ФИРМА "1С"

"Введение в конфигурирование в системе "1С:Предприятие 8". Основные объекты". Версия 8.3

Методические материалы
для слушателя сертифицированного курса

Август 2015

ПРАВО ТИРАЖИРОВАНИЯ И РАСПРОСТРАНЕНИЯ МЕТОДИЧЕСКИХ МАТЕРИАЛОВ ПРИНАДЛЕЖИТ ФИРМЕ "1С"

Фирма "1С", Москва, 123056, а/я 64
Отдел продаж: ул. Селезневская, д.21,
телефон: (495)737-92-57,
факс: (495) 681-44-07,
e-mail: 1c@1c.ru,
URL: http://www.1c.ru

Автор материалов: ООО "1С-Учебный центр №3",
(495) 542-19-94, uc3@1c.ru, www.1c-uc3.ru

08-15

Предложения по совершенствованию методических материалов просьба направлять в группу организации обучения фирмы "1С"
e-mail: cso@1c.ru

Содержание

Соглашения о терминах, обозначениях и дополнительные соглашения

Названия диалоговых кнопок, закладок диалоговых панелей, названия пунктов меню, имена других объектов будут даваться в двойных кавычках, например, "ОК", "Услуги", "Предприятие", "Контрагент" и т.д.

Значения и типы данных будут даваться в угловых скобках: <дата>, <СправочникСсылка.Организации>

Обращение к пункту меню будет даваться в последовательном перечислении родительских пунктов через знак "правый слеш" "/", например, "Конфигурация/ Поддержка/ Обновить конфигурацию"

Практические задания делятся на Практикумы (задания, выполняемые самостоятельно) и примеры (задания, выполняемые вместе с преподавателем). Практические задания оформляются:

Практикум № _____

_____

Ссылки на процедуры и функции в основном тексте будут даваться в двойных кавычках "", без скобок.

Важные дополнения к материалу даются:

Важно! _____

_____

Введение

"1С:Предприятие" является универсальной системой автоматизации учетной деятельности организаций. За счет своей универсальности система "1С:Предприятие" может быть использована для автоматизации самых различных участков деятельности организаций, предприятий.

В рамках программного комплекса "1С:Предприятие" существуют следующие понятия:

- База данных (в собственном формате, либо под управлением ряда СУБД).
- Конфигурация (конкретная "настройка" этой базы данных: состав таблиц информационной базы, описание логики поведения объектов и т.п.)
- Платформа (программный комплекс, который предоставляет возможность работы с базой данных). Т.е. это и среда исполнения "программы", и среда разработки конфигурации, и инструмент администрирования имеющихся информационных баз.

Следует обратить внимание, что довольно часто под "конфигурацией" понимают не только настройку базы данных (структуру таблиц базы данных, прикладную логику поведения объектов), а и полностью развернутую информационную базу.

Основной особенностью системы "1С:Предприятие" является ее конфигурируемость, т.е. возможность довольно просто менять конфигурацию системы (добавлять новые, изменять существующие объекты/таблицы базы данных, определять/переопределять логику их поведения).

Соответственно, функционирование системы делится на два процесса:

- конфигурирование (описание модели предметной области средствами, предоставляемыми системой).
- исполнение (обработка данных предметной области).

Процесс конфигурирования в свою очередь распадается на несколько составляющих:

- "визуальное" создание структуры конфигурации (таких объектов конфигурации как справочники, документы и т.п.)
- определение прав доступа к функциональности системы
- настройка диалоговых форм объектов
- определение специфики поведения объектов, форм (прописывание кода на языке системы в определенных местах конфигурации)

В процессе исполнения система уже оперирует конкретными понятиями, описанными на этапе конфигурирования (справочниками товаров и организаций, накладными и т.д.).

1. Цели и задачи курса

Данный курс рассчитан на слушателей, имеющих опыт работы с объектно-ориентированными языками программирования или работавших ранее с предыдущими версиями "1С:Предприятие".

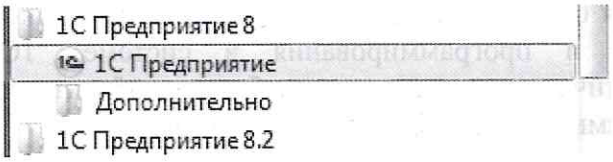
Курс является базовым. В процессе обучения Вы ознакомитесь с основами конфигурирования и программирования в системе "1С:Предприятие 8", приобретете практические навыки по работе с объектами конфигурации и написании программных модулей на языке системы.

Также следует отметить, что некоторые темы в рамках данного курса изучаются поверхностно или не изучаются совсем. Для более подробного изучения подобного материала существуют другие (последующие) курсы, обладающие уже нужной специализацией.

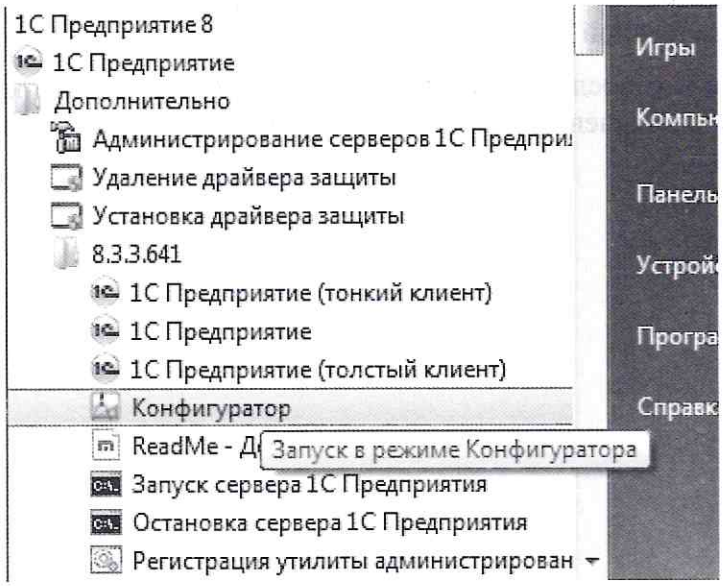
Считается, что после ознакомления с материалом данного курса вы будете иметь общее представление о возможностях программного комплекса и сферах его применения.

2. Общие принципы работы в программном комплексе

Работа с программным комплексом начинается с его запуска. Это можно сделать с помощью универсальной программы запуска:



Либо, если вы явно хотите запустить какую-либо версию/релиз, то следующим образом:



Следует обратить внимание на существование трех видов клиентов:

- Толстый
- Тонкий
- веб-клиент

Также платформа может работать в двух вариантах:

- Файловом (есть клиентское приложение, работающее с базой данных собственного формата).
- Клиент-серверном (клиентское приложение, кластер серверов "1С:Предприятие" и база данных под управлением СУБД).

В клиент-серверном варианте в качестве СУБД может использоваться:

- MS SQL Server
- PostgreSQL
- IBM DB2
- Oracle Database

В этом режиме (клиент-серверном) тонкий и веб-клиент не требует наличия постоянного соединения с источником данных (веб-сервером, кластером серверов "1С:Предприятие"). Следует отметить, что подобное возможно и в файловом варианте (при подключении через веб-сервер), но такой вариант можно назвать скорее "инженерным", чем рекомендуемым к использованию на практике.

Варианты подключения можно проиллюстрировать следующими схемами:

Клиент-серверный вариант

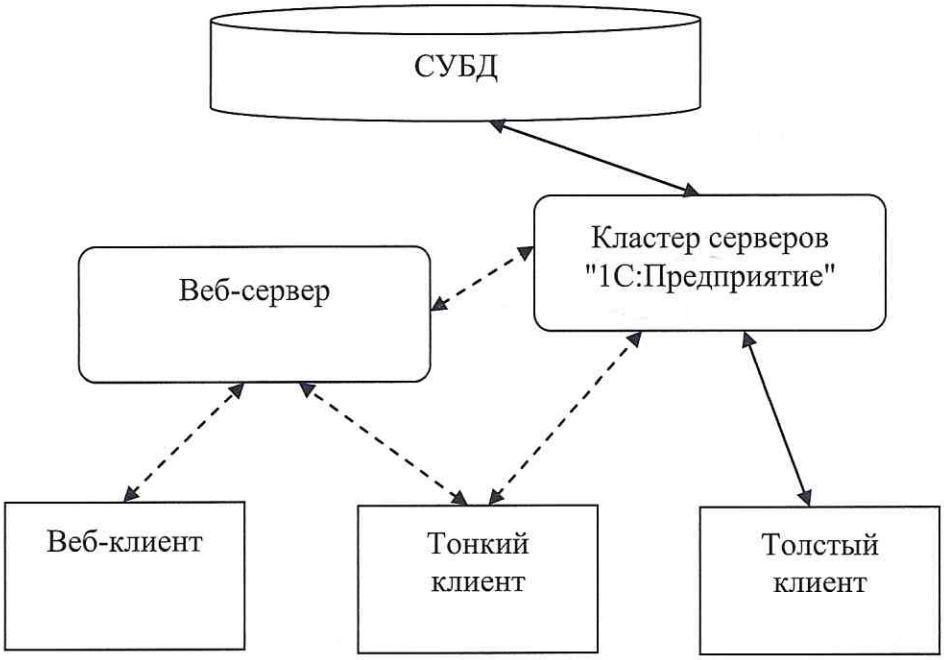


Схема 2.1

Файловый вариант

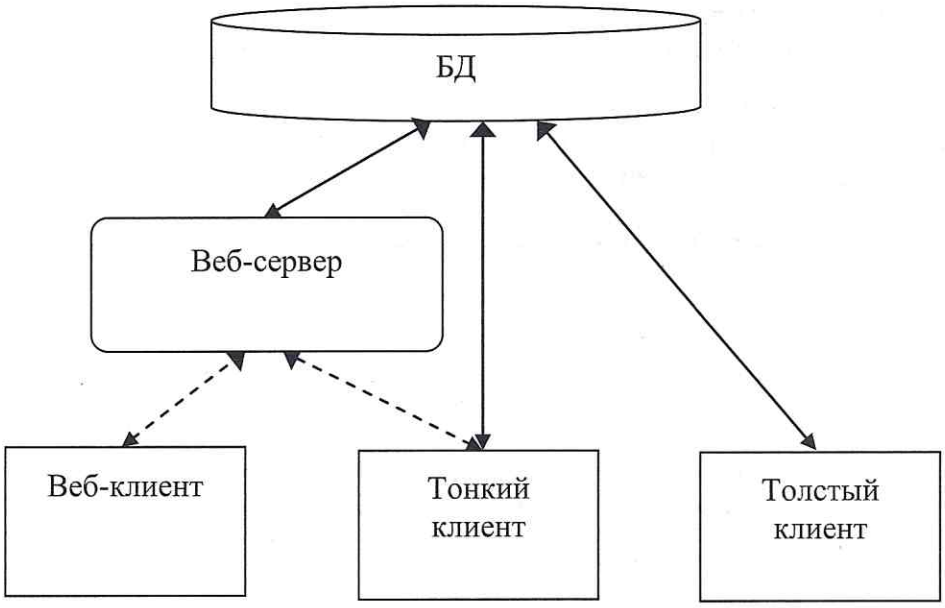
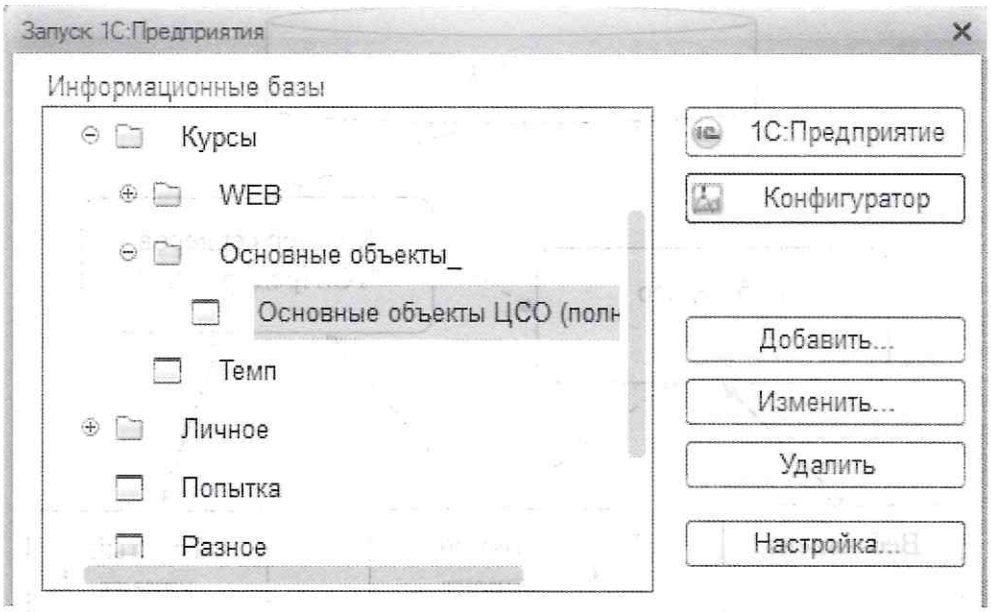


Схема 2.2

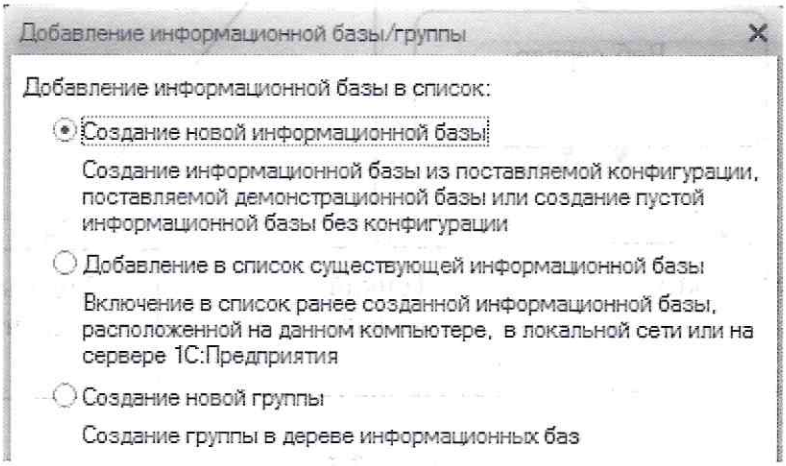
Вернемся к старту программы. После старта, в так называемом "окне запуска", можно выбрать один из режимов запуска: "1С:Предприятие" или "Конфигуратор". Делается это с помощью одноименных кнопок окна запуска



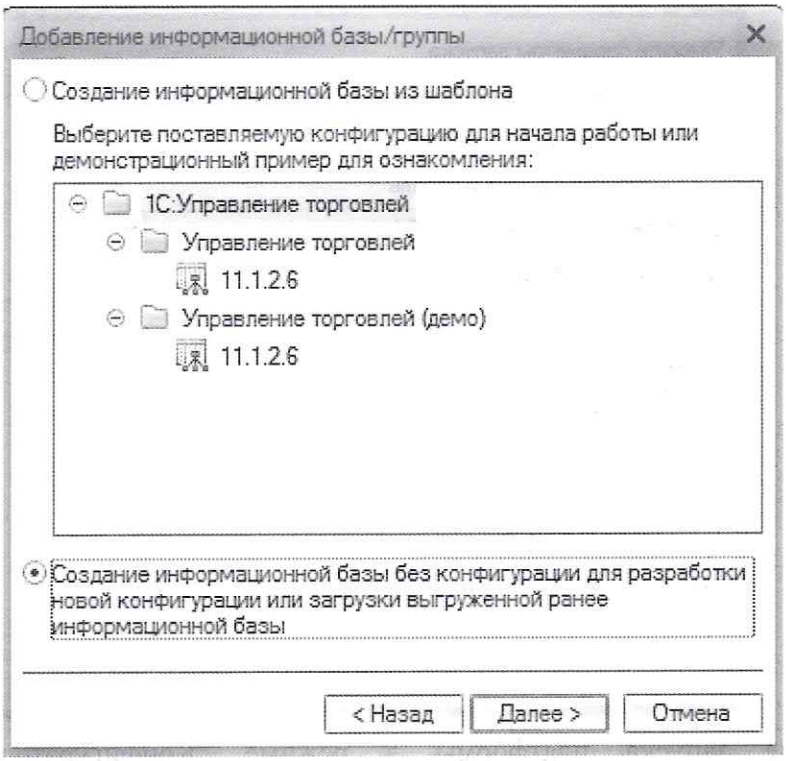
Режим "1С:Предприятие" это режим исполнения (можно сказать, "пользовательский режим"). Режим "Конфигуратор" это режим разработчика, администратора системы.

С точки зрения разработчика процесс конфигурирования заключается в том, что в режиме "Конфигуратор" в окне конфигурации создаются объекты конфигурации, в определенных местах (модулях) прописывается текст на встроенном языке (описывающий специфику поведения созданных объектов).

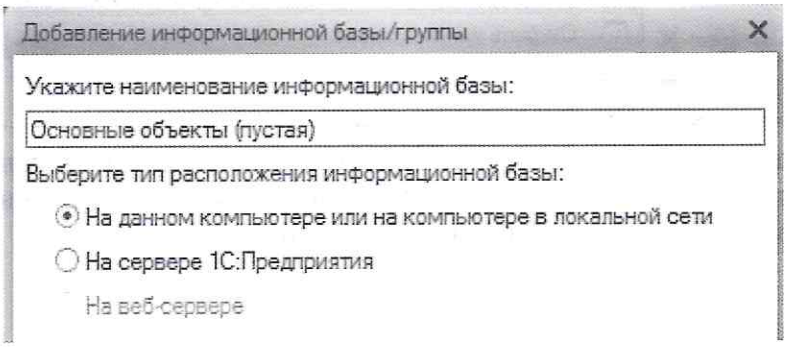
Начнем с создания новой информационной базы. После нажатия на кнопку "Добавить" в окне запуска, в открывшемся диалоге нужно отметить пункт "Создание новой информационной базы".



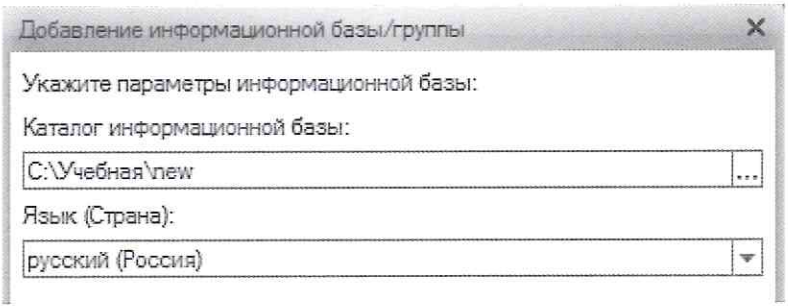
На следующем этапе (переход к следующему этапу осуществляется по нажатию кнопки "Далее") нужно отметить пункт "Создание информационной базы без конфигурации…"



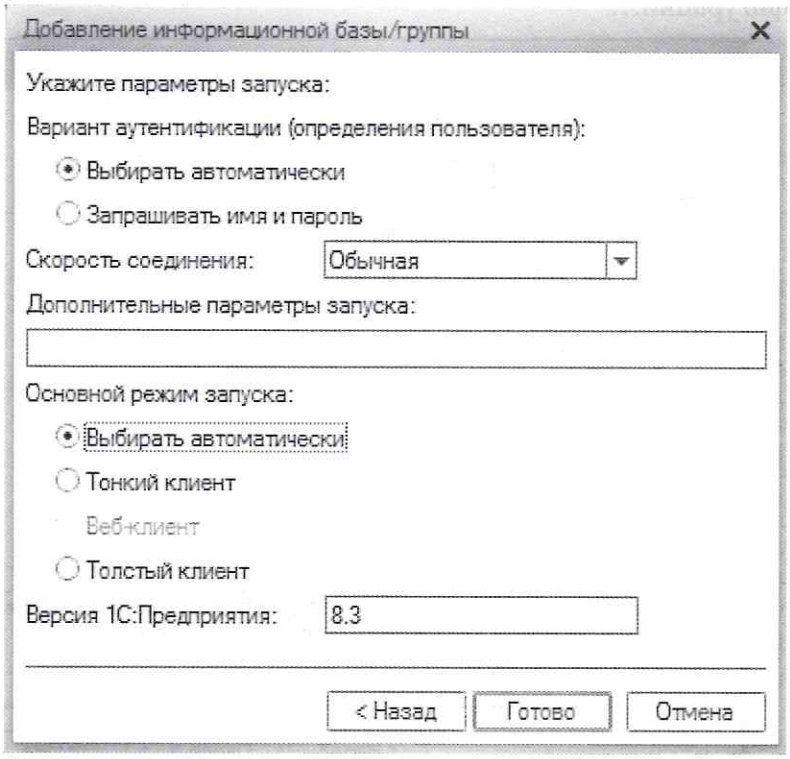
Далее нужно указать наименование информационной базы (как она будет называться в списке, в контексте этого пользователя) и выбрать вариант функционирования "На данном компьютере или на компьютере в локальной сети" (что соответствует файловому варианту функционирования платформы).



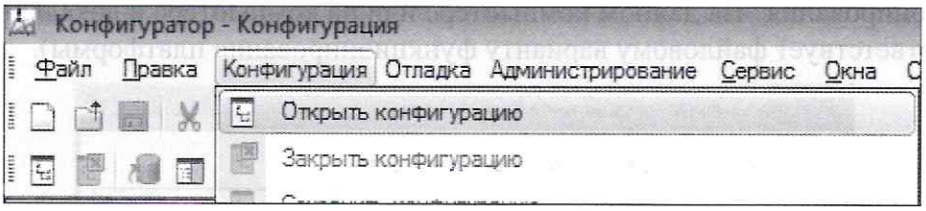
Далее указываем каталог, в котором будет размещаться база:



На следующем этапе можно настроить ряд других параметров запуска информационной базы.



После запуска платформы в режиме "Конфигуратор" работа по конфигурированию начинается с выполнения команды главного меню программы "Открыть конфигурацию"



Обратите внимание на тот факт, что есть команда "Открыть конфигурацию" и "Окно конфигурации". При выполнении команды "Открыть конфигурацию" открывается и конфигурация и окно конфигурации.

3. Объекты системы

"1С:Предприятие" относится к классу предметно-ориентированных систем (т.е. в данном программном комплексе "поддерживается" понятие **объект**).

Под **объектом** упрощенно можно понимать некий "черный ящик", обладающий определенной функциональностью. Он характеризуется каким-либо набором свойств, обладает какими-либо методами, реагирует на определенные события в системе. Методы объекта могут менять его "внутреннее состояние" (значения свойств), могут "заставлять" объект что-либо "делать".

Следует обратить внимание на то, что со словом "объект" в программном комплексе есть несколько понятий: объекты конфигурации (у них нет методов, внутренних состояний), объекты базы данных, системы.

При определении структуры конфигурации разработчик работает с объектами конфигурации, настраивает их свойства. Такие объекты располагаются внутри дерева объектов конфигурации.

Для некоторых из объектов конфигурации можно сказать, что они являются "конструкторами" таблиц информационной базы (в зависимости от состава и значений свойств такого объекта меняется состав полей таблицы базы данных).

Описывая алгоритмы обработки информации, разработчик работает с объектами системы, объектами базы данных (записями таблиц, идентифицирующимися ссылками).

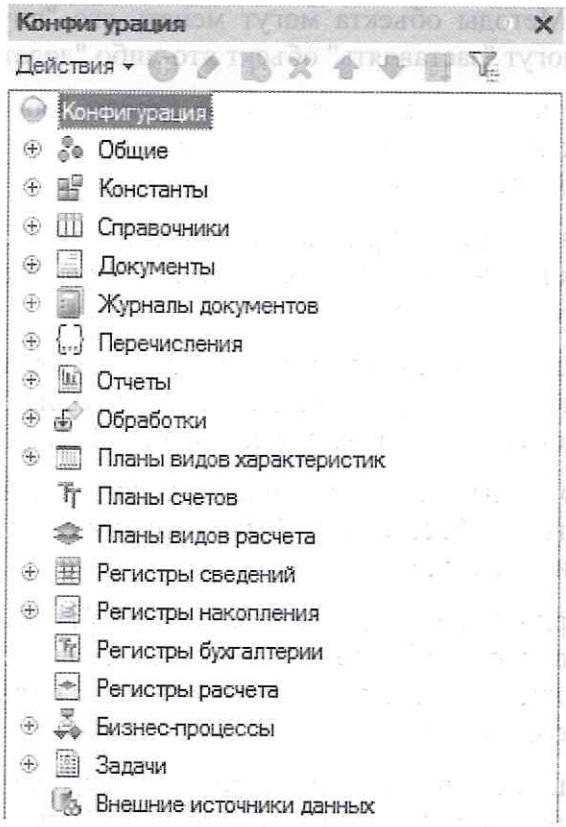
Несмотря на принципиальное отличие между объектами конфигурации и объектами базы данных, объектами системы, взаимосвязь присутствует. Определение какого-либо объекта конфигурации может привести к "появлению" объектов системы, базы данных.

Например, добавление объекта конфигурации "справочник Номенклатура" приводит к тому, что при описании алгоритмов обработки информации можно использовать такие объекты как:

- <СправочникСсылка.Номенклатура>
- <СправочникОбъект.Номенклатура>
- и т.д.

3.1. Классификация объектов конфигурации

Все объекты конфигурации, которые существуют в системе "1С:Предприятие", образуют несколько основных видов. Каждый вид объектов конфигурации представляет собой как раз те "строительные элементы", из которых будет создаваться конфигурация. Разбивку объектов по видам можно увидеть в дереве конфигурации (они находятся на первом его уровне).



Все объекты конфигурации можно подразделить на три основные группы:

- **Общие объекты.** Группа вспомогательных объектов конфигурации, с помощью которых осуществляется создание конфигурации, механизмов взаимодействия пользователей с учетными данными.
- **Прикладные объекты.** Их перечень можно увидеть на первом уровне дерева метаданных (исключая группу "Общие").
- **Подчиненные объекты.** К таким объектам относятся "Реквизиты", "Табличные части" и т.д.

3.1.1. Прикладные объекты

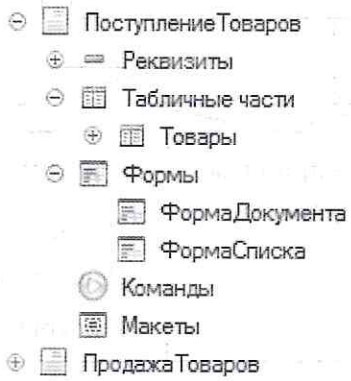
К объектам данной группы относятся объекты следующих видов:

- **Константы**. Предназначены для хранения постоянных, условно-постоянных величин.
- **Справочники.** Списки однородных элементов данных. Используются для хранения нормативно-справочной информации.
- **Документы**. Служат для ввода информации о совершаемых операциях в системе.
- **Журналы документов**. Служат для отображения списков документов различного вида.

- **Перечисления**. Списки значений, задаваемых на этапе конфигурирования.
- **Отчеты**. Средство получения выходной информации.
- **Обработки**. Используются для выполнения различных действий над информационной базой.
- **Планы счетов**. Совокупность синтетических счетов.
- **Планы видов характеристик**. Предназначены для описания множеств однотипных объектов аналитического учета.
- **Планы видов расчета**. Предназначены для описания множеств однотипных объектов механизмов расчета.
- **Регистры сведений**. Служат для хранения информации, состав которой развернут по определенной комбинации значений и, при необходимости, развернут во времени.
- **Регистры накопления**. Служат для накопления информации в разрезе измерений с возможностью получения остатков или оборотов числовых величин (или остатков и оборотов).
- **Регистры расчетов**. Служат для накопления информации о периодических расчетах.
- **Регистры бухгалтерии**. Используются для отражения в бухгалтерском учете информации о хозяйственных операциях.
- **Бизнес-процессы**. Используются для реализации "процессного" принципа работы. Данный принцип позволяет автоматизировать процесс прохождения и контроля цепочек событий, операций.
- **Задачи.** Совместно с бизнес-процессами реализуют процессный принцип. Они позволяют вести учет заданий по исполнителям и служат отражением продвижения бизнес-процессов по точкам маршрута
- **Внешние источники.** Позволяют организовать более удобную работу с внешними источниками данных (в основе лежит механизм ODBC).

3.1.2. Подчиненные объекты

В зависимости от вида объекта конфигурации (прикладного или общего) он может иметь различные подчиненные группы объектов.



Приведем перечень подчиненных объектов:

- **Реквизиты** – дополнительная информация об объекте, доступная только в пределах этого объекта. Можно сказать, что с помощью реквизитов можно определить дополнительные свойства объекта.

- **Табличные части** – наборы дополнительной информации об объекте, представленные в виде таблиц.
- **Реквизиты табличных частей** – состав табличной части объекта, доступны только в пределах табличной части объекта.
- **Формы** – используются для ввода, просмотра и редактирования информации.
- **Команды** – используются для реализации каких-либо действий, принадлежащих объекту.
- **Макеты** – предназначены для формирования печатных форм объекта.
- **Графы** – графы журнала документов.
- **Измерения** – для регистров это объекты конфигурации, в разрезе которых учитываются данные в регистре.
- **Ресурсы** – данные, учитываемые в регистре.

3.1.3. Концепция системы

В "1С:Предприятие" используется принцип учета "от документа". Т.е. деятельность организации разбивается на элементарные операции. Под каждую операцию создается объект "Документ".

Схема 3.1

Документами в систему вносится первичная информация о совершенной хозяйственной операции. При заполнении документов используется дополнительная справочная информация. Информация из документов попадает в

учетные объекты – регистры. Данные в регистрах могут быть откорректированы различными регламентами (регламентными документами, обработками).

3.2. Типы данных

Одним из основных свойств некоторых объектов конфигурации является **тип данных**. Это свойство определяет, какого рода информацию может содержать объект конфигурации.

Различают три основных группы типов данных:

- **Примитивные типы** (в их состав входят базовые типы данных).
- **Типы данных, зависимые от метаданных**, появившиеся после определения в конфигурации объектов конфигурации.
- **"Другие" типы**, не относящиеся к примитивным и "добавляемым", но поддержка которых во встроенном языке есть изначально.

К примитивным типам данных относятся:

- <Число> (десятичное число)
- <Строка> (строка фиксированной; переменной или неограниченной длины)
- <Дата> (дата, время, дата+время)
- <Булево> (истина или ложь)
- <Тип>
- <Неопределенно>
- <Null>

3.3. Универсальные коллекции значений

"Другие" типы, не относящиеся к примитивным и "добавляемым", но поддержка которых во встроенном языке есть изначально, иногда являются коллекциями (их можно "обойти" как по индексу, так и с помощью специального вида цикла *"Для Каждого Из"*). Часть из этих типов входит в так называемые "Универсальные коллекции значений". Универсальные коллекции значений предназначены для хранения временных наборов данных в течение сеанса работы пользователя. Они не являются объектами информационной базы и служат для вспомогательного сбора, группировки, анализа и обработки информации. Рассмотрим некоторые из них:

Массив

Объекты этого типа представляют собой совокупность значений любого типа, в том числе и типа "массив", что, в частности, позволяет организовывать многомерные массивы.

Объект создается из программного кода с использованием конструктора "Новый".

```
Массив = Новый Массив(Кол-во элем 1,....N);
```

Пример кода:

```
Массив = Новый Массив;

Массив.Добавить("Первый");

Массив.Добавить(2);

// так далее
```

Структура

Структура представляет собой динамический набор данных – коллекцию значений, каждый элемент которой состоит из пары "Ключ" и "Значение". Ключи структуры уникальны, и поэтому ими можно идентифицировать значения. Ключ структуры должен быть строковым и отвечать требованиям к именам переменных. К значениям структуры можно обращаться как к свойствам объекта, при этом ключ используется как имя свойства.

```
СтруктураОтбора = Новый Структура("Ключи",Значения);
```

Пример кода:

```
Отбор = Новый Структура("Валюта,Контрагент",Валюта,Контрагент);
```

Допустим другой вариант создания структуры:

```
СтруктураОтбора = Новый Структура;

СтруктураОтбора.Вставить("Валюта",Валюта);

СтруктураОтбора.Вставить("Контрагент",Контрагент);
```

Соответствие

Соответствие представляет собой динамический набор данных – коллекцию значений, каждый элемент которой состоит из пары "Ключ" и "Значение". Ключи соответствия уникальны, и поэтому ими можно идентифицировать значения. В отличие от ключа структуры, ключи соответствия могут быть произвольных типов. Рекомендуется, чтобы в качестве ключа выступало значение неизменяемого типа или другого типа, значение которого может только присваиваться, но не может менять свое содержимое.

```
Соотв = Новый Соответствие();
```

Список значений

Список значений это объект, позволяющий строить динамические наборы значений и манипулировать ими. Может быть наполнен значениями любых типов. Условно список значений можно представить как таблицу из четырех колонок: пометка, значение, представление, картинка. Каждое из значений характеризуется позицией в списке (индексом).

```
СПЗ = Новый СписокЗначений
```

Таблица значений

Таблица значения – объект, позволяющий строить динамические наборы значений и манипулировать ими. Он может быть наполнен значениями различных типов. Может иметь любое количество колонок и быть связанным с элементом "табличное поле".

```
ТЗ = Новый ТаблицаЗначений
```

Пример кода:

```
ТаблицаЗначений = Новый ТаблицаЗначений;

ТаблицаЗначений.Колонки.Добавить("Количество",,"Количество товара");

СтрокаТаблицыЗначений = ТаблицаЗначений.Добавить();

СтрокаТаблицыЗначений.Количество = 11;
```

Дерево значений

Объект, похожий на таблицу значений. Но, в отличие от нее, строки дерева значений могут образовывать иерархические структуры: каждая строка дерева может иметь набор подчиненных строк и т.д.

```
ДЗ = Новый ДеревоЗначений();
```

3.4. Встроенный язык системы

Необходимость наличия встроенного языка определена концепцией настраиваемости системы. Язык является предметно-ориентированным. Он поддерживает специализированные типы данных предметной области, определяемые конфигурацией системы. Работа с этими типами данных в языке организована с использованием объектной техники.

Язык поддерживает конструкции, позволяющие определять переменные, процедуры, функции. Операторы отделяются друг от друга символом ";".

Встроенный язык не чувствителен к регистру, допускается двуязычное описание конструкций (Если, If). Рекомендуется все же писать на языке типовых конфигураций.

```
Перем ИмяПеременной;


Процедура ИмяПроцедуры(ИмяПараметра1, ИмяПараметра2,....)

// текст комментария

// тело процедуры

КонецПроцедуры


Функция ИмяФункции(ИмяПараметра1, ИмяПараметра2,....)
```

```
// тело функции

Возврат(ВозвращаемоеЗначение);

КонецФункции
```

Имя переменной, процедуры, функции может состоять из букв, цифр и символов подчеркивания. Начинаться имя должно либо с буквы, либо с символа подчеркивания.

Порядок описания процедур, функций между собой значения не имеет. Как и в любом другом языке существуют конструкции, реализующие ветвление, циклы:

```
Если Условие Тогда

// код

ИначеЕсли Условие Тогда

// код

Иначе

// код

КонецЕсли;


Для ПеременнаяСчетчик = НачальноЗначение По Конечное Цикл

// тело цикла

КонецЦикла;


Для Каждого ПеременнаяЦикла Из ИмяКоллекции Цикл

// тело цикла

КонецЦикла;


Пока УсловиеЦикла Цикл

// тело цикла

КонецЦикла;
```

Очень часто (в основном) во встроенном языке придется иметь дело с некими объектными сущностями (с объектами, имеющими набор свойств, методов). Для обращения к свойству объекта можно использовать два подхода:

```
Наим = Спр.Наименование;

Наим = Спр["Наименование"];
```

Вызов методов объектов производится "через точку".

```
Спр.Печать();
```

Допускаются следующие конструкции:

```
Док.Контрагент.ПолучитьОбъект().ПечатьКарточкиКлиента();
```

Платформа "1С:Предприятие 8" сочетает в себе визуальные и языковые средства конфигурирования. Использование встроенного языка в системе имеет событийно-зависимую ориентацию, то есть языковые модули используются в конкретных местах для отработки отдельных алгоритмов, настраиваемых в процессе конфигурации. Программный код всегда помещается в модули.

Место размещения конкретного программного модуля предоставляется конфигуратором в тех точках конфигурации, которые требуют описания специфических алгоритмов функционирования. Эти алгоритмы следует оформлять в виде процедур или функций, которые могут быть вызваны самой системой в заранее предусмотренных ситуациях.

4. Основные объекты

4.1. Постановка задачи

Изучать основы конфигурирования и программирования в программном комплексе "1С:Предприятие 8" будем на примере написания простой конфигурации, позволяющей автоматизировать учет в некоей "мифической фирме". Эта конфигурация не будет претендовать на "законченное решение", но позволит разобраться с основными принципами работы в системе.

Деятельность компании заключается в том, что она закупает у своих поставщиков товары (по ценам закупки), которые затем продает своим покупателям (по ценам продажи).

Необходимо организовать хранение информации:

- о номенклатуре товаров
- о контрагентах нашей фирмы
- о сотрудниках компании

Мы должны организовать документооборот таким образом, чтобы пользователю было удобно работать с программой и не приходилось вводить информацию дважды.

Нам в любой момент времени необходимо иметь возможность получить следующую информацию:

- о покупках (у кого и сколько товаров мы купили)
- о продажах (в разрезе покупателей и проданных им товаров)
- о сотрудниках организации

Для простоты будем считать, что торговля ведется от имени одного юридического лица и всегда в одной валюте (цены и стоимость товаров учитываются в одной валюте). Но в рамках компании существует несколько складов.

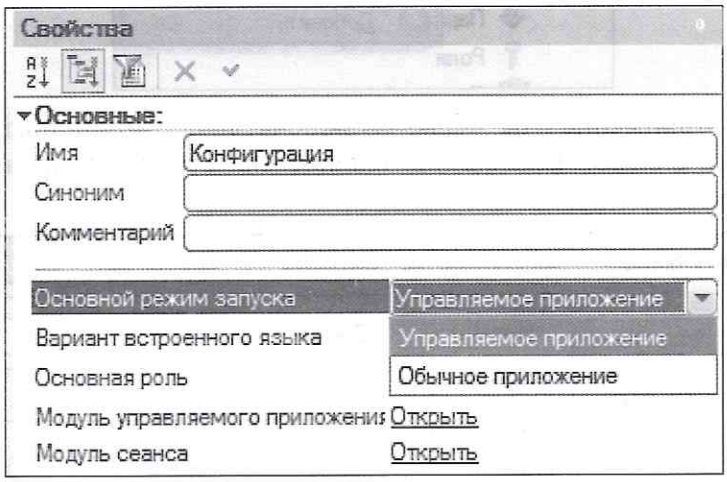
Итак, начнем.

4.2. Определение режима запуска

Толстый клиент "1С:Предприятие" может работать в двух режимах:

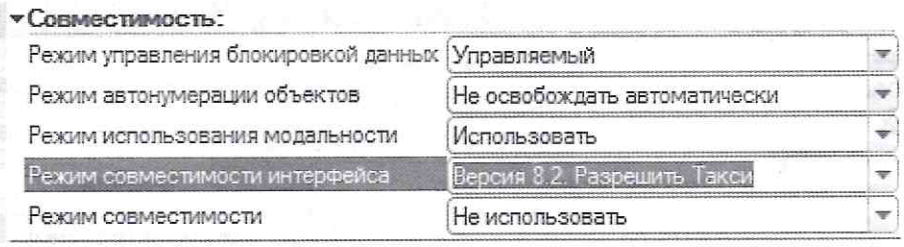
- **Обычное приложение** (режим, в котором работали предыдущие версии "1С:Предприятие 8")
- **Управляемое приложение** (новый режим)

В каком режиме будет работать толстый клиент, может быть определено в свойствах конфигурации, как показано на рисунке.



Следует отметить, что режим запуска также можно назначать каждому пользователю, также режим можно указывать в параметрах подключения информационной базы.

Также в соответствии с рисунком установим значения свойств "Режим совместимости интерфейса" и "Режим совместимости".



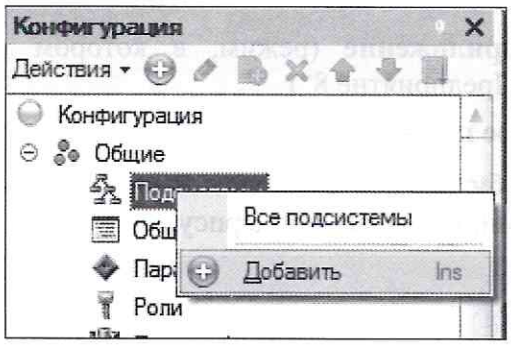
4.3. Командный интерфейс

Командный интерфейс – это основное средство доступа пользователя к функциональности приложения, средство, которое позволяет перемещаться между формами и выполнять те или иные действия. Одной из важных особенностей командного интерфейса является то, что он описывается декларативно. Разработчик не прорисовывает его в деталях, а просто описывает правила его формирования.

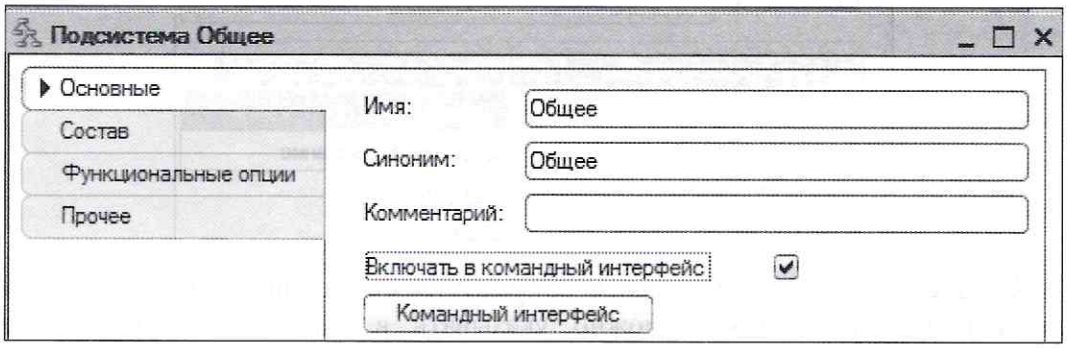
4.3.1. Подсистемы

Структура подсистем определяет структуру функциональности прикладного решения. Можно сказать, что структура подсистем определяет, каким образом пользователь будет осуществлять "навигацию" по функциональности предлагаемого решения.

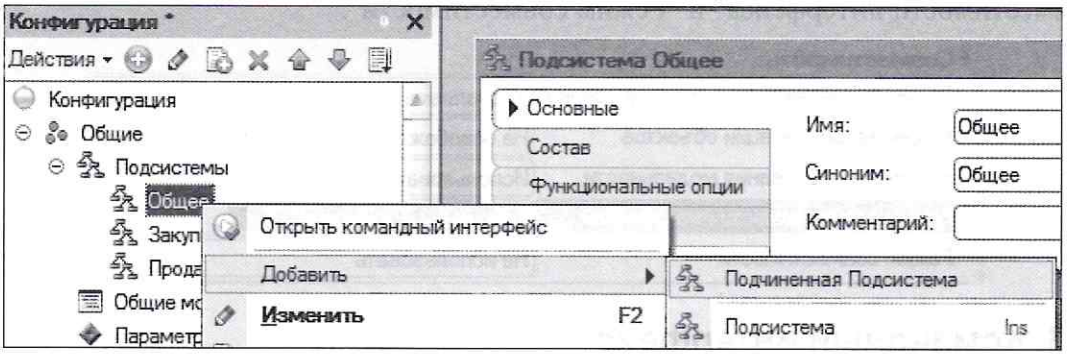
Для создания подсистем нужно зайти внутрь ветви "Общие" и выполнить команду "Добавить" контекстного меню ветви "Подсистемы" (можно воспользоваться кнопкой командной панели окна дерева объектов конфигурации).



