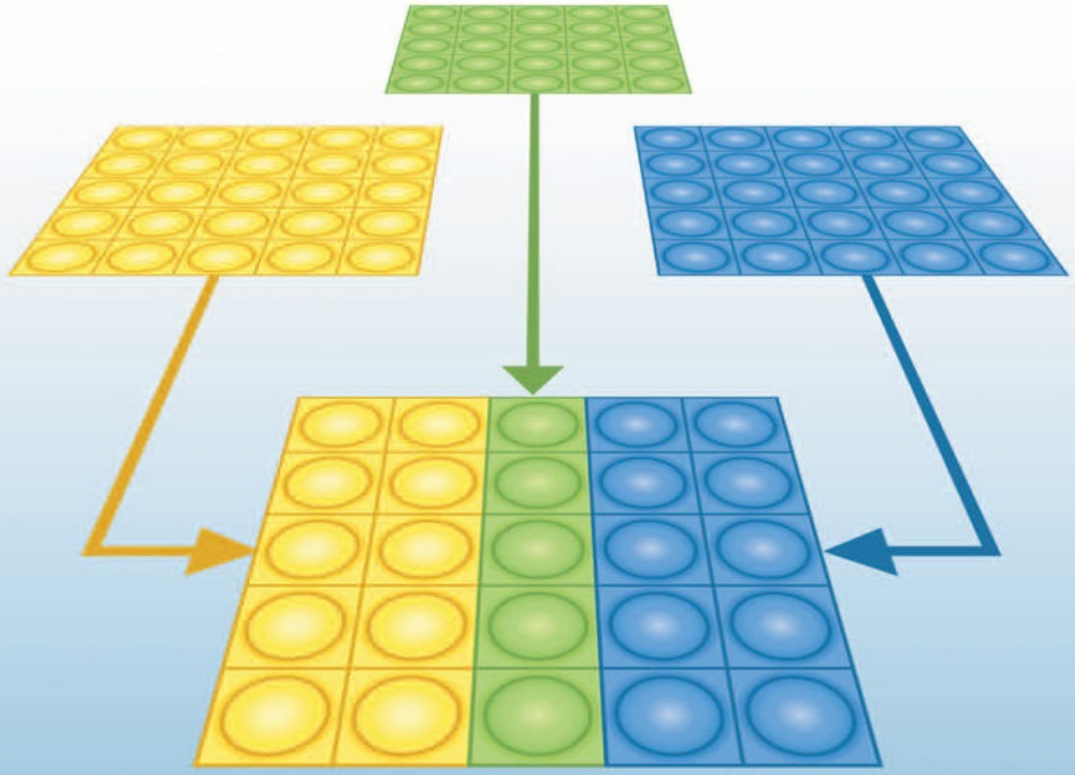
БИБЛИОТЕКА РАЗРАБОТЧИКА

Е. Ю. Хрусталева

ЯЗЫК ЗАПРОСОВ «1С:ПРЕДПРИЯТИЯ 8»



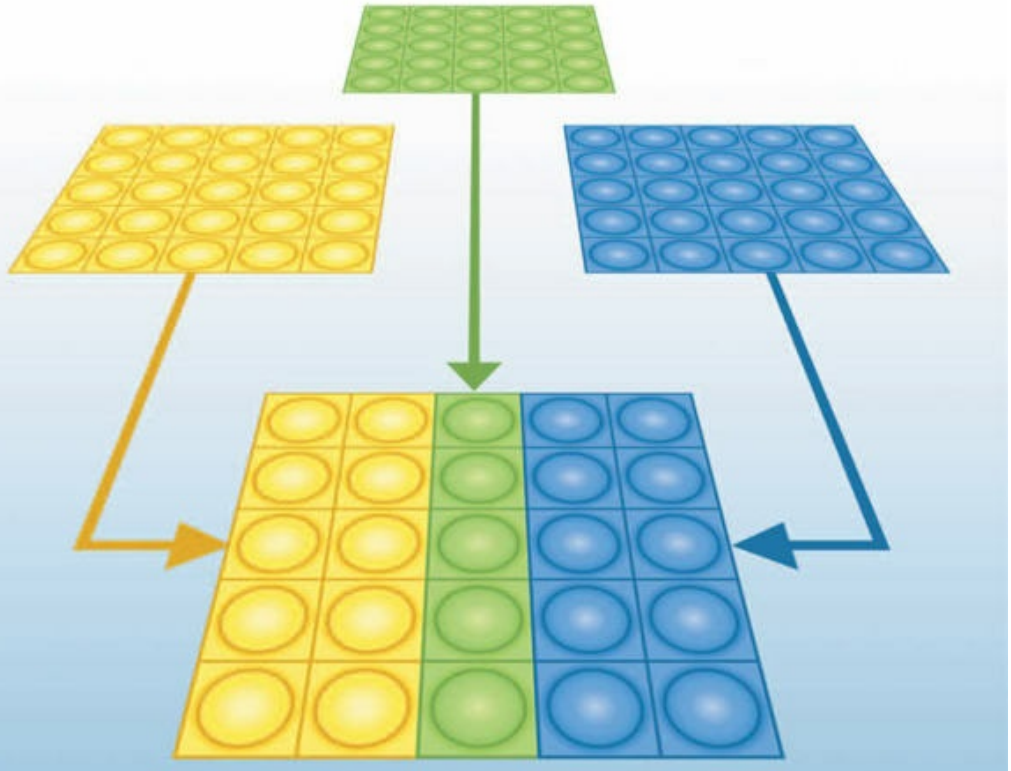
+CD

1С ПАБЛИШИНГ

БИБЛИОТЕКА РАЗРАБОТЧИКА

Е. Ю. Хрусталева

ЯЗЫК ЗАПРОСОВ «1С:ПРЕДПРИЯТИЯ 8»



+CD

1С ПАБЛИШИНГ

Е. Ю. Хрусталева

Язык запросов «1С:Предприятия 8»

Е. Ю. Хрусталева

Язык запросов «1С:Предприятия 8»

Электронная книга в формате ePub; ISBN 978-5-9677-1991-2.

Версия издания от 16.09.2013.

Электронный аналог издания "Язык запросов «1С:Предприятия 8»"
(ISBN 978-5-9677-1987-5, М.: ООО "1С-Паблишинг", 2013;
артикул печатной книги по прайс-листу фирмы "1С": 4601546108029;
по вопросам приобретения печатных изданий издательства "1С-Паблишинг"
обращайтесь к партнеру "1С",
обслуживающему вашу организацию, или к другим партнерам фирмы "1С", в магазины "1С Интерес", а также в книжные и интернет-магазины).

Запросы – это один из базовых механизмов «1С:Предприятия» наряду со встроенным языком, который позволяет читать и обрабатывать данные, хранящиеся в базе. Для составления запросов «1С:Предприятие» использует собственный язык, основанный на SQL.
Эта книга поможет начинающим разработчикам, не знакомым с SQL, освоить язык запросов «1С:Предприятия». Книга также будет полезна и тем, кто имеет опыт составления SQL-запросов в других средах разработки, поскольку язык запросов «1С:Предприятия» содержит значительное количество расширений, ориентированных на специфику финансово-экономических задач.
В книге рассматривается значительное количество практических примеров. Для создания примеров использована версия 8.3.3.687 платформы «1С:Предприятие».
Книга выпущена под редакцией Максима Радченко.

Дополнительные материалы

Приложение к книге включает демонстрационные конфигурации, иллюстрирующие эти примеры, и учебная версия платформы "1С:Предприятие 8.3".

Скачайте материалы и учебную версию на странице http://its.1c.ru/book_demo/, раскройте архив и следуйте инструкциям по установке.

Интернет-конференция для начинающих
разработчиков http://devtrainingforum.v8.1c.ru/forum/.

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

Глава 1. Механизм запросов

Как хранятся данные в «1С:Предприятии»

Прикладные решения, разработанные на платформе «1С:Предприятие», работают с данными, которые интерактивно вводит пользователь, заполняя различные формы ввода справочников, документов и т. д. Из этих форм данные с помощью встроенного языка записываются в базу данных и хранятся в ней. На основе введенных данных пользователю обычно требуется получить некоторую обобщенную информацию, необходимые ему итоговые данные, отчеты и т. д. Для этого используется механизм запросов, который рассматривается в данной книге.

Поэтому прежде чем начинать осваивать язык запросов, важно понять, как хранятся данные в «1С:Предприятии». Для хранения данных «1С:Предприятие» использует реляционные базы данных. *Реляционная база данных* представляет собой совокупность различной информации, представленной в виде двумерных *таблиц*. Таблица базы данных состоит из набора строк и столбцов. Каждая строка (*запись*) этой таблицы характеризуется рядом значений, содержащихся в ее столбцах (*полях*).

Для примера можно привести данные о клиентах компании, хранящиеся в справочнике клиентов. В самом упрощенном виде данные о клиентах хранятся в одной – *основной* таблице, где каждому клиенту соответствует одна запись, имеющая один и тот же набор полей, например, *Код*, *Наименование*, *Адрес*, *Телефон* и т. п. (табл. 1.1).

Таблица 1.1. Справочник клиентов в информационной базе (основная таблица)

Ссылка	Код	Наименование	Адрес	Телефон
Ref1	000000001	Соколов Иван Андреевич	Москва, …	8-916-222-33-55
Ref2	000000002	Орлов Сергей Иванович	Москва, …	8-926-555-66-77
Ref3	000000003	Маслова Ирина Николаевна	Санкт-Петербург, …	8-915-4447799

Рассмотрим подробнее, как формируется структура таблицы справочника в информационной базе «1С:Предприятия».

Например, при создании в конфигураторе справочника *Клиенты* платформа «1С:Предприятие» автоматически создает в информационной базе основную таблицу этого справочника с полями *Ссылка*, *Код*, *Наименование, ПометкаУдаления*, *Предопределенный* и *ВерсияДанных*.

ПРИМЕЧАНИЕ

Имена полей *Код*, *Наименование* и т. п. мы используем для простоты изложения. На самом деле имена полей таблиц в базе данных будут техногенными, например, *_Code*, *_Description* и т. п.

Затем пользователь в режиме *1С:Предприятие* заполняет справочник данными, которые сохраняются в базе данных (рис. 1.1).

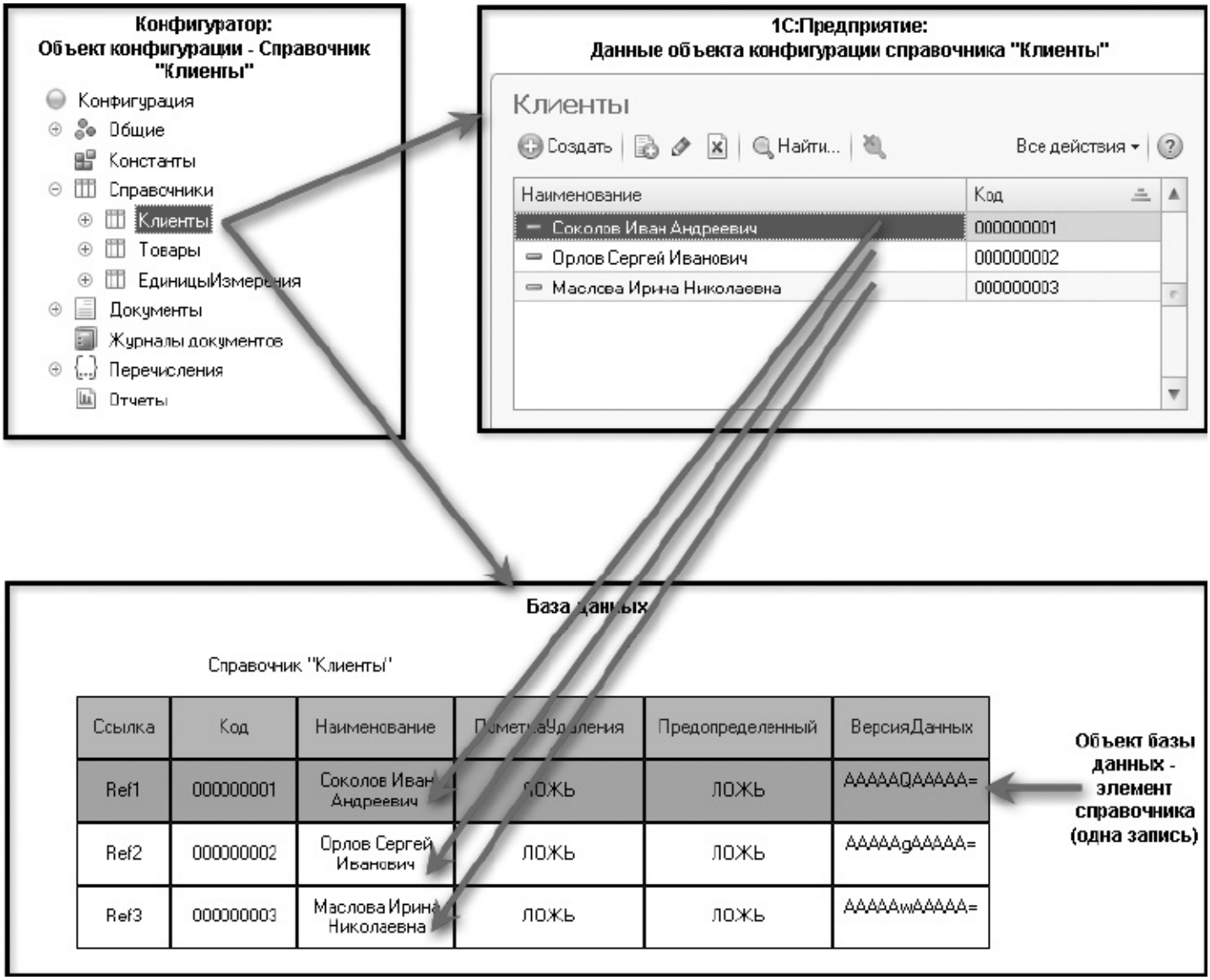


Рис. 1.1. Структура простого справочника в конфигураторе, в «1С:Предприятии» и в информационной базе

Поле *Ссылка* является уникальным идентификатором записи о клиенте, поля *Код*, *Наименование, ПометкаУдаления*, *Предопределенный* и *ВерсияДанных* являются стандартными реквизитами, которые платформа добавляет в любой справочник.

На рис. 1.1 также показаны три понятия, которые важно отличать. В конфигураторе создается *объект конфигурации Справочник Клиенты*. *Данные этого объекта конфигурации* (все записи справочника) вводятся в режиме *1С:Предприятие*, затем эти данные записываются в таблицу справочника *Клиенты* в информационной базе. В этой таблице *объектом базы данных* является одна запись (данные одного элемента справочника), однозначно идентифицирующаяся значением поля *Ссылка*.

Каждый элемент справочника, как правило, содержит некоторую дополнительную информацию, которая подробнее описывает этот элемент. Например, все элементы справочника *Клиенты* могут содержать дополнительную информацию об адресе и телефоне каждого клиента. Для описания этой информации используются *реквизиты*

справочника.

При добавлении реквизитов справочника в конфигураторе платформа создает поля соответствующего типа в основной таблице справочника (рис. 1.2).

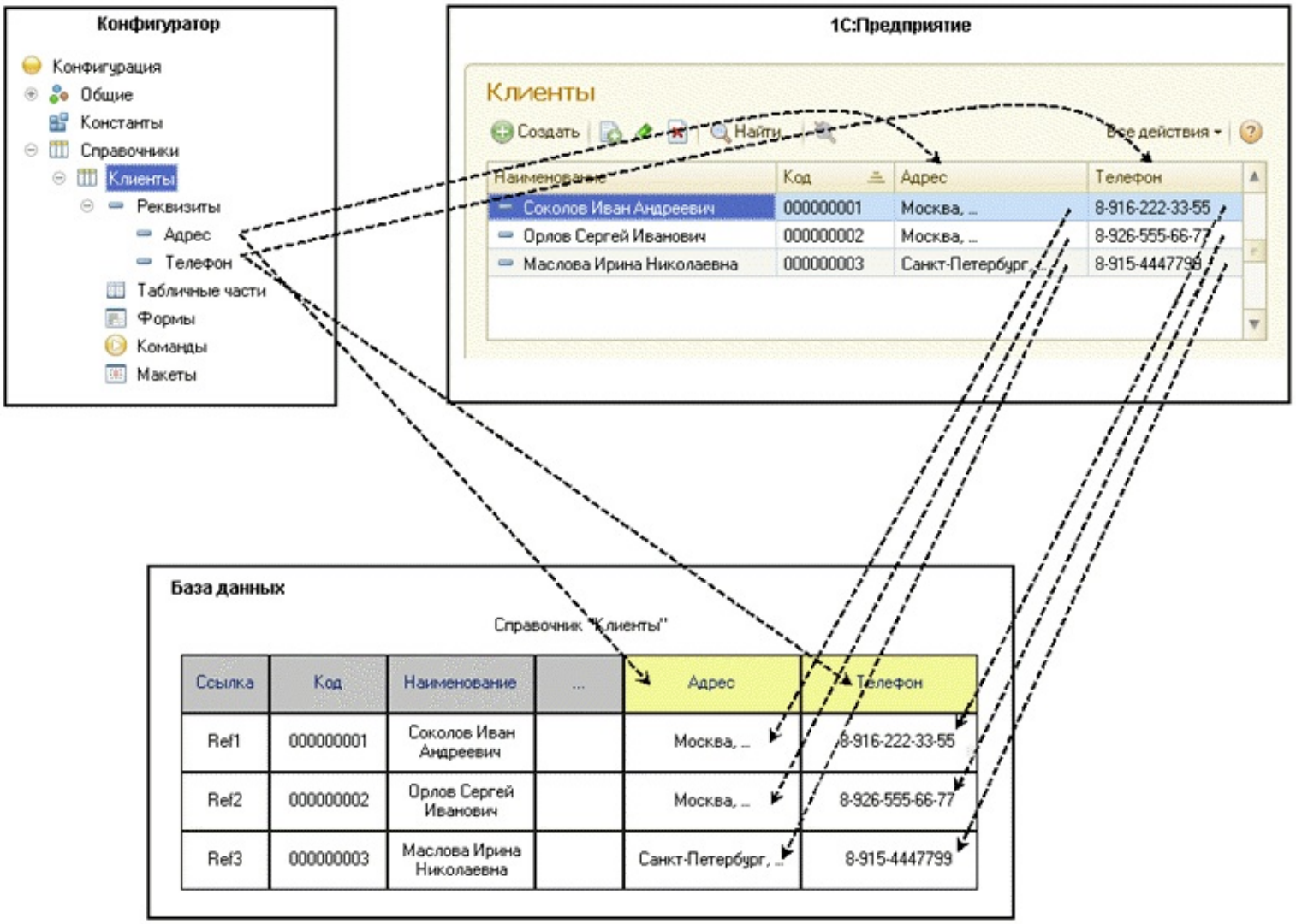


Рис. 1.2. Структура справочника, имеющего реквизиты, в конфигураторе, в «1С:Предприятии» и в информационной базе

Если справочник является иерархическим, то платформа добавляет в структуру основной таблицы справочника в информационной базе поле *Родитель* и поле *ЭтоГруппа* (в случае, если справочник имеет тип иерархии *Иерархия групп и элементов*), рис. 1.3.

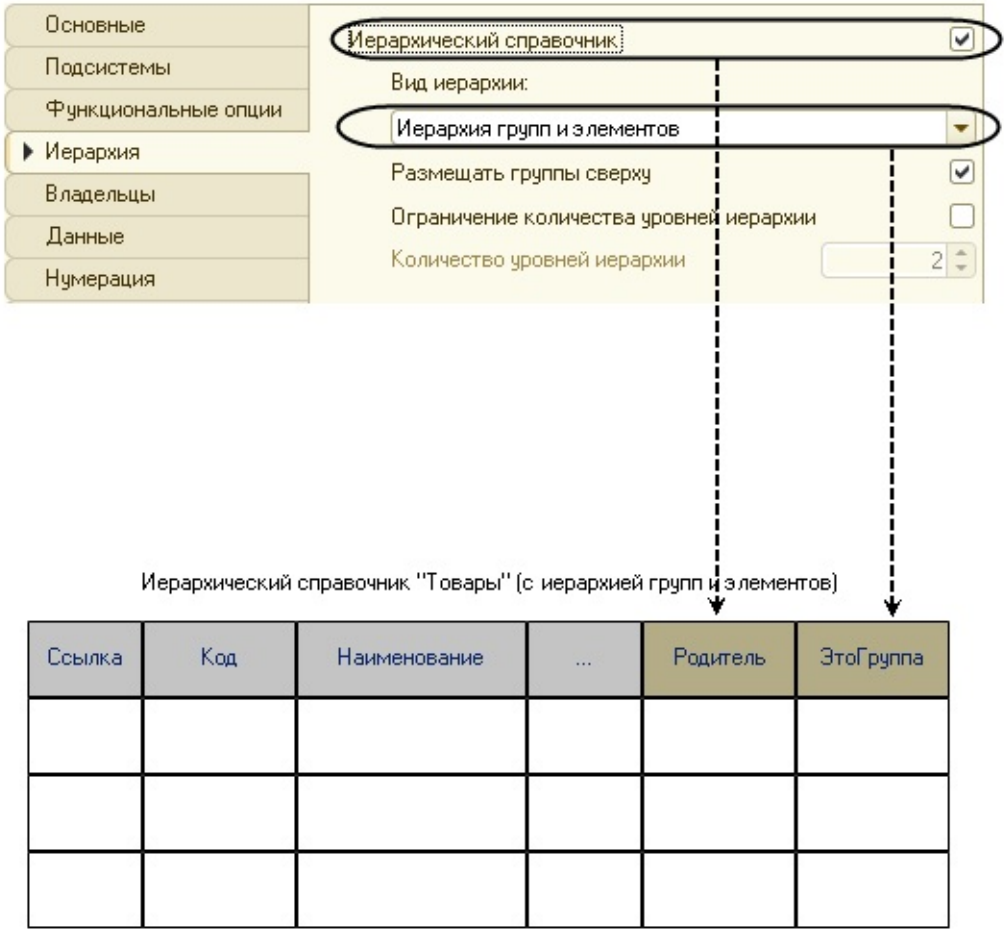


Рис. 1.3. Структура иерархического справочника в конфигураторе и в информационной базе

Например, справочник *Товары* является иерархическим с иерархией групп и элементов. При этом в таблице базы данных для записей, являющихся группой, поле *ЭтоГруппа* принимает значение *Истина*; для записей, не являющихся группой, поле *ЭтоГруппа* принимает значение *Ложь*, а поле *Родитель* является ссылкой на родительскую запись. Благодаря этому можно получить информацию о дочерних записях и родителях для каждого элемента справочника (рис. 1.4).

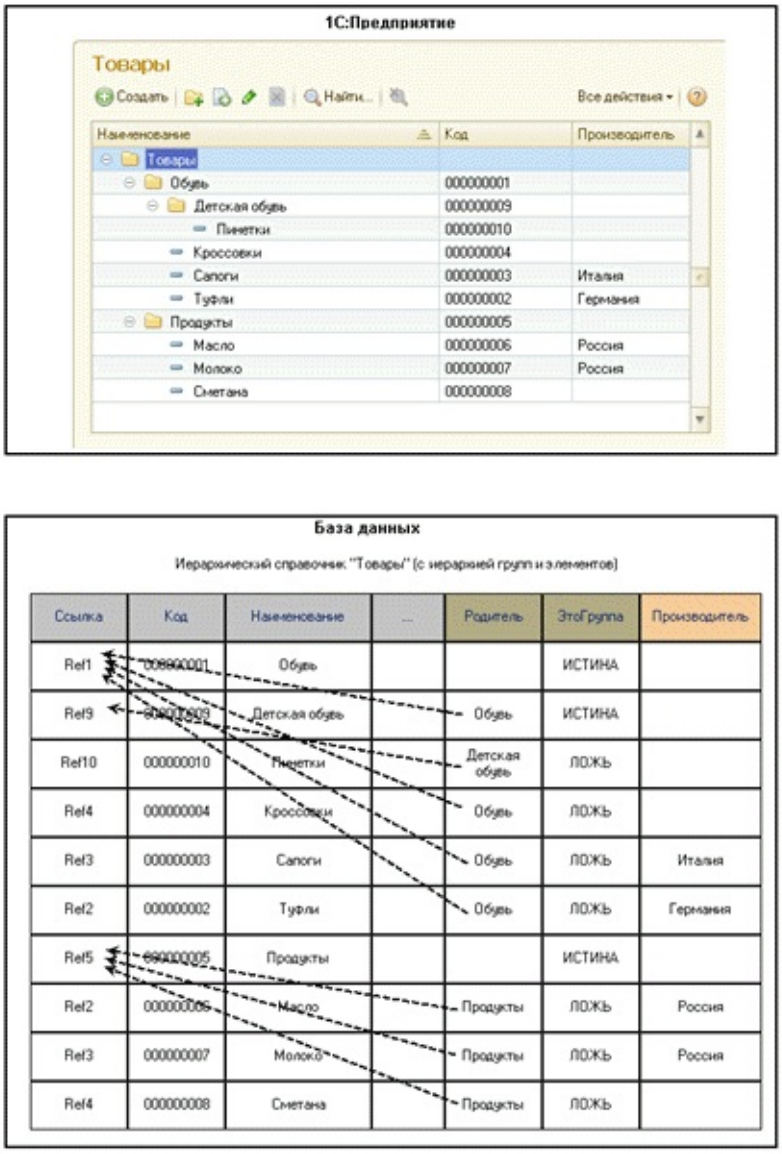


Рис. 1.4. Содержимое иерархического справочника в «1С:Предприятии» и в информационной базе

ПРИМЕЧАНИЕ

Поле *Родитель* на самом деле хранит ссылку на родительскую запись, но для большей ясности в таблице на рис. 1.4 в этом поле отражено представление ссылки в виде наименования.

Если справочник является подчиненным, например справочник *РасчетныеСчета* подчинен справочнику *Поставщики*, то в основную таблицу подчиненного справочника платформа добавляет поле *Владелец*, которое ссылается на элемент справочника-владельца (рис. 1.5).

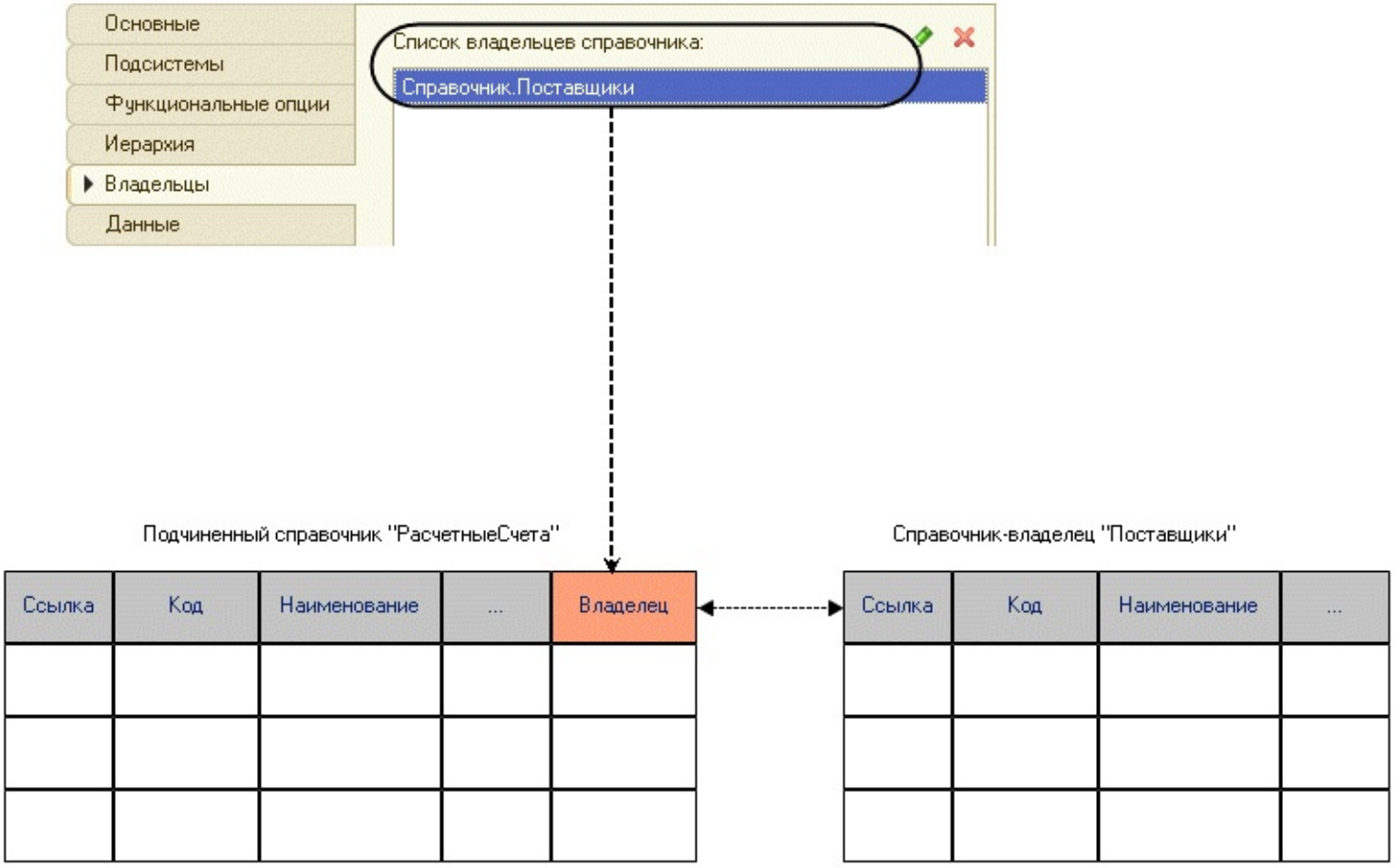


Рис. 1.5. Структура подчиненного справочника в конфигураторе и в информационной базе

Благодаря этому можно получить информацию, какие элементы справочника-владельца владеют какими элементами подчиненного справочника, например, какие расчетные счета относятся к конкретному поставщику (рис. 1.6).

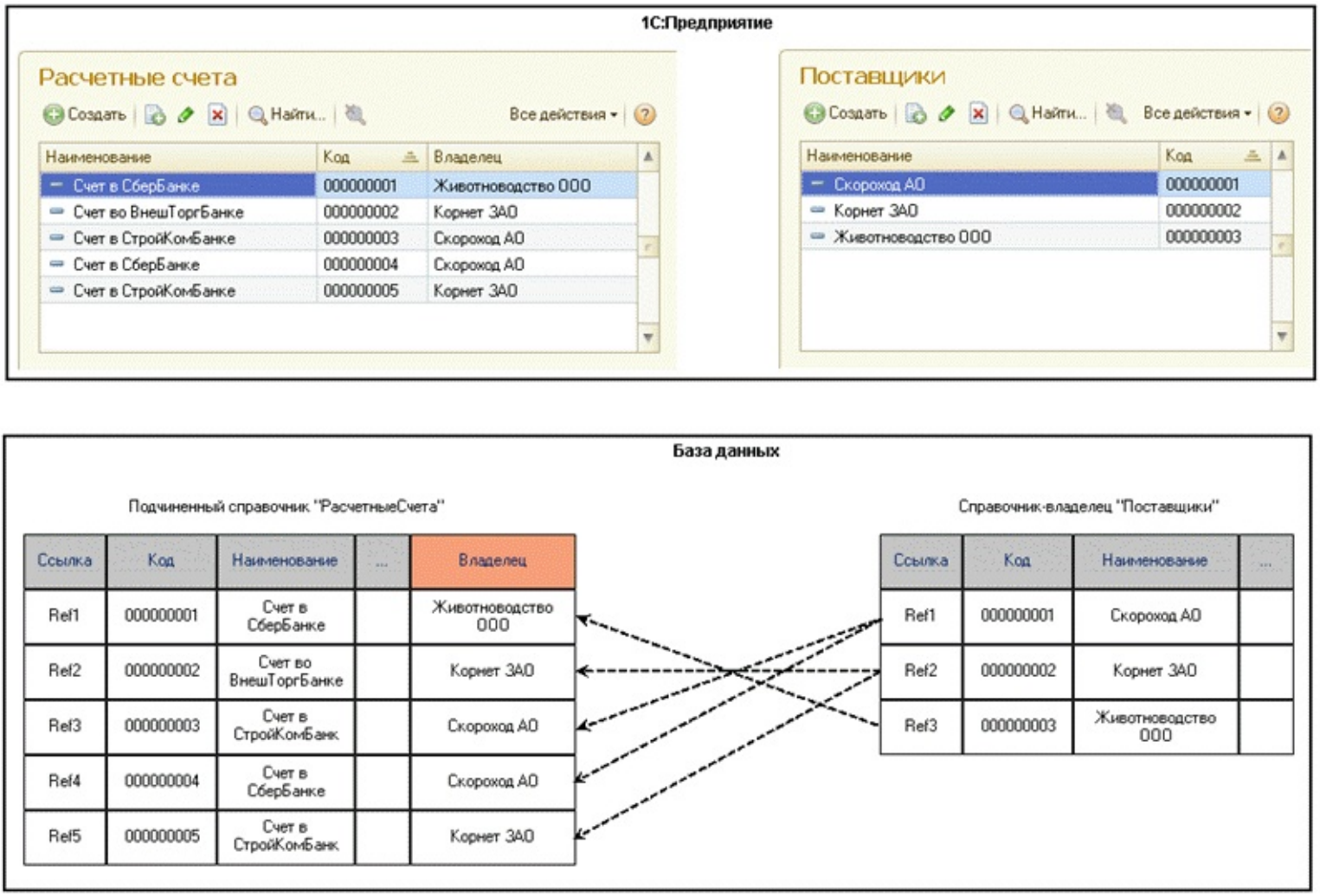


Рис. 1.6. Содержимое подчиненного справочника и справочника-владельца в «1С:Предприятии» и в информационной базе

ПРИМЕЧАНИЕ

Поле *Владелец* подчиненного справочника на самом деле хранит ссылку на запись справочника-владельца, но для большей ясности в таблице на рис. 1.6 в этом поле отражено представление ссылки в виде наименования.

Кроме реквизитов каждый элемент справочника может содержать некоторый набор информации, которая одинакова по своей структуре, но различна по количеству и относится к разным элементам справочника. Например, каждый элемент справочника *Поставщики* может содержать информацию о договорах, заключенных с этим поставщиком. Для каждого поставщика состав информации и количество записей в ней будет разным, а структура информации (например, дата начала и окончания действия договоров) – одинакова. Для описания подобной информации могут быть использованы *табличные части* справочника.

При добавлении в справочник табличной части в информационной базе создается подчиненная таблица со стандартными полями *Ссылка* и *НомерСтроки* и реквизитами табличной части, заданными в конфигураторе. Таблица, содержащая табличную часть, связана по полю *Ссылка* с основной таблицей. Благодаря этому можно получить информацию из табличной части, относящуюся к конкретному элементу справочника (рис. 1.7).

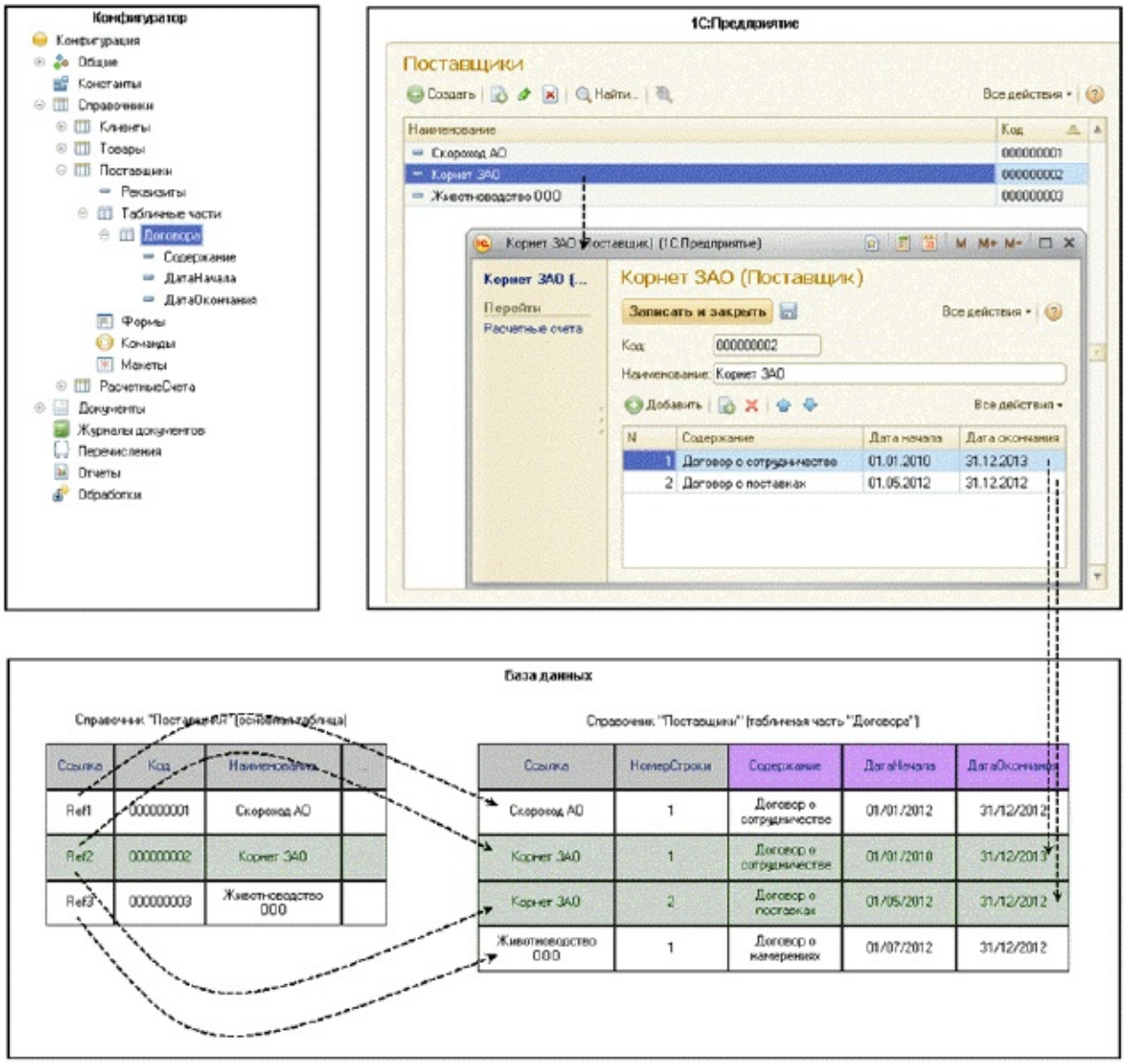


Рис. 1.7. Справочник с табличной частью в конфигураторе, в «1С:Предприятии» и в информационной базе

ПРИМЕЧАНИЕ

В поле *Ссылка* подчиненной таблицы, содержащей табличную часть, на самом деле хранится ссылка на запись основной таблицы, но для большей ясности в таблице на рис. 1.7 в этом поле отражено представление ссылки в виде наименования.

В информационной базе создается столько подчиненных таблиц, сколько табличных частей задано у справочника.

Таким образом, на примере справочника, имеющего табличную часть, мы видим, что одному объекту конфигурации в информационной базе могут соответствовать несколько таблиц – *основная* и одна или несколько *подчиненных* ей по полю *Ссылка* таблиц. При этом одному объекту базы данных соответствует одна запись в основной таблице и одна или несколько записей в подчиненных таблицах, содержащих табличные части (см. рис. 1.7).

Теперь рассмотрим другой пример, когда *поле ссылочного типа* служит для связи данных двух разных объектов конфигурации. Важно понимать, что ссылочные типы данных не существуют изначально в конфигурации, а появляются при создании

соответствующих объектов конфигурации. Причем для каждого объекта конфигурации во встроенном языке создается свой тип ссылки. То есть при создании справочника *Товары* появляется ссылочный тип данных *СправочникСсылка.Товары*, при создании справочника *Клиенты* – тип *СправочникСсылка.Клиенты*, при создании документа *Событие* – тип *ДокументСсылка.Событие* и т. д. Поля, содержащие данные такого типа, мы будем для краткости называть иногда *ссылочными полями*.

Например, в конфигурации существует документ, имеющий поле, ссылающееся на справочник *Клиенты* (рис. 1.8).

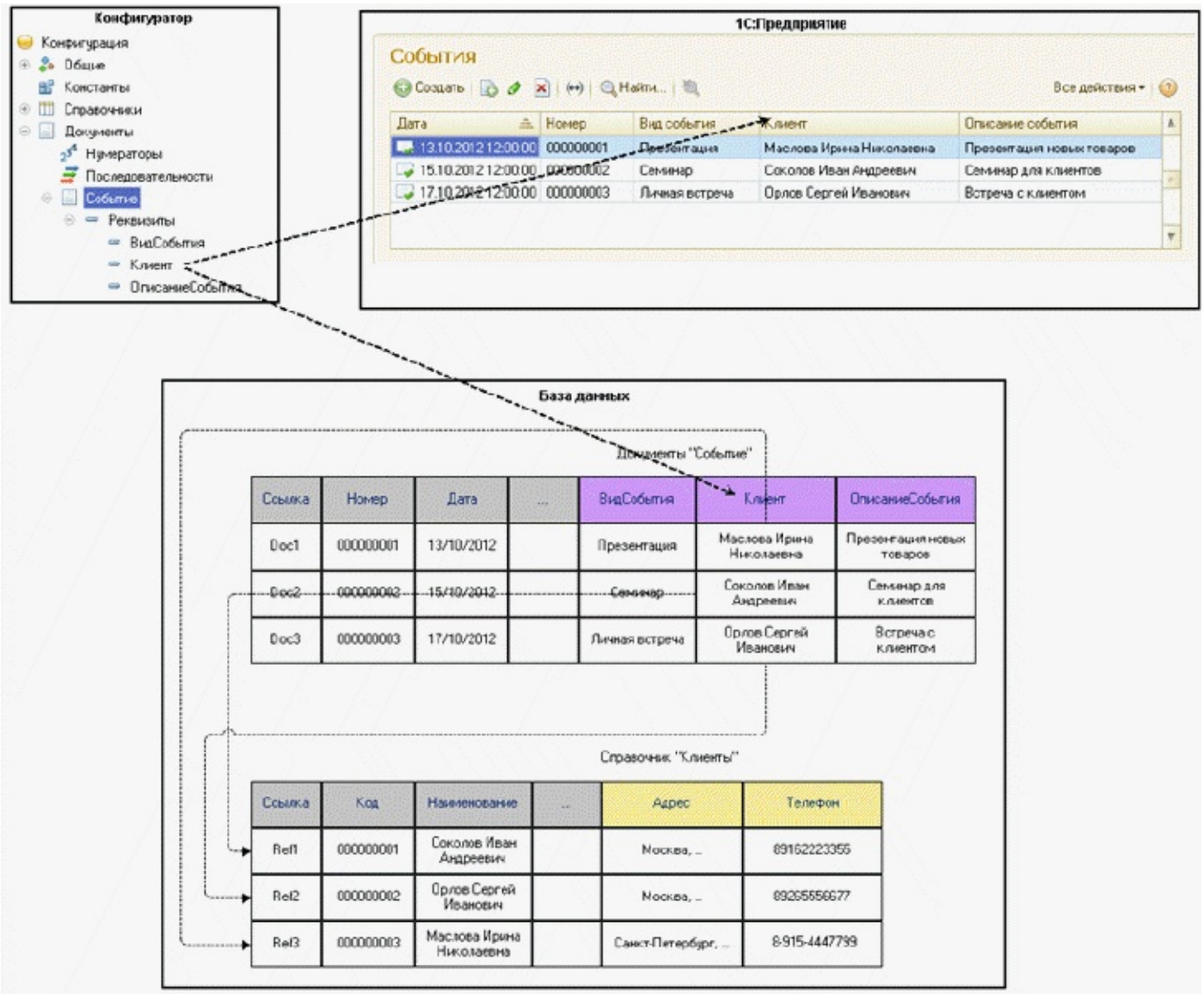


Рис. 1.8. Документ, имеющий поле ссылочного типа, в конфигураторе, в «1С:Предприятии» и в информационной базе

ПРИМЕЧАНИЕ

Поле *Клиент* документа *Событие* на самом деле хранит ссылку на запись справочника *Клиенты*, но для большей ясности в таблице на рис. 1.8 в этом поле отражено представление ссылки в виде наименования.

В приведенном примере поле *Клиент* документа *Событие* имеет ссылочный тип *СправочникСсылка.Клиенты*, а значениями этого поля являются ссылки на конкретные элементы справочника *Клиенты*. В этом случае, обращаясь к полю *Клиент* в документе,

мы можем получить любые данные о клиенте, на которого ссылается данное поле.

Таким образом, мы рассмотрели примеры хранения в информационной базе «1С:Предприятия» ссылочных типов данных, таких как справочники, документы, планы видов характеристик и т. д.

В заключение рассмотрим, как хранятся в информационной базе «1С:Предприятия» нессылочные данные, доступ к которым нельзя получить через поле *Ссылка*.

Например, периодический регистр сведений предназначен для хранения данных (ресурсов) в разрезе измерений с привязкой ко времени. Благодаря стандартному полю *Период*, регистр сведений может хранить не только актуальные значения данных, но и историю их изменения во времени.

На основе объекта конфигурации *Регистр сведений* платформа создает в информационной базе таблицу, в которой может храниться произвольная информация, «привязанная» к набору измерений и периоду. Например, периодический регистр сведений *Цены*, имеющий измерение *Товар* и ресурс *Цена*, хранит изменяющуюся во времени информацию о ценах на товары (рис. 1.9).

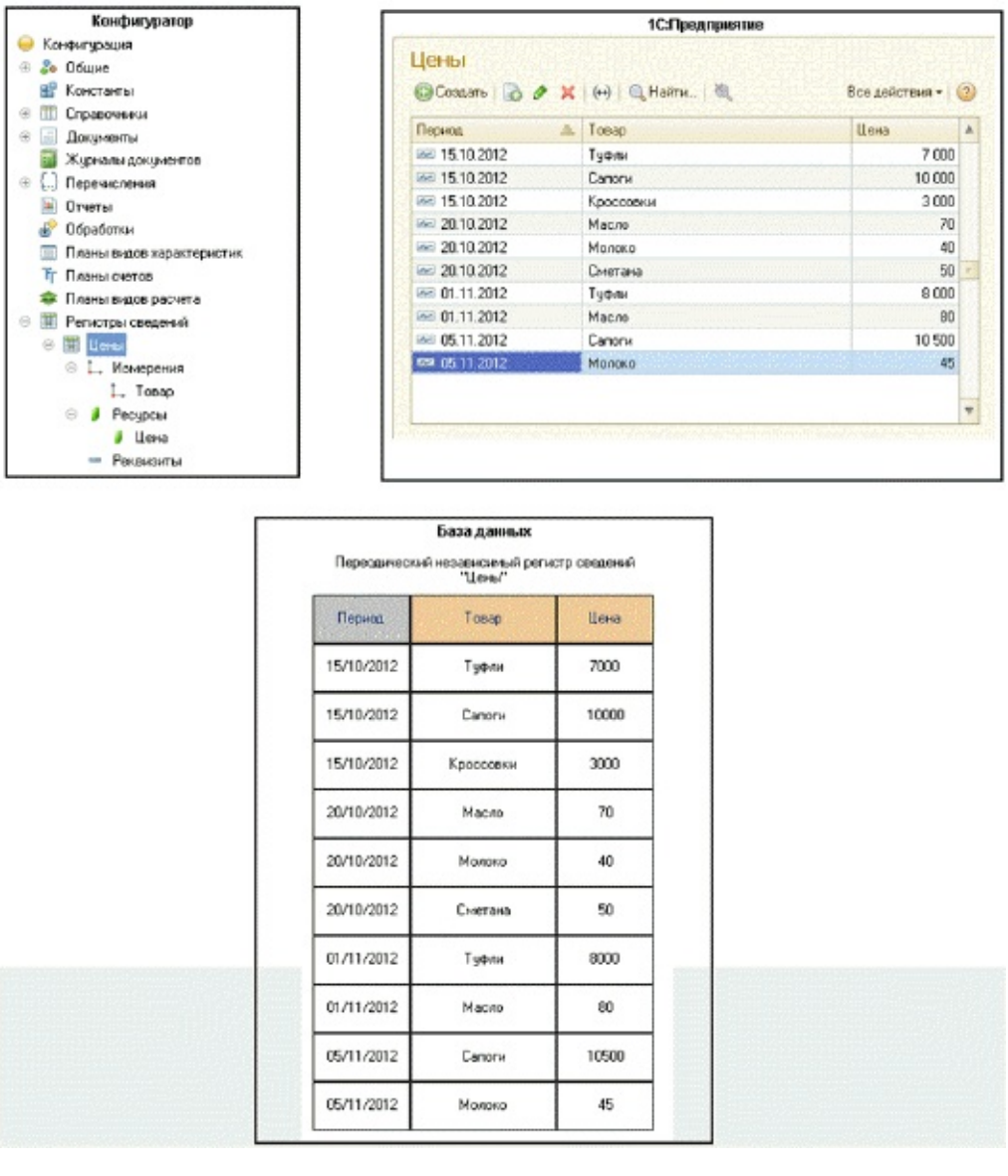


Рис. 1.9. Периодический регистр сведений в конфигураторе, в «1С:Предприятии» и в информационной базе

ПРИМЕЧАНИЕ

Поле *Товар* в таблице регистра на самом деле хранит ссылку на запись справочника *Товары*, но для большей ясности на рис. 1.9 в этом поле отражено представление ссылки в виде наименования.

Помимо измерений, ресурсов и реквизитов, заданных в конфигураторе, у таблицы периодического регистра сведений в информационной базе создается стандартное поле *Период*, благодаря этому регистр может хранить одинаковую информацию для одних и тех же измерений, но для различных периодов.

Как мы видим, в таблице регистра нет поля *Ссылка*, посредством которого мы можем сослаться на конкретную запись регистра. Ключом записи, однозначно идентифицирующим запись, является в случае периодического регистра сведений совокупность значений измерений регистра и периода.

В целом для нессылочных данных мы не можем получить какую-то конкретную запись из таблицы. Но мы можем получить некоторый набор записей по какому-либо условию (например, отобрать данные регистра сведений по периоду) и затем перебирать его в цикле.

Исходные таблицы для запросов

Прежде чем переходить к конкретным примерам использования языка запросов, остановимся также на составе таблиц базы данных, являющихся источниками запросов. Состав таблиц, доступных для запроса, и их описание мы можем увидеть в синтакс-помощнике в разделе *Работа с запросами* > *Таблицы запросов*.

Важно понимать, что прямого доступа к физическим таблицам, в которых хранится информация в базе данных, из «1С:Предприятия» получить нельзя. Это связано с тем, что в прикладном решении могут использоваться разные СУБД, имеющие свою специфику, а текст запроса должен быть универсальным и одинаково работать на любой используемой СУБД. Поэтому при выполнении запроса платформа автоматически транслирует текст запроса в набор инструкций, которые «понимает» конкретная СУБД. Кроме того, физические таблицы и поля в них имеют техногенные имена, из которых непонятно, что именно хранится в данном поле.

Поэтому с помощью запросов мы обращаемся к данным не напрямую, а через специальную «прослойку» в виде *таблиц языка запросов*. Этот процесс можно представить на следующей схеме (рис. 1.10).

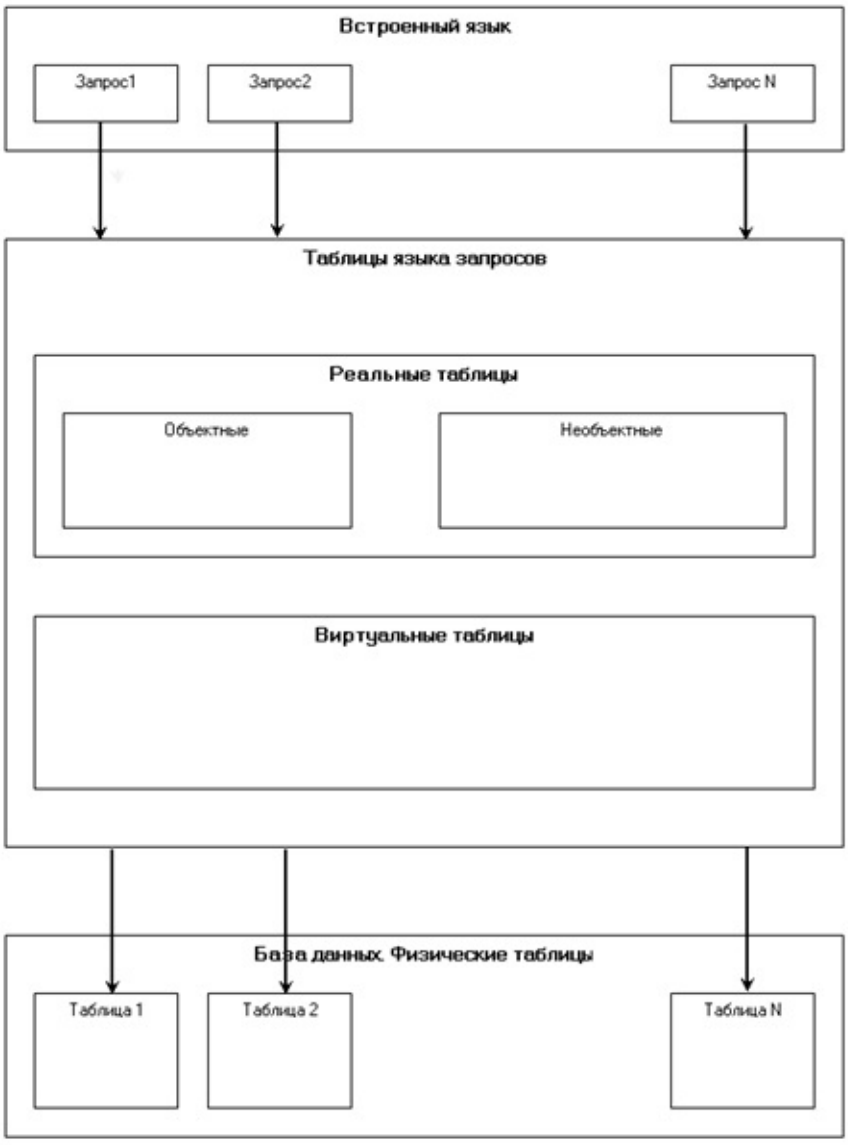


Рис. 1.10. Доступ к данным в «1С:Предприятии»

Таким образом, все таблицы, к которым можно обратиться с помощью языка запросов, являются придуманными, воображаемыми, в большей или меньшей степени соответствующими реальным физическим таблицам СУБД. Однако по степени похожести на физические таблицы их принято разделять на *реальные* и *виртуальные* таблицы.

Реальные таблицы

Отличительной особенностью реальных таблиц является то, что они содержат данные какой-либо одной физической таблицы, хранящейся в базе данных, и то, что реальная таблица очень похожа на свою физическую таблицу. Например, реальной является таблица *Справочник.Клиенты*, соответствующая справочнику *Клиенты,* или таблица *РегистрСведений.Цены*, соответствующая регистру сведений *Цены*.

Для примера сравним структуру реальной и физической таблицы, хранящей данные справочника (табл. 1.2).

Таблица 1.2. Реальная и физическая таблица справочника «Клиенты»

Реальная таблица	Физическая таблица

Справочник.Клиенты	_Reference<n>
Ссылка	_Id<suff>
ВерсияДанных	_Version
ПометкаУдаления	_Marked
Предопределенный	_IsMetadata
Родитель	_ParentId<suff>
Владелец	_OwnerId<suff>
ЭтоГруппа	_Folder
Код	_Code
Наименование	_Description
<Имя реквизита>	_Fld<n><suff>
<Имя общего реквизита>	_Fld<n><suff>
Представление	–
<Имя табличной части>	–

Как мы видим, не все поля реальной и физической таблицы соответствуют друг другу. Например, поле *Представление* – виртуальное, то есть оно не хранится в физической таблице базы данных, а генерируется в момент выполнения запроса.

подробнее

О поле *Представление* рассказано в разделе «Как получить текстовое представление ссылочного поля».

О поле *Имя табличной части* рассказано в разделе «Как получить данные из табличной части документа в качестве вложенной таблицы».

Но в целом реальная таблица очень похожа на физическую по набору полей, а также количество записей в обеих таблицах одинаково.

Реальные таблицы подразделяются на *объектные* (ссылочные) и *необъектные* (нессылочные).

В объектных (ссылочных) таблицах представлена информация ссылочных типов данных (справочники, документы, планы видов характеристик и т. д.). А в необъектных (нессылочных) – всех остальных типов данных (константы, регистры и т. д.).

Отличительной особенностью объектных (ссылочных) таблиц является то, что они включают в себя стандартное поле *Ссылка*, которое позволяет однозначно

идентифицировать каждую запись (данные об объекте базы данных). Эти таблицы могут быть иерархическими, подчиненными, и поля таких таблиц могут содержать вложенные таблицы (табличные части).

Виртуальные таблицы

Виртуальные таблицы формируются в момент выполнения запроса на основе реальных таблиц базы данных. Например, виртуальная таблица *РегистрСведений.Цены.СрезПоследних* формируется на основе таблицы регистра сведений *Цены*, рис. 1.11.

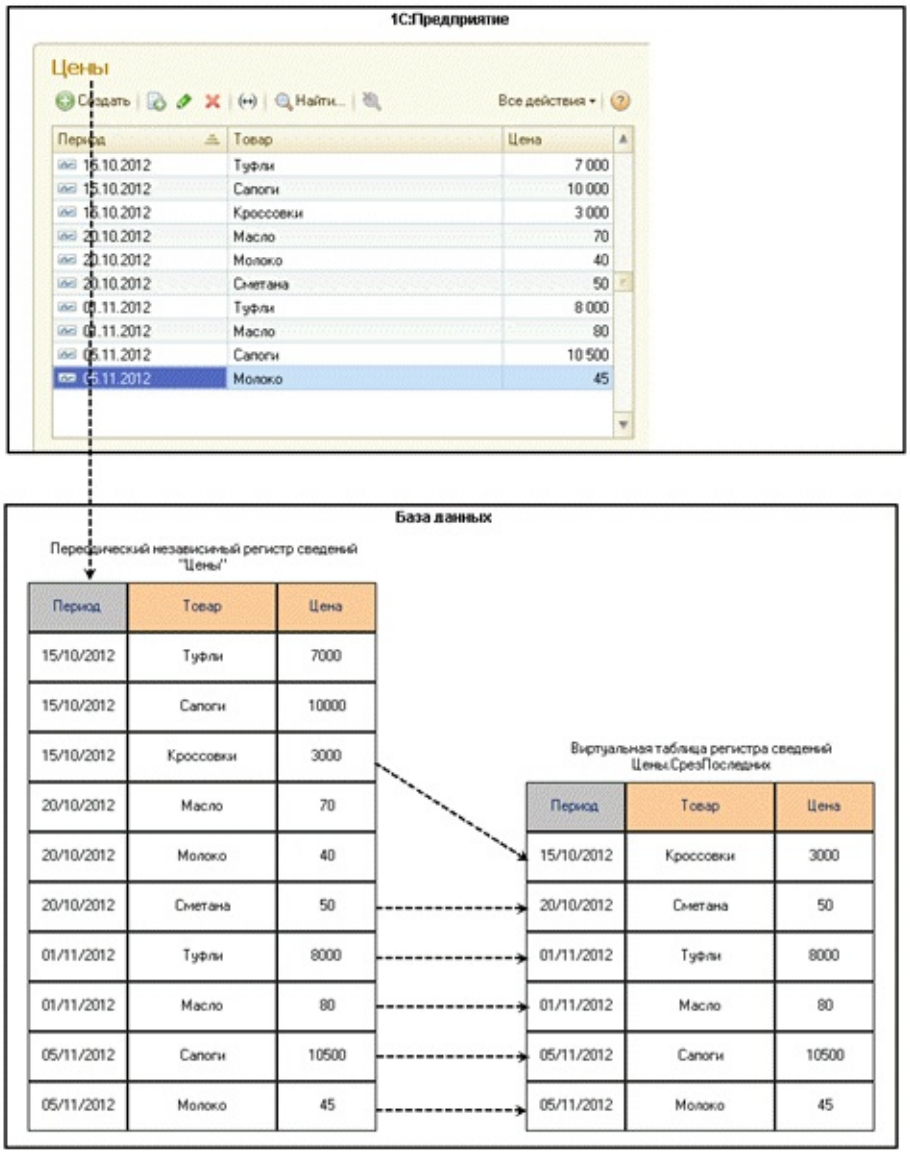


Рис. 1.11. Физическая и виртуальная таблица регистра сведений

ПРИМЕЧАНИЕ

Поле *Товар* в таблице регистра на самом деле хранит ссылку на запись справочника *Товары*, но для большей ясности на рис. 1.11 в этом поле отражено представление ссылки в виде наименования.

Как мы видим из рисунка 1.11, при заполнении цен товаров в регистре сведений *Цены* в «1С:Предприятии» данные за период по каждому товару сохраняются в физической таблице регистра сведений в базе данных.

Мы уже рассказывали выше про периодический регистр сведений (см. рис. 1.9). Благодаря стандартному полю *Период* регистр может хранить изменяющуюся во времени информацию для одних и тех же измерений, но для различных периодов.

При обращении запросом к виртуальной таблице *РегистрСведений.Цены.СрезПоследних* мы получим не все записи физической таблицы, а только последние по времени данные о ценах на товары. То есть срез последних записей регистра сведений возвращает по каждому значению измерения (*Товары*) одну наиболее позднюю (по времени, по значению поля *Период*) запись.

Таким образом, виртуальные таблицы мало похожи на какую-то физическую таблицу и содержат совсем иной состав записей, чем реальные таблицы.

Язык запросов «1С:Предприятия»

Механизм запросов позволяет получить доступ к разнообразной информации, хранящейся в базе данных «1С:Предприятия». Путем выполнения запроса к информационной базе из всей совокупности информации можно получить различные выборки данных из одной или нескольких взаимосвязанных таблиц, отобранных по определенному условию, отсортированных определенным образом и пр. Далее полученные данные могут быть проанализированы для решения различных прикладных задач, построения отчетов и т. п.

Однако следует иметь в виду, что с помощью запросов можно только прочитать нужную информацию из базы данных, но изменить ее и записать обратно при помощи запроса нельзя – для этого нужно использовать средства встроенного языка.

Общая схема выполнения запросов

Запрос формируется и выполняется разработчиком из встроенного языка. Для этого предназначены следующие программные объекты:

- *Запрос*,
- *РезультатЗапроса*,
- *ВыборкаИзРезультатаЗапроса*.

Не углубляясь в детали, рассмотрим самую распространенную и простейшую схему выполнения запроса.

подробнее

Другие варианты выполнения запросов и обработки их результатов будут рассмотрены позднее в разделе «Выполнение запросов из встроенного языка».

1. Сначала во встроенном языке создается объект *Запрос* (листинг 1.1).

Листинг 1.1. Создание запроса

```
Запрос = Новый Запрос;
```

- У объекта *Запрос* есть свойство *Текст*, в которое нужно поместить текст запроса, написанный на языке запросов. В тексте запроса описывается, какие данные, из каких таблиц нужно получить и как эти данные представить (листинг 1.2).

Листинг 1.2. Заполнение текста запроса

```
Запрос.Текст =
  "ВЫБРАТЬ
  |        Наименование
  |ИЗ
  |        Справочник.Товары";
```

- Далее запрос выполняется с помощью метода *Выполнить()* объекта *Запрос*. Именно в этот момент и происходит чтение данных из базы данных. Прочитанные данные возвращаются в виде объекта *РезультатЗапроса*, содержащего выбранные данные из базы данных (листинг 1.3).

Листинг 1.3. Выполнение запроса

```
РезультатЗапроса = Запрос.Выполнить();
```

- Чтобы обработать данные, содержащиеся в объекте *РезультатЗапроса*, из результата запроса получается выборка с помощью метода *Выбрать()*, который возвращает новый объект *ВыборкаИзРезультатаЗапроса*, то есть коллекцию данных, предназначенную для последовательного обхода ее элементов (листинг 1.4).

Листинг 1.4. Получение выборки из результата запроса

```
Выборка = РезультатЗапроса.Выбрать();
```

- Далее выборка обходится с помощью цикла *Пока Выборка.Следующий() Цикл*, а в теле цикла производятся какие-то действия над данными, полученными с помощью запроса (листинг 1.5).

Листинг 1.5. Обход выборки

```
Пока ВыборкаЗапроса.Следующий() Цикл
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = ВыборкаЗапроса.Наименование;
  Сообщение.Сообщить();

КонецЦикла;
```

В результате, если соединить вместе текст листингов 1.1–1.5, мы получим процедуру встроенного языка, в которой создается и выполняется запрос, в данном примере выводящий наименование всех товаров из справочника *Товары* в окно сообщений

(листинг 1.6).

Листинг 1.6. Вывод наименований всех товаров в окно сообщений

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        Наименование
  |ИЗ
  |        Справочник.Товары";

РезультатЗапроса = Запрос.Выполнить();
ВыборкаЗапроса = РезультатЗапроса.Выбрать();

Пока ВыборкаЗапроса.Следующий() Цикл
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = ВыборкаЗапроса.Наименование;
  Сообщение.Сообщить();

КонецЦикла;
```

Пример этой процедуры находится в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*. Результат выполнения данной процедуры представлен на рис. 1.12.

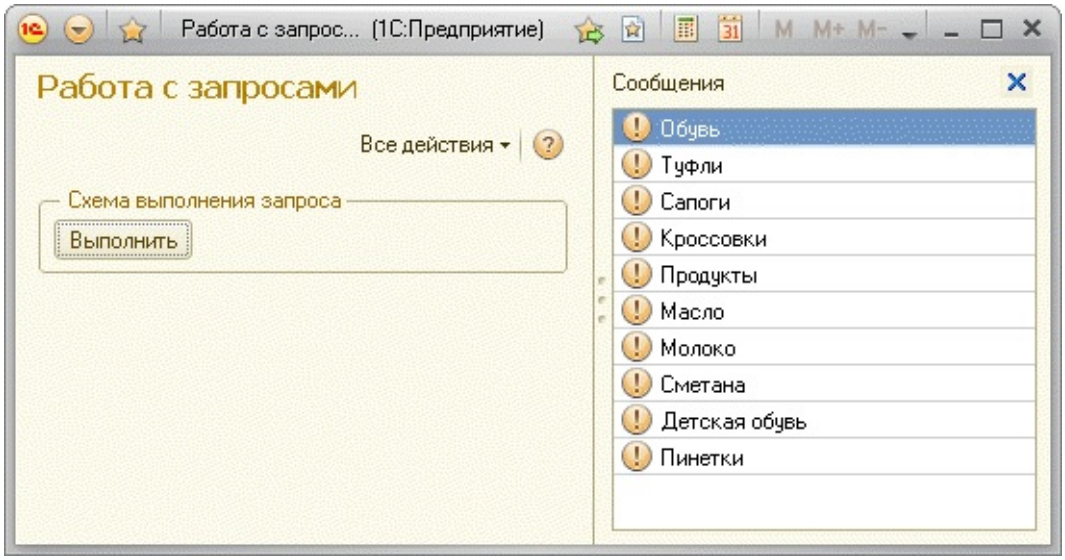


Рис. 1.12. Вывод наименований всех товаров в окно сообщений

Чтобы не усложнять восприятие материала, пока мы не будем подробно останавливаться на выводе и обработке результатов запросов. Сначала мы начнем изучать сам язык запросов, то есть научимся правильно составлять текст запросов. Для выполнения запросов и просмотра результатов мы будем использовать специальную обработку *Консоль запросов*. Данная обработка помогает отлаживать и просматривать результаты выполнения запросов в режиме *1С:Предприятие*.

Обработка *Консоль запросов* добавлена в демонстрационную конфигурацию «Язык запросов», прилагающуюся к книге на компакт-диске. Самая последняя версия этой обработки опубликована на ИТС (http://its.1c.ru/db/metod81#content:4500:1).

Синтаксис текста запросов

Язык запросов «1С:Предприятия» основан на стандартном SQL, но при этом содержит значительное количество расширений, ориентированных на финансово-экономические задачи, и значительно облегчает разработку бизнес-приложений.

Из определения следует, что язык запросов – мощный инструмент, предоставляющий разнообразные возможности получения данных. Но мы начнем его изучение с самых простых примеров, основанных на реальных небольших задачах, расположенных от простого к сложному. Но сначала немного теории.

Текст запроса состоит из нескольких частей (секций):

- описание запроса,
- объединение запросов,
- упорядочивание результатов,
- автоупорядочивание,
- описание итогов.

Обязательной частью запроса является только первая – описание запроса. Все остальные присутствуют в запросе по необходимости. Назначение каждой секции запроса будет рассмотрено ниже на конкретных примерах.

Для ознакомления приведем запрос, в котором присутствуют все указанные секции (рис. 1.13).

Рис. 1.13. Секции запроса

Синтаксически текст запроса состоит из набора *секций*, имеющих определенное назначение, например, выбрать записи из базы данных, отсортировать их, подсчитать итоги и т. д. Секции состоят из *предложений*, которые, в свою очередь, содержат *ключевые слова* (например, *ВЫБРАТЬ*, *ИЗ*, *ГДЕ* и т. п.), обозначающие определенное действие, которое нужно выполнить с базой данных. Ключевое слово, с которого начинается предложение, обычно дает название предложению языка запросов.

подробнее

Применение различных синтаксических конструкций языка запросов подробно описано во встроенной справке *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Синтаксис текста запросов*.

Одной из существенных особенностей языка запросов «1С:Предприятия» является то, что все ключевые слова имеют два варианта написания: на русском и английском языках. Поэтому язык написания запроса – дело привычки и вкуса, а результат выполнения запроса будет одинаков в обоих случаях.

В книге мы будем использовать ключевые слова языка запросов на русском языке, а для

тех, кто хочет использовать англоязычный вариант написания ключевых слов, они подробно описаны во встроенной справке.

подробнее

Встроенная справка: *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Синтаксис текста запросов > Двуязычное представление ключевых слов*.

Итак, начнем изучать язык запросов на конкретных примерах, от самых простых к более сложным. В процессе изучения мы узнаем, как и для чего используются различные ключевые слова языка запросов.

Для составления примеров будем использовать демонстрационную конфигурацию «Язык запросов», прилагающуюся к книге на компакт-диске.

Примеры использования языка запросов для получения данных из одной таблицы

Как получить все данные из таблицы

В нашей демонстрационной конфигурации существует справочник *Клиенты*. Предположим, нам нужно получить все данные из таблицы базы данных, соответствующей этому справочнику. Это можно сделать с помощью простейшего запроса, а результат посмотреть в консоли запросов.

Обработка *Консоль запросов* добавлена в демонстрационную конфигурацию «Язык запросов», и ее можно вызвать из группы команд *Сервис*. Но для того чтобы полностью использовать все функциональные возможности консоли запросов, ее нужно запускать в режиме *Толстый клиент*.

Запустим демонстрационную конфигурацию в этом режиме. Для этого в диалоге запуска «1С:Предприятия» нужно создать новую информационную базу с нашей конфигурацией, добавив базу из шаблона, установленного с диска, и указать в качестве основного режима запуска информационной базы *Толстый клиент* (рис. 1.14).

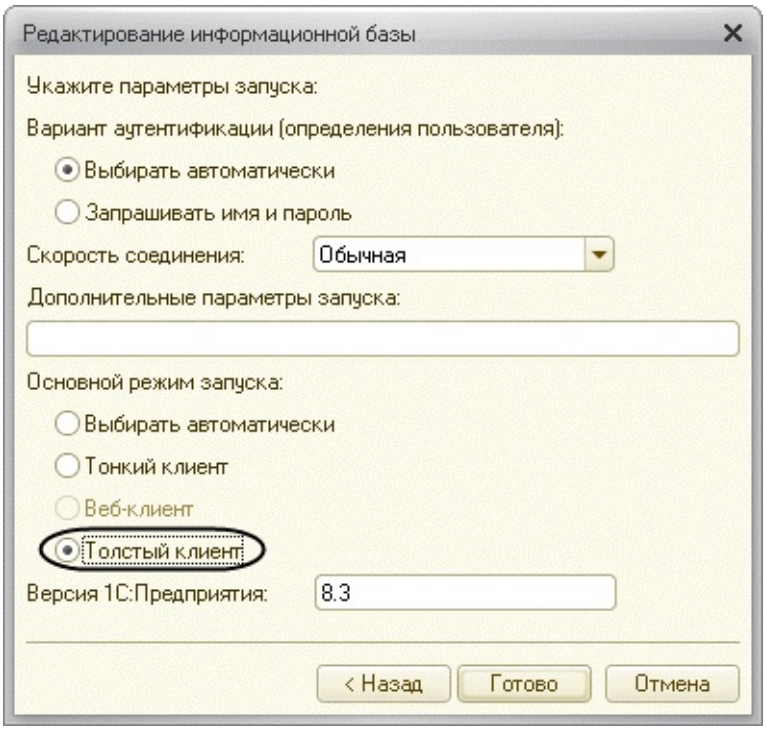


Рис. 1.14. Основной режим запуска демонстрационной конфигурации – «Толстый клиент»

Вызовем консоль запросов из группы команд *Сервис* и в среднем окне *Текст запроса* введем следующий текст (листинг 1.7).

Листинг 1.7. Вывод всех данных из таблицы

```
ВЫБРАТЬ
  Справочник.Клиенты.*
```

Нажмем кнопку *Выполнить* и в нижнем окне *Результат запроса* увидим результат выполнения нашего запроса (рис. 1.15).

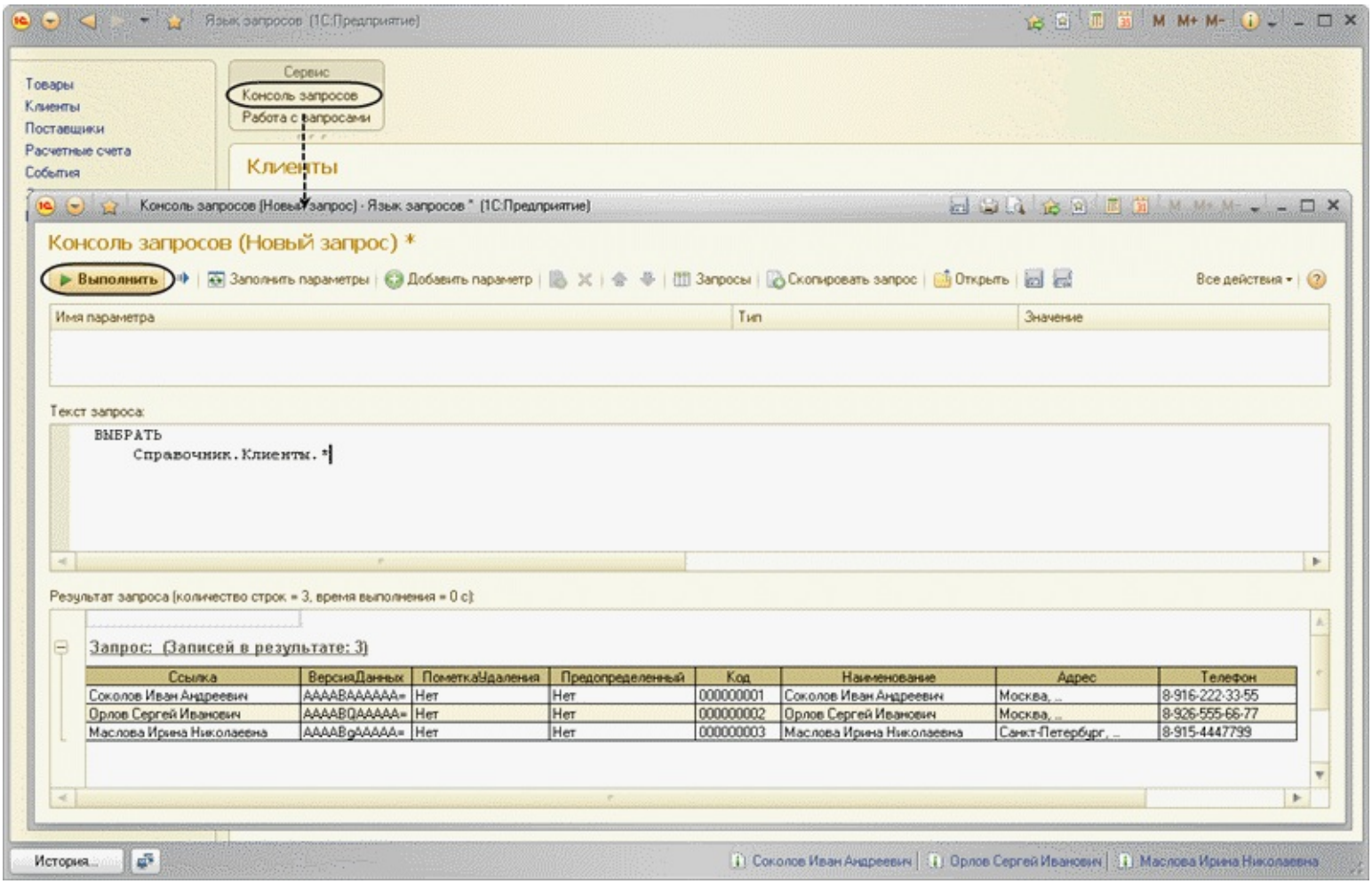


Рис. 1.15. Вывод всех данных из таблицы

После выполнения запроса в заголовке окна *Результат запроса* появится также количество строк в результате запроса и время выполнения запроса.

Теперь рассмотрим подробно текст нашего запроса (см. листинг 1.7).

Текст любого запроса всегда содержит секцию *описания запроса*, в которой определяются источники данных для запроса, список полей выборки и т. д. Все секции запроса приведены на рис. 1.13.

Описание запроса начинается с ключевого слова *ВЫБРАТЬ*. За ним следует список имен полей выборки запроса, перечисленных через запятую. Таким образом, с помощью ключевого слова *ВЫБРАТЬ* определяются требуемые поля результата запроса.

Справочник.Клиенты – это имя одной из исходных таблиц запроса, описанных выше. В языке запросов имена таблиц формируются по принципу: *<Имя класса объектов>.<Имя объекта конфигурации>*. Полное имя поля содержит имя таблицы и имя поля. Например, в строке *Справочник.Клиенты.Код Справочник.Клиенты* – это имя таблицы, а *Код* – это имя поля.

Подробнее

Посмотреть состав таблиц, доступных для запроса, и их описание можно в синтакс-помощнике в разделе *Работа с запросами > Таблицы запросов*.

В данном случае нам нужны все поля таблицы, поэтому вместо перечисления имен полей можно использовать звездочку «*».

Обратите внимание на структуру написания запроса. Правила оформления запросов предписывают все ключевые слова выделять заглавными буквами, каждое поле из списка выборки начинать с новой строки, со сдвигом относительно слова *ВЫБРАТЬ*.

Если вы напишите в одну строку – *«выбрать справочник.клиенты.*»*, то платформа вас поймет и так, и результат запроса не изменится (см. рис. 1.15). Но так писать – это моветон.

С остальными правилами оформления запросов мы будем знакомиться по ходу следующих примеров.

Итак, в данном примере мы получили все данные (стандартные и созданные разработчиком) из объектной таблицы, кроме виртуальных полей. Такие поля нельзя выбрать с помощью символа «*», их имена нужно указывать явно. Эта возможность рассматривается в следующем примере.

Как получить только определенные поля для всех записей из таблицы

В реальных задачах обычно требуется получить не все, а только некоторые конкретные поля из таблицы. В данном примере для всех записей справочника *Клиенты* получим только поля *Наименование*, *Телефон* и *Представление*.

Это можно сделать с помощью следующего запроса (листинг 1.8).

Листинг 1.8. Вывод определенных полей для всех записей из таблицы

```
ВЫБРАТЬ
  Справочник.Клиенты.Наименование,
  Справочник.Клиенты.Телефон,
  Справочник.Клиенты.Представление
```

В данном запросе после ключевого слова *ВЫБРАТЬ* перечислены полные имена требуемых полей результата запроса.

Результат выполнения запроса будет выглядеть следующим образом (рис. 1.16).

Запрос: (Записей в результате: 3)

Наименование	Телефон	Представление
Соколов Иван Андреевич	8-916-222-33-55	Соколов Иван Андреевич
Орлов Сергей Иванович	8-926-555-66-77	Орлов Сергей Иванович
Маслова Ирина Николаевна	8-915-4447799	Маслова Ирина Николаевна

Рис. 1.16. Вывод определенных полей для всех записей из таблицы

Итак, в данном примере мы получили из таблицы справочника *Клиенты* два реальных поля (*Наименование* и *Телефон*) и одно виртуальное (*Представление*).

Поле *Представление* – это текстовое представление объекта. При получении данного поля из базы данных запрос получает несколько полей, а при получении значения поля из результата запроса преобразовывает полученные значения в строку.

Мы видим, что в данном примере поле *Представление* совпадает с наименованием. Так произошло потому, что в свойствах справочника *Основное представление* установлено в значение *В виде наименования* (рис. 1.17).

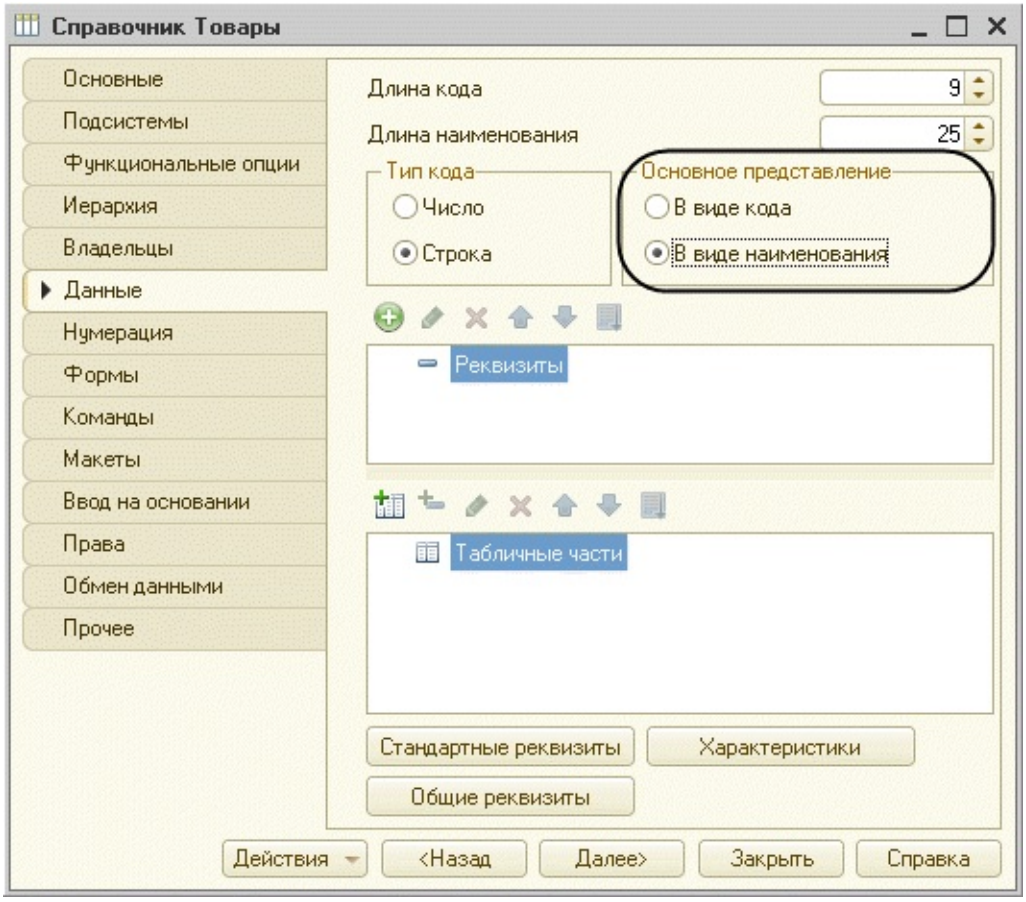


Рис. 1.17. Основное представление элементов справочника

Если мы установим свойство *Основное представление* в значение *В виде кода*, то результат выполнения запроса будет другим (рис. 1.18).

Запрос: (Записей в результате: 3)

Наименование	Телефон	Представление
Соколов Иван Андреевич	8-916-222-33-55	000000001
Орлов Сергей Иванович	8-926-555-66-77	000000002
Маслова Ирина Николаевна	8-915-4447799	000000003

Рис. 1.18. Вывод отдельных полей и всех записей из таблицы

подробнее

О поле *Представление* рассказано в разделе «Как получить текстовое представление ссылочного поля.

Итак, мы получили требуемые поля из таблицы с помощью ключевого слова *ВЫБРАТЬ*. Однако неудобно то, что для каждого поля нужно писать полный путь к нему *Справочник.Клиенты*. Имя таблицы справочника, служащей источником для запроса,

можно написать один раз после ключевого слова *ИЗ* (листинг 1.9).

Листинг 1.9. Вывод определенных полей для всех записей из таблицы

```
ВЫБРАТЬ
  Наименование,
  Телефон,
  Представление
ИЗ
  Справочник.Клиенты
```

Результат запроса будет аналогичен выполнению предыдущего запроса (см. рис. 1.16), то есть результат выполнения двух следующих запросов будет одинаковым (листинг 1.10).

Листинг 1.10. Вывод определенных полей для всех записей из таблицы

```
// Первый вариант
ВЫБРАТЬ
  Справочник.Клиенты.Наименование,
  Справочник.Клиенты.Телефон,
  Справочник.Клиенты.Представление

// Второй вариант
ВЫБРАТЬ
  Наименование,
  Телефон,
  Представление
ИЗ
  Справочник.Клиенты
```

Также с помощью ключевого слова *ИЗ* можно переписать текст запроса, используемого нами в первом примере, то есть результат выполнения двух следующих запросов будет одинаковым (листинг 1.11).

Листинг 1.11. Вывод всех полей и всех записей из таблицы

```
// Первый вариант
ВЫБРАТЬ
  Справочник.Клиенты.*

// Второй вариант
ВЫБРАТЬ
  *
ИЗ
  Справочник.Клиенты
```

Таким образом, после ключевого слова *ИЗ* через запятую перечисляются таблицы (реальные или виртуальные), являющиеся источниками для запроса.

В реальных задачах запросы могут быть сложными, и источников в них может быть несколько. Поэтому, следуя правилам оформления запросов, желательно давать

псевдонимы источникам и использовать их затем в других частях запроса. В некоторых случаях это является обязательным, например, когда из разных таблиц выбираются поля с одинаковыми именами.

Псевдонимы задаются с помощью ключевого слова *КАК*, после которого следует имя псевдонима. Имя псевдонима может писаться сразу после имени таблицы (например, *Справочник.Клиенты Клиенты*), но наличие ключевого слова *КАК* повышает наглядность и удобочитаемость текста запроса.

Таким образом, запрос, представленный в листинге 1.9, может быть изменен следующим образом (листинг 1.12).

Листинг 1.12. Вывод определенных полей для всех записей из таблицы

```
// Первый вариант без псевдонимов
ВЫБРАТЬ
  Наименование,
  Телефон,
  Представление
ИЗ
  Справочник.Клиенты

// Рекомендуемый вариант с псевдонимом таблицы-источника
ВЫБРАТЬ
  Клиенты.Наименование,
  Клиенты.Телефон,
  Клиенты.Представление
ИЗ
  Справочник.Клиенты КАК Клиенты
```

Также псевдонимы рекомендуется присваивать полям выборки. Дело в том, что на имена полей выборки могут быть завязаны алгоритмы обработки результатов запроса.

Если не присваивать псевдонимы полям выборки, то поля выборки будут иметь «стандартные» имена, соответствующие их именам в таблице запросов. В этом случае при выборе другого поля в запросе у поля выборки будет другое имя. А если использовать псевдонимы полей выборки, то имя поля выборки в результате запроса можно оставить без изменений, только оно уже будет указывать на другое поле в таблице запроса. Таким образом, при изменении полей выборки запроса алгоритм обработки результатов будет работать так же, как и раньше.

Для примера сравним две процедуры, в которых выполняется запрос и обрабатываются его результаты (листинги 1.13, 1.14).

Листинг 1.13. Вывод наименований всех товаров

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        Наименование КАК Название
  |ИЗ
```

```
  |        Справочник.Товары";

РезультатЗапроса = Запрос.Выполнить();
ВыборкаЗапроса = РезультатЗапроса.Выбрать();

Пока ВыборкаЗапроса.Следующий() Цикл
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = ВыборкаЗапроса.Название;
  Сообщение.Сообщить();

КонецЦикла;
```

Листинг 1.14. Вывод англоязычных наименований всех товаров

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        АнглоязычноеНаименование КАК Название
  |ИЗ
  |        Справочник.Товары";

РезультатЗапроса = Запрос.Выполнить();
ВыборкаЗапроса = РезультатЗапроса.Выбрать();

Пока ВыборкаЗапроса.Следующий() Цикл
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = ВыборкаЗапроса.Название;
  Сообщение.Сообщить();

КонецЦикла;
```

То есть в алгоритме обработки результатов запроса ничего не меняется, меняется только строка (или строки) в тексте запроса. И в результате в окно сообщений выводятся разные поля таблицы.

Как расположить полученные записи в нужном порядке

Следующий пример, который мы рассмотрим, показывает простую, но очень важную возможность языка запросов – возможность упорядочивания записей в результате запроса.

Важно понимать, что упорядочивание результата запроса – это обязательное правило хорошего тона при написании работоспособных запросов в «1С:Предприятии». Если не использовать упорядочивание, то порядок записей в результате запроса будет не определен. Предугадать его заранее не может ни пользователь, ни даже разработчик.

Например, один и тот же запрос, выполненный на разных СУБД, может дать разный порядок записей. С точки зрения пользователя, это может показаться странным, а с точки зрения разработчика, вообще может привести к ошибкам при обработке результатов запроса, если алгоритм обработки рассчитан на определенный порядок записей.

Поэтому в запросе очень важно всегда явно указывать, как должны быть упорядочены

записи. Даже тогда, когда, на первый взгляд, порядок записей не влияет на обработку результата. Потому что заранее неизвестно, как и кем будет дорабатываться дальше прикладное решение.

Порядок расположения записей в результате запроса определяется в секции *Упорядочивание результатов* текста запроса. Все секции запроса приведены на рис. 1.13.

Рассмотрим реальную задачу. Часто требуется выводить записи из регистра сведений *Цены* не в случайном порядке, а в порядке возрастания даты изменения цен товаров.

Для этого в языке запросов предназначено ключевое слово *УПОРЯДОЧИТЬ ПО*, после которого следует список полей упорядочивания и тип упорядочивания (по возрастанию, по убыванию, по иерархии) для каждого поля.

Например, чтобы вывести записи регистра сведений *Цены* в порядке возрастания поля *Период*, нужно выполнить следующий запрос (листинг 1.15).

Листинг 1.15. Упорядочивание записей регистра сведений по возрастанию поля «Период»

```
ВЫБРАТЬ
  Цены.Период КАК Период,
  Цены.Товар КАК Товар,
  Цены.Цена КАК Цена
ИЗ
  РегистрСведений.Цены КАК Цены

УПОРЯДОЧИТЬ ПО
  Период
```

Обратите внимание, предложение *УПОРЯДОЧИТЬ ПО* относится к секции упорядочивания результатов запроса, а предложения *ВЫБРАТЬ* и *ИЗ* – к секции описания запроса. В правилах оформления запросов принято отделять одну секцию запроса от другой, поэтому между секциями находится пустая строка.

Результат выполнения запроса будет следующим (рис. 1.19).

Запрос: РегистрСведений.Цены (Записей в результате: 10)

Период	Товар	Цена
15.10.2012 0:00:00	Туфли	7 000
15.10.2012 0:00:00	Сапоги	10 000
15.10.2012 0:00:00	Кроссовки	3 000
20.10.2012 0:00:00	Масло	70
20.10.2012 0:00:00	Молоко	40
20.10.2012 0:00:00	Сметана	50
01.11.2012 0:00:00	Туфли	8 000
01.11.2012 0:00:00	Масло	80
05.11.2012 0:00:00	Сапоги	10 500
05.11.2012 0:00:00	Молоко	45

Рис. 1.19. Вывод записей из таблицы в определенном порядке

Мы видим, что записи таблицы в результате запроса расположены по возрастанию поля, указанного в предложении *УПОРЯДОЧИТЬ ПО*, хотя тип упорядочивания не задан. Дело в том, что упорядочивание записей результата запроса по возрастанию выполняется по умолчанию, хотя можно указать явно тип упорядочивания. Для этого после имени поля нужно указать ключевое слово *ВОЗР* (листинг 1.16).

Листинг 1.16. Упорядочивание записей регистра сведений по возрастанию поля «Период»

```
ВЫБРАТЬ
  Цены.Период КАК Период,
  Цены.Товар КАК Товар,
  Цены.Цена КАК Цена
ИЗ
  РегистрСведений.Цены КАК Цены

УПОРЯДОЧИТЬ ПО
  Период ВОЗР
```

При этом поле, по которому производится упорядочивание, необязательно должно входить в список выборки запроса (листинг 1.17).

Листинг 1.17. Упорядочивание записей регистра сведений по возрастанию поля «Период»

```
ВЫБРАТЬ
  Цены.Товар КАК Товар,
  Цены.Цена КАК Цена
ИЗ
  РегистрСведений.Цены КАК Цены

УПОРЯДОЧИТЬ ПО
  Цены.Период ВОЗР
```

При выполнении данного запроса записи регистра сведений *Цены* будут расположены в порядке возрастания поля *Период*, но само это поле в результате запроса отражено не будет.

Чтобы расположить записи результата запроса по убыванию какого-либо поля, нужно в предложении упорядочивания указать после имени поля ключевое слово *УБЫВ*. Можно также упорядочивать иерархические данные по иерархии (этот пример будет рассмотрен позже).

В предложении *УПОРЯДОЧИТЬ ПО* могут участвовать несколько полей, каждое из этих полей может использовать разный способ упорядочивания (листинг 1.18). В этом случае записи результата запроса будут упорядочены сначала по первому полю, затем – по второму полю (в случае, если существует несколько записей с одинаковым значением первого поля) и т. д.

Листинг 1.18. Упорядочивание записей регистра сведений по нескольким полям

```
ВЫБРАТЬ
```

```
  Цены.Период КАК Период,
  Цены.Товар КАК Товар,
  Цены.Цена КАК Цена
ИЗ
  РегистрСведений.Цены КАК Цены

УПОРЯДОЧИТЬ ПО
  Период УБЫВ,
  Цена
```

Обратите внимание, что в тексте запроса принято не только поля в списке выборки, но и поля в списке упорядочивания располагать на разных строках со смещением относительно ключевого слова *УПОРЯДОЧИТЬ ПО*.

В результате мы видим, что записи регистра сведений расположены в порядке убывания поля *Период*, а внутри одного периода – в порядке возрастания поля *Цена* (рис. 1.20).

Запрос: РегистрСведений.Цены (Записей в результате: 10)

Период	Товар	Цена
05.11.2012 0:00:00	Молоко	45
05.11.2012 0:00:00	Сапоги	10 500
01.11.2012 0:00:00	Масло	80
01.11.2012 0:00:00	Туфли	8 000
20.10.2012 0:00:00	Молоко	40
20.10.2012 0:00:00	Сметана	50
20.10.2012 0:00:00	Масло	70
15.10.2012 0:00:00	Кроссовки	3 000
15.10.2012 0:00:00	Туфли	7 000
15.10.2012 0:00:00	Сапоги	10 000

Рис. 1.20. Вывод записей из таблицы в определенном порядке

Как упорядочить записи таблицы по ссылочному полю

В предыдущем примере мы рассматривали, как упорядочить данные результата запроса по полям примитивных типов – *Дата*, *Число* и т. п. Но как упорядочить данные по ссылочному полю?

Ведь ссылка на самом деле представляет собой некоторый произвольный уникальный идентификатор, никак не связанный с прикладной сущностью данных (набором символов), которые содержатся в этом поле. У элементов справочников – одна прикладная сущность, которая может выражаться кодом или наименованием. У документов – совсем другая прикладная сущность, которая выражается датой и номером документа и т. д.

Чтобы расположить объекты базы данных (записи ссылочных таблиц) в порядке, соответствующем их прикладной сущности, используется режим автоматического упорядочивания записей результата запроса.

В тексте запроса этот режим задается в секции *Автоупорядочивание*. Все секции запроса приведены на рис. 1.13.

С помощью автоупорядочивания можно вывести записи таблицы в наиболее естественном (ожидаемом пользователем) порядке. Для этого нужно упорядочить записи

таблицы непосредственно по ссылочному полю, а затем использовать конструкцию *АВТОУПОРЯДОЧИВАНИЕ* (листинг 1.19).

Листинг 1.19. Автоматическое упорядочивание записей таблицы

```
ВЫБРАТЬ
  ЗаказТовара.Дата,
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
  ЗаказТовара.СуммаЗаказа
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара

УПОРЯДОЧИТЬ ПО
  ЗаказТовара.Ссылка

АВТОУПОРЯДОЧИВАНИЕ
```

Мы видим, что при автоупорядочивании списка документов *ЗаказТовара* записи результата запроса расположены в порядке возрастания поля *Дата*, а в случае, если даты у документов одинаковые, записи упорядочиваются в порядке возрастания поля *Номер* (рис. 1.21).

Запрос: Документ.ЗаказТовара (Записей в результате: 7)

Дата	Номер	Клиент	СуммаЗаказа
03.10.2012 0:00:00	000000001	Соколов Иван Андреевич	25 000
10.10.2012 0:00:00	000000003	Орлов Сергей Иванович	3 000
20.10.2012 12:00:00	000000002	Соколов Иван Андреевич	40 000
22.10.2012 12:00:00	000000004	Маслова Ирина Николаевна	36 800
22.10.2012 12:00:00	000000005	Маслова Ирина Николаевна	86 000
25.10.2012 12:00:00	000000006	Соколов Иван Андреевич	30 000
25.10.2012 12:00:01	000000007	Орлов Сергей Иванович	2 700

Рис. 1.21. Вывод записей из таблицы в автоматическом порядке

Так произошло потому, что при выполнении данного запроса для каждого ссылочного поля были получены реальные поля, по которым его необходимо упорядочить (для документа – это дата и номер, для справочника – основное представление), и произведено упорядочивание по ним.

Таким образом, такого же результата можно было добиться, явно указав поля *Дата* и *Номер* при упорядочивании таблицы документа (листинг 1.20).

Листинг 1.20. Упорядочивание списка документов по полям «Дата» и «Номер»

```
ВЫБРАТЬ
  ЗаказТовара.Дата,
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
  ЗаказТовара.СуммаЗаказа
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара

УПОРЯДОЧИТЬ ПО
  ЗаказТовара.Дата,
```

```
  ЗаказТовара.Номер
```

Но в общем случае, чтобы расположить записи результата запроса в наиболее ожидаемом (естественном) порядке, рекомендуется более универсальный первый вариант – упорядочить записи таблицы непосредственно по ссылочному полю, а затем использовать конструкцию *АВТОУПОРЯДОЧИВАНИЕ*. В остальных случаях использовать автоупорядочивание записей результата запроса не рекомендуется.

Как получить текстовое представление ссылочного поля

Часто в отчетах (а также в тех местах конфигурации, где требуется вывести представление ссылочного поля в виде сообщения) бывает нужно отобразить значения полей ссылочного типа, например, вывести значение поля *Клиент* (являющееся ссылкой на справочник *Клиенты*) или поля *Товар* (являющееся ссылкой на справочник *Товары*).

При этом нужно понимать, что при выводе значения ссылочного поля для получения его представления будет выполняться дополнительный запрос к той таблице, на которую ссылается это ссылочное поле. В результате процесс вывода будет замедляться. Чтобы этого избежать, следует в запросе сразу получать текстовое представление ссылочного поля и затем уже его, а не саму ссылку, выводить в отчет или сообщение.

Например, рассмотрим простой запрос, выводящий записи регистра накопления *ОстаткиТоваров* (листинг 1.21).

Листинг 1.21. Получение ссылочных полей в запросе

```
ВЫБРАТЬ
  ОстаткиТоваров.Период,
  ОстаткиТоваров.Регистратор КАК Регистратор,
  ОстаткиТоваров.Товар КАК Товар,
  ОстаткиТоваров.Количество,
  ОстаткиТоваров.Сумма
ИЗ
  РегистрНакопления.ОстаткиТоваров КАК ОстаткиТоваров
```

Поля выборки запроса включают два ссылочных поля – *Регистратор* и *Товар*. Заменим значения этих полей их текстовыми представлениями. Это можно сделать с помощью полей *Представление* и функции *ПРЕДСТАВЛЕНИЕ()*. Рассмотрим их подробнее.

Каждая объектная таблица в информационной базе имеет виртуальное поле – *Представление*. Это текстовое представление объекта. Поскольку это поле – виртуальное, то при получении данного поля из базы данных запрос получает несколько полей, которые соответствуют прикладной сущности объекта, а при получении значения поля из результата запроса преобразовывает полученные значения в строку.

Для большинства объектов конфигурации (справочников, планов видов характеристик, планов счетов и т. п.) представление ссылок задается разработчиком в свойстве *Основное представление*. А для документов и бизнес-процессов система предоставляет

единственное неизменяемое представление ссылочных значений в виде совокупности синонима документа или бизнес-процесса, его номера и даты. Например: «Приходная накладная 000000003 от 03.11.2012 15:35:27».

В версии платформы 8.3 появилась возможность во встроенном языке задавать произвольные представления для ссылочных полей. Для этого у менеджеров справочников, документов и т. п. добавлены два обработчика: *ОбработкаПолученияПолейПредставления* и *ОбработкаПолученияПредставления*, в которых можно задать собственный алгоритм, формирующий представление.

При получении запросом поля *Представление* из базы данных будут получаться поля, которые указаны в обработчике *ОбработкаПолученияПолейПредставления*, а при получении значения поля из результата запроса они будут объединяться в строку в обработчике *ОбработкаПолученияПредставления*.

Функция *ПРЕДСТАВЛЕНИЕ()* предназначена для получения текстового представления любого значения, которое может быть получено при помощи языка запросов. В качестве параметра в функцию могут передаваться как ссылочные, так и примитивных типы данных.

Для ссылочных типов результат функции полностью аналогичен получению поля *Представление* от ссылки, переданной в качестве параметра функции. Например, *ПРЕДСТАВЛЕНИЕ(Товар)* в данном случае аналогично *Товар.Представление*, но первый вариант предпочтительнее.

Поэтому предыдущий запрос можно переписать в следующем виде (листинг 1.22).

Листинг 1.22. Получение ссылочных полей в запросе

```
// Получение самого ссылочного поля
ВЫБРАТЬ
  ОстаткиТоваров.Период,
  ОстаткиТоваров.Регистратор КАК Регистратор,
  ОстаткиТоваров.Товар КАК Товар,
  ОстаткиТоваров.Количество,
  ОстаткиТоваров.Сумма
ИЗ
  РегистрНакопления.ОстаткиТоваров КАК ОстаткиТоваров

// Получение текстового представления ссылочного поля
ВЫБРАТЬ
  ОстаткиТоваров.Период,
  ПРЕДСТАВЛЕНИЕ(ОстаткиТоваров.Регистратор) КАК Регистратор,
  ОстаткиТоваров.Товар.Представление КАК Товар,
  ОстаткиТоваров.Количество,
  ОстаткиТоваров.Сумма
ИЗ
  РегистрНакопления.ОстаткиТоваров КАК ОстаткиТоваров
```

В первом варианте запроса из базы данных будут получены сами ссылочные значения, а

во втором варианте – их текстовые представления. В обоих случаях результат выполнения запросов будет одинаковым, но в случае, если нужно только показать результат запроса пользователю, рекомендуется использовать второй вариант. Если ссылочные значения нужны не для показа пользователю, а для того, чтобы, например, поместить их в ячейку расшифровки отчета, то следует получать в одном запросе как сами ссылочные значения, так и их текстовые представления.

Практически в каждом примере мы выводим в консоли запросов значения ссылочных полей. В них мы не используем представления этих полей потому, что консоль запросов сама занимается выводом данных, а также для упрощения текста запросов.

Особенностью функции *ПРЕДСТАВЛЕНИЕ()* является то, что ее результат не может быть использован в выражении языка запросов, например, следующие два запроса ошибочны (листинги 1.23, 1.24).

Функцию *ПРЕДСТАВЛЕНИЕ()* нельзя использовать в условиях сравнения в предложении *ГДЕ* (листинг 1.23).

Листинг 1.23. Ошибочный запрос

```
ВЫБРАТЬ
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
  ЗаказТовара.СуммаЗаказа,
  ЗаказТовара.Состав
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
ГДЕ
  ПРЕДСТАВЛЕНИЕ(ЗаказТовара.Состав.Товар) = "Туфли".
```

Функцию *ПРЕДСТАВЛЕНИЕ()* нельзя использовать в различных выражениях в списке полей выборки, например, нельзя складывать в одну строку текстовые представления ссылочных полей (листинг 1.24).

Листинг 1.24. Ошибочный запрос

```
ВЫБРАТЬ
  ПРЕДСТАВЛЕНИЕ(ЗаказТовара.Ссылка) + ПРЕДСТАВЛЕНИЕ(ЗаказТовара.Клиент),
  ЗаказТовара.СуммаЗаказа,
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

Аналогично с полем *Представление* также нельзя выполнять никакие операции.

Как получить только первые несколько записей с наибольшими значениями некоторого поля

Часто требуется получить из большой таблицы только определенное количество записей с наибольшим или наименьшим значением какого-то поля. Например, нам нужно получить три заказа клиентов с наибольшей суммой заказа.

В нашей демонстрационной конфигурации существует список документов *ЗаказТовара*, который показывает сумму заказа по клиентам (см. ниже рис. 1.22).

Чтобы выбрать из таблицы три заказа с наибольшей суммой, зададим упорядочивание по убыванию поля *СуммаЗаказа* и затем воспользуемся конструкцией *ВЫБРАТЬ ПЕРВЫЕ 3* (листинг 1.25).

Листинг 1.25. Вывод трех документов «ЗаказТовара» с наибольшей суммой заказа

```
ВЫБРАТЬ ПЕРВЫЕ 3
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
  ЗаказТовара.СуммаЗаказа КАК СуммаЗаказа
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара

УПОРЯДОЧИТЬ ПО
  СуммаЗаказа УБЫВ
```

Конструкция *ВЫБРАТЬ ПЕРВЫЕ <Количество>* позволяет задать предельное количество строк в результате запроса.

В результате выполнения запроса на рис. 1.22 в нижней таблице мы видим три наибольших заказа в порядке убывания их суммы, а в верхней таблице для сравнения приведен список всех заказов клиентов.

Все заказы клиентов

Запрос: Документ.ЗаказТовара (Записей в результате: 7)

Номер	Клиент	СуммаЗаказа
000000001	Соколов Иван Андреевич	25 000
000000002	Соколов Иван Андреевич	40 000
000000003	Орлов Сергей Иванович	3 000
000000004	Маслова Ирина Николаевна	36 800
000000005	Маслова Ирина Николаевна	86 000
000000006	Соколов Иван Андреевич	30 000
000000007	Орлов Сергей Иванович	2 700

Три заказа клиентов с максимальной суммой заказа

Запрос: Документ.ЗаказТовара (Записей в результате: 3)

Номер	Клиент	СуммаЗаказа
000000005	Маслова Ирина Николаевна	86 000
000000002	Соколов Иван Андреевич	40 000
000000004	Маслова Ирина Николаевна	36 800

Рис. 1.22. Вывод трех документов «ЗаказТовара» с наибольшей суммой заказа

Чтобы выбрать три заказа с наименьшей суммой заказа, нужно в этом же примере поменять порядок сортировки по полю *СуммаЗаказа*.

Как получить записи, в которых определенные поля не содержат одинаковых значений
Часто записи в таблицах имеют одинаковые значения какого-то поля (полей), а нужно

получить только записи с неповторяющимися значениями этого поля.

Например, в списке документов *ЗаказТовара* повторяются значения поля *Клиент* (см. ниже рис. 1.23), то есть существуют несколько заказов для одного и того же клиента, а нас интересует, какими клиентами были сделаны заказы вообще. Для этого следует использовать конструкцию *ВЫБРАТЬ РАЗЛИЧНЫЕ* (листинг 1.26).

Листинг 1.26. Вывод различных клиентов из документов «ЗаказТовара»

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
  ЗаказТовара.Клиент
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

В результате выполнения запроса на рис. 1.23 справа мы видим трех различных клиентов, сделавших заказ товара, а слева для сравнения приведен список всех заказов клиентов.

Все клиенты

Запрос: Документ.ЗаказТовара (Записей в результате: 7)

Номер	Клиент	СуммаЗаказа
000000001	Соколов Иван Андреевич	25 000
000000002	Соколов Иван Андреевич	40 000
000000003	Орлов Сергей Иванович	3 000
000000004	Маслова Ирина Николаевна	36 800
000000005	Маслова Ирина Николаевна	86 000
000000006	Соколов Иван Андреевич	30 000
000000007	Орлов Сергей Иванович	2 700

Неповторящиеся клиенты

Запрос: Документ.ЗаказТовара (Записей в результате: 3)

Клиент
Соколов Иван Андреевич
Орлов Сергей Иванович
Маслова Ирина Николаевна

Рис. 1.23. Вывод различных клиентов из документов «ЗаказТовара»

Если в списке выборки указано несколько полей, то при использовании конструкции *ВЫБРАТЬ РАЗЛИЧНЫЕ* в результат запроса отбираются записи, содержащие неповторяющиеся комбинации значений сразу по нескольким полям (листинг 1.27).

Листинг 1.27. Вывод различных клиентов за различные даты из документов «ЗаказТовара»

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
  ЗаказТовара.Дата,
  ЗаказТовара.Клиент
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

В результате выполнения запроса на рис. 1.24 в нижней таблице мы видим список различных клиентов, сделавших заказ товара за различные даты, а в верхней таблице для сравнения приведен список всех заказов клиентов.

Все записи

Запрос: Документ.ЗаказТовара (Записей в результате: 7)

Номер	Дата	Клиент	СуммаЗаказа
000000001	03.10.2012 0:00:00	Соколов Иван Андреевич	25 000
000000002	20.10.2012 12:00:00	Соколов Иван Андреевич	40 000
000000003	10.10.2012 0:00:00	Орлов Сергей Иванович	3 000
000000004	22.10.2012 12:00:00	Маслова Ирина Николаевна	36 800
000000005	22.10.2012 12:00:00	Маслова Ирина Николаевна	86 000
000000006	25.10.2012 12:00:00	Соколов Иван Андреевич	30 000
000000007	25.10.2012 12:00:01	Орлов Сергей Иванович	2 700

Неповторяющиеся записи

Запрос: Документ.ЗаказТовара (Записей в результате: 6)

Дата	Клиент
03.10.2012 0:00:00	Соколов Иван Андреевич
10.10.2012 0:00:00	Орлов Сергей Иванович
20.10.2012 12:00:00	Соколов Иван Андреевич
22.10.2012 12:00:00	Маслова Ирина Николаевна
25.10.2012 12:00:00	Соколов Иван Андреевич
25.10.2012 12:00:01	Орлов Сергей Иванович

Рис. 1.24. Вывод различных клиентов за различные даты из документов «ЗаказТовара»

На рис. 1.24 в нижней таблице мы видим 6 записей, а всего в списке заказов клиентов содержится 7 документов. Так произошло потому, что в верхней таблице у одного из клиентов сделано два заказа за одну дату.

Необходимо помнить, что если в запросе указано ключевое слово *РАЗЛИЧНЫЕ* и в предложении *УПОРЯДОЧИТЬ ПО* указано поле, отсутствующее в списке выборки, то при выполнении такого запроса будет выдана ошибка. Например, следующий запрос вызовет ошибку (листинг 1.28).

Листинг 1.28. Ошибочный запрос

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
  ЗаказТовара.Дата,
  ЗаказТовара.Клиент
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
УПОРЯДОЧИТЬ ПО
  ЗаказТовара.Номер
```

Как получить общее количество записей в таблице и количество записей с различным значением некоторого поля

Этот пример является продолжением предыдущего примера, но в данном случае нам нужно получить не сами записи, а количество записей в результате запроса. Это делается с помощью агрегатной функции *КОЛИЧЕСТВО()*.

Агрегатные функции предназначены для обобщения значений параметра, переданного в функцию. Функция *КОЛИЧЕСТВО()* подсчитывает количество значений поля, указанного в параметре.

При выполнении следующего запроса выводится общее количество записей в таблице заказов товара клиентами и количество записей, содержащих различные значения в поле *Клиент* (листинг 1.29).

Листинг 1.29. Вывод общего количества записей в таблице и количества различных записей по полю «Клиент»

```
ВЫБРАТЬ
  КОЛИЧЕСТВО(*) КАК Всего,
  КОЛИЧЕСТВО(РАЗЛИЧНЫЕ ЗаказТовара.Клиент) КАК РазныеКлиенты
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

В данном запросе во второй строке запроса подсчитывается общее количество записей в результате запроса (*КОЛИЧЕСТВО(*)*). В следующей строке подсчитывается количество различных значений, отличных от *NULL*, для поля *Клиент*.

В этом примере показывается, что в списке полей выборки, после ключевого слова *ВЫБРАТЬ*, можно использовать не только поля таблиц запроса, но и различные выражения с их использованием, то есть агрегатные функции, функции языка запросов и т. п.

подробнее

Об использовании выражений в списке полей выборки будет рассказано в разделе «Примеры использования выражений в списке выборки».

О том, что такое значение *NULL*, будет рассказано в разделах " Как получить записи иерархической таблицы и расположить их в порядке иерархии" и " Левое внешнее соединение".

Результат выполнения запроса выводит соответственно количество записей при выполнении запросов из предыдущего примера (рис. 1.25).

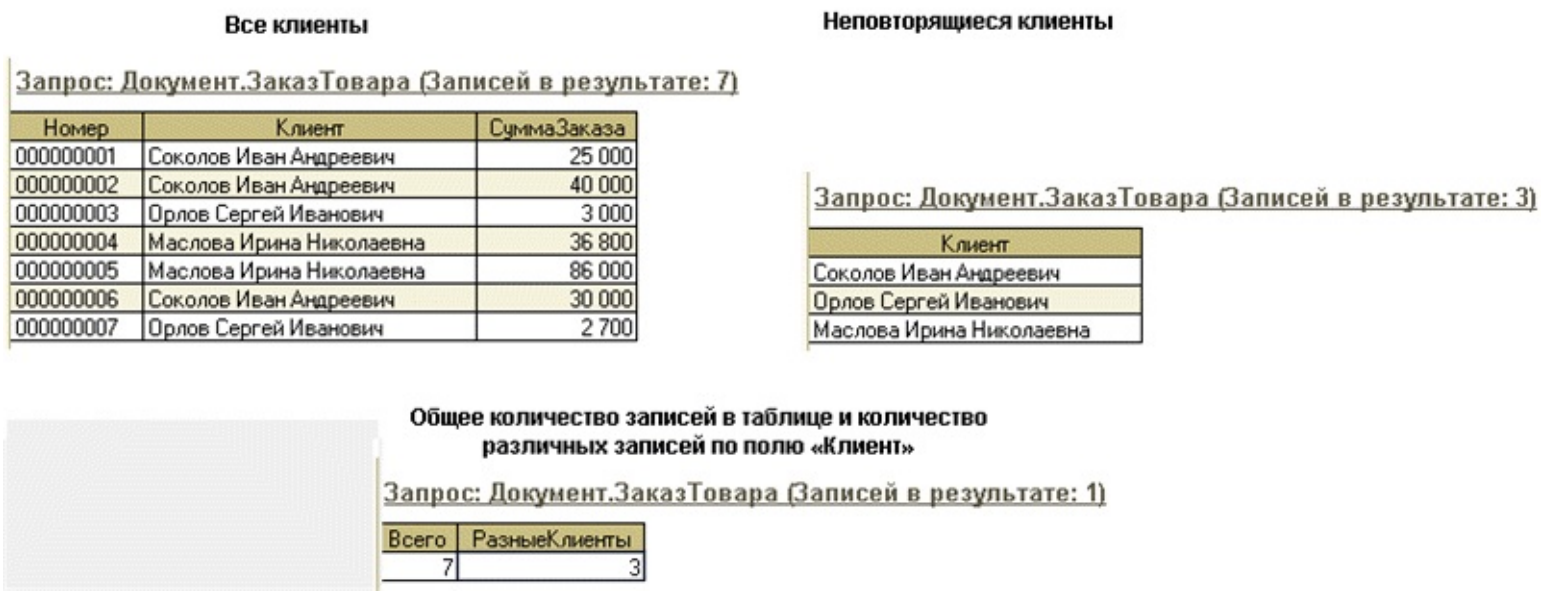


Рис. 1.25. Вывод общего количества записей в таблице и количества различных записей по полю «Клиент»

Если требуется узнать количество всевозможных значений, отличных от *NULL*, для некоторого поля в таблице, то можно просто написать – *КОЛИЧЕСТВО(<Имя поля>)*.

Как получить записи из таблицы, отобранные по некоторому условию

Теперь рассмотрим распространенную задачу, когда из большой таблицы нужно получить только некоторые записи, удовлетворяющие заданному условию.

В нашей демонстрационной конфигурации существует документ *ПриходнаяНакладная*. Полностью список накладных представлен далее, на рис. 1.26, в верхней таблице. Предположим, нам нужно отобрать из списка документов только накладные за текущий месяц.

Условие отбора данных из таблицы задается после ключевого слова *ГДЕ*. В результат запроса будут включены только те записи, которые удовлетворяют условию отбора.

Для того чтобы отобрать документы за текущий месяц, в условии отбора сравним поле документа *Дата* с литералом даты *ДАТАВРЕМЯ(2012, 11, 01)*, листинг 1.30.

Листинг 1.30. Вывод документов «ПриходнаяНакладная» за текущий месяц

```
ВЫБРАТЬ
  Накладная.Дата КАК Дата,
  Накладная.Номер КАК Номер,
  Накладная.Поставщик
ИЗ
  Документ.ПриходнаяНакладная КАК Накладная
ГДЕ
  Дата >= ДАТАВРЕМЯ(2012, 11, 01)
УПОРЯДОЧИТЬ ПО
  Дата, Номер
```

Литералы – это строковые константы, которые могут использоваться в тексте запроса. Литералы могут быть различных примитивных типов – *Булево* (*ИСТИНА*, *ЛОЖЬ*), *Число* (например, 222.77), *Строка* (например, «Мария»). Использование этих литералов мы рассмотрим в следующих примерах.

В данном запросе в условии отбора мы используем литерал типа *Дата*, значения которого задаются с помощью ключевого слова *ДАТАВРЕМЯ*, после которого в скобках последовательно указываются год, месяц, день, час, минута, секунда. Последние три значения указывать необязательно.

В результате выполнения запроса на рис. 1.26 в нижней таблице мы видим список накладных за текущий месяц, а в верхней таблице для сравнения приведен список всех приходных накладных.

Все накладные

Запрос: Документ.ПриходнаяНакладная (Записей в результате: 4)

Дата	Номер	Поставщик
17.10.2012 12:00:00	000000001	Скороход АО
25.10.2012 12:00:00	000000004	Животноводство ООО
02.11.2012 12:00:00	000000002	Животноводство ООО
05.11.2012 12:00:00	000000003	Корнет ЗАО

Накладные за текущий месяц

Запрос: Документ.ПриходнаяНакладная (Записей в результате: 2)

Дата	Номер	Поставщик
02.11.2012 12:00:00	000000002	Животноводство ООО
05.11.2012 12:00:00	000000003	Корнет ЗАО

Рис. 1.26. Отбор документов «ПриходнаяНакладная» по условию

В предложении *ГДЕ* можно использовать как имена полей, так и их псевдонимы. При этом совершенно необязательно, чтобы поле, фигурирующее в предложении *ГДЕ*, входило в список выборки (листинг 1.31).

Листинг 1.31. Вывод документов «ПриходнаяНакладная» за текущий месяц

```
ВЫБРАТЬ
  Накладная.Номер,
  Накладная.Поставщик
ИЗ
  Документ.ПриходнаяНакладная КАК Накладная
ГДЕ
  Накладная.Дата >= ДАТАВРЕМЯ(2012, 11, 01)
```

Условие отбора может определяться и как простое логическое выражение, и как более сложное, в котором простые логические выражения соединяются между собой логическими операторами *И*, *ИЛИ*, *НЕ* (листинг 1.32).

Листинг 1.32. Вывод документов «ПриходнаяНакладная» за прошлый месяц по сложному условию

```
ВЫБРАТЬ
  Накладная.Дата КАК Дата,
  Накладная.Номер,
  Накладная.Поставщик
ИЗ
  Документ.ПриходнаяНакладная КАК Накладная
ГДЕ
  Дата >= ДАТАВРЕМЯ(2012, 10, 01)
  И Дата < ДАТАВРЕМЯ(2012, 11, 01)
```

В условиях сначала вычисляются простые логические выражения, затем операции *НЕ*, затем операции *И*, в последнюю очередь – операции *ИЛИ*. Для того чтобы обеспечить другой порядок вычислений, можно использовать круглые скобки.

Подробнее

Документация «1С:Предприятие 8.3. Руководство разработчика», раздел 8.2.17 «Условия в языке запросов».

В данном случае (листинг 1.32) мы получили накладные за прошлый месяц с помощью сложного условия, где простые логические выражения соединены логическим союзом *И*. Но это сделано просто для примера. Такой же результат можно получить, если в условии отбора использовать оператор *МЕЖДУ*, который проверяет результат вхождения значения в диапазон, вместе с границами диапазона (листинг 1.33).

Листинг 1.33. Вывод документов «ПриходнаяНакладная» за прошлый месяц

```
// Первый вариант
ВЫБРАТЬ
  Накладная.Дата КАК Дата,
  Накладная.Номер,
  Накладная.Поставщик
ИЗ
  Документ.ПриходнаяНакладная КАК Накладная
ГДЕ
  Дата >= ДАТАВРЕМЯ(2012, 10, 01)
  И Дата < ДАТАВРЕМЯ(2012, 11, 01)

// Второй вариант
ВЫБРАТЬ
  Накладная.Дата КАК Дата,
  Накладная.Номер,
  Накладная.Поставщик
ИЗ
  Документ.ПриходнаяНакладная КАК Накладная
ГДЕ
  Дата МЕЖДУ ДАТАВРЕМЯ(2012, 10, 01) И ДАТАВРЕМЯ(2012, 10, 31)
```

Как получить записи таблицы, содержащие строки, соответствующие заданному шаблону

Этот пример является продолжением предыдущего примера, но в данном случае рассмотрим, как с помощью условия отбора в предложении *ГДЕ* получить записи таблицы, содержащие строки, соответствующие заданному шаблону.

В нашей демонстрационной конфигурации существует справочник *Клиенты*. Полностью список всех элементов справочника представлен далее, на рис. 1.27, в верхней таблице. Предположим, нам нужно отобрать из справочника тех клиентов, в наименовании которых встречается подстрока «Иван».

Для этого в тексте запроса после ключевого слова *ГДЕ* зададим условие отбора, в котором сравним значение поля *Наименование* со строкой шаблона «%Иван%» при помощи оператора *ПОДОБНО* (листинг 1.34).

Листинг 1.34. Отбор записей из справочника «Клиенты» по условию

```
ВЫБРАТЬ
  Клиенты.Наименование КАК Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон
```

```
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  Наименование ПОДОБНО "%Иван%"
```

Результатом выполнения оператора *ПОДОБНО* будет *Истина* или *Ложь* в зависимости от того, удовлетворяет шаблону значение выражения (в данном случае поля *Наименование*) или нет.

В условии отбора мы используем литерал строкового типа (*"%Иван%"*), который представляет собой набор символов, заключенных в кавычки. Символ «%» (процент) заменяет в шаблоне строки любую последовательность символов.

В результате выполнения запроса на рис. 1.27 в нижней таблице мы видим только тех клиентов, часть наименования которых совпадает с подстрокой «Иван», а в верхней таблице для сравнения приведен список всех элементов справочника *Клиенты*.

Все клиенты

Запрос: Справочник.Клиенты (Записей в результате: 3)

Наименование	Адрес	Телефон
Соколов Иван Андреевич	Москва, ...	8-916-222-33-55
Орлов Сергей Иванович	Москва, ...	8-926-555-66-77
Маслова Ирина Николаевна	Санкт-Петербург, ...	8-915-4447799

Клиенты, в наименовании которых встречается подстрока "Иван"

Запрос: Справочник.Клиенты (Записей в результате: 2)

Наименование	Адрес	Телефон
Соколов Иван Андреевич	Москва, ...	8-916-222-33-55
Орлов Сергей Иванович	Москва, ...	8-926-555-66-77

Рис. 1.27. Отбор записей из справочника «Клиенты» по условию

Теперь немного изменим условие, чтобы узнать, телефоны каких клиентов не соответствуют шаблону «_-___-___-__-__». В заданном шаблоне символ «-» (тире) означает наличие символа тире в строке. А символ «_» (подчеркивание) – это служебный символ, используемый в шаблонах. Он заменяет в шаблоне строки один произвольный символ. Например, строка с номером телефона, соответствующего шаблону, может иметь вид «8-916-222-33-55».

подробнее

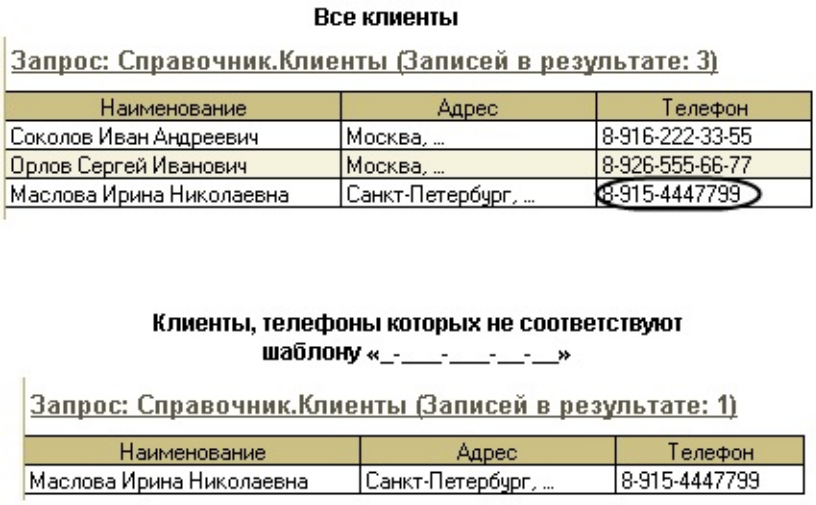
Встроенная справка: *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Синтаксис текста запросов > Использование выражений в языке запросов > Логические выражения > Подобно*.

Поставленная задача может быть решена с помощью следующего запроса (листинг 1.35).

Листинг 1.35. Отбор записей из справочника «Клиенты» по условию

```
ВЫБРАТЬ
  Клиенты.Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон КАК Телефон
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  НЕ Телефон ПОДОБНО "_-___-___-__-__"
```

В результате выполнения запроса на рис. 1.28 в нижней таблице мы видим только тех клиентов, телефоны которых не соответствуют шаблону «_-___-___-__-__», а в верхней таблице для сравнения приведен список всех элементов справочника *Клиенты*.



Все клиенты

Запрос: Справочник.Клиенты (Записей в результате: 3)

Наименование	Адрес	Телефон
Соколов Иван Андреевич	Москва, ...	8-916-222-33-55
Орлов Сергей Иванович	Москва, ...	8-926-555-66-77
Маслова Ирина Николаевна	Санкт-Петербург, ...	8-915-4447799

Клиенты, телефоны которых не соответствуют шаблону «_-___-___-__-__»

Запрос: Справочник.Клиенты (Записей в результате: 1)

Наименование	Адрес	Телефон
Маслова Ирина Николаевна	Санкт-Петербург, ...	8-915-4447799

Рис. 1.28. Отбор записей из справочника «Клиенты» по условию

Таким образом, с помощью оператора *НЕ … ПОДОБНО* можно найти те строки, которые не соответствуют некоторому шаблону, например, узнать, какие телефоны были введены в базу данных неправильно.

