Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информационные технологии»

Л. К. Титова

РАЗРАБОТКА УЧЕТНЫХ ПРИЛОЖЕНИЙ В 1С:ПРЕДПРИЯТИЕ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
для студентов специальности 1-40 05 01
«Информационные системы и технологии
(по направлениям)», направление специальности
1-40 05 01-01 «Информационные системы и технологии
(в проектировании и производстве)»
дневной и заочной форм обучения

Гомель 2022

УДК 004.4(075.8)
ББК 32.972я73
Т45

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 10 от 25.06.2021 г.*)

Рецензент: доц. каф. «Информатика» ГГТУ им. П. О. Сухого
канд. экон. наук, доц. *Н. В. Ермалинская*

Титова, Л. К.

Т45 Разработка учетных приложений в 1С:Предприятие : учеб.-метод. пособие для студентов специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)», направление специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)» днев. и заоч. форм обучения / Л. К. Титова. – Гомель : ГГТУ им. П. О. Сухого, 2022. – 150 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Содержит лекционный материал по учебной дисциплине «Разработка учетных приложений в 1С:Предприятие», который знакомит с основополагающими концептуальными решениями платформы 1С:Предприятие и основными приемами разработки несложных учетных приложений.

Для студентов специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)», направление специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)» дневной и заочной форм обучения.

УДК 004.4(075.8)
ББК 32.972я73

ВВЕДЕНИЕ

Система программ 1С:Предприятие предназначена для автоматизации деятельности предприятий, организаций и частных лиц. Она включает в себя платформу и прикладные решения, работающие на ее основе. Одним из основных этапов автоматизации является разработка прикладного решения.

1С:Предприятие 8.3 – это система прикладных продуктов, предназначенных для эффективного решения разнообразных задач управления предприятием. Благодаря своей универсальности и гибкости, она легко настраивается для нужд конкретной организации и позволяет решать широкий круг задач автоматизации любых разделов учета на предприятии, а также вести учет по нескольким организациям в одной информационной базе. Дальнейшее развитие концепции управляемого приложения и быстрой разработки систем поддержки принятия решений в конкретной предметной области предложено в новой версии 8.3. В соответствии с конкретной областью деятельности фирма «1С» разработала в составе системы 1С:Предприятие 8 несколько прикладных программных продуктов. Наиболее известными из них являются программы для управления бухгалтерским, торговым и кадровым учетом: «Бухгалтерия предприятия», «Управление торговлей», «Зарплата и управление персоналом». Любая конфигурация может быть установлена на компьютер как сама по себе, так и вместе с другими конфигурациями.

Особенностью системы 1С:Предприятие 8.3 является ее конфигурируемость. Система представляет собой совокупность объектов предметной области и механизмов, предназначенных для манипулирования этими объектами.

Конфигурация представляет собой систему, включающую набор объектов, структур информационных массивов, алгоритмов обработки информации, соответствующих решаемым задачам. На этапе конфигурирования производится формирование структуры учитываемой информации, создание форм, предназначенных для ввода исходных данных и просмотра различных списков данных, организуется структура для хранения исходной и итоговой информации, выполняется создание описаний отчетов и обработок, формируются интерфейсы для различных групп пользователей, определяется список пользователей и их прав.

На этапе конфигурирования система оперирует такими объектами, как Документ, Журнал документов, Справочник, Реквизит, Форма, Регистр и другими. Результатом конфигурирования является конфигурация, которая представляет собой информационную модель предметной области.

В режиме конфигурирования можно создавать новые конфигурации, редактировать ранее созданные, выполнять сравнение и объединение нескольких конфигураций.

Учебный план направления специальности «Информационные системы и технологии (в проектировании и производстве)» предполагает изучение дисциплины «Разработка учетных приложений в 1С:Предприятие». В результате изучения дисциплины студенты должны формировать и решать типовые задачи, используемые при принятии управленческих решений в автоматизированных информационных системах управления, разрабатывать алгоритмы реализации задач по управлению в автоматизированных информационных системах управления, проектировать структуру базы.

1 ОБЩИЕ СВЕДЕНИЯ О СИСТЕМЕ «1С:ПРЕДПРИЯТИЕ»

1.1 Составляющие системы программ «1С:Предприятие»

Система программ «1С:Предприятие» предназначена для решения широкого спектра задач автоматизации учета и управления, стоящих перед динамично развивающимися современными предприятиями.

«1С:Предприятие» представляет собой систему прикладных решений, построенных по единым принципам и на единой технологической платформе.

Программный продукт 1С состоит из двух составляющих – ***платформы и конфигурации***.

Платформа является базисом, который позволяет устанавливать конкретную конфигурацию 1С и уже после этого – работать с нею.

Конфигурация – та самая программа, в которой работают пользователи и разработчики компании.

Многим хорошо известно, что существует множество разновидностей программ 1С.

Среднестатистический пользователь, скорее всего, назовет три из них:

- 1С:Бухгалтерия;
- 1С:Зарплата и управление персоналом;
- 1С:Управление торговлей.

На самом деле, компанией «1С» разработано более тысячи различных пользовательских приложений, при этом внедряется «1С:Предприятие» фактически на каждом предприятии.

Все эти программы называются ***конфигурациями*** или ***прикладными решениями 1С***.

Важно знать и правильно определиться, как выбрать конфигурацию 1С, максимально подходящую для удовлетворения потребностей определенной фирмы.

Конфигурации 1С (прикладные решения 1С) – это программы, предназначенные для автоматизации деятельности различных организаций и частных лиц.

Конфигурация в 1С запускается только в том случае, если на компьютере установлена технологическая платформа 1С:Предприятие.

Технологическая платформа 1С:Предприятие – это специальная среда или оболочка, в которой запускаются и функционируют прикладные решения 1С.

При покупке 1С пользователь приобретает комплект программ, состоящий из платформы 1С:Предприятие и одной или нескольких конфигураций 1С.

Такой «комплект» (конфигурирование платформы и рабочих баз с программными инструментами управления) принято называть ***программным продуктом 1С***.

В программный продукт также включено консультационное и технологическое сопровождение.

Например, предоставляется доступ к справочной системе Информационно-технологическое сопровождение (1С:ИТС).

Примеры программных продуктов на базе платформы версии 8.3:

– программный продукт = платформа 1С:Предприятие 8.3 + 1С:Бухгалетрия 8.3 + 1С:Зарплата и управление персоналом 8.3 (для ведения бухгалтерского, налогового учета производственного предприятия и начисления зарплаты сотрудникам в отдельной программе).

– программный продукт = платформа 1С:Предприятие 8.3 + 1С:Бухгалетрия 8.3 + 1С:Управление торговлей 8.3+1С:Зарплата и управление персоналом 8.3 (для ведения бухгалтерского, налогового, складского учета торговой организации и начисления зарплаты сотрудникам в отдельной программе).

Все конфигурации 1С имеют похожий интерфейс, одинаковые объекты конфигурации (справочники, документы, регистры сведений и т.д.) и общие принципы работы. Таким образом, пользователь, освоивший основные действия в одной из конфигураций 1С, может с легкостью работать в других.

Некоторые из однотипных операций, доступных во всех прикладных решениях 1С:

– заполнение справочников. Создание элементов и групп в справочниках;

– удаление, копирование, перемещение, редактирование элементов и групп справочников;

– ввод входящих остатков;

– ввод документов в программе, в т.ч. создание документов путем копирования и ввод на основании;

– работа в журналах документов;

– формирование отчетов по итогам работы.

Технологическая платформа 1С:Предприятие разработана компанией «1С». Она постоянно развивается, учитывая потребности пользователей, обновления законодательства, а также новшества рынка. В результате, на свет постоянно появляются новые версии и релизы (текущие обновления) платформы 1С.

Кроме того, платформа содержит встроенный язык программирования, позволяющий внести изменения в готовую конфигурацию на основании пожеланий заказчика. Иногда, если это необходимо, на базе технологической платформы пишутся «с нуля» совершенно новые конфигурации для 1С.

Прикладные программы 1С создаются как самой фирмой «1С», так и другими разработчиками, фирмами-партнерами.

Конфигурации 1С, выпущенные непосредственно компанией «1С» называются типовыми.

Таким образом, в зависимости от разработчика, конфигурации 1С бывают двух видов: типовые и нетиповые (также называются отраслевыми и специализированными решениями).

Рисунок 1.1 – Конфигурации 1С

Большинство пользователей в составе программного продукта приобретают типовые решения 1С.

Достоинства типовых решений 1С. Типовые решения 1С являются универсальными, т.е. подходят для ведения учета в различных сферах деятельности. Например, в 1С:Бухгалтерии могут работать бухгалтеры производственных предприятий, сферы услуг,

торговых организаций. Конфигурация позволяет также вести учет в различных налоговых режимах.

Типовые конфигурации 1С постоянно совершенствуются компанией «1С», которая ведет мониторинг пожеланий покупателей и учитывает опыт широкого круга пользователей. Такие прикладные решения тщательно «отлажены», более надежны в использовании и обслуживании.

Недостатки типовых решений 1С. Потребитель использует лишь нужную ему часть возможностей типового решения, покупая при этом весь функционал программы.

Типовая конфигурация 1С нуждается в тщательной настройке под конкретную организацию, а иногда и в «доработке» силами программистов.

Фирма «1С» предлагает следующие типовые конфигурации:

Рисунок 1.2 – Типовые конфигурации для хозрасчетных и бюджетных организаций

Нетиповые конфигурации 1С. Внедрением программных продуктов 1С занимаются фирмы-партнеры. Они взаимодействуют непосредственно с заказчиками, занимаясь установкой программ 1С, их настройкой и адаптацией, ориентируясь на особенности конкретного предприятия и пожелания заказчика. Для этого IT-специалисты на специальном языке программирования «дорабатывают» типовые конфигурации 1С. Например, прикладное решение «1С:Лесозавод 8» представляет собой модифицированную типовую конфигурацию 1С:Управление производственным предприятием. Создают новые конфигурации на базе платформы 1С:Предприятие.

Прикладные решения, разработанные фирмами-партнерами «1С» не являются типовыми.

Для внедрения таких конфигураций необходимо пройти сертификацию в фирме «1С» и получить право на специальный логотип «1С:Совместимо».

Нетиповые конфигурации, как правило, «пишутся» под конкретный вид деятельности, для определенной отрасли.

Поэтому они называются также *отраслевыми и специализированными решениями 1С.*

Преимущества нетиповых конфигураций:

– позволяют сократить издержки потребителей при внедрении программного продукта за счет того, что поставляются в качестве готовых решений.

– содержат узкоспециализированные решения, которые учитывают специфику работы конкретной компании;

– позволяют избежать сложной настройки конфигурации.

Примеры отраслевых конфигураций: 1С:Общепит, 1С:Управление сельхозпредприятием, 1С:Управление строительной организацией.

Предположим, необходимо подобрать программный продукт для ведения бухгалтерского и налогового учета на предприятии пищевой промышленности.

Для этой цели можно установить и адаптировать типовую конфигурацию 1С:Бухгалтерия или выбрать одно из готовых отраслевых решений, сэкономив при этом время и средства, как показано на рисунке 1.3.

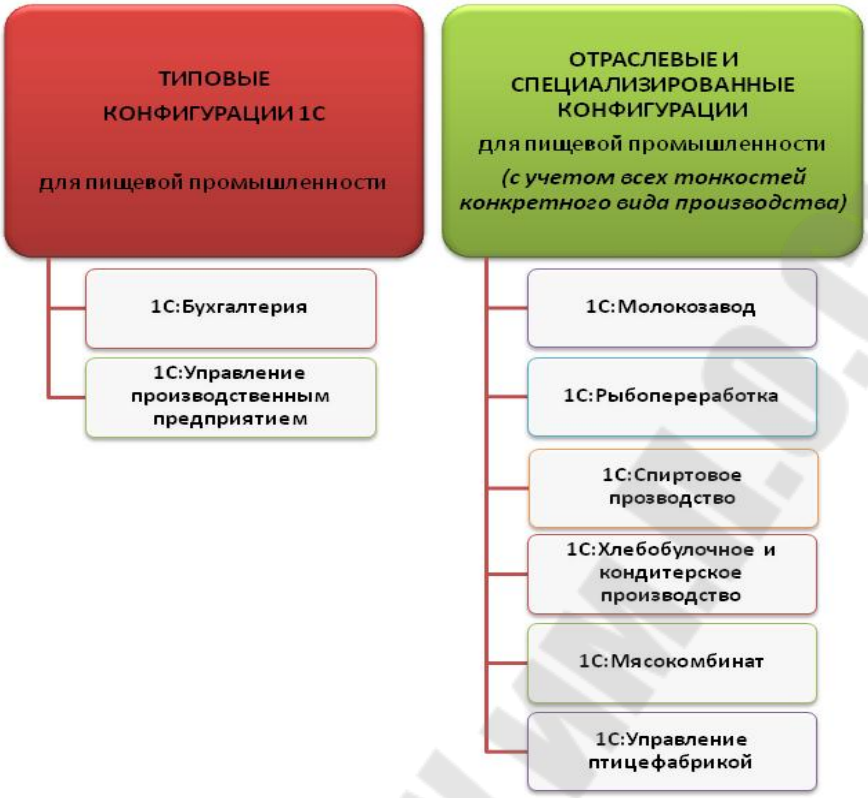


Рисунок 1.3 – Подбор программного продукта для ведения бухгалтерского и налогового учета на предприятии

Как видно, компания «1С» представляет широкую линейку программных продуктов 1С, отвечающую запросам и вкусам самых разных потребителей.

Одним из существенных преимуществ четкого разграничения между платформой и бизнес-приложением является высокий уровень адаптируемости решений под требования клиента.

Следует заметить, что именно для экономических задач особо важна возможность эффективного изменения готового решения человеком, не участвовавшим в его создании.

В индустрии систем управления бизнесом, в отличие от многих других областей, существенная часть программистов не пишет программное обеспечение (ПО) «с чистого листа», а дорабатывает и развивает типовые решения.

Это обстоятельство определяет особые требования к обеспечению наглядности и простоте понимания разработчиком существующих решений и максимально учитывается во всех механизмах платформы.

Разные версии платформы позволяют запускать разные релизы программных продуктов.

Поэтому, если не устанавливать последнюю выпущенную версию платформы 1С, то не получится обновить, например, 1С:Бухгалтерия или 1С:Управление торговлей.

Это крайне нежелательно, так как в каждом новом релизе специалисты 1С устраняют недостатки, учитывают новые законы и оптимизируют работу программы.

Чтобы идти в ногу со временем, необходимо своевременно обновлять и платформу, и конфигурации.

1.2 Обзор архитектуры платформы

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

Одновременно с этим платформа решает и традиционные задачи, связанные с производительностью, эргономикой, функциональностью и т.д.

В системе «1С:Предприятие» существует четкое разделение на ***платформу и прикладное решение***.

Платформа представляет собой *framework*, в котором функционирует прикладное решение:

– платформа служит фундаментом для построения прикладных решений,

– платформа является средой их исполнения,

– платформа содержит инструментарий, необходимый для разработки, администрирования и поддержки прикладных решений.

Прикладное решение является самостоятельной сущностью и может выступать в качестве отдельного программного продукта. Но полностью опирается на технологии платформы.

Возможности 1С на платформе 8.3:

– способ описания прикладного решения – метаданные;

– средства формирования запросов;

– динамические выборки;

– поддержка в полях таблиц составных типов данных;

– штатная реализация функции формирования многомерных итогов с произвольным порядком обхода измерений;

– виртуальные таблицы;

– построение прикладного решения на основе модели;

– стандартные прототипы прикладных объектов;

– согласованность технологий и инструментов;

– многозвенная архитектура работы;

– отказоустойчивый кластер с балансировкой нагрузки;

– высокоуровневая модель интерфейса.

Метаданные – способ описания прикладного решения.

Метаданные 1С – это список справочников, документов, их реквизитов и прочего, составляющего конфигурацию.

Из программы на языке 1С доступен список метаданных 1С и их свойства.

Это удобно, когда требуется получить список документов в конфигурации или отработать все реквизиты документа.

Метаданные – способ описания бизнес-приложения.

Метаданными описываются структуры данных, состав типов, связи между объектами, особенности их поведения и визуального представления, система разграничения прав доступа, пользовательский интерфейс и т. д.

В метаданных сосредоточены сведения не только о том, что хранится в базе данных, но и о том, зачем хранится та или иная информация, какова ее роль в системе и как связаны между собой информационные массивы.

Глобальная переменная Метаданные дает доступ к метаданным 1С всей конфигурации (т.е. к «корню» конфигурации). Далее через точку можно обращаться к «коллекциям» – документы, справочники и т.д., чтобы выйти на конкретный справочник или документ, у которых соответственно есть «коллекции» реквизиты, табличные части – все, как в конфигураторе, в дереве конфигурации.

Прикладное решение не пишется в прямом смысле на языке программирования. Язык программирования используется только там, где это действительно необходимо.

Что дает такой подход к построению бизнес-приложения?

Во-первых, при описании метаданных широко применяется визуальное редактирование. Это позволяет свести существенную часть

разработки к визуальному проектированию, не требующему кропотливого написания кода.

Однако у данного подхода есть и другие не менее важные преимущества. Описывая прикладное решение в терминах метаданных, разработчик «сообщает» платформе много очень полезной информации, которую та может эффективно использовать в самых различных целях.

На основе метаданных система автоматически выстраивает большую часть механизмов и объектов, отвечающих за функционирование прикладного решения.

Так, описания метаданных платформе достаточно для того, чтобы автоматически сформировать пользовательский интерфейс системы, обеспечивающий ввод и редактирование взаимосвязанной информации.

Другой пример – возможность построения даже конечным пользователем, не имеющим навыков программирования, весьма сложных отчетов.

Идеология метаданных в самых общих словах сводится к простому тезису: «Давайте не будем программировать все функции разрабатываемого решения. Расскажем платформе о составе, структуре, особенностях и взаимосвязи различных его частей, и пусть остальное она сделает сама».

Эта идеология (*Metadata Driven*) сегодня находит все более широкое применение во многих перспективных разработках.

Использование языка программирования ограничено решением тех задач, которые действительно требуют алгоритмического описания, например, расчета налогов, проверки корректности введенных данных и т.д.

Наряду с описанными выше методами манипулирования данными и формирования *запросов*, система предлагает еще один способ доступа к данным – ***динамические выборки***.

Этот механизм позволяет обращаться к очень большим объемам данных, обеспечивая считывание информации порциями. При этом разработчик только указывает, какие данные необходимо получать, а система автоматически выполняет обращения к базе данных (БД) с необходимой гранулярностью.

Важно, что для решения этой задачи не используются какие-либо специфические средства динамического считывания конкретной СУБД, требующие удержания в памяти открытой выборки, а

осуществляется автоматическое формирование запросов, последовательно выбирающих блоки записей.

Еще одним значимым решением в части работы с данными в «1С:Предприятии» является ***поддержка в полях таблиц составных типов данных.*** Эта возможность не имеет аналогов в других системах.

При описании типа поля какого-либо объекта можно выбрать не только один из доступных типов, но и практически любую (с некоторыми ограничениями) их комбинацию.

Так, в поле «Плательщик» в документе, отражающем операцию с банком, допускается хранение ссылки на юридическое или физическое лицо в зависимости от конкретной операции.

Хотя приведенный пример достаточно прост, в реальных приложениях возможность работы с составными типами позволяет решать такие задачи, как хранение произвольных характеристик товаров, ведение аналитического учета на бухгалтерских счетах по любому составу аналитических разрезов, настраиваемых пользователем, и т. д.

Надо отметить, что система не просто предоставляет возможность хранения в одном поле разнородных значений, а делает это прозрачным для разработчика способом.

Также необходимо отметить полную поддержку работы с полями составных типов «движка» базы данных и языка запросов.

Для этих полей поддерживается весь набор стандартных операций (сравнение, агрегирование и т. д.).

Другим важным моментом является поддержка составных типов в интерфейсных механизмах системы. Например, поле ввода, связанное с данными такого составного типа, предоставляет весь набор функций редактирования (выбор типа; редактирование значений всех типов, входящих в указанное поле; ограничение выбираемых типов).

Отдельного внимания заслуживают также ***средства формирования запросов*** «1С:Предприятия».

Они основаны на конструкциях стандартного языка *SQL*, но имеют ряд существенных расширений.

Прежде всего, следует отметить поддержку в языке запросов объектов, хранящихся в БД. Все операторы языка запросов обеспечивают работу со ссылочными типами (полями, хранящими

ссылки на объекты БД). Например, поддерживается обращение к полям в нотации «через точку» без ограничения количества уровней.

Можно указать в запросе выборку такого поля, как:

«Счет. Товар. Производитель. Страна. Наименование».

Для объектов базы данных допускается обращение к вложенным таблицам и как к отдельным таблицам, и как к обычным полям объекта, содержащим коллекции записей.

Другим важным качеством является штатная реализация функции формирования многомерных итогов с произвольным порядком обхода измерений.

При этом доступны такие возможности, как сочетание многомерного обхода измерений и многоуровневого обхода иерархии значений каждого измерения (например, многоуровневой структуры подразделений или многоуровневой группировки товаров).

Поддерживается также ряд специальных режимов обработки итогов.

Механизм запросов способен на основании свойств прикладных объектов, определенных в метаданных, упорядочивать выборки автоматически.

Это позволяет разработчику при создании запроса, предназначенного для получения отчета или визуализации данных, не указывать, по каким полям производить упорядочивание, а, включив автоматический режим, получить стандартную для выбираемых данных сортировку.

Еще одним мощным элементом механизма запросов являются ***виртуальные таблицы***.

Они обеспечивают доступ к производным данным, предоставляемым различными прикладными подсистемами, не требуя для этого написания сложных запросов.

Например, можно обратиться к виртуальной таблице для получения распределения остатков и оборотов товаров по складам и номенклатуре, а также по календарным периодам.

Следует отметить, что работа с виртуальными таблицами не является аналогом простого хранения типовых запросов (*view*). При использовании виртуальных таблиц разработчик указывает набор параметров, описывающих необходимую выборку, беря в их качестве не только конкретные значения, но и, например, сложные условия.

Разработчик прикладного решения работает с виртуальной таблицей практически так же, как и с обычной, но система формирует

запрос к БД таким образом, чтобы обеспечить максимальную эффективность. В частности, при обращении к данным, обрабатываемым учетными механизмами, могут использоваться хранимые ими промежуточные итоги.

В платформе заложена ориентация на ***построение прикладного решения на основе определенной модели***.

Под моделью понимается вся идеология построения прикладного решения. Сюда относятся способы построения структур данных, типы связей между данными, принципы манипулирования данными, формы описания бизнес-логики, способы связи данных с интерфейсными объектами, разделение функциональности по уровням системы и многое другое.

Важно, что все прикладные решения следуют принятой модели и этим обеспечивается единообразие и предсказуемость их поведения.

Стандартные прототипы прикладных объектов

В модели разработки "1С:Предприятия" все прикладное решение описывается метаданными в виде совокупности прикладных объектов, выбираемых из определенного набора прототипов (классов).

Каждый такой прототип отвечает за отражение в прикладном решении определенной совокупности объектов или процессов предметной области, имеющих схожие поведенческие характеристики и сходную роль в общей картине решения.

Примерами таких прототипов являются «Справочники», «Документы», «Регистры накопления».

Каждый прототип имеет некоторую базовую реализацию, которая определяет особенности функционирования создаваемых на основе данного прототипа объектов: структуру хранимых сущностей вместе с некоторыми предопределенными полями, набор типов языка программирования, методы, свойства и события, а также типовые для решаемой задачи операции, способы отображения и редактирования, методы регулирования прав доступа и т.д.

Таким образом, все прикладное решение, фактически, состоит из объектов, четко разделенных по тем ролям, которые они играют в прикладном решении.

Такой подход существенно усиливает эффект и от описания системы в терминах метаданных, и от построения приложения на основе модели.

Согласованность технологий и инструментов. Ключевым качеством платформы является достаточность ее средств для решения

задач, стоящих перед прикладными решениями. Это позволяет обеспечить очень хорошую согласованность всех технологий и инструментов, которыми пользуется разработчик.

Технологии			***Инструменты***	
Толстый клиент	*Тонкий клиент*	*Веб-клиент*	*Редактор форм*	*Конструктор запросов*

Многозвенная архитектура работы. Прикладные решения, работающие под управлением платформы, используют многозвенную архитектуру «клиентское приложение – кластер серверов 1С:Предприятия – сервер базы данных».

С одной стороны, это позволяет масштабировать систему от вариантов персонального использования, до работы в крупных, территориально распределенных холдинговых компаниях.

С другой стороны многозвенная архитектура позволяет выбирать между несколькими системами управления базами данных, которые будут использоваться для хранения прикладных данных.

Основные компоненты системы могут работать как под управлением операционной системы *Windows*, так и под управлением операционной системы *Linux*.

Кроме этого клиентская часть 1С:Предприятия может быть запущена и на компьютерах с другими операционными системами, например, *Apple iOS*.

Отказоустойчивый кластер с балансировкой нагрузки. Кластер серверов обеспечивает отказоустойчивость системы к таким событиям как выход из строя серверного компьютера (в том числе и центрального сервера), аварийное завершение рабочего процесса или менеджера кластера, физический разрыв соединения пользователя с кластером и последующее его восстановление.

Это достигается благодаря тому, что существует возможность резервирования как самого кластера, так и рабочих процессов, функционирующих в нем.

Кроме этого кластер предпринимает специальные действия для того, чтобы обеспечивать устойчивость к обрыву канала связи. Кластер автоматически выполняет балансировку нагрузки между своими рабочими процессами на основе актуального анализа их доступной производительности.

При этом администратор системы может корректировать нагрузку на отдельные рабочие серверы как за счет физического ограничения обслуживаемых информационных баз или соединений, так и за счет переназначения части функциональности кластера на другие рабочие серверы.

Высокоуровневая модель интерфейса. Основной идеей построения интерфейса является максимальное использование информации из метаданных, а также объектов манипулирования данными с тем, чтобы вся конструкция не требовала детальной настройки со стороны разработчика и функционировала по большей части автоматически.

Разработчику достаточно связать такой объект с элементом формы или с самой формой, и механизм интерфейса полностью возьмет на себя организацию просмотра и модификации данных.

Платформа автоматически подключит расширения, учитывающие тип данных, с которыми связан элемент управления или форма.

Платформа содержит целый набор механизмов, позволяющих создавать приложения на разных языках. Начиная от различных языков интерфейса платформы, поддержки национальных дат, чисел, и заканчивая средствами редактирования текстов интерфейса.

Веб-клиент и тонкий клиент. Для работы с прикладным решением пользователь может выбрать одно из трех клиентских приложений: толстый клиент, тонкий клиент, либо веб-клиент. Каждое из клиентских приложений обладает своими преимуществами.

Режим толстого клиента – «классический» режим работы системы «1С: Предприятие» при разработке и отладке прикладного решения. В режиме толстого клиента пользователю одновременно доступно как само приложение, со всем интерфейсом, всеми формами, так и база данных, которая содержит в себе всю информацию, выводимую в указанные формы приложения. При этом, поддерживается постоянное соединение между приложением и базой данных.

Тонкий клиент более функциональный за счёт того, что устанавливается на компьютере пользователя. Например, он имеет непосредственный доступ к файловой системе клиентского

компьютера, может использовать локальные лицензии, установленные на компьютере пользователя.

В то же время веб-клиент более универсальный за счёт того, что не требует предварительной установки. Он исполняется не в среде операционной системы компьютера, а в среде интернет-браузера. Поэтому пользователю достаточно всего лишь запустить свой браузер, ввести адрес веб-сервера, на котором опубликована информационная база – и веб-клиент «сам приедет» к нему на компьютер и начнет исполняться.

С помощью веб-клиента можно работать с прикладными решениями на компьютерах, которые заранее не подготовлены (или не могут быть подготовлены) для этого.

Важным фактом является то, что разработка прикладного решения ведется независимо от того, какое из клиентских приложений будет использоваться для работы с приложением. Клиентские модули, разработанные в конфигурации веб-клиент, автоматически компилируют из встроенного языка 1С:Предприятия 8 и непосредственно исполняют на своей стороне.

1.3 Улучшение существующего функционала 1С в релизе 8.3

Специалисты компании 1С в первую очередь занимаются совершенствованием опций уже реализованных в платформе. Именно поэтому функции, которыми пользовались в релизе 8.1 или 8.2, могут существенно отличаться в релизах 8.3.2 или 8.3.3. При этом с каждым релизом идет подробнейшее описание всех изменений функциональности платформы 1С. Техническая поддержка в компании 1С также работает с пользователями круглосуточно и готова ответить на все ваши вопросы.

Основными изменениями в релизе 8.3 считают:

– уникальный, подстраиваемый под конкретного пользователя интерфейс «Такси»;

– мобильная платформа 1С;

– облачные технологии;

– улучшение производительности функционирования и разработки на платформе 1С 8.3;

– оптимизация многих механизмов интерфейса и клиентской части;

– новые инструменты для разработчиков;
– операционна система семейства Linux.

1С Предприятие версии 8.3 претерпела много изменений и в плане новых возможностей интерфейса. Наиболее значимое влияние на популярность версий 8.3 и более поздних оказал новый ***интерфейс «Такси»***.

Интерфейс такси 1С – это новое слово в развитии технологической платформы 1С: Предприятие 8.3.

Новое лицо программы адаптировано под стандарты веб-приложений. Разработчики произвели большое количество юзабилити тестов и серьезно переработали механизмы взаимодействия с пользователями.

Программа теперь напоминает качественно сделанный веб-сайт с дружелюбным интерфейсом. В него заложены принципы максимального «очищения» рабочего стола от лишних меню, укрупненный шрифт и новые подходы к часто используемым функциям.

Огромным преимуществом «Такси» является возможность пользователя самостоятельно изменять интерфейс 1С 8.3, добавляя и убирая функции, иконки, команды и документы.

Возможность перемещать панели позволит сделать интерфейс 1С более лояльным к новым пользователям и ускорить работу уже опытных специалистов.

Теперь интерфейс можно полностью настроить под себя, расположив панели так, как удобно Вам (рисунок 1.4).

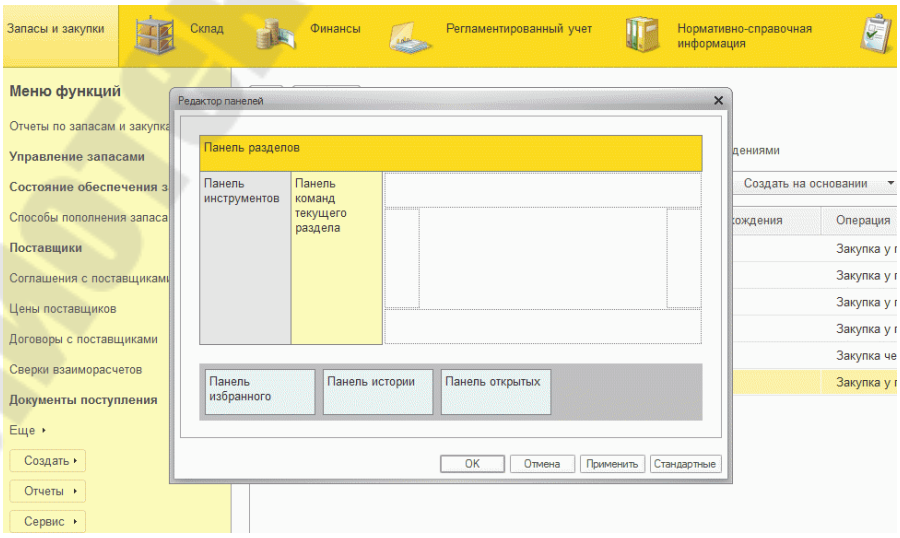


Рисунок 1.4 – Интерфейс такси 1С

Формы конфигурации управляемого приложения (8.2 и 8.3) способны перейти на «Такси» без дополнительных доработок, буквально в одно нажатие мыши.

Однако новый интерфейс такси получился более крупным, и стандартные приложения от 1С, как правило, не предназначены для работы в этом режиме.

В настоящий момент фирма 1С и партнеры активно осуществляют перевод и адаптацию конфигураций для нововведений.

Переход на Такси. Для того чтобы включить или выключить интерфейс Такси, достаточно установить в свойствах конфигурации значение «Режим совместимости интерфейса» режим «Такси».

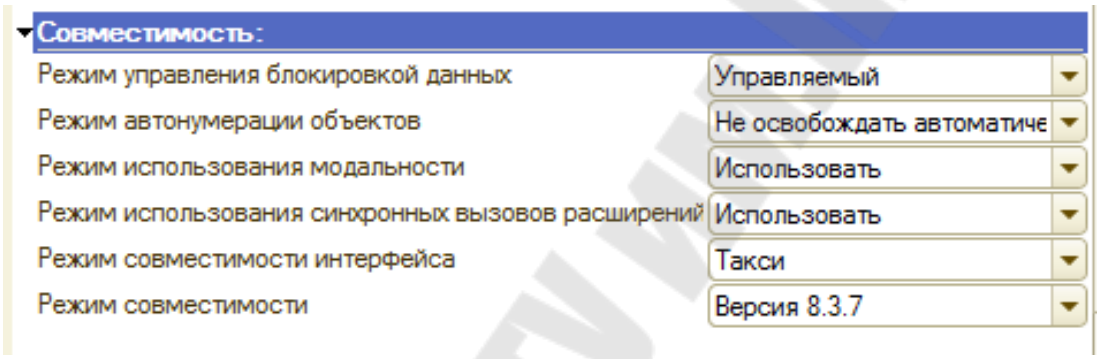


Рисунок 1.5 – Включение режима «Такси»

Причем в любой момент Вы можете вернуться к обычному интерфейсу.

Для работы на смартфонах и планшетах предоставляется специальная технология – ***мобильная платформа***. С её помощью создаются приложения, которые автономно функционируют на этих устройствах.

Основное назначение таких приложений заключается в том, чтобы быть частью распределенной информационной системы, обмениваться данными с основным приложением. А в те периоды, когда связь с основным приложением невозможна, обеспечивать полноценную автономную работу.

Разработка и отладка мобильных приложений ведётся теми же инструментами, в тех же терминах и понятиях, что и разработка обычных, «настольных» приложений. Отличие заключается лишь в том, что в конце, для получения дистрибутива мобильного приложения нужно выполнить ряд дополнительных действий, «собрать» мобильное приложение.

Облачные технологии. Прикладные решения 1С:Предприятия способны функционировать в облаке, благодаря набору технологий и механизмов.

Базисом для облачных технологий является механизм разделения данных, реализованный в платформе. Благодаря ему прикладные решения могут работать в архитектуре multitenancy (мультиарендности), когда единый экземпляр объекта приложения, запущенного на сервере, обслуживает множество клиентов или организаций.

Другим важным элементом является отказоустойчивый масштабируемый кластер серверов, обслуживающий большое количество одновременно работающих клиентов. Работа в облаке предполагает также подключение к информационной базе по протоколу *HTTP* (*HTTPS*), благодаря чему клиенты могут работать через интернет из любой точки земного шара.

Завершающим элементом облачных технологий является инфраструктура сервиса, позволяющая развертывать приложения 1С:Предприятия в модели *SaaS*.

Эта бизнес-модель подразумевает, что поставщик разрабатывает прикладное решение и самостоятельно управляет им, предоставляя потребителю доступ к прикладному решению через Интернет. Такой подход избавляет потребителя от всех затрат, связанных с установкой, обновлением и поддержкой оборудования и программного обеспечения. Потребитель оплачивает лишь пользование услугой. Созданы профили безопасности в кластере серверов. Они отвечают за настройку разрешений на потенциально опасные действия.

К ним 1С относит открытие внешних отчетов и обработок, запуск приложений и обращение к ресурсам из Интернета. Обновление клиентских приложений через Интернет.

Не обошли вниманием специалисты 1С и веб-клиент. Теперь нет необходимости проверять настройки браузера по блокировке всплывающих окон.

Также решили проблему с пакетной печатью документов через Интернет без дополнительных диалогов и сохранением файлов в *pdf*.

Небольшие изменения коснулись и СКД (Система Компоновки Данных, предоставляющая возможность с минимальными усилиями получить отчет с развитой функциональностью) – добавлены новые

функции, вывод полей из несвязанных наборов данных, новые режимы диаграмм.

Интеллектуальные механизмы подготовки отчетов. Средства подготовки отчетности тесно интегрированы с другими механизмами платформы и имеют мощные возможности для интерактивной работы.

Благодаря этому отчеты органично вписываются в общий интерфейс приложения.

Фактически, пользователь в процессе работы не видит грани между общим интерфейсом и механизмом отчетности.

Построение распределенных и интегрированных информационных систем. Платформа содержит мощный набор механизмов обмена, способный решать самые разнообразные задачи. От поддержки территориально распределенных информационных баз, до построения сложных информационных систем, включающих, наряду с решениями на платформе «1С:Предприятие», еще и внешние приложения.

Кроме этого, специалисты 1С поработали над тем, чтобы система 1С Предприятие 8.3 была максимально дружелюбна к пользователям. В частности оптимизированы системы быстрого поиска, механизм выпадающих списков, реализованы несколько видов подсказок.

Помимо работы над усовершенствованием уже существующих механизмов и функций специалисты 1С в версии 8.3 добавили новые возможности в части функционала.

В первую очередь стоит отметить ***новое клиентское приложение для операционных систем семейства Linux.*** Раньше пользователи в ОС семейства Linux работали при помощи веб-клиента из браузеров. С появлением 1С Предприятия версии 8.3 появилась возможность запускать клиентские приложения в трех форматах:

Конфигуратор для разработки и администрирования информационной базы;

Толстый клиент, поддерживающий устаревающий режим обычных форм;

Тонкий клиент – наиболее оптимизированный режим приложения 1С 8 версии.

Это нововведение позволит работать на *Linux* не только пользователям, но и разработчикам, и администраторам. Остается помнить, что существуют некоторые ограничения на серверах под *Linux*:

– отсутствие поддержки работы с объектами *COM*;

– отсутствие взаимодействия с СУБД *MYSQL*;

– аутентификация происходит по специальному протоколу *Kerberos.*

Отдельного упоминания заслуживает ***механизм автоматизированного тестирования.*** Суть его заключается в том, что разработчик описывает алгоритм действий пользователя на встроенном языке платформы 1С 8.3 и воспроизводит его. Затем необходимо сравнить результат выполнения с тем, что ожидалось, и проанализировать итоги.

Для анализа поведения пользователя появилась возможность записать все интерактивные действия в отдельный файл формата *XML*.

Существенные изменения коснулись не только пользователей, но и разработчиков. Появились достаточно мощные инструменты, призванные существенно облегчить и ускорить процесс доработки конфигураций, разработки нового функционала и обслуживания серверов. Произошла серьезная модификация хранилища конфигурации.

Новая платформа 1С Предприятие 8.3 отличается существенной оптимизацией и лучшей целостностью хранилища. За счет этого разработчики намного быстрее получают результаты своей работы и меньше зависят друг от друга.

Также появились полезные возможности для выполнения регулярных работ. Среди них – создание произвольных областей в тексте модулей, шаблонов обработчиков событий, ссылки на методическую информацию в синтаксис-помощнике.

Времени с выхода релиза 1С Предприятие версии 8.3 прошло достаточно, и специалисты сформировали свое мнение об этой версии.

Большинство идей, получивших начало или развитие в 8.3, пришлись по вкусу клиентам 1С. Поэтому стоит ожидать их развития и совершенствования.

На сегодняшний день готовиться выйти версия 8.4, преимущества которой должны затмить систему 1С Предприятие 8.3.

Многих клиентов волнуют системные требования, предъявляемые 1С Предприятие 8.3 к серверам и клиентским компьютерам.

Несмотря на общее мнение о том, что любое ПО от 1С крайне требовательно, официальные требования не требуют внушительных затрат.

На клиентские ПК системные требования выдвигаются следующие:

- процессор с частотой более 1,8 ГГц;
- оперативная память объемом более 512 Мб;
- жесткий диск емкостью не менее 40 Гб.
- для развертывания сервера на компьютере, он должен иметь:
- *Intel Pentium* 4 с частотой более 2,4 ГГц;
- более 1 Гб оперативной памяти;
- жесткий диск в зависимости от размеров базы, но не менее 40 Гб.

Учитывая постоянное развитие, выбрать и внедрить 1С на базе платформы 8.3 эффективнее, чем поддерживать многочисленные программы на разных языках программирования.

Во главу угла в версии 8.3 поставлено удобство пользователей, администраторов и разработчиков.

2 УСТАНОВКА И НАЧАЛО РАБОТЫ С «1С: ПРЕДПРИЯТИЕ 8»

2.1 Установка системы

Установка учебной версии системы 1С: Предприятие на локальный компьютер мало чем отличается от установки любого приложения под ОС Windows. Для этого достаточно будет:

– скачать дистрибутив учебной версии, загрузив сайт ***https://online.1c.ru/***;

– далее в верхнем меню выбираем ***Софт***;

– в левом меню – ***1С:Предприятие 8. Версия для обучения программированию;***

– ***Платформа «1С:Предприятие 8.3 Учебная версия» 8.3.18.1128 (Windows)***;

– ***Получить продукт бесплатно;***

– на следующей странице необходимо заполнить анкету (ввести свои данные и адрес электронной почты), после чего нажать на кнопку ***Отправить***;

– после этого на почту придет письмо, скачиваем архив с учебной версией, распаковываем его и запускаем файл ***autorun.exe***.

2.2 Запуск системы. Создание информационной базы

Для изучения основных механизмов системы «1С: Предприятие», в рамках данного курса разработаем приложение, которое будет автоматизировать деятельность некой организации. В целом, деятельность компании можно свести к закупке товаров различного предназначения по оптовым ценам (поставки), хранения данных товаров у себя на складе с их последующей перепродажей розничным покупателям, соответственно, с учетом наценки.

Кроме того, предположим, организация занимается оказанием дополнительных услуг (доставка товара, сборка, фасовка и т.д.). Для анализа деятельности необходим учет: товаров на складах, их себестоимости, количества проданного товара, выручки за проданные товары и оказанные услуги, и другая информация. Также (дополнительно) необходимо организовать бухгалтерский и кадровый учет в организации.

Это – общее описание проектируемой системы. Более конкретные задачи будем формулировать в процессе разработки. В данный момент целью не будет являться разработка технического задания и получение прикладного решения в четком соответствии с ним, а задачей будет изучить основные механизмы и принципы разработки программных приложений в среде «1С: Предприятие 8», хотя нужно понимать, что в реальной жизни необходимо будет наличие всех составляющих проекта, начиная от предварительной заявки и заканчивая документацией, включающей в себя тестирование и опытную эксплуатацию разрабатываемой системы.

Итак, при запуске «1С: Предприятие», появляется следующее окно (рисунок 2.1):

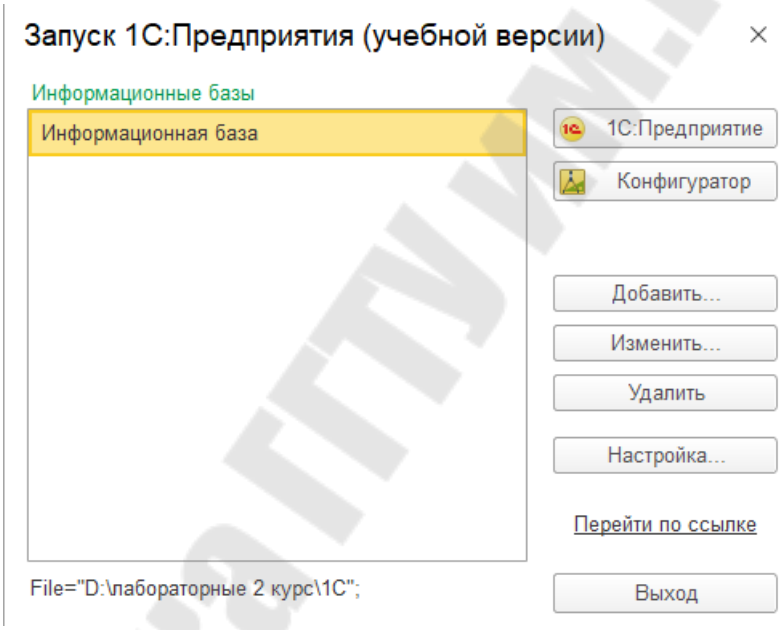


Рисунок 2.1 – Стартовое окно системы

Данное окно содержит список подключенных информационных баз. Под информационной базой будем подразумевать конфигурацию (или, проще говоря, само прикладное решение) в связке с базой данных, хранящей в своих таблицах учетные данные разработанного приложения. В случае, если информационных баз не создано, будет предложено создать новую информационную базу. Для ручного добавления, удаления, редактирования существующих информационных баз используются соответствующие кнопки.

