Курс «Практическое применение системы компоновки данных»

Методическое пособие

Содержание

Введение.....3
1. Шаг первый. Простой отчет без кодирования.....4
2. Немного теории. Устройство системы компоновки данных.....18
3. Шаг второй. Работа с параметрами и отборами.....21
4. Шаг третий. Работа с различными наборами данных. Внешние наборы данных.....33
5. Шаг четвертый. Макеты компоновки данных. Создание собственных макетов.....39
6. Шаг пятый. Использование характеристик. Условное оформление. Вычисляемые поля.....49
7. Шаг шестой. Расшифровка.....55

В данном пособии используются материалы из книги Е. Ю. Хрусталевой «Разработка сложных отчетов в 1С:Предприятии 8. Система компоновки данных»

Разработка отчетов с использованием системы компоновки данных

Курс предназначен для разработчиков, имеющих опыт работы в среде 1С:Предприятие 8.1 и желающих освоить работу с системой компоновки данных.

Введение.

Целью автоматизации деятельности любой фирмы в конечном итоге является формирование разнообразной отчетности, удовлетворяющей интересам бухгалтерского, финансового и административного учета. Для решения этих задач в платформе 1С:Предприятие 8.1 используется специальный механизм построения аналитической отчетности – Система компоновки данных.

1. Шаг первый. Простой отчет без кодирования

В этой части мы рассмотрим пример создания несложного отчета с помощью системы компоновки данных. Пример полностью основан на использовании визуальных средств разработки «без единой строчки кода».

Практическое знакомство с системой компоновки данных начнем с простого отчета, который будет отражать приход, расход и остатки товаров на складах.

Создадим новый внешний отчет, используя команду Файл – Новый..

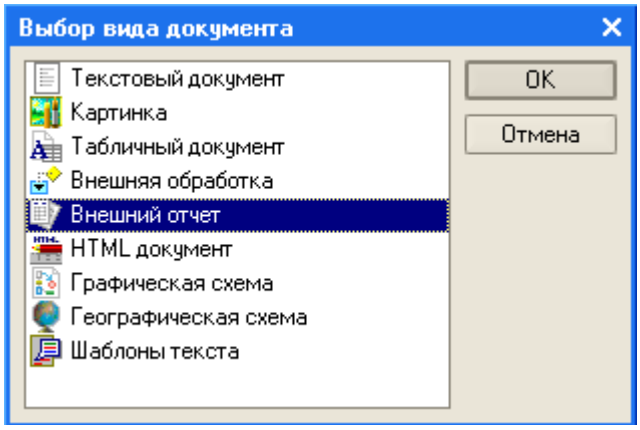


Рис. 1.1. Выбор типа создаваемого документа.

Выберем «Внешний отчет».

Зададим имя отчета **ОстаткиНоменклатуры**, сохраним отчет на диске под каким-нибудь именем. Создадим схему компоновки данных отчета, нажав кнопку **Открыть схему компоновки данных**

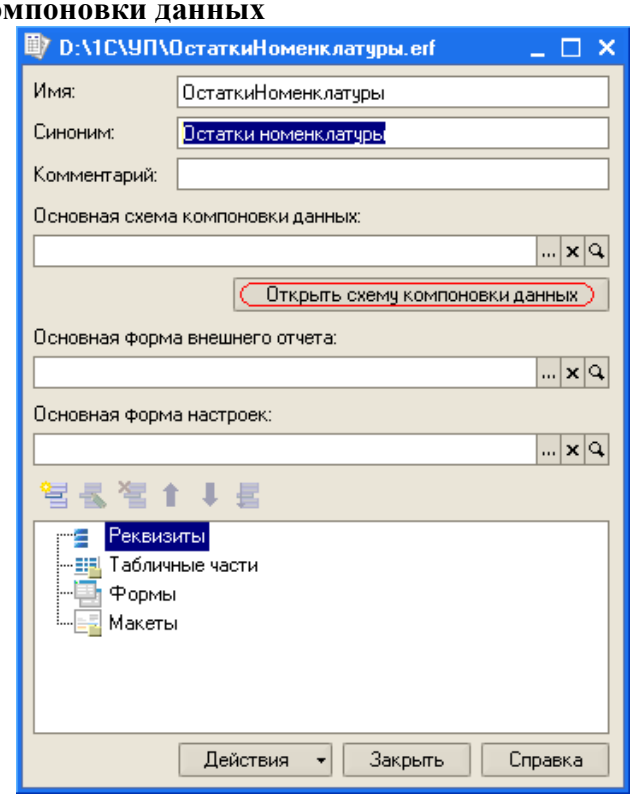


Рис. 1.2. Окно редактирования внешнего отчета

Так как у нашего отчета еще нет схемы компоновки данных, система предложит создать новую схему и откроет конструктор макета.

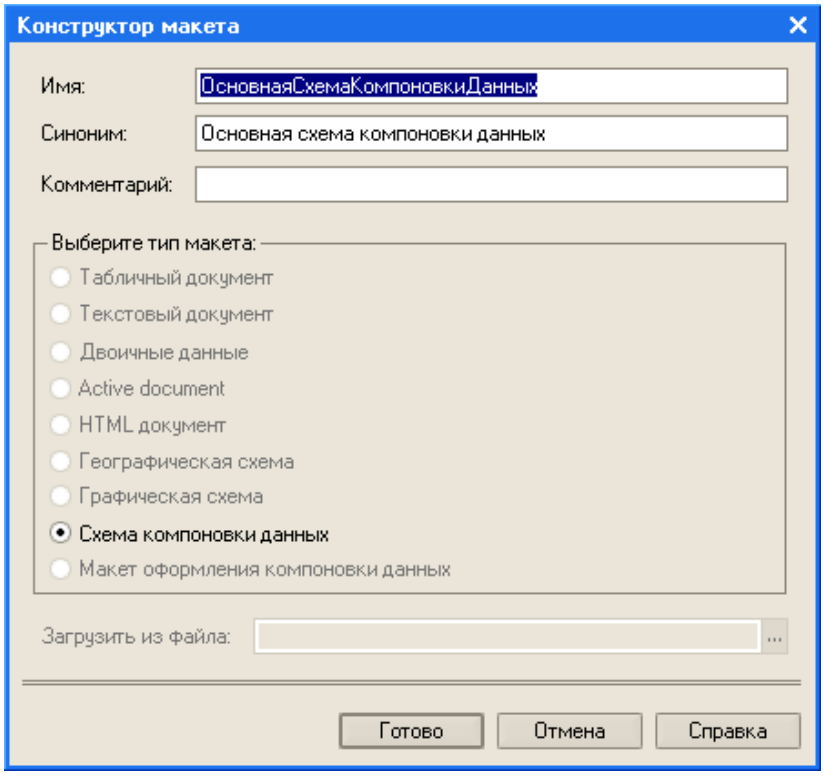


Рис. 1.3. Конструктор макета

Здесь можно изменить имя схемы, так как в общем случае, в отчете может быть несколько схем компоновки данных. Мы оставим предложенное имя и нажмем кнопку **Готово**

Схема компоновки данных хранится в специальном макете - **Схема компоновки данных**. В дальнейшем эту схему можно будет найти и отредактировать в разделе макетов наряду с другими макетами.

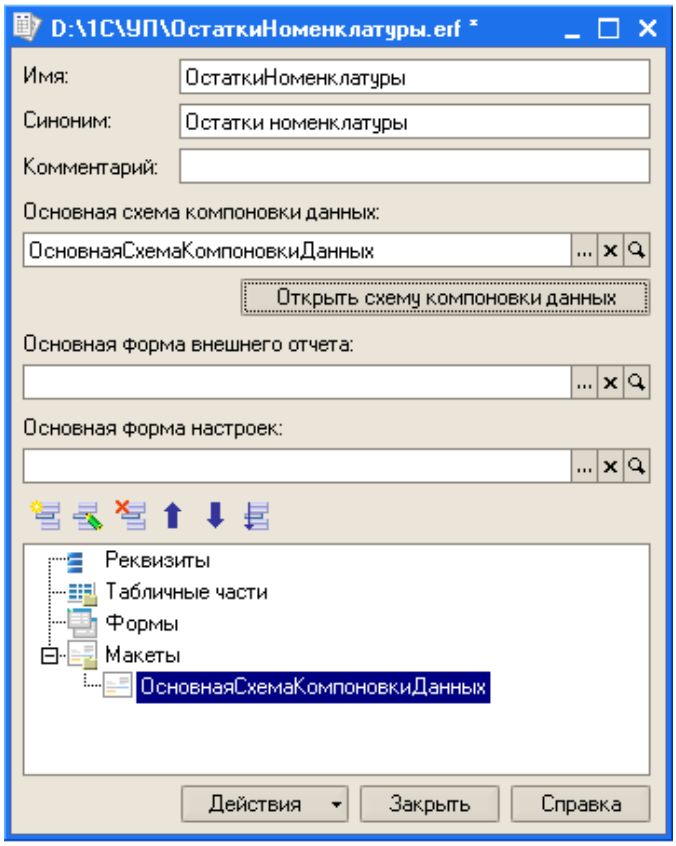


Рис. 1.4. Раздел макетов

Затем откроется окно конструктора схемы компоновки данных. В нем будет происходить вся работа по визуальному конструированию отчета.

Схема компоновки данных может содержать один или несколько наборов данных, являющихся «источниками» данных для отчета. В нашем примере мы будем использовать один набор данных, описанный с помощью запроса к базе данных.

В системе компоновки данных используются не все возможности языка запросов, например, нет секции итогов, так как система состоятельно рассчитывает итоге на основании настроек, сделанных разработчиком или пользователем. Также для системы компоновки данных разработано специальное расширение языка запросов, чтобы описывать дополнительную функциональность в отчетах.

Добавление набора данных

Создадим новый набор данных – запрос. Для этого нажмем кнопку **Добавить набор данных - запрос**.

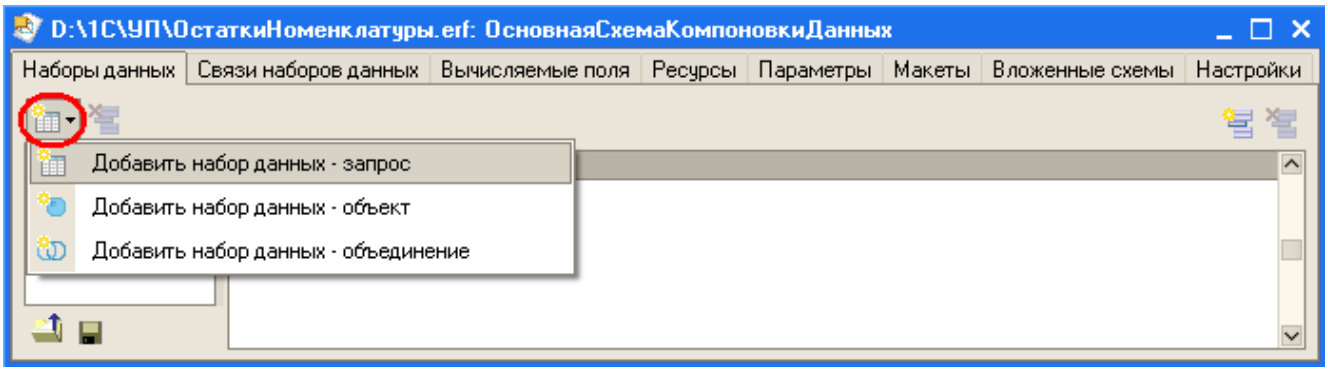


Рис. 1.5. Добавление набора данных

В правом нижнем окне конструктора схемы компоновки мы можем редактировать текст запроса, или воспользоваться конструктором запроса.

Создание запроса

Нажмем кнопку **Конструктор запроса**.

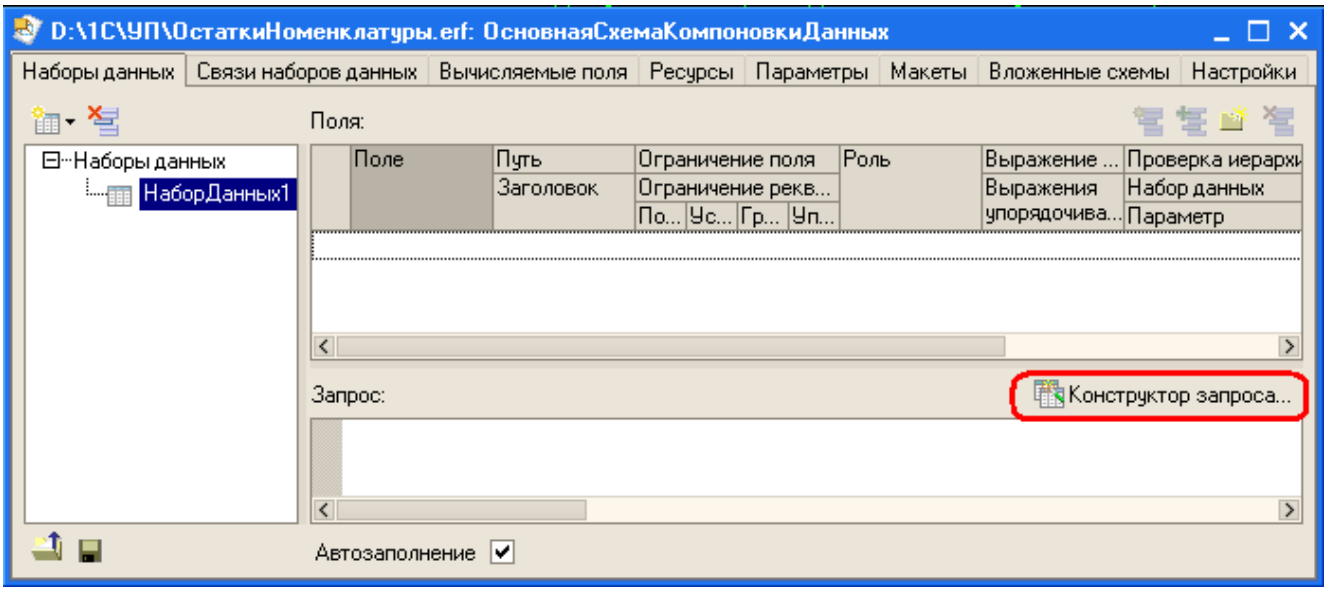


Рис. 1.6. Вызов конструктора запроса

Добавим виртуальную таблицу регистра **ОстаткиНоменклатуры.ОстаткиИОбороты**, двойным щелчком по имени таблицы в списке Таблицы в список полей добавим все доступные поля (рис. 1.7.).

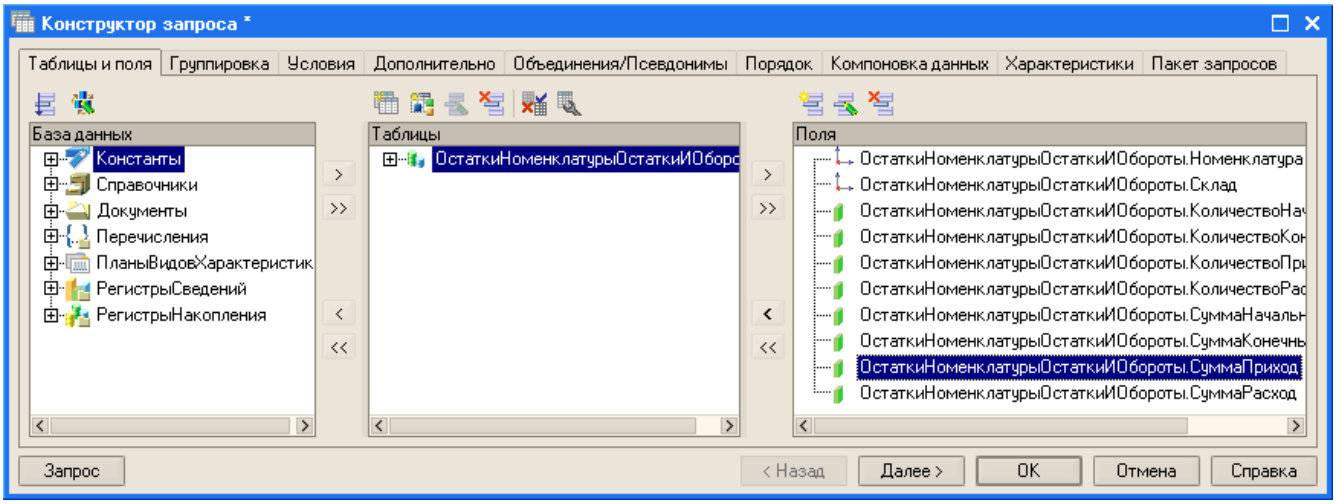


Рис. 1.7. Выбор таблицы и полей для запроса

Мы создали запрос. Нажмем **ОК** и вернемся в конструктор схемы компоновки данных (рис. 1.8)

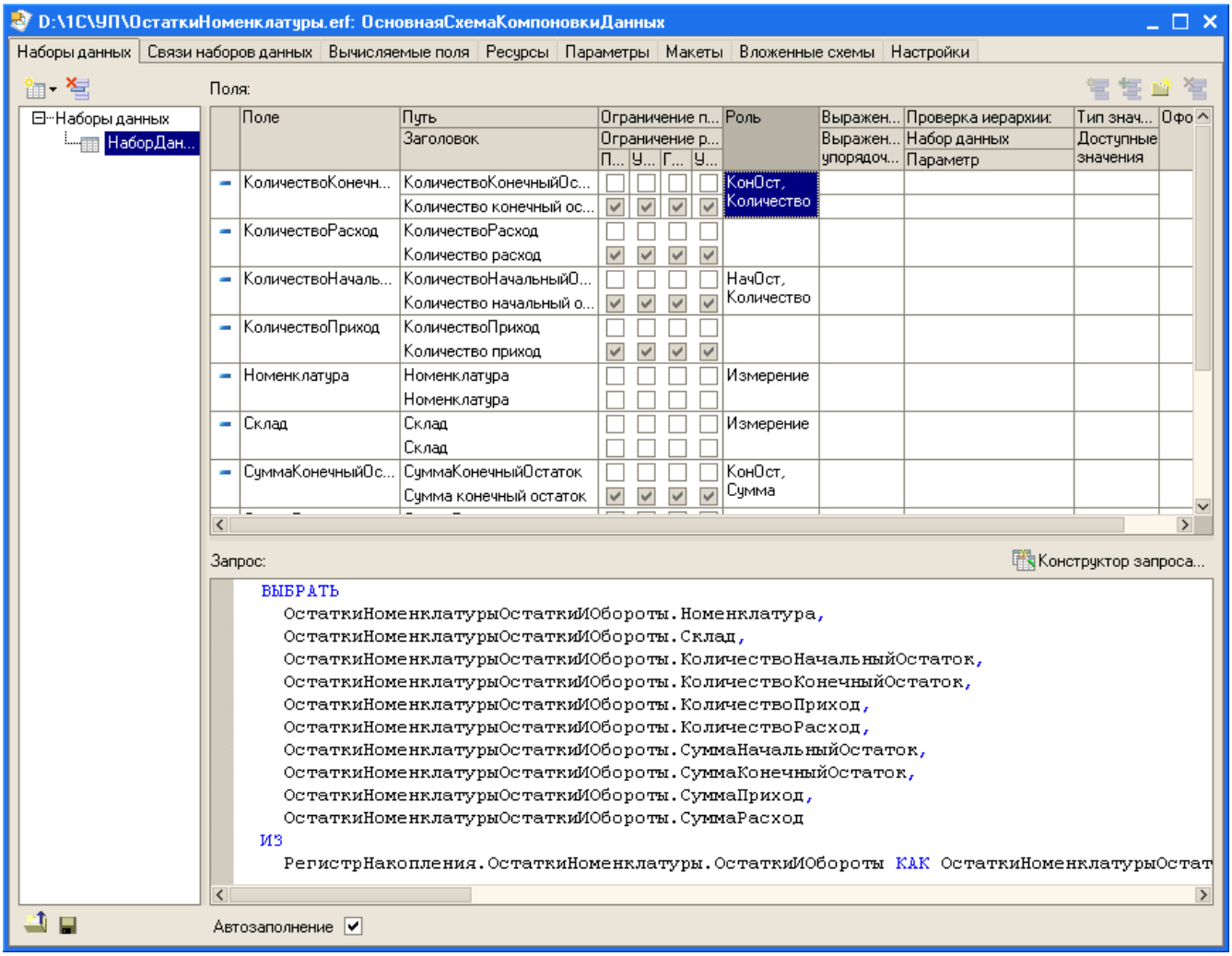


Рис. 1.8. Конструктор схемы компоновки данных

Так как свойство **Автозаполнение** включено, платформа автоматически заполнила список доступных полей набора данных исходя из запроса. Доступные поля – это поля, получаемые из источников данных, которыми оперирует схема компоновки данных. Они могут быть использованы в настройках отчета (сортировке, отборе), участвовать в группировках, формировании вычисляемых полей.

Создание стандартных настроек отчета

Стандартные настройки создаются разработчиком для того, чтобы пользователь, открыв отчет, сразу получил нужные данные и чтобы избавить пользователя от необходимости создавать эти настройки самому. Однако, при желании, пользователь может изменить эти настройки различными способами. Если его не устроил результат этих изменений, он может вернуться к стандартным настройкам, сделанных разработчиком.

Представим наш отчет в виде таблицы, где по строкам данные группируются по позициям номенклатуры, а по столбцам – по складам. Данные, по которым можно считать итоги в терминах СКД называются **Ресурсы.** Их мы должны определить на закладке **Ресурсы** конструктора схемы компоновки данных (рис. 1.9).

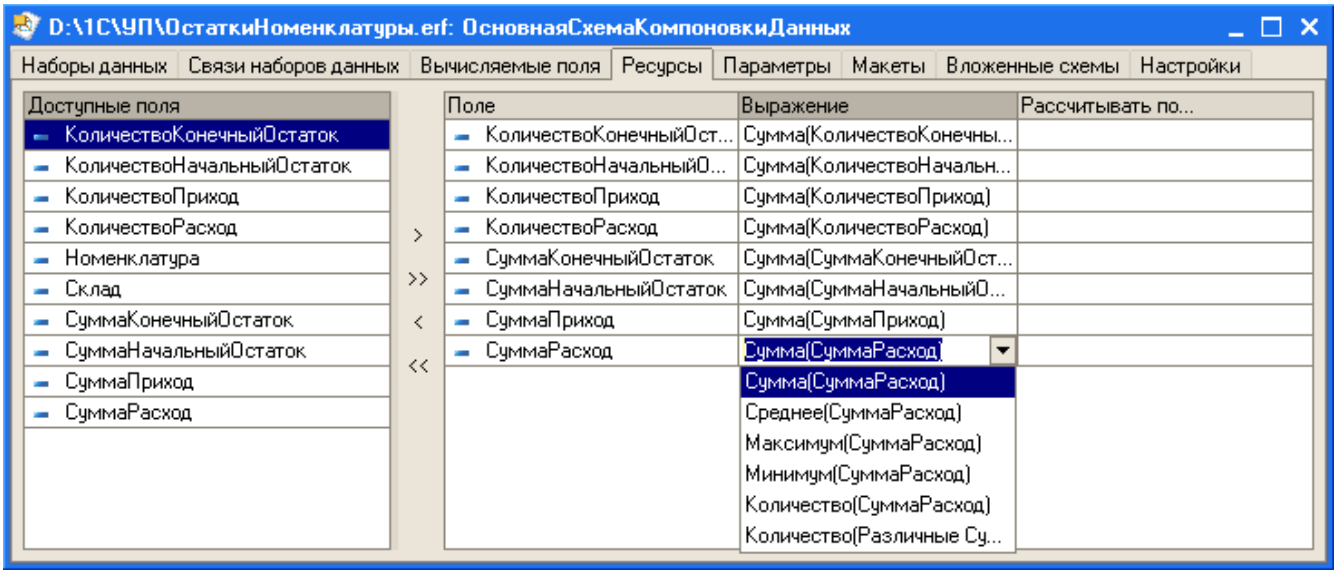


Рис. 1.9. Выбор ресурсов отчета

В колонке **Выражение** здесь содержится выражение (агрегирующая функция), по которому будет вычисляться значение ресурса. По умолчанию предлагается функция **Сумма**, но доступны и другие функции.

Создадим таблицу отчета, для этого перейдем на закладку **Настройки** и запустим **Конструктор настроек.**

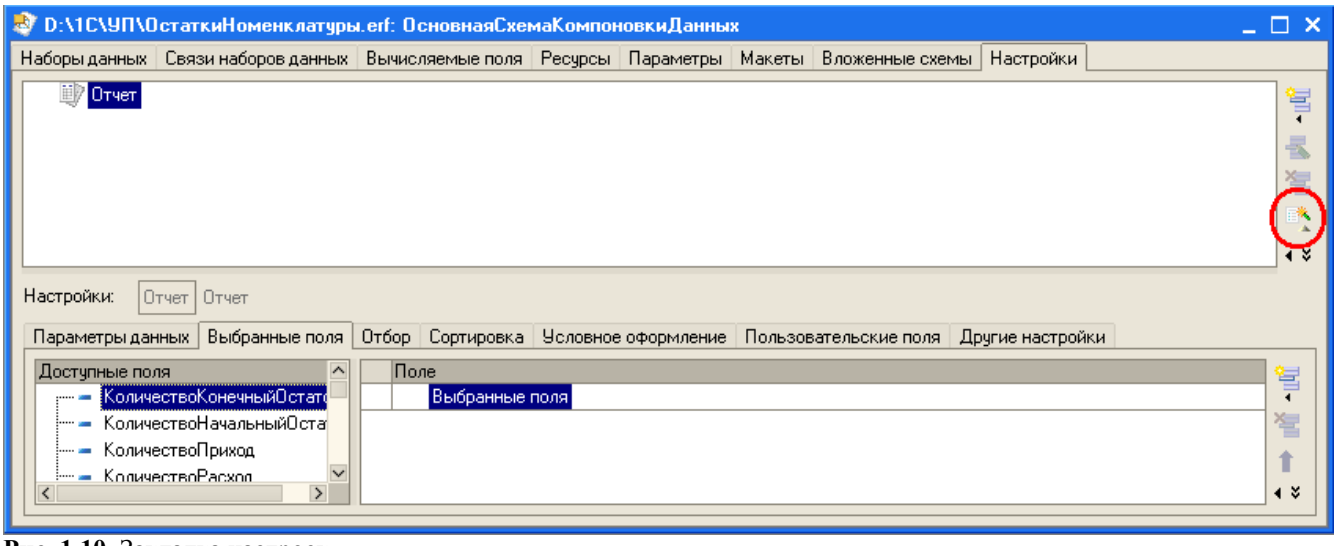


Рис. 1.10. Закладка настроек

На первом шаге выберем тип отчета **Таблица** (рис. 1.11).

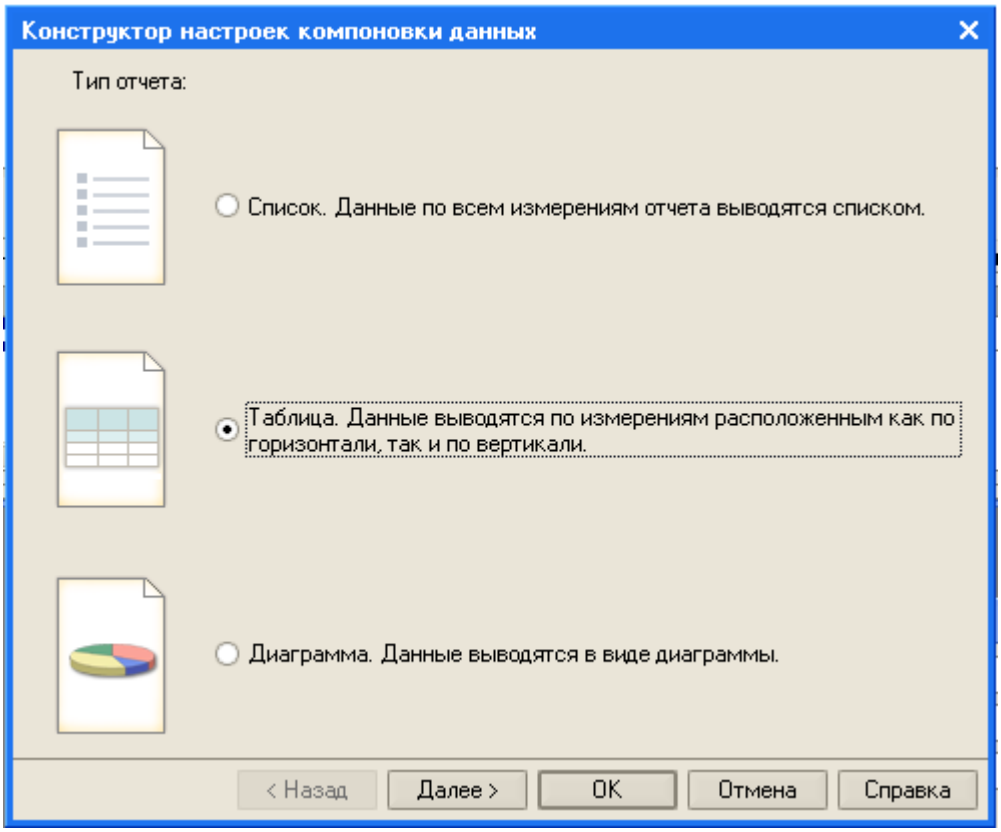


Рис. 1.11. Конструктор настроек. Шаг 1.

Нажмем кнопку **Далее**. На следующем шаге определим поля, выводящиеся в отчет. Перенесем из списка доступных полей все поля и расположим их как указано на рисунке (рис. 1.12).