Как задать произвольное значение отбора записей из таблицы

Этот пример является продолжением предыдущего примера, но в данном случае мы хотим показать, как задавать произвольные значения отбора записей из таблицы.

Например, пользователь в режиме *1С:Предприятие* может задать значение условия отбора при формировании отчета, при просмотре списка справочника и т. д. Запросу, формирующему отчет или список, заранее не известно значение отбора, которое введет пользователь. То есть в данном случае требуется передавать в запрос произвольное значение отбора, а не задавать его жестко в тексте запроса.

Для этого в условии отбора, в предложении *ГДЕ* языка запросов, можно использовать параметры. В тексте запроса параметры обозначаются символом «&», после которого следует имя параметра (например, параметр *&Клиент*).

Сначала для простоты научимся использовать параметры запроса в консоли запросов.

Откроем консоль и в окно *Текст запроса* введем текст запроса из примера, в котором мы отбирали из справочника тех клиентов, в наименовании которых встречается определенная подстрока. При помощи оператора *ПОДОБНО* мы сравнивали значение поля *Наименование* со строкой шаблона «%Иван%» (листинг 1.36).

Листинг 1.36. Отбор записей из справочника «Клиенты» по условию

```
ВЫБРАТЬ
  Клиенты.Наименование КАК Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  Наименование ПОДОБНО "%Иван%"
```

Изменим в данном запросе условие отбора следующим образом (листинг 1.37).

Листинг 1.37. Отбор записей из справочника «Клиенты» по параметризированному условию

```
ВЫБРАТЬ
  Клиенты.Наименование КАК Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  Наименование ПОДОБНО "%" + &ЧастьНаименования + "%"
```

В этом запросе мы используем параметр *ЧастьНаименования*. Символ «%» заменяет в шаблоне строки любую последовательность символов, а вместо строки «Иван» в исходном запросе (см. листинг 1.36) мы используем значение параметра *&ЧастьНаименования*.

В консоли запросов нажмем кнопку *Заполнить параметры*, и параметр *ЧастьНаименования* типа *Строка* будет добавлен в окно параметров консоли запросов. В поле *Значение* мы можем задавать произвольные значения параметра и смотреть на результат выполнения запроса.

Так, если мы зададим значение параметра равным строке «Иван», в результате выполнения запроса увидим только тех клиентов, часть наименования которых совпадает с подстрокой «Иван» (рис. 1.29).

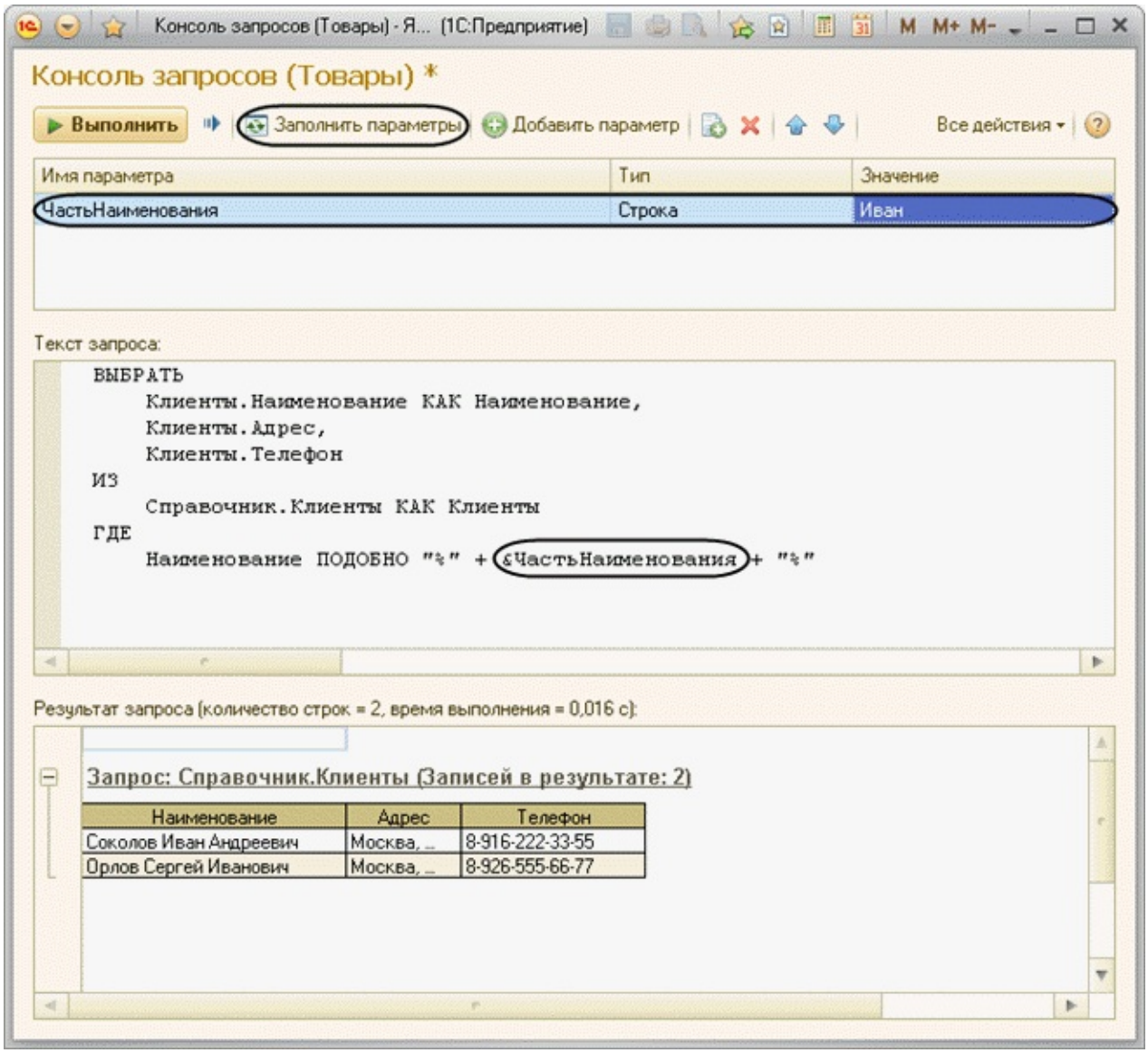


Рис. 1.29. Выполнение параметризированного запроса в консоли запросов

То есть результат запроса полностью совпадает с результатом при выполнении запроса, не использующего параметры в условии отбора (см. рис. 1.27). Но очевидно, что использование параметров делает запрос более гибким, так как значение отбора не записывается жестко в тексте запроса, а может быть задано пользователем в процессе работы и передано в запрос в качестве параметра.

Помимо того, что нам не понадобилось создавать специальный интерфейс для ввода значения параметра, в консоли запросов был скрыт от нас еще один важный момент. А именно то, что при выполнении запроса значения параметров должны быть переданы в запрос.

подробнее

Передача параметров в запрос с помощью встроенного языка рассматривается в разделе «Передача параметров в запрос».

Аналогично (с использованием параметров) мы можем переписать запрос для отбора из справочника тех клиентов, телефоны которых не соответствуют заданному шаблону (листинг 1.38).

Листинг 1.38. Отбор записей справочника «Клиенты» по условию

```
// Условие без параметров
ВЫБРАТЬ
  Клиенты.Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон КАК Телефон
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  НЕ Телефон ПОДОБНО "_-___-___-__-__"

// Параметризированное условие
ВЫБРАТЬ
  Клиенты.Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон КАК Телефон
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  НЕ Телефон ПОДОБНО &ШаблонТелефона
```

Нажмем кнопку *Заполнить параметры* и зададим значение параметра *ШаблонТелефона* как «_-___-___-__-__». Результат запроса (рис. 1.30) полностью совпадает с результатом при выполнении запроса, не использующего параметры в условии отбора (см. рис. 1.28).

Запрос: Справочник.Клиенты (Записей в результате: 1)

Наименование	Адрес	Телефон
Маслова Ирина Николаевна	Санкт-Петербург, ...	8-915-4447799

Рис. 1.30. Отбор записей справочника «Клиенты» по параметризированному условию

Как получить данные из табличной части некоторого документа

Часто может понадобиться вывести все данные из табличной части определенного документа или элемента справочника или другого объекта конфигурации, имеющего табличную часть.

Для этого в языке запросов существует удобная возможность – обращаться к табличной части как к отдельной таблице. При этом в тексте запроса можно использовать все ранее изученные нами конструкции: *ВЫБРАТЬ*, *ИЗ* и т. п. Синтаксис обращения к таблице-источнику включает также имя табличной части – *<Имя класса объектов>.<Имя объекта конфигурации>.<Имя табличной части>*, например *Документ.ЗаказТовара.Состав*.

Следующий запрос выводит данные всех табличных частей из всех документов *ЗаказТовара* (листинг 1.39).

Листинг 1.39. Вывод данных всех табличных частей из всех документов «ЗаказТовара»

```
ВЫБРАТЬ
  СоставЗаказа.Товар,
  СоставЗаказа.Количество,
  СоставЗаказа.Сумма
```

```
ИЗ
  Документ.ЗаказТовара.Состав КАК СоставЗаказа
```

В начале главы "Как хранятся данные в «1С:Предприятии»" мы рассматривали, как хранятся данные табличных частей в информационной базе «1С:Предприятия». Поля выборки запроса *Товар*, *Количество*, *Сумма* – это реквизиты табличной части документа, находящиеся в отдельной таблице базы данных, к которой мы обращаемся запросом. В этой таблице хранятся данные всех табличных частей документов определенного вида.

Результат выполнения запроса представлен на рис. 1.31.

Запрос: Документ.ЗаказТовара.Состав (Записей в результате: 16)

Товар	Количество	Сумма
Кроссовки	5	10 000
Туфли	3	15 000
Сапоги	5	25 000
Туфли	3	15 000
Масло	10	800
Сметана	20	1 000
Молоко	30	1 200
Масло	10	800
Сметана	20	1 000
Сапоги	5	35 000
Масло	15	1 000
Туфли	10	50 000
Сапоги	5	35 000
Кроссовки	10	30 000
Масло	20	1 200
Сметана	20	1 500

Рис. 1.31. Вывод данных всех табличных частей из всех документов «ЗаказТовара»

Однако в таком виде мы не можем сказать, к какому клиенту и к какому заказу относится каждая из записей результата запроса. Для повышения информативности данных мы можем добавить в список полей выборки реквизиты документа *Номер* и *Клиент*.

Мы знаем, что поля *Номер* и *Клиент* хранятся в основной таблице документа, а поля *Товар*, *Количество*, *Сумма* из табличной части документа – в подчиненной таблице, связанной с основной таблицей по полю *Ссылка*.

Поскольку в нашем примере источником запроса является подчиненная таблица документа *Документ.ЗаказТовара.Состав*, то, чтобы получить данные из основной таблицы документа, нужно обращаться к полям таблицы через точку от поля табличной части *Ссылка* (листинг 1.40).

Листинг 1.40. Вывод данных из основной и подчиненной таблицы документов «ЗаказТовара»

```
ВЫБРАТЬ
  СоставЗаказа.Ссылка.Номер,
  СоставЗаказа.Ссылка.Клиент,
  СоставЗаказа.Товар,
  СоставЗаказа.Количество,
  СоставЗаказа.Сумма
ИЗ
  Документ.ЗаказТовара.Состав КАК СоставЗаказа
```

Результат выполнения запроса представлен на рис. 1.32.

Запрос: Документ.ЗаказТовара.Состав (Записей в результате: 16)

Номер	Клиент	Товар	Количество	Сумма
000000001	Соколов Иван Андреевич	Кроссовки	5	10 000
000000001	Соколов Иван Андреевич	Туфли	3	15 000
000000002	Соколов Иван Андреевич	Сапоги	5	25 000
000000002	Соколов Иван Андреевич	Туфли	3	15 000
000000003	Орлов Сергей Иванович	Масло	10	800
000000003	Орлов Сергей Иванович	Сметана	20	1 000
000000003	Орлов Сергей Иванович	Молоко	30	1 200
000000004	Маслова Ирина Николаевна	Масло	10	800
000000004	Маслова Ирина Николаевна	Сметана	20	1 000
000000004	Маслова Ирина Николаевна	Сапоги	5	35 000
000000005	Маслова Ирина Николаевна	Масло	15	1 000
000000005	Маслова Ирина Николаевна	Туфли	10	50 000
000000005	Маслова Ирина Николаевна	Сапоги	5	35 000
000000006	Соколов Иван Андреевич	Кроссовки	10	30 000
000000007	Орлов Сергей Иванович	Масло	20	1 200
000000007	Орлов Сергей Иванович	Сметана	20	1 500

Рис. 1.32. Вывод данных из основной и подчиненной таблицы документов «ЗаказТовара»

Конечно, когда данных много, обычно требуется отобрать информацию из определенного документа. Для этого нужно добавить в запрос параметризированное условие, содержащее параметр, имеющий тип ссылки на документ *ЗаказТовара* (листинг 1.41).

Листинг 1.41. Вывод данных из определенного документа

```
ВЫБРАТЬ
  СоставЗаказа.Ссылка.Номер,
  СоставЗаказа.Ссылка.Клиент,
  СоставЗаказа.Товар,
  СоставЗаказа.Количество,
  СоставЗаказа.Сумма
ИЗ
  Документ.ЗаказТовара.Состав КАК СоставЗаказа
ГДЕ
  СоставЗаказа.Ссылка = &Документ
```

Нажмем кнопку *Заполнить параметры*, и параметр *Документ* будет добавлен в окно параметров консоли запросов. Причем консоль запросов из текста запроса автоматически определяет не только имя, но и тип параметра. В предыдущем примере (см. рис. 1.29) мы использовали в условии отбора параметр примитивного типа (*Строка*). В данном случае параметр запроса имеет ссылочный тип – *ДокументСсылка.ЗаказТовара*.

В результате выполнения запроса мы увидим только те записи, которые относятся к конкретному заказу товаров, указанному в параметре *Документ* (рис. 1.33).

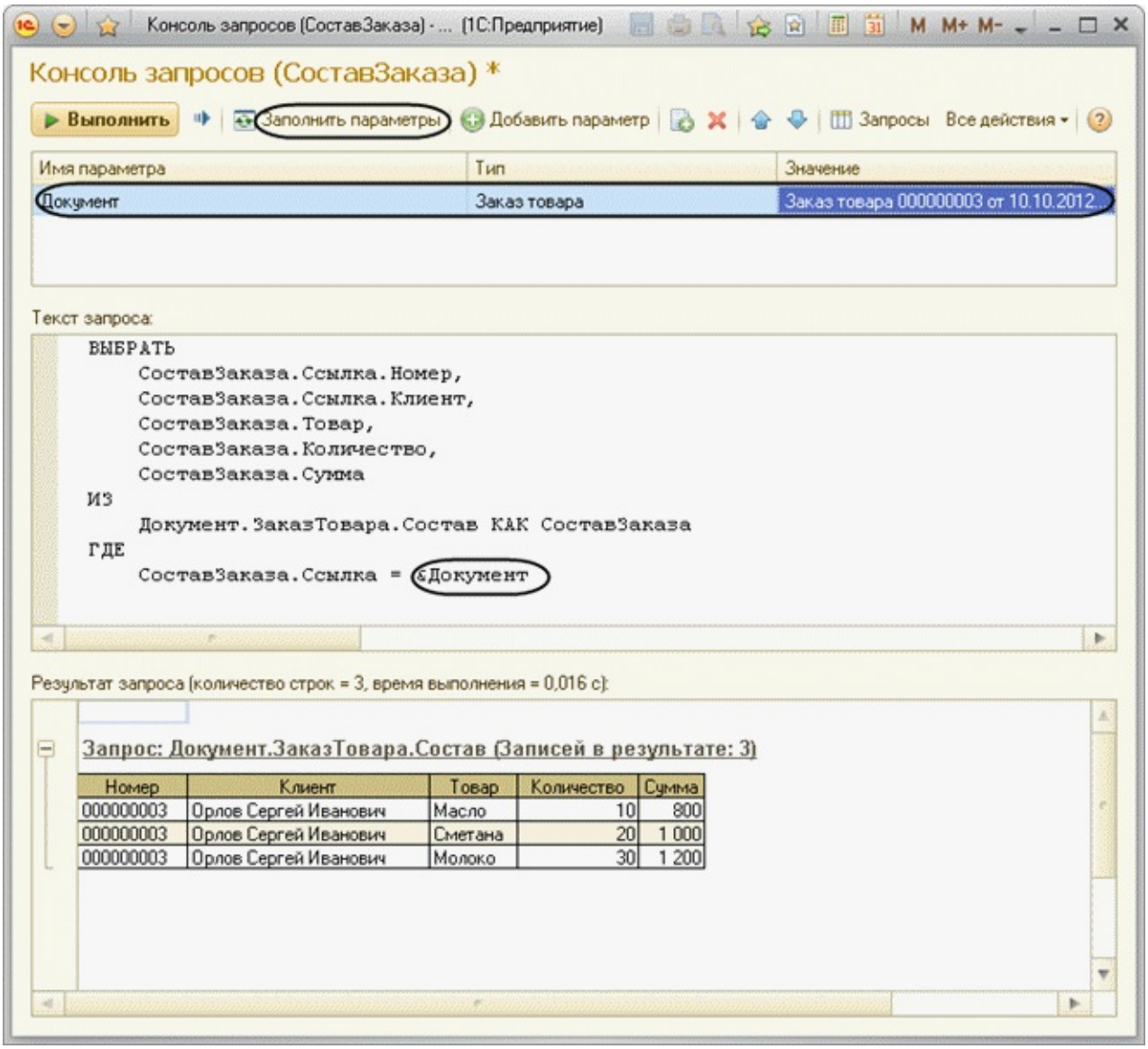


Рис. 1.33. Вывод данных из определенного документа

На самом деле данные для этого запроса выбирались из двух взаимосвязанных таблиц (основной и подчиненной таблиц документа, связанных по полю *Ссылка*), но подробнее мы остановимся на этом позже.

подробнее

Раздел «Как получить данные из таблицы, на которую ссылается поле другой таблицы».

Как получить данные из табличной части документа в качестве вложенной таблицы

Информацию из табличной части документа можно получить также, не обращаясь к подчиненной таблице, сразу из основной таблицы документа. В этом случае поле результата запроса, содержащее данные из табличной части (например, *ЗаказТовара.Состав*), будет иметь тип *РезультатЗапроса*, то есть содержать вложенный результат запроса, сформированный на основе табличной части (листинг 1.42).

Листинг 1.42. Выбор данных из табличной части в качестве вложенной таблицы

```
ВЫБРАТЬ
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
```

```
  ЗаказТовара.СуммаЗаказа,
  ЗаказТовара.Состав
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

Вложенный результат запроса к табличной части состоит из стандартных полей *Ссылка*, *НомерСтроки* и полей с именами, соответствующими именам реквизитов табличной части документа (рис. 1.34).

Документ "ЗаказТовара" (табличная часть "Состав")

Ссылка	НомерСтроки	Товар	Количество	Сумма

Рис. 1.34. Поля табличной части, выбираемые запросом

Можно указать конкретные поля табличной части, которые нужно выбрать запросом (листинг 1.43).

Листинг 1.43. Выбор данных из табличной части в качестве вложенной таблицы

```
// Первый вариант - одна вложенная таблица
ВЫБРАТЬ
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
  ЗаказТовара.СуммаЗаказа,
  ЗаказТовара.Состав.(
    Товар,
    Количество,
    Сумма
  )
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара

// Второй вариант - две вложенные таблицы
ВЫБРАТЬ
  ЗаказТовара.Номер,
  ЗаказТовара.Клиент,
  ЗаказТовара.СуммаЗаказа,
  ЗаказТовара.Состав.Товар,
  ЗаказТовара.Состав.Сумма
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

Важно понимать, что при выполнении первого запроса будет получена одна вложенная таблица *Состав*, содержащая поля *Товар*, *Количество* и *Сумма*. А при выполнении второго запроса будет получено две вложенных таблицы: таблица *Состав*, содержащая результат запроса, который включает единственное поле *Товар*, и таблица *Состав1*,

содержащая результат запроса, который включает единственное поле *Сумма* (рис. 1.35).

Одна вложенная таблица

Запрос: Документ.ЗаказТовара (Записей в результате: 7)

Номер	Клиент	СуммаЗаказа	Состав
000000001	Соколов Иван Андреевич	25 000	ТаблицаЗначений
000000002	Соколов Иван Андреевич	40 000	ТаблицаЗначений
000000003	Орлов Сергей Иванович	3 000	ТаблицаЗначений
000000004	Маслова Ирина Николаевна	36 800	ТаблицаЗначений
000000005	Маслова Ирина Николаевна	86 000	ТаблицаЗначений
000000006	Соколов Иван Андреевич	30 000	ТаблицаЗначений
000000007	Орлов Сергей Иванович	2 700	ТаблицаЗначений

Две вложенных таблицы

Запрос: Документ.ЗаказТовара (Записей в результате: 7)

Номер	Клиент	СуммаЗаказа	Состав	Состав1
000000001	Соколов Иван Андреевич	25 000	ТаблицаЗначений	ТаблицаЗначений
000000002	Соколов Иван Андреевич	40 000	ТаблицаЗначений	ТаблицаЗначений
000000003	Орлов Сергей Иванович	3 000	ТаблицаЗначений	ТаблицаЗначений
000000004	Маслова Ирина Николаевна	36 800	ТаблицаЗначений	ТаблицаЗначений
000000005	Маслова Ирина Николаевна	86 000	ТаблицаЗначений	ТаблицаЗначений
000000006	Соколов Иван Андреевич	30 000	ТаблицаЗначений	ТаблицаЗначений
000000007	Орлов Сергей Иванович	2 700	ТаблицаЗначений	ТаблицаЗначений

Рис. 1.35. Вывод данных из документов «ЗаказТовара» в консоли запросов

При выполнении этих запросов в консоли запросов вместо данных вложенной таблицы будет выведен текст *ТаблицаЗначений*. Это значит, что данные табличной части, выбранные в качестве вложенного результата запроса, могут быть обработаны только с помощью встроенного языка.

подробнее

Обход результата запроса, содержащего данные табличной части, рассматривается в разделе «Обход выборки результата запроса, содержащего данные табличной части».

Как получить записи иерархической таблицы и расположить их в порядке иерархии

В этом примере мы рассмотрим особенности получения данных из иерархического справочника.

В нашей демонстрационной конфигурации существует иерархический справочник *Товары* с типом иерархии *Иерархия групп и элементов*. Такой справочник состоит из групп и элементов. *Группы* – это элементы справочника, которые имеют (или могут иметь) подчиненные себе элементы или другие группы. В отличие от них, *элементы* не имеют (и не могут иметь) подчиненных себе записей.

Иерархия может быть многоуровневой, то есть внутри групп могут быть другие группы и элементы, не являющиеся группами (рис. 1.36).

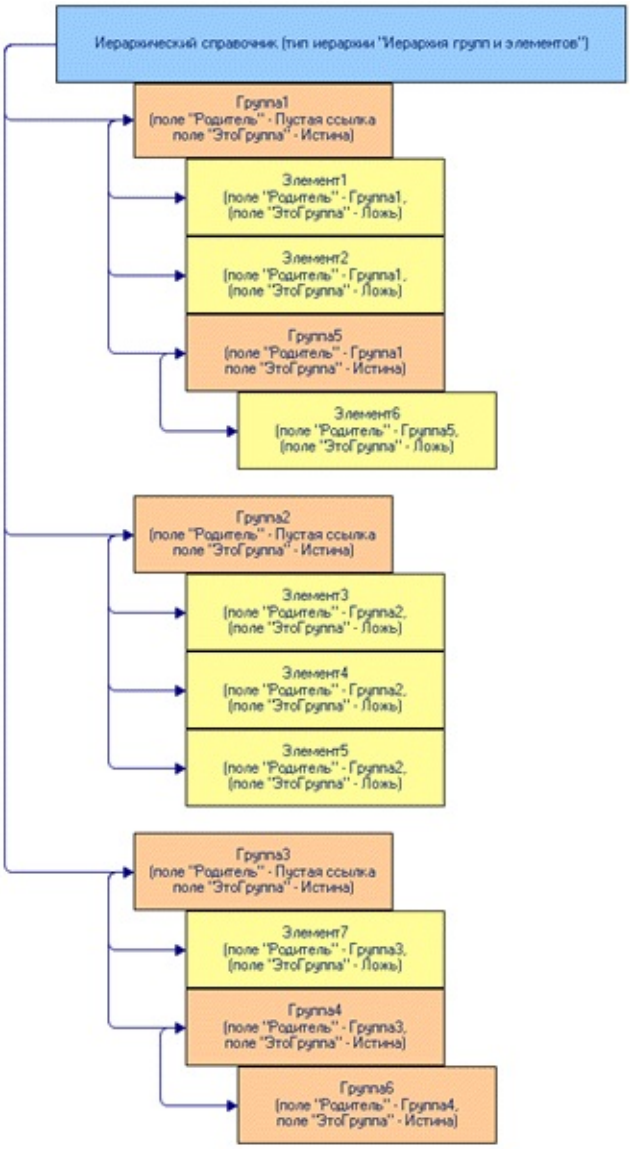


Рис. 1.36. Структура иерархического справочника

У рассмотренного на рис. 1.36 иерархического справочника есть стандартные поля: поле *Родитель* (ссылка на родительские записи у дочерних записей) и поле *ЭтоГруппа* (*ИСТИНА* для записей, являющихся группой, и *ЛОЖЬ* для записей, не являющихся группой).

В нашей демонстрационной конфигурации справочник *Товары* имеет трехуровневую иерархию и выглядит следующим образом (рис. 1.37).

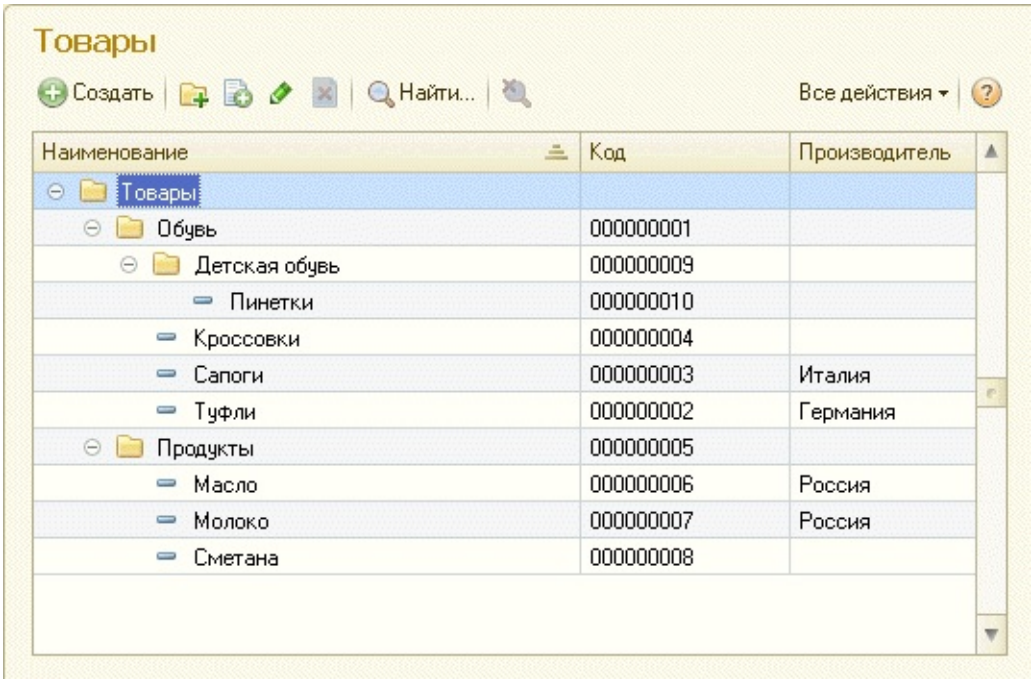


Рис. 1.37. Справочник «Товары» в режиме «1С:Предприятие»

Выберем поля *Код*, *Наименование*, *Родитель*, *ЭтоГруппа* из справочника *Товары* и отсортируем их по наименованию уже знакомым нам запросом (листинг 1.44).

Листинг 1.44. Вывод записей иерархического справочника «Товары»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование КАК Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
УПОРЯДОЧИТЬ ПО
  Наименование
```

Результат выполнения запроса представлен на рис. 1.38.

Запрос: Справочник.Товары (Записей в результате: 10)

Код	Наименование	Родитель	ЭтоГруппа
000000009	Детская обувь	Обувь	Да
000000004	Кроссовки	Обувь	Нет
000000006	Масло	Продукты	Нет
000000007	Молоко	Продукты	Нет
000000001	Обувь		Да
000000010	Пинетки	Детская обувь	Нет
000000005	Продукты		Да
000000003	Сапоги	Обувь	Нет
000000008	Сметана	Продукты	Нет
000000002	Туфли	Обувь	Нет

Рис. 1.38. Вывод записей иерархического справочника «Товары»

Мы видим, что записи справочника отсортированы по наименованию, но элементы групп (*Обувь*, *Детская обувь*, *Продукты*) идут вперемешку. Поскольку справочник является иерархическим, гораздо привычнее видеть записи этого справочника расположенными в иерархическом порядке. По умолчанию так и сделано в «1С:Предприятии».

Для этого изменим текст запроса. В предложении *УПОРЯДОЧИТЬ ПО* после имени поля *Наименование* напишем ключевое слово *ИЕРАРХИЯ* (листинг 1.45).

Листинг 1.45. Вывод записей справочника «Товары», расположенных в порядке иерархии

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование КАК Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
УПОРЯДОЧИТЬ ПО
  Ссылка ИЕРАРХИЯ,
  Наименование
```

В результате выполнения этого запроса записи справочника будут расположены в порядке иерархии – начиная от самых верхних корневых записей до самых последних элементов в цепочке иерархии (рис. 1.39).

Запрос: Справочник.Товары (Записей в результате: 10)

Код	Наименование	Родитель	ЭтоГруппа
000000001	Обувь		Да
000000009	Детская обувь	Обувь	Да
000000010	Пинетки	Детская обувь	Нет
000000004	Кроссовки	Обувь	Нет
000000003	Сапоги	Обувь	Нет
000000002	Туфли	Обувь	Нет
000000005	Продукты		Да
000000006	Масло	Продукты	Нет
000000007	Молоко	Продукты	Нет
000000008	Сметана	Продукты	Нет

Рис. 1.39. Вывод записей справочника «Товары», расположенных в порядке иерархии

Теперь покажем, как получать запросом реквизиты иерархического справочника с иерархией групп и элементов. Справочник *Товары* в демонстрационной конфигурации имеет реквизит *Производитель*, свойство *Использование* которого установлено в значение *Для элемента*.

Следует учитывать тот факт, что реквизиты, которые используются только для элементов справочника, будут содержать значение *NULL* в записях, которые являются группами. Аналогично реквизиты, у которых свойство *Использование* установлено в значение *Для группы*, будут содержать *NULL* в записях-элементах. Важно понимать, что значение *NULL* не является нулем или пустой строкой. Значения данного типа обозначают отсутствующие значения или значения, не имеющие смысла.

Для примера выполним запрос, в котором для записей, содержащих значение *NULL* в поле *Производитель*, выводится строка «NULL». В противном случае выводится содержимое поля *Производитель* (листинг 1.46).

Листинг 1.46. Вывод значения реквизита иерархического справочника

```
ВЫБРАТЬ
  Товары.Наименование КАК Наименование,
  Товары.ЭтоГруппа,
  ВЫБОР
    КОГДА (Товары.Производитель) ЕСТЬ NULL ТОГДА "NULL"
    ИНАЧЕ Товары.Производитель
  КОНЕЦ КАК Производитель
ИЗ
  Справочник.Товары КАК Товары
УПОРЯДОЧИТЬ ПО
  Наименование ИЕРАРХИЯ
```

Для формирования поля выборки *Производитель* используется *операция выбора* (*ВЫБОР (КОГДА … ТОГДА) ИНАЧЕ … КОНЕЦ*). В этой операции после ключевого слова *КОГДА* записывается условие выбора, после ключевого слова *ТОГДА* следует значение поля выборки в случае, если условие истинно. В общем случае в операции выбора может указываться неограниченное количество альтернативных одиночных выборов *КОГДА … ТОГДА*. Значение выражения, указанного после слова *ИНАЧЕ,* используется в качестве результата операции выбора в том случае, если ни одно из ранее указанных альтернативных условий выбора не было выполнено.

В результате выполнения запроса мы видим, что для записей, являющихся группами (*Обувь*, *Детская обувь*, *Продукты*), в поле выборки запроса *Производитель* выводится строка «NULL», а для элементов справочника, не являющихся группой, выводится содержимое поля *Производитель* (рис. 1.40).

Запрос: Справочник.Товары (Записей в результате: 10)

Наименование	ЭтоГруппа	Производитель
Обувь	Да	NULL
Детская обувь	Да	NULL
Пинетки	Нет	
Кроссовки	Нет	
Сапоги	Нет	Италия
Туфли	Нет	Германия
Продукты	Да	NULL
Масло	Нет	Россия
Молоко	Нет	Россия
Сметана	Нет	

Рис. 1.40. Вывод реквизитов иерархического справочника

При этом значения этого поля для некоторых записей (*Пинетки*, *Кроссовки*, *Сметана*) содержат пустую строку, но это не значение *NULL* – просто для этих элементов справочника значение реквизита *Производитель* было не заполнено (см. рис. 1.37).

Как отобрать записи иерархической таблицы по условию

В этом примере мы рассмотрим типичные задачи получения записей из иерархической таблицы. Для начала посмотрим еще раз, как выглядит справочник *Товары* в нашей демонстрационной конфигурации (рис. 1.41).

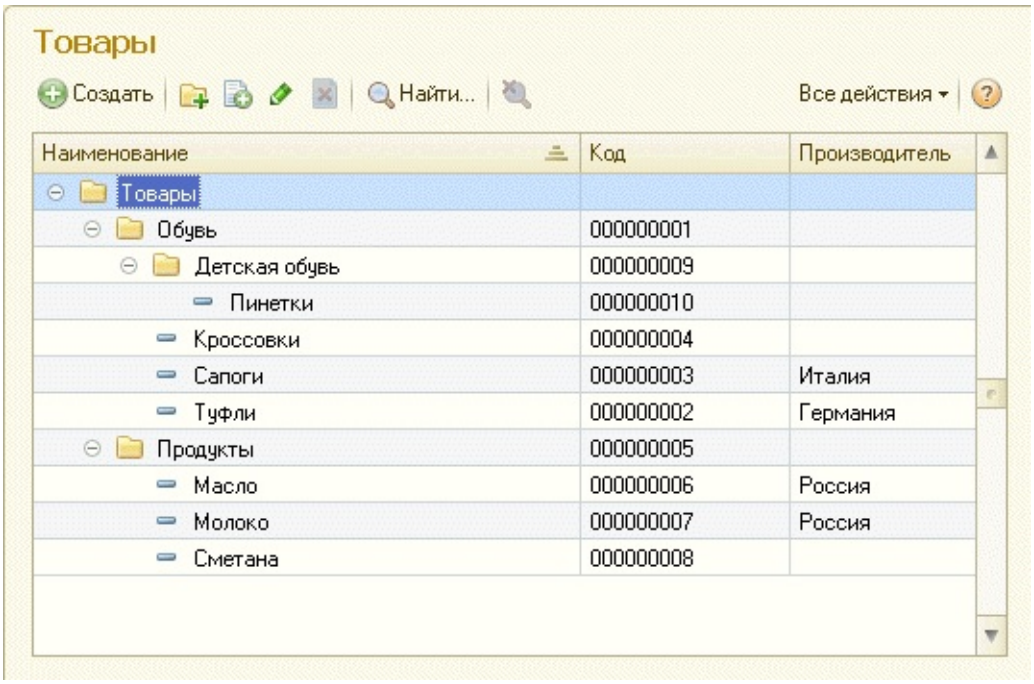


Рис. 1.41. Справочник «Товары» в режиме «1С:Предприятие»

На рис. 1.41 справочник представлен в виде дерева, на котором отображаются все уровни иерархии. Мы видим, что элементы *Обувь* и *Продукты* являются группами справочника (т. е. имеют подчиненные записи), а также они являются корневыми элементами, (т. е. не имеют родителей). Группа *Продукты* является родителем элементов *Масло*, *Молоко*, *Сметана*. Группа *Обувь* является родителем элементов *Кроссовки*, *Сапоги* и *Туфли*, а также имеет в подчинении (в иерархии) группу товаров *Детская обувь*, которая, в свою очередь, является родителем для элемента *Пинетки*. Все элементы справочника, кроме записей, являющихся группами (*Обувь*, *Детская обувь*, *Продукты*), не имеют и не могут иметь других подчиненных элементов.

Как получить записи иерархической таблицы, не являющиеся группой

Для решения поставленной задачи с помощью предложения *ГДЕ* зададим условие отбора записей из таблицы так, чтобы выводились только записи справочника, не являющиеся группой (листинг 1.47).

Листинг 1.47. Вывод записей справочника «Товары», не являющихся группой

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

В данном запросе в условии отбора мы сравниваем значение поля *ЭтоГруппа* с логическим литералом *ЛОЖЬ*. Условие выполняется для всех записей справочника, не являющихся группой, о чем говорилось выше.

Результат выполнения запроса представлен на рис. 1.42.

Запрос: Справочник.Товары (Записей в результате: 7)

Код	Наименование	Родитель	ЭтоГруппа
000000002	Туфли	Обувь	Нет
000000003	Сапоги	Обувь	Нет
000000004	Кроссовки	Обувь	Нет
000000006	Масло	Продукты	Нет
000000007	Молоко	Продукты	Нет
000000008	Сметана	Продукты	Нет
000000010	Пинетки	Детская обувь	Нет

Рис. 1.42. Вывод записей справочника «Товары», не являющихся группой

Соответственно, для вывода только групп из справочника нужно задать условие отбора *Товары.ЭтоГруппа = ИСТИНА*.

Теперь покажем, как сделать это же условие параметризированным и, следовательно, более гибким. Для этого перепишем предыдущий запрос с использованием параметров (листинг 1.48).

Листинг 1.48. Вывод записей справочника «Товары», не являющихся группой

```
// Условие без параметров
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ

// Параметризированное условие
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.ЭтоГруппа = &ЭтоГруппа
```

Нажмем кнопку *Заполнить параметры*, и параметр *ЭтоГруппа* будет добавлен в окно параметров консоли запросов. В данном случае тип параметра – *Булево*, поэтому в поле *Значение* мы можем выбрать возможные логические значения параметра (*Да*/*Нет*).

Если мы зададим значение параметра как *Нет*, то в результате выполнения запроса

увидим только те записи иерархического справочника *Товары*, которые не являются группой (рис. 1.43).

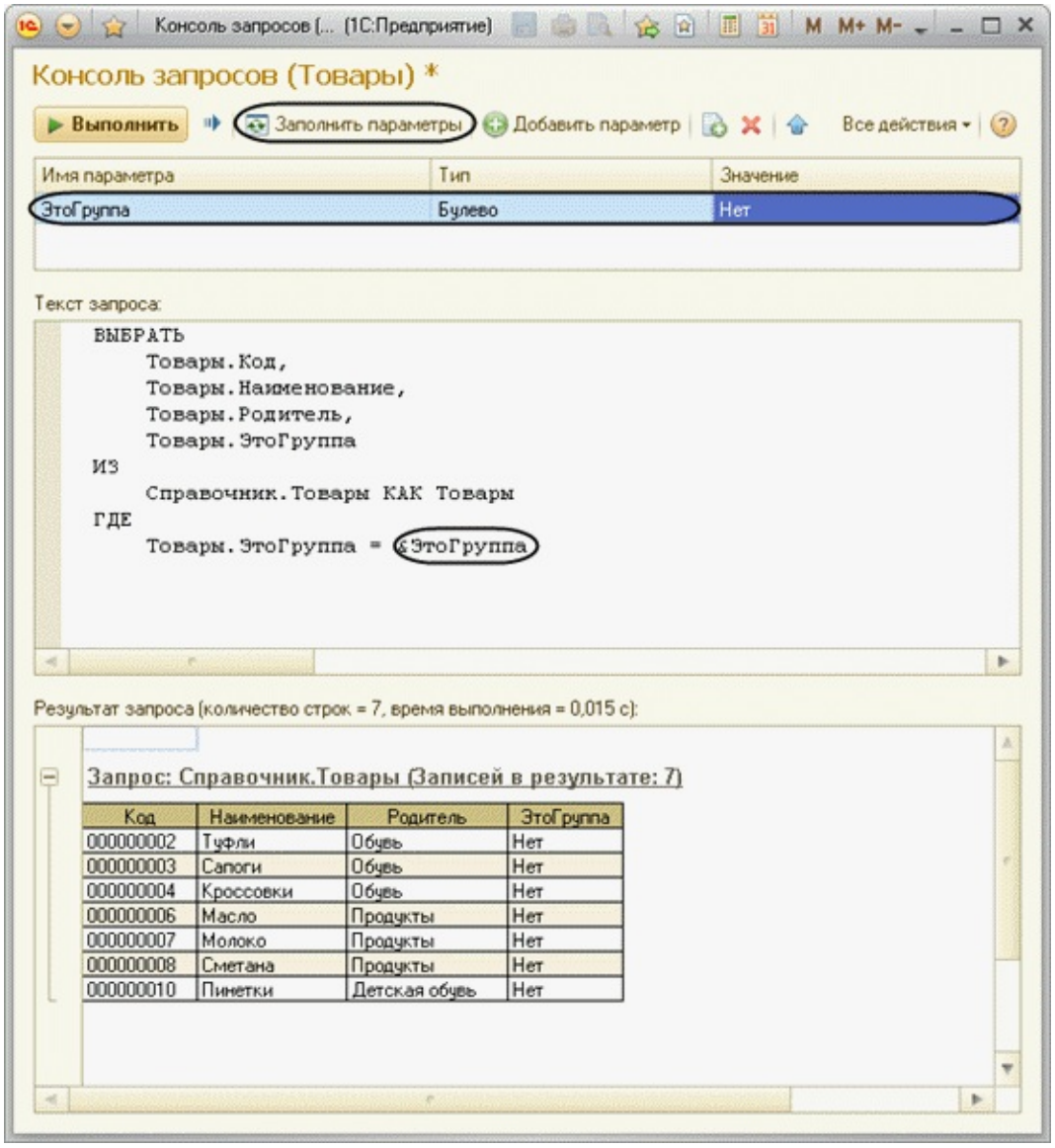


Рис. 1.43. Выполнение параметризированного запроса в консоли запросов

То есть результат запроса полностью совпадает с результатом при выполнении запроса, не использующего параметры в условии отбора (см. рис. 1.42). Изменив значение параметра *ЭтоГруппа* на *Да*, с помощью этого же параметризированного запроса мы получим только те записи справочника, которые являются группой.

Как получить записи иерархической таблицы, находящиеся в выбранной группе

Для решения поставленной задачи зададим параметризированное условие отбора записей из таблицы, родителем которых является выбранная группа. Для этого нужно использовать параметр ссылочного типа, имеющий тип ссылки на справочник *Товары*.

Добавим в условие отбора предыдущего запроса условие для выбора из иерархического справочника тех записей, для которых родителем является элемент справочника, указанный в параметре типа *СправочникСсылка.Товары*. Это можно сделать с помощью следующего запроса (листинг 1.49).

Листинг 1.49. Отбор записей справочника «Товары» по условию

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.ЭтоГруппа = &ЭтоГруппа
  И Товары.Родитель = &Родитель
```

В данном запросе условие отбора определено как сложное логическое выражение, в котором простые логические выражения соединяются между собой логическим оператором *И*. В первом выражении проверяется, является ли запись группой, во втором выражении проверяется, принадлежит ли запись указанной группе. В целом условие отбора будет истинно, если оба этих условия будут выполнены.

При нажатии кнопки *Заполнить параметры* параметр *ЭтоГруппа* и параметр *Родитель* типа *СправочникСсылка.Товары* будут добавлены в окно параметров консоли запроса. Укажем в качестве значения параметра группу *Обувь* справочника *Товары*, в качестве значения параметра *ЭтоГруппа – Нет* и выполним запрос (рис. 1.44).

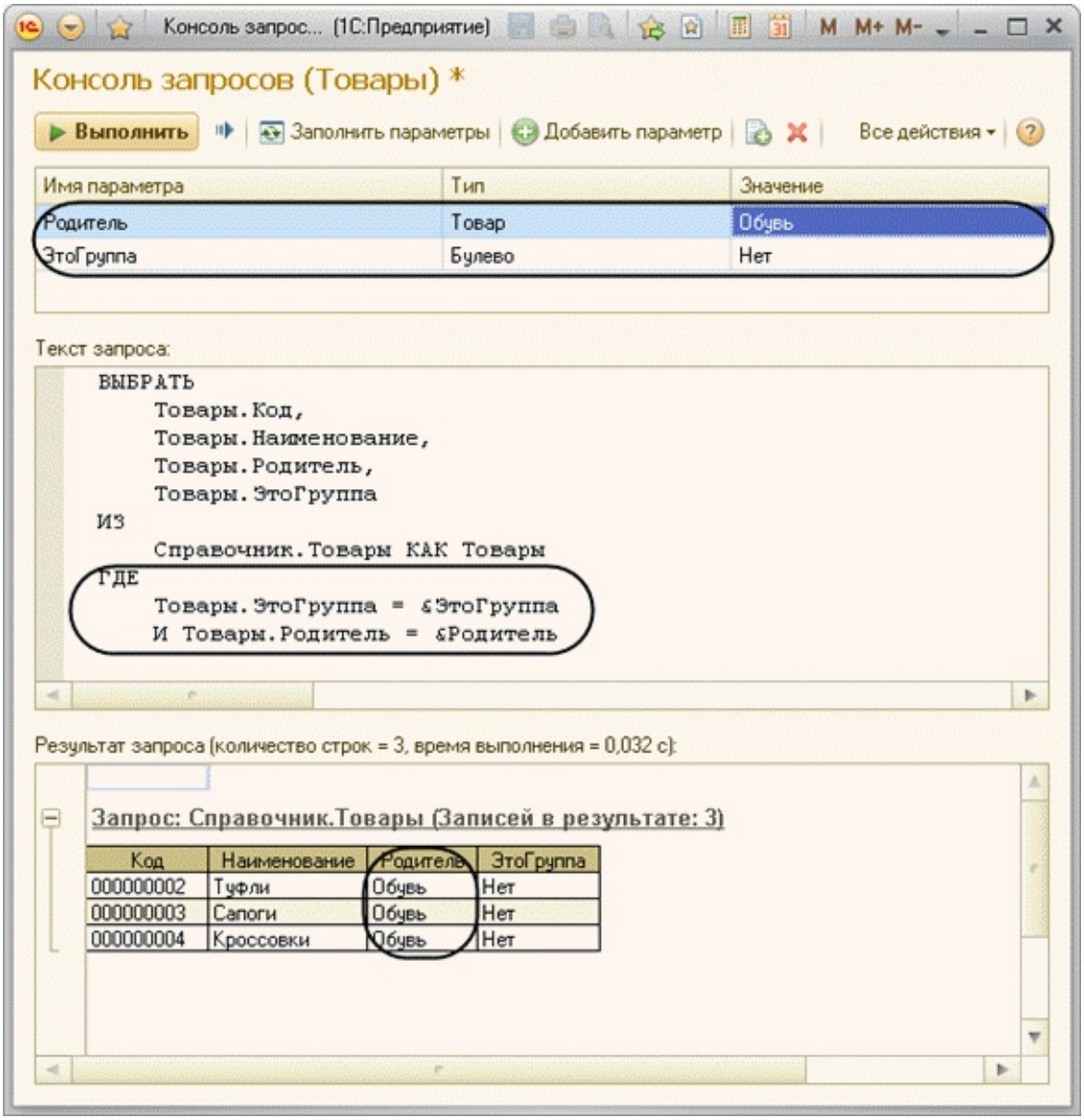


Рис. 1.44. Выполнение параметризированного запроса в консоли запросов

Как мы видим, в результат запроса попадают только те записи справочника, которые не являются группой и родителем которых является группа *Обувь*. Изменив значение параметра *ЭтоГруппа* на *Да*, с помощью этого же параметризированного запроса мы получим только те записи справочника, которые являются группой и родителем которых является группа *Обувь* (рис. 1.45).

Рис. 1.45. Вывод только групп из выбранной группы товаров

Как получить «корневые» записи иерархической таблицы

Для решения поставленной задачи нужно отобрать записи таблицы, которые находятся в «корне» иерархического справочника, то есть те записи, у которых нет родителя. Для этого зададим условие отбора записей из таблицы так, чтобы выводились только записи справочника, у которых в поле *Родитель* находится пустая ссылка на справочник *Товары* (листинг 1.50).

Листинг 1.50. Отбор корневых записей справочника «Товары»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.Родитель = ЗНАЧЕНИЕ(Справочник.Товары.ПустаяСсылка)
```

Запрос: Справочник.Товары (Записей в результате: 2)

Код	Наименование	Родитель	ЭтоГруппа
000000001	Обувь		Да
000000005	Продукты		Да

Рис. 1.46. Вывод корневых записей справочника «Товары»

В условии отбора данного запроса используется литерал функционального типа *ЗНАЧЕНИЕ()*, с помощью которого в языке запросов можно получить доступ к предопределенным данным конфигурации и к значениям системных перечислений. В данном случае проверяется, является ли пустой ссылкой на справочник *Товары* значение поля справочника *Родитель*. Если это так, то это корневые записи справочника.

Как получить записи иерархической таблицы, находящиеся в иерархии выбранной группы

Для решения поставленной задачи нужно отобрать записи таблицы, которые находятся в подчинении указанной группе. На первый взгляд это условие отбора схоже с отбором записей справочника по родителю, который мы рассматривали выше. Однако при таком условии мы увидим только те записи иерархического справочника, непосредственным родителем которых является выбранная группа товаров. Построить всю иерархию подчиненности элементов сверху вниз так не получится. Для этого в условии отбора нужно использовать ключевое слово *В ИЕРАРХИИ* и в качестве параметра передать в список иерархии группу товаров (листинг 1.51).

Листинг 1.51. Отбор записей справочника «Товары», находящихся в иерархии выбранной группы

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.Ссылка В ИЕРАРХИИ (&ГруппаТоваров)
УПОРЯДОЧИТЬ ПО
  Наименование ИЕРАРХИЯ
```

В результате выполнения данного запроса будут выбраны все записи справочника, которые иерархически принадлежат группе, указанной в параметре *ГруппаТоваров*.

Укажем в качестве значения параметра группу *Обувь* и получим не только записи,

родителем которых является выбранная группа, но и все элементы, находящиеся ниже по уровню иерархии (рис. 1.47).

Запрос: Справочник.Товары (Записей в результате: 6)

Код	Наименование	Родитель	ЭтоГруппа
000000001	Обувь		Да
000000009	Детская обувь	Обувь	Да
000000010	Пинетки	Детская обувь	Нет
000000004	Кроссовки	Обувь	Нет
000000003	Сапоги	Обувь	Нет
000000002	Туфли	Обувь	Нет

Рис. 1.47. Отбор записей справочника «Товары», находящихся в иерархии выбранной группы товаров

Мы видим, что сам элемент справочника, переданный в список иерархии (неважно, является он группой справочника или нет), также включается в результат запроса (первая запись в таблице). Этот момент связан с особенностью выполнения конструкции *В ИЕРАРХИИ* языка запросов.

Если мы укажем в качестве значения параметра группу *Детская обувь*, то результат выполнения запроса будет следующим (рис. 1.48).

Запрос: Справочник.Товары (Записей в результате: 2)

Код	Наименование	Родитель	ЭтоГруппа
000000009	Детская обувь	Обувь	Да
000000010	Пинетки	Детская обувь	Нет

Рис. 1.48. Отбор записей справочника «Товары», находящихся в иерархии выбранной группы товаров

Как получить всех родителей для выбранного элемента иерархической таблицы

Для решения поставленной задачи требуется построить всю цепочку вверх по иерархии от выбранного элемента иерархической таблицы. Для этого нужно использовать несколько запросов и уметь обходить результаты полученных запросов с помощью встроенного языка. Поэтому такой пример мы рассмотрим позже, в разделе "Примеры решения различных задач с использованием запросов".

Как узнать среднюю цену, по которой продавался товар

Часто значения некоторых полей в таблицах могут повторяться. Например, вводится несколько заказов для одного и того же клиента, или один и тот же товар продается в расходных накладных. Обычно в прикладных задачах требуется узнать, какова максимальная сумма заказа для каждого клиента, сколько и по каким ценам продали каждого товара и т. п.

Для решения подобных задач применяется *группировка* записей. Сущность этой операции заключается в том, что вместо нескольких записей из исходной таблицы, имеющих одинаковое значение некоторого поля, в результат запроса включается одна запись, в которой от значений других полей вычисляется некоторое обобщенное значение (рассчитывается *агрегатная* функция). Например, значения другого поля для группируемых записей суммируются, вычисляется максимальное, минимальное значение, количество и т. д.

Рассмотрим данную задачу на примере расходных накладных. Для начала мы просто выведем все записи из состава расходных накладных, упорядочив их по ссылкам товаров (листинг 1.52). Делается это простым запросом к табличной части документа, который мы рассматривали в разделе «Как получить данные из табличной части некоторого документа».

Листинг 1.52. Вывод всех записей из состава расходных накладных, упорядоченных по ссылкам товаров

```
ВЫБРАТЬ
  НакладнаяСостав.Товар КАК Товар,
  НакладнаяСостав.Количество,
  НакладнаяСостав.Цена,
  НакладнаяСостав.Сумма
ИЗ
  Документ.РасходнаяНакладная.Состав КАК НакладнаяСостав
УПОРЯДОЧИТЬ ПО
  Товар
```

Даже на нашей небольшой демонстрационной конфигурации получается довольно большой список товаров, но с его помощью будет понятна сущность группировки записей (рис. 1.49).

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 17)

Товар	Количество	Цена	Сумма
Туфли	3	6 000	18 000
Туфли	3	8 000	24 000
Туфли	3	7 000	21 000
Туфли	5	6 000	30 000
Сапоги	3	10 000	30 000
Сапоги	1	12 000	12 000
Сапоги	3	11 000	33 000
Сапоги	3	11 000	33 000
Кроссовки	10	3 500	35 000
Кроссовки	5	4 000	20 000
Кроссовки	5	3 000	15 000
Масло	20	100	2 000
Масло	20	90	1 800
Масло	10	90	900
Молоко	20	50	1 000
Молоко	10	60	600
Молоко	10	60	600

Рис. 1.49. Вывод всех записей из состава расходных накладных, упорядоченных по ссылкам товаров

Итак, в результате запроса содержится 17 записей, из них 4 записи для товара *Туфли*, 4 записи для товара *Сапоги*, 3 записи для товара *Кроссовки* и т. д. При этом записи с одинаковым значением поля *Товар* расположены друг за другом. Мы видим, что цена одного и того же товара может быть разной. Так, цена товара *Туфли* колеблется от 6 000 до 8 000.

Допустим, нас интересует минимальная, максимальная и средняя цена, по которой продавался каждый из товаров. Для решения поставленной задачи нужно собрать вместе записи исходной таблицы с одинаковым значением поля *Товар* и применить к каждому товару агрегатные функции *МИНИМУМ()*, *МАКСИМУМ()* и *СРЕДНЕЕ()* со значением цены товара в качестве параметра этих функций.

Для этого в языке запросов предназначено ключевое слово *СГРУППИРОВАТЬ ПО*, после которого следует список полей, по которым нужно сгруппировать записи исходной таблицы (листинг 1.53).

Листинг 1.53. Вывод минимальной, максимальной и средней цены каждого товара из состава расходных накладных

```
ВЫБРАТЬ
  НакладнаяСостав.Товар КАК Товар,
  МИНИМУМ(НакладнаяСостав.Цена) КАК Минимум,
  МАКСИМУМ(НакладнаяСостав.Цена) КАК Максимум,
  СРЕДНЕЕ(НакладнаяСостав.Цена) КАК Среднее
ИЗ
  Документ.РасходнаяНакладная.Состав КАК НакладнаяСостав
СГРУППИРОВАТЬ ПО
  НакладнаяСостав.Товар
```

Обратите внимание, что в списке полей выборки запроса, помимо полей, по которым группируются записи исходной таблицы, могут присутствовать только агрегатные функции *СУММА()*, *МИНИМУМ()*, *МАКСИМУМ()*, *СРЕДНЕЕ()*, *КОЛИЧЕСТВО()*, применяемые к исходным записям с одинаковым значением поля группировки.

Агрегатные функции *МИНИМУМ()*, *МАКСИМУМ()* и *СРЕДНЕЕ()* вычисляют соответственно минимальное, максимальное и среднее значение из всех записей с одинаковым значением поля группировки.

В результате выполнения запроса выводится 5 записей вместо 17. То есть для каждого товара вычисляется минимальная, максимальная и средняя цена его продаж. В результате запроса записи исходной таблицы с одинаковым значением поля *Товар* «сворачиваются» в одну запись, в которой помимо самого поля группировки выводятся значения агрегатных функций (рис. 1.50).

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 5)

Товар	Минимум	Максимум	Среднее
Туфли	6 000	8 000	6 750
Сапоги	10 000	12 000	11 000
Кроссовки	3 000	4 000	3 500
Масло	90	100	93,333333
Молоко	50	60	56,666667

Рис. 1.50. Вывод минимальной, максимальной и средней цены каждого товара из состава расходных накладных

В предложении *СГРУППИРОВАТЬ ПО* должны указываться именно имена полей, а не их псевдонимы, определенные в запросе.

При использовании агрегатных функций предложение *СГРУППИРОВАТЬ ПО* может быть и не указано совсем, при этом все записи исходной таблицы будут сгруппированы в одну-единственную строку.

Как узнать общее количество и сумму продаж каждого товара в разрезе покупателей

Сначала покажем простой запрос, с помощью которого можно узнать, сколько всего было продано каждого из товаров.

Для этого записи из состава расходных накладных с одинаковым значением поля *Товар* нужно собрать вместе, затем для каждого товара с помощью агрегатной функции *СУММА()* подсчитать суммарное значение поля *Количество* и вывести в полях выборки запроса (листинг 1.54).

Листинг 1.54. Вывод общего количества единиц проданных товаров из состава расходных накладных

```
ВЫБРАТЬ
  НакладнаяСостав.Товар КАК Товар,
  СУММА(НакладнаяСостав.Количество) КАК Количество
ИЗ
  Документ.РасходнаяНакладная.Состав КАК НакладнаяСостав
СГРУППИРОВАТЬ ПО
  НакладнаяСостав.Товар
```

Агрегатная функция *СУММА()* вычисляет арифметическую сумму всех записей с одинаковым значением поля группировки.

В результате запроса мы видим общее количество единиц проданных товаров из состава расходных накладных (рис. 1.51).

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 5)

Товар	Количество
Туфли	14
Сапоги	10
Кроссовки	20
Масло	50
Молоко	40

Рис. 1.51. Вывод общего количества единиц проданных товаров из состава расходных накладных

Часто требуется сгруппировать записи исходной таблицы по значению нескольких полей сразу. Для этого после ключевого слова *СГРУППИРОВАТЬ ПО* нужно перечислить через запятую список полей группировки.

Например, нам нужно увидеть, сколько и на какую сумму продано каждого товара по каждому покупателю. Это можно сделать с помощью следующего запроса (листинг 1.55).

Листинг 1.55. Группировка записей по нескольким полям

```
ВЫБРАТЬ
  НакладнаяСостав.Ссылка.Покупатель КАК Покупатель,
  НакладнаяСостав.Товар КАК Товар,
  СУММА(НакладнаяСостав.Количество) КАК Количество,
  СУММА(НакладнаяСостав.Сумма) КАК Сумма
ИЗ
  Документ.РасходнаяНакладная.Состав КАК НакладнаяСостав
СГРУППИРОВАТЬ ПО
  НакладнаяСостав.Ссылка.Покупатель,
  НакладнаяСостав.Товар
```

В данном запросе записи исходной таблицы с одинаковыми значениями полей *Покупатель* и *Товар* собираются вместе. Причем поскольку мы обращаемся к табличной части документа *РасходнаяНакладная*, для выбора реквизита *Покупатель* мы обращаемся к нему через точку от поля табличной части *Ссылка*. Затем для каждого товара по каждому покупателю с помощью агрегатной функции *СУММА()* подсчитывается суммарное значение полей *Количество* и *Сумма* из состава расходных накладных и выводится в полях выборки запроса.

Обратите внимание, что в тексте запроса имена полей в списке группировки принято располагать на разных строках со смещением относительно ключевого слова *СГРУППИРОВАТЬ ПО*.

В результате в нижней таблице мы видим общее количество и сумму продаж каждого товара по каждому покупателю, а в верхней таблице для сравнения приведен состав всех расходных накладных, упорядоченный по покупателям и товарам (рис. 1.52).

Состав расходных накладных, упорядоченный по покупателям и товарам

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 17)

Покупатель	Товар	Количество	Сумма
Соколов Иван Андреевич	Туфли	5	30 000
Соколов Иван Андреевич	Сапоги	3	33 000
Соколов Иван Андреевич	Кроссовки	10	35 000
Орлов Сергей Иванович	Масло	20	1 800
Орлов Сергей Иванович	Масло	20	2 000
Орлов Сергей Иванович	Молоко	20	1 000
Маслова Ирина Николаевна	Туфли	3	18 000
Маслова Ирина Николаевна	Туфли	3	24 000
Маслова Ирина Николаевна	Туфли	3	21 000
Маслова Ирина Николаевна	Сапоги	3	30 000
Маслова Ирина Николаевна	Сапоги	1	12 000
Маслова Ирина Николаевна	Сапоги	3	33 000
Маслова Ирина Николаевна	Кроссовки	5	20 000
Маслова Ирина Николаевна	Кроссовки	5	15 000
Маслова Ирина Николаевна	Масло	10	900
Маслова Ирина Николаевна	Молоко	10	600
Маслова Ирина Николаевна	Молоко	10	600

Общее количество и сумма продаж каждого товара по каждому покупателю

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 10)

Покупатель	Товар	Количество	Сумма
Соколов Иван Андреевич	Туфли	5	30 000
Соколов Иван Андреевич	Сапоги	3	33 000
Соколов Иван Андреевич	Кроссовки	10	35 000
Орлов Сергей Иванович	Масло	40	3 800
Маслова Ирина Николаевна	Туфли	9	63 000
Орлов Сергей Иванович	Молоко	20	1 000
Маслова Ирина Николаевна	Сапоги	7	75 000
Маслова Ирина Николаевна	Кроссовки	10	35 000
Маслова Ирина Николаевна	Масло	10	900
Маслова Ирина Николаевна	Молоко	20	1 200

Рис. 1.52. Общее количество и сумма продаж каждого товара по каждому покупателю

Иногда требуется наложить условие на значения агрегатных функций, применяемых к исходным записям с одинаковым значением поля группировки. Это можно сделать с

помощью ключевого слова *ИМЕЮЩИЕ*, после которого задается условие отбора аналогично условию в предложении *ГДЕ*, но только оно накладывается не на исходные записи, а на записи, получившиеся в результате группировки.

Например, нам нужно увидеть, какие товары и какому покупателю были проданы на сумму свыше 50 000. Это можно сделать с помощью следующего запроса (листинг 1.56).

Листинг 1.56. Отбор товаров, проданных на сумму более 50 000 по каждому покупателю

```
ВЫБРАТЬ
  НакладнаяСостав.Ссылка.Покупатель КАК Покупатель,
  НакладнаяСостав.Товар КАК Товар,
  СУММА(НакладнаяСостав.Количество) КАК Количество,
  СУММА(НакладнаяСостав.Сумма) КАК Сумма
ИЗ
  Документ.РасходнаяНакладная.Состав КАК НакладнаяСостав
СГРУППИРОВАТЬ ПО
  НакладнаяСостав.Ссылка.Покупатель,
  НакладнаяСостав.Товар
ИМЕЮЩИЕ
  СУММА(НакладнаяСостав.Сумма) > 50000
```

То есть мы просто добавили в текст предыдущего запроса конструкцию *ИМЕЮЩИЕ <Условие отбора>*.

В итоге из общего списка продаж каждого товара по каждому покупателю (верхняя таблица) в результат запроса попадут только те записи (нижняя таблица), в которых сумма продаж товара по каждому покупателю, вычисленная в результате группировки, больше 50 000 (рис. 1.53).

Список продаж каждого товара каждому покупателю

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 10)

Покупатель	Товар	Количество	Сумма
Соколов Иван Андреевич	Туфли	5	30 000
Соколов Иван Андреевич	Сапоги	3	33 000
Соколов Иван Андреевич	Кроссовки	10	35 000
Орлов Сергей Иванович	Масло	40	3 800
Маслова Ирина Николаевна	Туфли	9	63 000
Орлов Сергей Иванович	Молоко	20	1 000
Маслова Ирина Николаевна	Сапоги	7	75 000
Маслова Ирина Николаевна	Кроссовки	10	35 000
Маслова Ирина Николаевна	Масло	10	900
Маслова Ирина Николаевна	Молоко	20	1 200

Товары, проданные на сумму свыше 50000 по каждому покупателю

Запрос: Документ.РасходнаяНакладная.Состав (Записей в результате: 2)

Покупатель	Товар	Количество	Сумма
Маслова Ирина Николаевна	Туфли	9	63 000
Маслова Ирина Николаевна	Сапоги	7	75 000

Рис. 1.53. Отбор товаров, проданных на сумму более 50 000 по каждому покупателю

Как узнать среднюю цену поступления товара, не группируя сами записи

Очень часто требуется не только вывести определенные данные в результат запроса, но и рассчитать по ним некоторые обобщенные данные (*построить итоги*). Что значит рассчитать итоги и в чем отличие построения итогов от группировки записей, поясним на конкретном примере.

Для наглядности проведем аналогию с примером «Как узнать среднюю цену, по которой продавался товар». В нем мы группировали записи исходной таблицы по полю *Товар*, чтобы узнать, например, среднюю цену продаж каждого товара. Но при группировке записи, содержащие одинаковые значения полей, по которым группируются исходные данные, сворачиваются в одну, для каждой группы записей вычисляются значения агрегатных функций и помещаются в результат запроса.

При построении итогов (так же, как и при группировке) записи, содержащие одинаковые значения полей, по которым нужно рассчитать итоги, собираются вместе, для каждой группы записей вычисляются значения агрегатных функций. При этом результат вычислений помещается в *итоговые строки* и добавляется к записям исходной таблицы (*детальным записям*). То есть детальные записи не сворачиваются, а, наоборот, достраиваются относительно исходной таблицы и служат для детализации итоговых строк.

На рис. 1.54 демонстрируется отличие группировки записей из состава приходных накладных по полю *Товар* от расчета итогов по этому же полю.

Общий состав приходных накладных

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 12)

Товар	Количество	Цена	Сумма
Туфли	5	5 000	25 000
Туфли	10	8 000	80 000
Сапоги	5	10 000	50 000
Сапоги	5	10 000	50 000
Кроссовки	15	3 000	45 000
Кроссовки	20	3 000	60 000
Масло	20	80	1 600
Масло	30	70	2 100
Молоко	30	40	1 200
Молоко	50	40	2 000
Сметана	50	50	2 500
Сметана	20	50	1 000

Результат построения итогов по полю "Товар"

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 18)

Товар	Количество	Цена	Сумма
Кроссовки	35	3 000	105 000
Кроссовки	15	3 000	45 000
Кроссовки	20	3 000	60 000
Сапоги	10	10 000	100 000
Сапоги	5	10 000	50 000
Сапоги	5	10 000	50 000
Туфли	15	6 500	105 000
Туфли	5	5 000	25 000
Туфли	10	8 000	80 000
Масло	50	75	3 700
Масло	20	80	1 600
Масло	30	70	2 100
Молоко	80	40	3 200
Молоко	30	40	1 200
Молоко	50	40	2 000
Сметана	70	50	3 500
Сметана	50	50	2 500
Сметана	20	50	1 000

Результат группировки по полю "Товар"

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 6)

Товар	Количество	Цена	Сумма
Туфли	15	6 500	105 000
Сапоги	10	10 000	100 000
Кроссовки	35	3 000	105 000
Масло	50	75	3 700
Молоко	80	40	3 200
Сметана	70	50	3 500

Рис. 1.54. Отличие группировки записей таблицы от построения итогов по одному и тому же полю

В верхней таблице выводятся все записи из состава приходных накладных, в которых записи с одинаковым значением поля *Товар* расположены друг за другом.

В нижней левой таблице выводится результат построения итогов по полю *Товар*, при этом для каждого товара рассчитывается средняя цена поступления, а также общее количество и сумма поступления товара. Результат вычислений помещается в итоговые строки и добавляется к детальным записям.

В нижней правой таблице выводится результат группировки записей приходных накладных по полю *Товар*. При этом для каждого товара рассчитываются те же агрегатные функции, но в результате запроса для каждого товара выводится одна запись, в которую помещается результат вычисления агрегатных функций.

Нетрудно увидеть, что итоговые строки в левой таблице (18 = 12 + 6) совпадают со строками в правой таблице (6) и дополняются строками из верхней таблицы (12).

Для построения итогов используется предложение *ИТОГИ … ПО*, которое располагается в секции *Описание итогов* текста запроса. Все секции запроса приведены на рис. 1.13.

Получить результат запроса, показанный на рис. 1.54, содержащий итоги по полю *Товар*, можно с помощью следующего запроса (листинг 1.57).

Листинг 1.57. Вывод всех записей из состава приходных накладных с итоговыми данными для каждого товара

```
ВЫБРАТЬ
  НакладнаяСостав.Товар КАК Товар,
  НакладнаяСостав.Количество КАК Количество,
  НакладнаяСостав.Цена КАК Цена,
  НакладнаяСостав.Сумма КАК Сумма
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК НакладнаяСостав
ИТОГИ
  СУММА(Количество),
  СРЕДНЕЕ(Цена),
  СУММА(Сумма)
ПО
  Товар
```

Описание итогов начинается с ключевого слова *ИТОГИ*. После него следует список *итоговых полей*, перечисленных через запятую. В качестве итоговых полей обычно выступает результат вычисления агрегатных функций, рассчитанный для детальных записей с одинаковыми значениями полей, по которым требуется рассчитать итоги. Затем следует ключевое слово *ПО*, после которого перечисляется список *контрольных точек* – полей, по которым необходимо рассчитать итоги.

В данном случае *Товар* – это контрольная точка, а *СУММА(Количество)*, *СРЕДНЕЕ(Цена)*, *СУММА(Сумма)* – это итоговые поля, которые требуется рассчитать для каждой контрольной точки.

В результате выполнения запроса мы видим 18 записей – 12 записей из исходной таблицы (см. рис. 1.54) и 6 итоговых записей, по одной на каждый товар. Итоговая запись содержит представление самого товара, а также общее количество, сумму и среднюю цену его поступления (рис. 1.55).

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 18)

Товар	Количество	Цена	Сумма
Кроссовки	35	3 000	105 000
Кроссовки	15	3 000	45 000
Кроссовки	20	3 000	60 000
Сапоги	10	10 000	100 000
Сапоги	5	10 000	50 000
Сапоги	5	10 000	50 000
Туфли	15	6 500	105 000
Туфли	5	5 000	25 000
Туфли	10	8 000	80 000
Масло	50	75	3 700
Масло	20	80	1 600
Масло	30	70	2 100
Молоко	80	40	3 200
Молоко	30	40	1 200
Молоко	50	40	2 000
Сметана	70	50	3 500
Сметана	50	50	2 500
Сметана	20	50	1 000

Рис. 1.55. Вывод всех записей из состава приходных накладных с итоговыми данными для каждого товара

Заметьте, что итоговые поля могут рассчитываться необязательно для всех полей выборки запроса, как в данном случае. Например, итог может рассчитываться только для поля *Сумма* или не рассчитываться вовсе.

Расчет итогов для иерархического справочника

В данном примере в качестве детальных записей для расчета итогов мы также будем использовать записи из состава приходных накладных.

Поскольку поле *Товар*, для которого рассчитываются итоги, имеет тип ссылки на иерархический справочник *Товары*, то в результат запроса можно включить также итоги по иерархии этого справочника. Для этого после имени поля необходимо указать ключевое слово *ИЕРАРХИЯ* (листинг 1.58).

Листинг 1.58. Вывод итоговых данных по товарам с итогами по иерархии

```
ВЫБРАТЬ
  НакладнаяСостав.Товар КАК Товар,
  НакладнаяСостав.Количество КАК Количество,
  НакладнаяСостав.Цена КАК Цена,
  НакладнаяСостав.Сумма КАК Сумма
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК НакладнаяСостав
ИТОГИ
  СУММА(Количество),
  СУММА(Сумма)
ПО
  Товар ИЕРАРХИЯ КАК ТоварИерархия
```

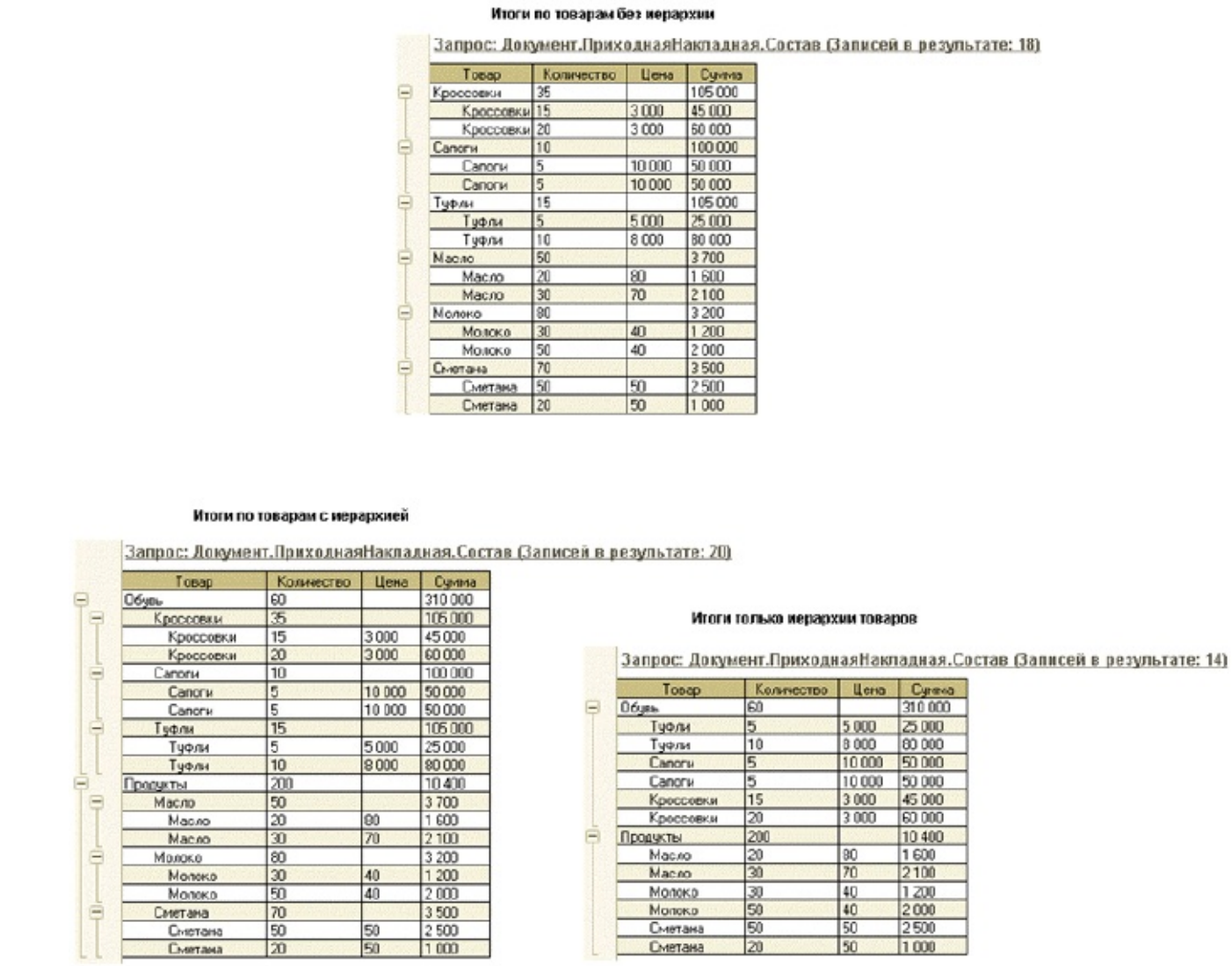
Обратите внимание, что для полей, по которым рассчитываются итоги, также как и для полей выборки запроса, можно назначить псевдоним для последующего обращения к ним из встроенного языка.

При необходимости можно рассчитать только итоги значений по иерархии, без расчета итогов в контрольных точках. Для этого перед ключевым словом *ИЕРАРХИЯ* нужно указать ключевое слово *ТОЛЬКО* (листинг 1.59).

Листинг 1.59. Вывод итоговых данных только по иерархии товаров

```
ВЫБРАТЬ
  НакладнаяСостав.Товар КАК Товар,
  НакладнаяСостав.Количество КАК Количество,
  НакладнаяСостав.Цена КАК Цена,
  НакладнаяСостав.Сумма КАК Сумма
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК НакладнаяСостав
ИТОГИ
  СУММА(Количество),
  СУММА(Сумма)
ПО
  Товар ТОЛЬКО ИЕРАРХИЯ КАК ТоварТолькоИерархия
```

В результате выполнения запросов, представленных в листингах 1.58 и 1.59, на рис. 1.56 в левой нижней таблице мы видим записи из состава приходных накладных с итогами по каждому товару, и дополнительно к этому рассчитываются итоги по иерархии для этих товаров. В правой нижней таблице рассчитываются итоги только по иерархии, без итогов по каждому товару. В верхней таблице для сравнения приведен состав приходных накладных с итогами только по товарам, без иерархии.



Итоги по товарам без иерархии

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 18)

Товар	Количество	Цена	Сумма
Кроссовки	35		105 000
Кроссовки	15	3 000	45 000
Кроссовки	20	3 000	60 000
Сапоги	10		100 000
Сапоги	5	10 000	50 000
Сапоги	5	10 000	50 000
Туфли	15		105 000
Туфли	5	5 000	25 000
Туфли	10	8 000	80 000
Масло	50		3 700
Масло	20	80	1 600
Масло	30	70	2 100
Молоко	80		3 200
Молоко	30	40	1 200
Молоко	50	40	2 000
Сметана	70		3 500
Сметана	50	50	2 500
Сметана	20	50	1 000

Итоги по товарам с иерархией

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 20)

Товар	Количество	Цена	Сумма
Обувь	60		310 000
Кроссовки	35		105 000
Кроссовки	15	3 000	45 000
Кроссовки	20	3 000	60 000
Сапоги	10		100 000
Сапоги	5	10 000	50 000
Сапоги	5	10 000	50 000
Туфли	15		105 000
Туфли	5	5 000	25 000
Туфли	10	8 000	80 000
Продукты	200		10 400
Масло	50		3 700
Масло	20	80	1 600
Масло	30	70	2 100
Молоко	80		3 200
Молоко	30	40	1 200
Молоко	50	40	2 000
Сметана	70		3 500
Сметана	50	50	2 500
Сметана	20	50	1 000

Итоги только иерархии товаров

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 14)

Товар	Количество	Цена	Сумма
Обувь	60		310 000
Туфли	5	5 000	25 000
Туфли	10	8 000	80 000
Сапоги	5	10 000	50 000
Сапоги	5	10 000	50 000
Кроссовки	15	3 000	45 000
Кроссовки	20	3 000	60 000
Продукты	200		10 400
Масло	20	80	1 600
Масло	30	70	2 100
Молоко	30	40	1 200
Молоко	50	40	2 000
Сметана	50	50	2 500
Сметана	20	50	1 000

Рис. 1.56. Вывод итоговых данных по каждому товару с иерархией, без иерархии и итогов только по иерархии товаров

Возможность построения иерархических итогов относится ко всем видам иерархических данных, таких как справочники, планы видов характеристик, планы видов расчета и т. п.

Расчет итогов по нескольким полям

Часто требуется рассчитать итоги результата запроса по значению нескольких полей сразу. Для этого после ключевого слова *ПО* нужно перечислить через запятую список полей – контрольных точек.

Например, нам нужно вывести все данные из состава приходных накладных, в разрезе товаров и поставщиков, при этом рассчитать итоги по полям *Товар* и *Поставщик*.

Итоговые строки должны содержать для каждого товара общее количество и сумму его поступления по каждому поставщику, а также показатели поступлений по каждому поставщику в целом. Это можно сделать с помощью следующего запроса (листинг 1.60).

Листинг 1.60. Вывод всех записей из состава приходных накладных с итоговыми данными для каждого товара по каждому поставщику

```
ВЫБРАТЬ
  НакладнаяСостав.Ссылка.Поставщик КАК Поставщик,
  НакладнаяСостав.Товар КАК Товар,
  НакладнаяСостав.Количество КАК Количество,
  НакладнаяСостав.Цена КАК Цена,
  НакладнаяСостав.Сумма КАК Сумма
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК НакладнаяСостав
ИТОГИ
  СУММА(Количество),
  СУММА(Сумма)
ПО
  Поставщик,
  Товар
```

Обратите внимание, что в тексте запроса имена итоговых полей и контрольных точек принято располагать на разных строках со смещением относительно ключевых слов *ИТОГИ* и *ПО*.

В результате итоговые поля рассчитываются сначала для каждого товара по каждому поставщику и затем отдельно для каждого поставщика по всем товарам в целом (рис. 1.57).

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 24)

Поставщик	Товар	Количество	Цена	Сумма
Скороход АО		25		120 000
Скороход АО	Кроссовки	15		45 000
Скороход АО	Кроссовки	15	3 000	45 000
Скороход АО	Сапоги	5		50 000
Скороход АО	Сапоги	5	10 000	50 000
Скороход АО	Туфли	5		25 000
Скороход АО	Туфли	5	5 000	25 000
Животноводство ООО		200		10 400
Животноводство ООО	Масло	50		3 700
Животноводство ООО	Масло	20	80	1 600
Животноводство ООО	Масло	30	70	2 100
Животноводство ООО	Молоко	80		3 200
Животноводство ООО	Молоко	30	40	1 200
Животноводство ООО	Молоко	50	40	2 000
Животноводство ООО	Сметана	70		3 500
Животноводство ООО	Сметана	50	50	2 500
Животноводство ООО	Сметана	20	50	1 000
Корнет ЗАО		35		190 000
Корнет ЗАО	Кроссовки	20		60 000
Корнет ЗАО	Кроссовки	20	3 000	60 000
Корнет ЗАО	Сапоги	5		50 000
Корнет ЗАО	Сапоги	5	10 000	50 000
Корнет ЗАО	Туфли	10		80 000
Корнет ЗАО	Туфли	10	8 000	80 000

Рис. 1.57. Вывод всех записей из состава приходных накладных с итоговыми данными для каждого товара по каждому поставщику

Расчет общих итогов

Помимо итогов по контрольным точкам, можно задать расчет общих итогов результата запроса. Для этого после ключевого слова *ПО* необходимо указать ключевое слово *ОБЩИЕ*.

Например, нам нужно вывести все данные из состава приходных накладных для каждого поставщика и при этом рассчитать общие итоги и итоги по полю *Поставщик*. Итоговые строки должны содержать для каждого поставщика общее количество и сумму поставленных им товаров, а также эти же показатели по всем поступлениям в целом. Это можно сделать с помощью следующего запроса (листинг 1.61).

Листинг 1.61. Вывод всех записей из состава приходных накладных с итоговыми данными для каждого поставщика и в целом для всех поступлений

```
ВЫБРАТЬ
  НакладнаяСостав.Ссылка.Поставщик КАК Поставщик,
  НакладнаяСостав.Товар КАК Товар,
  НакладнаяСостав.Количество КАК Количество,
  НакладнаяСостав.Сумма КАК Сумма
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК НакладнаяСостав
ИТОГИ
  СУММА(Количество),
  СУММА(Сумма)
ПО
  ОБЩИЕ,
  Поставщик
```

В результате итоговые поля рассчитываются для каждого поставщика, и, кроме того, в результат запроса добавляется самая верхняя итоговая строка, содержащая итоги для всех поступлений в целом (рис. 1.58).

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 16)

Поставщик	Товар	Количество	Сумма
		260	320 400
Скороход АО		25	120 000
Скороход АО	Кроссовки	15	45 000
Скороход АО	Сапоги	5	50 000
Скороход АО	Туфли	5	25 000
Животноводство ООО		200	10 400
Животноводство ООО	Масло	20	1 600
Животноводство ООО	Молоко	30	1 200
Животноводство ООО	Сметана	50	2 500
Животноводство ООО	Сметана	20	1 000
Животноводство ООО	Масло	30	2 100
Животноводство ООО	Молоко	50	2 000
Корнет ЗАО		35	190 000
Корнет ЗАО	Туфли	10	80 000
Корнет ЗАО	Кроссовки	20	60 000
Корнет ЗАО	Сапоги	5	50 000

Рис. 1.58. Вывод всех записей из состава приходных накладных с итоговыми данными для каждого поставщика и в целом для всех поступлений

Итоги часто используются совместно с группировкой. В этом случае для итогов может быть не указан список агрегатных функций, так как он будет автоматически

формироваться из агрегатных полей списка выборки (листинг 1.62).

Листинг 1.62. Совместное применение расчета итогов и группировки записей

```
ВЫБРАТЬ
  НакладнаяСостав.Ссылка.Поставщик КАК Поставщик,
  НакладнаяСостав.Товар КАК Товар,
  СУММА(НакладнаяСостав.Количество) КАК Количество,
  СУММА(НакладнаяСостав.Сумма) КАК Сумма
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК НакладнаяСостав
СГРУППИРОВАТЬ ПО
  НакладнаяСостав.Ссылка.Поставщик,
  НакладнаяСостав.Товар
ИТОГИ ПО
  ОБЩИЕ,
  Поставщик
```

В данном запросе после ключевого слова *ИТОГИ* не указаны итоговые поля, но в результате запроса они рассчитываются, исходя из агрегатных функций в списке полей выборки.

В целом результат запроса практически такой же, как при выполнении предыдущего запроса (см. рис. 1.58), но количество записей в нем меньше, так как строки с одинаковыми значениями полей группировки (*Поставщик* и *Товар*) сворачиваются в одну запись (рис. 1.59).

Запрос: Документ.ПриходнаяНакладная.Состав (Записей в результате: 13)

Поставщик	Товар	Количество	Сумма
		260	320 400
Скороход АО		25	120 000
Скороход АО	Туфли	5	25 000
Скороход АО	Сапоги	5	50 000
Скороход АО	Кроссовки	15	45 000
Корнет ЗАО		35	190 000
Корнет ЗАО	Туфли	10	80 000
Корнет ЗАО	Сапоги	5	50 000
Корнет ЗАО	Кроссовки	20	60 000
Животноводство ООО		200	10 400
Животноводство ООО	Масло	50	3 700
Животноводство ООО	Молоко	80	3 200
Животноводство ООО	Сметана	70	3 500

Рис. 1.59. Совместное применение расчета итогов и группировки записей

Примеры использования выражений в списке полей выборки запроса

С помощью языка запросов можно решать широкий спектр задач благодаря тому, что в тексте запросов можно использовать разнообразные выражения: литералы, агрегатные функции, функции языка запросов, операции выбора и т. п.

В данном разделе мы разберем несколько примеров использования выражений в списке полей выборки запроса. Заметим сразу, что все эти примеры придуманы просто для демонстрации тех или иных возможностей языка запросов и никакого прикладного значения не имеют.

Выражение языка запросов может содержать:

- Литералы типов: число, строка (в кавычках), булево (значения *Истина* и *Ложь*), *Null*, *Неопределено*. Чтобы указать литерал типа *Дата*, можно воспользоваться ключевым словом языка запросов *ДАТАВРЕМЯ* или передать дату через параметр запроса.
- Арифметические операции (+, -, /, *). Операция получения остатка % в языке запросов не поддерживается.
- Операцию конкатенации строк (+). Операцию конкатенации нельзя использовать для виртуальных полей.
- Встроенные функции языка запросов (*ДЕНЬ*, *МЕСЯЦ*, *ГОД* и т. д.).
- Агрегатные функции (*СУММА*, *МИНИМУМ*, *МАКСИМУМ*, *СРЕДНЕЕ*, *КОЛИЧЕСТВО*).
- Операцию выбора *ВЫБОР* – позволяет получить одно из возможных значений в соответствии с указанными условиями.
- Операцию приведения типов *ВЫРАЗИТЬ* – позволяет привести значение составного типа к одному из составляющих это значение типов. А также функцию *ВЫРАЗИТЬ()* используют для получения результатов нужной длины и точности.

Рассмотрим несколько чисто демонстрационных задач, позволяющих показать применение выражений в списке выборки запросов.

Например, мы хотим вывести записи из иерархического справочника *Товары*. При этом после поля *Наименование* в скобках мы хотим видеть *Код* товара. Также для каждой записи справочника мы хотим видеть текстовое обозначение, является ли эта запись группой или нет («Это группа»/«Это элемент»). Это можно сделать с помощью следующего запроса (листинг 1.63).

Листинг 1.63. Примеры использования выражений в списке полей выборки

```
ВЫБРАТЬ
  Товары.Наименование + " (" + Товары.Код + ")" КАК Товар,
  ВЫБОР
    КОГДА Товары.ЭтоГруппа = ИСТИНА ТОГДА "Это группа"
    ИНАЧЕ "Это элемент"
  КОНЕЦ КАК ПризнакГруппы
ИЗ
  Справочник.Товары КАК Товары
```

Результат выполнения запроса представлен на рис. 1.60.

Запрос: Справочник.Товары (Записей в результате: 10)

Товар	ПризнакГруппы
Обувь (000000001)	Это группа
Туфли (000000002)	Это элемент
Сапоги (000000003)	Это элемент
Кроссовки (000000004)	Это элемент
Продукты (000000005)	Это группа
Масло (000000006)	Это элемент
Молоко (000000007)	Это элемент
Сметана (000000008)	Это элемент
Детская обувь (000000009)	Это группа
Пинетки (000000010)	Это элемент

Рис. 1.60. Примеры использования выражений в списке полей выборки

Для формирования поля выборки *Товар* в данном запросе используется операция конкатенации (сложения) строк, в результате наименование и код товара выводятся в одном текстовом поле выборки результата запроса.

Для формирования поля выборки *ПризнакГруппы* используется операция выбора (*ВЫБОР (КОГДА … ТОГДА) ИНАЧЕ … КОНЕЦ*). В этой операции после ключевого слова *КОГДА* записывается условие выбора, после ключевого слова *ТОГДА* следует значение поля выборки в случае, если условие истинно. В общем случае в операции выбора может указываться неограниченное количество альтернативных одиночных выборов *КОГДА … ТОГДА*. Значение выражения, указанного после слова *ИНАЧЕ* используется в качестве результата операции выбора в том случае, если ни одно из ранее указанных альтернативных условий выбора не было выполнено.

В операции выбора проверяется значение логического поля *Товары.ЭтоГруппа*, и результат выбора выводится как строковой («Это группа»/«Это элемент») литерал.

Рассмотрим следующую задачу. Допустим, мы хотим вывести даты начала и окончания поступлений товаров и узнать, сколько дней прошло между ними. Это можно сделать с помощью следующего запроса (листинг 1.64).

Листинг 1.64. Примеры использования выражений в списке полей выборки

```
ВЫБРАТЬ
  МИНИМУМ(Накладная.Дата) КАК Начало,
  МАКСИМУМ(Накладная.Дата) КАК Окончание,
  РАЗНОСТЬДАТ(МИНИМУМ(Накладная.Дата), МАКСИМУМ(Накладная.Дата), ДЕНЬ) КАК Период
ИЗ
  Документ.ПриходнаяНакладная КАК Накладная
```

Результат выполнения запроса представлен на рис. 1.61.

Запрос: Документ.ПриходнаяНакладная (Записей в результате: 1)

Начало	Окончание	Период
17.10.2012 12:00:00	05.11.2012 12:00:00	19

Рис. 1.61. Примеры использования выражений в списке полей выборки

Для формирования полей выборки *Начало* и *Окончание* используются агрегатные

функции *МИНИМУМ()* и *МАКСИМУМ()*. В данном запросе группировка записей не используется, поэтому значения агрегатных функций вычисляются для всех записей запроса, и результат запроса составляет одна строка. В данном случае вычисляются минимальная и максимальные даты приходных накладных.

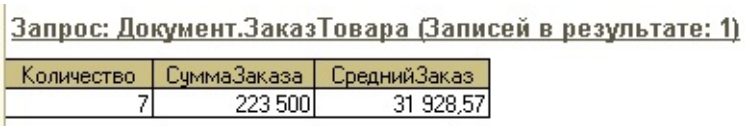
Для формирования поля выборки *Период* используется функция *РАЗНОСТЬДАТ()* с параметром *ДЕНЬ*, которая вычисляет, сколько прошло дней между началом и окончанием поступлений товаров.

Рассмотрим следующую задачу. Допустим, мы хотим вывести общее количество и сумму всех заказов товара и узнать среднюю сумму одного заказа. Это можно сделать с помощью следующего запроса (листинг 1.65).

Листинг 1.65. Примеры использования выражений в списке полей выборки

```
ВЫБРАТЬ
  КОЛИЧЕСТВО(*) КАК Количество,
  СУММА(ЗаказТовара.СуммаЗаказа) КАК СуммаЗаказа,
  ВЫРАЗИТЬ(СУММА(ЗаказТовара.СуммаЗаказа) / КОЛИЧЕСТВО(*) КАК ЧИСЛО(8,2)) КАК СреднийЗаказ
ИЗ
  Документ.ЗаказТовара КАК ЗаказТовара
```

Результат выполнения запроса представлен на рис. 1.62.



Запрос: Документ.ЗаказТовара (Записей в результате: 1)

Количество	СуммаЗаказа	СреднийЗаказ
7	223 500	31 928,57

Рис. 1.62. Примеры использования выражений в списке полей выборки

Для формирования поля выборки *Количество* используется агрегатная функция *КОЛИЧЕСТВО()*. В данном запросе группировка записей не используется, поэтому значения агрегатных функций вычисляются для всех записей запроса, и результат запроса составляет одна строка. В данном случае выводится общее количество всех заказов товара.

Для формирования поля выборки *СуммаЗаказа* используется агрегатная функция *СУММА()*. В результате вычисляется общая сумма всех заказов.

Для формирования поля выборки *СреднийЗаказ* сумма всех заказов (значение поля *СуммаЗаказа*) делится на общее количество заказов клиентов (значение поля *Количество*). Затем к результату деления применяется функция *ВЫРАЗИТЬ()*, с помощью которой получается результат заданной длины и точности (в данном случае результат деления округляется до двух десятых). Таким образом, мы можем узнать среднюю сумму заказа.

Обратите внимание (см. листинг 1.64, 1.65), что в качестве параметров функций должны указываться имена полей, а не их псевдонимы, и результат предыдущих вычислений с

помощью псевдонима нельзя использовать при расчете других полей.

Рассмотрим следующую задачу. Допустим, мы хотим вывести движения регистра накопления *ОстаткиТоваров* за определенный период. Причем дату движений мы хотим видеть на начало текущей недели, а в качестве контрагента мы хотим вывести поставщиков из приходных накладных и покупателей из расходных накладных. Это можно сделать с помощью следующего запроса (листинг 1.66).

Листинг 1.66. Примеры использования выражений в списке полей выборки

```
ВЫБРАТЬ
  НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, НЕДЕЛЯ) КАК Период,
  ВЫБОР
    КОГДА (ОстаткиТоваров.Регистратор ССЫЛКА Документ.ПриходнаяНакладная)
    ТОГДА ВЫРАЗИТЬ (ОстаткиТоваров.Регистратор КАК Документ.ПриходнаяНакладная).Поставщик
    КОГДА (ОстаткиТоваров.Регистратор ССЫЛКА Документ.РасходнаяНакладная)
    ТОГДА ВЫРАЗИТЬ (ОстаткиТоваров.Регистратор КАК Документ.РасходнаяНакладная).Покупатель
  КОНЕЦ КАК Контрагент
ИЗ
  РегистрНакопления.ОстаткиТоваров КАК ОстаткиТоваров
ГДЕ
  ОстаткиТоваров.Период МЕЖДУ &ДатаНачала И &ДатаОкончания
```

Результат выполнения запроса представлен на рис. 1.63.

Запрос: РегистрНакопления.ОстаткиТоваров (Записей в результате: 15)

Период	Контрагент
22.10.2012 0:00:00	Животноводство ООО
22.10.2012 0:00:00	Животноводство ООО
22.10.2012 0:00:00	Животноводство ООО
29.10.2012 0:00:00	Животноводство ООО
29.10.2012 0:00:00	Животноводство ООО
29.10.2012 0:00:00	Животноводство ООО
29.10.2012 0:00:00	Маслова Ирина Николаевна
29.10.2012 0:00:00	Маслова Ирина Николаевна
29.10.2012 0:00:00	Маслова Ирина Николаевна
29.10.2012 0:00:00	Маслова Ирина Николаевна
29.10.2012 0:00:00	Маслова Ирина Николаевна
29.10.2012 0:00:00	Маслова Ирина Николаевна
05.11.2012 0:00:00	Корнет ЗАО
05.11.2012 0:00:00	Корнет ЗАО
05.11.2012 0:00:00	Корнет ЗАО

Рис. 1.63. Примеры использования выражений в списке полей выборки

В данном запросе накладывается условие на нахождение периода движений регистра накопления *ОстаткиТоваров* в интервале между двумя датами. В условии отбора используется оператор *МЕЖДУ*, который проверяет результат вхождения значения в диапазон вместе с границами диапазона.

Для формирования поля выборки *Период* используется функция *НАЧАЛОПЕРИОДА()* с параметром *НЕДЕЛЯ*, позволяющая представить дату движений регистра как начало недели, за которую производились эти движения.

Мы знаем, что регистраторами регистра накопления *ОстаткиТоваров* являются

документы *ПриходнаяНакладная* и *РасходнаяНакладная*, следовательно, поле *Регистратор* регистра накопления является полем составного типа.

Для формирования поля выборки *Контрагент* используется операция выбора, в условии которой при помощи оператора *ССЫЛКА* проверяется, ссылкой на какой документ является поле *Регистратор* регистра накопления. Затем с помощью функции *ВЫРАЗИТЬ()* значение поля составного типа приводится к одному из составляющих это значение типов, но из приходных накладных выводится поле *Поставщик*, а из расходных накладных – поле *Покупатель*.

Применение функции *ВЫРАЗИТЬ()* для приведения значений составного типа к какому-либо определенному типу весьма желательно, так как позволяет более оптимально выполнить запрос.

подробнее

Раздел «Ограничить получение данных через точку от полей составного ссылочного типа».

Иногда функцию *ВЫРАЗИТЬ()* пытаются использовать не по назначению – для преобразования типов, например строки, в число и т. п. Это неправильно. Функция *ВЫРАЗИТЬ()* используется только для тех случаев, о которых было рассказано выше (листинг 1.65, 1.66).

Выражения в языке запросов используются не только в списке полей выборки, но и в условиях отбора, в предложении упорядочивания, при описании итогов и т. д. Выражения в условиях отбора мы уже рассматривали ранее в разделе "Как получить записи из таблицы, отобранные по некоторому условию", другие примеры использования выражений в языке запросов будут рассмотрены нами позднее.

Подробнее

Документация «1С:Предприятие 8.3. Руководство разработчика», раздел 8.2 «Язык запросов», а также встроенная справка *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Синтаксис текста запросов > Использование выражений в языке запросов*.

Примеры использования языка запросов для получения данных из нескольких таблиц

Как использовать данные одного запроса внутри другого запроса

Иногда внутри одного запроса необходимо использовать данные другого запроса. Например, требуется ограничить выборку значений в условии отбора одного запроса данными другого запроса. Второй запрос по отношению к первому является *вложенным*, а первый запрос по отношению ко второму является основным или *внешним*. Причем уровней вложенности запросов друг в друга в общем случае может быть несколько.

Рассмотрим эту ситуацию на примере. Допустим, необходимо узнать, в каких расходных

накладных и каким покупателям продавались товары, перечисленные в составе конкретной приходной накладной. Для этого нужно выполнить следующий запрос (листинг 1.67).

Листинг 1.67. Использование вложенного запроса в условии отбора основного запроса

```
ВЫБРАТЬ
  Расход.Ссылка КАК Документ,
  Расход.Покупатель КАК Покупатель
ИЗ
  Документ.РасходнаяНакладная КАК Расход
ГДЕ
  Расход.Состав.Товар В
    (
    ВЫБРАТЬ
      ПриходСостав.Товар КАК Товар
    ИЗ
      Документ.ПриходнаяНакладная.Состав КАК ПриходСостав
    ГДЕ
      ПриходСостав.Ссылка = &Документ
    )
```

В условии отбора (*Расход.Состав.Товар В ()*) значение поля *Товар* из табличной части *Состав* расходной накладной проверяется на попадание в перечень возможных значений. Для получения этого набора значений используется вложенный запрос. Описание вложенного запроса начинается с предложения *ВЫБРАТЬ* и ничем не отличается от обычного запроса.

Обратите внимание, что текст вложенного запроса принято располагать со смещением относительно основного запроса для повышения наглядности и структурированности запроса в целом.

Выполним в консоли запросов отдельно сам вложенный запрос. Для этого необязательно писать заново текст запроса (листинг 1.68). Достаточно просто выделить мышью текст вложенного запроса и нажать кнопку *Выполнить*.

Листинг 1.68. Описание вложенного запроса

```
ВЫБРАТЬ
  ПриходСостав.Товар КАК Товар
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК ПриходСостав
ГДЕ
  ПриходСостав.Ссылка = &Документ
```

Как мы видим, вложенный запрос является запросом к табличной части документа, который мы рассматривали в разделе «Как получить данные из табличной части некоторого документа».

В результате будет получен перечень товаров из табличной части приходной накладной,

выбранной в качестве значения параметра *Документ* (см. рис. 1.64, верхняя таблица).

При выполнении внешнего запроса сначала выполняется вложенный запрос, и результат его выполнения подставляется в условие отбора внешнего запроса.

В результате выполнения основного запроса (листинг 1.67) мы увидим список расходных накладных, в которых продавались товары, перечисленные в составе конкретной приходной накладной, указанной в параметре *Документ* (рис. 1.64).

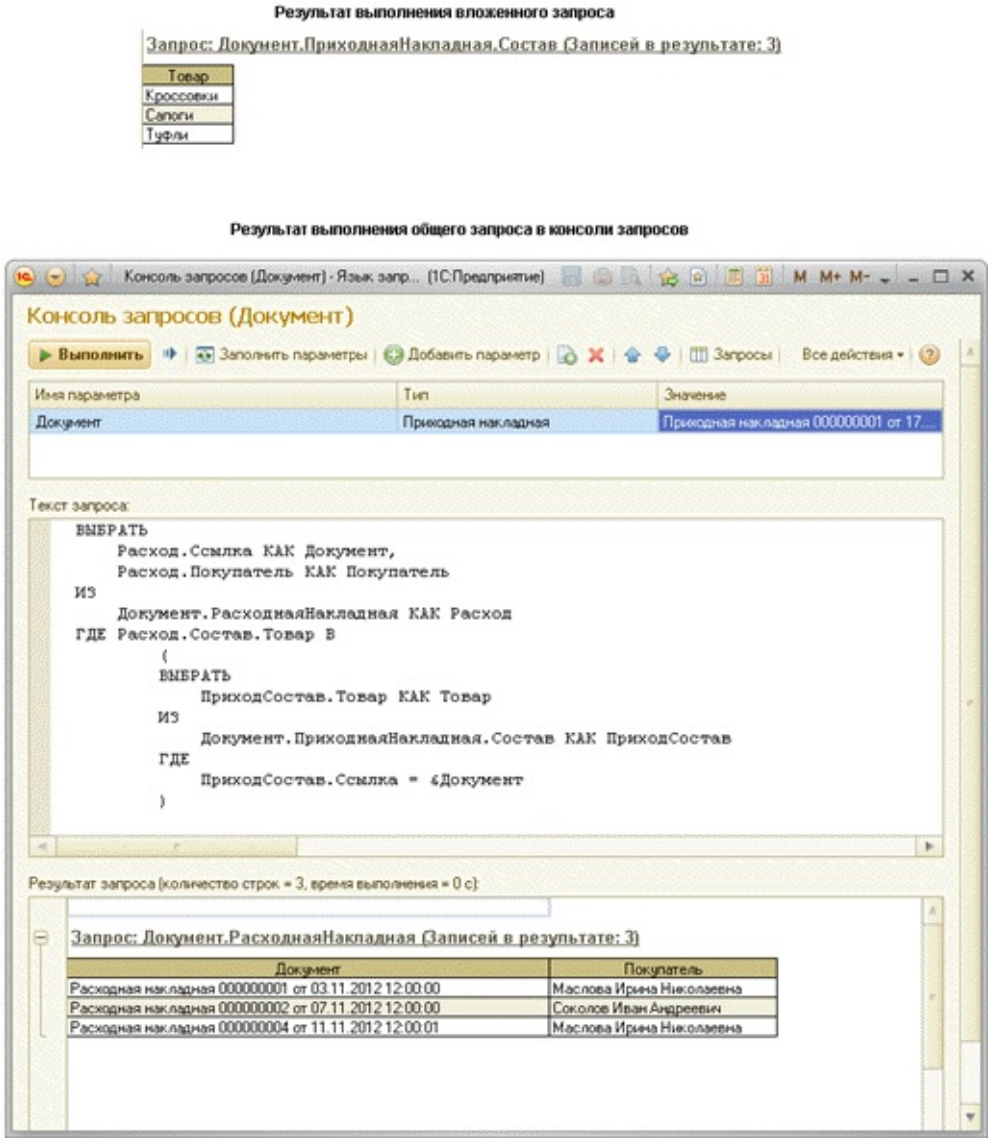


Рис. 1.64. Использование вложенного запроса в условии отбора

Таким же образом вложенные запросы могут использоваться в условии отбора, передаваемого в качестве параметра в виртуальную таблицу. В этом случае при формировании виртуальной таблицы на основе исходной (реальной) таблицы базы данных выборка значений из исходной таблицы будет ограничена набором значений, полученным в результате выполнения вложенного запроса.

Например, виртуальная таблица *РегистрСведений.Цены.СрезПоследних* возвращает срез последних записей регистра сведений *Цены* по каждому товару на определенную дату. Эта таблица имеет параметры *Период* и *Условие*, которые мы можем использовать

в языке запросов.

Подробнее

Посмотреть состав таблиц запросов можно во встроенной справке *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Таблицы запросов*.

В параметре *Период* мы можем задать дату, на которую должен быть выполнен срез последних записей регистра сведений. В параметре *Условие* мы можем задать условие отбора записей из исходной (реальной) таблицы регистра сведений при формировании виртуальной таблицы.

Допустим, нам нужно увидеть последние установленные цены товаров на сегодняшнюю дату. Но не на все товары, а только на те, которые перечислены в составе конкретной расходной накладной. Для этого нужно выполнить следующий запрос (листинг 1.69).

Листинг 1.69. Использование вложенного запроса в условии отбора, передаваемого в параметр виртуальной таблицы

```
ВЫБРАТЬ
  Цены.Период,
  Цены.Товар,
  Цены.Цена
ИЗ
  РегистрСведений.Цены.СрезПоследних( , Товар В
    (
    ВЫБРАТЬ
      РасходСостав.Товар КАК Товар
    ИЗ
      Документ.РасходнаяНакладная.Состав КАК РасходСостав
    ГДЕ
      РасходСостав.Ссылка = &Документ
    )
  ) КАК Цены
УПОРЯДОЧИТЬ ПО
  Период
```

В данном запросе нет ничего необычного, кроме того, что после имени виртуальной таблицы (после ключевого слова *ИЗ*) в скобках мы можем задать параметры для отбора записей регистра сведений в эту таблицу. Поскольку мы хотим увидеть самые последние цены на товары из регистра сведений, то параметр *Период* мы не указываем. В параметре *Условие* мы можем задать произвольное условие на языке запросов, использующее любые поля регистра сведений *Цены*.

В этом условии (*Товар В ()*) значение поля регистра сведений *Товар* проверяется на попадание в перечень возможных значений. Для получения этого набора значений используется вложенный запрос. Таким образом, при формировании виртуальной таблицы из исходной (реальной) таблицы регистра сведений *Цены* будут отбираться записи об изменении цен только тех товаров, которые перечислены в составе расходной накладной, выбранной в качестве значения параметра *Документ*.

В результате выполнения запроса мы увидим последние установленные цены на товары, перечисленные в составе конкретной расходной накладной (рис. 1.65).

Запрос: РегистрСведений.Цены.СрезПоследних((Записей в результате: 3)

Период	Товар	Цена
15.10.2012 0:00:00	Кроссовки	3 000
01.11.2012 0:00:00	Туфли	8 000
05.11.2012 0:00:00	Сапоги	10 500

Рис. 1.65. Использование вложенного запроса в условии отбора, передаваемого в параметр виртуальной таблицы

подробнее

В данном примере мы используем виртуальную таблицу *СрезПоследних* регистра сведений *Цены*. Подробнее о назначении и использовании виртуальных таблиц регистра сведений будет рассказано в разделе «Получение данных из периодических регистров сведений».

Заметим, что пример с использованием вложенного запроса в условии отбора, передаваемого в параметр виртуальной таблицы, показан в чисто демонстрационных целях, но на практике в таких ситуациях лучше использовать временные таблицы, о которых будет рассказано ниже.

подробнее

«Временные таблицы и пакетные запросы».

Как получить данные из разных таблиц для одного и того же поля

Часто в результате запроса требуется вывести данные из разных таблиц, связав их по значению некоторого поля, то есть какие-то поля вывести из одной таблицы, какие-то – из другой для одного и того же значения поля из этих таблиц. Например, можно вывести для одного и того же товара количество его поступлений и продаж.

В этом случае используются несколько источников данных, которые перечисляются после ключевого слова *ИЗ*. В качестве источников запроса могут выступать реальные и виртуальные таблицы, а также вложенные запросы, но мы пока для упрощения примера будем рассматривать ситуацию, когда в качестве источников запроса выступают две таблицы базы данных. Суть примера от этого не изменится.

Исходные таблицы запроса обычно связываются (соединяются) между собой по некоторому условию – условию связи. Поле, по которому производится связь, обычно имеет ссылочный тип. При соединении данных из исходных таблиц запроса для каждой записи из этих таблиц проверяется условие равенства значений ссылочных полей этого типа. Условие связи источников запроса задается в предложении *ИЗ*, после ключевого слова *ПО*. Например, *ПО Товары.Ссылка = Цены.Товар*.

В зависимости от того, должны ли записи каждой из таблиц удовлетворять условию связи, таблицы соединяются *внутренним*, *левым внешним*, *правым внешним* или *полным внешним соединением*. Ключевые слова, определяющие тип соединения (например, *ЛЕВОЕ СОЕДИНЕНИЕ*), располагаются между именами таблиц в

предложении *ИЗ*.

Рассмотрим пример, который позволит понять сущность каждого типа соединения. Допустим, мы хотим вывести перечень товаров из справочника *Товары* и при этом показать для каждого товара последнюю установленную на него цену.

В нашей демонстрационной конфигурации существует регистр сведений *Цены*, который хранит информацию об изменении цен товаров во времени. Виртуальная таблица *СрезПоследних* этого регистра сведений позволяет получить информацию о последних ценах, установленных для каждого товара.

Поэтому для выполнения поставленной задачи нам нужно соединить в запросе данные из таблиц *Справочник.Товары* и *РегистрСведений.Цены.СрезПоследних*.

подробнее

В данном примере для большей наглядности мы используем виртуальную таблицу *СрезПоследних* регистра сведений *Цены*. Подробнее о назначении и использовании виртуальных таблиц регистра сведений будет рассказано в разделе «Получение данных из периодических регистров сведений».

Регистр сведений *Цены* содержит поле *Товар*, ссылающееся на справочник *Товары*. По полю этого типа и будет производиться связь таблиц в запросе, то есть в условии связи для каждой записи из обеих таблиц будут сравниваться значения уникального поля *Ссылка* из справочника и ссылочного поля *Товар* из среза последних записей регистра сведений.

Исходные данные, используемые в примере о соединении таблиц в запросе, представлены на рис. 1.66.

Справочник "Товары" (10 записей)

Код	Наименование
000000001	Обувь
000000002	Туфли
000000003	Сапоги
000000004	Кроссовки
000000005	Продукты
000000006	Масло
000000007	Молоко
000000008	Сметана
000000009	Детская обувь
000000010	Пинетки

Срез последних записей регистра сведений "Цены" (6 записей)

Товар	Период	Цена
Туфли	01.11.2012 0:00:00	8 000
Сапоги	05.11.2012 0:00:00	10 500
Кроссовки	15.10.2012 0:00:00	3 000
Масло	01.11.2012 0:00:00	80
Молоко	05.11.2012 0:00:00	45
Сметана	20.10.2012 0:00:00	50

Рис. 1.66. Исходные записи справочника «Товары» и среза последних записей регистра сведений «Цены»

Однако, как мы видим на рисунке, не для всех товаров из справочника *Товары* существует соответствующая им цена в срезе последних записей регистра сведений *Цены*. В частности, для товара *Пинетки* не найдено цены в регистре сведений, так как товар – новый, и цена на него еще не устанавливалась. Также для записей справочника *Обувь*, *Продукты* и *Детская обувь* не найдено соответствий в регистре сведений, так

как эти записи являются группами товаров, и цены на них не имеют смысла.

Теперь посмотрим, что же произойдет при связывании данных двух этих таблиц разными видами соединений.

Внутреннее соединение

При внутреннем соединении таблиц в результат запроса попадут только те записи из таблиц-источников, которые удовлетворяют заданному условию связи (после ключевого слова *ПО*).

Предположим, нам необходимо вывести из справочника *Товары* те товары, на которые установлена цена в регистре сведений *Цены*, и показать актуальную цену для каждого товара. Товары, на которые не установлена цена, выводить не нужно, а также не нужно выводить последние установленные цены из регистра сведений для товаров, которых не существует в справочнике товаров.

Для этого нам нужно соединить в запросе данные из таблиц *Справочник.Товары* и *РегистрСведений.Цены.СрезПоследних* внутренним соединением. Из справочника *Товары* выведем поля *Код*, *Наименование* и *Производитель*, а из среза последних записей регистра сведений *Цены* – поле *Цена* (листинг 1.70).

Листинг 1.70. Внутреннее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
  Справочник.Товары КАК Товары
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК Цены
    ПО Товары.Ссылка = Цены.Товар
```

Результат запроса представлен на рис. 1.67.

Запрос: Справочник.Товары (Записей в результате: 6)

Код	Наименование	Производитель	Цена
000000002	Туфли	Германия	8 000
000000003	Сапоги	Италия	10 500
000000004	Кроссовки		3 000
000000006	Масло	Россия	80
000000007	Молоко	Россия	45
000000008	Сметана		50

Рис. 1.67. Внутреннее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

ПРИМЕЧАНИЕ

При соединениях источников запроса в заголовке таблицы, содержащей результат запроса, в консоли запросов выводится имя первой таблицы из списка соединений.

Мы видим, что в результат запроса попали только те записи из обеих таблиц, для которых было выполнено условие связи (*Товары.Ссылка = Цены.Товар*), то есть для каждой записи из справочника *Товары* была найдена соответствующая запись из среза последних записей регистра сведений *Цены*, в которой значение ссылочного поля *Товар* было равно значению поля *Ссылка*. Поэтому в результате запроса содержатся 6 записей, а в исходной таблице справочника – 10 (см. рис. 1.68), так как для четырех записей справочника не было найдено соответствий в регистре сведений.

В обратную сторону условие связи также должно выполняться, то есть если в справочнике товаров не существует товара, для которого установлена цена в регистре сведений, то такая запись также не попадет в результат запроса. Но обычно разработчики программно поддерживают целостность базы данных, и такие ситуации встречаются редко.

ПРИМЕЧАНИЕ

Ключевое слово *ВНУТРЕННЕЕ* в тексте запроса можно не указывать, но оно повышает наглядность и читаемость текста запроса.

При соединениях чаще всего применяется условие на равенство, но в языке запросов есть возможность использовать все операции сравнения и логические операции *И*, *ИЛИ*, *НЕ*.

Если в таблице будет найдено несколько записей, удовлетворяющих условию соединения, то в результат запроса будут включены все эти записи. Например, если в запросе использовать соединение таблицы справочника *Товары* и реальной таблицы регистра сведений *Цены* (листинг 1.71), то в результате мы увидим несколько установленных цен для одного и того же товара (рис. 1.68).

Листинг 1.71. Внутреннее соединение данных справочника «Товары» и регистра сведений «Цены»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена,
  Цены.Период
ИЗ
  Справочник.Товары КАК Товары
    СОЕДИНЕНИЕ РегистрСведений.Цены КАК Цены
    ПО Товары.Ссылка = Цены.Товар
УПОРЯДОЧИТЬ ПО
  Товары.Наименование
```

Запрос: Справочник.Товары (Записей в результате: 10)

Код	Наименование	Производитель	Цена	Период
000000004	Кроссовки		3 000	15.10.2012 0:00:00
000000006	Масло	Россия	70	20.10.2012 0:00:00
000000006	Масло	Россия	80	01.11.2012 0:00:00
000000007	Молоко	Россия	40	20.10.2012 0:00:00
000000007	Молоко	Россия	45	05.11.2012 0:00:00
000000003	Сапоги	Италия	10 000	15.10.2012 0:00:00
000000003	Сапоги	Италия	10 500	05.11.2012 0:00:00
000000008	Сметана		50	20.10.2012 0:00:00
000000002	Туфли	Германия	7 000	15.10.2012 0:00:00
000000002	Туфли	Германия	8 000	01.11.2012 0:00:00

Рис. 1.68. Внутреннее соединение данных справочника «Товары» и регистра сведений «Цены»

Левое внешнее соединение

Если связать таблицы регистров накопления левым соединением, то в результат запроса попадут записи из обеих таблиц, удовлетворяющие условию связи, и, кроме того, записи из первой таблицы, расположенной слева от ключевого слова *СОЕДИНЕНИЕ*, для которых не найдено соответствия во второй таблице.

Предположим, нам необходимо вывести все записи из справочника *Товары* и показать актуальную цену товаров, на которые установлена цена в регистре сведений *Цены*. При этом последние установленные цены из регистра сведений для товаров, которых не существует в справочнике товаров, выводить не нужно. Для этого нам нужно соединить в запросе данные из таблиц *Справочник.Товары* и *РегистрСведений.Цены.СрезПоследних* левым внешним соединением (листинг 1.72).

Листинг 1.72. Левое внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК Цены
    ПО Товары.Ссылка = Цены.Товар
```

Результат запроса представлен на рис. 1.69.

Запрос: Справочник.Товары (Записей в результате: 10)

Код	Наименование	Производитель	Цена
000000001	Обувь		
000000002	Туфли	Германия	8 000
000000003	Сапоги	Италия	10 500
000000004	Кроссовки		3 000
000000005	Продукты		
000000006	Масло	Россия	80
000000007	Молоко	Россия	45
000000008	Сметана		50
000000009	Детская обувь		
000000010	Пинетки		

Рис. 1.69. Левое внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

Мы видим, что в результат запроса попали все 10 записей из *Товары* (см. рис. 1.66) независимо от того, была найдена соответствующая им запись из среза последних записей регистра сведений *Цены* или нет. Так произошло потому, что таблица *Справочник.Товары* расположена слева от слова *СОЕДИНЕНИЕ*.

Строки результата запроса, для которых не найдено соответствующих условию записей из второго источника, будут содержать значение *NULL* в полях, формируемых на основании записей из этого источника. Обратите внимание, что значения *NULL* не являются нулем или пустой строкой. Значения данного типа обозначают неуказанные (отсутствующие) значения или значения, не имеющие смысла. В данном случае в поле *Цена* для товаров *Обувь*, *Продукты*, *Детская обувь* и *Пинетки* будет находиться значение *NULL*.

Из второй таблицы в результат запроса попадут только записи, удовлетворяющие условию связи, то есть если в справочнике товаров не существует товара, для которого установлена цена в регистре сведений, то такая запись не попадет в результат запроса.

ПРИМЕЧАНИЕ

Ключевое слово *ВНЕШНЕЕ* в тексте запроса можно не указывать, но оно повышает наглядность и читаемость текста запроса.

Таким образом, в случае, когда таблица справочника связывается левым внешним соединением с таблицей по полю, имеющему тип ссылки на этот справочник, вы можете быть уверены, что не потеряете нужных вам данных. Поэтому в отчетах используется в основном именно левое соединение таблиц.

Правое внешнее соединение

При правом соединении таблиц ситуация зеркально противоположная. То есть из правой таблицы в результат запроса попадут все записи, а из левой – только те, которые удовлетворяют условию связи.

Предположим, нам необходимо вывести из справочника *Товары* те товары, на которые установлена цена в регистре сведений *Цены*, и показать актуальную цену для каждого товара. При этом товары, на которые не установлена цена, выводить не нужно, а из регистра сведений нужно вывести срез всех последних записей. Для этого нам нужно соединить в запросе данные из таблиц *Справочник.Товары* и *РегистрСведений.Цены.СрезПоследних* правым внешним соединением (листинг 1.73).

Листинг 1.73. Правое внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
```

```
  Справочник.Товары КАК Товары
    ПРАВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК Цены
    ПО Товары.Ссылка = Цены.Товар
```

Результат запроса представлен на рис. 1.70.

Запрос: Справочник.Товары (Записей в результате: 6)

Код	Наименование	Производитель	Цена
000000002	Туфли	Германия	8 000
000000003	Сапоги	Италия	10 500
000000004	Кроссовки		3 000
000000006	Масло	Россия	80
000000007	Молоко	Россия	45
000000008	Сметана		50

Рис. 1.70. Правое внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

Мы видим, что в результат запроса попали все 6 записей из таблицы среза последних записей регистра сведений *Цены* (см. рис. 1.66), расположенной справа от слова *СОЕДИНЕНИЕ*. В нашем примере это не так очевидно: так, все товары, для которых установлены цены, присутствуют в справочнике *Товары*. Но в общем случае если в справочнике товаров не существует товара, для которого установлена цена в регистре сведений, то такая запись все равно попадет в результат запроса.

Строки результата запроса, для которых не найдено соответствующих условию записей из первого источника, будут содержать значение *NULL* в полях, формируемых на основании записей из этого источника.

Из первой таблицы в результат запроса попадут только записи, удовлетворяющие условию связи. То есть если в срезе последних записей регистра сведений *Цены* не найдена цена для какого-то товара, то такая запись из справочника *Товары* не попадет в результат запроса.

Правое соединение полностью аналогично левому, но при этом таблицы меняются местами в тексте запроса. Именно так и происходит в «1С:Предприятии».

Таким образом, если мы поменяем местами наши исходные таблицы и свяжем их правым соединением (первый вариант запроса), то мы получим результат, полностью аналогичный предыдущему примеру с левым соединением (см. рис. 1.69), листинг 1.74.

Листинг 1.74. Правое внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

```
// Первый вариант
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
  РегистрСведений.Цены.СрезПоследних КАК Цены
```

```
    ПРАВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ Справочник.Товары КАК Товары
    ПО Товары.Ссылка = Цены.Товар

// Второй вариант
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК Цены
    ПО Товары.Ссылка = Цены.Товар
```

При этом порядок следования полей таблиц в условии связи значения не имеет, важно лишь, какая таблица расположена слева, а какая – справа от слова *СОЕДИНЕНИЕ*.

Полное соединение

При полном соединении таблиц в результат запроса будут включены записи из обеих исходных таблиц, которые соответствуют указанному условию. Кроме того, в результат запроса будут включены также еще и те записи из обоих источников, для которых не найдено соответствий.

Предположим, нам необходимо вывести все записи из справочника *Товары* и срез всех последних записей из регистра сведений *Цены*. При этом нужно показать актуальную цену для тех товаров, на которые она устанавливалась. Для этого нам нужно соединить в запросе данные из таблиц *Справочник.Товары* и *РегистрСведений.Цены.СрезПоследних* полным внешним соединением (листинг 1.75).

Листинг 1.75. Полное внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
  Справочник.Товары КАК Товары
    ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК Цены
    ПО Товары.Ссылка = Цены.Товар
```

Результат запроса представлен на рис. 1.71.

Запрос: Справочник.Товары (Записей в результате: 10)

Код	Наименование	Производитель	Цена
000000001	Обувь		
000000002	Туфли	Германия	8 000
000000003	Сапоги	Италия	10 500
000000004	Кроссовки		3 000
000000005	Продукты		
000000006	Масло	Россия	80
000000007	Молоко	Россия	45
000000008	Сметана		50
000000009	Детская обувь		
000000010	Пинетки		

Рис. 1.71. Полное внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

Мы видим, что в результат запроса попали все записи из исходных таблиц запроса. Но в нашем частном случае мы получили результат, полностью аналогичный примеру с левым соединением (см. рис. 1.69), так как все товары, для которых установлены цены в регистре сведений, присутствуют в справочнике *Товары*. Для тех товаров, на которые устанавливалась цена, она найдена в срезе последних записей регистра сведений и отражена в результате запроса.

Строки результата запроса, для которых не найдено соответствующих условию записей из какого-либо источника, будут содержать значение *NULL* в полях, формируемых на основании записей из этого источника.

Следует иметь в виду, что при работе в клиент-серверном варианте, когда в качестве СУБД используется *PostgreSQL*, производительность выполнения запросов с конструкцией *ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ* значительно снижается. В особенности это касается случаев, когда в запросе встречаются две и более такие конструкции. Поэтому в общем случае не рекомендуется использовать полное внешнее соединение в запросах. И в тех случаях, где это возможно, рекомендуется переписать текст исходного запроса без использования этой конструкции.

Выше мы рассматривали различные виды соединения источников запроса на примере двух таблиц базы данных. В общем случае в запросе может содержаться не только одно соединение (двух источников), но и несколько соединений нескольких источников сразу. Об этом рассказывается в следующем примере.

Кроме того, в соединении источников могут участвовать и вложенные запросы. Однако соединения с вложенными запросами крайне нежелательны, так как в большинстве случаев могут привести к неоптимальному выполнению запроса.

Подробнее

Раздел «Не использовать соединения с вложенными запросами и с виртуальными таблицами».

Как получить данные из разных таблиц, связанных несколькими соединениями

Рассмотрим теперь более сложную задачу, когда в результате запроса используется

несколько соединений данных из нескольких таблиц, и проанализируем подробно каждый этап такого соединения. Предположим, нам необходимо вывести перечень товаров из справочника *Товары* и при этом показать остаток каждого товара и количество его продаж.

В нашей демонстрационной конфигурации существует регистр накопления *ОстаткиТоваров*, который накапливает информацию о приходе и расходе товаров. Виртуальная таблица *Остатки* этого регистра накопления позволяет получить информацию об остатках каждого товара. Также в конфигурации существует регистр накопления *Продажи*, который накапливает информацию о продажах товаров. Виртуальная таблица *Обороты* этого регистра накопления позволяет получить информацию о количестве и сумме оборотов (продаж) каждого товара.

Поэтому для выполнения поставленной задачи нам нужно соединить в запросе данные из таблиц *Справочник.Товары*, *РегистрНакопления.ОстаткиТоваров.Остатки* и *РегистрНакопления.Продажи.Обороты*.

подробнее

В данном примере для большей наглядности мы используем виртуальную таблицу ***Остатки*** регистра накопления ***ОстаткиТоваров*** и виртуальную таблицу ***Обороты*** регистра накопления ***Продажи***. Подробнее о назначении и использовании виртуальных таблиц регистра накопления будет рассказано в разделах «Получение остатков», «Получение оборотов».

