ФИРМА "1С"

Конфигурирование в системе "1С:Предприятие 8". Решение оперативных задач Версия 8.3

Методические материалы для слушателя сертифицированного курса

Февраль, 2014

ПРАВО ТИРАЖИРОВАНИЯ И РАСПРОСТРАНЕНИЯ МЕТОДИЧЕСКИХ МАТЕРИАЛОВ ПРИНАДЛЕЖИТ ФИРМЕ "1С"

Фирма "1С", Москва, 123056, а/я 64
Отдел продаж: ул. Селезневская, д.21,
телефон: (495)737-92-57,
факс: (495) 681-44-07,
e-mail: 1c@1c.ru,
URL: http://www.1c.ru

Автор материалов: ООО "1С-Учебный центр №3",
(495) 253-58-38, uc3@1c.ru, www.1c-uc3.ru

02-14

Предложения по совершенствованию методических материалов просьба направлять в группу организации обучения фирмы "1С"
e-mail: cso@1c.ru

Содержание

Соглашения о терминах, обозначениях и дополнительные соглашения

Названия диалоговых кнопок, закладок диалоговых панелей, названия пунктов меню, имена других объектов будут даваться в двойных кавычках, например, "ОК", "Услуги", "Предприятие", "Контрагент" и т.д.

Значения и типы данных будут даваться в угловых скобках: <дата>, <СправочникСсылка.Организации>

Обращение к пункту меню будет даваться в последовательном перечислении родительских пунктов через значок стрелки вправо "/", например, "Конфигурация/ Поддержка/ Обновить конфигурацию"

Практические задания делятся на практикумы (задания, выполняемые самостоятельно), упражнения (выполняемые вместе с преподавателем). Практические задания оформляются:

Практикум № _____

_____

Ссылки на процедуры и функции в основном тексте будут даваться в двойных кавычках "" без скобок

Важные дополнения к материалу даются:

Важно! _____

_____

1. Вводная часть

1.1. Введение

Настоящий курс является продолжением курса "Введение в конфигурирование в системе "1С:Предприятие 8". Основные объекты", поэтому подразумевается, что у обучаемого уже имеется определенная ясность по приемам работы в конфигураторе.

Курс рассчитан на подготовку разработчиков, которым придется впоследствии заниматься внедрением типовых решений на платформе "1С:Предприятие 8" или разработкой и автоматизацией прочих бизнес-решений.

Основная цель данного курса - получить навыки самостоятельной работы по созданию оперативных учетных и управленческих решений.

Поэтому будем придерживаться следующей технологии изучения материала каждой главы:

разбираем теоретические аспекты;

выполняем практическую работу вместе с преподавателем;

выполняем самостоятельную работу.

С прикладной точки зрения обучение будет происходить в процессе создания конфигурации, реализующей как учет основных торговых операций, так и управленческий блок планирования и контроля работы отдела активного маркетинга.

1.2. Объектная схема построения конфигураций для решения учетных и управленческих задач

В рамках предыдущего курса Вы познакомились с основными объектами, применяемыми в системе "1С:Предприятие 8". Начинающих разработчиков при этом всегда гнетет одна и та же мысль: "Хорошо, а как со всем этим работать? Когда и что применять?"

Вот поэтому, прежде чем приступать к постижению тайн организации решений оперативных задач в системе , сначала немного "осмотримся".

Итак, какие основные классы объектов нужно иметь для реализации учетных и управленческих задач?

Начнем с глубины веков. Когда появилось вообще понятие учета?

Правильно, учет появился только при появлении … ***документирования.*** Вот сохранились со времен Древнего Египта папирусы и таблички с долговыми расписками и "книгами продаж" , мы констатируем, да , был учет в Древнем Египте! А был ли раньше? Может, и был. Но раз документов не сохранилось , то предположить факт появления учета ранее … не представляется возможным.

Итак, основа любой системы учета - ДОКУМЕНТ. Потому как документ есть не что иное, как объект, предназначенный для регистрации событий, важных с точки зрения некоей предметной области.

Вот паспорт – документ? Документ! А какое событие он регистрирует? Правильно, признание вашего гражданства.

Трамвайный талон - документ? Правильно, документ, который фиксирует потенциальную оплату проезда (когда прокомпостируете , будет фиксировать уже реальную). И так далее…

Какой документ в окружающей жизни ни возьми – это объект для регистрации некоего события, да еще и запоминающий сопутствующую этому действию информацию.

Таким образом, в систему "1С:Предприятие" просто необходимо было внести класс объектов "Документ". Именно посредством объектов данного класса и будут вноситься и регистрироваться в системе произошедшие события бизнес-процессов, хозяйственной деятельности, в общем , предметной области.

Идем далее.

Есть класс объектов для регистрации входящей информации. Логично предположить необходимость класса объектов для ее обобщения, формирования и вывода в удобном для пользователя виде. Это ОТЧЕТЫ.

Можно было бы, конечно, ограничиться набором этих двух классов объектов.

Но давайте вспомним славную молодость на колхозных полях, когда колхозники помогают интеллигенции убирать урожай (те, кто этого в жизни не испытал – ну что ж, напрягите воображение).

На краю бурта обязательно сидит бригадир тетя Нюра и при высыпании очередного ведра корнеплодов в бурт рисует очередную палочку у себя в тетради.

Приезжает в конце дня председатель колхоза и спрашивает: "Тетя Нюра, сколько сегодня ведер?" И начинает тетя Нюра палочки считать в тетради…. В зависимости от ее способностей и аккуратности за время от 10 минут до получаса ответа дождаться можно. Вот это как раз пример "быстродействия" простейшей учетной схемы "Документы - Отчеты".

А вот если бы где-то еще промежуточные данные для отчета накапливать? А вот если бы, кроме записи в тетрадочке, тетя Нюра еще на дощечке после очередного высыпания исправляла мелком 67 ведер на 68, 68 на 69 и т.д., то по приезде председателя сколько времени понадобилось бы на выдачу точного отчета?

Таким образом, если ввести в систему класс объектов "Регистры" , получаем средство быстрого формирования отчетности за счет предварительного накопления и подготовки отчетных данных.

Платформа "1С:Предприятие 8" имеет в своем составе несколько классов регистров . Это:

- регистры сведений;
- регистры накопления;
- регистры бухгалтерии;
- регистры расчета.

Часть из них мы обстоятельно и подробно изучим в данном курсе. А именно в рамках данного курса будут тщательно изучены такие объекты конфигурации, как регистры накопления и регистры сведений. Сейчас бы хотелось заострить внимание только на их предназначении.

Да, создавая регистры , мы намеренно вносим в систему избыточность. Как правило, регистры получают информацию из документов, а не от пользователя напрямую. Но за счет использования регистров неимоверно возрастает простота и скорость получения отчетной информации.

Причем в системе "1С:Предприятие 8" с помощью регистров можно учитывать не только фактические, но и ПЛАНОВЫЕ операции (т.е. те, что должны произойти в будущем).

Итак, уже три класса объектов.

Перейдем к рассмотрению следующего класса.

Если дать возможность пользователям набирать наименования вручную при вводе товаров в накладные, то будьте готовы к следующей ситуации: товаровед оприходовал поступление товара "Огурцы свежие", а менеджер выписал в расходной накладной "Свежие огурцы"., И как системе понять , что это одно и то же?

Вот потому и необходимы классы объектов для ведения справочной информации.

Ситуация с огурцами легко преодолевается в том случае, если реквизит документа "Товар" имеет не строковый тип, а ссылку на элемент СПРАВОЧНИКА "Номенклатура" .Тогда и товаровед, и менеджер будут вынуждены выбирать один и тот же элемент.

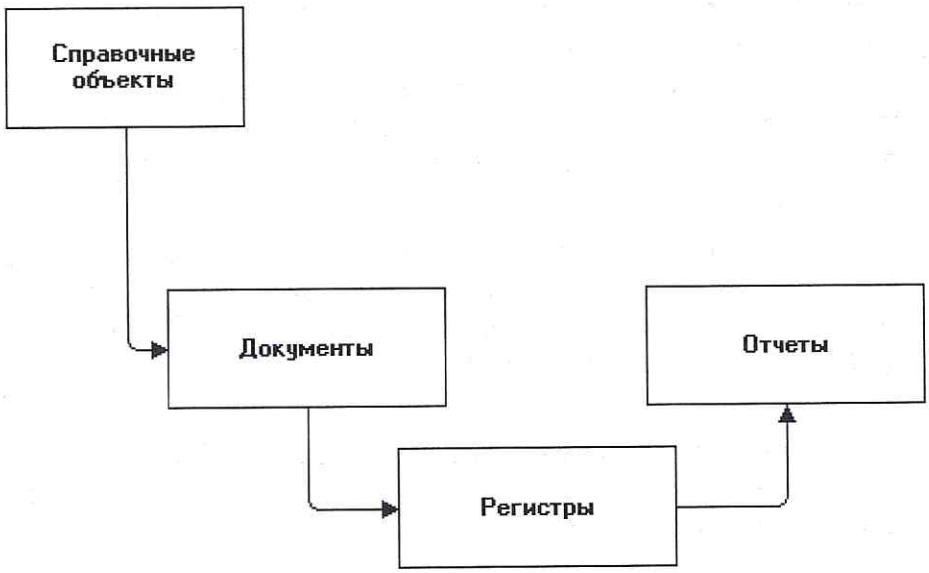


Рис. 1.1. Взаимосвязь основных классов объектов

Таким образом, классы справочных объектов служат, прежде всего, для организации правильного заполнении документов.

К подобным классам в системе "1С:Предприятие 8" можно отнести собственно сам класс объектов конфигурации "СПРАВОЧНИКИ", а также "ПЕРЕЧИСЛЕНИЯ", "КОНСТАНТЫ", "ПЛАНЫ СЧЕТОВ", "ПЛАНЫ ВИДОВ ХАРАКТЕРИСТИК", "ПЛАНЫ РАСЧЕТОВ" и т.д.

В результате мы с Вами логически вышли на ту систему классов объектов, что реализована средствами платформы "1С:Предприятие 8" (см. рис. 1.1).

1.3. Роль и место регистров

Задачи, которые Вам предстоит решать в системе "1С:Предприятие", скорее всего можно классифицировать как управленческие или учетные.

А для того чтобы утверждать, что какое-то явление полностью учтено (находится под управлением), обычно требуется доказать это.

В качестве доказательства предъявляют значения контролируемых ПОКАЗАТЕЛЕЙ.

Таким образом, мы вышли с Вами к корню автоматизации любой учетной задачи. Система, автоматизирующая учет или служащая для управления чем-либо, обязана максимально быстро, максимально точно быть готовой предоставлять информацию по тем ПОКАЗАТЕЛЯМ, состояние которых ей надлежит регистрировать.

Все, что мы с Вами делаем, имеет конечной целью выдачу информации по этим показателям, поскольку именно отчетность - это средство обеспечения существования бизнеса (регламентированная отчетность) и средство анализа (управленческая отчетность).

Давайте разберемся теперь с природой возможных показателей. Для любого явления их можно придумывать сотни и тысячи. (По законам Паркинсона любая бюрократическая система естественным образом увеличивает количество контролируемых показателей экспоненциально с течением времени вплоть до момента "удушения" дела, которым полагалось управлять.) Но все их можно классифицировать по трем группам:

1.3.1. Первая группа – "показатели остатка"

"Показатели остатка" используются для учета явлений, которые то прирастают, то убывают, но нам необходимо знать состояние их на последний (или на каждый) момент времени. Таких показателей великое множество: "количество товара на складе", "сумма денег в кармане", "число космонавтов на орбите" и так далее.

Для проверки "а не показатель ли это остатка?" можно сделать маленький тест:

"Я положил в карман за сегодняшний день 10 рублей, потом еще 5 рублей и вынул 11 рублей. Могу ли я точно сказать , сколько денег у меня в кармане?"

Думаете, 4?

Ну, мы же не в первом классе!

На самом деле, может быть и 4, и 104, и 997. Все зависит от того, что у меня со вчерашнего дня в кармане лежало!

Так вот, текущее состояние показателей остатка тесно связано с прошлым их состоянием. Это и является отличительным критерием!

Чаще всего показатели остатка используются в экономических задачах, потому как, например, рыночная экономика – есть не что иное, как оптимальный способ хозяйствования в условиях ограниченности ресурсов. Ну, а уж если априорно ресурсы ограничены , то не контролировать их состояние просто нельзя!

1.3.2. Вторая группа – "показатели оборотные"

Тот же пример: "Я положил в карман за сегодняшний день 10 рублей, потом еще 5 рублей и вынул 11 рублей".

Могу ли я точно сказать , сколько денег В ТЕЧЕНИЕ ДНЯ я положил в карман?

Могу! 15!

Показатели оборотные как раз и характеризуют движение за какой-то период. Причем, заметьте, учитываются явления, растущие только в одну сторону. И выдается информация по совокупному обороту этого явления за указанный период.

Критериями принадлежности показателя к классу "оборотный" являются:

- прежде всего, наличие периода в определении;
- независимость его состояния за текущий период от состояния за прошлый.

Наиболее часто оборотными показателями интересуются задачи статистики.

Например: "Прирост ВВП России за год", "Уменьшение человеко-жалоб за отчетный период"…

1.3.3. Третья группа - "показатели состояния"

Оба вышерассмотренных класса показателей, несмотря на разные подходы, имели одно общее свойство: это все показатели накопления, то есть события происходят одно за другим и что-то прибавляют или убавляют в состоянии того или иного показателя.

Но, кроме того, полезно выделить еще класс показателей, характеризующих состояния чего бы то ни было.

Например, такого-то числа такого-то месяца и года Вы стали пожизненным президентом такой-то страны. Вот стали, и все…

То, что Вы президент – это показатель? Ну, как минимум показатель Ваших возможностей. К какому же классу его отнести? Зависело это состояние от прошлых ваших состояний? Да вообще-то могло и не зависеть. Вот если бы Вас поточным методом и соседние страны стали президентом избирать, тогда показатель "Скольких стран президент" был бы показателем остатка. Но на это надежды мало. Хотя…

Может, это оборотный показатель? Тогда было бы так: "За отчетный квартал текущего года президентом скольких стран Вы стали?" А вот ни скольких!

Вы президент только одной страны еще с прошлого года. Вот единственная и достаточная характеристика текущего положения. Использование для нее показателей остатков слишком избыточно, а показателей оборотных – не информационно.

Таким образом, разовые явления, устанавливающие некие показатели в определенное состояние, лучше всего отражать посредством показателей состояния.

Критерием отнесения к классу показателей состояния является следующий: устанавливаемое состояние в принципе не зависит от прошлого состояния, но будет действовать не только в рамках какого-то жесткого периода, а или вечно,

или до следующей смены состояния. Причем зачастую смена состояния происходит в произвольный момент времени.

Примерами таких показателей могут быть котировки акций на бирже, курсы валют, местонахождение конкретного сотрудника (на работе, болен, в отпуске) и так далее…

1.3.4. Регистр – средство обеспечения учета показателей

Для чего мы подробнейшим образом разбирали и классифицировали все типы показателей?

Как уже выяснилось, задача любой системы учетной или управленческой – мгновенно выдавать информацию по состоянию того или иного показателя.

Мгновенно – это значит сводя к минимуму затраты времени на расчеты значения. А чтобы ничего не считать и сразу взять готовое значение , необходимо, чтобы что-то или кто-то это значение уже знал и на всякий случай держал.

Этим чем–то в системе "1С:Предприятие 8" являются регистры.

Итак, РЕГИСТРЫ - объекты, предназначенные для хранения и практически мгновенной выдачи значений тех или иных показателей в произвольных разрезах.

Для хранения показателей остатков и оборотов служат "Регистры накопления" с соответствующими видами.

Для хранения показателей состояния служат "Регистры сведений".

Кроме того, для более удобного хранения информации в интересах специфических механизмов учета есть еще "Регистры бухгалтерии" и "Регистры расчета".

Первые служат для обслуживания задач бухгалтерского учета (работа с планом счетов, с поддержкой корреспонденции или без нее).

Вторые - для реализации задач периодических расчетов, когда из периода в период по примерно одинаковым законам производят примерно однотипные расчеты, т.е. расчет зарплаты, биллинговые системы операторов сотовой связи и т.д.

В рамках данного курса с последними (бухгалтерскими и расчетными) регистрами мы знакомиться не будем. Сосредоточимся на классических задачах оперативного управления торговлей. Для них достаточно будет "Регистров накопления" и "Регистров сведений".