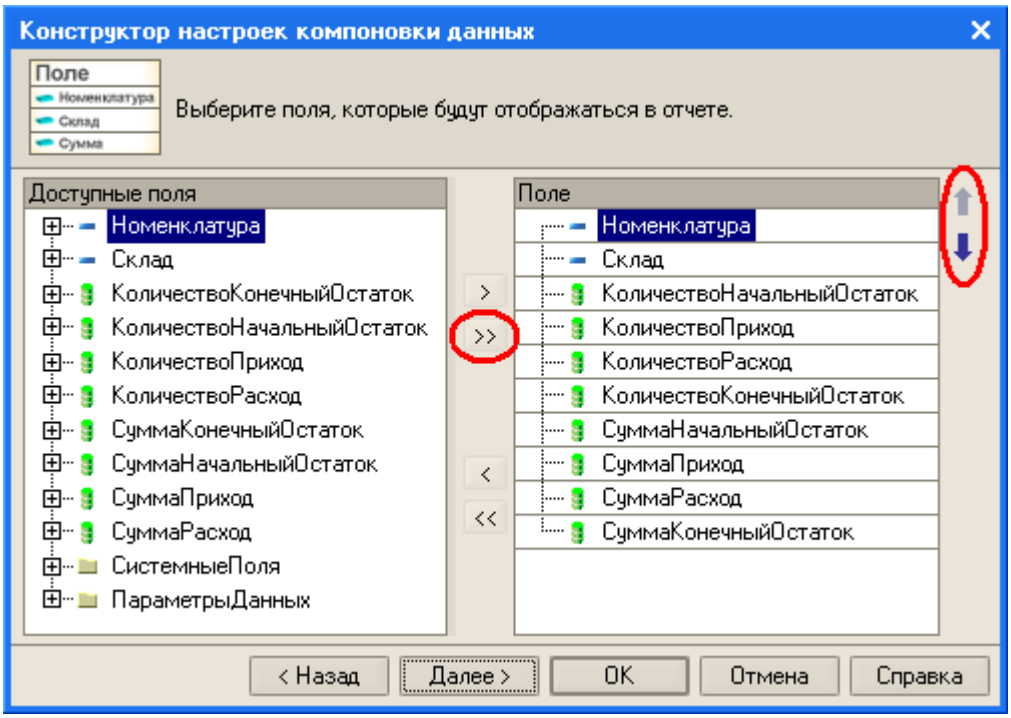


Рис. 1.12. Конструктор настроек. Шаг 2.

Нажмем кнопку **Далее**. На следующем шаге определим поля для группировки строк и колонок таблицы (рис. 1.13).

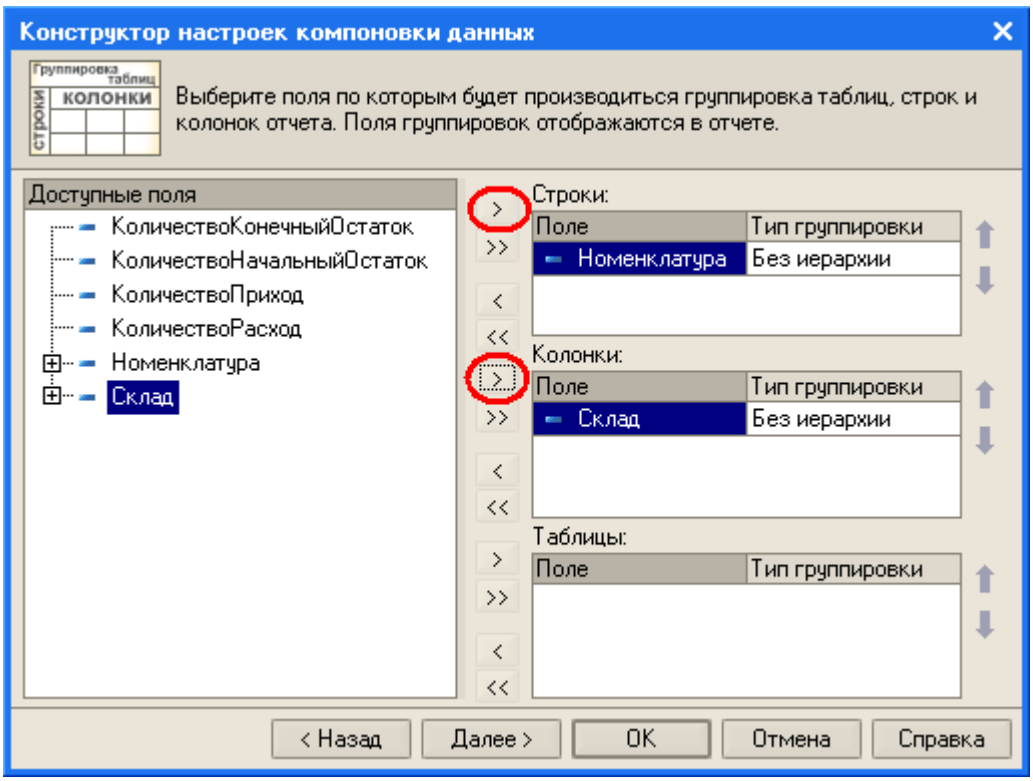


Рис. 1.13. Конструктор настроек. Шаг 3.

Нажмем кнопку **Далее**. Здесь выбираются поля для сортировки данных в отчете. Сортировка нам не важна, поэтому этот шаг пропустим.

Нажмем **ОК**. В окне настроек появляется таблица, в строках которой находится группировка **Номенклатура**, а в колонках – группировка **Склад**, а так же определен список выводимых полей (рис. 1.14). Аналогичный результат можно получить, самостоятельно создавая и настраивая нужные элементы настроек (группировки, таблицы, диаграммы).

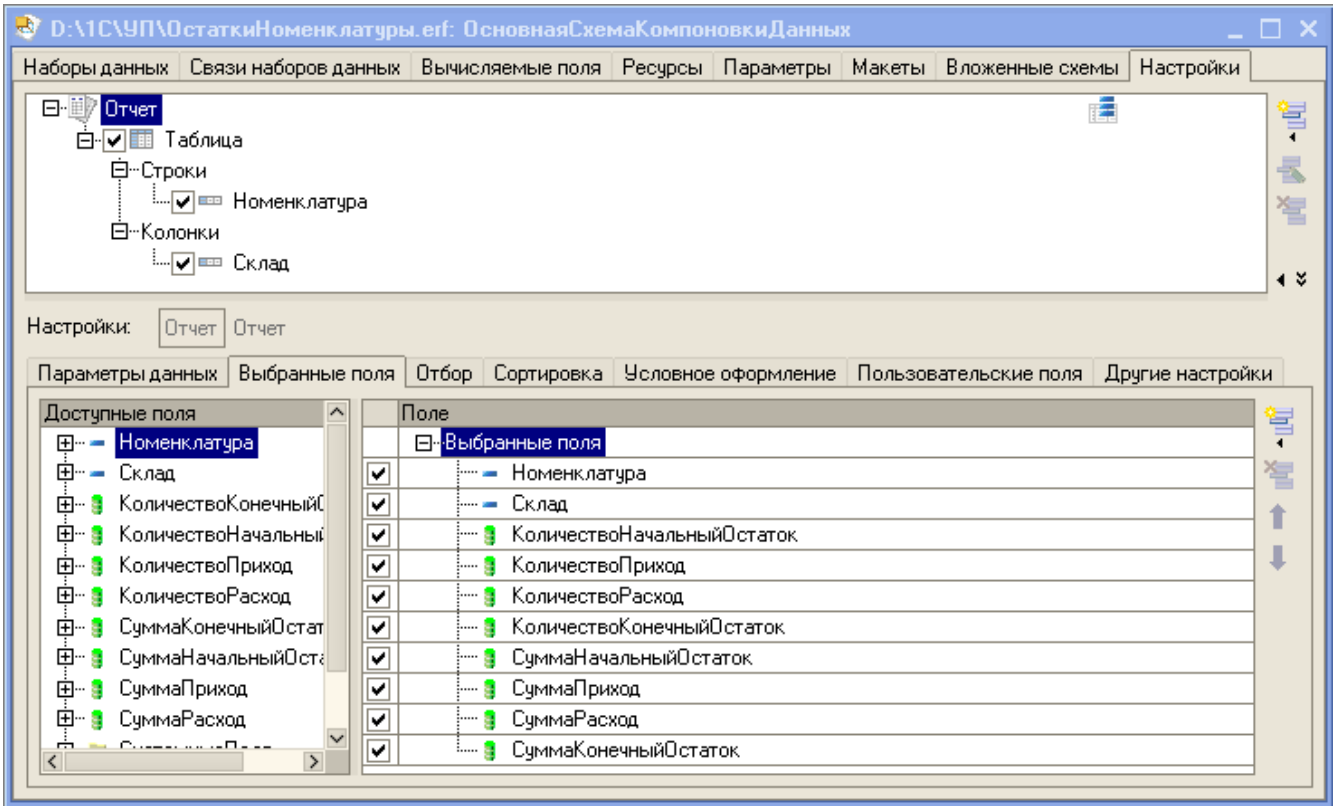
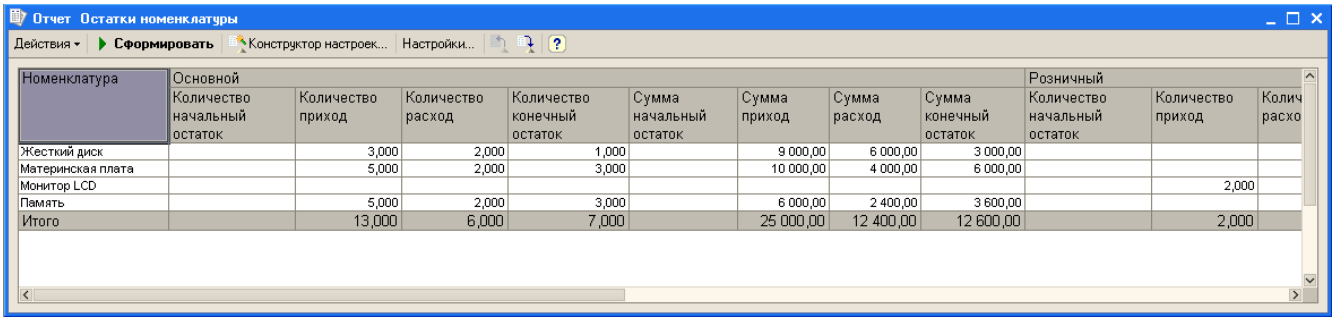


Рис. 1.14. Результат действия конструктора настроек.

Запустим отчет на исполнение в режиме 1С:Предприятие. Данные в отчете представлены в виде таблицы, строки которой сгруппированы по номенклатуре, а колонки – по складам, а так же выведены общие итоги по вертикали и горизонтали (рис. 1.15).



Номенклатура	Основной								Розничный		
	Количество начальный остаток	Количество приход	Количество расход	Количество конечный остаток	Сумма начальный остаток	Сумма приход	Сумма расход	Сумма конечный остаток	Количество начальный остаток	Количество приход	Колич расхо
Жесткий диск		3,000	2,000	1,000		9 000,00	6 000,00	3 000,00			
Материнская плата		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00			
Монитор LCD										2,000	
Память		5,000	2,000	3,000		6 000,00	2 400,00	3 600,00			
Итого		13,000	6,000	7,000		25 000,00	12 400,00	12 600,00		2,000	

Рис. 1.15. Результат отчета.

Настройка отчета в режиме 1С:Предприятие

Для начала сгруппируем ресурсы отчета (Количество и Сумма) по логическому признаку – Начальный остаток, Приход, Расход и Конечный остаток.

Откроем окно настроек отчета, нажав кнопку **Настройки..** в панели инструментов основного окна.. В нем на закладке **Выбранные поля** жмем на кнопку в верхнем правом углу и в появившемся меню выбираем **Новая группа.**

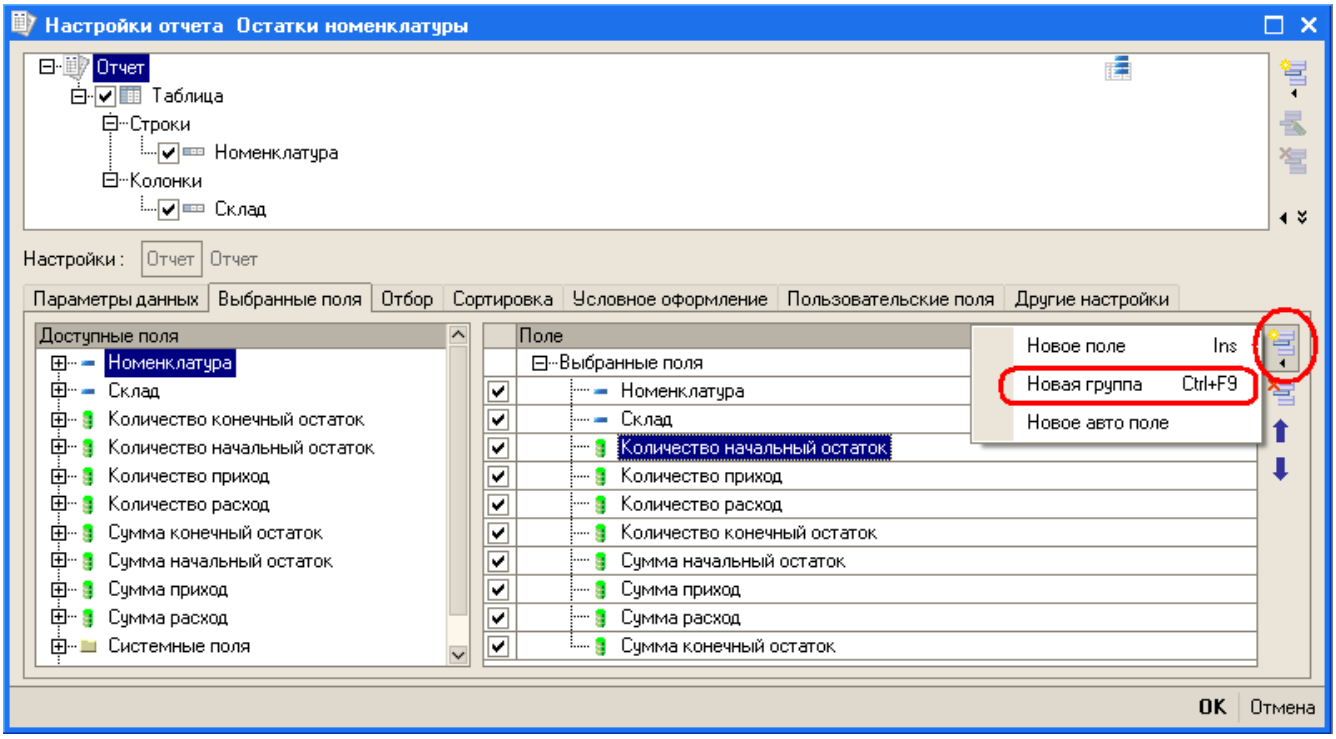


Рис. 1.16. Добавление группы полей.

Создав группу и назвав ее соответствующим образом, перетащим в нее мышкой нужные поля. Для разных групп попробуем задать различное расположение (рис. 1.17).

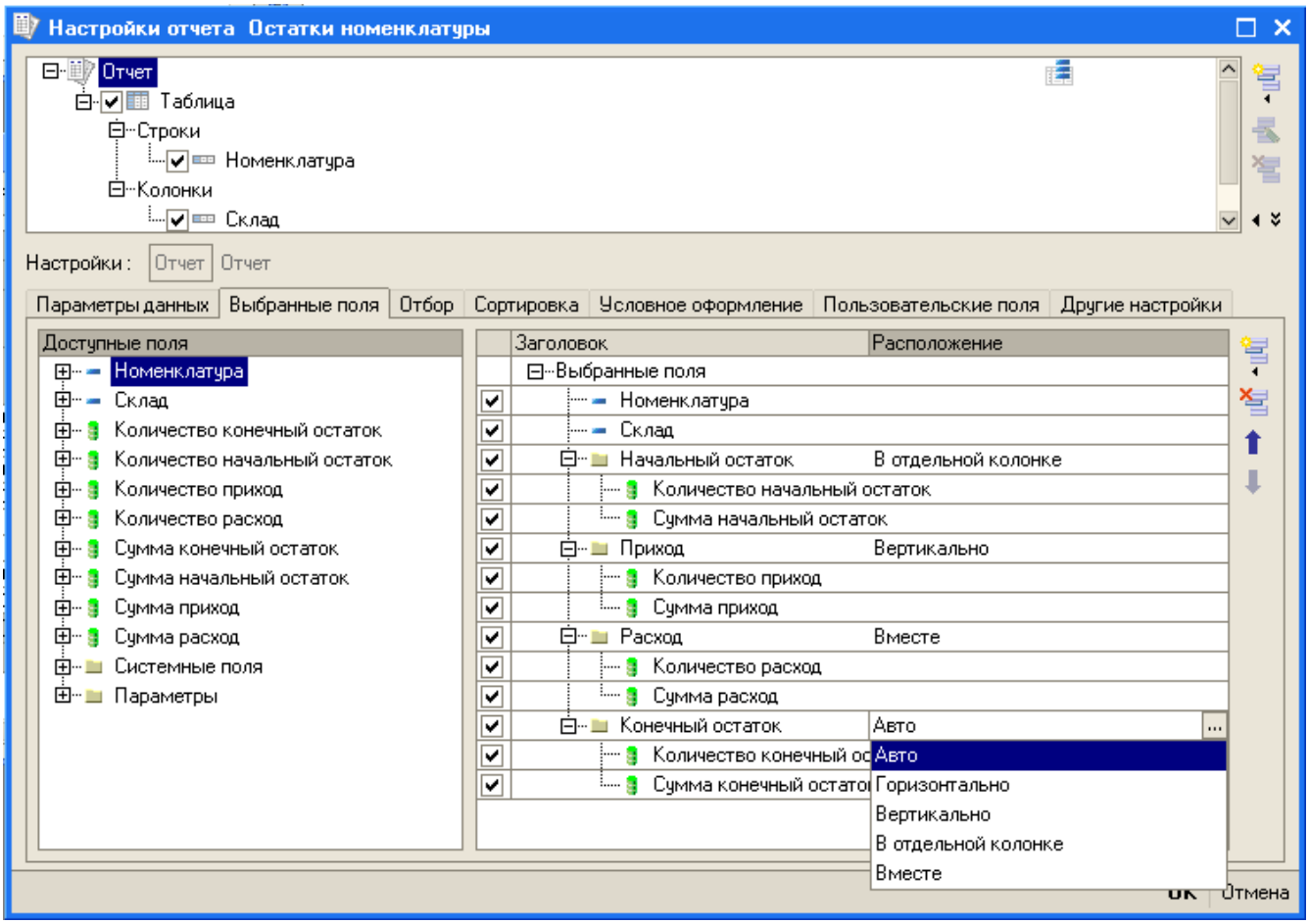


Рис. 1.17. Сгруппированные поля.

Результат должен получиться как на рисунке (рис. 1.18)

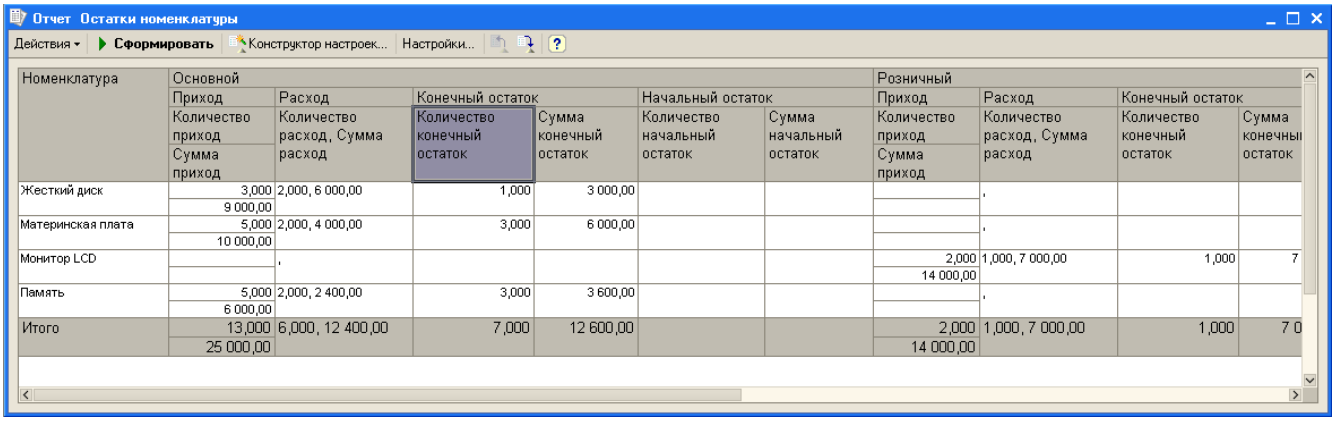


Рис. 1.18. Результат отчета со сгруппированными полями.

Иерархическая структура отчета может быть представлена тремя видами:

- Группировка – для вывода информации в виде линейного списка.
- Таблица – для вывода информации в виде таблицы.
- Диаграмма – для вывода информации в виде диаграммы.

Для графического представления данных продемонстрируем вывод в диаграмму. В окне настроек добавим в структуру отчета элемент **Диаграмма.**

Для добавления нового элемента выделим в дереве структуры отчета корневой элемент **Отчет** и вызовем его контекстное меню. В добавленном элементе **Диаграмма** выделим ветку **Точки** и добавим в нее новую группировку (рис. 1.19).

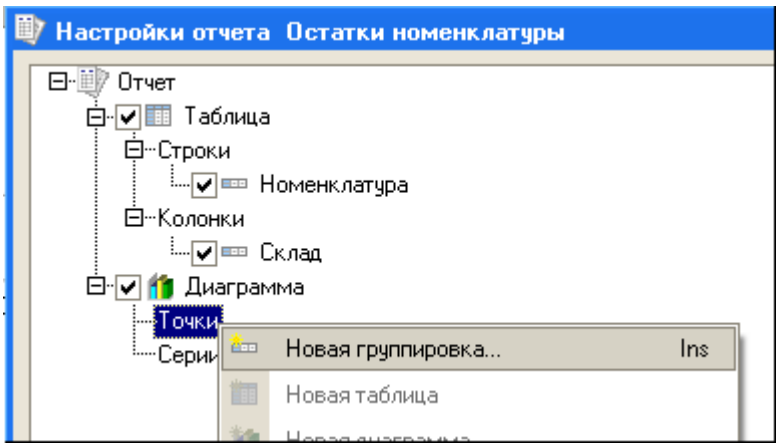


Рис. 1.19. Добавление группировки для точек.

В появившемся окне укажем поле **Склад.** Таким же образом для ветки **Серии** добавим группировку по полю **Номенклатура.** В результате иерархия элементов отчета будет выглядеть так (рис. 1.20).

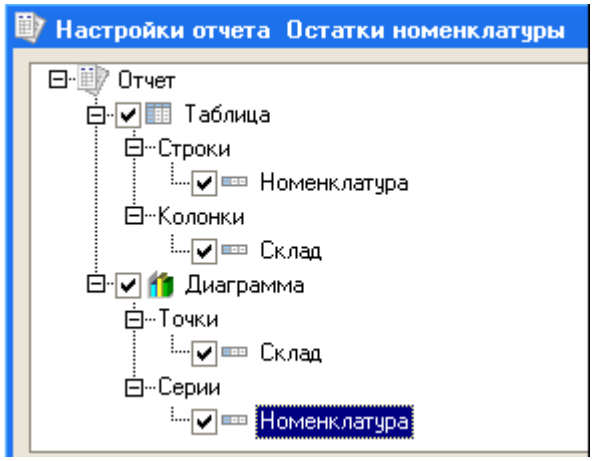


Рис. 1.20. Иерархия элементов отчета.

Осталось определить, каким ресурсом будет пользоваться платформа при выводе значений диаграммы. Для этого установим текущий элемент структуры настроек, нажав на ветку **Диаграмма** в дереве элементов отчета. Перетащим мышкой ресурс **Сумма конечный остаток** в список выбранных полей и снимем галочку использования у элемента **Авто** (рис 1.21).

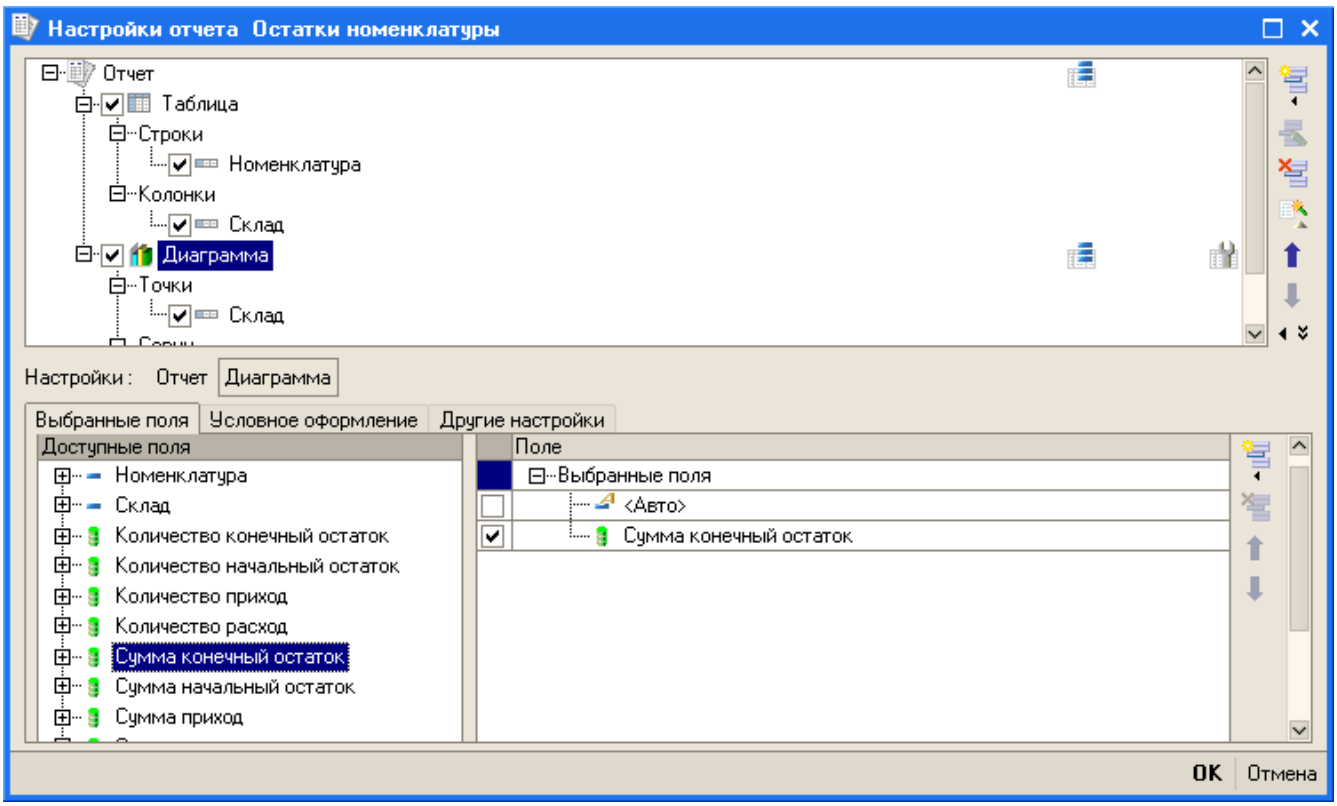


Рис. 1.21. Выбор ресурса диаграммы

На закладке **Другие настройки** выберем тип диаграммы **Гистограмма объемная**, а значение параметра **Отображать таблицу данных** установим в **Истина** (рис. 1.22)**.**

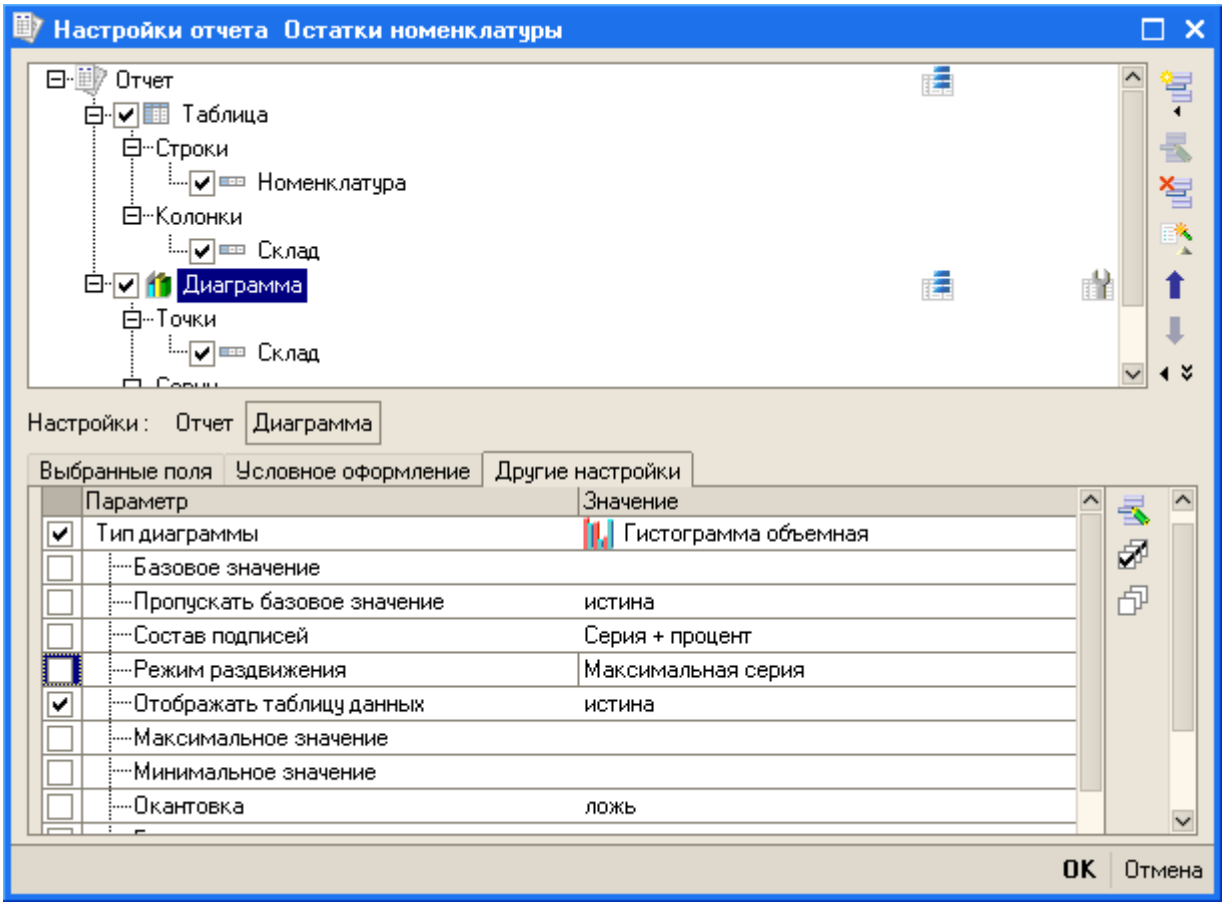


Рис. 1.22. Другие настройки диаграммы

После переформирования отчет примет такой вид (рис. 1.23).