Регистры накопления *ОстаткиТоваров* и *Продажи* содержат поле *Товар*, ссылающееся на справочник *Товары*. По полю этого типа и будет производиться связь таблиц в запросе, то есть в условии связи для каждой записи из исходных таблиц будут сравниваться значения уникального поля *Ссылка* из справочника и ссылочного поля *Товар* из виртуальных таблиц *Остатки* или *Обороты* регистров накопления.

Исходные данные, используемые в примере о соединении таблиц в запросе, представлены на рис. 1.72.

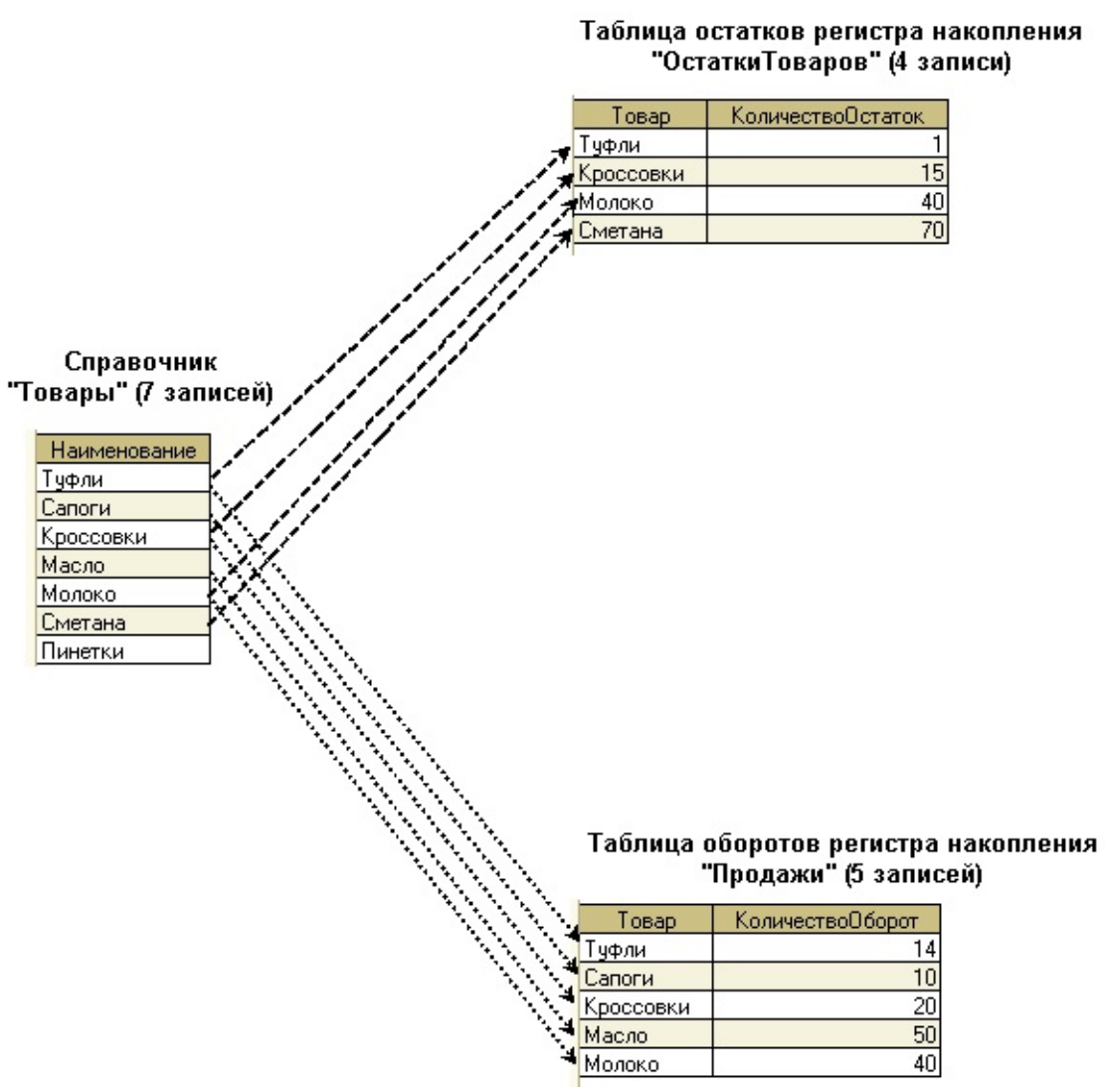


Рис. 1.72. Исходные записи справочника «Товары» и виртуальных таблиц регистров накопления «ОстаткиТоваров» и «Продажи»

На рис. 1.72 мы видим, что в справочнике *Товары* отражаются семь записей, которые не являются группой, так как на таблицу справочника наложено соответствующее условие. Также мы видим, что записи всех трех таблиц, используемых в примере, не полностью соответствуют друг другу по полю *Товар*. Так, товар *Пинетки* ни разу не продавался и не поступал (не имеет остатка), т. к. он новый. Товар *Сметана* имеет остаток, но он ни разу не продавался. Товары *Сапоги* и *Масло* полностью проданы, т. е. они не имеют остатка. Эти особенности будут отражены в результате запроса, когда мы будем связывать данные всех таблиц между собой разными видами соединений.

Сначала свяжем данные всех трех таблиц левыми соединениями и выведем из справочника *Товары* поле *Товар*, из виртуальной таблицы *Остатки* регистра накопления *ОстаткиТоваров* (таблицы остатков) – поле *КоличествоОстаток*, из виртуальной таблицы *Обороты* регистра накопления *Продажи* (таблицы оборотов) – поле *КоличествоОборот* (листинг 1.76).

Листинг 1.76. Левое соединение данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
```

```
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
    ПО Товары.Ссылка = Продажи.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

Результат запроса представлен на рис. 1.73.

Запрос: Справочник.Товары (Записей в результате: 7)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Сапоги		10
Кроссовки	15	20
Масло		50
Молоко	40	40
Сметана	70	
Пинетки		

Рис. 1.73. Левое соединение данных справочника «Товары» и данных таблиц остатков и оборотов товаров

Мы видим, что в результат запроса попали все записи (7 товаров) из справочника *Товары*, как и должно быть при левом соединении. Для каждого товара были найдены соответствующие записи в таблицах остатков и оборотов товаров и подставлены в результат запроса.

Если проанализировать текст запроса, представленный в листинге 1.76, то понятно, что сначала выполняется левое соединение справочника *Товары* с виртуальной таблицей остатков. Результат этого соединения мы видим на рис. 1.74, слева. Для каждого товара из справочника ищется соответствующая запись в таблице остатков, и найденный остаток выводится для конкретного товара. Товары, у которых нет остатка (*Сапоги*, *Масло*, *Пинетки*), также попадают в результат соединения.

Запрос: Справочник.Товары (Записей в результате: 7)

Наименование	КоличествоОстаток
Туфли	1
Сапоги	
Кроссовки	15
Масло	
Молоко	40
Сметана	70
Пинетки	

Запрос: Справочник.Товары (Записей в результате: 7)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Сапоги		10
Кроссовки	15	20
Масло		50
Молоко	40	40
Сметана	70	
Пинетки		

Рис. 1.74. Левое соединение данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе результат этого первого соединения (см. рис. 1.74, левая таблица) связывается левым соединением с виртуальной таблицей оборотов. Для каждого товара из результата первого соединения ищется соответствующая запись в таблице оборотов, и найденное количество продаж выводится для конкретного товара. Товары, которые ни разу не продавались (*Сметана*, *Пинетки*), также попадают в результат запроса (см. рис. 1.74, правая таблица).

Теперь посмотрим, как изменится результат запроса при различных комбинациях типов соединений исходных таблиц запроса. Оставим без изменения тип первого (по порядку следования в запросе) соединения справочника *Товары* с виртуальной таблицей остатков. А тип второго соединения изменим на внутреннее соединение (листинг 1.77).

Листинг 1.77. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
    ПО Товары.Ссылка = Продажи.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

При выполнении запроса сначала все происходит так же, как и в предыдущем случае. Таблица справочника *Товары* связывается с виртуальной таблицей остатков левым соединением. Результат этого соединения мы видим на рис. 1.75, слева. Для каждого товара из справочника ищется соответствующая запись в таблице остатков, и найденный остаток выводится для конкретного товара. Товары, у которых нет остатка (*Сапоги*, *Масло*, *Пинетки*), также попадают в результат соединения.

Запрос: Справочник.Товары (Записей в результате: 7)

Наименование	КоличествоОстаток
Туфли	1
Сапоги	
Кроссовки	15
Масло	
Молоко	40
Сметана	70
Пинетки	

Запрос: Справочник.Товары (Записей в результате: 5)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Сапоги		10
Кроссовки	15	20
Масло		50
Молоко	40	40

Рис. 1.75. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе результат этого первого соединения (см. рис. 1.75, левая таблица) связывается внутренним соединением с виртуальной таблицей оборотов. Поскольку тип соединения – внутренний, в результат запроса попадают только те товары из результата соединения, для которых найдена соответствующая запись в таблице оборотов. Товары, которые ни разу не продавались (*Сметана*, *Пинетки*) из результата первого соединения, не попадают в результат запроса (см. рис. 1.75, правая таблица).

Теперь свяжем внутренним соединением справочник *Товары* с виртуальной таблицей остатков. Тип второго соединения (по порядку следования в запросе) установим как левое соединение (листинг 1.78).

Листинг 1.78. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
    ПО Товары.Ссылка = Продажи.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

При выполнении запроса сначала происходит внутреннее соединение справочника *Товары* с виртуальной таблицей остатков. Результат этого соединения мы видим на рис. 1.76, слева. Поскольку тип соединения – внутренний, в результат попадают только четыре из семи товаров из справочника, у которых есть остаток.

Запрос: Справочник.Товары (Записей в результате: 4)

Наименование	КоличествоОстаток
Туфли	1
Кроссовки	15
Молоко	40
Сметана	70

Запрос: Справочник.Товары (Записей в результате: 4)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40
Сметана	70	

Рис. 1.76. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе результат этого первого соединения (см. рис. 1.76, левая таблица) связывается левым соединением с виртуальной таблицей оборотов. Для каждого товара из результата первого соединения ищется соответствующая запись в таблице оборотов, и найденное количество продаж выводится для конкретного товара. Товары, которые ни разу не продавались (*Сметана*) из результата первого соединения, также попадают в результат запроса (см. рис. 1.76, правая таблица).

Теперь свяжем все три исходные таблицы запроса внутренними соединениями (листинг 1.79).

Листинг 1.79. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
    ПО Товары.Ссылка = Продажи.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

При выполнении запроса сначала все происходит так же, как и в предыдущем случае. Таблица справочника *Товары* связывается с виртуальной таблицей остатков внутренним соединением. Результат этого соединения мы видим на рис. 1.77, слева. Поскольку тип соединения – внутренний, в результат попадают только четыре из семи товаров из справочника, у которых есть остаток.

Запрос: Справочник.Товары (Записей в результате: 4)

Наименование	КоличествоОстаток
Туфли	1
Кроссовки	15
Молоко	40
Сметана	70

Запрос: Справочник.Товары (Записей в результате: 3)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40

Рис. 1.77. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе результат этого первого соединения (см. рис. 1.77, левая таблица) связывается внутренним соединением с виртуальной таблицей оборотов. Поскольку тип соединения – внутренний, в результат запроса попадают только те товары из результата соединения, для которых найдена соответствующая запись в таблице оборотов. Товары, которые ни разу не продавались (*Сметана*) из результата первого соединения, не попадают в результат запроса (см. рис. 1.77, правая таблица).

По такому же принципу вы можете самостоятельно поэкспериментировать, что произойдет с результатом запроса, если связать таблицы комбинациями других, не рассмотренных нами соединений.

Мы же обратим внимание на другую тонкость. Как видно из рассмотренных выше случаев (листинги 1.76–1.79), три исходные таблицы запроса *Справочник.Товары*, *РегистрНакопления.ОстаткиТоваров.Остатки* и *РегистрНакопления.Продажи.Обороты* последовательно связывались разными видами соединений. Порядок расположения синтаксических конструкций в предложении *ИЗ* был следующий: *<Имя первой таблицы> <Тип соединения> <Имя второй таблицы> <Условие связи первой и второй таблицы> <Тип соединения> <Имя третьей таблицы> <Условие связи первой и третьей таблицы>*.

Но можно расположить соединения не последовательно, а как бы вложить их друг в друга и изменить условие связи таблиц, тогда результат запроса будет другим. Например, свяжем левым соединением таблицы остатков и оборотов и затем результат этого соединения свяжем левым соединением с таблицей справочника товаров (листинг 1.80).

Листинг 1.80. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
      ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
```

```
      ПО ОстаткиТоваров.Товар = Продажи.Товар
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

Из текста запроса мы видим, что второе (по порядку следования в запросе) соединение как бы вложено в первое. Но на самом деле оно выполняется первым как соединение с наибольшим уровнем вложенности. Этот уровень также подчеркнут смещением в тексте запроса относительно первого соединения. При этом первое (по порядку следования в запросе) условие связи относится ко второму (вложенному) соединению, а второе условие связи – к первому соединению. Чтобы не запутаться, разберем эту ситуацию на нашем конкретном примере.

При выполнении запроса (см. листинг 1.80) сначала таблица остатков связывается с таблицей оборотов левым соединением. Результат этого соединения мы видим на рис. 1.78, слева. Поскольку тип соединения – левый, в результат попадают все 4 товара, у которых есть остаток. При этом у одного из товаров (*Сметана*) нет продаж, а товары, которые проданы без остатка (*Сапоги*, *Масло*), в результат соединения не попадают.

Результат левого соединения таблицы остатков регистра накопления "Остатки товаров" и таблицы оборотов регистра накопления "Продажи" (4 записи)

Товар	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40
Сметана	70	

Результат левого соединения справочника "Товары" с левым соединением таблицы остатков и таблицы оборотов товаров (7 записей)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Сапоги		
Кроссовки	15	20
Масло		
Молоко	40	40
Сметана	70	
Пинетки		

Рис. 1.78. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе таблица справочника товаров связывается левым соединением с результатом первого соединения (см. рис. 1.78, левая таблица). Для каждого товара из справочника ищется соответствующая запись в результате первого соединения, и найденные остатки и обороты выводятся для конкретного товара. Товары, у которых нет остатка (*Сапоги*, *Масло*, *Пинетки*) из справочника *Товары*, также попадают в результат запроса (см. рис. 1.78, правая таблица).

Теперь изменим тип первого (по порядку следования в запросе) соединения на внутреннее соединение, тип второго соединения оставим без изменений (листинг 1.81).

Листинг 1.81. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
```

```
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
      ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
      ПО ОстаткиТоваров.Товар = Продажи.Товар
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

При выполнении запроса на первом этапе все происходит так же, как и в предыдущем случае (см. объяснение над рис. 1.78). Результат этого соединения мы видим на рис. 1.79, слева.

Результат левого соединения таблицы остатков регистра накопления "Остатки товаров" и таблицы оборотов регистра накопления "Продажи" (4 записи)

Товар	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40
Сметана	70	

Результат внутреннего соединения справочника "Товары" с левым соединением таблицы остатков и таблицы оборотов товаров (4 записи)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40
Сметана	70	

Рис. 1.79. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе таблица справочника товаров связывается внутренним соединением с результатом первого соединения (см. рис. 1.79, левая таблица). Поскольку тип соединения – внутренний, то в результат запроса (см. рис. 1.79, правая таблица) попадают только четыре из семи товаров из справочника, которые присутствуют в результате первого соединения.

Теперь установим тип первого (по порядку следования в запросе) соединения как левое соединение, а тип второго соединения – как внутреннее соединение (листинг 1.82).

Листинг 1.82. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
      ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
      ПО ОстаткиТоваров.Товар = Продажи.Товар
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

При выполнении запроса сначала таблица остатков связывается с таблицей оборотов внутренним соединением. Результат этого соединения мы видим на рис. 1.80, слева. Поскольку тип соединения – внутренний, в результат попадают только 3 товара, у которых есть и остаток, и оборот. При этом товар *Сметана*, у которого есть остаток, но нет продаж, а также товары, которые проданы без остатка (*Сапоги*, *Масло*), в результат

соединения не попадают.

Результат внутреннего соединения таблицы остатков регистра накопления "Остатки товаров" и таблицы оборотов регистра накопления "Продажи" (3 записи)

Товар	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40

Результат левого соединения справочника "Товары" с внутренним соединением таблицы остатков и таблицы оборотов товаров (7 записей)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Сапоги		
Кроссовки	15	20
Масло		
Молоко	40	40
Сметана		
Пинетки		

Рис. 1.80. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе таблица справочника товаров связывается левым соединением с результатом первого соединения (см. рис. 1.80, левая таблица). Для каждого товара из справочника ищется соответствующая запись в результате первого соединения, и найденные остатки и обороты выводятся для конкретного товара. Товары, которые ни разу не продавались (*Сметана*, *Пинетки*), и товары, у которых нет остатка (*Сапоги*, *Масло*, *Пинетки*) из справочника *Товары*, также попадают в результат запроса (см. рис. 1.80, правая таблица).

Теперь изменим тип первого (по порядку следования в запросе) соединения на внутреннее соединение, а тип второго соединения оставим без изменений (листинг 1.83).

Листинг 1.83. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

```
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
      ВНУТРЕННЕЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
      ПО ОстаткиТоваров.Товар = Продажи.Товар
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

При выполнении запроса на первом этапе все происходит так же, как и в предыдущем случае (см. объяснение над рис. 1.80). Результат этого соединения мы видим на рис. 1.81, слева.

Результат внутреннего соединения таблицы остатков регистра накопления "Остатки товаров" и таблицы оборотов регистра накопления "Продажи" (3 записи)

Товар	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40

Результат внутреннего соединения справочника "Товары" с внутренним соединением таблицы остатков и таблицы оборотов товаров (3 записи)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40

Рис. 1.81. Комбинация типов соединений данных справочника «Товары» и данных таблиц остатков и оборотов товаров

На следующем этапе таблица справочника товаров связывается внутренним соединением с результатом первого соединения (см. рис. 1.81, левая таблица). Поскольку тип соединения – внутренний, то в результат запроса (см. рис. 1.81, правая таблица) попадают только три из семи товаров из справочника, которые присутствуют в результате первого соединения.

Как видно из рассмотренных выше случаев (листинги 1.80–1.83), порядок расположения синтаксических конструкций в предложении *ИЗ* был следующий: *<Имя первой таблицы> <Тип соединения> <Имя второй таблицы> <Тип соединения> <Имя третьей таблицы> <Условие связи второй и третьей таблицы> <Условие связи первой и второй таблицы>*.

Таким образом, от порядка соединения исходных таблиц запроса и их местоположения при описании соединения напрямую зависит результат запроса. Поэтому в этом вопросе нужно проявлять повышенное внимание.

Как получить данные из таблицы, на которую ссылается поле другой таблицы

В языке запросов «1С:Предприятия» существует очень удобная возможность – обращаться не только к полям исходных таблиц запроса, перечисленным в предложении *ИЗ*, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса.

В начале главы "Как хранятся данные в «1С:Предприятии»" мы рассматривали, как хранятся данные таблиц, имеющих ссылочные поля на другие таблицы в информационной базе «1С:Предприятия» (рис. 1.82).

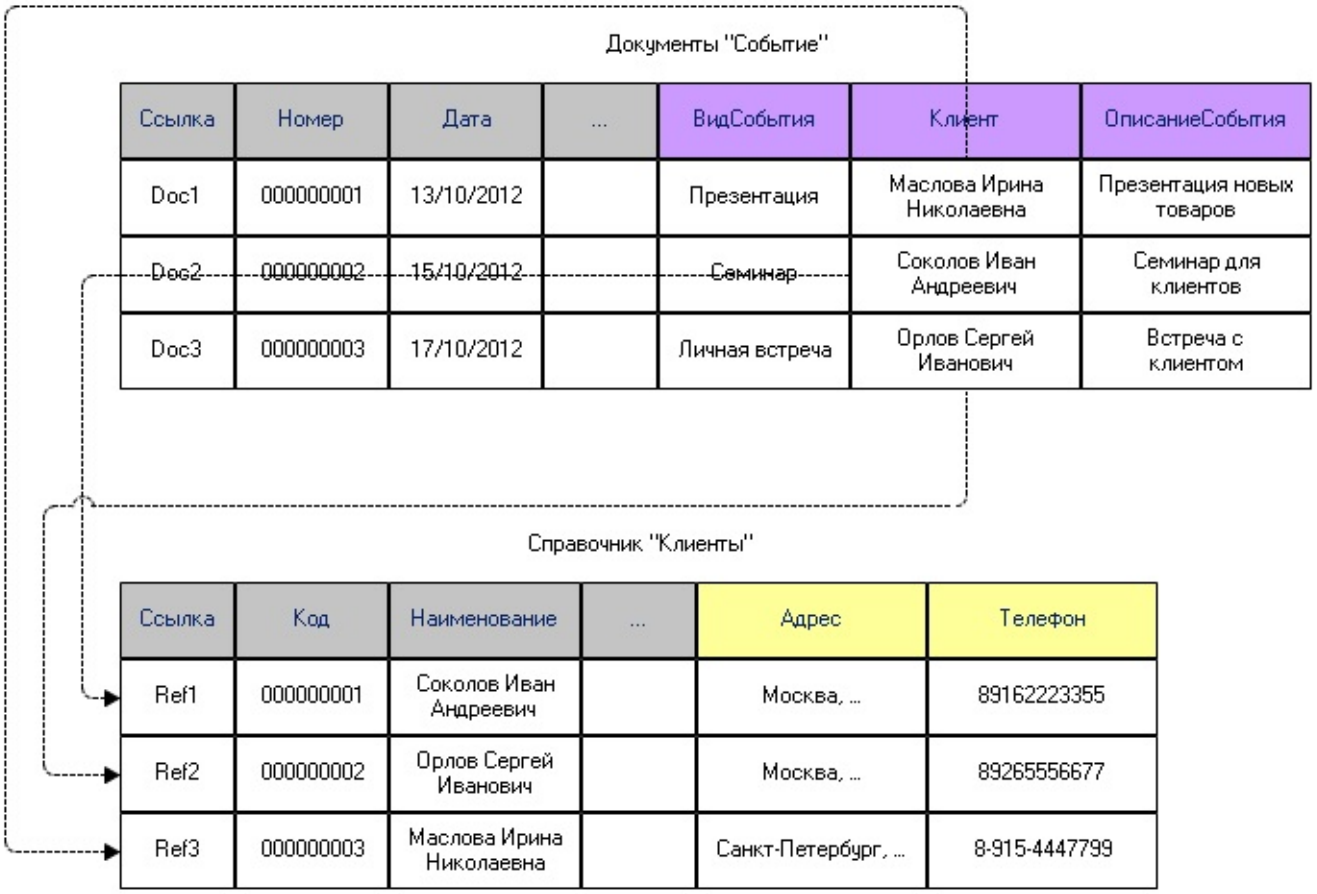


Рис. 1.82. Связь данных документа и справочника по ссылочному полю в информационной базе

В приведенном примере в качестве исходной таблицы выступает документ *Событие*, который имеет ссылочное поле *Клиент*. Это поле имеет тип ссылки на справочник *Клиенты*, а значениями поля *Клиент* в таблице документа являются ссылки на конкретные элементы справочника. В этом случае, обращаясь к полю *Клиент* в документе, мы можем получить из справочника *Клиенты* любые данные о клиенте, на которого ссылается данное поле.

Теперь посмотрим, как это реализуется с помощью языка запросов. Допустим, нам нужно вывести список документов *Событие* и при этом для каждого клиента отобразить его адрес и телефон. Это можно сделать с помощью следующего запроса (листинг 1.84).

Листинг 1.84. Вывод полей из таблицы, на которую ссылается поле исходной таблицы запроса

```
ВЫБРАТЬ
  Событие.Дата КАК Дата,
  Событие.ВидСобытия,
  Событие.Клиент,
  Событие.Клиент.Адрес КАК Адрес,
  Событие.Клиент.Телефон КАК Телефон
ИЗ
  Документ.Событие КАК Событие
УПОРЯДОЧИТЬ ПО
  Дата
```

В списке выборки запроса перечислены поля *Дата*, *ВидСобытия* и *Клиент* исходной таблицы запроса – документа *Событие*, а также поля *Адрес* и *Телефон* таблицы

справочника *Клиенты*, на которую ссылается поле документа *Клиент*. Имена полей таблицы справочника пишутся через точку, например, *Клиент.Адрес*.

Обращение к полям через точку называется операцией *разыменования* ссылочного поля. Причем в общем случае в цепочке разыменования может быть перечислено через точку несколько полей, например, *Номенклатура.Поставщик.Страна*. В этом примере поле *Номенклатура* исходной таблицы ссылается на другую таблицу (справочник *Поставщики*), имеющую, в свою очередь, поле *Поставщик*, ссылающееся на третью таблицу (справочник *Страны*), из которой запросом получается поле *Страна*.

В результате выполнения запроса (см. листинг 1.84) для каждой записи таблицы документа *Событие* по значению поля *Клиент* будет найдена соответствующая запись в таблице справочника *Клиенты*, в которой значение поля *Ссылка* будет равно значению поля *Клиент* в документе. Затем из этой записи справочника будут получены значения полей *Адрес* и *Телефон* аналогично тому, как выбираются значения полей *Дата* и *ВидСобытия* из исходной таблицы запроса (рис. 1.83).

Запрос: Документ.Событие (Записей в результате: 3)

Дата	ВидСобытия	Клиент	АдресКлиента	Телефон
13.10.2012 12:00:00	Презентация	Маслова Ирина Николаевна	Санкт-Петербург, ...	8-915-4447799
15.10.2012 12:00:00	Семинар	Соколов Иван Андреевич	Москва, ...	8-916-222-33-55
17.10.2012 12:00:00	Личная встреча	Орлов Сергей Иванович	Москва, ...	8-926-555-66-77

Рис. 1.83. Вывод полей из таблицы, на которую ссылается поле исходной таблицы запроса

Значения самого поля *Клиент* также на самом деле получаются из таблицы справочника в соответствии с его свойством *Основное представление*.

подробнее

О поле *Представление* рассказано в разделе «Как получить текстовое представление ссылочного поля».

Возможность обращения к полям через точку позволяет значительно упростить написание запросов. Рекомендуется всегда пользоваться разыменованием полей там, где это возможно, чтобы не усложнять запросы лишними конструкциями. Конечно, количество ссылочных полей, перечисленных в цепочке разыменования, должно быть ограничено пределами разумного, чтобы не снижать эффективность исполнения запроса.

Ниже мы увидим, что на самом деле применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей через точку. Поэтому при разыменовании ссылочных полей составного типа (например, поле *Регистратор* регистра накопления и т. п.) следует предпринимать дополнительные усилия по оптимизации запроса, так как в этом случае запрос может работать неоптимально.

Подробнее

Раздел «Ограничить получение данных через точку от полей составного ссылочного типа».

Рассмотрим, что же на самом деле происходит при получении значений полей через точку в языке запросов. При разыменовании ссылочных полей платформа добавляет таблицу, на которую ссылается поле исходной таблицы запроса, в список источников запроса (в предложение *ИЗ*) и выполняет соединение этих таблиц по полям ссылочного типа. То есть в условии связи (после ключевого слова *ПО*) для каждой записи из обеих таблиц проверяется условие равенства значения ссылочного поля исходной таблицы (в нашем случае поля *Клиент* документа *Событие*) и уникальной ссылки, идентифицирующей запись той таблицы, на которую указывает это поле (в нашем случае поля *Ссылка* справочника *Клиенты*).

Таким образом, предыдущий запрос можно представить в следующих вариантах (листинг 1.85).

Листинг 1.85. Вывод полей из таблицы, на которую ссылается поле исходной таблицы запроса

```
// Разыменование полей ссылочного типа
ВЫБРАТЬ
  Событие.Дата КАК Дата,
  Событие.ВидСобытия,
  Событие.Клиент,
  Событие.Клиент.Адрес КАК Адрес,
  Событие.Клиент.Телефон КАК Телефон
ИЗ
  Документ.Событие КАК Событие
УПОРЯДОЧИТЬ ПО
  Дата

// Соединение таблиц
ВЫБРАТЬ
  Событие.Дата КАК Дата,
  Событие.ВидСобытия,
  Событие.Клиент,
  Клиенты.Адрес,
  Клиенты.Телефон
ИЗ
  Документ.Событие КАК Событие
    ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Клиенты КАК Клиенты
    ПО Событие.Клиент = Клиенты.Ссылка
УПОРЯДОЧИТЬ ПО
  Дата
```

Во втором варианте запроса после ключевого слова *ИЗ* перечислены источники запроса – таблица *Документ.Событие*, содержащая ссылочное поле *Клиент*, и таблица *Справочник.Клиенты*, на которую ссылается это поле. Но таблицы не просто перечислены через запятую, а связаны между собой *левым соединением* (между именами таблиц находятся ключевые слова *ЛЕВОЕ СОЕДИНЕНИЕ*). Тип соединения таблиц определяет, должны ли записи каждой из таблиц удовлетворять условию связи.

подробнее

Типы соединений источников запроса подробно рассмотрены в разделе «Как получить данные из разных таблиц для одного и того же поля».

В условии связи источников запроса сравниваются значения ссылочных полей из обеих таблиц (*Событие.Клиент = Клиенты.Ссылка*). Как мы знаем, при левом соединении таблиц в результат запроса попадут записи из обеих таблиц, которые удовлетворяют условию связи, и, кроме того, записи из первой таблицы, расположенной слева от ключевого слова *СОЕДИНЕНИЕ*, для которых не найдено соответствия во второй таблице.

Результат запроса (см. рис. 1.83), использующего соединение таблиц, будет аналогичен выполнению предыдущего запроса, использующего разыменование полей ссылочного типа.

Как получить данные из разных таблиц, не связывая, а дополняя их

При выборке данных с помощью запросов бывает необходимо дополнить данные, получившиеся в результате выполнения одного запроса, данными другого запроса. Например, можно дополнить информацию из приходных накладных со структурой (*Контрагент*, *Товар*, *Количество*, *Цена*) данными из расходных накладных с такой же структурой.

Для этого в языке запросов существует возможность *объединения* нескольких запросов. При этом записи, полученные с помощью каждого из объединяемых запросов, собираются в один результат запроса. При объединении каждый запрос получает данные независимо, то есть у каждого из запросов – свое описание выбираемых полей (*ВЫБРАТЬ*), источников запроса (*ИЗ*), условий отбора (*ГДЕ*), полей группировки (*СГРУППИРОВАТЬ ПО*). Затем данные, получаемые в результате каждого запроса, объединяются, и уже над этим объединением выполняются такие операции, как упорядочивание результатов (*УПОРЯДОЧИТЬ ПО*) и расчет итогов (*ИТОГИ ПО*).

Для объединения запросов используется предложение *ОБЪЕДИНИТЬ*, которое располагается в секции *Объединение запросов* текста запроса. Все секции запроса приведены на рис. 1.13.

Предположим, нам нужно объединить вместе информацию из документов *Заказ товара* и информацию из документов *Расходная накладная*. При этом нужно вывести суммарное количество заказанных и проданных товаров по каждому клиенту. Также мы хотим видеть общий итог по результату запроса в целом и отдельно по каждому клиенту.

Для решения поставленной задачи нужно получить данные из таблицы *Документ.ЗаказТовара*, сгруппировать их по полям этой таблицы *Клиент* и *Товар*, затем получить данные из таблицы *Документ.РасходнаяНакладная*, сгруппировать их по полям этой таблицы *Покупатель* и *Товар* и объединить полученную информацию. Затем рассчитать общие итоги и итоги по полю *Клиент* для результата объединения запросов (листинг 1.86).

Листинг 1.86. Объединение данных о заказах товаров клиентами и данных о продажах товаров этим клиентам

```
ВЫБРАТЬ
```

```
  Заказ.Ссылка.Клиент КАК Клиент,
  Заказ.Товар КАК Товар,
  СУММА(Заказ.Количество) КАК Заказано,
  СУММА(0) КАК Продано
ИЗ
  Документ.ЗаказТовара.Состав КАК Заказ
СГРУППИРОВАТЬ ПО
  Заказ.Ссылка.Клиент,
  Заказ.Товар

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  Накладная.Ссылка.Покупатель,
  Накладная.Товар,
  СУММА(0),
  СУММА(Накладная.Количество)
ИЗ
  Документ.РасходнаяНакладная.Состав КАК Накладная
СГРУППИРОВАТЬ ПО
  Накладная.Ссылка.Покупатель,
  Накладная.Товар

ИТОГИ ПО
  ОБЩИЕ,
  Клиент
```

В тексте запроса сначала описывается первый запрос, получающий информацию из документа *ЗаказТовара*, затем следуют ключевые слова *ОБЪЕДИНИТЬ ВСЕ*, после которых следует описание присоединяемого запроса, получающего информацию из документа *РасходнаяНакладная*. После этого описывается необходимость расчета итогов для результата объединения запросов.

Названия полей результата запроса описываются в списке полей выборки первого из объединяемых запросов. Поля выборки второго запроса сопоставляются с полями результата в соответствии с порядком их следования в списке полей выборки.

Поэтому объединяемые запросы должны иметь одинаковое количество полей в списке полей выборки. В случае, если поля выборки объединяемых запросов имеют разный тип, поля результата запроса будут иметь составной тип. Но в нашем случае поле выборки результата объединения *Клиент* имеет тип *СправочникСсылка.Клиенты*, так как поле *Клиент* документа *ЗаказТовара* и поле *Покупатель* документа *РасходнаяНакладная* из объединяемых запросов также принадлежат к этому типу.

Результат выполнения запроса представлен на рис. 1.84.

Запрос: Документ.ЗаказТовара.Состав (Записей в результате: 24)

Клиент	Товар	Заказано	Продано
		191	134
Соколов Иван Андреевич		26	18
Соколов Иван Андреевич	Туфли	6	0
Соколов Иван Андреевич	Сапоги	5	0
Соколов Иван Андреевич	Кроссовки	15	0
Соколов Иван Андреевич	Туфли	0	5
Соколов Иван Андреевич	Сапоги	0	3
Соколов Иван Андреевич	Кроссовки	0	10
Орлов Сергей Иванович		100	60
Орлов Сергей Иванович	Масло	30	0
Орлов Сергей Иванович	Молоко	30	0
Орлов Сергей Иванович	Сметана	40	0
Орлов Сергей Иванович	Масло	0	40
Орлов Сергей Иванович	Молоко	0	20
Маслова Ирина Николаевна		65	56
Маслова Ирина Николаевна	Туфли	10	0
Маслова Ирина Николаевна	Сапоги	10	0
Маслова Ирина Николаевна	Масло	25	0
Маслова Ирина Николаевна	Сметана	20	0
Маслова Ирина Николаевна	Туфли	0	9
Маслова Ирина Николаевна	Сапоги	0	7
Маслова Ирина Николаевна	Кроссовки	0	10
Маслова Ирина Николаевна	Масло	0	10
Маслова Ирина Николаевна	Молоко	0	20

Рис. 1.84. Объединение данных о заказах товаров клиентами и данных о продажах товаров этим клиентам

В общем случае могут объединяться результаты выполнения сразу нескольких запросов.

По умолчанию при объединении запросов полностью одинаковые строки в результате запроса, сформированные разными запросами, заменяются одной. Если требуется, чтобы были оставлены разные строки, необходимо указать ключевое слово *ВСЕ*.

В общем случае при объединении в запросе результатов нескольких запросов следует использовать конструкцию *ОБЪЕДИНИТЬ ВСЕ*, а не *ОБЪЕДИНИТЬ*, поскольку во втором варианте при объединении запросов полностью одинаковые строки заменяются одной, на что затрачивается дополнительное время, даже в случаях, когда одинаковых строк в запросах заведомо быть не может.

Временные таблицы и пакетные запросы

Довольно часто бывают ситуации, когда в качестве источника запроса необходимо использовать не таблицы базы данных, а данные, полученные в результате выполнения другого запроса. Например, источником запроса могут служить данные приходных накладных, сгруппированные по товарам или отобранные по дате документа.

Запрос, служащий источником данных, является *вложенным*, а запрос, который эти данные использует, является основным или *внешним*.

Рассмотрим эту ситуацию на примере. Допустим, нам нужно вывести данные обо всех товарах и их производителях с соблюдением иерархии справочника *Товары*. Для каждого товара из справочника мы хотим видеть дату его поступления и поставщика из приходных накладных.

Казалось бы, в этом нет ничего особенно сложного. Свяжем левым соединением таблицы справочника *Товары* и документа *ПриходнаяНакладная* по ссылкам товаров и выведем требуемые поля из обеих таблиц. При этом упорядочим результат запроса по наименованиям товаров в порядке иерархии справочника (листинг 1.87).

Листинг 1.87. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении

```
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  Поступление.Ссылка.Дата,
  Поступление.Ссылка.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ Документ.ПриходнаяНакладная.Состав КАК Поступление
    ПО Товары.Ссылка = Поступление.Товар

УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

Результат выполнения запроса представлен на рис. 1.85.

Запрос: Справочник.Товары (Записей в результате: 16)

Наименование	Производитель	Дата	Поставщик
Обувь			
Детская обувь			
Пинетки			
Кроссовки		17.10.2012 12:00:00	Скороход АО
Кроссовки		05.11.2012 12:00:00	Корнет ЗАО
Сапоги	Италия	17.10.2012 12:00:00	Скороход АО
Сапоги	Италия	05.11.2012 12:00:00	Корнет ЗАО
Туфли	Германия	17.10.2012 12:00:00	Скороход АО
Туфли	Германия	05.11.2012 12:00:00	Корнет ЗАО
Продукты			
Масло	Россия	02.11.2012 12:00:00	Животноводство ООО
Масло	Россия	25.10.2012 12:00:00	Животноводство ООО
Молоко	Россия	02.11.2012 12:00:00	Животноводство ООО
Молоко	Россия	25.10.2012 12:00:00	Животноводство ООО
Сметана		02.11.2012 12:00:00	Животноводство ООО
Сметана		25.10.2012 12:00:00	Животноводство ООО

Рис. 1.85. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении

На реальной базе, конечно, список товаров может быть очень большим. Предположим, мы хотим получить тот же список, но с данными о поступлениях товаров только за прошедший месяц.

Для этого наложим условие на поле *Дата* приходной накладной так, чтобы выбирались данные из приходных накладных только за ноябрь (при этом, конечно же, за другие годы данных в нашей демонстрационной конфигурации нет), листинг 1.88.

Листинг 1.88. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  Поступление.Ссылка.Дата,
```

```
  Поступление.Ссылка.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ Документ.ПриходнаяНакладная.Состав КАК Поступление
    ПО Товары.Ссылка = Поступление.Товар
ГДЕ
  МЕСЯЦ(Поступление.Ссылка.Дата) = 11

УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

Мы видим, что результат запроса – несколько не такой, как мы ожидали (рис. 1.86).

Запрос: Справочник.Товары (Записей в результате: 6)

Наименование	Производитель	Дата	Поставщик
Кроссовки		05.11.2012 12:00:00	Корнет ЗАО
Масло	Россия	02.11.2012 12:00:00	Животноводство ООО
Молоко	Россия	02.11.2012 12:00:00	Животноводство ООО
Сапоги	Италия	05.11.2012 12:00:00	Корнет ЗАО
Сметана		02.11.2012 12:00:00	Животноводство ООО
Туфли	Германия	05.11.2012 12:00:00	Корнет ЗАО

Рис. 1.86. Вывод товаров, поступавших в ноябре, из справочника «Товары»

В результате выполнения запроса отражены только те товары, которые поступали в ноябре, а также «пропала» иерархия справочника товаров, так как в результате запроса нет данных для построения иерархии. Так произошло потому, что при накладывании условия отбора на поле правой таблицы при левом соединении тип соединения становится внутренним.

В этом и в некоторых других случаях выходом из ситуации является использование вложенного запроса в качестве источника внешнего запроса. Поэтому предыдущий запрос можно переписать следующим образом (1.89).

Листинг 1.89. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  ПоступлениеТоваров.Дата,
  ПоступлениеТоваров.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ
      (ВЫБРАТЬ
        Поступление.Товар,
        Поступление.Ссылка.Дата,
        Поступление.Ссылка.Поставщик
      ИЗ
        Документ.ПриходнаяНакладная.Состав КАК Поступление
      ГДЕ
        МЕСЯЦ(Поступление.Ссылка.Дата) = 11
      ) КАК ПоступлениеТоваров
    ПО Товары.Ссылка = ПоступлениеТоваров.Товар
УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

В тексте основного запроса вложенный запрос выделен жирным шрифтом и смещен относительно внешнего запроса. При выполнении основного запроса сначала выполняется вложенный запрос, в котором получаются данные о поступлении товаров за ноябрь. Затем справочник товаров связывается левым соединением с результатом вложенного запроса по ссылкам товаров, и в итоге мы получаем требуемый результат (рис. 1.87).

Запрос: Справочник.Товары (Записей в результате: 10)

Наименование	Производитель	Дата	Поставщик
Обувь			
Детская обувь			
Пинетки			
Кроссовки		05.11.2012 12:00:00	Корнет ЗАО
Сапоги	Италия	05.11.2012 12:00:00	Корнет ЗАО
Туфли	Германия	05.11.2012 12:00:00	Корнет ЗАО
Продукты			
Масло	Россия	02.11.2012 12:00:00	Животноводство ООО
Молоко	Россия	02.11.2012 12:00:00	Животноводство ООО
Сметана		02.11.2012 12:00:00	Животноводство ООО

Рис. 1.87. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

Однако мы видим, что текст запроса (см. листинг 1.89) стал довольно громоздким и сложным для восприятия. С этой точки зрения лучше поместить результат вложенного запроса во временную таблицу и затем использовать ее как источник нашего основного запроса.

Временная таблица не существует в базе данных. Это просто некоторая область в памяти компьютера, которая создается на ограниченное время, как, например, это происходит при создании какой-либо переменной встроенного языка. Эта область создается и заполняется данными при выполнении запроса, содержащего ключевое слово *ПОМЕСТИТЬ*, после которого следует произвольное имя временной таблицы.

Подробнее

Документация «1С:Предприятие 8.3. Руководство разработчика», раздел 8.2 «Язык запросов», а также встроенная справка *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Выполнение и работа с запросами во встроенном языке > Работа с временными таблицами*.

Дополним текст вложенного запроса из предыдущего запроса предложением *ПОМЕСТИТЬ*, которое располагается сразу после списка полей выборки (листинг 1.90).

Листинг 1.90. Помещение данных о поступлениях товаров за ноябрь во временную таблицу

```
ВЫБРАТЬ
  Поступление.Товар,
  Поступление.Ссылка.Дата КАК Дата,
  Поступление.Ссылка.Поставщик
ПОМЕСТИТЬ ПоступлениеТоваров
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК Поступление
ГДЕ
  МЕСЯЦ(Поступление.Ссылка.Дата) = 11
```

Заметим, что для быстрой и эффективной выборки данных из временной таблицы желательно проиндексировать эту таблицу по полям, которые будут затем участвовать в условии отбора или в условии соединения. Делается это с помощью предложения *ИНДЕКСИРОВАТЬ ПО*.

подробнее

«Эффективное использование индексов. Пример 5».

Чтобы увидеть данные, помещенные во временную таблицу, в консоли запросов нужно нажать кнопку *Выполнить запрос с временными таблицами* рядом с кнопкой *Выполнить* (рис. 1.88).

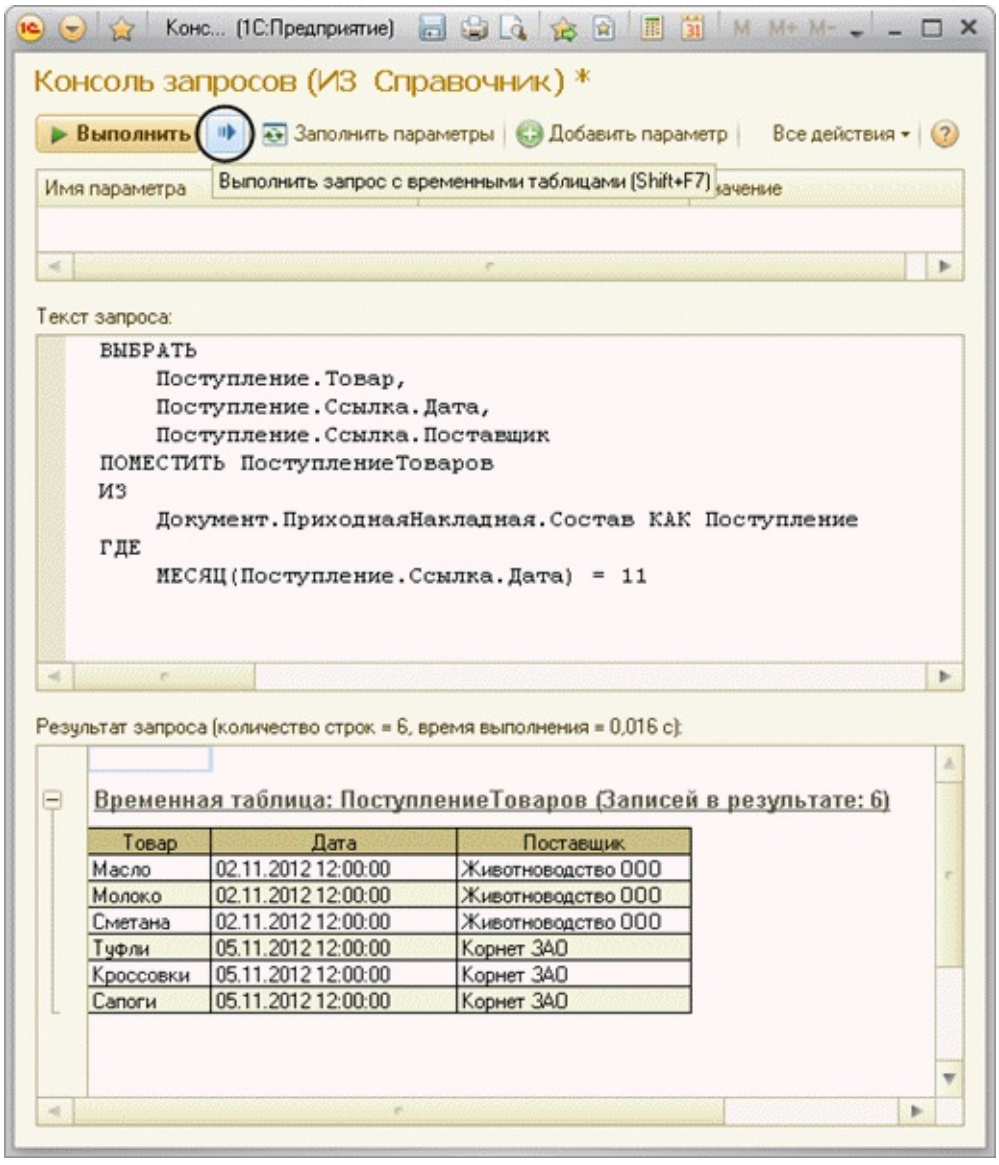


Рис. 1.88. Данные временной таблицы «ПоступлениеТоваров»

Итак, в результате выполнения запроса (листинг 1.90) мы поместили данные о поступлениях товаров за ноябрь во временную таблицу с именем *ПоступлениеТоваров*.

Теперь эту таблицу можно использовать в качестве источника других запросов точно так же, как и таблицу базы данных. Но сделать это можно либо из встроенного языка с

помощью менеджера временных таблиц (этот вариант мы рассмотрим позже в разделе "Использование временных таблиц с помощью встроенного языка"), либо с помощью пакетного запроса. Этот вариант мы и рассмотрим сейчас.

Пакетный запрос содержит последовательность запросов, разделенных символом «;» (точка с запятой). При выполнении пакетного запроса входящие в него запросы выполняются друг за другом. При этом если при выполнении одного из запросов была создана временная таблица, то следующие за ним запросы при их выполнении могут использовать данные этой временной таблицы.

Необходимо учитывать, что временные таблицы будут существовать до окончания исполнения всего пакета запроса или до исполнения в пакете запроса, уничтожающего данную временную таблицу с помощью конструкции *УНИЧТОЖИТЬ*.

Подробнее

Документация «1С:Предприятие 8.3. Руководство разработчика», раздел 8.2 «Язык запросов», а также встроенная справка *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Выполнение и работа с запросами во встроенном языке > Работа с пакетными запросами*.

Таким образом, разобьем запрос, содержащий вложенный запрос (см. листинг 1.89), на два запроса и объединим их в один пакетный запрос (листинг 1.91).

Листинг 1.91. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
ВЫБРАТЬ
  Поступление.Товар,
  Поступление.Ссылка.Дата КАК Дата,
  Поступление.Ссылка.Поставщик
ПОМЕСТИТЬ ПоступлениеТоваров
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК Поступление
ГДЕ
  МЕСЯЦ(Поступление.Ссылка.Дата) = 11
;
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  ПоступлениеТоваров.Дата,
  ПоступлениеТоваров.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ ПоступлениеТоваров КАК ПоступлениеТоваров
    ПО Товары.Ссылка = ПоступлениеТоваров.Товар
УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

В первом запросе мы помещаем данные о поступлении товаров за ноябрь во временную таблицу с именем *ПоступлениеТоваров*. Затем следует символ «;» (точка с запятой), который указывает на то, что это – пакетный запрос. В следующем запросе справочник

товаров связывается левым соединением с временной таблицей по ссылкам товаров, и в итоге мы получаем результат, аналогичный результату при выполнении запроса, содержащего вложенный запрос (см. рис. 1.87).

Таким образом, можно сделать вывод, что текст запроса в случае использования временных таблиц становится более понятным и осмысленным, чем при использовании вместо них вложенных запросов.

Также использование временных таблиц вместо вложенных запросов почти всегда делает запрос более оптимальным. С точки зрения эффективности исполнения запросов крайне не рекомендуется использовать соединения с вложенными запросами, так как в этом случае СУБД может выбрать неоптимальный план запроса, что на больших объемах данных приводит к временным задержкам и другим неприятностям при выполнении таких запросов.

подробнее

Раздел «Не использовать соединения с вложенными запросами и с виртуальными таблицами».

Пример, показывающий использование вложенного запроса для ограничения значений выборки в условии отбора (раздел «Как использовать данные одного запроса внутри другого запроса»), также можно переделать с использованием временной таблицы вместо вложенного запроса.

В заключение покажем, как можно переписать запрос, выполняющий несколько соединений нескольких таблиц (раздел «Как получить данные из разных таблиц, связанных несколькими соединениями») с использованием временной таблицы (листинг 1.92).

Листинг 1.92. Два варианта запроса, выполняющего несколько соединений данных нескольких таблиц

```
// Первый вариант с двумя левыми соединениями
ВЫБРАТЬ
  Товары.Наименование,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
      ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
      ПО ОстаткиТоваров.Товар = Продажи.Товар
    ПО Товары.Ссылка = ОстаткиТоваров.Товар
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ

// Второй вариант с временной таблицей
ВЫБРАТЬ
  ОстаткиТоваров.Товар,
  ОстаткиТоваров.КоличествоОстаток,
  Продажи.КоличествоОборот
ПОМЕСТИТЬ ВременнаяТаблица
```

```
ИЗ
  РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваров
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Продажи.Обороты КАК Продажи
    ПО ОстаткиТоваров.Товар = Продажи.Товар
;
ВЫБРАТЬ
  Товары.Наименование,
  ВременнаяТаблица.КоличествоОстаток,
  ВременнаяТаблица.КоличествоОборот
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ ВременнаяТаблица КАК ВременнаяТаблица
    ПО ВременнаяТаблица.Товар = Товары.Ссылка
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

Второй вариант, использующий временную таблицу, более понятен, по крайней мере полезен для изучения. Так как если выполнить сначала только первую часть запроса (помещающую данные во временную таблицу), а затем – весь запрос целиком, то можно поэтапно проследить, какие данные получаются в результате каждого соединения (рис. 1.89).

Временная таблица "ВременнаяТаблица - результат левого соединения таблицы остатоков с таблицей оборотов товаров (4 записи)

Товар	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Кроссовки	15	20
Молоко	40	40
Сметана	70	

Результат левого соединения справочника "Товары" с временной таблицей "ВременнаяТаблица" (7 записей)

Наименование	КоличествоОстаток	КоличествоОборот
Туфли	1	14
Сапоги		
Кроссовки	15	20
Масло		
Молоко	40	40
Сметана	70	
Пинетки		

Рис. 1.89. Левое соединение данных справочника «Товары» и данных таблиц остатков и оборотов товаров

Глава 2. Работа с запросами во встроенном языке

Конструктор запроса

В первой главе мы познакомились с азами языка запросов, научились создавать синтаксически правильные тексты запросов и выполнять их в консоли запросов. Для решения небольших демонстрационных задач мы писали текст запросов вручную, в специальном окне консоли запросов, и затем сразу же могли увидеть результат выполнения запроса. На начальном этапе это было очень полезно, чтобы набраться опыта при создании простых запросов и закрепить понимание пройденного материала.

Но в дальнейшем для решения реальных задач написание запросов вручную может быть довольно утомительным. Кроме того, помимо знания всех синтаксических конструкций и правил составления запросов, это потребует от разработчика знания имен таблиц языка запросов, их полей и параметров, что потребует постоянного обращения к синтакс-помощнику или встроенной справке.

Поэтому в платформе «1С:Предприятие» существуют инструменты для облегчения труда разработчика при написании запросов – *конструктор запроса* и *конструктор запроса с обработкой результата*.

С помощью конструктора запроса можно визуально сконструировать запрос, то есть с помощью мыши выбрать и перетащить нужные таблицы, поля, установить связи между ними, условия отбора и т. д. После нажатия кнопки *ОК* конструктор запроса создаст синтаксически правильный текст запроса.

С помощью конструктора запроса с обработкой результата можно, помимо самого текста запроса, создать готовый фрагмент кода для получения данных с помощью запроса и вывода их, например, в табличный документ. Этот вопрос мы рассмотрим позже, в разделе "Обработка результатов запроса с помощью конструктора запроса", а пока остановимся на общих принципах работы с конструктором запроса.

Чтобы открыть конструктор запроса, нужно перейти в модуль формы, обработки и т. п., вызвать правой кнопкой мыши контекстное меню и затем выбрать пункт *Конструктор запроса* или *Конструктор запроса с обработкой результата* (рис. 2.1).

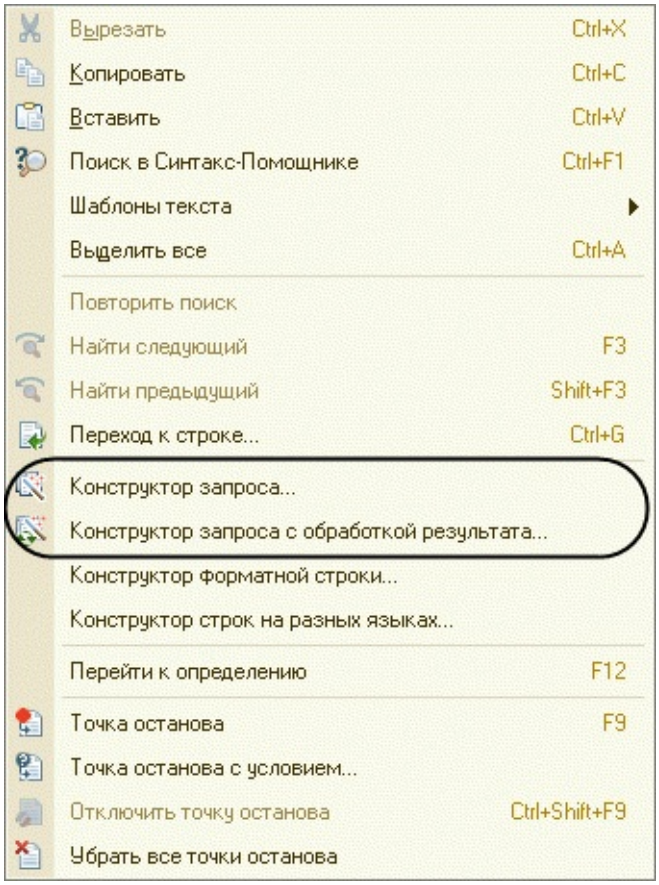


Рис. 2.1. Вызов конструктора запроса

Или же можно выполнить команду главного меню конфигуратора *Текст > Конструктор запроса/Конструктор запроса с обработкой результата*.

В случае, когда вызывается конструктор запроса без обработки результата, желательно предварительно в тексте модуля сделать заготовку для создания запроса и вызвать контекстное меню, поместив курсор между кавычек (листинг 2.1).

Листинг 2.1. Заготовка для создания запроса

```
Запрос = Новый Запрос;
Запрос.Текст = "";
```

Также конструктор запроса может быть вызван из схемы компоновки данных при создании набора данных, источником которого является запрос к информационной базе, при настройке свойств динамического списка, источником которого служит произвольный запрос, и т. д.

Попробуем с помощью конструктора запроса создать текст запросов для нескольких простых задач, используемых в первой главе. Мы не будем здесь объяснять, как и почему применена та или иная конструкция языка запросов. На этих примерах мы просто покажем, как создать аналогичный текст запроса не вручную, а с помощью конструктора.

Создание простого запроса

Для начала создадим с помощью конструктора запроса запрос для отображения в

порядке иерархии всех товаров из справочника *Товары*. Этот пример мы рассматривали в разделе «Как получить записи иерархической таблицы и расположить их в порядке иерархии» (листинг 2.2).

Листинг 2.2. Вывод записей справочника «Товары», расположенных в порядке иерархии

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование КАК Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
УПОРЯДОЧИТЬ ПО
  Наименование ИЕРАРХИЯ
```

Теперь наша задача – получить аналогичный запрос с помощью конструктора запроса.

Итак, откроем нашу демонстрационную конфигурацию *Язык запросов* в режиме *Конфигуратор* и откроем модуль формы обработки *РаботаСЗапросами*. Затем создадим в тексте модуля заготовку, показанную в листинге 2.1, поместим курсор между кавычек и вызовем из контекстного меню пункт *Конструктор запроса*. Подтвердим, что мы хотим создать новый запрос. После этого откроется конструктор запроса (рис. 2.2).

Сначала мы попадаем на основную закладку конструктора *Таблицы и поля*. На этой закладке в списке *База данных* можно выбрать реальные и виртуальные таблицы в качестве источников запроса. Раскроем группу справочников, выберем таблицу *Товары* и перенесем ее в список *Таблицы*. В этом списке можно выбрать нужные поля для отображения в запросе. Раскроем структуру выбранной таблицы и перенесем в список *Поля* поля *Код*, *Наименование*, *Родитель*, *ЭтоГруппа* (рис. 2.2).

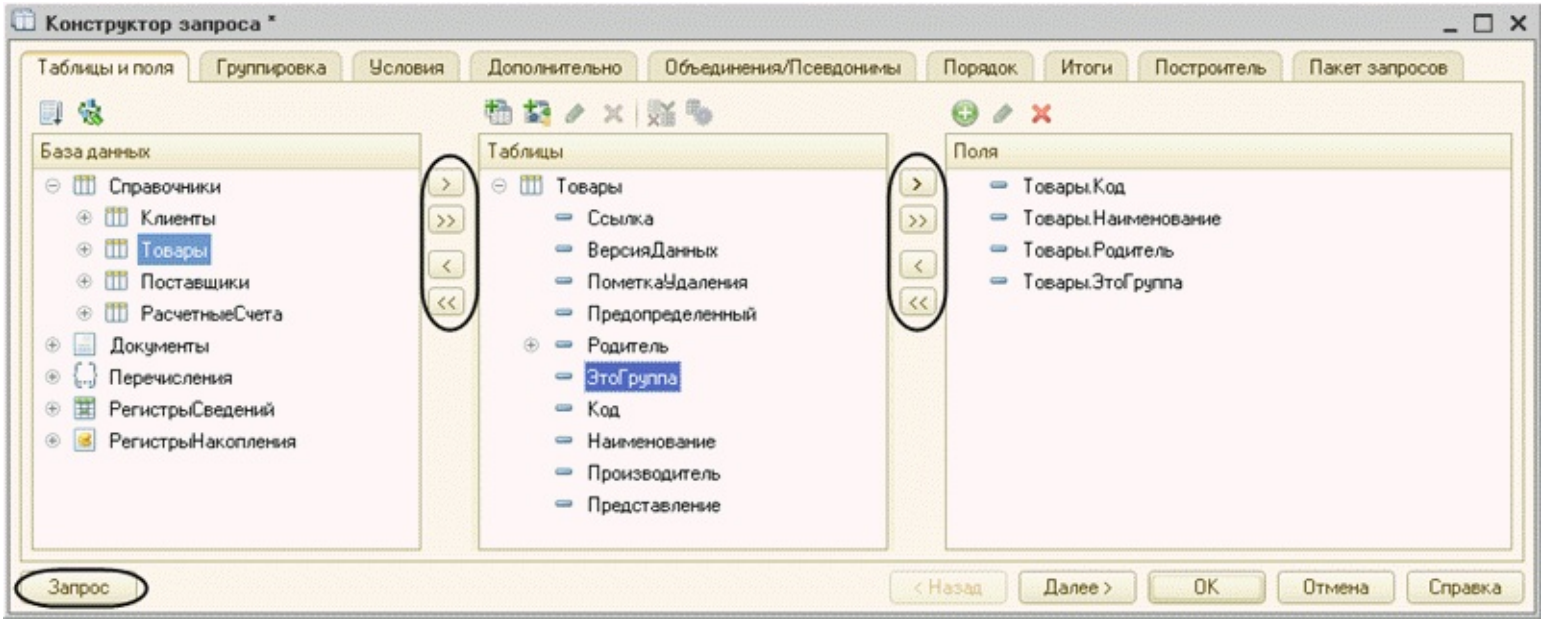


Рис. 2.2. Окно конструктора запроса

ПРИМЕЧАНИЕ

Выделенные элементы можно перенести из одного списка в другой перетаскиванием мышью или двойным щелчком на них. Либо можно использовать кнопки >, >>, <, <<.

На любом этапе создания запроса, не выходя из конструктора запроса (не нажимая *ОК*), можно увидеть, как изменяется текст запроса в процессе работы. Для этого нужно нажать кнопку *Запрос* в левом нижнем углу окна конструктора запроса. Нажмем эту кнопку и посмотрим, какой текст запроса создал конструктор (рис. 2.3).

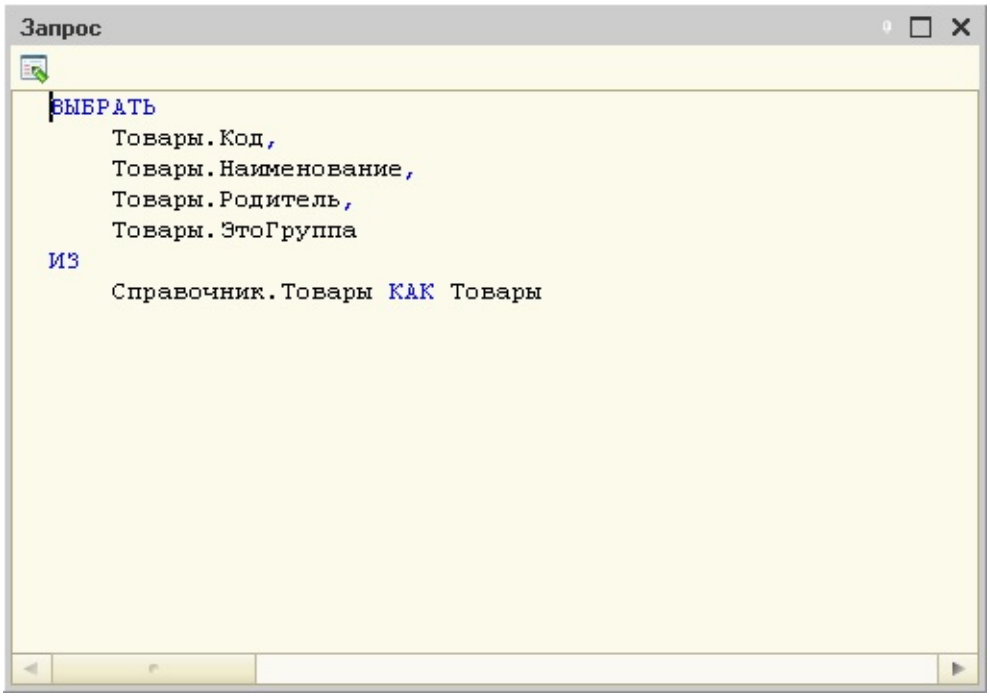


Рис. 2.3. Текст запроса, созданный конструктором

Мы видим, что в запросе выбираются поля *Код*, *Наименование*, *Родитель*, *ЭтоГруппа* из справочника *Товары*. Таким образом, на закладке *Таблицы и поля* конструктора запроса формируются основные предложения языка запросов *ВЫБРАТЬ*, *ИЗ*.

Теперь перейдем на закладку конструктора запроса *Объединения/Псевдонимы*, чтобы задать псевдонимы полей запроса. Зададим псевдоним *КодТовара* для поля *Код* (рис. 2.4).

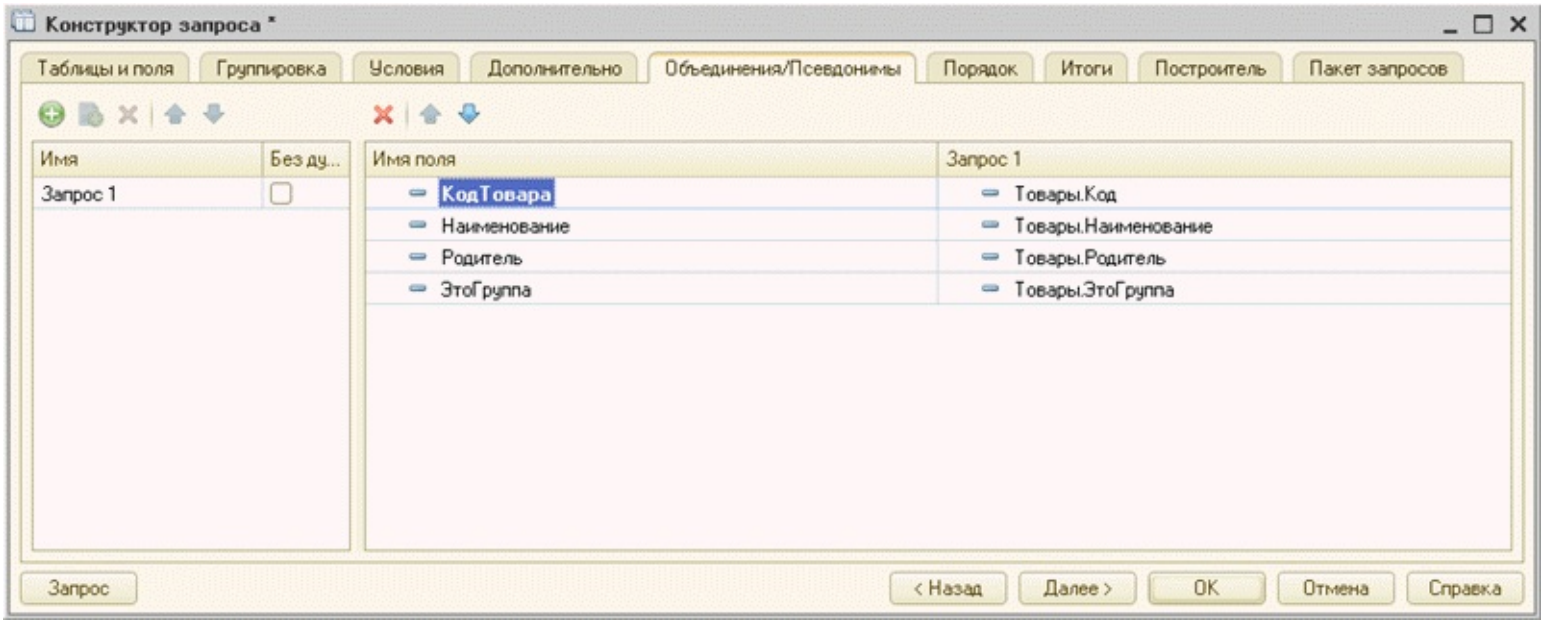


Рис. 2.4. Окно конструктора запроса

Нажмем кнопку *Запрос* и посмотрим, как изменился текст запроса (рис. 2.5).

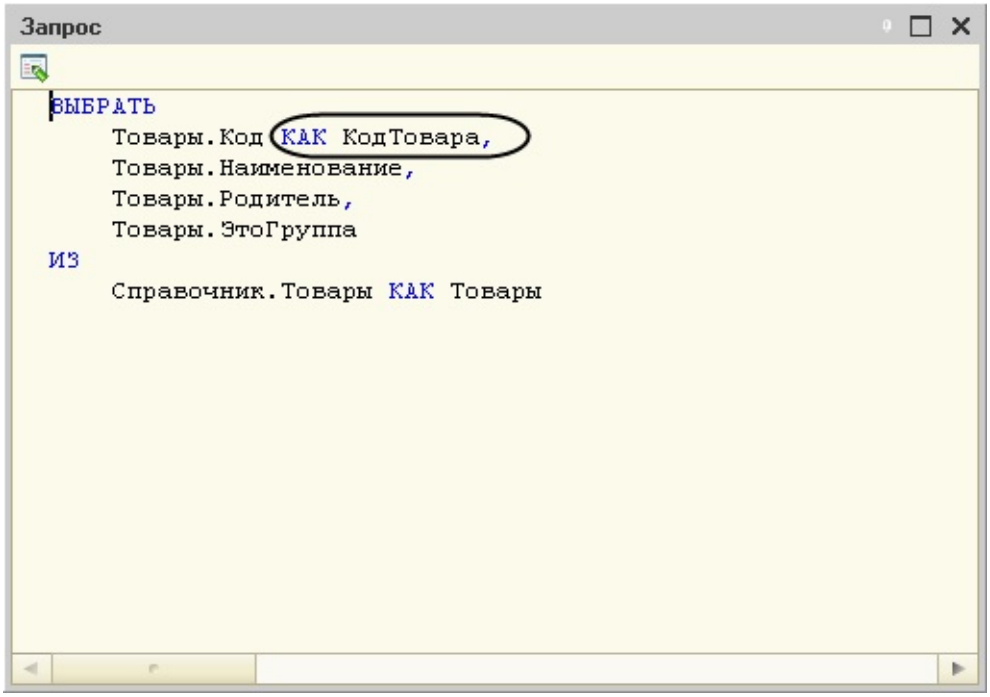


Рис. 2.5. Текст запроса, созданный конструктором

Мы видим, что к тексту запроса конструктор добавляет псевдонимы полей и ключевое слово *КАК*.

В заключение перейдем на закладку конструктора запроса *Порядок* и зададим упорядочивание записей результата запроса. Из списка полей запроса выберем поле *Наименование* и установим тип упорядочивания – *Иерархия* (рис. 2.6).

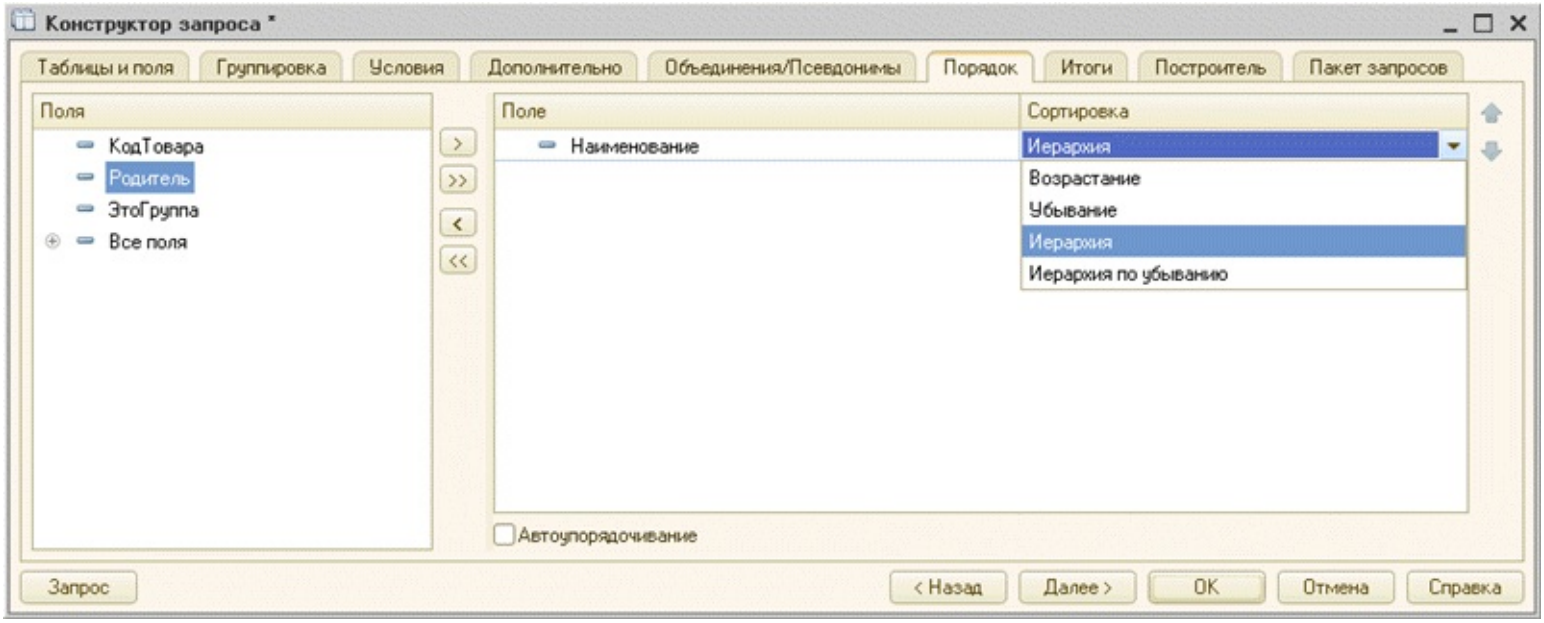


Рис. 2.6. Окно конструктора запроса

Нажмем кнопку *Запрос* и посмотрим, как изменился текст запроса (рис. 2.7).

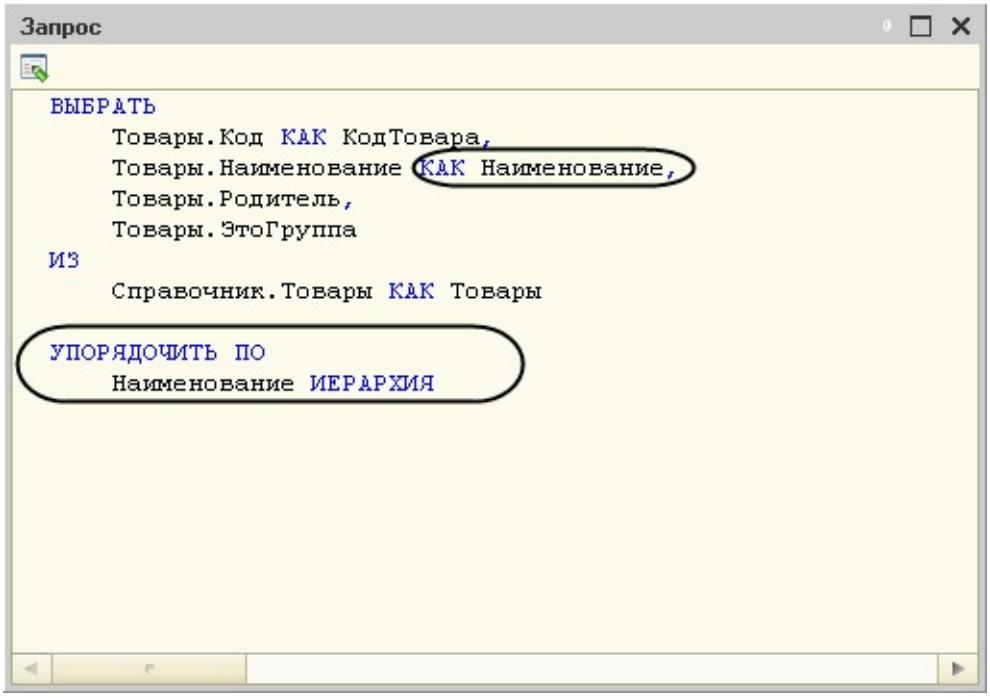


Рис. 2.7. Текст запроса, созданный конструктором

Мы видим, что к тексту запроса конструктор добавляет предложение *УПОРЯДОЧИТЬ ПО*. Кроме того, конструктор автоматически добавляет псевдоним для поля, по которому производится упорядочивание (*Наименование*).

Нажмем *ОК* и вернемся в модуль формы. Запрос, сформированный с помощью конструктора запроса (без обработки результата), будет выглядеть следующим образом (листинг 2.3).

Листинг 2.3. Запрос, созданный конструктором

```
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
    |        Товары.Код КАК КодТовара,
```

```
    |        Товары.Наименование КАК Наименование,
    |        Товары.Родитель,
    |        Товары.ЭтоГруппа
    |ИЗ
    |        Справочник.Товары КАК Товары
    |УПОРЯДОЧИТЬ ПО
    |        Наименование ИЕРАРХИЯ";
```

Для дальнейшего изменения запроса можно поместить курсор внутрь текста запроса и вызвать конструктор запроса из контекстного меню. Если при этом текст запроса содержит ошибки, то конструктор запроса укажет на них и не будет открыт, пока эти ошибки вручную не будут исправлены. Таким образом, конструктор запроса помогает разработчику создать запрос с нуля, а также найти собственные ошибки уже в существующем запросе.

Итак, с помощью конструктора запроса мы получили текст запроса, аналогичный тому, который мы писали вручную (см. листинг 2.2). Далее текст запроса, полученный при нажатии кнопки *Запрос* (см. рис. 2.7), можно скопировать в окно консоли запросов и выполнить. Или же полученный фрагмент кода (см. листинг 2.3), содержащий текст запроса, можно использовать для выполнения и обработки результатов запроса из встроенного языка.

подробнее

Раздел «Выполнение запросов из встроенного языка».

Теперь добавим в запрос условие отбора, чтобы в результате запроса отображались только те записи из справочника *Товары*, которые не являются группой. Этот пример мы рассматривали в разделе «Как получить записи иерархической таблицы, не являющиеся группой» (листинг 2.4).

Листинг 2.4. Вывод записей справочника «Товары», не являющихся группой

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Родитель,
  Товары.ЭтоГруппа
ИЗ
  Справочник.Товары КАК Товары
ГДЕ
  Товары.ЭтоГруппа = ЛОЖЬ
```

Чтобы получить подобный запрос, изменим текст запроса, ранее созданный конструктором (см. листинг 2.3). Отредактируем полученный запрос в конструкторе. Мы видим, что закладки конструктора *Таблицы и поля*, *Объединения/Псевдонимы* и *Порядок* уже заполнены так, как мы это задавали ранее (см. рис. 2.2, 2.4, 2.6).

Чтобы задать условие отбора записей из таблицы, перейдем на закладку конструктора

Условия и перетащим в список условий поле *ЭтоГруппа* из таблицы *Товары*. Платформа автоматически создаст параметризированное условие с этим полем, но в данном случае нам это не нужно. Поэтому зададим произвольное условие отбора (*Товары.ЭтоГруппа = ЛОЖЬ*), установив флажок *Произвольное* (рис. 2.8).

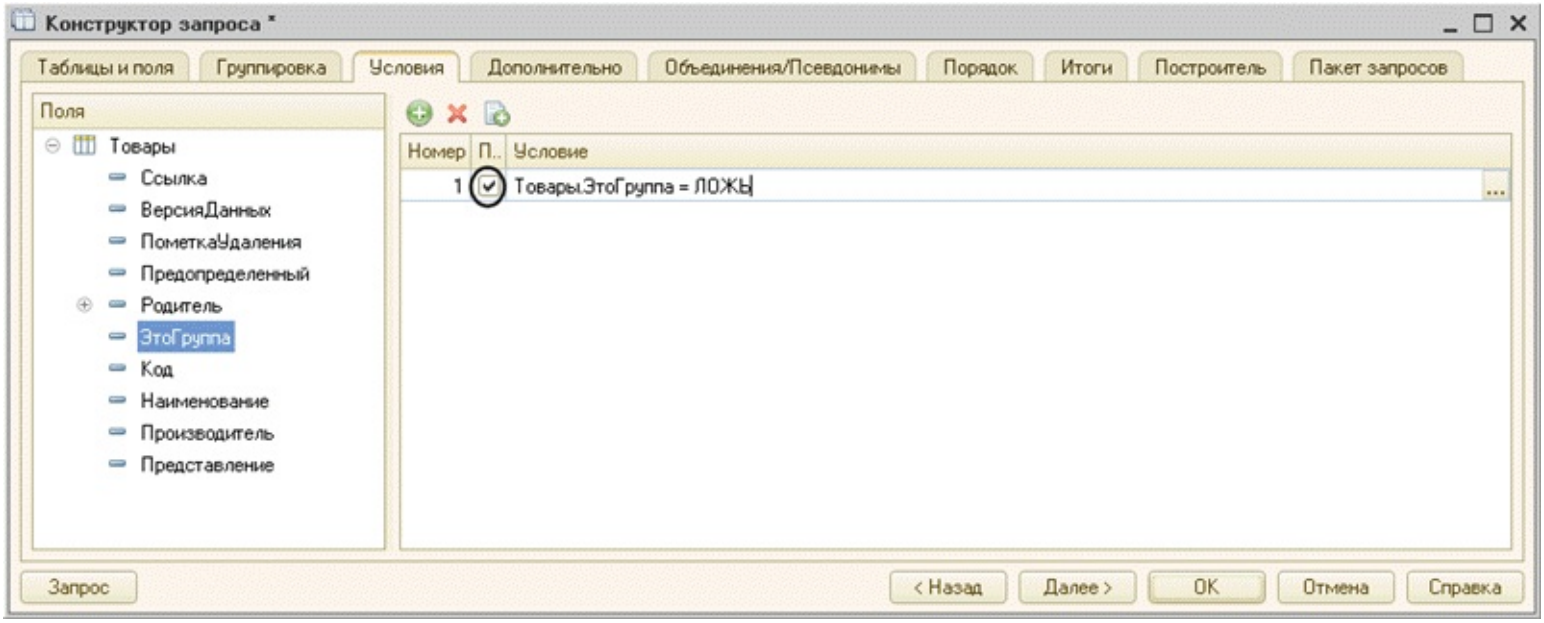


Рис. 2.8. Окно конструктора запроса

Перейдем на закладку конструктора запроса *Порядок* и удалим установленное ранее упорядочивание записей результата запроса (с помощью кнопки <), так как в данном случае оно не нужно.

Нажмем кнопку *Запрос* и посмотрим, как изменился текст запроса (рис. 2.9).

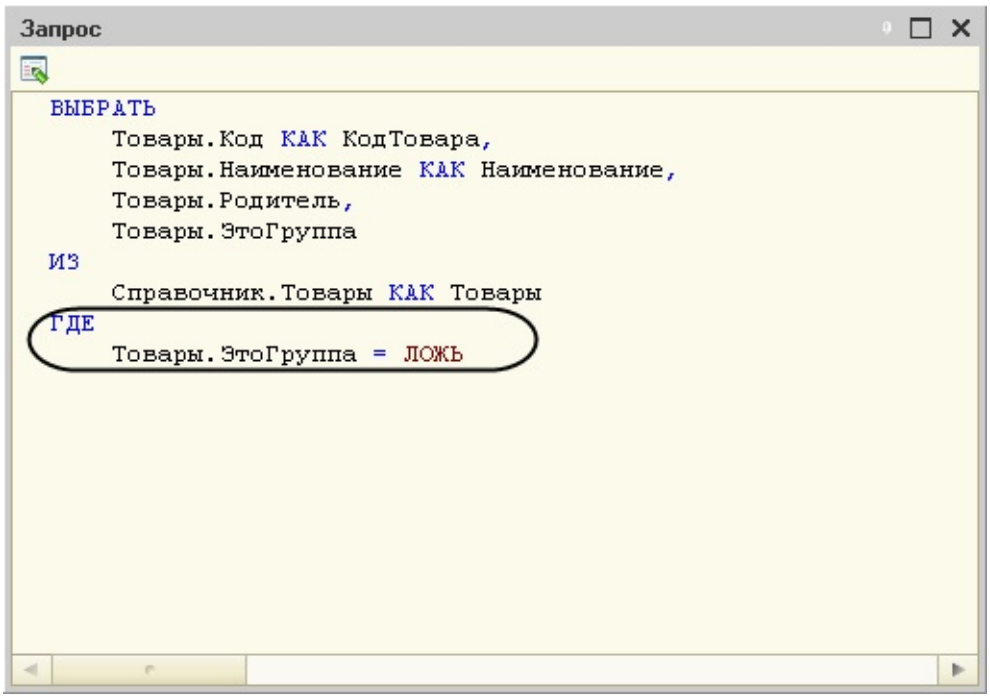


Рис. 2.9. Текст запроса, созданный конструктором

Мы видим, что к тексту запроса конструктор добавляет предложение *ГДЕ*.

Нажмем *ОК* и вернемся в модуль формы. Текст запроса, сформированный с помощью конструктора запроса, заменит собой прежний вариант. В результате мы получим текст запроса, аналогичный тому, который мы писали вручную (см. листинг 2.4).

Связи источников запроса

Теперь создадим с помощью конструктора запрос, выводящий перечень товаров из справочника *Товары* и показывающий актуальную цену для каждого товара. Этот пример мы рассматривали в разделе «Как получить данные из разных таблиц для одного и того же поля» (листинг 2.5).

Листинг 2.5. Левое внешнее соединение данных справочника «Товары» и среза последних записей регистра сведений «Цены»

```
ВЫБРАТЬ
  Товары.Код,
  Товары.Наименование,
  Товары.Производитель,
  Цены.Цена
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК Цены
    ПО Товары.Ссылка = Цены.Товар
```

Из текста запроса мы видим, что в качестве источников запроса здесь выступают две таблицы *Справочник.Товары* и *РегистрСведений.Цены.СрезПоследних*, которые связаны между собой левым внешним соединением.

Для того чтобы получить такой запрос с помощью конструктора запроса, нужно на закладке *Таблицы и поля* перенести обе эти таблицы в список источников запроса и выбрать из таблицы *Товары* поля *Код*, *Наименование*, *Производитель*, а из таблицы *Цены.СрезПоследних* – поле *Цена* (рис. 2.10).

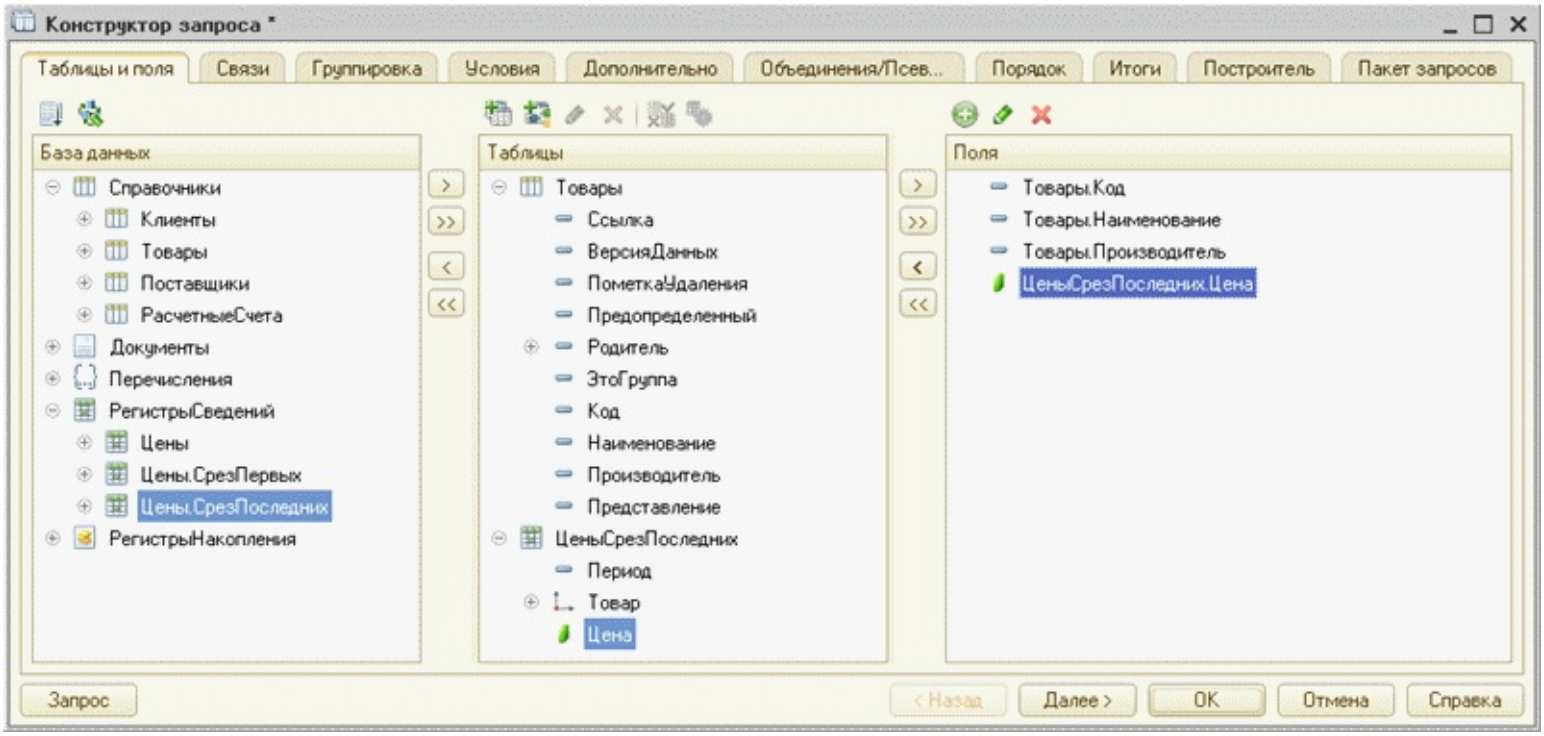


Рис. 2.10. Окно конструктора запроса

В случае, если в запросе участвуют несколько таблиц, в окне конструктора запроса становится доступной закладка *Связи*, на которой задается тип связи и условие связи между таблицами. В тех случаях, когда это возможно, платформа сама устанавливает связь между источниками запроса по ссылочным полям. Нам остается только на закладке *Связи* установить флажок *Все* у таблицы *Товары* и снять его у таблицы *Цены.СрезПоследних* (рис. 2.11).

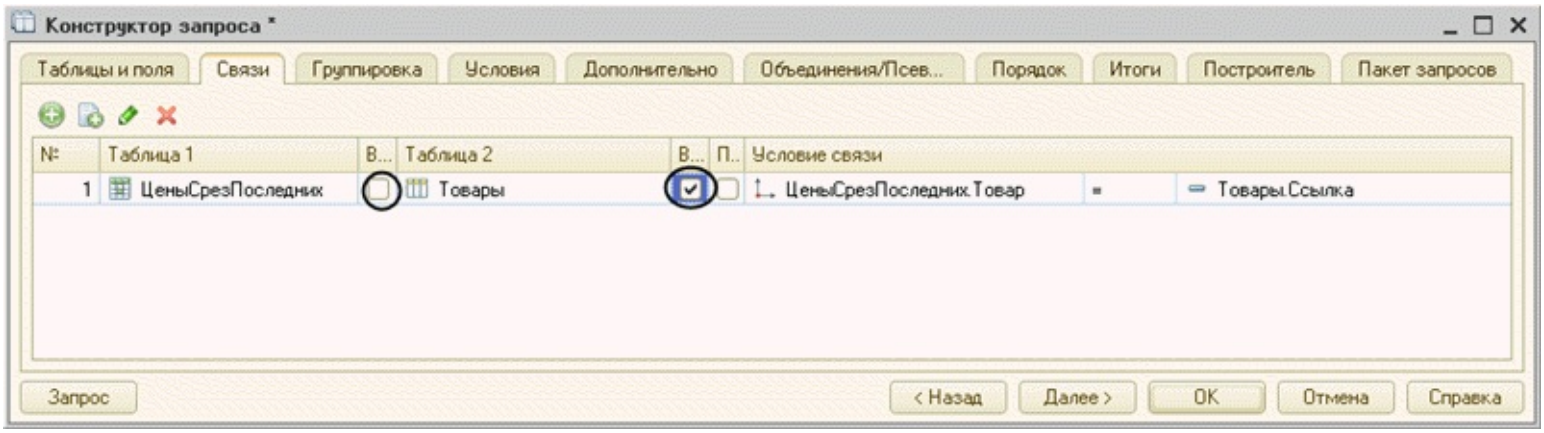


Рис. 2.11. Окно конструктора запроса

Состояние флажка *Все* справа от названия таблицы определяет тип соединения связанных таблиц. Если оба флажка сняты, то задается внутреннее соединение; если оба флажка установлены, то – полное соединение. Наличие установленного флажка *Все* у левой или у правой таблицы задает соответственно левое или правое соединение.

Заметим, что при самостоятельной установке связи двух таблиц (как в данном случае) конструктор запроса автоматически устанавливает флажок *Все* у левой таблицы, тем самым задавая левое соединение таблиц. Но нам нужно, чтобы в запросе слева была таблица *Товары*, поэтому мы изменили состояние флажка *Все* у обеих таблиц. При следующем редактировании текста запроса с помощью конструктора таблица *Товары*

будет находиться слева на закладке *Связи*. Кроме того, мы можем поменять условие связи таблиц или выбрать другую таблицу из списка источников запроса относительно того варианта связи, который был предложен нам конструктором.

Нажмем *ОК* и вернемся в модуль формы. Запрос, сформированный с помощью конструктора запроса, будет выглядеть следующим образом (листинг 2.6).

Листинг 2.6. Запрос, созданный конструктором

```
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
   |         Товары.Код,
   |         Товары.Наименование,
   |         Товары.Производитель,
   |         ЦеныСрезПоследних.Цена
   |ИЗ
   |         Справочник.Товары КАК Товары
   |                 ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цены.СрезПоследних КАК
  ЦеныСрезПоследних
   |                 ПО ЦеныСрезПоследних.Товар = Товары.Ссылка";
```

Таким образом, при описании связи таблиц к тексту запроса конструктор добавляет ключевые слова *ВНУТРЕННЕЕ / ЛЕВОЕ / ПРАВОЕ / ПОЛНОЕ СОЕДИНЕНИЕ… ПО*. В результате мы получили текст запроса, аналогичный тому, который мы писали вручную (см. листинг 2.5).

Объединение запросов

Теперь попробуем выполнить с помощью конструктора запроса более сложную задачу – создадим запрос, объединяющий информацию из двух запросов, в котором записи каждого запроса сгруппированы, а также рассчитаны итоговые данные для результата объединения запросов. Этот пример мы рассматривали в разделе «Как получить данные из разных таблиц, не связывая, а дополняя их» (листинг 2.7).

Листинг 2.7. Объединение данных о заказах товаров клиентами и данных о продажах товаров этим клиентам

```
ВЫБРАТЬ
  Заказ.Ссылка.Клиент КАК Клиент,
  Заказ.Товар КАК Товар,
  СУММА(Заказ.Количество) КАК Заказано,
  СУММА(0) КАК Продано
ИЗ
  Документ.ЗаказТовара.Состав КАК Заказ
СГРУППИРОВАТЬ ПО
  Заказ.Ссылка.Клиент,
  Заказ.Товар

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  Накладная.Ссылка.Покупатель,
  Накладная.Товар,
  СУММА(0),
```

```
  СУММА(Накладная.Количество)
ИЗ
  Документ.РасходнаяНакладная.Состав КАК Накладная
СГРУППИРОВАТЬ ПО
  Накладная.Ссылка.Покупатель,
  Накладная.Товар

