Федеральное государственное бюджетное образовательное учреждение
высшего образования «Кемеровский государственный университет»
Новокузнецкий институт (филиал)

Факультет информатики, математики и экономики
Кафедра информатики и вычислительной техники им. В. К. Буторина

О. А. Штейнбрехер

Информационные системы и технологии. Часть 2

Методические указания по выполнению практических работ дисциплине «Информационные системы и технологии» для обучающихся по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем Профиль «Программное и математическое обеспечение информационных технологий»

Новокузнецк
2020

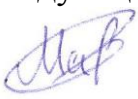
УДК [378.147.88:004.4](072)
ББК 74.484(2Рос-4Кем)я73+ 32.972я73
Ш88

Ш88 «Информационные системы и технологии. Методические указания по выполнению практических работ» : метод. указ (текст. электрон. изд.)/ О.А. Штейнбрехер ; Новокузнец. ин-т (фил.) Кемеров. гос. ун-та – Новокузнецк: НФИ КемГУ, 2020. – 87 с.

Приводятся методические указания по выполнению практических работ второго семестра дисциплины «Информационные системы и технологии».

Методические указания предназначены для студентов всех форм обучения направлений 02.03.03 «Математическое обеспечение и администрирование информационных систем».

Рекомендовано
на заседании кафедры
информатики и вычислительной
техники им. В. К. Буторина
17 сентября 2020 года.
Заведующий кафедрой



А. В. Маркидонов

Утверждено
методической комиссией факультета
информатики, математики и экономики
24 сентября 2020 года.
Председатель методкомиссии

Г.Н. Бойченко

УДК [378.147.88:004.4](072)
ББК 74.484(2Рос-4Кем)я73+ 32.972я7

Текст представлен в авторской редакции

Оглавление

Раздел 1. Принципы конфигурирования и основные объекты 1С: Предприятие

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Общие принципы работы в 1С: Предприятие

Система программ «1С: Предприятие 8» включает в себя платформу и прикладные решения, разработанные на ее основе, для автоматизации деятельности организаций и частных лиц. Сама платформа не является программным продуктом для использования конечными пользователями, которые обычно работают с одним из многих прикладных решений (конфигураций), разработанных на данной платформе. Такой подход позволяет автоматизировать различные виды деятельности, используя единую технологическую платформу.

Основная задача платформы заключается в повышении уровня абстракции при разработке и использовании прикладных решений. Это позволяет перейти от технических и низкоуровневых понятий к более содержательным и высокоуровневым. Позволяет приблизить эти понятия к языку пользователей и специалистов в предметной области. В конечном итоге это значительно ускоряет и унифицирует разработку прикладного решения и его сопровождение.

Создание конфигурации

Для работы с конфигурацией требуется создать информационную базу: в окне «Запуск 1С: Предприятие» выбрать кнопку «Добавить», в форме «Добавление информационной базы/группы» выбрать «Создание новой информационной базы», затем «Создание информационной базы без конфигурации для разработки новой конфигурации или загрузки выгруженной ранее информационной базы», затем указать наименование и каталог информационной базы. Для работы с конфигурацией требуется выбрать ее в перечне информационных баз и открыть в режиме Конфигуратор, затем в меню «Конфигурация» выбрать пункт «Открыть конфигурацию». В рабочем окне откроется дерево конфигурации (рисунок 1.1), в котором будут расположены все созданные объекты конфигурации.

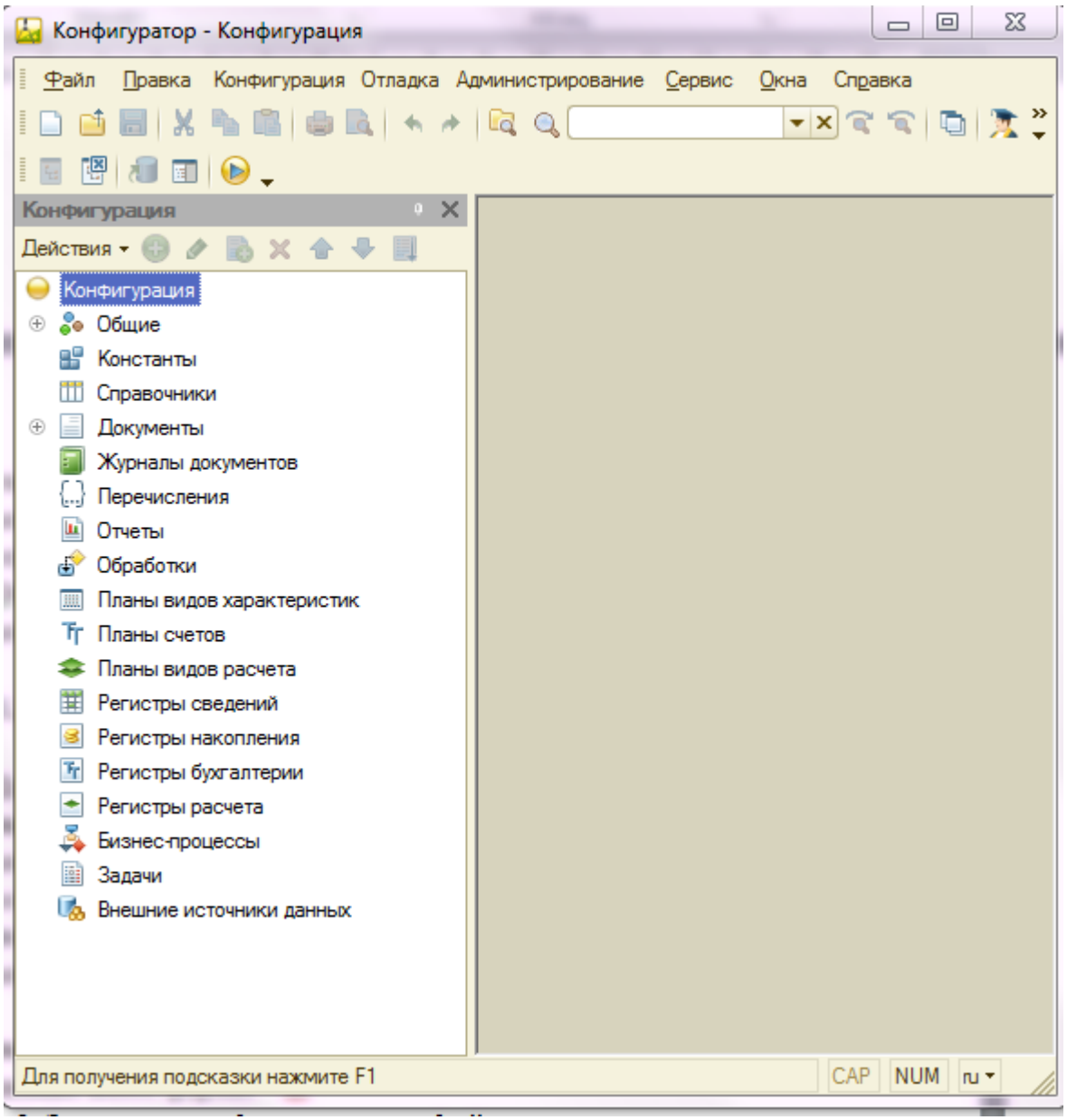


Рисунок 1.1 – Окно конфигуратора, дерево конфигурации

Сохранение конфигурации и информационной базы в файл

Прикладное решение состоит из конфигурации и информационной базы. Для того чтобы открыть конфигурацию на другом рабочем месте требуется выгрузить конфигурацию и информационную базу в файлы. Информационная база содержит в себе шаблон конфигурации.

Для сохранения конфигурации в файл в меню «*Конфигурация*» выбирается пункт «*Сохранить конфигурацию в файл*», затем выбирается путь сохранения. Файл конфигурации имеет расширение *.cf. В меню «*Конфигурация*» можно также сравнить текущую конфигурацию с выгруженной ранее и объединить их. Инструменты для управления поставкой конфигурации и обновления конфигурации так же находятся в данном меню.

Для выгрузки информационной базы в файл в меню «*Администрирование»* выбирается пункт «*Выгрузить информационную базу»*. Файл выгружаемых данных имеет формат *.dt. Следует заметить, что выгрузка информационной базы невозможна при работе конфигурации в пользовательском режиме (в том числе при отладке конфигурации).

Загрузка конфигурации из файла

Для открытия конфигурации на другом рабочем месте в пустой конфигурации (созданной заново) в меню «*Конфигурация»* выбирается пункт «*Загрузить конфигурацию из файла»* и выбирается ранее выгруженный файл расширения *.cf. После загрузки конфигурации загружается информационная база. Для этого в меню «*Администрирование»* выбрать «*Загрузить информационную базу»*. При этом происходит сравнение текущей конфигурации и конфигурации информационной базы.

Если есть доступ к каталогу, в котором содержатся все файлы конфигурации (каталог, который был указан при создании базы), то можно выбрать этот каталог при создании новой конфигурации. Для этого в форме «*Добавление информационной базы/группы»* выбрать *«Добавление в список существующей информационной базы»* и указать каталог расположения.

Основные объекты конфигурации

Платформа 1С: Предприятие обладает следующими стандартными видами объектов: константы, перечисления, справочники, документы, регистры и т.д. В данном разделе рассматриваются следующие объекты: справочники, перечисления, константы и документы.

Все объекты конфигурации имеют свойство «*Имя*» и «*Синоним*». «*Имя*» объекта имеет ограничения: оно не должно содержать пробелы и не должно начинаться с символа или цифры. «*Синоним*» отображается в режиме пользователя на формах.

Сложные объекты, такие как справочники, документы, регистры и так далее имеют поля – реквизиты. Рассматривая структуру конфигурации как базу данных – объекты базы являются таблицами базы, а реквизиты объектов – полями таблицы. При этом константы и перечисления являются таблицами, состоящими из одного поля. Реквизиты объектов определяются пользователем, но каждый объект имеет несколько реквизитов по умолчанию.

Константы

Константы - это прикладные объекты конфигурации. Они позволяют хранить в информационной базе данные, которые не изменяются во времени,

или изменяются очень редко. Каждая константа позволяет хранить одно значение.

Например, в константе может храниться наименование предприятия, его ИНН и другая информация. В прикладном решении может быть создано произвольное количество констант.

Перечисления

Перечисления - это прикладные объекты конфигурации. Они позволяют хранить в информационной базе наборы значений, которые не изменяются в процессе работы прикладного решения. Например, это может быть перечисление состояния заказов (Запланировано, ВРаботе, Выполнено) и пр.

Справочники

Справочники - это прикладные объекты конфигурации. Они позволяют хранить в информационной базе данные, имеющие одинаковую структуру и списочный характер. Это может быть, например, список сотрудников, перечень товаров, список поставщиков или покупателей.

Каждый элемент справочника характеризуется реквизитами «К*од»* и «Н*аименование»*. Система поддерживает режим автоматической нумерации элементов, при котором она самостоятельно может генерировать код для нового элемента справочника. Кроме этого система позволяет осуществлять контроль уникальности кодов справочника, не разрешая создавать элементы с одинаковыми кодами.

Помимо кода и наименования, каждый элемент справочника, как правило, содержит некоторую дополнительную информацию, которая подробно описывает этот элемент. Например, для товара это может быть информация об артикуле, упаковке и т.п. Набор такой информации является одинаковым для всех элементов конкретного справочника, и для ее хранения служат реквизиты справочника.

Кроме этого, каждый элемент справочника может содержать некоторый набор информации, которая одинакова по своей структуре, но различна по количеству, для разных элементов справочника. Например, для каждого сотрудника в справочнике «Физические лица» это может быть контактная информация или информация о составе семьи, образовании. Для хранения подобных данных служат табличные части справочника.

Справочники могут поддерживать иерархическое расположение элементов. Существует два вида иерархии: иерархия групп и элементов и иерархия элементов. Иерархия групп и элементов предполагает наличие групп, в которых будут располагаться элементы, относящиеся к этим

группам. Например, в справочнике «Номенклатура» могут быть созданы группы: Бытовая техника, Обувь, Продукты и т.д. Примером справочника с иерархией элементов может служить справочник «Подразделения», где элементы справочника относятся не к группам, а к другим элементам.

Разные справочники могут находиться в состоянии подчинения, т.е. элементы одного справочника могут быть подчинены элементам или группам другого справочника. Например, справочник «Кассы» может быть подчинен справочнику «Организации». Тогда при оформлении кассовых документов для некоторой организации можно будет выбрать кассу не среди всех имеющихся в программе касс, а среди касс, существующих только в этой организации.

Использование иерархии и/или подчинения активизирует встроенные реквизиты «Родитель» и «Владелец», соответственно. Тип данных реквизитов является ссылочным, одно из значений – пустая ссылка. Таким образом, при заполнении элемента справочника (рисунок 1.2) определяется его подчинение конкретному объекту и положение в иерархии.

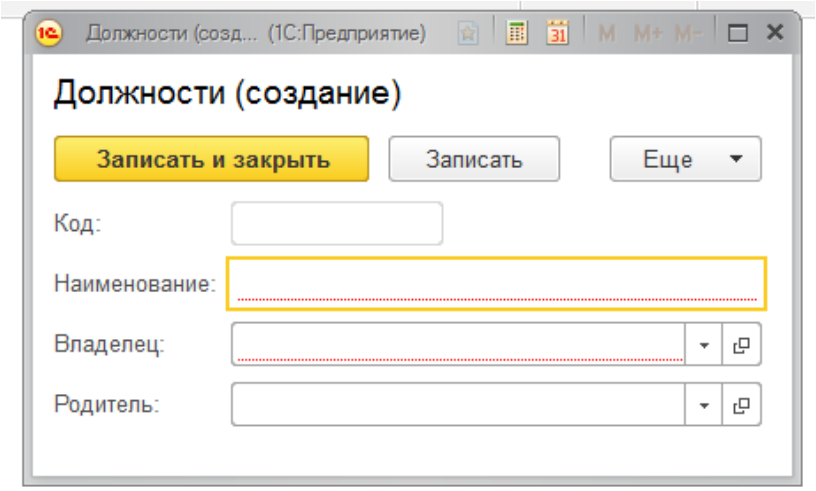


Рисунок 1.2 – Пример формы с назначением владельца и родителя

Справочники допускают также создание предопределенных элементов, которые существуют в справочнике всегда, вне зависимости от действий пользователя. Такие элементы справочника создаются разработчиком при разработке прикладного решения и не могут быть удалены или перемещены пользователем. Предопределенными элементами могут быть элементы справочника или группы. Предопределенные элементы могут использоваться при кодировании обработчиков событий.

Для создания предопределенных элементов нужно зайти на вкладку «Прочее» и выбрать «Предопределенные» или в меню «Действия» - «Открыть предопределенные данные» (рисунок 1.3).

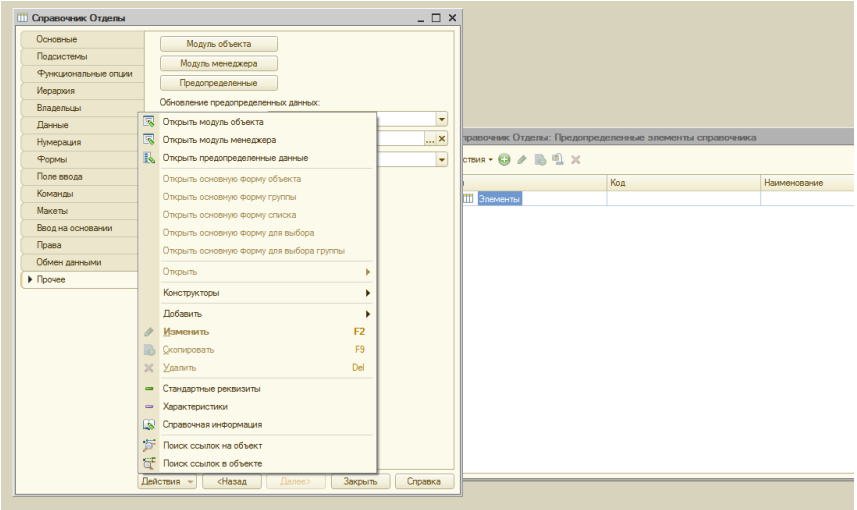


Рисунок 1.3 – Меню создания предопределенных элементов справочника

Документы

Документы - это прикладные объекты конфигурации. Они позволяют хранить в прикладном решении информацию о совершенных хозяйственных операциях или о событиях, произошедших в "жизни" предприятия вообще.

Каждый документ характеризуется номером, датой и временем. Система поддерживает режим автоматической нумерации документов, при котором она самостоятельно может генерировать номер для нового документа. Кроме этого система позволяет осуществлять контроль уникальности номеров документов, не разрешая создавать документы с одинаковыми номерами. Настройка параметров нумерации позволяет отслеживать уникальность, как среди всех элементов документа, так и с учетом периода. Важными характеристиками документа являются дата и время – обязательные стандартные реквизиты (рисунок 1.4). Они позволяют установить строгую временную последовательность совершения операций.

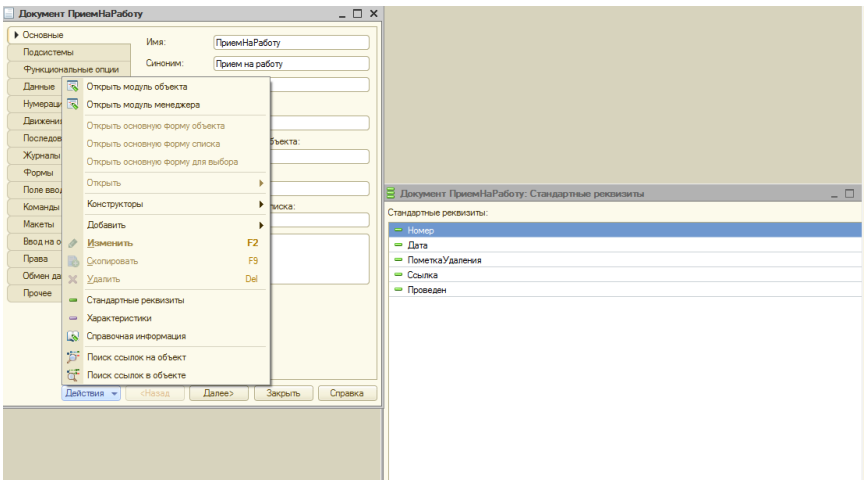


Рисунок 1.4 – Стандартные реквизиты документа

Кроме предопределенных реквизитов документ содержит некоторый набор информации, одинаковый по своей структуре. Кроме этого, каждый

документ может содержать некоторый набор информации, которая одинакова по своей структуре, но различна по количеству, для разных документов. Для хранения подобных данных служат табличные части документа.

Журнал Документов

Журналы документов - это прикладные объекты конфигурации. Они предназначены для просмотра документов разных видов. Для журнала документов могут быть определены графы, предназначенные для отображения реквизитов документов разного вида, отнесенных к данному журналу (рис. 1.5).

Дата	Номер	Тип документа	Контрагент	Расчетный счет
04.09.2012 12:00:00	000000020	Оплата	Животноводство О...	Основной
05.09.2012 11:14:05	000000019	Поступление денег	Магазин "Бытовая...	Вспомогательны
05.09.2012 12:10:05	000000013	Поступление денег	Магазин "Мясная ...	Основной
05.09.2012 13:46:50	000000014	Поступление денег	Попов Б.В. ИЧП	Дополнительный
06.09.2012 9:34:25	000000015	Поступление денег	Шлюзовая ООО	Вспомогательны
07.09.2012 12:00:00	000000023	Поступление денег	Шлюзовая ООО	Основной
10.09.2012 14:22:41	000000016	Поступление денег	Магазин "Продукты"	Вспомогательны
12.09.2012 15:32:00	000000017	Поступление денег	Попов Б.В. ИЧП	Основной

Рисунок 1.5 – Форма Журнала Документов

Типы данных

Платформа «1С: Предприятие» поддерживает собственную систему типов. Система типов — это особая система, по которой организуются данные, используемые прикладными решениями. Основной особенностью системы типов является то, что есть типы, существующие в любом прикладном решении. Сами эти типы определены на уровне платформы и присутствуют всегда, независимо от действий разработчика. Наряду с ними в конкретном прикладном решении могут существовать различные типы данных, присущие именно этому конкретному прикладному решению. Для таких типов данных на уровне платформы определены лишь общие правила их создания, шаблоны.

Можно выделить несколько основных категорий типов данных. Примитивные типы данных — это такие типы как Строка, Число, Дата, Булево и другие. Значения примитивных типов являются простыми неделимыми значениями, в которых нельзя выделить отдельные составляющие. Например, значениями типа Число могут быть 1, 8, 15 и др.

Также, существуют более сложные типы данных. Например, платформа поддерживает целый ряд типов, которые представляют собой универсальные коллекции значений: Массив, Структура, СписокЗначений и другие. Универсальные коллекции значений не используются для определения типов реквизитам объектов конфигурации.

Кроме этого в платформе реализованы специфические типы данных, реализующие ту или иную функциональность прикладных решений: ТекстовыйДокумент, ТабличныйДокумент, ХранилищеЗначения, ПостроительЗапроса и другие. Общие типы называют также общими объектами. Значения этих типов, в отличие от значений примитивных типов, представляют собой совокупность значений отдельных свойств объекта. Поэтому их называют экземплярами объектов.

Интерфейсные типы позволяют организовывать визуальное взаимодействие прикладного решения с пользователем. В основном это типы, связанные с работой форм и их элементов. Например, тип данных НастройкиФормы.

Однако, наряду с типами данных, которые определены на уровне платформы, конкретное прикладное решение может использовать уникальные типы данных, существующие только в этом конкретном прикладном решении. Как правило, появление новых типов данных в прикладном решении связано с использованием прикладных объектов конфигурации. Поэтому такие типы называют еще прикладными типами или прикладными объектами. На уровне платформы поддерживается несколько классов (шаблонов) прикладных объектов, которые сами по себе не могут быть использованы в конкретном прикладном решении. Например, можно перечислить такие классы прикладных объектов как Справочники, Документы, Регистры сведений, Планы видов характеристик и пр.

Например, разработчик может добавить в свое прикладное решение новый справочник «Номенклатура», который будет наследовать функциональность класса Справочники, или новый документ «КассовыйОтчет», который будет наследовать функциональность класса Документы.

Сразу же после такого добавления разработчику становятся доступны новые типы данных, состав которых определяется принадлежностью объекта конфигурации к тому или иному классу прикладных объектов.

Настройка пользовательского интерфейса

Подсистемы - это общие объекты конфигурации. На их основе платформа формирует командный интерфейс прикладного решения и визуально разделяет всю функциональность программы на крупные и мелкие блоки. Подсистемы могут иметь иерархическую структуру, т.е. одна подсистема может включать в себя несколько других подсистем. Каждый объект конфигурации можно включить в состав одной или нескольких подсистем. Таким образом, в терминах подсистем можно описать всю структуру прикладного решения.

Редактор командного интерфейса конфигурации - это один из инструментов разработки. Он предназначен для того, чтобы настроить порядок следования разделов в панели разделов и настроить видимость разделов для разных ролей, определенных в конфигурации. Вызвать редактор командного интерфейса конфигурации (рисунок 1.6) можно разными способами, например, командой контекстного меню в корне конфигурации.

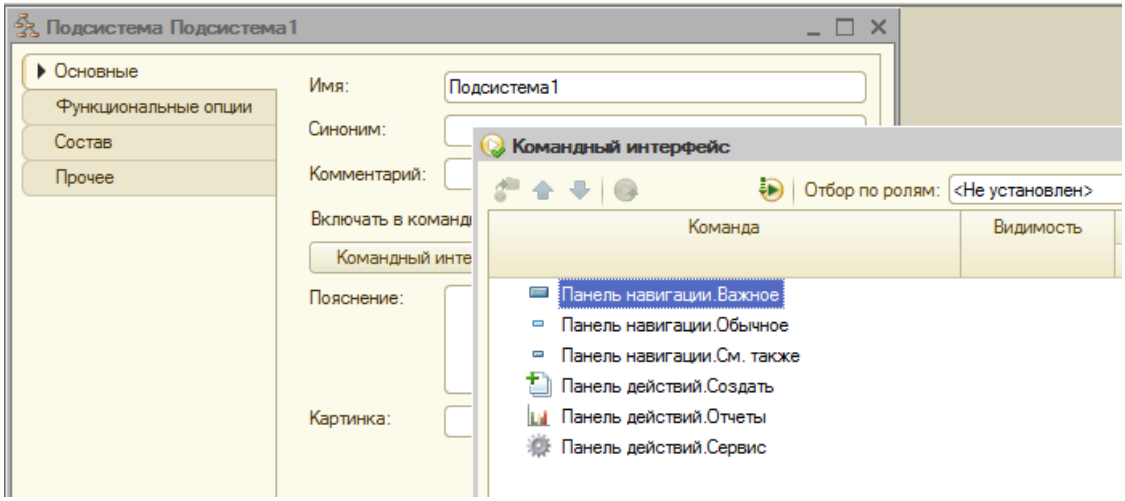


Рисунок 1.6 - формы редактирования Подсистем и Командный интерфейс подситем

Если подсистем много, а требуется настроить видимость только для некоторых из них, то подсистемы можно отобрать по ролям. При большом количестве ролей можно настроить состав колонок, отображаемых в редакторе. Для этого используется стандартная настройка списка, которая вызывается командой «*Настройка списка*» из контекстного меню.

Редактор командного интерфейса основного раздела - это один из инструментов разработки. Он предназначен для настройки команд основного раздела.Редактор позволяет настроить состав команд каждой командной панели, порядок отображения и видимость элементов командного интерфейса по ролям.

