Разработка расширения конфигурации

1С:Предприятие 8.3

Версия 07.11.2018

январь 2020

Е. Ю. Хрусталева

**Расширения конфигураций  
Адаптация прикладных решений с сохранением поддержки в облаках и на земле  
Разработка в системе 1С:Предприятие 8.3  
Издание 2**

Электронная книга для публикации в информационной системе ИТС ПРОФ; ISBN 978-5-9677-2847-1.

Версия издания от 07.11.2018.

*Электронный аналог издания "Расширения конфигураций. Адаптация прикладных решений с сохранением поддержки в облаках и на земле. Разработка в системе 1С:Предприятие 8.3. Издание 2"  
(ISBN978-5-9677-2834-1, М.: ООО "1С-Паблишинг", 2018; артикул печатной книги по прайс-листу фирмы "1С": 4601546137920; по вопросам приобретения печатных изданий издательства "1С-Паблишинг" обращайтесь к партнеру "1С", обслуживающему вашу организацию, или к другим партнерам фирмы "1С".)*

Книга адресована специалистам по внедрению прикладных решений системы «1С:Предприятие 8», а также разработчикам, адаптирующим типовые прикладные решения к требованиям заказчика. Она рассказывает о новой технологии – расширениях конфигурации.

Расширения конфигурации позволяют дорабатывать функциональность прикладных решений, не внося изменений в сами прикладные решения.

В результате прикладное решение остается на полной поддержке поставщика, что значительно упрощает его дальнейшее сопровождение.

Также при работе в модели сервиса, например, в 1С:Фреш (1cfresh.com), расширения конфигурации позволяют дорабатывать функциональность прикладного решения только для пользователей одной организации. При этом другие организации, работающие в этом же сервисе с этим же прикладным решением, этих доработок не увидят.  
Второе издание книги описывает возможности расширений конфигурации, реализованные в платформе версии 8.3.13, в том числе расширение прикладных данных.  
В первой и второй главах рассматривается общая архитектура расширений, и на большом количестве простых примеров показываются возможности доработки тех или иных элементов прикладного решения. Например, добавление учета товаров по характеристикам или адаптация внешнего вида прикладного решения к корпоративному стилю.  
В третьей главе книги показан пример разработки расширения для ведения управленческого учета на предприятии. Это учебное расширение, демонстрирующее общий подход к доработке прикладной функциональности.  
В четвертой главе показан полный процесс разработки «настоящего» расширения для тарификации, которое можно использовать совместно с типовой конфигурацией «1C:Зарплата и кадры государственного учреждения».  
Книга содержит большое количество рисунков и примеров кода на встроенном языке, снабженных подробными комментариями. Для создания демонстрационных примеров использована версия 8.3.13.1472 платформы «1С:Предприятие 8».  
Расширения, разрабатываемые в книге, опубликованы на портале 1С:ИТС. Вы можете скачать их по адресу http://its.1c.ru/download/book\_demo/.

© ООО «1С-Паблишинг», 2018  
© Оформление. ООО «1С-Паблишинг», 2018

Все права защищены.  
Материалы предназначены для пользователей 1С:Предприятия, имеющих действующий договор 1С:ИТС.  
Запрещено тиражирование, распространение материалов, предоставление доступа по сети к материалам без письменного разрешения правообладателей.  
Разрешено копирование фрагментов программного кода для использования в разрабатываемых прикладных решениях.

**Фирма "1С"**  
123056, Москва, а/я 64, Селезневская ул., 21.  
Тел.: (495) 737-92-57, факс: (495) 681-44-07.  
[1c@1c.ru](mailto:1c@1c.ru), <http://www.1c.ru/>  
**Издательство ООО "1С-Паблишинг"**  
127473, Москва, ул. Достоевского, 21/1, строение 1.  
Тел.: (495) 681-02-21, факс: (495) 681-44-07.  
[publishing@1c.ru](mailto:publishing@1c.ru), <http://books.1c.ru/>

Оглавление

[Дополнительные материалы 1](#_Toc31277949)

[Глава 1. Общее описание 1](#_Toc31277950)

[Назначение 1](#_Toc31277951)

[Типичные сценарии использования 3](#_Toc31277952)

[Устройство 3](#_Toc31277953)

[Создание и отладка 5](#_Toc31277954)

[Префикс 6](#_Toc31277955)

[Назначение 6](#_Toc31277956)

[Область действия 6](#_Toc31277957)

[Работа с расширением 7](#_Toc31277958)

[Применение расширения у заказчика 12](#_Toc31277959)

[Расширение подключается к информационной базе при старте сеанса, непосредственно перед вызовом события ***УстановкаПараметровСеанса***. Область применения 13](#_Toc31277960)

[Деактивация 14](#_Toc31277961)

[Удаление 14](#_Toc31277962)

[Проверка возможности применения 15](#_Toc31277963)

[Глава 2. Функциональность 18](#_Toc31277964)

[Формы 18](#_Toc31277965)

[Как доработать внешний вид формы 18](#_Toc31277966)

[Как добавить объекту собственную основную форму 28](#_Toc31277967)

[Как создать обработчик события для элемента формы 32](#_Toc31277968)

[Как перехватить обработчик события и обрамить его своими алгоритмами 38](#_Toc31277969)

[Как добавить в форму команду и создать ее обработчик 45](#_Toc31277970)

[Отчеты и обработки 53](#_Toc31277971)

[Как создать собственный отчет 53](#_Toc31277972)

[Как доработать отчет 56](#_Toc31277973)

[Как доработать макет табличного документа 60](#_Toc31277974)

[Что произойдет, если макет табличного документа изменится после обновления конфигурации 62](#_Toc31277975)

[Что произойдет, если к конфигурации будет применено еще одно расширение, изменяющее тот же макет табличного документа 64](#_Toc31277976)

[Зачем заимствовать предопределенные данные 65](#_Toc31277977)

[Что произойдет, если свойства предопределенного элемента будут изменены в конфигурации 67](#_Toc31277978)

[Модули 67](#_Toc31277979)

[Общая информация 67](#_Toc31277980)

[Аннотация «&Перед» 68](#_Toc31277981)

[Аннотация «&После» 68](#_Toc31277982)

[Комбинация аннотаций «&Перед» и «&После» 69](#_Toc31277983)

[Аннотация «&Вместо» 69](#_Toc31277984)

[Аннотация «&Вместо» совместно с «ПродолжитьВызов()» 69](#_Toc31277985)

[Стратегия выбора типа расширяющих методов 70](#_Toc31277986)

[Доработка модулей в расширении 71](#_Toc31277987)

[Взаимодействие нескольких расширений 71](#_Toc31277988)

[Работа расширения в безопасном режиме 73](#_Toc31277989)

[Модуль объекта. Как создать обработчик события 74](#_Toc31277990)

[Как создать собственный общий модуль 79](#_Toc31277991)

[Как доработать функцию общего модуля 80](#_Toc31277992)

[Модуль формы. Перехват процедур с помощью аннотаций 82](#_Toc31277993)

[Роли 86](#_Toc31277994)

[Как доработать роль 86](#_Toc31277995)

[Как создать собственную роль и добавить ее пользователю 88](#_Toc31277996)

[Подсистемы 90](#_Toc31277997)

[Как доработать подсистему 90](#_Toc31277998)

[Как создать собственную подсистему 92](#_Toc31277999)

[Как доработать начальную страницу 93](#_Toc31278000)

[Как доработать интерфейс основного раздела 96](#_Toc31278001)

[Стиль оформления прикладного решения 99](#_Toc31278002)

[Как создать собственный стиль оформления 99](#_Toc31278003)

[Как доработать стиль оформления 102](#_Toc31278004)

[Брендирование расширений 104](#_Toc31278005)

[Расширение данных 104](#_Toc31278006)

[Как реализовать учет по характеристикам в расширении 105](#_Toc31278007)

[Как сформировать движения собственных и заимствованных документов по собственным и заимствованным регистрам накопления 109](#_Toc31278008)

[Глава 3. Расширение для ведения управленческого учета на предприятии 113](#_Toc31278009)

[Базовые понятия 113](#_Toc31278010)

[Порядок наших действий 113](#_Toc31278011)

[Расширение для управленческого учета 114](#_Toc31278012)

[План счетов 114](#_Toc31278013)

[Регистр бухгалтерии 117](#_Toc31278014)

[Расширение модулей документов-регистраторов 119](#_Toc31278015)

[Формирование движений в регистре бухгалтерии «Управленческий» 125](#_Toc31278016)

[Отчеты по управленческому учету 126](#_Toc31278017)

[Глава 4. Расширение для тарификации сотрудников государственных медицинских учреждений 130](#_Toc31278018)

[Постановка задачи 131](#_Toc31278019)

[Базовые понятия тарификации и алгоритм расчета ФОТ 131](#_Toc31278020)

[Порядок наших действий 132](#_Toc31278021)

[Расширяемая конфигурация для ведения кадрового учета 132](#_Toc31278022)

[Расширение для выполнения тарификации 136](#_Toc31278023)

[Доработка исходной структуры кадровых данных 136](#_Toc31278024)

[Разработка документа для тарификации сотрудников 140](#_Toc31278025)

[Ввод тарификационных данных в режиме «1С:Предприятие» 149](#_Toc31278026)

[Исходные данные для тарификации 149](#_Toc31278027)

[Тестирование документа для тарификации сотрудников 154](#_Toc31278028)

[Отчет «Тарификация сотрудников отделения» 156](#_Toc31278029)

[Запрос 156](#_Toc31278030)

[Вычисляемые поля 163](#_Toc31278031)

[Ресурсы 165](#_Toc31278032)

[Параметры 165](#_Toc31278033)

[Поля набора данных 165](#_Toc31278034)

[Настройки 165](#_Toc31278035)

[Вызов отчета из формы документа «Расш1\_ТарификацияСотрудников» 167](#_Toc31278036)

[Форма отчета «Расш1\_ТарификацияСотрудниковОтделения» 168](#_Toc31278037)

[Отчет «Тарификационные списки сотрудников учреждений ФМБА» 169](#_Toc31278038)

[Запрос 170](#_Toc31278039)

[Вычисляемые поля и ресурсы 171](#_Toc31278040)

[Параметры 172](#_Toc31278041)

[Поля набора данных 172](#_Toc31278042)

[Настройки 172](#_Toc31278043)

[Доработка расширения для похожих внедрений 180](#_Toc31278044)

# Дополнительные материалы

Приложение к книге Расширения конфигураций. Адаптация прикладных решений с сохранением поддержки в облаках и на земле. Разработка в системе 1С:Предприятие 8.3. Издание 2

Приложение содержит демонстрационные конфигурации, которые можно установить на коммерческую или учебную версию 1С:Предприятия, и другие материалы.

[Скачайте дополнительные материалы](http://fserver.1c.ru/its/files/demo/books/pubextensions/extensions2.zip) (498 Kб), раскройте архив.

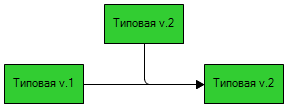
Учебную версию платформы "1С:Предприятие 8.3" можно бесплатно скачать [здесь](http://online.1c.ru/catalog/free/learning.php).

# Глава 1. Общее описание

## Назначение

На практике часто встречается ситуация, когда выпущенное типовое прикладное решение при внедрении у заказчика дорабатывается под его индивидуальные особенности и пожелания. Например, некоторые заказчики хотели бы, чтобы это прикладное решение имело еще пару дополнительных отчетов или чтобы внешний вид форм документов или отчетов был немного другим – таким, к которому они привыкли.

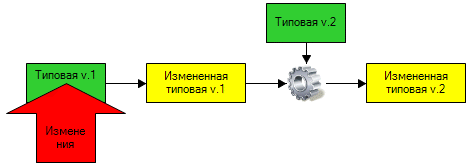
В платформе есть все инструменты для этого, но после изменения конфигурации ее поддержка и обновление усложняются. Раньше это выглядело следующим образом. До тех пор, пока в конфигурацию не вносились никакие изменения, она находилась на полной поддержке поставщика. Поэтому, когда поставщик выпускал новые (улучшенные) версии этой конфигурации, обновление старой версии конфигурации на новую выполнялось полностью автоматически. Таким образом, выполнять обновление мог любой пользователь, для этого не надо было обладать навыками конфигурирования. Это было быстро, легко и удобно. Данный процесс представлен на следующей схеме (рис. 1.1).



**Рис. 1.1.**Автоматическое обновление конфигурации на новую версию

Если требовалось изменить конфигурацию, надо было сначала снять ее с полной поддержки. После внесения изменений автоматическое обновление конфигурации на новую версию становилось невозможным.

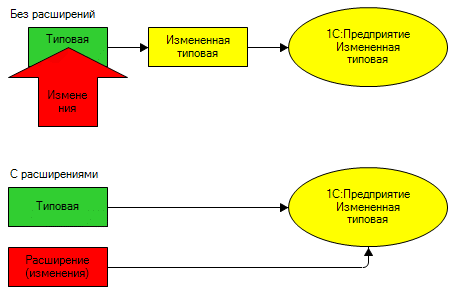
Для обновления измененной конфигурации уже требовалось участие квалифицированного специалиста. При этом чем больше и серьезнее были внесенные изменения, тем больше времени и усилий могло понадобиться специалисту, выполнявшему обновление. В случае пересечения изменений в обеих конфигурациях от специалиста могло потребоваться очень хорошее знание как самой типовой конфигурации, так и внесенных доработок. Этот процесс можно представить на следующей схеме (рис. 1.2).



**Рис. 1.2.**Обновление измененной конфигурации на новую версию

Чтобы упростить поддержку измененных конфигураций, в платформе «1С:Предприятие» был разработан (и продолжает развиваться) *механизм расширений*. С помощью расширений можно дорабатывать конфигурацию, не внося в нее изменений (не снимая с поддержки). Использование расширений существенно облегчает сопровождение типовых решений, которые нужно адаптировать к потребностям конкретного заказчика.

При этом все изменения выполняются в расширении, а не в самой конфигурации. Затем в режиме *1С:Предприятие* расширение применяется к конфигурации, и оно автоматически объединяется с ней. В результате заказчик работает с типовым решением, содержащим все необходимые ему доработки. Этот процесс можно представить на следующей схеме (рис. 1.3).

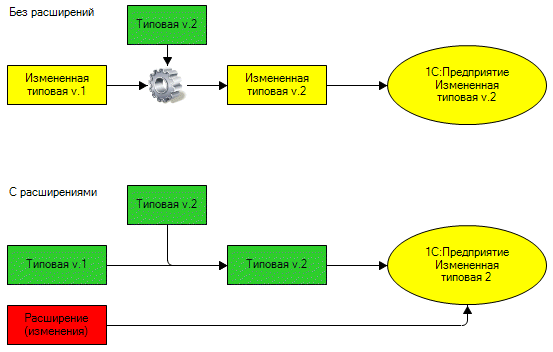


**Рис. 1.3.**Объединение конфигурации с расширением

Для конечного пользователя все выглядит так, как если бы изменения вносились по-прежнему в саму конфигурацию.

То есть теперь для доработки конфигурации не нужно снимать ее с поддержки и затем вручную выполнять обновление измененной конфигурации, как это было раньше. Обновление выполняется автоматически, поскольку сама конфигурация не изменяется и остается на полной поддержке поставщика.

Во время запуска обновленного прикладного решения платформа снова автоматически объединит обновленную конфигурацию с расширением. И вся функциональность, созданная по желанию заказчика, продолжит работать. Этот процесс можно представить на следующей схеме (рис. 1.4).



**Рис. 1.4.**Объединение обновленной конфигурации с расширением

## Типичные сценарии использования

Прежде всего, расширения очень удобно применять при работе в модели сервиса. С одной стороны, часто требуется, чтобы какая-то функциональность, добавленная в прикладное решение (например, новый отчет), была доступна только той организации, которая заказала данную доработку.

Раньше сделать это было сложно, так как для всех организаций в сервисе используется одна и та же конфигурация, поэтому в случае ее изменения новый отчет появляется также и для остальных организаций, работающих с этим же решением. Хотя они об этом не просили.

Теперь на помощь приходят расширения. При работе в модели сервиса расширение можно применить для конкретной области данных, т. е. для пользователей конкретной организации. В этом случае оно хранится в информационной базе в разрезе текущих значений разделителей. Набор этих значений определяет ту организацию, для которой это расширение будет применено.

Таким образом, при использовании расширений, применяемых для отдельных областей данных, изменения, добавленные для одной организации, никак не отразятся на остальных организациях, работающих с этим же решением в сервисе. Потому что эти изменения будут находиться в расширении, которое сможет применить конкретная организация, заказавшая данную доработку. В результате эта организация будет работать с доработанной по ее желанию конфигурацией, в то время как остальные организации по-прежнему будут использовать исходную конфигурацию.

С другой стороны, иногда бывает нужно, чтобы изменения, вносимые в конфигурацию с помощью расширения, были доступны сразу всем пользователям сервиса. Например, для оперативного исправления ошибок конфигурации. Такой вариант тоже возможен. Для этого при подключении расширения к конфигурации, работающей в режиме разделения данных надо указать, что оно будет действовать сразу на всю информационную базу.

Другой пример применения расширений описан в предыдущем разделе. Если конфигурацию требуется доработать под конкретного заказчика, то все эти доработки лучше выполнять в расширении. Тогда конфигурация останется на полной поддержке, что значительно упростит ее дальнейшее сопровождение.

Кроме того, преимущество расширений заключается в том, что доработки, внесенные с помощью расширения в одно прикладное решение, можно потом использовать еще раз в другом похожем прикладном решении. Для этого нужно всего лишь подключить к нему уже разработанное расширение. А раньше бы для этого пришлось в другом прикладном решении еще раз повторять те же изменения, что и в первом.

Ну и, наконец, бывают ситуации, когда многим пользователям типового прикладного решения нужна одна и та же функциональность, например тарификация для сотрудников государственных медицинских учреждений. Но вносить ее в типовую конфигурацию не имеет смысла, потому что алгоритм и показатели тарификации сильно разнятся по регионам и зависят от местных инструкций.

Чтобы не создавать несколько типовых решений под требования каждого региона, лучше всего создать несколько расширений для одного типового решения и применять соответствующее расширение в том регионе, для которого оно предназначено.

Пример разработки расширения, выполняющего тарификацию сотрудников медицинского учреждения, будет рассмотрен в четвертой главе [«Расширение для тарификации сотрудников государственных медицинских учреждений»](#_Глава_4._Расширение).

## Устройство

Основой расширения являются объекты, *заимствованные* из конфигурации. Чтобы доработать какой-то объект конфигурации, его нужно сначала заимствовать (то есть специальным образом скопировать из конфигурации в расширение) и затем, уже в расширении, выполнить необходимые изменения.

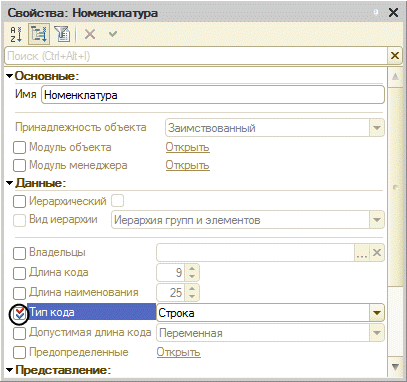
Кроме того, в расширение можно добавлять *собственные* объекты, созданные на основе прототипов, существующих внутри платформы. Работа с собственными объектами в расширении ничем не отличается от этого процесса в конфигурации.

Заимствование объектов из конфигурации выполняется с двумя целями.

Во-первых, чтобы расширение можно было применить к конфигурации. Заимствовать нужно те объекты, которые жизненно необходимы для работы расширения – когда они необходимы для визуального конструирования (форм, запросов и т. п.) или когда они используются во встроенном языке или языке запросов. Тогда, если при применении они не будут обнаружены в конфигурации, расширение не применится. При этом вы узнаете об ошибке сразу, при старте прикладного решения, а не во время исполнения. Такой подход является методологически правильным.

Если этого не сделать, то возможна следующая ситуация. При обновлении конфигурации из нее может исчезнуть, например, реквизит объекта, используемый в каком-то запросе. Тогда расширение применится к конфигурации без сообщения об ошибке и будет работать до того места, где выполняется запрос. Вследствие отсутствия нужных данных для запроса случится ошибка исполнения, причем самое неприятное, что это может произойти после долгих и сложных действий пользователя. И в результате он потеряет целый кусок своей работы.

Есть возможность и более тонкого контроля, когда можно проконтролировать не просто наличие, например, справочника *Номенклатура*, но и то, что его код должен иметь тип *Строка*. В этом случае разработчик может установить у этого свойства в расширении флажок https://its.1c.ua/db/content/pubextensions/src/pic01_01.png?_=1580137101 (с подсказкой «Проверять значение при подключении расширения»). Такое свойство становится *контролируемым* (рис. 1.5).

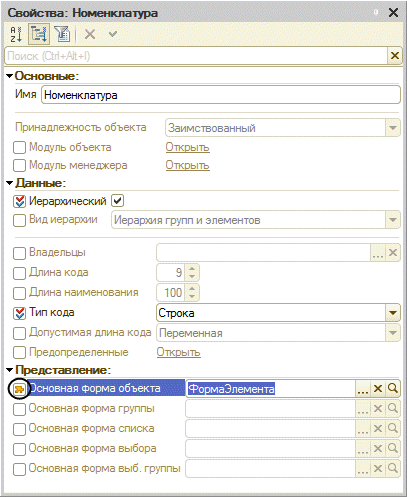


**Рис. 1.5.**Флажок контролируемого свойства в расширении

Это значит, что для применения расширения к конфигурации необходимо, чтобы в конфигурации тип реквизита был таким же, как и в расширении. Иначе расширение не будет применено.

Если для работы расширения не важно, какое значение имеет свойство заимствованного объекта, то можно снять у этого свойства флажок https://its.1c.ua/db/content/pubextensions/src/pic01_01.png?_=1580137101, и тогда значение свойства не будет контролироваться при применении расширения.

Во-вторых, объекты заимствуются из конфигурации для того, чтобы их можно было изменять в расширении. Те свойства, которые можно модифицировать, имеют возможность установки флажка https://its.1c.ua/db/content/pubextensions/src/pic01_02.png?_=1580137101 (с подсказкой «Модифицировать значение при подключении расширения»). Свойства с таким флажком являются *модифицируемыми* (рис. 1.6).



**Рис. 1.6.**Флажок модифицируемого свойства в расширении

Значение такого свойства в результате применения расширения к конфигурации будет получаться из расширения. Если флажок https://its.1c.ua/db/content/pubextensions/src/pic01_02.png?_=1580137101 у свойства снят, значение этого свойства будет получаться из конфигурации.

Заимствованный и доработанный в расширении объект становится *расширяющим* объектом. Свойство заимствованного объекта не может быть одновременно и контролируемым, и модифицируемым.

На момент написания книги (по версии платформы 8.3.13.1472) механизм расширений значительно продвинулся и сейчас предоставляет весьма широкие и вариативные возможности по расширению конфигураций. В расширении можно дорабатывать почти все объекты (в т. ч. справочники и документы), добавлять собственные объекты, в том числе прикладные объекты, содержащие данные. Но имеются некоторые ограничения и особенности.

В расширении конфигурации не поддерживается создание следующих собственных объектов:

* параметры сеанса,
* общие реквизиты,
* критерии отбора,
* подписки на события,
* регламентные задания,
* функциональные опции и параметры функциональных опций,
* определяемые типы,
* хранилища настроек,
* языки,
* константы,
* журналы документов,
* бизнес-процессы и задачи,
* внешние источники данных.

Не поддерживается расширение следующих объектов:

* Нельзя создавать подчиненные подсистемы.
* Состав значений перечислений.
* Не поддерживается добавление реквизитов и табличных частей для следующих объектов:
  + план видов характеристик;
  + план счетов;
  + план видов расчетов.
* Не поддерживается изменение структуры регистров всех видов. Поддерживается только расширение состава регистраторов.

**ПРИМЕЧАНИЕ**

*Состав расширяемых и собственных объектов может со временем изменяться. Подробнее см. документацию «1С:Предприятие 8.3.13. Глава 33.1.3. Расширение конфигурации. Общая информация. Особенности и ограничения».*

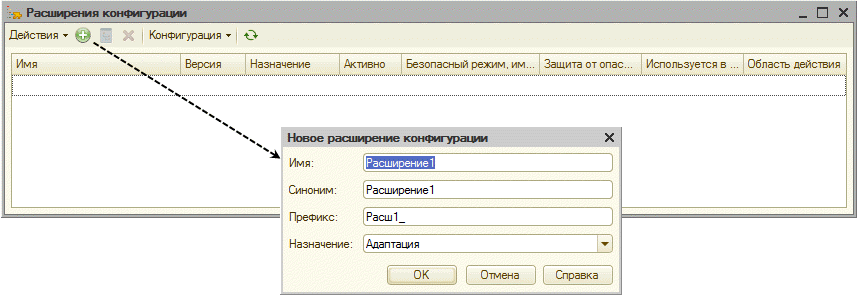
При разработке расширения нужно учитывать, что конфигурация в любой момент может быть изменена, например в результате обновления. Также следует учитывать тот факт, что в информационную базу может быть загружено сразу несколько расширений и автор каждого расширения (в общем случае) не знает, как функционирует другое расширение.

Во второй главе [«Функциональность»](#_Глава_2._Функциональность) мы рассмотрим примеры доработки различных частей конфигурации. И в конце каждого примера, по мере необходимости, покажем, что будет, если изменится конфигурация или появится другое расширение.

## Создание и отладка

Расширение очень похоже на обычную конфигурацию. Оно также представляется в виде дерева объектов. Для работы с расширением используются те же приемы, что и для работы с обычной конфигурацией.

Чтобы создать расширение конфигурации, нужно вызвать команду главного меню *Конфигурация – Расширения конфигурации* и в открывшемся списке расширений нажать кнопку *Добавить*. Затем необходимо указать имя расширения, его синоним, префикс и назначение расширения (рис. 1.7).



**Рис. 1.7.**Создание расширения

### Префикс

Конфигурация и расширение существуют в едином пространстве имен. Поэтому, чтобы не возникало конфликта, имена всех объектов, процедур и обработчиков, созданных в расширении, автоматически будут начинаться с префикса, заданного при создании расширения в поле *Префикс*. В дальнейшем этот префикс можно изменить в палитре свойств расширения, в свойстве *Префикс имен* (см. рис. 1.10).

### Назначение

Назначение расширения используется для того, чтобы задать порядок применения расширений к конфигурации (в случае если в информационную базу прикладного решения загружено несколько расширений). Назначение расширения может принимать значения *Исправление*, *Адаптация* и *Дополнение*.

Расширение с назначением *Исправление* предназначено для исправления ошибок в конфигурации. Поэтому оно применяется к конфигурации первым. Затем применяется расширение с назначением *Адаптация*. Оно содержит доработки конфигурации при внедрении у конкретного заказчика. И последним применяется расширение с назначением *Дополнение*. Оно содержит различные дополнительные сервисы, предназначенные для конфигурации (например, набор дополнительных отчетов).

Предполагается, что расширения с одинаковым назначением не должны «пересекаться» по функционалу и «мешать друг другу».

Назначение расширения сохраняется в свойстве *Назначение расширения конфигурации* и может быть изменено позже из палитры свойств расширения (см. рис. 1.10). По умолчанию у вновь создаваемого расширения устанавливается назначение *Адаптация*.

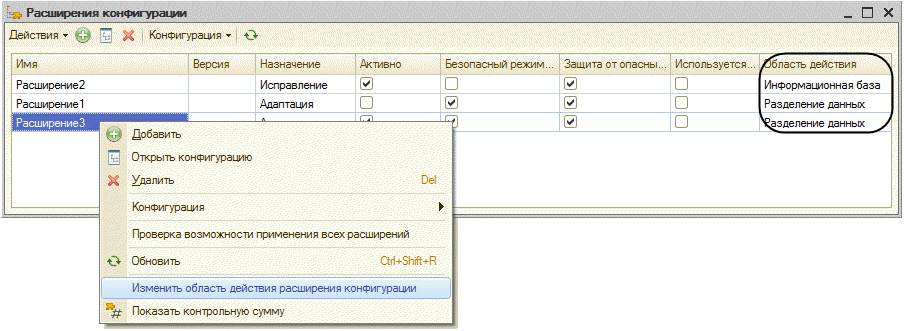
Следует учитывать, что устанавливать назначение расширения можно начиная с версии платформы 8.3.10. Применение расширений в порядке, соответствующем их назначению, работает только при отключенном режиме совместимости (свойства расширения *Режим совместимости расширения конфигурации*, *Режим совместимости* и свойство *Режим совместимости* конфигурации должны быть установлены в значение *Не использовать*).

### Область действия

Расширение конфигурации может быть применено как для всей информационной базы, так и для конкретной области данных (при условии, что расширение подключается к разделенной информационной базе). Область применения расширения отражается в колонке *Область действия* в списке расширений конфигурации (см. рис. 1.8).

Область действия расширения может быть изменена для конфигураций, предназначенных для работы в режиме разделения данных. То есть если конфигурация содержит разделители, которые могут разделять расширения (у общих реквизитов, разделяющих данные, свойство*Разделение расширений конфигурации* принимает значение *Разделять*), то область действия расширения может быть установлена как *Информационная база* или как *Разделение данных*.

При создании расширений у таких конфигураций область действия стандартно устанавливается как *Разделение данных*. Но это значение можно изменить командой *Изменить область действия расширения конфигурации* из контекстного меню выделенного расширения, за исключением случая, когда расширение расширяет данные. Для такого расширения сменить область действия не представляется возможным (рис. 1.8).



**Рис. 1.8.**Область действия расширения

Если конфигурация не содержит разделителей, которые могут разделять расширения, то при создании расширений у таких конфигураций область действия стандартно устанавливается как *Информационная база* и выбор другого значения недоступен. То же самое относится и к конфигурациям, в которых разделители, разделяющие расширения, есть, но они условно отключены.

Область действия расширения оказывает влияние на порядок подключения расширений к конфигурации – в рамках каждого назначения (*Исправление*, *Адаптация*, *Дополнение*) вначале подключаются расширения с областью действия *Информационная база*, а затем – *Разделение данных*. Если существует несколько расширений с одинаковым назначением и областью действия – порядок подключения таких расширений в общем случае не определен.

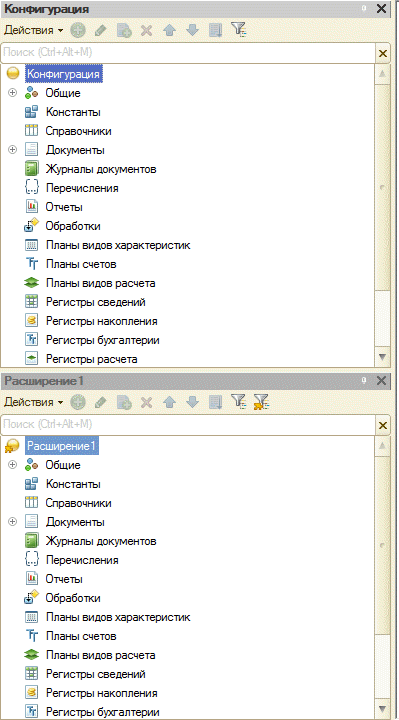
### Работа с расширением

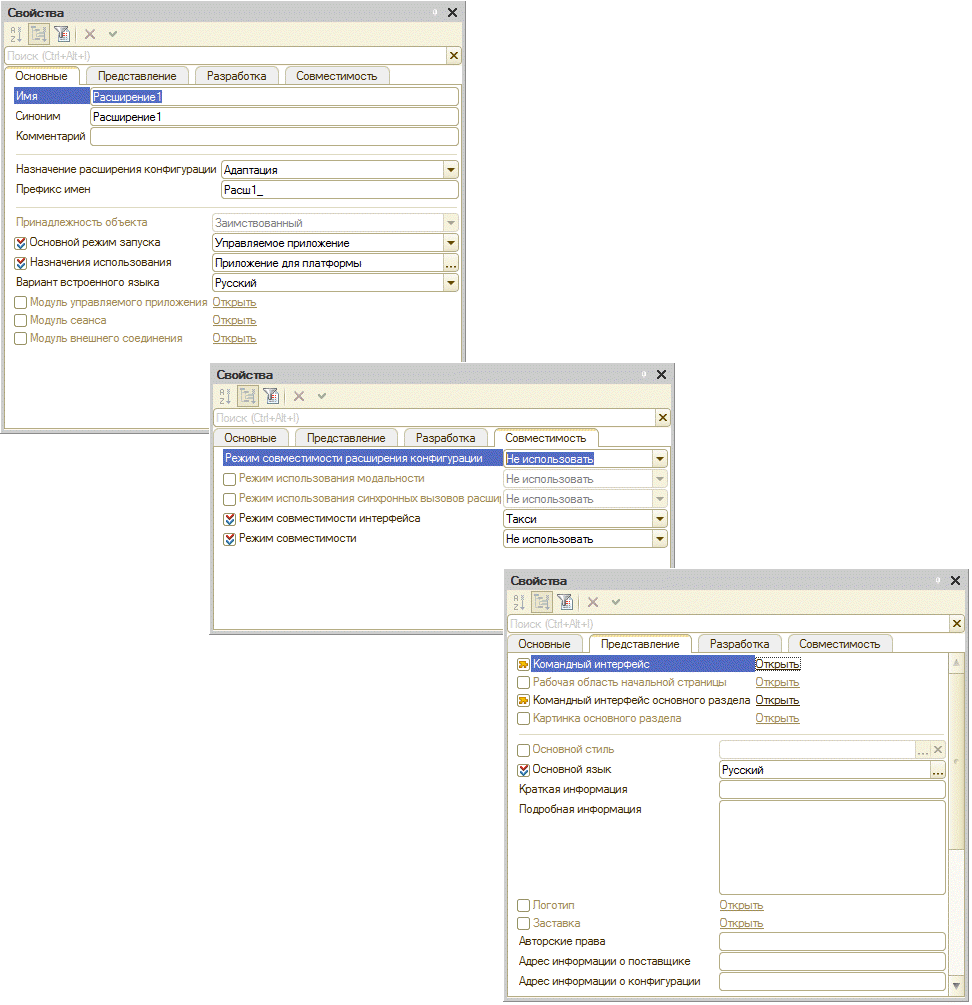
После создания расширение появится в списке расширений. Чтобы его открыть, нужно дважды щелкнуть на нем мышью или нажать кнопку *Открыть конфигурацию* в командной панели окна. В результате под окном дерева объектов конфигурации появится окно со структурой дерева расширения. В дальнейшем в нем будут появляться заимствованные и собственные объекты (рис. 1.9).

**ПРИМЕЧАНИЕ**

*Для удобства можно соединить окна конфигурации и расширения в одно окно с закладками.*

При создании нового расширения из конфигурации автоматически заимствуются следующие свойства: *Основной режим запуска*, *Назначение использования*, *Основной язык*, *Режим совместимости интерфейса* и *Режим совместимости*. Эти же свойства помечаются как контролируемые (рис. 1.10).

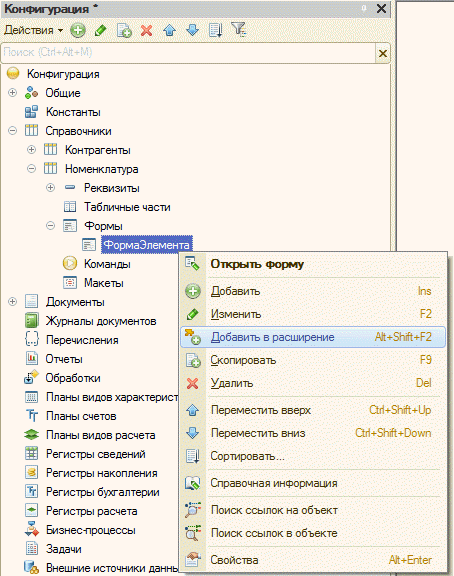
**Рис. 1.9.**Окна конфигурации и расширения



**Рис. 1.10.**Свойства расширения конфигурации

Также по умолчанию свойства *Командный интерфейс* и *Командный интерфейс основного раздела* помечаются как модифицируемые.

Чтобы позаимствовать объект, нужно выполнить команду его контекстного меню *Добавить в расширение* (рис. 1.11).



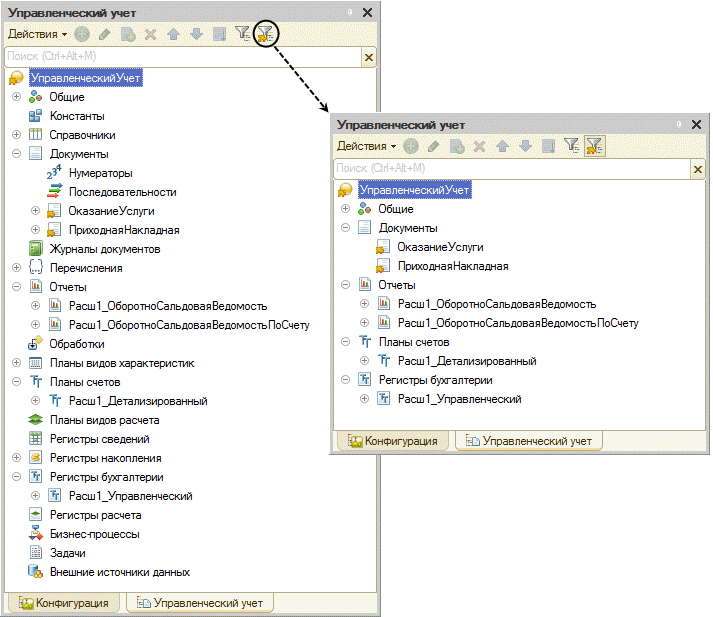
**Рис. 1.11.**Заимствование объекта конфигурации

Можно также создать расширение одновременно с заимствованием объекта. Если у конфигурации еще нет ни одного расширения, то после выполнения команды *Добавить в расширение* откроется пустой список выбора расширений. В нем нужно нажать кнопку *Добавить* и указать имя создаваемого расширения, синоним, префикс имен и назначение расширения (см. рис. 1.7).

После этого созданное расширение появится в списке выбора расширений. Нужно нажать кнопку *Выбрать*, и окно расширения с заимствованным объектом откроется под окном конфигурации.

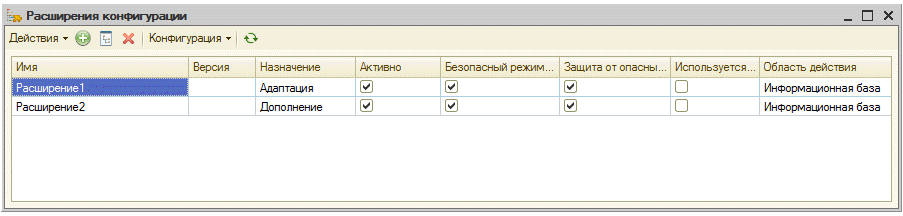
Все заимствованные объекты в структуре объектов расширения помечаются пиктограммой https://its.1c.ua/db/content/pubextensions/src/pic01_03.png?_=1580137101 (см. рис. 1.12). Следует учитывать, что заимствованные объекты автоматически расставляются в порядке заимствования, а не в том порядке, в котором они находятся в конфигурации. Если это важно для работы алгоритмов расширения, то можно отсортировать их в расширении так же, как и в конфигурации, например по алфавиту.

Чтобы было проще ориентироваться среди объектов расширения, можно отфильтровать их кнопкой *Измененные и добавленные в расширении* и увидеть только те объекты, которые вы сами добавили в расширение или заимствовали и доработали в расширении (рис. 1.12).



**Рис. 1.12.**Отображение только собственных и заимствованных объектов

В конфигураторе можно создавать и разрабатывать несколько расширений для одной конфигурации. Например, конфигурация может иметь несколько расширений для разных организаций-заказчиков. Все они сохраняются в информационной базе и отображаются в списке расширений (рис. 1.13).



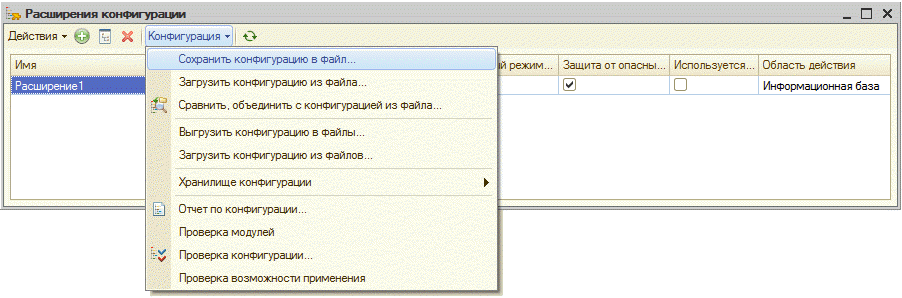
**Рис. 1.13.**Список расширений

Из этого списка можно открыть каждое расширение в конфигураторе. При этом нужно учитывать, что если открыто только одно расширение, то объекты конфигурации автоматически будут заимствоваться только в это открытое расширение. Если же открыто несколько расширений (или не открыто ни одного), то платформа выведет список выбора расширений, в котором нужно будет указать, в какое из расширений заимствовать объект.

Для отладки расширения из конфигуратора никаких особенных действий производить не надо. При запуске отладки все изменения в расширении сохраняются в информационной базе, затем расширение автоматически применяется к конфигурации и работает.

Чтобы увидеть, применено ли расширение (и вообще какие расширения применены), можно вызвать диалог *О программе* в режиме *1С:Предприятие*. Для получения более подробной информации о работе расширения нужно нажать гиперссылку *Информация для технической поддержки*.

После того как расширение отлажено и проверено, его можно сохранить в файл с расширением «.cfe», который понадобится затем для применения этого расширения к прикладному решению заказчика. Для этого в списке расширений нужно выделить нужное расширение и выполнить команду *Конфигурация – Сохранить конфигурацию в файл* в командной панели окна или контекстного меню списка (рис. 1.14).



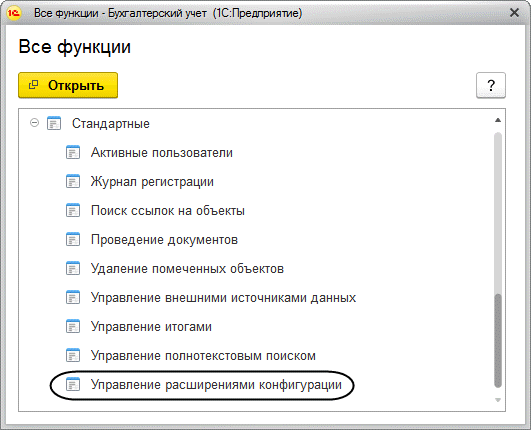
**Рис. 1.14.**Сохранение расширения в файл

## Применение расширения у заказчика

Применить расширение у заказчика можно в конфигураторе или в режиме *1С:Предприятие*.

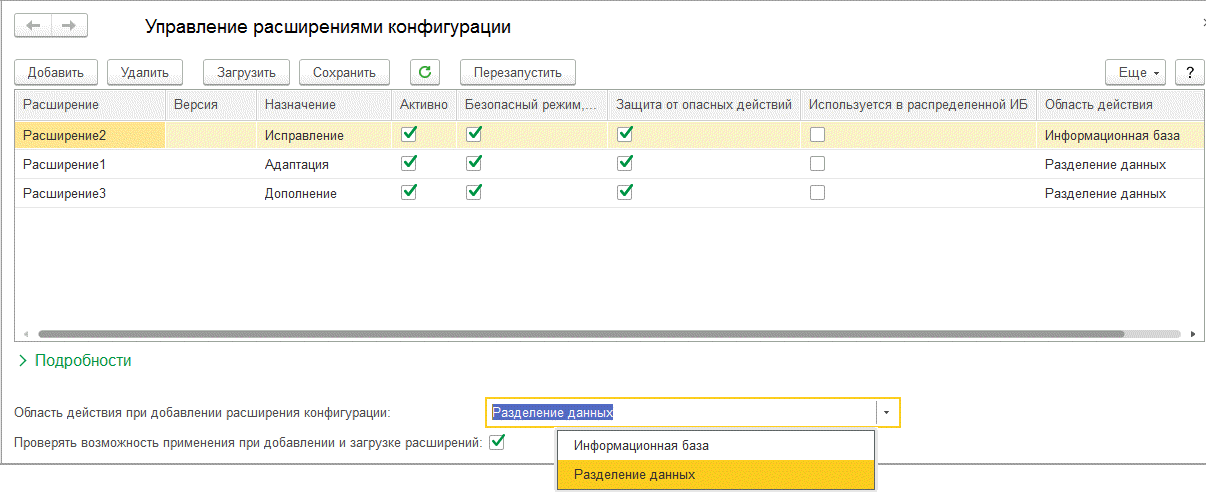
В конфигураторе это может сделать сам разработчик или специалист заказчика, имеющий доступ к конфигуратору. Для этого нужно открыть информационную базу прикладного решения в конфигураторе, создать в ней расширение и загрузить его из файла командой *Конфигурация – Загрузить конфигурацию из файла* в списке расширений (см. рис. 1.14). И после этого обновить конфигурацию базы данных. Расширение будет сохранено в информационной базе и применится к прикладному решению при старте клиентского сеанса.

В режиме *1С:Предприятие* применить расширение может пользователь заказчика, обладающий правом *Администрирование расширений конфигурации*. В этом случае разработчик может просто отправить файл с расширением заказчику по электронной почте. После этого пользователь может самостоятельно загрузить расширение в свою информационную базу, не заходя в конфигуратор. Для этого ему нужно в режиме *1С:Предприятие* вызвать стандартную функцию управления расширениями с помощью команды *Все функции – Стандартные – Управление расширениями конфигурации* (рис. 1.15).



**Рис. 1.15.**Управление расширениями конфигурации в режиме «1С:Предприятие»

В открывшемся диалоге сначала нужно загрузить расширение из файла кнопкой *Загрузить* и затем перезапустить сеанс кнопкой *Перезапустить* (рис. 1.16).



**Рис. 1.16.**Управление расширениями конфигурации в режиме «1С:Предприятие»

Чтобы расширение заработало у остальных пользователей, которые в данный момент работают с информационной базой, им также нужно перезапустить свои рабочие сеансы.

Расширение подключается к информационной базе при старте сеанса, непосредственно перед вызовом события ***УстановкаПараметровСеанса***. Область применения

### Область применения

Если расширение подключается к разделенной информационной базе, например, работающей в режиме сервиса, то для такого расширения имеется возможность указать область действия расширения: вся информационная база или конкретная область данных. Область применения расширения указывается в нижней части формы, в поле *Область действия при добавлении расширения конфигурации* (см. рис. 1.16).

Если область действия расширения – *Разделение данных*, то расширение будет применяться только в том случае, если набор его разделителей соответствует текущим значениям разделителей для данного сеанса. Таким образом, расширение будет применено не для всех пользователей, работающих с конфигурацией, а только для пользователей конкретной организации, работающей в сервисе (области данных).

Если область действия расширения – *Информационная база*, то расширение будет применяться, как и в случае с неразделенной базой, для всех пользователей сервиса.

Выбор области применения расширения в поле *Область действия при добавлении расширения конфигурации* доступен для изменения только в случае, если в информационной базе есть разделители, которые могут разделять расширения конфигурации. Подробнее об этом рассказывалось в разделе [«Область действия»](#_Область_действия).

Если в текущем сеансе такие разделители не используются (например, для администратора, работающего с базой в неразделенном режиме), то в этом случае расширение может быть подключено как с областью действия *Информационная база*, так и с областью действия *Разделение данных*.

Если в текущем сеансе такие разделители используются (например, для «рядовых» пользователей сервиса), то в этом случае расширение может быть подключено только с областью действия *Разделение данных*.

Если конфигурация не содержит разделителей, которые могут разделять расширения, или же если такие разделители есть, но они условно отключены, то расширение может быть подключено к конфигурации только с областью действия *Информационная база*.

### Деактивация

Возможна ситуация (особенно в процессе разработки расширения), когда после подключения расширения в прикладном решении что-то пошло не так. Чтобы понять, «виновато» ли в этом расширение или нет, имеется удобная возможность деактивировать его и посмотреть, как будет работать прикладное решение без применения расширения.

Для этого достаточно снять у расширения флажок *Активно* в списке расширений, причем сделать это можно как в конфигураторе, так и в режиме *1С:Предприятие* в стандартной обработке *Управление расширениями конфигурации*.

Таким образом можно, не удаляя само расширение, сделать его неактивным. В результате оно останется в списке расширений конфигурации, но не будет подключаться при старте прикладного решения. Чтобы вновь запустить работу конфигурации с расширением, потребуется только один щелчок мышью.

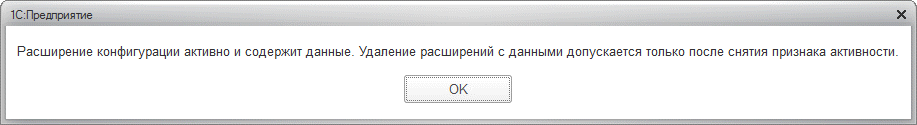
### Удаление

Кроме того, можно удалить расширение из списка кнопкой *Удалить* (как в конфигураторе, так и в режиме *1С:Предприятие*). Но надо иметь в виду следующее.

Помимо того что деактивация/активация расширений быстрей и удобней, чем их удаление и последующая загрузка, здесь еще важен вопрос безопасности при удалении расширений, особенно тех расширений, которые расширяют данные.

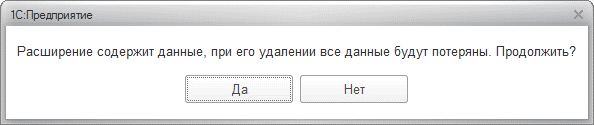
Для таких расширений создаются собственные структуры данных, которые были добавлены расширением. При удалении таких расширений данные, хранящиеся в этих структурах, будут потеряны.

Чтобы снизить риск потери данных, удаление расширений, расширяющих данные, выполняется в два этапа. Сначала расширение должно быть отключено – сброшен флажок *Активно* (рис. 1.17).



**Рис. 1.17.**Предупреждение при удалении расширений с данными

А затем уже расширение можно удалить из информационной базы, но потребуется в конфигураторе принять изменения в диалоге реструктуризации базы данных, а в режиме *1С:Предприятие* еще раз подтвердить удаление расширения(рис. 1.18).



**Рис. 1.18.**Предупреждение при удалении расширений с данными

Кроме того, во встроенном языке методом *ИзменяетСтруктуруДанных()*объекта *РасширениеКонфигурации* можно проверить, влияет ли расширение на структуры данных.

Надо понимать, что отключенное (деактивированное) расширение (с точки зрения работы внутренних механизмов) эквивалентно удаленному расширению, кроме того, что для отключенного расширения не удаляются структуры хранения данных (для расширения, которое расширяет данные). Поэтому лучше отключать такие расширения, а не удалять их.

Следует иметь в виду, что пока расширение открыто в конфигураторе, его невозможно удалить, деактивировать и т. п. в режиме *1С:Предприятие*.

## Проверка возможности применения

Конкретное расширение не всегда может примениться к конкретной конфигурации. Даже если само по себе расширение не содержит никаких ошибок.

Например, если в расширение заимствован справочник *Номенклатура*, а в конфигурации поставщик переименовал его в справочник *Товары*, то после обновления конфигурации расширение не сможет к ней примениться, пока в расширении имя справочника *Номенклатура* также не будет изменено на *Товары*.

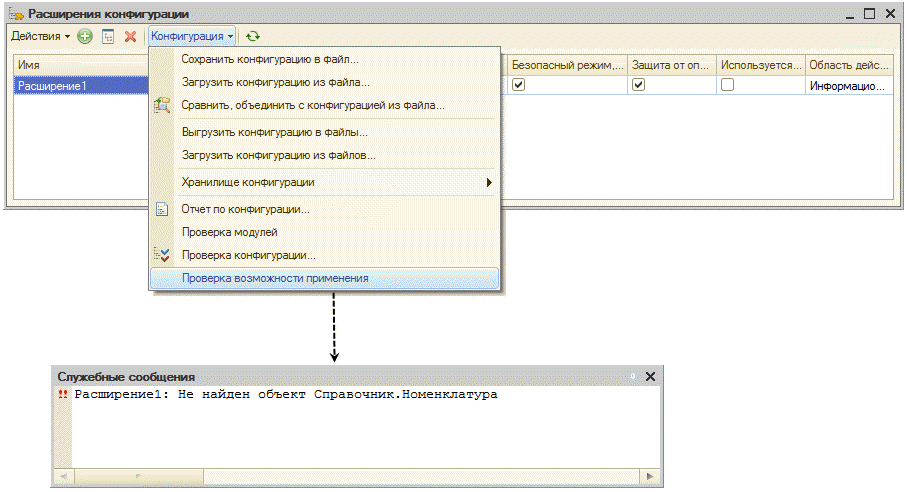
В этом случае при запуске «1С:Предприятия» пользователи прикладного решения увидят оповещение с сообщением об ошибке применения расширения. Это оповещение всплывет на несколько секунд в правом нижнем углу экрана и затем будет сохранено в центре оповещений. Чтобы увидеть более подробное сообщение, нужно нажать на это всплывающее окно или на сообщение из списка оповещений или нажать на гиперссылку *Информация для технической поддержки* в окне информации о программе (рис. 1.19).



**Рис. 1.19.**Сообщение об ошибке применения расширения в режиме «1С:Предприятие»

Гораздо удобнее проверить применимость расширений еще до их реального запуска вместе с конфигурацией. Для этого в платформе разработаны различные средства диагностики, которые позволяют это сделать в конфигураторе (в том числе при пакетном запуске конфигуратора), во встроенном языке и в режиме *1С:Предприятие* в стандартной обработке *Управление расширениями конфигурации*.

В конфигураторе для этого нужно создать у конфигурации расширение, загрузить его из файла и выполнить команду *Конфигурация – Проверка возможности применения*(рис. 1.20).



**Рис. 1.20.**Проверка возможности применения расширения к конфигурации из конфигуратора

Кроме того, можно проверить применимость сразу всех расширений, которые есть у конфигурации. Для этого нужно из контекстного меню списка расширений выполнить команду *Проверка возможности применения всех расширений*.

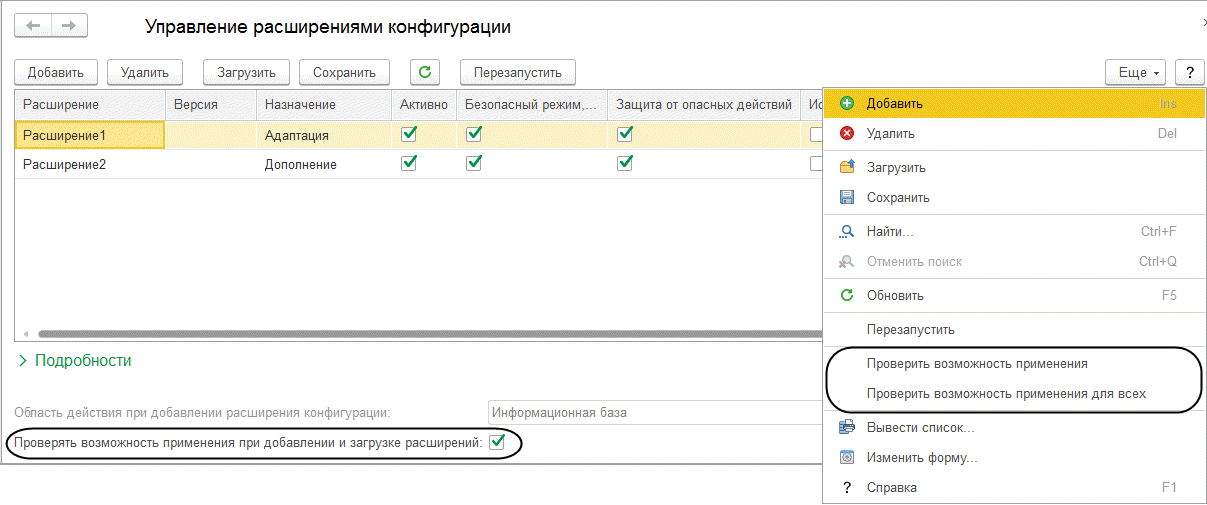
Аналогичную проверку можно выполнить и при пакетном запуске конфигуратора. Для этого нужно указать параметр командной строки */CheckCanApplyConfigurationExtensions*.

Во встроенном языке с помощью метода *ПроверитьВозможностьПримененияВсех()* объекта *МенеджерРасширенийКонфигурации*, а также метода *ПроверитьВозможностьПрименения()* объекта *РасширениеКонфигурации* можно проверить применимость всех (или одного) расширений информационной базы в текущей области данных по порядку загрузки с учетом уже участвовавших в проверке расширений.

Причем особенно важно и полезно то, что метод *ПроверитьВозможностьПрименения()* позволяет проверить применимость новой версии расширения еще до того, как она будет загружена в информационную базу. Новая версия расширения передается этому методу в виде двоичных данных.

И, наконец, в режиме *1С:Предприятие* в стандартной обработке *Управление расширениями конфигурации* перед загрузкой расширения в информационную базу стандартно проверяется его применимость к конфигурации. Так происходит потому, что флажок *Проверять возможность применения при добавлении и загрузке расширений* по умолчанию установлен.

Кроме того, в группе команд *Еще* находятся две команды, с помощью которых можно проверить применимость тех расширений, которые уже имеются в списке (рис. 1.21).



**Рис. 1.21.**Проверка возможности применения расширения к конфигурации из обработки «Управление расширениями конфигурации» в режиме «1С:Предприятие»

# Глава 2. Функциональность

Более подробно работа с расширениями будет рассмотрена на небольших отдельных примерах, описанных в данной главе. Эти примеры никак не связаны друг с другом, а конфигурация не имеет никакой прикладной ценности и используется просто в демонстрационных целях.

Общая стратегия применения расширений заключается в том, что изменения, содержащиеся в расширении, дополняют конфигурацию. Но когда изменения, сделанные в расширении и в конфигурации, «пересекаются» – применяется расширение. Рассмотрим это на конкретных примерах.

## Формы

С помощью расширения можно дорабатывать формы, существующие в конфигурации. Можно не только доработать внешний вид формы, но и добавить в нее недостающие данные, дополнить ее командный интерфейс, а главное – можно доработать модуль формы и, соответственно, изменить ее поведение. То есть дорабатывать заимствованные формы можно практически без ограничений.

Исключение составляют элементы, образующие командный интерфейс формы. Удалять команды, существующие в заимствованной форме, нельзя. В расширении можно только доработать обработчики этих команд или добавить в нее собственные команды.

Кроме того, можно добавить собственные формы и назначить их основными для объектов конфигурации или вызывать их программным образом.

### Как доработать внешний вид формы

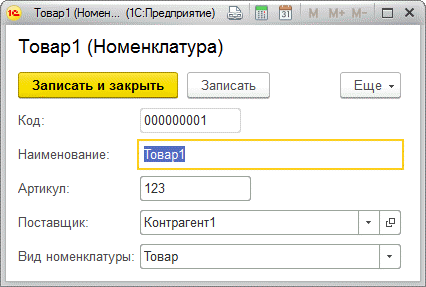
В этом примере мы покажем самые простые доработки формы:

* изменение порядка элементов;
* объединение элементов в группу;
* изменение заголовка поля.

Пусть в конфигурации существует справочник Номенклатура. Справочник имеет реквизиты:

* *ПолноеНаименование* типа *Строка*;
* *Артикул* типа *Строка*;
* *Поставщик* типа *СправочникСсылка.Контрагенты*;
* *ВидНоменклатуры* типа *ПеречислениеСсылка.ВидыНоменклатуры*;
* *ШтрихКод* типа *Строка*.

Основная форма элемента этого справочника выглядит следующим образом (рис. 2.1).



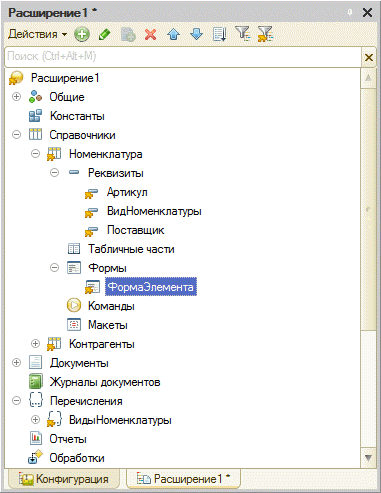
**Рис. 2.1.**Форма элемента справочника

Наша задача – доработать внешний вид этой формы с помощью расширения:

* Объединить элементы *Код* и *Наименование* в горизонтальную группу.
* Поле *Вид номенклатуры* расположить первым, сразу под группой.
* Заголовок поля *Вид номенклатуры* изменить на «Тип».

Для этого нам нужно создать расширение, позаимствовать форму в расширение и изменить внешний вид заимствованной формы.

Позаимствуем форму элемента справочника. После этого в структуре объектов расширения появится сама форма и заимствованные вместе с ней объекты (рис. 2.2).



**Рис. 2.2.**Структура объектов расширения

Расширение устроено так, что при заимствовании объекта заимствуется и его родительский объект (вместе с его стандартными реквизитами), и все его предки, если их еще нет в расширении. Поэтому кроме формы был заимствован и сам справочник *Номенклатура*. Подчиненные объекты не заимствуются.

Но когда заимствуется форма, то в расширение заимствуются также все объекты конфигурации, которые содержат данные, отображаемые в форме. Поэтому кроме формы элемента и самого справочника в расширение были заимствованы также:

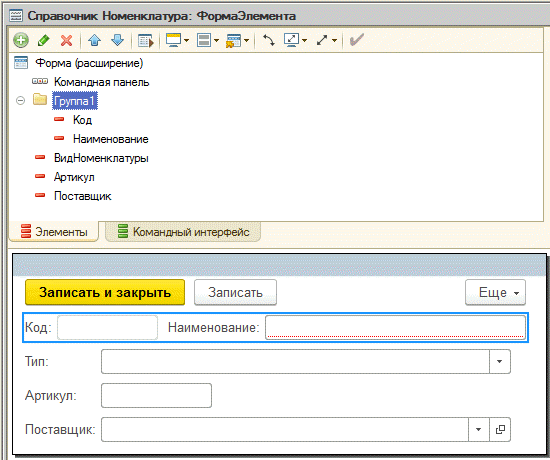
* те реквизиты справочника, которые присутствуют в форме: *Артикул*, *Поставщик* и *ВидНоменклатуры*;
* те объекты, которые образуют типы реквизитов, присутствующих в форме: справочник *Контрагенты* и перечисление *ВидыНоменклатуры*.

Как можно заметить (см. рис. 2.2), реквизиты справочника *ПолноеНаименование* и *Штрихкод* заимствованы не были, так как они не используются в форме. Также было заимствовано только само перечисление *ВидыНоменклатуры*, поскольку этот тип данных используется в форме, но не были заимствованы его значения, так как это подчиненные объекты перечисления.

Также необходимо учитывать, что в расширении свойства заимствованных объектов принимают значения по умолчанию. Поэтому свойство заимствованного справочника *Основная форма объекта* не копируется из конфигурации, а остается пустым. И поэтому заимствованную форму нельзя открыть командой *Открыть основную форму объекта* из контекстного меню справочника. Для этого нужно использовать команду *Открыть*. Но заимствованная и доработанная форма при применении расширения в режиме *1С:Предприятие* все равно будет вызываться как основная форма элемента справочника, потому что так задано в свойстве исходного справочника.

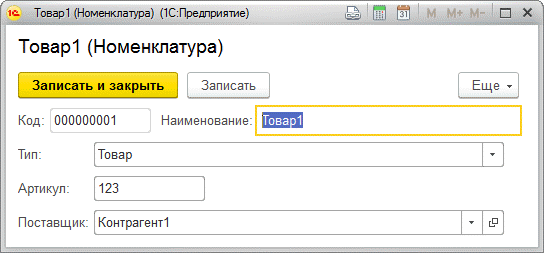
После заимствования форма элемента справочника автоматически откроется в редакторе формы. На данный момент она визуально совпадает с формой конфигурации. Доработаем ее в соответствии с нашей задачей.

Добавим в форму обычную группу и перенесем туда элементы *Код* и *Наименование*. У этой группы снимем флажок *Отображать заголовок*. Остальные элементы расположим в следующем порядке: *ВидНоменклатуры*, *Артикул* и *Поставщик*. У поля *ВидНоменклатуры* установим заголовок «Тип» (рис. 2.3).



**Рис. 2.3.**Редактирование формы элемента справочника в расширении

Затем запустим конфигурацию в режиме отладки. В результате форма элемента справочника *Номенклатура* будет выглядеть так, как мы и задали в расширении (рис. 2.4).



**Рис. 2.4.**Форма элемента справочника «Номенклатура»

#### Что произойдет, если расширяемая форма изменится после обновления конфигурации

При разработке расширений нужно иметь в виду, что конфигурация может в любой момент измениться в результате ее обновления поставщиком. Важно понимать, что происходит с формой в этом случае, какие изменения будут применены в результате и что нужно делать разработчику расширения.

Визуальная часть формы фиксируется в расширении на момент ее заимствования. Эта форма становится *сохраненной*. А заимствованная и доработанная в расширении форма становится *расширяющей* формой.

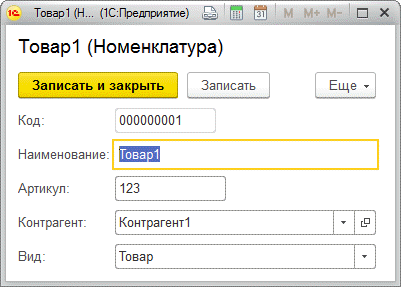
При запуске конфигурации в режиме *1С:Предприятие* для каждого элемента формы анализируются доработки относительно состояния сохраненной формы, которые были сделаны в форме конфигурации и в расширяющей форме. Если изменения были только в форме конфигурации, то берутся доработки из нее. В остальных случаях применяются доработки из расширяющей формы.

Рассмотрим, как будет вести себя расширение, разработанное нами ранее, в случае изменения расширяемой формы.

Предположим, в новой версии конфигурации поставщик доработал расширяемую форму:

* у поля *ВидНоменклатуры* заголовок изменился на «Вид»;
* у поля *Поставщик* – на «Контрагент».

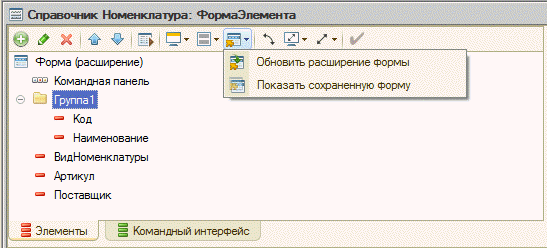
После обновления конфигурации форма элемента справочника выглядит следующим образом (рис. 2.5).



**Рис. 2.5.** Обновленная расширяемая форма элемента справочника

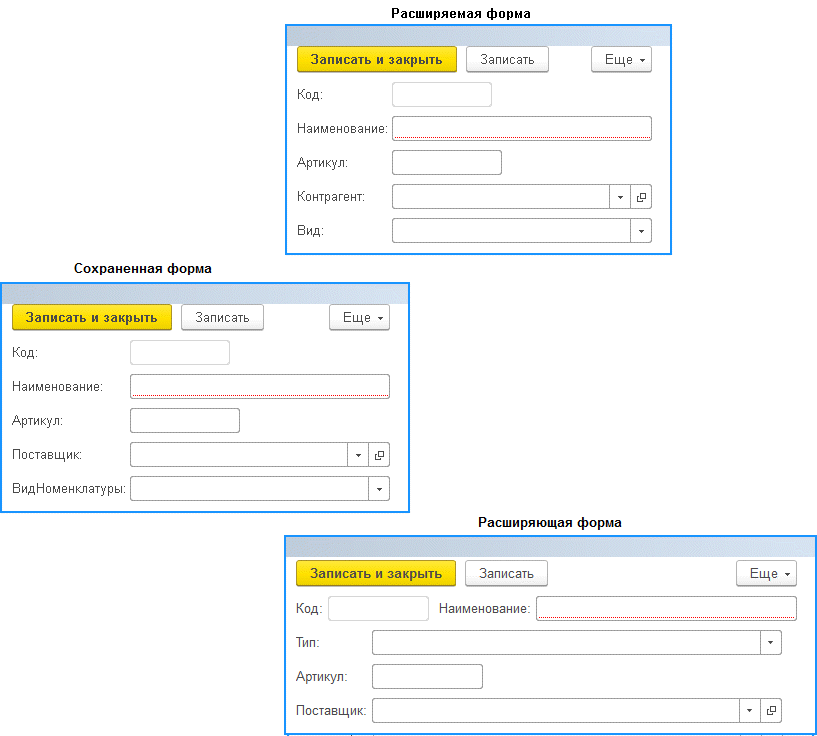
Чтобы понять, как изменится форма номенклатуры при применении расширения к обновленной конфигурации, проанализируем все три формы: сохраненную форму, расширяемую форму и расширяющую форму.

Чтобы увидеть сохраненную форму, нужно в редакторе формы в расширении выполнить команду *Расширения – Показать сохраненную форму* из командной панели дерева элементов формы (рис. 2.6).



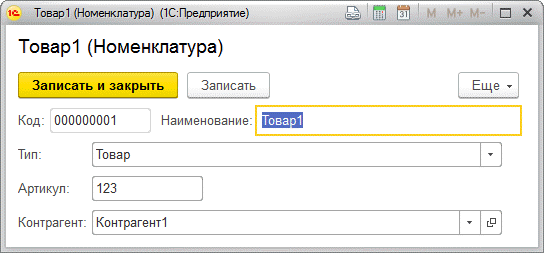
**Рис. 2.6.**Открытие сохраненной формы

Итак, по сравнению с сохраненной формой произошли следующие изменения. В расширяющей форме мы перегруппировали элементы, изменили их порядок и изменили заголовок у поля *ВидНоменклатуры* на «Тип». В расширяемой форме в результате обновления конфигурации заголовок поля *ВидНоменклатуры* изменился на «Вид», а у поля *Поставщик* – на «Контрагент» (рис. 2.7).



**Рис. 2.7.**Анализ изменений, сделанных в расширяемой и расширяющей формах по сравнению с сохраненной формой

В результате в режиме *1С:Предприятие* форма элемента справочника будет выглядеть следующим образом (рис. 2.8).



**Рис. 2.8.**Форма элемента справочника в режиме «1С:Предприятие»

Мы видим, что заголовок поля *Поставщик* – «Контрагент» – взят из расширяемой формы, так как по сравнению с сохраненной формой он менялся только там. Все остальные изменения взяты из расширяющей формы, как и было написано выше.

В данном случае, посмотрев на форму номенклатуры, получившуюся в результате применения расширения к обновленной конфигурации, разработчик может вообще не менять свое расширение. Ведь его доработки остались на месте. Поле *Вид номенклатуры* будет иметь заголовок «Тип», сколько бы его ни менял поставщик конфигурации. А какой заголовок будет у поля *Поставщик*, вообще особо не волнует разработчика расширения. Поэтому он его и не задавал в расширяющей форме.

Но могут быть и более серьезные изменения расширяемой формы, которые требуют вмешательства разработчика расширения.

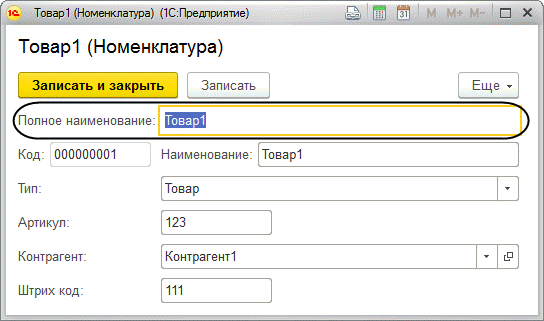
Например, поставщик конфигурации поместил в форму элемента справочника элементы для отображения реквизитов *ПолноеНаименование* и *Штрихкод* и расположил поле *ПолноеНаименование* сразу после поля *Наименование*, а поле *Штрихкод* – последним в списке элементов формы.

После обновления конфигурации форма элемента справочника выглядит следующим образом (рис. 2.9).



**Рис. 2.9.** Обновленная расширяемая форма элемента справочника

Но после применения расширения к обновленной конфигурации форма номенклатуры будет выглядеть как-то странно и некрасиво (рис. 2.10).