«Настройка» (рисунок 2.1) предназначена для задания параметров отображения информационных баз, указания каталогов,

содержащих в себе шаблоны конфигураций (напрмер, Бухгалтерия предприятия, Управление Торговлей и т.д.), а также используемых версий системы: «1С: Предприятие», также позволяет установить и использовать различные релизы.

Добавим в список новую информационную базу. В открывшемся мастере отметим, создание новой информационной базы с пустой конфигурацией (будем разрабатывать прикладное решение «с нуля»). Название для базы можно задать любое, к примеру, «Обучение программированию», местом хранения будет выступать локальный компьютер: в учебных целях работа с информационной базой в файловом варианте является наиболее оптимальной. На заключительной странице мастера (рисунок 2.2), укажем автоматический выбор режима аутентификации (т.е. посредством настроек пользователей самого разрабатываемого решения).

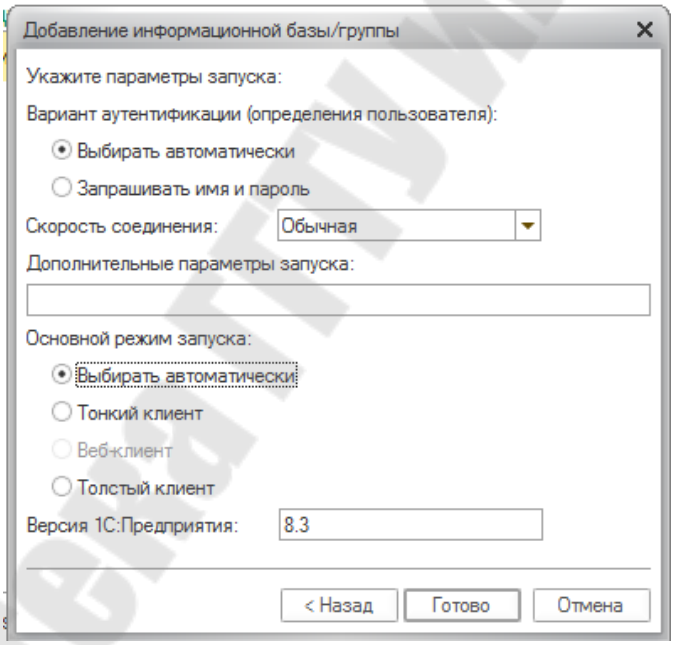


Рисунок 2.2 – Настройка аутентификации и режима запуска

Под основным режимом запуска подразумевается режим запуска системы в режиме исполнения. Система «1С: Предприятие» может функционировать в трех режимах:

– толстый клиент;
– тонкий клиент;
– веб-клиент;

Режим толстого клиента – «классический» режим работы системы «1С: Предприятие» при разработке и отладке прикладного решения. В режиме толстого клиента пользователю одновременно доступно как само приложение, со всем интерфейсом, всеми формами, так и база данных, которая содержит в себе всю информацию, выводимую в указанные формы приложения. При этом поддерживается постоянное соединение между приложением и базой данных.

Режим тонкого клиента подразумевает функционирование с разделением на клиентскую и серверную часть. На «клиенте» доступен лишь интерфейс приложения, со всеми формами, интерактивными элементами. А чтобы обратиться к данным, связанным с элементами формы, необходимо выполнять специальный серверный вызов, в результате которого происходит запрос к базе и получение требуемой информации. В режиме тонкого клиента не поддерживается постоянное соединение: система «накапливает» ряд серверных вызовов и отправляет их пакетом.

Режим веб-клиента – еще более облегченный «клиент-серверный» вариант работы приложения, реализующий доступ к функциональности через Интернет-браузер. Данный вариант не требует установки приложения на клиентский компьютер.

Разработка приложения возможна только в режиме толстого клиента. Для отладки и работы приложения достаточно (и рекомендуется) использование режима тонкого клиента (отметим соответствующий пункт). Таким образом, в современных версиях «1С: Предприятия» клиент-серверная идеология наблюдается как на физическом (установка приложения, использования сервера 1С, сервера СУБД), так и на логическом (внутренне устройство приложения) уровнях.

Настройка версии «1С: Предприятия» (рисунок 2.2) служит для того, чтобы при наличии нескольких установленных версий (включая релизы) системы, можно было бы создать информационную базу той версии, которая требуется.

2.3. Запуск «1С: Предприятие» в режиме разработки

При выборе информационной базы становятся активны кнопки «Предприятие» (для запуска конфигурации в режиме исполнения,

тонкий клиент) и «Конфигуратор» (для запуска конфигурации в режиме разработки/отладки, толстый клиент). Обратите внимание, что если установленная система использует аппаратный ключ защиты, то именно на данном этапе (при загрузке информационной базы) проверяется лицензия и, в случае ее отсутствия, система не запускается.

Для только что созданной пустой конфигурации режим «Предприятие» будет бесполезен, поэтому запустим систему в режиме «Конфигуратор». После запуска выберем пункт меню «Конфигурация» – «Открыть конфигурацию». В результате слева отобразится окно (дерево) создаваемой (или редактируемой) конфигурации (рисунок 2.3).

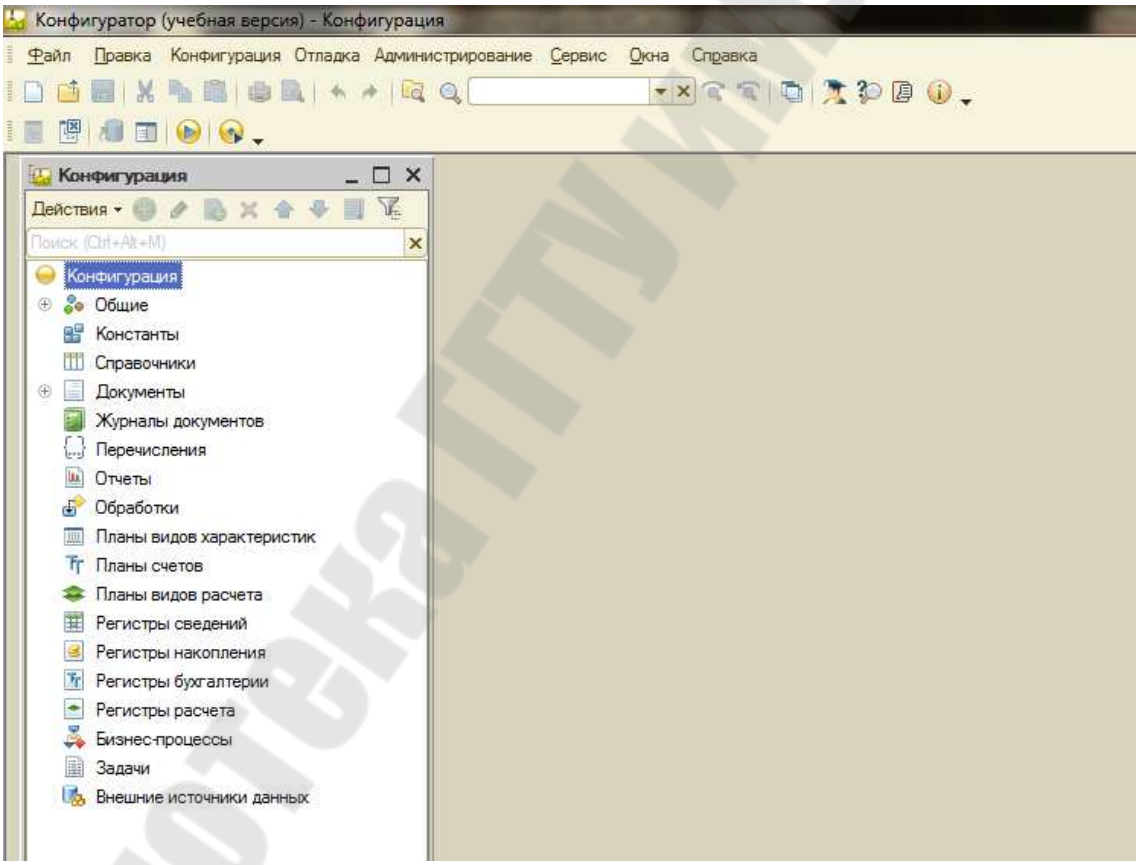


Рисунок 2.3 – Конфигуратор системы «1С: Предприятие»

Конфигурация состоит из множества прикладных объектов (Справочники, Документы, Отчеты и т.д.). Экземпляры прикладных объектов имеют отражение в таблицах базы данных, в которых будет

храниться учетная информация создаваемого приложения (список клиентов, документы поступления товара и т.д.).

Кроме прикладных, в конфигурации также имеются подчиненные объекты (реквизиты прикладных объектов), объекты встроенного языка, которые носят служебный (вспомогательный) характер, и используются для описания логики поведения системы, взаимодействия прикладных объектов друг с другом, а также взаимодействия системы с пользователем.

Обратите внимание, что при закрытии окна конфигурации, сама конфигурация никуда не исчезает и не закрывается. Для восстановления работы с деревом конфигурации необходимо в меню «Конфигурация» просто выбрать пункт «Окно конфигурации». Само окно можно сделать свободным, прикрепленным, прячущимся (сворачивается при редактировании какого-либо объекта конфигурации) при помощи соответствующих пунктов контекстного меню.

Меню «***Файл***», «***Правка***», «***Окна***», «***Справка***» главной панели достаточно стандартные, поэтому в особых пояснениях не нуждаются. Единственное, что стоит отметить – это наличие двух «справок». Первая – по системе в целом, стандартная справка. Вторая называется «Синтакс-помощник» и содержит в себе подробное описание прикладных объектов конфигурации, встроенного языка системы; примеры использования, правила написания программного кода, и т.д. «Синтакс-помощник» требуется разработчику при создании прикладного решения: любой программист просто не в состоянии запомнить все объекты встроенного языка, их свойства, методы, события, алгоритмы поведения и т.д. Он дополнительно может быть вызван при помощи значка  на командной панели «Конфигуратора», либо из любого модуля системы при помощи комбинации «Ctrl+F1».

Меню «***Конфигурация***» предназначено для работы с конфигурацией (несколькими – при их сравнении). Имеется возможность сохранить конфигурацию в файл, выгрузить/загрузить конфигурацию. Конфигурация – это само по себе прикладное решение, без конкретной информации (учетных данных) в базе данных; это множество прикладных объектов, связанных между собой и имеющих определенную модель поведения и взаимодействия (друг с другом, с пользователем). База данных – это хранимые

учетные данные объектов конфигурации. База данных и конфигурация – это две сущности, которые вместе образуют полноценно функционирующее приложение, в терминологии «1С» также имеется название «Информационная база». Так можно создать несколько информационных баз с одной и той же конфигурацией: приложение одно и то же, а учетная информация, содержащаяся в таблицах базы данных – различная. Также меню «Конфигурация» служит для настройки обновлений конфигураций типовых тиражных решений, а также собственно разработанных приложений.

Меню «***Отладка***» предназначено для запуска приложения в различных режимах, и его отладки. Как во многих языках программирования, возможно пошаговое исполнение программного кода, формирование точек останова и использование в них определенных отладочных механизмов, включающих в себя, в том числе, просмотр информации, которая в данный момент содержится в объектах конфигурации.

Меню «***Администрирование***» служит для создания пользователей решения, настройки их ролей, настройки правил аутентификации, а также ряд дополнительных сервисных функций по обслуживанию информационной базы. Дополнительно здесь имеется возможность произвести загрузку (или выгрузку) информационной базы. В данном случае формируется файл ***.dt**, который содержит в себе и конфигурацию, базу данных, и все настройки приложения вместе. Данная возможность часто используется при файловом варианте работы для сохранения резервных копий создаваемого приложения.

Меню «***Сервис***» предназначено для задания глобальных настроек системы, настройки режимов запуска приложения, настройки синтаксического контроля программных модулей системы и т.д.

Ниже главной панели располагается панель действий, состав которой зависит от текущего редактируемого объекта конфигурации, текущего программного модуля и т.д.

В остальной области «Конфигуратора» отображаются окна редактирования прикладных объектов и их свойств, тексты программных модулей и т.д.

3 СОЗДАНИЕ ОБЪЕКТОВ КОНФИГУРАЦИИ. КОНСТАНТЫ. ПОДСИСТЕМЫ

Отметим, что т.к. будем работать в файловом варианте, то периодически можно делать резервные копии (выгрузки) информационной базы через пункт меню ***«Администрирование» – «Выгрузить информационную базу»***. Для загрузки (и при этом полного замещения существующей открытой информационной базы без возможности ее восстановления), соответственно, используется пункт меню «Администрирование» – «Загрузить информационную базу».

Изучать основные механизмы системы «1С: Предприятие» будем поэтапно, рассматривая и создавая различные прикладные объекты. Одновременно с этим будет формироваться, расширяться функциональность разрабатываемого прикладного решения.

3.1 Константы

Константы – это постоянные, или, если точнее, условно-постоянные данные. Т.е. они содержат в себе информацию, которая не меняется с течением времени вообще, либо меняется, но очень редко, и при этом нет необходимости хранить историю данных изменений. Константа содержит в себе лишь одно значение определенного типа. К константам можно отнести, к примеру, название организации, дата создания, ФИО учредителя и т.п.

Для создания экземпляра любого прикладного объекта, необходимо в дереве конфигурации выбрать соответствующий тип объекта, и затем воспользоваться любым из перечисленных ниже способов:

– в панели действий дерева конфигурации нажать  ;

– в панели действий дерева конфигурации в меню «Действия» выбрать пункт «Добавить»;

– вызвать контекстное меню прикладного объекта и выбрать пункт «Добавить»;

– нажать клавишу «Insert».

При создании любого прикладного объекта, открывается окно свойств объекта конфигурации. На рисунке 3.1 представлено окно свойств при создании новой константы «НазваниеОрганизации».

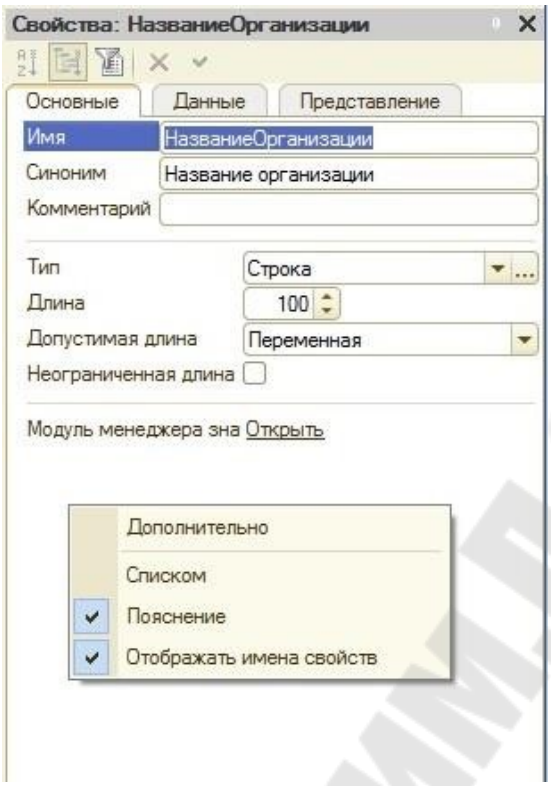


Рисунок 3.1 – Окно свойств константы

Все свойства объекта делятся на категории (Основные, Данные, Представление и др.), которые располагаются закладками. Однако можно все свойства расположить общим списком при помощи соответствующего пункта контекстном меню окна «Свойства». Список можно разделить на категории при помощи кнопки  , либо упорядочить по алфавиту при помощи  , при этом отключается деление на категории.

Для прикладных объектов есть два обязательных для заполнения свойства: «Имя» и «Синоним». Имя – это уникальное наименование объекта, по которому он будет идентифицироваться в системе. При этом имена экземпляров объектов различных классов (например, константы и справочники) могут совпадать.

В «1С: Предприятие» на имя объекта накладываются следующие ограничения:

- может состоять из букв и цифр, начинается с буквы;
- длина не больше 255 символов;
- не должно содержать спецсимволов, включая пробелы.

На основе имени заполняется синоним. Синоним – это представление имени, т.е. то, как данный объект будет отображаться в интерфейсе. На синоним нет таких жестких ограничений, как на имя т.к. синоним имеет строковое представление. Если имя состоит из несколько слов, написанных слитно, и каждое слово начинается с заглавной буквы, то синоним формируется таким образом, что второе и все последующие слова разделяются пробелом, а первая буква из заглавной преобразуется в строчную. Например, для имени константы «**НазваниеОрганизации**» система сформировала синоним «**Название организации**» (рис. 3.1). Синоним можно изменить, это никак не повлияет на имя. Но если поменять имя, то автоматически изменится и синоним.

Здесь, и в дальнейшем, когда будем создавать новые экземпляры различных объектов, то для удобства восприятия имя будет содержать пробелы. В системе таким объектам необходимо присвоить имя согласно требованиям «1С». Синоним можно задать на свое усмотрение.

Так, создадим следующие константы:

– **Название организации**: тип «Строка», длина 100 символов, переменная;
– **Дата основания**: тип «Дата», состав даты – «Дата»;
– **Бухгалтерский учет**: тип «Булево»;
– **Рейтинг надежности**: тип «Число», длина 1, точность 0.

Рассмотрим указанные типы данных чуть более подробно. В системе «1С: Предприятие» существует несколько типов данных: примитивные, ссылочные, составные. К примитивным типам данных относятся: число, строка, дата, булево, *NULL*, Неопределено и Тип.

Число: все вещественные числа. Указывается максимальная длина числа (с учетом знаков после запятой), а также точность (число знаков после запятой). Галочка «Неотрицательное» – для ввода только положительных чисел. Здесь следует быть осторожными: данная настройка не гарантирует, что пользователь не сможет ввести отрицательное число – система просто введенное число преобразует в положительное.

Строка: данные символьного/строкового типа. В поле «Длина» указывается максимальное число символов в строке. В свойстве «Допустимая длина» может быть выбран один из двух вариантов: «Фиксированная» или «Переменная». При фиксированной длине – число символов в строке максимально, вне зависимости от того,

сколько реально символов содержится в строке. Строки переменной длины могут содержать любое число символов в диапазоне от 0, до указанного максимального числа. Свойство «Неограниченная длина» устанавливает возможность использования строк произвольной длины. Это чем-то похоже на использование строк переменной длины, однако в таблицах базы данных такие строки хранятся по-другому.

Дата: данные о дате и времени. Дата в системе «1С: Предприятие» представляется в виде числа секунд от 00:00:00 01:01:0001 до 23:59:59 31.12.3999. Соответственно, внутренний формат представления даты – число. Свойство «Состав даты» определяет то, как дата будет представляться и вводиться пользователем: только время, только дата, либо и дата, и время. Однако это представление никак не влияет на внутреннее представление: объект все равно содержит в себе информацию, как о дате, так и о времени.

Булево: данные логического типа. Могут принимать значения «Ложь» (Нет) или «Истина» (Да).

NULL и ***НЕОПРЕДЕЛЕНО***: оба типа указывают отсутствие значения. **NULL** применятся тогда, когда значения не может быть в принципе. **НЕОПРЕДЕЛЕНО** используется тогда, когда значение в принципе может существовать, просто в данный момент времени оно еще не задано.

Тип: значения данного типа используются для идентификации типов значений. Это необходимо для определения и сравнения типов.

Ссылочные типы данных – это типы данных, образующиеся при создании ряда прикладных объектов, называемых типообразующими. К таким объектам относятся справочники, документы и др. Соответственно, на экземпляры (отдельные элементы) таких объектов можно сослаться.

Составной тип данных – тип данных, который содержит в себе указание на несколько (примитивных или ссылочных) типов. Соответственно, объект составного типа в различных ситуациях может иметь тот или иной тип: непосредственный тип данных указывается для каждого экземпляра объекта индивидуально, или же настраивается специальный механизм определения типа («Связь по типу»).

Созданные константы представлены на рисунке 3.2.

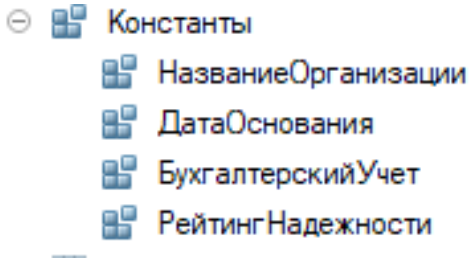


Рисунок 3.2 – Созданные константы

После того, как указанные константы созданы, их необходимо заполнить конкретными данными. Здесь необходимо обратить внимание на следующее. При разработке прикладного решения в системе обязательно присутствует две конфигурации: ***основная конфигурация*** и ***конфигурация базы данных***.

Основная конфигурация – это та конфигурация, которую разработчик интерактивно редактирует в «Конфигураторе»: создает объекты, описывает логику их взаимодействия и пр.

Конфигурация базы данных – это конфигурация, представленная в виде физических таблиц базы данных, описывающих те или иные объекты.

Пока программист не обновит конфигурацию базы данных – не произойдет реструктуризация таблиц (их создание или изменение), соответственно, ничего нового в приложении не появится. Поэтому существуют два варианта сохранения конфигурации: сохранение без реструктуризации таблиц базы данных и сохранение с внесением изменений в таблицы базы данных.

Для обновления конфигурации базы данных используется кнопка  на панели действий, либо соответствующий пункт в меню «Конфигурация» – «Конфигурация базы данных». После нажатия соответствующей кнопки система анализирует вновь созданные, или измененные объекты, которые повлияют на структуру таблиц, и выводит полный список изменений (рисунок 3.3).

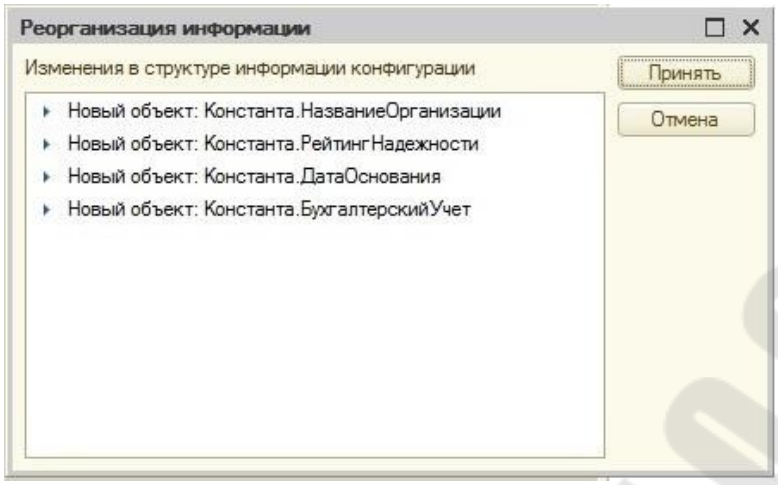


Рисунок 3.3 – Обновление конфигурации базы данных

Если разработчик согласен с внесением соответствующих изменений, то происходит обновление конфигурации базы данных и две конфигурации становятся полностью идентичными. После данной операции вернуться к исходной конфигурации базы данных нельзя. Однако если до обновления конфигурации базы данных разработчик решил, что необходимо отменить все изменения, сделанные в основной конфигурации (даже если она была сохранена) и вернуться к последней версии – это можно сделать при помощи команды «Вернуться к конфигурации базы данных» в меню «Конфигурация» – «Конфигурация базы данных».

После того, как конфигурация базы данных была обновлена, можно запустить приложение в режиме отладки при помощи кнопки  панели действий, или через соответствующий пункт меню «Отладка», и посмотреть результат. В большинстве случаев будем запускать приложение именно в режиме отладки, для анализа возникающих исключительных ситуаций и определения мест, где в конфигурации возникают ошибки. В результате откроется окно приложения (рисунок 3.4).

Верхняя область – панель навигации – содержит в себе создаваемые прикладные объекты, некоторые из которых объединены в группы. Так, например, созданные константы объединились в группу «Сервис». Такие группы образуют панель действий. В предыдущих версиях платформы «1С» панели навигации и действий были отделены: панель навигации располагалась слева, панель действий – вверху.

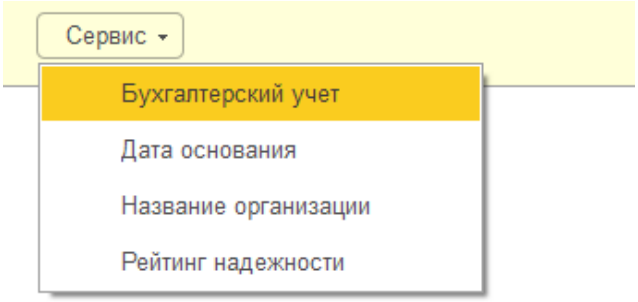


Рисунок 3.4 – Прикладное решение в режиме отладки/исполнения

При обращении к любому прикладному объекту, автоматически открывается основная форма (интерфейсное окно), связанная с этим объектом. Для константы основная форма представлена на рисунке 3.5.

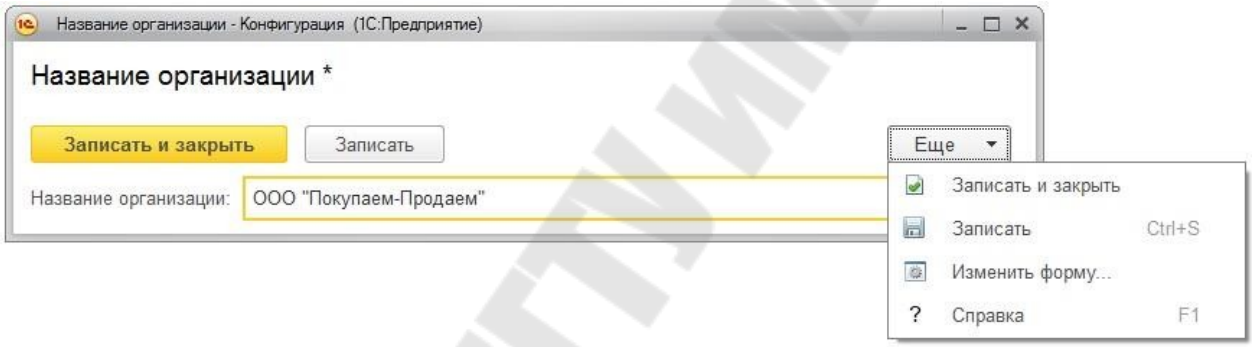


Рисунок 3.5 – Основная форма константы

Данная форма позволяет редактировать данные, которые содержит константа. После ввода данных в форму и их сохранения – они записываются в соответствующие таблицы базы данных. Впоследствии, при повторном открытии формы константы, в ней уже будут отображаться данные, которые автоматически загружаются из таблиц базы данных. В основной форме объекта (элемента) также доступен ряд действий в меню «Еще» (рис. 3.5).

Обратите внимание, что мы не сделали ничего, кроме создания объектов (констант), не написали никакого кода, не формировали интерфейс, не вдавались в особенности того, как информация из формы объекта будет заноситься в таблицы базы данных и прочее. Но на основании этой минимальной информации система автоматически спроектировала интерфейс приложения, разместила созданные объекты в определенных местах, создала формы для редактирования

информации, описала механизмы записи/извлечения информации в/из базы данных и т.д. Такое автоматическое создание интерфейса на основании минимального описания объектов – одна из ключевых особенностей разработки в системе «1С: Предприятие 8» (начиная с версии 8.2). Формируемый интерфейс носит название «***Управляемый интерфейс***», а создаваемые формы – «***Управляемые формы***». Под «управлением» в данном случае подразумевается управление самой системой. От разработчика требуется лишь концепция, общее описание того что, где должно быть и как функционировать. На основании этой концепции, на основании типа объекта, система формирует особое поведение, основные алгоритмы которого в нее уже заложены разработчиками. Только в том случае, если стандартное поведение системы нас не устраивает – только тогда программист изменяет стандартную функциональность, и описывает поведение системы на основании своих собственных алгоритмов.

Зададим значения созданных ранее констант и запишем их в таблицы базы данных:

- **Название организации**: ООО «Покупаем-Продаем»;
- **Дата основания**: 01.06.2020;
- **Рейтинг надежности**: 4;
- **Бухгалтерский учет**: Истина.

Можно заметить, что константы расположились не в том порядке, в котором мы их создали, а по алфавиту.

3.2. Подсистемы

Далее необходимо определиться со структурой организации. Не будет иметь совершенно никакого смысла сваливание в одну кучу информации обо всех сотрудниках, обо всех операциях поступления, продажи товара, обо всех отчетах и т.д. Такой информации будет достаточно много, она будут неупорядоченной, соответственно, поиск, анализ требуемых для ввода, получения, редактирования данных будет затруднителен. Поэтому самым очевидным является логическое разделение организации на ряд отделов, и с каждым из этих отделов можно будет связать только необходимые для его работы объекты. Например, отдел, который будет осуществлять закупку товаров, не будут интересовать информация о сотрудниках и начисленной им заработной плате. Зато данные о количестве товара

на складах, поступлении товара и т.д. – это будут те данные, которые необходимы для работы данного подразделения.

В системе «1С: Предприятие» для такого разделения используется объект конфигурации «Подсистема», расположенный в ветке «Общие» дерева конфигурации. Все остальные прикладные объекты конфигурации должны принадлежать хотя бы одной подсистеме (можно при необходимости нескольким). Те объекты, которые не будут включены ни в одну подсистему, не будут доступны в интерфейсе приложения – к ним можно будет обратиться, к примеру, программно.

Создадим следующие подсистемы:

– **Отдел закупок** – будет заниматься поступлением товаров;

– **Отдел продаж** – будет заниматься розничной продажей товаров;

– **Бухгалтерия** – будет заниматься бухгалтерским учетом на предприятии;

– **Отдел зарплаты** – будет заниматься начислением заработной платы;

– **Общий отдел** – будет содержать в себе общую информацию по предприятию.

Созданные подсистемы представлены на рисунке 3.6.

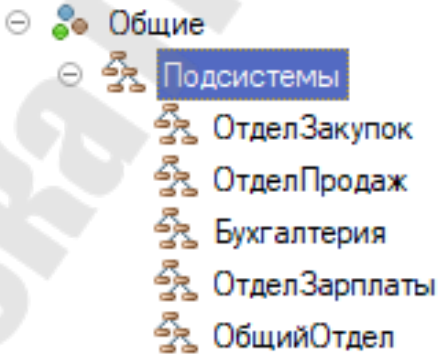


Рисунок 3.6 – Созданные подсистемы

Созданные ранее константы включены в подсистему «Общий отдел» (рисунок 3.7).

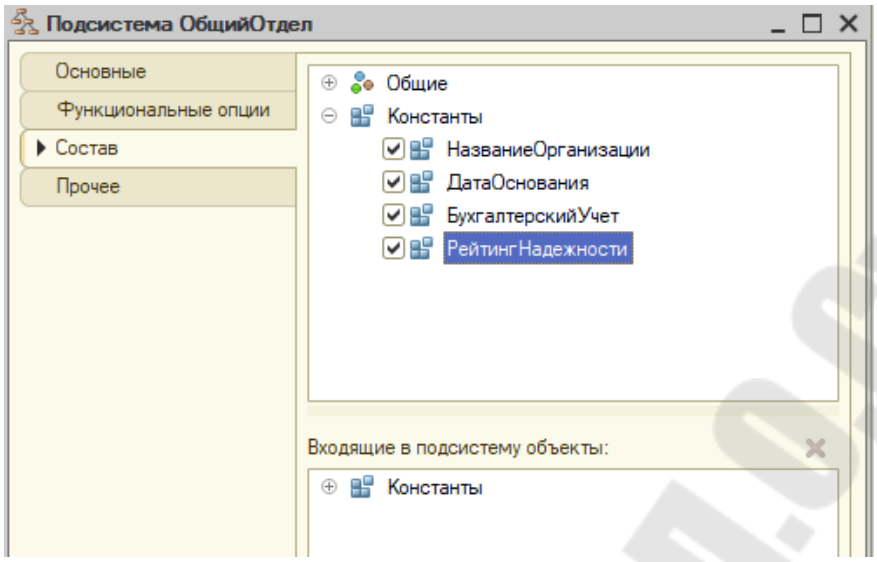


Рисунок 3.7 – Включение констант в подсистему «Общий отдел»

При создании подсистемы, кроме окна свойств, дополнительно открывается еще одно специальное окно (назовем его окном редактирования объекта) (рисунок 3.8).

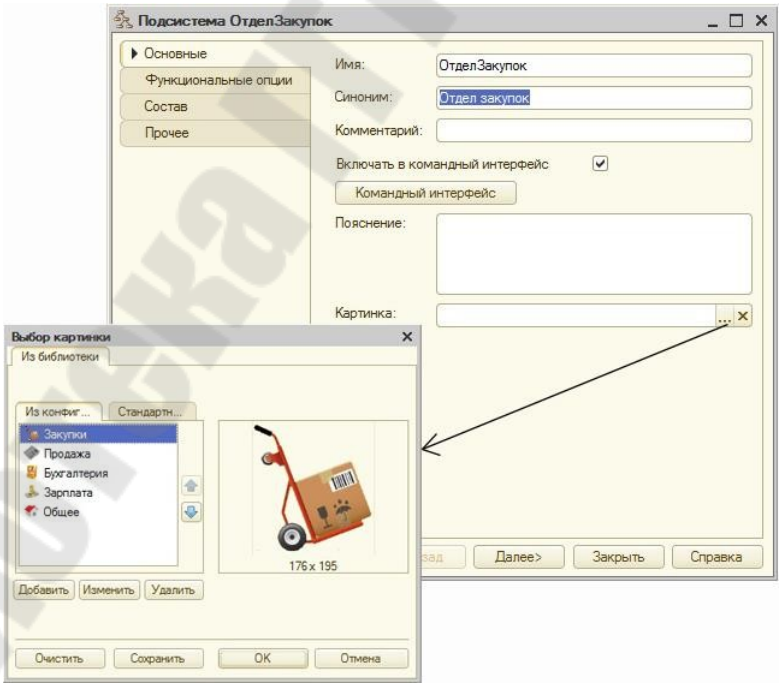


Рисунок 3.8 – Окно редактирования объекта (на примере объекта «Подсистема»)

Данное окно состоит из вкладок, содержащих наиболее часто настраиваемые свойства объекта.

При добавлении подсистемы на вкладке «Основные» (рис. 3.8) заполним поля «Имя» и «Синоним». Отметим галочку «Включать в командный интерфейс», чтобы созданная подсистема отобразилась в интерфейсе приложения. В свойстве «Картинка» укажем изображение для данной подсистемы. Изображение можно выбрать как из стандартных системных картинок, так и загрузить собственную картинку в конфигурацию – тогда загруженное изображение добавится в качестве объекта типа «Общие картинки» (в ветке «Общие» дерева конфигурации) с возможностью его дальнейшего использования.

На вкладке «Состав» (рис. 3.8) указываются те прикладные объекты, которые будут включены в подсистему. Состав подсистемы можно редактировать в любой момент. Сейчас у нас созданы лишь четыре константы, а их имеет смысл отнести лишь к подсистеме «Общий отдел».

Так, создадим и настроим указанные выше подсистемы. Обратите внимание, что внутри каждой подсистемы можно создать подчиненные подсистемы, т.е. образовать иерархию. Мы этого делать не будем т.к. на формирование интерфейса влияют лишь подсистемы верхнего уровня. Все вложенные подсистемы определяют лишь внутреннюю логическую организацию приложения. После создания подсистем, запустим приложение, и посмотрим на результат (рисунок 3.9).

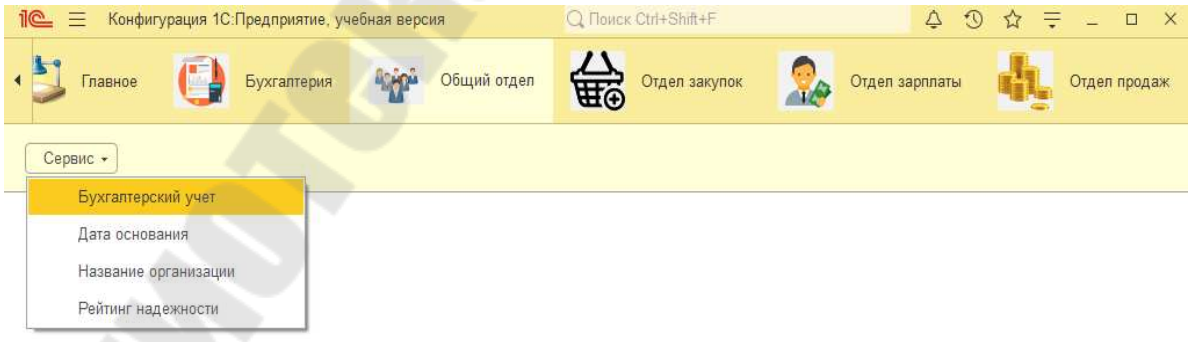


Рисунок 3.9 – Приложение с использованием подсистем

4 СПРАВОЧНИКИ. ПЕРЕЧИСЛЕНИЯ

4.1 Справочники

Справочник – это прикладной объект, содержащий себе информацию списочного вида. Примерами справочников могут быть список сотрудников, контрагентов, перечень товаров и услуг и т.д. Т.е. справочник содержит большой перечень однородной информации.

Объект «Справочник» является типообразующим объектом: при его создании в набор допустимых типов данных добавляется один новый тип данных «СправочникСсылка.<Имя_справочника>». Поэтому на любой элемент справочника можно указать ссылку. Например, при формировании документа продажи можно указать, какой сотрудник осуществляет продажу. Ссылка на элемент справочника формируется при его создании и представляется в виде хэш-строки. Данную ссылку нельзя изменить или удалить т.к. она формируется на основе типа создаваемого объекта и момента времени создания. Поэтому, даже если удалить элемент справочника, а затем создать «точно такой же» (т.е. содержащий идентичные данные) – то он все равно будет иметь другую ссылку, и система определит его как другой элемент справочника. Аналогичная ситуация наблюдается со всеми типообразующими (т.е. ссылочными) объектами.

Создадим справочник «**Номенклатура**» (рис. 4.1).

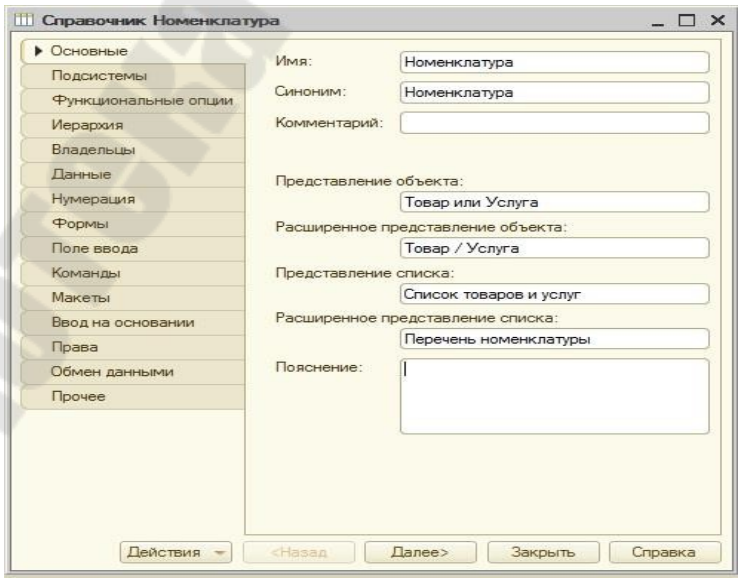


Рисунок 4.1 – Создание справочника «Номенклатура»

В окне редактирования объекта настроим свойства:

– **Представление объекта**: описывает то, как объект множественного числа будет называться при использовании в единственном числе (например, в панели действий в группе «Создать», для команды создания нового элемента объекта);

– **Расширенное представление объекта**: то же самое, что и «Представление объекта», но описывает заголовок формы создания/редактирования элемента объекта;

– **Представление списка**: описывает то, как объект единственного числа будет называться при использовании во множественном числе (например, в панели навигации для команды открытия списка элементов объекта);

– **Расширенное представление списка**: то же самое, что и «представление списка», но описывает заголовок формы списка элементов объекта.

Добавим созданный справочник в подсистемы «**Общий отдел**» и «**Бухгалтерия**» на вкладке «Подсистемы». Далее настроим иерархию на соответствующей вкладке (рис. 4.2). Иерархия позволяет создавать «подчинение» внутри справочника, с разделением всех элементов.

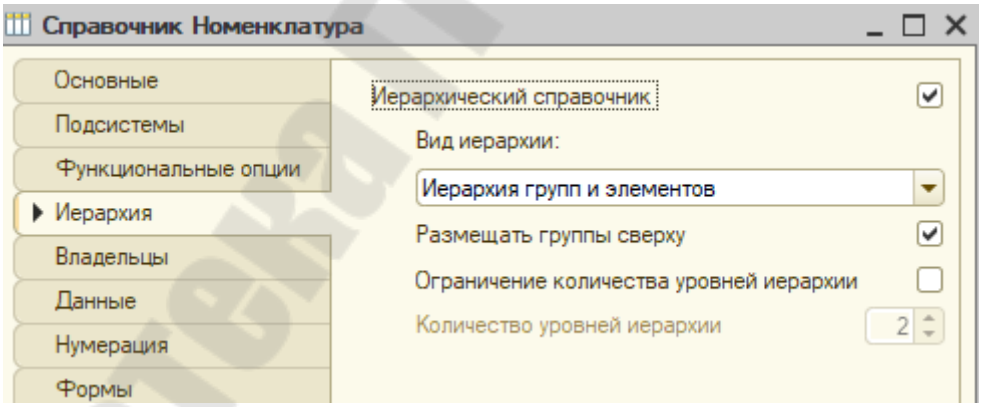


Рисунок 4.2 – Настройка иерархии справочника

Допустим, создается справочник, содержащий список сотрудников, каждый сотрудник работает в определенном отделе. Соответственно, если необходимо логически разнести всех сотрудников по отделам, то необходимо создать некоторые группы,

описывающие отделы, а в них уже разместить (сгруппировать) конкретных сотрудников. Это будет иерархия групп.

Если же сами по себе отделы могут являться объектами аналитики, а не просто как некоторые «хранилища» других элементов справочника, то тогда возможна иерархия элементов, когда одни элементы справочника как бы «группируют» другие.

В результате настройки иерархии выстраивается дерево, содержащее максимально столько уровней иерархии, сколько указано в настройке «Ограничение количества уровней иерархии» (рис. 4.2).

Для справочника «**Номенклатура**» имеет смысл создания групп товаров/услуг (например, канцелярские товары, мебель, оргтехника и т.д.). Внутри каждой группы также возможно деление на подгруппы (например, мебель для кухни, мебель для спален и т.д.) – количество таких вложенных уровней заранее неизвестно. Сами по себе группы нужны лишь для разделения товаров по категориям. Поэтому укажем, что созданный справочник будет иерархическим с иерархией групп и элементов без ограничения уровней иерархии.

На вкладке «Владельцы» (см. рис. 4.1) указываются другие справочники, которые «владеют» элементами данного справочника. Это – внешнее подчинение в явном виде, в отличие от иерархии – косвенного «подчинения» внутри справочника. Так, один элемент справочника-владельца может иметь в подчинении несколько элементов справочника- подчиненного. Самый наглядный пример: справочники «**Клиенты**» и «**Договора**» (с клиентами). С каждым клиентом может быть заключено несколько договоров. Соответственно, справочник «**Договора**» будет подчинен справочнику «**Клиенты**». У одного справочника может быть указано несколько справочников-владельцев, но при этом у отдельного элемента может быть указан лишь один владелец. Возможно существование нескольких вариантов подчинения: элементам, группам, элементам и группам.

Для справочника «**Номенклатура**» не будем организовывать подчинение.

