонецПроцедуры

При желании все кнопки можно собрать в одну группу с горизонтальной группировкой (чтобы все помещалась на экране).

Обработчики событий элементов формы

Мы уже познакомились с командами, которые создают кнопки в диалоге. Итак, пользователь вводит цену, количество и нажимает кнопку "Рассчитать". В результате получает рассчитанную сумму строки накладной, а вы, как программист, получаете икоту на всю оставшуюся жизнь (он вспоминает "этого" программиста "этой" программы, часто используя при этом инвективную лексику).

Правильно! Прописали бы формулу в ту ячейку, которую надо рассчитать. Когда бы "срабатывала" эта формула? При любом изменении значения в любой ячейке, которая в формуле задействована. С точки зрения платформы "1С:Предприятие" это шибко жирно будет пересчитывать весь диалог, когда кто-то что-то в какой-то (любой) ячейке хочет поменять (хотя организовать такое тоже можно).

Поэтому в "1С" (и любой другой среде разработки) формула привязывается не к полю, которое надо рассчитать, а к полю, при работе с

которым надо что-то рассчитать, т.е. в нашем случае формула (процедура) будет привязана не к "Сумме", а к "Цене" и к "Количеству".

Такие процедуры, которые система будет вызывать сама, при наступлении какого-то события, называются обработками событий.

Найти их можно в форме, в данном случае это форма документа "ПоступлениеТоваров", элемент формы *"ТоварыЦена"*, открываем окно свойств - и в самом низу раздел *"События"*.

Изменить (добавить новые) состав событий, которые можно обработать, нельзя. Можно только использовать существующие.

Как правило, это событие *ПриИзменении*. Зачем пользователь зашел в это поле? Чтобы изменить цену! Процедура, имя которой мы привяжем к этому событию, будет выполняться при интерактивном изменении цены в диалоге формы документа.

По нажатию кнопки "Открыть" в свойстве "ПриИзменении" будет создана процедура "На клиенте". Рассмотрим первый вариант (листинг).

Листинг. Обработчик события "При изменении". Первый вариант &НаКлиенте

```
Процедура ТоварыЦенаПриИзменении(Элемент)  
Элементы.Товары.ТекущиеДанные.Сумма=  
Элементы.Товары.ТекущиеДанные.Цена*  
Элементы.Товары.ТекущиеДанные.Количество;  
КонецПроцедуры
```

Если реквизиты "Сумма", "Цена" и "Количество" были бы реквизитами шапки документа, а не табличной части, то была бы одна строка, совсем простая (листинг).

Листинг. Синтаксис формулы расчета суммы для реквизитов шапки

```
Объект.Сумма = Объект.Цена * Объект.Количество
```

где "Сумма", "Цена" и "Количество" - реквизиты документа, к которым мы получаем доступ из формы через основной реквизит формы "Объект".

Но это ячейки строки табличной части. Какой строки? Той, в которой сейчас стоит курсор и которую редактирует пользователь. В ней и надо

рассчитывать сумму. Данные текущей строки можно получить, используя коллекцию элементов формы "Элементы". В ней можно найти по имени таблицу, а уже в той получить данные активной строки "ТекущиеДанные". А в строке есть ячейки "Сумма", "Цена" и "Количество".

Только выглядит это очень громоздко. При каждом обращении мы повторяем полный путь к ячейке "Элементы.Товары.ТекущиеДанные". В таких случаях "общее" принято "выносить за скобку" - на отдельную переменную (листинг).

Листинг. Обработчик события "При изменении". Второй вариант

&НаКлиенте

Процедура ТоварыЦенаПриИзменении(Элемент)

Стр=Элементы.Товары.ТекущиеДанные;

*Сир.Сумма=Стр.Цена*Стр.Количество;*

КонецПроцедуры

В данном примере переменная "Стр" придумана нами без труда и объявлена с одной единственной целью - чтобы проще общаться к данным текущей строки.

Общее в отдельную процедуру

А при изменении количества пересчет суммы выполнить надо? Сместем полагать - да. Что для этого можно сделать?

Первый вариант - зайти в свойства элемента формы "ТоварыКоличество" и привязать к событию при изменении ту же процедуру, выбрав ее из списка. Чем это решение нехорошо: в результате со временем выяснится, что для количества, кроме расчета суммы, еще что-то нужно выполнять. А процедура одна.

Второй вариант - сделать еще одну процедуру, совершенно такую же. Чем плохо? Потом опять-таки выяснится, что Сумма это не "Цена * Количество". А, скажем, "Цена * Количество * Коэффициент". Придется в обеих процедурах править. В одной поправим, в другой забудем... Результат - ошибка.

Третий вариант - вынести расчет строки в отдельную процедуру, и вызывать ее из обработчиков событий (листинг).

Листинг. Процедура "Расчет суммы", и ее вызов из обработчиков

&НаКлиенте

Процедура ТоварыЦенаПриИзменении(Элемент)

Стр=Элементы.Товары.ТекущиеДанные;

*//Стр.Сумма=Стр.Цена*Стр.Количество;*

РасчетСтроки(Стр);

КонецПроцедуры

&НаКлиенте

Процедура ТоварыКоличествоПриИзменении(Элемент)

Стр=Элементы.Товары.ТекущиеДанные;

РасчетСтроки(Стр);

КонецПроцедуры

&НаКлиенте

Процедура РасчетСтроки(ТекСтр)

*ТекСтр.Сумма=ТекСтр.Цена*ТекСтр.Количество;*

КонецПроцедуры

Где первые две процедуры - обработчики событий, привязанные к элементам формы. А третья процедура добавлена нами. Она получает на параметр (который в методических целях назван ТекСтр, а не Стр, чтобы показать, что параметры передаваемые и получаемые могут называться по-разному) данные строки документа, где пользователь изменил цену или количество и рассчитывающая в этой строке сумму.

Внимание!

Перед процедурой обязательно нужна директива *"НаКлиенте"*, потому что на сервере конкретно эта исполняться вообще не будет, а если директивы нет - значит на сервере.

А что произойдет, если пользователь изменит в строке сумму? Нехорошо получается! Надо или запретить изменять сумму (чтобы она всегда рассчитывалась и была равна произведению цены на количество), или дописать еще один расчет: если пользователь вводит сумму и цена равна нулю, а количество не равно (делить на ноль нельзя), то $Цена = Сумма / Количество$. Оставим пользователю возможность изменять сумму, а обработчик расчета "в обратную сторону" необходимо описать.

Кстати, делить на ноль нельзя (вдруг пользователь еще не успел ввести количество). Надо это обработать. Потребуется условный оператор.

Директива "НаСервере"

Когда пользователь изменяет номенклатуру, имеет смысл подставить в строку документа цену. Хранится эта цена в реквизите "Цена" справочника "Номенклатура".

Задача вроде бы несложная: создать обработчик изменения номенклатуры (исполняемый на клиенте, где собственно и происходит действие пользователя), получить строку таблицы (это мы уже умеем - используя *ТекущиеДанные*), получить из строки значение реквизита "Номенклатура", а затем выяснится, что на клиенте (имеется в виду "Тонкий клиент") отсутствует значительная часть возможностей встроенного языка. Например, мы не можем в процедуре исполняемой на клиенте, обратиться к базе данных и получить из нее значение реквизита "через точку" (листинг).

Листинг. Первый вариант решения

```
&НаКлиенте
Процедура ТоварыНоменклатураПриИзменении(Элемент)
Стр = Элементы.Товары.ТекущиеДанные;
Стр.Цена = Стр.Номенклатура.Цена;
КонецПроцедуры
```

Если вы проверите эту процедуру, получите сообщение, что "не найдено поле Цена". Оно не найдено на клиенте. Оно есть только на сервере, который может обратиться к базе данных и прочитает значение этого поля из таблицы справочника "Номенклатура".