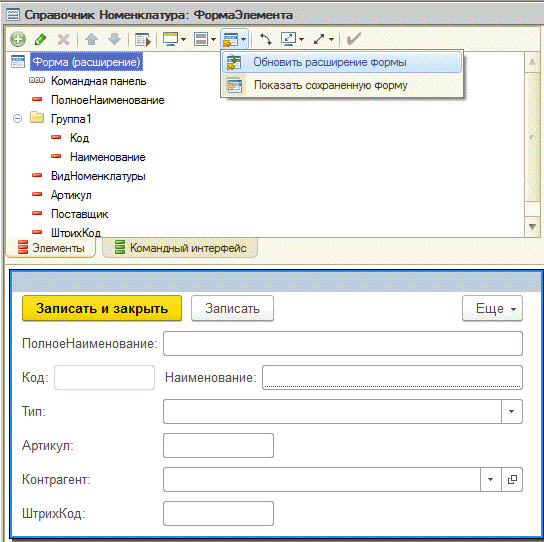


**Рис. 2.10.**Форма элемента справочника в режиме «1С:Предприятие»

Дело в том, что поле *Штрихкод* так и будет последним в списке элементов, так как оно последнее в расширяемой форме. А поле *Полное наименование* станет первым, так как элементы, которые были над ним в расширяемой форме (*Код* и *Наименование*), объединены в расширяющей форме в группу. Нас этот порядок не устраивает.

Для решения возникшей проблемы нам нужно перенести в расширяющую форму изменения, которые были сделаны в расширяемой форме после ее заимствования. И затем сделать необходимые доработки (в данном случае поменять порядок элементов формы). Для этого нужно выполнить команду *Расширения – Обновить расширение формы* из командной панели редактора расширяющей формы (см. рис. 2.11).

В результате обновления расширяющая форма будет выглядеть следующим образом (рис. 2.11).



**Рис. 2.11.** Обновленная расширяющая форма элемента справочника

Расположим поле *Полное наименование* сразу под группой. В результате в режиме *1С:Предприятие* форма элемента справочника будет выглядеть так, как мы и хотели (рис. 2.12).



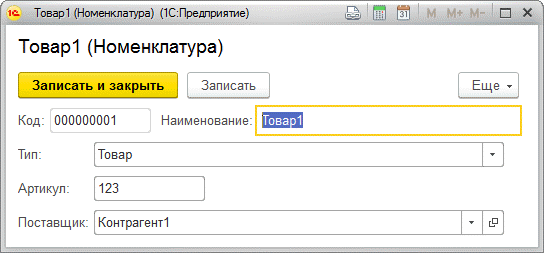
**Рис. 2.12.**Форма элемента справочника в режиме «1С:Предприятие»

#### Что произойдет, если к конфигурации будет применено еще одно расширение, дорабатывающее ту же форму конфигурации

При разработке расширений нужно учитывать, что в информационную базу может быть загружено сразу несколько расширений. И если они дорабатывают один и тот же объект, например форму, то изменения последнего применившегося к конфигурации расширения будут применены также последними. То есть в случае «пересечения» этих изменений они «затрут» изменения ранее применившихся расширений.

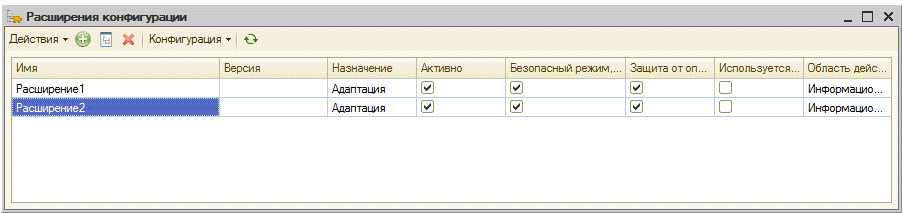
Порядок применения расширений можно задать в свойстве *Назначение расширения конфигурации*, которое может принимать значения *Исправление*, *Адаптация* и *Дополнение*. В зависимости от своего назначения расширения применяются к конфигурации именно в приведенном выше порядке.

В исходном примере (см. раздел [«Как доработать внешний вид формы»](#_Как_доработать_внешний)) мы рассматривали форму элемента справочника *Номенклатура*, доработанную расширением. При создании расширения ему было установлено назначение по умолчанию – *Адаптация*. После применения расширения форма имела следующий вид (рис. 2.13).



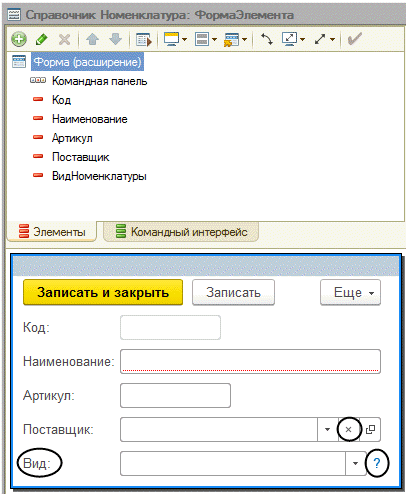
**Рис. 2.13.**Форма элемента справочника после применения первого расширения с назначением «Адаптация»

Сымитируем теперь ситуацию, когда другой разработчик тоже создал для конфигурации расширение, которое дорабатывает ту же форму справочника. По умолчанию второе расширение также имеет назначение *Адаптация* (рис. 2.14).



**Рис. 2.14.**Список расширений

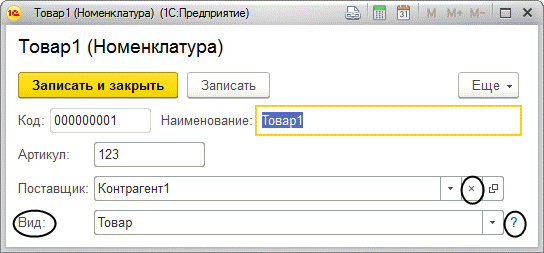
Во втором расширении полю *ВидНоменклатуры* был задан заголовок «Вид», а также введена подсказка, которая отображается в виде кнопки. Кроме того, у поля *Поставщик* явно добавлена кнопка очистки (рис. 2.15).



**Рис. 2.15.**Форма элемента справочника, разработанная во втором расширении

Теперь разберемся с порядком применения расширений. Второе расширение тоже имеет назначение *Адаптация*, как и первое, но оно было сохранено в информационной базе позже, чем первое. Поэтому оно применится к конфигурации последним.

Соответственно, в режиме *1С:Предприятие* мы увидим, что на расширяемую форму элемента справочника (см. рис. 2.1) были наложены сначала доработки первого расширения, а затем на получившийся результат были наложены доработки второго расширения (рис. 2.16).



**Рис. 2.16.**Форма элемента справочника после применения первого, а затем второго расширения

Таким образом, мы видим, что группировка элементов формы взята из первого расширения, а порядок несгруппированных элементов, заголовок и кнопка подсказки у поля *ВидНоменклатуры* и кнопка очистки у поля *Поставщик* – из второго расширения. То есть там, где изменения обоих расширений «пересекаются», в результате остаются доработки последнего применившегося к конфигурации расширения.

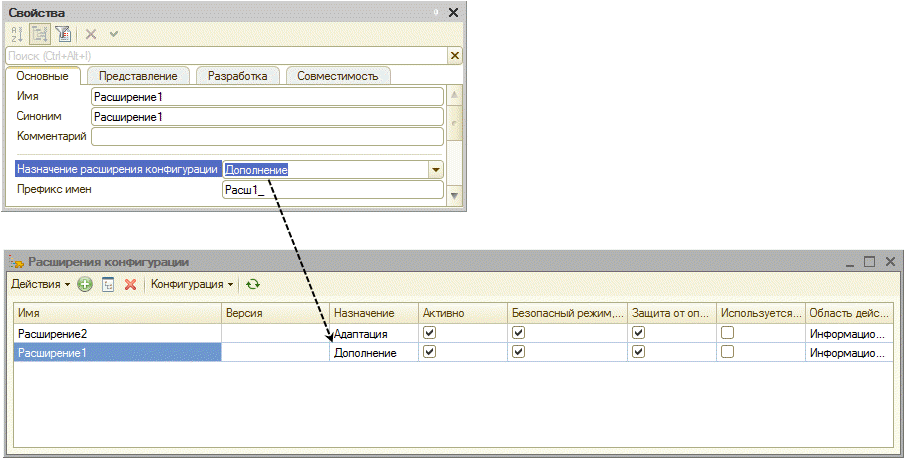
Однако, как правило, разработчик расширения знает, с какой целью он создает расширение. Если он вносит исправления в конфигурацию, то он должен установить назначение расширения как *Исправление*. В этом случае он может быть уверен, что его расширение применится к конфигурации первым. Затем применяется расширение с назначением *Адаптация*, содержащее доработки, сделанные для конкретного заказчика, и последним – расширение с назначением *Дополнение*, содержащее дополнительные сервисы для конфигурации в целом.

Мы специально не будем больше рассматривать ситуацию, когда несколько расширений имеют одно и то же назначение, так как предполагается, что расширения с одинаковым назначением не должны дорабатывать одни и те же объекты.

Предположим, разработчик считает, что его доработки наиболее важны. Тогда ему нужно обеспечить, чтобы его расширение применялось к конфигурации последним. Для этого ему нужно установить назначение расширения *Дополнение*.

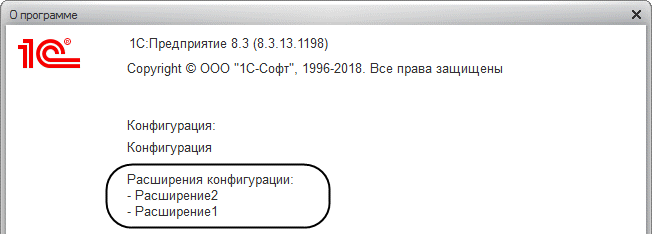
Продемонстрируем эту ситуацию на нашем примере с двумя расширениями (см. рис. 2.14). Свойство первого расширения *Назначение расширения конфигурации* установим в значение *Дополнение*. Заметьте, что назначение расширения в списке расширений изменяется только после обновления конфигурации базы данных.

Запустим конфигурацию в режиме отладки. После обновления конфигурации базы данных порядок расположения расширений в списке изменится в соответствии с порядком их применения (рис. 2.17).



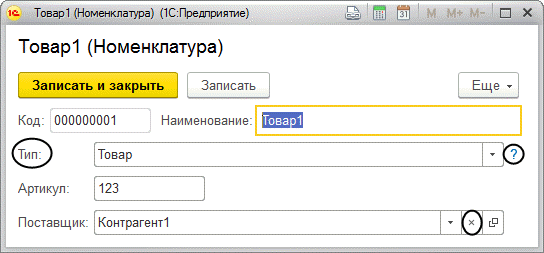
**Рис. 2.17.**Изменение порядка применения расширений в соответствии с их назначением

В режиме *1С:Предприятие* откроем окно информации о программе и убедимся, что расширения применились к конфигурации в том порядке, в котором планировалось (рис. 2.18).



**Рис. 2.18.**Порядок применения расширений в режиме «1С:Предприятие»

Откроем форму номенклатуры (рис. 2.19).



**Рис. 2.19.**Форма элемента справочника после применения второго, а затем первого расширения

Мы видим, что доработки, сделанные вторым расширением (кнопка очистки и кнопка подсказки), по-прежнему видны в результирующей форме. А вот заголовок поля *ВидНоменклатуры* «Вид» изменен первым расширением на «Тип». Порядок всех элементов формы также взят из первого расширения, так как оно применилось к конфигурации последним.

### Как добавить объекту собственную основную форму

Как уже было сказано, в расширение можно добавлять собственные формы и назначать их основными.

Для примера рассмотрим следующую задачу. В конфигурации существует основная форма справочника, которая нас не устраивает ни внешним видом, ни своей функциональностью. Поэтому вместо нее мы хотим использовать совсем другую форму.

Наша задача легко решается с помощью расширения. Для этого нам нужно добавить собственную форму в расширение, назначить ее основной для справочника и наполнить ее своей функциональностью.

Пусть в конфигурации существует справочник *Контрагенты* с реквизитами:

* *Адрес* типа *Строка*;
* *Телефон* типа *Строка*;
* *ЭлектроннаяПочта* типа *Строка* и др.

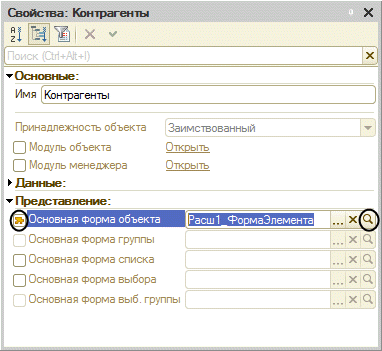
Основная форма элемента этого справочника выглядит следующим образом (рис. 2.20).



**Рис. 2.20.**Форма элемента справочника

Для решения нашей задачи позаимствуем справочник. Поскольку вместе со справочником заимствуются только его стандартные реквизиты, мы должны теперь самостоятельно позаимствовать остальные реквизиты справочника: *Телефон*, *Адрес* и *ЭлектроннаяПочта*.

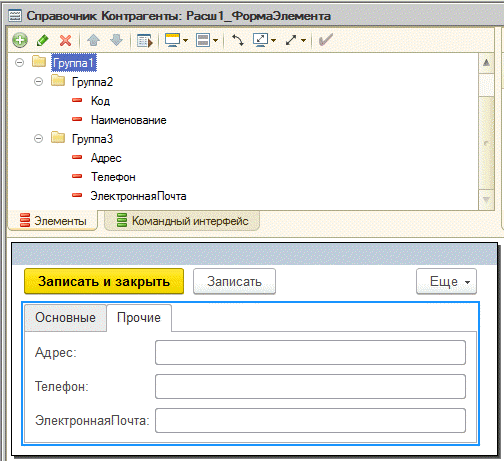
У свойства справочника *Основная форма объекта* установим флажок модифицированности. После этого нажмем кнопку открытия со значком лупы и добавим собственную основную форму элемента справочника. Ей автоматически присвоится имя *Расш1\_ФормаЭлемента* (рис. 2.21).



**Рис. 2.21.**Добавление основной формы элемента справочника

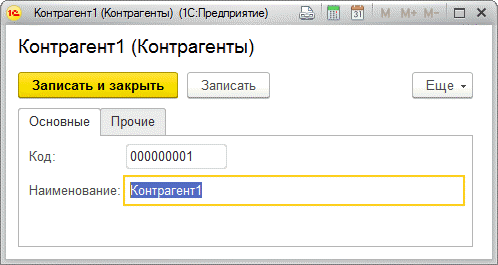
Дальнейшие действия с собственной формой ничем не отличаются от работы с формой в конфигурации.

Поэтому, просто для того чтобы посмотреть, какая форма теперь используется в качестве основной формы элемента справочника, доработаем собственную форму. Например, добавим в нее группу с двумя страницами. На первую страницу перенесем элементы для отображения стандартных реквизитов *Код* и *Наименование*, на вторую – все остальные элементы (рис. 2.22).



**Рис. 2.22.**Форма элемента справочника в редакторе формы

Запустим «1С:Предприятие» в режиме отладки и убедимся, что при открытии формы контрагента вызывается не расширяемая, а собственная форма (рис. 2.23).



**Рис. 2.23.**Форма элемента справочника в режиме «1С:Предприятие»

**ПРИМЕЧАНИЕ**

*Следует учитывать, что если какой-то реквизит отключен функциональной опцией, то вы его не увидите в форме в режиме 1С:Предприятие. То есть это не особенность работы расширения, а стандартное поведение платформы.*

Могут быть и другие примеры использования собственных форм. Например, можно добавить собственную форму документа, но не назначать ее основной, а показывать опционально. То есть при открытии документа обычно будет использоваться форма, назначенная основной в конфигурации. Но по запросу пользователя (например, установленному флажку) будет открываться другая, более специфичная, собственная форма расширения.

#### Что произойдет, если основная форма объекта изменится в конфигурации

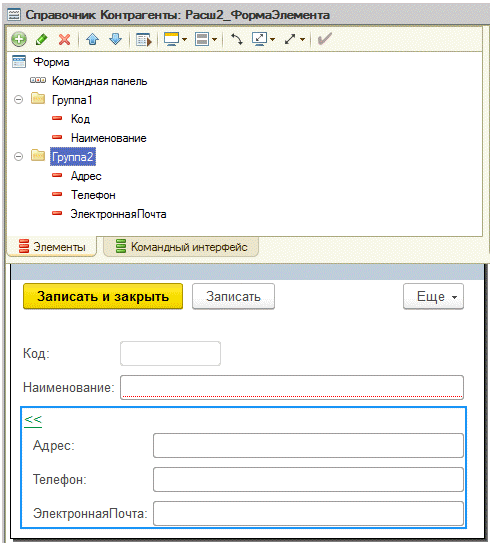
Предположим, что в новой версии конфигурации поставщик изменил основную форму элемента справочника *Контрагенты* или назначил в качестве основной совсем другую форму.

После применения расширения к обновленной конфигурации в режиме *1С:Предприятие* по-прежнему будет использоваться собственная форма, назначенная основной формой элемента справочника в расширении, потому что у свойства справочника *Основная форма объекта* нами установлен флажок модифицируемости. Это значит, что значение этого свойства в результате будет браться из расширения.

#### Что произойдет, если к конфигурации будет применено еще одно расширение, устанавливающее основную форму

В исходном примере (см. раздел [«Как добавить объекту собственную основную форму»](#_Как_добавить_объекту)) мы добавили в расширение собственную форму и назначили ее основной формой элемента справочника *Контрагенты* (см. рис. 2.23).

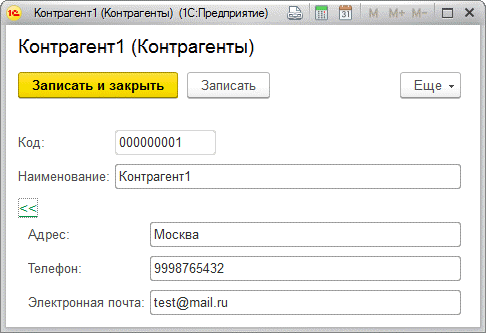
Предположим, другой разработчик сделал то же самое в другом расширении. То есть он заимствовал справочник, добавил в расширение собственную форму и назначил ее основной формой элемента справочника. В редакторе формы эта форма выглядит следующим образом (рис. 2.24).



**Рис. 2.24.**Форма элемента справочника в редакторе формы

Поскольку у свойства справочника *Основная форма объекта* в обоих расширениях установлен флажок модифицируемости, то, какая форма будет показана в режиме *1С:Предприятие*, будет зависеть непосредственно от порядка применения расширений.

Если у первого расширения будет установлено назначение *Исправление*, а у второго – *Адаптация* или *Дополнение* или же если у первого расширения будет установлено назначение *Адаптация*, а у второго – *Дополнение*, то сначала будет применено первое расширение, а затем – второе. Соответственно, в качестве основной формы элемента справочника будет использоваться форма из второго расширения (рис. 2.25).



**Рис. 2.25.**Форма элемента справочника в режиме «1С:Предприятие»

Если же в качестве формы контрагента должна показываться форма из первого расширения, то разработчик этого расширения должен установить ему назначение *Дополнение*. Тогда в режиме *1С:Предприятие*первое расширение применится последним, и при открытии формы контрагента будет вызываться разработанная нами выше форма (см. рис. 2.23) из первого расширения.

Надо понимать, что в наших примерах мы меняем назначение расширений просто для демонстрации порядка их применения к конфигурации. Однако в действительности назначение расширения должно соответствовать характеру доработок, которые в нем содержатся. Например, первое расширение действительно содержит исправления в конфигурации, а второе – необходимые доработки для заказчика.

Тогда менять порядок применения расширений (первое с назначением *Исправление*, второе с назначением *Адаптация*) только для того, чтобы показать нужную форму, неправильно. Если нужно, чтобы в качестве основной формы справочника использовалась форма из первого расширения и при этом оно применялось бы первым, нужно просто снять флажок модифицируемости у свойства справочника *Основная форма объекта* во втором расширении.

### Как создать обработчик события для элемента формы

У форм объектов, существующих в конфигурации, можно дорабатывать не только внешний вид, но и расширять их функциональность. Например, в модуле расширяющей формы можно создавать новые обработчики событий, которых еще нет в модуле расширяемой формы. То есть модуль формы конфигурации также можно дорабатывать, как и саму форму.

Пусть в конфигурации существует справочник *Сотрудники* с реквизитами:

* *Подразделение* типа *СправочникСсылка.Подразделения*;
* *Должность* типа *СправочникСсылка.Должности*;
* *Стаж* типа *Строка*;
* *Оклад* типа *Число*.

Справочник *Должности* имеет реквизиты:

* *МинОклад* типа *Число*;
* *МаксОклад* типа *Число*.

В этих реквизитах хранятся минимальный и максимальный оклады, установленные для каждой должности.

У справочника *Сотрудники* существует основная форма элемента. Наша задача – при изменении поля *Должность* в форме сотрудника подставить в реквизит *Оклад* соответствующее значение из справочника *Должности*.

Для этого нам нужно позаимствовать форму сотрудника в расширение, создать в ней обработчик события *ПриИзменении* у элемента формы *Должность* и написать в нем соответствующий код.

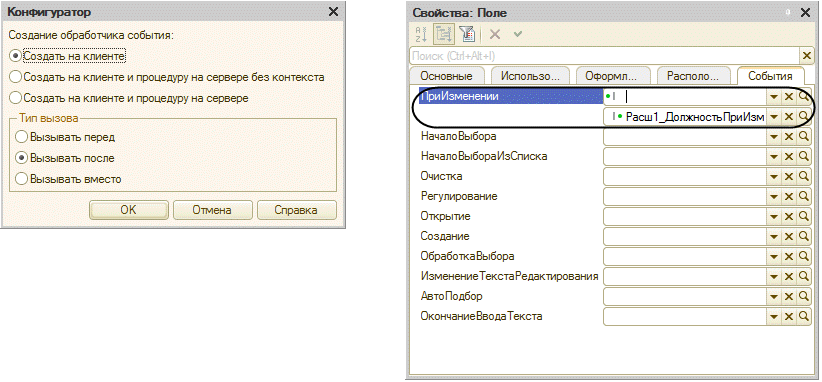
Казалось бы, это очень просто. Но в расширяющей форме процесс создания обработчиков событий и команд имеет свои особенности, которые нужно знать и учитывать.

Не вдаваясь в детали, о которых будет подробно рассказано в разделе «[Модули – Общая информация](#_Общая_информация)», поясним, что модуль заимствованной формы изначально пустой. Но в нем можно создавать обработчики различных событий, которые будут выполняться перед, после или вместо обработчиков конфигурации, перехватывая исполнение встроенного языка. Такие обработчики мы будем называть далее *расширяющий метод Перед*, *расширяющий метод После* и *расширяющий метод Вместо*.

В модуле расширяющей формы можно также создать обработчик события, который отсутствует в модуле формы конфигурации. Его мы тоже будем называть расширяющим методом, несмотря на то что расширяемого обработчика еще нет.

В соответствии с нашей задачей позаимствуем форму элемента справочника *Сотрудники*. Откроем ее и создадим обработчик события *ПриИзменении* для поля *Должность*. Появится диалог с вопросом о контексте выполнения обработчика. Но к нему еще добавлен выбор типа вызова обработчика – перед, после или вместо расширяемого обработчика (см. рис. 2.26).

Укажем, что обработчик будет выполняться на клиенте и вызываться после расширяемого обработчика. После этого в модуле формы появится шаблон процедуры-обработчика *Расш1\_ДолжностьПриИзмененииПосле*, и одновременно в палитре свойств элемента формы *Должность* эта процедура будет назначена в качестве обработчика события *ПриИзменении* (рис. 2.26).



**Рис. 2.26.**Создание шаблона обработчика события и привязка его к элементу расширяющей формы в палитре свойств

Имя процедуры-обработчика автоматически начинается с префикса *Расш1\_*. К имени обработчика также автоматически добавляется постфикс *После*, который указывает, когда будет вызван обработчик (перед, после или вместо) по отношению к расширяемому обработчику.

При этом в палитре свойств (см. рис. 2.26) в случае выбора расширяющего метода *Перед* или *После* для этого же события появляется свободная строка для назначения «парного» расширяющего метода. Его можно создать и привязать к тому же событию*ПриИзменении* поля формы, если требуется «обрамить» своими вызовами расширяемый обработчик.

Рядом с именем обработчика появляется пиктограмма, отображающая тип перехвата. Для расширяющего метода *Перед* или *После* пиктограмма имеет вид точки рядом с вертикальной чертой. Местонахождение точки – перед или после черты – характеризует тип расширяющего метода.

Для расширяющего метода *Вместо* (перекрывающего вызов расширяемого обработчика) парной строки не появляется и пиктограмма имеет вид одинарной точки.

Прежде чем писать процедуру для получения оклада при изменении должности, позаимствуем реквизиты *МинОклад*, *МаксОклад* из справочника *Должности*, так как эти реквизиты будут использоваться в коде встроенного языка.

Процедуру обработчика заполним следующим образом (листинг 2.1).

**Листинг 2.1.** Расширяющий метод «После» события «ПриИзменении» поля формы «Должность»

&НаКлиенте

Процедура Расш1\_ДолжностьПриИзмененииПосле(Элемент)

Объект.Оклад = Расш1\_ПолучитьОклад(Объект.Должность);

КонецПроцедуры

Для получения оклада сотрудника используем функцию *Расш1\_ПолучитьОклад()*, которая возвращает из справочника должностей минимальный оклад по переданной ссылке (листинг 2.2).

**Листинг 2.2.** Функция «Расш1\_ПолучитьОклад»

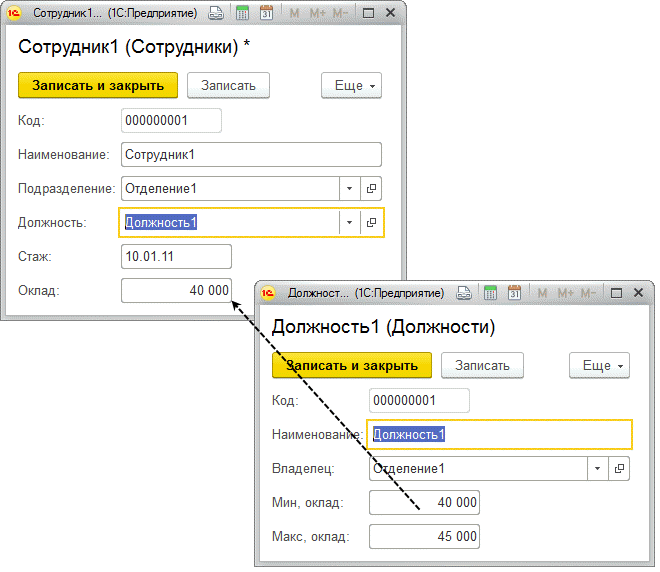
&НаСервереБезКонтекста

Функция Расш1\_ПолучитьОклад(Должность)

Возврат Должность.МинОклад;

КонецФункции

В результате в режиме *1С:Предприятие* при изменении значения в поле *Должность* сотруднику будет назначен минимальный оклад, соответствующий выбранной должности (рис. 2.27).



**Рис. 2.27.**Получение оклада при изменении должности в форме сотрудника

Таким образом, несмотря на то что обработчика события *ПриИзменении* у поля *Должность* в расширяемой форме сотрудника нет, все равно вызывается расширяющий метод *После*, разработанный нами в расширении, и нужный нам функционал добавляется в форму.

При выборе типа расширяющего метода необходимо учитывать особенности и порядок вызова расширяющих методов, описанные в разделе «[Модули – Стратегия выбора типа расширяющих методов](#_Стратегия_выбора_типа)».

#### Что произойдет, если в конфигурации появится обработчик события

Предположим, что в новой версии конфигурации поставщик создал обработчик события *ПриИзменении* для поля *Должность* и заполнил его следующим образом (листинг 2.3).

**Листинг 2.3.** Обработчик события «ПриИзменении» поля формы «Должность»

&НаКлиенте

Процедура ДолжностьПриИзменении(Элемент)

Объект.Оклад = ПолучитьОклад(Объект.Должность);

КонецПроцедуры

Для получения оклада используется функция *ПолучитьОклад()*, которая возвращает из справочника должностей максимальный оклад по переданной ссылке (листинг 2.4).

**Листинг 2.4.** Функция «ПолучитьОклад»

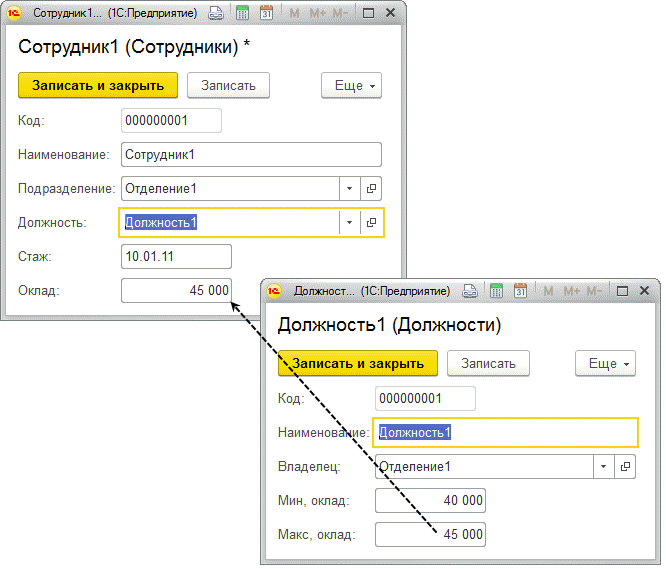
&НаСервереБезКонтекста

Функция ПолучитьОклад(Должность)

Возврат Должность.МаксОклад;

КонецФункции

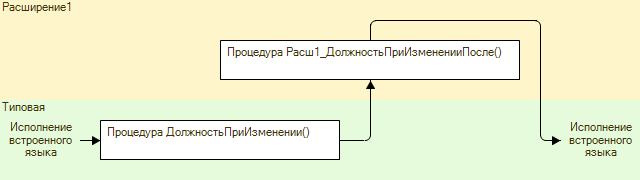
После запуска обновленной конфигурации в режиме *1С:Предприятие* без расширения при изменении значения в поле *Должность* сотруднику будет назначен максимальный оклад, соответствующий выбранной должности (рис. 2.28).



**Рис. 2.28.**Получение оклада при изменении должности в форме сотрудника

Однако после применения расширения к обновленной конфигурации результат будет таким же, как и до обновления конфигурации (см. рис. 2.27). Так получается потому, что расширяющий метод имеет тип *После*. Это значит, что выполняется сначала расширяемый обработчик, а потом наш расширяющий метод.

Последовательность выполнения обработчиков можно представить на следующей схеме (рис. 2.29).



**Рис. 2.29.**Порядок выполнения обработчиков событий при изменении поля «Должность»

В результате сотруднику при выборе должности (в нашем случае *Должность1*) присваивается сначала максимальный оклад (45000), а затем минимальный оклад (40000).

По аналогии понятно, что если бы расширяющий метод имел тип *Перед*, то последним выполнялся бы расширяемый обработчик и сотруднику присваивался бы максимальный оклад.

#### Что произойдет, если к конфигурации будет применено еще одно расширение, перехватывающее тот же обработчик события

Предположим, что другой разработчик в другом расширении тоже создал расширяющий метод *После* того же обработчика события и заполнил его следующим образом (листинг 2.5).

**Листинг 2.5.** Расширяющий метод «После» обработчика события «ПриИзменении» поля формы «Должность»

&НаКлиенте

Процедура Расш2\_ДолжностьПриИзмененииПосле(Элемент)

Объект.Оклад = Расш2\_ПолучитьОклад(Объект.Должность);

КонецПроцедуры

Для получения оклада используется функция *Расш2\_ПолучитьОклад()*, которая возвращает из справочника должностей среднее арифметическое из минимального и максимального оклада по переданной ссылке (листинг 2.6).

**Листинг 2.6.** Функция «Расш2\_ПолучитьОклад»

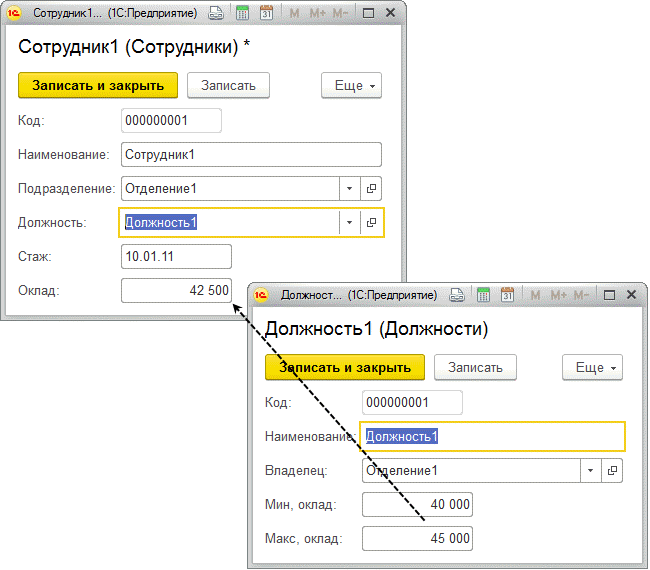
&НаСервереБезКонтекста

Функция Расш2\_ПолучитьОклад(Должность)

Возврат (Должность.МинОклад + Должность.МаксОклад) / 2;

КонецФункции

Результат будет зависеть от порядка применения расширений. Если второе расширение применится последним, то при изменении значения в поле *Должность* сотруднику будет назначен оклад, равный среднему арифметическому из минимального и максимального окладов для выбранной должности (рис. 2.30).

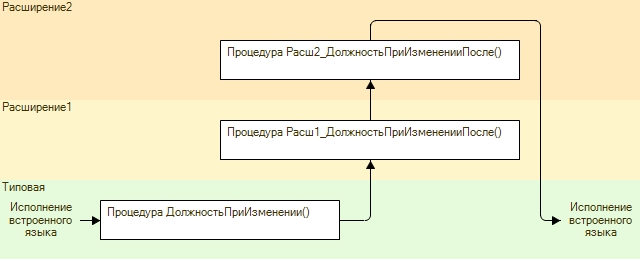


**Рис. 2.30.**Получение оклада при изменении должности в форме сотрудника

Порядок выполнения нескольких расширяющих методов с различными типами перехвата подробно рассматривается в разделе «[Модули – Взаимодействие нескольких расширений](#_Взаимодействие_нескольких_расширени)».

В нашем случае выполняется сначала расширяемый обработчик (если он существует), потом расширяющий метод из первого расширения, а затем расширяющий метод из второго расширения.

Последовательность выполнения обработчиков можно представить на следующей схеме (рис. 2.31).



**Рис. 2.31.**Порядок выполнения обработчиков событий при изменении поля «Должность»

В результате сотруднику при выборе должности (в нашем случае *Должность1*) присваивается сначала максимальный оклад (45000), затем минимальный оклад (40000), а затем среднее арифметическое двух окладов (42500).

Если же последним применится первое расширение, то сотруднику будет назначен минимальный оклад (см. рис. 2.27).

### Как перехватить обработчик события и обрамить его своими алгоритмами

В этом примере мы рассмотрим ситуацию, когда нужно перехватить обработчик события – например, выполнить до и/или после него какие-то свои алгоритмы.

Пусть в конфигурации существует справочник *Сотрудники* с реквизитами:

* *Подразделение* типа *СправочникСсылка.Подразделения*;
* *Должность* типа *Строка*;
* *Стаж* типа *Строка*;
* *Оклад* типа *Число*.

У справочника существует основная форма элемента, в которой для поля *Должность* включен режим выбора из списка. Список выбора программно формируется в обработчике события *ПриСозданииНаСервере* (листинг 2.7).

**Листинг 2.7.** Обработчик события «ПриСозданииНаСервере»

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

СписокДолжностей = Элементы.Должность.СписокВыбора;

СписокДолжностей.Вставить(0, "Должность1");

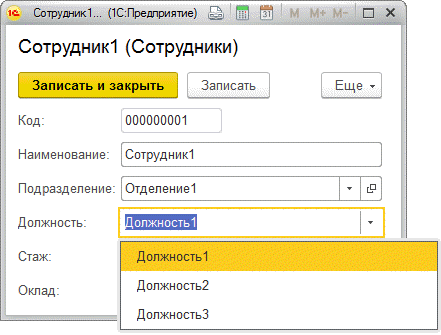
СписокДолжностей.Вставить(1, "Должность2");

СписокДолжностей.Вставить(2, "Должность3");

КонецПроцедуры

В этом обработчике наименования должностей вставляются в первые три позиции списка выбора.

В результате в режиме *1С:Предприятие* в форме сотрудника при нажатии кнопки выпадающего списка у поля *Должность* появляется следующий список выбора (рис. 2.32).

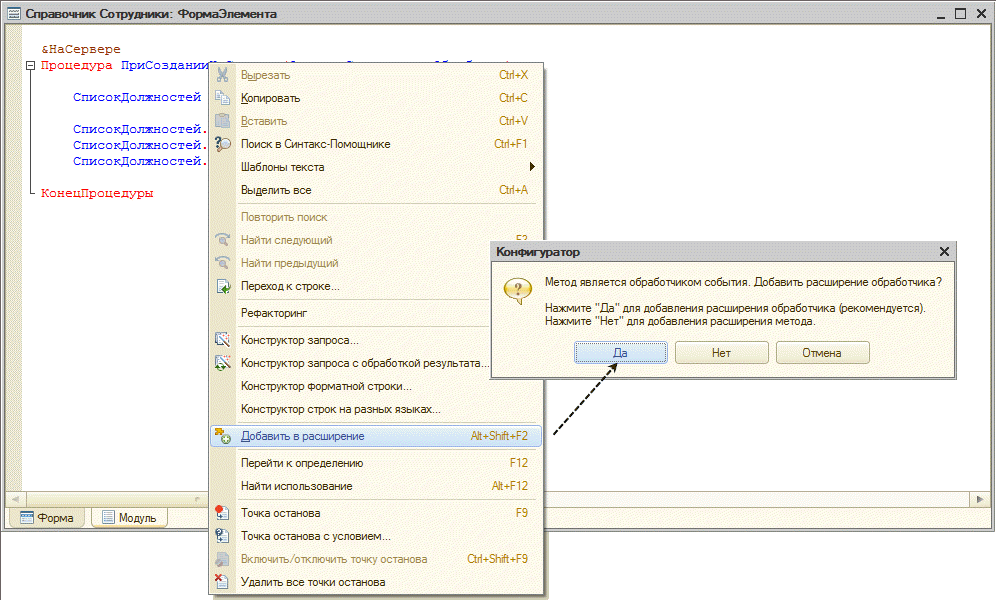


**Рис. 2.32.**Список выбора у поля «Должность» в форме сотрудника

Наша задача – перед заполнением списка выбора исходными значениями добавить еще две должности в этот список, а после заполнения списка выбора – удалить одну из строк списка.

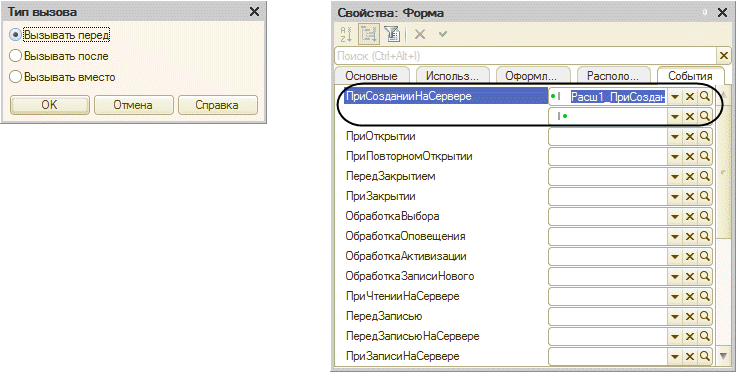
Для этого нам нужно в заимствованной форме справочника создать расширяющие обработчики *Перед* и *После*, перехватывающие исполнение обработчика события *ПриСозданииНаСервере* и выполняющиеся до и после этого обработчика. И затем заполнить их соответствующим образом.

Чтобы сделать это наиболее быстро и удобно, нужно открыть модуль формы конфигурации, установить курсор на имя обработчика события и вызвать из контекстного меню команду *Добавить в расширение*. После этого форма конфигурации заимствуется и появляется диалог с вопросом: «Метод является обработчиком события. Добавить расширение обработчика?» Ответим «Да» (рис. 2.33).



**Рис. 2.33.**Создание расширяющего метода обработчика события конфигурации

Затем заимствованная форма открывается и задается вопрос о типе расширяющего метода. Выберем тип *Перед*, так как мы хотим, чтобы создаваемый расширяющий метод выполнялся перед расширяемым обработчиком. В результате в палитре свойств формы появится привязка расширяющего метода к событию формы (рис. 2.34).



**Рис. 2.34.**Создание расширяющего метода «Перед»

Расширяющую процедуру *Расш1\_ПриСозданииНаСервереПеред* заполним следующим образом (листинг 2.8).

**Листинг 2.8.** Расширяющий метод «Перед» события «ПриСозданииНаСервере»

&НаСервере

Процедура Расш1\_ПриСозданииНаСервереПеред(Отказ, СтандартнаяОбработка)

СписокДолжностей = Элементы.Должность.СписокВыбора;

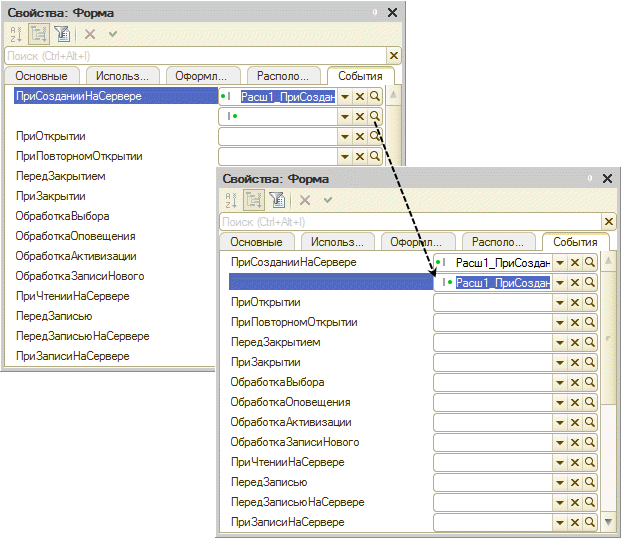
СписокДолжностей.Добавить("Бухгалтер");

СписокДолжностей.Добавить("Экономист");

КонецПроцедуры

В этой процедуре две новых должности добавляются в конец списка выбора.

Парный расширяющий метод *После* создадим с помощью палитры свойств расширяющей формы. Для этого нажмем на значок лупы в свободной строке события *ПриСозданииНаСервере* с пиктограммой типа вызова *После* – точка после черты. Указывать тип вызова для парного расширяющего метода не требуется: если уже назначен расширяющий метод *Перед*, то теперь будет *После*(рис. 2.35).



**Рис. 2.35.**Создание расширяющего метода «После»

Расширяющую процедуру *Расш1\_ПриСозданииНаСервереПосле* заполним следующим образом (листинг 2.9).

**Листинг 2.9.** Расширяющий метод «После» события «ПриСозданииНаСервере»

&НаСервере

Процедура Расш1\_ПриСозданииНаСервереПосле(Отказ, СтандартнаяОбработка)

СписокДолжностей = Элементы.Должность.СписокВыбора;

Должность = СписокДолжностей.НайтиПоЗначению("Должность2");

Если Должность <> Неопределено Тогда

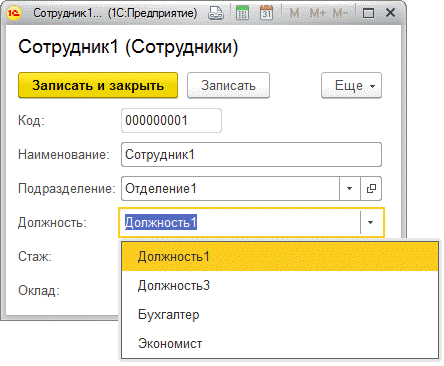
СписокДолжностей.Удалить(Должность);

КонецЕсли;

КонецПроцедуры

В этой процедуре позиция *Должность2* находится по значению в списке выбора. Если такая должность есть, она удаляется.

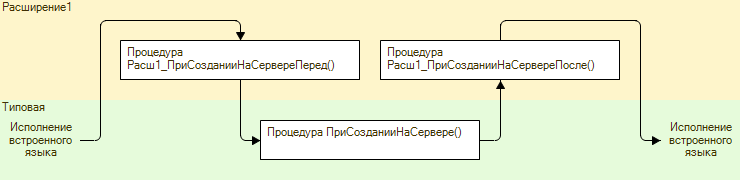
В результате при применении расширения к конфигурации в форме сотрудника при нажатии кнопки выпадающего списка у поля *Должность* появляется следующий список выбора (рис. 2.36).



**Рис. 2.36.**Список выбора у поля «Должность» в форме сотрудника

Этот список получается следующим образом. При создании формы элемента справочника вызывается обработчик события *ПриСозданииНаСервере*. Но он перехватывается расширяющим методом *Перед*, поэтому сначала выполняется процедура расширяющего метода *Расш1\_ПриСозданииНаСервереПеред* (см. листинг 2.8). В результате чего в конец списка выбора последовательно добавляются должности *Бухгалтер* и *Экономист*. Затем отрабатывает сам расширяемый обработчик (см. листинг 2.7). В результате в начало списка выбора вставляются позиции *Должность1*, *Должность2*, *Должность3*. И после него выполняется расширяющая процедура *После* *Расш1\_ПриСозданииНаСервереПосле* (см. листинг 2.9). В результате позиция *Должность2* удаляется из списка выбора.

Последовательность выполнения обработчиков можно представить на следующей схеме (рис. 2.37).



**Рис. 2.37.**Порядок выполнения обработчиков событий «ПриСозданииНаСервере»

#### Что произойдет, если в конфигурации изменится обработчик события

Предположим, что в новой версии конфигурации поставщик изменил обработчик события *ПриСозданииНаСервере* – например, добавил в конец списка выбора позицию *Должность5* (листинг 2.10).

**Листинг 2.10.** Расширяемый обработчик события «ПриСозданииНаСервере»

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

СписокДолжностей = Элементы.Должность.СписокВыбора;

СписокДолжностей.Вставить(0, "Должность1");

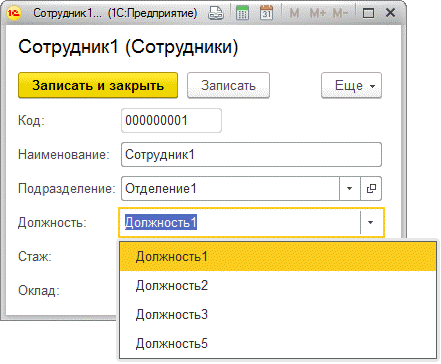
СписокДолжностей.Вставить(1, "Должность2");

СписокДолжностей.Вставить(2, "Должность3");

СписокДолжностей.Добавить("Должность5");

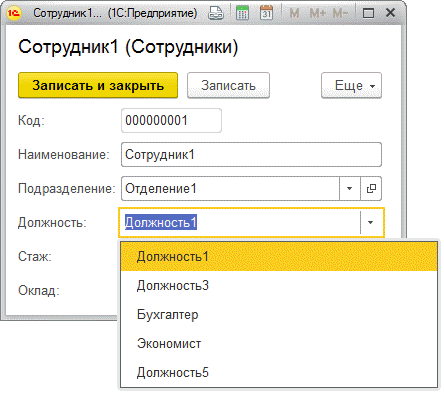
КонецПроцедуры

После запуска обновленной конфигурации в режиме *1С:Предприятие* без расширения в форме сотрудника при нажатии кнопки выпадающего списка у поля *Должность* появится следующий список выбора (рис. 2.38).



**Рис. 2.38.**Список выбора у поля «Должность» в форме сотрудника

После применения расширения к обновленной конфигурации результат изменится следующим образом (рис. 2.39).



**Рис. 2.39.**Список выбора у поля «Должность» в форме сотрудника

Список выбора формируется следующим образом. Сначала выполняется расширяющая процедура *Расш1\_ПриСозданииНаСервереПеред* (см. листинг 2.8). В результате в конец списка выбора последовательно добавляются должности *Бухгалтер* и *Экономист*. Затем отрабатывает сам расширяемый обработчик (см. листинг 2.10). В результате в начало списка выбора вставляются позиции *Должность1*, *Должность2*, *Должность3*, а позиция *Должность5* добавляется в самый конец списка выбора. И после него выполняется расширяющая процедура *После* *Расш1\_ПриСозданииНаСервереПосле* (см. листинг 2.9). В результате позиция *Должность2* удаляется из списка выбора.

#### Что произойдет, если к конфигурации будет применено еще одно расширение, перехватывающее тот же обработчик события

Предположим, что другой разработчик в другом расширении тоже создал расширяющие методы *Перед* и *После* того же обработчика события и заполнил их следующим образом (листинг 2.11, листинг 2.12).

**Листинг 2.11.** Расширяющий метод «Перед» события «ПриСозданииНаСервере»

&НаСервере

Процедура Расш2\_ПриСозданииНаСервереПеред(Отказ, СтандартнаяОбработка)

СписокДолжностей = Элементы.Должность.СписокВыбора;

СписокДолжностей.Добавить("Программист");

КонецПроцедуры

В этой процедуре в конец списка выбора добавляется позиция *Программист*.

**Листинг 2.12.** Расширяющий метод «После» события «ПриСозданииНаСервере»

&НаСервере

Процедура Расш2\_ПриСозданииНаСервереПосле(Отказ, СтандартнаяОбработка)

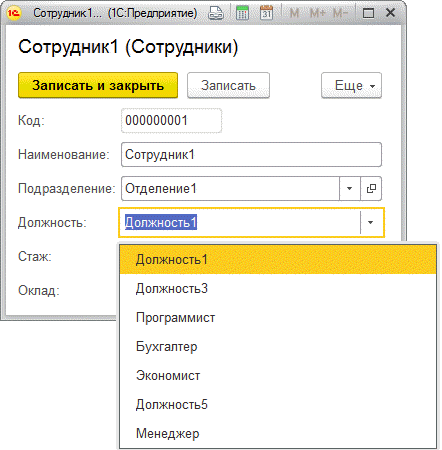
СписокДолжностей = Элементы.Должность.СписокВыбора;

СписокДолжностей.Добавить("Менеджер");

КонецПроцедуры

В этой процедуре в конец списка выбора добавляется позиция *Менеджер*.

Результат будет зависеть от порядка применения расширений. Если второе расширение применится последним, то в режиме *1С:Предприятие* в форме сотрудника при нажатии кнопки выпадающего списка у поля *Должность* появится следующий список выбора (рис. 2.40).



**Рис. 2.40.**Список выбора у поля «Должность» в форме сотрудника

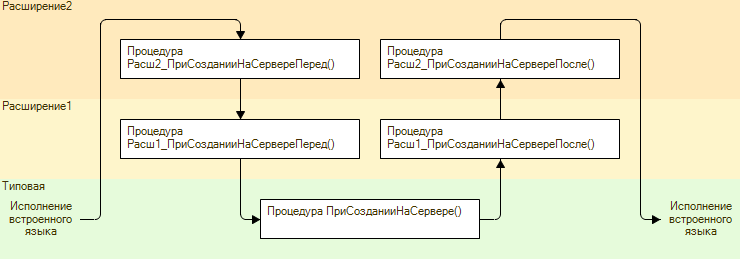
Порядок выполнения нескольких расширяющих методов с различными типами перехвата подробно рассматривается в разделе «[Модули – Взаимодействие нескольких расширений](#_Взаимодействие_нескольких_расширени)».

Сначала будет выполнен расширяющий метод *Перед* из второго расширения (см. листинг 2.11). В результате в конец списка выбора будет добавлена позиция *Программист*. Затем будет выполнен расширяющий метод *Перед* из первого расширения (см. листинг 2.8). В результате в конец списка выбора будут добавлены позиции *Бухгалтер* и *Экономист*.

Затем будет выполнен сам расширяемый обработчик (см. листинг 2.7). В результате в начало списка выбора будут вставлены позиции *Должность1*, *Должность2*, *Должность3*.

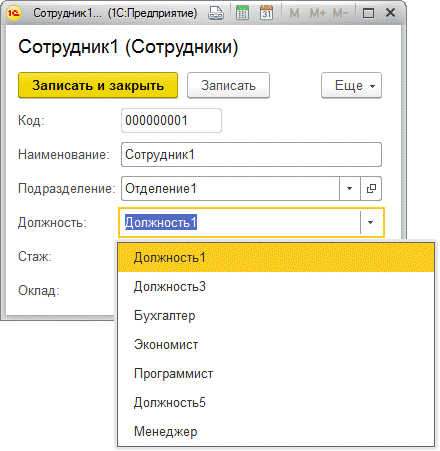
После расширяемого обработчика будет выполнен сначала расширяющий метод *После* из первого расширения (см. листинг 2.9). В результате позиция *Должность2* будет удалена из списка выбора. Затем будет выполнен расширяющий метод *После* из второго расширения (см. листинг 2.12). В результате в конец списка выбора будет добавлена позиция *Менеджер*.

Последовательность выполнения обработчиков можно представить на следующей схеме (рис. 2.41).



**Рис. 2.41.**Порядок выполнения обработчиков событий «ПриСозданииНаСервере»

Если же последним применится первое расширение, то список выбора будет иметь следующий вид (рис. 2.42).



**Рис. 2.42.**Список выбора у поля «Должность» в форме сотрудника

Подробный разбор формирования списка выбора вы в этом случае можете произвести самостоятельно. При этом нужно ориентироваться на следующий порядок выполнения процедур встроенного языка:

1. Расширяющий метод *Перед* из первого расширения.
2. Расширяющий метод *Перед* из второго расширения.
3. Расширяемый обработчик события.
4. Расширяющий метод *После* из второго расширения.
5. Расширяющий метод *После* из первого расширения.

### Как добавить в форму команду и создать ее обработчик

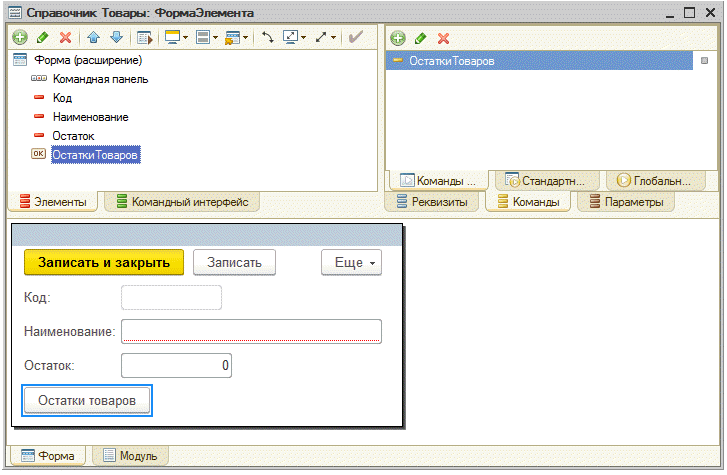
Командный интерфейс формы можно дополнить за счет собственных команд. Кроме того, можно перехватывать обработчики команд формы и тем самым изменять ее поведение.

Пусть в конфигурации существует справочник *Товары* с реквизитом *Остаток* типа *Число*.

Наша задача – поместить в форму элемента справочника команду для получения и отображения остатка по текущему товару. И если товара осталось мало, надо показать пользователю предупреждение об этом.

Для этого нам нужно позаимствовать форму товара в расширение, добавить в нее собственную команду и создать ее обработчик.

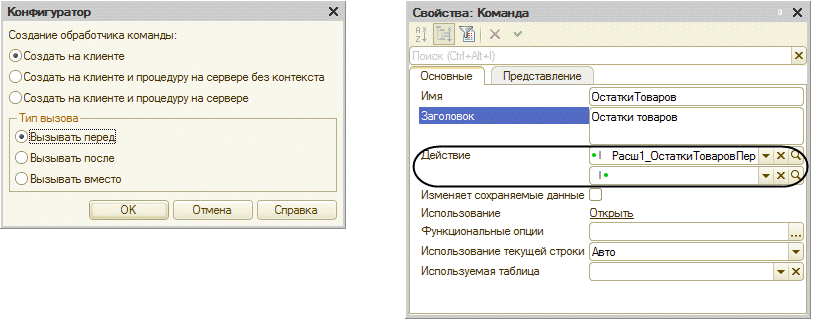
Позаимствуем форму, откроем ее, добавим команду *ОстаткиТоваров*и поместим ее в форму (рис. 2.43).



**Рис. 2.43.**Расширяющая форма товара в конфигураторе

Кроме того, добавим в форму декорацию-надпись *ДекорацияОстаток* и поместим над кнопкой *Остатки товаров*. Дадим декорации заголовок «Осталось менее трех штук товара!», установим для нее особый цвет текста и отключим ее видимость.

Затем создадим обработчик команды *ОстаткиТоваров*. При этом, так же как и для обработчиков событий в форме, нужно указать тип вызова *Перед*, *После* или *Вместо* расширяемого обработчика (рис. 2.44).



**Рис. 2.44.**Создание обработчика команды «ОстаткиТоваров» в расширяющей форме

Выберем тип вызова *Перед*, несмотря на то что расширяемого обработчика пока нет. Если позднее после обновления конфигурации в расширяемой форме появятся команда с именем *ОстаткиТоваров* и ее обработчик, то созданный нами расширяющий метод *Перед* будет перехватывать выполнение расширяемого обработчика команды и выполняться перед ним.

Процедуру *Расш1\_ОстаткиТоваровПеред()* заполним следующим образом (листинг 2.13).

**Листинг 2.13.** Расширяющий метод «ОстаткиТоваров» с типом перехвата «Перед»

&НаКлиенте

Процедура Расш1\_ОстаткиТоваровПеред(Команда)

Объект.Остаток = Расш1\_ОстаткиТоваров(Объект.Ссылка);

Модифицированность = Истина;

Если Объект.Остаток < 3 Тогда

Элементы.ДекорацияОстаток.Видимость = Истина;

Иначе

Элементы.ДекорацияОстаток.Видимость = Ложь;

КонецЕсли;

КонецПроцедуры

Для получения остатка по товару используется функция *Расш1\_ОстаткиТоваров()*, в которую передается ссылка на товар (листинг 2.14). И затем, в зависимости от полученного остатка, показывается или скрывается текст декорации для привлечения внимания пользователя.

**Листинг 2.14.** Функция «Расш1\_ОстаткиТоваров()»

&НаСервереБезКонтекста

Функция Расш1\_ОстаткиТоваров(ТоварСсылка)

Запрос = Новый Запрос;

Запрос.Текст =

"ВЫБРАТЬ

|        ОстаткиТоваров.КоличествоОстаток

| ИЗ

|        РегистрНакопления.ОстаткиТоваров.Остатки( , Товар = &Товар) КАК ОстаткиТоваров";

Запрос.УстановитьПараметр("Товар", ТоварСсылка);

Результат = Запрос.Выполнить();

Если Результат.Пустой() Тогда

Возврат 0;

КонецЕсли;

Остатки = Результат.Выбрать();

Остатки.Следующий();

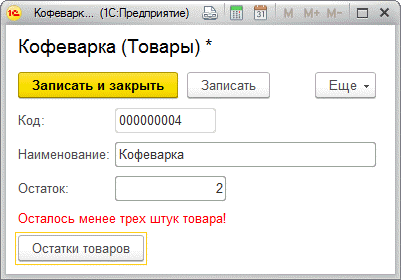
Остаток = Остатки.КоличествоОстаток;

Возврат Остаток;

КонецФункции

Обратите внимание, что в запросе для получения остатка товара используются регистр накопления *ОстаткиТоваров*, его измерение *Товар* и ресурс *Количество*. Их тоже надо заимствовать в расширение. Потому что, как мы уже объясняли ранее, существует простое правило: «Те метаданные, которые вы используете в коде, должны быть заимствованы».

В результате в режиме *1С:Предприятие*при нажатии кнопки *Остатки* *товаров* в форме товара в поле *Остаток* будет отображен актуальный остаток товара. И если товара осталось менее трех штук, пользователю будет показано соответствующее предупреждение (рис. 2.45).



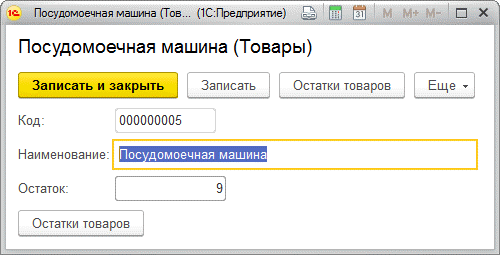
**Рис. 2.45.**Выполнение команды «ОстаткиТоваров» в форме товара

#### Что произойдет, если в конфигурации появится одноименная команда со своим обработчиком

Предположим, что в новой версии конфигурации поставщик тоже добавил в расширяемую форму команду с именем *ОстаткиТоваров*, но поместил ее не в саму форму, а в командную панель и создал обработчик этой команды. В нем программно формируется отчет *Остатки товаров* с отбором по текущему товару.

В результате после обновления конфигурации в режиме *1С:Предприятие* в форме товара появятся две одинаковых кнопки *Остатки* *товаров*. Одна кнопка с именем *ФормаОстаткиТоваров* находится в командной панели – эта кнопка попала сюда из расширяемой формы. Другая кнопка с именем *ОстаткиТоваров* находится внизу формы – эта кнопка попала сюда из расширяющей формы.

Кнопок получилось две, потому что команда была помещена в разные части формы, а по умолчанию платформа по-разному формирует имена элементов, которые помещаются в форму или в командную панель (рис. 2.46).



**Рис. 2.46.**Две одинаковые кнопки «Остатки товаров» в форме товара

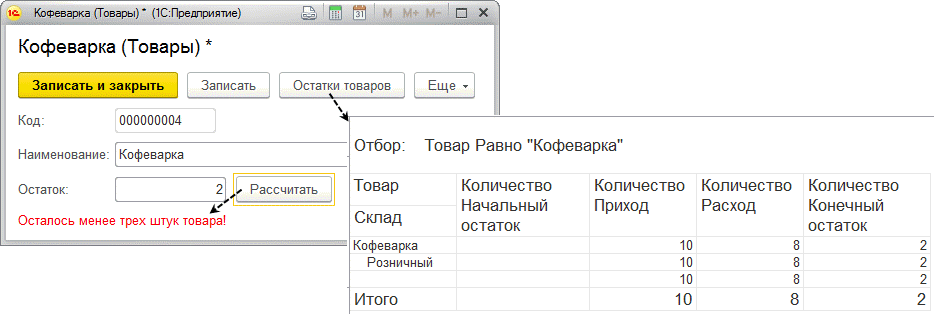
Но, несмотря на то что имена у кнопок разные, они связаны с одной и той же командой формы – *ОстаткиТоваров*. Поэтому при нажатии на каждую из этих кнопок сначала в поле *Остаток* будет отображен актуальный остаток товара с показом предупреждающего сообщения, а также будет открыт и сформирован отчет *Остатки* *товаров* с отбором по текущему товару.

При нажатии на любую из кнопок *Остатки товаров* вызывается расширяемый обработчик команды (формирующий отчет), но он перехватывается расширяющим методом *Перед* (рассчитывающим остаток) из расширения. Поэтому выполняется сначала расширяющий метод *Перед*, а затем расширяемый обработчик.

Получается путаница. Не только потому, что пользователь не будет знать, какой из кнопок ему пользоваться. Но и потому, что у расширяемого и у расширяющего обработчика команды совершенно разное назначение, а в каждом случае выполняются они оба.

В этом случае разработчику расширения нужно просто переименовать свою команду, так как расширяемые и расширяющие объекты связываются прежде всего по именам.

Дадим команде *ОстаткиТоваров* в расширяющей форме имя *Рассчитать*. Заголовок связанной с ней кнопки изменится автоматически в соответствии с заголовком команды. Поместим эту кнопку рядом с полем формы *Остаток*. В результате каждая из кнопок будет выполнять соответствующее ей действие, потому что теперь они связаны с разными командами (рис. 2.47).



**Рис. 2.47.**Выполнение команд «ОстаткиТоваров» и «Рассчитать» в форме товара

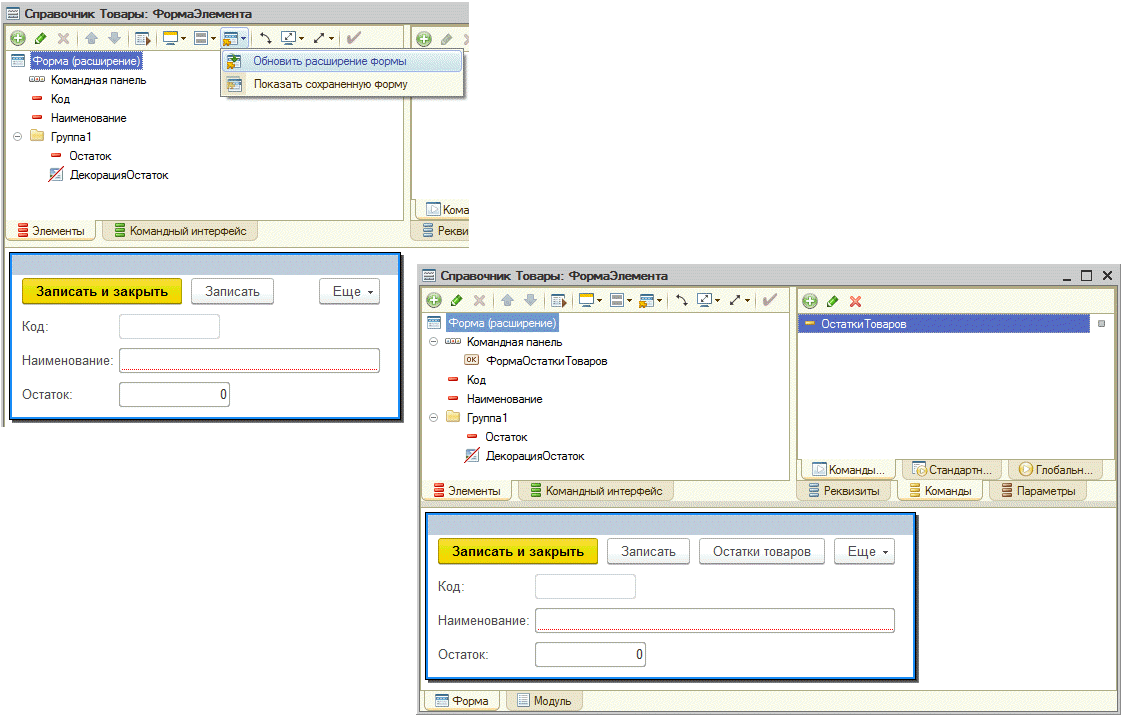
В случае если у расширяемого и у расширяющего обработчика команды одинаковая функциональность, разработчику расширения нужно просто удалить свою команду и ее обработчик.

А вот если расширяемый обработчик делает примерно то же самое, что и расширяющий, но хочется немного дополнить функциональность расширяемой команды, то нужно сначала удалить свою команду и ее обработчик из расширяющей формы. А затем обновить эту форму и доработать расширяемую команду так, как требуется.

Например, в расширяемом обработчике команды рассчитывается остаток по текущему товару и записывается в реквизит *Остаток*. Это то, что мы и хотели сделать в расширении, но кроме этого мы хотели еще вывести надпись, привлекающую внимание пользователя, в случае если товара осталось мало.

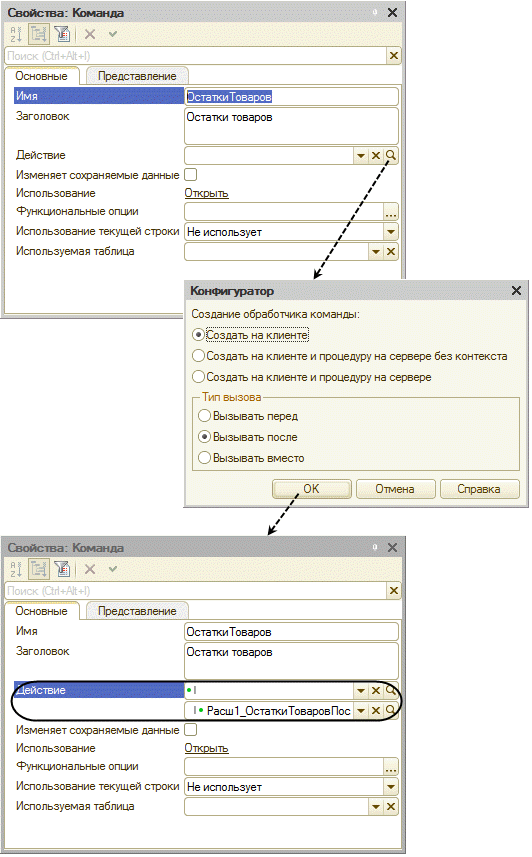
Поэтому расчет остатка мы будем выполнять в расширяемом обработчике, а затем в расширяющем методе *После* включать/отключать видимость предупреждающей надписи.

Удалим команду *Рассчитать*, связанную с ней кнопку и обработчик команды из расширяющей формы. Обновим форму командой *Расширения – Обновить расширение формы*. После этого команда *ОстаткиТоваров* снова попадет в форму, но это будет уже расширяемая команда (рис. 2.48).



**Рис. 2.48.**Обновление расширяющей формы в редакторе

Теперь создадим расширяющий метод *После* из палитры свойств этой команды (рис. 2.49).



**Рис. 2.49.**Создание расширяющего метода «После»

В расширяющую процедуру *Расш1\_ОстаткиТоваровПосле()* поместим только код для отображения декорации (листинг 2.15).

**Листинг 2.15.** Расширяющий метод «После» обработчика команды «ОстаткиТоваров»

Процедура Расш1\_ОстаткиТоваровПосле(Команда)

Если Объект.Остаток < 3 Тогда

Элементы.ДекорацияОстаток.Видимость = Истина;

Иначе

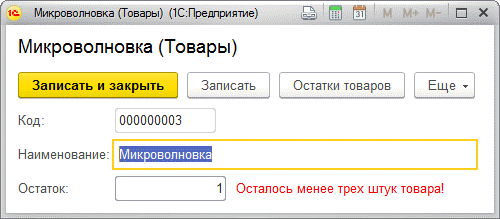
Элементы.ДекорацияОстаток.Видимость = Ложь;

КонецЕсли;

КонецПроцедуры

Заимствованный регистр можно удалить из расширения, так как получение остатка с помощью запроса, в котором используется регистр, выполняется теперь в конфигурации.

В результате в режиме *1С:Предприятие* при нажатии кнопки *Остатки* *товаров* в форме товара выполняется сначала расширяемый обработчик команды – после этого в поле *Остаток* будет отображен актуальный остаток товара. Затем выполняется расширяющий метод *После* из расширения. В результате, если товара осталось менее трех штук, пользователю будет показано соответствующее предупреждение (рис. 2.50).



**Рис. 2.50.**Выполнение команды «ОстаткиТоваров» в форме товара

#### Что произойдет, если к конфигурации будет применено еще одно расширение, перехватывающее тот же обработчик команды

Предположим, что другой разработчик в другом расширении с назначением *Дополнение* тоже решил доработать эту же команду. Например, ему хочется, чтобы при нажатии кнопки *Остатки* *товаров* в форме товара не только рассчитывался актуальный остаток товара, но и, в случае если товара осталось мало, пользователю предлагалось бы зарезервировать товар.

Для этого он позаимствовал форму товара, добавил в нее собственную команду *Зарезервировать*, поместил ее в форму в виде гиперссылки и отключил ее видимость. Затем он создал расширяющий метод *После* обработчика команды *ОстаткиТоваров*, в котором программно устанавливается видимость гиперссылки в зависимости от остатка товара (листинг 2.16).

**Листинг 2.16.** Расширяющий метод «После» обработчика команды «ОстаткиТоваров»

Процедура Расш2\_ОстаткиТоваровПосле(Команда)

Если Объект.Остаток <= 5 Тогда

Элементы.Зарезервировать.Видимость = Истина;

Иначе

Элементы.Зарезервировать.Видимость = Ложь;

КонецЕсли;

КонецПроцедуры

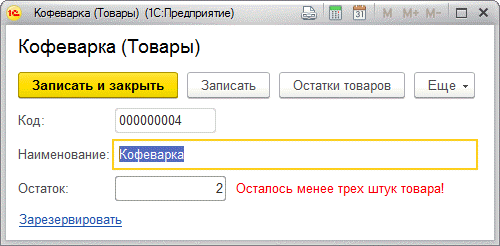
В соответствии со своим назначением в режиме *1С:Предприятие* к конфигурации применится сначала первое расширение, а затем второе.

Порядок выполнения нескольких расширяющих методов с различными типами перехвата подробно рассматривается в разделе «[Модули – Взаимодействие нескольких расширений](#_Взаимодействие_нескольких_расширени)».

В результате при нажатии кнопки *Остатки* *товаров* в форме товара выполняется сначала расширяемый обработчик команды – после этого в поле *Остаток* будет отображен актуальный остаток товара.

Затем выполняется расширяющий метод *После* из первого расширения. В результате, если товара осталось менее трех штук, пользователю будет показано соответствующее предупреждение.