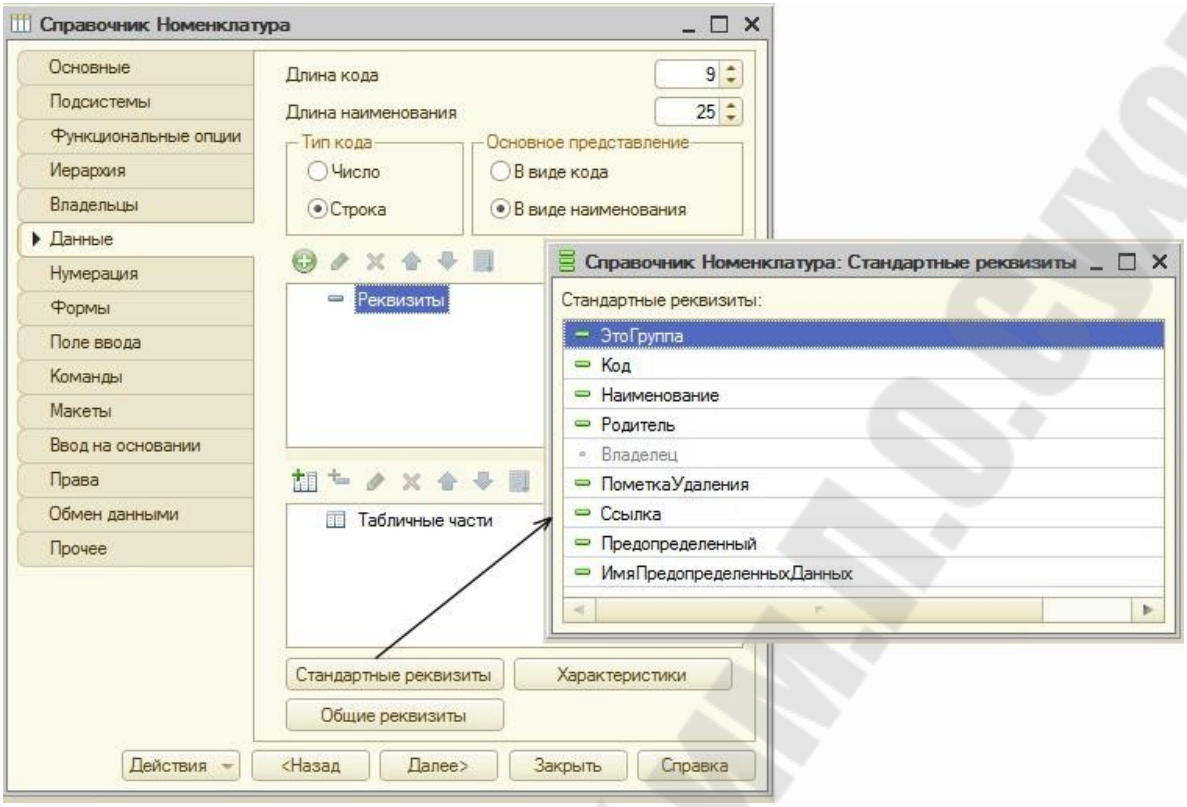


Рисунок 4.3.– Настройка реквизитов справочника

На вкладке «Данные» (рис. 4.3) настраиваются реквизиты справочника. Реквизиты – это некоторые характеристики элементов справочника. Для справочника всегда присутствуют два реквизита: «**Код**» и «**Наименование**».

«**Код**» может быть представлен в виде числа (максимально 38 цифр) или в виде строки (максимально 50 символов).

«**Наименование**» – только в виде строки. Настройка «Основное представление» описывает то, как элементы справочника будут отображаться в различных формах (при выборе, в полях ввода и т.д.). Для «**Кода**» или «**Наименования**» можно задать длину 0, тогда данный реквизит не будет отображаться в формах (и, соответственно, заполняться). Но, как минимум, один из данных реквизитов должен быть. Эти реквизиты относятся к списку стандартных реквизитов справочника, т.е. тех, которые присутствуют, изначально, а не настраиваются разработчиком. Также к стандартным реквизитам справочника относятся:

– **Ссылка** – уникальный идентификатор элемента справочника;

– **ЭтоГруппа** – реквизит типа Булево, указывает на то, что

элемент является группой; есть только при иерархии групп и элементов;

– **Родитель** – ссылка на элемент-родитель; есть только для иерархических справочников;

– **Владелец** – ссылка на элемент-владелец из справочника-владельца; есть только для подчиненных справочников;

– **ПометкаУдаления** – реквизит типа Булево, указывает на то, что элемент помечен для удаления;

– **Предопределенный** – реквизит типа Булево, указывает на то, что данный элемент является предопределенным;

– **ИмяПредопределенныхДанных** – только для предопределенных элементов: идентификатор (имя) предопределенного элемента.

Для стандартных реквизитов можно настроить их представление в интерфейсе, но нельзя самостоятельно ни отключить, ни удалить, ни изменить тип и т.д.

Также на вкладке «Данные» производится добавление собственных реквизитов элементов справочника. Выделяют реквизиты шапки (раздел «Реквизиты») и табличные части (раздел «Табличные части»).

Реквизиты шапки содержат информацию, однозначно характеризующую элемент справочника. Например, для справочника «**Клиенты**» это могут быть, допустим, полное наименование организации, ФИО директора, юридический адрес, ИНН и т.п. Реквизит шапки может содержать одно значение.

Табличная часть состоит из строк, содержащих некоторую дополнительную информацию. Данная информация однородна по структуре, но ее число может быть различно. Например, для сотрудников компании это может быть информация о контактных номерах телефонов, об образовании, о детях и т.п. Каждая строка табличной части складывается из набора реквизитов – реквизитов табличной части. Каждый реквизит табличной части может содержать одно значение. Самих строк табличной части может быть сколько угодно много, каждая из которых описывается полным наборов реквизитов. Табличных частей также может быть много.

Для справочника «**Номенклатура**» создадим реквизит шапки:

– «**Вид номенклатуры»**, тип – Строка, длина 20.

Вкладка «Нумерация» позволяет настроить уникальность «**Кода**»:

– в пределах всего справочника;

– в пределах подчинения (внутри каждой группы иерархического справочника своя нумерация элементов);

– в пределах подчинения владельцу (своя нумерация элементов подчиненного справочника, имеющих одного владельца);

Отключение свойства «Автонумерация» приводит к тому, что пользователю для каждого создаваемого элемента необходимо будет вручную заполнять код. Контроль уникальности в таком режиме затруднителен.

Для справочника «**Номенклатура**» оставим автоматическую нумерацию с включенным контролем уникальности серий кодов во всем справочнике.

На вкладке «Прочее» можно настроить предопределенные элементы справочника (рис. 4.4).

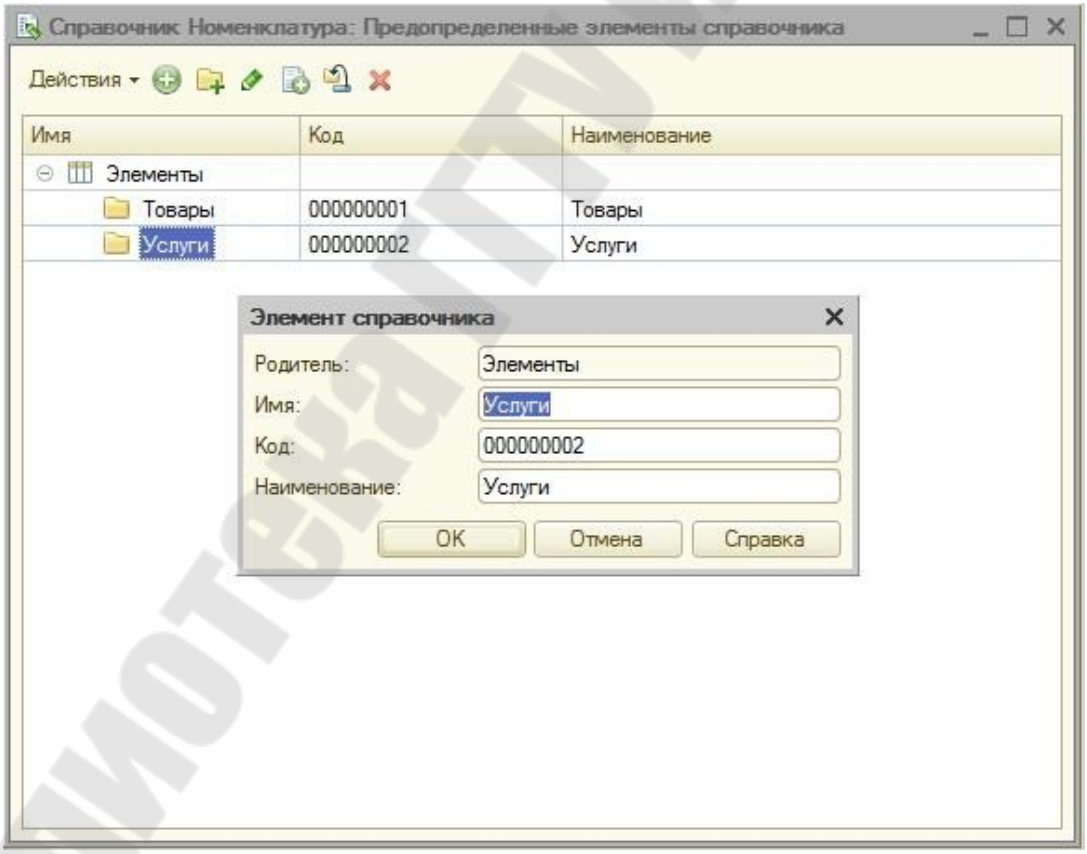


Рисунок 4.4 – Настройка предопределенных элементов справочника

Предопределенные элементы (в т.ч. группы) – это элементы, создаваемые на этапе разработки в Конфигураторе. Их нельзя удалить в режиме исполнения – только заполнить значениями и настроить представление. Каждый предопределенный элемент имеет уникальное «Имя» – это идентификатор, по которому данный элемент определяется в системе. При создании предопределенного элемента формируется жесткая ссылка, которая пользователем не может быть удалена или изменена. Благодаря этому в алгоритмах имеется возможность опираться на предопределенные данные т.к. они будут присутствовать в системе вне зависимости от действий пользователя. В отличие от предопределенных, элементы, создаваемые в режиме исполнения, могут быть удалены. Соответственно, нельзя гарантировать их присутствие в системе, поэтому программные алгоритмы не могут их использовать.

При создании предопределенных элементов иерархического необходимо обращать внимание на текущий уровень иерархии: создаваемые элементы будут автоматически размещаться внутри указанного уровня.

Для изменения элемента-родителя используется кнопка  (рис. 4.4).

Для справочника «***Номенклатура***» создадим две предопределенные группы: «***Товары***» и «***Услуги***». Имена групп зададим точно такие же. Внутри групп не будем создавать никаких элементов.

Аналогичным образом создадим еще несколько справочников.

Справочник «***Контрагенты***»: включен в подсистемы «***Общий отдел***» и «***Бухгалтерия***»; иерархия групп и элементов с ограничением в 2 уровня; длина «***Кода***» – 6, «***Наименования***» – 30 символов. Используется Автонумерация кода, контроль уникальности серий кодов во всем справочнике.

Есть две предопределенные группы: «***Покупатели***» и «***Поставщики***». Реквизиты шапки:

– **Полное наименование** – строка, 100 символов, переменная длина;

– **ИНН** – строка, 12 символов, переменная длина;

– **Расчетный счет** – строка, 12 символов, фиксированная длина.

Обратите внимание, что «**ИНН**» и «**Расчетный счет**» задали в виде строки. Дело в том, что эти данные носят информационный характер и не будут их использоваться в арифметических операциях, поэтому нет смысла задавать их в виде числа и занимать достаточно большой объем памяти (под 12 знаков). Для того, чтобы пользователь ввел в эти поля только цифры, необходимо настроить маску ввода. Для того, чтобы определить, как задается маска необходимо обратиться к ***Синтакс-помощнику***.

В разделе «***Индекс***» введем строку поиска и выберем соответствующую информацию (рис. 4.5).

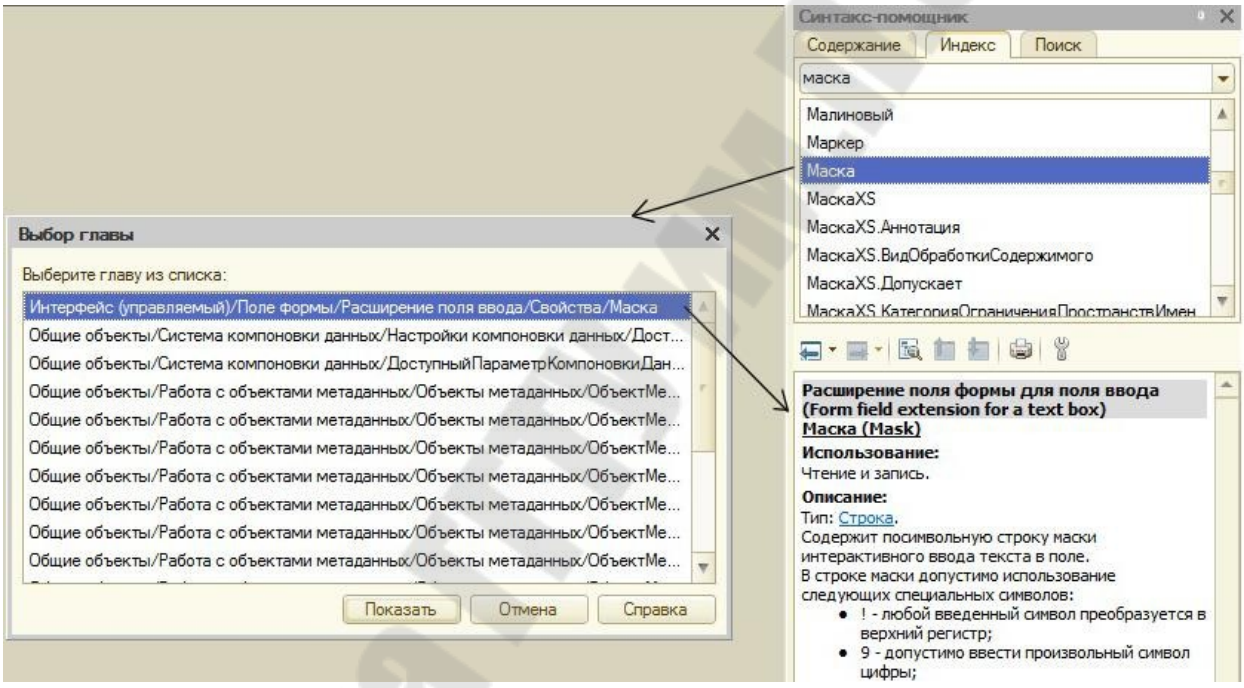


Рисунок 4.5 – Поиск информации в Синтакс-помощнике

В результате откроется окно со списком глав, в которых присутствует искомая информация. Дело в том, что в различных прикладных объектах могут использоваться механизмы, свойства, события и т.д., которые имеют одинаковые имена, но реализуют разную функциональность. Нашему запросу соответствует самый первый вариант по работе с полями формы управляемого интерфейса. Дважды щелкнув мышью на выбранной главе, открывается раздел, содержащий запрашиваемую справочную информацию (рис. 4.5).

Так, видим, что для задания в строке ввода цифры используется маска «9». Для реквизитов справочника маска задается на вкладке

«Представление», поле «Маска». Соответственно, для «**ИНН**» и «**Расчетного счета**» маска будет содержать 12 цифр «9».

Справочник «**Договора**»: включен в подсистемы «**Общий отдел**» и «**Бухгалтерия**», неиерархический, подчинен справочнику «**Контрагенты**» (подчинение элементам); длина кода – 0, наименования – 50.

Здесь указали длину кода 0. На вкладке «Нумерация» сразу же исчезла возможность какой-либо настойки. Дополнительно, следует сказать, что «1С: Предприятие» поддерживает поиск по индексированным данным. По умолчанию, «**Код**» и «**Наименование**» индексируются. При отключении любого из указанных реквизитов необходимо убрать и поиск по соответствующему реквизиту. Для этого переходим на вкладку «Поле ввода» (см. рис. 4.1), выбираем раздел «Ввод по строке», в открывшемся списке убираем из выбранных полей «**Код**». В список реквизитов, которые могут быть использованы для ввода по строке также можно добавить и другие. Необходимым условием для этого является включение индексации на вкладке «Использование» при настройке свойств соответствующего реквизита.

Справочник «**Сотрудники**»: включен в подсистемы «**Общий отдел**» и «**Отдел зарплаты**»; справочник неиерархический, неподчиненный; длина кода – 5, наименования –30.

Реквизиты шапки:

– **ФИО**: синоним «ФИО (полностью)», строка, 100 символов, переменная;

– **Дата рождения**: тип Дата, состав даты – дата;

– **Семейное положение**: синоним «Холост / Женат», тип Булево.

Изменим представление «Кода» и «Наименования». Для этого откроем «Стандартные реквизиты» и дважды щелкнем по реквизиту «**Код**», чтобы открыть окно свойств. В качестве синонима укажем «Табельный номер». Аналогичным образом зададим синоним «ФИО» для реквизита «**Наименование**».

Также для справочника «**Сотрудники**» опишем табличную часть «**Образование**».

Реквизиты табличной части:

– **Учреждение**: строка, 100 символов, переменная;

– **Вид учебного заведения**, строка, 20 символов, переменная;

– **Специальность**: строка, 50 символов, переменная;

– **Год окончания**: строка, 4 символа, фиксированная длина, настройка маски для ввода только цифр.

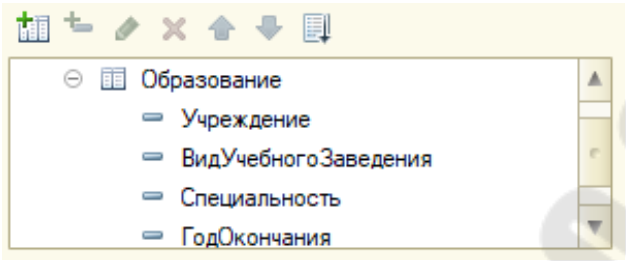


Рисунок 4.6 – Добавление реквизитов табличной части «Образование»

Обновим конфигурацию базы данных и запустим приложение в режиме отладки. Перейдем к подсистеме «**Общий отдел**» и зафиксируем появившиеся изменения. Откроем, к примеру, справочник «**Список товаров и услуг**» и добавим новую номенклатурную позицию (рис. 4.7).

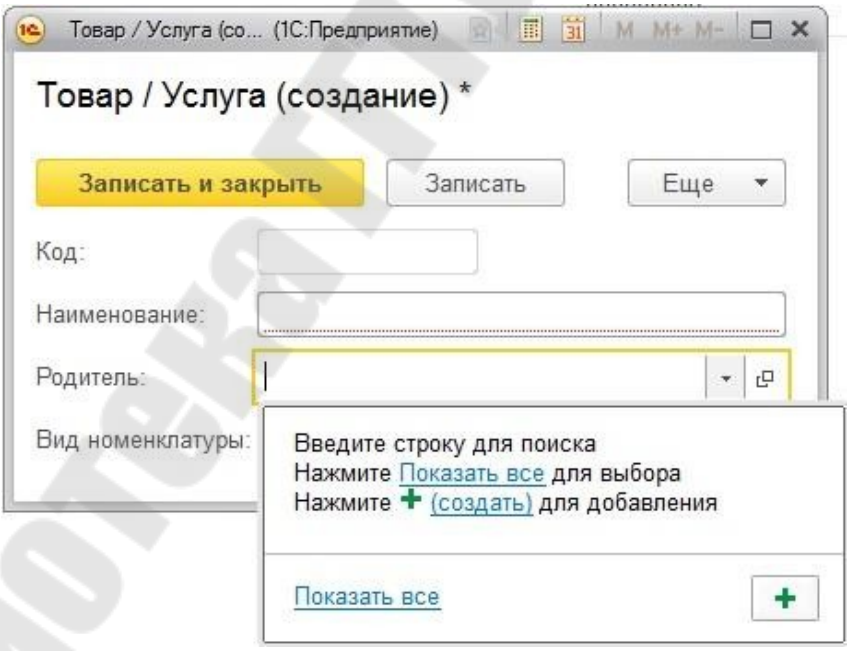


Рисунок 4.7 – Добавление элемента в справочник «**Номенклатура**»

Обратим внимание, что поле «**Наименование**» подчеркнуто красной пунктирной линией. Это означает, что данное поле обязательно для заполнения. Соответственно, если попытаемся

записать элемент справочника без заполнения данного поля, то система выдаст ошибку и элемент не будет записан. Настройку проверки заполнения можно сделать для любых реквизитов, табличных частей и реквизитов табличных частей при помощи свойства «Проверка заполнения» на вкладке «Представление» настройки свойств реквизита.

Поле родитель сейчас не заполнено. Это означает, что создаваемая номенклатурная позиция будет размещена в корне справочника, а не в какой-либо группе. Для выбора группы необходимо нажать на стрелку справа (рис. 4.6). Открывается всплывающая подсказка. Т.к. поле «Родитель» это ссылка, то необходимо либо выбрать родителя из уже имеющихся групп справочника (ссылка «Показать все»), либо создать новую группу при помощи  .

Создадим группу «**Канцтовары**» в предопределенной группе «**Товары**». Создадим номенклатурную позицию «**Карандаш**» (Вид номенклатуры – «**Товар**») и поместим его в группу «**Канцтовары**». Обратите внимание, что даже если вы случайно указали не ту группу при создании элемента, то это всегда можно изменить: достаточно открыть созданный ранее элемент и выбрать нужную группу-родителя.

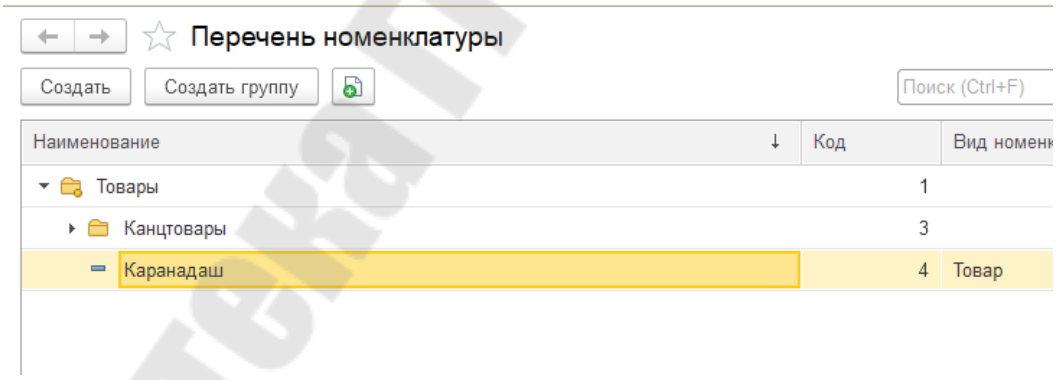


Рисунок 4.8 – Создание позиции «Карандаш»

Пока что не будем заполнять все справочники нужными данными. Сделаем это чуть позже, когда добавим еще одну функциональность, которая существенно упростит ввод данных. Поэтому систему в режиме исполнения можно закрыть.

4.2 Перечисления

В созданных справочниках ряд реквизитов (например, «**Вид номенклатуры**», «**Вид учебного заведения**») заданы в виде строки. Однако их будем заполнять преимущественно одними и теми же значениями. Например, для «**Вида номенклатуры**» это будут всего два значения: «**Товар**» или «**Услуга**». Каждый раз, заполняя данный реквизит, мы вынуждены полностью заполнять данные строки текста. Дополнительно, если необходимо будет аналитика по виду номенклатуры, необходимо будет производить посимвольное сопоставление строк, что также не очень удобно. Также пользователь может попросту допустить опечатку.

Необходимо, чтобы данные, имеющие ограниченный перечень принимаемых значений, можно было бы один раз задать, а потом производить выбор из них. Конечно, для этих целей можно создать отдельный справочник и выбирать элементы из него.

К недостаткам такого варианте можно отнести возможность изменять элементы справочника, что сделает затруднительным их однозначное использование. Поэтому в системе «1С: Предприятие» есть прикладной объект, который будет содержать требуемые списки значений, задаваемые и изменяемые только в Конфигураторе: в режиме исполнения пользователь сможет лишь делать выбор из данных списков. Такие списки целесообразно создавать тогда, когда необходимость их редактирования возникает крайне редко. Такой объект конфигурации называется «***Перечисление***». Аналогично предопределенным элементам справочника, все элементы перечисления имеют жесткие ссылки. В режиме исполнения в перечисление нельзя добавлять новые элементы или редактировать уже созданные.

Перечисление содержит только значения элементов – без указания типа. Создадим перечисление «**Виды учебного заведения**», добавим его в подсистему «**Общий отдел**», и в данном перечислении зададим несколько значений (на вкладке «Данные»): **школа**, **НПО**, **СПО**, **ВУЗ** (рис. 4.9).

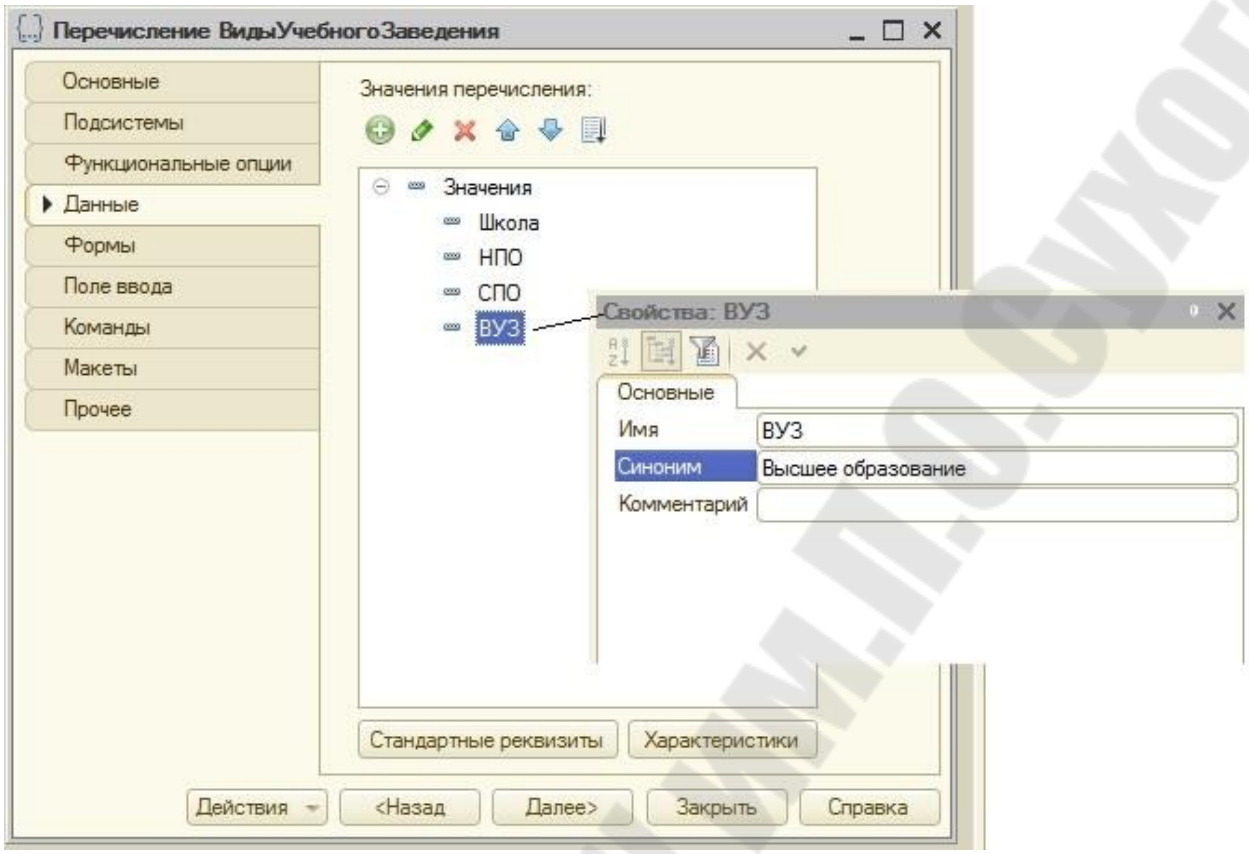


Рисунок 4.9 – Заполнение значения перечисления

Впоследствии, если мы захотим программно обратиться к конкретным значениям перечисления, то это можно будет сделать по полному имени, например:

Перечисления.ВидыУчебногоЗаведения.СПО

Для того, чтобы в реквизит «**Вид учебного заведения**» табличной части «**Образование**» справочника «**Сотрудники**» можно было подставлять значения из созданного перечисления, необходимо изменить тип реквизита: вместо типа «Строка» указать тип «**ПеречислниеСсылка.ВидыУчебногоЗаведения**» (рис. 4.10).

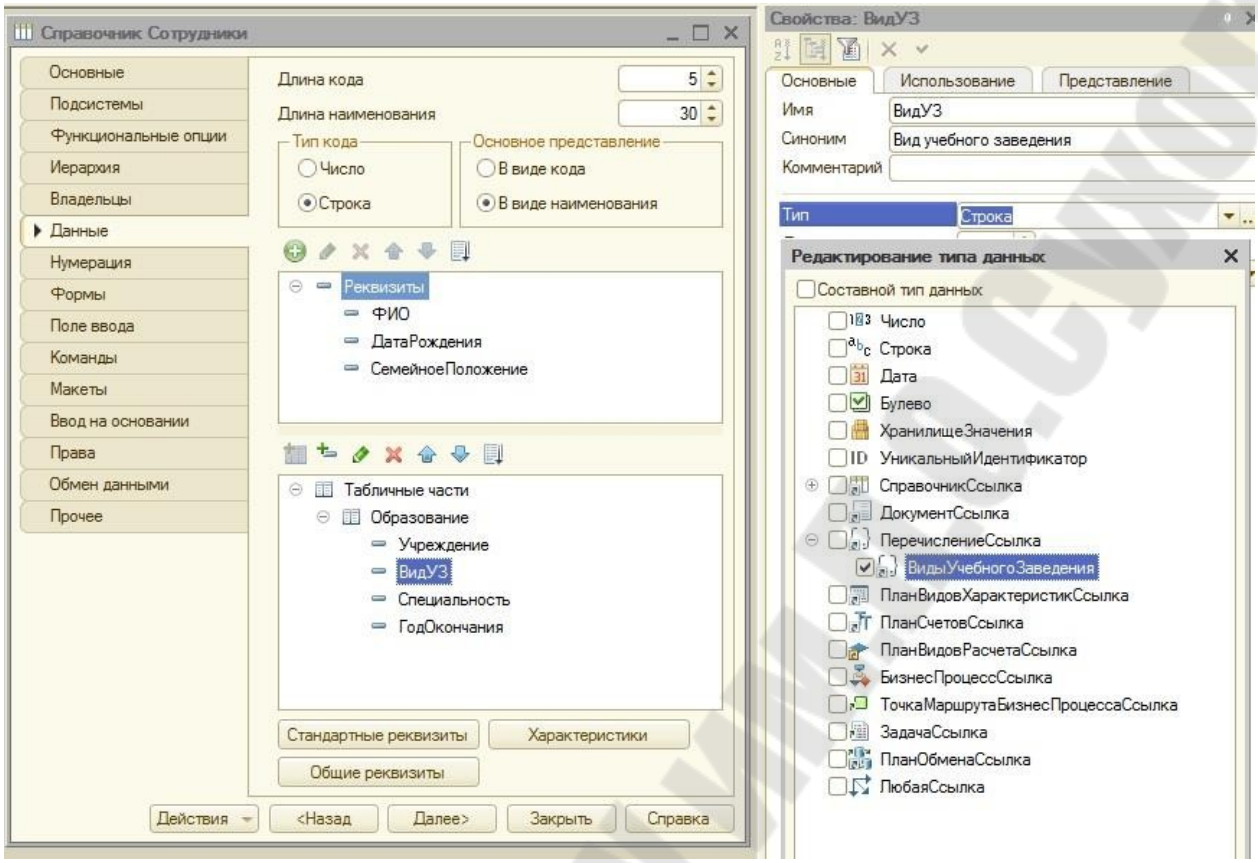


Рисунок 4.10 – Выбор ссылочного типа данных (Перечисления)

Аналогичным образом создадим перечисление «**Виды номенклатуры**» (значения «**Товар**» и «**Услуга**») и изменим тип реквизита «**Вид номенклатуры**» справочника «**Номенклатура**». Дополнительно, для данного реквизита на вкладке «Представление» свойство «Быстрый выбор» установим в значение «Использовать».

Запустим систему в режиме отладки. Откроем для редактирования созданную ранее номенклатурную позицию и зададим вид номенклатуры. Обратите внимание: список возможных вариантов невозможно отрыть в отдельном окне – конкретное значение можно выбрать только из выпадающего вниз списка. Такое поведение связано с установленным свойством «Быстрый выбор». Данное свойство имеет смысл настраивать, если перечисление (или другие данные ссылочного типа) содержат немного значений.

Для сравнения, создадим сотрудника и заполним соответствующие реквизиты шапки и реквизиты табличной части (рис. 4.11).

Обратите внимание, что здесь, аналогично предыдущему случаю, список возможных вариантов был в виде выпадающего списка внизу. Это связано с тем, что настройка

«Быстрый выбор» в разделе «Представление» была установлена в режим «Авто». Соответственно, система сама определила, что в данном перечислении немного элементов и приняла решение использовать быстрый выбор. Для сравнения, установите режим быстрого выбора в режим «Не использовать» для данного реквизита и посмотрите на то, как это проявится в режиме исполнения.

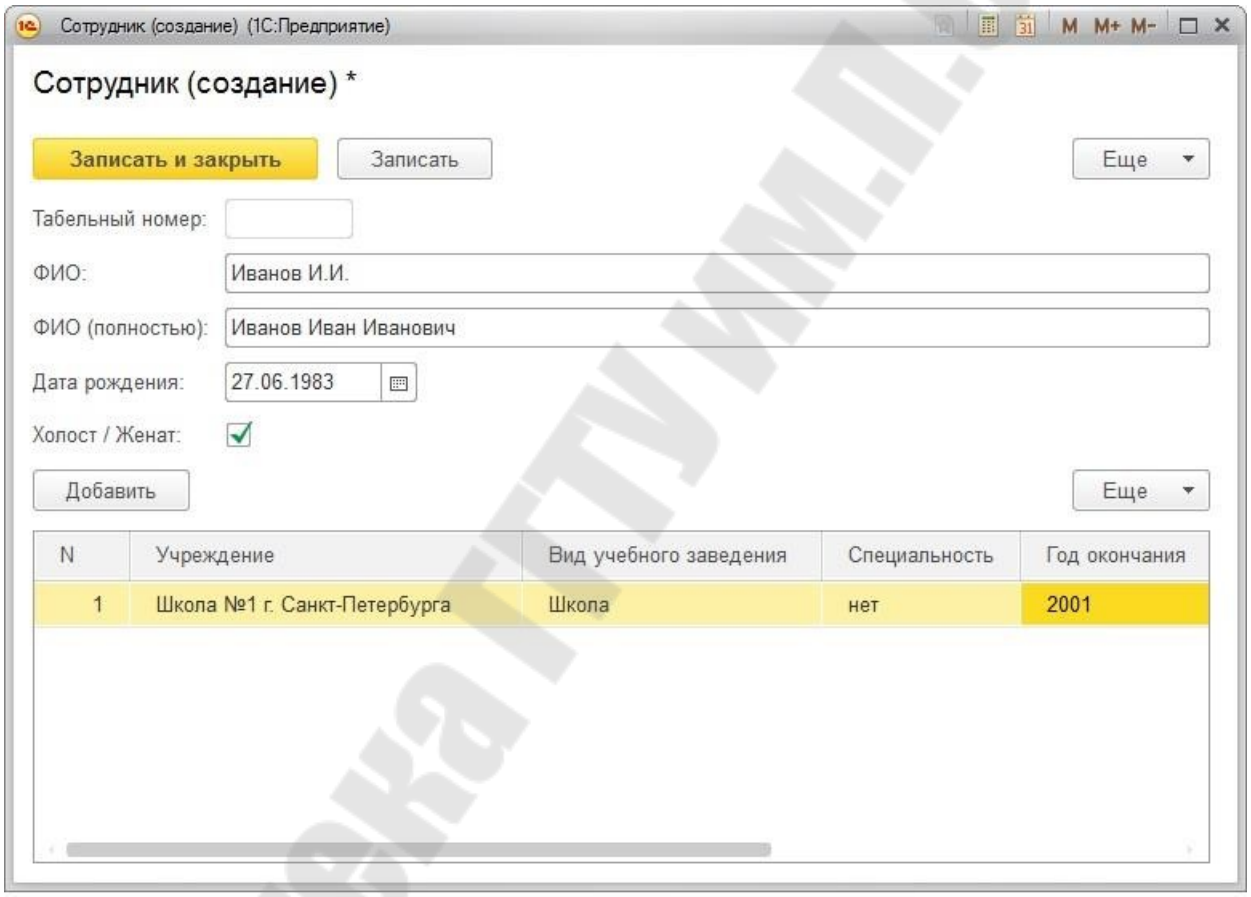


Рисунок 4.11 – Создание сотрудника (реквизиты шапки, реквизиты табличной части)

Самостоятельно заполните созданные справочники различными группами товаров, товарами и услугами; сотрудниками (включая заполнение табличной части); контрагентами (покупателями и поставщиками) и договорами с ними (по несколько договоров с каждым контрагентом).

Обратите внимание, что при добавлении/открытии контрагента вверху появляется ссылка «**Список договоров**». Перейдя по этой ссылке, можно просмотреть, добавить, изменить договора, связанные с выбранным контрагентом. Такая возможность существует благодаря тому, что справочник «**Договора**» подчинен справочнику «**Контрагенты**», и система автоматически отображает содержимое справочника «**Договора**» с отбором по выбранному (открытому в данный момент) контрагенту.

Также следует отметить, что зачастую приходится заносить множество похожей информации. Соответственно, можно просто скопировать уже введенную информацию (созданный элемент справочника), а затем внести лишь необходимые правки. В системе «1С: Предприятие» в режиме исполнения для этого необходимо выделить элемент, на основании которого будет создаваться копия, и в меню «Еще» выбрать пункт «Скопировать». После чего откроется новый элемент справочника в режиме редактирования, в который нужно просто внести все необходимые изменения. Через вышеуказанное меню также можно выполнить ряд других действий: «Изменить», «Пометить на удаление», «Создать», «Переместить в группу» и т.д.

5 ОСОБЕННОСТИ ПРИМЕНЕНИЯ ПРОГРАММНОГО КОДА В СИСТЕМЕ

5.1 Встроенный язык системы

До сих пор при разработке приложения использовали лишь стандартные механизмы «1С: Предприятия», без использования программного кода. Система многие события обрабатывает самостоятельно путем уже имеющихся механизмов. Тем не менее, существуют моменты, когда стандартного поведения системы оказывается недостаточно для реализации требуемой функциональности. В нашем случае явным примером является добавление элементов в справочник «Номенклатура». Вы наверняка заметили, что для каждой номенклатурной позиции необходимо явно указывать вид номенклатуры. А это может показаться нелогичным: мы и так находимся в определенной группе товаров, так почему система не может это проанализировать и самостоятельно заполнить

вид номенклатуры? Стандартного механизма для такого анализа у системы нет. Для того чтобы систему «научить» выполнять такой анализ, необходимо использовать встроенный язык системы.

Встроенный язык системы является некоторой смесью множества известных вам языков программирования, включая объектно-ориентированное программирование. Главное отличие заключается в том, что весь программный код пишется на русском языке. Возможно использование англоязычных аналогов операторов, но, в большинстве случаев, программисты все же пишут в «1С» на русском языке.

Переменные, операторы

Объявление переменной: **Перем <*имя_переменной*>;**
Присвоение переменной значения: **<*имя_переменной*>** = <***значение***>;

Отметим, что в системе нет предопределенного задания типа и жесткой типизации переменных. Тип определяется по первому операнду в выражении, после чего производится попытка автоматического приведения остальных операндов к данному типу. – Например, для объявления массива:

<***имя_переменной***> = **Новый Массив**;

Запись даты:

Сегодня = '20140221122045' (12:25:45 21 февраля 2014 года). Можно написать: **Сегодня = '2014-02-21 12:25:45'** – все незначащие символы (в данном случае не цифры) игнорируется т.к. в одинарных кавычках пишется только дата. Если не указывать время, то оно будет равно 00:00:00. Есть понятие **ПустаяДата** – 00:00:00 на 01:01:01.

Дата хранится в виде числа секунд от 00:00:00 на 01:01:01. Соответственно с датой можно производить арифметические операции сложения, вычитания для получения другой даты. Можно из одной даты вычесть другую и получить разницу в секундах.

Строки всегда записываются в двойных кавычках. Если строка очень длинная, к примеру, текст запроса, тогда имеется возможность переноса при помощи вертикальных разделителей:

A = “Строка1
| Строка2
| Строка3
….
| СтрокаN”;

Для объединения нескольких строк используется конкатенация символом «+».

Числа записываются как обычно, без всяких спецсимволов (10, 12.5). Основные операции: сложение, умножение, вычитание, деление, % (остаток от деления нацело).

Переменные булевого типа принимают два значения:
Истина или **Ложь**.

Все операторы отделяются друг от друга точкой с запятой. Все незначащие символы (пробелы, переносы строк и т.д.) – просто игнорируются. Например, запись **A = 10%4** будет эквивалентна

A =

10

%4;

Комментарии обозначаются символом // в начале каждой строки комментария.

Стандартные алгоритмические конструкции

1. Условный оператор:

Если (Условие) Тогда
[ИначеЕсли (Условие) Тогда]
….
[Иначе]
….

КонецЕсли;

2. Оператор цикла с параметром:

*Для Каждого <**переменная**> Из <**СписокПеременных**> Цикл*
….
КонецЦикла;

3. Оператор цикла с условием:

*Пока <**условие**> Цикл*
….
КонецЦикла;

4. Процедуры и функции:

Процедура ИмяПроцедуры ([Параметр1, Параметр2,…])
….
КонецПроцедуры;

Вызов процедуры:

ИмяПроцедуры (***значение1***, ***значение2***);

По умолчанию передача параметров происходит по ссылке, т.е. значения внутри процедуры, функции могут изменяться. Для передачи параметров по значению (для сохранения значения при выполнении), перед именем параметра указывается ключевое слово **Знач**.

Для вызова процедур, функций в других модулях (а не там, где они описаны) используется ключевое слово **Экспорт**:

Процедура ИмяПроцедуры ([Параметр1, Параметр2,…]) Экспорт

Отличие функции от процедуры – в конце присутствует ключевое слово **Возврат** для передачи управления в точку вызова. Результат работы функции должен быть записан в некоторую переменную.

5. Булевы операции: возвращают результат «Истина» или «Ложь». В результате операций сравнения переменная получает значение типа булево: например, переменная **А** в результате операции **А = 10<3** примет значение «Истина».

Для описания операции отрицания используется конструкция «НЕ»: **В = НЕ А**.

Есть операторы логических связок «И», «ИЛИ», «НЕ». Пишутся такие операторы подряд: **А = 10 = 10 ИЛИ 3<>5**.

Если необходимо сформировать приоритеты операций используются круглые скобки.

5.2 Виды программных модулей

Система «1С: Предприятие» является событийной. Т.е. все, что происходит, не происходит просто так, а лишь при наступлении определенных событий. Например, пользователь нажимает кнопку создания элемента справочника, и система реагирует на это действие – возникает событие и производится его обработка. Поэтому при использовании встроенного языка программный код не пишется линейно, чтобы некоторые действия выполнялись по заданному алгоритму: любой программный код привязывается к конкретному событию системы, связанному либо с объектами (включая элементы интерфейса), либо с поведением системы в целом. В зависимости от видов событий в системе «1С: Предприятие» выделяется ряд программных модулей, каждый из которых отвечает за обработку тех или иных событий и имеет определенное предназначение. Выделяются следующие виды программных модулей.

Модуль формы. Здесь происходит обработка событий, связанная с элементами формы (интерфейсом приложения).

Форма – основной объект для интерактивного отображения данных. К примеру, когда нажимаем кнопку создания нового элемента справочника, открывается форма элемента для объекта типа «Справочник» со своим набором данных, свойств, методов. Модуль формы связан с интерактивными действиями пользователя (ввод данных в поля формы, выбор данных, и т.д.). Тип формы зависит от отображаемых данных. Существуют различные виды форм, связанные с одним и тем же объектом: форма элемента, форма списка, форма выбора, а также собственные разработанные программистом формы. Соответственно, каждая форма будет иметь

свой перечень элементов и свой модуль, содержащий обработчики событий для данной формы.

Модуль объекта. В модуле объекта происходит обработка событий, связанная с объектом в целом, вне зависимости от формы. Такие события обычно связаны с записью информации в таблицы базы данных и не зависят от интерактивных действий пользователя. В отличие от модуля формы, модуль объекта содержит обработчики, реагирующие на любые программные обращения к объекту. Если рассматривать последовательность обработки событий, то вначале происходит обработка событий, возникающих в какой-либо форме объекта (интерактивные события), а потом уже обработка событий объекта (программные события).

Примечание. Модуль объекта отсутствует у констант, перечислений, регистров. У регистров вместо модуля объекта присутствует ***модуль набора записей***.

Модуль менеджера. Появился только в версии 8.2. Очень похож на модуль объекта, но модуль объекта содержит реализацию методов и обработчиков событий, относящихся к конкретному экземпляру объекта, а модуль менеджера содержит методы, которые расширяют/переопределяют методы, относящиеся к всему набору экземпляров объекта, без привязки к конкретному экземпляру. Модуль менеджера есть у всех прикладных объектов.