Обратите внимание на флаг "включать в командный интерфейс". Подсистемы со снятым флагом не влияют на структуру функциональности программного комплекса.

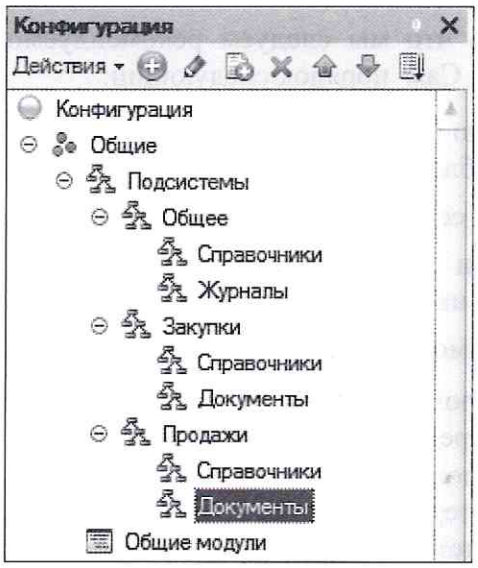


К подсистемам первого уровня можно добавлять подчиненные подсистемы, и так далее.

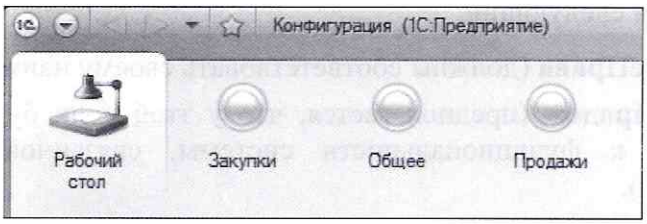


Количество уровней подсистем не ограничено (можно сказать, что ограничено только здравым смыслом).

Определите подсистемы, таким же образом, как на рисунке снизу:



Подсистемы первого уровня определяют структуру так называемой "панели разделов". Это можно увидеть, запустив "1С:Предприятие".

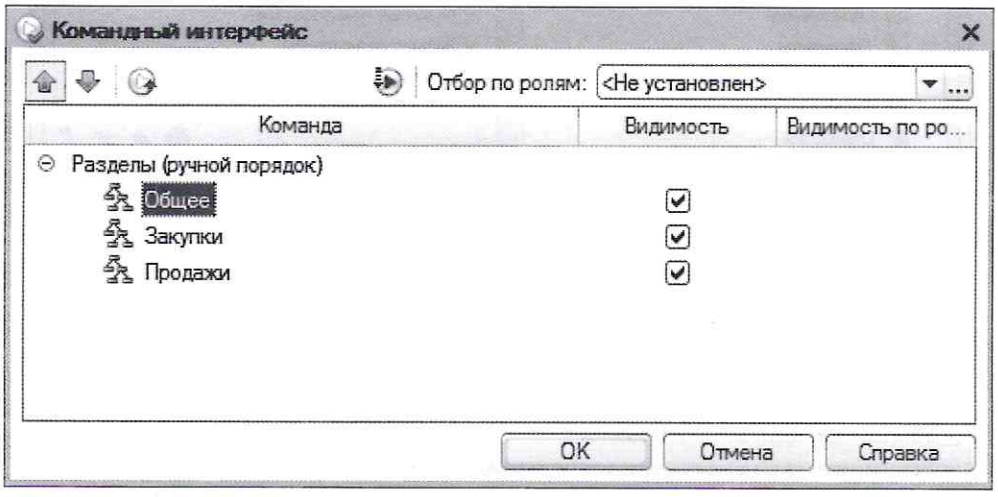


Сразу же бросается в глаза, что порядок разделов в панели отличается от порядка подсистем, определенных при конфигурировании (они отсортированы по алфавиту).

Как уже говорилось, интерфейс описывается декларативно. "Внешний вид" программы формируется "на лету". Можно сказать, что интерфейс собирается как мозаика, и каждый элемент мозаики описывается в системе в определенных точках конфигурации.

Для доступа к нужному в данный момент "элементу мозаики" нужно выполнить команду контекстного меню корня дерева объектов конфигурации "Открыть командный интерфейс конфигурации". Либо в палитре свойств найти одноименные свойства и воспользоваться гиперссылкой.

В открывшемся окне можно "поправить" последовательность разделов.



4.3.2. Роли

Следует отметить, что мы следуем рекомендуемому порядку разработки командного интерфейса. Сам порядок следующий:

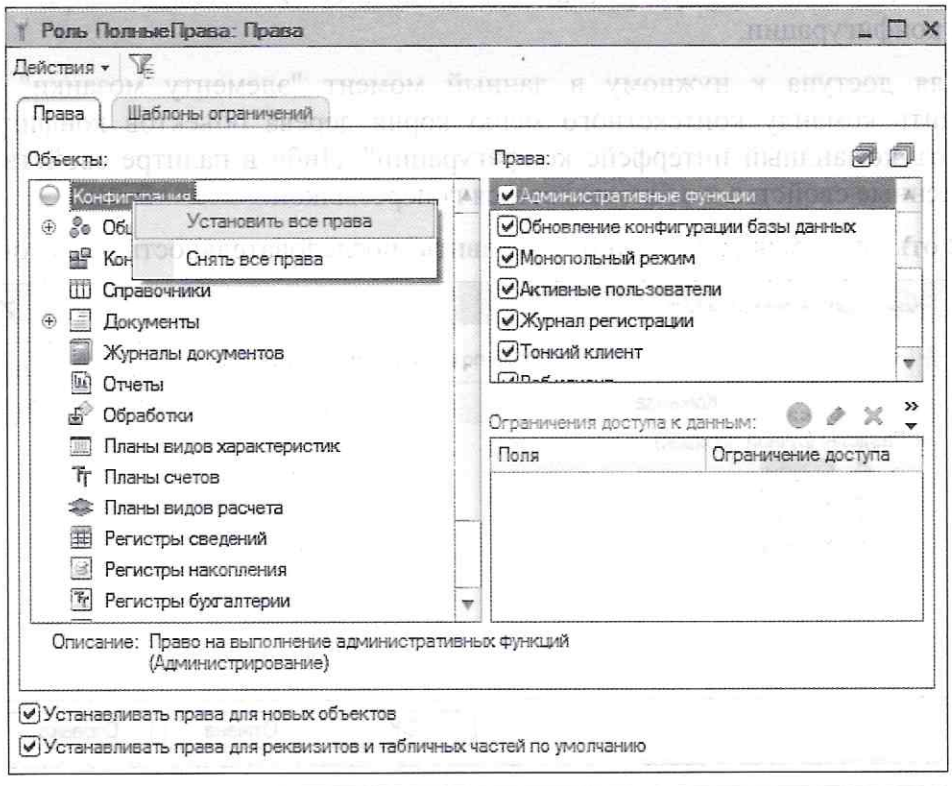
- Первоначально определяется структура системы с точки зрения прикладной области (определяется структура подсистем).
- Определяется состав ролей (хотя это можно сделать и позже).
- При создании объектов конфигурации они относятся к нужным подсистемам, настраиваются права доступа.
- При необходимости меняется расположение и видимость команд.

Переходим к следующему этапу: определим состав ролей. С помощью ролей в дальнейшем будем определять доступность какой-либо функциональности для определенной группы пользователей конфигурации (следует отметить, что роли теперь влияют и на интерфейс системы, а не просто, как раньше, регулируют права доступа). Соответственно, этот состав определим, исходя из возможного перечня таких групп (ролей, которые они будут выполнять при работе в данной конфигурации).

Состав ролей следующий:

- **ПолныеПрава** (должны соответствовать своему наименованию).
- **ОтделПродаж** (предполагается, что у этой роли будет отсутствовать доступ к функциональности системы, связанной с проведением закупок).
- **ОтделЗакупок** (предполагается, что у этой роли будет отсутствовать доступ к функциональности системы, связанной с проведением продаж).

Первой роли установим все права (как на рисунке далее). Также следует не забыть про флаг "Устанавливать права для новых объектов" (совсем не обязательно, но на первых порах данный флаг упростит работу).



Следует отметить, что в реальной жизни нужно очень аккуратно относиться к такому элементу роли, как "Интерактивное удаление" (включение этого флага приводит к возможности удаления объектов без контроля ссылочной целостности).

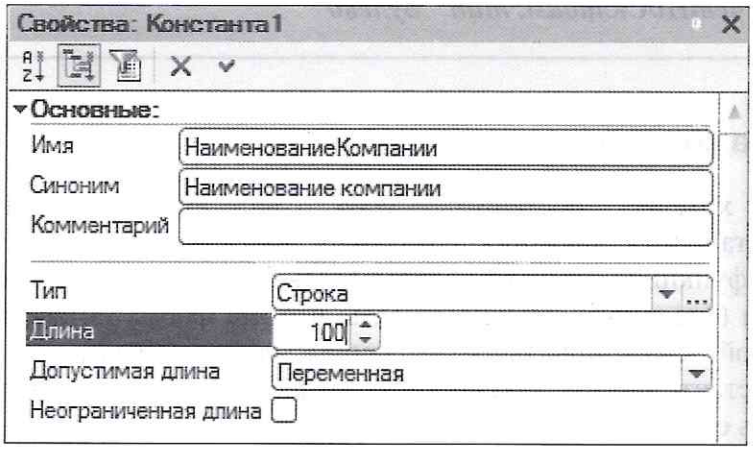
Следующая важная составляющая командного интерфейса - сами команды, но с ними мы будем знакомиться позже.

4.4. Константы

В любой организации существует набор "значений", которые не меняются довольно длительное время. К ним можно отнести название организации (если учет в конфигурации ведется от имени одной организации), юридический адрес, фамилии ответственных лиц и т.д. Для хранения таких значений идеально подходят константы.

4.4.1. Определение, настройка свойств

Создадим константу "НаименованиеКомпании". Для этого сделаем щелчок правой клавишей мыши на ветке "Константы" и выберем пункт контекстного меню "Добавить". В открывшемся окне свойств заполним их необходимыми значениями.



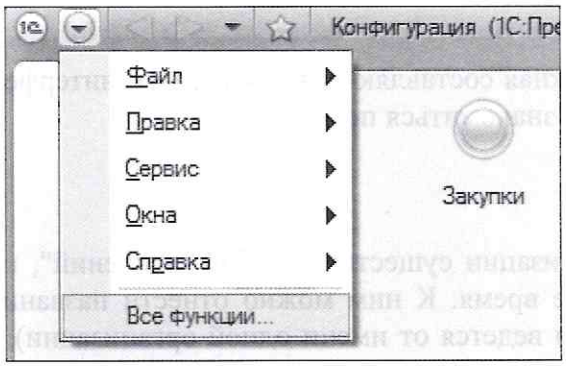
Создав, таким образом, константу, мы определили ВОЗМОЖНОСТЬ хранения в базе значений указанного типа.

Разработчик прикладного решения может работать с константой следующим образом:

```
// на чтение

МояПеременная=Константы.НаименованиеКомпании.Получить();


// установка значения

НовоеЗначение="Новая компания";

Константы.НаименованиеКомпании.Установить(НовоеЗначение);
```

Следует отметить, что данный код приведен для примера (для того, чтобы он стал "рабочим" он в обязательном порядке должен выполняться на сервере, но об этом несколько позже).

Пользователь может посмотреть/установить значение константы, используя команду "Все функции" в режиме исполнения (как показано на рисунке далее).



В открывшейся форме (специальный интерфейсный объект, с функциональностью которого будем знакомиться чуть позже) пользователь будет работать с константами, к которым у него есть доступ (определяемый в ролях).

Практикум №1

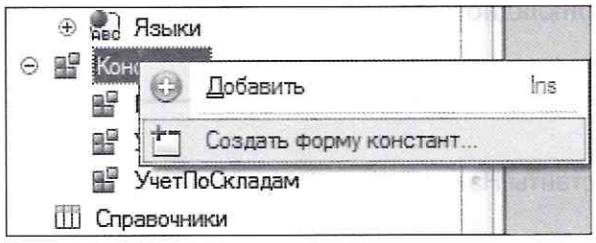
Создайте дополнительно две константы:

- ***Имя: "УчетПоСериям", тип: "Булево"***
- ***Имя: УчетПоСкладам, тип "Булево"***

4.4.2. Форма констант

Когда хотели поставить себя на место пользователя и посмотреть/установить значения созданных нами констант, то использовали команду "Все функции". При этом открывалась форма, сформированная системой автоматически ("на лету"). Можно сказать, что эта форма не реализует никакой дополнительной функциональности, кроме как отображение и изменение значений констант. Форму можно (правильнее даже сказать "нужно") создать явно и научить ее "уму-разуму".

Создадим форму констант. Для этого выполним команду "Создать форму констант" контекстного меню ветви "Константы".

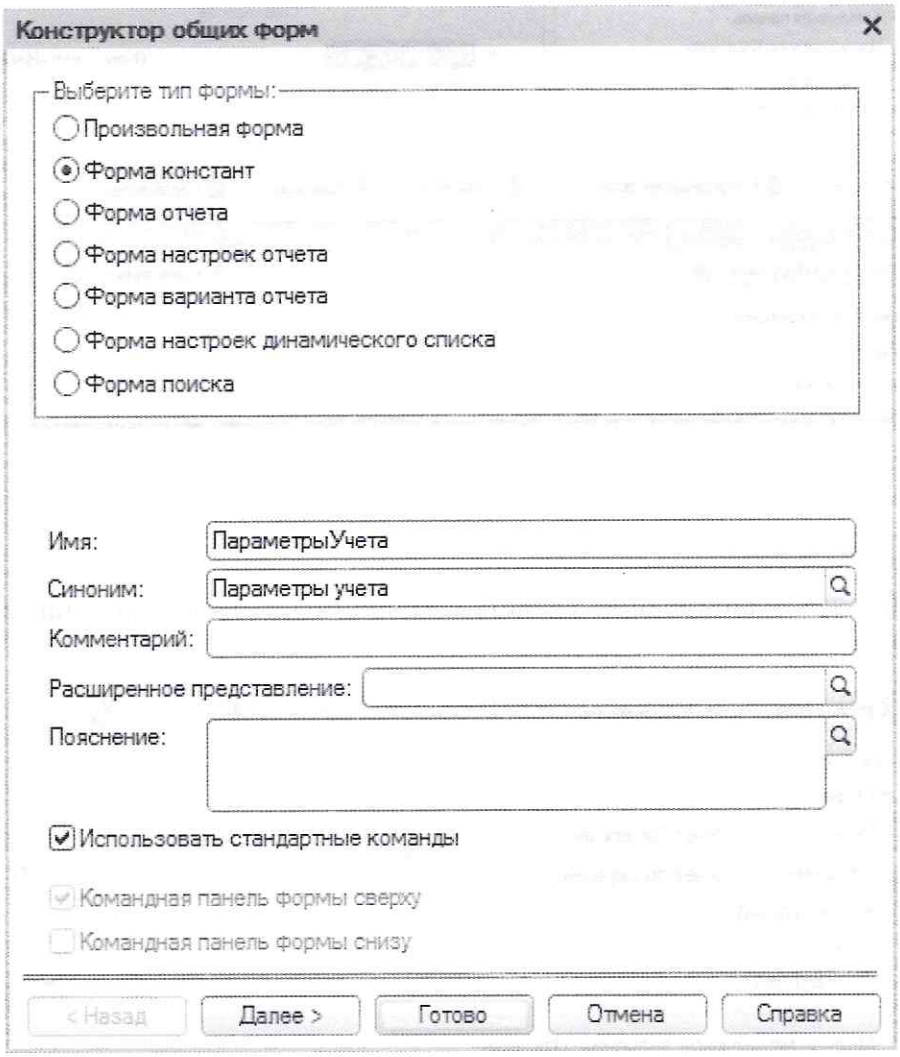


Результатом выполнения команды будет запуск конструктора формы. Следует обратить внимание на то, что повторно запустить этот конструктор для уже созданной формы не получится.

Общие принципы работы с конструктором форм одинаковы и не зависят от того, для какого объекта вы создаете форму. Это несколько этапов:

- Выбор типа формы (один из ВАЖНЕЙШИХ моментов). Выбор типа напрямую влияет на функциональность формы, предоставляемой по умолчанию.

- Определение имени формы.
- Контроль значения флага "Использовать стандартные команды" (установка этого флага приводит к появлению стандартных команд в интерфейсе в тех подсистемах, к которым относится данный объект, в нашем случае это форма). По умолчанию флаг установлен.
- После нажатия на кнопку "Далее" можно определить состав отображаемых (в нашем случае) констант.



После нажатия на кнопку "Готово" откроется окно настройки формы.

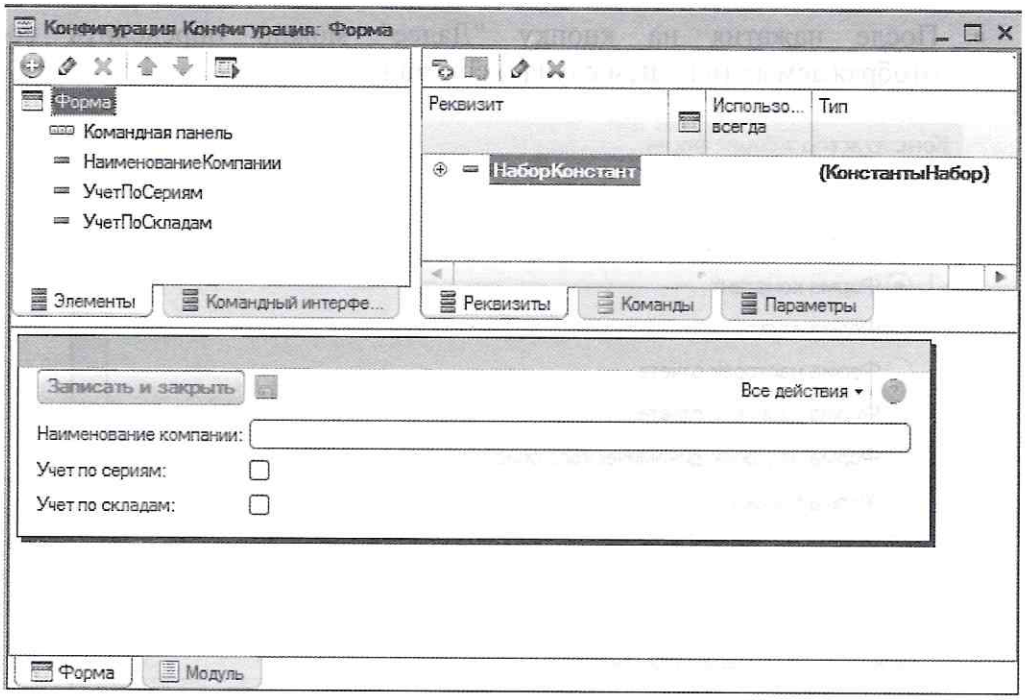
Сразу не будем пытаться "пробежаться" по всем существующим закладкам, с определенными возможностями будем знакомиться позже. Но первое знакомство проведем.

В правой верхней части окна определяется список реквизитов формы (того, что входит в понятие "Данные формы"). Реквизиты также можно назвать источниками функциональности формы.

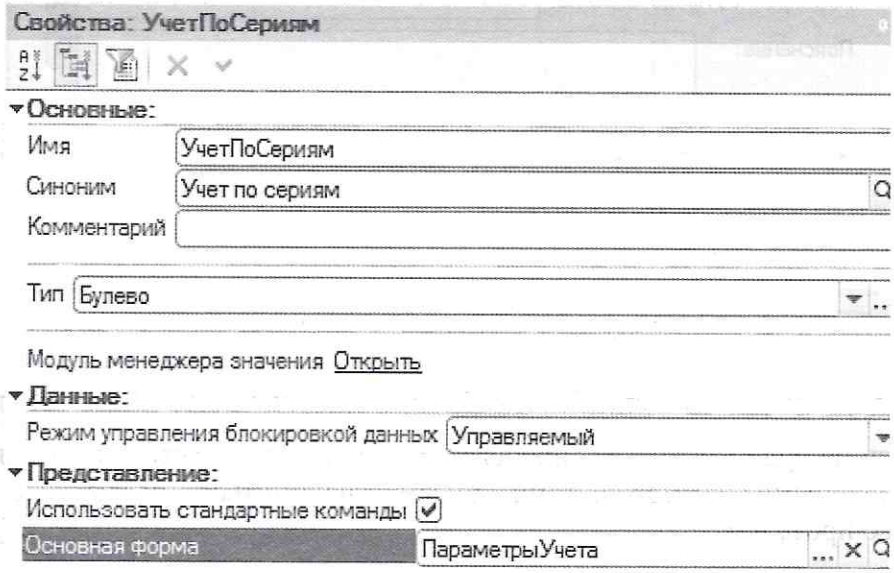
В левой верхней части находится дерево элементов. Если нужно изменить какие-либо свойства элементов, определить обработчики событий, то делать это нужно, начиная с данного дерева. Корень данного дерева определяет саму форму (для изменения свойств формы в целом нужно работать со свойствами именно корневого объекта "Форма").

В нижней части находится область предварительного просмотра (какой внешний вид имеет форма в результате всех ваших действий).

На закладке "Модуль" (ярлыки находятся в самом низу окна) можно прописывать код на встроенном языке системы.



Используя одноименное свойство константы, каждой константе можно назначить свою основную форму:



Если свойство не использовать, то на каждую константу система будет создавать форму автоматически (команда "Все функции").

4.4.3. Механизм работы формы. Первое знакомство

Так как форма является некоторым промежуточным объектом, в ней можно выделить две составляющие: реквизиты и элементы управления. Реквизиты "ближе" к базе данных, элементы управления "смотрят" на пользователя.

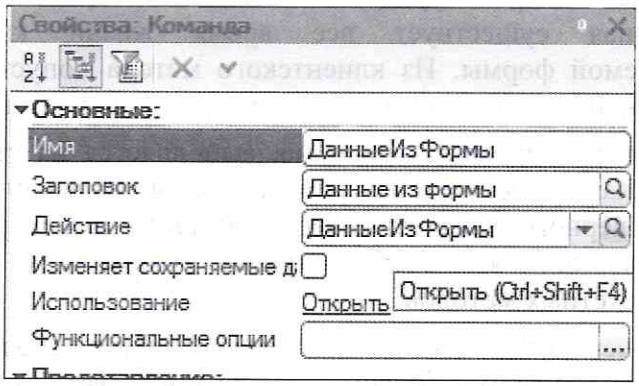
Хочется сделать акцент на том, что сейчас механизм формы рассматривается упрощенно, более подробное знакомство будет позже.

Как же работает сама форма. Когда пользователь хочет что-либо "посмотреть" (в нашем случае - значения констант), он дает команду на открытие этой формы. Система создает экземпляр формы и производит чтение значений констант. Прочитанные значения "копируются" в реквизит формы "НаборКонстант". Форма отправляется клиентскому приложению. Элементы управления отображают значения констант. Связь элемента управления со значением свойства реквизита осуществляется через свойство "Данные" элемента.

При настройке логики работы формы принят следующий подход: элементы управления выступают источниками событий. Программист описывает процедуры – обработчики событий. В обработчиках событий он работает с реквизитами формы. Измененные в обработчиках значения реквизитов автоматически отображаются элементами управления.

При такой логике получается, что в контексте формы есть "собственная копия" данных из базы. Проверим это на практике.

На закладке "Команды формы" окна настройки формы (закладка правой верхней части) создадим команду "ДанныеИзФормы".



Определим обработчик события следующим образом:

```
&НаКлиенте

Процедура ДанныеИзФормы(Команда)

    ЗначениеКонстанты=НаборКонстант.НаименованиеКомпании;

    ПоказатьОповещениеПользователя(ЗначениеКонстанты);

КонецПроцедуры
```

Подобным образом определим команду "ДанныеИзБазы". Текст обработчика команды представлен ниже:

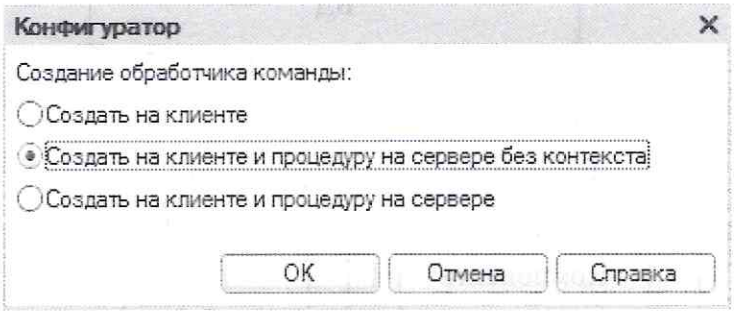
```
&НаКлиенте

Процедура ДанныеИзБазы(Команда)

    ЗначениеКонстанты=ДанныеИзБазыСервер();

    ПоказатьОповещениеПользователя(ЗначениеКонстанты);

КонецПроцедуры


&НаСервереБезКонтекста

Функция ДанныеИзБазыСервер()

    ЗначениеКонстанты=Константы.НаименованиеКомпании.Получить();

    Возврат(ЗначениеКонстанты);

КонецФункции
```

Следует отметить, что каждая процедура, функция или объявление переменной модуля формы должны предваряться одной из следующих директив компиляции:

- &НаКлиенте –процедура/функция выполняется на стороне клиента, а переменная существует все время жизни клиентской части управляемой формы. Из клиентского метода допустимыми являются вызовы клиентских, серверных и серверных внеконтекстных методов.
- &НаСервере –процедура/функция выполняется на стороне сервера, а переменная существует только во время вызова выполнения серверного или серверного внеконтекстного вызова. Для серверных методов допустимыми являются вызовы серверных и серверных внеконтекстных методов.
- &НаСервереБезКонтекста –процедура/функция исполняется на сервере вне контекста формы. Переменные не могут быть внеконтекстными. В таких методах недоступен контекст формы (включая данные формы). Допустимыми являются вызовы только других внеконтекстных методов. При вызове этих методов не выполняется передача данных формы на сервер и обратно. Применение внеконтекстных методов позволяет существенно уменьшить объем передаваемых данных при вызове серверной процедуры из среды клиентского приложения.
- &НаКлиентеНаСервереБезКонтекста – процедура/функция может исполняться в управляемом клиенте или на сервере, при этом контекст формы не доступен.

Отсутствие директивы компиляции перед процедурой означает использование директивы "&НаСервере".

При создании обработчиков событий можно сразу выбирать: создается одна клиентская процедура, или еще создается процедура с местом исполнения – сервер.



В платформе действуют следующие правила по возможностям вызова процедур (проиллюстрированы схемой).

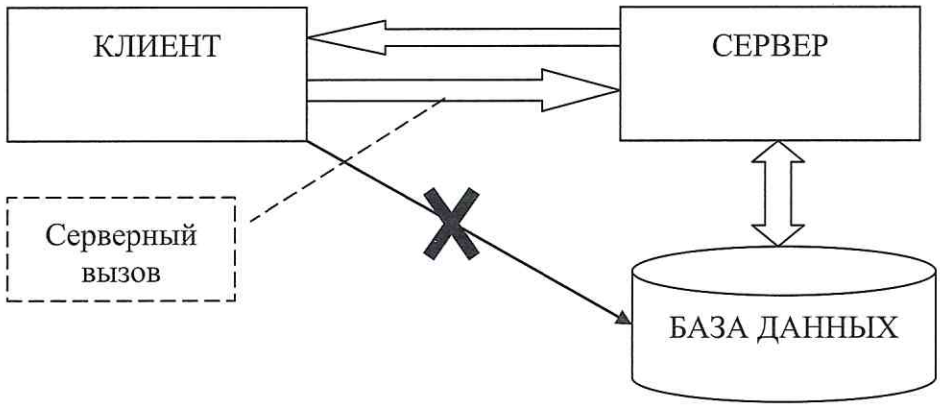


Схема 4.1

Передача управления (с отправкой данных) с сервера на клиента производится автоматически (либо при выполнении неких "системных" действий, либо когда выполнилась последняя строка кода процедуры, исполняемой на сервере). Серверные вызовы (передача управления с клиента на сервер) выполняются автоматически (при выполнении неких "системных" операций), либо в результате явных действий разработчика. При этом серверный вызов может сопровождаться передачей управления и формы на сервер (директива "&НаСервере"), либо передачей управления и только параметров функции/процедуры (директива "&НаСервереБезКонтекста").

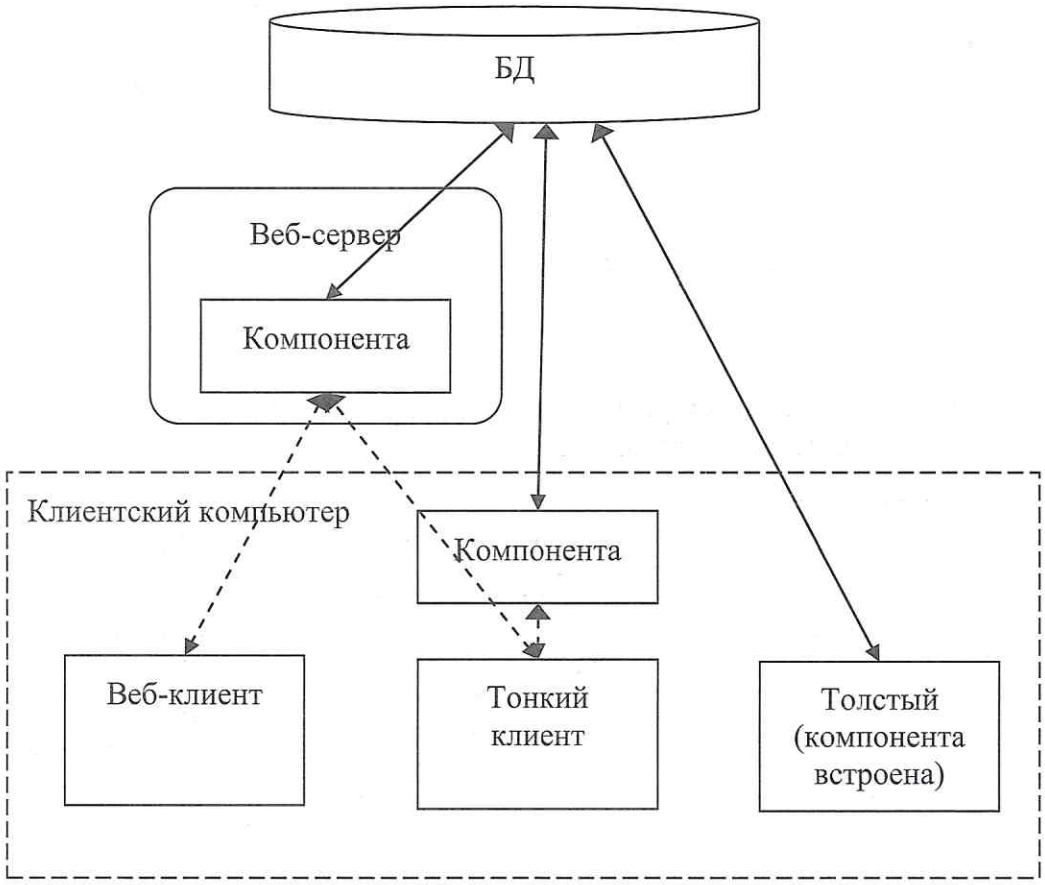
Явная передача управления с сервера на клиент невозможна (т.е. вызывать процедуры/функции с директивой "&Клиент" из процедур/функций, исполняемых на сервере, нельзя).

Из процедуры/функции, объявленной с директивой "&НаСервереБезКонтекста", нельзя произвести вызов процедуры/функции, объявленной с директивой "&НаСервере".

Следует обратить внимание на то, что нужно относиться к "Клиенту" и "Серверу" как к разным, но активно взаимодействующим приложениям (смотрите схему).

В клиент-серверном варианте приложение "Клиент" и является клиентским приложением, "Сервер" это кластер серверов 1С:Предприятие. В файловом

варианте все несколько сложнее.



Работа тонкого и веб клиентов с файловым вариантом базы осуществляется через некую специальную компоненту. При подключении через веб-сервер она запускается в контексте этого сервера (по одному экземпляру на каждую базу). Когда тонкий клиент "напрямую" подключается к базе, компонента запускается в оперативной памяти клиентской машины (и вся обработка данных выполняется в этой компоненте на стороне пользовательского компьютера). Как раз эта компонента и является местом исполнения под названием "Сервер".

4.5. Справочники

Для работы с некоторым множеством значений в системе используются объекты типа "Справочник". Обычно справочниками являются списки материалов, товаров, организаций, валют, сотрудников и др. Название и структура конкретного справочника определяется при его создании в конфигураторе.

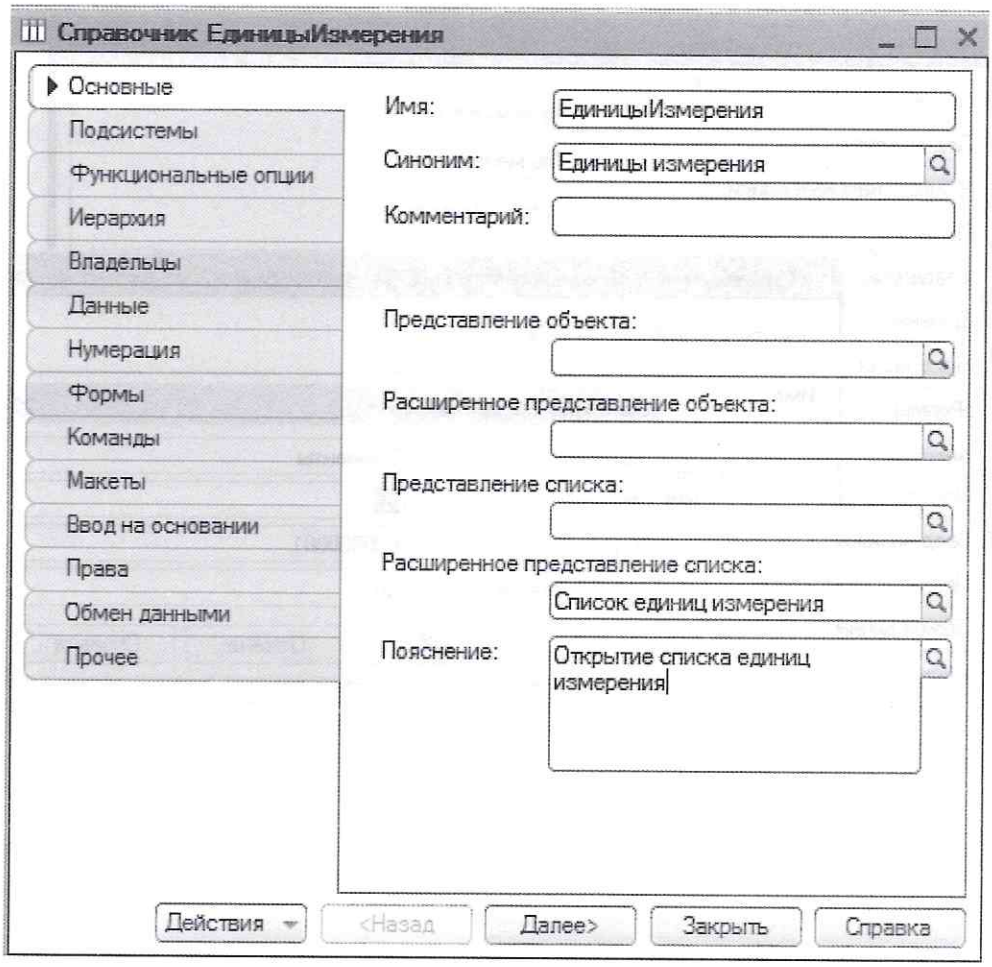
Справочники могут быть иерархическими (реализовано два вида иерархии) и подчиненными. При настройке может определяться состав реквизитов (дополнительных свойств, колонок), табличных частей и ряд других настроек.

4.5.1. Справочники. Первое знакомство

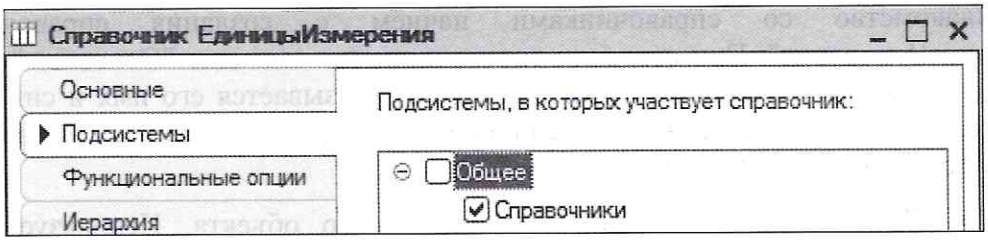
Знакомство со справочниками начнем с создания справочника "ЕдиницыИзмерения". При создании справочника (на закладке "Основные" окна редактирования свойств объекта конфигурации) указывается его имя и синоним (имя должно соответствовать соглашению об именах), и различные представления для этого объекта конфигурации.

- Представление объекта: название одного объекта. Используется в представлении стандартной команды создания объекта.
- Расширенное представление объекта. Используется для формирования заголовка формы объекта.
- Представление списка: название списка объектов. Используется в представлении стандартной команды (команда открытия списка объектов).
- Расширенное представление списка. Используется для формирования заголовка формы списка.

Определим эти свойства в соответствии с рисунком.



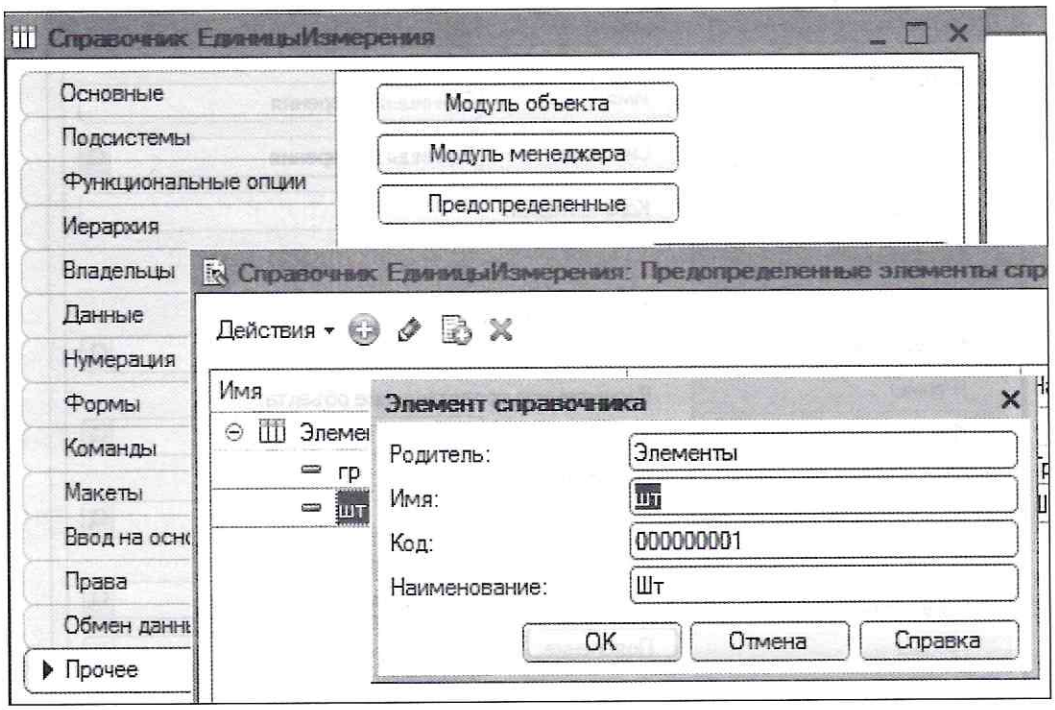
Один из важных этапов: отнесение объекта конфигурации к какой-либо/каким-либо подсистемам.



Отнесем справочник к подсистеме "Справочники", подчиненной подсистеме "Общие". В результате, казалось бы, такого простого действия, в режиме исполнения в панели навигации появилась новая навигационная команда по открытию списка элементов справочника.

На закладке "Данные" можно настроить длину кода и наименования, а также переопределить свойства стандартных реквизитов справочника.

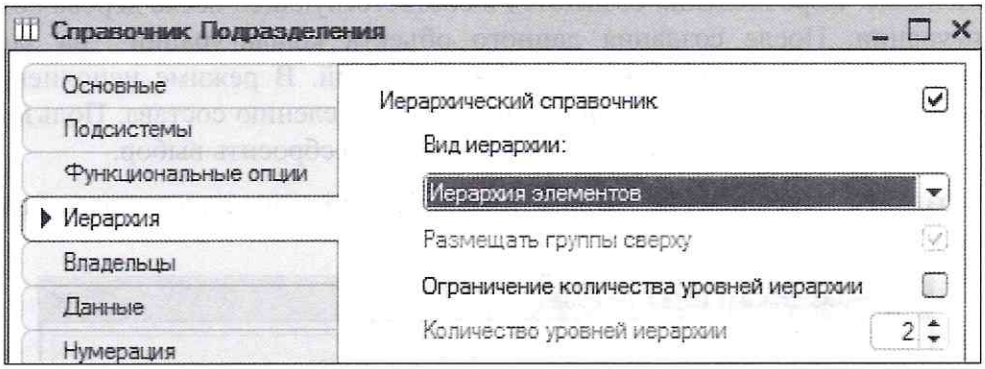
На закладке "Прочие" определим предопределенные элементы. Добавляются они по одноименной кнопке, в форме, открываемой при нажатии на кнопку "Предопределенные".



4.5.2. Иерархия элементов

Иерархия – это возможность связать между собой записи, находящиеся в рамках одной таблицы.

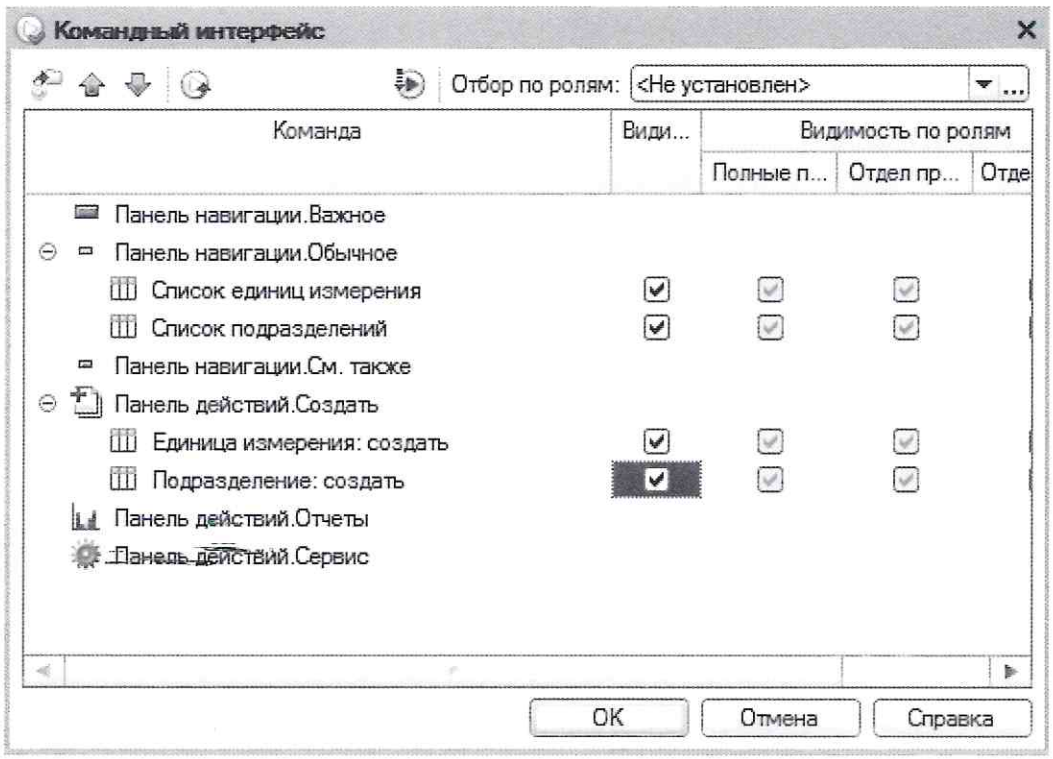
Разбираться с иерархией элементов будем параллельно с созданием справочника "Подразделения".