Правильно будет написать еще одну функцию, которая должна будет "сходить" на сервер, получить цену и вернуть ее на клиента (листинг).

Листинг. Первый рабочий вариант решения

```
&НаКлиенте
Процедура ТоварыНоменклатураПриИзменении(Элемент)
Стр=Элементы.Товары.ТекущиеДанные;
Стр.Цена = ПолучитьЦену(Стр.Номенклатура);
РасчетСтроки(Стр);
КонецПроцедуры
&НаСервере
Функция ПолучитьЦену(Товар)
Возврат Товар.Цена;
КонецФункции
```

Почему функция, а не процедура? Потому что так удобнее: мы ей ссылку на номенклатуру, а она нам - цену товара. Намного проще обмениваться данными между сервером и клиентом, если данные простые (примитивные и ссылка).

А зачем мы еще раз сумму пересчитываем? Мы ведь чуть раньше сказали, что если изменяется цена, сумму пересчитать. Цена изменилась! Это, конечно, да. Только она изменилась программно. Обработчики событий формы исполняются только при интерактивном (руками пользователя) изменении.

Обычная форма документа

А как бы это выглядело в режиме обычного приложения? Мы нарисовали обычную форму документа, давайте теперь посмотрим на то, как научить ее рассчитывать сумму. Найти обработчики событий можно, вызвав окошко свойств на поле ввода реквизитов "Цена", "Количество", "Номенклатура". А знакомый нам программный код будет выглядеть немного проще (листинг), но иметь существенно меньше способов управления производительностью.

Листинг. Модуль обычной формы

```
Процедура ТоварыЦенаПриИзменении(Элемент)
Стр=ЭлементыФормы.Товары.ТекущиеДанные; РасчетСтроки(Стр);
КонецПроцедуры
Процедура ТоварыКоличествоПриИзменении(Элемент)
Стр=ЭлементыФормы.Товары.ТекущиеДанные; РасчетСтроки(Стр);
КонецПроцедуры
Процедура РасчетСтроки(ТекСтр)
ТекСтр.Сумма=ТекСтр.Цена*ТекСтр.Количество; КонецПроцедуры
Процедура ТоварыНоменклатураПриИзменении(Элемент)
Стр=ЭлементыФормы.Товары.ТекущиеДанные;
Стр.Цена=Стр.Номенклатура.Цена; КонецПроцедуры
```

Основные отличия: нет директив, платформа сама будет разбирать, какую часть работы, где делать и как (почти все сделает на клиентской машине), несколько иначе называются свойства (*ЭлементыФормы*).

Работа в отладчике

Каждый раз, когда мы запускаем нашу конфигурацию для проверки в пользовательском режиме, мы нажимаем кнопку "Начать / продолжить отладку" или клавишу F5. Что такое "отладка" программы?

В первую очередь, это возможность остановить выполнение кода на нужной строчке и посмотреть, какое значение в какой переменной на этом этапе находится. И по ходу действия понять структуру объектов и их свойств. Кроме того, это возможность:

- пройти выполняемый модуль по шагам,
- выполнить замер производительности и найти "узкие" места кода,
- остановиться при появлении ошибки и завершить выполнение кода при "зацикливании", не пользуясь "тремя кнопками" и выключением компьютера.

Все эти задачи (и многие другие, но реже) и решает отладчик. Из конфигуратора запускаем программу по кнопке "Начать отладку" и начинаем.

Точка останова

Итак, мы хотим понять, как работает наш механизм получения цены номенклатуры из справочника. Чтобы это понять, мы должны установить точку останова на интересующую нас строку содержимого этой процедуры. Установить ее можно либо двойным щелчком мыши на левой серой полоске модуля, либо с помощью меню "Отладка".

Поставить точку останова можно только на уже сохраненной строке кода. Если эта строка только что написана и еще отсутствует в "собранной" скомпилированной программе, отладчик скажет, что не может это сделать. Придется сначала "перезапустить" режим отладки.

В пользовательском режиме открываем документ "Поступление товаров и услуг" и выполняем с ним проверяемое действие - изменяем в строке позицию номенклатуры, т.е. в строке документа начинаем редактирование поля "Номенклатура", открываем список справочника и выбираем из него позицию номенклатуры.

Как только программа дойдет до точки останова, отладчик возьмет управление на себя, переключит нас в конфигуратор, подсветив обрабатываемую строчку кода стрелочкой III.

Чтобы продолжить выполнение кода построчно, нужно войти в меню "откладка" и найти там пункты меню "Шагнуть..." или сделать это с клавиатуры. Чаще всего используется "F11" и "Shift + F10". Первый из них позволяет идти по строчкам. Вторым (идти до курсора, называется) - перепрыгивать через неинтересные куски кода, останавливаясь только на интересных строчках, на которые для этого мы и ставим курсор.

При работе в табло можно продолжать отладку и наблюдать, как изменяются значения рассматриваемых в табло переменных.

Значения переменных, вычислить выражение

Если в процессе отладки подвести мышку к переменной (например, *Строка.Цена*), в подсказке мы увидим текущее значение. Само собой, чтобы это значение увидеть, строка уже должна быть вычислена, т.е. "стрелочка" должна быть ниже строки, в которой мы хотим посмотреть значение переменной. Если переменная сложная (например, *Элементы* или *Строка*), то этот метод не подходит. Лучше установить курсор на проверяемую переменную и нажать Shift + F9 или пункт контекстного меню "Вычислить выражение".

Чтобы обновить изображение выражения или рассчитать значение любой переменной модуля (свойства объекта), нужно набрать имя выражения и нажать кнопку "Рассчитать".

Кстати, обратите внимание, отладчик находится сейчас в процедуре, которая выполняется на клиенте, видим мы только те свойства, которые программа принесла с сервера на клиентскую машину. Например, про номенклатуру мы знаем только представление (*Паркер*) и тип (*СправочникСсылка.Номенклатура*).

Если проверить, как работает "Вычислить выражение" на сервере, мы бы получили возможность "раскрыть дерево" свойств товара. Открывшийся

диалог позволяет увидеть значения сложных переменных, если раскрыть дерево полей или по кнопке открыть значение этой переменной в отдельном окне.

Табло

У диалога "Вычислить выражение" есть существенный недостаток. Диалог модальный, и, пока он открыт, выполнение модуля (переход на следующую строку) продолжить нельзя.

Чтобы при переходе на следующую строку не приходилось опять искать в "дереве" реквизитов нужный, можно включить его (или их, так как их обычно несколько) в табло.

Это можно сделать по кнопке "Включить в табло" в окне выражения или просто, открыв табло по кнопке из "Меню / Отладка", и перетаскив в него нужную переменную из модуля.

Стек вызовов

В нашей конфигурации структура вызовов процедур и функций не слишком запутанная. В реальных решениях имеется много вызовов функций и процедур общих модулей, и просто других модулей (модуль формы, модуль объекта, модуль менеджера). При этом можно быстро запутаться, из какой строки одного модуля была вызвана та или иная процедура, где был выполнен переход в другой модуль.

Для анализа точек перехода используется стек вызовов, который также можно открыть из *"Меню / Отладка"*.

Завершение отладки

Чтобы закончить отладку, когда все стало понятно, можно нажать *"Отладка / Продолжить отладку"*. Это в том случае, если все нормально, все работает. Если же стало понятно, что дальше продолжать бессмысленно (ошибка ясна, ее надо исправлять), нажимаем *"Отладка / Завершить отладку"*.

Лабораторная работа № 11. Программное выполнение запроса.

Цель работы: Конструктор запроса с обработкой результата. Обработка результата запроса.