Модуль управляемого приложения. Содержит обработчики событий, возникающих при старте системы в режиме «Управляемого приложения» (начинается с версии 8.2). Также *может* содержать процедуры и функции, вызываемые в других частях приложения, хотя для этого, преимущественно все же используются другие модули.

Модуль сеанса. Содержит обработчики событий, связанные с приложением в целом, вне зависимости от его типа (обычное или управляемое). Происходит установка параметров сеанса до старта системы.

Общие модули. Содержат экспортные процедуры и функции, доступные и вызываемые из всех других модулей приложения. Очень

близки к понятию «внешние модули» (UNITS) в классических языках программирования.

5.3 Работа приложения в управляемом режиме. Управляемые формы

Рассмотрим, как система устроена изнутри, как она получает данные, обрабатывает их и сохраняет.

Начиная с версии 8.2 основной режим работы «1С: Предприятие» – это работа в режиме «Управляемого приложения» с использованием управляемых форм. ***Управляемые формы*** – это такие формы, которые разработчиком не создаются самостоятельно «с нуля», как это было раньше (в предыдущих версиях системы). Система сама, на основании типа объекта и типа формы уже «знает», как форма должна выглядеть, как функционировать, какие события у нее должны быть и какие обработчики должны быть связаны с этими событиями. От программиста единственное что требуется – это указать системе, какие элементы на форме расположить и где. Остальное система все сделает сама. И только в том случае, если стандартное поведение системы не будет устраивать, только тогда программист может изменять, добавлять требуемую функциональность.

Внутри приложение, разработанное на базе «1С: Предприятия», функционирует по модели «клиент-сервер». «Клиент» содержит все интерактивные элементы управления (кнопки, меню, поля ввода и т.д.), но сам по себе не содержит никаких данных. Все данные находятся на «сервере» – в таблицах базы данных: непосредственно с клиента мы не можем обратиться к ним. И только при специальных серверных вызовах эти данные с сервера поступают к клиенту.

Форма (отображающая те или иные данные), как программный объект, достаточно уникальна. Она одновременно существует и на «клиенте», и на «сервере». Если быть более точным, то на сервере она создается (программная копия), после чего передается клиенту и отображается (интерактивная копия, визуальная проекция). У формы есть реквизиты, имеющие тип «Данные формы». Один из реквизитов является основным (у формы лишь один реквизит может быть основным) и связан с таблицами базы данных. При серверном вызове происходит:

1. формирование специального SQL-запроса к таблицам базы данных;

2. извлечение требуемых данных;

3. преобразование данных из формата базы данных к типу «Данные формы»;

4. помещение данных в соответствующие реквизиты основного реквизита формы.

Кроме основного, форма также может иметь произвольное число других реквизитов. Данные реквизиты (не основной) не связаны с таблицами базы данных, а содержащаяся в них информация, формируется в результате работы различных программных алгоритмов при вызове или интерактивной работе с формой. При закрытии формы информация, хранящаяся этих реквизитах, никуда не записывается, а пропадает (если не реализовано какого-либо программного алгоритма по передаче указанных данных).

Для отображения содержимого реквизитов формы используются элементы формы, имеющие тип «Элементы формы». Элементы формы – это те интерактивные элементы (поля ввода, надписи, кнопочки, переключатели и т.д.), которые видит пользователь при интерактивной работе с формой и которыми он управляет.

Рассмотрим пример. Допустим, мы открываем уже созданный ранее элемент справочника. При этом формируется команда на открытие формы элемента данного справочника. На сервере создается форма, и извлекаются данные из таблиц базы данных (этап 1, рис. 5.1).

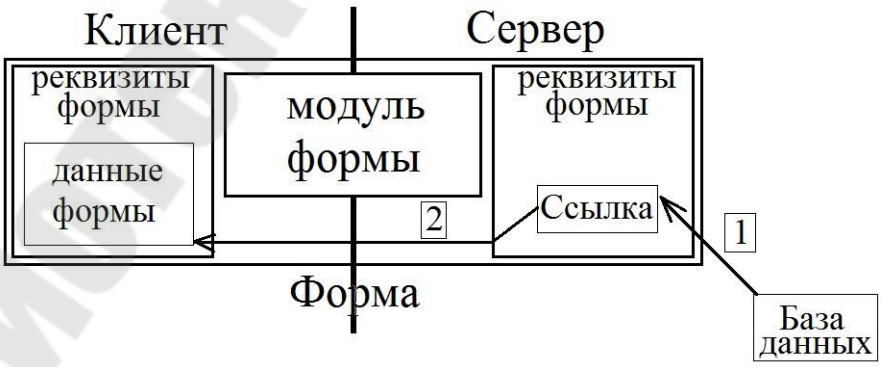


Рисунок 5.1 – Принцип открытия управляемой формы

Следует отметить, что при этом извлекаются не все данные, а лишь ссылка и ее представление (Код или Наименование). Ссылка нужна для того, чтобы впоследствии к этим данным можно было обратиться, а представление – для их вывода в интерфейсе. После этого форма «переезжает» на клиент; извлеченные данные преобразуются и помещаются в основной реквизит формы (Данные формы) (этап 2, рис. 5.2). Затем выполняется команда открытия формы, в результате которой элементы формы отображают содержимое реквизитов формы. Элементы формы связаны с данными формы через специальное свойство «Путь к данным».

Далее, допустим, в уже открытой на клиенте форме пользователь что-то изменил. Вопрос: эти изменения происходят где: на клиенте или на сервере? Конечно же, то, что пользователь изменил, т.е. ввел в интерактивные элементы формы, производится на клиенте. Сервер пока что «не знает», что на клиенте что-то поменялось. Однако если происходит серверный вызов, то измененные данные с клиента «переезжают» на сервер и происходит синхронизация данных на клиенте и на сервере.

Если же при работе с формой, пользователь на клиенте запросил какие-то данные с сервера, то происходит следующее. Реквизиты формы на клиенте содержат измененные данные из интерактивных элементов (рис. 5.2).

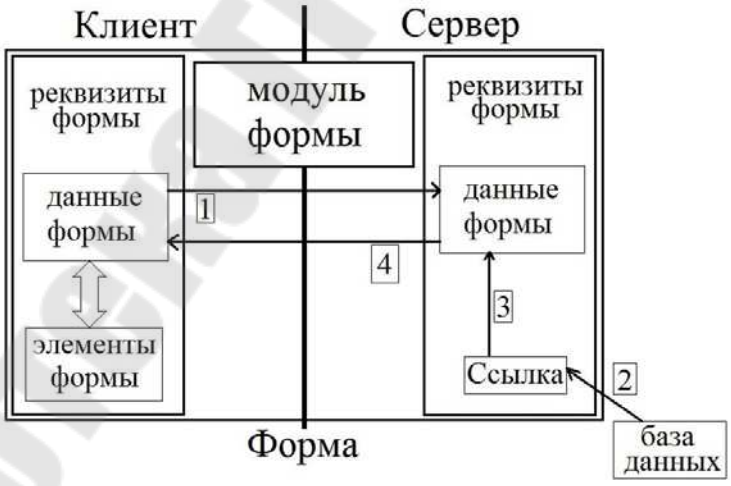


Рисунок 5.2 – Работа управляемой формы при серверном вызове

Измененная копия формы «переезжает» на сервер и происходит синхронизация с реквизитами формы на сервере (этап 1, рис. 5.2). Далее выполняется требуемый серверный вызов, например, запрос к базе данных (этап 2, рис. 5.2). Из базы данных извлекается требуемая

информация и помещается в реквизиты формы на сервере (этап 3, рис. 5.2). После чего форма обратно «переезжает» на клиент, и происходит снова синхронизация данных на клиенте и на сервере (этап 4, рис. 5.2). В самом конце измененные реквизиты отображаются в соответствующих элементах формы. При обращении к базе данных (например, при записи данных), данные формы преобразуются к формату таблиц базы данных при помощи специального метода **ДанныеФормыВЗначение()**, при помещении данных из базы данных в реквизиты формы происходит обратное преобразование при помощи метода **ЗначениеВДанныеФормы()**.

Модуль формы существует одновременно и на клиенте и на сервере. Для того чтобы система знала, к данным и методам, клиента или сервера, в данный момент следует обращаться, существует 4 директивы, предваряющие любые процедуры и функции, описанные в модуле формы:

– **&НаКлиенте**: доступны все элементы и реквизиты формы на стороне клиента, нет доступа к базе данных – необходим серверный вызов;

– **&НаСервере**: доступны все реквизиты формы на стороне сервера, есть доступ к базе данных;

– **&НаСервереБезКонтекста**: доступны только те реквизиты формы на стороне сервера, которые были явно переданы в качестве параметров вызова, есть доступ к базе данных;

– **&НаКлиентеНаСервереБезКонтекста**: доступны все элементы формы и те реквизиты формы, которые были явно переданы, есть доступ к базе данных.

Основное отличие в директивах &НаСервере и &НаСервереБезКонтекста – это объем передаваемых данных. При контекстном вызове передаются все данные и описание формы. При безконтекстном вызове передаются лишь необходимые данные, без описания. Это обеспечивает сокращение передаваемых по сети данных и, как следствие, увеличение быстродействия системы.

В процессе разработки управляемых форм чаще всего используются директивы &НаКлиенте и &НаСервереБезКонтекста.

5.4 Использование встроенного языка. Создание приветствия пользователя

Теперь, когда разобрали особенности функционирования системы, попробуем написать небольшой программный код и посмотреть, как он будет работать.

Самое элементарное, что можно попытаться сделать – это поприветствовать пользователя, запустившего систему. При запуске системы происходит выполнение действий, описанных в модуле управляемого приложения. Т.к. будем разрабатывать только управляемые приложения (обычных приложений и обычных форм касаться не будем), то это как раз то, что нам нужно.

Для того, чтобы открыть модуль управляемого приложения необходимо вызвать контекстное меню конфигурации и выбрать соответствующий пункт. В результате откроется пустой текст модуля. Как было отмечено ранее, система является событийной. Соответственно, необходимо описать события, которые и будут выполняться в данном модуле. Для того, чтобы посмотреть список всех событий и выбрать нужное, используется специальная кнопка  на панели инструментов (выделена красным, рис. 5.3).

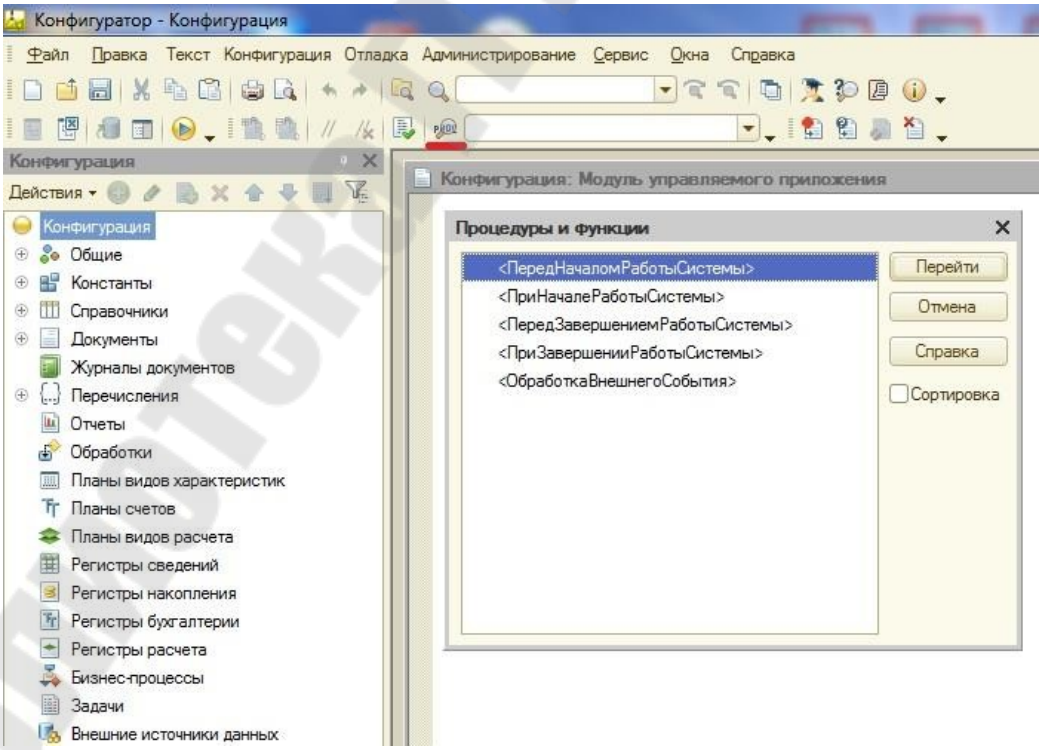


Рисунок 5.3 – События модуля управляемого приложения

Т.к. будем выводить приветствие при загрузке разработанного решения, то нам необходимо событие «При начале работы системы». В результате выбора сформируется шаблон процедуры, которую необходимо заполнить. В настоящий момент процедура пустая. Но это не значит, что данное событие никак не обрабатывается. Система производит обработку по умолчанию. В том случае, если обработчик события будет изменен (добавлен программный текст), тогда обработка по умолчанию будет замещена созданной обработкой. Если в дальнейшем, созданная обработка будет удалена, или закомментирована, тогда система снова будет производить обработку по умолчанию.

Добавим в обработчик события следующий текст:

Сообщить ("Сегодня "+ТекущаяДата());

Данная строка будет выводить сообщение с текущей датой в отдельном окне. ТекущаяДата() – это функция получения текущей системной даты. Обратите внимание, что здесь использовали конкатенацию строк, и система автоматические производит преобразование даты к строке. Попробуйте поменять дату и текстовую строку местами и посмотрите на результат. Для проверки правильности синтаксиса написанного программного кода используется кнопка  , расположенная рядом с кнопкой вызова списка всех возможных событий.

Запустим систему в режиме отладки и посмотрим на результат.

Усложним задачу. Кроме приветствия пользователя, покажем ему список тех сотрудников из соответствующего справочника, у кого сегодня день рождения. Данные по всем сотрудникам хранятся в базе данных, т.е. «на сервере». Вспомним, что модуль управляемого приложения исполняется на стороне клиента. Соответственно, непосредственно в модуле управляемого приложения мы не можем получить доступ к данным – необходимо делать специальный серверный вызов для получения требуемой информации.

Для решения поставленной задачи самый лучший способ – это создать общий модуль, который будет исполняться на сервере. В этом модуле следует описать функцию, получающую список именинников. А в модуле управляемого приложения, соответственно, сделать вызов данной функции (серверный вызов), т.к. общие модули являются

местом описания процедур и функций, доступных из любой точки программы.

Создадим общий модуль (раскроем в дереве конфигурации ветвь «Общие» и найдем соответствующий класс объектов) и назовем его «**Общие механизмы**» (рис. 5.4).

Для любого общего модуля обязательным является указание: глобальный он или нет (настройка «Глобальный», рис. 5.4). Процедуры и функции глобальных общих модулей доступны в любой части приложения и вызываются просто по имени. При этом имена процедур и функций, описанных в разных глобальных общих модулях не должны пересекаться. В неглобальных общих модулях процедуры и функции могут иметь одинаковые имена, а для их вызова вначале указывается имя общего модуля, а затем через точку имя вызываемой процедуры/функции.

Затем галочками необходимо отметить, где выполняются процедуры и функции данного модуля: на клиенте, на сервере или и там и там (рис. 5.4). Если отметить что-то одно, то перед именами процедур/функций не нужно будет указывать предваряющую их директиву. Если же отметить обе галочки (клиент, сервер), то перед каждой процедурой/функцией необходимо будет располагать директиву, указывающую, где данная процедура/функция выполняется.

Процедуры/функции исполняемые на клиенте доступны для вызова из клиентских модулей, на сервере – соответственно из серверных. Однако если на клиенте необходимо сделать вызов серверной процедуры/функции, описанной в общем модуле, то в настройке модуля необходимо отметить галочку «Вызов сервера» (рис. 5.4).

Общий модуль не содержит никаких событий – только те процедуры/функции, которые будут вызываться в других частях приложения. Соответственно, их компиляция производится только в момент вызова.

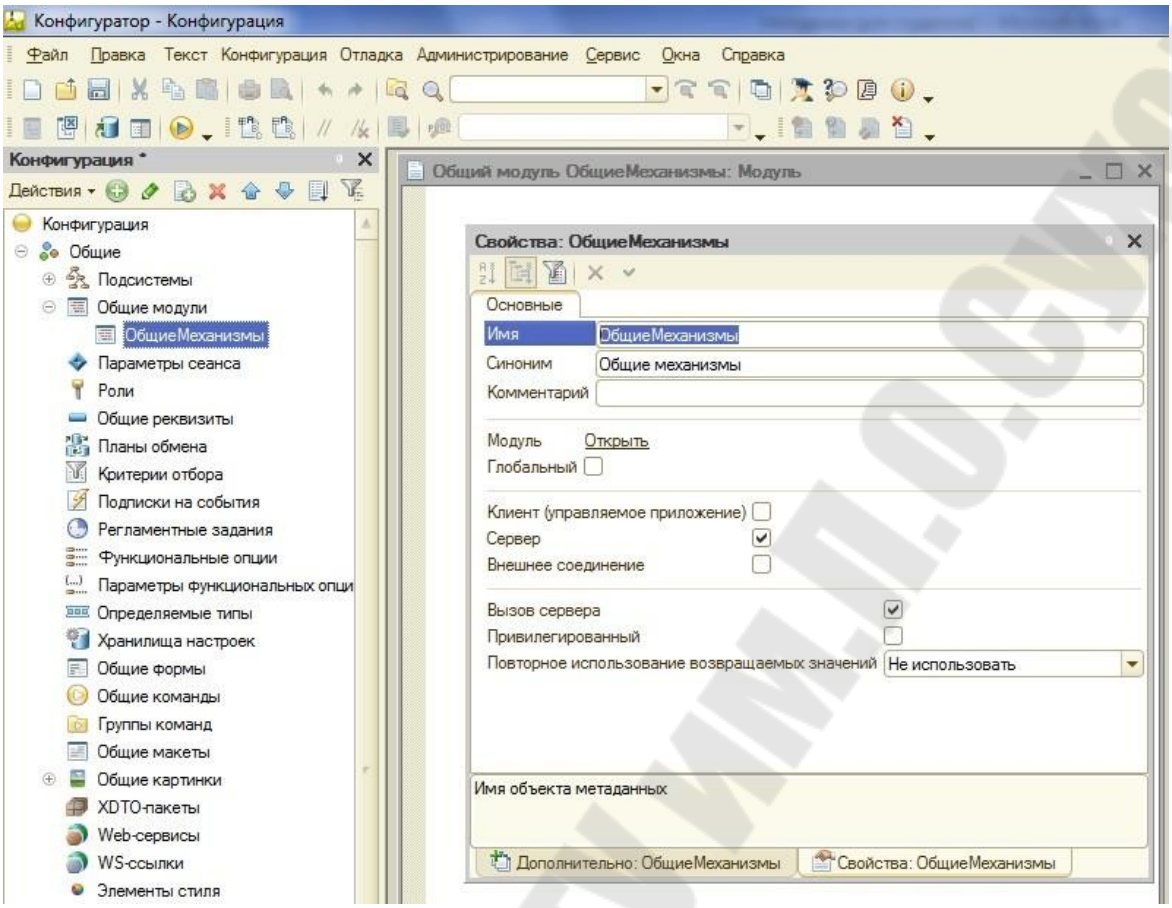


Рисунок 5.4 – Создание общего модуля

Опишем в общем модуле следующую функцию:

```
Функция ПолучитьСписокИменинников () Экспорт
ТД_День = День(ТекущаяДата());
ТД_Месяц = Месяц(ТекущаяДата());
Мас = Новый Массив;
Выборка = Справочники.Сотрудники.Выбрать();
Пока Выборка.Следующий() Цикл
    ДР_День = День(Выборка.ДатаРождения);
    ДР_Месяц = Месяц(Выборка.ДатаРождения);
    Если ДР_День = ТД_День И ДР_Месяц = ТД_Месяц Тогда
      Мас.Добавить(Выборка.Наименование);
    КонецЕсли;
КонецЦикла;
Возврат Мас;
КонецФункции
```

При описании процедуры/функции, даже если она не содержит принимаемых параметров, все равно указываются открывающая и

закрывающая скобки. Также обратите внимание: после имени функции идет ключевое слово «Экспорт». Оно необходимо для того, чтобы данная процедура/функция была видна в других частях приложения. Согласитесь, бывает необходимость в создании служебных процедур/функций, которые будут вызываться другими процедурами/функциями только этого модуля для каких-то промежуточных действий.

Поясним написанный текст. Первые две строчки содержат функции получения дня и месяца из текущей даты. Далее создается массив, который будет заполняться списком имен тех сотрудников, у кого сегодня день рождения. Следующая строчка формирует выборку всех элементов справочника «Сотрудники» и помещает их в переменную типа «Выборка данных». Переменная данного типа содержит ссылки на все элементы выбранных данных. Далее следует цикл по всем элементам сформированной выборки. Для перехода к следующему элементу используется специальный метод ***Следующий()***, который получает ссылку на следующий элемент выборки. Соответственно, по данной ссылке мы можем обратиться ко всем реквизитам справочника. Так получаем день и месяц рождения сотрудника (очередного элемент выборки). Затем проверяем: если день и месяц рождения сотрудника совпадает с текущим днем и месяцем, тогда в описанный ранее массив добавляем имя сотрудника (реквизит «Наименование»). После чего переходим к следующему элементу выборки, и т.д. пока метод ***Следующий()*** не вернет признак конца выборки т.е. значение NULL. В самом конце – функция в качестве возвращаемого значения передает весь сформированный массив.

Теперь в модуле управляемого приложения необходимо получить сформированный массив и вывести его в информационное окно. Для этого изначально обращаемся к описанной функции соответствующего общего **неглобального** модуля. Затем формируем цикл обхода всех элементов полученного массива, в котором выводим очередной элемент на экран при помощи функции ***Сообщить()***. Соответствующий программный код будет следующим:

```
СписокИменниников = ОбщиеМеханизмы.ПолучитьСписокИменинников();
Для Каждого Именинник ИЗ СписокИменниников Цикл
    Сообщить("Сегодня День рождения у "+Именинник);
КонецЦикла;
```

Запустим систему в режиме отладки и проверим работу механизма (при необходимости для ряда сотрудников укажем в качестве даты рождения текущую дату). Для большего понимания написанного программного кода необходимо обратиться к соответствующим разделам ***Синтакс-помощника***.

5.5 Использование отладки. Работа с данными заполнения

Вернемся к первоначальной задаче: как сделать так, что система автоматически заполняла реквизит «**Вид номенклатуры**» на основании того, в какой номенклатурной группе происходит создание очередного элемента справочника?

Первое, что необходимо сделать, это реализовать возможность заполнять реквизит «**Вид номенклатуры**» не только у отдельных элементов, но и у групп элементов. Соответственно, при создании очередного элемента внутри группы, данный реквизит будет «наследоваться» от «Родителя». Для этого перейдем к настройке реквизита «**Вид номенклатуры**» справочника «**Номенклатура**», и на вкладке «Использование» укажем «Использование для группы и элемента». Запустим приложение в режиме отладки, перейдем к справочнику Номенклатура и откроем любую имеющуюся группу номенклатурных позиций (допустим, «**Товары**») в режиме редактирования (используем соответствующую пункт меню в группе команд «Еще») и заполним данный реквизит.

Для всех групп внутри выбранной группы не произойдет автоматического заполнения указанного реквизита т.к. данные группы были созданы ранее, а в системе отсутствует соответствующий механизм. Поэтому на данном этапе вручную отредактируем уже имеющиеся группы товаров, и заполним данный реквизит соответствующими значениями. Затем попытаемся в любой номенклатурной группе создать новую позицию. Увидим, что реквизит «**Вид номенклатуры**» все равно не заполняется автоматически, т.к. стандартный механизм «не знает» как это делать.

Нашей целью является заполнение соответствующего реквизита при создании нового элемента. При этом неважно, какая форма будет открываться при его добавлении. Также обратим внимание на то, что при создании элемента мы обращаемся к данным группы, в которой этот элемент создается, т.е. фактически к тем данным, которые уже

хранятся в базе данных, т.е. «на сервере». Из всего этого можно сделать вывод, что требуемый программный код не имеет смысла располагать в модуле формы, т.к. она связана с интерактивной работой пользователя. Соответственно, требуемую обработку имеет смысл поместить в модуль объекта справочника «**Номенклатура**».

Открыть модуль объекта можно двумя способами:

– открыть окно редактирования объекта, перейти на вкладку «Прочее» и нажать на кнопку «Модуль объекта»;

– открыть контекстное меню объекта в дереве конфигурации и выбрать соответствующий пункт меню.

Откроем модуль объекта и посмотрим список событий, связанных с данным объектом (аналогично тому, как мы это делали при работе с модулем управляемого приложения). Среди всех событий нас будет интересовать событие ***ОбработкаЗаполнения()***. Данное событие возникает при создании новых экземпляров объекта (не при редактировании существующих). В этот момент реквизиты вновь создаваемого объекта заполняются данными из специального набора реквизитов, называемого «ДанныеЗаполнения». Соответственно, для того, чтобы объект получал требуемые значения, необходимо добавить в «ДанныеЗаполнения» требуемую информацию.

Перед тем, как добавить информацию в «ДанныеЗаполнения», необходимо определить, какой тип этих данных и что они уже содержат. Для этого воспользуемся еще одним необходимым инструментом разработчика – *отладкой*. Для этого в любом месте обработчика события ***ОбработкаЗаполнения()*** добавим текст, к примеру, ***А = 1;***. Обновим конфигурацию базы данных, и затем, при помощи контекстного меню, установим в данной части процедуры точку останова (рис. 5.5).

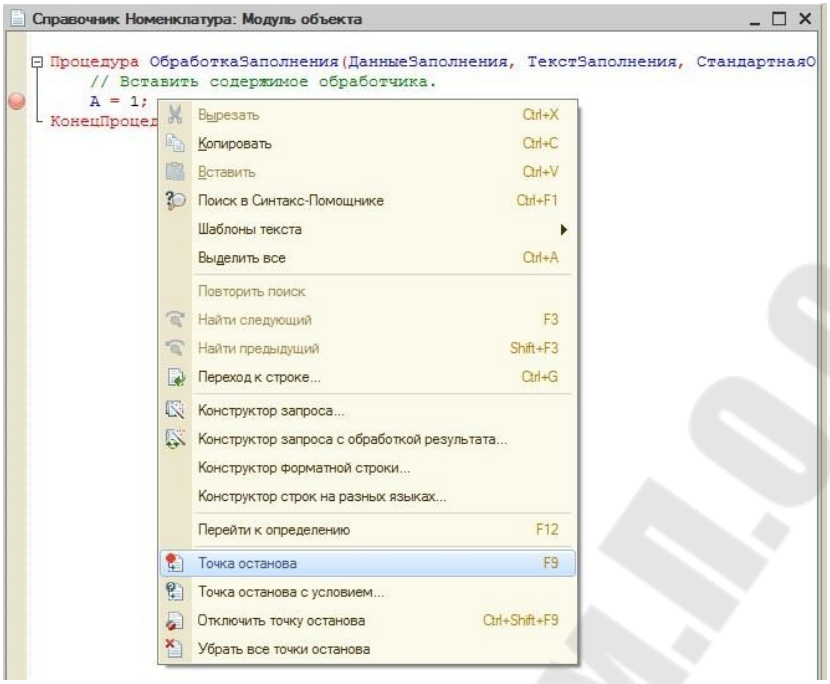


Рисунок 5.5 – Добавление точки останова для отладки приложения

Запустим приложение в режиме отладки. Откроем справочник «**Номенклатура**» и попытаемся создать новый элемент. При этом система автоматически перебрасывает нас в «Конфигуратор», в то самое место, где поставили точку останова. На панели инструментов становятся доступны инструменты отладки (рис. 5.6). Соответствующие инструменты также доступны через контекстное меню в модуле и через меню «Отладка».

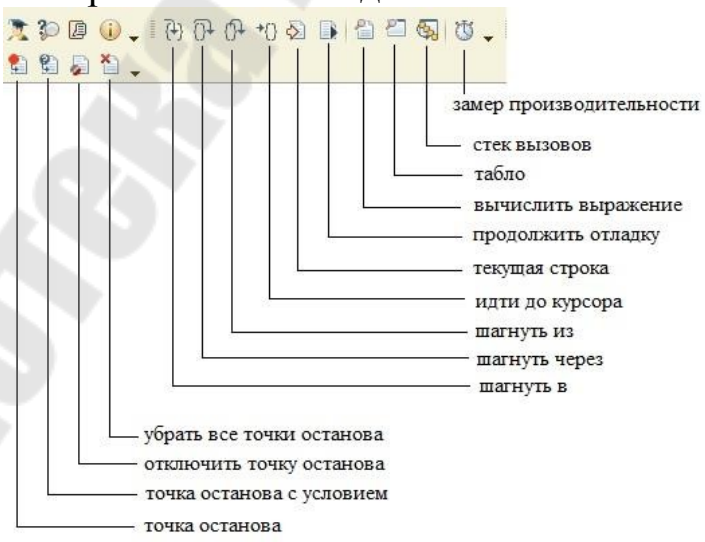


Рисунок 5.6 – Панель отладки

В процессе отладки возможна установка новых точек останова без перезапуска приложения, движение по точкам останова, переход к следующим строчкам кода и т.д. В наибольшей степени нас будут интересовать следующие инструменты (рис. 5.6):

– «**вычислить выражение**» – используется для определения значения различных объектов в текущий момент выполнения программы (можно выделить непосредственно объект, переменную и система покажет данные выделенного объекта); можно определять получает ли программа в нужный момент времени доступ к определенным данным, а также, какие данные содержатся в тех или иных объектах, их корректность и т.д.;

– «**табло**» – служит для добавления данных, которые могут меняться в процессе работы программы, тем самым имеется возможность отслеживать правильность выполнения кода, заполнения реквизитов, выполнения расчетов и т.д.;

– «**шагнуть через**» – переход к следующей строчке программного кода и ее выполнение;

– «**шагнуть в**» и «**шагнуть из**» связаны с переходом в/из некоторого цикла, содержащего программный код;

– «**продолжить отладку**» – вернуться в режим исполнения для дальнейших действий пользователя;

Выделим переменную «ДанныеЗаполнения» (в параметрах обработчика события) и нажмем инструмент «Вычислить выражение». В результате появится окно (рис. 5.7).

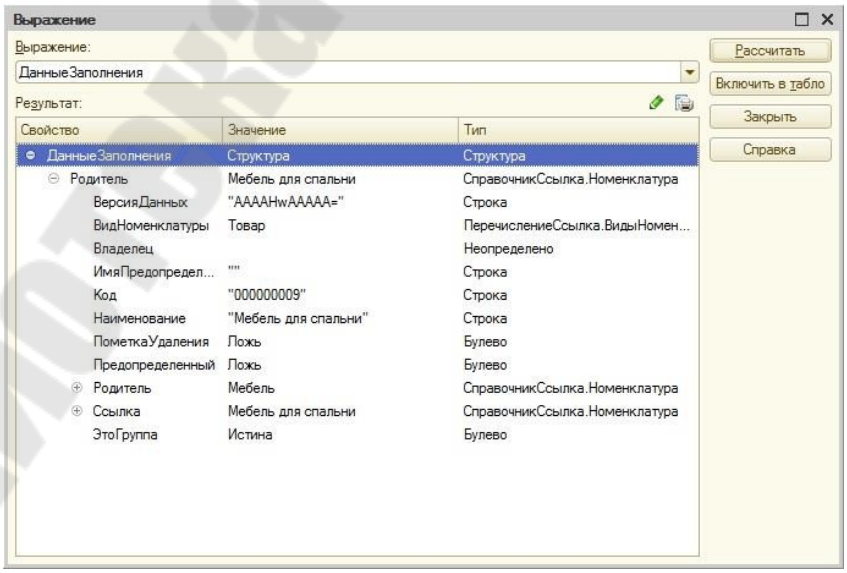


Рисунок 5.7 – Окно «Вычислить выражение» в режиме отладки

На основе данных, имеющихся в этом окне можно сделать ряд важных выводов:

– переменная «ДанныеЗаполнения» имеет особый тип – Структура;

– в «ДанныеЗаполнения» передается лишь значение поля «Родитель».

Поле «Родитель» – это реквизит ссылочного типа, соответственно, имеется возможность его «раскрыть» и подробно посмотреть его содержимое (значения всех реквизитов). В том случае, если в оке «Вычислить выражение» оказываются не только реквизиты ссылочного типа, а, табличные части, то для просмотра содержимого табличной части необходимо ее выделить, после чего нажать кнопку «Показать значение в отдельном окне»  (рис. 5.7). Для удобства, аналогичные действия можно выполнить с любыми данными в окне «Вычислить выражение».

Закроем окно «Вычислить выражение» и завершим отладку (через соответствующий пункт меню «Отладка»).

Самый простой способ решить поставленную задачу – это принудительное заполнение значения реквизита «Вид номенклатуры» таким же значением, как и у группы, в которой данный элемент создается, т.е. у «Родителя». У Родителя значение реквизита ВидНоменклатуры можно получить из «ДанныхЗаполнения» (рис. 5.7). Добавим в обработчик следующий код:

```
ВидНоменклатуры = ДанныеЗаполнения.Родитель.ВидНоменклатуры;
```

Проверим, что точка останова находится там же, где и раньше. Запустим систему в режиме отладки. В момент остановки выполнения, запустим инструмент «Вычислить выражение», в поле «Выражение» введем имя реквизита, значение которого мы хотим увидеть (**ВидНоменклатуры**), и нажмем кнопку «Рассчитать» (см. рис. 5.7). В результате увидим, что значение реквизита заполнилось верными данными.

Этот код достаточно прост и правильно работает. А теперь попытаемся создать новый элемент справочника в его корне (там, где у нас имеются предопределенные группы «**Товары**» и «**Услуги**»). Возникающая ошибка (рис. 5.8) связана с тем, что у элементов,

находящихся в корне справочника поле «Родитель» не заполнено, а «ДанныеЗаполнения» не определены.

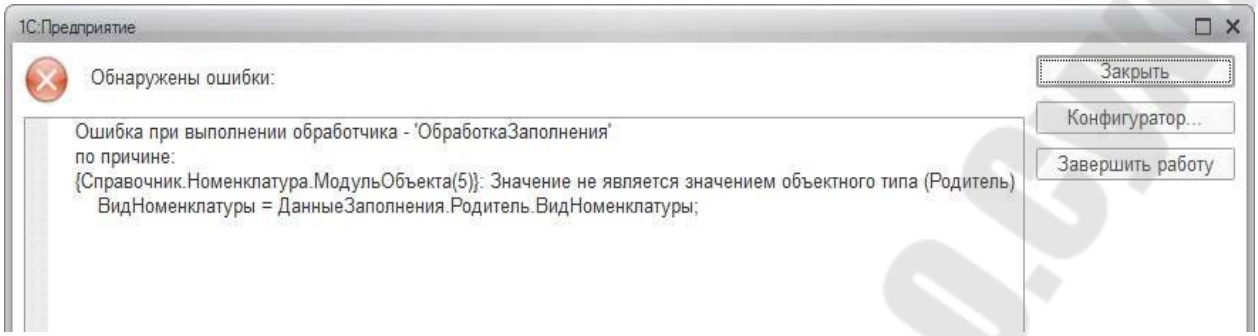


Рис. 5.7 – Ошибка при создании элемента справочника в его корне

Для исключения данной ошибки перед заполнением реквизита «**Вид номенклатуры**» необходимо проверить – заполнена структура «**ДанныеЗаполнения**» или нет. Для этого выше добавим следующий код:

```
Если ДанныеЗаполнения = Неопределено
    Тогда Возврат;
КонецЕсли;
```

Напоминаем, что НЕОПРЕДЕЛЕНО означает, что значение принципиально может существовать, но не заполнено в данный момент. «**Возврат**» означает прерывание работы процедуры/функции, в которой указан данный метод, в данном случае процедура ***ОбработкаЗаполнения()*** прерывается, и мы продолжаем работу с системой.

Запустите систему в режиме отладки и проверьте работоспособность. Обратите внимание, что процедура действительно прерывается т.к. мы не доходим до точки останова (она пока что не отключена).

И так, разобрались с тем, что же такое «ДанныеЗаполнения», для чего они нужны. Но пока что никак не воспользовались информацией о том, что это Структура. На самом деле, «ДанныеЗаполнения» могут содержать то, что в них будет передано. И возможно возникновение ситуации, когда «ДанныеЗаполнения» не будут иметь тип «Неопределено», т.е. в них будут переданы какие-то данные, но среди этих данных будет отсутствовать поле «**Родитель**». Такое возможно при различных программных обращениях к

справочнику «**Номенклатура**», когда отключается стандартное поведение системы и разработчик меняет функциональность по своему усмотрению.

Соответственно, задача дополнительно сводится к тому, чтобы проверить – имеется ли в «ДанныхЗаполнения» поле «**Родитель**». Для решения данной задачи вначале следует обратиться к Синтакс-помощнику и определить, что представляет собой тип данных «Структура» (найти его можно среди «Универсальных коллекций значений»). Открыв соответствующий раздел, увидим, что структура – это набор «***ключ – значение***». В качестве ключей используются имена полей (реквизитов), в нашем случае – это поле «Родитель». В качестве значения – значения конкретных реквизитов. Таким образом, при создании нового элемента, структура «ДанныеЗаполнения» должна, к примеру, выглядеть следующим образом:

```
ДанныеЗаполнения = («Родитель», Мебель для спальни)
```

Понятно, что «**Мебель для спальни**» – это представление. Реально же там содержится ссылка в виде хэш-строки. Также в справке по типу данных «Структура» видим, что к значениям, хранящимся в ней, можно обращаться по имени ключа, что мы и неосознанно сделали, когда написали самый первый программный код по заполнению реквизита «**Вид номенклатуры**». Также объекты типа «Структура» имеют ряд методов, в частности, метод «Свойство», который определяет, присутствует ли заданный ключ в структуре или нет, что требуется для проверки того, что в «ДанныеЗаполнения» попадает поле «**Родитель**» со всеми его реквизитами. Поэтомупосле проверки на наличие «ДанныеЗаполнения» добавим следующую проверку перед заполнением реквизита «**Вид номенклатуры**»:

```
Если    ДанныеЗаполнения.Свойство("Родитель") Тогда
ВидНоменклатуры = ДанныеЗаполнения.Родитель.ВидНоменклатуры;
КонецЕсли;
```

Запустим систему в режиме отладки и посмотрим, что все наши действия привели к желаемому результату. Если точка останова все еще имеется, то ее можно удалить вместе с содержащимся в ней кодом, после чего обновить конфигурацию базы данных.

6 ДОКУМЕНТЫ. КОНСТРУКТОР ФОРМЫ

6.1 Документы

Документ – объект, отражающий факт хозяйственной деятельности на предприятии. К документам относятся документы поступления и продажи товара, приказы о начислении заработной платы, премировании сотрудников, формирование цен на товары и услуги и т.д. Ключевыми особенностями документа являются: наличие даты (точное положение на оси времени, к которому привязывается документ) и номера документа, а также возможность его проведения (специальное действие, которое меняет учет, к примеру, материальное или финансовое состояние на предприятии).

Документы, как и справочники, являются типообразующими объектами. Ссылка на документ формируется на основе даты создания и типа документа. К основным реквизитам, характеризующим документы одного вида, относят дату и номер. Перечень всех документов одного вида располагается в порядке времени их создания. Т.к. в одну секунду может быть создано множество документов одного вида (вплоть до нескольких тысяч), то для правильной хронологической последовательности, кроме даты, каждый документ дополнительно характеризуется моментом времени, который формируется на основании даты и ссылки на документ.

Создадим документ «**Приходная накладная**», который будет отображать факт поступления товаров на предприятие. Добавим его в подсистему «**Отдел закупок**».

На вкладке «Данные» откроем список «Стандартных реквизитов», увидим, что «**Дата**» и «**Номер**» уже есть, поэтому нет необходимости их добавлять. Добавим следующие реквизиты шапки:

- **Контрагент**: тип СправочникСсылка.Контрагенты;
- **Договор**: тип СправочникСсылка.Договора;
- **Сотрудник**: тип СправочникСсылка.Сотрудники;
- **Сумма документа**: тип Число, длина 15, точность 2.

Добавим табличную часть «**Товары**» (в которой будем отображать поступление товаров), создадим реквизиты табличной части:

- **Номенклатура**: тип СправочникСсылка.Номенклатура;
- **Цена**: тип Число, длина 15, точность 2;

– **Количество**: тип Число, длина 15, точность 2;
– **Сумма**: тип Число, длина 15, точность 2.

Откроем вкладку «Нумерация» окна редактирования документа. Нумерация отвечает за то, как будет формироваться номер документа. Как правило, рекомендуется оставлять автоматическую нумерацию с контролем уникальности. При этом, для документов можно указать периодичность. Непериодические документы будут иметь сквозную нумерацию в течение всего времени существования системы. Если установить некий период, то в течение этого периода нумерация будет уникальная, но по окончании, когда начнется новый период, нумерация начнется сначала. В организациях очень часто таким периодом является год. Настройка «Нумератор» нужна для того, чтобы организовать в системе сквозную нумерацию для документов различного вида, у каждого из которых указать один и тот же нумератор (создается отдельно). Оставим настройки по умолчанию.

Запустим систему в режиме отладки и попробуем ввести первую приходную. И практически сразу мы увидим следующие недостатки:

1. в качестве Контрагента можно выбрать как Поставщика, так и Покупателя;
2. отображается список всех договоров, а не только тех, которые заключены с выбранным Контрагентом;
3. можно выбрать приход, как Товаров, так и Услуг;
4. сумма не считается автоматически как цена, умноженная на количество.

Попробуем исправить все вышеперечисленные недостатки. При выборе Контрагента необходимо осуществлять выбор лишь тех, кто находятся в группе «**Поставщики**». Т.к. группа «**Поставщики**» предопределенная, то можно на нее ссылаться при использовании различных механизмов и написании программного кода. Для решения поставленной задачи необходимо настроить соответствующее отбор-условие и наложить его на реквизит «**Контрагент**». Для этого открываем свойства реквизита, переходим на вкладку «Представление», открываем настройку свойства «Параметры выбора» (рис. 6.1). Настройки, указанные в «Параметрах выбора», будут накладывать условия отбора на реквизиты элементов справочника «Контрагенты» (т.е. того справочника, на элементы которого мы ссылаемся в данном реквизите) в открывающейся форме

выбора. Таким образом, ограничиваем список возможных элементов справочника, представляемых в форме выбора.

Т.к. справочник Контрагенты имеет всего 2 уровня иерархии, на первом из которых указана группа («**Покупатели**» или «**Поставщики**»), а на втором – конкретные Контрагенты, то данные группы являются «родителями» для всех контрагентов внутри них. Таким образом, мы должны наложить отбор на поле «**Родитель**» при выборе элементов из справочника «**Контрагенты**».

В открывшемся окне настроек параметров выбора в панели инструментов нажимаем кнопку  для добавления отбора и в раскрывшемся списке (рис. 6.1 a) среди всех реквизитов справочника «**Контрагенты**» выбираем реквизит «**Родитель**».