Справочник нужно отнести к той же подсистеме, что и "Единицы измерения".

После установки флага "Иерархический справочник" в таблице справочника появляется поле "Родитель". В этом поле содержится ссылка на другую запись из этой же таблицы. Выбранный вид иерархии "Иерархия элементов" подразумевает то, что все записи в справочнике равнозначны, и любая запись может выступать "родителем" для другой записи (система не дает "закольцовывать" записи).

Реализуем возможность создания элементов ранее определенных справочников не из командной панели списков, а напрямую с панели действий. Для этого нужно зайти в настройки командного интерфейса подсистемы "Справочники" (подчиненной "Общее").



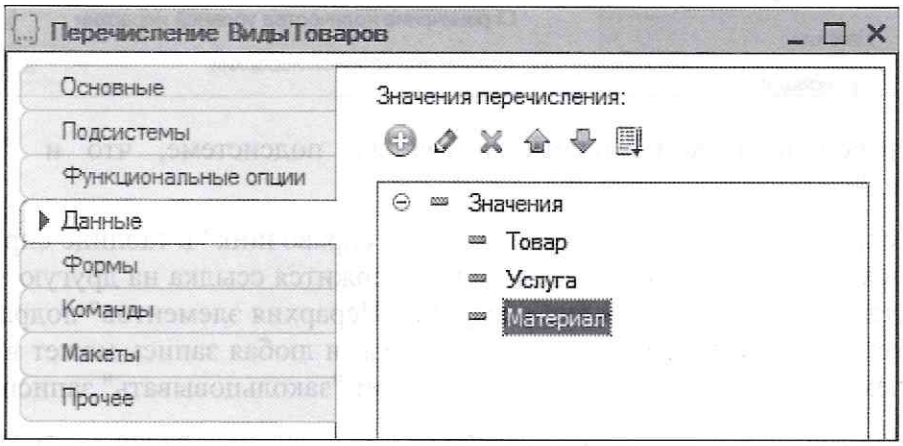
Проверьте работу системы на практике.

4.5.3. Перечисления

Иногда возникает необходимость задать в системе некий линейный и не изменяемый список. Это может быть список видов номенклатуры, операций того или иного документа, времен года, сторон света и т.п.

Как раз для решения подобных задач и предназначен такой объект, как перечисление. Перечисления создаются в соответствующей ветке дерева объектов конфигурации. После создания данного объекта конфигурации на закладке "Данные" определяется нужный перечень значений. В режиме исполнения это список не подлежит ни исправлению, ни переопределению состава. Пользователь может либо выбрать значение из этого списка, либо сбросить выбор.

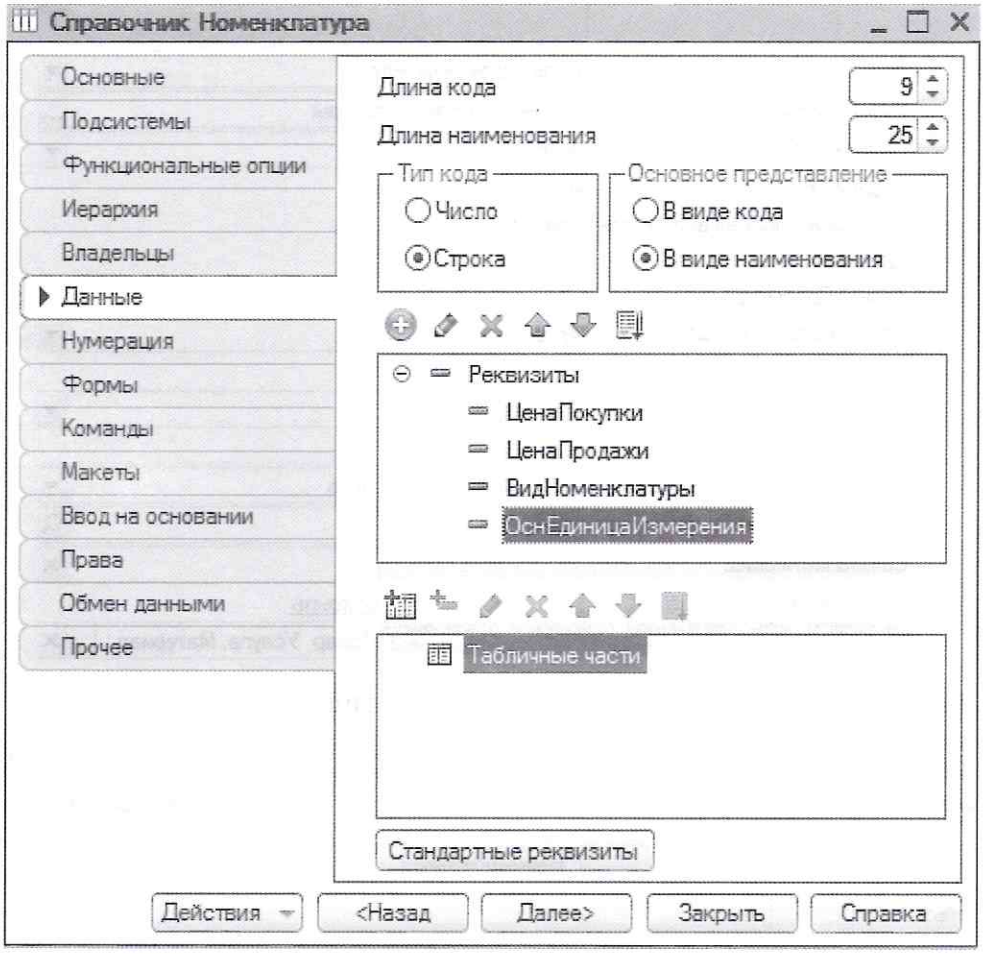
Создайте перечисление "Виды товаров". Перечень видов представлен на рисунке:



4.5.4. Иерархия групп

При установке вида иерархии в значение "иерархия групп" в таблице справочника появляется дополнительное поле "ЭтоГруппа" (тип "Булево"). Все записи в справочнике начинают делиться на две категории: группы и элементы. Группы и элементы могут характеризоваться разными свойствами. При таком виде иерархии в качестве родителей могут выбираться только группы.

Создайте справочник "Номенклатура" с таким видом иерархии.

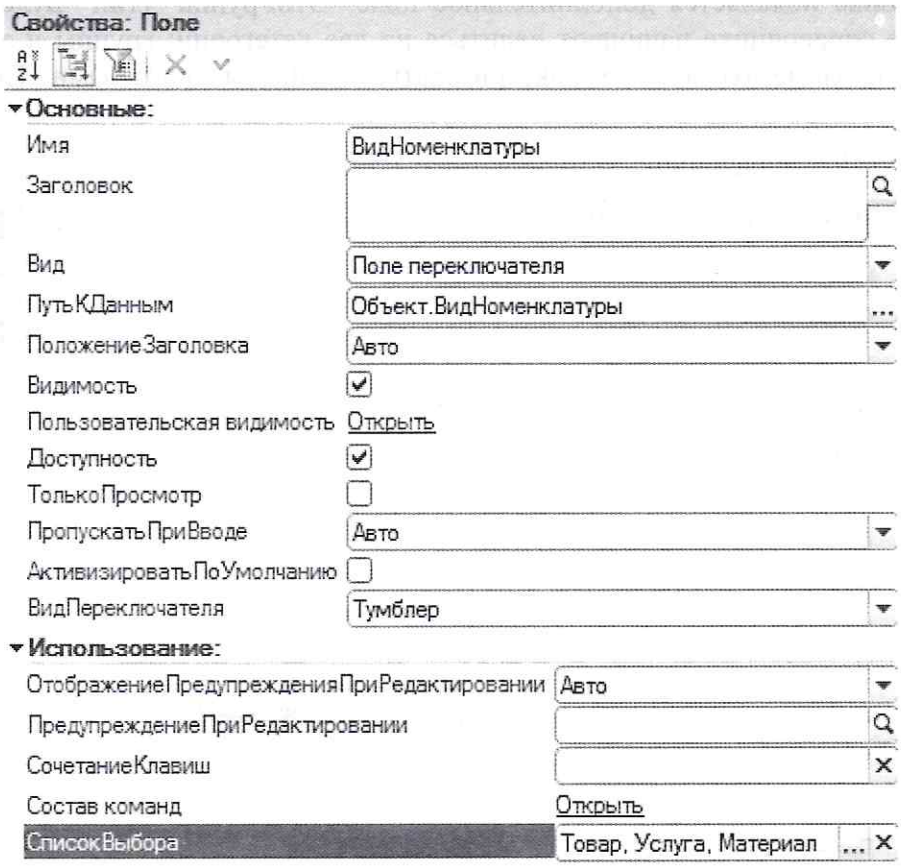


На закладке "Данные" определите дополнительные реквизиты в соответствии с рисунком, представленным ранее.

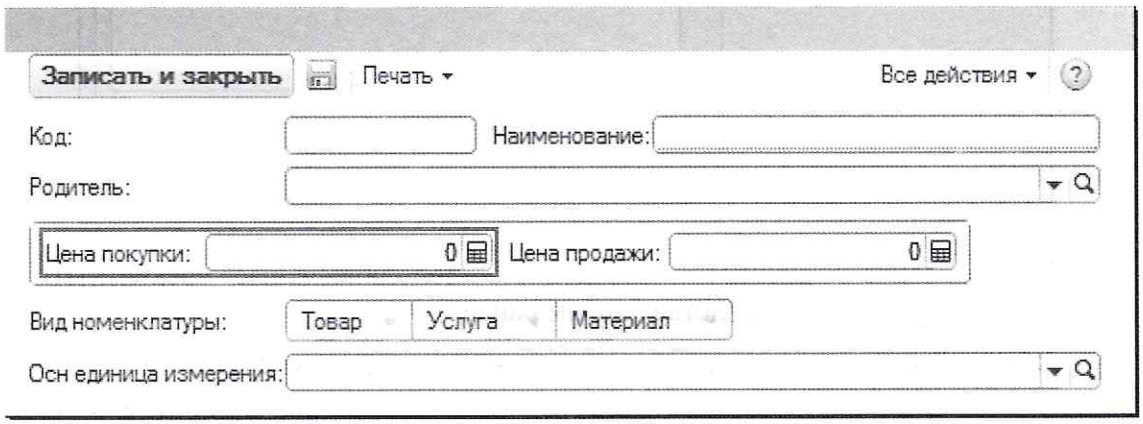
Обращайте внимание на правильное указание типов реквизитов.

Уделим внимание внешнему виду формы элемента справочника. Создадим ее предварительно. Затем в дереве элементов управления определим дополнительный элемент "Группа Обычная группа без отображения", для него определим настройки в соответствии с рисунком. После этого перенесем код и наименование внутрь этой группы. По аналогии поступим с ценами.

Элемент формы "ВидНоменклатуры" оформим как переключатель с видом переключателя "Тумблер". Потребуется работа со свойствами: "Вид", "Вид переключателя" и "Список выбора".

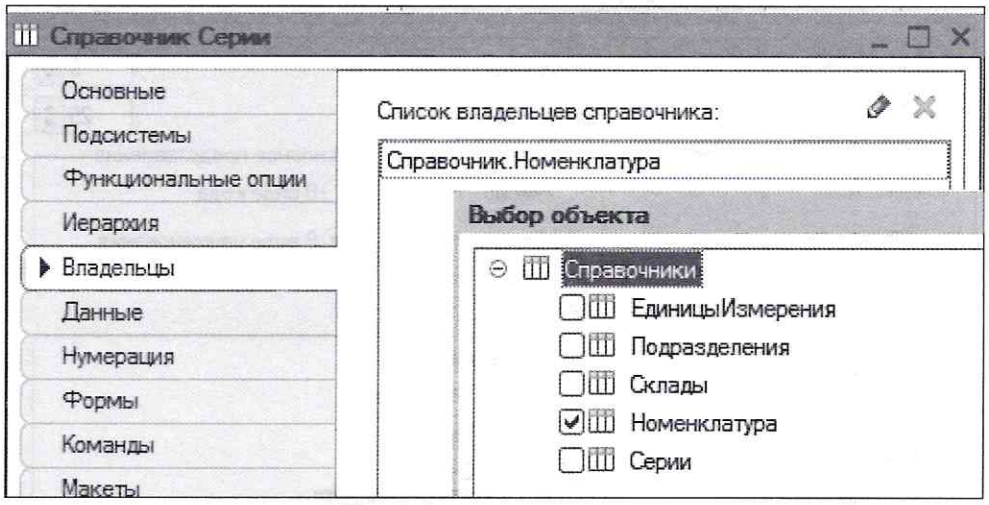


В итоге форма должна принять следующий вид:



4.5.5. Подчиненные справочники

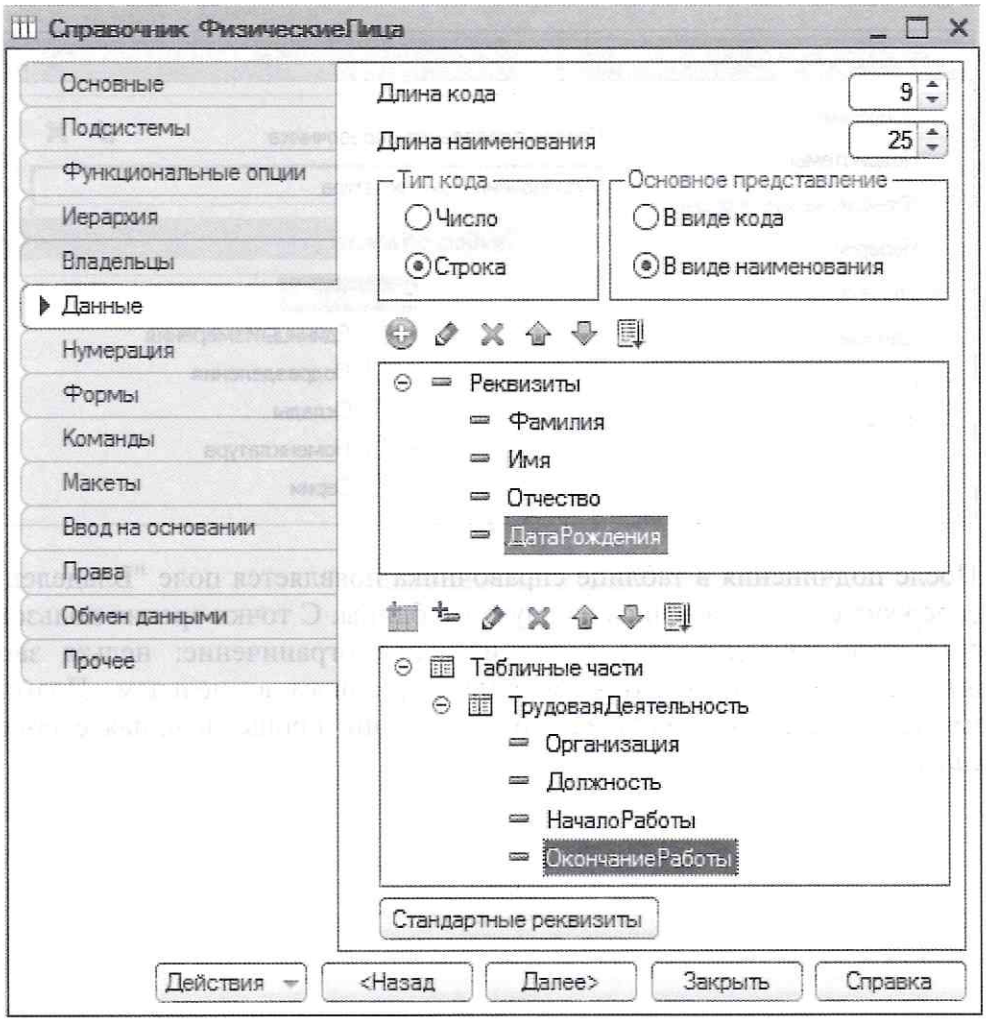
Создадим справочник "Серии". На закладке "Владельцы" укажем, что справочник "Номенклатура" будет являться владельцем создаваемого справочника.



После подчинения в таблице справочника появляется поле "Владелец". Это поле содержит ссылку на запись из другой таблицы. С точки зрения пользователя на справочник накладывается дополнительное ограничение: нельзя записать элемент подчиненного справочника с не выбранным владельцем. Поэтому на данном этапе заполнять подчиненный справочник проще, начиная с объекта – владельца.

4.5.6. Табличные части

Создадим справочник "Физические лица". Состав реквизитов приведен на рисунке. Справочник без иерархии, без подчинения.



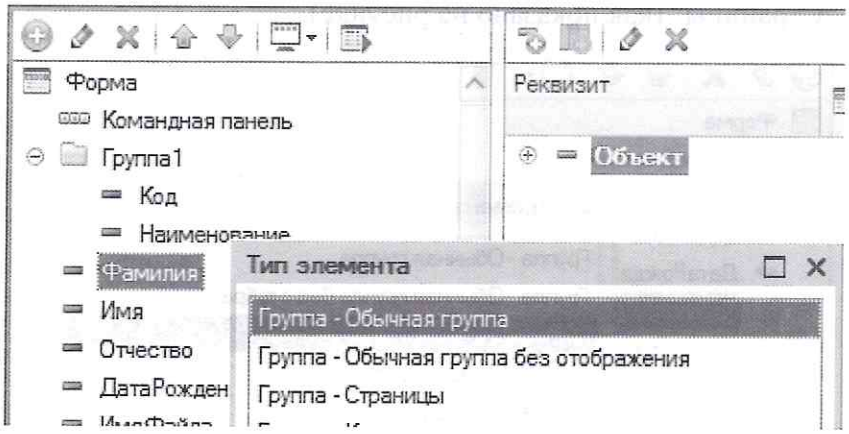
Также нужно создать табличную часть "Трудовая деятельность", состав ее реквизитов тоже приведен на рисунке. При создании табличной части обратите внимание на то, что есть команда по созданию табличной части, а есть команда по созданию реквизита табличной части.

На закладке "Данные", воспользовавшись кнопкой "Стандартные реквизиты", переопределим синоним реквизита "Наименование" на "ФИО".

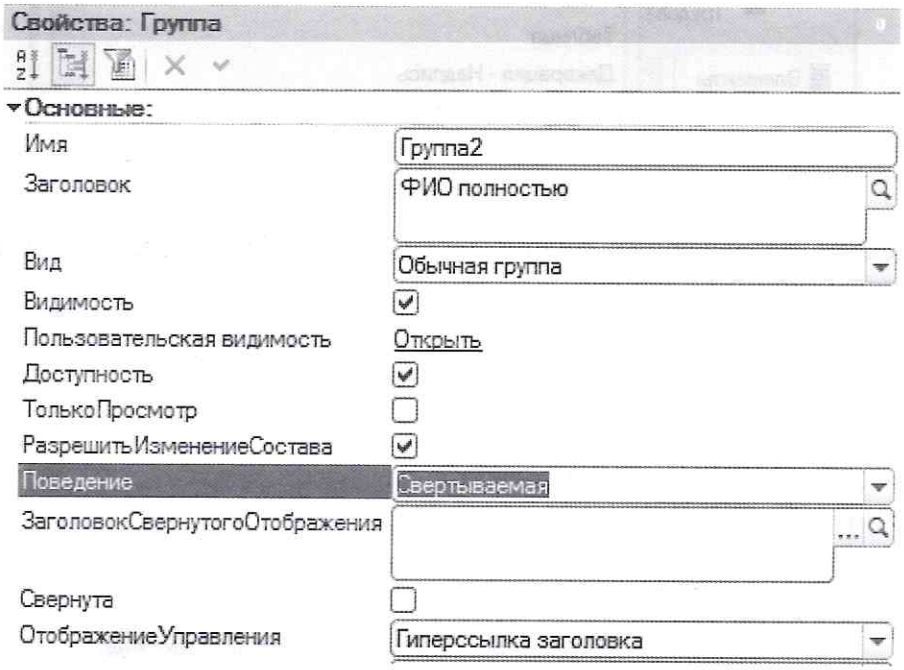
Создайте форму списка справочника, обратите внимание на тип и другие свойства основного реквизита формы.

После этого создадим форму элемента справочника и приведем ее к желаемому виду.

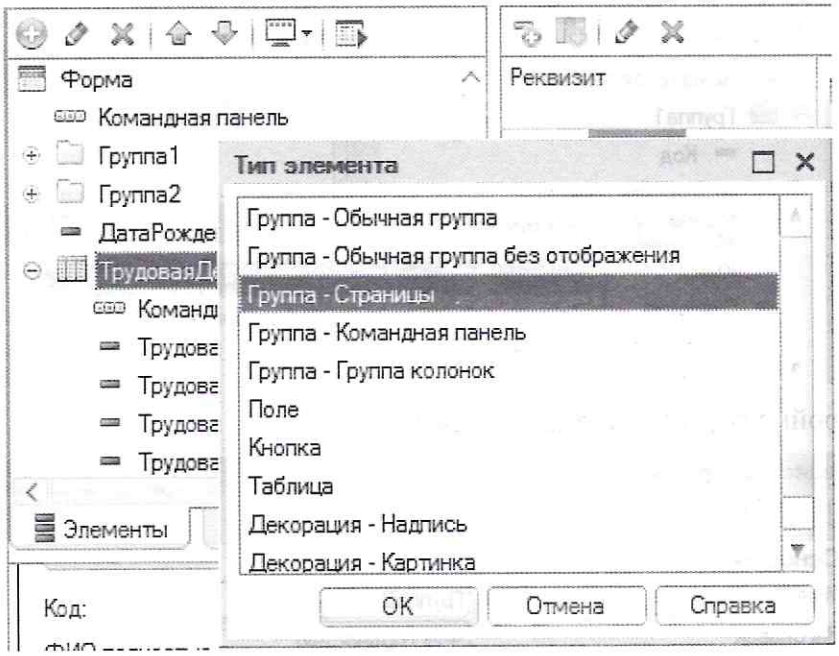
Создадим группу без отображения, разместим в ней код и наименование. Создадим обычную группу, разместим в ней элементы формы "Фамилия", "Имя", "Отчество".



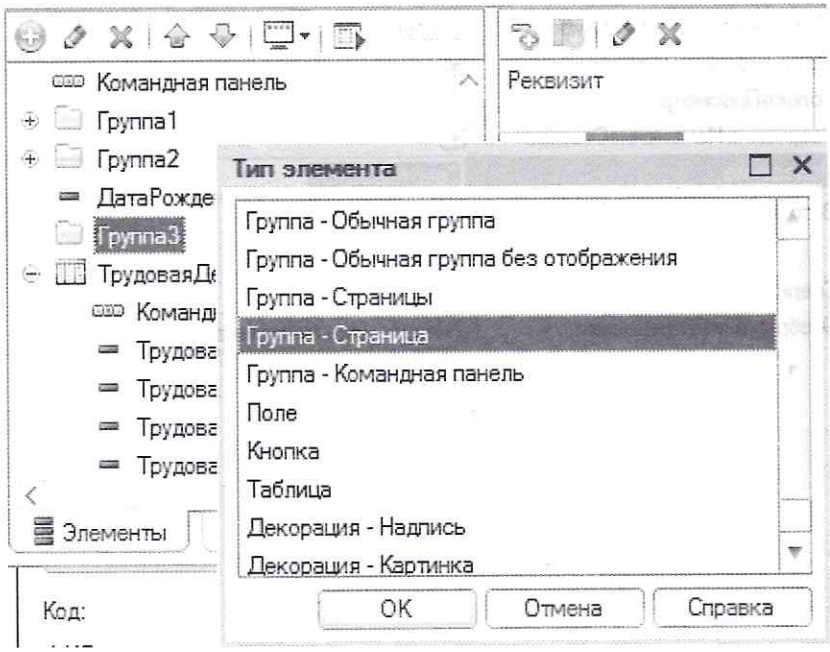
Настройки группы приведены ниже:



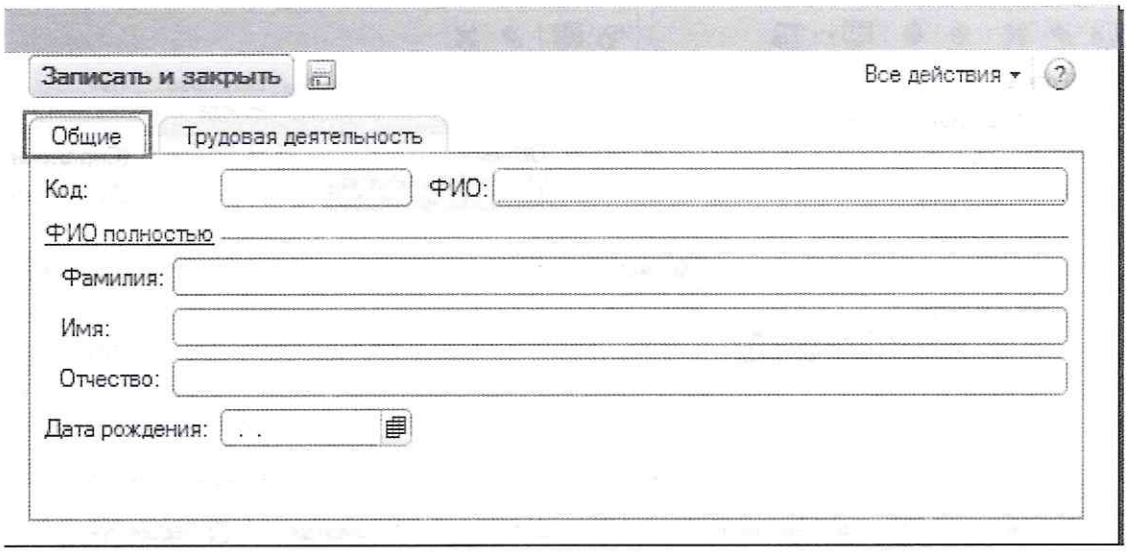
Также требуется, чтобы на форме было две странички. На первой располагались реквизиты справочника, на второй страничке - табличная часть и командная панель этой табличной части. Добавим в форму новый элемент, тип элемента - "Страницы" (как показано на рисунке).



Сделаем добавленную группу текущей и добавим два новых элемента управления с типом "Страница".



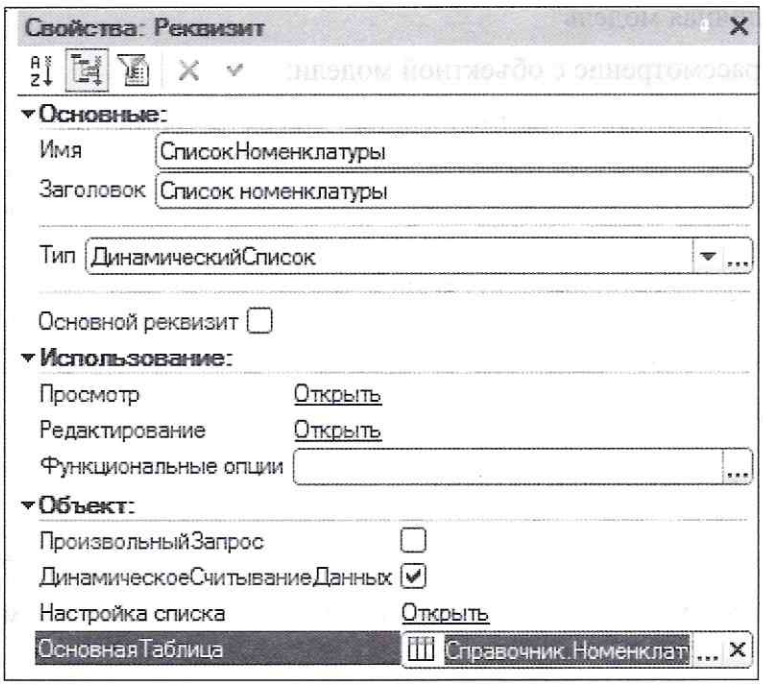
Остается "перетащить" элементы управления на нужные страницы, чтобы добиться внешнего вида формы, приведенного далее.



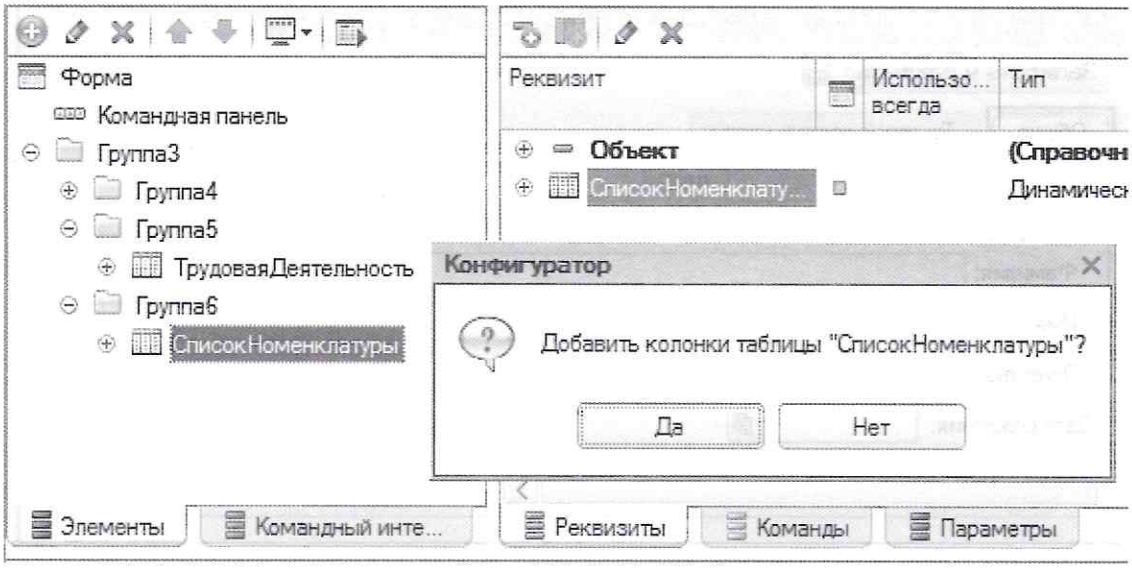
4.5.7. Расширение функциональности формы

Сразу хочется обратить внимание на то, что рассматриваемый пример не имеет смысла с точки зрения реальных учетных задач. На этом примере рассматривается потенциальная возможность платформы (пример можно назвать "Знакомство с формой. Продолжение").

Добавим (используя правую верхнюю область окна настройки формы) новый реквизит "СписокНоменклатуры", определим у него тип "ДинамическийСписок". В качестве основной таблицы выберем "Справочник.Номенклатура".



После добавления реквизита его нужно "перетащить" в дерево элементов управления. На вопрос о добавлении колонок ответить положительно.



Таким же образом можно попытаться добавить еще один реквизит, с типом "СправочникОбъект.Номенклатура" и разобраться с тем, как он "работает".

4.5.8. Работа с данными справочника

Разберемся, каким образом можно манипулировать данными справочника.

Следует отметить, что "1С:Предприятие" реализует две модели по работе с данными:

- Объектная модель
- Табличная модель

Начнем рассмотрение с объектной модели:

Чтобы прочитать наименования всех элементов справочника "Номенклатура", потребуется написать следующий код:

```
// На чтение

Выборка = Справочники.Номенклатура.Выбрать();

Пока Выборка.Следующий() Цикл

    Переменная = Выборка.Наименование;

    // обработка полученного значения

КонецЦикла;
```

Это был пример на чтение данных, разберем пример на модификацию данных.

Для выборки и переноса всех элементов в предопределенную группу:

```
// на запись
ПредопределеннаяГруппа=Справочники.Номенклатура.ДляПереноса;
Выборка = Справочники.Номенклатура.Выбрать();
Пока Выборка.Следующий() Цикл
    Если Выборка.Ссылка = ПредопределеннаяГруппа Тогда
        Продолжить;
    КонецЕсли;
    ПолученныйОбъект = Выборка.ПолучитьОбъект();
    ПолученныйОбъект.Родитель = ПредопределеннаяГруппа;
    ПолученныйОбъект.Записать();
КонецЦикла;
```

Практикум №2

Подумайте, к какому результату приведет попытка выполнения следующего фрагмента кода:

```
Пока Справочники.Номенклатура.Выбрать().Следующий() Цикл
// в теле цикла "что-то делаем" с наименованием…
КонецЦикла
```

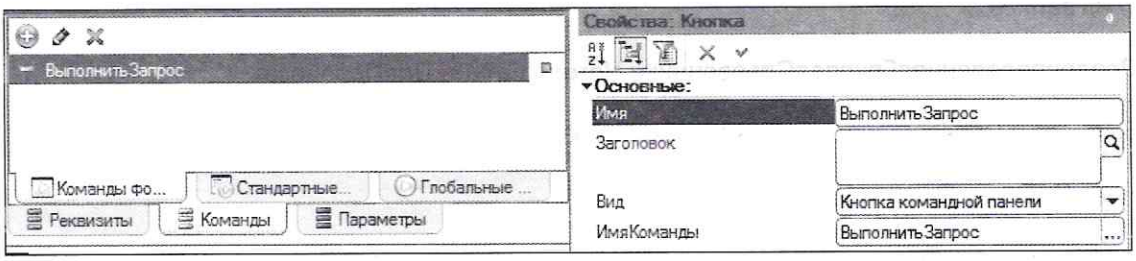
Табличная модель

С составом таблиц (и некоторыми конструкциями языка запросов), с помощью которых можно работать с данными справочника, познакомимся с помощью обработки "ТабличнаяМодель".

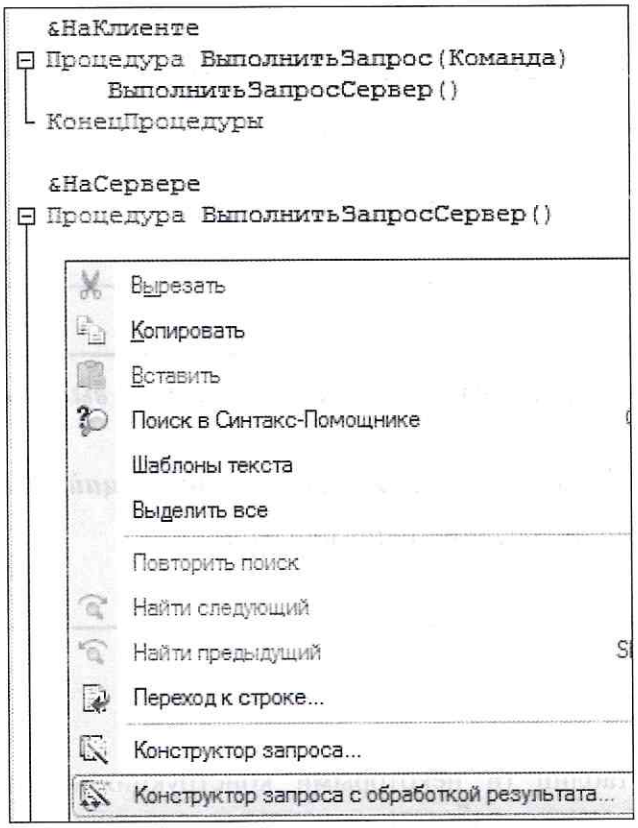
Обработки могут использоваться для выполнения каких-либо "сервисных" манипуляций с данными (выгрузка, загрузка данных, групповая установка какого-либо реквизита и т.д.). Обработки (забегая вперед, также как и отчеты) могут входить в конфигурацию и могут находиться в виде отдельных файлов.

Создадим обработку в соответствующей ветви дерева объектов конфигурации. Создадим основную форму обработки.

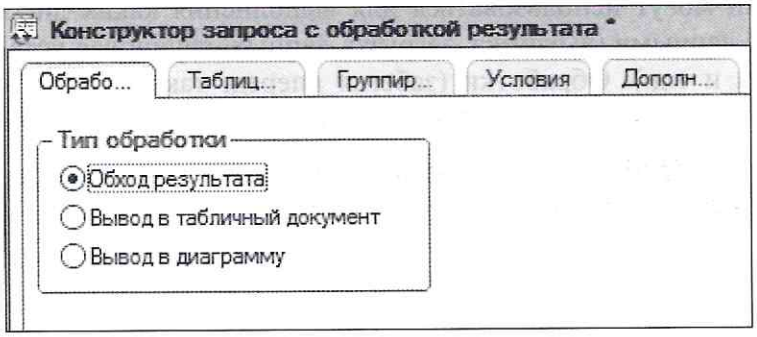
В форме обработки определим команду "ВыполнитьЗапрос" (на закладке "Команды" зайти в закладку "Команды формы").



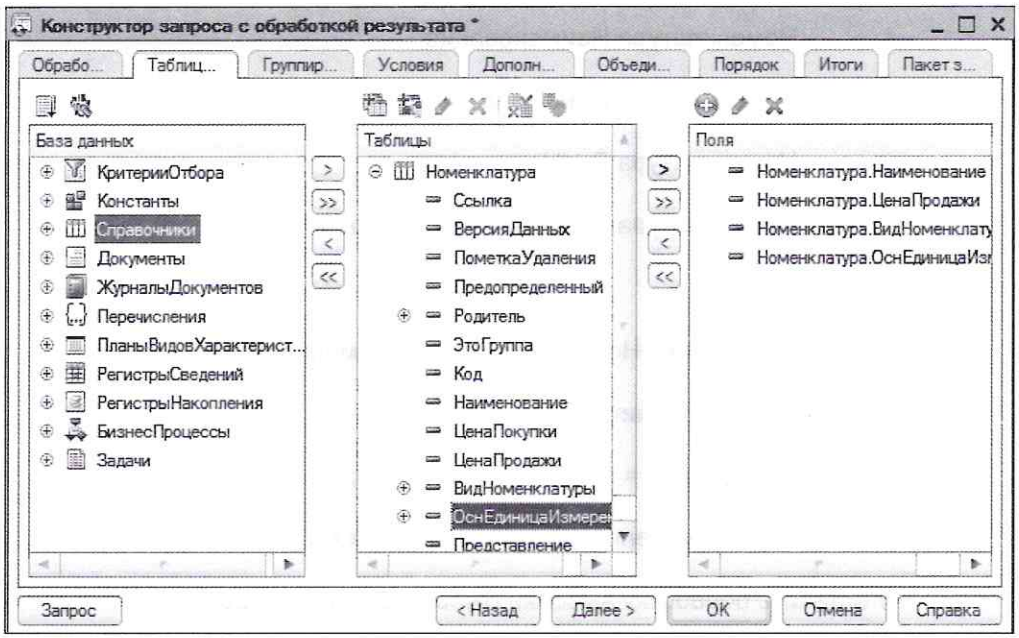
При создании обработчика команды выберем вариант "клиентская процедура с серверным вызовом".



В форме конструктора укажем тип обработки, как показано на рисунке:



После этого перейдем к закладке "Таблицы и поля".



В левой части окна приведены все существующие таблицы базы данных (по которым может быть построен запрос). В средней части указываются таблицы, которые будут выступать в качестве источников данных запроса. В правой части - поля, выбранные из таблиц-источников.

На любом этапе работы с конструктором запроса по нажатию на кнопку "Запрос" можно просмотреть текст запроса.

```
ВЫБРАТЬ

        Номенклатура.Наименование,  Номенклатура.ЦенаПродажи,

        Номенклатура.ВидНоменклатуры,

        Номенклатура.ОснЕдиницаИзмерения

ИЗ

        Справочник.Номенклатура КАК Номенклатура
```

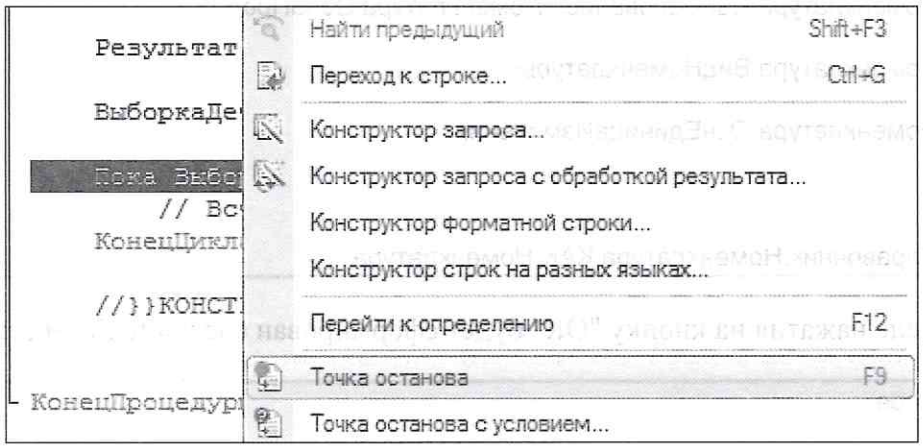
После нажатия на кнопку "ОК" будет сформирован следующий текст:

```
&НаСервере

Процедура ВыполнитьЗапросСервер()

        //{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА

        // Данный фрагмент построен конструктором.

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



        Запрос = Новый Запрос;

        Запрос.Текст =
```

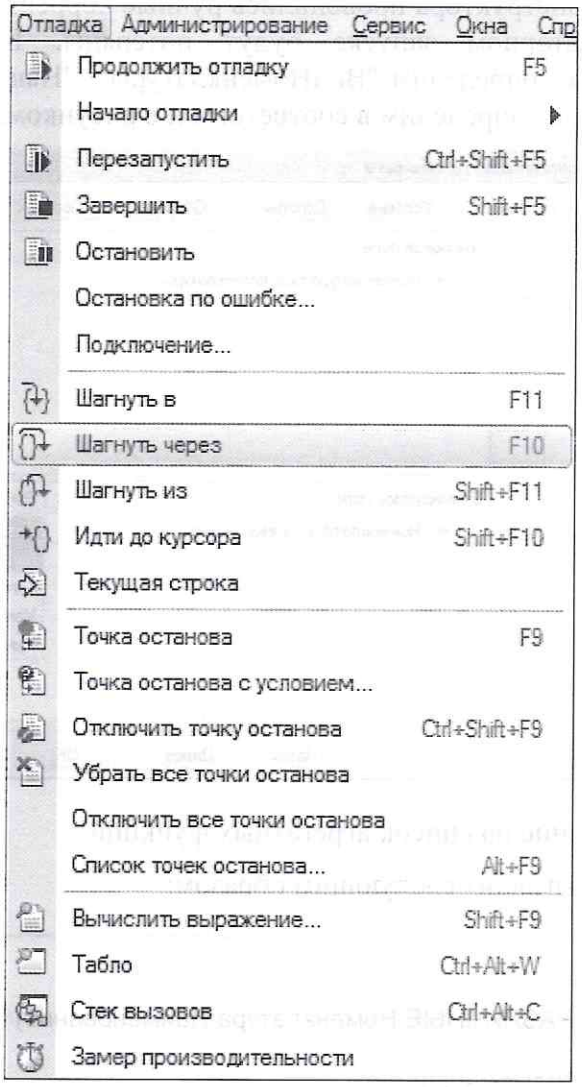
```
        "ВЫБРАТЬ
        |    Номенклатура.Наименование,
        |    Номенклатура.ЦенаПродажи,
        |    Номенклатура.ВидНоменклатуры,
        |    Номенклатура.ОснЕдиницаИзмерения
        |ИЗ
        |    Справочник.Номенклатура КАК Номенклатура";
    Результат = Запрос.Выполнить();
    ВыборкаДетальныеЗаписи = Результат.Выбрать();
    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
        // Вставить обработку выборки ВыборкаДетальныеЗаписи
    КонецЦикла;
    //}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
КонецПроцедуры
```

Знакомиться с данными, полученными с помощью запроса, будем с использованием встроенного в конфигуратор отладчика.