Порядок выполнения работы:

До этого момента запрос мы использовали только в отчетах, в механизме компоновки и в форме списка, совершенно не задумываясь, как он выполняется, как ему передать текст запроса, как получить результат и что с ним делать.

Ранее нами в модуле формы документа "Поступление товаров" был описан несложный механизм получения цены из элемента справочника "Номенклатура". Для этого использовалась объектная модель и обращение "через точку" к реквизиту выбранного элемента справочника номенклатуры.

В функции, выполняемой на сервере приложения, была одна строка кода и возврат результата (листинг).

Листинг. Обращение к реквизиту в объектной модели

```
&НаСервереВезКонтекста  
Функция ПолучитьЦену(Товар)  
Возврат Товар.Цена;  
КонецФункции
```

Среди прочего, упоминалось, что для выбранного случая это вполне приемлемый вариант, но можно было бы сделать и запросом, чтобы оптимизировать получение данных. Давайте перепишем эту функцию таким образом, чтобы она получала данные запросом.

Запрос - объект платформы, который, получив на себя текст запроса и значения параметров, в нем используемых, позволяет на стороне сервера выполнить запрос к базе данных и обработать полученный результат. Запрос используется многими механизмами платформы неявно (динамический список, механизм компоновки данных и другие).

Конструктор запроса с обработкой результата

Итак, мы изменяем функцию ПолучитьЦену. Закомментируем (или удалим) старое ее содержимое (одну строку) и напишем алгоритм получения цены с сервера по-новому.

Для написания текста запроса и "оболочки" работы с ним воспользуемся конструктором запроса с обработкой результата , который можно вызвать из контекстного меню по правой кнопке мыши в модуле формы.

На первой странице "Обработка результата" конструктора определим, для чего этот запрос будет использоваться (рисунок).

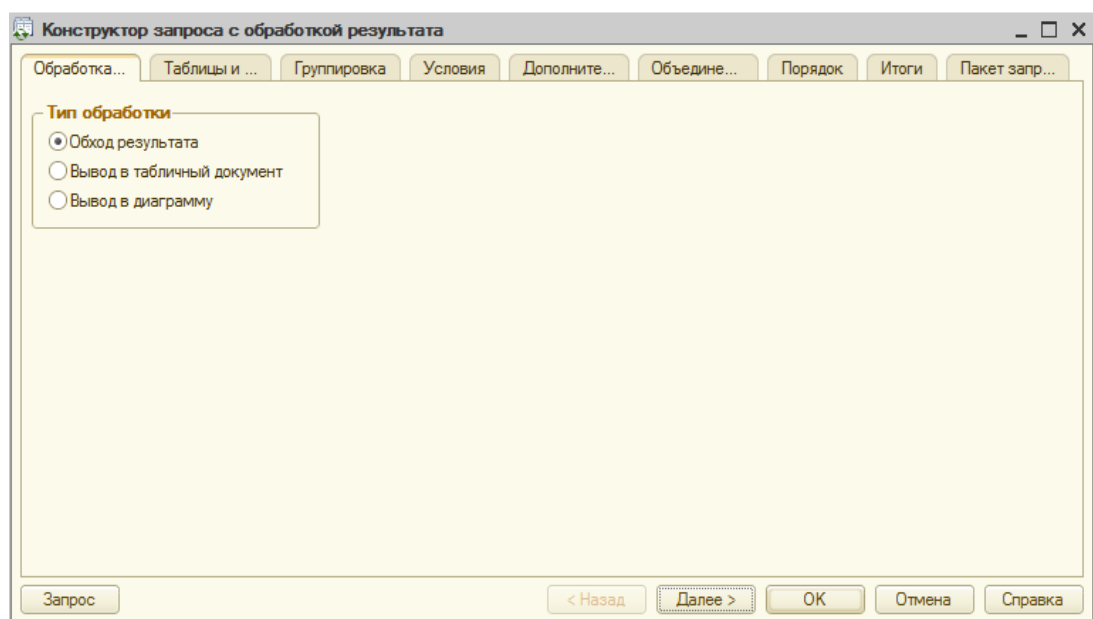


Рисунок 1- Страница «Обработка результата» конструктора

Результат запроса может быть нужен для вывода в табличный документ (построение отчета без использования механизма компоновки данных, подготовка печатной формы документа и другие аналогичные задачи), для вывода в диаграмму (опять-таки без использования механизма компоновки данных) и просто для того, чтобы обратиться к нему программно - "Обход результата". Этот вариант нам и нужен.

Далее страницы конструктора фактически полностью соответствуют страницам конструктора запроса, который мы использовали для написания текста запроса при построении отчетов с помощью механизма компоновки

данных (нет страниц, связанных с характеристиками и специальным расширением языка запросов для компоновки). Выбираем источником данных таблицу справочника "Номенклатура". Из всей таблицы нам нужно поле "Цена".

На закладке "Условие" ставим отбор по ссылке, тому самому служебному полю, который хранит уникальный идентификатор элемента справочника, который и содержится в реквизите типа <СправочникСсылка.Номенклатура>.

Можно нажать на кнопку "Запрос" в левом нижнем углу, чтобы до окончания работы в конструкторе посмотреть на текст запроса (листинг).

Листинг. Текст запроса получения цены номенклатуры

```
ВЫБРАТЬ
    Номенклатура.Цена
ИЗ
    Справочник.Номенклатура КАК Номенклатура
ГДЕ
    Номенклатура.Ссылка = &Ссылка
```

Нажимаем на кнопку "ОК" и получаем почти готовый для работы кусок кода (листинг), вносим в него изменения - и можно проверять.

Листинг. Функция ПолучитьЦену запросом

&НаСервереБезКонтекста

Функция ПолучитьЦену (Товар)

```
Запрос = Новый Запрос; //1
Запрос.Текст = //2
    "ВЫБРАТЬ
    |     Номенклатура.Цена
    |ИЗ
    |     Справочник.Номенклатура КАК Номенклатура
    |ГДЕ
    |     Номенклатура.Ссылка = &Ссылка"; //3
Запрос.УстановитьПараметр("Ссылка", Товар); //4
Результат = Запрос.Выполнить(); //5
Выборка = Результат.Выбрать(); //6
Если Выборка.Следующий() Тогда //7
    Возврат Выборка.Цена; //8
КонецЕсли; //9
Возврат 0; //10
```

КонецФункции

Обработка результата запроса

1. Для программного выполнения запроса в платформе предусмотрен специальный объект "Запрос". Конструктор "Новый" его создает. Далее, с помощью созданной переменной типа <Запрос> мы сможем его выполнить. Запрос можно представить "котлом": мы туда положим некие условия, а он вернет нам результат после выполнения.

2. Для работы запросу обязательно нужен текст, написанный на языке запросов, с которым мы уже знакомы. Сам текст мы уже знаем, а вот над тем, что это длинная строка, может быть, и не задумывались. Это самая обычная строка, которая начинается с двойной кавычки и ею же заканчивается на строке с комментарием //3. Но писать эту строку "в одну строку модуля" неудобно - на экране не помещается. Для переноса текстовой строки на новую строку модуля без обрыва используется знак "|". Этот знак используется не только в текстах запроса, но и, например, при написании сообщений пользователю.

3. Итак, двойные кавычки закрылись, точка с запятой закрывает строку кода, начавшуюся ранее.

4. Если в тексте запроса встречаются параметры &, их надо в запрос передать. Когда платформа будет выполнять запрос, она заменит в тексте запроса & *ИмяПараметра* на значение, которое найдет под этим именем в структуре переданных параметров. Метод *УстановитьПараметр* записывает параметры в структуру Параметры запроса.