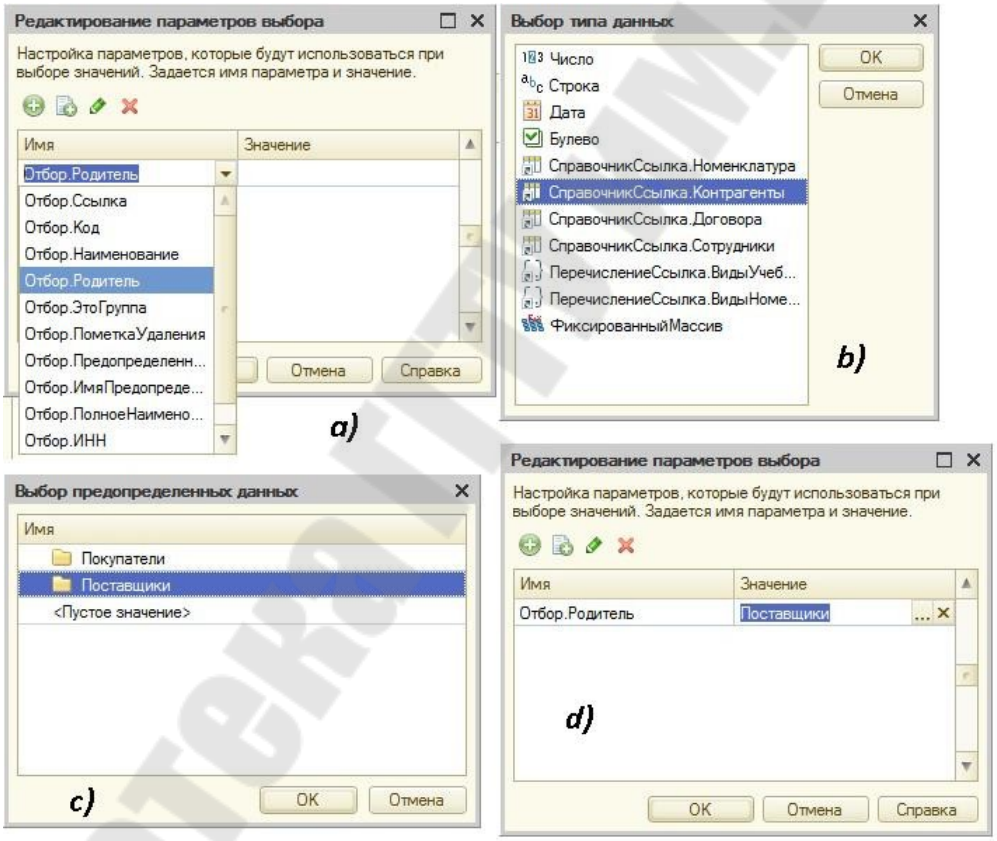


Рисунок 6.1 – Настройка «Параметров выбора»

В столбце «Значение» при помощи  необходимо выбрать тип значения, которое будет накладываться как условие отбора (рис. 6.1 b). Т.к. значение содержится в самом справочнике «Контрагенты», то указываем СправочникСсылка.Контрагенты. Затем для выбора

конкретного значения нажимаем ..., и из предопределенных данных (рис. 6.1 c) выбираем группу «Поставщики». После выбора требуемых данных отбор выглядит так, как показано на рис. 6.1 d. При необходимости можно наложить и другие условия отбора.

Для ограничения формы выбора «**Договора**» элементами справочника, связанными с выбранным Контрагентом, необходимо реквизиты «**Контрагент**» и «**Договор**» связать между собой. Для связи реквизитов между собой в пределах одного объекта используется настройка «Связи параметров выбора», расположенная на вкладке «Представление» (рис. 6.2). Выполним такую настройку для реквизита «**Договор**», т.к. на его форму выбора должны быть наложены условия отбора. В окне «Связей параметров выбора» слева указываются другие реквизиты текущего объекта (рис. 6.2 a), от значений которых будет зависеть форма выбора настраиваемого реквизита. В нашем случае это реквизит «**Контрагент**»: выбираем и переносим его в правую часть окна.

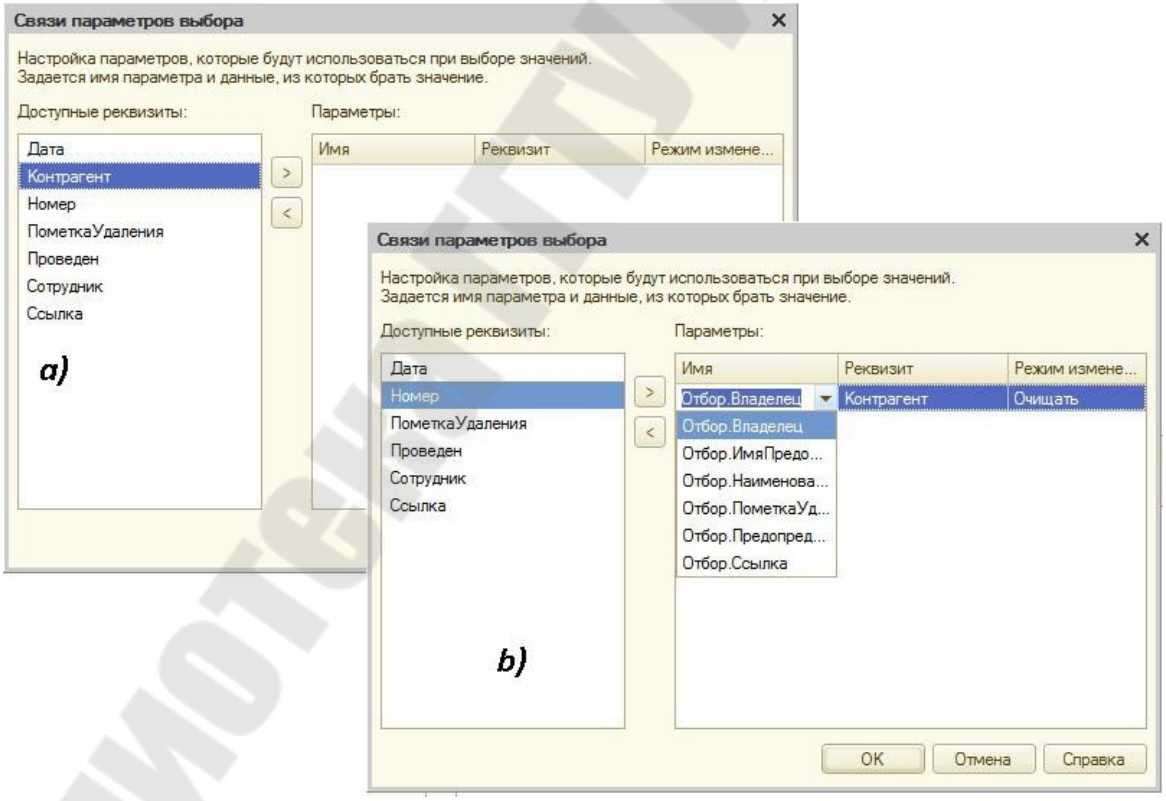


Рисунок 6.2 – Настройка «Связи параметров выбора»

Для данного реквизита по умолчанию указывается отбор по владельцу (рис. 6.2 b). Это говорит о том, что в форме выбора

настраиваемого реквизита («**Договор**») будут отображены лишь те элементы, для которых поле «**Владелец**» заполнено значением, указанным в выбранном реквизите («**Контрагент**»). В нашем случае это действительно так, и мы оставляем данную настройку отбора. Однако среди различных вариантов может быть указан отбор и по другим полям настраиваемого реквизита (рис. 6.2 b). Дополнительным параметром при настройке отбора является «Режим изменения» связанного реквизита (рис. 6.2 b). Режим «Очищать» говорит о том, что при изменении связанного реквизита («**Контрагент**») автоматически будет очищаться настраиваемый реквизит («**Договор**»). В большинстве случаев так и требуется, тем не менее, данную настройку при необходимости можно и изменить.

Для настройки выбора только Товаров, а не Услуг, в качестве значений реквизита «**Номенклатура**» табличной части «**Товары**», следует воспользоваться настройкой «Параметров выбора» (рис. 6.1). Настраиваемым реквизитом при этом будет, как нетрудно догадаться, «**Вид номенклатуры**». Тип значения, соответственно, будет ПеречислениеСсылка.ВидыНоменклатуры, откуда в качестве значения выбирается «**Товар**».

6.2 Конструктор управляемой формы. Модуль формы

Таким образом, первые три проблемы из обозначенного списка были решены с использованием стандартных механизмов платформы. Однако настройку автоматического расчета суммы таким же образом не сделать. Система просто не может знать алгоритм, который бы брал данные из двух, трех и т.д. ячеек табличной части документа, что-то с ними делал и помещал бы в третью ячейку: таких алгоритмов может быть сколь угодно много и всех их предусмотреть просто невозможно. Поэтому необходимо систему «научить» нужному алгоритму с использованием встроенного языка 1С.

Мы заполняем табличную часть в режиме исполнения: выбираем номенклатуру, вводим цену, количество. Т.е. все действия, которые выполняются – выполняются интерактивно. Соответственно, обработку необходимо применить к интерактивным действиям пользователя, т.е. соответствующий программный код должен располагаться в модуле формы. Поэтому в окне редактирования объекта переходим на вкладку «Формы» (рис. 6.3).

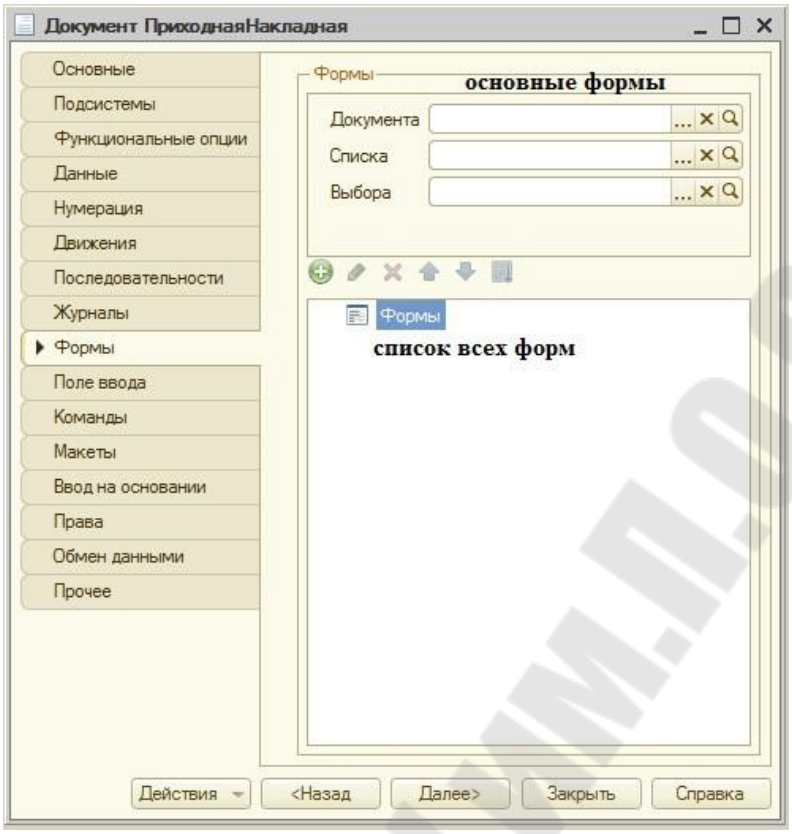


Рисунок 6.3 – Настройка форм объекта

В верхней части окна «Формы» располагается перечень возможных типов форм (элемента, списка и т.д.) для объекта данного вида. Для каждого типа назначается форма, являющаяся основной. Программист может создать сколько угодно большое число форм одного типа, отличающихся функциональностью. Список всех форм располагается ниже панели инструментов (рис. 6.3). Для объекта, из всех форм определенного типа, только одна является основной. *Основная форма* – форма, загружаемая по умолчанию, при интерактивном обращении к ней в режиме исполнения. Все остальные формы могут загружаться программно в результате работы различных алгоритмов. Для создания основной формы необходимо выбрать требуемый тип формы и нажать , либо выбрать нужную форму из списка всех форм при помощи ...

При создании/редактировании экземпляра документа открывается форма элемента (форма документа). Сейчас система ее создает автоматически, нам же данный механизм необходимо изменить. Нажимаем для создания основной формы документа, в

результате запускается специальный мастер. Оставляем все параметры по умолчанию. По окончании работы мастера запускается конструктор формы документа (рис. 6.4).

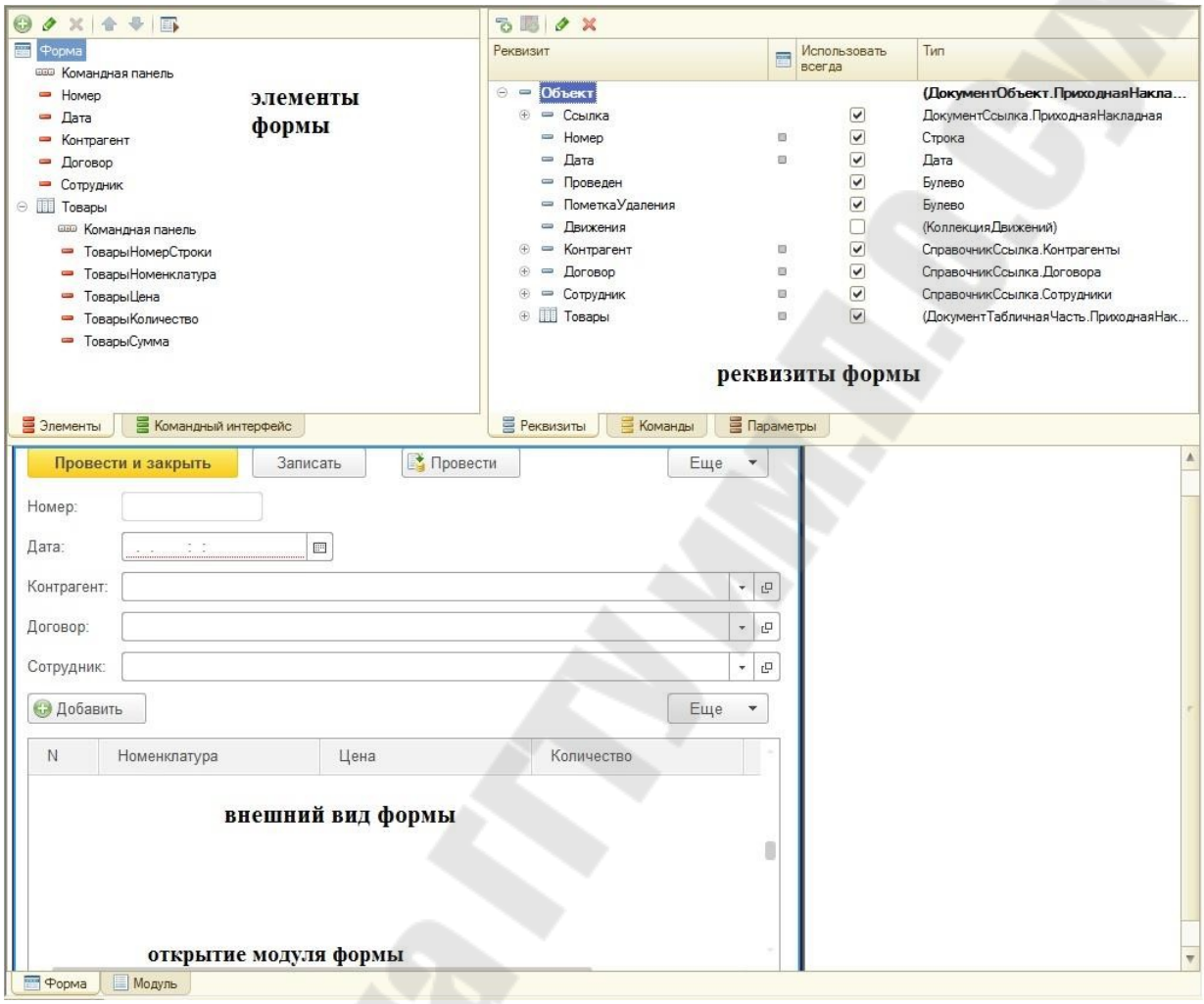


Рисунок 6.4 – Конструктор формы

В верхней правой части конструктора формы располагаются элементы, отвечающие за данные формы. Сюда относятся, в первую очередь, реквизиты формы, команды формы, которые могут быть расположены в виде кнопок или гиперссылок. Это те данные, которые, которые отвечают за функциональность формы. Также здесь можно добавить параметры формы, которые либо будут передаваться в другие открываемые формы, либо использоваться в различных программных алгоритмах.

В левой верхней части конструктора располагается перечень интерактивных элементов – т.е. тех элементов, которые будут

расположены на форме, и которые будут доступны для редактирования, использования конечным пользователем. Здесь производится настройка внешнего интерфейса: как элементы будут располагаться, как группироваться и т.д. Интерактивные элементы, выполняющие какие-либо действия, а не являющиеся просто декорациями, связаны с реквизитами и командами формы (через специальное свойство «ПутьКДанным»). Также здесь настраивается командный интерфейс: панель навигации для перехода к определенным разделам приложения и командная панель.

Внизу отображается внешний вид – как создаваемая форма будет выглядеть в режиме исполнения. Также имеется закладка для перехода к модулю формы – это то место, где располагается программный код, описание обработчиков событий. В модуле формы описываются как некоторые служебные (вспомогательные) процедуры и функции, так и обработчики, связанные с элементами формы. Каждый элемент формы (и сама форма в целом) имеет ряд событий, которые могут возникать в результате работы системы. Для просмотра списка всех событий необходимо открыть свойства конкретного элемента в дереве элементов (см. рис. 6.4), и затем перейти к вкладке «События» (рис. 6.5).

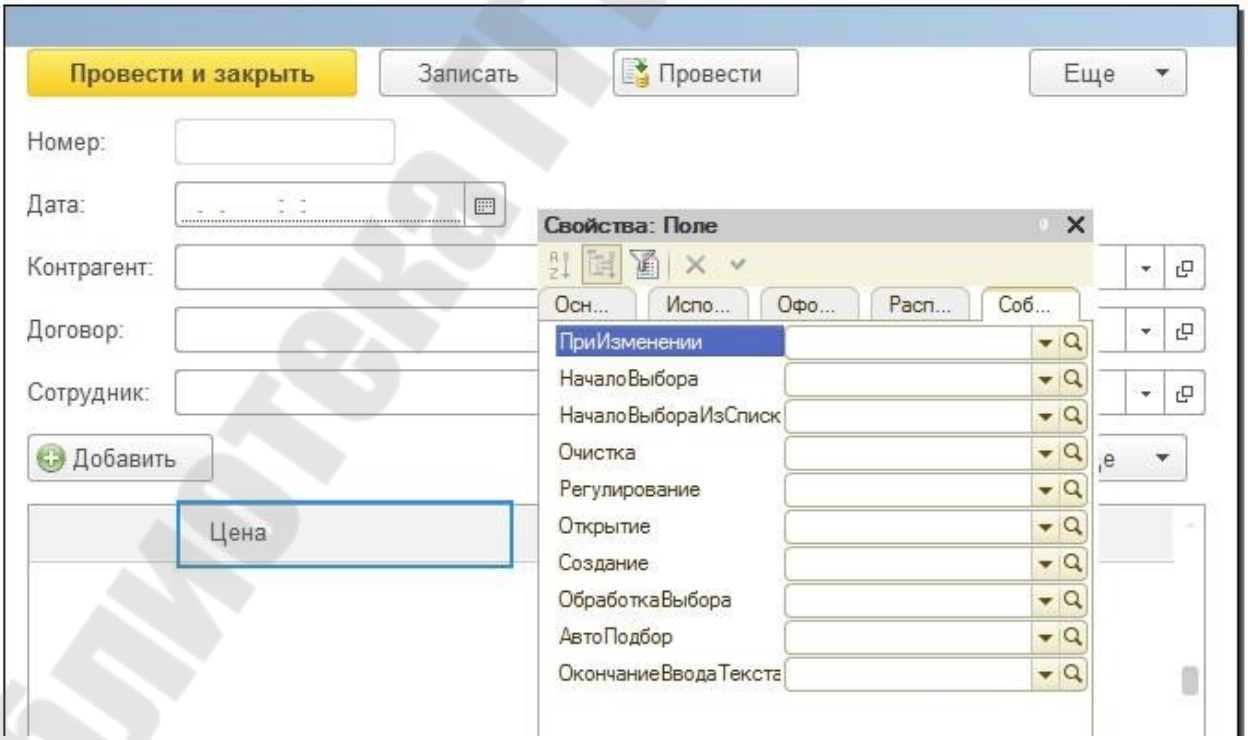


Рисунок 6.5 – События элементов формы

Каждое событие интерактивного элемента формы может быть связано лишь с одной процедурой – обработчиком события. Имя процедуры может быть произвольным. Сами процедуры-обработчики описываются в модуле формы.

Для того, чтобы связать событие с процедурой, которая станет обработчиком данного события, используется выпадающий список  , в котором из списка всех процедур указывается та, которую следует назначить в качестве обработчика.

Для автоматического создания процедуры-обработчика определенного события и связи события и его обработчика, можно использовать кнопку  около требуемого события элемента.

Следует обратить внимание, что часть событий выполняется на стороне клиента, а часть – на стороне сервере. Поэтому в модуле формы любую процедуру, функцию (не только обработчики событий) необходимо предварять директивой, указывающей, где происходит выполнение программного кода.

Для реализации автоматического расчета Суммы, необходимо для элемента «**Цена**» описать обработчик события «**При изменении**». Система задает вопрос, какие процедуры необходимо создать:

– обработчик на клиенте;

– обработчик на клиенте и безконтекстную процедуру на сервере (для случая, когда необходимо получать какие-то данные с сервера);

– обработчик на клиенте и контекстную серверную процедуры (аналогично предыдущему случаю, но передается весь контекст формы).

Для нас будет достаточно обработчика на клиенте т.к. никакие данные с сервера (из базы данных нам не потребуются). В результате создается шаблон процедуры-обработчика.

Теперь необходимо определить, как обратиться к данным, которые сейчас находятся в табличной части документа? Для этого воспользуемся отладкой: введем в процедуре заглушку ***A=1;*** и установим в этой строке точку останова. Запустим систему в режиме отладки и попытаемся создать один документ «**Приходная накладная**».

Добавим строку табличной части и введем в поле «**Цена**», допустим, «**5**». Система перебрасывает нас в модуль объекта в точку

останова. Используем инструмент «Вычислить выражение». Можем попытаться обратиться к значениям реквизитов или элементов формы.

Все реквизиты хранятся в основном реквизите – «**Объект**». В поле «Вычислить» вводим «Объект», в результате получаем значения реквизитов (рис. 6.6).

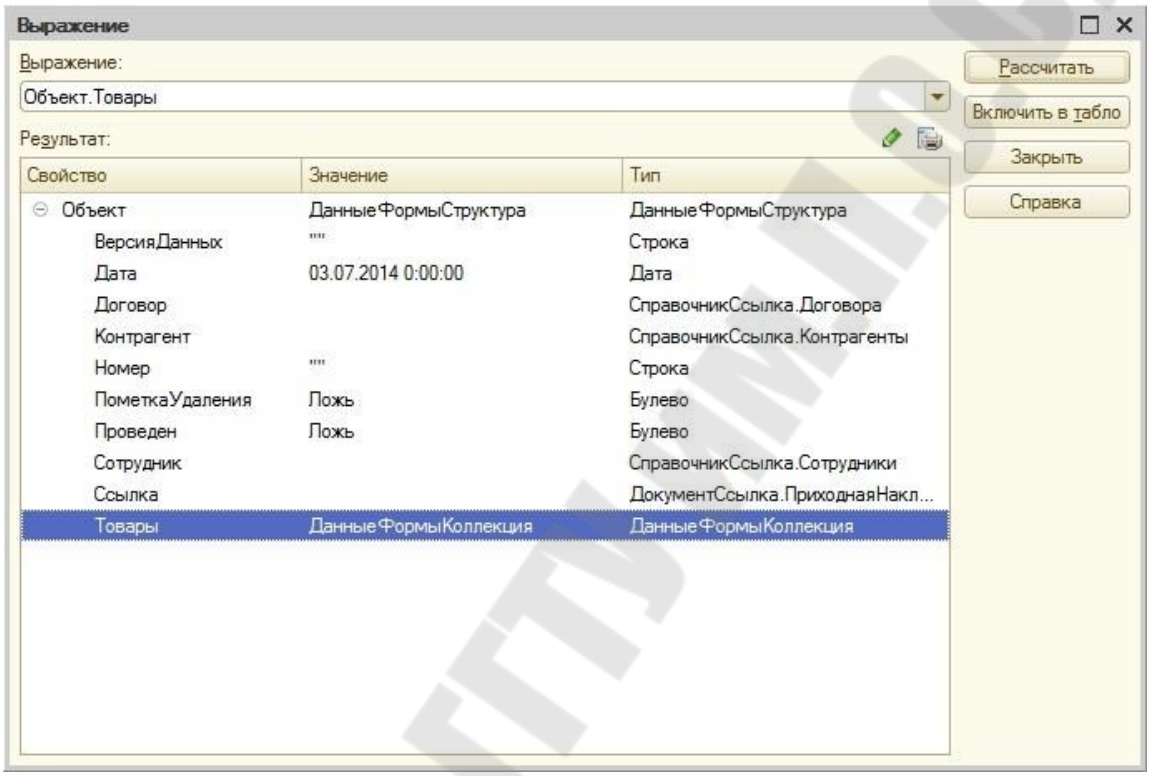


Рисунок 6.6 – Содержимое реквизитов документа

Однако тем самым не получили желаемого: не видим строку табличной части с ценой.

Известно, что табличная часть называется «Товары», поэтому находим ее среди всех полученных реквизитов (рис. 6.6). Данный реквизит имеет тип ДанныеФормыКоллекция. Коллекцию значений мы можем посмотреть в отдельном окне при помощи кнопки , в результате чего открывается табличная часть «Товары» (рис. 6.7).

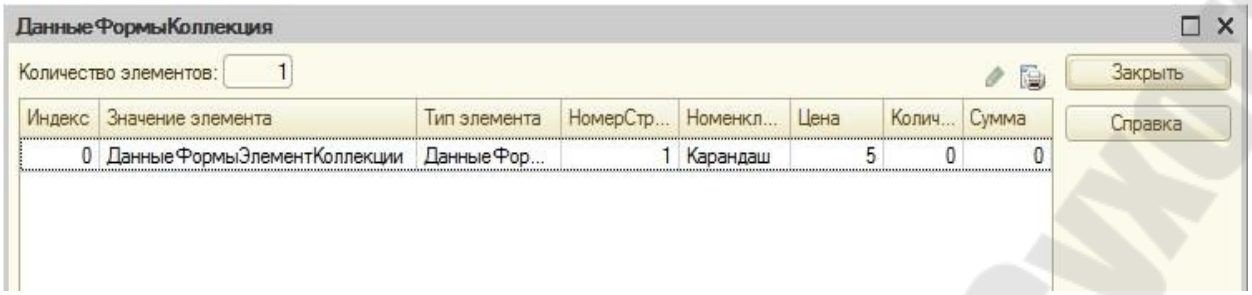


Рис. 6.7 – Просмотр табличной части в отдельном окне

Здесь видим наш товар с заданной ценой, но опять – нет ничего, чтобы указывало на то, как к этому полю обратиться. Пробуем просмотреть соответствующую запись также в отдельном окне (рис. 6.8).

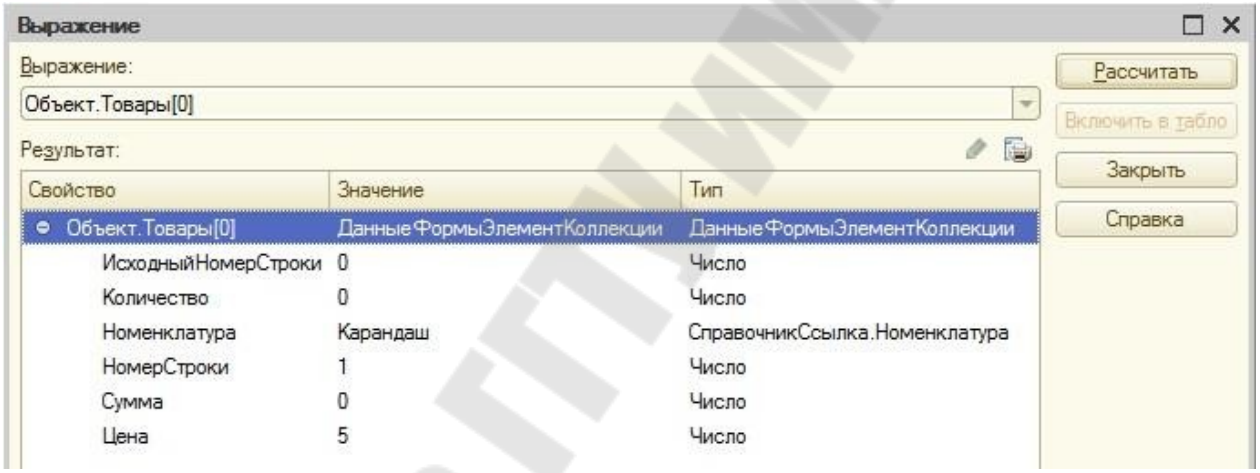


Рисунок 6.8 – Просмотр выделенной строки табличной части в отдельном окне

В результате срабатывает инструмент «Вычислить выражение» для конкретной строки табличной части: **Объект.Товары[0]**. Данный вариант невозможен к использованию, т.к. строк может быть великое множество, и нам будет неизвестно, в какой конкретно строке мы находимся в данный момент. Таким образом, поиск решения среди реквизитов объекта не дал желаемого результата.

Для обращения к элементам формы следует обратиться к коллекции «Элементы». В результате мы получим окно, содержащее гораздо больше элементов, доступных для просмотра (рис. 6.9).

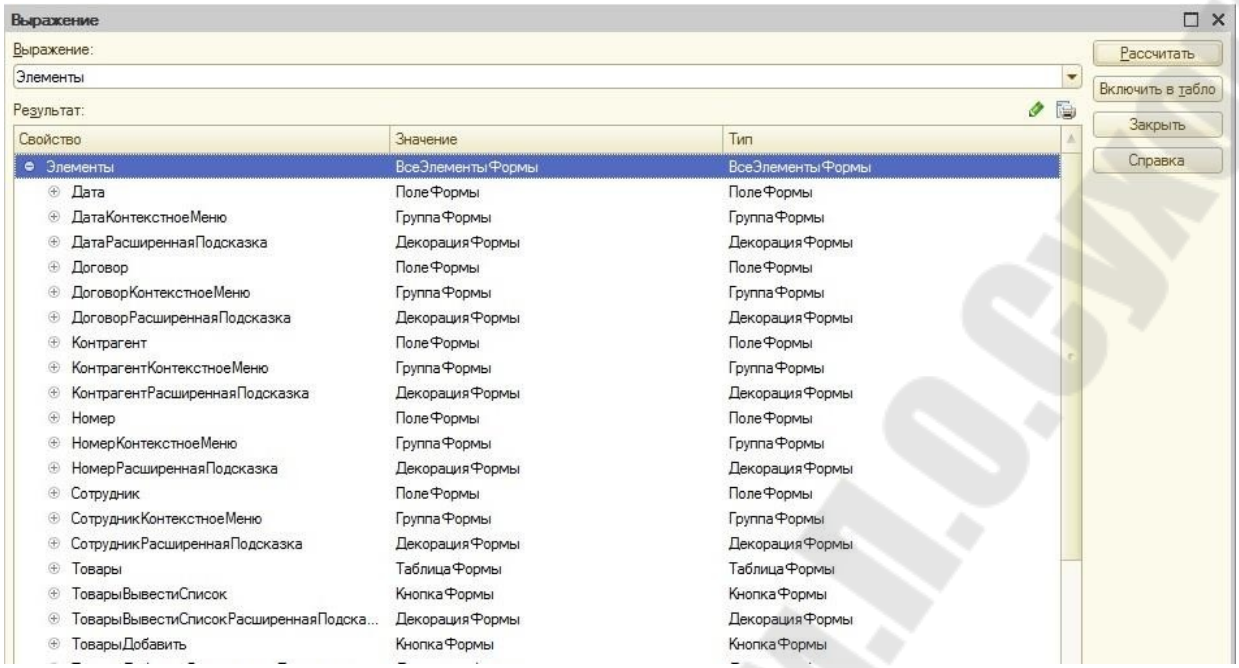


Рисунок 6.9 – Список интерактивных элементов, доступных для просмотра при отладке

Это на самом деле ожидаемо т.к. элементов пользовательского интерфейса достаточно много, настроек и параметров и характеристик – тоже. Главная проблема – как среди всего этого множества отыскать то, что требуется именно нам.

В первую очередь поиск необходимо начинать по основному наименованию элемента. Табличная часть называется «**Товары**», поэтому в первую очередь и ищем – **Элементы.Товары**. Раскроем найденную группу «**Товары**». Табличная часть имеет тип – ДанныеФормыКоллекция (см. рис. 6.6), а конкретная строка табличной части – ДанныеФормыЭлементКоллекции. Среди перечня всех элементов найдем элемент соответствующего типа. Это элемент под названием **ТекущиеДанные**. Раскрыв данный элемент, увидим данные в строке, в которой в данный момент находимся. Более того, выделив поле «**Цена**», в поле «Выражение» инструмента «Вычислить выражение» увидим полный путь к этому полю: **Элементы.Товары.ТекущиеДанные.Цена**, что нам и требовалось получить. Аналогичным образом, осуществляется доступ к содержимому полей «**Количество**» и «**Сумма**».

Закрываем отладку приложения, возвращаемся к редактированию модуля формы документа. В созданной процедуре-

обработчике убираем точку останова и текст-заглушку. Пишем следующий программный код:

```
СтрокаРасчета = Элементы.Товары.ТекущиеДанные;
СтрокаРасчета.Сумма = СтрокаРасчета.Цена*СтрокаРасчета.Количество;
```

Первая строчка содержит в себе строку-ссылку на текущие данные (введена для большей читабельности кода). Вторая строка производит требуемые действия с элементами, входящими в указанную строку-ссылку.

Аналогичным образом создадим обработчик события ПриИзменении» для поля «Количество» и добавим в него точно такие же строчки программного кода.

Запустим систему в режиме отладки и попробуем добавить приходную накладную теперь. В конце нажимаем кнопку «Записать» (без проведения), после чего закрываем текущую приходную.

Создадим документ «**Расходная накладная**» копированием. В системе «1С: Предприятие» очень сильно развит инструмент создания объектов (прикладных и подчиненных) «на основании» других. Для того, чтобы просто создать точную копию объекта, необходимо в дереве конфигурации выделить требуемый объект и нажать кнопку «Добавить копированием»  , либо использовать функциональную клавишу F9. После этого необходимо изменить имя прикладного объекта, и выполнить требуемую настройку реквизитов и поведения объекта. Реквизиты шапки, табличные части и их реквизиты объекта также можно добавлять копированием.

Также имеется возможность создания на основе прикладного объекта одного вида, прикладного объекта другой сущности (не стоит путать с инструментом «Ввод на основании», далее в пособии). К примеру, из объекта типа «Справочник» создать объект типа «Документ». Для этого необходимо в дереве конфигурации выделить исходный объект и перетащить его на класс объектов, экземпляр которого необходимо создать. При этом скопируются только те настройки, реквизиты, табличные части (и их реквизиты) которые могут существовать у объектов различных классов и имеют одинаковый тип данных.

В нашем примере не будем создавать копированием объекты различных сущностей. Выделим документ «**Приходная накладная**» и при помощи «Добавить копированием» создадим

новый документ «**Расходная накладная**». Включим документ в подсистему «**Отдел продаж**». Для реквизита «**Контрагент**» изменим настройку «Параметры выбора»: теперь мы должны выбирать контрагентов из списка «**Покупателей**».

Для реквизита «**Номенклатура**» табличной части «**Товары**» сбросим настройку «Параметры выбора» – мы продаем не только товары, но и оказываем услуги, и все это делаем в рамках одного документа «Расходная накладная».

После того, как внесли все необходимые изменения, можно запустить систему в режиме отладки и посмотреть на результат. Если вдруг оказалось, что при создании документа копированием, система сохранила условия отбора, накладываемые на формы выбора, как у исходного документа (к примеру, можно выбрать только контрагентов из списка поставщиков) – тогда необходимо удалить форму документа «**Расходная накладная**» и просто заново ее создать с описанием требуемого функционала.

Обратим внимание на следующее: в модуле формы документа «**Приходная накладная**» для события ПриИзменении полей «**Цена**» и «**Количество**» был прописан одинаковый программный код. Создав копированием документ «**Расходная накладная**», данный код также оказался дважды написан в модуле документа. Таким образом, одни и те же действия несколько раз были описаны в различных частях конфигурации. Это не совсем рационально. Правильнее этот программный код описать в одном единственном месте (в общем модуле), а в остальных частях конфигурации делать вызов соответствующей процедуры/функции.

Создадим общий модуль «**Работа с документами**», который будет выполняться исключительно на клиенте (установите соответствующую галочку в настройках модуля). Добавим в модуль следующую процедуру:

```
Процедура ПересчитатьСумму(СтрокаРасчета) Экспорт
    СтрокаРасчета.Сумма = СтрокаРасчета.Цена*СтрокаРасчета.Количество;
КонецПроцедуры
```

СтрокаРасчета – параметр процедуры, передаваемый ей в точке вызова. В модуле объекта для документов «**Приходная накладная**», «**Расходная накладная**» для имеющихся обработчиков событий изменим программный код на следующий:

```
СтрокаРасчета = Элементы.Товары.ТекущиеДанные;
РаботаСДокументами.ПересчитатьСумму(СтрокаРасчета);
```

В первой строке кода получаем текущие данные, для которых нужен обработчик. Во второй строке вызываем процедуру из общего неглобального модуля, и передаем эти данные через параметр процедуры.

Подсказка: При написании программного кода обратите внимание на то, что система автоматически выдает список конструкций, которые могут быть. Также при описании прикладных объектов, при нажатии на точку система показывает список возможных реквизитов объекта, из которых можно выбрать. Вы всегда можете ввести первые символы конструкции (или прикладного объекта, реквизита и т.д.), которую хотите использовать, после чего нажать комбинацию клавиш «Ctrl+Пробел».

В результате система выполнить синтаксический поиск по первым введенным символам и выведет список возможных методов, событий, объектов, реквизитов и т.д., которые можно использовать; вам будет достаточно в появившемся списке выбрать подходящий вариант. Это позволяет, во-первых, разрабатывать код более быстро и эффективно, а, во-вторых, снижает вероятность возникновения ошибок из-за опечаток при наборе.

Однако это будет действовать не всегда: например, вы присвоили переменной имя объекта. Не факт, что система всегда сможет определить реквизиты данной переменной, как и объекта; у исходного объекта она обязана это делать, но если было переприсвоение, то это не совсем обязательно. В данном случае вам необходимо выполнять контроль корректности ввода данных самостоятельно.

6.3 Журнал документов. Команды формы

Иногда бывают ситуации, когда пользователю необходимо ряд документов различного вида представить в едином списке в хронологическом порядке. В нашем случае – это перечень всех документов «**Приходная накладная**» и «**Расходная накладная**». Для таких целей используется объект «Журнал документов». Создадим новый журнал «**Приходно-расходные операции**», включим в подсистему «**Общий отдел**».

На закладке «Данные» указываем перечень всех документов, которые необходимо отобразить в журнале: укажем оба наших документа (рис. 6.10). По умолчанию, если больше ничего не настраивать, то при загрузке журнала система динамически сформирует список, который будет в хронологическом порядке содержать дату, номер и тип документа. Если необходимо вывести какую-то дополнительную информацию, то для этого следует воспользоваться добавлением граф на вкладке «Данные» (рис. 6.10).

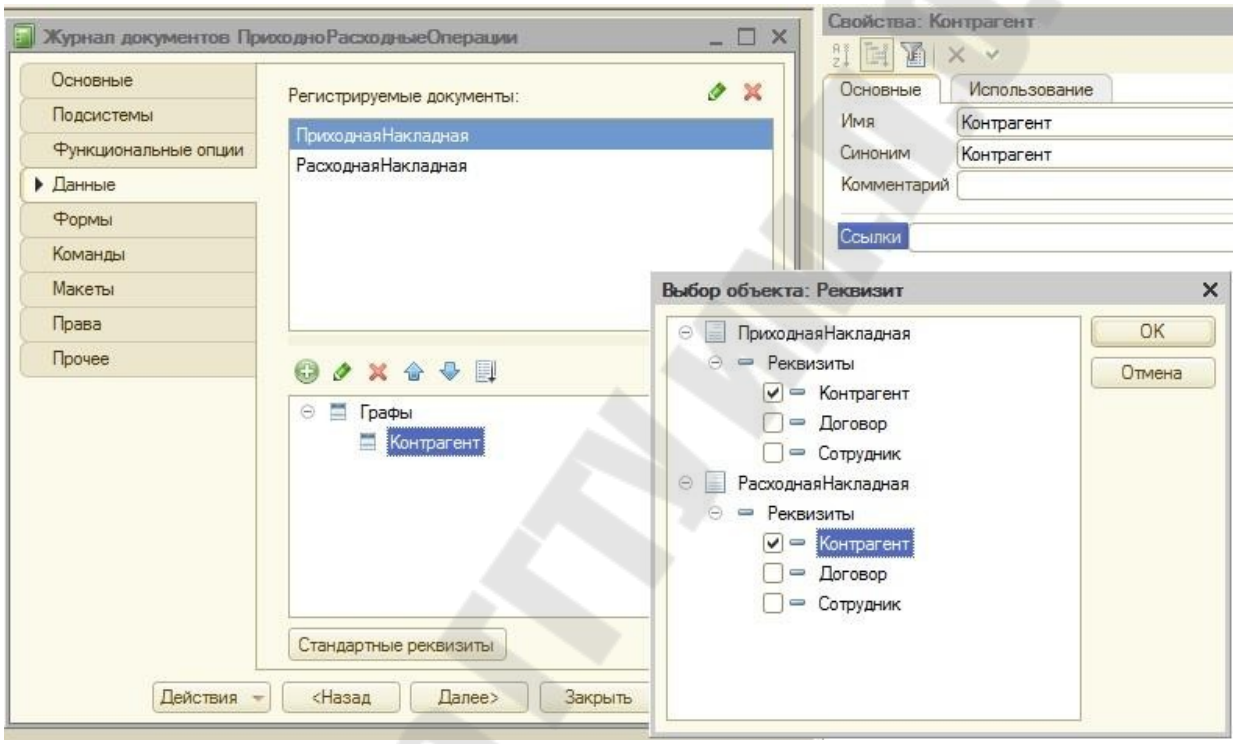


Рисунок 6.10 – Журнал документов

При добавлении очередной графы указывается ее имя и синоним, а в свойстве «Ссылки» – реквизиты, которые будут выбираться из документов каждого вида, входящих в журнал. Допустимо указание только реквизитов шапки, реквизиты табличных частей недоступны для выбора. Дополнительно следует обратить внимание, что тип графы журнала – составной. Это означает, что из различных документов в данной графе можно указать реквизиты различных типов, если это необходимо. В нашем случае создадим графы «**Контрагент**» и «**Сотрудник**», и укажем для каждого из документов соответствующие реквизиты. Запустим систему и посмотрим результат.

Далее предположим, что в журнале необходимо отображать также сумму документа. Рассмотрим решение на примере документа «**Приходная накладная**». Для этого создадим еще один реквизит шапки в документе: «**Сумма документа**», тип Число, длина 15, точность 2.

По той причине, что форма документа уже была ранее создана, данный реквизит автоматически на форме не появится (проверьте это, запустив систему в режиме исполнения). Для добавления вновь созданного реквизита, необходимо отрыть форму документа, в разделе «Данные формы» найти основной реквизит «**Объект**», в нем «**Сумму документа**», и перетащить его к «Элементам формы» (расположим ниже табличной части «Товары», но не в ней).

Далее указанный реквизит необходимо заполнить нужным значением. Создадим специальную кнопку – команду формы, которая будет рассчитывать сумму документа и заносить ее в указанный реквизит. В области «Данные формы» откроем вкладку «Команды», далее «Команды формы». Добавим новую команду «**Рассчитать сумму**» и перетащим ее к элементам формы, разместив в одну строку с реквизитом «**Сумма документа**». По умолчанию, система все элементы формы автоматически выстраивает друг под другом.

Для того чтобы изменить такое расположение, элементы необходимо группировать. Для группировки в «Элементы формы» можно добавить  следующие группы:

- Обычная группа – для простой группировки элементов;
- Страницы – для создания группы, которая будет содержать несколько закладок, в каждой из которых может быть несколько элементов;
- Страница – отдельная закладка, создается внутри группы типа «Страницы»;
- Командная панель формы – для создания команд на форме на специальной панели.

Создадим обычную группу и переместим в нее элемент «**Сумма документа**» и кнопку «**Рассчитать сумму**». В панели свойств созданной группы можно настроить положение элементов, отображение заголовков и т.д. Для настройки положения элементов используется свойство «**Группировка**». Аналогичным образом расположим элементы формы «**Дата**» и «**Номер**» в одну строку, дополнительно изменив их заголовки. В результате всех преобразований форма должна выглядеть, как показано на рис. 6.11.