Предварительно (используя контекстное меню) установим в коде "Точку останова" (как показано на рисунке).

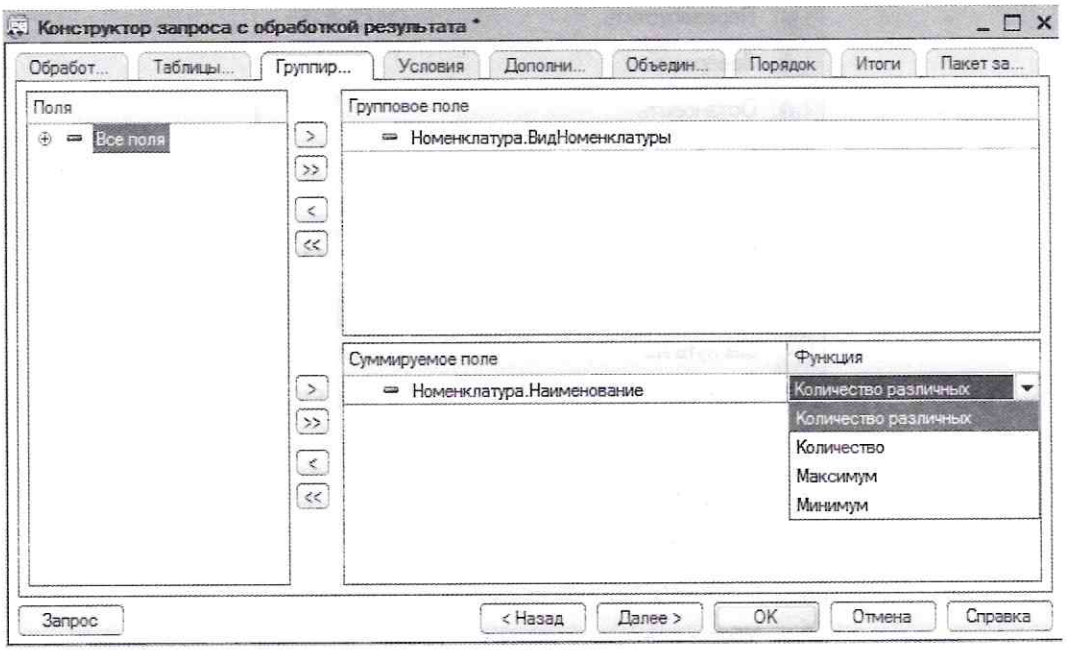


Остается запустить программу в режиме отладки и воспользоваться командами, расположенными в пункте главного меню программы "Отладка".



На данном этапе будут полезны команды "Шагнуть через", "Вычислить выражение", "Табло".

После проведения знакомства, используя контекстное меню, повторно запустим конструктор запроса с обработкой результата. Следует иметь в виду, что если после запуска конструктора проводились ручные корректировки кода, то эти изменения при повторном запуске будут потеряны. В качестве полей, выбираемых запросом, определим "ВидНоменклатуры", "Наименование". Состав закладки "Группировка" определим в соответствии с рисунком.



Обратите внимание на список агрегатных функций.

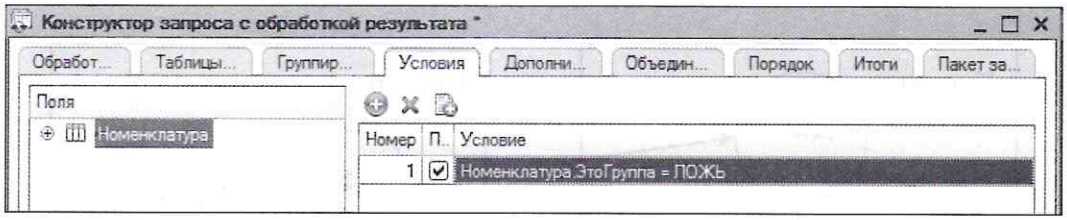
Текст запроса выглядит следующим образом:

```
ВЫБРАТЬ
    КОЛИЧЕСТВО(РАЗЛИЧНЫЕ Номенклатура.Наименование) КАК Наименование,
    Номенклатура.ВидНоменклатуры
ИЗ
    Справочник.Номенклатура КАК Номенклатура

СГРУППИРОВАТЬ ПО
    Номенклатура.ВидНоменклатуры
```

Просмотрите результат ваших действий в отладчике.

Следует обратить внимание, что если в тексте запроса используется раздел "Сгруппировать", то к полям выборки запроса предъявляется следующее требование: "Все поля выборки должны делиться на поля, по которым выполняется группировка, поля, получаемые от них через точку, и на агрегатные функции".

Запустите повторно конструктор запроса с обработкой результата. Доопределим закладку "Условия".



В результате получим следующий текст запроса:

```
ВЫБРАТЬ
    КОЛИЧЕСТВО(РАЗЛИЧНЫЕ Номенклатура.Наименование) КАК Наименование,
    Номенклатура.ВидНоменклатуры
ИЗ
    Справочник.Номенклатура КАК Номенклатура
ГДЕ
    Номенклатура.ЭтоГруппа = ЛОЖЬ
СГРУППИРОВАТЬ ПО
    Номенклатура.ВидНоменклатуры
```

Проверьте работу механизма с помощью отладчика.

Практикум №3

Выполните запросы к другим справочникам, определенным на данный момент в системе.

4.5.9. Реквизиты формы, объекты базы

Основной реквизит формы элемента справочника "Физические лица" имеет тип "СправочникОбъект.ФизическиеЛица". Следует обратить внимание на то, что это специальный тип данных, а не объект, с помощью которого можно манипулировать данными элемента справочника в базе (объект базы не может существовать на стороне клиента).

Схематично представить работу механизма формы можно следующим образом:

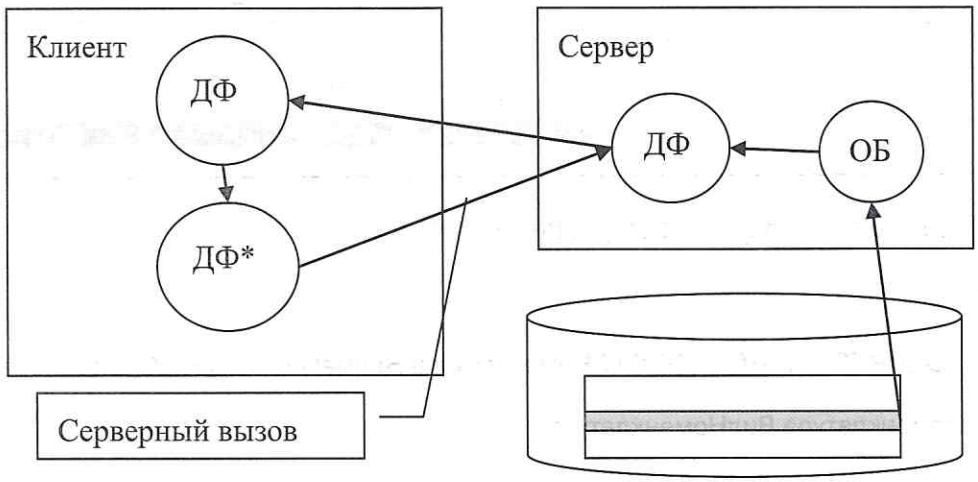


Схема 5.1

Где ОБ – Объект базы данных (тот, у которого есть метод "Записать" и т.д.)

ДФ – Данные формы (некое "представление" объекта базы данных, способное существовать как на стороне сервера, так и на стороне клиента).

ДФ* - Данные формы, содержащие изменения, сделанные пользователем на стороне клиента.

При серверном вызове производится синхронизация данных с объектом формы, находящемся на сервере.

Преобразование ОБ в ДФ (и наоборот) система в ряде случаев выполняет автоматически. Также существуют методы, которые могут выполнять это явным образом:

```
ОБ = РеквизитФормыВЗначение("ДФ");

ЗначениеВРеквизитФормы(ОБ, "ДФ");
```

Пример использования данных методов будет рассмотрен позже.

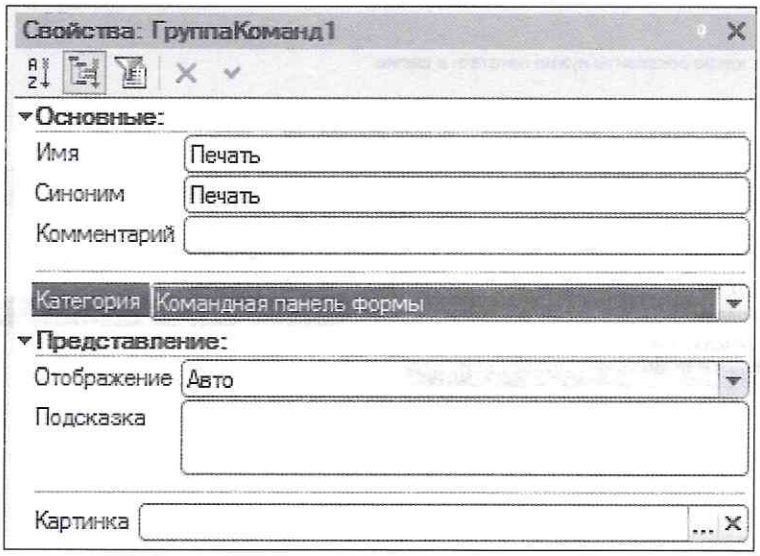
4.5.10. Создание печатных форм

Начнем с того, что вернемся к основному режиму запуска "Управляемое приложение". Для этого нужно откорректировать одноименное свойство конфигурации.

При создании печатных форм придется работать с такими объектами как макет и табличный документ. Суть алгоритма печати:

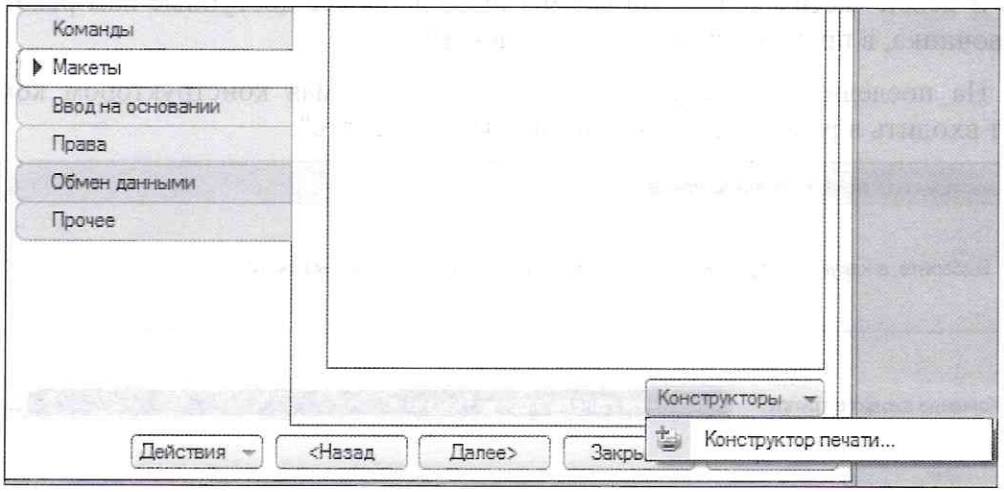
Макет содержит набор именованных областей. В каждой именованной области определены элементы оформления и набор параметров, параметров расшифровки. В процедуре, реализующей печать, первоначально создается "ТабличныйДокумент". Из макета получается очередная именованная область. Выбранные данные (которые хотим вывести на печать) записываются в параметры, параметры расшифровки области. Заполненная данными область включается в табличный документ. В конце алгоритма табличный документ открывается.

Для создания печатной формы воспользуемся соответствующим конструктором, но предварительно создадим группу команд "Печать" (внутри ветви "Общие" дерева объектов конфигурации).

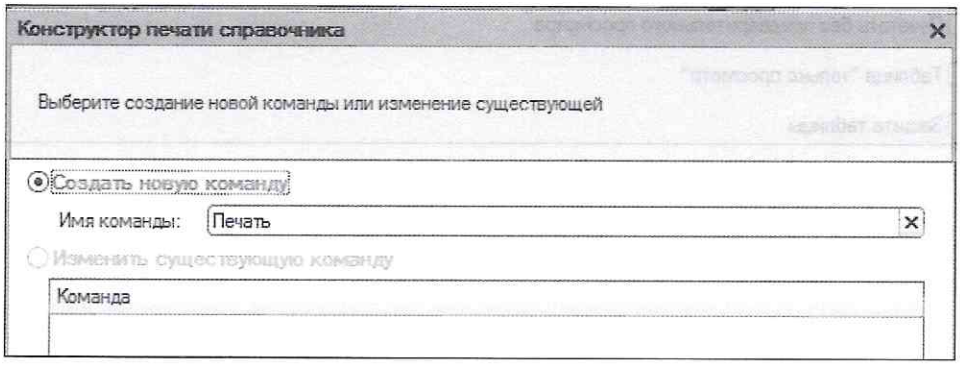


Один из важных моментов при создании группы это определение категории.

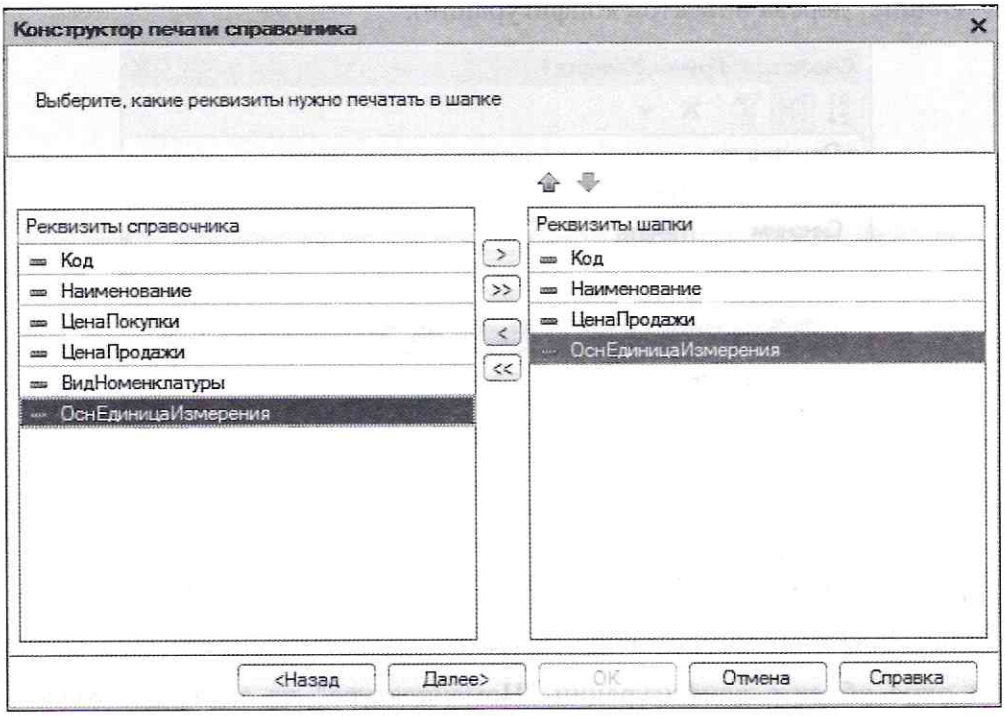
В окне объекта конфигурации "Номенклатура" на закладке "Макеты" по кнопке "Конструктор печати" откроем одноименный конструктор.



В открывшейся форме указываем, что создается новая команда. На этом же этапе можно переопределить имя создаваемой команды.

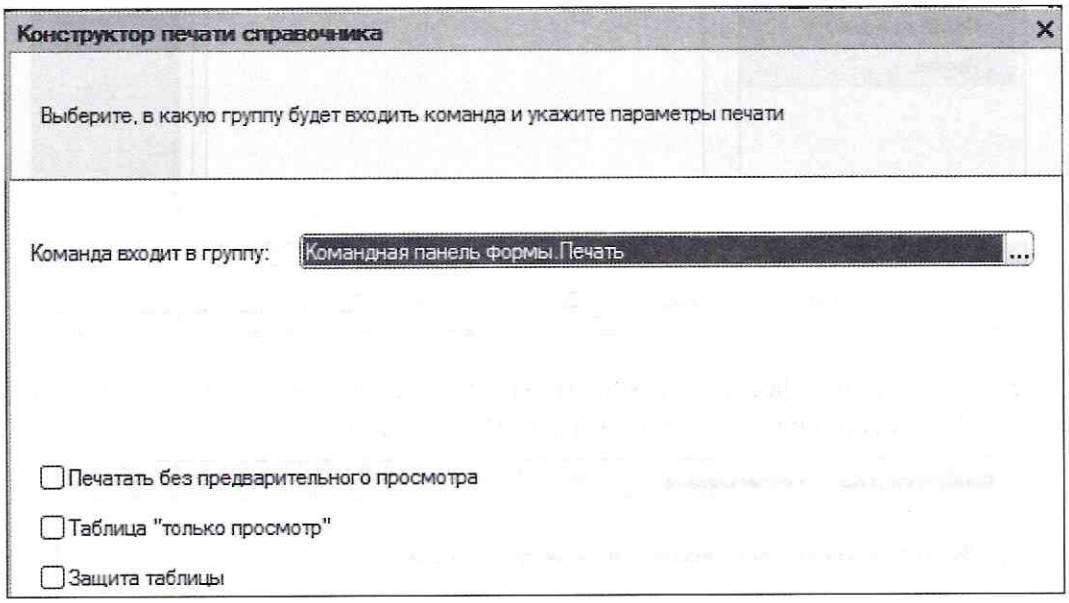


На следующем этапе из реквизитов справочника выбираем те, которые хотим видеть в печатной форме:

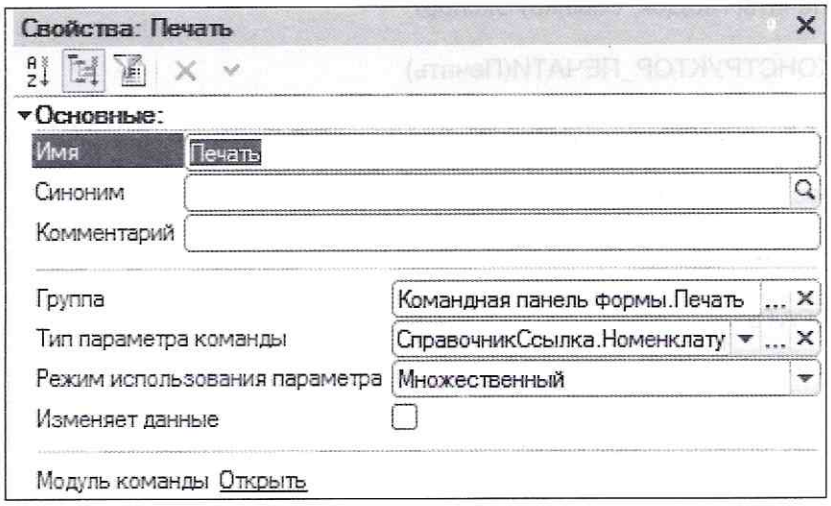


В левой части диалоговой формы приведены все доступные нам реквизиты справочника, в правой - выбранные для печати.

На последнем этапе указываем, что создаваемая конструктором команда будет входить в ранее созданную нами группу "Печать".



В результате работы конструктора у объекта конфигурации была создана команда. Ее свойства приведены на рисунке ниже:



В модуле команды определен следующий код:

```
&НаКлиенте

Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

    //{{_КОНСТРУКТОР_ПЕЧАТИ(Печать)

    ТабДок = Новый ТабличныйДокумент;

    Печать(ТабДок, ПараметрКоманды);

    ТабДок.ОтображатьСетку = Ложь;

    ТабДок.Защита = Ложь;

    ТабДок.ТолькоПросмотр = Ложь;

    ТабДок.ОтображатьЗаголовки = Ложь;

    ТабДок.Показать();

    //}}

КонецПроцедуры


&НаСервере

Процедура Печать(ТабДок, ПараметрКоманды)

    Справочники.Номенклатура.Печать(ТабДок, ПараметрКоманды);

КонецПроцедуры
```

В модуле менеджера справочника:

```
Процедура Печать(ТабДок, Ссылка) Экспорт
	//{{_КОНСТРУКТОР_ПЕЧАТИ(Печать)
	Макет = Справочники.Номенклатура.ПолучитьМакет("Печать");
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ
	|	Номенклатура.Код,
	|	Номенклатура.Наименование,
	|	Номенклатура.ОснЕдиницаИзмерения,
	|	Номенклатура.ЦенаПродажи
	|ИЗ
	|	Справочник.Номенклатура КАК Номенклатура
	|ГДЕ
	|	Номенклатура.Ссылка В (&Ссылка)";                    //1
	Запрос.Параметры.Вставить("Ссылка", Ссылка);             //2
	Выборка = Запрос.Выполнить().Выбрать();

	ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
	Шапка = Макет.ПолучитьОбласть("Шапка");
	ТабДок.Очистить();

	ВставлятьРазделительСтраниц = Ложь;
	Пока Выборка.Следующий() Цикл
		Если ВставлятьРазделительСтраниц Тогда
			ТабДок.ВывестиГоризонтальныйРазделительСтраниц();
		КонецЕсли;

		ТабДок.Вывести(ОбластьЗаголовок);
		Шапка.Параметры.Заполнить(Выборка);
		ТабДок.Вывести(Шапка, Выборка.Уровень());
```

```
            ВставлятьРазделительСтраниц = Истина;

        КонецЦикла;

        //}}

КонецПроцедуры
```

Проверьте механизм на практике.

Обратите внимание на то, что в запросе, выполняемом к данным, в качестве значения условия используется "внешнее" (по отношению к запросу) значение.

//1 "внешнее" значение определяется как параметр запроса.

//2 параметру запроса присваивается значение.

В принципе, на этом этапе пришло время познакомиться с классификацией команд.

Команды подразделяются на:

- Независимые команды
- Параметризуемые глобальные команды
- Команды формы

По способу создания они делятся на:

- Стандартные
- Созданные в конфигурации

По выполняемому действию:

- Навигационные
- Команды действия

Для независимых команд существенную роль играет отнесение ее к той или иной подсистеме. Это будет определять, в каких разделах/подразделах появится команда в интерфейсе управляемого приложения.

Для параметризованных команд очень важную роль играет свойство "Тип параметра".

Стандартные команды предоставляются системой для ряда объектов конфигурации, их представление формируется только на основе свойств соответствующего объекта метаданных. Изменить поведение стандартных команд "штатными" средствами нельзя.

Команды, созданные в контексте каких-либо объектов конфигурации, относятся к тем же подсистемам, к которым отнесен сам объект конфигурации.

Практикум №4

Создайте следующие справочники:

- ***"Склады". Справочник без иерархии, без подчинения. Реквизитов и табличных частей нет.***
- ***"Контрагенты". Справочник иерархический (иерархия групп и элементов), без подчинения. Дополнительный реквизит "НаименованиеПолное", тип "Строка" 300 символов.***

- ***"КонтактныеЛица". Справочник без иерархии, подчинен справочнику "Контрагенты". Дополнительный реквизит "Телефон", тип "Строка" 15 символов.***
- ***"Должности". Справочник без иерархии, без подчинения. Реквизитов и табличных частей нет. В нем определены 3 предопределенных элемента с именами "Бухгалтер", "ГлавныйБухгалтер", "Кассир".***

4.6. Документы

Документ – одно из основных понятий системы "1С:Предприятие". При помощи документов организуется ввод в систему первичной информации о совершаемых хозяйственных операциях.

В большинстве своем документы, которые создаются в процессе настройки конфигурации, являются электронными аналогами стандартных бумажных документов, однако, использование этого типа данных может выходить далеко за рамки простой фиксации информации о хозяйственных операциях.

Дата и время – наиболее важные характеристики документов, так как позволяют устанавливать строгую временную последовательность совершения операций. У документа может быть любое количество табличных частей.

4.6.1. Создание документов

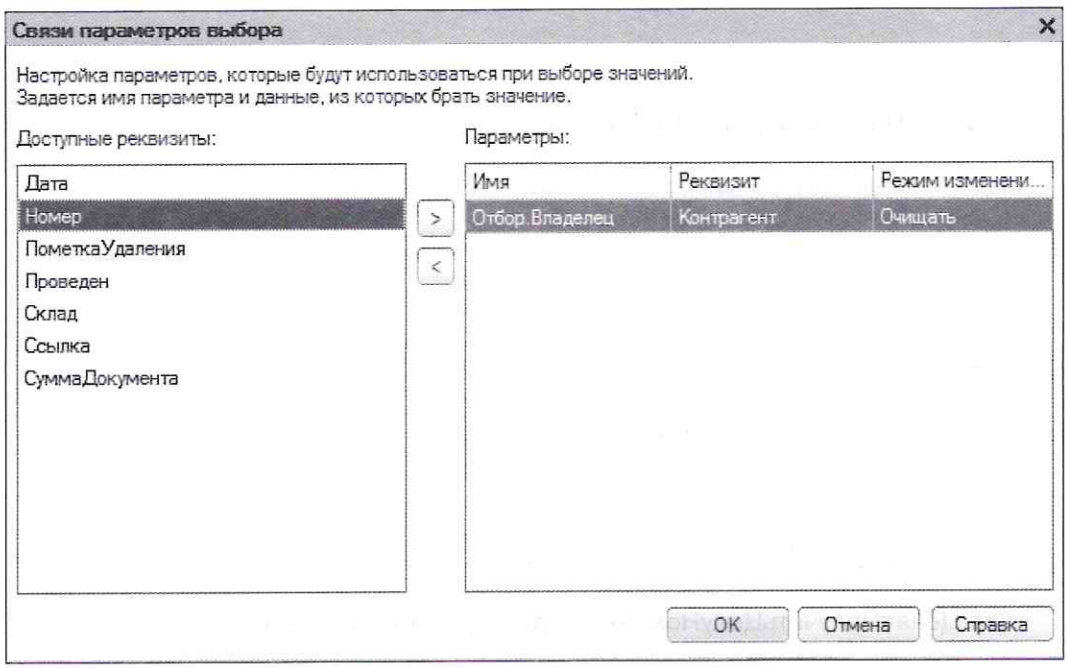
В рамках нашей задачи создадим документ "ПоступлениеТоваров". Он будет являться электронным аналогом расходной накладной поставщика. Состав реквизитов следующий:

- "Контрагент" (тип <СправочникСсылка.Контрагенты>)
- "КонтактноеЛицо" (тип <СправочникСсылка.КонтактныеЛица>)
- "Сотрудник" (тип <СправочникСсылка.ФизическиеЛица>)
- "Склад" (тип <СправочникСсылка.Склады>)
- "СуммаДокумента" (тип <Число> длина 15, точность 2)

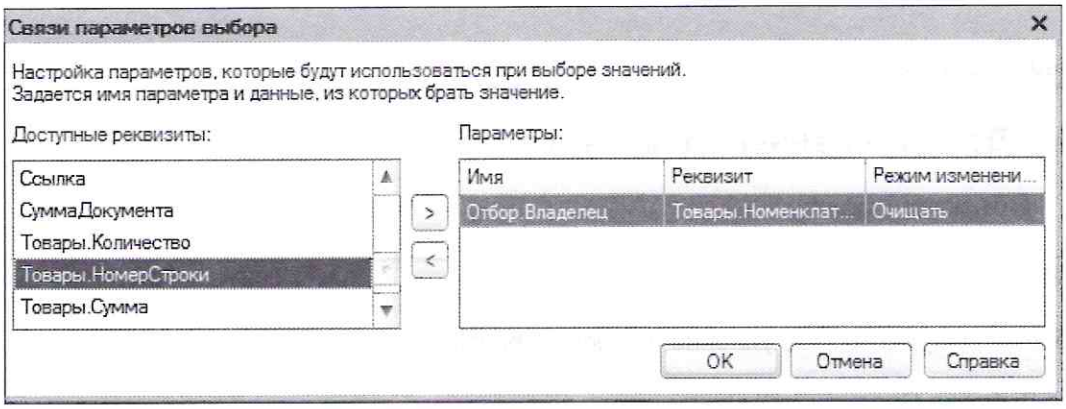
Определим одну табличную часть "Товары", ее состав:

- Номенклатура (тип <СправочникСсылка.Номенклатура>)
- Количество (тип <Число> длина 10, точность 0)
- Цена (тип <Число> длина 10, точность 2)
- Сумма (тип <Число> длина 10, точность 2)
- Серия (тип <СправочникСсылка.Серии>)

Для реквизита документа "КонтактноеЛицо" настроим связи параметров выбора.



Такую же связь настроим для реквизита табличной части "Серии".



Создайте форму документа, форму списка документа.

В форме документа элементы управления "Номер" и "Дата" определите в одной горизонтальной группе. Также необходимо настроить обработчики событий.

```
&НаКлиенте

Процедура КоличествоПриИзменении(Элемент)


    Стр = Элементы.Товары.ТекущиеДанные;

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


КонецПроцедуры
```

```
&НаСервереБезКонтекста

Функция ПолучитьЦенуНоменклатуры(Номенклатура)

    Возврат Номенклатура.ЦенаПокупки;

КонецФункции

&НаКлиенте

Процедура НоменклатураПриИзменении(Элемент)

    Стр = Элементы.Товары.ТекущиеДанные;

    Стр.Цена=ПолучитьЦенуНоменклатуры(Стр.Номенклатура);

    КоличествоПриИзменении(Элемент);

КонецПроцедуры
```

4.6.2. Доступ к данным документа

Объектная модель.

```
// выборка документов

Выборка=Документы.ПоступлениеТоваров.Выбрать();

Пока Выборка.Следующий() Цикл

    ТекКонтрагент=Выборка.Контрагент;

    // перебор табличной части документа

    Для Каждого ТекСтрока Из Выборка.Товары Цикл

        ТекНоменклатура=ТекСтрока.Номенклатура;

    КонецЦикла;

КонецЦикла;
```

Табличная модель.

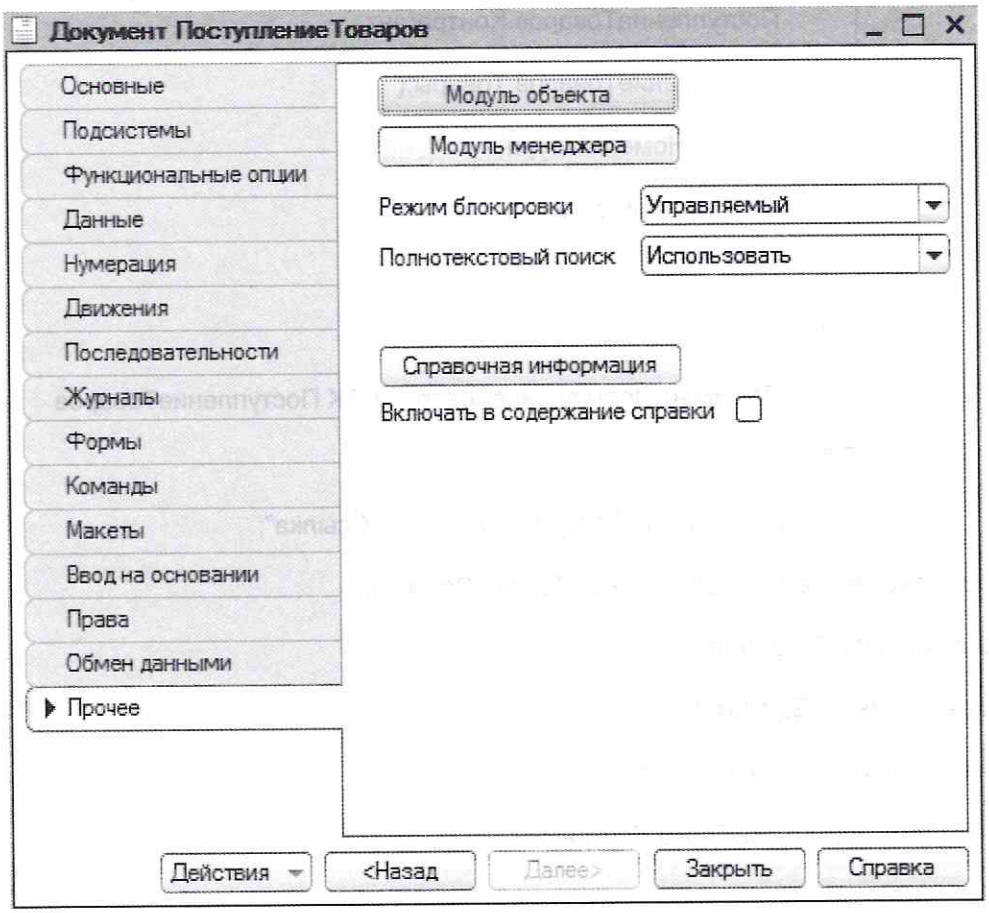
```
Запрос=Новый Запрос;
Запрос.Текст= "ВЫБРАТЬ
              |    ПоступлениеТоваров.Контрагент,
              |    ПоступлениеТоваров.Товары.(
              |        Номенклатура,
              |        Количество
              |    )
              |ИЗ
              |    Документ.ПоступлениеТоваров КАК ПоступлениеТоваров
              |ГДЕ
              |    ПоступлениеТоваров.Ссылка = &Ссылка";
Запрос.УстановитьПараметр("Ссылка",Объект.Ссылка);
Результат=Запрос.Выполнить();
Выборка=Результат.Выбрать();
Пока Выборка.Следующий()  Цикл
    ВыбКонтрагент=Выборка.Контрагент;
    //.....
    ВыборкаТЧ=Выборка.Товары.Выбрать();
    Пока ВыборкаТЧ.Следующий() Цикл
        ТекНоменклатура=ВыборкаТЧ.Номенклатура;
        // .....
    КонецЦикла;
КонецЦикла;
```

Практикум №5

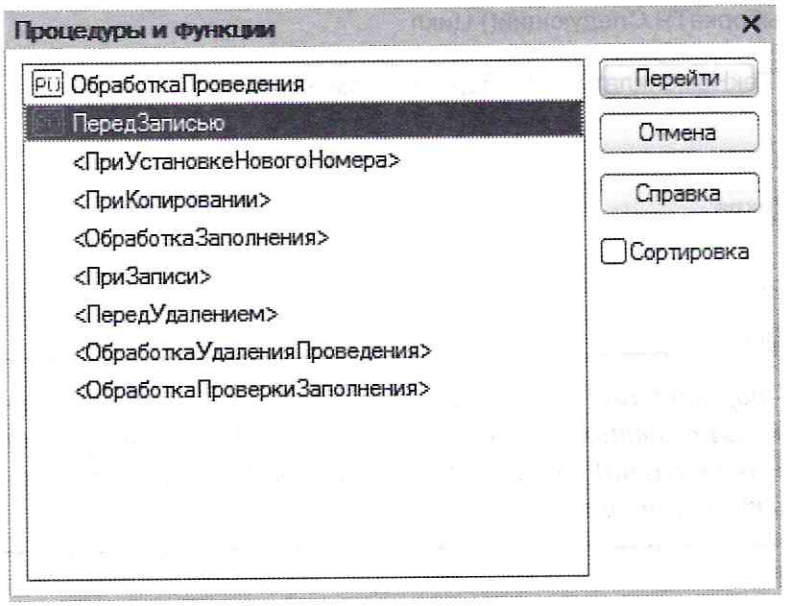
Определите обработчик события, который позволил бы после выбора контактного лица автоматически устанавливать в документе значение контрагента (владельца). При реализации этого обработчика необходимо использовать табличную модель.

4.6.3. Модуль объекта

Для открытия модуля объекта можно использовать либо одноименную команду контекстного меню объекта конфигурации, либо кнопку, расположенную на закладке "Прочее" формы объекта конфигурации.



В модуле объекта можно определить ряд обработчиков событий, исполняемых на сервере (контекст модуля объекта не доступен на клиенте).



Определим в данном модуле обработчик события "ПередЗаписью". Текст обработчика следующий:

```
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

    СуммаДокумента=Товары.Итог("Сумма");

КонецПроцедуры
```

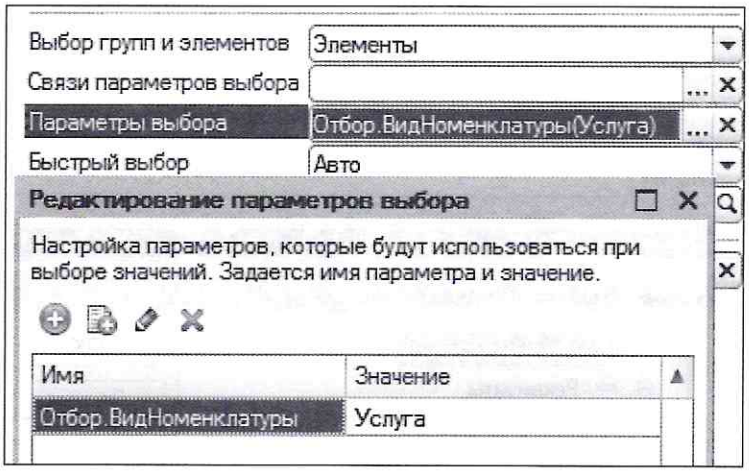
4.6.4. Создание объектов копированием

Бывают ситуации, когда создать объект проще копированием, т.е. скопировать объект и "доработать" копию. Копировать объект можно, используя кнопки работы с буфером обмена, используя команды контекстного меню или просто "перетаскивая" объект.

Создайте документ "ПродажаТоваров" путем копирования и изменения "ПоступленияТоваров". У создаваемого документа состав реквизитов шапки такой же, как и у исходного, но на одну табличную часть больше. Состав реквизитов табличной части "Услуги":

- **Номенклатура** (тип <СправочникСсылка.Номенклатура>)
- **Количество** (тип <Число> длина 10, точность 0)
- **Цена** (тип < Число > длина 10, точность 2)
- **Сумма** (тип < Число > длина 10, точность 2)

Измените необходимые обработчики событий. Для реквизита "Номенклатура" табличной части "Услуги" настройте свойство "Параметры выбора".

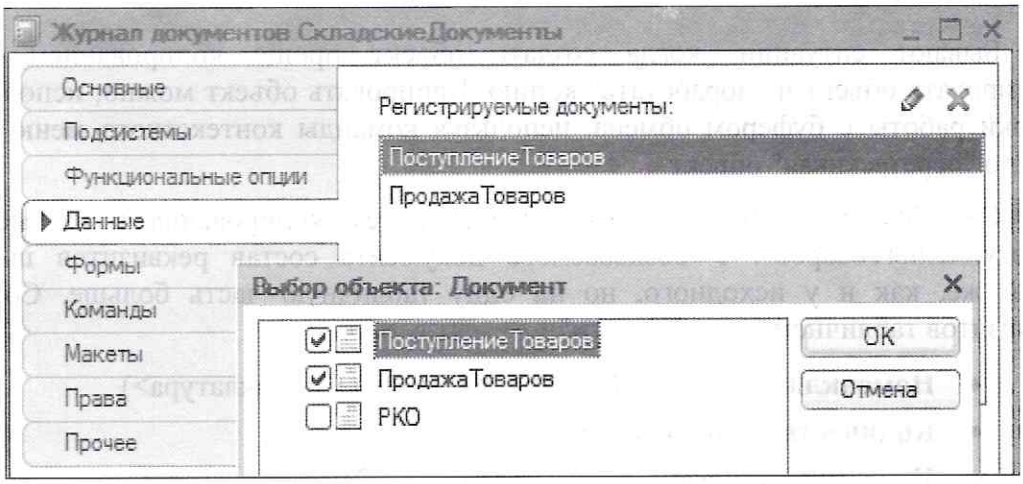


Реализуйте возможность получения печатной формы документа.

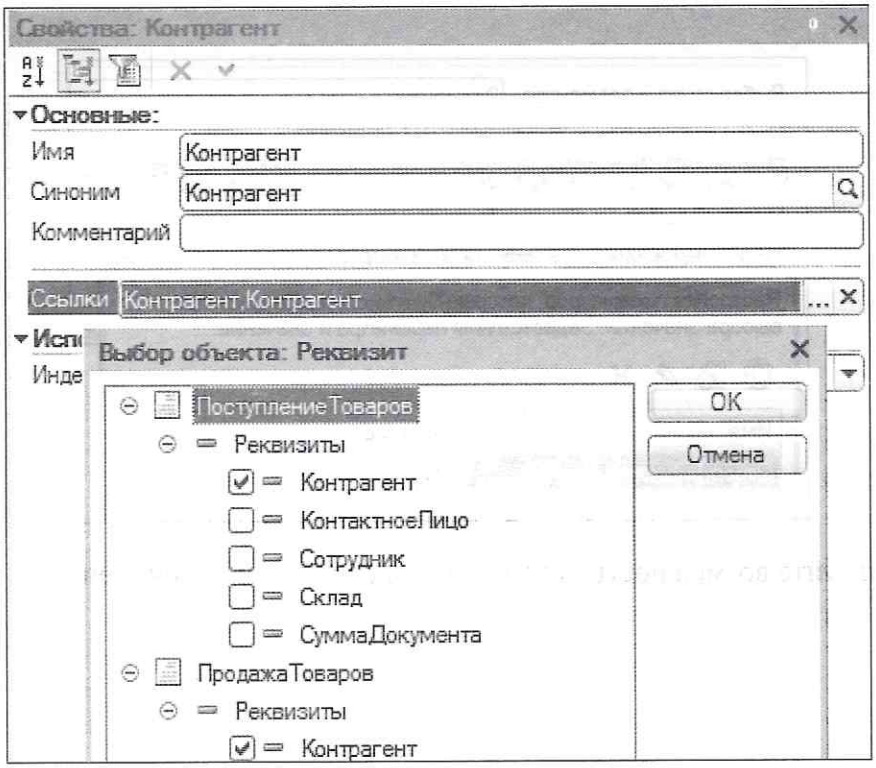
4.7. Журналы документов

До этого момента для работы с уже выписанными документами использовались их формы списка. Эти формы списка не позволяли просматривать (в одном списке) документы разных видов. Если же в этом есть необходимость, то здесь могут помочь журналы.

Создайте журнал "СкладскиеДокументы". В нем должны регистрироваться документы "Поступление товаров" и "Продажа товаров".



В журнале может быть любое (в разумных пределах) количество граф. При создании графы необходимо от каждого вида документа выбрать по реквизиту (только по одному, но можно разных типов), который будет в ней отображаться.



Создайте три графы: "Контрагент", "Склад", "СуммаДокумента".

Проверьте работу объекта на практике.

4.8. Регистры сведений

Основная задача регистра сведений – хранить существенную для прикладной задачи информацию, состав которой развернут по определенной комбинации измерений и, при необходимости, развернут во времени. Эта информация хранится в регистре в виде записей. На запись нельзя сделать ссылку из информационной базы. В регистре может быть только одна запись с определенной комбинацией измерений и периодом.