5. Когда у запроса есть все необходимое (у запроса есть не только текст и параметры, но и другие свойства, хотя в рамках знакомства с запросом мы их рассматривать не будем), запрос надо выполнить. Именно в этот момент платформа выполнит запрос (файловая версия) или отдаст его СУБД (клиент-серверный вариант), и через какое-то время получит его результат.

6. С результатом запроса можно делать самые разные вещи. Можно превратить его в таблицу значений и просмотреть, можно передать в сводную таблицу или диаграмму, а можно обработать самостоятельно в

модуле и получить из него данные. В любом случае результат - таблица. Колонки этой таблицы соответствуют полям запроса и так же называются.

Если мы назначали псевдонимы полям запроса, имена колонок будут соответствовать псевдонимам. В нашем случае в таблице результата только одна колонка по имени "*Цена*". Чтобы пройти по строкам запроса, их сначала надо получить, т.е. "*Выбрать*". Выборка будет содержать коллекцию строк запроса. Имея одну из этих строк и имена колонок, можно получить значение ячейки из строки.

7. Мы изменили текст кода, который предложил конструктор. Он предполагал, что в таблице результата может быть несколько строк, поэтому предложил обойти их в цикле. Метод "Следующий" имеет два предназначения в этом случае. Во-первых, он возвращает ИСТИНА до тех пор, пока есть следующая строка выборки запроса, и может использоваться как условие цикла (листинг).

Листинг. Пример обхода выборка запроса циклом

```
Пока Выборка.Следующий() Цикл
    Цена = Выборка.Цена;
КонецЦикла;
```

Во-вторых, он позиционирует нас на строке, т.е. после вызова метода "Следующий" мы можем обращаться к ячейкам строки по именам колонок. Поэтому, даже если заранее известно, что строка в результате запроса может быть только одна (ну, не может же быть у одного элемента справочника двух разных значений одного реквизита цена!), метод "Следующий" все равно нужен. И люди аккуратные пишут его в условном операторе (листинг).

Листинг. Позиционирование на строке запроса в условном операторе

```
Если Выборка.Следующий() Тогда
    Возврат Выборка.Цена;
КонецЕсли;
```

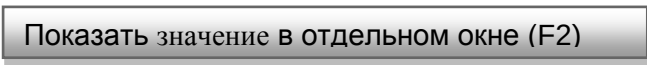
Зачем нам условие? Мы же заранее знаем - строка одна. А если номенклатура в строке пустая, сколько будет строк? Правильно - ни одной. И что тогда? Тогда результат запроса будет пустой, в переменной "Выборка"

окажется "Неопределено", а при попытке получить из нее "Цену" мы получим ошибку.

8. Из строки выборки запроса значение ячейки мы получаем по имени колонки. Нашу таблицу можно представить так (таблица):

Индекс	Цена
0	Значение цены типа <Число>

Чтобы увидеть эту таблицу, можно воспользоваться отладчиком. Установите точку останова внутри условного оператора и посмотрите на запрос, его свойства и производные в диалоге "вычислить выражение".

Кнопка , особенно актуальна для просмотра текста запроса, если тот собирался в модуле динамически (программно из кусков).

Для просмотра таблицы результата запроса, его следует выгрузить: напишите в строке выражения (листинг).

Листинг. Выгрузка результата в таблицу значений

Результат.Выгрузить()

Тогда можно будет в отдельном окне увидеть таблицу, которую мы будем обходить в модуле при обработке результата.

9. Закрыв условный оператор, мы вспоминаем о том, что это функция. Если в функции не выполнялась строка "Возврат", это не означает, что она ничего не вернет. Она вернет "ничего", которое в объектной модели данных представлено специальным типом данных <НЕОПРЕДЕЛЕНО>. <НЕОПРЕДЕЛЕНО> возникает тогда, когда, например, искали и ничего не нашли. Чем-то похоже на <NULL> в табличной модели данных.

И делать с <НЕОПРЕДЕЛЕНО> ничего нельзя. Уж точно нельзя умножить <НЕОПРЕДЕЛЕНО> на <число> "Количество" и получить "Сумму". Поэтому мы подстраховываемся: если мы ничего не получили в результате запроса, и следующей (вообще ни одной) строки в выборке запроса не было, мы возвращаем "ноль". В функции может быть (и обычно так и есть) несколько строк с директивой "Возврат". Во-первых, они

останавливают выполнение функции, во- вторых, возвращают какое-то значение. Это значит, что если мы войдем внутрь условного оператора (есть цена) и выполнится строка //8, до строки //10 мы просто не дойдем. Если же условие не выполнится, мы пропустим строку //8 и закончим выполнение функции на строке //10.

Если нужно переписать текст запроса (не трогая обработку результата), можно воспользоваться конструктором запроса, который можно запустить из



контекстного меню по правой кнопке мыши прямо на тексте запроса.

Анализ производительности

Каждый раз, когда мы запускаем режим отладки, в левом нижнем, как правило, углу, рядом с одноименной кнопкой, показывается всплывающая подсказка "Показатели производительности" (рисунок).

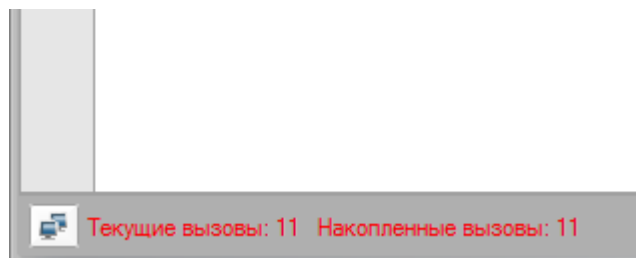


Рисунок 2- Подсказка показателей производительности

Зачем нужны показатели производительности

Задача этой подсказки обратить внимание разработчика на количество и "качество" вызовов, которые делает пользователь конфигурации к серверу. Чем их меньше, тем лучше.

Обычно диалог показывает количество текущих (одно действие повлекло столько-то вызовов сервера) и количество накопленных (итог с момента запуска или после принудительного обнуления счетчика) вызовов. Обнулить счетчик, а также настроить подсказку можно с помощью контекстного меню, вызванного по кнопке "Показатели производительности" (рисунок).

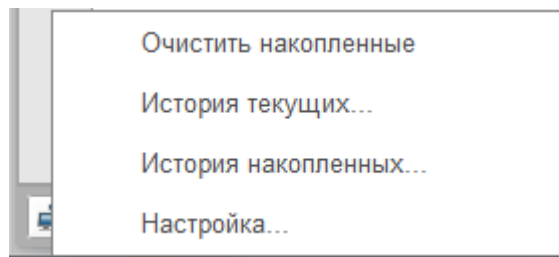


Рисунок 3- Меню подсказки

Настройка позволит управлять детальностью выводимых данных. На количество вызовов обязательно нужно обращать внимание при разработке конфигурации.

Результат один - скорость разная

А теперь решим игрушечную задачку: добавим в диалог кнопку, при нажатии на которую программа будет для каждой строки документа получать из справочника цену товара, подставлять ее в строку и пересчитывать сумму.

Добавим новую команду "Сделать" с одноименной процедурой-действием и флажком "Изменяет сохраняемые данные". Флажок нужен, чтобы после нажатия на кнопку форма знала, что ее изменили, и не разрешала закрыть себя, не задав пользователю вопрос о сохранении.

В модуле формы опишем содержимое обработчика (листинг)

Листинг. Первый вариант решения

&НаКлиенте

Процедура *Сделать(Команда)*

Для каждого Стр Из Объект.Товары Цикл

Стр.Цена = ПолучитьЦену(Стр.Номенклатура);

РасчетСтроки(Стр);

КонецЦикла;

КонецПроцедуры

Готово. Попробуйте провести эксперимент. Введите, например, сотню строк в накладную (можно копированием, клавиша "F9" позволяет это сделать).