Затем выполняется расширяющий метод *После* из второго расширения. В результате, если остаток товара меньше или равен пяти, пользователю будет показана гиперссылка, при нажатии на которую товар будет зарезервирован (рис. 2.51).



**Рис. 2.51.**Выполнение команды «ОстаткиТоваров» в форме товара

## Отчеты и обработки

В расширении можно создавать новые отчеты и обработки таким же образом, как это делалось в конфигурации, не использующей расширения. В большинстве случаев объекты данных, используемые в этих отчетах и обработках, нужно заимствовать вручную.

Кроме того, вы можете доработать формы отчетов и обработок так же, как это было показано выше в разделе [«Формы»](#_Формы).

### Как создать собственный отчет

Предположим, пользователю прикладного решения понадобился еще один новый отчет, содержащий информацию о том, продажа каких товаров принесла предприятию наибольшую выручку.

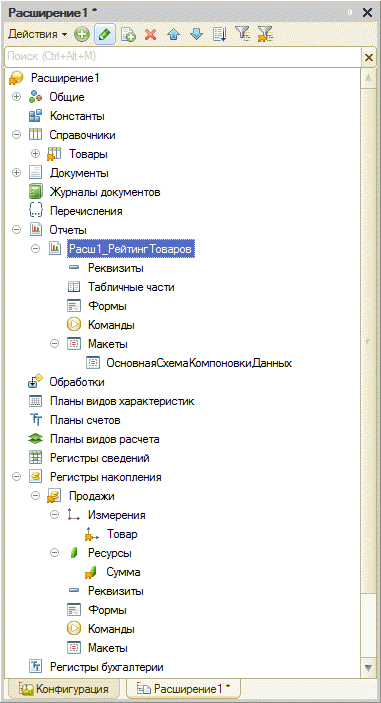
Для этого нам нужно добавить отчет в расширение, заимствовать объекты, содержащие данные, на которых будет строиться этот отчет, и создать для него основную схему компоновки данных.

Информация о продажах хранится в оборотном регистре накопления *Продажи*. Поэтому нам нужно этот регистр, вместе с его измерением *Товар* и ресурсом *Сумма*, заимствовать в расширение. А также для отчета нам понадобится справочник *Товары*, так как мы хотим в запросе схемы компоновки данных связать его левым соединением с регистром накопления.

Поскольку при заимствовании объектов в расширение подчиненные объекты конфигурации не заимствуются, то проще и удобней всего начать с самого «низа».

Сначала позаимствуем из регистра измерение *Товар*. Одновременно с ним будет заимствован сам регистр, а также справочник *Товары* (так как измерение регистра имеет тип ссылки на этот справочник). Осталось позаимствовать из регистра ресурс *Сумма*.

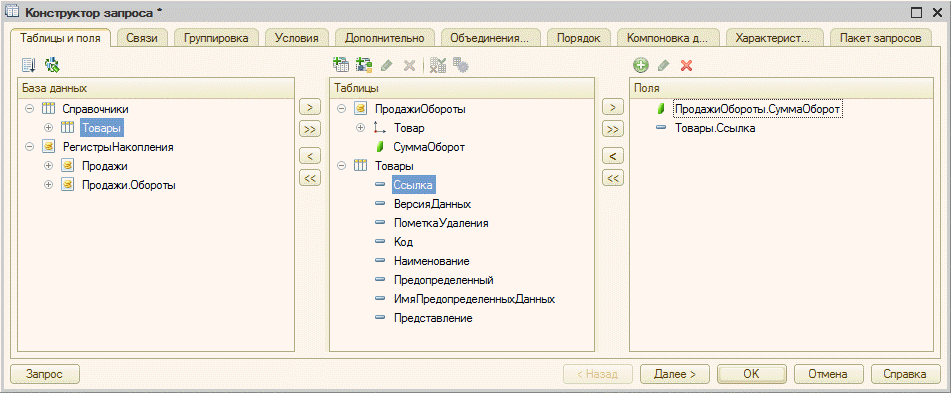
И все данные для нашего отчета в расширении уже есть. Теперь добавим сам отчет *Расш1\_РейтингТоваров* и создадим его основную схему компоновки данных (рис. 2.52).



**Рис. 2.52.**Новый отчет «РейтингТоваров» и заимствованные объекты данных для отчета

В конструкторе схемы компоновки данных создадим *Набор данных – запрос* и вызовем конструктор запроса. На закладке *Таблицы и поля* мы видим только те объекты, которые мы позаимствовали. Перенесем в список таблиц – источников данных для отчета виртуальную таблицу регистра *Продажи.Обороты* и таблицу справочника *Товары*. Выберем из этих таблиц следующие поля (рис. 2.53):

* *ПродажиОбороты.СуммаОборот*;
* *Товары.Ссылка*.



**Рис. 2.53.**Построение запроса для схемы компоновки данных отчета «РейтингТоваров» с помощью конструктора запроса

Обратите внимание, что у таблицы регистра не отображаются измерение *Покупатель* и ресурс *Количество*, так как мы их не заимствовали, потому что они не нужны в данном отчете.

Таблицу справочника левым соединением свяжем с таблицей регистра по товару. Зададим псевдонимы *Сумма* и *Товар* полям запроса. Упорядочим по убыванию суммы. В результате конструктор запроса сформирует следующий запрос (листинг 2.17).

**Листинг 2.17.** Запрос для получения данных для отчета «РейтингТоваров»

ВЫБРАТЬ

ПродажиОбороты.СуммаОборот КАК Сумма,

Товары.Ссылка КАК Товар

ИЗ

Справочник.Товары КАК Товары

ЛЕВОЕ СОЕДИНЕНИЕ

РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты

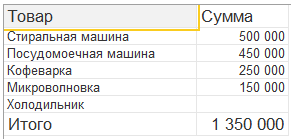
ПО ПродажиОбороты.Товар = Товары.Ссылка

УПОРЯДОЧИТЬ ПО

Сумма УБЫВ

Создадим ресурс *Сумма*, в выбранные поля добавим *Товар* и *Сумма*, а выводить будем только детальные записи.

В результате в режиме *1С:Предприятие*отчет *РейтингТоваров* будет отображать итоговые суммы продаж по товарам в порядке убывания, включая и те товары, которые не были проданы ни разу (рис. 2.54).



**Рис. 2.54.**Отчет «РейтингТоваров» в режиме «1С:Предприятие»

#### Что произойдет, если в конфигурации появится одноименный отчет

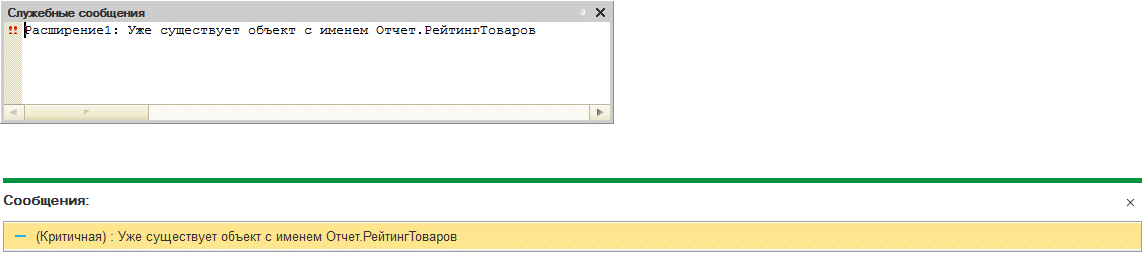
Как уже неоднократно говорилось, при добавлении собственных объектов в расширение перед именем объекта по умолчанию добавляется префикс, установленный при создании расширения (в нашем случае «Расш1\_»). Этот префикс настоятельно рекомендуется использовать во избежание дублирования имен с объектами конфигурации.

Но может случиться, что по ошибке или невнимательности этот префикс из имени пропадет. Например, если разработчик начал писать имя отчета и случайно затер префикс «Расш1\_» – таким образом имя отчета *РейтингТоваров* оказалось без префикса.

Если в конфигурации уже есть отчет с таким именем, то платформа сразу же после ввода имени отчета в расширении сообщит: «Имя не уникально!» – и не даст вам использовать это имя.

Но возможна ситуация, когда отчет уже был добавлен в расширение, а затем поставщик в новой версии конфигурации добавил отчет с таким же именем *РейтингТоваров*. Что делает этот отчет – то же самое или что-то другое по сравнению с отчетом в расширении, – неважно.

Важно то, что расширение не применится к обновленной конфигурации. И при записи расширения в информационную базу в конфигураторе, и при загрузке расширения в режиме *1С:Предприятие* вы увидите сообщение о невозможности применения расширения к конфигурации (рис. 2.55).



**Рис. 2.55.**Сообщение об ошибке применения расширения к конфигурации

#### Что произойдет, если к конфигурации будет применено еще одно расширение, с одноименным отчетом

Предположим, что другой разработчик в другом расширении тоже добавил отчет с именем *РейтингТоваров*. И оно применяется к конфигурации последним.

В этом случае к конфигурации сможет примениться только первое расширение. Второе расширение не применится по причине: «Уже существует объект с именем *Отчет.РейтингТоваров*». Это сообщение будет видно в окне информации для технической поддержки.

Соответственно, если назначение расширений будет таким, что первое расширение будет применяться к конфигурации последним, то тогда уже оно не сможет примениться по аналогичной причине.

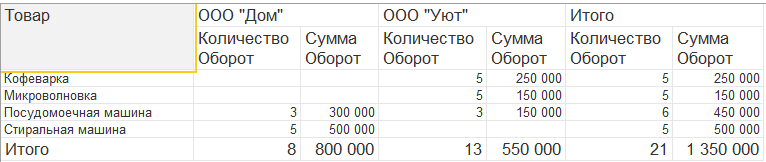
### Как доработать отчет

Доработать отчет конфигурации можно двумя способами:

* Заимствовать отчет и создать для него собственную схему компоновки данных.
* Доработать исходную схему компоновки данных отчета. Для этого ее нужно скопировать (не заимствовать!) из конфигурации, назначить основной для заимствованного отчета и изменить так, как вам нужно.

Рассмотрим подробнее оба варианта.

Предположим, что в прикладном решении у пользователя уже есть отчет *Продажи*, представляющий данные о продажах товаров покупателям в виде таблицы, в которой по строкам расположены товары, а по столбцам покупатели (рис. 2.56).

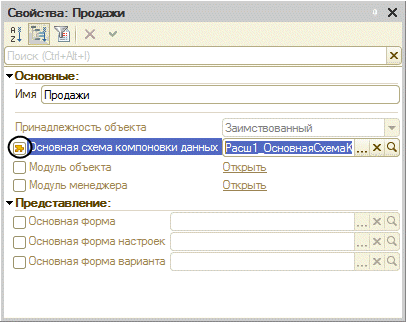


**Рис. 2.56.**Отчет «Продажи» в конфигурации

Но пользователь хочет в этом отчете видеть другие данные, по-другому сгруппированные и расположенные.

Для решения этой задачи нужно позаимствовать отчет и создать для него собственную схему компоновки данных. Позаимствуем отчет *Продажи*.

Установим флажок модифицированности значения у свойства отчета *Основная схема компоновки данных* и создадим схему компоновки (рис. 2.57).



**Рис. 2.57.**Создание собственной схемы компоновки данных для отчета «Продажи»

Нам нужно, чтобы отчет *Продажи* просто выводил список расходных накладных, сгруппированный по покупателям, с подсчетом итоговых сумм продаж по товарам и покупателям.

В качестве источника данных для этого отчета нам понадобится таблица документа *РасходнаяНакладная*. Позаимствуем реквизит его табличной части *Товар*. Одновременно будут заимствованы сам документ и его табличная часть *Товары*. Позаимствуем еще реквизит документа *Контрагент* и реквизит его табличной части *Сумма*.

Теперь в конструкторе схемы компоновки данных создадим *Набор данных – запрос* и сконструируем запрос для получения данных (листинг 2.18).

**Листинг 2.18.** Запрос для получения данных для отчета «Продажи»

ВЫБРАТЬ

РасходнаяНакладная.Дата КАК Дата,

РасходнаяНакладная.Номер КАК Номер,

РасходнаяНакладная.Контрагент КАК Контрагент,

РасходнаяНакладная.Товары.(

НомерСтроки КАК НомерСтроки,

Товар КАК Товар,

Сумма КАК Сумма) КАК Товары

ИЗ

Документ.РасходнаяНакладная КАК РасходнаяНакладная

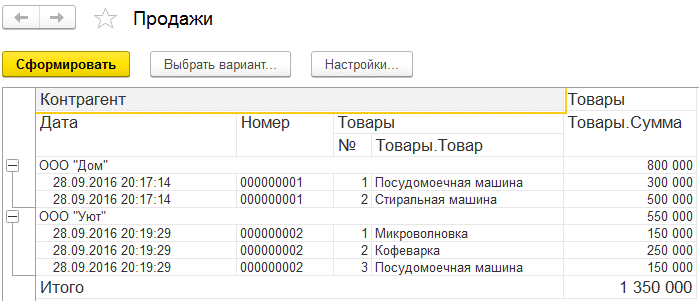
УПОРЯДОЧИТЬ ПО

Дата,

Номер

Создадим ресурс *Товары.Сумма*. В структуру отчета добавим группировку *Контрагент*, а в нее – детальные записи. В выбранные поля добавим *Дата*, *Номер* и таблицу *Товары*. Переименуем поля папки *Товары* в*№*, *Товар* и *Сумма*.

В результате в режиме *1С:Предприятие* отчет *Продажи* будет отображать итоговые суммы продаж по покупателям в разрезе товаров с выводом дат и номеров расходных накладных (рис. 2.58).



**Рис. 2.58.**Измененный отчет «Продажи» в режиме «1С:Предприятие»

Теперь рассмотрим второй вариант доработки отчета. Этот вариант используется, когда нужный отчет можно получить на тех же данных, что и расширяемый, и требуется, например, немного изменить его настройки.

Предположим, пользователь хочет только немного изменить внешний вид отчета *Продажи*, представляющего данные о продажах товаров покупателям в виде таблицы (см. рис. 2.56), или добавить новый вариант этого же отчета.

В этом случае нужно позаимствовать отчет, затем скопировать основную схему компоновки данных отчета из конфигурации через буфер обмена (*Ctrl + C / Ctrl + V*), доработать ее и назначить в качестве основной схемы компоновки данных для заимствованного отчета.

**ПРИМЕЧАНИЕ**

*Схему компоновки данных надо именно копировать, а не заимствовать в расширение, так как макеты схем компоновки не входят в список расширяемых объектов конфигурации.*

*Вообще заимствование схемы компоновки данных может понадобиться в том случае, когда расширение использует эту схему (без изменений) для программной модификации и программного исполнения. Но надо иметь в виду, что заимствование схемы компоновки происходит только по имени, без содержимого, и больше ничего с ней делать вы не сможете.*

Итак, позаимствуем отчет *Продажи*. Скопируем его схему компоновки данных из конфигурации и вставим ее в ветвь макетов заимствованного отчета. Назначим ее основной схемой отчета.

Источником для получения данных для отчета служит простой запрос, выбирающий все данные из таблицы *ПродажиОбороты* (листинг 2.19).

**Листинг 2.19.** Запрос для получения данных для отчета «Продажи»

ВЫБРАТЬ

ПродажиОбороты.Товар КАК Товар,

ПродажиОбороты.Покупатель КАК Покупатель,

ПродажиОбороты.КоличествоОборот КАК КоличествоОборот,

ПродажиОбороты.СуммаОборот КАК СуммаОборот

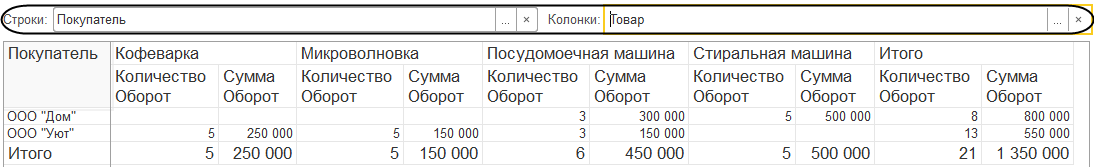
ИЗ

РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты

Теперь нам нужно позаимствовать регистр накопления *Продажи* вместе с его измерениями *Товар* и *Покупатель* и ресурсами *Количество* и *Сумма*.

Итак, позаимствуем регистр и откроем скопированную схему компоновки данных. Для элемента структуры отчета *Таблица* включим в состав быстрых пользовательских настроек отчета *Группировки строк* и *Группировки колонок*.

В результате в режиме *1С:Предприятие* отчет *Продажи* будет по-прежнему отображаться в виде таблицы, но непосредственно в форме отчета пользователь может поменять местами строки и столбцы отчета (рис. 2.59).

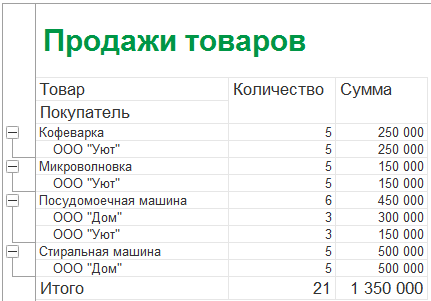


**Рис. 2.59.**Отчет «Продажи» в режиме «1С:Предприятие»

А теперь, ничего не изменяя в расширяемом отчете, просто добавим новый вариант заимствованного отчета *Продажи*. Для этого снова скопируем схему компоновки данных отчета из конфигурации и вставим ее в ветвь макетов заимствованного ранее отчета. Назначим ее основной схемой отчета.

Откроем ее и добавим новый вариант отчета *ПродажиТоваров*. В структуру отчета добавим группировку *Товар*, а в нее – группировку *Покупатель*. В выбранные поля добавим *КоличествоОборот* (с заголовком *Количество*) и *СуммаОборот* (с заголовком *Сумма*). Назовем отчет *Продажи товаров*.

В результате в режиме *1С:Предприятие* для отчета *Продажи* появится новый вариант отчета *Продажи товаров* (рис. 2.60).



**Рис. 2.60.**Вариант «Продажи товаров» отчета «Продажи» в режиме «1С:Предприятие»

При этом старый вариант отчета *Продажи* в виде таблицы продаж товаров покупателям останется без изменений, и пользователь прикладного решения им также сможет пользоваться.

#### Что произойдет, если основная схема компоновки данных отчета изменится в конфигурации

Предположим, что в новой версии конфигурации поставщик изменил основную схему компоновки данных отчета *Продажи* или назначил в качестве основной совсем другую схему компоновки.

После применения расширения к обновленной конфигурации в режиме *1С:Предприятие* для отчета по-прежнему будет использоваться наша, доработанная в расширении, схема компоновки данных. Потому что у свойства отчета *Основная схема компоновки данных*нами установлен флажок модифицируемости. Это значит, что значение этого свойства в результате будет браться из расширения.

И это не зависит от того, каким из способов, рассмотренных выше, мы модифицировали отчет.

#### Что произойдет, если к конфигурации будет применено еще одно расширение, устанавливающее основную схему компоновки данных этого же отчета

Предположим, другой разработчик в другом расширении тоже заимствовал отчет *Продажи*, скопировал его схему компоновки данных, доработал ее и назначил основной схемой для отчета.

Поскольку у свойства отчета *Основная схема компоновки данных*в обоих расширениях установлен флажок модифицируемости, то, какая форма будет показана в режиме *1С:Предприятие*, будет зависеть непосредственно от порядка применения расширений.

Если к конфигурации сначала будет применено первое расширение, а затем – второе, то в качестве основной схемы компоновки данных будет использоваться схема из второго расширения. Если же последним применится первое расширение, то будет использоваться схема из первого расширения.

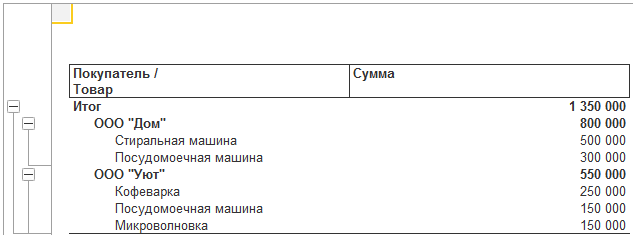
## Как доработать макет табличного документа

В расширении можно дорабатывать внешний вид макета табличного документа, при этом не меняя состав данных, на которых он строится. Этот процесс во многом аналогичен доработке форм.

Но, в отличие от расширения форм, при доработке макета табличного документа можно указать способ расширения *Объединять* или *Замещать*. Тем самым определяется, каким образом будет расширяться табличный документ: объединением расширяемой таблицы и таблицы из расширения или таблица расширения будет целиком замещать собой расширяемый табличный документ. Мы будем рассматривать вариант объединения изменений макета табличного документа в расширении с расширяемым макетом или с изменениями макета в другом расширении.

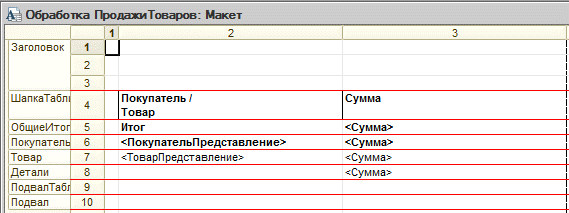
После заимствования макета табличного документа его копия сохраняется в расширении. При получении макета в режиме *1С:Предприятие* анализируются доработки макета конфигурации и доработки макета в расширении по сравнению с сохраненным макетом табличного документа. Сопоставление изменений выполняется на уровне именованных строк, отдельных строк и отдельных ячеек. Если эти доработки пересекаются, то применяется расширение.

Рассмотрим пример. Пусть в конфигурации существует обработка *Продажи товаров*, в которой в табличный документ выводятся суммы продаж по покупателям в разрезе товаров (рис. 2.61).



**Рис. 2.61.**Обработка «Продажи товаров»

Табличный документ заполняется данными на основе следующего макета (рис. 2.62).



**Рис. 2.62.**Макет табличного документа в конфигурации

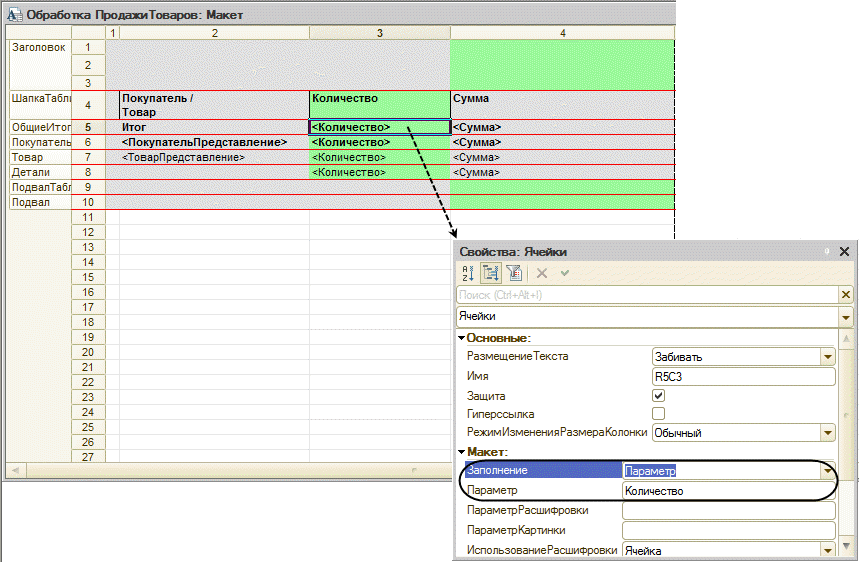
Наша задача – доработать внешний вид табличного документа:

* Добавить в макет колонку с данными.
* Изменить шапку документа.
* Изменить форматирование отдельных ячеек макета.

Для этого нам нужно позаимствовать макет табличного документа и изменить его в соответствии с поставленной задачей.

Позаимствуем макет. Добавим в него новую колонку *Количество*. Соответствующее поле получается в запросе для заполнения табличного документа, но в расширяемом макете эти данные не выводятся в табличный документ.

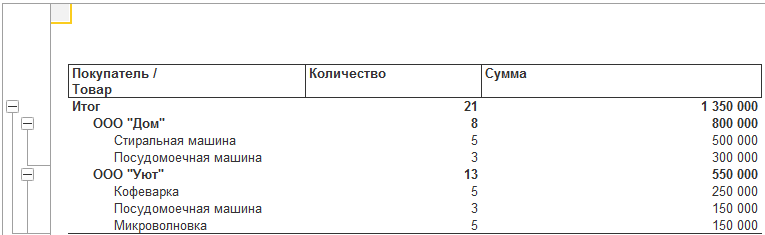
Для удобства включим режим отображения изменений в расширяемом макете с помощью команды главного меню *Таблица – Расширение – Отображать изменения* (рис. 2.63).



**Рис. 2.63.**Расширяющий макет табличного документа

Серым цветом в расширяющем макете отображены ячейки, которые не изменились по сравнению с сохраненным макетом, а зеленым – новые добавленные ячейки.

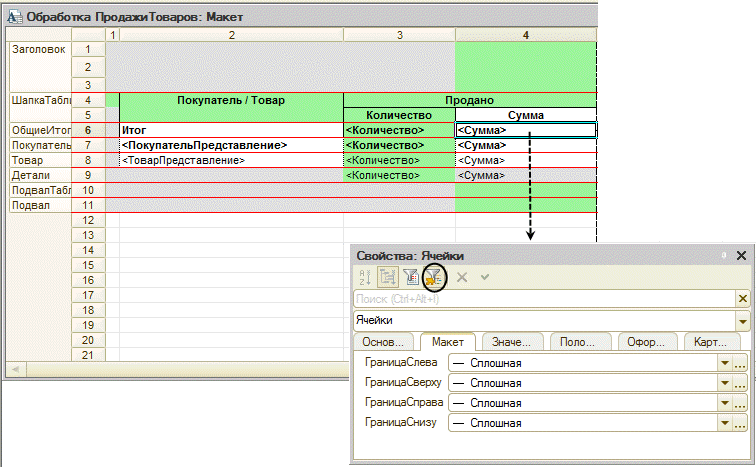
В результате в режиме *1С:Предприятие* в обработке *Продажи товаров* будет сформирован следующий табличный документ (рис. 2.64).



**Рис. 2.64.**Обработка «Продажи товаров»

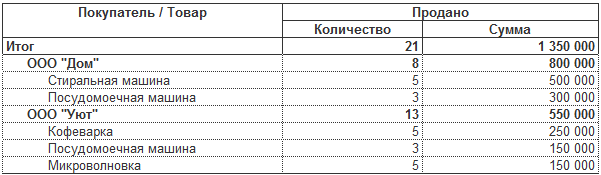
Теперь изменим форматирование отдельных ячеек и изменим шапку таблицы так, чтобы вид табличного документа больше напоминал печатную форму.

Белым цветом в расширяющем макете отображены ячейки, которые были изменены по сравнению с сохраненным макетом. Чтобы посмотреть, какие именно свойства этих ячеек изменились, нужно нажать кнопку *Показывать только измененные* в командной панели палитры свойств ячейки (рис. 2.65).



**Рис. 2.65.**Расширяющий макет табличного документа

В результате в режиме *1С:Предприятие* в обработке *Продажи товаров* будет сформирован следующий табличный документ (рис. 2.66).

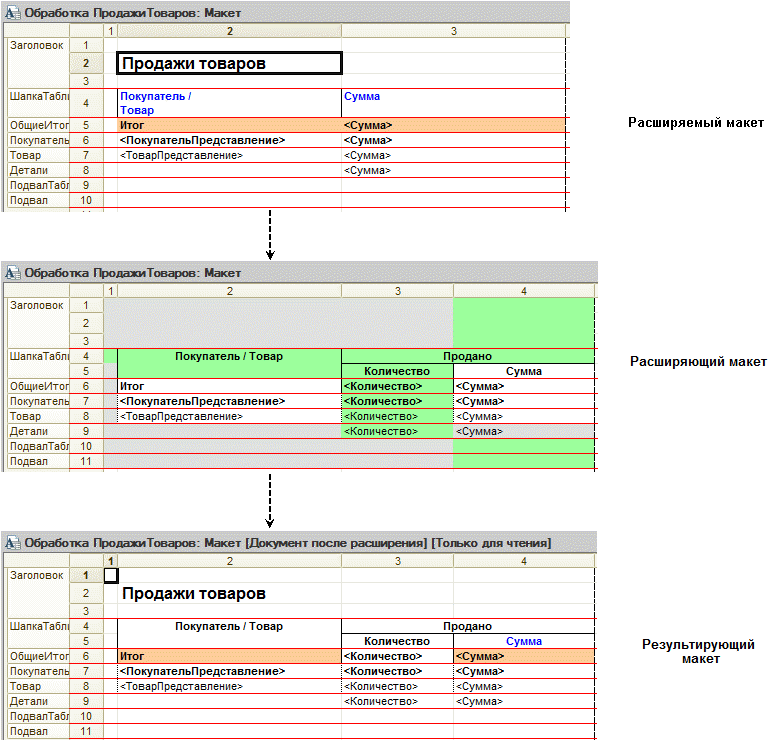


**Рис. 2.66.**Обработка «Продажи товаров»

### Что произойдет, если макет табличного документа изменится после обновления конфигурации

Предположим, в новой версии конфигурации поставщик изменил макет табличного документа. Например, он внес чисто косметические изменения: добавил в макет заголовок, установил синий цвет текста для ячеек шапки, а также установил коралловый фон для итоговой строки.

На следующем рисунке отображены обновленный макет конфигурации и расширяющий макет табличного документа. Под ними находится макет, который будет получен в результате сопоставления изменений расширяемого и расширяющего макета по сравнению с сохраненным макетом. Этот результат можно увидеть еще на этапе разработки с помощью команды главного меню *Таблица – Расширение – Показать результат* (рис. 2.67).



**Рис. 2.67.**Расширяемый, расширяющий и результирующий макеты табличного документа

В результате в режиме *1С:Предприятие* в обработке *Продажи товаров* будет сформирован следующий табличный документ (рис. 2.68).



**Рис. 2.68.**Обработка «Продажи товаров»

Этот результат можно было предсказать. Заголовок документа «Продажи товаров» установлен только в расширяемом документе, поэтому он отображается в результате.

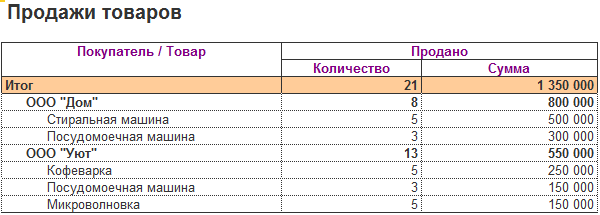
Цвет текста изменен на синий в соответствии с расширяемым макетом, но только у ячейки шапки таблицы с заголовком «Сумма». Потому что остальные ячейки шапки, в связи с построением двухуровневой шапки таблицы в расширяющем макете, считаются добавленными (они отображены зеленым цветом на рис. 2.67).

Цвет фона итоговой строки также взят из расширяемого макета, но ячейка с параметром *Количество* им не закрашена, потому что эта ячейка – новая в расширяющем макете.

Все остальные изменения взяты из расширяющего макета.

Увидев результат работы своего расширения после обновления конфигурации, разработчик может легко исправить эти косметические дефекты. Например, он может задать для всех ячеек шапки таблицы фиолетовый цвет текста и для ячейки итоговой строки с параметром *Количество* коралловый цвет фона.

В результате в режиме *1С:Предприятие* в обработке *Продажи товаров* будет сформирован следующий табличный документ (рис. 2.69).



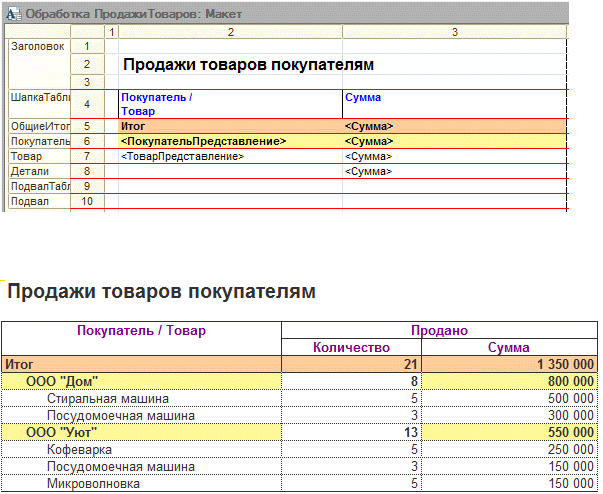
**Рис. 2.69.**Обработка «Продажи товаров»

Как вы видите, у ячеек шапки таблицы фиолетовый цвет текста, так как изменения расширяющего макета накладываются на изменения этого же свойства (синий цвет текста) в расширяемом макете.

### Что произойдет, если к конфигурации будет применено еще одно расширение, изменяющее тот же макет табличного документа

Предположим, другой разработчик в другом расширении тоже доработал этот макет табличного документа. Например, он изменил заголовок макета на «Продажи товаров покупателям» и установил желтый цвет фона для группировки *Покупатель*.

Причем в соответствии со своим назначением в режиме *1С:Предприятие* это (второе) расширение применяется к конфигурации первым, а затем применяется разработанное нами ранее (первое) расширение. В результате в обработке *Продажи товаров* будет сформирован следующий табличный документ (рис. 2.70).



**Рис. 2.70.**Обработка «Продажи товаров»

То есть мы видим, что на расширяемый макет табличного документа сначала наложены изменения второго расширения, а затем на получившийся результат наложены изменения первого расширения.

## Зачем заимствовать предопределенные данные

Чтобы иметь возможность обращаться к предопределенному элементу объекта (например, справочника) из встроенного языка или использовать его в языке запросов, нужно заимствовать в расширение не только сам справочник, но и его предопределенный элемент. При заимствовании нужно согласиться с предложением платформы установить необходимость контроля состава предопределенных элементов этого справочника. И тогда при применении расширения будет проверяться наличие предопределенных элементов с такими же именем, кодом и наименованием в справочнике конфигурации.

Например, в конфигурации существует справочник *Склады*, имеющий предопределенный элемент *Офисный*. А также у справочника *Товары* существует форма списка, основанная на динамическом списке, который получает данные из основной таблицы справочника.

Наша задача – доработать форму списка так, чтобы она получала данные для динамического списка с помощью произвольного запроса, в котором используется предопределенный элемент справочника.

Для этого нам нужно позаимствовать форму списка в расширение и установить нужный запрос у динамического списка.

Позаимствуем форму списка товаров. В свойствах динамического списка установим следующий произвольный запрос для получения данных списка (листинг 2.20).

**Листинг 2.20.** Запрос для получения данных динамического списка

ВЫБРАТЬ

СправочникТовары.Ссылка КАК Ссылка,

СправочникТовары.Код КАК Код,

СправочникТовары.Наименование КАК Наименование,

ОстаткиТоваровОстатки.КоличествоОстаток КАК Остаток,

ОстаткиТоваровОстатки.Склад КАК Склад

ИЗ

Справочник.Товары КАК СправочникТовары

ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки( ,

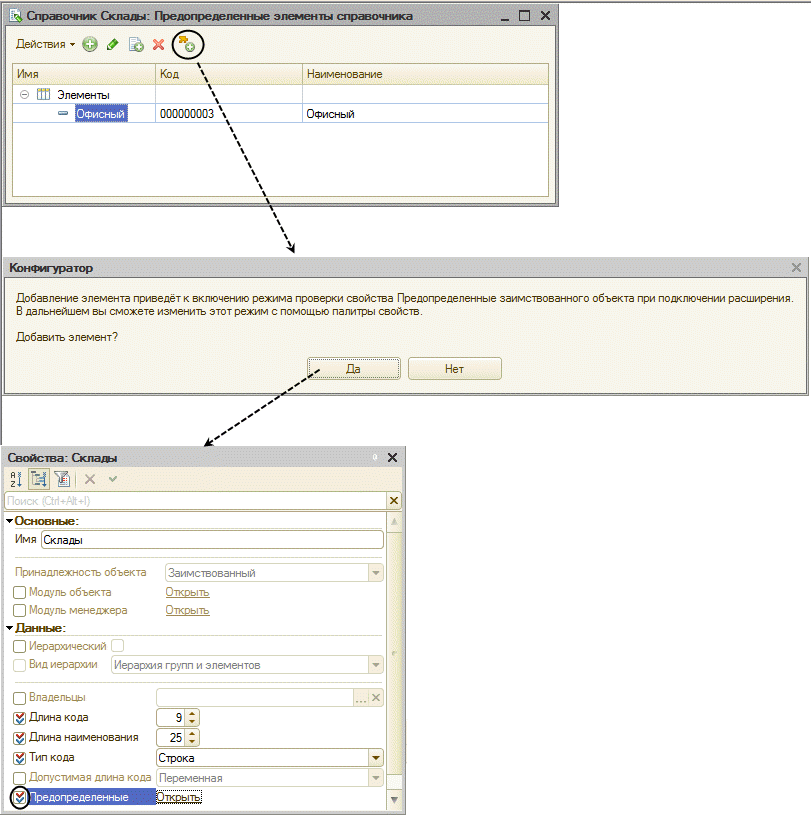
Склад = ЗНАЧЕНИЕ(Справочник.Склады.Офисный)) КАК ОстаткиТоваровОстатки

ПО (ОстаткиТоваровОстатки.Товар = СправочникТовары.Ссылка)

В этом запросе левым соединением связывается таблица справочника *Товары* с таблицей остатков регистра накопления *ОстаткиТоваров*. В параметры виртуальной таблицы передается отбор по складу, который должен быть равен ссылке на предопределенный элемент *Офисный* справочника *Склады*.

Теперь нам нужно позаимствовать все данные, на которых строится запрос: регистр *ОстаткиТоваров* вместе с его измерениями *Склад*, *Товар* и ресурсом *Количество*. А также нам нужно позаимствовать не только справочник *Склады*, но и его предопределенный элемент *Офисный*. Справочник *Товары* уже был заимствован вместе с формой списка.

Для заимствования предопределенного элемента выделим нужный элемент в списке предопределенных элементов справочника в конфигурации и нажмем кнопку *Добавить в расширение*. После этого в палитре свойств заимствованного справочника *Склады* свойство *Предопределенные* автоматически пометится как контролируемое (рис. 2.71).



**Рис. 2.71.**Заимствование предопределенного элемента справочника «Склады»

В результате при применении расширения к конфигурации будет проверяться, существует ли в конфигурации предопределенный элемент справочника *Склады* с такими же именем, кодом (длина и тип) и наименованием, что и в расширении. При несовпадении этих значений будет выдаваться ошибка и расширение не будет применено.

Надо заметить, что если бы форма списка с таким запросом существовала в конфигурации и вы бы заимствовали ее в расширение, то платформа самостоятельно проанализировала бы запрос динамического списка. И, исходя из этого, автоматически добавила бы все данные для запроса, включая предопределенный элемент и установку флажка контролируемости у свойства справочника *Предопределенные*.

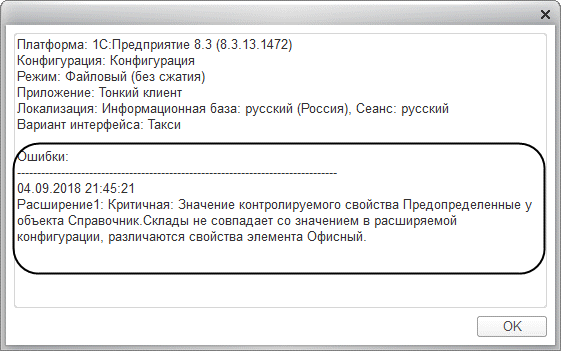
Перенесем поля динамического списка *Наименование*, *Остаток* и *Склад* в форму и запустим конфигурацию. В результате в режиме *1С:Предприятие* в списке товаров будут показаны все товары и их остатки по складу *Офисный*.

### Что произойдет, если свойства предопределенного элемента будут изменены в конфигурации

Предположим, в новой версии конфигурации поставщик изменил свойства расширяемого предопределенного элемента (например, *Наименование*).

В этом случае после обновления конфигурации наше расширение не сможет к ней примениться, так как свойство *Предопределенные* у справочника *Склады* в расширении установлено как контролируемое.

В результате при запуске обновленной конфигурации будет получена ошибка при применении расширения (рис. 2.72). И конфигурация будет запущена, но без расширения.



**Рис. 2.72.**Ошибка при применении расширения

## Модули

### Общая информация

Одной из самых мощных возможностей механизма расширений является возможность доработки практически любых модулей расширяемой конфигурации. Исключение составляют глобальные серверные модули. Также следует помнить, что расширение привилегированного общего модуля будет выполняться в непривилегированном режиме (если иное не разрешено профилем безопасности).

Кроме того, вы можете добавлять в расширение собственные общие модули, но они не могут быть привилегированными и глобальными серверными.

При разработке расширяющего модуля нужно учитывать, что этот модуль будет находиться в одном пространстве имен с расширяемым модулем. Поэтому из расширяющего модуля вы можете обращаться к переменным и методам расширяемого модуля. Также нужно помнить, что экспортные методы и переменные, созданные в расширяющем модуле, становятся частью публичного контекста расширяемого модуля. Поэтому они вам будут доступны из других модулей расширения.

В расширяющем модуле вы можете:

* создавать собственные методы и переменные;
* назначать собственные обработчики на события, которые не обрабатываются в конфигурации;
* перехватывать любой метод расширяемого модуля и вызывать до и/или после него собственные методы.

Создание собственного метода или переменной в расширяющем модуле ничем не отличается от того же процесса в конфигурации при работе без расширений. Но при этом настоятельно рекомендуется предварять имена создаваемых методов и переменных префиксом, указанным в свойстве *Префикс имен* расширения, чтобы избежать конфликта имен с конфигурацией.

Перехват методов (процедур или функций) конфигурации используется в том случае, когда перед и/или после расширяемого метода требуется вызвать собственный метод из расширяющего модуля. Имеется возможность также полностью заменить расширяемый метод собственным.

Для назначения собственных обработчиков событий (в том числе и тех, которые не назначены в конфигурации) также используется механизм перехвата методов конфигурации.

Перехват выполняется с помощью специального механизма *аннотаций*. Это структурные элементы встроенного языка – *&Перед*, *&После*, *&Вместо*, – с помощью которых указывается последовательность исполнения расширяемого и расширяющего методов.

В общем случае аннотированный метод из расширяющего модуля выглядит следующим образом (листинг 2.21).

**Листинг 2.21.** Пример аннотированного расширяющего метода

&Перед("ИмяМетода")

Процедура Префикс\_Имя()

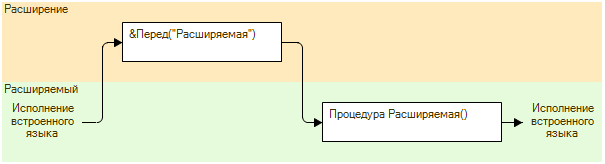
// текст метода

КонецПроцедуры

В данном примере *&Перед* – одна из трех возможных аннотаций, *ИмяМетода* – имя расширяемого метода, *Префикс\_Имя* – имя расширяющего метода.

### Аннотация «&Перед»

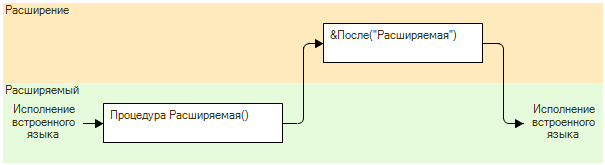
Аннотация с таким именем означает, что при вызове расширяемого метода сначала будет выполнен расширяющий метод, а затем – расширяемый метод. На следующей схеме показана последовательность исполнения методов встроенного языка при перехвате с аннотацией *&Перед* (рис. 2.73).



**Рис. 2.73.**Последовательность исполнения методов встроенного языка при перехвате с аннотацией «&Перед»

### Аннотация «&После»

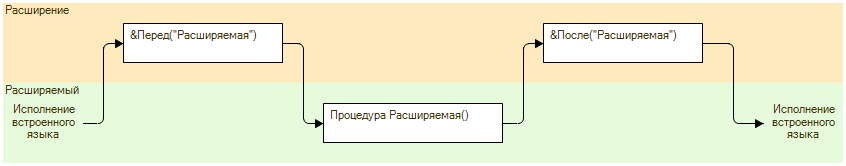
Эта аннотация означает, что при вызове расширяемого метода сначала будет выполнен расширяемый метод, а затем – расширяющий метод (рис. 2.74).



**Рис. 2.74.**Последовательность исполнения методов встроенного языка при перехвате с аннотацией «&После»

### Комбинация аннотаций «&Перед» и «&После»

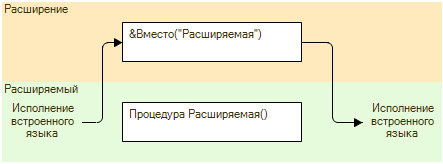
Если в расширяющем модуле созданы расширяющие методы, которые отмечены аннотациями *&Перед* и *&После*, то это означает, что при вызове расширяемого метода сначала будет вызван расширяющий метод, который отмечен аннотацией *&Перед*, затем расширяемый метод, а затем расширяющий метод, отмеченный аннотацией *&После* (рис. 2.75).



**Рис. 2.75.**Последовательность исполнения методов встроенного языка при перехвате с аннотациями «&Перед» и «&После»

### Аннотация «&Вместо»

Эта аннотация означает, что расширяющий метод полностью замещает собой расширяемый метод. Другими словами, при вызове расширяемого метода будет выполнен только расширяющий метод (рис. 2.76).



**Рис. 2.76.**Последовательность исполнения методов встроенного языка при перехвате с аннотацией «&Вместо»

Надо понимать, что при этом способе перехвата расширяемый метод не будет выполнен вообще. Причем он не будет выполнен не только в текущей версии конфигурации, но и в будущих версиях этой конфигурации тоже.

Следует учитывать, что для расширения функций вы можете использовать только расширяющий метод с аннотацией *&Вместо*.

### Аннотация «&Вместо» совместно с «ПродолжитьВызов()»

Чтобы иметь возможность модифицировать результат работы расширяемого метода, а также для расширения функций предусмотрен метод глобального контекста *ПродолжитьВызов()*. В качестве параметров метода должны быть указаны фактические параметры, которые переданы в расширяющий метод (рис. 2.77).



**Рис. 2.77.**Последовательность исполнения методов встроенного языка при перехвате с аннотацией «&Вместо» совместно с «ПродолжитьВызов()»

Таким образом, если, например, вызвать этот метод внутри расширяющей функции, то исполнится расширяемая функция, после чего исполнение кода вернется в расширяющую функцию.

На встроенном языке такая расширяющая функция может выглядеть следующим образом (листинг 2.22).

**Листинг 2.22.** Пример расширяющей функции

&Вместо("Расширяемая")

Функция Расш1\_Функция1(Парам1, Парам2)

// текст функции

…

Перем1 = ПродолжитьВызов(Парам1, Парам2);

…

// текст функции

Возврат Перем1;

КонецФункции

Метод *ПродолжитьВызов()* можно использовать не только при перекрытии функций, но и при перекрытии процедур. В этом случае результат его применения будет по смыслу такой же, как при использовании пары расширяющих методов *&Перед* и *&После*. Разница будет лишь в том, что в этом случае часть расширяющей процедуры, исполняющаяся до вызова расширяемой процедуры, и часть расширяющей процедуры, исполняющаяся после нее, будут существовать внутри одного контекста. В некоторых ситуациях это может быть удобно.

Однако использование пары аннотаций *&Перед/&После* лучше отражает факт дополнения, а не произвольной модификации конфигурации. Разработчику не придется смотреть внутрь метода, чтобы понять, что расширяемый метод безусловно вызывается.

### Стратегия выбора типа расширяющих методов

При выборе типа расширяющих методов разработчику расширения, помимо конкретной прикладной задачи, нужно учитывать, что расширение разрабатывается с целью добавить свою функциональность, а не навсегда отказаться от того, что есть сейчас и что появится в дальнейшем в конфигурации.

Поэтому самым плохим способом перехвата можно назвать перехват методов с помощью аннотации *&Вместо*. Даже если сейчас вам кажется, что метод конфигурации вам больше не нужен, то в дальнейшем в новых версиях конфигурации в него могут быть внесены полезные изменения. Но эти изменения у вас никогда не отработают, так как вы их полностью перекрыли своим расширением.

Надо понимать, что, используя перехват с аннотацией *&Вместо*, вы тем самым перекладываете ответственность за работоспособность и функциональность конфигурации на свое расширение. Поэтому рекомендуется использовать данный способ только в том случае, когда нет возможности применить другие способы расширения методов.

В качестве исключения аннотация *&Вместо* может быть использована в расширениях с назначением *Исправление*, которые предназначены для исправления ошибок в конфигурации.

Для расширения функций рекомендуется использовать расширяющие методы с аннотацией *&Вместо*совместно с методом *ПродолжитьВызов()*.

Во всех остальных случаях лучше использовать расширяющие методы с аннотациями *&Перед* и *&После*. Тогда расширяемый метод будет выполнен в любом случае. При этом все дальнейшие версии этого метода в случае изменения конфигурации также будут отрабатывать совместно с вашим расширением.

### Доработка модулей в расширении

Чтобы доработать модуль конфигурации, нужно добавить его в расширение. Заимствовать отдельно сам модуль нужно только в том случае, если вы хотите добавить туда собственную процедуру или функцию.

Чтобы доработать уже существующий метод модуля, достаточно просто поместить курсор внутрь этого метода и выбрать из контекстного меню команду *Добавить в расширение*. При этом если модуль еще не заимствован в расширение, то он заимствуется.

Поведение платформы в случаях, когда расширяется метод из модуля формы и когда расширяется метод из любого другого модуля, различается.

При расширении методов из любых модулей, кроме модулей форм, происходит следующее:

* Разработчику предлагается выбрать аннотацию для расширения метода (*Перед*, *После* или *Вместо*). При этом если в модуле уже существуют расширения данного метода, то имена использованных для этого аннотаций выделяются полужирным шрифтом с пометкой «уже создан».
* Если выбран способ расширения метода, уже существующий в расширении, то выполняется переход к этому методу. При этом возможна корректировка объявления метода, если оно стало отличаться от расширяемого метода.
* Если выбран несуществующий способ расширения, то в расширении будет создан новый метод, который будет предваряться соответствующей аннотацией.
* При этом надо учитывать:
  + что для функций недоступны аннотации *Перед* и *После*;
  + для процедуры недоступны:
    - аннотация *Вместо*, если уже существуют методы с аннотациями *Перед* или *После*;
    - аннотация *Перед*/*После*, если уже существует метод с аннотацией *Вместо*.

При расширении методов из модулей форм происходит следующее:

* Если расширяемый метод является обработчиком одного события или одной команды, то предлагается выбрать, каким образом выполнить расширение данного метода: как расширение для обработчика события/команды или как расширение обычного метода с аннотацией.
* Если расширяемый метод является обработчиком для нескольких событий или команд, то будет сформировано предупреждение о том, что данный метод не может быть расширен как обработчик события/команды и он будет расширен только как обычный метод. Затем будет предложено выбрать способ расширения метода.
* Для обычного метода, не являющего обработчиком какого-либо события, будет выполнено расширение метода с использованием аннотаций.

Следует учитывать, что если расширяемый метод обрамлен инструкциями препроцессора, то эти инструкции не будут перенесены в расширение.

Директивы компиляции, предваряющие расширяемую процедуру в модуле формы, должны совпадать с директивами этой процедуры в расширении. Иначе расширяющий метод выполняться не будет.

В процессе работы часто требуется перейти к расширяемому методу непосредственно из расширения. Для этого нужно просто поместить курсор на имя расширяемого метода в аннотации, а затем выполнить команду контекстного меню *Перейти к определению*.

### Взаимодействие нескольких расширений

Как мы знаем, для одной и той же конфигурации может быть разработано несколько расширений. Они должны работать автономно и независимо друг от друга. Но, как уже говорилось, порядок их применения к конфигурации определяется их назначением (*Исправление/Адаптация/Дополнение*).

При создании метода, расширяющего метод конфигурации, разработчик должен учитывать, что другое расширение также может перехватывать этот метод. И что другой расширяющий метод, в зависимости от своей аннотации и от порядка применения расширения, может быть выполнен либо перед, либо после, а может быть, даже вместо него.

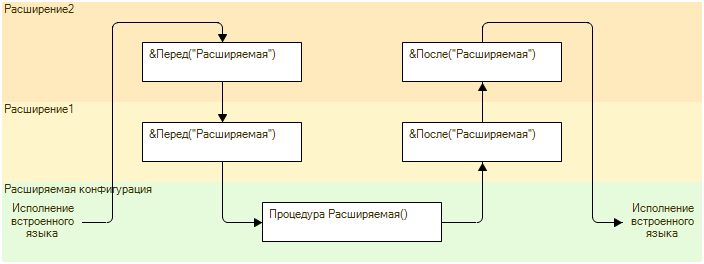
Поэтому при разборе сложных ситуаций, когда ваш расширяющий метод работает как-то не так, нужно не только исключить собственные ошибки, но и проанализировать порядок исполнения методов встроенного языка в случае работы нескольких расширений. Исправить ситуацию можно с помощью изменения аннотации вашего расширяющего метода или с помощью изменения назначения расширения.

Рассмотрим различные сценарии исполнения методов встроенного языка при условии применения к конфигурации двух расширений: сначала применяется *Расширение1* с назначением *Адаптация*, а затем применяется *Расширение2* с назначением *Дополнение*.

При этом *Расширение1* будет расширять непосредственно конфигурацию, а *Расширение2* будет расширять конфигурацию, которая является объединением *Расширения1* и конфигурации. Таким образом, в самом низу этой «слоистой» конструкции расположена конфигурация, откуда всегда начинается исполнение встроенного языка, непосредственно над ней расположен слой *Расширения1*, и на самом верху находится слой *Расширения2*.

При вызове расширяемого метода, перехваченного расширениями, исполнение встроенного языка передается в самый верхний расширяющий метод, затем опускается вниз, насколько это возможно. Если исполнение доходит до расширяемого метода и у него есть расширяющие методы, выполняющиеся после него, то они выполняются снизу вверх, начиная с самого нижнего расширяющего метода, а затем исполнение встроенного языка снова передается в конфигурацию.

Например, если в двух расширениях перехвачен (обрамлен) один и тот же расширяемый метод, то последовательность вызова расширяющих методов будет следующей (рис. 2.78).

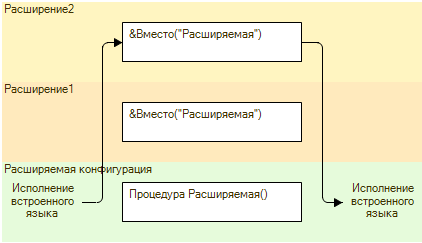


**Рис. 2.78**. Последовательность исполнения встроенного языка при перехвате метода двумя расширениями

При попытке вызвать расширяемый метод будет применяться следующая последовательность исполнения встроенного языка:

* Сначала будет вызван метод, отмеченный аннотацией *&Перед* из *Расширения2*.
* Затем будет вызван метод, отмеченный аннотацией *&Перед* из *Расширения1*.
* Затем будет вызван сам расширяемый метод, так как больше нет расширяющих методов, препятствующих его исполнению.
* Затем, в обратной последовательности, будут вызваны расширяющий метод *&После* из *Расширения1* и расширяющий метод *&После* из *Расширения2*.

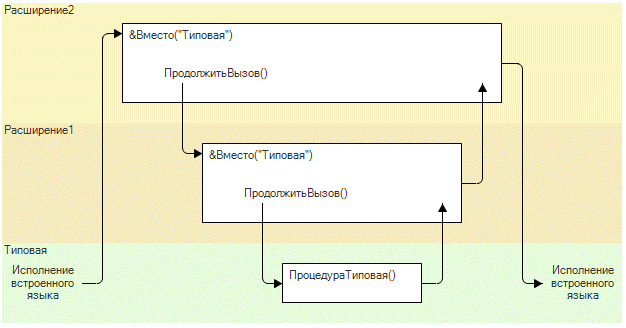
Если в каждом расширении расширяемый метод перехватывается полностью, то последовательность вызова расширяющих методов будет следующей (рис. 2.79).



**Рис. 2.79.** Последовательность исполнения встроенного языка при перехвате метода двумя расширениями

Эта схема хорошо иллюстрирует тот факт, что при перекрытии с помощью аннотации *&Вместо* перекрывается не только вызов расширяемого метода, но также и расширяющих методов, находящихся «под ним». В этом примере вместо расширяемого метода будет выполнен только расширяющий метод *&Вместо* из *Расширения2*.

Эту проблему частично может решить использование расширяющего метода *&Вместо* совместно с методом *ПродолжитьВызов()* (рис. 2.80).



**Рис. 2.80.** Последовательность исполнения встроенного языка при перехвате метода двумя расширениями

В этом случае из расширяющего метода *Расширения2* с помощью метода *ПродолжитьВызов()* вызывается расширяемый метод. Но поскольку он тоже перекрыт расширяющим методом из *Расширения1*, то выполняется сначала этот расширяющий метод, а уже из него вызывается исходный метод. После отработки расширяемого метода исполнение встроенного языка возвращается опять в расширяющий метод из *Расширения1*, а после его завершения в расширяющий метод из *Расширения2*.

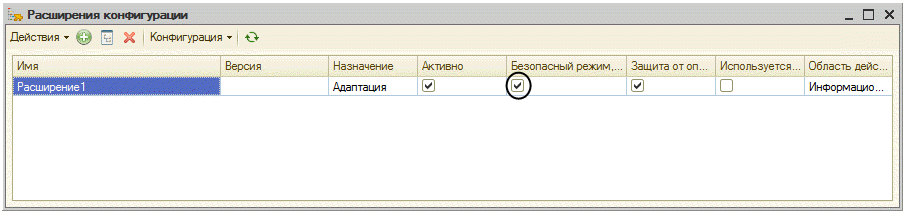
### Работа расширения в безопасном режиме

Если при старте «1С:Предприятия» ваше расширение успешно применилось к конфигурации, это еще не гарантия того, что все расширяющие методы, которые находятся в расширении, будут выполняться. Это связано с безопасностью.

В обычном режиме исполнения встроенного языка будут расширяться все методы конфигурации – и клиентские, и серверные.

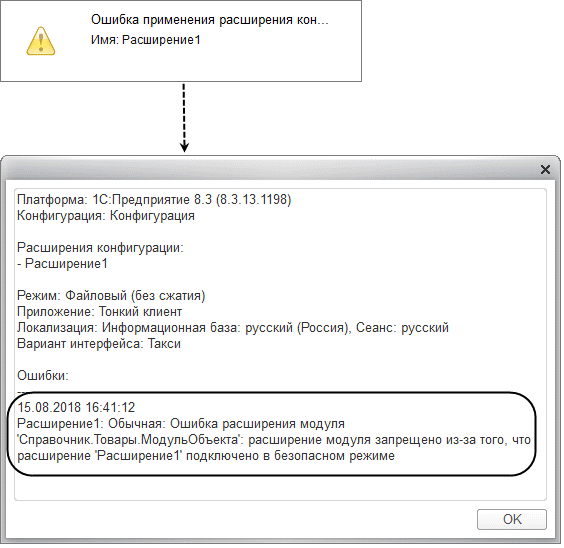
Если же прикладное решение работает в файловом варианте (как в нашем случае) или в клиент-серверном варианте без профилей безопасности, то при применении расширения в безопасном режиме исполнения встроенного языка будут расширяться только клиентские методы и серверные обработчики форм, которые установлены через панель свойств. К остальным серверным методам (в т. ч. и к аннотированным серверным методам модулей) расширение применяться не будет.

Стандартно расширение создается с установленным флажком *Безопасный режим* (рис. 2.81).



**Рис. 2.81.**Установка безопасного режима в свойствах расширения

Но в режиме *1С:Предприятие* при открытии, например, формы элемента справочника, в модуле которого создан расширяющий метод, появится всплывающее окно с сообщением об ошибке расширения модуля объекта справочника. Чтобы увидеть более подробное сообщение, нужно нажать на это всплывающее окно или на гиперссылку *Информация для технической поддержки* в окне информации о программе (рис. 2.82).



**Рис. 2.82.**Сообщение об ошибке расширения модуля объекта в режиме «1С:Предприятие»

Из сообщения видно, что расширение применено к конфигурации, но та часть, которая отвечает за расширение модуля, работать не будет. Причина ошибки в том, что расширение применено к конфигурации в безопасном режиме.

Чтобы расширение модуля заработало, нужно снять у него флажок *Безопасный режим* в списке расширений.

Если же ваше прикладное решение работает в клиент-серверном варианте, то возможность расширения серверных модулей регулируется профилем безопасности. Имя профиля безопасности задается здесь же, в списке расширений, в колонке *Безопасный режим, имя профиля безопасности* (см. рис. 2.81). В самом профиле безопасности вы можете или разрешить расширение для всех серверных модулей, а исключения перечислить в свойстве *Недоступные для расширения модули*, или просто перечислить все разрешенные расширения модулей в свойстве *Доступные для расширения модули*.

### Модуль объекта. Как создать обработчик события

В модуле объекта в расширении можно создать обработчик события, который отсутствует в конфигурации. Для этого также используются аннотации. В качестве примера рассмотрим следующую прикладную задачу.

Пусть в конфигурации существует справочник *Склады*, содержащий предопределенный элемент *Офисный*. И нам нужно перед удалением элемента справочника программно проверить и запретить удаление объекта, если это предопределенный элемент. Хотя обычно это делается с помощью прав доступа к данным, но на всякий случай можно проверить это в обработчике события *ПередУдалением* в модуле объекта в расширении.

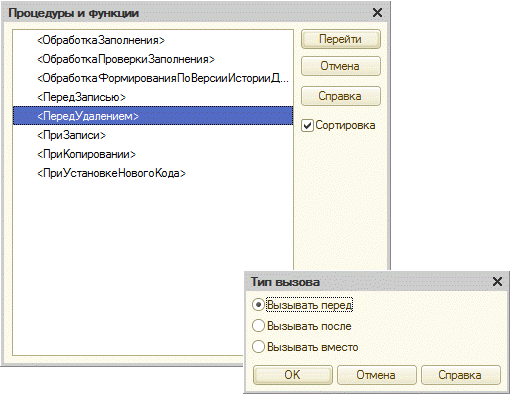
**ПРИМЕЧАНИЕ**

*Если предопределенный элемент используется в алгоритмах расширения, нужно также дополнительно заимствовать и сам элемент. Как и зачем это делать, описывается в разделе «*[*Зачем заимствовать предопределенные данные*](#_Зачем_заимствовать_предопределенные)*».*

Для этого нам нужно позаимствовать в расширение справочник *Склады*. И затем в модуле заимствованного объекта создать расширяющий метод *ПередУдалением*с аннотацией *&Перед*. В данном случае мы знаем, что обработчик события конфигурации отсутствует, и, казалось бы, перехватывать нечего. Но, как уже говорилось раньше, этот обработчик может появиться в конфигурации в дальнейшем, и тогда наш расширяющий метод будет выполнен перед ним.

Для установки отказа (в нашем случае – отказа от удаления) рекомендуется использовать расширяющий метод *&Перед*.

Итак, позаимствуем справочник. Затем откроем модуль объекта заимствованного справочника. В панели инструментов нажмем кнопку *Процедуры и Функции* и выберем событие *ПередУдалением*. После этого в появившемся диалоге укажем, что наш расширяющий метод будет выполняться перед расширяемым обработчиком (рис. 2.83).



**Рис. 2.83.**Создание расширяющего метода «ПередУдалением» с аннотацией «&Перед»

Расширяющую процедуру *ПередУдалением* с аннотацией *&Перед* заполним следующим образом (листинг 2.23).

**Листинг 2.23.** Обработчик события «ПередУдалением» с аннотацией «&Перед» в расширяющем модуле объекта

&Перед("ПередУдалением")

Процедура Расш1\_ПередУдалением(Отказ)

Если Предопределенный Тогда

Сообщение = Новый СообщениеПользователю;

Сообщение.Текст = "Нельзя удалить предопределенный элемент";

Сообщение.Сообщить();

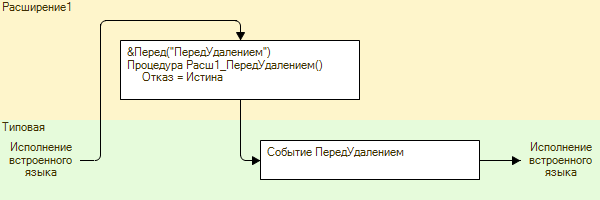
Отказ = Истина;

КонецЕсли;

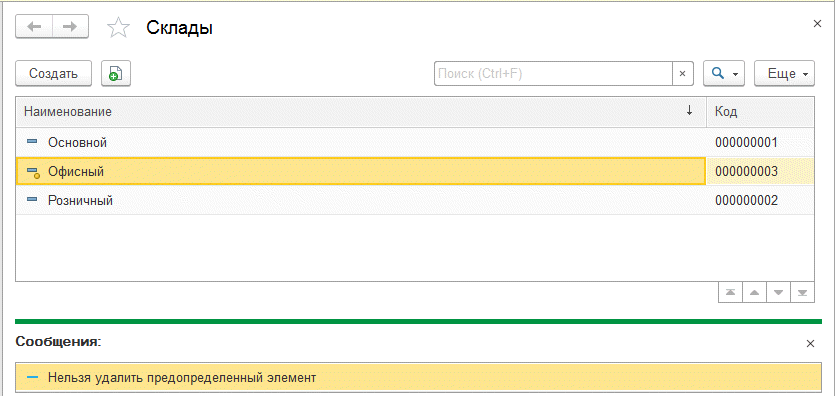
КонецПроцедуры

Перед тем как проверять результат, нужно не забыть снять флажок *Безопасный режим* у нашего расширения. Без этого в файловом варианте расширение модуля работать не будет.

Теперь запустим «1С:Предприятие», откроем список элементов справочника *Склады*, выделим элемент *Офисный* и выполним команду *Еще – Удалить*. В результате перед удалением элемента справочника выполнится наш расширяющий метод (см. листинг 2.23), и вам не удастся удалить предопределенный элемент (рис. 2.84, 2.85).



**Рис. 2.84.**Последовательность выполнения обработчиков события «ПередУдалением»



**Рис. 2.85.**Запрет удаления предопределенного элемента справочника «Склады»

#### Что произойдет, если в конфигурации появится обработчик события

Предположим, что в новой версии конфигурации поставщик создал обработчик события *ПередУдалением* в модуле объекта справочника *Склады*.

После применения расширения к обновленной конфигурации при вызове расширяемого обработчика события *ПередУдалением* перед ним будет вызван расширяющий метод из расширения с аннотацией *&Перед*, в котором запрещается удаление предопределенных элементов – параметр *Отказ* устанавливается в значение *Истина*.

Поскольку расширяющий метод выполняется первым, то расширяемый обработчик получает параметр *Отказ = Истина*. Поэтому удаление предопределенного элемента выполнено не будет. Чтобы удостовериться в этом, заполним расширяемый обработчик события *ПередУдалением* следующим образом (листинг 2.24).

**Листинг 2.24.** Расширяемый обработчик события «ПередУдалением»

Процедура ПередУдалением(Отказ)

Если Отказ Тогда

Сообщение = Новый СообщениеПользователю;

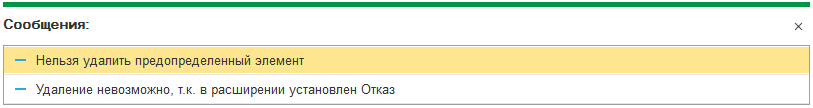
Сообщение.Текст = "Удаление невозможно, т.к. в расширении установлен Отказ";

Сообщение.Сообщить();

КонецЕсли;

КонецПроцедуры

В результате в режиме *1С:Предприятие* при попытке удалить предопределенный склад отработает сначала наш расширяющий метод, а затем расширяемый обработчик события. Элемент не будет удален, так как в расширяющем методе установлен параметр *Отказ* (рис. 2.86).



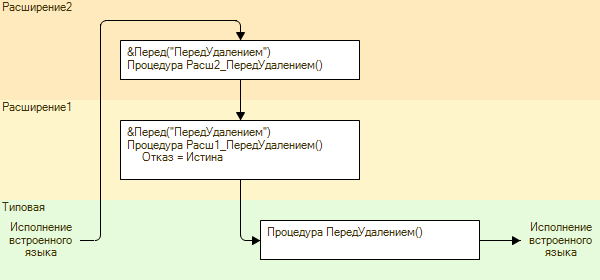
**Рис. 2.86.**Запрет удаления предопределенного элемента справочника «Склады»

#### Что произойдет, если к конфигурации будет применено еще одно расширение, перехватывающее тот же обработчик события. Обработка отказа при работе нескольких расширений

Предположим, что другой разработчик в другом расширении тоже создал расширяющий метод *ПередУдалением* в модуле объекта справочника *Склады*. В этом случае возможность удаления предопределенного элемента будет зависеть от аннотаций расширяющих методов и от порядка применения расширений.

Если расширяющий метод второго расширения будет иметь аннотацию *&Перед* или *&После*,то при любом порядке применения расширений в расширяемый обработчик события будет передан параметр *Отказ = Истина*. Поэтому удаление предопределенного элемента выполнено не будет.

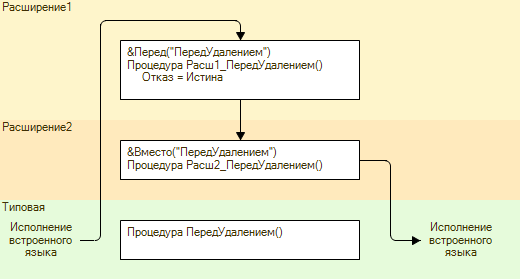
Отказ от удаления будет передан в расширяемый обработчик из первого расширения (рис. 2.87).



**Рис. 2.87.**Последовательность выполнения обработчиков события «ПередУдалением»

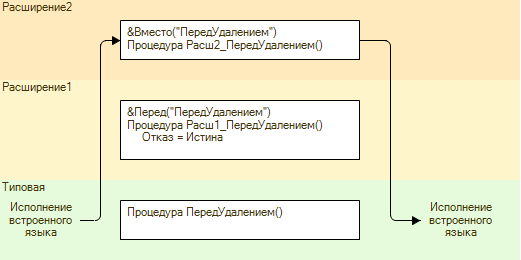
Нужно только учитывать, что при отсутствии обработчика в конфигурации расширяющие методы с аннотацией *&Перед* нижележащих расширений не запускаются, если в вышележащем расширении был установлен отказ. Поэтому если первое расширение (содержащее расширяющий метод с аннотацией *&Перед* (см. листинг 2.23), в котором запрещается удаление предопределенных элементов), применится последним, то расширяющий метод из второго (нижележащего) расширения вызван не будет (при отсутствии обработчика в конфигурации).

Если расширяющий метод второго расширения будет иметь аннотацию *&Вместо* и оно применится первым, то удаления также не произойдет. Так как при вызове расширяемого обработчика будет сначала вызван расширяющий метод из первого расширения. Затем расширяемый обработчик будет полностью перекрыт расширяющим методом из второго расширения, в которое передается отказ от удаления (рис. 2.88).



**Рис. 2.88.**Последовательность выполнения обработчиков события «ПередУдалением»

А вот если второе расширение будет применено последним, то расширяющий метод *&Вместо* перекроет не только расширяемый обработчик, но и расширяющий метод первого расширения. И элемент будет удален (рис. 2.89).



**Рис. 2.89.**Последовательность выполнения обработчиков события «ПередУдалением»

Мы уже говорили выше, что лучше не использовать аннотацию *&Вместо*. И вот лишнее этому подтверждение.

Ситуацию может исправить использование метода *ПродолжитьВызов()* (листинг 2.25).

**Листинг 2.25.** Перехватчик обработчика события «ПередУдалением» с аннотацией «&Вместо»

&Вместо("ПередУдалением")

Процедура Расш2\_ПередУдалением(Отказ)

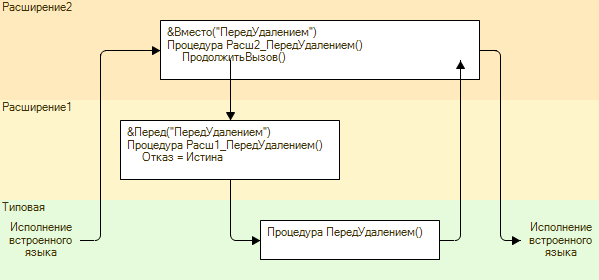
…

ПродолжитьВызов(Отказ);

…

КонецПроцедуры

В этом случае сначала начнет выполняться расширяющий метод *&Вместо* из второго расширения, затем с помощью метода *ПродолжитьВызов()* из него будет вызван расширяемый обработчик. Но доступ к нему закрывает еще расширяющий метод *&Перед* из первого расширения, отменяющий удаление предопределенного элемента. Поэтому сначала выполнится этот расширяющий метод, а затем – сам расширяемый обработчик, в который будет передан отказ от удаления. После этого продолжит выполняться расширяющий метод из второго расширения (рис. 2.90).