Регистры сведений, информация в которых развернута во времени, называют периодическими.

Регистр может характеризоваться выбранным режимом записи:

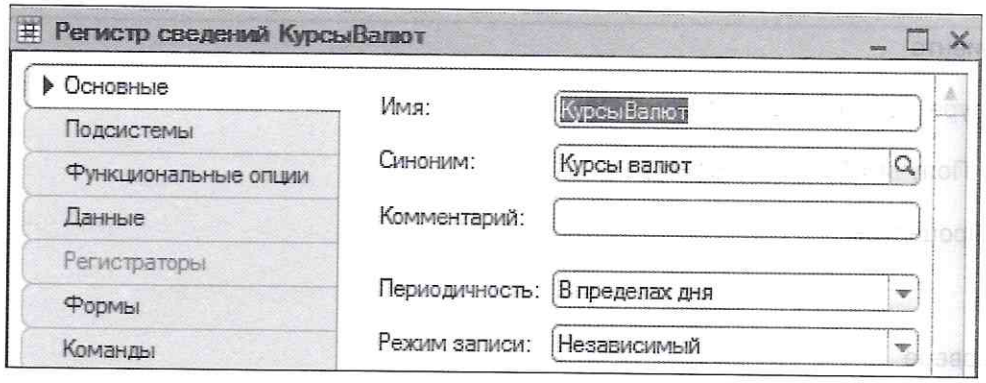
- Независимый
- Подчинение регистратору

При выборе режима "Подчинение регистратору" запись жестко подчиняется документу-регистратору. При другом режиме ("Независимый") записи "подчиняются" так называемым "ведущим" измерениям. При проектировании регистра необходимо обращать внимание на порядок следования измерений.

4.8.1. Создание регистра сведений

Создайте справочник "Валюты", определите реквизит "НаименованиеПолное".

После этого создайте регистр сведений "КурсыВалют".



У него определите одно измерение "Валюта" (ведущее, тип <СправочникСсылка.Валюта>).

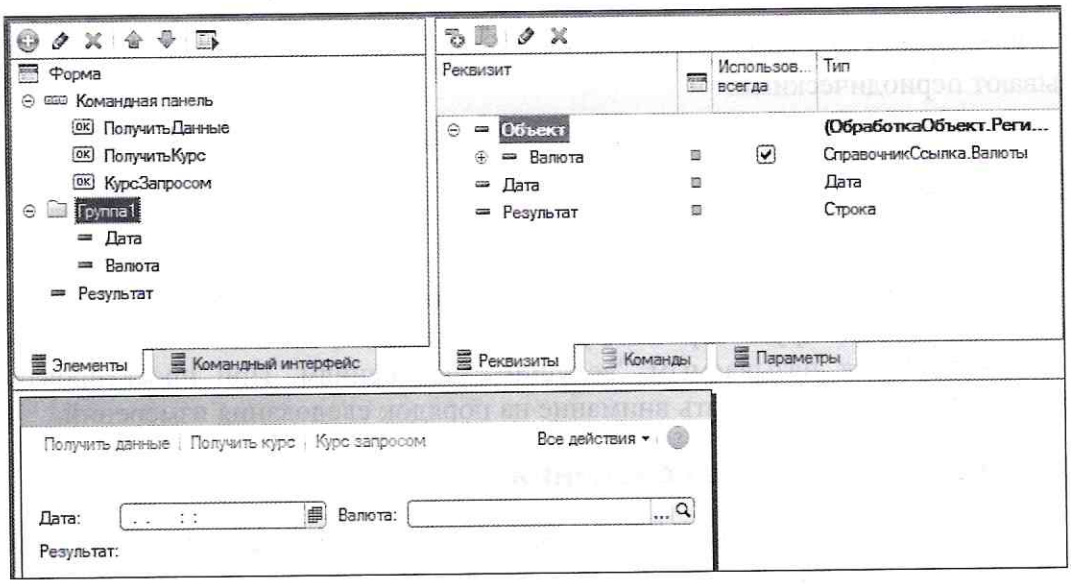
Ресурс "Курс" (тип <Число>), ресурс "Кратность" (тип <Число>).

После создания регистра в режиме "1С:Предприятие" попробуйте ввести записи с одинаковым периодом и валютой.

4.8.2. Работа с данными регистра

Использование объектной модели

Определим обработку "РегистрыСведений": определим реквизит обработки "Валюта" (тип <СправочникСсылка.Валюты>), реквизиты основной формы "Дата" (тип <Дата>), "Результат" (тип <Строка>). Разместим соответствующие реквизитам элементы управления в форме (как показано на рисунке).



Определим две команды и опишем обработчики события к ним:

```
&НаКлиенте
Процедура ПолучитьДанные(Команда)
        ПолучитьДанныеСервер();
КонецПроцедуры

&НаСервере
Процедура ПолучитьДанныеСервер()
        Отбор=Новый Структура("Валюта",Объект.Валюта);
        ВыборкаКурсов=РегистрыСведений.КурсыВалют.Выбрать(Дата,,Отбор);
        СтрРезультата="";
        Пока ВыборкаКурсов.Следующий() Цикл
                СтрРезультата=СтрРезультата+Строка(ВыборкаКурсов.Курс)+"; ";
        КонецЦикла;
        Результат=СтрРезультата;
КонецПроцедуры
```

```
&НаКлиенте
Процедура ПолучитьКурс(Команда)
        ПолучитьКурсНаСервере();
КонецПроцедуры
&НаСервере
Процедура ПолучитьКурсНаСервере()
        Отбор=Новый Структура("Валюта",Объект.Валюта);
        Запись=РегистрыСведений.КурсыВалют.ПолучитьПоследнее(Дата,Отбор);
        Результат=Строка(Запись.Курс)+":"+Строка(Запись.Кратность);
КонецПроцедуры
```

Табличная модель

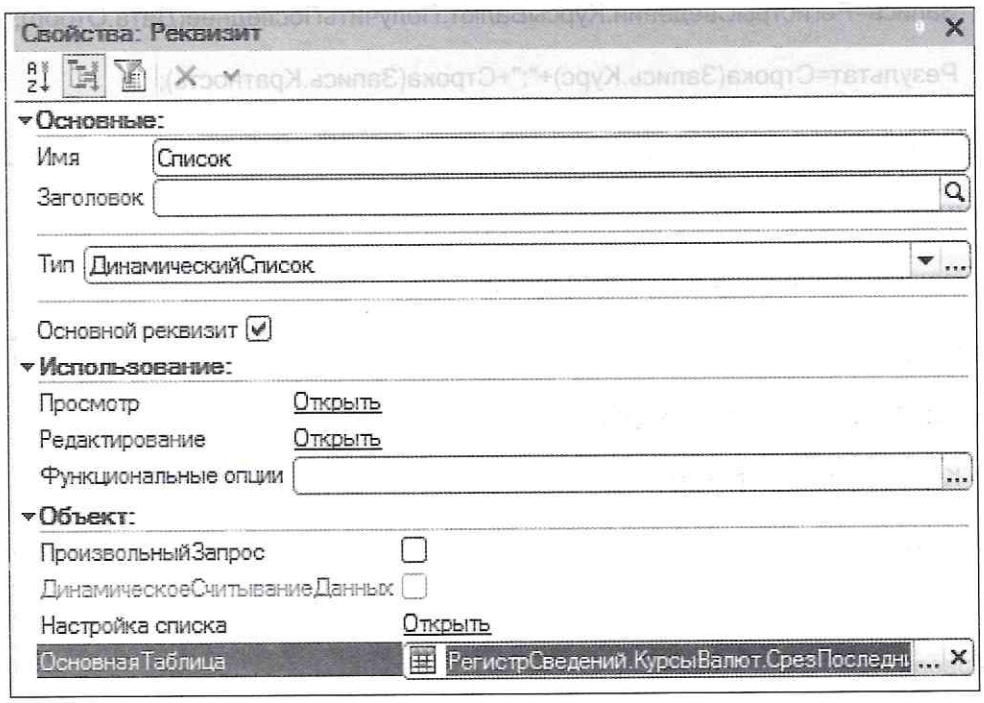
Создадим еще одну команду формы:

```
&НаКлиенте
Процедура КурсЗапросом(Команда)
        КурсЗапросомСервер()
КонецПроцедуры

&НаСервере
Процедура КурсЗапросомСервер()
        Запрос=Новый Запрос;
        Запрос.Текст="ВЫБРАТЬ
                |       КурсыВалютСрезПоследних.Период,
                |       КурсыВалютСрезПоследних.Курс,
                |       КурсыВалютСрезПоследних.Кратность
                |ИЗ
                |       РегистрСведений.КурсыВалют.СрезПоследних(&Дата, Валюта =
&Валюта) КАК КурсыВалютСрезПоследних";
        Запрос.УстановитьПараметр("Дата",Дата);
        Запрос.УстановитьПараметр("Валюта",Объект.Валюта);
        РезультатЗапроса=Запрос.Выполнить();
        Выборка=РезультатЗапроса.Выбрать();
```

```
    Если Выборка.Следующий() Тогда

        Результат=Строка(Выборка.Курс)+":"+Строка(Выборка.Кратность);

    КонецЕсли;

КонецПроцедуры
```

4.8.3. Форма списка регистра

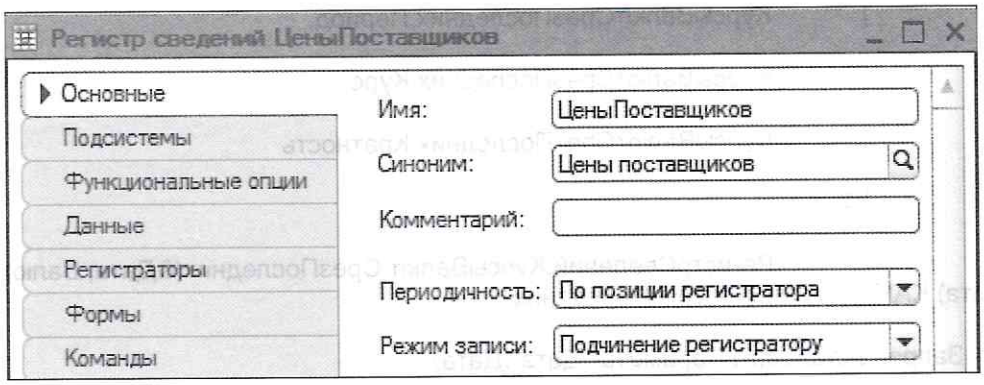
Создадим форму списка регистра, сделайте ее основной формой. В свойствах основного реквизита формы изменим основную таблицу (на "РегистрСведений.КурсыВалют.СрезПоследних").



Не забудьте эту форму отразить в интерфейсе.

4.8.4. Режим записи "Подчинение регистратору"

Создадим регистр сведений "ЦеныПоставщиков" (как на рисунке).

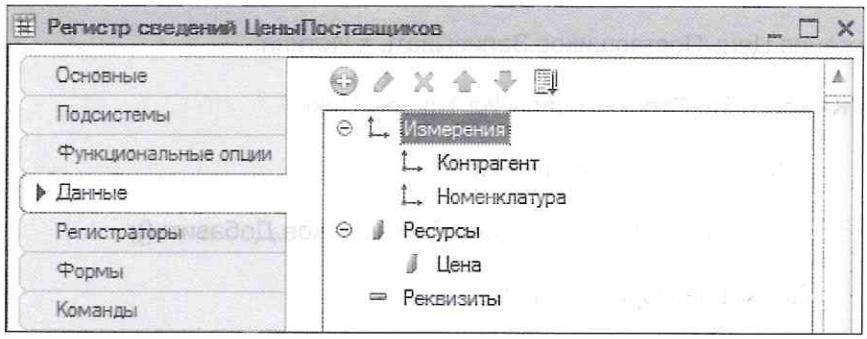


Измерения регистра:

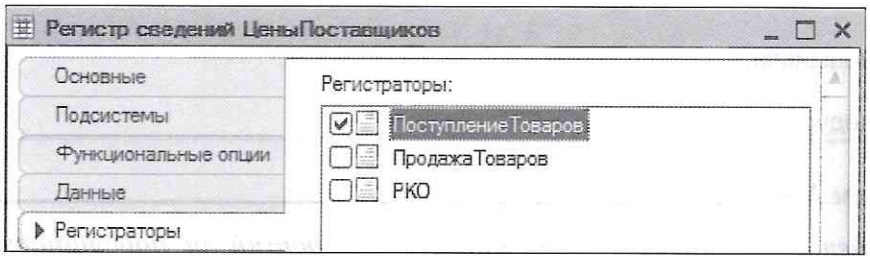
- Контрагент (тип <СправочникСсылка.Контрагенты>)
- Номенклатура (тип <СправочникСсылка.Номенклатура>)

Ресурс:

- Цена (тип <число>)



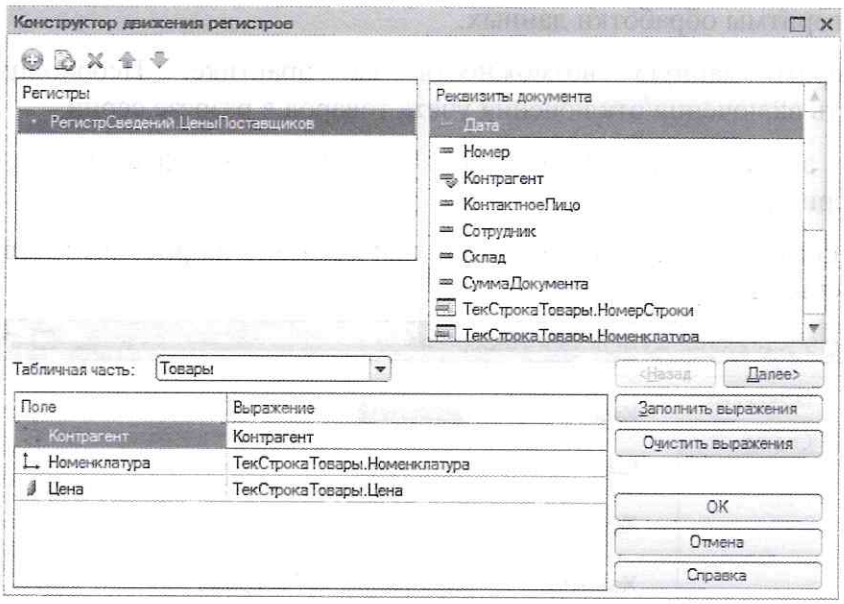
Документ имеет режим записи "Подчинение регистратору", в соответствии с этим стала доступной закладка "Регистраторы". В качестве регистратора нужно отметить документ "Поступление товаров".



Штатным режимом записи данных в такой регистр является запись при проведении документа (настраивается в обработчике события документа "Обработка проведения"). Воспользуемся возможностью создания этого обработчика с помощью конструктора.

Откроем форму объекта конфигурации "Документ.ПоступлениеТоваров". На закладке "Движения" удостоверимся, что свойство "Проведение" установлено в значение "Разрешить". Вызов конструктора осуществляется по нажатию кнопки "Конструктор движений" на этой закладке.

Настройте открывшуюся форму в соответствии с приведенным ниже рисунком (не забудьте выбрать табличную часть):



В результате работы конструктора получится код следующего вида:

```
Процедура ОбработкаПроведения(Отказ, Режим)

    Движения.ЦеныПоставщиков.Записывать = Истина;

    Для Каждого ТекСтрокаТовары Из Товары Цикл

        // регистр ЦеныПоставщиков

        Движение = Движения.ЦеныПоставщиков.Добавить();

        Движение.Период = Дата;

        Движение.Контрагент = Контрагент;

        Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;

        Движение.Цена = ТекСтрокаТовары.Цена;

    КонецЦикла;

КонецПроцедуры
```

Практикум №6

Переопределите обработчик события отвечающий за подстановку цены в документе "Поступление товаров". Цена должна выбираться из регистра сведений "Цены поставщиков" (на дату документа). Для получения цены необходимо использовать табличную модель.

4.9. Функциональные опции

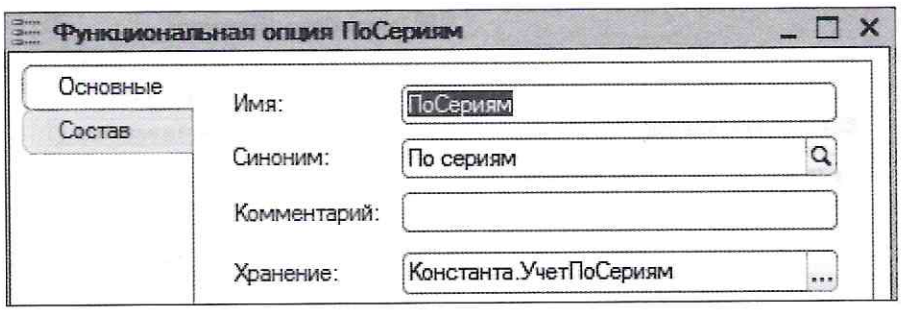
Функциональные опции позволяют разработчику описать возможности конфигурации, которые можно оперативно включать или выключать на этапе внедрения и/или в процессе работы системы.

Функциональные опции могут влиять как на пользовательский интерфейс, так и на алгоритмы обработки данных.

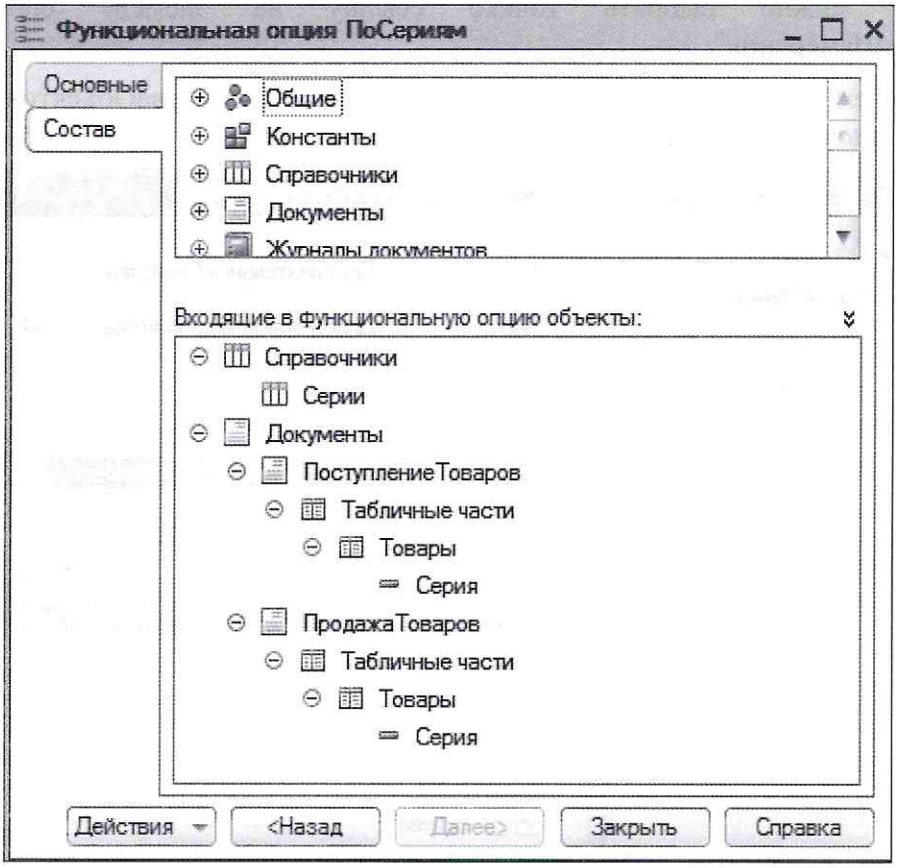
Проверим данную возможность на практике. Необходимо иметь возможность включения/отключения учета товаров в разрезе серий.

Свое значение функциональная опция будет хранить в константе "УчетПоСериям".

Создадим функциональную опцию как показано на рисунке (ветвь "Общие" дерева объектов конфигурации):



Следующим действием необходимо указать, какие объекты конфигурации относятся к созданной функциональной опции. Сделаем это так, как показано на рисунке:



Проверьте механизм на практике.

Следует отметить, что функциональные опции могут хранить свои значения не только в константах, но и в справочниках, и в регистрах сведений. Тогда для определения значения функциональной опции используются "Параметры функциональных опций".

Практикум №7

Определите в системе возможность включения/отключения ведения складского учета в разрезе складов. Для хранения значения этой опции используйте константу "Учет по складам".

4.10. Планы видов характеристик

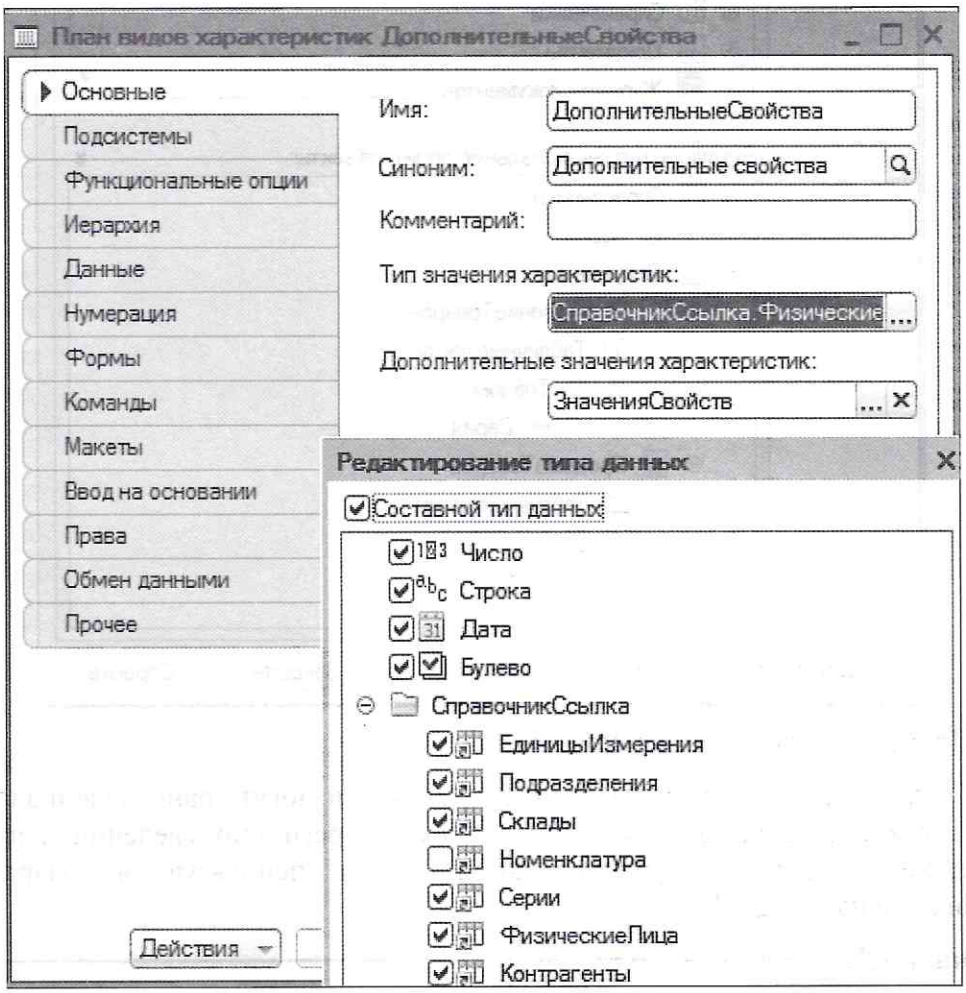
Данный вид объекта имеет много схожих характеристик со справочником. Самое существенное его отличие (от справочника) в том, что для каждого элемента существует такое свойство как "Тип".

С помощью данного объекта попробуем для справочника "Номенклатура" реализовать механизм внесения любого количества свойств различных типов.

Следует отметить, что подобный механизм можно реализовать и через справочник, но в том случае, если нужно контролировать типы свойств, наиболее просто его реализовать через план видов характеристик (в записи плана видов характеристик будет находиться не только "наименование" свойства, но и его тип). Используя возможность хранения "типа свойства" можно реализовывать

всевозможные механизмы контроля. Например, если указан для свойства "Фасовка" (элемент плана видов характеристик) тип "СправочникСсылка.ЕдиницыИзмерения", то в качестве значений для данного свойства можно выбрать только ссылку на элемент справочника "ЕдиницыИзмерения".

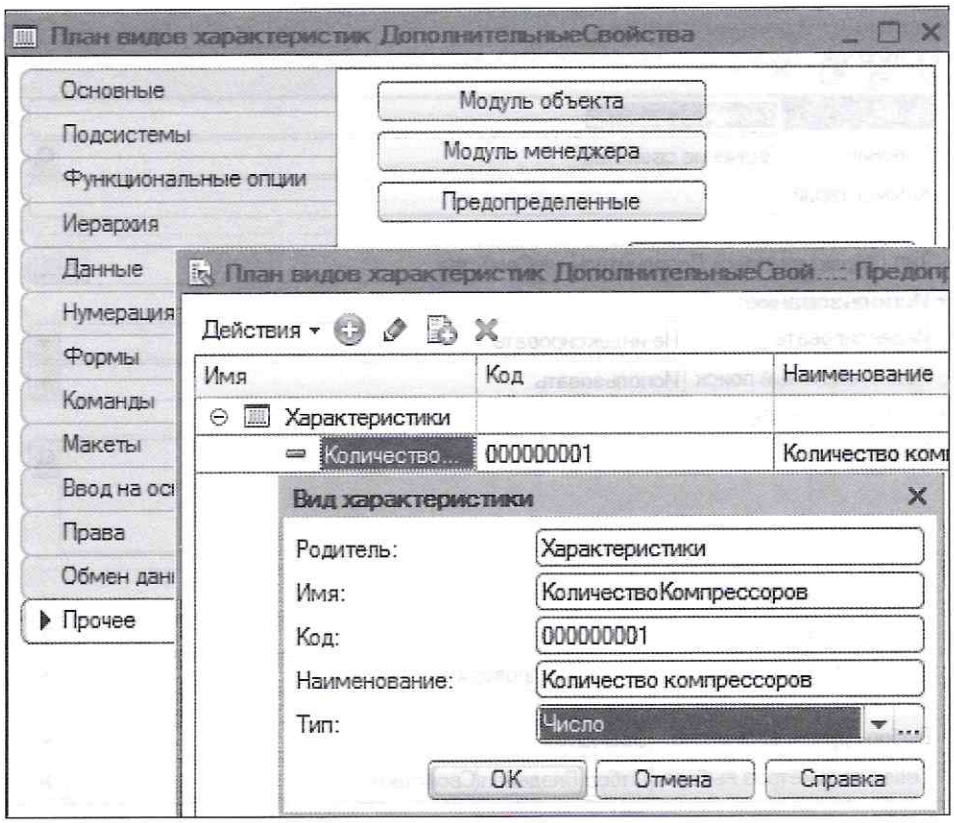
Начнем с того, что необходимо создать объект "СвойстваОбъектов" в ветви "Планы видов характеристик" дерева метаданных.



В типе значения характеристик выбираем типы, которые будут использоваться при определении вида характеристики (назовем свойство – характеристикой, свойство товара плюс проверочная информация о типе значения свойства – видом характеристики).

После создания плана видов характеристик необходимо определить справочник "ЗначенияСвойств" и подчинить его объекту "СвойстваОбъектов". Этот справочник также нужно отнести в перечень разрешенных для использования типов. После этого его нужно выбрать в свойство "Дополнительные значения характеристик".

На закладке "Прочие" можно определить перечень предопределенных видов характеристик.



Для хранения информации о том, у какой номенклатуры какое свойство есть, и в каком оно значении, лучше использовать регистр сведений.

Создадим регистр сведений "СвойстваНоменклатуры" (независимый, не периодический, редактируется в диалоге).

Состав измерений:

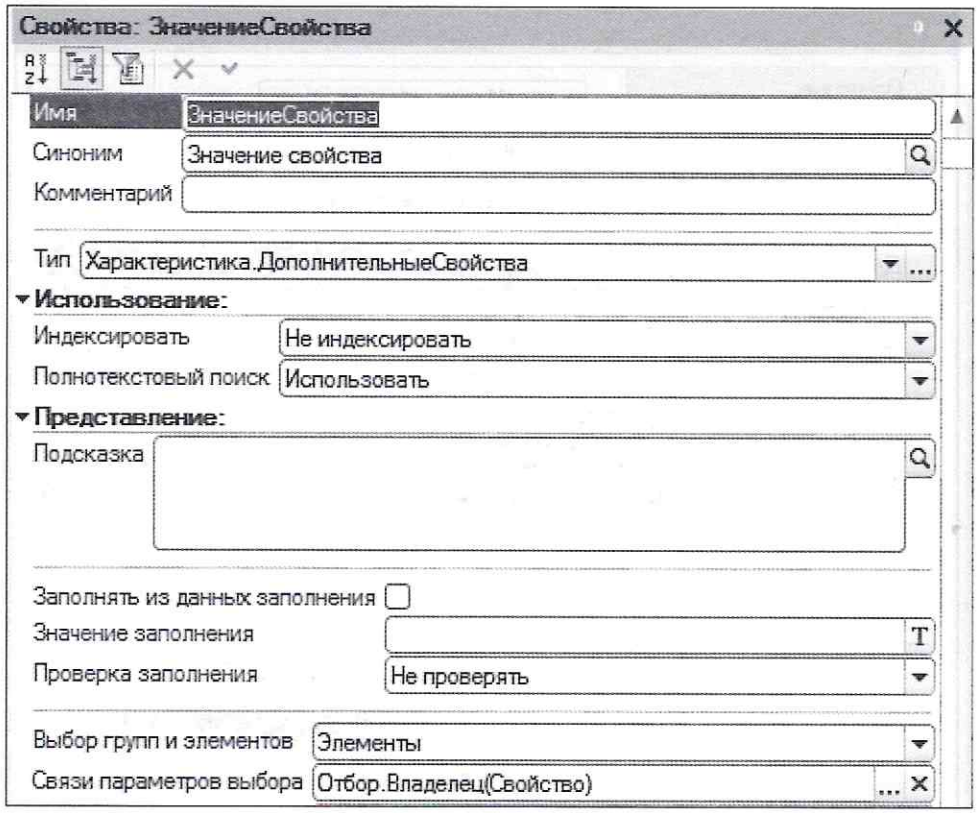
- Объект (тип <СправочникСсылка.Номенклатура>)
- Свойство (тип <ПланВидовХарактеристикСсылка.СвойстваОбъектов>)

Ресурс:

- ЗначениеСвойства (тип <Характеристика.СвойстваОбъектов>)

После создания вышеуказанных объектов в системе был реализован механизм определения любого количества свойств товаров (с проверочной информацией о типах значения каждого свойства), хранения значений свойств товаров. Но до сих пор не решена задача использования проверочной информации о типах.

Для решения этой части задачи необходимо у ресурса настроить свойство "Связи параметров выбора" (в соответствии с приведенным рисунком).



Проверьте созданный механизм на практике.

4.11. Учетные объекты

В "1С:Предприятие" принята следующая модель: деятельность организации разбивается на набор операций (в принципе, об этой модели говорилось и ранее). Для регистрации факта осуществления хозяйственной операции создается документ. Т.е. документы содержат первичную информацию о совершенной хозяйственной операции. Фактом принятия данных документа к учету является проведение документа. Таким образом получается, что данные, принятые к учету, находятся в регистрах.

Все регистры имеют ряд схожих черт:

- Записи регистров не имеют объектной природы.
- Одной из важных характеристик любого регистра является состав его измерений, ресурсов, реквизитов.
- У каждого регистра есть некие "специфические умения".

Регистры накопления используются в системе для накопления информации о наличии и движении средств – товарных, денежных и других. Они позволяют очень быстро получать информацию об остатках, оборотах, остатках с оборотами на интересующие даты, за указанные периоды.

Различают два вида регистров накопления:

- Регистры остатков
- Регистры оборотов

На регистрах накопления решают задачи оперативного учета.

Ведение бухгалтерского учета в системе "1С:Предприятие" обеспечивают объекты конфигурации "Планы счетов" и "Регистры бухгалтерии". Средства системы позволяют организовать учет по нескольким планам счетов, при этом для каждого плана счетов может строиться произвольная иерархия субсчетов большой вложенности.

Планом счетов называется совокупность синтетических счетов, предназначенных для группировки информации о хозяйственной деятельности предприятия. Информация, накапливаемая на таких синтетических счетах, позволяет получить полную картину состояния средств предприятия.

Дополнительные аналитические разрезы определяются в плане видов характеристик (так называемые "Субконто"). Максимальное количество аналитических разрезов определяется в самом плане счетов.

Для отражения в бухгалтерском учете информации о хозяйственных операциях в системе "1С:Предприятие" используются регистры бухгалтерии.

В конфигурации может быть определено несколько (или один) планов счетов, несколько регистров бухгалтерии. Но каждый регистр бухгалтерии связан только с одним планом счетов.

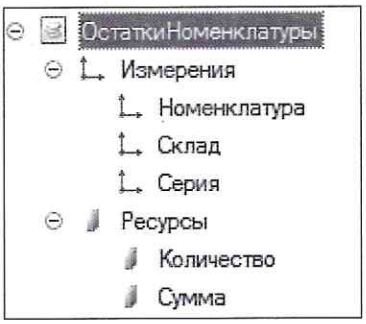
Следует отметить, что регистр бухгалтерии может не иметь измерений, реквизитов, но обязательно у него должен быть хоть один ресурс (попросту говоря, "сумма проводки").

Для решения расчетных задач (расчет заработной платы) используются такие объекты, как планы видов расчета и регистры расчета.

Планы видов расчета содержат перечень видов расчета.

Регистр расчета – это объект конфигурации, который позволяет организовать учет результатов вычислений, осуществляемых с некоторой периодичностью, тесно связанных друг с другом по некоторым правилам и взаимно влияющих друг на друга в пределах определенного периода.

Создайте регистр накопления "ОстаткиНоменклатуры". Вид регистра "Остатки". Состав измерений и ресурсов приведен на рисунке:

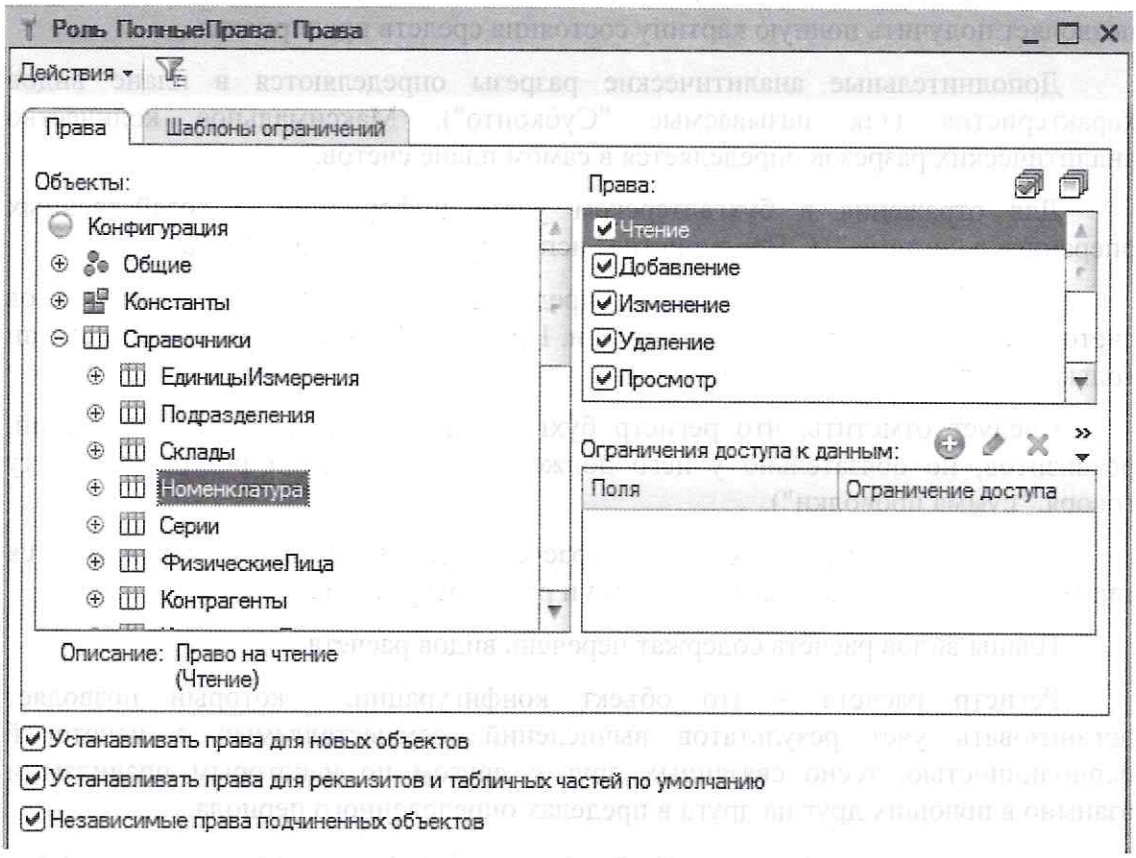


Создайте регистр накопления "Продажи". Вид регистра "Обороты". Состав измерений и ресурсов приведен на рисунке:

Таким же образом, как и для регистра сведений, настройте движения документов "Поступление товаров" и "Продажа товаров". Проверьте работу механизмов на практике.

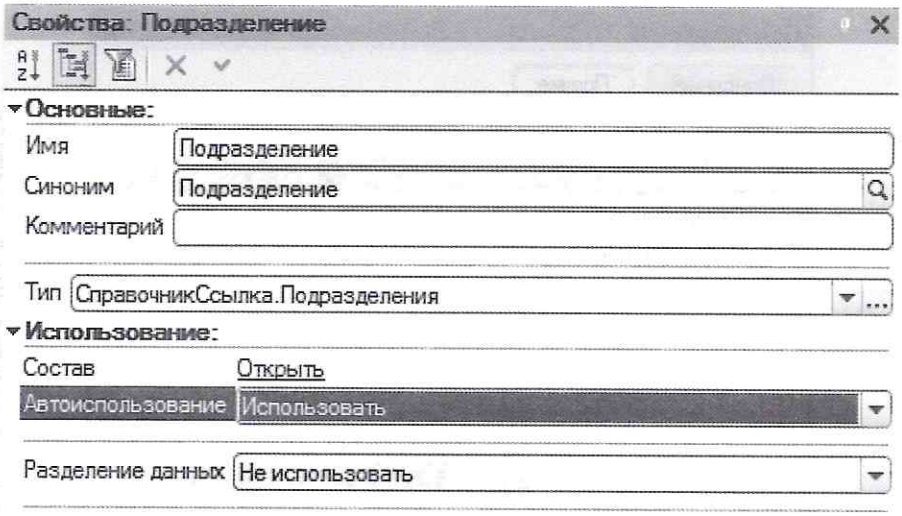
4.12. Элементы администрирования

К этому моменту уже создано некоторое количество объектов конфигурации. Можно вернуться и еще раз взглянуть на роли.



Права делятся на программные и интерактивные. С помощью ограничения доступа к данным можно определить доступ только к определенным записям таблицы (например, организовать доступ только к документам какой-либо группы контрагентов).

С точки зрения разделения доступа также стоит упомянуть "механизм разделения данных". В основе данного механизма лежат "общие реквизиты". Находятся эти объекты в ветви "Общие" конфигурации. Как следует из названия, с их помощью можно создавать реквизиты общие для всех объектов (один раз определили и реквизит с таким именем и типом "появится" у всех нужных объектов).



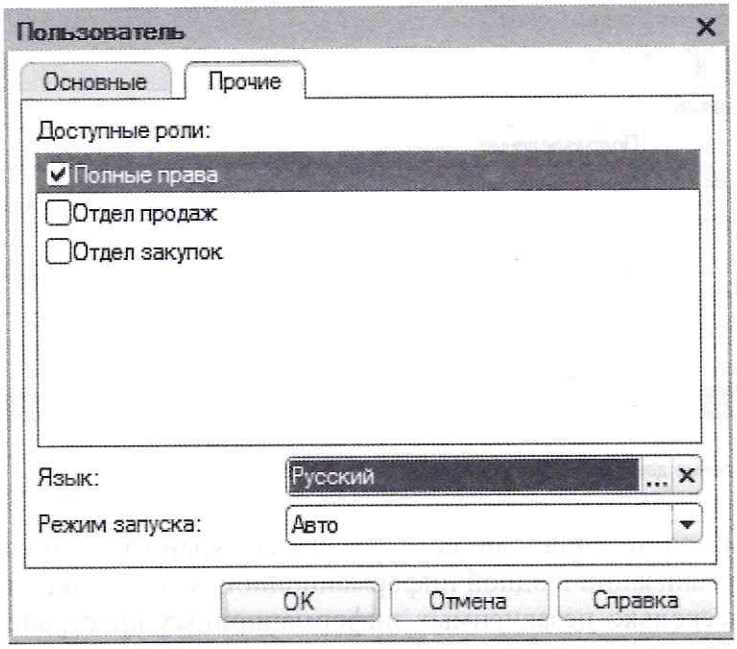
Механизм разделения данных позволяет хранить данные нескольких независимых организаций в одной информационной базе. Т.е. в одной базе можно организовать несколько независимых информационных пространств. Они могут быть полностью изолированными или пересекаться по нужным объектам (например может быть общий справочник валют, регистр сведений с курсами валют и т.п.).

Это становится возможным благодаря тому, что общие реквизиты объектов конфигурации можно использовать не только как "одинаковый реквизит, который есть у всех объектов", но и как идентификатор того, что данные относятся к какой-то одной из нескольких независимых областей.

Более подробно данный механизм на курсе не рассматривается (поясняющий пример можно найти на сайте фирмы 1С в "Толковом словаре 1С:Предприятие").

4.12.1. Определение пользователей

После настройки ролей их нужно "раздать". Список пользователей открывается при выполнении команды "Администрирование/Пользователи". В открывшемся окне можно указать параметры авторизации и отметить доступные данному пользователю роли.



4.12.2. Выгрузка/загрузка

Для получения архива всей базы можно использовать команду главного меню "Администрирование/Выгрузить информационную базу". Сформируется файл с расширением "dt". Это полная выгрузка данных, пользователей и всех настроек сделанных вами.

Обратная операция делается с помощью команды "Администрирование/Выгрузить информационную базу"

4.13. Запросы

Несмотря на то, что с запросами мы уже знакомы, начнем с самого начала.

Когда речь заходит о запросе, возникает ряд сопряженных с ним понятий:

- Источники данных (табличная модель данных)
- Структура запроса (описание запроса)
- Обработка результата запроса

4.13.1. Источники данных

Таблицы в "1С:Предприятии 8" подразделяются на два основных класса: реальные и виртуальные.

Если исходить из упрощенной модели системы, то:

- Реальные таблицы "хранятся" в базе данных. В случае использования реальной таблицы могут присутствовать вычисляемые поля, значения которых рассчитываются как функция нескольких разных полей.

- Виртуальные таблицы в базе данных не "хранятся". При обращении к информации виртуальных таблиц система автоматически "собирает" информацию из реальных таблиц для выполнения запроса. Виртуальная таблица может быть параметризована.

Отдельный подкласс таблиц образуют так называемые объектные таблицы. Эти таблицы предназначены для хранения состояния объектов системы, таких как справочники, документы и т.д. В таких таблицах присутствует поле "Ссылка" (ссылка на объект, данные которого содержит текущая запись таблицы).