Редактор "Все подсистемы" - один из инструментов разработки. Он предназначен для решения следующих задач:

- редактирование командного интерфейса любой подсистемы,
- задание состава и свойств любой подсистемы,
- настройка порядка следования подсистем,
- редактирование состава подсистем.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №1

1. Создать информационную базу без конфигурации. Перейти в режим конфигуратора. Настроить параметры конфигурации – название, версию и т.д.
2. Создать подсистемы «Сотрудники», «Склад» и «Клиенты».
3. Для подсистем определить картинку отображения и свойства командного интерфейса.
4. Создать справочник «ФизическиеЛица». Определить реквизиты справочника:
    a. ФамилияСотрудника, тип: строка, длина: неограниченная, переменная
    b. ИмяСотрудника, тип: строка, длина: неограниченная, переменная
    c. ОтчествоСотрудника, тип: строка, длина: неограниченная, переменная
    d. ДатаРождения, тип: Дата
    e. Образование, тип: строка, длина: неограниченная, переменная
    f. СемейноеПоложение, тип: строка, длина: неограниченная, переменная
    g. Табличная часть – Дети
        i. ГодРождение, тип: число
        ii. Пол, тип: строка
        iii. Имя, тип: строка
        iv. Отчество, тип: строка
        v. Фамилия, тип: строка
    h. Табличная часть – КонтактныеДанные
        i. ВидСвязи, тип: строка
        ii. Значение, тип: строка

5. Создать справочник «Сотрудники». Определить справочник, как иерархический с иерархией групп. Определить реквизиты справочника:
    a. ФамилияСотрудника, тип: строка, длина: неограниченная, переменная
    b. ИмяСотрудника, тип: строка, длина: неограниченная, переменная
    c. ОтчествоСотрудника, тип: строка, длина: неограниченная, переменная
    d. Отдел, тип: строка, длина: неограниченная, переменная
    e. Должность, тип: строка, длина: неограниченная, переменная
6. Для удобства работы со справочниками создать необходимые перечисления. Изменить в справочниках тип данных соответсвующих полей (ПеречислениеСсылка.<НаименованиеПеречисления>).
7. Для связи между справочниками определить в справочнике «Сотрудники» реквизит для ссылки на справочник «ФизическиеЛица». Предусмотреть механизм ввода на основании из справочника «ФизическиеЛица».
8. Для автоматизации выбора должности в справочнике «Сотрудники», добавить в конфигурацию объект справочник «ПодразделенияОрганизации».
9. Для автоматизации выбора должности в справочнике «Сотрудники», добавить в конфигурацию объект справочник «ДолжностиОрганизации».
10. В справочнике «Сотрудники» изменить тип реквизита «Должность» – вместо текстового выбрать СправочникСсылка.ДолжностиОрганизации.
11. Для справочника «Должности» назначить владельца – справочник «ПодразделенияОрганизации»
12. Определить для справочников «Должности» и «ПодразделенияОрганизации» предопределенные значения.
13. Определить подсистемы для созданных объектов.
14. В пользовательском режиме ввести данные в справочнике – не менее 5 записей.
15. Создать документ «ПриемНаРаботу». Определить реквизиты документа:
    a. Должность, тип: СправочникСсылка.Должности
    b. Отдел, тип: СправочникСсылка.ПодразделенияОрганизации

c. ДатаПриема, тип: дата
d. Фамилия, тип: строка, длина: неограниченная, переменная
e. Имя, тип: строка, длина: неограниченная, переменная
f. Отчество, тип: строка, длина: неограниченная, переменная
g. ДатаРождения, тип: дата
h. Табличная часть – Контакты
    i. ТипСвязи, тип: строка
    ii. Номер, тип: строка
i. Табличная часть – Зарплата
    i. ТипВыплаты, тип: строка
    ii. Сумма, тип: число
    iii. Периодичность, тип: строка

16. Для удобства работы с документом создать необходимые перечисления. Изменить тип данных соответсвующих полей (ПеречислениеСсылка.<НаименованиеПеречисления>).
17. Добавить документ в подсистему «Сотрудники».
18. Для документа «ПриемНаРаботу» на вкладке «*Ввод на основании*» в окне «*Является основанием*» для добавить Справочник.Сотрудники.
19. В справочнике «Сотрудники» на вкладке «*Ввод на основании*» вызвать Конструктор ввода на основании и заполнить его (рисунок 1.7).

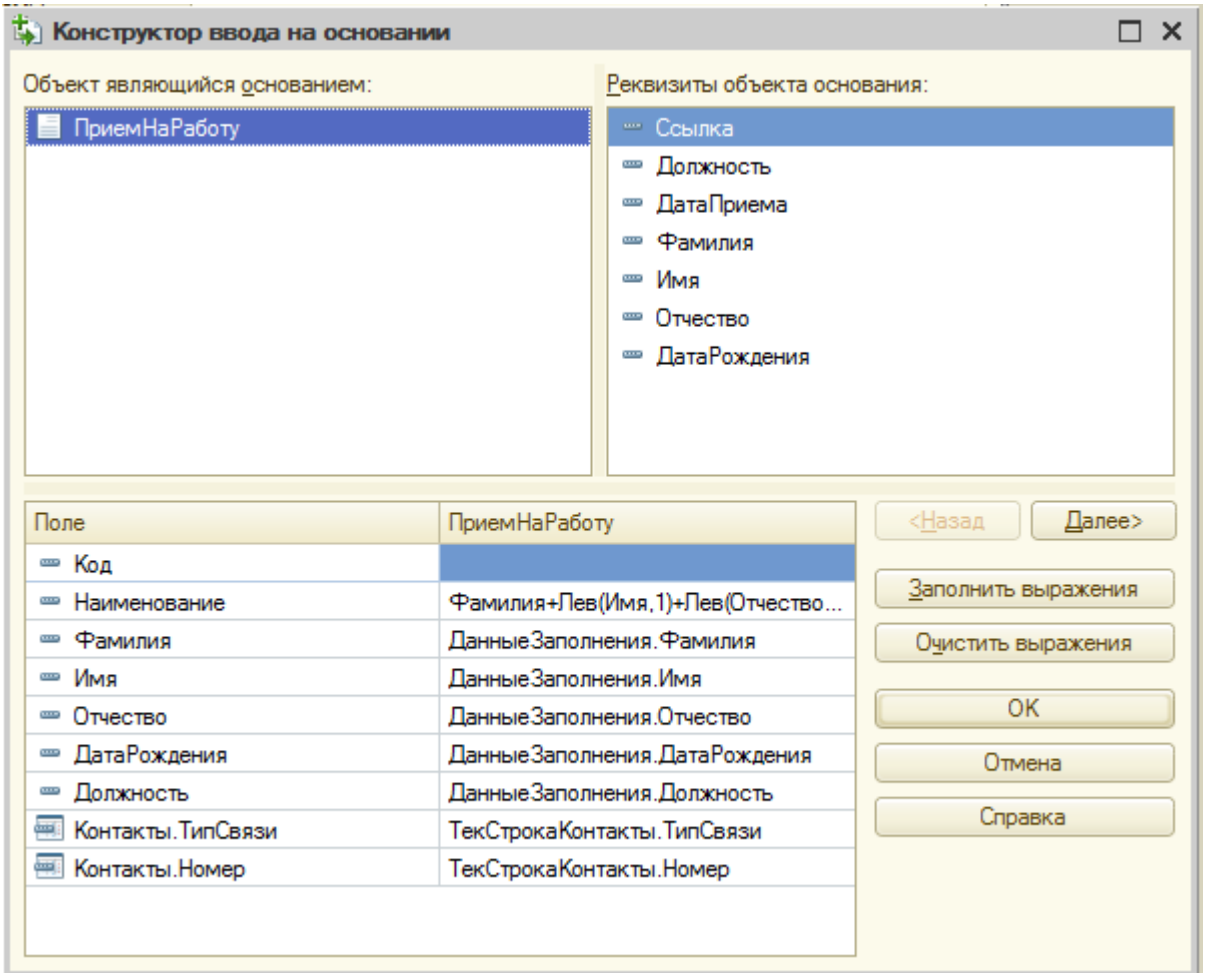


Рисунок 1.7 – Модуль *Конструктора ввода на основании*

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Дайте определение понятию «конфигурирование».
2. Что такое информационная база в платформе «1С: Предприятие»?
3. Каким образом выгрузить информационную базу в файл?
4. Как сравнить две конфигурации?
5. Что такое объект конфигурации Справочник?
6. Какие реквизиты по умолчанию есть в объекте Справочник?
7. Что такое предопределенные элементы в Справочнике?
8. Каким образом создаются подчиненные Справочники?
9. Что такое Форма элемента Справочника?
10. Чем отличается иерархия элементов и иерархия групп и элементов в Справочнике?
11. Чем отличается объект Перечисления от объекта Справочник?
12. Зачем используется объект Константа в прикладном решении?
13. Как обратиться к Константе?
14. Что такое Подсистема?
15. Что такое Командный интерфейс подсистемы?

16.Каким образом настраивается интерфейс Подсистемы?
17.Какие реквизиты характеризуют экземпляр объекта Документ?

Раздел 2. Встроенный язык и управляемые формы

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Встроенный язык 1С: Предприятие

Встроенный язык является важной частью технологической платформы «1С:Предприятие 8», поскольку позволяет разработчику описывать собственные алгоритмы функционирования прикладного решения.

Встроенный язык имеет много общих черт с другими языками, такими как Pascal, JavaScript, Basic, что облегчает его освоение начинающими разработчиками. Однако он не является прямым аналогом какого-либо из перечисленных языков.

Вот лишь некоторые, наиболее значимые особенности встроенного языка:

- предварительная компиляция — перед исполнением модули, содержащие текст на встроенном языке, преобразуются во внутренний код;
- кэширование скомпилированных модулей в памяти;
- мягкая типизация — тип переменной определяется типом значения, которое она содержит, и может изменяться в процессе работы;
- отсутствие программного описания объектов конфигурации — разработчик может использовать либо встроенные в платформу объекты, либо объекты, созданные системой в результате визуального конструирования прикладного решения.

Встроенный язык поддерживает двойной синтаксис – на русском и английском языках. Каждый объект и оператор имеет два аналога. Можно использовать и русский язык, и английский, и комбинацию двух языков. Но существуют рекомендации использовать русский язык, потому что в системе достаточно большое количество объектов, которые имеют длинные названия.

Прикладные решения в 1С:Предприятие 8 не кодируются целиком. Большая часть прикладного решения создается разработчиком путем визуального конструирования — создания новых объектов конфигурации, задания их свойств, форм представления, взаимосвязей и пр. Встроенный язык используется лишь для того, чтобы определить поведение объектов прикладного решения, отличное от типового, и создать собственные алгоритмы обработки данных.

По этой причине модули, содержащие текст на встроенном языке, используются системой в конкретных, заранее известных ситуациях, которые могут возникнуть в процессе работы прикладного решения. Такие ситуации называются событиями. События могут быть связаны с функционированием объектов прикладного решения или с самим прикладным решением, как таковым. Например, с функционированием объекта прикладного решения Справочник связан ряд событий, среди которых есть событие ПередЗаписью. Это событие возникает непосредственно перед тем, как данные элемента справочника должны быть записаны в базу данных. Разработчик, используя встроенный язык, может описать алгоритм, который, например, будет проверять корректность данных, введенных пользователем.

Таким образом, можно сказать, что встроенный язык является скриптовым языком для программирования бизнес-логики, а использование модулей на встроенном языке является событийно-зависимым, т. е. выполнение модулей происходит при возникновении определенных событий в процессе функционирования прикладного решения.

Переменные

каждый модуль описывает поведение конфигурации в определенной точке. В модуле содержится, прежде всего, раздел описания переменных. Т.е. мы можем объявить в модуле некоторые переменные.

В дальнейшем они могут быть использованы в процедурах и функциях этого модуля. Если переменная определена с ключевым словом Экспорт, то она будет доступна вне данного модуля. Пример строки объявления переменных:

```
Перем Склад, Подразделение, Кладовщик Экспорт;
```

После объявления переменных содержится раздел процедур и функций.

За ними располагается раздел основной программы, который будет выполняться в момент обращения к данному модулю.

Разделителем операторов является символ «;» (точка с запятой). Этот знак является признаком окончания оператора. Точку с запятой можно не ставить в завершающем операторе данной конструкции, например, процедуры.

Переменные предназначены для того, чтобы хранить некоторые значения любого типа данных. Они используются для промежуточного хранения информации, для обработки. 1С: Предприятие поддерживает мягкую типизацию, поэтому переменная может содержать значения различных типов данных.

Описывать переменные можно двумя способами:

- неявный способ (упоминание в левой части оператора присваивания описывает данную переменную, нет предварительного описания переменной со словом Перем, т.е. нет специального раздела описания переменных);
- явное описание переменных (Перем КонтрольныеДанные;). Явное описание переменных используется, например, если предполагается последующая передача этой переменной в функцию.

Для названия переменных используется классическое описание идентификатора. Идентификатор состоит из букв, цифр и знаков подчеркивания. Начинаться идентификатор должен либо с буквы, либо со знака подчеркивания.

Операторы языка

Операторы языка можно разделить на три группы:

1. Операторы определения переменных – выполняют объявление переменной (присваивают ей символьное имя) и ее инициализацию некоторыми данными, чтобы другие операторы могли их использовать. Символьное имя переменной называется идентификатором, а содержащиеся данные – значением;
2. Управляющие операторы – это операторы, которые сами не производят каких-либо вычислений, но управляют этим процессом. К ним относятся операторы ветвления, операторы циклической обработки и операторы передачи управления;
3. Исполняемые операторы – это операторы, выполняющие непосредственную обработку данных. К ним относят системные процедуры и функции, заложенные разработчиками платформы, а также процедуры и функции, описанные разработчиком прикладной задачи (конфигурации). Все исполняемые операторы характеризуются наличием идентификатора и, возможно, дополнительных параметров.

Оператор присваивания

Оператор присваивания = (равно) используется для инициализации переменных, реквизитов объектов или элементов массива некоторыми значениями, а также для переопределения их значений при последующих упоминаниях. В качестве источника значений могут выступать: символьная

константа, имя переменной, имя реквизита объекта агрегатного типа, элемент массива или выражения с их участием.

Первое упоминание имени переменной в левой части оператора присваивания считается неявным определением переменной. Неявное определение переменных используется в том случае, когда область видимости переменной не важна и ограничена текущим контекстом выполнения.

Оператор присваивания = имеет следующий синтаксис:

***Получатель* = *Источник*;**

А также альтернативный англоязычный синтаксис:

***Destination* = *Source*;**

Операторы ветвления

Оператор ***Если*** управляет выполнением программы, основываясь на результаты одного или более логических выражений. Оператор может содержать любое количество групп операторов, возглавляемых конструкциями ***ИначеЕсли – Тогда***.

Синтаксис:

```
Если <Логическое выражение> Тогда
// Операторы
[ИначеЕсли <Логическое выражение> Тогда]
// Операторы
[Иначе]
// Операторы
КонецЕсли;
```

Англоязычный синтаксис:

```
If <Логическое выражение> Then
// Операторы
[ElsIf <Логическое выражение> Then]
// Операторы
[Else]
// Операторы
EndIf;
```

Параметры:

<Логическое выражение> – логическое выражение.

Тогда – операторы, следующие за ***Тогда***, выполняются, если результатом логического выражения является значение ***Истина***.

// Операторы – исполняемый оператор или последовательность таких операторов.

ИначеЕсли – логическое выражение, следующее за ключевым словом ***ИначеЕсли***, вычисляется только тогда, когда условия в ***Если*** и всех предшествующих ***ИначеЕсли*** оказались равны ***Ложь***. Операторы, следующие за конструкцией ***ИначеЕсли*** – ***Тогда***, выполняются, если результат логического выражения в данном ***ИначеЕсли*** равен ***Истина***.

Иначе – Операторы, следующие за ключевым словом ***Иначе***, выполняются, если результаты логических выражений в конструкции ***Если*** и всех предшествующих конструкциях ***ИначеЕсли*** оказались равны ***Ложь***.

КонецЕсли– Ключевое слово, которое завершает структуру оператора условного выполнения.

Операторы цикла

Оператор цикла ***Для*** предназначен для циклического повторения операторов, находящихся внутри конструкции ***Цикл – КонецЦикла***. Перед началом выполнения цикла значение ***<Выражение 1>*** присваивается переменной ***<Имя переменной>***. Значение ***<Имя переменной>*** автоматически увеличивается при каждом проходе цикла. Величина приращения счетчика при каждом выполнении цикла равна 1. Цикл выполняется, пока значение переменной ***<Имя переменной>*** меньше или равно значению ***<Выражение 2>***. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла.

Синтаксис:

```
Для <Имя переменной> = <Выражение 1> По <Выражение 2> Цикл
// Операторы
[Прервать;]
// Операторы
[Продолжить;]
// Операторы
КонецЦикла;
```

Англоязычный синтаксис:

```
For <Имя переменной> = <Выражение 1> To <Выражение 2> Do
// Операторы
[Break;]
// Операторы
[Continue;]
// Операторы
EndDo;
Параметры:
```

<Имя переменной> – идентификатор переменной (счетчика цикла), значение которой автоматически увеличивается на 1 при каждом повторении цикла. Так называемый счетчик цикла.

<Выражение 1> – числовое выражение, которое задает начальное значение, присваиваемое счетчику цикла при первом проходе цикла.

По – Синтаксическая связка для параметра ***<Выражение 2>***

<Выражение 2> – максимальное значение счетчика цикла. Когда переменная ***<Имя переменной>*** становится больше чем ***<Выражение 2>***, выполнение оператора цикла ***Для*** прекращается.

Цикл – операторы, следующие за ключевым словом ***Цикл***, выполняются, пока значение переменной ***<Имя переменной>*** меньше или равно значению ***<Выражение 2>***.

// Операторы – исполняемый оператор или последовательность таких операторов.

Прервать – позволяет прервать выполнение цикла в любой точке. После выполнения этого оператора управление передается оператору, следующему за ключевым словом ***КонецЦикла.***

Продолжить – немедленно передает управление в начало цикла, где производится вычисление и проверка условий выполнения цикла. Операторы, следующие в теле цикла за ним, на данной итерации обхода не выполняются.

КонецЦикла – ключевое слово, которое завершает структуру оператора цикла.

Оператор цикла ***Для каждого*** предназначен для циклического обхода коллекций значений. При каждой итерации цикла возвращается новый элемент коллекции. Обход осуществляется до тех пор, пока не будут перебраны все элементы коллекции, или может быть завершен досрочно при выполнении оператора ***Прервать***.

Синтаксис:

```
Для каждого <Имя переменной 1> Из <Имя переменной 2> Цикл
// Операторы
[Прервать;]
// Операторы
[Продолжить;]
// Операторы
КонецЦикла;
```

Англоязычный синтаксис:

```
For each <Имя переменной 1> In <Имя переменной 2> Do
// Операторы
[Break;]
// Операторы
[Continue;]
```

```
// Операторы
EndDo;
```

Оператор цикла ***Пока*** предназначен для циклического повторения операторов, находящихся внутри конструкции ***Цикл – КонецЦикла***. Цикл выполняется, пока логическое выражение равно ***Истина***. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла.

Синтаксис:

```
Пока <Логическое выражение> Цикл
// Операторы
[Прервать;]
// Операторы
[Продолжить;]
// Операторы
КонецЦикла
```

Англоязычный синтаксис:

```
While <Логическое выражение> Do
// Операторы
[Break;]
// Операторы
[Continue;]
// Операторы
EndDo;
```

Параметры:

<Логическое выражение> – логическое выражение

Цикл – операторы, следующие за ключевым словом ***Цикл***, выполняются, пока результат логического выражения равен ***Истина***.

// Операторы – Исполняемый оператор или последовательность таких операторов.

Прервать – позволяет прервать выполнение цикла в любой точке. После выполнения этого оператора управление передается оператору, следующему за ключевым словом ***КонецЦикла***.

Продолжить – немедленно передает управление в начало цикла, где производится вычисление и проверка условий выполнения цикла. Операторы, следующие в теле цикла за ним, на данной итерации обхода не выполняются.

КонецЦикла – ключевое слово, которое завершает структуру оператора цикла.

Процедуры и функции

Ключевое слово ***Процедура*** начинает секцию исходного текста, выполнение которого можно инициировать из любой точки программного

модуля, просто указав ***ИмяПроцедуры()*** со списком параметров (если параметры не передаются, то круглые скобки, тем не менее, обязательны). Если в модуле приложения или общем программном модуле в теле описания процедуры использовано ключевое слово ***Экспорт***, то это означает, что данная процедура является доступной из всех других программных модулей конфигурации.

При выполнении оператора ***Возврат*** процедура заканчивается и возвращает управление в точку вызова. Если в тексте процедуры не встретился оператор ***Возврат***, то после выполнения последнего исполняемого оператора происходит выполнение неявного оператора ***Возврат***. Конец программной секции процедуры определяется по оператору ***КонецПроцедуры***.

Переменные, объявленные в теле процедуры в разделе Объявления локальных переменных, являются локальными переменными данной процедуры, поэтому доступны только в этой процедуре (за исключением случая передачи их как параметров при вызове других процедур, функций или методов).

Ключевые слова ***Процедура***, ***КонецПроцедуры*** являются не операторами, а операторными скобками, поэтому не должны заканчиваться точкой с запятой (это может приводить к ошибкам выполнения модуля).

Синтаксис:

```
Процедура <ИмяПроцедуры>([[Знач] <Парам 1> [=<ДефЗнач>], …
,[Знач] <Парам N> [=<ДефЗнач>]])[Экспорт]
// Объявления локальных переменных;
// Операторы;
…
[Возврат;]
// Операторы;
…
КонецПроцедуры
```

Англоязычный синтаксис:

```
Procedure <ИмяПроцедуры>([[Val] <Парам 1> [=<ДефЗнач>], … ,[Val]
<Парам N>[=<ДефЗнач>]])[Export]
// Объявления локальных переменных;
// Операторы;
…
[Return;]
// Операторы;
…
EndProcedure
```

Ключевое слово ***Функция*** начинает секцию исходного текста функции, выполнение которой можно инициировать из любой точки программного модуля, просто указав ***ИмяФункции*** со списком параметров (если параметры не передаются, то круглые скобки, тем не менее, обязательны). Если в модуле приложения или общем программном модуле в теле описания функции использовано ключевое слово ***Экспорт***, то это означает, что данная функция является доступной из всех других программных модулей конфигурации.

Выполнение функции заканчивается оператором ***Возврат***. Функции отличаются от процедур только тем, что возвращают ***ВозвращаемоеЗначение***. Конец программной секции функции определяется по оператору ***КонецФункции***.

Вызов любой функции в тексте программного модуля можно записывать как вызов процедуры, т. е. в языке допускается не принимать от функции возвращаемое значение.

Если ключевое слово ***Возврат*** в теле функции не указано или строка модуля, его содержащая, не выполнена, то функция возвращает значение типа ***Неопределено***.

Переменные, объявленные в теле функции в разделе объявления локальных переменных, являются локальными переменными данной функции, поэтому доступны только в этой функции (за исключением случая передачи их как параметров при вызове других процедур, функций или методов).

Ключевые слова ***Функция***, ***КонецФункции*** являются не операторами, а операторными скобками, поэтому не должны заканчиваться точкой с запятой (это может приводить к ошибкам выполнения модуля).

Синтаксис:

```
Функция <ИмяФункции>([[Знач] <Парам 1>[=<ДефЗнач>], … ,[Знач]
<Парам N>[=<ДефЗнач>]])[Экспорт]
// Объявления локальных переменных;
// Операторы;
…
Возврат <Возвращаемое значение>;
// Операторы;
…
КонецФункции
```

Англоязычный синтаксис:

```
Function <ИмяФункции>([[Val] <Парам 1>[=<ДефЗнач>], … ,[Val]
<Парам N>[=<ДефЗнач>]])[Export]
// Объявления локальных переменных;
// Операторы;
…
Return <Возвращаемое значение>;
// Операторы;
…
EndFunction
```

Формы в 1С: Предприятие

Формы в 1С:Предприятие предназначены для отображения и редактирования информации, содержащейся в базе данных. Формы могут принадлежать конкретным объектам конфигурации или существовать отдельно от них и использоваться всем прикладным решением в целом. Каждый объект конфигурации может использоваться для выполнения некоторых стандартных действий. Например, для любого справочника может потребоваться отображать список его элементов, отображать отдельные элементы справочника, отображать группу справочника, выбирать элементы и группы элементов из справочника. Для любого документа список таких действий будет гораздо меньше: просмотр списка документов, выбор из списка документов и просмотр отдельного документа.

Важной особенностью системы 1С:Предприятие 8 является механизм автогенерируемых форм. Этот механизм освобождает разработчика от необходимости создания всех возможных форм для каждого из объектов конфигурации. Разработчику достаточно добавить новый объект конфигурации, а система сама сгенерирует в нужные моменты работы пользователя необходимые формы для отображения информации, содержащейся в этом объекте.

Таким образом, разработчику нужно создавать собственные формы объектов прикладного решения лишь в том случае, если они должны иметь отличия (другой дизайн или специфическое поведение) от форм, автоматически генерируемых системой.

Принадлежность формы тому или иному объекту конфигурации не определяет состав данных, которые отображаются в форме. То, что форма принадлежит, например, справочнику Номенклатура, позволяет назначить ее одной из основных форм для этого справочника, но никак не определяет, какие же именно данные будет отображать эта форма, и каково будет ее поведение.

Для того чтобы связать форму с данными, используются реквизиты формы, в которых указывается перечень данных, отображаемых формой. Все формы, сами по себе, имеют одинаковое поведение, независимо от того, какие данные они отображают. Однако один из реквизитов формы может быть назначен для нее основным (он выделяется жирным шрифтом), и в этом случае стандартное поведение формы и ее свойства будут дополнены в зависимости от того, какой тип имеет основной реквизит формы.

Управляемые формы

Управляемое приложение поддерживает следующие типы клиентов:

- толстый клиент (обычный и управляемый режим запуска)
- тонкий клиент
- веб-клиент

В управляемом приложении используются формы, построенные на новой технологии. Они называются Управляемые формы. Для облегчения перехода прежние формы (т.н. Обычные формы) также поддерживаются, но их функциональность не развивается и они доступны только в режиме запуска толстого клиента.

Основные отличия управляемых форм для разработчика:

- декларативное, а не «по пикселям» описание структуры, конкретное размещение элементов выполняется системой автоматически при отображении формы;
- вся функциональность формы описывается в виде реквизитов и команд; реквизиты – это данные, с которыми работает форма, а команды – выполняемые действия;
- форма выполняется и на сервере и на клиенте;
- в контексте клиента, недоступны практически все прикладные типы, и соответственно невозможно изменить данные в информационной базе;
- для каждого метода или переменной формы обязательно должна быть указана директива компиляции, определяющая, место выполнения (клиент или сервер) и доступ к контексту формы.

Основные директивы компиляции методов формы:

- &НаКлиенте;
- &НаСервере;
- &НаСервереБезКонтекста;
- &НаКлиентеНаСервереБезКонтекста.

Структура формы

Основная особенность форм заключается в том, что они не нарисованы разработчиком детально, «по пикселям». Форма в конфигурации представляет собой логическое описание состава формы. А конкретное размещение элементов выполняется системой автоматически при отображении формы.

