

## Практическая работа № 18

### *Поиск в базе данных (1:30)*

В этой работе мы познакомимся с механизмом полнотекстового поиска в данных и разработаете отчет, который позволит выполнять поиск в БД на основе полнотекстового индекса. На примере этого отчета вы научитесь разрабатывать форму «с нуля», наполняя ее реквизитами и командами.

В большой информационной базе просто необходим поиск. Поэтому система содержит механизм полнотекстового поиска в данных. Преимущества этого механизма в том, что он позволяет искать данные, вводя поисковый запрос в простой и естественной форме, например: «телефон абдулова». При этом можно использовать специальные операторы, вроде тех, что применяются при поиске в интернете.

Полнотекстовый поиск очень удобен, когда мы не знаем точно, где находятся нужные данные (например, в каком справочнике), и не знаем точно, что нужно искать (не помним точное название номенклатуры).

Кроме этих возможностей, он позволяет находить данные там, где другие методы поиска крайне трудоемки или требуют создания специальных алгоритмов и обработок.

#### **Общие сведения о механизме полнотекстового поиска в данных**

Механизм полнотекстового поиска 1С:Предприятия 8 основан на использовании двух составляющих:

- Полнотекстового индекса,
- Средств выполнения полнотекстового поиска

Для выполнения полнотекстового поиска обязательно должен существовать полнотекстовый индекс. Он создается один раз и затем должен периодически обновляться.

Поиск осуществляется по данным, которые содержатся в полнотекстовом индексе. Т.о., если ведется интенсивная работа с БД, то полнотекстовый индекс следует обновлять как можно чаще. Если же объем изменяемых или новых данных невелик, то обновление полнотекстового индекса

можно выполнять реже, например раз в сутки, в период наименьшей загрузки системы.

Создание и обновление индекса может выполняться как интерактивно, в режиме 1С:Предприятие, так и программно, средствами встроенного языка. В этой работе мы рассмотрим возможности интерактивного индексирования, а в следующей – как можно обновлять полнотекстовый индекс в автоматическом режиме.

В процессе работы информационной базы система отслеживает факт изменения данных в тех объектах конфигурации, которые могут участвовать в полнотекстовом поиске (планы счетов, планы видов расчета, планы обмена, справочники, документы, планы видов характеристик, регистры сведений, накопления, бухгалтерии, расчета).

Впоследствии при создании или обновлении полнотекстового индекса система анализирует данные, содержащиеся в реквизитах этих объектов, и включает эти данные в полнотекстовый индекс. При этом анализироваться могут не все реквизиты, а только которые имеют тип **Строка, Число, Дата, ХранилищеЗначения** или ссылочный тип.

Собственно сам полнотекстовый поиск выполняется средствами встроенного языка и в соответствии с правами пользователя. Если какая-то информация недоступна данному пользователю, он не сможет получить ее и при помощи полнотекстового поиска.

Приступим к первой части необходимых действий – созданию полнотекстового индекса. Второй частью будет создание отчета, который будет собственно выполнять полнотекстовый поиск, используя созданный нами индекс.

## **Полнотекстовый индекс**

### **В режиме Конфигуратор**

Каждый объект конфигурации, данные которого могут участвовать в полнотекстовом индексировании, имеет свойство **ПолнотекстовыйПоиск**. По умолчанию при создании нового объекта это свойство установлено в значение **Использовать**.

Кроме объектов конфигурации свойство ПолнотекстовыйПоиск существует и у реквизитов этих объектов. Т.о. мы имеем возможность указывать конкретные реквизиты, данные которых должны участвовать в полнотекстовом индексировании. По умолчанию оно также включено.

Для знакомства откройте свойства справочника **Номенклатура**. Найдите свойство **Полнотекстовый поиск** и убедитесь, что оно включено.

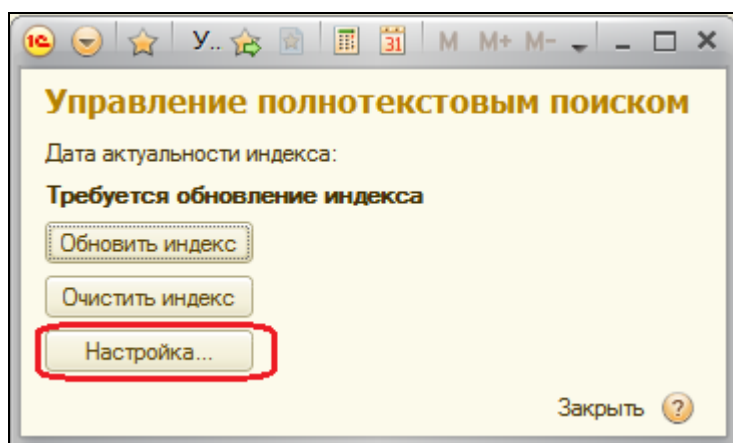
Теперь откройте свойства реквизита **ВидНоменклатуры** справочника **Номенклатура** и также убедитесь, что свойство включено.

Т.о. по умолчанию в нашей конфигурации полнотекстовый индекс используется для всех возможных реквизитов всех объектов конфигурации.

Перейдем в режим 1С:Предприятие.

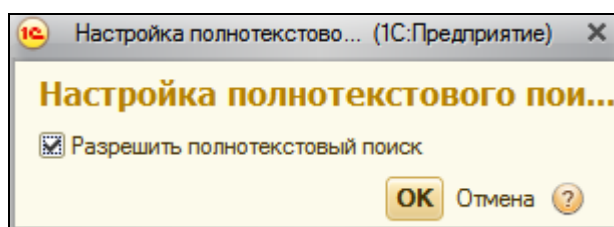
В режиме 1С:Предприятие

Выполним команду меню **Все функции – Стандартные – Управление полнотекстовым поиском**.



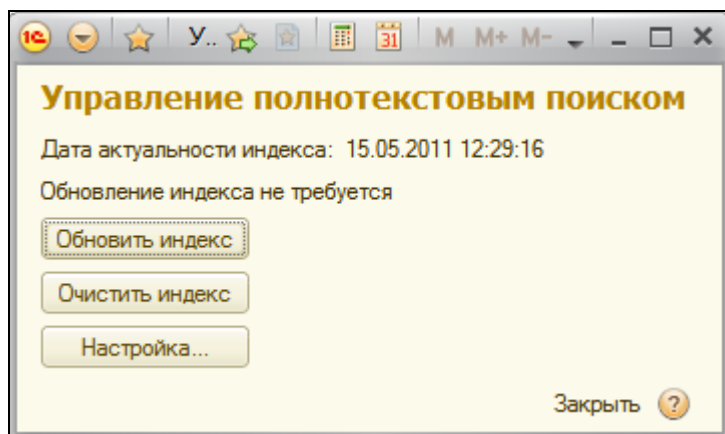
Это окно позволяет создавать и обновлять полнотекстовый индекс **интерактивно**. Кроме этого, позволяет разрешать или запрещать вообще все операции, связанные с полнотекстовым поиском: обновление, очистка полнотекстового индекса, полнотекстовый поиск.

Чтобы узнать, разрешены ли сейчас операции полнотекстового поиска, нажмите кнопку **Настройка...** Система откроет окно настройки полнотекстового поиска.



По умолчанию полнотекстовый поиск разрешен. Закройте это окно.

Система сообщает нам, что требуется обновление полнотекстового индекса, потому что в нашем случае индекс вообще отсутствует. Для создания или обновления индекса, нажмите кнопку **Обновить индекс**.



При больших размерах информационной базы создание и обновление полнотекстового индекса может занимать несколько минут.

Мы создали полнотекстовый индекс для нашей информационной базы.

Теперь перейдем к созданию отчета, который позволит выполнять полнотекстовый поиск в базе данных.

### Отчет для поиска данных

#### В режиме Конфигуратор

Добавим в конфигурацию новый отчет с именем **ПоискДанных**.

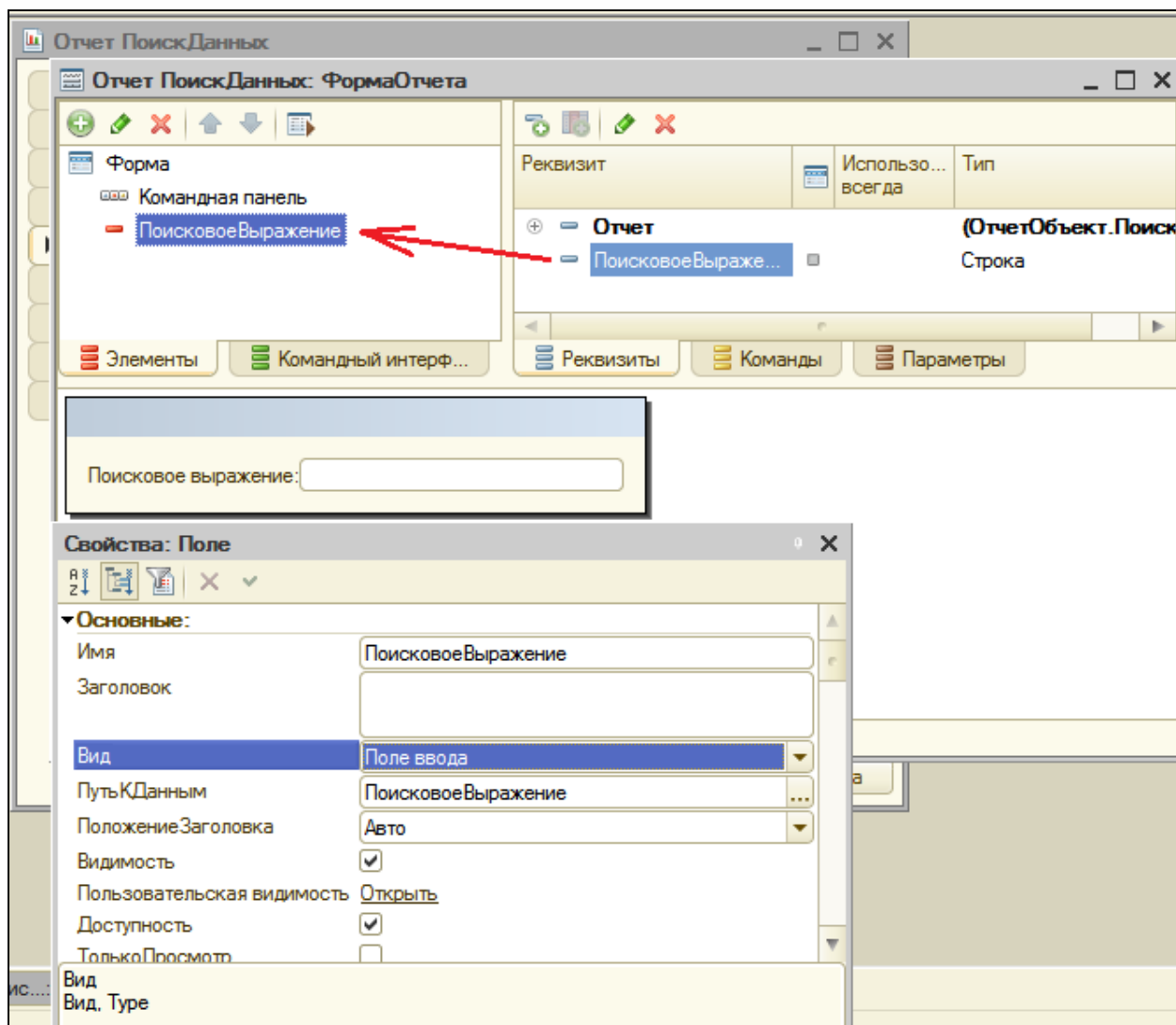
Перейдем на закладку **Формы**, нажмем кнопку открытия, создадим основную форму отчета и займемся ее редактированием.

Как мы уже говорили, элементы формы обязательно должны быть связаны с данными, иначе они не будут отображены. Поэтому сначала создадим соответствующие реквизиты и команды формы и затем перетащим их в окно элементов форм.

На закладке **Реквизиты** создадим реквизит **ПоисковоеВыражение** и перетащим его в окно элементов формы.

В открывшемся окне свойств поля **ПоисковоеВыражение** зададим его заголовок **Фраза**.

В поле **Вид** автоматически подставилось значение **Поле ввода** – это нам и нужно.

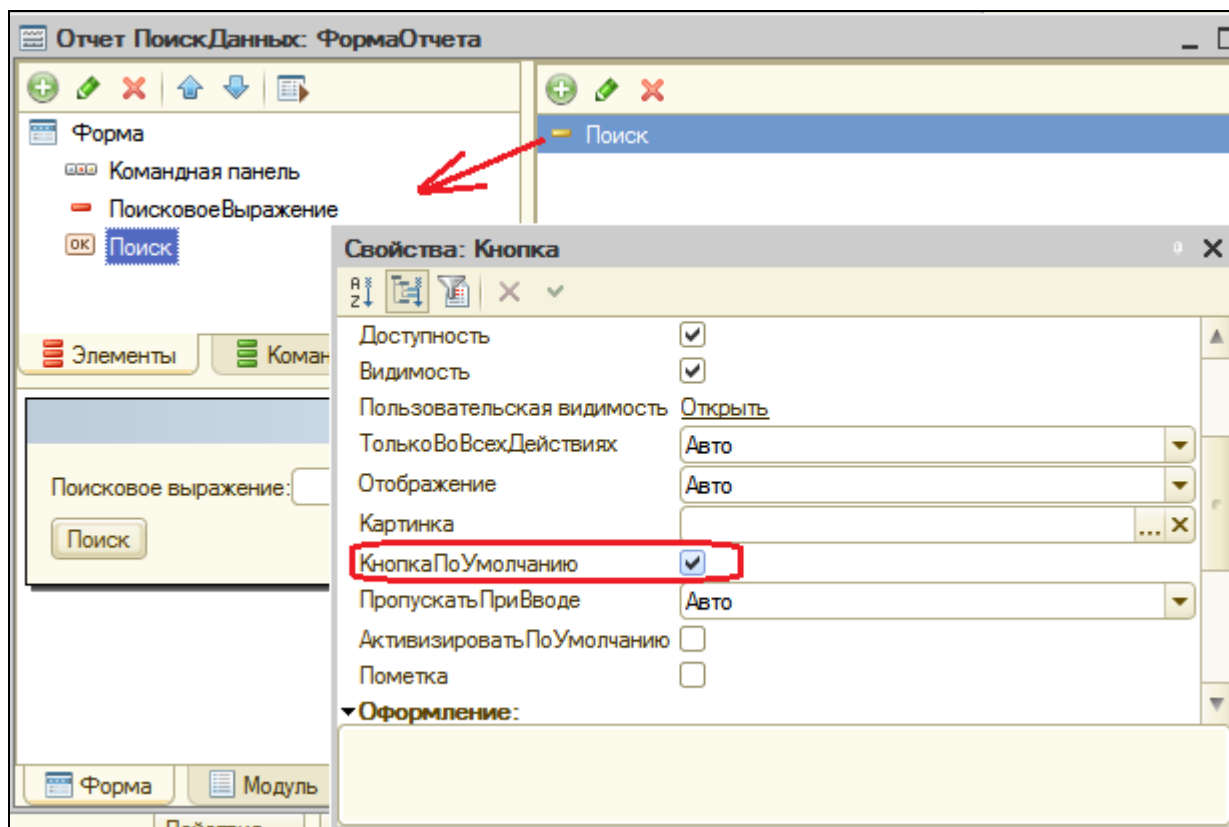


В поле ввода **ПоисковоеВыражение** мы будем вводить фразу для поиска в базе данных.

Затем на закладке **Команды** создадим команду **Поиск** и нажмем кнопку открытия в строке **Действие**.

Шаблон обработчика команды, открывшийся в модуле формы, пока заполнять не будем, а перейдем на закладку **Форма** и перетащим эту команду в окно элементов формы.

В открывшемся окне свойств кнопки **Поиск** поставим флажок **КнопкаПоУмолчанию**.

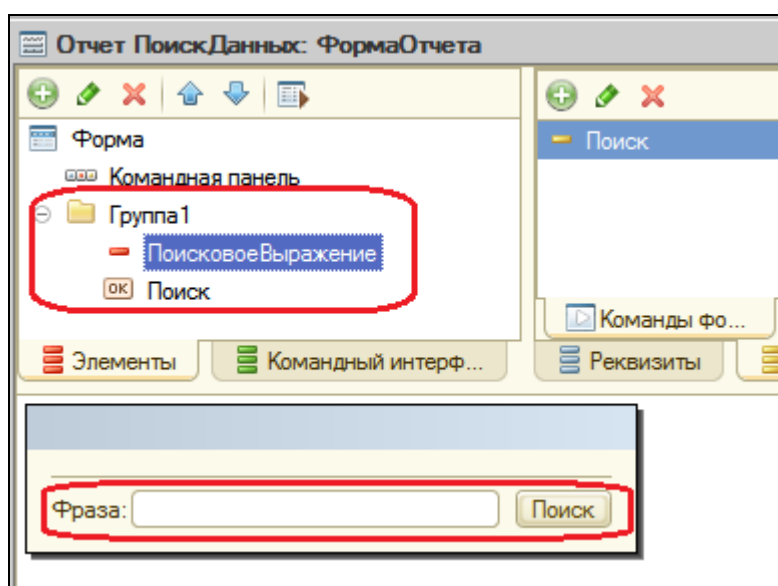
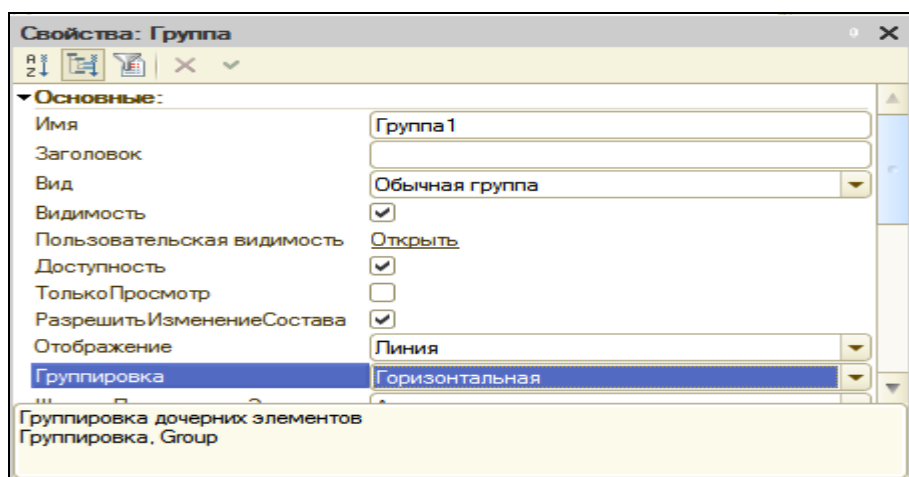
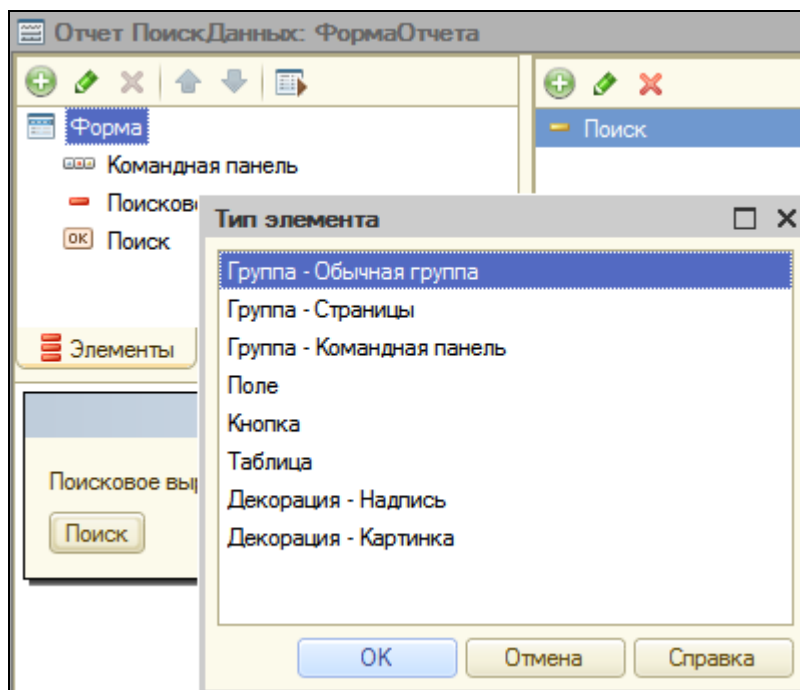


Однако мы видим, что все добавляемые элементы формы располагаются вертикально – друг под другом. Это потому что в свойствах формы установлена группировка элементов формы по умолчанию – **Вертикальная**. Нас это устраивает, но некоторые элементы формы, в частности поле **ПоисковоеВыражение** и кнопку **Найти**, хотелось бы расположить горизонтально – рядом друг с другом.

Для этого нужно добавить в форму группу и определить в ней тип группировки элементов **Горизонтальная**.

Выделим строку **Форма** в дереве элементов формы, нажмите кнопку **Добавить** в командной панели и выберем тип элемента **Группа – Обычная группа**.

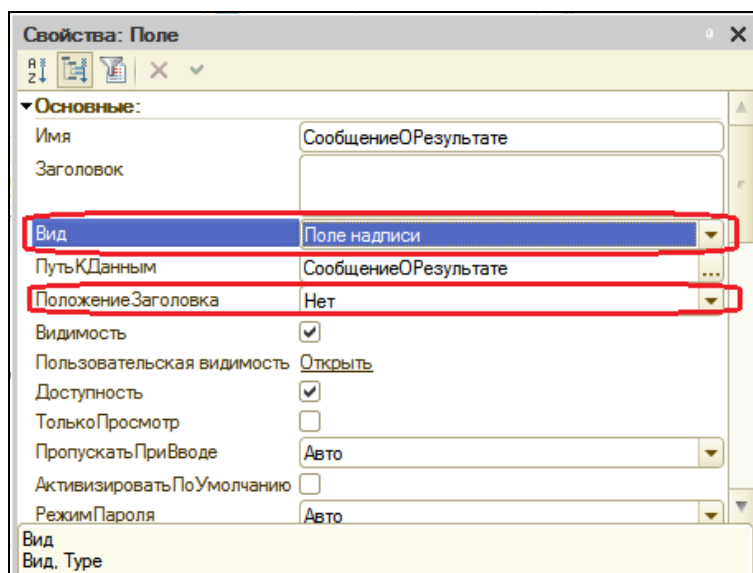
В открывшемся окне свойств группы зададим тип группировки **Горизонтальная**. Затем мышью перетащим в эту группу элементы **ПоисковоеВыражение** и **Поиск**. Теперь мы добились желаемого расположения элементов.



Добавим в форму реквизит **СообщениеОРезультате** и перетащим его в окно элементов формы в группу 1. Появится окно свойств поля. В нем зададим **ПоложениеЗаголовка** в значение **Нет**.

В поле **Вид** установим значение **Поле надписи**.

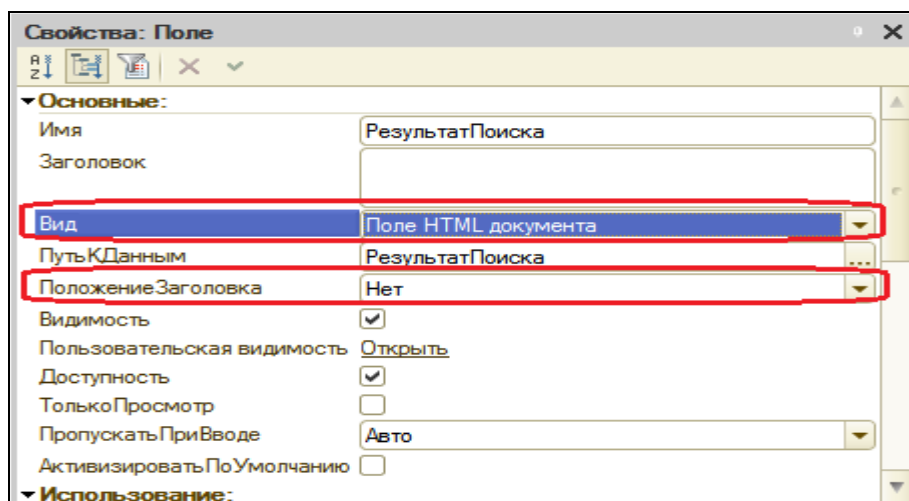
В поле надписи **СообщениеОРезультате** мы будем выводить сообщение о результате поиска.



Добавим в форму реквизит **РезультатПоиска** и перетащим его в окно элементов формы в **Группу1**.

В свойствах поля зададим **ПоложениеЗаголовка** в значение **Нет**. В поле **Вид** установим значение **Поле HTML документа**.

В поле HTML документа **РезультатПоиска** мы будем выводить найденные элементы поиска.

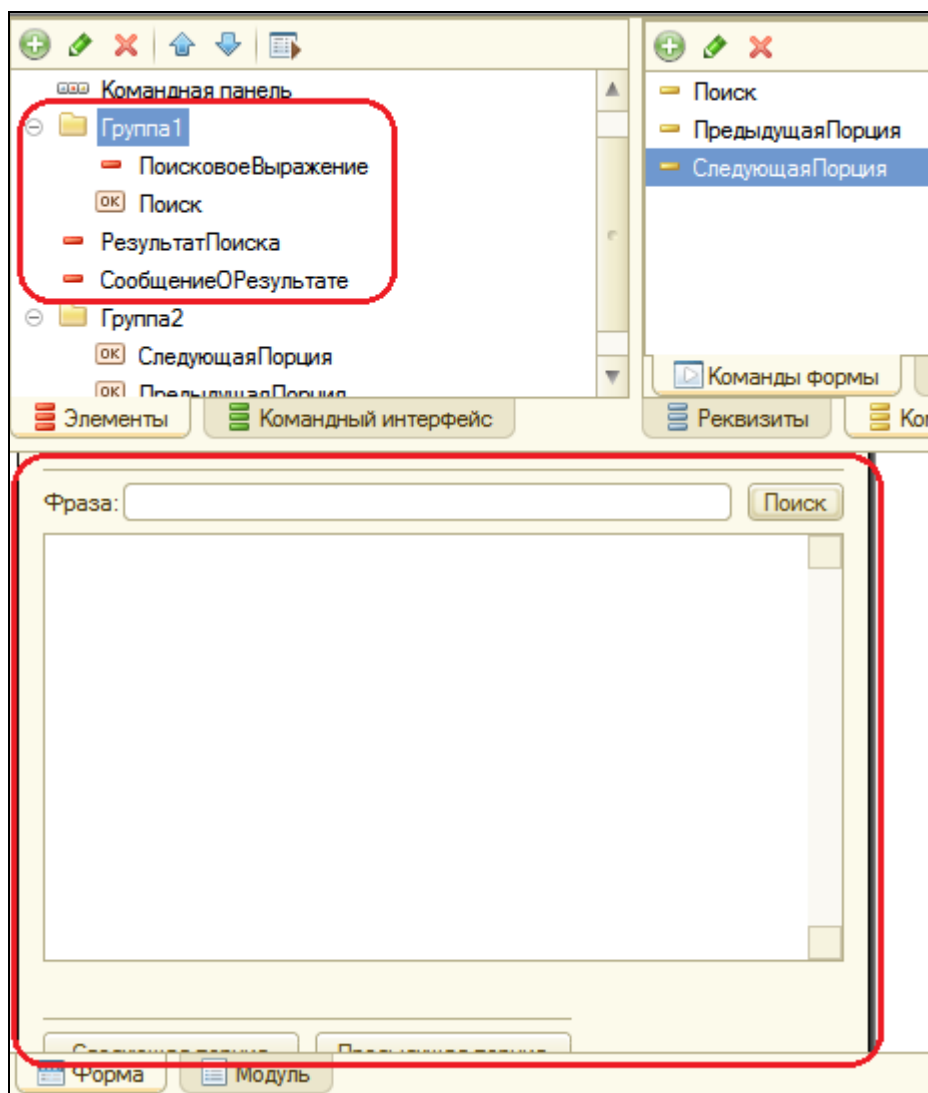




На закладке **Команды** поочередно создадим команды **ПредыдущаяПорция** и **СледующаяПорция**.