1.4. Постановка задачи на создание конфигурации для подразделений активных продаж

Чтобы изучение курса не свелось к восприятию голой теории, поставим практическую задачу, которую надо будет решить в процессе его прохождения.

Необходимо создать конфигурацию, предназначенную для автоматизации небольшого торгового предприятия:

учета торговых операций (покупка, продажа товара, контроль остатка товара на складе и взаиморасчетов с контрагентами);

управления отделом сервисного обслуживания. Считаем, что автоматизируемое предприятие кроме продажи товаров занимается

послепродажным сервисом. Например, кроме продажи стиральной машины можем еще установить и подключить ее силами наших сервис-менеджеров и т.п. Соответственно, система должна позволять планировать задания для сервис-менеджеров и контролировать результат их выполнения.

Кроме того в конфигурации должны присутствовать аналитические отчеты, позволяющие оценивать как выгодность торговли теми или иными товарами, так и активность работы отдела сервис-менеджеров.

2. Работа с регистрами (на примере регистра накопления остатков)

2.1. Регистр накопления остатков. Структура простейшего регистра. Измерения и ресурсы. Регистратор и Период

Регистры являются средством хранения состояния показателей остатков, оборотов или состояния и формирования текущих итогов. Именно поэтому очень важно правильно организовать регистры, а для этого необходимо знать, как они устроены и как в них заносится информация.

Для работы с регистрами накопления необходимо будет уяснить следующие понятия:

- измерения;
- ресурсы;
- реквизиты;
- регистратор;
- период записи;
- период рассчитанных итогов;
- агрегаты.

Структура регистров задается в конфигураторе (регистры - элементы конфигурации) и определяется, прежде всего, набором измерений (объектов, в разрезе которых ведется учет) и набором ресурсов (показателей и характеристик, которые требуется подсчитывать или учитывать).

Состав любого регистра должен быть таким, чтобы для каждого набора разрезов учета (измерений) можно было бы в любой момент времени однозначно определить значение любого показателя (ресурса).

Кроме измерений и ресурсов в регистре могут присутствовать реквизиты. Реквизиты позволяют хранить вспомогательную информацию о конкретных записях, содержащихся в регистре.

При проектировании структуры регистров нужно учитывать следующие обстоятельства:

- измерениями регистров могут быть объекты ссылочных или примитивных типов данных. Рекомендуется использовать именно ссылочные;
- количество измерений определяет размерность регистра и, следовательно, от него сильно зависит физический размер таблиц регистра;
- ресурсы регистров могут быть только числового типа;
- реквизиты – дополнительная информация о каждом движении (записи) регистра.

Понятие "Период рассчитанных итогов" связано с внутренним порядком хранения данных в реальных таблицах регистра. На каждый регистр накопления с хранением итогов выделяется две таблицы. Первая хранит сами записи движений (приращения). Вторая – рассчитанные итоги по этим записям.

Рассмотрим функциональность системы при включенном использовании итогов и текущих итогов регистра накопления остатков.

При добавлении очередной записи в таблицу движений система сама определяет, какие записи таблицы итогов необходимо откорректировать.

Рассмотрим на примере регистра накопления остатков, у которого есть два измерения: "Товар", "Склад" и два ресурса "Кол" и "Стоимость".

Тогда все движения хранятся в "таблице движений".

Таблица 2.1. Таблица движений регистра

Период (записи)	ВидДв	Товар	Склад	Кол	Стоимость
01.01.2014	+	Яблоко	№1	10	100
03.01.2014	+	Груша	№1	3	18
25.01.2014	-	Яблоко	№1	3	30
30.01. 2014	+	Яблоко	Главный	100	800
28.02. 2014	-	Груша	№1	3	18

Ниже мы будем подробно изучать четыре способа формирования движений. Но сейчас нам совершенно неважно , каким способом были сформированы движения.

Важно другое: система автоматически формирует записи во второй таблице. Они будут содержать информацию о состоянии итогов ресурсов по комбинациям значений измерений.

Таблица 2.2. Таблица итогов регистра

Период (итога)	Товар	Склад	Кол	Стоимость
Январь 2014	Яблоко	№1	7	70
Январь 2014	Груша	№1	3	18
Январь 2014	Яблоко	Главный	100	800
Февраль 2014	Яблоко	№1	7	70
Февраль 2014	Яблоко	Главный	100	800
01.11.3999г	Яблоко	№1	7	70
01.11.3999г	Яблоко	Главный	100	800

Обратите внимание: система каждый месяц (период хранения итогов) как бы подтверждает наличие конкретного товара на конкретном складе. Кроме того, в таблице итогов хранятся актуальные итоги. Это самые последние итоги на максимально возможное значение периода для записей регистров остатков (1 ноября 3999 года).

Если же товар с какого-то склада был продан полностью , то нет нужды хранить в таблице итогов запись со всеми нулями в значениях итогов ресурсов (груши в феврале "выпали").

При включенном использовании итогов и текущих итогов регистра накопления остатков формирование и изменение записей таблицы итогов происходит автоматически в следующих случаях:

1) при любом изменении записей в таблице движений (например, добавление новых, замена старых на новые, удаление старых) пересчитываются актуальные итоги (на 01.11.3999г);
2) при изменении записей в таблице движений – пересчитываются все промежуточные итоги, у которых период итога больше периода измененных записей движений;
3) при задействовании системных функций управления итогами (установка периода рассчитанных итогов, пересчет итогов).

Что это дает с точки зрения потребления информации из регистра? Напомним, что основная функциональность регистра связана с оперативным предоставлением информации о состоянии показателей (ресурсов).

Каким бы мы средством не запросили регистр выдать значение о состоянии остатков товаров на складах , само выполнение этой операции будет производиться следующим образом:

Если мы попросим выдать актуальные остатки , система мгновенно получит эту информацию из соответствующего фрагмента таблицы итогов. При этом совершенно неважно, сколько лет и сколько записей складывалось в таблицу движений. Готовые актуальные итоги ведь уже лежат в базе данных!

Если от системы потребуется выдать информацию по остаткам на начало 1 февраля, она опять же быстро выдаст в ответ информацию из таблицы итогов (это данные , характеризующие конец января).

Если на 15 февраля , то система сначала обратится к таблице итогов – возьмет данные на конец февраля, а потом "досчитает обратным счетом", взяв из таблицы записей информацию за 15 дней февраля.

Если на 17 марта , то система сначала обратится к таблице итогов , возьмет актуальные данные, а потом попытается взять записи в интервале между 17 марта и 01.11.3999, чтобы "досчитать обратным счетом".

В любом случае это все не в пример быстрее, нежели пытаться собирать информацию только по таблице записей!

Итак, резюмируем:

При получении итогов система всегда посчитает их правильно, но чем ближе к реальной дате граница периода рассчитанных данных в таблице итогов, тем быстрее система будет выдавать требуемые данные. Если точнее, то при получении итогов на "круглую дату" (помесячно) система берет уже готовые

итоги, а на "некруглую" дату учитывает еще движения от ближайшей "круглой даты".

Управлять положением границы периода рассчитанных итогов можно программно. Почитайте, пожалуйста, на досуге соответствующие статьи в Синтакс-помощнике, в разделе "Прикладные объекты / Регистры накопления / РегистрыНакопленияМенеджер<Имя регистра накопления> / Методы".

А можно и в пользовательской части программы.

Доступ – в пользовательском режиме через команду "Все функции" (см.рис. 2.1. Вызов команды "Все функции".)

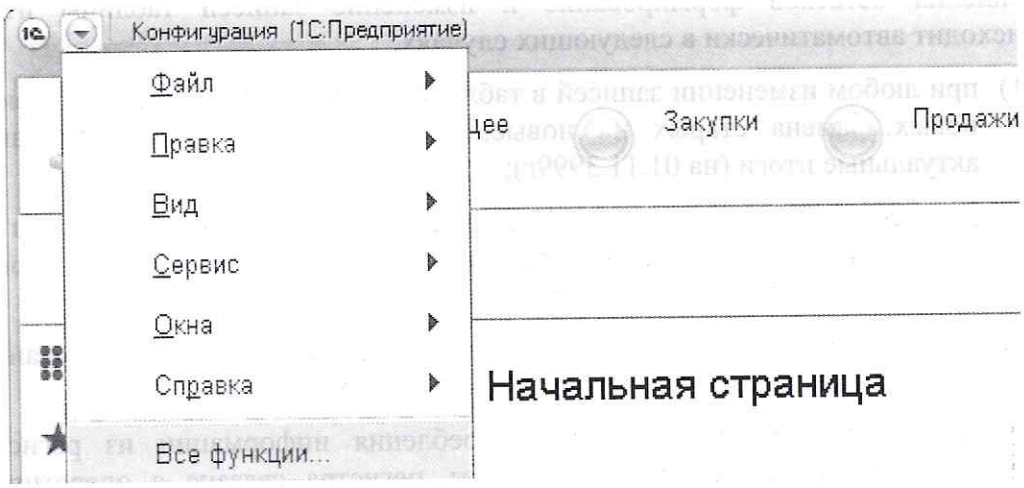


Рис. 2.1. Вызов команды "Все функции"

Далее выбираем пункт "Управление итогами" в разделе "Стандартные".

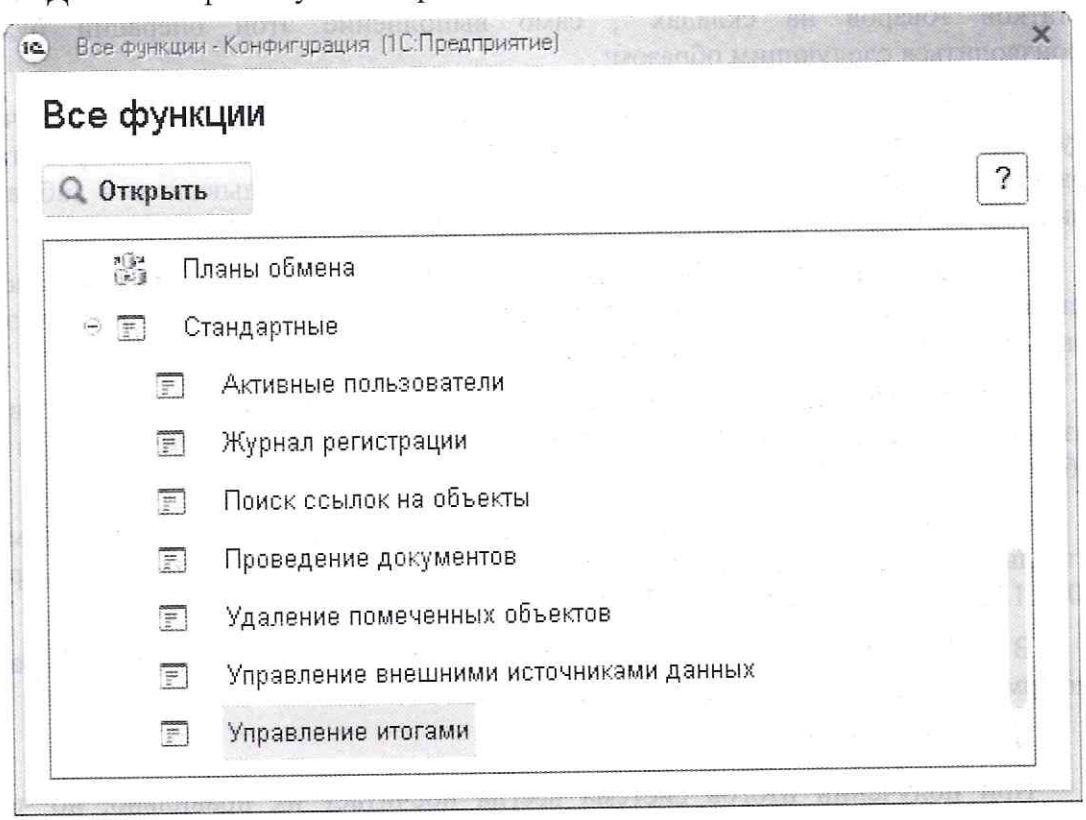


Рис. 2.2. Управление итогами

Как видите, можно управлять итогами для всех регистров сразу, а можно отдельно.

Кроме того, пользователь может управлять установкой режима разделения итогов, то есть разрешать или не разрешать системе при параллельной записи в таблицу итогов несколькими сеансами вводить отдельные записи вместо ожидания предоставления возможности корректировки занятых другими сеансами итоговых записей.

Подобная возможность повышает параллельность работы в системе (сеансам в случае конкуренции при работе с итогами регистров не надо дожидаться освобождения ресурсов), но принципиально может приводить к существенному разрастанию таблицы итогов. Ведь "сворачивание" разделенных итогов выполняется только при пересчете итогов.

Чтобы этот режим активировать , необходимо сначала в конфигурации уставить для регистра флаг "Разрешить разделение итогов", а далее уже в пользовательском режиме "Включить разделение итогов" (см.рис.2.2. Установка режима разделения итогов).

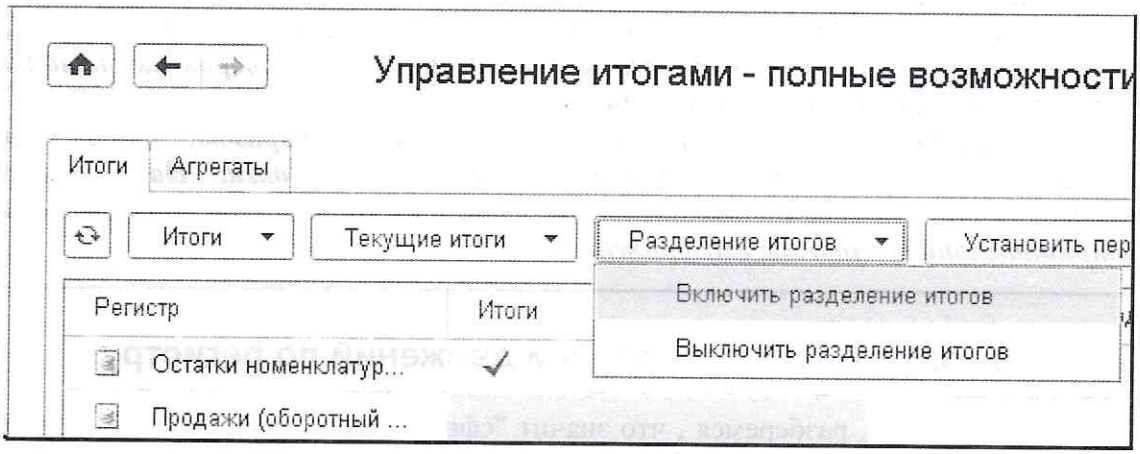


Рис. 2.3. Установка режима разделения итогов

Понятие "Агрегаты" и работу с ними мы обсудим чуть позже в разделе, описывающем практическое использование оборотных регистров.

Пока вкратце можно так сказать: агрегаты – это средство оптимизации времени доступа к данным оборотных регистров за счет заблаговременного хранения в базе данных дополнительных таблиц с различными вариантами обобщения ресурсов в разрезе комбинаций измерений и стандартных интервалов (помесячно, понедельно, подекадно, поквартально, по годам и т.п.)

Теперь разберем необходимость наличия еще двух полей в составе таблицы движений регистров.

В любой строгой системе учета недостаточно только предоставлять информацию о состоянии того или иного ресурса. Например: "Спирта на складе больше нет".

Необходимо еще иметь возможность предоставить по первому требованию проверяющих органов документальное подтверждение правомерности этого состояния: "Вот акт приемки от такого-то числа, вот приказ от такого-то за подписью директора выдать такого-то числа группе монтеров 10 литров, а вот расписка о реквизиции остатков, полученная вчера!"

Именно поэтому в регистрах накопления для корректного внесения записи (приращения ресурсов) обязательны для заполнения поля "Регистратор" и "Период".

Регистратор – это поле, содержащее ссылку на документ, изменивший состояние ресурсов регистра.

Период записи – поле, содержащее значение даты, с которой запись начинает действовать. Точнее, запись начинает действовать с момента времени. А чисто технически момент времени определяется как раз парой значений полей: "Период" и "Регистратор". Таким образом, всегда можно сравнить положение записи регистра на временной оси с любым другим моментом времени. Например, выяснить на момент времени проводимого документа, какие записи регистра уже действуют, а какие нет.