ИТОГИ ПО
  ОБЩИЕ,
  Клиент
```

Для того чтобы получить такой запрос с помощью конструктора запроса, нужно сначала задать исходную таблицу и список полей выборки первого запроса, а также на закладке *Группировка* установить порядок группировки записей исходной таблицы запроса. Затем на закладке *Объединения/Псевдонимы* добавить еще один запрос и задать исходную таблицу, список полей выборки и порядок группировки записей второго запроса. После этого на закладке *Итоги* нужно указать, какие итоги требуется рассчитать для результата объединения запросов.

Итак, сформируем первый запрос. На закладке *Таблицы и поля* перенесем табличную часть *Состав* документа *ЗаказТовара* в список источников запроса и выберем из этой таблицы поля: *Товар* и *Количество*. Для того чтобы выбрать поле основной таблицы документа *Клиент*, раскроем поле *Ссылка* табличной части *Состав*, выберем поле *Клиент* и получим обращение через точку от ссылки – *ЗаказТовараСостав.Ссылка.Клиент*. Затем в списке выбранных полей запроса нажмем кнопку *Добавить* и укажем произвольное выражение – «0» (рис. 2.12).

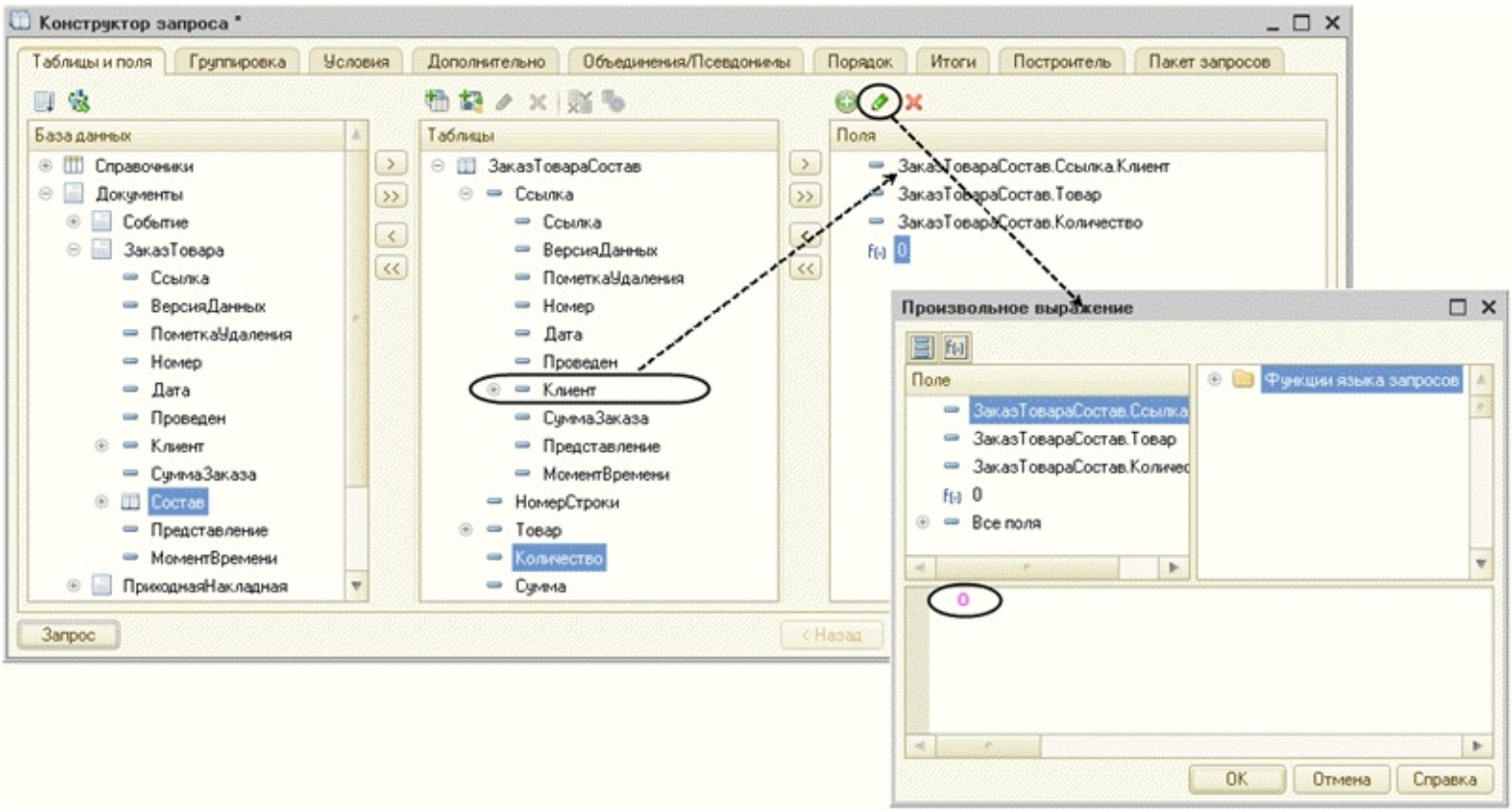


Рис. 2.12. Окно конструктора запроса

ПРИМЕЧАНИЕ

Порядок выбираемых полей очень важен для объединяемых запросов. Он должен соответствовать рисунку 2.12.

Затем перейдем на закладку *Объединения/Псевдонимы* и зададим псевдонимы полей *Количество* и произвольного выражения как *Заказано* и *Продано* соответственно (рис. 2.13).

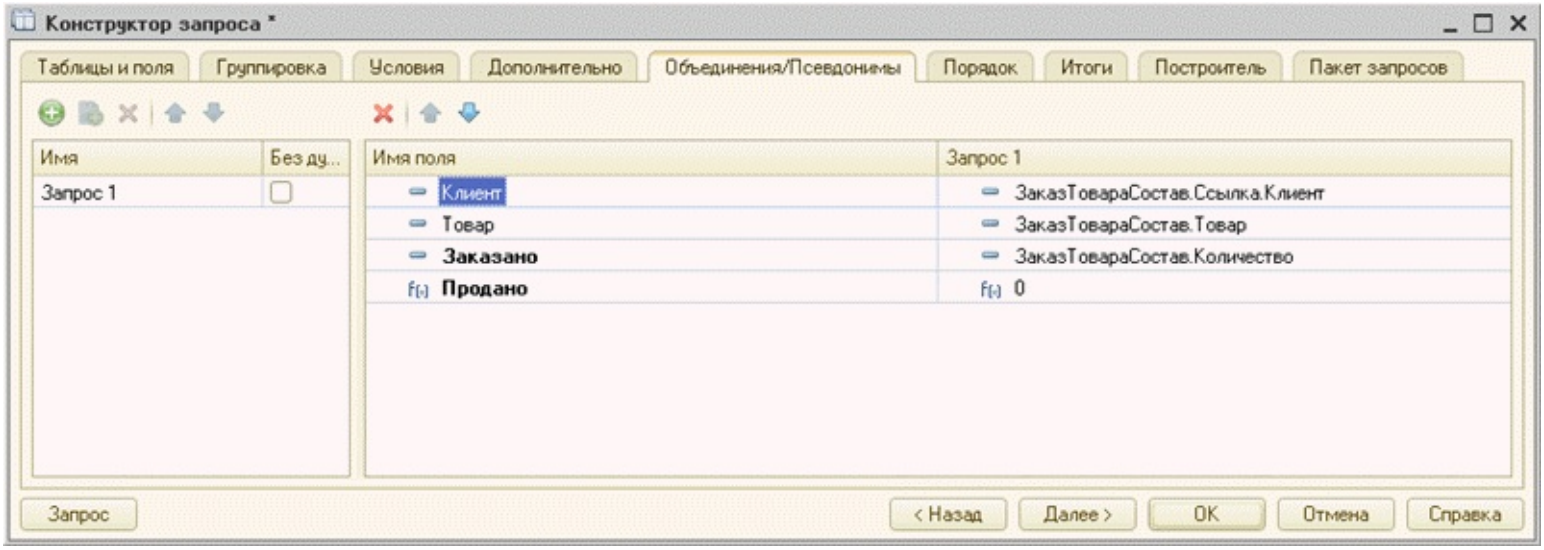


Рис. 2.13. Окно конструктора запроса

Затем перейдем на закладку *Группировка* и зададим порядок группировки записей исходной таблицы запроса. В окно *Групповое поле* перенесем поля выборки запроса *Клиент* и *Товар*, а в окно *Суммируемое поле* перенесем поля *Количество* и *0*. Агрегатная функция, применяемая по умолчанию к суммируемым полям – *СУММА*, но можно выбрать и другие агрегатные функции (рис. 2.14).

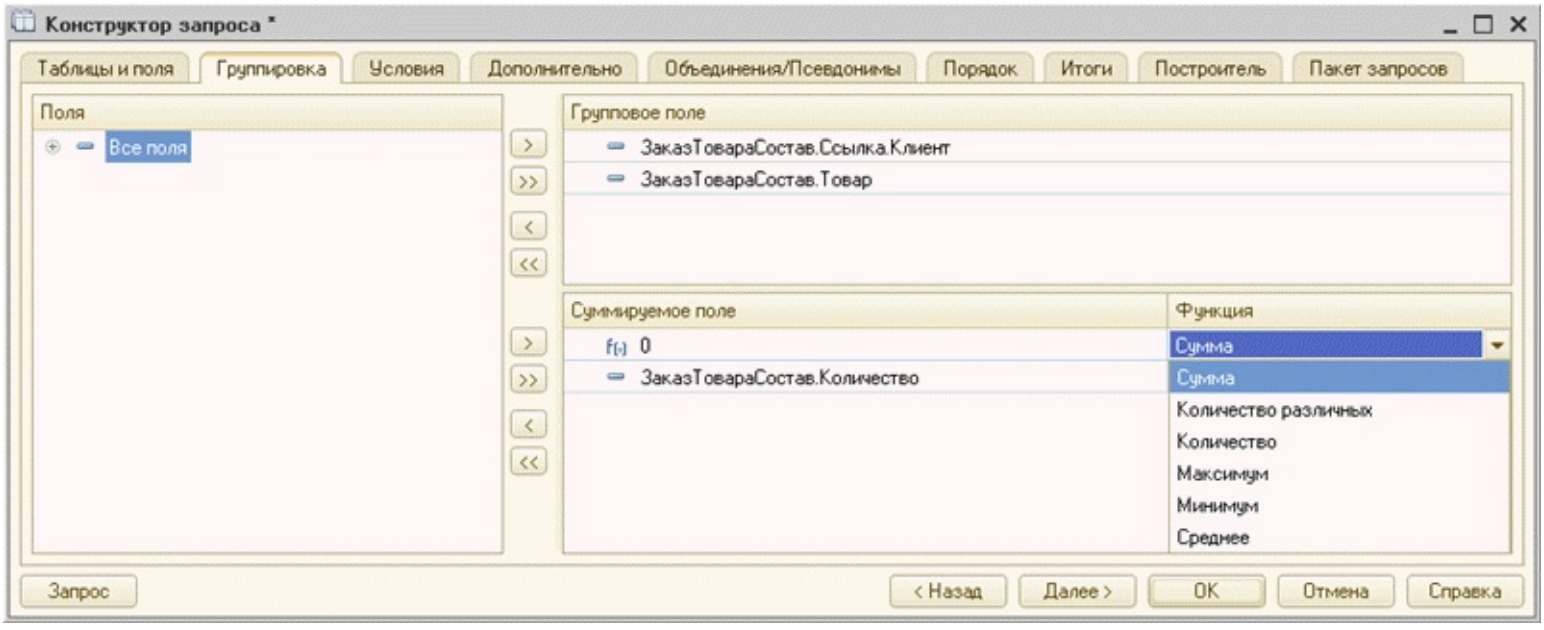


Рис. 2.14. Окно конструктора запроса

Тем самым мы указали, что записи исходной таблицы запроса *ЗаказТовараСостав* нужно сгруппировать по полям *Клиент* и *Товар*, а в результате группировки вывести суммарное

значение полей *Количество* и *0*.

Нажмем кнопку *Запрос* и посмотрим, как изменился текст запроса (рис. 2.15).

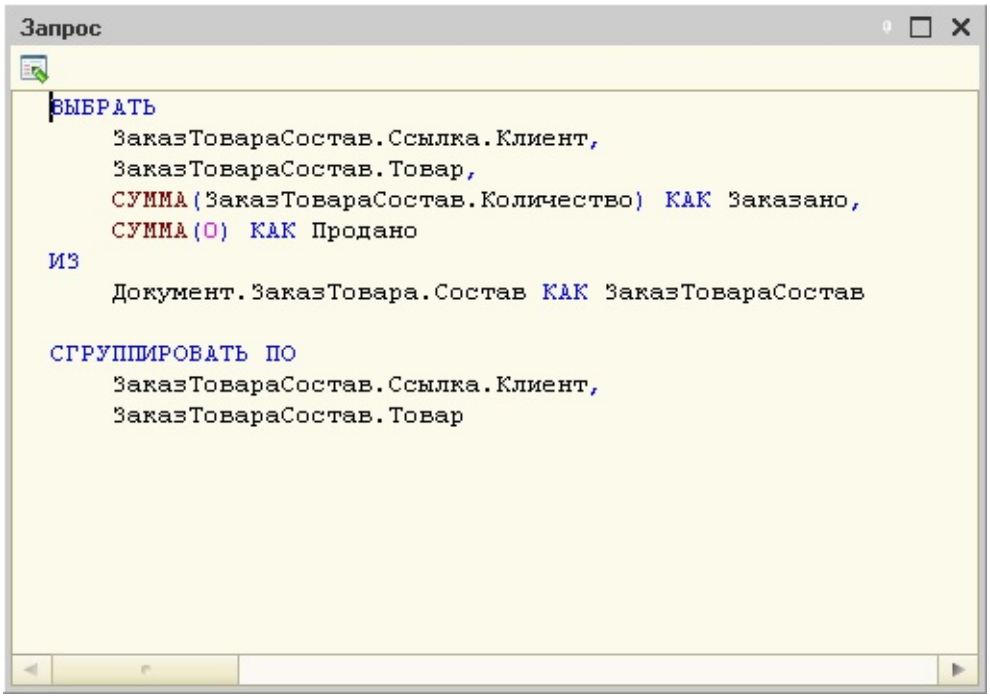


Рис. 2.15. Текст запроса, созданный конструктором

Итак, мы полностью сформировали описание первого запроса, входящего в объединение запросов.

Теперь объединим с этим запросом еще один запрос. Для этого перейдем на закладку *Объединения/Псевдонимы*, в списке запросов (слева) нажмем кнопку *Добавить* и добавим запрос. После этого на закладке *Таблицы и поля*, а также на закладках *Группировка*, *Условия*, *Дополнительно* станут доступны отдельные вкладки для каждого запроса.

Определим исходные данные для второго запроса. Перенесем табличную часть *Состав* документа *РасходнаяНакладная* в список источников запроса и выберем из этой таблицы поля: *Товар* и *Количество*. Для того чтобы выбрать поле основной таблицы документа *Покупатель*, раскроем поле *Ссылка* табличной части *Состав*, выберем поле *Покупатель* и получим обращение через точку от ссылки – *РасходнаяНакладнаяСостав.Ссылка.Покупатель*. Также в списке выбранных полей запроса нажмем кнопку *Добавить* и укажем произвольное выражение – «0» (рис. 2.16).

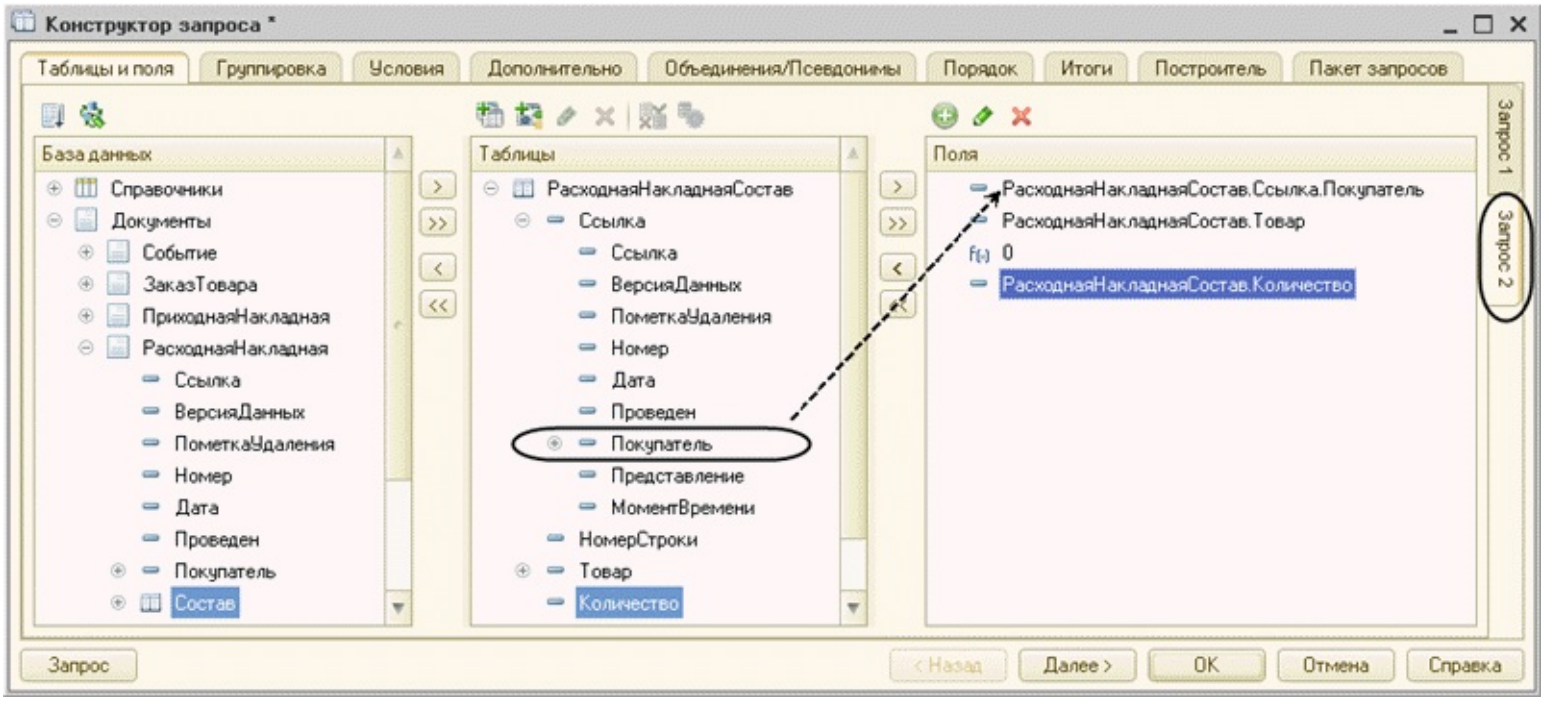


Рис. 2.16. Окно конструктора запроса

ПРИМЕЧАНИЕ

Порядок выбираемых полей очень важен для объединяемых запросов. Он должен соответствовать рисунку 2.16.

На закладке *Группировка*, на соответствующей вкладке, зададим порядок группировки записей исходной таблицы второго запроса. В окно *Групповое поле* перенесем поля выборки запроса *Покупатель* и *Товар*, а в окно *Суммируемое поле* перенесем поля *Количество* и *0*. Тем самым мы указали, что записи исходной таблицы запроса *РасходнаяНакладнаяСостав* нужно сгруппировать по полям *Покупатель* и *Товар*, а в результате группировки вывести суммарное значение полей *Количество* и *0* (рис. 2.17).

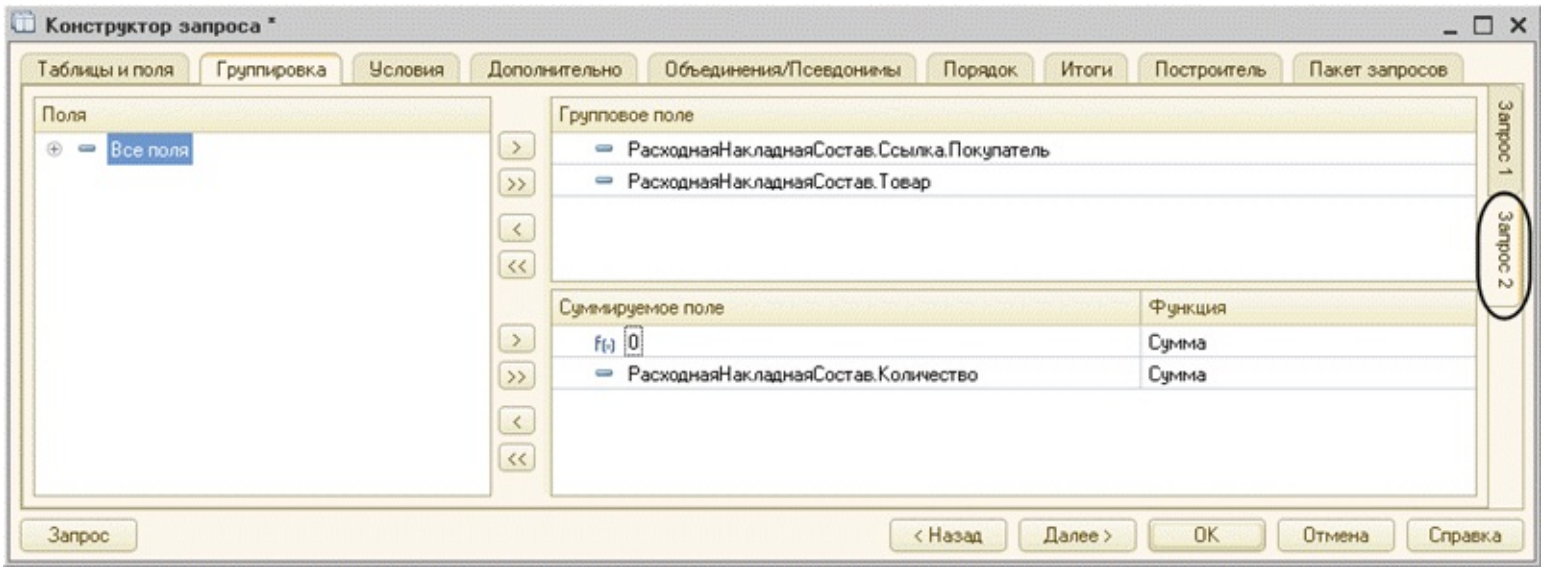


Рис. 2.17. Окно конструктора запроса

Теперь перейдем на закладку *Объединения/Псевдонимы* и сопоставим поля второго запроса полям первого запроса, в частности поле *Клиент* первого запроса и поле *Покупатель* второго запроса (рис. 2.18).

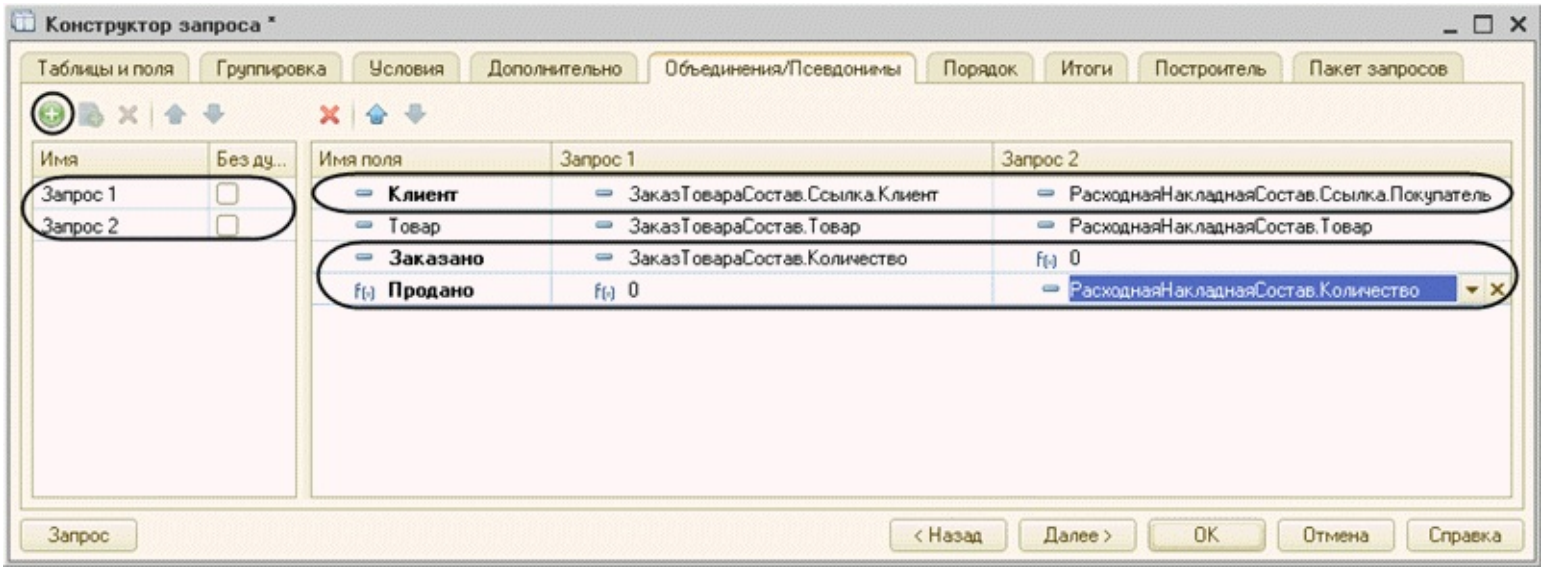


Рис. 2.18. Окно конструктора запроса

Затем перейдем на закладку *Итоги* и укажем, какие итоги нужно рассчитать для результата объединения запросов. В окно *Группировочное поле* перенесем поле выборки запроса *Клиент*, а окно *Итоговое поле* можно оставить пустым, так как список итоговых полей будет автоматически формироваться из агрегатных полей списка выборки (с псевдонимами *Заказано* и *Продано*). А также установим флажок *Общие итоги* (рис. 2.19).

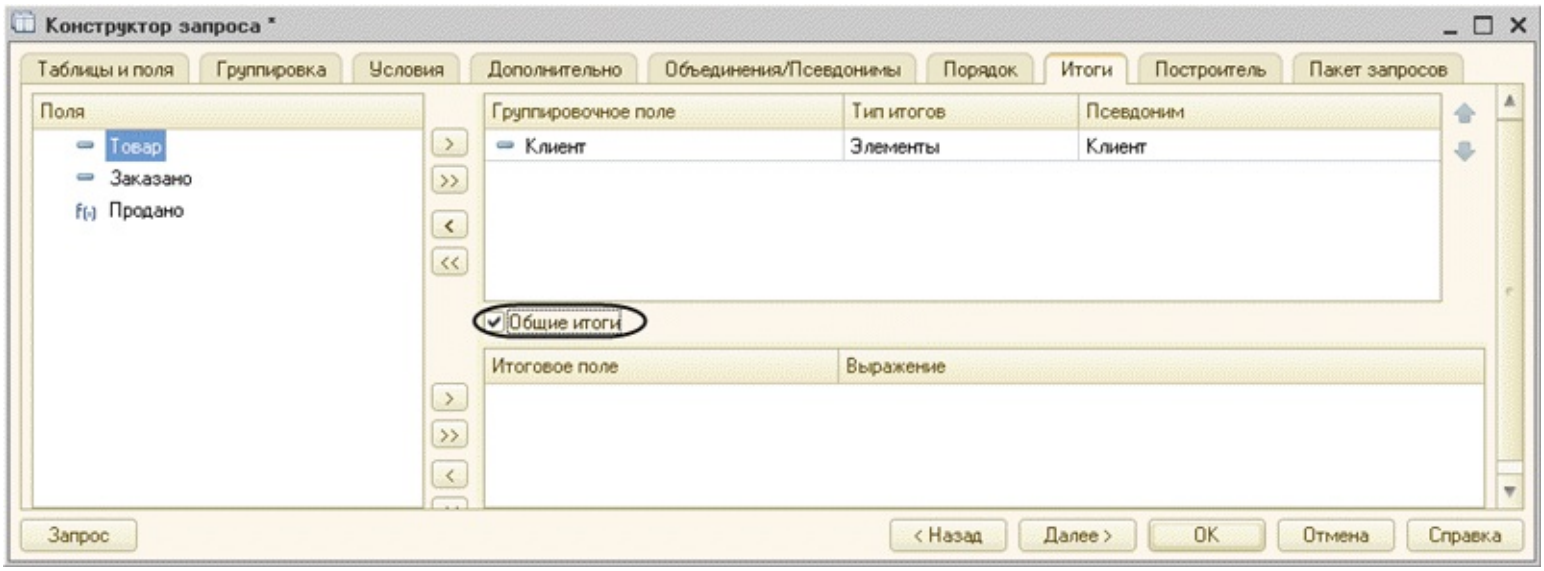


Рис. 2.19. Окно конструктора запроса

Тем самым мы указали, что для каждого клиента из документа *ЗаказТовара* и для каждого покупателя из документа *РасходнаяНакладная*, а также для всего результата запроса в целом нужно подсчитать суммарное количество полей *Заказано* и *Продано*.

Нажмем кнопку *Запрос* и посмотрим, как изменился текст запроса (рис. 2.20).

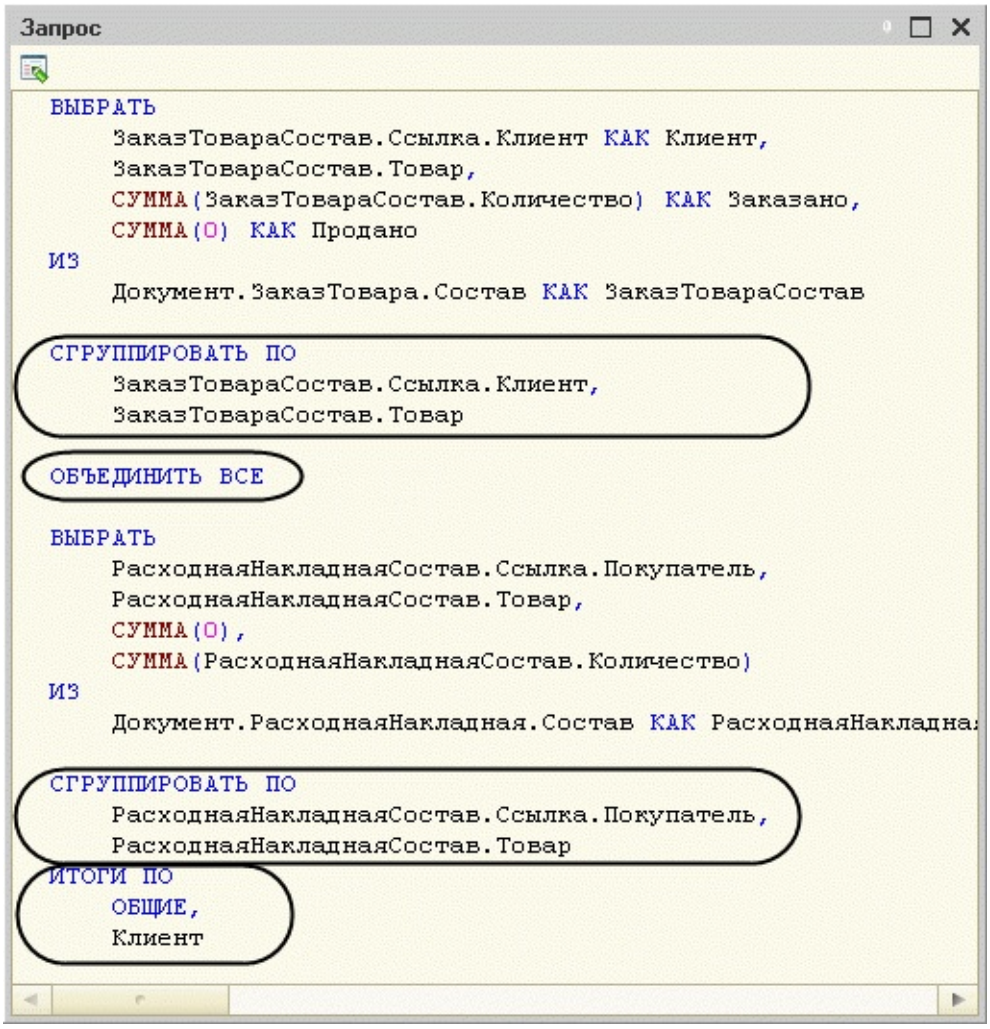


Рис. 2.20. Текст запроса, созданный конструктором

Итак, в результате наших действий на закладках *Объединения/Псевдонимы*, *Группировка* и *Итоги* конструктор запроса добавил к тексту запроса предложения *ОБЪЕДИНИТЬ ВСЕ*, *СГРУППИРОВАТЬ ПО* и *ИТОГИ ПО*. В итоге мы получили текст запроса, аналогичный тому, который мы писали вручную (см. листинг 2.7).

Создание пакетного запроса, использующего временную таблицу

В этом разделе мы покажем, как с помощью конструктора запроса создать временную таблицу и затем использовать ее в пакетном запросе. Этот пример мы рассматривали в разделе «Временные таблицы и пакетные запросы» (листинг 2.8).

Листинг 2.8. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
ВЫБРАТЬ
  Поступление.Товар,
  Поступление.Ссылка.Дата,
  Поступление.Ссылка.Поставщик
ПОМЕСТИТЬ ПоступлениеТоваров
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК Поступление
ГДЕ
  МЕСЯЦ(Поступление.Ссылка.Дата) = 11
;
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
```

```
  ПоступлениеТоваров.Дата,
  ПоступлениеТоваров.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ ПоступлениеТоваров КАК ПоступлениеТоваров
    ПО Товары.Ссылка = ПоступлениеТоваров.Товар

УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

Для того чтобы получить такой запрос с помощью конструктора запроса, нужно сначала сформировать первый запрос для заполнения данными временной таблицы. Затем на закладке *Пакет запросов* добавить в пакет запросов запрос, который будет использовать данные этой таблицы, после этого на закладке *Таблицы и поля* создать описание этой временной таблицы и использовать ее как источник второго запроса.

Итак, создадим запрос для получения временной таблицы, содержащей информацию о поступлении товаров за ноябрь. На закладке *Таблицы и поля* перенесем табличную часть *Состав* документа *ПриходнаяНакладная* в список источников запроса и выберем из этой таблицы поле *Товар*. Для того чтобы выбрать поля основной таблицы документа *Дата* и *Поставщик*, раскроем поле *Ссылка* табличной части *Состав*, выберем эти поля и получим обращение через точку от ссылки – *ПриходнаяНакладнаяСостав.Ссылка.Дата*, *ПриходнаяНакладнаяСостав.Поставщик* (рис. 2.21).

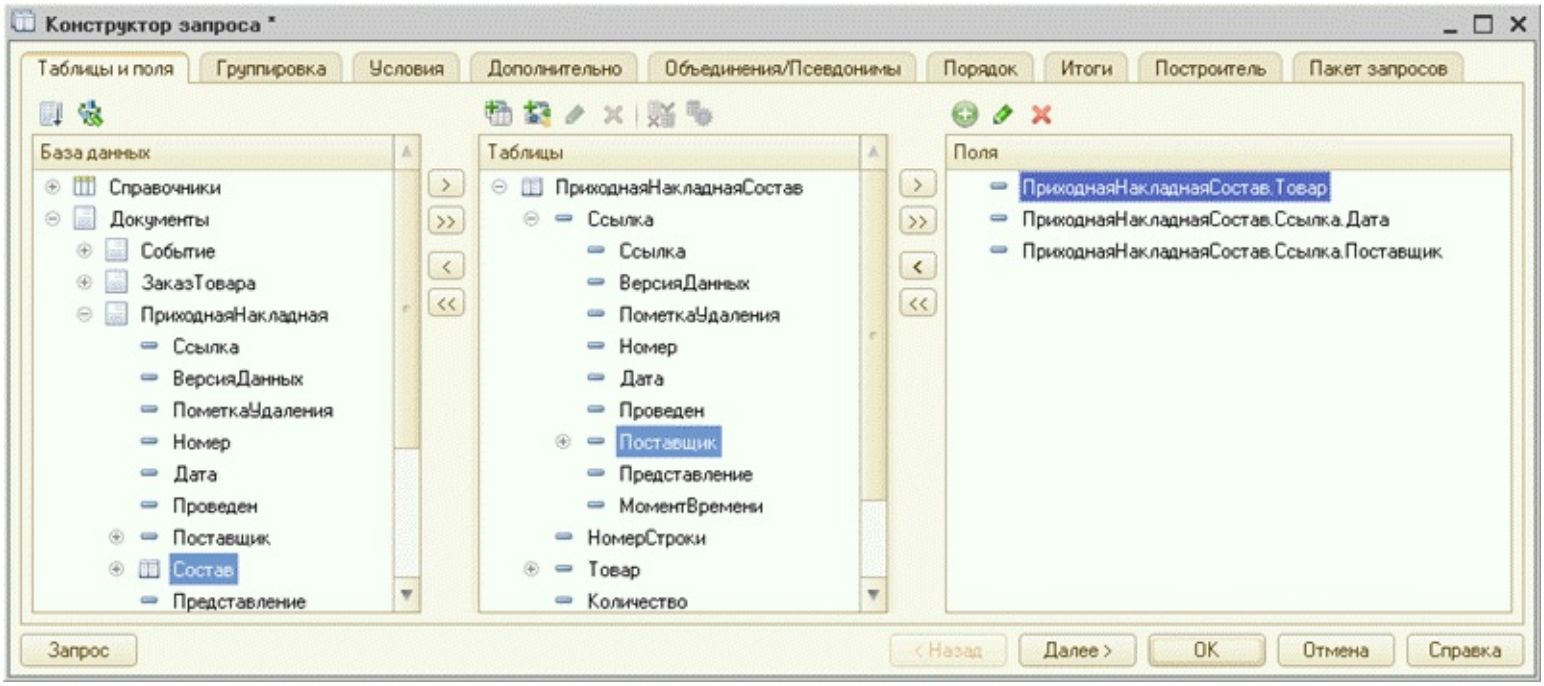


Рис. 2.21. Окно конструктора запроса

На закладке *Условия* зададим произвольное условие отбора записей из приходных накладных так, чтобы во временную таблицу попадали только данные о поступлении товаров за ноябрь (рис. 2.22).

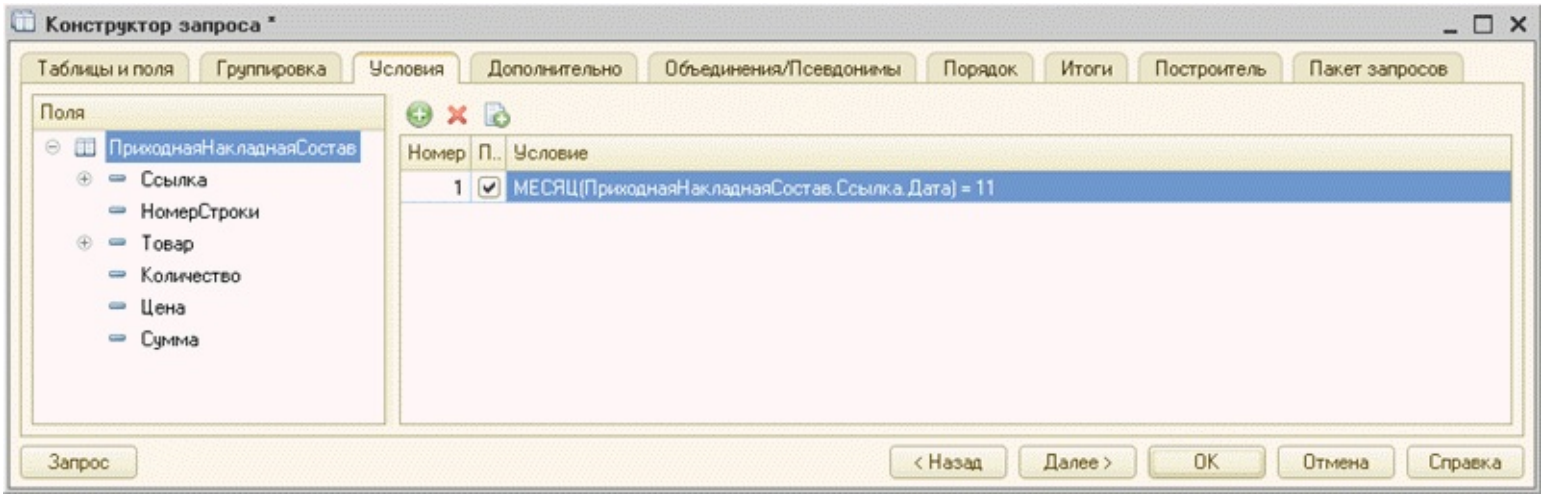


Рис. 2.22. Окно конструктора запроса

На закладке *Дополнительно* включим опцию *Создание временной таблицы* и укажем, что результат запроса нужно поместить во временную таблицу с именем *ПоступлениеТоваров* (рис. 2.23).

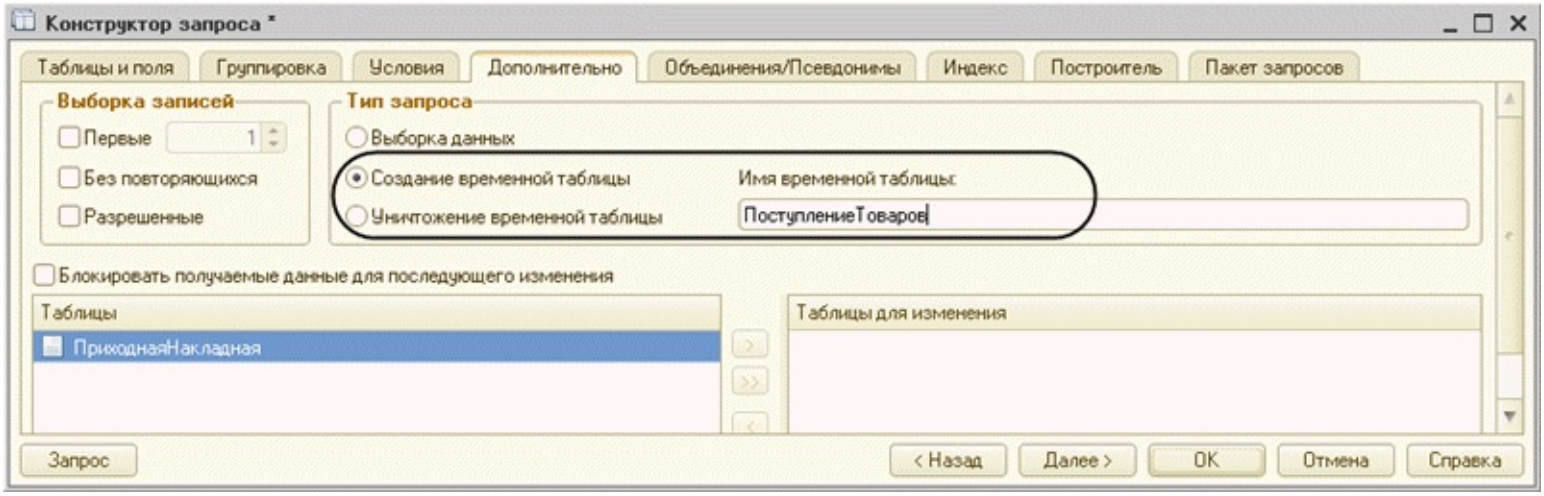


Рис. 2.23. Окно конструктора запроса

Обратите внимание, что при создании временной таблицы в конструкторе запроса исчезают закладки *Порядок* и *Итоги* и появляется закладка *Индекс*, на которой можно задать сортировку записей временной таблицы.

Кроме того, на закладке *Дополнительно* можно задать дополнительные критерии отбора записей в результат запроса (флажки *Первые*, *Без повторяющихся*, *Разрешенные*) и задать необходимость блокировки таблиц – источников запроса.

Нажмем кнопку *Запрос* и посмотрим, как изменился текст запроса (рис. 2.24).

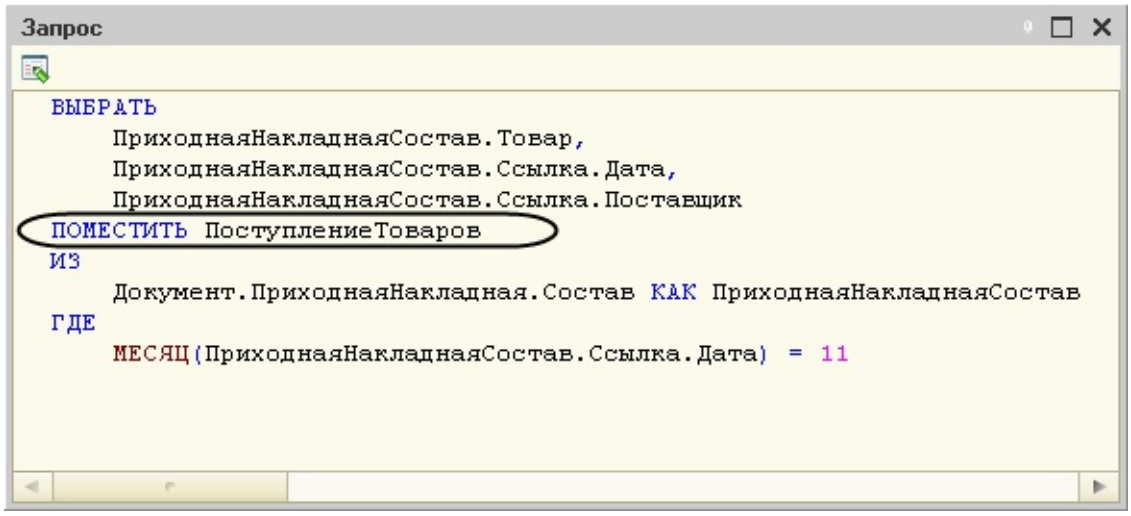


Рис. 2.24. Текст запроса, созданный конструктором

Мы видим, что в тексте запроса появилась конструкция *ПОМЕСТИТЬ*, с помощью которой данные помещаются во временную таблицу. Итак, мы сформировали первый запрос из пакетного запроса (см. листинг 2.8).

Теперь перейдем на закладку *Пакет запросов*. Мы видим, что в пакете запросов уже присутствует один запрос *ПоступлениеТоваров* для получения данных временной таблицы. Нажмем кнопку *Добавить* и добавим в пакет запросов еще один запрос. После этого на закладке *Таблицы и поля*, а также на закладках *Группировка*, *Условия*, *Дополнительно*, *Объединения/Псевдонимы* станут доступны отдельные вкладки для каждого запроса из пакета (рис. 2.25).

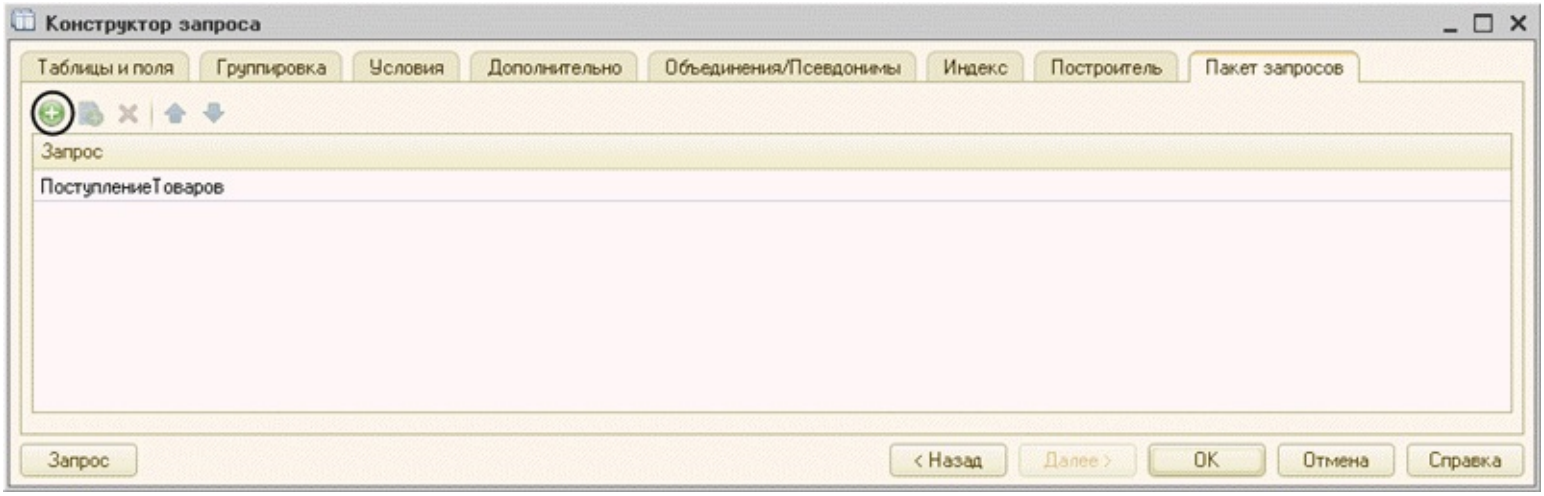


Рис. 2.25. Окно конструктора запроса

На закладке *Таблицы и поля* нажмем кнопку *Создать описание временной таблицы* и зададим имя временной таблицы и поля, указанные в первом запросе (рис. 2.26).

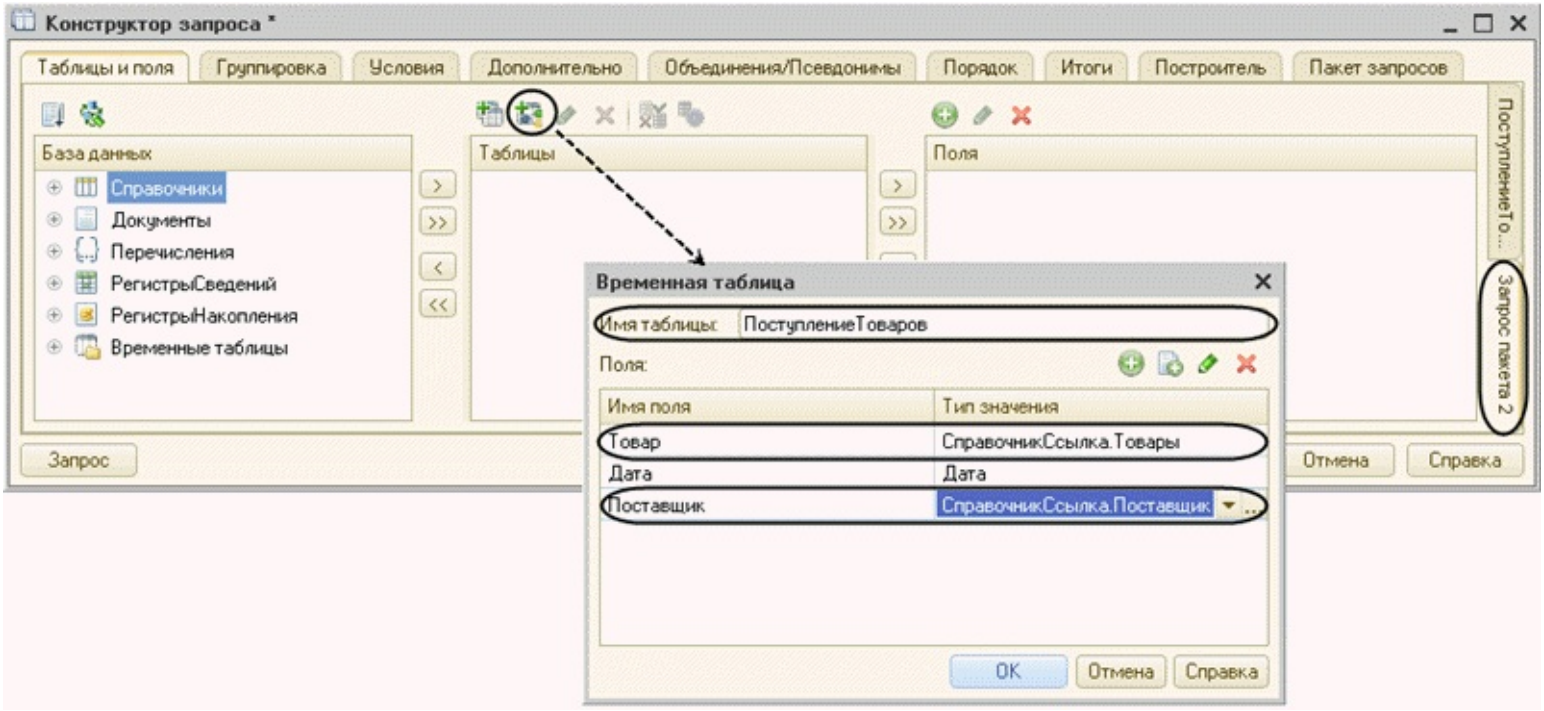


Рис. 2.26. Окно конструктора запроса

Обратите внимание, что нужно указать также тип значения полей временной таблицы. Например, в нашей таблице поля *Товар* и *Поставщик* имеют тип ссылки на справочник. Тогда при выборе полей из временной таблицы мы можем выбирать также поля исходной таблицы, на которую ссылается это ссылочное поле, например, *ПоступлениеТоваров.Товар.Наименование*.

После описания временной таблицы она попадает в список источников второго запроса, и мы можем выбирать из нее поля, как из любой другой таблицы базы данных.

Сформируем исходные данные для второго запроса. Перенесем справочник *Товары* в список источников запроса и выберем из этой таблицы поля *Наименование* и *Производитель*. Из временной таблицы *ПоступлениеТоваров* выберем поля *Дата* и *Поставщик* (рис. 2.27).

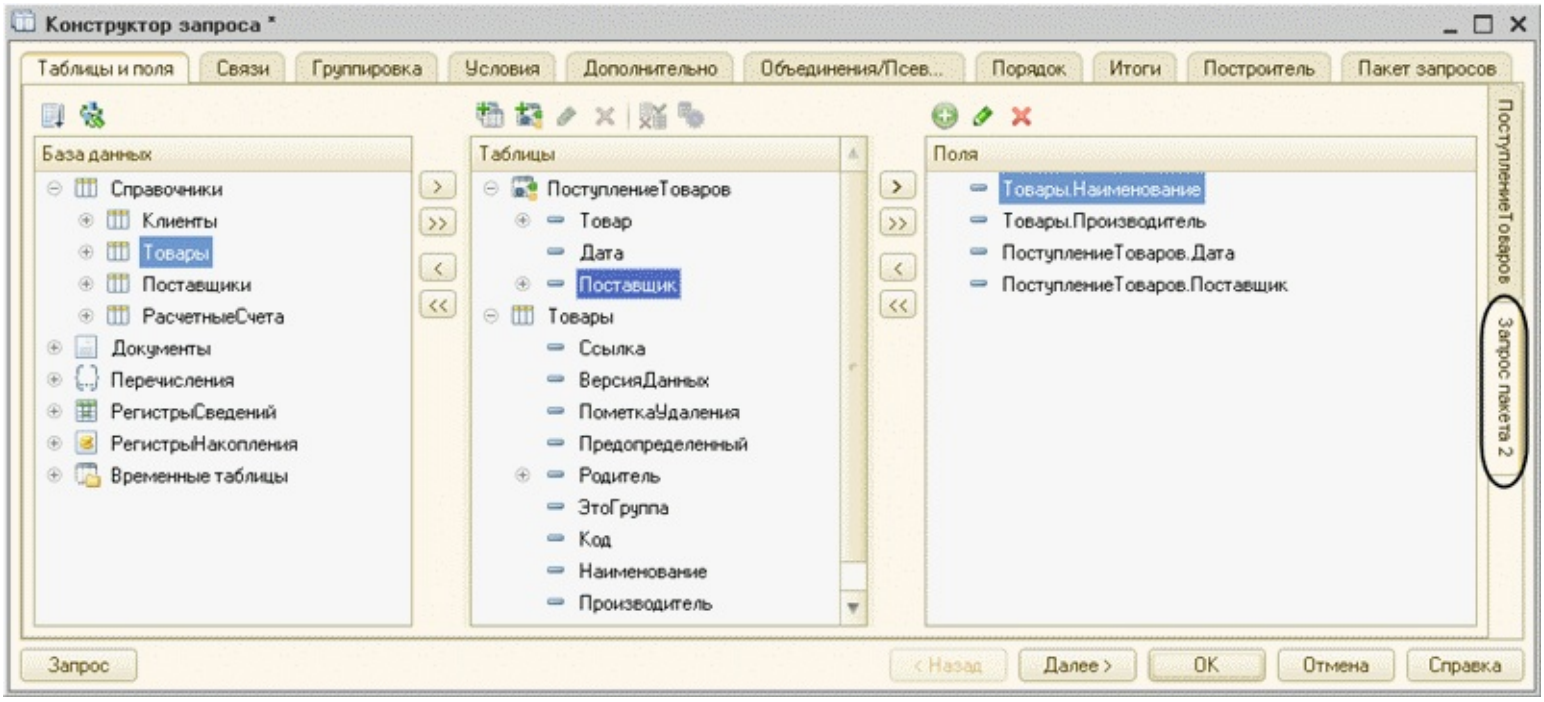


Рис. 2.27. Окно конструктора запроса

Поскольку в списке источников запроса содержатся две таблицы, свяжем левым соединением таблицу *Товары* с временной таблицей *ПоступлениеТоваров* (рис. 2.28).

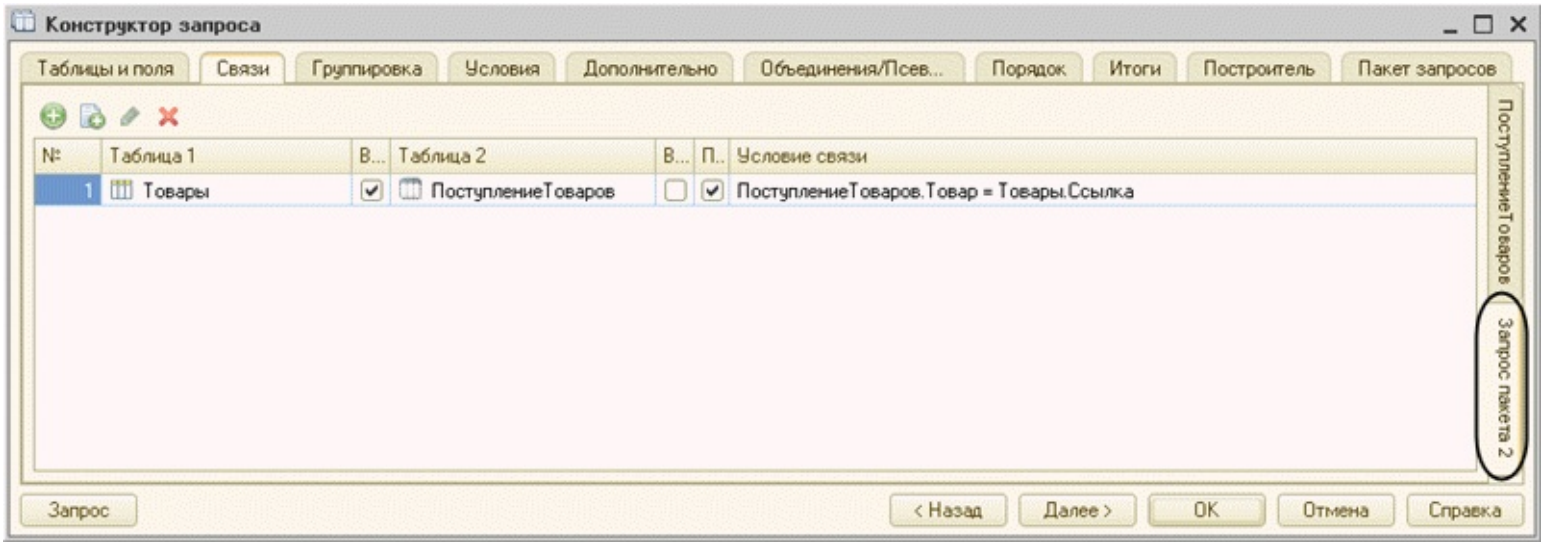


Рис. 2.28. Окно конструктора запроса

подробнее

Раздел «Связи источников запроса».

В заключение на закладке *Порядок* зададим упорядочивание записей результата запроса по наименованиям товаров в порядке иерархии (рис. 2.29).

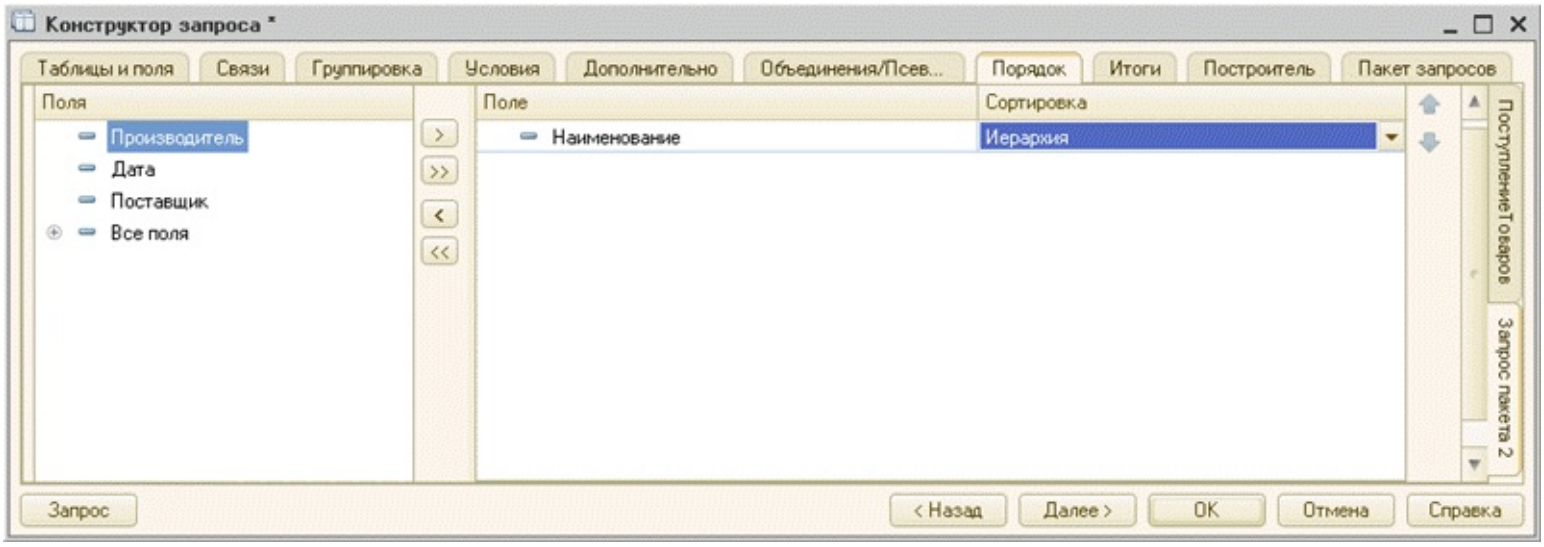


Рис. 2.29. Окно конструктора запроса

Поскольку при описании временной таблицы *ПоступлениеТоваров* мы задали тип значения ее полей, то мы могли бы выбрать все нужные для второго запроса поля из этой таблицы, не используя таблицу *Товары* (рис. 2.30).

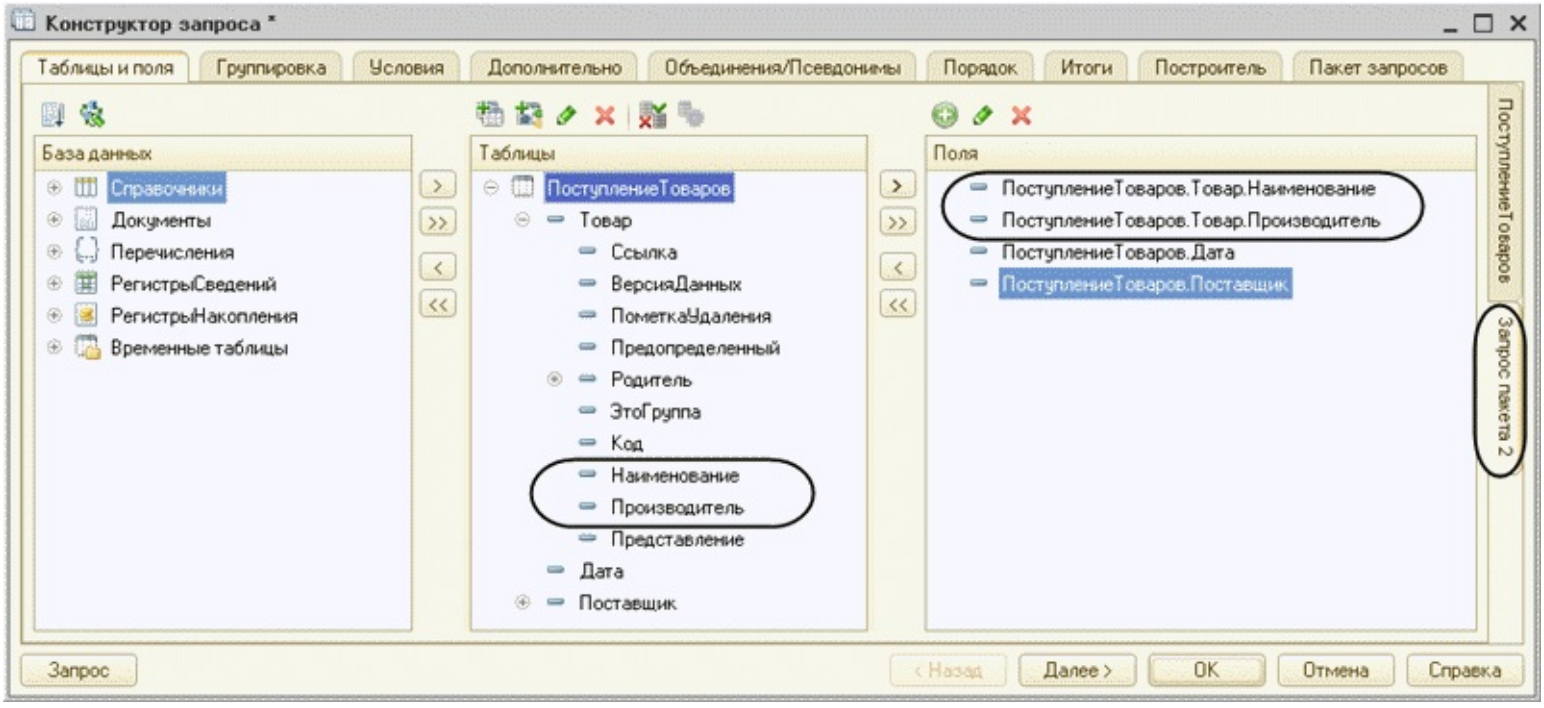


Рис. 2.30. Окно конструктора запроса

Однако тогда мы не сможем выполнить иерархическое упорядочивание записей результата запроса, так как у нас не будет данных для построения иерархии. Поэтому в данном случае так делать не нужно.

Нажмем *ОК* и вернемся в модуль формы. Запрос, сформированный с помощью конструктора запроса, будет выглядеть следующим образом (листинг 2.9).

Листинг 2.9. Запрос, созданный конструктором

```
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
   |         ПриходнаяНакладнаяСостав.Товар,
```

```
|        ПриходнаяНакладнаяСостав.Ссылка.Дата,
|        ПриходнаяНакладнаяСостав.Ссылка.Поставщик
|ПОМЕСТИТЬ ПоступлениеТоваров
|ИЗ
|        Документ.ПриходнаяНакладная.Состав КАК ПриходнаяНакладнаяСостав
|ГДЕ
|        МЕСЯЦ(ПриходнаяНакладнаяСостав.Ссылка.Дата) = 11
|;
|
|////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|        Товары.Наименование КАК Наименование,
|        Товары.Производитель,
|        ПоступлениеТоваров.Дата,
|        ПоступлениеТоваров.Поставщик
|ИЗ
|        Справочник.Товары КАК Товары
|                ЛЕВОЕ СОЕДИНЕНИЕ ПоступлениеТоваров КАК ПоступлениеТоваров
|                ПО (ПоступлениеТоваров.Товар = Товары.Ссылка)
|
|УПОРЯДОЧИТЬ ПО
|        Наименование ИЕРАРХИЯ";
```

В результате мы получили текст пакетного запроса, аналогичный тому, который мы писали вручную (см. листинг 2.8).

Выполнение запросов из встроенного языка

На данном этапе изучения языка запросов мы можем, наконец, более подробно остановиться на том моменте, который раньше опускали в целях упрощения восприятия материала. Речь пойдет о выполнении запросов и обработке их результатов во встроенном языке.

В разделе «Общая схема выполнения запросов» мы рассматривали в самом простейшем виде схему выполнения запроса. Теперь рассмотрим ее более подробно. Выполнение запроса во встроенном языке состоит из следующих этапов:

- Создание объекта *Запрос* с нужным текстом запроса на языке запросов.
- Установка параметров запроса с помощью метода *УстановитьПараметр*.
- Выполнение запроса.

Затем либо:

- Получение выборки из результата запроса.
- Обход выборки и обработка данных, то есть выполнение действий, для которых был нужен запрос, например, вывод области при формировании табличного документа.

Либо:

1. Выгрузка результата в таблицу значений или дерево значений.

2. Обработка данных таблицы/дерева значений (например, перебор строк).

Графически это можно представить следующим образом (рис. 2.31):

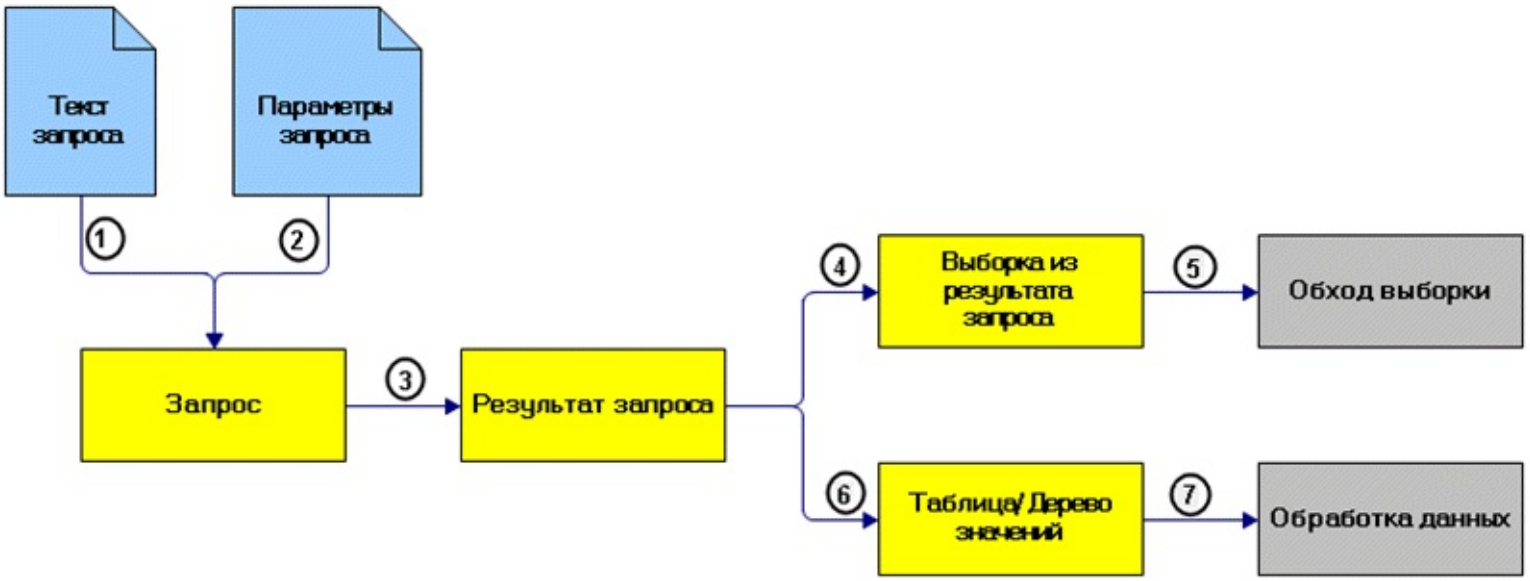


Рис. 2.31. Схема выполнения запроса

Для формирования и выполнения запроса, а также для получения и обработки его результатов во встроенном языке предназначены следующие программные объекты:

- *Запрос*,
- *РезультатЗапроса*,
- *ВыборкаИзРезультатаЗапроса*.

Важно помнить, что вся работа с запросами выполняется только на сервере.

Создание запроса

Прежде всего, во встроенном языке нужно создать объект *Запрос*. Затем, используя свойство *Текст* объекта *Запрос*, нужно поместить в него текст запроса, написанный на языке запросов. В тексте запроса описывается, какие данные, из каких таблиц нужно получить и как эти данные представить. Например, в следующем фрагменте кода создается программный объект *Запрос*, в тексте которого описывается, что нужно извлечь значения полей *Наименование*, *Адрес* и *Телефон* для всех записей справочника *Клиенты* (листинг 2.10).

Листинг 2.10. Пример запроса, созданного во встроенном языке

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |         Клиенты.Наименование КАК Наименование,
  |         Клиенты.Адрес,
  |         Клиенты.Телефон
  |ИЗ
  |         Справочник.Клиенты КАК Клиенты";
```

В начале второй главы в разделе «Конструктор запроса» мы научились создавать с помощью конструктора объект *Запрос* с нужным текстом запроса на языке запросов. При этом в тексте модуля формы, обработки и т. п. конструктор запроса создавал соответствующий фрагмент кода. Конечно, можно подобный код написать и самостоятельно, но если запрос достаточно большой, конструктор экономит время разработчика и избавляет от необходимости следить за всякими тонкостями синтаксиса языка запросов.

Таким образом, первый этап выполнения запроса, показанный на схеме (см. рис. 2.31), нам уже хорошо знаком.

Передача параметров в запрос

Теперь рассмотрим подробно второй этап выполнения запроса, показанный на схеме (см. рис. 2.31).

Допустим, мы хотим добавить в запрос, получающий все записи из справочника *Клиенты* (см. листинг 2.10), параметризированное условие так, чтобы видеть только тех клиентов, в наименовании которых встречается определенная подстрока. Этот пример мы рассматривали в разделе «Как задать произвольное значение отбора записей из таблицы» (листинг 2.11).

Листинг 2.11. Отбор записей из справочника «Клиенты» по параметризированному условию

```
ВЫБРАТЬ
  Клиенты.Наименование КАК Наименование,
  Клиенты.Адрес,
  Клиенты.Телефон
ИЗ
  Справочник.Клиенты КАК Клиенты
ГДЕ
  Наименование ПОДОБНО "%" + &ЧастьНаименования + "%"
```

В этом запросе мы использовали параметр *ЧастьНаименования*. Символ «%» заменяет в шаблоне строки любую последовательность символов, а значение параметра *&ЧастьНаименования* будет содержать подстроку для поиска в наименовании клиента.

Однако приведенный текст запроса (см. листинг 2.11) мы писали и выполняли в консоли запросов. Чтобы создать нужный фрагмент кода на встроенном языке, поместим курсор внутрь текста запроса без условия (см. листинг 2.10) и откроем конструктор запроса. Затем на закладке *Условия* создадим требуемое параметризированное условие (рис. 2.32).

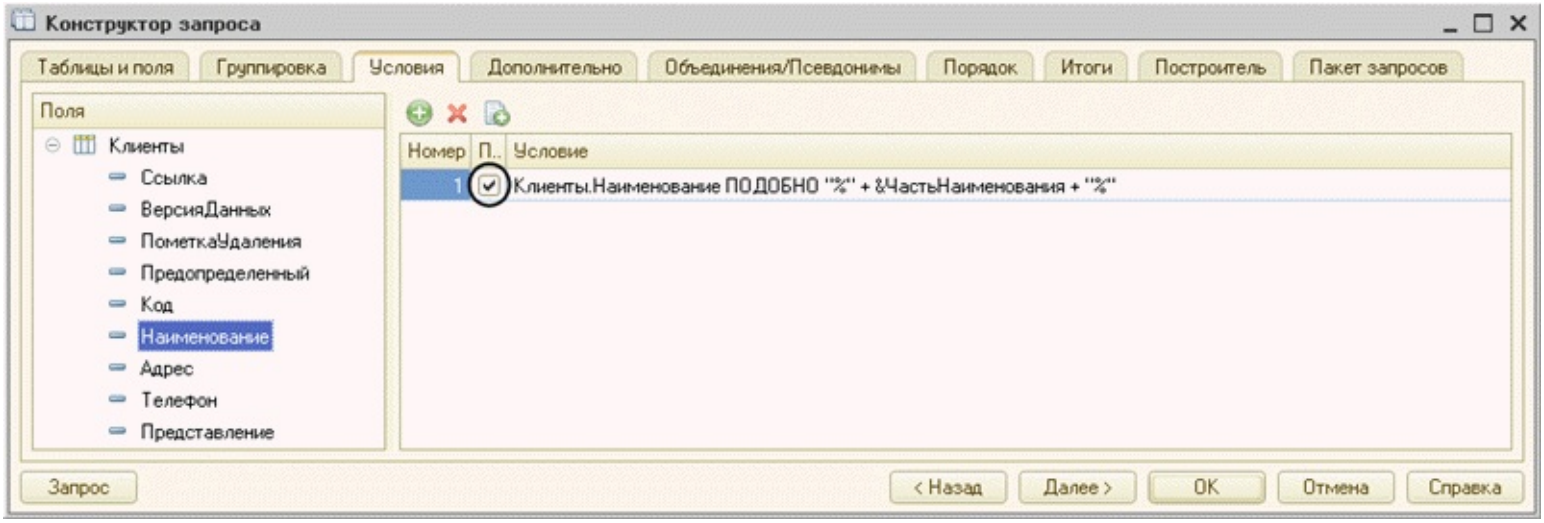


Рис. 2.32. Окно конструктора запроса

В результате конструктор запроса создаст следующий запрос (листинг 2.12).

Листинг 2.12. Запрос, созданный конструктором

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |         Клиенты.Наименование КАК Наименование,
  |         Клиенты.Адрес,
  |         Клиенты.Телефон
  |ИЗ
  |         Справочник.Клиенты КАК Клиенты
  |ГДЕ
  |         Наименование ПОДОБНО ""%"" + &ЧастьНаименования + ""%""";
```

Обратите внимание, что в процедуре на встроенном языке, в отличие от текста запроса в консоли запросов, каждую кавычку («") в шаблоне строки нужно удваивать (листинг 2.13).

Листинг 2.13. Запись условия отбора

```
// В консоли запросов
ГДЕ
  Наименование ПОДОБНО "%" + &ЧастьНаименования + "%"

// В процедуре на встроенном языке
|ГДЕ
|         Наименование ПОДОБНО ""%"" + &ЧастьНаименования + ""%""";
```

В условии отбора мы используем литерал строкового типа (*"%"*), который представляет собой набор символов, заключенных в кавычки. Во встроенном языке для задания в строке символа «"» (кавычка) необходимо записать две кавычки подряд.

Итак, мы написали на встроенном языке текст запроса, содержащего параметризированное условие (см. листинг 2.12). Но если текст запроса содержит параметры, то перед выполнением запроса значения параметров должны быть переданы

в запрос.

Это выполняется с помощью метода *УстановитьПараметр()* объекта *Запрос* (листинг 2.14).

Листинг 2.14. Установка параметров запроса во встроенном языке

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        Клиенты.Наименование КАК Наименование,
  |        Клиенты.Адрес,
  |        Клиенты.Телефон
  |ИЗ
  |        Справочник.Клиенты КАК Клиенты
  |ГДЕ
  |        Наименование ПОДОБНО ""%"" + &ЧастьНаименования + ""%""";

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

Первым параметром в метод *УстановитьПараметр()* передается строка с именем параметра запроса (*"ЧастьНаименования"*), а вторым параметром передается значение реквизита формы обработки *ЧастьНаименования*. Этот фрагмент можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Рассмотрим еще один подобный пример. Напишем на встроенном языке запрос для отбора из справочника тех клиентов, телефоны которых не соответствуют заданному шаблону. Условие отбора этого запроса содержит параметр *ШаблонТелефона*, значение которого перед выполнением запроса передается в запрос (листинг 2.15).

Листинг 2.15. Установка параметров запроса во встроенном языке

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        Клиенты.Наименование КАК Наименование,
  |        Клиенты.Адрес,
  |        Клиенты.Телефон
  |ИЗ
  |        Справочник.Клиенты КАК Клиенты
  |ГДЕ
  |        НЕ Клиенты.Телефон ПОДОБНО &ШаблонТелефона";

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

Первым параметром в метод *УстановитьПараметр()* передается строка с именем параметра запроса (*"ШаблонТелефона"*), а вторым параметром передается значение реквизита формы обработки *ШаблонТелефона*. Этот фрагмент можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Получение выборки из результата запроса

После присвоения текста и установки параметров запрос запускается на выполнение с помощью метода *Выполнить()* объекта *Запрос*. Именно в этот момент и происходит чтение данных из базы данных. Прочитанные данные возвращаются в виде объекта *РезультатЗапроса*, содержащего выбранные данные из базы данных (листинг 2.16).

Листинг 2.16. Выполнение запроса

```
РезультатЗапроса = Запрос.Выполнить();
```

Таким образом, происходит третий этап выполнения запроса, показанный на схеме (см. рис. 2.31).

Результат выполнения запроса может не содержать строк. Проверку этого следует выполнять с помощью метода *Пустой()* объекта *РезультатЗапроса* (листинг 2.17).

Листинг 2.17. Проверка результата запроса на наличие записей

```
…
РезультатЗапроса = Запрос.Выполнить();

Сообщение = Новый СообщениеПользователю;
Если РезультатЗапроса.Пустой() Тогда
  Сообщение.Текст = "Записей по условию не найдено";
  Сообщение.Сообщить();
  Возврат;

КонецЕсли;

Выборка = РезультатЗапроса.Выбрать();
…
```

Проверку на пустой результат запроса следует выполнять до получения выборки из результата запроса методом *Выбрать()*, поскольку на получение выборки будет затрачиваться дополнительное время.

Далее, чтобы обработать данные, содержащиеся в объекте *РезультатЗапроса*, из него получается выборка с помощью метода *Выбрать()*, который возвращает новый объект *ВыборкаИзРезультатаЗапроса*, то есть коллекцию данных, предназначенную для обхода ее элементов (листинг 2.18).

Листинг 2.18. Получение выборки из результата запроса

```
Выборка = РезультатЗапроса.Выбрать();
```

Таким образом, происходит четвертый этап выполнения запроса, показанный на схеме (см. рис. 2.31).

Обход выборки из результата запроса

Теперь рассмотрим подробно пятый этап выполнения запроса, показанный на схеме (см. рис. 2.31). На этом этапе и происходит то, ради чего, собственно, создавался и выполнялся запрос – программная обработка результатов запроса для последующего представления их пользователю, например, вывода результатов запроса в табличный документ.

Для обхода выборки из результата запроса нужно организовать цикл, в котором перебираются элементы коллекции данных, содержащихся в объекте *ВыборкаИзРезультатаЗапроса*. Для этого используется метод выборки *Следующий()*, который позволяет перейти к следующей записи результата запроса в соответствии с порядком обхода выборки (об этом будет рассказано ниже).

Этот метод вызывается в цикле *Пока Выборка.Следующий() … Цикл* до тех пор, пока не будет получено значение *Ложь*. При первом проходе цикла метод *Следующий()* позиционирует выборку на первую запись.

Следует иметь в виду, что если выборка из результата запроса получена, но ее метод *Следующий()* еще ни разу не вызывался (то есть выборка еще не спозиционирована), то значения полей выборки будут не определены, например, их нельзя будет посмотреть в отладчике.

При обходе выборки в теле цикла производятся необходимые действия над данными, полученными с помощью запроса (листинг 2.19).

Листинг 2.19. Обход выборки

```
Пока Выборка.Следующий() Цикл
  Сообщение = Новый СообщениеПользователю;
  Сообщение.Текст = Выборка.Наименование;
  Сообщение.Сообщить();

КонецЦикла;
```

Результат запроса может содержать записи, не имеющие иерархии (например, список документов) или, наоборот, записи, обладающие иерархией. Например, список иерархического справочника, содержащий группы, и подчиненные им элементы. Устройство иерархического справочника мы подробно рассматривали в разделе «Как получить записи иерархической таблицы и расположить их в порядке иерархии».

Второй случай получения иерархических записей, когда в запросе рассчитываются итоги. Дело в том, что при расчете итогов образуется некая иерархическая структура, поскольку записи с одинаковым значением поля (полей), для которых рассчитываются итоги, в результате запроса собираются (группируются) вместе и достраиваются итоговой строкой. Эта итоговая строка является по отношению к ним родительской строкой, стоящей на более высоком уровне иерархии.

Поэтому существует несколько разных способов обхода выборки – в линейном порядке, в иерархическом порядке или по группировкам. Тип обхода выборки задается значением системного перечисления *ОбходРезультатаЗапроса* и передается в качестве параметра в метод *Выбрать()* объекта *РезультатЗапроса*.

При *линейном обходе* выборки будут последовательно получаться все записи из результата запроса, при *иерархическом обходе* получаются только записи результата запроса, находящиеся на одном уровне иерархии, при *обходе по группировкам* будут получены только родительские записи, являющие групповыми итогами. Различные способы обхода выборки и их отличие друг от друга представлены на рис. 2.33.

Рассмотрим эти варианты обхода выборки на примере запроса, в котором выводятся наименования товаров и количество их поступления из состава приходных накладных и рассчитываются иерархические итоги по полю *Товар* (листинг 2.20).

Листинг 2.20. Текст запроса

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |         ПриходнаяНакладнаяСостав.Товар КАК Товар,
  |         ПриходнаяНакладнаяСостав.Количество КАК Количество
  |ИЗ
  |         Документ.ПриходнаяНакладная.Состав КАК ПриходнаяНакладнаяСостав
  |
  |УПОРЯДОЧИТЬ ПО
  |         Товар
  |ИТОГИ
  |         СУММА(Количество)
  |ПО
  |         Товар ИЕРАРХИЯ";
```

Результат запроса представлен в таблице 2.1, в которой каждую запись мы снабдили порядковым номером для ее идентификации в результате запроса. Итоговые записи в таблице выделены курсивом, итоговые записи для иерархических уровней справочника выделены жирным шрифтом.

Таблица 2.1. Результат запроса

№	Товар	Количество
1	**Обувь**	**60**
2	*Туфли*	*15*
3	Туфли	5
4	Туфли	10
5	*Сапоги*	*10*
6	Сапоги	5

7	Сапоги	5
8	*Кроссовки*	*35*
9	Кроссовки	15
10	Кроссовки	20
11	**Продукты**	**200**
12	*Масло*	*50*
13	Масло	20
14	Масло	30
15	*Молоко*	*80*
16	Молоко	30
17	Молоко	50
18	*Сметана*	*70*
19	Сметана	50
20	Сметана	20

На следующей схеме для сравнения представлены все три вида обхода выборки из этого результата запроса, в каждом из них цветом выделены те записи, которые попадут в выборку (рис. 2.33).

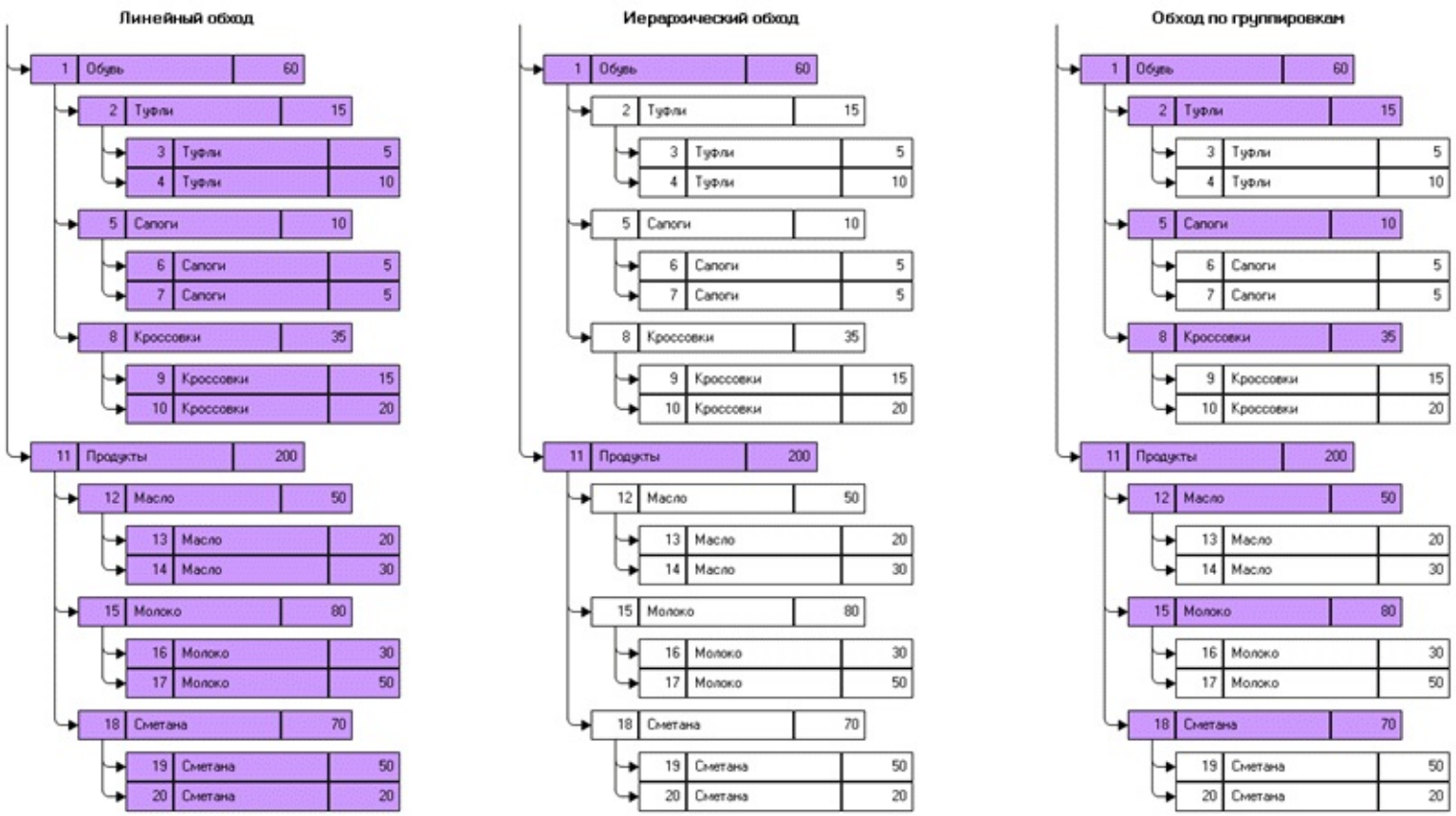


Рис. 2.33. Виды обхода выборки из результата запроса

Рассмотрим каждый вид обхода выборки более подробно.

Линейный (прямой) порядок обхода

Линейный обход результата запроса – самый простой способ обхода выборки. При линейном обходе выборка будет выдавать записи в той последовательности, в которой они располагаются в результате запроса. В нашем примере (см. рис. 2.33) это будут записи с номерами 1, 2, 3, 4, 5 и так далее до записи с номером 20.

Для получения линейной выборки из результата запроса необходимо вызвать метод *Выбрать()* объекта *РезультатЗапроса* без параметров либо с параметром *ОбходРезультатаЗапроса.Прямой* (листинг 2.21).

Листинг 2.21. Прямой порядок обхода выборки

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ПриходнаяНакладнаяСостав.Товар КАК Товар,
  |        ПриходнаяНакладнаяСостав.Количество КАК Количество
  |ИЗ
  |        Документ.ПриходнаяНакладная.Состав КАК ПриходнаяНакладнаяСостав
  |
  |УПОРЯДОЧИТЬ ПО
  |        Товар
  |ИТОГИ
  |        СУММА(Количество)
  |ПО
  |        Товар ИЕРАРХИЯ";

РезультатЗапроса = Запрос.Выполнить();

СпособВыборки = ОбходРезультатаЗапроса.Прямой;
ВыборкаЗапроса = РезультатЗапроса.Выбрать(СпособВыборки);

Сообщение = Новый СообщениеПользователю;
Пока ВыборкаЗапроса.Следующий() Цикл
  Сообщение.Текст = "Товар: " + ВыборкаЗапроса.Товар.Наименование +
    " Итого: " + ВыборкаЗапроса.Количество +
    " Тип записи: " + ВыборкаЗапроса.ТипЗаписи() +
    " Уровень: " + ВыборкаЗапроса.Уровень() +
    " Группировка: " + ВыборкаЗапроса.Группировка();
  Сообщение.Сообщить();

КонецЦикла;
```

Пример этой процедуры находится в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

В этом примере в окно сообщений формы обработки последовательно выдаются данные о поступлении товаров с иерархическими итогами по каждому товару.

Иерархический порядок обхода

При иерархическом обходе результата запроса обходятся только записи, находящиеся на одном уровне иерархии. Для получения иерархической выборки из результата запроса необходимо вызвать метод *Выбрать()* объекта *РезультатЗапроса* с параметром *ОбходРезультатаЗапроса.ПоГруппировкамСИерархией*.

Однако при первом вызове метода *Выбрать()* с этим типом обхода будут получены только записи, находящиеся на самом верхнем уровне иерархии. То есть в нашем примере (см. рис. 2.33) это будут записи с номерами 1 и 11. Только две эти записи попадут в первый проход иерархической выборки. А как получить остальные записи результата запроса?

Для обхода записей всех уровней иерархии в цикле обхода первой выборки нужно для каждой записи выборки методом *Выбрать()* объекта *ВыборкаИзРезультатаЗапроса* получать вложенные иерархические выборки, которые будут содержать подчиненные записи текущей записи выборки.

В нашем примере в момент, когда первая выборка будет позиционирована на запись с номером 1, мы запросим у нее подчиненную иерархическую выборку. Таким образом, мы получим выборку, которая нам вернет записи с номерами 2, 5, 8. А когда выборка верхнего уровня будет позиционирована на запись с номером 11, полученная у нее иерархическая выборка вернет записи с номерами 12, 15, 18. Так реализуется иерархический обход результатов запроса.

Рассмотрим, как реализуется данная методика на примере обхода результата запроса, в котором получаются записи из состава приходных накладных и рассчитываются иерархические итоги по полю *Товар*. В процедуре выполнения запроса из результата запроса получается выборка с типом обхода *ПоГруппировкамСИерархией*, содержащая записи, находящиеся на самом верхнем уровне иерархии (листинг 2.22).

Листинг 2.22. Иерархический обход результата запроса

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос()

  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |        ПриходнаяНакладнаяСостав.Товар КАК Товар,
    |        ПриходнаяНакладнаяСостав.Количество КАК Количество
    |ИЗ
    |        Документ.ПриходнаяНакладная.Состав КАК ПриходнаяНакладнаяСостав
    |
    |УПОРЯДОЧИТЬ ПО
    |        Товар
    |ИТОГИ
    |        СУММА(Количество)
    |ПО
    |        Товар ИЕРАРХИЯ";

  РезультатЗапроса = Запрос.Выполнить();
```

```
  СпособВыборки = ОбходРезультатаЗапроса.ПоГруппировкамСИерархией;
  ВыборкаЗапроса = РезультатЗапроса.Выбрать(СпособВыборки);

  ВыдатьВсеВложения(ВыборкаЗапроса);

КонецПроцедуры
```

Затем вызывается процедура *ВыдатьВсеВложения()*, в которую передается полученная выборка. В этой процедуре иерархическая выборка обходится в цикле, и в окно сообщений выводятся данные этой выборки. После этого в теле цикла для текущей записи выборки вызывается метод *Выбрать()*, и из нее получается подчиненная выборка записей, находящихся на следующем уровне иерархии. При этом в случае получения в запросе иерархических итогов нужно анализировать тип записи выборки. Если тип записи – *Итог по иерархии*, то вторым параметром в метод *Выбрать()* нужно передавать имя группировки, для которой были рассчитаны иерархические итоги.

Затем рекурсивно вызывается процедура *ВыдатьВсеВложения()*, в которую передается полученная иерархическая выборка, содержащая подчиненные записи родительской записи выборки (листинг 2.23).

Листинг 2.23. Обход иерархической выборки

```
&НаСервереБезКонтекста
Процедура ВыдатьВсеВложения(ИерархическаяВыборка)

  Сообщение = Новый СообщениеПользователю;
  Пока ИерархическаяВыборка.Следующий() Цикл

    Сообщение.Текст = "Товар: " + ИерархическаяВыборка.Товар.Наименование +
      " Количество: " + ИерархическаяВыборка.Количество +
      " Тип записи: " + ИерархическаяВыборка.ТипЗаписи() +
      " Уровень: " + ИерархическаяВыборка.Уровень() +
      " Группировка: " + ИерархическаяВыборка.Группировка();
    Сообщение.Сообщить();

    // Продолжим выборку подчиненных записей
    СпособВыборки = ОбходРезультатаЗапроса.ПоГруппировкамСИерархией;

    Если ИерархическаяВыборка.ТипЗаписи() = ТипЗаписиЗапроса.ИтогПоИерархии Тогда
    ДочерняяВыборка = ИерархическаяВыборка.Выбрать(СпособВыборки,
    ИерархическаяВыборка.Группировка());

    Иначе
    ДочерняяВыборка = ИерархическаяВыборка.Выбрать(СпособВыборки);

    КонецЕсли;

    ВыдатьВсеВложения(ДочерняяВыборка);
  КонецЦикла;

КонецПроцедуры
```

Пример этой процедуры находится в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Результат иерархического обхода выборки представлен на рис. 2.34.

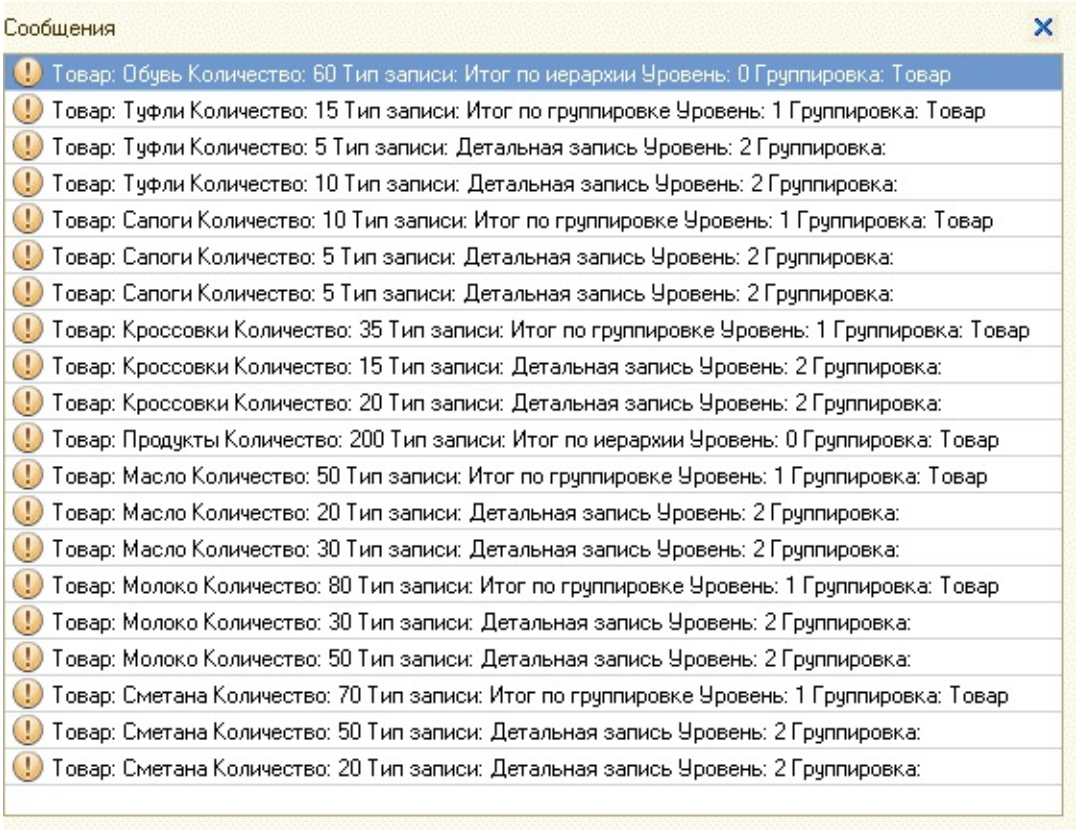


Рис. 2.34. Результат обхода иерархической выборки

Мы видим, что помимо суммарного значения количества поступления каждого товара и его наименования в окно сообщений выводятся такие характеристики текущей записи выборки, как: *Уровень*, *Тип записи* и *Группировка*. Эти характеристики получаются с помощью одноименных методов выборки (объекта *ВыборкаИзРезультатаЗапроса*):

- *Уровень()* – определяет уровень текущей записи в иерархии и группировках. Уровень считается от начальной выборки из результата запроса. Уровень начальной выборки равен нулю.
- *ТипЗаписи()* – определяет принадлежность записи к одному из следующих типов, перечисленных в системном перечислении *ТипЗаписиЗапроса*:
- общий итог,
- итог по иерархии,
- итог по группировке,
- детальная запись.
- *Группировка()* – определяет имя поля, по которому были рассчитаны итоги. Для детальных записей возвращается пустая строка.

На схеме для нашего примера это будет выглядеть следующим образом (рис. 2.35).

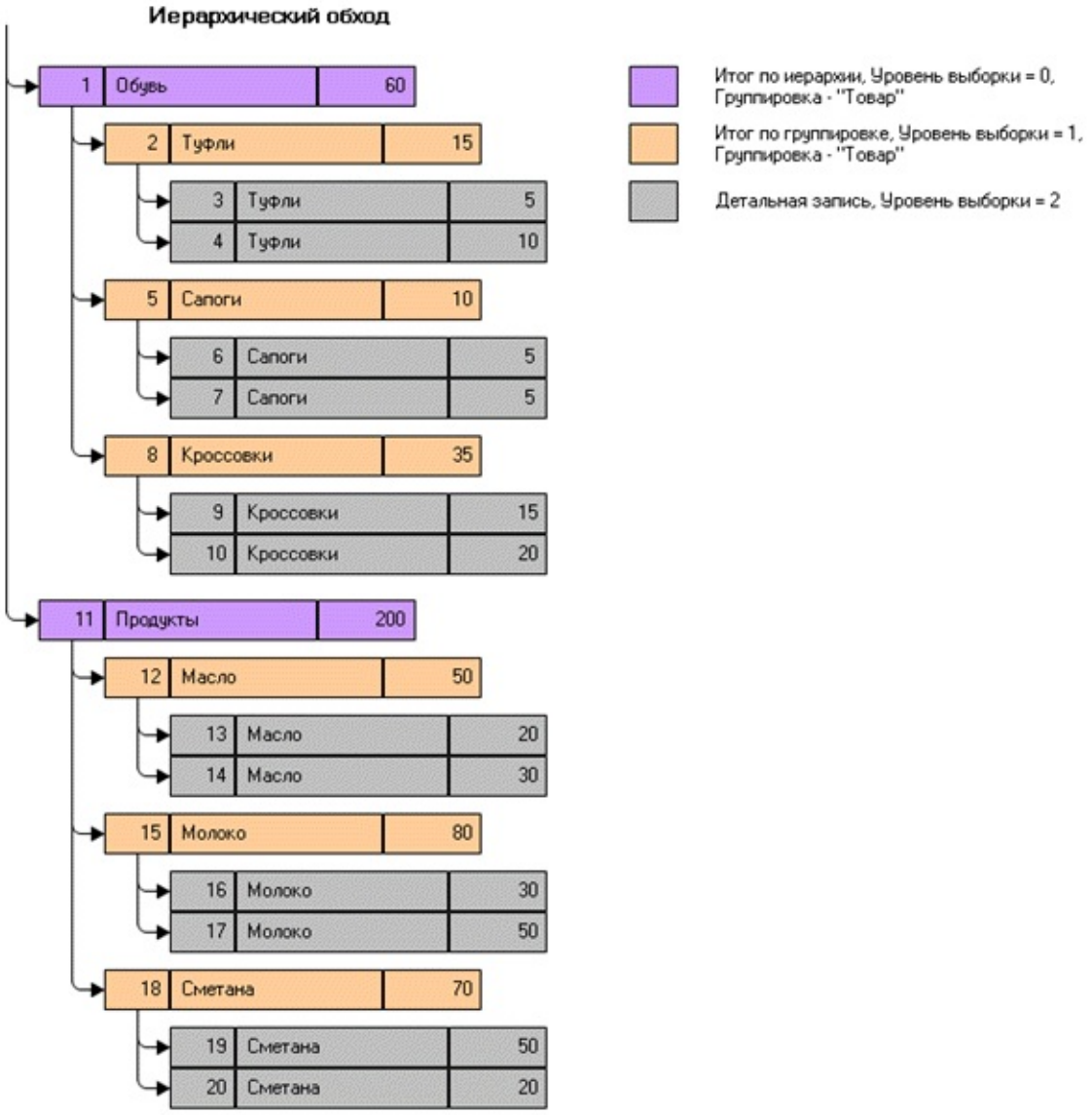


Рис. 2.35. Иерархический обход результата запроса

Мы видим, что в результате запроса рассчитаны итоги по иерархии товаров. На самом верхнем уровне иерархии находятся записи с номерами 1, 11 (на схеме выделены сиреневым цветом) – *Обувь* и *Продукты*. Эти записи будут получены в результате первого прохода иерархической выборки. Уровень выборки для этих записей равен 0, тип записи – *Итог по иерархии*, а группировка – поле, по которому рассчитаны итоги, – *Товар*. На втором проходе иерархической выборки будут получены записи с номерами 2, 5, 8, 12, 15, 18 (на схеме выделены бежевым цветом), находящиеся на следующем уровне иерархии – *Туфли*, *Сапоги*, *Кроссовки* и *Масло*, *Молоко*, *Сметана*. Уровень выборки для этих записей равен 1, тип записи – *Итог по группировке*, а группировка – *Товар*. И на третьем проходе иерархической выборки будут получены детальные записи с номерами 3, 4, 6, 7, 9, 10, 13, 14, 16, 17, 19, 20 (на схеме выделены серым цветом). Уровень выборки для этих записей равен 2, тип записи – *Детальная запись*, а группировка – пустая строка. Термин «Группировка» здесь используется для указания поля, по которому были рассчитаны итоги и по значению которого записи в результате запроса собираются вместе.

Обход по группировкам

Обход результата запроса по группировкам используется в случае получения в

результате запроса итоговых данных (в запросе присутствует слово *ИТОГИ ПО*). В нашем примере (см. рис. 2.33) это будут записи с номерами 1, 2, 5, 8, 11, 12, 15, 18.

Для получения выборки по группам из результата запроса необходимо вызвать метод *Выбрать()* объекта *РезультатЗапроса* с параметром *ОбходРезультатаЗапроса.ПоГруппировкам*.

При вызове метода *Выбрать()* с этим типом обхода будут получены только родительские записи, являющие групповыми итогами. Для обхода детальных записей в цикле обхода групповой выборки нужно для каждой записи выборки методом *Выбрать()* объекта *ВыборкаИзРезультатаЗапроса* линейным способом получать дочерние выборки, которые будут содержать подчиненные записи текущей записи выборки.

Для примера рассмотрим тот же запрос, что и в двух предыдущих случаях (листинг 2.24).

Листинг 2.24. Обход результата запроса по группам

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос()

  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |        ПриходнаяНакладнаяСостав.Товар КАК Товар,
    |        ПриходнаяНакладнаяСостав.Количество КАК Количество
    |ИЗ
    |        Документ.ПриходнаяНакладная.Состав КАК ПриходнаяНакладнаяСостав
    |
    |УПОРЯДОЧИТЬ ПО
    |        Товар
    |ИТОГИ
    |        СУММА(Количество)
    |ПО
    |        Товар ИЕРАРХИЯ";

  РезультатЗапроса = Запрос.Выполнить();

  СпособВыборки = ОбходРезультатаЗапроса.ПоГруппировкам;

  // групповые итоги
  ВыборкаЗапроса = РезультатЗапроса.Выбрать(СпособВыборки);

  Сообщение = Новый СообщениеПользователю;
  Пока ВыборкаЗапроса.Следующий() Цикл
    Сообщение.Текст = "Товар: " + ВыборкаЗапроса.Товар.Наименование +
      " Итого: " + ВыборкаЗапроса.Количество +
      " Тип записи: " + ВыборкаЗапроса.ТипЗаписи() +
      " Уровень: " + ВыборкаЗапроса.Уровень() +
      " Группировка: " + ВыборкаЗапроса.Группировка();
    Сообщение.Сообщить();

    // детальные записи
    ВыдатьДочерниеЗаписи(ВыборкаЗапроса.Выбрать());
  КонецЦикла;
```

```
КонецПроцедуры
```

В процедуре выполнения запроса для получения групповых итогов из результата запроса получается выборка с типом обхода *ПоГруппировкам*. Эта выборка обходится в цикле, и в окно сообщений выводятся данные этой выборки.

После этого в теле цикла для текущей записи второй выборки вызывается метод *Выбрать()* с прямым порядком обхода (т. к. нужно обойти детальные записи). Затем вызывается процедура *ВыдатьДочерниеЗаписи()*, в которую передается полученная выборка, содержащая подчиненные записи родительской записи выборки.

В процедуре *ВыдатьДочерниеЗаписи()* обходятся детальные записи для каждой группировки (итоговой строки), и в окно сообщений выводится суммарное количество поступлений каждого товара и его наименование, а также такие характеристики текущей записи выборки, как: *Уровень*, *Тип записи* и *Группировка* (листинг 2.25). Эти характеристики были рассмотрены в предыдущем разделе.

Листинг 2.25. Обход дочерней выборки

```
&НаСервереБезКонтекста
Процедура ВыдатьДочерниеЗаписи(ДочерняяВыборка)

  Сообщение = Новый СообщениеПользователю;
  Пока ДочерняяВыборка.Следующий() Цикл
    Сообщение.Текст = "Товар: " + ДочерняяВыборка.Товар.Наименование +
      " Количество: " + ДочерняяВыборка.Количество +
      " Тип записи: " + ДочерняяВыборка.ТипЗаписи() +
      " Уровень: " + ДочерняяВыборка.Уровень() +
      " Группировка: " + ДочерняяВыборка.Группировка();
    Сообщение.Сообщить();

  КонецЦикла;

КонецПроцедуры
```

Пример этой процедуры находится в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Результат обхода выборки по группировкам представлен на рис. 2.36.

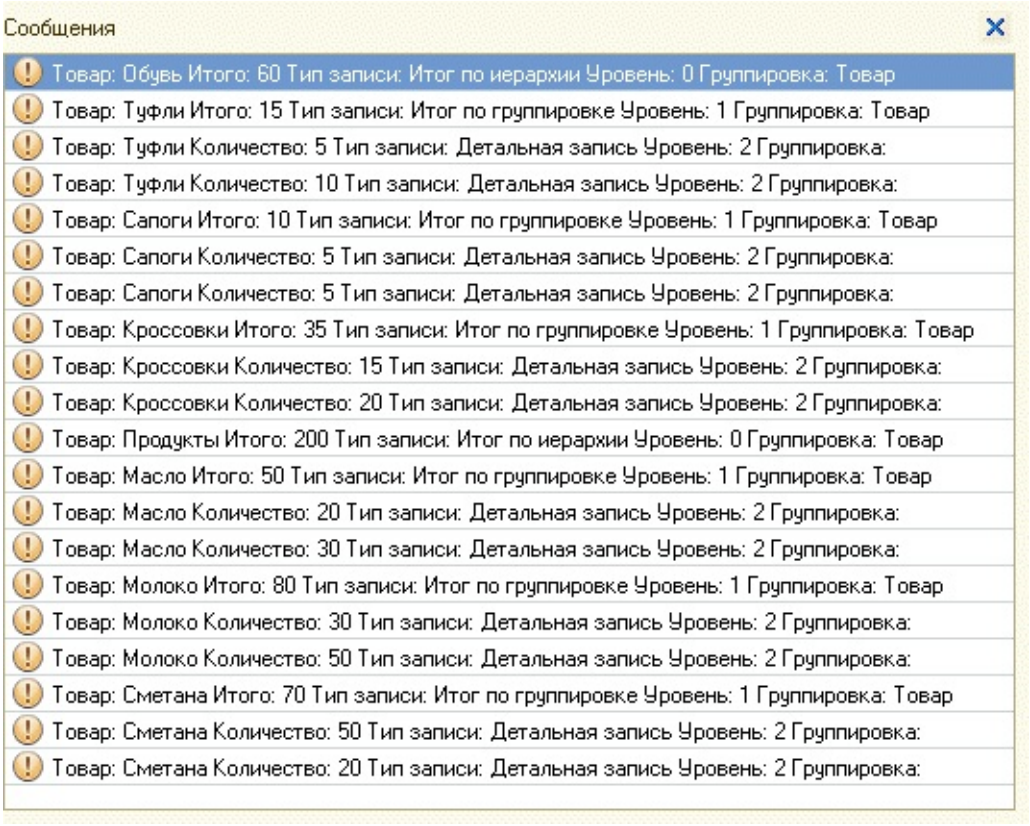


Рис. 2.36. Результат обхода выборки по группировкам

Если в запросе получается общий итог, то эта запись занимает место в корне иерархии с нулевым уровнем и типом записи *Общий Итог*, а все остальные записи как бы вложены в нее, и их уровень соответственно возрастает на единицу.

Обход выборки результата запроса, содержащего данные табличной части

Часто объекты базы данных, например документы, имеют основную таблицу, в которой содержатся реквизиты документа, и подчиненную таблицу, в которой содержатся данные табличной части документа. Эти данные можно получить, обращаясь отдельно к таблице с данными табличной части (раздел "Как получить данные из табличной части некоторого документа"), а также их можно получить из основной таблицы в качестве вложенного результата запроса. Второй вариант мы и рассмотрим ниже.

Как уже объяснялось ранее в разделе "Как получить данные из табличной части документа в качестве вложенной таблицы", при выполнении запроса к основной таблице поле результата запроса, содержащее данные из табличной части (например, *ЗаказТовара.Состав*), будет иметь тип *РезультатЗапроса*, то есть содержать вложенный результат запроса, сформированный на основе табличной части.

Чтобы обработать данные вложенной таблицы, нужно в цикле обхода основной выборки из результата запроса для каждой записи выборки методом *Выбрать()* объекта *ВыборкаИзРезультатаЗапроса* получить вложенную выборку, которая будет содержать данные табличной части для текущей записи основной выборки.

Ниже приводится пример процедуры встроенного языка, в которой обрабатываются как реквизиты документа, так и данные его табличной части и выводятся в окно сообщений (листинг 2.26).

Листинг 2.26. Пример обработки результата запроса, выбирающего данные из документа с табличной частью

```
&НаСервере
Процедура ВыполнитьЗапрос()

  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |         ЗаказТовара.Номер,
    |         ПРЕДСТАВЛЕНИЕ(ЗаказТовара.Клиент) КАК Клиент,
    |         ЗаказТовара.СуммаЗаказа,
    |         ЗаказТовара.Состав.(
    |                 ПРЕДСТАВЛЕНИЕ(Товар) КАК Товар,
    |                 Количество,
    |                 Сумма
    |         )
    |ИЗ
    |         Документ.ЗаказТовара КАК ЗаказТовара
    |ГДЕ
    |         ЗаказТовара.Ссылка = &Документ";

  Запрос.УстановитьПараметр("Документ", Документ);

  РезультатЗапроса = Запрос.Выполнить();
  // основная выборка
  ВыборкаЗапроса = РезультатЗапроса.Выбрать();

  Сообщение = Новый СообщениеПользователю;
  Пока ВыборкаЗапроса.Следующий() Цикл
```

```
    // выборка табличной части
    СоставВыборка = ВыборкаЗапроса.Состав.Выбрать();

    Пока СоставВыборка.Следующий() Цикл
      Сообщение.Текст = ВыборкаЗапроса.Номер + ", " + ВыборкаЗапроса.Клиент + ": " +
      СоставВыборка.Товар;
      Сообщение.Сообщить();

    КонецЦикла;

  КонецЦикла;

КонецПроцедуры
```

При обработке результатов данного запроса получается и обходится выборка основного запроса (*ВыборкаЗапроса*). А внутри цикла обхода этой выборки получается и обходится выборка результатов вложенного запроса (*СоставВыборка*), содержащая данные табличной части. Обе выборки имеют линейный тип обхода, так как в запросе нет итогов и иерархии.

Внутри цикла обхода вложенной выборки реквизиты документа и данные его табличной части выводятся в окно сообщений в виде одной строки. Чтобы сложить строки, в запросе получаются текстовые представления от ссылочных полей (*Клиент*, *Товар*) с помощью функции *ПРЕДСТАВЛЕНИЕ()*.

подробнее

О функции *Представление()* рассказано в разделе «Как получить текстовое представление ссылочного поля».

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Обработка результатов запроса с помощью конструктора запроса

В предыдущих примерах обхода выборки результаты запроса выводились в окно сообщений, но это сделано только для упрощения примера. На самом деле результаты выполнения запросов выводятся обычно в табличный документ, таблицу значений и т. п.

Рассмотрим пример обхода выборки и вывода результатов запроса в табличный документ и в диаграмму при помощи конструктора запроса. В данном случае нам понадобится *Конструктор запроса с обработкой результата*, который помогает не только визуально сконструировать запрос, но и создать готовый фрагмент кода для получения данных с помощью запроса и обработки его результатов.

Обход выборки

Откроем модуль формы и вызовем из контекстного меню пункт *Конструктор запроса с обработкой результата*. Подтвердим, что мы хотим создать новый запрос. После этого откроется конструктор запроса с обработкой результата (рис. 2.37).

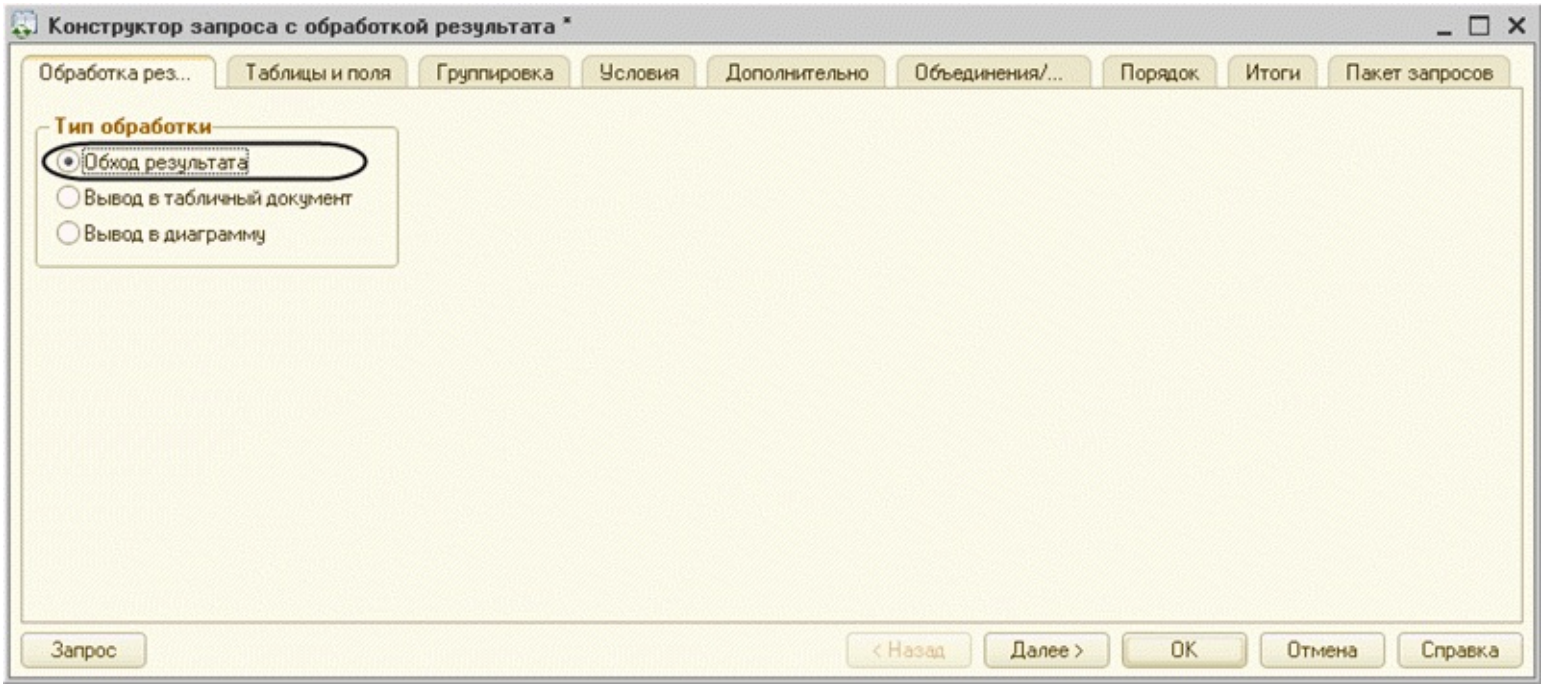


Рис. 2.37. Окно конструктора запроса с обработкой результата

Как мы видим, он очень похож на обычный конструктор запроса по составу и назначению своих закладок, за исключением самой первой закладки *Обработка результатов*, на которой и определяется, как будет обработан результат запроса. По умолчанию включена опция *Обход результата*. Оставим пока эту опцию без изменений.

В этом случае конструктор не только создаст текст запроса, но и напишет за разработчика фрагмент кода, в котором запрос с нужным текстом создается, выполняется, из результата запроса получается выборка, а также конструктор создаст цикл (или циклы) для обхода этой выборки.

Далее, как и в обычном конструкторе, определим исходные данные для запроса. На закладке *Таблицы и поля* перенесем табличную часть *Состав* документа *ЗаказТовара* в список источников запроса и выберем из этой таблицы поля: *Товар*, *Количество* и *Сумма*. Для того чтобы выбрать поля основной таблицы документа *Клиент* и *Дата*, раскроем поле *Ссылка* табличной части *Состав*, выберем эти поля и получим обращение через точку от ссылки – *ЗаказТовараСостав.Ссылка.Клиент* и *ЗаказТовараСостав.Ссылка.Дата* (рис. 2.38).

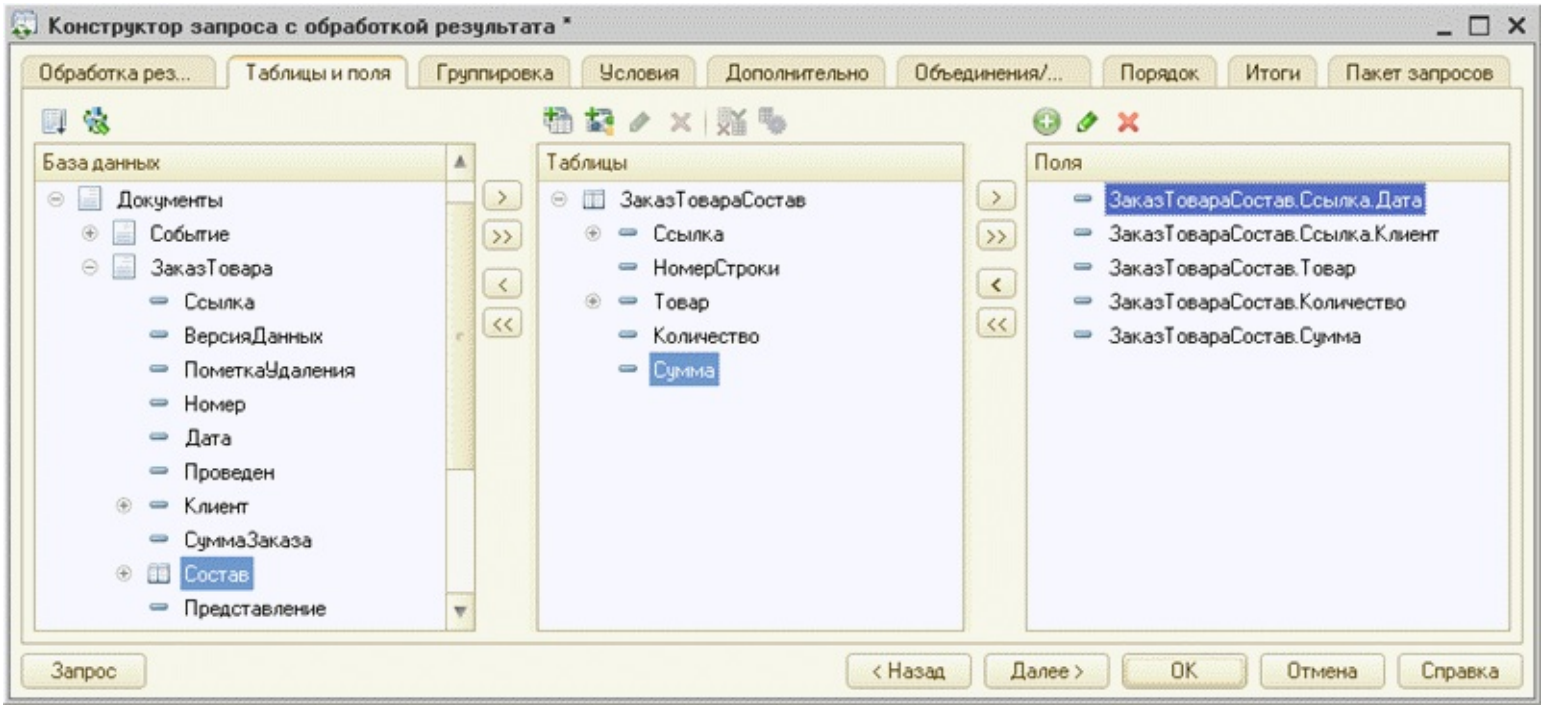


Рис. 2.38. Окно конструктора запроса с обработкой результата

Затем перейдем на закладку *Итоги* и укажем, какие итоги нужно рассчитать для результата запроса. В окно *Группировочное поле* перенесем поля выборки запроса *Клиент* и *Товар*, а в окно *Итоговое поле* перенесем поля *Количество* и *Сумма* (рис. 2.39).

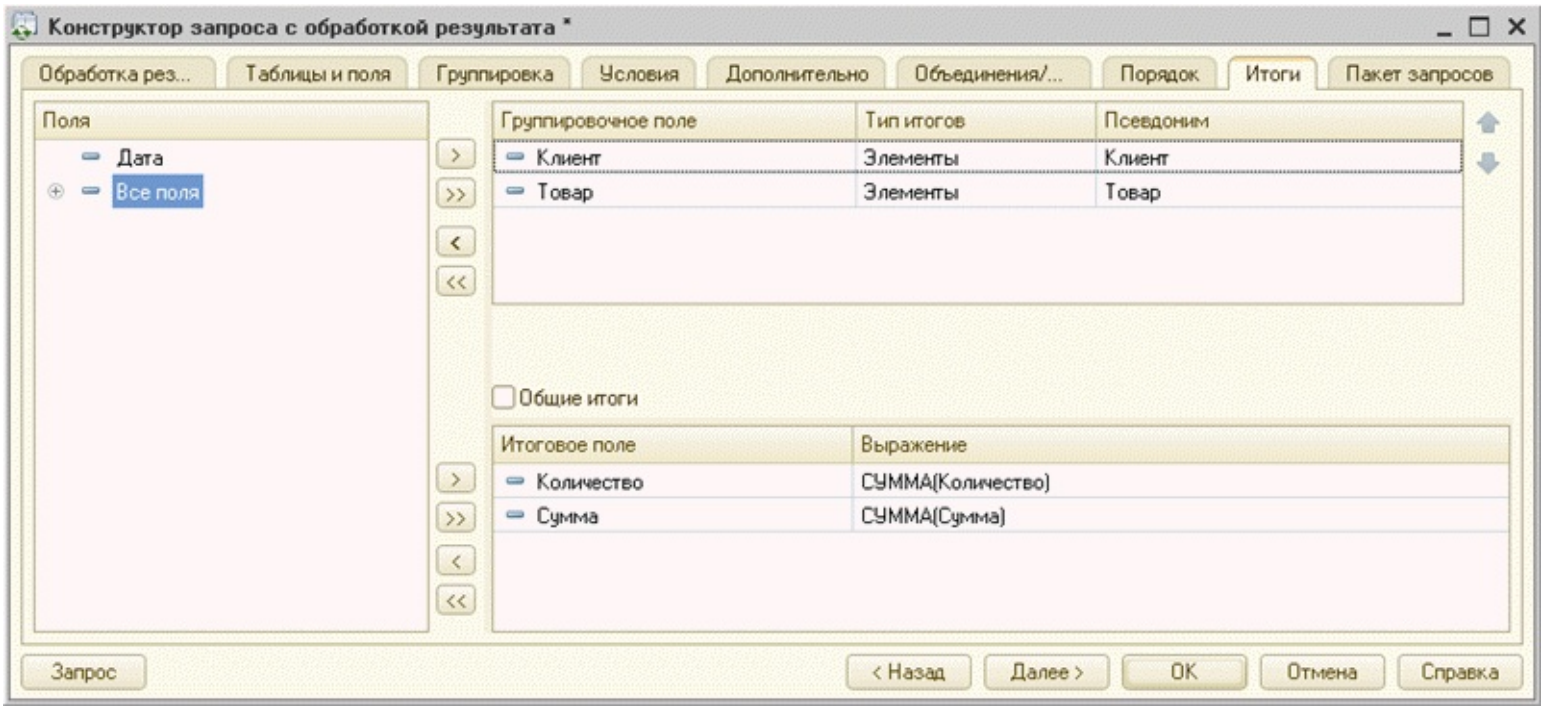


Рис. 2.39. Окно конструктора запроса с обработкой результата

Тем самым мы указали, что для каждого клиента по каждому товару из документа *ЗаказТовара* нужно подсчитать суммарное количество полей *Количество* и *Сумма*.

Нажмем *ОК*. Конструктор запроса с обработкой результата сформирует следующий фрагмент кода (листинг 2.27).

Листинг 2.27. Фрагмент процедуры для обхода выборки из результата запроса, созданный с помощью конструктора запроса с обработкой результата

```
//{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
// Данный фрагмент построен конструктором.
// При повторном использовании конструктора внесенные вручную изменения будут утеряны!!!

Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ЗаказТовараСостав.Ссылка.Дата,
  |        ЗаказТовараСостав.Ссылка.Клиент КАК Клиент,
  |        ЗаказТовараСостав.Товар КАК Товар,
  |        ЗаказТовараСостав.Количество КАК Количество,
  |        ЗаказТовараСостав.Сумма КАК Сумма
  |ИЗ
  |        Документ.ЗаказТовара.Состав КАК ЗаказТовараСостав
  |ИТОГИ
  |        СУММА(Количество),
  |        СУММА(Сумма)
  |ПО
  |        Клиент,
  |        Товар";

РезультатЗапроса = Запрос.Выполнить();

ВыборкаКлиент = РезультатЗапроса.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

Пока ВыборкаКлиент.Следующий() Цикл
  // Вставить обработку выборки ВыборкаКлиент

  ВыборкаТовар = ВыборкаКлиент.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

  Пока ВыборкаТовар.Следующий() Цикл
    // Вставить обработку выборки ВыборкаТовар

    ВыборкаДетальныеЗаписи = ВыборкаТовар.Выбрать();

    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
      // Вставить обработку выборки ВыборкаДетальныеЗаписи

    КонецЦикла;

  КонецЦикла;

КонецЦикла;

//}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
```

Если мы сравним этот фрагмент с процедурой выполнения запроса (см. листинги 2.24, 2.25), в котором обходилась выборка с типом обхода *ПоГруппировкам*, то мы увидим, что конструктор запроса с обработкой результата сделал практически всю работу, за исключением самой программной обработки выборок. Эти фрагменты в тексте листинга 2.27 выделены жирным шрифтом, и их должен заполнить сам разработчик.

Кроме того, мы видим, что конструктор проанализировал текст запроса, в котором

рассчитываются итоги для полей *Клиент* и *Товар*. Соответственно, чтобы обойти групповые итоги и детальные записи результата запроса, конструктор создал три вложенных друг в друга цикла для обхода выборок – выборки по группе записей *Клиент*, вложенной в нее выборки по группе *Товар* и вложенной в нее выборки детальных записей. Для групповых записей получается выборка с типом обхода *ПоГруппировкам*, а для детальных записей – с прямым типом обхода.

Вывод в табличный документ

Теперь с помощью конструктора выведем результат этого же запроса в табличный документ. Поскольку табличный документ заполняется данными на сервере, а выводится пользователю на клиенте, сначала создадим реквизит формы *Результат* типа *ТабличныйДокумент*, который и будет содержать данные результата запроса, и перетащим его в дерево элементов формы обработки. После этого создадим в модуле формы небольшую заготовку (листинг 2.28).

Листинг 2.28. Процедуры для вывода результата запроса в табличный документ

```
&НаКлиенте
Процедура ВыводВТабличныйДокумент(Команда)

  ЗаполнитьТД(Результат);

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ЗаполнитьТД(Табдок)

  // Фрагмент для вывода в табличный документ

КонецПроцедуры
```

В клиентской процедуре *ВыводВТабличныйДокумент()* вызывается серверная внеконтекстная процедура *ЗаполнитьТД()*, в которую передается реквизит формы *Результат*. В этой процедуре табличный документ заполняется данными, и при возвращении на клиента отображается пользователю в поле формы вида *Поле табличного документа*.

Теперь создадим собственно фрагмент кода для вывода результата запроса в табличный документ. Для этого установим курсор внутрь процедуры *ЗаполнитьТД()* и вызовем из контекстного меню пункт *Конструктор запроса с обработкой результата*. Подтвердим, что мы хотим создать новый запрос.

На закладке конструктора *Обработка результатов* в группе *Тип обработки* выберем опцию *Вывод в табличный документ*, все остальные опции оставим без изменения (рис. 2.40).

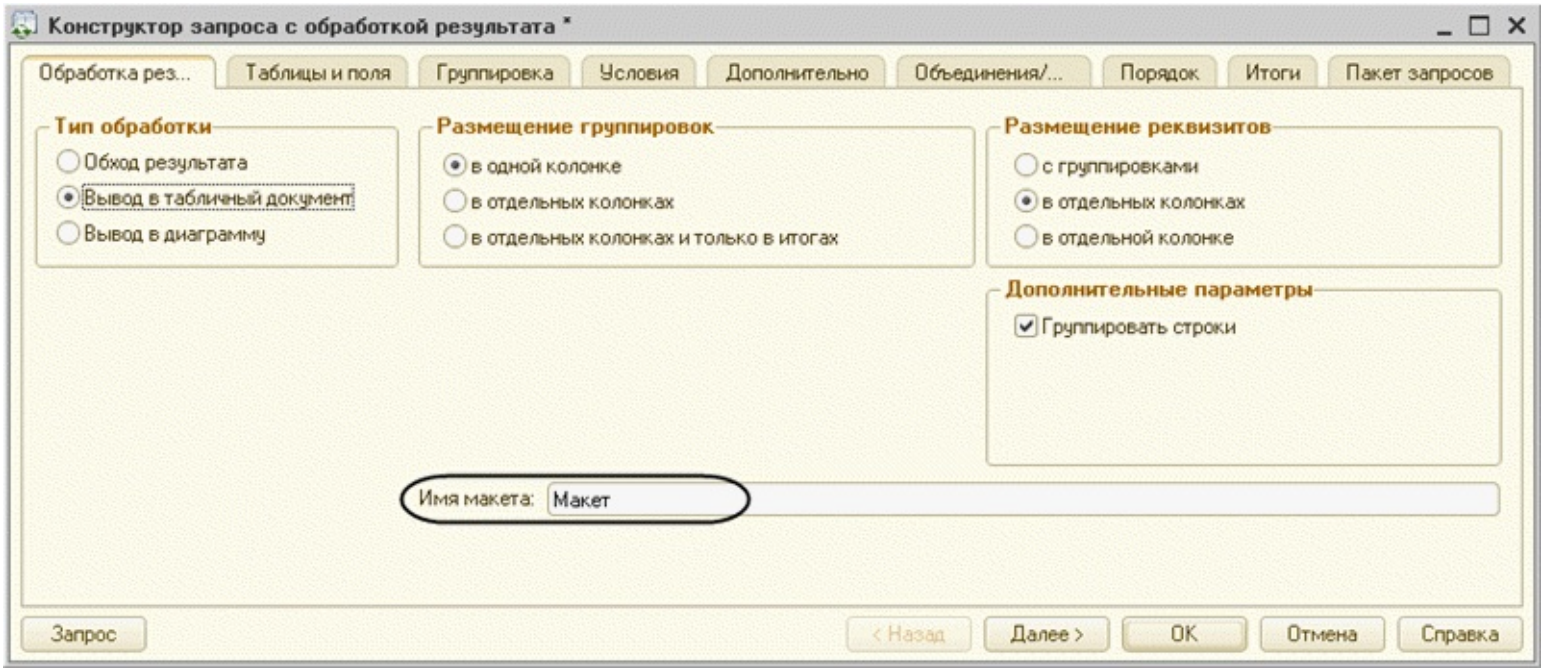


Рис. 2.40. Окно конструктора запроса с обработкой результата

Далее, как и в предыдущем случае, определим исходные данные для запроса. При выборе из таблицы *ЗаказТовара.Состав* ссылочных полей *Клиент* и *Товар* конструктор автоматически добавит в список выборки текстовое представление этих полей, полученное с помощью функции *Представление()*, рис. 2.41.

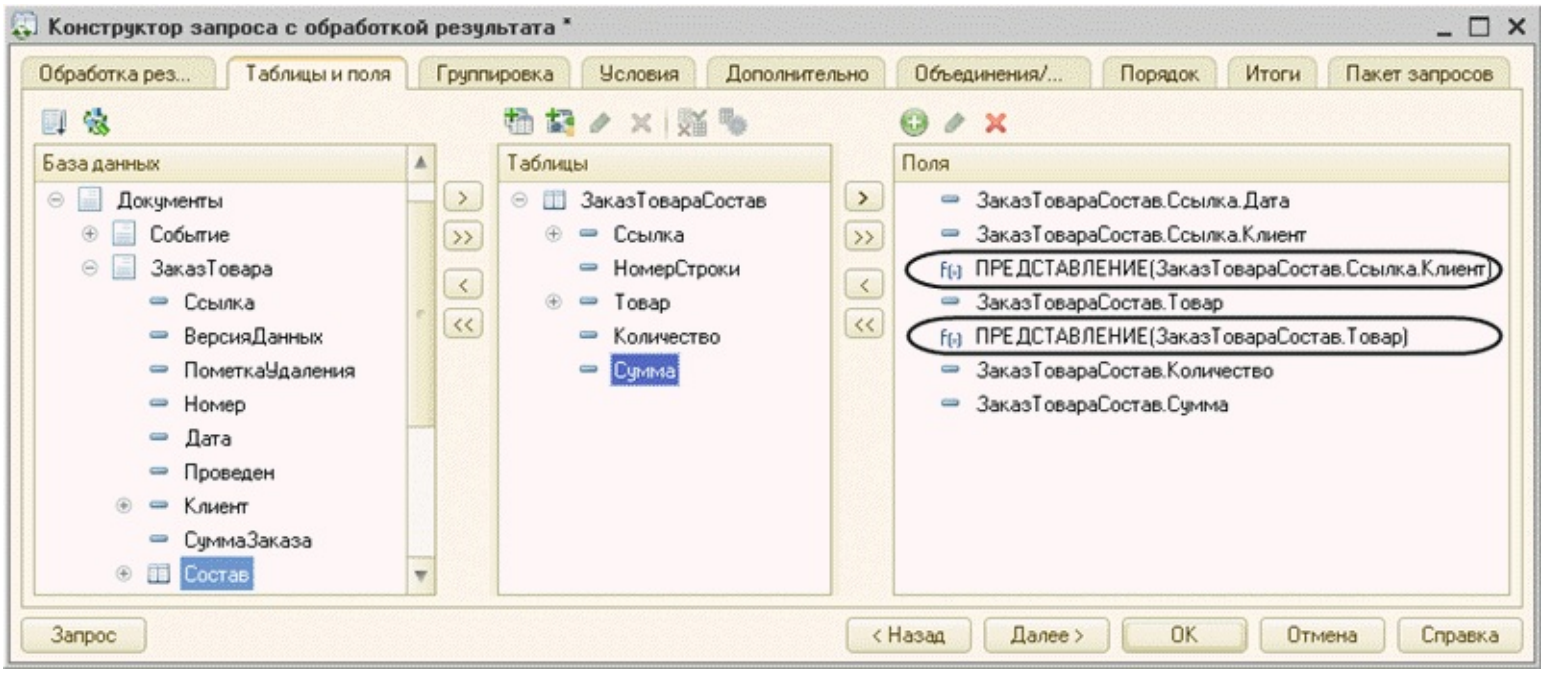


Рис. 2.41. Окно конструктора запроса с обработкой результата

Это сделано для того, чтобы избежать замедления при выполнении запроса. Дело в том, что при выводе значения ссылочного поля, для получения его представления, будет выполняться дополнительный запрос к той таблице, на которую ссылается это ссылочное поле. Поэтому следует в запросе сразу получать текстовое представление ссылочного поля и затем уже его, а не саму ссылку, выводить в табличный документ.

подробнее

О функции *Представление()* рассказано в разделе «Как получить текстовое представление

ссылочного поля».

Рекомендации по оптимизации запросов при выводе ссылочных полей в отчет даны в разделе "Исключить вывод ссылочных полей в отчет".

Нажмем *ОК*. Конструктор запроса создаст подчиненный объект обработки (в которой мы поместили процедуру для вывода результата запроса в табличный документ) – макет табличного документа с полями, перечисленными в выборке запроса. Исходя из текста запроса, в макете присутствуют области итоговых группировок *Клиент*, *Товар* и область для вывода детальных записей, а также области для вывода заголовка и подвала табличного документа, шапки и подвала таблицы (рис. 2.42).

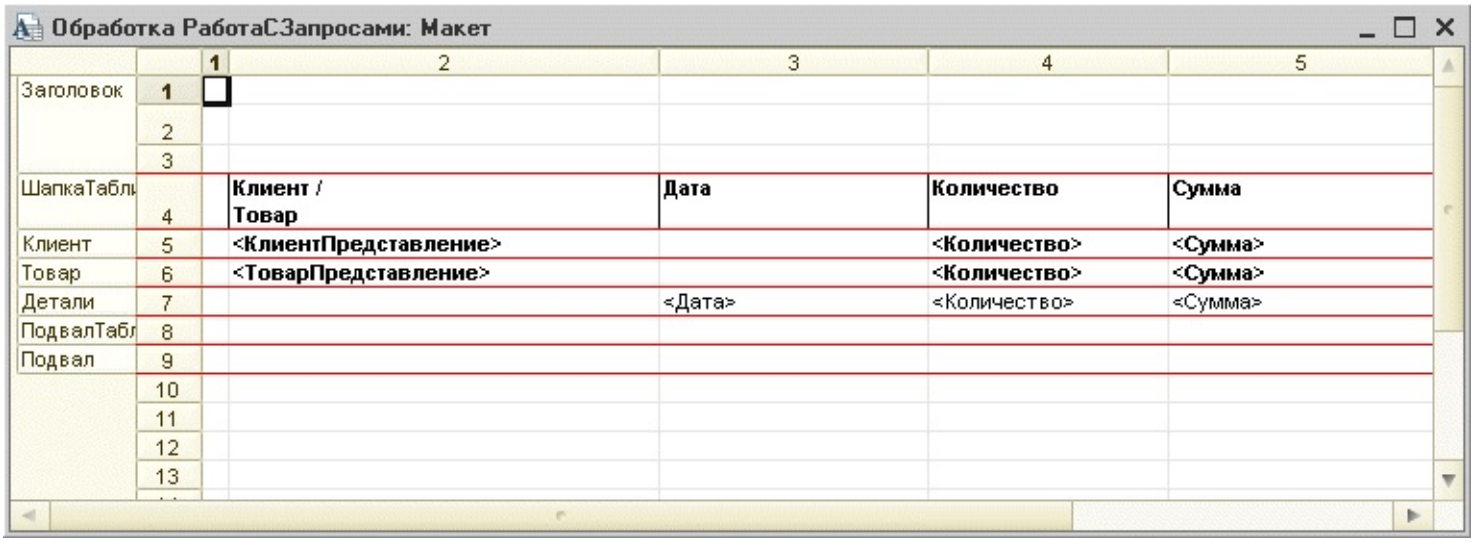


Рис. 2.42. Макет табличного документа, созданный конструктором

Также в функции *ЗаполнитьТД()* (откуда был вызван конструктор) конструктор запроса с обработкой результата сформирует следующий фрагмент кода (листинг 2.29).

Листинг 2.29. Фрагмент процедуры для вывода результата запроса в табличный документ, созданный с помощью конструктора запроса с обработкой результата

```
//{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
// Данный фрагмент построен конструктором.
// При повторном использовании конструктора внесенные вручную изменения будут утеряны!!!

Макет = Обработки.РаботаСЗапросами.ПолучитьМакет("Макет");
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |         ЗаказТовараСостав.Ссылка.Дата,
  |         ЗаказТовараСостав.Ссылка.Клиент КАК Клиент,
  |         ПРЕДСТАВЛЕНИЕ(ЗаказТовараСостав.Ссылка.Клиент),
  |         ЗаказТовараСостав.Товар КАК Товар,
  |         ПРЕДСТАВЛЕНИЕ(ЗаказТовараСостав.Товар),
  |         ЗаказТовараСостав.Количество КАК Количество,
  |         ЗаказТовараСостав.Сумма КАК Сумма
  |ИЗ
  |         Документ.ЗаказТовара.Состав КАК ЗаказТовараСостав
  |ИТОГИ
  |         СУММА(Количество),
  |         СУММА(Сумма)
```

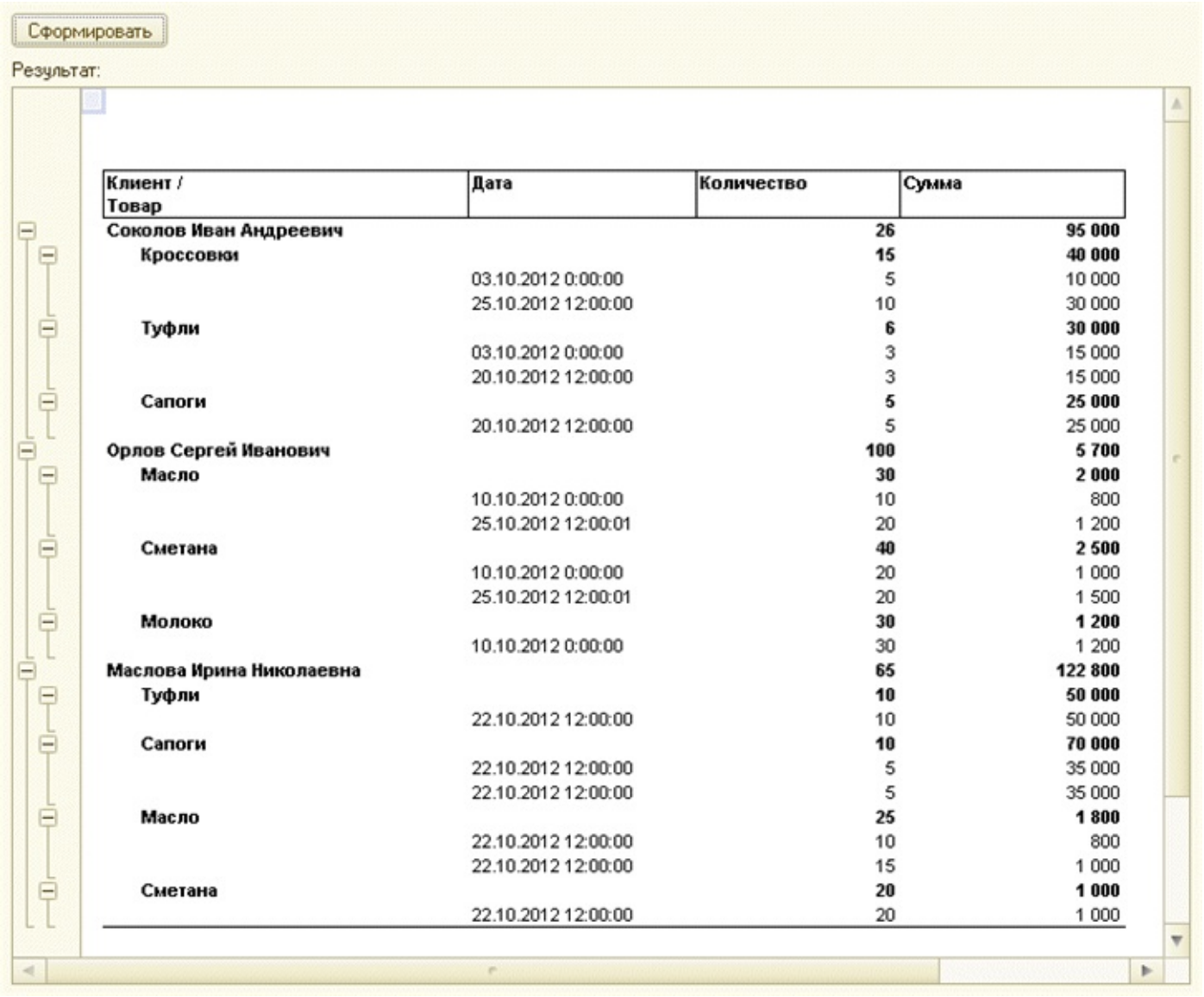
```
    |ПО
    |        Клиент,
    |        Товар";
РезультатЗапроса = Запрос.Выполнить();

ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");
ОбластьШапкаТаблицы = Макет.ПолучитьОбласть("ШапкаТаблицы");
ОбластьПодвалТаблицы = Макет.ПолучитьОбласть("ПодвалТаблицы");
ОбластьКлиент = Макет.ПолучитьОбласть("Клиент");
ОбластьТовар = Макет.ПолучитьОбласть("Товар");
ОбластьДетальныхЗаписей = Макет.ПолучитьОбласть("Детали");

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

ВыборкаКлиент = РезультатЗапроса.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
Пока ВыборкаКлиент.Следующий() Цикл
  ОбластьКлиент.Параметры.Заполнить(ВыборкаКлиент);
  ТабДок.Вывести(ОбластьКлиент, ВыборкаКлиент.Уровень());

  ВыборкаТовар = ВыборкаКлиент.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
  Пока ВыборкаТовар.Следующий() Цикл
    ОбластьТовар.Параметры.Заполнить(ВыборкаТовар);
    ТабДок.Вывести(ОбластьТовар, ВыборкаТовар.Уровень());

    ВыборкаДетальныеЗаписи = ВыборкаТовар.Выбрать();
    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
      ОбластьДетальныхЗаписей.Параметры.Заполнить(ВыборкаДетальныеЗаписи);
      ТабДок.Вывести(ОбластьДетальныхЗаписей, ВыборкаДетальныеЗаписи.Уровень());

    КонецЦикла;

  КонецЦикла;

КонецЦикла;

ТабДок.ЗакончитьАвтогруппировкуСтрок();
ТабДок.Вывести(ОбластьПодвалТаблицы);
ТабДок.Вывести(ОбластьПодвал);

//}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
```

В результате в процедуре *ЗаполнитьТД()* сначала методом *ПолучитьМакет()* получается макет табличного документа с именем, указанным в конструкторе (см. рис. 2.40). Затем создается и выполняется запрос, определенный в конструкторе. Затем методом *ПолучитьОбласть()* получаются области табличного документа, заданные в макете. После этого табличный документ *ТабДок*, переданный в процедуру, очищается, и в него методом *Вывести()* выводятся области заголовка и шапки таблицы, полученные из макета. Затем начинается автоматическая группировка строк табличного документа методом *НачатьАвтогруппировкуСтрок()*, так как в конструкторе запроса с обработкой результата была по умолчанию включена опция *Группировать строки* (см. рис. 2.40).

Далее, как и предыдущем случае (см. листинг 2.27), начинается обход трех вложенных друг в друга выборок – итоговых группировок *Клиент*, *Товар* и детальных записей. В цикле обхода каждой выборки параметры соответствующих областей табличного документа, определенные в макете, заполняются данными выборок, и затем области с указанием уровня выборки выводятся в табличный документ методом *Вывести()*. Параметр *Уровень* этого метода используется для автоматической группировки строк результирующего табличного документа.

В заключение заканчивается автоматическая группировка строк табличного документа, и в него выводятся области подвала документа и подвала таблицы, полученные из макета.

Таким образом, при вызове процедуры *ВыводВТабличныйДокумент()* в поле табличного документа мы получим следующий результат (рис. 2.43).



Клиент / Товар	Дата	Количество	Сумма
Соколов Иван Андреевич		**26**	**95 000**
Кроссовки		**15**	**40 000**
	03.10.2012 0:00:00	5	10 000
	25.10.2012 12:00:00	10	30 000
Туфли		**6**	**30 000**
	03.10.2012 0:00:00	3	15 000
	20.10.2012 12:00:00	3	15 000
Сапоги		**5**	**25 000**
	20.10.2012 12:00:00	5	25 000
Орлов Сергей Иванович		**100**	**5 700**
Масло		**30**	**2 000**
	10.10.2012 0:00:00	10	800
	25.10.2012 12:00:01	20	1 200
Сметана		**40**	**2 500**
	10.10.2012 0:00:00	20	1 000
	25.10.2012 12:00:01	20	1 500
Молоко		**30**	**1 200**
	10.10.2012 0:00:00	30	1 200
Маслова Ирина Николаевна		**65**	**122 800**
Туфли		**10**	**50 000**
	22.10.2012 12:00:00	10	50 000
Сапоги		**10**	**70 000**
	22.10.2012 12:00:00	5	35 000
	22.10.2012 12:00:00	5	35 000
Масло		**25**	**1 800**
	22.10.2012 12:00:00	10	800
	22.10.2012 12:00:00	15	1 000
Сметана		**20**	**1 000**
	22.10.2012 12:00:00	20	1 000

Рис. 2.43. Вывод результата запроса в табличный документ

Мы видим, что в табличном документе записи сгруппированы по клиентам и товарам, по ним рассчитаны итоги, и мы можем свернуть или развернуть эти уровни благодаря группировке строк табличного документа.

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

подробнее

О работе с табличным документом подробнее можно прочитать в книге «Решение специальных прикладных задач в «1С:Предприятии 8.2» из серии «Профессиональная разработка».

Вывод в диаграмму

Теперь с помощью конструктора выведем результат этого же запроса в диаграмму. Поскольку диаграмма заполняется данными на сервере, а выводится пользователю на клиенте, сначала создадим реквизит формы *ДиаграммаЗаказов* типа *Диаграмма*, который и будет содержать данные диаграммы, и перетащим его в дерево элементов формы обработки. После этого создадим в модуле формы небольшую заготовку (листинг 2.30).

Листинг 2.30. Процедуры для вывода результата запроса в диаграмму

```
&НаКлиенте
Процедура ВыводВДиаграмму(Команда)

  ЗаполнитьДиаграмму(ДиаграммаЗаказов);

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ЗаполнитьДиаграмму(Диаграмма)

  // Фрагмент для вывода в табличный документ

КонецПроцедуры
```

В клиентской процедуре *ВыводВДиаграмму()* вызывается серверная внеконтекстная процедура *ЗаполнитьДиаграмму()*, в которую передается реквизит формы *ДиаграммаЗаказов*. В этой процедуре диаграмма заполняется данными и при возвращении на клиента отображается пользователю в поле формы вида *Поле диаграммы*.

Теперь создадим собственно фрагмент кода для вывода результата запроса в диаграмму. Для этого установим курсор внутрь процедуры *ЗаполнитьДиаграмму()* и вызовем из контекстного меню пункт *Конструктор запроса с обработкой результата*. Подтвердим, что мы хотим создать новый запрос.

На закладке конструктора *Обработка результатов* в группе *Тип обработки* выберем опцию *Вывод в диаграмму* (рис. 2.44).

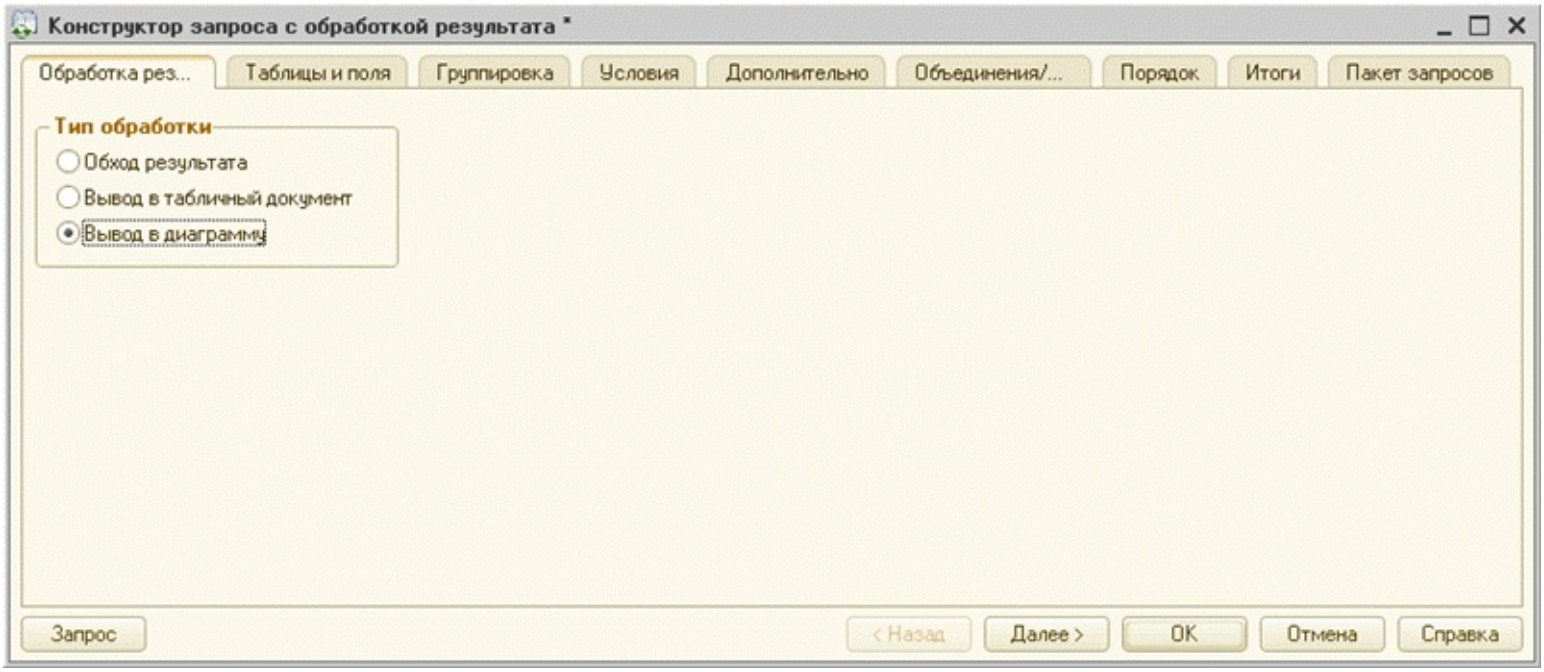


Рис. 2.44. Окно конструктора запроса с обработкой результата

На закладке конструктора *Таблицы и поля* несколько изменим запрос – уберем из списка полей выборки поля *Дата* и *Сумма*, так как они в этом примере не нужны (рис. 2.45).

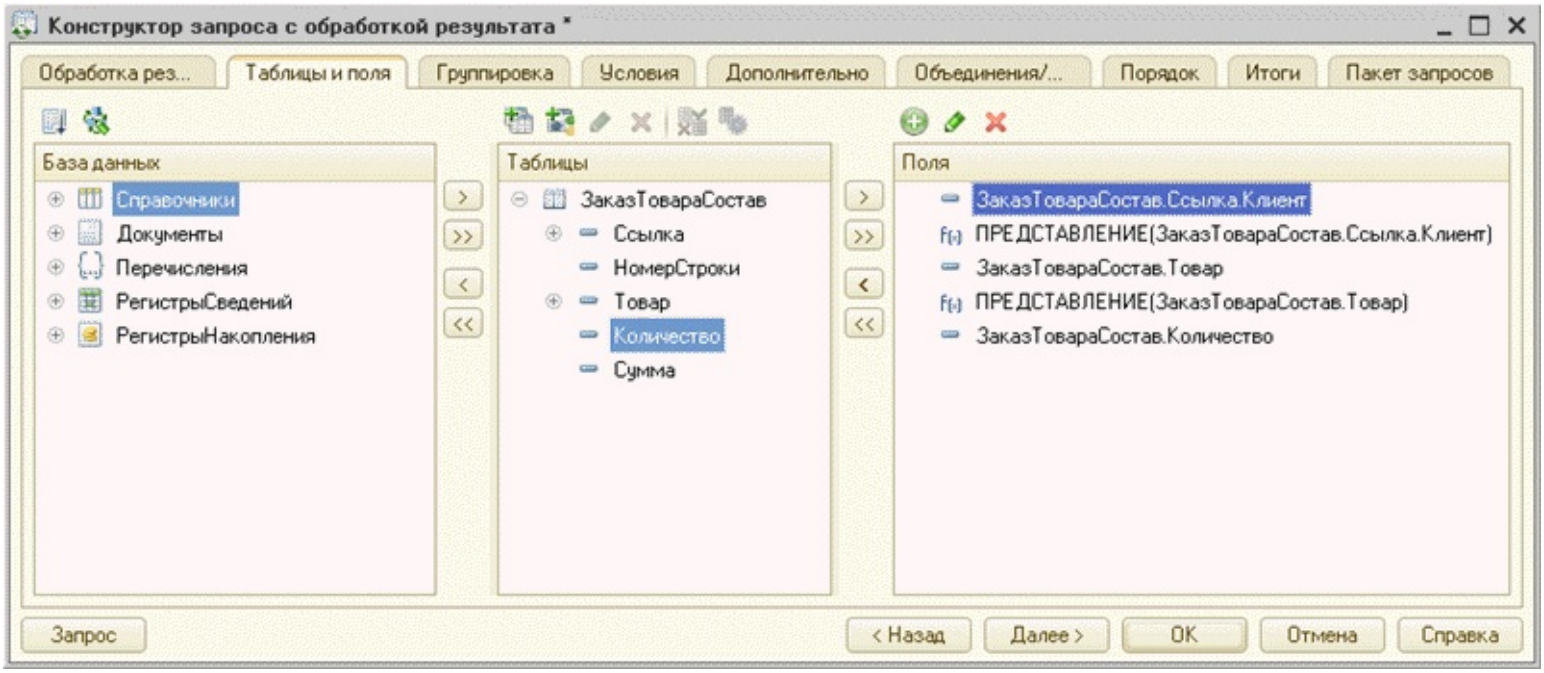


Рис. 2.45. Окно конструктора запроса с обработкой результата

На закладке конструктора *Итоги* из итоговых полей также уберем поле *Сумма* (рис. 2.46).

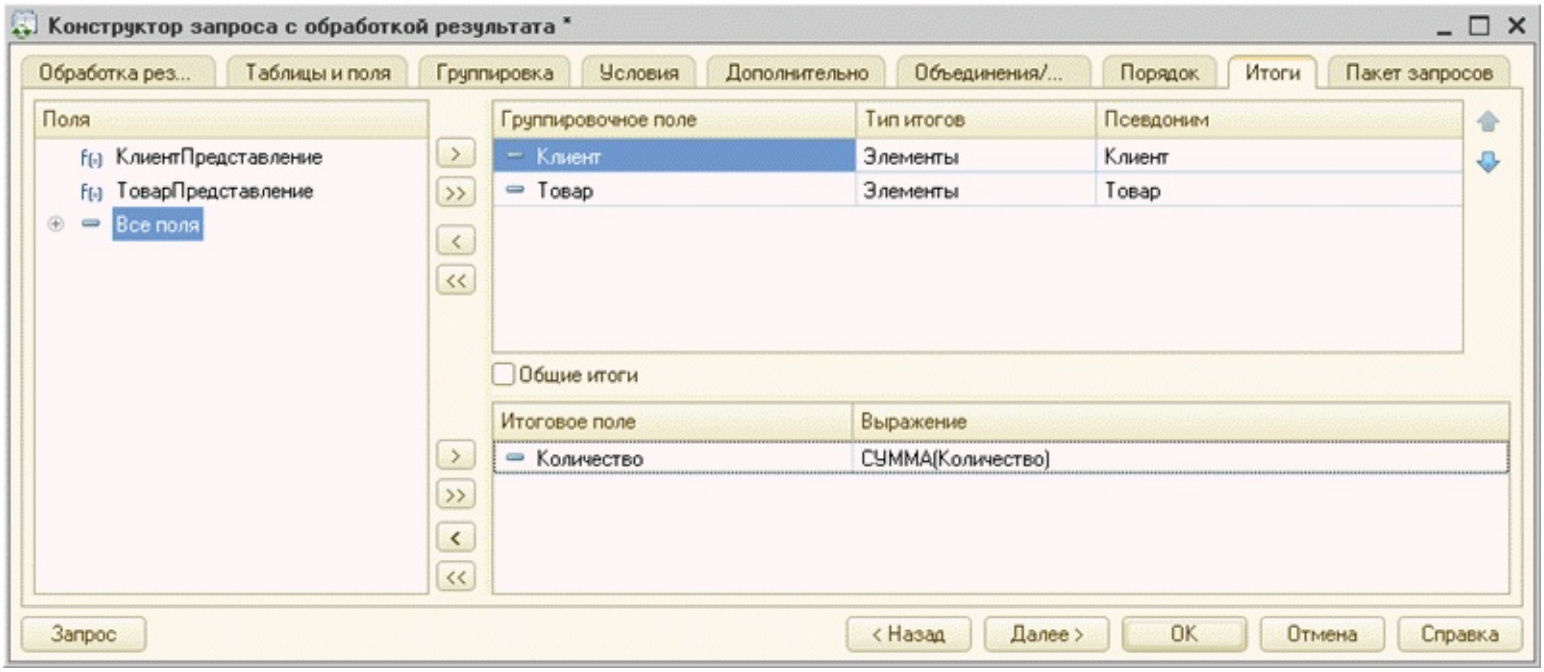


Рис. 2.46. Окно конструктора запроса с обработкой результата

Нажмем *ОК*. В процедуре *ЗаполнитьДиаграмму()* (откуда был вызван конструктор) конструктор запроса с обработкой результата сформирует следующий фрагмент кода (листинг 2.31).

Листинг 2.31. Фрагмент процедуры для вывода результата запроса в диаграмму, созданный с помощью конструктора запроса с обработкой результата

```
//{{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
// Данный фрагмент построен конструктором.
// При повторном использовании конструктора внесенные вручную изменения будут утеряны!!!

Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ЗаказТовараСостав.Ссылка.Клиент КАК Клиент,
  |        ПРЕДСТАВЛЕНИЕ(ЗаказТовараСостав.Ссылка.Клиент),
  |        ЗаказТовараСостав.Товар КАК Товар,
  |        ПРЕДСТАВЛЕНИЕ(ЗаказТовараСостав.Товар),
  |        ЗаказТовараСостав.Количество КАК Количество
  |ИЗ
  |        Документ.ЗаказТовара.Состав КАК ЗаказТовараСостав
  |ИТОГИ
  |        СУММА(Количество)
  |ПО
  |        Клиент,
  |        Товар";

РезультатЗапроса = Запрос.Выполнить();

Диаграмма.Обновление = Ложь;
Диаграмма.Очистить();
Диаграмма.АвтоТранспонирование = Ложь;

ВыборкаКлиент = РезультатЗапроса.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

Пока ВыборкаКлиент.Следующий() Цикл
  Серия = Диаграмма.УстановитьСерию(ВыборкаКлиент.Клиент);
```

```
    Серия.Текст = ВыборкаКлиент.КлиентПредставление;
    Серия.Расшифровка = ВыборкаКлиент.Клиент;

    ВыборкаТовар = ВыборкаКлиент.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
    Пока ВыборкаТовар.Следующий() Цикл
      Точка = Диаграмма.УстановитьТочку(ВыборкаТовар.Товар);
      Точка.Текст = ВыборкаТовар.ТоварПредставление;
      Точка.Расшифровка = ВыборкаТовар.Товар;
      Диаграмма.УстановитьЗначение(Точка, Серия, ВыборкаТовар.Количество);

    КонецЦикла;

КонецЦикла;

Диаграмма.АвтоТранспонирование = Истина;
Диаграмма.Обновление = Истина;

//}}КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
```

В результате в процедуре *ЗаполнитьДиаграмму()* сначала создается и выполняется запрос, определенный в конструкторе. Затем диаграмма *Диаграмма*, переданная в функцию, очищается, и на время ее заполнения данными отключается обновление и автотранспонирование диаграммы. Тип диаграммы здесь не определяется, но по умолчанию он устанавливается как *Гистограмма объемная*.

Далее начинается обход двух вложенных друг в друга выборок – итоговых группировок *Клиент* и *Товар*. В цикле обхода первой выборки *ВыборкаКлиент* методом *УстановитьСерию()* определяются серии диаграммы. В цикле обхода вложенной в нее выборки *ВыборкаТовар* методом *УстановитьТочку()* определяются точки диаграммы и методом *УстановитьЗначение()* устанавливается значение серии диаграммы в точке.

В заключение после заполнения данными диаграммы включается ее обновление и автотранспонирование. Это позволяет избежать многократной перерисовки диаграммы во время ее формирования.

Таким образом, при вызове процедуры *ВыводВДиаграмму()* в поле диаграммы мы получим следующий результат (рис. 2.47).

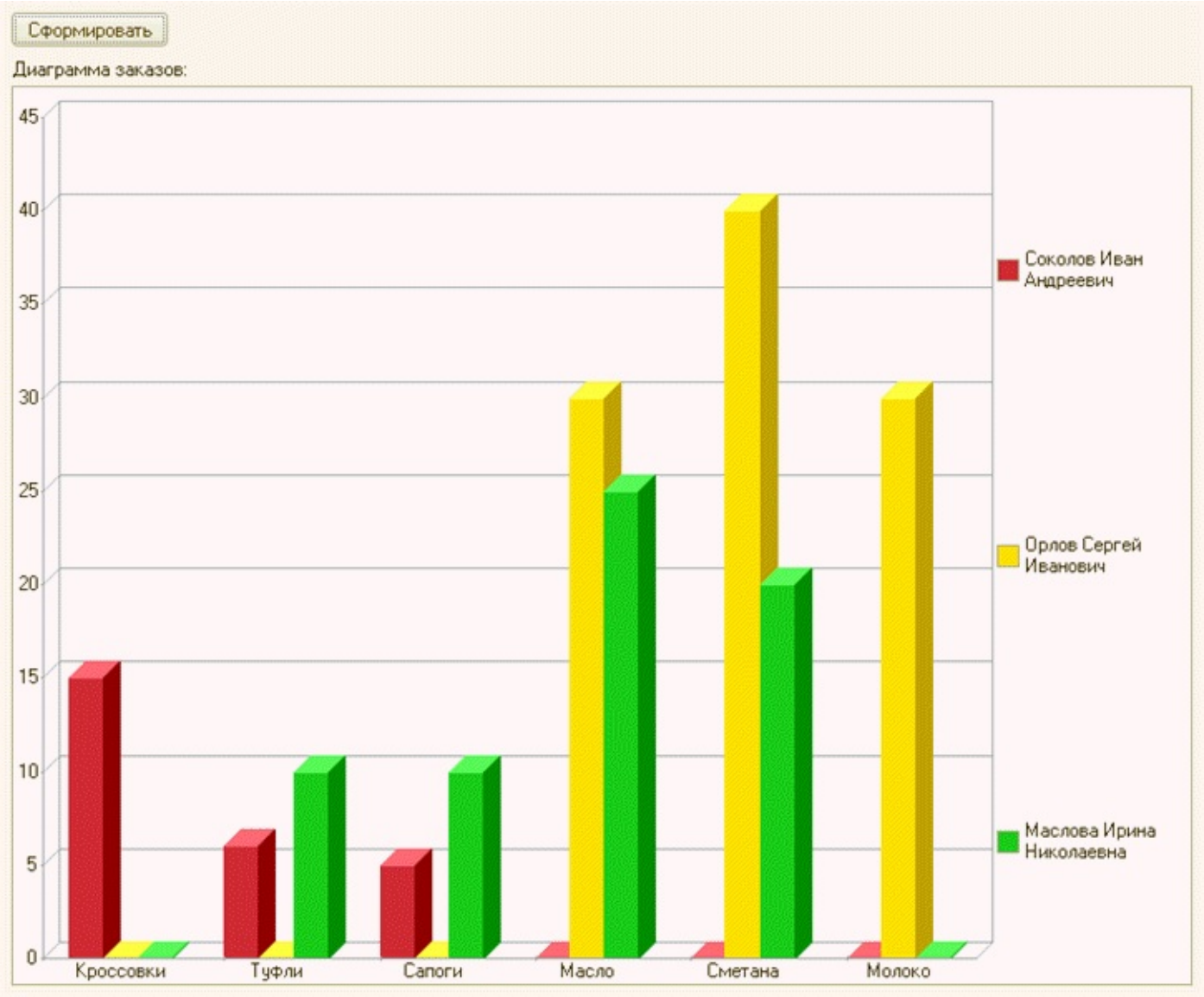


Рис. 2.47. Вывод результата запроса в диаграмму

Таким образом, мы видим те же данные о количестве заказанных товаров по клиентам, что и в поле табличного документа (см. рис. 2.43), но представленные в виде гистограммы.

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

подробнее

О работе с диаграммой подробнее можно прочитать в книге «Решение специальных прикладных задач в «1С:Предприятии 8.2» из серии «Профессиональная разработка».

Выгрузка результата запроса в таблицу или дерево значений

Иногда возможен вариант, при котором результат запроса выгружается в таблицу или дерево значений и затем выполняется программная обработка результатов запроса путем перебора строк этой таблицы значений.

Результат запроса можно выгрузить в таблицу значений или дерево значений с помощью метода *Выгрузить()* объекта *РезультатЗапроса*, который возвращает объект

ТаблицаЗначений или *ДеревоЗначений*, в зависимости от переданного параметра *ТипОбхода*.

Если устанавливается прямой тип обхода (по умолчанию), то будет создана таблица значений, иначе – дерево значений. Далее таблица значений/дерево значений может быть обработана средствами встроенного языка или показана пользователю в виде таблицы в форме (листинг 2.32).

Листинг 2.32. Варианты выгрузки результата запроса в таблицу значений

```
ТабЗнач = РезультатЗапроса.Выгрузить(); //по умолчанию прямой тип обхода
ДеревоЗнач = РезультатЗапроса.Выгрузить(ОбходРезультатаЗапроса.ПоГруппировкам);
```

Рассмотрим пример. Допустим, требуется показать в форме в виде таблицы список поставщиков и принадлежащих им расчетных счетов. Для этого нам потребуется выбрать информацию из справочника *Поставщики* и подчиненного ему справочника *РасчетныеСчета*.

Создадим реквизит формы *СписокПоставщиков* типа *ТаблицаЗначений* с колонками *Код*, *Наименование* и *Счета*, который и будет содержать данные результата запроса, и перетащим его в дерево элементов формы обработки. *СписокПоставщиков* будет заполняться данными в серверной процедуре *ЗаполнитьТЗ()* и при возвращении на клиент отображаться пользователю в соответствующей таблице формы (листинг 2.33).

Листинг 2.33. Заполнение таблицы значений результатами запроса

```
&НаСервере
Процедура ЗаполнитьТЗ()

  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |         Поставщики.Ссылка КАК Владелец,
    |         Поставщики.Код,
    |         Поставщики.Наименование
    |ИЗ
    |         Справочник.Поставщики КАК Поставщики";

  РезультатЗапроса = Запрос.Выполнить();

  ТабЗнач = РезультатЗапроса.Выгрузить();

  СписокПоставщиков.Очистить();

  Запрос2 = Новый Запрос;
  Запрос2.Текст =
    "ВЫБРАТЬ
    |         СчетаПоставщика.Наименование
    |ИЗ
    |         Справочник.РасчетныеСчета КАК СчетаПоставщика
    |ГДЕ
    |         СчетаПоставщика.Владелец = &Поставщик";
```

```
  Для Каждого Поставщик Из ТабЗнач Цикл
    НоваяСтрока = СписокПоставщиков.Добавить();
    НоваяСтрока.Код = Поставщик.Код;
    НоваяСтрока.Наименование = Поставщик.Наименование;

    Запрос2.УстановитьПараметр("Поставщик", Поставщик.Владелец);
    РезультатЗапроса = Запрос2.Выполнить();

    Выборка = РезультатЗапроса.Выбрать();

    СчетаПоставщика = "";
    Пока Выборка.Следующий() Цикл
      СчетаПоставщика = СчетаПоставщика + СОКРЛП(Выборка.Наименование) + ", ";

    КонецЦикла;

    НоваяСтрока.Счета = СчетаПоставщика;

  КонецЦикла;

КонецПроцедуры
```

В данной процедуре сначала выполняется запрос к справочнику-владельцу *Поставщики*. Результат запроса выгружается в таблицу значений *ТабЗнач*. Эта таблица значений будет содержать поля, соответствующие полям выборки запроса – *Код*, *Наименование* и *Владелец* (ссылка на поставщика). Затем реквизит *СписокПостащиков* очищается.

После этого создается второй запрос к подчиненному справочнику *РасчетныеСчета* с параметром *Поставщик*, по которому и будут отбираться записи, относящиеся к владельцу.

Далее организуется перебор записей таблицы значений *ТабЗнач* с помощью цикла *Для Каждого Из … Цикл*. В этом цикле добавляются записи в *СписокПостащиков*, и поля *Код* и *Наименование* этой таблицы значений заполняются результатами первого запроса.

Затем параметр *Поставщик* второго запроса устанавливается как значение поля *Владелец* текущей записи таблицы значений *ТабЗнач* (это значение содержит ссылку на текущего поставщика). После этого второй запрос выполняется, из результата запроса получается выборка. В результате обхода этой выборки формируется строка, содержащая список расчетных счетов, относящихся к данному поставщику, и записывается в поле *Счета* таблицы значений *СписокПоставщиков*.

Таким образом, при вызове процедуры *ЗаполнитьТЗ()* мы получим следующий результат (рис. 2.48).

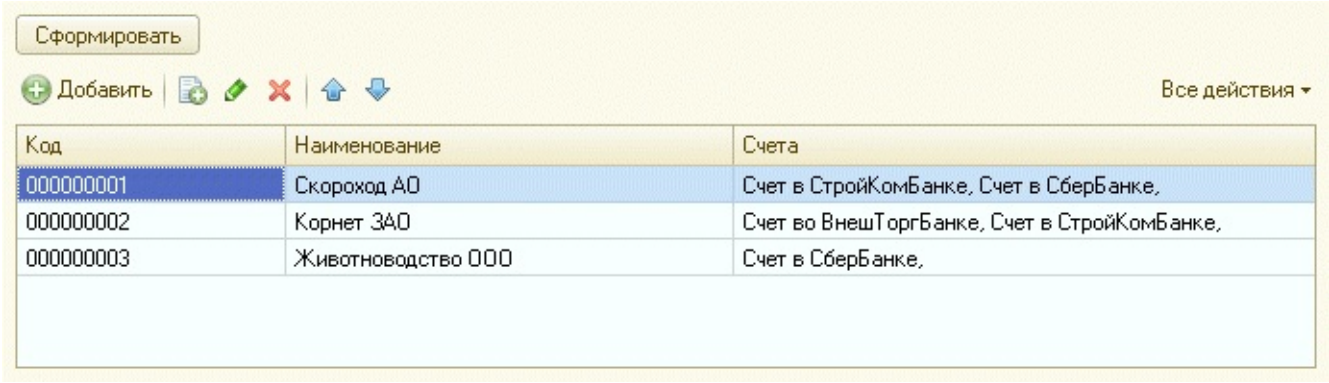


Рис. 2.48. Вывод результата запроса в таблицу значений

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Необходимо иметь в виду, что в данном примере (см. листинг 2.33) второй запрос выполняется в цикле обхода записей таблицы значений, содержащей результат запроса. Это сделано в чисто демонстрационных целях, на небольшой демонстрационной базе. Вообще же таких ситуаций (выполнения запросов в цикле) в целях оптимизации запросов нужно стараться избегать.

подробнее

Раздел «Не использовать запросы в цикле».

Отладка запросов

Возможность выгрузки результатов запроса в таблицу или дерево значений также может использоваться для отладки запросов. Отлаживать запросы, конечно, удобнее в консоли запросов. Там сразу можно посмотреть результат выполнения запроса, изменить текст и быстро оценить результат изменений.

Но бывают ситуации, когда, находясь в модуле, нужно быстро посмотреть, какие записи выбрал запрос. При этом переносить запрос в консоль сложно: например, устанавливается много специфических параметров запроса, или текст запроса формируется программно и т. п. Тогда можно выгрузить результат запроса в таблицу значений и посмотреть ее в отладчике.

Например, в предыдущем разделе в процедуре *ЗаполнитьТЗ()* после выполнения второго запроса и до получения выборки выгрузим результат запроса в таблицу значений (листинг 2.34).

Листинг 2.34. Фрагмент процедуры «ЗаполнитьТЗ()»

```
…
РезультатЗапроса = Запрос2.Выполнить();

ТЗ = РезультатЗапроса.Выгрузить();

Выборка = РезультатЗапроса.Выбрать();
…
```

Для того чтобы таблица значений *ТЗ* заполнилась значениями выполнения запроса, установим точку останова на следующую строку (*Выборка = РезультатЗапроса.Выбрать()*). Запустим «1С:Предприятие» в режиме отладки. Выполним команду для вывода результата запроса в таблицу значений в обработке *Работа с запросами*.

После остановки программы двойным щелчком выделим слово *ТЗ* и нажмем кнопку *Вычислить выражение* (*Shift + F9*) на панели инструментов *Отладка конфигурации*. Выделим строку *ТЗ* в окне *Результат* и нажмем кнопку *Показать значения в отдельном окне* (или *F2*) над окном результата, и мы увидим таблицу значений, содержащую результат выполнения запроса (рис. 2.49).

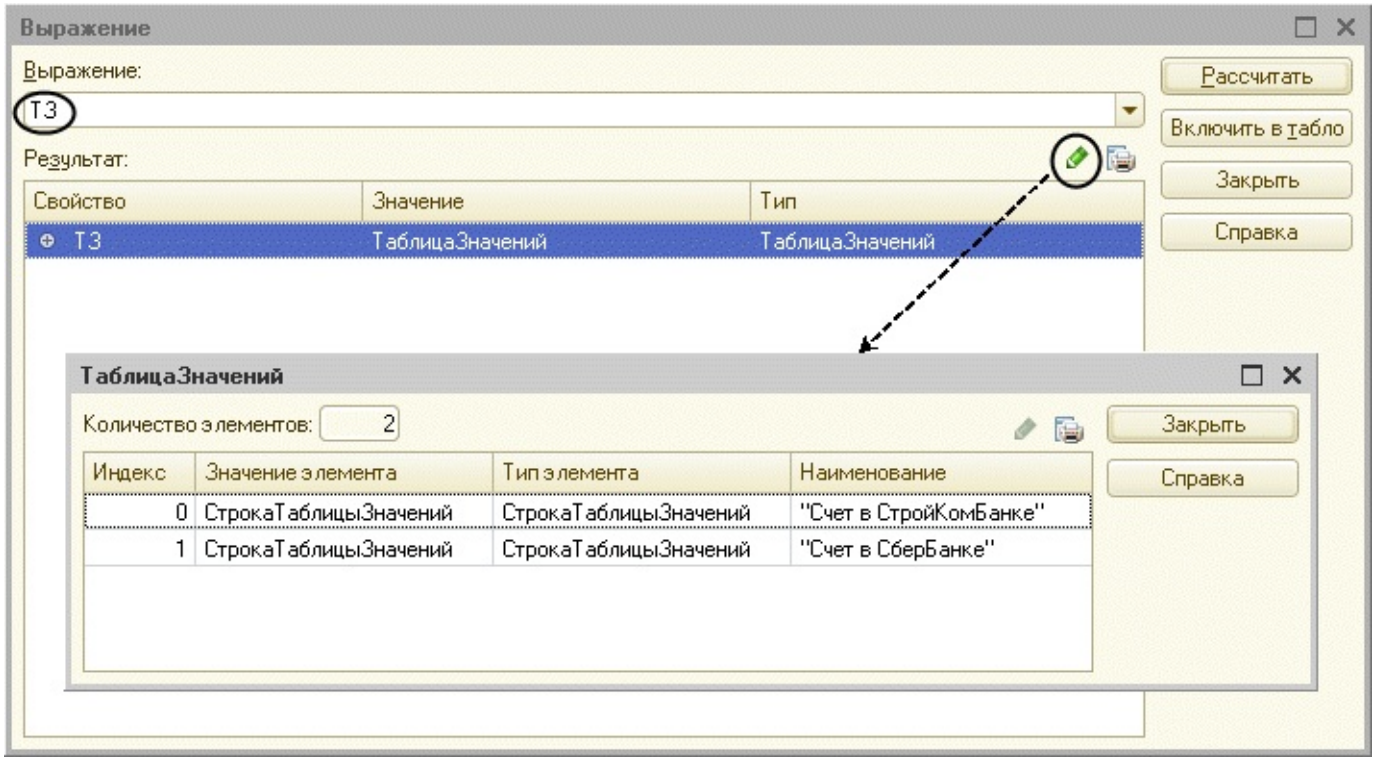


Рис. 2.49. Просмотр таблицы значений, содержащей результат запроса

Использование временных таблиц с помощью встроенного языка

В этом разделе мы рассмотрим, как использовать данные временной таблицы с помощью встроенного языка. В разделе «Временные таблицы и пакетные запросы» мы использовали данные временной таблицы в пакетном запросе. При этом в первом запросе пакета создавалась временная таблица *ПоступлениеТоваров*, содержащая данные о поступлении товаров за определенный период. А во втором запросе пакета таблица справочника *Товары* связывалась левым соединением с этой временной таблицей, и в результате выполнения данного пакетного запроса в консоли запросов мы получали список всех товаров в порядке иерархии справочника *Товары* с данными об их поступлении за период (листинг 2.35).

Листинг 2.35. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
ВЫБРАТЬ
  Поступление.Товар,
  Поступление.Ссылка.Дата КАК Дата,
```

```
  Поступление.Ссылка.Поставщик
ПОМЕСТИТЬ ПоступлениеТоваров
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК Поступление
ГДЕ
  МЕСЯЦ(Поступление.Ссылка.Дата) = 11
;
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  ПоступлениеТоваров.Дата,
  ПоступлениеТоваров.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ ПоступлениеТоваров КАК ПоступлениеТоваров
    ПО Товары.Ссылка = ПоступлениеТоваров.Товар

УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

Теперь выполним то же самое из встроенного языка.

Вспомним, что временная таблица не существует в базе данных, это просто некоторая область в памяти компьютера, которая создается и заполняется данными на ограниченное время.

Чтобы получить доступ к этим данным, используется объект встроенного языка *МенеджерВременныхТаблиц*, предназначенный для хранения данных временных таблиц. Для этого следует создать программный объект *МенеджерВременныхТаблиц*, затем создать объект *Запрос* и связать его с созданным менеджером временных таблиц через свойство запроса *МенеджерВременныхТаблиц* (листинг 2.36).

Листинг 2.36. Создание менеджера временных таблиц и установка его связи с запросом

```
МенеджерВТ = Новый МенеджерВременныхТаблиц;
Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
```

После этого все временные таблицы, созданные при помощи этого запроса, будут доступны в других запросах, использующих этот же менеджер временных таблиц.

Таким образом, пакетный запрос (см. листинг 2.35) можно переписать следующим образом (листинг 2.37).

Листинг 2.37. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос()

  МенеджерВТ = Новый МенеджерВременныхТаблиц;
  Запрос = Новый Запрос;
  Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
```

```
  Запрос.Текст = "ВЫБРАТЬ
  |        ПриходнаяНакладнаяСостав.Товар,
  |        ПриходнаяНакладнаяСостав.Ссылка.Дата,
  |        ПриходнаяНакладнаяСостав.Ссылка.Поставщик.Наименование КАК Поставщик
  |ПОМЕСТИТЬ ПоступлениеТоваров
  |ИЗ
  |        Документ.ПриходнаяНакладная.Состав КАК ПриходнаяНакладнаяСостав
  |ГДЕ
  |        МЕСЯЦ(ПриходнаяНакладнаяСостав.Ссылка.Дата) = 11";

  Запрос.Выполнить();

  Запрос2 = Новый Запрос;
  Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
  Запрос2.Текст = "ВЫБРАТЬ
  |        Товары.Наименование КАК Наименование,
  |        Товары.Производитель,
  |        ПоступлениеТоваров.Дата,
  |        ПоступлениеТоваров.Поставщик
  |ИЗ
  |        Справочник.Товары КАК Товары
  |                ЛЕВОЕ СОЕДИНЕНИЕ ПоступлениеТоваров КАК ПоступлениеТоваров
  |                ПО (ПоступлениеТоваров.Товар = Товары.Ссылка)
  |
  |УПОРЯДОЧИТЬ ПО
  |        Наименование ИЕРАРХИЯ";

  РезультатЗапроса = Запрос2.Выполнить();

  ВыборкаЗапроса = РезультатЗапроса.Выбрать();
  Сообщение = Новый СообщениеПользователю;

  Пока ВыборкаЗапроса.Следующий() Цикл
    Сообщение.Текст = "Товар: " + ВыборкаЗапроса.Наименование +
      " Производитель: " + ВыборкаЗапроса.Производитель +
      " Дата: " + ВыборкаЗапроса.Дата +
      " Поставщик: " + ВыборкаЗапроса.Поставщик;
    Сообщение.Сообщить();

  КонецЦикла;

КонецПроцедуры
```

В процедуре сначала создается менеджер временных таблиц *МенеджерВТ*, затем создается запрос *Запрос*, выбирающий данные о поступлении товаров за период. С ним связывается менеджер временных таблиц. При выполнении этого запроса (*Запрос.Выполнить()*) данные, выбранные запросом, помещаются во временную таблицу *ПоступлениеТоваров*. Результат выполнения первого запроса включает одну колонку *Количество*, содержащую количество строк, помещенных во временную таблицу.

Затем создается второй запрос *Запрос2*. В качестве менеджера временных таблиц для него также указывается *МенеджерВТ*. Это значит, что второй запрос получает доступ к временной таблице, созданной при выполнении первого запроса. То есть он может использовать временную таблицу *ПоступлениеТоваров* в качестве источника данных точно так же, как и таблицу базы данных.

После выполнения второго запроса выполняется обход выборки результата запроса, и в итоге в окне сообщений мы получаем аналогичный результат, что и при выполнении пакетного запроса (см. листинг 2.35) в консоли запросов (рис. 2.50).

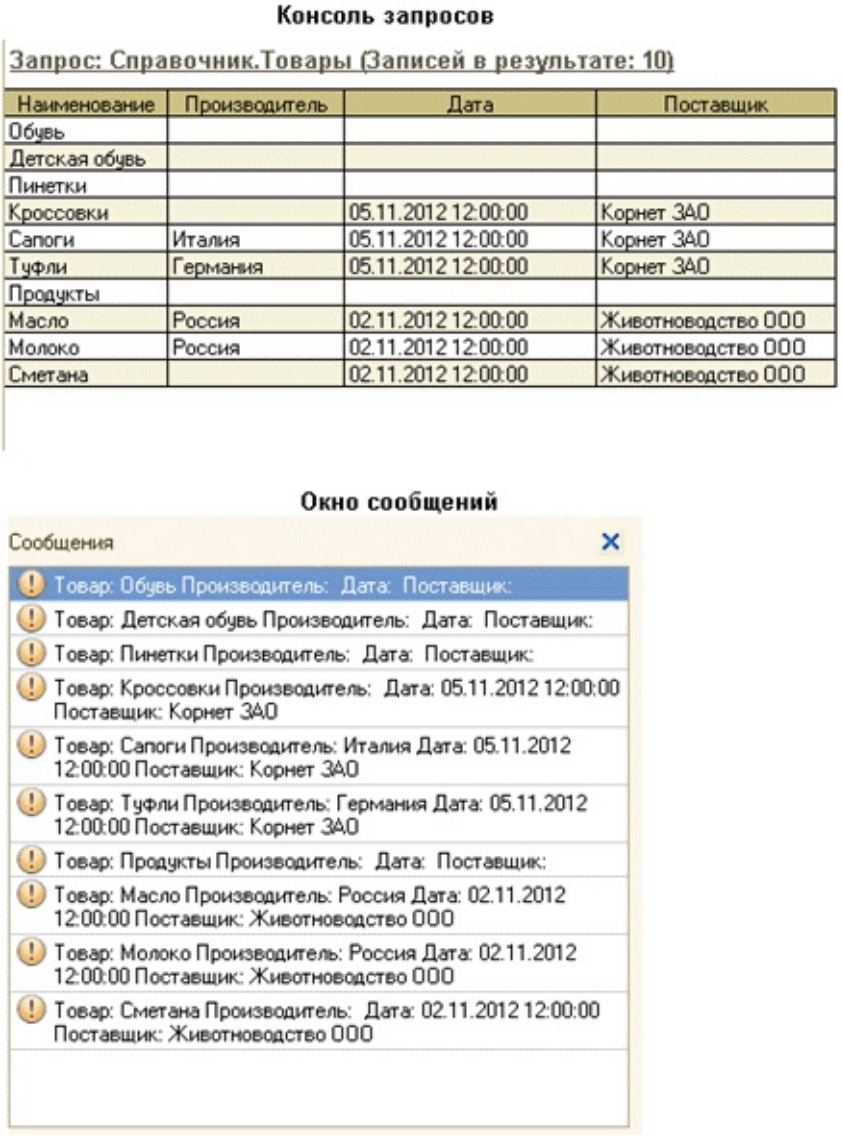


Рис. 2.50. Использование данных временной таблицы с помощью встроенного языка

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

При этом запросов, использующих общий менеджер временных таблиц, может быть несколько, и во всех этих запросах будут доступны данные всех временных таблиц, созданных с помощью запросов с использованием того же менеджера временных таблиц.

Необходимо учитывать, что временные таблицы будут существовать в памяти компьютера до закрытия менеджера временных таблиц (автоматически или принудительно методом *Закрыть()*) или до исполнения запроса, связанного с этим менеджером, уничтожающего временную таблицу с помощью конструкции *УНИЧТОЖИТЬ*.

Необходимо учитывать, что в одном прикладном решении может быть создано произвольное количество менеджеров временных таблиц, каждый из которых хранит свой

набор временных таблиц. Каждая временная таблица однозначно идентифицируется своим именем, и в пределах одного менеджера временных таблиц все временные таблицы должны иметь уникальные имена.

Использование таблицы значений в качестве источника временной таблицы

Временная таблица может быть создана как на основе таблиц базы данных, так и на основе внешнего источника данных. В качестве внешнего источника могут выступать:

- таблица значений,
- табличная часть,
- результат запроса.

Для того чтобы создать временную таблицу на основании внешнего источника, следует в тексте запроса в списке источников указать имя параметра, в который будет помещен внешний источник. Остальной синтаксис идентичен обычному созданию временной таблицы.

Рассмотрим пример создания временной таблицы на основе таблицы значений и дальнейшее использование этих данных в других запросах. Предположим, существует некая таблица значений, хранящая данные об оценках товаров (в числовом выражении), поставленных покупателями при покупке товаров. У каждого товара может быть множество оценок в этой таблице значений. Нам требуется вывести список всех товаров из справочника *Товары* и наряду с наименованием товара вывести соответствующую ему среднюю оценку пользователей, полученную из таблицы значений.

Как это реализовать? Для выполнения поставленной задачи сначала нам потребуется поместить данные из таблицы значений во временную таблицу. Затем запросом к этой временной таблице сгруппировать данные по товарам, получить среднюю оценку для каждого товара и поместить результат группировки в другую временную таблицу. После этого в основном запросе нам нужно связать левым соединением справочник товаров со второй временной таблицей по наименованиям товаров, выбрать нужные поля из таблиц и затем программно обработать результаты этого основного запроса.

Начнем выполнять наш план по частям и разбирать наши действия по мере выполнения.

Прежде всего, создадим таблицу значений об оценках товаров, поставленных покупателями, и заполним ее простым добавлением строк. Конечно, на самом деле таблица значений программно заполняется данными более сложным образом, но нас сейчас это не интересует. Мы рассмотрим самый простой вариант (листинг 2.38).

Листинг 2.38. Создание и заполнение таблицы значений

```
&НаСервереБезКонтекста
Функция ЗаполнитьТаблицуЗначений()

  КЧ = Новый КвалификаторыЧисла(12,2);
  КС = Новый КвалификаторыСтроки(20);
```

```
    Массив = Новый Массив;
    Массив.Добавить(Тип("Строка"));
    ОписаниеТиповС = Новый ОписаниеТипов(Массив, , КС);
    Массив.Очистить();
    Массив.Добавить(Тип("Число"));
    ОписаниеТиповЧ = Новый ОписаниеТипов(Массив, , ,КЧ);

    // Создание таблицы значений
    ТЗ = Новый ТаблицаЗначений;
    ТЗ.Колонки.Добавить("Товар",ОписаниеТиповС);
    ТЗ.Колонки.Добавить("Оценка",ОписаниеТиповЧ);

    // добавим строки в таблицу значений
    НоваяСтрока = ТЗ.Добавить();
    НоваяСтрока.Товар = "Туфли";
    НоваяСтрока.Оценка = 8;
    НоваяСтрока = ТЗ.Добавить();
    НоваяСтрока.Товар = "Молоко";
    НоваяСтрока.Оценка = 3;
    …
    НоваяСтрока = ТЗ.Добавить();
    НоваяСтрока.Товар = "Сапоги";
    НоваяСтрока.Оценка = 8;
    НоваяСтрока = ТЗ.Добавить();
    НоваяСтрока.Товар = "Туфли";
    НоваяСтрока.Оценка = 6;

    Возврат ТЗ;

КонецФункции
```

В данной функции мы создаем и заполняем данными таблицу значений *ТЗ*. Она будет содержать две колонки: *Товар* (типа *Строка*) и *Оценка* (типа *Число*). Здесь очень важен один нюанс. Обратите внимание, что при добавлении колонок в коллекцию колонок таблицы значений вторым параметром передается объект *ОписаниеТипов*, описывающий допустимые типы значений для колонки. Это важно. Если этого не сделать, то при выполнении запроса, в котором таблица значений выступает как источник данных, будет получена следующая ошибка (рис. 2.51).

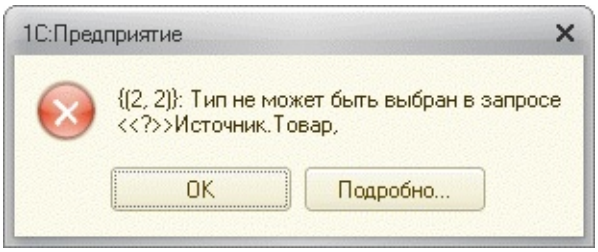


Рис. 2.51. Ошибка при использовании в запросе нетипизированной таблицы значений

Таким образом, чтобы запрос мог использовать в качестве внешнего источника таблицу значений, у этой таблицы значений должны быть явно указаны типы значений, содержащиеся в колонках. Фрагмент кода для создания типизированной таблицы значений выделен в листинге 2.38 жирным шрифтом.

Теперь поместим данные из таблицы значений во временную таблицу (листинг 2.39).

Листинг 2.39. Использование таблицы значений в качестве источника запроса

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос()

  ТЗ = ЗаполнитьТаблицуЗначений();

  МенеджерВТ = Новый МенеджерВременныхТаблиц;
  Запрос = Новый Запрос;
  Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
  Запрос.Текст = "ВЫБРАТЬ
    |         Источник.Товар,
    |         Источник.Оценка
    |ПОМЕСТИТЬ ОценкиТоваров
    |ИЗ
    |         &ВнешнийИсточник КАК Источник";

  Запрос.УстановитьПараметр("ВнешнийИсточник", ТЗ);
  Запрос.Выполнить();
  …
```

В данном фрагменте кода мы вызываем функцию *ЗаполнитьТаблицуЗначений()*, которая возвращает заполненную данными типизированную таблицу значений. После этого мы создаем менеджер временных таблиц *МенеджерВТ*, затем создаем запрос *Запрос*, который помещает данные из таблицы значений во временную таблицу *ОценкиТоваров*. С ним связывается менеджер временных таблиц. Это значит, что все другие запросы, использующие этот менеджер, будут иметь доступ к данным временной таблицы.

Особенность использования внешнего источника в качестве источника запроса состоит в том, что в списке источников в предложении *ИЗ* мы указываем параметр *&ВнешнийИсточник* и затем при установке значения параметра передаем в него таблицу значений *ТЗ*.

Теперь, поскольку у каждого товара может быть множество оценок в таблице значений, нам нужно сгруппировать данные полученной временной таблицы *ОценкиТоваров* по товарам, получить среднюю оценку для каждого товара и поместить результат группировки в другую временную таблицу (листинг 2.40).

Листинг 2.40. Группировка данных временной таблицы, получение усредненных оценок по каждому товару

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос()

  ТЗ = ЗаполнитьТаблицуЗначений();

  …
  Запрос2 = Новый Запрос;
  Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
  Запрос2.Текст = "ВЫБРАТЬ
    |         ОценкиТоваров.Товар,
    |         СРЕДНЕЕ(ОценкиТоваров.Оценка)
    |ПОМЕСТИТЬ СредниеОценки
    |ИЗ
```

```
    |        ОценкиТоваров КАК ОценкиТоваров
    |СГРУППИРОВАТЬ ПО
    |        ОценкиТоваров.Товар";

  Запрос2.Выполнить();
  …
```

В данном фрагменте кода мы создаем второй запрос *Запрос2*. В качестве менеджера временных таблиц для него также указываем *МенеджерВТ*. Это значит, что второй запрос получает доступ к временной таблице, созданной при выполнении первого запроса.

При выполнении второго запроса данные временной таблицы *ОценкиТоваров* группируются по товарам, для каждого товара получается усредненная оценка, и результат группировки помещается в другую временную таблицу *СредниеОценки*. Таким образом, в менеджере временных таблиц *МенеджерВТ* хранятся данные уже двух временных таблиц.

Напрашивается вывод, что можно было сгруппировать данные еще на первом этапе, но дело в том, что если временная таблица создается на основании внешнего источника, в запросе есть ряд ограничений. В частности, нельзя использовать операции группировки, объединения и соединения и др.

Подробнее

Документация «1С:Предприятие 8.3. Руководство разработчика», раздел 8.2 «Язык запросов», а также встроенная справка *Справка > Содержание справки > 1С:Предприятие > Встроенный язык > Работа с запросами > Выполнение и работа с запросами во встроенном языке > Работа с временными таблицами*.

Теперь создадим собственно основной запрос для вывода списка товаров из справочника *Товары* и соответствующих им оценок, полученных из таблицы значений. Так как нужно вывести данные о средних оценках товаров, то основной запрос будет использовать данные временной таблицы *СредниеОценки*, полученной при выполнении второго запроса (листинг 2.41).

Листинг 2.41. Вывод всех товаров из справочника «Товары» с соответствующими им усредненными пользовательскими оценками

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос()

  ТЗ = ЗаполнитьТаблицуЗначений();

  …
  Запрос3 = Новый Запрос;
  Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
  Запрос3.Текст = "ВЫБРАТЬ
    |        Товары.Наименование,
    |        ЕСТЬNULL(СредниеОценки.Оценка, 0) КАК Оценка
    |ИЗ
    |        Справочник.Товары КАК Товары
```

```
    |                ЛЕВОЕ СОЕДИНЕНИЕ СреднииОценки КАК СреднииОценки
    |                ПО Товары.Наименование = СреднииОценки.Товар
    |ГДЕ
    |         Товары.ЭтоГруппа = ЛОЖЬ";

  РезультатЗапроса = Запрос3.Выполнить();

  ВыборкаЗапроса = РезультатЗапроса.Выбрать();

  Сообщение = Новый СообщениеПользователю;

  Пока ВыборкаЗапроса.Следующий() Цикл
    Рейтинг = "";
    Для Индекс = 1 По ВыборкаЗапроса.Оценка Цикл
      Рейтинг = Рейтинг + "*";

    КонецЦикла;

    Сообщение.Текст = "Товар: " + ВыборкаЗапроса.Наименование + " " + Рейтинг;
    Сообщение.Сообщить();

  КонецЦикла;

КонецПроцедуры
```

В данном фрагменте кода мы создаем третий запрос *Запрос3*. В качестве менеджера временных таблиц для него также указываем *МенеджерВТ*. Это значит, что третий запрос получает доступ к временным таблицам, созданным при выполнении первого и второго запросов.

При выполнении третьего запроса данные справочника товаров связываются левым соединением со второй временной таблицей *СреднииОценки* по наименованиям товаров. Из справочника товаров выводятся наименования всех товаров, а также для каждого товара мы выводим значение поля *Оценка* из временной таблицы.

Как уже говорилось ранее, для тех товаров из справочника, для которых не найдено соответствий во временной таблице, в поле выборки запроса *Оценка* будут находиться значения типа *NULL*. При этом если посмотреть значение этого поля в отладчике, то мы увидим просто пустую строку, так как представление *NULL* значений – пустая строка. Чтобы иметь возможность при обходе выборки обращаться к значениям поля *Оценка* как к числовым значениям, мы при помощи функции языка запросов *ЕСТЬNULL()* заменяем значения типа *NULL* числом *0*.

Затем выполняется обход выборки результата запроса, и в итоге в окне сообщений мы видим список товаров и их оценок, поставленных пользователями. Причем оценка товара выводится в виде строки символов «*», число которых соответствует числовой оценке товара (рис. 2.52).

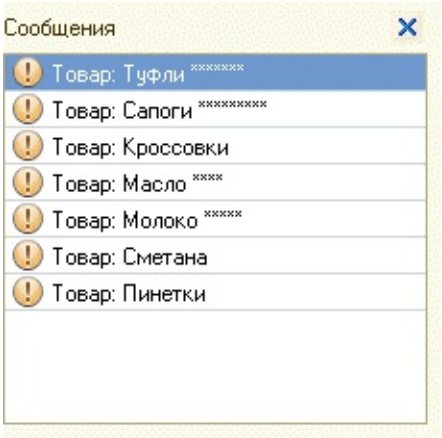


Рис. 2.52. Использование таблицы значений в качестве источника временной таблицы

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Примеры решения различных задач с использованием запросов

В этом разделе мы рассмотрим примеры решения некоторых часто встречающихся задач, касающихся использования запросов из встроенного языка, которые еще не были показаны во второй главе этой книги.

Поиск всех родителей для элемента иерархического справочника

Часто требуется построить всю цепочку вверх по иерархии, то есть получить список всех родительских записей, от выбранного элемента иерархической таблицы. Для этого рекомендуется перебирать в цикле его родителей небольшими порциями.

Например, в следующем запросе в окно сообщений выводится список всех родителей для ссылки на текущий элемент справочника *Товары*, которая передается в процедуру (листинг 2.42).

Листинг 2.42. Получение всех родителей для элемента иерархического справочника

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос(ТекущийЭлементНоменклатуры)

  Запрос = Новый Запрос;
  Запрос.Текст = "ВЫБРАТЬ
    |         Товары.Родитель,
    |         Товары.Родитель.Родитель,
    |         Товары.Родитель.Родитель.Родитель,
    |         Товары.Родитель.Родитель.Родитель.Родитель,
    |         Товары.Родитель.Родитель.Родитель.Родитель.Родитель
    |ИЗ
    |         Справочник.Товары КАК Товары
    |ГДЕ
    |         Товары.Ссылка = &ТекущийЭлементНоменклатуры";

  Сообщение = Новый СообщениеПользователю;

  Пока Истина Цикл
    Запрос.УстановитьПараметр("ТекущийЭлементНоменклатуры", ТекущийЭлементНоменклатуры);
    РезультатЗапроса = Запрос.Выполнить();
```

```
    Если РезультатЗапроса.Пустой() Тогда
      Прервать;

    КонецЕсли;

    Выборка = РезультатЗапроса.Выбрать();
    Выборка.Следующий();

    Для НомерКолонки = 0 По РезультатЗапроса.Колонки.Количество() - 1 Цикл
      ТекущийЭлементНоменклатуры = Выборка[НомерКолонки];
      Если ТекущийЭлементНоменклатуры = Справочники.Товары.ПустаяСсылка() Тогда
      Прервать;

      Иначе
        Сообщение.Текст = ТекущийЭлементНоменклатуры.Наименование;
        Сообщение.Сообщить();

      КонецЕсли;

    КонецЦикла;

    Если ТекущийЭлементНоменклатуры = Справочники.Товары.ПустаяСсылка() Тогда
      Прервать;

    КонецЕсли;

  КонецЦикла;

КонецПроцедуры
```

В этой процедуре сначала создается запрос, получающий родителей пяти уровней подчиненности (родитель, родитель от родителя и т. д.) для элемента иерархического справочника. Затем организуется цикл, в котором в качестве значения параметра запроса устанавливается ссылка на текущий элемент справочника (*ТекущийЭлементНоменклатуры*), переданная в процедуру, и запрос выполняется.

Если результат выполнения запроса не пустой, то из него получается выборка. Выборка запроса включает одну-единственную запись, на которую она позиционируется методом *Следующий()*, и пять колонок, содержащих значения полей результата запроса (*Родитель*, *РодительРодитель*, *РодительРодительРодитель* и т. д.).

После этого организуется вложенный в него цикл для обхода элементов коллекции колонок результата запроса. В теле этого цикла ссылка на текущий элемент справочника становится равной значению текущей колонки результата запроса (т. е. текущему родителю) и выводится в окно сообщений. В случае, если эта ссылка пустая, элемент справочника больше не имеет родителей, и цикл прерывается.

Если же обход всех колонок результата запроса (пяти уровней подчинения) завершен и ссылка на текущий элемент справочника не пустая, то внешний цикл продолжается. При этом в качестве параметра запроса устанавливается ссылка на последнего родителя, полученного во вложенном цикле, и выбираются новые пять родителей для этого

элемента.

Процесс продолжается до тех пор, пока в результате запроса не будет получена пустая ссылка на справочник *Товары*. Это значит, что родителей у элемента справочника больше нет.

Если количество уровней иерархии в справочнике ограничено и невелико, то можно получить всех родителей одним запросом без внешнего цикла. Например, в нашей демонстрационной конфигурации справочник *Товары* имеет трехуровневую иерархию, поэтому приведенную выше процедуру можно упростить следующим образом (листинг 2.43).

Листинг 2.43. Получение всех родителей для элемента иерархического справочника

```
&НаСервереБезКонтекста
Процедура ВыполнитьЗапрос(ТекущийЭлементНоменклатуры)

  Запрос = Новый Запрос;
  Запрос.Текст = "ВЫБРАТЬ
    |         Товары.Родитель,
    |         Товары.Родитель.Родитель
    |ИЗ
    |         Справочник.Товары КАК Товары
    |ГДЕ
    |         Товары.Ссылка = &ТекущийЭлементНоменклатуры";

  Запрос.УстановитьПараметр("ТекущийЭлементНоменклатуры", ТекущийЭлементНоменклатуры);
  РезультатЗапроса = Запрос.Выполнить();

  Если РезультатЗапроса.Пустой() Тогда
    Возврат;

  КонецЕсли;

  Сообщение = Новый СообщениеПользователю;

  Выборка = РезультатЗапроса.Выбрать();
  Выборка.Следующий();

  Для НомерКолонки = 0 По РезультатЗапроса.Колонки.Количество() - 1 Цикл
    ТекущийЭлементНоменклатуры = Выборка[НомерКолонки];
    Если ТекущийЭлементНоменклатуры = Справочники.Товары.ПустаяСсылка() Тогда
      Прервать;

    Иначе
      Сообщение.Текст = ТекущийЭлементНоменклатуры.Наименование;
      Сообщение.Сообщить();

    КонецЕсли;

  КонецЦикла;

КонецПроцедуры
```

Например, в корне справочника *Товары* есть группа товаров *Обувь*. Эта группа содержит

подчиненную группу *Детская обувь*, которая, в свою очередь, включает товар *Пинетки*. Если мы выберем элемент справочника *Пинетки*, то получим следующий результат (рис. 2.53).

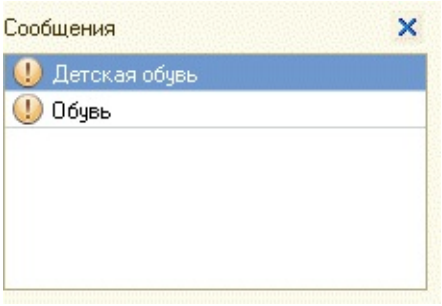


Рис. 2.53. Получение всех родителей для элемента иерархического справочника

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Создание запроса из произвольного источника

Иногда бывает нужно сформировать и выполнить запрос из произвольного источника. Например, требуется получить список товаров, использованных в любой табличной части любого документа конфигурации. То есть заранее мы не знаем, из какого документа и из какой табличной части нужно получать данные, знаем только, что интересующий нас реквизит табличной части имеет имя *Товар* и ссылочный тип на справочник *Товары*.

Для реализации поставленной задачи с помощью свойства глобального контекста *Метаданные* нам нужно обойти коллекцию всех документов, существующих в конфигурации, и всех табличных частей этих документов. И при этом динамически сформировать текст запроса, объединяющего запросы, которые получают данные из каждой табличной части каждого документа (листинг 2.44).

Листинг 2.44. Динамическое формирование текста запроса

```
&НаСервереБезКонтекста
Процедура ПолучитьСписокТоваров()

  Запрос = Новый Запрос;

  Для Каждого Док Из Метаданные.Документы Цикл
    Для Каждого ТЧ Из Док.ТабличныеЧасти Цикл
      ИмяДокумента = Док.Имя;
      ИмяТЧ = ТЧ.Имя;

      Если НЕ Запрос.Текст = "" Тогда
        Запрос.Текст = Запрос.Текст + Символы.ПС +
          "
          |ОБЪЕДИНИТЬ ВСЕ
          |";
      КонецЕсли;

      Запрос.Текст = Запрос.Текст +
        "
        |ВЫБРАТЬ РАЗЛИЧНЫЕ
        |        Товары.Товар.Наименование КАК Товар,
```

```
        |        """ + ИмяДокумента + "." + ИмяТЧ + """ КАК Источник
        |ИЗ
        |        Документ." + ИмяДокумента + "." + ИмяТЧ + " КАК Товары";

    КонецЦикла;

  КонецЦикла;

  Запрос.Текст = Запрос.Текст + Символы.ПС +
    "
    |ИТОГИ ПО
    |        Источник";

  РезультатЗапроса = Запрос.Выполнить();

  Если РезультатЗапроса.Пустой() Тогда
    Возврат;

  КонецЕсли;

  Сообщение = Новый СообщениеПользователю;

  ВыборкаИсточник = РезультатЗапроса.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

  Пока ВыборкаИсточник.Следующий() Цикл
    Сообщить(" Документ: " + ВыборкаИсточник.Источник);

    ВыборкаТовар = ВыборкаИсточник.Выбрать();

    Пока ВыборкаТовар.Следующий() Цикл
      Сообщение.Текст = "Товар: " + ВыборкаТовар.Товар;
      Сообщение.Сообщить();

    КонецЦикла;

  КонецЦикла;

КонецПроцедуры
```

В данной процедуре организуется цикл для обхода элементов коллекции документов конфигурации. Затем организуется вложенный в него цикл для обхода табличных частей каждого документа. В теле этого цикла динамически формируется имя источника запроса, исходя из имени документа и имени табличной части документа в метаданных (*"Документ." + ИмяДокумента + "." + ИмяТЧ + " КАК Товары"*).

Текст запроса, объединяющего запросы из каждого источника, формируется в несколько проходов вложенного цикла. Если текст запроса не пустой, то в него добавляется конструкция *ОБЪЕДИНИТЬ ВСЕ*. Обратите внимание, что помимо наименования товара в каждом запросе выбирается строковой литерал с именем табличной части документа под псевдонимом *Источник*. Чтобы использовать строку с именем источника в тексте запроса, мы должны вокруг имени источника (*""" + ИмяДокумента + "." + ИмяТЧ + """ КАК Источник*) написать подряд две кавычки, плюс еще одну кавычку, которая ограничивает присоединяемый фрагмент текста запроса.

После выхода из циклов мы присоединяем к общему запросу секцию описания итогов по полю *Источник*. Это сделано просто для лучшего визуального восприятия, чтобы таким образом сгруппировать товары, относящиеся к одному документу.

Чтобы посмотреть текст запроса, сформированного в результате обхода документов и их табличных частей, установим точку останова на строку *РезультатЗапроса = Запрос.Выполнить()*. Запустим «1С:Предприятие» в режиме отладки. После остановки программы двойным щелчком выделим слово *Запрос* и нажмем кнопку *Вычислить выражение (Shift + F9)* на панели инструментов *Отладка конфигурации*. Выделим строку *Текст* в окне *Результат* и нажмем кнопку *Показать значения в отдельном окне* (или *F2*) над окном результата, и мы увидим динамически сформированный текст запроса (рис. 2.54).

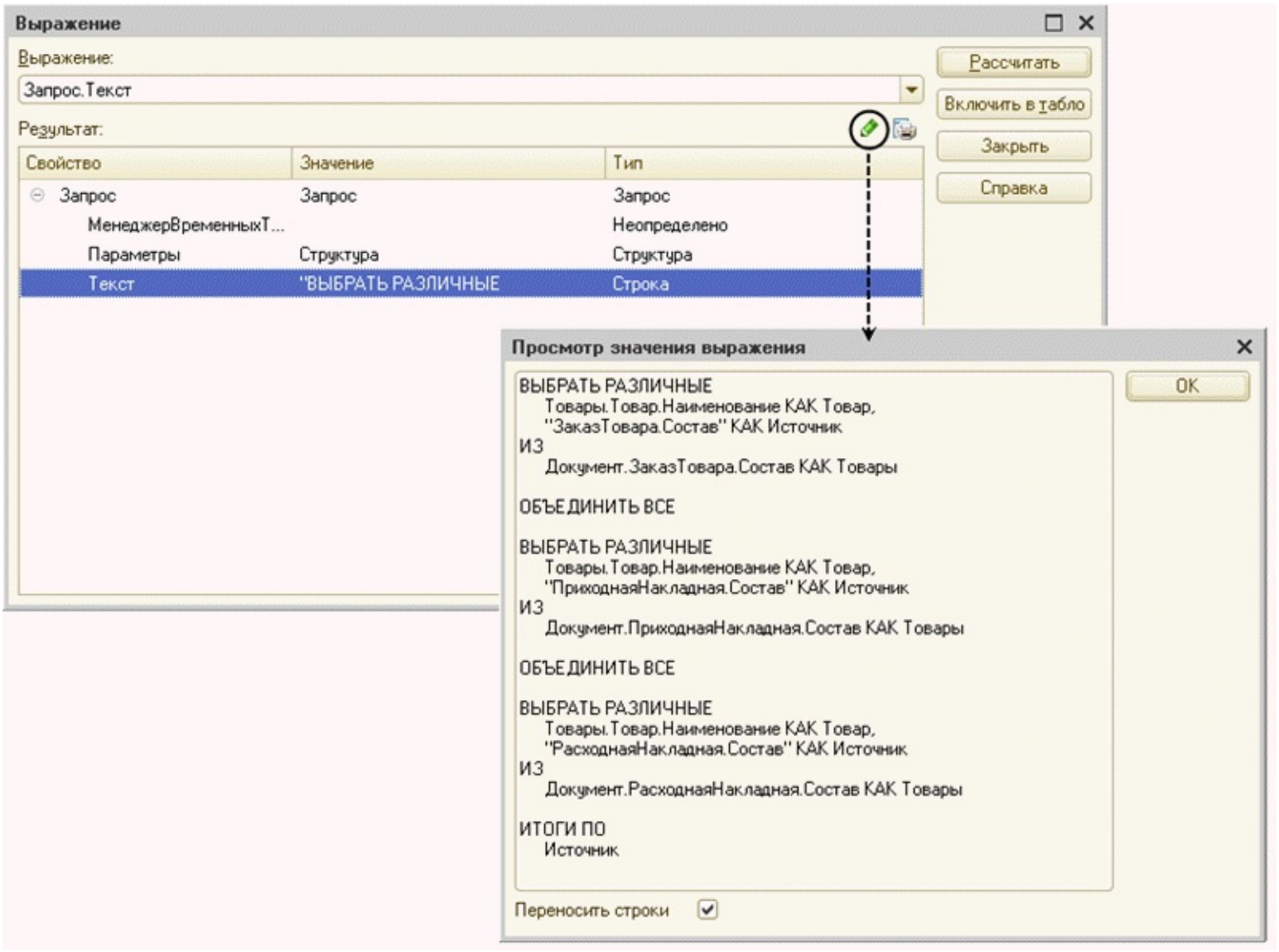


Рис. 2.54. Отладка текста запроса

После формирования запроса он выполняется, и список всех неповторяющихся товаров (*ВЫБРАТЬ РАЗЛИЧНЫЕ*) из табличной части каждого документа выводится в окно сообщений (рис. 2.55).

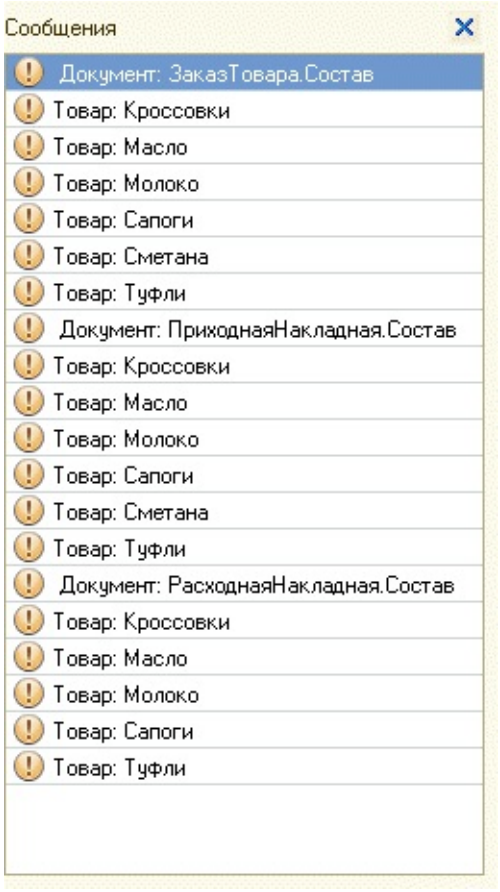


Рис. 2.55. Создание запроса из произвольного источника

В результате для каждой табличной части каждого документа конфигурации формируется свой список товаров. Заметьте, что документы, не имеющие табличных частей (например, документ *Событие* в нашей демонстрационной конфигурации), не попадают в обход.

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

В случае, если нас интересует общий список товаров, независимо от того, в каком документе он используется, то можно убрать из запроса секцию описания итогов. Для того чтобы перечень товаров не повторялся, для объединения запросов нужно использовать конструкцию *ОБЪЕДИНИТЬ*, а не *ОБЪЕДИНИТЬ ВСЕ*. Результат запроса в этом случае нужно обходить линейным способом.

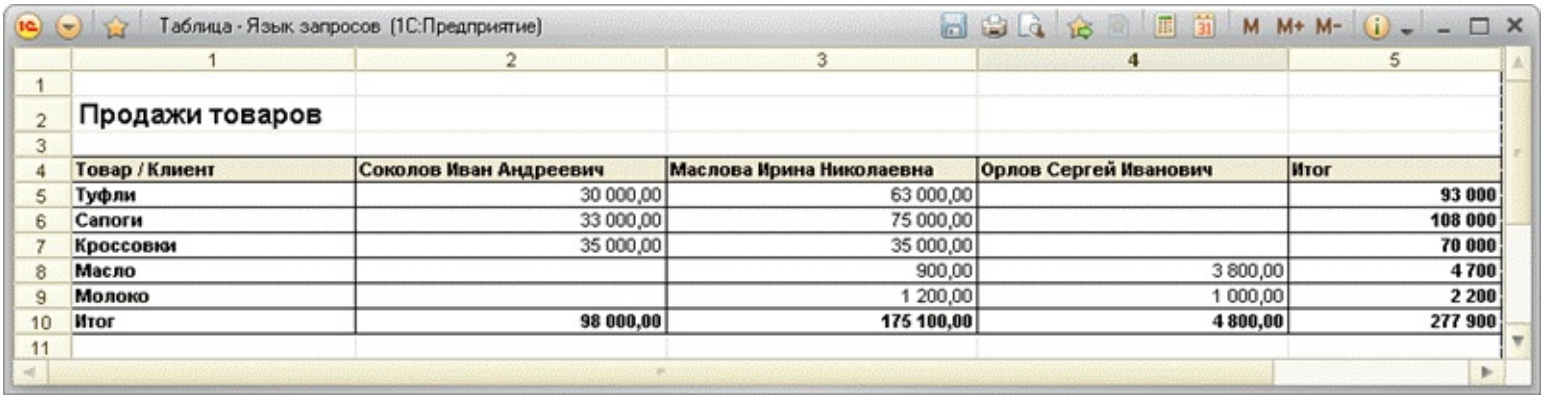
Создание кросс-отчета

Иногда бывает нужно обойти выборку результата отчета с типом обхода *ПоГруппировкам* и при этом получить значения всех группировок в пределах другой группировки. Например, получить итоговые значения продаж по всем клиентам для каждого товара и т. п.

Такая возможность может понадобиться при построении различных кросс-отчетов, представляющих информацию в табличном виде. Хотя кросс-отчеты легче, быстрее и, главное, методически правильнее формировать с помощью системы компоновки данных, в этом разделе мы покажем, как получить данные для кросс-отчета и вывести его в табличный документ с помощью встроенного языка.

Особенность получения данных для кросс-отчета (см. рис. 2.56) состоит в том, что при этом нужно, чтобы запрос внутри родительской группировки (например, по товарам) получил не только те вложенные группировки (например, по клиентам), по которым есть данные (продажи), но и те, по которым данных нет. Например, группировку по клиенту *Орлов* для товара *Туфли* и группировку по клиенту *Соколов* для товара *Молоко*. То есть чтобы в каждой строке таблицы было одинаковое количество столбцов, иначе таблица «поедет». Поэтому при получении вложенных группировок нужно использовать третий параметр *ВСЕ* метода выборки *Выбрать()*.

Рассмотрим пример. Предположим, нам необходимо получить кросс-отчет по продажам товаров различным покупателям. Список товаров необходимо вывести в строках таблицы, а покупателей – в колонках (рис. 2.56).



Продажи товаров

Товар / Клиент	Соколов Иван Андреевич	Маслова Ирина Николаевна	Орлов Сергей Иванович	Итог
Туфли	30 000,00	63 000,00		93 000
Сапоги	33 000,00	75 000,00		108 000
Кроссовки	35 000,00	35 000,00		70 000
Масло		900,00	3 800,00	4 700
Молоко		1 200,00	1 000,00	2 200
Итог	98 000,00	175 100,00	4 800,00	277 900

Рис. 2.56. Кросс-отчет по продажам товаров

Для получения данных о продажах мы будем использовать виртуальную таблицу *Обороты* регистра накопления *Продажи*, при обращении к которой автоматически формируются итоговые данные о продажах в разрезе товаров и покупателей.

подробнее

Подробнее о назначении и использовании виртуальной таблицы оборотов регистра накопления будет рассказано в разделе «Получение оборотов».

Данные для отчета будут получаться при выполнении следующего запроса (листинг 2.45).

Листинг 2.45. Запрос для получения данных

```
Запрос.Текст =
  "ВЫБРАТЬ
  |         ПродажиОбороты.Товар КАК Товар,
  |         ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Товар),
  |         ПродажиОбороты.Клиент КАК Клиент,
  |         ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Клиент),
  |         ПродажиОбороты.ВыручкаОборот КАК ВыручкаОборот
  |ИЗ
  |         РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
  |
  |ИТОГИ
  |         СУММА(ВыручкаОборот)
  |ПО
  |         ОБЩИЕ,
```

```
  |        Товар,
  |        Клиент";
```

Из текста запроса видно, что в результате запроса должны присутствовать общие итоги, а также итоги по полям *Товар* и *Клиент*. Для обхода итоговых группировок мы будем использовать обход выборки из результата запроса с типом обхода *ПоГруппировкам*.

Этот тип обхода мы рассматривали подробно в разделе «Обход по группировкам». Как уже говорилось, тип обхода передается первым параметром в метод выборки *Выбрать()*. Но в данном случае нам понадобится использовать второй и третий параметр этого метода.

Вторым параметром в метод выборки *Выбрать()* передается список группировок, разделенных запятыми, по которым будет производиться обход, а третьим параметром – список группировок, из которых будут выбираться значения группировок для обхода.

Поясним назначение этих параметров на конкретном примере. Для решения поставленной задачи создадим следующую клиентскую процедуру (листинг 2.46).

Листинг 2.46. Вывод результата отчета в табличном документе

```
&НаКлиенте
Процедура ПродажиТоваров(Команда)

  Табдок = Новый ТабличныйДокумент;
  РезультатОтчета = СформироватьОтчет(ТабДок);
  РезультатОтчета.Показать();

КонецПроцедуры
```

В процедуре создается новый табличный документ и передается в серверную внеконтекстную функцию *СформироватьОтчет()*, в которой он заполняется данными. После возвращения на клиента табличный документ показывается пользователю.

Процедура формирования табличного документа выглядит следующим образом (листинг 2.47).

Листинг 2.47. Заполнение данными табличного документа

```
&НаСервереБезКонтекста
Функция СформироватьОтчет(Табдок)

  Макет = Обработки.РаботаСЗапросами.ПолучитьМакет("Макет1");
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |        ПродажиОбороты.Товар КАК Товар,
    |        ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Товар),
    |        ПродажиОбороты.Клиент КАК Клиент,
    |        ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Клиент),
```

```
    |        ПродажиОбороты.ВыручкаОборот КАК ВыручкаОборот
    |ИЗ
    |        РегистрНакопления.Продажи.Обороты КАК ПродажиОбороты
    |ИТОГИ
    |        СУММА(ВыручкаОборот)
    |ПО
    |        ОБЩИЕ,
    |        Товар,
    |        Клиент";

Результат = Запрос.Выполнить();

ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");
ОбластьШапкаТаблицы = Макет.ПолучитьОбласть("ШапкаТаблицы");
ОбластьТовар = Макет.ПолучитьОбласть("Товар");
ОбластьКлиент = Макет.ПолучитьОбласть("Клиент");
ОбластьКлиентШапка = Макет.ПолучитьОбласть("КлиентШапка");
ОбластьЗаголовкаИтога = Макет.ПолучитьОбласть("ЗаголовокИтога");
ОбластьИтогПоКолонке = Макет.ПолучитьОбласть("ИтогПоКолонке");
ОбластьИтогПоСтроке = Макет.ПолучитьОбласть("ИтогПоСтроке");
ОбластьЗаголовокИтогаПоСтроке = Макет.ПолучитьОбласть("ЗаголовокИтогаПоСтроке");
ОбластьОбщийИтог = Макет.ПолучитьОбласть("ОбщийИтог");

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

ВыборкаКлиентИтог = Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам, "Клиент");
Пока ВыборкаКлиентИтог.Следующий() Цикл
  ОбластьКлиентШапка.Параметры.Заполнить(ВыборкаКлиентИтог);
  ТабДок.Присоединить(ОбластьКлиентШапка);

КонецЦикла;

ТабДок.Присоединить(ОбластьЗаголовокИтогаПоСтроке);

ВыборкаТовар = Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам, "Товар");
Пока ВыборкаТовар.Следующий() Цикл
  ОбластьТовар.Параметры.Заполнить(ВыборкаТовар);
  ТабДок.Вывести(ОбластьТовар);

  ВыборкаКлиент = ВыборкаТовар.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам, "Клиент",
  "ВСЕ");
  Пока ВыборкаКлиент.Следующий() Цикл
    ОбластьКлиент.Параметры.Заполнить(ВыборкаКлиент);
    ТабДок.Присоединить(ОбластьКлиент);

  КонецЦикла;

  ОбластьИтогПоСтроке.Параметры.Заполнить(ВыборкаТовар);
  ТабДок.Присоединить(ОбластьИтогПоСтроке);

КонецЦикла;

ТабДок.Вывести(ОбластьЗаголовкаИтога);
ВыборкаКлиентИтог.Сбросить();
Пока ВыборкаКлиентИтог.Следующий() Цикл
  ОбластьИтогПоКолонке.Параметры.Заполнить(ВыборкаКлиентИтог);
  ТабДок.Присоединить(ОбластьИтогПоКолонке);
```

```
    КонецЦикла;

    ВыборкаОбщийИтог = Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
    ВыборкаОбщийИтог.Следующий();
    ОбластьОбщийИтог.Параметры.Заполнить(ВыборкаОбщийИтог);
    ТабДок.Присоединить(ОбластьОбщийИтог);
    ТабДок.Вывести(ОбластьПодвал);

    Возврат ТабДок;

КонецФункции
```

В функции *СформироватьОтчет()* сначала получается макет табличного документа, созданный нами для формы обработки (рис. 2.57).

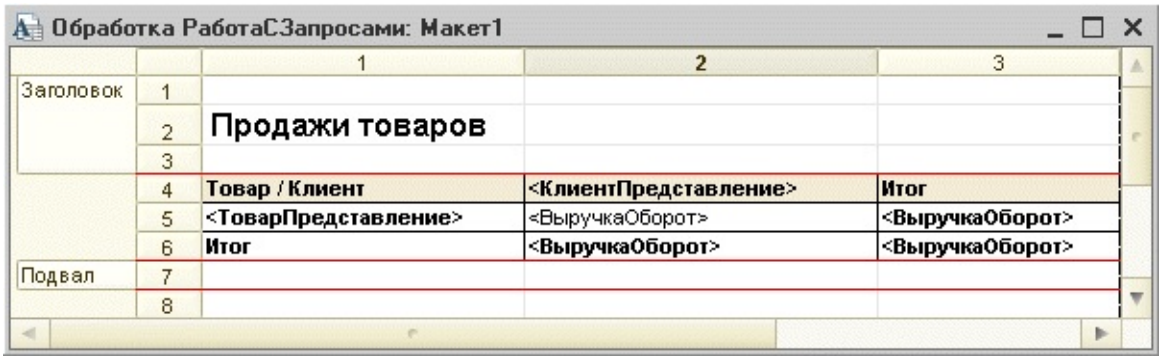


Рис. 2.57. Макет табличного документа

Затем создается и выполняется запрос для получения данных о продажах. Далее получаются области табличного документа, заданные в макете. После этого в табличный документ *ТабДок*, переданный в функцию, методом *Вывести()* выводится область заголовка, полученная из макета.

Далее получаются итоговые данные по клиентам. Для этого из результата запроса получается выборка *ВыборкаКлиентИтог* с типом обхода *ПоГруппировкам* и списком группировок для обхода *"Клиент"* (второй параметр метода *Выбрать()*). В результате обхода этой выборки заполняются наименования клиентов в колонках. Соответствующие области выводятся в результирующий табличный документ методом *Присоединить()*, т. е. слева направо.

После этого получаются итоговые данные по товарам. Для этого из результата запроса получается выборка *ВыборкаТовар* с типом обхода *ПоГруппировкам* и списком группировок для обхода *"Товар"*. В результате обхода этой выборки заполняются наименования товаров в строках. Соответствующие области выводятся в результирующий табличный документ методом *Вывести()*, т. е. сверху вниз.

Внутри цикла обхода этой выборки для каждого товара получаются итоговые значения продаж для каждого клиента. Для этого у текущей записи выборки *ВыборкаТовар* методом *Выбрать()* получается еще одна выборка *ВыборкаКлиент* с типом обхода *ПоГруппировкам*, списком группировок для обхода *"Клиент"* (второй параметр метода) и списком значений группировок для обхода *"ВСЕ"* (третий параметр метода).

В результате обхода этой выборки заполняются собственно значения продаж каждого товара по каждому клиенту, которые находятся в ячейках таблицы, на пересечении строк и колонок. Если продаж данного товара данному клиенту не было, то значения в ячейках будут пусты. Соответствующие области выводятся в результирующий табличный документ методом *Присоединить()*, т. е. слева направо.

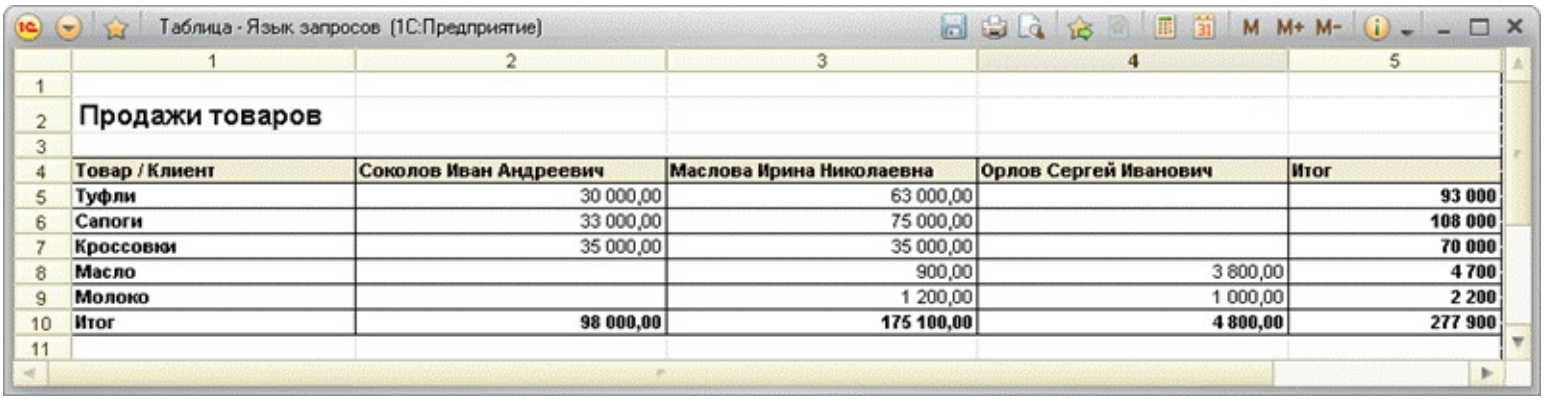
После завершения вложенного цикла для каждого товара из выборки *ВыборкаТовар* заполняется итоговое поле (итог по строке) и присоединяется к результирующему табличному документу в конце строки.

После обхода всех товаров выборка *ВыборкаКлиентИтог* опять позиционируется на начало, в результате обхода этой выборки заполняются итоговые поля для каждого клиента (итог по колонке) и присоединяются к результирующему табличному документу в соответствующих колонках итоговой строки.

Далее получается выборка из результата запроса *ВыборкаОбщийИтог* с типом обхода *ПоГруппировкам*, без второго и третьего параметра метода *Выбрать()*. В результате заполняется общее итоговое значение всего отчета и присоединяются к результирующему табличному документу в нижней правой ячейке таблицы.

В заключение в табличный документ выводится область подвала документа, полученная из макета.

Таким образом, при вызове процедуры *ПродажиТоваров()* создается и показывается пользователю табличный документ, содержащий кросс-отчет о продажах товаров, где товары расположены в строках, а клиенты – в колонках таблицы (рис. 2.58).



	1	2	3	4	5
1					
2	**Продажи товаров**				
3					
4	**Товар / Клиент**	**Соколов Иван Андреевич**	**Маслова Ирина Николаевна**	**Орлов Сергей Иванович**	**Итог**
5	**Туфли**	30 000,00	63 000,00		**93 000**
6	**Сапоги**	33 000,00	75 000,00		**108 000**
7	**Кроссовки**	35 000,00	35 000,00		**70 000**
8	**Масло**		900,00	3 800,00	**4 700**
9	**Молоко**		1 200,00	1 000,00	**2 200**
10	**Итог**	**98 000,00**	**175 100,00**	**4 800,00**	**277 900**
11					

Рис. 2.58. Кросс-отчет по продажам товаров

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Вывод итогов по периодам с заданной периодичностью

Иногда требуется рассчитать и вывести в отчет некоторые итоговые значения в заданном интервале с указанной периодичностью. Например, требуется вывести данные о продажах товаров покупателям за указанный период и при этом дополнить результат запроса итогами с заданной периодичностью (например, на начало каждой недели), вне

зависимости от того, были ли продажи за конкретную неделю или нет.

Для решения поставленной задачи нужно рассчитать итоги по полю типа *Дата* с использованием конструкции языка запросов *ПЕРИОДАМИ*. Данная конструкция указывается в предложении *ИТОГИ* после имени поля, по которому нужно рассчитать итоги. После ключевого слова *ПЕРИОДАМИ* в скобках указывается вид периода (*Секунда*, *Минута*, *Час*, *День*, *Неделя* и т. д.), а также могут быть указаны начальная и конечные даты интересуемого периода. В случае, если начальные и конечные даты не указаны, будут использованы первая и последняя даты, участвующие в результате запроса.

Поясним вышесказанное на конкретном примере. Создадим клиентскую процедуру, в которой создается новый табличный документ, заполняется данными на сервере и, после возвращения на клиента, показывается пользователю (листинг 2.48).

Листинг 2.48. Вывод результата отчета в табличном документе

```
&НаКлиенте
Процедура ПродажиТоваровПоДатам(Команда)

  Табдок = Новый ТабличныйДокумент;
  РезультатОтчета = СформироватьОтчетСДополнениемДат(ТабДок, ДатаНачала, ДатаОкончания);
  РезультатОтчета.Показать();

КонецПроцедуры
```

В серверную внеконтекстную функцию *СформироватьОтчетСДополнениемДат()* помимо самого табличного документа передаются также даты начала и окончания отчетного периода, которые вводятся в полях формы обработки перед формированием табличного документа.

Процедура формирования табличного документа выглядит следующим образом (листинг 2.49).

Листинг 2.49. Заполнение данными табличного документа

```
&НаСервереБезКонтекста
Функция СформироватьОтчетСДополнениемДат(Табдок, ДатаНачала, ДатаОкончания)

  Макет = Обработки.РаботаСЗапросами.ПолучитьМакет("Макет2");
  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |         ПродажиОбороты.Период КАК Период,
    |         ПродажиОбороты.Товар,
    |         ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Товар),
    |         ПродажиОбороты.Клиент,
    |         ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Клиент),
    |         ПродажиОбороты.ВыручкаОборот КАК ВыручкаОборот
    |ИЗ
    |         РегистрНакопления.Продажи.Обороты(, , Неделя, ) КАК ПродажиОбороты
    |
```

```
    |УПОРЯДОЧИТЬ ПО
    |        Период
    |ИТОГИ
    |        СУММА(ВыручкаОборот)
    |ПО
    |        Период ПЕРИОДАМИ(НЕДЕЛЯ, &ДатаНачала, &ДатаОкончания)";

  Запрос.УстановитьПараметр("ДатаНачала", ДатаНачала);
  Запрос.УстановитьПараметр("ДатаОкончания", ДатаОкончания);

  РезультатЗапроса = Запрос.Выполнить();

  ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
  ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");
  ОбластьШапкаТаблицы = Макет.ПолучитьОбласть("ШапкаТаблицы");
  ОбластьПодвалТаблицы = Макет.ПолучитьОбласть("ПодвалТаблицы");
  ОбластьПериод = Макет.ПолучитьОбласть("Период");
  ОбластьДетальныхЗаписей = Макет.ПолучитьОбласть("Детали");

  ТабДок.Очистить();

  ОбластьЗаголовок.Параметры.ДатаНачала = Формат(ДатаНачала,"ДФ=дд.ММ.гггг");
  ОбластьЗаголовок.Параметры.ДатаОкончания = Формат(ДатаОкончания,"ДФ=дд.ММ.гггг");
  ТабДок.Вывести(ОбластьЗаголовок);
  ТабДок.Вывести(ОбластьШапкаТаблицы);
  ТабДок.НачатьАвтогруппировкуСтрок();

  ВыборкаПериод = РезультатЗапроса.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам,
  "Период", "ВСЕ");
  Пока ВыборкаПериод.Следующий() Цикл
    ОбластьПериод.Параметры.Заполнить(ВыборкаПериод);
    ТабДок.Вывести(ОбластьПериод, ВыборкаПериод.Уровень());

    ВыборкаДетальныеЗаписи = ВыборкаПериод.Выбрать();

    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
      ОбластьДетальныхЗаписей.Параметры.Заполнить(ВыборкаДетальныеЗаписи);
      ТабДок.Вывести(ОбластьДетальныхЗаписей, ВыборкаДетальныеЗаписи.Уровень());

    КонецЦикла;

  КонецЦикла;

  ТабДок.ЗакончитьАвтогруппировкуСтрок();
  ТабДок.Вывести(ОбластьПодвалТаблицы);
  ТабДок.Вывести(ОбластьПодвал);

  Возврат ТабДок;

КонецФункции
```

В функции *СформироватьОтчетСДополнениемДат()* получаются и выводятся в табличный документ данные о продажах товаров клиентам, при этом рассчитываются итоги на начало каждой недели за указанный период. То есть результат запроса дополняется итоговыми данными с периодичностью *НЕДЕЛЯ*.

Мы не будем детально комментировать весь текст функции, так как вывод данных в

табличный документ уже рассматривался подробно в разделе «Вывод в табличный документ». Обратите особое внимание на выделенные жирным шрифтом фрагменты.

В качестве источника запроса используется виртуальная таблица *Обороты* регистра накопления *Продажи* с периодичностью *НЕДЕЛЯ*. Это значит, что при обращении к ней будут автоматически формироваться итоговые данные о продажах с заданной периодичностью.

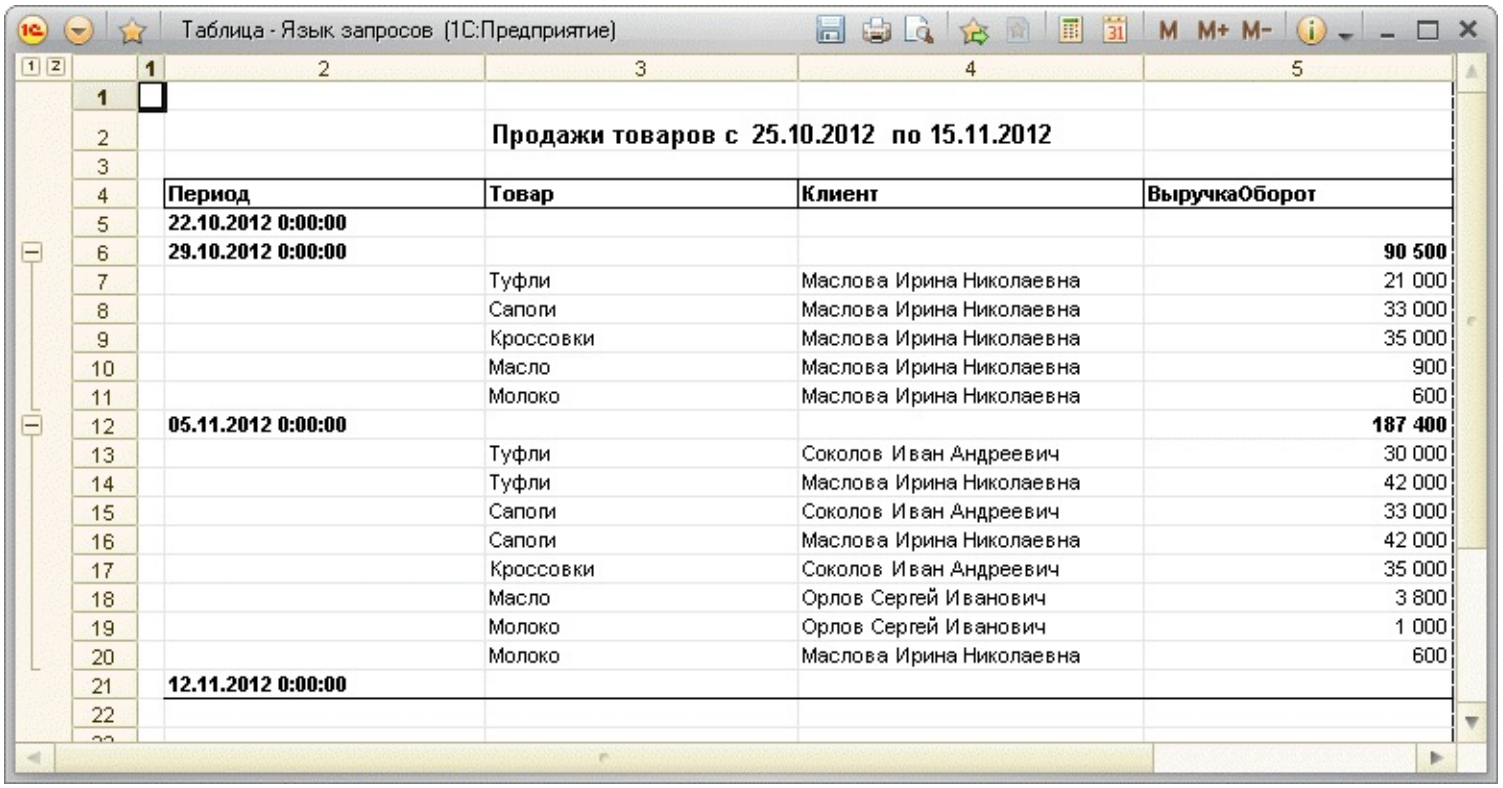
подробнее

Подробнее о назначении и использовании виртуальной таблицы оборотов регистра накопления будет рассказано в разделе «Получение оборотов».

По полю *Период* рассчитываются итоги с недельной периодичностью за требуемый интервал времени. Для этого при описании итогов по полю *Период* используется ключевое слово *ПЕРИОДАМИ*. После него в скобках указывается периодичность – *НЕДЕЛЯ* и период для получения данных, который устанавливается с помощью параметров запроса *&ДатаНачала* и *&ДатаОкончания*, значения которых передаются в функцию.

Затем обратите внимание, что выборка *ВыборкаПериод* получается из результата запроса методом *Выбрать()* с типом обхода *ПоГруппировкам*, списком группировок для обхода *"Период"* (второй параметр метода) и списком значений группировок для обхода *"ВСЕ"* (третий параметр метода). Если при получении выборки третий параметр опустить, то в нее не попадут дополненные периоды, то есть те периоды, за которые не было продаж.

Таким образом, если мы сформируем отчет о продажах товаров за период 25.10.2012 – 15.11.2012, мы получим следующий результат (рис. 2.59).



Продажи товаров с 25.10.2012 по 15.11.2012

Период	Товар	Клиент	ВыручкаОборот
22.10.2012 0:00:00			
29.10.2012 0:00:00			**90 500**
	Туфли	Маслова Ирина Николаевна	21 000
	Сапоги	Маслова Ирина Николаевна	33 000
	Кроссовки	Маслова Ирина Николаевна	35 000
	Масло	Маслова Ирина Николаевна	900
	Молоко	Маслова Ирина Николаевна	600
05.11.2012 0:00:00			**187 400**
	Туфли	Соколов Иван Андреевич	30 000
	Туфли	Маслова Ирина Николаевна	42 000
	Сапоги	Соколов Иван Андреевич	33 000
	Сапоги	Маслова Ирина Николаевна	42 000
	Кроссовки	Соколов Иван Андреевич	35 000
	Масло	Орлов Сергей Иванович	3 800
	Молоко	Орлов Сергей Иванович	1 000
	Молоко	Маслова Ирина Николаевна	600
12.11.2012 0:00:00			

Рис. 2.59. Отчет о продажах товаров за период с дополнением дат с недельной периодичностью

Мы видим, что на начало каждой недели, когда происходили продажи товаров (29 октября и 5 ноября 2012 года) в отчете рассчитаны итоговые суммы продаж. Кроме того, поскольку отчетный период (25.10.2012 – 15.11.2012) включает недели, за которые не было продаж, то на начало каждой такой недели в результат добавляется пустая итоговая строка (22 октября и 12 ноября 2012 года).

Этот пример можно посмотреть в демонстрационной конфигурации «Язык запросов», прилагающейся к книге, в обработке *Работа с запросами*.

Следует иметь в виду, что если в конструкции *ПЕРИОДАМИ()* не указать даты отчетного периода, в пределах которых должно происходить дополнение отчета периодами, то будет выполнено дополнение пустых периодов только внутри результата отчета (если пустые периоды там присутствуют).

Глава 3. Решение прикладных задач

В этой главе речь пойдет о решении типичных задач хранения информации, задач учета движения средств, бухгалтерского учета и задач периодических расчетов, возникающих в процессе хозяйственной деятельности предприятия. Будут рассмотрены таблицы базы данных, в которых хранится информация для учета, и примеры получения данных из них с помощью языка запросов.

Если вы не знакомы или плохо знакомы с назначением и устройством регистров сведений, регистров накопления, регистров бухгалтерии и других объектов конфигурации, с помощью которых решаются прикладные задачи, мы рекомендуем сначала обратиться к документации «1С:Предприятие 8.3. Руководство разработчика»:

- раздел 5.13 «Планы видов характеристик»,
- раздел 5.14 «Регистры»,
- глава 11 «Бухгалтерский учет»,
- глава 12 «Периодические расчеты».

Хранение информации

Прежде чем использоваться для каких-то учетных задач, информация, введенная пользователем в «1С:Предприятие», должна где-то храниться. Для хранения информации предназначены такие объекты конфигурации, как справочники, перечисления, планы видов характеристик, регистры сведений и т. п. В данном разделе мы рассмотрим вопросы получения информации из регистров сведений и планов видов характеристик как наиболее сложных с точки зрения языка запросов источников данных для отчетов.

Все примеры, используемые в данном разделе, можно посмотреть в демонстрационной конфигурации «Хранение информации», которая находится на прилагаемом компакт-диске.

Регистры сведений

Данные каждого регистра сведений хранятся в отдельной таблице информационной базы, в так называемой основной таблице регистра. Эта таблица доступна посредством запросов и имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;

- *<Имя реквизита>* – поле, содержащее значения реквизита регистра с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для регистра как объекта конфигурации;
- *<Имя ресурса>* – поле, содержащее значения ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации;
- *Активность* – имеет тип *Булево*. Содержит признак активности записи и влияния на получение «первых» и «последних» записей регистра. Это поле существует только для регистров с режимом записи *Подчинение регистратору*;
- *Момент времени* – виртуальное поле, не хранится в информационной базе. Содержит объект *МоментВремени* (который включает в себя дату и ссылку на документ-регистратор). Это поле существует только для регистров с режимом записи *Подчинение регистратору*;
- *Период* – дата записи. Определяет положение данной записи на временной оси. Это поле существует только для периодических регистров;
- *Регистратор* – содержит ссылку на документ, которому подчинена данная запись. Это поле существует только для регистров с режимом записи *Подчинение регистратору*;
- *НомерСтроки* – уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*. Это поле существует только для регистров с режимом записи *Подчинение регистратору*.

Получение данных из независимых непериодических регистров сведений

В нашей демонстрационной конфигурации существует непериодический независимый регистр *Сотрудники*, хранящий информацию о должностях, занимаемых сотрудниками, в разрезе физических лиц и подразделений. То есть ресурсом регистра является *Должность*, а измерениями – *Физическое лицо* и *Подразделение* (рис. 3.1).

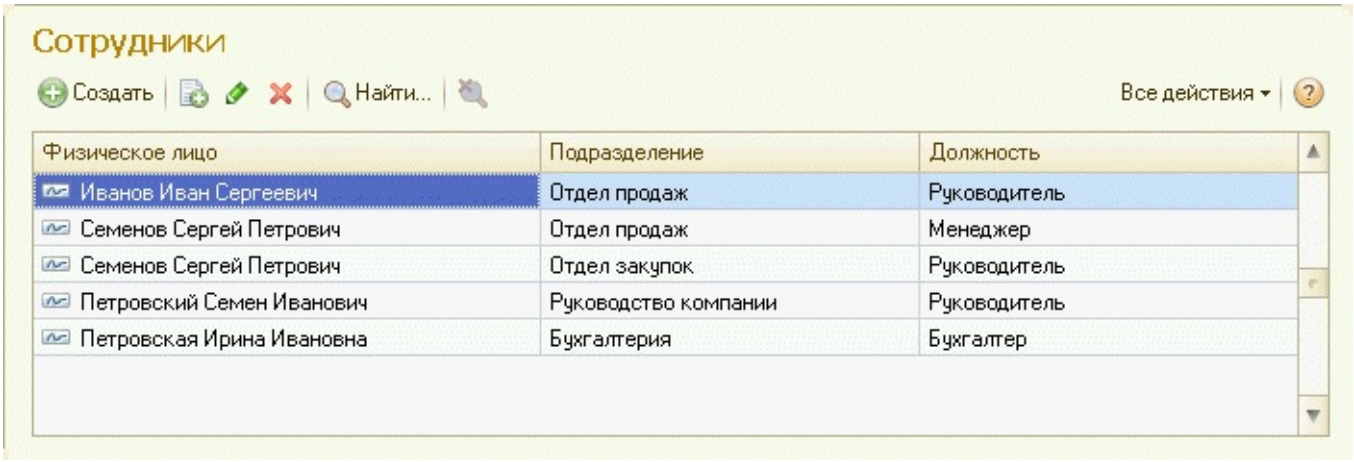


Рис. 3.1. Непериодический независимый регистр сведений

Основная таблица регистра содержит следующие данные (табл. 3.1).

Таблица 3.1. Пример заполнения регистра сведений «Сотрудники»

Физическое лицо	Подразделение	Должность

Иванов Иван Сергеевич	Отдел продаж	Руководитель
Семенов Сергей Петрович	Отдел продаж	Менеджер
Семенов Сергей Петрович	Отдел закупок	Руководитель
Петровский Семен Иванович	Руководство компании	Руководитель
Петровская Ирина Ивановна	Бухгалтерия	Бухгалтер

Это самый простой вид регистра, для которого в информационной базе создается таблица, содержащая набор измерений и ресурсов, определенных в конфигураторе. Получение информации из такого регистра с помощью языка запросов аналогично обращению к обычному справочнику. Например, с помощью следующего запроса можно выбрать из регистра тех сотрудников, в наименовании подразделения которых встречается подстрока «Отдел» (листинг 3.1).

Листинг 3.1. Получение информации из регистра сведений «Сотрудники»

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
  *
ИЗ
  РегистрСведений.Сотрудники КАК Сотрудники
ГДЕ
  Сотрудники.Подразделение.Наименование ПОДОБНО "%Отдел%"
```

Результат выполнения этого запроса в консоли запросов представлен на рис. 3.2.

Запрос: РегистрСведений.Сотрудники (Записей в результате: 3)

ФизическоеЛицо	Подразделение	Должность
Иванов Иван Сергеевич	Отдел продаж	Руководитель
Семенов Сергей Петрович	Отдел продаж	Менеджер
Семенов Сергей Петрович	Отдел закупок	Руководитель

Рис. 3.2. Отбор в регистре сведений

Рассмотрим следующую задачу. Предположим, что из регистра сведений нам нужно получить информацию о внутренних совместителях, то есть вывести данные о сотрудниках, которые занимают несколько должностей или в одном, или в различных подразделениях предприятия. Для этого нужно выполнить следующий запрос (листинг 3.2).

Листинг 3.2. Получение информации из регистра сведений «Сотрудники»

```
// Поместим ссылки на совместителей во временную таблицу
ВЫБРАТЬ
  Сотрудники.ФизическоеЛицо КАК ФизЛицо
ПОМЕСТИТЬ Совместители
ИЗ
  РегистрСведений.Сотрудники КАК Сотрудники
СГРУППИРОВАТЬ ПО
  Сотрудники.ФизическоеЛицо
ИМЕЮЩИЕ
  КОЛИЧЕСТВО(Сотрудники.ФизическоеЛицо) > 1
```

```
;
// Основной запрос к регистру сведений
ВЫБРАТЬ
  Сотрудники.ФизическоеЛицо КАК ФизЛицо,
  Сотрудники.Подразделение КАК Подразделение,
  Сотрудники.Должность КАК Должность
ИЗ
  РегистрСведений.Сотрудники КАК Сотрудники
ГДЕ
  Сотрудники.ФизическоеЛицо В
    (ВЫБРАТЬ
      Совместители.ФизЛицо
    ИЗ
      Совместители КАК Совместители)
```

В данном пакетном запросе сначала ссылки на совместителей помещаются во временную таблицу *Совместители*. Для этого записи регистра сведений группируются по полю *ФизическоеЛицо* и в условии *ИМЕЮЩИЕ* отбираются только группировки, содержащие больше одной записи. Затем ссылки на такие физические лица из временной таблицы подставляются в условие отбора основного запроса.

Результат выполнения этого запроса представлен на рис. 3.3.

Запрос: РегистрСведений.Сотрудники (Записей в результате: 2)

ФизЛицо	Подразделение	Должность
Семенов Сергей Петрович	Отдел продаж	Менеджер
Семенов Сергей Петрович	Отдел закупок	Руководитель

Рис. 3.3. Информация о совместителях

Иногда требуется представить данные регистра сведений в виде дерева с некоторым подчинением. Например, требуется вывести информацию о сотрудниках по подразделениям. Для этого можно сформировать итоги по полю *Подразделение* без расчета самих итоговых полей (листинг 3.3).

Листинг 3.3. Получение информации из регистра сведений «Сотрудники»

```
ВЫБРАТЬ
  Сотрудники.Подразделение КАК Подразделение,
  Сотрудники.ФизическоеЛицо КАК ФизЛицо,
  Сотрудники.Должность КАК Должность
ИЗ
  РегистрСведений.Сотрудники КАК Сотрудники
ИТОГИ ПО
  Сотрудники.Подразделение
```

Результат выполнения этого запроса представлен на рис. 3.4.

Запрос: РегистрСведений.Сотрудники (Записей в результате: 9)

Подразделение	ФизЛицо	Должность
Отдел продаж		
Отдел продаж	Иванов Иван Сергеевич	Руководитель
Отдел продаж	Семенов Сергей Петрович	Менеджер
Отдел закупок		
Отдел закупок	Семенов Сергей Петрович	Руководитель
Руководство компании		
Руководство компании	Петровский Семен Иванович	Руководитель
Бухгалтерия		
Бухгалтерия	Петровская Ирина Ивановна	Бухгалтер

Рис. 3.4. Информация о сотрудниках по подразделениям

Получение данных из периодических регистров сведений

Рассмотрим теперь пример периодического регистра сведений, который хранит методы списания себестоимости в разрезе организаций с заданной периодичностью. Ресурсом регистра *Учетная политика предприятия* является *Метод списания себестоимости*, измерением – *Организация*, периодичность регистра – *В пределах года* (рис. 3.5).

Учетная политика предприятия

Создать | Найти... | Все действия

Период	Организация	Метод списания себестоимости
01.01.2011	ООО "АвтоматикаСвязьПроект"	FIFO
01.01.2011	ООО "Информационные системы"	Средневзвешенный
01.01.2011	ООО "Мультимедиа Плюс"	Средневзвешенный
01.01.2012	ООО "АвтоматикаСвязьПроект"	Средневзвешенный
01.01.2012	ООО "Мультимедиа Плюс"	LIFO
01.01.2013	ООО "Информационные системы"	LIFO

Рис. 3.5. Периодический независимый регистр сведений

В случае указания периодичности регистра (свойство *Периодичность* установлено в значение *В пределах дня*, *В пределах месяца*, *В пределах года* и т. п.) в структуру основной таблицы регистра добавляется стандартное поле *Период*, благодаря которому в регистр сведений добавляется хронологический разрез хранения данных. То есть периодический регистр позволяет хранить данные для одного и того же значения измерения (или комбинации значений измерений) за разные периоды времени.

Например, в регистре *Учетная политика предприятия* хранятся методы списания себестоимости, установленные в конкретной организации на начало года. Таблица, хранящая информацию регистра в информационной базе, содержит следующие данные (табл. 3.2).

Таблица 3.2. Пример заполнения регистра сведений «Учетная политика предприятия»

Период	Организация	Метод списания себестоимости
01.01.2011	ООО "АвтоматикаСвязьПроект"	FIFO

01.01.2011	ООО "Информационные системы"	Средневзвешенный
01.01.2011	ООО "Мультимедиа Плюс"	Средневзвешенный
01.01.2012	ООО "АвтоматикаСвязьПроект"	Средневзвешенный
01.01.2012	ООО "Мультимедиа Плюс"	LIFO
01.01.2013	ООО "Информационные системы"	LIFO

Чтобы получить такие данные, достаточно выполнить следующий запрос (листинг 3.4).

Листинг 3.4. Получение информации из периодического регистра сведений

```
ВЫБРАТЬ
  *
ИЗ
  РегистрСведений.УчетнаяПолитикаПредприятия

УПОРЯДОЧИТЬ ПО
  Период
```

Однако когда записей в регистре много и периодичность регистра более частая, вывод такого списка особого смысла не имеет, так как трудно проследить историю изменения во времени какого-то значения. В таких случаях обычно бывает нужно получить актуальные данные из регистра, то есть срез записей регистра на определенный момент времени.

Для решения таких задач платформа позволяет использовать виртуальные таблицы периодических регистров сведений – *СрезПоследних* и *СрезПервых*. Срез последних записей регистра возвращает по одной, наиболее поздней по времени, для каждого значения измерения (или комбинации значений измерений) записи регистра сведений. Срез первых, наоборот, возвращает наиболее ранние записи регистра для каждого значения измерения.

Как и другие виртуальные таблицы, эти таблицы не хранятся в информационной базе, а формируются на основе данных основной таблицы в момент обращения к ней запросом. Таким образом, разработчик может манипулировать сразу «производными» данными, необходимыми для решения прикладных задач. А тонкости организации оптимального получения этих данных обеспечиваются системой.

Виртуальные таблицы *СрезПоследних* и *СрезПервых* имеют следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования

разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;

- *<Имя реквизита>* – поле, содержащее значения реквизита регистра с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для регистра как объекта конфигурации;
- *<Имя ресурса>* – поле, содержащее значения ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации;
- *Активность* – имеет тип *Булево*. Содержит признак активности записи и влияния на получение «первых» и «последних» записей регистра. Это поле существует только для регистров с режимом записи *Подчинение регистратору*;
- *Период* – дата записи. Определяет положение данной записи на временной оси. Это поле существует только для периодических регистров;
- *Регистратор* – содержит ссылку на документ, которому подчинена данная запись. Это поле существует только для регистров с режимом записи *Подчинение регистратору*;
- *НомерСтроки* – уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*. Это поле существует только для регистров с режимом записи *Подчинение регистратору*;

При построении этих виртуальных таблиц могут использоваться параметры, с помощью которых настраивается или уточняется состав получаемых данных:

- *Период* – имеет тип *Дата*, *МоментВремени* или *Граница*. Указывает период, на который должен быть получен срез. Если параметр не задан, будут выбираться наиболее поздние/ранние записи, то есть без ограничения по времени;
- *Условие* – содержит конструкцию языка запросов – условие. Строится по любым полям регистра сведений (или подчиненным им полям). Используется для ограничения состава исходных записей, по которым при построении виртуальной таблицы будет выполняться срез. То есть условие будет применяться к исходным записям, а не к уже отобранным. Если параметр не указан, будут использоваться все активные записи регистра.

Рассмотрим все вышесказанное на примере. Для получения среза первых записей регистра *Учетная политика предприятия* можно использовать следующий запрос (листинг 3.5).

Листинг 3.5. Получение среза первых записей периодического регистра сведений

```
ВЫБРАТЬ
  *
ИЗ
  РегистрСведений.УчетнаяПолитикаПредприятия.СрезПервых()

УПОРЯДОЧИТЬ ПО
  Период
```

Срез последних записей регистра *Учетная политика предприятия* можно получить с помощью следующего запроса (листинг 3.6).

Листинг 3.6. Получение среза последних записей периодического регистра сведений

```
ВЫБРАТЬ
  *
ИЗ
  РегистрСведений.УчетнаяПолитикаПредприятия.СрезПоследних()

УПОРЯДОЧИТЬ ПО
  Период
```

На рис. 3.6 показано, как формируются данные виртуальных таблиц на основе информации, хранящейся в основной таблице регистра сведений.

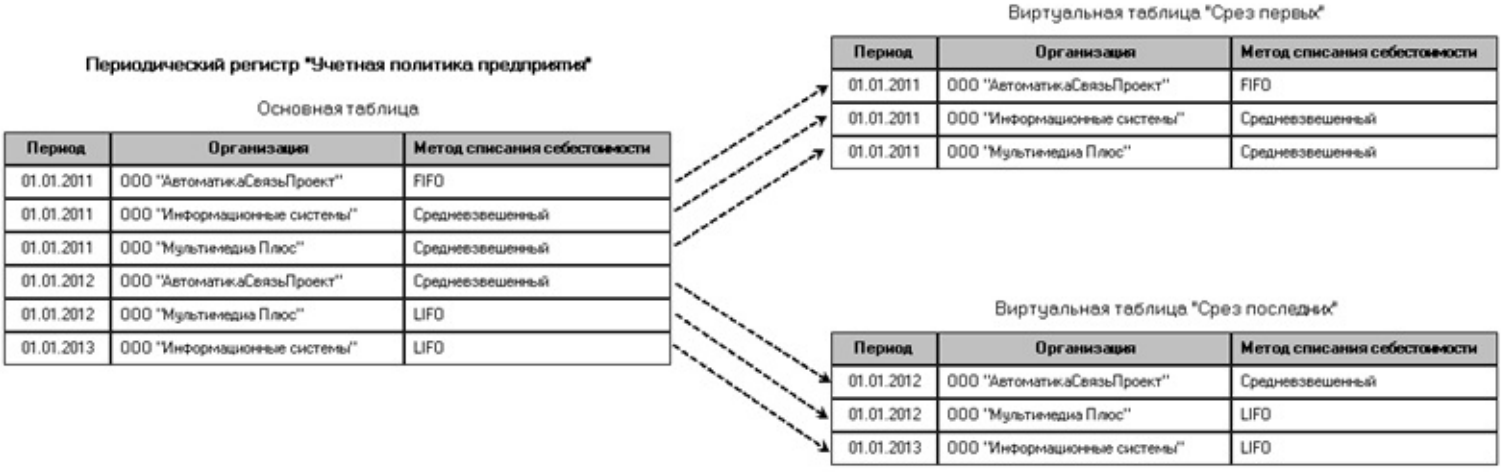


Рис. 3.6. Срез первых и последних записей регистра сведений

На рисунке слева показана основная таблица регистра сведений, из которой видно, что для трех разных организаций установлены методы списания себестоимости на 2011 год. Эти три записи попадут в виртуальную таблицу *Срез первых*, так как это самые ранние записи в регистре для каждой организации. Затем для двух организаций изменены методы списания себестоимости в 2012 году, а для третьей организации – в 2013 году. Эти три записи попадут в виртуальную таблицу *Срез последних*, так как это самые поздние записи в регистре для каждой организации.

В предыдущих двух случаях мы получали срез первых и последних записей регистра без указания периода, на который должен быть получен срез. Но если мы зададим дату, на которую нужно выполнить срез последних записей регистра, как 10.05.2012 (листинг 3.7), мы получим следующий результат (рис. 3.7).

Листинг 3.7. Получение среза последних записей регистра сведений на 10.05.2012

```
ВЫБРАТЬ
  *
ИЗ
  РегистрСведений.УчетнаяПолитикаПредприятия.СрезПоследних(ДАТАВРЕМЯ(2012, 05, 10))
```

```
УПОРЯДОЧИТЬ ПО
  Период
```

Рис. 3.7. Срез последних записей регистра сведений на заданную дату

На рисунке видно, что из основной таблицы регистра в виртуальную таблицу *Срез последних* попадут две записи на начало 2012 года. А для организации, для которой методы списания себестоимости изменены в 2011 и в 2013 году, в срез последних попадет запись за 2011 год (она же входит и в срез первых), так как запись за 2013 год не удовлетворяет условию отбора по периоду.

Рассмотрим еще один периодический регистр сведений *Персонифицированный прайс-лист*, который хранит цены номенклатуры в разрезе товаров и покупателей с заданной периодичностью. Ресурсом регистра является *Цена*, измерениями – *Номенклатура* и *Покупатель*, периодичность регистра – *В пределах дня* (рис. 3.8).

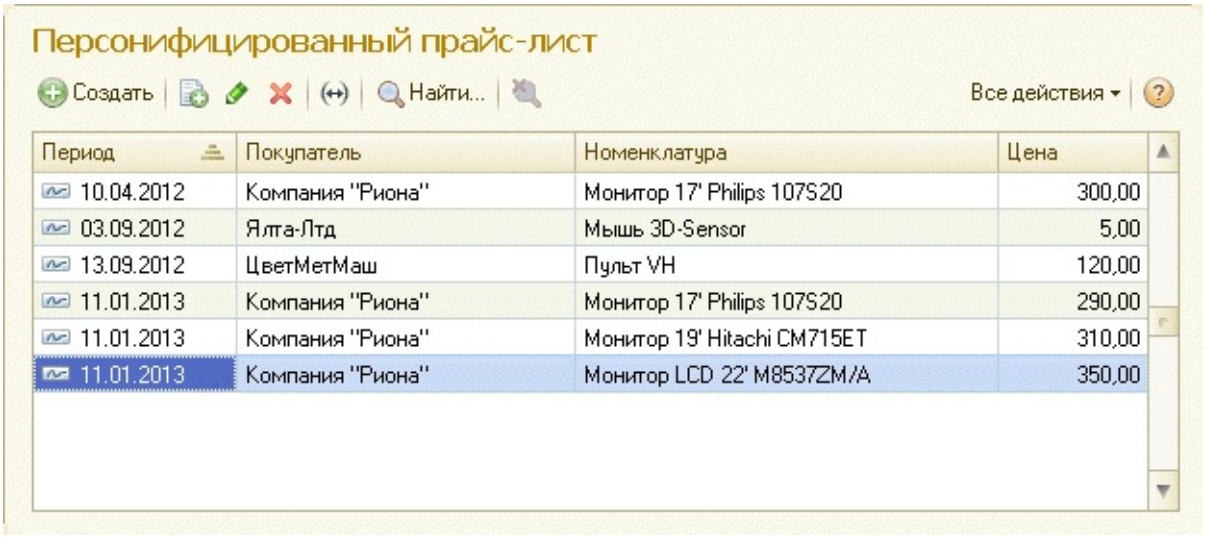


Рис. 3.8. Периодический независимый регистр сведений

На примере этих данных мы рассмотрим одну тонкость, которую необходимо учитывать при установке отбора в виртуальных таблицах.

Срез актуальных цен номенклатуры по покупателям можно получить с помощью следующего запроса (листинг 3.8).

Листинг 3.8. Получение среза последних записей периодического регистра сведений

```
ВЫБРАТЬ
```

```
  *
ИЗ
  РегистрСведений.ПерсонифицированныйПрайс.СрезПоследних()
```

Результат выполнения этого запроса представлен на рис. 3.9.

Запрос: РегистрСведений.ПерсонифицированныйПрайс.СрезПоследних() (Записей в результате: 5)

Период	Покупатель	Номенклатура	Цена
03.09.2012 0:00:00	Ялта-Лтд	Мышь 3D-Sensor	5
11.01.2013 0:00:00	Компания "Риона"	Монитор 19' Hitachi CM715ET	310
11.01.2013 0:00:00	Компания "Риона"	Монитор LCD 22' M8537ZM/A	350
11.01.2013 0:00:00	Компания "Риона"	Монитор 17' Philips 107S20	290
13.09.2012 0:00:00	ЦветМетМаш	Пульт VH	120

Рис. 3.9. Срез последних записей регистра сведений

Как мы видим, в результат запроса попали пять записей из шести, так как для покупателя *Компания «Риона»* на номенклатуру *Монитор 17' Philips 107S20* цена устанавливалась дважды, и в срезе последних мы видим последнюю цену – 290, установленную 11.01.2013.

Предположим, нам нужно получить срез последних цен номенклатуры, которые были установлены больше или равными 300. Для этого потребуется наложить отбор на записи виртуальной таблицы среза последних. Это можно сделать двумя способами: в условии отбора *ГДЕ* языка запросов или в параметре виртуальной таблицы. В общем случае для увеличения скорости выполнения запросов рекомендуется использовать второй вариант. Этот вопрос будет подробно рассмотрен в разделе «Использовать параметры виртуальных таблиц».

Но в данном случае при использовании разных способов отбора может быть получен различный результат. В первом случае будут получены все записи среза последних, и затем уже к ним будет применен отбор. Во втором случае уже на этапе формирования виртуальной таблицы будут отсечены записи основной таблицы регистра сведений, которые не удовлетворяют условию отбора, и затем из оставшихся записей будет получен срез последних. Поэтому в данном случае нужно учитывать логику исполнения запроса, в зависимости от решаемой задачи.

Проверим вышесказанное на примере. Наложим отбор на записи среза последних с помощью предложения *ГДЕ* языка запросов (листинг 3.9).

Листинг 3.9. Отбор записей виртуальной таблицы среза последних периодического регистра сведений

```
ВЫБРАТЬ
  *
ИЗ
  РегистрСведений.ПерсонифицированныйПрайс.СрезПоследних()
ГДЕ
  Цена >= 300
```

При выполнении этого запроса сначала из основной таблицы регистра сведений (см. рис.

3.8) получается срез последних записей (см. рис. 3.9), а затем на эти записи накладывается отбор (*Цена >= 300*). Таким образом мы получаем тех покупателей, для которых последние цены на номенклатуру установлены больше или равными 300. В результате из пяти записей отбираются две (рис. 3.10).

Запрос: РегистрСведений.ПерсонифицированныйПрайс.СрезПоследних() (Записей в результате: 2)

Период	Покупатель	Номенклатура	Цена
11.01.2013 0:00:00	Компания "Риона"	Монитор 19' Hitachi CM715ET	310
11.01.2013 0:00:00	Компания "Риона"	Монитор LCD 22' M8537ZM/A	350

Рис. 3.10. Отбор среза последних записей регистра сведений

Теперь передадим условие отбора во втором параметре виртуальной таблицы *Условие* (листинг 3.10).

Листинг 3.10. Отбор записей виртуальной таблицы среза последних периодического регистра сведений

```
ВЫБРАТЬ
  *
ИЗ
  РегистрСведений.ПерсонифицированныйПрайс.СрезПоследних(, Цена >= 300)
```

При выполнении этого запроса из основной таблицы регистра сведений (см. рис. 3.8) отбираются записи, удовлетворяющие условию отбора (*Цена >= 300*), а затем из них получается срез последних записей. Таким образом, мы получаем тех покупателей, для которых цены на номенклатуру хотя бы раз устанавливались больше или равными 300 (рис. 3.11).

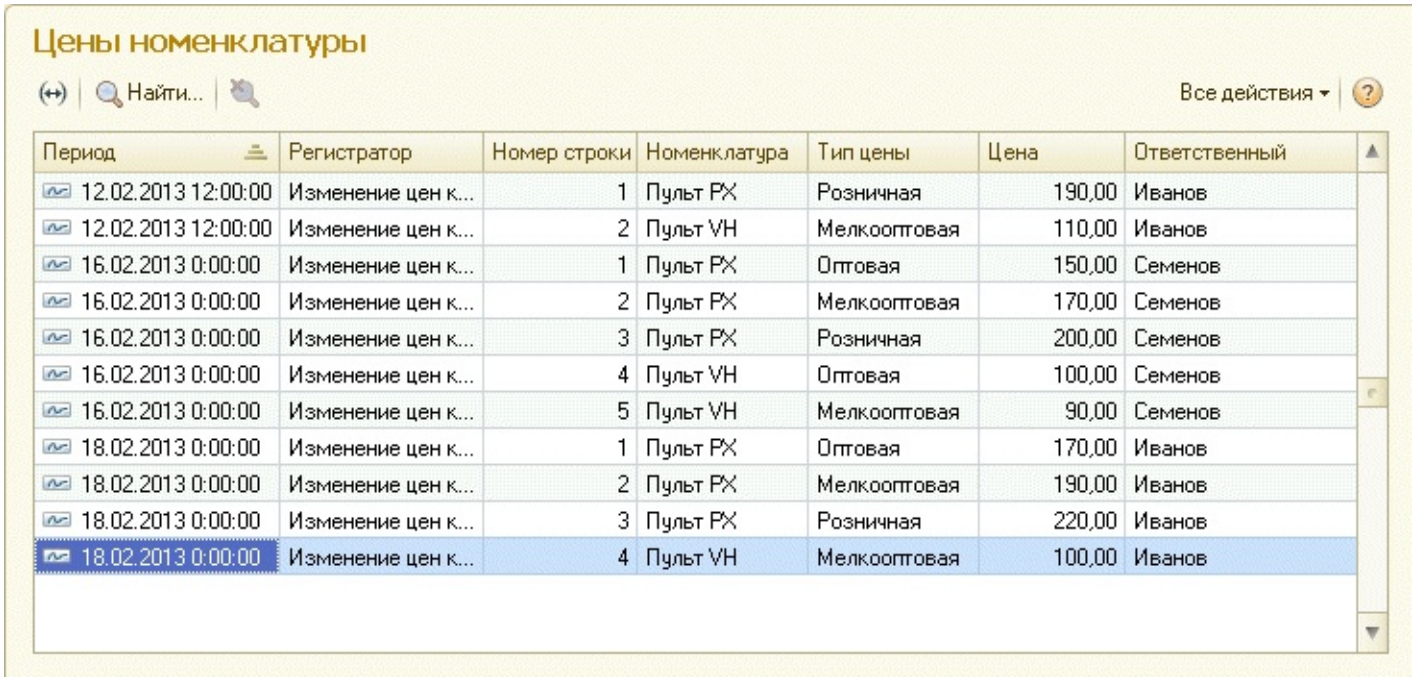
Запрос: РегистрСведений.ПерсонифицированныйПрайс.СрезПоследних(, (Записей в результате: 3)

Период	Покупатель	Номенклатура	Цена
11.01.2013 0:00:00	Компания "Риона"	Монитор 19' Hitachi CM715ET	310
11.01.2013 0:00:00	Компания "Риона"	Монитор LCD 22' M8537ZM/A	350
10.04.2012 0:00:00	Компания "Риона"	Монитор 17' Philips 107S20	300

Рис. 3.11. Отбор среза последних записей регистра сведений

Получение данных из регистров сведений, подчиненных регистратору

Рассмотрим теперь пример периодического зависимого регистра сведений *Цены номенклатуры*, подчиненного регистратору – документу *Изменение цен компании*. Ресурсом регистра является *Цена*, измерениями – *Номенклатура* и *Тип Цены*, периодичность регистра – *По позиции регистратора*. Кроме того, регистр имеет реквизит *Ответственный*, в котором хранится дополнительная информация о записях регистра (рис. 3.12).



Цены номенклатуры

Найти... | Все действия

Период	Регистратор	Номер строки	Номенклатура	Тип цены	Цена	Ответственный
12.02.2013 12:00:00	Изменение цен к...	1	Пульт PX	Розничная	190,00	Иванов
12.02.2013 12:00:00	Изменение цен к...	2	Пульт VH	Мелкооптовая	110,00	Иванов
16.02.2013 0:00:00	Изменение цен к...	1	Пульт PX	Оптовая	150,00	Семенов
16.02.2013 0:00:00	Изменение цен к...	2	Пульт PX	Мелкооптовая	170,00	Семенов
16.02.2013 0:00:00	Изменение цен к...	3	Пульт PX	Розничная	200,00	Семенов
16.02.2013 0:00:00	Изменение цен к...	4	Пульт VH	Оптовая	100,00	Семенов
16.02.2013 0:00:00	Изменение цен к...	5	Пульт VH	Мелкооптовая	90,00	Семенов
18.02.2013 0:00:00	Изменение цен к...	1	Пульт PX	Оптовая	170,00	Иванов
18.02.2013 0:00:00	Изменение цен к...	2	Пульт PX	Мелкооптовая	190,00	Иванов
18.02.2013 0:00:00	Изменение цен к...	3	Пульт PX	Розничная	220,00	Иванов
18.02.2013 0:00:00	Изменение цен к...	4	Пульт VH	Мелкооптовая	100,00	Иванов

Рис. 3.12. Регистр сведений, подчиненный регистратору

В случае указания зависимости регистра от регистратора (свойство *Режим записи* установлено в значение *Подчинение регистратору*) в структуру основной таблицы регистра добавляются стандартные поля: *Регистратор*, *НомерСтроки* и *Активность* (это поле по умолчанию не выводится в форме списка). В этих полях указывается ссылка на документ-регистратор, порядковый номер записи в наборе записей регистра, подчиненного регистратору, и признак активности этих записей в регистре.

Если регистр подчинен регистратору, то может быть задана периодичность регистра *По позиции регистратора* (как в нашем случае), но в общем случае периодичность регистра может быть другой либо отсутствовать вовсе.

В нашей демонстрационной конфигурации при проведении документа *Изменение цен компании* в регистр *Цены номенклатуры* добавляется набор записей, связанных по полю *Регистратор* с этим документом. Исходя из документа-регистратора, заполняется поле *Период* как дата документа, а также измерения, ресурс и реквизит регистра. Таким образом, в регистре хранится информация о ценах номенклатуры в разрезе товаров и типов цены с привязкой к документу, который установил эти цены, а также дополнительная информация о сотруднике, который за это изменение цен ответственен.

Исходя из описанной выше функциональности регистра, с помощью языка запросов можно получить цены номенклатуры, установленные определенным документом (листинг 3.11).

Листинг 3.11. Отбор записей подчиненного регистра сведений по регистратору

```
ВЫБРАТЬ
  ЦеныНоменклатуры.Период,
  ЦеныНоменклатуры.НомерСтроки,
  ЦеныНоменклатуры.Номенклатура,
  ЦеныНоменклатуры.ТипЦены,
  ЦеныНоменклатуры.Цена,
```

```
  ЦеныНоменклатуры.Ответственный
ИЗ
  РегистрСведений.ЦеныНоменклатуры КАК ЦеныНоменклатуры
ГДЕ
  ЦеныНоменклатуры.Регистратор = &РегистраторОтбора
```

Результат выполнения этого запроса представлен на рис. 3.13.

Запрос: РегистрСведений.ЦеныНоменклатуры (Записей в результате: 5)

Период	НомерСтроки	Номенклатура	ТипЦены	Цена	Ответственный
16.02.2013 0:00:00	1	Пульт PX	Оптовая	150	Семенов
16.02.2013 0:00:00	2	Пульт PX	Мелкооптовая	170	Семенов
16.02.2013 0:00:00	3	Пульт PX	Розничная	200	Семенов
16.02.2013 0:00:00	4	Пульт VH	Оптовая	100	Семенов
16.02.2013 0:00:00	5	Пульт VH	Мелкооптовая	90	Семенов

Рис. 3.13. Регистр сведений, подчиненный регистратору

Поскольку регистр периодический, можно получить срез последних цен номенклатуры, установленных определенным сотрудником (листинг 3.12).

Листинг 3.12. Отбор записей виртуальной таблицы среза последних по условию