**Рис. 2.90.**Последовательность выполнения обработчиков события «ПередУдалением»

Расширение модуля менеджера и модуля команды принципиально ничем не отличается от расширения модуля объекта. Поэтому мы не будем отдельно на этом останавливаться.

### Как создать собственный общий модуль

Создание общих модулей и работа с ними в расширении в целом ничем не отличаются от этих же процессов в конфигурации при работе без расширений. Кроме того, что в расширении вы не можете создавать глобальные серверные и привилегированные модули.

Рассмотрим небольшой пример. Пусть в конфигурации существует документ *РасходнаяНакладная* с табличной частью *Товары* (реквизиты: *Товар*, *Количество*, *Цена*, *Сумма*). При изменении количества и цены товара в строке таблицы документа автоматически пересчитывается сумма по строке. Для этого используется процедура *РассчитатьСумму()* общего модуля *РаботаСДокументами*.

А также в конфигурации существует периодический регистр сведений *ЦеныТоваров* с измерением *Товар* и ресурсом *Цена*, который хранит цены товаров на определенные даты.

Мы хотим доработать конфигурацию так, чтобы при изменении товара в строке таблицы документа автоматически подставлялась бы актуальная цена товара из регистра сведений.

Для этого создадим в расширении общий неглобальный серверный модуль *Расш1\_РаботаСоСправочниками* с установленным флажком *Вызов сервера*. Поместим в этом модуле функцию *ПолучитьЦену()* и заполним ее следующим образом (листинг 2.26).

**Листинг 2.26.** Функция «ПолучитьЦену()»

Функция ПолучитьЦену(Дата, Товар) Экспорт

Отбор = Новый Структура("Товар", Товар);

Ресурсы = РегистрыСведений.ЦеныТоваров.ПолучитьПоследнее(Дата, Отбор);

Возврат Ресурсы.Цена;

КонецФункции

В эту функцию передаются дата и ссылка на товар, для которого нужно получить цену, актуальную на эту дату. Затем из регистра сведений *ЦеныТоваров* мы получаем последнюю запись по переданному товару на дату актуальности. И возвращаем значение ресурса регистра *Цена*.

Регистр сведений *ЦеныТоваров*, его измерение *Товар* и ресурс *Цена* нужно позаимствовать в расширение, так как они используются в коде функции.

Для того чтобы эта функция вызывалась при изменении товара в строке таблицы документа, нужно позаимствовать форму документа, создать в ней расширяющий метод *ПриИзменении* колонки *Товар* таблицы документа.

Этот процесс мы подробно рассматривали в разделе [«Как создать обработчик события для элемента формы»](#_Как_создать_обработчик), поэтому не будем здесь еще раз это объяснять.

Расширяющую процедуру заполним следующим образом (листинг 2.27).

**Листинг 2.27.** Расширяющая процедура «Расш1\_ТоварыТоварПриИзмененииПосле»

&НаКлиенте

Процедура Расш1\_ТоварыТоварПриИзмененииПосле(Элемент)

СтрокаТЧ = Элементы.Товары.ТекущиеДанные;

СтрокаТЧ.Цена = Расш1\_РаботаСоСправочниками.ПолучитьЦену(Объект.Дата, СтрокаТЧ.Товар);

РаботаСДокументами.РассчитатьСумму(СтрокаТЧ);

КонецПроцедуры

В этой процедуре мы получаем текущую строку табличной части документа. И передаем дату документа и значение колонки *Товар* текущей строки в функцию *ПолучитьЦену()* общего модуля *Расш1\_РаботаСоСправочниками*. А затем с помощью процедуры *РассчитатьСумму()* общего модуля расширяемой конфигурации *РаботаСДокументами* пересчитываем сумму в текущей строке табличной части с учетом только что полученной цены товара.

**ПРИМЕЧАНИЕ**

*Механизм расширений не позволяет создавать собственные глобальные серверные и привилегированные общие модули. Поэтому если в свойствах модуля Расш1\_РаботаСоСправочниками вы попытаетесь установить флажок Глобальный или Привилегированный, то в момент обновления конфигурации базы данных получите критическую ошибку.*

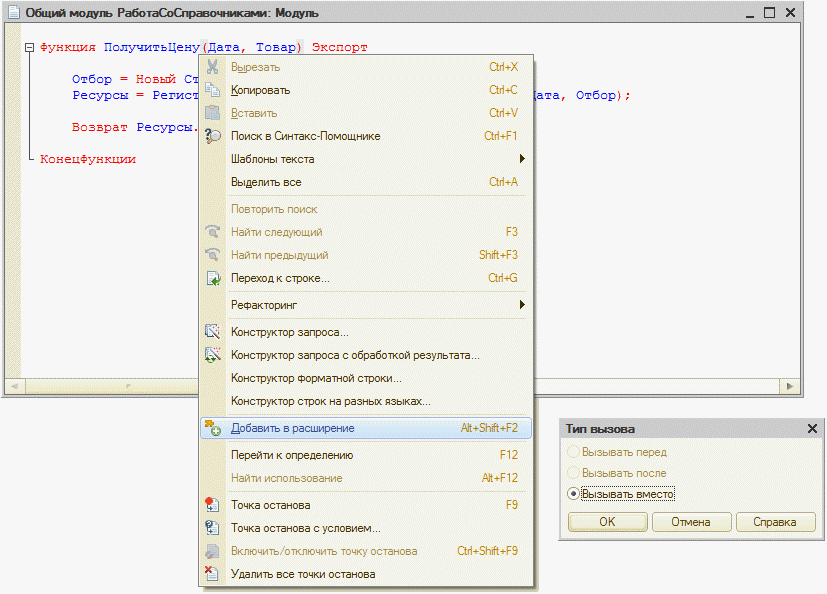
### Как доработать функцию общего модуля

Теперь предположим, что подстановка актуальной цены товара в документ *РасходнаяНакладная* уже реализована в конфигурации. Для этого используется функция *ПолучитьЦену()* (см. листинг 2.26) общего модуля *РаботаСоСправочниками*.

Но мы хотим, чтобы эта функция возвращала цену товара, увеличенную на некоторую надбавку (например, 20 %).

Для этого нам нужно создать расширяющий метод функции конфигурации и изменить значение, возвращаемое этой функцией.

Откроем общий модуль конфигурации *РаботаСоСправочниками*, установим курсор на заголовок функции *ПолучитьЦену()* и выполним команду контекстного меню *Добавить в расширение*. При этом модуль заимствуется и предлагается выбрать тип вызова. Для функций доступен только тип перехвата *Вместо* (рис. 2.91).



**Рис. 2.91.**Перехват функции

После этого в заимствованном общем модуле будет создан шаблон расширяющей функции с аннотацией *&Вместо* и с вызовом функции глобального контекста *ПродолжитьВызов()*. Нам нужно только изменить последнюю строку функции так, чтобы возвращаемый результат увеличивался на 20 % (листинг 2.28).

**Листинг 2.28.** Расширяющая функция «Расш1\_ПолучитьЦену()»

&Вместо("ПолучитьЦену")

Функция Расш1\_ПолучитьЦену(Дата, Товар)

// Вставить содержимое метода.

Результат = ПродолжитьВызов(Дата, Товар);

Возврат Результат \* 1.2;

КонецФункции

При выполнении этой функции с помощью метода *ПродолжитьВызов()* вызывается расширяемая функция, в которую передаются те же параметры, что и в расширяющую функцию (дата актуальности и ссылка на товар). Эта функция возвращает цену товара из регистра сведений, которую мы увеличиваем на надбавку.

Теперь надо не забыть снять флажок *Безопасный режим* у нашего расширения в списке расширений. Как уже говорилось, расширение модулей не может выполняться в безопасном режиме в файловом варианте работы.

После этого запустим «1С:Предприятие» и откроем расходную накладную. Мы видим, что теперь при изменении товара в строке таблицы документа подставляется актуальная на дату документа цена товара, увеличенная на 20-процентную надбавку.

При заимствовании общих модулей необходимо учитывать следующее:

* Нельзя заимствовать глобальные серверные модули.
* Код из расширения будет исполняться только в непривилегированном режиме (если иное не разрешено в профиле безопасности).

Сама операция заимствования глобального серверного модуля не запрещена в дереве конфигурации, но на этапе обновления конфигурации базы данных вы получите ошибку, и обновление выполнено не будет.

### Модуль формы. Перехват процедур с помощью аннотаций

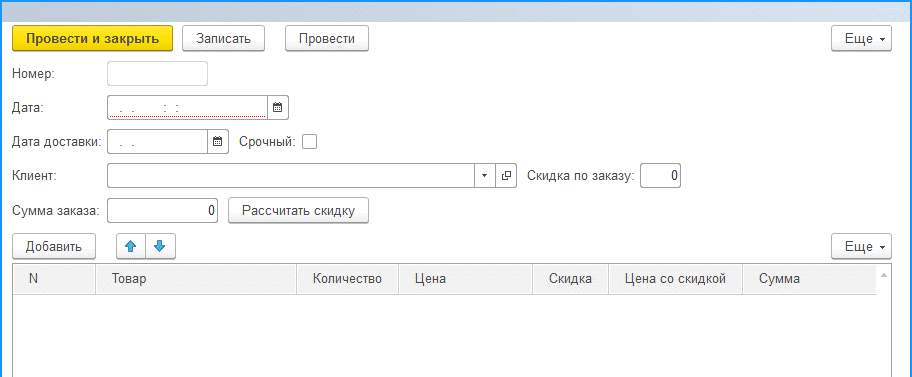
Перехват обработчиков событий/команд формы мы уже показывали в разделах [«Как создать обработчик события для элемента формы»](#_Как_создать_обработчик), [«Как перехватить обработчик события и обрамить его своими алгоритмами](#_Как_перехватить_обработчик)», [«Как добавить в форму команду и создать ее обработчик»](#_Как_добавить_в), поэтому в этом разделе мы покажем перехват процедур, не являющихся обработчиками событий. Он выполняется с помощью аннотаций.

Как вы знаете, в модуле формы все обработчики событий назначаемые и не имеют фиксированных имен. Чтобы платформа понимала, чем нужно обрабатывать то или иное событие, в конфигураторе, в панели свойств, должна существовать привязка конкретной процедуры к конкретному событию.

Именно по этой причине, и только при перехвате обработчиков событий в форме, вам нужно использовать не аннотации, а палитру свойств. Хотя любые другие методы модуля, не являющиеся обработчиками событий, вы можете перехватывать с помощью аннотаций.

Рассмотрим пример. Пусть в конфигурации существует документ *Заказ*, в котором отражаются заказы товаров, сделанные клиентами. У клиентов, в зависимости от истории их заказов, существуют персональные скидки. А также каждому клиенту устанавливается статус: *Новый*, *Давний*, *Важный* и пр. (значения перечисления *ВидыКлиентов*).

Не вдаваясь в детали, опишем функциональность формы заказа для лучшего понимания задачи, которую мы хотим выполнить с помощью расширения. Форма заказа в конфигураторе выглядит следующим образом (рис. 2.92).



**Рис. 2.92.**Форма заказа

При добавлении товара в таблицу заказанных товаров из справочника товаров автоматически подставляется его цена. При изменении количества товара пересчитывается сумма по строке табличной части документа. Общий итог по полю *Сумма* сохраняется в реквизите *СуммаЗаказа* документа *Заказ*.

При изменении клиента его персональная скидка записывается в реквизит *СкидкаПоЗаказу*. Чтобы пересчитать суммы заказанных товаров с учетом скидки, нужно нажать на кнопку *Рассчитать скидку*.

При этом вызывается серверная процедура *ПересчитатьСкидкуПоЗаказу()*, которая записывает скидку по заказу в каждую строку табличной части, пересчитывает суммы по товарам с учетом скидки и затем пересчитывает общую сумму заказа (листинг 2.29).

**Листинг 2.29.** Процедура «ПересчитатьСкидкуПоЗаказу»

&НаСервере

Процедура ПересчитатьСкидкуПоЗаказу()

ПерсональнаяСкидка = Объект.СкидкаПоЗаказу;

Для каждого ТекущаяСтрока из Объект.Товары Цикл

ТекущаяСтрока.Скидка = ПерсональнаяСкидка;

ТекущаяСтрока.ЦенаСоСкидкой = ТекущаяСтрока.Цена \* (1 - ТекущаяСтрока.Скидка/100);

ТекущаяСтрока.Сумма = ТекущаяСтрока.Количество \* ТекущаяСтрока.ЦенаСоСкидкой;

КонецЦикла;

Объект.СуммаЗаказа = Объект.Товары.Итог("Сумма");

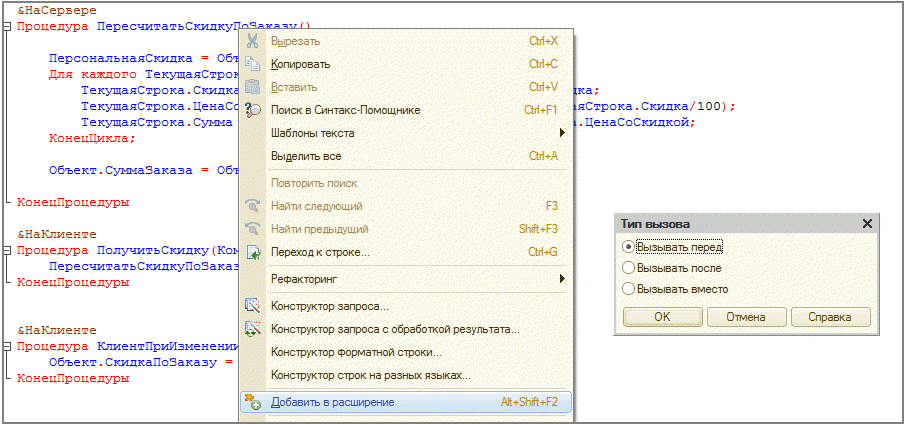
КонецПроцедуры

Теперь нам нужно с помощью расширения добавить в форму заказа следующую функциональность. Мы хотим, чтобы при нажатии на кнопку *Рассчитать скидку*, перед тем как пересчитывать скидку по заказу, для клиентов со статусом *Новый*, если они не имеют персональной скидки, устанавливалась бы скидка по заказу, равная 5 %. А для клиентов со статусом *Важный* устанавливалась бы скидка по заказу, равная сумме персональной скидки + 3 %.

Кроме того, мы хотим, чтобы после нажатия на кнопку *Рассчитать скидку* персональная скидка клиента рассчитывалась бы в зависимости от общей суммы заказа и записывалась бы в справочник клиентов.

Для этого нам нужно перехватить процедуру конфигурации *ПересчитатьСкидкуПоЗаказу()* с аннотацией *&Перед* и написать там соответствующий код.

Откроем модуль формы документа *Заказ*, установим курсор на заголовок процедуры *ПересчитатьСкидкуПоЗаказу()*и выполним команду контекстного меню *Добавить в расширение*. При этом форма заимствуется и предлагается выбрать тип вызова. Выберем тип перехвата *Перед* (рис. 2.93).



**Рис. 2.93.**Перехват процедуры в модуле формы

После этого в модуле заимствованной формы будет создан шаблон расширяющей процедуры с аннотацией *&Перед*. Заполним его следующим образом (листинг 2.30).

**Листинг 2.30.** Расширение процедуры «ПересчитатьСкидкуПоЗаказу()» с аннотацией «&Перед»

&НаСервере

&Перед("ПересчитатьСкидкуПоЗаказу")

Процедура Расш1\_ПересчитатьСкидкуПоЗаказу()

Если Объект.Клиент.Статус = ПредопределенноеЗначение("Перечисление.ВидыКлиентов.Новый") И Объект.СкидкаПоЗаказу = 0 Тогда

Объект.СкидкаПоЗаказу = 5;

ИначеЕсли Объект.Клиент.Статус = ПредопределенноеЗначение("Перечисление.ВидыКлиентов.Важный") Тогда

Объект.СкидкаПоЗаказу = ПолучитьСкидкуПоКлиенту(Объект.Клиент) + 3;

КонецЕсли;

КонецПроцедуры

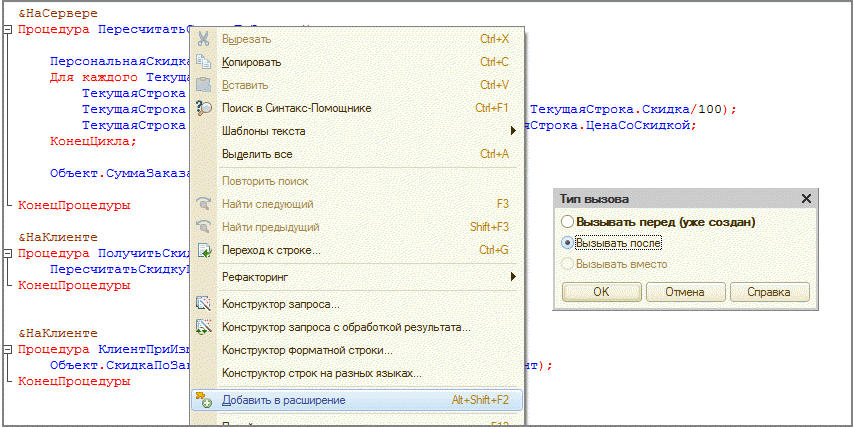
Справочник *Клиенты* и его реквизит *Статус*, а также значения *Новый* и *Важный* перечисления *ВидыКлиентов* нужно заимствовать в расширение, так как они используются в коде процедуры.

Снимем флажок *Безопасный режим* у нашего расширения (так как иначе не будут расширяться серверные аннотированные методы форм), запустим «1С:Предприятие» и убедимся, что все работает, как мы хотели.

Вернемся в конфигуратор и добавим еще функциональность, которая будет выполняться после нажатия на кнопку *Рассчитать скидку*. Нам нужно, чтобы после пересчета скидки по заказу персональная скидка клиента рассчитывалась бы в зависимости от общей суммы заказа и записывалась бы в справочник клиентов.

Для этого нам нужно перехватить процедуру конфигурации *ПересчитатьСкидкуПоЗаказу()* с аннотацией *&После* и написать там соответствующий код.

Откроем модуль формы документа *Заказ*, установим курсор на заголовок процедуры *ПересчитатьСкидкуПоЗаказу()* и выполним команду контекстного меню *Добавить в расширение*. При этом форма заимствуется и предлагается выбрать тип вызова. Мы видим, что расширяющий метод с типом перехвата *Перед* уже создан. Выберем тип перехвата *После* (рис. 2.94).



**Рис. 2.94.**Перехват процедуры в модуле формы

После этого в модуле заимствованной формы будет создан шаблон расширяющей процедуры с аннотацией *&После*. Заполним его следующим образом (листинг 2.31).

**Листинг 2.31.** Расширение процедуры «ПересчитатьСкидкуПоЗаказу()» с аннотацией «&После»

&НаСервере

&После("ПересчитатьСкидкуПоЗаказу")

Процедура Расш1\_ПересчитатьСкидкуПоЗаказу1()

СкидкаКлиента = 0;

Если Объект.СуммаЗаказа > 15000 Тогда

СкидкаКлиента = 15;

ИначеЕсли Объект.СуммаЗаказа > 10000 Тогда

СкидкаКлиента = 10;

ИначеЕсли Объект.СуммаЗаказа > 5000 Тогда

СкидкаКлиента = 5;

КонецЕсли;

Клиент = Объект.Клиент.ПолучитьОбъект();

Если СкидкаКлиента > Клиент.Скидка Тогда

Клиент.Скидка = СкидкаКлиента;

Клиент.Записать();

КонецЕсли;

КонецПроцедуры

Справочник *Клиенты* и его реквизит *Скидка* нужно заимствовать в расширение, так как они используются в коде процедуры.

Запустим «1С:Предприятие» и убедимся, что для нового клиента устанавливается первоначальная предварительная 5-процентная скидка по заказу. Для важного клиента его персональная скидка увеличивается на 3 %. Это отрабатывает расширяющий метод *Перед* процедуры *ПересчитатьСкидкуПоЗаказу()*.

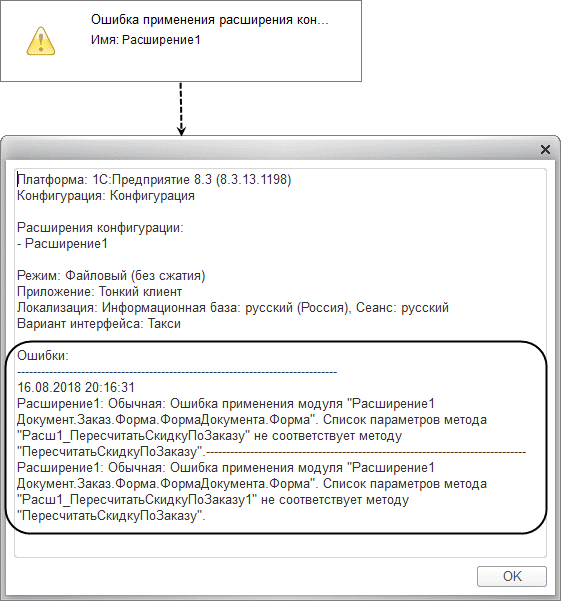
Затем выполняется пересчет всех сумм заказанных товаров с учетом скидки, а также пересчитывается общая сумма заказа. Это отрабатывает расширяемая процедура *ПересчитатьСкидкуПоЗаказу()*.

После этого в зависимости от общей суммы заказа рассчитывается персональная скидка клиента, и если она больше его текущей скидки, то она записывается в справочник клиентов. Это отрабатывает расширяющий метод *После* процедуры *ПересчитатьСкидкуПоЗаказу()*.

#### Что произойдет, если в конфигурации изменится объявление метода

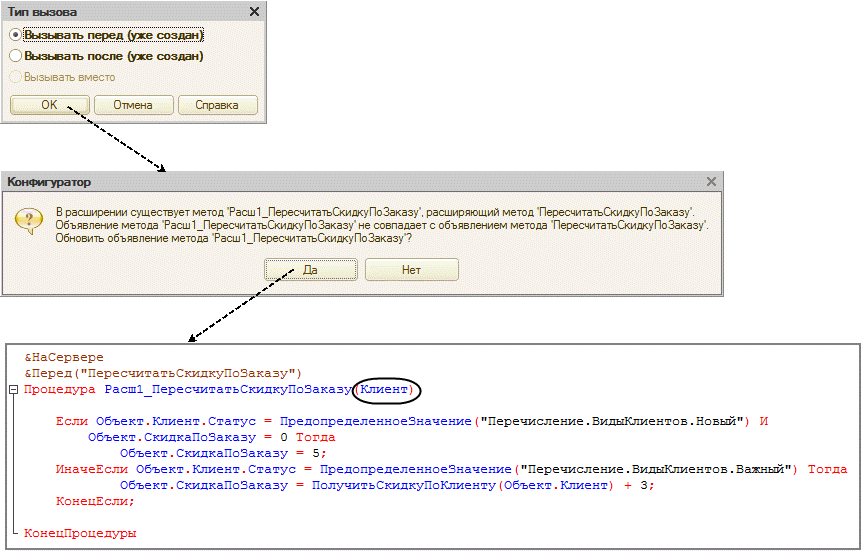
Предположим, что поставщик конфигурации изменил объявление метода. Например, процедура *ПересчитатьСкидкуПоЗаказу()* теперь имеет параметр и описывается как *ПересчитатьСкидкуПоЗаказу(Клиент)*.

В результате при старте клиентского сеанса расширение будет применено к конфигурации, но расширение метода конфигурации выполняться не будет из-за разности объявлений метода конфигурации и расширяющего метода (рис. 2.95).



**Рис. 2.95.**Ошибка расширения метода

Чтобы исправить ситуацию, можно еще раз добавить расширяющий метод с существующим типом перехвата из формы конфигурации. В результате объявление расширяющего метода будет приведено в соответствие с методом конфигурации (рис. 2.96).



**Рис. 2.96.**Обновление объявления расширяющего метода в соответствии с методом конфигурации

## Роли

С помощью расширений можно дорабатывать существующие в конфигурации роли. Для этого надо позаимствовать роль в расширение и установить или снять права этой роли на собственные или заимствованные объекты. Таким образом, с помощью заимствованной роли имеется возможность как расширить права пользователя, так и сузить их.

Кроме того, в расширение можно добавить собственную роль и программно назначить ее пользователю информационной базы.

При работе с ролями расширения следует иметь в виду, что если расширение не содержит собственных ролей, то доступ к объектам расширения предоставляется всем пользователям информационной базы. В то же время если в расширении существуют собственные роли, которые регламентируют доступ к объектам расширения, то всем пользователям информационной базы, которые должны иметь доступ к объектам расширения, следует добавить роли из расширения. В противном случае объекты расширения не будут доступны пользователям.

Также надо учитывать, что в файловом варианте поддерживаются любые возможности изменения прав доступа: как ограничение прав, так и их расширение.

В клиент-серверном варианте изменение прав доступа регулируется наличием профиля безопасности и его настройками:

* Если прикладное решение работает без использования профиля безопасности, то расширение ролей не допускается.
* Если прикладное решение работает с использованием профиля безопасности, то ограничивать права доступа можно в любом случае, а расширять права доступа можно только при соответствующей настройке профиля безопасности.

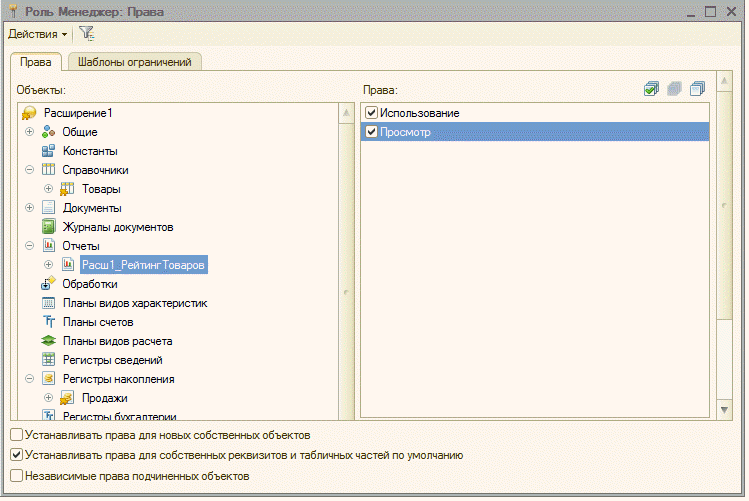
### Как доработать роль

Рассмотрим два примера: один, когда у роли конфигурации требуется расширить набор прав, и второй, когда у роли требуется сузить набор прав.

Пусть в конфигурации существуют роли *Администратор*, *Менеджер* и *Продавец*. Менеджер и продавец имеют права на один и тот же набор объектов, за исключением того, что продавец не может изменять справочники, а продавец – документы. Но они оба имеют доступ к одним и тем же отчетам конфигурации.

Но теперь в расширение добавлен новый отчет *Расш1\_РейтингТоваров*, который разработан специально для менеджеров и руководства.

Поэтому нам нужно позаимствовать роль *Менеджер* в расширение и предоставить ей права на отчет *Расш1\_РейтингТоваров* (рис. 2.97).



**Рис. 2.97.**Добавление прав у роли «Менеджер» на отчет «Расш1\_РейтингТоваров»

В результате пользователь с ролью *Менеджер*, в отличие от других пользователей, может увидеть и сформировать отчет *Рейтинг товаров*.

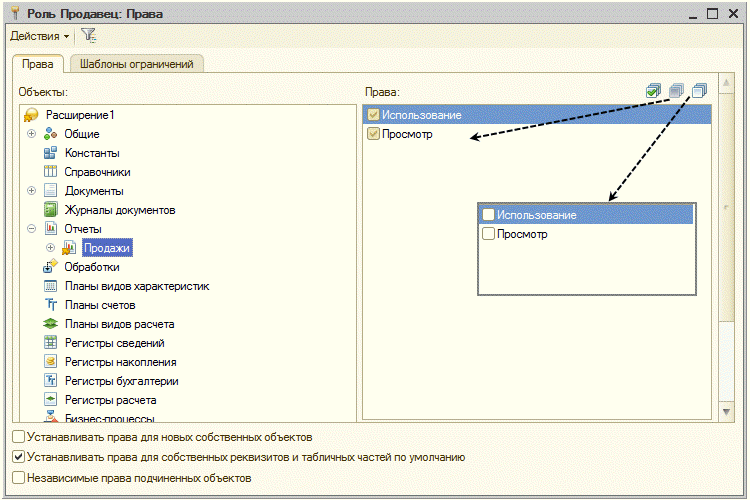
Нужно учитывать, что, пока в расширении нет ролей, все объекты из него доступны всем пользователям. Как только появляется необходимость в ролевом регулировании, то достаточно добавить в расширение любую роль, и объекты из расширения становятся доступны только тем ролям, которые заимствованы в расширение и имеют соответствующие права.

Теперь рассмотрим случай, когда у роли конфигурации требуется сузить набор прав.

Предположим, и менеджеры, и продавцы имеют доступ к отчетам по продажам. Но на конкретном внедрении требуется, чтобы продавцы не могли просматривать итоговые данные о продажах.

Для этого нам нужно позаимствовать роль *Продавец* и отчет *Продажи* в расширение и снять права заимствованной роли на этот отчет.

Стандартно права всех заимствованных ролей на заимствованные объекты берутся из конфигурации. Флажки прав при этом затенены. Но если в расширении флажки явно сняты или установлены, то права берутся из расширения. В нашем случае мы должны снять права (рис. 2.98).



**Рис. 2.98.**Снятие прав у роли «Продавец» на отчет «Продажи»

В результате пользователь с ролью *Продавец* больше не сможет просматривать отчет по продажам.

### Как создать собственную роль и добавить ее пользователю

Часто бывает удобно создать в расширении собственную роль, настроить в ней доступ к объектам, относящимся к определенной функциональности расширения, и затем уже эту готовую роль добавить нужным пользователям информационной базы.

Например, в расширении разработана функциональность по заказам товаров покупателями. Для этого в расширение заимствованы и доработаны там справочник *Клиенты*, документ *Заказы* и регистр сведений *ЦеныТоваров*. А также в расширении разработаны собственные отчеты *Расш1\_ИсполнениеЗаказов* и *Расш1\_ПричиныОтказов*.

В организации заказчика работой с заказами товаров занимаются отдельные люди. Поэтому удобнее создать отдельную роль с комплексным набором прав на перечисленные выше объекты и программно добавить ее нужным пользователям.

Сначала нам нужно позаимствовать в расширение роль с административными правами *Администратор*.

Затем добавим новую роль *Расш1\_Заказы* с синонимом «Работа с заказами» и установим для нее права на объекты: справочник *Клиенты*, документ *Заказы*, регистр сведений *ЦеныТоваров*, отчеты *Расш1\_ИсполнениеЗаказов* и *Расш1\_ПричиныОтказов*.

Обратите внимание, что у этой роли нет возможности редактирования прав доступа, которые задаются для корневого объекта конфигурации (административные права, возможность использования того или иного клиентского приложения и т. д.).

Далее добавим в расширение общую команду *Расш1\_НоваяРоль* с синонимом «Назначить новую роль» и зададим у нее свойство *Группа – Панель действий.Сервис*. Право на просмотр этой команды предоставим заимствованной роли *Администратор*.

Модуль команды заполним следующим образом (листинг 2.32).

**Листинг 2.32.** Модуль команды «Расш1\_НоваяРоль»

&НаКлиенте

Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

НазначитьНовуюРоль();

КонецПроцедуры

&НаСервере

Процедура НазначитьНовуюРоль()

ПользовательИБ = ПользователиИнформационнойБазы.НайтиПоИмени("Самойлов");

ПользовательИБ.Роли.Добавить(Метаданные.Роли.Расш1\_Заказы);

ПользовательИБ.Записать();

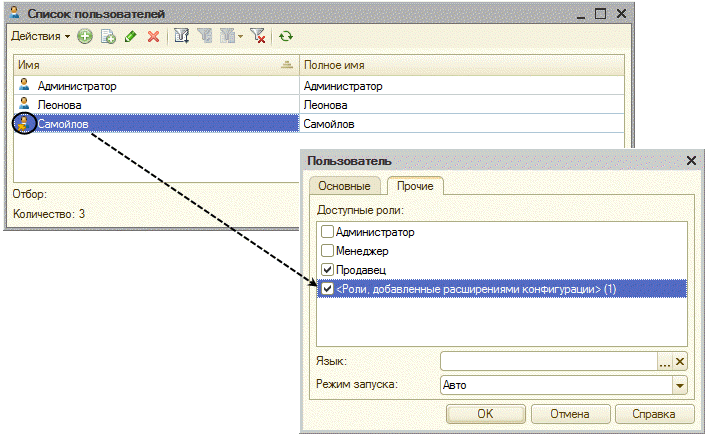
КонецПроцедуры

В процедуре *НазначитьНовуюРоль()*, вызывающейся из обработчика команды, пользователь информационной базы находится по имени. И ему добавляется новая роль.

В заключение добавим пользователю *Самойлов* роль *Продавец* из конфигурации. Для наглядности мы сделали ее «пустой», но в ней по умолчанию включены права на запуск приложения и вывод. Эти права нельзя задать в новой роли из расширения. Поэтому тот пользователь, которому добавляется роль из расширения, должен иметь хотя бы одну роль из конфигурации.

Запустим «1С:Предприятие» от имени администратора и из группы команд *Сервис* выполним команду *Назначить новую роль*. В окне предупреждения безопасности мы должны разрешить администрирование пользователей и выполнить эту команду еще раз.

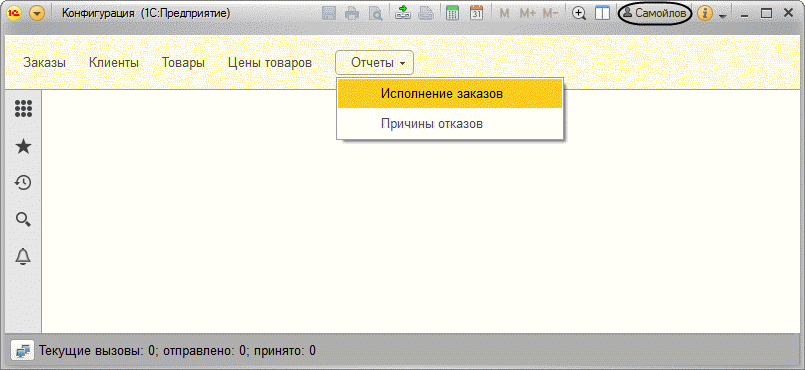
После этого в конфигураторе, в списке пользователей, пользователь, которому добавлена роль из расширения, будет отмечен специальной пиктограммой. А в свойствах этого пользователя на закладке *Прочее* появится флажок *Роли, добавленные расширениями конфигурации* (рис. 2.99).



**Рис. 2.99.**Свойства пользователя, которому добавлена роль из расширения

Для удаления роли или сразу всех добавленных расширениями ролей пользователя нужно снять этот флажок.

Запустим «1С:Предприятие» от имени пользователя с новой ролью (*Самойлов*). Мы видим, что ему доступны все объекты, относящиеся к заказам товаров, как мы и хотели (рис. 2.100).



**Рис. 2.100.**Запуск «1С:Предприятия» от имени пользователя с новой ролью

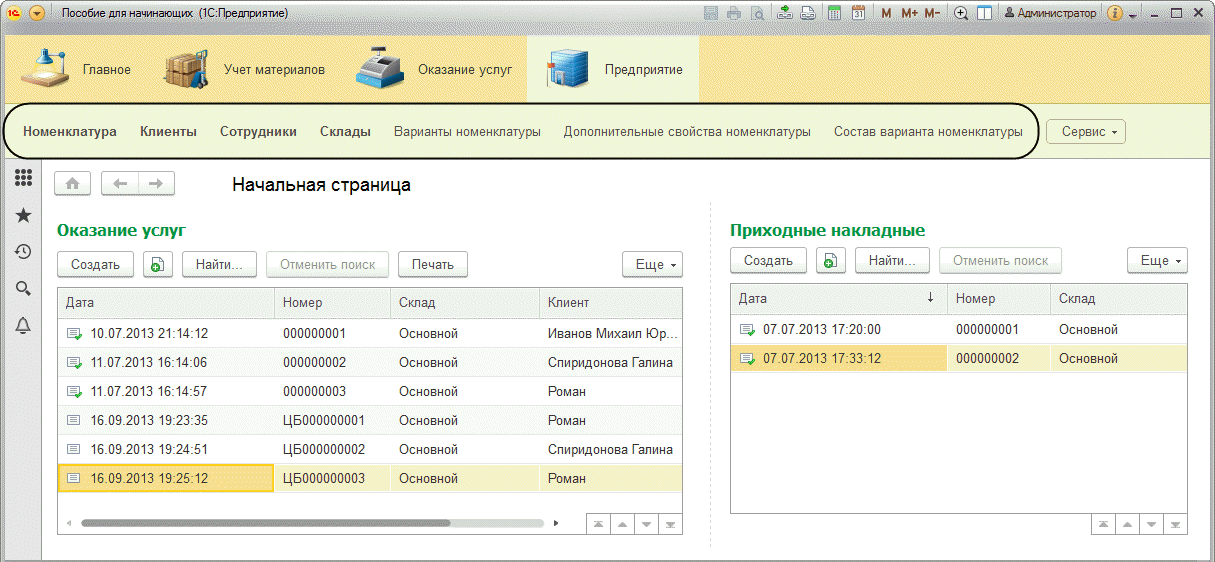
## Подсистемы

С помощью расширения можно доработать существующие в конфигурации подсистемы. Для этого надо заимствовать подсистему в расширение и добавить в нее собственные или заимствованные объекты. Удалить объекты, уже присутствующие в заимствованной подсистеме, вы не сможете.

Кроме того, в случае если расширение дополняет функциональность конфигурации, можно добавить в расширение собственную подсистему и включить в нее как собственные, так и заимствованные объекты.

### Как доработать подсистему

Пусть в конфигурации существует подсистема *Предприятие*. При запуске приложения командный интерфейс подсистемы выглядит следующим образом (рис. 2.101).

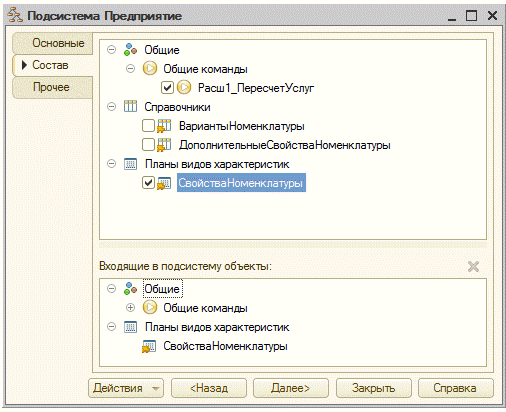


**Рис. 2.101.**Командный интерфейс подсистемы «Предприятие» при запуске приложения

С помощью расширения мы хотим добавить в подсистему *Предприятие* собственную команду *Пересчет услуг*, отобразить в подсистеме план видов характеристик *СвойстваНоменклатуры*, существующий в конфигурации, и убрать из подсистемы справочники *ВариантыНоменклатуры* и *ДополнительныеСвойстваНоменклатуры*.

Для этого позаимствуем в расширение справочники и план видов характеристик. Затем добавим в расширение общую команду *Расш1\_ПересчетУслуг* с синонимом «Пересчет услуг» и зададим у нее свойство *Группа* – *Панель действий.Сервис*.

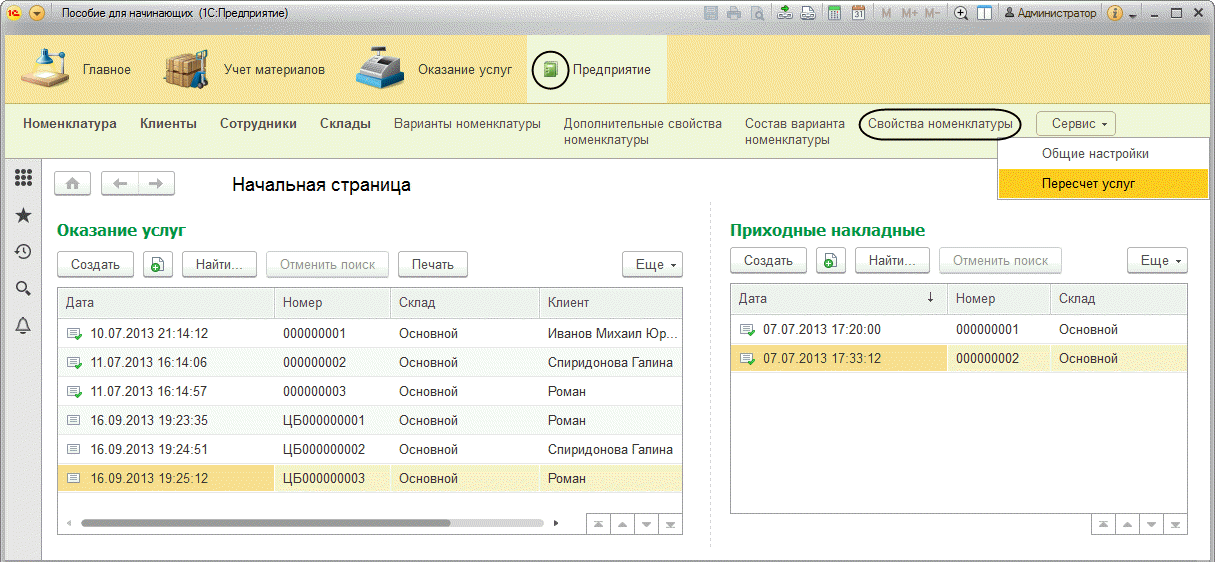
Далее позаимствуем в расширение подсистему *Предприятие*. Включим в состав подсистемы общую команду и план видов характеристик. У заимствованных справочников не будем устанавливать флажки принадлежности к подсистеме (рис. 2.102).



**Рис. 2.102.**Состав объектов заимствованной подсистемы «Предприятие»

Кроме того, установим для заимствованной подсистемы новую картинку.

В результате при запуске приложения состав объектов заимствованной подсистемы дополнит состав расширяемой подсистемы. И мы увидим команду для открытия плана видов характеристик (*Свойства номенклатуры*), а также новую команду *Пересчет услуг* в группе команд *Сервис*. Картинка у подсистемы изменится на новую (рис. 2.103).



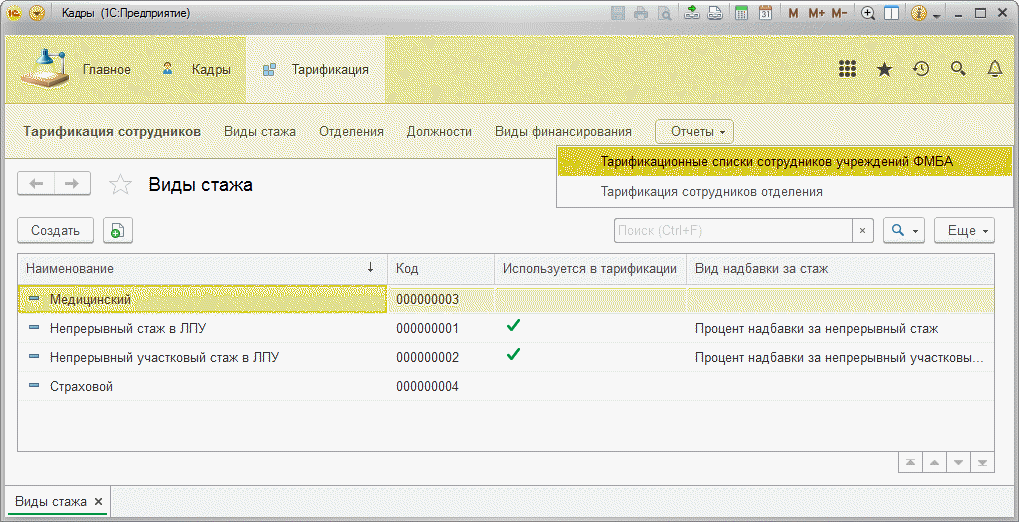
**Рис. 2.103.**Командный интерфейс подсистемы «Предприятие» при запуске приложения

Однако, несмотря на то что у справочников не включена принадлежность к заимствованной подсистеме, команды для их открытия все равно попали в итоговый командный интерфейс из расширяемой подсистемы. Можно было их и не заимствовать (мы сделали это исключительно в качестве примера).

### Как создать собственную подсистему

Пример создания собственной подсистемы *Тарификация* в расширении, которое дополняет типовую функциональность, подробно рассмотрен в четвертой главе [«Расширение для тарификации сотрудников государственных медицинских учреждений»](#_Глава_4._Расширение).

Эта подсистема содержит как собственные объекты расширения (документ *Тарификация сотрудников*; справочник *Виды финансирования*; отчеты *Тарификационные списки сотрудников учреждений ФМБА*, *Тарификация сотрудников отделения*), так и заимствованные из конфигурации объекты (справочники *Виды стажа*, *Отделения*, *Должности*), рис. 2.104.



**Рис. 2.104.**Командный интерфейс подсистемы «Тарификация»

## Как доработать начальную страницу

С помощью расширения можно доработать начальную страницу конфигурации. Вы можете заимствовать форму объекта и поместить ее на начальную страницу в расширении или заимствовать объект, создать его форму и поместить ее на начальную страницу.

При этом состав форм на начальной странице конфигурации дополняется собственными или заимствованными формами, помещенными на начальную страницу в расширении. Удалить формы с расширяемой начальной страницы или поменять их расположение вы не сможете.

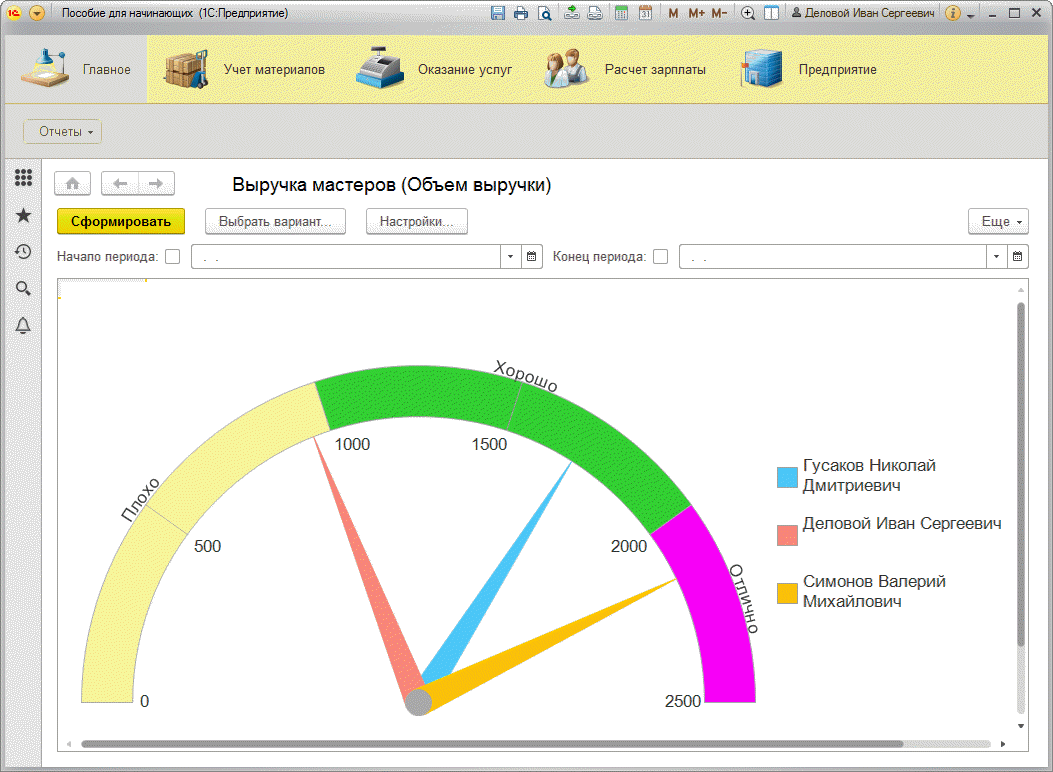
Рассмотрим пример. Пусть в конфигурации существуют роли *Администратор*, *Директор* и *Расчетчик*. Видимость форм на начальной странице настроена по ролям. И хотя наличие ролей прямого отношения к теме примера не имеет, мы покажем пример доработки начальной страницы с настройкой ролевой видимости, так как это наиболее реальная ситуация.

Для пользователя с ролью *Расчетчик* на начальную страницу приложения выводится список документов *Начисления сотрудникам* (рис. 2.105).



**Рис. 2.105.**Начальная страница пользователя с ролью «Расчетчик»

Для пользователя с ролью *Директор* на начальную страницу выводится отчет *Выручка мастеров* (рис. 2.106).

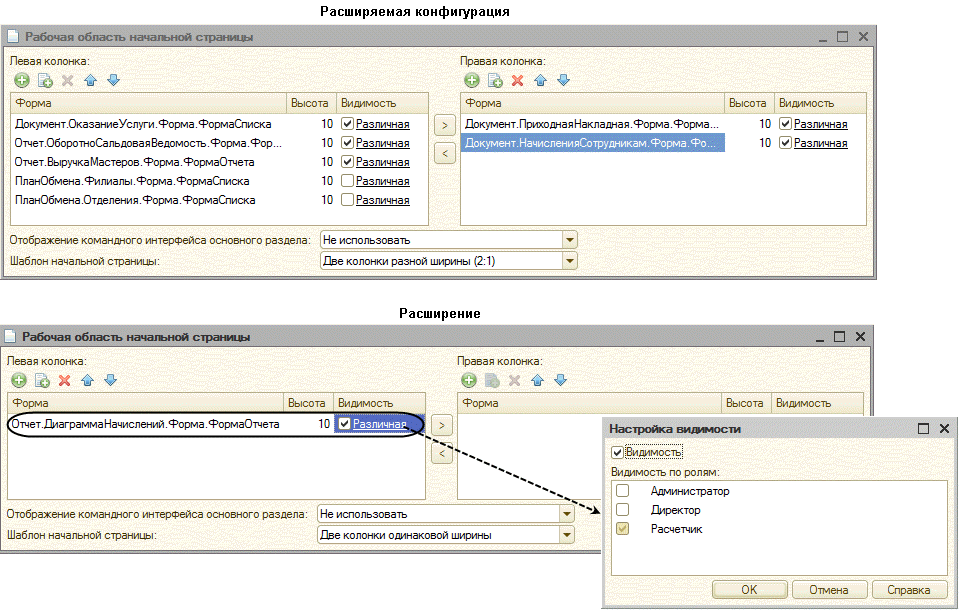


**Рис. 2.106.**Начальная страница пользователя с ролью «Директор»

С помощью расширения мы хотим поместить на начальную страницу пользователя с ролью *Расчетчик* отчет *ДиаграммаНачислений*.

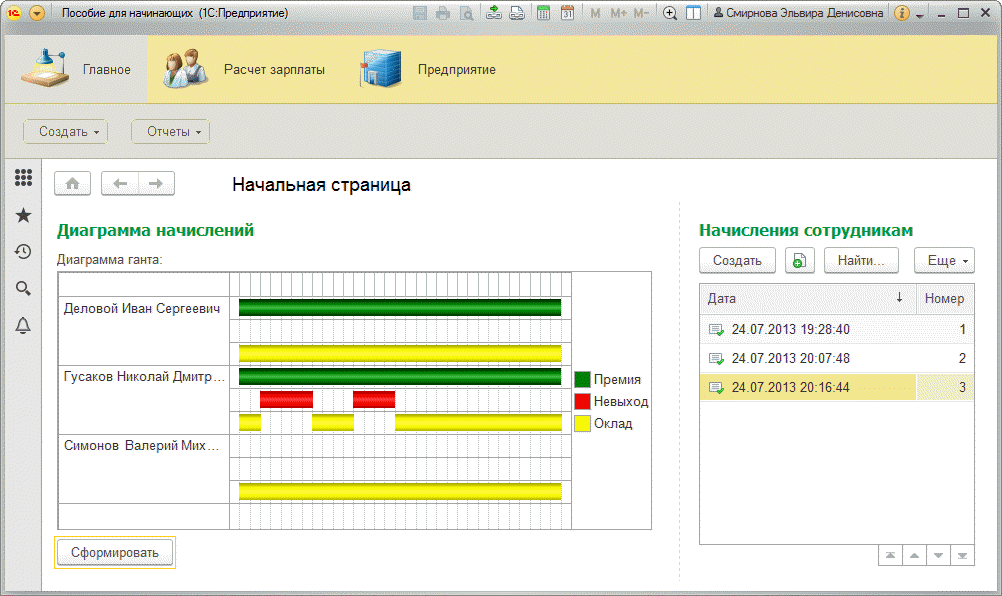
Этот отчет имеет форму в конфигурации. Позаимствуем ее. Затем выделим корень дерева объектов расширения и выполним команду *Открыть рабочую область начальной страницы*. Поскольку форма списка документа *НачисленияСотрудникам* находится в правой части расширяемой начальной страницы (см. вверху на рис. 2.107), поместим заимствованную форму отчета *ДиаграммаНачислений* слева на начальной странице в расширении (см. внизу на рис. 2.107).

На отчет *ДиаграммаНачислений* имеют права роли *Администратор*, *Директор* и *Расчетчик*. Позаимствуем эти роли и снимем видимость формы отчета на начальной странице в расширении у ролей *Администратор* и *Директор*, чтобы отчет отображался только для пользователя с ролью *Расчетчик* (рис. 2.107).



**Рис. 2.107.**Настройка начальной страницы в расширении

В результате для пользователя с ролью *Расчетчик* слева на начальной странице будет выводиться отчет *Диаграмма начислений* (рис. 2.108).

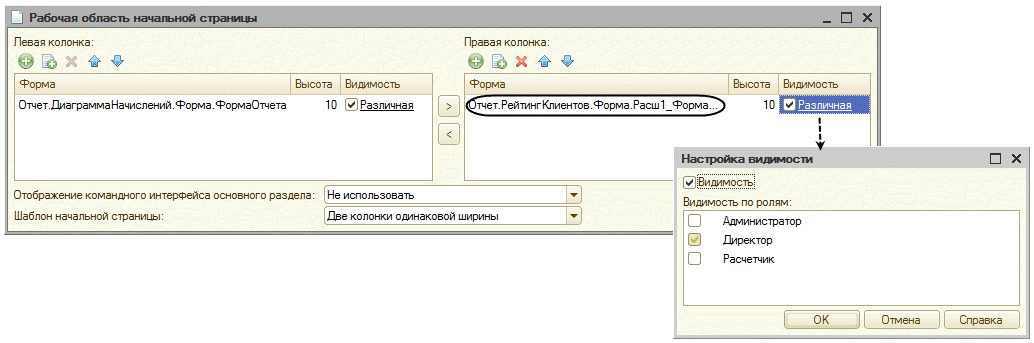


**Рис. 2.108.**Начальная страница пользователя с ролью «Расчетчик»

Теперь мы хотим поместить на начальную страницу пользователя с ролью *Директор* отчет *РейтингКлиентов*.

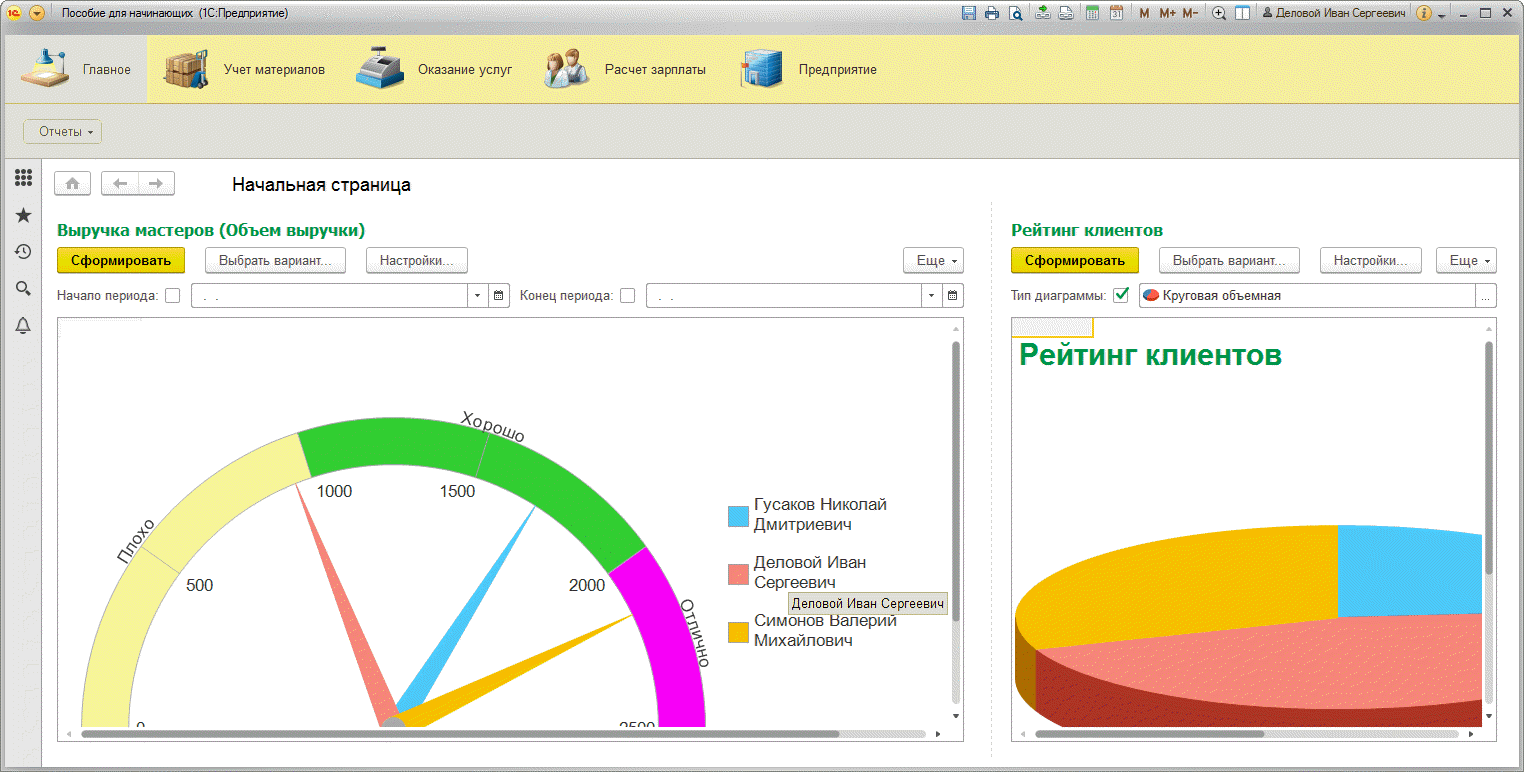
У этого отчета нет своей формы в конфигурации. Создадим ее. Для этого позаимствуем отчет *РейтингКлиентов* в расширение. В свойствах заимствованного отчета установим флажок модифицированности у свойства *Основная форма* и создадим форму отчета с установленным флажком *Использовать систему компоновки данных*.

Откроем форму настройки начальной страницы в расширении и поместим только что созданную форму отчета в правую часть начальной страницы. Снимем видимость формы отчета на начальной странице в расширении у ролей *Администратор* и *Расчетчик*, чтобы отчет отображался только для пользователя с ролью *Директор* (рис. 2.109).



**Рис. 2.109.**Настройка начальной страницы в расширении

В результате для пользователя с ролью *Директор* справа на начальной странице будет выводиться отчет *Рейтинг клиентов* (рис. 2.110).



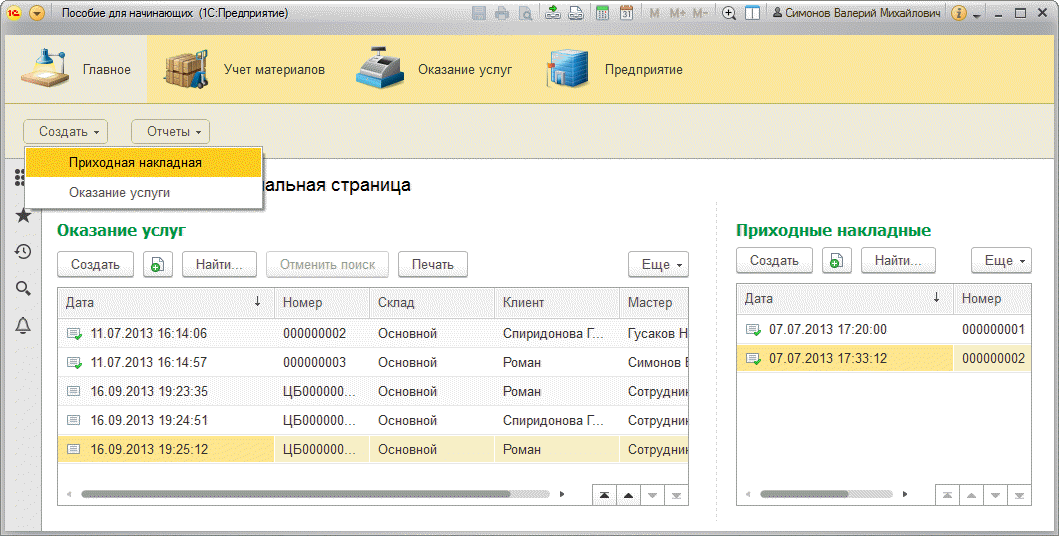
**Рис. 2.110.**Начальная страница пользователя с ролью «Директор»

Как вы видите, пропорция разбиения на части у итоговой начальной страницы (2:1) берется из конфигурации, а не из расширения (1:1, см. рис. 2.107).

## Как доработать интерфейс основного раздела

С помощью расширения можно доработать командный интерфейс основного раздела конфигурации. Вы можете добавить в основной раздел собственные или заимствованные команды или отключить (сделать невидимыми) какие-то существующие в конфигурации команды основного раздела.

Рассмотрим пример. Пусть в конфигурации существует роль *Мастер*. Видимость команд основного раздела настроена по ролям. Для пользователя с ролью *Мастер* основной раздел приложения содержит команды для открытия отчетов и команды для создания приходных и расходных накладных (рис. 2.111).



**Рис. 2.111.**Команды основного раздела для пользователя с ролью «Мастер»

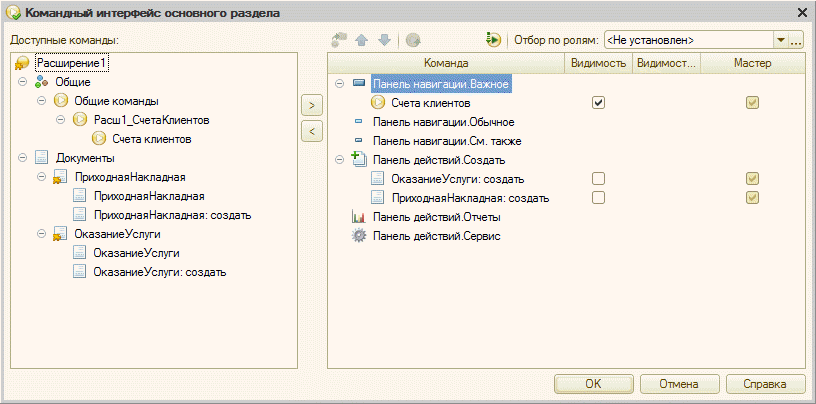
С помощью расширения мы хотим убрать из основного раздела команды для создания приходных и расходных накладных, так как эти документы можно создать прямо на начальной странице приложения. Кроме того, мы хотим добавить в основной раздел для пользователей с ролью *Мастер* собственную команду из расширения для выставления счетов клиентам.

Для этого сначала позаимствуем в расширение подсистему *ОказаниеУслуг*.

Добавим в расширение общую команду *Расш1\_СчетаКлиентов* с синонимом «Счета клиентов», зададим у нее свойство *Группа* – *Панель навигации.Важное* и укажем, что она будет входить в состав заимствованной подсистемы *ОказаниеУслуг*. Затем позаимствуем роль *Мастер* и дадим ей права на эту команду.

Далее позаимствуем в расширение документы *ПриходнаяНакладная* и *ОказаниеУслуг*.

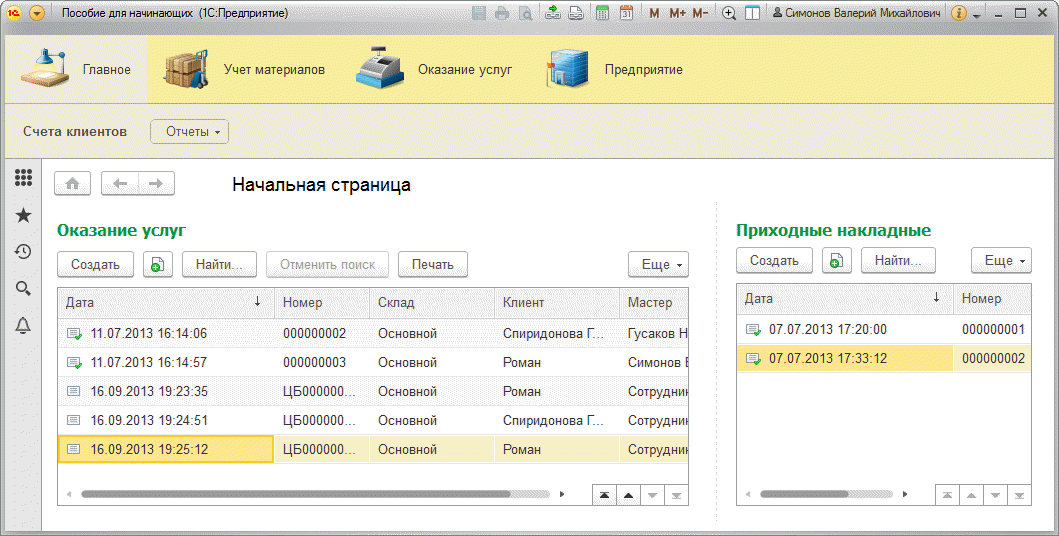
Теперь выделим корень дерева объектов расширения и выполним команду *Открыть командный интерфейс основного раздела*. Перенесем из списка доступных команд в правую часть формы настройки основного раздела команду *Расш1\_СчетаКлиентов* и команды для создания заимствованных документов *ПриходнаяНакладная* и *ОказаниеУслуг* (рис. 2.112).



**Рис. 2.112.**Командный интерфейс основного раздела в расширении

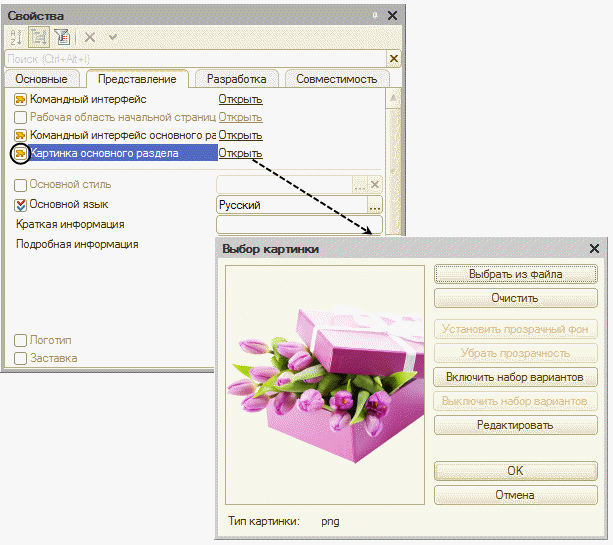
Стандартно видимость общей команды включена, а команды для создания документов невидимы. Но мы все равно должны явно добавить команды для создания документов в основной раздел расширения, так как в конфигурации эти команды для роли *Мастер* видимы.

В результате для пользователя с ролью *Мастер* основной раздел приложения будет содержать команды для открытия отчетов и новую команду для выставления счетов клиентам. Команды для создания документов видны не будут, так как это задано в расширении (рис. 2.113).



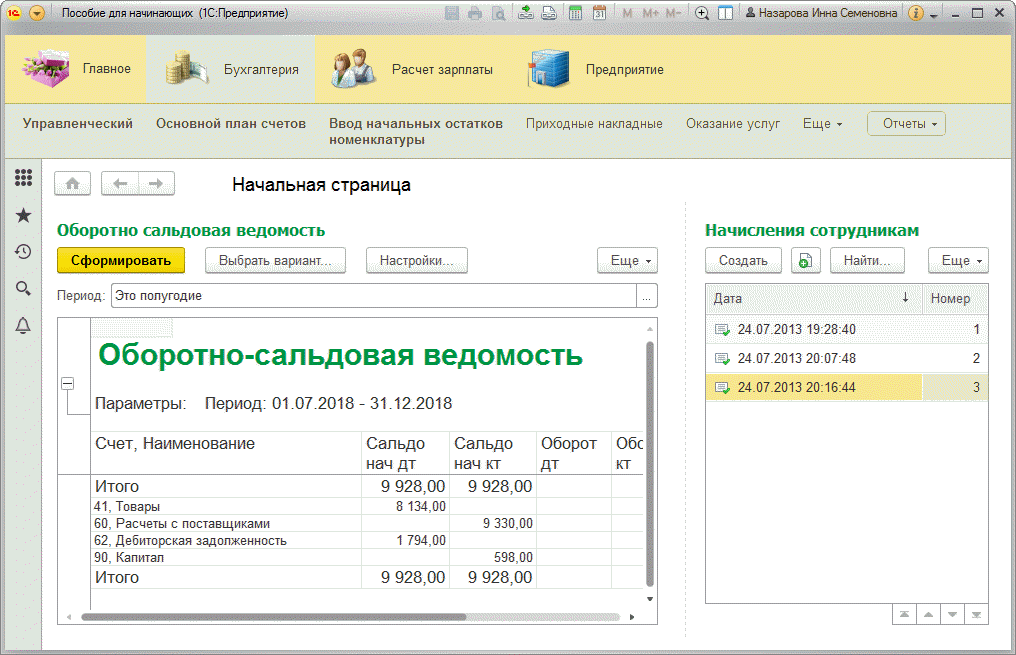
**Рис. 2.113.**Команды основного раздела для пользователя с ролью «Мастер»

Кроме того, в расширении можно установить картинку основного раздела, и она полностью заменит картинку из расширяемой конфигурации, если она там была установлена. Для этого нужно установить флажок модифицируемости у свойства расширения *Картинка основного раздела* и выбрать изображение картинки из файла (рис. 2.114).



**Рис. 2.114.**Изменение картинки основного раздела в расширении

Например, руководство решило сделать к празднику сюрприз своим сотрудницам из бухгалтерии и установило для типовой бухгалтерии предприятия расширение, где на картинке основного раздела изображен букет цветов. В результате, начав работать с программой, сотрудницы бухгалтерии вместо поднадоевшей и будничной картинки увидят на своей начальной странице праздничный вариант (рис. 2.115).



**Рис. 2.115.**Картинка основного раздела, измененная расширением

Через некоторое время расширение можно отключить и вернуться к обычному режиму работы.

Следует иметь в виду, что в качестве картинки основного раздела (как и для всех картинок, изменяемых в расширении) будет применяться та картинка, которая установлена в последнем подключившемся к конфигурации расширении.

## Стиль оформления прикладного решения

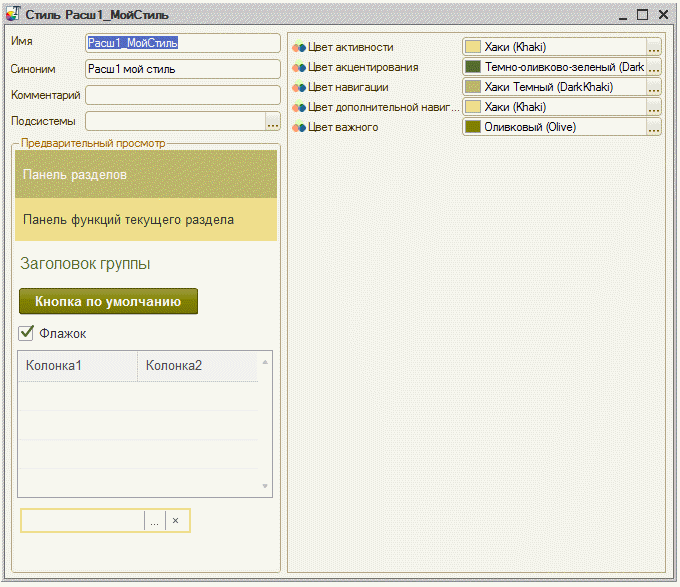
Прикладные решения «1С:Предприятия» основаны на интерфейсе «Такси», поэтому они имеют единый узнаваемый интерфейс (см. рис. 2.115, 2.113). Но не всегда эта узнаваемость бывает нужна. Например, в организации принят единый корпоративный стиль оформления используемых программных продуктов, и требуется, чтобы прикладное решение «1С:Предприятия» также было оформлено в этом стиле. Или же просто у заказчика свои специфичные эстетические вкусы насчет внешнего вида прикладного решения.