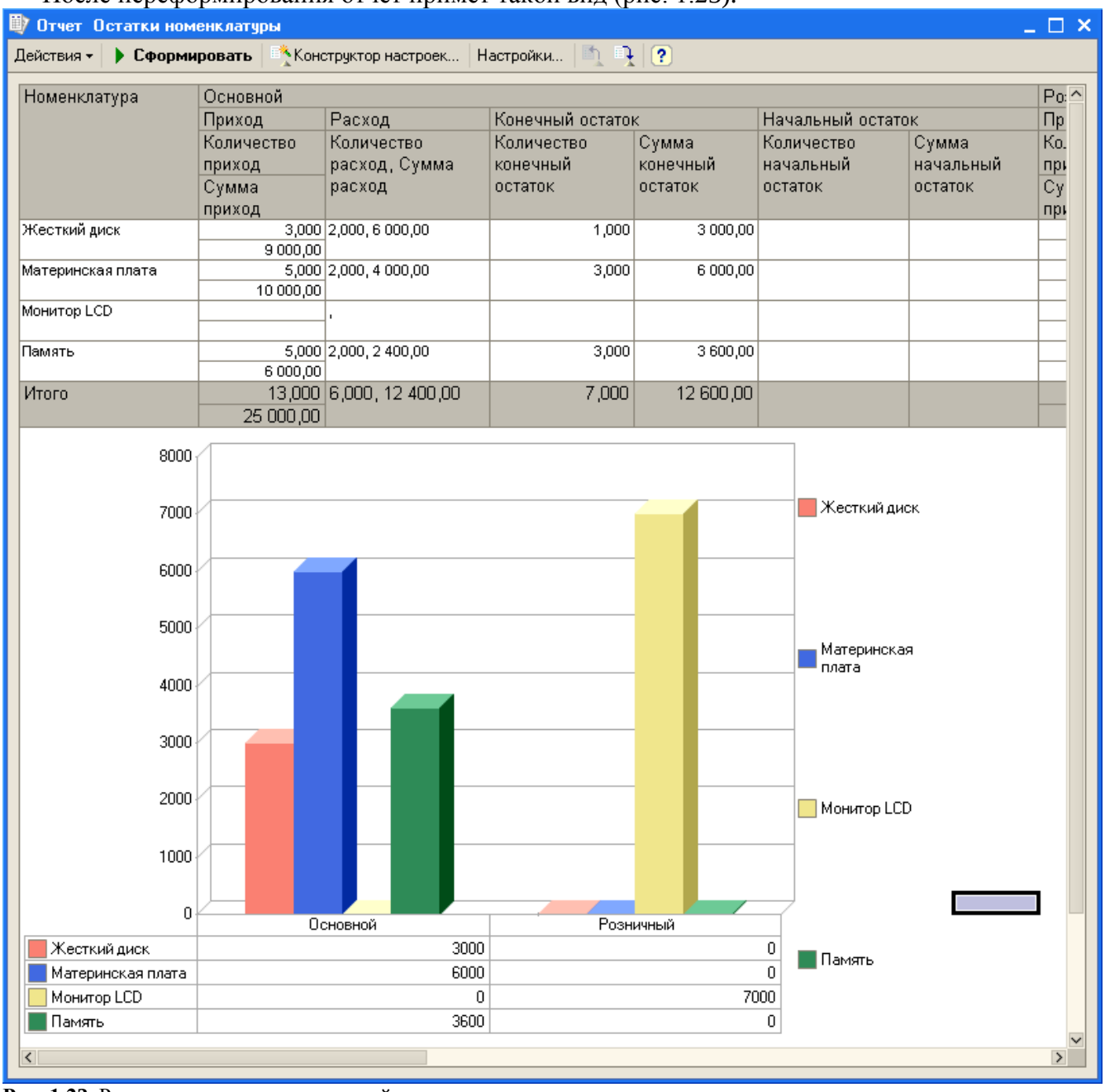


Рис. 1.23. Результат отчета с диаграммой

Все настройки, которые мы сделали в режиме 1С:Предприятие, разработчик может внести и в режиме конфигурирования. В этом случае, эти настройки будут доступны любому пользователю отчета как настройки «по умолчанию».

Настройки, сделанные в режиме 1С:Предприятие, можно использовать и на других рабочих местах. Для этого необходимо воспользоваться кнопками в правом нижнем углу окна иерархии элементов отчета. Сначала настройки можно сохранить в файл с расширением .xml, а потом восстановить из этого файла (рис. 1.24).

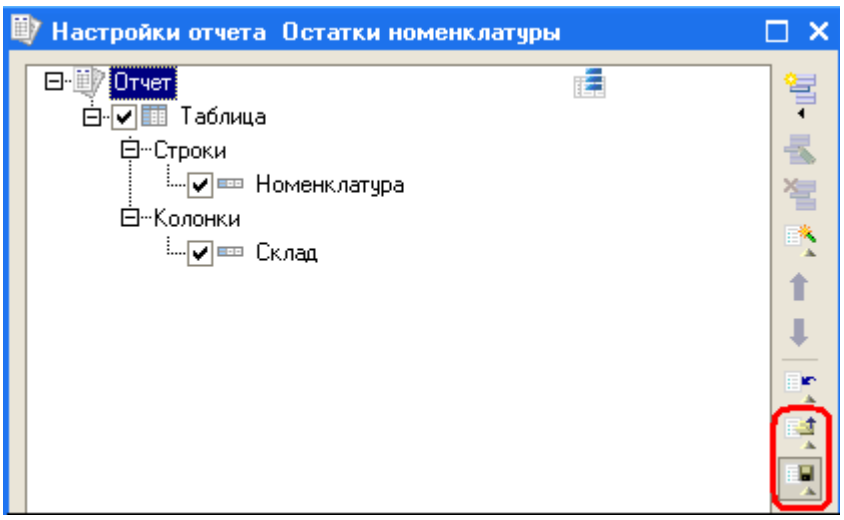


Рис. 1.24. Сохранение и восстановление пользовательских настроек через файл

2. Немного теории. Устройство системы компоновки данных

Упрощенное описание

Сначала система извлекает данные из наборов данных, связывает их. Все полученные данные представлены внутри СКД в виде таблиц, а значит, все их можно вывести построчно в отчет. Это детальные записи.

Но, во-первых, эти данные привязаны к некоторым иерархическим данным (например, в отчет выводятся остатки номенклатуры, где номенклатура содержит многоуровневые группы). Во-вторых, данные могут быть сгруппированы по разрезам учета. Поэтому возникает желание сгруппировать записи отчете по иерархическим группам или по другим группировкам. Для этого система компоновки данных может рассчитывать и добавлять в отчет записи, которые называются итоговыми. Этих записей в явном виде нет в базе данных, но они получаются в результате исполнения отчета, на основании некоторой обработки детальных записей, которую выполняет платформа.

Общее устройство системы компоновки данных

Система компоновки данных состоит из набора элементов, каждый из которых имеет свое декларативное описание, возможность программного доступа и сериализации, то есть сохранения в XML или загрузки из XML. Это позволяет гибче управлять всеми этапами выполнения отчета.

Формирование отчета с использованием системы компоновки данных разбивается на несколько этапов. При этом разбиении на этапы - не «логическое», а «физическое». То есть в каждый из этих этапов разработчик может вмешаться, программно и визуально, выполнять различные этапы компоновки данных в различных модулях и даже на различных компьютерах (рис. 2.1).

Рис. 2.1. Взаимодействие объектов системы компоновки данных

На схеме представлены этапы формирования отчета:

1. Разработчик создает схему компоновки данных и стандартные настройки компоновки, которые содержат полную информацию о компоновке.
2. На основе схемы компоновки данных и настроек компоновки компоновщик макета создает макет компоновки данных, который является уже готовым заданием для процессора компоновки.
3. Процессор компоновки данных в соответствии с макетом компоновки извлекает данные из информационной базы, агрегирует, оформляет эти данные и формирует результат компоновки. Результат компоновки состоит из массива элементов, которые можно получать последовательно, а можно получить целиком.
4. Процессор вывода обрабатывает Результат компоновки и выводит его в различные форматы – табличный документ, таблицу значений или дерево значений.

Процесс разработки отчета с помощью системы компоновки данных и его дальнейшего использования на практике реализуется следующим образом.

Исходя из прикладной задачи, для которой разрабатывается отчет, разработчик создает схему компоновки данных, в которой описывает наборы данных, связи между ними, определяет поля отчета, параметры получения данных и т. п., а также задает стандартные настройки компоновки данных структуру отчета, порядок, отбор и другие.

Пользователь запускает отчет. Затем выполняется программная компоновка данных и вывод результатов компоновки пользователю. Данные в отчете выводятся в соответствии стандартными настройками.

Однако пользователь может самостоятельно изменить настройки отчета и выполнить его. При этом данные будут представлены в соответствии с пользовательскими настройками.

Таким образом, на основе одной схемы компоновки данных может быть получено множество различных отчетов. Применение настроек компоновки данных, созданных разработчиком или измененных пользователем, к схеме компоновки данных и определяет конечный результат.

3. Шаг второй. Работа с параметрами и отборами

В шаге первом мы создали отчет, показывающий остатки номенклатуры на складах. По умолчанию это отчет показывал начальный остаток «с начала учета», то есть всегда ноль, а конечный остаток – на дату последней записи регистра. В жизни, как правило, пользователь интересуется данными о деятельности за определенный период или на определенную дату. Кроме того, у пользователя наверняка возникнет желание посмотреть остатки, например, по определенному складу.

Определенные возможности отборов система нам уже предоставила, и сейчас мы посмотрим, как ими пользоваться. А так же мы научимся добавлять в отчет произвольные параметры и отборы и делать их удобными для использования.

Работа с параметрами стандартными средствами СКД

Откроем схему компоновки данных нашего отчета на закладке **Параметры**.

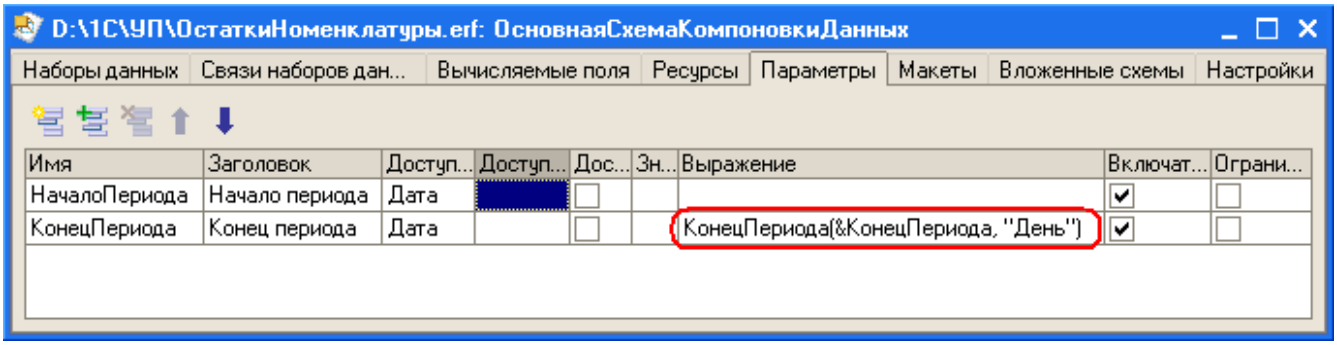


Рис. 3.1. Параметры компоновки данных

Мы видим, что система уже добавила два параметра в список. Это происходит тогда, когда в запросе присутствуют виртуальные таблицы, или параметры, явно указанные в запросе, а так же установлен флажок **Автозаполнение**.

Заметим, что если в отчет передается параметр, содержащий значение типа Дата, то такой параметр содержит и время с точностью до секунды. Однако пользователя, как правило, не интересуют данные с такой точностью. Чтобы пользователь мог вводить даты без времени, изменим тип параметров, выбрав его в колонке **Доступные типы**. В данном случае установим состав даты в значение **Дата** для обоих параметров.

Но в таком случае, задав дату конца периода, например, 31.08.2009, итоги регистра рассчитаются на начало дня 31.08.2009, т.е. данные за 31-е число в отчет не попадут. Чтобы учесть последний день, введем в поле **Выражение** такой текст

```
КонецПериода(&КонецПериода, "День")
```

Это выражение приводит исходную дату к концу какого-либо периода, например, дня.
В результате получим возможность ввода параметров как на рисунке (рис. 3.2)

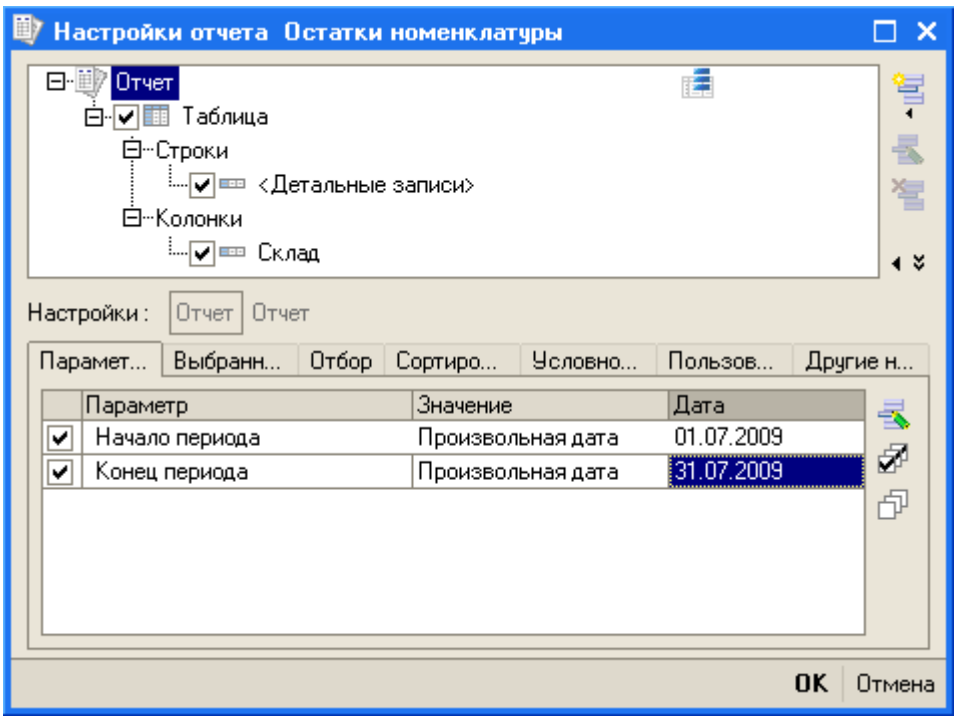


Рис. 3.2. Настройка параметров

Также система компоновки данных позволяет использовать стандартные периоды для формирования отчета.

На закладке **Параметры** добавим параметр с именем **Период** типа **СтандартныйПериод**, а для параметров **НачалоПериода** и **КонецПериода** укажем выражение для расчета и запретим их редактирование пользователем (рис. 3.3).

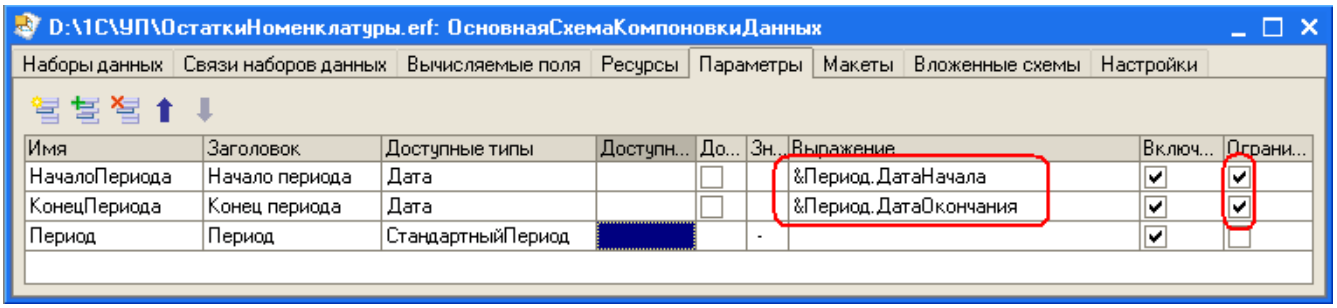


Рис. 3.3. Добавление стандартного периода

Обратим внимание, что свойство **ДатаОкончания** параметра **Период** всегда содержит дату со временем 23:59:59, поэтому не нужно использовать функцию **КонецПериода()**.

Установка периода с использование механизма стандартного периода теперь должна выглядеть как на рисунке (рис. 3.4).

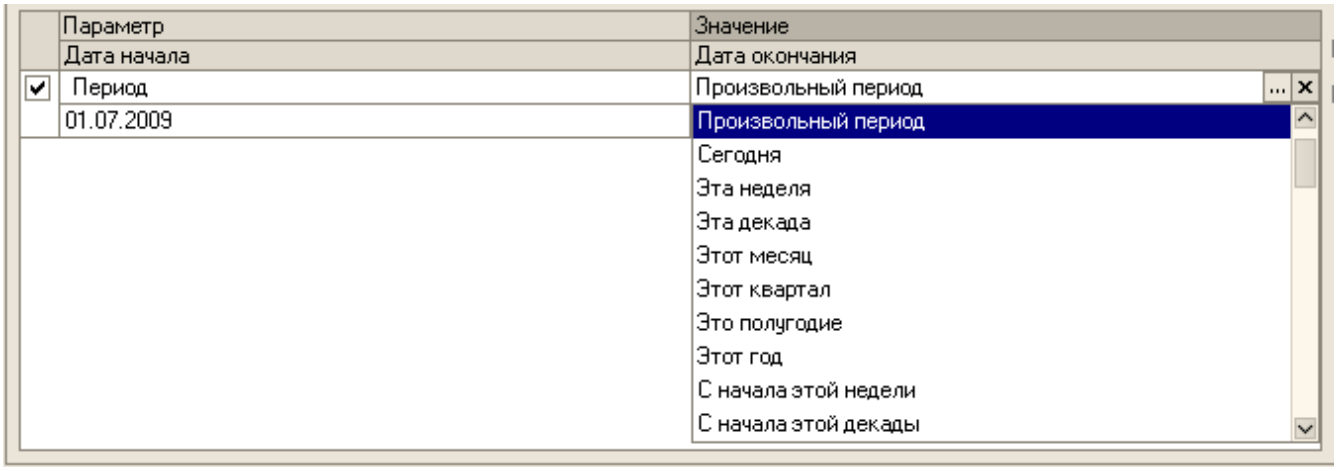


Рис. 3.3. Использование стандартного периода

Заметим, что если задать значение стандартного периода в режиме конфигурирования, реальные значения дат будут определяться в момент использования отчета. Это позволит пользователю, не задавая значения вручную, всегда получать данные, например, за прошлую неделю.

Для повышения удобства работы пользователя имеется возможность скрывать параметры отчета от пользователя используя свойства **Включать в доступные поля** и **Ограничение доступности.**

- **Включать в доступные поля –** если флаг установлен, этот параметр будет включен в список доступных полей и виден в настройках пользователя.
- **Ограничение доступности –** если флаг установлен, этот параметр не будет виден в настройках пользователя на закладке **Параметры.**

Работа с отборами стандартными средствами СКД

Теперь посмотрим, как система компоновки данных умеет фильтровать данные. Например, дадим системе задание вывести в отчет движения по определенному складу. Для этого на закладке отбор в окне настроек добавим поле и зададим его значение (рис. 3.4).

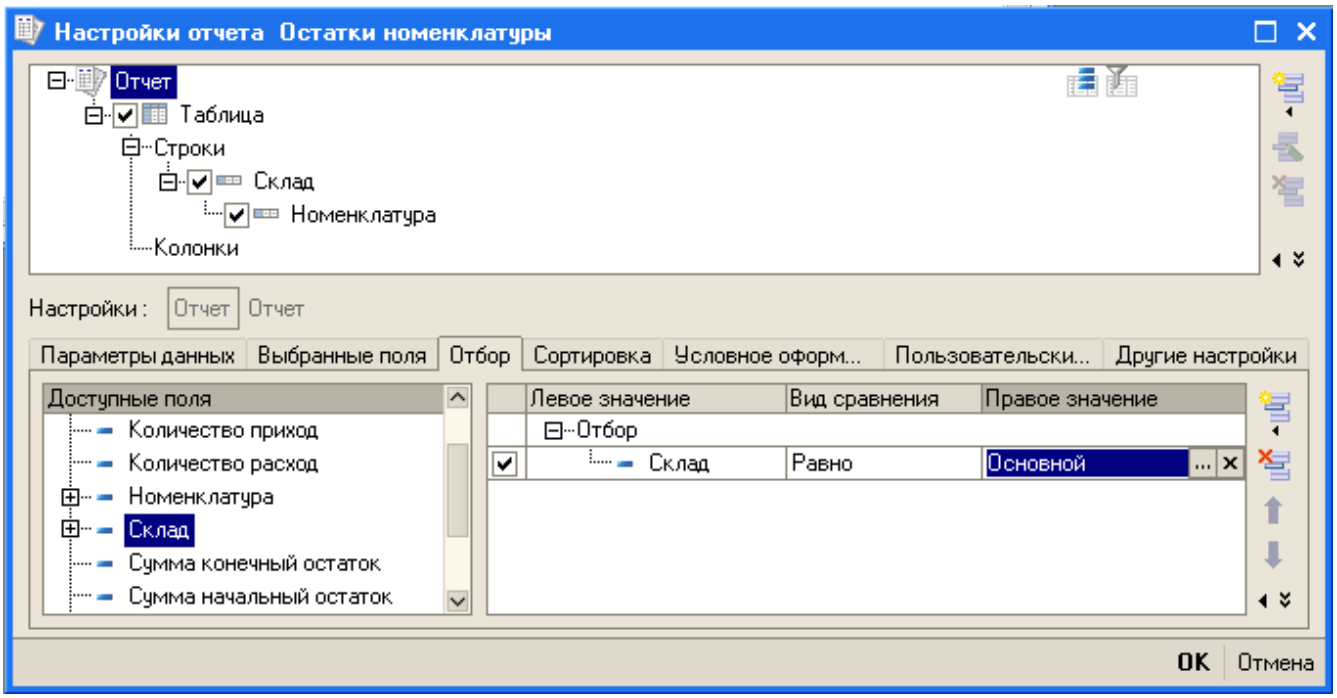
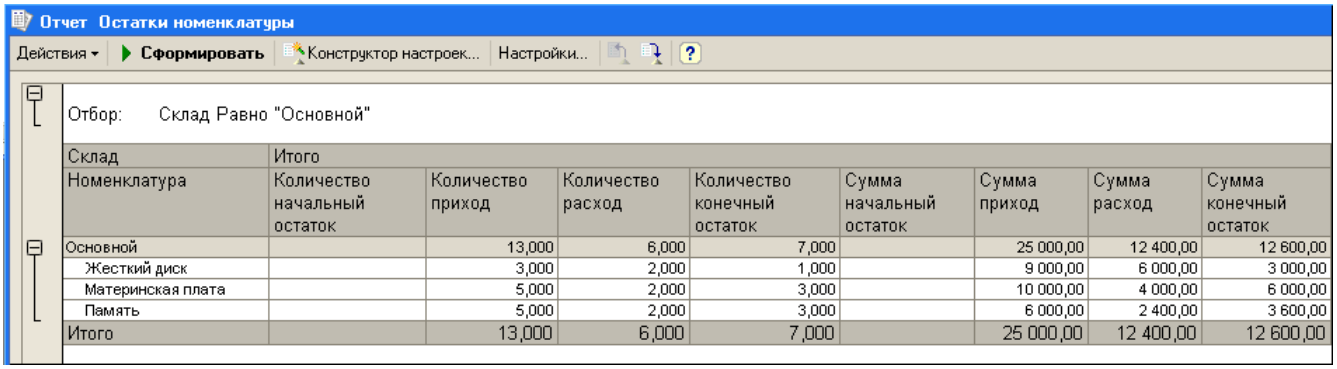


Рис. 3.4. Добавление поля отбора

Результат выполнения будет таким, как на рисунке (рис. 3.5)



Склад / Номенклатура	Итого: Количество начальный остаток	Количество приход	Количество расход	Количество конечный остаток	Сумма начальный остаток	Сумма приход	Сумма расход	Сумма конечный остаток
Основной		13,000	6,000	7,000		25 000,00	12 400,00	12 600,00
Жесткий диск		3,000	2,000	1,000		9 000,00	6 000,00	3 000,00
Материнская плата		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00
Память		5,000	2,000	3,000		6 000,00	2 400,00	3 600,00
Итого		13,000	6,000	7,000		25 000,00	12 400,00	12 600,00

Рис. 3.5. Результат отчета с отбором по складу

А теперь попробуем вывести отчет данные, которые соответствуют условию «Совокупный суммовой остаток на складе больше 5 000 на конечную дату». Добавим в отчет отбор по ресурсу **Сумма конечный остаток** как указано на рисунке (рис. 3.6).

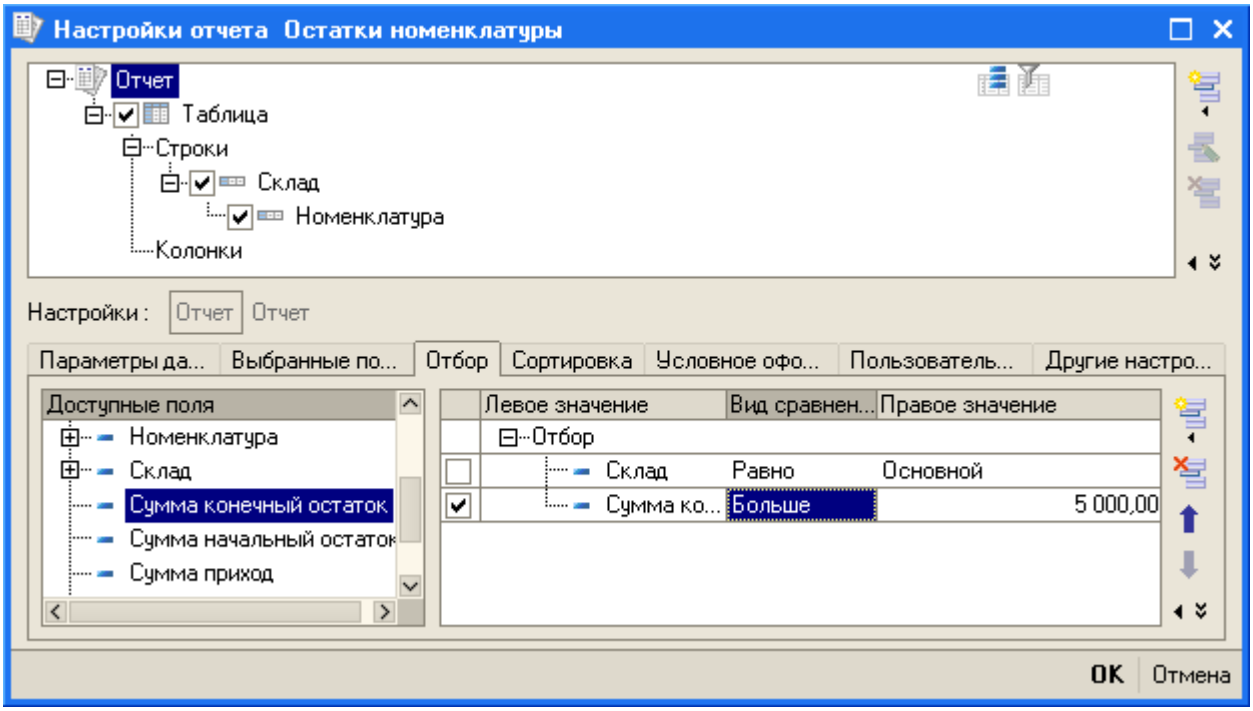


Рис. 3.6. Добавление отбора по ресурсу

В результате у нас в отчете будут только те записи, сумма конечного остатка по которым превышает 5000, а вовсе не то, что предполагалось (рис. 3.7).