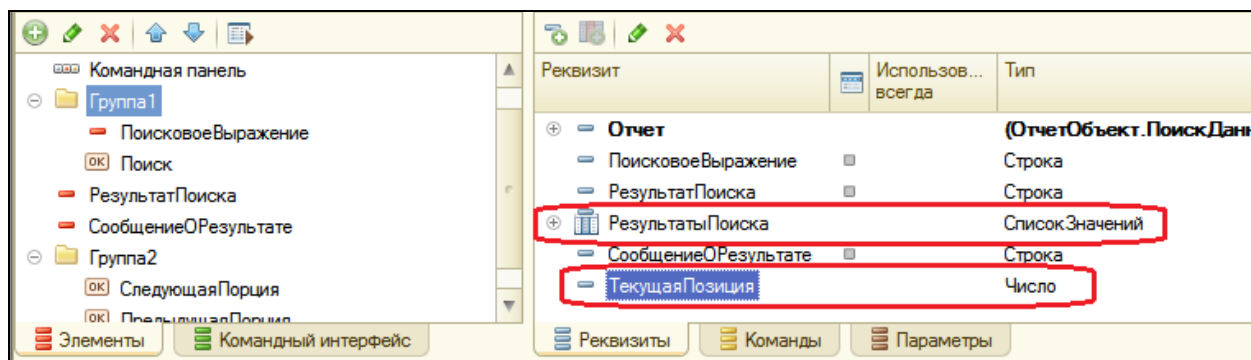
Нажмем кнопку открытия в строке **Действие** для каждой команды.

Шаблоны обработчиков событий пока заполнять не будем, а перейдем на закладку **Форма**, выделим корень дерева элементов и добавим новую группу. Зададим для этой группы тип группировки **Горизонтальная**. Затем перетащим наши команды в эту группу. Перетащите **Группу1** в командную панель, чтобы форма приняла вид:



Затем добавим в форму реквизит **РезультатыПоиска** типа **СписокЗначений** для хранения найденных элементов поиска. А также добавим в форму реквизит **ТекущаяПозиция** типа **Число** для хранения текущей позиции списка.

Эти реквизиты играют вспомогательную роль и в форму их перетаскивать не нужно.



Теперь реализуем работу формы с помощью кода на встроенном языке.

Для обработчиков событий нажатия кнопок **Поиск**, **Предыдущая порция** и **Следующая порция** напишем код, который позволит нам выполнять поиск в соответствии с направлением поиска (искать сначала, искать вперед или назад). Это делается на вкладке **Команды** в свойствах команды.

```
&НаКлиенте
Процедура Поиск()
    Искать(0);
КонiecПроцедуры

&НаКлиенте
Процедура ПредыдущаяПорция()
    Искать(-1);
КонiecПроцедуры

&НаКлиенте
Процедура СледующаяПорция()
    Искать(1);
КонiecПроцедуры
```

Все эти обработчики вызывают процедуру **Искать()**. В ней проверяется, задано ли выражение для поиска, и вызывается собственно процедура полнотекстового поиска, выполняющаяся на сервере **ИскатьСервер()**, в которую передается направление поиска.

После этих процедур вставим следующий текст процедуры поиска на клиенте:

```
&НаКлиенте
// Процедура поиска, получение и отображение результата
Процедура Искать(Направление)
    Если ПустаяСтрока(ПоисковоеВыражение) Тогда
        Предупреждение("Не задана строка поиска.");
        Возврат;
    КонiecЕсли;
```

```
ИскатьСервер(Направление);  
КонецПроцедуры
```

И процедуру поиска на сервере:

```
&НаСервере  
Процедура ИскатьСервер(Направление) Экспорт  
    СписокПоиска = ПолнотекстовыйПоиск.СоздатьСписок();  
    СписокПоиска.СтрокаПоиска = ПоисковоеВыражение;  
  
    Если Направление = 0 Тогда  
        СписокПоиска.ПерваяЧасть();  
    ИначеЕсли Направление = -1 Тогда  
        СписокПоиска.ПредыдущаяЧасть(ТекущаяПозиция);  
    ИначеЕсли Направление = 1 Тогда  
        СписокПоиска.СледующаяЧасть(ТекущаяПозиция);  
    КонецЕсли;  
  
    РезультатыПоиска.Очистить();  
    Для Каждого Результат Из СписокПоиска Цикл  
        РезультатыПоиска.Добавить(Результат.Значение);  
    КонецЦикла;  
  
    РезультатПоиска =  
    СписокПоиска.ПолучитьОтображение(ВидОтображенияПолнотекстовогоПоиска.HTMLТекст);  
    ТекущаяПозиция = СписокПоиска.НачальнаяПозиция();  
    ПолноеКоличество = СписокПоиска.ПолноеКоличество();  
  
    Если СписокПоиска.Количество() <> 0 Тогда  
        СообщениеОРезультате = "Показаны " + Строка(ТекущаяПозиция +  
1) + " - " + Строка(ТекущаяПозиция + СписокПоиска.Количество()) + " из " +  
Строка(ПолноеКоличество);  
        Элементы.СледующаяПорция.Доступность = (ПолноеКоличество -  
ТекущаяПозиция) > СписокПоиска.Количество();  
        Элементы.ПредыдущаяПорция.Доступность = (ТекущаяПозиция > 0);  
    Иначе  
        СообщениеОРезультате = "Не найдено";  
        Элементы.СледующаяПорция.Доступность = Ложь;  
        Элементы.ПредыдущаяПорция.Доступность = Ложь;  
    КонецЕсли;  
КонецПроцедуры
```

Сначала в этой процедуре мы создаем список поиска, используя метод **СоздатьСписок()** объекта **ПолнотекстовыйПоиск**, и сохраняем его в переменной **СписокПоиска**.

Затем устанавливаем поисковое выражение, введенное пользователем в качестве строки поиска. Затем в зависимости от направления поиска выполняем метод **ПерваяЧасть()**, **ПредыдущаяЧасть()** или **СледующаяЧасть()**, который собственно запускает полнотекстовый

поиск и возвращает соответственно первую порцию результатов, либо предыдущую порцию, либо следующую порцию в зависимости от текущей позиции поиска. По умолчанию порция содержит 20 элементов. Синонимом порции является страница с результатами поиска.

Затем мы очищаем список значений **РезультатыПоиска** и заполняем его найденными элементами.

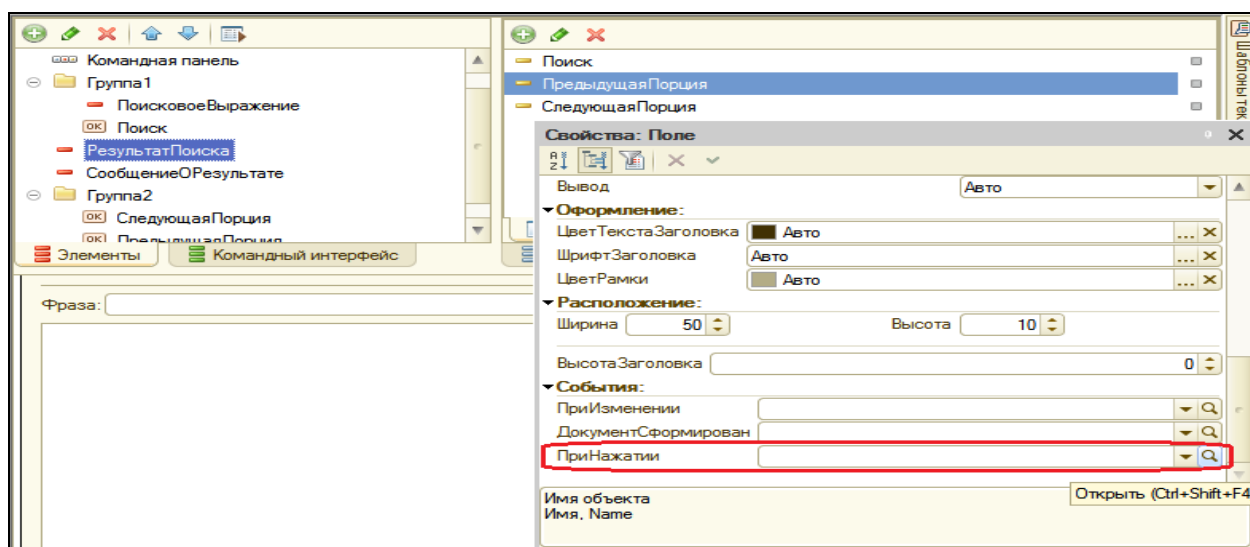
Получаем результат полнотекстового поиска в виде HTML-текста и сохраняем этот текст в реквизите **РезультатПоиска**, имеющим тип HTML-документа.

После этого мы анализируем количество элементов в списке поиска. Если он не содержит ни одного элемента, то мы выводим в форму соответствующее сообщение. В противном случае мы формируем сообщение о том, какие элементы отображены и сколько всего элементов найдено. В зависимости от того, какая порция полученных результатов отображена, мы устанавливаем доступность кнопок Предыдущая порция и Следующая порция.

Заключительным штрихом будет создание обработчика события **ПриНажатии** поля HTML-документа **РезультатПоиска**, расположенного в форме.

Результат поиска содержит гиперссылки на номера элементов списка поиска. Нам бы хотелось, чтобы при нажатии на ссылку система открывала форму того объекта, который содержится в этом элементе списка.

Для этого создадим обработчик события **ПриНажатии** поля HTML-документа **РезультатПоиска**:



```

&НаКлиенте
Процедура РезультатПоискаПриНажатии(Элемент, ДанныеСобытия,
СтандартнаяОбработка)
    ЭлементHTML = ДанныеСобытия.Event.srcElement;
Если (ЭлементHTML.id = "FullTextSearchListItem") Тогда

    // Получить имя файла (номер строки списка поиска), содержащегося в
гиперссылке
    НомерВСписке = Число(ЭлементHTML.nameProp);

    // Получить строку списка поиска по номеру
    ВыбраннаяСтрока = РезультатыПоиска[НомерВСписке].Значение;

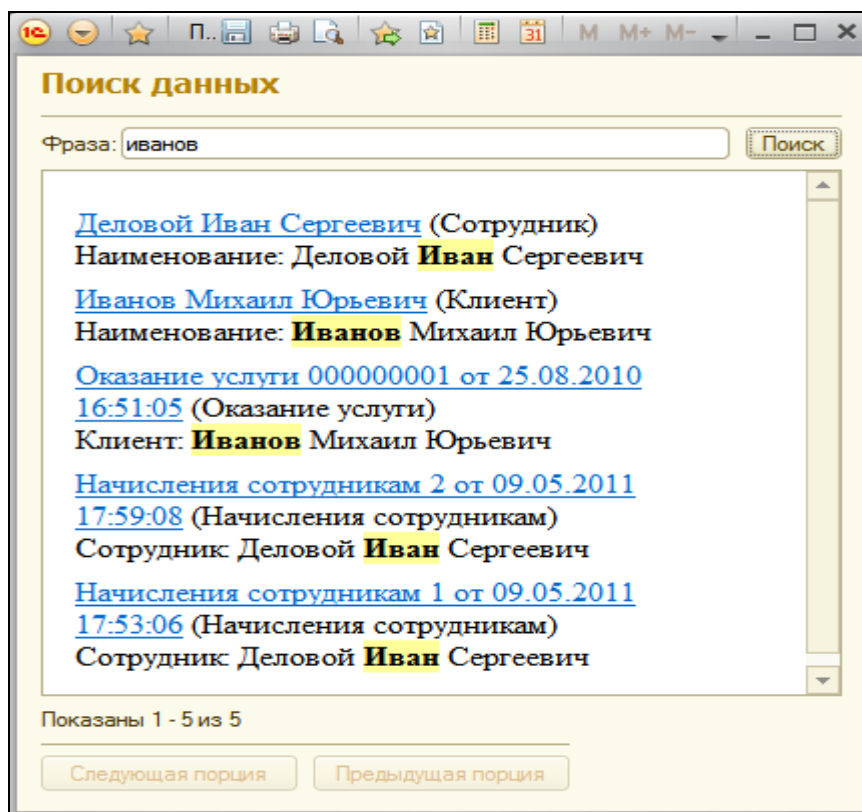
    // Открыть форму найденного объекта
    ОткрытьЗначение(ВыбраннаяСтрока);
    СтандартнаяОбработка = Ложь;
КонецЕсли;
КонецПроцедуры

```

В окне редактирования отчета **ПоискДанных** на закладке **Подсистемы** отметим все подсистемы, чтобы все пользователи в соответствии с их правами могли пользоваться поиском данных.

В режиме 1С:Предприятие

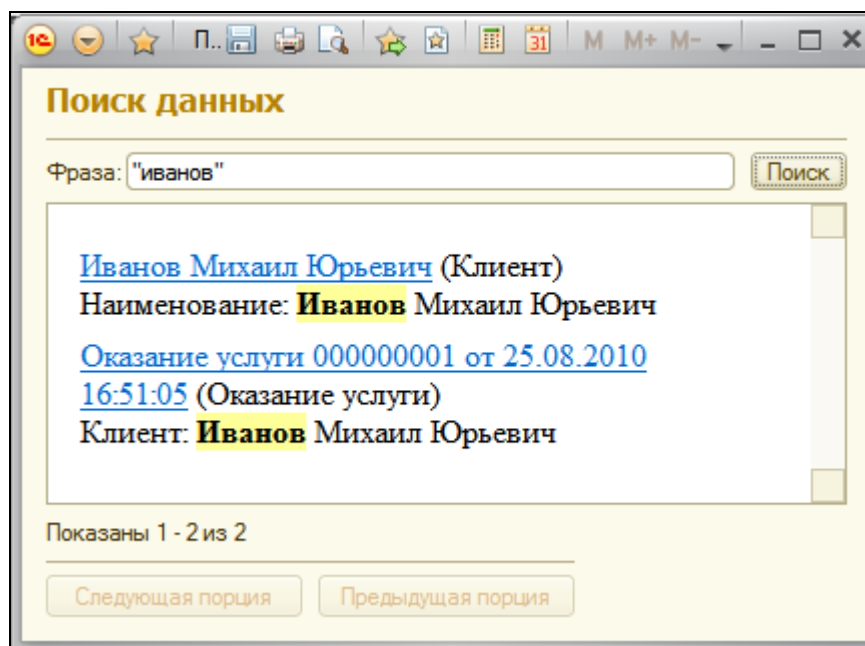
Для начала попробуем найти данные, связанные с Ивановым. Введем **иванов** и нажмем ctrl+enter или кнопку Поиск.



Результат поиска содержит 5 элементов и найденные слова в реквизитах этих документов выделены желтым фоном.

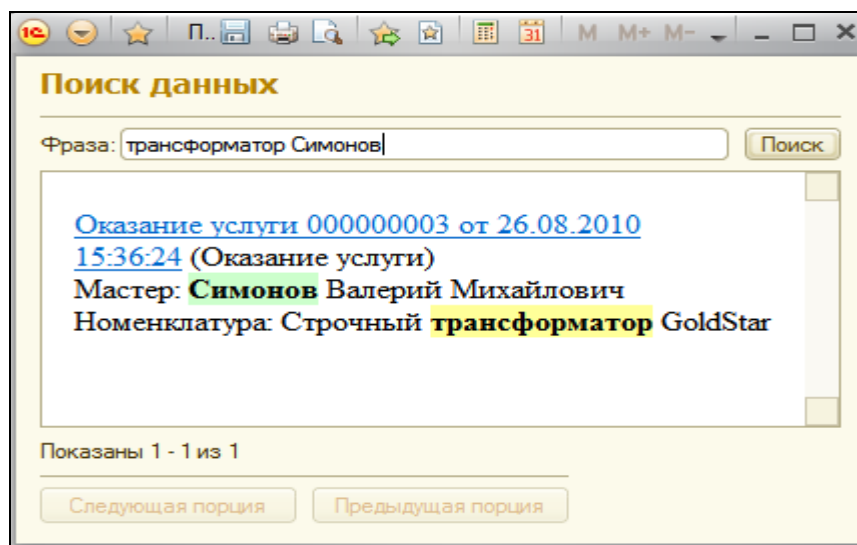
Обратите внимание, что система нашла все словоформы введенного выражения (Иван).

Чтобы выполнить точный поиск по указанному выражению, его необходимо заключить в кавычки (также и в интернете).



Для получения наилучших результатов полнотекстового поиска рекомендуется использовать в поисковом выражении пару слов.

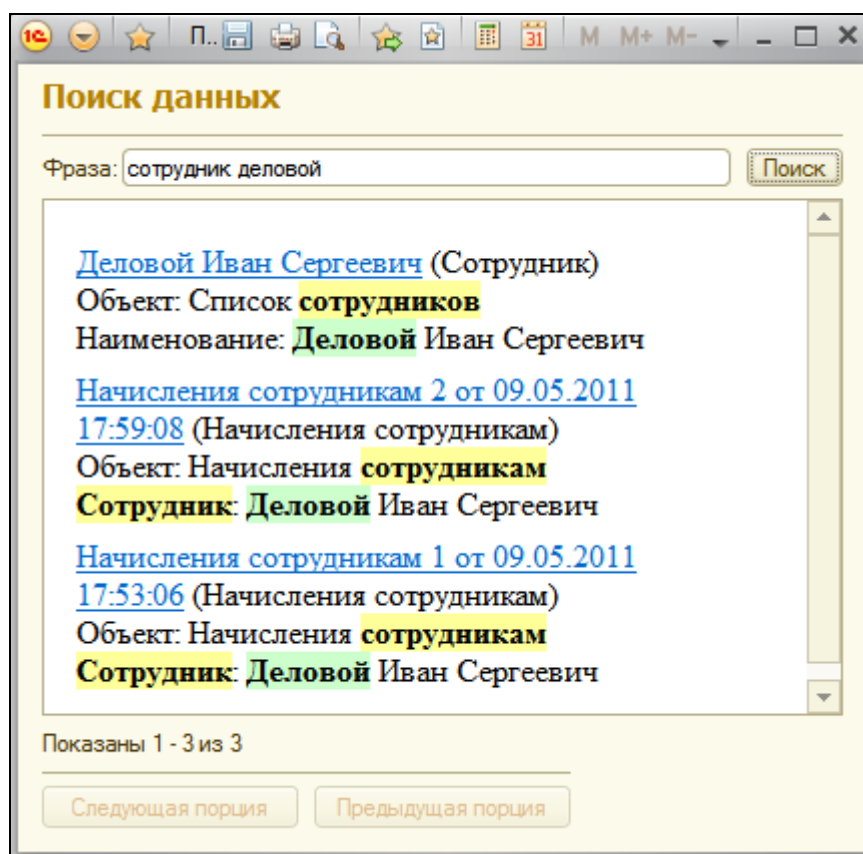
Например, если требуется узнать, какого числа клиенту Симонову заменили трансформатор в телевизоре, можно ввести поисковое выражение **трансформатор Симонов**.



При нажатии на гиперссылку система откроет документ Оказание услуги №3 и можно будет просмотреть полный перечень работ, выполненных для этого клиента.

Система индексирует не только данные, содержащиеся в объектах конфигурации, но и имена реквизитов и объектов метаданных.

Поэтому, например, если требуется найти информацию о сотруднике по фамилии Деловой, в поисковом выражении следует указать имя справочника, который нас интересует: **сотрудник деловой**.



Как видите, система нашла также и документы начисления сотрудникам, которые содержат формы слова «сотрудники», но, несмотря на это, искомый нами справочник Сотрудники находится первым в списке найденных.

### Контрольные вопросы

- ✓ Для чего предназначен полнотекстовый поиск в данных.
- ✓ Какова стратегия полнотекстового индексирования инфобазы.
- ✓ Как создать отчет, выполняющий поиск в данных.
- ✓ Как составлять простейшие поисковые выражения.

## Практическая работа № 19

### *Выполнение заданий по расписанию (1:00)*

#### **Внимание!**

Если вы используете учебную версию программы, то пример, приведенный в этой работе, удастся проверить в работе лишь частично. Для полной проверки примера требуется одновременный запуск двух клиентских сеансов: планировщика заданий и обычного пользовательского сеанса.

Учебная версия позволяет запускать только один пользовательский сеанс. Поэтому их работу можно будет проверить только по отдельности. Сначала запустить планировщик заданий и убедиться, что регламентные задания запускаются. Затем закрыть сеанс планировщика и открыть обычный пользовательский сеанс, чтобы выполнить поиск в данных.

Любая информационная база системы 1С:Предприятие требует периодического выполнения определенного набора регламентных операций.

Например, по мере изменения существующих данных или добавления новых необходимо выполнять резервное копирование. Тогда, если в результате сбоя инфобазы окажется неработоспособной, основную часть данных можно будет восстановить из резервной копии. Поэтому чем чаще выполняется резервное копирование, тем меньшее количество данных придется вводить повторно в случае сбоя.

Другой пример – полнотекстовый поиск данных. Для него нужно индексировать данные по мере их обновления, т.е. с некоторой периодичностью.

Для автоматизации подобных операций существует механизм заданий. Он позволяет создавать задания, каждое из которых представляет собой некоторую последовательность действий, описанных с помощью встроенного языка. Для каждого задания может быть назначено расписание, в соответствии с которым это задание будет автоматически запущено на исполнение.

В этой работе мы рассмотрим использование механизма заданий на примере автоматизации двух регламентных операций, связанных с полнотекстовым поиском: операции полнотекстового индексирования и операции слияния индексов.



Мы опишем эти операции средствами встроенного языка, установим расписание для их автоматического выполнения и узнаем, как обеспечить автоматическое выполнение этих заданий по расписанию в случае файлового варианта работы системы.

## **Постановка задачи**

В предыдущей работе мы узнали, что для возможностей выполнения полнотекстового поиска обязательно должен существовать полнотекстовый индекс. Он создается один раз и затем должен периодически обновляться.

На самом деле полнотекстовый индекс состоит из двух индексов: основного и дополнительного.

Поиск осуществляется по обоим индексам. Но отличие в следующем.

*Основной индекс* сделан так, чтобы обеспечивать максимальную скорость поиска при большом объеме данных, но добавление в него данных осуществляется медленно.

*Дополнительный индекс* противоположен основному: добавление данных в него осуществляется быстро, но при значительном объеме данных поиск будет выполняться медленно.

Стратегия использования индексов такова. Основная масса данных находится в основном индексе, что позволяет выполнить быстрый поиск. Новые данные, измененные или добавленные в систему, добавляются в дополнительный индекс непосредственно во время работы системы и пользователей с требуемой периодичностью (раз в час, раз в минуту). Такое добавление происходит быстро и не оказывает влияния на производительность системы. Пока объем данных в дополнительном индексе невелик, поиск по нему также выполняется быстро.

В период малой активности пользователей или в период выполнения регламентных действий с инфобазой выполняется слияние основного и дополнительного индексов (раз в сутки). В результате новые данные помещаются в основной индекс, а дополнительный индекс очищается и готов к быстрому приему следующих данных.

В результате для автоматизации полнотекстового индексирования нам понадобится два задания. Первое будет выполнять индексирование без слияния и запускаться каждую минуту. Второе будет выполнять слияние индексов и запускаться раз в сутки ночью.

## Что такое регламентное задание

*Регламентные задания* располагаются в дереве объектов конфигурации, в ветке **Общие**. Каждое регламентное задание содержит два основных свойства: **Имя метода** и **Расписание**.

Свойство **Имя метода** связывает регламентное задание с некоторой процедурой или функцией общего модуля, которая и будет исполняться. Эта процедура должна содержать алгоритм на встроенном языке, описывающий все операции, которые должны быть выполнены.

Свойство **Расписание** позволяет задать периодичность выполнения этой процедуры.

Кроме перечисленных свойств регламентное задание содержит другие свойства, например **Интервал повтора при аварийном завершении** и **Количество повторов при аварийном завершении**.

Если по какой-то причине выполнение регламентного задания закончится неудачно, система может автоматически запустить это задание указанное количество раз по прошествии указанного периода времени.

### Создание регламентных заданий

#### В режиме Конфигуратор

Сначала создадим первое регламентное задание по обновления индекса.


Раскроем ветвь **Общие** дерева объектов конфигурации. Выделим строку **Регламентные задания** и добавим новый объект **Регламентное задание** с именем **ОбновлениеИндекса**.

После этого создадим процедуру, которая и будет выполнять обновление полнотекстового индекса нашей информационной базы.

В качестве такой процедуры может выступать любая процедура или функция неглобального общего модуля, которую можно вызвать на сервере (у общего модуля должно быть установлено свойство **Сервер**).

Добавим в конфигурацию общий модуль с именем **РегламентныеПроцедуры** и установим флажок **Вызов сервера** для видимости его экспортных процедур и функций.

Вернемся к свойствам регламентного задания **ОбновлениеИндекса**.

Нажмем кнопку выбора  у поля ввода **Имя метода**. Откроется окно выбора общего модуля. Выберем модуль **РегламентныеПроцедуры**.

В этом модуле будет создан шаблон процедуры **ОбновлениеИндекса()**.

Заполним его следующим образом:

```
Процедура ОбновлениеИндекса() Экспорт
    Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =
РежимПолнотекстовогоПоиска.Разрешить Тогда
        Если Не ПолнотекстовыйПоиск.ИндексАктualен() Тогда
            ПолнотекстовыйПоиск.ОбновитьИндекс( , Истина);
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

Сначала в этой процедуре проверяется возможность выполнения операций, связанных с полнотекстовым поиском (ведь они могут быть запрещены, например, интерактивно).

Если они разрешены, то проверяется, актуален ли полнотекстовый поиск (если после последнего индексирования данные, подлежащие полнотекстовому индексированию, не изменялись, то индекс будет актуален и повторное индексирование не требуется).

В случае необходимости индексирования вызывается метод **ОбновитьИндекс()** менеджера полнотекстового поиска.

Первый параметр этого метода отвечает за слияние индексов и по умолчанию имеет параметр **Ложь**, что значит, слияние выполняться не будет.

Второй параметр метода определяет, какое количество данных будет индексироваться: сразу все или порциями. Наша задача – выполнить индексирование как можно быстрее, поэтому указываем индексирование порциями (значение **Истина**).

Размер одной порции фиксирован – 10 000 объектов. Т.о., если в данный момент требуется проиндексировать 15 000 объектов, то при вызове этого метода из них будет проиндексировано 10 000 (первая порция), а оставшиеся объекты – при следующем вызове этого метода (при следующем запуске регламентного задания).

Перейдем к составлению расписания запуска регламентного задания.

Нажмите ссылку **Открыть** в свойствах регламентного задания в строке Расписание. Откроется диалог редактирования расписания. Он содержит несколько вкладок, которые позволяют задать различные виды расписаний, в нижней части диалога отображается итоговый результат всех установок.

Наша задача – запускать регламентное задание ежедневно, каждую минуту.

Поэтому на закладке **Общие** укажем, что запуск задания должен повторяться каждый день (Повторять каждые: 1 дн.).

Теперь перейдем на закладку **Дневное** и зададим порядок запуска задания в течение дня.

Укажем, что запуск задания должен повторяться каждые 60 секунд (Повторять через: 60 сек.).

Вроде бы мы получили что хотели, но наша фирма не работает круглосуточно, и запуск этого задания в ночное время будет явно бесполезным - данные в базе не изменяются. В то же время вполне возможно, что некоторые сотрудники задерживаются после окончания рабочего дня.

Поэтому доработаем расписание: укажем Время начала: 08:00.

Нажмите OK.

В качестве последнего штриха установим в свойствах регламентного задания флажок **Предопределенное**.

Свойства: ОбновлениеИндекса

▼ Основные:

Имя: ОбновлениеИндекса

Синоним: Обновление индекса

Комментарий:

Имя метода: РегламентныеПроцедуры.С ...

Наименование:

Ключ:

Расписание: Открыть

Использование: ☒

Предопределенное: ☒

Количество повторов при аварийном завершении: 3


Интервал повтора при аварийном завершении: 10

Предопределенное задание

Установка этого свойства означает, что после запуска системы в режиме 1С:Предприятие будет создано одно предопределенное регламентное задание. В противном случае такое задание пришлось бы создавать средствами встроенного языка.

На этом создание регламентного задания **Обновление индекса** завершено.

Теперь по аналогии создадим второе регламентное задание – **СлияниеИндексов**.

В свойствах нажмем кнопку выбора  у поля ввода **Имя метода**. В открывшемся диалоге выберем модуль **РегламентныеПроцедуры**.

В этом модуле будет создан шаблон процедуры **СлияниеИндексов()**. Заполним его следующим образом:

```
Процедура СлияниеИндексов() Экспорт
    Если ПолнотекстовыйПоиск.ПолучитьРежимПолнотекстовогоПоиска() =
РежимПолнотекстовогоПоиска.Разрешить Тогда
        Если Не ПолнотекстовыйПоиск.ИндексАктуален() Тогда
            ПолнотекстовыйПоиск.ОбновитьИндекс(Истина);
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

В свойствах задания установим флажок **Предопределенное** и приступим к редактированию расписания.