```
ВЫБРАТЬ
  ЦеныНоменклатурыСрезПоследних.Период,
  ЦеныНоменклатурыСрезПоследних.Номенклатура,
  ЦеныНоменклатурыСрезПоследних.ТипЦены,
  ЦеныНоменклатурыСрезПоследних.Цена,
  ЦеныНоменклатурыСрезПоследних.Ответственный
ИЗ
  РегистрСведений.ЦеныНоменклатуры.СрезПоследних(, Ответственный = &Ответственный)
    КАК ЦеныНоменклатурыСрезПоследних
```

Результат выполнения этого запроса представлен на рис. 3.14.

Запрос: РегистрСведений.ЦеныНоменклатуры.СрезПоследних(, (Записей в результате: 4)

Период	Номенклатура	ТипЦены	Цена	Ответственный
18.02.2013 0:00:00	Пульт PX	Мелкооптовая	190	Иванов
18.02.2013 0:00:00	Пульт PX	Оптовая	170	Иванов
18.02.2013 0:00:00	Пульт PX	Розничная	220	Иванов
18.02.2013 0:00:00	Пульт VH	Мелкооптовая	100	Иванов

Рис. 3.14. Регистр сведений, подчиненный регистратору

В целом из регистров сведений, подчиненных регистратору, можно получить информацию таким же образом, как и из независимых непериодических (раздел "Получение данных из независимых непериодических регистров сведений") и независимых периодических регистров (раздел "Получение данных из периодических регистров сведений").

Планы видов характеристик

В данном разделе мы рассмотрим примеры использования планов видов характеристик для решения различных прикладных задач с помощью языка запросов.

Данные каждого плана видов характеристик хранятся в отдельной таблице

информационной базы, доступной посредством запросов. Эта таблица имеет следующий состав полей:

- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный план видов характеристик;
- *<Имя реквизита>* – поле, содержащее значения реквизита вида характеристики с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для плана видов характеристик как объекта конфигурации;
- *<Имя табличной части>* – поле, содержащее табличные части вида характеристики. Имена полей соответствуют именам табличных частей плана видов характеристик, как они заданы в конфигураторе. Имеет тип *РезультатЗапроса*. Результат запроса к табличной части состоит из колонки *НомерСтроки* и колонок с именами, соответствующими именам реквизитов табличной части;
- *Код* – имеет тип *Строка*. Содержит код вида характеристики;
- *Наименование* – имеет тип *Строка*. Содержит наименование вида характеристики;
- *ПометкаНаУдаление* – имеет тип *Булево*. Содержит признак пометки на удаление вида характеристики;
- *Предопределенный* – имеет тип *Булево*. Содержит признак того, что данный вид характеристики определен в метаданных и над ним нельзя производить некоторые операции;
- *Представление* – виртуальное поле, не хранится в информационной базе. Содержит представление вида характеристики;
- *Родитель* – содержит ссылку на родителя вида характеристики. Существует только для многоуровневых планов видов характеристик;
- *Ссылка* – содержит ссылку на вид характеристики;
- *ТипЗначения* – имеет тип *ОписаниеТипов*. Содержит тип значения характеристик данного вида. Является полем неограниченной длины;
- *ЭтоГруппа* – имеет тип *Булево*. Содержит признак того, является ли данная характеристика группой. Существует только для многоуровневых планов видов характеристик.

Как мы видим, по составу полей таблица плана видов характеристик очень похожа на таблицу справочника, за исключением поля *ТипЗначения*, в котором хранится описание типа значения характеристик конкретного вида.

Рассмотрим некоторые примеры получения данных из плана видов характеристик.

В нашей демонстрационной конфигурации существует план видов характеристик *Виды характеристик*, который хранит информацию о дополнительных характеристиках номенклатуры (рис. 3.15).

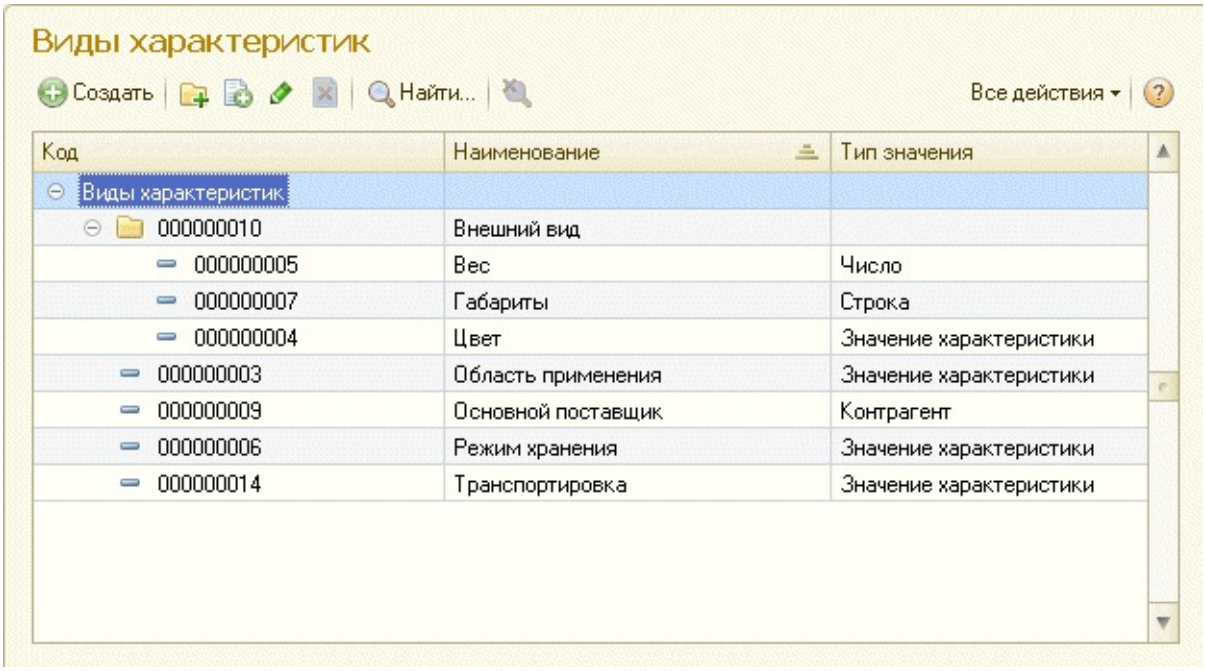


Рис. 3.15. План видов характеристик

План характеристик, также как и справочник, может быть иерархическим, как, например, в нашем случае. На рис. 3.15 мы видим, что виды характеристик, касающиеся внешнего вида товаров, собраны в группу *Внешний вид*.

Пример 1

В случае, если план видов характеристик иерархический, иногда требуется вывести список характеристик в виде дерева. Для этого можно сформировать итоги по полю *Ссылка* без расчета самих итоговых полей (листинг 3.13).

Листинг 3.13. Вывод списка видов характеристик в виде дерева

```
ВЫБРАТЬ
  ВидыХарактеристик.Ссылка КАК ВидХарактеристики
ИЗ
  ПланВидовХарактеристик.ВидыХарактеристик КАК ВидыХарактеристик
ГДЕ
  НЕ ВидыХарактеристик.ЭтоГруппа

ИТОГИ ПО
  ВидХарактеристики ТОЛЬКО ИЕРАРХИЯ
```

Условие отбора в запросе задано для того, чтобы сами группы видов характеристик не попадали в список два раза.

Результат выполнения этого запроса представлен на рис. 3.16.

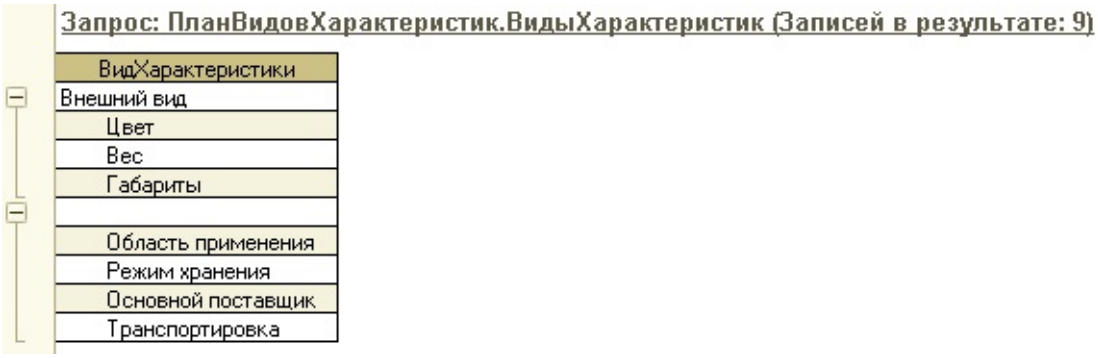


Рис. 3.16. Иерархический план видов характеристик в виде дерева

Пример 2

Рассмотрим следующую задачу. Иногда требуется выбрать виды характеристик, в типе значения которых содержится требуемый тип данных.

Для нашего плана видов характеристик *Виды характеристик* определен составной тип данных значений характеристик, содержащий ссылки на справочники *Значения характеристик*, *Контрагенты*, а также примитивные типы данных (рис. 3.17).

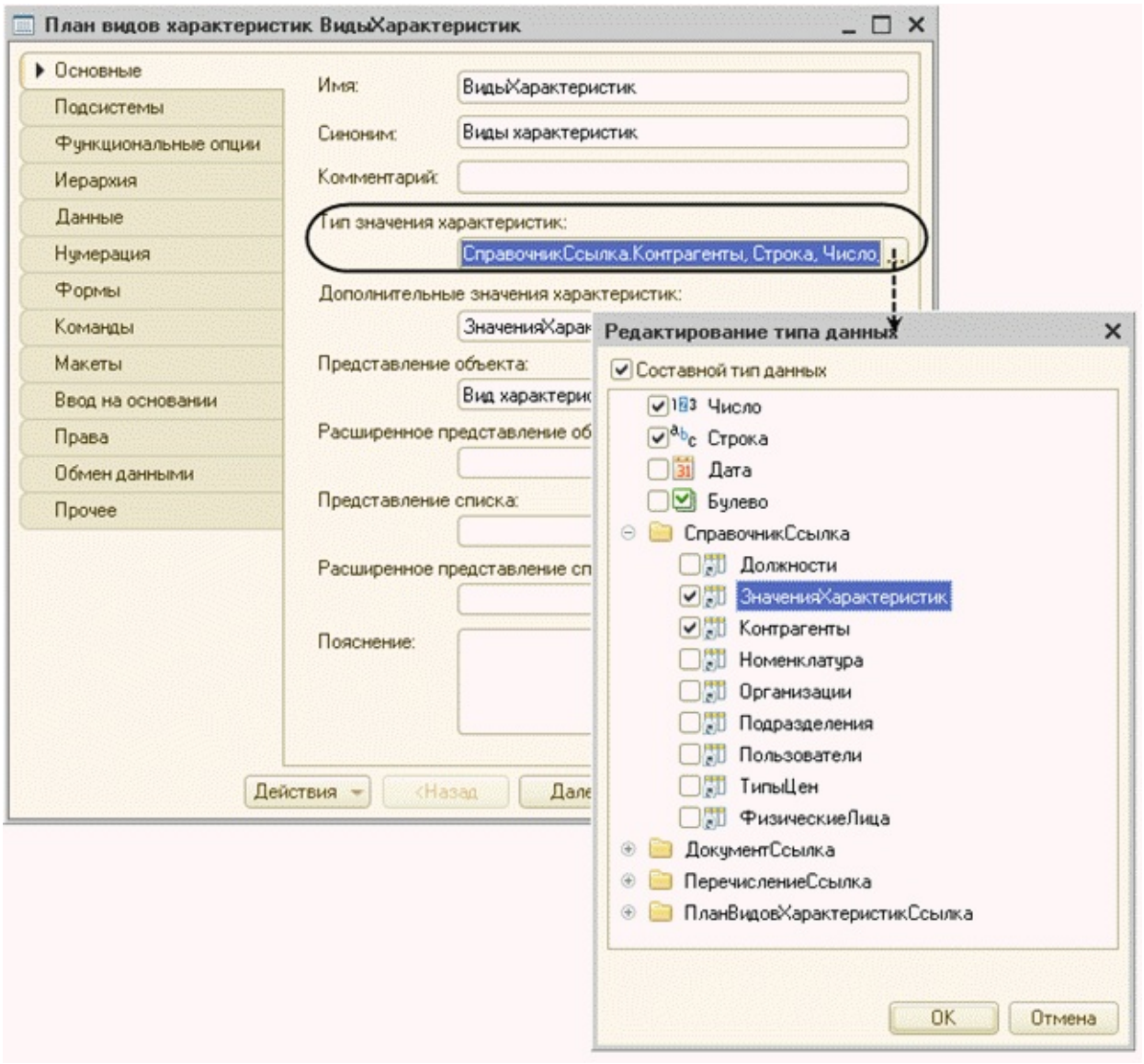


Рис. 3.17. Тип значения характеристик

Справочник *Значения характеристик* был добавлен в прикладное решение для того,

чтобы пользователь мог вводить новые значения дополнительных характеристик объектов, не обращаясь к разработчику (рис. 3.18).

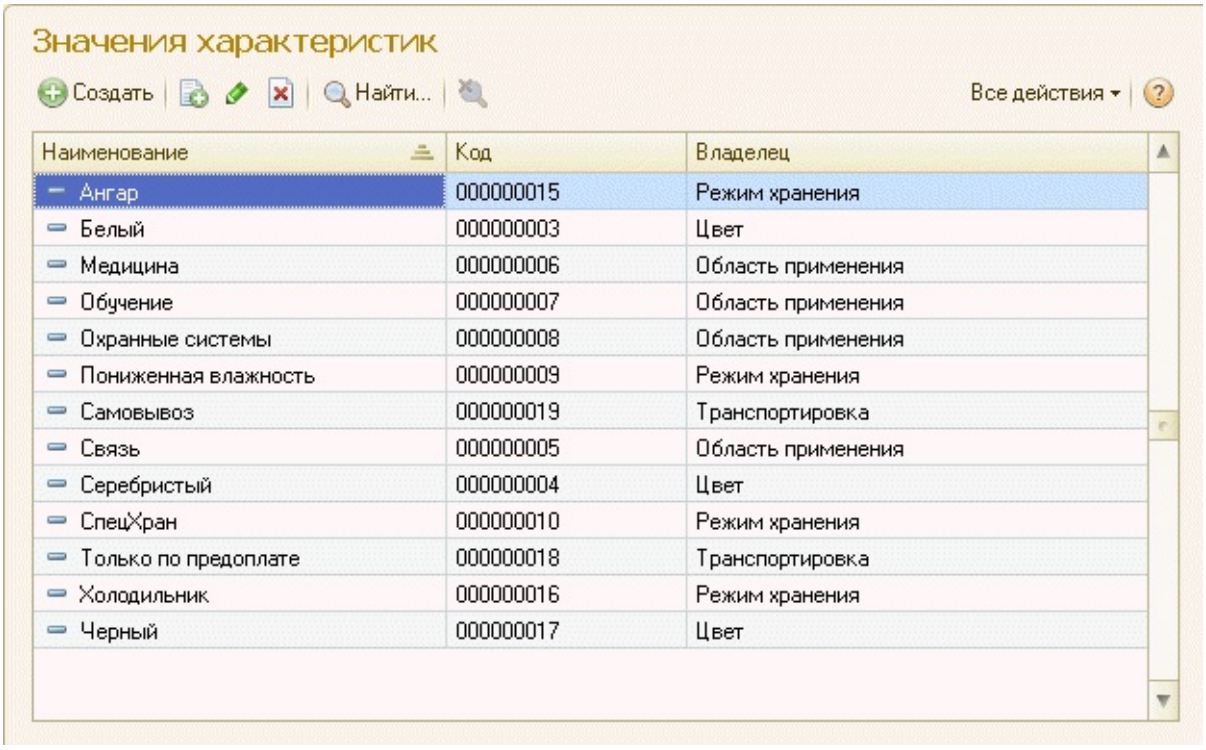


Рис. 3.18. Справочник «Значения характеристик»

Предположим, нам нужно вывести список видов характеристик, в типе значений которых содержится ссылочный тип данных на справочник *Значения характеристик*.

Описание типа значения характеристик конкретного вида хранится в поле *ТипЗначения* (типа *ОписаниеТипов*) таблицы плана видов характеристик. Чтобы решить поставленную задачу, получим значение этого поля запросом и воспользуемся возможностями встроенного языка для анализа содержащихся в нем типов значений (листинг 3.14).

Листинг 3.14. Вывод видов характеристик с заданным типом значения

```
&НаСервереБезКонтекста
Процедура ПолучитьТипыЗначенийХарактеристики()

  Запрос = Новый Запрос;
  Запрос.Текст = "ВЫБРАТЬ
    |        ВидыХарактеристик.Наименование КАК Наименование,
    |        ВидыХарактеристик.ТипЗначения
    |ИЗ
    |        ПланВидовХарактеристик.ВидыХарактеристик КАК ВидыХарактеристик
    |ГДЕ
    |        НЕ ВидыХарактеристик.ЭтоГруппа
    |
    |УПОРЯДОЧИТЬ ПО
    |        Наименование";

  Результат = Запрос.Выполнить();
  Выборка = Результат.Выбрать();

  Сообщение = Новый СообщениеПользователю;
  Пока Выборка.Следующий() Цикл
```

```
  Если Выборка.ТипЗначения.СодержитТип(Тип("СправочникСсылка.ЗначенияХарактеристик"))
  Тогда
    Сообщение.Текст = Выборка.Наименование + ", тип значения: " + Выборка.ТипЗначения;
    Сообщение.Сообщить();

  КонецЕсли;

  КонецЦикла;

КонецПроцедуры
```

В процедуре выполняется запрос, получающий все записи из плана видов характеристик, кроме групп. В цикле обхода выборки из результата запроса анализируется поле выборки *ТипЗначения*. Поскольку поле имеет тип *ОписаниеТипов*, то выполняется метод этого объекта *СодержитТип()*. Если метод возвращает истину, то запрашиваемый тип (*Тип("СправочникСсылка.ЗначенияХарактеристик")*) содержится в типе значений характеристики.

В результате в окно сообщений выводятся виды характеристик, в типе значений которых содержится ссылочный тип данных на справочник *Значения характеристик* (рис. 3.19).

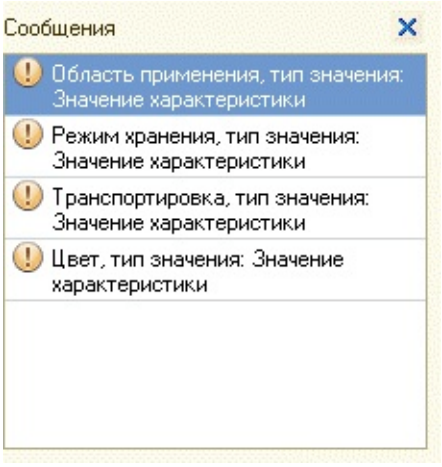


Рис. 3.19. Вывод видов характеристик с типом значения «Значения характеристик»

Пример 3

Как уже говорилось, справочник *Значения характеристик* позволяет пользователям самостоятельно добавлять значения дополнительных характеристик объектов. Чтобы значения характеристик для разных видов не смешивались в диалогах выбора, справочник *Значения характеристик* подчинен плану видов характеристик *Виды характеристик*.

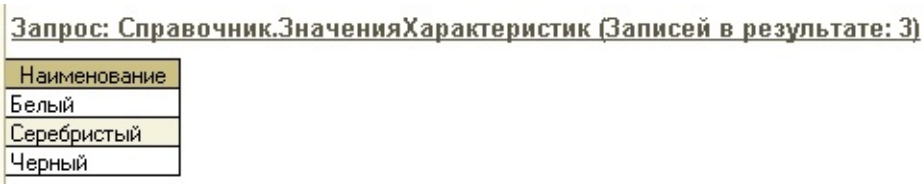
Таким образом, при помощи отбора по полю *Владелец* из справочника *Значения характеристик* можно получить перечень значений для заданного вида характеристики, указанного в параметре *ВидХарактеристики* (листинг 3.15).

Листинг 3.15. Вывод возможных значений заданной характеристики

```
ВЫБРАТЬ
  ЗначенияХарактеристик.Наименование
```

```
ИЗ
  Справочник.ЗначенияХарактеристик КАК ЗначенияХарактеристик
ГДЕ
  ЗначенияХарактеристик.Владелец = &ВидХарактеристики
```

Если мы выберем в качестве значения параметра вид характеристики *Цвет* и выполним запрос в консоли запросов, то мы увидим следующий результат (рис. 3.20).


Запрос: Справочник.ЗначенияХарактеристик (Записей в результате: 3)

Наименование
Белый
Серебристый
Черный

Рис. 3.20. Значения вида характеристики «Цвет»

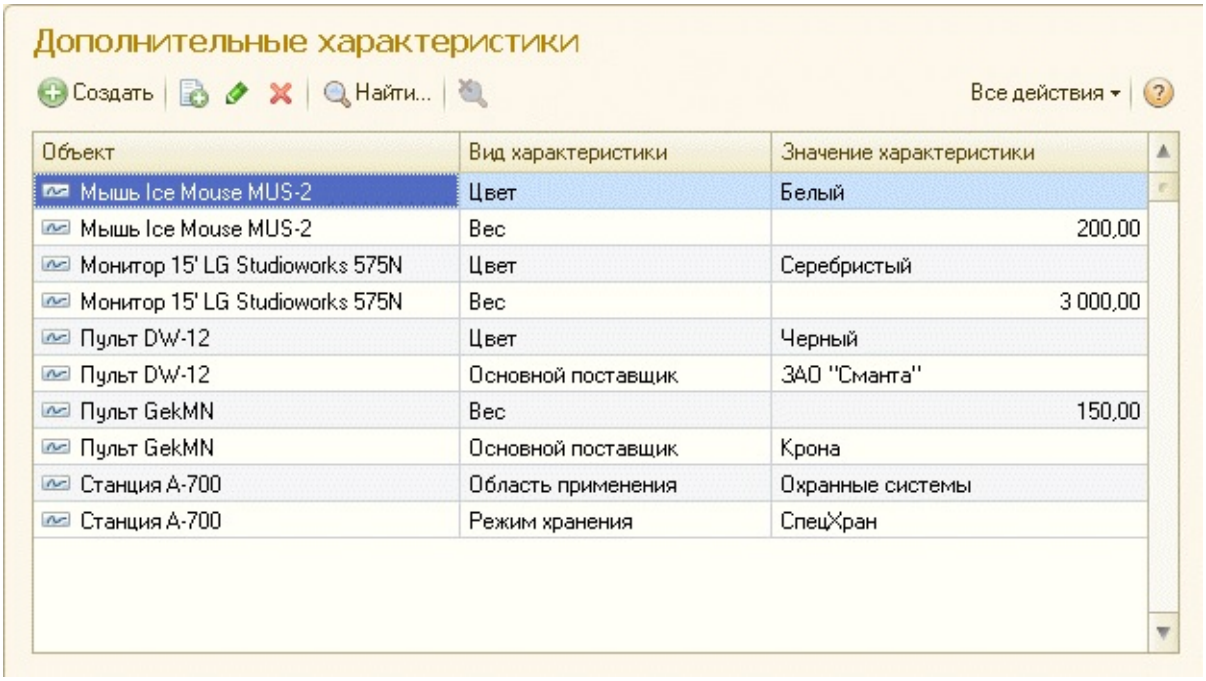
Получение значений характеристик из регистра сведений

Поскольку сам план видов характеристик хранит только описание дополнительных характеристик объектов, то значения этих характеристик должны где-то храниться. Существуют различные способы хранения характеристик в прикладном решении.

подробнее

О способах хранения дополнительных характеристик подробнее можно прочитать в книге «Реализация прикладных задач в системе «1С:Предприятие 8.2» из серии «Профессиональная разработка».

Мы рассмотрим самый простой способ хранения значений характеристик объектов – в регистре сведений. В нашей демонстрационной конфигурации для этого используется регистр сведений *Дополнительные характеристики* (рис. 3.21).



Объект	Вид характеристики	Значение характеристики
Мышь Ice Mouse MUS-2	Цвет	Белый
Мышь Ice Mouse MUS-2	Вес	200,00
Монитор 15' LG Studioworks 575N	Цвет	Серебристый
Монитор 15' LG Studioworks 575N	Вес	3 000,00
Пульт DW-12	Цвет	Черный
Пульт DW-12	Основной поставщик	ЗАО "Сманта"
Пульт GekMN	Вес	150,00
Пульт GekMN	Основной поставщик	Крона
Станция A-700	Область применения	Охранные системы
Станция A-700	Режим хранения	СпецХран

Рис. 3.21. Регистр сведений «Дополнительные характеристики»

Измерение регистра *Объект* содержит ссылку на объект, для которого хранятся значения

характеристик. Измерение *ВидХарактеристики* содержит ссылку на вид характеристики. Ресурс *ЗначениеХарактеристики* содержит, собственно, сами значения характеристик объектов.

Таким образом, при помощи отбора по полю *ВидХарактеристики* из регистра сведений *Дополнительные характеристики* можно получить значения характеристик конкретных объектов номенклатуры, вид характеристик которых указан в параметре *ВидХарактеристики* (листинг 3.16).

Листинг 3.16. Вывод значений характеристик заданного вида из регистра сведений

```
ВЫБРАТЬ
  ДополнительныеХарактеристики.Объект,
  ДополнительныеХарактеристики.ВидХарактеристики,
  ДополнительныеХарактеристики.ЗначениеХарактеристики
ИЗ
  РегистрСведений.ДополнительныеХарактеристики КАК ДополнительныеХарактеристики
ГДЕ
  ДополнительныеХарактеристики.ВидХарактеристики = &ВидХарактеристики
```

В результате выполнения запроса для вида характеристики *Цвет* мы видим, что в регистре сведений *Дополнительные характеристики* содержится информация о цвете следующих объектов номенклатуры (рис. 3.22).

Запрос: РегистрСведений.ДополнительныеХарактеристики (Записей в результате: 3)

Объект	ВидХарактеристики	ЗначениеХарактеристики
Монитор 15' LG Studioworks 575N	Цвет	Серебристый
Пульт DW-12	Цвет	Черный
Мышь Ice Mouse MUS-2	Цвет	Белый

Рис. 3.22. Вывод значений характеристик «Цвет» из регистра сведений

Иногда бывает нужно проверить, есть ли виды характеристик, которые не используются? Для этого нужно выполнить следующий запрос (листинг 3.17).

Листинг 3.17. Вывод видов характеристик, не используемых в регистре сведений

```
ВЫБРАТЬ
  ВидыХарактеристик.Наименование КАК Наименование,
  ДополнительныеХарактеристики.ЗначениеХарактеристики
ИЗ
  ПланВидовХарактеристик.ВидыХарактеристик КАК ВидыХарактеристик
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.ДополнительныеХарактеристики КАК
    ДополнительныеХарактеристики
    ПО ВидыХарактеристик.Ссылка = ДополнительныеХарактеристики.ВидХарактеристики
ГДЕ
  НЕ ВидыХарактеристик.ЭтоГруппа
  И ДополнительныеХарактеристики.ЗначениеХарактеристики ЕСТЬ NULL
```

В запросе выполняется левое соединение плана видов характеристик с регистром сведений, и отбираются те записи из плана видов характеристик, для которых не найдено соответствий в регистре сведений.

В результате выполнения запроса мы видим, что в регистре сведений не использованы два вида характеристик *Габариты* и *Транспортировка* (рис. 3.23).

Запрос: ПланВидовХарактеристик.ВидыХарактеристик (Записей в результате: 2)

Наименование	ЗначениеХарактеристики
Габариты	
Транспортировка	

Рис. 3.23. Вывод видов характеристик, не используемых в регистре сведений

Можно также получить из регистра сведений записи с заданным значением характеристики. Например, можно получить номенклатуру, основным поставщиком которой является выбранный поставщик (листинг 3.18).

Листинг 3.18. Отбор записей из регистра сведений по значению характеристики

```
ВЫБРАТЬ
  ДополнительныеХарактеристики.Объект,
  ДополнительныеХарактеристики.ВидХарактеристики,
  ДополнительныеХарактеристики.ЗначениеХарактеристики
ИЗ
  РегистрСведений.ДополнительныеХарактеристики КАК ДополнительныеХарактеристики
ГДЕ
  ДополнительныеХарактеристики.ЗначениеХарактеристики = &Поставщик
```

В результате выполнения запроса для значения характеристики *Крона* (ссылка на справочник *Контрагенты*) мы видим, что в регистре сведений *Дополнительные характеристики* содержится информация об одной номенклатуре, основным поставщиком которой является контрагент *Крона* (рис. 3.24).

Запрос: РегистрСведений.ДополнительныеХарактеристики (Записей в результате: 1)

Объект	ВидХарактеристики	ЗначениеХарактеристики
Пульт GekMN	Основной поставщик	Крона

Рис. 3.24. Отбор записей из регистра сведений по значению характеристики

Поскольку в нашей демонстрационной конфигурации существует константа *Основной поставщик*, то аналогичного результата можно добиться следующим запросом (листинг 3.19).

Листинг 3.19. Отбор записей из регистра сведений по значению характеристики

```
ВЫБРАТЬ
  ДополнительныеХарактеристики.Объект,
  ДополнительныеХарактеристики.ВидХарактеристики,
  ДополнительныеХарактеристики.ЗначениеХарактеристики
ИЗ
  РегистрСведений.ДополнительныеХарактеристики КАК ДополнительныеХарактеристики
ГДЕ
  ДополнительныеХарактеристики.ЗначениеХарактеристики В
    (ВЫБРАТЬ
      ОсновнойПоставщик.Значение
    ИЗ
      Константа.ОсновнойПоставщик КАК ОсновнойПоставщик)
```

Во вложенном запросе получается значение константы, и затем ссылка на контрагента подставляется в условие отбора основного запроса.

Учет движения средств

Регистры накопления

Для учета движения средств в системе «1С:Предприятие» используются регистры накопления, которые накапливают информацию о состоянии различных показателей, характеризующих хозяйственную деятельность предприятия. В данном разделе мы рассмотрим примеры решения различных прикладных задач, в которых используется обращение к регистрам накопления с помощью языка запросов.

Все примеры, используемые в данном разделе, можно посмотреть в демонстрационной конфигурации «Учет движения средств», которая находится на прилагаемом компакт-диске.

Вся базовая информация для получения итоговых показателей хранится в таблице движений регистра накопления – в основной таблице регистра информационной базы данных, доступной с помощью запросов. Основная таблица содержит следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя реквизита>* – поле, содержащее значения реквизита регистра с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для регистра как объекта конфигурации;
- *<Имя ресурса>* – поле, содержащее значения ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации;
- *Активность* – имеет тип *Булево*. Содержит признак активности записи и влияния на получение итогов регистра;
- *Вид движения* – имеет тип системного перечисления *ВидДвиженияНакопления*. Содержит вид движения данной записи (*Приход* или *Расход*). Обозначает направление приращения указанных в записи ресурсов. Существует только для регистров накопления остатков;
- *Момент времени* – виртуальное поле, не хранится в информационной базе. Содержит объект *МоментВремени* (который включает в себя дату и ссылку на документ-регистратор):

- *Период* – дата записи. Совместно с полями *Регистратор* и *НомерСтроки* определяет положение данной записи на временной оси;
- *Регистратор* – содержит ссылку на документ, которому подчинена данная запись;
- *НомерСтроки* – уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*.

В нашей демонстрационной конфигурации существует регистр накопления остатков *Товары на складах*, который накапливает данные об остатке товаров на складах. Данный регистр имеет измерения *Номенклатура* и *Склад*, ресурс *Количество*, а также реквизиты *Поставщик* и *ВидОперации* для хранения дополнительной информации о движениях. Движения в регистре формируются при проведении документов *Поступление товаров* и *Реализация товаров*, которые являются регистраторами регистра *Товары на складах*.

Кроме того, в демонстрационной конфигурации существует оборотный регистр накопления *Продажи*, который накапливает данные об оборотах товаров по контрагентам. Данный регистр имеет измерения *Номенклатура* и *Контрагент*, ресурсы *Количество* и *Сумма*, а также реквизит *ВидОперации* для хранения дополнительной информации о движениях. Движения в регистре формируются при проведении документа *Реализация товаров*, который является регистратором регистра *Продажи*.

Также в демонстрационной конфигурации существует регистр накопления остатков *Резервы номенклатуры*, который накапливает данные об остатке зарезервированных товаров. Данный регистр имеет измерение *Номенклатура* и ресурс *Количество*. Движения в регистре формируются при проведении документа *Заказ покупателя*, который является регистратором регистра *Резервы номенклатуры*.

На примере этих регистров рассмотрим типичные задачи по получению данных из регистров накопления.

Получение движений регистра накопления

Пример 1

Предположим, необходимо выбрать содержимое движений регистра *ТоварыНаСкладах* с отбором по заданному регистратору, т. е. конкретному документу, который произвел движения в регистре. Для этого можно выполнить следующий запрос (листинг 3.20).

Листинг 3.20. Отбор движений из регистра накопления по регистратору

```
ВЫБРАТЬ
  ТоварыНаСкладах.Период,
  ТоварыНаСкладах.Регистратор КАК Регистратор,
  ТоварыНаСкладах.НомерСтроки КАК НомерСтроки,
  ТоварыНаСкладах.ВидДвижения,
  ТоварыНаСкладах.Номенклатура,
  ТоварыНаСкладах.Склад,
  ТоварыНаСкладах.Количество
ИЗ
  РегистрНакопления.ТоварыНаСкладах КАК ТоварыНаСкладах
```

```
ГДЕ
  ТоварыНаСкладах.Регистратор = &Регистратор
```

В данном запросе на список движений регистра *ТоварыНаСкладах* накладывается отбор по полю *Регистратор*. Выбранный документ-регистратор *ПоступлениеТоваров* или *РеализацияТоваров* передается как значение параметра *&Регистратор*.

Результат выполнения этого запроса представлен на рис. 3.25.

Запрос: РегистрНакопления.ТоварыНаСкладах (Записей в результате: 8)

Период	Регистратор	НомерСтроки	ВидДвижения	Номенклатура	Склад	Количество
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	1	Приход	1С:Аспект 7.7	Фили-2	5
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	2	Приход	1С:Бухгалтерия 7.7 Базовая версия	Фили-2	5
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	3	Приход	1С:Бухгалтерия 7.7 Стандартная версия	Фили-2	2
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	4	Приход	Пульт DW-12	Фили-2	1
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	5	Приход	Пульт GekMN	Фили-2	1
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	6	Приход	Windows XP Home Edition Russian CD	Фили-2	5
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	7	Приход	Windows XP Home Edition Russian UPG CD	Фили-2	10
25.01.2013 0:00:00	Поступление товаров 000000003 от 25.01.2013	8	Приход	Windows XP Professional Russian CD	Фили-2	2

Рис. 3.25. Отбор движений из регистра накопления по регистратору

Пример 2

Усложним задачу. Предположим, необходимо выбрать содержимое движений документов *РеализацияТоваров* по регистру *ТоварыНаСкладах* с отбором по заданному складу. При этом необходимо посчитать количество движений, выполненных каждым регистратором. Для этого можно использовать следующий запрос (листинг 3.21).

Листинг 3.21. Отбор движений из регистра накопления по складу и виду движения

```
ВЫБРАТЬ
  ТоварыНаСкладах.Период,
  ТоварыНаСкладах.Регистратор КАК Регистратор,
  ТоварыНаСкладах.НомерСтроки КАК НомерСтроки,
  ТоварыНаСкладах.ВидДвижения,
  ТоварыНаСкладах.Номенклатура,
  ТоварыНаСкладах.Склад,
  ТоварыНаСкладах.Количество
ИЗ
  РегистрНакопления.ТоварыНаСкладах КАК ТоварыНаСкладах
ГДЕ
  ТоварыНаСкладах.Склад = &Склад
  И ТоварыНаСкладах.Регистратор ССЫЛКА Документ.РеализацияТоваров
ИТОГИ
  КОЛИЧЕСТВО(НомерСтроки)
ПО
  Регистратор
```

В данном запросе на список движений регистра *ТоварыНаСкладах* накладывается отбор по полю *Склад*. Кроме того, выбираются только те движения, которые произвел в регистре документ *РеализацияТоваров*. Для этого при помощи оператора *ССЫЛКА* проверяется, ссылкой на какой документ является поле *Регистратор* регистра накопления.

Поскольку у регистра остатков есть поле *ВидДвижения*, то вторую часть условия отбора можно заменить на *ТоварыНаСкладах.ВидДвижения = ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход)*. Результат будет аналогичным, но для этого нужно быть уверенным, что никакой другой документ больше не производит движений типа *Расход*.

Затем в запросе рассчитываются итоги количества значений поля *НомерСтроки* в разрезе по регистраторам (в принципе подсчет количества можно было реализовать по любому из полей).

Результат выполнения этого запроса представлен на рис. 3.26.

Запрос: РегистрНакопления.ТоварыНаСкладах (Записей в результате: 19)

Период	Регистратор	НомерСтроки	ВидДвижения	Номенклатура	Склад	Количество
	Реализация товаров 000000001 от 12.02.2013	2				
12.02.2013 0:00:00	Реализация товаров 000000001 от 12.02.2013	1	Расход	1С:Бухгалтерия 7.7 Базовая версия	Фили-2	2
12.02.2013 0:00:00	Реализация товаров 000000001 от 12.02.2013	2	Расход	1С:Бухгалтерия 7.7 Стандартная версия	Фили-2	2
	Реализация товаров 000000003 от 13.02.2013	1				
13.02.2013 0:00:00	Реализация товаров 000000003 от 13.02.2013	1	Расход	Пульт PX	Фили-2	1
	Реализация товаров 000000005 от 15.02.2013	3				
15.02.2013 0:00:00	Реализация товаров 000000005 от 15.02.2013	1	Расход	Windows XP Professional Russian CD	Фили-2	1
15.02.2013 0:00:00	Реализация товаров 000000005 от 15.02.2013	2	Расход	Пульт DW-12	Фили-2	1
15.02.2013 0:00:00	Реализация товаров 000000005 от 15.02.2013	3	Расход	Пульт GekMN	Фили-2	1
	Реализация товаров 000000006 от 15.02.2013	1				
15.02.2013 0:00:00	Реализация товаров 000000006 от 15.02.2013	1	Расход	Windows XP Home Edition Russian UPG CD	Фили-2	1
	Реализация товаров 000000033 от 01.03.2013	4				
01.03.2013 0:00:00	Реализация товаров 000000033 от 01.03.2013	1	Расход	Windows XP Home Edition Russian UPG CD	Фили-2	5
01.03.2013 0:00:00	Реализация товаров 000000033 от 01.03.2013	2	Расход	Windows XP Professional Russian CD	Фили-2	1
01.03.2013 0:00:00	Реализация товаров 000000033 от 01.03.2013	3	Расход	1С:Аспект 7.7	Фили-2	3
01.03.2013 0:00:00	Реализация товаров 000000033 от 01.03.2013	4	Расход	1С:Бухгалтерия 7.7 Базовая версия	Фили-2	2
	Реализация товаров 000000034 от 01.03.2013	2				
01.03.2013 0:00:00	Реализация товаров 000000034 от 01.03.2013	1	Расход	1С:Бухгалтерия 7.7 Базовая версия	Фили-2	2
01.03.2013 0:00:00	Реализация товаров 000000034 от 01.03.2013	2	Расход	1С:Бухгалтерия 7.7 Стандартная версия	Фили-2	2

Рис. 3.26. Отбор движений из регистра накопления по складу и виду движения

Пример 3

Рассмотрим следующую задачу. Пусть необходимо подсчитать по данным регистра *ТоварыНаСкладах*, в каком количестве поставлялись товары в разрезе поставщиков. При этом нужны только поступления от производителей за определенный период. Для этого понадобится использовать отбор по дополнительной информации, характеризующей каждое движение, которая хранится в реквизитах регистра *Поставщик* и *ВидОперации*.

Данная задача может быть решена при помощи следующего запроса (листинг 3.22).

Листинг 3.22. Отбор движений из регистра накопления по дополнительной информации

```
ВЫБРАТЬ
  ТоварыНаСкладах.Поставщик,
  ТоварыНаСкладах.Номенклатура,
  СУММА(ТоварыНаСкладах.Количество) КАК Количество
ИЗ
  РегистрНакопления.ТоварыНаСкладах КАК ТоварыНаСкладах
ГДЕ
  ТоварыНаСкладах.Период МЕЖДУ &ДатаНачала И &ДатаОконачания
  И ТоварыНаСкладах.ВидОперации =
  ЗНАЧЕНИЕ(Перечисление.ВидыОпераций.ПоступлениеОтПроизводителей)

СГРУППИРОВАТЬ ПО
```

```
    ТоварыНаСкладах.Поставщик,
    ТоварыНаСкладах.Номенклатура
```

В данном запросе на список движений регистра *ТоварыНаСкладах* накладывается отбор по периоду. А также в условии отбора проверяется, что поле *ВидОперации* должно принимать значение *ПоступлениеОтПроизводителей* перечисления *ВидыОпераций*.

Записи в результате запроса группируются по полям *Поставщик* и *Номенклатура*, и в поле выборки *Количество* выводится суммарное количество поступивших товаров от каждого поставщика.

Результат выполнения этого запроса представлен на рис. 3.27.

Запрос: РегистрНакопления.ТоварыНаСкладах (Записей в результате: 8)

Поставщик	Номенклатура	Количество
Крона	Windows XP Home Edition Russian CD	15
Крона	Windows XP Home Edition Russian UPG CD	30
Крона	Windows XP Professional Russian CD	7
Крона	Пульт DW-12	4
Крона	Пульт GekMN	4
Крона	1С:Аспект 7.7	15
Крона	1С:Бухгалтерия 7.7 Базовая версия	15
Крона	1С:Бухгалтерия 7.7 Стандартная версия	7

Рис. 3.27. Отбор движений из регистра накопления по дополнительной информации

Получение остатков

Получение итогов остатков регистра накопления возможно только в отношении регистра накопления остатков. Для получения данных по остаткам запросом используется виртуальная таблица остатков *Остатки()*. Ее данные представляют собой итоги ресурсов в разрезе измерений. Итоги ресурсов получаются только по активным записям.

Виртуальная таблица остатков не хранится в информационной базе, а строится в момент обращения к ней. Для построения виртуальной таблицы всегда используются данные таблицы итогов регистра (эта таблица недоступна для запросов, поэтому мы ее не рассматриваем) и, при необходимости, таблицы движений регистра. При этом учитываются значения параметров виртуальной таблицы.

Виртуальная таблица *Остатки* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*), с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя ресурса>Остаток* – поле, содержащее остатки ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов,

определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Остаток*.

При построении этой виртуальной таблицы могут использоваться параметры, с помощью которых настраивается или уточняется состав получаемых данных:

- *Период* – имеет тип *Дата*, *МоментВремени* или *Граница*. Используется для указания периода, на значение которого будут рассчитаны остатки. Если параметр не задан, остатки рассчитываются по самую последнюю запись;
- *Условие* – содержит конструкцию языка запросов – условие. Строится по полям регистра накопления (или по подчиненным им полям). Используется для ограничения состава исходных записей, по которым при построении виртуальной таблицы будут получаться итоги. То есть условие будет применяться к исходным записям, а не к уже отобранным. Если параметр не указан, для получения итогов будут анализироваться все активные записи регистра.

Пример 1

Рассмотрим распространенную задачу, когда требуется получить остатки определенного товара на складах на определенный период времени. Для этого нужно выполнить следующий запрос (листинг 3.23).

Листинг 3.23. Вывод остатков заданного товара на складах на заданный период

```
ВЫБРАТЬ
  ТоварыНаСкладахОстатки.Номенклатура КАК Номенклатура,
  ТоварыНаСкладахОстатки.Склад КАК Склад,
  ТоварыНаСкладахОстатки.КоличествоОстаток КАК Количество
ИЗ
  РегистрНакопления.ТоварыНаСкладах.Остатки(&Период, Номенклатура = &Товар) КАК
  ТоварыНаСкладахОстатки
```

В данном запросе в виртуальную таблицу остатков мы передаем период, на который должны быть получены остатки, и условие отбора конкретного товара из списка номенклатуры. Это значит, что при формировании таблицы остатков отбор по товару, указанному в параметре запроса *&Товар*, и отбор по периоду, указанному в параметре *&Период*, будет применен к исходным записям регистра накопления *ТоварыНаСкладах*. Данные регистра накопления остатков *ТоварыНаСкладах* группируются по измерениям *Номенклатура* и *Склад*. В итоге мы получаем суммарное количество значений ресурса *Количество* для каждого товара в разрезе складов.

В результате для товара *Клавиатура Apple Pro Keyboards* на дату *03.02.2013* мы получим следующий результат (рис. 3.28).

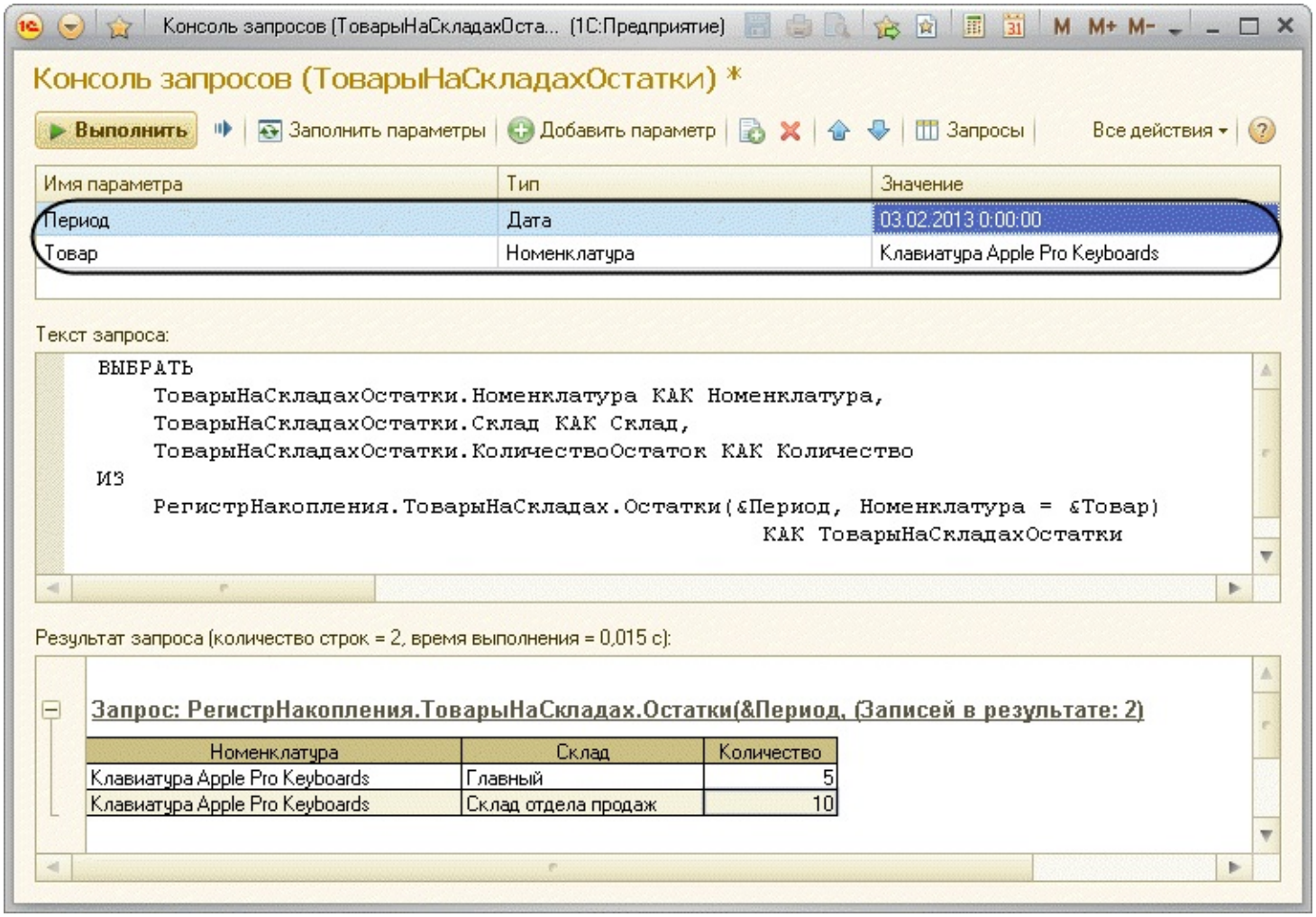


Рис. 3.28. Вывод остатков заданного товара на складах на заданный период

На этом же примере поясним ряд особенностей использования периодов и моментов времени при получении остатков.

Параметр *Период* виртуальной таблицы остатков может принимать значения типа *Дата*, *МоментВремени* и *Граница*. В рассмотренном примере (см. рис. 3.28) *Период* имеет тип *Дата*.

При получении таблицы остатков с указанием параметра *Период* типа *Дата* необходимо учитывать, что дата в системе «1С:Предприятие» включает в свой состав время. Даже если время не указано явно, оно принимает значение *00:00:00*. Таблица остатков строится на начало секунды, т. е. *не включая* границы заданного периода. Поэтому результат, полученный выше (см. рис. 3.28), не включает движений по регистру, сформированных документами за *03.02.2013*.

Таких документов было два – *Поступление товаров №5* и *Поступление товаров №6*. Первым документом было зафиксировано поступление товара *Клавиатура Apple Pro Keyboards* на склад *Главный* в количестве *10*. Вторым документом было зафиксировано поступление товара *Клавиатура Apple Pro Keyboards* на склад *Склад отдела продаж* в количестве *5*.

Теперь зададим значение параметра *Период* как *Момент времени* документа

Поступление товаров №6 от *03.02.2013* и посмотрим, как изменится результат (рис. 3.29).

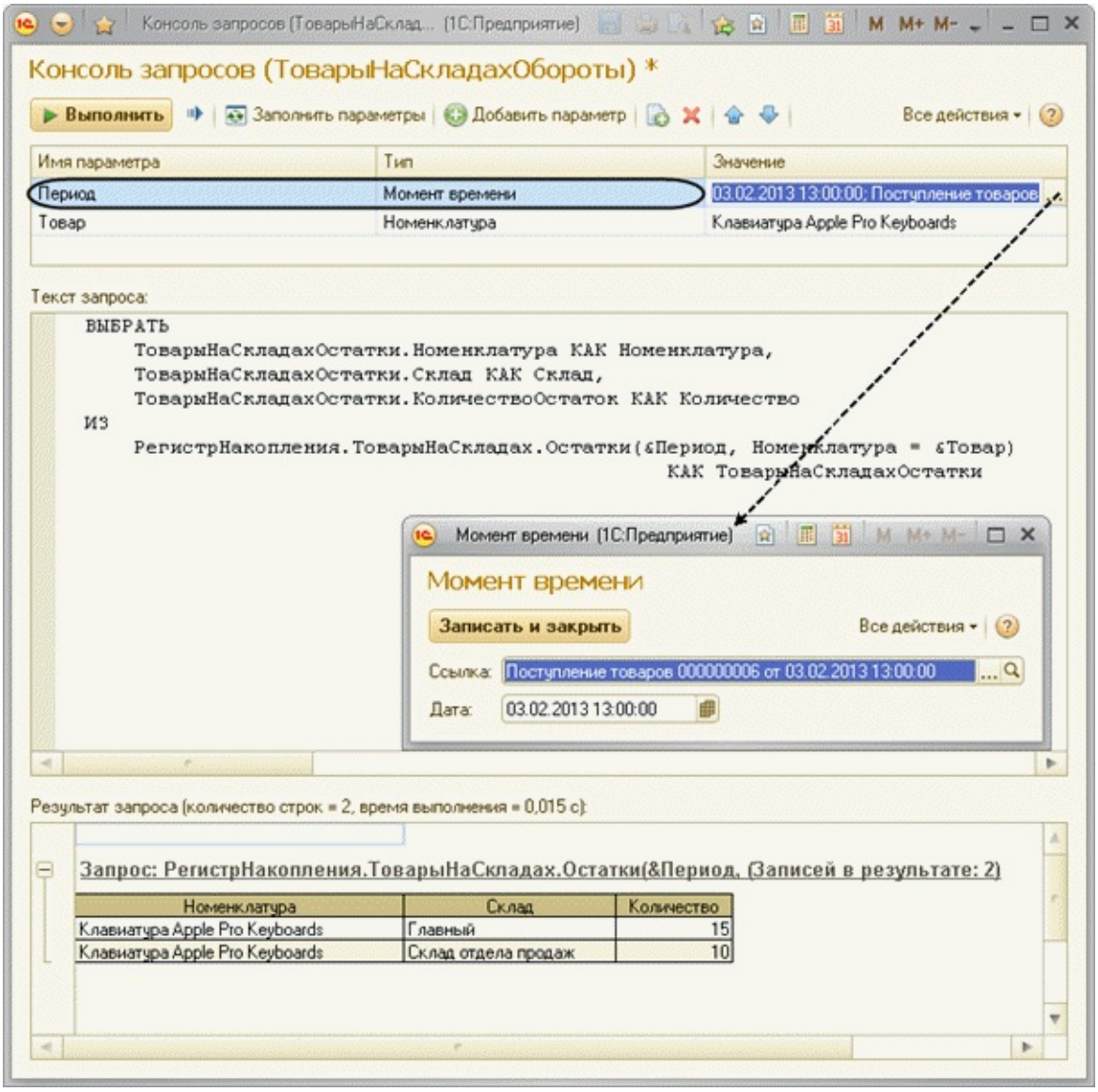


Рис. 3.29. Вывод остатков заданного товара на складах на заданный период

Мы видим, что в результат запроса попали движения документа *Поступление товаров №5*, но не включены движения самого документа *Поступление товаров №6*.

При получении таблицы остатков с указанием параметра *Период* типа *МоментВремени*, полученного из даты документа и ссылки на документ, необходимо иметь в виду, что данные получаются, *исключая* записи движений самого документа.

Для случаев, когда необходимо получить данные об остатках, включая движения, относящиеся к дате или моменту времени, применяют значения параметра *Период* с типом значения *Граница* и видом границы *Включая*.

Зададим значение параметра *Период* как *Граница* на момент времени документа *Поступление товаров №6* от *03.02.2013* с видом границы *Включая* и посмотрим, как изменится результат (рис. 3.30).

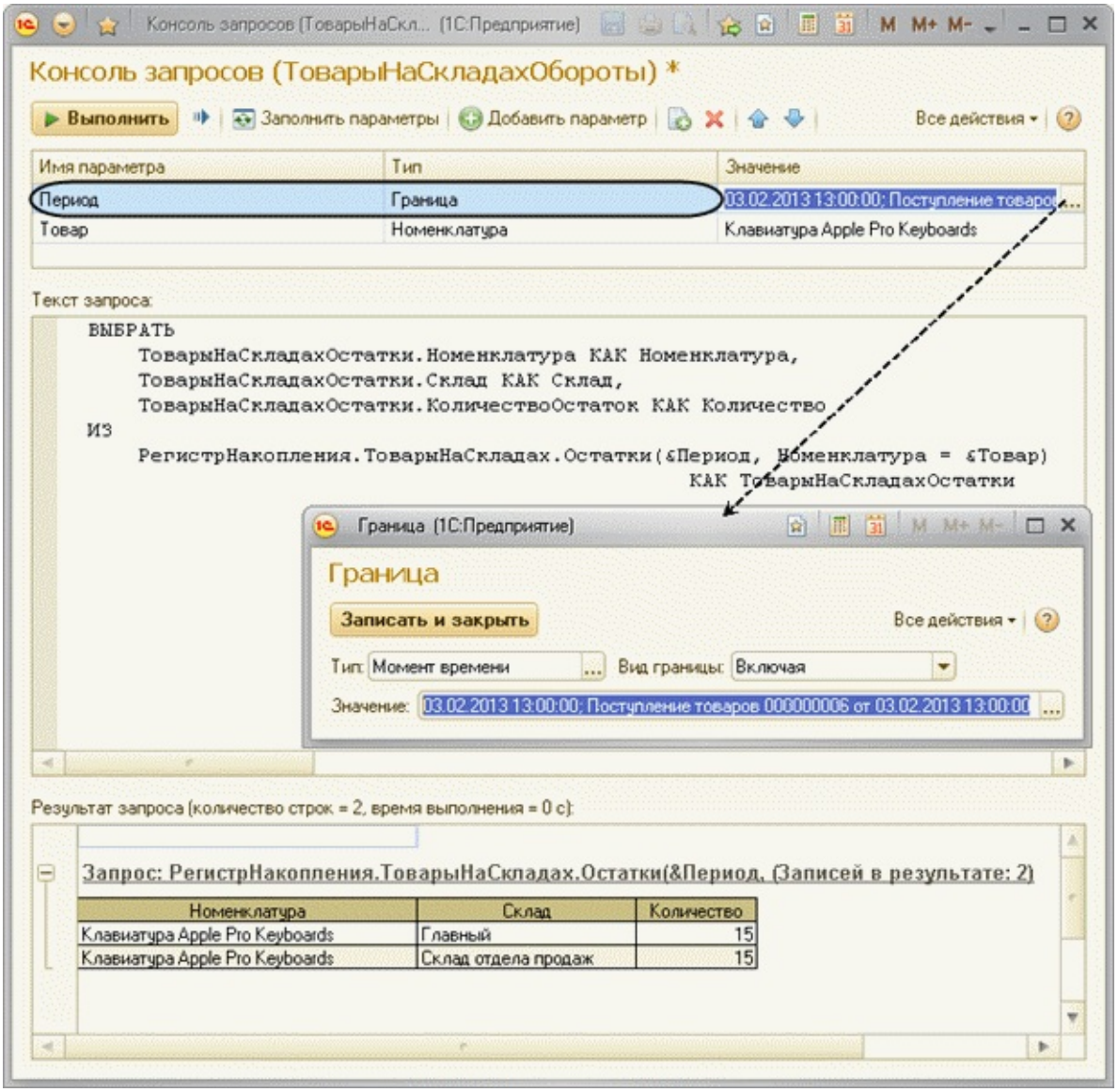


Рис. 3.30. Вывод остатков заданного товара на складах на заданный период

Мы видим, что теперь остатки получены, включая движения указанного документа. Поскольку больше документов за *03.02.2013*, влияющих на остатки товара *Клавиатура Apple Pro Keyboards* в информационной базе нет, то аналогичный результат можно получить, задав значение параметра *Период* как дату – *04.02.2013*.

Пример 2

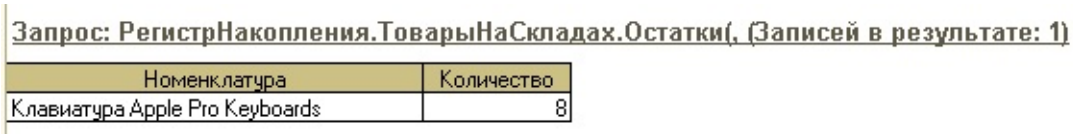
Рассмотрим еще одну тонкость, касающуюся отбора по товару при получении таблицы остатков.

Получим остатки заданного товара за весь период времени без разреза по складам (листинг 3.24).

Листинг 3.24. Вывод остатков заданного товара за весь период

```
ВЫБРАТЬ
  ТоварыНаСкладахОстатки.Номенклатура КАК Номенклатура,
  ТоварыНаСкладахОстатки.КоличествоОстаток КАК Количество
ИЗ
  РегистрНакопления.ТоварыНаСкладах.Остатки( , Номенклатура = &Товар) КАК
  ТоварыНаСкладахОстатки
```

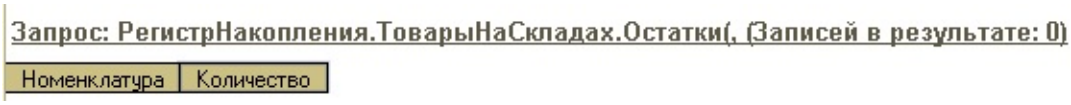
В итоге для товара *Клавиатура Apple Pro Keyboards* мы получим следующий результат (рис. 3.31).


Запрос: РегистрНакопления.ТоварыНаСкладах.Остатки(, (Записей в результате: 1)

Номенклатура	Количество
Клавиатура Apple Pro Keyboards	8

Рис. 3.31. Вывод остатков заданного товара за весь период

Если в качестве значения параметра *Товар* задать отсутствующий в остатке товар, например, *Телефон LG W7200*, то результат запроса будет пустым (рис. 3.32).


Запрос: РегистрНакопления.ТоварыНаСкладах.Остатки(, (Записей в результате: 0)

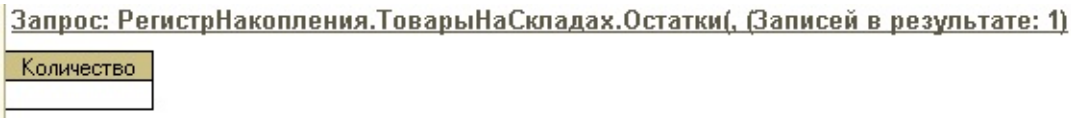
Номенклатура	Количество

Рис. 3.32. Вывод остатков товара, которого нет в остатке

Если же из текста запроса убрать все измерения (листинг 3.25), то результат запроса для отсутствующего в остатке товара будет содержать одну строку с нулевым количеством (рис. 3.33).

Листинг 3.25. Вывод остатков заданного товара за весь период

```
ВЫБРАТЬ
  ТоварыНаСкладахОстатки.КоличествоОстаток КАК Количество
ИЗ
  РегистрНакопления.ТоварыНаСкладах.Остатки( , Номенклатура = &Товар) КАК
  ТоварыНаСкладахОстатки
```


Запрос: РегистрНакопления.ТоварыНаСкладах.Остатки(, (Записей в результате: 1)

Количество

Рис. 3.33. Вывод остатков товара, которого нет в остатке

Это происходит потому, что при формировании виртуальной таблицы остатков агрегация данных производится по измерениям, указанным в запросе. Чтобы не получать такой несколько странный результат, в запросе рекомендуется всегда указывать хотя бы одно измерение.

Рассмотрим следующую задачу. Предположим, необходимо выбрать только те склады, у которых нет никаких остатков по номенклатуре. Это можно сделать с помощью следующего запроса (листинг 3.26).

Листинг 3.26. Вывод складов, не имеющих остатков по номенклатуре

```
ВЫБРАТЬ
  Склады.Наименование
ИЗ
  Справочник.Склады КАК Склады
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки(&Период) КАК
    ТоварыНаСкладахОстатки
    ПО ТоварыНаСкладахОстатки.Склад = Склады.Ссылка
```

```
ГДЕ
  ТоварыНаСкладахОстатки.КоличествоОстаток ЕСТЬ NULL
```

В запросе справочник *Склады* левым соединением связывается с таблицей остатков регистра накопления *ТоварыНаСкладах*. В результат запроса попадают только те записи из справочника складов, для которых не найдено соответствий в таблице остатков, т. е. на этих складах нет остатка по номенклатуре. Такие записи в поле *КоличествоОстаток* содержат значения *NULL*.

В результате на дату *01.02.2013* мы получим следующий результат (рис. 3.34).

Рис. 3.34. Вывод складов, не имеющих остатков по номенклатуре

Пример 3

Рассмотрим следующую задачу. Предположим, необходимо выбрать номенклатуру из регистра остатков, которая числилась на складе в течение дня. То есть нужно выбрать номенклатуру, имевшую ненулевой остаток и на начало, и на конец дня.

Для этого нужно таблицу остатков, полученную на начало дня, связать полным соединением по номенклатуре и складу с этой же таблицей, полученной на конец дня. И в результате отобрать те записи, которые имеют ненулевой остаток в обеих таблицах (листинг 3.27).

Листинг 3.27. Вывод номенклатуры, которая числилась на складе в течение дня

```
ВЫБРАТЬ
  ОстаткиТоваровНаНачалоДня.Номенклатура КАК Номенклатура,
  ОстаткиТоваровНаНачалоДня.Склад КАК Склад,
  ОстаткиТоваровНаНачалоДня.КоличествоОстаток КАК ОстатокНаНачалоДня,
  ОстаткиТоваровНаКонецДня.КоличествоОстаток КАК ОстатокНаКонецДня
ИЗ
  РегистрНакопления.ТоварыНаСкладах.Остатки(НАЧАЛОПЕРИОДА(&Период, ДЕНЬ)) КАК
  ОстаткиТоваровНаНачалоДня
    ПОЛНОЕ СОЕДИНЕНИЕ
      РегистрНакопления.ТоварыНаСкладах.Остатки(КОНЕЦПЕРИОДА(&Период, ДЕНЬ)) КАК
      ОстаткиТоваровНаКонецДня
    ПО ОстаткиТоваровНаНачалоДня.Номенклатура = ОстаткиТоваровНаКонецДня.Номенклатура
      И ОстаткиТоваровНаНачалоДня.Склад = ОстаткиТоваровНаКонецДня.Склад
ГДЕ
  НЕ ОстаткиТоваровНаНачалоДня.КоличествоОстаток ЕСТЬ NULL
  И НЕ ОстаткиТоваровНаКонецДня.КоличествоОстаток ЕСТЬ NULL

УПОРЯДОЧИТЬ ПО
  Номенклатура

АВТОУПОРЯДОЧИВАНИЕ
```

В итоге на дату *01.03.2013* мы получим следующий результат (рис. 3.35).

Запрос: РегистрНакопления.ТоварыНаСкладах.Остатки(НАЧАЛОПЕРИОДА(&Период, (Записей в результате: 53)

Номенклатура	Склад	ОстатокНаНачалоДня	ОстатокНаКонецДня
1С:Аспект 7.7	Главный	12	12
1С:Аспект 7.7	Склад отдела продаж	1	1
1С:Аспект 7.7	Фили-2	11	8
1С:Бухгалтерия 7.7 Базовая версия	Главный	10	10
1С:Бухгалтерия 7.7 Базовая версия	Фили-2	9	5
1С:Бухгалтерия 7.7 Стандартная версия	Главный	5	5
1С:Бухгалтерия 7.7 Стандартная версия	Фили-2	3	1
Windows XP Home Edition Russian CD	Главный	9	9
Windows XP Home Edition Russian CD	Склад отдела продаж	1	1
Windows XP Home Edition Russian CD	Фили-2	12	12
Windows XP Home Edition Russian UPG CD	Главный	20	20
Windows XP Home Edition Russian UPG CD	Склад отдела продаж	3	3
Windows XP Home Edition Russian UPG CD	Фили-2	21	16
Windows XP Professional Russian CD	Главный	2	2
Windows XP Professional Russian CD	Фили-2	5	4

Рис. 3.35. Вывод номенклатуры, которая числилась на складе в течение дня (фрагмент)

Проверка остатков

Данные об остатках товаров на складах часто используются для проверки наличия товара в требуемом количестве при проведении документов, учитывающих расход товара. Общая методика контроля остатков при проведении документа заключается в том, что сначала формируются и записываются движения документа в регистрах. После этого, когда движения уже записаны, читаются остатки товаров из регистров. Если появились отрицательные остатки, значит, такой документ не должен проводиться. Нужно сообщить пользователю, каких товаров не хватает, и отменить проведение документа.

Обычно контроль остатков выполняется при оперативном проведении документов, но могут быть случаи, когда нужно проконтролировать остатки товаров при проведении документов, которым запрещено оперативное проведение. Например, в нашей демонстрационной конфигурации существует документ *ЗаказПокупателя*, который проводится неоперативно. При проведении данного документа необходимо отложить товар в резерв (сформировать движения по регистру *РезервыНоменклатуры*), но при этом выполнить контроль возможности проведения данной операции.

Отложить товар в резерв можно только в том случае, если его хватает, то есть если достаточно свободного (незарезервированного) остатка товара на складах. То есть остаток товара, накопленный в регистре *ТоварыНаСкладах*, не должен быть меньше, чем остаток товара, накопленный в регистре *РезервыНоменклатуры* (с учетом резерва по текущему документу), иначе документ *ЗаказПокупателя*, резервирующий товар, не может быть проведен.

Ниже (листинг 3.28) приведен текст процедуры *ОбработкаПроведения()* документа *ЗаказПокупателя*. Процедура приводится полностью для того, чтобы лучше был понятен контекст применения запросов, используемых при проведении документа. Но дальше мы подробно рассмотрим только фрагменты, выделенные жирным шрифтом, касающиеся использования языка запросов в обработке проведения.

Листинг 3.28. Пример процедуры «ОбработкаПроведения» документа «Заказ покупателя»

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
```

```
// Укажем, движения по каким регистрам нужно записывать.
Движения.РезервыНоменклатуры.Записывать = Истина;

// Создать менеджер временных таблиц.
МенеджерВТ = Новый МенеджерВременныхТаблиц;

Запрос = Новый Запрос;

// Укажем, какой менеджер временных таблиц использует этот запрос.
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

Запрос.Текст =
  "ВЫБРАТЬ
  |        ЗаказПокупателяСостав.Номенклатура,
  |        СУММА(ЗаказПокупателяСостав.Количество) КАК КоличествоВДокументе
  |ПОМЕСТИТЬ НоменклатураДокумента
  |ИЗ
  |        Документ.ЗаказПокупателя.Состав КАК ЗаказПокупателяСостав
  |ГДЕ
  |        ЗаказПокупателяСостав.Ссылка = &Ссылка
  |СГРУППИРОВАТЬ ПО
  |        ЗаказПокупателяСостав.Номенклатура";

Запрос.УстановитьПараметр("Ссылка", Ссылка);

Результат = Запрос.Выполнить();

Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос2.Текст =
  "ВЫБРАТЬ
  |        НоменклатураДокумента.Номенклатура,
  |        НоменклатураДокумента.КоличествоВДокументе
  |ИЗ
  |        НоменклатураДокумента КАК НоменклатураДокумента";

// Установим необходимость блокировки данных в регистре РезервыНоменклатуры.
Движения.РезервыНоменклатуры.БлокироватьДляИзменения = Истина;
// Запишем пустые наборы записей, чтобы читать резервы без учета данных в документе.
Движения.РезервыНоменклатуры.Записать();

Результат = Запрос2.Выполнить();

ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

  // Сформировать движения по регистру РезервыНоменклатуры.
  Движение = Движения.РезервыНоменклатуры.Добавить();
  Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
  Движение.Период = Дата;
  Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
  Движение.Количество = ВыборкаДетальныеЗаписи.КоличествоВДокументе;

КонецЦикла;

// Запишем движения.
Движения.Записать();
```

```
  // Контроль остатков.
  Запрос3 = Новый Запрос;
  Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
  Запрос3.Текст =
    "ВЫБРАТЬ
    |        НоменклатураДокумента.Номенклатура,
    |        НоменклатураДокумента.Номенклатура.Представление КАК Товар,
    |        ЕСТЬNULL(ТоварыНаСкладахОстатки.КоличествоОстаток, 0) КАК Остаток,
    |        ЕСТЬNULL(РезервыНоменклатурыОстатки.КоличествоОстаток, 0) КАК Резерв
    |ИЗ
    |        НоменклатураДокумента КАК НоменклатураДокумента
    |        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки(
    |                        &Момент,
    |                        Номенклатура В
    |                                        (ВЫБРАТЬ
    |                                                НоменклатураДокумента.Номенклатура
    |                                        ИЗ
    |                                                НоменклатураДокумента)) КАК
    ТоварыНаСкладахОстатки
    |        ПО НоменклатураДокумента.Номенклатура = ТоварыНаСкладахОстатки.Номенклатура
    |        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.РезервыНоменклатуры.Остатки(
    |                        &Момент,
    |                        Номенклатура В
    |                                        (ВЫБРАТЬ
    |                                                НоменклатураДокумента.Номенклатура
    |                                        ИЗ
    |                                                НоменклатураДокумента)) КАК
    РезервыНоменклатурыОстатки
    |        ПО НоменклатураДокумента.Номенклатура =
    РезервыНоменклатурыОстатки.Номенклатура
    |ГДЕ
    |        ЕСТЬNULL(ТоварыНаСкладахОстатки.КоличествоОстаток,0) -
    ЕСТЬNULL(РезервыНоменклатурыОстатки.КоличествоОстаток,0) ;

  ГраницаПоДокумент = Новый Граница(МоментВремени(), ВидГраницы.Включая);
  Запрос3.УстановитьПараметр("Момент", ГраницаПоДокумент);

  РезультатСНехваткой = Запрос3.Выполнить();

  ВыборкаРезультатаСНехваткой = РезультатСНехваткой.Выбрать();

  Пока ВыборкаРезультатаСНехваткой.Следующий() Цикл

    Сообщение = Новый СообщениеПользователю();
    Сообщение.Текст = "Не хватает " +
      Строка(- (ВыборкаРезультатаСНехваткой.Остаток -
        ВыборкаРезультатаСНехваткой.Резерв)) +
        " единиц товара """ + ВыборкаРезультатаСНехваткой.Товар + """!";
    Сообщение.Сообщить();
    Отказ = Истина;

  КонецЦикла;

КонецПроцедуры
```

Сначала в первом запросе *Запрос* данные табличной части документа *ЗаказПокупателя* помещаются во временную таблицу *НоменклатураДокумента*. Эти данные посредством менеджера временных таблиц будут затем использованы в запросе для формирования

движений документа и в запросе к регистрам накопления для контроля свободных (незарезервированных) остатков товаров.

Поскольку в документе может присутствовать нескольких строк с одинаковой номенклатурой, данные табличной части группируются по полю *Номенклатура*, чтобы получить суммарное *Количество* товара в документе.

Далее во втором запросе *Запрос2* выполняется запрос к временной таблице *НоменклатураДокумента*, содержащей перечень заказанных товаров, и в цикле обхода выборки результата запроса формируются движения по регистру *РезервыНоменклатуры*.

Напомним, что временная таблица, полученная при выполнении первого запроса, становится доступной второму и третьему запросу, благодаря использованию общего менеджера временных таблиц *МенеджерВТ*.

После этого в третьем запросе *Запрос3* выполняется контроль достаточности свободного (незарезервированного) остатка товаров. Для этого выполняется запрос для получения отрицательных свободных остатков номенклатуры, содержащейся в табличной части проводимого документа. Запрос реализован левым соединением данных временной таблицы с данными виртуальных таблиц остатков из регистров *ТоварыНаСкладах* и *РезервыНоменклатуры*. В результат запроса отбираются записи, в которых количество резерва по товару в регистре *РезервыНоменклатуры* превосходит его остаток в регистре *ТоварыНаСкладах*.

Поскольку документу запрещено оперативное проведение (свойство документа *Оперативное проведение* установлено в значение *Запретить*), остатки номенклатуры нужно контролировать на момент времени документа, включая движения самого документа. Поэтому в качестве параметра *Период* при установке параметра запроса в виртуальные таблицы остатков передается значение момента времени документа, включая движения самого документа (*Новый Граница(МоментВремени()*, *ВидГраницы.Включая)*).

Обратите внимание, что виртуальные таблицы остатков получены с применением отбора к исходным записям регистров накопления по тем номенклатурным позициям, которые входят в состав проводимого документа. Как уже говорилось, это повышает быстродействие и эффективность выполнения запроса за счет ограничения объема расчета остатков на этапе построения виртуальной таблицы.

подробнее

Раздел «Использовать параметры виртуальных таблиц».

Поскольку может иметь место ситуация, когда данные по неким номенклатурным позициям в регистре отсутствуют, то для выходных полей запроса *КоличествоОстаток* применена функция *ЕСТЬNULL()* для преобразования возможных *Null* значений в

числовое значение *0*. Это сделано для того, чтобы впоследствии выполнять арифметические действия с данными результата запроса, так как иначе они окажутся непригодными для подобных действий.

После этого выборка записей запроса обходится в цикле, и если есть такие записи, в которых количество резерва по товару в регистре *РезервыНоменклатуры* превосходит его остаток в регистре *ТоварыНаСкладах*, сообщение о них выводится пользователю. В этом случае параметр *Отказ* процедуры обработчика события получает значение *Истина*, что приведет к откату транзакции записи документа с проведением.

Для вывода пользователю информации о нехватке товаров в запросе получается представление номенклатуры, и именно это представление показывается пользователю (рис. 3.36).

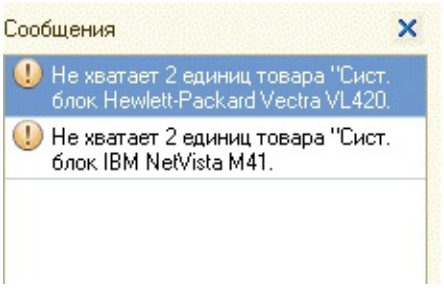


Рис. 3.36. Информация о нехватке товаров при их резервировании

Получение оборотов

Задачи получения итогов оборотов могут решаться и для регистров накопления остатков, и для оборотных регистров. Для получения данных по оборотам из регистра накопления запросом используется виртуальная таблица оборотов *Обороты()*. Ее данные представляют собой обороты ресурсов в разрезе измерений. Обороты ресурсов получаются только по активным записям.

Виртуальная таблица *Обороты* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*), с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя ресурса>Оборот* – поле, содержащее обороты ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Оборот*. Для регистров оборотов оборот подсчитывается как сумма всех движений. Для регистров остатков оборот подсчитывается как сумма всех движений *Приход* со знаком «+» (плюс) и *Расход* со знаком «-» (минус);

- *<Имя ресурса>Приход* – поле, содержащее сумму всех движений типа *Приход* по ресурсу регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Приход*. Существует только для регистра накопления остатков;
- *<Имя ресурса>Расход* – поле, содержащее сумму всех движений типа *Расход* по ресурсу регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Расход*. Существует только для регистра накопления остатков;
- *НомерСтроки* – имеет тип *Число*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Запись*. Содержит значение поля *НомерСтроки* записи движения регистра;
- *Период* – имеет тип *Дата*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Секунда*, *Минута*, *Час*, *Регистратор* или *Запись*. Данное поле содержит начальную дату и время периода, к которому относится оборот регистра. Это поле не может быть использовано в условии отбора записей;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Регистратор* или *Запись*. Данное поле содержит ссылку на документ-регистратор, к которому относится оборот регистра. Это поле не может быть использовано в условии отбора записей.

При построении этой виртуальной таблицы могут использоваться параметры, с помощью которых настраивается или уточняется состав получаемых данных:

- *НачалоПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Используется для указания начала периода расчета оборотов. Значение этого параметра по умолчанию включается в период расчета оборотов. Если параметр не задан, обороты рассчитываются с самой первой записи;
- *КонецПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Используется для указания конца периода расчета оборотов. Значение этого параметра по умолчанию включается в период расчета оборотов. Если параметр не задан, обороты рассчитываются по самую последнюю запись;
- *Периодичность* – содержит конструкцию языка запросов. Задается одним из следующих вариантов: *Период*, *Запись*, *Регистратор*, *Секунда*, *Минута*, *Час*, *День*, *Неделя*, *Декада*, *Месяц*, *Квартал*, *Полугодие*, *Год*, *Авто*. Используется для указания дополнительного разворота оборотов по периодичности. При указании значения *Период* обороты представляются за весь период (с *НачалоПериода* по *КонецПериода*) без дополнительных разворотов. В остальных случаях – с разворотом по записям, регистраторам, неделям, месяцам, кварталам и т. п. соответственно. Значение по умолчанию – *Период*;

- *Условие* – содержит конструкцию языка запросов – условие. Строится по полям регистра накопления (или по подчиненным им полям). Используется для ограничения состава исходных записей, по которым при построении виртуальной таблицы оборотов будут получаться обороты. То есть условие будет применяться к исходным записям, а не к уже отобранным. Если параметр не указан, для получения оборотов будут анализироваться все активные записи регистра.

Пример 1

Рассмотрим пример получения оборотов из регистра накопления остатков *ТоварыНаСкладах*. Предположим, требуется получить данные об оборотах, приходе и расходе товаров по складам за период. Это можно сделать с помощью следующего запроса (листинг 3.29).

Листинг 3.29. Вывод оборотов товаров на складах за заданный период

```
ВЫБРАТЬ
  ТоварыНаСкладахОбороты.Номенклатура,
  ТоварыНаСкладахОбороты.Склад,
  ТоварыНаСкладахОбороты.КоличествоОборот,
  ТоварыНаСкладахОбороты.КоличествоПриход,
  ТоварыНаСкладахОбороты.КоличествоРасход
ИЗ
  РегистрНакопления.ТоварыНаСкладах.Обороты( &НачалоПериода, КОНЕЦПЕРИОДА(&КонецПериода,
  ДЕНЬ), , ) КАК ТоварыНаСкладахОбороты

УПОРЯДОЧИТЬ ПО
  Номенклатура

АВТОУПОРЯДОЧИВАНИЕ
```

В данном запросе в виртуальную таблицу оборотов мы передаем границы интервала, за который нужно получить обороты. Чтобы получить конец дня, мы используем функцию языка запросов *КОНЕЦПЕРИОДА()*. При построении виртуальной таблицы оборотов данные регистра накопления остатков *ТоварыНаСкладах* группируются по измерениям *Номенклатура* и *Склад*. В итоге мы получаем суммарное количество оборотов, приходов и расходов по ресурсу *Количество* для каждого товара в разрезе складов.

Если мы зададим значение параметров *НачалоПериода* и *КонецПериода* как *12.02.13*, мы получим обороты товаров на складах в течение дня (рис. 3.37).

Запрос: РегистрНакопления.ТоварыНаСкладах.Обороты((Записей в результате: 8)

Номенклатура	Склад	КоличествоОборот	КоличествоПриход	КоличествоРасход
1С:Аспект 7.7	Фили-2	1	1	
1С:Бухгалтерия 7.7 Базовая версия	Фили-2	-1	1	2
1С:Бухгалтерия 7.7 Стандартная версия	Фили-2	-1	1	2
Windows XP Home Edition Russian CD	Фили-2	2	2	
Windows XP Home Edition Russian UPG CD	Фили-2	2	2	
Windows XP Professional Russian CD	Фили-2	2	2	
Пульт DW-12	Фили-2	1	1	
Пульт GekMN	Фили-2	1	1	

Рис. 3.37. Вывод оборотов товаров на складах за заданный период

В результате запроса отражены движения типа *Приход* по документу *Поступление товаров №14* от *12.02.13* и движения типа *Расход* по документу *Реализация товаров №1* от *12.02.13*.

На этом же примере поясним ряд особенностей использования периодов и моментов времени при получении оборотов.

Параметры *НачалоПериода* и *КонецПериода* виртуальной таблицы оборотов могут принимать значения типа *Дата*, *МоментВремени* и *Граница*. В рассмотренном примере (см. рис. 3.37) параметры *НачалоПериода* и *КонецПериода* имеют тип *Дата*.

При получении таблицы оборотов с указанием параметров *НачалоПериода* и *КонецПериода* типа *Дата* необходимо учитывать, что дата в системе «1С:Предприятие» включает в свой состав время. Даже если время не указано явно, оно принимает значение *00:00:00*. Поэтому чтобы включить в результат запроса данные на конец дня, указанного в параметре *КонецПериода*, нужно использовать функцию языка запросов *КОНЕЦПЕРИОДА()*.

Таблица оборотов строится с начала секунды значения параметра *НачалоПериода* по конец секунды параметра *КонецПериода*, то есть *включая* границы заданного периода.

Теперь зададим значение параметра *НачалоПериода* как *Момент времени* документа *Поступление товаров №14* от *12.02.2013* и посмотрим, как изменится результат (рис. 3.38).

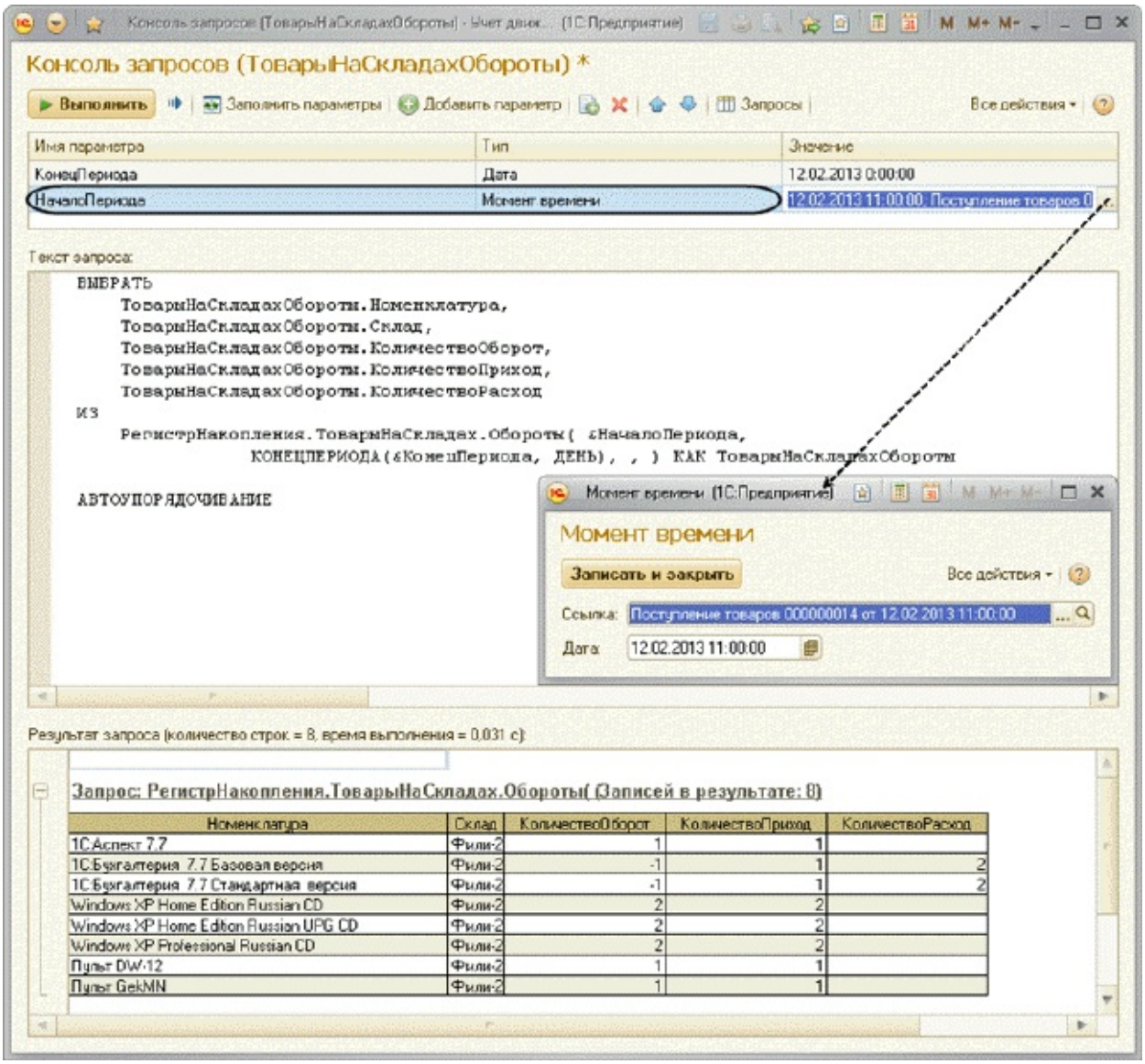


Рис. 3.38. Вывод оборотов товаров на складах за заданный период

Мы видим, что результат запроса не изменился, то есть он включает движения документа *Поступление товаров №14*.

При получении таблицы оборотов с указанием параметров *НачалоПериода* и *КонецПериода* типа *МоментВремени*, полученного из даты документа и ссылки на документ, необходимо иметь в виду, что данные получаются, *включая* записи движений самого документа.

Для случаев, когда необходимо получить данные об оборотах, исключая движения, относящиеся к дате или моменту времени, применяют значения параметров *НачалоПериода* и *КонецПериода* с типом значения *Граница* и видом границы *Исключая*.

Зададим значение параметра *НачалоПериода* как *Граница* на момент времени документа *Поступление товаров №14* от *12.02.2013* с видом границы *Исключая* и посмотрим, как изменится результат (рис. 3.39).

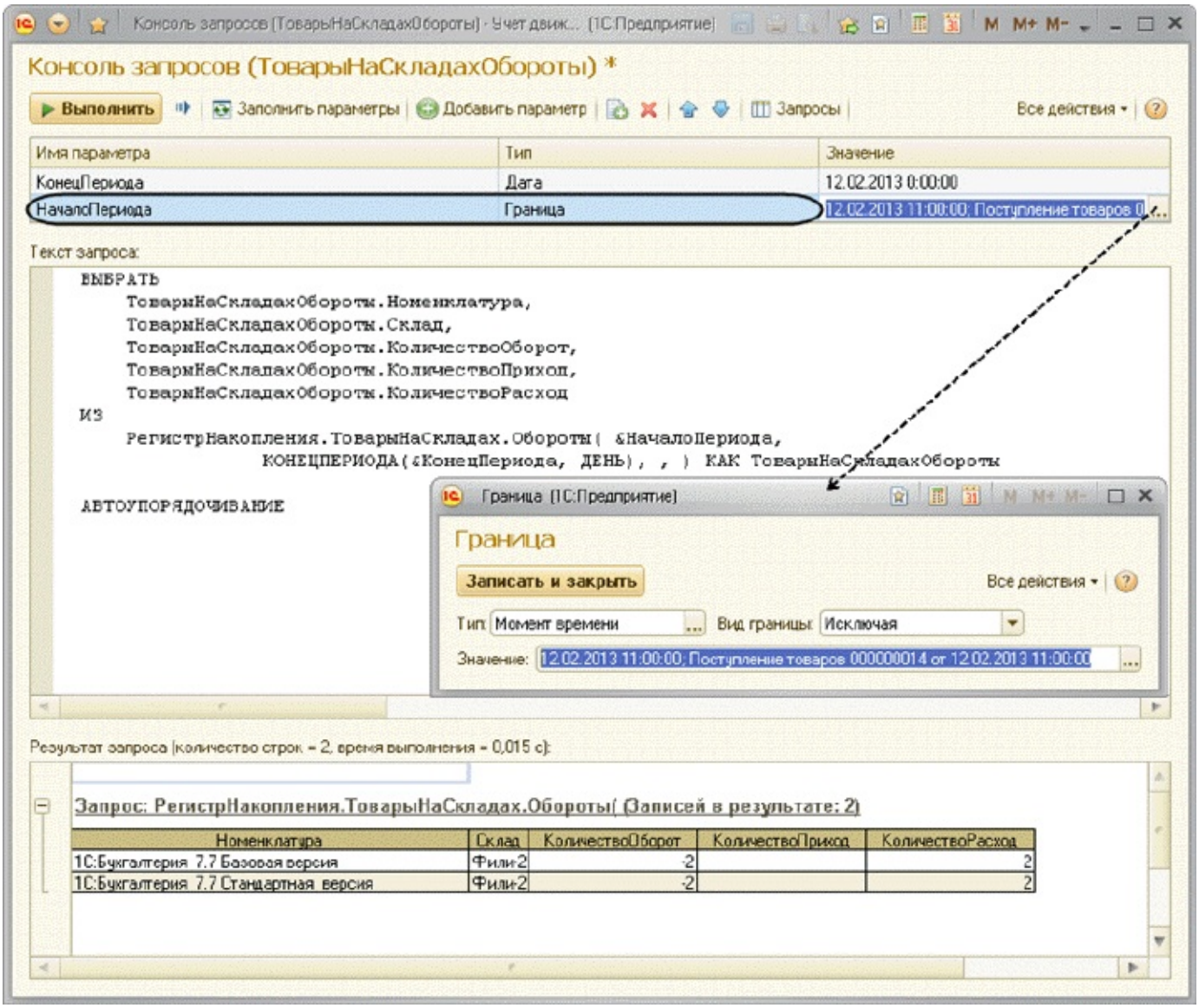


Рис. 3.39. Вывод оборотов товаров на складах за заданный период

Мы видим, что в результате запроса отсутствуют данные о приходе товара, т. е. обороты получены, исключая движения указанного документа.

Пример 2

Рассмотрим теперь следующий пример получения оборотов из оборотного регистра накопления *Продажи*. Предположим, требуется получить данные о продажах определенного товара по контрагентам за определенный период. Это можно сделать с помощью следующего запроса (листинг 3.30).

Листинг 3.30. Вывод оборотов заданного товара по контрагентам за заданный период

```
ВЫБРАТЬ
  ПродажиОбороты.Номенклатура,
  ПродажиОбороты.Контрагент,
  ПродажиОбороты.КоличествоОборот КАК Количество,
  ПродажиОбороты.СуммаОборот КАК Сумма
ИЗ
  РегистрНакопления.Продажи.Обороты(&НачалоПериода, КОНЕЦПЕРИОДА(&КонецПериода, ДЕНЬ), ,
  Номенклатура = &Товар) КАК ПродажиОбороты
```

В данном запросе в виртуальную таблицу оборотов мы передаем границы интервала, за который нужно получить обороты, и товар, по которому отбираются исходные записи оборотного регистра накопления. Чтобы получить конец дня, мы используем функцию языка запросов *КОНЕЦПЕРИОДА()*. При построении виртуальной таблицы оборотов данные оборотного регистра накопления *Продажи* группируются по измерениям *Номенклатура* и *Контрагент*. В итоге мы получаем суммарное количество значений ресурсов *Количество* и *Сумма* для заданного товара в разрезе контрагентов.

В результате для товара *Клавиатура Apple Pro Keyboards* за период *01.02.2013 – 28.02.2013* мы получим следующий результат (рис. 3.40).

Запрос: РегистрНакопления.Продажи.Обороты(&НачалоПериода, (Записей в результате: 6)

Номенклатура	Контрагент	Количество	Сумма
Клавиатура Apple Pro Keyboards	Магазин "Грузинский вал"	14	1 015
Клавиатура Apple Pro Keyboards	Радонеж, ООО	1	75
Клавиатура Apple Pro Keyboards	Глаголев Максим Олегович	1	75
Клавиатура Apple Pro Keyboards	Компания "Риона"	8	600
Клавиатура Apple Pro Keyboards	ЦветМетМаш	3	225
Клавиатура Apple Pro Keyboards	Ялта-Лтд	52	3 900

Рис. 3.40. Вывод оборотов заданного товара по контрагентам за заданный период

Вывод оборотов с заданной периодичностью

Часто бывает нужно получить дополнительный разворот оборотов по периодичности, например, получить данные о продажах товаров по месяцам в течение года или по неделям в течение месяца. Для этого при построении виртуальной таблицы оборотов нужно использовать параметр *Периодичность*. При этом в запросе к виртуальной таблице становится доступным поле *Период*.

В следующем запросе (листинг 3.31) получаются обороты заданного товара за заданный период с периодичностью *НЕДЕЛЯ*.

Листинг 3.31. Вывод оборотов заданного товара за заданный период с периодичностью «Неделя»

```
ВЫБРАТЬ
  ПродажиОбороты.Период,
  ПродажиОбороты.Номенклатура,
  ПродажиОбороты.КоличествоОборот КАК Количество,
  ПродажиОбороты.СуммаОборот КАК Сумма
ИЗ
  РегистрНакопления.Продажи.Обороты(&НачалоПериода, КОНЕЦПЕРИОДА(&КонецПериода, ДЕНЬ),
  НЕДЕЛЯ, Номенклатура = &Товар) КАК ПродажиОбороты

УПОРЯДОЧИТЬ ПО
  Период
```

В результате для товара *Клавиатура Apple Pro Keyboards* за период *01.02.2013 – 28.02.2013* мы получаем данные о продажах с разворотом по неделям (рис. 3.41).

Запрос: РегистрНакопления.Продажи.Обороты(&НачалоПериода, (Записей в результате: 3)

Период	Номенклатура	Количество	Сумма
11.02.2013 0:00:00	Клавиатура Apple Pro Keyboards	7	490
18.02.2013 0:00:00	Клавиатура Apple Pro Keyboards	18	1 350
25.02.2013 0:00:00	Клавиатура Apple Pro Keyboards	54	4 050

Рис. 3.41. Вывод оборотов заданного товара за заданный период с периодичностью «Неделя»

Мы видим, что данные о продажах за первую и вторую неделю февраля (*01.02.2013*, *04.02.2013*) отсутствуют, т. к. реализация товаров началась с *12.02.2013*.

Вывод оборотов продаж товаров за период с определенной периодичностью удобно представить в виде кросс-отчета, в котором товары расположены в строках, а периоды – в колонках таблицы.

Данные для такого отчета будут получаться при выполнении следующего запроса (листинг 3.32).

Листинг 3.32. Запрос для получения данных

```
Запрос.Текст =
  "ВЫБРАТЬ
  |         ПродажиОбороты.Номенклатура КАК Товар,
  |         ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Номенклатура) КАК ТоварПредставление,
  |         ПродажиОбороты.Период КАК Период,
  |         ПродажиОбороты.КоличествоОборот КАК КоличествоОборот
  |ИЗ
  |         РегистрНакопления.Продажи.Обороты(&НачалоПериода, &КонецПериода, НЕДЕЛЯ, ) КАК
  ПродажиОбороты
  |
  |УПОРЯДОЧИТЬ ПО
  |         Номенклатура,
  |         Период
  |АВТОУПОРЯДОЧИВАНИЕ
  |
  |ИТОГИ
  |         СУММА(КоличествоОборот)
  |ПО
  |         ОБЩИЕ,
  |         Товар,
  |         Период ПЕРИОДАМИ(НЕДЕЛЯ, &НачалоПериода, &КонецПериода)";
```

В запросе получаются обороты продаж по товарам за период с периодичностью *НЕДЕЛЯ* и рассчитываются общие итоги, а также итоги по полям *Товар* и *Период*. Причем если за какой-то период (неделю) не было продаж, колонка с этим периодом также попадет в результат запроса, благодаря использованию конструкции *ПЕРИОДАМИ()* при описании итогов по полю *Период*.

подробнее

Этот вопрос подробно рассматривался в разделе «Вывод итогов по периодам с заданной периодичностью».

Этот запрос может служить источником набора данных для получения отчета с помощью

системы компоновки данных. Следует иметь в виду, что в системе компоновки данных секция описания итогов не используется, а дополнение периодов указывается в настройках отчета для группировки *Период*.

Можно вывести кросс-отчет также средствами встроенного языка.

подробнее

Этот вопрос подробно рассматривался в разделе «Создание кросс-отчета».

Этот пример можно посмотреть в демонстрационной конфигурации «Учет движения средств», прилагающейся к книге, в обработке *Продажи номенклатуры*. Поясним только, что при выводе кросс-отчета в табличный документ нужно учитывать следующие моменты.

Во-первых, при обходе выборки результата запроса для группировки *Период* нужно использовать третий параметр метода *Выбрать()* со значением *"ВСЕ"*. Если при получении выборки третий параметр опустить, то в нее не попадут дополненные периоды, то есть те периоды, за которые не было продаж (листинг 3.33).

Листинг 3.33. Заполнение данными табличного документа

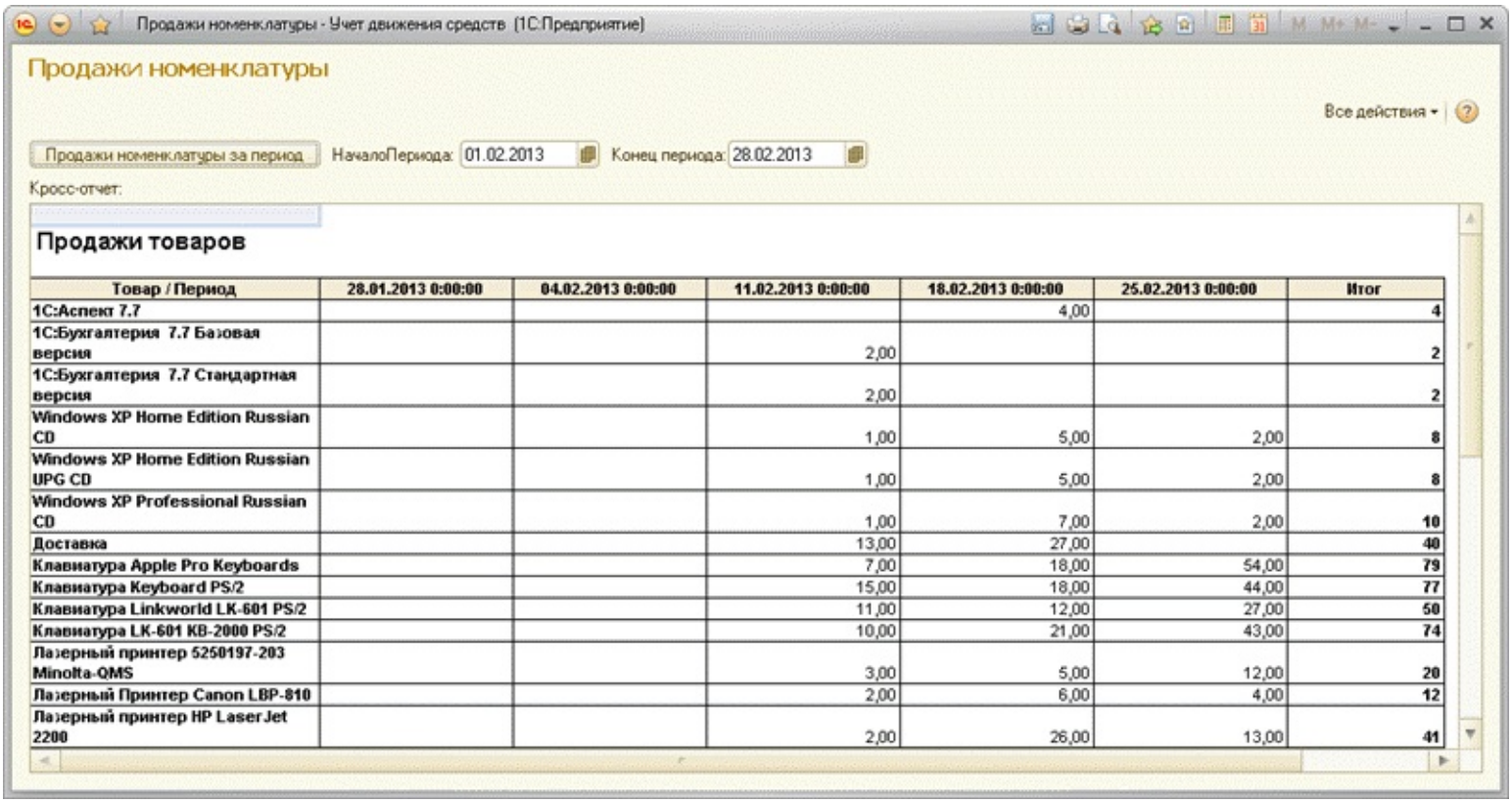
```
ВыборкаПериодИтог = РезультатЗапроса.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам,
"Период", "ВСЕ");
```

Во-вторых, при обходе выборки группировки *Товар* для вложенной в нее группировки *Период* нужно также использовать третий параметр метода *Выбрать()* со значением *"ВСЕ"*. Иначе те периоды, за которые не было продаж конкретного товара, не попадут в выборку, и таблица «поплывет» (листинг 3.34).

Листинг 3.34. Заполнение данными табличного документа

```
ВыборкаПериод = ВыборкаТовар.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам, "Период",
"ВСЕ");
```

В результате мы получим данные о продажах товаров за указанный период с периодичностью *НЕДЕЛЯ* в виде кросс-отчета с дополнением периодов продаж (рис. 3.42).



Товар / Период	28.01.2013 0:00:00	04.02.2013 0:00:00	11.02.2013 0:00:00	18.02.2013 0:00:00	25.02.2013 0:00:00	Итог
1С:Аспект 7.7				4,00		4
1С:Бухгалтерия 7.7 Базовая версия			2,00			2
1С:Бухгалтерия 7.7 Стандартная версия			2,00			2
Windows XP Home Edition Russian CD			1,00	5,00	2,00	8
Windows XP Home Edition Russian UPG CD			1,00	5,00	2,00	8
Windows XP Professional Russian CD			1,00	7,00	2,00	10
Доставка			13,00	27,00		40
Клавиатура Apple Pro Keyboards			7,00	18,00	54,00	79
Клавиатура Keyboard PS/2			15,00	18,00	44,00	77
Клавиатура Linkworld LK-601 PS/2			11,00	12,00	27,00	50
Клавиатура LK-601 KB-2000 PS/2			10,00	21,00	43,00	74
Лазерный принтер 5250197-203 Minolta-QMS			3,00	5,00	12,00	20
Лазерный Принтер Canon LBP-810			2,00	6,00	4,00	12
Лазерный принтер HP LaserJet 2200			2,00	26,00	13,00	41

Рис. 3.42. Кросс-отчет по продажам товаров

Получение остатков и оборотов

Для решения прикладных задач управления ресурсами предприятий часто требуется одновременно оценить не только состояние показателей остатков, но и увидеть в динамике, как эти показатели были получены.

Например, для контроля наличия товара на складах предприятия пользователю часто необходимо видеть картину и остатков, и оборотов за определенный период. То есть информацию и о начальном остатке значения ресурса (*КоличествоНачальныйОстаток*), и о том, какой за указанный промежуток времени был оборот данного ресурса (*КоличествоОборот*), и о том, как этот оборот был получен (*КоличествоПриход*, *КоличествоРасход*), и о том, какое конечное значение ресурса после этого получилось (*КоличествоКонечныйОстаток*).

Для эффективного решения подобных задач в системе «1С:Предприятие» реализована виртуальная таблица остатков и оборотов *ОстаткиИОбороты()* для регистра накопления остатков. Эта таблица позволяет получать итоговые значения и остатков, и оборотов ресурсов за временной интервал. Итоговые значения могут быть получены в разрезе комбинаций значений измерений и/или в развороте по дополнительной периодичности. Итоги получаются только по активным записям.

Виртуальная таблица *ОстаткиИОбороты* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;

- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*), с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя ресурса>КонечныйОстаток* – поле, содержащее конечные остатки ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *КонечныйОстаток*;
- *<Имя ресурса>НачальныйОстаток* – поле, содержащее начальные остатки ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *НачальныйОстаток*;
- *<Имя ресурса>Оборот* – поле, содержащее обороты ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Оборот*. Оборот подсчитывается как сумма всех движений *Приход* со знаком «+» (плюс) и *Расход* со знаком «-» (минус);
- *<Имя ресурса>Приход* – поле, содержащее сумму всех движений типа *Приход* по ресурсу регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Приход*;
- *<Имя ресурса>Расход* – поле, содержащее сумму всех движений типа *Расход* по ресурсу регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации. Имена полей соответствуют именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Расход*;
- *НомерСтроки* – имеет тип *Число*. Существует только в случаях, если указано значение параметра виртуальной таблицы остатков и оборотов *Периодичность*: *Запись*. Содержит значение поля *НомерСтроки* записи движения регистра;
- *Период* – имеет тип *Дата*. Существует только в случаях, если указано значение параметра виртуальной таблицы остатков и оборотов *Периодичность*: *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Секунда*, *Минута*, *Час*, *Регистратор* или *Запись*. Данное поле содержит начальную дату и время периода, к которому относятся итоги регистра. Это поле не может быть использовано в условии отбора записей;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Существует только в случаях, если указано значение параметра виртуальной таблицы остатков и оборотов *Периодичность*: *Регистратор* или *Запись*. Данное поле содержит ссылку на документ-регистратор, к которому относятся итоги регистра. Это поле не может быть использовано в условии отбора записей.

При построении этой виртуальной таблицы могут использоваться параметры, с помощью которых настраивается или уточняется состав получаемых данных:

- *НачалоПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Используется для указания начала периода расчета итогов. Значение этого параметра по умолчанию включается в период расчета итогов. Если параметр не задан, итоги рассчитываются с самой первой записи;
- *КонецПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Используется для указания конца периода расчета итогов. Значение этого параметра по умолчанию включается в период расчета итогов. Если параметр не задан, итоги рассчитываются по самую последнюю запись;
- *Периодичность* – содержит конструкцию языка запросов. Задается одним из следующих вариантов: *Период*, *Запись*, *Регистратор*, *Секунда*, *Минута*, *Час*, *День*, *Неделя*, *Декада*, *Месяц*, *Квартал*, *Полугодие*, *Год*, *Авто*. Используется для указания дополнительного разворота итогов по периодичности. При указании значения *Период* итоги представляются за весь период (с *НачалоПериода* по *КонецПериода*) без дополнительных разворотов. В остальных случаях – с разворотом по записям, регистраторам, неделям, месяцам, кварталам и т. п. соответственно. Значение по умолчанию – *Период*;
- *МетодДополнения* – конструкция языка запросов. Задается одним из следующих вариантов: *Движения*, *ДвиженияИГраницыПериода*. Используется для указания включения граничных периодов в состав выдаваемых виртуальной таблицей записей. Если выбирается вариант *Движения*, то граничные периоды не будут включаться (будут выданы записи только за те периоды, когда имели место движения). Если выбирается вариант *ДвиженияИГраницыПериода*, то граничные периоды будут включаться независимо от наличия или отсутствия движений в рамках интервала между *НачалоПериода* и *КонецПериода*. Значение по умолчанию: *ДвиженияИГраницыПериода*;
- *Условие* – содержит конструкцию языка запросов – условие. Строится по полям регистра накопления (или по подчиненным им полям). Используется для ограничения состава исходных записей, по которым при построении виртуальной таблицы остатков и оборотов будут получаться итоги. То есть условие будет применяться к исходным записям, а не к уже отобранным. Если параметр не указан, для получения итогов будут анализироваться все активные записи регистра.

Для построения виртуальной таблицы всегда используются данные и таблицы итогов, и таблицы движений регистра из базы данных. Это довольно ресурсоемкая операция, поэтому, если данные можно получить только из таблицы остатков или только из таблицы оборотов, то лучше лишний раз не использовать таблицу остатков и оборотов.

Рассмотрим пример получения остатков и оборотов из регистра накопления остатков *ТоварыНаСкладах*. Предположим, требуется получить данные о начальных и конечных остатках, а также об оборотах, приходе и расходе определенного товара по складам за период с заданной периодичностью. Это можно сделать с помощью следующего запроса

(листинг 3.35).

Листинг 3.35. Вывод остатков и оборотов товаров на складах за период с заданной периодичностью

```
ВЫБРАТЬ
  ТоварыНаСкладахОстаткиИОбороты.Период,
  ТоварыНаСкладахОстаткиИОбороты.Номенклатура,
  ТоварыНаСкладахОстаткиИОбороты.Склад,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоНачальныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоПриход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоРасход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоОборот,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоКонечныйОстаток
ИЗ
  РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты(
    &НачалоПериода,
    КОНЕЦПЕРИОДА(&КонецПериода, ДЕНЬ),
    День,
    Движения,
    Номенклатура = &Товар
    )
  КАК ТоварыНаСкладахОстаткиИОбороты
```

В данном запросе в виртуальную таблицу оборотов мы передаем границы интервала, за который нужно получить остатки и обороты, и товар, по которому отбираются исходные записи регистра накопления. Чтобы получить конец дня, мы используем функцию языка запросов *КОНЕЦПЕРИОДА()*. В параметре *Периодичность* мы задаем значение *День*, т. е. указываем, что нам требуется детализировать данные до дня.

При построении виртуальной таблицы остатков и оборотов данные регистра накопления остатков *ТоварыНаСкладах* группируются по измерениям *Номенклатура* и *Склад*. В итоге мы получаем остатки и обороты по ресурсу *Количество* для заданного товара в разрезе складов с разворотом по дням.

В результате для товара *Ноутбук Rover Computers Explorer* за период *01.02.2013 – 28.02.2013* мы получим следующий результат (рис. 3.43).

Запрос: РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты((Записей в результате: 8)

Период	Номенклатура	Склад	КоличествоНачальныйОстаток
03.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Главный	2
15.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Главный	3
01.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	
03.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	1
18.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	5
22.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	4
24.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	2
26.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	1

КоличествоНачальныйОстаток	КоличествоПриход	КоличествоРасход	КоличествоОборот	КоличествоКонечныйОстаток
2	1		1	3
3		2	-2	1
	1		1	1
1	4		4	5
5		1	-1	4
4		2	-2	2
2		1	-1	1
1		1	-1	

Рис. 3.43. Вывод остатков и оборотов товаров на складах за период с заданной периодичностью

В приведенном примере (см. листинг 3.35) в параметрах виртуальной таблицы был указан метод дополнения *Движения*, поэтому в результате были получены итоги только за те периоды, когда имели место движения. По умолчанию параметр виртуальной таблицы *МетодДополнения* имеет значение *ДвиженияИГраницыПериода*. Поэтому в следующем примере (листинг 3.36) граничные периоды будут включаться независимо от наличия или отсутствия движений в рамках интервала между *НачалоПериода* и *КонецПериода*.

Листинг 3.36. Вывод остатков и оборотов товаров на складах за период с заданной периодичностью

```
ВЫБРАТЬ
  ТоварыНаСкладахОстаткиИОбороты.Период,
  ТоварыНаСкладахОстаткиИОбороты.Номенклатура,
  ТоварыНаСкладахОстаткиИОбороты.Склад,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоНачальныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоПриход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоРасход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоОборот,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоКонечныйОстаток
ИЗ
  РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты(
    &НачалоПериода,
    КОНЕЦПЕРИОДА(&КонецПериода, ДЕНЬ),
    День,
    ,
    Номенклатура = &Товар
    )
  КАК ТоварыНаСкладахОстаткиИОбороты
```

В результате для товара *Ноутбук Rover Computers Explorer* за период *01.02.2013 – 28.02.2013* мы получим следующий результат (рис. 3.44).

Запрос: РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты((Записей в результате: 10)

Период	Номенклатура	Склад	КоличествоНачальныйОстаток
01.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Главный	2
03.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Главный	2
15.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Главный	3
28.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Главный	1
01.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	
03.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	1
18.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	5
22.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	4
24.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	2
26.02.2013 0:00:00	Ноутбук Rover Computers Explorer	Склад отдела продаж	1

КоличествоПриход	КоличествоРасход	КоличествоОборот	КоличествоКонечныйОстаток
			2
1		1	3
	2	-2	1
			1
1		1	1
4		4	5
	1	-1	4
	2	-2	2
	1	-1	1
	1	-1	

Рис. 3.44. Вывод остатков и оборотов товаров на складах за период с заданной периодичностью

Мы видим, что для комбинации товара *Ноутбук Rover Computers Explorer* и склада *Главный* результат запроса дополнен граничными записями, содержащими начальный и конечный остаток товара на складе, хотя движения за эти даты отсутствуют. А для комбинации товара *Ноутбук Rover Computers Explorer* и склада *Склад отдела продаж* результат запроса остался прежним, т. к. на границы периода начального и конечного остатка товара на этом складе не было.

При использовании виртуальной таблицы остатков и оборотов итоги по остаткам и оборотам рассчитываются, включая граничные периоды. Поэтому все особенности использования параметров *НачалоПериода* и *КонецПериода* аналогичны виртуальной таблице оборотов.

Расчет итогов по полям остатка

В заключение рассмотрим отдельно вопрос расчета итогов по полям остатков. Для расчета итогов по полю остатка достаточно получить его в списке выборки запроса и указать необходимость расчета итога по этому полю. Но дело в том, что с прикладной точки зрения остатки не всегда могут быть получены простым суммированием, а должны рассчитываться по специальному алгоритму. Рассмотрим эти варианты.

Например, для следующего запроса (листинг 3.37) итоги будут получены простым суммированием данных соответствующих записей, т. к. в нем не используется периодичность и не выводится поле *Период*.

Листинг 3.37. Пример получения остатков и оборотов регистра накопления без использования поля «Период»

```
ВЫБРАТЬ
  ТоварыНаСкладахОстаткиИОбороты.Номенклатура КАК Номенклатура,
```

```
  ТоварыНаСкладахОстаткиИОбороты.Склад,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоНачальныйОстаток КАК
  КоличествоНачальныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоКонечныйОстаток КАК КоличествоКонечныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоОборот КАК КоличествоОборот,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоПриход КАК КоличествоПриход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоРасход КАК КоличествоРасход
ИЗ
  РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты(,,,, Номенклатура В ИЕРАРХИИ(&Товар))
  КАК ТоварыНаСкладахОстаткиИОбороты
ИТОГИ
  СУММА(КоличествоНачальныйОстаток),
  СУММА(КоличествоКонечныйОстаток),
  СУММА(КоличествоОборот),
  СУММА(КоличествоПриход),
  СУММА(КоличествоРасход)
ПО
  Номенклатура
```

В итоге для группы товаров *Ноутбуки* мы получим следующий результат (рис. 3.45).

Запрос: РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты(,,,, (Записей в результате: 6)

Номенклатура	Склад	КоличествоНачальныйОстаток	КоличествоКонечныйОстаток
Ноутбук Rover Computers Explorer		0	1
Ноутбук Rover Computers Explorer	Главный	0	1
Ноутбук Rover Computers Explorer	Склад отдела продаж	0	0
Ноутбук Rover Computers Navigator KT7		0	4
Ноутбук Rover Computers Navigator KT7	Главный	0	0
Ноутбук Rover Computers Navigator KT7	Склад отдела продаж	0	4

КоличествоНачальныйОстаток	КоличествоКонечныйОстаток	КоличествоОборот	КоличествоПриход	КоличествоРасход
0	1	1	8	7
0	1	1	3	2
0	0	0	5	5
0	4	4	11	7
0	0	0	3	3
0	4	4	8	4

Рис. 3.45. Пример получения остатков и оборотов регистра накопления без использования поля «Период»

Обратите внимание, что записи комбинации товара *Ноутбук Rover Computers Explorer* и склада *Склад отдела продаж* и комбинации товара *Ноутбук Rover Computers Navigator KT7* и склада *Главный* все равно попадают в результат запроса, несмотря на нулевые остатки, т. к. по ним есть обороты – приход и расход.

Но если нам требуется разворот по периодичности и мы выводим в запросе поле *Период* (листинг 3.38), то будет применен специальный алгоритм расчета итогов остатков.

Листинг 3.38. Пример получения остатков и оборотов регистра накопления с использованием поля «Период»

```
ВЫБРАТЬ
  ТоварыНаСкладахОстаткиИОбороты.Номенклатура КАК Номенклатура,
  ТоварыНаСкладахОстаткиИОбороты.Склад,
  ТоварыНаСкладахОстаткиИОбороты.Период КАК Период,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоНачальныйОстаток КАК
  КоличествоНачальныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоКонечныйОстаток КАК КоличествоКонечныйОстаток,
```

```
  ТоварыНаСкладахОстаткиИОбороты.КоличествоОборот КАК КоличествоОборот,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоПриход КАК КоличествоПриход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоРасход КАК КоличествоРасход
ИЗ
  РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты( , , Месяц, , Номенклатура В
  ИЕРАРХИИ(&Товар) ) КАК ТоварыНаСкладахОстаткиИОбороты
ИТОГИ
  СУММА(КоличествоНачальныйОстаток),
  СУММА(КоличествоКонечныйОстаток),
  СУММА(КоличествоОборот),
  СУММА(КоличествоПриход),
  СУММА(КоличествоРасход)
ПО
  Номенклатура
```

В данном случае для расчета итогов по полям остатков (*КоличествоНачальныйОстаток*, *КоличествоКонечныйОстаток*) будет использоваться следующий алгоритм: записи упорядочиваются по полям первого уровня (не полученные через точку от другого поля из списка выборки, в нашем случае – *Номенклатура* и *Склад*), а затем – в хронологическом порядке. После этого осуществляется обход записей. Первые записи для комбинации полей первого уровня используются для суммирования начального остатка, последние – для расчета конечного остатка.

В результате для группы товаров *Ноутбуки* мы получим следующий результат (рис. 3.46).

Запрос: РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты(,, (Записей в результате: 8)

Номенклатура	Склад	Период	КоличествоНачальныйОстаток	КоличествоКонечныйОстаток
Ноутбук Rover Computers Explorer			0	1
Ноутбук Rover Computers Explorer	Главный	01.01.2013 0:00:00	0	2
Ноутбук Rover Computers Explorer	Главный	01.02.2013 0:00:00	2	1
Ноутбук Rover Computers Explorer	Склад отдела продаж	01.02.2013 0:00:00	0	0
Ноутбук Rover Computers Navigator KT7			0	4
Ноутбук Rover Computers Navigator KT7	Главный	01.01.2013 0:00:00	0	1
Ноутбук Rover Computers Navigator KT7	Главный	01.02.2013 0:00:00	1	0
Ноутбук Rover Computers Navigator KT7	Склад отдела продаж	01.02.2013 0:00:00	0	4

КоличествоНачальныйОстаток	КоличествоКонечныйОстаток	КоличествоОборот	КоличествоПриход	КоличествоРасход
0	1	1	8	7
0	2	2	2	0
2	1	-1	1	2
0	0	0	5	5
0	4	4	11	7
0	1	1	1	0
1	0	-1	2	3
0	4	4	8	4

Рис. 3.46. Пример получения остатков и оборотов регистра накопления с использованием поля «Период»

В случае же, если среди итогов есть группировка *Период* (листинг 3.39), алгоритм расчета итогов остатков еще более сложный.

Листинг 3.39. Пример получения остатков и оборотов регистра накопления с использованием поля «Период» и группировкой по периоду

```
ВЫБРАТЬ
  ТоварыНаСкладахОстаткиИОбороты.Номенклатура КАК Номенклатура,
```

```
  ТоварыНаСкладахОстаткиИОбороты.Период КАК Период,
  ТоварыНаСкладахОстаткиИОбороты.Склад,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоНачальныйОстаток КАК
  КоличествоНачальныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоКонечныйОстаток КАК КоличествоКонечныйОстаток,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоОборот КАК КоличествоОборот,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоПриход КАК КоличествоПриход,
  ТоварыНаСкладахОстаткиИОбороты.КоличествоРасход КАК КоличествоРасход
ИЗ
  РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты( , , Месяц, , Номенклатура В
  ИЕРАРХИИ(&Товар) ) КАК ТоварыНаСкладахОстаткиИОбороты
ИТОГИ
  СУММА(КоличествоНачальныйОстаток),
  СУММА(КоличествоКонечныйОстаток),
  СУММА(КоличествоОборот),
  СУММА(КоличествоПриход),
  СУММА(КоличествоРасход)
ПО
  Номенклатура,
  Период
```

В данном случае для расчета итогов по полям остатков будет использоваться следующий алгоритм:

- расчет итогов для группировки (группировок) перед группировкой *Период* (в нашем примере это расчет для группировки *Номенклатура*) выполняется по тому же алгоритму, что в предыдущем примере;
- для расчета итогов по группировке *Период* записи упорядочиваются по периоду. После чего обходятся записи для каждой даты, рассчитывается сумма для различных комбинаций значений полей первого уровня, по которым еще не была получена выборка для даты, больше либо равной рассчитываемой для начального остатка и меньше либо равной рассчитываемой для конечного остатка.

В результате для группы товаров *Ноутбуки* мы получим следующий результат (рис. 3.47).

Запрос: РегистрНакопления.ТоварыНаСкладах.ОстаткиИОбороты(,, (Записей в результате: 12)

Номенклатура	Период	Склад	КоличествоНачальныйОстаток	КоличествоКонечныйОстаток
Ноутбук Rover Computers Explorer			0	1
Ноутбук Rover Computers Explorer	01.01.2013 0:00:00		0	2
Ноутбук Rover Computers Explorer	01.01.2013 0:00:00	Главный	0	2
Ноутбук Rover Computers Explorer	01.02.2013 0:00:00		2	1
Ноутбук Rover Computers Explorer	01.02.2013 0:00:00	Главный	2	1
Ноутбук Rover Computers Explorer	01.02.2013 0:00:00	Склад отдела продаж	0	0
Ноутбук Rover Computers Navigator KT7			0	4
Ноутбук Rover Computers Navigator KT7	01.01.2013 0:00:00		0	1
Ноутбук Rover Computers Navigator KT7	01.01.2013 0:00:00	Главный	0	1
Ноутбук Rover Computers Navigator KT7	01.02.2013 0:00:00		1	4
Ноутбук Rover Computers Navigator KT7	01.02.2013 0:00:00	Главный	1	0
Ноутбук Rover Computers Navigator KT7	01.02.2013 0:00:00	Склад отдела продаж	0	4

КоличествоНачальныйОстаток	КоличествоКонечныйОстаток	КоличествоОборот	КоличествоПриход	КоличествоРасход
0	1	1	8	7
0	2	2	2	0
0	2	2	2	0
2	1	-1	6	7
2	1	-1	1	2
0	0	0	5	5
0	4	4	11	7
0	1	1	1	0
0	1	1	1	0
1	4	3	10	7
1	0	-1	2	3
0	4	4	8	4

Рис. 3.47. Пример получения остатков и оборотов регистра накопления с использованием поля «Период» и группировкой по периоду

Заметьте, что «особыми» алгоритмами были рассчитаны только поля остатков, поля же оборотов рассчитываются простым суммированием в рамках требуемых группировок.

В общем случае, когда среди группировочных полей итогов есть и предшествующие полю *Период*, и находящиеся после него, алгоритм расчета итогов включает в себя следующие этапы:

- расчет итогов для группировок, предшествующих группировке *Период*;
- расчет итогов для группировки *Период*;
- расчет итогов для группировок после группировки *Период*. Производится как для обычных полей, то есть простым суммированием.

Следует иметь в виду, что в любом случае применение специальных алгоритмов происходит уже после того, как виртуальная таблица остатков и оборотов была получена и сохранена во временной таблице базы данных.

Обратим внимание еще на ряд важных особенностей при расчете итогов по полям остатка:

- Для того чтобы запрос мог рассчитать итоги по полям остатка, необходимо, чтобы в запросе получались оба значения остатка за период (начальный и конечный). В случае, если в запросе получается только один остаток и по нему ведется расчет итога, система неявно добавит в запрос для получения данных поле парного остатка.
- Если поле-остаток используется в выражении, программа пытается найти для начального остатка аналогичное выражение, в котором вместо начального остатка используется конечный остаток. В случае, если такое выражение не будет найдено,

система неявно добавит такое поле в исполняемый запрос. Если в одном выражении используется более одного поля начального или конечного остатка, система не воспринимает данное выражение как выражение остатка и будет производить расчет итогов по данному выражению простым суммированием.

- В результате из-за того, что регистратор, по сути, является уточнением периода, при расчете итогов по регистратору необходимо иметь в виду, что получать итоги по регистратору можно только внутри группировки *Период*. Если получить итоги по регистратору до получения итогов по периоду, они могут оказаться некорректными! Аналогично итоги по номеру строки можно получать только внутри группировки по периоду и регистратору.

подробнее

Об особенностях расчета итогов по полям остатка подробнее можно прочитать в статье ИТС «Расчет итогов по полям остатка».

Бухгалтерский учет

Для ведения бухгалтерского учета в системе «1С:Предприятие» используются такие объекты конфигурации, как планы счетов, планы видов характеристик (для хранения видов субконто) и регистры бухгалтерии (для хранения бухгалтерских проводок). В данном разделе мы рассмотрим примеры использования этих объектов для решения различных прикладных задач с помощью языка запросов.

Все примеры, используемые в данном разделе, можно посмотреть в демонстрационной конфигурации «Бухгалтерский учет», которая находится на прилагаемом компакт-диске.

Планы счетов

По свой сути план счетов – это справочник счетов, в разрезе которых ведется бухгалтерский учет. Данные каждого плана счетов хранятся в отдельной таблице информационной базы, доступной посредством запросов. Эта таблица имеет следующий состав полей:

- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный план видов счетов;
- *<Имя признака учета>* – поле, содержащее значения признаков учета счета с именем, заданным в конфигурации. Количество таких полей равно количеству признаков учета, определенных для плана счетов как объекта конфигурации;
- *<Имя реквизита>* – поле, содержащее значения реквизита счета с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для плана счетов как объекта конфигурации;
- *<Имя табличной части>* – поле, содержащее табличные части счета. Имена полей соответствуют именам табличных частей плана счетов, как они заданы в конфигураторе. Имеет тип *РезультатЗапроса*. Результат запроса к табличной части состоит из колонки *НомерСтроки* и колонок с именами, соответствующими именам реквизитов табличной части;
- *Вид* – имеет тип системного перечисления *ВидСчета*. Содержит вид счета (*Активный*, *Пассивный* или *АктивноПассивный*);
- *Виды субконто* – специализированная табличная часть, содержащая перечень видов субконто для счета. Имеет тип *РезультатЗапроса*. Количество строк этой табличной части ограничено значением свойства *МаксКоличествоСубконто* для плана счетов. Каждая строка табличной части *ВидыСубконто* содержит следующие поля: *ВидСубконто*, *Предопределенное*, *ТолькоОбороты* и признаки учета субконто, определенные в конфигураторе;
- *Забалансовый* – имеет тип *Булево*. Содержит признак забалансовости счета;
- *Код* – имеет тип *Строка*. Содержит код счета;
- *Наименование* – имеет тип *Строка*. Содержит наименование счета;
- *ПометкаНаУдаление* – имеет тип *Булево*. Содержит признак пометки на удаление счета;
- *Порядок* – имеет тип *Строка*. Содержит строку с порядком, по которому можно выполнить упорядочивание списка счетов. Если при конфигурировании у плана счетов установлено свойство *Автопорядок по коду*, то при формировании результатов запроса будет отмечено, что сортировка выполняется по полю *Код*, однако на самом деле будет использоваться сортировка по полю *Порядок*;
- *Предопределенный* – имеет тип *Булево*. Содержит признак того, что данный счет определен в метаданных и над ним нельзя производить некоторые операции;
- *Представление* – виртуальное поле, не хранится в информационной базе. Содержит

представление счета;
- *Родитель* – содержит ссылку на родителя счета;
- *Ссылка* – содержит ссылку на счет.