Отбор: Сумма конечный остаток Больше "5 000"

Склад / Номенклатура	Итого: Количество начальный остаток	Количество приход	Количество расход	Количество конечный остаток	Сумма начальный остаток	Сумма приход	Сумма расход	Сумма конечный остаток
Основной		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00
Материнская плата		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00
Итого		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00

Рис. 3.7. Результат отчета с отбором по ресурсу

Чтобы выполнить поставленную задачу необходимо применить отбор для определенной группировки, в нашем случае, это группировка **Склад**. Отключим старый отбор, в иерархии элементов выделим группировку **Склад**, в средней панели выделим надпись **Склад** и в окне отбора добавим отбор по нужному ресурсу (рис. 3.8).

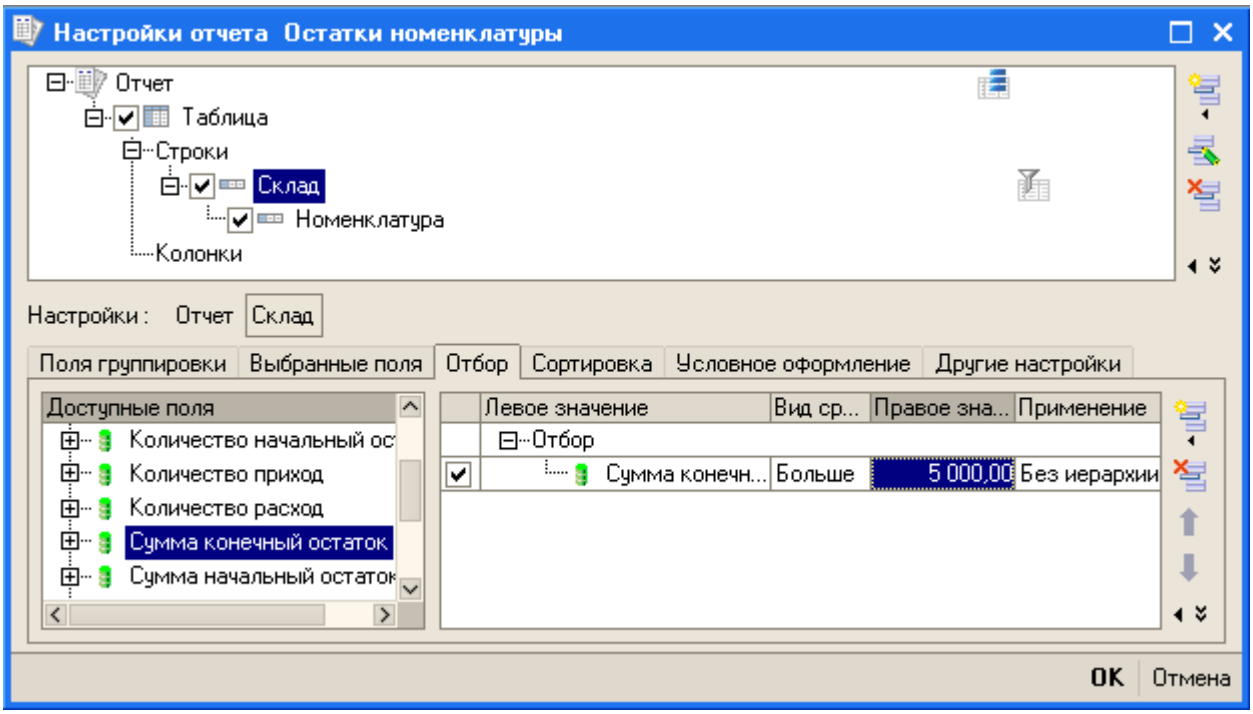


Рис. 3.8. Добавление отбора по ресурсу для группировки

Теперь результат отчета принял нужный вид, в него попали все данные по складу **Основной**, так как на складе **Розничный** сумма конечного остатка меньше требуемой (рис 3.9).

Склад	Итого							
Номенклатура	Количество начальный остаток	Количество приход	Количество расход	Количество конечный остаток	Сумма начальный остаток	Сумма приход	Сумма расход	Сумма конечный остаток
Отбор: Сумма конечный остаток Больше "5 000"								
Основной		13,000	6,000	7,000		25 000,00	12 400,00	12 600,00
Жесткий диск		3,000	2,000	1,000		9 000,00	6 000,00	3 000,00
Материнская плата		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00
Память		5,000	2,000	3,000		6 000,00	2 400,00	3 600,00
Итого		13,000	6,000	7,000		25 000,00	12 400,00	12 600,00

Рис. 3.9. Результат отчета с отбором по ресурсу для группировки Склад

Рассмотрим еще одну интересную возможность отборов в СКД, а именно групповые условия.

Предположим, нас интересует, какие позиции залеживаются на складе. Сформулируем условие так: «Номенклатура, по которой был приход за период и остатки по которой на конец периода более 3 единиц или сумма остатка более 5000 рублей».

Для того чтобы ввести в отбор логическое «ИЛИ» необходимо добавить группу «Или». В панели отбора вызовем контекстное меню и выберем пункт **Новая группа** (рис. 3.10).

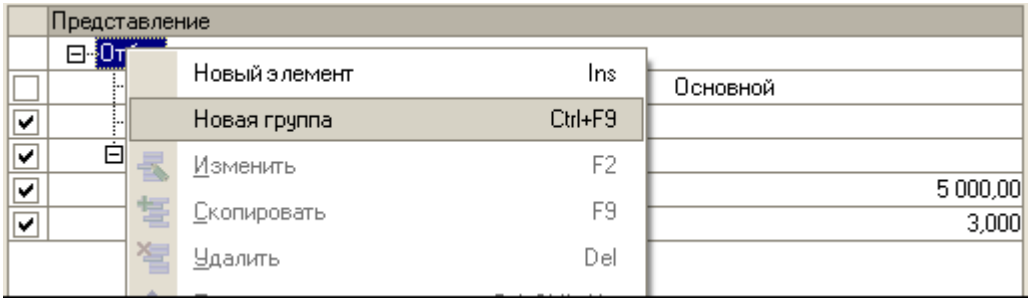
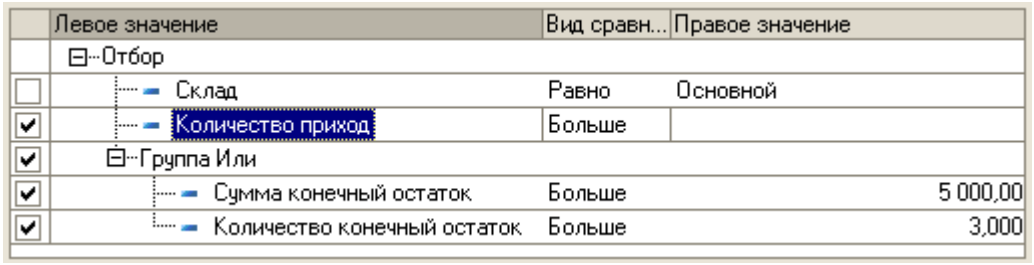


Рис. 3.10. Добавление новой логической группы в отбор

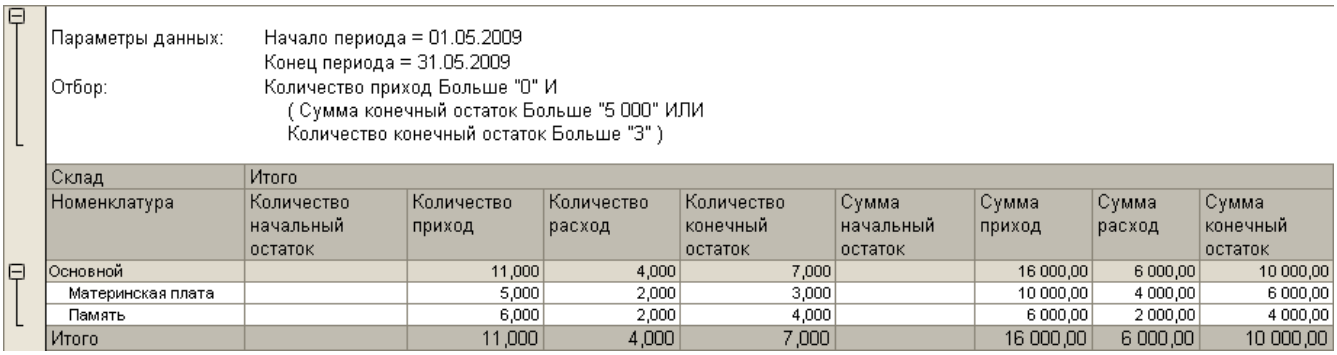
В новом поле выберем значение **Группа Или**. Выделив эту группу и используя контекстное меню, добавим в эту группу условия «Количество конечный остаток больше 3» и «Сумма конечный остаток больше 5000». Далее в корень отбора добавим условие «Количество приход больше 0». Панель отбора в результате будет выглядеть примерно так (рис. 3.11).



Левое значение	Вид сравн...	Правое значение
Отбор		
Склад	Равно	Основной
Количество приход	Больше	
Группа Или		
Сумма конечный остаток	Больше	5 000,00
Количество конечный остаток	Больше	3,000

Рис. 3.11. Отбор с использованием группы Или

В результате получим такой отчет (рис. 3.12)



Параметры данных: Начало периода = 01.05.2009
Конец периода = 31.05.2009
Отбор: Количество приход Больше "0" И
(Сумма конечный остаток Больше "5 000" ИЛИ
Количество конечный остаток Больше "3")

Склад / Номенклатура	Итого: Количество начальный остаток	Количество приход	Количество расход	Количество конечный остаток	Сумма начальный остаток	Сумма приход	Сумма расход	Сумма конечный остаток
Основной		11,000	4,000	7,000		16 000,00	6 000,00	10 000,00
Материнская плата		5,000	2,000	3,000		10 000,00	4 000,00	6 000,00
Память		6,000	2,000	4,000		6 000,00	2 000,00	4 000,00
Итого		11,000	4,000	7,000		16 000,00	6 000,00	10 000,00

Рис. 3.12. Результат отчета с использование групповых условий

Установка параметров средствами встроенного языка

До сих пор для настройки нашего отчета мы использовали только штатные средства системы компоновки данных и убедились, что возможности их достаточно широки.

Но представим, что пользователь привык задавать период формирования отчета в самой форме отчета и требует этого от нашего отчета.

Сначала надо создать форму отчета. Для этого надо щелкнуть значок открытия у реквизита **Основная форма внешнего отчета** формы настройки отчета. В открывшемся конструкторе жмем кнопку **Готово** (рис 3.13).

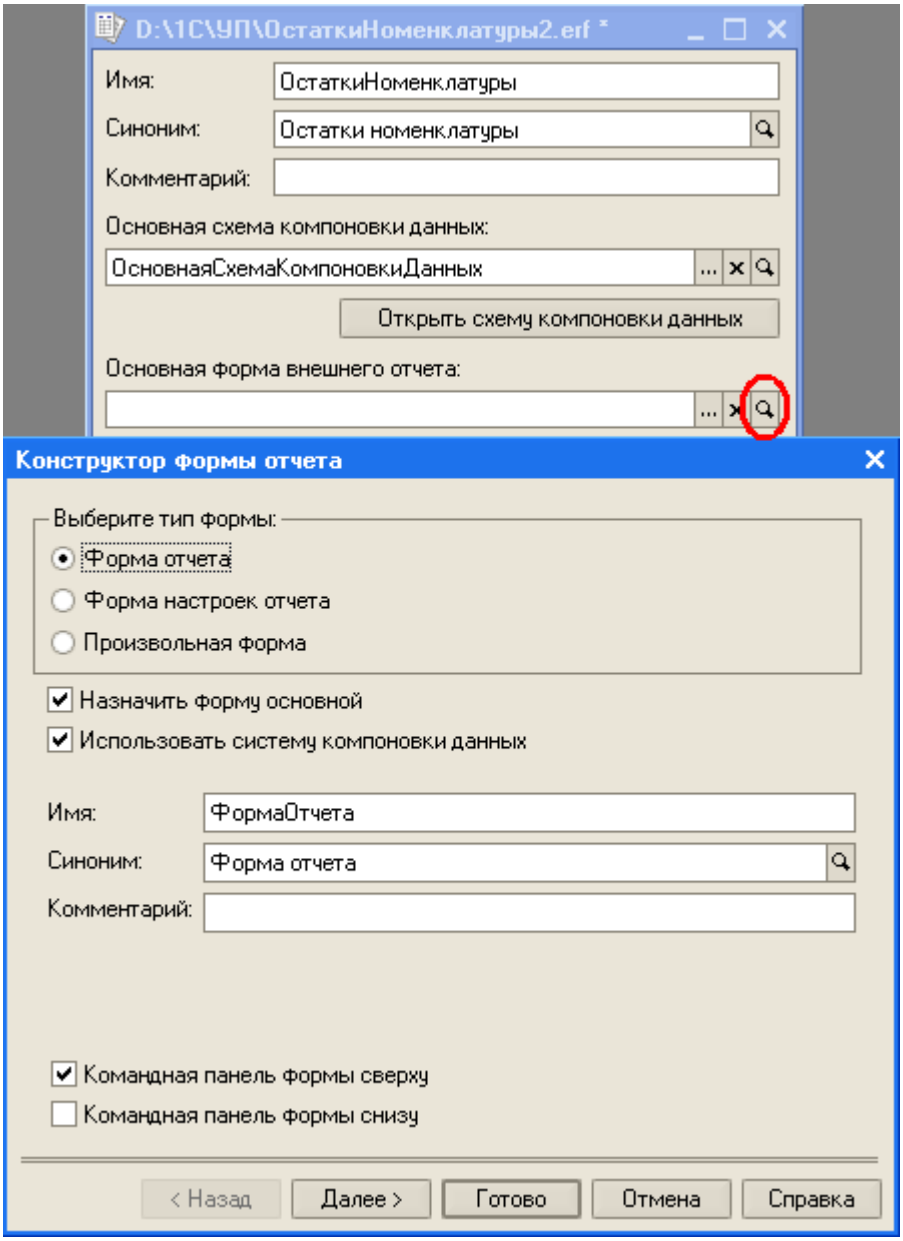


Рис. 3.13. Создание формы отчета

На форме отчета выше табличного документа **Результат** поместим элемент управления **Выбор периода** с параметрами, предлагаемыми по умолчанию (рис 3.14).

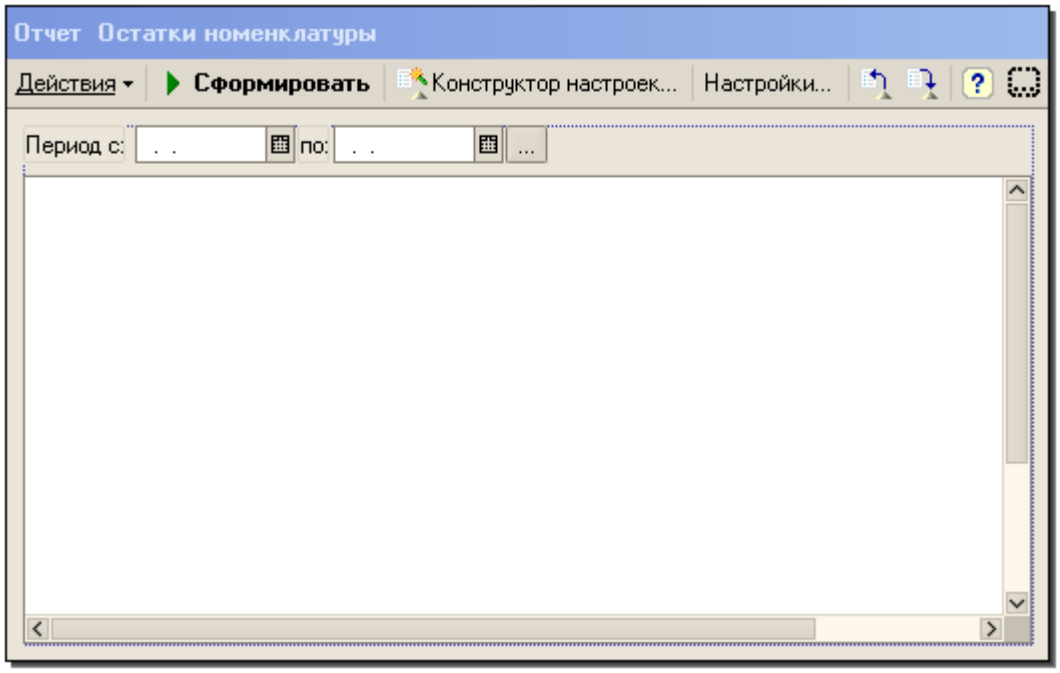


Рис. 3.14. Добавление элемента управления «Выбор периода»

Далее в модуль формы добавим следующий код, в том числе скорректируем содержимое процедуры **ВыбПериодНажатие()**

```
    Процедура УстановитьПараметрыПериода()

        Параметр = КомпоновщикНастроек.Настройки.ПараметрыДанных.НайтиЗначениеПараметра(Новый
ПараметрКомпоновкиДанных("КонецПериода"));;
        Параметр.Значение = КонецДня(КонПериода);
        Параметр.Использование = Истина;

        Параметр = КомпоновщикНастроек.Настройки.ПараметрыДанных.НайтиЗначениеПараметра(Новый
ПараметрКомпоновкиДанных("НачалоПериода"));;
        Параметр.Значение = НачПериода;
      Параметр.Использование = Истина;

    КонецПроцедуры

    Процедура ВыбПериодНажатие(Элемент)

        НастройкаПериода = Новый НастройкаПериода;
        НастройкаПериода.УстановитьПериод(НачПериода, ?(КонПериода='0001-01-01', КонПериода,
КонецДня(КонПериода)));
        НастройкаПериода.РедактироватьКакИнтервал = Истина;
        НастройкаПериода.РедактироватьКакПериод = Истина;
        НастройкаПериода.ВариантНастройки = ВариантНастройкиПериода.Период;
        Если НастройкаПериода.Редактировать() Тогда
                НачПериода = НастройкаПериода.ПолучитьДатуНачала();
                КонПериода = НастройкаПериода.ПолучитьДатуОкончания();
        КонецЕсли;

        УстановитьПараметрыПериода();

    КонецПроцедуры

    Процедура ПриОткрытии()

        УстановитьПараметрыПериода();

    КонецПроцедуры

    Процедура НачПериодаПриИзменении(Элемент)
        УстановитьПараметрыПериода();
    КонецПроцедуры

    Процедура КонПериодаПриИзменении(Элемент)
        УстановитьПараметрыПериода();
    КонецПроцедуры
```

Привяжем процедуры **КонПериодаПриИзменении()** и **НачПериодаПриИзменении()** к соответствующим обработчикам событий элементов **НачПериода** и **КонПериода.** Теперь наш отчет позволяет выбирать период непосредственно в форме, как и хотелось консервативному пользователю (рис 3.15).

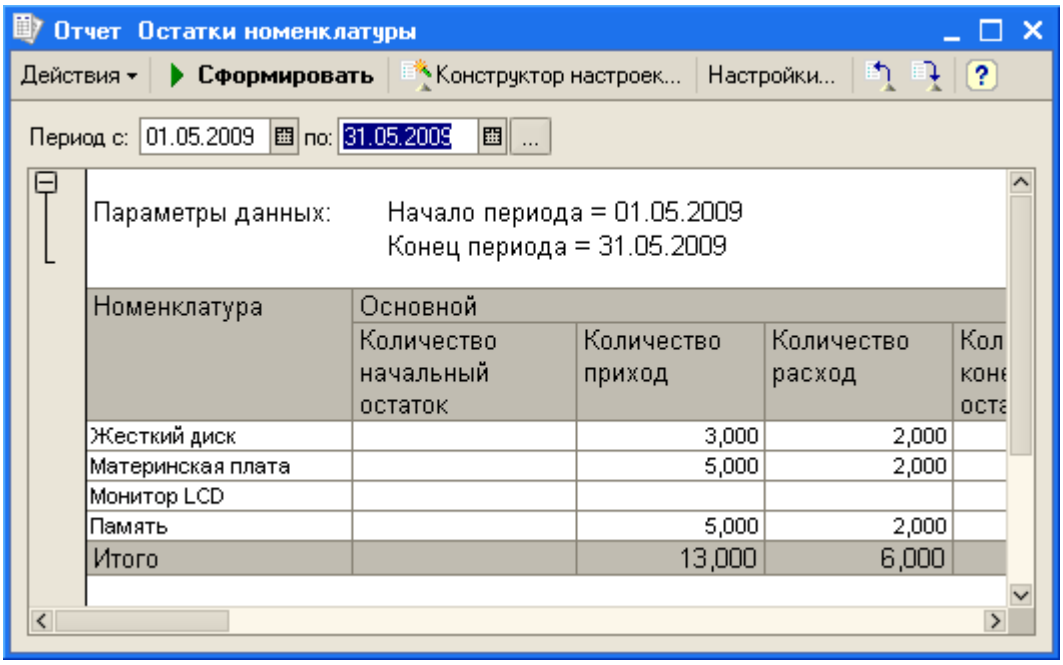


Рис. 3.15. Отчет с удобным выбором периода

Вот таким образом можно программно установить значение параметра в СКД:

```
Параметр = КомпоновщикНастроек.Настройки.ПараметрыДанных.НайтиЗначениеПараметра(Новый
    ПараметрКомпоновкиДанных(<Имя параметра>));;
Параметр.Значение = <Значение параметра>;
Параметр.Использование = Истина;
```

где **КомпоновщикНастроек** – свойство объекта отчета, т.е. к нему можно обратиться и извне.

Работа с отборами средствами встроенного языка

Сначала просто выведем общие отборы отчета в отдельную панель на форме. Для этого поместим на форму элемент управления **Табличное поле**, и свяжем его с данными **КомпоновщикНастроек.Настройки.Отбор** (рис 3.16).

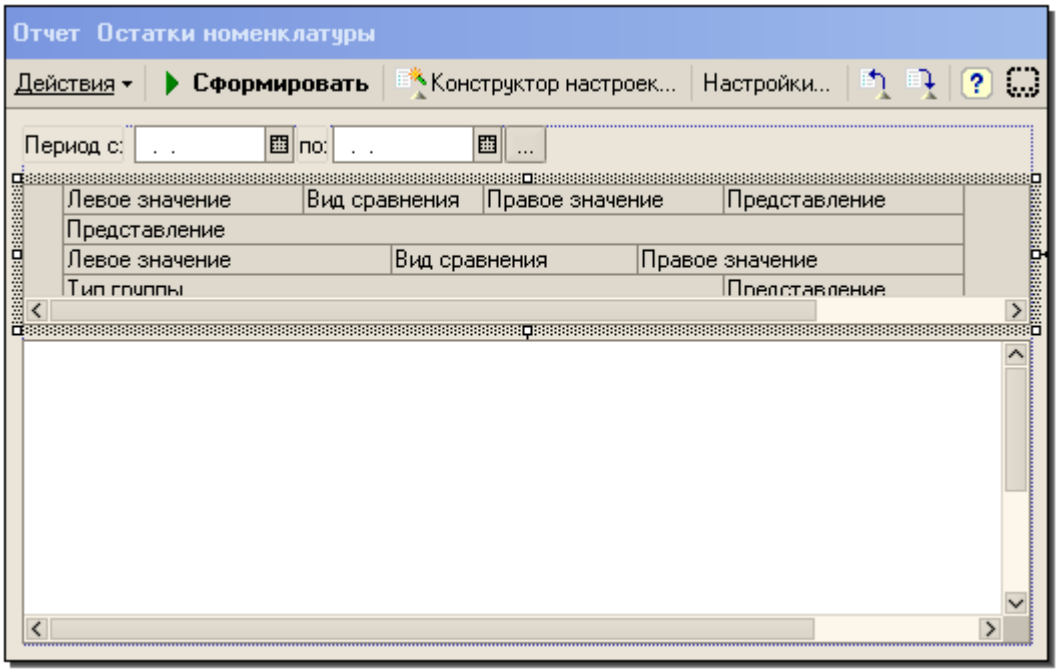


Рис. 3.16. Добавление элемента «Табличное поле»

Теперь отборы в отчете можно задавать непосредственно в форме (рис 3.17). Отметим, что отборы, задаваемые через это поле, являются глобальными и применяются ко всему отчету.

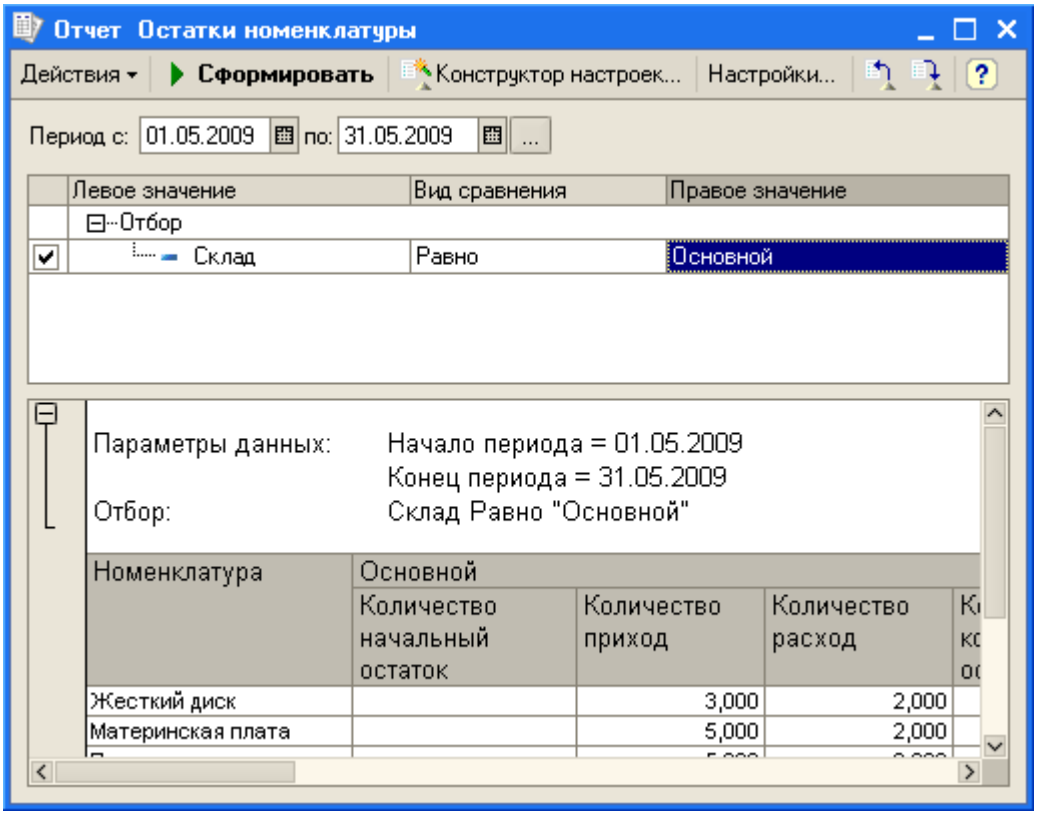


Рис. 3.17. Отчет с удобным выбором периода и установкой отборов

Теперь представим, что пользователь настоял на том, чтобы отбор по складу сделать еще проще: разместить реквизит в форме.

На форме отчета рядом с выбором периода поместим элемент управления **Поле ввода** с типом данных **СправочникСсылка.Склад** (рис 3.18).

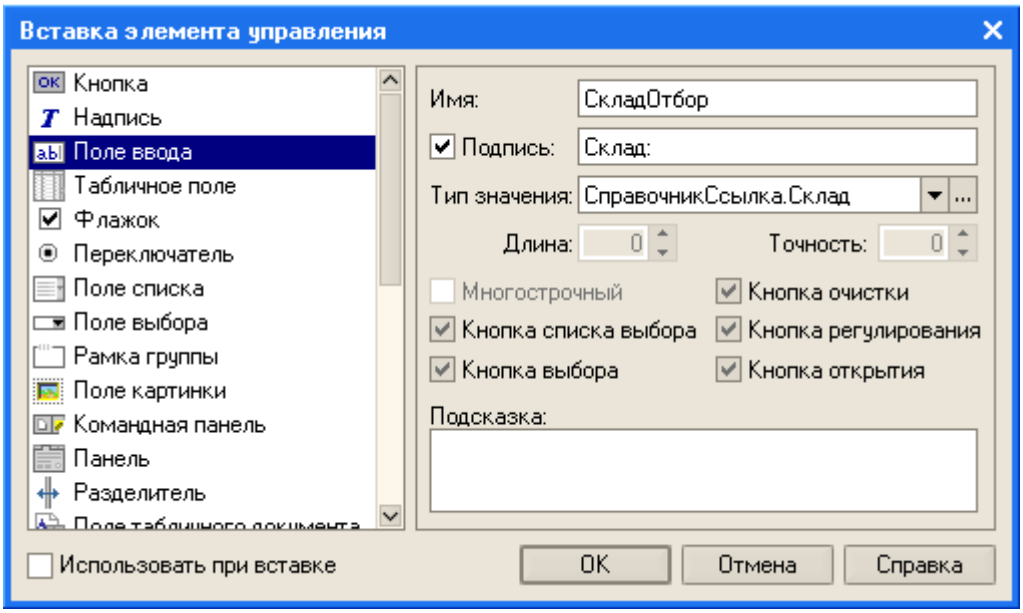


Рис. 3.18. Свойства поля ввода

В модуль формы добавим обработчик события ПриИзменении только что добавленного поля:

```
Процедура СкладОтборПриИзменении(Элемент)

    ПолеСклад = Новый ПолеКомпоновкиДанных("Склад");

    Для каждого Элемент Из КомпоновщикНастроек.Настройки.Отбор.Элементы Цикл
      Если Элемент.ЛевоеЗначение = ПолеСклад Тогда
        ЭлементОтбора = Элемент;
          Прервать;
      КонецЕсли;
    КонецЦикла;

    Если ЭлементОтбора = Неопределено Тогда
        ЭлементОтбора =
    КомпоновщикНастроек.Настройки.Отбор.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
    КонецЕсли;

    ЭлементОтбора.ВидСравнения = ВидСравненияКомпоновкиДанных.Равно;
    ЭлементОтбора.ЛевоеЗначение = ПолеСклад;
    ЭлементОтбора.ПравоеЗначение = СкладОтбор;
    ЭлементОтбора.Использование = НЕ СкладОтбор.Пустая();

КонецПроцедуры
```

Здесь мы сначала проверяем, есть ли в составе отбора поле **Склад**, и если есть, используем его, а если нет, создаем и добавляем в коллекцию **КомпоновщикНастроек.Настройки.Отбор.Элементы**.