На закладке **Общее** укажем, что задание будет запускаться каждый день, а на закладке **Дневное** укажем время начала выполнения задания – **01:00**.

В результате мы получим следующее расписание запуска регламентного задания: **Выполнять: каждый день; с 1:00:00 один раз в день**.

На этом создание регламентного задания **СлияниеИндексов** закончено.

### **Планировщик заданий**

1С:Предприятие поддерживает два варианта работы: файловый и клиент-серверный.

Файловый вариант прост в установке и практически не требует никакого администрирования. Именно в нем работает наша демонстрационная база. Однако, за любую простоту использования нужно чем-то расплачиваться.

Если бы наша информационная база работала в клиент-серверном варианте, то для автоматического запуска и выполнения созданных нами заданий не требовалось бы дополнительных действий. Можно было бы обновить конфигурацию базы данных, и менеджер кластера серверов 1С:Предприятия начал бы самостоятельно выполнять задания в соответствии с указанным расписанием.

В нашем случае (файловый вариант работы) такого «промежуточного звена», которое могло бы взять на себя автоматическое выполнение заданий, не существует – есть только клиенты и информационная база.

Поэтому за простоту файлового варианта придется заплатить тем, что для автоматического выполнения заданий необходимо всегда иметь одно работающее клиентское соединение с информационной базой, которое будет заниматься только запуском заданий по расписанию. Назовем такое соединение *планировщик заданий*.

В нашем примере мы создадим простую обработку, которая будет запускать задания по расписанию. Затем соединимся с нашей информационной базой в режиме 1С:Предприятие и запустим эту обработку.

Главное – не закрывать обработку и не закрывать это окно 1С:Предприятия, т.к. именно в нем будут выполняться регламентные задания.

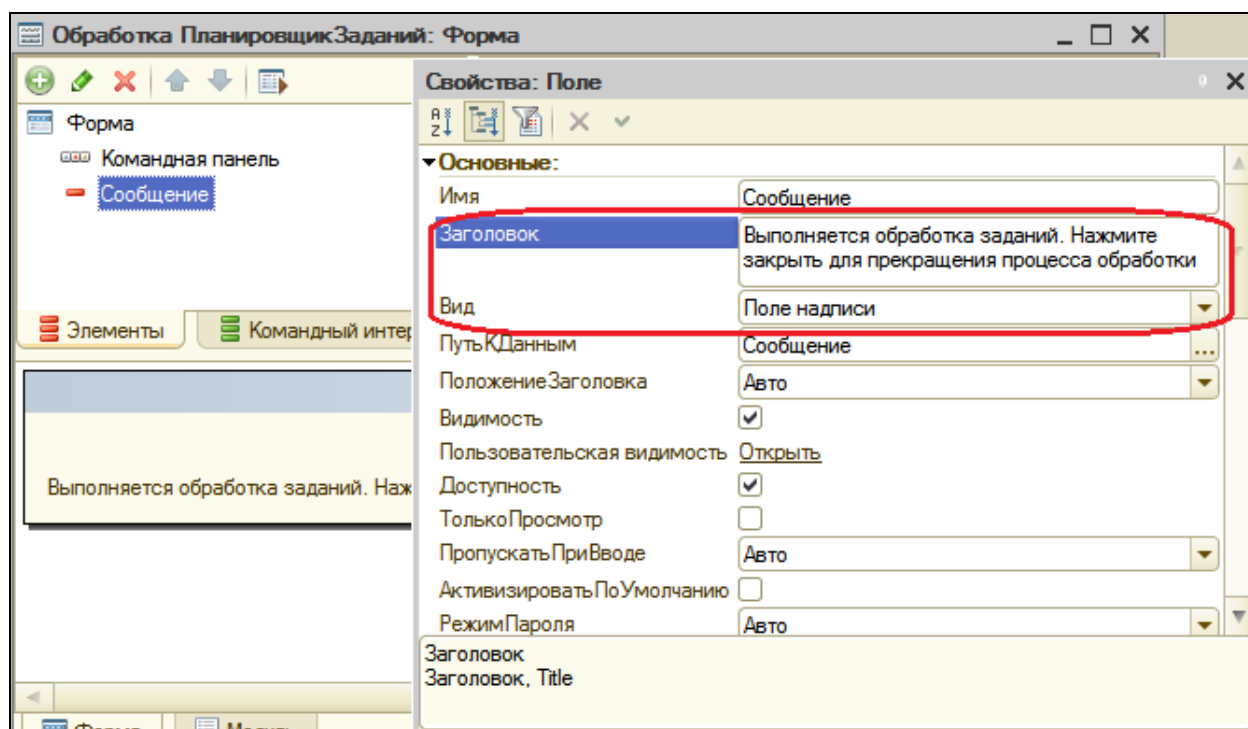
## В режиме Конфигуратор

Выделим ветвь **Обработки** в дереве объектов конфигурации и добавим новый объект *Обработка* с именем **ПланировщикЗаданий**.

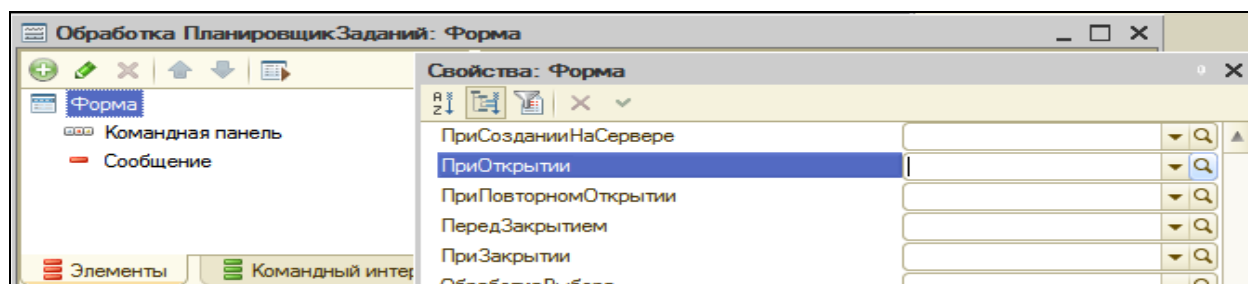
На закладке **Формы** создадим основную форму обработки.

В окне редактора форм на закладке **Реквизиты** добавим реквизит формы **Сообщение** и перетащим его в окно элементов формы.

В открывшемся окне свойств поля **Сообщение** зададим вид поля – **Поле надписи** и заголовок – **Выполняется обработка заданий. Нажмите закрыть для прекращения процесса обработки**.



Откроем события формы и нажмем кнопку открытия у события **ПриОткрытии**. В обработчик этого события поместим следующий текст:





```

&НаКлиенте
Процедура ПриОткрытии(Отказ)
    #Если ТолстыйКлиентУправляемоеПриложение Тогда
        ПодключитьОбработчикОжидания("ОбработкаЗаданий", 3);
    #Иначе
        Предупреждение("Обработка может быть запущена только в толстом
клиенте!");
        Закреть();
    #КонецЕсли
КонецПроцедуры

```

А также в модуле формы поместим сам обработчик ожидания – процедуру **ОбработкаЗаданий()**.

```

&НаКлиенте
Процедура ОбработкаЗаданий()

    #Если ТолстыйКлиентУправляемоеПриложение Тогда
        ВыполнитьОбработкуЗаданий();
    #КонецЕсли

КонецПроцедуры

```

Таким образом, при открытии формы выполняется подключение в качестве обработчика ожидания процедуры с именем **ОбработкаЗаданий()**.

Эта процедура будет вызываться каждые три секунды.

В свою очередь, процедура **ОбработкаЗаданий()** выполняет одно-единственное действие – вызывает метод **ВыполнитьОбработкуЗаданий()**, который проверяет, существуют ли задания, которые в соответствии с их расписанием, должны быть выполнены. Если такие задания существуют, он запускает их на выполнение.

В обеих процедурах используются инструкции препроцессора (после символа #) для того, чтобы указать, что фрагмент кода будет присутствовать только в том случае, если запущен толстый клиент в управляемом режиме, т.к. именно в этом режиме будет запускаться наша обработка **ПланировщикЗаданий**.

Для того, чтобы проверить, что запуск действительно происходит, добавим в начало процедуры **ОбновлениеИндекса** (общий модуль **РегламентныеПроцедуры**) следующую строку:

```
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = "Запуск регламентного задания Обновление индекса " +  
ТекущаяДата();  
Сообщение.Сообщить();
```

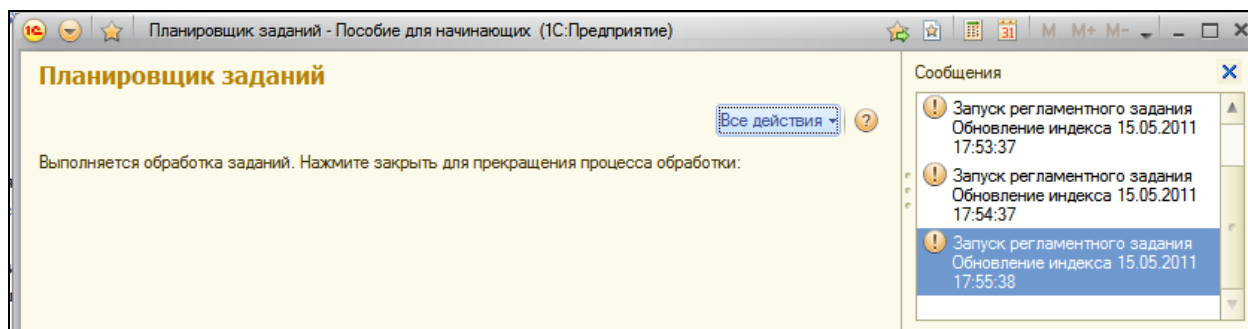
В заключение в окне редактирования объекта *Обработка ПланировщикЗаданий* отметьте подсистему **Предприятие**.

В режиме 1С:Предприятие

Обновим конфигурацию базы данных, нажав кнопку в Конфигураторе в панели инструментов **Обновить конфигурацию базы данных (F7)** и после этого запустим систему в режиме 1С:Предприятие (толстый клиент). Для этого зайдите в **Пуск - Все программы-1С Предприятие 8.2 - Дополнительно - <номер версии> - 1С Предприятие (толстый клиент)**. В окне запуска откроем в режиме 1С:Предприятие нашу информационную базу.

В разделе **Предприятие** откроем обработку **Планировщик заданий** и подождем несколько минут.

В результате в окно сообщений будет выведен, например, такой текст:



Таким образом, мы видим, что задание обновления индекса запускается каждые 60 секунд, как мы и указали в расписании.

В соединении, которое занято запуском регламентных заданий, не рекомендуется выполнять какие-либо другие задачи. Для обычной работы с нашей информационной базой необходимо еще раз запустить систему в режиме 1С:Предприятие, выбрать нашу информационную базу и создать тем самым второе соединение, в котором уже и выполнять обычную работу пользователя.

## **Контрольные вопросы**

- ✓ Для чего предназначены регламентные задания.
- ✓ Как задать расписание для автоматического запуска заданий.
- ✓ Как обеспечить запуск заданий по расписанию в файловом варианте работы.
- ✓ Что такое основной и дополнительный полнотекстовые индексы.

## Практическая работа № 20

### *Редактирование движений в форме документа (00:40)*

В нашей информационной базе, как и в любой другой, следует предусмотреть возможность ввода начальных остатков в регистры. Это необходимо для того, чтобы пользователи могли начать работу с нашей информационной базой не с чистого листа, а с некоторого исходного состояния, которое было в их прежней системе учета (на бумаге например).

Задача ввода начальных остатков отличается от прочих алгоритмов изменения состояния регистров тем, что подразумевает изменение данных непосредственно в регистрах, без использования промежуточных алгоритмов (заполнения документов данными, проведения документов, контроля правильности данных, указанных в документах и т.д.).

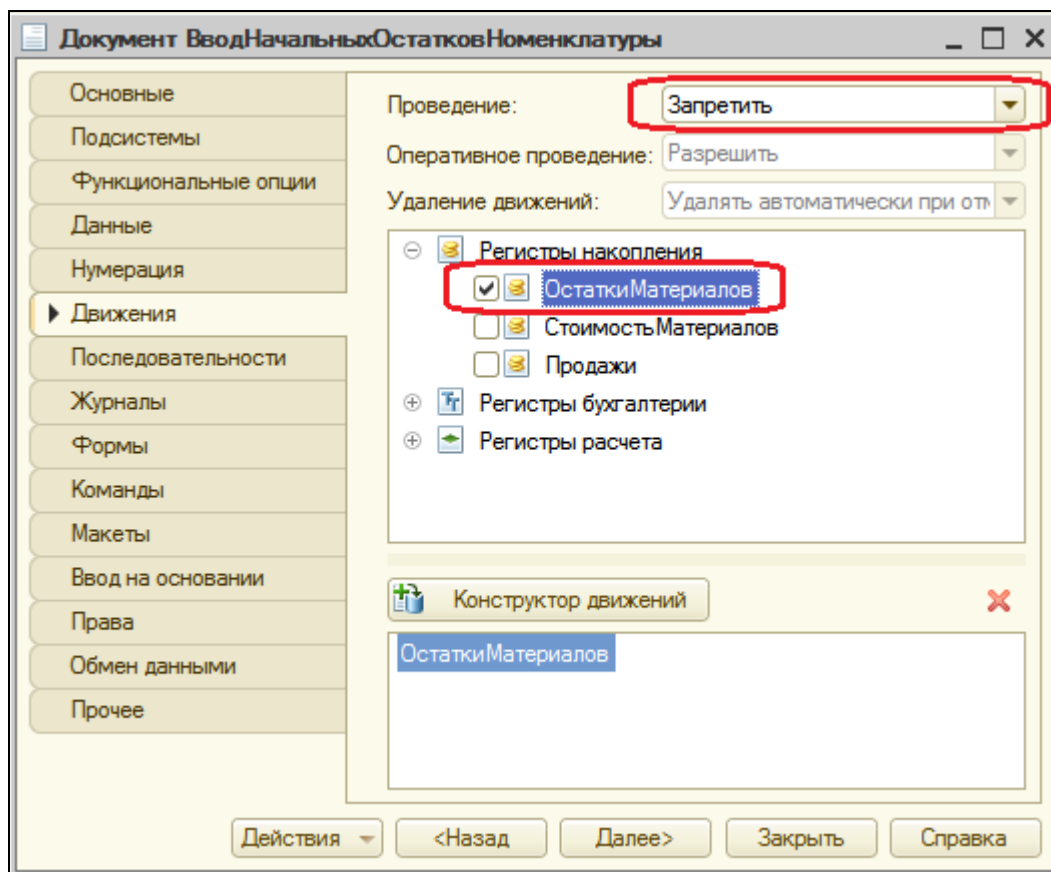
Рассмотрим пример ввода начальных остатков регистра накопления **ОстаткиМатериалов**.

Для выполнения этой задачи мы создадим документ, в котором будем вручную редактировать его движения по регистру **ОстаткиМатериалов** прямо в форме документа.

В режиме Конфигуратор

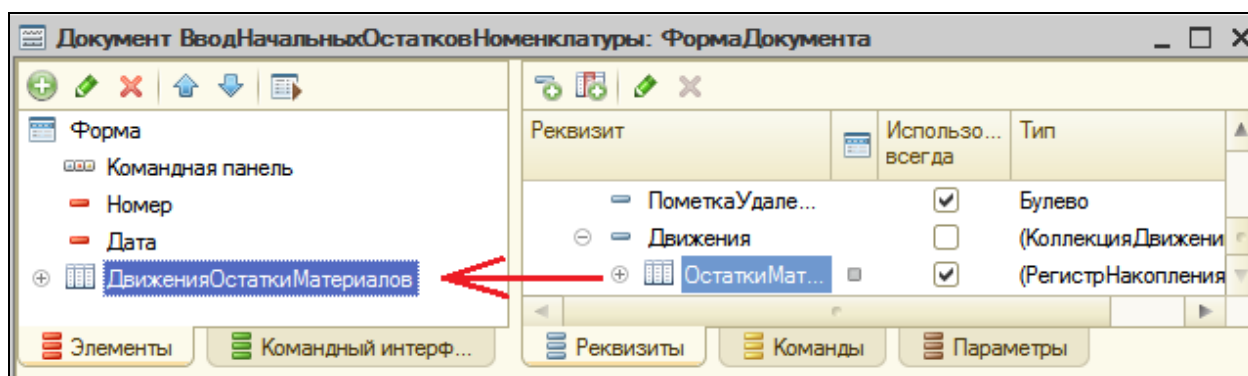
Создадим новый объект документ с именем **ВводНачальныхОстатковНоменклатуры**.

На закладке **Движения** запретим проведение документа (поскольку сами будем формировать записи регистра) и отметим, что движения документа будут находиться в регистре накопления **ОстаткиМатериалов**.



После этого перейдем на закладку **Формы** и создадим основную форму документа.

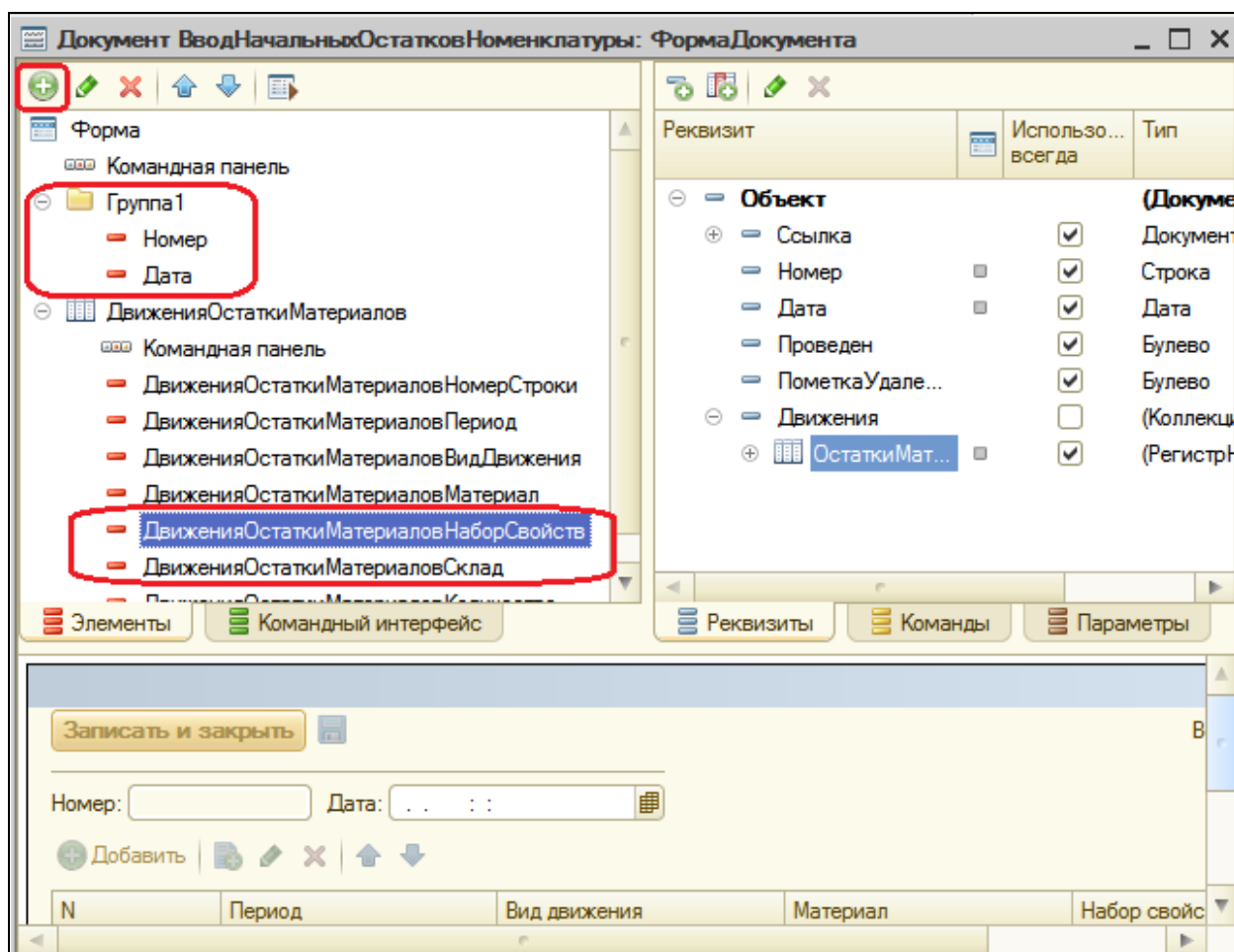
В окне редактора форм на закладке **Реквизиты** раскроем основной реквизит формы **Объект**, затем раскроем **Движения**, найдем строку **ОстаткиМатериалов** и перетащим ее в окно элементов формы. На вопрос системы «добавить колонки таблицы?» ответим «да».



Немного изменим внешний вид формы.

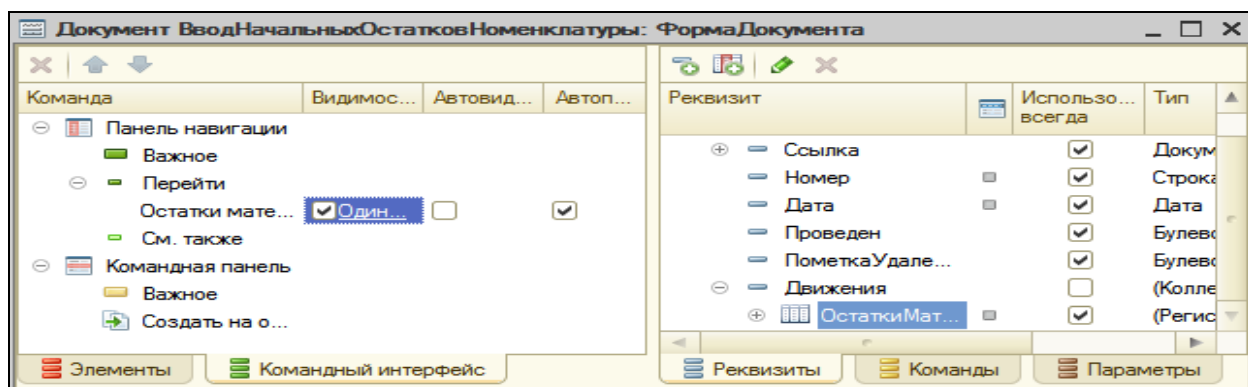
В окне элементов формы добавим группу полей с типом группировки **Горизонтальная** и перетащим в нее поля документа **Номер** и **Дата**. А

также поменяем местами поля таблицы **ДвиженияОстаткиМатериаловНаборСвойств** и **Склад**.



В заключение отредактируем командный интерфейс формы документа, чтобы в панели навигации формы иметь возможность переходить к списку записей регистра **ОстаткиМатериалов**, связанному с документом. Для этого в левом верхнем окне редактора форм перейдем на вкладку **Командный интерфейс**.

В группе **Панель навигации** в подгруппе **Перейти** установим видимость для команды открытия регистра **Остатки материалов**.



В окне редактирования документа **Ввод Начальных Остатков Номенклатуры** установим принадлежность к подсистеме **Бухгалтерия**.

В заключение отредактируем командный интерфейс этой подсистемы – **Общие – Подсистемы – Все подсистемы**.

Выделим в списке подсистем **Бухгалтерию** и в списке команд установим видимость команды **Ввод начальных остатков номенклатуры: создать** в группе **Панель действий.Создать**.

В режиме 1С:Предприятие

Запустим режим отладки и проверим работу нашего документа.

Выполним команду **Ввод начальных остатков номенклатуры** в панели действий раздела **Бухгалтерия**.

Создадим документ для ввода начальных остатков в регистр **Остатки Материалов** и внесем в него следующие данные.

N	Период	Вид движения	Материал	Набор свойств	Склад	Количество
1	02.07.2010 0:00:00	Приход	Кабель электрический		Основной	35,000
2	10.07.2010 0:00:00	Приход	Шланг резиновый		Основной	127,000

Обратите внимание, что дата документа не совпадает с датами отдельных записей, которые мы создаем в движениях документа.

Нажмем **Записать** и в панели навигации перейдем к движениям нашего документа в регистре **Остатки Материалов**.

Период	Регистратор	N..	Материал	Склад	Набор сво...	Количество
+ 02.07.2010 0:00:00	Ввод начальных остатков номенклатуры ...	1	Кабель электрический	Основ...		35,000
+ 10.07.2010 0:00:00	Ввод начальных остатков номенклатуры ...	2	Шланг резиновый	Основ...		127,000

Таким образом, мы добились поставленной цели: с одной стороны, задавая дату документа, мы можем фиксировать момент внесения изменений в записи регистра, с другой стороны – для каждой создаваемой нами записи регистра мы можем указать индивидуальное значение поля **Период**.

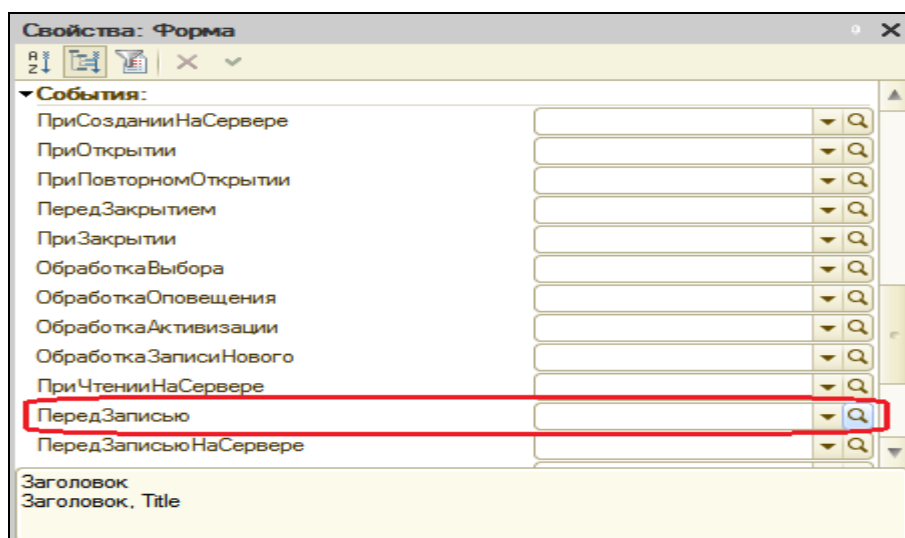
Теперь займемся ужесточением требований к тому, как наш документ формирует записи регистра, и рассмотрим два типичных варианта.