С помощью расширения (начиная с версии платформы 8.3.13) можно легко решить задачу по доработке стиля оформления прикладного решения. Причем можно создать в расширении собственный стиль прикладного решения и назначить его в качестве основного стиля приложения или же можно заимствовать и доработать в расширении уже существующий стиль прикладного решения. Рассмотрим оба эти варианта.

### Как создать собственный стиль оформления

Для создания собственного стиля оформления в расширении (так же, как и в конфигурации) предназначена ветка объектов дерева расширения *Общие – Стили*. Работа со стилями в расширении ничем не отличается от работы в конфигурации.

Добавим в расширение объект *Стиль* с именем *Расш1\_МойСтиль* (рис. 2.116).

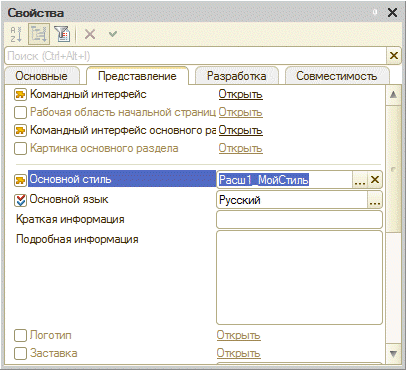


**Рис. 2.116.**Создание стиля в расширении

В свойствах стиля мы можем задать пять основных стилевых цветов:

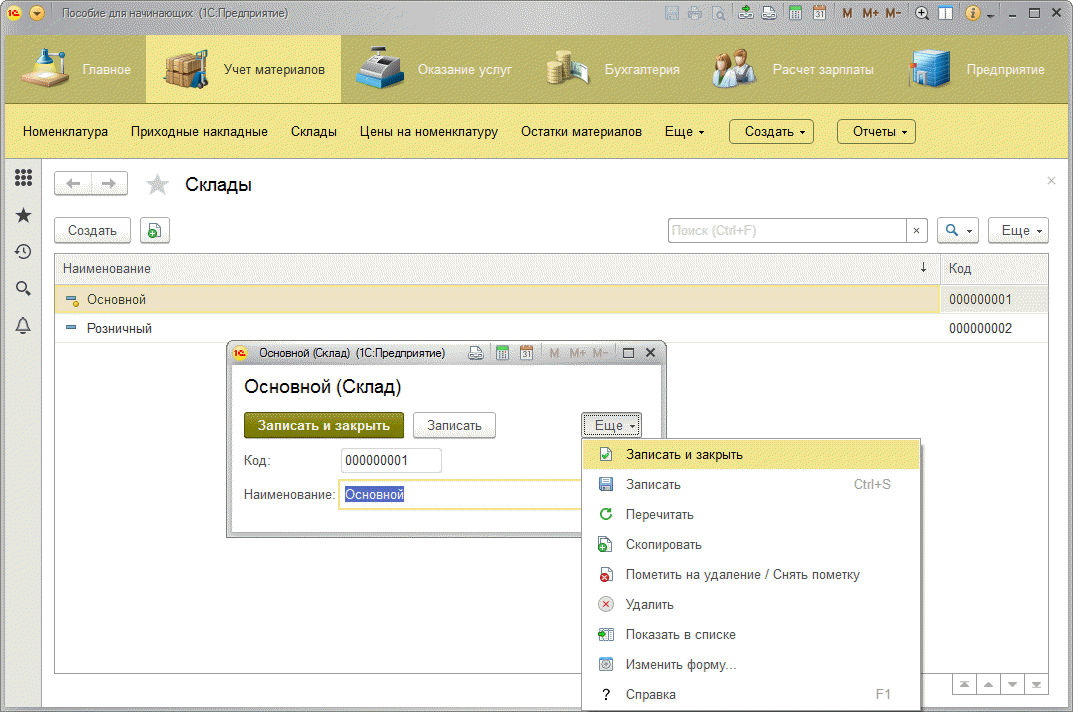
* *Цвет навигации* устанавливает цвет фона панели разделов;
* *Цвет дополнительной навигации* устанавливает цвет фона панели функций текущего раздела;
  + *Цвет активности* применяется к следующим элементам интерфейса:
  + рамка текущего элемента формы,
  + фон текущего элемента меню,
  + фон текущей строки и ячейки таблицы;
* *Цвет важного* устанавливает цвет фона кнопки по умолчанию;
* *Цвет акцентирования* применяется к следующим элементам интерфейса:
  + текст заголовка группы,
  + отметка флажка, переключателя и полосы регулирования,
  + индикатор текущей формы в панели открытых.

Затем установим флажок модифицируемости у свойства расширения *Основной стиль* и выберем ранее созданный стиль *Расш1\_МойСтиль* (рис. 2.117).



**Рис. 2.117.**Установка свойства расширения «Основной стиль»

В результате применения расширения стиль оформления прикладного решения изменится следующим образом (рис. 2.118).



**Рис. 2.118.**Стиль оформления прикладного решения, измененный расширением

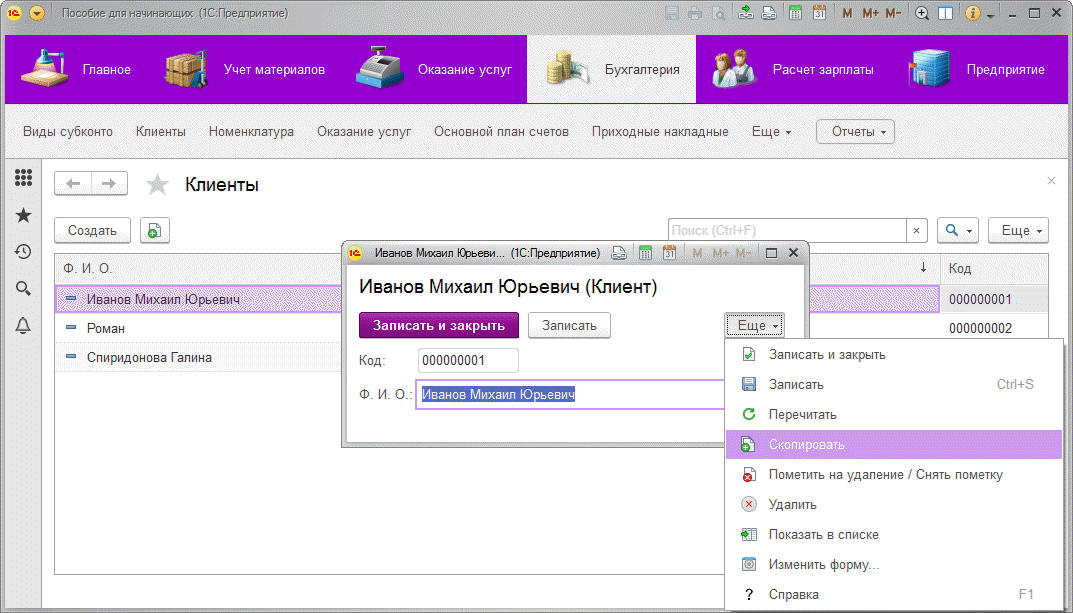
Обратите внимание, что для облегчения вашей работы платформа берет на себя некоторые дополнительные действия, связанные с выбранными вами цветами. Например, если для фона вы выбираете достаточно темный цвет, то платформа самостоятельно инвертирует цвет текста так, чтобы он хорошо читался на этом фоне. Кроме этого, часть цветов платформа рассчитывает от стилевых (выбранных вами), например: цвет рамки кнопок, цвет текущей строки неактивной таблицы и тому подобное.

Задать основной стиль приложения также можно с помощью встроенного языка. Для этого нужно в обработчике события *УстановкаПараметровСеанса()* установить свойство глобального контекста *ГлавныйСтиль*.

### Как доработать стиль оформления

Теперь рассмотрим вариант, когда у прикладного решения уже есть свой стиль оформления и нам нужно доработать его в расширении.

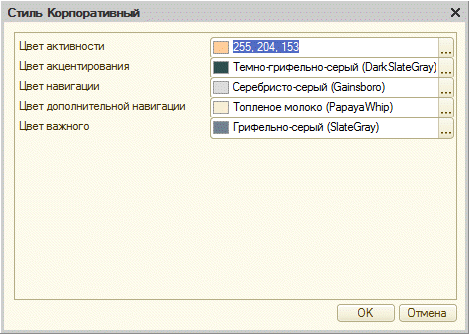
Пусть в расширяемой конфигурации создан стиль *Корпоративный*, и он выбран в свойстве *Основной стиль* конфигурации. В результате прикладное решение оформлено в фиолетовых тонах (рис. 2.119).



**Рис. 2.119.**Внешний вид прикладного решения без расширения

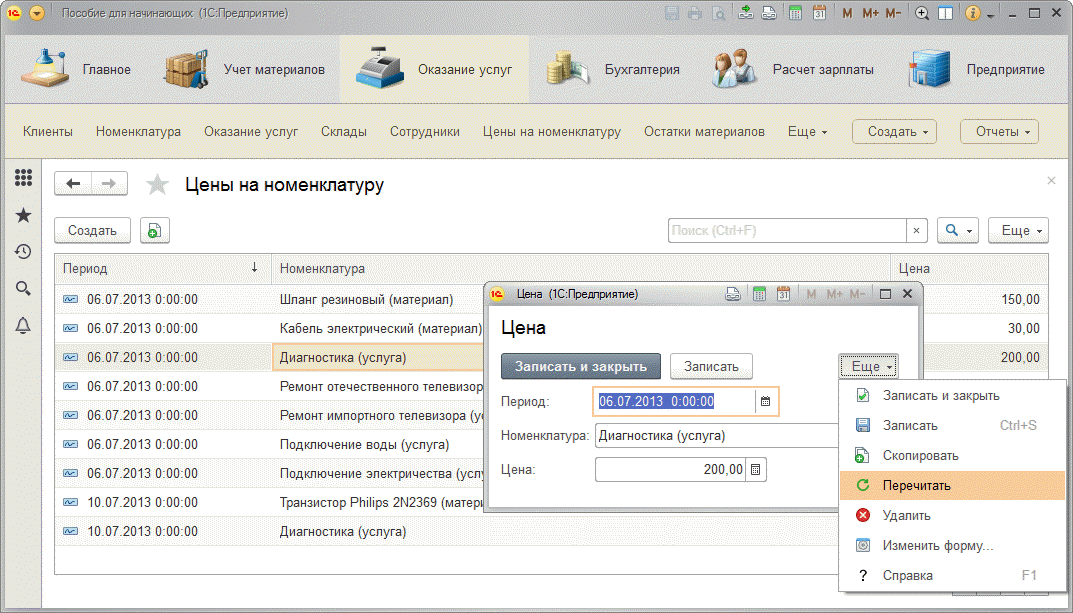
Предположим, новое руководство хочет, чтобы у приложения был более спокойный, но свой, индивидуальный стиль оформления. Сделать это очень просто.

Позаимствуем стиль *Корпоративный* в расширение и изменим его стилевые цвета следующим образом (рис. 2.120).



**Рис. 2.120.**Изменение стиля «Корпоративный» в расширении

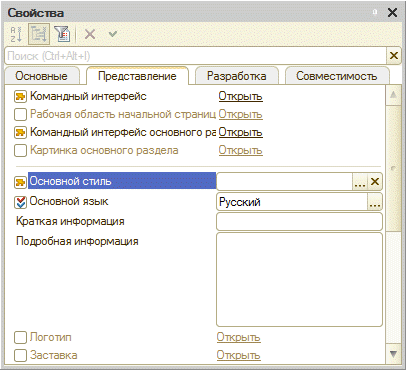
В результате применения расширения прикладное решение будет оформлено в серо-розовой гамме (рис. 2.121).



**Рис. 2.121.**Стиль оформления прикладного решения, измененный расширением

Заметьте, что мы не устанавливали стиль *Корпоративный* в свойстве расширения *Основной стиль*, так как это уже задано в расширяемой конфигурации.

А вот если установить флажок модифицируемости у этого свойства, а значение свойства оставить пустым (рис. 2.122), то прикладное решение примет привычный стандартный вид интерфейса «Такси».



**Рис. 2.122.**Установка свойства расширения «Основной стиль»

Так произойдет потому, что мы задали, что основной стиль берется из расширения, а сам стиль не указали.

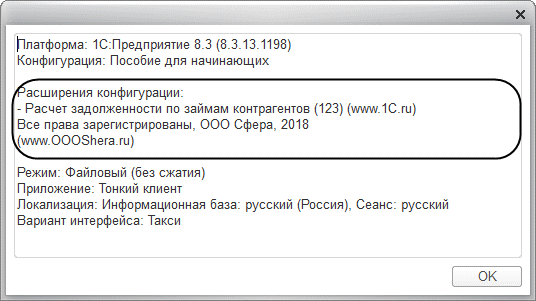
### Брендирование расширений

Возможность брендирования расширений не связана непосредственно со стилем прикладного решения, но позволяет вам обозначить авторство своего расширения и дать ссылки на подробную информацию, связанную с ним. Тем самым информация о прикладном решении с примененным к нему расширением выглядит более законченной и профессиональной.

В свойствах расширения можно указать следующие поля:

* *Краткая информация* и *Подробная информация* позволяют задать описание расширения;
* *Авторские права* позволяет указать информацию о разработчике расширения;
* *Адрес информации о поставщике* позволяет задать URL, перейдя по которому, можно будет получить более подробную информацию о поставщике расширения;
* *Адрес информации о конфигурации*позволяет задать URL, перейдя по которому, можно получить более подробную информацию о данном расширении;
* *Версия* позволяет задать версию расширения.

Информация из этих полей отображается в окне *О программе* для каждого из установленных расширений (рис. 2.123).



**Рис. 2.123.**Окно информации о программе

## Расширение данных

До сих пор мы рассматривали расширение таких элементов конфигурации, которые не связаны с хранением прикладных данных. Доработка этих элементов никак не отражалась на структуре базы данных и на составе прикладных данных.

Это были либо общие объекты конфигурации (роли, подсистемы), либо подчиненные объекты метаданных (модули, формы, макет табличного документа).

Все эти элементы важны и используются часто, однако основной «костяк» любой конфигурации составляют прикладные объекты конфигурации, такие как справочники, документы, регистры и другие. Общей особенностью этих объектов является то, что помимо прочего они описывают структуры данных, которые платформа «1С:Предприятие» создает для хранения прикладных данных. Именно в этих структурах (таблицах базы данных) будут храниться данные, введенные пользователями.

Доработка прикладных объектов с помощью расширения подразумевает, что в результирующей конфигурации, например, справочник будет иметь не только те реквизиты, которые содержатся в конфигурации, но и ряд других реквизитов, которые добавлены расширением. А это значит, что в базе данных, с которой работает такая конфигурация, таблица справочника должна содержать дополнительные колонки, соответствующие реквизитам из расширения.

Получается, что, расширяя, например, справочник, вы добавляете не только новую функциональность к тем метаданным, которые описаны в конфигурации, но и добавляете возможность хранения новых прикладных данных, которых нет в исходной конфигурации.

Именно поэтому, говоря о расширении прикладных объектов конфигурации, мы будем говорить о расширении данных, так как основная цель доработки прикладных объектов заключается в том, чтобы хранить «дополнительные» данные, на которые «не рассчитана» расширяемая конфигурация.

На момент написания книги (версия платформы 8.3.13.1472) возможности расширения прикладных объектов конфигурации выглядят следующим образом.

Можно добавлять следующие собственные объекты:

* планы обмена,
* справочники,
* документы,
* перечисления,
* планы видов характеристик,
* планы счетов,
* планы видов расчета,
* регистры сведений,
* регистры накопления (не поддерживается создание агрегатов),
* регистры бухгалтерии,
* регистры расчета.

В заимствованные планы обмена можно добавлять реквизиты, табличные части и реквизиты табличных частей.

В заимствованные справочники и документы можно добавлять реквизиты и табличные части.

Документы могут формировать движения по любым регистрам, за исключением заимствованных оборотных регистров накопления с включенными агрегатами.

Поскольку расширение прикладных объектов конфигурации ведет к появлению новых данных, то к удалению таких расширений нужно подходить с особенной осторожностью. При удалении расширения данные, добавленные расширением, тоже будут удалены. Поэтому вместо удаления в большинстве случаев лучше использовать деактивацию расширения. Об этом подробно рассказывалось в разделе [«Применение расширения у заказчика – Деактивация»](#_Деактивация).

Ниже мы рассмотрим небольшие примеры, показывающие добавление в расширение собственных структур хранения данных: справочника, плана вида характеристик, регистра сведений, документа, регистра накопления. Причем документ из расширения будет производить движения в заимствованном из расширяемой конфигурации регистре накопления, а заимствованный документ – в собственном регистре из расширения.

Подробный пример с расширением объектов бухгалтерского учета (плана счетов, регистра бухгалтерии) показан в третьей главе книги [«Расширение для ведения управленческого учета на предприятии»](#_Глава_3._Расширение).

### Как реализовать учет по характеристикам в расширении

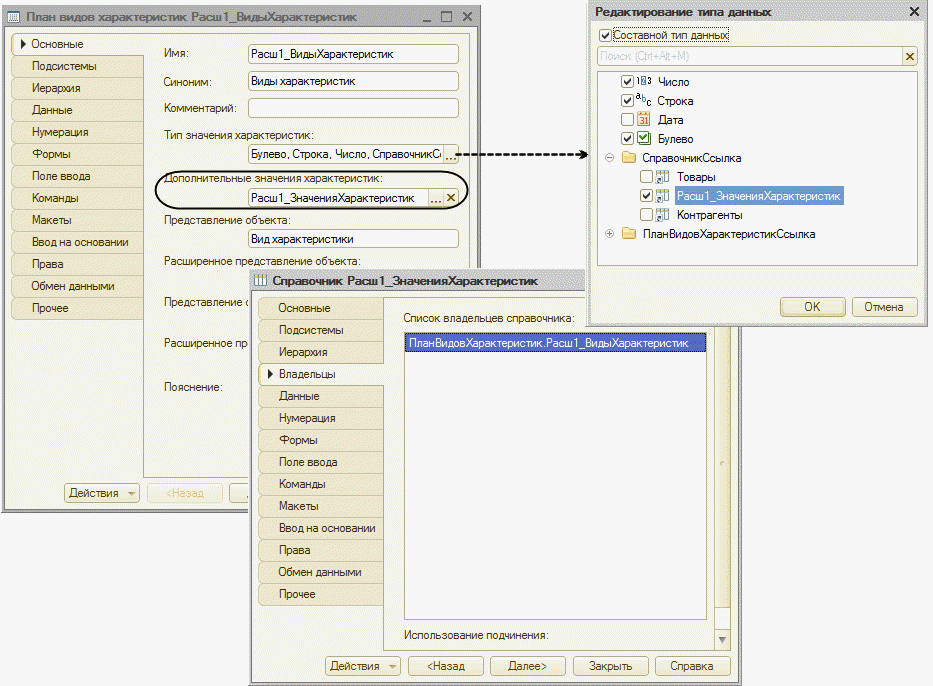
Предположим, что в прикладном решении требуется учитывать характеристики товаров, выводить их в отчеты и т. п. С появлением расширения данных такая задача больше не является проблемой.

Нужно только добавить в расширение собственные структуры данных: план видов характеристик для хранения видов характеристик товаров, справочник для хранения дополнительных значений видов характеристик (для которых нет подходящих типов в конфигурации) и регистр сведений для хранения значений характеристик товаров. Затем связать их нужным образом и вывести в отчет.

Итак, добавим в расширение собственный план видов характеристик *Расш1\_ВидыХарактеристик*.

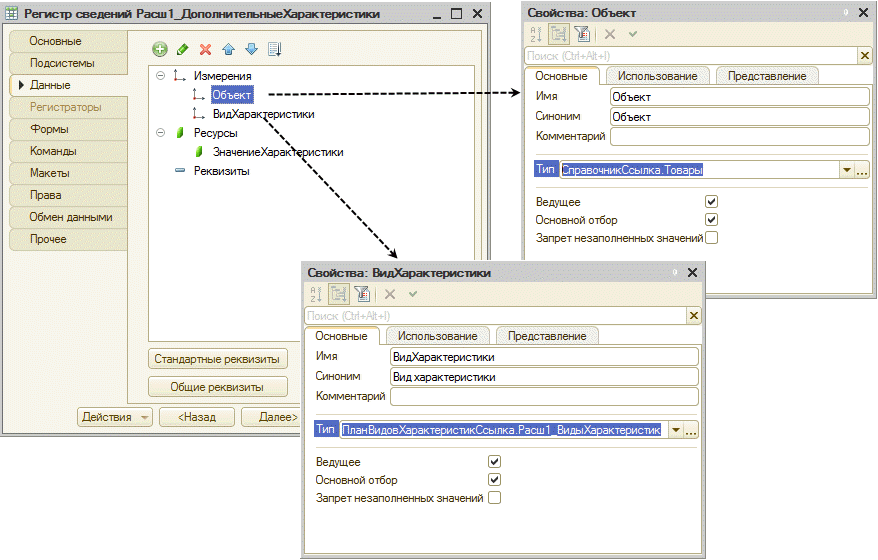
Затем добавим в расширение собственный справочник *Расш1\_ЗначенияХарактеристик*. В качестве владельца справочника установим наш созданный ранее план видов характеристик.

Теперь в свойстве плана видов характеристик *Дополнительные значения характеристик* выберем наш ранее созданный справочник, а в свойстве *Тип значения* *характеристик* укажем составной тип данных, куда также входит ссылка на этот справочник (рис. 2.124).



**Рис. 2.124.**Связь плана видов характеристик со справочником дополнительных значений видов характеристик

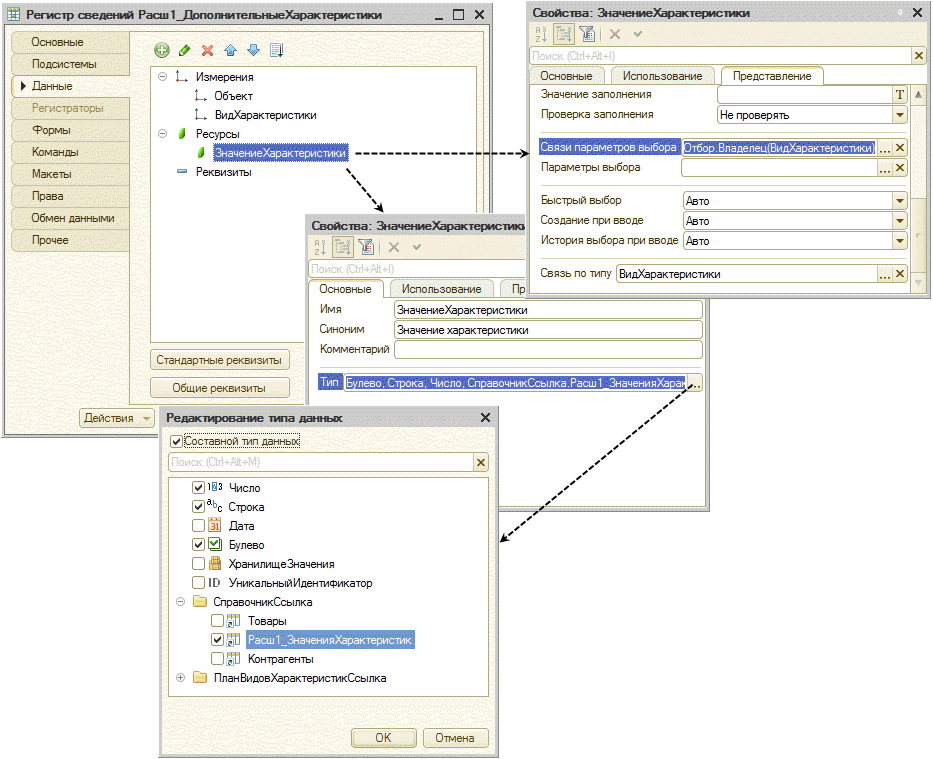
Затем добавим в расширение собственный регистр сведений *Расш1\_ДополнительныеХарактеристики*. Создадим измерение регистра *Объект* типа *СправочникСсылка.Товары* со свойством *Ведущее* и измерение *ВидХарактеристики* типа *ПланВидовХарактеристикСсылка.Расш1\_ВидыХарактеристик*со свойством *Ведущее* (рис. 2.125).



**Рис. 2.125.**Регистр сведений для хранения значений характеристик товаров

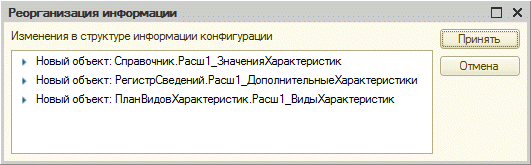
Для того чтобы указать тип значения измерения *Объект*, мы должны позаимствовать в наше расширение справочник *Товары*.

Создадим также ресурс регистра *ЗначениеХарактеристики* такого же составного типа данных, который мы указывали в свойстве *Тип значения* *характеристик* плана видов характеристик *Расш1\_ВидыХарактеристик* (см. рис. 2.124). Кроме того, чтобы при выборе дополнительных значений характеристик из справочника *Расш1\_ЗначенияХарактеристик* предлагались только те значения, которые относятся к выбранному виду характеристик, зададим в свойстве *Связи параметров выбора* отбор по владельцу – измерению регистра *ВидХарактеристики*. В свойстве *Связь по типу* зададим это же измерение регистра (рис. 2.126).



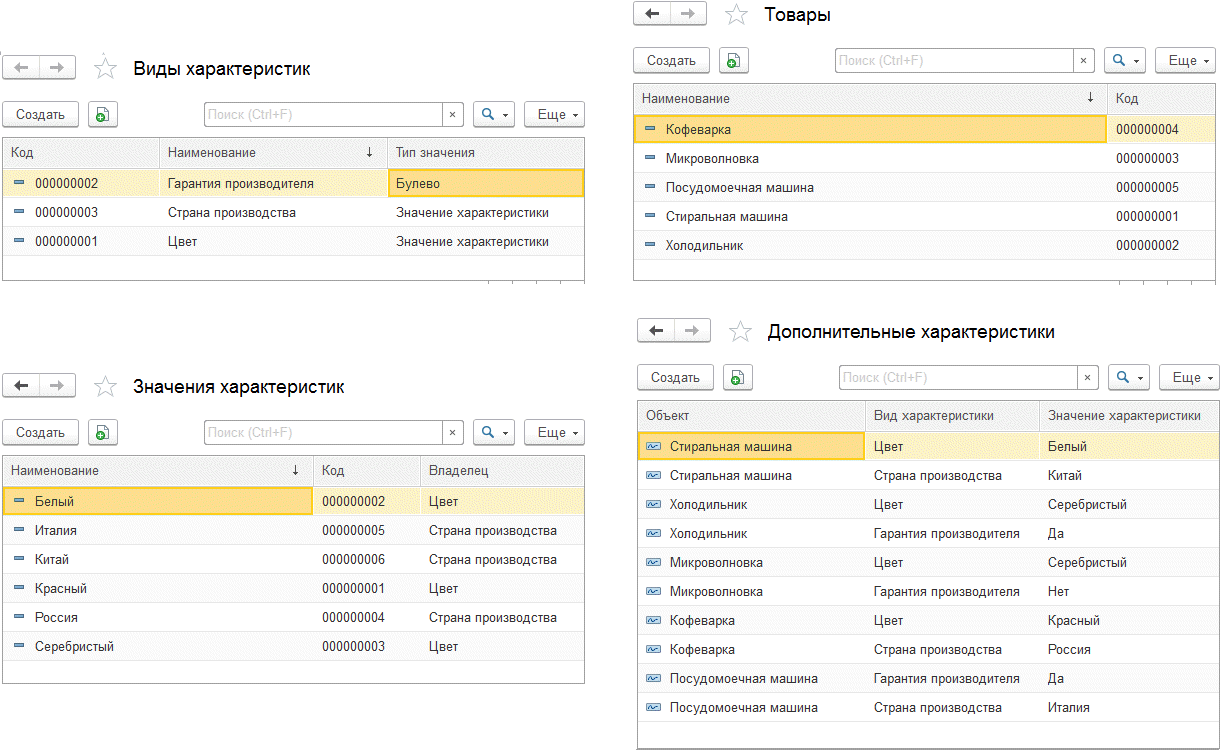
**Рис. 2.126.**Регистр сведений для хранения значений характеристик товаров

Запустим «1С:Предприятие» в режиме отладки. При сохранении расширения в информационной базе будет показан диалог реструктуризации данных, содержащий созданные нами объекты для хранения данных, добавленных расширением. Примем эти изменения (рис. 2.127).



**Рис. 2.127.**Окно реорганизации информации базы данных

В результате для каждого товара и каждого вида характеристик из плана видов характеристик *Виды характеристик* регистр сведений *Дополнительные характеристики* будет хранить значения характеристик, тип которых указан в составе соответствующего плана видов характеристик (рис. 2.128).

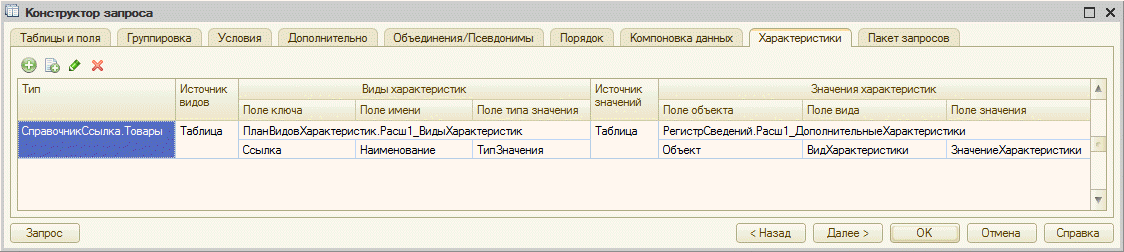


**Рис. 2.128.**Характеристики товаров

Теперь можно использовать эти характеристики товаров в каком-нибудь отчете. Например, в расширяемой конфигурации есть отчет *Продажи*, показывающий данные о продажах товаров покупателям. Выведем в него характеристики проданных товаров.

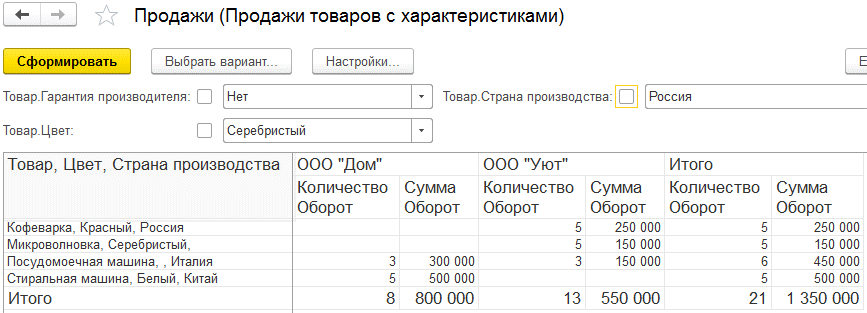
Чтобы доработать отчет в расширении, позаимствуем отчет *Продажи*, скопируем схему компоновки данных отчета в расширение и установим ее в качестве основной схемы компоновки данных для заимствованного отчета. Кроме того, позаимствуем регистр накопления *Продажи*, в котором хранится информация о продажах товаров, вместе с измерениями *Товар*, *Покупатель* и ресурсами *Количество*, *Сумма*.

Затем откроем схему компоновки данных и изменим запрос для получения набора данных отчета. В конструкторе запроса на закладке *Характеристики* опишем таблицы данных для получения характеристик. Укажем, что источник видов характеристик находится в плане видов характеристик *Расш1\_ВидыХарактеристик* (поле ключа – *Ссылка*, поле имени – *Наименование*, поле типа значения – *ТипЗначения*), а источник значений характеристик – в регистре сведений *Расш1\_ДополнительныеХарактеристики* (поле объекта – *Объект*, поле вида – *ВидХарактеристики*, поле значения – *ЗначениеХарактеристики*), (рис. 2.129).



**Рис. 2.129.**Описание характеристик в запросе отчета

В результате можно отобразить характеристики проданных товаров в отчете, установить по ним отбор, условное оформление и т. д. (рис. 2.130).



**Рис. 2.130.**Отчет по продажам товаров с характеристиками

Нужно иметь в виду, что описание характеристик в метаданных в окне *Характеристики* в расширении недоступно. Поэтому, например, вывести характеристики в автоматически генерируемую форму списка товаров не получится. Но можно создать форму списка товаров самостоятельно, установить для динамического списка произвольный запрос и описать в конструкторе запроса, откуда получать характеристики товаров.

### Как сформировать движения собственных и заимствованных документов по собственным и заимствованным регистрам накопления

Пусть у нас есть конфигурация, с помощью которой ведется учет работы интернет-магазина по продаже товаров. Для понимания дальнейшей задачи кратко опишем ее.

Сначала товары поступают на склад, затем создаются заказы покупателей (документы *Заказ*), после доставки товаров и выполнения заказа формируются расходные накладные. Приходные и расходные документы и заказы производят движения в регистрах накопления *ОстаткиТоваров*, *Продажи* и *ПричиныОтказа*, на основе которых затем формируются отчеты о деятельности интернет-магазина.

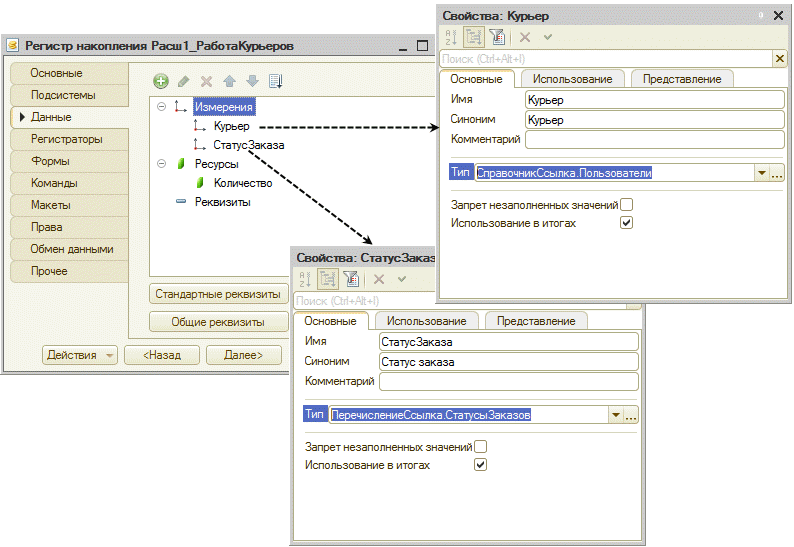
С помощью расширения мы хотим добавить функциональность, которая будет более подробно учитывать работу курьеров. Например, руководство интернет-магазина хочет видеть отчет, показывающий за период, сколько заказов у курьера в работе, сколько выполнено, закрыто и т. п.

Для этого нам понадобится добавить в расширение собственный оборотный регистр накопления *РаботаКурьеров* и сделать так, чтобы существующий в расширяемой конфигурации документ *Заказ* при своем проведении формировал движения по этому регистру, содержащие статусы заказов курьеров (из перечисления *СтатусыЗаказов*). А затем уже на основе данных регистра получить нужный отчет.

Вторая задача, которую нам нужно решить с помощью расширения, – это организовать списание товаров (например, по браку), которое не проводилось раньше. Для этого нам понадобится добавить в расширение собственный документ *СписаниеТоваров* и сделать так, чтобы при своем проведении этот документ уменьшал остатки товаров в существующем в расширяемой конфигурации регистре *ОстаткиТоваров*, то есть формировал движения расхода по этому регистру.

Итак, сначала реализуем в расширении учет работы курьеров.

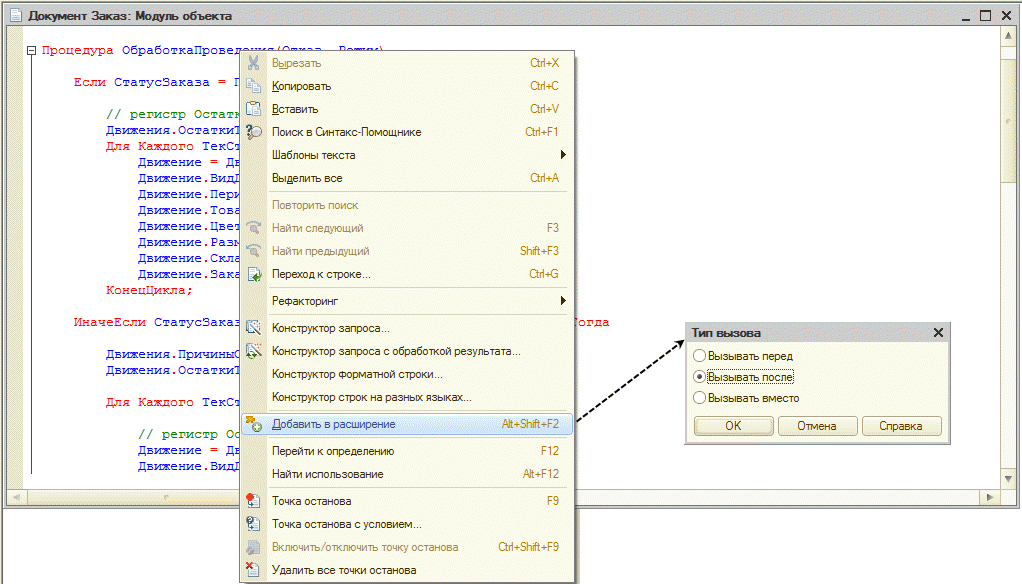
Добавим в расширение собственный регистр накопления оборотов *Расш1\_РаботаКурьеров*. Создадим измерение регистра *Курьер* типа *СправочникСсылка.Пользователи*, измерение *СтатусЗаказа* типа *ПеречислениеСсылка.СтатусыЗаказов*и числовой ресурс *Количество* (рис. 2.131).



**Рис. 2.131.**Измерения регистра «Расш1\_РаботаКурьеров»

Для того чтобы указать типы значений измерений регистра, мы должны позаимствовать в наше расширение справочник *Пользователи* и перечисление *СтатусыЗаказов*.

Затем расширим процедуру *ОбработкаПроведения()* в модуле документа *Заказ* из расширяемой конфигурации. Для этого откроем модуль документа, установим курсор на имя процедуры и вызовем из контекстного меню команду *Добавить в расширение*. Оставим по умолчанию тип расширяющей процедуры (*Вызывать после*) и нажмем *ОК* (рис. 2.132).



**Рис. 2.132.**Создание расширяющей процедуры в модуле документа

В результате документ *Заказ* заимствуется в расширение, а в модуле документа появится шаблон расширяющей процедуры *Расш1\_ОбработкаПроведения()*, предваренный аннотацией *&После("ОбработкаПроведения")*. Заполним эту процедуру следующим образом (листинг 2.34).

**Листинг 2.34.** Процедура «Расш1\_ОбработкаПроведения»

&После("ОбработкаПроведения")

Процедура Расш1\_ОбработкаПроведения(Отказ, Режим)

// регистр РаботаКурьеров

Движения.Расш1\_РаботаКурьеров.Записывать = Истина;

Движение = Движения.Расш1\_РаботаКурьеров.Добавить();

Движение.Период = ДатаДоставки;

Движение.Курьер = Курьер;

Движение.СтатусЗаказа = СтатусЗаказа;

Движение.Количество = 1;

КонецПроцедуры

Таким образом, при проведении заказа сначала отработает обработка проведения документа из расширяемой конфигурации, а затем обработка проведения документа из расширения, в которой движения документа записываются в регистр накопления *Расш1\_РаботаКурьеров*.

На закладке *Регистраторы* в окне редактирования свойств этого регистра укажем, что заимствованный документ *Заказ* будет являться регистратором регистра *Расш1\_РаботаКурьеров*.

Для корректной работы расширения необходимо позаимствовать реквизиты документа *ДатаДоставки*, *Курьер* и *СтатусЗаказа*. Кроме того, необходимо снять флажок *Безопасный режим* у нашего расширения в окне списка расширений конфигурации, так как расширение модулей не будет работать в безопасном режиме в файловом варианте работы.

Затем добавим в расширение отчет *Расш1\_РаботаКурьеров*. Запрос для получения набора данных отчета заполним следующим образом (листинг 2.35).

**Листинг 2.35.** Запрос для получения набора данных отчета «Расш1\_РаботаКурьеров»

ВЫБРАТЬ

Расш1\_РаботаКурьеровОбороты.Курьер КАК Курьер,

Расш1\_РаботаКурьеровОбороты.СтатусЗаказа КАК СтатусЗаказа,

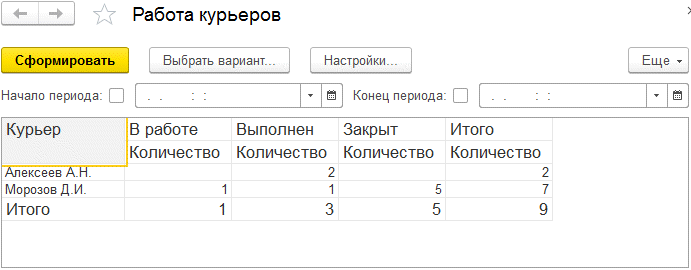
Расш1\_РаботаКурьеровОбороты.КоличествоОборот КАК Количество

ИЗ

РегистрНакопления.Расш1\_РаботаКурьеров.Обороты КАК Расш1\_РаботаКурьеровОбороты

Зададим структуру отчета в виде таблицы, в строках которой записи группируются по курьерам, а в столбцах – по статусам заказов.

В режиме *1С:Преприятие* перепроведем все заказы товаров. После этого, сформировав отчет *Работа курьеров*, мы можем увидеть нужную информацию о работе курьеров с заказами (рис. 2.133).



**Рис. 2.133.**Отчет «Работа курьеров»

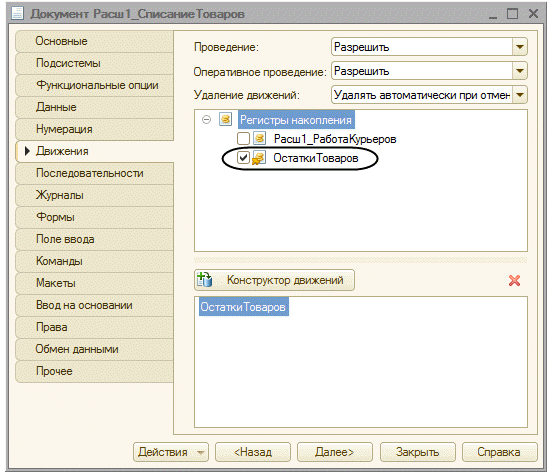
Теперь реализуем в расширении учет списания товаров. Для этого добавим собственный документ *Расш1\_СписаниеТоваров* со следующей структурой:

* Реквизит *Склад*типа *СправочникСсылка.Склады*.
* Табличная часть *Товары* с реквизитами:
  + *Товар* типа *СправочникСсылка.Товары*,
  + *Цвет* типа *СправочникСсылка.Цвета*,
  + *Размер* типа *СправочникСсылка.Размеры*,
  + *Количество* типа *Число 10*.

Для того чтобы указать типы значений реквизитов документа, мы должны позаимствовать в наше расширение справочники *Товары*, *Цвета*, *Размеры* и *Склады*.

Затем позаимствуем из расширяемой конфигурации регистр накопления *ОстаткиТоваров* вместе с измерениями *Товар*, *Цвет*, *Размер*, *Склад*и ресурсом *Количество*.

На закладке *Движения* в окне редактирования свойств документа укажем, что он будет производить движения в регистре *ОстаткиТоваров* (рис. 2.134).



**Рис. 2.134.**Свойства документа «Расш1\_СписаниеТоваров»

Процедуру *ОбработкаПроведения()* в модуле документа заполним следующим образом (листинг 2.36).

**Листинг 2.36.** Процедура «ОбработкаПроведения»

Процедура ОбработкаПроведения(Отказ, Режим)

// регистр ОстаткиТоваров Расход

Движения.ОстаткиТоваров.Записывать = Истина;

Для Каждого ТекСтрокаТовары Из Товары Цикл

Движение = Движения.ОстаткиТоваров.ДобавитьРасход();

Движение.Период = Дата;

Движение.Товар = ТекСтрокаТовары.Товар;

Движение.Цвет = ТекСтрокаТовары.Цвет;

Движение.Размер = ТекСтрокаТовары.Размер;

Движение.Склад = Склад;

Движение.Количество = ТекСтрокаТовары.Количество;

КонецЦикла;

КонецПроцедуры

В режиме *1С:Предприятие* создадим и проведем документы *Списание товаров*. В результате в регистре *ОстаткиТоваров* будут сформированы движения с типом движений *Расход*. Соответственно количество товаров в отчете *Остатки товаров на складах* будет уменьшено на количество списанных единиц.

# Глава 3. Расширение для ведения управленческого учета на предприятии

В этой главе мы рассмотрим несложный пример расширения конфигурации, в которой ведется упрощенный бухгалтерский учет деятельности предприятия по оказанию услуг. В качестве расширяемой конфигурации мы возьмем конфигурацию, описанную в популярной книге «1С:Предприятие 8.3. Практическое пособие» (выгрузка информационной базы «20 занятий», приложенная к книге на компакт-диске).

В этой конфигурации представлен лишь небольшой фрагмент ведения бухгалтерского учета, основной план счетов содержит только четыре счета (хотя в действительности их намного больше), и в регистре бухгалтерии отражаются лишь некоторые хозяйственные операции. Но сути это не меняет.

При проведении первичных документов (документов о поступлении материалов и документов об оказании услуг) в регистре бухгалтерии формируются движения, отражающие бухгалтерские проводки по счетам основного плана счетов. И на основе этих проводок формируется оборотно-сальдовая ведомость по счетам бухгалтерского учета.

Но предположим, в какой-то момент руководство фирмой по оказанию услуг задумалось об оптимизации своей деятельности и решило расширить бухгалтерский учет на предприятии, то есть добавить к нему управленческий учет.

Сначала поясним, что это такое.

## Базовые понятия

*Бухгалтерский учет* является необходимым, он строго регламентирован по формам и срокам предоставления бухгалтерской отчетности о результатах хозяйственной деятельности предприятия, которая представляется внешним пользователям: государству, акционерам, кредиторам и др.

*Управленческий учет* не регламентирован и может вообще не вестись. Управленческий учет позволяет анализировать зависимость между затратами, объемом продаж и прибылью. Он нужен для внутреннего использования руководством предприятия для принятия управленческих решений в целях оптимизации финансового результата хозяйственной деятельности.

## Порядок наших действий

Управленческий учет может вестись как на основе бухгалтерского учета, так и отдельно. Мы рассмотрим первый вариант, так как нам нужно, чтобы при тех же самых действиях пользователя (при проведении первичных документов) параллельно с бухгалтерским учетом велся еще и управленческий учет, отражающий информацию о результатах хозяйственной деятельности предприятия (как правило, более подробно и в других разрезах).

Очевидно, что для этого нужен еще один план счетов и еще один регистр бухгалтерии, которые мы и добавим в своем расширении. А процедуры обработки проведения первичных документов (документов об оказании услуг и документов о поступлении материалов) мы расширим и добавим туда движения по новому (добавленному в расширении) регистру бухгалтерии, содержащие проводки по новому (добавленному в расширении) плану счетов.

В управленческом учете, в расширении, мы выделим из суммы выручки от продажи материалов их себестоимость и будем учитывать эти суммы на разных субсчетах. Кроме того, мы реализуем учет услуг, который просто не велся в расширяемой конфигурации. И также будем учитывать материалы и услуги отдельно, выделяя при этом затраты предприятия на оказание услуг.

Кроме того, по сравнению с планом счетов в расширяемой конфигурации для нового плана счетов мы увеличим количество субконто (т. е. разрезов аналитического учета), чтобы иметь возможность более гибкого анализа хозяйственной деятельности.

Ну, и, конечно, добавим в расширение пару отчетов, чтобы сформировать оборотно-сальдовую ведомость по управленческому учету. На самом деле отчетов может быть гораздо больше – столько, сколько захочет руководство.

В результате без применения расширения расширяемая конфигурация будет работать абсолютно так же, как и раньше. А с расширением наряду с бухгалтерским учетом на предприятии будет вестись управленческий учет, как мы и хотели.

Файлы выгрузки расширяемой конфигурации «БухУчет.dt» и применяемого к ней расширения «УправленческийУчет.cfe» находятся на прилагаемом к книге компакт-диске.

## Расширение для управленческого учета

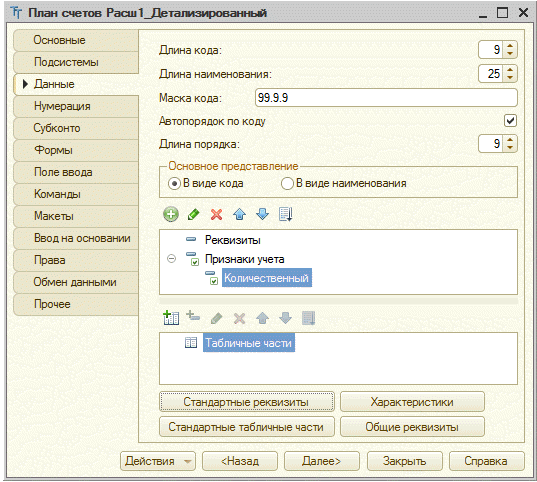
Итак, создадим у конфигурации *БухгалтерскийУчет* расширение с именем *УправленческийУчет* и реализуем в нем всю недостающую функциональность.

Добавим в наше расширение подсистему с именем *Расш1\_УправленческийУчет*и синонимом «Управленческий учет», в которой мы будем размещать все заимствованные и собственные объекты, необходимые для ведения управленческого учета.

### План счетов

Прежде всего в расширении нам понадобится собственный план счетов с более подробной детализацией и большим количеством субконто на счетах. Добавим в расширение план счетов с именем *Расш1\_Детализированный*, синонимом «Детализированный план счетов» и расширенным представлением списка «План счетов управленческого учета». Укажем, что он будет виден в подсистеме *Расш1\_УправленческийУчет*.

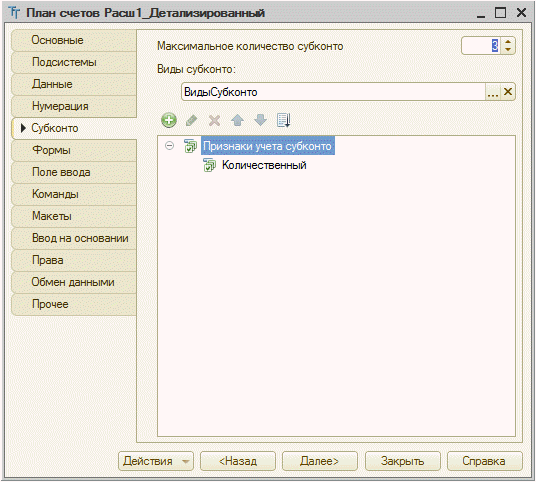
Так как мы хотим детализировать учет, то счета нашего плана счетов будут иметь двухуровневую иерархическую структуру. Чтобы счета было удобнее вводить и сортировать, на закладке *Данные* зададим свойство *Маска кода* как *99.9.9* и установим флажок*Автопорядок по коду*. Чтобы иметь возможность вести на счетах количественный учет, в группе реквизитов *Признаки учета* добавим признак учета *Количественный* (рис. 3.1).



**Рис. 3.1.**Свойства плана счетов «Расш1\_Детализированный» на закладке «Данные»

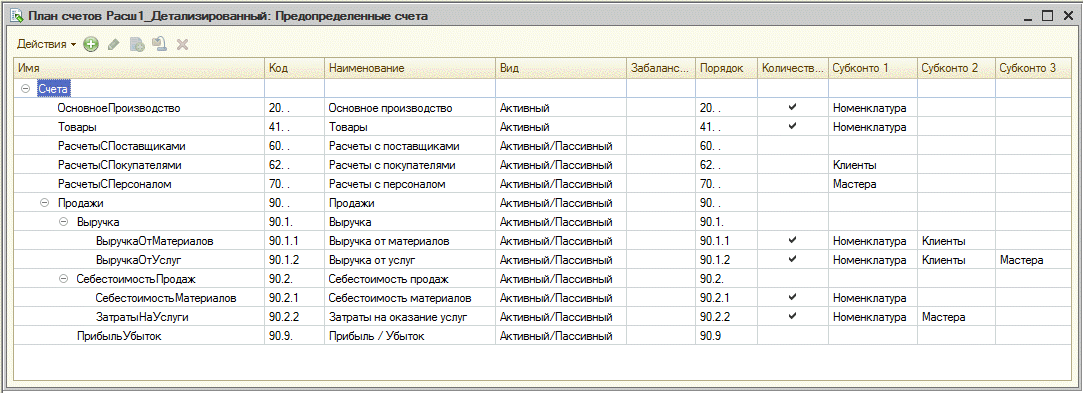
На закладке *Субконто* нам нужно указать, в каком плане видов характеристик будут содержаться виды субконто для нашего плана счетов. Мы будем использовать план видов характеристик *ВидыСубконто*, уже существующий в расширяемой конфигурации. Позаимствуем его в расширение. И не только его, а и его предопределенные элементы: *Номенклатура*, *Клиенты*, *Мастера*. Эти виды субконто уже используются в расширяемой конфигурации, и мы также будем использовать их для управленческого учета.

Выберем заимствованный план видов характеристик в поле *Виды субконто*. Максимальное количество субконто на счете установим равным трем. Также создадим признак учета субконто *Количественный* (рис. 3.2).



**Рис. 3.2.**Свойства плана счетов «Расш1\_Детализированный» на закладке «Субконто»

Теперь на закладке *Прочее* создадим предопределенные счета нашего плана счетов *Расш1\_Детализированный*(рис. 3.3).

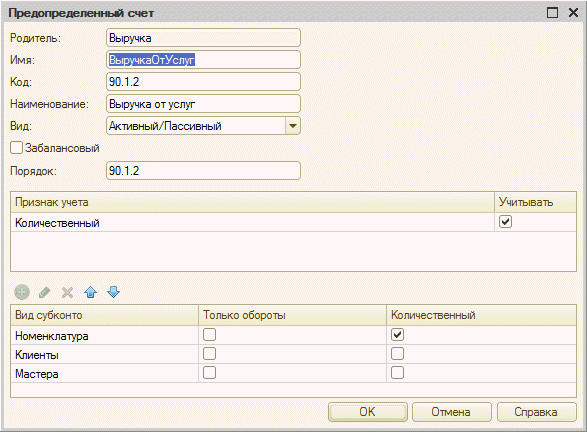


**Рис. 3.3.**Предопределенные счета плана счетов «Расш1\_Детализированный»

Конечно, несмотря на детализацию, счетов будет немного, в отличие от действительности. Но их будет вполне достаточно, чтобы отразить наши хозяйственные операции и продемонстрировать принцип ведения параллельного управленческого учета в расширении.

Итак, добавим следующие предопределенные элементы нашего плана счетов:

* Счет *ОсновноеПроизводство*, код *20*, вид *Активный*, признак учета *Количественный*, первое субконто *Номенклатура* с признаком учета *Количественный*.
* Счет *Товары*, код *41*, вид *Активный*, признак учета *Количественный*, первое субконто *Номенклатура* с признаком учета *Количественный*.
* Счет *РасчетыСПоставщиками*, код *60*, вид *Активный/Пассивный*, субконто на счете не ведутся.
* Счет *РасчетыСПокупателями*, код *62*, вид *Активный/Пассивный*, первое субконто *Клиенты*.
* Счет *РасчетыСПерсоналом*, код *70*, вид *Активный/Пассивный*, первое субконто *Мастера*.
* Счет *Продажи*, код *90*, вид *Активный/Пассивный*.
  + Счет *Выручка*, код *90.1*, вид *Активный/Пассивный*, родитель *Продажи* (счет 90.1 является субсчетом к счету 90).
  + Счет *ВыручкаОтМатериалов*, код *90.1.1*, вид *Активный/Пассивный*, родитель *Выручка* (счет 90.1.1 является субсчетом к счету 90.1). Признак учета *Количественный*, первое субконто *Номенклатура* с признаком учета *Количественный*, второе субконто *Клиенты*.
  + Счет *ВыручкаОтУслуг*, код *90.1.2*, вид *Активный/Пассивный*, родитель *Выручка* (счет 90.1.2 является субсчетом к счету 90.1). Признак учета *Количественный*, первое субконто *Номенклатура* с признаком учета *Количественный*, второе субконто *Клиенты*, третье субконто *Мастера* (рис. 3.4).
  + Счет *СебестоимостьПродаж*, код *90.2*, вид *Активный/Пассивный*, родитель *Продажи* (счет 90.2 является субсчетом к счету 90).
  + Счет *СебестоимостьМатериалов*, код *90.2.1*, вид *Активный/Пассивный*, родитель *СебестоимостьПродаж* (счет 90.2.1 является субсчетом к счету 90.2). Признак учета *Количественный*, первое субконто *Номенклатура* с признаком учета *Количественный*.
  + Счет *ЗатратыНаУслуги*, код *90.2.2*, вид *Активный/Пассивный*, родитель *СебестоимостьПродаж* (счет 90.2.2 является субсчетом к счету 90.2). Признак учета *Количественный*, первое субконто *Номенклатура* с признаком учета *Количественный*, второе субконто *Мастера*.
  + Счет *ПрибыльУбыток*, код *90.9*, вид *Активный/Пассивный*, родитель *Продажи* (счет 90.9 является субсчетом к счету 90).



**Рис. 3.4.**Свойства предопределенного счета «ВыручкаОтУслуг»

Одно из основных направлений управленческого учета – это анализ затрат и их влияния на прибыль в результате хозяйственной деятельности предприятия. Чтобы предоставить руководству предприятия возможность всестороннего анализа, мы детализировали счет *Продажи*. Этот счет имеет двухуровневую иерархическую структуру.

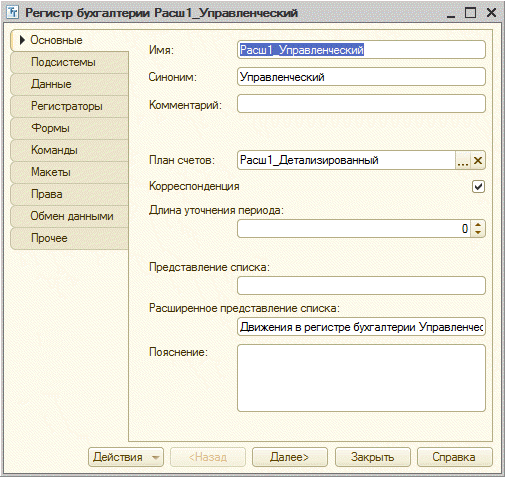
Счет *Продажи* включает в себя субсчета: *Выручка* и *СебестоимостьПродаж*. А каждый из этих субсчетов разбит соответственно еще на два субсчета: *ВыручкаОтМатериалов* и *ВыручкаОтУслуг*, а также *СебестоимостьМатериалов* и *ЗатратыНаУслуги*. Это позволит нам вести учет проданных материалов и оказанных услуг отдельно друг от друга, при этом выделяя суммы затрат из общей суммы выручки.

Кроме того, на счетах нашего плана счетов может вестись до трех субконто, что, безусловно, делает аналитику более развернутой и подробной.

На некоторых счетах и видах субконто включен признак учета *Количественный*. Это позволит нам вести на них количественный учет, например, при учете количества поступившей номенклатуры.

### Регистр бухгалтерии

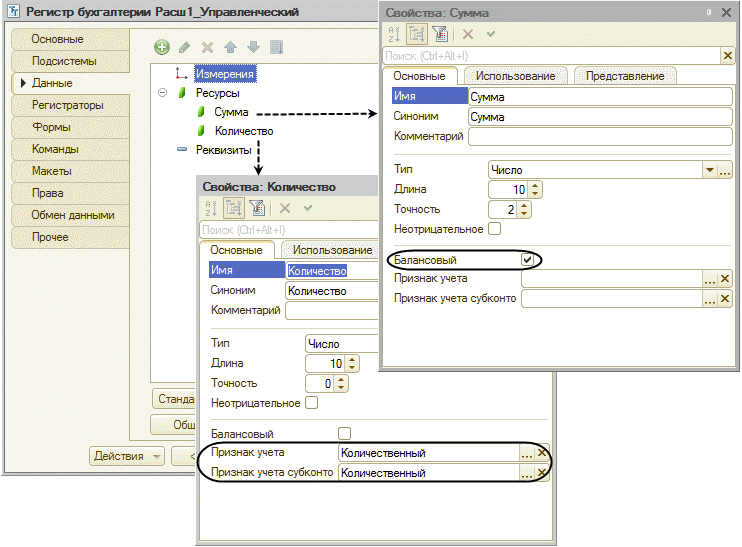
Теперь добавим в расширение регистр бухгалтерии с именем *Расш1\_Управленческий*, синонимом «Управленческий» и расширенным представлением списка «Движения в регистре бухгалтерии Управленческий». Укажем, что он будет использовать план счетов *Расш1\_Детализированный* (ранее добавленный нами в расширение) с установленным флажком *Корреспонденция* (рис. 3.5).



**Рис. 3.5.**Свойства регистра бухгалтерии «Расш1\_Управленческий»

На закладке *Подсистемы* установим видимость регистра бухгалтерии в подсистеме *Расш1\_УправленческийУчет*.

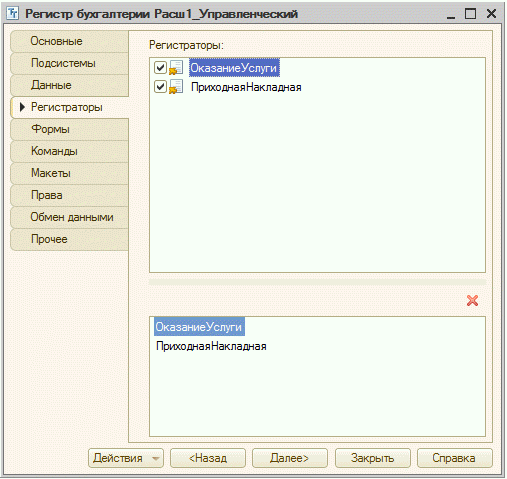
На закладке *Данные* создадим ресурсы регистра: балансовый ресурс *Сумма* (тип *Число* 10, 2) и небалансовый ресурс *Количество* (тип *Число* 10), с признаком учета *Количественный* и признаком учета субконто *Количественный* (рис. 3.6).



**Рис. 3.6.**Ресурсы регистра бухгалтерии «Расш1\_Управленческий»

Теперь нам нужно указать, какие документы производят движения в нашем регистре бухгалтерии. Но для этого сначала нужно позаимствовать эти объекты из расширяемой конфигурации.

Позаимствуем в расширение документы *ОказаниеУслуги* и *ПриходнаяНакладная*. И затем на закладке *Регистраторы* в окне редактирования свойств регистра бухгалтерии *Расш1\_Управленческий* укажем, что эти документы являются регистраторами нашего регистра (рис. 3.7).



**Рис. 3.7.**Регистраторы регистра бухгалтерии «Расш1\_Управленческий»

Тем самым мы задали, что документы *ОказаниеУслуги* и *ПриходнаяНакладная*, которые при своем проведении уже совершают движения в регистре бухгалтерии в расширяемой конфигурации, помимо этого также будут производить движения и в регистре бухгалтерии управленческого учета, который мы добавили в расширение.

Исходя из этого мы должны расширить процедуры обработки проведения этих документов и добавить туда движения по регистру бухгалтерии *Расш1\_Управленческий*, которые будут содержать проводки по плану счетов *Расш1\_Детализированный*.

### Расширение модулей документов-регистраторов

Итак, откроем поочередно модули документов *ОказаниеУслуги* и *ПриходнаяНакладная* в расширяемой конфигурации, установим курсор на заголовке процедуры *ОбработкаПроведения()*, вызовем контекстное меню и выполним команду *Добавить в расширение*. В появившемся окне выбора типа вызова расширяющего метода оставим значение по умолчанию – *Вызывать после* и нажмем кнопку *ОК*.

В результате в модуле документов в расширении появится шаблон процедуры *Расш1\_ОбработкаПроведения()*, предваренный аннотацией *&После("ОбработкаПроведения")*.

#### Приходная накладная

Для документа *ПриходнаяНакладная* заполним расширяющую процедуру следующим образом (листинг 3.1).

**Листинг 3.1.** Процедура «Расш1\_ОбработкаПроведения»

&После("ОбработкаПроведения")

Процедура Расш1\_ОбработкаПроведения(Отказ, Режим)

Движения.Расш1\_Управленческий.Записывать = Истина;

Для Каждого ТекСтрокаМатериалы Из Материалы Цикл

// Регистр Управленческий

// Проводка: Д 41 (Товары) – К 60 (Расчеты с поставщиками)

Движение = Движения.Расш1\_Управленческий.Добавить();

Движение.СчетДт = ПланыСчетов.Расш1\_Детализированный.Товары;

Движение.СчетКт = ПланыСчетов.Расш1\_Детализированный.РасчетыСПоставщиками;

Движение.Период = Дата;

Движение.Сумма = ТекСтрокаМатериалы.Сумма;

Движение.КоличествоДт = ТекСтрокаМатериалы.Количество;

Движение.СубконтоДт.Номенклатура = ТекСтрокаМатериалы.Материал;

КонецЦикла;

КонецПроцедуры

Поясним текст процедуры. При проведении приходной накладной сумма поступивших материалов относится в дебет счета *Товары* (41) и в кредит счета *РасчетыСПоставщиками* (60) нашего плана счетов *Расш1\_Детализированный*. К предопределенным счетам мы обращаемся просто по имени, через точку от имени плана счетов.

По мере обхода табличной части документа *Материалы* мы добавляем движения в регистр бухгалтерии *Расш1\_Управленческий* и заполняем счет дебета (*СчетДт*), счет кредита (*СчетКт*) и сумму (*Сумма*) каждого движения.

Поскольку на счете дебета *Товары* ведется количественный учет, то мы заполняем поле движения *КоличествоДт* количеством поступивших материалов. Кроме того, мы заполняем субконто этого счета *Номенклатура* ссылкой на материал. Для обращения к субконто мы используем обращение по имени к предопределенному виду субконто через точку от коллекции субконто дебета (*СубконтоДт.Номенклатура*).

Для корректной работы процедуры в расширении нам нужно позаимствовать из расширяемой конфигурации табличную часть *Материалы* документа *ПриходнаяНакладная* вместе с ее реквизитами *Материал*, *Количество*, *Сумма*. При этом в расширение также автоматически заимствуется справочник *Номенклатура*.

Таким образом, при проведении приходной накладной сначала отрабатывает процедура *ОбработкаПроведения()* из модуля документа в расширяемой конфигурации. В результате добавляются движения в регистры накопления *ОстаткиМатериалов*, *СтоимостьМатериалов* и в регистр бухгалтерии *Финансовый*.

А после нее выполняется процедура *Расш1\_ОбработкаПроведения()* из модуля документа в расширении. В результате добавляются движения в регистр бухгалтерии *Расш1\_Управленческий*.

#### Оказание услуги

Теперь заполним расширяющую процедуру для документа *ОказаниеУслуги*. Прежде всего поясним общую схему формирования бухгалтерских проводок при продаже товаров (материалов) и оказании услуг.

При проведении документа об оказании услуг сумма общая сумма выручки относится в дебет счета *РасчетыСПокупателями* (62) и в кредит счета *Продажи* (90). А сумма себестоимости товаров (материалов) и затраты на оказание услуг относятся в дебет счета *Продажи* (90) и в кредит счета *Товары* (41) или *ОсновноеПроизводство* (20).

Таким образом, на счете *Продажи* (90) накапливается разница между выручкой и затратами предприятия. И затем в конце каждого месяца путем сопоставления оборотов по счету происходит определение финансового результата.

Напомним, что задача нашего расширения – детализировать учет на счете *Продажи*. То есть нам нужно вести учет проданных материалов и оказанных услуг отдельно друг от друга, при этом выделяя суммы затрат из общей суммы выручки.

Затратами на оказание услуг для простоты будем считать процент от суммы услуг, оказанных мастером. А себестоимость проданных (использованных при оказании услуг) материалов мы будем автоматически рассчитывать, исходя из суммарной стоимости оставшихся материалов, разделенной на их количество.

Поэтому для проводок в нашем регистре бухгалтерии управленческого учета мы будем использовать субсчета второго уровня: *ВыручкаОтМатериалов* (90.1.1) и *ВыручкаОтУслуг* (90.1.2), а также *СебестоимостьМатериалов* (90.2.1) и *ЗатратыНаУслуги* (90.2.2).

Итак, заполним расширяющую процедуру для документа *ОказаниеУслуги* следующим образом (листинг 3.2).

**Листинг 3.2.** Процедура «Расш1\_ОбработкаПроведения»

&После("ОбработкаПроведения")

Процедура Расш1\_ОбработкаПроведения(Отказ, Режим)

Движения.Расш1\_Управленческий.Записывать = Истина;

// Создать менеджер временных таблиц

МенеджерВТ = Новый МенеджерВременныхТаблиц;

Запрос = Новый Запрос;

// Укажем, какой менеджер временных таблиц использует этот запрос

Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос.Текст = "ВЫБРАТЬ

|        ОказаниеУслугиПереченьНоменклатуры.Номенклатура,

|        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры КАК ВидНоменклатуры,

|        СУММА(ОказаниеУслугиПереченьНоменклатуры.Количество) КАК КоличествоВДокументе,

|        СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК СуммаВДокументе

|ПОМЕСТИТЬ НоменклатураДокумента

|ИЗ

|        Документ.ОказаниеУслуги.ПереченьНоменклатуры КАК ОказаниеУслугиПереченьНоменклатуры

|ГДЕ

|        ОказаниеУслугиПереченьНоменклатуры.Ссылка = &Ссылка

|

|СГРУППИРОВАТЬ ПО

|        ОказаниеУслугиПереченьНоменклатуры.Номенклатура,

|        ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры";

Запрос.УстановитьПараметр("Ссылка", Ссылка);

РезультатЗапроса = Запрос.Выполнить();

Запрос2 = Новый Запрос;

Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос2.Текст = "ВЫБРАТЬ

|        НоменклатураДокумента.Номенклатура,

|        НоменклатураДокумента.ВидНоменклатуры,

|        НоменклатураДокумента.КоличествоВДокументе,

|        НоменклатураДокумента.СуммаВДокументе,

|        ЕСТЬNULL(ОстаткиМатериаловОстатки.КоличествоОстаток, 0) КАК Количество,

|        ЕСТЬNULL(СтоимостьМатериаловОстатки.СтоимостьОстаток, 0) КАК Стоимость

|ИЗ

|        НоменклатураДокумента КАК НоменклатураДокумента

|                ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СтоимостьМатериалов.Остатки( ,

|                                  Материал В ВЫБРАТЬ НоменклатураДокумента.Номенклатура

|                                          ИЗ НоменклатураДокумента)) КАК СтоимостьМатериаловОстатки

|                ПО НоменклатураДокумента.Номенклатура = СтоимостьМатериаловОстатки.Материал

|                ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиМатериалов.Остатки( ,

|                                  Материал В ВЫБРАТЬ НоменклатураДокумента.Номенклатура

|                                          ИЗ НоменклатураДокумента)) КАК ОстаткиМатериаловОстатки

|                ПО НоменклатураДокумента.Номенклатура = ОстаткиМатериаловОстатки.Материал";

РезультатЗапроса = Запрос2.Выполнить();

ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

Если ВыборкаДетальныеЗаписи.Количество = 0 Тогда

СтоимостьМатериала = 0;

Иначе

СтоимостьМатериала = ВыборкаДетальныеЗаписи.Стоимость / ВыборкаДетальныеЗаписи.Количество;

КонецЕсли;

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

// Регистр Управленческий

// Первая проводка: Д 62 (Расчеты с покупателями) – К 90.1.1 (Выручка от материалов) - Розничная сумма

Движение = Движения.Расш1\_Управленческий.Добавить();

Движение.СчетДт = ПланыСчетов.Расш1\_Детализированный.РасчетыСПокупателями;

Движение.СчетКт = ПланыСчетов.Расш1\_Детализированный.ВыручкаОтМатериалов;

Движение.Период = Дата;

Движение.Сумма = ВыборкаДетальныеЗаписи.СуммаВДокументе;

Движение.СубконтоДт.Клиенты = Клиент;

Движение.КоличествоКт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.СубконтоКт.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;

Движение.СубконтоКт.Клиенты = Клиент;

// Вторая проводка: Д 90.2.1 (Себестоимость материалов) – К 41 (Товары)

Движение = Движения.Расш1\_Управленческий.Добавить();

Движение.СчетДт = ПланыСчетов.Расш1\_Детализированный.СебестоимостьМатериалов;

Движение.СчетКт = ПланыСчетов.Расш1\_Детализированный.Товары;

Движение.Период = Дата;

Движение.Сумма = СтоимостьМатериала \* ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.КоличествоДт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.СубконтоДт.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;

Движение.КоличествоКт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.СубконтоКт.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;

Иначе

// Регистр Управленческий

// Первая проводка:      Д 62 (Расчеты с покупателями) – К 90.1.2 (Выручка от услуг) Розничная сумма

Движение = Движения.Расш1\_Управленческий.Добавить();

Движение.СчетДт = ПланыСчетов.Расш1\_Детализированный.РасчетыСПокупателями;

Движение.СчетКт = ПланыСчетов.Расш1\_Детализированный.ВыручкаОтУслуг;

Движение.Период = Дата;

Движение.Сумма = ВыборкаДетальныеЗаписи.СуммаВДокументе;

Движение.СубконтоДт.Клиенты = Клиент;

Движение.КоличествоКт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.СубконтоКт.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;

Движение.СубконтоКт.Клиенты = Клиент;

Движение.СубконтоКт.Мастера = Мастер;

// Вторая проводка: Д 90.2.2 (Затраты на услуги) – К 20 (Осн. производство)

Движение = Движения.Расш1\_Управленческий.Добавить();

Движение.СчетДт = ПланыСчетов.Расш1\_Детализированный.ЗатратыНаУслуги;

Движение.СчетКт = ПланыСчетов.Расш1\_Детализированный.ОсновноеПроизводство;

Движение.Период = Дата;

Движение.Сумма = ВыборкаДетальныеЗаписи.СуммаВДокументе \* 0.3;

Движение.КоличествоДт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.СубконтоДт.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;

Движение.СубконтоДт.Мастера = Мастер;

Движение.КоличествоКт = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

Движение.СубконтоКт.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;

КонецЕсли;

КонецЦикла;

КонецПроцедуры

Процедура обработки проведения документа до того момента, где формируются движения в регистре бухгалтерии *Расш1\_Управленческий*, взята нами из расширяемой конфигурации. Кратко опишем ее.