Также необходимо упомянуть еще одно свойство регистра накопления остатков – "ВидДвижения". Посредством данного свойства указывается, что необходимо сделать: прирастить ресурсы или убавить.

Практикум №1

В базе от предыдущего курса есть регистр "ОстаткиНоменклатуры", позволяющий вести учет количественно-стоимостной товаров на складах. Учтем еще и состояние задолженностей.

Создайте регистр "Задолженности" таким образом, чтобы он позволил в разрезе контрагентов учитывать их долги. На будущее договоримся, если долг положительный – это нам контрагент должен, если отрицательный – это мы ему должны.

2.2. Возможные способы записи движений по регистру

Прежде всего, разберемся , что значит "сформировать движения". А значит это очень простую вещь. Требуется сформировать записи в наборе записей регистра, куда попадут нужные данные. Чаще всего данные берутся непосредственно из документа, то есть считываются из соответствующих реквизитов. А потом набор записей регистра надо записать в сам регистр.

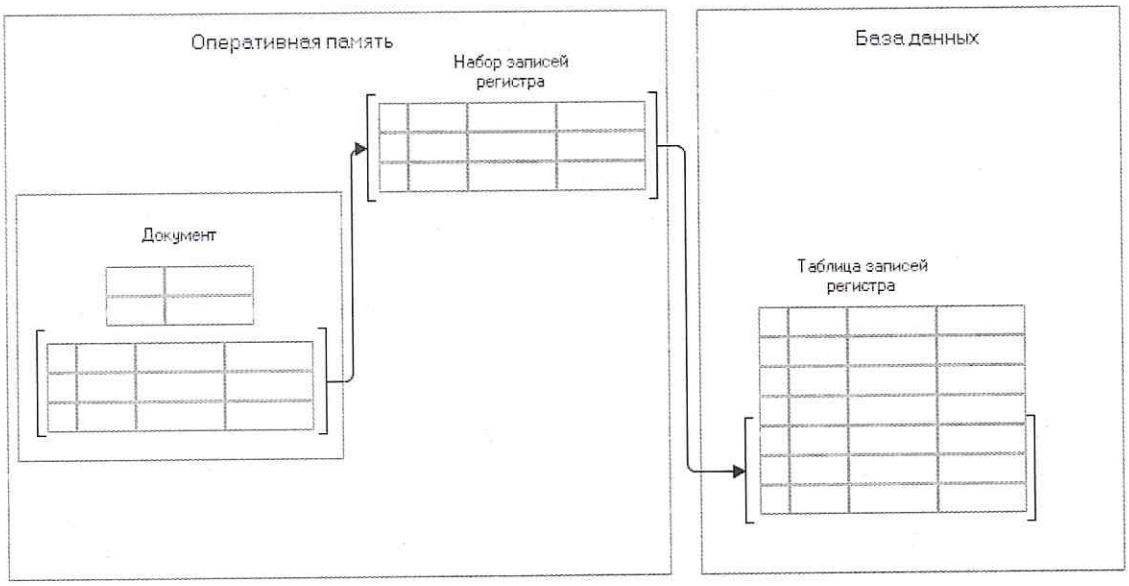


Рис. 2.4. Формирование записи в наборе записей

Всего же можно классифицировать четыре различных способа формирования движений:

2.2.1. При проведении документа

Движения можно сформировать в модуле объекта "Документ" посредством процедуры - обработчика событий "ОбработкаПроведения".

Разберем подробно этот способ на примере проведения приходной накладной. Тело процедуры сформировано конструктором движений еще в прошлом курсе. На всякий случай напомним , как именно это выполнялось.

В конфигураторе было открыто окно редактирования объекта конфигурации "Документ.ПоступлениеТоваров". В окне активизирована закладка "Движения" и нажата кнопка "Конструктор движений".

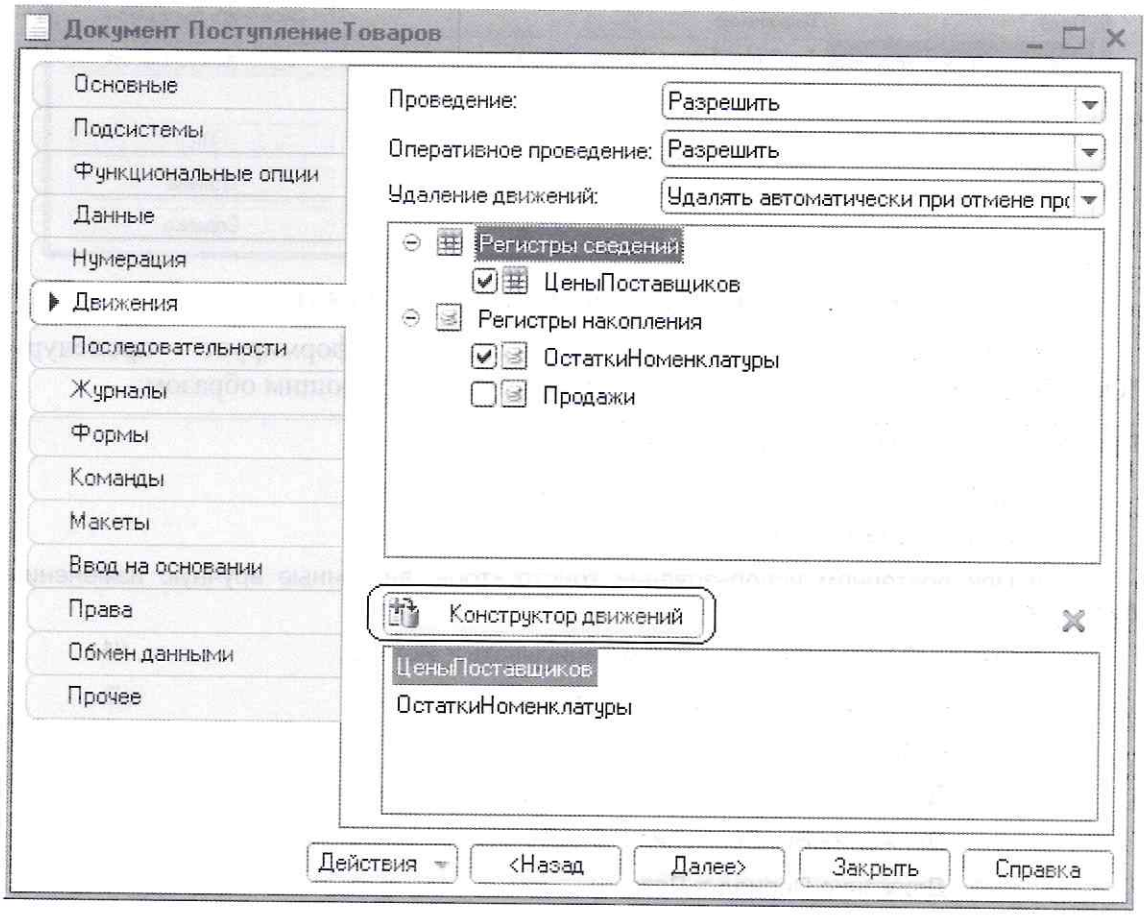


Рис. 2.5. Вызов конструктора движений

В открывшемся окне конструктора последовательно были сформированы движения для регистра "ОстаткиНоменклатуры" и "ЦеныПоставщиков".

Формирование движений в конструкторе сводится к выбору регистра в левой верхней части окна и далее к указанию, какими значениями реквизитов документа (верхняя правая часть окна конструктора) заполнять поля регистра (в нижней левой части окна конструктора).

В случае нахождения нужных значений в какой-то из табличных частей документа указание табличной части можно производить в средней части окна конструктора.

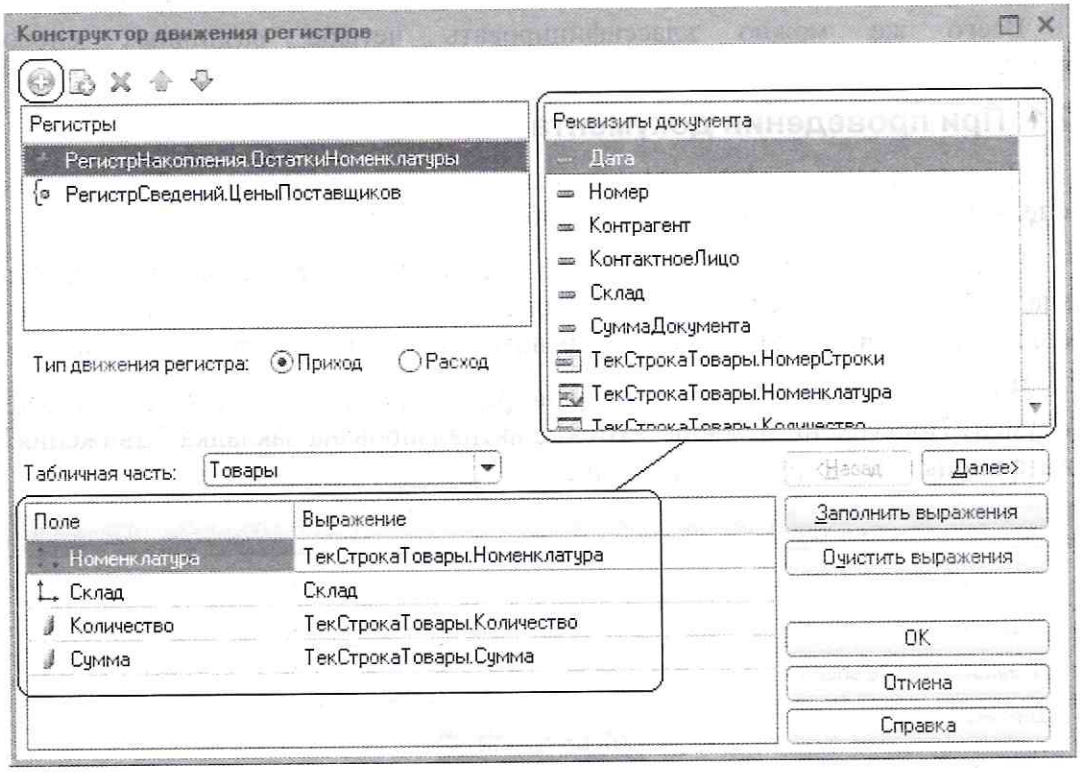


Рис. 2.6. Заполнение конструктора движений

После нажатия кнопки "ОК" система формирует процедуру "ОбработкаПроведения". Сама процедура выглядит следующим образом:

```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
	Движения.ОстаткиНоменклатуры.Записывать = Истина;	//1
	Для Каждого ТекСтрокаТовары Из Товары Цикл	//2
		// регистр ОстаткиНоменклатуры Приход
		Движение = Движения.ОстаткиНоменклатуры.Добавить();	//3
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;	//4
		Движение.Период = Дата;	//5
		Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;	//6
		Движение.Склад = Склад;
		Движение.Количество = ТекСтрокаТовары.Количество;
		Движение.Сумма = ТекСтрокаТовары.Сумма;
	КонецЦикла;
	Движения.ЦеныПоставщиков.Записывать = Истина;	//7
	Для Каждого ТекСтрокаТовары Из Товары Цикл
		// регистр ЦеныПоставщиков
		Движение = Движения.ЦеныПоставщиков.Добавить();
		Движение.Период = Дата;
		Движение.Контрагент = Контрагент;
```

```
            Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
            Движение.Цена = ТекСтрокаТовары.Цена;
        КонецЦикла;
        //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Комментарий к тексту процедуры:

//1 ДокументОбъект (а именно в нем мы сейчас находимся) имеет свойство "Движения". Данное свойство предоставляет доступ к коллекции наборов записей движений документа. Идет обращение к одному из элементов этой коллекции, а именно , к набору записей по регистру "ОстаткиНоменклатуры". Свойству "Записывать" этого набора записей устанавливается значение "Истина". Это делается для того, чтобы в конце транзакции система знала, какие именно модифицированные наборы записей из оперативной памяти надо записать в таблицы регистров.

//2 в регистре имеется измерение "Номенклатура". В документе "Номенклатура" является реквизитом табличной части "Товары". Соответственно, чтобы иметь возможность по каждому товару сформировать движение, необходимо это делать в цикле перебора строк табличной части документа.

//3 добавляется новая запись в набор записей движений данного документа по регистру "ОстаткиНоменклатуры", при этом контекст этой записи получается на промежуточной переменной ("Движение"). Впоследствии, если потребуется что-то сделать с записью , достаточно будет обратиться к этой переменной.

//4 заполнение полей новой записи начинается с "ВидаДвижения". Производится это следующим образом:

Существует системное перечисление "ВидДвиженияНакопления". У него есть только два значения "Приход" и "Расход". Нам нужен как раз "Приход". Именно это свойство и выбирается.

Существует возможность определить вид движения, как в момент создания записи, так и в процессе ее заполнения.

Вместо конструкции

```
Движение = Движения.ИмяРегистра.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
Можно было бы написать:
Движение = Движения.ИмяРегистра.ДобавитьПриход();
```

Эффект был бы тот же.

//5 указывается период действия данного движения. В данном было присвоено значение, взятое из свойства документа "Дата". (Как Вы помните, в системе "1С:Предприятие 8" дата – это не только день, но еще и часы, минуты, секунды.) Но с таким же успехом мы могли бы присвоить и любую другую дату. В "1С:Предприятии 8" период действия движения может и не совпадать с датой документа. Это позволяет, например, использовать документы "замедленного действия", результат проведения которых скажется в определенный, Вами заданный момент! Например, при решении задач планирования.

//6 заполняются поля записи регистра значениями, взятыми из реквизитов строки табличной части документа.

//7 аналогичные действия выполняются для формирования движений по следующему регистру.

Теперь давайте добавим движения документа "ПоступлениеТоваров" по регистру "Задолженности".

Для этого опять же в конфигураторе открываем окно редактирования объекта конфигурации "Документ ПоступлениеТоваров" на закладке "Движения".

Нажимаем кнопку "Конструктор движений" - и почему-то вместо окна конструктора получаем окно предупреждения об опасности данного мероприятия.

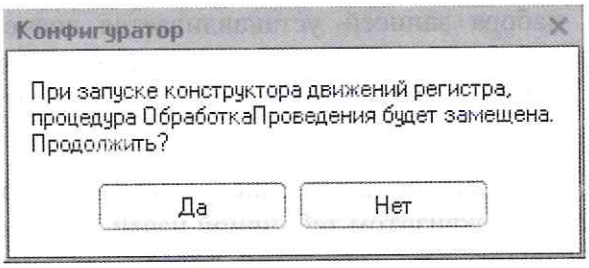


Рис. 2.6. Окно предупреждения

То есть конструктор движений перекладывает на Вас ответственность за потенциально возможные конфликты старого алгоритма обработки проведения с вносимыми Вами изменениями. Причем, хотя и достаточно жестоким, но надежным способом. Если сейчас согласимся, то система попытается интерпретировать программный код в уже существующей процедуре в поля конструктора. А после нажатия кнопки "ОК" в конструкторе произойдет замещение всей процедуры "ОбработкаПроведения" на заново сформированную конструктором.

Иногда (как в случае нашего документа) это вполне безопасно. Никакой специфичной обработки на текущий момент в процедуре не содержится. Но чаще хочется и старую процедуру сохранить, и реализовать новый фрагмент алгоритма обработки проведения уже по новому регистру.

Для реализации задуманного достаточно воспользоваться тем, что "область замещения программного кода" будет ограничиваться процедурой под названием "ОбработкаПроведения".

Если же старую процедуру переименовать, например, добавить цифру "2" к названию , то заменять система ничего не будет, а спокойно будет конструктором движений собирать текст процедуры "ОбработкаПроведения".

Итак, переименуем процедуру в модуле объекта документа "ПоступлениеТоваров":