Рассмотрим некоторые примеры получения данных из плана счетов.

В нашей демонстрационной конфигурации существует план счетов *Основной план счетов*, который хранит информацию о составе счетов, используемых для ведения бухучета на предприятии (рис. 3.48).

Рис. 3.48. План счетов

План счетов является иерархическим и позволяет создавать неограниченное число счетов и уровней вложенности. Наш план счетов имеет два уровня вложенности – счета первого уровня *Активы*, *Обязательства* и *Капитал* и несколько субсчетов к счетам *Активы* и *Обязательства*.

Следует иметь в виду, что на практике иерархичность счетов и иерархичность кодов счетов могут различаться, хотя в данном случае (см. рис. 3.48) они совпадают. При написании запросов нужно учитывать, что иерархичность счетов определяется по полю *Родитель*, а не по полю *Код*.

Обратите внимание, как заполнены строковые поля *Порядок* и *Код* в плане счетов (см. рис. 3.48). Например, у счета *Касса* код введен как *1.1*, а порядок как *1. 1* (1 + точка + пробел + 1). Если сортировать счета по коду, то счет *Новый счет* будет следовать за счетом *Касса*, а если сортировать счета по порядку, то он будет занимать свое «законное» место за счетом *Контрагенты* с кодом *1.5*. Так происходит потому, что у плана счетов установлено свойство *Автопорядок по коду*. И даже если в списке счетов установить упорядочивание по полю *Код*, на самом деле сортировка списка будет выполняться по полю *Порядок*.

Подобная же сортировка (по полю *Порядок*) будет выполняться и в запросах при указании в предложении *УПОРЯДОЧИТЬ ПО* в качестве поля сортировки поля *Код* (листинг 3.40).

Листинг 3.40. Сортировка счетов по коду с соблюдением автопорядка

```
ВЫБРАТЬ
  ОсновнойПланСчетов.Код,
  ОсновнойПланСчетов.Порядок,
  ОсновнойПланСчетов.Наименование,
  ОсновнойПланСчетов.Вид,
  ОсновнойПланСчетов.Количественный,
  ОсновнойПланСчетов.Забалансовый,
  ОсновнойПланСчетов.Предопределенный,
  ОсновнойПланСчетов.Родитель
ИЗ
  ПланСчетов.ОсновнойПланСчетов КАК ОсновнойПланСчетов

УПОРЯДОЧИТЬ ПО
  Код
```

Результат выполнения данного запроса представлен на рис. 3.49.

Запрос: ПланСчетов.ОсновнойПланСчетов (Записей в результате: 11)

Код	Порядок	Наименование	Вид	Количественный	Забалансовый	Предопределенный	Родитель
1	1	Активы	Активный	Нет	Нет	Да	
1.1	1. 1	Касса	Активный	Нет	Нет	Да	Активы
1.2	1. 2	Покупатели	Активный	Нет	Нет	Да	Активы
1.3	1. 3	Товары	Активный	Да	Нет	Да	Активы
1.4	1. 4	Материалы	Активный	Да	Нет	Да	Активы
1.5	1. 5	Контрагенты	Активный/Пассивный	Нет	Нет	Да	Активы
1.11	1.11	Новый счет	Активный	Нет	Нет	Нет	Активы
2	2	Обязательства	Пассивный	Нет	Нет	Да	
2.1	2. 1	Сотрудники	Пассивный	Нет	Нет	Да	Обязательства
2.2	2. 2	Поставщики	Пассивный	Нет	Нет	Да	Обязательства
3	3	Капитал	Пассивный	Нет	Нет	Да	

Рис. 3.49. Список плана счетов

Поле счета *Вид* имеет тип системного перечисления *ВидСчета*. Предположим, нам необходимо вывести все активно-пассивные счета из плана счетов. Это можно сделать с помощью следующего запроса (листинг 3.41).

Листинг 3.41. Вывод активно-пассивных счетов

```
ВЫБРАТЬ
  ОсновнойПланСчетов.Код,
  ОсновнойПланСчетов.Наименование
ИЗ
  ПланСчетов.ОсновнойПланСчетов КАК ОсновнойПланСчетов
ГДЕ
  ОсновнойПланСчетов.Вид = ЗНАЧЕНИЕ(ВидСчета.АктивноПассивный)

УПОРЯДОЧИТЬ ПО
  Код
```

Результат выполнения данного запроса представлен на рис. 3.50.

Запрос: ПланСчетов.ОсновнойПланСчетов (Записей в результате: 1)

Код	Наименование
1.5	Контрагенты

Рис. 3.50. Список активно-пассивных счетов

В листинге 3.40 выбираются значения признаков учета *Количественный* и *Забалансовый* для счетов (см. рис. 3.49). Эти значения имеют булевый тип. Таким образом, при помощи следующего запроса мы можем получить список тех счетов, на которых ведется количественный учет (листинг 3.42).

Листинг 3.42. Вывод счетов, на которых ведется количественный учет

```
ВЫБРАТЬ
  ОсновнойПланСчетов.Код,
  ОсновнойПланСчетов.Наименование
ИЗ
  ПланСчетов.ОсновнойПланСчетов КАК ОсновнойПланСчетов
ГДЕ
  ОсновнойПланСчетов.Количественный = ИСТИНА

УПОРЯДОЧИТЬ ПО
  Код
```

Результат выполнения данного запроса представлен на рис. 3.51.

Запрос: ПланСчетов.ОсновнойПланСчетов (Записей в результате: 2)

Код	Наименование
1.3	Товары
1.4	Материалы

Рис. 3.51. Список счетов, на которых ведется количественный учет

Поскольку план счетов поддерживает иерархию, то с ним можно выполнять любые операции с иерархическими данными. Например, с помощью следующего запроса можно получить список счетов, находящихся в иерархии заданного счета (листинг 3.43).

Листинг 3.43. Вывод счетов в иерархии заданного счета

```
ВЫБРАТЬ
  ОсновнойПланСчетов.Код,
  ОсновнойПланСчетов.Наименование
ИЗ
  ПланСчетов.ОсновнойПланСчетов КАК ОсновнойПланСчетов
ГДЕ
  ОсновнойПланСчетов.Ссылка В ИЕРАРХИИ (&Счет)

УПОРЯДОЧИТЬ ПО
  Код
```

Если мы выберем в качестве значения параметра *Счет* счет *Обязательства*, то получим следующий результат (рис. 3.52).

Запрос: ПланСчетов.ОсновнойПланСчетов (Записей в результате: 3)

Код	Наименование
2	Обязательства
2.1	Сотрудники
2.2	Поставщики

Рис. 3.52. Список счетов в иерархии заданного счета

Планы видов характеристик – виды субконто

Теперь рассмотрим, как реализуется ведение аналитического учета на счетах. Для организации аналитического учета используются такие понятия, как *вид субконто* и *субконто*. Субконто (например, контрагент *Иванов*) – это один из объектов аналитического учета, а вид субконто объединяет однородные объекты аналитического учета (например, справочник *Контрагенты*) вместе.

Для хранения списка видов субконто в нашей демонстрационной конфигурации используется план видов характеристик *Виды субконто* (рис. 3.53).

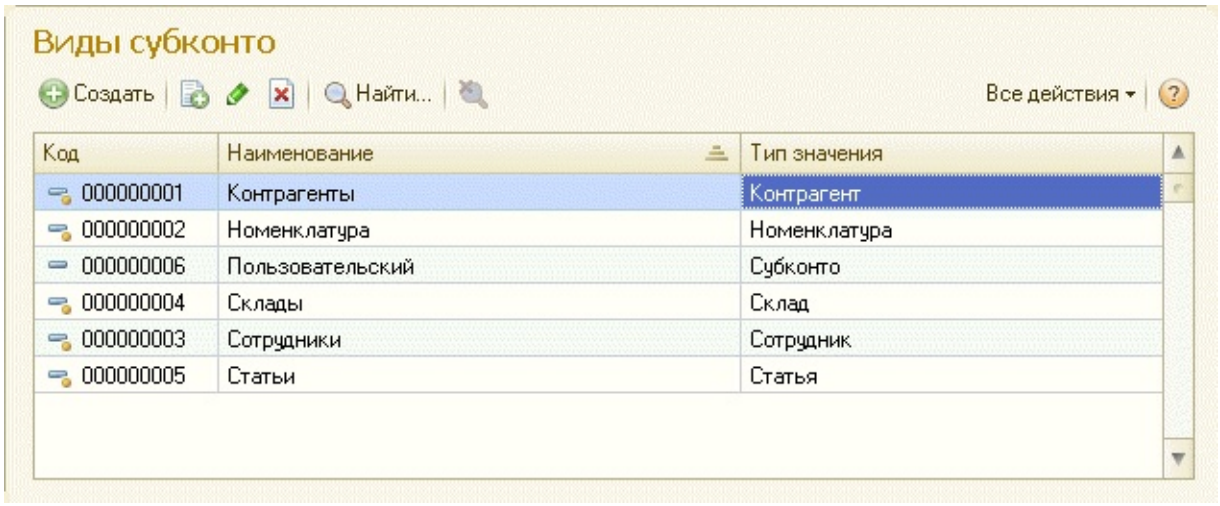


Рис. 3.53. План видов характеристик «Виды субконто»

подробнее

Про получение данных с помощью языка запросов из плана видов характеристик подробнее рассказано в разделе «Планы видов характеристик».

В применении к бухгалтерской специфике вид субконто соответствует виду характеристики, субконто – отдельной характеристике, а план видов характеристик объединяет все виды субконто вместе.

В свойствах нашего плана счетов задано максимальное количество субконто на счете – *2* и указан объект метаданных, описывающий план видов характеристик – *ВидыСубконто*. В этом плане видов характеристик содержатся виды субконто, используемые в данном плане счетов. Таким образом, добавляются два дополнительных аналитических разреза учета для каждого счета.

В результате подобной привязки плана счетов и плана видов характеристик для каждого счета создается специализированная табличная часть *ВидыСубконто*, содержащая не более двух видов субконто (разрезов аналитики), учитываемых на счете. Каждая из строк

табличной части содержит ссылку на вид характеристики плана видов характеристик выбранного в свойстве плана счетов *Виды субконто* и дополнительные свойства булевого типа: *Предопределенное*, *ТолькоОбороты* и признаки учета субконто (*Количественный*, *Суммовой*), введенные в конфигураторе (рис. 3.54).

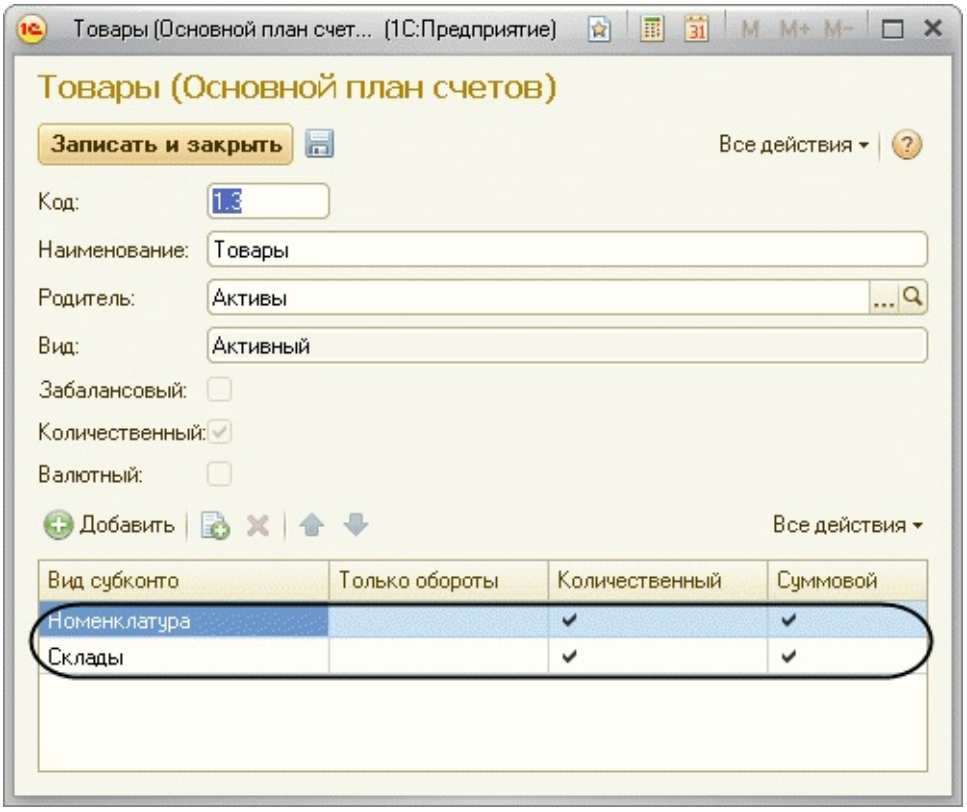


Рис. 3.54. Свойства счета «Товары»

ВидСубконто – свойство типа *ПланВидовХарактеристикСсылка.<имя>*, позволяющее получить доступ к свойствам вида характеристики плана видов характеристик. Это поле задает вид субконто, в разрезе которого будет вестись аналитический учет.

Выведем теперь свойства видов субконто, учитываемых на определенном счете, с помощью запроса к плану счетов (листинг 3.44).

Листинг 3.44. Свойства видов субконто, учитываемых на заданном счете

```
&НаСервереБезКонтекста
Процедура ВывестиСвойстваВидаСубконто(Счет)

  Сообщение = Новый СообщениеПользователю();

  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |        ОсновнойПланСчетов.ВидыСубконто
    |ИЗ
    |        ПланСчетов.ОсновнойПланСчетов КАК ОсновнойПланСчетов
    |ГДЕ
    |        ОсновнойПланСчетов.Ссылка = &Счет";

  Запрос.УстановитьПараметр("Счет", Счет);

  Результат = Запрос.Выполнить();
```

```
  Выборка = Результат.Выбрать();

  Пока Выборка.Следующий() Цикл
    // выборка видов субконто
    ВидыСубконтоВыборка = Выборка.ВидыСубконто.Выбрать();

    Пока ВидыСубконтоВыборка.Следующий() Цикл
      Вид = ВидыСубконтоВыборка.ВидСубконто;
      ТипЗначения = Вид.ТипЗначения;
      Наименование = Вид.Наименование;

      ВведенВКонфигураторе = ВидыСубконтоВыборка.Предопределенное;
      Оборотный = ВидыСубконтоВыборка.ТолькоОбороты;
      Количественный = ВидыСубконтоВыборка.Количественный;
      Суммовой = ВидыСубконтоВыборка.Суммовой;

      Сообщение.Текст = "Вид субконто: " + Наименование +
        ", Предопределенное - " + Строка(ВведенВКонфигураторе) +
        ", Только обороты - " + Строка(Оборотный) +
        ", Количественный - " + Строка(Количественный) +
        ", Суммовой - " + Строка(Суммовой);
      Сообщение.Сообщить();

    КонецЦикла;

  КонецЦикла;

КонецПроцедуры
```

В запросе мы выбираем данные табличной части *ВидыСубконто* для счета, заданного в параметре *Счет*. Эти данные будут иметь тип *РезультатЗапроса*, то есть содержать вложенный результат запроса, сформированный на основе табличной части.

Затем с помощью встроенного языка мы выполняем запрос, и в цикле обхода его результатов получаем вложенную выборку. В цикле обхода этой вложенной выборки мы выводим в окно сообщений данные табличной части, т. е. свойства видов субконто, учитываемых на счете.

Для счета *Товары* мы получим следующий результат (рис. 3.55).

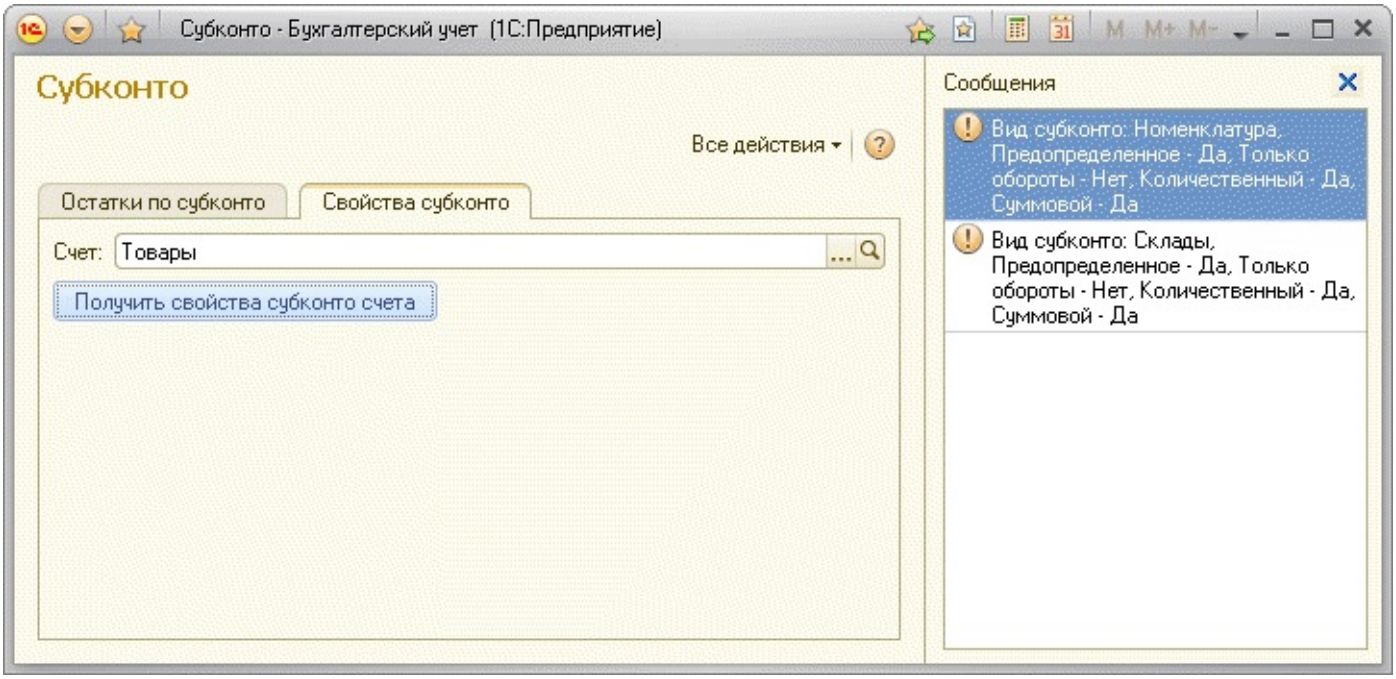


Рис. 3.55. Свойства видов субконто, учитываемых на заданном счете

Этот пример можно посмотреть в демонстрационной конфигурации «Бухгалтерский учет», прилагающейся к книге, в обработке *Субконто*.

К специализированной табличной части *ВидыСубконто* также можно обратиться, как к любой другой табличной части, через точку от имени основной таблицы плана счетов (листинг 3.45).

Листинг 3.45. Свойства видов субконто, учитываемых на заданном счете

```
ВЫБРАТЬ
  *
ИЗ
  ПланСчетов.ОсновнойПланСчетов.ВидыСубконто КАК ОсновнойПланСчетов
ГДЕ
  ОсновнойПланСчетов.Ссылка = &Счет
```

Для счета *Товары* мы получим следующий результат (рис. 3.56).

Запрос: ПланСчетов.ОсновнойПланСчетов.ВидыСубконто (Записей в результате: 2)

Ссылка	НомерСтроки	ВидСубконто	Предопределенное	ТолькоОбороты	Количественный	Суммовой
Товары	1	Номенклатура	Да	Нет	Да	Да
Товары	2	Склады	Да	Нет	Да	Да

Рис. 3.56. Свойства видов субконто, учитываемых на заданном счете

В заключение обратим внимание на такой момент – при установке отборов по счетам нужно всегда использовать или ссылку на счет (полученную, например, путем выбора в диалоге элемента из плана счетов), или неизменяемое имя предопределенного счета, а не использовать код счета.

Например, следующее условие отбора по коду счета (листинг 3.46) даст такой же результат, что и в предыдущем случае, но это условие будет методически

неправильным. Стоит только пользователю изменить код счета, как условие станет неверным.

Листинг 3.46. Неверное условие отбора по счету

```
ГДЕ
  ОсновнойПланСчетов.Ссылка.Код = ""1.3""
```

Регистры бухгалтерии

Регистр бухгалтерии – это основной объект для автоматизации бухгалтерского учета, используемый в связке с планом счетов и видами субконто. На основе информации, хранящейся в этих объектах в информационной базе, строятся виртуальные таблицы, содержащие специальным образом подготовленные данные для формирования разнообразной бухгалтерской отчетности. Речь об этих таблицах пойдет ниже.

Сначала мы рассмотрим, какие данные хранятся в основной таблице (таблице движений) регистра бухгалтерии с поддержкой корреспонденции в информационной базе, доступной с помощью запросов. Основная таблица содержит следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения балансового измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству балансовых измерений, определенных для регистра как объекта конфигурации;
- *<Имя измерения>ДТ* – поле, содержащее значения дебетового небалансового измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству небалансовых измерений, определенных для регистра как объекта конфигурации;
- *<Имя измерения>Кт* – поле, содержащее значения кредитового небалансового измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству небалансовых измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя реквизита>* – поле, содержащее значения реквизита регистра с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для регистра как объекта конфигурации;
- *<Имя ресурса>* – поле, содержащее значения балансового ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству балансовых ресурсов, определенных для регистра как объекта конфигурации;
- *<Имя ресурса>Дт* – поле, содержащее значения дебетового небалансового ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству небалансовых ресурсов, определенных для регистра как объекта конфигурации;

- *<Имя ресурса>Кт* – поле, содержащее значения кредитового небалансового ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству небалансовых ресурсов, определенных для регистра как объекта конфигурации;
- *Активность* – имеет тип *Булево*. Содержит признак активности записи и влияния на получение итогов регистра;
- *Момент времени* – виртуальное поле, не хранится в информационной базе. Содержит объект *МоментВремени* (который включает в себя дату и ссылку на документ-регистратор);
- *Период* – имеет тип *Дата*. Содержит дату записи. Совместно с полями *Регистратор* и *НомерСтроки* определяет положение данной записи на временной оси;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Содержит ссылку на документ, которому подчинена данная запись;
- *НомерСтроки* – имеет тип *Число*. Содержит уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*;
- *СчетДт* – имеет тип *ПланСчетовСсылка.<Имя>*. Содержит ссылку на дебетуемый счет;
- *СчетКт* – имеет тип *ПланСчетовСсылка.<Имя>*. Содержит ссылку на кредитуемый счет.

Основная таблица хранит проводки (движения регистра бухгалтерии) без данных аналитического учета. Для хранения данных аналитического учета предназначена таблица значений субконто. В этой таблице хранится информация о виде и значении каждого субконто проводки. Таблица значений субконто не содержит числовых характеристик и используется в связке (по периоду, регистратору и виду движения) с основной таблицей регистра бухгалтерии для получения виртуальной таблицы *ДвиженийССубконто*, речь о которой пойдет ниже.

Задача любого регистра – учет значений некоторых показателей (ресурсов) в разрезе некоторых измерений (счета проводки, измерения регистра и дополнительные измерения регистра – субконто), с возможностью оставить дополнительную информацию о каждой записи регистра (реквизиты).

В нашей демонстрационной конфигурации существует регистр бухгалтерии *Основной регистр бухгалтерии*, который использует *Основной план счетов* и поддерживает корреспонденцию. Данный регистр имеет измерения *Организация* и *Валюта* и ресурсы *Сумма*, *Количество*, *ВалютнаяСумма* и *СуммаХолдинга*, а также реквизит *Содержание* для хранения дополнительной информации о содержании проводок. Движения в регистре формируются при проведении документов *Операция* и *Приходная накладная*, которые являются регистраторами регистра бухгалтерии.

На примере этого регистра рассмотрим типичные задачи по получению данных из регистра бухгалтерии.

Получение движений регистра бухгалтерии

Предположим, необходимо выбрать движения регистра бухгалтерии с отбором по описанию проводок, которое хранится в реквизите регистра *Содержание*. Для этого можно выполнить следующий запрос (листинг 3.47).

Листинг 3.47. Отбор движений регистра бухгалтерии по описанию проводок

```
ВЫБРАТЬ
  ОсновнойРегистрБухгалтерии.Период,
  ОсновнойРегистрБухгалтерии.Регистратор,
  ОсновнойРегистрБухгалтерии.НомерСтроки КАК НомерСтроки,
  ОсновнойРегистрБухгалтерии.СчетДт,
  ОсновнойРегистрБухгалтерии.КоличествоДт,
  ОсновнойРегистрБухгалтерии.СчетКт,
  ОсновнойРегистрБухгалтерии.КоличествоКт,
  ОсновнойРегистрБухгалтерии.Сумма
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии КАК ОсновнойРегистрБухгалтерии
ГДЕ
  ОсновнойРегистрБухгалтерии.Содержание ПОДОБНО "%Покупка%"

УПОРЯДОЧИТЬ ПО
  ОсновнойРегистрБухгалтерии.Период
```

В данном запросе отбираются все движения регистра бухгалтерии, в описании которых содержится слово «Покупка». Результат выполнения этого запроса представлен на рис. 3.57.

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии (Записей в результате: 3)

Период	Регистратор	НомерСтроки	СчетДт	КоличествоДт	СчетКт	КоличествоКт	Сумма
23.06.2013	Приходная накладная 000000001 от 23.06.2013	1	Товары	5	Поставщики		50
23.06.2013	Приходная накладная 000000001 от 23.06.2013	2	Товары	5	Поставщики		25
25.06.2013	Приходная накладная 000000002 от 25.06.2013	1	Товары	3	Поставщики		15

Рис. 3.57. Отбор движений регистра бухгалтерии по описанию проводок

Рассмотрим также задачу отбора движений регистра бухгалтерии по заданному регистратору, т. е. конкретному документу, который произвел движения в регистре. Для этого можно выполнить следующий запрос (листинг 3.48).

Листинг 3.48. Отбор движений регистра бухгалтерии по регистратору

```
ВЫБРАТЬ
  ОсновнойРегистрБухгалтерии.Период,
  ОсновнойРегистрБухгалтерии.НомерСтроки КАК НомерСтроки,
  ОсновнойРегистрБухгалтерии.СчетДт,
  ОсновнойРегистрБухгалтерии.КоличествоДт,
  ОсновнойРегистрБухгалтерии.СчетКт,
  ОсновнойРегистрБухгалтерии.КоличествоКт,
  ОсновнойРегистрБухгалтерии.Сумма
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии КАК ОсновнойРегистрБухгалтерии
ГДЕ
  ОсновнойРегистрБухгалтерии.Регистратор = &Регистратор
```

В данном запросе на список движений регистра бухгалтерии накладывается отбор по полю *Регистратор*. Выбранный документ-регистратор *Операция* или *Приходная накладная* передается как значение параметра *&Регистратор*.

Для документа *Операция №3* мы получим следующий результат (рис. 3.58).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии (Записей в результате: 2)

Период	НомерСтроки	СчетДт	КоличествоДт	СчетКт	КоличествоКт	Сумма
01.03.2013 12:00:00	1	Товары	10	Поставщики		100
01.03.2013 12:00:00	2	Товары	5	Сотрудники		50

Рис. 3.58. Отбор движений регистра бухгалтерии по регистратору

Получение остатков

Для построения бухгалтерских отчетов и анализа данных в большинстве случаев используются виртуальные таблицы регистров бухгалтерии. Эти таблицы предназначены для решения конкретных задач.

Для получения остатков по счетам в разрезе субконто и измерений используется таблица остатков *Остатки()*. Агрегация данных производится по счетам, а также измерениям и субконто, указанным в запросе. Итоги ресурсов получаются только по активным записям.

Виртуальная таблица *Остатки* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*) с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя ресурса>Остаток* – поле, содержащее остаток ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Остаток*. Содержит абсолютный остаток без учета вида счета – дебетовый остаток показывается положительным числом, кредитовый – отрицательным числом;
- *<Имя ресурса>ОстатокДт* – поле, содержащее дебетовый остаток ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *ОстатокДт*. Содержит дебетовый остаток с учетом вида счета. Если счет пассивный, значение этого поля всегда равно нулю. Если счет активный, значение поля равно значению поля *Остаток*. Если счет активно-пассивный, значение поля равно значению поля *Остаток*, если *Остаток* больше или равен нулю. Если *Остаток* меньше нуля, значит – ноль;
- *<Имя ресурса>ОстатокКт* – поле, содержащее кредитовый остаток ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *ОстатокКт*. Содержит кредитовый остаток с учетом вида счета. Если счет активный, значение этого поля всегда равно нулю. Если счет пассивный, равно -

Остаток. Если счет активно-пассивный, значение поля равно нулю, если значение поля *Остаток* больше или равно нулю. Если значение поля *Остаток* меньше нуля, значение этого поля равно *-Остаток*;

- *<Имя ресурса>РазвернутыйОстатокДт* – содержит развернутый дебетовый остаток, который считается развернуто по всем измерениям, указанным в запросе. Имеет смысл только при использовании в запросе итогов по измерениям. Для детальной записи значение этого поля равно значению поля *<Имя ресурса>ОстатокДт*. Для итоговой записи равно сумме дебетовых остатков всех детальных записей;
- *<Имя ресурса>РазвернутыйОстатокКт* – содержит развернутый кредитовый остаток, который считается развернуто по всем измерениям, указанным в запросе. Имеет смысл только при использовании в запросе итогов по измерениям. Для детальной записи значение этого поля равно значению поля *<Имя ресурса>ОстатокКт*. Для итоговой записи равно сумме кредитовых остатков всех детальных записей;
- *Субконто<Номер субконто>* – имеет тип *Характеристика.<имя>*. Содержит значение субконто. Количество полей *Субконто* зависит от максимального количества субконто на счете плана счетов. Номер субконто начинается с *1*. Набор и порядок этих полей определяются параметром *Субконто*, переданным в виртуальную таблицу;
- *Счет* – имеет тип *ПланСчетовСсылка.<имя>*. Позволяет получить остатки, сгруппированные по счетам.

При построении этой виртуальной таблицы могут использоваться параметры, с помощью которых настраивается или уточняется состав получаемых данных. Параметры следует задавать строго в порядке их описания:

- *Период* – имеет тип *Дата*, *МоментВремени* или *Граница*. Момент времени, на который нужно посчитать остатки. Если параметр не задан, то будут получены актуальные остатки, включающие движения последнего проведенного документа;
- *УсловиеСчета* – содержит конструкцию языка запросов. Позволяет установить фильтр по счету или счетам. Как правило, содержит следующие условия: *Счет = (В ИЕРАРХИИ, В) &Счет*;
- *Субконто* – в этот параметр передается ссылка, или массив ссылок, или фиксированный массив ссылок, или список значений, содержащий ссылки, на виды субконто. Задает набор и порядок субконто, которыми можно оперировать в запросе. А также служит для отбора записей регистра по видам субконто. Если параметр задан, то будут выбираться данные только по тем счетам, у которых определены все указанные виды субконто. Если параметр не задан, то ограничений по видам субконто нет. Субконто определяются позиционно по соответствующему счету;
- *Условие* – содержит конструкцию языка запросов. Позволяет устанавливать отбор данных виртуальной таблицей по значениям субконто и измерений регистра бухгалтерии.

Рассмотрим примеры построения запросов к таблице остатков регистра бухгалтерии.

Предположим, требуется получить абсолютный количественный остаток определенного товара со счета товаров. Для этого нужно выполнить следующий запрос (листинг 3.49).

Листинг 3.49. Вывод остатка заданного товара со счета товаров

```
ВЫБРАТЬ
  Остатки.КоличествоОстаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет = &СчетТоваров, ,
  Субконто1 = &Номенклатура) КАК Остатки
```

В данном запросе в параметр *УсловиеСчета*, используемый при построении виртуальной таблицы остатков, мы передаем ссылку на счет товаров (ссылку для предопределенного счета можно получить по имени, например: *СчетТоваров* = *ПланыСчетов.ОсновнойПланСчетов.Товары*). В параметр *Условие* передаем условие отбора конкретного товара по значению субконто, т. е. ссылку на элемент справочника *Номенклатура*, который используется на счете товаров как первый вид субконто. Результат запроса будет содержать одну строку в одном поле: абсолютный остаток по ресурсу регистра *Количество*.

По аналогии с помощью следующего запроса можно получить стоимостной остаток материалов на определенном складе на счете материалов. Для этого нужно выполнить следующий запрос (листинг 3.50).

Листинг 3.50. Вывод остатка материалов на заданном складе на счете материалов

```
ВЫБРАТЬ
  Остатки.СуммаОстаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет = &СчетМатериалов, ,
  Субконто1 = &Склад) КАК Остатки
```

В данном запросе в виртуальную таблицу остатков мы передаем ссылку на счет материалов и условие отбора конкретного склада по значению субконто, т. е. ссылку на элемент справочника *Склады*, который используется на счете материалов как первый вид субконто.

Результат запроса будет содержать одну строку в одном поле: абсолютный остаток по ресурсу регистра *Сумма* (рис. 3.59).

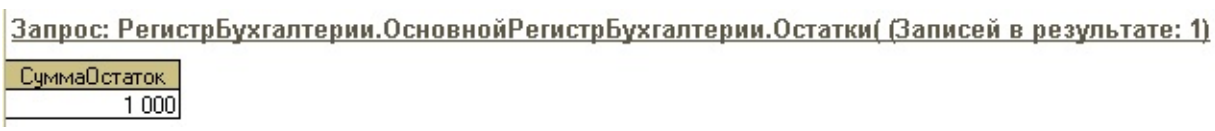


Рис. 3.59. Вывод остатка материалов на заданном складе на счете материалов

Следующий запрос позволяет отобрать остатки в разрезе валют по всем счетам, на

которых ведется валютный учет (листинг 3.51).

Листинг 3.51. Вывод остатков на валютных счетах

```
ВЫБРАТЬ
  Остатки.Счет,
  Остатки.Валюта,
  Остатки.ВалютнаяСуммаОстаток,
  Остатки.СуммаОстаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет.Валютный = ИСТИНА, , ) КАК
  Остатки
```

Результат выполнения данного запроса представлен на рис. 3.60.

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки((Записей в результате: 5)

Счет	Валюта	ВалютнаяСуммаОстаток	СуммаОстаток
Касса			-6 762
Новый счет	Доллар	10	285
Касса	Рубль	470	470
Касса	Доллар	100	2 850
Касса	Евро	100	3 500

Рис. 3.60. Вывод остатков на валютных счетах

В данном запросе в параметр *УсловиеСчета* передается условие отбора по счетам с признаком учета *Валютный*. Остатки подсчитываются в разрезе счетов и валют (значения измерения *Валюта*).

Параметр «Субконто»

На примере таблицы остатков рассмотрим особенности третьего параметра *Субконто*, используемого при построении виртуальной таблицы остатков. Все нижесказанное актуально также для виртуальных таблиц оборотов (имена параметров *Субконто*, *КорСубконто*), остатков и оборотов, оборотов ДтКт (имена параметров *СубконтоДт*, *СубконтоКт*).

Все перечисленные параметры могут принимать значения *ПланыВидовХарактеристикСсылка.<имя>* или содержать массив, состоящий из значений указанного типа данных. То есть, точнее говоря, в параметре *Субконто* можно передать не значение субконто, а значение вида субконто или массив видов субконто. А для отбора по значению субконто используется параметр *Условие* (см. листинги 3.49, 3.50).

Рассмотрим два аспекта использования параметра *Субконто*. Прежде всего, это отбор итогов по виду субконто.

Продемонстрируем это на примере нашей демонстрационной конфигурации. Предположим, нам необходимо получить отчет по остаткам товарно-материальных запасов. Для этого мы должны получить остатки со счетов учета материальных ценностей. Таких счетов два: предопределенный счет *Товары*, на котором первым

субконто учитывается *Номенклатура*, а вторым – *Склады*. И предопределенный счет *Материалы*, на котором первым субконто учитываются *Склады*, а вторым – *Номенклатура*.

Выведем стоимостные остатки со счетов учета материальных ценностей в разрезе субконто *Номенклатура* с помощью следующего запроса (листинг 3.52).

Листинг 3.52. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура»

```
ВЫБРАТЬ
  Остатки.Субконто1 КАК Товар,
  Остатки.СуммаОстаток КАК Остаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( ,Счет В (&СчетТоваров,
  &СчетМатериалов), &ВидСубконто, ) КАК Остатки
ИТОГИ
  СУММА(Остаток)
ПО
  ОБЩИЕ
```

В консоли запросов установим значение параметра *ВидСубконто* как субконто *Номенклатура* типа *СправочникСсылка.Номенклатура*, значения параметров *СчетМатериалов* и *СчетТоваров* как ссылки на предопределенные счета *Материалы* и *Товары* соответственно. Выполним запрос (рис. 3.61).

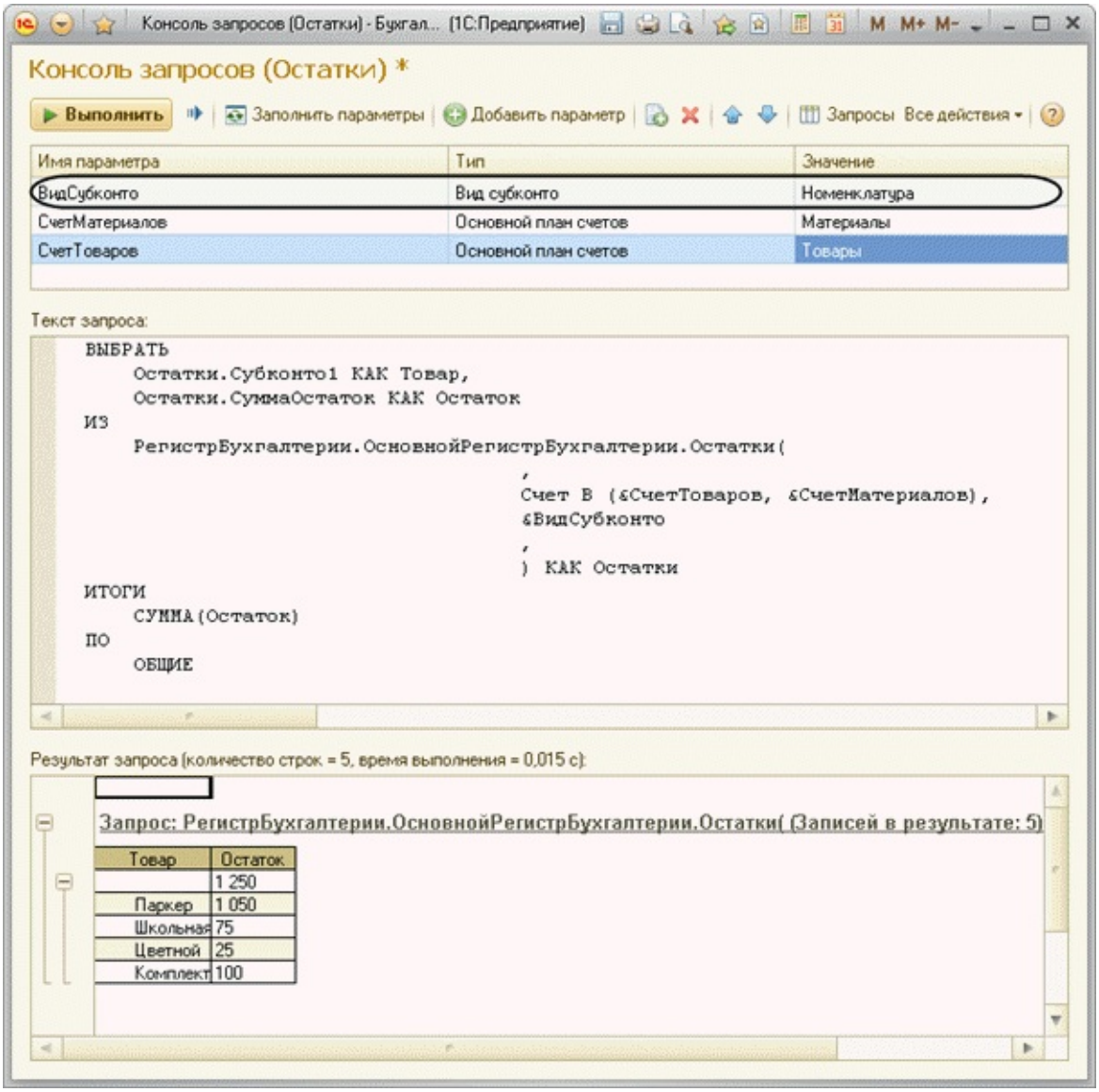


Рис. 3.61. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура»

В результате итоговые остатки товарно-материальных ценностей на обоих счетах (*Товары* и *Материалы*) подсчитаны в разрезе номенклатуры, которая учитывается как аналитика на этих счетах.

Теперь рассмотрим второй аспект использования параметра *Субконто* – для определения набора и порядка следования субконто в результате запроса.

В нашем примере проблема в том, что субконто *Номенклатура* прикреплено первым субконто на счете *Товары* и вторым – на счете *Материалы*. А обращаемся мы к ним в запросе именно по номеру (*Субконто1*, *Субконто2*).

Предположим, требуется включить в отчет две группировки по субконто: сначала пользователь хочет увидеть итоги по номенклатуре, а потом детализацию по складам, на каком сколько числится.

Если при этом не указать параметр *Субконто* (листинг 3.53), то поле *Субконто1* будет содержать и товары для счета товаров, и склады для счета материалов. Поле *Субконто2* будет содержать склады для счета товаров и товары для счета материалов

(рис. 3.62).

Листинг 3.53. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

```
ВЫБРАТЬ
  Остатки.Субконто1 КАК Товар,
  Остатки.Субконто2 КАК Склад,
  Остатки.СуммаОстаток КАК Остаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет В (&СчетТоваров,
  &СчетМатериалов), , ) КАК Остатки
ИТОГИ
  СУММА(Остаток)
ПО
  ОБЩИЕ
```

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки((Записей в результате: 8)

Товар	Склад	Остаток
		1 250
Цветной	Офис	10
Цветной	Филиал	15
Комплект	Офис	100
Филиал	Паркер	1 000
Паркер	Офис	-50
Паркер	Филиал	100
Школьная	Офис	75

Рис. 3.62. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

В итоге мы видим строку результата запроса – *Филиал Паркер 1000*, где *Филиал* – это склад (но находится в колонке *Товар*), а *Паркер* – товар (но находится в колонке *Склад*).

Чтобы избежать подобной путаницы и добиться в результате запроса нужного порядка следования субконто на счетах (без изменения настройки плана счетов), передадим в параметр *ВидыСубконто* массив видов субконто, следующих в нужной последовательности (листинг 3.54).

Листинг 3.54. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто1 КАК Товар,
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто2 КАК Склад,
  |        ОсновнойРегистрБухгалтерииОстатки.СуммаОстаток КАК Остаток
  |ИЗ
  |        РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет В (&СчетТоваров,
  &СчетМатериалов), &ВидыСубконто, ) КАК ОсновнойРегистрБухгалтерииОстатки
  |
  |ИТОГИ
  |        СУММА(Остаток)
  |ПО
  |        ОБЩИЕ,
  |        Товар";

Запрос.УстановитьПараметр("СчетТоваров", ПланыСчетов.ОсновнойПланСчетов.Товары);
Запрос.УстановитьПараметр("СчетМатериалов", ПланыСчетов.ОсновнойПланСчетов.Материалы);
```

```
мВидыСубконто = Новый Массив;
мВидыСубконто.Добавить(ПланыВидовХарактеристик.ВидыСубконто.Номенклатура);
мВидыСубконто.Добавить(ПланыВидовХарактеристик.ВидыСубконто.Склады);
Запрос.УстановитьПараметр("ВидыСубконто", мВидыСубконто);

Результат = Запрос.Выполнить();
Остатки = Результат.Выбрать();

// Вывод таблицы остатков.
…
```

В результате мы получим нужные итоги (рис. 3.63).

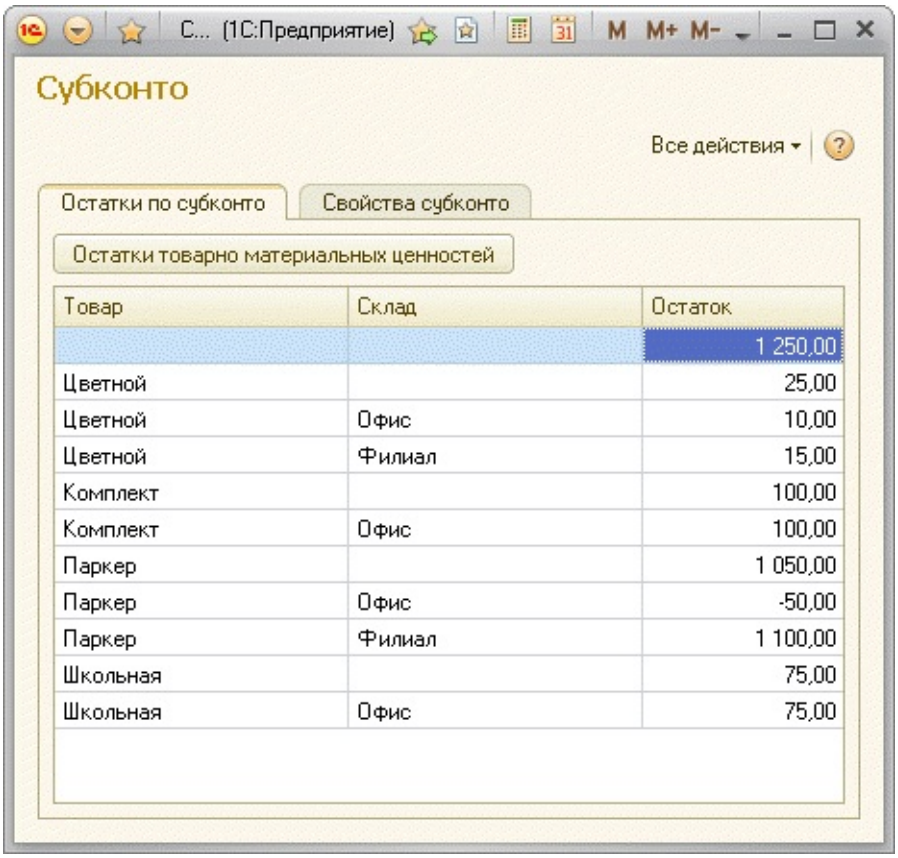


Рис. 3.63. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

Таким образом, теперь независимо от счета первым субконто в нашем запросе является *Номенклатура*, вторым – *Склады*. Причем если будет нужно изменить последовательность группировок в запросе (сначала группировать по складам, потом внутри склада – по номенклатурным позициям), нам достаточно изменить порядок следования элементов массива, который передается в параметр виртуальной таблицы *Субконто*.

Этот пример можно посмотреть в демонстрационной конфигурации «Бухгалтерский учет», прилагающейся к книге, в обработке *Субконто*.

Теперь рассмотрим еще одну тонкость, касающуюся отбора по виду субконто. Дело в том, что если параметр *Субконто* задан, то данные выбираются только по тем счетам, у которых определены все указанные виды субконто.

Продемонстрируем эту особенность на следующем примере. В нашей демонстрационной конфигурации по счету *Материалы* существует всего одна проводка. В дебет счета начисляется *1000*, значение первого субконто – *Паркер* (товар), второго – *Филиал* (склад).

Изменим аналитику на счете материалов. В конфигураторе удалим для предопределенного счета *Материалы* второй вид субконто, а первый вид субконто изменим на субконто *Номенклатура*. Для счета *Товары* аналитику оставим без изменений. Первым субконто на нем учитывается *Номенклатура*, вторым – *Склады*. В документе *Операция №8* изменим значение первого субконто на *Паркер* (товар), а второго субконто на счете материалов теперь нет.

После этого выполним команду *Остатки товарно-материальных ценностей* в обработке *Субконто* (рис. 3.64).

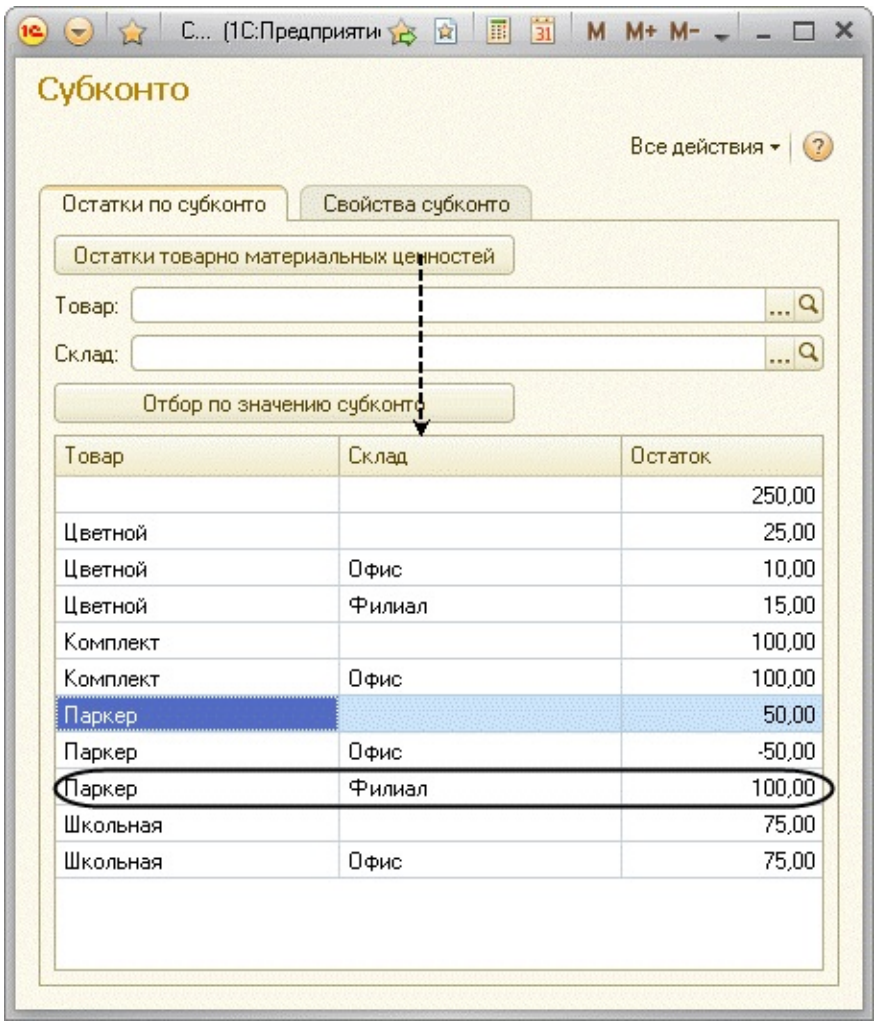


Рис. 3.64. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

Если сравнить результат с правильным вариантом (см. рис. 3.63), то мы видим, что пропала как раз строка результата запроса – *Паркер Филиал 1000*, т. к. в параметре *Субконо* передается массив из двух видов субконто (*Номенклатура*, *Склады*), а у счета материалов теперь один вид субконто – *Номенклатура*.

Ситуацию можно исправить путем объединения запросов к каждому счету (*Товары* и

Материалы) отдельно (листинг 3.55).

Листинг 3.55. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто1 КАК Товар,
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто2 КАК Склад,
  |        ОсновнойРегистрБухгалтерииОстатки.СуммаОстаток КАК Остаток
  |ИЗ
  |        РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет = &СчетТоваров, ,
  ) КАК ОсновнойРегистрБухгалтерииОстатки
  |
  |ОБЪЕДИНИТЬ ВСЕ
  |
  |ВЫБРАТЬ
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто1,
  |        NULL,
  |        ОсновнойРегистрБухгалтерииОстатки.СуммаОстаток
  |ИЗ
  |        РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( , Счет =
  &СчетМатериалов, , ) КАК ОсновнойРегистрБухгалтерииОстатки
  |
  |ИТОГИ
  |        СУММА(Остаток)
  |ПО
  |        ОБЩИЕ,
  |        Товар";

Запрос.УстановитьПараметр("СчетТоваров",    ПланыСчетов.ОсновнойПланСчетов.Товары);
Запрос.УстановитьПараметр("СчетМатериалов", ПланыСчетов.ОсновнойПланСчетов.Материалы);

Результат = Запрос.Выполнить();
Остатки = Результат.Выбрать();

// Вывод таблицы остатков.
…
```

В запросе используется объединение данных запроса к счету товаров и запроса к счету материалов. Параметр *Субконто* в этих запросах не задан, т. к. в этом случае виды субконто и их последовательность определяются позиционно по соответствующему счету. Для счета товаров – это *Номенклатура* и *Склады*, для счета материалов – *Номенклатура*. Затем для результата объединения этих запросов рассчитываются общие итоги и итоги по полю *Товар*.

В результате мы получим нужные итоги (рис. 3.65).

Рис. 3.65. Вывод остатков товарно-материальных ценностей в разрезе субконто «Номенклатура» и «Склады»

Если же требуется отобрать остатки по конкретным значениям субконто, то в описанной выше ситуации (когда на одном счете два субконто, а на втором – только одно из них) можно использовать следующий запрос (листинг 3.56).

Листинг 3.56. Вывод остатков товарно-материальных ценностей с отбором по значению субконто

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто1 КАК Товар,
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто2 КАК Склад,
  |        ОсновнойРегистрБухгалтерииОстатки.СуммаОстаток КАК Остаток
  |ИЗ
  |        РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки( ,Счет В (&СчетТоваров,
  &СчетМатериалов), , ) КАК ОсновнойРегистрБухгалтерииОстатки
  |ГДЕ
  |        ОсновнойРегистрБухгалтерииОстатки.Субконто1 = &Номенклатура И
  |        ВЫБОР КОГДА ОсновнойРегистрБухгалтерииОстатки.Счет = &СчетТоваров
  |        ТОГДА ОсновнойРегистрБухгалтерииОстатки.Субконто2 = &Склад ИНАЧЕ ИСТИНА КОНЕЦ
  |
  |ИТОГИ
  |        СУММА(Остаток)
  |ПО
  |        Товар";

Запрос.УстановитьПараметр("СчетТоваров", ПланыСчетов.ОсновнойПланСчетов.Товары);
Запрос.УстановитьПараметр("СчетМатериалов", ПланыСчетов.ОсновнойПланСчетов.Материалы);
Запрос.УстановитьПараметр("Номенклатура", Товар);
```

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

Результат = Запрос.Выполнить();
Остатки = Результат.Выбрать();

// Вывод таблицы остатков.
…
```

В данном запросе в условии отбора в предложении *ГДЕ* применяется конструкция *ВЫБОР КОГДА … ИНАЧЕ … КОНЕЦ*. В результате условие отбора по значению второго субконто применяется только для счета товаров.

Выберем в обработке *Субконто* в качестве товара *Паркер*, а в качестве склада *Филиал* и получим следующий результат (рис. 3.66).

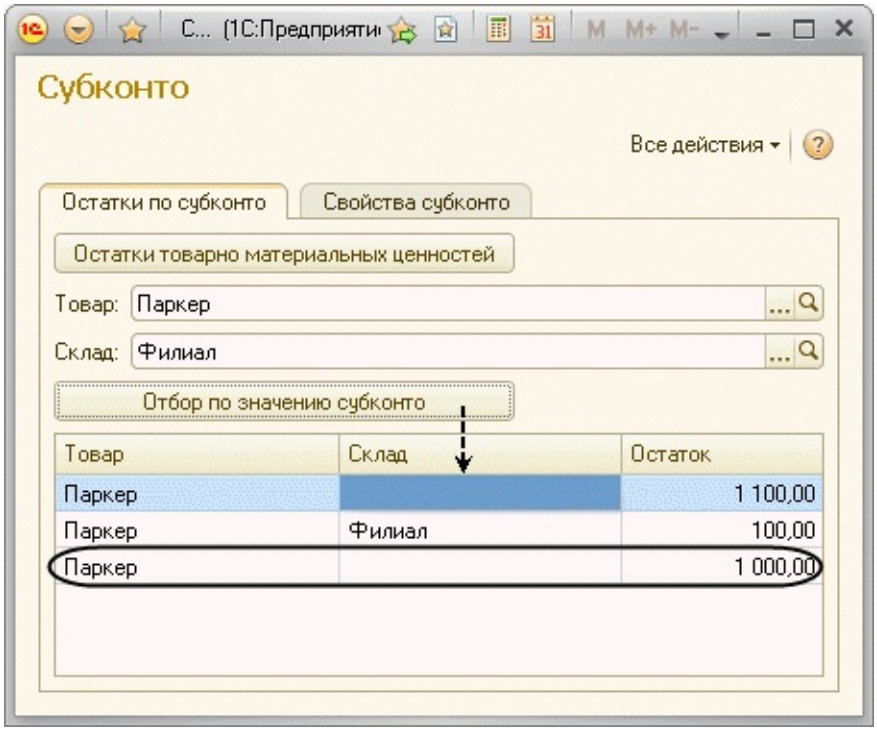


Рис. 3.66. Вывод остатков товарно-материальных ценностей с отбором по значению субконто

Особенности использования периодов и моментов времени при получении остатков

Все виртуальные таблицы регистра бухгалтерии позволяют устанавливать отбор итогов или на момент времени, или за период. Для установки такого отбора итогов можно использовать значения типа *Дата*, *МоментВремени* или *Граница*. Рассмотрим использование этих объектов в параметре *Период* виртуальной таблицы остатков, которая получает остатки на момент времени, указанный в этом параметре.

В нашей демонстрационной конфигурации журнал проводок по счету *Касса* за июнь 2013 года содержит четыре проводки за первое и второе июня по 10 рублей каждая. Входящий остаток составил 15 рублей, исходящий остаток – 55 рублей (табл. 3.3).

Таблица 3.3. Проводки по счету «Касса» за июнь 2013

Период	Счет дебета	Счет кредита	Сумма	Текущий остаток

Остаток 31.05.2013				15,00
01.06.2013 12:00:00	Касса	Покупатели	10.00	25,00
01.06.2013 23:59:59	Касса	Покупатели	10.00	35,00
Остаток 01.06.2013				35,00
02.06.2013 0:00:00	Касса	Покупатели	10.00	45,00
02.06.2013 12:00:00	Касса	Покупатели	10.00	55,00
Остаток 02.06.2013				55,00

Для анализа остатка мы воспользуемся функцией, которая получает остаток на счете *Касса* при помощи следующего запроса (листинг 3.57).

Листинг 3.57. Пример получения остатков регистра бухгалтерии

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        Остатки.СуммаОстаток
  |ИЗ
  |        РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Остатки(&Момент, Счет = &Счет, ,
  ) КАК Остатки";

Запрос.УстановитьПараметр("Момент", Момент);
Запрос.УстановитьПараметр("Счет", ПланыСчетов.ОсновнойПланСчетов.Касса);

Если Результат.Пустой() Тогда
  Возврат 0;

КонецЕсли;

Остатки = Результат.Выбрать();
Остатки.Следующий();

Возврат Остатки.СуммаОстаток;
```

В запросе получается остаток по счету *Касса* на тот момент времени, который мы будем передавать ему в качестве параметра. Наша задача – получить остаток на счете *Касса*, который там сформировался к концу 1 июня (или, другими словами, к началу дня 2 июня).

Для этого установим значение поля ввода *Дата* обработки *Работа с виртуальными таблицами* как *01.06.2013* и рассмотрим следующие варианты:

1. Сначала установим значение параметра запроса *Момент* равным дате (листинг 3.58).

Листинг 3.58. Первый вариант установки параметра запроса

```
Момент = Дата
```

Поскольку дата в системе «1С:Предприятие» включает в свой состав время, то значение параметра *Момент* станет равным *01.06.2013 00:00:00*.

1. Установим значение параметра запроса *Момент* равным концу дня указанной даты (листинг 3.59).

Листинг 3.59. Второй вариант установки параметра запроса

```
Момент = КонецДня(Дата);
```

Значение параметра *Момент* станет равным *01.06.2013 23:59:59*.

1. Установим значение параметра запроса *Момент* равным началу следующего дня от указанной даты (листинг 3.60).

Листинг 3.60. Третий вариант установки параметра запроса

```
ОдинДень = 60 * 60 * 24;
Момент = НачалоДня(Дата + ОдинДень);
```

Значение параметра *Момент* станет равным *02.06.2013 00:00:00*.

1. Установим значение параметра запроса *Момент* равным концу дня указанной даты, используя объект *Граница* с параметром *ВидГраницы.Включая* (листинг 3.61).

Листинг 3.61. Четвертый вариант установки параметра запроса

```
КонДня = КонецДня(Дата);
Граница = Новый Граница(КонДня, ВидГраницы.Включая);
Момент = Граница.Значение;
```

Значение параметра *Момент* станет равным *01.06.2013 23:59:59*, включая движения документа, проведенного в последнюю секунду этого дня.

Результаты различных вариантов установки момента времени при получении остатков приведены в следующей таблице (табл. 3.4).

Таблица 3.4. Сводная таблица полученных результатов

№	Метод	Момент	Остаток	Комментарий
1		01.06.10 0:00:00	15,00	Не было учтено время, получили остаток на начало дня
2	Конец текущего дня	01.06.10 23:59:59	25,00	Получили остаток на конец дня, но в остатки не попала последняя проводка, сделанная в 23:59:59
	Начало	02.06.10		

3	следующего дня	0:00:00	35,00	Правильно!
4	Граница (включая)	Граница	35,00	Правильно!

Эти результаты демонстрируют тот факт, что таблица остатков регистра бухгалтерии (также как и таблица остатков регистра накопления) строится на начало секунды, то есть *не включая* границы заданного периода.

Поэтому результат, полученный в первом случае, не включает движений по регистру, сформированных документами за *01.06.2013*. Во втором случае результат не включает движения документа, проведенного в последнюю секунду этого дня. В третьем случае результат правильный, т. к. получает данные на начало следующего дня, но при этом не включает движений по регистру, сформированных документами за *02.06.2013*. И в четвертом случае результат получается с включением всех движений за дату *01.06.2013*, что нам и требовалось.

Итак, для получения правильного остатка на конец периода, включающего все операции, необходимо или получать остатки на начало следующего периода, или использовать объект *Граница* с параметром *ВидГраницы.Включая*.

В случаях, когда нужно получить итоги с точностью «до ссылки», требуется использовать параметр *Период* типа *МоментВремени*, полученный из даты документа и ссылки на документ. При этом необходимо иметь в виду, что, также как и для таблицы остатков регистра накопления, данные получаются, *исключая* записи движений самого документа.

Получение оборотов

Для получения оборотов по счету в разрезе субконто и измерений и оборотов счета с корреспондирующими счетами (как дебетуемыми, так и кредитуемыми) используется виртуальная таблица *Обороты()*. Таблица оборотов может быть использована при разработке таких отчетов, как «Анализ счета/субконто/по датам», «Журнальный ордер», «Ведомость по счету» и др.

Виртуальная таблица *Обороты* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра, в разрезе которого посчитан оборот. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя измерения>Кор* – поле, содержащее значения корреспондирующего небалансового измерения регистра, в разрезе которого посчитан оборот;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*) с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;

- *<Имя ресурса>Оборот* – поле, содержащее оборот ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Оборот*. Содержит разницу оборотов (оборот дебета минус оборот кредита);
- *<Имя ресурса>ОборотДт* – поле, содержащее дебетовый оборот ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *ОборотДт*;
- *<Имя ресурса>ОборотКт* – поле, содержащее кредитовый оборот ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *ОборотКт*;
- *<Имя ресурса>КорОборот* – поле, содержащее оборот небалансового ресурса регистра корреспондирующих счетов, по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *КорОборот*. Содержит разницу оборотов (кор оборот дебета минус кор оборот кредита);
- *<Имя ресурса>КорОборотДт* – поле, содержащее дебетовый оборот небалансового ресурса регистра корреспондирующих счетов, по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *КорОборотДт*;
- *<Имя ресурса>КорОборотКт* – поле, содержащее кредитовый оборот небалансового ресурса регистра корреспондирующих счетов, по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *КорОборотКт*;
- *Счет* – имеет тип *ПланСчетовСсылка.<имя>*. Содержит счет, в разрезе которого посчитан оборот;
- *КорСчет* – имеет тип *ПланСчетовСсылка.<имя>*. Содержит корреспондирующий счет, в разрезе которого посчитан оборот;
- *Субконто<Номер субконто>* – имеет тип *Характеристика.<имя>*. Содержит значение субконто, в разрезе которого посчитан оборот. Количество полей *Субконто* зависит от максимального количества субконто на счете плана счетов. Номер субконто начинается с *1*. Набор и порядок субконто определяются параметром *Субконто*;
- *КорСубконто<Номер субконто>* – имеет тип *Характеристика.<имя>*. Содержит значение корреспондирующего субконто, в разрезе которого посчитан оборот. Количество полей *Субконто* зависит от максимального количества субконто на счете плана счетов. Номер субконто начинается с *1*. Набор и порядок субконто определяются параметром *КорСубконто*;
- *НомерСтроки* – имеет тип *Число*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Запись*. Содержит значение поля *НомерСтроки* записи движения регистра;
- *Период* – имеет тип *Дата*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Секунда*, *Минута*, *Час*, *Регистратор* или *Запись*. Данное поле содержит начальную дату и время периода, к которому относится оборот регистра;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*:

Регистратор или *Запись*. Данное поле содержит ссылку на документ-регистратор, к которому относится оборот регистра.

Параметры виртуальной таблицы оборотов позволяют задать условие отбора данных из информационной базы. Параметры следует задавать строго в порядке их описания:

- *НачалоПериода*, *КонецПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Период времени, за который будут получены обороты. Если параметры не заданы, то будут получены все обороты регистра;
- *Периодичность* – содержит конструкцию языка запросов. Позволяет задать дополнительную группировку данных по стандартным периодам. Возможные значения: *Период*, *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Час*, *Минута*, *Секунда*, *Регистратор*, *Запись*. Если периодичность не задана или задана как *Период,* дополнительной группировки не выполняется;
- *УсловиеСчета* – содержит конструкцию языка запросов. Позволяет установить фильтр по счету или счетам. Как правило, содержит следующие условия: *Счет = (В ИЕРАРХИИ, В) &Счет*;
- *Субконто* – имеет тип *ПланВидовХарактеристикСсылка.<имя>* или содержит массив значений этого типа. Задает набор и порядок субконто, которыми можно оперировать в запросе. А также служит для отбора оборотов регистра по видам субконто. Если параметр задан, то будут выбираться данные только по тем счетам, у которых определены все указанные виды субконто. Если параметр не задан, то ограничений по видам субконто нет. Субконто определяются позиционно по соответствующему счету;
- *Условие* – содержит конструкцию языка запросов. Позволяет устанавливать отбор данных виртуальной таблицей по значениям субконто и измерений регистра бухгалтерии;
- *УсловиеКорСчета* – содержит конструкцию языка запросов. Позволяет установить фильтр по корреспондирующему счету или счетам. Как правило, содержит следующие условия: *КорСчет = (В ИЕРАРХИИ, В) &КорСчет*;
- *КорСубконто* – имеет тип *ПланВидовХарактеристикСсылка.<имя>* или содержит массив значений этого типа. В этот параметр таблицы можно передать ссылку на вид субконто для получения отбора по этому виду или массив видов для установки отбора и упорядочивания видов субконто в результате запроса.

Рассмотрим примеры построения запросов к таблице оборотов регистра бухгалтерии. Все запросы обращаются к итогам за весь период, который есть в регистре (не указываются параметры *НачалоПериода* и *КонецПериода*).

Пример 1

Предположим, требуется узнать, сколько было куплено и списано различных товаров. Для этого нужно получить дебетовые и кредитовые обороты (*КоличествоОборотДт*, *КоличествоОборотКт*) со счета товаров в разрезе первого субконто (*Номенклатура*), прикрепленного к счету товаров.

Также нужно узнать с какого корреспондирующего счета (субконто корреспондирующего счета) были получены те или иные товары. Для этого нужно получить обороты в разрезе поля *КорСчет* и *КорСубконто* (листинг 3.62).

Листинг 3.62. Поступление и списание товаров со счета «Товары»

```
ВЫБРАТЬ
  Обороты.Счет,
  Обороты.Субконто1,
  Обороты.КорСчет,
  Обороты.КорСубконто1,
  Обороты.КоличествоОборот,
  Обороты.КоличествоОборотДт,
  Обороты.КоличествоОборотКт
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты( , , , Счет = &СчетТоваров, , , ,
  ) КАК Обороты
```

В запрос в качестве параметра передается условие по счету, где *СчетТоваров* – ссылка на предопределенный счет *Товары* плана счетов.

В результате мы видим, какой именно товар (*Субконто1*) с какого корреспондирующего счета был получен (*КорСчет*), от какого контрагента (*КорСубконто1*) и в каком количестве был получен (рис. 3.67).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты((Записей в результате: 7)

Счет	Субконто1	КорСчет	КорСубконто1	КоличествоОборот	КоличествоОборотДт	КоличествоОборотКт
Товары	Паркер	Товары	Комплект	-10		10
Товары	Школьная	Сотрудники	Иванов	5	5	
Товары	Комплект	Товары	Паркер	1	1	
Товары	Паркер	Поставщики	СтройТоргВсе	10	10	
Товары	Паркер	Поставщики	Дисконт центр	5	5	
Товары	Цветной	Поставщики	МонтажДоставка	3	3	
Товары	Школьная	Поставщики	Дисконт центр	5	5	

Рис. 3.67. Поступление и списание товаров со счета «Товары»

Заметьте, что в результате запроса присутствуют две строки, где и основным, и корреспондирующим счетом является счет *Товары*. При этом в дебет счета поступила одна единица номенклатуры *Комплект*, а с кредита счета списан товар *Паркер* в количестве *10* штук.

Чтобы проанализировать, из какого количества авторучек *Паркер* какое количество *Комплектов* было собрано, нам потребуется использовать в запросе поле с постфиксом *КорОборот*. Наглядно представить эту информацию можно с помощью следующего запроса (листинг 3.63).

Листинг 3.63. Корреспонденция счета «Товары» с самим собой. Комплектация товаров

```
ВЫБРАТЬ
  Обороты.Субконто1,
  Обороты.КоличествоОборотДт,
  Обороты.КорСубконто1,
```

```
  Обороты.КоличествоКорОборотДт
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты( , , , Счет = &СчетТоваров, , ,
  КорСчет = &СчетТоваров, ) КАК Обороты
```

В запросе мы устанавливаем фильтр по полям *Счет* и *КорСчет*, и получим таким образом в качестве итогов обороты по всем проводкам, где корреспондировали между собой счет *Товары* и счет *Товары* (рис. 3.68).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты((Записей в результате: 1)

Субконто1	КоличествоОборотДт	КорСубконто1	КоличествоКорОборотДт
Комплект	1	Паркер	10

Рис. 3.68. Корреспонденция счета «Товары» с самим собой. Комплектация товаров

Пример 2

Рассмотрим следующую задачу. Предположим, необходимо получить оборот между счетами, отражающий объем выручки, полученной в кассу от покупателей. Для этого нужно получить обороты за период в дебет счета *Касса* с кредита счета *Покупатели* в разрезе контрагентов (субконто *Контрагенты* прикреплено к счету *Покупатели* плана счетов). Также нужно получить сумму возвращенных покупателям средств из кассы и разницу между выручкой и возвратами. Это можно сделать с помощью следующего запроса (листинг 3.64).

Листинг 3.64. Обороты между счетами «Касса» и «Покупатели»

```
// Первый вариант.
ВЫБРАТЬ
  Обороты.КорСубконто1 КАК Покупатель,
  Обороты.СуммаОборотДт КАК Выручка,
  Обороты.СуммаОборотКт КАК Возвраты,
  Обороты.СуммаОборот КАК Результат
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты(&НачПериода, &КонПериода, , Счет =
  &СчетКассы, , , КорСчет = &СчетПокупателей, ) КАК Обороты
ИТОГИ
  СУММА(Выручка),
  СУММА(Возвраты),
  СУММА(Результат)
ПО
  ОБЩИЕ

// Второй вариант.
ВЫБРАТЬ
  Обороты.Субконто1 КАК Покупатель,
  Обороты.СуммаОборотКт КАК Выручка,
  Обороты.СуммаОборотДт КАК Возвраты,
  Обороты.СуммаОборотКт - Обороты.СуммаОборотДт КАК Результат
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты(&НачПериода, &КонПериода, , Счет =
  &СчетПокупателей, , , КорСчет = &СчетКассы, ) КАК Обороты
ИТОГИ
  СУММА(Выручка),
  СУММА(Возвраты),
  СУММА(Результат)
```

```
ПО
   ОБЩИЕ
```

В запросе накладывается отбор по счетам *Касса* (основной счет) и *Покупатели* (корреспондирующий счет, но можно и наоборот: основным счетом запроса сделать счет *Покупатели*, а корреспондирующим – счет *Касса*). При этом данные группируются по первому субконто корреспондирующего счета (*КорСубконто1 – Контрагенты*). Отчетом анализируются дебетовый и кредитовый оборот счета *Касса* и разница между ними (*СуммаОборот*).

Результат выполнения запроса представлен на рис. 3.69.

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты((Записей в результате: 5)

Покупатель	Выручка	Возвраты	Результат
	490	80	410
	40	0	40
СтройТоргВсе	200	0	200
МонтажДоставка	150	0	150
Дисконт центр	100	80	20

Рис. 3.69. Обороты между счетами «Касса» и «Покупатели»

Особенности использования периодов и моментов времени при получении оборотов

Параметр *Период* виртуальной таблицы оборотов может принимать значения типа *Дата*, *МоментВремени* или *Граница*. Рассмотрим особенности использования этих объектов при получении оборотов.

В нашей демонстрационной конфигурации журнал проводок по счету *Касса* за июнь 2013 года содержит четыре проводки за первое и второе июня по 10 рублей каждая (табл. 3.5).

Таблица 3.5. Проводки по счету «Касса» за июнь 2013

Период	Счет дебета	Счет кредита	Сумма
01.06.2013 12:00:00	Касса	Покупатели	10.00
01.06.2013 23:59:59	Касса	Покупатели	10.00
Оборот за 01.06.2013			20,00
02.06.2013 0:00:00	Касса	Покупатели	10.00
02.06.2013 12:00:00	Касса	Покупатели	10.00
Оборот за 02.06.2013			20,00

Для анализа оборотов мы воспользуемся функцией, которая получает обороты на счете *Касса* за период с *НачПериода* по *КонПериода* при помощи следующего запроса (листинг 3.65).

Листинг 3.65. Пример получения оборотов регистра бухгалтерии

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        Обороты.СуммаОборот
  |ИЗ
  |        РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.Обороты(&НачПериода, &КонПериода,
  , Счет = &Счет, , , , ) КАК Обороты";

Запрос.УстановитьПараметр("НачПериода", НачПериода);
Запрос.УстановитьПараметр("КонПериода", КонПериода);
Запрос.УстановитьПараметр("Счет", ПланыСчетов.ОсновнойПланСчетов.Касса);
Результат = Запрос.Выполнить();
Если Результат.Пустой() Тогда
  Возврат 0;
КонецЕсли;
Обороты = Результат.Выбрать();
Обороты.Следующий();
Оборот = Обороты.СуммаОборот;
```

При получении оборотов следует иметь в виду, что таблица оборотов регистра бухгалтерии (также как и таблица оборотов регистра накопления) строится с начала секунды значения параметра *НачалоПериода* по конец секунды параметра *КонецПериода*, то есть *включая* границы заданного периода. Поэтому чтобы получить все обороты за указанный интервал, включая указанные дни, нам достаточно использовать функции *НачалоДня()* и *КонецДня()*, листинг 3.66.

Листинг 3.66. Пример установки начала и конца периода при получении оборотов

```
НачПериода = НачалоДня(Дата);
КонПериода = КонецДня(Дата);
```

Если мы установим значение поля ввода *Дата* обработки *Работа с виртуальными таблицами* как *01.06.2013*, то результатом работы функции будет значение *20*, соответствующее обороту за первое июня 2013 года.

В случаях, когда нужно получить итоги с точностью «до ссылки», требуется использовать параметр *Период* типа *МоментВремени*, полученный из даты документа и ссылки на документ. При этом необходимо иметь в виду, что (также как и для таблицы оборотов регистра накопления) данные получаются, *включая* записи движений самого документа.

Все вышесказанное справедливо также для виртуальных таблиц остатков и оборотов, оборотов ДтКт и таблицы движений с субконто. Итоги по ним рассчитываются, включая граничные периоды.

Получение оборотов между корреспондирующими счетами

Для получения оборотов между корреспондирующими счетами в разрезе субконто и измерений используется виртуальная таблица *ОборотыДтКт()*. Таблица присутствует только у регистра с поддержкой корреспонденции и позволяет узнать оборот в дебет

счета с кредита счета (субконто, измерения). Таблица оборотов ДтКт может быть использована при разработке таких отчетов, как «Шахматный баланс (шахматка)», «Сводные проводки» и др.

Основное отличие таблицы оборотов ДтКт от таблицы оборотов состоит в том, что таблица оборотов ДтКт позволяет анализировать обороты между счетами, где заранее известно, какой счет дебетуется, а какой кредитуется. А таблица оборотов кроме этого позволяет анализировать обороты по счету (без указания второго). При этом одним обращением к таблице можно получить как дебетовые, так и кредитовые корреспонденции.

Виртуальная таблица *ОборотыДтКт* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения балансового измерения регистра, в разрезе которого посчитан оборот. Количество таких полей равно количеству балансовых измерений, определенных для регистра как объекта конфигурации;
- *<Имя измерения>Дт* – поле, содержащее значения дебетового небалансового измерения регистра, в разрезе которого посчитан оборот;
- *<Имя измерения>Кт* – поле, содержащее значения кредитового небалансового измерения регистра, в разрезе которого посчитан оборот;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*) с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя ресурса>Оборот* – поле, содержащее оборот балансового ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *Оборот*. Содержит разницу оборотов (оборот дебета минус оборот кредита);
- *<Имя ресурса>ОборотДт* – поле, содержащее дебетовый оборот небалансового ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *ОборотДт*;
- *<Имя ресурса>ОборотКт* – поле, содержащее кредитовый оборот небалансового ресурса регистра по именам ресурсов, как они заданы в конфигураторе, с добавлением слова *ОборотКт*;
- *СчетДт* – имеет тип *ПланСчетовСсылка.<имя>*. Содержит дебетуемый счет, в разрезе которого посчитан оборот;
- *СчетКт* – имеет тип *ПланСчетовСсылка.<имя>*. Содержит кредитуемый счет, в разрезе которого посчитан оборот;
- *СубконтоДт<Номер субконто>* – имеет тип *Характеристика.<имя>*. Содержит значение субконто дебета, в разрезе которого посчитан оборот. Количество полей *Субконто* зависит от максимального количества субконто на счете плана счетов. Номер субконто начинается с *1*. Набор и порядок субконто определяются параметром *СубконтоДт*;
- *СубконтоКт<Номер субконто>* – имеет тип *Характеристика.<имя>*. Содержит

значение субконто кредита, в разрезе которого посчитан оборот. Количество полей *Субконто* зависит от максимального количества субконто на счете плана счетов. Номер субконто начинается с *1*. Набор и порядок субконто определяются параметром *СубконтоКт*;

- *НомерСтроки* – имеет тип *Число*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Запись*. Содержит значение поля *НомерСтроки* записи движения регистра;
- *Период* – имеет тип *Дата*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Секунда*, *Минута*, *Час*, *Регистратор* или *Запись*. Данное поле содержит начальную дату и время периода, к которому относится оборот регистра;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Регистратор* или *Запись*. Данное поле содержит ссылку на документ-регистратор, к которому относится оборот регистра.

Параметры виртуальной таблицы оборотов ДтКт позволяют задать условие отбора данных из информационной базы. Параметры следует задавать строго в порядке их описания:

- *НачалоПериода*, *КонецПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Период времени, за который будут получены обороты. Если параметры не заданы, то будут получены все обороты регистра;
- *Периодичность* – содержит конструкцию языка запросов. Позволяет задать дополнительную группировку данных по стандартным периодам. Возможные значения: *Период*, *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Час*, *Минута*, *Секунда*, *Регистратор*, *Запись*. Если периодичность не задана или задана как *Период,* дополнительная группировка не выполняется;
- *УсловиеСчетаДт* – этот параметр содержит конструкцию языка запросов. Отбор по дебетуемому счету, как правило, содержит следующие условия: *СчетДт = (В ИЕРАРХИИ, В) &СчетДт*;
- *СубконтоДт* – имеет тип *ПланВидовХарактеристикСсылка.<имя>* или содержит массив значений этого типа. Задает набор и порядок субконто дебета, которыми можно оперировать в запросе. А также служит для отбора дебетовых оборотов регистра по видам субконто. Если параметр задан, то будут выбираться данные только по тем дебетуемым счетам, у которых определены все указанные виды субконто. Если параметр не задан, то ограничений по видам субконто нет. Субконто дебета определяются позиционно по соответствующему счету;
- *УсловиеСчетаКт* – этот параметр содержит конструкцию языка запросов. Отбор по кредитуемому счету, как правило, содержит следующие условия: *СчетКт = (В ИЕРАРХИИ, В) &СчетДт*;
- *СубконтоКт* – имеет тип *ПланВидовХарактеристикСсылка.<имя>* или содержит массив значений этого типа. Задает набор и порядок субконто кредита, которыми

можно оперировать в запросе. А также служит для отбора кредитовых оборотов регистра по видам субконто. Если параметр задан, то будут выбираться данные только по тем кредитуемым счетам, у которых определены все указанные виды субконто. Если параметр не задан, то ограничений по видам субконто нет. Субконто кредита определяются позиционно по соответствующему счету;

- *Условие* – содержит конструкцию языка запросов. Позволяет устанавливать отбор данных виртуальной таблицей по значениям субконто и измерений регистра бухгалтерии.

Рассмотрим примеры построения запросов к таблице оборотов ДтКт регистра бухгалтерии. Все запросы обращаются к итогам за весь период, который есть в регистре (не указываются параметры *НачалоПериода* и *КонецПериода*).

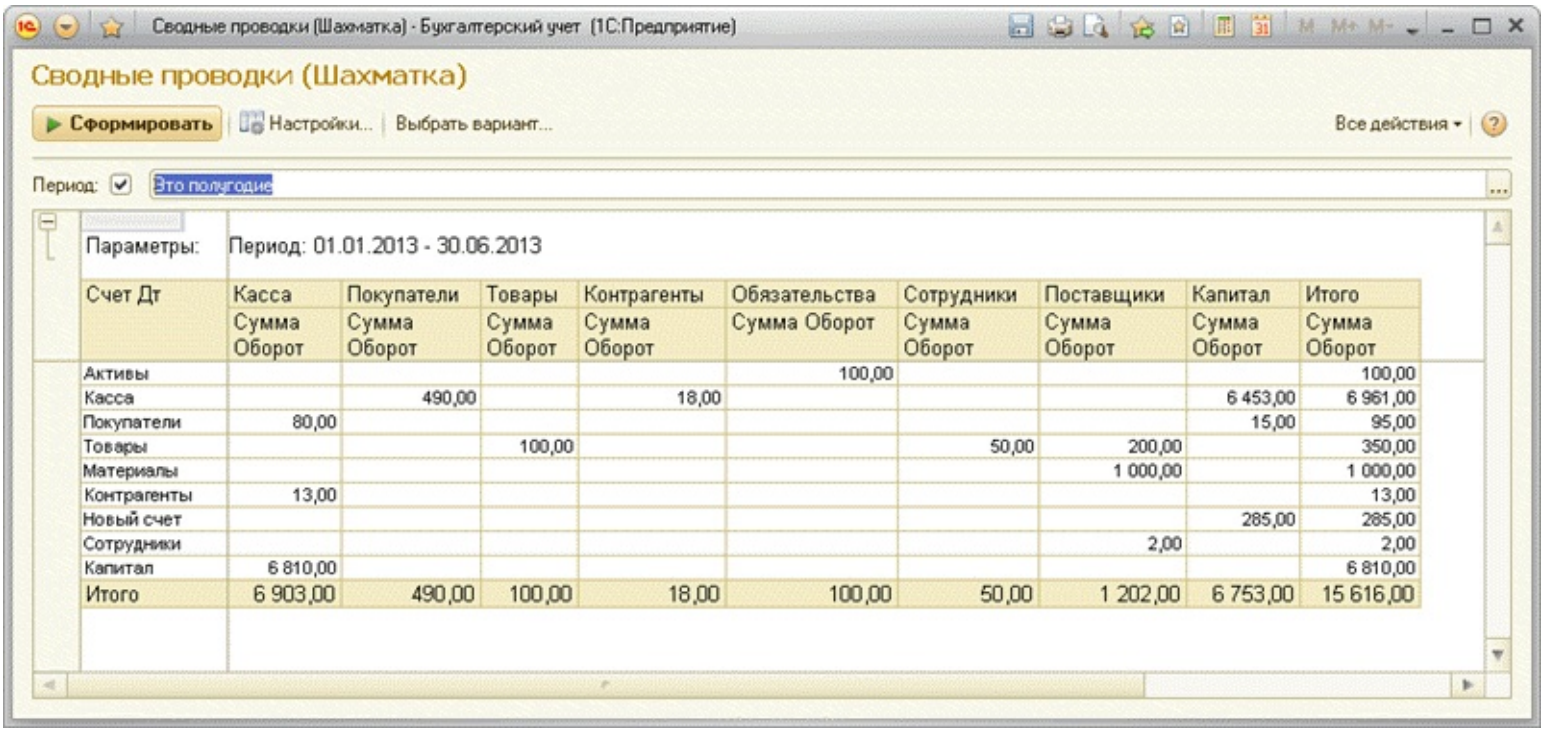
Пример 1

Самым распространенным примером отчета, который формируется с использованием таблицы оборотов ДтКт, является отчет «Сводные проводки», отражающий сводные обороты между дебетуемыми и кредитуемыми счетами. Для этого используется следующий запрос (листинг 3.67).

Листинг 3.67. Запрос для формирования отчета «Сводные проводки»

```
ВЫБРАТЬ
  ОборотыДтКт.СчетДт,
  ОборотыДтКт.СчетКт,
  ОборотыДтКт.СуммаОборот
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт КАК ОборотыДтКт
```

Если использовать этот запрос как источник данных для отчета в системе компоновки данных и выводить отчет в виде таблицы, в строках которой находятся дебетуемые счета, в колонках – кредитуемые счета, а в ячейках таблицы – обороты между ними, то мы получим следующий результат (рис. 3.70).



Счет Дт	Касса	Покупатели	Товары	Контрагенты	Обязательства	Сотрудники	Поставщики	Капитал	Итого
	Сумма Оборот	Сумма Оборот	Сумма Оборот	Сумма Оборот	Сумма Оборот	Сумма Оборот	Сумма Оборот	Сумма Оборот	Сумма Оборот
Активы					100,00				100,00
Касса		490,00		18,00				6 453,00	6 961,00
Покупатели	80,00							15,00	95,00
Товары			100,00			50,00	200,00		350,00
Материалы							1 000,00		1 000,00
Контрагенты	13,00								13,00
Новый счет								285,00	285,00
Сотрудники							2,00		2,00
Капитал	6 810,00								6 810,00
Итого	6 903,00	490,00	100,00	18,00	100,00	50,00	1 202,00	6 753,00	15 616,00

Рис. 3.70. Отчет «Сводные проводки»

Пример 2

Рассмотрим следующую задачу. Например, с помощью таблицы оборотов ДтКт можно узнать, на какую сумму, в каком количестве и каких товаров поставлено различными поставщиками. Для этого нужно получить обороты за период в дебет счета *Товары* с кредита счета *Поставщики* в разрезе товаров (*СубконтоДт1* – первое субконто на дебетуемом счете) и контрагентов (*СубконтоКт1* – первое субконто на кредитуемом счете), листинг 3.68.

Листинг 3.68. Обороты между счетами «Товары» и «Поставщики»

```
ВЫБРАТЬ
  ОборотыДтКт.СубконтоДт1 КАК Товар,
  ОборотыДтКт.СубконтоКт1 КАК Поставщик,
  ОборотыДтКт.СуммаОборот КАК Сумма,
  ОборотыДтКт.КоличествоОборотДт КАК Количество
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт( , , , СчетДт = &СчетТоваров,
  , СчетКт = &СчетПоставщиков, , ) КАК ОборотыДтКт
```

В запросе накладывается отбор по счетам *Товары* (счет дебета) и *Поставщики* (счет кредита). Обратите внимание, что при получении оборотов между счетами в суммовом выражении мы обращаемся в запросе к полю *СуммаОборот*, т. к. ресурс *Сумма* является балансовым ресурсом регистра бухгалтерии, а при получении оборотов в количественном выражении мы обращаемся в запросе к полю *КоличествоОборотДт*, т. к. ресурс *Количество* – небалансовый ресурс.

Результат выполнения запроса представлен на рис. 3.71.

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт((Записей в результате: 5)

Товар	Поставщик	Сумма	Количество
Паркер	СтройТоргВсе	100	10
Цветной		10	
Цветной	МонтажДоставка	15	3
Паркер	Дисконт центр	50	5
Школьная	Дисконт центр	25	5

Рис. 3.71. Обороты между счетами «Товары» и «Поставщики»

Периодичность таблиц оборотов

Виртуальные таблицы – таблица оборотов, таблица оборотов и остатков и таблица оборотов ДтКт – содержат параметр *Периодичность*. При помощи этого параметра можно задать дополнительный разворот оборотов по периодичности. Параметр может принимать одно из следующих значений (табл. 3.6).

Таблица 3.6. Значения, которые может принимать параметр «Периодичность»

Период	Комментарий
Период или пусто	Не разворачивать
Год	Разворачивать по годам
Полугодие	Разворачивать по полугодиям
Квартал	Разворачивать по кварталам
Месяц	Разворачивать по месяцам
Декада	Разворачивать по декадам
Неделя	Разворачивать по неделям
День	Разворачивать по дням
Час	Разворачивать по часам
Минута	Разворачивать по минутам
Секунда	Разворачивать по секундам
Регистратор	Разворачивать по регистраторам
Запись	Разворачивать по записям (движениям регистра)
Авто	Периодичность определяется автоматически, в зависимости от используемых в запросе полей периодов

Значение по умолчанию, если параметр не заполнен – *Период*.

При установке в параметре *Периодичность* значения, отличного от *Период*, в таблице появляются новые поля, по которым можно сгруппировать данные.

В случае, если периодичность задана и не равна *Период*, в таблице появляется поле

Период, содержащее дату начала периода. Например, если выбрана периодичность *Месяц*, то в таблице появятся первые числа всех месяцев, за которые были обороты.

Рассмотрим использование периодичности при построении виртуальной таблицы оборотов ДтКт. Например, с помощью следующего запроса можно получить все обороты между счетами *Касса* и *Покупатели* с разворотом по месяцам (листинг 3.69).

Листинг 3.69. Обороты между счетами с периодичностью «Месяц»

```
ВЫБРАТЬ
  ОборотыДтКт.Период,
  ОборотыДтКт.СчетДт,
  ОборотыДтКт.СчетКт,
  ОборотыДтКт.СуммаОборот
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт( , , Месяц, СчетДт =
  &СчетКассы, , СчетКт = &СчетПокупателей, , ) КАК ОборотыДтКт

УПОРЯДОЧИТЬ ПО
  Период
```

При периодичности запроса *Месяц* поле *Период* будет содержать дату начала каждого месяца. Как мы видим на рисунке внизу, часть проводок по корреспондирующим счетам *Касса* и *Покупатели* сделаны в мае 2013 года на сумму 450, а часть – в июне 2013 года, на сумму 40 (рис. 3.72).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт((Записей в результате: 2)

Период	СчетДт	СчетКт	СуммаОборот
01.05.2013 0:00:00	Касса	Покупатели	450
01.06.2013 0:00:00	Касса	Покупатели	40

Рис. 3.72. Обороты между счетами «Касса» и «Покупатели» с периодичностью «Месяц»

Предположим, мы хотим детализировать обороты между этими счетами с точностью до регистратора (листинг 3.70).

Листинг 3.70. Обороты между счетами с периодичностью «Регистратор»

```
ВЫБРАТЬ
  ОборотыДтКт.Период,
  ОборотыДтКт.Регистратор,
  ОборотыДтКт.СчетДт,
  ОборотыДтКт.СчетКт,
  ОборотыДтКт.СуммаОборот
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт( , , Регистратор, СчетДт =
  &СчетКассы, , СчетКт = &СчетПокупателей, , ) КАК ОборотыДтКт

УПОРЯДОЧИТЬ ПО
  Период
```

В случае, если периодичность равна *Регистратор*, в таблице, помимо поля *Период*, появляется поле *Регистратор*, содержащее ссылку на документ-регистратор.

Например, мы видим, какими документами были проведены операции между счетами *Касса* и *Покупатели* за июнь 2013 года (рис. 3.73).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт((Записей в результате: 5)

Период	Регистратор	СчетДт	СчетКт	СуммаОборот
01.05.2013 12:00:00	Операция 000000007 от 01.05.2013 12:00:00	Касса	Покупатели	450
01.06.2013 12:00:00	Операция 000000011 от 01.06.2013 12:00:00	Касса	Покупатели	10
01.06.2013 23:59:59	Операция 000000012 от 01.06.2013 23:59:59	Касса	Покупатели	10
02.06.2013 0:00:00	Операция 000000013 от 02.06.2013 0:00:00	Касса	Покупатели	10
02.06.2013 12:00:01	Операция 000000014 от 02.06.2013 12:00:01	Касса	Покупатели	10

Рис. 3.73. Обороты между счетами «Касса» и «Покупатели» с периодичностью «Регистратор»

Если мы хотим еще более детализировать информацию с точностью до проводки (записи движения регистра), то при получении оборотов следует использовать периодичность *Запись* (листинг 3.71).

Листинг 3.71. Обороты между счетами с периодичностью «Запись»

```
ВЫБРАТЬ
  ОборотыДтКт.Период,
  ОборотыДтКт.Регистратор,
  ОборотыДтКт.НомерСтроки,
  ОборотыДтКт.СчетДт,
  ОборотыДтКт.СчетКт,
  ОборотыДтКт.СуммаОборот
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт( , , Запись, СчетДт =
  &СчетКассы, , СчетКт = &СчетПокупателей, , ) КАК ОборотыДтКт

УПОРЯДОЧИТЬ ПО
  Период
```

В случае, если периодичность равна *Запись*, в таблицах, помимо полей *Период* и *Регистратор*, появляется поле *НомерСтроки*, содержащее порядковый номер записи движения регистра. Например, мы видим, из каких проводок складывался оборот за май 2013 года между счетами *Касса* и *Покупатели* (рис. 3.74).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОборотыДтКт((Записей в результате: 7)

Период	Регистратор	НомерСтроки	СчетДт	СчетКт	СуммаОборот
01.05.2013 12:00:00	Операция 000000007 от 01.05.2013 12:00:00	1	Касса	Покупатели	100
01.05.2013 12:00:00	Операция 000000007 от 01.05.2013 12:00:00	2	Касса	Покупатели	150
01.05.2013 12:00:00	Операция 000000007 от 01.05.2013 12:00:00	3	Касса	Покупатели	200
01.06.2013 12:00:00	Операция 000000011 от 01.06.2013 12:00:00	1	Касса	Покупатели	10
01.06.2013 23:59:59	Операция 000000012 от 01.06.2013 23:59:59	1	Касса	Покупатели	10
02.06.2013 0:00:00	Операция 000000013 от 02.06.2013 0:00:00	1	Касса	Покупатели	10
02.06.2013 12:00:01	Операция 000000014 от 02.06.2013 12:00:01	1	Касса	Покупатели	10

Рис. 3.74. Обороты между счетами «Касса» и «Покупатели» с периодичностью «Запись»

Все вышесказанное справедливо также для виртуальных таблиц оборотов, остатков и оборотов. Однако таблица остатков и оборотов имеет свои особенности, которые будут рассмотрены в разделе «Периодичность таблицы остатков и оборотов».

Получение остатков и оборотов

Для получения остатков и оборотов по счету в разрезе субконто и измерений используется виртуальная таблица *ОстаткиИОбороты()*. Таблица остатков и оборотов позволяет получить обороты по счету, аналогично таблице оборотов, однако не предоставляет возможности анализировать обороты с корреспондирующими счетами, субконто, измерениями. Таблица остатков и оборотов может быть использована при разработке оборотно-сальдовых ведомостей и других отчетов, где для каждой строки необходимо показать остаток на начало периода, обороты за период и остаток на конец периода.

Виртуальная таблица *ОстаткиИОбороты* имеет следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра, в разрезе которого посчитаны остатки и обороты. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*) с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя ресурса>НачальныйОстаток*, *<Имя ресурса>НачальныйОстатокДт*, *<Имя ресурса>НачальныйОстатокКт*, *<Имя ресурса>НачальныйРазвернутыйОстатокДт, <Имя ресурса>НачальныйРазвернутыйОстатокКт* – эти поля имеют тип *Число*. Поля аналогичны полям таблицы остатков (подробнее можно прочитать в разделе «Получение остатков»). Если значение параметра виртуальной таблицы *Периодичность* не задано или задано как *Период*, то начальные остатки рассчитываются на дату начала интервала, указанного в параметре таблицы остатков и оборотов *НачалоПериода*, иначе – на начало периода, к которому относится данная запись;
- *<Имя ресурса>КонечныйОстаток*, *<Имя ресурса>КонечныйОстатокДт*, *<Имя ресурса>КонечныйОстатокКт*, *<Имя ресурса>КонечныйРазвернутыйОстатокДт*, *<Имя ресурса>КонечныйРазвернутыйОстатокКт* – эти поля имеют тип *Число*. Поля аналогичны полям таблицы остатков (подробнее можно прочитать в разделе «Получение остатков»). Если значение параметра виртуальной таблицы *Периодичность* не задано или задано как *Период*, то начальные остатки рассчитываются на дату начала интервала, указанного в параметре таблицы остатков и оборотов *КонецПериода*, иначе – на конец периода, к которому относится данная запись;
- *<Имя ресурса>Оборот, <Имя ресурса>ОборотДт, <Имя ресурса>ОборотКт* – эти поля имеют тип *Число.* Поля аналогичны полям таблицы оборотов, подробнее можно прочитать в разделе «Получение оборотов»;
- *Счет* – имеет тип *ПланСчетовСсылка.<имя>*. Содержит счет, в разрезе которого посчитаны остатки и обороты;

- *Субконто<Номер субконто>* – имеет тип *Характеристика.<имя>*. Содержит значение субконто, в разрезе которого посчитаны остатки и обороты. Количество полей *Субконто* зависит от максимального количества субконто на счете плана счетов. Номер субконто начинается с *1*. Набор и порядок субконто определяются параметром *Субконто*;
- *НомерСтроки* – имеет тип *Число*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Запись*. Содержит значение поля *НомерСтроки* записи движения регистра;
- *Период* – имеет тип *Дата*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Секунда*, *Минута*, *Час*, *Регистратор* или *Запись*. Данное поле содержит начальную дату и время периода, к которому относится оборот регистра;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Существует только в случаях, если указано значение параметра виртуальной таблицы оборотов *Периодичность*: *Регистратор* или *Запись*. Данное поле содержит ссылку на документ-регистратор, к которому относится оборот регистра.

Параметры виртуальной таблицы остатков и оборотов позволяют задать условие отбора данных из информационной базы. Параметры следует задавать строго в порядке их описания:

- *НачалоПериода*, *КонецПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*. Период времени, за который будут получены остатки и обороты. Если параметры не заданы, то будут получены все итоги по регистру;
- *Периодичность* – содержит конструкцию языка запросов. Позволяет задать дополнительную группировку данных по стандартным периодам. Возможные значения: *Период*, *Год*, *Полугодие*, *Квартал*, *Месяц*, *Декада*, *Неделя*, *День*, *Час*, *Минута*, *Секунда*, *Регистратор*, *Запись*. Если периодичность не задана или задана как *Период*, дополнительной группировки не выполняется;
- *МетодДополнения* – этот параметр содержит конструкцию языка запросов. Возможны два значения: *Движения* или *ДвиженияИГраницыПериода*. Управляет включением в отчет периодов, не имеющих оборотов, но имеющих остатки;
- *УсловиеСчета* – содержит конструкцию языка запросов. Позволяет установить фильтр по счету или счетам. Как правило, содержит следующие условия: *Счет = (В ИЕРАРХИИ, В) &Счет*;
- *Субконто* – имеет тип *ПланВидовХарактеристикСсылка.<имя>* или содержит массив значений этого типа. Задает набор и порядок субконто, которыми можно оперировать в запросе. А также служит для отбора оборотов регистра по видам субконто. Если параметр задан, то будут выбираться данные только по тем счетам, у которых определены все указанные виды субконто. Если параметр не задан, то ограничений по видам субконто нет. Субконто определяются позиционно по соответствующему счету;
- *Условие* – содержит конструкцию языка запросов. Позволяет устанавливать отбор

данных виртуальной таблицей по значениям субконто и измерений регистра бухгалтерии.

Рассмотрим примеры построения запросов к таблице оборотов регистра бухгалтерии.

Самым распространенным примером отчета, который формируется с использованием таблицы остатков и оборотов, является отчет «Оборотно-сальдовая ведомость», текст которого приведен в листинге 3.72.

Листинг 3.72. Запрос для формирования отчета «Оборотно-сальдовая ведомость»

```
ВЫБРАТЬ
  ОстаткиИОбороты.Счет КАК Счет,
  ОстаткиИОбороты.СуммаНачальныйОстатокДт КАК НачОстДт,
  ОстаткиИОбороты.СуммаНачальныйОстатокКт КАК НачОстКт,
  ОстаткиИОбороты.СуммаОборотДт КАК ОборотДт,
  ОстаткиИОбороты.СуммаОборотКт КАК ОборотКт,
  ОстаткиИОбороты.СуммаКонечныйОстатокДт КАК КонОстДт,
  ОстаткиИОбороты.СуммаКонечныйОстатокКт КАК КонОстКт
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты(&НачПериода, &КонПериода,
  , , , , ) КАК ОстаткиИОбороты

УПОРЯДОЧИТЬ ПО
  ОстаткиИОбороты.Счет.Код
ИТОГИ
  СУММА(НачОстДт),
  СУММА(НачОстКт),
  СУММА(ОборотДт),
  СУММА(ОборотКт),
  СУММА(КонОстДт),
  СУММА(КонОстКт)
ПО
  Счет ИЕРАРХИЯ
```

Если использовать этот запрос как источник данных для отчета в системе компоновки данных (без секции описания итогов) и выводить отчет в виде иерархических группировок по счетам, то мы получим следующий результат (рис. 3.75).

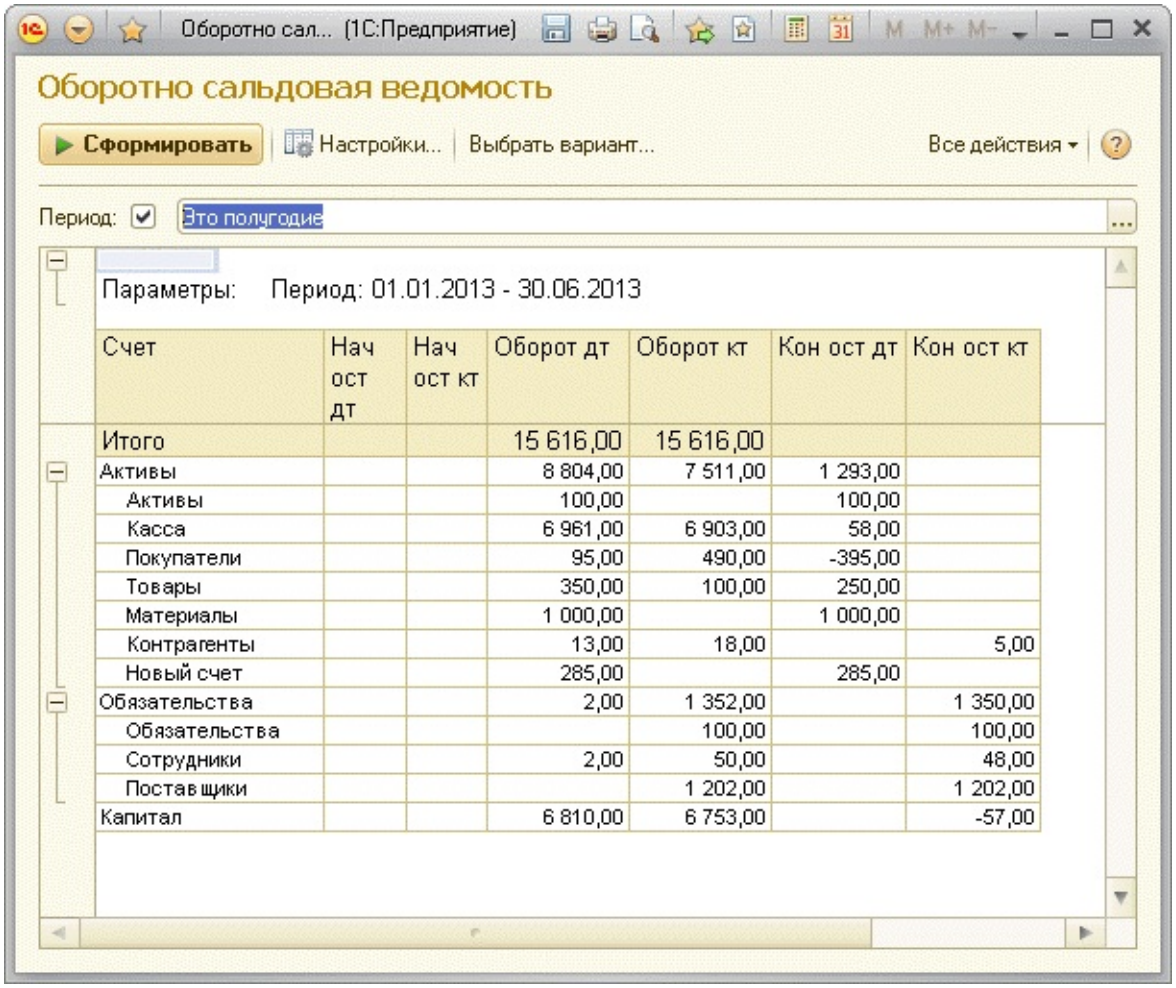


Рис. 3.75. Отчет «Оборотно-сальдовая ведомость»

ВНИМАНИЕ!

Таблица остатков и оборотов универсальна и содержит поля, необходимые во многих отчетах. Однако платой за универсальность всегда является производительность. Формирование таблицы может включать в себя выполнение до трех запросов. Использование таблицы имеет смысл только в случаях, когда в одной группировке отчета требуется получение и остатков на начало, и оборотов и остатков на конец периода. Если же этого не требуется, то, возможно, более производительным будет вариант с несколькими запросами к таблице остатков или к таблице оборотов.

Периодичность таблицы остатков и оборотов

Для получения дополнительного разворота оборотов по периодичности при построении таблицы остатков и оборотов используется параметр *Периодичность*. Его назначение и использование аналогично другим таблицам оборотов.

подробнее

Раздел «Периодичность таблиц оборотов».

При построении таблиц оборотов, оборотов ДтКт с определенной периодичностью в них попадают только те периоды, которые содержали обороты.

Таблица остатков и оборотов позволяет в одной группировке запроса получить и остатки (входящие и исходящие), и обороты. Поэтому в таблице присутствует еще один параметр

МетодДополнения, который может принимать два значения: *ДвиженияИГраницыПериода* (по умолчанию) и *Движения*. В первом случае в результат отчета включаются периоды, за которые были обороты, и, кроме того, еще и границы интервала, если на эти даты были остатки. Во втором случае в результат отчета включаются только те периоды, за которые были обороты.

Рассмотрим эти отличия на примере запроса, получающего остатки и обороты по счету товаров за период с *01.06.2013* по *30.06.2013* включительно (листинг 3.73).

Листинг 3.73. Остатки и обороты по счету «Товары»

```
ВЫБРАТЬ
  ОстаткиИОбороты.Период,
  ОстаткиИОбороты.СуммаНачальныйОстаток КАК НачОстаток,
  ОстаткиИОбороты.СуммаОборотДт КАК ОборотДт,
  ОстаткиИОбороты.СуммаОборотКт КАК ОборотКт,
  ОстаткиИОбороты.СуммаКонечныйОстаток КАК КонОстаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты(&НачПериода,
  КОНЕЦПЕРИОДА(&КонПериода, День), День, ДвиженияИГраницыПериода, Счет = &СчетТоваров, , )
  КАК ОстаткиИОбороты

УПОРЯДОЧИТЬ ПО
  Период
```

Если параметр *МетодДополнения* принимает значение *ДвиженияИГраницыПериода* или если он не задан, то в отчет включается дата начала интервала отбора итогов (01.06.2013, так как на эту дату был входящий остаток), и дата конца интервала (30.06.2013, по той же причине), рис. 3.76.

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты((Записей в результате: 4)

Период	НачОстаток	ОборотДт	ОборотКт	КонОстаток
01.06.2013 0:00:00	160			160
23.06.2013 0:00:00	160	75		235
25.06.2013 0:00:00	235	15		250
30.06.2013 0:00:00	250			250

Рис. 3.76. Остатки и обороты по счету «Товары»

В случае, если параметр *МетодДополнения* принимает значения *Движения* (листинг 3.74), в отчет попадают только те даты, за которые были обороты по счету *Товары* (рис. 3.77).

Листинг 3.74. Остатки и обороты по счету «Товары»

```
ВЫБРАТЬ
  ОстаткиИОбороты.Период,
  ОстаткиИОбороты.СуммаНачальныйОстаток КАК НачОстаток,
  ОстаткиИОбороты.СуммаОборотДт КАК ОборотДт,
  ОстаткиИОбороты.СуммаОборотКт КАК ОборотКт,
  ОстаткиИОбороты.СуммаКонечныйОстаток КАК КонОстаток
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты(&НачПериода,
  КОНЕЦПЕРИОДА(&КонПериода, День), День, Движения, Счет = &СчетТоваров, , ) КАК
```

```
  ОстаткиИОбороты

УПОРЯДОЧИТЬ ПО
  Период
```

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты((Записей в результате: 2)

Период	НачОстаток	ОборотДт	ОборотКт	КонОстаток
23.06.2013 0:00:00	160	75		235
25.06.2013 0:00:00	235	15		250

Рис. 3.77. Остатки и обороты по счету «Товары»

Развернутые остатки

В некоторых случаях требуется получить не просто остатки на счетах, а развернутые остатки, которые считаются развернуто для измерений и субконто, используемых в запросе. Например, на активно-пассивных счетах может быть как дебетовый, так и кредитовый остаток. В этом случае важно видеть не общий остаток, а остаток с разворотом по аналитике, учитываемой на счете.

Например, в нашей демонстрационной конфигурации существует активно-пассивный счет *Контрагенты*, по которому проведены следующие операции (табл. 3.7).

Таблица 3.7. Проводки по счету «Контрагенты»

№	Период	Счет дт	Аналитика	Счет кт	Аналитика	Сумма
1	31.05.2013	Касса		Контрагенты	Иванов	8
2	31.05.2013	Контрагенты	Петров	Касса		3
3	20.06.2013	Контрагенты	Иванов	Касса		10
4	20.06.2013	Касса		Контрагенты	Петров	10

Предположим, требуется узнать за период с *01.06.2013* по *30.06.2013* начальный остаток, обороты и конечный остаток на счете *Контрагенты*. Причем требуется развернуть остатки по контрагентам, т. е. получить остатки в разрезе аналитики (*Субконто1*), ведущейся на счете.

Это можно сделать с помощью следующего запроса (листинг 3.75).

Листинг 3.75. Получение развернутых остатков из таблицы остатков и оборотов

```
ВЫБРАТЬ
  ОстаткиИОбороты.Счет КАК Счет,
  ОстаткиИОбороты.Субконто1 КАК Контрагент,
  ОстаткиИОбороты.СуммаНачальныйОстатокДт КАК СвернутыйНачальныйДт,
  ОстаткиИОбороты.СуммаНачальныйОстатокКт КАК СвернутыйНачальныйКт,
  ОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокДт КАК РазвернутыйНачальныйДт,
  ОстаткиИОбороты.СуммаНачальныйРазвернутыйОстатокКт КАК РазвернутыйНачальныйКт,
  ОстаткиИОбороты.СуммаОборотДт КАК ОборотДт,
  ОстаткиИОбороты.СуммаОборотКт КАК ОборотКт,
  ОстаткиИОбороты.СуммаКонечныйОстатокДт КАК СвернутыйКонечныйДт,
```

```
  ОстаткиИОбороты.СуммаКонечныйОстатокКт КАК СвернутыйКонечныйКт,
  ОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокДт КАК РазвернутыйКонечныйДт,
  ОстаткиИОбороты.СуммаКонечныйРазвернутыйОстатокКт КАК РазвернутыйКонечныйКт
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты(&НачПериода,
  КОНЕЦПЕРИОДА(&КонПериода, День), , , Счет = &Счет, , ) КАК ОстаткиИОбороты
ИТОГИ
  СУММА(СвернутыйНачальныйДт),
  СУММА(СвернутыйНачальныйКт),
  СУММА(РазвернутыйНачальныйДт),
  СУММА(РазвернутыйНачальныйКт),
  СУММА(СвернутыйКонечныйДт),
  СУММА(СвернутыйКонечныйКт),
  СУММА(РазвернутыйКонечныйДт),
  СУММА(РазвернутыйКонечныйКт),
  СУММА(ОборотДт),
  СУММА(ОборотКт)
ПО
  Счет
```

Развернутый остаток не имеет смысла без критерия, по которому он будет «разворачиваться». В практике учета чаще всего получают развернутые остатки счета с разворотом по аналитике этого счета. Поэтому в выборку запроса включено поле *Субконто1* (в качестве первого субконто на счете *Контрагенты* учитываются *Контрагенты*).

Поля развернутых остатков, присутствующие в виртуальных таблицах в детальной выборке запроса, полностью повторяют содержимое полей остатков (свернутых). Развернутые остатки имеют смысл только при наличии в запросе итогов. Поэтому в запросе мы рассчитываем итоги по полю *Счет,* суммируя все числовые поля.

В результате будет получена следующая динамика остатков и оборотов по счету *Контрагенты* за июнь 2013 года (рис. 3.78).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ОстаткиИОбороты((Записей в результате: 3)

Счет	Контрагент	СвернутыйНачальныйДт	СвернутыйНачальныйКт	РазвернутыйНачальныйДт	РазвернутыйНачальныйКт
Контрагенты		0	5	3	8
Контрагенты	Иванов	0	8	0	8
Контрагенты	Петров	3	0	3	0

ОборотДт	ОборотКт	СвернутыйКонечныйДт	СвернутыйКонечныйКт	РазвернутыйКонечныйДт	РазвернутыйКонечныйКт
10	10	0	5	2	7
10	0	2	0	2	0
0	10	0	7	0	7

Рис. 3.78. Остатки, развернутые остатки и обороты по счету «Контрагенты»

Мы видим, что на *01.06.2013* контрагент *Иванов* был нашим кредитором, и мы были ему должны *8* рублей. В текущем отчетном периоде мы заплатили ему *10* рублей. Переплатили *2* рубля, и теперь он наш дебитор (он нам должен *2* рубля). Контрагент *Петров,* наоборот, был дебитором (был нам должен *3* рубля), в текущем отчетном

периоде он сделал платеж на *10* рублей. Переплатил нам *7* рублей. Теперь мы ему должны *7* рублей.

В результате мы имеем одинаковый свернутый начальный (*5=8-3*) и конечный (*5=7-2*) кредитовый остаток на счете *Контрагенты*. Но развернутые остатки, подсчитанные отдельно по дебету и отдельно по кредиту, на начало периода равны *3* (Дт) и *8* (Кт), а на конец периода равны *2* (Дт) и *7* (Кт).

Можно было и не использовать в запросе поле *Счет*, мы могли бы включить в детальную выборку поле *Субконто1* и подвести общие итоги. Но в данном случае итоги разворачиваются по всем полям детальной выборки (в нашем примере по полям *Счет* и *Субконто1*) и рассчитываются по всем числовым полям.

Также развернутые остатки можно получить из виртуальной таблицы остатков.

Получение движений с субконто

Виртуальная таблица движений с субконто включает в себя все поля основной таблицы регистра бухгалтерии и таблицы значений субконто и получается их соединением, поэтому одна из отличительных особенностей этой таблицы состоит в том, что в нее попадают неактивные записи.

Таблица движений с субконто позволяет получить информацию о записях регистра бухгалтерии вместе со значениями субконто. Эта таблица может быть использована при разработке таких отчетов, как «Карточка счета/субконто», «Отчет по проводкам», а также служит основой для формы списка регистра бухгалтерии.

Виртуальная таблица движений с субконто содержит следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения балансового измерения регистра. Количество таких полей равно количеству балансовых измерений, определенных для регистра как объекта конфигурации;
- *<Имя измерения>Дт* – поле, содержащее значения дебетового небалансового измерения регистра с именем, заданным в конфигурации;
- *<Имя измерения>Кт* – поле, содержащее значения кредитового небалансового измерения регистра с именем, заданным в конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, являющихся разделителями (режим разделения данных – *Разделять*) с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя реквизита>* – поле, содержащее значения реквизита регистра с именем, заданным в конфигурации;
- *<Имя ресурса>* – поле, содержащее значения балансового ресурса регистра по именам ресурсов, как они заданы в конфигураторе;
- *<Имя ресурса>Дт* – поле, содержащее значения дебетового небалансового ресурса

регистра по именам ресурсов, как они заданы в конфигураторе;

- *<Имя ресурса>ОборотКт* – поле, содержащее значения кредитового небалансового ресурса регистра по именам ресурсов, как они заданы в конфигураторе;
- *Активность* – имеет тип *Булево*. Содержит признак активности записи и влияния на получение итогов регистра;
- *ВидСубконтоДт<Номер субконто>* – имеет тип *ПланВидовХарактеристикСсылка.<имя>*. Количество таких полей зависит от максимального количества субконто на счете плана счетов. Номер вида субконто начинается с *1*. Набор и порядок видов субконто соответствуют набору и порядку видов субконто у счета дебета;
- *ВидСубконтоКт<Номер субконто>* – имеет тип *ПланВидовХарактеристикСсылка.<имя>*. Количество таких полей зависит от максимального количества субконто на счете плана счетов. Номер вида субконто начинается с *1*. Набор и порядок видов субконто соответствуют набору и порядку видов субконто у счета кредита;
- *Момент времени* – виртуальное поле, не хранится в информационной базе. Содержит объект *МоментВремени* (который включает в себя дату и ссылку на документ-регистратор).
- *Период* – имеет тип *Дата*. Содержит дату записи. Совместно с полями *Регистратор* и *НомерСтроки* определяет положение данной записи на временной оси;
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Содержит ссылку на документ, которому подчинена данная запись;
- *НомерСтроки* – имеет тип *Число*. Содержит уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*;
- *СубконтоДт<Номер субконто>* – имеет тип *Характеристика.<имя>*, тип этого поля определяется типом субконто. Содержит значение субконто дебета. Количество таких полей зависит от максимального количества субконто на счете плана счетов. Номер вида субконто начинается с *1*. Набор и порядок видов субконто соответствуют набору и порядку видов субконто у счета дебета;
- *СубконтоКт<Номер субконто>* – имеет тип *Характеристика.<имя>*, тип этого поля определяется типом субконто. Содержит значение субконто кредита. Количество таких полей зависит от максимального количества субконто на счете плана счетов. Номер вида субконто начинается с *1*. Набор и порядок видов субконто соответствуют набору и порядку видов субконто у счета кредита;
- *СчетДт* – имеет тип *ПланСчетовСсылка.<Имя>*. Содержит ссылку на дебетуемый счет;
- *СчетКт* – имеет тип *ПланСчетовСсылка.<Имя>*. Содержит ссылку на кредитуемый счет.

При построении этой виртуальной таблицы могут использоваться параметры, с помощью которых настраивается или уточняется состав получаемых данных. Параметры следует задавать строго в порядке их описания:

- *НачалоПериода*, *КонецПериода* – имеет тип *Дата*, *МоментВремени* или *Граница*.

Период времени, за который будут отобраны проводки;

- *Условие* – содержит конструкцию языка запросов. Конструкция позволяет установить условие на любое поле виртуальной таблицы. Однако в условии нельзя использовать сравнение на неравно с полями *Субконто*, *СубконтоДт*, *СубконтоКт*;
- *Порядок* – содержит конструкцию языка запросов. Конструкция позволяет задать упорядочивание проводок;
- *Первые* – имеет тип *Число*. С его помощью задается ограничение максимального количества записей.

Рассмотрим пример использования таблицы движений с субконто для получения перечня всех проводок регистра бухгалтерии по счетам, подчиненным определенному счету. Это можно сделать с помощью следующего запроса (листинг 3.76).

Листинг 3.76. Получение всех проводок по счетам, подчиненным счету «Касса»

```
ВЫБРАТЬ
  ДвиженияССубконто.Период КАК Период,
  ДвиженияССубконто.СчетДт,
  ДвиженияССубконто.СчетКт,
  ДвиженияССубконто.Сумма,
  ДвиженияССубконто.Содержание
ИЗ
  РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ДвиженияССубконто( , , Счет = В ИЕРАРХИИ
  (&СчетКассы), , ) КАК ДвиженияССубконто

УПОРЯДОЧИТЬ ПО
  Период
```

В результате выполнения данного запроса мы получим все проводки регистра бухгалтерии по счетам, подчиненным счету *Касса* (рис. 3.79).

Запрос: РегистрБухгалтерии.ОсновнойРегистрБухгалтерии.ДвиженияССубконто((Записей в результате: 17)

Период	СчетДт	СчетКт	Сумма	Содержание
30.04.2013 12:00:00	Касса	Капитал	2 850	
30.04.2013 12:00:00	Касса	Капитал	3 500	
30.04.2013 12:00:00	Касса	Капитал	100	
01.05.2013 12:00:00	Касса	Покупатели	100	
01.05.2013 12:00:00	Касса	Покупатели	150	
01.05.2013 12:00:00	Касса	Покупатели	200	
01.05.2013 12:00:00	Покупатели	Касса	80	
01.05.2013 12:00:02	Капитал	Касса	6 810	
31.05.2013 12:00:00	Контрагенты	Касса	3	
31.05.2013 12:00:00	Касса	Контрагенты	8	
01.06.2013 12:00:00	Касса	Покупатели	10	
01.06.2013 23:59:59	Касса	Покупатели	10	
02.06.2013 0:00:00	Касса	Покупатели	10	
02.06.2013 12:00:01	Касса	Покупатели	10	
20.06.2013 12:00:00	Контрагенты	Касса	10	
20.06.2013 12:00:00	Касса	Контрагенты	10	
30.06.2013 12:00:00	Касса	Капитал	3	

Рис. 3.79. Получение всех проводок по счетам, подчиненным счету «Касса»

Следует иметь в виду, что список полей таблицы, которые можно использовать для группировки данных, и полей, на которые можно ставить условие в параметре

виртуальной таблицы, не совпадает. Перечень полей, на которые можно устанавливать условия, существенно шире. Если для включения в отчет нам доступны только физически существующие поля, полученные виртуальной таблицей из реальных таблиц первичных движений, то для выполнения отборов доступны также и виртуальные поля, такие как *Счет* (без указания стороны проводки), *Субконто*, *ВидСубконто*, *КорСчет*, *<Ресурс>Кор*.

Сложные периодические расчеты

Для решения задач сложных периодических расчетов (например, расчет заработной платы, расчет стоимости долгосрочной аренды и др.) в системе «1С:Предприятие» используются такие объекты конфигурации, как планы видов расчета и регистры расчета. В этом разделе мы рассмотрим примеры использования этих объектов для решения различных прикладных задач с помощью языка запросов.

Все примеры, используемые в данном разделе, можно посмотреть в демонстрационной конфигурации «Сложные периодические расчеты», которая находится на прилагаемом компакт-диске.

Планы видов расчета

Вид расчета является одним из основных понятий механизма периодических расчетов. Например, для целей расчета зарплаты различными видами расчета могут быть оклад, надбавка за вредность, оплата сверхурочных. Для хранения видов расчета и описания их взаимосвязи используется объект конфигурации *План видов расчета*.

Данные каждого плана видов расчета хранятся в отдельной таблице базы данных, доступной посредством запросов. Эта таблица имеет следующий состав полей:

- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный план видов расчета;
- *<Имя реквизита>* – поле, содержащее значения реквизита вида расчета с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для плана видов расчета как объекта конфигурации;
- *<Имя табличной части>* – поле, содержащее табличные части вида расчета. Имена полей соответствуют именам табличных частей плана видов расчета, как они заданы в конфигураторе. Имеет тип *РезультатЗапроса*. Результат запроса к табличной части состоит из колонки *НомерСтроки* и колонок с именами, соответствующими именам реквизитов табличной части;
- *БазовыеВидыРасчета* – предопределенная табличная часть плана видов расчета. Содержит вложенную таблицу базовых видов расчета. Имеет тип *РезультатЗапроса*. Результат запроса к табличной части состоит из колонок

НомерСтроки, *ВидРасчета* и *Предопределенный*. Существует только для планов видов расчета, у которых задано свойство *Зависимость от базы*;

- *ВедущиеВидыРасчета* – предопределенная табличная часть плана видов расчета. Содержит вложенную таблицу ведущих видов расчета. Имеет тип *РезультатЗапроса*. Результат запроса к табличной части состоит из колонок *НомерСтроки*, *ВидРасчета* и *Предопределенный*;
- *ВытесняющиеВидыРасчета* – предопределенная табличная часть плана видов расчета. Содержит вложенную таблицу вытесняющих видов расчета. Имеет тип *РезультатЗапроса*. Результат запроса к табличной части состоит из колонок *НомерСтроки*, *ВидРасчета* и *Предопределенный*. Существует только для планов видов расчета, у которых задано свойство *Использует период действия*;
- *Код* – имеет тип *Строка* или *Число*. Содержит код вида расчета;
- *Наименование* – имеет тип *Строка*. Содержит наименование вида расчета;
- *ПериодДействияБазовый* – имеет тип *Булево*. Содержит признак того, что для учетных записей с этим видом расчета период действия является базовым периодом. Существует только для планов видов расчета, у которых заданы свойства *Зависимость от базы* и *Использует период действия*;
- *ПометкаНаУдаление* – имеет тип *Булево*. Содержит признак пометки на удаление вида расчета;
- *Предопределенный* – имеет тип *Булево*. Содержит признак того, что данный вид расчета определен в метаданных и над ним нельзя производить некоторые операции;
- *Представление* – виртуальное поле, не хранится в информационной базе. Содержит представление вида расчета;
- *Ссылка* – содержит ссылку на вид расчета;

Рассмотрим некоторые примеры получения данных из планов видов расчетов.

Для хранения списка видов расчетов в нашей демонстрационной конфигурации используются планы видов расчетов *Основные начисления* и *Дополнительные начисления* (рис. 3.80).