Нажмите на кнопку. Ну, как? Так вот, чтобы пересчитать одну накладную наша программа больше сотни раз сбегала в Швецию (мы там сервер арендуем) и обратно. С точки зрения жизни - плохо, с точки зрения экзамена по программированию³⁴ в фирме "1С" - единица.

Оптимизируем код. Уберем вызов сервера в цикле. Нам это, кстати, сразу предлагали, когда мы создавали обработчик действия кнопки, надо было выбрать вариант "Создать на клиенте и процедуру на сервере".

А почему не "на сервере без контекста"? Потому что контекст формы нам в этот раз нужен (мы его и собираемся пересчитывать). Итак, второй вариант.

Выносим весь пересчет на сервер приложения. Правда при этом расчет строки работать перестанет – мы не можем обращаться с сервера к клиентским процедурам. Придется изменить директиву перед процедурой «РасчетСтроки». С новой директивой она будет выполняться на клиенте, если ее вызывают с клиента (например, при изменении цены), и на сервере, если ее вызывают с сервера (наш новый случай).

Листинг. Второй вариант решения

```
&НаКлиентеНаСервереБезКонтекста
Процедура РасчетСтроки(ТекСтр)
    ТекСтр.Сумма=ТекСтр.Цена*ТекСтр.Количество;
КонецПроцедуры

&НаКлиенте
Процедура Сделать(Команда)
    СделатьНаСервере();
КонецПроцедуры

&НаСервере
Процедура СделатьНаСервере()
    Для каждого Стр Из Объект.Товары Цикл
        Стр.Цена = ПолучитьЦену(Стр.Номенклатура);
        РасчетСтроки(Стр);
    КонецЦикла;
КонецПроцедуры
```

Лабораторная работа № 12. Экспортируемые процедуры и общие модули.

Цель работы: Свойства общего модуля. Перенос процедуры в общий модуль. Вызов процедуры из общего модуля.

Порядок выполнения работы:

Любая процедура, функция, переменная, объявленная в модуле, видна только в этом модуле. Если нам необходимо получить возможность вызова процедур (функций), такие процедуры (функции) должны быть "экспортируемыми". Экспортируемые процедуры, функции, переменные отмечаются в тексте модуля ключевым словом "Экспорт".

У нас есть процедура *"РасчетСтроки"*, расположенная в модуле формы документа *"ПоступлениеТоваров"*. Рано или поздно, но мы доберемся до второго нашего документа *"РеализацияТоваров"*. Реквизиты ("Цена", "Количество", "Сумма")

Чтобы не писать один и тот же алгоритм два раза (и второй раз с ошибкой), процедуру можно и нужно вынести в библиотеку процедур - общий модуль.

Свойства общего модуля

Общие модули расположены в ветке *"Общие"* и представляют собой библиотеки процедур и функций: такой модуль не может содержать раздела объявления переменных и тела модуля, только процедуры и функции. Как правило, они экспортируемые, исключение составляют лишь те процедуры и функции модуля, которые используются только в этом же модуле. Добавим новый модуль, перенесем туда процедуру *РасчетСтроки* (рисунок).

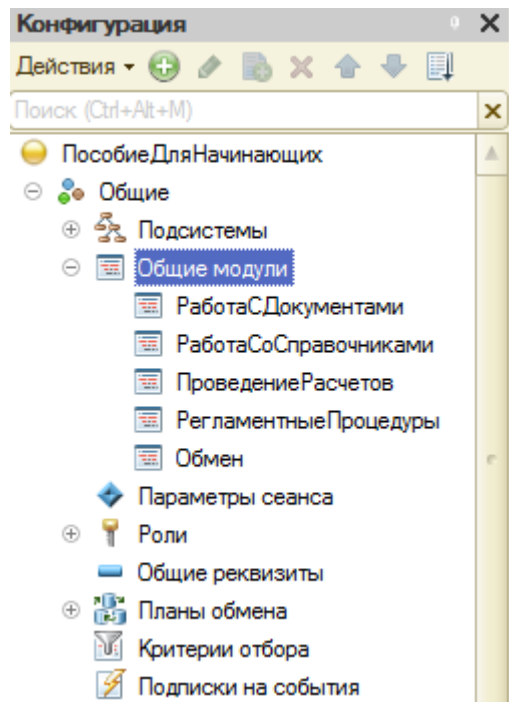


Рисунок 1- Общий модуль

Называют модули, как правило, так, чтобы было понятно, какие процедуры там содержатся и где исполняются. В нашем случае, мы назвали его *"ПроцедурыДокументовКлиентСервер"*. Исполняться этот модуль будет на клиенте и сервере (мы отмечаем это флажками, которые здесь используются вместо директив препроцессора). Обратим внимание также и на другие свойства модуля. Флаг "Глобальный" означает, что экспортируемые процедуры и функции модуля будут загружены в память при старте программы и будут там "висеть" в течение всего сеанса работы. К таким процедурам и функциям можно обращаться из любого места программы без каких либо оговорок, как будто они расположены в этом же модуле. Как правило, глобальных модулей мало (один-два), и размещается там только то, что нужно всем пользователям. Даже в разных глобальных общих модулях имена процедур не могут повторяться.

Неглобальные модули загружаются в память при первом обращении к любой их процедуре или функции. И остаются там до конца сеанса работы пользователя. Таким образом, если в нашей программе работает расчетчик зарплаты и кладовщик, одному из них нужен расчет среднего заработка, а

другому - расчет средней стоимости списания товара, то имеет смысл сделать два неглобальных модуля. Тогда у каждого из наших пользователей в памяти будет только нужная ему процедура. Обращение к экспортной процедуре осуществляется через имя модуля (листинг).

Листинг. Обращение к экспортируемой процедуре не глобального модуля

ПроцедурыДокументовКлиентСервер.РасчетСтроки(Стр);

Флаг "Внешнее соединение" делает содержимое модуля доступным при работе с программой через СОМ-соединение. Флаг "Вызов сервера" необходим в том случае, если модуль выполняется на сервере и обращается к данным сервера, но вызывать его процедуры и функции мы будем с клиента. Флаг "Привилегированный" позволит отключить на время выполнения модуля контроль доступа пользователя на уровне записей. В нашей конфигурации этого нет, но если бы мы хотели разрешить отдельным пользователям работать только с документами одной организации, пришлось бы добавить. Этот контроль занимает существенное время и не всегда нужен. Свойство "Повторное использование..." позволит сохранять в кэше сеанса выполненные ранее в этом сеансе (в этом же вызове) значения функций (только функций) модуля, в зависимости от переданных параметров. При повторном вызове можно будет взять готовое. Этот механизм надо использовать очень аккуратно.

Перенос процедуры в общий модуль

А теперь посмотрим на содержимое нашего модуля (листинг).

Листинг. Процедура РасчетСтроки общего модуля

*Процедура РасчетСтроки(ТекСтр) Экспорт
ТекСтр.Сумма=ТекСтр.Цена*ТекСтр.Количество;
КонецПроцедуры*

Осталось изменить вызов этой процедуры из обработчиков событий формы и модулей объектов "Документ" для всех документов, и все будет работать.

Вызов процедуры из общего модуля

Чтобы вызвать процедуру из общего модуля, пишем в начале имя модуля.