```
Процедура ОбработкаПроведения2(Отказ, Режим)
    ...
```

Соберем конструктором движения по регистру "Задолженности". Особых сложностей подбор значений для полей регистра в данном случае не вызывает.

Единственное, на чем хотелось бы сейчас подробнее остановиться, – это вопрос выбора типа движения регистра.

Для снижения количества ошибок при выборе варианта лучше пользоваться правилом "Нулевой позиции".

Вот как это выглядит:

Представим, что наш регистр ("Задолженности") находится в "нулевой позиции". Ну что это может значить? Правильно ,это когда никто никому не должен.

И вот в этом состоянии первым документом мы вводим в систему наш документ ("ПоступлениеТоваров"), то есть отражаем событие-приход товара к нам.

Кто кому теперь за этот товар будет должен? Правильно , мы будем должны контрагенту, который нам этот товар поставил. То есть регистр должен "уйти" в ту область, где мы должны контрагенту.

Когда регистр создавали , ведь договорились о том, что, если значение суммы долга положительное , то это контрагент нам должен. Если отрицательное – это мы должны контрагенту. Таким образом, становится понятно, что наш документ должен увести регистр из "нулевой позиции" в "отрицательную область", то есть сделать движение "Расход".

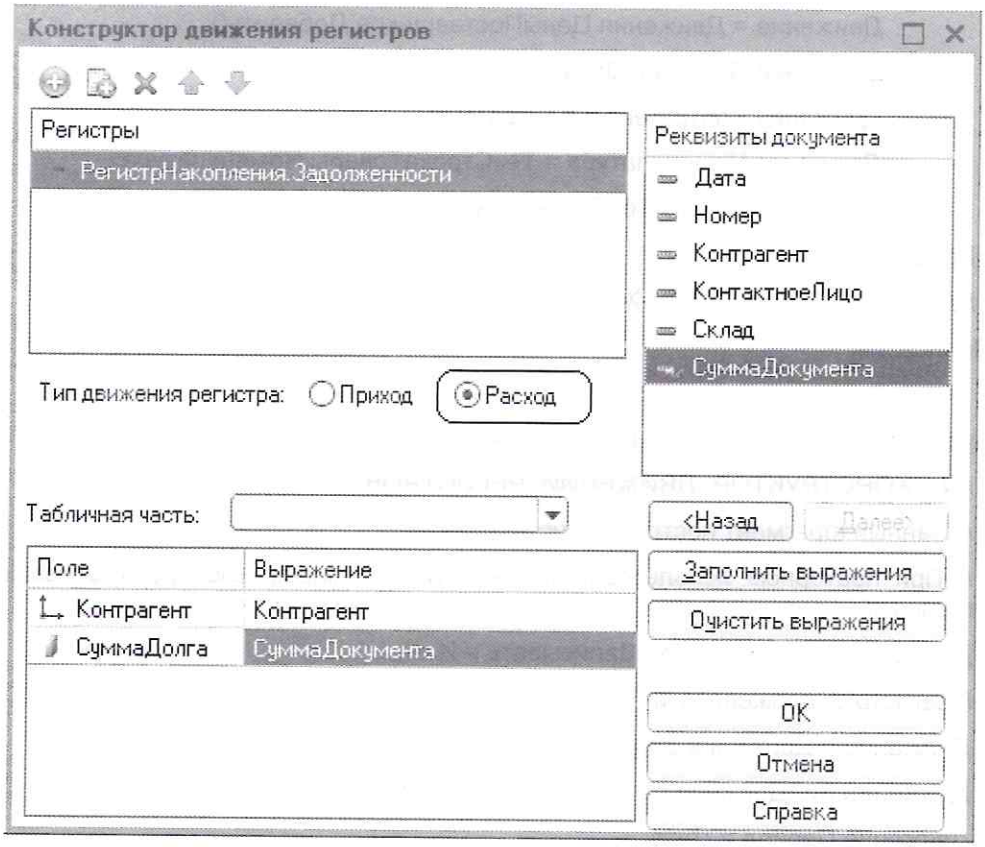


Рис. 2.7. Сборка конструктором - движений по регистру "Задолженности"

После нажатия кнопки "ОК" в конструкторе получим в модуле объекта документа две процедуры:

```
Процедура ОбработкаПроведения2(Отказ, Режим)
    //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
    // Данный фрагмент построен конструктором.
```

```
        // При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
        Движения.ОстаткиНоменклатуры.Записывать = Истина;
        Для Каждого ТекСтрокаТовары Из Товары Цикл
                // регистр ОстаткиНоменклатуры Приход
                Движение = Движения.ОстаткиНоменклатуры.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
                Движение.Склад = Склад;
                Движение.Количество = ТекСтрокаТовары.Количество;
                Движение.Сумма = ТекСтрокаТовары.Сумма;
        КонецЦикла;
        Движения.ЦеныПоставщиков.Записывать = Истина;
        Для Каждого ТекСтрокаТовары Из Товары Цикл
                // регистр ЦеныПоставщиков
                Движение = Движения.ЦеныПоставщиков.Добавить();
                Движение.Период = Дата;
                Движение.Контрагент = Контрагент;
                Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
                Движение.Цена = ТекСтрокаТовары.Цена;
        КонецЦикла;
        //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры

Процедура ОбработкаПроведения(Отказ, Режим)
        //{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
        // Данный фрагмент построен конструктором.
        // При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
        Движения.Задолженности.Записывать = Истина;
        // регистр Задолженности Расход
        Движение = Движения.Задолженности.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Контрагент = Контрагент;
        Движение.СуммаДолга = СуммаДокумента;

        //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Можно текст программного кода этих процедур теперь соединить в одну процедуру "ОбработкаПроведения". А можно, наоборот, разделить алгоритм

обработки проведения на составные части, обрабатываемые отдельными процедурами:

```
Процедура СформироватьДвиженияПоОстаткамНоменклатурыИЦенамПоставщиков()
        Движения.ОстаткиНоменклатуры.Записывать = Истина;
        Движения.ЦеныПоставщиков.Записывать = Истина;
        Для Каждого ТекСтрокаТовары Из Товары Цикл
                // регистр ОстаткиНоменклатуры Приход
                Движение = Движения.ОстаткиНоменклатуры.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
                Движение.Склад = Склад;
                Движение.Количество = ТекСтрокаТовары.Количество;
                Движение.Сумма = ТекСтрокаТовары.Сумма;
                // регистр ЦеныПоставщиков
                Движение = Движения.ЦеныПоставщиков.Добавить();
                Движение.Период = Дата;
                Движение.Контрагент = Контрагент;
                Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
                Движение.Цена = ТекСтрокаТовары.Цена;
        КонецЦикла;
КонецПроцедуры

Процедура СформироватьДвиженияПоВзаиморасчетам()
        Движения.Задолженности.Записывать = Истина;
        // регистр Задолженности Расход
        Движение = Движения.Задолженности.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Контрагент = Контрагент;
        Движение.СуммаДолга = СуммаДокумента;
КонецПроцедуры

Процедура ОбработкаПроведения(Отказ, РежимПроведения)
        СформироватьДвиженияПоОстаткамНоменклатурыИЦенамПоставщиков();
        СформироватьДвиженияПоВзаиморасчетам();
КонецПроцедуры
```

Практикум №2

Добавить в конфигурацию документы "Приходный кассовый ордер" и "Расходный кассовый ордер" (реквизиты "Контрагент" и "Сумма") и обеспечить их проведение по регистру "Задолженности". Приходный кассовый ордер обычно отражает поступление денег от контрагента. Расходный кассовый ордер – оплату денег контрагенту. Но помните, что у нас в регистре "Задолженности" хранятся не наличные деньги, а долги контрагентов!

2.2.2. Из объекта документа, но без проведения

Предыдущим способом одновременно формировались движения документа, и документ проводился. Признак проведенности документа на самом деле означает факт ПОЛНОГО завершения обработки документа пользователем. Однако что делать, если до окончания обработки еще далеко, а какие-то движения по регистрам документ уже совершить должен?

Так вот, любая процедура в модуле объекта документа может записать новые движения. Если эта процедура называется не "ОбработкаПроведения" (и не вызывается из нее), тогда создастся ситуация, при которой документ не проведен, но движения у него есть.

Чтобы оценить прикладное значение этой возможности, добавим в документ "ПродажаТоваров" процедуру предварительного бронирования товара.

С прикладной точки зрения это важно в случаях одновременной работы с "большими" документами. Пока пользователь заполняет такой документ, его коллеги могут "перехватить" товар, и при проведении документа может вдруг выясниться, что отпустить половину номенклатурных позиций уже невозможно. Они отданы другим покупателям.

Именно поэтому для некоторых предприятий полезно еще на этапе заполнения документа каким-то образом бронировать товар ("захватывать" самому), причем в указанных в документе количествах.

Лучший способ запоминать подобную информацию реализуется посредством соответствующего регистра.

Итак, приступаем:

создаем регистр "БронированиеТоваров", определяем у него измерения "Номенклатура" и "Склад", один ресурс "Количество".

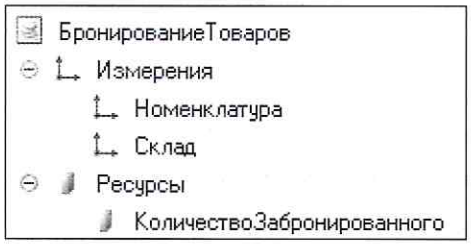


Рис. 2.9. Состав регистра "БронированиеТоваров"

Разрешим документу "ПродажаТоваров" выполнять движения по данному регистру. И, кроме того, убедимся что свойство "Удаление движений" стоит в положении "Удалять автоматически при отмене проведения" (см. рис. 2.10).

Это обеспечит автоматическое удаление движений по всем регистрам, имеющим включение в коллекцию движений нашего документа при отмене проведения или при сохранении документа с пометкой удаления.

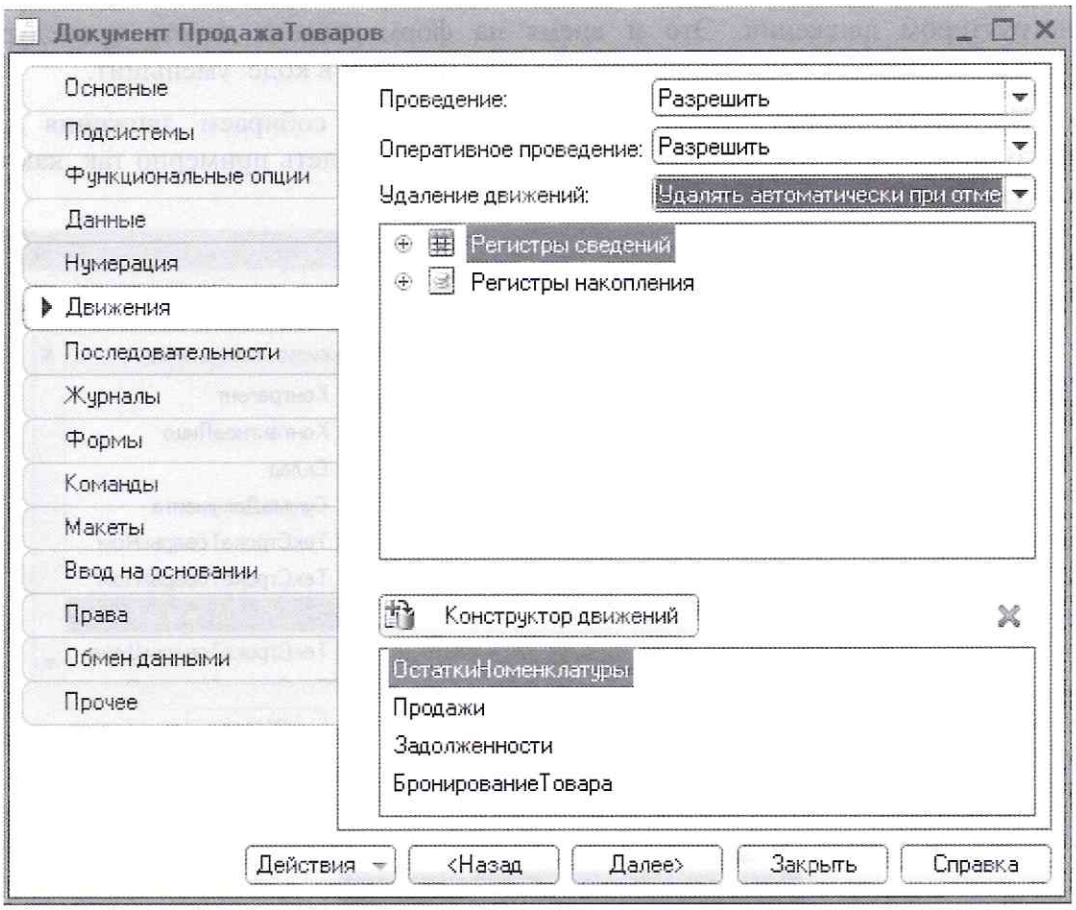


Рис. 2.10. Движения документа "ПродажаТоваров"

Теперь необходимо будет на форме документа дать возможность пользователю нажимать некую кнопку в любой момент, когда пользователь захочет введенные в табличную часть документа товары забронировать.

При нажатии на эту кнопку надо будет программно выполнять движения по регистру "БронированиеТоваров".

Ну а впоследствии мы собираемся все механизмы, завязанные на проверку свободных для отгрузки товарных позиций на складе, обязать учитывать следующий факт: если в некий момент времени в регистре "ОстаткиНоменклатуры" десять холодильников, но в регистре "БронированиеТоваров" два , то это означает, что свободно можно распоряжаться только восемью холодильниками.

Технологию сборки самого программного механизма бронирования товаров из формы документа можно применять любую.

Применим один из вариантов , достаточно красивый с точки зрения скорости разработки и минимизации количества ошибок.

Идея заключается вот в чем: "Как бы мы не разрабатывали сам алгоритм, апофеозом его действия будет формирование наборов записей для регистра". А кто лучше всего справляется с формированием кода для этой, рутинной на самом деле, операции?

Правильно, конструктор движений.

Итак, сначала текст процедуры в модуле документа, которая посредством свойства документа "Движения" запишет в регистр новые движения , соберем конструктором движений. Это и время на формирование текста процедуры сократит, и потенциальное количество ошибок (описок) в коде уменьшит.

Нажимаем кнопку "Конструктор движений" и собираем движения для нашего нового регистра. Скорее всего, это будет выглядеть примерно так, как на рис.2.11.

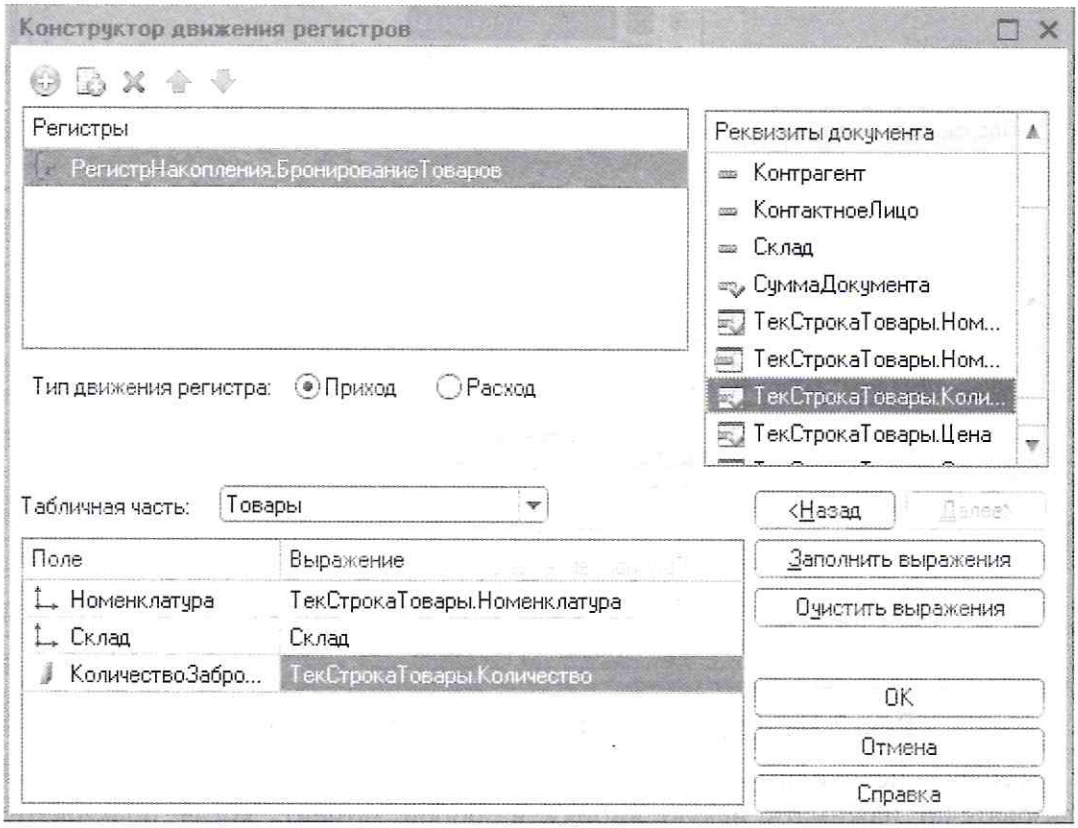


Рис. 2.11. Сборка процедуры конструктором

Полученный текст процедуры необходимо переработать.