В целях повышения скорости проведения документа и автоматического расчета средней стоимости оставшихся материалов мы формируем запрос к табличной части документа *ПереченьНоменклатуры* и регистрам накопления *ОстаткиМатериалов* и *СтоимостьМатериалов*.

В запросе мы используем временную таблицу *НоменклатураДокумента*, в которую мы помещаем данные табличной части документа, сгруппированные по номенклатуре и виду номенклатуры.

Затем связываем ее левым соединением с таблицей *СтоимостьМатериалов.Остатки* по номенклатуре и таким же образом связываем ее с таблицей *ОстаткиМатериалов.Остатки*. И выбираем запросом поля этой временной таблицы, а также общую суммарную стоимость и общий остаток каждого материала, который содержится в табличной части документа.

В начале цикла обхода выборки результата запроса мы рассчитываем среднюю стоимость единицы текущего материала как частное от деления всей стоимости материалов, полученной запросом (*Стоимость*), на общее количество материала на всех складах (*Количество*):

*СтоимостьМатериала = ВыборкаДетальныеЗаписи.Стоимость / ВыборкаДетальныеЗаписи.Количество*.

По мере обхода выборки мы добавляем движения в регистр бухгалтерии *Расш1\_Управленческий*. Причем проводки в регистре бухгалтерии различаются в зависимости от того, является номенклатура материалом или услугой.

Если текущая номенклатура является материалом (*ВыборкаДетальныеЗаписи.Номенклатура.ВидНоменклатуры = Перечисления.ВидыНоменклатуры.Материал*), то проводки, формируемые в регистре бухгалтерии, отражают продажу клиентам материалов, требующихся при оказании услуг.

В первой проводке сумма выручки от продажи материала (*СуммаВДокументе*) относится в дебет счета *РасчетыСПокупателями* (62) и в кредит счета *ВыручкаОтМатериалов* (90.1.1).

Поскольку на счете дебета *РасчетыСПокупателями* ведется учет в разрезе клиентов, мы заполняем субконто этого счета *Клиенты* значением соответствующего реквизита документа (*Клиент*).

Поскольку на счете кредита *ВыручкаОтМатериалов* ведется количественный учет, то мы заполняем поле движения *КоличествоКт* количеством проданных материалов (*КоличествоВДокументе*). Кроме того, мы заполняем первое субконто этого счета *Номенклатура* ссылкой на материал, а второе субконто *Клиенты* – ссылкой на клиента, которому был продан этот материал.

Во второй проводке сумма себестоимости материалов относится в дебет счета *СебестоимостьМатериалов* (90.2.1) и в кредит счета *Товары* (41). Себестоимость материала рассчитывается как произведение полученной выше средней стоимости единицы материала на количество материала в документе (*СтоимостьМатериала \* ВыборкаДетальныеЗаписи.КоличествоВДокументе*).

Поскольку на счете дебета *СебестоимостьМатериалов* ведется количественный учет, то мы заполняем поле движения *КоличествоДт* количеством проданных материалов (*КоличествоВДокументе*). Кроме того, мы заполняем субконто этого счета *Номенклатура* ссылкой на материал.

Поскольку на счете кредита *Товары* ведется количественный учет, то мы заполняем поле движения *КоличествоКт* количеством проданных материалов (*КоличествоВДокументе*). Кроме того, мы заполняем субконто этого счета *Номенклатура* ссылкой на материал.

Если же текущая номенклатура не является материалом, значит, это услуга. В этом случае проводки, формируемые в регистре бухгалтерии, отражают оказание клиентам услуг.

В первой проводке сумма выручки от оказания услуги (*СуммаВДокументе*) относится в дебет счета *РасчетыСПокупателями* (62) и в кредит счета *ВыручкаОтУслуг* (90.1.2).

Поскольку на счете дебета *РасчетыСПокупателями* ведется учет в разрезе клиентов, мы заполняем субконто этого счета *Клиенты* значением соответствующего реквизита документа (*Клиент*).

Поскольку на счете кредита *ВыручкаОтУслуг* ведется количественный учет, то мы заполняем поле движения *КоличествоКт* количеством оказанных услуг (*КоличествоВДокументе*). Кроме того, мы заполняем первое субконто этого счета *Номенклатура* ссылкой на услугу, второе субконто *Клиенты* – ссылкой на клиента, которому была оказана услуга, а третье субконто *Мастера* – ссылкой на мастера, оказавшего услугу.

Во второй проводке сумма затрат на оказание услуги относится в дебет счета *ЗатратыНаУслуги* (90.2.2) и в кредит счета *ОсновноеПроизводство* (20). Затратами на оказание услуги мы договорились считать 30 % от суммы услуги, оказанной мастером (*ВыборкаДетальныеЗаписи.СуммаВДокументе \* 0.3*).

Поскольку на счете дебета *ЗатратыНаУслуги* ведется количественный учет, то мы заполняем поле движения *КоличествоДт* количеством оказанных услуг (*КоличествоВДокументе*). Кроме того, мы заполняем первое субконто этого счета *Номенклатура* ссылкой на услугу, а второе субконто *Мастера* – ссылкой на мастера, оказавшего услугу.

Поскольку на счете кредита *ОсновноеПроизводство* ведется количественный учет, то мы заполняем поле движения *КоличествоКт* количеством оказанных услуг (*КоличествоВДокументе*). Кроме того, мы заполняем субконто этого счета *Номенклатура* ссылкой на услугу.

Для корректной работы процедуры в расширении нам нужно позаимствовать из расширяемой конфигурации реквизиты *Клиент* и *Мастер* документа *ОказаниеУслуги*, а также его табличную часть *ПереченьНоменклатуры* вместе с ее реквизитами *Номенклатура*, *Количество*, *Сумма*. При этом справочники *Клиенты* и *Сотрудники* будут заимствованы в расширение автоматически.

Кроме того, нам понадобятся реквизит *ВидНоменклатуры* справочника *Номенклатура* и значение *Материал* перечисления *ВидыНоменклатуры*. А также нам нужно позаимствовать регистры накопления *ОстаткиМатериалов* (вместе с измерением *Материал* и ресурсом *Количество*) и *СтоимостьМатериалов* (вместе с измерением *Материал* и ресурсом *Стоимость*).

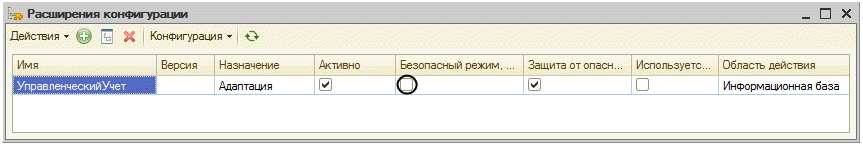
Таким образом, при проведении документа об оказании услуг сначала отрабатывает процедура *ОбработкаПроведения()* из модуля документа в расширяемой конфигурации. В результате добавляются движения в регистры накопления *ОстаткиМатериалов*, *СтоимостьМатериалов*, *Продажи* и в регистр бухгалтерии *Финансовый*.

А после нее выполняется процедура *Расш1\_ОбработкаПроведения()* из модуля документа в расширении. В результате добавляются движения в регистр бухгалтерии *Расш1\_Управленческий*.

### Формирование движений в регистре бухгалтерии «Управленческий»

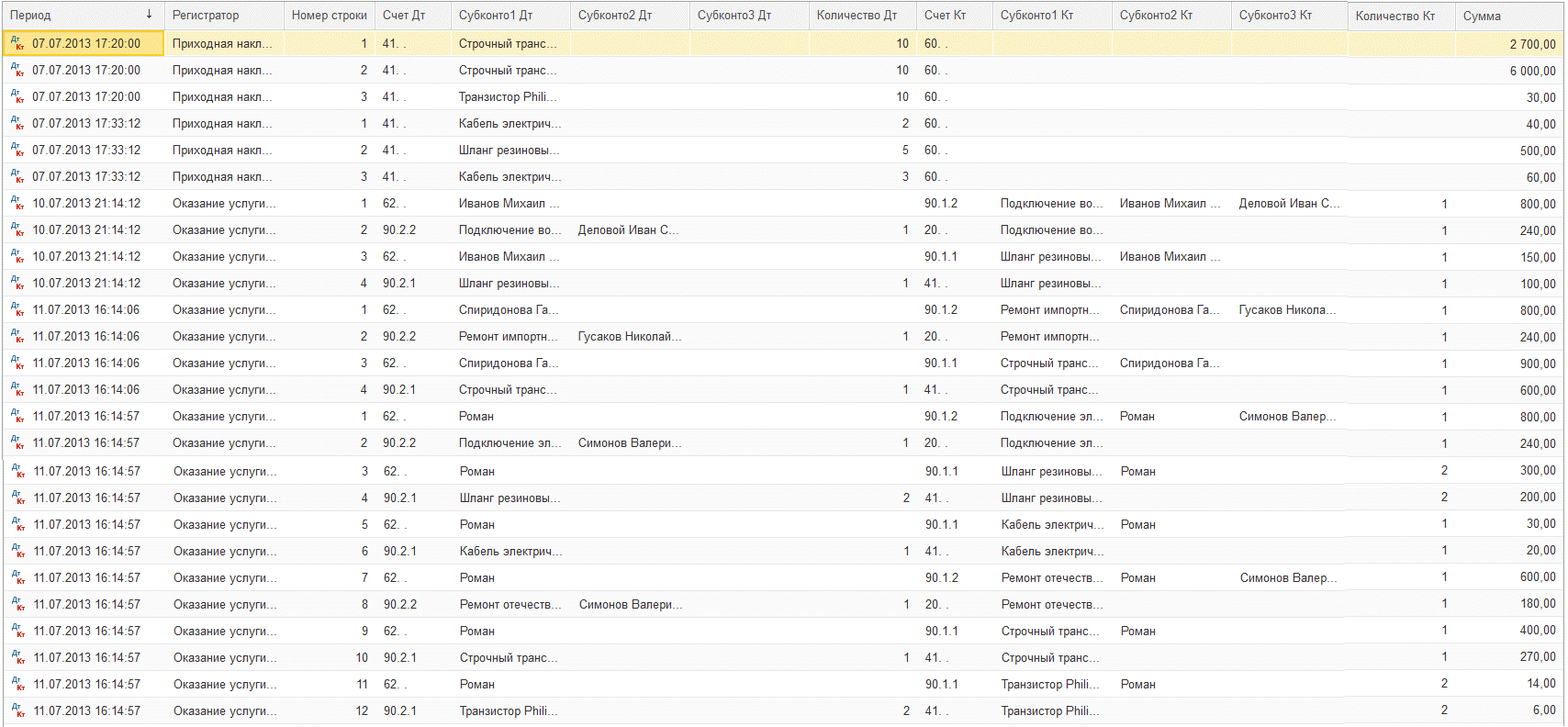
Теперь в режиме *1С:Предприятие* перепроведем все приходные накладные и документы об оказании услуг и посмотрим на движения, сформированные ими в регистре бухгалтерии управленческого учета.

Но перед тем, как запускать «1С:Предприятие», снимем флажок *Безопасный режим* у нашего расширения *УправленческийУчет* в окне списка расширений конфигурации (рис. 3.8). Дело в том, что расширение модулей, как уже говорилось во второй главе, в безопасном режиме работать не будет, хотя само расширение подключится без ошибок.



**Рис. 3.8.**Отключение безопасного режима у расширения «УправленческийУчет»

После проведения документов в регистре бухгалтерии управленческого учета будут сформированы следующие движения (рис. 3.9).



**Рис. 3.9.**Движения в регистре бухгалтерии «Расш1\_Управленческий»

Все формируется правильно. Но надо понимать, что наш пример не охватывает полный цикл хозяйственной деятельности предприятия за какой-то период, например за месяц. Это сделано для упрощения примера и потому, что наш пример и в таком виде уже хорошо демонстрирует принцип параллельного ведения управленческого учета в расширении.

В действительности, исходя из процентов от суммы услуг, выполненных мастерами, в конце месяца им будет начислена зарплата. Она будет отнесена в дебет счета *ОсновноеПроизводство* (20) и в кредит счета *РасчетыСПерсоналом* (70). При закрытии месяца все затраты, скопившиеся на счете *ОсновноеПроизводство* (20), будут списаны в дебет счета *Прибыль/Убыток* (90.9). На этот субсчет будут списаны и накопленные суммы выручки и затрат с субсчетов *Выручка* (90.1) и *СебестоимостьПродаж* (90.2). В результате сальдо на субсчете *Прибыль/Убыток* будет отражать промежуточный финансовый результат деятельности предприятия.

В нашем примере мы не учитываем также никакие налоги, расчеты с покупателями и поставщиками и т. д.

### Отчеты по управленческому учету

Теперь, чтобы увидеть и оценить результаты нашего управленческого учета, разработаем два отчета. Первый отчет *Оборотно-сальдовая ведомость* будет аналогом такого же отчета в расширяемой конфигурации, но только он будет основан на нашем детализированном плане счетов и регистре бухгалтерии, который мы добавили в расширение.

Второй отчет *Оборотно-сальдовая ведомость по счету* будет показывать обороты и сальдо по выбранному счету с детализацией по субсчетам, в разрезе каждого субконто.

#### Оборотно-сальдовая ведомость

Итак, добавим в расширение отчет *Расш1\_ОборотноСальдоваяВедомость* с синонимом «Оборотно-сальдовая ведомость по управленческому учету» и создадим схему компоновки отчета. Запрос для получения набора данных заполним следующим образом (листинг 3.3).

**Листинг 3.3.** Запрос для получения набора данных отчета

ВЫБРАТЬ

Детализированный.Ссылка КАК Счет,

УправленческийОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокДт КАК СальдоНаНачалоДт,

УправленческийОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокКт КАК СальдоНаНачалоКт,

УправленческийОстаткиИОбороты.СуммаОборотДт КАК ОборотДт,

УправленческийОстаткиИОбороты.СуммаОборотКт КАК ОборотКт,

УправленческийОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокДт КАК СальдоНаКонецДт,

УправленческийОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокКт КАК СальдоНаКонецКт

ИЗ

ПланСчетов.Расш1\_Детализированный КАК Детализированный

ЛЕВОЕ СОЕДИНЕНИЕ РегистрБухгалтерии.Расш1\_Управленческий.ОстаткиИОбороты КАК УправленческийОстаткиИОбороты

ПО (УправленческийОстаткиИОбороты.Счет = Детализированный.Ссылка)

Данные для нашего отчета мы будем получать из двух таблиц: *ПланСчетов.Расш1\_Детализированный* и *РегистрБухгалтерии.Расш1\_Управленческий.ОстаткиИОбороты*, связанных левым соединением. Причем слева должна находиться таблица плана счетов, чтобы те счета, по которым не было оборотов за отчетный период, тоже были отражены в отчете.

В списке полей выборки из виртуальной таблицы остатков и оборотов регистра бухгалтерии мы получаем развернутые остатки по дебету и кредиту счетов на начало и конец отчетного периода, а также обороты по дебету и кредиту счетов.

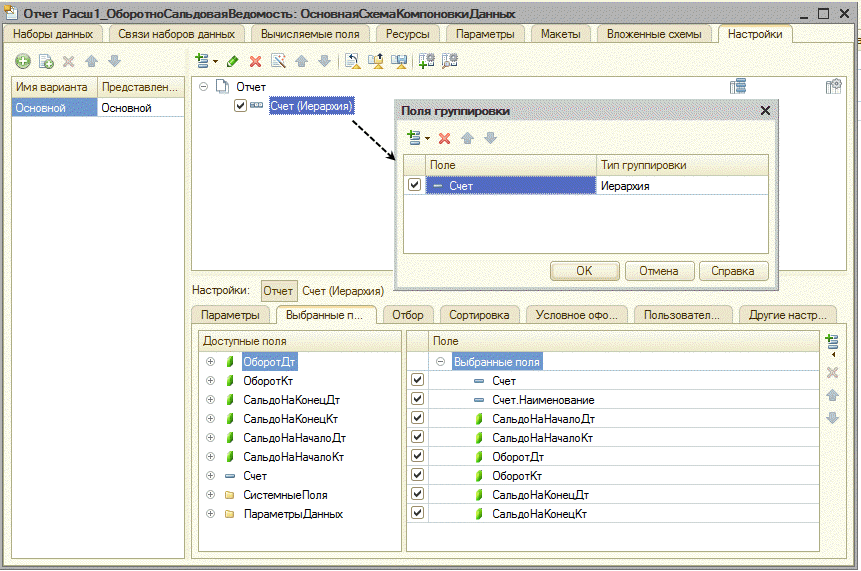
На закладке *Ресурсы* схемы компоновки данных добавим все возможные поля набора данных в ресурсы отчета.

На закладке *Параметры* мы видим автоматически добавленные в схему компоновки данных параметры *НачалоПериода* и *КонецПериода*, т. к. бухгалтерские отчеты всегда формируются за определенный период. Для параметра *КонецПериода* в колонке *Выражение* зададим выражение для его расчета. Это нужно для того, чтобы все данные за последний день тоже попали в отчет (листинг 3.4).

**Листинг 3.4.** Выражение для расчета параметра «КонецПериода»

КонецПериода(&КонецПериода, "День")

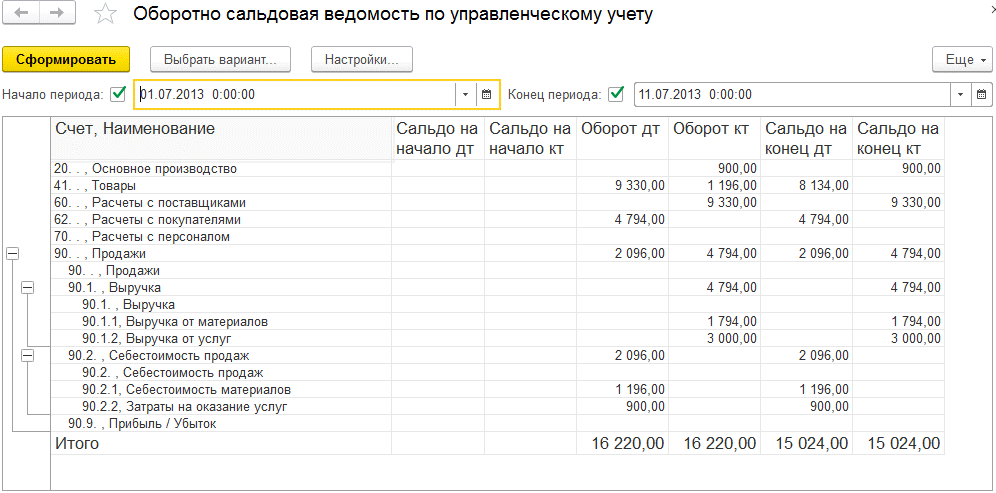
На закладке *Настройки* добавим в структуру отчета группировку по полю *Счет* с типом группировки *Иерархия*. Это позволит нам видеть в отчете всю структуру подчиненности плана счетов, включая субсчета с подитогами по ним. Выведем в отчет поля: *Счет*, *Счет.Наименование*, *СальдоНачалоДт*, *СальдоНачалоКт*, *ОборотДт*, *ОборотКт*, *СальдоНаКонецДт*, *СальдоНаКонецКт* (рис. 3.10).



**Рис. 3.10. Структура отчета**«Расш1\_ОборотноСальдоваяВедомость»

На закладке *Другие настройки* параметр *Выводить параметры* установим в значение *Не выводить*. На закладке *Параметры* включим параметры *Начало периода* и *Конец периода* в состав быстрых пользовательских настроек отчета.

В результате оборотно-сальдовая ведомость по управленческому учету, вызванная из раздела *Управленческий учет – Отчеты*, будет иметь следующий вид (рис. 3.11).



**Рис. 3.11.**Отчет «Оборотно-сальдовая ведомость по управленческому учету»

Для сравнения можно открыть оборотно-сальдовую ведомость (по бухгалтерскому учету), которая существует в расширяемой конфигурации, из раздела *Бухгалтерия – Отчеты*. Этот отчет в укрупненном виде аналогичен нашему, но в нем нет сумм по услугам, так как учет услуг в расширяемой конфигурации не производился.

#### Оборотно-сальдовая ведомость по счету

Теперь добавим в расширение отчет *Расш1\_ОборотноСальдоваяВедомостьПоСчету* с синонимом «Оборотно-сальдовая ведомость по счету» и создадим схему компоновки отчета. Запрос для получения набора данных заполним следующим образом (листинг 3.5).

**Листинг 3.5.** Запрос для получения набора данных отчета

ВЫБРАТЬ

УправленческийОстаткиИОбороты.Счет КАК Счет,

УправленческийОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокДт КАК СальдоНаНачалоДт,

УправленческийОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокКт КАК СальдоНаНачалоКт,

УправленческийОстаткиИОбороты.СуммаОборотДт КАК ОборотДт,

УправленческийОстаткиИОбороты.СуммаОборотКт КАК ОборотКт,

УправленческийОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокДт КАК СальдоНаКонецДт,

УправленческийОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокКт КАК СальдоНаКонецКт,

УправленческийОстаткиИОбороты.Субконто1 КАК Субконто1,

УправленческийОстаткиИОбороты.Субконто2 КАК Субконто2,

УправленческийОстаткиИОбороты.Субконто3 КАК Субконто3,

ТИПЗНАЧЕНИЯ(УправленческийОстаткиИОбороты.Субконто1) КАК ВидСубконто1,

ТИПЗНАЧЕНИЯ(УправленческийОстаткиИОбороты.Субконто2) КАК ВидСубконто2,

ТИПЗНАЧЕНИЯ(УправленческийОстаткиИОбороты.Субконто3) КАК ВидСубконто3

ИЗ

РегистрБухгалтерии.Расш1\_Управленческий.ОстаткиИОбороты( , , , , Счет В ИЕРАРХИИ (&Счет), , ) КАК УправленческийОстаткиИОбороты

Данные для нашего отчета мы будем получать из таблицы *РегистрБухгалтерии.Расш1\_Управленческий.ОстаткиИОбороты*. Причем в параметрах этой виртуальной таблицы задан отбор по счету с иерархией (*Счет В ИЕРАРХИИ (&Счет)*). То есть в отчете будут показаны все движения регистра по выбранному счету, включая все его субсчета.

Чтобы показывать данные в разрезе субконто, из виртуальной таблицы остатков и оборотов регистра бухгалтерии мы получаем поля *Субконто1*, *Субконто2*, *Субконто3*, а также тип значения этих полей. В остальном запрос для получения набора данных нашего отчета аналогичен предыдущему отчету.

На закладке *Ресурсы* схемы компоновки данных добавим все возможные поля набора данных в ресурсы отчета.

На закладке *Параметры* мы видим автоматически добавленные в схему компоновки данных параметры *НачалоПериода* и *КонецПериода*, т. к. бухгалтерские отчеты всегда формируются за определенный период. Для параметра *КонецПериода* в колонке *Выражение* зададим выражение для его расчета, чтобы все данные за последний день тоже попали в отчет (листинг 3.6).

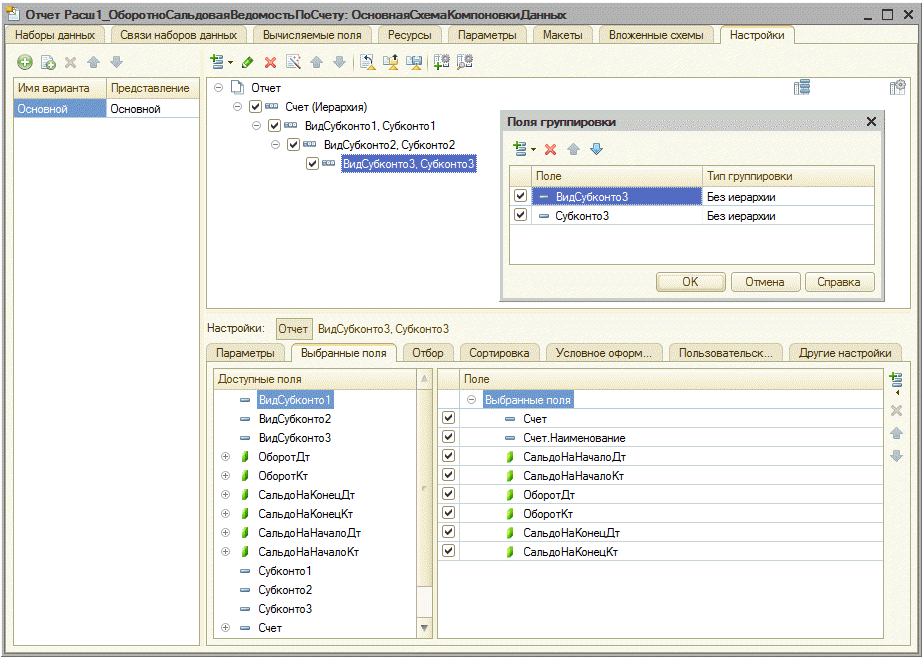
**Листинг 3.6.** Выражение для расчета параметра «КонецПериода»

КонецПериода(&КонецПериода, "День")

У параметра *Счет* установим свойство *Значение* в *ПланСчетов.Расш1\_Детализированный.ПустаяСсылка* и свойство *Использование* в значение *Всегда*.

На закладке *Настройки* создадим структуру отчета. Сначала добавим группировку по полю *Счет* с типом группировки *Иерархия*. Затем вложим в эту группировку составную группировку по полям *ВидСубконто1* и *Субконто1*. Затем вложим в эту группировку составную группировку по полям *ВидСубконто2* и *Субконто2*. Затем вложим в эту группировку составную группировку по полям *ВидСубконто3* и *Субконто3* (см. рис. 3.12).

Выведем в отчет поля: *Счет*, *Счет.Наименование*, *СальдоНачалоДт*, *СальдоНачалоКт*, *ОборотДт*, *ОборотКт*, *СальдоНаКонецДт*, *СальдоНаКонецКт* (рис. 3.12).

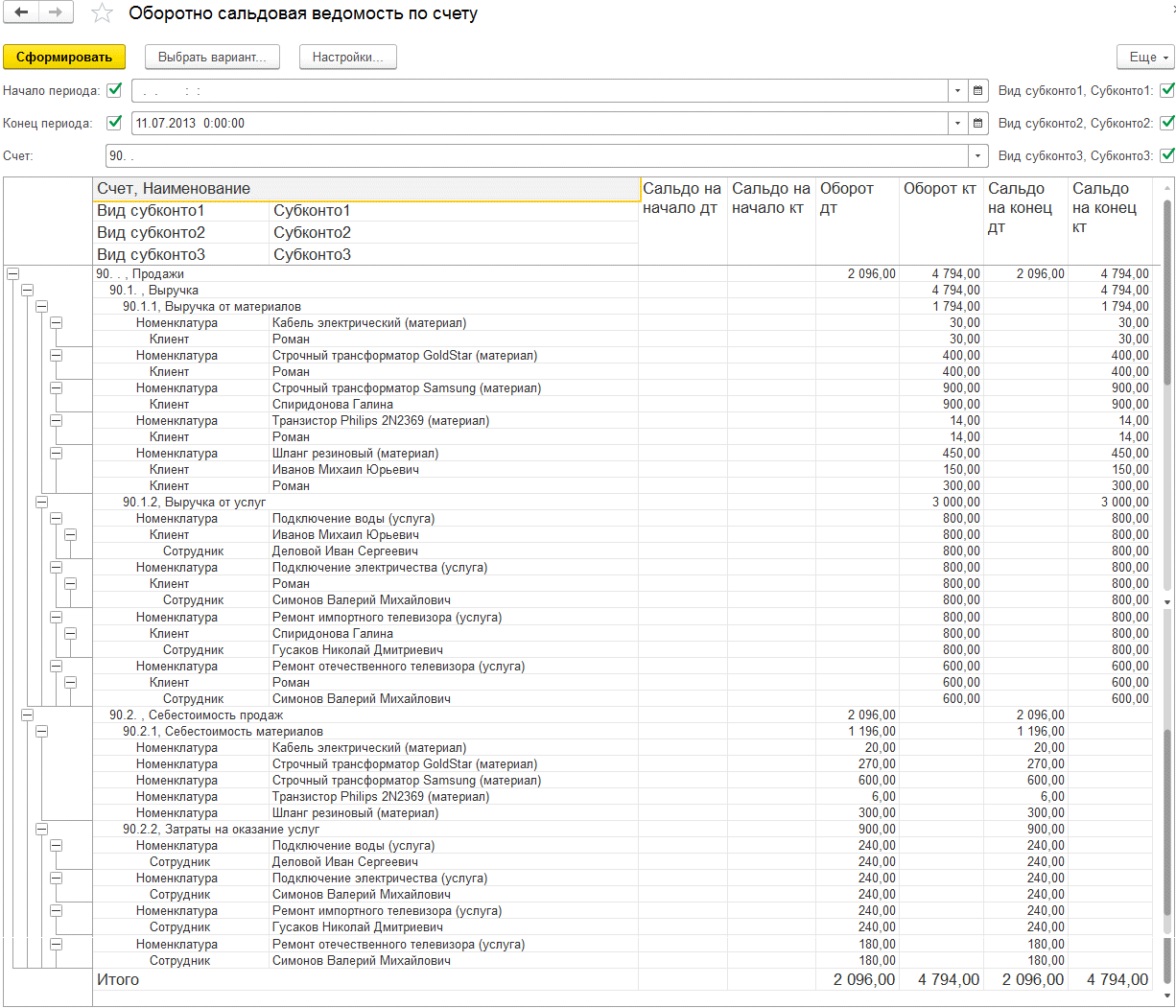


**Рис. 3.12.**Структура отчета «Расш1\_ОборотноСальдоваяВедомостьПоСчету»

На закладке *Другие настройки* параметр *Выводить параметры* установим в значение *Не выводить*. На закладке *Параметры* включим параметры *Начало периода*, *Конец периода* и *Счет* в состав быстрых пользовательских настроек отчета.

Кроме того, выделив каждую из трех составных группировок, нажмем кнопку *Свойства элемента пользовательских настроек* и включим группировку в состав быстрых пользовательских настроек отчета. Это позволит пользователю непосредственно перед формированием отчета включать или выключать показ той или иной группировки, тем самым укрупняя или детализируя данные в отчете.

В результате оборотно-сальдовая ведомость по счету, вызванная из раздела *Управленческий учет – Отчеты*, будет иметь следующий вид (рис. 3.13).



**Рис. 3.13.**Отчет «Оборотно-сальдовая ведомость по счету»

Таким образом, данные оборотно-сальдовой ведомости по выбранному счету детализированы по субсчетам и представлены в разрезе субконто. Показ группировок по субконто можно по желанию включить или выключить, тем самым укрупняя или детализируя данные в отчете.

С помощью подобного отчета, выбрав нужный счет, руководящий персонал, например, может подвергнуть всестороннему анализу движения на этом счете.

# Глава 4. Расширение для тарификации сотрудников государственных медицинских учреждений

В начале книги говорилось, что разработка расширений для конкретных заказчиков (или их групп) необходима в тех случаях, когда функциональность типового решения нужна многим пользователям, но ее необходимо немного доработать. Это вызвано либо отличием регламентирующих инструкций и отчетности в конкретном регионе заказчика, либо потребностями и пожеланиями конкретного заказчика. Для тарификации в медицинских учреждениях это особенно актуально.

В этой главе мы разработаем реально работающее расширение для тарификации сотрудников государственных медицинских учреждений Московской области. Это расширение будет учитывать специфику конкретного внедрения, оно будет ориентировано на конкретного заказчика, на его пожелания и особенности его работы.

То есть наше расширение не будет универсальным (у нас и не было такой задачи), но при желании его можно легко доработать под требования других похожих внедрений. Об этом мы поговорим в разделе [«Доработка расширения для похожих внедрений»](#_Доработка_расширения_для).

## Постановка задачи

Само понятие *тарификация* довольно специфичное и понимается в каждой организации по-своему. При разработке нашего расширения под тарификацией мы будем понимать процесс установки должностных окладов сотрудников медицинского учреждения, а также надбавок к ним, на основе которых по определенному алгоритму определяется месячный фонд оплаты труда сотрудника (*ФОТ*). Конечным результатом тарификации будем считать получение тарификационных списков сотрудников медицинского учреждения, представленных в четырех регламентированных формах.

Следует иметь в виду, что в каждом регионе существуют собственные постановления и инструкции, регулирующие тарификацию сотрудников медицинского учреждения. При разработке нашего расширения мы будем руководствоваться последней редакцией приказа Минздрава МО № 242 от 24.08.2007 «Об исполнении постановления Правительства Московской области от 03.07.2007 № 483/23 "Об оплате труда работников государственных учреждений здравоохранения Московской области" (с изменениями на 28 июля 2016 года)».

То есть наше расширение будет подходить для всех государственных медицинских учреждений Московской области, за исключением каких-то нерегламентированных отчетов и настроек, которые нужны для конкретной организации. Как уже говорилось, в этом случае можно просто немного доработать наше законченное расширение.

### Базовые понятия тарификации и алгоритм расчета ФОТ

Прежде чем начинать разработку расширения, необходимо познакомиться с некоторыми базовыми понятиями тарификации и алгоритмом расчета фонда оплаты труда (*ФОТ*).

Тарификационные списки сотрудников лечебно-профилактического учреждения (*ЛПУ*) группируются по категориям персонала (врачи, средний медицинский персонал, младший медицинский персонал и т. д.). Фонды оплаты труда получаются в разбивке по видам финансирования.

Рассмотрим подробнее, как формируется ФОТ по сотрудникам и по вакантным ставкам.

Должностной оклад сотрудника устанавливается исходя из его профессионально-квалификационного уровня – *ПКУ* (например, «врач-кардиолог»), который имеет базовый тариф оплаты.

Затем, если сотрудник занимает руководящую должность (кроме руководителей учреждения и рабочих), ему устанавливается надбавка *за руководство*.

Если у сотрудника есть ученая степень, ему устанавливается надбавка *за ученую степень* (только для медицинских работников).

Если у сотрудника есть почетное звание, ему устанавливается надбавка *за почетное звание* (кроме рабочих).

Все эти надбавки устанавливаются в процентном отношении к должностному окладу, установленному согласно ПКУ. Размер надбавок округляется до целого.

Затем должностной оклад и описанные выше надбавки суммируются. Получается итоговый должностной оклад, который умножается на занимаемую по штатному расписанию ставку (*объем работ* – 1,0; 0,75; 0,5 и т. п.). Получается так называемый «месячный должностной оклад».

И уже к нему применяется надбавка за вредные или опасные условия труда (*за вредность*) и надбавка за продолжительность непрерывной работы в ЛПУ (*за непрерывный стаж*). Надбавки устанавливаются в процентном отношении к месячному должностному окладу, размер этих надбавок не округляется.

Отдельно нужно сказать о надбавке за непрерывный стаж. Принято различать два вида стажа, которые влияют на размер надбавки за стаж: непрерывный стаж в ЛПУ и непрерывный участковый стаж в ЛПУ. Участковый стаж исчисляется у сотрудников, которые работают в должностях участковых врачей и медицинских сестер и работают по вызовам населения. У остальных сотрудников, работающих в медицинском учреждении, исчисляется «обычный» непрерывный стаж. Участковый стаж имеет отдельную, более высокую (чем обычный стаж) шкалу надбавок к должностному окладу, поэтому учитывается отдельно.

В результате месячный фонд оплаты труда сотрудника (ФОТ) образуется из суммы его месячного должностного оклада и надбавок за вредность и за непрерывный стаж.

Оклад по вакантным должностям рассчитывается исходя из среднего должностного оклада по позициям штатного расписания. Для этого используется так называемая «вилка окладов» в штатном расписании. Максимальный и минимальный оклад по вакантной позиции штатного расписания усредняется и умножается на количество вакантных ставок.

Так образуется месячный должностной оклад по строке вакансий, который затем увеличивается на 15-процентную надбавку за непрерывный стаж и на надбавку за вредность, если она указана для позиции штатного расписания. В результате получается вакантный ФОТ по вакантным должностям.

В процессе разработки расширения мы приведем более формализованный вариант расчета ФОТ.

### Порядок наших действий

Для упрощения примера в качестве расширяемой конфигурации мы возьмем простую конфигурацию, в которой ведется кадровый учет сотрудников ЛПУ. То есть в этом прикладном решении ведутся справочники отделений, должностей, ПКУ, видов стажа и т. п.; ведется штатное расписание и список сотрудников, принятых на работу по позициям штатного расписания; для каждого сотрудника учитываются его категории, ученые степени, почетные звания и стажи, размер которых пересчитывается на конкретную дату.

Надо понимать, что это не реально работающее кадровое решение, а лишь урезанный и упрощенный пример конфигурации, к которой можно применить разработанное нами расширение для тарификации. Более подробно расширяемая конфигурация описывается в разделе [«Расширяемая конфигурация для ведения кадрового учета»](#_Расширяемая_конфигурация_для).

Наше расширение добавит к кадровым данным все данные, необходимые для тарификации сотрудников. Например, шкалу надбавок за стаж для видов стажа, виды финансирования отделений, группы учета должностей в тарификации и т. п. На основе этих данных расширение сможет рассчитать ФОТ сотрудников и построить необходимые отчеты по тарификации.

Поскольку бюджетное ЛПУ – довольно крупная организация, наше расширение будет выполнять тарификацию сотрудников по отделениям.

Для расчета ФОТ сотрудников мы разработаем в расширении собственный документ, с помощью которого пользователь, выполняющий тарификацию (например, экономист), может вводить и редактировать показатели тарификации сотрудников в удобном и привычном для него табличном виде.

Для отражения всех данных по тарификации сотрудников и расчета фонда оплаты труда мы разработаем отчет, в котором будут представлены тарификационные списки сотрудников по отделениям. Его можно будет вызывать как отдельно, так и непосредственно из формы документа, в которой вводятся показатели тарификации.

В заключение мы разработаем регламентированный отчет, представляющий тарификационные списки сотрудников медицинского учреждения в разбивке по четырем формам тарификационной отчетности. Эти списки в качестве результата тарификации сотрудников учреждения подаются в вышестоящие организации.

Кроме того, мы реализуем сводный вариант отчета для внутреннего использования, в котором ставки штатного расписания и фонды оплаты труда (в том числе фактические и вакантные) разбиты по статьям финансирования.

Файлы выгрузки расширяемой конфигурации «Кадры.dt» и применяемого к ней расширения «Тарификция.cfe» находятся на прилагаемом к книге компакт-диске.

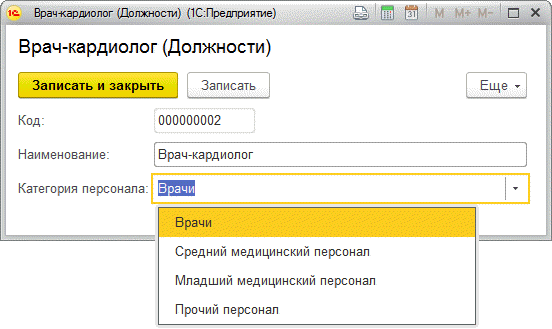
## Расширяемая конфигурация для ведения кадрового учета

Прежде чем разрабатывать наше расширение, необходимо понять, чего нам не хватает для выполнения тарификации, какие объекты надо заимствовать из расширяемой конфигурации и доработать в расширении, а какие – добавить. Поэтому сначала кратко опишем состав и функциональность нашей расширяемой конфигурации с именем *Кадры*.

В конфигурации существуют так называемые «простые» справочники (без специальных реквизитов и табличных частей, со стандартной функциональностью, использующие автоматически генерируемые формы):

* *Виды стажа*,
* *Ученые степени*,
* *Почетные звания*,
* *Категории*,
* *Специальности*,
* *Отделения*.

В справочнике *Должности* содержится список должностей сотрудников с указанием категории персонала, к которой должность относится. Категория персонала хранится в реквизите справочника *КатегорияПерсонала*, ссылающемся на перечисление *КатегорииПерсонала* со значениями: *Врачи*, *Средний медицинский персонал*, *Младший медицинский персонал*, *Прочий персонал*(рис. 4.1).

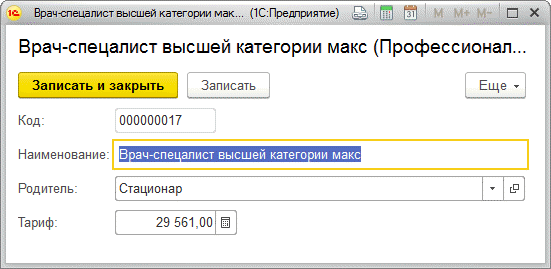


**Рис. 4.1.**Должность с указанием категории персонала

**ПРИМЕЧАНИЕ**

*Категория персонала будет использоваться в дальнейшем в отчетах по тарификации, но она нужна и для чисто кадровых отчетов и отчетов по медицинской статистике.*

В справочнике *ПКУ* содержится список профессионально-квалификационных уровней сотрудников с указанием базового тарифа оплаты каждого уровня. Тариф оплаты хранится в реквизите справочника *Тариф* (рис. 4.2).



**Рис. 4.2.**ПКУ с указанием тарифа оплаты

**ПРИМЕЧАНИЕ**

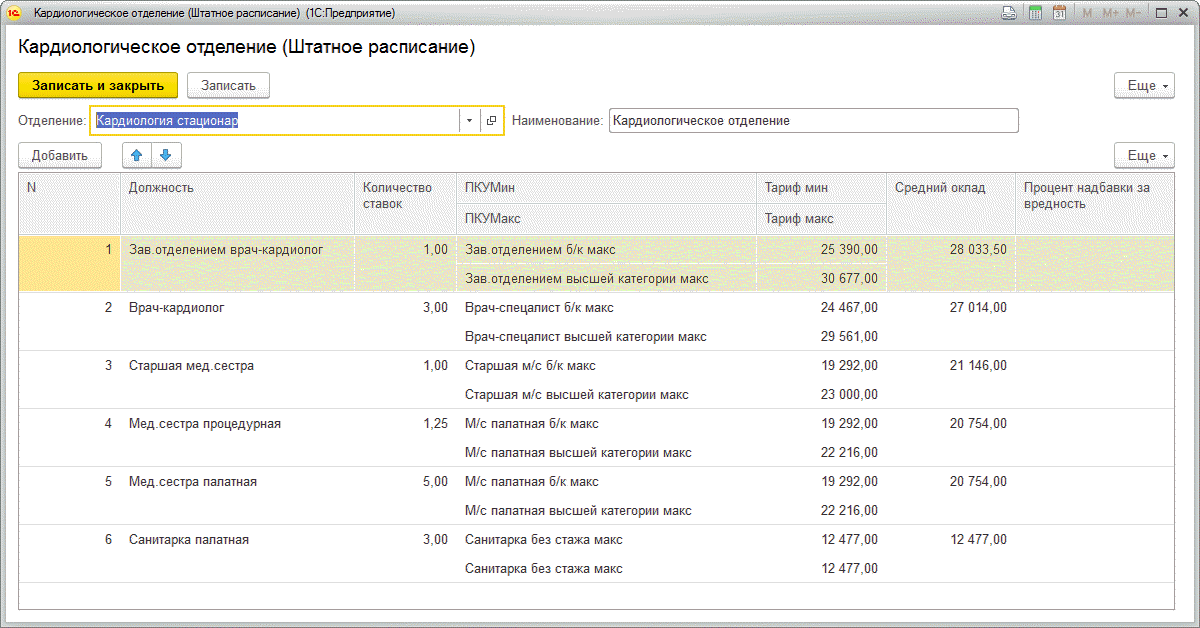
*Базовый тариф оплаты по ПКУ будет использоваться в дальнейшем в отчетах по тарификации, но прежде всего он нужен, чтобы определить должностной оклад, с которым сотрудник принимается на работу.*

В справочнике *ШтатноеРасписание* хранится штатное расписание ЛПУ по отделениям. Рассмотрим его состав подробнее, так как поля справочника будут затем использоваться в расширении, в отчетах по тарификации.

Справочник *ШтатноеРасписание* содержит реквизит *Отделение* (тип *СправочникСсылка.Отделения*) и табличную часть *Штаты* с реквизитами:

* *Должность* (тип *СправочникСсылка.Должности*),
* *КоличествоСтавок* (тип *Число* 5, 2),
* *ПКУМин* (тип *СправочникСсылка.ПКУ*),
* *ПКУМакс* (тип *СправочникСсылка.ПКУ*),
* *ТарифМин* (тип *Число* 10, 2),
* *ТарифМакс* (тип *Число* 10, 2),
* *СреднийОклад* (тип *Число* 10, 2),
* *ПроцентНадбавкиЗаВредность* (тип *Число* 2).

У справочника существует основная форма элемента, в которой для каждой должности указываются минимальный и максимальный ПКУ, определяющие соответственно минимальный и максимальный тариф оплаты, так называемую «вилку окладов» по позиции штатного расписания (рис. 4.3):



**Рис. 4.3.**Штатное расписание кардиологического отделения

При изменении минимального или максимального ПКУ по должности средний оклад пересчитывается как среднее арифметическое из соответствующих этим уровням тарифов. Таким образом, в реквизите табличной части *СреднийОклад* хранится усредненный оклад по должности, который будет затем использоваться для расчета ФОТ по строке вакансий.

Кроме того, для каждой должности (если для нее предусмотрена надбавка за вредность) указывается процент надбавки за вредность, который хранится в реквизите табличной части *ПроцентНадбавкиЗаВредность*. Этот показатель также будет использоваться для тарификации сотрудников.

Ну, и, наконец, в справочнике *Сотрудники* хранится список сотрудников ЛПУ. Рассмотрим его состав подробнее, так как поля справочника будут затем использоваться в расширении – в документах тарификации сотрудников и в отчетах по тарификации.

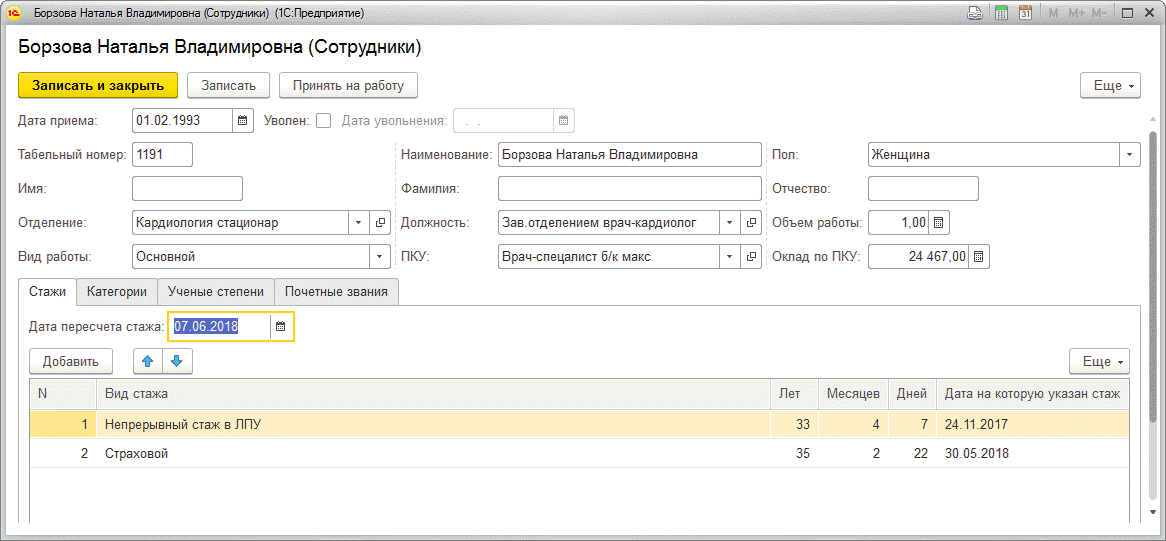
Справочник *Сотрудники* содержит реквизиты:

* *Имя* (тип *Строка* 10),
* *Отчество* (тип *Строка* 10),
* *Фамилия* (тип *Строка* 15),
* *Пол*(тип *ПеречислениеСсылка.Пол* со значениями: *Мужчина*, *Женщина*),
* *ТабельныйНомер*(тип *Строка* 5),
* *ДатаПриема*(тип *Дата*),
* *Отделение* (тип *СправочникСсылка.Отделения*),
* *Должность* (тип *СправочникСсылка.Должности*),
* *ОбъемРаботы*(тип *Число* 4, 2),
* *ВидРаботы*(тип *ПеречислениеСсылка.ВидыРаботы*со значениями: *Основной*, *Внутреннее совместительство*, *Внешнее совместительство*),
* *ПКУ* (тип *СправочникСсылка.ПКУ*),
* *ОкладПоПКУ*(тип *Число* 10, 2),
* *ДатаПересчетаСтажа*(тип *Дата*),
* *ДатаУвольнения*(тип *Дата*),
* *Уволен* (тип *Булево*).

Справочник *Сотрудники* содержит табличные части:

* *Стажи* с реквизитами:
  + *ВидСтажа* (тип *СправочникСсылка.ВидыСтажа*),
  + *Лет*(тип *Число* 2),
  + *Месяцев*(тип *Число* 2),
  + *Дней*(тип *Число* 2),
  + *ДатаНаКоторуюУказанСтаж*(тип *Дата*).
* *Категории*с реквизитами:
  + *Категория*(тип *СправочникСсылка.Категории*),
  + *Специальность*(тип *СправочникСсылка.Специальности*),
  + *ДатаПрисвоения*(тип *Дата*),
  + *СрокДействия*(тип *Дата*).
* *УченыеСтепени*с реквизитами:
  + *УченаяСтепень*(тип *СправочникСсылка.УченыеСтепени*),
  + *СрокДействия*(тип *Дата*).
* *ПочетныеЗвания*с реквизитами:
  + *ПочетноеЗвание* (тип *СправочникСсылка.ПочетныеЗвания*),
  + *СрокДействия*(тип *Дата*).

У справочника существует основная форма элемента, в которой для каждого сотрудника указываются дата его приема на работу, отделение, должность, на которую он принимается, объем работы (количество занимаемых сотрудником ставок) и вид работы (основной, внешнее совместительство и т. п.). При изменении отделения или должности сотруднику подставляется максимальный ПКУ и максимальный тариф оплаты, соответствующий этой должности в штатном расписании отделения (рис. 4.4).



**Рис. 4.4.** Форма сотрудника

Если требуется, ПКУ сотрудника можно изменить вручную, и оклад, соответствующий ПКУ, подставится автоматически.

Если сотрудник увольняется, ему включается признак увольнения и вносится дата увольнения.

На закладках *Стажи*, *Категории*, *Ученые степени*, *Почетные звания* вводятся данные, хранящиеся в соответствующих табличных частях справочника. При изменении даты, на которую указан стаж, размер всех видов стажей сотрудника изменяется на разницу между датой первоначального ввода и датой последнего пересчета стажей сотрудника.

## Расширение для выполнения тарификации

Итак, мы хотим, чтобы с помощью нашего прикладного решения выполнялся не только кадровый учет, но и тарификация сотрудников. Для этого создадим у конфигурации *Кадры* расширение и реализуем в нем все недостающую для тарификации функциональность.

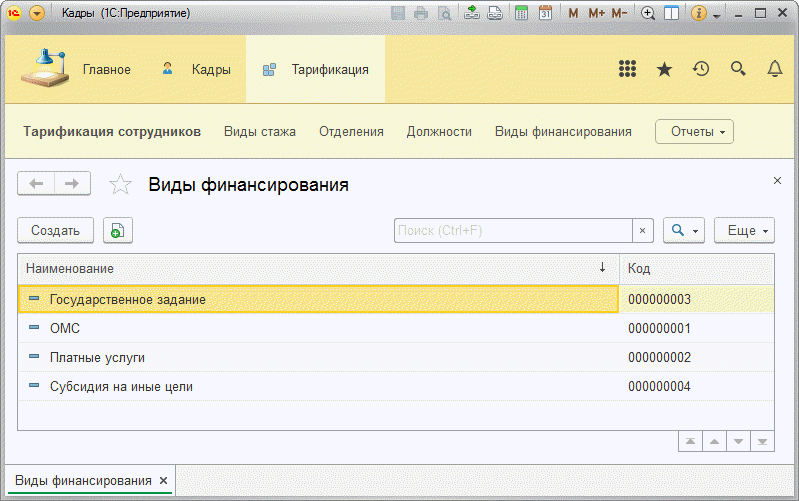
Укажем имя расширения – *Тарификация*, остальные свойства оставим по умолчанию.

Первое, что мы сделаем в расширении, – это добавим подсистему с именем *Расш1\_Тарификация* и синонимом *Тарификация*, в которой будем размещать все заимствованные и собственные объекты, специфичные для тарификации. Заметьте, что перед именем подсистемы автоматически добавился префикс *Расш1\_*, который был стандартно установлен в свойствах нашего расширения. Установим у подсистемы какую-нибудь узнаваемую картинку, например картинку *Константа* из стандартной библиотеки картинок.

### Доработка исходной структуры кадровых данных

В этом разделе мы рассмотрим, как нужно доработать расширяемую кадровую конфигурацию, чтобы использовать ее для тарификации. Прежде чем разрабатывать собственно документ, выполняющий тарификацию сотрудников, мы должны доработать в расширении структуры кадровых данных (в нашем случае, справочников) так, чтобы в них можно было вводить и хранить информацию, которая будет использоваться для тарификации.

Как говорилось в разделе [«Базовые понятия тарификации и алгоритм расчета ФОТ»](#_Базовые_понятия_тарификации), фонды оплаты труда (ФОТ) получаются в разбивке по видам финансирования. Поэтому прежде всего нам нужно добавить в расширение собственный справочник, в котором будут храниться виды финансирования (рис. 4.5).

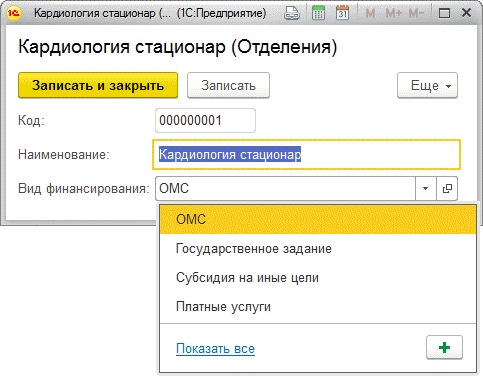


**Рис. 4.5.**Список видов финансирования в разделе «Тарификация»

Добавим в расширение «простой» справочник с именем *Расш1\_ВидыФинансирования* и синонимом «Виды финансирования». Укажем, что справочник будет виден в подсистеме *Расш1\_Тарификация*.

Теперь для каждого отделения ЛПУ нам нужно иметь возможность указать, из какого источника оно финансируется. Справочник *Отделения* уже существует в расширяемой конфигурации. Поэтому нам нужно заимствовать его в расширение и добавить в него реквизит соответствующего типа.

Заимствуем справочник *Отделения* и добавим ему реквизит *Расш1\_ВидФинансирования*, ссылающийся на ранее добавленный в расширение справочник (рис. 4.6).



**Рис. 4.6.**Вид финансирования отделения

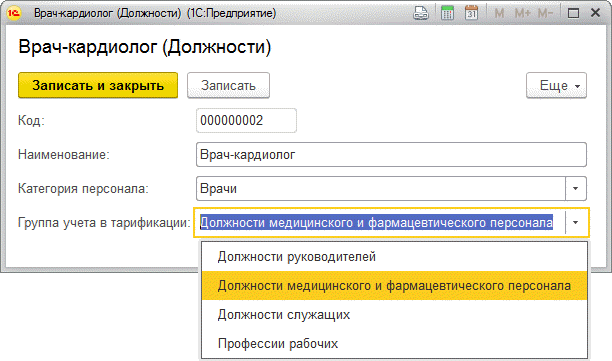
Затем для каждой должности нужно указать тарификационную группу, к которой она относится. Например: должности медицинского и фармацевтического персонала, должности руководителей, должности служащих и т. п.

Это необходимо для того, чтобы при формировании отчетов – тарификационных списков сотрудников ФМБА разбить всех сотрудников на тарификационные группы и отражать их в соответствующих регламентируемых формах.

Поэтому сначала добавим в расширение собственное перечисление *Расш1\_ГруппыУчетаВТарификации* с синонимом «Группы учета в тарификации», содержащее значения:

* *ДолжностиРуководителей*,
* *ДолжностиМедицинскогоИФармацевтическогоПерсонала*,
* *ДолжностиСлужащих*,
* *ПрофессииРабочих*.

Затем заимствуем из расширяемой конфигурации справочник *Должности* и добавим ему реквизит *Расш1\_ГруппаУчетаВТарификации*, ссылающийся на ранее добавленное в расширение перечисление (рис. 4.7).



**Рис. 4.7.**Группа по тарификации для должности

Далее нам потребуется доработать в расширении справочник *Виды стажа*, который существовал в расширяемой конфигурации в виде простого списка. В справочнике содержались виды стажа, которые использовались в кадровом учете сотрудников: медицинский, страховой, непрерывный и т. п.

Из этого списка нам нужно выделить виды стажа, размер которых определяет надбавку за стаж к должностному окладу сотрудника. Для этого у каждого вида стажа потребуется включить соответствующий признак, указывающий, используется ли он в тарификации.

Позаимствуем справочник *ВидыСтажа* в расширение и добавим ему реквизит *Расш1\_ИспользуетсяВТарификации* булевого типа.

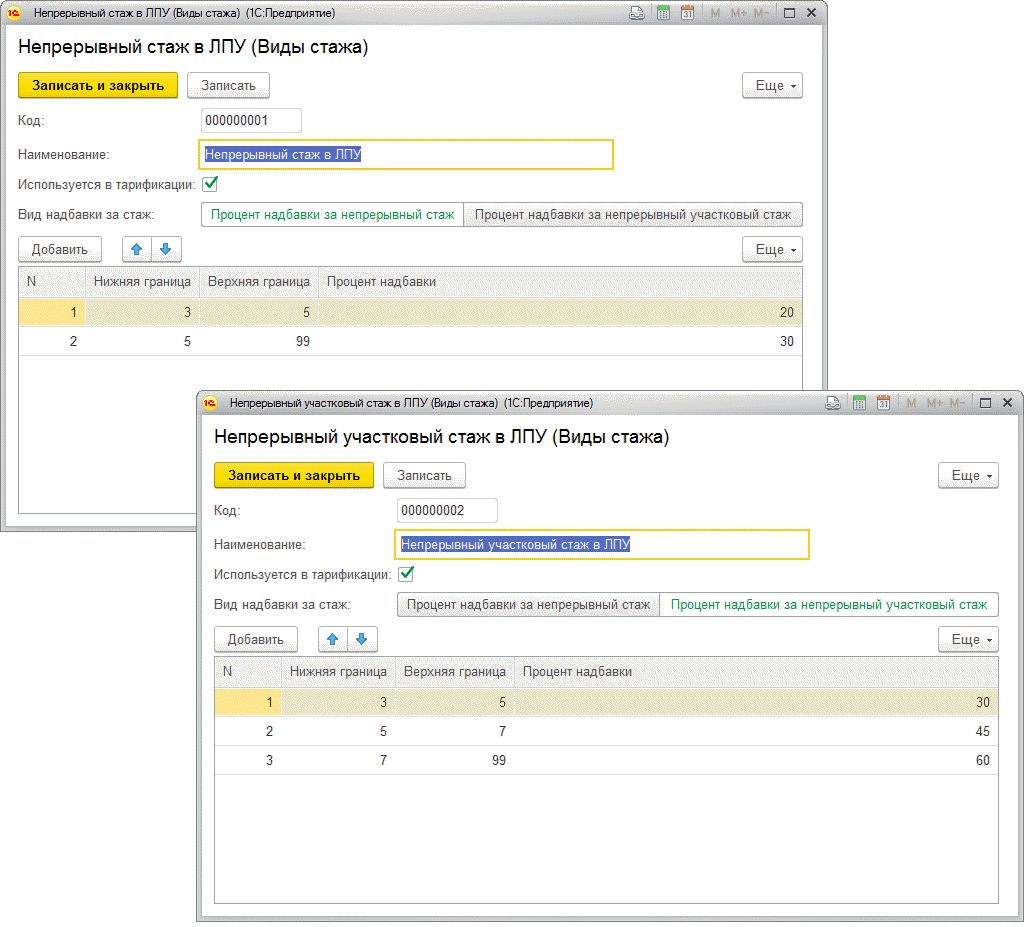
Как говорилось в разделе [«Базовые понятия тарификации и алгоритм расчета ФОТ»](#_Базовые_понятия_тарификации), на размер надбавки за стаж влияют два вида стажа: непрерывный стаж в ЛПУ и непрерывный участковый стаж в ЛПУ. У каждого из этих видов стажа существует своя шкала надбавок к должностному окладу. Например, если непрерывный стаж сотрудника от 3 до 5 лет, то ему назначается 20-процентная надбавка к окладу. А если сотрудник имеет такой же участковый непрерывный стаж, то ему полагается уже 30-процентная надбавка.

Во встроенном языке нам потребуется различать эти два вида стажа. Чтобы не «закладываться» на код или наименование стажа, добавим справочнику еще один реквизит *Расш1\_ВидНадбавкиЗаСтаж*, который будет ссылаться на еще одно собственное перечисление *Расш1\_ВидыНадбавокЗаСтаж* со значениями:

* *ПроцентНадбавкиЗаНепрерывныйСтаж*,
* *ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж*.
* А также для хранения шкалы надбавок за стаж добавим в справочник табличную часть *Расш1\_ШкалаНадбавокЗаСтаж*с реквизитами:
* *НижняяГраница* (тип *Число* 2, 0),
* *ВерхняяГраница* (тип *Число* 2, 0),
* *ПроцентНадбавки* (тип *Число* 2, 0),

Теперь в расширении у заимствованного справочника *ВидыСтажа* создадим основную форму элемента. В расширяемой конфигурации этой формы не было, поэтому для редактирования элементов справочника использовалась автоматически генерируемая форма. Теперь нам нужна специальная форма, чтобы реализовать в ней нужную функциональность.

В этой форме у поля, отображающего значения реквизита справочника *Расш1\_ВидНадбавкиЗаСтаж*, установим свойство *Вид* в значение *Поле переключателя*, свойство *ВидПереключателя* в значение *Тумблер*, а свойство *СписокВыбора* заполним значениями перечисления *Расш1\_ВидыНадбавокЗаСтаж* (рис. 4.8).



**Рис. 4.8.**Шкала надбавок стажей, используемых в тарификации

Создадим обработчик события *ПриИзменении* флажка *Расш1\_ИспользуетсяВТарификации* и заполним его следующим образом (листинг 4.1).

**Листинг 4.1.** Обработчик «Расш1\_ ИспользуетсяВТарификацииПриИзменении»

&НаКлиенте

Процедура Расш1\_ИспользуетсяВТарификацииПриИзменении(Элемент)

ОпределитьДоступностьЭлементов(ЭтотОбъект);

КонецПроцедуры

В процедуре *ОпределитьДоступностьЭлементов()* устанавливается видимость поля переключателя и таблицы со шкалой надбавок в зависимости от состояния флажка *Расш1\_ИспользуетсяВТарификации* (листинг 4.2).

**Листинг 4.2.** Процедура «ОпределитьДоступностьЭлементов»

&НаКлиентеНаСервереБезКонтекста

Процедура ОпределитьДоступностьЭлементов(Форма)

Если Форма.Объект.Расш1\_ИспользуетсяВТарификации Тогда

Форма.Элементы.Расш1\_ШкалаНадбавокЗаСтаж.Видимость = Истина;

Форма.Элементы.Расш1\_ВидНадбавкиЗаСтаж.Видимость = Истина;

Иначе

Форма.Элементы.Расш1\_ШкалаНадбавокЗаСтаж.Видимость = Ложь;

Форма.Элементы.Расш1\_ВидНадбавкиЗаСтаж.Видимость = Ложь;

КонецЕсли;

КонецПроцедуры

Продублируем определение видимости элементов формы в обработчике события *ПриСозданииНаСервере* (листинг 4.3).

**Листинг 4.3.** Обработчик «ПриСозданииНаСервере»

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

ОпределитьДоступностьЭлементов(ЭтотОбъект);

КонецПроцедуры

Таким образом, данные, специфичные для тарификации (поле переключателя и таблица со шкалой надбавок), будут показываться только для тех видов стажа, у которых установлен флажок *Используется в тарификации*.

### Разработка документа для тарификации сотрудников

Теперь, когда подготовительная работа завершена, добавим в расширение собственный документ для тарификации сотрудников по отделениям. В целях упрощения примера в табличной части документа напрямую будут содержаться все необходимые колонки, с помощью которых реализуется алгоритм расчета ФОТ, описанный в разделе [«Базовые понятия тарификации и алгоритм расчета ФОТ»](#_Базовые_понятия_тарификации). Некоторые поля просто дублируются из справочника *Сотрудники*.

Добавим в расширение документ *Расш1\_ТарификацияСотрудников* с синонимом «Тарификация сотрудников» и укажем, что он будет виден в подсистеме *Расш1\_Тарификация*. Документ будет содержать реквизит *Отделение* (тип *СправочникСсылка.Отделения*) и табличную часть *Тарификация* с реквизитами:

* *Сотрудник* (тип *СправочникСсылка.Сотрудники*),
* *ТабельныйНомер* (тип *Строка* 5),
* *Должность* (тип *СправочникСсылка.Должности*),
* *ОкладПоПКУ* (тип *Число* 10, 2),
* *ПроцентНадбавкиЗаРуководство*(тип *Число* 2, 0),
* *НадбавкаЗаРуководство*(тип *Число* 10, 0),
* *ПроцентНадбавкиЗаУченуюСтепень*(тип *Число* 2, 0),
* *НадбавкаЗаУченуюСтепень*(тип *Число* 10, 0),
* *ПроцентНадбавкиЗаПочетноеЗвание*(тип *Число* 2, 0),
* *НадбавкаЗаПочетноеЗвание*(тип *Число* 10, 0),
* *ИтогоОклад* (тип *Число* 10, 2),
* *ЗанимаемаяСтавка*(тип *Число* 5, 2),
* *МесячныйОклад* (тип *Число* 10, 2),
* *ПроцентНадбавкиЗаВредность* (тип *Число* 2, 0),
* *НадбавкаЗаВредность* (тип *Число* 10, 2),
* *ПроцентНадбавкиЗаНепрерывныйСтаж*(тип *Число* 2, 0),
* *ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж*(тип *Число* 2, 0),
* *НадбавкаЗаСтаж*(тип *Число* 10, 2),
* *ФОТ*(тип *Число* 10, 2).

Перед тем как добавлять документ в расширение, нам нужно позаимствовать из расширяемой конфигурации справочник *Сотрудники* (справочники *Должности* и *Отделения* мы уже заимствовали ранее). Это необходимо для того, чтобы в нашем расширении были доступны нужные типы данных.

Теперь создадим основную форму документа и реализуем в ней нужную функциональность.

Прежде всего сделаем так, чтобы при изменении отделения список сотрудников в табличной части документа заполнялся сотрудниками (и их кадровыми данными), работающими в отделении на дату создания документа. И чтобы при этом каждому сотруднику назначались бы полагающиеся ему надбавки за вредность и за стаж (в процентном отношении). И затем рассчитывался бы ФОТ сотрудника.

Для этого создадим обработчик события *ПриИзменении* поля формы документа *Отделение* и заполним его следующим образом (листинг 4.4).

**Листинг 4.4.** Обработчик события «ОтделениеПриИзменении»

&НаКлиенте

Процедура ОтделениеПриИзменении(Элемент)

Объект.Тарификация.Очистить();

ЗаполнитьСписокСотрудников(Истина, Истина);

КонецПроцедуры

При изменении отделения мы очищаем табличную часть документа *Тарификация* и заполняем ее тарификационными данными сотрудников отделения в процедуре *ЗаполнитьСписокСотрудников()*, которую мы подробно разберем ниже (см. листинг 4.6).

Чтобы случайно не удалить ранее введенные данные по тарификации, в обработчике события формы документа *ПриЧтенииНаСервере* укажем, что поле *Отделение* будет недоступно для изменения для всех существующих документов с заполненной табличной частью (листинг 4.5).

**Листинг 4.5.** Обработчик события «ПриЧтенииНаСервере»

&НаСервере

Процедура ПриЧтенииНаСервере(ТекущийОбъект)

Если Объект.Тарификация.Количество() > 0 Тогда

Элементы.Отделение.Доступность = Ложь;

КонецЕсли;

КонецПроцедуры

**Листинг 4.6.** Процедура «ЗаполнитьСписокСотрудников»

&НаСервере

Процедура ЗаполнитьСписокСотрудников(

ЗаполнитьСтаж, ЗаполнитьВредность);

ЗапросСотрудники = ПолучитьЗапросСотрудники();

ЗапросСотрудники.УстановитьПараметр("Отделение", Объект.Отделение);

ЗапросСотрудники.УстановитьПараметр("ДатаТарификации", Объект.Дата);

РезультатЗапроса = ЗапросСотрудники.Выполнить();

ВыборкаСотрудники = РезультатЗапроса.Выбрать();

Пока ВыборкаСотрудники.Следующий() Цикл

Строки = Объект.Тарификация.НайтиСтроки(Новый Структура("Сотрудник", ВыборкаСотрудники.Сотрудник));

Если Строки.Количество() > 0 Тогда

СтрокаСотрудника = Строки[0];

Иначе

СтрокаСотрудника = Объект.Тарификация.Добавить();

СтрокаСотрудника.Сотрудник = ВыборкаСотрудники.Сотрудник;

КонецЕсли;

СтрокаСотрудника.ТабельныйНомер = ВыборкаСотрудники.ТабельныйНомер;

СтрокаСотрудника.Должность = ВыборкаСотрудники.Должность;

СтрокаСотрудника.ОкладПоПКУ = ВыборкаСотрудники.ОкладПоПКУ;

СтрокаСотрудника.ЗанимаемаяСтавка = ВыборкаСотрудники.ОбъемРаботы;

Если ЗаполнитьСтаж Тогда

РассчитатьНадбавкуЗаСтаж(СтрокаСотрудника);

КонецЕсли;

Если ЗаполнитьВредность Тогда

РассчитатьНадбавкуЗаВредность(СтрокаСотрудника);

КонецЕсли;

РассчитатьФОТПоСтроке(СтрокаСотрудника);

КонецЦикла;

КонецПроцедуры

В процедуре *ЗаполнитьСписокСотрудников()* сначала с помощью функции *ПолучитьЗапросСотрудники()* мы получаем запрос к справочнику *Сотрудники*, который выбирает сотрудников, работающих в отделении на дату документа (см. листинг 4.7).

Затем устанавливаем значения параметров запроса *Отделение* и *ДатаТарификации* равными отделению и дате документа. Выполняем запрос и обходим выборку результата запроса в цикле.