## Программное редактирование записей регистра

### В режиме Конфигуратор

Первое требование будет заключаться в том, что записи регистра должны формироваться той же датой, что и дата документа. Иначе говоря, синхронизируем дату движений с датой документа.

Для этого создадим для формы документа обработчик события **ПередЗаписью** и добавим в него следующий текст.



&НаКлиенте

Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)

**Для Каждого ЗаписьРегистра Из Объект.Движения.ОстаткиМатериалов**

**Цикл**

**ЗаписьРегистра.Период = Объект.Дата;**

**КонецЦикла;**

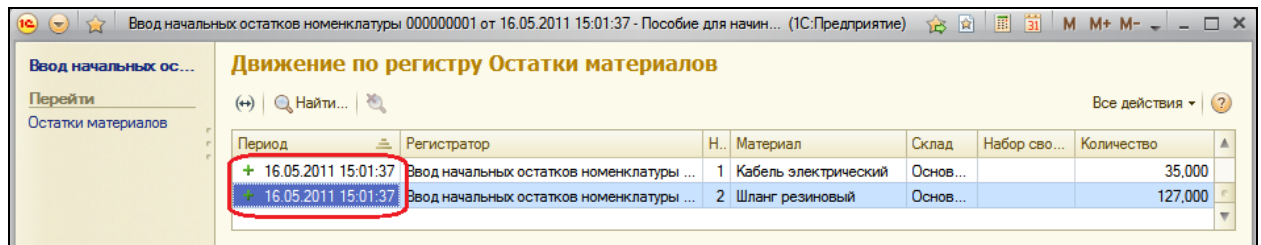
КонецПроцедуры



## В режиме 1С:Предприятие

Запустим отладку, откроем наш документ и нажмем **Записать**.

Открыв движения документа в регистре **ОстаткиМатериалов**, увидим, что значение поля **Период** у всех записей стало равно дате документа.



Период	Регистратор	Н...	Материал	Склад	Набор сво...	Количество
16.05.2011 15:01:37	Ввод начальных остатков номенклатуры ...	1	Кабель электрический	Основ...		35,000
16.05.2011 15:01:37	Ввод начальных остатков номенклатуры ...	2	Шланг резиновый	Основ...		127,000

Можно сказать, что мы достигли поставленной цели, но лишь для интерактивной записи документа. Если программно вызвать метод **Записать()** у объекта нашего документа, он будет записан без участия формы документа. Это значит, что событие **ПередЗаписью** формы документа вызвано не будет и наш код обработчика не сработает.

Чтобы предусмотреть возможность синхронизации периода движений документа с датой документа и в случае программной записи объекта **Документ**, следует использовать обработчик события **ПередЗаписью** объекта **Документ**, а не формы документа. Событие **ПередЗаписью** в случае интерактивной записи документа сначала будет вызвано у формы документа, а затем у объекта **Документ**.

## В режиме Конфигуратор

Вернемся в конфигуратор и удалим из модуля формы добавленный нами текст и создадим обработчик события **ПередЗаписью** в модуле документа **ВводНачальныхОстатковНоменклатуры**.

Для этого откроем на закладке **Прочее** окна редактирования этого объекта модуль объекта и внесем в него следующий текст.

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

    // Определить, нужно ли обновлять дату в движениях
    ОбновитьДатуДвижений = ЭтоНовый() Или
Движения.ОстаткиМатериалов.Модифицированность();
    Если Не ОбновитьДатуДвижений Тогда
        // Проверить, что дата изменилась
        Запрос = Новый Запрос;
        Запрос.УстановитьПараметр("ТекущийДокумент", Ссылка);
        Запрос.Текст =
            "ВЫБРАТЬ
            | Дата
            | ИЗ
```

```

| Документ.ВводНачальныхОстатковНоменклатуры
|ГДЕ Ссылка = &ТекущийДокумент";

Выборка = Запрос.Выполнить().Выбрать();
Выборка.Следующий();
ОбновитьДатуДвижений = Выборка.Дата <> Дата;
КонецЕсли;

// Установить всем новую дату, если нужно
Если ОбновитьДатуДвижений Тогда
    Если Не Движения.ОстаткиМатериалов.Выбран() И
        Не Движения.ОстаткиМатериалов.Модифицированность() Тогда
        Движения.ОстаткиМатериалов.Прочитать();
    КонецЕсли;
    Для Каждого ЗаписьРегистра Из Движения.ОстаткиМатериалов Цикл
        ЗаписьРегистра.Период = Дата;
    КонецЦикла;
КонецЕсли;