Редактор формы используется для создания и редактирования форм объектов прикладного решения. Формы объектов используются системой для визуального отображения данных в процессе работы пользователя.

Любая форма представляет совокупность нескольких составляющих:

- элементов — объектов, определяющих визуальное представление формы и осуществляющих взаимодействие с пользователем,
- командного интерфейса — совокупности команд, отображаемых в форме;
- реквизитов — объектов, данные которых форма использует в своей работе.
- команд — действий, которые определены в данной конкретной форме,
- параметров — объектов, значения которых характеризуют саму форму, используются при ее создании и остаются постоянными в процессе «жизни» формы,
- модуля — программы на встроенном языке, отвечающей за работу с элементами и за обработку событий.

Редактор формы содержит несколько закладок, обеспечивающих редактирование всех составляющих формы.

В отдельном окне, в нижней части редактора, отображается внешний вид формы в режиме 1С:Предприятие.

Отображаемая часть формы (видимая пользователю) описывается как дерево, включающее элементы формы (рисунок 2.1).

Элементы могут представлять собой поля ввода, флажки, переключатели, кнопки и т. д. Кроме того, элемент может быть группой, включающей другие элементы. Группа может представляться как панель с рамкой, панель со страницами (закладками), собственно страница, командная панель. Помимо этого элемент может представлять собой таблицу, которая тоже включает элементы (колонки). Структура элементов описывает то, как будет выглядеть форма (рисунок 2.2).

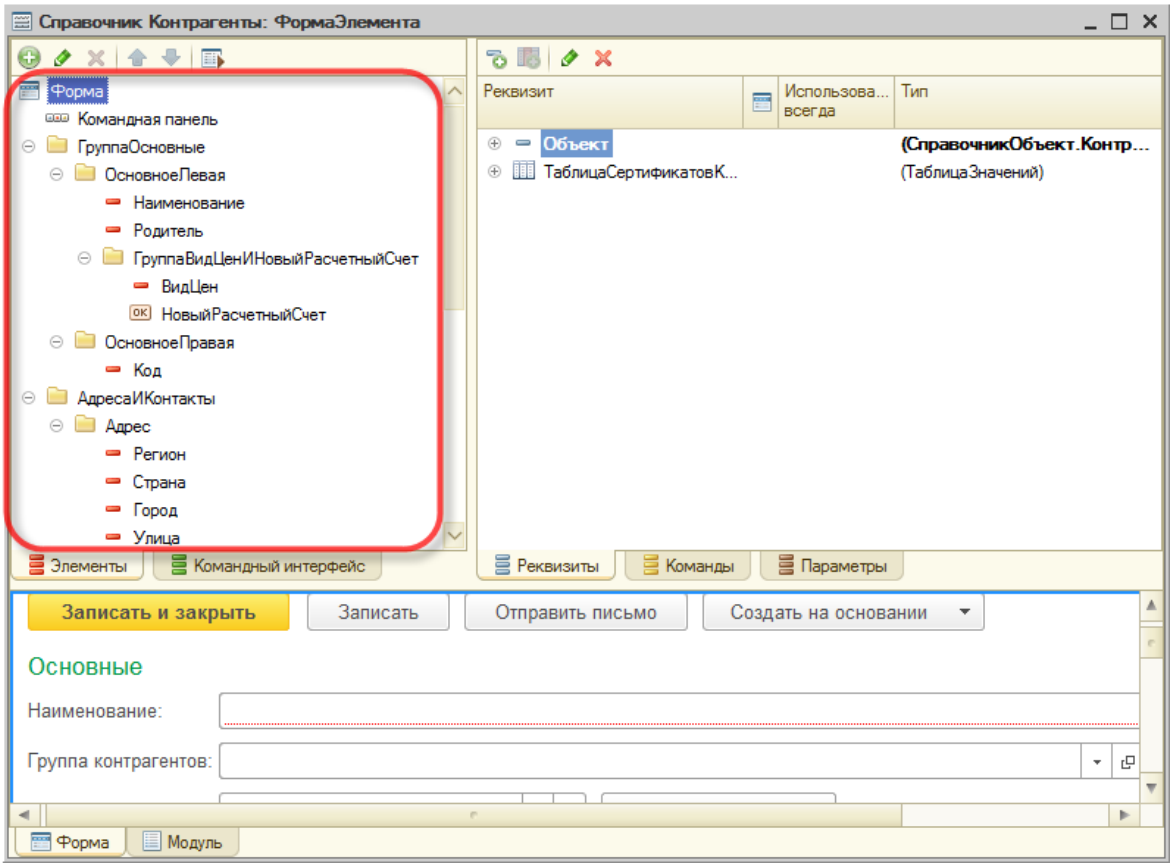


Рисунок 2.1 – Редактор форм для формы элемента Справочника

Редактор форм позволяет добавлять в форму специальные элементы, которые помогают придать форме собственный узнаваемый стиль. Редактор позволяет добавить в форму несколько элементов Группа — Страницы, каждая из которых может содержать несколько элементов Группа — Страница. Например, форма документа может содержать один элемент Группа — Страницы, которому подчинены несколько элементов Группа — Страница с заголовками Изображение, Характеристики и Описание (рисунок 2.3). Заголовок каждой группы — страницы отображается на отдельной закладке.

Разработчик имеет возможность задать режим отображения закладок: снизу или сверху. Редактор позволяет добавлять в форму различные элементы. Добавлять элементы можно с помощью команды добавления или путем перетаскивания реквизитов формы в дерево элементов. Все элементы формы представляются в виде иерархической структуры, корнем которой является сама форма. Это позволяет быстро перемещаться к нужному элементу формы. Располагая элементы выше/ниже в дереве, подчиняя их другим элементам и задавая свойства элементов-групп можно задавать порядок, в котором пользователь будет обходить элементы управления формы при вводе и редактировании данных. В режиме 1С: Предприятие элементы формы будут обходиться в порядке их иерархии и в соответствии

с тем, какой тип группировки выбран для групп: вертикальная или горизонтальная.

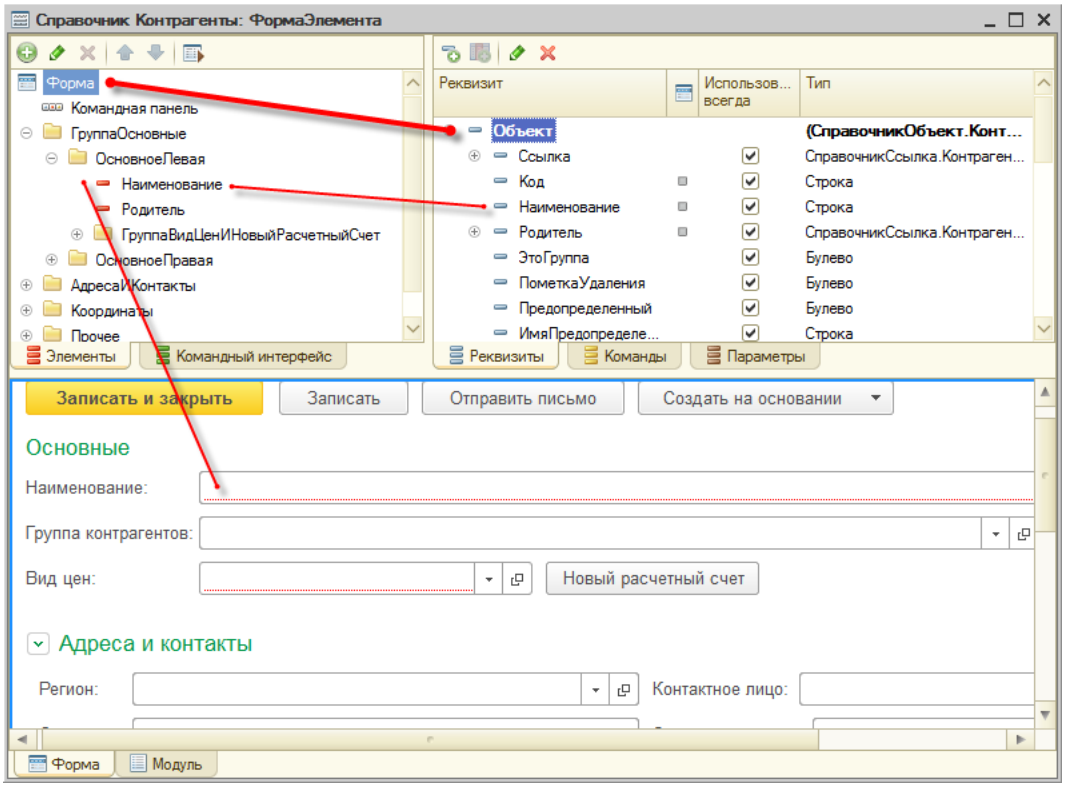


Рисунок 2.2 – Соответствие элементов, объектов и вида формы

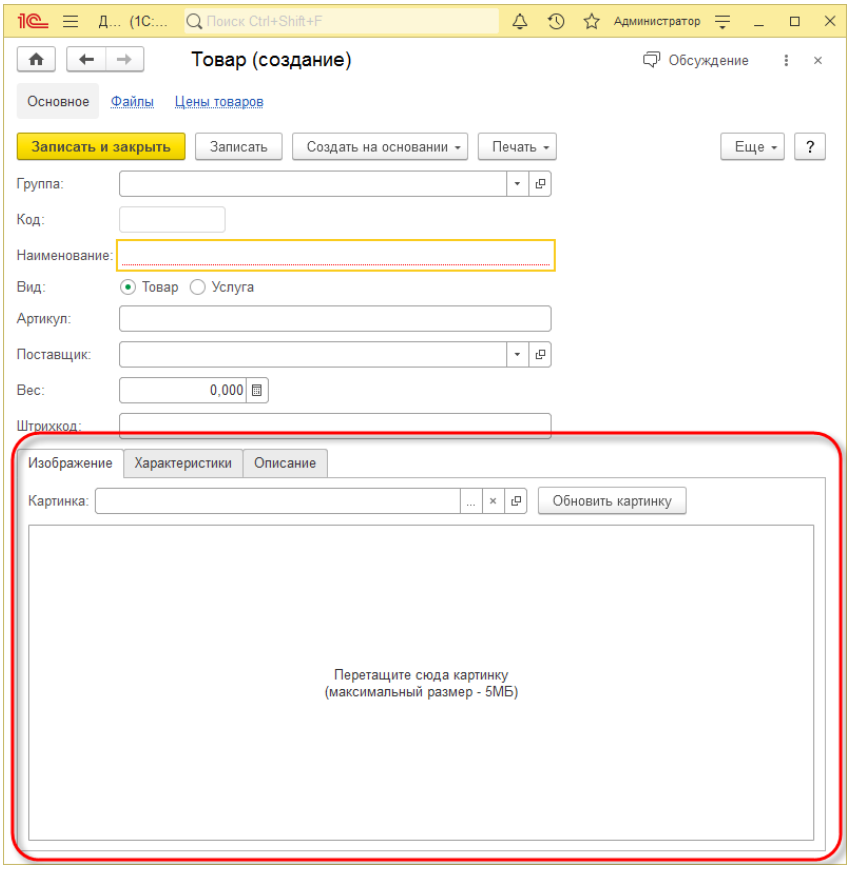


Рисунок 2.3 – Пример использования элементов настройки форм

Разделители являются специальными элементами, с помощью которых возможно перераспределение пространства формы без изменения ее размеров. Платформа в режиме 1С:Предприятие самостоятельно добавляет эти элементы в форму. Разделитель обладает способностью «захватываться» мышью и перемещаться внутри формы в ее пределах с учетом возможности расположения других элементов и ориентации разделителя (рисунок 2.4).

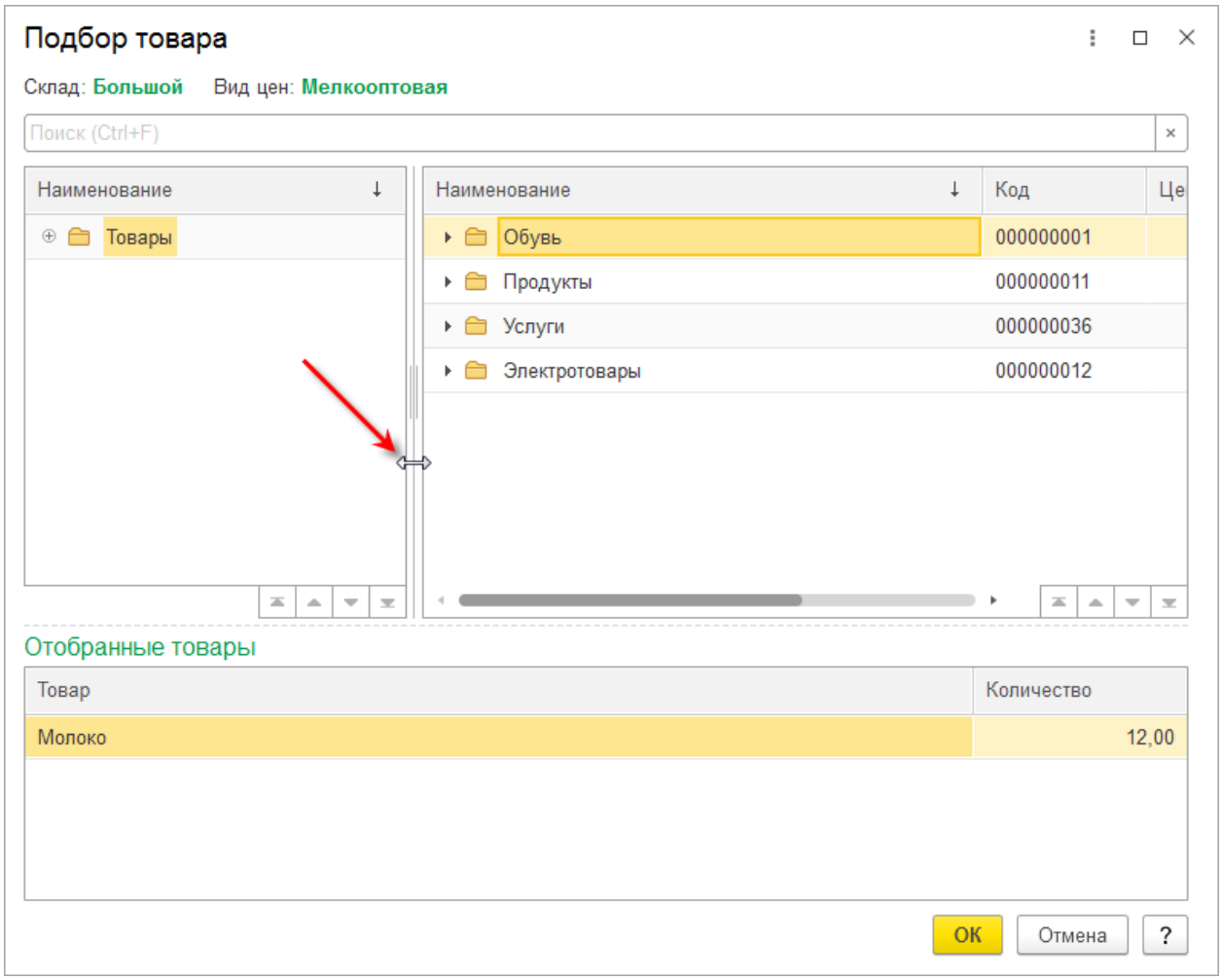


Рисунок 2.4 – Пример использования разделителей

Вся функциональность формы описывается в виде реквизитов и команд. Реквизиты — это данные, с которыми работает форма, а команды — выполняемые действия. Таким образом, разработчик в редакторе формы должен включить в форму необходимые реквизиты и команды, создать отображающие их элементы формы и, если необходимо, скомпоновать элементы в группы.

На основе этого логического описания система автоматически формирует внешний вид формы для отображения пользователю. При этом системой учитываются различные свойства отображаемых данных (например, тип), чтобы максимально удобно для пользователя расположить элементы формы. Разработчик может влиять на расположение элементов

различными установками. Он может определять порядок элементов, указывать желаемую ширину и высоту. Однако это является только некоторой дополнительной информацией, помогающей системе отобразить форму.

В формах разработчик может использовать не только команды самой формы, но и глобальные команды, используемые в командном интерфейсе всей конфигурации (рисунок 2.5). Кроме того, реализована возможность создания параметризуемых команд, которые будут открывать другие формы с учетом конкретных данных текущей формы. Например, это может быть вызов отчета по остаткам на том складе, который выбран сейчас в форме расходной накладной.

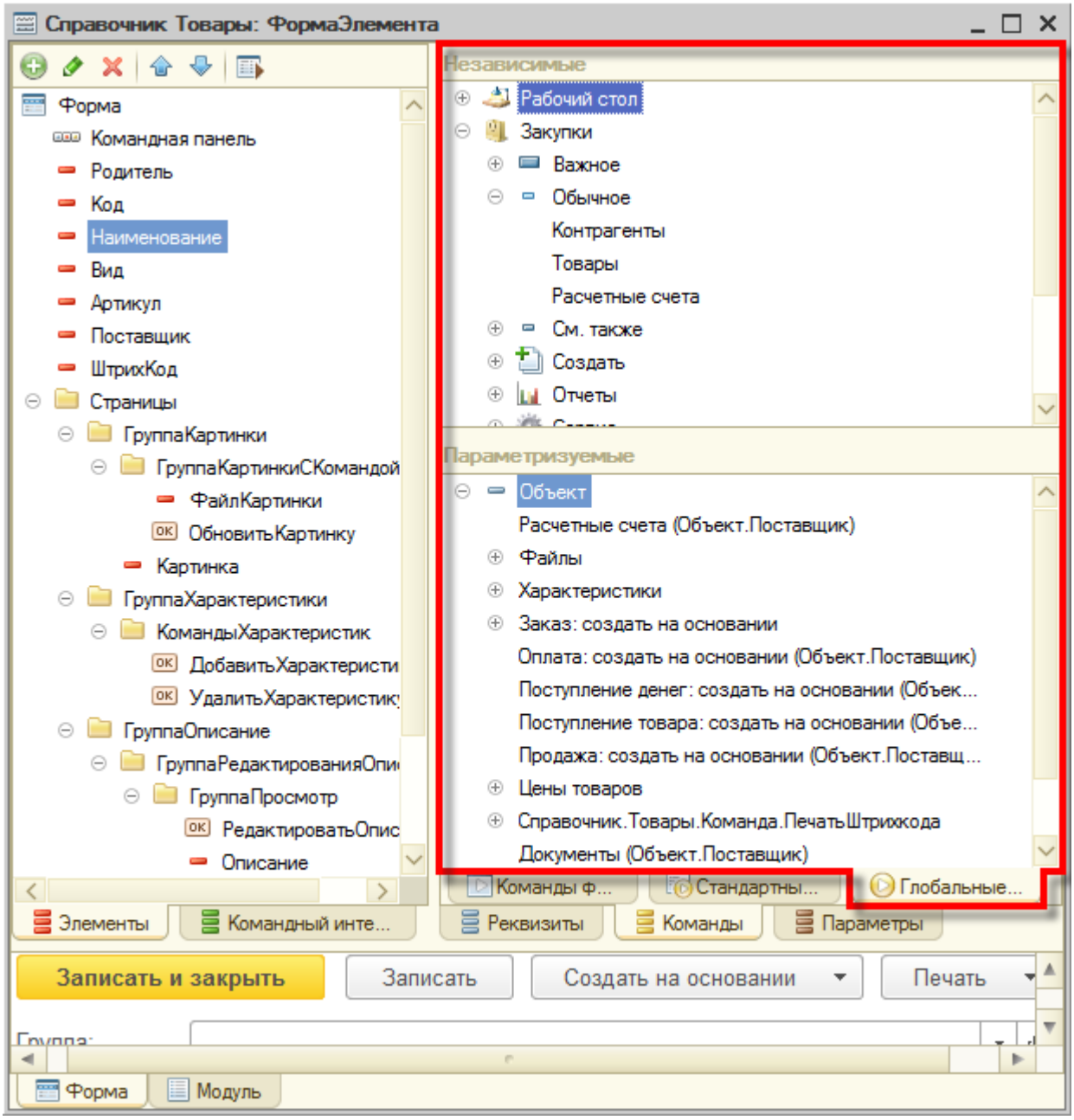


Рисунок 2.5 – Пример настройки команд формы

Пример

Рассмотрим настройку форм для элемента справочника. Для того чтобы пользователь мог просматривать и изменять данные, содержащиеся в справочнике, система поддерживает несколько форм представления справочника. Система может автоматически генерировать все нужные формы справочника. Наряду с этим разработчик имеет возможность создать собственные формы, которые система будет использовать вместо форм по умолчанию.

Для просмотра и изменения данных отдельных элементов справочника используется форма элемента. Как правило, она представляет данные в удобном для восприятия и редактирования виде. Форма элемента имеет «*Модуль формы*». Данный модуль предназначен для того, чтобы обработать действия пользователя. Например, описать алгоритм реакции программы при нажатии кнопки. Или, например, в момент ввода в поле значения сразу же выполнить проверку на корректность. Кроме событий, связанных с элементами управления формы (кнопки, поля ввода) существуют события, связанные непосредственно с самой формой. Например, можно обработать событие открытия формы и провести некую начальную инициализацию. Также можно обработать событие закрытия формы и проверить, а все ли правильно ввел пользователь.

Программный код основной программы будет выполняться в момент инициализации формы, т.е. когда пользователь начинает ее открывать. Список событий управляемой формы виден также в списке свойств непосредственно для самой формы. Данный список вызывается в редакторе управляемых форм.

Обращение к элементам формы в модуле через объект ***Элементы.<ИмяЭлементаФормы>***, а непосредственно к реквизитам справочника через объект ***Объект.<ИмяРеквизитаСправочника>***. Обращение к элементам табличной части происходит через объявление и инициализацию переменной, содержащей текущие данные: ***Элементы.<ИмяТабличнойЧасти>.ТекущиеДанные***. Дальнейшее обращение к элементам таблицы происходит следующим образом: ***<ИмяПеременной>.<ИмяРеквизитаТабличнойЧасти>***.

Получение данных из справочника без запросов

Для получения данных из справочника по ссылке можно получить следующим образом. Для элемента формы (связанного с объектом справочника типа *СправочникСсылка*) добавить обработчик события

ОбработкаВыбора. На сервере создается функция, которая обращается к содержимому объекта по ссылке:

```
&НаСервереБезКонтекста
Функция ПолучитьЗначение(ВыбранноеЗначение)
    Возврат ВыбранноеЗначение.<ИмяРеквизита>;
КонецФункции
```

Настройка печатных форм и других макетов

Документу и справочнику могут быть сопоставлены несколько макетов, содержащих данные, необходимые для обеспечения работы документа. Макеты могут использоваться для формирования печатных форм документа или для отображения дополнительной информации, имеющей отношение к документу.

Для создания печатной формы можно воспользоваться Конструктором печати (рисунок 2.6).

Печатная форма (рисунок 2.7) содержит: область заголовка, область шапки, область табличных частей и область подвала. В режиме редактирования можно добавить пользовательские области и изменить текущие.