После чего делать отбор по складу становится совсем просто (рис. 3.19).

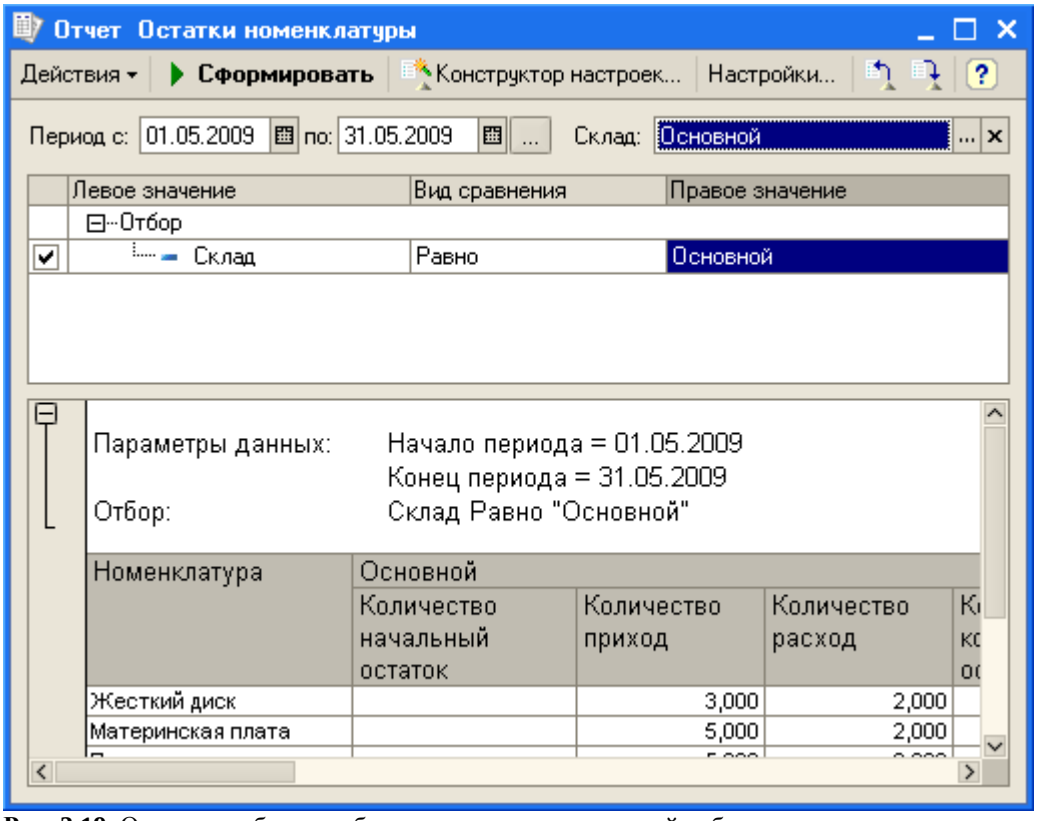


Рис. 3.19. Отчет с удобным выбором периода и установкой отбора по складу

Таким же образом можно задавать отбор извне отчета, например, вызвав этот отчет из формы справочника «Склады».

4. Шаг третий. Работа с различными наборами данных. Внешние наборы данных.

Поставим довольно сложную для учебного примера задачу. Создать отчет, выводящий как обычную задолженность контрагентов, так и просроченную, а так же количество дней просрочки, причем пользователь может видеть задолженность как в целом по контрагенту, так и по документам, ее образующим. Сложность задачи в том, что если в регистре взаиморасчетов не ведется учет в разрезе документов, дату возникновения задолженности приходится рассчитывать методом «ретроспективного анализа», т.е. обратным перебором цепочки документов, образующих задолженность. Тем не менее, эту задачу довольно часто ставят пользователи торгующих предприятий.

Создадим новый внешний отчет, «Задолженность контрагентов». Добавим реквизит отчета **ДатаОтчета** типа **Дата.** Создадим схему компоновки данных. Добавим набор данных типа **Запрос,** назовем его **ОстаткиВзаиморасчетов.** Текст запроса будет таким

```
ВЫБРАТЬ
    ВзаиморасчетыКонтрагентов.Контрагент
ИЗ
    РегистрНакопления.ВзаиморасчетыКонтрагентов.Остатки КАК ВзаиморасчетыКонтрагентов
ГДЕ
    ВзаиморасчетыКонтрагентов.СуммаОстаток > 0
```

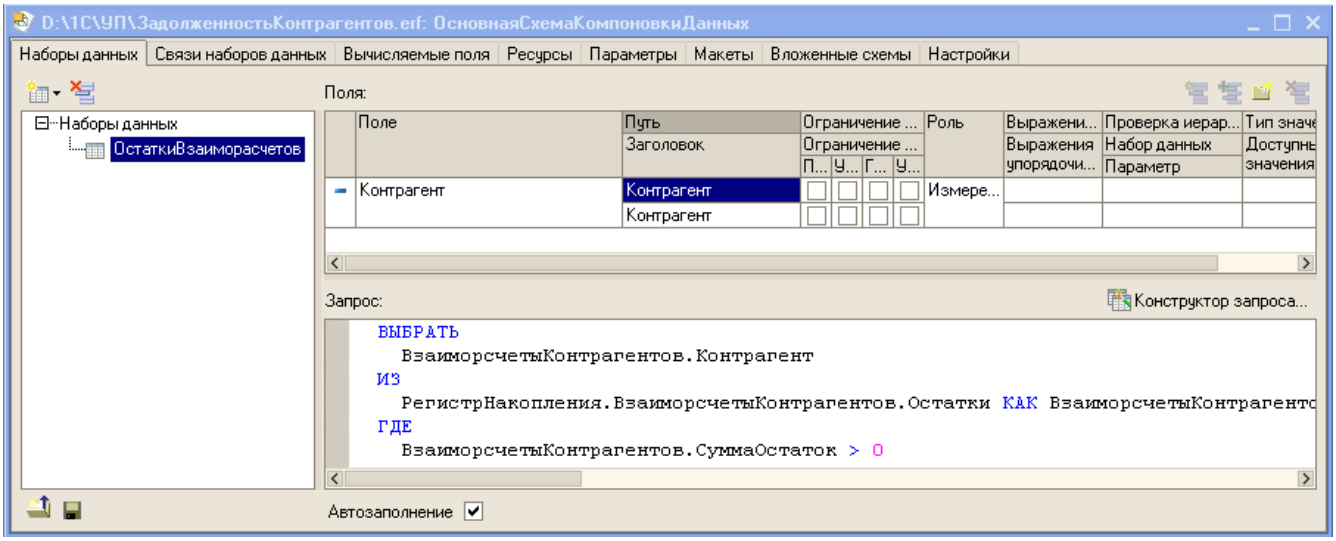


Рис. 4.1. Добавление набора данных типа Запрос

То есть первый набор даст нам список контрагентов, у которых есть остаток долга на заданную дату. Добавим второй набор данных типа **Объект**, назовем его **ВзаиморасчетыПоДокументам.** Так как запрос мы не задаем, поля этого набора придется добавить вручную.

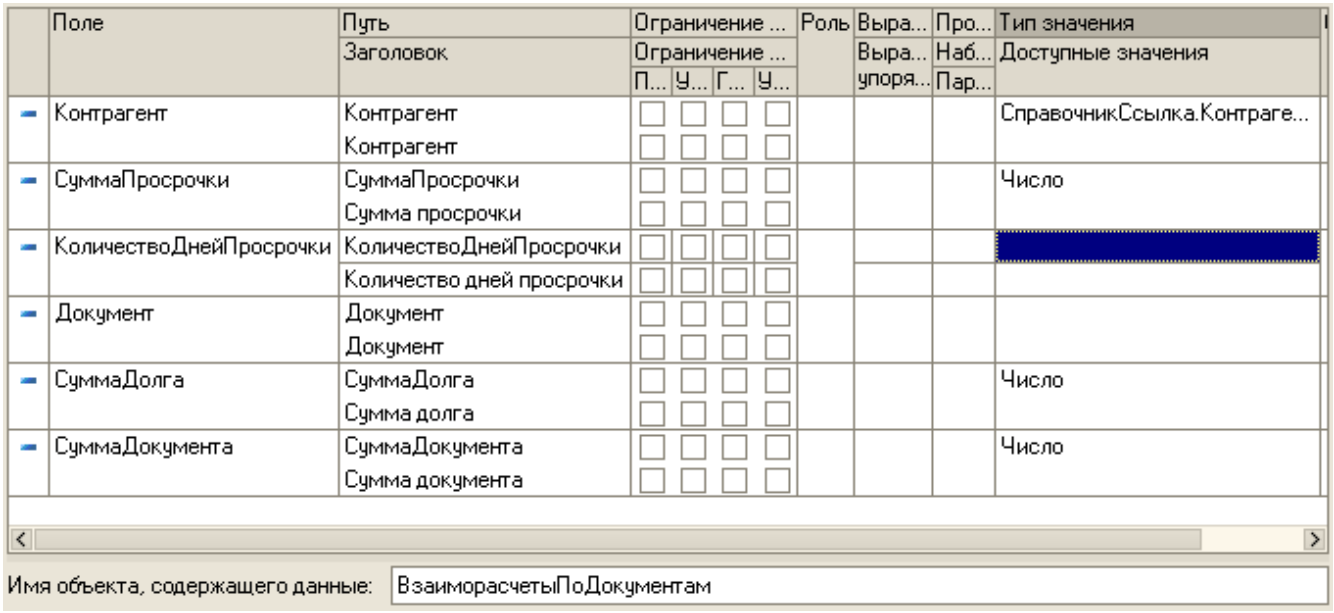


Рис. 4.2. Поля набора данных типа Объект

Добавим поля: **Контрагент** типа **СправочникСсылка.Контрагенты**, **СуммаПросрочки**, **СуммаДолга**, **СуммаДокумента** типа **Число**(14, 2), **Документ**, **КоличествоДнейПросрочки** – можно тип не указывать (рис. 4.1). Обязательно зададим имя объекта, содержащего данные – **ВзаиморасчетыПоДокументам**.

Зададим связь наборов данных по полю **Контрагент.**

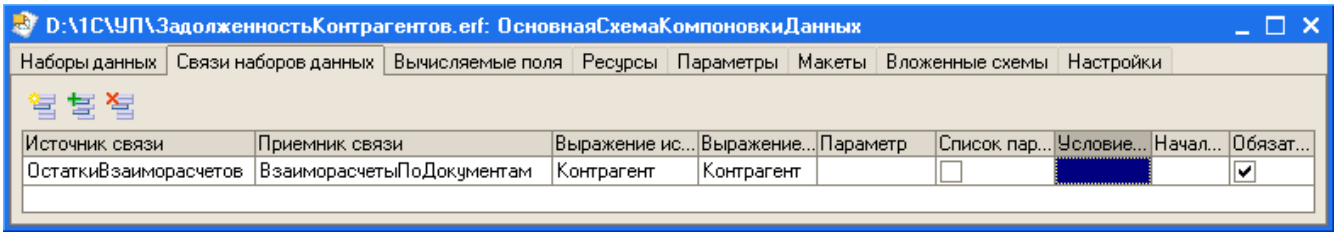


Рис. 4.3. Связь наборов данных

Все связи, созданные в системе компоновки данных, считаются левыми внешними соединениями, то есть в отчет пойдут записи из обоих наборов данных, удовлетворяющих условию **Контрагент = Контрагент**. Кроме того, у родительского набора (в нашем случае – **ОстаткиВзаиморасчетов**) будут отражаться в отчете все записи, даже в наборе **ВзаиморасчетыПоДокументам** соответствующих записей не будет.

Добавим ресурсы, как указано на рисунке (рис. 4.4). Функция поля **КоличествоДнейПросрочки – Максимум().** В колонке **Рассчитывать по..** для этого ресурса отметим поля **Контрагент и Документ.** Действительно, получение итога по количеству дней просрочки для всего отчета в целом вряд ли имеет смысл.

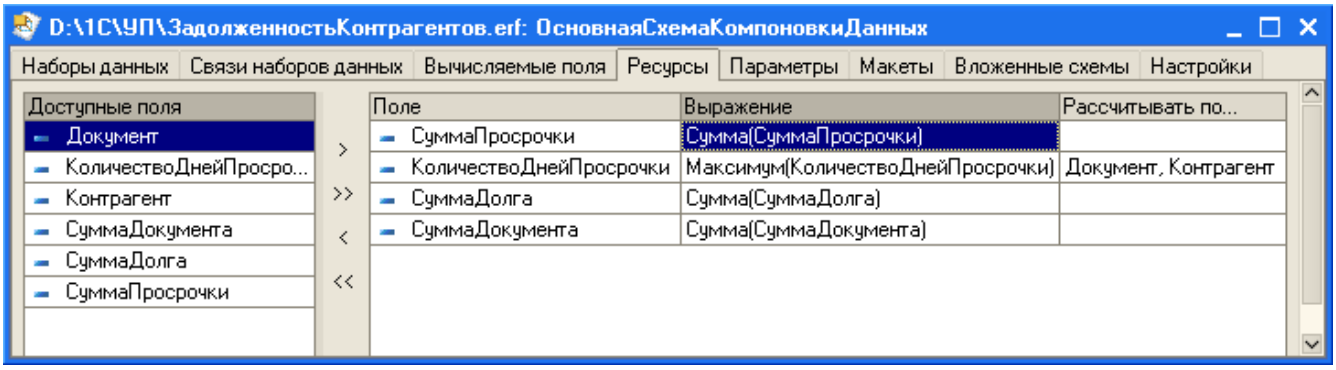


Рис. 4.4. Ресурсы отчета.

Зададим структуру отчета и состав полей, как на рисунке (рис. 4.5).

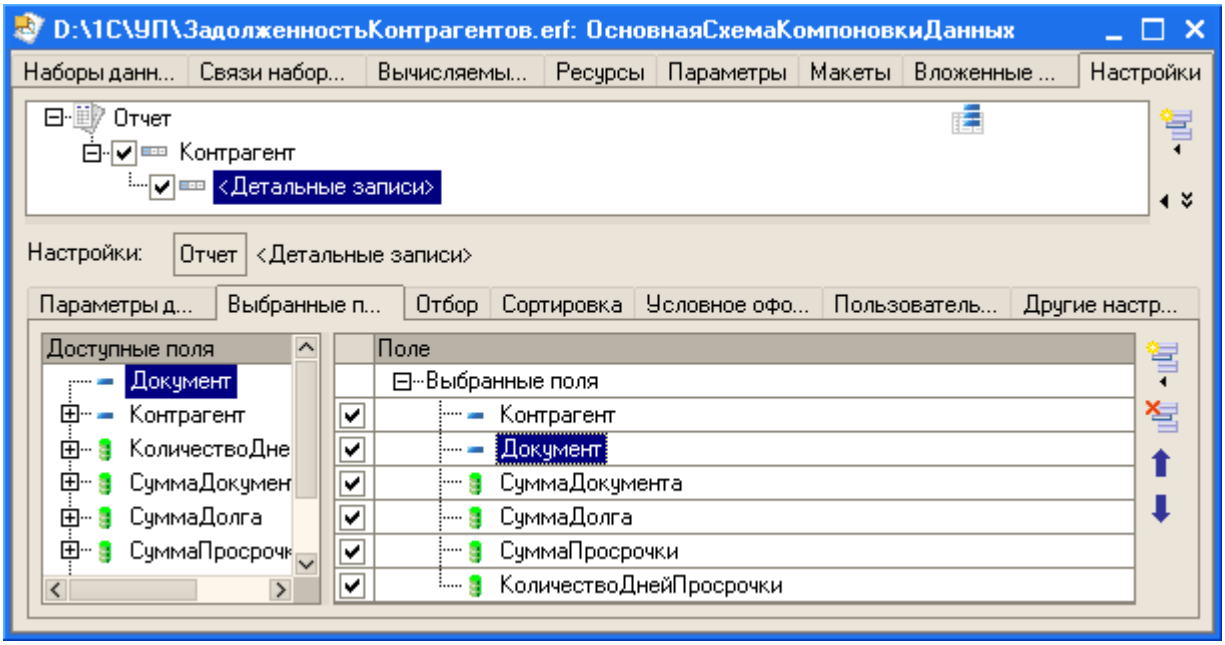


Рис. 4.5. Структура и состав полей.

Создадим форму отчета, как мы делали в предыдущем примере про отборы. На форме разместим реквизит **ДатаОтчета**. В панели действий формы удалим стандартную кнопку **Сформировать** и добавим свою, с тем же именем, и зададим ей обработчик нажатия:

```
Процедура ДействияФормыСформировать(Кнопка)

    СформироватьОтчет(ЭлементыФормы.Результат , ДанныеРасшифровки);

КонецПроцедуры
```

В модуль отчета вставим следующий код

```
Процедура СформироватьОтчет(ТабличныйДокумент , ДанныеРасшифровки) Экспорт

    // Получим таблицу задолженности по документам
    ТаблицаРезультат = Новый ТаблицаЗначений;
    ТаблицаРезультат.Колонки.Добавить("Контрагент");
    ТаблицаРезультат.Колонки.Добавить("КоличествоДнейПросрочки",
ФункцииДляОтчетов.ПолучитьОписаниеТиповЧисла(3, 0));
    ТаблицаРезультат.Колонки.Добавить("СуммаПросрочки",
ФункцииДляОтчетов.ПолучитьОписаниеТиповЧисла(14, 2));
    ТаблицаРезультат.Колонки.Добавить("СуммаДолга", ФункцииДляОтчетов.ПолучитьОписаниеТиповЧисла(14,
2));
    ТаблицаРезультат.Колонки.Добавить("СуммаДокумента",
ФункцииДляОтчетов.ПолучитьОписаниеТиповЧисла(14, 2));
    ТаблицаРезультат.Колонки.Добавить("Документ");
```

```
        Запрос = Новый Запрос;
        Запрос.УстановитьПараметр("НаДату", ДатаОтчета);
        Запрос.Текст =
        "ВЫБРАТЬ
        |       ВзаиморасчетыКонтрагентовОстатки.Контрагент,
        |       ВзаиморасчетыКонтрагентовОстатки.СуммаОстаток
        |ИЗ
        |       РегистрНакопления.ВзаиморасчетыКонтрагентов.Остатки(&НаДату, ) КАК
ВзаиморасчетыКонтрагентовОстатки
        |ГДЕ
        |       ВзаиморасчетыКонтрагентовОстатки.СуммаОстаток > 0";

        ВыборкаПоОстаткуДолга = Запрос.Выполнить().Выбрать();

        Запрос.УстановитьПараметр("ВидДвиженияПриход", ВидДвиженияНакопления.Приход);
        Запрос.Текст =
      "ВЫБРАТЬ
      | ВзаиморасчетыКонтрагентов.Регистратор,
      | СУММА(ВзаиморасчетыКонтрагентов.Сумма) КАК Сумма,
      | ВзаиморасчетыКонтрагентов.Контрагент КАК Контрагент,
      | ВзаиморасчетыКонтрагентов.Регистратор.Дата КАК ДатаДокумента,
      | РАЗНОСТЬДАТ(НАЧАЛОПЕРИОДА(ВзаиморасчетыКонтрагентов.Регистратор.Дата, ДЕНЬ), &НаДату, ДЕНЬ) КАК
СрокДолга
      |ИЗ
      | РегистрНакопления.ВзаиморасчетыКонтрагентов КАК ВзаиморасчетыКонтрагентов
      |ГДЕ
      | ВзаиморасчетыКонтрагентов.ВидДвижения = &ВидДвиженияПриход
      | И ВзаиморасчетыКонтрагентов.Период < &НаДату
      |
      |СГРУППИРОВАТЬ ПО
      | ВзаиморасчетыКонтрагентов.Регистратор,
      | ВзаиморасчетыКонтрагентов.Контрагент,
      | ВзаиморасчетыКонтрагентов.Регистратор.Дата,
      | РАЗНОСТЬДАТ(НАЧАЛОПЕРИОДА(ВзаиморасчетыКонтрагентов.Регистратор.Дата, ДЕНЬ), &НаДату, ДЕНЬ)
      |
      |УПОРЯДОЧИТЬ ПО
      | Контрагент,
      | ДатаДокумента УБЫВ";

        ВыборкаПоДокументам = Запрос.Выполнить().Выбрать();

        Пока ВыборкаПоОстаткуДолга.Следующий() Цикл

                СуммаДолгаОсталась = ВыборкаПоОстаткуДолга.СуммаОстаток;
                Контрагент = ВыборкаПоОстаткуДолга.Контрагент;

                ВыборкаПоДокументам.Сбросить();
                Пока ВыборкаПоДокументам.НайтиСледующий(Контрагент, "Контрагент") Цикл

                  Если СуммаДолгаОсталась <= 0 Тогда
                        Прервать;
                      КонецЕсли;

                      НоваяСтрока = ТаблицаРезультат.Добавить();
                      НоваяСтрока.Контрагент = Контрагент;
                      НоваяСтрока.Документ = ВыборкаПоДокументам.Регистратор;
                      НоваяСтрока.СуммаДокумента = ВыборкаПоДокументам.Сумма;
                      НоваяСтрока.СуммаДолга = Мин(СуммаДолгаОсталась, ВыборкаПоДокументам.Сумма);

                      Если ВыборкаПоДокументам.СрокДолга > Контрагент.КоличествоДнейОтсрочкиПлатежа
Тогда
                        НоваяСтрока.СуммаПросрочки= Мин(СуммаДолгаОсталась, ВыборкаПоДокументам.Сумма);
                            НоваяСтрока.КоличествоДнейПросрочки = ВыборкаПоДокументам.СрокДолга -
Контрагент.КоличествоДнейОтсрочкиПлатежа;
                      КонецЕсли;

                      СуммаДолгаОсталась = СуммаДолгаОсталась - ВыборкаПоДокументам.Сумма;

                КонецЦикла;

        КонецЦикла;

        // Сформируем структуру внешних данных
        ВнешниеНаборыДанных = Новый Структура;
      ВнешниеНаборыДанных.Вставить("ВзаиморасчетыПоДокументам", ТаблицаРезультат);

        // Установим параметр компоновки данных
        Параметр = КомпоновщикНастроек.Настройки.ПараметрыДанных.НайтиЗначениеПараметра(Новый
ПараметрКомпоновкиДанных("Период"));;
```

```
        Параметр.Значение = ДатаОтчета;
        Параметр.Использование = Истина;

        Настройки =    КомпоновщикНастроек.Настройки;
        ДанныеРасшифровки = Новый ДанныеРасшифровкиКомпоновкиДанных;
        КомпоновщикМакета = Новый КомпоновщикМакетаКомпоновкиДанных;
        МакетКомпоновки = КомпоновщикМакета.Выполнить(СхемаКомпоновкиДанных, Настройки,
ДанныеРасшифровки);

        // Инициализируем процессор СКД
        ПроцессорКомпоновкиДанных = Новый ПроцессорКомпоновкиДанных;
      ПроцессорКомпоновкиДанных.Инициализировать(МакетКомпоновки, ВнешниеНаборыДанных,
ДанныеРасшифровки);

        ТабличныйДокумент.Очистить();

        // Инициализируем процессор вывода
        ПроцессорВывода = Новый ПроцессорВыводаРезультатаКомпоновкиДанныхВТабличныйДокумент;

        ПроцессорВывода.УстановитьДокумент(ТабличныйДокумент);

        ПроцессорВывода.Вывести(ПроцессорКомпоновкиДанных);

    КонецПроцедуры
```

Разберем его. Вначале мы создаем таблицу **ТаблицаРезультат** с составом полей идентичным составу полей набора **ВзаиморасчетыПоДокументам.** Как вы уже догадались, эта таблица и будет служить источником данных для набора. Потом идет алгоритм перебора цепочек документов с целью определить сумму просроченной задолженности, а так же состав документов, сформировавших задолженность. Не будем углубляться в алгоритм, скажу лишь, что он далек от оптимального и служит для демонстрации работы отчета.

Далее мы формируем структуру для передачи таблицы значений в набор данных:

```
    ВнешниеНаборыДанных = Новый Структура;
    ВнешниеНаборыДанных.Вставить("ВзаиморасчетыПоДокументам", ТаблицаРезультат);
```

Далее инициализируем параметр компоновки данных:

```
    Параметр = КомпоновщикНастроек.Настройки.ПараметрыДанных.НайтиЗначениеПараметра(Новый
ПараметрКомпоновкиДанных("Период"));;
        Параметр.Значение = ДатаОтчета;
        Параметр.Использование = Истина;
```

Далее создаем макет компоновки путем выполнения компоновщика макета:

```
        КомпоновщикМакета = Новый КомпоновщикМакетаКомпоновкиДанных;
        МакетКомпоновки = КомпоновщикМакета.Выполнить(СхемаКомпоновкиДанных, Настройки,
ДанныеРасшифровки);
```

Инициализируем процессор компоновки данных:

```
      ПроцессорКомпоновкиДанных = Новый ПроцессорКомпоновкиДанных;
      ПроцессорКомпоновкиДанных.Инициализировать(МакетКомпоновки, ВнешниеНаборыДанных,
ДанныеРасшифровки);
```

Процессор вывода выводит результат компоновки данных в табличный документ:

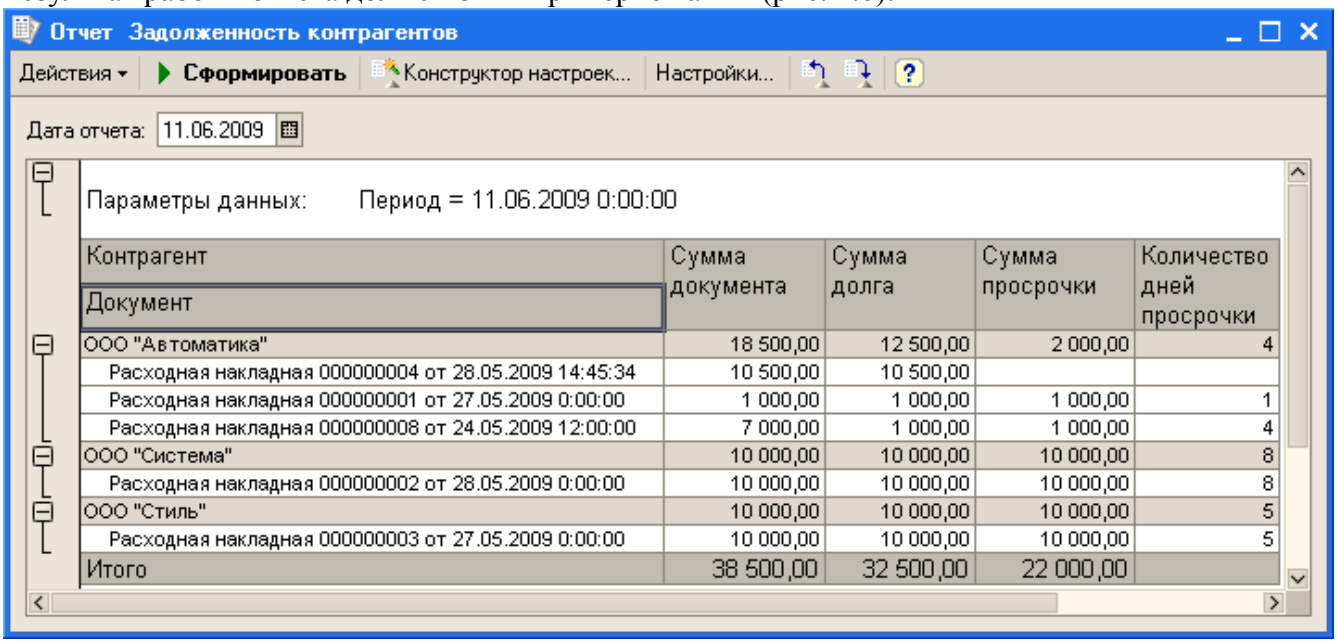
```
        ПроцессорВывода = Новый ПроцессорВыводаРезультатаКомпоновкиДанныхВТабличныйДокумент;
        ПроцессорВывода.УстановитьДокумент(ТабличныйДокумент);
        ПроцессорВывода.Вывести(ПроцессорКомпоновкиДанных);
```

Справедливости ради замечу, что данную задачу можно было выполнить другими способами:

1) Сделать один набор данных типа Объект и данные для него получить программно из таблицы значений

2) Сделать один набор данных типа Запрос и включить в него недостающий данные в виде временной таблицы.