КонецПроцедуры

```

Как вы видите, в этом случае обработчик содержит больше кода за счет дополнительных проверок, которые выполняются в результате возможности как интерактивной, так и программной записи объекта.

Поясним содержание обработчика. Если записывается новый документ или были изменены его движения, следует обновить дату движений. В противном случае мы считаем запросом дату документа из базы и сравниваем ее с датой, установленной у записываемого объекта. Если даты разные, также следует обновить дату движений.

Перед установкой даты мы проверяем, был ли прочитан набор записей в свойстве **Движения** объекта и изменился ли он. Если оба этих условия ложны, значит набор записей в свойстве **Движения** объекта пуст, и это состояние не связано с его изменением. В этом случае, чтобы предотвратить ошибочное удаление записей в регистре (перезаписать пустым набором записей), мы предварительно читаем движения из регистра в набор записей в свойстве **Движения**.

Затем, как и в предыдущем случае, устанавливаем нужную дату для всех записей этого набора. При выполнении записи объекта документ этот набор будет записан в регистр накопления.

## В режиме 1С:Предприятие

Запустим режим отладки и убедимся, что указав новую дату для нашего документа и записав его, мы получим движения в регистре накопления с новой датой.

В процессе записи нашего документа можно управлять не только периодом записей регистра накопления, но и значениями других полей регистра.

Например, по аналогичному принципу может быть создан документ **Операция**, позволяющий вводить ручные операции в регистр бухгалтерии.

При этом вероятно, что кроме управления периодом записей регистра вам потребуется управлять значением поля **Активность** (включать и выключать проводки документа) и т.д.

### Где создавать обработчики событий

Выбор обработчика, в который будет помещен текст процедуры, зависит от логики работы создаваемого объекта. Если конфигурация не предусматривает программной записи объекта, можно выбрать обработчик модуля формы. Если предполагается и программная модификация объекта, следует выбирать обработчик модуля объекта.

Заметьте, что оба этих способа не исключают модификацию записей регистра через объект **Регистр<...>НаборЗаписей.<имя регистра>**. Поэтому если логика конфигурации подразумевает возможность программной модификации объекта НаборЗаписей, код обработки следует помещать в обработчик события набора записей. Все попытки изменить данные регистра будут сведены к записи именно набора записей.

### Контрольные вопросы

- ✓ Для чего предназначен документ для ввода начальных остатков и как его создать.
- ✓ Как программно изменить значение регистра при вводе начальных остатков.
- ✓ В каких случаях использовать модуль формы, а в каких – модуль объекта для размещения обработчиков событий.

## Практическая работа № 21

### *Список пользователей и их роли (1:00)*

#### Что такое Роль

После создания всех основных объектов конфигурации, можно приступить к определению ролей пользователей.

Очевидно, что не всем пользователям нужно разрешать просматривать и изменять всю базу данных. Кладовщику нельзя просматривать и изменять данные о сотрудниках и бухгалтерии, например.

Для описания подобных разрешений используются объекты конфигурации *Роль*.

Как правило, роли создаются отдельно для каждого вида деятельности и каждому пользователю системы ставится в соответствие одна или несколько ролей. Если пользователю поставлено в соответствие несколько ролей, то предоставление доступа будет осуществляться по следующему алгоритму:

- Если хотя бы в одной роли есть разрешение, то доступ будет открыт;
- Если во всех ролях разрешение отсутствует, то доступ будет закрыт.

#### Создание ролей

#### В режиме Конфигуратор

При создании ролей исходят, как правило, из того, какие полномочия требуются различным группам пользователей на доступ к информации. Для этого мы воспользуемся подсистемами, которые значительно облегчат нашу задачу.

#### Администратор

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

Раскроем ветвь **Общие** дерева объектов конфигурации. Добавим новый объект **Роль** с именем **Администратор**.

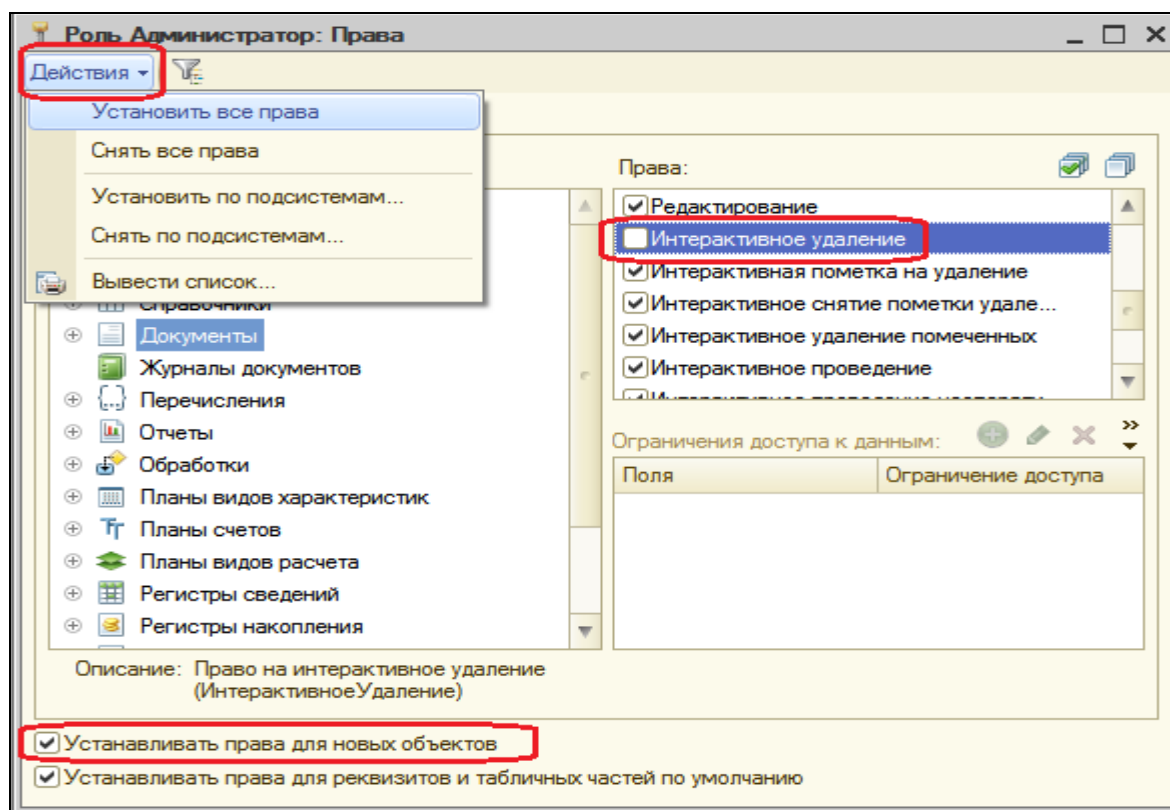
Откроется окно редактирования этой роли. Слева, в списке объектов, перечислены все объекты и виды объектов конфигурации, а справа, в окне прав, - доступные права для выбранного объекта или видов объектов конфигурации.

Администратор должен иметь права на все объекты и все виды объектов. Для этого выполним команду **Действия – Установить все права** в командной панели окна. После этого все права для всех объектов будут помечены.

Снимите разрешение на интерактивное удаление для всех объектов. Это нужно, чтобы администратор случайно не мог удалить какой-либо объект базы данных.

Для этого пройдемся по всем видам объектов конфигурации (Справочники, Документы и т.д.) и снимем отметку с команды **Интерактивное удаление**.

Для того, чтобы наш **Администратор** мог работать с объектами, которые мы будем создавать после расстановки прав, зададим для него параметр **Устанавливать права для новых объектов**. На этом создание роли Администратор закончено.

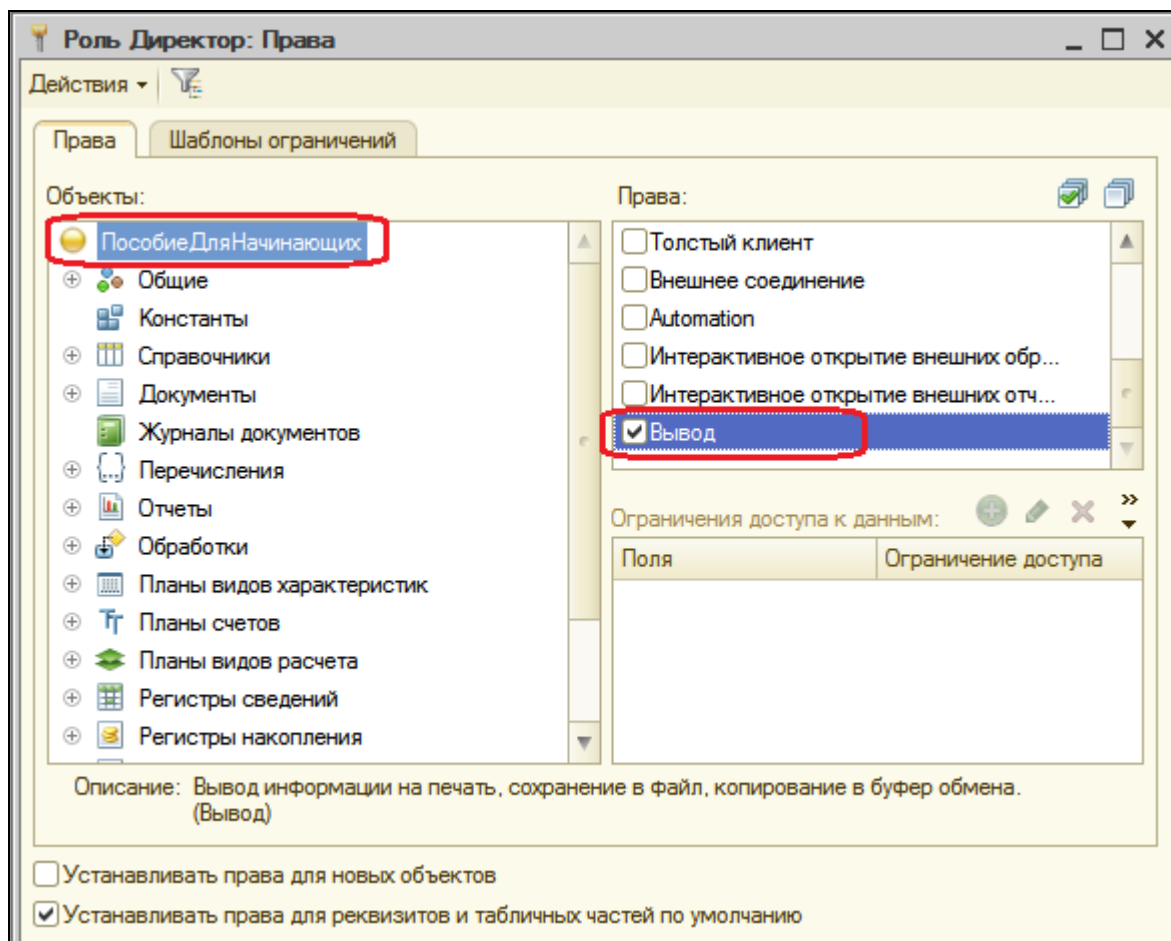


## Директор

Создадим новый объект конфигурации **Роль** с именем **Директор**.

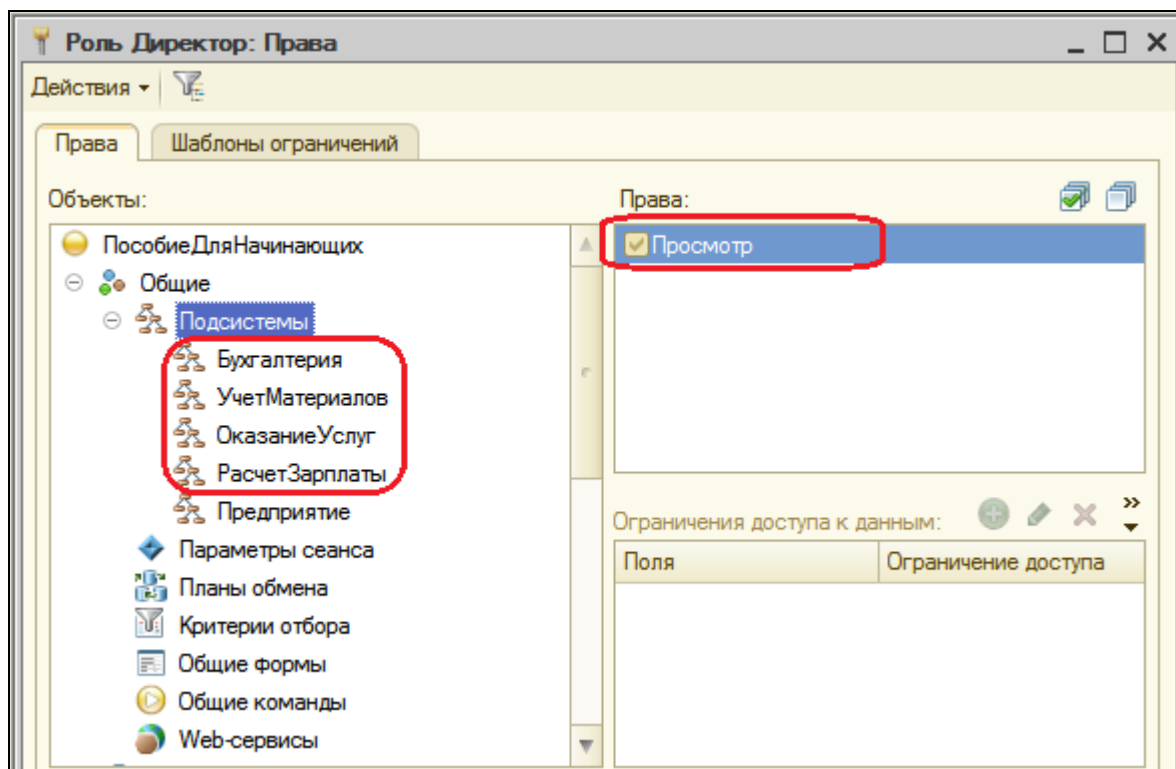
Нас устраивает, что у новой роли нет прав на доступ ко всем объектам, за исключением тех видов объектов, для которых не создано ни одного объекта. Для таких видов останутся установленными полные права.

Установим право **Вывод** для всей конфигурации.



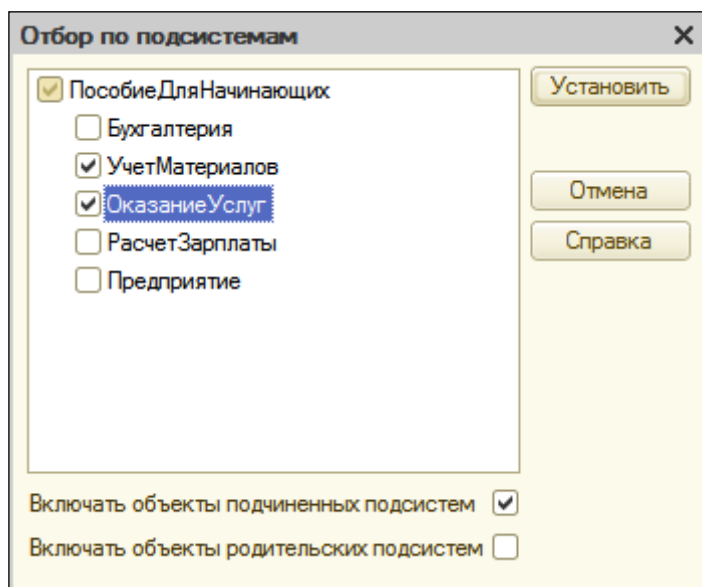
Теперь остается пройти по всем видам объектов конфигурации и установить для них право **Просмотр** (права **Чтение** и **Использование** при этом установятся автоматически).

Затем раскроем ветвь **Общие**, выделим ветвь **Подсистемы** и отметим право **Просмотр** у всех подсистем, кроме подсистемы **Предприятие**. Тем самым мы предоставим директору возможность просматривать все данные информационной базы, но исключим из его интерфейса все действия, которые по логике нашей конфигурации не относятся к прикладной ее части.



## Мастер

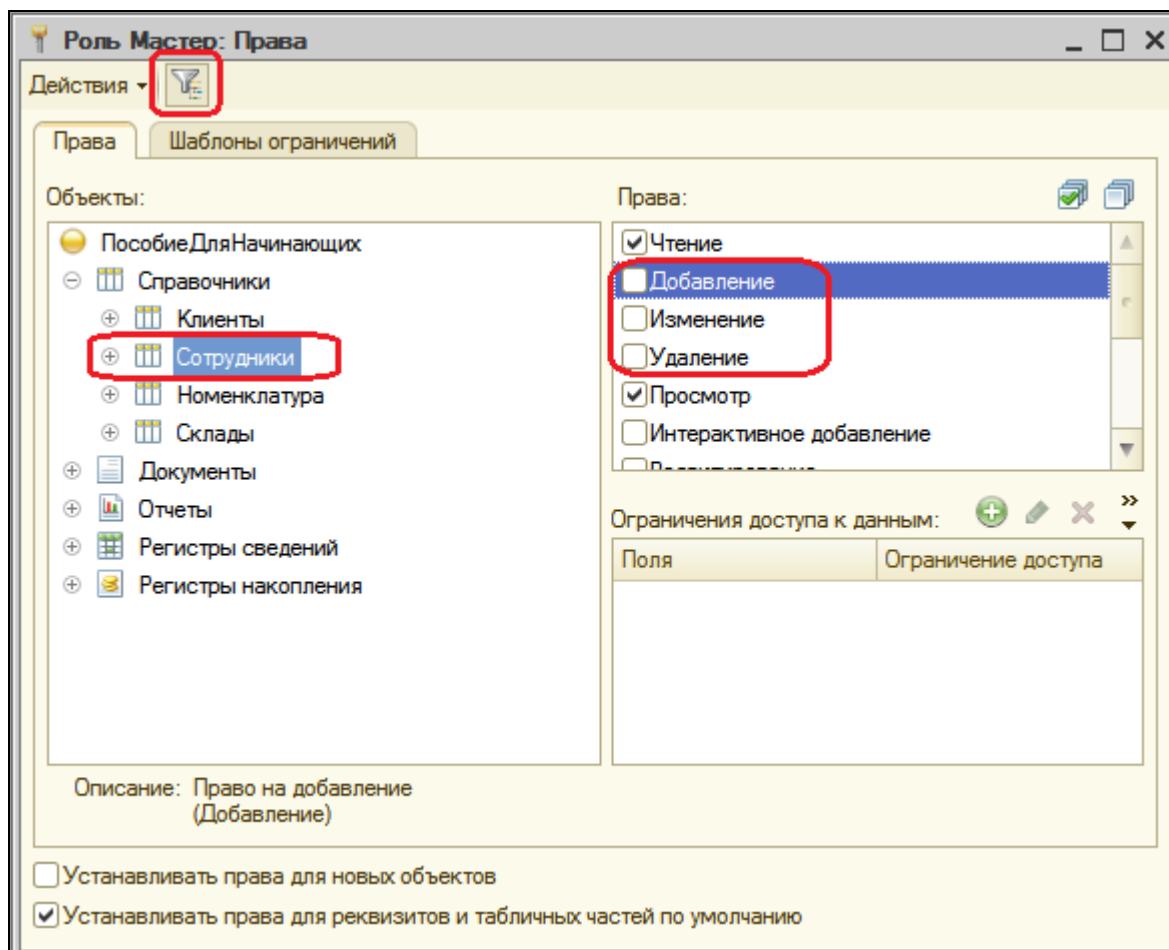
Снова добавим роль с именем **Мастер**. Выполним команду **Действия – Установить по подсистемам** и выберем подсистемы **УчетМатериалов** и **ОказаниеУслуг**.



В результате будут установлены все права на объекты конфигурации, относящиеся к данным подсистемам.

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

В частности для справочника **Сотрудники** мы запретим права **Добавление, Изменение, Удаление**.



Обратите внимание, что при запрете права **Добавление** исчезла отметка и у права **Интерактивное добавление**, т.к. оно является уточнением права **Добавление**. Точно также уточненные права запрещаются и при отмене прав на изменение и удаление.

Теперь пройдем по всем видам объектов и снимем у всех право **Интерактивное удаление**.

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

- Справочник **ВариантыНоменклатуры**,
- Справочник **ДополнительныеСвойстваНоменклатуры**,
- План видов характеристик **СвойстваНоменклатуры**,
- Регистр сведений **ЗначенияСвойствНоменклатуры**.

Эти объекты мы не привязывали ни к каким подсистемам, но они будут нужны для работы с характеристиками номенклатуры.



## Расчетчик

В заключение нам осталось создать две роли: **Бухгалтер** и **Расчетчик**.

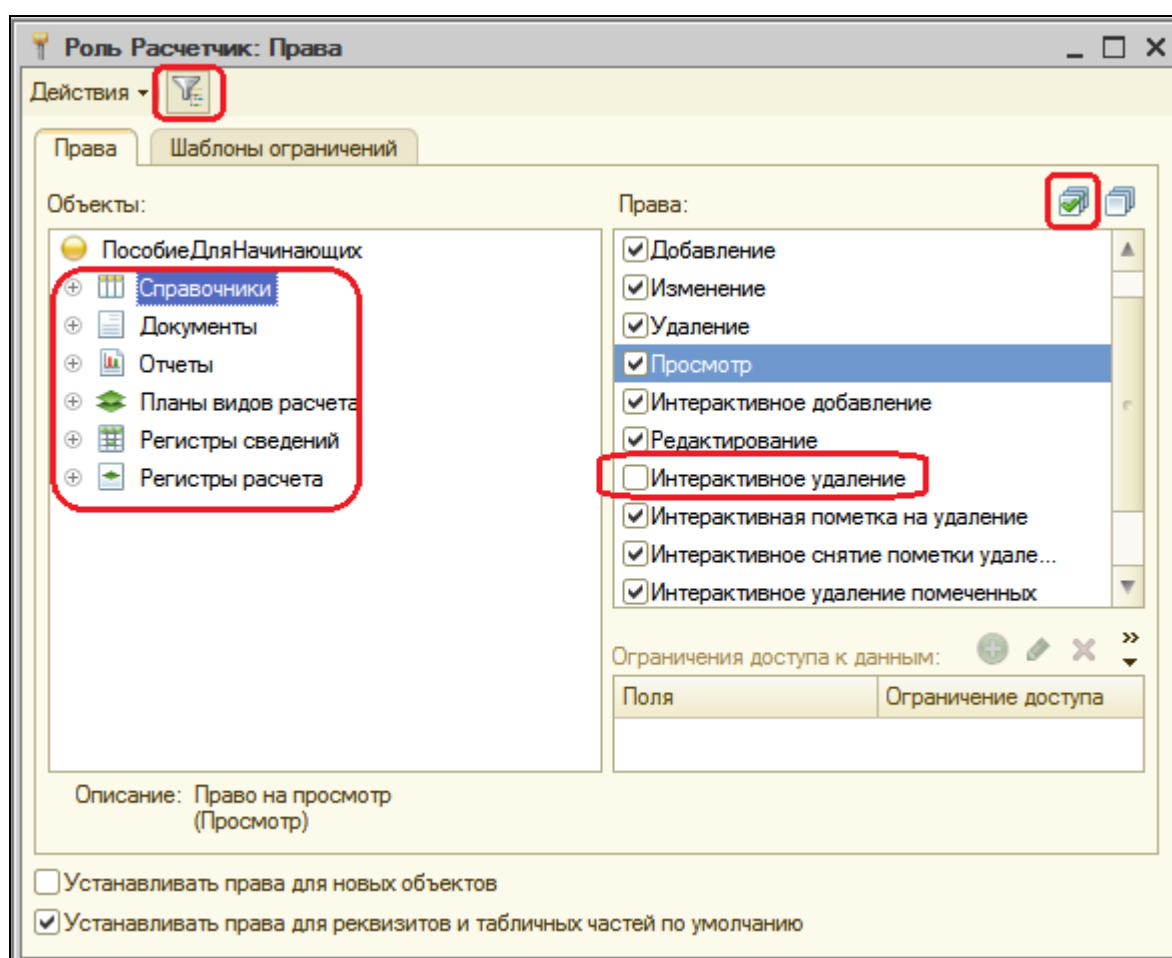
Мы разделим права по расчету зарплаты и по ведению бухгалтерского учета.

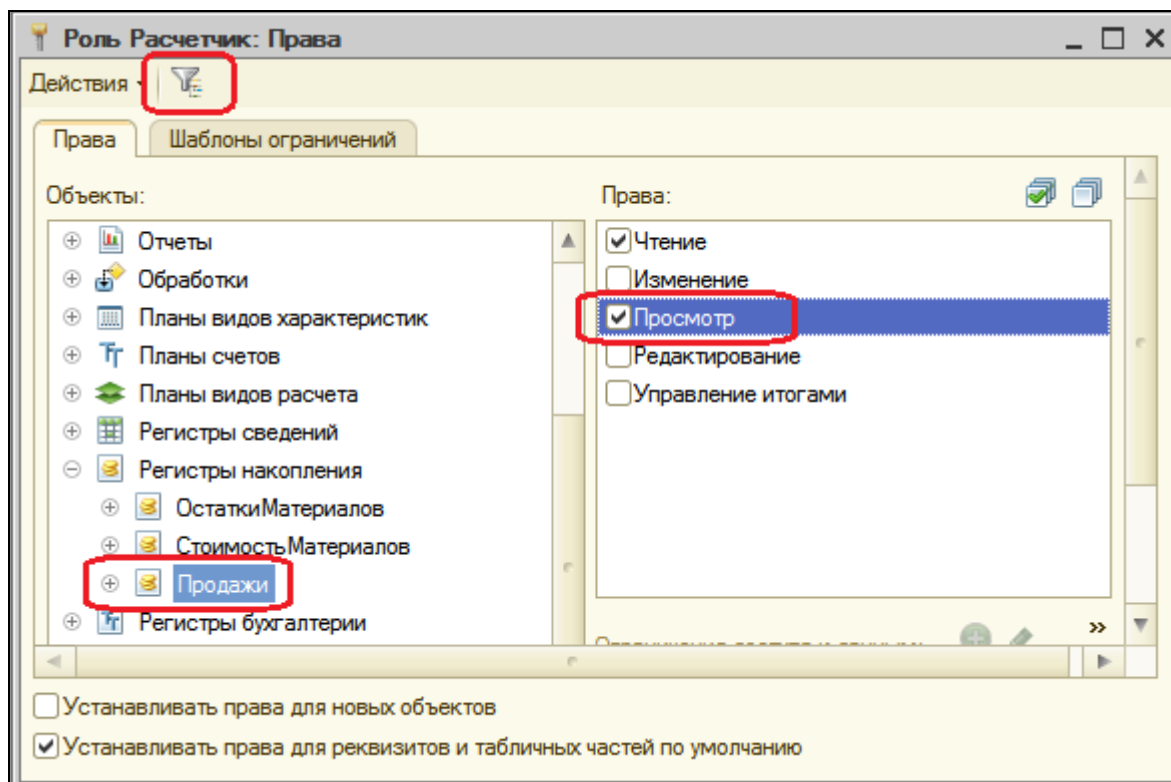
Дело в том, что в нашей фирме есть бухгалтер и его помощник. Помощник бухгалтера будет занят в основном расчетом зарплаты, но иногда это делает и главный бухгалтер.

Поэтому главному бухгалтеру необходимо будет назначить обе роли, а помощнику – только роль **Расчетчик**.

Создадим новую роль **Расчетчик**. В окне редактирования прав установим все права по подсистеме **РасчетЗарплаты**, пользуясь фильтром (и не забудем запретить интерактивное удаление).

А также установим право **Просмотр** для регистра накопления **Продажи**, отключив фильтр.

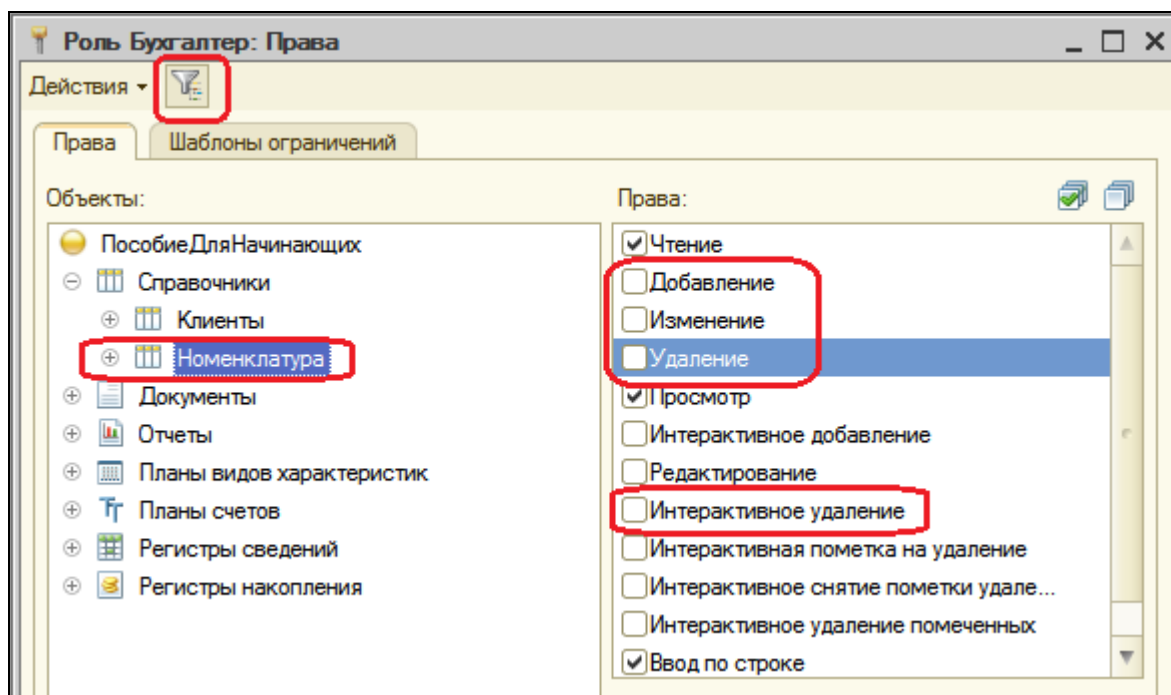




## Бухгалтер

Создадим роль с именем **Бухгалтер**. В окне редактирования прав установим их по подсистеме **Бухгалтерия**. После этого отфильтруем список объектов по этой подсистеме и для справочника **Номенклатура** запретим добавление, изменение и удаление.

Также запретим **интерактивное удаление** для всех объектов.



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

- Справочник **Субконто**,
- Регистр бухгалтерии **Управленческий**.

Установим право **Просмотр** для следующих объектов:

- Справочник **Склады**,
- Справочник **ВариантыНоменклатуры**,
- Справочник **ДополнительныеСвойстваНоменклатуры**,
- План видов характеристик **СвойстваНоменклатуры**,
- Регистр сведений **ЗначенияСвойствНоменклатуры**.

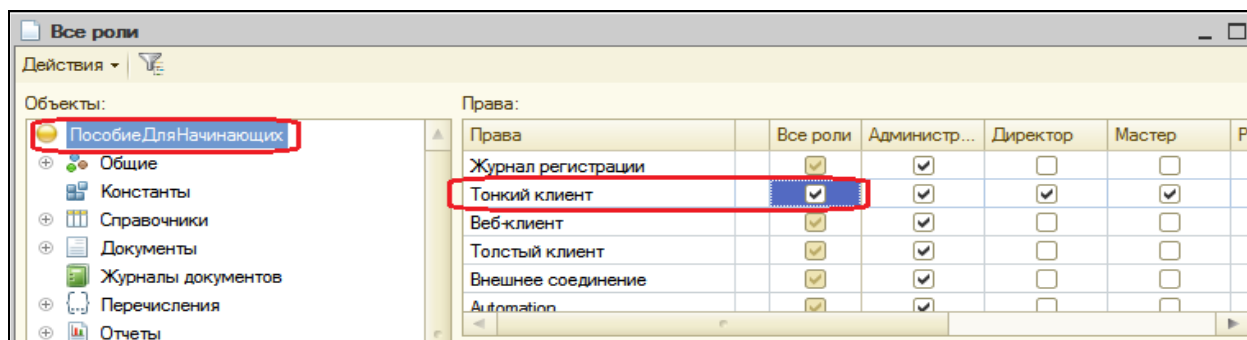
### Права на запуск клиентских приложений

В заключение установим права на запуск клиентского приложения для каждой роли. Для этого воспользуемся другим, более удобным инструментом – редактором **Все роли**.

В дереве объектов конфигурации выделим ветку **Роли** и в контекстном меню выполним команду **Все роли**.

В списке объектов конфигурации выделим корень и для всех ролей установим право **Тонкий клиент**.

Тем самым мы разрешили всем пользователям подключаться к информационной базе с помощью тонкого клиента. Администратор же имеет возможность подключаться и с помощью других клиентских приложений. Это может понадобиться ему, например, для создания планировщика заданий, о котором шла речь в разделе «Планировщик заданий».




Список прав для каждой роли можно получить, выполнив в окне редактирования прав команду **Действия – Вывести список**. Аналогичный список, но только для всех ролей, которые есть в конфигурации, можно получить из редактора **Все роли**.

Это пример списка прав для роли **Бухгалтер**.