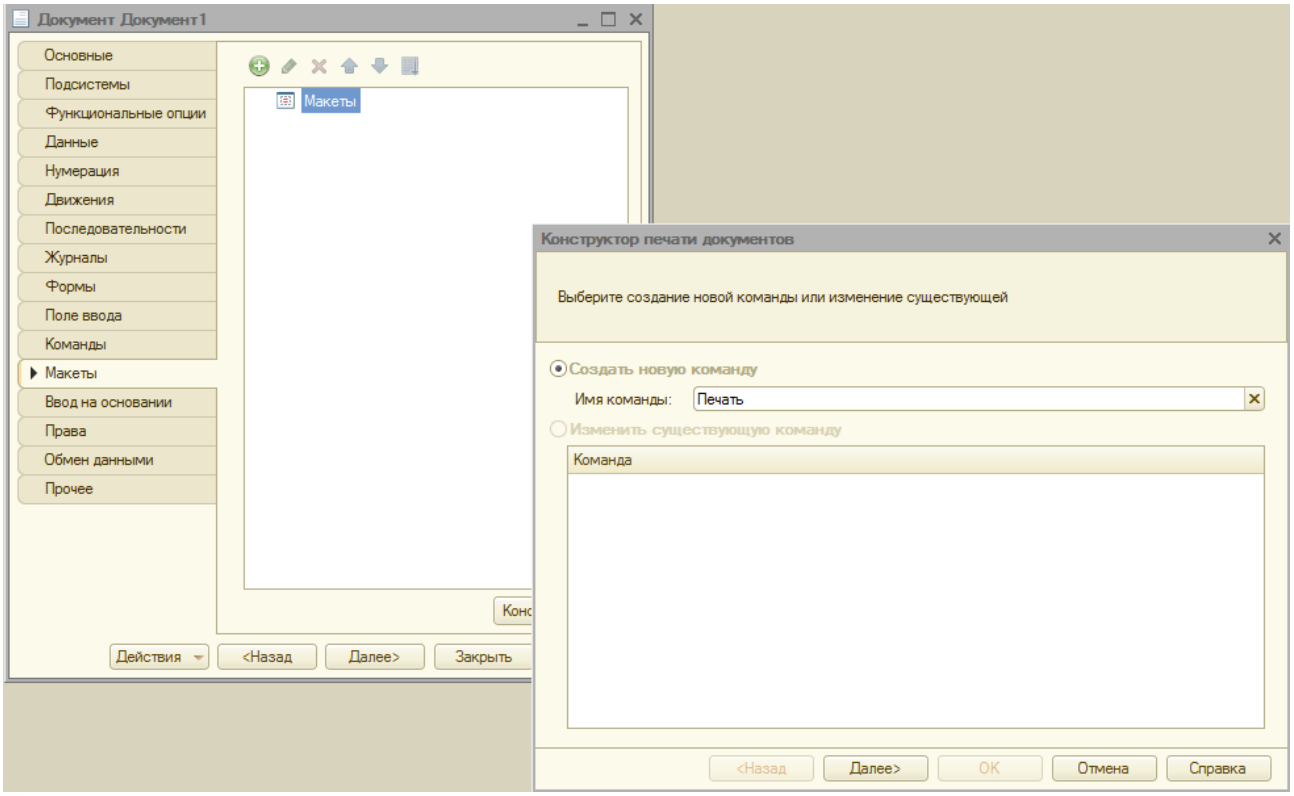


Рисунок 2.6 - Форма Конструктор печати

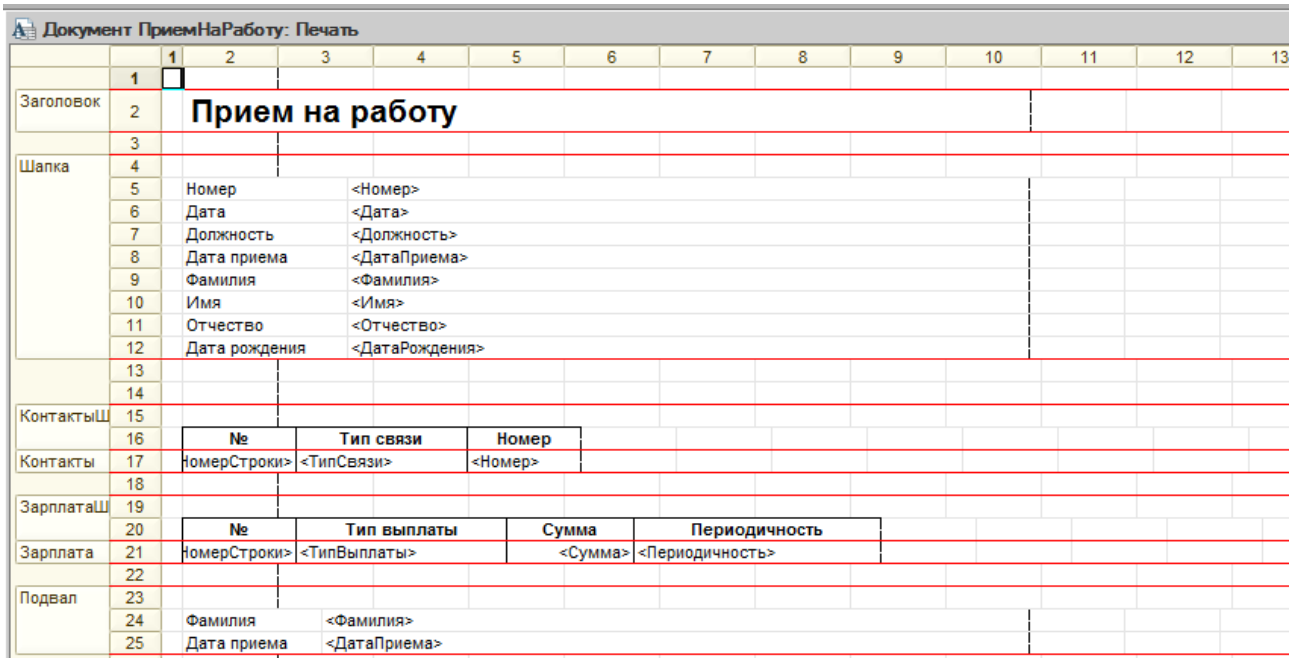


Рисунок 2.7 – Редактирование *печатной формы документа*

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №2

1. Создать Форму элемента справочника для справочника «ФизическиеЛица».
2. Для автоматического заполнения поля «Наименование», добавить для поля «Фамилия» обработчик события «ПриИзменении». В модуле появится соответствующая процедура.
3. В процедуру вставить следующий обработчик: ***Элементы.Наименование.ВыделенныйТекст=Элементы.ТекстРедактирования+""+Лев(Элементы.Имя.ТекстРедактирования,1)+"."+Лев(Элементы.Отчество.ТекстРедактирования,1)+".";***.
4. Создать обработку события «ПриИзменении» так же для полей «Имя» и «Отчество».
5. Реализовать подобные обработки для справочника «Сотрудники».
6. Для удобство работы с документом «ПриемНаРаботу» добавить в спрвочник «Должности» поле «Оклад» и добавить в форму документа «ПриемНаРаботу» обработку выбора должности, добавляющую новую строку в табличную часть «Зарплата» и получающую данные из справочника по ссылке:

```
&НаКлиенте
Процедура ДолжностьОбработкаВыбора(Элемент, ВыбранноеЗначение,
СтандартнаяОбработка)
      Элементы.Зарплата.ДобавитьСтроку();
      ТС = Элементы.Зарплата.ТекущиеДанные;
      ТС.ТипВыплаты ="Оклад";
      Оклад =ПолучитьДанныеИзСправ(ВыбранноеЗначение);
      ТС.Сумма=Оклад;
      ТС.Периодичность="по дням";
КонецПроцедуры

&НаСервереБезКонтекста
ФункцияПолучитьДанныеИзСправ(ВыбранноеЗначение)
      Возврат ВыбранноеЗначение.Оклад;
КонецФункции
```

Практическая работа №3

1. Реализовать форму элемента для справочника «Клиенты», на основании данных контактного лица, представленную на рисунке 2.8.

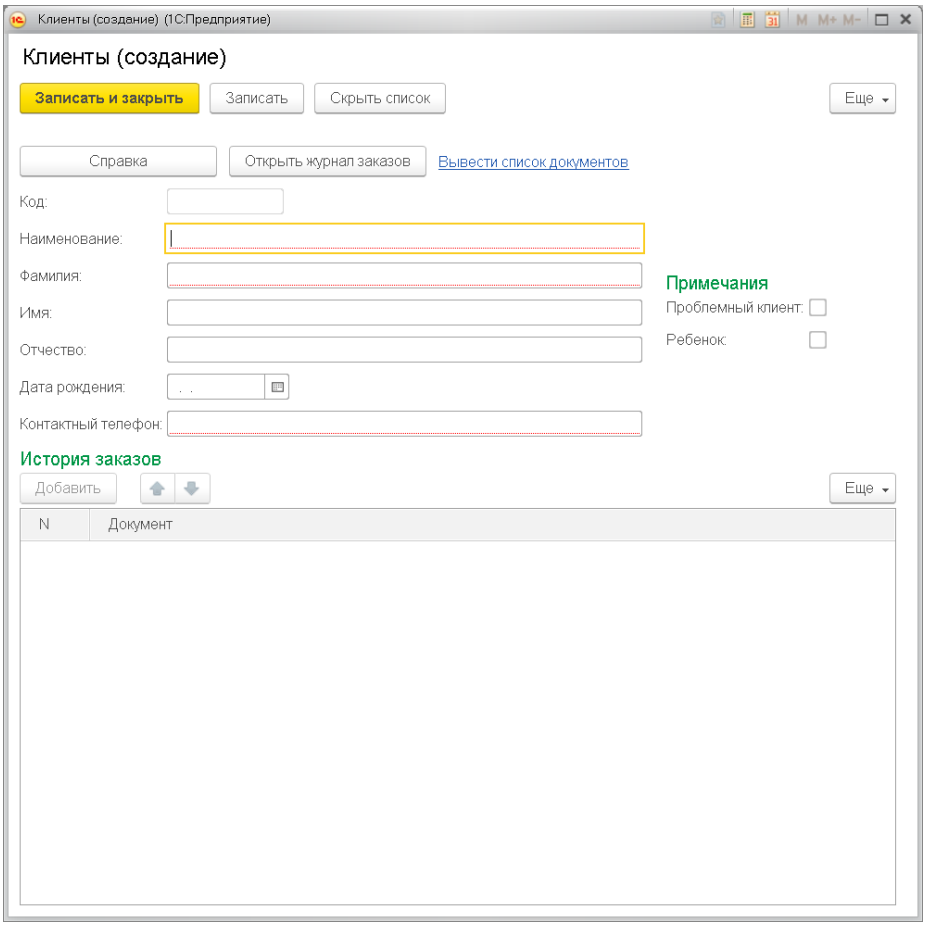


Рисунок 2.8 – Форма элемента справочника «Клиенты»

2. Реквизит «Наименование» должен заполняться автоматически.

3. Табличная часть ограничения становится невидимой при нажатии на кнопку «Скрыть». После этого кнопка скрыть меняется на кнопку «Показать табличную часть».

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Каким образом можно создать Печатную форму Документа?
2. Как подсчитать количество строк и итоги Табличной части документа?
3. Чем отличается процедура НаКлиенте и НаСервере?
4. Что такое Конструктор запроса?
5. Создать документ, который регламентировал бы оказание услуг клиентам. Для этого создать справочники «Клиенты» и «Услуги», перечисление «ТипОрганизации» и документ «ДоговорОбОказанииУслуг». Объекты конфигурации должны содержать следующие поля:
   a. справочник «Клиенты»: организация, тип организации (ОАО, АО, ЗАО, ООО), контактное лицо, адрес, телефон;
   b. документ «ДоговорОбОказанииУслуг»: организация, тип организации (ОАО, АО, ЗАО, ООО), контактное лицо, адрес, телефон, итоговая сумма, табличную часть услуги (наименование, количество, цена за ед., сумма).
6. В форме документа создать необходимые обработчики для автоматизации расчетов суммы в табличной части и итоговой суммы документы.
7. Автоматизировать процедуру создания элемента справочника при заполнении документа, используя механизм Ввод на основании.

Раздел 3. Учетные механизмы

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Регистр сведений

Регистры сведений - это прикладные объекты конфигурации. Они позволяют хранить в прикладном решении произвольные данные в разрезе нескольких измерений. Например, в регистре сведений можно хранить курсы валют в разрезе валют, или цены предприятия в разрезе номенклатуры и типа цен. Информация в регистре сведений хранится в виде записей, каждая из которых содержит значения измерений и соответствующие им значения ресурсов.

Измерения регистра описывают разрезы, в которых хранится информация, а ресурсы регистра непосредственно содержат хранимую информацию. Вместе с каждой записью, находящейся в регистре сведений, можно хранить дополнительную произвольную информацию. Для этого служат реквизиты регистра сведений.

Одной из возможностей регистра сведений является хранение данных не только в разрезе указанных измерений, но и в разрезе времени. Разработчик может указать минимальную периодичность, с которой записи будут заноситься в регистр. В этом случае к каждой записи регистра будет добавляться поле «*Период*», хранящее дату, которой были внесены записи в регистр. Использование периодичности регистра сведений позволяет не просто хранить статические данные, но и отслеживать их изменение во времени.

Внесение изменений в регистр сведений может выполняться как вручную, так и при помощи документов. В случае, когда изменения в регистр сведений вносятся с помощью документов, к каждой записи регистра добавляется специальное поле, в котором хранится информация о регистраторе - документе, с которым связана эта запись. В процессе создания прикладного решения разработчик указывает, какой именно режим записи будет использоваться данным регистром сведений.

Использование режима записи «*Подчинение регистратору*» может потребоваться в случае, когда логика работы прикладного решения требует того, чтобы изменения, выполняемые в регистре сведений, были жестко связаны с документами, фиксирующими факты хозяйственной деятельности.

Система обеспечивает контроль уникальности записей, хранящихся в регистре сведений. Таким образом, в регистре сведений не может находиться двух одинаковых записей. Одинаковыми считаются записи, у которых

совпадает ключ записи. Ключ записи формируется системой автоматически, на основании значений, содержащихся в полях записи, и зависит от вида регистра сведений.

В общем случае в формировании ключа записи будут участвовать значения регистратора, периода и значения измерений.

Основными функциональными возможностями, которые предоставляет регистр сведений разработчику, являются:

- создание, изменение и удаление записей;
- выбор записей в заданном интервале по заданным критериям;
- выбор записей по регистратору;
- получение значений ресурсов записей, соответствующих указанному периоду и значениям измерений;
- получение значений ресурсов наиболее ранних и наиболее поздних записей регистра, соответствующих указанному периоду и значениям измерений.

Регистр накоплений

Регистры накопления - это прикладные объекты конфигурации. Они составляют основу механизма учета движения средств (финансов, товаров, материалов и т.д.), который позволяет автоматизировать такие направления, как складской учет, взаиморасчеты, планирование.

Регистр накопления образует многомерную систему измерений и позволяет "накапливать" числовые данные в разрезе нескольких измерений. Например, в таком регистре можно накапливать информацию об остатках товаров в разрезе номенклатуры и склада, или информацию об объемах продаж в разрезе номенклатуры и подразделения компании.

Информация в регистре накопления хранится в виде записей, каждая из которых содержит значения измерений и соответствующие им значения ресурсов.

Измерения регистра описывают разрезы, в которых хранится информация, а в ресурсах регистра накапливаются нужные числовые данные.

Изменение состояния регистра накопления происходит, как правило, при проведении документа. Поэтому каждая запись регистра связана с определенным документом - регистратором, номером строки этого документа, и датой –периодом. В общем случае значение поле «*Период*» может не совпадать с датой документа.

Существует два вида регистров накопления: регистры накопления остатков и регистры накопления оборотов. Регистр накопления остатков

позволяет хранить как итоговые значения ресурсов - остатки, так и изменения этих ресурсов - обороты. Регистр накопления оборотов является более "специализированным" видом регистра накопления и позволяет хранить только изменения ресурсов - обороты.

Существование регистра накопления оборотов связано с тем, что при автоматизации экономической деятельности существует большое количество ситуаций, когда требуется накапливать только обороты, а значения остатков не имеют смысла.

Основными функциональными возможностями, которые предоставляет регистр накопления разработчику, являются:

- выбор записей в заданном интервале по заданным критериям;
- выбор записей по регистратору;
- получение остатков и оборотов на указанный момент времени по заданным значениям измерений;
- режим работы с разделением итогов, который обеспечивает более высокую параллельность записи в регистр;
- отключение использования текущих итогов;
- расчет итогов на указанную дату;
- чтение, изменение и запись набора записей в регистр;
- возможность записи в регистр без пересчета итогов;
- полный пересчет итогов и пересчет итогов за указанный период.

Движение документа. Конструктор движения

Важным свойством документа является возможность его проведения. Если документ проводится, то он может изменить состояние тех или иных учитываемых данных. Если же документ не является «проводимым» это значит, что событие, которое он отражает, не влияет на состояние учета, который ведется в данном прикладном решении.

Алгоритм, на основании которого документ вносит те или иные изменения в состояние учетных данных при своем проведении, описывается средствами встроенного языка на этапе разработки прикладного решения. Система содержит «*Конструктор движений*», который помогает разработчику создавать алгоритмы проведения документа.

Конструктор движений - это один из инструментов разработки. Он используется только для документов и помогает создать процедуру обработки проведения документа на встроенном языке. Конструктор может быть вызван, например, из окна редактирования документа.

Конструктор позволяет выбрать регистры, в которые будут вноситься записи и затем вручную или автоматически заполнить выражения, которые будут записаны в поля регистра. Работа с Конструктором движения похожа на работу с Конструктором ввода на основании.

Результатом работы конструктора является готовая процедура на встроенном языке с именем **ОбработкаПроведения()**. Эта процедура располагается в модуле документа и будет вызвана системой в момент проведения документа.

Получение данных из регистра сведений

Объект «*РегистрСведенийМенеджер»* позволяет обращаться к «итогам» регистра. Под «итогами» периодического регистра сведений понимаются первые или последние значения ресурсов по указанным измерениям. При этом применяются следующие методы:

- ***Получить (<Период>, <Отбор>)***– возвращает в виде структуры значения ресурсов одной записи регистра, соответствующей указанным значениям всех измерений регистра и периоду.
- ***ПолучитьПоследнее (<Конец периода>, <Отбор>)***– этот метод возвращает актуальное значение ресурсов, действовавшее на заданную дату. Если он не находит запись в регистре по данной комбинации измерений точно на заданный период, то возвращается структура, содержащая значения ресурсов ближайшей более поздней записи.
- ***ПолучитьПервое (<Начало периода>, <Отбор>)*** – этот метод действует аналогично методу ***ПолучитьПоследнее***, но если записи на данный момент не находится, то возвращается структура, содержащая значения ресурсов ближайшей более ранней записи.
- ***СрезПоследних (<Конец периода>, <Отбор>), СрезПервых (<Начало периода>, <Отбор>)*** – эти методы аналогичны методам ***ПолучитьПоследнее***и ***ПолучитьПервое***соответственно, но при их использовании, как правило, не указывается одно или несколько измерений. В результате возвращается не структура, как в предыдущих случаях, а таблица значений, заполненная данными найденных записей регистра сведений.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №4

1. Создать регистр сведений «ДолжностиНовичок» с режимом записи «Подчинение регистратору» и периодичностью «по позиции регистратора». В качестве измерения указать «Должность», типа СправочникСсылка.Должности, а в качестве ресурса – «Сотрудник», типа СправочникСсылка.Сотрудники. В качестве регистратора определить документ «ПриемНаРаботу». Добавить регистр в соответствующую подсистему.
2. В документе «ПриемНаРаботу» на вкладке Движение открыть Конструктор движения и заполнить его (рисунок 3.1)

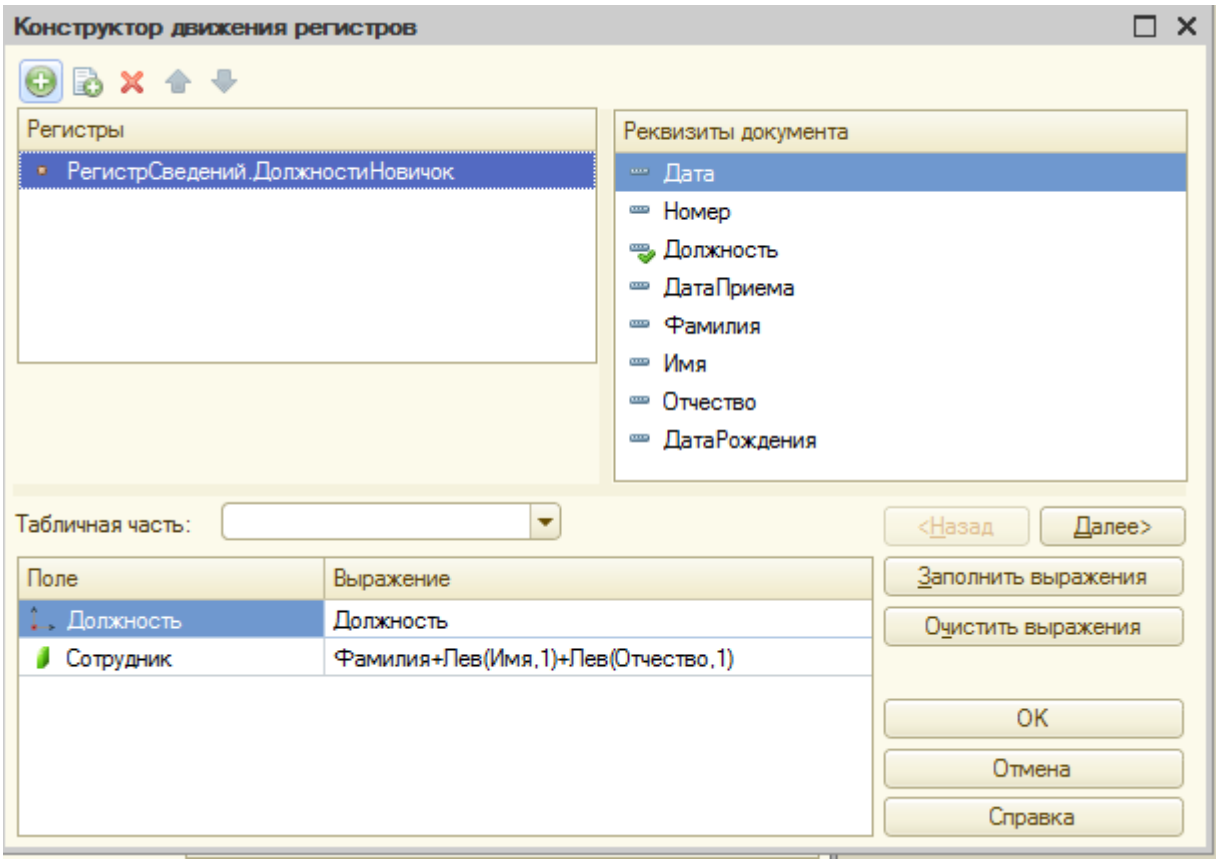


Рисунок 3.1 – Форма Конструктора движения регистров

3. В подсистеме «Склад» реализовать учет пришедших товаров и товаров, принятых и списанных со склада. Для этого создать справочники «Номенклатура» и «МестаХранения», документы «ПоставкаТовара», «СписаниеТовараСоСклада», «ПоставкаНаСклад», регистр сведений «ПоследниеЦеныПоставки» и необходимые перечисления. Объекты конфигурации должны содержать следующие поля:
   a. Справочник «Номенклатура»: единица хранения, страна производитель, артикул, способ хранения;

b. Справочник «МестаХранения»: способ хранения;
c. Документ «ПоставкаТовара»: поставщик, ответственный кладовщик, сумма поставки, товары, наименование, количество, цена, сумма, способ хранения;
d. Документ «СписаниеТовараСоСклада»: ответственное лицо, сумма списания, товары, наименование, количество, цена, сумма, способ хранения, место хранения, назначение;
e. Документ «ПоставкаНаСклад»: ответственный кладовщик, количество товара, сумма товара, товары, наименование, количество, цена, сумма, способ хранения, место хранения;
f. Регистр Сведений «ПоследниеЦеныПоставки»: измерение: номенклатура, ресурс: цена.

4. В формах документов необходимо добавить обработчики для автоматизации расчетов суммы и итоговой суммы. Для создания документа «ПоставкаНаСклад» создать ввод на основании из документа «ПоставкаТовара». Для документов создать печатные формы.
5. Для документа «СписаниеТовараСоСклада» автоматизировать заполнение цен в табличной части по регистру сведений «ПоследниеЦеныПоставки». Для этогов форме документа создать обработку события «ОбработкаВыбора» для элемента «Наименование» табличной части:

```
&НаКлиенте
Процедура ТоварыНаименованиеПриИзменении(Элемент)
     СТЧ =Элементы.Товары.ТекущиеДанные;
     СТЧ.Цена=ПолучитьЦену(СТЧ.Наименование);
КонецПроцедуры

&НаСервере
Функция ПолучитьЦену(НаименованиеТовара)
Отбор =Новый Структура("Номенклатура",НаименованиеТовара);
Цены =
РегистрыСведений.ПоследниеЦеныПоставки.ПолучитьПоследнее(Текущая
Дата(),Отбор);

Возврат Цены.Цена;
КонецФункции
```

Практическая работа №5

1. Создать регистр накопления «КоличествоУслуг», вида обороты, с измерением «Услуга» типа СправочникСсылка.Услуги и

ресурсом «Количество». В качетсве регистратора добавить документ «ДоговорОбОказанииУслуг». Из документа вызвать Конструктор движения регистров и заполнить поля.

2. Создать регистр накопления «Склад», вида остатки, с измерением «Товар» типа СправочникСсылка.Номенклатура и ресурсом «Количество». В качестве регистратора добавить документы «ПоставкаНаСклад» и «СписаниеТовараСоСклада». Из документа «ПоставкаНаСклад» вызвать Конструктор движения регистров. В качетве типа движения регистра указать «Приход». Заполнить поля регистра из реквизитов табличной части документа. Для документа «СписаниеТовараСоСклада» указать тип движения – «Расход».
3. Создать регистры накопления, отслеживающие количество сотрудников на каждой должности. Регистратором в данном случае выступает документ «ПриемНаРаботу».
4. Добавить регистр накопления в подсистему «Сотрудники», обеспечивающий подсчет сотрудников на каждой должности.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Охарактеризуйте свойство проводимость Документа.
2. Чем отличается Регистр накоплений Остатки и Регистр накоплений Обороты?
3. Что такое Измерения и Ресурсы в регистрах?
4. Каким образом получить данные из Регистра Сведений?
5. Как обратиться к реквизитам Справочника по наименованию?
6. Что такое Конструктор движения?
7. Что такое регистратор?
8. Создать регистры накопления, отслеживающие количество сотрудников на каждой должности. Регистратором в данном случае выступает документ «ПриемНаРаботу». Добавить регистр накопления в подсистему «Сотрудники», обеспечивающий подсчет сотрудников на каждой должности.

Раздел 4. Язык запросов

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Механизм запросов - это один из способов доступа к данным, которые поддерживает платформа. Используя этот механизм, разработчик может читать и обрабатывать данные, хранящиеся в информационной базе; изменение данных с помощью запросов невозможно. Это объясняется тем, что запросы специально предназначены для быстрого получения и обработки некоторой выборки из больших массивов данных, которые могут храниться в базе данных.

Запросы реализуют табличный способ доступа к данным, которые хранятся в базе данных. Это означает, что все данные представляются в виде совокупности связанных между собой таблиц, к которым можно обращаться как по-отдельности, так и к нескольким таблицам во взаимосвязи.

Язык запросов 1С: Предприятие

Для того чтобы разработчик имел возможность использовать запросы для реализации собственных алгоритмов, в платформе реализован язык запросов. Этот язык основан на SQL, но при этом содержит значительное количество расширений, ориентированных на отражение специфики финансово-экономических задач и на максимальное сокращение усилий по разработке прикладных решений. Можно перечислить наиболее существенные возможности, реализуемые языком запросов:

- обращение к полям через точку - ПоставкаТовара.ОтветственныйКладовщик.Должность;
- обращение к вложенным таблицам (табличным частям документов и элементов справочника) – ***ДоговорОбОказанииУслуг.Товары.(Наименование, Количество, Цена, Сумма);***
- автоматическое упорядочивание;
- многомерное и многоуровневое формирование итогов;
- поддержка виртуальных таблиц;
- стандартные SQL операции;
- временные таблицы;
- пакетные запросы – создание временной таблицы и ее использование помещаются в один запрос.

В таблице 4.1 представлены основные операторы языка запросов 1С: Предприятие.

Таблица 4.1 – Соответствие основных операторов языка запросов 1С операторам SQL

Операторы языка запросов 1С	**Оператор SQL**
ВЫБРАТЬ	SELECT
ГДЕ	WHERE
РАЗЛИЧНЫЕ	DISTINCT
ПОДОБНО	LIKE
ПЕРВЫЕ	TOP
ВЫБОР	CASE
СОЕДИНЕНИЕ	JOIN
СГРУППИРОВАТЬ ПО	GROUP BY
ОБЪЕДИНИТЬ	UNION
УПОРЯДОЧИТЬ ПО	ORDER BY
ИМЕЮЩИЕ	HAVING
ИЗ	FROM
В	IN

Конструктор запроса

Для облегчения труда разработчика технологическая платформа содержит два специальных конструктора.

Конструктор запроса - это один из инструментов разработки. Он позволяет составить текст запроса на языке запросов исключительно визуальными средствами. С помощью кнопок «*Далее»* и «*Назад»* можно перемещаться по закладкам конструктора и указывать, какие данные должны присутствовать в результате запроса, как они связаны, сгруппированы, какие итоги следует рассчитать, работать с временными таблицами, редактировать пакет запросов.