Результат работы отчета должен быть примерно таким (рис. 4.6):



Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки
ООО "Автоматика"	18 500,00	12 500,00	2 000,00	4
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00		
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	1
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	4
ООО "Система"	10 000,00	10 000,00	10 000,00	8
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	8
ООО "Стиль"	10 000,00	10 000,00	10 000,00	5
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	5
Итого	38 500,00	32 500,00	22 000,00	

Рис. 4.6. Результат отчета «Задолженность контрагентов».

5. Шаг четвертый. Макеты компоновки данных. Создание собственных макетов.

На этом этапе попробуем сделать наш отчет «Задолженность контрагентов» более выразительным, настраивая макеты компоновки данных и используя другие возможности.

Работа с предопределенными макетами

Для примера попробуем вывести вместо стандартного заголовка более разумную надпись типа «Задолженность по контрагентам на такую-то дату», а так же сделаем более симпатичной шапку таблицы.

В случае с заголовком задача немного сложнее настройки макета. Для того, чтобы в отчете появилась какая-либо надпись, необходимо «эту надпись» как-то обозначить в структуре отчета. Если мы хотим вывести надпись со значение параметра отчета, нам нужно добавить группировку с параметром данных.

В структуре элементов отчета к корневому элементу **Отчет** добавим новую группировку без указания поля. Подвинем эту группировку вверх с использованием стрелок. В состав выбранных полей этой группировки добавим поле **ПараметрыДанных.Период** и снимем флажок у поля **Авто** (рис. 5.1).

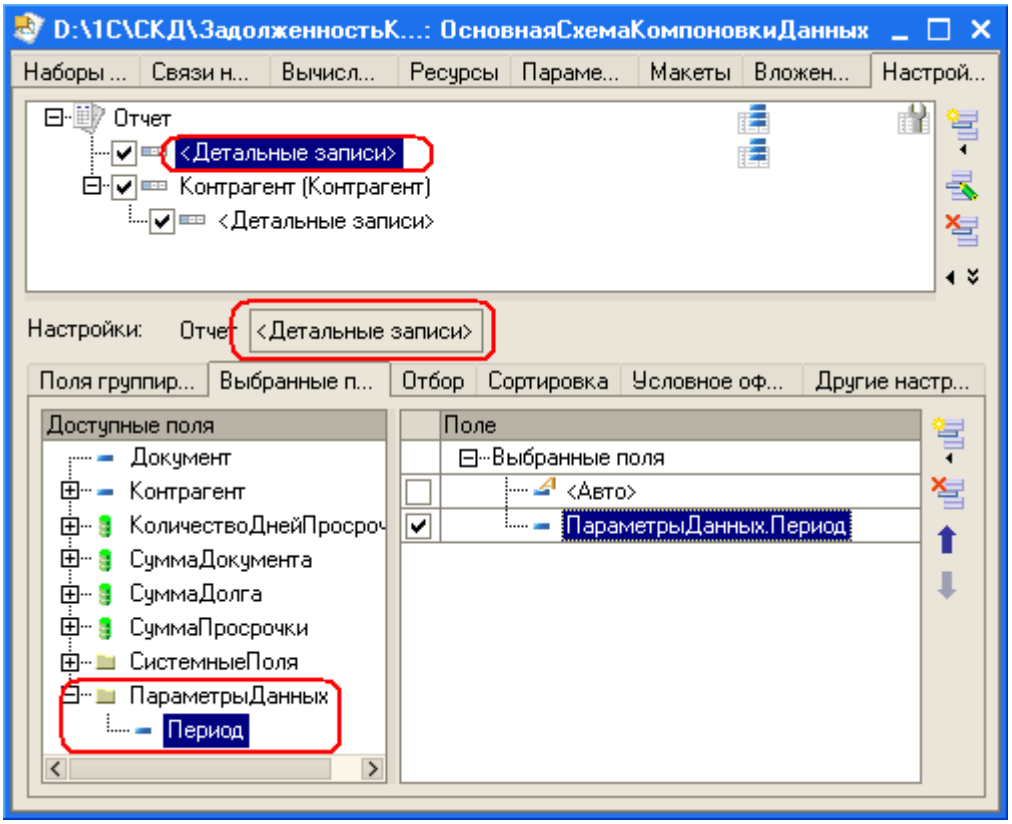


Рис. 5.1. Добавление группировки с параметром данных

Для того, чтобы в макете как-то обратиться к этой группировке, зададим ей имя «Шапка», вызвав контекстное меню.

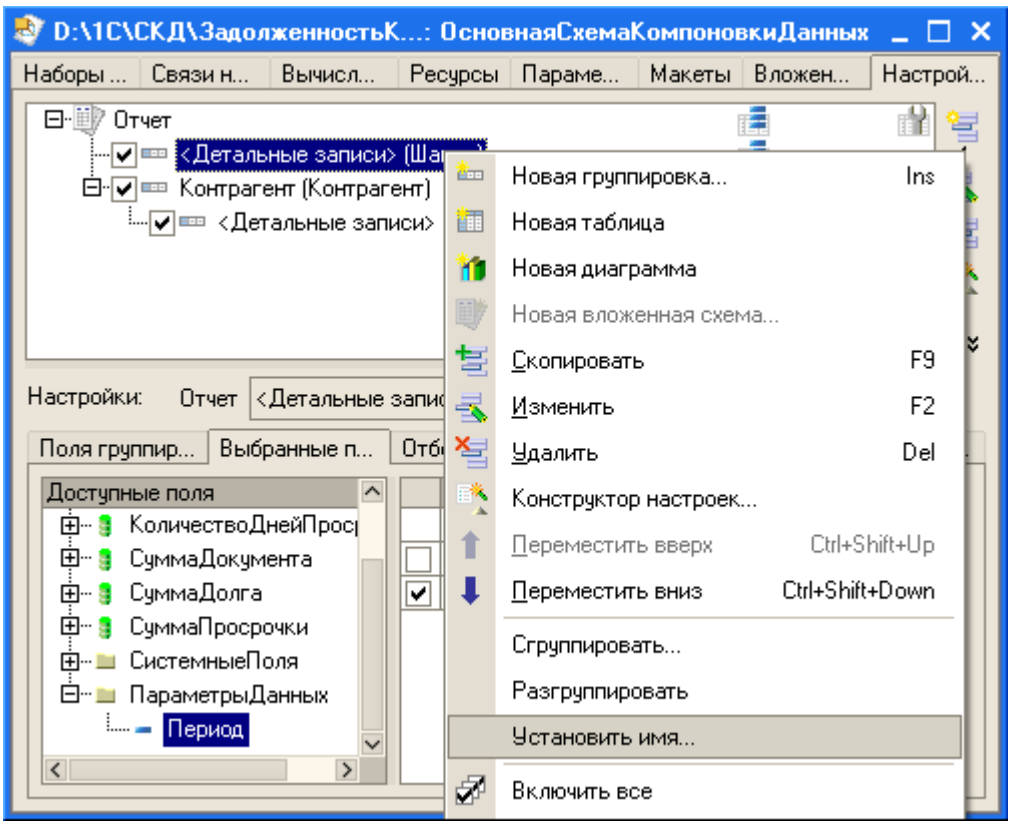


Рис. 5.2. Установка имени группировки

На закладке **Макеты** добавим новый макет группировки (рис. 5.3):

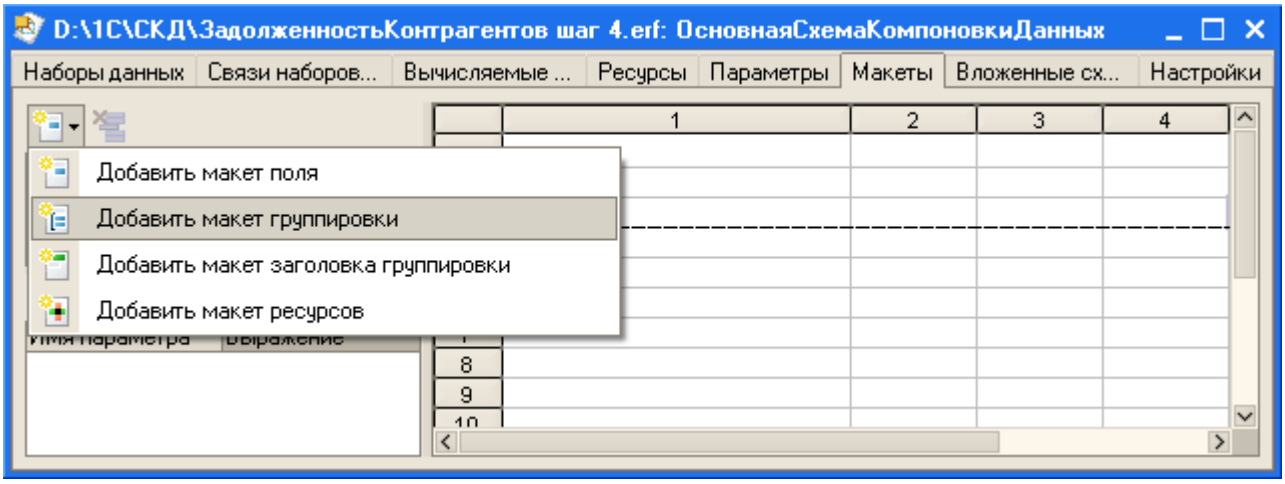


Рис. 5.3. Добавление макета группировки

В появившемся окне выберем имя группировки **Шапка** (рис. 5.4):

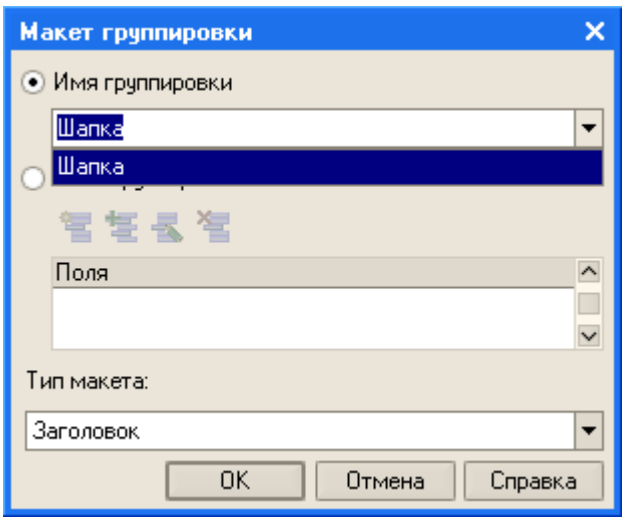


Рис. 5.4. Выбор параметров макета

В верхнюю левую ячейку введем текст с шаблоном «Просроченная задолженность контрагентов на [Период]» и зададим свойства ячейки (заполнение, свойства текста и т.д.). Заполним выражение в окне параметров макета:

Формат(ПараметрыДанных.Период, "ДФ=dd.MM.yyyy")

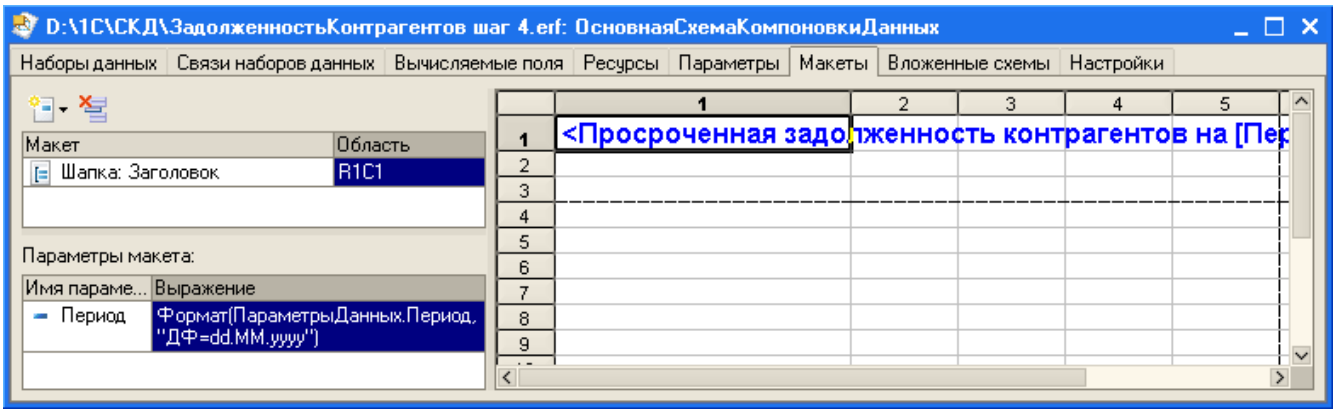


Рис. 5.4. Создание макета группировки «Шапка»

Для того, чтобы общие настройки оформления не распространялись на нашу группировку, на закладке **Другие настройки** в параметре **Макет оформления** укажем **Без оформления.** Так же для всего отчета на этой же закладке установим параметр **Выводить параметры данных** в значение **Не выводить.** В отчет вместо стандартного заголовка выведется наш (рис. 5.5).

В данном подходе, как вы могли заметить, кроется минус: если пользователь уберет из структуры отчета нашу группировку, шапка выводиться не будет. Как вывести «устойчивую» шапку, мы рассмотрим ниже.

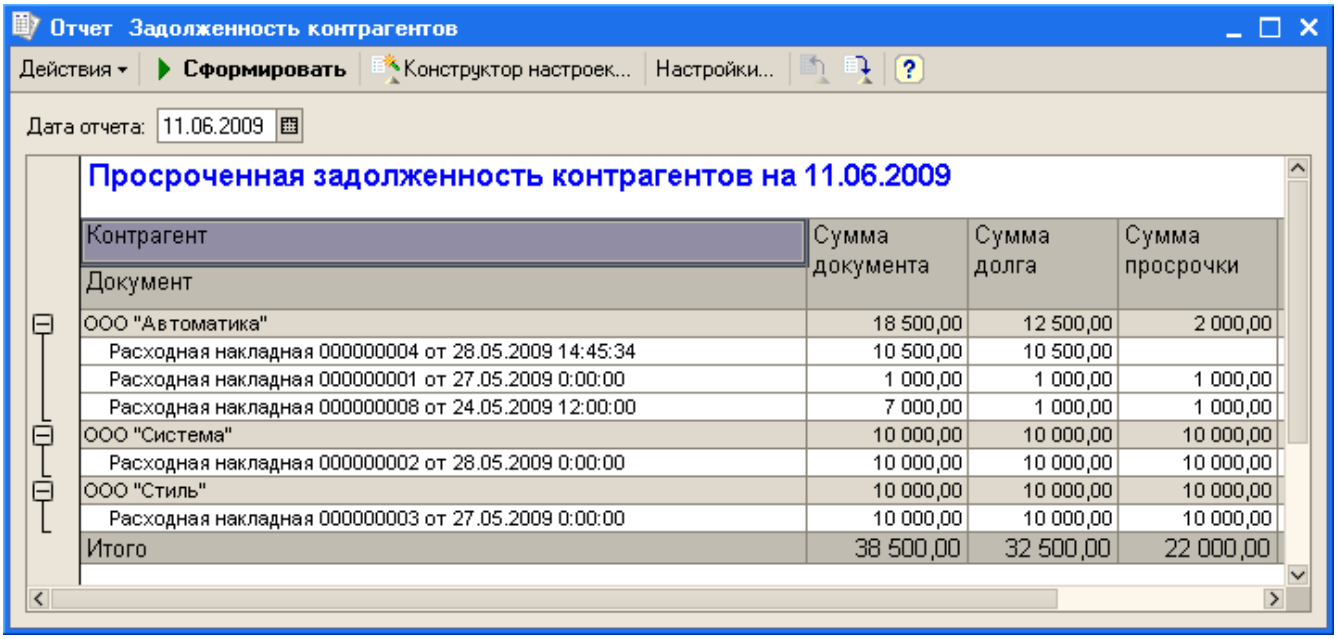


Рис. 5.5. Отчет с группировкой «Шапка»

Теперь займемся раскрашиванием нашего отчета. Сначала в окне иерархии элементов отчета зададим имена всем группировкам (рис. 5.6).

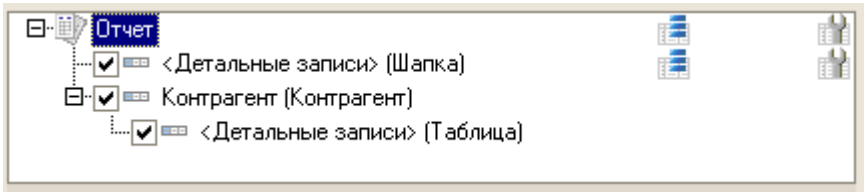


Рис. 5.6. Все группировки поименованы

Затем добавим несколько макетов: макеты вида **Макет заголовка группировки** для группировки **Контрагент**, макеты вида **Макет группировки** для группировки **Таблица и Контрагент**. Зададим элементы оформления ячеек: шрифт, стиль, цвет текста, цвет фона, текст надписей (рис. 5.7).

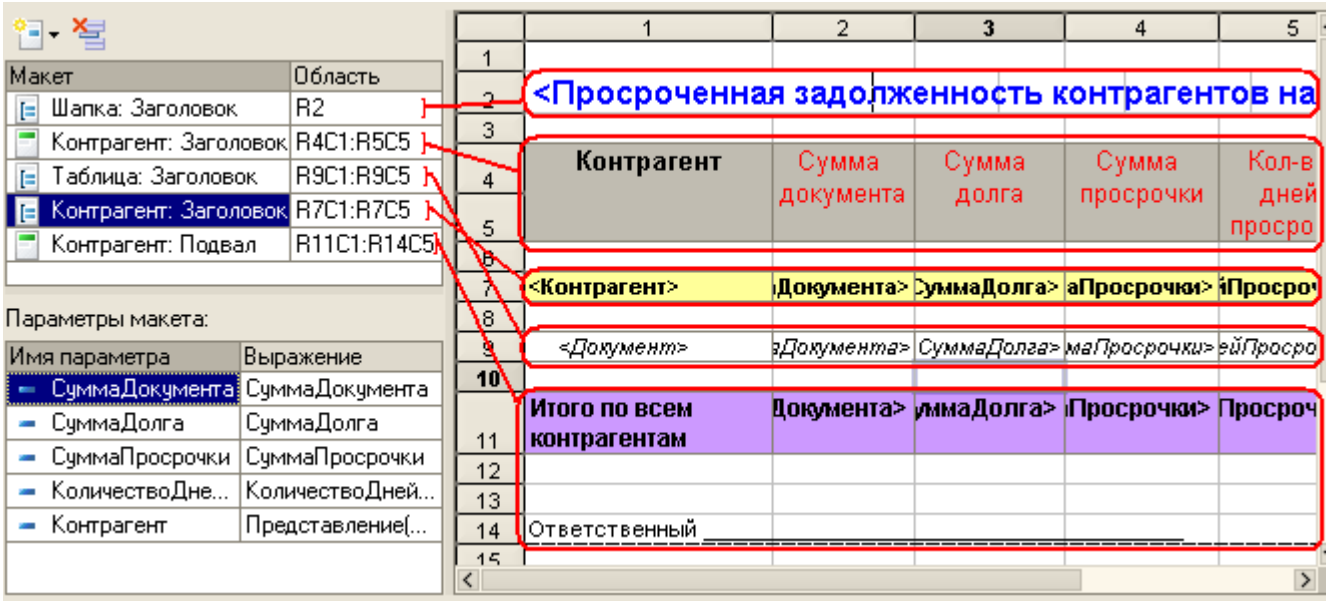


Рис. 5.7. «Раскраска» макетов отчета

Обратим внимание, что после определения границ области для макета система автоматически заполняет параметры макета и соответствующие выражения.

Для того, чтобы система не выводила еще один общий итог, на закладке **Другие настройки** для всего отчета в целом в параметре **Расположение общих итогов по вертикали** укажем **Нет.**

Отчет выполнится с нашими предопределенными макетами (рис. 5.8):

Просроченная задолженность контрагентов на 11.06.2009

Контрагент	Сумма документа	Сумма долга	Сумма просрочки	Кол-во дней просрочки
ООО "Автоматика"	**18 500,00**	**12 500,00**	**2 000,00**	**4**
Расходная накладная 000000004 от 28.05.2009 14:45:34	*10 500,00*	*10 500,00*		
Расходная накладная 000000001 от 27.05.2009 0:00:00	*1 000,00*	*1 000,00*	*1 000,00*	*1*
Расходная накладная 000000008 от 24.05.2009 12:00:00	*7 000,00*	*1 000,00*	*1 000,00*	*4*
ООО "Система"	**10 000,00**	**10 000,00**	**10 000,00**	**8**
Расходная накладная 000000002 от 28.05.2009 0:00:00	*10 000,00*	*10 000,00*	*10 000,00*	*8*
ООО "Стиль"	**10 000,00**	**10 000,00**	**10 000,00**	**5**
Расходная накладная 000000003 от 27.05.2009 0:00:00	*10 000,00*	*10 000,00*	*10 000,00*	*5*
Итого по всем контрагентам	**38 500,00**	**32 500,00**	**22 000,00**	

Ответственный _____

Рис. 5.8. Отчет с нашими макетами.

Обращу ваше внимание на то, что задав предопределенные макеты таким образом, пользователь лишается возможности изменять состав полей отчета: система выведет только те поля, которые явно заданы в наших макетах.

Работа со стандартными макетами оформления

Макеты оформления являются более универсальным средством оформления отчетов. Система компоновки данных позволяет задавать различные макеты оформления как для всего отчета в целом, так и для отдельных группировок. Для этого в режиме настроек конструктора схемы компоновки данных, на закладке **Другие настройки**, нужно перейти в режим настроек нужной группировки или отчета в целом и установить для них значение параметра **Макет оформления** (рис 5.9).

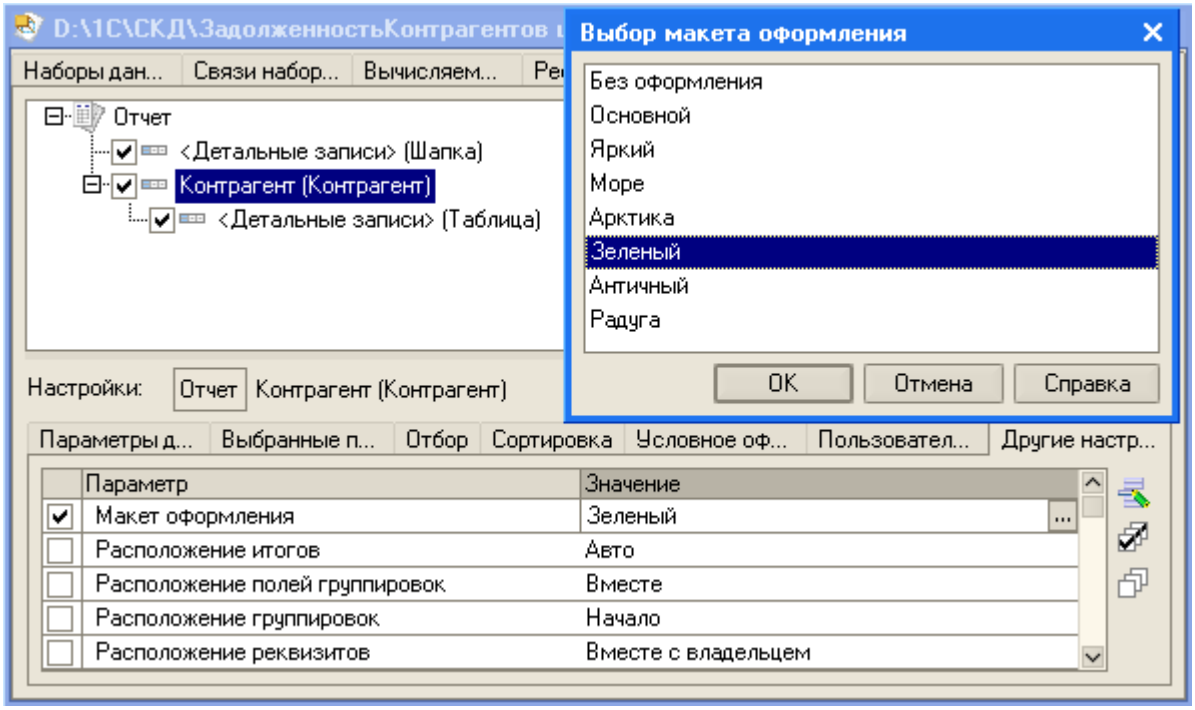


Рис. 5.9. Выбор предопределенного макета оформления

Например, если выбрать для группировки **Контрагент** макет **Зеленый**, для детальных записей – **Радуга**, а для всего отчета оставить **Основной**, отчет будет выглядеть как на рисунке (рис. 5.10):

.

Параметры данных: Период = 11.06.2009 0:00:00

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки
ООО "Автоматика"	**18 500,00**	**12 500,00**	**2 000,00**
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00	
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00
ООО "Система"	**10 000,00**	**10 000,00**	**10 000,00**
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00
ООО "Стиль"	**10 000,00**	**10 000,00**	**10 000,00**
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00
Итого	**38 500,00**	**32 500,00**	**22 000,00**

Рис. 5.10. Отчет с макетами оформления группировок

Создание собственного макета оформления

Собственный макет можно создать, используя конструктор макетов. Для этого добавим в раздел макетов нашего отчета новый макет. В открывшемся конструкторе установим тип макета **Макет оформления компоновки данных**.

Редактор макетов позволяет редактировать почти все те же параметры оформления, что и предопределенные макеты (рис.5.11)

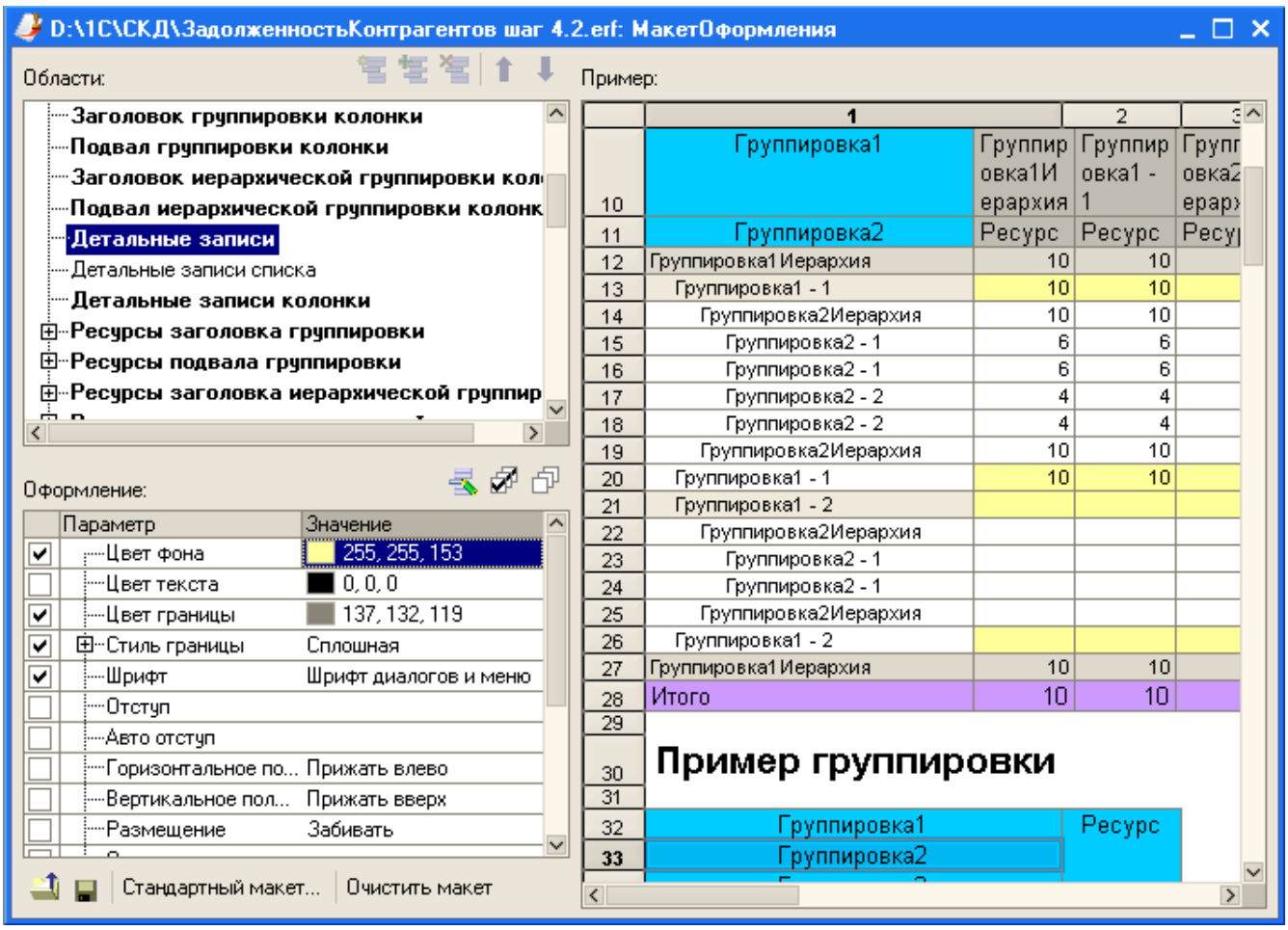


Рис. 5.11. Редактор макета оформления

Выделив в верхнем левом углу нужную область макета, можно изменить элементы ее оформления и в нижнем левом окне и сразу увидеть результат в правом окне редактора макета.

Для примера, возьмем за основу стандартный макет **Основной** с помощью кнопки **Стандартный макет..** в нижнем левом углу. Отредактируем в нем область **Заголовок таблицы**, **Детальные записи** и **Заголовок группировки – Уровень 1**, а так же область **Общие итоги по вертикали – Уровень 1.** Изменим цвет фона по вашему усмотрению.