Во-первых, как минимум его надо переименовать, иначе наша процедура будет являться одним из обработчиков события "Запись документа с проведением", а в самом начале мы говорили, что хотим иметь возможность формировать движения из объекта документа, но без проведения.

Во-вторых, надо обеспечить выполнение записи в регистр уже в самой процедуре (ведь теперь у нас не будет транзакции "запись документа с проведением", в рамках которой система в конце заботливо запишет все модифицированные в оперативной памяти наборы записей на жесткий диск, в регистр).

В-третьих, надо обеспечить запись самого документа в базу данных в актуальном состоянии ПЕРЕД записью данных в регистр.

Реализация этих доработок может выглядеть следующим образом:

```
Процедура БронированиеТоваров() экспорт
    Записать();                                                    //1
    Для Каждого ТекСтрокаТовары Из Товары Цикл                     //2
        // регистр БронированиеТоваров Приход
        Движение = Движения.БронированиеТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Период = Дата;
        Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
        Движение.Склад = Склад;
        Движение.КоличествоЗабронированного = ТекСтрокаТовары.Количество;
    КонецЦикла;
    Движения.БронированиеТоваров.Записать();                       //3
КонецПроцедуры
```

Прокомментируем текст процедуры:

//1 При записи движений по регистру накопления обязательно заполнение в каждой записи поля "Регистратор". Заполнение поля будет производиться автоматически, но, если документ еще не записан, с этим возникнут проблемы. Поэтому первым делом записываем сам документ.

//2 технология формирования движения одинакова во всех случаях. Мы ее только что разбирали на примере проведения документа "ПоступлениеТоваров". Поэтому только коротко повторим основные этапы: Цикл перебора строк табличной части (чтобы добраться до каждого товара и его количества). Внутри него, посредством свойства "Движения" объекта документа, получаем возможность добавить записи по регистру "БронированиеТоваров". Формируем очередную запись, заполняя необходимые атрибуты.

//3 записываем сформированный набор записей регистра "БронированиеТоваров".

Для чего в заголовке процедуры слово "Экспорт", помните? Совершенно верно, в предыдущем курсе разбирали, как именно получается возможность вызывать процедуры модуля объекта из другого модуля.

Теперь отработаем вызов процедуры из формы документа. Мы хотим, чтобы кнопка бронирования была доступна в форме документа. Не в формах списка документов "ПродажаТоваров", не в журналах, а только в форме самого документа.

Для этого открываем форму документа "ПродажаТоваров".

В окне редактирования формы документа в правом верхнем углу активизируем закладку "Команды", далее на закладке "Команды формы" нажимаем кнопку "Добавить". В открывшемся окне свойств добавляемой команды пишем имя команды "Забронировать", потом нажимаем "кнопочку с лупой" в поле свойства "Действие".

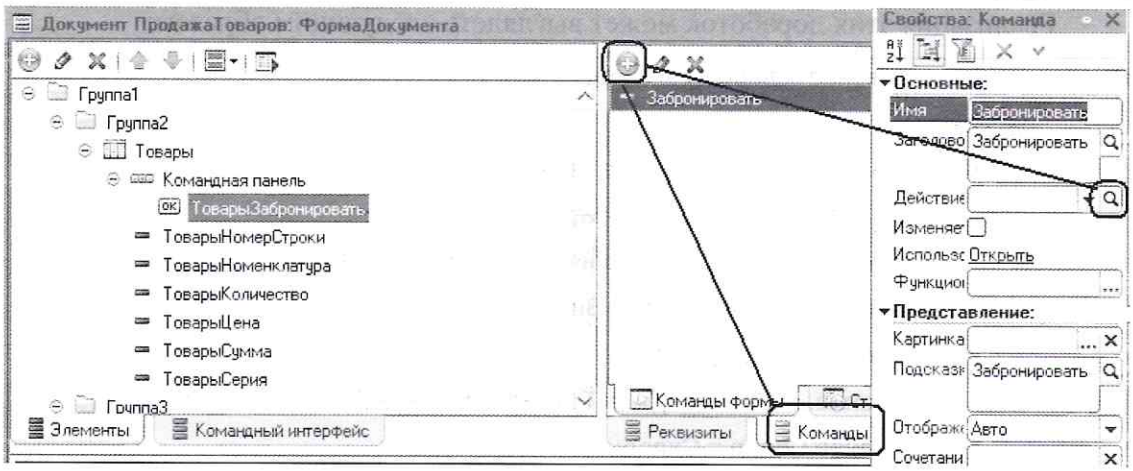


Рис. 2.12. Создание новой команды формы

В результате система создаст обработчик события нажатия кнопки – процедуру "Забронировать".

```
&НаКлиенте
Процедура Забронировать(Команда)
        // Вставить содержимое обработчика.
КонецПроцедуры
```

Однако сейчас надо вспомнить о том, что управляемая форма существует одновременно на клиенте и на сервере. И для каждой процедуры в модуле формы должна быть указана директива компиляции – где именно эта процедура должна исполняться. Обработчик события нажатия кнопки на форме может исполняться только на стороне клиента (иначе управляемая форма на стороне клиента его просто не найдет). Но желаемая нами процедура "БронированиеТоваров" располагается мало того что на сервере, так еще и в объекте нашего документа. Поэтому передача управления на желаемую процедуру должна выполняться:

а) в процедуре модуля формы, исполняемой на сервере;

б) при обращении к контексту объекта документа, который надо еще получить из данных формы.

Собираем такую процедуру в модуле формы документа и вызываем ее из процедуры "Забронировать":

```
&НаКлиенте
Процедура Забронировать(Команда)
        ВыполнитьБронированиеНаСервере();
КонецПроцедуры

&НаСервере
Процедура ВыполнитьБронированиеНаСервере()
        ОбъектДокумента =
          ДанныеФормыВЗначение(Объект,Тип("ДокументОбъект.ПродажаТоваров")); //1
        ОбъектДокумента.БронированиеТоваров();                              //2
        ЗначениеВДанныеФормы(ОбъектДокумента,Объект);                       //3
КонецПроцедуры
```

Комментарий к тексту процедуры:

//1 преобразуем данные формы в объект документа;

//2 вызываем желаемую процедуру из контекста объекта документа;

//3 поскольку процедура "БронированиеТоваров", кроме всего прочего, записывала документ в базу данных, то теперь надо преобразовать объект документа опять в данные формы. Таким образом, данные текущей формы документа будут приведены в соответствие с данными самого документа в базе данных, то есть экземпляры отличаться не будут.

Остается последний аккорд – размещение вызова команды "Забронировать" на форме документа.

Для этого вернемся в форму документа и мышкой перетащим вызов команды "Забронировать" в область элементов командной панели, обслуживающей табличное поле "Товары".

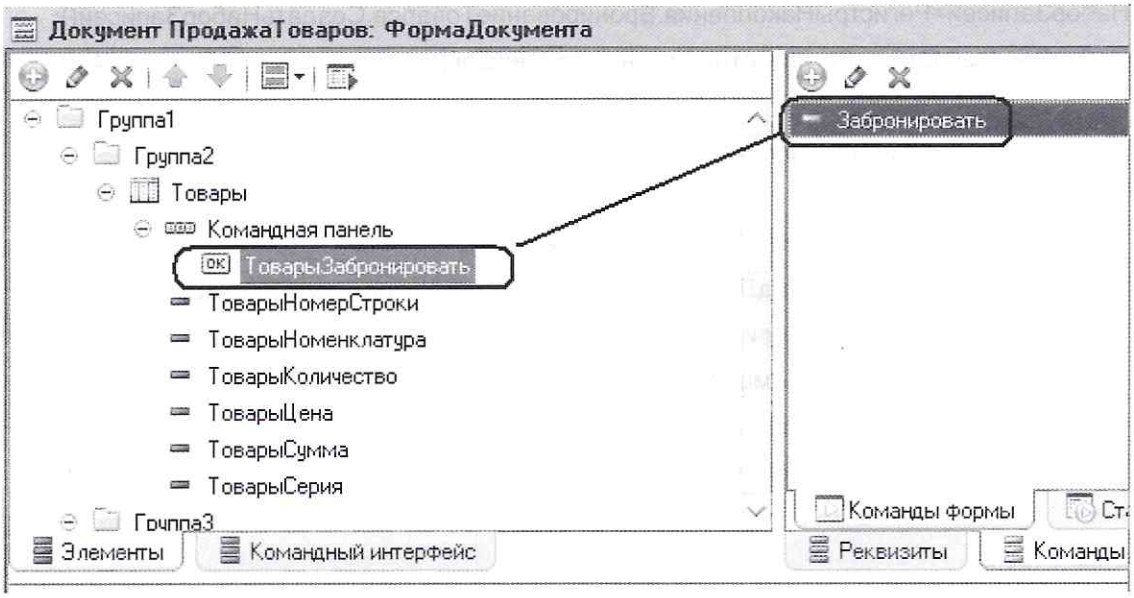


Рис. 2.13. Размещение новой команды в интерфейсе формы

Итак, функциональность выполнения бронирования из формы документа нами реализована.

Сохранив все изменения в конфигурации базы данных, и запустив пользовательский режим , в этом можно убедиться.

2.2.3. Извне объекта документа

Любой объект, содержащий процедуру работы с набором записей регистра, может приписать любому возможному регистратору данного регистра любые движения.

Для демонстрации этой возможности вынесем механизм бронирования товаров из модуля объекта документа.

С прикладной точки зрения теперь задача будет звучать так: "Пользователь, просматривая любой список документов продажи, должен иметь возможность забронировать товары для тех документов, которые он выберет. Причем, если документ уже проведен (т.е. товар продан), больше для него бронировать ничего не надо".

Начнем решение данной задачи опять с реализации самого программного механизма. Местоположение исполнительной процедуры практически ничем, кроме здравого смысла, не ограничивается. Но если исходить из того, что сами программные действия будут касаться именно документов "ПродажаТоваров" ,то вполне подходящим видится использование для размещения процедуры модуля менеджера документа "ПродажаТоваров".

Размещаем в нем следующую процедуру:

```
Процедура ВыполнитьБронирование(ТекДок) Экспорт                    //1

Если ТекДок.Проведен Тогда                                         //2
    Возврат;
КонецЕсли;
                                                                   //3
НаборЗаписей=РегистрыНакопления.БронированиеТоваров.СоздатьНаборЗаписей();
НаборЗаписей.Отбор.Регистратор.Установить(ТекДок);                 //4

Для каждого ТекСтрокаТовары Из ТекДок.Товары Цикл

        Движение=НаборЗаписей.Добавить();                          //5
        Движение.ВидДвижения=ВидДвиженияНакопления.Приход;
        Движение.Период=ТекДок.Дата;
        Движение.Номенклатура=ТекСтрокаТовары.Номенклатура;
        Движение.Склад=ТекДок.Склад;
        Движение.КоличествоЗабронированного=ТекСтрокаТовары.Количество;

КонецЦикла;

НаборЗаписей.Записать();                                           //6

КонецПроцедуры
```

Комментарий к тексту процедуры:

//1 подразумеваем, что при вызове процедуры в качестве параметра будем получать ссылку на документ, для которого надо сформировать движения;

//2 поскольку бронировать есть смысл только для непроведенных документов - проведенные отсекаем;

//3 создаем набор записей по регистру "БронированиеТоваров";

//4 указываем, что движения будут касаться только конкретного регистратора (нашего документа), для чего устанавливаем отбор по регистратору. Если этого не сделать , ничего не получится при данном способе формирования движений;

//5 формируем в цикле перебора строк документа очередную запись набора;

//6 записываем все сформированные записи в регистр.

Обратите внимание: нижняя часть процедуры мало отличается от аналогичных фрагментов наполнения наборов записей регистров в предыдущих двух технологиях формирования движений (при проведении документа и из объекта документа). Но все же отличия есть. Они касаются того, что в текущем случае мы находимся вне документа. Например, приходится "объяснять", где именно можно найти табличную часть "Товары".

Но самым главным отличием является программное создание набора записей с установкой отбора по регистратору. Ранее эта функциональность реализовывалась обращением к элементу коллекции "Движения" объекта документа. Да, было меньше кода. Но зато сейчас нам необязательно обращаться к объекту. Как следствие -мы не напрягаем ресурсы системы необходимостью загружать объект из базы данных. А это означает потенциальный выигрыш в скорости исполнения разрабатываемого программного механизма.

Теперь определимся , как будем осуществлять вызов нашей процедуры.

Поскольку при постановке задачи звучала фраза , что данный механизм может вызываться из любой формы документов продажи , то достаточно уместным будет решение с использованием объекта <Команда>, параметризованным ссылкой на документ "ПродажаТоваров".

Хотя в общем случае размещение такой команды возможно в любом разделе метаданных, но уместнее будет создать ее в рамках раздела команд самого документа "ПродажаТоваров". Добавляем в этом разделе новую команду. Называем ее "ЗабронироватьСнаружи".

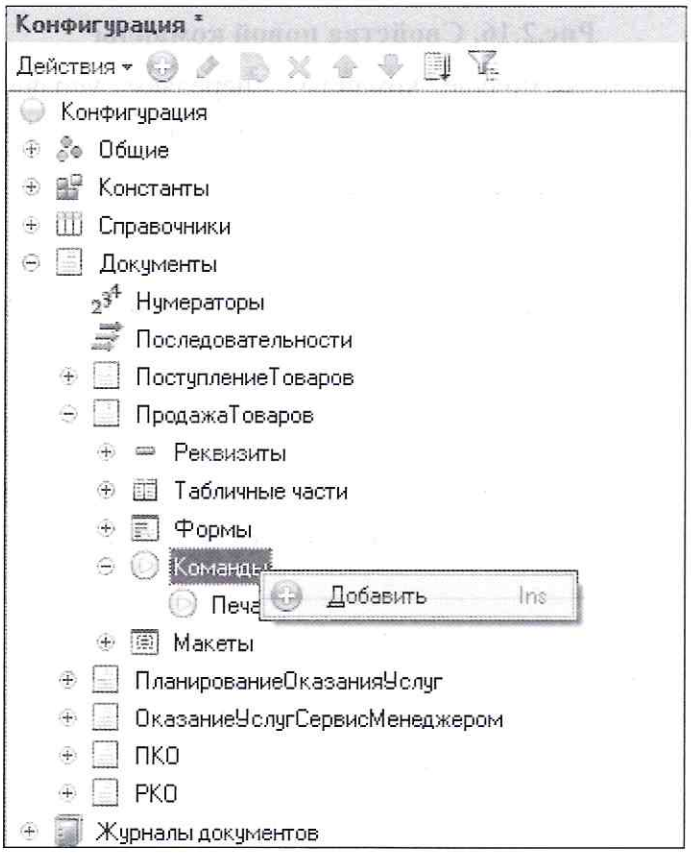


Рис. 2.15. Добавление новой команды

В свойствах новой команды

- указываем название: "ЗабронироватьСнаружи";
- указываем группу, в которую команда будет входить по умолчанию: <Командная панель формы. Важное>;
- устанавливаем тип параметра команды - <ДокументСсылка.ПродажаТоваров >.

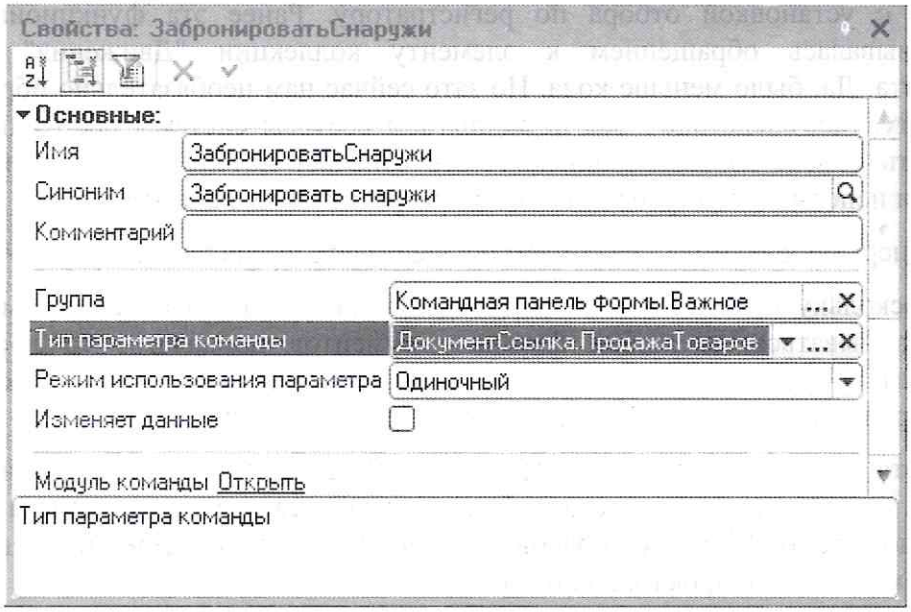


Рис.2.16. Свойства новой команды

А в открывшемся модуле команды передаем управление процедуре "ВыполнитьБронирование" через вызов промежуточной процедуры, исполняемой на стороне сервера.