Результатом работы конструктора будет являться синтаксически правильный текст запроса. Таким образом, разработчик может составить работоспособный запрос, даже не владея синтаксисом языка запросов - необходимые синтаксические конструкции конструктор сгенерирует автоматически. Готовый текст запроса может быть сразу же вставлен в текст модуля или скопирован в буфер обмена.

Кроме этого конструктор запросов позволяет редактировать уже имеющийся в программе текст запроса. Для этого достаточно установить курсор внутри существующего текста запроса и вызвать конструктор. Имеющийся текст запроса будет проанализирован и представлен в

конструкторе в виде соответствующих выбранных полей базы данных и набора заданных связей, группировок, условий и т.д.

Конструктор запроса с обработкой результата - это один из инструментов разработки. Он позволяет составить текст запроса и сформировать фрагмент программного кода, который исполняет запрос и выводит результаты в табличный документ или диаграмму.

На первом шаге своей работы конструктор предлагает выбрать один из возможных вариантов обработки результата запроса: просто обход результата для его дальнейшей программной обработки или вывод данных в табличный документ или диаграмму (рисунок 4.1).

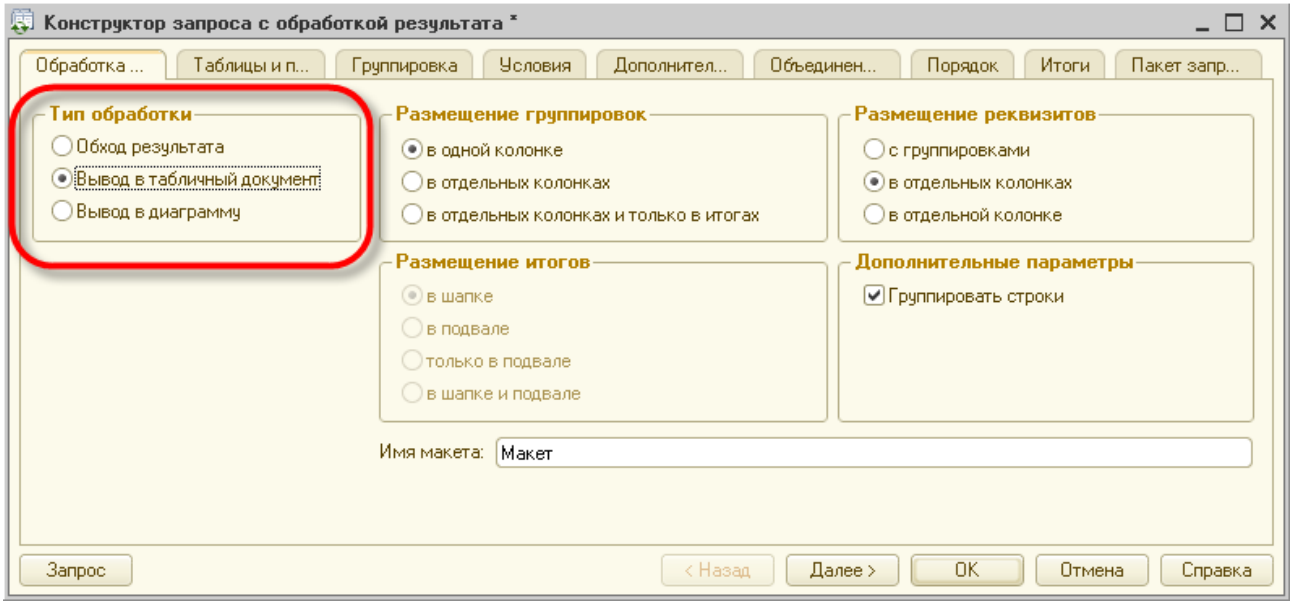


Рисунок 4.1 – Модуль Конструктора запроса с обработкой результата

Следующие шаги работы конструктора позволяют создать текст запроса к базе данных. Эти возможности аналогичны тем, которые предоставляет конструктор запроса.

Результатом работы конструктора запроса с обработкой результата является готовый фрагмент программного кода (рисунок 4.2) и, например, макет табличного документа. Разработчик может внести в них, при необходимости, свои изменения.

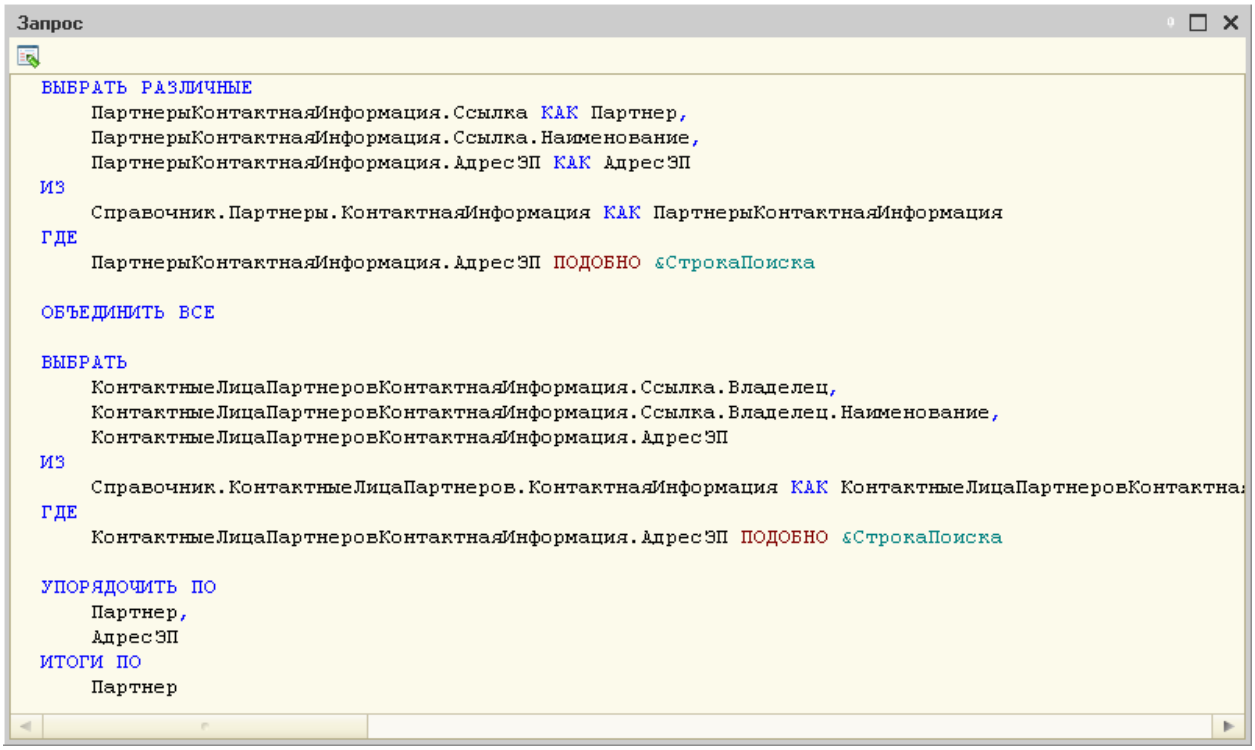


Рисунок 4.2 – Пример готового фрагмента кода, генерируемого конструктором запроса

Выполнение запроса 1С из программного кода осуществляется при помощи объекта встроенного языка ***«Запрос»***. Пример написания запроса к базе данных с использованием встроенного языка программирования:

```
Запрос = Новый Запрос;
Запрос.Текст =
                        "ВЫБРАТЬ
                        | Синоним.Ссылка КАК Ссылка
                        |ИЗ
                        | Справочник.Справочник1 КАК
Синоним";
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
        // Вставить обработку выборки
ВыборкаДетальныеЗаписи
КонецЦикла;
```

Метод ***«Выполнить»*** выполняет запрос, метод ***«Выбрать»*** возвращает значение типа «ВыборкаИзРезультатаЗапроса». Также можно использовать метод ***«Выгрузить»***, который возвращает таблицу значений.

Параметры запроса хранятся в свойстве ***«Параметры»*** (в данном случае это структура, поэтому все методы структуры тут применимы – вставить, удалить и т.д.).

Пример установки параметра ***«Запрос.Параметры.Вставить» («Справочник», СправочникСсылка)***. В запросе обратиться к параметрам можно через амперсанд ***«&Справочник»***. Ниже пример запроса с использованием параметров:

```
Запрос = Новый Запрос;
Запрос.Текст =
                "ВЫБРАТЬ
                |    Пользователи.Ссылка КАК Ссылка,
                |    Пользователи.Родитель КАК Родитель,
                |    Пользователи.Наименование КАК
Наименование
                |ИЗ
                |    Справочник.Пользователи КАК Пользователи
                |ГДЕ
                |    Пользователи.Ссылка = &Справочник";
 Запрос.Параметры.Вставить("Справочник", СправочникСсылка);
 Выборка = Запрос.Выполнить().Выбрать();
 Пока Выборка.Следующий() Цикл
     // Вставить обработку выборки ВыборкаДетальныеЗаписи
 КонецЦикла;
```

Язык запросов предназначен только для чтения данных из базы, поэтому в нем отсутствуют аналоги таких операторов SQL, как INS ERT и UPDATE. Данные можно модифицировать только через объектную модель встроенного языка программирования 1С. Также в языке запросов 1С существуют операторы, аналогов которых нет в SQL, например:

- В ИЕРАРХИИ;
- ПОМЕСТИТЬ;
- ИНДЕКСИРОВАТЬ ПО.

В ИЕРАРХИИ – позволяет выбрать все элементы иерархического справочника, которые входят в иерархию переданной ссылки. Пример запроса с использованием **В ИЕРАРХИИ**:

```
ВЫБРАТЬ
     Товары.Ссылка,
     Товары.Артикул
ИЗ
     Справочник.Товары КАК Товары
ГДЕ
     Товары.Ссылка В ИЕРАРХИИ(&Цитрусовые)"
```

В данном случае в результат вернутся все подчиненные элементы справочника номенклатуры «Цитрусовые», неважно, сколько уровней иерархии есть у данного справочника.

ПОМЕСТИТЬ – Данный оператор помещает результат во временную таблицу. Пример запроса:

```
ВЫБРАТЬ
	Пользователи.Ссылка КАК Ссылка,
	Пользователи.Родитель КАК Родитель,
	Пользователи.Наименование КАК Наименование
ПОМЕСТИТЬ ОтобранныеПользователи
ИЗ
	Справочник.Пользователи КАК Пользователи
ГДЕ
	Пользователи.Ссылка = &Справочник;
ВЫБРАТЬ
	ОтобранныеПользователи.Ссылка КАК Ссылка,
	ОтобранныеПользователи.Родитель КАК Родитель,
	ОтобранныеПользователи.Наименование КАК Наименование
ИЗ
	ОтобранныеПользователи КАК ОтобранныеПользователи
```

Данный запрос на SQL будет выполнен несколькими запросами:

- создание временной таблицы (платформа умеет «переиспользовать» ранее созданные временные таблицы, поэтому создание происходит не всегда);
- размещение данных во временную таблицу;
- выполнение основного запроса, а именно SEL ECT из этой временной таблицы;
- уничтожение/очистка временной таблицы.

Временная таблица может быть уничтожена явно, через конструкцию **УНИЧТОЖИТЬ**, либо неявно – при закрытии менеджера временных таблиц.

У объекта ***«Запрос»*** встроенного языка программирования есть свойство ***«МенеджерВременныхТаблиц»***, которое предназначено для работы с временными таблицами. Пример кода:

```
МВТ = Новый МенеджерВременныхТаблиц();
Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МВТ;
```

После выполнения запроса переменную МВТ можно использовать второй раз в другом запросе, что, несомненно, является еще одним плюсом использования временных таблиц. В данном случае временная таблица из

базы будет удалена при вызове метода «Закрыть»: ***МВТ.Закрыть();*** или при очистке переменной из памяти, то есть при выполнении метода, в котором переменная была объявлена. Временные таблицы повышают нагрузку на дисковую подсистему, поэтому не следует создавать очень много временных подсистем (в цикле, например), или подсистем большого объема.

ИНДЕКСИРОВАТЬ ПО – этот оператор применяется совместно с оператором **ПОМЕСТИТЬ.** При создании временной таблицы этим оператором можно проиндексировать создаваемую таблицу, что существенно ускоряет работу с ней (но только, если индекс подходит под ваш запрос).

Контроль отрицательных остатков

При проведении некоторых видов документов (например, «Реализация товара») необходимо организовать контроль остатков. Если товара на остатках недостаточно, документ не проводится и выдается диагностическое сообщение.

Для реализации контроля остатков записи будут списываться, а далее будет проводиться проверка, образовались ли отрицательные остатки по товарам. При получении отрицательных остатков будет проводиться отмена проведения документа. Таким образом, требуется провести документ и проверить остатки в регистре накопления.

Процедура ***ОбработкаПроведения()*** расположена в «Модуле объекта» документа. Алгоритм будет включать в себя: получение списка записей из табличной части, формирование движения по регистру, получение остатков из регистра. Для получения данных из табличной части документа и регистра данных используются запросы. Для формирования запроса можно использовать «Конструктор запроса».

Рассмотрим модификацию процедуры ***ОбработкаПроведения()*** для документа «СписаниеТовараСоСклада» пошагово.

1. Вызвать модуль объекта документа «СписаниеТовараСоСклада»
2. В обработке проведения создать запрос на получение списка товаров из табличной части документа «СписаниеТовараСоСклада», используя конструктор запроса:
   a. На вкладке Таблицы и поля перенести в колонку таблицы табличную часть документа «Товары», а в колонку поля – наименование (СписаниеТовараСоСкладаТовары.Наименование) и количество (СписаниеТовараСоСкладаТовары.Количество).
   b. На вкладке Группировка в групповое поле перемести «СписаниеТовараСоСкладаТовары.Наименование», а в

суммируемое поле – «СписаниеТовараСоСкладаТовары.Количество» – с функцией *Сумма*.

c. На вкладке Условия перенести вправо поле «Ссылка».
d. На вкладке Дополнительно выбрать тип запроса Создание временной таблицы, указав имя временной таблицы: «ДокТЧ».
e. На вкладке Пакет запросов переименовать «Запрос пакета 1» в «ДокТЧ» и добавить новый запрос. В новом запросе из временной таблицы ДокТЧ перенести в колонку поля – «ДокТЧ.Наименование» и «ДокТЧ.Количество». Общий вид получения списка товаров будет иметь вид:

```
//Получение списка товаров из документа
     Запрос= Новый Запрос;
      Запрос.МенеджерВременныхТаблиц = Новый
МенеджерВременныхТаблиц;
      Запрос.Текст =     "ВЫБРАТЬ
      |   СписаниеТовараСоСкладаТовары.Наименование,
      |   СУММА(СписаниеТовараСоСкладаТовары.Количество) КАК
Количество
      |ПОМЕСТИТЬ ДокТЧ
      |ИЗ
      |   Документ.СписаниеТовараСоСклада.Товары КАК
СписаниеТовараСоСкладаТовары
      |ГДЕ
      |   СписаниеТовараСоСкладаТовары.Ссылка = &Ссылка
      |
      |СГРУППИРОВАТЬ ПО
      |   СписаниеТовараСоСкладаТовары.Наименование
      |;
      |
      |ВЫБРАТЬ
      |  ДокТЧ.Наименование,
      |  ДокТЧ.Количество
      | ИЗ
      | ДокТЧ КАК ДокТЧ";
     Запрос.УстановитьПараметр("Ссылка",Ссылка);
     РезультатЗапроса=Запрос.Выполнить();
```

3. Для Формирования движения по регистру добавить обход выборки:

```
//Формирование движения
```

```
// регистр Склад Расход
Движения.Склад.Записывать = Истина;
Выборка = РезультатЗапроса.Выбрать();
Пока Выборка.Следующий() Цикл
    Движение = Движения.Склад.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Номенклатура = Выборка.Наименование;
    Движение.Количество = Выборка.Количество;
КонецЦикла;
Движения.Записать();
```

4. Для получения остатков из регистра построить запрос, получающий из регистра сведений СкладОстатки поля Товар и Количество. При этом на вкладке Условия Конструктора запросов добавить условие СкладОстатки.Количество<0:

```
//Получение остатков из регистра
    Запрос.Текст ="ВЫБРАТЬ
                  |   СкладОстатки.Номенклатура,
                  |   СкладОстатки.КоличествоОстаток
                  |ИЗ
                  |
    РегистрНакопления.Склад.Остатки(,Номенклатура В (ВЫБРАТЬ
ДокТЧ.Наименование ИЗ ДокТЧ КАК ДокТЧ)) КАК СкладОстатки
                  |ГДЕ
                  |   СкладОстатки.КоличествоОстаток < 0";
    РезультатЗапроса = Запрос.Выполнить();
```

5. Затем в цикле производится обход результатов запроса и выводится сообщение:

```
//Обработка результата остатков из регистра
    Выборка = РезультатЗапроса.Выбрать();
    Пока Выборка.Следующий() Цикл
        Сообщение=Новый СообщениеПользователю;
        Сообщение.Текст ="Не хватает товара "+
Выборка.Номенклатура +", после проведения документа остаток
составит "+ Выборка.КоличествоОстаток;
        Сообщение.Сообщить();
       Отказ =Истина;
    КонецЦикла;
```

Программное создание элемента справочника, внесение изменений в элемент справочника

При сохранении элементов справочника или проведение документов может возникнуть необходимость перемещения или создания нового элемента другого справочника.

Элемент справочника для изменения можно найти по реквизиту, наименованию или коду. Для этого можно использовать запросы или методы объекта Справочники ***НайтиПоКоду("<Код>")***, ***НайтиПоНаименованию("<Наименование>")***, ***НайтиПоРеквизиту("<Реквизит>", <ЗначениеРеквизита>)***. Данные методы возвращают ссылку на запись справочника. Например:

```
ССылкаНаКлиента = Справочники.Клиенты.НайтиПоКоду("000000001");
```

Для работы с полученной записью требуется по ссылке получить объект - ***ПолучитьОбъект()***. В полученном объекте можно изменять значения реквизитов. После изменения экземпляра справочника требуется записать изменения на сервер. Например:

```
СотрудникОбъект = СправочникСсылка.ПолучитьОбъект();
СотрудникОбъект.Родитель = Справочники.Сотрудники.Работающие;
СотрудникОбъект.Записать();
```

Создание элемента справочника производится через новый объект типа справочники. При этом прямое объявление не обязательно. Например:

```
Спр = Справочники.Сотрудники;
НовЭл = Спр.СоздатьЭлемент();
```

Дальнейшая работа с объектом аналогична работе с имеющимся справочником.

Рассмотрим создание элемента справочника на примере проведения документа «ПриказОПриемеНаРаботу» (используемые структуры объектов не соответствуют созданным при выполнении работ, код приведен для примера!). Справочник «Сотрудники» является иерархическим и имеет две предопределенных группы – «Работающие» и «Уволенные». В данном примере при проведении документа об увольнении сотрудник переносится в соответствующую группу. Поэтому при проведении документа о приеме на работу проверяется именно расположение элемента сотрудника в иерархии. Такое решение не является абсолютно корректным – в этом случае логично вести регистр сведений. Кроме того, данный пример не учитывает возможности совмещения и смены паспорта.

Документ «ПриказОПриемеНаРаботу» и справочник «Сотрудники» содержат реквизиты «Фамилия», «Имя», «Отчество», «Должность», «СерияИНомерПаспорта».

Создание нового элемента справочника содержится в модуле объекта документа в обработчике ***ОбработкаПроведения()***.

Первый запрос получает данные текущего документа. Связь с текущим документом обеспечивает параметр запроса Ссылка. Параметр запроса устанавливается до выполнения запроса: ***Запрос.УстановитьПараметр("Ссылка",Ссылка);***.

После этого в цикле через запрос происходит получение данных из справочника «Сотрудники» по номеру паспорта, при обходе данной выборки устанавливаются параметры «ПоказательНаличияСотрудника» и «ПоказательУвольненияСотрудника». «ПоказательналичияСотрудника» останется ***Ложь***, если выборка не содержит записей, т.е. такого сотрудника не было в справочнике «Сотрудники». Если Родитель (группа иерархического справочника) = уволенные, то ***ПоказательУвольненияСотрудника = Истина***.

Если сотрудник уже есть в справочнике Сотрудники, то в зависимости от группы – если он находится в папке Уволенные, то программно Родитель меняется на Работающие (запись в справочнике переносится в другую папку). Если запись уже находится в группе Работающие, то документ не проводится.

Если такого сотрудника нет в справочнике – то создается новый элемент, в реквизиты которого вносятся данные из документа. Слева в обходе выборки – реквизиты Справочника «Сотрудники», справа – реквизиты Документа «ПриемНаРаботу» из первоначальной выборки. (В запросе поля можно переименовывать, задавая им псевдонимы).

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
     Запрос = Новый Запрос;
     Запрос.Текст = "ВЫБРАТЬ
          | ПриказОПриемеНаРаботу.Фамилия КАК Фамилия,
          | ПриказОПриемеНаРаботу.Имя КАК Имя,
          | ПриказОПриемеНаРаботу.Отчество КАК Отчество,
          | ПриказОПриемеНаРаботу.Должность КАК Должность,
          | ПриказОПриемеНаРаботу.СерияИНомерПаспорта КАК
     СерияИНомерПаспорта
          |ИЗ
```

```
      | Документ.ПриказОПриемеНаРаботу КАК
ПриказОПриемеНаРаботу
      |ГДЕ
      | ПриказОПриемеНаРаботу.Ссылка = &Ссылка";
Запрос.УстановитьПараметр("Ссылка",Ссылка);
Результат = Запрос.Выполнить();
Выборка = Результат.Выбрать();
Пока Выборка.Следующий() Цикл
      Запрос.Текст = "ВЫБРАТЬ
            | Сотрудники.НомерИСерияПаспорта КАК
      НомерИСерияПаспорта,
            | Сотрудники.Родитель КАК Родитель
            |ИЗ
            | Справочник.Сотрудники КАК Сотрудники
            |ГДЕ
            | Сотрудники.НомерИСерияПаспорта =
      &НомерИСерияПаспорта";
      Запрос.УстановитьПараметр("НомерИСерияПаспорта",Выборка
.СерияИНомерПаспорта);
      Результат = Запрос.Выполнить();
      ВыборкаИзСправочника = Результат.Выбрать();
      ПоказательНаличияСотрудника = Ложь;
      ПоказательУвольненияСотрудника = Ложь;
      Пока ВыборкаИзСправочника.Следующий() Цикл
      ПоказательНаличияСотрудника = Истина;
      Если ВыборкаИзСправочника.Родитель =
Справочники.Сотрудники.Уволенные Тогда
            ПоказательУвольненияСотрудника = Истина;
      КонецЕсли
КонецЦикла;
Если ПоказательНаличияСотрудника=Истина Тогда
      Если ПоказательУвольненияСотрудника = Истина Тогда
            СправочникСсылка
      =Справочники.Сотрудники.НайтиПоРеквизиту("НомерИСерияПа
      спорта", Выборка.СерияИНомерПаспорта);
            СотрудникОбъект =
      СправочникСсылка.ПолучитьОбъект();
            СотрудникОбъект.Родитель =
      Справочники.Сотрудники.Работающие;
            СотрудникОбъект.Записать();
Иначе
            СообщениеПользователю = Новый
      СообщениеПользователю;
```

```
                    СообщениеПользователю.Текст = "Сотрудник
              "+Выборка.Фамилия +" с паспортными данными "
              +Выборка.СерияИНомерПаспорта+" уже есть в справочнике";
                    СообщениеПользователю.Сообщить();
                    Отказ = Истина;
              КонецЕсли
        Иначе
              Спр = Справочники.Сотрудники;
              НовЭл = Спр.СоздатьЭлемент();
              НовЭл.Фамилия = Выборка.Фамилия;
              НовЭл.Имя = Выборка.Имя;
              НовЭл.Отчество = Выборка.Отчество;
              НовЭл.Должность = Выборка.Должность;
              НовЭл.НомерИСерияПаспорта =
        Выборка.СерияИНомерПаспорта;
              НовЭл.Наименование = Выборка.СерияИНомерПаспорта;
              НовЭл.Родитель = Справочники.Сотрудники.Работающие;
              //Запись нового элемента на сервер
              НовЭл.Записать();
              КонецЕсли
        КонецЦикла
КонецПроцедуры
```

Программное создание и проведение документа

Создание нового документа происходит с использованием методов класса Документы: ***СоздатьДокумент()***. Например:

```
ОбъектДокумента = Документы.ОказаниеУслуги.СоздатьДокумент();
ОбъектДокумента.Дата = ТекущаяДата();
ОбъектДокумента.Клиент = Клиент;
```

При программном создании документа и справочника важно помнить о заполнении стандартных реквизитов: дата, наименование и т.д.

При сохранении созданного или измененного документа на сервере следует учитывать свойство проводимости документа. Возможно сохранение без проведения или с отменой проведения, это устанавливается в параметрах метода ***Записать()*** через ***РежимЗаписиДокумента***. Например, ***ОбъектДокумента.Записать(РежимЗаписиДокумента.Проведение);***.

Если в обработке проведения прописано создание документа именно этого наименования, при проведении документа ***ОбработкаПроведения()*** выполняется для вновь созданного. А выполнение текущей обработки приостанавливается – как в рекурсивных функциях.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №6

1. Для запрета проведения документа «СписаниеТовараСоСклада», приводящего к отрицательным значениям в регистре «Склад» требуется изменить Обработку проведения (код обработчика приведен в теоретических сведениях)
2. Реализовать в системе объекты учитывающие увольнение сотрудника – создать документ УвольнениеСотрудникаи внести необходимые изменения в регистр накопления. Для документа создать обработку проведения, контролирующую остатки в регистре накоплений.

Практическая работа №7

1. Реализовать в системе объекты для учета расчетов с контрагентами. Система должна учитывать авансовые и окончательные платежи. Внести изменения в документы «СписаниеТовараСоСклада» и «ПоставкаТовара» для учета оплатных документов.
2. Реализовать регистры накопления рассчитывающие дебиторскую и кредиторскую задолженность на основе документов «ПоставкаТовара», «СписаниеТовараСоСклада» и документов об оплате (авансовых и окончательных платежей). Создать Журналы документов для удобства работы.
3. Реализовать обработку проведения документов «ПоставкаТовара» и «СписаниеТовараСоСклада», учитывающие задолженность.
4. Реализовать обработку проведения, для контроля оплаты (оплачиваться должен лишь товар, который поставлен или списан).

Практическая работа №8

1. Реализовать программное создание новых объектов справочников «ФизическиеЛица» и «Сотрудники» при проведении документа «ПриемНаРаботу». Справочники должны иметь иерархию групп и элементов и предопределенные группы – «Работающие» и «Уволенные». При Приеме на работу работающего человека, документ «ПриемНаРаботу» проводиться не должен.

2. Реализовать программный перенос объектов справочников в предопределенные группы при увольнении сотрудника и при приеме на работу. Пример приведен в теоретических сведениях.

Практическая работа №9

1. Для справочника «Сотрудники» реализовать команду (и кнопку на форме) позволяющую проводить документ «УвольнениеСотрудника». Следует учитывать, что при проведении документа «УвольнениеСотрудника» будет выполнен обработчик проведения – перенос элементов справочника (изменение поля Родитель) в иерархии групп и контроль остатков.
2. Спроектировать решение для автоматического создания и проведения документов авансовых и фактических платежей при выборе соответствующих позиций на форме документа «ПоставкаТовара».

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Что такое Конструктор запроса?
2. Как реализуется контроль отрицательных остатков?
3. Как обратиться к реквизитам Справочника по наименованию?
4. Что такое Схема компоновки данных?
5. Зачем используются временные таблицы в запросах?
6. Зачем используются виртуальные таблицы в запросах?
7. Что такое пакетный запрос?
8. Каким образом создать пакетный запрос в Конструкторе запросов?
9. Каким образом следует изменить логику проведения документов и формирования объектов конфигурации в работе №7 для реализации поставки только после авансовой оплаты?

Раздел 5. Отчеты

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Отчеты - это прикладные объекты конфигурации. Они предназначены для обработки накопленной информации и получения сводных данных в удобном для просмотра и анализа виде. Конфигуратор позволяет формировать набор различных отчетов, достаточных для удовлетворения потребности пользователей системы в достоверной и подробной выходной информации.

Как правило, для формирования выходных данных отчет использует систему компоновки данных. Но, вообще говоря, отчет может содержать произвольный алгоритм формирования «бумажного» или «электронного» отчета на встроенном языке.

Отчет может содержать одну или несколько форм, с помощью которых, при необходимости, можно организовать ввод каких-либо параметров, влияющих на ход алгоритма.

Система компоновки данных представляет собой механизм, основанный на декларативном описании отчетов. Он предназначен для построения отчетов, а также вывода информации, имеющей сложную структуру и содержащий произвольный набор таблиц и диаграмм.

Система компоновки данных позволяет реализовать следующие возможности:

- создание отчета без программирования;
- использование автоматически генерируемых форм просмотра и настройки отчета;
- разбиение исполнения отчета на этапы;
- исполнение отдельных этапов построения отчета на различных компьютерах;
- независимое использование отдельных частей системы компоновки данных;
- программное управление процессом выполнения отчета.

У объекта конфигурации Отчет реализовано свойство "Основная схема компоновки данных". При нажатии кнопки открытия для этого свойства, вызывается конструктор макета, который позволяет создать макет отчета, содержащий схему компоновки данных. После нажатия кнопки "Готово" будет открыт конструктор схемы компоновки данных.

Конструктор схемы компоновки данных позволяет описать исходные данные, которые будет использовать отчет: наборы данных, связи между наборами данных, вычисляемые поля, ресурсы и т.д.

Также конструктор схемы компоновки данных предоставляет возможность описать настройки компоновки данных, которые будут использоваться по умолчанию (в том случае, если пользователь не задаст собственные настройки). Настройки компоновки данных могут быть созданы с помощью специального конструктора настроек компоновки данных, или вручную.

Конструктор схемы компоновки данных (рисунок 5.1) позволяет разработчику полностью описать схему компоновки данных исключительно визуальными средствами. Перемещаясь по закладкам конструктора можно указывать, какие данные должны присутствовать в отчете, как они связаны, сгруппированы и какие ресурсы следует рассчитать и т.д.

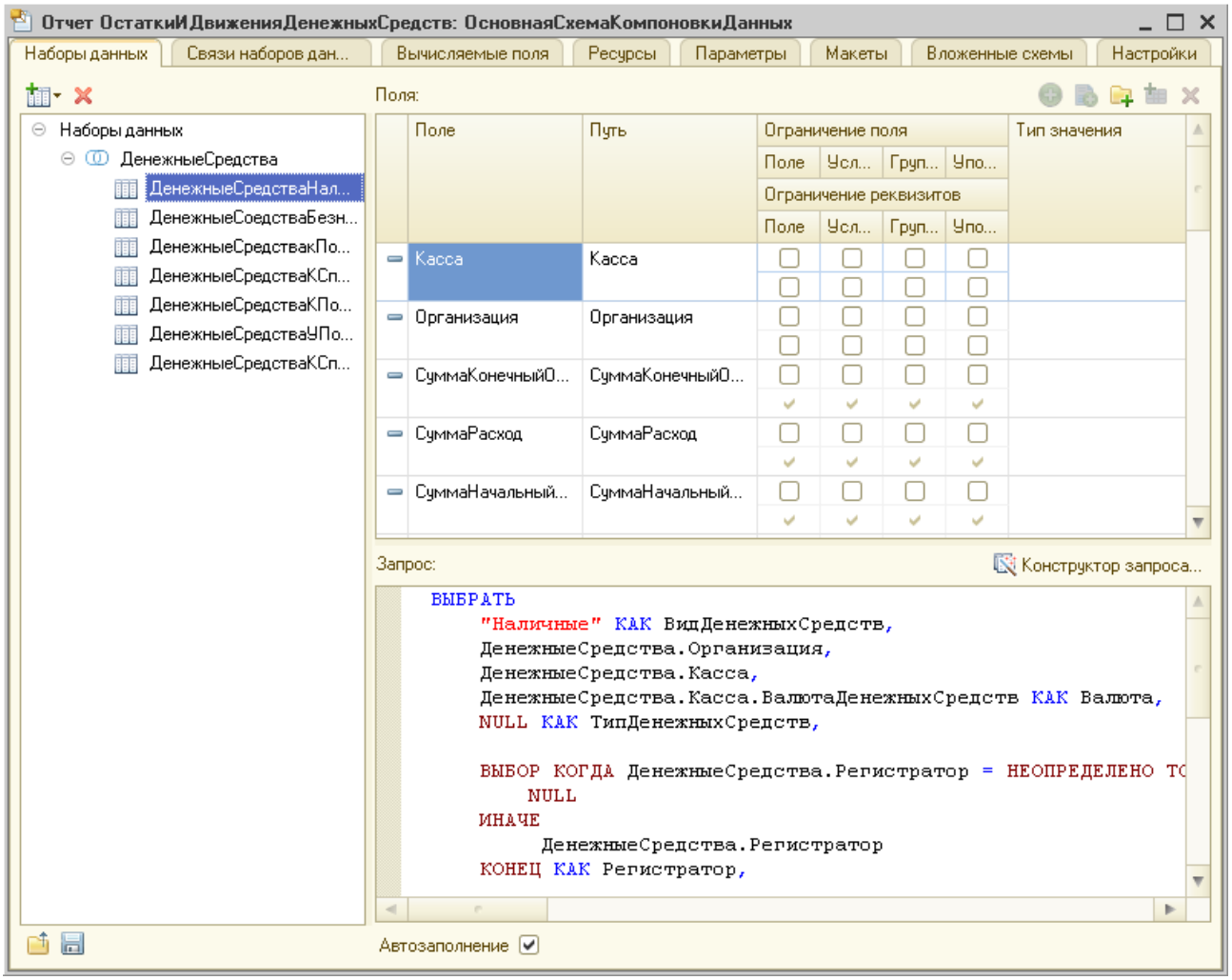


Рисунок 5.1 – Форма конструктора схемы компоновки данных

Конструктор настроек компоновки данных позволяет настроить отчет, созданный с использованием системы компоновки данных. Назначение

конструктора заключается в том, чтобы предоставить разработчику или пользователю возможность быстрой настройки типичных отчетов нескольких видов: список, таблица и диаграмма.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №10

1. Создать отчет «ПоследниеЦены». Открыть основную схему компоновки данных. В форме Конструктора схемы компоновки данных на вкладке Наборы данных добавить новый набор данных – запрос. Открыть Конструктор запроса.
2. В Конструкторе запроса перенести в колонку поля данные из регистра сведений «ПоследниеЦеныПоставки.СрезПоследних» поля «ПоследниеЦеныПоставкиСрезПоследних.Номенклатура» и «ПоследниеЦеныПоставкиСрезПоследних.Цена».Для сортировки данных по способу хранения или стране производителя можно добавить поля: «ПоследниеЦеныПоставкиСрезПоследних.Номенклатура.Способ Хранения» и «ПоследниеЦеныПоставкиСрезПоследних.Номенклатура.Страна Производитель».
3. На вкладке Ресурсы перенести вправо поле «Цена».
4. На вкладке Настройки задать несколько вариантов отчетов, для этого в левой колонке добавить новые строки. Для каждого варианта, используя Конструктор настройки компоновки данных.
    a. ОсновнойВариант. Тип отчета: список. Отображаемые поля: «Номенклатура», «Цена», «НоменклатураСпособХранения», «НоменклатураСтранаПроизводитель». Группировка отсутсвует. Упоряджочивание (сортировка): «Номенклатура».
    b. ТабличноеПредставлениеПоСтранам. Тип отчета: таблица. Отображаемые поля: «Номенклатура», «Цена». Группировка: строки: «НоменклатураСпособХранения»; таблицы: «НоменклатураСтранаПроизводитель». Упорядочивание: «НоменклатураСтранаПроизводитель», «Номенклатура».
    c. ДиаграммаЦеныНоменклатуры. Тип отчета: диаграмма. Отображаемые поля: «Цена». (Тип отчета диаграмма не

возможен без выбора ресурса в отчете) Группировка: серии: «Номенклатура».

d. ДиаграммаЦеныПоСтранам. Тип отчета: диаграмма. Отображаемые поля: «Цена». Группировка: серии: НоменклатураСтранаПроизводитель; точки: «Номенклатура». Упорядочивание: «НоменклатураСтранаПроизводитель». Тип диаграммы: гистограмма обыкновенная.

5. Реализовать в подсистемах «Склад» и «Клиенты» списание товара для выполнения заказанной услуги. Созданный документ «СписаниеТовараДляУслуги» должен создаваться на основании документа «ДоговорОбОказанииУслуг» и являться основанием для документа «СписаниеТовараСоСклада». Создать регистры сведений и накоплений требующиеся для контроля остатков и автоматизации расчетов и учета.
6. Изменить справочник услуги таким образом, чтобы количество необходимых товаров было указано для каждого элемента справочника. Внести необходимые изменения в документы «ДоговорОбОказанииУслуг» и «СписаниеТовараДляУслуги» для получения данных из справочника, используя Конструктор запросов.
7. Для документов создать печатные формы. Сформировать отчеты для наглядного представления данных в виде таблиц, списков и диаграмм.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Каким образом создать отчет в виде диаграммы?
2. Охарактеризуйте поля группировки для каждого типа отчета в Конструкторе настроек компоновки данных.
3. Каким образом добавить варианты отчета в Конструкторе компоновки данных?
4. Сформировать отчет по количеству сотрудников на каждой должности в виде гистограммы. Сформировать отчет, выводящий информацию о сотрудниках по каждому подразделению.

Раздел 6. Система прав доступа

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Система прав доступа позволяет описывать наборы прав, соответствующие должностям пользователей или виду деятельности. Структура прав определяется конкретным прикладным решением.

Кроме этого, для объектов, хранящихся в базе данных (справочники, документы, регистры и т.д.) могут быть определены права доступа к отдельным полям и записям. Например, пользователь может оперировать документами (накладными, счетами и т.д.) определенных контрагентов и не иметь доступа к аналогичным документам других контрагентов.

Среди действий над объектами, хранящимися в базе данных (справочниками, документами и т.д.), есть действия, отвечающие за чтение или изменение информации, хранящейся в базе данных. К таким действиям относятся:

- чтение - получение записей или их фрагментов из таблицы базы данных;
- добавление - добавление новых записей без изменения существующих;
- изменение - изменение существующих записей;
- удаление - удаление некоторых записей без внесения изменений в оставшиеся.

Для этих действий в процессе настройки ролей могут быть заданы дополнительные условия на данные (ограничение доступа к данным). В этом случае над конкретным объектом, хранимым в базе данных, может быть выполнено запрошенное действие только в том случае, если ограничение доступа к данным для данных этого объекта принимает значение "истина". Аналогичные условия могут быть заданы и для таблиц базы данных, не имеющих объектной природы (регистров).

Роли

Роли - это общие объекты конфигурации. Они предназначены для реализации ограничения прав доступа в прикладных решениях. Роль в конфигурации может соответствовать должностям или видам деятельности различных групп пользователей, для работы которых предназначена данная конфигурация (рисунок 6.1). Роль определяет, какие действия, над какими объектами метаданных может выполнять пользователь, выступающий в этой роли (рисунок 6.2).

Для редактирования состава ролей платформа содержит два редактора: Редактор роли и Редактор «Все роли».

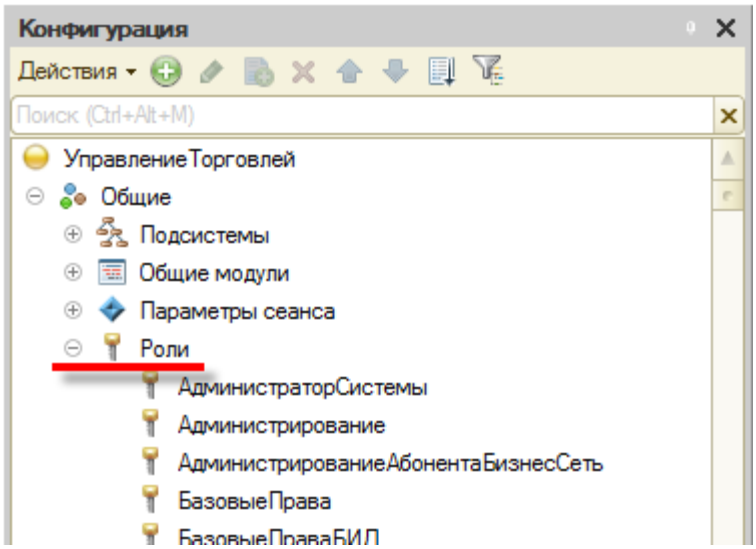


Рисунок 6.1. – Объект *Роли* в дереве конфигурации

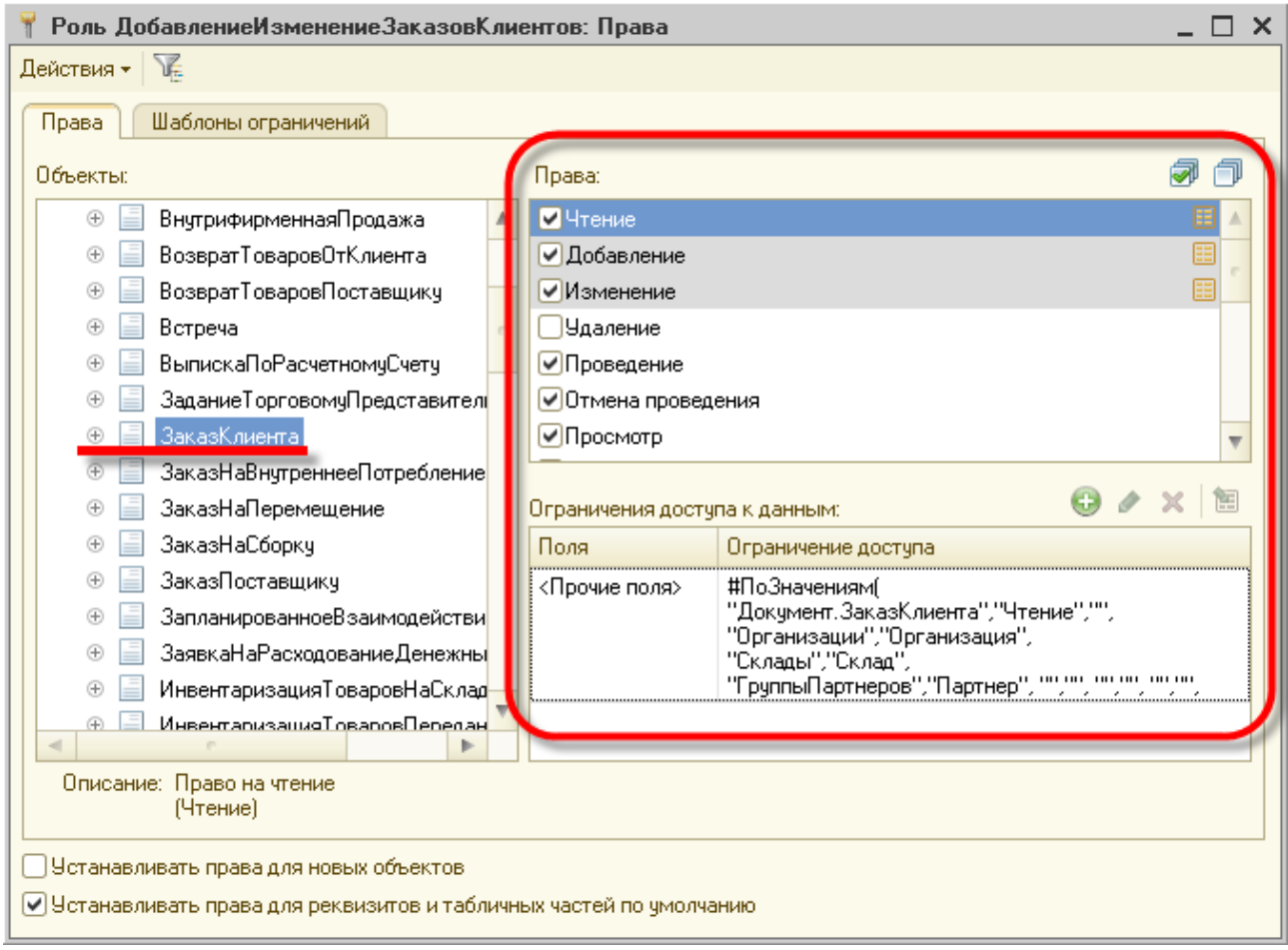


Рисунок 6.2 – Форма редактирования объекта *Роли*

Пользователи

Для создания пользователей в меню «*Администрирование*» следует выбрать пункт «*Пользователи*». В появившейся форме «*Список пользователей*» можно добавлять и редактировать список пользователей.

Окно редактирования пользователя содержит разделы «*Основные*» и «*Прочие*». В разделе «*Основные*» содержатся параметры аутентификации пользователя по паролю, операционной системе и OpenID. В процессе ведения списка пользователей прикладного решения каждому пользователю ставится в соответствие одна или несколько ролей (рисунок 6.3).

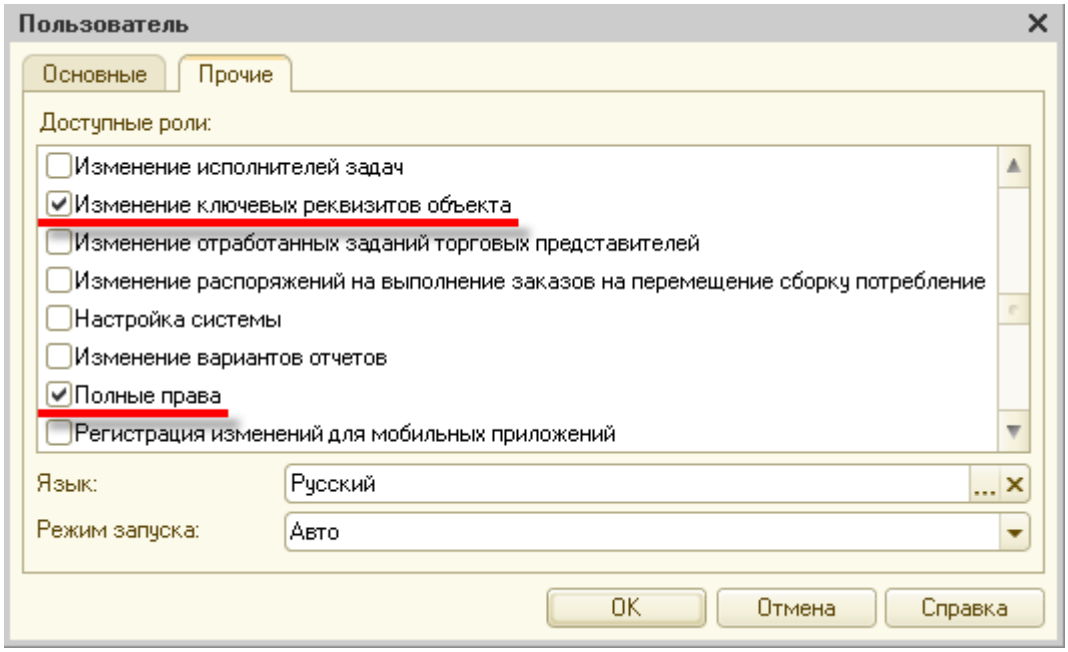


Рисунок 6.3 – Форма редактирования *Пользователь*

Меню «*Активные пользователи*» и «*Журнал регистрации*» содержат информацию по пользователям, в настоящий момент имеющим доступ к информационной базе и их действиям (рисунок 6.4).

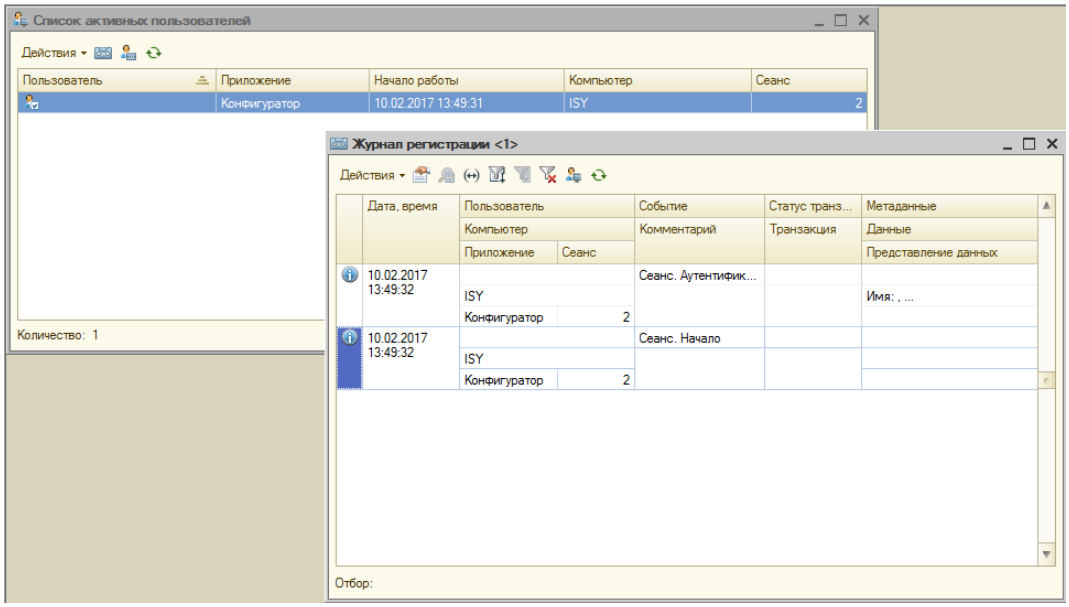


Рисунок 6.4 - Формы *Список активных пользователей* и *Журнал регистрации*

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №11

1. Определить роли пользователей: «Администратор», «Менеджер», «Директор». Роль «Администратор» должна обладать правами администрирования и изменения всех объектов конфигурации.
2. Настроить командный интерфейс подсистем и права на просмотр и изменение объектов конфигурации.
3. Создать пользователей. Пользователь с ролью «Администратор» должен быть создан первым. Перед созданием пользователей информационная база должна быть обновлена.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Что такое система ролей в конфигурации?
2. Каким образом организованна работа с журналом пользователей?

Раздел 7. Бизнес-процессы

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Механизм бизнес-процессов - это один из прикладных механизмов платформы. Он позволяет описывать, создавать и управлять выполнением бизнес-процессов в прикладных решениях.

Целью этого механизма является автоматизация цепочек связанных операций, направленных на достижение общей цели, обычно в контексте организационной структуры, определяющей функциональные роли и связи.