Листинг. Вызов процедуры общего модуля из клиентской процедуры

&НаКлиенте

Процедура ТоварыЦенаПриИзменении(Элемент)

Стр=Элементы.Товары.ТекущиеДанные ;

*//Стр.Сумма=Стр.Цена*Стр.Количество;*

//РасчетСтроки(Стр);

ПроцедурыДокументовКлиентСервер.РасчетСтроки(Стр);

КонецПроцедуры

Листинг. Вызов процедуры общего модуля из серверной процедуры

&НаСервере

Процедура СделатьНаСервере()

Для каждого Стр Из Объект.Товары Цикл

Стр.Цена = ПолучитьЦену(Стр.Номенклатура);

ПроцедурыДокументовКлиентСервер.РасчетСтроки(Стр); КонецЦикла;

КонецПроцедуры

Готово. То же самое можно сделать в других документах, для которых подходит процедура.

Лабораторная работа № 13. Модуль объекта. События объекта.

Цель работы: Перед записью документа. Условное оформление формы.

Порядок выполнения работы:

На этапе конфигурирования мы добавили в документ "ПоступлениеТоваров" один склад в "шапку" документа. А в документ "РеализацияТоваров" два склада: один в "шапку" документа (если все товары продаются с одного склада) и один в табличную часть документа, чтобы можно было указать свой склад для каждой строки (если одна накладная списывает товары с разных складов).

Пора подумать над тем, какая будет взаимосвязь между складом в "шапке" и в табличной части. Предлагаю сделать так: если пользователь заполнил склад в "шапке" документа, то выбранный пользователем склад сам (с помощью нашей магии) прописывается в каждую строку табличной части

документа. При этом колонка со складами в табличной части должна быть невидима.

Если же склад в *"шапке"* не выбран, то колонка со складами в таблице видна пользователю, и он может в каждой строке выбрать свой склад.

Перед записью документа

Во-первых, сделаем, чтобы выбранный в пользователем в *"шапке"* склад прописывался в строки табличной части. В какой момент это должно происходить? Видимо, когда документ начал записываться в базу, но еще не записался. Это событие называется *"ПередЗаписью"*. Оно есть и у формы *"Документы"* и объекта *"Документ"*. Но событие формы будет выполняться только при интерактивной записи документа. Например, если пользователь запустит обработку, которая *"переносит"* документы из одной программы (оперативный контур) в другую (бухгалтерский), то оно не выполнится - ведь форму документа никто не открывал и на кнопки *"записать"* никто не нажимал.

Поэтому надо воспользоваться событием *"ПередЗаписью"* объекта документ. Где его найти? Вкладка *"Прочее"* свойств документа, *"МодульОбъекта"*.

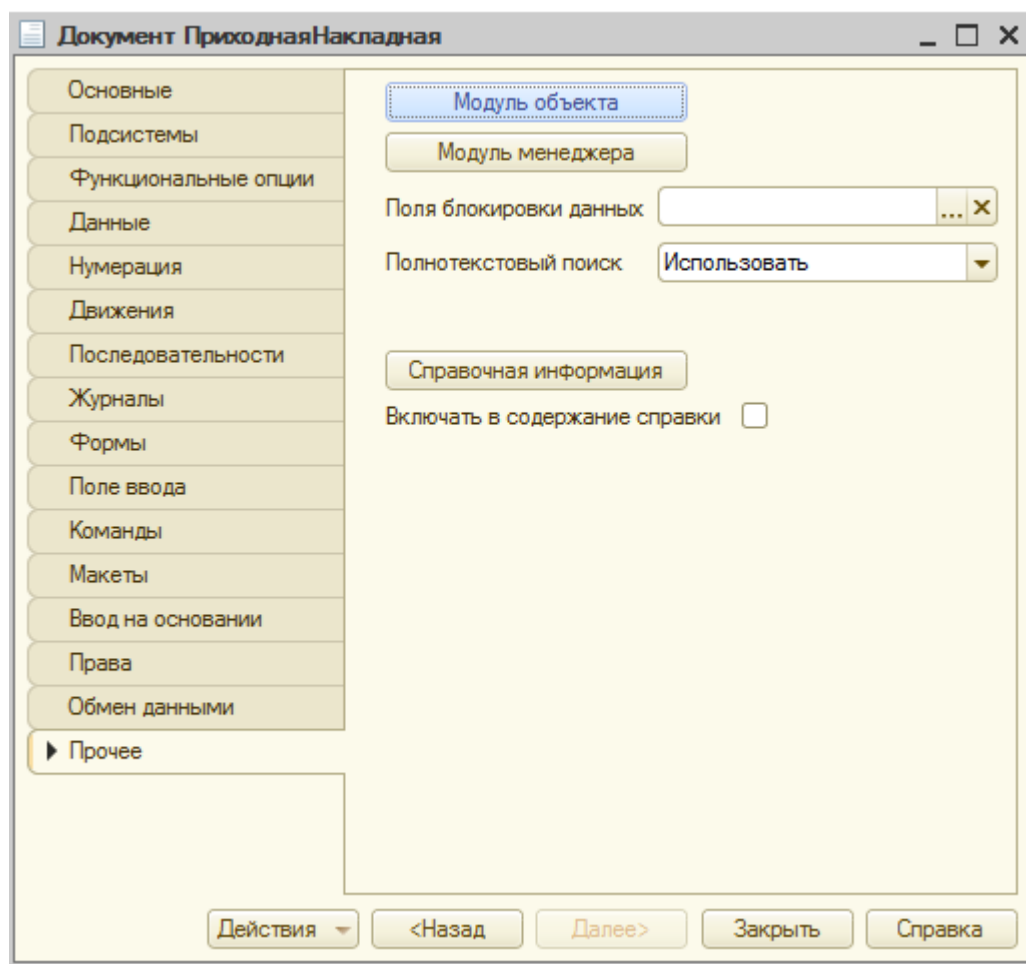


Рисунок 1- Обработчик события ПередЗаписью объекта

Обработчик можно добавить с помощью помощника, расположенного на верхней панели инструментов.

Содержание обработчика напишем сами, активно используя контекстную подсказку (листинг). Все должно работать, как и заказывали.

Листинг. Перед записью объекта «Документ»

Процедура ПередЗаписью(Отказ, РежимЗаписи. РежимПроведения)

Если Склад.Пустая() Тогда

Возврат;

КонецЕсли;

Для каждого Стр Из Товары Цикл

Стр.Склад = Склад;

КонецЦикла;

КонецПроцедуры

Обратите внимание, что при работе с событиями формы мы использовали директивы *«НаСервере»*, *«НаКлиенте»* и др. Здесь их нет. Связано это с тем, что весь модуль объекта всегда выполняется на сервере.

Второе. На что нужно обратить внимание – название процедур-обработчиков. При работе с обработчиками событий элементов формы, мы могли назвать процедуру так, как нам нравится. Важно было лишь то, чтобы именно это (возможно и нами придуманное название) было «привязано» в свойствах элемента формы к нужному событию. При работе модуля объекта привязать процедуру не к чему. Единственное, каким образом программа догадается, что процедура *«ПередЗаписью»* должна быть выполнена именно «перед записью» документа, это по ее названию. Если название изменить, она или не будет выполнена вообще, или будет выполнена при наступлении другого события.

Условное оформление формы

Теперь немного «про красоту». Если пользователь заполнил склад в «шапке» документа, то колонку со складами в табличной части надо скрыть. И наоборот, если склад в «шапке» очистил – колонку в табличной части надо сделать видимой. Это можно сделать и программно, и декларативно. Когда есть выбор, надо выбрать декларативный вариант.

Если форма документа *«РеализацияТоваров»* еще не создана, создайте форму документа. Чтобы получить доступ к декларативному управлению формой, откройте свойства корня дерева элементов в левом окне «Форма» (рисунок).