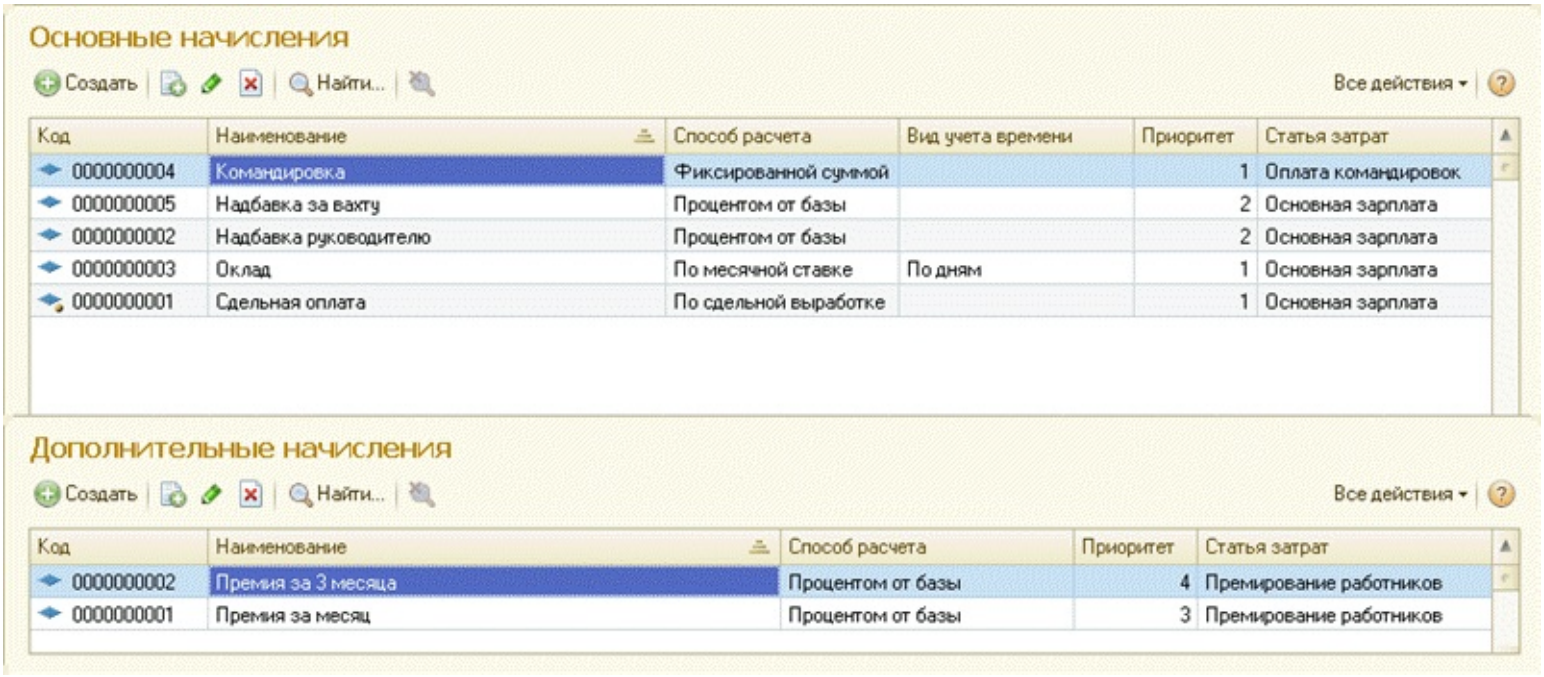


Рис. 3.80. Планы видов расчета для основных и дополнительных начислений при расчете заработной платы

Предположим, нам нужно отобрать те виды расчетов из плана видов расчета *Основные начисления*, которые имеют наивысший приоритет, т. е. записи с этим видом расчета рассчитываются в первую очередь. Для этого нужно просто выполнить отбор видов расчета по значению реквизита *Приоритет* (листинг 3.77).

Листинг 3.77. Отбор видов расчета по значению реквизита «СтатьяЗатрат»

```
ВЫБРАТЬ
  ОсновныеНачисления.Наименование
ИЗ
  ПланВидовРасчета.ОсновныеНачисления КАК ОсновныеНачисления
ГДЕ
  ОсновныеНачисления.Приоритет = 1
```

Результат выполнения запроса представлен на рис. 3.81.

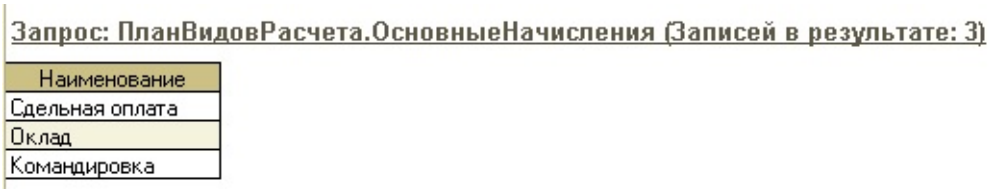


Рис. 3.81. Отбор видов расчета по значению реквизита «СтатьяЗатрат»

У всех планов видов расчета независимо от настройки существует предопределенная табличная часть *ВедущиеВидыРасчета*. Эта табличная часть позволяет для каждого вида расчета указать список ведущих расчетов, описанных в любом из планов видов расчета конфигурации, независимо от их настройки.

У планов видов расчета, использующих период действия, существует также предопределенная табличная часть *ВытесняющиеВидыРасчета*. В ней для каждого вида расчета задается список вытесняющих видов расчета, которые могут храниться только в данном плане видов расчета.

Если у плана видов расчета установлена зависимость от базы, в его структуре будет присутствовать еще одна предопределенная табличная часть – *БазовыеВидыРасчета*. В ней для каждого вида расчета можно указать список базовых видов расчета, которые могут быть описаны в тех планах видов расчета, которые отмечены в качестве базовых для данного плана видов расчета.

Например, для вида расчета *Премия за месяц* из плана видов расчета *Дополнительные начисления* в список базовых видов расчета входят виды расчета, описанные в плане видов расчета *Основные начисления* (рис. 3.82).

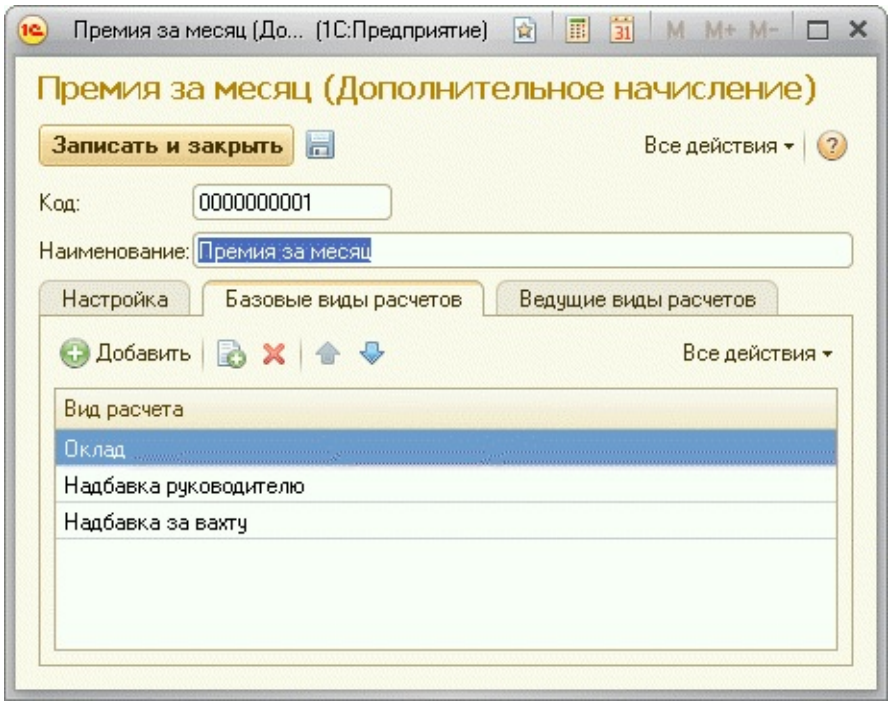


Рис. 3.82. Предопределенные табличные части плана видов расчета «Дополнительные начисления»

Чтобы получить записи из предопределенной табличной части вида расчета, можно обратиться к имени табличной части через точку от имени основной таблицы плана видов расчета и наложить отбор на конкретный вид расчета по полю *Ссылка* (листинг 3.78).

Листинг 3.78. Получение списка базовых видов расчета для заданного вида расчета

```
ВЫБРАТЬ
  *
ИЗ
  ПланВидовРасчета.ДополнительныеНачисления.БазовыеВидыРасчета КАК
  ДополнительныеНачисления
ГДЕ
  ДополнительныеНачисления.Ссылка = &ВидРасчета
```

В результате для вида расчета *Премия за месяц* из плана видов расчета *Дополнительные начисления* мы получим следующий список базовых видов расчета (рис. 3.83).

Запрос: ПланВидовРасчета.ДополнительныеНачисления.БазовыеВидыРасчета (Записей в результате: 3)

Ссылка	НомерСтроки	ВидРасчета	Предопределенный
Премия за месяц	1	Оклад	Нет
Премия за месяц	2	Надбавка руководителю	Нет
Премия за месяц	3	Надбавка за вахту	Нет

Рис. 3.83. Получение списка базовых видов расчета для заданного вида расчета

В поле *Ссылка* хранится ссылка на вид расчета (строку основной таблицы), которому принадлежит данная строка табличной части. В поле *ВидРасчета* хранится ссылка на вид расчета, являющийся базовым видом расчета для данного вида расчета (на который указывает поле *Ссылка*). Поле *Предопределенный* хранит информацию о том, является ли данная строка табличной части заданной предопределенно в конфигураторе.

Усложним задачу. Предположим, нам нужно узнать, для каких видов расчета заданный вид расчета является ведущим, базовым или вытесняющим. Это можно сделать с помощью следующего запроса (листинг 3.79).

Листинг 3.79. Получение списка видов расчета, на которые влияет заданный вид расчета

```
ВЫБРАТЬ
  "Основные Ведущие Расчеты" КАК ВидыРасчетов,
  ОсновныеНачисленияВедущиеРасчеты.Ссылка
ИЗ
  ПланВидовРасчета.ОсновныеНачисления.ВедущиеВидыРасчета КАК
  ОсновныеНачисленияВедущиеРасчеты
ГДЕ
  ОсновныеНачисленияВедущиеРасчеты.ВидРасчета = &ВидРасчета

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  "Основные Базовые Расчеты" КАК ВидыРасчетов,
  ОсновныеНачисленияБазовыеРасчеты.Ссылка
ИЗ
  ПланВидовРасчета.ОсновныеНачисления.БазовыеВидыРасчета КАК
  ОсновныеНачисленияБазовыеРасчеты
ГДЕ
  ОсновныеНачисленияБазовыеРасчеты.ВидРасчета = &ВидРасчета

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  "Основные Вытесняющие Расчеты" КАК ВидыРасчетов,
  ОсновныеНачисленияВытесняющиеРасчеты.Ссылка
ИЗ
  ПланВидовРасчета.ОсновныеНачисления.ВытесняющиеВидыРасчета КАК
  ОсновныеНачисленияВытесняющиеРасчеты
ГДЕ
  ОсновныеНачисленияВытесняющиеРасчеты.ВидРасчета = &ВидРасчета

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  "Дополнительные Ведущие Расчеты" КАК ВидыРасчетов,
  ДополнительныеНачисленияВедущиеРасчеты.Ссылка
ИЗ
```

```
  ПланВидовРасчета.ДополнительныеНачисления.ВедущиеВидыРасчета КАК
  ДополнительныеНачисленияВедущиеРасчеты
ГДЕ
  ДополнительныеНачисленияВедущиеРасчеты.ВидРасчета = &ВидРасчета

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  "Дополнительные Базовые Расчеты" КАК ВидыРасчетов,
  ДополнительныеНачисленияБазовыеРасчеты.Ссылка
ИЗ
  ПланВидовРасчета.ДополнительныеНачисления.БазовыеВидыРасчета КАК
  ДополнительныеНачисленияБазовыеРасчеты
ГДЕ
  ДополнительныеНачисленияБазовыеРасчеты.ВидРасчета = &ВидРасчета

ИТОГИ ПО
  ВидыРасчетов
```

В запросе мы объединяем информацию из всех предопределенных табличных частей для всех планов видов расчета в демонстрационной конфигурации. При выборе информации из каждой табличной части накладывается условие на значение предопределенного реквизита *ВидРасчета*. Таким образом, мы получаем записи из тех табличных частей видов расчета, где встречается заданный вид расчета. В выборку результата запроса включается стандартное поле табличной части *Ссылка*, в котором хранится ссылка на тот вид расчета, к которому относится табличная часть.

Таким образом, для вида расчета *Оклад* мы получим следующий результат (рис. 3.84).

Запрос: ПланВидовРасчета.ОсновныеНачисления.ВедущиеВидыРасчета (Записей в результате: 9)

ВидыРасчетов	Ссылка
Основные Ведущие Расчеты	
Основные Ведущие Расчеты	Надбавка руководителю
Основные Ведущие Расчеты	Надбавка за вахту
Основные Базовые Расчеты	
Основные Базовые Расчеты	Надбавка руководителю
Основные Базовые Расчеты	Надбавка за вахту
Дополнительные Базовые Расчеты	
Дополнительные Базовые Расчеты	Премия за месяц
Дополнительные Базовые Расчеты	Премия за 3 месяца

Рис. 3.84. Получение списка видов расчетов, на которые влияет вид расчета «Оклад»

Мы видим, что вид расчета *Оклад* из плана видов расчета *Основные начисления* является ведущим и базовым расчетом для видов расчета *Надбавка руководителю* и *Надбавка за вахту* из плана видов расчета *Основные начисления*, а также является базовым расчетом для видов расчета *Премия за месяц* и *Премия за 3 месяца* из плана видов расчета *Дополнительные начисления*.

Регистры расчета

Регистры расчета – это прикладные объекты конфигурации, предназначенные для периодической регистрации данных о произведенных расчетах. В зависимости от свойств регистра, таких как учет протяженных во времени расчетов, использование зависимости

по базовому периоду и др., реализуются расчетные механизмы по выполнению периодических расчетов.

Основная таблица регистра расчета в информационной базе содержит следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения регистра с именем, заданным в конфигурации. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации;
- *<Имя общего реквизита>* – поле, содержащее значения общего реквизита с именем, заданным в конфигурации. Такие поля создаются для общих реквизитов, не являющихся разделителями, или для разделителей с режимом использования разделяемых данных *НезависимоИСовместно*, в которых участвует данный регистр;
- *<Имя реквизита>* – поле, содержащее значения реквизита регистра с именем, заданным в конфигурации. Количество таких полей равно количеству реквизитов, определенных для регистра как объекта конфигурации;
- *<Имя ресурса>* – поле, содержащее значения ресурса регистра с именем, заданным в конфигурации. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации;
- *Активность* – имеет тип *Булево*. Содержит признак активности записи (участие записи в конкуренции за период действия, влияние на получение базы);
- *БазовыйПериодНачало* – имеет тип *Дата*. Содержит дату начала интервала базового периода учетной записи. Существует только для регистров, у которых установлено свойство *Базовый период*;
- *БазовыйПериодКонец* – имеет тип *Дата*. Содержит дату окончания интервала базового периода учетной записи. Существует только для регистров, у которых установлено свойство *Базовый период*;
- *ВидРасчета* – имеет тип *ПланВидовРасчетаСсылка.<имя>*. Содержит ссылку на вид расчета учетной записи;
- *НомерСтроки* – имеет тип *Число*. Содержит уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*. Таким образом, совокупность значений *Регистратор* и *НомерСтроки* позволяют идентифицировать конкретную запись регистра расчета;
- *ПериодДействияНачало* – имеет тип *Дата*. Содержит дату начала интервала периода действия учетной записи. Существует только для регистров, у которых установлено свойство *Период действия*;
- *ПериодДействияКонец* – имеет тип *Дата*. Содержит дату окончания интервала периода действия учетной записи. Существует только для регистров, у которых установлено свойство *Период действия*;
- *ПериодДействия* – имеет тип *Дата*. Отражает период регистра расчета, в котором действовала запись. Эта дата всегда имеет значение начала первого дня соответствующего периода (*ПериодДействияНачало*). Размер периода определяется периодичностью регистра расчета. Существует только для регистров,

у которых установлено свойство *Период действия*;
- *Период регистрации* – имеет тип *Дата*. Отражает период регистра расчета, к которому относится данная запись. Размер периода определяется периодичностью регистра расчета.
- *Регистратор* – имеет тип *ДокументСсылка.<имя>*. Содержит ссылку на документ, который ввел запись в регистр;
- *Сторно* – имеет тип *Булево*. Содержит признак того, является ли данная запись сторно-записью. Сторно-записи записываются в регистр со значением *Истина* в этом поле.

В нашей демонстрационной конфигурации существует регистр расчета *Основные начисления регламентированные*, который хранит данные с периодичностью *Месяц* и использует план видов расчета *Основные начисления*. Данный регистр позволяет учитывать расчеты, протяженные во времени, – для работы с такими расчетами регистр расчета использует регистр сведений *Графики работы*. Кроме того, регистр поддерживает механизм зависимости по базовому периоду.

Регистр расчета имеет измерения *ФизЛицо* и *Организация*, ресурс *Результат*, а также реквизиты *Размер*, *ГрафикРаботы*, *ВидУчетаВремени*, *Подразделение* и *СтатьяЗатрат* для хранения дополнительной информации о расчете. Движения в регистре формируются при проведении документа *Начисление зарплаты*, который является регистратором регистра расчета.

Также в демонстрационной конфигурации существует регистр расчета *Дополнительные начисления регламентированные*, который хранит данные с периодичностью *Месяц* и использует план видов расчета *Дополнительные начисления*. Данный регистр поддерживает механизм зависимости по базовому периоду.

Регистр расчета имеет измерения *ФизЛицо* и *Организация*, ресурс *Результат*, а также реквизиты *Размер*, *Подразделение* и *СтатьяЗатрат* для хранения дополнительной информации о расчете. Движения в регистре формируются при проведении документа *Начисление зарплаты*, который является регистратором регистра расчета.

На примере этих регистров рассмотрим типичные задачи по получению данных из регистров расчета.

Получение данных из регистра расчета

В рамках данной книги мы не будем рассматривать технологию формирования и расчета записей регистров расчета. Эти вопросы подробно рассматриваются в документации.

Будем исходить из того, что в ресурсе регистра расчета *Результат* хранится уже посчитанное значение для различных видов расчета, используемых при начислении зарплаты сотрудникам, в соответствии с алгоритмами их расчета. Наша задача – получить это значение из регистра в разрезах, интересующих пользователя.

Пример 1

Предположим, необходимо получить суммарные основные начисления по каждому сотруднику за каждый месяц (период действия), в котором производился расчет, с разбивкой по видам расчета. Это можно сделать с помощью следующего запроса (листинг 3.80).

Листинг 3.80. Получение суммарных основных начислений сотрудникам по месяцам

```
ВЫБРАТЬ
  Начисления.ПериодДействия КАК ПериодДействия,
  Начисления.ФизЛицо КАК Сотрудник,
  Начисления.ВидРасчета КАК ВидРасчета,
  Начисления.Результат КАК Результат
ИЗ
  РегистрРасчета.ОсновныеНачисленияРегл КАК Начисления

УПОРЯДОЧИТЬ ПО
  ПериодДействия

ИТОГИ
  СУММА(Результат)
ПО
  ОБЩИЕ,
  ПериодДействия,
  Сотрудник
```

Поскольку регистр расчета *ОсновныеНачисленияРегл* имеет периодичность *Месяц*, то в поле *ПериодДействия* будут храниться даты начала каждого месяца, рассчитанные от даты начала периода действия той записи, по которой был произведен расчет. В запросе рассчитываются общие итоги, а также итоги по полям *ПериодДействия* и *Сотрудник*.

В результате мы видим сумму основных начислений по каждому сотруднику за каждый месяц, а также видим, из каких начислений с разбивкой по видам расчета складывалась общая сумма начислений каждого сотрудника (рис. 3.85).

Запрос: РегистрРасчета.ОсновныеНачисленияРегл (Записей в результате: 17)

ПериодДействия	Сотрудник	ВидРасчета	Результат
			24 600
01.02.2013 0:00:00			3 000
01.02.2013 0:00:00	Иванов		3 000
01.02.2013 0:00:00	Иванов	Оклад	3 000
01.03.2013 0:00:00			21 600
01.03.2013 0:00:00	Иванов		2 815
01.03.2013 0:00:00	Иванов	Оклад	3 000
01.03.2013 0:00:00	Иванов	Надбавка руководителю	300
01.03.2013 0:00:00	Иванов	Надбавка за вахту	29
01.03.2013 0:00:00	Иванов	Командировка	200
01.03.2013 0:00:00	Иванов	Оклад	-714
01.03.2013 0:00:00	Петров		11 100
01.03.2013 0:00:00	Петров	Сдельная оплата	11 100
01.03.2013 0:00:00	Кузнецова		7 685
01.03.2013 0:00:00	Кузнецова	Оклад	4 714
01.03.2013 0:00:00	Кузнецова	Командировка	2 500
01.03.2013 0:00:00	Кузнецова	Надбавка руководителю	471

Рис. 3.85. Получение суммарных основных начислений сотрудникам по месяцам

Пример 2

Рассмотрим следующую задачу. Пусть нам необходимо узнать сумму основных начислений, начисленных за март 2013 года каждому сотруднику в каждом подразделении в разрезе статей затрат. Это можно сделать с помощью следующего запроса (листинг 3.81).

Листинг 3.81. Получение суммарных основных начислений сотрудникам в разрезе статей затрат и подразделений

```
ВЫБРАТЬ
  Начисления.СтатьяЗатрат КАК СтатьяЗатрат,
  Начисления.Подразделение КАК Подразделение,
  Начисления.ФизЛицо КАК Сотрудник,
  Начисления.Результат КАК Результат
ИЗ
  РегистрРасчета.ОсновныеНачисленияРегл КАК Начисления
ГДЕ
  Месяц(Начисления.ПериодДействия)= 3
  И Год(Начисления.ПериодДействия)= 2013

ИТОГИ
  СУММА(Результат)
ПО
  ОБЩИЕ,
  СтатьяЗатрат,
  Подразделение,
  Сотрудник
```

Результат выполнения запроса представлен на рис. 3.86.

Запрос: РегистрРасчета.ОсновныеНачисленияРегл (Записей в результате: 22)

СтатьяЗатрат	Подразделение	Сотрудник	Результат
			21 600
Основная зарплата			18 900
Основная зарплата	Производство		11 100
Основная зарплата	Производство	Петров	11 100
Основная зарплата	Бухгалтерия		2 615
Основная зарплата	Бухгалтерия	Иванов	2 615
Основная зарплата	Администрация		5 185
Основная зарплата	Администрация	Кузнецова	5 185
Оплата командировок			2 700
Оплата командировок	Бухгалтерия		200
Оплата командировок	Бухгалтерия	Иванов	200
Оплата командировок	Администрация		2 500
Оплата командировок	Администрация	Кузнецова	2 500

Рис. 3.86. Получение суммарных основных начислений сотрудникам в разрезе статей затрат и подразделений

Получение данных о фактическом периоде действия записи для расчета

Для корректного расчета каждой записи в регистре расчета, прежде всего, необходимо правильно определить фактический период действия этой записи. Дело в том, что при вводе в регистр расчета записей с вытесняющими видами расчета (например, командировок, больничных) фактический период действия записей с вытесняемыми видами расчета (например, оклада) меняется.

При этом данные о фактическом периоде действия не хранятся в основной таблице

регистра расчета. Для получения фактического периода действия используется виртуальная таблица фактического периода действия *ФактическийПериодДействия()*. Эта таблица формируется платформой только для регистров расчета, использующих период действия.

Структура полей этой таблицы полностью идентична структуре основной таблицы регистра расчета. Единственное ее отличие от основной таблицы состоит в том, что поля *ПериодДействияНачало* и *ПериодДействияКонец* обозначают не собственно период действия, а интервал фактического периода действия. В случае, если фактический период действия записи задается несколькими интервалами, в виртуальной таблице такая запись будет представлена несколькими строками. Например, если в регистр расчета введены следующие записи (табл. 3.8), то в таблице фактического периода действия для этих записей будут присутствовать следующие строки (табл. 3.9).

Таблица 3.8. Основная таблица регистра расчета

ПериодРегистрации	ВидРасчета	ПериодДействия	ПериодДействияНачало	Период
01.02.2013	Оклад	01.02.2013	01.02.2013	28.02.2
01.02.2013	Надбавка	01.02.2013	01.02.2013	28.02.2
01.02.2013	Больничный	01.02.2013	14.02.2013	17.02.2

Таблица 3.9. Виртуальная таблица «ФактическийПериодДействия»

ПериодРегистрации	ВидРасчета	ПериодДействия	ПериодДействияНачало	Период
01.02.2013	Оклад	01.02.2013	01.02.2013	13.02.2
01.02.2013	Оклад	01.02.2013	18.02.2013	28.02.2
01.02.2013	Надбавка	01.02.2013	01.02.2013	28.02.2
01.02.2013	Больничный	01.02.2013	14.02.2013	17.02.2

Так как вид расчета *Больничный* вытесняет по фактическому периоду действия *Оклад*, то в виртуальной таблице фактического периода действия запись об окладе представлена двумя строками, исключающими временной интервал, в котором действует больничный. Запись о надбавке одинакова в обеих таблицах, т. к. надбавка не вытесняется больничным.

Рассмотрим пример получения данных из таблицы фактического периода действия. В следующем запросе подсчитывается количество календарных дней фактического периода действия каждой записи документа-регистратора (листинг 3.82).

Листинг 3.82. Получение фактического периода действия для записей документа

```
&НаСервереБезКонтекста
Процедура ПолучитьФПДЗапросом(Ссылка);
```

```
  Запрос = Новый Запрос;
  Запрос.Текст = "
  |ВЫБРАТЬ
  |        ФактическийПериод.НомерСтроки,
  |        СУММА(РАЗНОСТЬДАТ(ФактическийПериод.ПериодДействияНачало,
  КОНЕЦПЕРИОДА(ФактическийПериод.ПериодДействияКонец, ДЕНЬ), ДЕНЬ)) + 1 КАК КоличествоДней
  |ИЗ
  |        РегистрРасчета.ОсновныеНачисленияРегл.ФактическийПериодДействия(Регистратор =
  &Регистратор) КАК ФактическийПериод
  |
  |СГРУППИРОВАТЬ ПО
  |        ФактическийПериод.НомерСтроки";

  Запрос.УстановитьПараметр("Регистратор", Ссылка);
  Выборка = Запрос.Выполнить().Выбрать();

  Пока Выборка.Следующий() Цикл
    Сообщение = Новый СообщениеПользователю();
    Сообщение.Текст = "Количество дней фактического периода для строки № " +
      Выборка.НомерСтроки + " : " + Выборка.КоличествоДней;
    Сообщение.Сообщить();
КонецЦикла;

КонецПроцедуры
```

У виртуальной таблицы фактического периода действия существует единственный параметр *Условие*. В этот параметр мы передаем условие отбора по регистратору (*Регистратор = &Регистратор*), движения которого нас интересуют.

В результате мы видим, что для сотрудника *Кузнецова* записи о командировке вытеснили записи об окладе. Поэтому количество календарных дней фактического периода действия записи об окладе *17 = 31 - 14* (рис. 3.87).

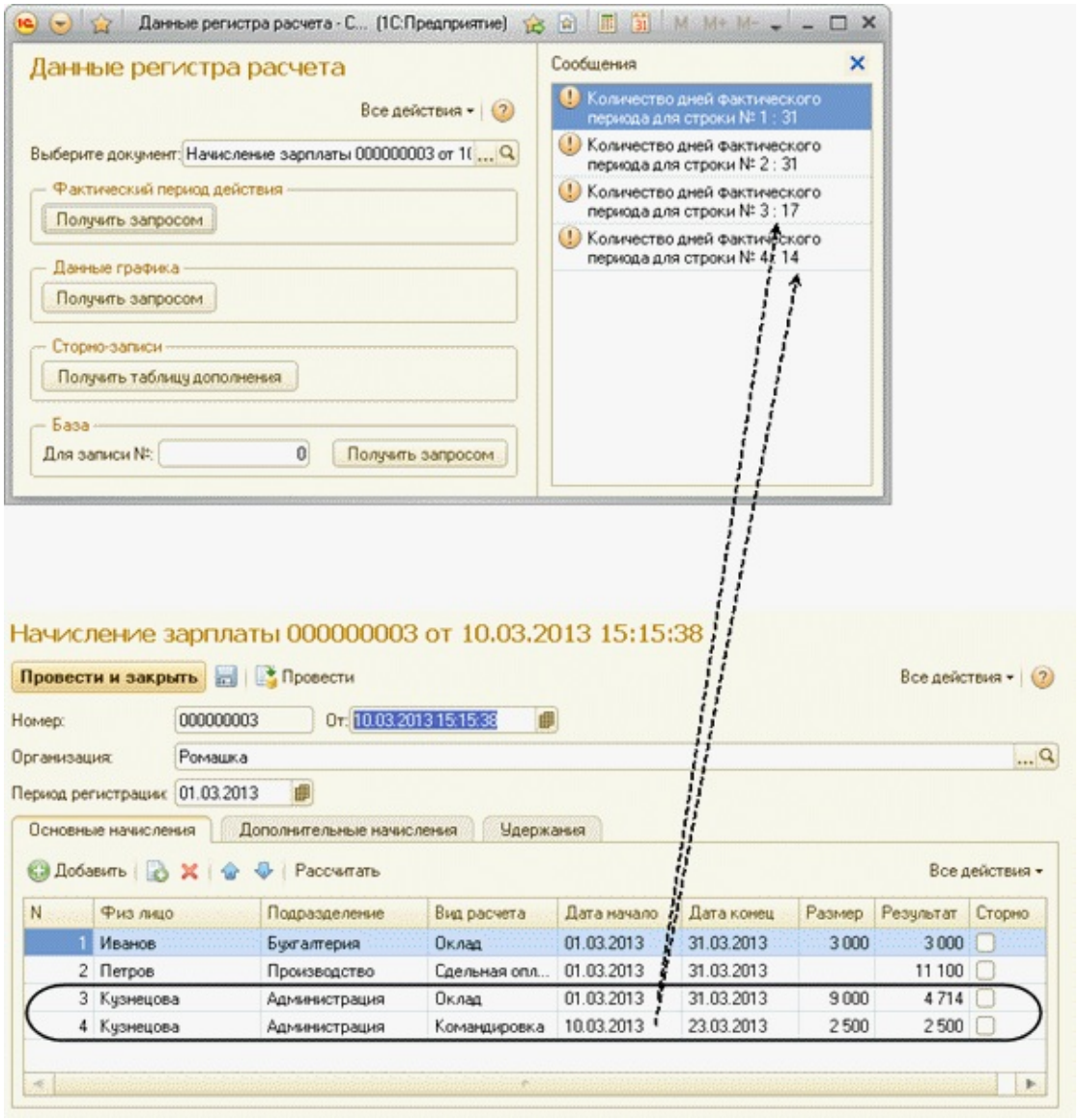


Рис. 3.87. Получение фактического периода действия для записей документа

Этот фрагмент можно посмотреть в обработке *ДанныеРегистраРасчета* в демонстрационной конфигурации «Сложные периодические расчеты», которая находится на прилагаемом компакт-диске.

Получение данных графика для расчета записи

Для расчета протяженных во времени расчетов необходимо получить данные графика, привязанного к регистру расчета. Например, при расчете оклада сотрудника за месяц нужно знать количество рабочих дней в месяце всего и количество фактически отработанного этим сотрудником времени. При этом сам график работы хранится в регистре сведений, в котором рабочие периоды имеют значение 1, а выходные – 0.

Для получения данных графика для записей регистра расчета используется виртуальная таблица *ДанныеГрафика()*. Эта таблица формируется на основе регистра расчета в соответствии с условием отбора, переданным в параметре виртуальной таблицы *Условие*.

Количество строк в таблице данных графика равно количеству записей регистра расчета, удовлетворяющих условию, заданному в параметре виртуальной таблицы. Часть полей

виртуальной таблицы данных графика полностью совпадает с полями основной таблицы регистра. Кроме того, на каждый ресурс регистра сведений, назначенного в качестве графика регистра расчета, в таблице данных графика будет добавлено 4 поля, в которых будут просуммированы значения графика по данному ресурсу за каждый из возможных периодов (период действия, фактический период действия, базовый период и период регистрации):

- *<Имя ресурса графика>ПериодДействия* – имеет тип *Число*. Содержит сумму соответствующего ресурса регистра сведений, назначенного в качестве графика регистра расчета. Суммирование проводится по всему периоду действия записи регистра расчета (месяц, квартал, год и т. д.), заданному в реквизите *ПериодДействия*;
- *<Имя ресурса графика>ФактическийПериодДействия* – имеет тип *Число*. Содержит сумму соответствующего ресурса регистра сведений за фактический период действия записи регистра расчета;
- *<Имя ресурса графика>БазовыйПериод* – имеет тип *Число*. Содержит сумму соответствующего ресурса регистра сведений за весь базовый период записи регистра расчета, т. е. за период с *БазовыйПериодНачало* по *БазовыйПериодКонец*;
- *<Имя ресурса графика>ПериодРегистрации* – имеет тип *Число*. Содержит сумму соответствующего ресурса регистра сведений за весь период регистрации записи регистра расчета (месяц, квартал, год и т. д.), заданный в реквизите *ПериодРегистрации*.

При построении виртуальной таблицы данных графика происходит соединение таблиц регистра расчета и регистра сведений. Кроме того, в случае получения поля *<Имя ресурса графика>ФактическийПериодДействия* происходит также соединение с таблицей фактического периода действия регистра расчета. Так как данные для всех четырех перечисленных полей получаются путем соединения с таблицей регистра сведений по разным условиям, то в общем случае будет выполнено четыре соединения с таблицей регистра сведений. Поэтому без необходимости не следует получать поля виртуальной таблицы данных графика «на всякий случай», т. к. это может существенно понизить производительность запроса.

Рассмотрим пример получения данных из таблицы данных графика. В следующем запросе получаются данные графика за фактический период действия и за весь период действия в целом для всех записей документа-регистратора (листинг 3.83).

Листинг 3.83. Получение данных графика для записей документа

```
ВЫБРАТЬ
  ДанныеГрафика.НомерСтроки,
  ДанныеГрафика.ЗначениеФактическийПериодДействия КАК Факт,
  ДанныеГрафика.ЗначениеПериодДействия КАК Норма
ИЗ
  РегистрРасчета.ОсновныеНачисленияРегл.ДанныеГрафика(Регистратор = &Регистратор) КАК
  ДанныеГрафика
```

В результате для того же документа о начислениях, что и в предыдущем примере (см. рис. 3.87), мы видим, что норма рабочих дней за период действия (март 2013) у всех сотрудников одинакова, а значение фактически отработанного времени зависит от фактического периода действия записи регистра расчета (рис. 3.88).

Запрос: РегистрРасчета.ОсновныеНачисленияРегл.ДанныеГрафика((Записей в результате: 4)

НомерСтроки	Факт	Норма
1	21	21
2	21	21
3	11	21
4	10	21

Рис. 3.88. Получение данных графика для записей документа

Получение базы для расчета записей

Для расчета записей, зависимых по базовому периоду от других записей регистров расчета, необходимо получить расчетную базу этих записей. Например, при расчете надбавки сотруднику в определенной организации за определенный период нужно получить значение оклада этого сотрудника в данной организации за этот период, а при расчете премии сотрудника нужно учитывать его оклад и начисленные ему надбавки.

Для получения базы расчетов при помощи запроса используются виртуальные таблицы базовых данных. Эти таблицы формируются только для регистров расчета, поддерживающих базовый период. У каждого регистра таких таблиц может быть несколько.

Например, план видов расчета *ДополнительныеНачисления* зависит по базе от планов видов расчета *ОсновныеНачисления* и *ДополнительныеНачисления*. Эти планы видов расчета, в свою очередь, используются в регистрах регламентированного и управленческого учета *ОсновныеНачисленияРегл*, *ОсновныеНачисленияУпр* и *ДополнительныеНачисленияРегл* и *ДополнительныеНачисленияУпр*.

Это означает, что любые записи регистра *ДополнительныеНачисленияРегл* могут зависеть по базовому периоду от записей любого из четырех регистров. В этом случае у регистра *ДополнительныеНачисленияРегл* будет 4 виртуальные таблицы базовых данных.

- *ДополнительныеНачисленияРегл.БазаДополнительныеНачисленияРегл*,
- *ДополнительныеНачисленияРегл.БазаДополнительныеНачисленияУпр*,
- *ДополнительныеНачисленияРегл.БазаОсновныеНачисленияРегл*,
- *ДополнительныеНачисленияРегл.БазаОсновныеНачисленияУпр*.

Имя виртуальной таблицы формируется по следующей схеме:

<ИмяОсновногоРегистра>.База<ИмяБазовогоРегистра>

Таким образом, например, таблица

ДополнительныеНачисленияРегл.БазаОсновныеНачисленияРегл позволяет получить записи регистра *ОсновныеНачисленияРегл*, входящие в базу расчета записей регистра *ДополнительныеНачисленияРегл*.

При формировании виртуальных таблиц базовых данных требуется указать параметры, чтобы определить, какие именно данные, в каких разрезах нужно получить:

- *ИзмеренияОсновногоРегистра* – имеет тип *Массив* или *СписокЗначений*. В этот параметр нужно передать массив, элементами которого являются строки с именами измерений основного регистра, по которым нужно будет отбирать записи в базовых регистрах. Значения перечисленных в данном параметре измерений будут сопоставляться со значениями в соответствующих измерениях базового регистра расчета, заданных параметром *<ИзмеренияБазовогоРегистра>*;
- *ИзмеренияБазовогоРегистра* – имеет тип *Массив* или *СписокЗначений*. В этот параметр нужно передать массив, элементами которого являются строки с именами измерений базового регистра, по которым будет произведен отбор записей в базовом регистре. При этом в качестве значений отбора будут использованы значения соответствующих измерений основного регистра, заданных параметром *<ИзмеренияОсновногоРегистра>*. Число и порядок элементов этого массива должны совпадать с числом и порядком элементов массива измерений основного регистра;
- *Разрезы* – имеет тип *Массив* или *СписокЗначений*. В этот параметр передается массив, содержащий названия полей базового регистра, по которым необходимо получить разрез базы. По перечисленным в данном параметре полям будет выполняться дополнительная группировка суммируемых базовых данных. В запросе можно получить разрез не только по измерениям и реквизитам, но и по следующим предопределенным полям базового регистра: *НомерСтроки*, *Регистратор*, *ВидРасчета*, *ПериодРегистрации*, *ПериодДействия*;
- *Условие* – содержит конструкцию языка запросов. В этот параметр можно передать произвольное условие на записи основного регистра. База будет получена только для записей, удовлетворяющих этому условию.

Часть полей виртуальной таблицы базовых данных полностью совпадает с полями основной таблицы регистра расчета. Кроме того, будут добавлены поля для получения базы по всем ресурсам базового регистра, названия которых формируются по схеме – *<Имя ресурса базового регистра>База*.

Если параметр *Разрезы* не задан, то в таблице не будет других полей, и для каждой записи основного регистра, удовлетворяющей условию, будет рассчитана база общей суммой по каждому ресурсу базового регистра. Если разрезы заданы, то в виртуальной таблице будут присутствовать дополнительные поля по числу элементов массива разрезов. Названия этих полей в виртуальной таблице базовых данных будут формироваться по схеме – *<Имя разреза базового регистра>Разрез*.

Например, если в параметр *Разрезы* переданы значения *ПериодРегистрации* и

ВидРасчета, то в виртуальной таблице будут доступны поля *ПериодРегистрацииРазрез* и *ВидРасчетаРазрез*. Остальные поля с постфиксом *Разрез*, которые можно увидеть в конструкторе запроса, не будут доступны, и при выборе таких полей будет получена ошибка при выполнении запроса.

Рассмотрим пример получения базовых данных для записей регистра расчета *ДополнительныеНачисленияРегл* (основной регистр) из регистра расчета *ОсновныеНачисленияРегл* (базовый регистр). В следующем запросе получаются базовые данные для конкретной записи документа-регистратора в разрезе периодов регистрации и видов расчета (листинг 3.84).

Листинг 3.84. Получение базовых данных для записи документа

```
Запрос = Новый Запрос;
Запрос.Текст = "
  |ВЫБРАТЬ
  |        База.ПериодРегистрацииРазрез КАК ПериодРегистрацииРазрез,
  |        База.ВидРасчетаРазрез,
  |        СУММА(База.РезультатБаза) КАК РезультатБаза
  |ИЗ
  |
  РегистрРасчета.ДополнительныеНачисленияРегл.БазаОсновныеНачисленияРегл(&Измерения,
  &Измерения, &Разрезы, Регистратор = &Регистратор И НомерСтроки = &НомерСтроки) КАК База
  |СГРУППИРОВАТЬ ПО
  |        База.ПериодРегистрацииРазрез,
  |        База.ВидРасчетаРазрез
  |
  |УПОРЯДОЧИТЬ ПО
  |        ПериодРегистрацииРазрез
  |
  |ИТОГИ
  |        СУММА(РезультатБаза)
  |ПО
  |        ОБЩИЕ,
  |        ПериодРегистрацииРазрез";

// Сформировать массив измерений основного и базового регистров
// (названия измерений совпадают, поэтому используется один массив).
Измерения = Новый Массив(2);
Измерения[0] = "ФизЛицо";
Измерения[1] = "Организация";

// Сформировать массив разрезов.
Разрезы = Новый Массив(2);
Разрезы[0] = "ПериодРегистрации";
Разрезы[1] = "ВидРасчета";

// Передать параметры в запрос.
Запрос.УстановитьПараметр("Измерения", Измерения);
Запрос.УстановитьПараметр("Разрезы", Разрезы);

// Запрос строится по конкретной записи документа с номером ВыбранныйНомерСтроки
Запрос.УстановитьПараметр("Регистратор", СсылкаНаДокумент);
Запрос.УстановитьПараметр("НомерСтроки", ВыбранныйНомерСтроки);

Результат = Запрос.Выполнить();
```

...

В качестве измерений основного и базового регистра мы передаем массив *Измерения*, содержащий имена измерений этих регистров *ФизЛицо* и *Организация*. Так как состав измерений у основного и базового регистра идентичен, то в первые два параметра виртуальной таблицы мы передаем одно и то же значение. В результате при получении базы по основным начислениям, например, для премии, начисленной сотруднику *Иванов* в организации *Ромашка*, в виртуальную таблицу базовых данных попадут только базовые записи с аналогичными значениями измерений.

В качестве разрезов базового регистра мы передаем массив *Разрезы*, содержащий имена полей базового регистра *ПериодРегистрации* и *ВидРасчета*, по которым мы хотим получить разрез базовых данных. Благодаря этому мы можем использовать в запросе поля *ПериодРегистрацииРазрез* и *ВидРасчетаРазрез*.

В параметр *Условие* мы передаем условие отбора конкретной записи из определенного документа-регистратора.

В качестве базы выводится суммарное значение поля *РезультатБаза*. Записи в запросе группируются по разрезам базовых данных, а также в запросе рассчитываются общие итоги и итоги по периоду регистрации.

В результате для записи *№1* документа *Начисление зарплаты №5*, в котором содержится запись о начислении премии сотруднику *Иванов*, мы получим базу для расчета этой премии в разрезе видов расчета и периодов регистрации (рис. 3.89).

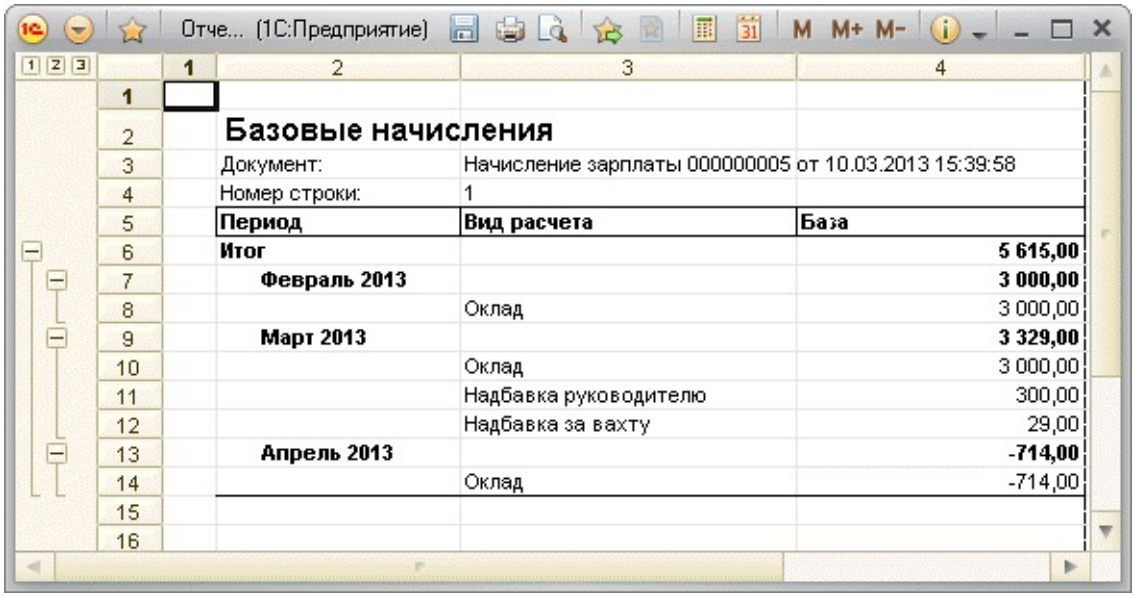


Рис. 3.89. Получение базовых данных для записи документа

Этот фрагмент можно посмотреть в обработке *ДанныеРегистраРасчета* в демонстрационной конфигурации «Сложные периодические расчеты», которая находится на прилагаемом компакт-диске.

В случае необходимости получить базу по нескольким регистрам нужно объединять в запросе данные из нескольких виртуальных таблиц базовых данных. Или, наоборот, в случае, когда в одном регистре хранятся базовые данные для нескольких основных регистров расчета, нужно объединять в запросе данные из основных регистров.

Например, в следующем запросе получаются базовые данные из регистра расчета *ОсновныеНачисленияРегл* для регистров расчета *ДополнительныеНачисленияРегл* и *ОсновныеНачисленияРегл* (листинг 3.85).

Листинг 3.85. Получение базовых данных для нескольких регистров расчета

```
Запрос = Новый Запрос;
  Запрос.Текст = "
  |ВЫБРАТЬ
  |        База.ПериодРегистрацииРазрез КАК ПериодРегистрацииРазрез,
  |        База.ВидРасчетаРазрез,
  |        СУММА(База.РезультатБаза) КАК РезультатБаза
  |ИЗ
  |
  РегистрРасчета.ДополнительныеНачисленияРегл.БазаОсновныеНачисленияРегл(&Измерения,
  &Измерения, &Разрезы, Регистратор = &Регистратор И НомерСтроки = &НомерСтроки) КАК База
  |СГРУППИРОВАТЬ ПО
  |        База.ПериодРегистрацииРазрез,
  |        База.ВидРасчетаРазрез
  |
  |ОБЪЕДИНИТЬ ВСЕ
  |
  |ВЫБРАТЬ
  |        База1.ПериодРегистрацииРазрез КАК ПериодРегистрацииРазрез,
  |        База1.ВидРасчетаРазрез,
  |        СУММА(База1.РезультатБаза) КАК РезультатБаза
  |ИЗ
  |        РегистрРасчета.ОсновныеНачисленияРегл.БазаОсновныеНачисленияРегл(&Измерения,
  &Измерения, &Разрезы, Регистратор = &Регистратор И НомерСтроки = &НомерСтроки) КАК База1
  |СГРУППИРОВАТЬ ПО
  |        База1.ПериодРегистрацииРазрез,
  |        База1.ВидРасчетаРазрез
  |
  |УПОРЯДОЧИТЬ ПО
  |        ПериодРегистрацииРазрез
  |ИТОГИ
  |        СУММА(РезультатБаза)
  |ПО
  |        ОБЩИЕ,
  |        ПериодРегистрацииРазрез";

  // Сформировать массив измерений основного и базового регистров
  // (названия измерений совпадают, поэтому используется один массив).
  Измерения = Новый Массив(2);
  Измерения[0] = "ФизЛицо";
  Измерения[1] = "Организация";

  // Сформировать массив разрезов.
  Разрезы = Новый Массив(2);
  Разрезы[0] = "ПериодРегистрации";
  Разрезы[1] = "ВидРасчета";
```

```
// Передать параметры в запрос.
Запрос.УстановитьПараметр("Измерения", Измерения);
Запрос.УстановитьПараметр("Разрезы", Разрезы);

// Запрос строится по конкретной записи документа с номером ВыбранныйНомерСтроки.
Запрос.УстановитьПараметр("Регистратор", СсылкаНаДокумент);
Запрос.УстановитьПараметр("НомерСтроки", ВыбранныйНомерСтроки);

Результат = Запрос.Выполнить();
```

Перерасчеты

Часто при вводе и изменении одних записей регистра расчета другие записи могут потерять свою актуальность. Например, при изменении оклада надбавки, зависящие от его значения, уже не будут ему соответствовать. Платформа позволяет автоматически отслеживать состав записей, потерявших актуальность в результате изменения или ввода других записей в регистры расчета. Для этого используется механизм перерасчетов.

Перерасчет в системе представляет собой отдельную таблицу базы данных. В ней хранится информация о записях конкретного регистра расчета, которые необходимо перерассчитать.

Таблица перерасчета в информационной базе содержит следующий состав полей:

- *<Имя измерения>* – поле, содержащее значения измерения перерасчета с именем, заданным в конфигурации;
- *ВидРасчета* – имеет тип *ПланВидовРасчетаСсылка.<имя>*. Содержит ссылку на вид расчета, который необходимо перерассчитать для объекта перерасчета (документа);
- *ОбъектПерерасчета* – имеет тип *ДокументСсылка.<имя>*. Содержит ссылку на документ, который необходимо перерассчитать.

В нашей демонстрационной конфигурации созданы два объекта перерасчета: *ПерерасчетОсновныхНачислений* с измерениями *ФизЛицо* и *Организация*, который хранит информацию о том, какие записи нужно перерассчитать в регистре расчета *ОсновныеНачисленияРегл,* и *ПерерасчетДополнительныхНачислений* с измерениями *ФизЛицо* и *Организация*, который хранит информацию о том, какие записи нужно перерассчитать в регистре расчета *ДополнительныеНачисленияРегл*.

Формирование записей таблицы перерасчета происходит при записи в регистр расчета данных по видам расчета, которые являются ведущими по отношению к другим видам расчета. Например, в списке ведущих для таких видов расчета, как *Надбавка руководителю* и *Надбавка за вахту*, содержится вид расчета *Оклад*. Это означает, что при вводе или изменении записи об окладе в таблицу перерасчета *ПерерасчетОсновныхНачислений* попадет запись о тех надбавках, которые в результате этого потеряли актуальность.

Так как ведущими могут выступать виды расчета любых планов видов расчета конфигурации, при вводе записи в один регистр расчета могут возникать записи в таблицах перерасчета нескольких регистров. Поэтому для получения всех записей, которые требуется перерассчитать, нужно объединять в запросе данные из нескольких таблиц перерасчета (листинг 3.86).

Листинг 3.86. Получение списка записей, которые требуется перерассчитать

```
ВЫБРАТЬ
  ПерерасчетОсновныхНачислений.ОбъектПерерасчета,
  ПерерасчетОсновныхНачислений.ВидРасчета,
  ПерерасчетОсновныхНачислений.ФизЛицо,
  ПерерасчетОсновныхНачислений.Организация
ИЗ
  РегистрРасчета.ОсновныеНачисленияРегл.ПерерасчетОсновныхНачислений КАК
  ПерерасчетОсновныхНачислений

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
  ПерерасчетДополнительныхНачислений.ОбъектПерерасчета,
  ПерерасчетДополнительныхНачислений.ВидРасчета,
  ПерерасчетДополнительныхНачислений.ФизЛицо,
  ПерерасчетДополнительныхНачислений.Организация
ИЗ
  РегистрРасчета.ДополнительныеНачисленияРегл.ПерерасчетДополнительныхНачислений КАК
  ПерерасчетДополнительныхНачислений
```

Результат выполнения запроса представлен на рис. 3.90.

Запрос: РегистрРасчета.ОсновныеНачисленияРегл.ПерерасчетОсновныхНачислений (Записей в результате: 3)

ОбъектПерерасчета	ВидРасчета	ФизЛицо	Организация
Начисление зарплаты 000000004 от 10.03.2013 15:24:50	Надбавка руководителю	Иванов	Ромашка
Начисление зарплаты 000000004 от 10.03.2013 15:24:50	Надбавка руководителю	Кузнецова	Ромашка
Начисление зарплаты 000000004 от 10.03.2013 15:24:50	Надбавка за вахту	Иванов	Ромашка

Рис. 3.90. Получение списка записей, которые требуется перерассчитать

Важно понимать, что записи перерасчета имеют рекомендательный характер, то есть платформа лишь сигнализирует, что определенные записи потеряли актуальность. Собственно алгоритм перерасчета записей разработчик должен предусмотреть самостоятельно. По завершении перерасчета записи должны быть удалены из таблицы перерасчета. Поэтому при формировании процедуры перерасчета разработчик должен предусмотреть удаление записей перерасчета средствами встроенного языка.

Глава 4. Оптимизация запросов

Теперь, после того как мы научились создавать синтаксически правильные запросы, выполнять и обрабатывать результаты запросов во встроенном языке и применять их для решения различных прикладных задач в системе «1С:Предприятие», рассмотрим подробно некоторые приемы по оптимизации запросов.

Дело в том, что решить одну и ту же задачу можно разными способами, но при этом важно не только добиться поставленной цели, но и сделать это максимально быстро, с минимальной нагрузкой на систему в целом. В результате повысится работоспособность прикладного решения и комфортность работы пользователей с ним.

Напомним, что информация, получаемая с помощью запросов, хранится в базе данных – файловой базе «1С:Предприятия» или в базах под управлением сторонних СУБД: SQL Server, PostgreSQL, IBM DB2, Oracle Database. Для получения нужных данных платформа транслирует запрос, созданный разработчиком, с учетом параметров, заданных пользователем, в запрос к базе данных на языке SQL.

В тексте запроса содержится информация, какие данные, из каких таблиц и по каким условиям требуется получить. Но каким образом выполнять запрос, «решает» сама СУБД. Для этого на основании текста запроса, имеющихся индексов таблиц и статистики оптимизатор СУБД строит план запроса – набор физических операторов, которые необходимо выполнить для получения запрошенных данных. Посмотреть, какой план запроса используется, можно через технологический журнал или средствами самой СУБД.

Как сделать так, чтобы СУБД выбирала оптимальный план запроса для получения информации из базы данных? Как сделать текст запроса наиболее эффективным для выполнения? Этим вопросам и будет посвящена данная глава.

Индексирование таблиц

Одним из основных условий для построения эффективного плана запроса является наличие подходящих индексов. Индексы нужны для ускорения получения данных из таблиц. Они служат своеобразной картой для нахождения нужных данных.

Если провести аналогию с адресом проживания человека, то понятно, что, зная полный адрес (улицу, дом, подъезд, этаж, квартиру), можно легко и довольно быстро до него добраться. Но если вы не знаете квартиру, придется обходить весь этаж. Если вы не знаете этаж, придется обходить весь подъезд. Если вы не знаете подъезд, придется обходить весь дом и т. д. А если вы знаете только улицу и квартиру, но не знаете дом, то тут вообще «комментарии излишни».

Для быстрого получения нужной записи из таблицы, в которой могут быть тысячи, десятки тысяч и миллионы записей, также требуется индекс, наиболее полно и точно описывающий, как эту запись найти. Поиск по индексу позволяет максимально быстро

получить требуемые данные.

При отсутствии индекса у таблицы, чтобы выбрать нужные записи, придется последовательно перебирать все записи таблицы и смотреть, подходит текущая запись под условия или нет. При этом выполняется сканирование всей таблицы. Это наиболее медленная операция, особенно на больших объемах данных.

При наличии индекса, неточно описывающего условия поиска, понадобится сканировать этот индекс. Конечно, сканирование индекса быстрее, чем сканирование таблицы, но не намного.

Чтобы представлять, как происходит поиск данных в таблицах, необходимо понимать, что структура индекса в СУБД представляет собой дерево значений проиндексированных полей. На первом уровне дерева находятся значения первого поля индекса, на втором – второго и так далее. Чтобы выполнить поиск данных по индексу, сначала необходимо провести поиск по значению первого поля индекса, затем – второго и так далее. Если, например, условие по первому полю индекса не указано, то индекс уже не сможет обеспечить быстрый поиск. Если указано условие по нескольким первым полям индекса, а затем одно или несколько полей индекса не задано, то индекс может быть использован только частично.

Таким образом, индексы будут использованы эффективно только в том случае, когда индекс покрывает условия запроса, то есть с самого начала содержит без пропуска поля, на которые накладываются условия.

Условия используются в следующих секциях запроса:

- в предложении *ГДЕ*,
- в условии соединения таблиц *ПО*,
- в параметрах виртуальных таблиц,
- в предложении *ИМЕЮЩИЕ*.

Для каждого условия должен существовать подходящий индекс (полностью покрывающий условия запроса), т. е. индекс, удовлетворяющий следующим требованиям:

1. Индекс должен содержать все поля, перечисленные в условии.
2. Эти поля должны находиться в самом начале индекса.
3. Эти поля должны следовать друг за другом подряд, то есть между ними не должны «вклиниваться» поля, не участвующие в условии запроса.

Что значит подходящий индекс? Рассмотрим пример.

Предположим, нам нужно получить данные об остатке товара *Клавиатура* на складе

Основной в организации *Компьютерный мир* на *01.05.13* из регистра накопления *Остатки товаров*.

В базе данных для таблицы остатков регистра накопления (без разделителей итогов) автоматически создается индекс *Период + Измерение1 + Измерение2 + … + ИзмерениеN*. При этом имена измерений и порядок следования их друг за другом в индексе соответствуют порядку, заданному в конфигураторе.

Регистр накопления *Остатки товаров* имеет измерения *Организация*, *Склад*, *Номенклатура*. В базе данных для таблицы остатков этого регистра накопления будет создан индекс *Период + Организация + Склад + Номенклатура*.

Для решения поставленной задачи в запросе к таблице остатков регистра накопления требуется задать условие отбора записей в параметрах виртуальной таблицы. Это условие отбора по полю *Период* (в параметре *Период*) и по полям *Организация*, *Склад* и *Номенклатура* (в параметре *Условие*). Таким образом, индекс удовлетворяет всем трем требованиям, описанным выше. Поэтому поиск нужных записей в таблице будет эффективным (рис. 4.1).

Таблица остатков регистра накопления "ОстаткиТоваров"

Период	Организация	Склад	Товар	Остаток
01.04.13	...	...	...	...
01.05.13	...	...	...	...
01.05.13	Компьютер Лэнд	...	...	...
01.05.13	Компьютерный мир	...	...	...
01.05.13	Компьютерный мир	Розничный	...	...
01.05.13	Компьютерный мир	Оптовый	...	...
01.05.13	Компьютерный мир	Оптовый	Монитор	30
01.05.13	Компьютерный мир	Оптовый	Клавиатура	100
01.05.13	Компьютерный мир	Оптовый	Принтер	50
01.05.13	Компьютерный мир	Оптовый	...	...
01.05.13	Компьютерный мир	Розничный	...	...
01.05.13	Компьютерный мир	...	...	...
01.05.13	Компьютер Лэнд	...	...	...
01.05.13	...	...	...	...
01.06.13	...	...	...	...

Рис. 4.1. Поиск записей в таблице по индексу

Если же изменить условие задачи, то ситуация будет другой. Например, нужно получить данные об остатке товара *Клавиатура* в организации *Компьютерный мир* на *01.05.13*.

Для поиска нужных данных в таблице будут выбраны все записи, относящиеся к периоду *01.05.13* и организации *Компьютерный мир*. Затем последовательным перебором запрос выберет те из них, которые относятся к товару *Клавиатура* (рис. 4.2).

Таблица остатков регистра накопления "ОстаткиТоваров"

Период	Организация	Склад	Товар	Остаток
01.04.13	...	...	...	...
01.05.13	...	...	...	...
01.05.13	Компьютер Лэнд	...	...	...
01.05.13	Компьютерный мир	...	...	...
01.05.13	Компьютерный мир	Розничный	...	...
01.05.13	Компьютерный мир	Оптовый	...	...
01.05.13	Компьютерный мир	Оптовый	Монитор	30
01.05.13	Компьютерный мир	Оптовый	Клавиатура	100
01.05.13	Компьютерный мир	Оптовый	Принтер	50
01.05.13	Компьютерный мир	Оптовый	...	...
01.05.13	Компьютерный мир	Розничный	...	...
01.05.13	Компьютерный мир	...	...	...
01.05.13	Компьютер Лэнд	...	...	...
01.05.13	...	...	...	...
01.06.13	...	...	...	...

Рис. 4.2. Поиск записей в таблице путем сканирования индекса

Так происходит потому, что не выполняется третье требование к индексу – между полями условия запроса *Организация* и *Товар* «вклинивается» поле *Склад*. В этом случае индекс может быть использован частично, и для поиска нужных данных, скорее всего, будет применено сканирование индекса.

подробнее

Подробные рекомендации по эффективному использованию индексов будут рассмотрены в этом разделе.

При создании таблиц, соответствующих тем или иным объектам конфигурации, платформа автоматически создает набор индексов, обеспечивающих эффективную работу с этими

таблицами. Индексов, как правило, создается несколько, для того чтобы можно было выбрать различные данные, по различным условиям. Основные индексы, создаваемые платформой:

- индекс по уникальному идентификатору (ссылке) для всех объектных сущностей (справочники, документы и т. д.);
- индекс по регистратору (ссылке на документ) для таблиц движений регистров, подчиненных регистратору;
- индекс по периоду и значениям всех измерений для итоговых таблиц регистров накопления;
- индекс по периоду, счету и значениям всех измерений для итоговых таблиц регистров бухгалтерии.

подробнее

Состав индексов, создаваемых платформой, можно найти в статье ИТС «Индексы таблиц базы данных».

Посмотреть, какие индексы создаются на конкретной информационной базе, можно с помощью метода глобального контекста *ПолучитьСтруктуруХраненияБазыДанных()*.

Но нужно понимать, что чем сложнее индекс и чем больше количество индексов в таблице, тем больше времени тратится на запись данных в эту таблицу. Поэтому нужно стремиться оптимизировать состав и количество индексов так, чтобы индексов не было слишком много и они не были слишком сложными (не содержали много полей), но при этом позволяли точно и быстро выбирать данные из таблиц по различным критериям.

Способы индексирования таблиц

Платформа не может предусмотреть индексы на все случаи жизни, так как запросы могут быть самыми разнообразными. Поэтому разработчик может дополнительно, если существует такая необходимость, указать те или иные поля, которые также должны участвовать в построении индекса.

Существуют следующие способы создания и/или изменения индексов:

- изменение порядка следования измерений объектов метаданных;
- использование свойства *Индексировать* у реквизитов объектов метаданных;
- включение реквизитов объектов метаданных в критерий отбора.

Например, для непериодического независимого регистра сведений (при наличии хотя бы одного измерения) в базе данных автоматически создается индекс:

Измерение1 + *Измерение2* + … + *ИзмерениеN*

При этом имена полей индекса и порядок следования их друг за другом соответствуют

заданному в конфигураторе.

- Если поменять порядок следования друг за другом измерений регистра в конфигураторе, то, соответственно, изменится порядок следования полей в индексе таблицы регистра в базе данных (рис. 4.3).

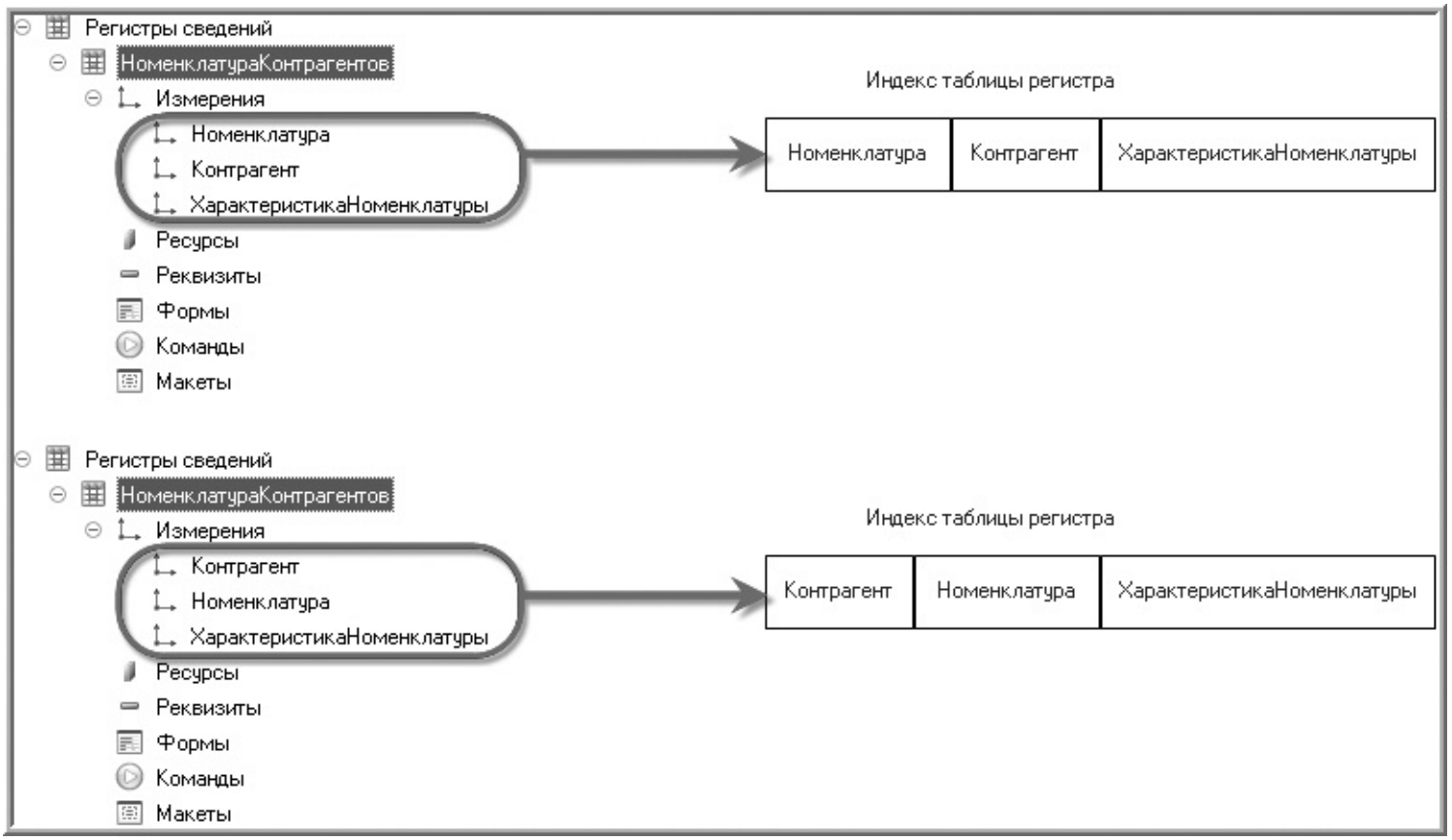


Рис. 4.3. Изменение порядка следования измерений регистра в конфигураторе и полей индекса таблицы регистра в базе данных

- При установке свойств измерений *Ведущее* и *Индексировать* (рис. 4.4) создается дополнительный индекс:

ИзмерениеК + Измерение1 + Измерение2 + … + ИзмерениеN.

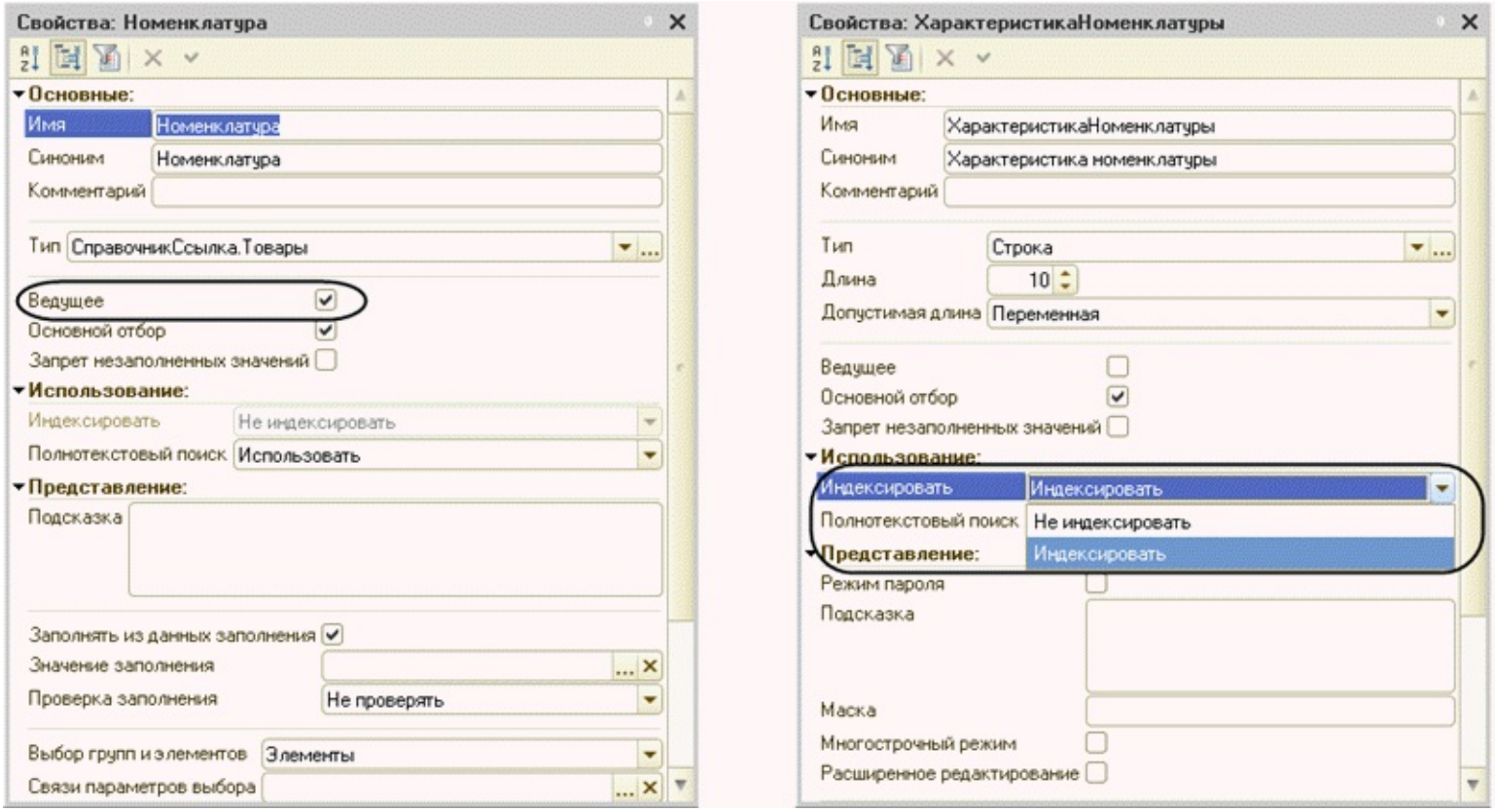


Рис. 4.4. Свойства измерений «Ведущее» и «Индексировать»

Поэтому в случае установки свойства *Ведущее* для измерения *Номенклатура*, а затем установки свойства *Индексировать* для измерения *ХарактеристикаНоменклатуры* у таблицы регистра в базе данных будут созданы дополнительные индексы, показанные на рис. 4.5.

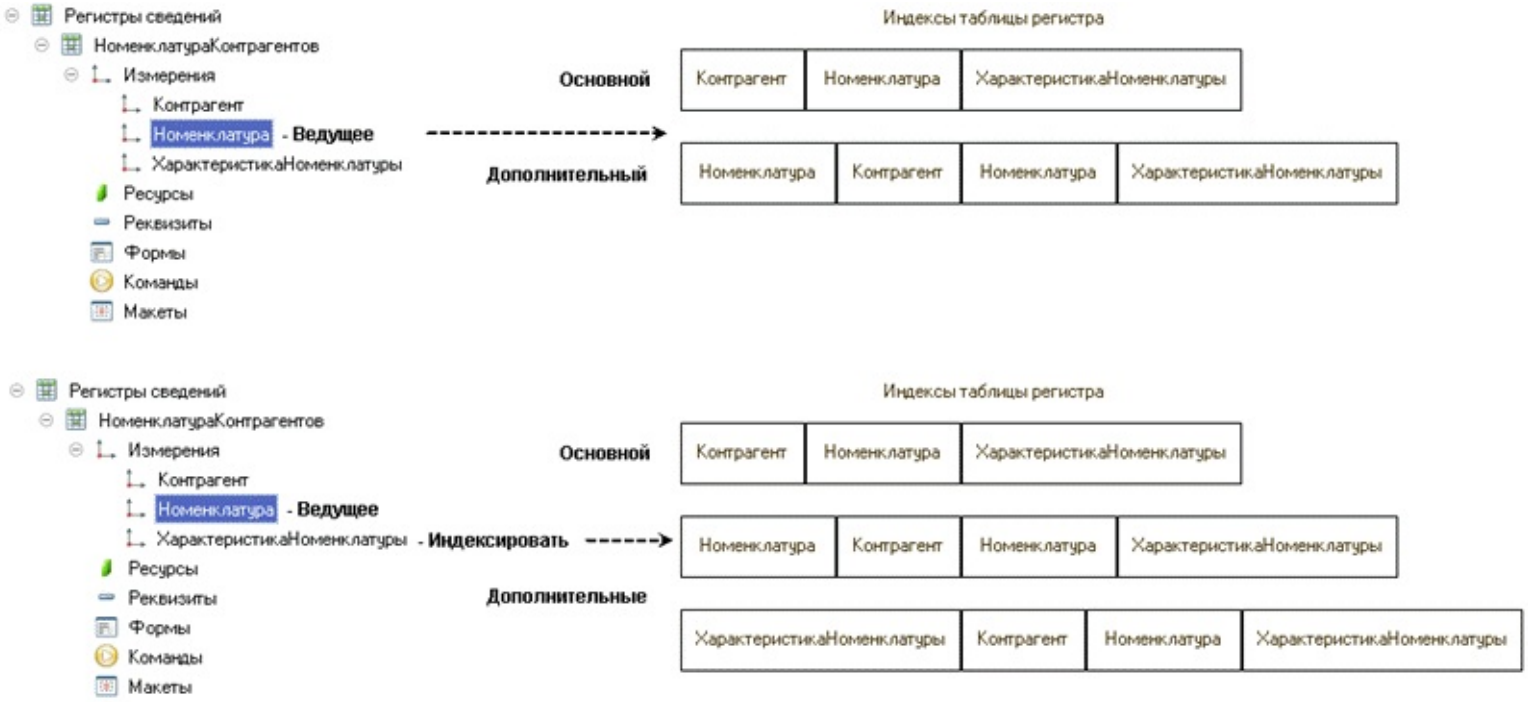


Рис. 4.5. Основные и дополнительные индексы таблицы регистра при индексировании измерений

Понятно, что в некоторых случаях, чтобы не усложнять индексы, достаточно изменить порядок следования измерений регистра в конфигураторе. Например, если поставить ведущее измерение на первое место, то вместо двух индексов будет создан один.

Свойство *Индексировать* есть также у некоторых реквизитов объектов метаданных, например у справочников, документов и т. п. Это свойство позволяет указать системе, что нужно создать дополнительный индекс, содержащий соответствующий реквизит. Кроме значения *Индексировать* для большинства объектов можно установить значение *Индексировать с доп. упорядочиванием*.

Например, для неиерархического неподчиненного справочника в базе данных автоматически создаются индексы:

- *Ссылка*;
- *Код* + *Ссылка* (если длина кода не равна 0);
- *Наименование* + *Ссылка* (если длина наименования не равна 0).

Если для реквизита справочника установить свойство *Индексировать* в значение *Индексировать/Индексировать с доп. упорядочиванием,* для таблицы справочника в базе данных будут созданы дополнительные индексы, показанные на рис. 4.6:

- *Реквизит* + *Ссылка* (если для реквизита свойство *Индексировать* установлено в значение *Индексировать*);
- *Реквизит* + *Код* + *Ссылка* (если для реквизита свойство *Индексировать* установлено в значение *Индексировать с доп. упорядочиванием*, длина кода не равна 0 и основное представление справочника задано в виде кода);
- *Реквизит* + *Наименование* + *Ссылка* (если для реквизита свойство *Индексировать* установлено в значение *Индексировать с доп. упорядочиванием*, длина наименования не равна 0 и основное представление справочника задано в виде наименования).

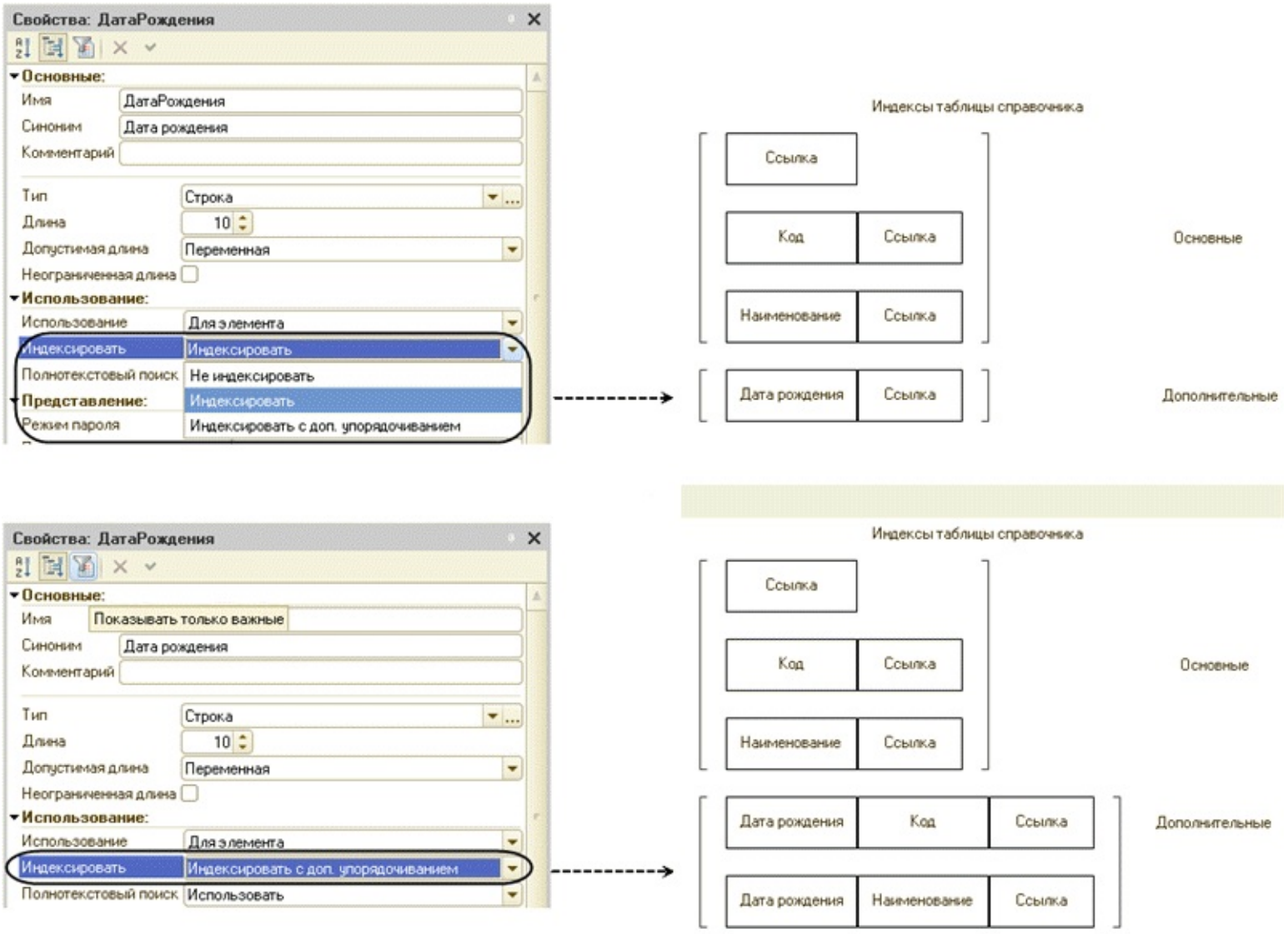


Рис. 4.6. Основные и дополнительные индексы таблицы справочника при индексировании реквизитов

Вариант индексирования *Индексировать с доп. упорядочиванием* предназначен, прежде всего, для использования в динамических списках. В этом случае индекс строится по реквизиту, а также по некоторому полю, которое обычно используется для упорядочивания объектов этого типа.

При выборе значения свойства *Индексировать* нужно исходить из того, какие варианты выборки информации необходимо оптимизировать в первую очередь. Если требуется только поиск с помощью запроса объектов по данному реквизиту без упорядочивания, то можно установить значение *Индексировать*, чтобы создаваемый индекс требовал меньше ресурсов системы. Если требуется просмотр списка с отбором по реквизиту, то имеет смысл использовать вариант *Индексировать с доп. упорядочиванием*.

Если объект конфигурации включен в критерий отбора, то создается дополнительный индекс по реквизиту, который включен в состав критерия отбора. Например, если справочник включен в критерий отбора через реквизит *ДатаРождения*, то у таблицы справочника будет создан дополнительный индекс по этому реквизиту (рис. 4.7).

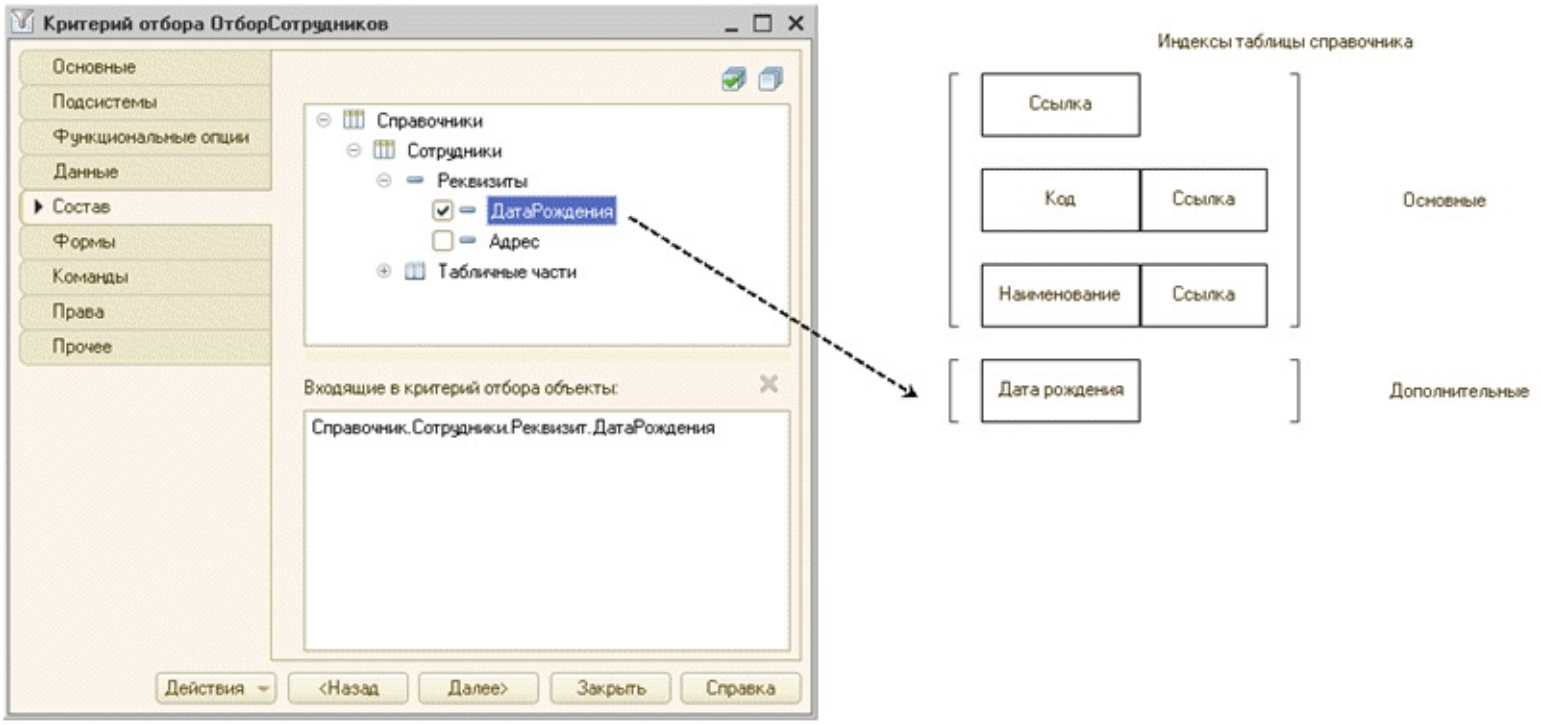


Рис. 4.7. Основные и дополнительные индексы таблицы справочника при включении реквизита в состав критерия отбора

Эффективное использование индексов

Чтобы понять, эффективно запрос использует индексы или нет, нужно посмотреть текст запроса, определить таблицы – источники запроса и выделить те поля, которые используются в условиях запроса (в предложении *ГДЕ*, в условии соединения таблиц *ПО*, в параметрах виртуальных таблиц, в предложении *ИМЕЮЩИЕ*).

Затем нужно сравнить их с индексами, которые создает система для исходных таблиц запроса. Если нужных для эффективного выполнения запроса индексов не хватает, следует создать дополнительные индексы описанными выше способами.

Ниже мы рассмотрим некоторые задачи по получению информации из таблиц базы данных и варианты индексирования этих таблиц, оптимальные для выполнения соответствующих запросов.

Пример 1

Пусть необходимо получить движения регистра *ТоварыНаСкладах*, отражающие поступления от производителей. Для этого нужно выполнить отбор движений по значению реквизита регистра *ВидОперации* с помощью следующего запроса (листинг 4.1).

Листинг 4.1. Отбор движений регистра накопления по значению реквизита

```
ВЫБРАТЬ
  ТоварыНаСкладах.Период,
  ТоварыНаСкладах.Регистратор КАК Регистратор,
  ТоварыНаСкладах.Поставщик,
  ТоварыНаСкладах.Номенклатура,
  ТоварыНаСкладах.Склад,
  ТоварыНаСкладах.Количество
ИЗ
```

```
  РегистрНакопления.ТоварыНаСкладах КАК ТоварыНаСкладах
ГДЕ
  ТоварыНаСкладах.ВидОперации =
  ЗНАЧЕНИЕ(Перечисление.ВидыОпераций.ПоступлениеОтПроизводителей)
```

В данном запросе источником данных является таблица движений регистра накопления остатков *ТоварыНаСкладах*, который имеет измерения *Номенклатура* и *Склад*, ресурс *Количество* и реквизиты *Поставщик, ВидОперации*. Измерения и реквизиты регистра не проиндексированы, т. е. свойство *Индексировать* для них установлено в значение *Не индексировать*.

В этом случае для таблицы движений этого регистра платформа автоматически создаст индексы:

- *Период + Регистратор + НомерСтроки*,
- *Регистратор + НомерСтроки*.

В условии запроса, в предложении *ГДЕ*, накладывается отбор на значение реквизита регистра *ВидОперации*. То есть подходящий индекс, содержащий в самом начале поле, на которое накладывается условие, в таблице отсутствует.

В этом случае при выполнении запроса будут выбраны все записи движений регистра, а затем, путем перебора этих записей, будут получены записи с нужным значением реквизита.

Чтобы при выполнении запроса происходило обращение сразу к записям с указанным значением реквизита, для реквизита *ВидОперации* свойство *Индексировать* следует установить в значение *Индексировать*. В этом случае в таблице движений регистра будет создан дополнительный индекс:

ВидОперации + Период + Регистратор + НомерСтроки.

Этот индекс полностью покрывает условия запроса и позволит осуществить эффективный поиск записей в таблице движений регистра накопления.

Пример 2

Пусть необходимо получить данные о продажах товаров конкретному покупателю за указанный период с разворотом по неделям из оборотного регистра накопления *Продажи*. Для этого нужно выполнить следующий запрос (листинг 4.2).

Листинг 4.2. Вывод оборотов товаров по выбранному контрагенту за заданный период с периодичностью «Неделя»

```
ВЫБРАТЬ
  ПродажиОбороты.Период,
  ПродажиОбороты.Номенклатура,
  ПродажиОбороты.КоличествоОборот КАК Количество,
  ПродажиОбороты.СуммаОборот КАК Сумма
```

```
ИЗ
  РегистрНакопления.Продажи.Обороты(&НачалоПериода, КОНЕЦПЕРИОДА(&КонецПериода, ДЕНЬ),
  НЕДЕЛЯ, Контрагент = &Покупатель) КАК ПродажиОбороты

УПОРЯДОЧИТЬ ПО
  Период
```

В данном запросе источником данных является таблица оборотов регистра накопления *Продажи*, который имеет измерения *Номенклатура* и *Контрагент*, ресурсы *Количество* и *Сумма*, а также реквизит *ВидОперации*. Измерения и реквизит регистра не проиндексированы, т. е. свойство *Индексировать* для них установлено в значение *Не индексировать*.

В базе данных для таблицы оборотов регистра накопления будет создан индекс: *Период + Номенклатура + Контрагент*.

В параметре *Условие* виртуальной таблицы мы передаем условие отбора по полю *Контрагент*, по значению которого должны быть отобраны записи итоговой таблицы регистра накопления при построении виртуальной таблицы.

Для эффективного отбора записей в таблице оборотов должен существовать подходящий индекс, содержащий в самом начале поле *Контрагент*, на которое накладывается условие. Как мы видим, этот индекс в таблице отсутствует.

В этом случае для выполнения запроса серверу СУБД придется перебирать (сканировать) все записи таблицы. Время выполнения этой операции напрямую зависит от количества записей в таблице оборотов регистра и может быть очень большим.

Для решения проблемы возможны следующие варианты:

- Проиндексировать измерение *Контрагент*. В этом случае в таблице оборотов регистра накопления будет создан дополнительный индекс: *Контрагент + Период*.
- Поставить измерение *Контрагент* первым в списке измерений. Однако при использовании этого метода нужно быть внимательным, так как в конфигурации могут присутствовать другие запросы, которые могут замедлиться в результате этой перестановки.

Пример 3

Пусть необходимо получить данные об остатке конкретного товара в определенной организации на указанный период из регистра накопления остатков *ОстаткиТоваров*. Для этого нужно выполнить следующий запрос (листинг 4.3).

Листинг 4.3. Вывод остатков конкретного товара на заданный период в определенной организации

```
ВЫБРАТЬ
  ОстаткиТоваровОстатки.Организация,
  ОстаткиТоваровОстатки.Склад,
```

```
  ОстаткиТоваровОстатки.Номенклатура,
  ОстаткиТоваровОстатки КоличествоОстаток КАК Количество
ИЗ
  РегистрНакопления.ОстаткиТоваров.Остатки (&Период, Организация = & Организация И
  Номенклатура = &Товар) КАК ОстаткиТоваровОстатки
```

В данном запросе источником данных является таблица остатков регистра накопления *ОстаткиТоваров*, который имеет непроиндексированные измерения *Организация*, *Склад*, *Номенклатура*. В базе данных для таблицы остатков регистра накопления будет создан индекс:

Период + *Организация* + *Склад* + *Номенклатура*.

Для решения поставленной задачи в запросе к таблице остатков регистра накопления задано условие отбора записей в параметрах виртуальной таблицы. Это условие отбора по полю *Период* (в параметре *Период*) и по полям *Организация* и *Номенклатура* (в параметре *Условие*).

Таким образом, существующий индекс является неэффективным, так как между полями условия запроса *Организация* и *Товар* «вклинивается» поле *Склад*. В этом случае индекс может быть использован частично, и для поиска нужных данных, скорее всего, будет применено сканирование индекса.

Для решения проблемы возможны следующие варианты:

- добавить в запрос условие по измерению *Склад*,
- поменять местами измерения *Склад* и *Номенклатура*.

Заметьте, что порядок следования условий в запросе не обязан совпадать с порядком следования полей в индексе. Главное, чтобы в индексе присутствовали все поля, использованные в условии, они находились в начале индекса и шли без пропусков друг за другом.

Пример 4
Пусть необходимо вывести данные обо всех товарах и их производителях с соблюдением иерархии справочника *Товары*. Для каждого товара из справочника нужно вывести дату его поступления и поставщика из приходных накладных. Для этого нужно выполнить следующий запрос (листинг 4.4).

Листинг 4.4. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении

```
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  Поступление.Ссылка.Дата,
  Поступление.Ссылка.Поставщик
ИЗ
  Справочник.Товары КАК Товары
```

```
    ЛЕВОЕ СОЕДИНЕНИЕ Документ.ПриходнаяНакладная.Состав КАК Поступление
    ПО Товары.Ссылка = Поступление.Товар

УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

В данном запросе источником данных являются таблицы справочника *Товары* и документа *ПриходнаяНакладная*, связанные между собой левым соединением. Для эффективного поиска записей в таблицах поля, участвующие в условии соединения таблиц в запросе (*ПО Товары.Ссылка = Поступление.Товар*), также должны быть проиндексированы.

В таблице справочника *Товары* автоматически создается индекс по полю *Ссылка*, а для документа *ПриходнаяНакладная* реквизит *Товар* нужно проиндексировать. В этом случае в таблице документа будет создан дополнительный индекс: *Товар + Ссылка*.

В результате при выполнении запроса для каждого товара из справочника СУБД будет сразу обращаться к записям о поступлении этого товара из таблицы документа, и связь данных в таблицах будет выполнена более оптимально.

Пример 5

Немного усложним условие предыдущего примера так, чтобы в результат запроса попадали все данные из справочника *Товары* вместе с данными о поступлении товаров за ноябрь. В этом случае самым оптимальным вариантом решения задачи будет использование пакетного запроса и временной таблицы с данными о поступлении товаров. Для этого нужно выполнить следующий запрос (листинг 4.5).

Листинг 4.5. Вывод всех товаров в порядке иерархии справочника «Товары» с данными об их поступлении за ноябрь

```
ВЫБРАТЬ
  Поступление.Товар,
  Поступление.Ссылка.Дата КАК Дата,
  Поступление.Ссылка.Поставщик
ПОМЕСТИТЬ ПоступлениеТоваров
ИЗ
  Документ.ПриходнаяНакладная.Состав КАК Поступление
ГДЕ
  МЕСЯЦ(Поступление.Ссылка.Дата) = 11
ИНДЕКСИРОВАТЬ ПО Товар
;
ВЫБРАТЬ
  Товары.Наименование,
  Товары.Производитель,
  ПоступлениеТоваров.Дата,
  ПоступлениеТоваров.Поставщик
ИЗ
  Справочник.Товары КАК Товары
    ЛЕВОЕ СОЕДИНЕНИЕ ПоступлениеТоваров КАК ПоступлениеТоваров
    ПО Товары.Ссылка = ПоступлениеТоваров.Товар

УПОРЯДОЧИТЬ ПО
  Товары.Наименование ИЕРАРХИЯ
```

В данном пакетном запросе сначала данные о поступлении товаров за ноябрь помещаются во временную таблицу *ПоступлениеТоваров*. В следующем запросе справочник товаров связывается левым соединением с этой временной таблицей по ссылкам товаров.

Для быстрой и эффективной выборки данных из временной таблицы необходимо проиндексировать эту таблицу по полям, которые участвуют в условии соединения (в данном случае по полю *Товар*). Делается это с помощью предложения *ИНДЕКСИРОВАТЬ ПО*. В результате соединение с временной таблицей, проиндексированной по полям, участвующим в условии связи, будет выполнено более эффективно.

Если же вместо временной таблицы использовать вложенный запрос и в условии соединения использовать непроиндексированное поле, то такое соединение будет выполняться крайне медленно. Время сканирования таблицы, участвующей в соединении, увеличивается в разы. Поэтому таких ситуаций настоятельно рекомендуется избегать.

Общие рекомендации

Рекомендации по созданию дополнительных индексов в таблицах, приведенные выше, не следует воспринимать буквально. Их нужно критически переосмысливать в зависимости от специфики конкретной прикладной области, объема используемой информационной базы и приоритетов пользователей по решению тех или иных задач. Нужно понимать, что оптимизировать сразу все не получится. Какие-то отчеты пользователям необходимо получать часто и быстро, а в редких случаях они могут и подождать.

Поэтому не нужно нагружать систему индексами на все случаи жизни. Во-первых, чем больше индексов в таблице, тем больше тратится времени на запись данных в эту таблицу. А во-вторых, когда индексы слишком сложные и их много, то СУБД не может подобрать среди них подходящий индекс за ограниченное время и может выбрать неоптимальный план запроса.

Также следует иметь в виду, что на маленьких таблицах (порядка нескольких тысяч записей и меньше) СУБД практически всегда использует сканирование таблицы либо сканирование кластерного (автоматически созданного системой) индекса, так как это наиболее простое решение в данном случае. Так что нет смысла дополнительно индексировать заведомо небольшие таблицы. С другой стороны, часто трудно заранее предугадать размер таблицы.

То есть понятно, что создание дополнительных индексов может положительно сказаться на выборке данных из больших таблиц и не дать никакого эффекта на маленьких объемах данных (а иногда и замедлить быстродействие системы в целом).

Не следует также создавать индексы по низкоселективным полям. Селективность индекса отражает процент записей, которые по условию можно выбрать из таблицы.

Максимально высокая селективность индекса – у первичного ключа. Если реквизит имеет тип *Булево*, то индексировать его имеет смысл только в том случае, если незначительная часть записей таблицы всегда принимает одно значение (например, *ЛОЖЬ*) и нужно выбрать из таблицы записи с этим значением.

В целом к расстановке индексов в таблицах нужно подходить осмысленно, творчески и учитывать накопленный опыт, как личный, так и опыт функционирования подобных прикладных решений.

Причины неоптимальной работы запросов и основные направления их оптимизации

В данном разделе рассматриваются типичные ошибки при написании запросов, которые могут быть легко обнаружены при анализе кода конфигурации и структуры метаданных. В результате анализа этих ошибок даются рекомендации по их исправлению, направленные на повышение эффективности запросов.

Примеры, используемые в данном разделе, можно посмотреть в демонстрационной конфигурации «Учет движения средств», которая находится на прилагаемом компакт-диске, в обработке *Оптимизация запросов*.

Общие рекомендации

Хотя может показаться, что ниже повторяются общеизвестные вещи, но все же подобные ошибки довольно часто встречаются:

- Нужно стараться минимизировать объем выборки, то есть выбирать запросом только те данные, которые нужны.
- Не нужно пытаться любой ценой перенести всю функциональность задачи в СУБД и сосредоточивать всю бизнес-логику в одном запросе. Часто это приводит к существенному усложнению запроса, и СУБД, скорее всего, ошибется при выборе плана такого запроса.
- Нужно стараться писать сами запросы и условия соединения таблиц в них как можно более простыми. Особенно плохо использовать в соединении подзапросы.
- Не следует использовать в запросе много источников (более 5–7 таблиц). При этом время на анализ запроса оптимизатором СУБД растет нелинейно, и в итоге за лимитированное время может быть выбран неоптимальный план запроса.
- Нужно стараться по возможности не использовать запросы к базе данных, результат которых может неограниченно расти с ростом самой информационной базы, так как это может привести к снижению производительности и повышению вероятности аварийного завершения работы запроса. Для обработки данных потенциально неограниченного объема следует организовывать «курсорный» механизм получения результата запроса:
- устанавливать уникальный порядок сортировки получаемых данных;

- получать выборки по частям с помощью оператора *ПЕРВЫЕ*, используя данные последней полученной записи предыдущей выборки в качестве начального условия при получении очередной выборки.

Не использовать запросы в цикле

Многократное выполнение однотипных запросов в цикле является распространенной ошибкой новичков. Например, нужно получить остатки на складах для нескольких элементов номенклатуры, содержащихся в массиве номенклатуры *МассивНоменклатуры*.

Для решения данной задачи можно написать запрос к таблице остатков регистра накопления *ОстаткиНаСкладах* и передать в качестве параметра в виртуальную таблицу условие отбора по конкретному элементу номенклатуры. И затем выполнять данный запрос в цикле для каждого элемента из массива номенклатуры. Это можно сделать с помощью следующего запроса (листинг 4.6).

Листинг 4.6. Выполнение запроса для получения остатков номенклатуры в цикле – неправильный вариант

```
Для Каждого ЭлементНоменклатуры Из МассивНоменклатуры Цикл

  Запрос = Новый Запрос;
  Запрос.Текст =
    "ВЫБРАТЬ
    |         ТоварыНаСкладахОстатки.Номенклатура.Наименование КАК Номенклатура,
    |         ТоварыНаСкладахОстатки.КоличествоОстаток
    |ИЗ
    |         РегистрНакопления.ТоварыНаСкладах.Остатки( ,Номенклатура =
    &ВыбраннаяНоменклатура) КАК ТоварыНаСкладахОстатки";

  Запрос.УстановитьПараметр("ВыбраннаяНоменклатура", ЭлементНоменклатуры);

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

  // Обработка результатов запроса.
  …

КонецЦикла;
```

Однако такое решение будет неоптимальным, так как в результате запрос будет выполнен столько раз, сколько элементов содержится в массиве номенклатуры. Многократное выполнение одного и того же или похожих запросов в цикле занимает гораздо больше времени, чем выполнение одного запроса, получающего нужную совокупность данных за один раз.

Поэтому такой фрагмент кода нужно переписать следующим образом (листинг 4.7):

Листинг 4.7. Выполнение запроса для получения остатков номенклатуры за один раз – правильный вариант

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
```

```
  |        ТоварыНаСкладахОстатки.Номенклатура.Наименование КАК Номенклатура,
  |        ТоварыНаСкладахОстатки.КоличествоОстаток
  |ИЗ
  |        РегистрНакопления.ТоварыНаСкладах.Остатки(, Номенклатура В
  (&МассивНоменклатуры)) КАК ТоварыНаСкладахОстатки";

Запрос.УстановитьПараметр("МассивНоменклатуры", МассивНоменклатуры);

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

// Обработка результатов запроса.
…
```

Таким образом, остатки всех элементов номенклатуры могут быть получены за одно обращение к таблице остатков регистра накопления.

Но иногда требуемое условие запроса, позволяющее отказаться от выполнения в цикле, сформулировать бывает затруднительно или очень сложно. В этом случае как минимум нужно вынести создание запроса за пределы цикла, а в цикле изменять только параметры запроса. Это позволит избежать многократной синтаксической проверки текста запроса (листинг 4.8).

Листинг 4.8. Выполнение запроса для получения остатков номенклатуры в цикле

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ТоварыНаСкладахОстатки.Номенклатура.Наименование КАК Номенклатура,
  |        ТоварыНаСкладахОстатки.КоличествоОстаток
  |ИЗ
  |        РегистрНакопления.ТоварыНаСкладах.Остатки(,Номенклатура =
  &ВыбраннаяНоменклатура) КАК ТоварыНаСкладахОстатки";

Для Каждого ЭлементНоменклатуры Из МассивНоменклатуры Цикл

  Запрос.УстановитьПараметр("ВыбраннаяНоменклатура", ЭлементНоменклатуры);

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

  // Обработка результатов запроса.
  …

КонецЦикла;
```

Не использовать в запросе функции от параметров

В языке запросов существуют разнообразные функции – функции для работы со строками, функции для работы с датами и т. п. Нежелательно вычислять в запросе значение этих функций от переданных в запрос параметров.

Например, необходимо получить данные приходных накладных за определенный период. Для этого можно выполнить следующий запрос (листинг 4.9).

Листинг 4.9. Вывод документов «Поступление товаров» за определенный период – неправильный вариант

```
ВЫБРАТЬ
  Накладная.Дата,
  Накладная.Номер,
  Накладная.Контрагент
ИЗ
  Документ.ПоступлениеТоваров КАК Накладная
ГДЕ
  Дата МЕЖДУ НАЧАЛОПЕРИОДА(&Дата, МЕСЯЦ) И КОНЕЦПЕРИОДА(&Дата, МЕСЯЦ)
```

Казалось бы, все правильно – вся логика сосредоточена в одном месте. Но такой запрос будет работать неоптимально, так как СУБД «не поймет», что в условии запроса используется константа (значение параметра *Дата*).

Поэтому лучше сначала вычислить значение функции во встроенном языке и затем передать его в запрос как параметр (листинг 4.10).

Листинг 4.10. Вывод документов «Поступление товаров» за определенный период – правильный вариант

```
ВЫБРАТЬ
  Накладная.Дата,
  Накладная.Номер,
  Накладная.Контрагент
ИЗ
  Документ.ПоступлениеТоваров КАК Накладная
ГДЕ
  Дата МЕЖДУ &ДатаНачала И &ДатаОкончания

Запрос.УстановитьПараметр("ДатаНачала",НачалоМесяца(Дата));
Запрос.УстановитьПараметр("ДатаОкончания",КонецМесяца(Дата));
```

Использовать параметры виртуальных таблиц

Одним из основных условий оптимального выполнения запросов является максимальное использование параметров виртуальных таблиц во всех запросах, где это не искажает суть выполняемых действий.

Например, нужно получить остатки на складах для номенклатуры, переданной в параметре *Номенклатура*.

Для решения данной задачи можно написать запрос к таблице остатков регистра накопления *ОстаткиНаСкладах* и наложить условие отбора по конкретному элементу номенклатуры в предложении *ГДЕ* (листинг 4.11).

Листинг 4.11. Выполнение запроса для получения остатков номенклатуры с условием отбора в предложении «ГДЕ» – неправильный вариант

```
ВЫБРАТЬ
  ТоварыНаСкладахОстатки.Номенклатура.Наименование КАК Номенклатура,
  ТоварыНаСкладахОстатки.КоличествоОстаток
ИЗ
```

```
  РегистрНакопления.ТоварыНаСкладах.Остатки() КАК ТоварыНаСкладахОстатки
ГДЕ
  ТоварыНаСкладахОстатки.Номенклатура = &Номенклатура
```

Однако такое решение будет неоптимальным, так как сначала виртуальная таблица выберет все записи из таблицы остатков, а затем уже на полученную выборку будет наложен отбор по значению номенклатуры. Это может привести к значительному замедлению работы запроса, хотя результат будет верным.

Для оптимального выполнения подобного запроса следует ограничивать объем выбираемых данных в самой виртуальной таблице. Поэтому такой фрагмент кода следует переписать следующим образом (листинг 4.12):

Листинг 4.12. Выполнение запроса для получения остатков номенклатуры с условием отбора в параметре виртуальной таблицы – правильный вариант

```
ВЫБРАТЬ
  ТоварыНаСкладахОстатки.Номенклатура.Наименование КАК Номенклатура,
  ТоварыНаСкладахОстатки.КоличествоОстаток
ИЗ
  РегистрНакопления.ТоварыНаСкладах.Остатки( , Номенклатура = &Номенклатура) КАК
  ТоварыНаСкладахОстатки
```

При этом не следует забывать о логике выполнения запроса. Например, в случае накладывания условия отбора на виртуальные таблицы регистра сведений (*СрезПервых*, *СрезПоследних*) оба рассмотренных выше варианта (накладывание условия отбора в предложении *ГДЕ* или в параметре виртуальной таблицы) дадут разный результат, так как в этих случаях прикладной смысл выполняемого запроса меняется.

подробнее

Подробнее этот вопрос рассмотрен в разделе "Получение данных из периодических регистров сведений".

Соответствие индексов и условий запроса

Для построения оптимального плана запроса очень важно наличие подходящих индексов в таблицах – источниках запроса по полям, которые используются в условиях запроса (в предложении *ГДЕ*, в условии соединения таблиц *ПО*, в параметрах виртуальных таблиц, в предложении *ИМЕЮЩИЕ*).

подробнее

Подробные рекомендации по эффективному использованию индексов рассматривались в разделе «Индексирование таблиц».

Не использовать соединения с вложенными запросами и с виртуальными таблицами

В запросах часто требуется получить данные из нескольких источников, связанных по

определенному условию. При этом источниками запроса могут выступать не только реальные таблицы информационной базы, но и виртуальные таблицы, и вложенные запросы.

Соединения с вложенными запросами

Вложенный запрос (подзапрос) часто бывает нужен для того, чтобы отобрать по условию или каким-то образом сгруппировать данные реальной таблицы, прежде чем использовать их в соединении.

При написании таких запросов нужно помнить, что соединение с вложенным запросом может крайне замедлить выполнение запроса, сделать работу запроса нестабильной (чувствительной к используемой СУБД и аппаратуре) и вообще является потенциально опасным моментом для быстродействия системы.

Например, требуется получить остатки товаров из регистра накопления *ТоварыНаСкладах* для позиций номенклатуры, присутствующих в конкретном документе *РеализацияТоваров*, которые не являются услугами и остаток которых меньше, чем требуемое количество в документе.

Для этого нужно выбрать номенклатуру из табличной части выбранного документа, исключить номенклатуру, являющуюся услугами, и сгруппировать состав документа по номенклатуре, чтобы для каждого товара получить его суммарное количество (на случай, если один товар включен в состав документа несколько раз). Затем данные этого вложенного запроса (подзапрос выделен в листинге 4.13 жирным шрифтом) нужно связать с данными таблицы остатков регистра накопления по ссылкам товаров и отобрать записи, в которых остаток товара меньше его количества в документе.

В итоге получается довольно громоздкий и запутанный запрос (листинг 4.13).

Листинг 4.13. Выполнение запроса для получения остатков номенклатуры, содержащихся в документе, – неправильный вариант

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |         СоставДокумента.Номенклатура,
  |         СоставДокумента.Номенклатура.Представление КАК Товар,
  |         СоставДокумента.Количество КАК КоличествоВДокументе,
  |         ЕСТЬNULL(ТоварыНаСкладахОстатки.КоличествоОстаток, 0) КАК Остаток
  |ИЗ
  |         (ВЫБРАТЬ
  |                 РеализацияТоваровСостав.Номенклатура КАК Номенклатура,
  |                 СУММА(РеализацияТоваровСостав.Количество) КАК Количество
  |         ИЗ
  |                 Документ.РеализацияТоваров.Состав КАК РеализацияТоваровСостав
  |         ГДЕ
  |                 РеализацияТоваровСостав.Ссылка = &Ссылка
  |                 И РеализацияТоваровСостав.Номенклатура.Услуга = ЛОЖЬ
  |         СГРУППИРОВАТЬ ПО
  |                 РеализацияТоваровСостав.Номенклатура
  |         ) КАК СоставДокумента
```

```
  |        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки(,
  |                        Номенклатура В (ВЫБРАТЬ
  |
  Документ.РеализацияТоваров.Состав.Номенклатура
  |                                                    ИЗ
  |
  Документ.РеализацияТоваров.Состав
  |                                                    ГДЕ
  |
  Документ.РеализацияТоваров.Состав.Ссылка = &Ссылка)
  |                                                    ) КАК ТоварыНаСкладахОстатки
  |        ПО ТоварыНаСкладахОстатки.Номенклатура = СоставДокумента.Номенклатура
  |ГДЕ
  |        ЕСТЬNULL(ТоварыНаСкладахОстатки.КоличествоОстаток,0) ;

Запрос.УстановитьПараметр("Ссылка", Документ);
Результат = Запрос.Выполнить();

// Обработка результатов запроса.
```

Но мало того, что запрос запутанный, он еще и неоптимальный. В левой части соединения используется подзапрос с псевдонимом *СоставДокумента*. На самом деле не важно, в какой части соединения (правой или левой) используется подзапрос, также это плохо для любого вида соединения.

При выполнении такого запроса, независимо от используемой СУБД, весьма вероятны ошибки оптимизатора СУБД при выборе плана, что приведет к катастрофическому падению производительности запроса. В данном случае проблемой для оптимизатора является выбор правильного способа соединения двух выборок данных в запросе. Выбор алгоритма соединения зависит от того, сколько записей будет содержаться в одной и в другой выборке. В случае соединения двух физических таблиц СУБД может легко определить объем обеих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно.

Поэтому вместо вложенных запросов в соединениях ВСЕГДА нужно использовать временные таблицы. При этом их необходимо проиндексировать по полям, участвующим в условии соединения (листинг 4.14).

Листинг 4.14. Выполнение запроса для получения остатков номенклатуры, содержащихся в документе, – правильный вариант

```
МенеджерВТ = Новый МенеджерВременныхТаблиц;

Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос.Текст =
  "ВЫБРАТЬ
  |        РеализацияТоваровСостав.Номенклатура КАК Номенклатура,
  |        СУММА(РеализацияТоваровСостав.Количество) КАК Количество
  |ПОМЕСТИТЬ СоставДокумента
  |ИЗ
  |        Документ.РеализацияТоваров.Состав КАК РеализацияТоваровСостав
```

```
   |ГДЕ
   |        РеализацияТоваровСостав.Ссылка = &Ссылка
   |        И РеализацияТоваровСостав.Номенклатура.Услуга = ЛОЖЬ
   |СГРУППИРОВАТЬ ПО
   |        РеализацияТоваровСостав.Номенклатура
   |ИНДЕКСИРОВАТЬ ПО
   |        Номенклатура";

Запрос.УстановитьПараметр("Ссылка", Документ);
Результат = Запрос.Выполнить();

Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос2.Текст =
  "ВЫБРАТЬ
   |            СоставДокумента.Номенклатура,
   |            СоставДокумента.Номенклатура.Представление КАК Товар,
   |            СоставДокумента.Количество КАК КоличествоВДокументе,
   |        ЕСТЬNULL(ТоварыНаСкладахОстатки.КоличествоОстаток, 0) КАК Остаток
   |ИЗ
   |        СоставДокумента
   |        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки(,
   |                                Номенклатура В (
   |                                                        ВЫБРАТЬ
   |
  СоставДокумента.Номенклатура
   |                                                        ИЗ
   |                                                                СоставДокумента)
   |                                                                ) КАК
  ТоварыНаСкладахОстатки
   |        ПО ТоварыНаСкладахОстатки.Номенклатура = СоставДокумента.Номенклатура
   |ГДЕ
   |        ЕСТЬNULL(ТоварыНаСкладахОстатки.КоличествоОстаток,0) ;

Результат = Запрос2.Выполнить();

// Обработка результатов запроса.
```

В данном запросе вместо вложенного запроса в соединении используется временная таблица *СоставДокумента*. Таким образом, СУБД сможет легко определить размер каждой выборки, участвующей в соединении. Это позволит СУБД гарантированно выбрать самый быстрый из всех возможных планов. Переписанный подобным образом запрос будет работать одинаково хорошо и, главное, стабильно на любых СУБД, что особенно важно при разработке тиражных решений.

Кроме того, список номенклатуры из временной таблицы ограничивает объем выборки в виртуальной таблице остатков, что также делает запрос более эффективным.

И, наконец, приведенный запрос лучше читается, проще для понимания и отладки.

Соединения с виртуальными таблицами

Виртуальные таблицы регистров накопления, регистров бухгалтерии, регистров расчета разработаны специально для решения различных прикладных задач в области учета движения средств, бухгалтерского учета и сложных периодических расчетов.

При обращении к этим виртуальным таблицам платформа автоматически извлекает из регистров накопления нужные данные, группирует их по измерениям и выдает пользователю некоторую обобщенную информацию – итоговые значения ресурсов регистров в разрезе измерений. Для этого используются итоговые таблицы регистров, к которым невозможно обратиться с помощью языка запросов.

Если в запросе выбираются данные из виртуальных таблиц, то при трансляции запроса в язык SQL платформа сама обращается к итоговым таблицам регистров. В случаях, когда часть информации получается из таблицы итогов, а часть – из таблицы движений регистра, соединение с виртуальной таблицей на уровне SQL может быть реализовано как соединение с подзапросом. В этом случае оптимизатор СУБД может точно так же выбрать неоптимальный план, как при работе с подзапросом в явном виде (см. предыдущий раздел).

Соответственно, если в запросе используется соединение с виртуальной таблицей и запрос работает с неудовлетворительной производительностью, то рекомендуется вынести обращение к виртуальной таблице в отдельный запрос с сохранением результатов во временной таблице.

Таким образом, листинг 4.14 из предыдущего раздела, в котором выполнялось соединение с виртуальной таблицей остатков *РегистрНакопления.ТоварыНаСкладах.Остатки()*, можно переписать следующим образом (листинг 4.15).

Листинг 4.15. Выполнение запроса для получения остатков номенклатуры, содержащихся в документе, – правильный вариант

```
МенеджерВТ = Новый МенеджерВременныхТаблиц;

Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос.Текст =
  "ВЫБРАТЬ
  |        РеализацияТоваровСостав.Номенклатура КАК Номенклатура,
  |        СУММА(РеализацияТоваровСостав.Количество) КАК Количество
  |ПОМЕСТИТЬ СоставДокумента
  |ИЗ
  |        Документ.РеализацияТоваров.Состав КАК РеализацияТоваровСостав
  |ГДЕ
  |        РеализацияТоваровСостав.Ссылка = &Ссылка
  |        И РеализацияТоваровСостав.Номенклатура.Услуга = ЛОЖЬ
  |СГРУППИРОВАТЬ ПО
  |        РеализацияТоваровСостав.Номенклатура
  |ИНДЕКСИРОВАТЬ ПО
  |        Номенклатура";

Запрос.УстановитьПараметр("Ссылка", Документ);
Результат = Запрос.Выполнить();

Запрос2 = Новый Запрос;
Запрос2.МенеджерВременныхТаблиц = МенеджерВТ;
```

```
Запрос2.Текст =
  "ВЫБРАТЬ
  |        ТоварыНаСкладахОстатки.Номенклатура КАК Номенклатура,
  |        ТоварыНаСкладахОстатки.КоличествоОстаток КАК Остаток
  |ПОМЕСТИТЬ ОстаткиТоваров
  |ИЗ РегистрНакопления.ТоварыНаСкладах.Остатки( ,
  |                                  Номенклатура В (
  |                                                  ВЫБРАТЬ
  |
  СоставДокумента.Номенклатура
  |                                                   ИЗ
  |                                                           СоставДокумента)
  |                                               ) КАК ТоварыНаСкладахОстатки
  |ИНДЕКСИРОВАТЬ ПО
  |        Номенклатура";

Результат = Запрос2.Выполнить();

Запрос3 = Новый Запрос;
Запрос3.МенеджерВременныхТаблиц = МенеджерВТ;
Запрос3.Текст =
  "ВЫБРАТЬ
  |            СоставДокумента.Номенклатура,
  |            СоставДокумента.Номенклатура.Представление КАК Товар,
  |            СоставДокумента.Количество КАК КоличествоВДокументе,
  |            ЕСТЬNULL(ОстаткиТоваров.Остаток, 0) КАК Остаток
  |ИЗ
  |        СоставДокумента ЛЕВОЕ СОЕДИНЕНИЕ ОстаткиТоваров
  |        ПО ОстаткиТоваров.Номенклатура = СоставДокумента.Номенклатура
  |ГДЕ
  |        ЕСТЬNULL(ОстаткиТоваров.Остаток,0) ;

Результат = Запрос3.Выполнить();

// Обработка результатов запроса.
```

В данном запросе используется еще одна временная таблица *ОстаткиТоваров*, в которую помещаются данные виртуальной таблицы остатков. Обратите внимание, что объем выборки в виртуальной таблице остатков ограничивает список номенклатуры из первой временной таблицы *СоставДокумента*. Затем в третьем запросе данные этих двух временных таблиц связываются левым соединением. Данные временных таблиц доступны в этом запросе, так как все три запроса используют один менеджер временных таблиц.

Не использовать вложенные запросы в условиях соединения

По тем же причинам, по которым «вредно» использовать соединения с вложенными запросами (раздел "Соединения с вложенными запросами"), также потенциально опасно и использование вложенных запросов в условиях соединения *ПО*.

Исключить получение поля «Ссылка» через точку

Многие таблицы объектов конфигурации содержат поля, ссылающиеся на другие таблицы. Например, реквизит *Склад* документа *ПоступлениеТоваров* имеет ссылочный

тип на справочник *Склады*. Это значит, что в поле *Склад* таблицы документа содержатся ссылки на конкретные элементы справочника складов.

Поэтому совершенно излишним является получение в запросе поля *Ссылка* через точку от полей ссылочного типа. Например, требуется отобрать документы по значению ссылочного поля *Склад*. В этом случае следующий вариант запроса будет неоптимальным (листинг 4.16).

Листинг 4.16. Вывод документов «Поступление товаров» с отбором по складу – неправильный вариант

```
ВЫБРАТЬ
  Накладная.Дата,
  Накладная.Номер,
  Накладная.Контрагент
ИЗ
  Документ.ПоступлениеТоваров КАК Накладная
ГДЕ
  Накладная.Склад.Ссылка = &Склад
```

При выполнении запроса будет выполнено ненужное в данном случае соединение с таблицей справочника *Склады*. Поэтому запрос нужно переписать следующим образом (листинг 4.17).

Листинг 4.17. Вывод документов «Поступление товаров» с отбором по складу – правильный вариант

```
ВЫБРАТЬ
  Накладная.Дата,
  Накладная.Номер,
  Накладная.Контрагент
ИЗ
  Документ.ПоступлениеТоваров КАК Накладная
ГДЕ
  Накладная.Склад = &Склад
```

Этот вариант запроса оптимальнее, так как вся нужная информация уже есть в таблице документа.

Ограничить получение данных через точку от полей составного ссылочного типа

Довольно часто ссылочные поля объектов конфигурации имеют составной тип. В частности, у регистров, имеющих несколько документов-регистраторов, поле *Регистратор* будет иметь тип ссылки на таблицу каждого из документов, формирующих движения в регистре. Получение значений через точку от поля ссылочного типа (т. е. операция разыменования ссылочных полей в запросе) приводит к неявному соединению с дополнительными таблицами, на которые ссылается это поле.

Если в запросе используется получение значения через точку от поля составного ссылочного типа, то при выполнении этого запроса будет выполняться соединение со всеми таблицами объектов, входящими в этот составной тип. В результате текст запроса может сильно усложниться, и при его выполнении оптимизатор СУБД может выбрать

неоптимальный план. Это может привести к значительному падению производительности и даже к неработоспособности запроса в отдельных случаях.

Поэтому нежелательно обращаться к реквизитам регистратора регистра (например, *ТоварыНаСкладах.Регистратор.Дата*).

Например, поле *Регистратор* регистра *ТоварыНаСкладах* имеет тип ссылки на любой документ. При выполнении следующего запроса будет выполнено соединение с таблицами всех документов, имеющихся в конфигурации (листинг 4.18).

Листинг 4.18. Получение реквизитов регистратора из регистра накопления – неправильный вариант

```
ВЫБРАТЬ
  ТоварыНаСкладах.Регистратор.Номер,
  ТоварыНаСкладах.Регистратор.Дата,
  ТоварыНаСкладах.Количество,
  ТоварыНаСкладах.Поставщик
ИЗ
  РегистрНакопления.ТоварыНаСкладах КАК ТоварыНаСкладах
```

Если четко известно, что поле составного типа является ссылкой на конкретную таблицу, то настоятельно рекомендуется использовать функцию *ВЫРАЗИТЬ()* для ограничения количества таблиц в запросе.

Например, по условию задачи нужно отобрать только движения регистра *Товары на складах*, произведенные документом *Поступление товаров*. В этом случае оптимальным будет следующий вариант (листинг 4.19).

Листинг 4.19. Получение реквизитов регистратора из регистра накопления – правильный вариант

```
ВЫБРАТЬ
  ВЫРАЗИТЬ(ТоварыНаСкладах.Регистратор КАК Документ.ПоступлениеТоваров).Номер,
  ВЫРАЗИТЬ(ТоварыНаСкладах.Регистратор КАК Документ.ПоступлениеТоваров).Дата,
  ТоварыНаСкладах.Количество,
  ТоварыНаСкладах.Поставщик
ИЗ
  РегистрНакопления.ТоварыНаСкладах КАК ТоварыНаСкладах
ГДЕ
  ТоварыНаСкладах.Регистратор ССЫЛКА Документ.ПоступлениеТоваров
```

Это позволит значительно ускорить работу запроса, ограничив количество соединений при помощи функции приведения типа *ВЫРАЗИТЬ()*. В данном примере компактность и универсальность кода пожертвована в пользу его производительности. Но в результате будет выполнено только одно дополнительное соединение таблицы регистра с таблицей документом *Поступление товаров*.

Если ограничение количества соединений в запросе таким способом некорректно, то можно рекомендовать:

1. Избегать избыточности при создании полей составных ссылочных типов. Не следует без необходимости использовать типы *Любая ссылка* или *Ссылка на любой документ* и т. п. Нужно более тщательно проанализировать прикладную логику и назначить для поля ровно те возможные типы ссылок, которые необходимы для решения задачи.
2. Пожертвовать компактностью хранения данных ради производительности. Если в запросе требуется значение, полученное через ссылку, то как вариант это значение можно хранить непосредственно в самом объекте. Например, в реквизите регистра можно хранить дополнительную информацию о дате регистратора. Это приведет к дублированию информации и некоторому (незначительному) увеличению ее объема, но может существенно повысить производительность и стабильность работы запроса.
3. Если данный запрос является универсальным и используется в нескольких разных ситуациях (где типы ссылки могут быть разными), то можно формировать запрос динамически, подставляя в функцию *ВЫРАЗИТЬ* тот тип, который необходим при данных условиях. Это увеличит объем исходного кода и, возможно, сделает его менее универсальным, но может существенно повысить производительность и стабильность работы запроса.

Исключить вывод ссылочных полей в отчет

Если отчет формируется не с помощью системы компоновки данных, а с помощью встроенного языка (например, результат запроса выводится в табличный документ), то нежелательно выводить в отчет непосредственно сами ссылочные значения, так как это может существенно замедлить выполнение отчета.

Дело в том, что при выводе значения ссылочного поля для получения его представления будет выполняться дополнительный запрос к той таблице, на которую ссылается это ссылочное поле. И этот процесс будет выполняться многократно, для каждой записи отчета.

Чтобы этого избежать, следует в запросе с помощью функции *Представление()* получать текстовое представление ссылочного поля и затем уже его, а не саму ссылку, выводить в отчет или сообщение.

подробнее

О функции *Представление()* рассказано в разделе «Как получить текстовое представление ссылочного поля».

Например, в демонстрационной конфигурации «Учет движения средств», прилагающейся к книге, в обработке *Продажи номенклатуры* рассмотрен пример, представляющий продажи товаров за каждую неделю в виде кросс-отчета.

Данные для отчета получаются при выполнении следующего запроса (листинг 4.20) и выводятся затем в табличный документ.

Листинг 4.20. Запрос для получения данных

```
Запрос = Новый Запрос;
Запрос.Текст =
  "ВЫБРАТЬ
  |        ПродажиОбороты.Номенклатура КАК Товар,
  |        ПРЕДСТАВЛЕНИЕ(ПродажиОбороты.Номенклатура) КАК ТоварПредставление,
  |        ПродажиОбороты.Период КАК Период,
  |        ПродажиОбороты.КоличествоОборот КАК КоличествоОборот
  |ИЗ
  |        РегистрНакопления.Продажи.Обороты(&НачалоПериода, &КонецПериода, НЕДЕЛЯ, ) КАК
  ПродажиОбороты
  |
  |УПОРЯДОЧИТЬ ПО
  |        Товар,
  |        Период
  |АВТОУПОРЯДОЧИВАНИЕ
  |
  |ИТОГИ
  |        СУММА(КоличествоОборот)
  |ПО
  |        ОБЩИЕ,
  |        Товар,
  |        Период ПЕРИОДАМИ(НЕДЕЛЯ, &НачалоПериода, &КонецПериода)";
```

В запросе получаются как сами ссылочные значения, так и их текстовые представления. В табличный документ выводятся представления ссылочного поля *ТоварПредставление*, а значения самой ссылки *Товар* помещаются в ячейку расшифровки отчета, для того чтобы затем пользователь в отчете мог открыть запись из справочника, соответствующую этому товару