Теперь нам нужно привязать наш макет к компоновщику макета. Для этого добавим в модуль отчета строку:

```
МакетОформления = ПолучитьМакет("МакетОформления");
```

И добавим наш макет в параметры выполнения компоновщика макета:

```
МакетКомпоновки = КомпоновщикМакета.Выполнить(СхемаКомпоновкиДанных, Настройки, ДанныеРасшифровки, МакетОформления);
```

После моих настроек отчет стал выглядеть, как попугай (рис. 5.12):

Параметры данных: Период = 11.06.2009 0:00:00

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки
ООО "Автоматика"	18 500,00	12 500,00	2 000,00	4
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00		
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	1
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	4
ООО "Система"	10 000,00	10 000,00	10 000,00	8
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	8
ООО "Стиль"	10 000,00	10 000,00	10 000,00	5
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	5
Итого	38 500,00	32 500,00	22 000,00	

Рис. 5.12. Результат отчета.

Добавление собственных секций шапки и подвала

Часто на предприятиях есть установленные внутренние формы отчетности, в которых есть определенная шапка отчета и определенный подвал, и зачастую в этих секциях присутствуют некие параметры, например, ФИО главного бухгалтера или название подразделения. Для того чтобы добавить такую информацию в отчет с использованием СКД, необходимо создать отдельный макет с этим секциями и выводить эти секции в отчет в явном виде.

Для примера, возьмем последний отчет и сделаем шапку с названием организации и периодом, а в подвале поместим ФИО финансового директора.

В разделе макетов отчета добавим новый макет, и выберем тип – Табличный документ. В макете создадим две секции – Шапка и Подвал и вставим в них необходимые данные (рис. 5.13):

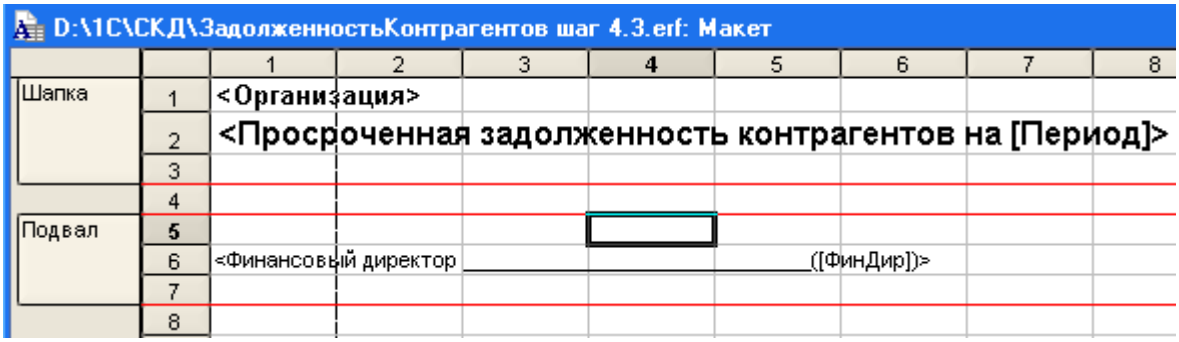


Рис. 5.13. Дополнительный макет

В модуль отчета вставим необходимый код, чтобы проинициализировать параметры областей макетов и вывести их в отчет:

```
ТабличныйДокумент.Очистить();

Макет = ПолучитьМакет("Макет");
ОбластьШапка = Макет.ПолучитьОбласть("Шапка");
ОбластьШапка.Параметры.Организация = Константы.Организация.Получить();
ОбластьШапка.Параметры.Период = Формат(ДатаОтчета, "ДФ=dd.ММ.уууу");

ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");
ОбластьПодвал.Параметры.ФинДир = Константы.ФИОФинДиректора.Получить();

ТабличныйДокумент.Вывести(ОбластьШапка);

// Инициализируем процессор вывода
ПроцессорВывода = Новый ПроцессорВыводаРезультатаКомпоновкиДанныхВТабличныйДокумент;
```

```
ПроцессорВывода.УстановитьДокумент(ТабличныйДокумент);

ПроцессорВывода.Вывести(ПроцессорКомпоновкиДанных);

ТабличныйДокумент.Вывести(ОбластьПодвал);
```

Думаю, данный код в комментариях не нуждается.

ООО "Гуси-лебеди"

Просроченная задолженность контрагентов на 11.06.2009

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	К д п
ООО "Автоматика"	18 500,00	12 500,00	2 000,00	
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00		
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	
ООО "Система"	10 000,00	10 000,00	10 000,00	
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	
ООО "Стиль"	10 000,00	10 000,00	10 000,00	
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	
Итого	38 500,00	32 500,00	22 000,00	

Финансовый директор _____(Владимир Владимирович)

Рис. 5.14. Результат отчета

Фиксация шапки отчета

Для «длинных» и «широких» отчетов часто для удобства пользователя нужно зафиксировать шапку, для того, чтобы пользователь не забыл смысл колонок при прокрутке результата отчета.

В отчетах с использованием СКД невозможно заранее рассчитать высоту шапки, так как она зависит от многих настроек, например, от структуры, отборов, параметров и пр.

Для фиксации шапки в таких отчетах нужно воспользоваться возможностью процессора вывода выводить отчет поэлементно. Вместо строки

```
ПроцессорВывода.Вывести(ПроцессорКомпоновкиДанных);
```

в модуле отчета вставим следующий код:

```
//Обозначим начало вывода
ПроцессорВывода.НачатьВывод();

ТаблицаЗафиксирована = Ложь;

ТабличныйДокумент.ФиксацияСверху = 0;
//Основной цикл вывода отчета
Пока Истина Цикл

        ОбработкаПрерыванияПользователя();
        //Получим следующий элемент результата компоновки
        ЭлементРезультата = ПроцессорКомпоновкиДанных.Следующий();

        Если ЭлементРезультата = Неопределено Тогда
                //Следующий элемент не получен - заканчиваем цикл вывода
                Прервать;

        Иначе

                // Зафиксируем шапку
                Если Не ТаблицаЗафиксирована
```

```
                        И ЭлементРезультата.ЗначенияПараметров.Количество() > 0
                        И ТипЗнч(КомпоновщикНастроек.Настройки.Структура[0]) <>
Тип("ДиаграммаКомпоновкиДанных") Тогда
                        ТаблицаЗафиксирована = Истина;
                        ТабличныйДокумент.ФиксацияСверху = ТабличныйДокумент.ВысотаТаблицы;
                КонецЕсли;

                //Элемент получен - выведем его при помощи процессора вывода
                ПроцессорВывода.ВывестиЭлемент(ЭлементРезультата);

        КонецЕсли;

КонецЦикла;

//Обозначем завершение вывода
ПроцессорВывода.ЗакончитьВывод();
```

Разберем этот код. Метод **Следующий()** объекта **ПроцессорКомпоновкиДанных** возвращает элемент результата компоновки данных, если есть очередной элемент, и **Неопределенно**, если элементы закончились. Условный оператор

```
Если Не ТаблицаЗафиксирована И ЭлементРезультата.ЗначенияПараметров.Количество() > 0
        И ТипЗнч(КомпоновщикНастроек.Настройки.Структура[0]) <> Тип("ДиаграммаКомпоновкиДанных") Тогда
```

проверяет, закончен ли вывод шапки отчета, анализируя свойство **ЗначенияПараметров.** Оно представляет собой коллекцию значений параметров макета компоновки данных, и как только в ней появляются элементы, это означает, что процессор компоновки начал выдавать данные отчета.

Отчет с фиксированной шапкой должен выглядеть как на рисунке (рис 5.15):

ООО "Гуси-лебеди"

Просроченная задолженность контрагентов на 11.06.2009

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки
ООО "Автоматика"	18 500,00	13 500,00	3 000,00	4
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00		
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	1
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	2 000,00	2 000,00	4
ООО "Система"	10 000,00	10 000,00	10 000,00	8
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	8
ООО "Стиль"	10 000,00	10 000,00	10 000,00	5
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	5
Итого	38 500,00	33 500,00	23 000,00	

Финансовый директор _____(Владимир Владимирович)

Рис. 5.15. Результат с фиксированной шапкой

6. Шаг пятый. Использование характеристик. Условное оформление. Вычисляемые поля.

На этом шаге мы продолжим разбирать возможности системы компоновки данных. Мы рассмотрим механизм характеристик, условное оформление отдельных частей отчета и создание пользовательских полей.

Использование характеристик объектов

В платформе 1С:Предприятие 8 есть специальный объект – *План видов характеристик*, с помощью которого можно настроить механизм, позволяющий пользователю присваивать объектам произвольные свойства.

Система компоновки данных позволяет использовать эти характеристики при разработке отчетов. Для визуального описания характеристик в СКД используется закладка **Характеристики** конструктора запросов.

В учебной конфигурации реализована возможность добавления дополнительных свойств к справочнику Контрагенты. В базе данных уже заведены такие характеристики как *Регион*, *Надежный партнер* и *Лучший покупатель*. Отразим их в отчете **ЗадолженностьКонтрагентов**.

Откроем конструктор схемы компоновки данных, вызовем конструктор запросов и перейдем на закладку **Характеристики**. Нажмем кнопку Добавить и заполним поля, как указано на рисунке (рис. 6.1). В поле **Тип значения** введем тип данных, для которого предназначены характеристики – **СправочникСсылка.Контрагенты**. Затем опишем источник списка характеристик. В нашем случае это план видов характеристик **СвойстваОбъектов**. Поэтому в поле **Источник** выберем **Таблица**, а в поле **Список характеристик** укажем **ПланВидовХарактеристик.СвойстваОбъектов**. Далее опишем назначение полей источника списка характеристик: в поле **Идентификатор** выберем **Ссылка**, в поле **Имя** – **Наименование**, а в поле **Тип** – **ТипЗначения**. Теперь опишем источник значений характеристик. В нашей конфигурации значения характеристик хранятся в регистре сведений **ЗначенияСвойствОбъектов**, поэтому в поле **Источник** укажем **Таблица**, а в поле **Значение характеристик** – **РегистрСведений.ЗначенияСвойствОбъектов.** Далее опишем назначение полей регистра. В поле **Объект** выберем измерение регистра **Контрагент**, в поле **Идентификатор** – измерение регистра **Свойство**, а в поле **Значение** - ресурс регистра **Значение**.

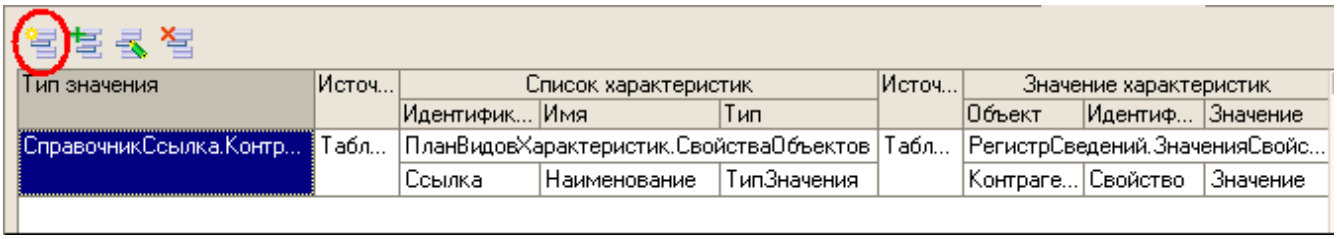


Рис. 6.1. Описание полей характеристик

Сохраним запрос. В результате текст запроса примет вид:

```
ВЫБРАТЬ
    ВзаиморасчетыКонтрагентов.Контрагент
ИЗ
    РегистрНакопления.ВзаиморасчетыКонтрагентов.Остатки КАК ВзаиморасчетыКонтрагентов
```

```
ГДЕ
    ВзаиморсчетыКонтрагентов.СуммаОстаток > 0
{ХАРАКТЕРИСТИКИ
    ТИП(Справочник.Контрагенты)
    СПИСОК ПланВидовХарактеристик.СвойстваОбъектов
    ИДЕНТИФИКАТОР Ссылка
    ИМЯ Наименование
    ТИПЗНАЧЕНИЯ ТипЗначения
    ЗНАЧЕНИЯ РегистрСведений.ЗначенияСвойствОбъектов
    ОБЪЕКТ Контрагент
    ХАРАКТЕРИСТИКА Свойство
    ЗНАЧЕНИЕ Значение }
```

Однако использовать свойства контрагентов для настройки отчета в режиме конфигурирования не удастся, так как список свойств хранится в информационной базе.

Откроем отчет в режиме 1С:Предприятие. Для использования характеристик в отчете вызовем окно настроек. В иерархию элементов в самый верхний уровень добавим группировку по полю **Контрагент.Регион** (рис. 6.2).

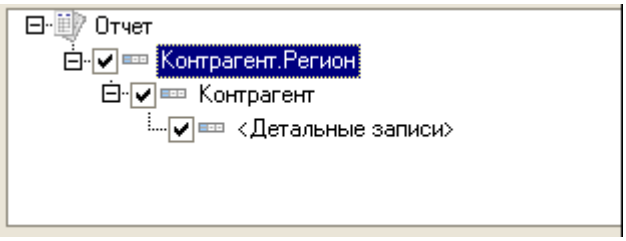


Рис. 6.2. Иерархия элементов отчета

В результате получим отчет, в котором контрагенты сгруппированы по свойству **Регион** (рис. 6.3):

ООО "Гуси-лебеди"

Просроченная задолженность контрагентов на 11.06.2009

Контрагент.Регион Контрагент Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки
Королев	18 500,00	12 500,00	2 000,00	
ООО "Автоматика"	18 500,00	12 500,00	2 000,00	4
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00		
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	1
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	4
Москва	20 000,00	20 000,00	20 000,00	
ООО "Система"	10 000,00	10 000,00	10 000,00	8
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	8
ООО "Стиль"	10 000,00	10 000,00	10 000,00	5
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	5
Итого	38 500,00	32 500,00	22 000,00	

Финансовый директор _____(Владимир Владимирович)

Рис. 6.3. Результат отчета

Точно так же мы можем использовать свойства контрагентов в других возможностях настроек: полях группировок, отборах, сортировке и т.д.

Условное оформление

Для более наглядного представления данных в отчете система компоновки данных позволяет использовать условной оформление отдельных областей отчета. При помощи условного

оформления можно задавать цвет текста или фона, шрифт, положение и прочие параметры как отчета в целом, так и отдельных его частей.

Для примера в отчете **ЗадолженностьКонтрагентов** выделим красным фоном контрагентов, у которых количество дней просрочки долга превышает 5.

В режиме конфигурирования откроем схему компоновки данных, на закладке **Настройки** выделим группировку **Контрагент** и откроем нижнюю закладку **Условное оформление**. Добавим новый элемент условного оформления и заполним его параметры.

Область – область отчета, к которой применяется условное оформление, это может быть поле или набор полей, а если поля не указаны – оформление будет применяться ко всему отчету, группировке или таблице, в зависимости от того, с каким элементом отчета мы работаем. Для нашего примера оставим это поле пустым.

Отбор – условие, при выполнении которого применяется условное оформление. В нашем примере установим отбор, как на рисунке (рис. 6.4):

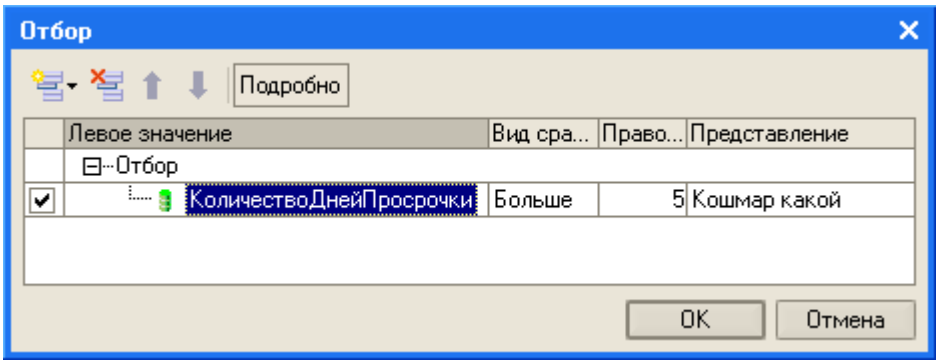


Рис. 6.4. Установка отбора

Оформления – описание применяемого оформления. В нашем примере установим цвет фона в красный.

Представление – необязательный элемент для представления условного оформления пользователю.

Описание условного оформления будет таким (рис. 6.5):

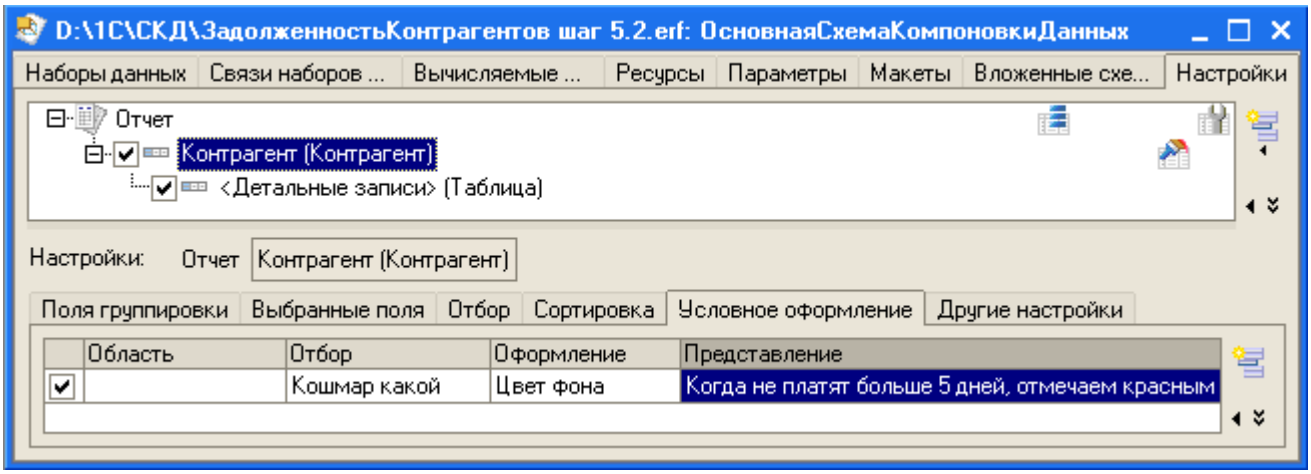


Рис. 6.5. Настройка условного оформления

А пользователь при выполнении отчета получит такую картину (рис. 6.6):

ООО "Гуси-лебеди"

Просроченная задолженность контрагентов на 11.06.2009

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки
ООО "Автоматика"	18 500,00	12 500,00	2 000,00	4
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00		
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	1
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	4
ООО "Система"	10 000,00	10 000,00	10 000,00	8
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	8
ООО "Стиль"	10 000,00	10 000,00	10 000,00	5
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	5
Итого	38 500,00	32 500,00	22 000,00	

Финансовый директор _____(Владимир Владимирович)

Рис. 6.6. Результат отчета

Вычисляемые поля

При разработке отчетов не всегда хватает полей, определенных в наборе данных. Иногда требуется отразить какие-то другие поля, которые рассчитываются на основе существующих по определенному алгоритму.

Для того, чтобы система рассчитала значение вычисляемого поля, необходимо указать выражение этого поля на языке выражений системы компоновки данных. В этом языке есть ограничение: в нем нельзя использовать вычисляемые поля. В остальном ограничений нет – можно использовать все синтаксические конструкции языка выражений, поля наборов данных, а так же функции общих модулей конфигурации.

Для примера добавим в наш отчет **ЗадолженностьКонтрагентов** поле **Пеня,** которое мы будем рассчитывать так – при просрочке менее пяти дней пеня не взимается, в другом случае пеня равняется одном проценту от суммы просроченного долга за каждый день просрочки.

Откроем схему компоновки данных и на закладке **Вычисляемые поля** добавим поле. Дадим ему имя (Путь к данным) – **Пеня**, в колонку выражение введем такой код:

```
Выбор
        Когда КоличествоДнейПросрочки >=  5
               Тогда СуммаПросрочки * 0.01 * КоличествоДнейПросрочки
        Иначе 0
Конец
```

Зададим так же тип значения – **Число** и формат – «ЧЦ=14; ЧДЦ=2»

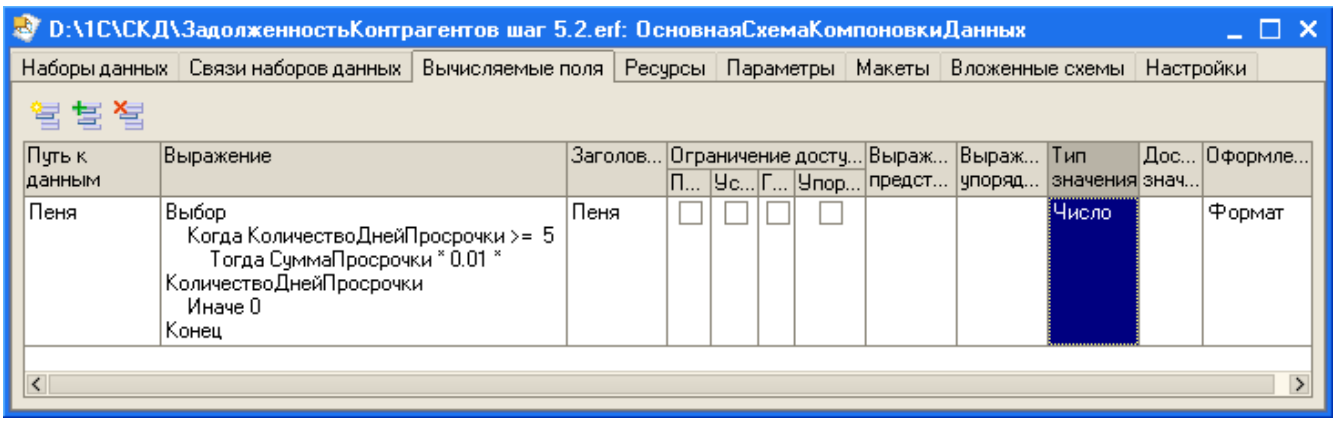


Рис. 6.7. Добавление вычисляемого поля

Добавим поле Пеня в список ресурсов с функцией **Сумма()**, чтобы вычислять итоги по контрагентам и в целом.

В результате выполнения отчета в поле **Пеня** по контрагенту будет отражаться сумма пеней за каждый документ, по которому выполняется условие начисления пеней (рис 6.8).

ООО "Гуси-лебеди"

Просроченная задолженность контрагентов на 15.06.2009

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки	Пеня
ООО "Автоматика"	18 500,00	12 500,00	12 500,00	8	130,00
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00	10 500,00	4	
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	5	50,00
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	8	80,00
ООО "Система"	10 000,00	10 000,00	10 000,00	12	1 200,00
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	12	1 200,00
ООО "Стиль"	10 000,00	10 000,00	10 000,00	9	900,00
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	9	900,00
Итого	38 500,00	32 500,00	32 500,00		2 230,00

Финансовый директор _____(Владимир Владимирович)

Рис. 6.8. Отчет, который суммирует пени по документам

Однако на практике чаще всего пеня рассчитывается по-другому. Срок и сумма просрочки берутся в целом по контрагенту, а не по каждому документу в отдельности.

Для того чтобы отчет выводил нужные данные по контрагенту, необходимо усложнить выражение агрегирования для ресурса **Пеня**:

```
Сумма(СуммаПросрочки) * 0.01 * Максимум(КоличествоДнейПросрочки)
```

Если мы введем такое выражение в колонку **Выражение** для ресурса **Пеня**, получим правильный расчет по контрагентам (рис. 6.9):

ООО "Гуси-лебеди"

Просроченная задолженность контрагентов на 15.06.2009

Контрагент / Документ	Сумма документа	Сумма долга	Сумма просрочки	Количество дней просрочки	Пеня
ООО "Автоматика"	18 500,00	12 500,00	12 500,00	8	1 000,00
Расходная накладная 000000004 от 28.05.2009 14:45:34	10 500,00	10 500,00	10 500,00	4	
Расходная накладная 000000001 от 27.05.2009 0:00:00	1 000,00	1 000,00	1 000,00	5	50,00
Расходная накладная 000000008 от 24.05.2009 12:00:00	7 000,00	1 000,00	1 000,00	8	80,00
ООО "Система"	10 000,00	10 000,00	10 000,00	12	1 200,00
Расходная накладная 000000002 от 28.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	12	1 200,00
ООО "Стиль"	10 000,00	10 000,00	10 000,00	9	900,00
Расходная накладная 000000003 от 27.05.2009 0:00:00	10 000,00	10 000,00	10 000,00	9	900,00
Итого	38 500,00	32 500,00	32 500,00		3 900,00

Финансовый директор _____(Владимир Владимирович)

Рис. 6.9. Отчет, который считает пени по контрагентам

7. Шаг шестой. Расшифровка.

На этом шаге мы познакомимся с механизмом расшифровки в отчетах с применением системы компоновки данных.

Система компоновки данных предоставляет возможность детализировать информацию в отчете. То есть, сформировав отчет и щелкнув по одной из ячеек, пользователь может получить другой отчет, связанный со значением, находящимся в этой ячейке.

Возможности стандартной расшифровки

Для отчетов, которые формируются стандартной командой **Сформировать**, данные расшифровки помещаются в свойство расширения формы отчета **ДанныеРасшифровки**. Для таких отчетов система уже предоставляет определенные возможности расшифровки.

Для примера возьмем отчет **ОстаткиНоменклатуры**. Изменим текст запроса так, чтобы добавилось поле **Регистратор**.

```
ВЫБРАТЬ
    ИсточникДанных.Номенклатура,
    ИсточникДанных.Склад,
    ИсточникДанных.Регистратор,
    ИсточникДанных.КоличествоНачальныйОстаток,
    ИсточникДанных.КоличествоПриход,
    ИсточникДанных.КоличествоРасход,
    ИсточникДанных.КоличествоКонечныйОстаток,
    ИсточникДанных.СуммаНачальныйОстаток,
    ИсточникДанных.СуммаПриход,
    ИсточникДанных.СуммаРасход,
    ИсточникДанных.СуммаКонечныйОстаток
ИЗ
    РегистрНакопления.ОстаткиНоменклатуры.ОстаткиИОбороты(, , Авто, , ) КАК ИсточникДанных
```

Здесь слово **Авто** в списке параметров виртуальной таблицы означает, что система будет корректно рассчитывать остатки при использовании дополнительных группировок по периодам.

Система автоматически добавит новое поле в список полей. Сформируем отчет в режиме 1С:Предприятие. Дважды щелкнем на поле группировки. Откроется контекстное меню расшифровки (рис 7.1):

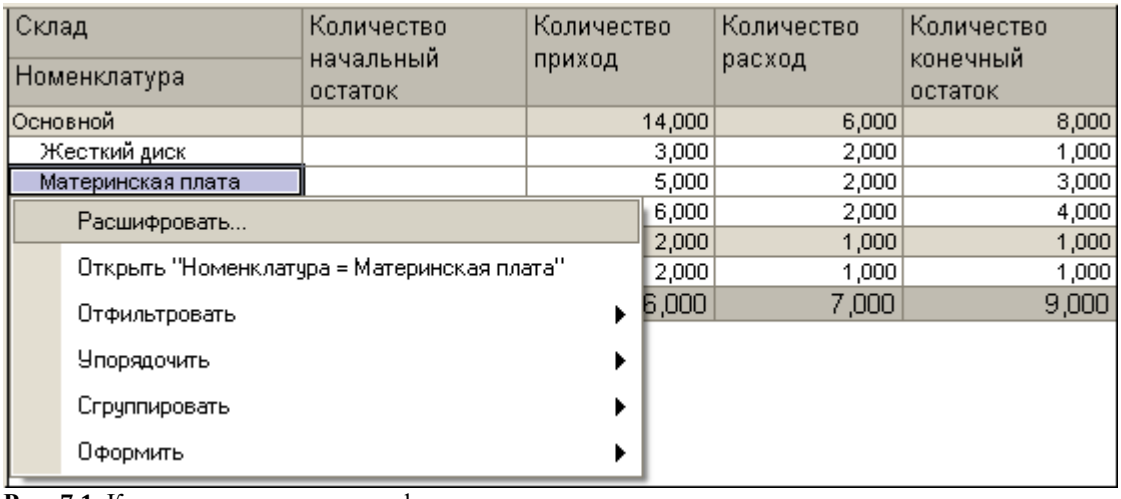
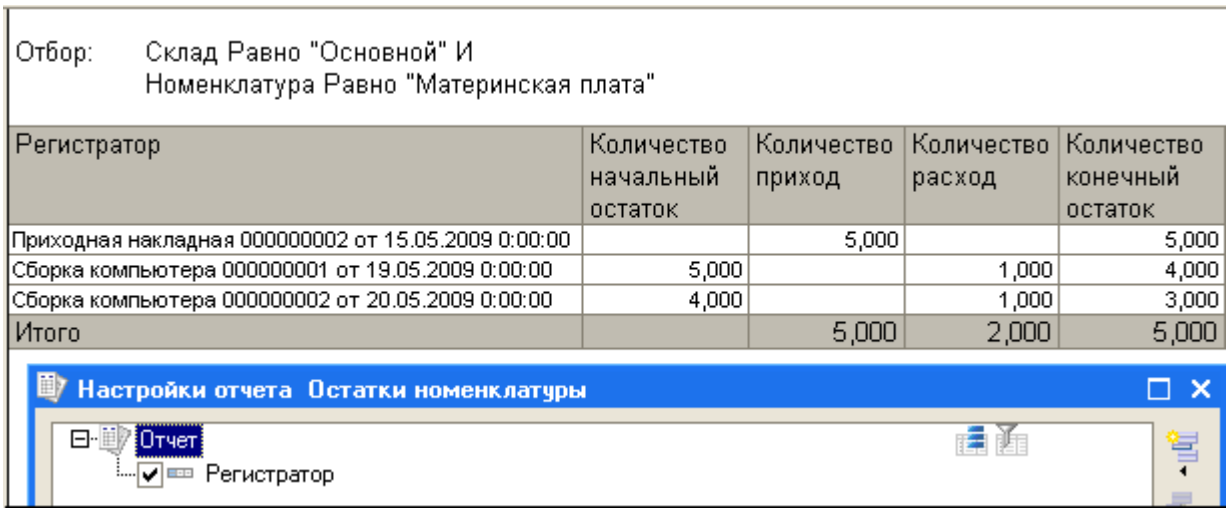


Рис. 7.1. Контекстное меню расшифровки

Система предлагает сформировать новый отчет с предварительным заданием некоторых настроек.