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)
        ВызватьВыполнениеБронированияНаСервере(ПараметрКоманды);
КонецПроцедуры

&НаСервере
Процедура ВызватьВыполнениеБронированияНаСервере(ССылкаНаДокумент)
        Документы.ПродажаТоваров.ВыполнитьБронирование(ССылкаНаДокумент);
КонецПроцедуры
```

Подводя итог под описанием решения нашей задачи, хотелось бы еще раз отметить, что основой программного формирования движений по любому регистру является выполнение следующих действий:

1) Создание набора записей регистра.

2) Установка необходимых отборов на этот набор записей.

3) Добавление строк к набору записей и заполнение этих строк (а иногда речь может идти об одной только строке).

4) Запись набора записей в базу данных.

А уж что из этого набора будет делать программист при помощи кода, а что, обращаясь к средствам того или иного контекста ,определяется ситуацией и месторасположением самого алгоритма. В случае формирования движений "извне" все действия мы выполнили именно за счет написания кода.

2.2.4. Интерактивное внесение данных в регистр (ручная операция)

Достаточно редко, но бывают ситуации, когда логику установки или изменения значений показателей в регистре по каким-то причинам сложно, не нужно или невозможно описать заранее, на этапе разработки конфигурации. Но при этом необходимо дать возможность пользователю все же данные в регистре изменять.

В таких ситуациях обычно применяют интерактивный способ внесения информации в регистры.

С простейшим случаем интерактивного внесения информации в независимый регистр Вы сталкивались еще на начальном курсе. Свидетельством тому – наличие в нашей учебной базе регистра сведений "КурсыВалют".

Давайте для полноты картины еще отработаем ситуацию интерактивного ввода данных в зависимый регистр – в регистр накопления остатков "БронированиеТоваров".

Реализуем посредством специального документа (регистратора), которому будет запрещено делать движения программным способом. Но при этом предоставим интерактивные возможности непосредственного создания записей в регистре.

Итак, добавим в конфигурацию документ "ИнтерактивноеБронированиеТоваров".

Отметим принадлежность нашего документа к подсистеме <Общие / Журналы>:

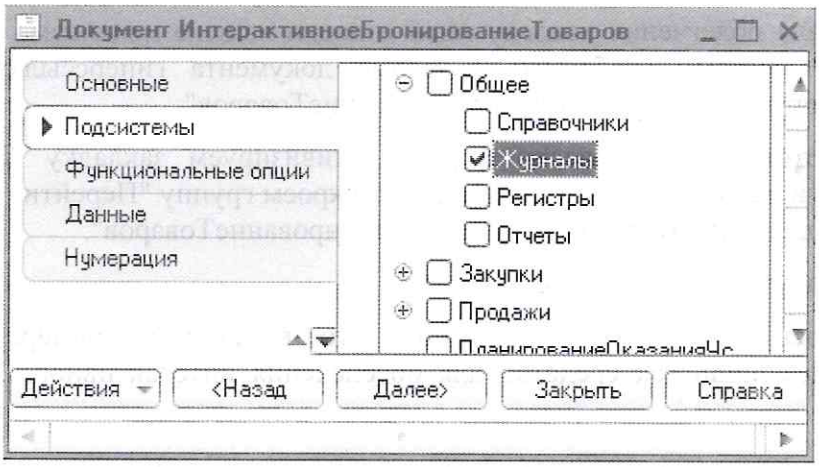


Рис. 2.17. Документ "ИнтерактивноеБронированиеТоваров"

Никаких реквизитов и табличных частей документ содержать не должен. Более того, у него должно быть запрещено проведение!

Документ будет служить только для ручного формирования движений по регистру "БронированиеТоваров"

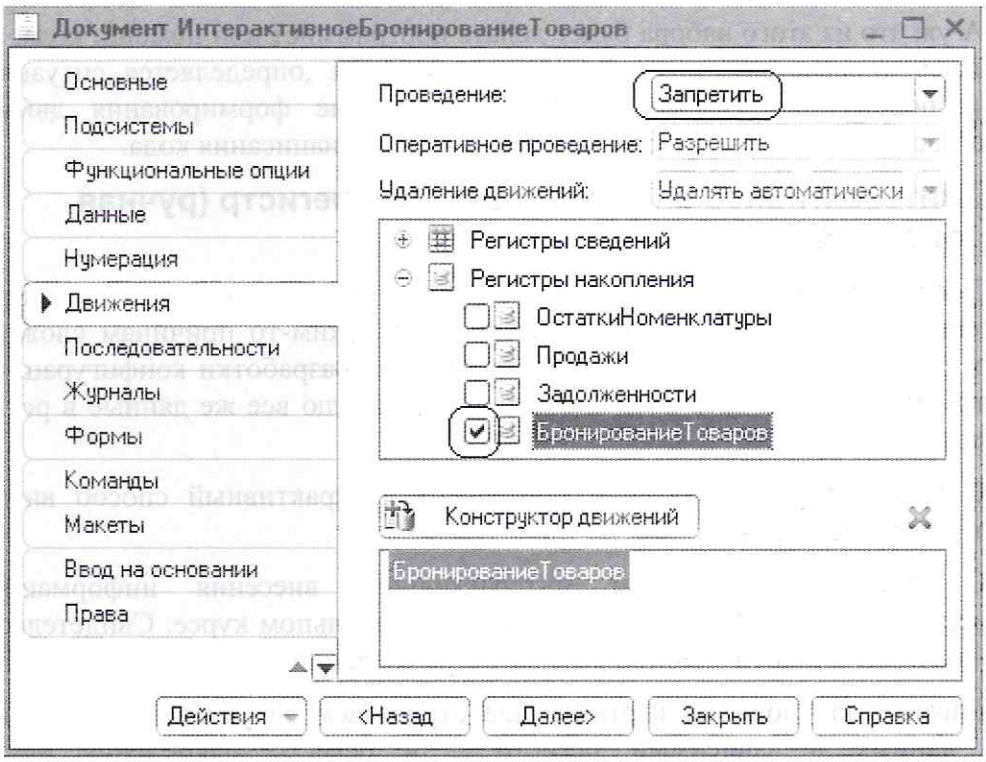


Рис. 2.18. Движения документа "ИнтерактивноеБронированиеТоваров"

Далее построим конструктором форму нашего документа.

В открывшемся окне редактирования формы в правом верхнем углу (в окне реквизитов формы) раскроем основной реквизит формы, потом раскроем коллекцию "Движения". Мышкой "схватим" элемент коллекции "БронированиеТоваров" и перетащим в окно элементов формы (левая верхняя часть окна редактирования формы).

На вопрос системы "Добавить колонки таблицы "Объект.Движения.БронированиеТоваров?" лучше ответить утвердительно.

Ну и для облегчения последующей отладки нашего механизма не забудем "попросить" систему отобразить в форме документа гиперссылку на окно движений документа по регистру "БронированиеТоваров".

Для этого в левом верхнем углу активизируем закладку "Командный интерфейс", в разделе "Панель навигации" раскроем группу "Перейти" и поставим флажок "Видимость" напротив регистра "БронированиеТоваров".

Все, документом можно пользоваться.

Хотелось бы обратить внимание, что в данном примере документ использовался лишь как средство для обеспечения интерактивного заполнения регистра.

Кроме того, вся технология не содержала ни строчки кода , все и исполнялось интерактивно.

Но!

Тогда пользователю предоставляется полная свобода действий, иногда даже во вред самому пользователю.

Например, можно ввести с привязкой к некому документу записи с указанием абсолютно разных периодов.

То есть, на временной оси документ "ИнтерактивноеБронированиеТоваров" будет занимать одно положение, а его движения – совершенно иные. Это может привести к проблемам с надежностью ведения учета.

Поэтому для обеспечения корректности использования системы как раз способ "интерактивного формирования движений" может потребовать от разработчика самого кропотливого кодирования, сопровождающего любой шаг пользователя.

Но возможности для этого есть.

Например, для реализации вышеописанной проблемы с "выравниванием" даты документа "ИнтерактивноеБронированиеТоваров" и периодов сформированных для него движений можно воспользоваться обработчиком события "ПередЗаписью" даже не документа, а модуля набора записей регистра "БронированиеТоваров":

```
Процедура ПередЗаписью(Отказ, Замещение)
    // определение значения документа-регистратора
    Документ = ЭтотОбъект.Отбор.Регистратор.Значение;

    //обход записей набора записей
    Для каждого ТекЗапись Из ЭтотОбъект Цикл
        ТекЗапись.Период = Документ.Дата;
    КонецЦикла;
КонецПроцедуры
```

Теперь независимо от того, каким способом формировался набор записей, система перед его записью будет определять дату документа-регистратора и записывать это значение в поле "Период" каждой из записей.

Кроме того, необходимо позаботиться о еще одном моменте. Дело в том, что наши документы "ПоступлениеТоваров" и "ПродажаТоваров" используют механизм "Оперативного проведения". Особенности этого механизма будут подробнее рассмотрены в нашем курсе ниже, в разделе "Оперативное и неоперативное проведение". Но уже сейчас, поскольку документ "ИнтерактивноеБронированиеТоваров" активно вмешивается в жизнь регистра "БронированиеТоваров", а данные самого регистра впоследствии будут использоваться при оперативном проведении документа "ПродажаТоваров" , важно не дать нашему документу испортить работу механизма.

А как он может испортить? Да очень просто. Поскольку документ отключен от механизма проведения вообще , то системе сейчас абсолютно все равно, какой датой пользователь выписывает "ИнтерактивноеБронированиеТоваров". И если сегодня документ будет выписан датой следующего года , то вот это уже как раз может повредить оперативному проведению документа "ПродажаТоваров", в частности, механизму контроля отрицательных остатков на складе, который опять же мы будем разрабатывать далее по курсу.

Итак, уже сейчас нам необходимо запретить записывать документы "ИнтерактивноеБронированиеТоваров" с датой, превосходящей нашу текущую дату.

Для этого открываем модуль объекта документа "ИнтерактивноеБронированиеТоваров". Далее нажимаем кнопку вызова "Процедуры и функции" на командной панели "Модуль" (см. рисунок 2.20. Формирование обработчика события "ПередЗаписью" в модуле документа) и двойным кликом выбираем обработчик "ПередЗаписью".

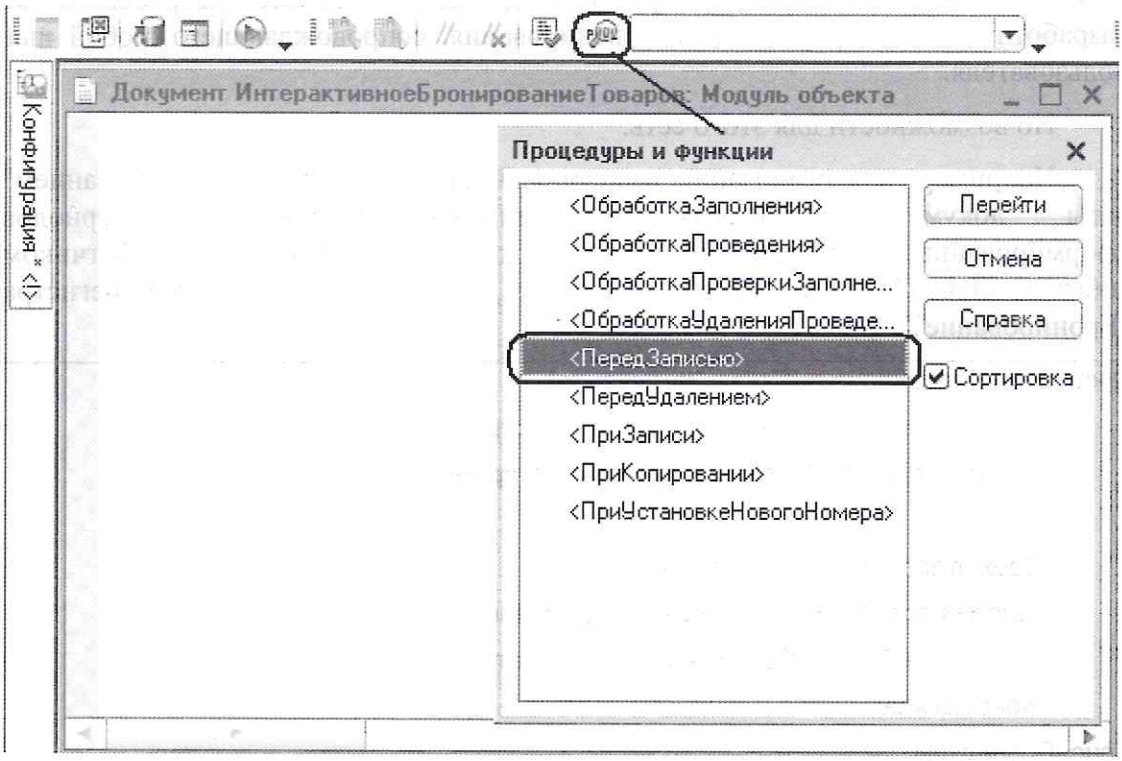


Рис. 2.20. Формирование обработчика события "ПередЗаписью" в модуле документа

Текст процедуры делаем таким:

```
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)
 Если НачалоДня( Дата) > НачалоДня(ПолучитьОперативнуюОтметкуВремени()) Тогда
                Отказ = Истина;
                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст = "Нельзя вводить документ будущей датой! ";
                Сообщение.Сообщить();
 КонецЕсли;
КонецПроцедуры
```

Комментарий к тексту процедуры:

Для проверки нужного условия выполняется превышение дня даты документа над днем даты оперативной отметки времени.

Если условие выполнилось , присваиваем управляющему параметру "Отказ" значение истина (т.е. теперь не состоится то событие, обработчиком которого является процедура) и выдаем соответствующее сообщение.

Практикум №3

Создайте документ "Ручная коррекция задолженности" с возможностью формирования "ручных" записей по регистру "Задолженности".

2.3. Возможные способы получения данных из регистра остатков

2.3.1. Использование объектной модели системы "1С:Предприятие" ("РегистрНакопленияМенеджер")

Одним из способов получения данных из регистров является способ обращения с использованием объектной модели получения данных системы "1С:Предприятие".

В частности, чтобы получить накопившиеся итоги в регистре остатков, достаточно воспользоваться методом ***"Остатки"*** объекта "РегистрНакопленияМенеджер".

Найдем описание данного метода в Синтакс-помощнике (Раздел "Прикладные объекты/ Регистры накопления/ РегистрНакопленияМенеджер<Имя регистра накопления>/ Методы/ Остатки").

При чтении статьи особое внимание уделите работе с параметрами.

Результат выполнения метода - таблица значений, заполненная итогами и содержащая колонки с измерениями, указанными в параметре "Измерения", и колонки с ресурсами, указанными в параметре "Ресурсы" (табл.2.3).

Таблица 2.3. Результат применения метода "Остаток"

Номенклатура	Склад	Количество	Сумма
Паркер "Golg"	Основной	40	3 800.00
Kohinor тм	Основной	60	360.00

Разберемся с данной технологией на следующем примере.

Необходимо дать возможность пользователю контролировать в старых расходных накладных (документах "ПродажаТоваров"): "Какие остатки были на складе на тот момент, когда документ выписывался?"

Для этого покажем на форме документа остатки для товаров, попавших в табличную часть, причем актуальные на момент самого документа.

Прежде всего, для этого добавим новый реквизит формы "Ост", тип <Строка> длиной символов в 30.

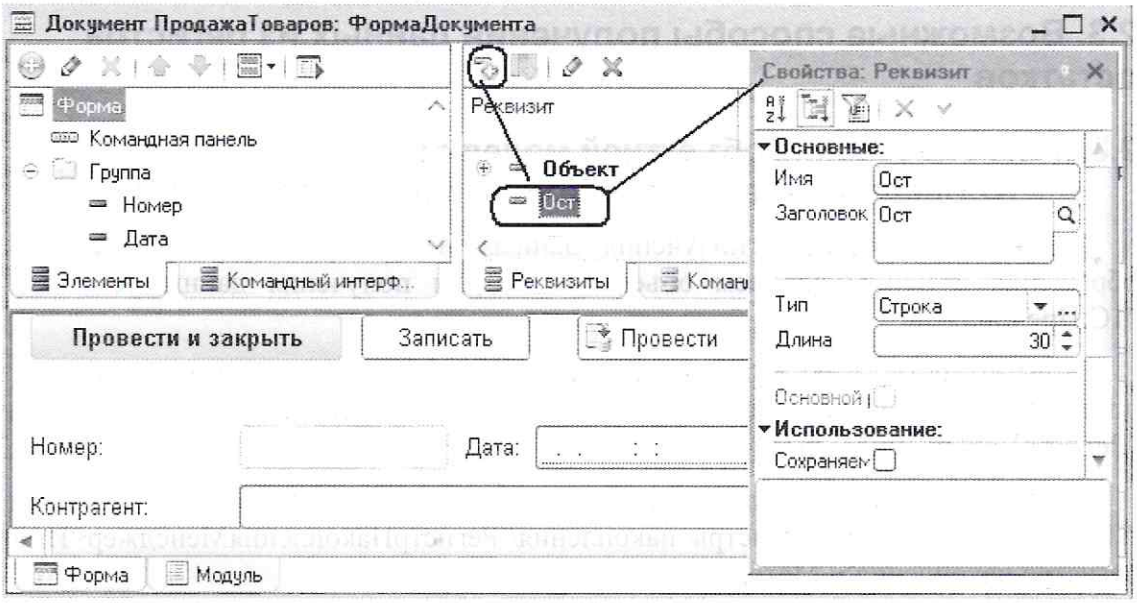


Рис. 2.21. Добавление нового реквизита формы

Далее перетащим наш новый реквизит формы в область элементов формы.

Теперь определимся с заполнением его данных. Как и любое другое программное действие , оно должно быть "ответом" на некое событие. Очевидно, что событие должно быть как-то связано с табличным полем "Товары" формы документа.

Находим в перечне событий данного табличного поля "ПриАктивизацииСтроки" и формируем для него процедуру-обработчик события.

Поскольку обработчик события на форме может выполняться только на клиентской части формы, а данные, нужные нам , могут получаться только из серверной части, то опять применяем вызов промежуточной функции, которая исполняется на стороне сервера.

```
&НаКлиенте
Процедура ТоварыПриАктивизацииСтроки(Элемент)
        Стр = Элементы.Товары.ТекущиеДанные;                    //1
        Если Стр<>Неопределено Тогда                            //2
                Ост = ОпределениеОстаткаТовара(Стр.Номенклатура,
                        Объект.Склад, Объект.Дата,Объект.Ссылка);  //3
        КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Функция ОпределениеОстаткаТовара(ТекТовар,Склад,Дата,Ссылка )
        ОстаткиТов=РегистрыНакопления.ОстаткиНоменклатуры;      //4

        ФильтрТов=Новый Структура;                              //5
        ФильтрТов.Вставить("Номенклатура",ТекТовар);
        ФильтрТов.Вставить("Склад",Склад);
```

```
        Если ЗначениеЗаполнено(Ссылка) Тогда                                    //6
                ВременнойМомент=Новый МоментВремени(Дата,Ссылка)
        иначе
                ВременнойМомент=Неопределено;
        КонецЕсли;

        ТаблицаОст=ОстаткиТов.Остатки(ВременнойМомент,ФильтрТов,"Номенклатура",
"Количество");                                                                  //7

        Возврат ТаблицаОст.Итог("Количество");                                  //8

КонецФункции
```

Комментарий к тексту процедуры "ТоварыПриАктивизацииСтроки":

//1 для того что бы знать, по какому товару считать остатки , выясняем, а из какой строки табличного поля надо этот товар брать;

//2 все это имеет смысл делать, только если текущая строка есть в табличном поле;

//3 само получение остатка будет реализовано в функции, исполняемой на серверной части. В качестве параметра передаем ссылку на товар из текущей активной строки табличного поля и другие необходимые данные.

Комментарий к тексту функции "ОпределениеОстаткаТовара":

//4 создаем переменную, дающую впоследствии доступ к менеджеру нужного нам регистра;

//5 готовим структуру, содержащую фильтр по нужному товару на нужном складе;

//6 определяемся с временным моментом, на который надо получить остатки. Если документ уже записывался в базу данных , значит, у него есть и дата, и ссылка, из которых можно собрать момент времени этого документа. Если же документ новый (а это будет видно по тому, что ссылка у объекта пока пуста) , то вполне уместно будет определять актуальные (самые последние) остатки из регистра;

//7 получаем в таблицу значений остатки нужного товара, актуальные на нужный момент времени;

//8 возвращаем в качестве результата функции итог по колонке "Количество" полученной выше таблицы значений.

Практикум №4

Позаботимся о следующем сервисе для пользователя:

Допустим, он начинает выписывать Приходный кассовый ордер (ПКО) для конкретного контрагента. А программа сразу подсказывает: сколько этот контрагент нам должен. Поэтому:

При выборе контрагента в документе "ПКО" создать процедуру, которая проверит задолженность данной организации перед нами. И в

случае, если та положительная, пропишет эту задолженность в поле "Сумма".

При выборе "Контрагента" в документе "РКО" создать процедуру, которая проверит задолженность данного контрагента перед нами. И в случае, если та отрицательная (то есть мы должны), пропишет эту задолженность в поле "Сумма".

Разработанные нами примеры все же не так часто реализуются на практике посредством объектной модели чтения. Дело в том, что пока нам было нужно получать итоги с отборами типа "равно определенному значению", объектная модель вполне позволяет получить данные за одно обращение к базе данных.

Но если задачу слегка изменить, например:

нужно показать остатки по нашему складу и по фирме в целом;

нужно получить остатки по всем товарам, попавшим в табличную часть документа;

нужно получить свободный остаток по товару, т.е. вычесть данные остатка товара из регистра "БронированиеТоваров" из остатка товара в регистре "ОстаткиНоменклатуры".

То все эти примеры посредством объектной модели не могут быть реализованы эффективно, т.е. за одно обращение к базе данных!

Важно!

Есть четыре основных причины применения объектной модели обращения к базе данных:

1. Когда прикладной объект надо записать в базу данных.
2. Когда прикладной объект надо модифицировать в базе данных (т.е. прочитать в оперативную память, изменить и опять записать в базу данных).
3. Когда в алгоритме нужно использовать механизм динамического чтения данных больших массивов прикладных объектов (суть механизма – чтение данных из базы данных порционно , блоками по небольшому количеству записей) , можно использовать объект манипулирования данными типа <СправочникВыборка.<ИмяСправочника>>, <ДокументВыборка.<ИмяДокумента>>, <РегистрНакопленияВыборка.<ИмяРегистра>> и т.п.
4. Когда данные уже лежат в оперативной памяти и надо к ним просто обратиться.

В остальных случаях применение запроса для решения задачи или одинаково эффективно с объектной моделью (наш пример, разобранный в этом разделе) или гораздо эффективнее, чем то, что можно сделать объектной моделью.

Причем последних случаев подавляющее большинство!

2.3.2. Использование табличной модели системы "1С:Предприятие" ("Запрос")

Самым быстрым по производительности способом получения данных в системе "1С:Предприятие" является работа с запросом. Поэтому чаще всего Вам придется пользоваться именно этой техникой.

Работа с запросами была разобрана в предыдущем курсе "Введение в конфигурирование...".

Для решения же прикладных задач с использованием запросов важно приучить себя предварять работу в программе составлением схемы запроса.

Важно!

Составление схемы запроса обычно выполняется по следующей технологии:

1. нарисовать эскиз выходной таблицы запроса с вариантом заполнения данных;
2. обозначить на эскизе количество выходных полей;
3. обозначить для каждого поля таблицу-источник (по возможности консолидируя что?):
4. обозначить дополнительные действия в запросе.

К дополнительным действиям чаще всего относятся:

- фильтрация, отбор (если речь идет об отборе по значению измерения виртуальной таблицы, то надо делать отбор в параметрах самой таблицы, в остальных случаях отборы чаще всего обеспечиваются операндом запроса ГДЕ);
- свертка таблицы (СГРУППИРОВАТЬ ПО);
- разворачивание таблицы за счет добавления строк с промежуточными итогами (ИТОГИ ПО);
- сортировка (УПОРЯДОЧИТЬ ПО)

Теперь сосредоточимся только на особенностях, которые присущи запросам по регистрам остатков.

Самым важным является то, что кроме реальной таблицы записей регистра остатков, запрос может манипулировать еще и тремя виртуальными таблицами: "Таблицей остатков", "Таблицей оборотов" и "Таблицей остатков и оборотов".

2.3.2.1 Таблица остатков

Для первого плотного знакомства с "Таблицей остатков" построим отчет "ОстаткиНоменклатуры", в котором надо увидеть, на каких складах сколько товаров и на какую сумму.

Составляем схему запроса.

Согласно постановке задачи, нам понадобится таблица, составленная из четырех выходных полей, все поля берутся из одного источника - <РегистрНакопления.ОстаткиНоменклатуры.Остатки>. Никаких дополнительных действий не требуется (см. рис.2.24).

1	2	3	4
Номенклатура	**Склад**	**Количество**	**Сумма**
Паркер "Golg"	Основной	40	3 800.00
Kohinor тм	Основной	60	360.00

РегистрНакопления.ОстаткиНоменклатуры.Остатки

Рис. 2.24. Схема запроса для отчета "ОстаткиНоменклатуры"

Когда схема готова, можно приступать к реализации.

Добавляем к конфигурации новый отчет "Остатки Номенклатуры".

В открывшемся окне редактирования отчета нажимаем кнопку "Открыть схему компоновки данных". Нажимаем потому, что формирование отчетов посредством схемы компоновки данных (СКД) на текущий момент самый технологичный способ формирования отчетов. Кроме получения отчета "один в один" с исходной схемой запроса он позволяет на базе этого же отчета получить великое множество модификаций, настроек и проч., не прибегая к написанию кода.

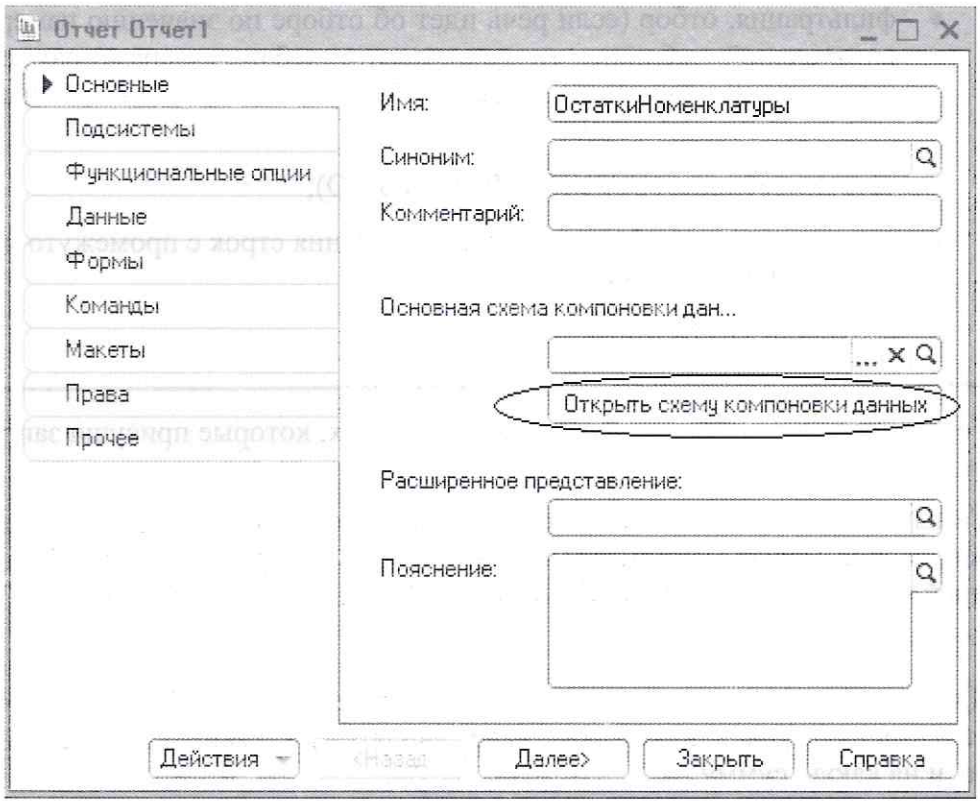


Рис. 2.25. Создание отчета "ОстаткиНоменклатуры"

После нажатия кнопки "Открыть схему компоновки данных" система откроет конструктор макета схемы компоновки данных. Там можно просто нажать кнопку "Готово".

Далее откроется окно основной схемы компоновки данных нашего отчета. В нем добавляем новый набор данных – запрос.

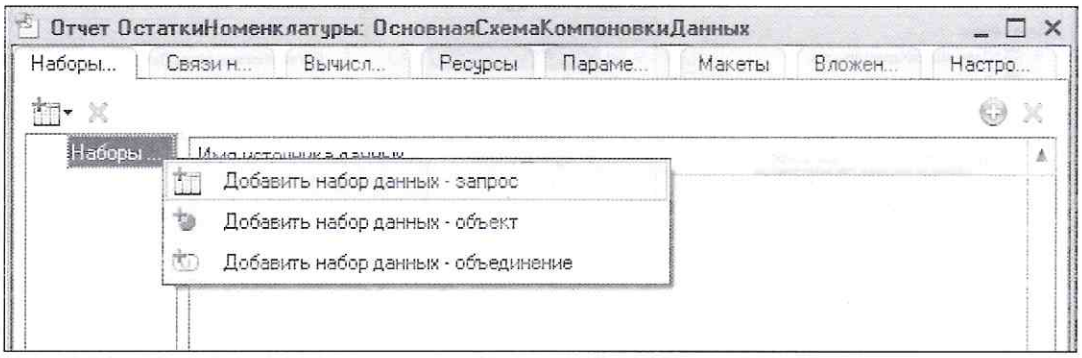


Рис. 2.26. Добавление нового набора данных - запрос

Сам запрос оптимально создать, пользуясь конструктором запроса:

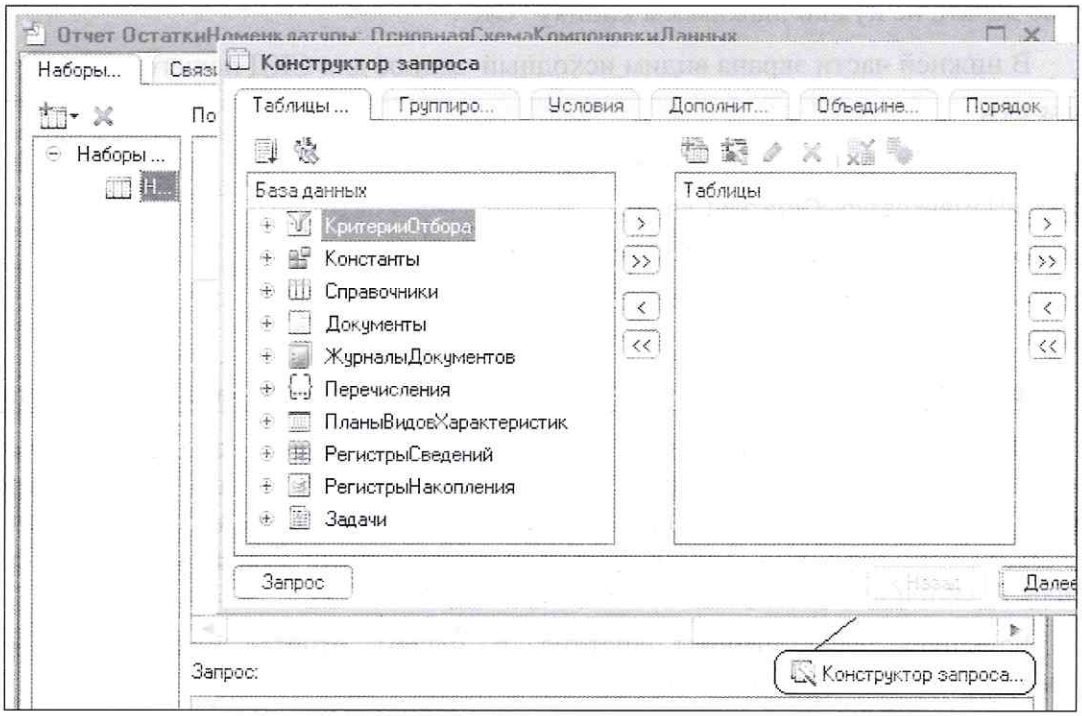


Рис. 2.27. Вызов конструктора запроса

Согласно схеме запроса на основной закладке конструктора "Таблицы и поля" указываем таблицу-источник "ОстаткиНоменклатуры.Остатки".

Далее из источника выбираем выходные поля тоже согласно схеме запроса.

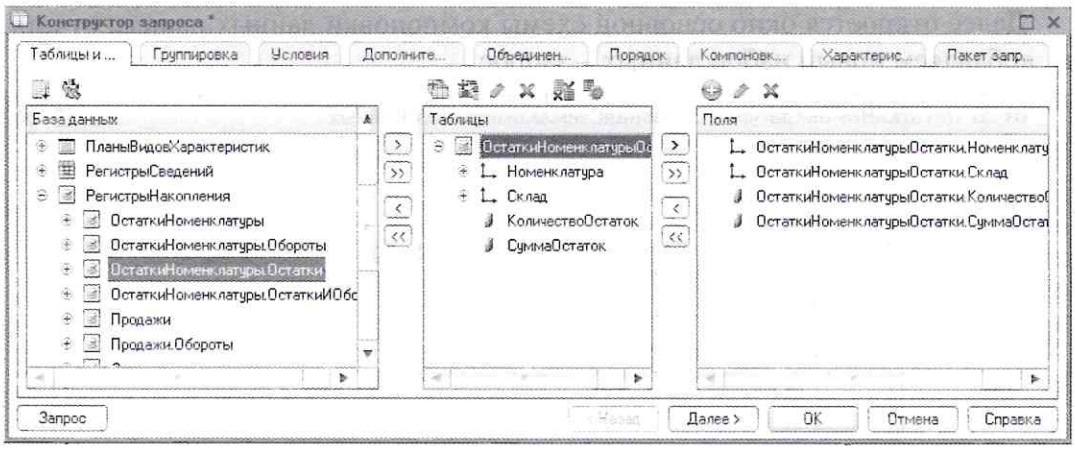


Рис. 2.28. Таблицы и поля

Поскольку больше никаких дополнительных действий в запросе нам по схеме делать не нужно ,нажимаем кнопку "ОК".

В нижней части экрана видим исходный запрос для СКД нашего отчета.