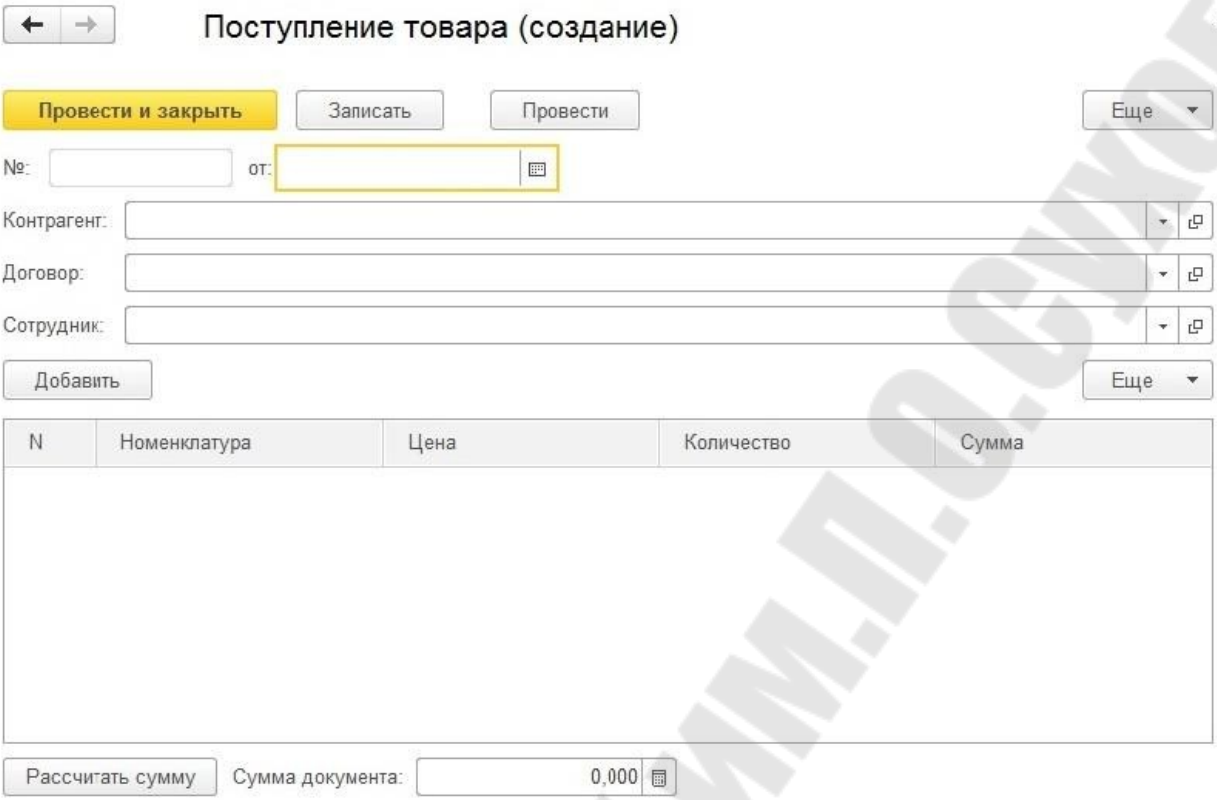


Рисунок 6.11 – Измененная форма документа «**Приходная накладная**»

Затем для кнопки необходимо описать механизм заполнения суммы. Для этого выбирается созданная команда формы, и в панели свойств открывается «Действие» для данной команды (на клиенте). В открывшуюся процедуру добавляем следующий код:

```
Объект.СуммаДокумента = Объект.Товары.Итог("Сумма");
```

В данном программном коде применяется функция ***Итог()***, которая производит суммирование числовых значений в указанном столбце таблицы (столбец «**Сумма**»). Результат записывается в нужный реквизит. Осталось проверить механизм работы на практике.

7 РЕГИСТРЫ НАКОПЛЕНИЯ. ПРОВЕДЕНИЕ ДОКУМЕНТОВ

7.1. Регистры накопления

Документ отражает факт хозяйственной деятельности на предприятии. Однако на текущий момент просто создан ряд документов с информацией, не более. Нет информации о том, сколько всего товара поступило, сколько всего товара продано. Также отсутствует анализ того, могли ли мы продавать товар (вдруг его не было на складе); нет анализа того на какую сумму мы продали товар, какую выручку получили, и сколько прибыли (выручка минус себестоимость) и т.д. На данный момент у нас нет никаких учетных механизмов, которые являлись бы показателями финансовой и хозяйственной деятельности предприятия, отражали бы изменение материального, финансового и т.д. состояния организации.

В качестве механизмов, отражающих учет на предприятии, используется прикладной объект «Регистры накопления». Они служат для накопления информации о различных аспектах деятельности организации: сколько товара пришло, ушло, сколько есть в наличии (в количественном или стоимостном выражении); на какую сумму продали товара, какова выручка и т.д. Эти сводные, сгруппированные данные в дальнейшем можно использовать для формирования различных отчетов, бухгалтерских, финансовых и т.д. – то есть то, ради чего и разрабатывается автоматизированная система.

Записи в регистрах накопления могут формировать только документы, отражающие факт хозяйственной деятельности. Здесь вводится понятие «Проведение» документа. Пока документ не проведен – он является просто «черновиком», который показывает, что определенный вид деятельности на предприятии ведется. Но, как только документ становится проведенным – это является признаком совершения (вернее, завершения) операции, которая приводит к формированию записей в регистре накопления, связанным с данным документом. Эти записи и отражают изменение состояния (материального, финансового и т.д.) на предприятии. Таким образом, только проведение документа фактически показывает, что учетные данные на предприятии изменились. В терминах «1С:Предприятие» проведение документа означает формирование «движений» (записей) в соответствующем регистре накопления. Документ, формирующий

движения, называется «Регистратором» для данного регистра. Так, для ведения учета, необходимо описать соответствующие регистры накопления, и сформировать механизмы их движений.

Для учета материальных ценностей (приход, расход) на предприятии создадим регистр накопления «**Остатки товаров**». Существует два типа регистров накопления: регистр остатков и регистр оборотов.

В регистре остатков каждая запись регистра имеет соответствующий тип: приход (товары в плюс) или расход (товары в минус).

Регистр накопления имеет особенность в хранении данных, заключающуюся в том, что он дополнительно хранит итоговые записи (суммарный приход, расход, оборот, остатки товаров) на конец каждого месяца. Это позволяет системе производить более быстрое извлечение требуемых данных на определенную дату.

Регистр накопления типа «Обороты» не разделяет данные на приход и расход: все записи, которые формируются в регистре, накапливаются и идут «в плюс». Обороты нужны при накоплении информации, например, о выручке: никогда выручка не пойдет в минус – это те деньги, которые мы получили от продажи товара (вариант возврата товара сейчас не рассматривается, это более сложный механизм). Соответственно, в оборотном регистре не будет остатков – в нем будет информация лишь о накопленных данных за определенный период.

Для ведения материального учета необходимо выбрать тип регистра «**Остатки**». Включим регистр в подсистемы «**Отдел закупок**», «**Отдел продаж**» и «**Бухгалтерия**». Перейдем к закладке Данные (рис. 7.1).

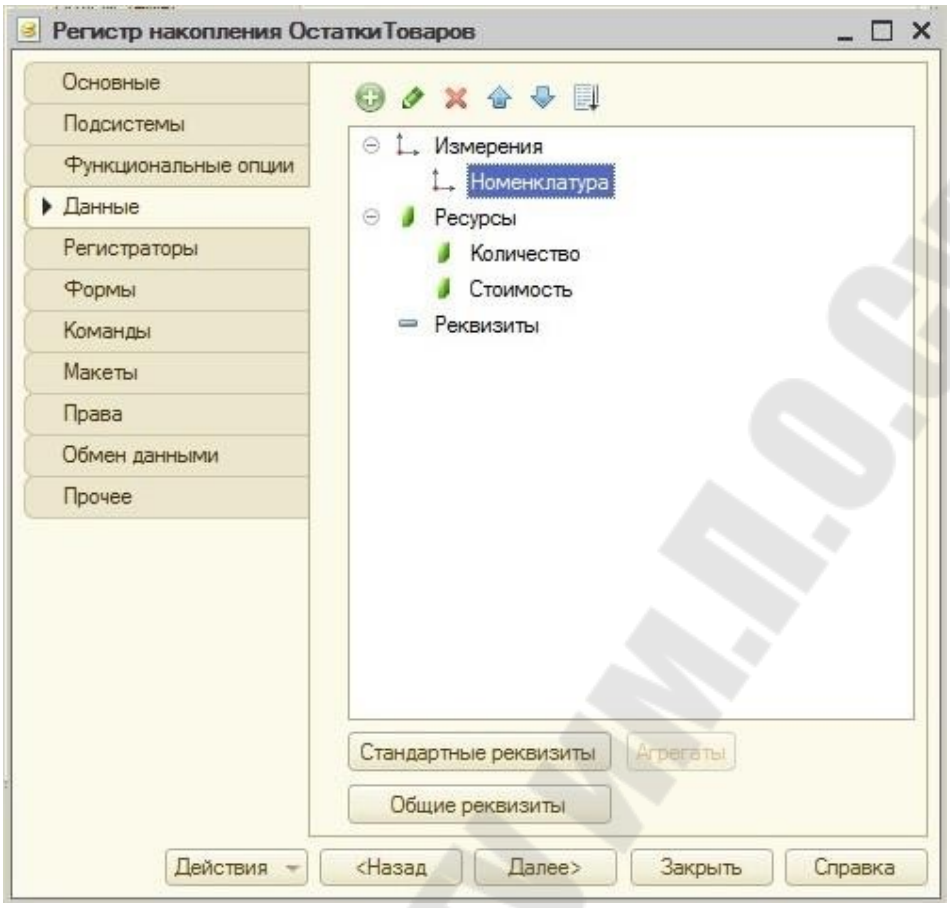


Рисунок 7.1 – Измерения и ресурсы регистра накопления

Регистры не являются типообразующими объектами, т.е. никогда нельзя указать ссылку на какую-то конкретную запись регистра. Регистр служит для хранения и быстрого извлечения данных, но не для ссылки на них. Регистр – это таблица с набором полей. Все поля делятся на:

- стандартные реквизиты;
- измерения;
- ресурсы;
- реквизиты.

К стандартным реквизитам относятся поля, обязательным образом присутствующие у всех записей регистра:

- **Период**: дата записи; периодичность записи зависит от настроек регистра, для регистра накопления периодом является дата, устанавливаемая в процедуре формирования движений;
- **Регистратор**: документ, формирующий движения регистра;
- **Номер строки**: номер строки в документе-регистраторе (в

табличной части), уникален в пределах регистратора;

– **Активность**: отвечает за попадание записи в виртуальные таблицы (о виртуальных таблицах позже);

– **Вид движения**: Приход/Расход для регистра типа «Остатки».

Измерения – это разрезы, в рамках которых хранятся данные. Набор измерений характеризует уникальность при идентификации конкретных учетных данных. В нашем случае измерение будет одно – «**Номенклатура**»: мы хотим накапливать информацию о поступлении и продаже конкретных товаров. Если бы мы, допустим, хранили товары на разных складах, то нас бы интересовало, сколько какого товара хранится на конкретном складе. Тогда вторым измерением был бы «**Склад**». Одно измерение для регистра накопления является стандартным и относится к стандартным реквизитам – «**Период**».

Ресурсы – это то, что мы храним (количество товара, его себестоимость и т.д.). Для регистра накопления ресурс может быть только числового типа.

Реквизиты – это некоторые дополнительные данные, вспомогательная информация, которая никак не влияет на формирование оборотов и остатков. Остатки и обороты, хранящиеся в регистре накопления, рассчитываются только в зависимости от набора измерений.

Добавим измерение «**Номенклатура**», тип СправочникСсылка.Номенклатура, ресурсы «**Количество**» (длина 15, точность 2) и «**Стоимость**» (длина 15, точность 2) (рис. 7.1). Для измерения «**Номенклатура**» можно отметить галочку «Запрет незаполненных» – тем самым нельзя будет сформировать записи с пустым значением данного измерения.

На вкладке «Регистраторы» укажем оба документа («**Приходная накладная**» и «**Расходная накладная**») в качестве документов, которые могут формировать записи регистра. «**Приходная накладная**» будет формировать записи «приход», «**Расходная накладная**» – соответственно «расход».

7.2. Процедура проведения документов

Предположим, необходимо в соответствующих документах описать правила формирования движений (записей регистра). Для начала опишем процедуру проведения документа «**Приходная накладная**». Для этого откроем окно редактирования документа и перейдем на закладку «***Движения***» (рис. 7.2).

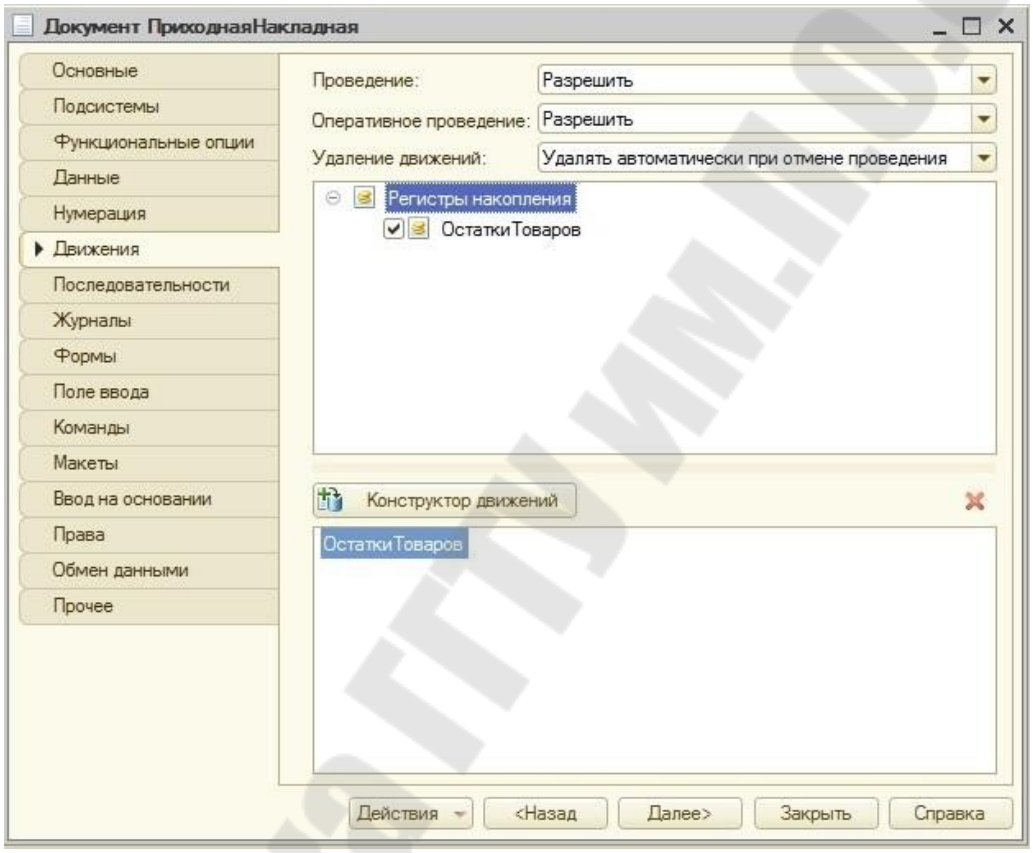


Рисунок 7.2 –. Настройка движений для документа

В самом верху располагаются общие настройки проведения. Проведение можно «Разрешить» или «Запретить». Для документов, у которых разрешено проведение, имеется специальная кнопка «Провести» на форме документа. При этом в форме списка проведенные документы помечаются зеленой галочкой на значке документа. Те документы, для которых проведение запрещено в принципе, на форме документа нет кнопки «Провести» и вообще отсутствуют какие-либо команды, связанные с проведением документа. Все документы такого вида системой автоматически

помечаются как проведенные (чтобы пользователь даже не пытался их проводить).

Пункт «***Оперативное проведение***» предназначен для использования проведения документов в режиме реального времени, т.е. текущей датой. Как правило, в большинстве учетных систем используется оперативное проведение, отличительная черта которого заключается в невозможности проведения документов будущим временем. Проведение «задним числом» допустимо, которое является уже неоперативным проведением. Если запретить оперативное проведение документов, то это приведет лишь к тому, что документы можно будет проводить как прошлой или текущей датой, так и будущей.

Раздел «***Удаление движений***» предназначен для описания правил перепроведения документов. По умолчанию, движения (т.е. записи регистра) удаляются при отмене проведения. Если же документ перепроводится, то сохраняются старые движения и дополнительно записываются новые (измененные). Это бывает необходимо в том случае, когда нужна история формирования движений. Настройка «***Удалять автоматически***» приводит к тому, что все старые движения будут автоматически удаляться и замещаться новыми как при отмене проведения документа, так и при его перепроведении. При выборе «***Не удалять автоматически***» все движения документа будут сохраняться. Выберем автоматическое удаление движений.

Ниже в окне выбираются регистры (сведений, накопления и т.д.) в которых данный документ будет формировать движения. Список выбранных регистров отображается в нижнем окне. Для настройки правил проведения документа, т.е. формирования движений, в самом простейшем случае, воспользуемся конструктором движений, нажав соответствующую кнопку (рис. 7.3).

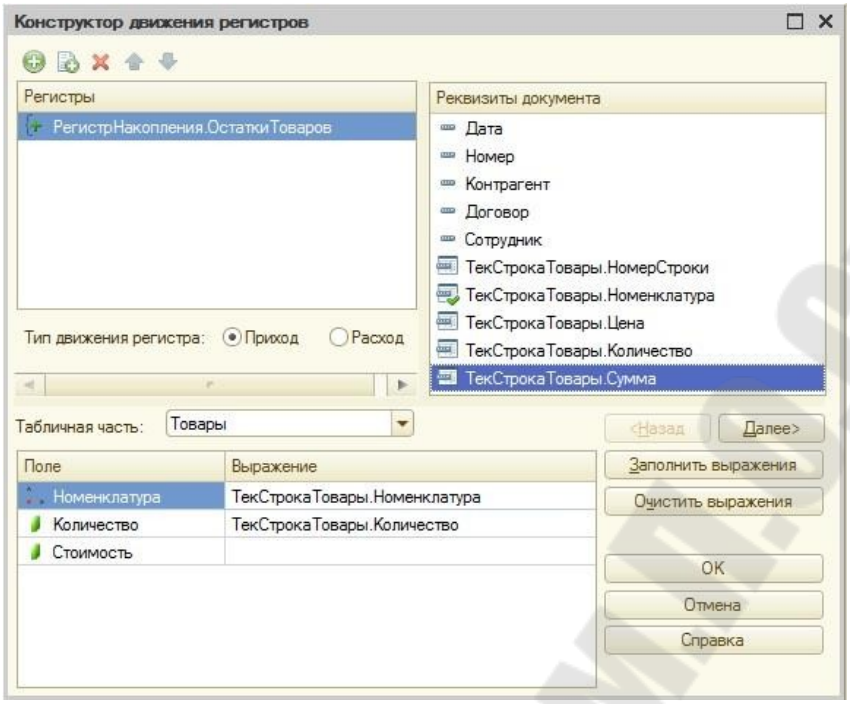


Рисунок 7.3 – Конструктор движений документа

В окне «Регистры» указывается перечень регистров, по которым формируются движения. Для добавления еще одного регистра, используется кнопка «Добавить». Следует отметить, что по одному и тому же регистру документ может формировать несколько различных движений, поэтому ничто не запрещает один и тот же регистр добавить несколько раз, и для каждого случая описать различные правила.

В правом окне «Реквизиты документа» указывается перечень реквизитов, значения которых будут подставляться в измерения и ресурсы соответствующего регистра. Если требуемые реквизиты расположены в табличной части документа, то необходимо выбрать табличную часть в соответствующем пункте.

Конструктор движений автоматически определяет соответствие типа данных измерения/ресурса регистра и реквизита документа. И те реквизиты, которые по типу совпадают с текущим выбранным для заполнения измерением/ресурсом, помечаются специальным зеленым маркером (рис.7.3). После нажатия на кнопку «***Заполнить выражение***» система для данных реквизитов дополнительно проверяет строковое совпадение их имен с именами измерений/ресурсов регистра, и заполняет поле «***Выражение***» для тех данных, для которых было найдено точное совпадение по типу и

по имени. Данное выражение при необходимости можно вручную отредактировать. Если оказалось, что для поля регистра автоматически не нашлось соответствия с реквизитами документа, то необходимо в нужном поле вручную указать имя реквизита документа (или написать произвольное выражение). Так, ресурс «**Стоимость**» можно заполнить значением реквизита «**ТекСтрокаТовары.Сумма**» (рис. 7.3).

В зависимости от типа регистра, в конструкторе движений имеется ряд настроек. В частности, для регистра накоплений типа «Остатки» - это вид движения. В нашем случае документ «**Приходная накладная**» будет формировать движения типа «**Приход**» по соответствующему регистру накопления.

После сохранения движений открывается модуль объекта с автоматически сформированной процедурой проведения документа. Если в полученный код внести изменения, то при повторном запуске конструктора движений, все изменения будут утеряны. Поэтому, для ручного исправления процедуры проведения необходимо самостоятельно открывать модуль объекта для редактирования.

Аналогичным образом сформируйте движения документа «**Расходная накладная**» по регистру накопления «**Остатки товаров**».

Далее, обратите внимание, что в документе «**Расходная накладная**» не только продаются товары, но и оказываются услуги. Но услуги не могут поступать или расходоваться. Поэтому сформированную процедуру необходимо изменить, добавив условие, проверяющее, является ли выбранная номенклатурная позиция товаром или услугой. Измененная процедура проведения будет выглядеть следующим образом (жирным выделен добавленный фрагмент кода):

```
Движения.ОстаткиТоваров.Записывать = Истина;
Для Каждого ТекСтрокаТовары Из Товары Цикл
    Если ТекСтрокаТовары.Номенклатура.ВидНоменклатуры =
                    Перечисления.ВидыНоменклатуры.Товар Тогда
    Движение = Движения.ОстаткиТоваров.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
    Движение.Период = Дата;
    Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
    Движение.Количество = ТекСтрокаТовары.Количество;
    Движение.Стоимость = ТекСтрокаТовары.Сумма;
   КонецЕсли;
КонецЦикла;
```

Запустим систему в режиме отладки, и проверим работу рассмотренного механизма проведения. Для этого создадим несколько документов поступления товаров (различные товары, различные поставщики) и несколько документов продажи товаров (различные товары, услуги, различные покупатели). Для просмотра записей (движений) в соответствующем регистре накоплений воспользуемся меню «Все функции»  и выберем регистр накопления «**Остатки товаров**».

Создадим регистр накопления «**Продажи**» (добавим в подсистему «**Отдел продаж**»), который будет накапливать информацию о продажах товаров и оказании услуг – будет хранить выручку компании. Тип регистра в данном случае будет «Обороты» т.к. выручка идет только в плюс (не рассматриваем ситуации возврата товара). Измерения: «**Номенклатура**» и «**Контрагент**», ресурсы – «**Количество**» и «**Сумма**». В качестве регистратора – документ «**Расходная накладная**».

Т.к. для указанного документа уже была создана и вручную отредактирована процедура проведения, то не будем ее портить повторным запуском конструктора движений. Откроем модуль объекта для данного документа. Добавим в конец процедуры проведения следующий код:

```
Движения.Продажи.Записывать = Истина;
Для Каждого ТекСтрокаТовары Из Товары Цикл
    Движение = Движения.Продажи.Добавить();
    Движение.Период = Дата;
    Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
    Движение.Контрагент = Контрагент;
    Движение.Количество = Движение.Количество = ТекСтрокаТовары.Количество;
    Движение.Сумма = ТекСтрокаТовары.Сумма;
КонецЦикла;
```

Запустим систему в режиме отладки, перепроведем документы «**Расходная накладная**» и через меню «Все функции» проверим формирование записей в регистре накопления «**Продажи**».

8 РЕГИСТРЫ СВЕДЕНИЙ. ОБЪЕКТНАЯ МОДЕЛЬ ДОСТУПА К ДАННЫМ

При продаже товаров цена каждый раз вводится вручную. Но на одни и те же товары цена будет одинаковая, как минимум, в течение определенного периода. Соответственно, необходим объект, который будет хранить цену, и откуда ее можно будет легко извлечь. Реквизиты справочника для этой цели не подойдут, т.к. информация, хранимая в них, постоянна, а если и меняется, то не важна история этих изменений. В примере с ценами это не так – пусть редко, но цена может меняться, при этом важно знать – когда и сколько какой товар стоил (это будет влиять на оценку прибыли в определенные периоды существования системы). Для хранения такого рода информации используется специальный объект – ***регистр сведений***.

Регистр сведений может быть периодическим и непериодическим. В непериодическом регистре сведений хранится информация, которая неизменна с течением времени, например, характеристики товаров. Периодический регистр сведений хранит изменяемую со временем информацию, например, цены товаров, курсы валют и т.д. При этом можно задать различные периоды.

Особенность регистра сведений заключается в том, что в нем поддерживается автоматический контроль уникальности записей, т.е. двух одинаковых записей быть не может. Например, в один день мы не сможем установить разные цены на один и тот же товар (при периодичности в течение дня). Поэтому использование регистра сведений особенно важно при таком контроле. Каждая запись регистра сведений состоит из измерений и ресурсов.

Измерения – это то, что мы храним в регистре. Набор измерений должен быть уникальным.

Ресурс – значение, которое мы храним в регистре.

В примере с ценами, измерениями будут Дата и Номенклатура, а ресурсом – Цена. Измерение и ресурс могут быть любых типов, кроме типа данных «Хранилище значения».

Добавим регистр сведений «**Цены номенклатуры**» (рис. 7.4).

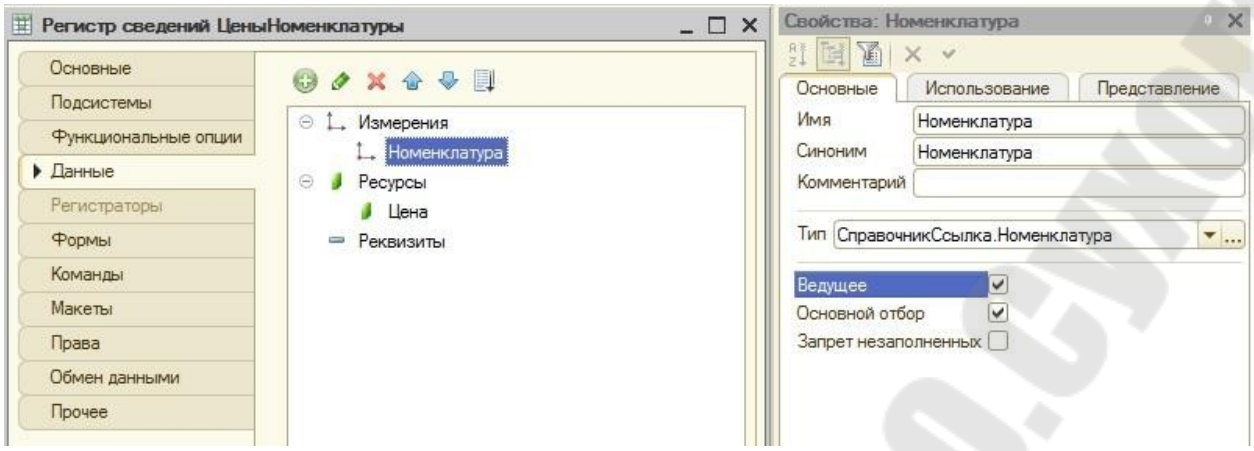


Рисунок 7.4 – Создание периодического регистра сведений «Цены номенклатуры»

На вкладке «Основные» установим периодичность регистра «**В пределах дня**», режим записи – «**Независимый**». Записи в регистре сведений можно формировать вручную, а можно при помощи документа-регистратора. Включим регистр в подсистему «**Общий отдел**». Измерение регистра – «**Номенклатура**» (тип СправочникСсылка.Номенклатура), ресурс – «**Цена**» (тип Число, длина 15, точность 2) (рис. 41). Обратите внимание, что «**Дата**» не была задана в качестве измерения неслучайно: как только было указано, что регистр является периодическим, система автоматически добавляет измерение, отвечающее за дату записи (стандартное измерение называется «**Период**»).

Также следует отметить галочку «**Ведущее**» для измерения «**Номенклатура**» (рис. 7.4). Установка данной опции привязывает записи регистра сведений элементам справочника «**Номенклатура**». Тем самым элементы справочника «**Номенклатура**» «владеют» записями регистра сведений, в которых они фигурируют в качестве измерений. Это приводит к следующему:

1. при удалении элемента из справочника «**Номенклатура**», автоматически удаляются все записи в регистре сведений, где данный элемент был указан в качестве измерения;

2. при открытии формы элемента справочника «**Номенклатура**» в панели навигации формы будет расположена ссылка для перехода к записям регистра сведений с установленным отбором по данной номенклатурной позиции (аналогично подчиненным справочникам).

Запустите систему в режиме исполнения и внесите ряд записей в регистр сведений. Попробуйте внести несколько одинаковых записей, т.е. для одной и той же номенклатуры на одну и ту же дату установите разные цены и посмотрите на работу системы.

Таким образом, основное назначение регистра сведений – хранить уникальную информацию, которую можно достаточно быстро извлекать.

Следует отметить, что все регистры не имеют объектной природы, т.е. на их записи нельзя сослаться, их можно лишь считать. Извлечение (считывание) записей из регистров происходит намного быстрее, нежели чтение реквизитов справочников, документов и т.д. Это связано с тем, что при обращении к записям регистра, система обращается не к физическим данным, а к специальным виртуальным таблицам. Для каждого типа регистра перечень виртуальных таблиц свой. Данные таблицы создаются «на лету» и содержат итоговые, максимально сгруппированные записи. Так, например, для регистра сведений создается виртуальная таблица «***СрезПоследних***», содержащая последние записи с уникальными комбинациями измерений на указанную дату. А для регистра накоплений, допустим, «**Остатки товаров**» типа «Остатки» создается виртуальная таблица «**ОстаткиТоваровОстатки**», содержащая сгруппированные итоговые записи о том, сколько и какого товара есть на указанную дату, без разворачивания конкретных записей поступления, расходования. При формировании виртуальных таблиц им передается ряд параметров, которые определяют перечень выдаваемых сгруппированных записей. Для регистров сведений и накоплений одним из таких параметров является дата. Параметры виртуальных таблиц ограничивают количество записей, считываемых из реальных таблиц и используемых при их группировке, формировании итогов; тем самым достигается наилучшее быстродействие, по сравнению со считыванием данных из таблиц справочников, документов и т.д.

Реализуем возможность считывания из регистра сведений информации о цене номенклатуры при ее продаже. Необходимо, чтобы в документе «**Расходная накладная**» при выборе определенной номенклатурной позиции происходило обращение к регистру сведений, и в документ подставлялась цена.

Откроем форму документа, перейдем к столбцу «**Номенклатура**» табличной части «**Товары**», в панели свойств определим обработчик события «**ПриИзменении**»: обработчик на

клиенте и функцию на сервере. Дело в том, что цены номенклатуры хранятся в регистре сведений (вернее, в соответствующей таблице базы данных), т.е. на сервере. С клиента, при выборе номенклатурной позиции, к ним просто так не обратиться, необходимо сделать специальный серверный вызов, который считает требуемые данные, и передаст их клиенту. Соответствующий код будет выглядеть следующим образом:

```
&НаКлиенте
Процедура ТоварыНоменклатураПриИзменении(Элемент)
 СтрокаРасчета = Элементы.Товары.ТекущиеДанные;
 СтрокаРасчета.Цена =
 ТоварыНоменклатураПриИзмененииНаСервере(СтрокаРасчета.Номенклатура,Объект.Дата);
 РаботаСДокументами.ПересчитатьСумму(СтрокаРасчета);
КонецПроцедуры
&НаСервереБезКонтекста
Функция ТоварыНоменклатураПриИзмененииНаСервере(Номенклатура, ДатаДок)
 СтруктураОтбора = Новый Структура;
 СтруктураОтбора.Вставить("Номенклатура", Номенклатура);
 ЦенаДок =
 РегистрыСведений.ЦеныНоменклатуры.ПолучитьПоследнее(ДатаДок,СтруктураОтбора);
Возврат ЦенаДок.Цена;
КонецФункции
```

Первая процедура выполняется на клиенте и производит:

– определение текущих рассчитываемых данных (получает текущую строку табличной части);

– передачу на сервер данных, необходимых для получения цены (т.е. ссылку на выбранную номенклатурную позицию и дату документа);

– получение цены с сервера и запись в соответствующий элемент формы;

– пересчет суммы в текущей строке табличной части с учетом полученной цены.

Вторая функция – внеконтекстная, все необходимые данные («Номенклатура» и «Дата») она получает через параметры.

Для получения требуемых данных производится:

1. формирование ***Отбора***, который представляет собой структуру (см. Синтакс- помощник). Ключами структуры являются измерения регистра сведений (здесь ключ – «**Номенклатура**») за исключением измерения «**Период**»;

2. обращение к виртуальной таблице ***СрезПоследних*** при

помощи метода ***ПолучитьПоследнее()***. В метод в качестве первого параметра передается **Период** (здесь – **ДатаДок**, переданная дата документа), в качестве второго – сформированный ***Отбор***;

3. возврат в результате работы метода структуры (здесь – **ЦенаДок**), ключами которой являются измерения и ресурсы регистра сведений, к которым можно обращаться по их имени;

4. передача ресурса – полученной цены **ЦенаДок.Цена** – в клиентскую процедуру в точку вызова.

Необходимо запустить систему в режиме исполнения и проверьте изученный механизм на практике.

Такой способ считывания данных из базы, когда «через точку» происходит обращение к полям, свойствам и методам объектов, носит название «***объектная модель доступа к данным***». Данная модель является достаточно простой и реализует в себе механизмы работы с системой в рамках объектно-ориентированного подхода. Использование объектной модели позволяет, как считывать данные, так и изменять сами объекты, и информацию в них содержащуюся. Однако она не является универсальной т.к. с ее помощью достаточно трудно, а иногда невозможно, формировать любые, включая достаточно сложные, обращения к таблицам базы данных, накладывая множество условий. Для этих целей используется *табличная модель (запросы) доступа к данным*, о которой речь пойдет ниже.

Регистр сведений может быть не только с «Независимым» режимом записи, но и с режимом записи «Подчинение регистратору». В этом случае записи формируется не вручную, а при проведении документа. При таком способе записи для периодического регистра появляется возможность установки периода «По позиции регистратора». Дело в том, что в 1 секунду технически может быть создано очень много документов (вплоть до нескольких тысяч). И определить какой документ был создан ранее, а какой позже, и, соответственно, какие записи в регистре были сделаны более ранним документом, а какие более поздним, оказывается затруднительно. Для этого имеется специальный реквизит «**Момент времени**», который формируется на основании текущей даты (дата+время) и ссылки на документ. Соответственно, момент времени позволяет точно расположить все документы на оси времени. Для регистра сведений с периодичностью «По позиции регистратора» производится позиционирование по «Моменту времени» проведения документа.

9 ЗАПРОСЫ. ТАБЛИЧНАЯ МОДЕЛЬ ДОСТУПА К ДАННЫМ

9.1. Табличная модель доступа к данным. Запросы

Наряду с рассмотренной выше объектной моделью доступа к данным, в системе «1С: Предприятие» реализована табличная модель. В основе ее – использование запросов к таблицам базы данных. В результате запроса возвращается набор данных, таблиц, которые можно анализировать и использовать. При помощи запроса программист никоим образом не может изменить данные. В этом отчасти преимущество запросов в системе «1С»: пользователь, программист не может случайно испортить информацию. В основе языка запросов лежит язык SQL с рядом ограничений (и, естественно, на русском языке).

Т.к. запрос производит чтение информации из таблиц базы данных, то, очевидно, что использовать запросы мы можем лишь на стороне сервера: либо в модуле формы при серверных вызовах, либо в модуле объекта (чаще всего). Текст запроса, его обработку и выполнение можно написать вручную, используя специальные конструкции, полный перечень которых приведен в Синтакс-помощнике. Однако в среде «1С: Предприятие» реализован более удобный инструмент для визуального создания запросов – конструктор запроса. Для вызова конструктора запроса в любой точке программного кода в модуле (формы, объекта) необходимо в контекстном меню выбрать либо «Конструктор запроса», либо «Конструктор запроса с обработкой результата». Конструктор запроса с обработкой результата отличается тем, что при его вызове автоматически формируется код специальных инструкций по выполнению запроса, передаче в него нужных параметров и обход результатов с целью их обработки и анализа; при вызове конструктора запроса без обработки результата производится формирование только текста запроса.

Решим следующую задачу. При поступлении товаров оказывается, что пользователь в табличную часть один и тот же товар может ввести несколько раз, при том даже по разным ценам. В результате при проведении документа в регистре накопления «**Остатки товаров**» будет сформировано несколько похожих записей. Необходимо эти записи сгруппировать, чтобы для одного

товара было одно движение. А для регистра сведений «**Цены поставщиков**» установить цену, максимальную для данного товара.

Отрываем процедуру обработки проведения в модуле объекта документа «**Приходная накладная**», и в начале процедуры запускаем «конструктор запроса с обработкой результата». На предложение создать новый запрос даем положительный ответ. В результате откроется следующее окно (рис. 9.1).

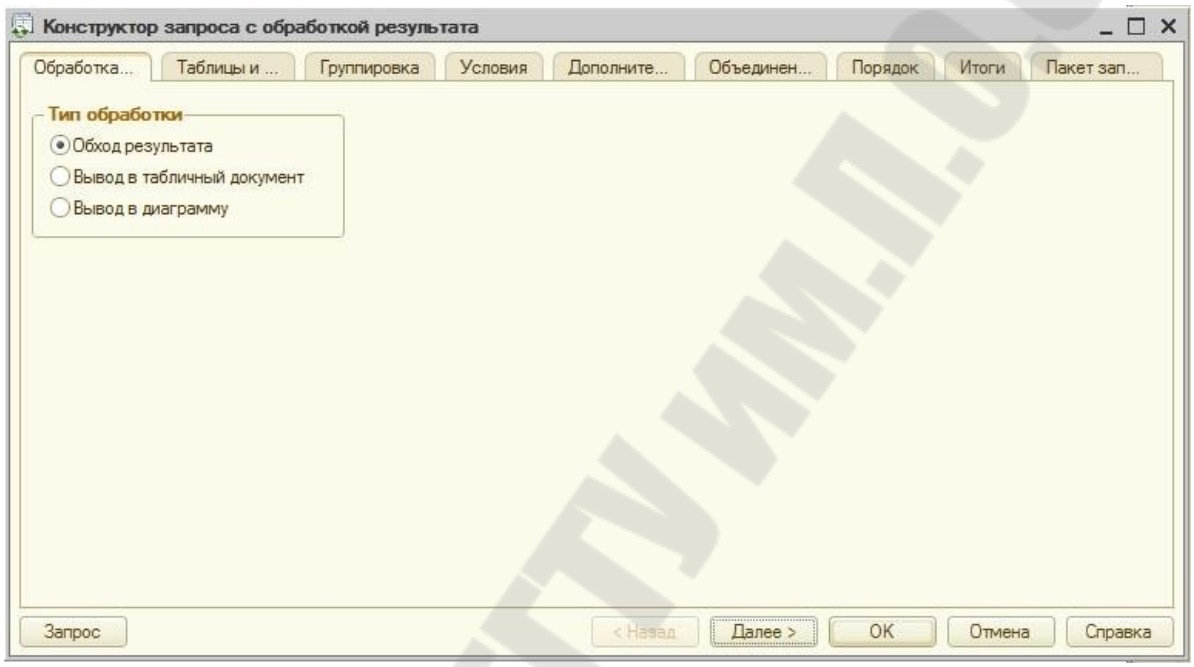


Рисунок 9.1 – Конструктор запроса с обработкой результата

Выбираем тип обработки «Обход результата» и переходим к следующей вкладке. На вкладке «Таблицы и поля» (рис. 9.2) слева представлена структура разрабатываемого приложения в виде таблиц базы данных и виртуальных таблиц. Это – источники данных. В нашей задаче источником данных будет таблица, представляющая табличную часть документа «**Приходная накладная**». Дело в том, что физически в базе данных реквизиты шапки (справочника, документа) хранятся в одной таблице, а под табличные части элементов создается дополнительная таблица, в которой указывается ссылка на «элемент- владелец» (справочника, документа), к которому данные записи относятся. Из выбранной таблицы необходимо выбрать поля: «**Номенклатура**», «**Цена**», «**Количество**», «**Сумма**» (рис. 9.2).

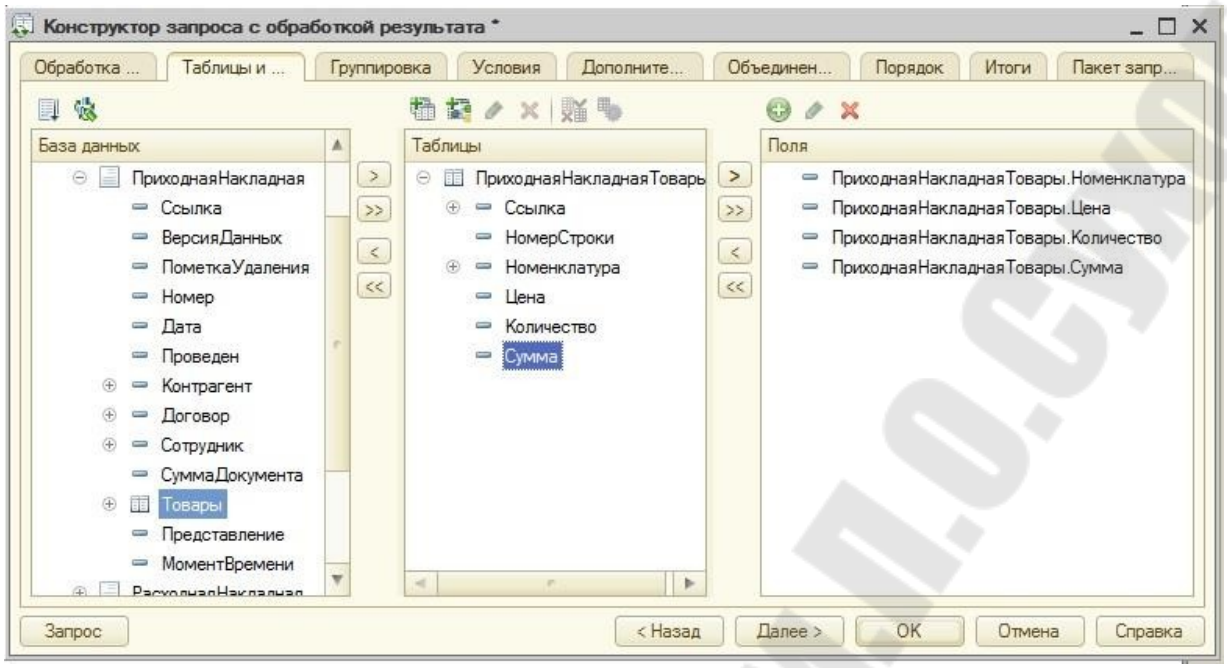


Рисунок 9.2 – Выбор источников данных и полей

На вкладке группировка (рис. 9.3) накладываются условия группировки: все поля разносятся по двум категориям: поля группировки и группируемые поля. Не должно возникать ситуации «висящих» полей, не принадлежащих ни той, ни другой группе.

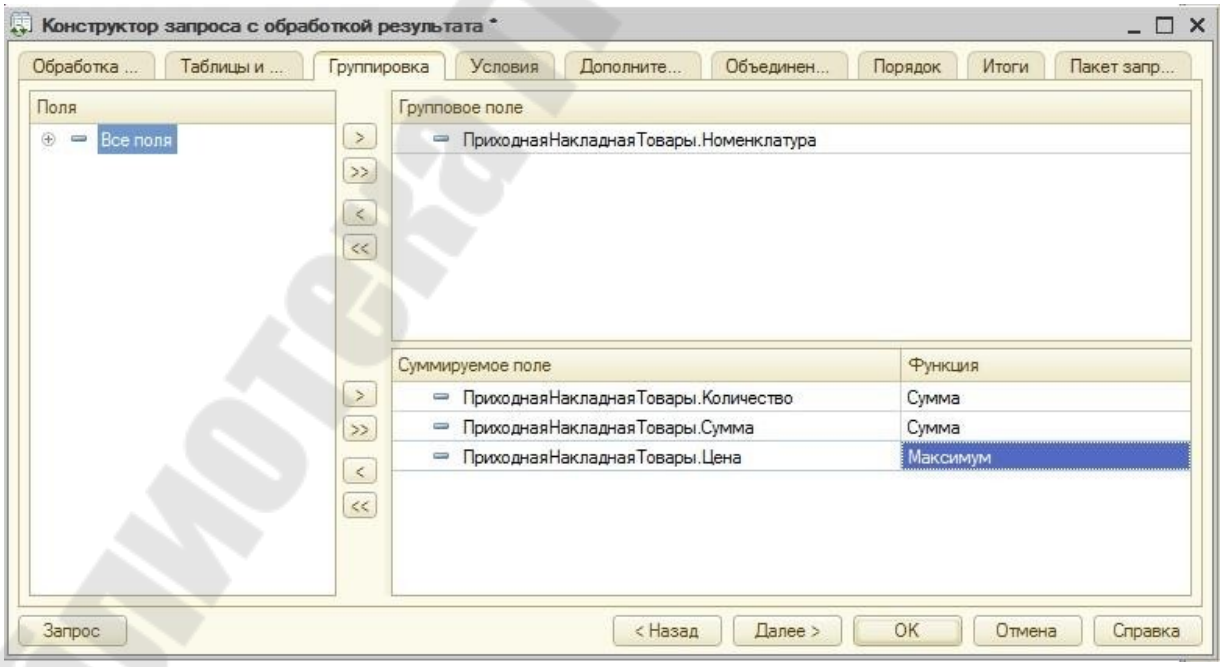


Рисунок 9.3 – Группировка полей в запросе

В нашей задаче полем группировки будет «**Номенклатура**» (одинаковые товары будут фигурировать в движениях один раз), все остальные поля – группируемые. К группируемым полям применяются агрегатные функции («Сумма», «Сумма различных», «Максимум» и т.д.). Для полей «**Количество**» и «**Сумма**» применяется агрегатная функция «Сумма» (для простого суммирования одинаковой номенклатуры, находящейся в различных строчках), а к полю «**Цена**» - функция «Максимум», для выбора наибольшей установленной цены для данной номенклатурной позиции.