Для каждого сотрудника, полученного запросом, проверяем, имеются ли в табличной части *Тарификация* строки со ссылкой на этого сотрудника. Если да, то для простоты считаем, что каждый сотрудник отделения используется в табличной части только один раз, – и берем первую найденную строку. Если нет, добавляем в табличную часть строку со ссылкой на полученного запросом сотрудника.

Затем заполняем кадровые поля табличной части (*Сотрудник*, *ТабельныйНомер*, *Должность*, *ОкладПоПКУ*, *ЗанимаемаяСтавка*) соответствующими данными полей выборки запроса. Поскольку кадровые данные сотрудников не могут быть изменены в документе по тарификации, у соответствующих им колонок таблицы формы установим свойство *ТолькоПросмотр*. Кроме того, установим для этих колонок свойство *ФиксацияВТаблице* в значение *Лево*.

Далее, если параметр *ЗаполнитьСтаж*, переданный в процедуру, истинен, то мы вызываем процедуру *РассчитатьНадбавкуЗаСтаж()* (см. листинг 4.8), в которой стажи текущего сотрудника пересчитываются на дату документа и заполняются соответствующие им проценты надбавок за стаж в строке табличной части.

Если параметр *ЗаполнитьВредность*, переданный в процедуру, истинен, то мы вызываем процедуру *РассчитатьНадбавкуЗаВредность()* (см. листинг 4.11), в которой текущему сотруднику устанавливается процент надбавки за вредность, который соответствует должности сотрудника в штатном расписании отделения.

После этого в процедуре *РассчитатьФОТПоСтроке()* пересчитывается ФОТ текущего сотрудника (см. листинг 4.13).

**Листинг 4.7.** Функция «ПолучитьЗапросСотрудники»

&НаСервереБезКонтекста

Функция ПолучитьЗапросСотрудники()

ЗапросСотрудники = Новый Запрос;

ЗапросСотрудники.Текст =

"ВЫБРАТЬ

|        Сотрудники.Ссылка КАК Сотрудник,

|        Сотрудники.ОкладПоПКУ КАК ОкладПоПКУ,

|        Сотрудники.ОбъемРаботы КАК ОбъемРаботы,

|        Сотрудники.ТабельныйНомер КАК ТабельныйНомер,

|        Сотрудники.Должность КАК Должность,

|        Сотрудники.Отделение КАК Отделение

|ИЗ

|        Справочник.Сотрудники КАК Сотрудники

|ГДЕ

|        Сотрудники.Отделение = &Отделение

|                И Сотрудники.ДатаПриема <= &ДатаТарификации

|                И (Сотрудники.Уволен = ЛОЖЬ ИЛИ Сотрудники.ДатаУвольнения > &ДатаТарификации)

|

|УПОРЯДОЧИТЬ ПО

|        ТабельныйНомер";

Возврат ЗапросСотрудники;

КонецФункции

Для корректной работы запроса в расширении нужно не забыть позаимствовать из справочника *Сотрудники* все реквизиты, которые используются в запросе:

* *ОкладПоПКУ*,
* *ОбъемРаботы*,
* *ТабельныйНомер*,
* *Должность*,
* *Отделение*,
* *ДатаПриема*,
* *Уволен*,
* *ДатаУвольнения*.

**Листинг 4.8.** Процедура «РассчитатьНадбавкуЗаСтаж»

&НаСервере

Процедура РассчитатьНадбавкуЗаСтаж(СтрокаСотрудника)

Сотрудник = СтрокаСотрудника.Сотрудник;

ПересчитатьСтажиНаДату(Сотрудник);

ЗапросСтажи = ПолучитьЗапросСтажи();

ЗапросСтажи.УстановитьПараметр("Сотрудник", Сотрудник);

РезультатЗапроса = ЗапросСтажи.Выполнить();

ВыборкаСтажи = РезультатЗапроса.Выбрать();

СтрокаСотрудника.ПроцентНадбавкиЗаНепрерывныйСтаж = 0;

СтрокаСотрудника.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж = 0;

Пока ВыборкаСтажи.Следующий() Цикл

Если ВыборкаСтажи.Лет >= ВыборкаСтажи.НижняяГраница

И ВыборкаСтажи.Лет < ВыборкаСтажи.ВерхняяГраница Тогда

Если ВыборкаСтажи.ВидНадбавкиЗаСтаж = Перечисления.Расш1\_ВидыНадбавокЗаСтаж.ПроцентНадбавкиЗаНепрерывныйСтаж Тогда

СтрокаСотрудника.ПроцентНадбавкиЗаНепрерывныйСтаж = ВыборкаСтажи.ПроцентНадбавки;

ИначеЕсли ВыборкаСтажи.ВидНадбавкиЗаСтаж = Перечисления.Расш1\_ВидыНадбавокЗаСтаж.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж Тогда

СтрокаСотрудника.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж = ВыборкаСтажи.ПроцентНадбавки;

КонецЕсли;

КонецЕсли;

КонецЦикла;

КонецПроцедуры

В процедуре *РассчитатьНадбавкуЗаСтаж()* мы получаем ссылку на текущего сотрудника из соответствующего поля строки табличной части, переданной в процедуру в качестве параметра.

Сначала с помощью процедуры *ПересчитатьСтажиНаДату()* (листинг 4.9) мы пересчитываем стажи сотрудника на дату документа.

Затем с помощью функции *ПолучитьЗапросСтажи()* мы получаем запрос к табличной части *Стажи* справочника *Сотрудники*, который выбирает размер стажей сотрудника, использующихся в тарификации, и соответствующих им шкал надбавок за стаж (см. листинг 4.10).

После этого устанавливаем значение параметра запроса *Сотрудник* равным ссылке на текущего сотрудника. Выполняем запрос и обходим выборку результата запроса в цикле.

Для каждого вида стажа, полученного запросом, находим градацию шкалы надбавок за стаж, в которую попадает число лет стажа сотрудника, и получаем соответствующий процент надбавки за стаж. Сравнив вид надбавки за стаж со значениями перечисления *Расш1\_ВидыНадбавокЗаСтаж*, мы заполняем этим значением либо поле *ПроцентНадбавкиЗаНепрерывныйСтаж*, либо *ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж* в текущей строке сотрудника.

**Листинг 4.9.** Процедура «ПересчитатьСтажиНаДату»

&НаСервере

Процедура ПересчитатьСтажиНаДату(Сотрудник)

ДатаДокумента = Объект.Дата;

ДатаПересчета = Сотрудник.ДатаПересчетаСтажа;

Если ЗначениеЗаполнено(ДатаПересчета) Тогда

Разница = (ДатаДокумента – ДатаПересчета);

СотрудникОбъект = Сотрудник.ПолучитьОбъект();

СотрудникОбъект.ПересчитатьСтажи(Разница);

СотрудникОбъект.ДатаПересчетаСтажа = ДатаДокумента;

СотрудникОбъект.Записать();

КонецЕсли;

КонецПроцедуры

В процедуре *ПересчитатьСтажиНаДату()* из переданной ссылки на сотрудника получается объект и вызывается метод *ПересчитатьСтажи()*, реализованный в модуле объекта справочника *Сотрудники* в расширяемой конфигурации. С помощью этого метода размер всех видов стажей сотрудника изменяется на разницу между датой документа и датой последнего пересчета стажей сотрудника. После чего дата пересчета стажей становится равной дате документа.

Для корректной работы процедуры в расширении нужно не забыть позаимствовать из справочника *Сотрудники* реквизит *ДатаПересчетаСтажа*, который используется в процедуре.

**Листинг 4.10.** Функция «ПолучитьЗапросСтажи»

&НаСервереБезКонтекста

Функция ПолучитьЗапросСтажи()

ЗапросСтажи = Новый Запрос;

ЗапросСтажи.Текст =

"ВЫБРАТЬ

|        СотрудникиСтажи.ВидСтажа КАК ВидСтажа,

|        СотрудникиСтажи.Лет КАК Лет,

|        СотрудникиСтажи.Месяцев КАК Месяцев,

|        СотрудникиСтажи.Дней КАК Дней,

|        ШкалаНадбавокЗаСтаж.НижняяГраница КАК НижняяГраница,

|        ШкалаНадбавокЗаСтаж.ВерхняяГраница КАК ВерхняяГраница,

|        ШкалаНадбавокЗаСтаж.ПроцентНадбавки КАК ПроцентНадбавки,

|        ШкалаНадбавокЗаСтаж.Ссылка.Расш1\_ВидНадбавкиЗаСтаж КАК ВидНадбавкиЗаСтаж

|ИЗ

|        Справочник.Сотрудники.Стажи КАК СотрудникиСтажи

|                ЛЕВОЕ СОЕДИНЕНИЕ Справочник.ВидыСтажа.Расш1\_ШкалаНадбавокЗаСтаж КАК ШкалаНадбавокЗаСтаж

|                ПО СотрудникиСтажи.ВидСтажа = ШкалаНадбавокЗаСтаж.Ссылка

|ГДЕ

|        СотрудникиСтажи.Ссылка = &Сотрудник

|                 И ШкалаНадбавокЗаСтаж.Ссылка.Расш1\_ИспользуетсяВТарификации = ИСТИНА";

Возврат ЗапросСтажи;

КонецФункции

Для корректной работы запроса в расширении нужно не забыть позаимствовать из табличной части *Стажи* справочника *Сотрудники* все реквизиты, которые используются в запросе:

* *ВидСтажа*,
* *Лет*,
* *Месяцев*,
* *Дней*.

**Листинг 4.11.** Процедура «РассчитатьНадбавкуЗаВредность»

&НаСервере

Процедура РассчитатьНадбавкуЗаВредность(СтрокаСотрудника)

ЗапросВредность = ПолучитьЗапросВредность();

ЗапросВредность.УстановитьПараметр("Отделение", Объект.Отделение);

ЗапросВредность.УстановитьПараметр("Должность", СтрокаСотрудника.Должность);

РезультатЗапроса = ЗапросВредность.Выполнить();

Если НЕ РезультатЗапроса.Пустой() Тогда

ВыборкаВредность = РезультатЗапроса.Выбрать();

ВыборкаВредность.Следующий();

СтрокаСотрудника.ПроцентНадбавкиЗаВредность = ВыборкаВредность.ПроцентНадбавкиЗаВредность;

КонецЕсли;

КонецПроцедуры

В процедуре *РассчитатьНадбавкуЗаВредность()* сначала с помощью функции *ПолучитьЗапросВредность()* мы получаем запрос к штатному расписанию, который выбирает процент надбавки за вредность, установленный для конкретной должности в штатном расписании отделения (см. листинг 4.12).

Затем устанавливаем значения параметров запроса *Отделение* и *Должность* равными отделению, указанному в документе, и должности, полученной из соответствующего поля текущей строки сотрудника, переданной в процедуру в качестве параметра.

Выполняем запрос и получаем процент надбавки за вредность, который соответствует должности сотрудника в штатном расписании отделения. И заполняем этим значением поле *ПроцентНадбавкиЗаВредность* в текущей строке сотрудника.

**Листинг 4.12.** Функция «ПолучитьЗапросВредность»

&НаСервереБезКонтекста

Функция ПолучитьЗапросВредность()

ЗапросВредность = Новый Запрос;

ЗапросВредность.Текст =

"ВЫБРАТЬ ПЕРВЫЕ 1

|        ШтатноеРасписаниеШтаты.ПроцентНадбавкиЗаВредность КАК ПроцентНадбавкиЗаВредность

|ИЗ

|        Справочник.ШтатноеРасписание.Штаты КАК ШтатноеРасписаниеШтаты

|ГДЕ

|        ШтатноеРасписаниеШтаты.Должность = &Должность

|                 И ШтатноеРасписаниеШтаты.Ссылка.Отделение = &Отделение";

Возврат ЗапросВредность;

КонецФункции

Для корректной работы запроса в расширении нужно не забыть позаимствовать из справочника *ШтатноеРасписание* реквизит *Отделение* и реквизиты табличной части *Штаты*, которые используются в запросе:

* *Должность*,
* *ПроцентНадбавкиЗаВредность*.

В заключение выполняется процедура *РассчитатьФОТПоСтроке()*, в результате чего пересчитывается ФОТ сотрудника в соответствии с алгоритмом, описанным в разделе [«Базовые понятия тарификации и алгоритм расчета ФОТ»](#_Базовые_понятия_тарификации) (листинг 4.13).

**Листинг 4.13.** Процедура «РассчитатьФОТПоСтроке»

&НаКлиентеНаСервереБезКонтекста

Процедура РассчитатьФОТПоСтроке(СтрокаТЧ)

СтрокаТЧ.НадбавкаЗаРуководство = ОКР(СтрокаТЧ.ОкладПоПКУ \* СтрокаТЧ.ПроцентНадбавкиЗаРуководство/100, 0);

СтрокаТЧ.НадбавкаЗаУченуюСтепень = ОКР(СтрокаТЧ.ОкладПоПКУ \* СтрокаТЧ.ПроцентНадбавкиЗаУченуюСтепень/100, 0);

СтрокаТЧ.НадбавкаЗаПочетноеЗвание = ОКР(СтрокаТЧ.ОкладПоПКУ \* СтрокаТЧ.ПроцентНадбавкиЗаПочетноеЗвание/100, 0);

СтрокаТЧ.ИтогоОклад = СтрокаТЧ.ОкладПоПКУ + СтрокаТЧ.НадбавкаЗаРуководство + СтрокаТЧ.НадбавкаЗаУченуюСтепень + СтрокаТЧ.НадбавкаЗаПочетноеЗвание;

СтрокаТЧ.МесячныйОклад = СтрокаТЧ.ИтогоОклад \* СтрокаТЧ.ЗанимаемаяСтавка;

СтрокаТЧ.НадбавкаЗаВредность = СтрокаТЧ.МесячныйОклад \* СтрокаТЧ.ПроцентНадбавкиЗаВредность/100;

ПроцентНадбавкиЗаСтаж = ?(СтрокаТЧ.ПроцентНадбавкиЗаНепрерывныйСтаж > СтрокаТЧ.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж, СтрокаТЧ.ПроцентНадбавкиЗаНепрерывныйСтаж, СтрокаТЧ.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж);

СтрокаТЧ.НадбавкаЗаСтаж = СтрокаТЧ.МесячныйОклад \* ПроцентНадбавкиЗаСтаж/100;

СтрокаТЧ.ФОТ = СтрокаТЧ.МесячныйОклад + СтрокаТЧ.НадбавкаЗаВредность + СтрокаТЧ.НадбавкаЗаСтаж;

КонецПроцедуры

В целом алгоритм расчета ФОТ довольно простой и понятный. Отдельно остановимся на формуле расчета процента надбавки за стаж:

*ПроцентНадбавкиЗаСтаж =*

*?(СтрокаТЧ.ПроцентНадбавкиЗаНепрерывныйСтаж >*

*СтрокаТЧ.ПроцентНадбавки ЗаНепрерывныйУчастковыйСтаж,*

*СтрокаТЧ.ПроцентНадбавкиЗаНепрерывныйСтаж,*

*СтрокаТЧ.ПроцентНадбавки ЗаНепрерывныйУчастковыйСтаж);*

Эта формула означает, что для расчета надбавки за непрерывный стаж используется тот показатель – за непрерывный стаж в ЛПУ или за непрерывный участковый стаж, – где процент надбавки больше. В результате проценты надбавки за стаж у нас разбиты на две колонки (по видам стажа), а сумма надбавки за стаж отражается в общей колонке.

Теперь создадим обработчик события *ПриИзменении* у колонки *ПроцентНадбавкиЗаРуководство* таблицы формы *Тарификация*, отображающей данные табличной части документа, и заполним его следующим образом (листинг 4.14).

**Листинг 4.14.** Обработчик события «ПроцентНадбавкиЗаРуководствоПриИзменении»

&НаКлиенте

Процедура ТарификацияПроцентНадбавкиЗаРуководствоПриИзменении(Элемент)

СтрокаТЧ = Элементы.Тарификация.ТекущиеДанные;

РассчитатьФОТПоСтроке(СтрокаТЧ);

КонецПроцедуры

В этом обработчике мы получаем текущие данные строки табличной части, в которой произошли изменения, и передаем эту строку в процедуру для расчета ФОТ.

Создадим аналогичные обработчики событий *ПриИзменении* для всех колонок таблицы *Тарификация*, при изменении которых пользователем должен пересчитываться ФОТ:

* *ПроцентНадбавкиЗаРуководство*,
* *ПроцентНадбавкиЗаУченуюСтепень*,
* *ПроцентНадбавкиЗаПочетноеЗвание*,
* *ПроцентНадбавкиЗаВредность*,
* *ПроцентНадбавкиЗаНепрерывныйСтаж*,
* *ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж*.

Соответственно, у всех расчетных колонок таблицы *Тарификация* установим свойство *ТолькоПросмотр*:

* *НадбавкаЗаРуководство*,
* *НадбавкаЗаУченуюСтепень*,
* *НадбавкаЗаПочетноеЗвание*,
* *ИтогоОклад*,
* *МесячныйОклад*,
* *НадбавкаЗаВредность*,
* *НадбавкаЗаСтаж*,
* *ФОТ*.

Кроме того, отразим суммы расчетных колонок в подвале таблицы, чтобы можно было видеть основные суммы, из которых складывается ФОТ по отделению, не формируя отчет. Для этого включим у таблицы формы *Тарификация* свойство *Подвал* и заполним свойство *ПутьКДаннымПодвала* у расчетных колонок следующим образом:

* *ОкладПоПКУ* – *Объект.Тарификация.ИтогОкладПоПКУ*,
* *ИтогоОклад* – *Объект.Тарификация.Итог*,
* *ЗанимаемаяСтавка* – *Объект.Тарификация.ИтогЗанимаемаяСтавка*,
* *МесячныйОклад* – *Объект.Тарификация.ИтогМесячныйОклад*,
* *НадбавкаЗаВредность* – *Объект.Тарификация.ИтогНадбавкаЗаВредность*,
* *НадбавкаЗаСтаж* – *Объект.Тарификация.ИтогНадбавкаЗаСтаж*,
* *ФОТ* – *Объект.Тарификация.ИтогФОТ*.

Поясним, что, хотя проценты надбавок за вредность и за стаж устанавливаются программно на основании данных в штатном расписании и стажевых данных сотрудников, у пользователя должна быть возможность их изменения, так как бывают ситуации, когда в информационную базу данные вносятся несвоевременно или с ошибками.

На всякий случай предоставим пользователю возможность вернуться к чисто программному заполнению надбавок за стаж, за вредность и заполнению списка сотрудников по нажатию отдельных кнопок.

Создадим команды формы *ЗаполнитьНадбавкуЗаВредность*, *ЗаполнитьНадбавкуЗаСтаж*, *ЗаполнитьСотрудниковОтделения* и перетащим их в командную панель формы. Из клиентских обработчиков первых двух команд будем вызывать следующие серверные процедуры (листинги 4.15, 4.16).

**Листинг 4.15.** Процедура «ЗаполнитьНадбавкуЗаВредностьНаСервере»

&НаСервере

Процедура ЗаполнитьНадбавкуЗаВредностьНаСервере()

Для Каждого ТекущаяСтрока Из Объект.Тарификация Цикл

РассчитатьНадбавкуЗаВредность(ТекущаяСтрока);

РассчитатьФОТПоСтроке(ТекущаяСтрока);

КонецЦикла;

КонецПроцедуры

В процедуре *ЗаполнитьНадбавкуЗаВредностьНаСервере()* мы обходим табличную часть документа *Тарификация* в цикле и для каждой строки вызываем процедуру *РассчитатьНадбавкуЗаВредность()*, которую мы подробно рассматривали в листинге 4.11. И затем пересчитываем ФОТ по строке табличной части.

**Листинг 4.16.** Процедура «ЗаполнитьНадбавкуЗаСтажНаСервере»

&НаСервере

Процедура ЗаполнитьНадбавкуЗаСтажНаСервере()

Для Каждого ТекущаяСтрока Из Объект.Тарификация Цикл

РассчитатьНадбавкуЗаСтаж(ТекущаяСтрока);

РассчитатьФОТПоСтроке(ТекущаяСтрока);

КонецЦикла;

КонецПроцедуры

В процедуре *ЗаполнитьНадбавкуЗаСтажНаСервере()* мы обходим табличную часть документа *Тарификация* в цикле и для каждой строки вызываем процедуру *РассчитатьНадбавкуЗаСтаж()*, которую мы подробно рассматривали в листинге 4.8. И затем пересчитываем ФОТ по строке табличной части.

Клиентский обработчик команды *ЗаполнитьСотрудниковОтделения* заполним следующим образом (листинг 4.17).

**Листинг 4.17.** Обработчик команды «ЗаполнитьСотрудниковОтделения»

&НаКлиенте

Процедура ЗаполнитьСотрудниковОтделения(Команда)

Если НЕ ЗначениеЗаполнено(Объект.Отделение)Тогда

ПоказатьПредупреждение(, "Отделение тарификации не может быть пустым.");

Возврат;

КонецЕсли;

ЗаполнитьСписокСотрудников(Ложь, Ложь);

ПоказатьОповещениеПользователя("Список сотрудников отделения заполнен. ФОТ сотрудников пересчитан.");

КонецПроцедуры

В этом обработчике, в случае если отделение документа заполнено, мы вызываем процедуру *ЗаполнитьСписокСотрудников()*, которую мы подробно рассматривали в листинге 4.6. И затем показываем пользователю соответствующее оповещение.

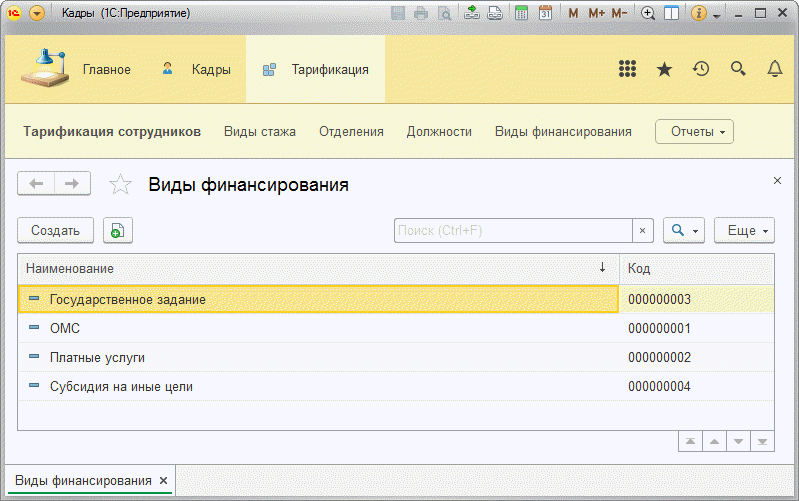
В результате выполнение этих трех команд вернет документ по тарификации сотрудников отделения к исходному состоянию.

## Ввод тарификационных данных в режиме «1С:Предприятие»

### Исходные данные для тарификации

Теперь введем исходные данные, на которых мы будем тестировать функциональность нашего расширения по тарификации сотрудников.

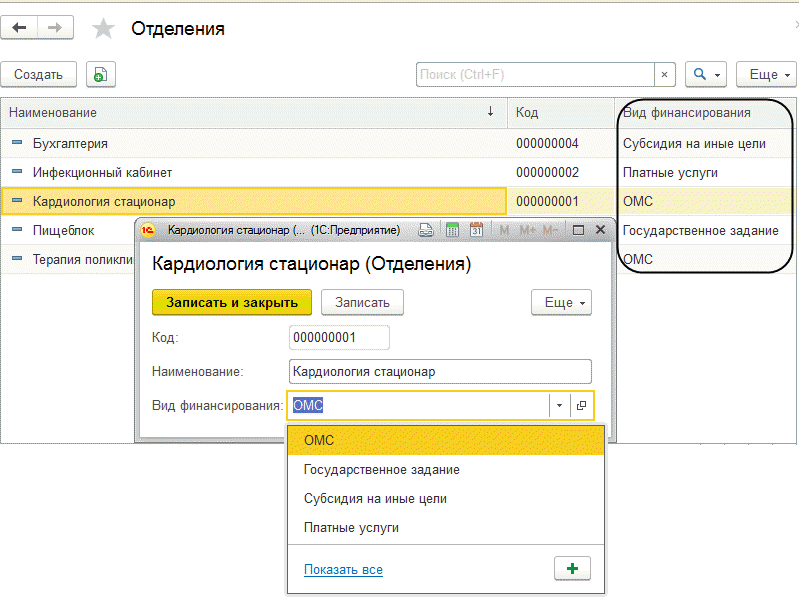
Сначала заполним справочник видов финансирования. Этот список нужен нам для того, чтобы иметь возможность получать отчеты по тарификации в разрезе видов финансирования (рис. 4.9).



**Рис. 4.9.**Справочник видов финансирования

Этого справочника не было в расширяемой кадровой конфигурации, но мы его добавили своим расширением для тарификации.

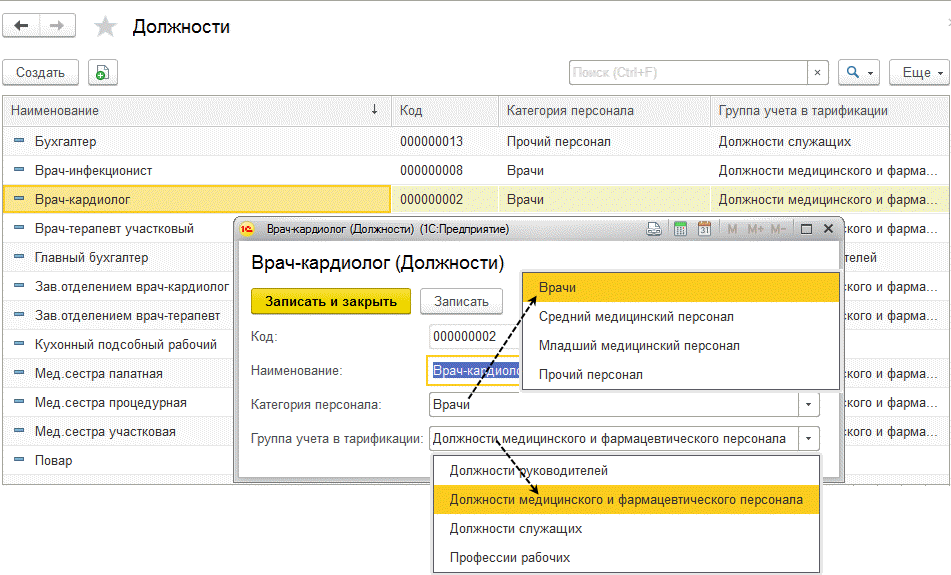
После этого заполним список отделений нашего ЛПУ. Укажем для каждого отделения, из какого источника оно финансируется (рис. 4.10).



**Рис. 4.10.**Справочник отделений

Справочник отделений уже был реализован в расширяемой кадровой конфигурации, но мы его расширили. То есть с помощью нашего расширения для тарификации добавили возможность вводить и хранить для каждого отделения вид финансирования.

Далее заполним список должностей, на которых будут работать сотрудники. Чтобы иметь возможность получать отчеты по тарификации в разрезе тарификационных категорий, укажем для каждой должности категорию персонала и тарификационную группу, к которой она относится (рис. 4.11).

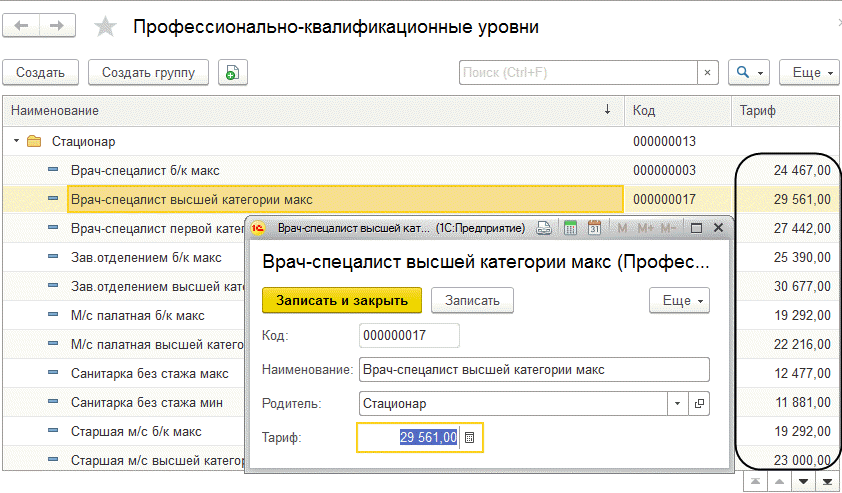


**Рис. 4.11.**Справочник должностей

Справочник должностей уже был реализован в расширяемой кадровой конфигурации с возможностью указать категорию персонала, к которой относится должность. Категория персонала ссылается на перечисление, существующее в расширяемой конфигурации. Но помимо чисто кадровой специфики мы будем использовать категорию персонала также и для тарификации сотрудников.

Кроме того, с помощью нашего расширения для тарификации мы добавили возможность вводить и хранить для каждой должности группу учета в тарификации, которая ссылается на собственное перечисление, добавленное расширением.

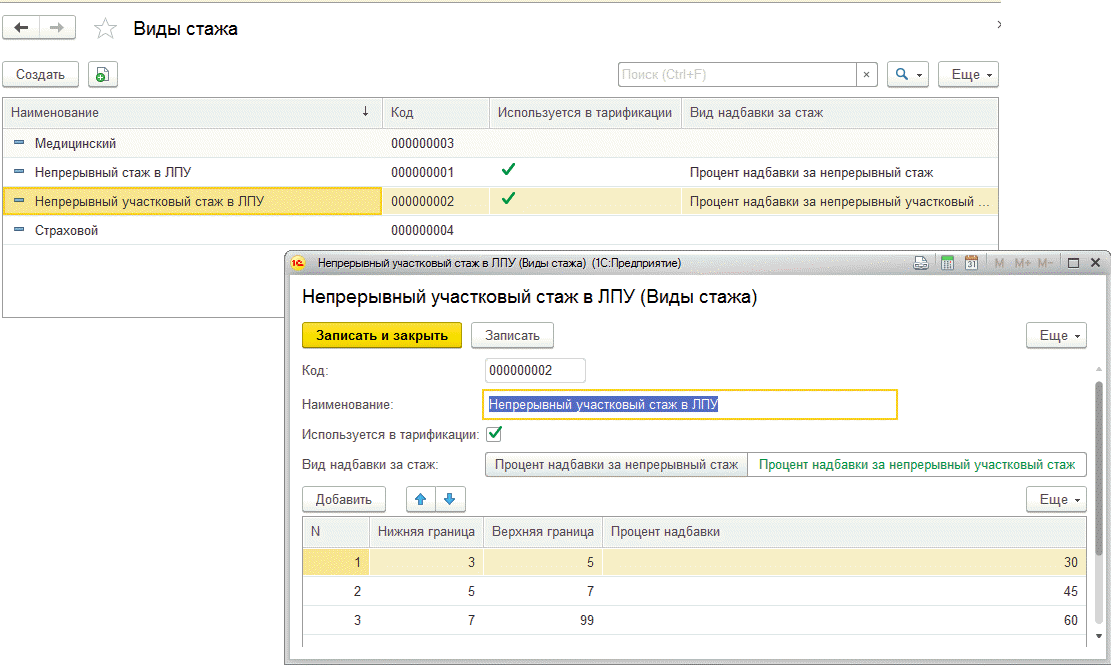
Затем заполним профессионально-квалификационные уровни (ПКУ), которые будут присваиваться сотрудникам. У каждого ПКУ установим тариф, на основе которого будет определяться должностной оклад сотрудника (рис. 4.12).



**Рис. 4.12.**Справочник ПКУ

Справочник ПКУ уже был реализован в расширяемой кадровой конфигурации. В расширении мы его не изменяли, но данные справочника мы также будем использовать для тарификации сотрудников.

Далее заполним справочник видов стажей. Для тех видов стажей, которые мы будем использовать для тарификации сотрудников, укажем признак *Используется в тарификации* и выберем вид надбавки за стаж (рис. 4.13).



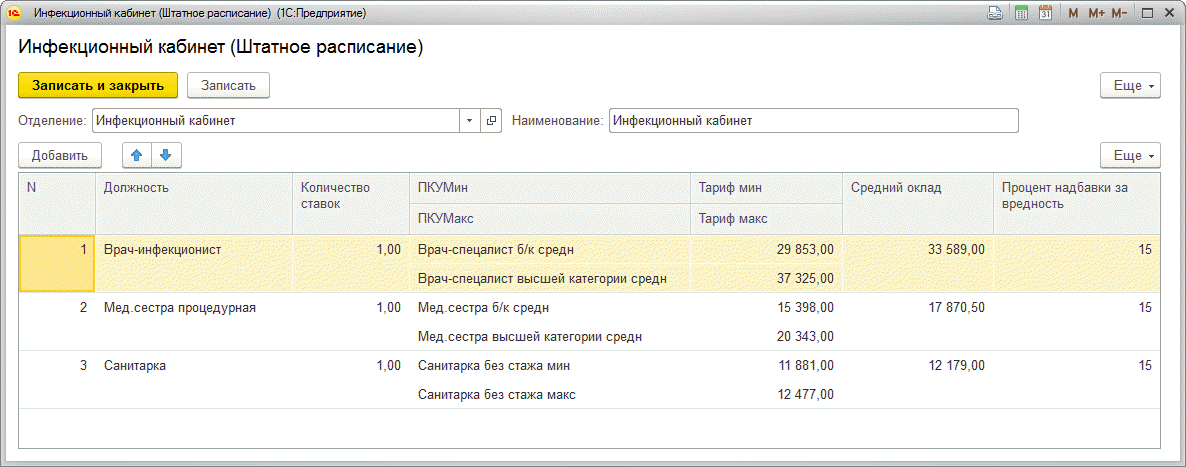
**Рис. 4.13.**Справочник видов стажа

Напомним, что для тарификации сотрудников мы будем использовать два вида стажа: *Непрерывный стаж в ЛПУ* (вид надбавки за стаж – *Процент надбавки за непрерывный стаж*) и *Непрерывный участковый стаж в ЛПУ* (вид надбавки за стаж – *Процент надбавки за непрерывный участковый стаж*).

И для каждого из этих стажей укажем свою шкалу надбавок за стаж. Это таблица, в которой временному интервалу (числу лет стажа сотрудника) соответствует определенный процент надбавки за стаж.

Чтобы реализовать требуемую функциональность, мы значительно расширили справочник видов стажа, существующий в расширяемой кадровой конфигурации. А именно – с помощью расширения для тарификации мы добавили возможность вводить и хранить для видов стажа признак использования в тарификации, вид надбавки за стаж и шкалу надбавок за стаж (в виде табличной части справочника, добавленной расширением). Виды надбавки за стаж (значения тумблера) выбираются из значений собственного расширения, которое мы также добавили в расширение.

Теперь зададим штатное расписание. Для каждого отделения заполним список должностей и количество ставок, предусмотренных для этой должности в штатном расписании отделения. Затем для каждой должности укажем минимальный и максимальный ПКУ, тарифы оплаты по которым будут определять так называемую «вилку окладов» по позиции штатного расписания. Кроме того, если для должности предусмотрена надбавка за вредность, укажем для нее процент надбавки за вредность (рис. 4.14):

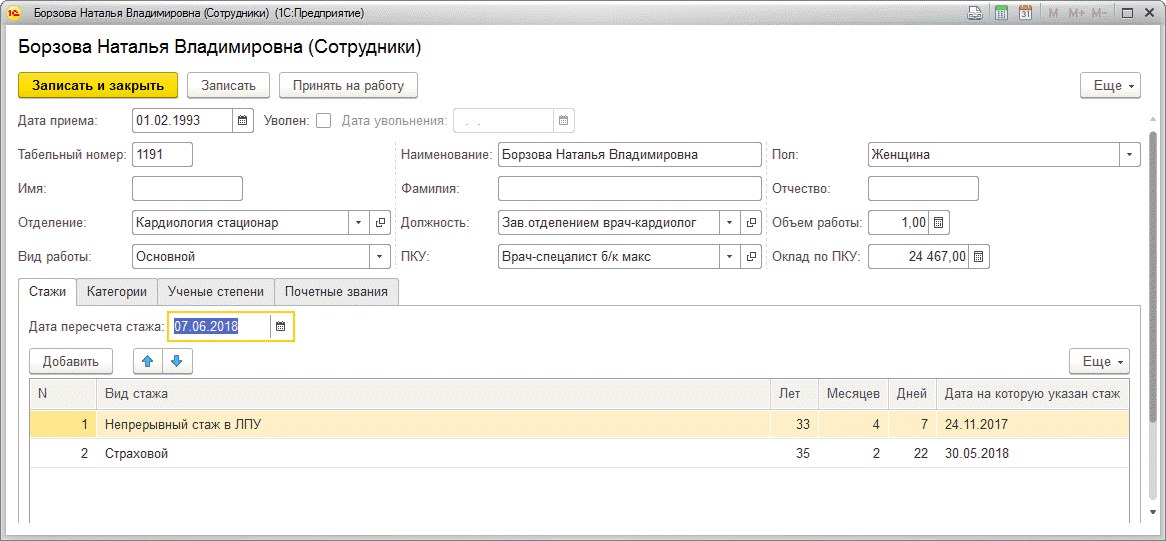


**Рис. 4.14.**Штатное расписание инфекционного кабинета

Средний оклад по должности автоматически рассчитывается как среднее арифметическое минимального и максимального тарифов. Этот усредненный оклад по должности будет затем использоваться для расчета ФОТ по строке вакансий.

Справочник *Штатное расписание* и вся его функциональность уже были реализованы в расширяемой кадровой конфигурации. В расширении мы его не изменяли, но данные штатного расписания мы также будем использовать для тарификации сотрудников.

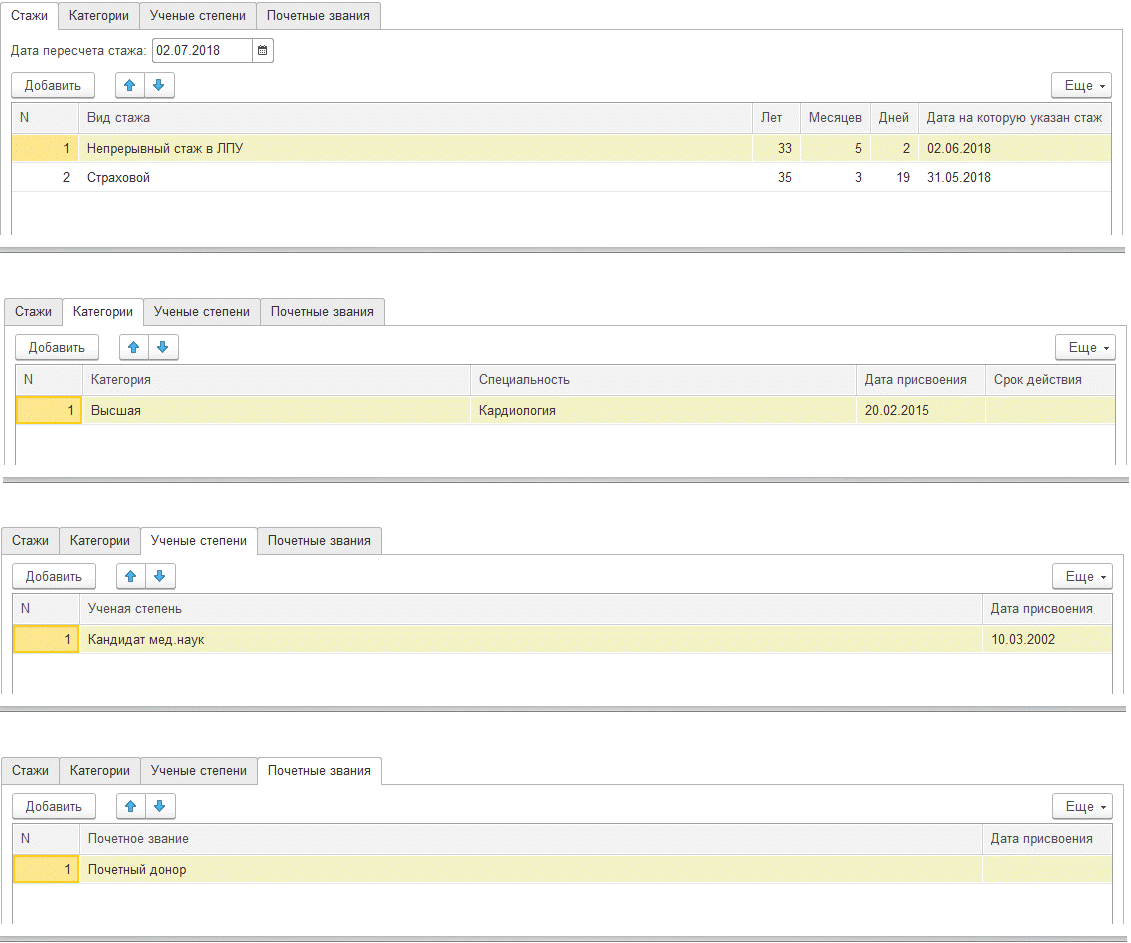
В заключение заполним список сотрудников нашего ЛПУ, который хранится в справочнике *Сотрудники*. Для каждого сотрудника укажем дату его приема на работу, отделение, должность, на которую он принимается, объем работы (количество занимаемых сотрудником ставок) и вид работы (основной, внешнее совместительство и т. п.), рис. 4.15.



**Рис. 4.15.**Сведения о сотруднике

При этом сотруднику автоматически подставится максимальный ПКУ и максимальный тариф оплаты, соответствующий этой должности в штатном расписании отделения. Однако ПКУ сотрудника можно изменить вручную, и оклад, соответствующий ПКУ, подставится автоматически.

На закладках *Стажи*, *Категории*, *Ученые степени*, *Почетные звания* введем данные о стажах, категориях, ученых степенях и почетных званиях сотрудника (рис. 4.16).



**Рис. 4.16.**Сведения о сотруднике

Справочник сотрудников и вся его функциональность уже были реализованы в расширяемой конфигурации. В расширении мы его не изменяли, но кадровые данные сотрудников мы, конечно же, будем использовать для тарификации.

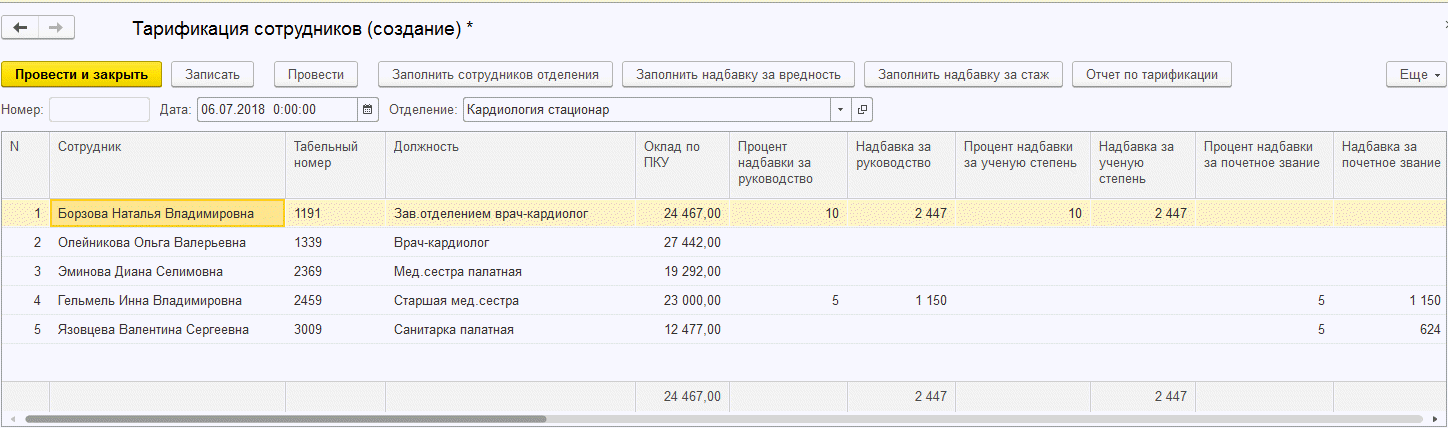
### Тестирование документа для тарификации сотрудников

Теперь мы можем приступать к проверке работы документа *Расш1\_ТарификацияСотрудников*, в котором выполняется собственно тарификация сотрудников и на основе которого формируются в дальнейшем все тарификационные отчеты. Разработку этого документа мы рассматривали подробно в разделе [«Разработка документа для тарификации сотрудников»](#_Разработка_документа_для). Проверим, как это работает.

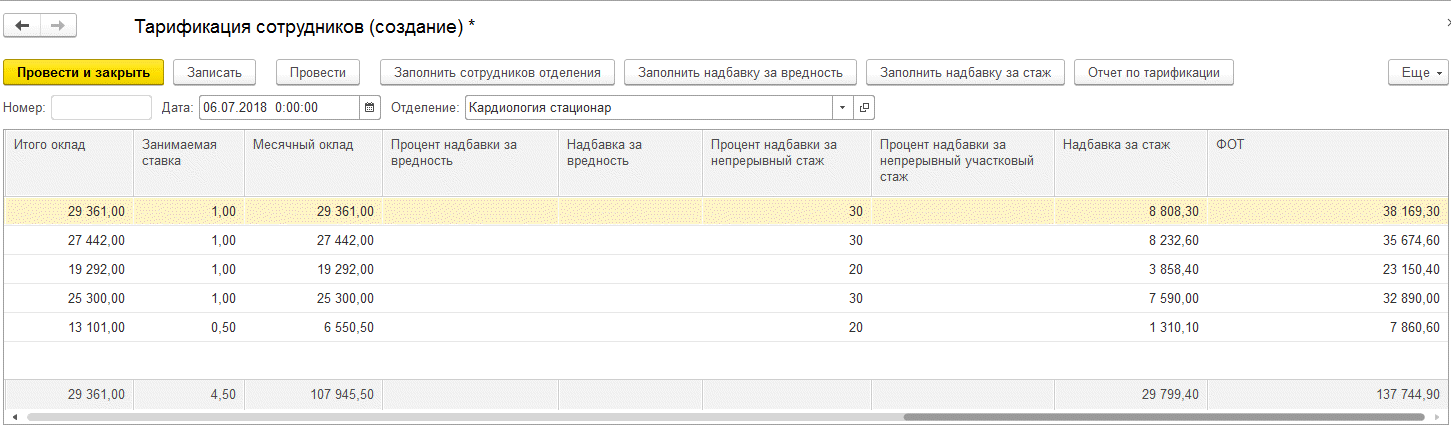
Запустим «1С:Предприятие» и выполним команду *Тарификация сотрудников* из раздела *Тарификация*. В списке документов тарификации нажмем кнопку *Создать*. Откроется форма нового документа *Тарификация сотрудников*.

Выберем отделение *Кардиология стационар*, и после этого таблица документа сразу же заполнится списком сотрудников (и их кадровыми данными), работающих в отделении на дату создания документа. Кроме того, каждому сотруднику заполнятся полагающиеся ему надбавки за вредность и за стаж (в процентном отношении). На основе этих данных будет рассчитан ФОТ сотрудника и заполнятся все остальные расчетные колонки таблицы (см. рис. 4.17, 4.18).

Тем сотрудникам, у которых есть для этого основания, введем проценты надбавок к окладу за руководство, за ученую степень и за почетное звание непосредственно в соответствующие колонки таблицы. При этом суммы надбавок, итоговый оклад, месячный оклад и ФОТ этих сотрудников пересчитаются (рис. 4.17, 4.18).



**Рис. 4.17.**Документ «Тарификация сотрудников» (часть 1)



**Рис. 4.18.**Документ «Тарификация сотрудников» (часть 2)

Проверим расчет ФОТ для сотрудницы Борзовой Натальи Владимировны. При приеме на работу в должности заведующей кардиологическим отделением ей был установлен оклад – 24 467,00. Этот оклад скопировался из кадровых данных в колонку *Оклад по ПКУ*.

Так как Борзова Н.В. руководит отделением, мы установили для нее 10-процентную надбавку за руководство. Поскольку у нее есть ученая степень (кандидат медицинских наук), мы установили ей 10-процентную надбавку за ученую степень. После этого пересчитались суммы надбавок в колонках *Надбавка за руководство* = 2 447 и *Надбавка за ученую степень* = 2 447. Обе эти надбавки составляют 10 % от оклада, но результат округляется до целого.

В сумме с окладом эти надбавки составляют *Итого оклад*: 24 467 + 2 447 + 2 447 = 29 361,00. Значение колонки *Занимаемая ставка* – 1,00 копируется из кадровых данных. Затем это значение умножается на итоговый оклад. Получается *Месячный оклад*: 29 361,00 \* 1,00 = 29 361,00.

И затем от него считается значение колонки *Надбавка за стаж*: 29 361,00 \* 30 / 100 = 8 808,30. Значение колонки *Процент надбавки за непрерывный стаж* заполнилось автоматически. Оно составляет 30 %, т. к. непрерывный стаж в ЛПУ у Борзовой Н.В. 33 года (больше 5 лет), что соответствует шкале надбавок для этого вида стажа. Причем перед начислением надбавки за стаж размер стажей каждого сотрудника пересчитывается на дату документа.

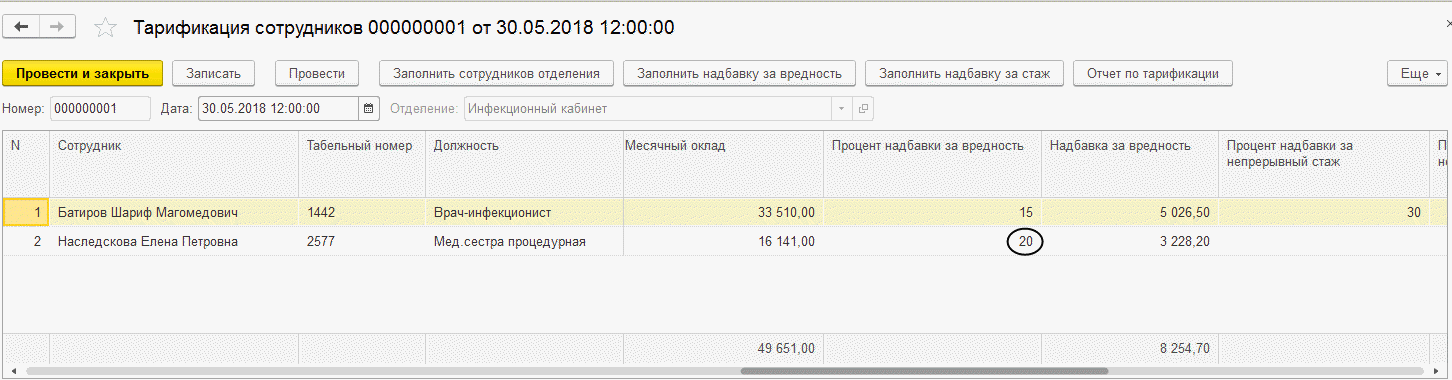
Итого ФОТ: 29 361 + 8 808,30 = 38 169,30. Надбавка за вредность этой сотруднице по должности не полагается.

Хотя проценты надбавок за вредность и за стаж устанавливаются программно на основании данных в штатном расписании и стажевых данных сотрудников, мы можем вручную изменить их непосредственно в соответствующих колонках таблицы документа, и суммы надбавок за вредность и за стаж (и, соответственно, ФОТ) сразу же пересчитаются. Такая возможность может пригодиться, в случае если данные в информационную базу были внесены несвоевременно или с ошибками.

Например, медицинской сестре, работающей в инфекционном кабинете, полагается 15-процентная надбавка за вредность, а в штатном расписании у этой должности надбавка за вредность отсутствует (не внесена по ошибке). Или стаж сотрудника достиг 5 лет, и ему полагается 30-процентная надбавка за стаж, но программно ему начисляется 20 %, т. к. стаж сотрудника изначально был внесен в базу меньше, чем он есть на самом деле.

Однако, если потребуется снова вернуться к программному заполнению надбавок, мы можем нажать кнопки *Заполнить надбавку за вредность* и *Заполнить надбавку за стаж*, и программные значения восстановятся.

Отдельно остановимся на тарификации сотрудников, имеющих надбавку за вредность. В штатном расписании инфекционного кабинета мы указали у всех должностей 15-процентную надбавку за вредность. При заполнении сотрудников этого отделения в документе по тарификации это значение перенеслось всем сотрудникам, работающим на должностях с установленной вредностью. Но, если нужно, прямо в документе мы можем вручную изменить процент надбавки за вредность у конкретного сотрудника, например на 20 % (рис. 4.19).



**Рис. 4.19.**Тарификация сотрудников, работающих с вредностью

Например, у сотрудницы инфекционного кабинета Наследсковой Елены Петровны мы установили процент надбавки за вредность – 20. Сумма надбавки (16 141,00 \* 20 / 100 = 3 228,20) и ФОТ сотрудника (16 141,00 + 3 228,20 = 19 369,20) сразу же пересчитаются. Надбавки за стаж у этой сотрудницы пока нет (т. к. непрерывный стаж только год).

А вот при расчете вакансий по этой позиции штатного расписания будет использоваться исходный процент надбавки – 15%.

## Отчет «Тарификация сотрудников отделения»

Теперь на основе введенных данных по тарификации сотрудников в нашем расширении мы разработаем два собственных, довольно похожих друг на друга отчета. Один для внутреннего использования – для сверки показателей тарификации сотрудников по отделениям и предоставления данных в бухгалтерию для расчета зарплаты сотрудников. Другой – для предоставления тарификационных списков сотрудников в вышестоящие инстанции, в разбивке по четырем регламентированным формам.

Сначала мы разработаем отчет, отображающий тарификацию сотрудников и итоговые фонды оплаты труда в разбивке по отделениям и поместим вызов этого отчета в разработанную ранее форму документа *Расш1\_ТарификацияСотрудников*.

Добавим в расширение отчет *Расш1\_ТарификацияСотрудниковОтделения*с синонимом «Тарификация сотрудников отделения» и укажем, что он будет виден в подсистеме *Расш1\_Тарификация*.

Затем создадим схему компоновки данных для отчета. Запрос для получения набора данных схемы компоновки для лучшего понимания разобьем на части и затем каждую из них прокомментируем.

### Запрос

Сначала нам нужно найти список самых последних на дату тарификации документов для каждого отделения. Для этого мы группируем документы по тарификации *Расш1\_ТарификацияСотрудников* по отделению, выбираем максимальную дату документа для каждого отделения и помещаем во временную таблицу *ДокументыТарификации*. Затем выбираем из табличной части *Тарификация* документов *Расш1\_ТарификацияСотрудников* (с максимальной датой тарификации по отделению) все поля, которые будут использоваться в отчете и помещаем во временную таблицу *ТарификацияСотрудников* (листинг 4.18).

**Листинг 4.18.** Фрагмент запроса для получения набора данных

ВЫБРАТЬ

МАКСИМУМ(ТарификацияОтделений.Дата) КАК ДатаДокумента,

ТарификацияОтделений.Отделение КАК Отделение

ПОМЕСТИТЬ ДокументыТарификации

ИЗ

Документ.Расш1\_ТарификацияСотрудников КАК ТарификацияОтделений

ГДЕ

ТарификацияОтделений.Дата <= &ДатаТарификации

И ТарификацияОтделений.Проведен = ИСТИНА

И ТарификацияОтделений.ПометкаУдаления = ЛОЖЬ

СГРУППИРОВАТЬ ПО

ТарификацияОтделений.Отделение

;

ВЫБРАТЬ

ТарификацияСотрудниковТарификация.Ссылка КАК Ссылка,

ТарификацияСотрудниковТарификация.Сотрудник КАК Сотрудник,

ТарификацияСотрудниковТарификация.ПроцентНадбавкиЗаРуководство КАК ПроцентНадбавкиЗаРуководство,

ТарификацияСотрудниковТарификация.НадбавкаЗаРуководство КАК НадбавкаЗаРуководство,

ТарификацияСотрудниковТарификация.ПроцентНадбавкиЗаУченуюСтепень КАК ПроцентНадбавкиЗаУченуюСтепень,

ТарификацияСотрудниковТарификация.НадбавкаЗаУченуюСтепень КАК НадбавкаЗаУченуюСтепень,

ТарификацияСотрудниковТарификация.ПроцентНадбавкиЗаПочетноеЗвание КАК ПроцентНадбавкиЗаПочетноеЗвание,

ТарификацияСотрудниковТарификация.НадбавкаЗаПочетноеЗвание КАК НадбавкаЗаПочетноеЗвание,

ТарификацияСотрудниковТарификация.ИтогоОклад КАК ИтогоОклад,

ТарификацияСотрудниковТарификация.ЗанимаемаяСтавка КАК ЗанимаемаяСтавка,

ТарификацияСотрудниковТарификация.МесячныйОклад КАК МесячныйОклад,

ТарификацияСотрудниковТарификация.ПроцентНадбавкиЗаВредность КАК ПроцентНадбавкиЗаВредность,

ТарификацияСотрудниковТарификация.НадбавкаЗаВредность КАК НадбавкаЗаВредность,

ТарификацияСотрудниковТарификация.ПроцентНадбавкиЗаНепрерывныйСтаж КАК ПроцентНадбавкиЗаНепрерывныйСтаж,

ТарификацияСотрудниковТарификация.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж КАК ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж,

ТарификацияСотрудниковТарификация.НадбавкаЗаСтаж КАК НадбавкаЗаСтаж,

ТарификацияСотрудниковТарификация.ФОТ КАК ФОТ

ПОМЕСТИТЬ ТарификацияСотрудников

ИЗ

Документ.Расш1\_ТарификацияСотрудников.Тарификация КАК ТарификацияСотрудниковТарификация

ВНУТРЕННЕЕ СОЕДИНЕНИЕ ДокументыТарификации

ПО ТарификацияСотрудниковТарификация.Ссылка.Отделение = ДокументыТарификации.Отделение

И ТарификацияСотрудниковТарификация.Ссылка.Дата = ДокументыТарификации.ДатаДокумента

;

Для получения актуальных данных из табличной части документов тарификации мы связываем внутренним соединением таблицу *Документ.Расш1\_ТарификацияСотрудников.Тарификация* с временной таблицей *ДокументыТарификации* по полям *Подразделение* и *Дата*. И затем помещаем эти данные во временную таблицу *ТарификацияСотрудников*, которую мы будем использовать в итоговой выборке запроса.

Затем мы находим последние актуальные категории сотрудника по каждой специальности. Для этого мы выбираем из табличной части *Категории* справочника *Сотрудники* максимальные даты присвоения категории для каждого сотрудника по каждой специальности и помещаем во временную таблицу *МаксимальныеСотрудникиКатегории*. И затем получаем категорию и специальность сотрудников соответствующие максимальным датам присвоения категории по специальности (листинг 4.19).

**Листинг 4.19.** Фрагмент запроса для получения набора данных

ВЫБРАТЬ

МАКСИМУМ(СотрудникиКатегории.ДатаПрисвоения) КАК ДатаПрисвоения,

СотрудникиКатегории.Ссылка КАК Сотрудник,

СотрудникиКатегории.Специальность

ПОМЕСТИТЬ МаксимальныеСотрудникиКатегории

ИЗ

Справочник.Сотрудники.Категории КАК СотрудникиКатегории

СГРУППИРОВАТЬ ПО

СотрудникиКатегории.Ссылка,

СотрудникиКатегории.Специальность

;

ВЫБРАТЬ

СотрудникиКатегории.ДатаПрисвоения КАК ДатаПрисвоения,

СотрудникиКатегории.Категория КАК Категория,

СотрудникиКатегории.Специальность КАК Специальность,

СотрудникиКатегории.Ссылка КАК Сотрудник

ПОМЕСТИТЬ КатегорииСотрудников

ИЗ

Справочник.Сотрудники.Категории КАК СотрудникиКатегории

ВНУТРЕННЕЕ СОЕДИНЕНИЕ МаксимальныеСотрудникиКатегории

ПО СотрудникиКатегории.Ссылка = МаксимальныеСотрудникиКатегории.Сотрудник

И СотрудникиКатегории.ДатаПрисвоения = МаксимальныеСотрудникиКатегории.ДатаПрисвоения

;

Для получения актуальных сведений о категориях сотрудников мы связываем внутренним соединением таблицу *Справочник.Сотрудники.Категории* с временной таблицей *МаксимальныеСотрудникиКатегории* по полям *Ссылка* и *ДатаПрисвоения*. И затем помещаем эти данные во временную таблицу *КатегорииСотрудников*, которую мы будем использовать в итоговой выборке запроса.

Для корректной работы запроса в расширении, нужно не забыть позаимствовать из табличной части *Категории* справочника *Сотрудники* все реквизиты, которые используются в запросе:

* *Специальность*,
* *Категория*,
* *ДатаПрисвоения*.

Одновременно с этими данными в расширение будут заимствованы типы данных, на которые они ссылаются – справочники *Категории* и *Специальности*.

После этого мы получаем в итоговой выборке запроса все данные (из справочников *Сотрудники*, *ШтатноеРасписание* и временных таблиц *ТарификацияСотрудников*, *КатегорииСотрудников*), которые требуются для отчета. Причем, мы объединяем выборку для отображения ФОТ по фактически занятым сотрудниками ставкам (листинг 4.20) и выборку для отображения ФОТ по строкам вакансий штатного расписания (листинг 4.21). Для лучшего понимания рассмотрим эти секции запроса отдельно.

**Листинг 4.20.** Первая секция итогового запроса для получения набора данных

ВЫБРАТЬ

Сотрудники.Наименование + " (" + Сотрудники.ТабельныйНомер + ")" КАК ФИО,

Сотрудники.Отделение КАК Отделение,

Сотрудники.Должность КАК Должность,

Сотрудники.ПКУ КАК ПКУ,

Сотрудники.ОкладПоПКУ КАК ОкладПоПКУ,

Сотрудники.ОбъемРаботы КАК КоличествоСтавок,

Сотрудники.ВидРаботы КАК ВидРаботы,

ТарификацияСотрудников.Ссылка КАК Документ,

ТарификацияСотрудников.ПроцентНадбавкиЗаРуководство КАК ПроцентНадбавкиЗаРуководство,

ТарификацияСотрудников.НадбавкаЗаРуководство КАК НадбавкаЗаРуководство,

ТарификацияСотрудников.ПроцентНадбавкиЗаУченуюСтепень КАК ПроцентНадбавкиЗаУченуюСтепень,

ТарификацияСотрудников.НадбавкаЗаУченуюСтепень КАК НадбавкаЗаУченуюСтепень,

ТарификацияСотрудников.ПроцентНадбавкиЗаПочетноеЗвание КАК ПроцентНадбавкиЗаПочетноеЗвание,

ТарификацияСотрудников.НадбавкаЗаПочетноеЗвание КАК НадбавкаЗаПочетноеЗвание,

ТарификацияСотрудников.ИтогоОклад КАК       ИтогоОклад,

ТарификацияСотрудников.ЗанимаемаяСтавка КАК ЗанятоСтавок,

ТарификацияСотрудников.МесячныйОклад КАК МесячныйОклад,

ТарификацияСотрудников.ПроцентНадбавкиЗаВредность КАК ПроцентНадбавкиЗаВредность,

ТарификацияСотрудников.НадбавкаЗаВредность       КАК НадбавкаЗаВредность,

ТарификацияСотрудников.ПроцентНадбавкиЗаНепрерывныйСтаж КАК ПроцентНадбавкиЗаНепрерывныйСтаж,

ТарификацияСотрудников.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж КАК ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж,

ТарификацияСотрудников.НадбавкаЗаСтаж КАК НадбавкаЗаСтаж,

ТарификацияСотрудников.ФОТ КАК ФОТ,

Сотрудники.Должность.Расш1\_ГруппаУчетаВТарификации КАК ГруппаУчетаВТарификации,

Сотрудники.Должность.КатегорияПерсонала КАК КатегорияПерсонала,

СотрудникиНепрерывныйСтаж.Лет КАК НепрерывныйСтажЛет,

СотрудникиНепрерывныйСтаж.Месяцев КАК НепрерывныйСтажМесяцев,

СотрудникиНепрерывныйСтаж.Дней КАК НепрерывныйСтажДней,

СотрудникиНепрерывныйУчастковыйСтаж.Лет КАК НепрерывныйУчастковыйСтажЛет,

СотрудникиНепрерывныйУчастковыйСтаж.Месяцев КАК НепрерывныйУчастковыйСтажМесяцев,

СотрудникиНепрерывныйУчастковыйСтаж.Дней КАК НепрерывныйУчастковыйСтажДней,

КатегорииСотрудников.Категория КАК Категория,

КатегорииСотрудников.Специальность КАК СпециальностьПоКатегории,

КатегорииСотрудников.ДатаПрисвоения КАК ДатаПрисвоенияКатегории

ИЗ

Справочник.Сотрудники КАК Сотрудники

ЛЕВОЕ СОЕДИНЕНИЕ

Справочник.Сотрудники.Стажи КАК СотрудникиНепрерывныйСтаж

ВНУТРЕННЕЕ СОЕДИНЕНИЕ Справочник.ВидыСтажа КАК ВидыСтажа

ПО СотрудникиНепрерывныйСтаж.ВидСтажа = ВидыСтажа.Ссылка

И (ВидыСтажа.Расш1\_ВидНадбавкиЗаСтаж = ЗНАЧЕНИЕ(Перечисление.Расш1\_ВидыНадбавокЗаСтаж.ПроцентНадбавкиЗаНепрерывныйСтаж))

И (ВидыСтажа.Расш1\_ИспользуетсяВТарификации = ИСТИНА)

ПО Сотрудники.Ссылка = СотрудникиНепрерывныйСтаж.Ссылка

ЛЕВОЕ СОЕДИНЕНИЕ

Справочник.Сотрудники.Стажи КАК СотрудникиНепрерывныйУчастковыйСтаж

ВНУТРЕННЕЕ СОЕДИНЕНИЕ

Справочник.ВидыСтажа КАК ВидыСтажаДляТарификации

ПО СотрудникиНепрерывныйУчастковыйСтаж.ВидСтажа = ВидыСтажаДляТарификации.Ссылка

И (ВидыСтажаДляТарификации.Расш1\_ВидНадбавкиЗаСтаж = ЗНАЧЕНИЕ(Перечисление.Расш1\_ВидыНадбавокЗаСтаж.ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж))

И (ВидыСтажаДляТарификации.Расш1\_ИспользуетсяВТарификации = ИСТИНА)

ПО Сотрудники.Ссылка = СотрудникиНепрерывныйУчастковыйСтаж.Ссылка

ЛЕВОЕ СОЕДИНЕНИЕ ТарификацияСотрудников

ПО Сотрудники.Ссылка = ТарификацияСотрудников.Сотрудник

ЛЕВОЕ СОЕДИНЕНИЕ КатегорииСотрудников

ПО Сотрудники.Ссылка = КатегорииСотрудников.Сотрудник

ГДЕ

Сотрудники.ДатаПриема <= &ДатаТарификации

И (Сотрудники.Уволен = ЛОЖЬ ИЛИ Сотрудники.ДатаУвольнения > &ДатаТарификации)

ОБЪЕДИНИТЬ ВСЕ

…

В первой секции итоговой выборки запроса мы получаем кадровые и тарификационные данные сотрудников, необходимые для отображения ФОТ по фактически занятым ставкам.

При этом таблицу *Справочник.Сотрудники* связываем левым соединением с временной таблицей *ТарификацияСотрудников* по полю *Ссылка* и затем таким же образом связываем таблицу *Справочник.Сотрудники* с временной таблицей *КатегорииСотрудников*.

Отдельно остановимся на получении стажевых данных сотрудников. Дело в том, что в отчете нужно отображать не только проценты и суммы надбавок за стаж из документов по тарификации, но и сам размер стажей (количество лет, месяцев, дней), которые влияют на стажевую надбавку. Размер стажей мы получаем из табличной части *Стажи* справочника *Сотрудники*.

При этом для каждого вида стажа сначала таблицу *Справочник.Сотрудники.Стажи* связываем внутренним соединением с таблицей *Справочник.ВидыСтажа* по полю *ВидСтажа*. Кроме того, из справочника *ВидыСтажа* мы отбираем только те виды стажа, которые используются в тарификации и вид надбавки за стаж которых равен соответствующему значению перечисления *Расш1\_ВидыНадбавокЗаСтаж* (*ПроцентНадбавкиЗаНепрерывныйСтаж* либо *ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж*). После этого таблицу *Справочник.Сотрудники* связываем левым соединением с таблицей *Справочник.Сотрудники.Стажи* по полю *Ссылка*.

В заключение на выборку первой секции запроса мы накладываем условие, чтобы из справочника *Сотрудники* в результат запроса отбирались бы только сотрудники, работающие на дату формирования отчета (параметр схемы компоновки данных *ДатаТарификации*).

Для корректной работы запроса в расширении нужно не забыть позаимствовать из справочника *Сотрудники* не заимствованные ранее реквизиты, которые используются в запросе:

* *ПКУ*,
* *ВидРаботы*.

Кроме того, из справочника *Должности* нужно позаимствовать реквизит *КатегорияПерсонала*, который используется в запросе.

Одновременно с этими данными в расширение будут заимствованы типы данных, на которые они ссылаются: справочник *ПКУ* и перечисления *ВидыРаботы* и *КатегорииПерсонала*.

**Листинг 4.21.** Вторая секция итогового запроса для получения набора данных

…

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ

"ВАКАНСИЯ",

ШтатноеРасписаниеШтаты.Ссылка.Отделение,

ШтатноеРасписаниеШтаты.Должность,

NULL,

ШтатноеРасписаниеШтаты.СреднийОклад,

ЕСТЬNULL(ШтатноеРасписаниеШтаты.КоличествоСтавок, 0) – ЕСТЬNULL(Сотрудники.ОбъемРаботы, 0),

NULL,

NULL,

0,

0,

0,

0,

0,

0,

ШтатноеРасписаниеШтаты.СреднийОклад,

0,

ЕСТЬNULL(ШтатноеРасписаниеШтаты.СреднийОклад, 0) \* (ЕСТЬNULL(ШтатноеРасписаниеШтаты.КоличествоСтавок, 0) – ЕСТЬNULL(Сотрудники.ОбъемРаботы, 0)),

ШтатноеРасписаниеШтаты.ПроцентНадбавкиЗаВредность,

ЕСТЬNULL(ШтатноеРасписаниеШтаты.СреднийОклад, 0) \* (ЕСТЬNULL(ШтатноеРасписаниеШтаты.КоличествоСтавок, 0) – ЕСТЬNULL(Сотрудники.ОбъемРаботы, 0)) \* ЕСТЬNULL(ШтатноеРасписаниеШтаты.ПроцентНадбавкиЗаВредность, 0) / 100,

15,

0,

ЕСТЬNULL(ШтатноеРасписаниеШтаты.СреднийОклад, 0) \* (ЕСТЬNULL(ШтатноеРасписаниеШтаты.КоличествоСтавок, 0) – ЕСТЬNULL(Сотрудники.ОбъемРаботы, 0)) \* 0.15,

ЕСТЬNULL(ШтатноеРасписаниеШтаты.СреднийОклад, 0) \* (ЕСТЬNULL(ШтатноеРасписаниеШтаты.КоличествоСтавок, 0) – ЕСТЬNULL(Сотрудники.ОбъемРаботы, 0)) \* (1 + 0.15 + ЕСТЬNULL(ШтатноеРасписаниеШтаты.ПроцентНадбавкиЗаВредность, 0) / 100),

ШтатноеРасписаниеШтаты.Должность.Расш1\_ГруппаУчетаВТарификации,

ШтатноеРасписаниеШтаты.Должность.КатегорияПерсонала,

0,

0,

0,

0,

0,

0,

NULL,

NULL,

NULL

ИЗ

Справочник.ШтатноеРасписание.Штаты КАК ШтатноеРасписаниеШтаты

ЛЕВОЕ СОЕДИНЕНИЕ (ВЫБРАТЬ

СУММА(Сотрудники.ОбъемРаботы) КАК ОбъемРаботы,

Сотрудники.Должность КАК Должность,

Сотрудники.Отделение КАК Отделение

ИЗ

Справочник.Сотрудники КАК Сотрудники

ГДЕ

Сотрудники.ДатаПриема <= &ДатаТарификации

И (Сотрудники.Уволен = ЛОЖЬ ИЛИ Сотрудники.ДатаУвольнения > &ДатаТарификации)

СГРУППИРОВАТЬ ПО

Сотрудники.Должность,

Сотрудники.Отделение) КАК Сотрудники

ПО ШтатноеРасписаниеШтаты.Должность = Сотрудники.Должность И ШтатноеРасписаниеШтаты.Ссылка.Отделение = Сотрудники.Отделение

ГДЕ

ЕСТЬNULL(ШтатноеРасписаниеШтаты.КоличествоСтавок, 0) – ЕСТЬNULL(Сотрудники.ОбъемРаботы, 0) > 0

Во второй секции итоговой выборки запроса мы получаем данные штатного расписания, необходимые для отображения ФОТ по строке вакансий.