В свою очередь любая таблица состоит из набора полей. В качестве поля таблицы может фигурировать:

- Обычное поле (содержащее какое-либо значение, либо значение типа "Null")
- Вложенная таблица

Основное отличие обычного поля от вложенной таблицы состоит в том, что в рамках одной записи обычному полю соответствует одно единственное значение, а вложенной таблице соответствует значение типа "РезультатЗапроса" с заранее заданным набором колонок.

Можно проиллюстрировать данное понятие следующим образом:

Дата	**Время**	**Контрагент**	**№**	**Товар**	**Кол-во**	**Сумма**
			Товары			
01,01,03	11:00	ООО "Все"	1	Карандаш	5	100
			2	Ручка	3	200
02,03,03	11:51	ООО "Куда"	1	Кнопки	10	20

Если продолжить разговор о полях, содержащих какие-либо значения, можно отметить: поле может содержать значение одного типа, может содержать значения нескольких типов (иметь составной тип), при этом для конкретной записи поле содержит значение одного типа.

4.13.2. Структура запроса (описание запроса)

Для выполнения запроса (получения необходимой выборки данных) необходимо составить текст запроса. Текст запроса – это инструкция, в соответствии с которой должен быть выполнен запрос.

После составления текста запроса его необходимо выполнить. По окончании выполнения полученный результат необходимо "обойти" (разобрать результат запроса).

Для формирования текста запроса существует специализированный язык запросов. Он определяет используемые синтаксические конструкции, структуру запроса.

Можно сказать, что текст запроса состоит из следующих секций:

- *Описание запроса*
- *Объединение запросов*
- *Упорядочивание результатов или Автоупорядочивание*

- *Описание итогов*

Из всех вышеуказанных секций обязательно наличие только описания запроса. В свою очередь, данная секция имеет следующую структуру:

Выбрать [Различные] [Первые <Количество>]

<Список полей выборки>

[Из <Список источников>]

[Где <Условие отбора>]

[Сгруппировать По <Поля группировки>]

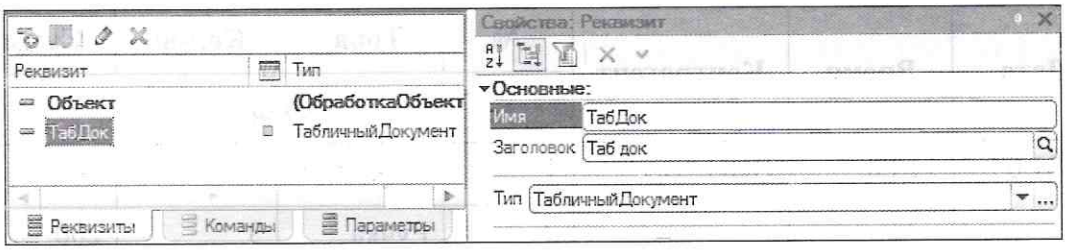
[Имеющие <Условия отбора>]

[Для изменения [[Of]<Список таблиц верхнего уровня>]]

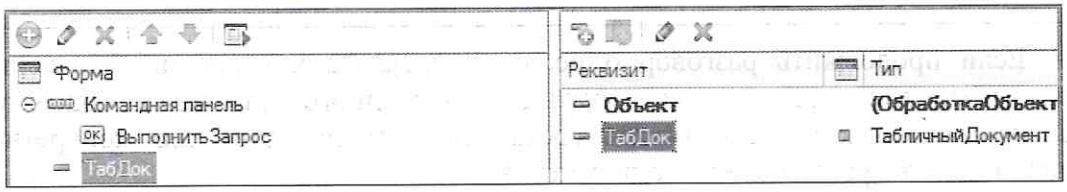
Уже в рамках данной секции обязательно только наличие "Выбрать" и указание полей выборки. Все остальные структурные элементы могут опускаться.

4.13.3. Использование конструктора запросов

Вернемся к обработке "Табличная модель". Добавим реквизит с именем "ТабДок" и типом "ТабличныйДокумент".

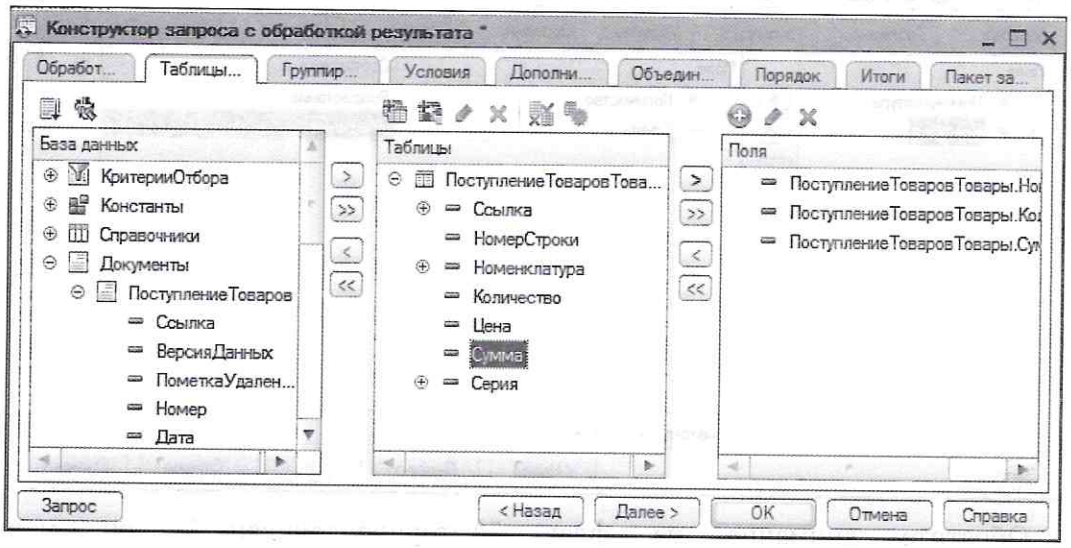


"Перетащим" этот реквизит в дерево элементов формы.

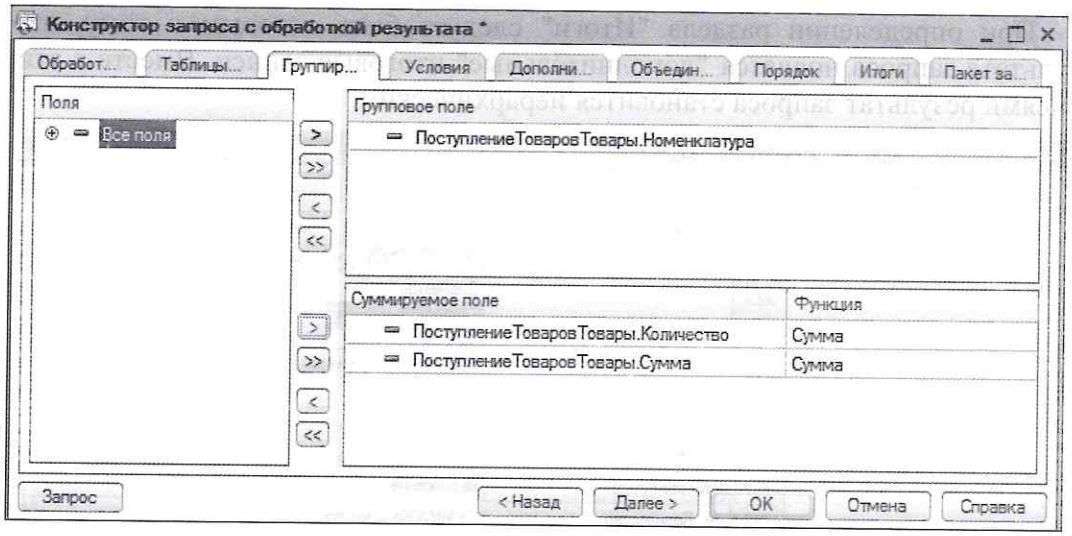


Для знакомства с механизмом запросов будем использовать конструктор запросов с обработкой результата, но тип обработки следует выбрать "Вывод в табличный документ"

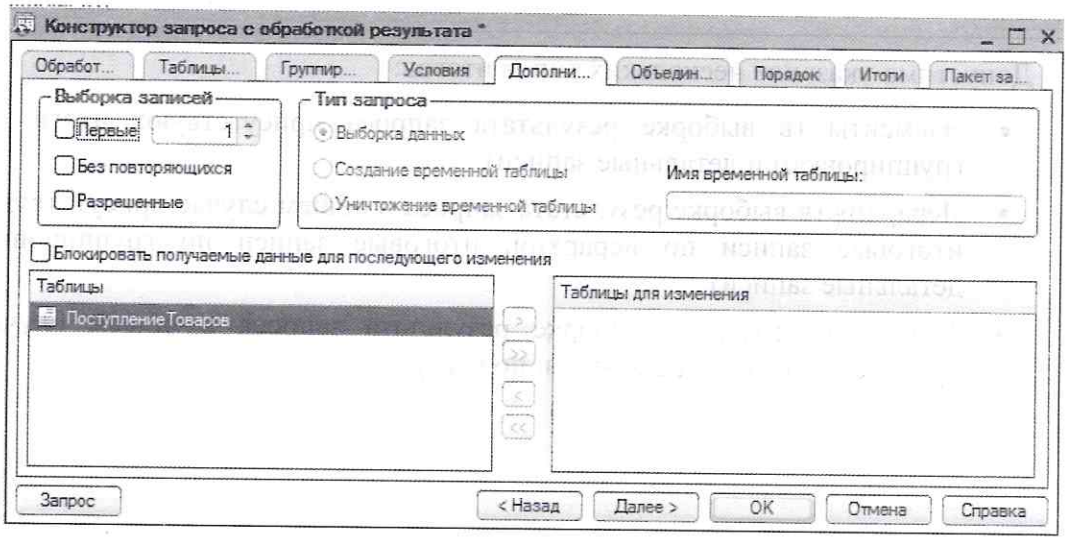
С первыми тремя закладками конструктора вы уже знакомы. В качестве источника данных выберем табличную часть "Товары" документа "ПоступлениеТоваров":



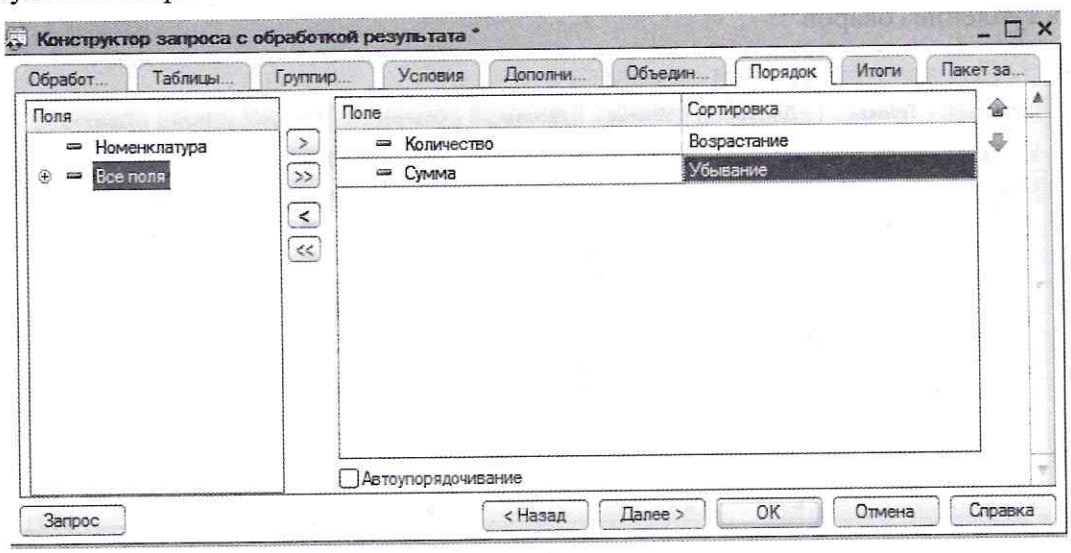
Группировки определим следующим образом:



На закладке "Дополнительно" можно отметить ряд флагов (касающихся ключевого слова "Выбрать" языка запросов) и определить состав таблиц, предназначенных для изменения.

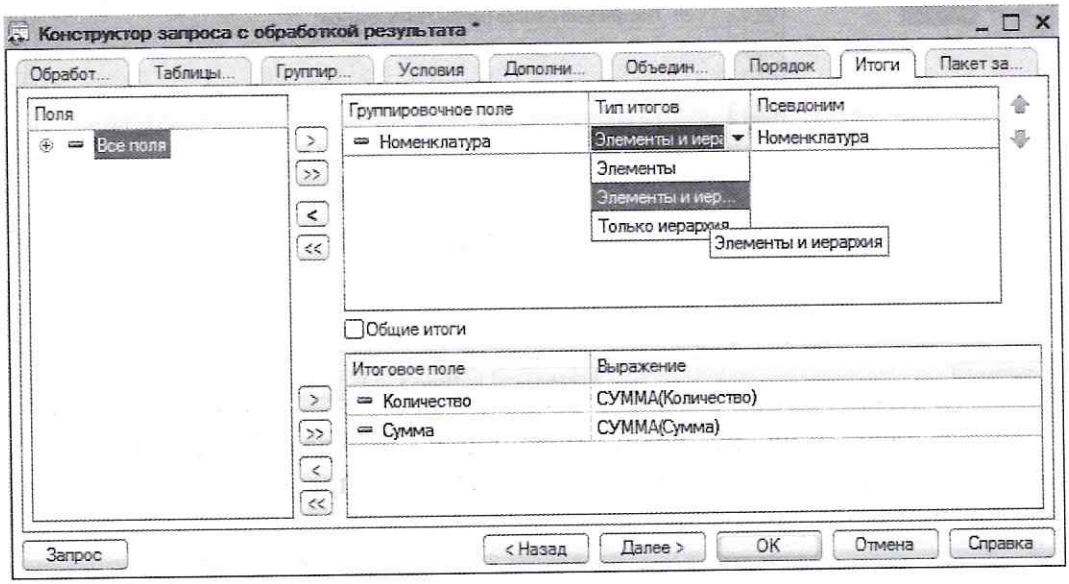


На закладке "Порядок" можно определить порядок сортировки записей в результате запроса.



Обратите внимание на флаг "Автоупорядочивание", он может использоваться для упорядочивания по полям ссылочного типа.

При определении раздела "Итоги" следует быть готовым к тому, что в результате запроса появятся "дополнительные" итоговые записи. Вместе с этими записями результат запроса становится иерархическим.



Допустимо указание нескольких типов итогов:

- Элементы (в выборке результата запроса присутствуют итоги по группировкам и детальные записи).
- Иерархия (в выборке результата запроса в общем случае присутствуют итоговые записи по иерархии, итоговые записи по группировке, детальные записи).
- Только иерархия (в выборке результата запроса в общем случае присутствуют итоговые записи по иерархии, детальные записи).

После нажатия на кнопку "ОК" конструктора, в модуле формы будет сформирован программный код следующего вида:

```
&НаКлиенте

Процедура ВыполнитьЗапрос(Команда)

        ВыполнитьЗапросСервер()

КонецПроцедуры


&НаСервере

Процедура ВыполнитьЗапросСервер()

        //{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА

        // Данный фрагмент построен конструктором.

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


        Макет = Обработки.ТабличнаяМодель.ПолучитьМакет("Макет");

        Запрос = Новый Запрос;

        Запрос.Текст =

                "ВЫБРАТЬ

                |       ПоступлениеТоваровТовары.Номенклатура КАК Номенклатура,

                |       СУММА(ПоступлениеТоваровТовары.Количество) КАК
Количество,

                |       СУММА(ПоступлениеТоваровТовары.Сумма) КАК Сумма

                |ИЗ

                |       Документ.ПоступлениеТоваров.Товары КАК
ПоступлениеТоваровТовары

                |

                |СГРУППИРОВАТЬ ПО

                |       ПоступлениеТоваровТовары.Номенклатура

                |

                |УПОРЯДОЧИТЬ ПО

                |       Количество,

                |       Сумма УБЫВ

                |ИТОГИ
```

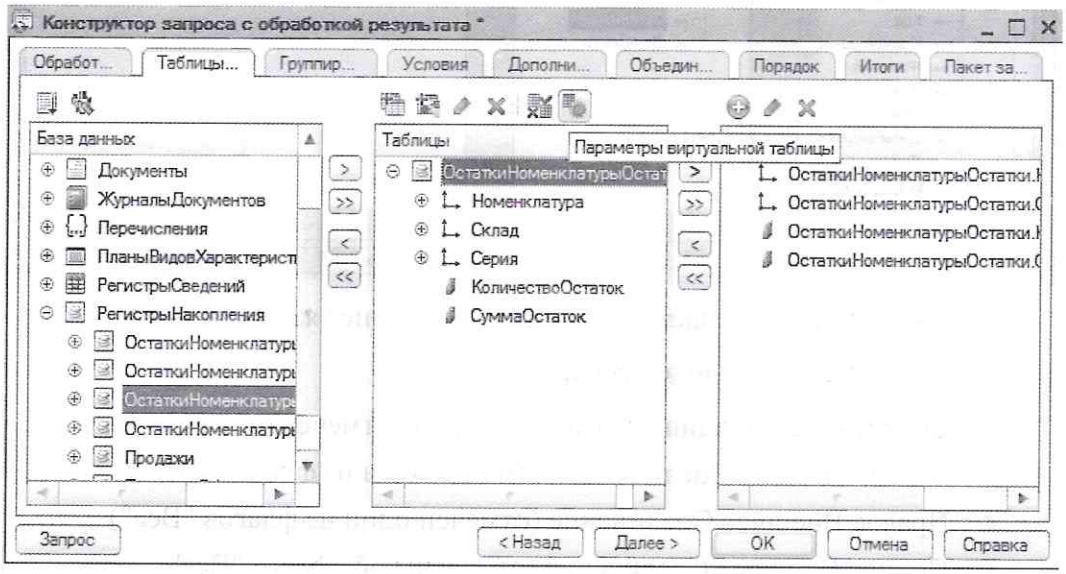
```
            |        СУММА(Количество),
            |        СУММА(Сумма)
            |ПО
            |        Номенклатура ИЕРАРХИЯ";

    Результат = Запрос.Выполнить();

    ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
    ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");
    ОбластьШапкаТаблицы = Макет.ПолучитьОбласть("ШапкаТаблицы");
    ОбластьПодвалТаблицы = Макет.ПолучитьОбласть("ПодвалТаблицы");
    ОбластьНоменклатураИерархия =
Макет.ПолучитьОбласть("НоменклатураИерархия");
    ОбластьНоменклатура = Макет.ПолучитьОбласть("Номенклатура");

    ТабДок.Очистить();
    ТабДок.Вывести(ОбластьЗаголовок);
    ТабДок.Вывести(ОбластьШапкаТаблицы);

    ВыборкаНоменклатура =
Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

    Пока ВыборкаНоменклатура.Следующий() Цикл
        Если ВыборкаНоменклатура.ТипЗаписи() =
ТипЗаписиЗапроса.ИтогПоИерархии Тогда
            Область = ОбластьНоменклатураИерархия;
        Иначе
            Область = ОбластьНоменклатура;
        КонецЕсли;

        Область.Параметры.Заполнить(ВыборкаНоменклатура);
        ТабДок.Вывести(Область);
    КонецЦикла;
```

```
    ТабДок.Вывести(ОбластьПодвалТаблицы);

    ТабДок.Вывести(ОбластьПодвал);


    //}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА

КонецПроцедуры
```

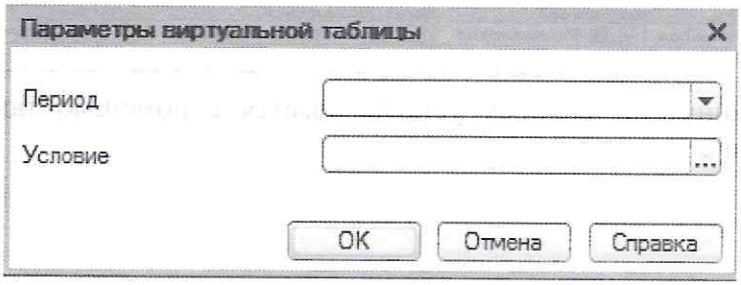
Выполняя повторный запуск конструктора с обработкой результата, попробуйте выполнить разные варианты упорядочивания, разные типы итогов. При этом обращайте внимание, как на изменение текста запроса, так и на изменение кода, формируемого конструктором в модуле формы.

4.13.4. Особенности работы с виртуальными таблицами

Работа с виртуальными таблицами имеет некоторую особенность. Заключается она в том, что в общем случае обязательно использование параметров виртуальных таблиц. Правильное использование данных параметров в некоторых случаях напрямую сказывается на оптимальности и быстродействии выполнения запроса.



Для задания параметров необходимо нажать на кнопку "Параметры виртуальной таблицы" и в открывшемся диалоговом окне указать их значения (имена параметров запроса).



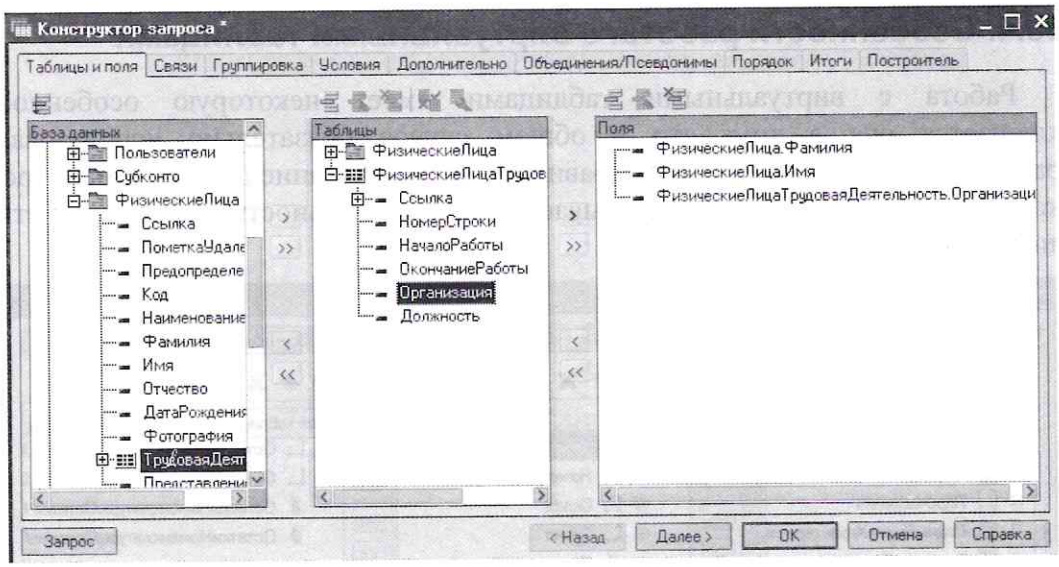
Состав параметров напрямую зависит от выбранной виртуальной таблицы. При определении условия можно повторно вложено вызывать конструктор запроса для определения подзапроса.

4.13.5. Построение запросов по нескольким таблицам

В реальной практике очень редко бывает ситуация, когда данные, которые необходимо получить определены в одной таблице. Чаще всего они находятся в различных таблицах, но должны быть получены вместе (при этом должна соблюдаться "связность" данных).

На закладке "Таблицы и поля" можно одновременно указывать несколько таблиц-источников данных. Для их увязки можно использовать раздел "Условия", либо можно использовать закладку "Связи". Данная закладка появляется в конструкторе запроса в случае, если в качестве источников данных выбрана более чем одна таблица.

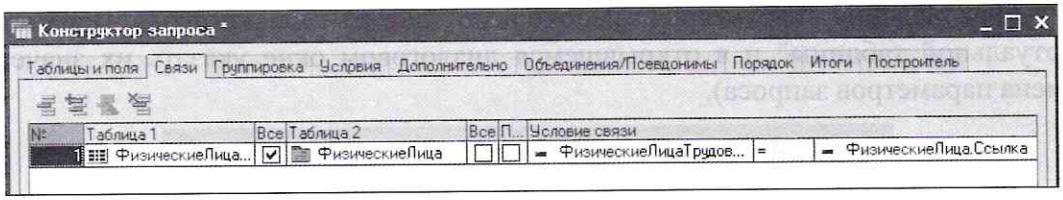
Если никакой связи между явно определенными источниками не определять, то в результат запроса войдет полная комбинация записей из определенных таблиц.



Закладка "Связи" отвечает за раздел "Соединение" языка запроса.

Различают несколько видов соединения:

- Внутреннее (ни один из флагов "Все" не отмечен)
- Левое Внешнее Соединение (отмечен один из флагов "Все")
- Правое Внешнее Соединение (отмечен один из флагов "Все")
- Полное Внешнее Соединение (отмечены оба флага "Все")



Связь данных в таблицах устанавливается с помощью так называемого "Условия связи".

Рассмотрим различие в вариантах соединения на следующем примере:

Есть две таблицы:

Таблица №1	
Номен	Номер1
Ручка	1
Карандаш	2
Вилка	3

Таблица №2	
ЕдИзм	Номер2
Шт.	1
Гр	3
Кг	4
банка	1

Условием соединения будет:

Таблица1.Номер1=Таблица2.Номер2

В качестве полей запроса определим две колонки: "Номер" из первой таблицы и "ЕдИзм" из второй таблицы.

В соответствии с условием можно выделить записи, для которых условие выполняется:

Ручка	1	1	Шт.
		1	банка
Вилка	3	3	Гр.

Записи, не удовлетворяющие условию соединения:

Из Таблицы №1

Карандаш	2

	Null

Из Таблицы №2

Null	

4	Кг.

Теперь рассмотрим варианты соединения:

Внутреннее соединение: в результат выполнения запроса войдут только данные записей из обеих таблиц, для которых выполняется условие соединения, т.е.

Ручка	Шт.
Ручка	банка
Вилка	Гр.

Левое внешнее соединение: в результат выполнения запроса войдут данные из записей, для которых выполняется условие соединения и "не вошедшие" из Таблицы №1. Можно сказать, что в результат запроса войдут все данные из Таблицы №1, и для тех записей результата запроса, для которых выполнялось условие соединения в полях, куда помещаются данные из таблицы №2, будут стоять значения, для которых условие не выполняется, будет стоять Null.

Ручка	Шт.
Ручка	банка
Вилка	Гр.
Карандаш	Null

Правое внешнее соединение обратно левому.

Ручка	Шт.
Ручка	банка
Вилка	Гр.
Null	Кг.

Полное внешнее соединение. В результат запроса войдут как записи, для которых выполнялось условие соединения, так и записи, полученные из "не вошедших" данных из обеих таблиц.

Ручка	Шт.
Ручка	банка
Вилка	Гр.
Карандаш	Null
Null	Кг.

В любом случае, даже в результате использования полного внешнего соединения не получается полная комбинация значений. (В случае полного соединения добавляются записи, не удовлетворяющие условию, а не все возможные их комбинации).

Практикум № 8

1. Получите данные о контактных лицах, их телефонах, полном наименовании контрагентов.

2. Получите список пяти наиболее дорогих (по ценам продажи) товаров.

3. Получите данные о том, какой контрагент, на какую сумму поставил нашей компании товара. В результате запроса должны присутствовать итоги и по группам справочника "Контрагенты".

4. Получите список из 5 самых продаваемых (по количеству) товаров.

4.13.6. Работа с временными таблицами

Использование временных таблиц помогает повысить скорость выполнения запросов, сделать процесс построения сложных запросов более простым и организовать исполнение таких запросов "поэтапно".

Возможность использования временных таблиц определяется наличием:

- Объекта "МенеджерВременныхТаблиц".
- Свойства "МенеджерВременныхТаблиц" объекта "Запрос".
- Расширением языка запросов по работе с временными таблицами.

В тексте запроса для работы с временными таблицами можно использовать следующие ключевые слова:

- Поместить
- Индексировать По
- Уничтожить

В качестве иллюстрации использования механизма временных таблиц рассмотрим следующий код:

```
// Создание менеджера временной таблицы
МенеджерВТ=Новый МенеджерВременныхТаблиц;


Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос.Текст = "ВЫБРАТЬ
|   ПродажаТоваров.Номенклатура,
|   СУММА(ПродажаТоваров.Количество) КАК Количество,
|   СУММА(ПродажаТоваров.Сумма) КАК Сумма
|ПОМЕСТИТЬ ТЧР
|ИЗ
|   Документ.ПродажаТоваров.Товары КАК ПродажаТоваров
```

```
|ГДЕ
|       ПродажаТоваров.Ссылка = &ЗначениеСсылки
|
|СГРУППИРОВАТЬ ПО
|       ПродажаТоваров.Номенклатура";
        Запрос.УстановитьПараметр("ЗначениеСсылки", ПродажаТоваров);


        //Создание временной таблицы
        Результат = Запрос.Выполнить();
```

Использование временной таблицы:

```
        Запрос = Новый Запрос;
        // в свойство "МенеджерВременныхТаблиц" записываем менеджер,
        // в контексте которого была создана временная таблица
        Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
        Запрос.Текст = "ВЫБРАТЬ
        |       ТЧР.Номенклатура Как Товар,
        |       Остатки.КоличествоОстаток, ТЧР.Количество
        |ИЗ
        | ТЧР
        | Левое Соединение
        |       РегистрНакопления.ОстаткиНоменклатуры.Остатки(,Номенклатура В
(Выбрать Номенклатура Из ТЧР)) КАК Остатки
        | По ТЧРасходной.Номенклатура=Остатки.Номенклатура";


        Результат = Запрос.Выполнить();
```

В качестве источников данных для временных таблиц могут использоваться:

- Таблица значений
- Табличная часть
- Результат запроса

Пример использования данной возможности:

```
Запрос=Новый Запрос;


Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос.Текст= "ВЫБРАТЬ

|        Товар,

|        Количество

|ПОМЕСТИТЬ ТЧРеализацииИОстатки

|ИЗ

|        &ВнешнийИсточник Как Внешний ";

Запрос.УстановитьПараметр("ВнешнийИсточник",ТЗВнешнийИсточник);

Результат = Запрос.Выполнить();
```

Для удаления всех временных таблиц, созданных в контексте экземпляра объекта "МенеджерВременныхТаблиц", нужно использовать метод "Закрыть()".

```
МенеджерВТ.Закрыть();
```

Для удаления одной таблицы:

```
Запрос = Новый Запрос;


Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос.Текст = "Уничтожить ТЧРеализацииИОстатки ";

Результат = Запрос.Выполнить();
```

4.13.7. Использование предопределенных данных

В тексте запроса можно использовать предопределенные данные системы (не передавать их через параметры, а описывать их использование "напрямую").

```
Запрос = Новый Запрос;

Запрос.Текст = "ВЫБРАТЬ

        |        Номенклатура.Код,

        |        Номенклатура.Наименование

        |ИЗ

        |        Справочник.Номенклатура КАК Номенклатура

        |ГДЕ
```

```
        |        Номенклатура.ОснЕдиницаИзмерения                    =
ЗНАЧЕНИЕ(Справочник.ЕдиницыИзмерения.Штука)";

Результат = Запрос.Выполнить();

// Обход результата запроса
```

Можно использовать системные перечисления:

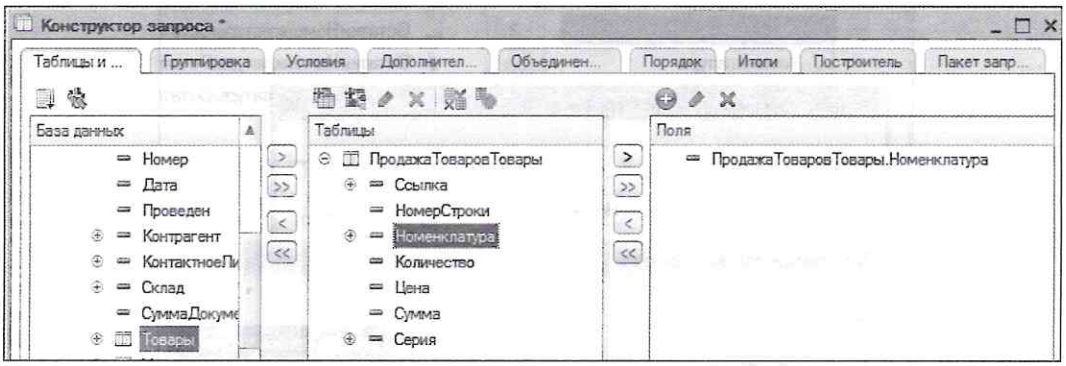
```
Запрос = Новый Запрос;

Запрос.Текст = "ВЫБРАТЬ

        |        Основной.Код,

        |        Основной.Наименование

        |ИЗ

        |        ПланСчетов.Основной КАК Основной

        |ГДЕ

        |        Основной.Вид = ЗНАЧЕНИЕ(ВидСчета.Активный)";


Результат = Запрос.Выполнить();

// Обход результата запроса
```

Напрямую указывать пустые ссылки:

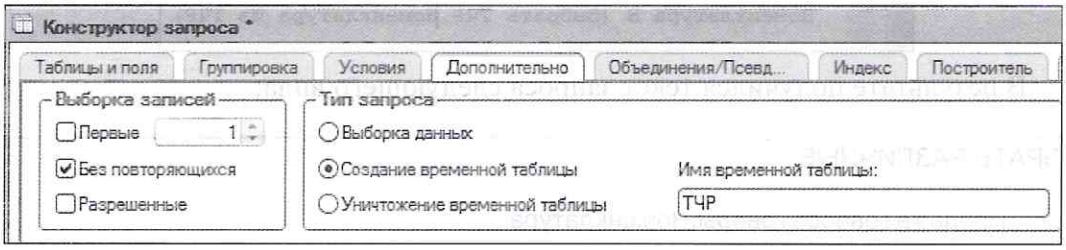
```
Запрос = Новый Запрос;

Запрос.Текст = "ВЫБРАТЬ

        |        ПРЕДСТАВЛЕНИЕ(ПродажаТоваровТовары.Ссылка),

        |        ПродажаТоваровТовары.НомерСтроки

        |ИЗ

        |        Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары

        |ГДЕ

        |        ПродажаТоваровТовары.Номенклатура                   =
ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка)";

Результат=Запрос.Выполнить();

// Обход результата запроса
```

4.13.8. Пакетные запросы

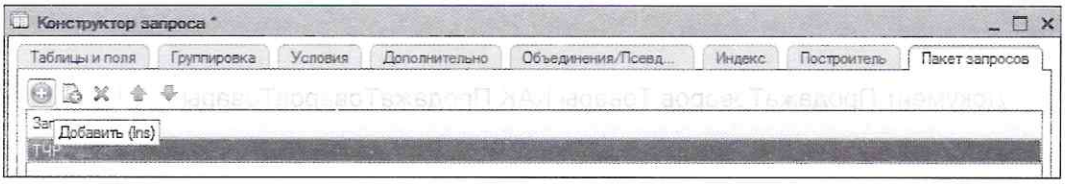
Определим первую закладку конструктора, как это сделано на рисунке:



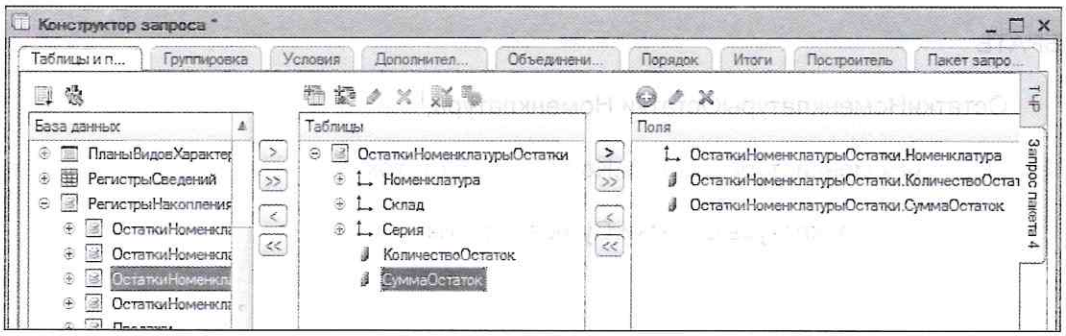
На закладке "Дополнительно" отметим флаг "Без повторяющихся" и укажем, что в результате исполнения этого запроса должна создаться временная таблица с именем "ТЧР".



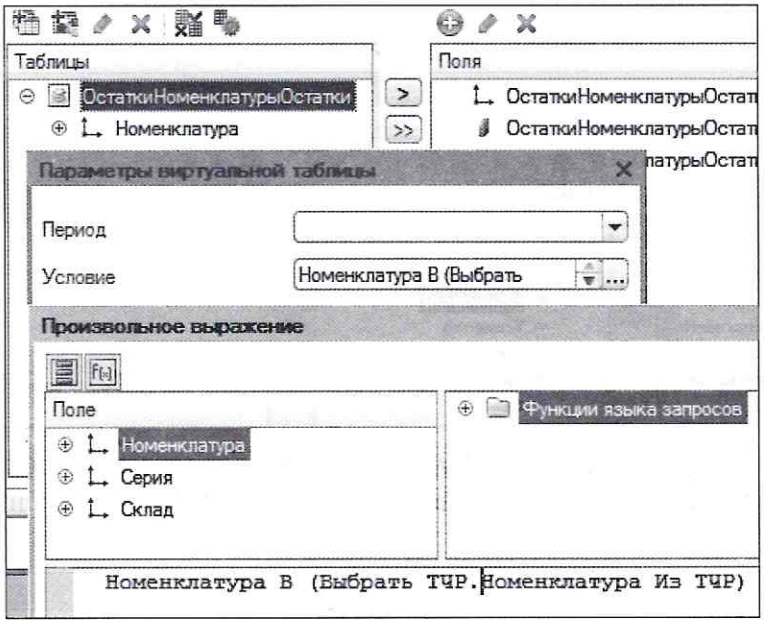
На закладке "Пакет запросов" добавим еще один запрос.