Этот механизм включает средства для описания схем бизнес-процессов, и их ролевой маршрутизации, средства для формирования заданий, выполняющихся в каждой точке маршрута, средства для управления бизнес-процессом и организации его связи с другими функциями прикладного решения.

Бизнес-процессы в "1С:Предприятие" нужны для того, чтобы объединять отдельные операции (выписка счета, прием наличной оплаты, отпуск товара со склада и т. д.) в цепочки взаимосвязанных действий, приводящих к достижению конкретной цели (например, продажа товара за наличный расчет). Участие сотрудников в жизненном цикле бизнес-процесса достигается ролевой маршрутизацией.

Механизм бизнес-процессов в 1С обеспечивается сразу несколькими объектами конфигурирования: бизнес-процессы, задачи, регистр сведений и параметр сессии. Как правило, типы реквизитов адресации задачи и измерений регистра сведений назначаются ссылками на соответствующие справочники, поэтому к четырем вышеперечисленным видам добавляются еще справочники.

Основные объекты механизма бизнес-процессов — это бизнес-процессы и задачи. Они используют друг друга и еще три вспомогательных объекта — параметр сеанса, регистр сведений и справочники. Вспомогательные объекты не используют ни друг друга, ни основные объекты.

Задача предназначена для учета заданий и описывает способ их распределения по исполнителям с учетом организационной структуры предприятия. Адресация заданий сотрудникам определяется реквизитами, в которых можно предусмотреть многомерную ролевую маршрутизацию, например по ролям, рабочим группам, подразделениям, помещениям, филиалам и т. д. При этом задачи могут создаваться не только бизнес-процессами, но и другими объектами информационной базы и

непосредственно пользователями. Более того, в общем случае исполнителем задания может быть не только сотрудник, но и любая внешняя система, например другая учетная система.

Понятие задачи фактически определяет лишь интерфейс взаимодействия бизнес-процесса с заданием, выполнение которого может быть, в общем случае, не связано с выполнением операций в самой системе. Например, бизнес-процесс по ходу своего выполнения может потребовать согласования какого-то вопроса с руководителем фирмы. Сформулированная таким образом задача будет, к примеру, адресована секретарю, который станет решать ее любыми доступными ему способами: по электронной почте, по телефону и т. д. Задача будет считаться выполненной, когда в систему поступят сведения о получении нужного согласования.

Объект "Бизнес-процесс" описывает логику выполнения операций для достижения той или иной цели и управляет жизненным циклом созданных бизнес-процессов (их экземпляров) от момента старта до момента завершения. Логика бизнес-процесса (взаимосвязь и последовательность обхода точек маршрута, условные переходы и пр.) наглядно описывается в виде карты маршрута, которая позволяет визуально описывать маршрут бизнес-процесса в виде связного графа и позволяет легко описывать алгоритмы условных переходов и реакцию бизнес-процесса на различные события.

Операции, выполняемые в ходе бизнес-процесса, представлены на карте маршрута точками действий, которые содержат информацию о том, кто и что должен сделать на данном этапе. Исполнитель может определяться персонально (Иванов) или с учетом ролевой маршрутизации («Кладовщик», "Руководитель отдела продаж"). При переходе бизнес-процесса на точку действия он автоматически формирует задачи, устанавливая в них предусмотренные реквизиты адресации. После того как исполнитель отметит задачу как выполненную, бизнес-процесс автоматически переходит к следующей точке маршрута в соответствии с картой.

В точке действия возможно также назначение групповых и коллективных задач. В первом случае действие должны выполнить все члены группы, — например, когда всем менеджерам нужно предоставить ежемесячный отчет. Во втором — действие должен выполнить только один из членов группы (например, завизировать документ у одного из старших менеджеров). В точке действия можно описать проверку необходимых условий выполнения задачи, интерактивный диалог с пользователем при

переходе далее по маршруту и указать, например, какие документы следует открывать при активации задач, связанных с этой точкой маршрута бизнес-процесса.

Механизм бизнес-процесса в 1С допускает несколько видов маршрутизации.

• Жесткая. Бизнес-процесс имеет карту, не содержащую условных и параллельных переходов с жестко определенными адресатами для каждой точки маршрута. Отклонение таких бизнес-процессов не допускается.

• Свободная. Адресаты точки карты маршрута бизнес-процесса не установлены и определяются программно или интерактивно в течение жизненного цикла бизнес-процесса.

• Условная. Карта маршрута предусматривает проверку условий и переход по соответствующим ветвям. Переходы могут быть как бинарными (условие), так и множественными (выбор варианта)

• Параллельная. Карта маршрута предусматривает разделение бизнес-процесса на параллельные ветви с возможностью последующего слияния (ожидания). Продвижение бизнес-процесса по каждой из параллельных ветвей происходит независимо, по мере выполнения соответствующих задач.

Как правило, в реальных картах бизнес-процессов встречаются все эти типы маршрутизации.

Карта маршрута

Прохождение в 1С бизнес процесса отображается посредством графической блок-схемы, называемой картой маршрута, которая дает наглядное представление, что, в каком порядке, при выполнении каких условий происходит. Карта маршрутов бизнес-процессов разделена на этапы. Этап в 1С отделен точкой маршрута, в которой нужно выполнить определенную задачу. Задача – это также объект карты маршрута в программе 1С. В задаче указывается исполнитель (или исполнители), кому адресована эта задача, сроки выполнения и важность. Исполнители – пользователи 1С. Адресатом задачи может быть назначен конкретный сотрудник, один из участников рабочей группы (отдела, подразделения) или сотрудник, занимающий определенную должность (например, кассир, директор, кладовщик).

Карта (рисунок 7.1) начинается с пункта Старт, без которого бизнес-процесс не может быть начат (стартован). Точек старта может быть несколько, но в нашем примере условие выбора появляется после него, и продолжение маршрута зависит от результата заключения сделки.

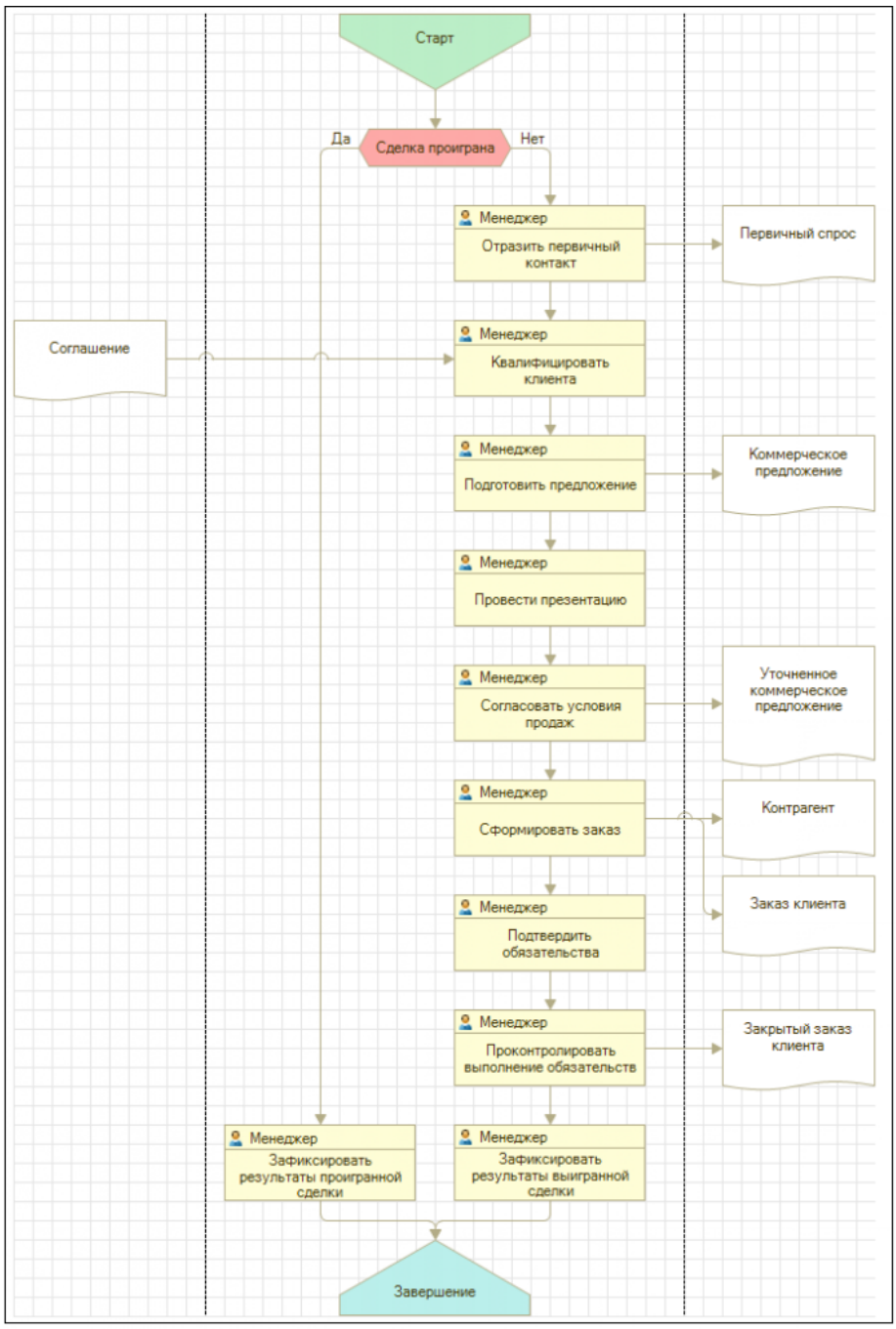


Рисунок 7.1 – Пример карты маршрута

Далее на блок-схеме присутствуют желтые прямоугольники – точки маршрута с указанием в них сотрудника, который должен выполнить поставленную задачу. Все завершенные задачи будут заштрихованы. Конечная точка – Завершение. Белые прямоугольные сноски – справки – пояснение к точкам маршрута.

Адресация

Под адресацией обычно подразумевают объект, которому поручается конкретная задача. Адресация может быть как жесткой, в этом случае объект адресации назначается при ее формировании, так и произвольной, в этом случае задаче назначается не конкретный объект адресации, а, например, его

роль, должность или иное значение, косвенно обозначающее круг объектов адресации, для которых формируется задача.

Для описания правил адресации используют регистр сведений. Для назначения адресации система ориентируется на измерения этого регистра, ресурсы и реквизиты самой системой для адресации не используются, хотя и могут присутствовать в регистре. Одним из измерений регистра должно быть измерение, хранящее конкретных исполнителей, дополнительные измерения будут использоваться для произвольной адресации. На сегодняшний момент на уровне системы не поддерживается периодика адресации. То есть регистр сведений, хранящий правила адресации, должен быть не периодическим.

Пример адресации (рисунок 7.2): если в качестве исполнителя задачи при ее формировании указывается конкретный объект адресации (сотрудник, пользователь системы), то в любом случае назначен будет он. Если конкретный исполнитель не указан, то вступает в действие механизм произвольной адресации. Система ориентируется на соответствие измерений регистра. Если в регистре адресации два измерения (одно для исполнителя, и еще одно для какого-либо признака адресации - например, подразделение), то задача будет назначена всем исполнителям, для которых в регистре есть записи с дополнительным признаком адресации.

Исполнитель	Подразделение
Иванов	Администрация
Петров	Администрация
Сидоров	Отдел продаж
Иванюхин	Отдел рекламации

Рисунок 7.2 – Пример регистра сведений для адресации задач

Иногда важно иметь возможность назначать задачи исполнителям, которые работают с конкретными контрагентами и их контактными лицами. Тогда регистр сведений будет иметь несколько измерений.

Так как задачи создаются ради их назначения конкретным исполнителям, работающим с системой, необходимо своевременно оповещать пользователя о появлении новой задачи. Для этого система должна "знать" вошедшего пользователя. Ссылка на текущего пользователя должна храниться в параметре сеанса, значение которого необходимо инициализировать при старте системы. Кроме этого, так как в регистре адресации может быть несколько измерений, системе важно указать в котором из них необходимо искать пользователя-исполнителя.

Адресация задач Объект «Задача» предоставляет возможность использования вспомогательного регистра сведений, который обеспечивает распределение задач по исполнителям. Этот регистр называется регистром

адресации. Измерениями регистра адресации должны выступать те значения, по которым возможно однозначно определить исполнителя задачи. При этом в качестве одного из измерений чаще всего используется справочник «Пользователи», так как с помощью него удобно связать текущего пользователя программы с его задачами. Однако использование справочника «Пользователи» как единственного измерения адресации на практике неудобно: например, конкретный пользователь может заболеть или сменить место работы, тогда придется перенастраивать адресацию задач. Таким образом, задачи удобнее привязывать не напрямую к пользователю, а к набору ролей, и одного измерения адресации чаще всего недостаточно.

Общая схема создания бизнес-процесса в 1С

1. Создаем регистр адресации
    a. Создаем формы
2. Создаем задачу
    a. Заполняем вкладку адресация
    b. Данные заполняем реквизитами, передаваемыми между задачами и самим бизнес-процессом
    c. Создаем формы
3. Создаем бизнес-процесс
    a. Заполняем задачу, реквизиты, создаем формы
    b. Рисуем карту маршрута

Порядок выполнения обработчиков бизнес-процесса.

1. Форма: перед выполнением
2. Форма: перед записью (сначала на клиенте, затем на сервере)
3. Модуль задачи: перед выполнением
4. Бизнес-процесс: перед выполнением
5. Модуль задачи: при выполнении
6. Модуль задачи: перед записью
7. Модуль задачи: при записи
8. Бизнес-процесс: при выполнении
9. Форма: после записи (сначала на сервере, затем на клиенте)
10. Интерактивные процедуры не выполняются в управляемом режиме.

Создание бизнес-процесса

Создадим в регистре адресации 2 измерения: «Исполнитель» и «РольИсполнителя». Здесь используется справочник «Роли исполнителей», который имеет следующие предопределенные элементы: МладшийКадровик,

Расчетчик и СтаршийКадровик. Создадим у задачи реквизиты адресации тех же типов, что и измерения регистра сведений (рисунок 7.3).

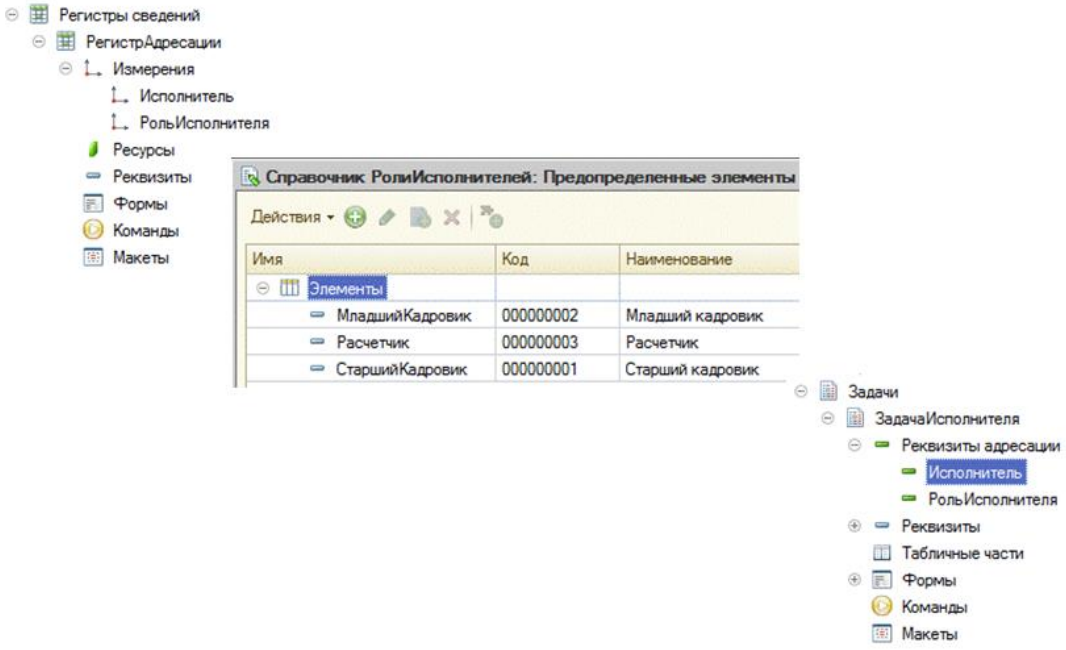


Рисунок 7.3 – Настройка адресации при создании бизнес-процесса

Укажем для задачи регистр адресации. Для реквизитов адресации задачи настроим соответствие измерениям выбранного регистра сведений (рисунок 7.4). Теперь в карте маршрута бизнес-процесса доступна настройка адресации.

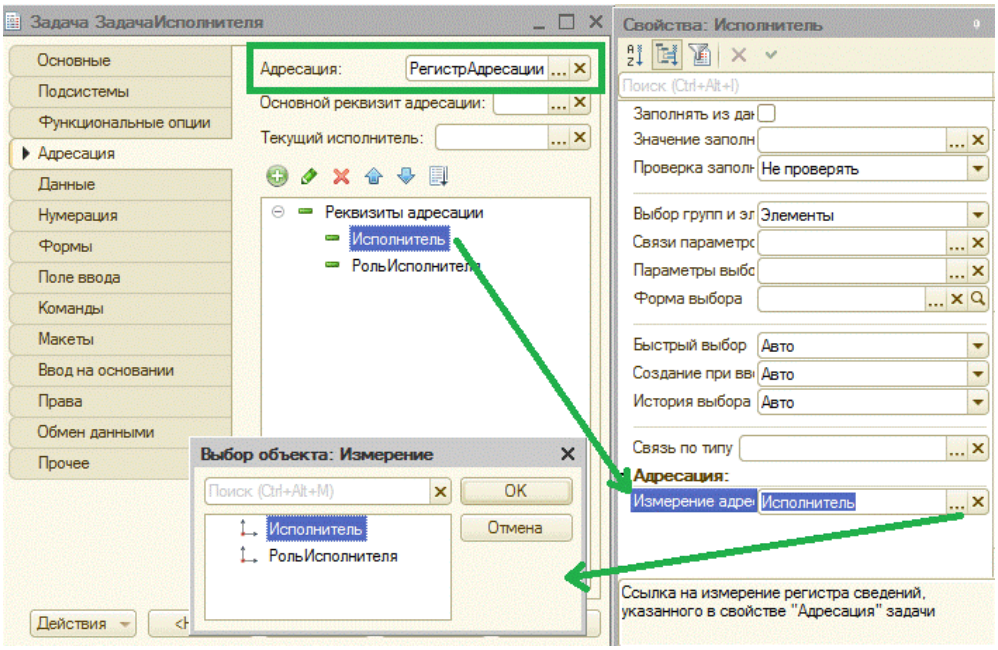


Рисунок 7.4 – Настройка реквизитов адресации в задаче

Объект «Задача» может использоваться отдельно от объекта «Бизнес-процесс», но никак не наоборот. Используемая задача указывается в настройках бизнес-процесса в конфигураторе, и будет автоматически

создаваться в пользовательском режиме при переходе на следующую точку маршрута. На рисунке 7.5 приведен процесс связи бизнес-процесса и задачи. Несмотря на то, что в разных точках бизнес-процесса могут создаваться и редактироваться различные объекты системы, тип создаваемых задач на каждом из этапов будет одинаковым. Например, для решаемой задачи будут создаваться элементы справочников “Физические лица” и “Сотрудники”, а также документы “Прием на работу”. При этом в каждой точке бизнес-процесса будут создаваться задачи “Задача исполнителя”. Более того, в типовых решениях 1С («Документооборот», «Управление торговлей, ред. 11») в разных бизнес-процессах используется один и тот же тип задач, чаще всего он называется «Задача исполнителя». Это делается для того, чтобы пользователь мог видеть общий список своих задач, относящихся к разным видам бизнес-процессов, как в примере выше из «Документооборота».

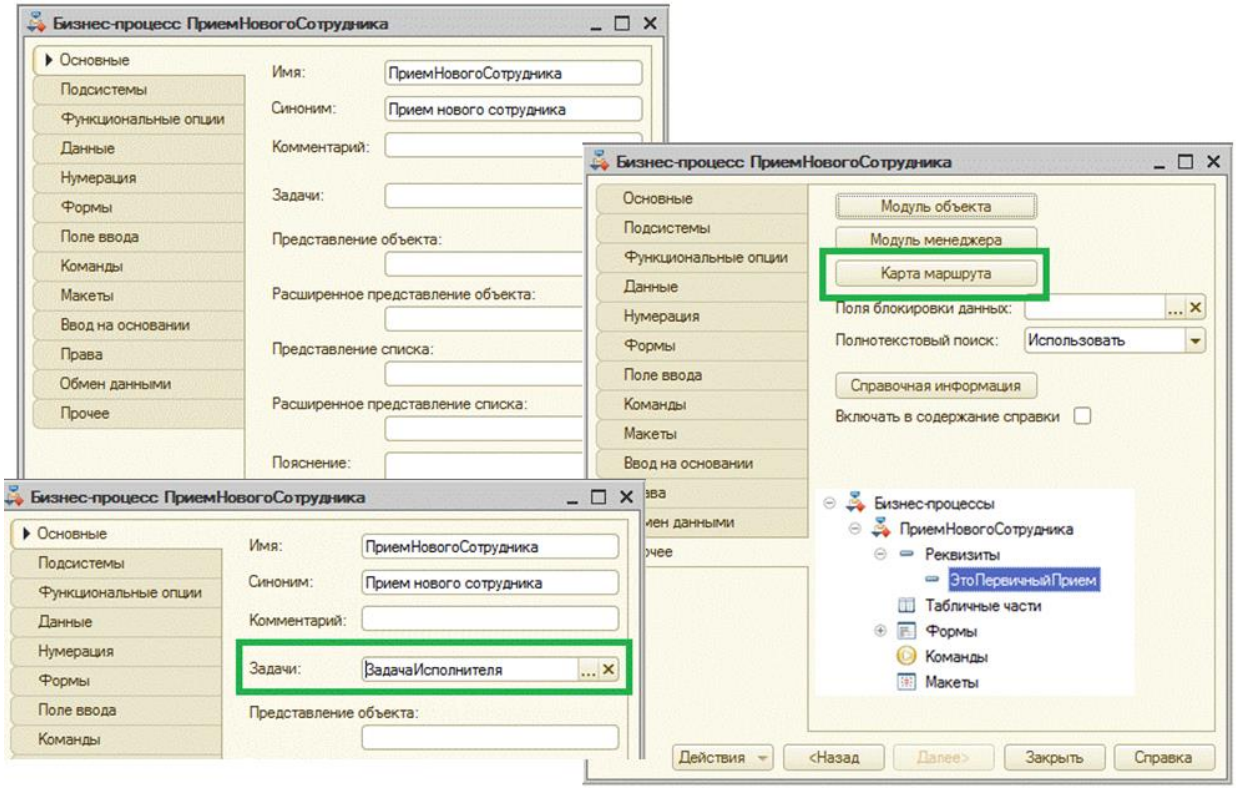


Рисунок 7.5 – Связь объектов «Бизнес-процесс» и «Задача»

В пользовательском режиме новый бизнес-процесс будет создаваться каждый раз при приеме нового сотрудника. Изобразим карту маршрута бизнес-процесса для поставленной задачи (рисунок 7.6).

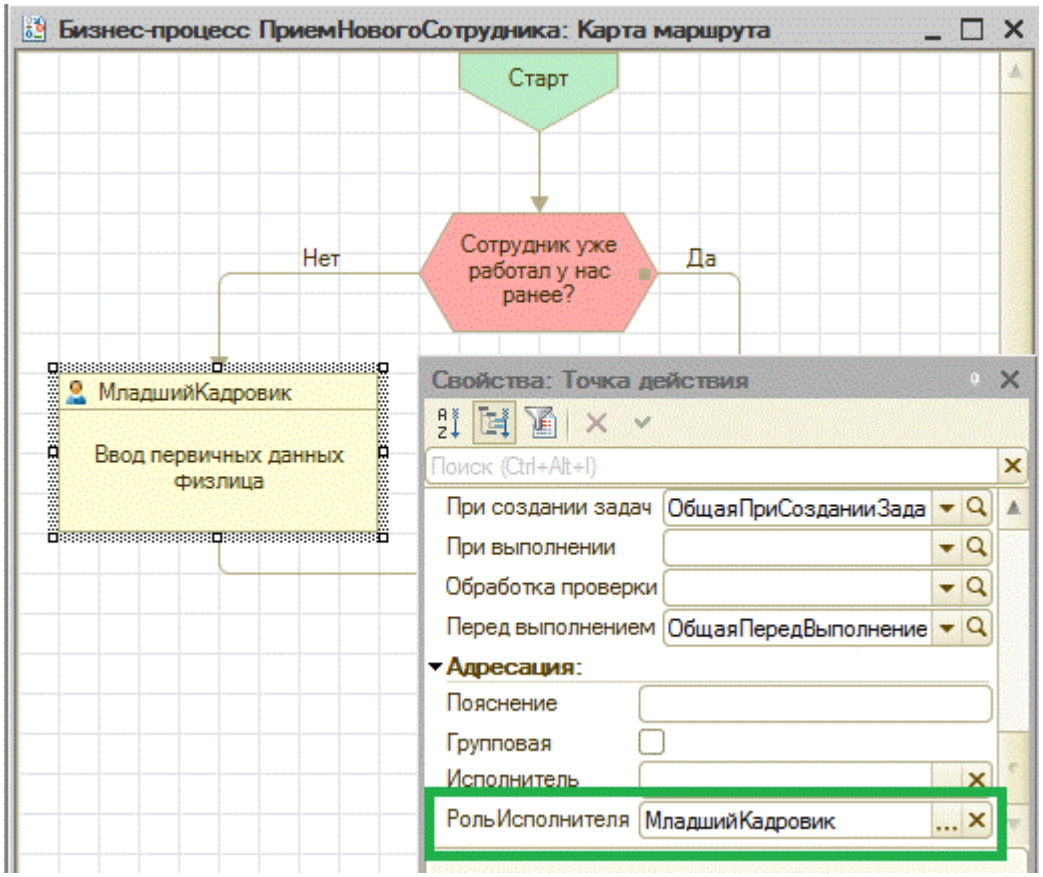


Рисунок 7.6 – Карта маршрута бизнес-процесса «ПриемНовогоСотрудника»

В точке условия нужно определить, работал ли принимаемый сотрудник в нашей организации ранее. Если это так, то в базе уже заведен нужный элемент справочника «Физические лица» и заполнены личные данные. Добавим в бизнес-процесс реквизит (тип Булево), который позже обработаем в точке условия (то есть считаем, что пользователь сам определяет при приеме – новый это сотрудник или нет). Теперь, когда у нас есть схема бизнес-процесса, необходимо организовать последовательность действий и распределение этих действий по ответственным лицам.

Чтобы знать, какой пользователь запустил текущий сеанс, потребуется параметр сеанса. На рисунке 7.7 проиллюстрирован процесс создания и настройки параметра сеанса. Сделаем так, чтобы при запуске «1С: Предприятие» в параметр сеанса ТекущийПользователь подбиралось нужное значение из справочника «Пользователи». Соответствие будем устанавливать по имени, и если элемент справочника «Пользователи» с нужным именем не найден, то создадим его. Код функции УстановкаПараметровСеанса() в модуле сеанса:

```
Процедура УстановкаПараметровСеанса(ТребуемыеПараметры)
    ИмяПольз = ИмяПользователя();
```

```
    ТекПользователь =
Справочники.Пользователи.НайтиПоНаименованию(ИмяПольз, Истина);
     Если Не ЗначениеЗаполнено(ТекПользователь) Тогда
          НовыйПользователь =
    Справочники.Пользователи.СоздатьЭлемент();
          НовыйПользователь.Наименование = ИмяПольз;
          НовыйПользователь.Код = ИмяПольз;
          НовыйПользователь.Записать();
          ТекПользователь = НовыйПользователь.Ссылка;
    КонецЕсли;
    ПараметрыСеанса.ТекущийПользователь = ТекПользователь;
КонецПроцедуры
```

После того, как все нужные элементы справочника «Пользователи» будут созданы, заполним регистр адресации в режиме «1С:Предприятие». Настройка адресации бизнес-процесса с использованием справочника (в нашем случае это справочник «Роли исполнителей») подразумевает использование предопределенных элементов этого справочника, только они доступны для выбора в точке действия бизнес-процесса.

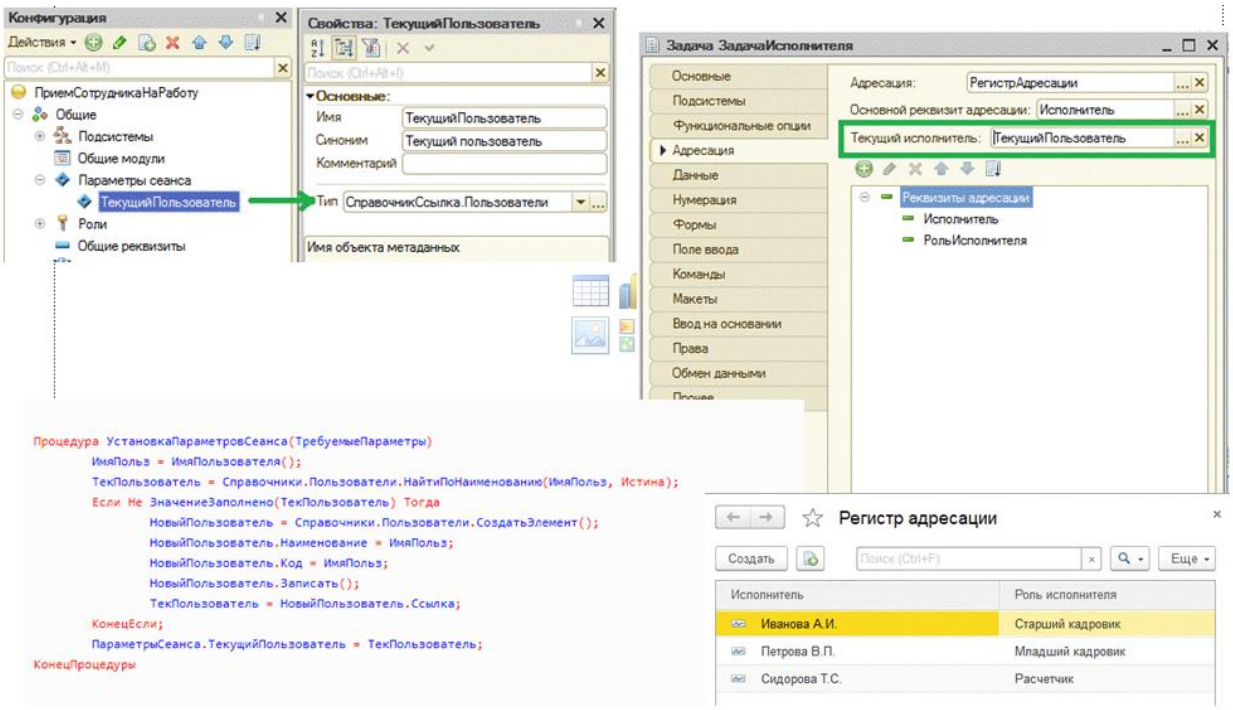


Рисунок 7.7 – Создание параметра сеанса «ТекущийПользователь» для определения текущего пользователя

Пользователю хотелось бы, чтобы по умолчанию в списке задач отображались только его задачи (рисунок 7.8). Для этого создадим форму списка задачи «Задача исполнителя» и в настройках основного реквизита Список поменяем основную таблицу на ***Задача.ЗадачаИсполнителя.ЗадачиПоИсполнителю***. Также можно

оставить только еще не выполненные задачи, добавив в настройку списка соответствующий отбор (рисунок 7.9).

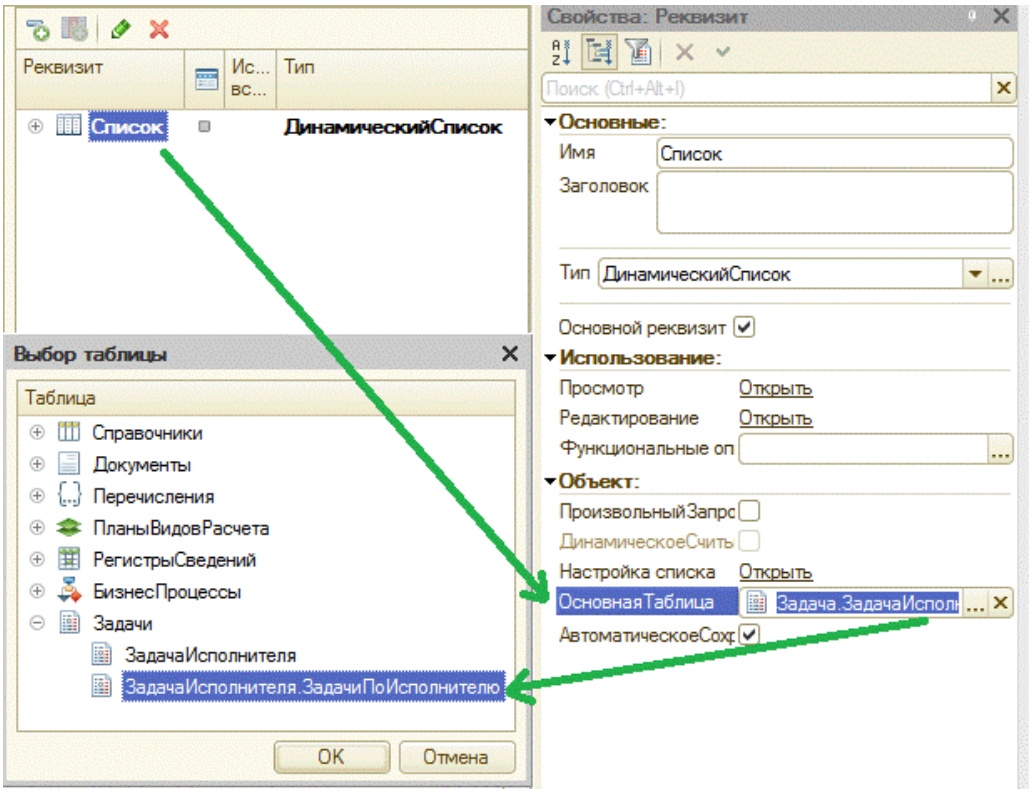


Рисунок 7.8 – Отображение задач по исполнителям

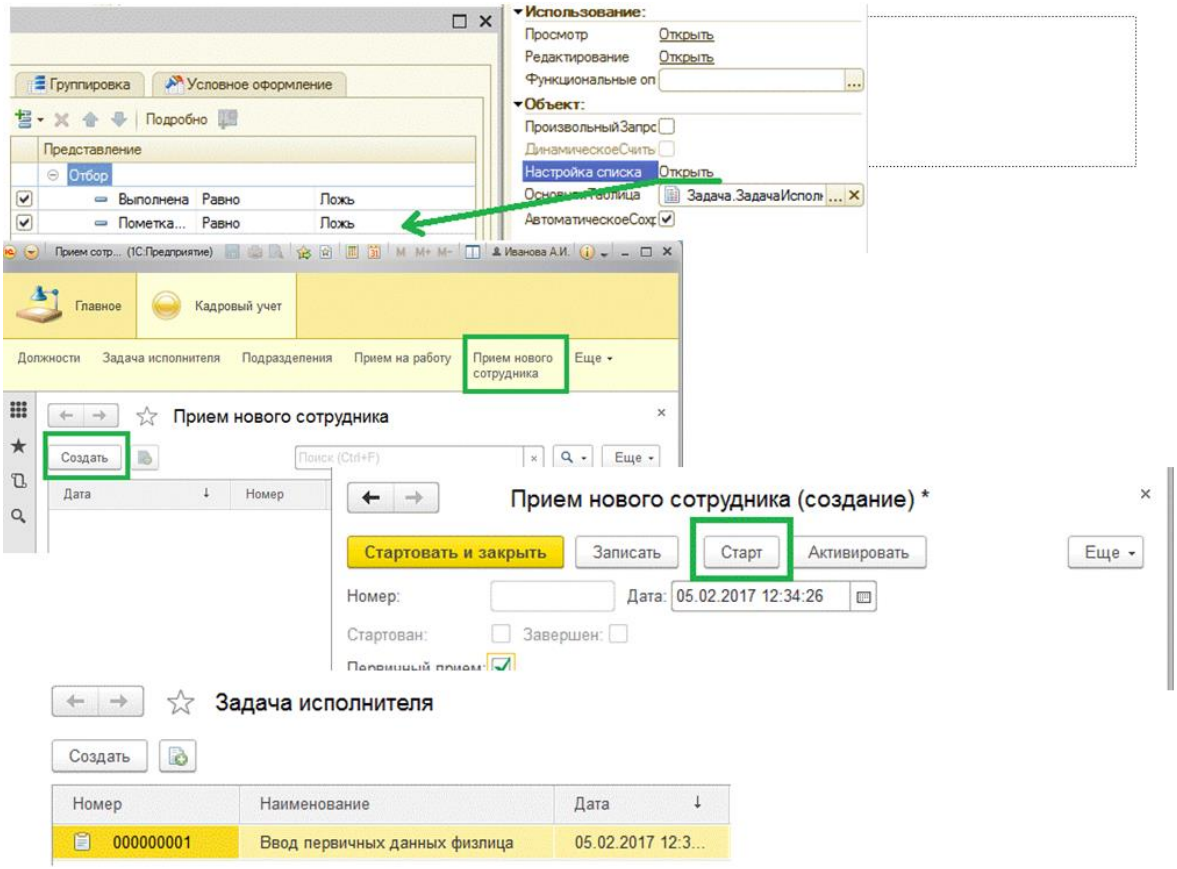


Рисунок 7.9 – Настройка отображения задач и пример работы в пользовательском режиме

Вернемся к карте маршрута. Чтобы обеспечить ветвление в точке условия, требуется обработчик проверки условия – функция в модуле объекта бизнес-процесса, которая возвращает значение Ложь или Истина. Создадим такой обработчик для точки маршрута «ПовторныйПрием» (рисунок 7.10).

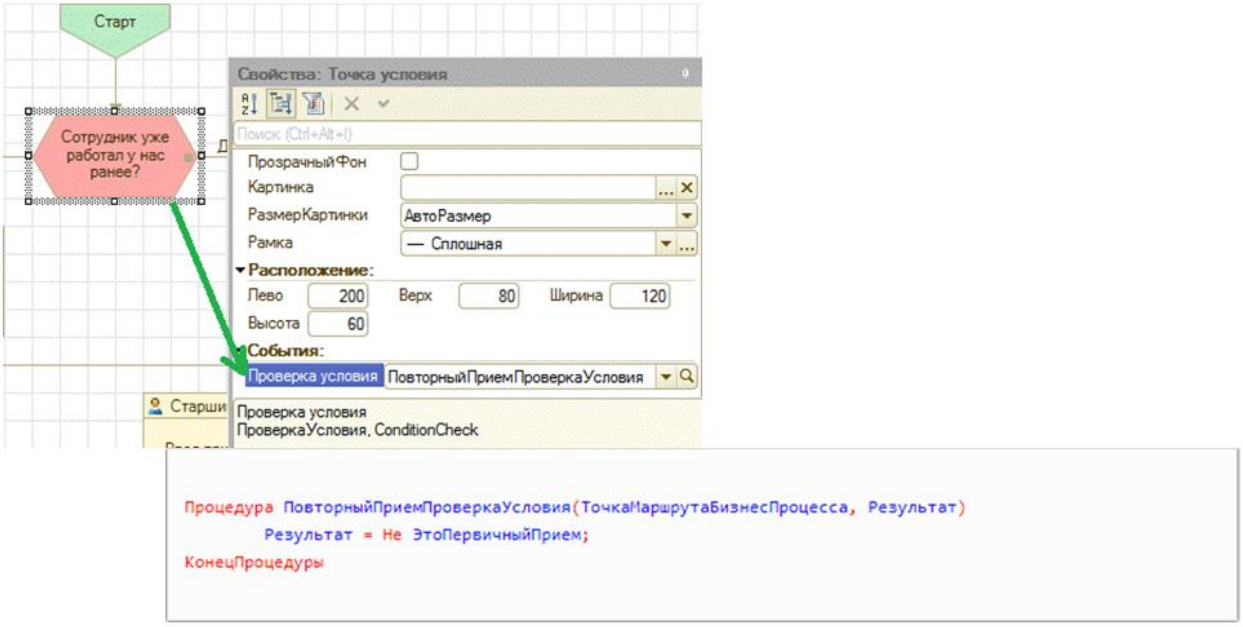


Рисунок 7.10 – Настройка проверки условия для блока карты маршрута

Настроим визуализацию карты маршрута для пользователя (рисунок 7.11). Для этого создадим форму бизнес-процесса и добавим на нее реквизит «КартаБП» типа ГрафическаяСхема, а также выведем элемент управления на форму. Затем в модуле формы бизнес-процесса создадим процедуру «ОбновитьКартуМаршрута». Вызовем эту процедуру в обработчике события «ПриЧтенииНаСервере» формы бизнес-процесса. После этого при открытии формы бизнес-процесса на карте маршрута будет отмечаться текущее положение. Мы используем обработчик «ПриЧтенииНаСервере», т.к. он вызывается самым первым в процессе открытия управляемой формы объекта и выполняется только для уже записанных объектов. Использовать обработчик «ПриСозданииНаСервере» для выполнения кода также допустимо, но данный обработчик вызывается и для тех объектов, которые еще не записаны. В нашем случае это приведет к выполнению лишних действий, ведь вновь создаваемый бизнес-процесс еще не стартован, а значит, он всегда находится в своей начальной точке. Сделаем так, чтобы при нажатии кнопки «Старт» на форме бизнес-процесса карта маршрута обновлялась. Для этого вызовем ту же процедуру «ОбновитьКартуМаршрута» в обработчике «ПослеЗаписиНаСервере».

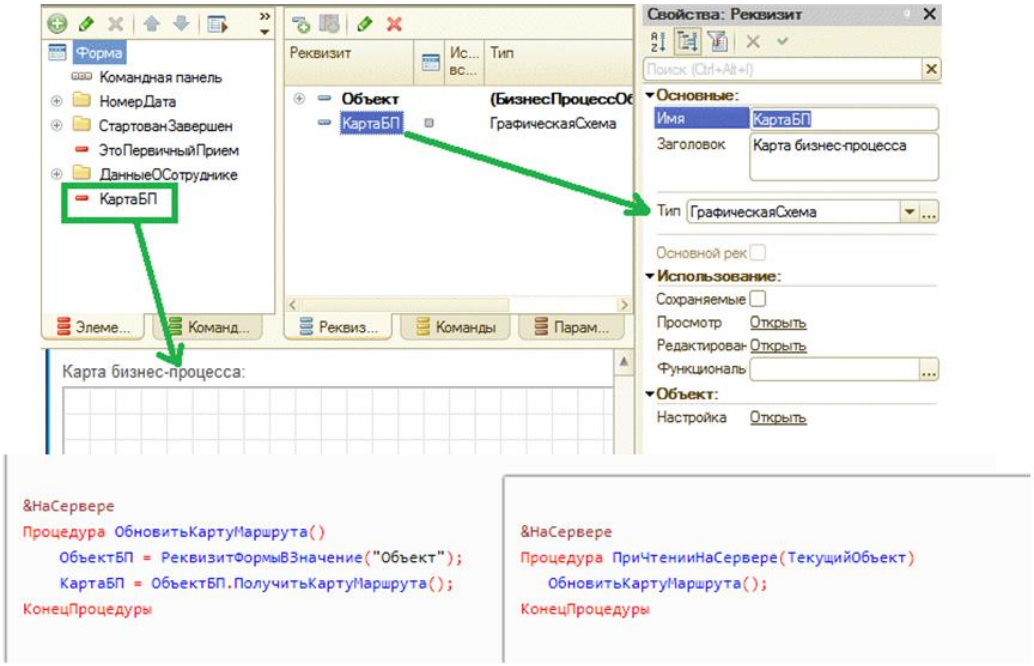


Рисунок 7.11 – Добавление графической схемы карты маршрута на форму

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Практическая работа №12

1. Для выполнения адресации задач требуется создать регистр сведений РолиИсполнителейЗадач, в котором определены измерения: подразделение, должность и пользователь (рисунок 7.12).

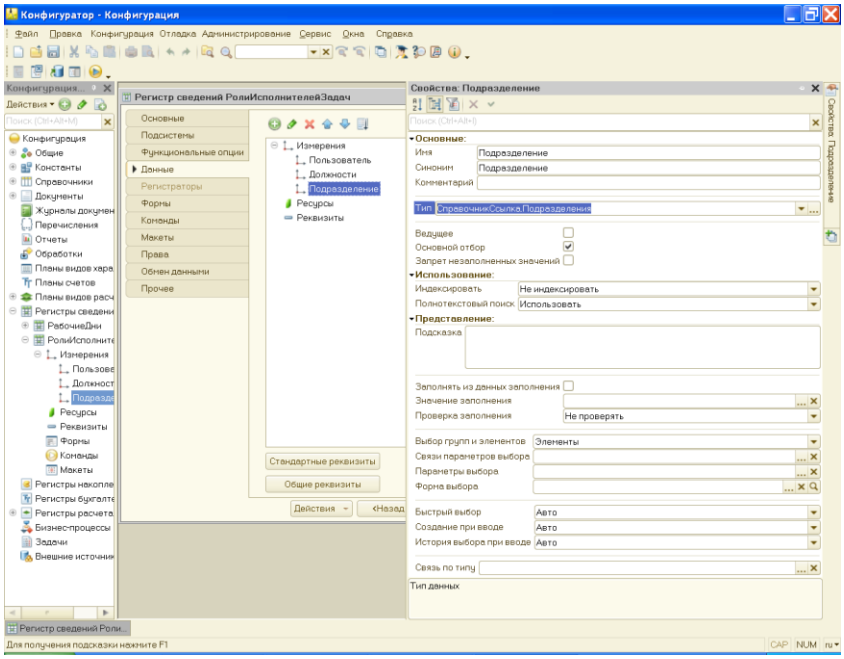


Рисунок 7.12 – Настройка регистра сведений

2. Создать объект конфигурации задачи. На вкладке Адресация, связать объект с регистром сведений РолиИсполнителейЗадач и

параметром сеанса ТекущийПользователь. Установить реквизиты адресации, связывая их с измерением адресации (измерениями регистра) (рисунок 7.13).

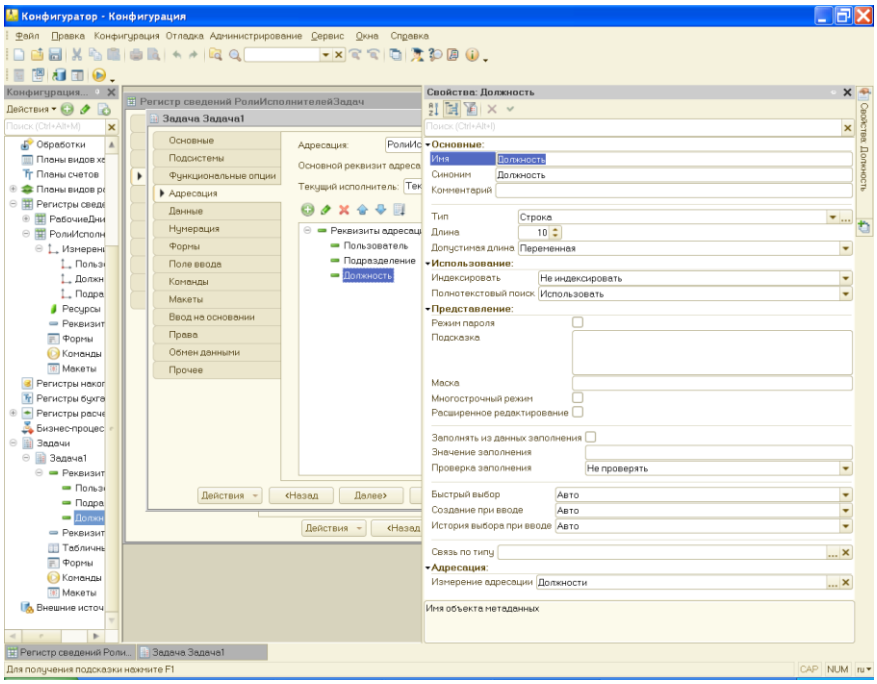


Рисунок 7.13 – Настройка реквизитов адресации

3. Создать объект конфигурации бизнес-процесс Оплата. Связать его с задачей и установить реквизит ОплатаИзКассы.
4. На вкладке Прочее открыть карту маршрута и начертить карту маршрута (рисунок 7.14), связывая действия с предопределенными элементами справочников в адресации (установить подразделения, должности или конкретных пользователей). Для этого тип данных реквизитов в задаче должен соответствовать справочникам.

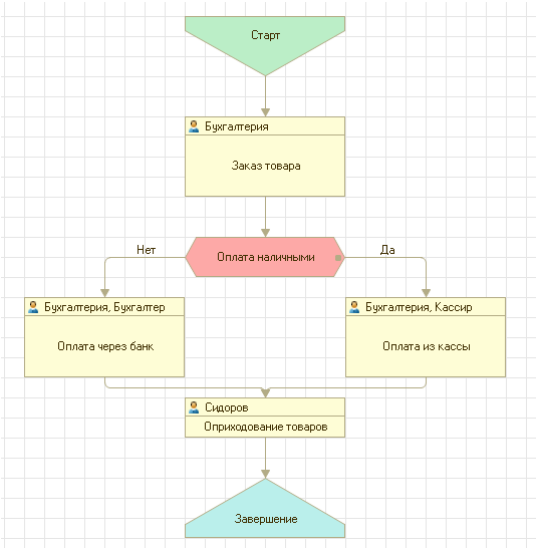


Рисунок 7.14 – Карта маршрута

5. Для блока условия установить обработчик события:

```
Процедура
ОплатаНаличнымиПроверкаУсловия(ТочкаМаршрутаБизнесПроцесса,
Результат)
     // Если параметр "Результат" равен ИСТИНА, то процесс подет
по ветке "ДА", и наоборот.
     Результат = ОплатаИзКассы; // "ОплатаИзКассы" - реквизит
бизнес-процесса
КонецПроцедуры
```

6. Создать форму для объекта бизнес-процесс. Добавить объект типа ГрафическаяСхема. Добавить объект на форму.
7. Добавить команду формы и связанную с ней кнопку на форму. Для команды прописать действие.

```
&НаКлиенте
Процедура ОбновитьКарту(Команда) // Обработчик команды формы
     ОбновитьКартуСервер();
КонецПроцедуры    &НаСервере
Процедура ОбновитьКартуСервер() // Серверная контекстная процедура
получения карты маршрута
     // Конвертируем объект формы в объект бизнес-процесса
     ОбъектБП = РеквизитФормыВЗначение("Объект");

     // Вызываем метод получения карты маршрута текущего бизнес-
процесса
     Карта = ОбъектБП.ПолучитьКартуМаршрута();
КонецПроцедуры
```

8. Для вывода на рабочий стол списка задач, создать в объекте задачи форму списка РабочийСтол**.** Связать список с объектом Задачи.ЗадачиПоИсполнителю
9. Добавить в рабочую область рабочего стола. Для этого открыть рабочую область начальной страницы (рисунок 7.15).

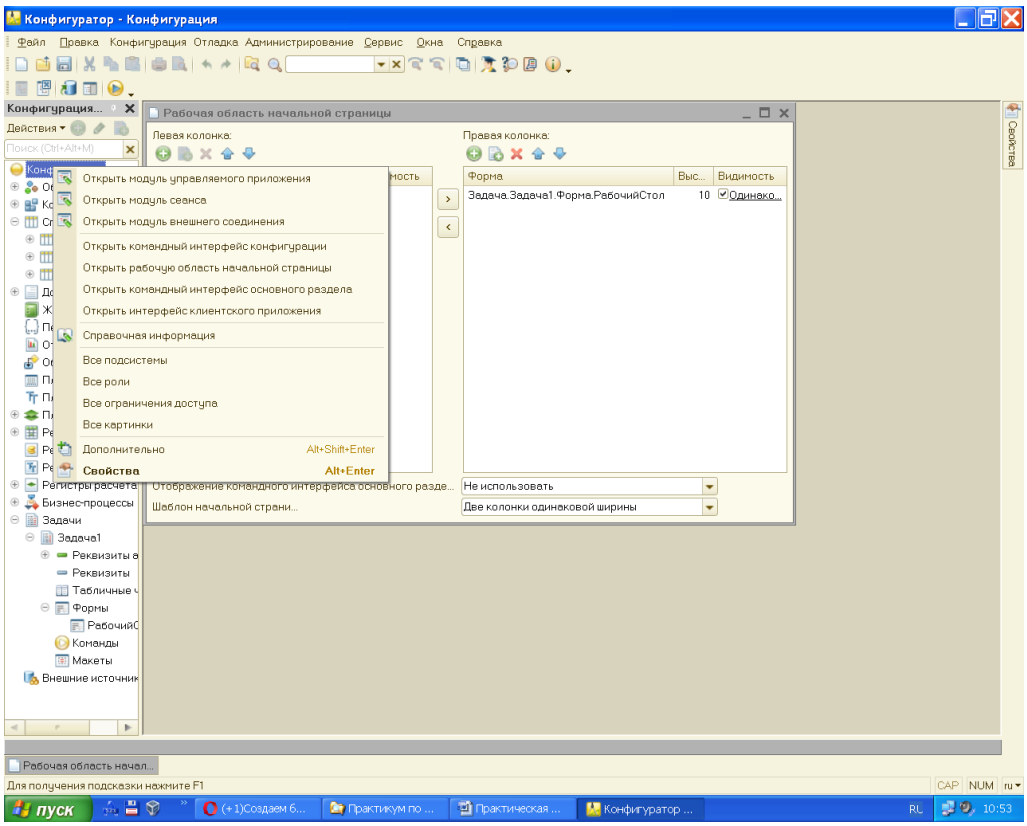


Рисунок 7.15 – Настройка рабочей области начальной страницы

10. В режиме пользователя заполнить регистр сведений «РолиИсполнителейЗадач», в соответствии с бизнес-процессом.

Практическая работа №13

1. Реализовать бизнес-процесс приема на работу на основе примера, приведенного в тексте теоретических сведений.

Практическая работа №14

1. Реализовать (модернизировать) бизнес-процесс продажи товара, с учетом авансовых платежей и выдачи товара со склада.

Практическая работа №15

1. Разработать процедуру начисления премий продавцам, в зависимости от количества оказанных (проданных) услуг.
2. Реализовать необходимые справочники, документы и регистры. Настроить формы справочников и документов для автоматических расчетов.
3. Реализовать необходимые отчеты.
4. Реализовать бизнес-процесс начисления премий и удержания штрафов из премий.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ САМОКОНТРОЛЯ

1. Охарактеризуйте назначение объекта «Бизнес-процесс».

2. Каким образом связываются объекты «Бизнес-процесс» и «Задача»?
3. Охарактеризуйте особенности адресации задач в конфигурации.