На закладке «Условия» (рис. 9.4) можно наложить условия на поля запроса, тем самым ограничить число записей, возвращаемых в результате запроса. В нашей задаче мы хотим получить сгруппированные записи не по всем документам, а только по текущему. Для этого нужно наложить условие на реквизит «Ссылка», конкретную ссылку на документ будем передавать через параметр запроса (рис. 9.4).

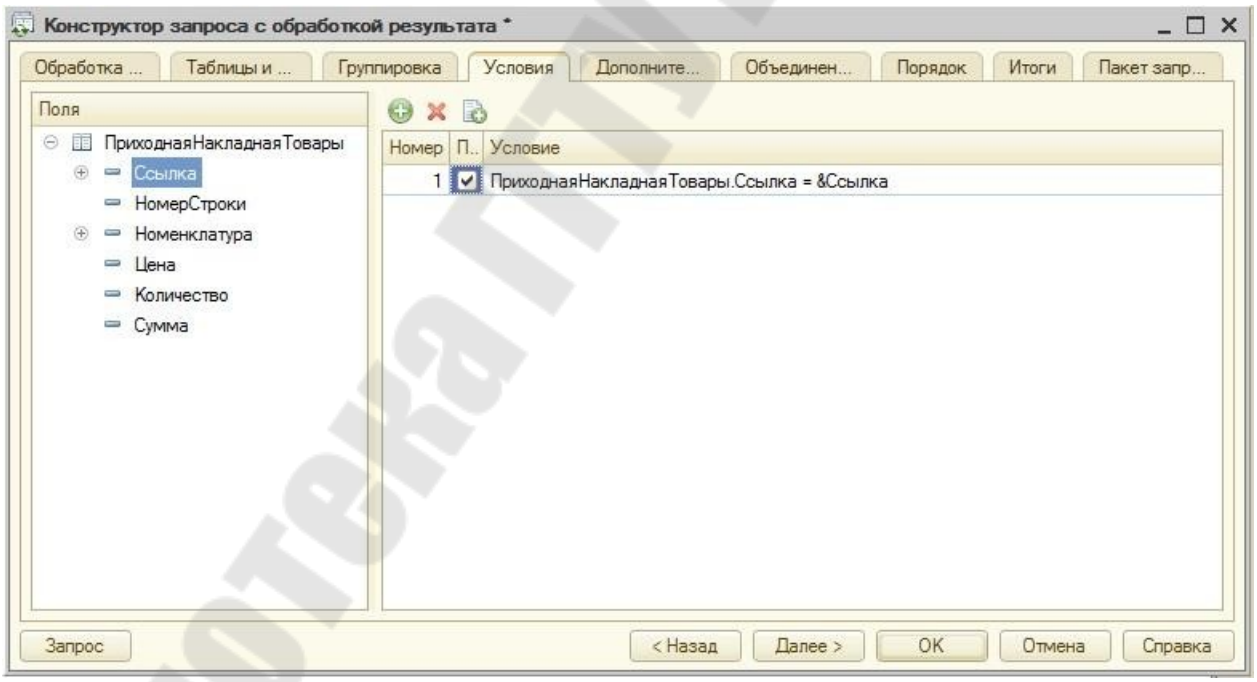


Рисунок 9.4 – Наложение условий в запросе

На вкладке «Объединения/Псевдонимы» можно задать псевдонимы для всех полей, возвращаемых в результате запроса. В нашем случае изменять ничего не нужно.

Для просмотра текста сформированного запроса можно нажать кнопку «Запрос». Для завершения формирования запроса нажимаем «ОК». В результате в модуле объекта сформировался следующий код:

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
| ПриходнаяНакладнаяТовары.Номенклатура,
| МАКСИМУМ(ПриходнаяНакладнаяТовары.Цена) КАК Цена,
| СУММА(ПриходнаяНакладнаяТовары.Количество) КАК Количество,
| СУММА(ПриходнаяНакладнаяТовары.Сумма) КАК Сумма
|ИЗ
| Документ.ПриходнаяНакладная.Товары КАК
ПриходнаяНакладнаяТовары
|ГДЕ
| ПриходнаяНакладнаяТовары.Ссылка = &Ссылка
|
|СГРУППИРОВАТЬ ПО
| ПриходнаяНакладнаяТовары.Номенклатура";
Запрос.УстановитьПараметр("Ссылка", Ссылка);
РезультатЗапроса = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
// Вставить обработку выборки ВыборкаДетальныеЗаписи
КонецЦикла;
```

В данном тексте вначале создается специальный объект типа «Запрос», после чего пишется сам текст запроса. Далее идет раздел задания параметров (УстановитьПараметр), в котором конструктор автоматически определяет параметры, передаваемые в запрос: слева указывается имя параметра в строковом представлении, справа – его значение. Если значение параметра, сформированное автоматически, оказалось неправильным, то его нужно исправить. Далее идет инструкция, выполняющая запрос, после чего следует передача результата запроса: в таблицу, диаграмму, или выборку. Выборка используется для обхода каждой записи в результате запроса. Используя метод ***Выбрать()*** система позиционируется на начале выборки. Для перехода к каждому следующему элементу выборки и считыванию значений конкретных полей используется метод ***Следующий()***. Соответственно, представленный цикл осуществляет переход к каждому следующему элементу выборки до конца. Внутри данного цикла мы можем обращаться к полям запроса как к полям выборки, например, **ВыборкаДетальныеЗаписи.Номенклатура**.

В нашей задаче параметр, передаваемый в запрос верный: мы передаем ссылку на текущий обрабатываемый документ. Цикл обхода элементов выборки будет выглядеть следующим образом (жирным выделены изменения в формировании движений, по сравнению с предыдущим вариантом):

```
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
Движение = Движения.ОстаткиТоваров.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
Движение.Период = Дата;
Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
Движение.Количество = ВыборкаДетальныеЗаписи.Количество;
Движение.Стоимость = ВыборкаДетальныеЗаписи.Сумма;
Движение = Движения.ЦеныПоставщиков.Добавить();
Движение.Период = Дата;
Движение.Контрагент = Контрагент;
Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
Движение.Цена = ВыборкаДетальныеЗаписи.Цена;
КонецЦикла;
```

После цикла, не забываем оставить возможность записи движений в регистры:

```
Движения.ОстаткиТоваров.Записывать = Истина;
Движения.ЦеныПоставщиков.Записывать = Истина;
```

Весь оставшийся предыдущий код процедуры обработки проведения, удаляем. Запускаем систему и проверяем данный механизм на практике. Обратите внимание, что запрос используется лишь для получения, группировки и обработки данных. При этом запрос работает намного быстрее, чем обращение через точку (объектная модель). Но для записи данных все равно используется объектная модель.

9.2. Контроль остатков на складе

При проведении документа «**Расходная накладная**» необходимо проверять, хватает ли товара, который мы хотим продать или нет. В случае, если товара хватает, документ проводится, иначе – выдается сообщение о том, какого товара и в каком количестве недостаточно и документ не проводится. Для решения поставленной задачи необходимо сформировать запрос, который будет возвращать

количество продаваемого товара и остатки по данным номенклатурным позициям.

Исправим имеющийся запрос в модуле объекта документа «**Расходная накладная**». Для этого спозиционируемся на уже имеющемся тексте первого запроса (по «**Товарам**») и вызовем «Конструктор запроса».

Группировку строк табличной части оставляем т.к. если фигурирует один и тот же товар много раз, необходимо знать, сколько ВСЕГО указанного товара необходимо продать, а затем проверить есть ли он на складе.

Воспользуемся механизмом пакетных запросов, для контроля остатков по данным сгруппированным записям. Для этого перейдем на вкладку «Дополнительно» (см. рис. 42) выберем пункт «Создание временной таблицы», укажем «**ТоварыТЧ**» в качестве имени временной таблицы. Таким образом, после выполнения первого запроса из пакета (группировки) будет создана еще одна таблица – источник для следующих запросов в пакете.

Переходим на вкладку «Пакет запросов» (см. рис. 9.1) и добавляем еще один запрос.

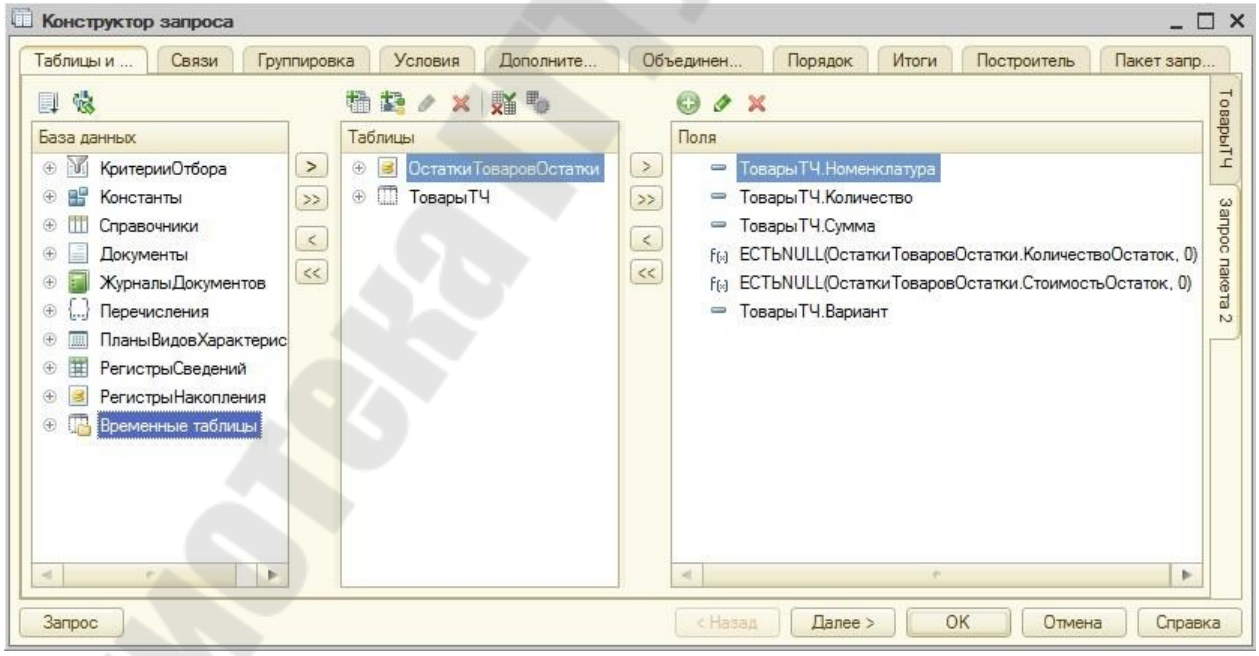


Рисунок 9.5 – Конструктор запроса в режиме пакетных запросов

Для перехода между двумя запросами из пакета справа имеются соответствующие вкладки (рис. 9.5). Во втором запросе к источникам

данных добавился еще один источник «Временные таблицы». Выбираем временную таблицу «**ТоварыТЧ**» в качестве одного источника. В качестве второго источника выбираем виртуальную таблицу «**ОстаткиТоваровОстатки**» регистра накопления «**Остатки товаров**». Данная виртуальная таблица содержит сгруппированные итоговые записи по остаткам товаров на дату, которая будет передаваться ей в качестве параметра.

В качестве полей, возвращаемых запросом из таблицы «**ТоварыТЧ**» выбираем все поля, а из виртуальной таблицы остатков – «**КоличествоОстаток**» и «**СтоимостьОстаток**». Для виртуальной таблицы необходимо настроить параметры, для того, чтобы ограничиться остатками только тех товаров, которые мы продаем на дату документа «Расходная накладная». В противном случае (если не задать параметры) будут:

– получены остатки на текущую дату (это плохо при создании документов «задним числом» или при перепроведении старых документов);

– получены остатки по всем товарам, т.е. все записи регистра. Это существенно снижает быстродействие: представьте, что вы продаете 5 номенклатурных позиций, а запрос получит остатки по тысячам позиций, т.е. вернет излишнюю информацию.

Для задания параметров используется кнопка  (рис. 9.6).

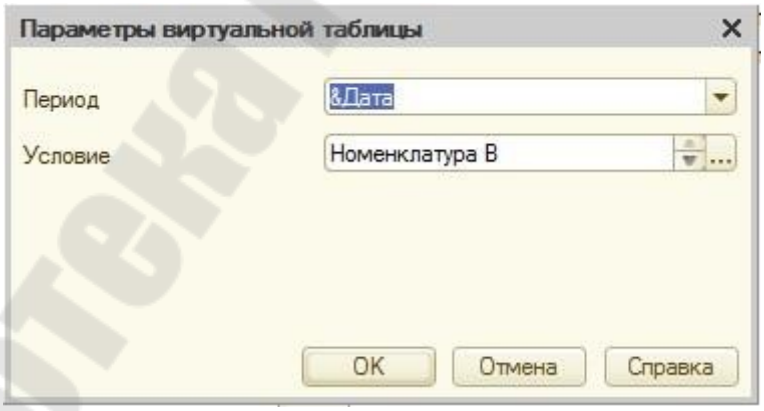


Рисунок 9.6 – Параметры виртуальной таблицы Остатки

В поле «Период» укажем: ***&Дата***. Тем самым из модуля объекта в запрос будет передана дата документа, на которую необходимо проверить остатки. В поле «Условия» можем указать дополнительные условия на измерения регистра накопления. Это

необходимо для того, чтобы ограничить число возвращаемых результатов. В нашей задаче мы должны ограничиться получением остатков лишь по той номенклатуре, которую в данный момент продаем, а не по всей, что есть. Запишем следующее условие, нажав  :

```
Номенклатура В (ВЫБРАТЬ
 ТоварыТЧ.Номенклатура
ИЗ
 ТоварыТЧ КАК ТоварыТЧ)
```

Данным «внутренним» запросом мы выбираем сгруппированную ранее номенклатуру из временной таблицы. К полям «**КоличествоОстаток**» и «**СтоимостьОстаток**» дополнительно применяем функцию ***ЕстьNULL()***. Это необходимо для того, чтобы при отсутствии значений (нет записей об остатках) запрос возвращал не NULL, а некоторое число (в нашем случае ноль). В противном случае, мы не сможем сравнить требуемое для продажи количество товара с NULL. Для применения функций языка запросов, а также любого иного редактирования полей, возвращаемых запросом, используется  . Задаем следующие функции вместо полей «**КоличествоОстаток**» и «**СтоимостьОстаток**» соответственно:

```
ЕстьNULL(ОстаткиТоваровОстатки.КоличествоОстаток, 0)
ЕстьNULL(ОстаткиТоваровОстатки.СтоимостьОстаток, 0)
```

В запросе присутствуют более одного источника (таблицы), поэтому их необходимо связать для соответствия записей (полное, левое и т.д. соединения). В нашем случае необходимо использовать левоесоединение таблиц: выбираются все записи из «**ТоварыТЧ**» (продаваемые товары), и для тех записей, для которых есть соответствие в «**ОстаткиТоваровОстатки**» (т.е. есть остатки по продаваемым товарам) происходит присоединение этих записей из второй таблицы (остатков). Связь производится по полю «**Номенклатура**» (рис. 9.7).

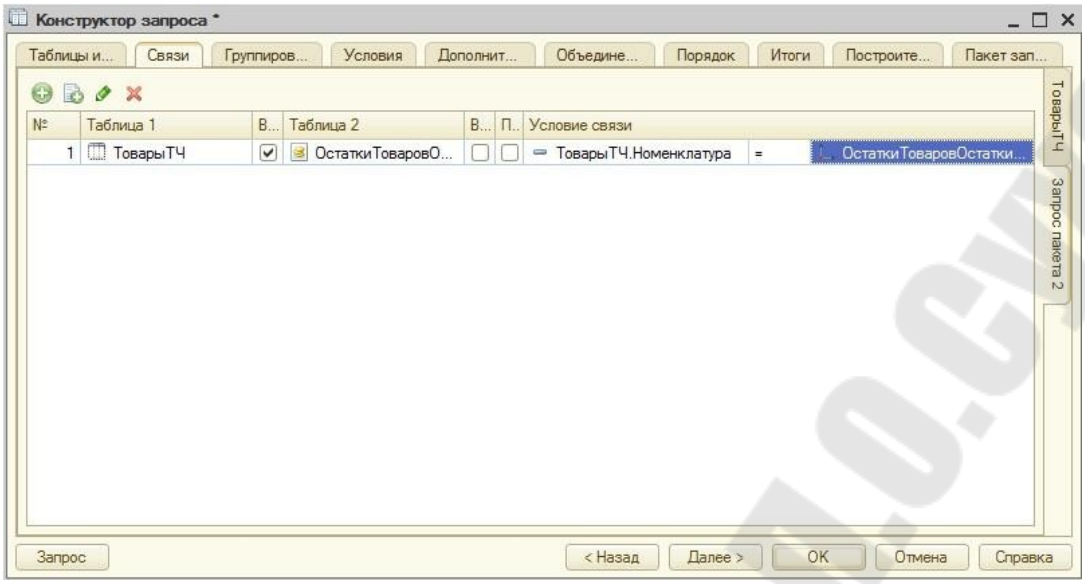


Рисунок 9.7 – Настройка левого соединения таблиц-источников

В заключении на вкладке «Объединения/Псевдонимы» отредактируем имена полей для остатков количества и стоимости: «**КоличествоОстаток**» и «**СтоимостьОстаток**». На этом формирование запроса завершено, возвращаемся в модуль объекта.

Т.к. производилось редактирование лишь текста запроса, то новые параметры необходимо указать самостоятельно. Так, после установки параметры «Ссылка», добавим следующий текст:

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

Затем исправим цикл обхода результата, в котором для каждого элемента выборки результата запроса необходимо произвести контроль остатков. Для этого перед формированием движений добавим следующую проверку:

```
Если ВыборкаДетальныеЗаписи.КоличествоОстаток <
ВыборкаДетальныеЗаписи.Количество Тогда
Сообщить("Не хватает "+ВыборкаДетальныеЗаписи.Номенклатура+" в количестве "+
(ВыборкаДетальныеЗаписи.КоличествоВыборкаДетальныеЗаписи.КоличествоОстаток));
Отказ = Истина;
Движения.Продажи.Записывать = Ложь;
Движения.ОстаткиТоваров.Записывать = Ложь;
Продолжить;
КонецЕсли;
```

В этом листинге проверяется, что в остатках товара меньше, чем требуется. Если это так, то формируется сообщение, в котором указано, сколько и какого товара не хватает. Далее идет инструкции прерывающие транзакцию проведения документа и запрещающие запись в регистры. В конце команда ***Продолжить*** прерывает текущую итерацию цикла и переходит к следующему элементу выборки (чтобы проверить остатки по всем продаваемым товарам).

Отметим, что при проведении продажи в регистр накопления «**Остатки товаров**» записывается сумма продажи. Это не совсем верно: ведь мы должны списывать себестоимость (именно она хранится в регистре), а не сумму продажи. Себестоимость в самом простейшем случае рассчитывается по методу средневзвешенной скользящей:

```
СебестоимостьПродажи =
(КоличествоПродажи/КоличествоОстаток)*СебестоимостьОстаток
```

Поэтому исправим движение по себестоимости (стоимости) в регистре «**Остатки товаров**» (в «**Продажах**» все верно, мы указываем выручку) следующим образом:

```
Движение.Стоимость = ВыборкаДетальныеЗаписи.Количество /
ВыборкаДетальныеЗаписи.КоличествоОстаток *
ВыборкаДетальныеЗаписи.СтоимостьОстаток;
```

Запустите систему и проверьте механизм контроля остатков. Следует отметить, что при работе с регистрами в большинстве случаев удобнее и эффективнее работать именно с виртуальными, а не реальными таблицами. Т.к. система формирует их «на лету», то можно достаточно гибко управлять их содержимым через настройку параметров. Использование параметров виртуальных таблиц обязательно: чем больше параметров мы укажем, тем сильнее ограничим возвращаемые данные лишь необходимыми, тем быстрее система обработает запрос. Согласно концепции системы, вначале формируются виртуальные таблицы, а только потом запрос обрабатывает данные. Поэтому чем меньше исходных данных, тем эффективнее работа самого запроса.

10 ПЛАН ВИДОВ ХАРАКТЕРИСТИК. ОТЧЕТЫ

10.1. План видов характеристик

В реальных учетных системах часто возникает ситуации, когда необходимо задать какие-то характеристики: товаров, сотрудников, материалов и т.д. В разрабатываемой системе необходимо указывать ряд свойств товаров (цвет, размер, производитель и т.д.) и их значения. При этом заранее неизвестно, какой товар, какими характеристиками будет описываться и сколько их будет.

Для задания характеристик можно завести специальные реквизиты: для описания вида и значения характеристики. Реквизит, хранящий значение характеристики, следует сделать составного типа (т.к. характеристики могут быть различного типа). Соответственно, везде в конфигурации, где следует описывать характеристику, для реквизита-значения необходимо заполнять данный составной тип. Такой подход несет в себе ряд проблем, особенно при добавлении новой характеристики нового типа: в больших конфигурациях нереально будет перебрать все объектам и настроить соответствующие реквизиты-характеристики и реквизиты-значения с их составными типами.

Еще одна проблема, которая возникает при описании характеристик – это характеристики «нестандартных», собственных типов. Допустим, с размером или датой производства все понятно – это будут число или дата соответственно. А как, к примеру, быть с цветом (красный, желтый, и т.д.)? Если цвет задать как строку, то тогда:

– необходимо будет каждый раз вводить одинаковые значения, что неэффективно;

– невозможно проводить аналитику: к примеру, сколько товаров из США, или сколько материалов красного цвета и т.д.

Имеет смысл реализовать возможность создания пользователем значений таких «нестандартных» характеристик для их дальнейшего использования и как-либо образом хранить их в базе. Перечисление использовать нельзя, т.к. в режиме исполнения мы не сможем добавлять новые значения. Для хранения значений «нестандартных» типов возможно использование отдельного справочника, но необходимо как-то связать значение характеристики с ее видом.

Для этого «1С: Предприятие» имеется прикладной объект «План видов характеристик». План видов характеристик (ПВХ) – это тоже справочник. Данный справочник состоит из элементов – наименований (видов) характеристик с указанием типа значения для каждой характеристики (число, дата, ссылка на перечисление, и т.д.). Для «нестандартных» характеристик реализована возможность хранения их значений в отдельном подчиненном справочнике (в качестве «Владельца» будет указан элемент ПВХ – вид характеристики).

При создании ПВХ в систему добавляются два новых типа данных: «ПВХСсылка» – ссылка на виды характеристик (элементы ПВХ) и «ХарактеристикаПВХ» – составной тип данных, содержащий перечень типов, которые могут быть заданы для значений характеристик. Соответственно, в системе при описании реквизитов-значений характеристик указывается тип данных «ХарактеристикаПВХ». Если впоследствии, в ПВХ добавится новая характеристика с новым типом данных, то все описанные ранее реквизиты-значения автоматически будут «видеть» этот тип данных.

Значения характеристик можно хранить в реквизитах шапки, реквизитах табличных частей, в ресурсах регистра сведений. Использование реквизитов шапки весьма ограничивает возможности по хранению нескольких характеристик. Использование табличной части, являясь самым простым способом, решает данную задачу, но приводит к тому, что значения характеристик могут быть неуникальными. К примеру, в одной строке можно указать, что цвет красный, а в другой – что зеленый. Вторая особенность, связанная с использование табличной части, заключается в трудностях при проведении аналитики по характеристикам.

Поэтому, наиболее корректным способом, позволяющим реализовать контроль уникальности характеристик, является использование регистра сведений: в измерениях указывается набор, определяющий уникальность характеристик (например, номенклатура плюс вид характеристики), а в ресурсе хранится значение характеристики.

Создадим ПВХ «**Характеристики номенклатуры**», добавим в подсистему «**Отдел закупок**». На вкладке «Основные» настройки свойств в разделе «Тип значения характеристик» (рис. 10.1) отметим «Составной тип данных» и укажем все примитивные типы, а также ссылки на справочник «**Номенклатура**» и перечисление «**Виды**

номенклатуры». На самом деле, здесь можно указать абсолютно произвольно число типов, входящих в составной: ни на производительность, ни на объем хранимой информации это не влияет.

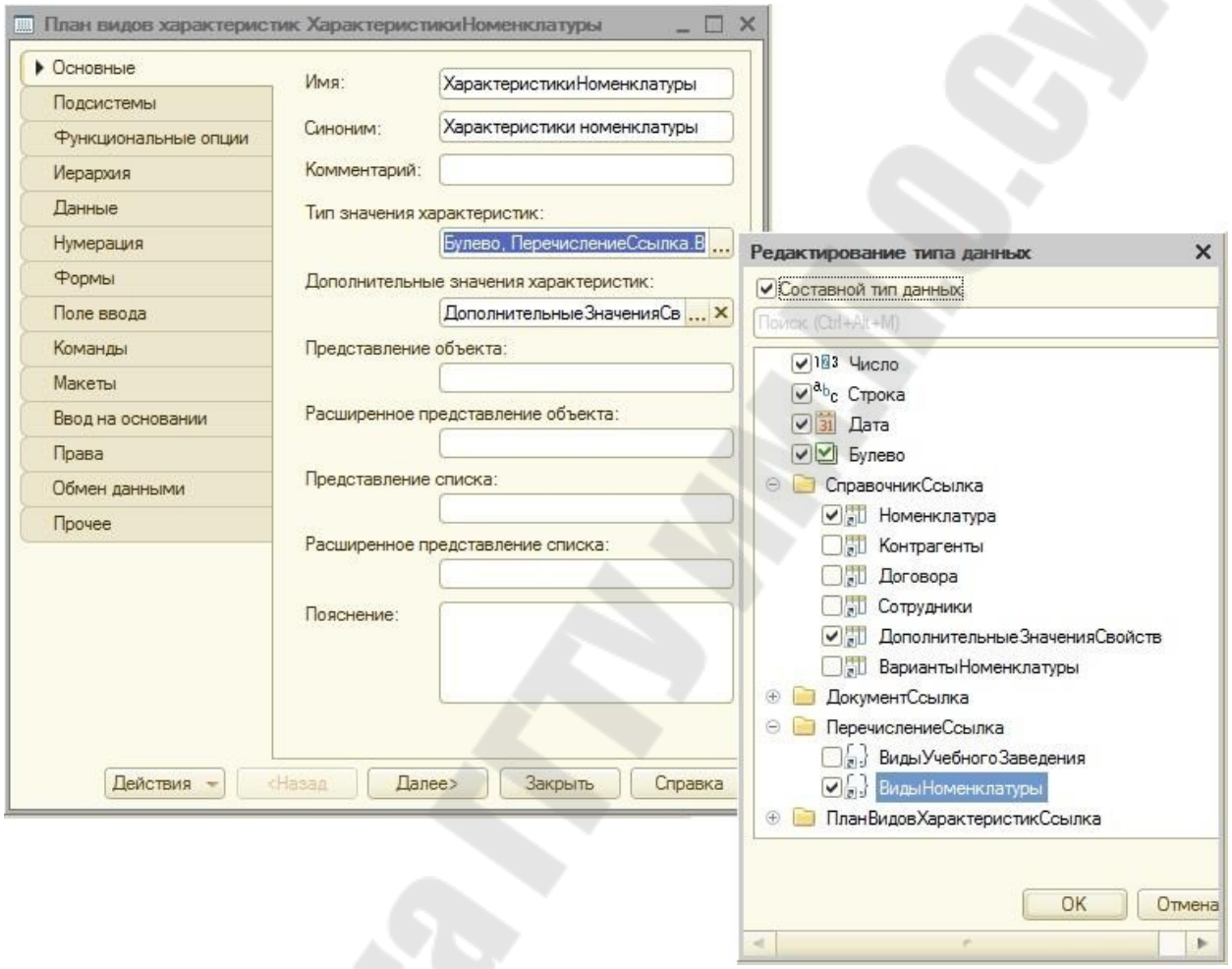


Рисунок 10.1 – Создание плана видов характеристик

Далее создадим справочник «**Дополнительные значение свойств**», включим в подсистему «**Отдел закупок**» и сделаем его подчиненным созданному ПВХ «**Характеристики номенклатуры**». Данный справочник будет хранить значения «нестандартных» характеристик. Подчинение позволяет однозначно связать вид характеристики со значением.

В ПВХ добавим в «Тип значения характеристик» созданный подчиненный справочник. В разделе «Дополнительные значения характеристик» (рис. 10.1) также укажем созданный подчиненный справочник: тем самым мы указываем системе, где будут храниться

значения «нестандартных» характеристик. Также данная настройка приводит к тому, что при создании новой характеристики система по умолчанию будет создавать характеристику «нестандартного» типа (чаще всего так и бывает), однако это можно изменить.

Реализуем хранение значений характеристик.

Создадим регистр сведений «**Значения свойств товаров**», непериодический, независимый, включим в подсистему «**Отдел закупок**». Измерения: «**Номенклатура**» (тип СправочникСсылка.Номенклатура, Ведущее) и «**Свойство**» (тип ПВХСсылка.ХарактеристикиНоменклатуры); ресурс – «**Значение свойства**» (тип Характеристика.ХарактеристикиНоменклатуры).

Для ресурса на вкладке «Представление» панели свойств настроим «Связи параметров выбора»: отбор по владельцу для измерения «**Свойство**». Это необходимо для того, чтобы при выборе «нестандартных» характеристик, в форму выбора был представлен список не всех значений, хранящихся в подчиненном справочнике, а только тех, которые связаны с выбранным ранее «нестандартным» видом характеристики (аналогично использованию связей параметров выбора при работе с подчиненным справочником).

Дополнительно для ресурса следует настроить «Связь по типу», где указать измерение «**Свойство**». Дело в том, что тип значений «Характеристика» – это составной тип данных, соответственно, ресурс может принимать значение любого из перечисленных в нем типов. Мы, в свою очередь, знаем, что ресурс должен быть того типа, какой тип у выбранного вида характеристики, указанной в измерении «Свойство». Поэтому реализуется такая связь по типу.

Запустите систему в режиме отладки и проверьте реализованный механизм: задайте несколько характеристик стандартных (например, дата производства) и «нестандартных» (например, цвет и страна происхождения) типов. Для «нестандартных» характеристик заполните ряд значений. Заполните регистр сведений характеристиками некоторых товаров. Проверьте контроль уникальности.

10.2 Отчеты

Отчеты – это то, к чему в итоге сводится деятельность организации. Необходимо получать итоговые отчеты о том, сколько и какого товары было продано за определенный период и на какую

сумму; какую выручку получила организация; кто из сотрудников, сколько договоров заключил и какую прибыль принес; какие контрагенты наиболее часто работают с организацией; сколько товара есть на складах на данный момент; и т.д. Такая сводная, итоговая информация хранится в регистрах (сведений, накоплений и др.), а, если точнее, в виртуальных таблицах, формируемых на основании того, какие именно сгруппированные данные необходимо получить в отчете. Основным инструментом формирования отчетов служат запросы.

В нынешних версиях «1С: Предприятия» для создания отчетов используется специальный инструмент «Система компоновки данных» (СКД).

Создадим новый отчет «**Остатки товаров**», включим в его подсистему «**Отдел закупок**». На вкладке «Основные» окна настройки свойств выберем команду «Открыть схему компоновки данных» и нажмем «Готово». В результате откроется пустая схема компоновки данных, включающая в себя наборы данных для отчета и его настройки (рис. 10.2).

Изначально необходимо при помощи запроса сформировать данные, выводимые в отчет. Для этого добавляем очередной набор данных типа «запрос» (рис. 10.2). В результате формируется пустой набор данных, для которого с помощью специальной кнопки запускаем конструктор запроса.

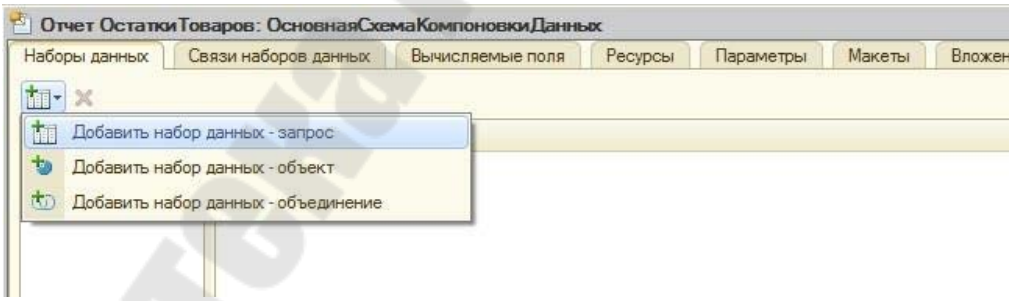


Рисунок 10.2 – Система компоновки данных. Добавление набора данных

Для получения остатков товаров, как и ранее, используем виртуальную таблицу «**ОстаткиТоваровОстатки**». В данной таблице содержится информация только о тех товарах, которые имеются на складе. Если мы хотим вывести остатки по всей номенклатуре, то необходимо с данной виртуальной таблицей левым соединением

связать таблицу – справочник «**Номенклатура**». С учетом того, что в базе реализован учет различных вариаций одного и того же товара, то для получения остатков по вариантам товаров, еще одним источником данных будет справочник «**Варианты номенклатуры**» (который в свою очередь нужно связать со справочником-владельцем – «**Номенклатура**»). В запросе необходимо вывести все элементы справочников «**Номенклатура**» и «**Варианты номенклатуры**», с учетом взаимосвязей между ними, а для тех комбинаций «**Номенклатура+Вариант**», для которых есть остатки – выведем количество имеющегося товара из виртуальной таблицы.

В указанном запросе мы добавили таблицу «**Номенклатура**» (справочник), и в то же время в качестве измерения второго источника (виртуальной таблицы «Остатки») также фигурирует «**Номенклатура**». Данная ситуация с точки зрения конструктора может вызвать неоднозначную интерпретацию того, к чему идет обращение: к справочнику или к измерению регистра. Для исключения конфликта, справочник-источник данных необходимо переименовать, допустим, в «**спрНоменклатура**».

Далее из «**спрНоменклатура**» выбираем поле «**Наименование**», из справочника «**Варианты номенклатуры**» – тоже «**Наименование**», а из виртуальной таблицы остатков – «**КоличествоОстаток**» и «**СтоимостьОстаток**». Переходим на закладку связи и организуем следующие левые соединения (рис. 10.3):

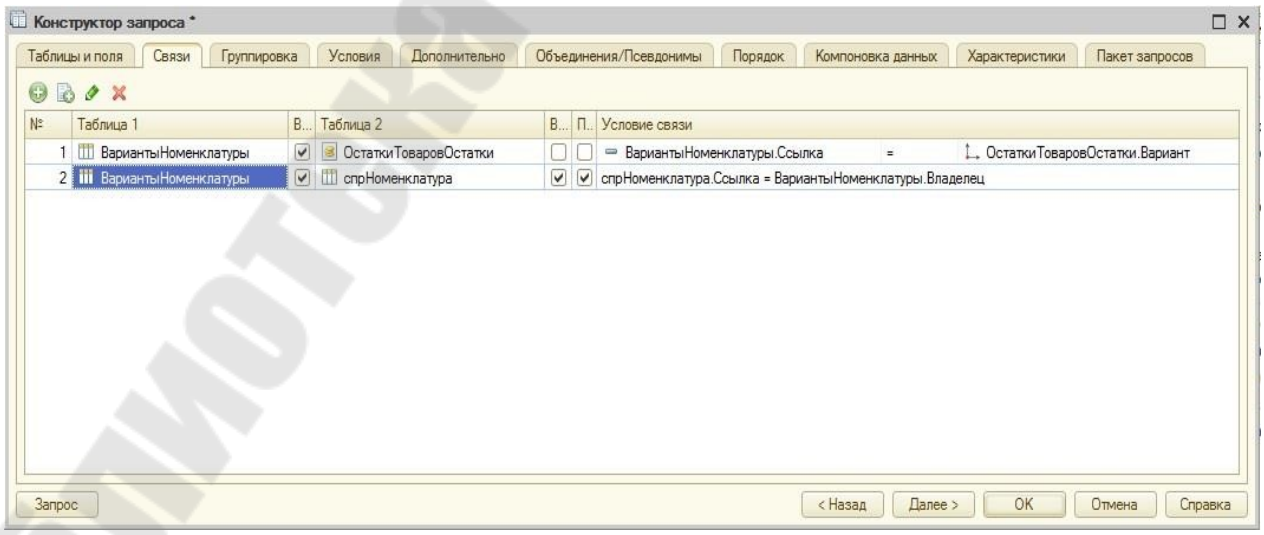


Рисунок 10.3 – Связи между таблицами - источниками данных в отчете «**Остатки товаров**»

Поясним описанные связи. Первая связь – левое соединение таблицы «**Варианты номенклатуры**» и «**ОстаткиТоваровОстатки**»: выбираются все возможные вариации номенклатурных позиций, а для тех из них, для которых есть соответствие в виртуальной таблице, присоединяются соответствующие записи. После введения ПВХ и возможности ведения учета товаров в разрезе вариантов, в регистре накопления хранится информация по каждому варианту, а не по номенклатуре в целом. Вторая связь – это полное внутреннее соединение таблиц «**Варианты номенклатуры**» и «**Номенклатура**»: производится поиск соответствия между всеми элементами справочников по соответствующим полям условия связи (рис. 10.3). Напомним, что справочник «**Варианты номенклатуры**» подчинен справочнику «**Номенклатура**». Поэтому, поле «**Владелец**» у справочника «**Варианты номенклатуры**» заполняется ссылкой на элемент справочника «**Номенклатура**». Описанная связь каждому элементу подчиненного справочника находит его «Владельца» среди элементов справочника-владельца.

Определим ряд условий в запросе на соответствующей вкладке. Справочник «**Номенклатура**» имеет иерархию групп и элементов. Остатки товаров задаются лишь в разрезе элементов, не групп. Соответственно, из результата запроса нужно исключить записи, содержащие группы; добавляем условие:

```
спрНоменклатура.ЭтоГруппа = ЛОЖЬ
```

В справочнике «Номенклатура» есть как товары, так и услуги. Остатки накапливаются лишь в разрезе товаров, поэтому из результата запроса нужно исключить записи, содержащие услуги: необходимо наложить условие на поле «**Вид номенклатуры**». В конструкторе запроса данное поле перенесем из левого окна в правое. Сформированное условие будет в виде:

```
спрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры
```

В тексте запроса указать конкретное значение (хоть «**Вид номенклатуры**» задается в конфигураторе) не удастся т.к. оно хранится в перечислении (а в запрос можно передавать лишь значения предопределенных элементов справочника). Поэтому

&ВидНоменклатуры – это параметр запроса, значение которого мы заполним позже в СКД.

На вкладке «Объединения/Псевдонимы» переименуем поля «**Наименование**» в «**Номенклатура**» и «**Вариант**» для соответствующих таблиц. На этом формирование запроса завершим и вернемся к настройкам СКД (рис. 10.4).

В результате сформированного запроса получим все необходимые данные, поэтому нет необходимости в дополнительных наборах данных. На вкладке «Вычисляемые поля» (рис. 10.4) можно добавить поля, которые будут рассчитываться на основе получаемых из запроса полей.

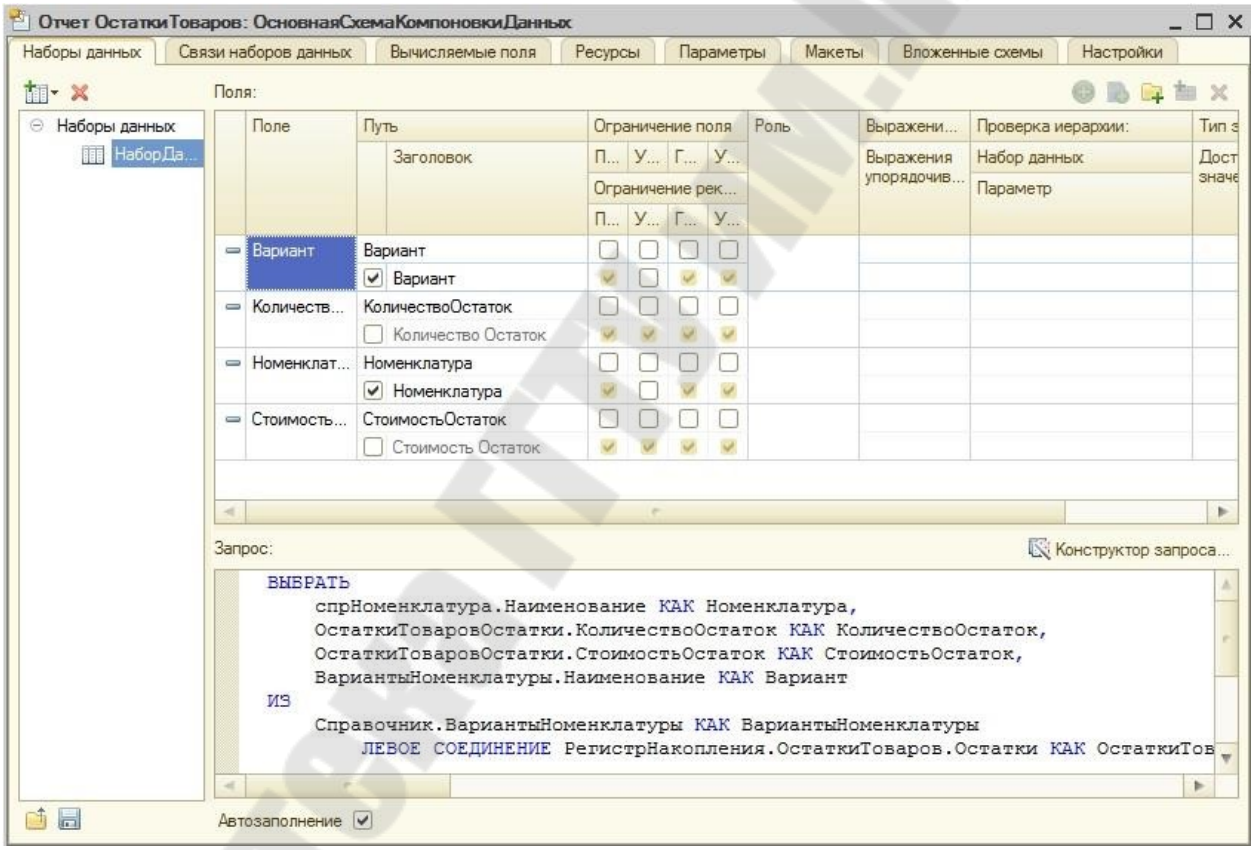


Рисунок 10.4 – Настройка СКД после формирования запроса (набора данных)

Так, необходимо, к примеру, получить среднюю себестоимость единицы каждой номенклатурной позиции. Средняя себестоимость единицы товара – **СтоимостьОстаток/КоличествоОстаток**.

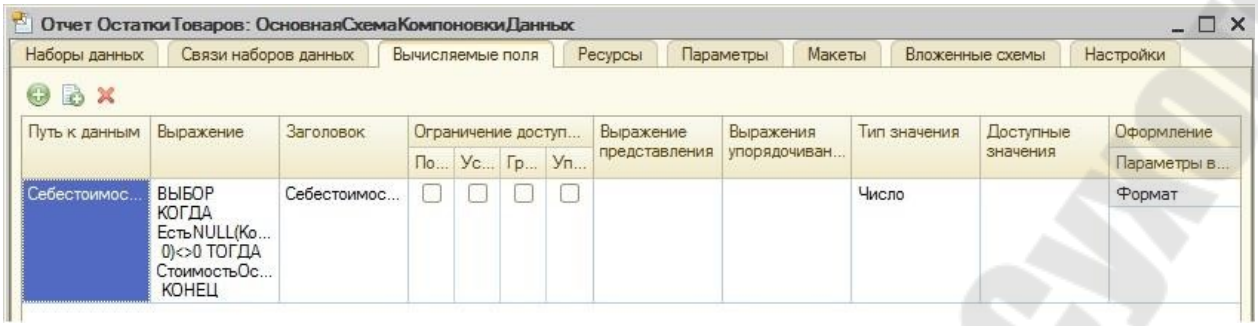


Рисунок 10.5 – Настройка вычисляемых полей

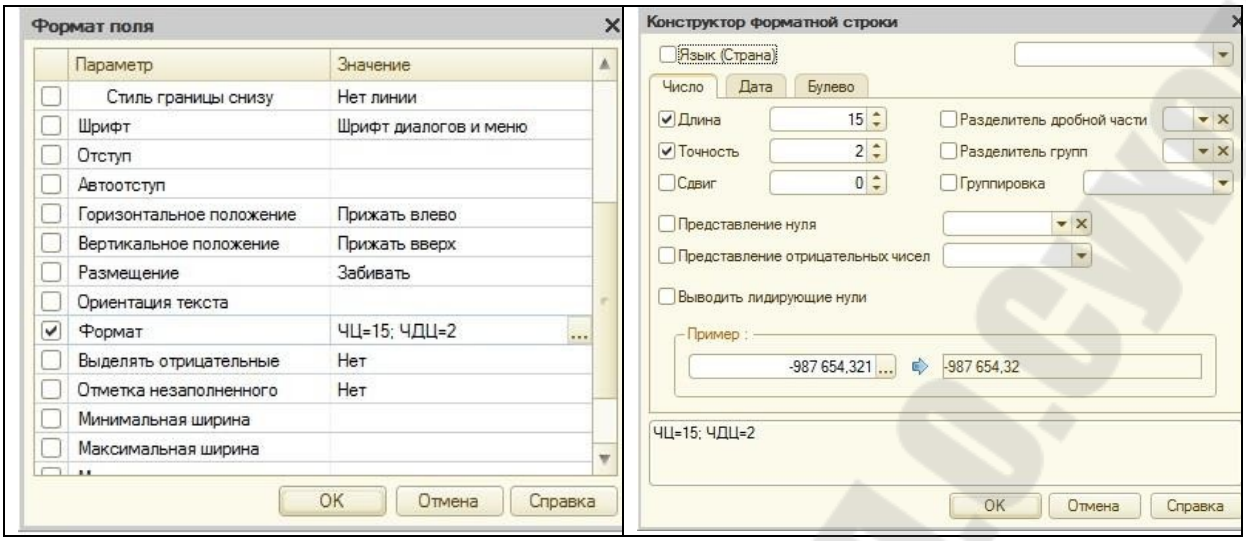
В поле «Путь данным» (рис. 10.5) указываем имя вычисляемого поля «**Себестоимость**». В результате автоматически сформируется «**Заголовок**» (типа псевдонима), который при необходимости можно изменить. В поле «**Выражение**» необходимо задать выражение для расчета. Запишем следующее выражение:

```
ВЫБОР
КОГДА ЕстьNULL(КоличествоОстаток, 0)<>0 ТОГДА
СтоимостьОстаток/КоличествоОстаток
КОНЕЦ
```

Поясним данный код. Поле «**КоличествоОстаток**» может принимать значение NULL (для тех товаров, по которым остатков нет). При помощи функции ***ЕстьNULL*** производится приведение данного поля к заданному числовому представлению – нулю. Также очевидно, что когда остаток равен нулю, себестоимость не должна считаться; во всех остальных случаях она вычисляется по приведенной ранее формуле.

Для реализации такого условного оператора, в языке СКД (как и в языке запросов) есть специальная конструкция ***ВЫБОР КОГДА…ТОГДА***, которую мы используем для проверки на неравенство нулю знаменателя («**КоличествоОстаток**») и вычисления себестоимости когда это допустимо.

В поле «Тип значения» (рис. 10.5) в раскрывающемся списке выбираем: Число, длина 15, точность 2. В конце настраиваем представление данного числа. В поле «Оформление» открываем список, настраивающий формат поля (рис. 10.6 а).



(a) (b)

Рисунок 10.6 – Настройка формата поля (a) при помощи конструктора форматной строки (b)

Для этого выбираем параметр «Формат», и в поле «Значение» открываем конструктор форматной строки (рис. 10.6 b). Данный конструктор в удобном режиме позволяет настроить формат выводимых данных. Производится настройка вывода числа, поэтому указываем длину 15, точность 2. После чего закрываем конструктор форматной строки и настройку формата поля.

На вкладке «Ресурсы» (рис. 10.4) можно добавить те поля, по которым система будет формировать Итоговые (суммируемые записи) по полям группировок (рис. 10.7).

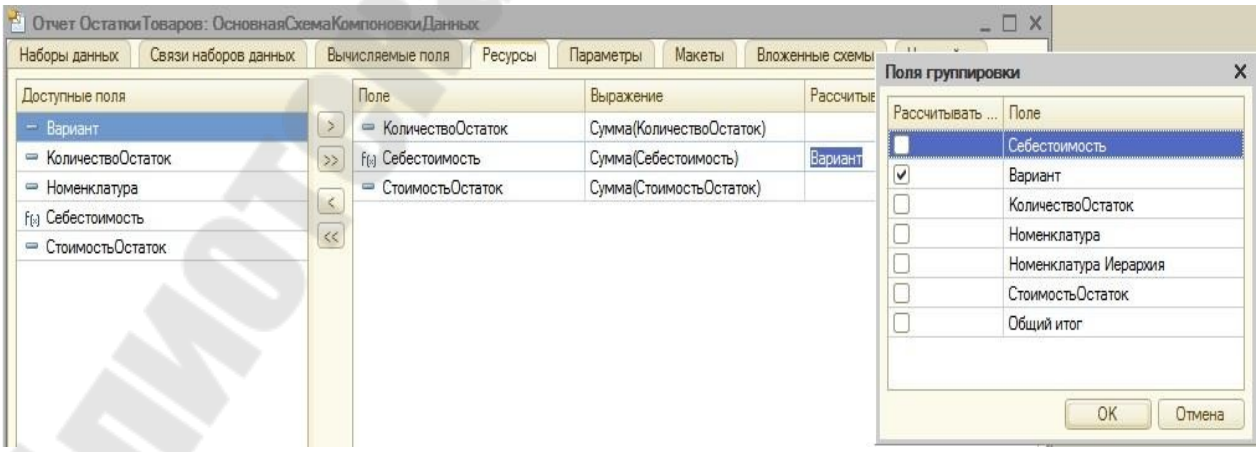


Рисунок 10.7 – Настройка ресурсов в отчете

Поля группировок будут настроены позже, но, забегая вперед, отметим, что ими будет поле «**Номенклатура**»: мы получим остатки не только по каждому варианту некоторой номенклатуры, но и по номенклатуре в целом. Нажав >> (рис. 10.7), мы предоставим системе возможность автоматически определить, какие ресурсы можно считать, т.е. все числовые поля. В поле «Рассчитывать по» вкладки «Ресурсы» для каждого ресурса можно настроить поля запроса, по которым необходимо вычислять итоги. По умолчанию итоги вычисляются по всем полям. В нашем примере, нет смысла считать ресурс «**Себестоимость**» (для единицы товара) по всей номенклатуре, или даже по каждой группе: нас будет интересовать себестоимость лишь отдельных вариантов номенклатуры. Поэтому среди всех полей отметим лишь поле «Вариант».

На вкладке «Параметры» СКД (рис. 10.4) настраиваются и задаются параметры, передаваемые в запрос.

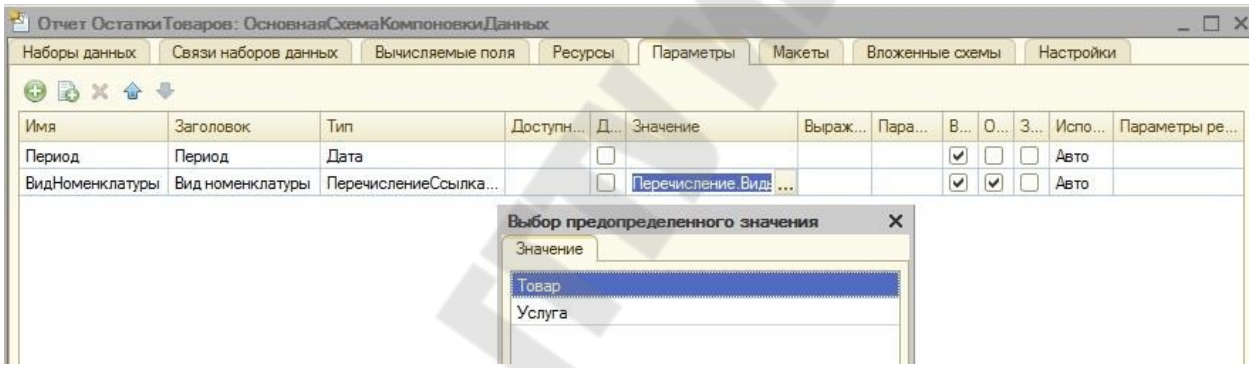


Рисунок 10.8 – Настройка параметров в СКД

Видим два параметра (рис. 10.8): «**Период**» и «**ВидНоменклатуры**». В конструкторе запроса для отчета мы не настраивали параметры виртуальной таблицы остатков неслучайно: СКД уже «знает» что такое виртуальные таблицы и как с ними работать. Поэтому при их добавлении в запрос СКД автоматически определяет параметры, особенно связанные с периодом, которые можно задать. Поэтому ряд «стандартных» параметров виртуальных таблиц можно не задавать вручную – все равно в СКД при формировании отчета они будут присутствовать.

Параметр «**ВидНоменклатуры**» - это параметр, в котором необходимо указать на то, что остатки необходимо определять лишь

для «Товаров», «Услуги» следует исключить из результата запроса. Для этого заполняем поле «Значение» (рис. 10.8) соответствующим образом. Чтобы пользователь не смог этот параметр изменить, отмечаем галочку «Ограничение доступности». Если есть необходимость, чтобы при формировании отчета пользователь мог изменять значение параметра (как, например, для «**Периода**»), то «Ограничение доступности» следует отключить. В представленной настройке (рис. 10.8), также возможно добавить свои собственные параметры, настроить их тип, значение и т.д.

На вкладке «Настройка» (рис. 10.4) формируется внешний вид отчета. Все, что было сделано до этого, относилось лишь к данным, которыми оперирует и которые выводятся в отчете. А то, как эти данные будут отображены в интерфейсе, устанавливается лишь при настройке отчета. Во-первых, сразу отметим, что в СКД есть возможность создать несколько типов отчета (списком, таблицей, диаграммой). Дополнительно можно сделать несколько различных вариантов (рис.10.9), а в режиме исполнения выбирать нужный способ представления.

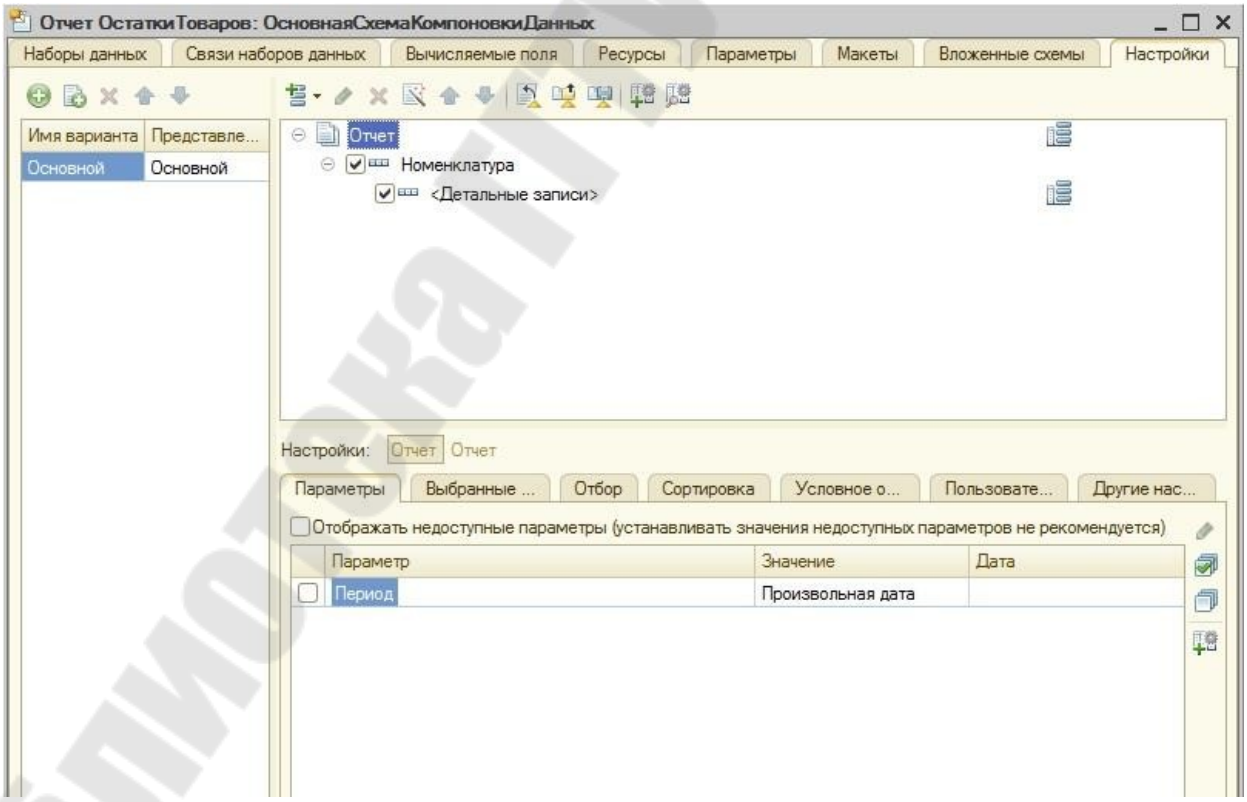


Рисунок 10.9 – Настройка вариантов отчета в СКД

Для формирования любого из вариантов, самый простой способ – использование конструктора настроек. Для этого в панели инструментов нажмем  (рис. 10.9). Выберем формирование отчета в виде списка. Далее укажем вывод в отчет всех полей. Далее укажем группировку по полю «**Номенклатура**», тип группировки «Иерархия». При настройке варианта отчета возможно три типа группировки: без иерархии (будут выводиться все записи без учета иерархии), иерархия (будут выводиться все записи с учетом иерархии), только иерархия (будут выводиться лишь записи верхнего уровня, т.е. только элементы-родители). На заключительной вкладке конструктора настроек не будем задавать упорядочение.

В результате сформируются настройки по умолчанию. Сформированные настройки в дальнейшем можно изменить: отредактировать выбранные поля, настроить условное оформление и т.д. (рис. 10.7). При этом все настройки можно выполнить как для отчета в целом («Отчет»), так и для отдельных группировок («Номенклатура», «Детальные записи»), что указано на рис. 10.7. Следует отметить, что группировка «Детальные записи» существует всегда, вне зависимости от того, были иные группировки или нет. Данная группировка содержит все записи, возвращаемые запросом в наборе данных.

На вкладке «Параметры» окна «Настройка» (рис. 10.9) необходимо указать параметры, которые необходимо отображать в интерфейсе. Здесь представлен список параметров, указанных в соответствующем разделе ранее (рис. 10.8) для которых не установлен флаг «Ограничение доступности». В данном случае – среди параметров присутствует только «Период», для указания даты, на которую необходимо получить остатки номенклатуры. Для отображения данных параметров в интерфейсе, указанные настройки необходимо включить в «Пользовательские настройки» (рис. 10.10) при помощи  (рис. 10.9).

«Быстрый доступ» (рис. 10.10) необходим для того, чтобы при запуске отчета указанные параметры отображались на главной странице отчета. Если режим редактирования изменить на «Обычный», то для задания параметров отчета необходимо перейти в раздел «Еще - Настройка».

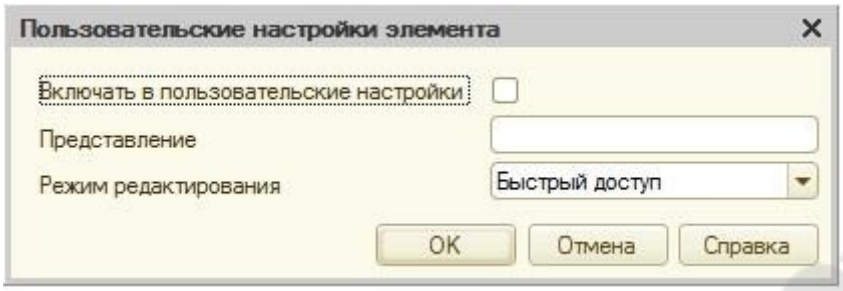


Рисунок 10.10 – Свойства элементов пользовательских настроек

После того, как все указанные действия (формирование набора данных запросом, создание вычисляемых полей, задание ресурсов, параметров отчета, настройка представления отчета) выполнены, можно запустить систему в режиме исполнения и сформировать отчет на текущую дату, а также на ряд предыдущих (чтобы посмотреть, как изменялись остатки). Обратите внимание на формирование итоговых записей (те поля, которые были добавлены в ресурсы отчета).

11 ДОРАБОТКА ИНТЕРФЕЙСА КОНФИГУРАЦИИ

В целом, можно сказать, что на этом разработанное прикладное решение уже представляет собой некоторый пусть черновой, упрощенный, но законченный вариант по автоматизации деятельности нашей организации (за исключением бухгалтерского и зарплатного учета). Сейчас уделим еще немного времени настройке интерфейса.

11.1. Использование общих форм

До этого момента создавали формы для конкретных объектов, а также назначали основную форму определенного типа. В системе также имеется возможность создания ряда общих форм, связь их с объектами, и добавление в интерфейс.

Добавим общую форму констант. Откроем ветку «Общие», выберем «Общие формы» и создадим новую форму. Укажем тип формы – «Форма констант», назовем форму «**Параметры учета**». На следующей вкладке отметим все константы для отображения в форме и нажмем «Готово». Добавим данную форму в подсистему «**Общий отдел**». Дополнительно, в свойствах каждой константы на вкладке

«Представление» настроим основную форму – укажем только что созданную общую форму. Запустим систему в режиме отладки и теперь попробуем заполнить значения имеющихся констант. Вы увидите, что все константы отображаются в одной форме, что, несомненно, является удобным. Теперь все константы можно убрать из меню «Сервис», оставив только форму «Параметры учета». Для этого необходимо настроить командный интерфейс подсистемы.

11.2 Настройка командного интерфейса

Командный интерфейс – средство доступа пользователя к функциональности системы, позволяет перемещаться между формами и выполнять те или иные действия. Командный интерфейс в «1С: Предприятие» 8.3 разработчик не прорисовывает досконально, а описывает декларативно. Командный интерфейс состоит из подсистем, со всеми формами и командами в них содержащимися, а также включает в себя настройку ролей – разграничение прав доступа к отдельным составляющим системы.

Для настройки командного интерфейса подсистем (какие команды, и в каком порядке отображать) необходимо выбрать в контекстное меню раздела «Подсистемы» пункт «Все подсистемы». Для каждой подсистемы необходимо настроить свойство «Видимость» напротив команд панели действий, навигации и др. (рис. 11.1).

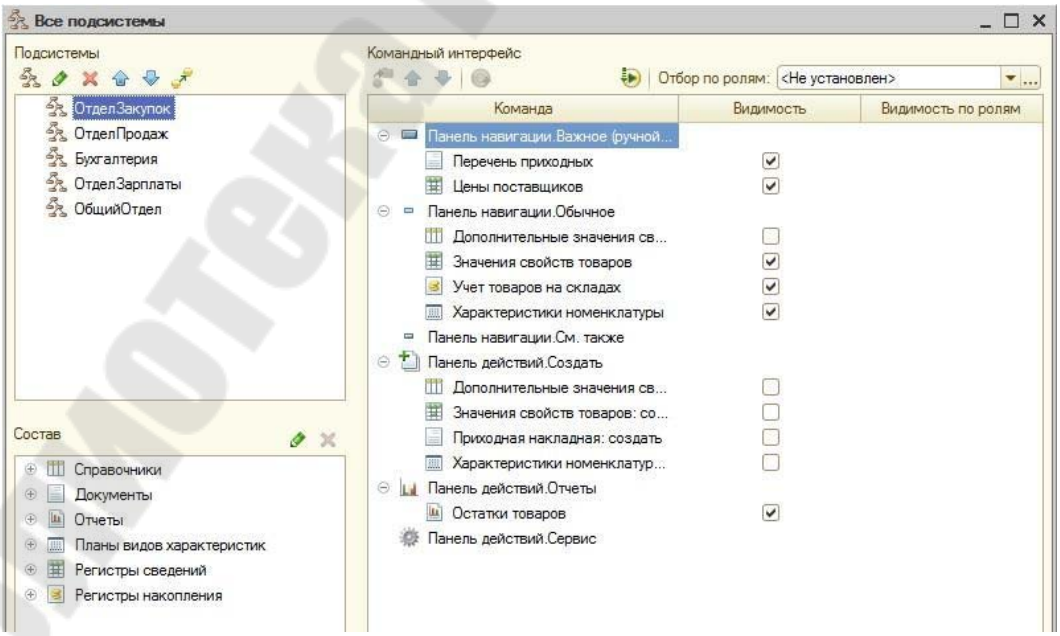


Рисунок 11.1 – Настройка командного интерфейса подсистем

Также при настройке командного интерфейса конфигурации все команды панели навигации можно разделить по трем категориям: важное, обычное и смотри также (рис. 11.1). Команды, помещенные в раздел «Важное» в интерфейсе помечаются жирным шрифтом. При работе в версии платформы 8.2, в интерфейсе раздел «Смотри также» дополнительно выделяется в отдельную область: в версии 8.3 особым образом выделен только раздел «Важное».

Для подсистемы «Общий отдел» из «Панели действий. Сервис» скроем команды открытия констант, и оставим команду открытия общей формы «**Параметры учета**». В «Панели действий. Создать» отметим видимость для команд создания элементов справочников «**Контрагенты**» и «**Номенклатура**». Также из «Панели навигации. Обычное» уберем видимость команд перехода к справочникам «**Варианты номенклатуры**» и «**Договора**».А переход к справочникам «**Контрагенты**» и «**Номенклатура**» переместим в раздел «Панель навигации. Важное».

Аналогичным образом настройте командный интерфейс других подсистем (на ваше усмотрение).

Для настройки командного интерфейса конфигурации в целом, необходимо вызвать контекстное меню всей конфигурации и выбрать пункт «Командный интерфейс». Здесь можно настроить видимость подсистем в панели разделов и порядок их расположения. Зададим следующий порядок расположения подсистем:

1. Общий отдел;
2. Отдел закупок;
3. Отдел продаж;
4. Бухгалтерия;
5. Отдел зарплаты;

Запустите систему в режиме отладки и посмотрите, как изменился командный интерфейс.

11.3 Критерии отбора

Допустим, перед нами стоит задача: при открытии элемента справочника «**Номенклатура**» определить, в каких документах продажи и поступления товара фигурирует данная номенклатурная позиция. Для этого необходимо сформировать специальный отбор, накладываемый на соответствующие реквизиты указанных

документов. Для этих целей в системе существует специальный объект «Критерии отбора», находящийся в ветке «Общие».

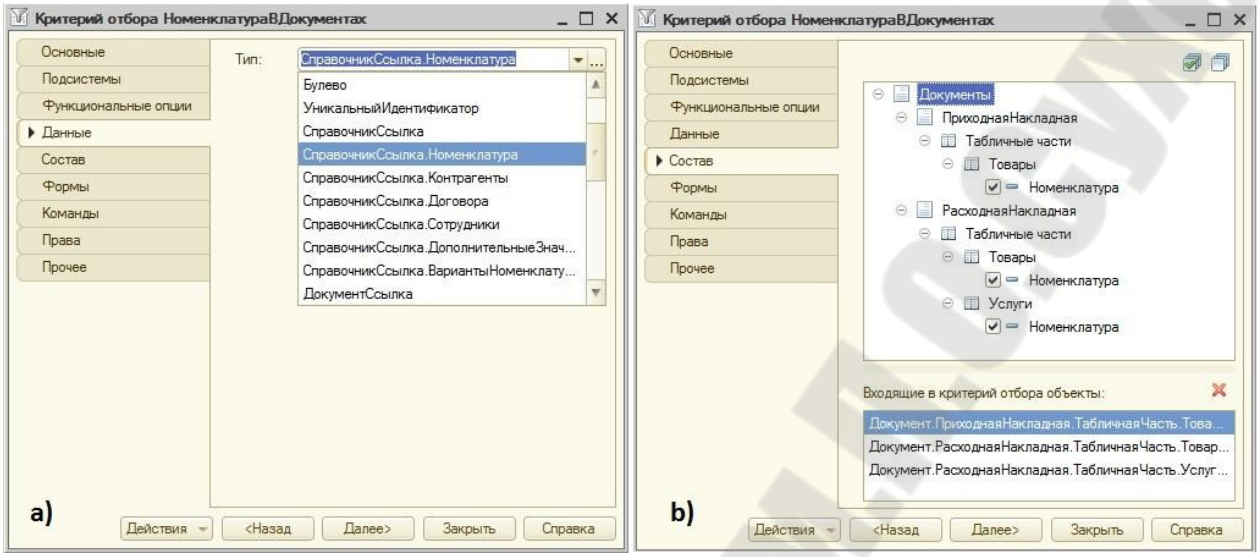


Рисунок11.2 – Критерий отбора

Добавим критерий отбора «Номенклатура в документах», добавим в подсистему «**Общий отдел**». На вкладке «Данные» (рис. 11.2 a) укажем тип отбора – СправочникСсылка.Номенклатура. На вкладке «Состав» (рис. 11.2 b) укажем соответствующие реквизиты в документах, на которые будет накладываться условие отбора (система автоматически определит реквизиты, совпадающие по типу с типом отбора, указанным на вкладке «Данные»).

Для возможности использования данного критерия (перехода к соответствующим документам), в панель навигации формы элемента справочника «**Номенклатура**», необходимо добавить соответствующую команду. Для этого откроем форму документа, в разделе «Элементы» перейдем на вкладку «Командный интерфейс», и в «Панели навигации» настроим «Видимость» для соответствующей команды.

Запустите систему в режиме отладки и проверьте работу механизма.

11.4 Функциональные опции

В ряде случаев могут возникать ситуации, когда в различных организациях используются очень близкие конфигурации, за исключением наличия/отсутствия определенной функциональности.

Допустим, несколько филиалов одной фирмы. Но в одном из филиалов, к примеру, не ведется бухгалтерия, а в другом не занимаются учетом заработной платы. Для того чтобы использовать близкие конфигурации с несколько отличающейся функциональностью нет необходимости разрабатывать две и более конфигурации: достаточно реализовать механизм включения/отключения требуемой функциональности.

Для этих целей в системе «1С: Предприятие» введен механизм использования функциональных опций. Добавим функциональную опцию «Бухгалтерский учет». В поле «**Хранение**» (рис. 11.3 a) укажем, где будет храниться значение функциональной опции. Значение функциональной опции можно хранить в константе, реквизите справочника или в ресурсе регистра сведений. При использовании реквизита или ресурса регистра, для включения/отключения опции, и ее настройки необходимо использовать «Параметры функциональных опций» и реализовывать достаточно сложные механизмы. Самый простой вариант – сохранить значение опции в константе. Соответственно, если константа примет значение Истина, то опция включена, если Ложь – выключена.

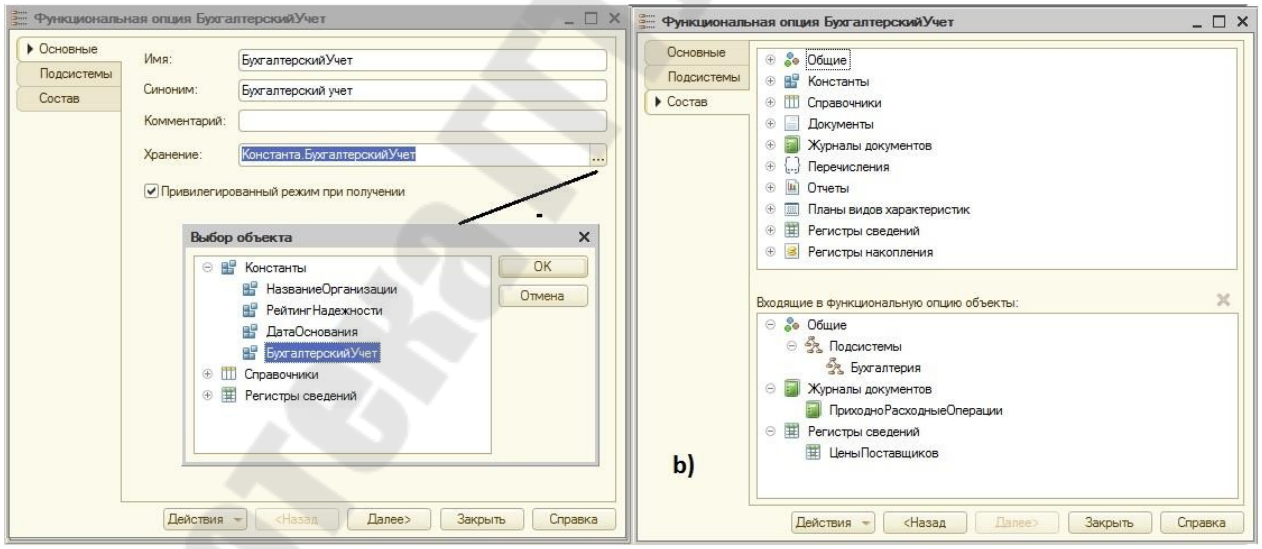


Рисунок 11.3 – Функциональная опция «Бухгалтерский учет»

Сохраним значение опции «**Бухгалтерский учет**» в константе «**Бухгалтерский учет**». На вкладке «Состав» (рис. 11.3 b) необходимо настроить объекты системы, включенные в данную опцию. При включении опции соответствующие объекты будут доступны, при выключении – нет. Включим, к примеру, в опцию

подсистему «**Бухгалтерия**», журнал документов «**Приходно-расходные операции**» и регистр сведений «**Цены поставщиков**».

Опция при этом влияет только на интерфейс, никакая реструктуризация таблиц базы данных не происходит. Т.е., если, допустим, в системе был реализован многоскладской учет. Затем была добавлена опция «**Учет склада**», в которую включили все справочники, реквизиты, измерения и т.д. связанные со складами. Выключив опцию, мы лишь убрали отображение соответствующих объектов в интерфейсе. Но все, что было сделано до этого, не исчезло. При включении опции, все данные будут отображаться как раньше.

Запустим систему в режиме отладки. В форме «**Параметры учета**» выключим соответствующую опцию. Теперь, переходя к журналу документов или регистру сведений, мы увидим, что они пустые (вернее, не отображаются). Но к подсистеме «**Бухгалтерия**» все равно есть доступ. Перезапустим систему. После перезапуска мы увидим, что ссылки на все объекты, включая подсистему исчезли. Это связано с тем, что интерфейс при изменении опции автоматически не обновляется. После перезапуска система загрузилась с учетом настроек опции.

Для того чтобы система обновлялась динамически при включении/выключении опции необходимо в режиме Конфигуратора открыть общую форму «**Параметры учета**», и описать событие формы «***ПослеЗаписи***» на клиенте, добавив следующую строчку кода:

```
ОбновитьИнтерфейс();
```

Запустите систему и проверьте результат.

11.5 Команды

Ранее было рассмотрено создание команд формы. Кроме таких команд имеется возможность создания общих (глобальных команд), а также целых групп команд (в группу команд можно занести множество различных команд).

Для начала создадим группу команд «**Печать**» (в ветке «Общие»). В настройке свойств группы имеется раздел «Категория», указывающий на то, в какой части командного интерфейса будет располагаться данная группа команд:

– в панели навигации (где мы переходим по ссылкам к объектам конфигурации, например, к определенному справочнику);

– в панели навигации формы (где мы переходим к связанным с формой объектам, например, к элементам справочника, подчиненного данному, с отбором по текущему элементу);

– в панели действий (где находятся разделы «Сервис», «Создать» и т.д.);

– в командной панели формы (где находятся стандартные команды работы с формой, например, «Записать», «Закрыть» и т.д.);

Созданную группу команд отнесем к «Командной панели формы».

Далее опишем создание команды «**Печать расходной**», которая будет формировать простую печатную форму документа «**Расходная накладная**». Данная команда не будет относиться к какой-то конкретной форме документа, списка и т.д. Поэтому она не может быть создана при описании какой-либо формы. Для создания такой общей команды, связанной с объектом «**Расходная накладная**», откроем окно настройки данного объекта конфигурации, и перейдем к вкладке «Команды». Здесь можно описать собственные команды и отнести их к конкретной группе, в т.ч. созданной выше.

Однако для создания команды формирования печатной формы документа необходимо знать очень много особенностей, связанных с тем, как получить данные и как сформировать эту печатную форму. На данном этапе обучения мы не будем глубоко вдаваться в эти особенности, а воспользуемся конструктором. Для этого откроем вкладку «Макеты» и запустим единственно доступный конструктор – «Конструктор печати».

Макеты необходимы для создания различных печатных форм. Макет представляет собой табличный документ с именованными областями. Каждая область может содержать либо текст, либо параметр (в который передаются данные), либо параметр расшифровки (для перехода к связанным объектам конфигурации). Для заполнения макета существует специальная процедура: она считывает необходимую информацию из таблиц, получает очередную область макета, заполняет указанную область считанными данными, после чего выводит сформированный макет в табличный документ и отображает его. Т.к. зачастую макеты используются для создания печатной формы не одного документа, а целого перечня, то оказывается, что процедура не связана с конкретным экземпляром

объекта, и, поэтому не может располагаться в модуле объекта. Данная процедура заполнения макета связана с объектом конфигурации в целом, без привязки к конкретному экземпляру, поэтому она размещается в специальном модуле – модуле менеджера, который отвечает за управление объектом конфигурации в целом, а не отдельным его экземпляром. Дополнительно необходимо создать команду, которая будет вызывать данную процедуру формирования печатной формы. Конструктор печатной формы все указанные действия делает автоматически.

В конструкторе укажем имя команды «**Печать расходной**». Далее в реквизитах шапки укажем печать реквизитов «**Дата**», «**Контрагент**», «**Договор**». Далее укажем, что из табличной части «**Товары**» будем выводить на печать все доступные реквизиты. Далее, то же самое сделаем и для табличной части «**Услуги**». Далее в подвале будем печатать содержимое реквизита «**Сумма документа**». На заключительной вкладке укажем, что созданная команда должна располагаться в группе команд «**Командная панель формы. Печать**» (созданная ранее группа команд). После завершения работы конструктора автоматически будет создана команда «**Печать расходной**», соответствующий макет, представляющий вид формируемой печатной формы, а также команда в модуле менеджера, заполняющая макет требуемыми данными.

Запустите систему в режиме отладки и посмотрите результат. Обратите внимание, где (т.е. в каких частях командного интерфейса, в каких формах) оказалась доступна созданная группа команд «**Печать**» и команда «**Печать расходной**».

Аналогичным образом можно создать глобальные команды, не связанные с определенными классами объектов, которые будут размещаться в ветке «Общие», в разделе «Общие команды».

11.6 Рабочий стол

В версии платформы 8.3 «Рабочий стол» носит название «Начальная страница». Если в приложении не будет реализовано деление на подсистемы, то данный раздел не будет существовать и его настройка будет невозможна.

На рабочем столе располагаются формы, доступ к которым осуществляется сразу при запуске приложения (своего рода автозагрузка). Вы наверняка заметили, что при загрузке

конфигурации не открываются подсистемы и то, что в них расположено, а открывается страница, называющаяся «Главное». На этой странице имеется возможность разместить те формы, доступ к которым осуществляется наиболее часто. Для каждой роли набор форм на рабочем столе может быть настроен свой. Следует отметить, что на рабочий стол нельзя поместить автоматически создаваемые системой формы, а только те, которые программист создал вручную. Создадим следующие формы:

– для документа «**Приходная накладная**» – форму списка «**Список приходных**»;

– для документа «**Расходная накладная**» – форму списка «**Список расходных**»;

– для регистра сведений «**Цены номенклатуры**» – форму списка «**Цены**»;

Для настройки рабочего стола, вызовем контекстное меню всей конфигурации и выберем пункт «Открыть рабочую область начальной страницы». В окне настройки (рис. 11.4) в первую очередь указывается шаблон рабочей области: в виде одной или двух (одинаковой ширины, либо в соотношении 2:1) колонок. Укажем шаблон в виде двух колонок в соотношении 2:1.

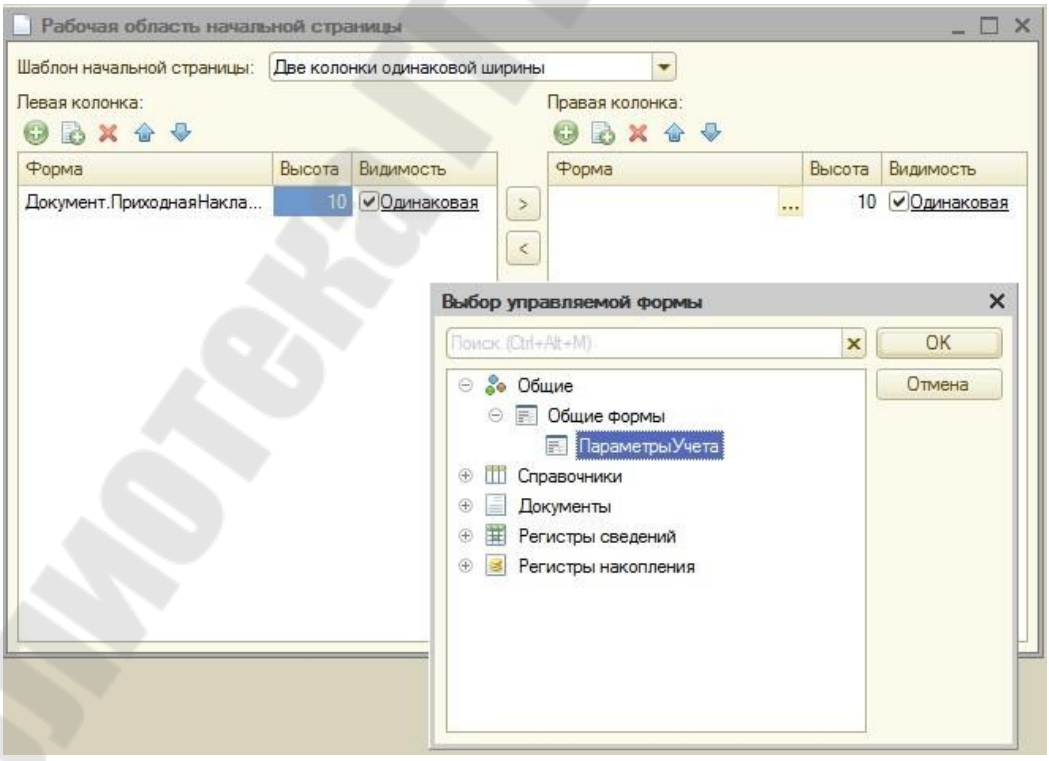


Рисунок 11.4 – Настройка начальной страницы

Каждая колонка может содержать произвольное (ограничено лишь здравым смыслом) число форм. В левую колонку добавим отображение форм «**Список приходных**» и «**Список расходных**», в правую – «**Цены**» и общую форму «**Параметры учета**». Понятно, что число форм в каждой колонке может быть различным, и это не обязательно будут формы списка – это может быть и форма отчета, и форма элемента справочника и т.д.; главное – чтобы форма была описана вручную.

Для каждой формы в колонке можно настроить высоту. По умолчанию стоит число «10». Если для всех форм колонок оставить значения по умолчанию, то все формы будут одинаковой высоты. В левой колонке так и сделаем, а в правой для формы «**Цены**» укажем высоту «14», для «**Параметров учета**» - «6». При этом в режиме исполнения размеры форм в рабочей области можно менять динамически.

Также для каждой формы можно настроить «Видимость» по ролям. Т.е. в зависимости от роли, форма будет или не будет отображаться в рабочей области. Одинаковая видимость говорит о том, что форма будет отображаться всегда. Таким образом, для каждой роли в конфигурации рабочую область можно настроить по-своему. Запустив систему в режиме отладки, увидим результат.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. 1С:Бухгалтерия 8.0 : учебная версия. - Москва : 1С-Паблишинг, 2006. - 562с. + 1 компакт-диск
2. 1С:Бухгалтерия 8.0 [Электронный ресурс]. - Москва : 1С-Константа, [2000]. - 1 электрон. опт. диск (CD-ROM) : зв., цв.. - (Компьютерный видеокурс)
3. Бухгалтерский учет. Настройка для Республики Беларусь : 1С: Предприятие 7.7 : руководство пользователя / Н. Шуляковская [и др.]. - Минск : ЮКОЛА-ИНФО, 2007. - 322 с. + 1 электр. опт. диск
4. Севостьянов А. Д. 1С:Бухгалтерия 8.0 : практика применения. - Изд. 3-е, перераб. и доп.. - Москва : 1С : Константа, 2007. - 220с
5. 1С: Предприятие 8.1. Версия для обучения программированию (комплект из 4 книг и 3 CD-ROM). – Москва : Питер, 2007. - 869 с.
6. Бартеньев, О. 1С: Предприятие. Программирование для всех / О. Бартеньев. - Москва : Диалог МИФИ, 2010. - 464 с.
7. Бартеньев, О. В. 1С: Предприятие: программирование для всех / О.В. Бартеньев. - Москва : Диалог-Мифи, 2005. - 464 с.
8. Белоусов, П.С. 1С: Предприятие: от 8.0 к 8.1 + CD / П.С. Белоусов, А.В. Островерх. - Москва : 1С-Паблишинг; СПб: Питер, 2008. - 286 с.
9. Бойко, Э. В. 1С: Предприятие 8.0. Универсальный самоучитель / Э.В. Бойко. - Москва : Омега-Л, 2011. - 232 с.
10. Бояркин, В.Э. 1С: Предприятие 8. Конвертация данных: обмен данными между прикладными решениями + 1 CD-ROM / В.Э. Бояркин, А.И. Филатов. - Москва : 1С: Паблишинг; СПб: Питер, 2008. - 180 с.
11. Габец, А.П. 1С: Предприятие 8.0. Простые примеры разработки / А.П. Габец, Д.И. Гончаров. - М.: 1С: Паблишинг, 2005. - 420 с.
12. Габец, А.П. 1С: Предприятие 8.1. Простые примеры разработки + 1 CD-ROM / А.П. Габец, Д.И. Гончаров. - М.: 1С: Паблишинг; СПб: Питер, 2008. - 383 с.
13. Кашаев, С. М. 1С: Предприятие 8.1. Разработка прикладных решений / С.М. Кашаев. - М.: Вильямс, 2009. - 368 с.

14. Кашаев, Сергей 1С: Предприятие 8.1. Учимся программировать на примерах (+CD-ROM) / Сергей Кашаев. - М.: БХВ-Петербург, 2009. - 400 с.

15. Постовалов, С.Н. 1С: Предприятие 7.7. Уроки программирования / С.Н. Постовалов, А.Ю. Постовалова - Москва : СПб: BHV, 2006. - 320 с.

16. Постовалов, С. Н. 1С: Предприятие 7.7. Уроки программирования / С.Н. Постовалов, А.Ю. Постовалова. - Москва : БХВ-Петербург, 2012. - 320 с.

17. Радченко, М.Г. 1С: Предприятие 8.1. Практическое пособие разработчика + CD / М.Г. Радченко. - Москва : ООО «1С-Паблишинг», СПб: Питер, 2007. – 512 с.

18. Радченко, М. Г. 1С:Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые приемы / М. Г. Радченко, Е. Ю. Хрусталева. - Москва: ООО «1С-Паблишинг», 2009. - 874 с: ил.

19. Радченко, М. Г. 1С:Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые приемы / М. Г. Радченко, Е. Ю. Хрусталева. - Москва : ООО «1С-Паблишинг», 2013. - 964 с.

СОДЕРЖАНИЕ

Титова Людмила Константиновна

РАЗРАБОТКА УЧЕТНЫХ ПРИЛОЖЕНИЙ В 1С:ПРЕДПРИЯТИЕ

Учебно-методическое пособие для студентов специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)», направление специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)» дневной и заочной форм обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 14.10.22.

Рег. № 51Е.
http://www.gstu.by