Второй запрос пакета определим в соответствии с рисунком



Определим параметр виртуальной таблицы "Условие" следующим образом:



В результате получился текст запроса следующего вида:

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
	ПродажаТоваровТовары.Номенклатура
ПОМЕСТИТЬ ТЧР
ИЗ
	Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
;
////////////////////////////////////////////////////////////////////////////////
ВЫБРАТЬ
	ОстаткиНоменклатурыОстатки.Номенклатура,
	ОстаткиНоменклатурыОстатки.КоличествоОстаток,
	ОстаткиНоменклатурыОстатки.СуммаОстаток
ИЗ
	РегистрНакопления.ОстаткиНоменклатуры.Остатки(
			,
			Номенклатура В
				(ВЫБРАТЬ
					ТЧР.Номенклатура
				ИЗ
					ТЧР)) КАК ОстаткиНоменклатурыОстатки
```

4.14. Отчеты

Любая система автоматизации учета только тогда выполняет свои функции, когда она имеет средства обработки накопленной в системе информации и получения сводных данных в удобном для просмотра и анализа виде.

Для получения разнообразной выходной информации в системе чаще всего используются объекты конфигурации, называемые "Отчетами".

Приемы работы с отчетами и обработками похожи и мало чем отличаются от приемов работы с другими объектами системы. Но функциональность отчетов шире за счет возможности использования системы компоновки данных.

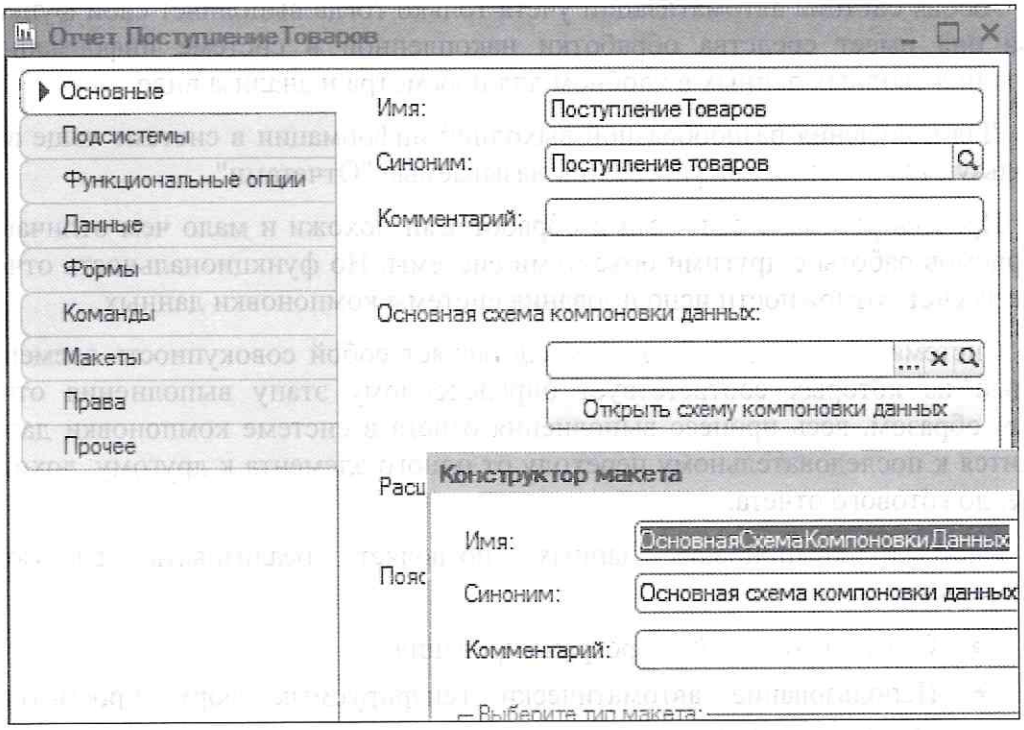
Система компоновки данных представляет собой совокупность элементов, каждый из которых соответствует определенному этапу выполнения отчета. Таким образом, весь процесс выполнения отчета в системе компоновки данных сводится к последовательному переходу от одного элемента к другому, доходя, в итоге, до готового отчета.

Система компоновки данных позволяет реализовать следующие возможности:

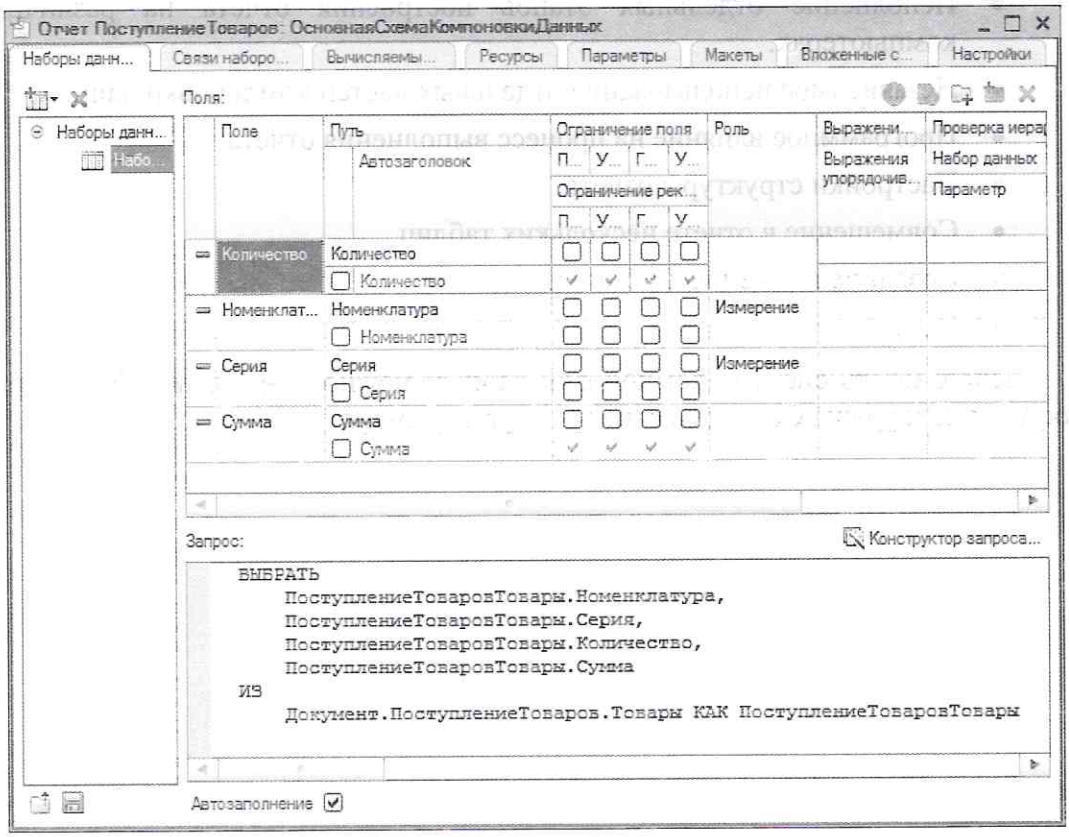
- Создание отчета без программирования.
- Использование автоматически генерируемых форм просмотра и настройки отчета.
- Разбиение исполнения отчета на этапы.
- Исполнение отдельных этапов построения отчета на различных компьютерах.
- Независимое использование отдельных частей компоновки данных.
- Программное влияние на процесс выполнения отчета.
- Настройки структуры отчета.
- Совмещение в отчете нескольких таблиц.
- Создание вложенных отчетов.
- И д.р.

Использовать систему компоновки данных можно по-разному. Рассмотрим простейший вариант. Создадим отчет о закупках товара.

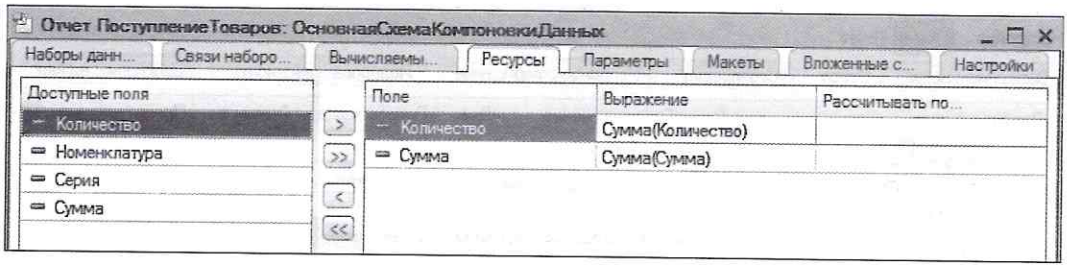
Создадим новый отчет и на первой закладке ("Основные") нажмем на кнопку "Открыть схему компоновки данных".



В открывшемся конструкторе схемы компоновки данных добавим новый набор данных (тип "Запрос") и определим запрос следующего вида:

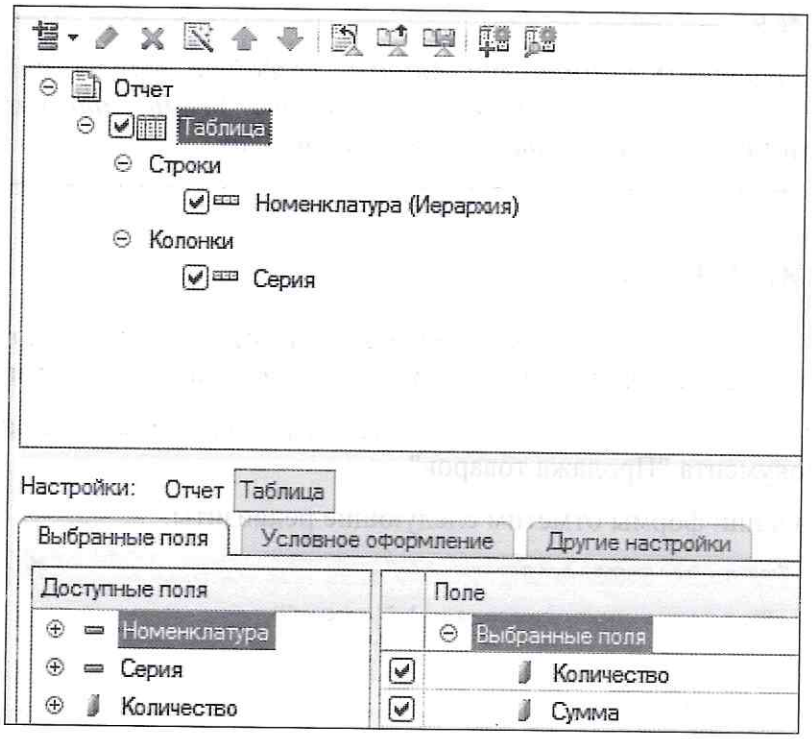


На закладке "Ресурсы" в качестве ресурсов нужно определить поля "Количество" и "Сумма".



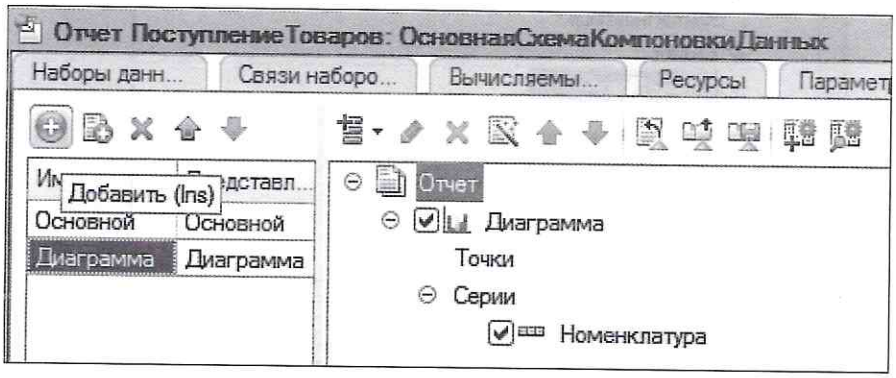
На закладке "Настройка" определим предварительный вид формируемой выходной формы. Из контекстного меню строчки "Отчет" выберем команду по созданию таблицы. В таблице по строкам определим группировку "Номенклатура" (с иерархией), в колонках определим "Контрагента". Спозиционироваться на строке "Таблица". В выводимых полях таблицы указать "Количество", "Сумма".

Что должно получиться, приведено на рисунке.

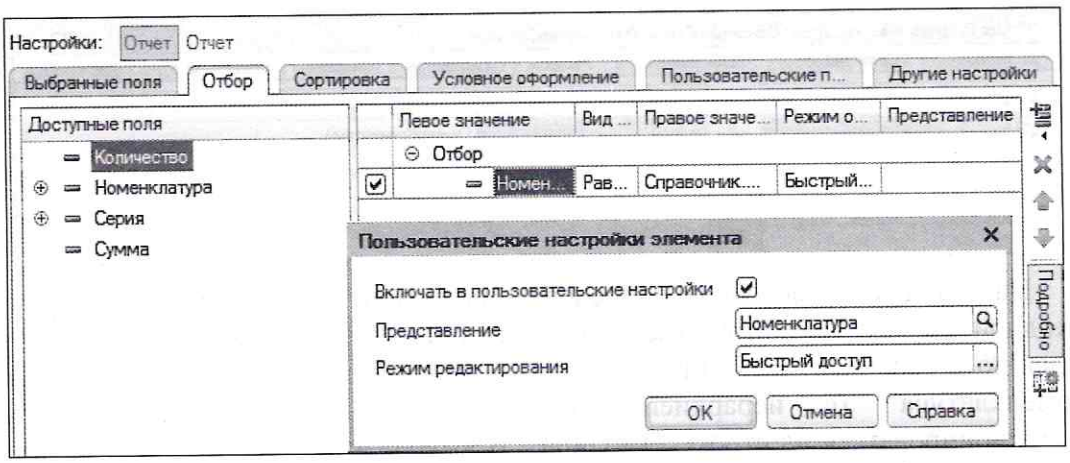


В пользовательском режиме поменяйте настройки, определенные по умолчанию. Команда "Все действия/Изменить вариант".

Создадим у этого отчета еще один вариант (на последней закладке конструктора схемы компоновки)



Для организации более простого доступа к настройкам можно использовать так называемые "пользовательские настройки".



Следует отметить, что такой тип как "Динамический список" работает "на основе" механизма компоновки данных.

Практикум № 9

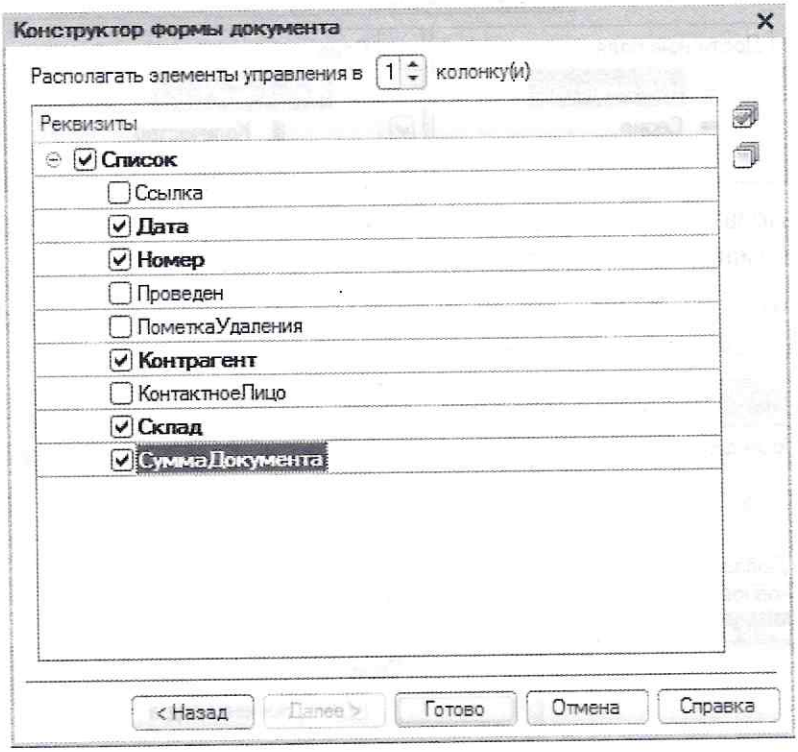
С помощью системы компоновки данных создайте отчет по продажам номенклатуры. Определите у него два варианта, реализуйте возможность выбора контрагента в пользовательских настройках.

4.15. Формы списка

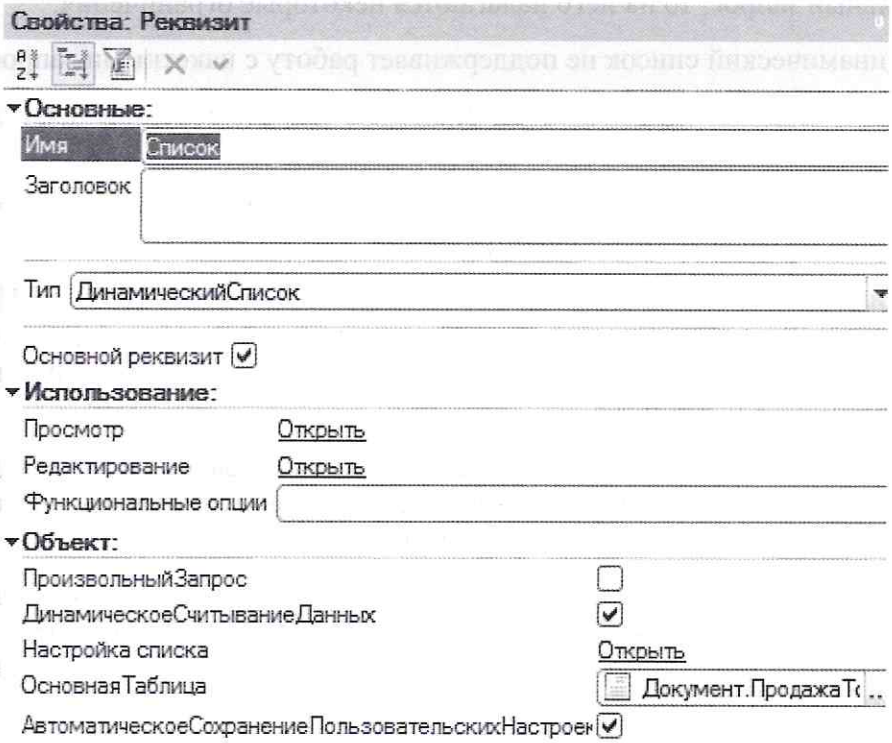
Возможности формы списка определяются возможностями такого типа, как "Динамический список". В его основе лежит механизм компоновки данных.

Познакомимся с возможностями данного типа на примере создания формы списка для документа "Продажа товаров".

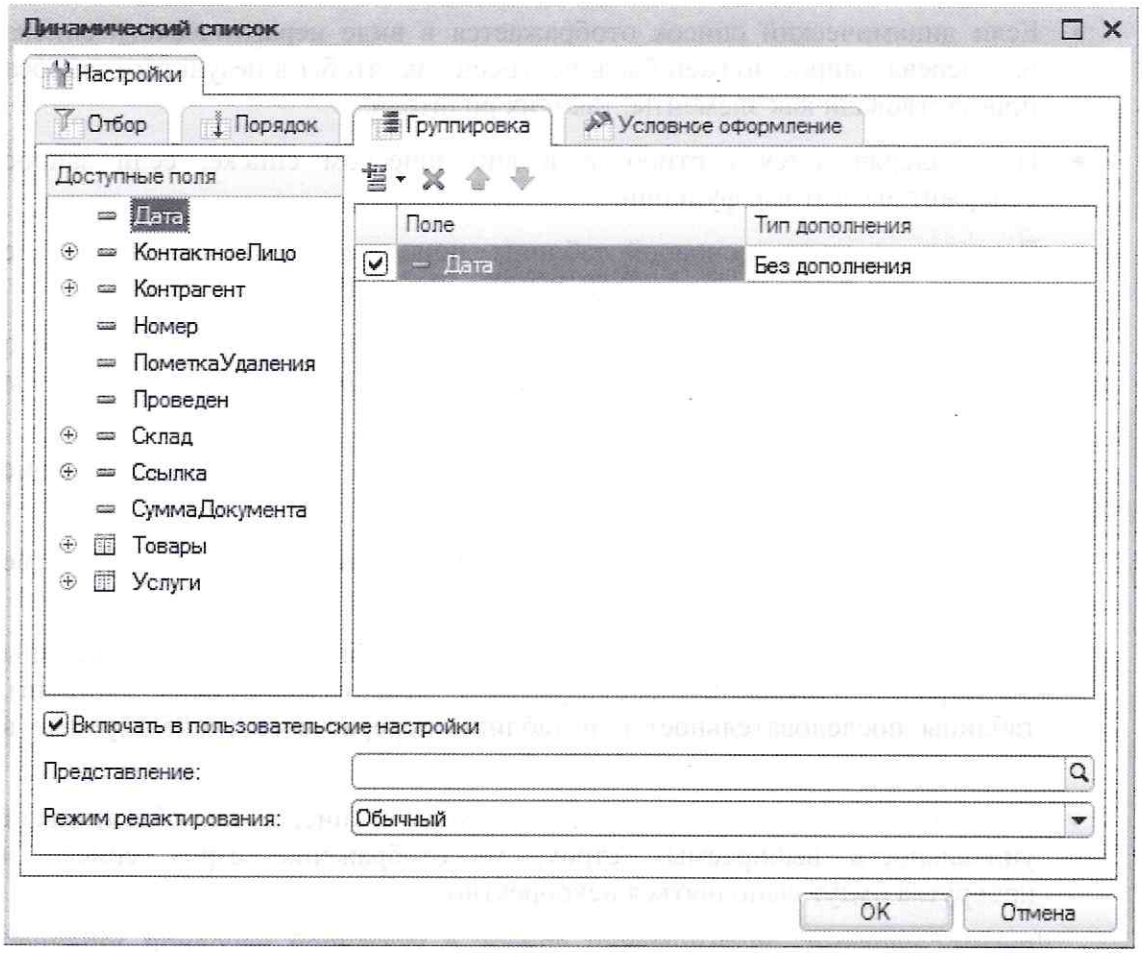
При создании формы отметим следующие реквизиты.



Обратите внимание на свойства основного реквизита:



Зайдем по гиперссылке в настройки и установим группировку по дате.



Остается проверить работоспособность механизма.

Следует иметь ввиду, что если используется возможность определить произвольный запрос, то на него налагаются некоторые ограничения:

- Динамический список не поддерживает работу с пакетными запросами.
- Динамический список не поддерживает в запросе объединения, если задана основная таблица.
- В динамических списках не поддерживается группировка, сортировка и отбор по реквизитам табличных частей.
- Динамический список не должен содержать секции УПОРЯДОЧИТЬ ПО, если задана основная таблица. Нужно использовать запрос без основной таблицы или задавать необходимое упорядочивание через настройки динамического списка.
- В числе полей запроса нельзя использовать поля подзапросов, возвращающих множественное количество значений. Необходимо использовать запрос без основной таблицы.
- Запрос не может содержать группировок и агрегатных функций, если задана основная таблица. Нужно использовать запрос без основной таблицы или задавать необходимые группировки через настройки динамического списка.
- Если динамический список отображается в виде иерархического списка или дерева, запрос не должен содержать условий отбора по родителю.
- Если динамический список отображается в виде иерархического списка или дерева, запрос должен быть построен так, чтобы в результате запроса присутствовали как элементы, так и их родители.
- Не поддерживается сортировка в динамическом списке, если запрос содержит агрегатные функции.
- В случае указания основной таблицы динамического списка запрос не должен содержать инструкций ПЕРВЫЕ и РАЗЛИЧНЫЕ.
- Не поддерживается указание в качестве основной таблицы динамического списка таблицы, которая используется в запросе только во внешнем соединении.

При создании динамического списка нужно учитывать следующие особенности:

- Не поддерживается указание табличных частей объектов в качестве основной таблицы динамического списка.
- В качестве основной таблицы не могут быть выбраны таблицы регистрации изменений (используемые в механизмах обмена данными), таблицы последовательностей и таблицы перерасчетов (используемые в механизмах периодических расчетов).
- Если запрос, заданный для динамического списка, не обеспечивает уникальность выбираемых строк, то отображение строк списка и прокрутка будут выполняться некорректно.

Другими словами, динамический список с указанной основной таблицей будет работать корректно в том случае, если в результате выполнения запроса,

указанного в качестве источника данных, не увеличивается количество строк, получаемых из основной таблицы (с учетом наложенного отбора).

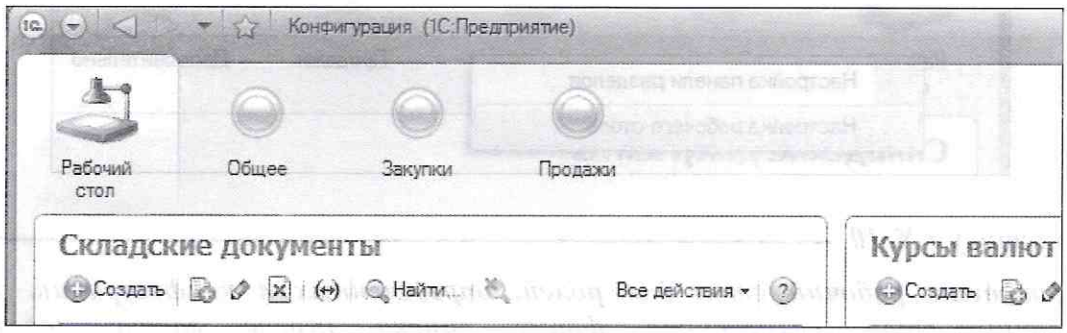
Если же в результате выполнения запроса количество строк, получаемых запросом из основной таблицы, увеличивается, это будет приводить к нарушению уникальности ключа записей таблицы, отображаемой списком.

В этом случае необходимо отключить использование основной таблицы.

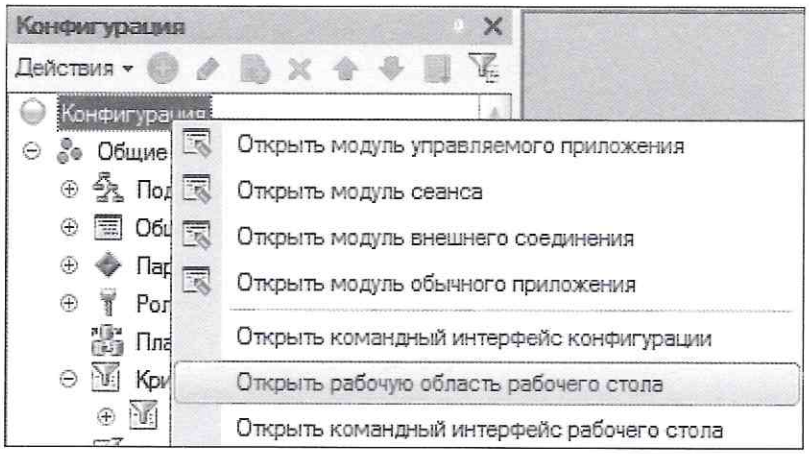
- При программном изменении свойств динамического списка не происходит автоматического повторного заполнения командных панелей, связанных с этим динамическим списком.
- Для динамического списка, который отображает список перечисления, отсутствует возможность интерактивной настройки списка.

4.16. Рабочий стол

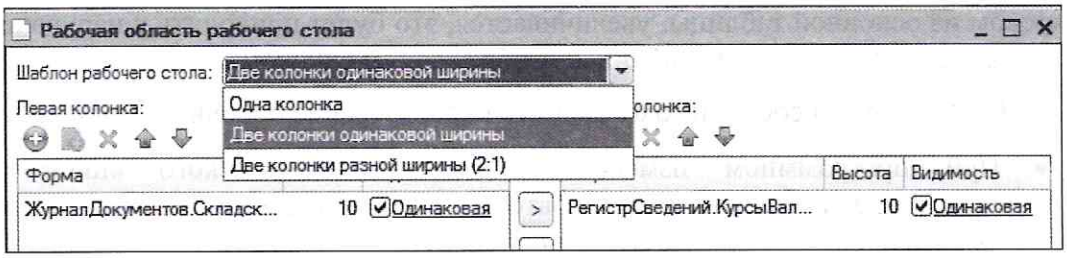
Предполагается, что работа пользователя в программном комплексе начинается с неких "контрольных точек" – форм. В зависимости от роли пользователя перечень этих "контрольных точек" может быть различным.



Для размещения форм, с которых начинается работа пользователя, предусмотрен "Рабочий стол". Для его настройки необходимо предварительно выполнить команду контекстного меню корневого объекта дерева объектов конфигурации.



В диалоговом окне можно выбрать один из трех вариантов оформления рабочего стола, в разрезе ролей пользователь определит формы, которые будут в нем располагаться.



На рабочем столе могут быть размещены только явно созданные формы ("автоформы" разместить не получится).

Состав форм, размещенных на рабочем столе, может быть откорректирован пользователем (команда контекстного меню раздела "Рабочий стол").

Практикум № 10

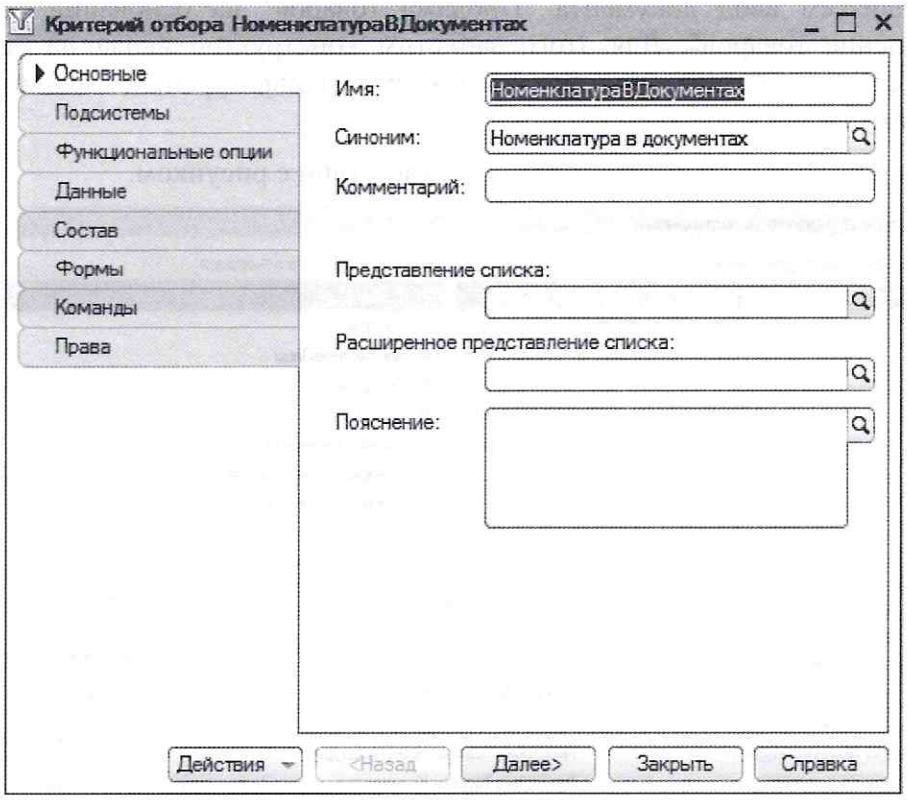
Настройте рабочий стол для ролей, определенных в конфигурации. На рабочий стол разместите форму списка курсов валют (сделав предварительно ее не основной формой). Посмотрите результат.

4.17. Критерии отбора

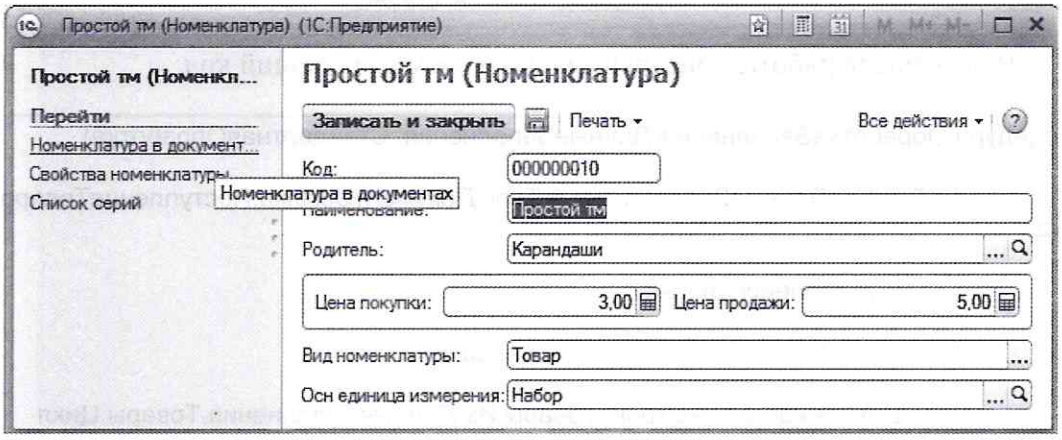
Критерий отбора представляет собой правило "поиска" указанного типа значения в объектах системы. Определяются внутри ветви "Общие".

Поставим себе задачу: необходимо отбирать документы, в которых встречается интересующая нас номенклатурная позиция.

Создадим критерий отбора "НоменклатураВДокументах". Определим тип критерия "СправочникСсылка.Номенклатура" (закладка "Данные"). На закладке "Состав" определим, какие объекты (по значениям каких реквизитов, реквизитов табличных частей) будут входить в результат отбора.



После его определения в панели навигации объекта появляется дополнительная команда.



Проверьте результат работы механизма на практике.

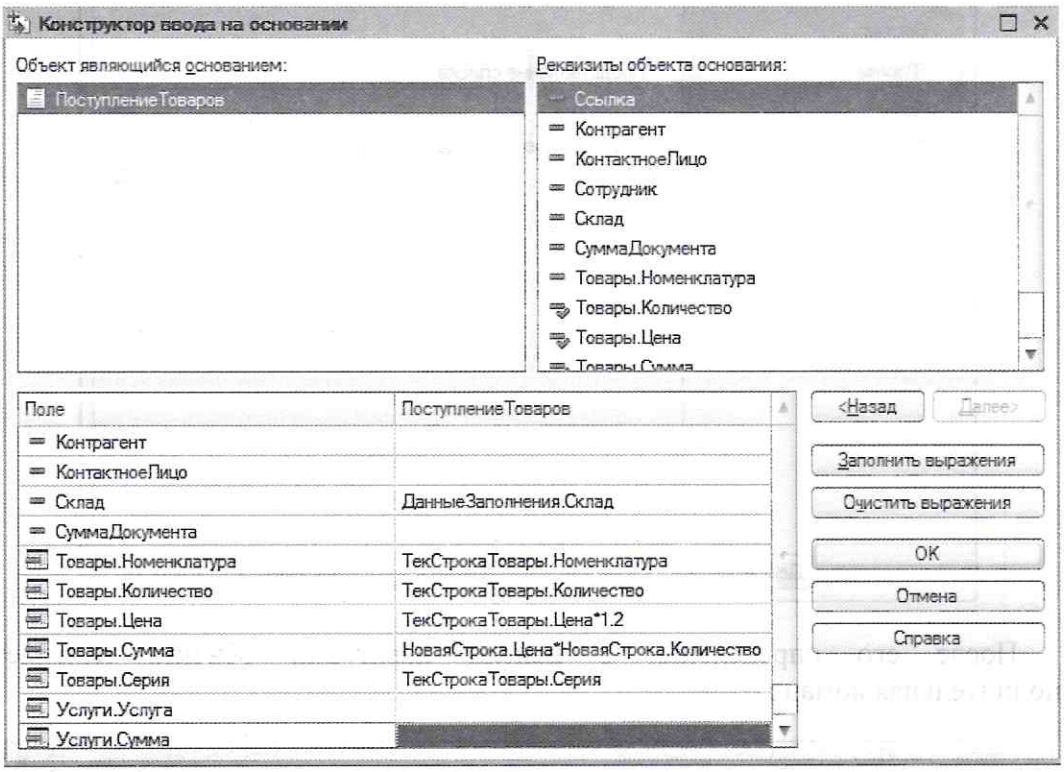
4.18. Обработка заполнения

В ряде случаев при создании объекта требуется какие-либо его реквизиты устанавливать в значении "по умолчанию". В том случае, если эти значения являются предопределенными, можно использовать свойство объекта конфигурации "Значение заполнения" (и флаг "Заполнять из данных заполнения"), в другом случае можно использовать событие "ОбработкаЗаполнения".

Это событие может использоваться как для начальной инициализации объекта, так и для организации механизма ввода на основании.

Реализуем ввод документа "Продажа товаров" на основании документа "Поступление товаров". Для этого запустим конструктор ввода на основании (одноименная закладка на форме объекта конфигурации, вводимого на основании).

Воспользуйтесь конструктором в соответствии с рисунком.



В результате работы конструктора получаем следующий код:

```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)

    Если ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ПоступлениеТоваров")
Тогда

        // Заполнение шапки

        Склад = ДанныеЗаполнения.Склад;

        Для Каждого ТекСтрокаТовары Из ДанныеЗаполнения.Товары Цикл

            НоваяСтрока = Товары.Добавить();

            НоваяСтрока.Количество = ТекСтрокаТовары.Количество;
```

```
                НоваяСтрока.Номенклатура = ТекСтрокаТовары.Номенклатура;

                НоваяСтрока.Цена = ТекСтрокаТовары.Цена*1.2;

                НоваяСтрока.Серия = ТекСтрокаТовары.Серия;

                НоваяСтрока.Сумма =
НоваяСтрока.Цена*НоваяСтрока.Количество;

            КонецЦикла;

        КонецЕсли;

КонецПроцедуры
```

Проверьте работоспособность механизма.

4.19. Обращение к методам объекта

В модуле объекта "Поступление товаров" определите экспортную процедуру, которая переподставляла бы цены покупки. Цены берутся из регистра сведений "Цены поставщиков" в соответствии с контрагентом документа, на дату документа по каждой номенклатуре, представленной в табличной части.

Необходимо организовать возможность пересчета цен после смены контрагента в документе. Код обработки события следующий:

```
&НаКлиенте

Процедура КонтрагентПриИзменении(Элемент)


    ПересчитатьНаСервере();                                    //1


КонецПроцедуры


&НаСервере

Процедура ПересчитатьНаСервере()


    Документ = РеквизитФормыВЗначение("Объект");               //2

    Документ.Пересчитать();                                    //3

    ЗначениеВРеквизитФормы(Документ, "Объект");                //4


КонецПроцедуры
```

Где "Пересчитать" - это экспортная процедура модуля объекта.

"Приблизительный код" метода:

```
Процедура Пересчитать() Экспорт

   Запрос=Новый Запрос;
   Запрос.Текст= "ВЫБРАТЬ
         |      ПоступлениеТоваровТовары.Номенклатура,
         |      ПоступлениеТоваровТовары.Количество,
         |      ПоступлениеТоваровТовары.Серия
         |ПОМЕСТИТЬ ТЧР
         |ИЗ
         |      Документ.ПоступлениеТоваров.Товары КАК ПоступлениеТоваровТовары
         |ГДЕ
         |      ПоступлениеТоваровТовары.Ссылка = &Ссылка
         |;
         |
         |////////////////////////////////////////////////////////////////////////////
         |ВЫБРАТЬ
         |      ТЧР.Номенклатура,
         |      ТЧР.Количество,
         |      ТЧР.Серия,
         |      ЦеныПоставщиковСрезПоследних.Цена,
         |      ЦеныПоставщиковСрезПоследних.Цена * ТЧР.Количество КАК Сумма
         |ИЗ
         |      ТЧР КАК ТЧР
         |            ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.ЦеныПоставщиков.СрезПоследних(
         |                  &Дата,
         |                  Контрагент = &Контрагент
         |                        И Номенклатура В
         |                              (ВЫБРАТЬ РАЗЛИЧНЫЕ
         |                                    ТЧР.Номенклатура
         |                              ИЗ
```

```
                |                                                    ТЧР)) КАК
ЦеныПоставщиковСрезПоследних

                |                ПО ТЧР.Номенклатура =
ЦеныПоставщиковСрезПоследних.Номенклатура";


        Запрос.УстановитьПараметр("Дата",Дата);

        Запрос.УстановитьПараметр("Ссылка",Ссылка);

        Запрос.УстановитьПараметр("Контрагент",Контрагент);

    ТЗ=Запрос.Выполнить().Выгрузить();

        Товары.Очистить();

    Товары.Загрузить(ТЗ);