Новый5			
1	2	3	
1	Объекты	Права	Бухгалтер
2	Конфигурация.ПособиеДляНачинающих		
3	Все права		
4	Административные функции		Нет
5	Обновление конфигурации базы данных		Нет
6	Монопольный режим		Нет
7	Активные пользователи		Нет
8	Журнал регистрации		Нет
9	Тонкий клиент		Да
10	Веб-клиент		Нет
11	Толстый клиент		Нет
12	Внешнее соединение		Нет
13	Automation		Нет
14	Интерактивное открытие внешних обработок		Нет
15	Интерактивное открытие внешних отчетов		Нет
16	Вывод		Нет
17	Общие		
18	Подсистемы		
19	Просмотр		Нет
20	Подсистема.Бухгалтерия		
21	Просмотр		Да
22	Подсистема.УчетМатериалов		
23	Просмотр		Нет
24	Подсистема.ОказаниеУслуг		
25	Просмотр		Нет
26	Подсистема.РасчетЗарплаты		
27	Просмотр		Нет
28	Подсистема.Предприятие		
29	Просмотр		Нет

## Добавление новых пользователей

### В режиме Конфигуратор

Прежде чем мы приступим к созданию пользователей, необходимо выполнить обновление конфигурации базы данных (**Конфигурация – обновить конфигурацию базы данных (F7)** ) поскольку пользователю можно поставить в соответствие только те роли, которые существуют в конфигурации.

После обновления, выполним команду главного меню **Администрирование – Пользователи**. Откроется список пользователей системы. Пока что он пуст, поэтому добавим нового пользователя (**Действия – Добавить**) или нажмем кнопку **Добавить**.

## Внимание!

Если вы используете учебную версию платформы, то возможность задания паролей пользователей и аутентификация операционной системы будут недоступны. Это ограничения учебной версии.

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

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

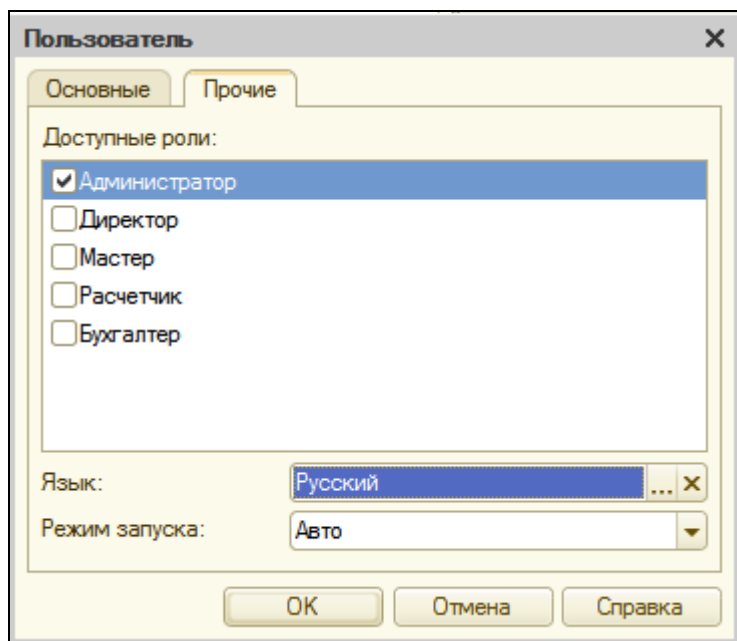
**Аутентификация средствами 1С:Предприятия** подразумевает, что после запуска системы пользователю будет предложено выбрать имя пользователя и ввести пароль. Если введенный пароль соответствует сохраненному в системе для этого идентификатора пользователя, система открывается с правами этого пользователя.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

**Аутентификация операционной системы** подразумевает, что при запуске системы 1С:Предприятие 8 от пользователя не требуется никакой дополнительной информации. Система определяет, под каким пользователем запущена операционная система и затем обращается к своему списку пользователей. Если она находит в нем пользователя с

именем текущего пользователя операционной системы, то информационная база открывается с правами этого пользователя.

Приступим к созданию пользователей. Зададим имя пользователя **Администратор**, полное имя тоже **Администратор**. Перейдем на закладку **Прочие**.

Отметим роль **Администратор** и язык конфигурации выберем **Русский**.

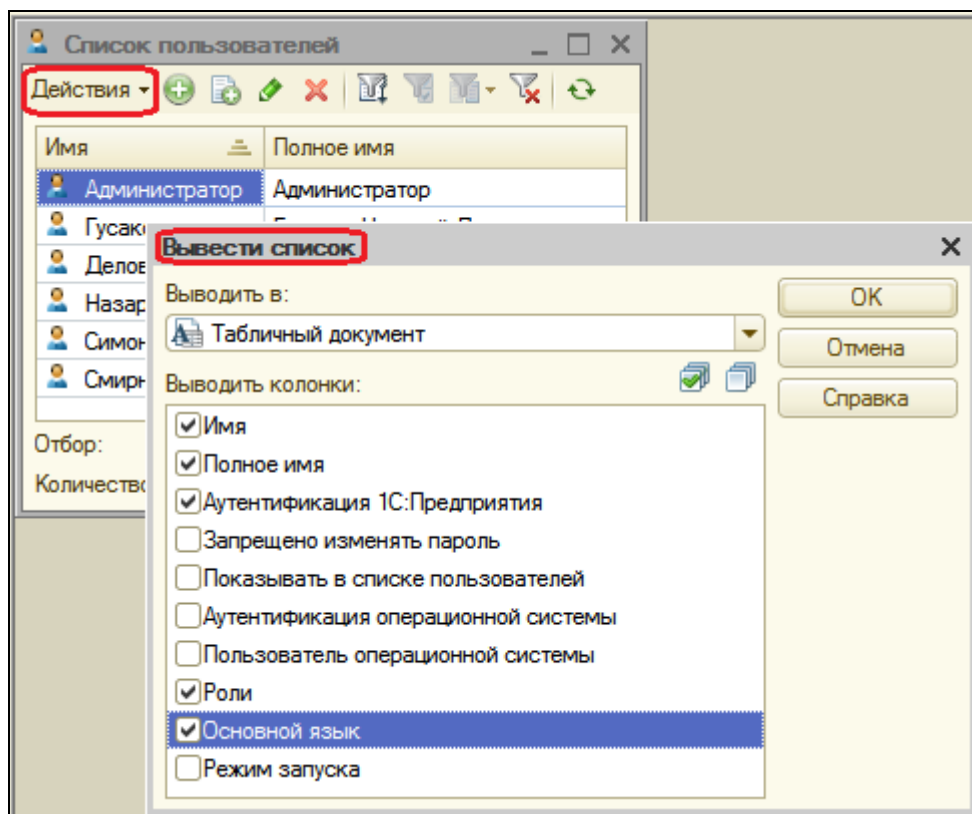


Нажмите ОК.

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

Если некоторые колонки, например, **Роли**, не видны в списке пользователей, можно выполнить настройку списка **Действия – Настройка списка** и добавить нужные колонки (или при выводе списка сразу поставить нужные галочки).

Список пользователей можно получить, выполнив команду **Действия – Вывести список**.



	1	2	3	4	5	6
1	Имя	Полное имя	Аутентификация 1С:Предприятия	Роли	Основной язык	
2	Администратор	Администратор	Истина	Администратор	Русский	
3	Гусак	Гусак Николай Дмитриевич	Истина	Мастер	Русский	
4	Деловой	Деловой Иван Сергеевич	Истина	Мастер	Русский	
5	Назарова	Назарова Инна Семеновна	Истина	Расчетчик, Бухгалтер	Русский	
6	Симонов	Симонов Валерий Михайлович	Истина	Мастер	Русский	
7	Смирнова	Смирнова Эльвира Денисовна	Истина	Расчетчик	Русский	
8						
9						
10						
11						
12						

## Ограничение доступа к данным на уровне записей и полей базы данных

Для более точного ограничения доступа к БД в системе 1С:Предприятие 8 используется механизм ограничения доступа на уровне записей и полей базы данных. Этот механизм позволяет для четырех основных прав (чтение, добавление, изменение, удаление) уточнить, какие же именно данные информационной базы будут доступны пользователю.

Такое уточнение записывается на специальном языке, являющемся подмножеством языка запросов.

Далее, на примере документа **Начисления сотрудникам**, мы рассмотрим небольшой пример, когда мастерам нужно дать возможность просмотреть начисленную им зарплату, но руководство запрещает им доступ к информации о начисленной премии.

Другими словами, мастерам нужно запретить просмотр тех документов **Начисления сотрудникам**, в которых есть записи о начислении премии.

## В режиме Конфигуратор

Для решения этой задачи сначала установим для роли **Мастер** право **Просмотр** для документа **НачисленияСотрудникам**.

Поскольку этот документ принадлежит подсистеме **РасчетЗарплаты**, дадим право на просмотр этой подсистемы.

Также дадим права на просмотр справочника **ВидыГрафиковРаботы** и плана видов расчета **Основные начисления**, т.к. ссылки на эти объекты используются в документах **НачисленияСотрудникам**.

Вернемся к редактированию прав роли **Мастер**.

Как мы видим, при установке права **Просмотр** право **Чтение** документа **НачисленияСотрудникам** установилось автоматически. Выделим его.

В правой нижней части экрана находится поле **Ограничение доступа к данным**. Нажмем кнопку **Добавить**.

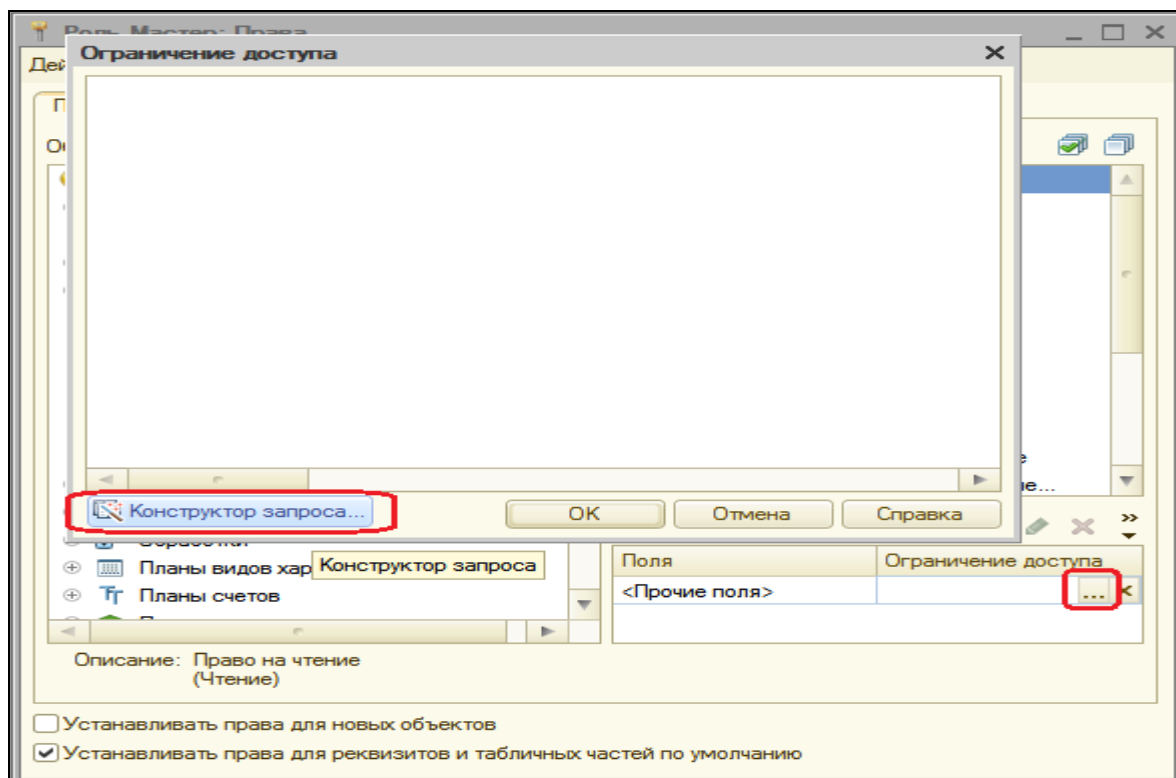
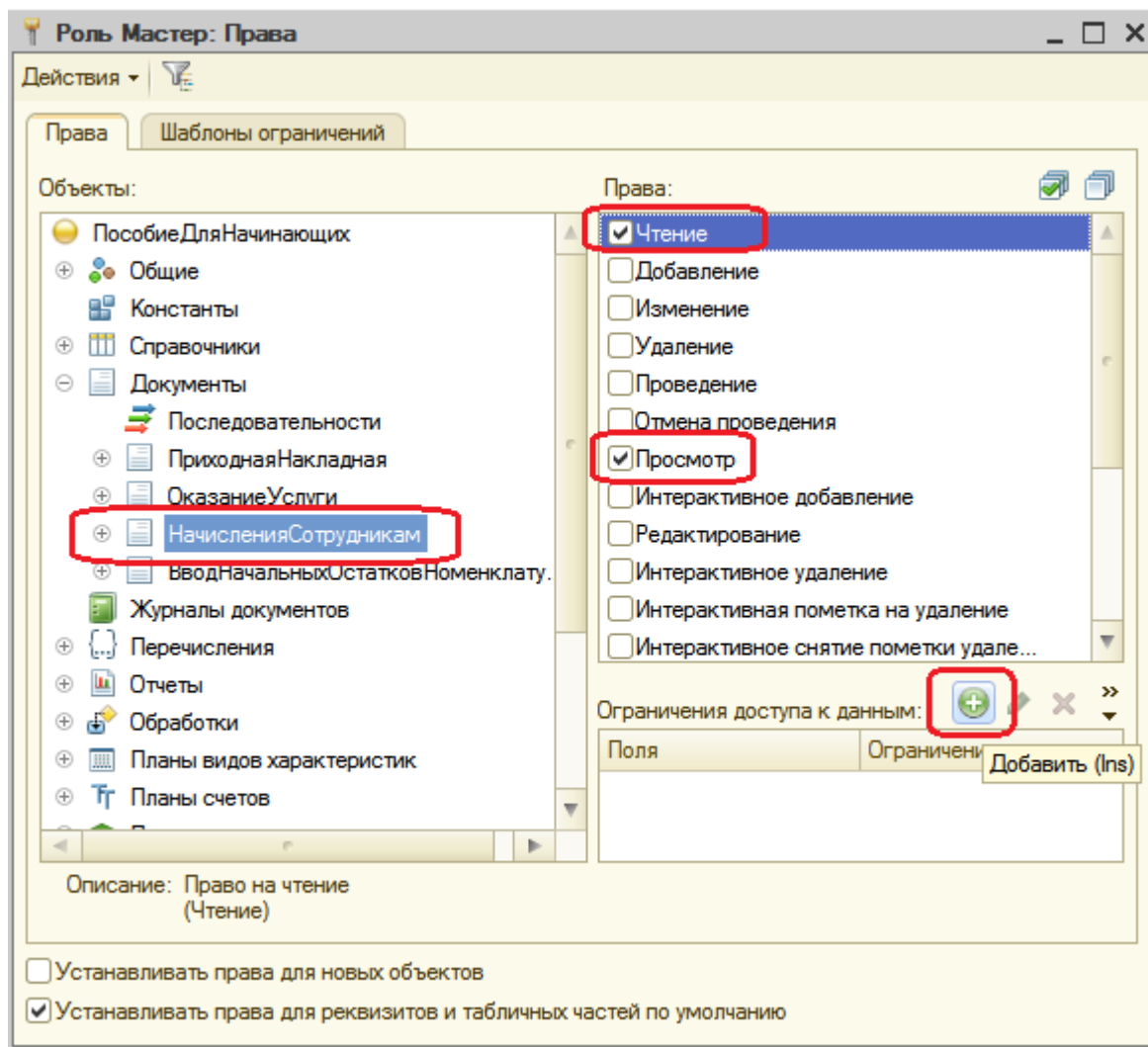
Мы хотим запретить доступ ко всем полям документа **Начисление сотрудникам**.

Поэтому мы не будем выбирать поля, а нажмем кнопку выбора в поле **Ограничение доступа**.

Откроется окно **Ограничение доступа**, в котором можно задать текст на специальном языке. Для облегчения работы воспользуемся конструктором запросов. Нажмем кнопку **Конструктор запроса**.

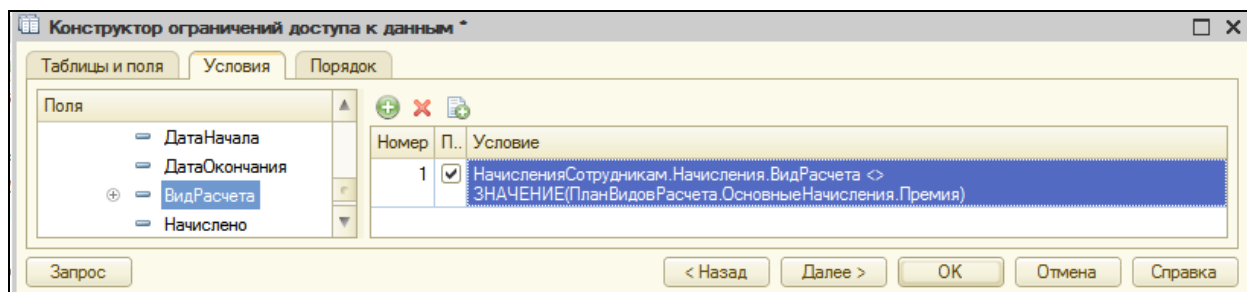
Таблица **НачисленияСотрудникам** автоматически попала на закладку **Таблицы и поля**, а конструктор открылся на вкладке **Условия**.



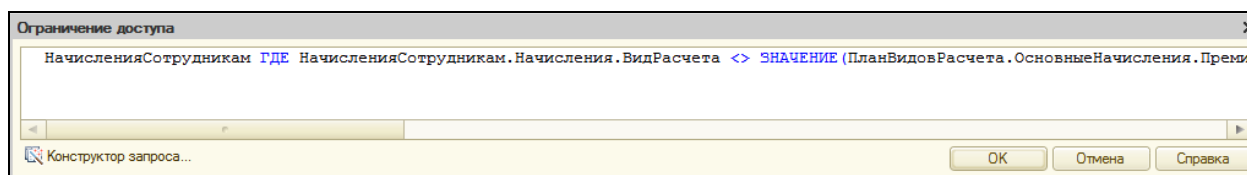


Перенесем в список условий поле **ВидРасчета** табличной части **Начисления**, поставим флажок **Произвольное** и заполним поле условия:

НачисленияСотрудникам.Начисления.ВидРасчета <>  
ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)

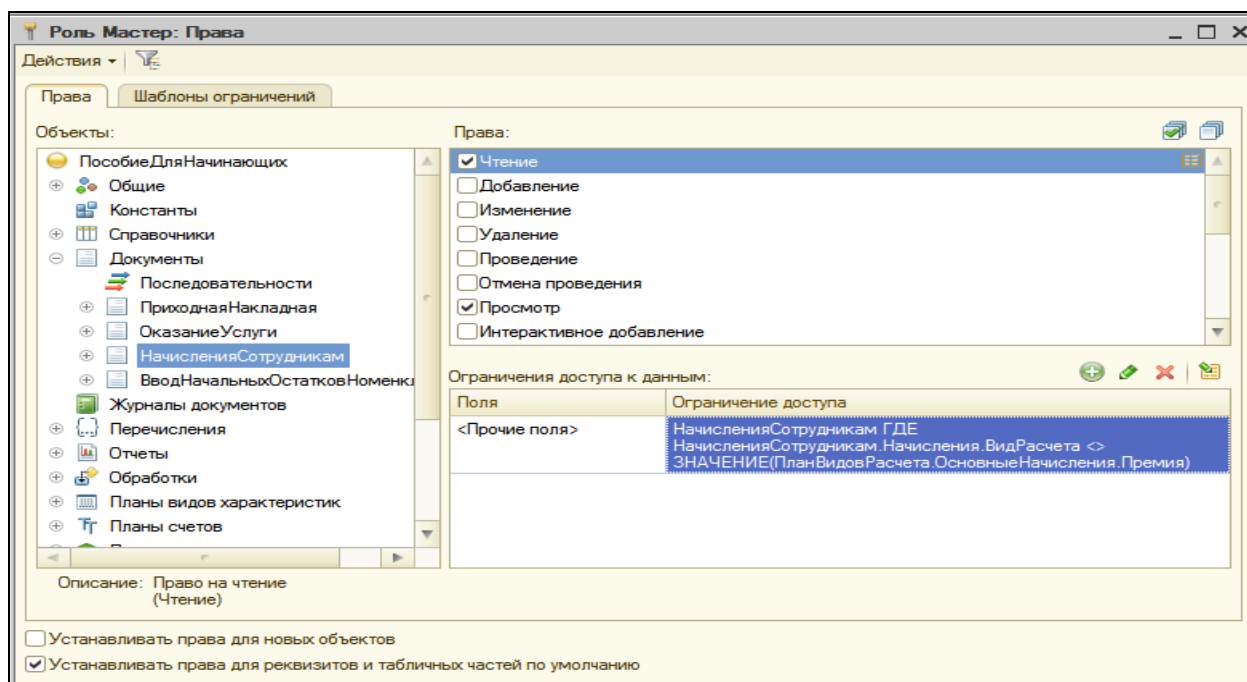


Нажмите OK.



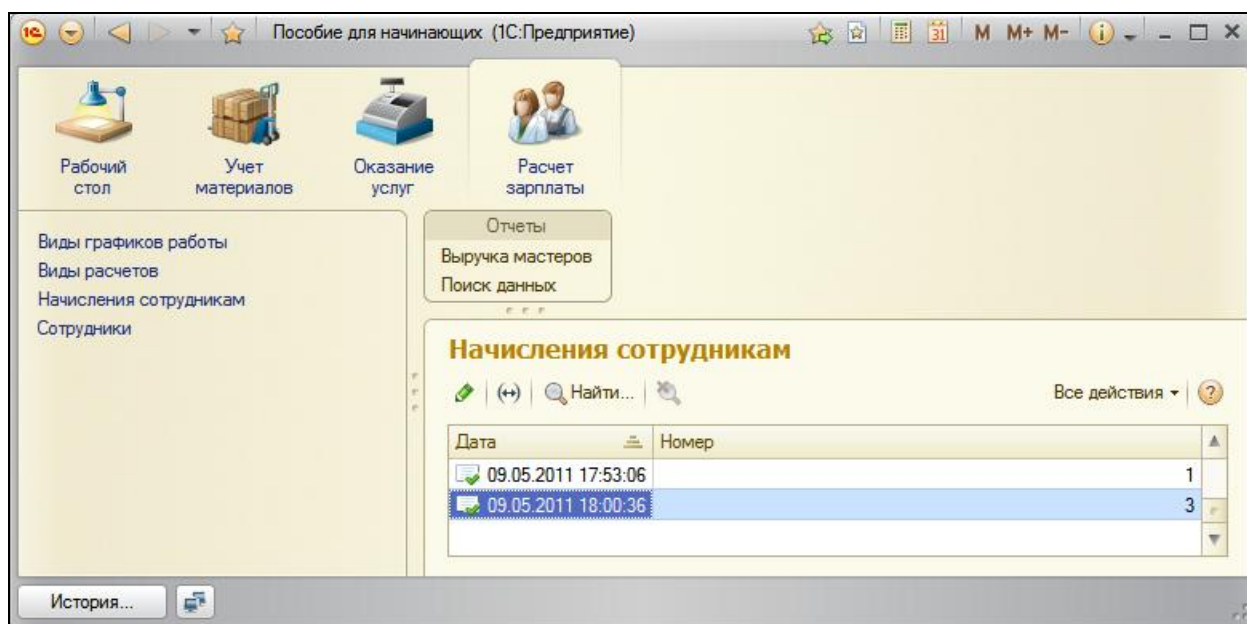
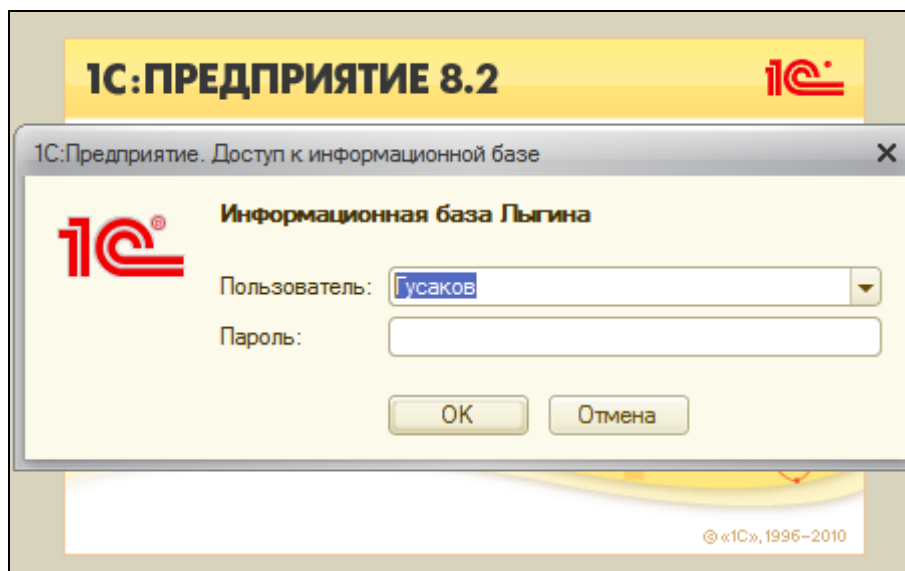
Текущий пользователь имеет право прочитать или изменить некоторый объект базы данных только в том случае, если ограничение доступа предоставляет ему такое право. Т.е. когда условие ограничения истинно.

Окно ограничений доступа к данным для роли **Мастер** будет выглядеть так:



## В режиме 1С:Предприятие

Обновим информационную базу и запустим 1С:Предприятие для пользователя с ролью **Мастер**, например, для пользователя **Гусаков**. В разделе **Расчет зарплаты** откроем список документов **Начисления Сотрудникам**.



Как мы видим, показаны только три подсистемы и только документы №1 и №3, т.к. в документе №2 начисляется премия.

Немного усложним задачу. Мы не хотим, чтобы мастер видел начисленные премии, но и не хотим скрывать от него факт существования такого документа.

Другими словами, в списке документов мастер должен его видеть, но не должен иметь возможность открыть его.

## В режиме Конфигуратор

Вернемся в конфигуратор и посмотрим на наше ограничение.

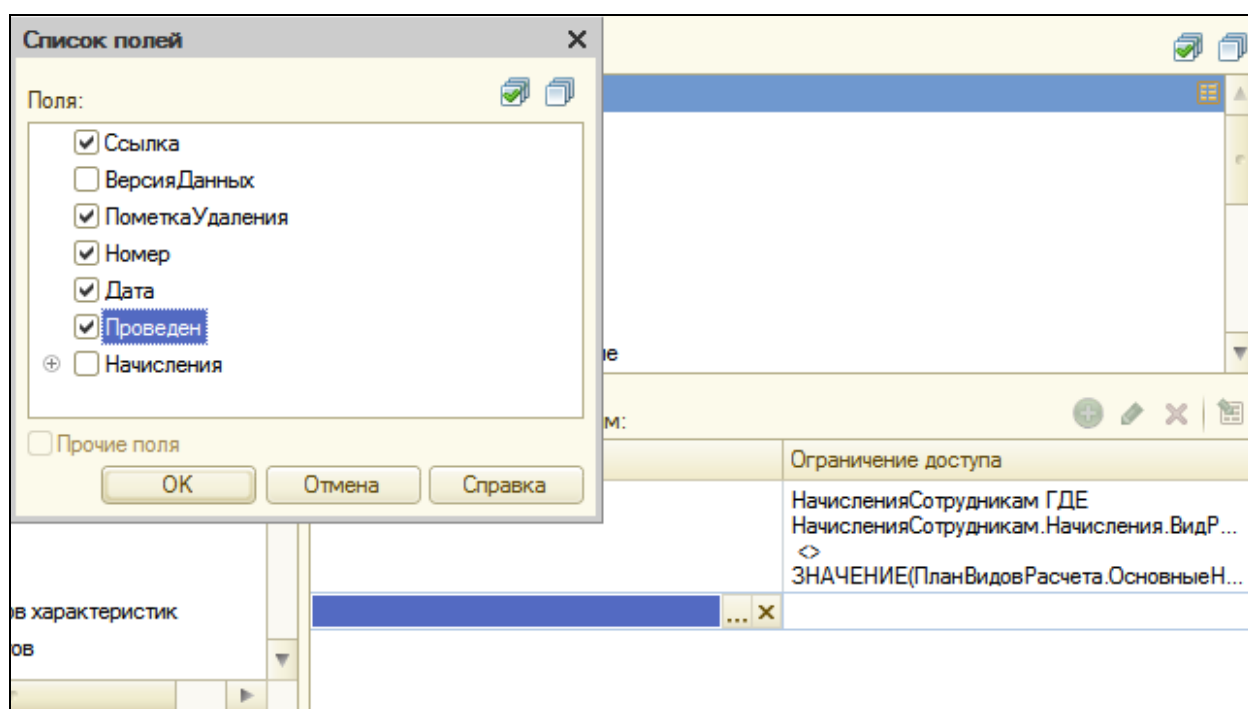
Мы не задавали никаких полей, поэтому ограничение применяется ко всем полям документа. Поэтому сейчас разрешим читать те поля, которые необходимы для отображения документа в списке.

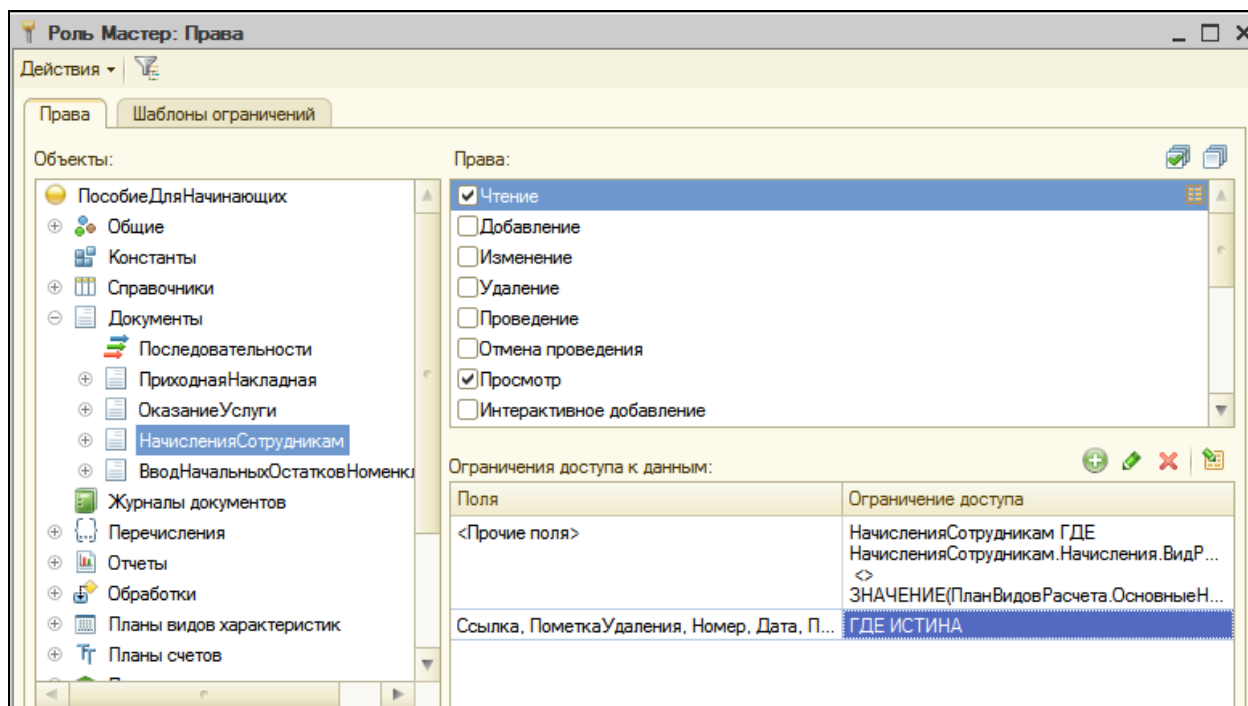
Но т.к. существующее условие на прочие поля мы удалять не будем, то открыть документ, как и раньше, можно будет только в том случае, если в его табличной части есть виды расчета, отличные от вида **Премия**.

Добавим к ограничениям доступа еще одно условие. В списке полей выберем поля:

- **Ссылка,**
- **ПометкаУдаления,**
- **Номер,**
- **Дата,**
- **Проведен.**

В ограничении доступа напишем ГДЕ ИСТИНА.



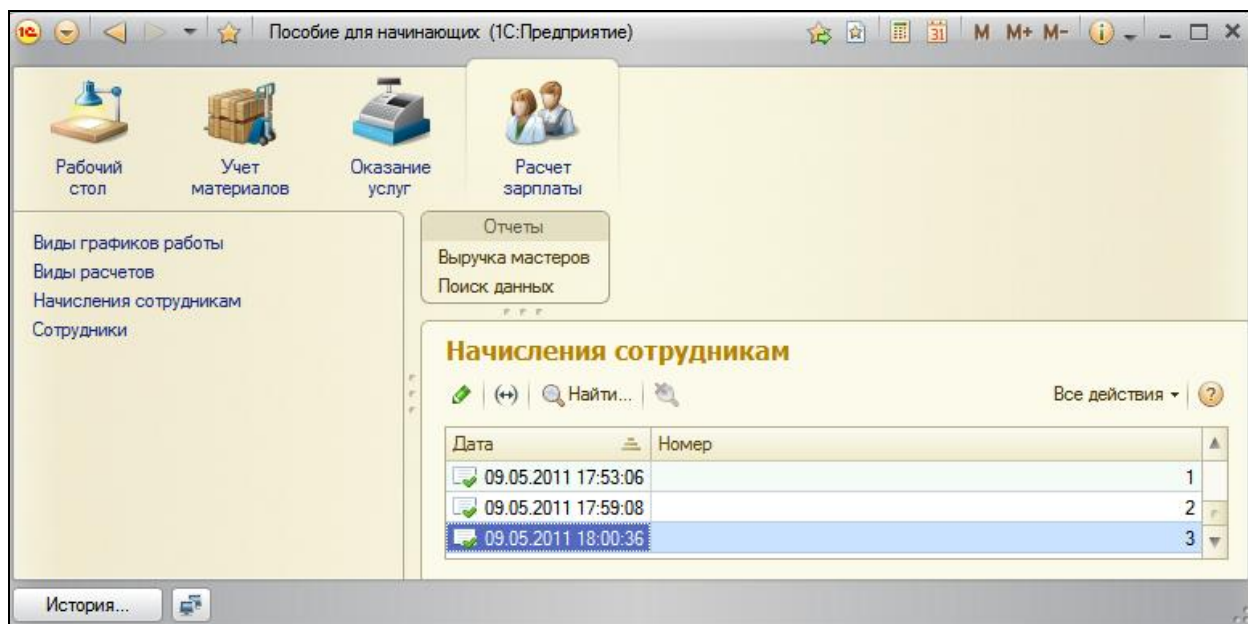


Закроем окно редактирования прав.

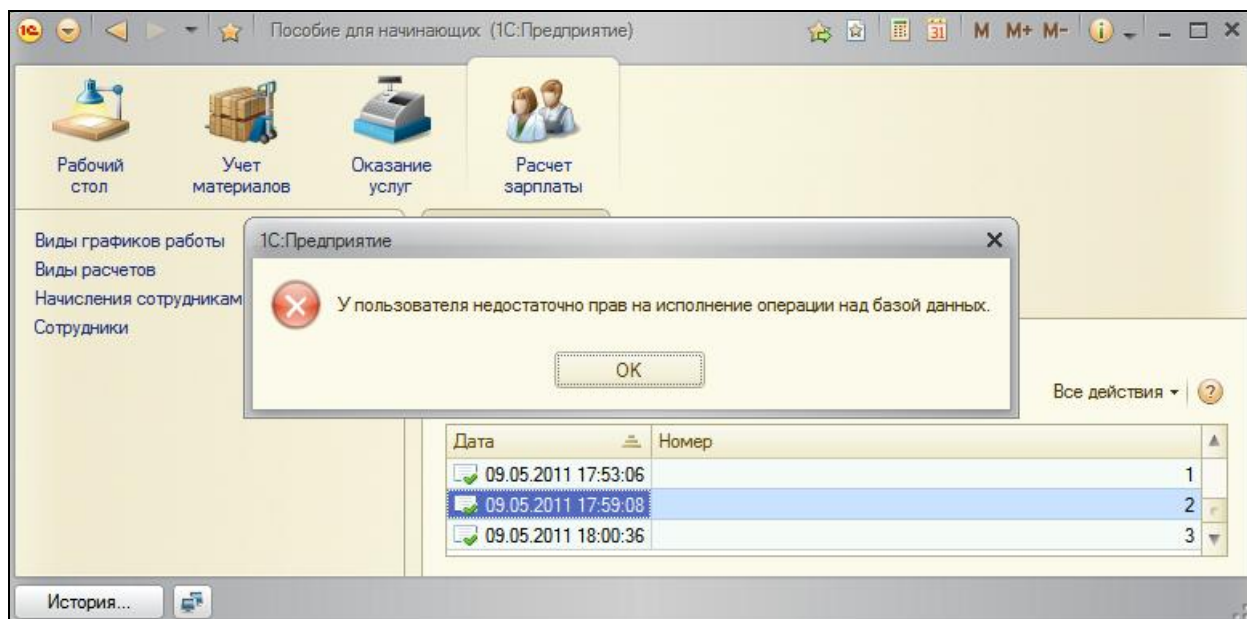
В режиме 1С:Предприятие

Обновим информационную базу и запустим отладку для пользователя с ролью **Мастер Гусаков**.

В разделе **Расчет зарплаты** откроем список документов **НачисленияСотрудникам**.



В списке документов мы увидим все документы начислений. Документы №1 и №3 мы сможем открыть и просмотреть, но при попытке открыть документ №2 мы получим сообщение о нарушении прав доступа.



Т.е. мы добились того, чего хотели.

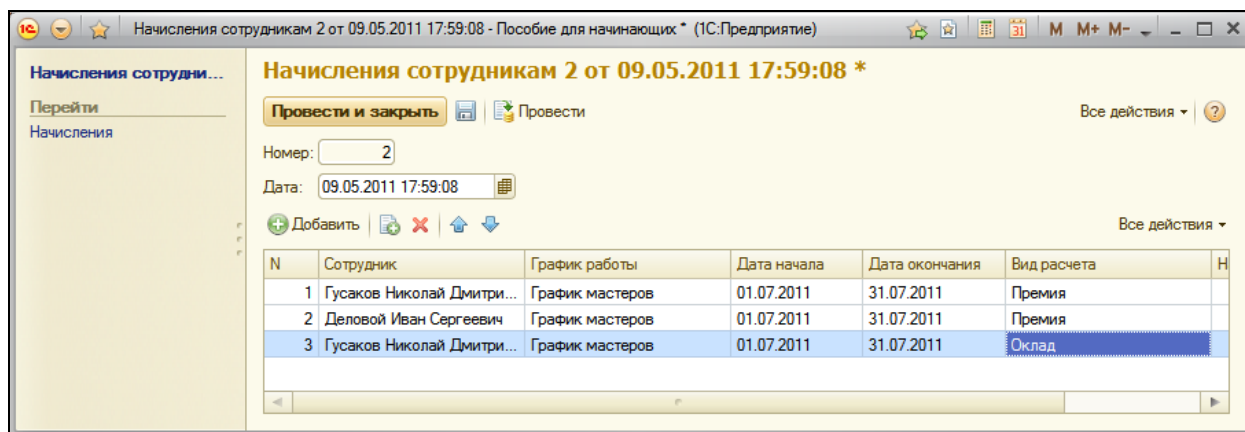
Все хорошо, пока в документе №2 содержатся записи только о расчете премии. Но вспомним, как регулируется наше ограничение доступа: пользователь сможет прочитать документ **Начисления сотрудникам** только в случае, если в его табличной части **Начисления** есть виды расчета, не являющиеся видом расчета **Премия**.

Это значит, что если в этом документе окажутся виды расчета, отличные от **Премия**, мастер сможет его открыть и просмотреть.

Убедимся в этом. Запустим 1С:Предприятие от имени пользователя **Администратор**.

В разделе **Расчет зарплаты** откроем список документов **Начисления сотрудникам**.

Откроем документ №2 и скопируем любую его строку. В новой строке изменим вид расчета на **Оклад**. Проведем и закроем документ. Завершим сеанс работы.



Теперь запустим 1С:Предприятие от имени пользователя **Гусаков**. Откроем список документов **НачисленияСотрудникам**. Откроем **Документ №2**. Документ откроется, и мы увидим все его строки.

## В режиме Конфигуратор

Вернемся в конфигуратор.

Для того чтобы документ невозможно было просмотреть и в этой ситуации, нам нужно будет изменить существующее условие ограничения доступа.

Новое условие будет более сложным, поэтому мы заодно продемонстрируем использование шаблонов в ограничениях доступа.

Откроем роль **Мастер** и перейдем на вкладку **Шаблоны ограничений**.

Здесь добавим новый шаблон с именем **ЕстьПремия**. Текст шаблона будет выглядеть так:

```

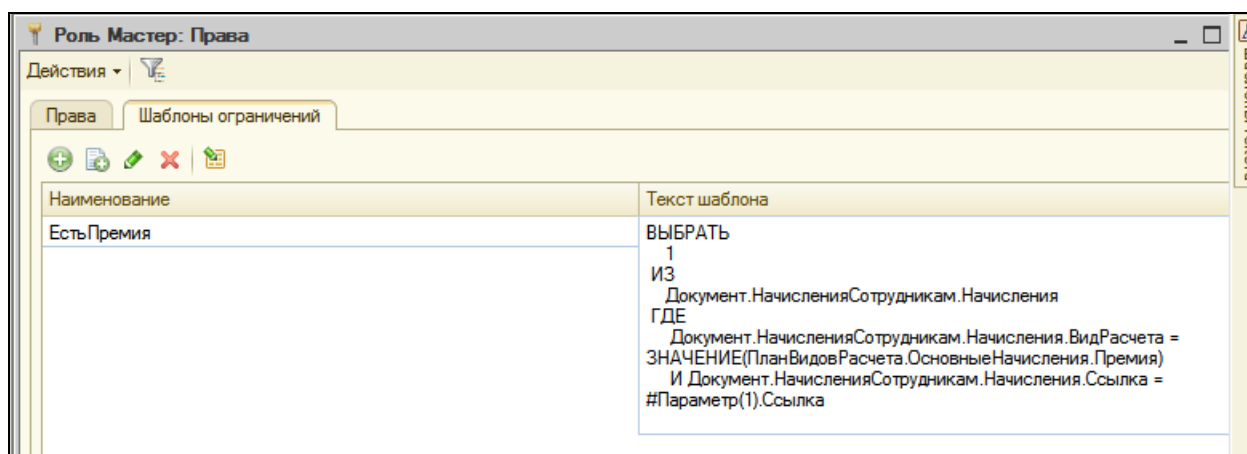
ВЫБРАТЬ
    1
ИЗ
    Документ.НачисленияСотрудникам.Начисления
ГДЕ
    Документ.НачисленияСотрудникам.Начисления.ВидРасчета =
ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)
    И Документ.НачисленияСотрудникам.Начисления.Ссылка =
#Параметр(1).Ссылка

```

По сути это запрос к табличной части документа **НачисленияСотрудникам**, который либо не вернет нам ничего, либо вернет одну запись с одним полем со значением 1. Такую запись он



вернет нам в случае, если в табличной части документа есть вид расчета **Премия**.

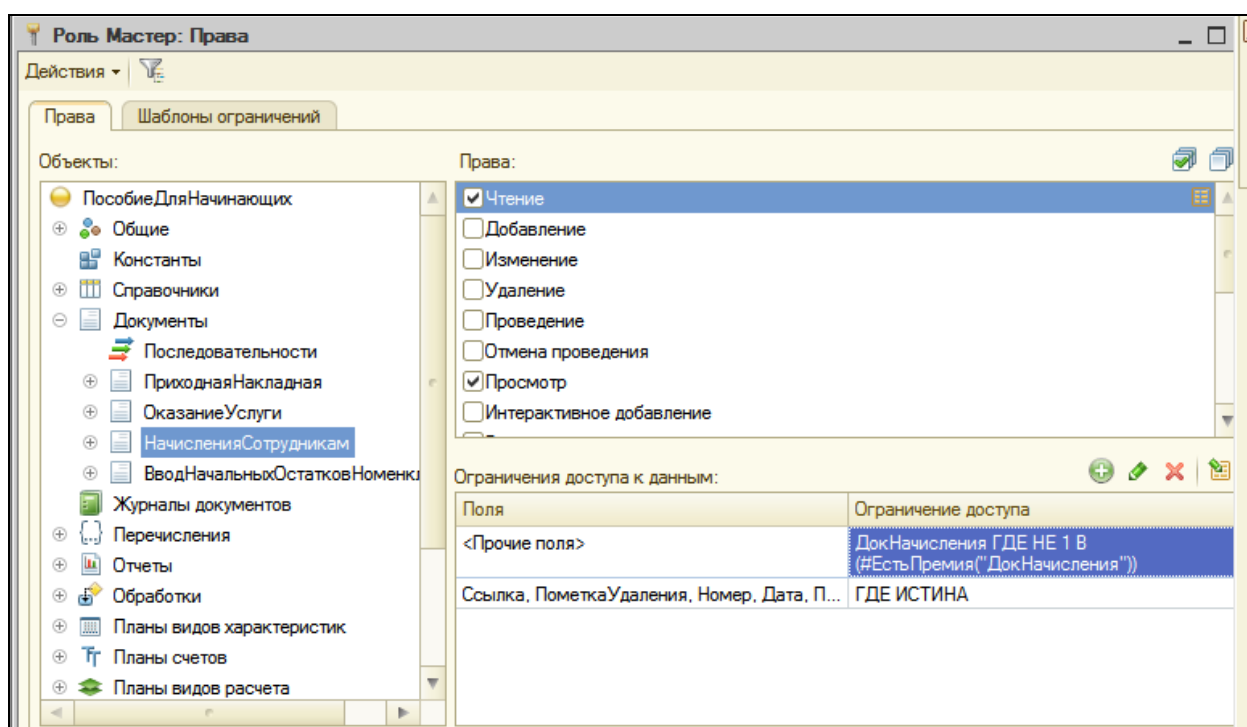


Второе условие (И Документ.НачисленияСотрудникам.Начисления.Ссылка = #Параметр(1).Ссылка) в этом запросе нужно нам для того, чтобы указать, табличная часть какого именно документа нас интересует. В этом условии используется возможность указания параметров в шаблоне.

Вместо #Параметр(1). будет подставлена та строка, которую мы укажем при вызове этого шаблона в условии ограничения доступа.

Теперь вернемся на закладку **Права**. В имеющемся ограничении прав для прочих полей заменим старый текст новым:

ДокНачисления ГДЕ НЕ 1 В (#ЕстьПремия("ДокНачисления"))





Здесь с помощью конструкции #ЕстьПремия("ДокНачисления") мы обращаемся к нашему шаблону. Текст шаблона просто будет подставлен в это место, причем строка ДокНачисления заменит собой первый параметр шаблона (#Параметр(1)).

Поэтому это условие разрешит нам прочитать ДокНачисления тогда, когда запрос из шаблона не возвращает 1.: ГДЕ НЕ 1 В (#ЕстьПремия("ДокНачисления"))

Т.е. тогда, когда в табличной части нет начисления **Премия**.

Можно было бы записать это условие и без использования шаблонов. Но, во-первых, такая запись была бы менее читаемой (листинг ниже), а во-вторых, использование шаблонов позволяет выделить и не дублировать части условий ограничений, которые могут использоваться в разных условиях.

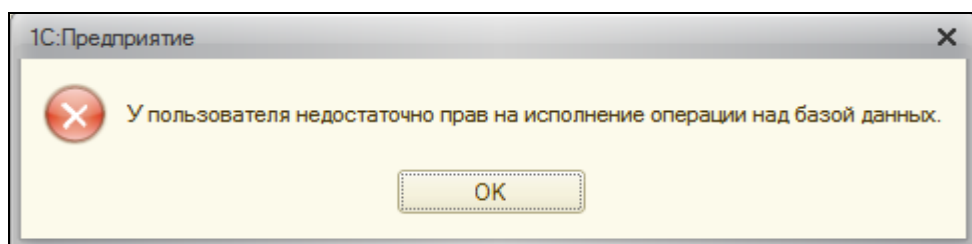
```
ДокНачисления ГДЕ НЕ 1 В (  
    ВЫБРАТЬ  
        1  
    ИЗ  
        Документ.НачисленияСотрудникам.Начисления  
    ГДЕ  
        Документ.НачисленияСотрудникам.Начисления.ВидРасчета =  
        ЗНАЧЕНИЕ(ПланВидовРасчета.ОсновныеНачисления.Премия)  
        И Документ.НачисленияСотрудникам.Начисления.Ссылка =  
        ДокНачисления.Ссылка)
```

Закроем окно редактирования прав и проверим как это работает.

В режиме 1С:Предприятие

Обновим конфигурацию и запустим Предприятие от имени **Гусакова**.

В разделе **Расчет зарплаты** откроем список документов **Начисления сотрудникам**. Как вы помните, в документе №2 есть строки и с видом расчета **Премия**, и с видом расчета **Оклад**. Раньше документ открывался. Попробуем открыть его теперь. Мы получим сообщение о нарушении прав доступа, что нам и требовалось.



## В режиме Конфигуратор

Поскольку пример с ограничением прав доступа на уровне записей и полей базы данных мы делали скорее в демонстрационных целях, вернемся к исходному состоянию конфигурации.

Снимем для роли **Мастер** право **Чтение** для документа **НачисленияСотрудникам**.

Право **Просмотр** для подсистемы **РасчетЗарплаты**.

Право **Чтение** для справочника **ВидыГрафиковРаботы** и для плана видов расчета **Основные начисления**.

Запустим 1С:Предприятие от имени **Администратора**.

В разделе расчет зарплаты откроем список документов Начисления сотрудникам, откроем документ №2 и удалим последнюю строку, которую мы добавляли. Проведем и закроем документ.

### Контрольные вопросы

- ✓ Для чего предназначен объект конфигурации Роль.
- ✓ Как создать роль, используя подсистемы конфигурации.
- ✓ Как создать список пользователей системы и определить их права.
- ✓ Чем аутентификация средствами 1С:Предприятия отличается от аутентификации операционной системы.

## Практическая работа № 22

### *Рабочий стол и настройка командного интерфейса (1:10)*

В этой работе мы придадим нашей конфигурации «товарный» вид, т.е. усовершенствуем командный интерфейс приложения, настроим рабочий стол и видимость команд по ролям для созданных нами пользователей.

Это сделает интерфейс более завершенным и более удобным для пользователя, что очень важно для работы с приложением.

#### **Командный интерфейс разделов**

Пришло время заняться организацией командного интерфейса разделов (подсистем), т.е. осмысленно отсортировать команды, разложить их по группам в зависимости от приоритета и частоты использования.

#### **В режиме Конфигуратор**

Для начала зададим синоним для отчета **ПоискДанных** как **Поиск в данных**. Так будет более понятно для пользователя.

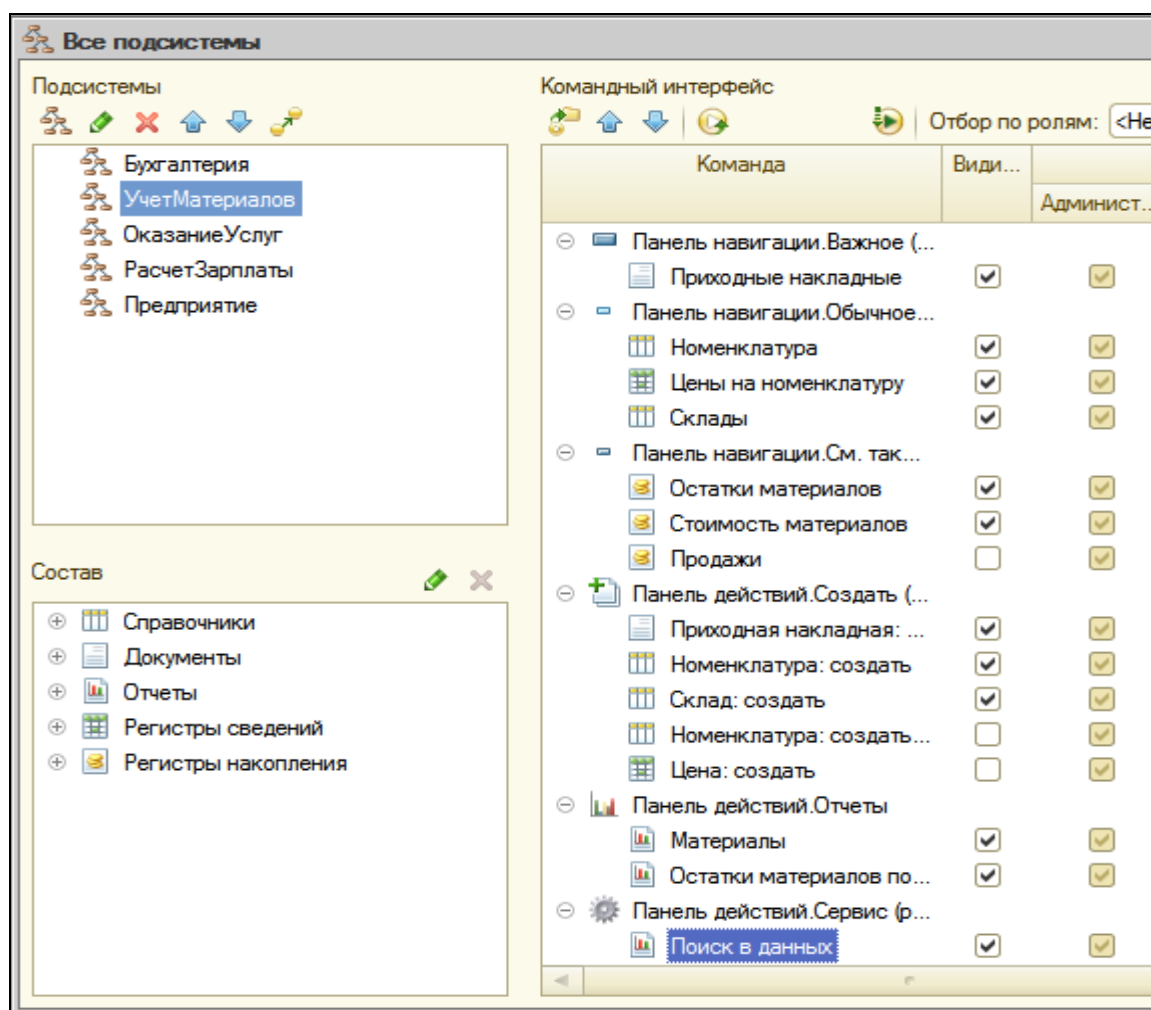
Затем вызовем редактор командного интерфейса подсистем **Все подсистемы (Общие – Подсистемы – Все подсистемы)**.

Выделим в списке подсистем **УчетМатериалов** и в окне командный интерфейс произведем следующие изменения.

- С помощью мыши переместим команду **Приходные накладные** из группы **Панель навигации.Обычное** в группу команд **Панель навигации.Важное**.
- В группе **Панель навигации.Обычное** зададим следующий порядок расположения команд:
  - **Номенклатура,**
  - **Цены на Номенклатуру,**
  - **Склады.**
- В группе **Панель навигации.См. также** уберем видимость у команды **Продажи** и зададим следующий порядок:
  - **Остатки материалов,**

- **Стоимость материалов.**
- В группе **Панель действий.Создать** зададим порядок расположения видимых команд (порядок невидимых команд нам не важен):
  - **Приходная накладная: создать,**
  - **Номенклатура: создать,**
  - **Склад: создать.**
- В группе **Панель действий.Отчеты** зададим порядок:
  - **Материалы,**
  - **Остатки материалов по свойствам.**
- Переместим команду **Поиск в данных** из группы **Панель действий.Отчеты** в группу команд **Панель действий.Сервис.**

В результате окно **Подсистемы** примет вид:



Поясним наши действия.

В разделе **Учет материалов** наиболее часто пользователю может понадобиться создавать приходные накладные и просматривать их список. Поэтому мы поместили команду для просмотра списка приходных накладных (**Приходные накладные**) в группу **Важное** панели навигации, а команду для создания приходных накладных поместили первой в панели действий подсистемы **Учет материалов**.

Довольно часто пользователю может понадобиться создавать новую номенклатуру и просматривать список номенклатуры. Поэтому мы поместили команду для просмотра списка номенклатуры (Номенклатура) в группу **Обычное** панели навигации, а команду для создания номенклатуры поместили второй в панели действий подсистемы Учет материалов. Тоже самое, но с меньшим приоритетом, относится к справочнику складов.

В группе **См.также** панели навигации раздела мы поместили команды **Остатки материалов** и **Стоимость материалов** для просмотра записей соответствующих регистров накопления и расположили их в порядке частоты использования. А команду для просмотра записей регистра **Продажи** вообще убрали, т.к. в разделе **Учет материалов** вряд ли она может понадобиться.

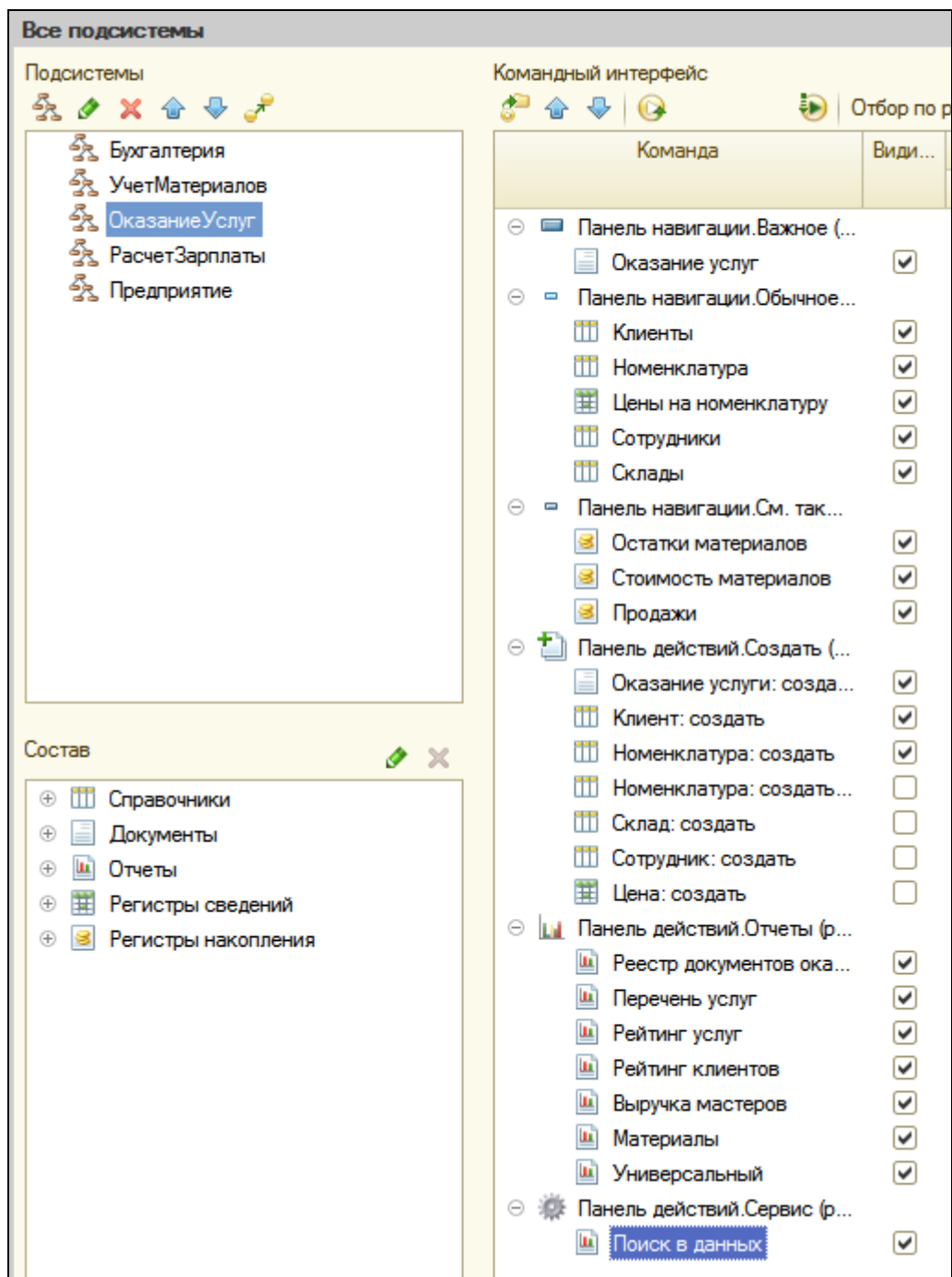
Отчеты в панели действий раздела мы расположили в порядке их приоритета. А команду для открытия отчета **Поиск в данных** мы перенесли из группы **Отчеты** в группу **Сервис** панели действий, т.к. на самом деле поиск данных – это скорее сервисная операция, чем классический отчет.

Руководствуясь подобными соображениями, отредактируем командный интерфейс остальных подсистем.

Подсистема **ОказаниеУслуг** будет иметь следующий интерфейс:

- Группа **Панель навигации.Важное:**
  - **Оказание услуг.**
- Группа **Панель навигации.Обычное:**
  - **Клиенты,**
  - **Номенклатура,**
  - **Цены на Номенклатуру,**

- **Сотрудники,**
- **Склады.**
- **Группа Панель навигации.См. также:**
  - **Остатки материалов,**
  - **Стоимость материалов,**
  - **Продажи.**
- **Панель действий.Создать (только видимые команды):**
  - **Оказание услуги: создать,**
  - **Клиент: создать,**
  - **Номенклатура: создать.**
- **Панель действий.Отчеты:**
  - **Реестр документов Оказание услуги,**
  - **Перечень услуг,**
  - **Рейтинг услуг,**
  - **Рейтинг клиентов,**
  - **Выручка мастеров,**
  - **Материалы**
  - **Универсальный.**
- **Панель действий.Сервис:**
  - **Поиск в данных.**



Подсистема **Бухгалтерия** будет иметь следующий командный интерфейс:

- **Панель навигации.Важное:**
  - **Приходные накладные,**
  - **Оказание услуг.**
- **Панель навигации.Обычное:**
  - **Клиенты,**

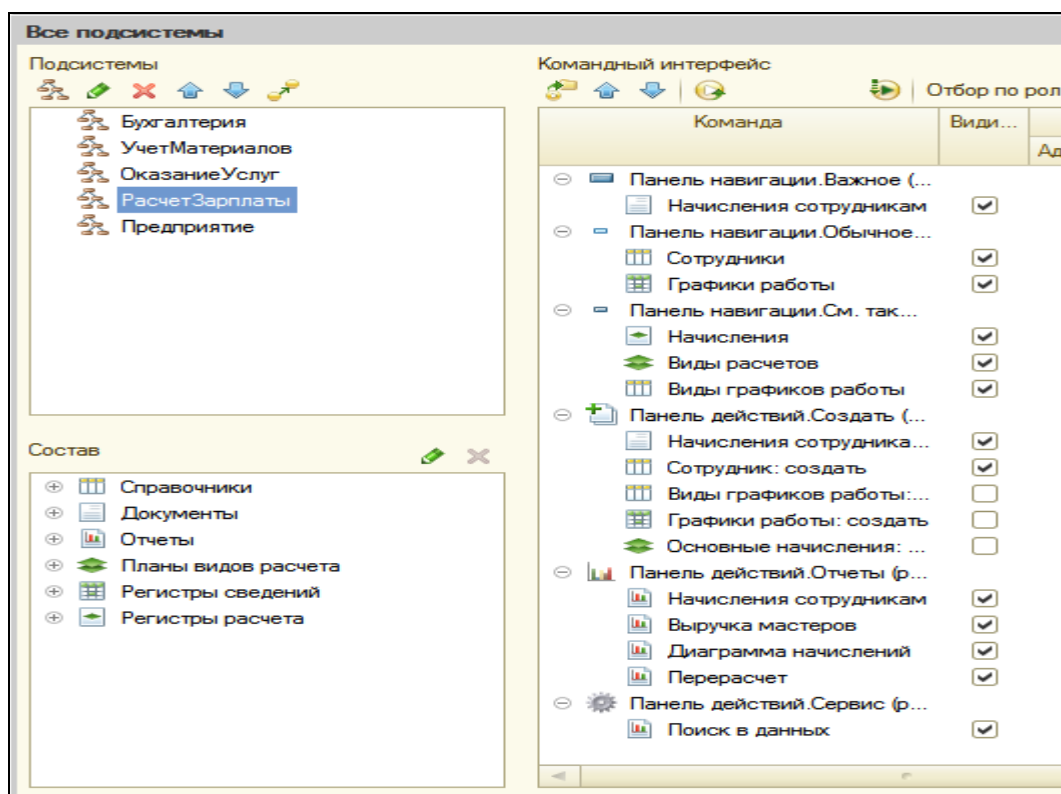
- Номенклатура,
- Цены на номенклатуру.
- **Панель навигации.См. также:**
  - Остатки материалов,
  - Стоимость материалов,
  - Продажи,
  - Ввод начальных остатков номенклатуры,
  - Основной план счетов,
  - Виды субконто
- **Панель действий.Создать (только видимые):**
  - Ввод начальных остатков номенклатуры: создать.
- **Панель действий.Отчеты:**
  - Оборотно-сальдовая ведомость,
  - Начисления сотрудникам,
  - Перечень услуг,
  - Рейтинг услуг,
  - Рейтинг клиентов,
  - Материалы,
  - Остатки материалов по свойствам.
- **Панель действий.Сервис:**
  - Поиск в данных.

Подсистема **РасчетЗарплаты:**

- **Панель навигации.Важное:**
  - Начисления сотрудникам.
- **Панель навигации.Обычное:**
  - Сотрудники,