Для расчета количества занятых сотрудниками ставок по каждой должности в каждом отделении мы используем вложенный подзапрос к справочнику *Сотрудники*, в котором суммируется значение реквизита *ОбъемРаботы*, записи справочника группируются по должности и по отделению и в результат подзапроса отбираются только работающие на дату тарификации сотрудники.

При этом таблицу *Справочник.ШтатноеРасписание.Штаты* связываем левым соединением с вложенным подзапросом по полям *Должность* и *Отделение*.

Таким образом, количество вакантных ставок по каждой должности в каждом отделении будет рассчитываться как разница между общим количеством ставок в штатном расписании (реквизит *КоличествоСтавок* табличной части *Штаты* справочника *ШтатноеРасписание*) и суммарным количеством занятых ставок из вложенного подзапроса (реквизит *ОбъемРаботы*справочника *Сотрудники*).

Как мы уже говорили, при расчете строки вакансий в наших отчетах будет использоваться средний должностной оклад, рассчитанный исходя из «вилки окладов» в штатном расписании. Это значение хранится в реквизите *СреднийОклад* табличной части *Штаты* справочника *ШтатноеРасписание*.

Этот средний оклад (реквизит *СреднийОклад*) мы получаем в качестве значений полей выборки *ОкладПоПКУ* и *ИтогоОклад*, определенных в первой секции запроса.

В качестве значения поля выборки *КоличествоСтавок* мы получаем количество вакантных ставок, рассчитанных по описанному выше алгоритму.

В качестве значения поля выборки *МесячныйОклад*мы получаем средний оклад, умноженный на количество вакантных ставок.

В качестве значения поля выборки *ПроцентНадбавкиЗаВредность*мы получаем процент надбавки за вредность, установленный для должности в штатном расписании отделения.

В качестве значения поля выборки *ПроцентНадбавкиЗаНепрерывныйСтаж*задаем константу 15. Это нужно для того, чтобы по вакансиям также рассчитывалась 15-процентная надбавка за стаж, так как неизвестно, с каким стажем в будущем придет сотрудник на вакантную должность.

Суммы надбавок за стаж, за вредность и ФОТ для строки вакансий рассчитываются по аналогии с фактически занятыми ставками в соответствии с алгоритмом, описанным в разделе [«Базовые понятия тарификации и алгоритм расчета ФОТ»](#_Базовые_понятия_тарификации).

В заключение на выборку второй секции запроса мы накладываем условие, чтобы из штатного расписания выбирались только те записи, по которым есть вакансии (количество вакантных ставок больше нуля).

Для корректной работы запроса в расширении нужно не забыть позаимствовать из справочника *ШтатноеРасписание* ранее не заимствованные реквизиты табличной части *Штаты*, которые используются в запросе:

* *КоличествоСтавок*,
* *СреднийОклад*.

Итак, мы описали и пояснили весь запрос для получения набора данных нашего отчета. Все сказанное также относится и ко второму отчету, который мы разработаем в следующем разделе.

Чтобы быстро узнать, какие недостающие объекты, используемые в запросе, нам нужно позаимствовать в расширение, попробуем открыть конструктор запроса в схеме компоновки данных.

Если вы ничего не забыли, конструктор откроет запрос в редакторе, но мы не будем его изменять. Нам нужно было только убедиться в правильности наших действий.

### Вычисляемые поля

Добавим в схему компоновки вычисляемые поля, которые мы будем выводить в отчет. Кратко прокомментируем выражения, по которым они рассчитываются.

Прежде всего добавим вычисляемые поля *СтажВЛПУ* и *УчастковыйСтаж*, с помощью которых размер непрерывного стажа мы представляем понятной для восприятия строкой – «лет.месяцев.дней» (листинги 4.22, 4.23).

**Листинг 4.22.** Выражение для расчета вычисляемого поля «СтажВЛПУ»

Выбор

Когда ЕстьNULL(НепрерывныйСтажДней,0) = 0 Тогда ""

Иначе (

Выбор Когда НепрерывныйСтажЛет < 10 Тогда "0"

Иначе ""

Конец + Строка(НепрерывныйСтажЛет) +"."+

Выбор Когда НепрерывныйСтажМесяцев < 10 Тогда "0"

Иначе ""

Конец + Строка(НепрерывныйСтажМесяцев) +"."+

Выбор Когда НепрерывныйСтажДней < 10 Тогда "0"

Иначе ""

Конец + Строка(НепрерывныйСтажДней) )

Конец

**Листинг 4.23.** Выражение для расчета вычисляемого поля «УчастковыйСтаж»

Выбор

Когда ЕстьNULL(НепрерывныйУчастковыйСтажДней,0) = 0 Тогда ""

Иначе (

Выбор Когда НепрерывныйУчастковыйСтажЛет < 10 Тогда "0"

Иначе ""

Конец + Строка(НепрерывныйУчастковыйСтажЛет) +"."+

Выбор Когда НепрерывныйУчастковыйСтажМесяцев < 10 Тогда "0"

Иначе ""

Конец + Строка(НепрерывныйУчастковыйСтажМесяцев) +"."+

Выбор Когда НепрерывныйУчастковыйСтажДней < 10 Тогда "0"

Иначе ""

Конец + Строка(НепрерывныйУчастковыйСтажДней) )

Конец

Затем добавим вычисляемое поле *ПроцентЗаСтаж*, с помощью которого мы будем отображать тот процент надбавки за стаж, который больше (листинг 4.24).

**Листинг 4.24.** Выражение для расчета вычисляемого поля «ПроцентЗаСтаж»

Выбор

Когда

ПроцентНадбавкиЗаНепрерывныйСтаж > ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж

Тогда ПроцентНадбавкиЗаНепрерывныйСтаж

Иначе

ПроцентНадбавкиЗаНепрерывныйУчастковыйСтаж

Конец

Затем добавим вычисляемые поля, которые нам будут нужны для отображения строк по тарификации фактически занятых сотрудников. Дело в том, что в обоих наших отчетах нам нужны записи как по сотрудникам, так и по вакансиям штатного расписания.

Поэтому мы используем для отчета объединенный запрос, и, соответственно, поля выборки отчета (например, *ФОТ*, *МесячныйОклад*, *НадбавкаЗаВредность* и т. д.) содержат общие суммы и по занятым, и по вакантным ставкам. Чтобы вычленить из них фактические суммы по сотрудникам, мы анализируем значение поля выборки *ЗанятоСтавок*. У вакансий количество занятых ставок равно нулю.

Поэтому мы рассчитываем значение вычисляемых полей *ОкладПоПКУФактический*, *ИтогоОкладФактический*, *МесячныйОкладФактический*, *ЗаВредностьФактически*, *ЗаСтажФактически*, *ФОТФактический* по однотипной формуле – как значение соответствующих полей выборки запроса, исключая строки вакансий (листинг 4.25-4.30).

**Листинг 4.25.** Выражение для расчета вычисляемого поля «ОкладПоПКУФактический»

Выбор

Когда ЗанятоСтавок = 0 Тогда 0

Иначе ОкладПоПКУ

Конец

**Листинг 4.26.** Выражение для расчета вычисляемого поля «ИтогоОкладФактический»

Выбор

Когда ЗанятоСтавок = 0 Тогда 0

Иначе ИтогоОклад

Конец

**Листинг 4.27.** Выражение для расчета вычисляемого поля «МесячныйОкладФактический»

Выбор

Когда ЗанятоСтавок = 0 Тогда 0

Иначе МесячныйОклад

Конец

**Листинг 4.28.** Выражение для расчета вычисляемого поля «ЗаВредностьФактически»

Выбор

Когда ЗанятоСтавок = 0 Тогда 0

Иначе НадбавкаЗаВредность

Конец

**Листинг 4.29.** Выражение для расчета вычисляемого поля «ЗаСтажФактически»

Выбор

Когда ЗанятоСтавок = 0 Тогда 0

Иначе НадбавкаЗаСтаж

Конец

**Листинг 4.30.** Выражение для расчета вычисляемого поля «ФОТФактический»

Выбор

Когда ЗанятоСтавок = 0 Тогда 0

Иначе ФОТ

Конец

Кроме того, для отображения в отчете итогового расчета по категориям персонала нам понадобятся вычисляемые поля *ФОТВакантный* и *ВакантноСтавок*.

Для расчета вакантного ФОТ (и др. вакантных сумм) исключаются строки, в которых количество занятых ставок больше нуля (листинг 4.31).

**Листинг 4.31.** Выражение для расчета вычисляемого поля «ФОТВакантный»

Выбор

Когда ЗанятоСтавок > 0 Тогда 0

Иначе ФОТ

Конец

Для расчета количества вакантных ставок от общего количества ставок (включая вакансии) отнимается количество занятых ставок (листинг 4.32).

**Листинг 4.32.** Выражение для расчета вычисляемого поля «ВакантноСтавок»

КоличествоСтавок – ЗанятоСтавок

Установим формат представления всех вычисляемых полей (за исключением полей *СтажВЛПУ*, *УчастковыйСтаж*, *ПроцентЗаСтаж*) как *ЧДЦ=2*, чтобы они отображались в отчете с двумя знаками после запятой.

### Ресурсы

В ресурсы отчета перенесем все доступные поля. Туда попадут и все созданные нами вычисляемые поля. Для полей ресурсов *ПроцентНадбавкиЗаРуководство*, *ПроцентНадбавкиЗаУченуюСтепень*, *ПроцентНадбавкиЗаПочетноеЗвание*, *ПроцентНадбавкиЗаВредность*, *ПроцентЗаСтаж* в свойстве *Рассчитывать по…* выберем группировку *ФИО*.

Это значит, что значения всех этих полей должны рассчитываться только для сотрудника. Действительно, нет смысла суммировать, например, проценты надбавки за стаж по всем сотрудникам отделения.

### Параметры

У параметра отчета *ДатаТарификации*установим выражение для расчета, чтобы данные по тарификации, введенные за день, попадали в отчет (листинг 4.33).

**Листинг 4.33.** Выражение для расчета значения параметра «ДатаТарификации»

КонецПериода(&ДатаТарификации, "День")

### Поля набора данных

При описании свойств полей набора данных отчета *МесячныйОклад*, *НадбавкаЗаВредность*, *НадбавкаЗаСтаж*, *ФОТ*, *КоличествоСтавок*, *ЗанятоСтавок*установим формат представления этих полей как *ЧДЦ=2*, чтобы они отображались в отчете с двумя знаками после запятой.

### Настройки

Нам нужно, чтобы в отчете отображались списки сотрудников по отделениям с данными по тарификации. И внизу под этим списком располагались бы итоговые суммы ФОТ в разбивке по категориям персонала (врачи, средний медицинский персонал, младший медицинский персонал и т. д.).

В соответствии с этим добавим в структуру отчета группировку *Отделение* и вложим в нее детальные записи. Также добавим группировку *КатегорияПерсонала* и поместим ее на один уровень с группировкой *Отделение* (рис. 4.20).



**Рис. 4.20.**Структура отчета «Расш1\_ТарификацияСотрудниковОтделения»

Теперь на уровень глобального отчета поместим поля:

* *НомерПоПорядку* (системные поля);
* *ФИО* (заголовок «ФИО, Табельный номер»);
* *Должность*;
* *ВидРабот*;
* группа полей «Категория»:
  + *Категория*;
  + *СпециальностьПоКатегории*;
  + *ДатаПрисвоенияКатегории*;
* *ПКУ*.

Эти поля будут отображаться только в детальных записях отчета. Остальные поля ресурсов мы поместим на уровень каждой группировки отдельно.

На уровень группировки *Отделение* поместим поля:

* *ОкладПоПКУФактический*(вычисляемое поле, заголовок «Должностной оклад»).
* Группа полей «За руководство»:
  + *ПроцентНадбавкиЗаРуководство*(заголовок «%»);
  + *НадбавкаЗаРуководство*(заголовок «Сумма»).
* Группа полей «За ученую степень»:
  + *ПроцентНадбавкиЗаУченуюСтепень*(заголовок «%»);
  + *НадбавкаЗаУченуюСтепень*(заголовок «Сумма»).
* Группа полей «За почетное звание»:
  + *ПроцентНадбавкиЗаПочетноеЗвание*(заголовок «%»);
  + *НадбавкаЗаПочетноеЗвание*(заголовок «Сумма»).
* *ИтогоОкладФактический* (вычисляемое поле, заголовок «Итого должностной оклад»).
* *ЗанятоСтавок* (заголовок «Ставка»).
* *МесячныйОкладФактический* (вычисляемое поле, заголовок «Итого месячный должностной оклад»).
* Группа полей «За вредность»:
  + *ПроцентНадбавкиЗаВредность*(заголовок «%»);
  + *ЗаВредностьФактически* (вычисляемое поле, заголовок «Сумма»).
* Группа полей «Непрерывный стаж»:
  + *СтажВЛПУ*(вычисляемое поле, заголовок «Стаж в ЛПУ»);
  + *УчастковыйСтаж* (вычисляемое поле).
* Группа полей «За стаж»:
  + *ПроцентЗаСтаж*(вычисляемое поле, заголовок «%»);
  + *ЗаСтажФактически* (вычисляемое поле, заголовок «Сумма»).
* *ФОТФактический* (вычисляемое поле, заголовок «ФОТ»).

На уровень группировки *КатегорияПерсонала* поместим поля:

* Группа полей «Всего»:
  + *КоличествоСтавок*(заголовок «Ставок»);
  + *ФОТ*.
* Группа полей «Занято»:
  + *ЗанятоСтавок* (заголовок «Ставок»);
  + *ФОТФактический* (вычисляемое поле, заголовок «ФОТ»).
* Группа полей «Вакантно»:
  + *ВакантноСтавок* (вычисляемое поле, заголовок «Ставок»);
  + *ФОТВакантный* (вычисляемое поле, заголовок «ФОТ»).
* Группа полей «Фактически»:
  + *МесячныйОкладФактический*(вычисляемое поле, заголовок «Месячный оклад»);
  + *ЗаВредностьФактически* (вычисляемое поле, заголовок «За вредность»);
  + *ЗаСтажФактически* (вычисляемое поле, заголовок «За стаж»).

Затем установим сортировку, отбор и другие настройки нашего отчета.

На глобальном уровне отчета установим отбор по полю *Отделение*, включим его в состав быстрых пользовательских настроек и отключим пока его использование. Таким образом мы предоставим пользователю возможность включать отбор по отделению непосредственно перед формированием отчета.

Для группировки *Детальные записи* установим еще один отбор – по полю *ЗанятоСтавок* с условием *Больше нуля*. Это условие нужно потому, что мы не хотим видеть в списке сотрудников строки вакансий. Но для расчета фондов в группировке *КатегорияПерсонала* они нужны.

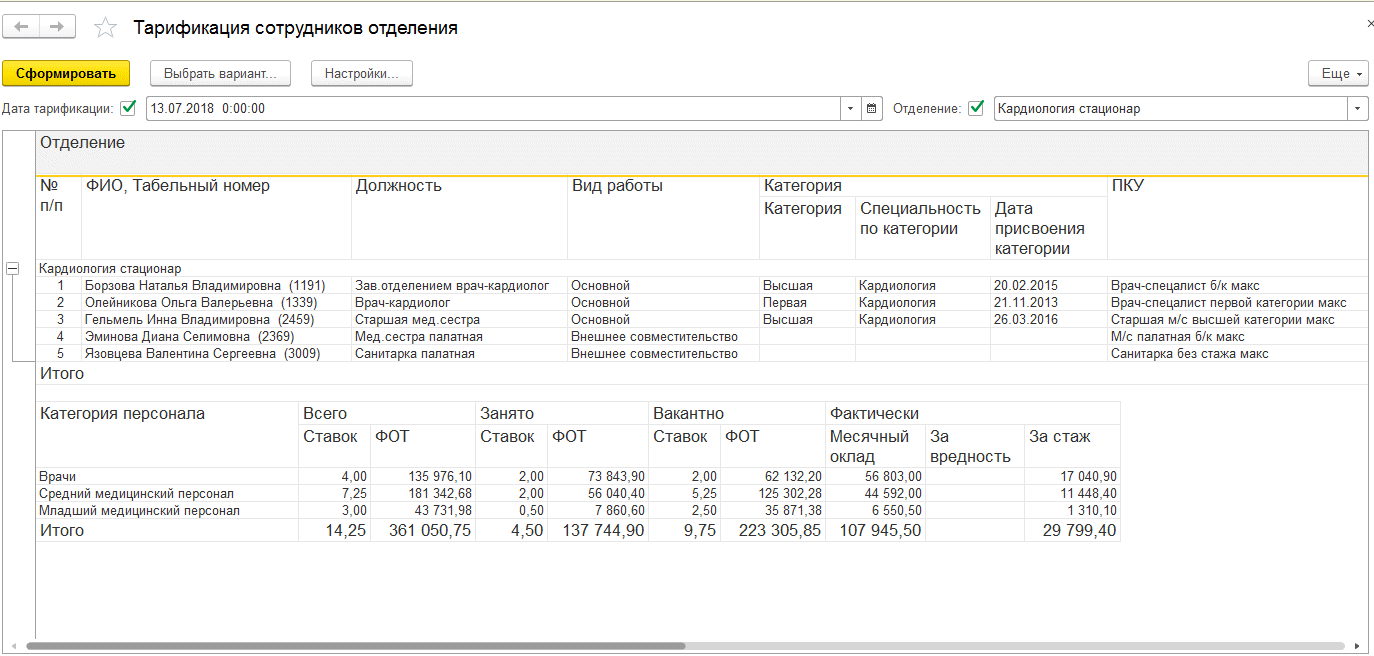
На глобальном уровне отчета установим сортировку по возрастанию полей *КатегорияПерсонала* и *ФИО*. Нам нужно, чтобы сотрудники в списке располагались по алфавиту, но в пределах каждой категории персонала.

Для группировки *Детальные записи* установим параметр *Выводить отбор* в значение *Не выводить*. На глобальном уровне отчета также установим параметры *Выводить отбор* и *Выводить параметры* в значение *Не выводить*.

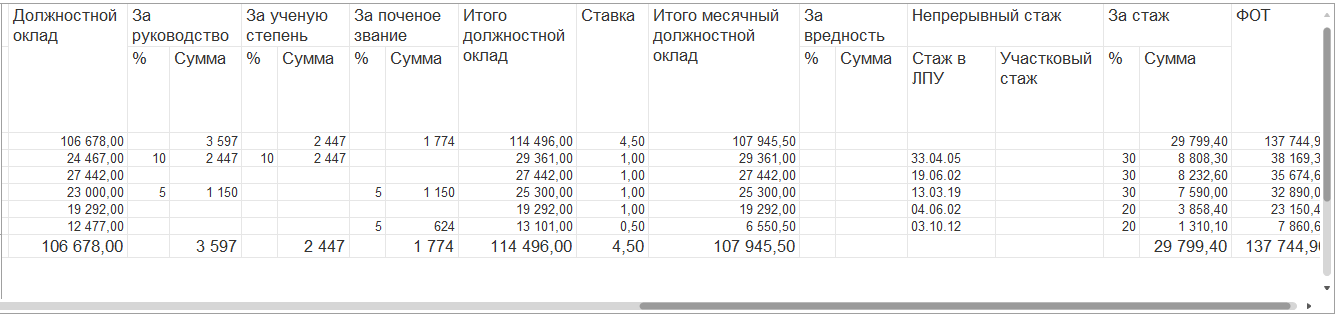
Для параметра *ДатаТарификации* зададим стандартное значение – *Начало этого дня* и включим его в состав быстрых пользовательских настроек.

На этом работа над нашим первым отчетом закончена.

Теперь мы можем вызвать его из раздела *Тарификация – Отчеты* командой *Тарификация сотрудников отделения*. В форме отчета в поле *Отделение* выберем из списка «Кардиология стационар» и сформируем отчет (рис. 4.21, 4.22).



**Рис. 4.21.**Отчет «ТарификацияСотрудниковОтделения» по кардиологическому отделению (часть 1)



**Рис. 4.22.**Отчет «ТарификацияСотрудниковОтделения» по кардиологическому отделению (часть 2)

### Вызов отчета из формы документа «Расш1\_ТарификацияСотрудников»

Конечно, будет гораздо удобней, если пользователь рассчитает тарификацию сотрудников отделения и сразу же из формы документа вызовет соответствующий отчет, в котором будет заранее установлен отбор по отделению, и отчет будет формироваться автоматически без нажатия кнопки *Сформировать*.

Предоставим пользователю такую возможность. Для этого нам нужно еще немного доработать форму документа *Расш1\_ТарификацияСотрудников*.

Добавим в форму документа новую команду *ОтчетПоТарификации*и поместим ее в командную панель формы. Затем создадим обработчик этой команды и заполним его следующим образом (листинг 4.34).

**Листинг 4.34.** Обработчик команды «ОтчетПоТарификации»

&НаКлиенте

Процедура ОтчетПоТарификации(Команда)

Если Модифицированность Тогда

ПоказатьОповещениеПользователя("Перед формированием отчета документ будет записан и проведен.");

ТарификацияПоОтделениюНаСервере();

Модифицированность = Ложь;

КонецЕсли;

ПараметрыФормы = Новый Структура();

ПараметрыФормы.Вставить("СформироватьПриОткрытии", Истина);

ПараметрыФормы.Вставить("ДатаТарификации", Объект.Дата);

ПараметрыФормы.Вставить("Отделение", Объект.Отделение);

ОткрытьФорму("Отчет.Расш1\_ТарификацияСотрудниковОтделения.Форма", ПараметрыФормы);

КонецПроцедуры

Если пользователь внес какие-то изменения в тарификацию сотрудников, то перед формированием отчета мы должны сохранить эти изменения в документе и провести документ. Поэтому, если форма была модифицирована, мы информируем об этом пользователя и вызываем процедуру модуля формы *ТарификацияПоОтделениюНаСервере()* (листинг 4.35).

Затем открываем форму нашего отчета, в параметры формы передаем отделение (для установки отбора в отчете), дату документа (для передачи в параметр отчета) и необходимость формирования отчета сразу при открытии.

**Листинг 4.35.** Процедура «ТарификацияПоОтделениюНаСервере»

&НаСервере

Процедура ТарификацияПоОтделениюНаСервере()

Документ = РеквизитФормыВЗначение("Объект");

Документ.Записать(РежимЗаписиДокумента.Проведение);

КонецПроцедуры

В процедуре *ТарификацияПоОтделениюНаСервере()* мы конвертируем основной реквизит формы в прикладной объект. А затем записываем документ с проведением.

### Форма отчета «Расш1\_ТарификацияСотрудниковОтделения»

В случае если отчет открывается из документа по тарификации, в качестве параметров открытия формы отчета в нее передаются отделение и дата документа. Эти значения мы должны установить в соответствующих полях быстрых пользовательских настроек.

Для этого создадим форму отчета *Расш1\_ТарификацияСотрудниковОтделения* и добавим в нее реквизиты *ДатаОтчета* (тип *Дата*) и *Отделение* (тип *СправочникСсылка.Отделения*).

Затем создадим обработчик события формы *ПриСозданииНаСервере*, в котором, в случае если параметры *ДатаТарификации* и *Отделение* заданы, мы сохраняем значения этих параметров в соответствующих реквизитах формы (листинг 4.36).

**Листинг 4.36.** Обработчик события «ПриСозданииНаСервере»

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

Если Параметры.Свойство("ДатаТарификации") Тогда

ДатаОтчета = Параметры.ДатаТарификации;

Иначе

ДатаОтчета = ТекущаяДата();

КонецЕсли;

Если Параметры.Свойство("Отделение") Тогда

Отделение = Параметры.Отделение;

КонецЕсли;

КонецПроцедуры

Затем создадим обработчик события формы *ПередЗагрузкойПользовательскихНастроекНаСервере* и изменим значения пользовательских настроек перед их загрузкой в компоновщик настроек отчета (листинг 4.37).

**Листинг 4.37.** Обработчик события «ПередЗагрузкойПользовательскихНастроекНаСервере»

&НаСервере

Процедура ПередЗагрузкойПользовательскихНастроекНаСервере(Настройки)

ПараметрДанных = Настройки.Элементы[0];

ПараметрДанных.Значение = ДатаОтчета;

ПараметрДанных.Использование = Истина;

ОтборПоОтделению = Настройки.Элементы[1];

Если НЕ Отделение.Пустая() Тогда

ОтборПоОтделению.ПравоеЗначение = Отделение;

ОтборПоОтделению.Использование = Истина;

КонецЕсли;

КонецПроцедуры

Для простоты мы считаем, что в первом элементе пользовательских настроек содержится параметр данных *ДатаТарификации*, а во втором – элемент отбора по отделению.

В случае если пользовательские настройки еще ни разу не сохранялись, продублируем установку настроек отчета в обработчике события *ПередЗагрузкойВариантаНаСервере* (листинг 4.38).

**Листинг 4.38.** Обработчик события «ПередЗагрузкойВариантаНаСервере»

&НаСервере

Процедура ПередЗагрузкойВариантаНаСервере(Настройки)

Настройки.ПараметрыДанных.УстановитьЗначениеПараметра("ДатаТарификации", ДатаОтчета);

ОтборПоОтделению = Настройки.Отбор.Элементы[0];

Если НЕ Отделение.Пустая() Тогда

ОтборПоОтделению.Использование = Истина;

ОтборПоОтделению.ПравоеЗначение = Отделение;

КонецЕсли;

КонецПроцедуры

## Отчет «Тарификационные списки сотрудников учреждений ФМБА»

Этот отчет предназначен для предоставления тарификационных списков сотрудников в вышестоящие инстанции в разбивке по четырем регламентированным формам. Поэтому он будет содержать четыре варианта настроек. Кроме того, мы разработаем еще пятый вариант отчета, представляющий сводные итоги по тарификационным группам в разбивке по видам финансирования.

Как мы уже говорили, этот отчет очень похож на отчет *Расш1\_ТарификацияСотрудниковОтделения*, описанный в предыдущем разделе [«Отчет "Тарификация сотрудников отделения"»](Тарификация_сотрудников#_Отчет_). Он будет основываться на том же запросе, лишь с небольшими добавлениями, о которых мы расскажем отдельно.

Итак, добавим в расширение отчет *Расш1\_ТарификационныеСпискиФМБА* с синонимом «Тарификационные списки сотрудников учреждений ФМБА» и укажем, что он будет виден в подсистеме *Расш1\_Тарификация*.

Затем создадим схему компоновки данных для отчета. Запрос для получения набора данных схемы компоновки скопируем из отчета *Расш1\_ТарификацияСотрудниковОтделения*. И немного изменим его.

### Запрос

Весь текст запроса для получения набора данных первого отчета был подробно описан и разобран в разделе [«Отчет "Тарификация сотрудников отделения" – Запрос»](#_Запрос). Здесь мы рассмотрим лишь фрагменты, которые мы изменили или добавили в запрос второго отчета (листинг 4.39).

**Листинг 4.39.** Фрагмент итогового запроса для получения набора данных

ВЫБРАТЬ

Сотрудники.Наименование + " " + ВЫБОР

КОГДА Сотрудники.ВидРаботы = ЗНАЧЕНИЕ(Перечисление.ВидыРаботы.Основной)

ТОГДА "(осн.)"

КОГДА Сотрудники.ВидРаботы = ЗНАЧЕНИЕ(Перечисление.ВидыРаботы.ВнешнееСовместительство)

ТОГДА "(внеш. совм.)"

КОГДА Сотрудники.ВидРаботы = ЗНАЧЕНИЕ(Перечисление.ВидыРаботы.ВнутреннееСовместительство)

ТОГДА "(совм.)"

КОНЕЦ КАК ФИО,

Сотрудники.Отделение КАК Отделение,

…

КатегорииСотрудников.ДатаПрисвоения КАК ДатаПрисвоенияКатегории,

Сотрудники.Отделение.Расш1\_ВидФинансирования КАК ВидФинансирования

ИЗ

…

ГДЕ

…

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ

"ВАКАНСИЯ",

…

NULL,

ШтатноеРасписаниеШтаты.Ссылка.Отделение.Расш1\_ВидФинансирования

ИЗ

…

ГДЕ

…

По сравнению с первым отчетом мы получаем здесь вид финансирования отделения, который используется для отображения сводных итогов по тарификационным группам в разбивке по видам финансирования. Вид финансирования хранится в реквизите *Расш1\_ВидФинансирования* (который мы добавили в расширении) справочника *Отделения*.

В первой секции запроса мы получаем вид финансирования из справочника *Сотрудники* через точку от отделения. Кроме того, рядом с ФИО сотрудника в скобках выводим вид работы сотрудника.

Для корректной работы запроса в расширении нужно не забыть позаимствовать значения перечисления *ВидыРаботы*, которые используются в запросе:

* *Основной*,
* *ВнутреннееСовместительство*,
* *ВнешнееСовместительство.*

Во второй секции запроса мы получаем вид финансирования из справочника *ШтатноеРасписание* через точку от отделения.

### Вычисляемые поля и ресурсы

Добавим в схему компоновки вычисляемые поля, которые мы будем выводить в отчет. Этот отчет будет использовать такие же (с некоторыми добавлениями) вычисляемые поля, как и отчет *Расш1\_ТарификацияСотрудниковОтделения*. Эти поля, их назначение и выражения для расчета были подробно описаны в разделе [«Отчет "Тарификация сотрудников по отделению" – Вычисляемые поля»](#_Вычисляемые_поля).

Однако, в отличие от этого отчета, в детальных строках второго отчета нам нужно видеть как строки тарификации сотрудников, так и строки вакансий. Поэтому вычисляемые поля нам понадобятся в основном для сводного варианта отчета.

Наряду с вычисляемыми полями *ЗаВредностьФактически*, *ЗаСтажФактически* и *МесячныйОкладФактический* нам понадобятся поля *ЗаВредностьНаВакансии* (листинг 4.40), *ЗаСтажНаВакансии*(листинг 4.41) и*МесячныйОкладВакантный*(листинг 4.42).

**Листинг 4.40.** Выражение для расчета вычисляемого поля «ЗаВредностьНаВакансии»

Выбор

Когда ЗанятоСтавок > 0 Тогда 0

Иначе НадбавкаЗаВредность

Конец

**Листинг 4.41.** Выражение для расчета вычисляемого поля «ЗаСтажНаВакансии»

Выбор

Когда ЗанятоСтавок > 0 Тогда 0

Иначе НадбавкаЗаСтаж

Конец

**Листинг 4.42.** Выражение для расчета вычисляемого поля «МесячныйОкладВакантный»

Выбор

Когда ЗанятоСтавок > 0 Тогда 0

Иначе МесячныйОклад

Конец

Вычисляемые поля *СтажВЛПУ*, *УчастковыйСтаж*, *ПроцентЗаСтаж ОкладПоПКУФактический*, *ИтогоОкладФактический*, *МесячныйОкладФактический*, *ЗаВредностьФактически*, *ЗаСтажФактически*, *ФОТФактический*, *ФОТВакантный* и *ВакантноСтавок* мы оставляем без изменений по сравнению с предыдущим отчетом (см. листинги 4.22–4.32 в разделе [«Отчет "Тарификация сотрудников по отделению" – Вычисляемые поля»](https://its.1c.ua/db/content/pubextensions/src/103.html?_=1580137101#h107).).

Добавим в схему компоновки еще одно вычисляемое поле для вывода сведений о квалификационной категории сотрудника. В выражении для расчета этого поля категория, дата ее присвоения и специальность по категории, в случае если они заполнены, выводятся в одной строке, через запятую (листинг 4.43).

**Листинг 4.43.** Выражение для расчета вычисляемого поля «ДанныеОКатегории»

Выбор

Когда ДатаПрисвоенияКатегории = ДатаВремя(1, 1, 1) Тогда " "

Иначе Строка(Категория) + " " +

Формат(ДатаПрисвоенияКатегории, "ДЛФ=D") +

Выбор

Когда Строка(СпециальностьПоКатегории) = "" Тогда ""

Иначе " (" + Строка(СпециальностьПоКатегории) + ")"

Конец

Конец

Затем добавим еще одно пустое строковое вычисляемое поле *Дополнительно* для возможности указывать дополнительные сведения о сотруднике. Хотя рассчитывать тут ничего не нужно, соответствующий ресурс отчета нужен, чтобы эта пустая колонка выводилась в отчете крайней справа.

Установим формат представления всех вычисляемых полей (за исключением полей *СтажВЛПУ*, *УчастковыйСтаж*, *ПроцентЗаСтаж*) как *ЧДЦ=2*, чтобы они отображались в отчете с двумя знаками после запятой.

Все сказанное про ресурсы предыдущего отчета также справедливо и здесь. Все новые вычисляемые поля, кроме поля *ДанныеОКатегории*, нужно также добавить в ресурсы отчета.

### Параметры

У параметра отчета *ДатаТарификации*установим выражение для расчета, чтобы данные по тарификации, введенные за день, попадали в отчет (листинг 4.44).

**Листинг 4.44.** Выражение для расчета значения параметра «ДатаТарификации»

КонецПериода(&ДатаТарификации, "День")

### Поля набора данных

При описании свойств полей набора данных отчета *МесячныйОклад*, *НадбавкаЗаВредность*, *НадбавкаЗаСтаж*, *ФОТ*, *КоличествоСтавок*, *ЗанятоСтавок*установим формат представления этих полей как *ЧДЦ=2*, чтобы они отображались в отчете с двумя знаками после запятой.

### Настройки

Чтобы формировать тарификационные списки для различных тарификационных групп сотрудников, относящихся к соответствующим значениям перечисления *Расш1\_ГруппыУчетаВТарификации* (*ДолжностиРуководителей*, *ДолжностиМедицинскогоИФармацевтическогоПерсонала*, *ДолжностиСлужащих*, *ПрофессииРабочих*), разработаем четыре варианта настроек нашего отчета. Плюс еще один сводный вариант.

#### Вариант «Тарификационный список руководителей»

Сначала добавим в настройки отчета вариант, в котором будет формироваться тарификационный список руководителя учреждения, его заместителей, главного бухгалтера и т. п., с именем *ТарификационныйСписокРуководства* и представлением «Тарификационный список руководителей».

В структуру данного варианта отчета добавим группировку *Отделение*, вложим в нее группировку *Должность*, а в нее вложим детальные записи. Группировка по полю *Должность* нам нужна потому, что сначала в отчете должны идти сотрудники, занимающие конкретную должность, а затем – строка вакансий по этой должности (рис. 4.23).



**Рис. 4.23.**Структура отчета «Расш1\_ТарификационныеСпискиФМБА», вариант «Тарификационный список руководителей»

Теперь поместим в отчет поля:

* *ФИО*(заголовок «Фамилия, Имя, Отчество»).
* *ДанныеОКатегории*(вычисляемое поле, заголовок «Наличие квалификационной категории (квалификационная категория, дата присвоения, специальность, по которой присвоена категория)»).
* *ОкладПоПКУ* (заголовок «Должностной оклад»).
* Группа полей «Повышение должностного оклада за наличие ученой степени»:
* *ПроцентНадбавкиЗаУченуюСтепень*(заголовок «%»);
* *НадбавкаЗаУченуюСтепень*(заголовок «Сумма»).
* Группа полей «Повышение должностного оклада за наличие почетного звания»:
  + *ПроцентНадбавкиЗаПочетноеЗвание*(заголовок «%»);
  + *НадбавкаЗаПочетноеЗвание*(заголовок «Сумма»).
* *ИтогоОклад*(заголовок «Итого должностной оклад»).
* Группа полей «Повышение должностного оклада в связи с опасными для здоровья и особо тяжелыми условиями труда»:
  + *ПроцентНадбавкиЗаВредность* (заголовок «%»);
  + *НадбавкаЗаВредность*(заголовок «Сумма»).
* *СтажВЛПУ*(вычисляемое поле, заголовок «Стаж непрерывной работы»).
* Группа полей «Надбавка за продолжительность непрерывной работы»:
  + *ПроцентЗаСтаж* (вычисляемое поле, заголовок «%»);
  + *НадбавкаЗаСтаж*(заголовок «Сумма»).
* *ФОТ* (заголовок «Итого месячный фонд заработной платы»).
* *Дополнительно*(вычисляемое поле).

Затем установим отборы нашего варианта отчета. Добавим отбор по полю *Отделение*, установим для него режим отображения в значение *Обычный* и отключим его использование.

Добавим еще отбор по полю *Должность*, установим для него вид сравнения в значение *В списке*, режим отображения в значение *Обычный* и отключим его использование.

Чтобы в данный вариант отчета отбирались только сотрудники с тарификационной группой *ДолжностиРуководителей*, добавим еще отбор по полю *ГруппаУчетаВТарификации*, установим для него правое значение в соответствующее значение перечисления *Расш1\_ГруппыУчетаВТарификации* и режим отображения в значение *Недоступный*.

Затем установим сортировку варианта отчета по возрастанию поля *ФИО*. Нам нужно, чтобы сотрудники в пределах каждой должности располагались по алфавиту, а в заключение шла бы строка вакансий по этой должности, в которой значение поля *ФИО* равно «ВАКАНСИЯ».

Поэтому мы зададим собственное выражение упорядочивания для поля набора данных *ФИО* (листинг 4.45).

**Листинг 4.45.** Выражение упорядочивания для поля набора данных «ФИО»

Выбор

Когда ФИО = "ВАКАНСИЯ" Тогда 2

Иначе 1

Конец возр

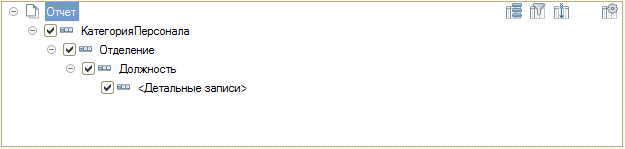
Далее установим параметры *Выводить отбор* и *Выводить параметры* в значение *Не выводить*.

Для параметра *ДатаТарификации* зададим стандартное значение – *Начало этого дня* и включим его в состав быстрых пользовательских настроек.

#### Вариант «Тарификационный список медицинского персонала»

Затем добавим в настройки отчета вариант, в котором будет формироваться тарификационный список работников на должностях медицинского и фармацевтического персонала (в т. ч. врачебный персонал, средний медицинский персонал, младший персонал), с именем *ТарификационныйСписокМедПерсонала*и представлением «Тарификационный список медицинского персонала».

В структуру данного варианта отчета добавим группировку *КатегорияПерсонала*, вложим в нее группировку *Отделение*, вложим в нее группировку *Должность*, а в нее вложим детальные записи. Группировка по полю*КатегорияПерсонала* нам нужна потому, что весь медицинский персонал в отчете должен быть сгруппирован по соответствующим тарификационным категориям, а затем уже разбит по отделениям и должностям. После списка сотрудников, работающих по каждой должности, должна идти строка вакансий по этой должности (рис. 4.24).



**Рис. 4.24.**Структура отчета «Расш1\_ТарификационныеСпискиФМБА», вариант «Тарификационный список медицинского персонала»

Теперь поместим в отчет поля:

* *ФИО*(заголовок «Фамилия, Имя, Отчество»).
* *ДанныеОКатегории*(вычисляемое поле, заголовок «Наличие квалификационной категории (квалификационная категория, дата присвоения, специальность, по которой присвоена категория)»).
* *ОкладПоПКУ* (заголовок «Должностной оклад»).
* Группа полей «Повышение должностного оклада за руководство структурным подразделением»:
  + *ПроцентНадбавкиЗаРуководство*(заголовок «%»);
  + *НадбавкаЗаРуководство*(заголовок «Сумма»).
* Группа полей «Повышение должностного оклада за наличие ученой степени»:
  + *ПроцентНадбавкиЗаУченуюСтепень*(заголовок «%»);
  + *НадбавкаЗаУченуюСтепень*(заголовок «Сумма»).
* Группа полей «Повышение должностного оклада за наличие почетного звания»:
  + *ПроцентНадбавкиЗаПочетноеЗвание*(заголовок «%»);
  + *НадбавкаЗаПочетноеЗвание*(заголовок «Сумма»).
* *ИтогоОклад*(заголовок «Итого должностной оклад»).
* *КоличествоСтавок*(заголовок «Объем работы»).
* *МесячныйОклад* (заголовок «Итого должностной оклад с учетом объема работ»).
* Группа полей «Повышение должностного оклада в связи с опасными для здоровья и особо тяжелыми условиями труда»:
  + *ПроцентНадбавкиЗаВредность* (заголовок «%»);
  + *НадбавкаЗаВредность*(заголовок «Сумма»).
* Группа полей «Стаж непрерывной работы»:
  + *СтажВЛПУ*(вычисляемое поле, заголовок «Стаж в ЛПУ»);
  + *УчастковыйСтаж* (вычисляемое поле).
* Группа полей «Надбавка за продолжительность непрерывной работы»:
  + *ПроцентЗаСтаж* (вычисляемое поле, заголовок «%»);
  + *НадбавкаЗаСтаж*(заголовок «Сумма»).
* *ФОТ* (заголовок «Итого месячный фонд заработной платы»).
* *Дополнительно*(вычисляемое поле).

Затем установим отборы нашего варианта отчета. Добавим отбор по полю *Отделение*, установим для него режим отображения в значение *Обычный* и отключим его использование.

Добавим еще отбор по полю *Должность*, установим для него вид сравнения в значение *В списке*, режим отображения в значение *Обычный* и отключим его использование.

Чтобы в данный вариант отчета отбирались только сотрудники с тарификационной группой *ДолжностиМедицинскогоИФармацевтическогоПерсонала*, добавим еще отбор по полю *ГруппаУчетаВТарификации*, установим для него правое значение в соответствующее значение перечисления *Расш1\_ГруппыУчетаВТарификации* и режим отображения в значение *Недоступный*.

Затем установим сортировку варианта отчета по возрастанию поля *ФИО*. Нам нужно, чтобы сотрудники в пределах каждой должности располагались по алфавиту, а в заключении шла бы строка вакансий по этой должности, в которой значение поля *ФИО* равно «ВАКАНСИЯ».

Поэтому мы используем собственное выражение упорядочивания для поля набора данных *ФИО*, которое мы задавали при описании настроек варианта отчета *ТарификационныйСписокРуководства*(см. листинг 4.45).

Далее установим параметры *Выводить отбор* и *Выводить параметры* в значение *Не выводить*.

Для параметра *ДатаТарификации* зададим стандартное значение – *Начало этого дня* и включим его в состав быстрых пользовательских настроек.

#### Вариант «Тарификационный список служащих»

Затем добавим в настройки отчета вариант, в котором будет формироваться тарификационный список работников, занимающих должности служащих, в т. ч. руководителей структурных подразделений (не врачей) и их заместителей и др., с именем *ТарификационныйСписокСлужащих* и представлением «Тарификационный список служащих».

В структуру данного варианта отчета добавим группировку *Отделение*, вложим в нее группировку *Должность*, а в нее вложим детальные записи (см. рис. 4.23).

Теперь поместим в отчет поля:

* *ФИО*(заголовок «Фамилия, Имя, Отчество»).
* *ОкладПоПКУ* (заголовок «Должностной оклад»).
* Группа полей «Повышение должностного оклада за наличие почетного звания»:
  + *ПроцентНадбавкиЗаПочетноеЗвание*(заголовок «%»);
  + *НадбавкаЗаПочетноеЗвание*(заголовок «Сумма»).
* *ИтогоОклад*(заголовок «Итого должностной оклад»).
* *КоличествоСтавок*(заголовок «Объем работы»).
* *МесячныйОклад* (заголовок «Итого должностной оклад с учетом объема работ»).
* Группа полей «Повышение должностного оклада в связи с опасными для здоровья и особо тяжелыми условиями труда»:
  + *ПроцентНадбавкиЗаВредность* (заголовок «%»);
  + *НадбавкаЗаВредность*(заголовок «Сумма»).
* *СтажВЛПУ*(вычисляемое поле, заголовок «Стаж непрерывной работы»).
* Группа полей «Надбавка за продолжительность непрерывной работы»:
  + *ПроцентЗаСтаж* (вычисляемое поле, заголовок «%»);
  + *НадбавкаЗаСтаж*(заголовок «Сумма»).
* *ФОТ* (заголовок «Итого месячный фонд заработной платы»).
* *Дополнительно*(вычисляемое поле).

Затем установим отборы нашего варианта отчета. Добавим отбор по полю *Отделение*, установим для него режим отображения в значение *Обычный* и отключим его использование.

Добавим еще отбор по полю *Должность*, установим для него вид сравнения в значение *В списке*, режим отображения в значение *Обычный* и отключим его использование.

Чтобы в данный вариант отчета отбирались только сотрудники с тарификационной группой *ДолжностиСлужащих*, добавим еще отбор по полю *ГруппаУчетаВТарификации*, установим для него правое значение в соответствующее значение перечисления *Расш1\_ГруппыУчетаВТарификации* и режим отображения в значение *Недоступный*.

Затем установим сортировку варианта отчета по возрастанию поля *ФИО*. Нам нужно, чтобы сотрудники в пределах каждой должности располагались по алфавиту, а в заключении шла бы строка вакансий по этой должности, в которой значение поля *ФИО* равно «ВАКАНСИЯ».

Поэтому мы используем собственное выражение упорядочивания для поля набора данных *ФИО*, которое мы задавали при описании настроек варианта отчета *ТарификационныйСписокРуководства*(см. листинг 4.45).

Далее установим параметры *Выводить отбор* и *Выводить параметры* в значение *Не выводить*.

Для параметра *ДатаТарификации* зададим стандартное значение – *Начало этого дня* и включим его в состав быстрых пользовательских настроек.

#### Вариант «Тарификационный список рабочих»

Затем добавим в настройки отчета вариант, в котором будет формироваться тарификационный список работников по профессиям рабочих, с именем *ТарификационныйСписокРабочих* и представлением «Тарификационный список рабочих».

В структуру данного варианта отчета добавим группировку *Отделение*, вложим в нее группировку *Должность*, а в нее вложим детальные записи (см. рис. 4.23).

Теперь поместим в отчет поля:

* *ФИО*(заголовок «Фамилия, Имя, Отчество»).
* *ИтогоОклад*(заголовок «Тарифная ставка»).
* *КоличествоСтавок*(заголовок «Объем работы»).
* *МесячныйОклад* (заголовок «Итого тарифная ставка с учетом объема работ»).
* Группа полей «Повышение должностного оклада в связи с опасными для здоровья и особо тяжелыми условиями труда»:
  + *ПроцентНадбавкиЗаВредность* (заголовок «%»);
  + *НадбавкаЗаВредность*(заголовок «Сумма»).
* *СтажВЛПУ*(вычисляемое поле, заголовок «Стаж непрерывной работы»).
* Группа полей «Надбавка за продолжительность непрерывной работы»:
  + *ПроцентЗаСтаж* (вычисляемое поле, заголовок «%»);
  + *НадбавкаЗаСтаж*(заголовок «Сумма»).
* *ФОТ* (заголовок «Итого месячный фонд заработной платы»).
* *Дополнительно*(вычисляемое поле).

Затем установим отборы нашего варианта отчета. Добавим отбор по полю *Отделение*, установим для него режим отображения в значение *Обычный* и отключим его использование.

Добавим еще отбор по полю *Должность*, установим для него вид сравнения в значение *В списке*, режим отображения в значение *Обычный* и отключим его использование.

Чтобы в данный вариант отчета отбирались только сотрудники с тарификационной группой *ПрофессииРабочих*, добавим еще отбор по полю *ГруппаУчетаВТарификации*, установим для него правое значение в соответствующее значение перечисления *Расш1\_ГруппыУчетаВТарификации* и режим отображения в значение *Недоступный*.

Затем установим сортировку варианта отчета по возрастанию поля *ФИО*. Нам нужно, чтобы сотрудники в пределах каждой должности располагались по алфавиту, а в заключении шла бы строка вакансий по этой должности, в которой значение поля *ФИО* равно «ВАКАНСИЯ».

Поэтому мы используем собственное выражение упорядочивания для поля набора данных *ФИО*, которое мы задавали при описании настроек варианта отчета *ТарификационныйСписокРуководства*(см. листинг 4.45).

Далее установим параметры *Выводить отбор* и *Выводить параметры* в значение *Не выводить*.

Для параметра *ДатаТарификации* зададим стандартное значение – *Начало этого дня* и включим его в состав быстрых пользовательских настроек.

#### Вариант отчета «Сводный тарификационный список»

Затем добавим в настройки отчета вариант, в котором будут рассчитываться и выводиться сводные итоги по видам финансирования в разрезе тарификационных категорий сотрудников, с именем *ТарификационныйСписокСводный* и представлением «Сводный тарификационный список».

В структуру данного варианта отчета добавим группировку *ВидФинансирования* и вложим в нее группировку *КатегорияПерсонала* (рис. 4.25).



**Рис. 4.25.**Структура отчета «Расш1\_ТарификационныеСпискиФМБА», вариант «Сводный тарификационный список»

Теперь поместим в отчет поля:

* Группа полей «Число штатных должностей»:
  + *КоличествоСтавок*(заголовок «Всего ставок»;.
  + *ЗанятоСтавок*;
  + *ВакантноСтавок* (вычисляемое поле).
* Группа полей «Месячный должностной оклад»:
  + *МесячныйОклад* (заголовок «Месячный оклад всего»);
  + *МесячныйОкладФактический* (вычисляемое поле);
  + *МесячныйОкладВакантный* (вычисляемое поле).
* Группа полей «Сумма надбавки за опасные условия труда»:
  + *НадбавкаЗаВредность*(заголовок «Надбавка за вредность всего»);
  + *ЗаВредностьФактически*(вычисляемое поле, заголовок «Надбавка за вредность фактически»);
  + *ЗаВредностьНаВакансии* (вычисляемое поле, заголовок «Надбавка за вредность на вакансии»).
* Группа полей «Сумма надбавки за непрерывный стаж работы»:
  + *НадбавкаЗаСтаж*(заголовок «Надбавка за стаж всего»);
  + *ЗаСтажФактически*(вычисляемое поле, заголовок «Надбавка за стаж фактически»);
  + *ЗаСтажНаВакансии* (вычисляемое поле, заголовок «Надбавка за стаж на вакансии»).
* Группа полей «Итого месячный фонд заработной платы»:
  + *ФОТ* (заголовок «ФОТ всего»).;
  + *ФОТФактический* (вычисляемое поле, заголовок «ФОТ фактический»);
  + *ФОТВакантный* (вычисляемое поле, заголовок «ФОТ вакантный»).

Затем установим отборы нашего варианта отчета. Добавим отбор по полю *Отделение*, установим для него режим отображения в значение *Обычный* и отключим его использование.

Добавим еще отбор по полю *Должность*, установим для него вид сравнения в значение *В списке*, режим отображения в значение *Обычный* и отключим его использование.

Добавим еще отбор по полю *ГруппаУчетаВТарификации*, установим для него вид сравнения в значение *Не равно*, правое значение как пустую ссылку на перечисление *Расш1\_ГруппыУчетаВТарификации* и режим отображения в значение *Недоступный*.

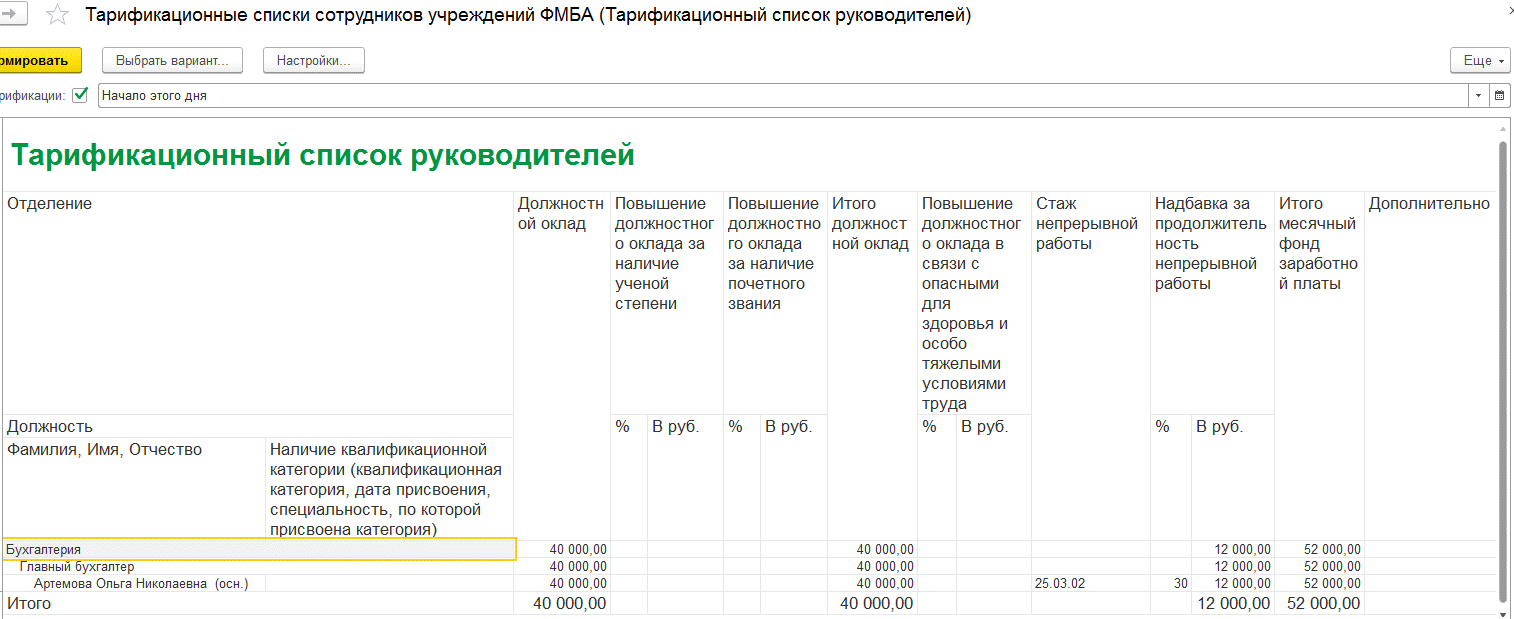
Далее установим параметры *Выводить отбор* и *Выводить параметры* в значение *Не выводить*.

Для параметра *ДатаТарификации* зададим стандартное значение – *Начало этого дня* и включим его в состав быстрых пользовательских настроек.

На этом наш второй отчет закончен.

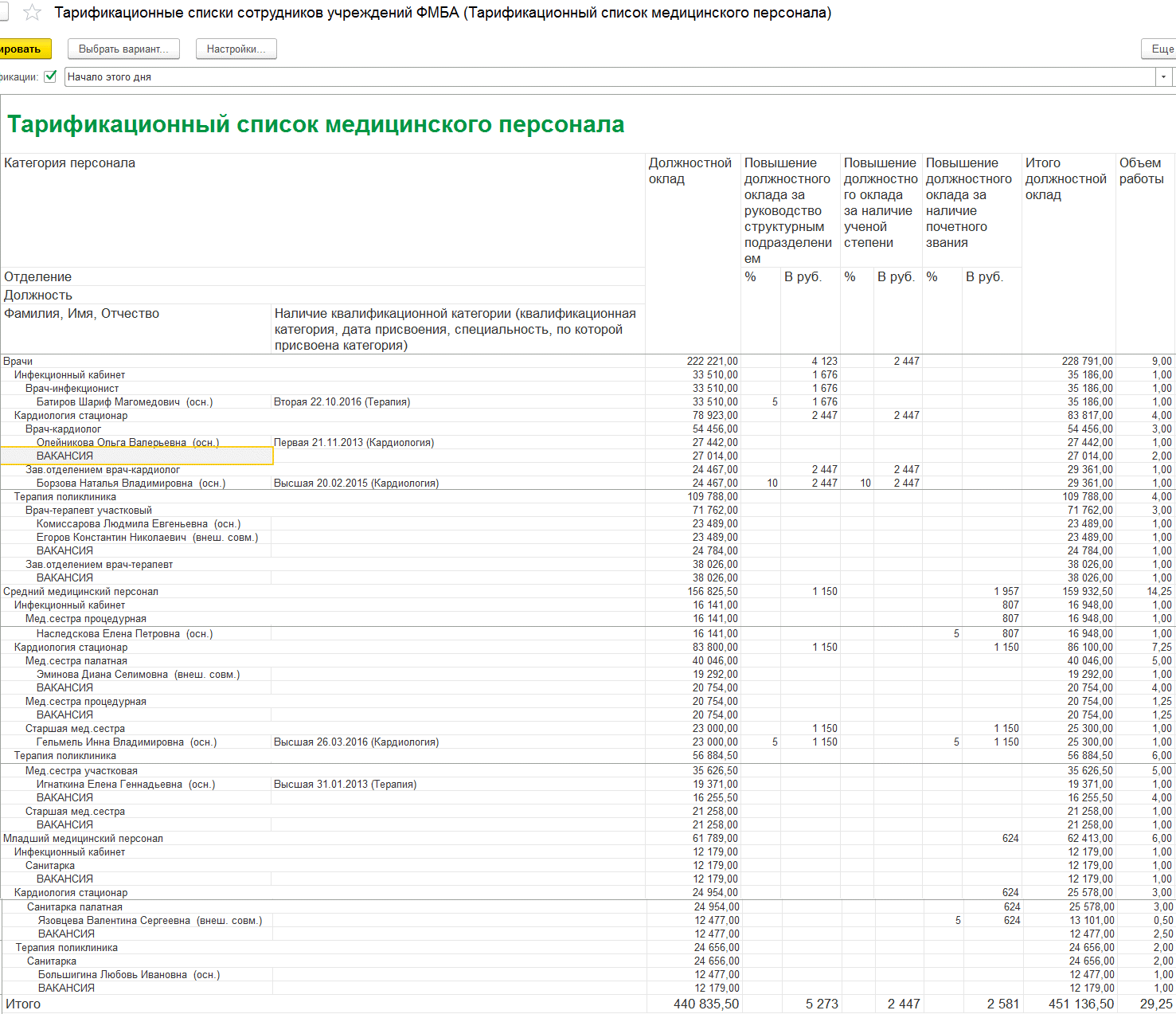
Теперь мы можем вызвать его из раздела *Тарификация – Отчеты* командой *Тарификационные списки сотрудников учреждений ФМБА*.

Тарификационный список руководителей будет выглядеть следующим образом (рис. 4.26).

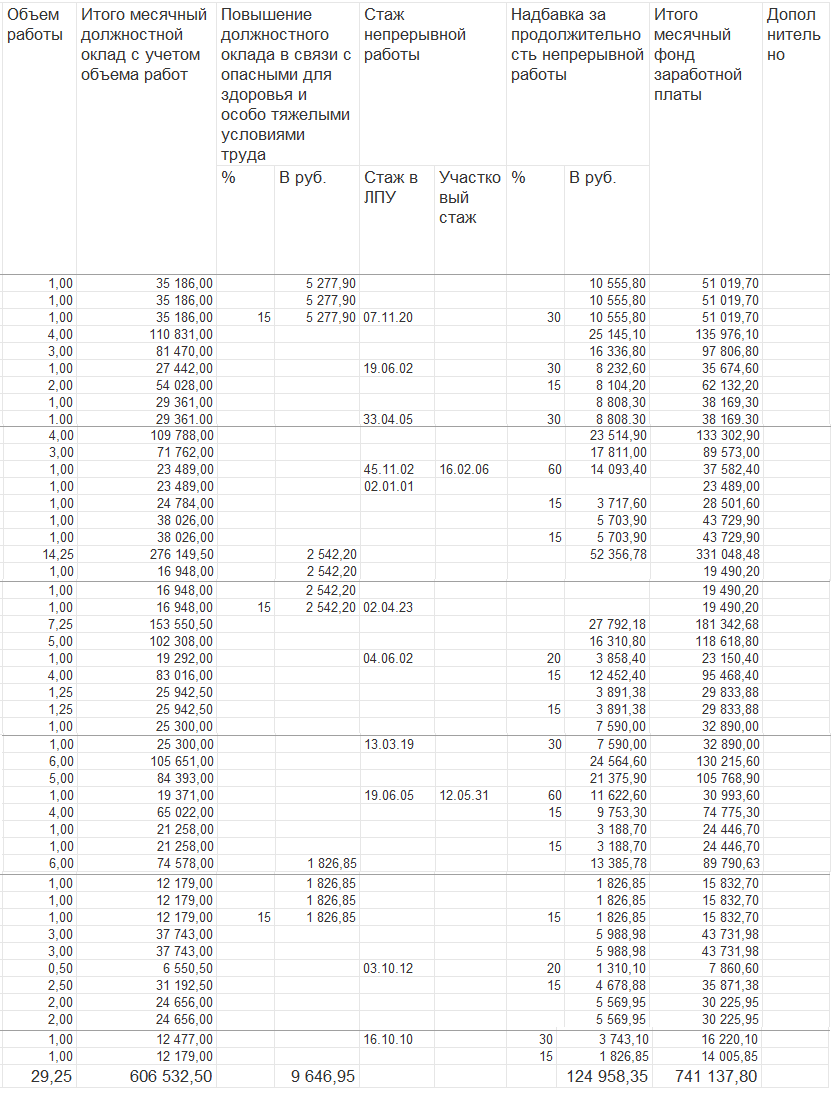


**Рис. 4.26.**Вариант отчета «Тарификационный список руководителей»

Тарификационный список медицинского персонала будет выглядеть следующим образом (рис. 4.27, 4.28).

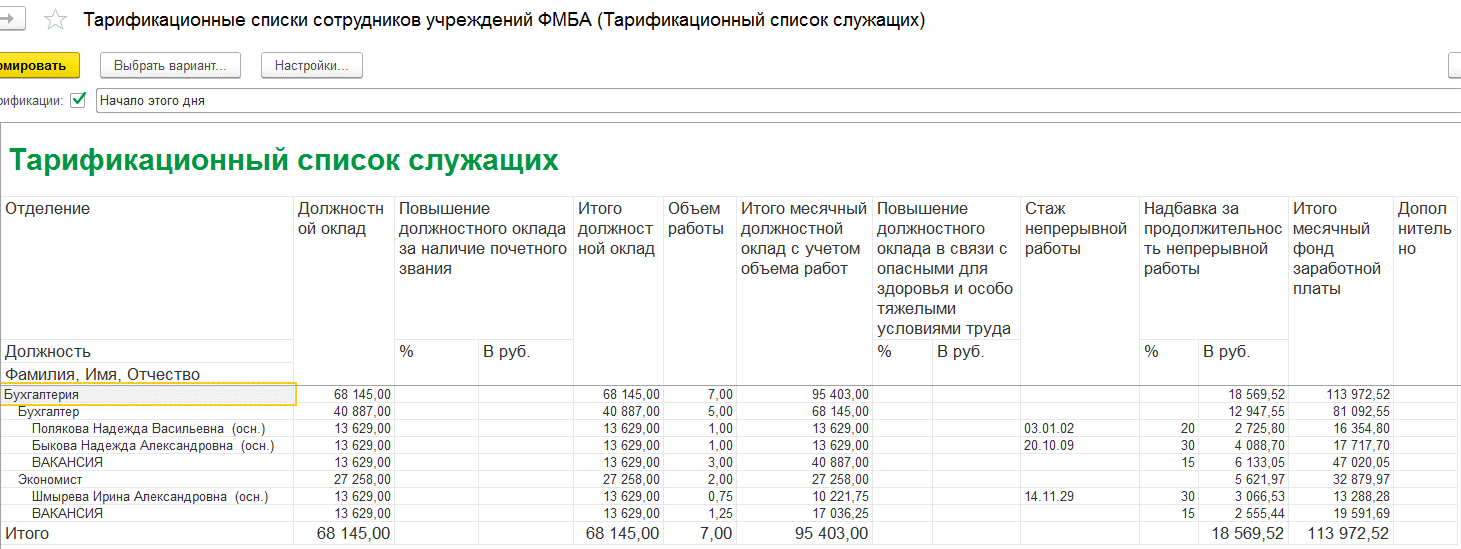


**Рис. 4.27.**Вариант отчета «Тарификационный список медперсонала» (часть 1)



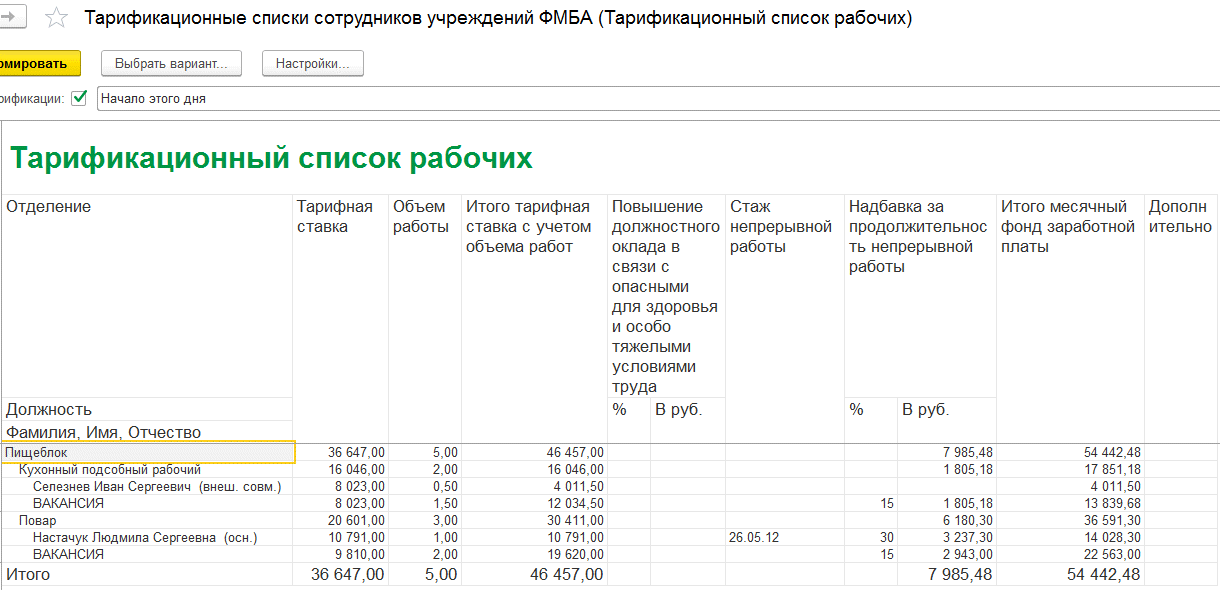
**Рис. 4.28.**Вариант отчета «Тарификационный список медперсонала» (часть 2)

Тарификационный список служащих будет выглядеть следующим образом (рис. 4.29).



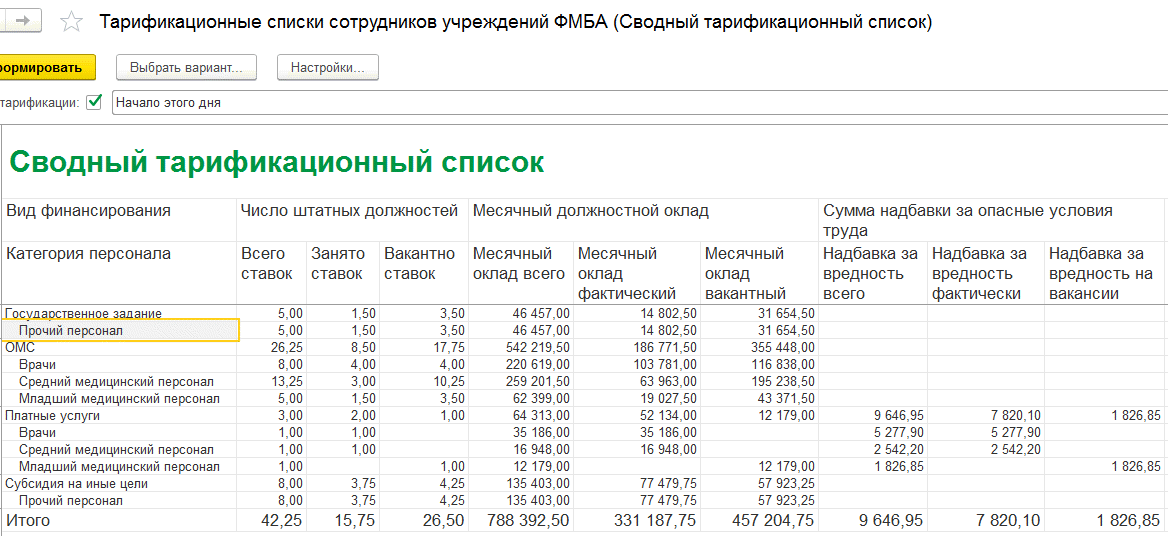
**Рис. 4.29.**Вариант отчета «Тарификационный список служащих»

Тарификационный список рабочих будет выглядеть следующим образом (рис. 4.30).

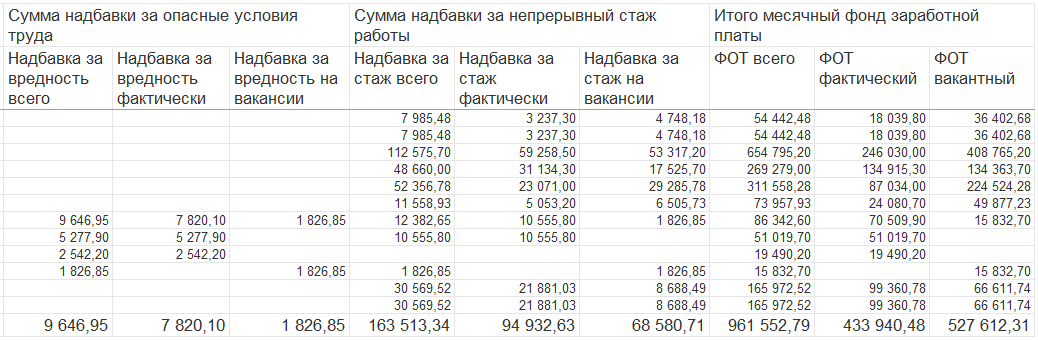


**Рис. 4.30.**Вариант отчета «Тарификационный список рабочих»

Сводный тарификационный список будет выглядеть следующим образом (рис. 4.31, 4.32).



**Рис. 4.31.**Вариант отчета «Сводный тарификационный список» (часть 1)



**Рис. 4.32.**Вариант отчета «Сводный тарификационный список» (часть 2)

## Доработка расширения для похожих внедрений

Довольно часто бывают ситуации, когда в целом результат работы расширения заказчика устраивает, но хотелось бы что-то изменить.

Если речь идет о доработке отчетов, форм или других объектов расширяемой конфигурации и эти доработки не расширяют данные, то это решается довольно просто. Примеры таких доработок рассмотрены во второй главе.

Но могут понадобиться и более серьезные изменения, например в самом алгоритме расчета ФОТ, описанном нами выше в этой главе.

Напомним, что для выполнения тарификации сотрудников мы добавили в расширение собственный документ, в табличной части которого напрямую содержались все необходимые колонки, с помощью которых реализовался алгоритм расчета ФОТ.

Если состав тарификационных надбавок в организации немного другой – например, для расчета ФОТ к базовому окладу сотрудника прибавляется еще и надбавка за категорию, то это потребует уже изменения структуры данных самого документа (добавления соответствующей колонки в табличную часть документа).

Однако сегодняшний уровень развития механизма расширений позволяет расширить практически любые данные. А изменить что-то в структуре данных собственного документа – вообще не проблема.

Остается подкорректировать процедуру формы документа, в которой прописан алгоритм расчета ФОТ, потом добавить новую колонку в отчет – и задача решена.

Или, например, в организации используется какое-то другое деление на группы учета сотрудников в тарификации. Поскольку у нас эти группы заданы в собственном перечислении, то названия и состав значений этого перечисления можно изменять так, как требуется. Или же, чтобы предоставить большую свободу пользователям, можно вообще использовать в расширении вместо этого перечисления собственный справочник.

Если же в организации по-другому рассчитывается ФОТ по вакантным позициям штатного расписания, то для этого достаточно просто изменить текст запроса отчетов, в которых рассчитывается ФОТ вакансий. Например, можно использовать в качестве базового оклада для расчета ФОТ вакансий не усредненный (как в нашем примере), а минимальный или максимальный оклад по должности штатного расписания.

Если требуется вести историю изменений кадровых данных сотрудников, то можно организовать хранение кадровых данных в периодических регистрах сведений. То есть можно добавить в расширение собственный регистр сведений и при проведении существующих в кадровой конфигурации документов формировать в нем движения.

В целом доработка описанного расширения не представляет больших трудностей. Даже в случае довольно серьезных изменений возможность расширения данных позволяет легко реализовать самые разные пожелания заказчика.