КонецПроцедуры
```

ОБРАТИТЕ ВНИМАНИЕ на то, что в коде допущена логическая ошибка. Вам необходимо ее найти и исправить. В качестве помощи (для поиска ошибки) приводим схему выполняемых преобразований. В ней указаны помеченные строки кода обработчика события.

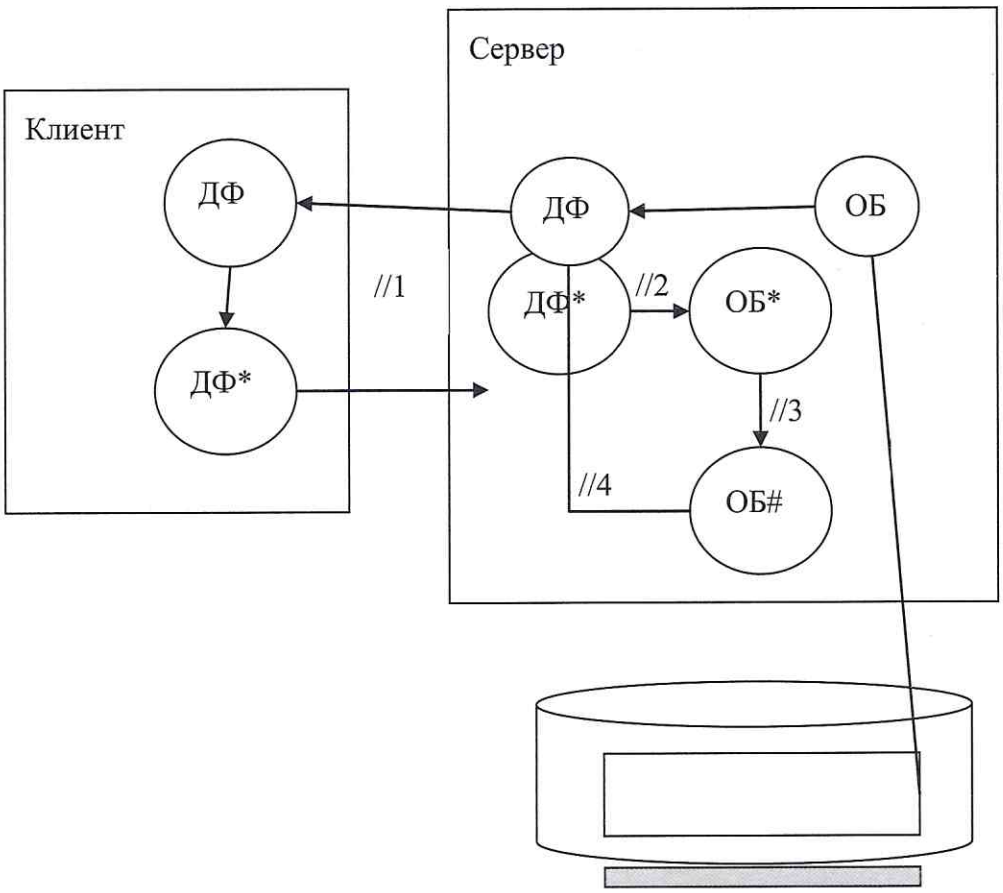


Схема 4.3

Где ОБ – Объект базы данных (тот, у которого есть метод "Записать" и т.д.).

ДФ – Данные формы (некое "представление" объекта базы данных, способное существовать как на стороне сервера, так и на стороне клиента).

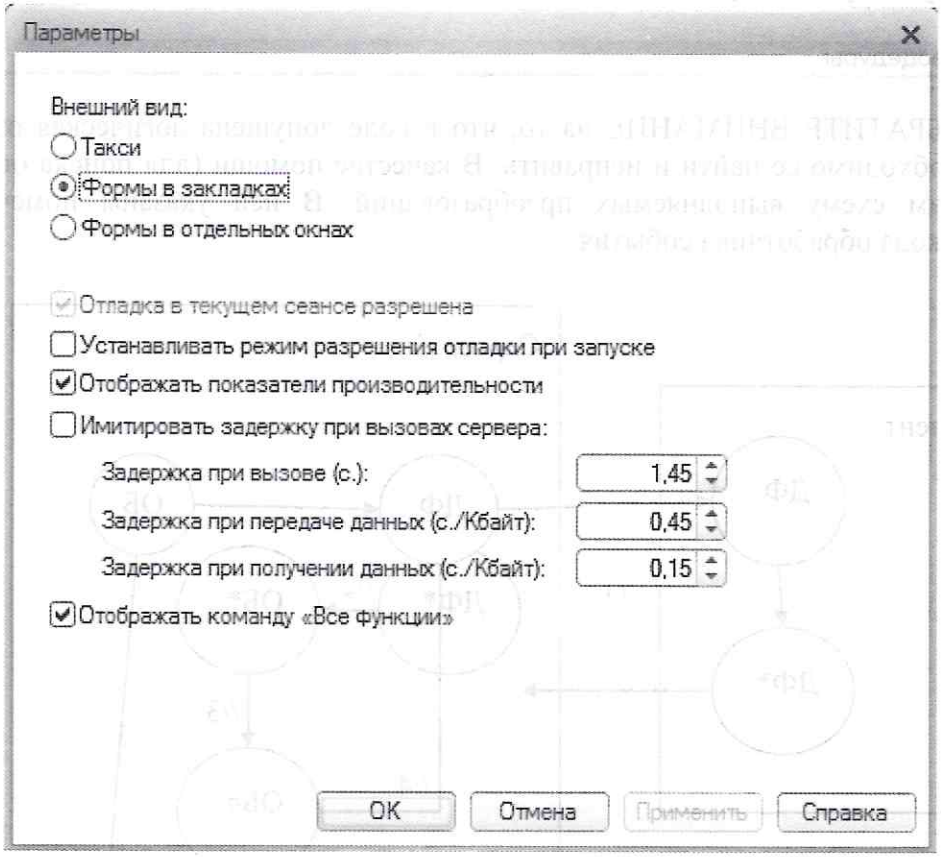
ДФ* - Данные формы, содержащие изменения, сделанные пользователем на стороне клиента.

ОБ* - Объект базы данных, содержащий изменения, сделанные пользователем на стороне клиента.

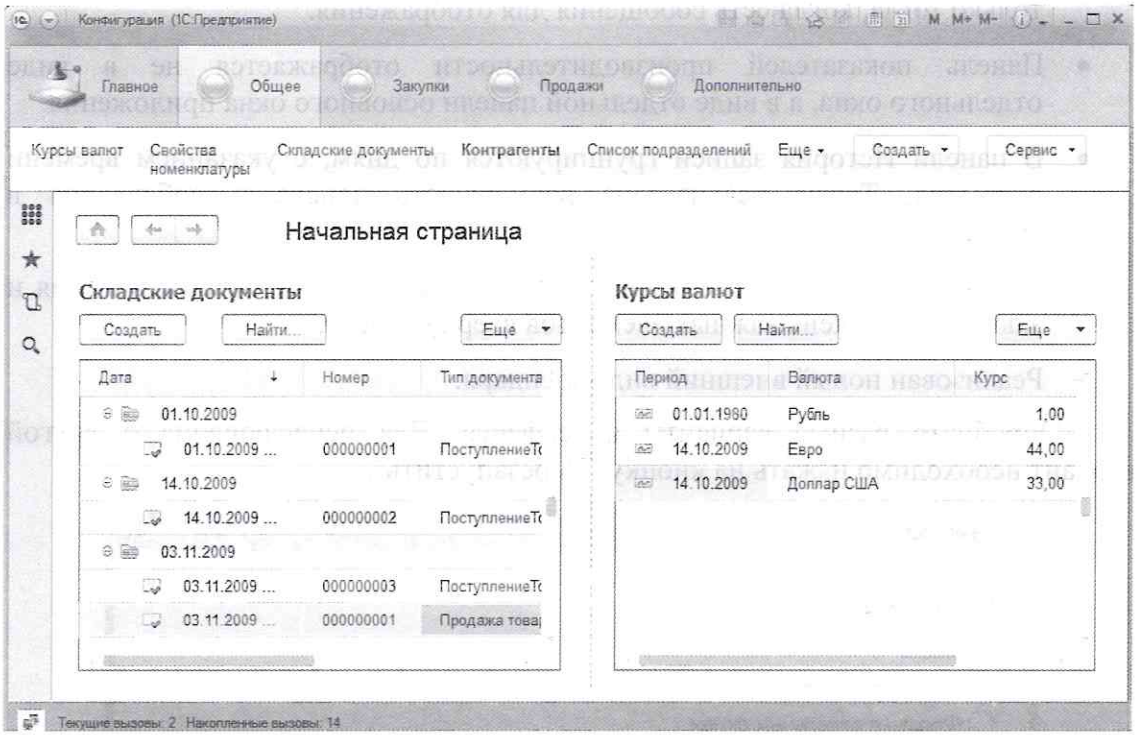
ОБ# - Объект базы данных, содержащий результат использования метода объекта.

4.20. Интерфейс "Такси"

Используя команду (в режиме исполнения) "Главное меню/Сервис/Параметры" можно менять внешний вид приложения:

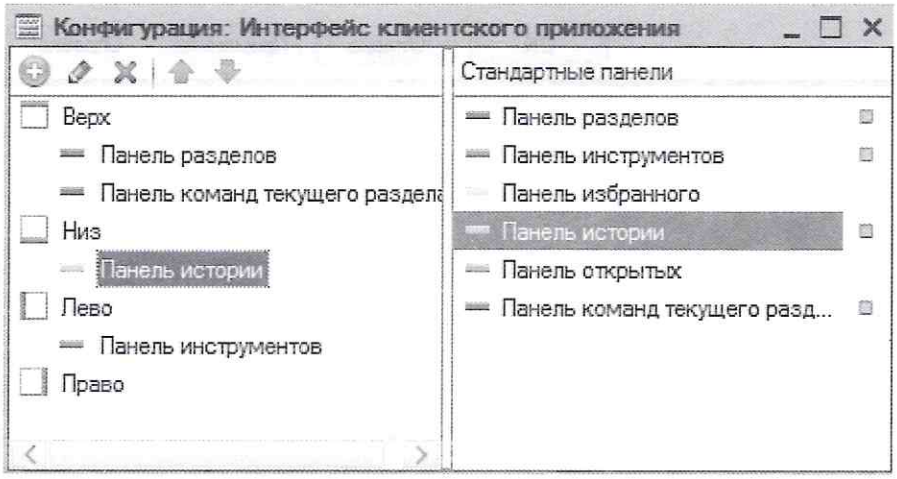


Новый вариант интерфейса клиентского приложения: Такси.



Интерфейс Такси включает в себя большое количество изменений, среди которых можно выделить:

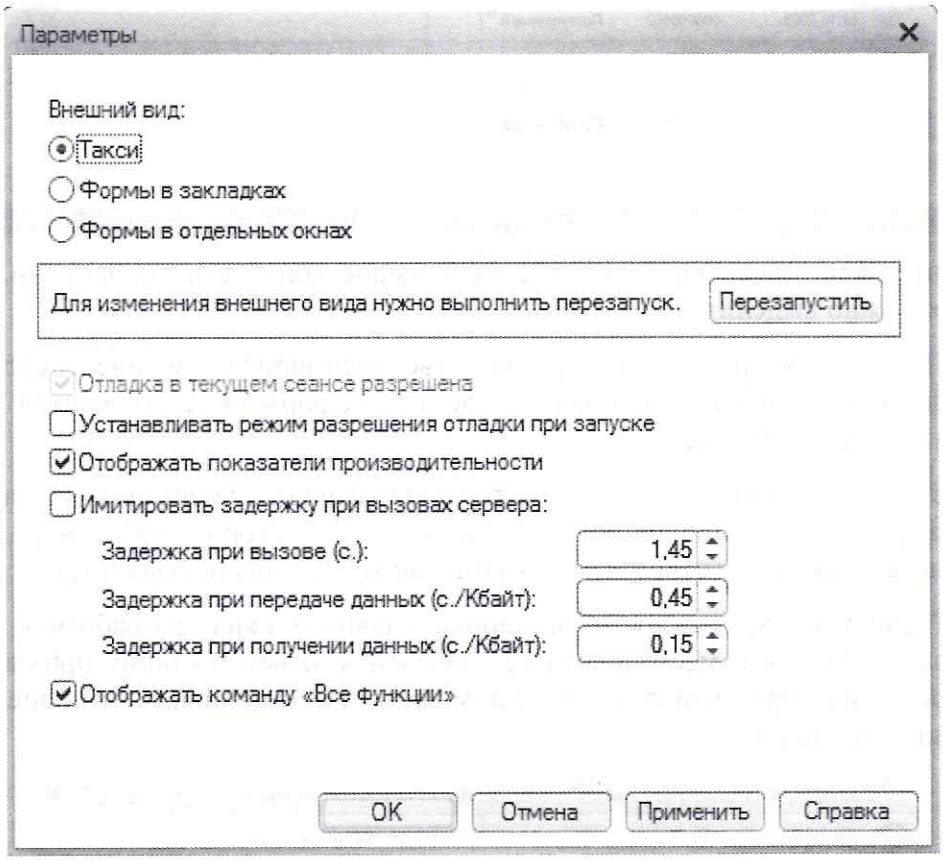
- Изменено оформление интерфейса: увеличен шрифт, увеличены отступы в формах, изменены основные цвета и оформление элементов форм, увеличен их размер.
- Усовершенствована навигация по приложению: ориентация на работу с одним окном, наличие специальных панелей (истории, избранного и т.д.), переходы между открытыми окнами как вперед, так и назад и т.д.
- Расширены возможности настройки главного окна: разработчик может задать состав и местоположение панелей, соответствующих прикладному решению, при этом пользователь может перекомпоновать эти панели так, как удобно ему.



- Использовать полнотекстовый поиск можно с помощью системной формы.
- Панель навигации формы размещена в верхней части формы.

- Окно сообщений расположено в нижней части формы и отображается только тогда, когда есть сообщения для отображения.
- Панель показателей производительности отображается не в виде отдельного окна, а в виде отдельной панели основного окна приложения.
- В панели История записи группируются по дням, с указанием времени открытия. Также реализована возможность поиска и добавления в Избранное.
- В панели Избранное реализована возможность поиска, переименования и закрепления очень важных элементов вверху списка.
- Реализован новый внешний вид календаря.

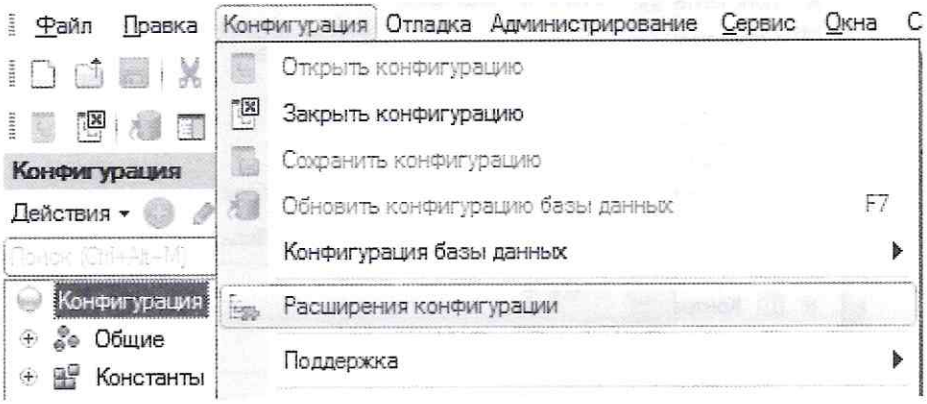
Опробуйте разные варианты интерфейса. Для переключения в другой вариант необходимо нажать на кнопку "Перезапустить".



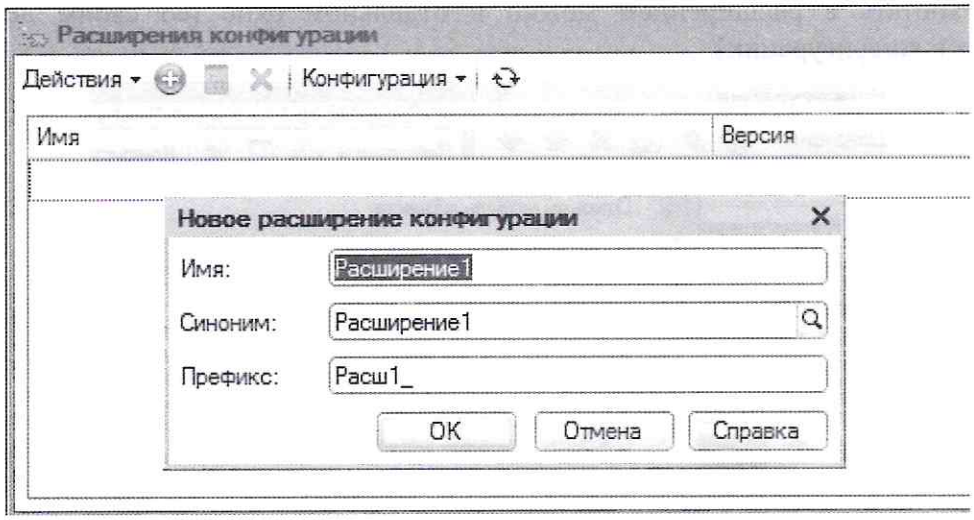
5. Дополнительно

5.1. Расширения конфигурации:

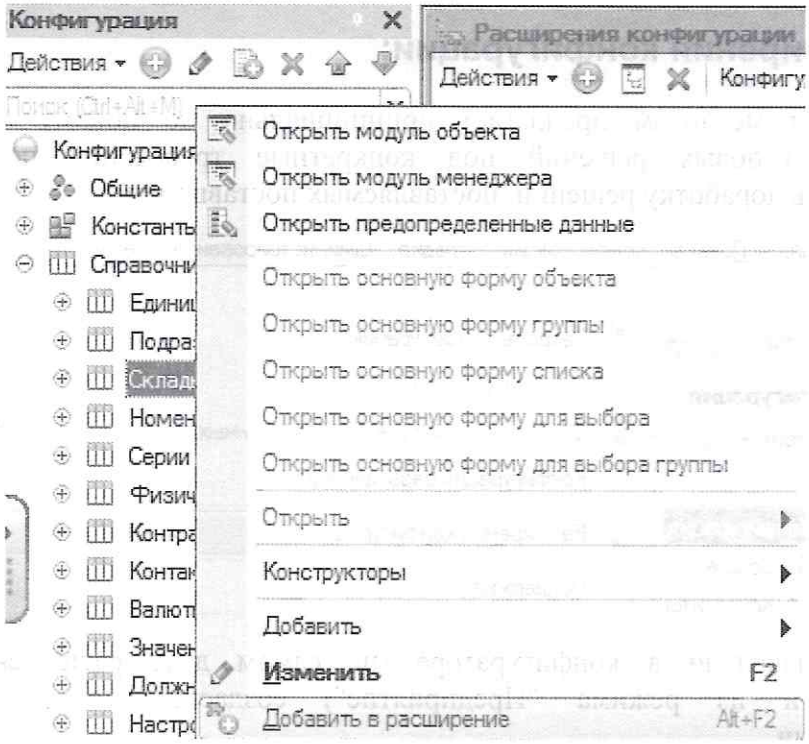
Данный механизм предлагает принципиально по новому производить адаптацию типовых решений под конкретные требования пользователей (производить доработку решений, поставляемых поставщиком).



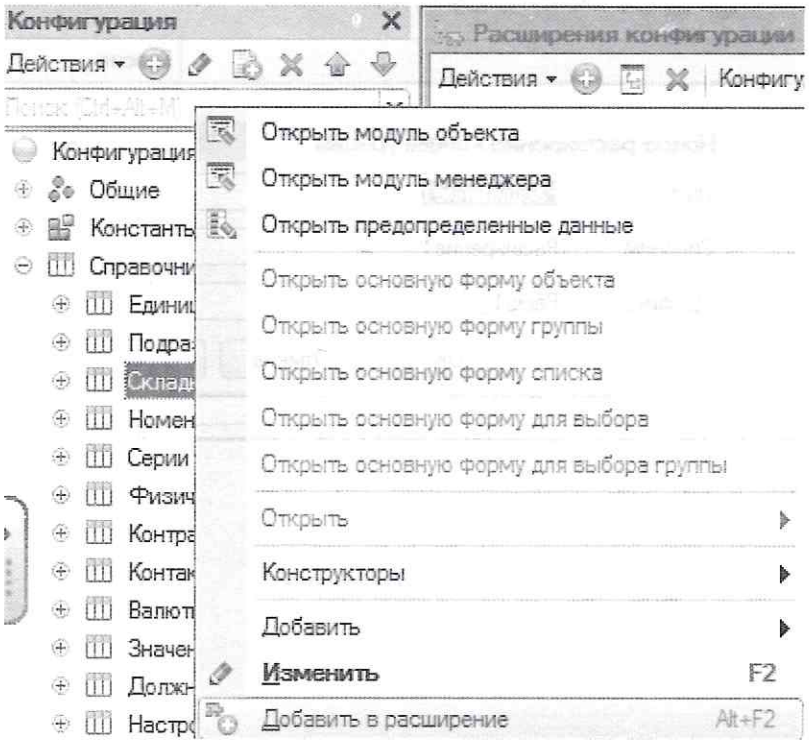
Первоначально в конфигураторе (на самом деле расширения можно загружать и из режима "Предприятие") создается новое расширение конфигурации.



Если расширение зависит от существующих объектов конфигурации, то для контроля при его загрузке можно данные объекты включить в расширение.



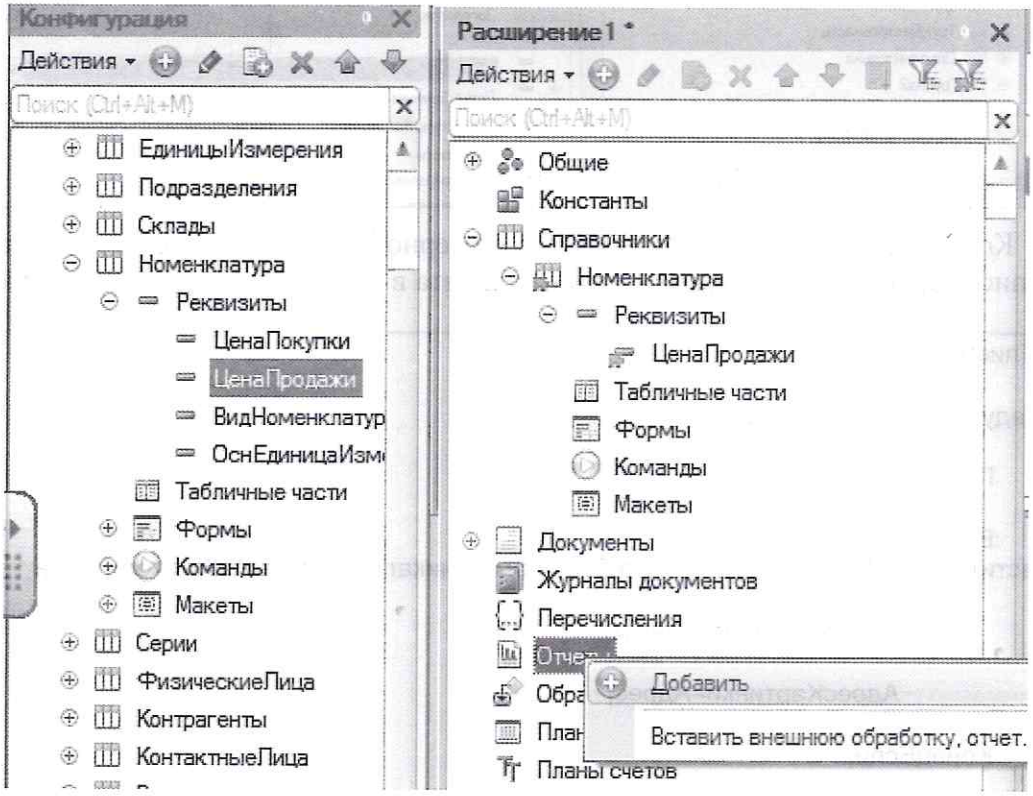
Работать с расширением можно в отдельном окне (со своим деревом объектов конфигурации).



На текущий момент с помощью расширения можно:

- Изменять управляемые формы, существующие в типовой конфигурации;
- Добавлять новые подсистемы.
- Изменять состав подсистем, имеющихся в типовой конфигурации;

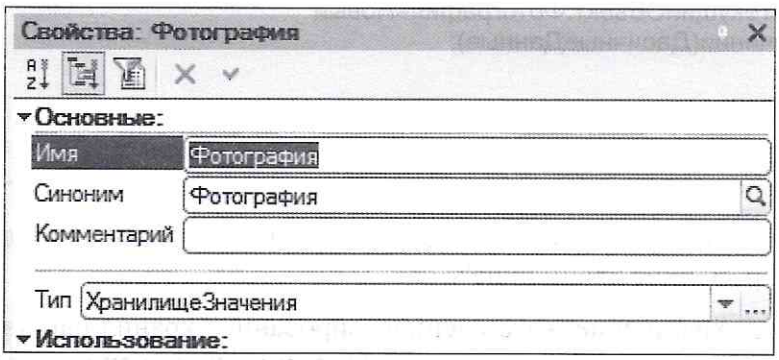
- Изменять роли типовой конфигурации, добавляя в них объекты, созданные в расширении;
- Изменять командный интерфейс типовой конфигурации (основного раздела, подсистем);
- Добавлять новые отчёты и обработки.



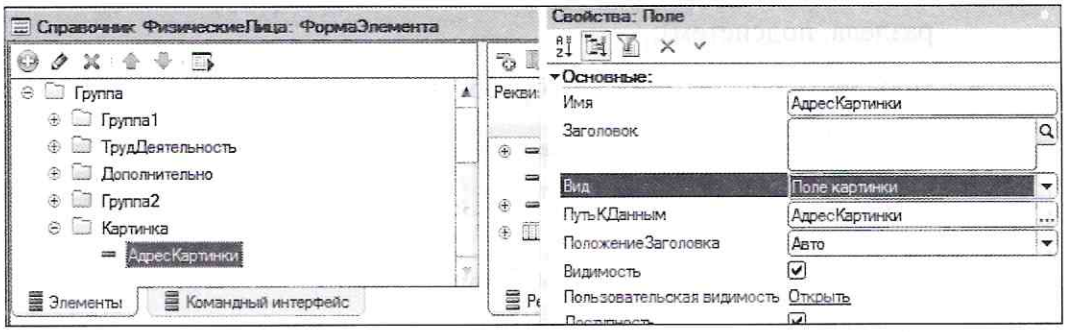
5.2. Хранилище значений (работа с картинками)

С помощью такого типа как "ХранилищеЗначения" можно организовать хранение в информационной базе "1С:Предприятие" "чего угодно". Важно понимать, что хранить в информационной базе следует "что-то", что нужно при решении задач учета, также следует обращать внимание на "место хранения" (идеальный вариант – регистр сведений) и размер сохраняемых данных.

Для демонстрации возможности работы с данным типом в справочнике "Физические лица" определим реквизит "Фотография" (как показано на рисунке).



Для отображения картинки, сохраненной в базе данных, определим реквизит формы "АдресКартинки" (тип "Строка"). После "переноса" реквизита в дерево элементов управления необходимо установить у него вид "Поле картинки".



Код обработчика события (вместе с серверной процедурой), ответственного за запись картинки из файловой системы клиента в базу данных, приведен ниже.

```
&НаКлиенте

Процедура Загрузить(Команда)

    Перем Имя,Адрес;

    Если
ПоместитьФайл(Адрес,"",Имя,Истина,ЭтаФорма.УникальныйИдентификатор) Тогда

        Объект.ИмяФайла=Имя;

        АдресКартинки=Адрес;

    КонецЕсли;

КонецПроцедуры


&НаСервере

Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

    Если ЭтоАдресВременногоХранилища(АдресКартинки) Тогда

        ДвоичныеДанные=ПолучитьИзВременногоХранилища(АдресКартинки);

        ТекущийОбъект.Фотография=Новый
ХранилищеЗначения(ДвоичныеДанные);

    КонецЕсли;

КонецПроцедуры
```

Следует отметить, что в данном фрагменте кода используется так называемое "Временное хранилище".

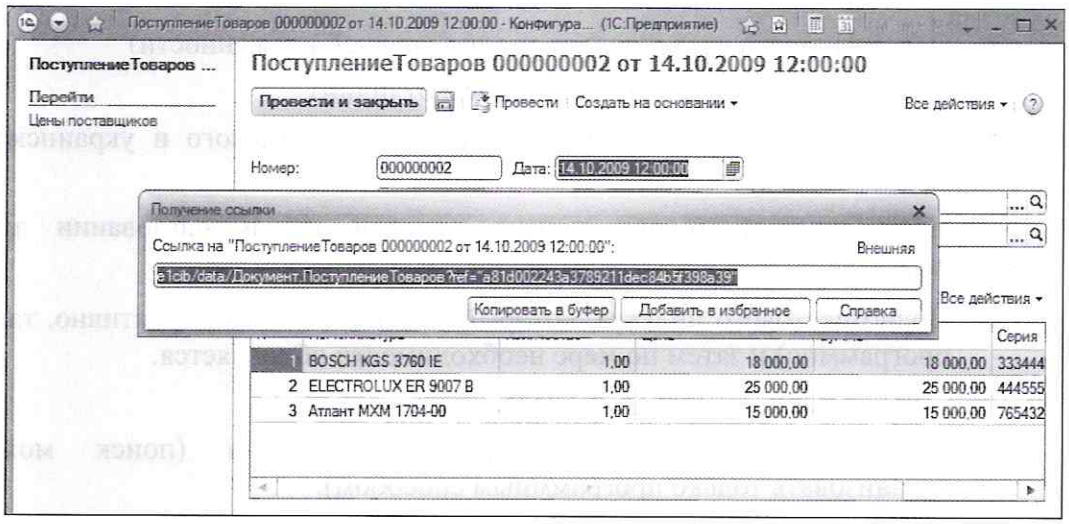
Временное хранилище – это специализированное хранилище информации, в которое может быть помещено значение. Основное назначение – это временное хранение информации при клиент-серверном взаимодействии до ее переноса в базу данных.

Можно использовать временное хранилище как универсальное хранилище с контролируемым временем жизни данных. Продолжительность хранения зависит от продолжительности жизни формы, к которой данные отнесены. После удаления объекта формы в хранилище удаляются принадлежащие ей данные.

Код, "ответственный" за отображение картинки (при открытии формы), выглядит следующим образом:

```
&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    АдресКартинки=ПолучитьНавигационнуюССылку(Объект.Ссылка,"Фотография");

КонецПроцедуры
```

Следует отметить, что механизм навигационных ссылок доступен и в режиме исполнения.



Код, "ответственный" за выгрузку файла картинки из базы данных в файловую систему клиента, приведен ниже.

```
&НаКлиенте

Процедура Выгрузить(Команда)

    АдресКартинки=ПолучитьНавигационнуюССылку(Объект.Ссылка,"Фотография");

    ПолучитьФайл(АдресКартинки,Объект.ИмяФайла,Истина);

КонецПроцедуры
```

Проверьте механизм на практике.

5.3. Механизм полнотекстового поиска

Область применения:

- Когда сложно сформулировать точные условия поиска
- Неизвестно, что искать.
- Неизвестно, где искать, когда традиционные методы не работают.
- Поиск в больших текстовых полях.
- Поиск в "хранилище значения".
- Поиск на естественном языке.

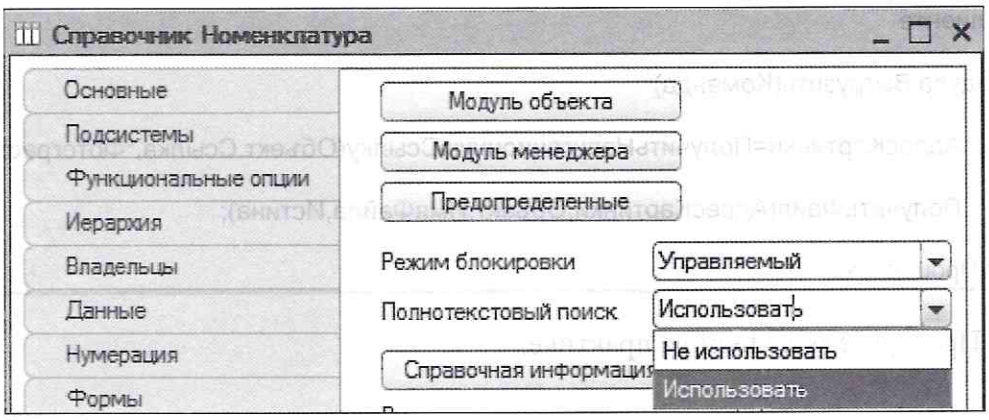
Механизм полнотекстового поиска в данных системы "1С:Предприятие 8" позволяет осуществлять поиск в базе данных с указанием поисковых операторов: И, ИЛИ, НЕ, РЯДОМ. Можно использовать:

- /дистанция по номерам слов (Иванов/2 Иванович)
- (только в конце слова)
- "" (точное соответствие выражению)
- () (группировка слов, сколько угодно уровней вложенности)
- #Число (нечеткий поиск, с указанием отличий)
- ! (Поиск с учетом синонимов русского, английского и украинского языков.)

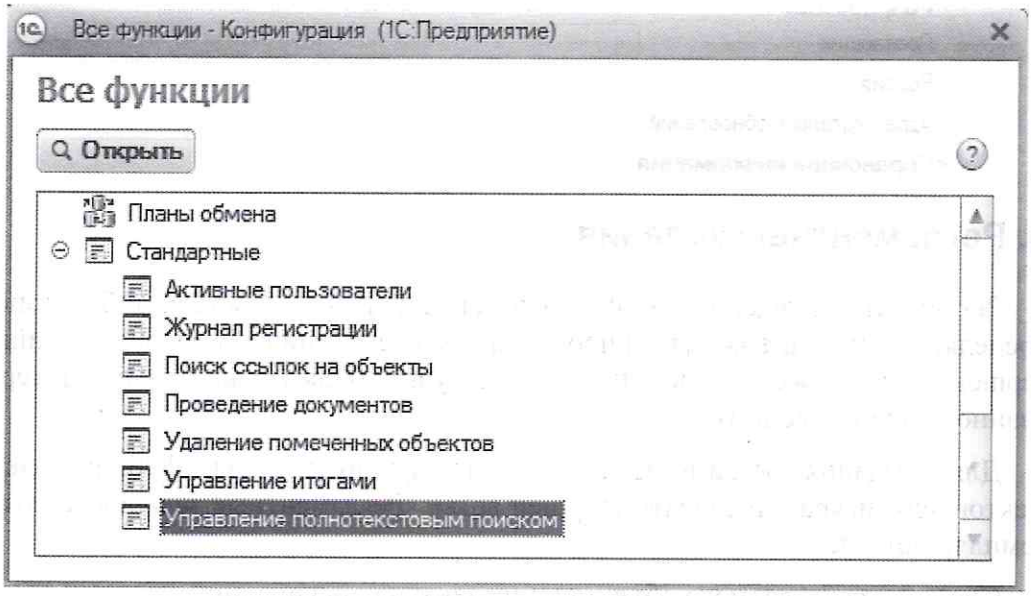
Механизм полнотекстового поиска основан на использовании трех составляющих:

- Полнотекстового индекса, который создается (как интерактивно, так и программно) и затем по мере необходимости обновляется.
- Журнал полнотекстового поиска.
- Средств выполнения полнотекстового поиска (поиск можно организовать только программным способом).

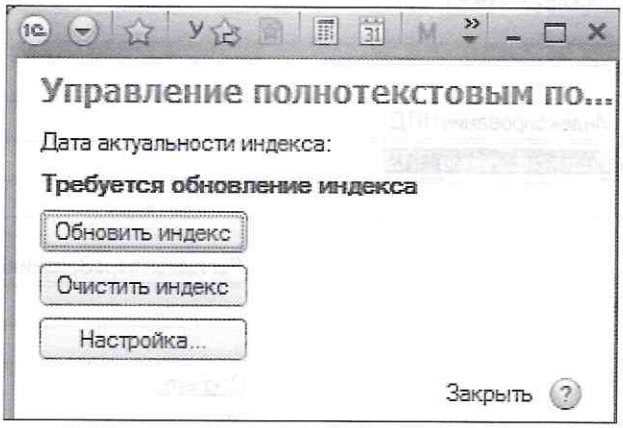
Для включения/исключения данных объекта (наборов записей) из механизма построения индекса используется соответствующее свойство объекта конфигурации (реквизита, реквизита табличной части):



Перед использованием данного механизма необходимо удостовериться, что он включен в платформе. Выполнить это проверку (и при необходимости включить механизм) можно через пункт главного меню программы "Все функции".



Открывается окно следующего вида:

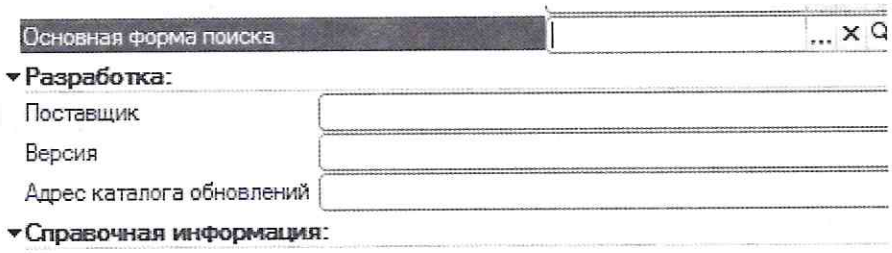


В управлении полнотекстового поиска также можно проводить обновление индекса, но это можно сделать и из встроенного языка:

```
Процедура ОбновитьИндексНажатие(Элемент)

    Пока Не ПолнотекстовыйПоиск.ИндексАктуален() Цикл

        ПолнотекстовыйПоиск.ОбновитьИндекс(Ложь, Истина);

    КонецЦикла;

КонецПроцедуры
```

В интерфейсе "Такси" реализована системная форма для выполнения полнотекстового поиска (данная форма может быть переопределена, для этого в

свойствах корневого объекта дерева объектов конфигурации существует специальное свойство).

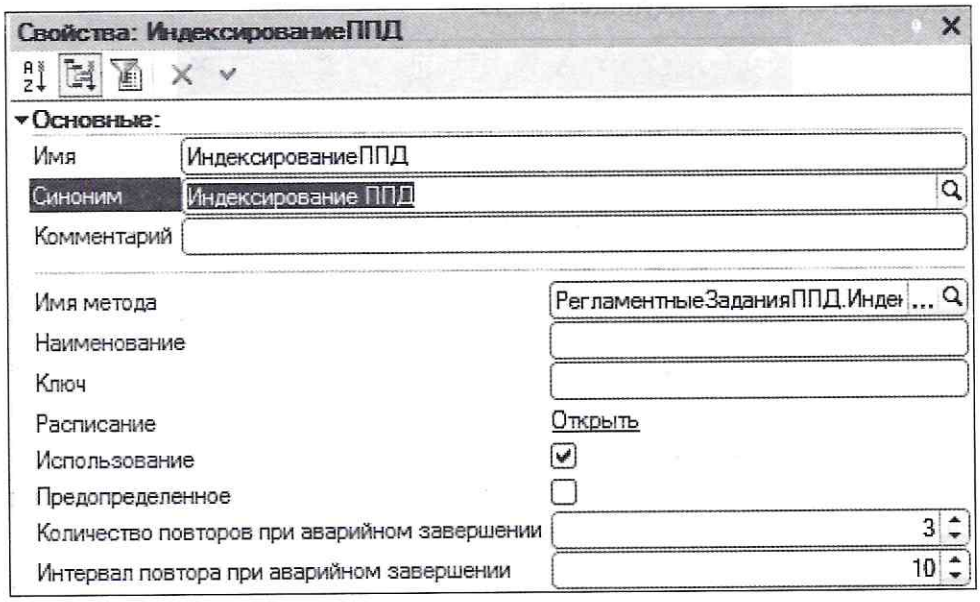


5.4. Регламентные задания

Регламентные задания позволяют организовать запуск каких-либо процедур (определяется на этапе конфигурирования) по расписанию. В случае аварийного завершения возможен перезапуск процедуры (через указанный интервал, указанное количество раз).

Для создания регламентного задания нужно в ветви "Общие" дерева объектов конфигурации сделать текущей ветвь "Регламентные задания" и создать экземпляр объекта.

Свойства создаваемого объекта приведены на рисунке ниже:

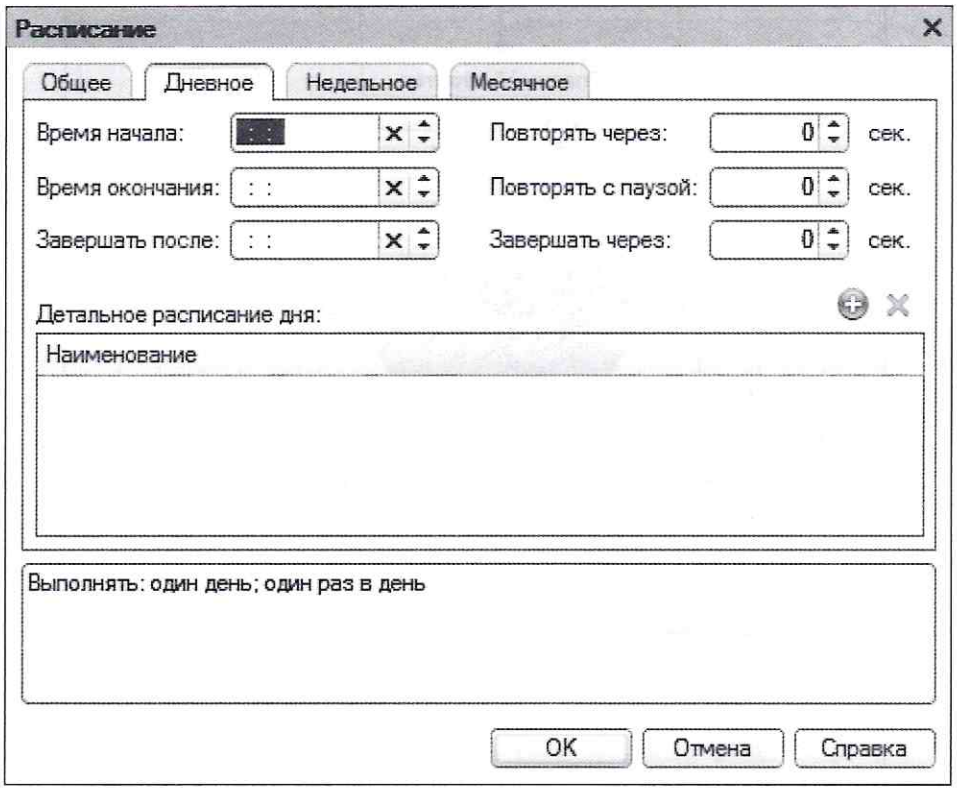


В процедуре, которая используется как метод регламентного задания, производится обновление индекса полнотекстового поиска:

```
Процедура ОбновлениеИндексаППД() Экспорт

    Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =

        РежимПолнотекстовогоПоиска.Разрешить Тогда


        Если Не ПолнотекстовыйПоиск.ИндексАктуален() Тогда

            ПолнотекстовыйПоиск.ОбновитьИндекс(Ложь, Истина);

        КонецЕсли;
```

```
    КонецЕсли;

КонецПроцедуры
```

Нажав на гиперссылку "Открыть", переходим в диалог редактирования расписания.

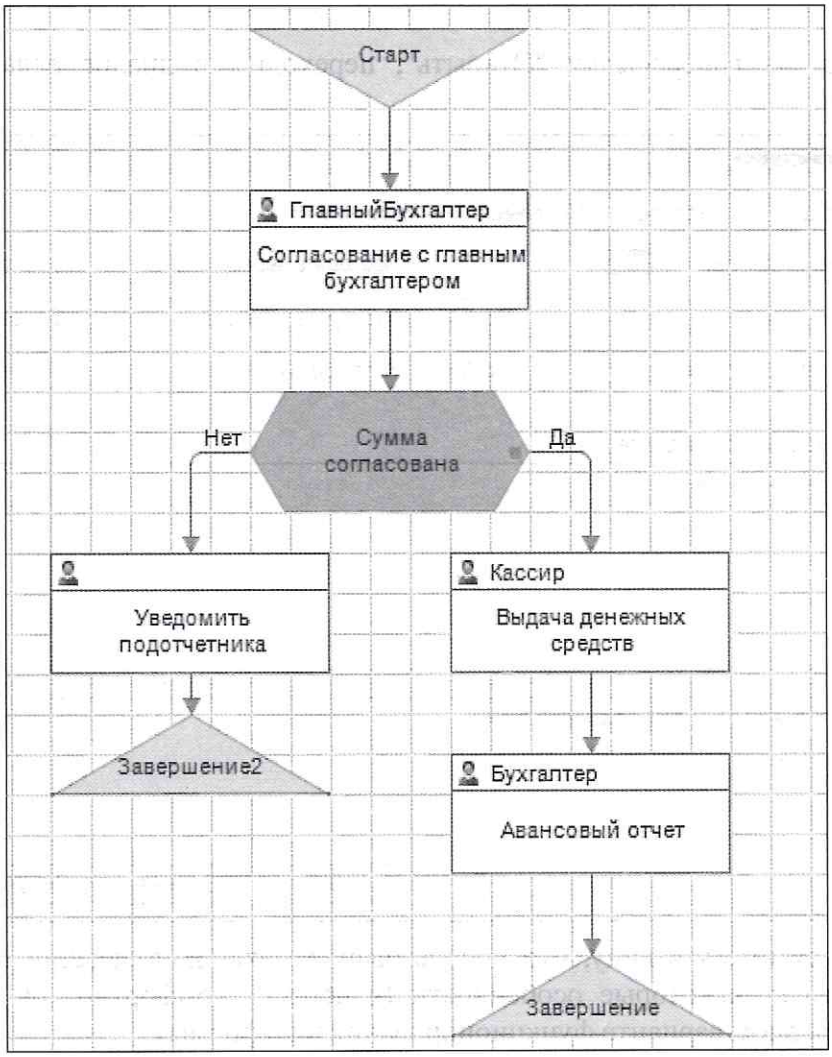


Следует иметь в виду, что использование механизма заданий в файловом варианте имеет некоторые особенности (наиболее "комфортно" он работает в клиент-серверном варианте функционирования платформы).

5.5. Бизнес-процессы, задачи

Бизнес-процессы в "1С:Предприятии 8" предназначены для объединения отдельных операций в цепочки взаимосвязанных действий, приводящих к достижению конкретной цели. Они дают возможность перейти к процессному управлению деятельности компании. Все возможные состояния бизнес-процесса и переходы между ними представляются с помощью карты маршрута бизнес-процесса. Карта маршрута описывает логику бизнес-процесса и весь его жизненный цикл от точки старта до точки завершения, в виде схематического изображения последовательности прохождения взаимосвязанных точек маршрута.

Задачи в "1С:Предприятии 8" позволяют вести учет заданий по исполнителям и служат отражением продвижения бизнес-процессов по точкам маршрута. При этом задачи могут создаваться не только бизнес-процессами, но и другими объектами информационной базы, и непосредственно пользователями.



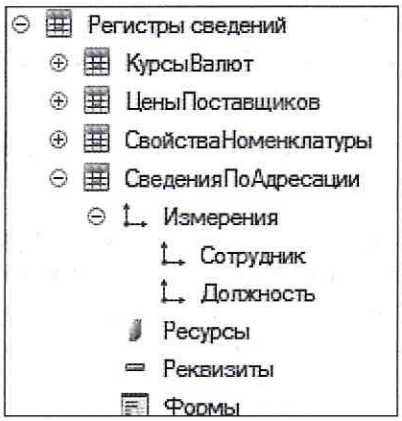
Бизнес-процессы в "1С:Предприятии" допускают следующие виды маршрутизации:

- Жесткая. Бизнес-процесс имеет строгую карту маршрута, не включающую в себя условных и параллельных переходов, с жестко определенными адресатами для каждой точки маршрута. Данный вид не допускает свободной и условной маршрутизации.
- 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.

Другим важным понятием является "Адресация". Основное назначение системы адресации - обеспечить возможность не только персональной, но и ролевой адресации задач участникам бизнес-процессов.

Для хранения данных об адресации используется регистр сведений.



Для отображения текущего состояния бизнес-процесса может использоваться следующий фрагмент кода:

```
&НаКлиенте

Процедура ОбновитьКарту(Команда)

    ОбновитьКартуСервер();

КонецПроцедуры


&НаСервере

Процедура ОбновитьКартуСервер()

    КартаМаршрута = Объект.Ссылка.ПолучитьОбъект().ПолучитьКартуМаршрута();

КонецПроцедуры
```

Предполагается, что этот код размещается в модуле формы бинес-процесса.

6. Большая самостоятельная работа

Автоматизируемая нами фирма занимается закупками у своих поставщиков и продажей своим покупателям различных товаров. В качестве дополнительной услуги существует бесплатная доставка купленных товаров в случае, если общая сумма заказа превышает 1000 рублей.

Необходимо в рамках нашей конфигурации создать отдельную ветвь учета использования транспорта организации. Должен быть реализован следующий функционал:

Должен вестись перечень транспортных средств организации.

В начале дня на каждую бригаду (а бригада состоит из водителя и двух грузчиков) оформляется документ. Этот документ определяет состав бригады (он может меняться произвольным образом) и производит допуск к работе (в документе должны быть отметки о допуске водителя врачом к рейсам и отметка о прохождении инструктажа по технике безопасности). Этим же документом бригада "прикрепляется" к определенной автомашине, при этом указывается начальное значение счетчика спидометра.

Далее, при оформлении документа "ПродажаТоваров", в случае, если сумма покупки превышает 1000 рублей, должно выдаваться сообщение о возможности предоставления бесплатной доставки, и только в этом случае менеджер может выписать на основании расходного документа документ "Заявка на транспорт". В данном документе указывается покупатель, контактное лицо (в диалоге должен быть виден телефон), дата и время доставки (оно может быть любым, но не раньше текущей даты). Документ не имеет табличной части, но хранит ссылку на документ основание.

Сотрудник транспортного отдела рассматривает документ заявку, выбирает машину. Если на эту машину не определена бригада, выдается предупреждение и производится сброс выбранного значения. В противном случае автоматически в документ записывается водитель и грузчики. Далее заявка печатается. В печатной форме документа должна присутствовать информация об адресе доставки (данные четко привязаны к контактному лицу), перечне доставляемых товаров.

Кроме всего, в документе "Заявка на транспорт" проставляется текущее состояние заказа ("не выехали", "в дороге к клиенту", "у клиента", "в дороге обратно" и "отработан") и километраж (расстояние в километрах "туда и обратно").

На основании всей этой информации необходимо видеть: какая машина где находится; на какой машине какие бригады за выбранный период работали; какой водитель в скольких доставках, с каким общим километражем участвовал.

Кроме этого, необходимо за период получать контрольный отчет по машине: состояние счетчика на начало каждого дня, все поездки (километраж), расчетное состояние счетчика.

Приступайте……

В качестве подсказки можно предложить один из вариантов организации структуры базы данных:

1. Необходимо создать справочники "ТранспортныеСредства", "Водители" (хранит ссылку на справочник "ФизическиеЛица"), "Грузчики" (хранит ссылку на справочник "ФизическиеЛица").

2. Необходимо модифицировать документ "ПродажаТоваров" (при превышении суммы закупки 1000 рублей должно выводиться сообщение).
3. Создайте документ "Формирование бригады", "Заявка", настройте их. Документ "Заявка" должен заводиться на основании документа "ПродажаТоваров".
4. Для хранения состава бригады, назначенной машины, начального значения счетчика можно использовать регистр сведений. Другой регистр сведений можно использовать для отслеживания "состояния" бригад.
5. Для накопления данных о количестве выездов, километраже можно использовать регистр накопления

Заключение

Что можно посоветовать для достижения более эффективного результата в обучении конфигурированию и программированию в среде "1С:Предприятие 8"?

Во-первых, пройти эту методику еще раз самостоятельно, но на этот раз в спокойном темпе, не пропуская ни одного практикума. Это будет уже намного проще, так как учебная конфигурация, полученная Вами в результате обучения, поможет Вам вспомнить особенности работы с изучаемыми объектами.

Во-вторых, имеет смысл продолжить изучение методов встроенного языка и типовых конфигураций на наших курсах.

В-третьих, применять полученные знания на практике. Ничто так не укрепляет знания, как решение реальных задач.

Успехов в работе!

Заказ М-2018