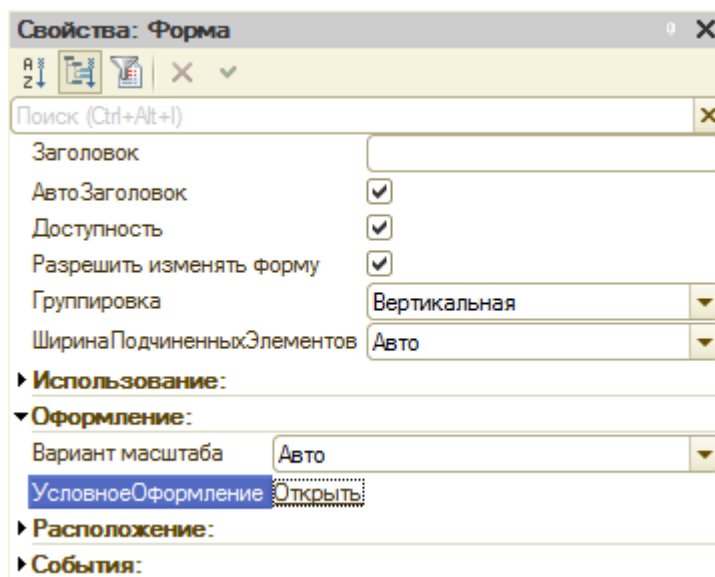


Рисунок 2- Свойства «Формы»

В разделе «Оформление» откройте «Условное оформление» и настройте его в соответствии с рисунком (рисунок).

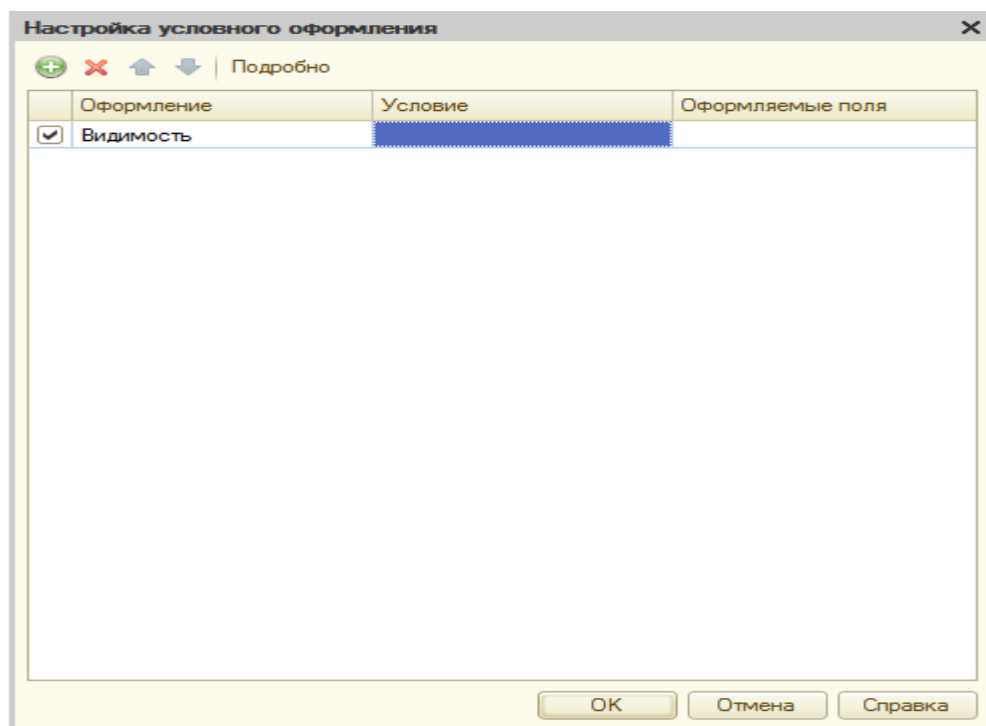


Рисунок 3- Условное оформление формы

Таким образом, если склад в "шапке" заполнен, то видимость колонки *ТоварыСклад* равна Ложь.

Проверьте работу механизма.

Лабораторная работа № 14. Создание обработок.

Цель работы: Создание внешней обработки. Использование внешней обработки.

Порядок выполнения работы:

Создание обработок

Этот объект дерева конфигурации мы не проходили, когда изучали конфигурирование. Дело в том, что без программирования он просто не имеет смысла. Обработка - это оболочка, в которой находится программный код, который, по идее, должен как-то изменять данные. Групповое удаление, проведение, обмен данными - это все обработки.

Теперь, когда мы начали программировать, можно попытаться свои силы в написании обработок.

Одна из отличительных особенностей обработки состоит в том, что, как и отчет, она может быть внешней. Таким образом, мы можем не изменять конфигурацию, если нам необходимо написать какую-то групповую обработку данных. Мы так и поступим - мы создадим внешнюю обработку, которая будет изменять документы.

Тренироваться будем на документе *"ПоступлениеТоваров"*, только по тому, что у нас их больше в базе данных. Мы более обстоятельно занимались приходными накладными и отчетом о закупках, чем расходными и отчетом о продажах.

Посмотрите на список документов *"ПоступлениеТоваров"*. Нумерация у документов сквозная, причем, автор, например, вводил документы задним числом, поэтому получилось не очень красиво.

Нужно добиться, чтобы нумерация была отдельной по каждой организации и в порядке дат документов.

Решить задачу раздельной нумерации по организациям поможет префикс номера. Это одна-две-три буквы (краткое название организации), с которого будет начинаться номер. Таким образом, программа будет поддерживать раздельную нумерацию по каждому значению префикса.

В качестве префикса мы будем использовать первую букву названия организации, "ф"- для Фабрика и "Т"- для Торговый дом. Чтобы можно было использовать префикс номера, нужно чтобы номер был строковым - у нас именно так.

Если документ новый, то задать логику формирования префикса поможет обработчик события, который располагается в модуле объекта "Документ" (листинг).

Листинг. Обработчик события "При установке нового номера"

```
Процедура ПриУстановкеНовогоНомера(СтандартнаяОбработка, Префикс)  
Префикс=Лев(Организация.Наименование, 1);  
КонецПроцедуры
```

Обработчик события возникает при записи нового документа, т.е. перезапись старых документов его не вызовет. Но если создать (скопировать) и записать новый документ, то он будет выполнен.

Как он работает, вам расскажет синтакс-помощник (Ctrl+F1). Если мы присвоим внутри обработчика параметру "Префикс" какое-то значение, то программа будет присваивать следующий номер документу по значению выбранного префикса.

Для того чтобы "отрезать" от наименования организации один (левый) первый символ, мы использовали функцию "Лев".

А что делать с документами, которые уже записаны в базу данных, как поменять нумерацию у них?

Придется писать обработку. Причем, т.к. мы исправили модуль объекта документ и дальше все документы нумеруются правильно, то обработка эта будет "разовая" - один раз выполнить и "выбросить". Сделаем ее внешней.

Создание внешней обработки

Используя меню "Файл/Новый" configurатора создадим внешнюю обработку (рисунок).

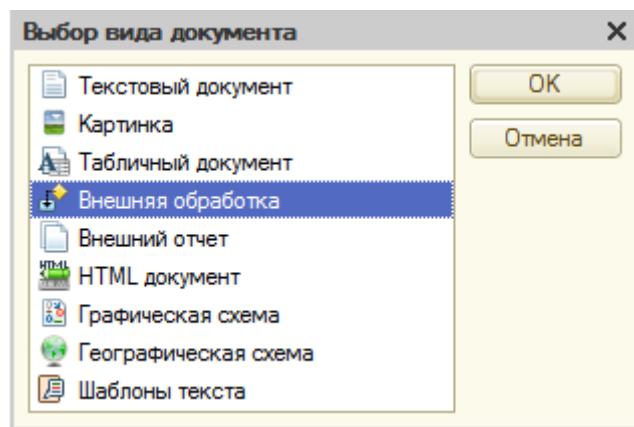


Рисунок 1- Меню выбора вида создаваемого документа

Диалог свойств внешней обработки немного отличается от обработки встроенной, но лишь внешне (рисунок). Состав свойств совпадает.