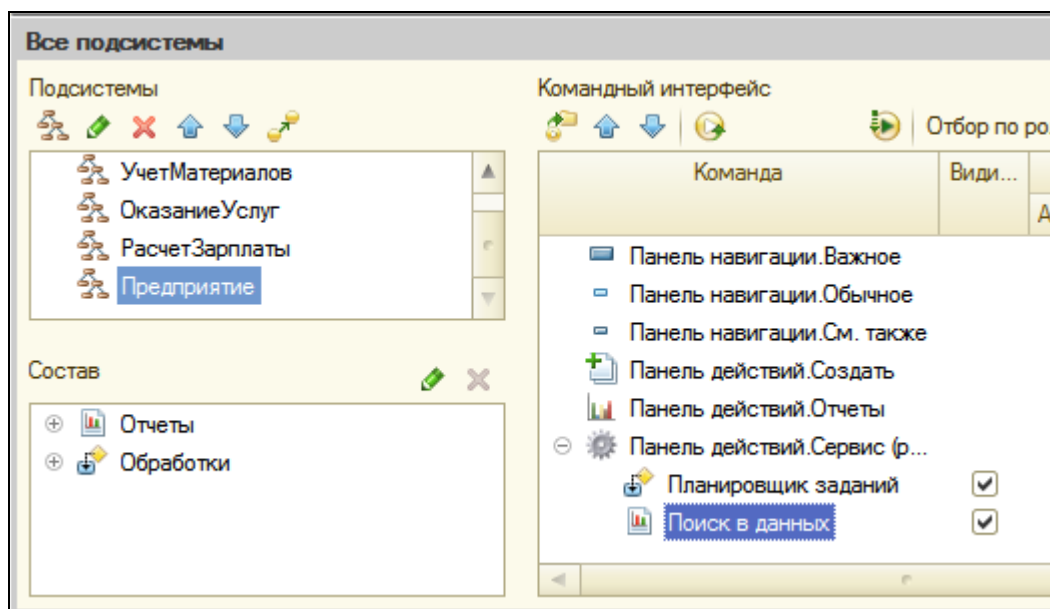


- **Графики работы.**
- **Панель навигации.См. также:**
  - **Начисления,**
  - **Виды расчетов,**
  - **Виды графиков работы.**
- **Панель действий.Создать (только видимые):**
  - **Начисления сотрудникам: создать,**
  - **Сотрудник: создать.**
- **Панель действий.Отчеты:**
  - **Начисления сотрудникам,**
  - **Выручка мастеров,**
  - **Диаграмма начислений,**
  - **Перерасчет.**
- **Панель действий.Сервис:**
  - **Поиск в данных.**



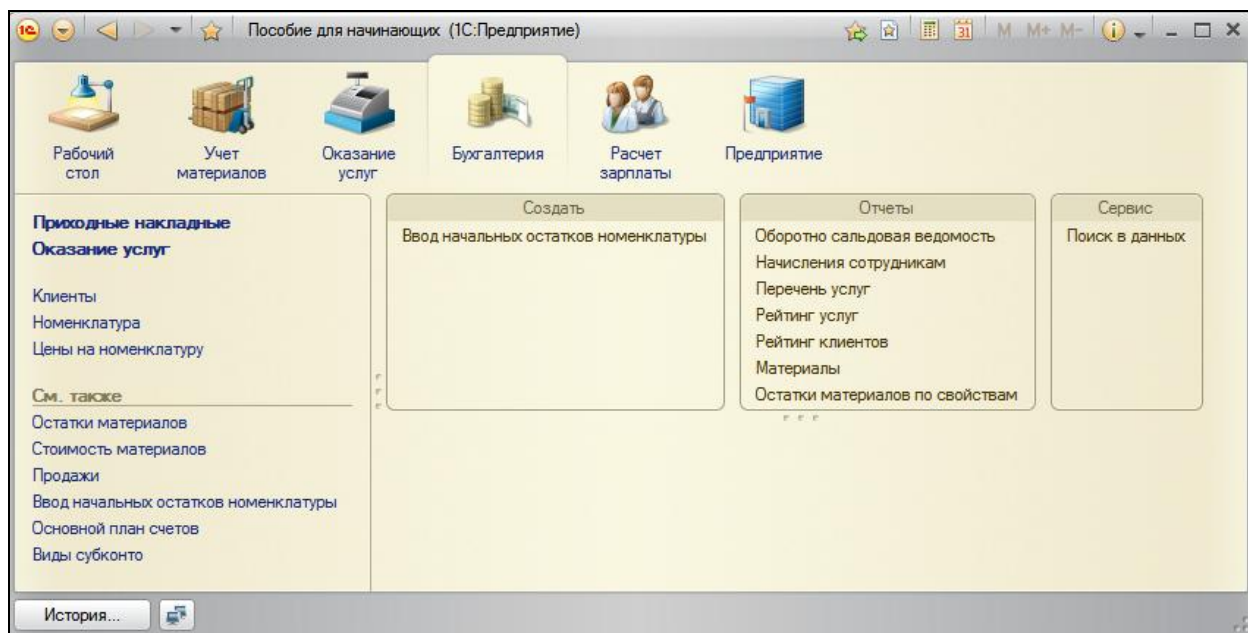
## Подсистема **Предприятие**:

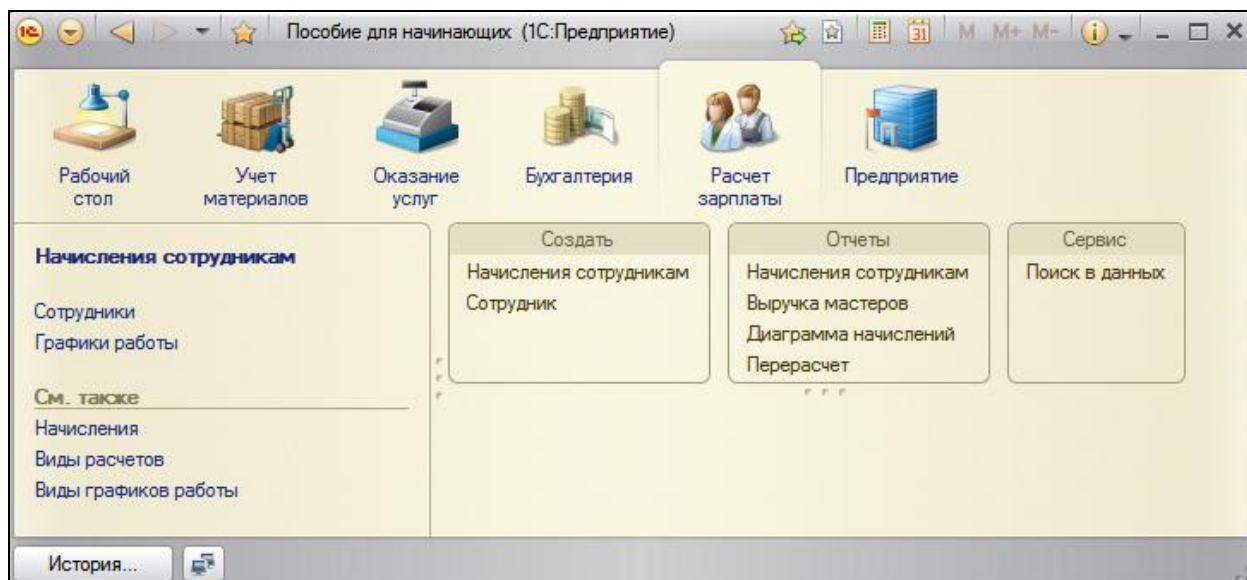
- **Панель действий.Сервис:**
  - **Планировщик заданий,**
  - **Поиск в данных.**



## В режиме 1С:Предприятие

Запустим режим отладки под Администратором. Так будет выглядеть интерфейс разделов **Бухгалтерия** и **Расчет зарплаты**:





Как мы видим, все группы команд зрительно отделены друг от друга, а наиболее важные команды выделены жирным шрифтом.

Однако, если у пользователя есть другие предпочтения в настройке интерфейса, то в режиме 1С:Предприятие он может сам настроить интерфейс, выполнив команду главного меню **Сервис – Настройка интерфейса**.

## Рабочий стол

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

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

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

## В режиме Конфигуратор

Начнем настройку рабочего стола.

Выделим корень дерева объектов конфигурации, вызовем контекстное меню и выберем пункт **Открыть рабочую область рабочего стола**.

Откроется окно настройки рабочего стола. Сначала вверху окна выберем шаблон рабочего стола **Две колонки разной ширины (2:1)**.

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

Можно выбрать другой шаблон, при котором колонки будут одинаковой ширины, или будет всего одна колонка. Но кажется, что предпочтительнее первый вариант, т.к. в этом случае взгляд пользователя сразу будет падать на наиболее приоритетные для работы формы, которые мы расположим в левой колонке.

Формы в каждой колонке будут располагаться друг под другом. Оптимально, если в левой колонке будет располагаться одна, максимум две формы, а в правой – две-три формы для каждого пользователя.

Следует иметь в виду, что автоматически созданные системой формы нельзя располагать на рабочем столе. Поэтому нужно создать форму в явном виде в конфигурации. Чтобы не путаться, будем создавать формы прямо по ходу.

Начнем настройку рабочего стола для роли Мастер.

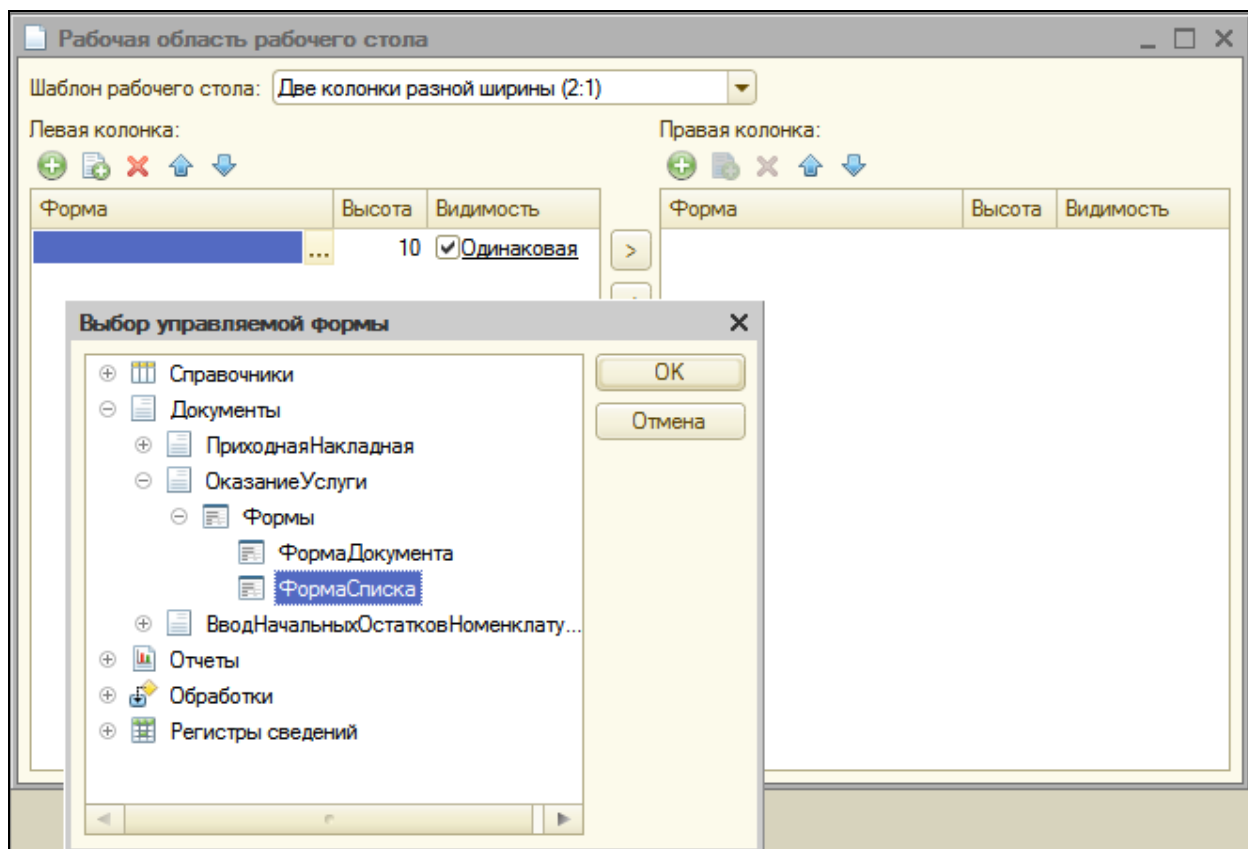
Наша фирма специализируется на оказании услуг и мастера имеют к этому непосредственное отношение. Поэтому логично, если для них в левой колонке рабочего стола будет располагаться список документов об оказании услуг, а в правой – список приходных накладных и список клиентов.

Перечисленные формы списка отсутствуют в конфигурации, поэтому создадим формы списка для объектов конфигурации:

- Справочник **Клиенты**,
- Документ **Приходная Накладная**,
- Документ **Оказание Услуги**.

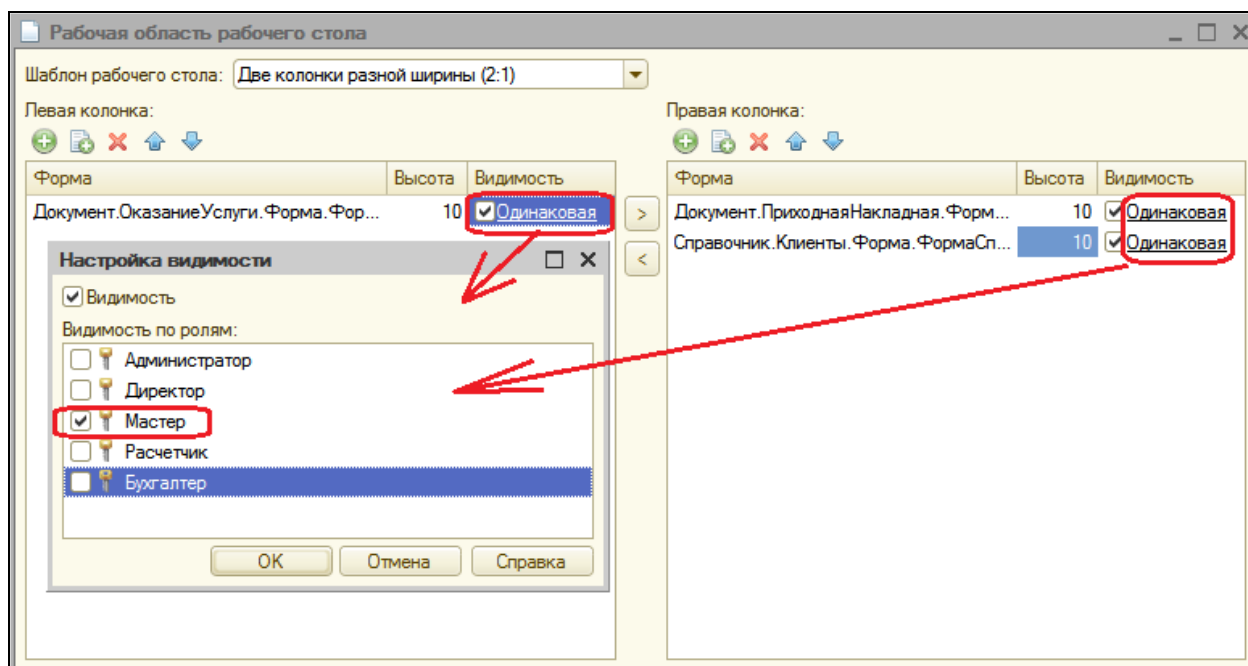
После этого перейдем в окно настройки рабочего стола и над списком форм левой колонки нажмем **Добавить**.

Выберем форму списка документа **Оказание Услуги**.



Аналогичным образом добавим в правую колонку формы списка документа **ПриходнаяНакладная** и справочника **Клиенты**.

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



Возможно, эти формы понадобятся на рабочем столе для пользователей с другими ролями, но мы сейчас, чтобы не запутаться, ограничимся только ролью **Мастер**.

Теперь настроим рабочий стол для роли **Бухгалтер**.

Предположим, что бухгалтер наиболее часто будет пользоваться оборотно-сальдовой ведомостью и отчетом о начислениях сотрудникам. Расположим эти отчеты в левой колонке рабочего стола, а правую оставим пустой.

Создадим форму отчета для отчета **ОборотноСальдоваяВедомость**, а для отчета **НачисленияСотрудникам** форму отчета мы уже создали ранее.

Затем перейдем в окно настройки рабочего стола, добавим эти формы в левую колонку и установим видимость этих форм только для роли **Бухгалтер**.

Аналогично настроим рабочий стол для роли **Расчетчик**.

По роду деятельности расчетчик в основном пользуется документами и отчетами о начислениях сотрудникам. Расположим отчет о начислениях сотрудникам в левой колонке, а в правой – список документов о начислениях.

Для этого создадим форму списка документа **НачисленияСотрудникам**.

Теперь перейдем в окно настройки рабочего стола, добавим форму списка документа НачисленияСотрудникам в правую колонку и установим видимость этой формы только для Расчетчика.

Отчет **НачисленияСотрудникам** мы уже добавили для роли **Бухгалтер**, поэтому установим также его видимость для роли **Расчетчик**.

Затем настроим рабочий стол для роли **Директор**.

Мы предполагаем, что эта роль будет назначена пользователю с руководящими функциями. Ему не нужно вводить никаких документов, да и у него и нет на это прав.

Но ему понадобится регулярно просматривать отчеты о деятельности фирмы, чтобы принимать решения.

Поэтому расположим на его столе в левой колонке отчет **Рейтинг услуг**, а в правой – отчеты **Перечень услуг** и **Выручка мастеров**.

Создадим формы отчета для этих отчетов. Затем перейдем в окно настройки рабочего стола и добавим эти формы, установим видимость только для роли **Директор**.

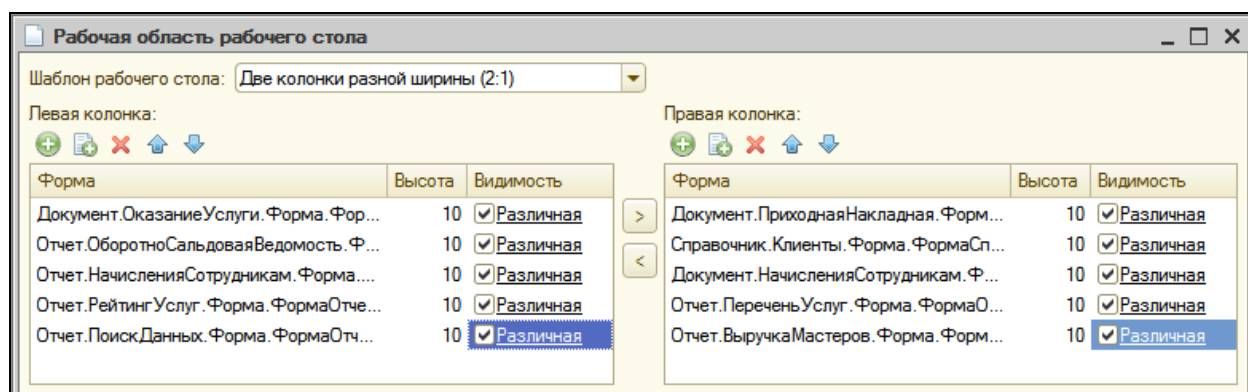
Рабочий стол для роли **Администратор**.

По роду деятельности администратор отвечает за состояние информационной базы и может иметь доступ ко всем объектам конфигурации. Но для администратора не должно быть проблемой найти и выполнить любую команду нашей конфигурации. Поэтому лучше узнать у администратора, что ему хочется иметь под рукой.

Для примера расположим на его рабочем столе в левой колонке отчет для поиска данных, а правую оставим пустой.

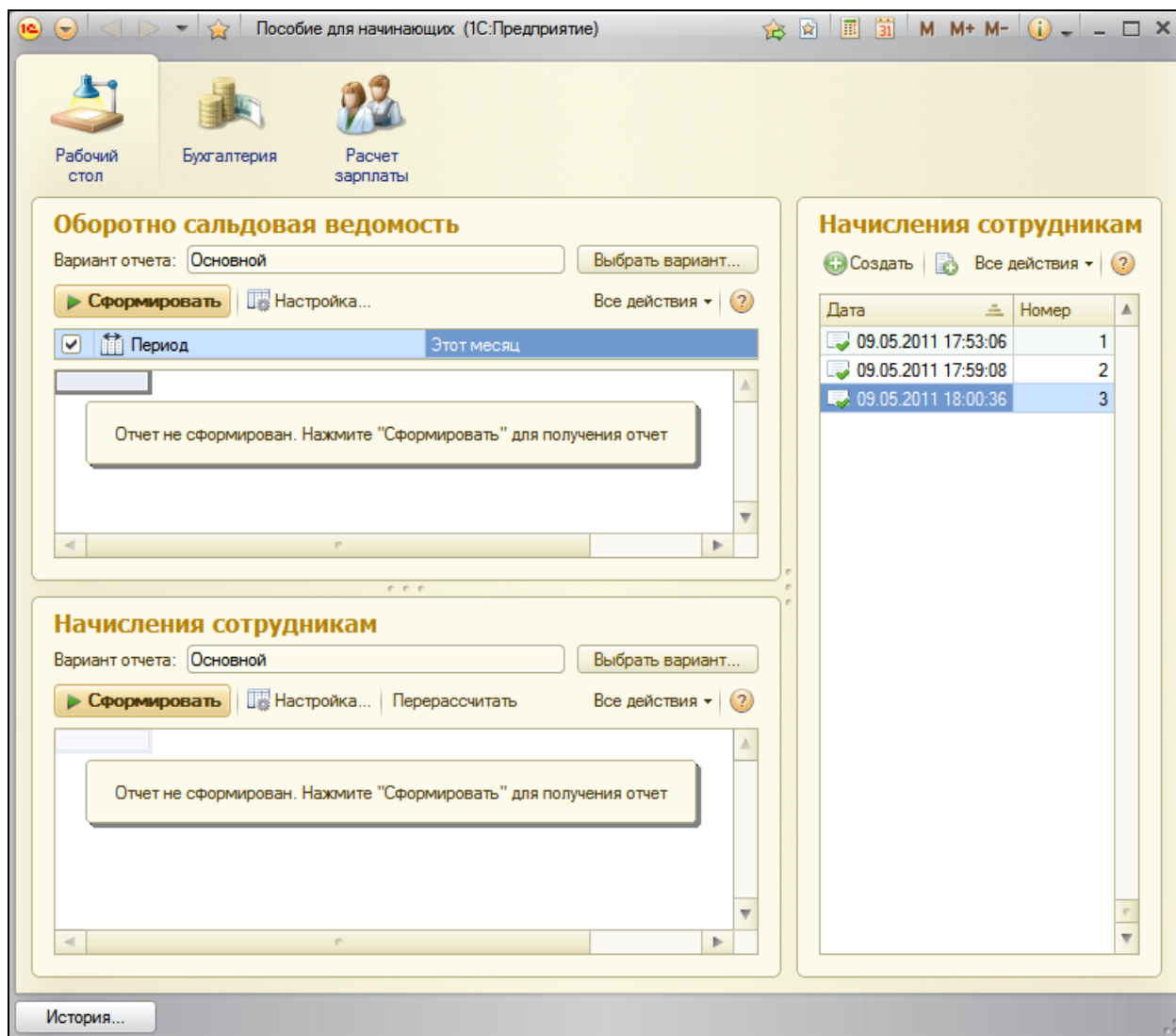
Форму для отчета **ПоискДанных** мы уже создали ранее. Теперь перейдем в окно настройки рабочего стола, добавим форму в левую колонку и установим видимость для роли **Администратор**.

В результате окно настройки рабочего стола должно принять вид:



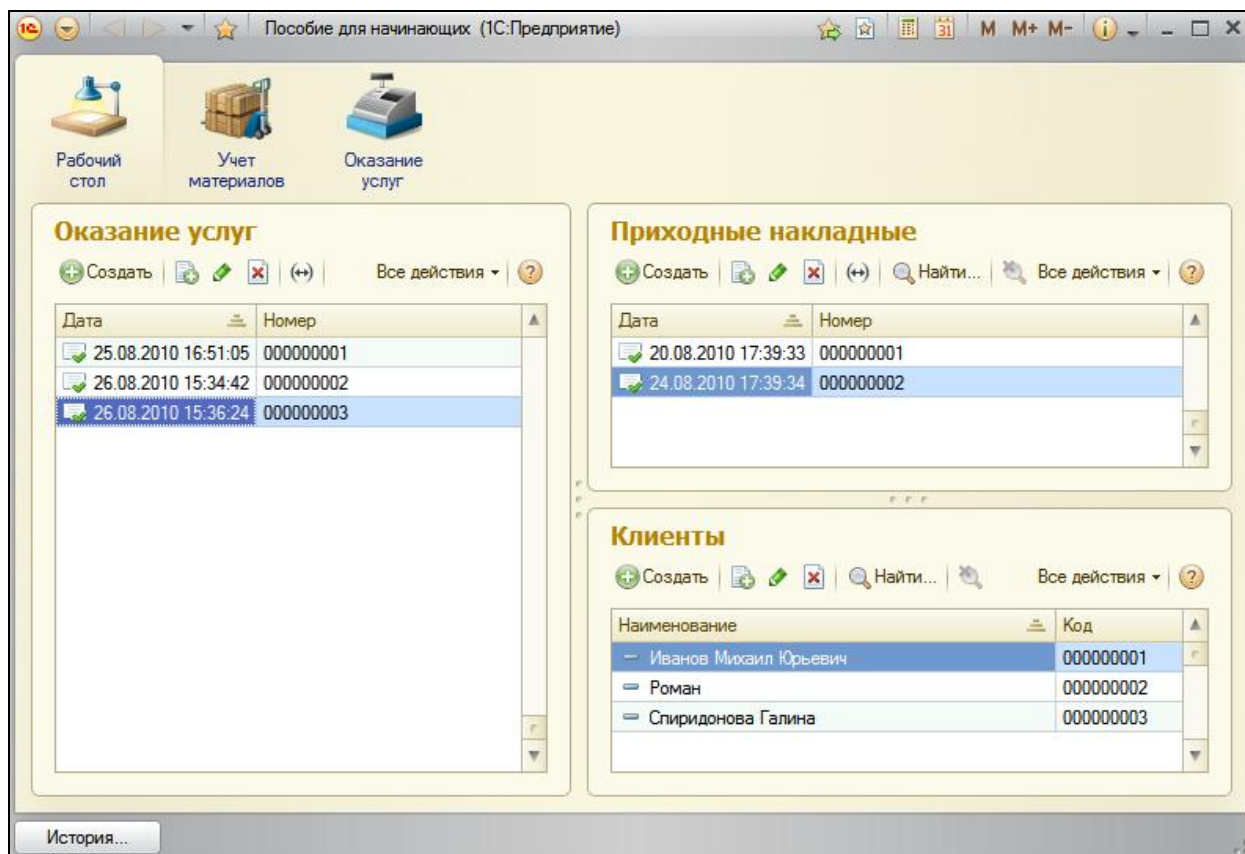
В режиме 1С:Предприятие

Зайдем в режим отладки под пользователем **Назарова (Бухгалтер и Расчетчик)** – потребуется создать отдельный сеанс через **Пуск – 1С...(Тонкий клиент)**, увидим такой рабочий стол:



А для пользователя **Гусаков** (с ролью **Мастер**) рабочий стол будет таким:





Можно также настроить панель навигации и панель действий рабочего стола, выполнив команду **Открыть командный интерфейс рабочего стола**, но вряд ли в этом есть необходимость, т.к. все нужные команды есть в панели действий и в панели навигации разделов, доступных пользователю.


Наконец, в процессе работы 1С:Предприятия 8 пользователь может настраивать рабочий стол по своему усмотрению, выполнив команду главного меню **Сервис – Настройка интерфейса – Рабочий стол**.

Но следует помнить, что пользователь может размещать на своем рабочем столе только формы, заданные разработчиком в конфигурации.

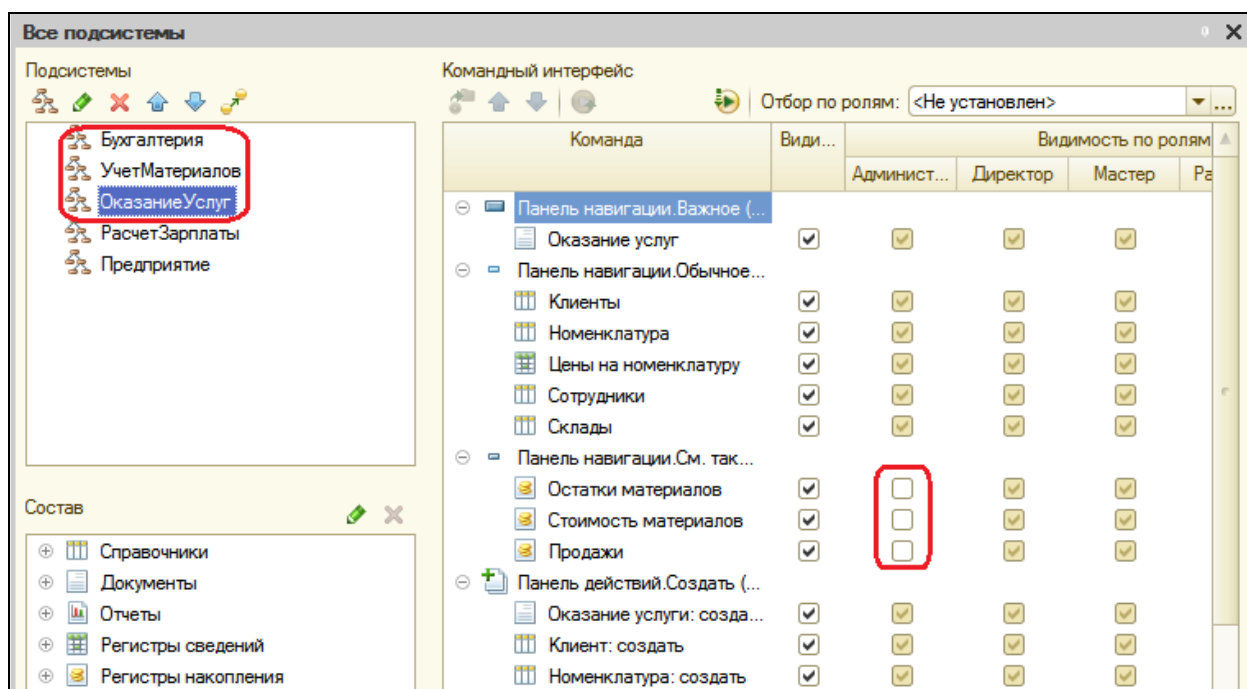
### Видимость команд по ролям

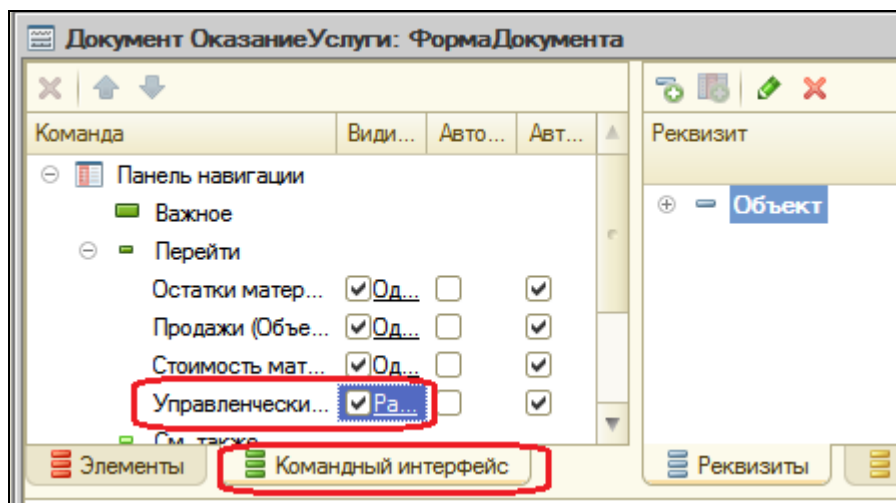
Используя команду **Общие – Подсистемы – Все подсистемы**, мы не раз редактировали командный интерфейс подсистем – меняли порядок, включали видимость команд и т.д. после появления ролей в конфигурации появилась возможность настройки видимости этих команд по ролям.

Ролевое редактирование видимости команд по умолчанию – это средство, позволяющее настроить начальную «насыщенность» глобального командного интерфейса в первую очередь для пользователей с широкими правами доступа.

Для администратора системы, с его широкими правами доступа, число команд оказалось слишком большим. Предположим, что командой открытия регистров накопления  из панели навигации разделов он будет пользоваться крайне редко.

Для облегчения его командного интерфейса, опять перейдем к редактированию глобального командного интерфейса (**Общие – Подсистемы – Все подсистемы**) и для соответствующих команд каждой подсистемы снимем видимость для роли **Администратор**. А также откроем формы документов **ОказаниеУслуги** и **ПриходнаяНакладная** и на закладке **Командный интерфейс** в панели навигации снимем видимость у команд перехода к регистру бухгалтерии **Управленческий** для роли **Администратор**.





## В режиме 1С:Предприятие

Запустите режим отладки и убедитесь, что команды открытия регистров накопления не видны для администратора, хотя он имеет к ним доступ. Например, через главное меню: Все функции – Регистры накопления.

В реальной конфигурации, конечно, роли пользователей и наполнение их рабочих столов будут другими. Это зависит от специфики работы предприятия и пожеланий заказчика.

В последних двух работах мы показали только принцип организации интерфейса прикладного решения по ролям и ограничения доступа к отдельным командам и разделам в целом в зависимости от прав пользователя.

## Контрольные вопросы

- ✓ Что такое рабочий стол
- ✓ Как настроить рабочий стол для различных пользователей
- ✓ Как настроить видимость команд по ролям.