Расшифровать.. – система предложит на выбор неиспользованные, но возможные для выбора поля, и после выбора сформирует отчет с группировкой по этому полю и отбором по тем значениям группировок, которые находятся в выбранной вами ячейке. Например, в нашем случае можно сделать расшифровку по полю **Регистратор** (рис. 7.2):



Отбор: Склад Равно "Основной" И
Номенклатура Равно "Материнская плата"

Регистратор	Количество начальный остаток	Количество приход	Количество расход	Количество конечный остаток
Приходная накладная 000000002 от 15.05.2009 0:00:00		5,000		5,000
Сборка компьютера 000000001 от 19.05.2009 0:00:00	5,000		1,000	4,000
Сборка компьютера 000000002 от 20.05.2009 0:00:00	4,000		1,000	3,000
Итого		5,000	2,000	5,000

Рис. 7.2. Расшифровка по регистратору

Открыть.. – система откроет значение ближайшей к выбранной ячейке группировки.

Отфильтровать – система создаст отчет с отбором по значению в выбранной ячейке.

Упорядочить - система переформирует отчет с упорядочиванием по значению в выбранной ячейке.

Сгруппировать - система переформирует отчет с группировкой по полю в выбранной ячейке.

Оформить - система переформирует отчет с условным оформлением по полю в выбранной ячейке.

Но если в отчете процесс компоновки данных осуществляется программно, нужно явно указывать объект **ДанныеРасшифровки** при выполнении компоновщика макета и при инициализации процессора компоновки данных:

```
ДанныеРасшифровки = Новый ДанныеРасшифровкиКомпоновкиДанных;
КомпоновщикМакета = Новый КомпоновщикМакетаКомпоновкиДанных;
МакетКомпоновки = КомпоновщикМакета.Выполнить(СхемаКомпоновкиДанных, Настройки, ДанныеРасшифровки);

ПроцессорКомпоновкиДанных = Новый ПроцессорКомпоновкиДанных;
ПроцессорКомпоновкиДанных.Инициализировать(МакетКомпоновки,ВнешниеНаборыДанных, ДанныеРасшифровки);
```

Тогда при двойном щечке на поле отчета появляется стандартное контекстное меню.

Однако если мы попробуем применить стандартную расшифровку в нашем отчете **ЗадолженностьКонтрагентов,** система выдаст ошибку (рис 7.3):

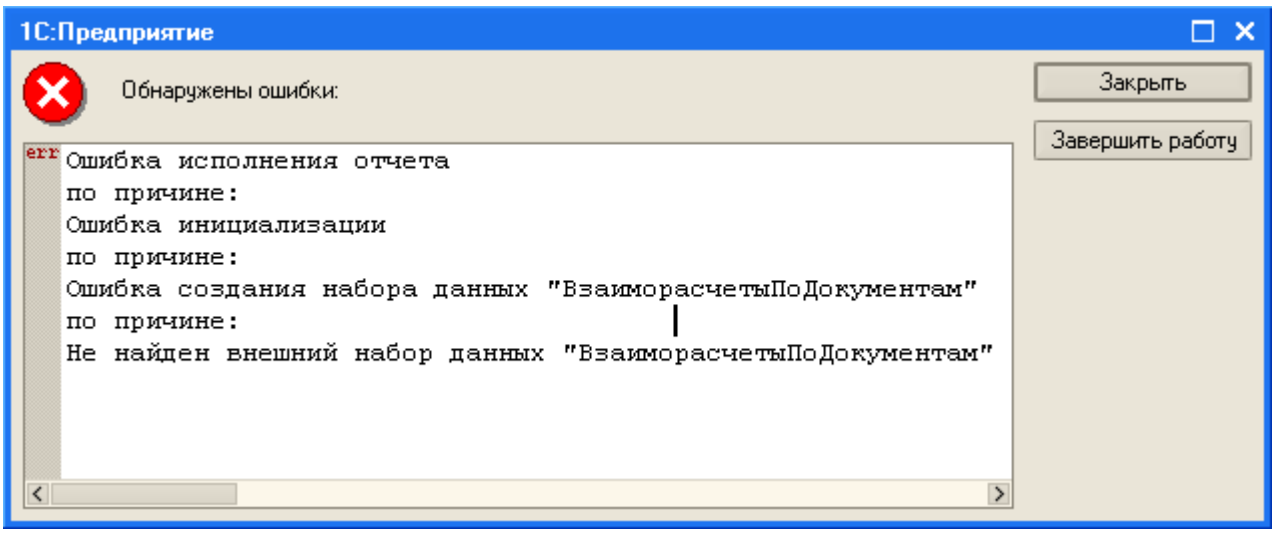


Рис. 7.3. Ошибка расшифровки

Это связано с тем, что система пытается провести процесс компоновки автоматически, не используя процедуру **СформироватьОтчет()**, и набор данных **ВзаиморасчетыПоДокументам** остается не инициализированным.

Но мы можем реализовать свой механизм расшифровки. В этом случае мы можем изменить список действий расшифровки, выполнить другие процедуры и пр.

Для этого в форме отчета нужно создать обработчик расшифровки для поля табличного документа **Результат** (рис 7.4)

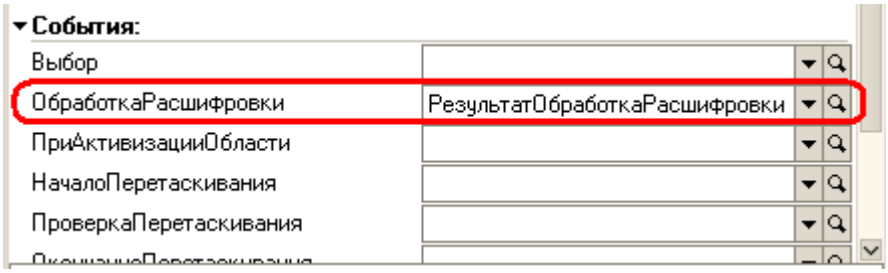


Рис. 7.4. Создание обработчика расшифровки

В модуле формы в процедуру обработчика расшифровки внесем следующий код:

```
Процедура РезультатОбработкаРасшифровки(Элемент, Расшифровка, СтандартнаяОбработка)

    Перем ВыполненноеДействие;

    // Запретим стандартную обработку расшифровки
    СтандартнаяОбработка = Ложь;
    // Создадим и инициализируем обработчик расшифровки
    ОбработкаРасшифровки = Новый ОбработкаРасшифровкиКомпоновкиДанных(ДанныеРасшифровки, Новый
ИсточникДоступныхНастроекКомпоновкиДанных(СхемаКомпоновкиДанных));

    ДоступныеДействия = Новый Массив();
    ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.ОткрытьЗначение);
    ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.Отфильтровать);
    ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.Упорядочить);
    ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.Расшифровать);

    // Осуществим выбор действия расшифровки пользователем
    Настройки = ОбработкаРасшифровки.Выполнить(Расшифровка, ВыполненноеДействие, ДоступныеДействия);
    Если Настройки <> Неопределено Тогда
        // Пользователь выбрал действие, для которого нужно менять настройки

        Если ВыполненноеДействие = ДействиеОбработкиРасшифровкиКомпоновкиДанных.Упорядочить Тогда
            КомпоновщикНастроек.ЗагрузитьНастройки(Настройки);
            СформироватьОтчет(ЭлементыФормы.Результат, ДанныеРасшифровки);
```

```
            Иначе
                // При других действиях - создадим новый отчет, откроем форму, сформируем отчет в
ней
                Отчет = ВнешниеОтчеты.Создать(ИспользуемоеИмяФайла);
                Форма = Отчет.ПолучитьФорму("ФормаОтчета");
                Форма.Открыть();
                Отчет.КомпоновщикНастроек.ЗагрузитьНастройки(Настройки);
                Отчет.СформироватьОтчет(Форма.ЭлементыФормы.Результат, Форма.ДанныеРасшифровки);
            КонецЕсли;
        КонецЕсли;

    КонецПроцедуры
```

Здесь мы отменяем стандартную обработку расшифровки, создаем новый обработчик расшифровки – объект **ОбработкаРасшифровкиКомпоновкиДанных**, затем создаем список доступных действий расшифровки **ОткрытьЗначение**, **Отфильтровать**, **Упорядочить**, **Расшифровать**, выполняем выбор действия расшифровки, загружаем настройки для выбранной расшифровки и переформировываем отчет.

Рассмотрим более сложную задачу, например, при щелчке на ячейке с детальной записью нужно сформировать документ оплаты на оставшуюся неоплаченной сумму документа отгрузки и открыть его форму.

Так как пользователь может щелкнуть на любом поле детальной записи, нам нужно найти поле **Документ** в этой же детальной записи. Для того чтобы сделать это в отчете с использованием СКД, сначала надо разобраться с взаимосвязью элементов расшифровки.

Элементы расшифровки – это массив специальных объектов, который доступен сразу после выполнения отчета в свойстве **ДанныеРасшифровки.Элементы.** Элементом массива может быть объект **ЭлементРасшифровкиКомпоновкиДанныхГруппировка** или **ЭлементРасшифровкиКомпоновкиДанныхПоля.** Каждый элемент типа **ЭлементРасшифровкиКомпоновкиДанныхПоля** соответствует одной ячейке отчета с данными (кроме общих итогов). Рассмотрим структуру элементов расшифровки для сравнительно простой структуры нашего отчета (рис. 7.5):

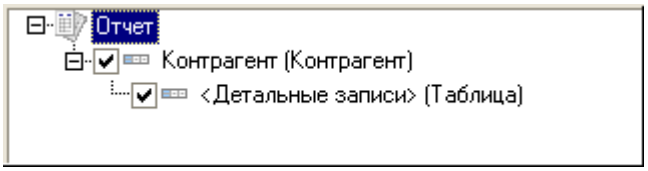


Рис. 7.5. Структура отчета

Иерархия элементов расшифровки для такой структуры изображена на схеме (рис. 7.6):

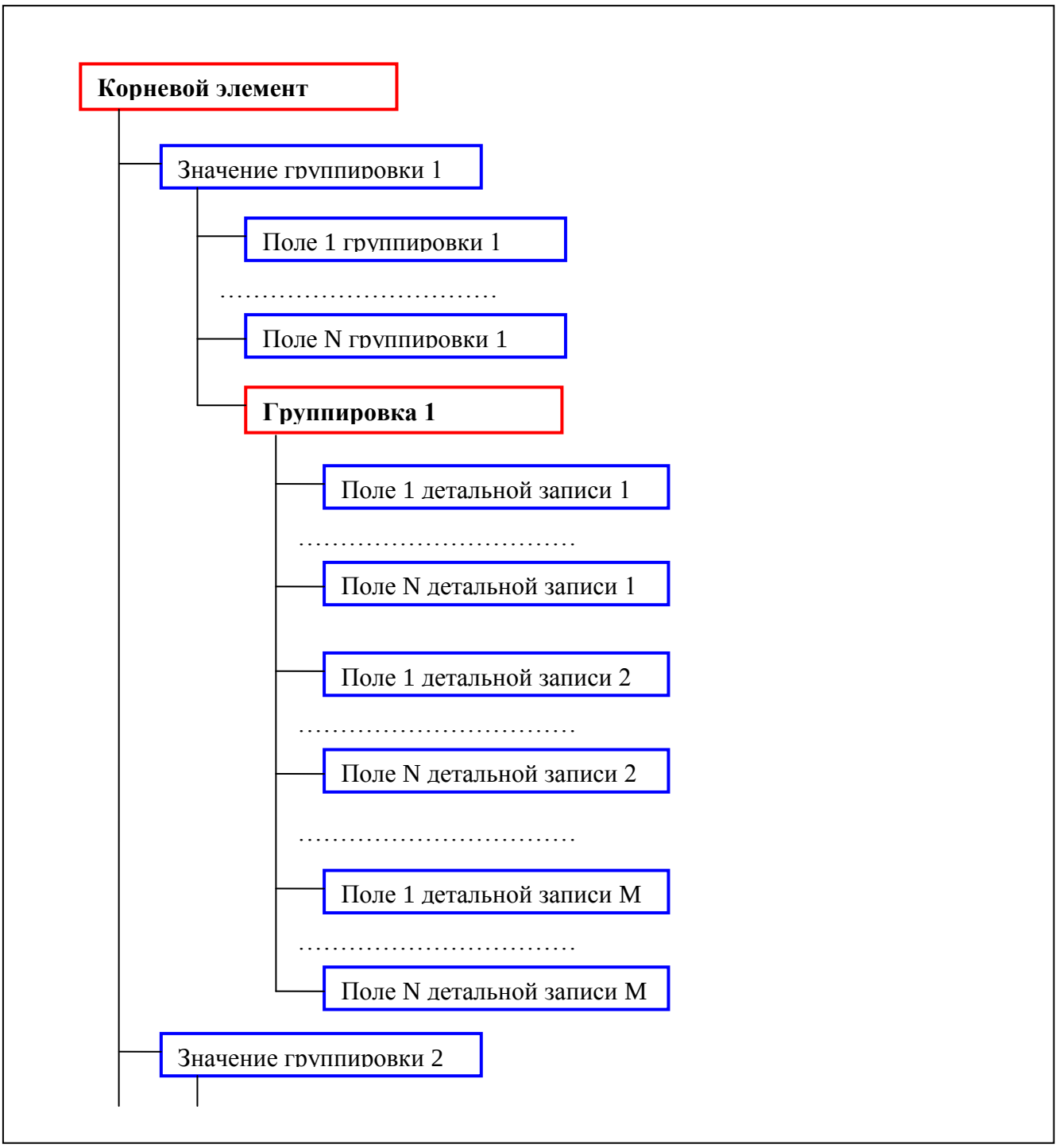


Рис. 7.6. Иерархия элементов расшифровки

Где N – количество полей в детальной записи.
M – количество детальных записей в группировке 1.
Синим цветом выделены элементы типа **ЭлементРасшифровкиКомпоновкиДанныхПоля.**
Красным цветом выделены элементы типа **ЭлементРасшифровкиКомпоновкиДанныхГруппировка.**

Как мы видим из схемы, поля разных детальных записей одной группировки находятся на одном уровне, поэтому нет никакой возможности найти определенное поле в конкретной детальной записи используя только иерархию элементов.

К сожалению, универсального алгоритма поиска нужных полей расшифровки не существует, так как способ поиска сильно зависит от структуры отчета. Для структуры нашего отчета я

придумал алгоритм поиска в массиве элементов расшифровке по индексу. Вот полный текст процедуры обработчика расшифровки с алгоритмом поиска:

```
Процедура РезультатОбработкаРасшифровки(Элемент, Расшифровка, СтандартнаяОбработка)

   Перем ВыполненноеДействие;

   ДополнительныеДействия = Новый СписокЗначений;
   ЭлементРасшифровки = ДанныеРасшифровки.Элементы[Расшифровка];

   ЗначениеРасшифровки = ЭлементРасшифровки.ПолучитьПоля()[0].Значение;
   ИдентификаторРодителя = ЭлементРасшифровки.ПолучитьРодителей()[0].Идентификатор;
   РодительскийЭлементРасшифровки = ДанныеРасшифровки.Элементы[ИдентификаторРодителя];

   Если ТипЗнч(РодительскийЭлементРасшифровки) = Тип("ЭлементРасшифровкиКомпоновкиДанныхГруппировка")
Тогда
    // это детальная запись
        // Определим количество актуальных полей в структуре отчета
                КоличествоПолей = 0;
                Для каждого Поле Из КомпоновщикНастроек.Настройки.Выбор.Элементы Цикл
                        Если Поле.Использование Тогда
                          КоличествоПолей = КоличествоПолей + 1;
                        КонецЕсли;
                КонецЦикла;
    // Определим количество актуальных группировок в структуре отчета
                КоличествоГруппировок = 0;
                Для каждого ЭлементСтруктуры Из КомпоновщикНастроек.Настройки.Структура Цикл
                        Если ТипЗнч(ЭлементСтруктуры) = Тип("ГруппировкаКомпоновкиДанных") И
ЭлементСтруктуры.Использование
                                И ЭлементСтруктуры.Имя <> "Таблица" И ЭлементСтруктуры.Имя <> "" Тогда
                          КоличествоГруппировок = КоличествоГруппировок + 1;
                        КонецЕсли;
                КонецЦикла;
                // Определим количество полей детальных записей
                КоличествоПолейДетальныхЗаписей = КоличествоПолей - КоличествоГруппировок;
                // Определим индекс первого поля в данной группировке
                ИндексПервогоПоля = 0;
                Для Ы = 1 По ДанныеРасшифровки.Элементы.Количество() - 1 Цикл
                        Родители = ДанныеРасшифровки.Элементы[Ы].ПолучитьРодителей();
                        Если Родители.Количество() > 0 И Родители[0].Идентификатор =
ИдентификаторРодителя Тогда
                                ИндексПервогоПоля = Число(ДанныеРасшифровки.Элементы[Ы].Идентификатор);
                                Прервать;
                        КонецЕсли;
                КонецЦикла;
                // Определяем номер (с нуля) по порядку текущей детальной записи
                НомерДетальнойЗаписи = Цел((Число(Расшифровка) - ИндексПервогоПоля) /
КоличествоПолейДетальныхЗаписей);
    // Найдем документ и сумму недостающей оплаты по документу из значений полей расшифровки
                НайденныйДокумент = Неопределено;
                СуммаОплаты = 0;
                Для Ы = 0 По КоличествоПолейДетальныхЗаписей - 1 Цикл
                        Индекс = НомерДетальнойЗаписи * КоличествоПолейДетальныхЗаписей + Ы +
ИндексПервогоПоля;
                        ПолеРасшифровки = ДанныеРасшифровки.Элементы[Индекс].ПолучитьПоля()[0];
                        ЗначениеРасшифровки = ПолеРасшифровки.Значение;
                        Если ТипЗнч(ЗначениеРасшифровки) = Тип("ДокументСсылка.РасходнаяНакладная") Тогда
                          НайденныйДокумент = ЗначениеРасшифровки;
                        ИначеЕсли ПолеРасшифровки.Поле = "СуммаДолга" Тогда
                          СуммаОплаты = ЗначениеРасшифровки;
                        КонецЕсли;
                КонецЦикла;
                // Группировки по контрагентам может и не быть, поэтому контрагента возьмем из документа

                Если НЕ НайденныйДокумент = Неопределено Тогда
                  ДополнительныеДействия.Добавить("Создать документ оплаты");
                КонецЕсли;

        КонецЕсли;


        // Запретим стандартную обработку расшифровки
        СтандартнаяОбработка = Ложь;
        // Создадим и инициализируем обработчик расшифровки
        ОбработкаРасшифровки = Новый ОбработкаРасшифровкиКомпоновкиДанных(ДанныеРасшифровки, Новый
ИсточникДоступныхНастроекКомпоновкиДанных(СхемаКомпоновкиДанных));

        ДоступныеДействия = Новый Массив();
```

```
        ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.ОткрытьЗначение);
        ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.Отфильтровать);
        ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.Упорядочить);
        ДоступныеДействия.Добавить(ДействиеОбработкиРасшифровкиКомпоновкиДанных.Расшифровать);


        // Осуществим выбор действия расшифровки пользователем
        Настройки = ОбработкаРасшифровки.Выполнить(Расшифровка, ВыполненноеДействие, ДоступныеДействия,
ДополнительныеДействия);
        Если Настройки <> Неопределено Тогда
                // Пользователь выбрал действие, для которого нужно менять настройки

                Если ВыполненноеДействие = ДействиеОбработкиРасшифровкиКомпоновкиДанных.Упорядочить Тогда
                        КомпоновщикНастроек.ЗагрузитьНастройки(Настройки);
                        СформироватьОтчет(ЭлементыФормы.Результат, ДанныеРасшифровки);
                Иначе
                        // При других действиях - создадим новый отчет, откроем форму, сформируем отчет в
ней
                        Отчет = ВнешниеОтчеты.Создать(ИспользуемоеИмяФайла);
                        Форма = Отчет.ПолучитьФорму("ФормаОтчета");
                        Форма.Открыть();
                        Отчет.КомпоновщикНастроек.ЗагрузитьНастройки(Настройки);
                        Отчет.СформироватьОтчет(Форма.ЭлементыФормы.Результат, Форма.ДанныеРасшифровки);
                КонецЕсли;

        ИначеЕсли ВыполненноеДействие = "Создать документ оплаты" Тогда

                ДокументОбъект = Документы.ОплатаПокупателя.СоздатьДокумент();
                ДокументОбъект.Дата = ДатаОтчета;
                ДокументОбъект.ДокументОснование = НайденныйДокумент;
                ДокументОбъект.Контрагент = НайденныйДокумент.Контрагент;
                ДокументОбъект.Сумма = СуммаОплаты;
                ДокументОбъект.ПолучитьФорму("ФормаДокумента").Открыть();

        КонецЕсли;

КонецПроцедуры
```

В данном коде обратим внимание на несколько моментов. Для того, чтобы определить, является выделенная ячейка частью детальной записи или группировки, необходим обратиться к родительскому элементу элемента расшифровки:

```
   ИдентификаторРодителя = ЭлементРасшифровки.ПолучитьРодителей()[0].Идентификатор;
   РодительскийЭлементРасшифровки = ДанныеРасшифровки.Элементы[ИдентификаторРодителя];

   Если ТипЗнч(РодительскийЭлементРасшифровки) = Тип("ЭлементРасшифровкиКомпоновкиДанныхГруппировка")
Тогда
   ……
```

Определяя номер по порядку текущей детальной записи, мы считаем, что все ресурсы по детальным записям рассчитаны, то есть элементы расшифровки есть для каждого поля:

```
    НомерДетальнойЗаписи = Цел((Число(Расшифровка)-ИндексПервогоПоля)/КоличествоПолейДетальныхЗаписей);
```

Поиск документа выполняем, перебирая все элементы из текущей детальной записи:

```
НайденныйДокумент = Неопределено;
СуммаОплаты = 0;
Для Ы = 0 По КоличествоПолейДетальныхЗаписей - 1 Цикл
   Индекс = НомерДетальнойЗаписи * КоличествоПолейДетальныхЗаписей + Ы + ИндексПервогоПоля;
   ПолеРасшифровки = ДанныеРасшифровки.Элементы[Индекс].ПолучитьПоля()[0];
   ЗначениеРасшифровки = ПолеРасшифровки.Значение;
   Если ТипЗнч(ЗначениеРасшифровки) = Тип("ДокументСсылка.РасходнаяНакладная") Тогда
       НайденныйДокумент = ЗначениеРасшифровки;
   ИначеЕсли ПолеРасшифровки.Поле = "СуммаДолга" Тогда
       СуммаОплаты = ЗначениеРасшифровки;
   КонецЕсли;
КонецЦикла;
```

Для того чтобы в стандартное меню расшифровки добавить свои пункты, необходимо создать список значений с этими пунктами:

```
ДополнительныеДействия = Новый СписокЗначений;
ДополнительныеДействия.Добавить("Создать документ оплаты");
```

Затем при вызове обработчика расшифровке указываем этот список:

```
Настройки = ОбработкаРасшифровки.Выполнить(Расшифровка, ВыполненноеДействие, ДоступныеДействия,
ДополнительныеДействия);
```

И если пользователь выбирает в этом меню дополнительное действие, анализируем значение параметра **ВыполненноеДействие**:

```
ИначеЕсли ВыполненноеДействие = "Создать документ оплаты" Тогда

        ДокументОбъект = Документы.ОплатаПокупателя.СоздатьДокумент();
        ДокументОбъект.Дата = ДатаОтчета;
        ДокументОбъект.ДокументОснование = НайденныйДокумент;
        ДокументОбъект.Контрагент = НайденныйДокумент.Контрагент;
        ДокументОбъект.Сумма = СуммаОплаты;
        ДокументОбъект.ПолучитьФорму("ФормаДокумента").Открыть();

КонецЕсли;
```

Контекстное меню расшифровки будет выглядеть так (рис 7.7):

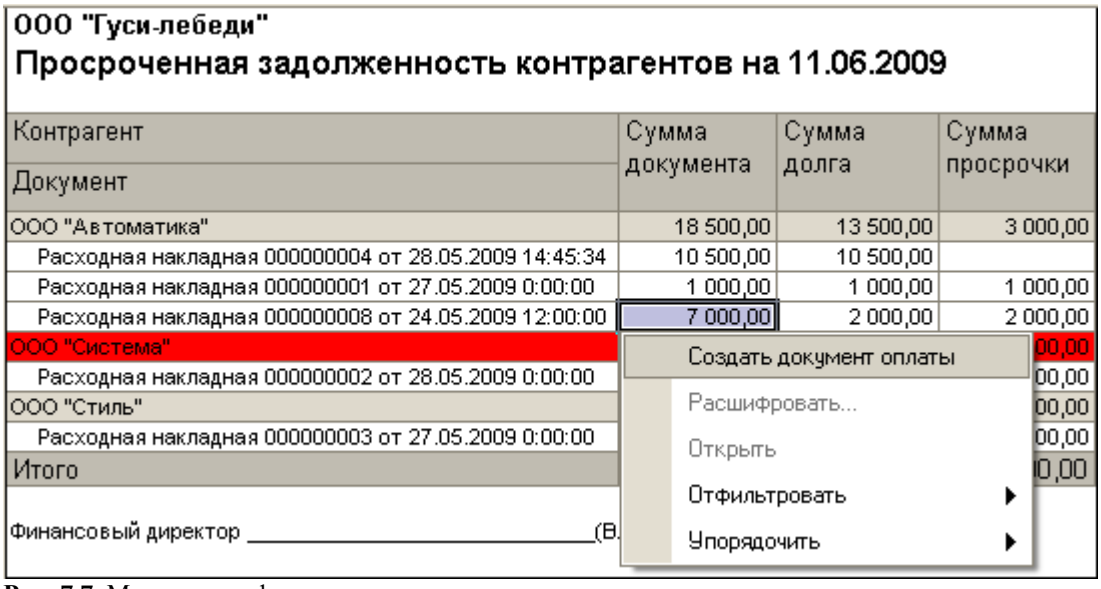


Рис. 7.7. Меню расшифровки

При выборе пункта «Создать документ оплаты» откроется форма нового документа **Оплата покупателя** (рис 7.8):

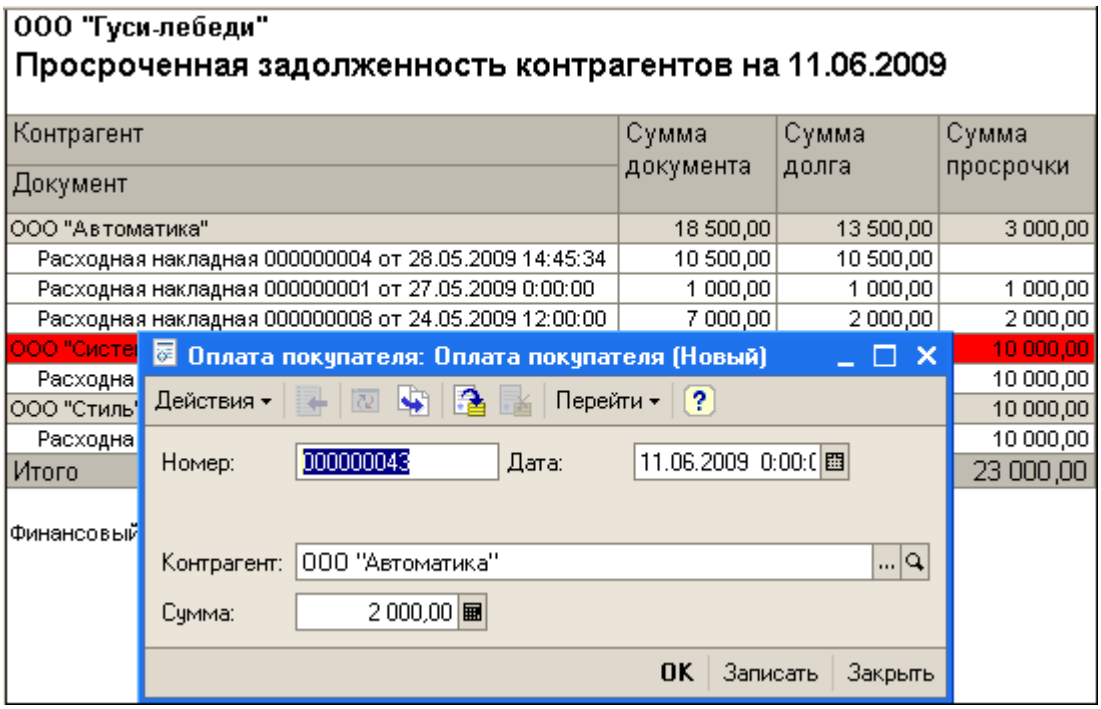


Рис. 7.8. Документ Оплата покупателя