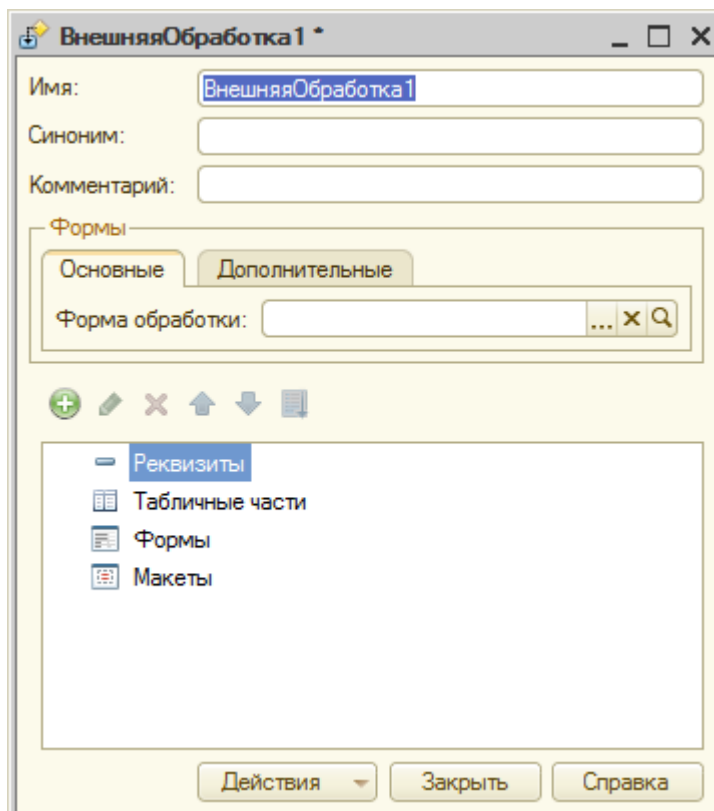


Рисунок 2- Главное окно внешней обработки

Что такое обработка? Окно, в нем кнопка, к которой привязана процедура. Это самый простой случай. У нас именно так и будет. Создадим форму (совершенно пустую) нашей обработки.

Создание формы обработки

Если мы хотим расположить в этой форме кнопку, надо добавить команду формы и эту команду перенести в элементы формы.

Написание модуля обработки

К действию кнопки должна быть привязана процедура. Создать и привязать ее можно нажатием одной кнопки "Открыть" напротив "Действия". И вот тут надо задуматься...

Где находимся мы? На клиенте! А где находится таблица документов, строки которой (номера) мы собираемся изменять? Правильно - на сервере! Поэтому нам обязательно нужна связка из двух процедур: одна здесь на клиенте привязана к кнопке диалога, вызывает вторую, расположенную на сервере. Вызывает с передачей контекста или без? Без контекста лучше. Тут контекста-то - "кот наплакал" - одна кнопка! Вывод: сделаем с контекстом. Так будет проще - если вдруг мы решим усложнить обработку, добавить в диалог интервал дат, например, за который выбирать документы, не нужно будет думать, как их передать на сервер. Обработка захватит с собой весь контекст формы, включая даты.

Напишем содержимое обработчика, активно используя контекстную подсказку и синтакс-помощник (листинг).

Листинг. Модуль формы обработки

```
&НаКлиенте
Процедура Сделать(Команда)
СделатьНаСервере();
КонецПроцедуры
&НаСервере
Процедура СделатьНаСервере()
Выборка=Документы.ПоступлениеТоваров.Выбрать();
Пока Выборка.Следующий() Цикл
Префикс=Лев(Выборка.Организация.Наименование,1);
ДокОбъект=Выборка.ПолучитьОбъект();
ДокОбъект.УстановитьНовыйНомер(Префикс); ДокОбъект.Записать();
КонецЦикла;
КонецПроцедуры
```

Прокомментируем модуль. Конструкцию-связку из двух процедур за нас написала платформа, спасибо ей. Содержимое второй процедуры мы писали сами.

Что такое выборка? Специфический объект, который содержит в себе таблицу с ссылками на все, в нашем случае, документы "ПоступлениеТоваров". То же самое можно было сделать с помощью запроса. Здесь все равно выполняется запрос, только в случае с запросом мы бы смогли установить любые произвольные условия отбора, сортировки и т.д. Здесь тоже можно, но намного меньше.

Строки табличной части принято обходить в цикле "Для каждого". Строки выборки принято обходить циклом "Пока". Причем внутри цикла та же переменная "Выборка"- это уже не табличка с ссылками, а одна ссылка из этой таблички.

Имея эту строку, мы можем получить из ссылки любые реквизиты, но пока не можем ничего изменить. Только посмотреть...

А чтобы изменить, мы должны получить ("захватить") объект документа по его ссылке. Используя метод "*УстановитьНовыйНомер*" устанавливаем номер с учетом префикса и записываем документ обратно в базу данных.

Записываем обработку (например, на диск) и проверяем, как она работает.

Использование внешней обработки

Открываем обработку (рисунок), нажимаем единственную кнопку "Сделать" и смотрим на результаты.

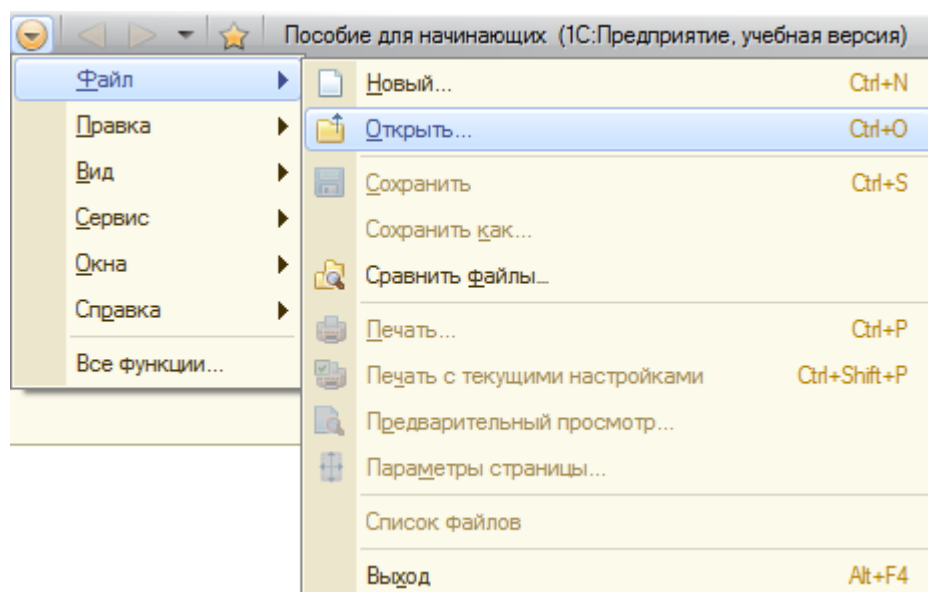


Рисунок 3- Открытие внешней обработки

А если у вас результатов никаких, ничего в нумерации документов не изменилось - ничего страшного. Значит, список документов уже был открыт. И надо или его закрыть и открыть заново, или обновить *"Еще /Обновить"*, чтобы список документов был заново получен из базы данных с новыми номерами.

ЗАДАНИЕ:

Доработайте функциональность нашей обработки:

- 1 Пользователь должен иметь возможность выбрать интервал дат, за которые нужно перезаписать документы.
- 2 Должны обрабатываться только документы, не помеченные на удаление.
- 3 В идеале, данные о документах должны получаться запросом, а не в объектной модели.