```
ВЫБРАТЬ
ОстаткиНоменклатурыОстатки.Номенклатура,
ОстаткиНоменклатурыОстатки.Склад,
ОстаткиНоменклатурыОстатки.КоличествоОстаток,
ОстаткиНоменклатурыОстатки.СуммаОстаток
ИЗ
РегистрНакопления.ОстаткиНоменклатуры.Остатки КАК ОстаткиНоменклатурыОстатки
```

Почему говорим "исходный"? Потому что в момент исполнения отчета система обязательно перечитывает настройки СКД, которые выберет пользователь, в оперативной памяти внесет изменения в наш текст запроса с целью оптимизации под эти настройки и только тогда станет исполнять сам запрос и наполнять отчет результатом этого запроса.

Большую часть работы система в нашем отчете будет выполнять автоматически. Но кое-что давайте поможем ей сделать

Во-первых, стоит объяснить , какие выходные поля надо воспринимать в качестве ресурсов, т.е. какие данные будут впоследствии интерпретироваться как показатели в отчете.

В нашем случае это "КоличествоОстаток" и "СуммаОстаток". Переходим на закладку "Ресурсы" и двойным кликом выбираем эти поля.

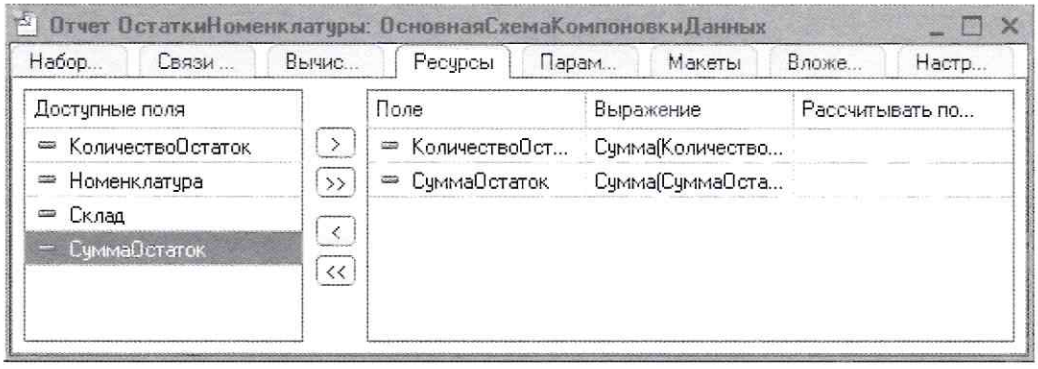


Рис. 2.29. Указание ресурсов

Во-вторых, стоит создать основную настройку отчета. Для этого открываем закладку "Настройки", там добавляем новую группировку.

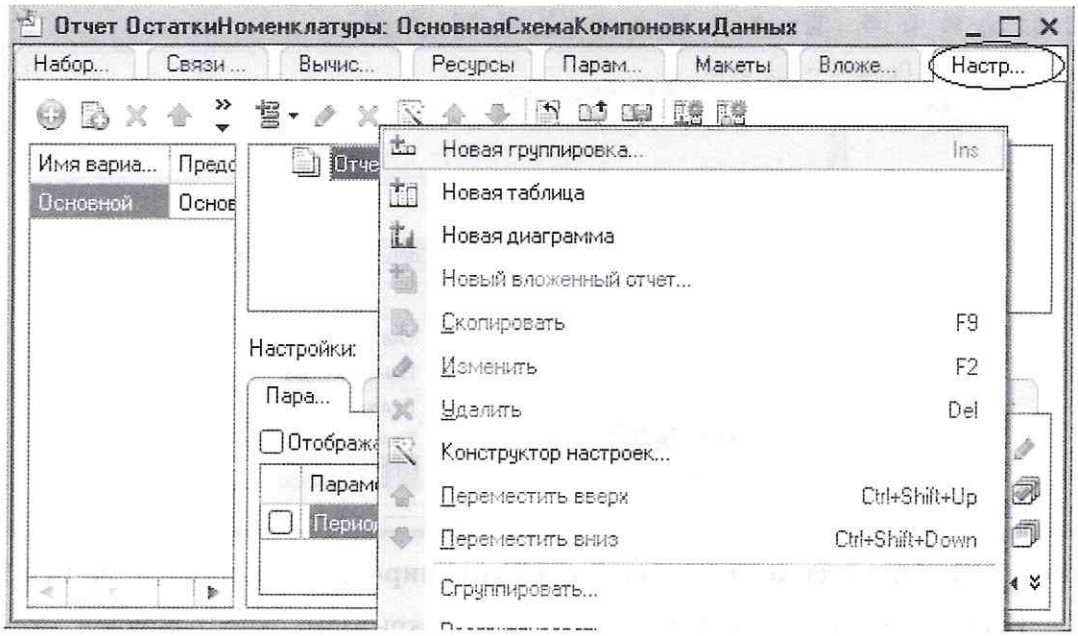


Рис. 2.30. Добавление группировки к настройке

Поскольку нам интересен сейчас результат нашего запроса в "неприкрашенном виде", то качестве значения группировки ничего не выбираем, оставляем поле пустым:

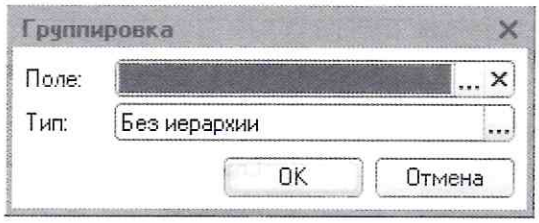


Рис. 2.31. Заполнение значения группировки

После нажатия кнопки "ОК" система введет в основную настройку нашего отчета группировку <Детальные записи>.

Теперь заполним выходные поля этой группировки всеми полями, которые мы получили в результирующей таблице запроса.

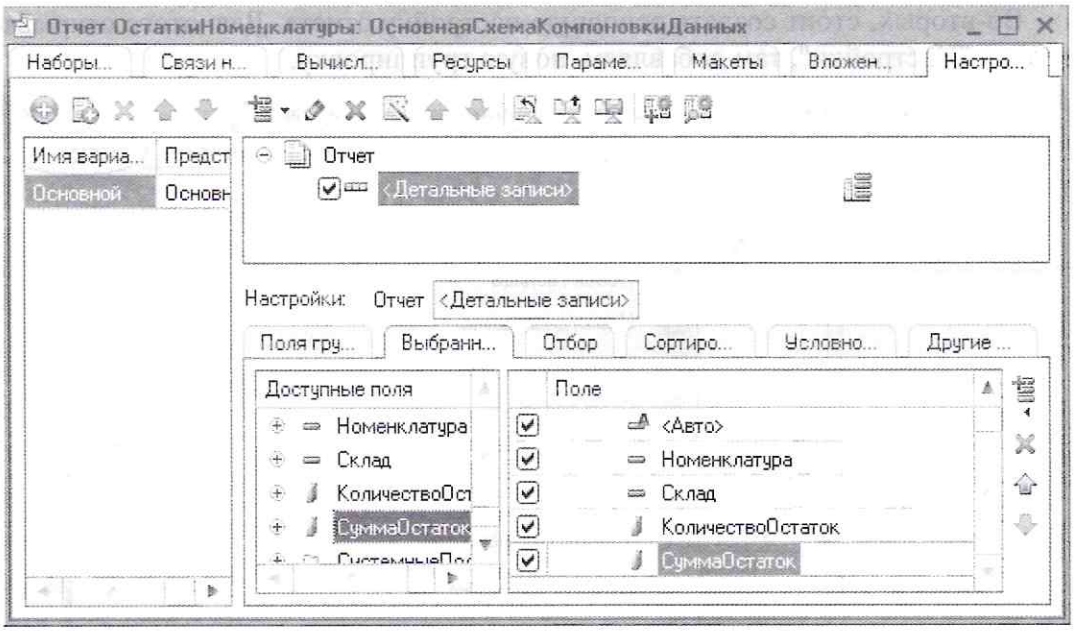


Рис. 2.32. Заполнение выходных полей группировки "Детальные записи"

Итак, основные действия мы сделали. Закрываем окно основной схемы компоновки данных.

Остается только привязать наш отчет к какой-нибудь из подсистем. Например, так, как на рисунке 2.33. Выбор подсистемы для отчета.

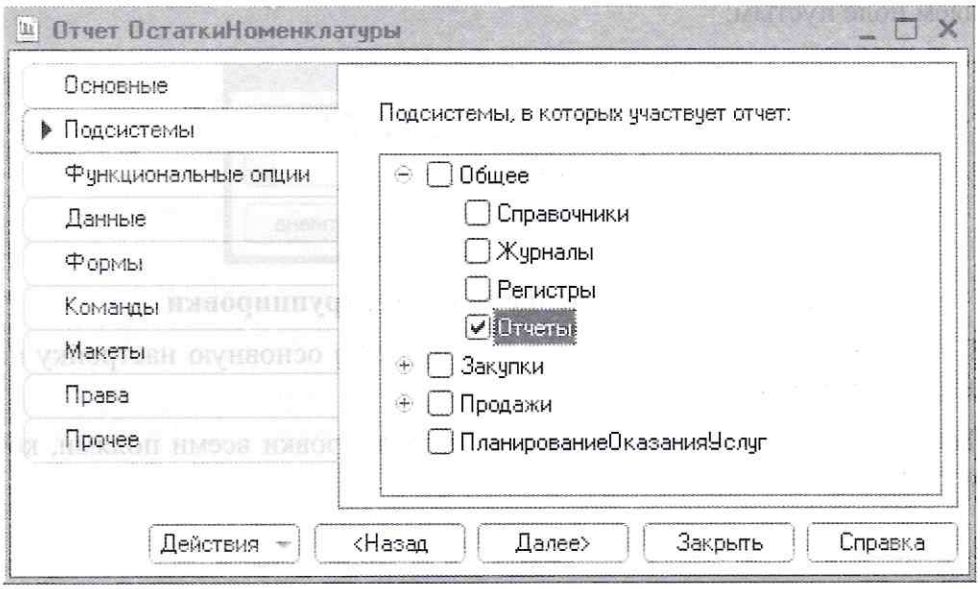
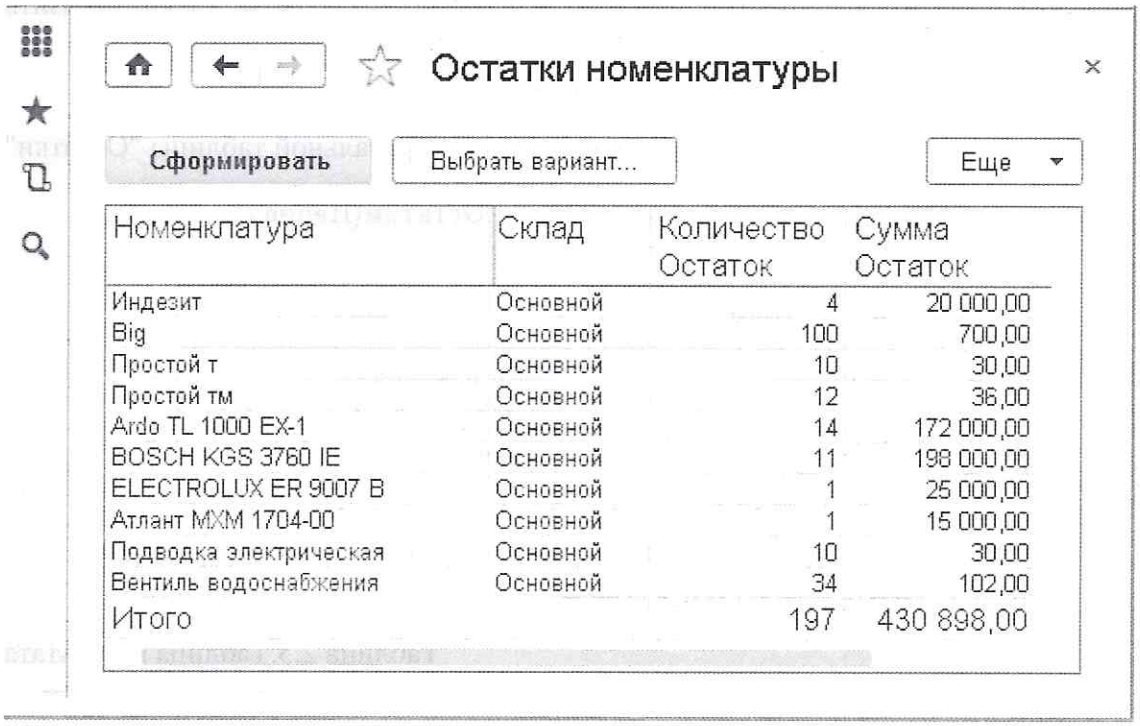


Рис. 2.33.Выбор подсистемы для отчета

Запускаем пользовательский режим, сохраняя все изменения.

При формировании отчета убеждаемся, что получили то, что хотели.



Номенклатура	Склад	Количество Остаток	Сумма Остаток
Индезит	Основной	4	20 000,00
Big	Основной	100	700,00
Простой т	Основной	10	30,00
Простой тм	Основной	12	36,00
Ardo TL 1000 EX-1	Основной	14	172 000,00
BOSCH KGS 3760 IE	Основной	11	198 000,00
ELECTROLUX ER 9007 B	Основной	1	25 000,00
Атлант МХМ 1704-00	Основной	1	15 000,00
Подводка электрическая	Основной	10	30,00
Вентиль водоснабжения	Основной	34	102,00
Итого		197	430 898,00

Рис. 2.34.Отчет глазами пользователя

Создание и сохранение новых настроек и вариантов отчета Вы проходили в предыдущем курсе "Введение в конфигурирование…". Сейчас к этой теме возвращаться не будем.

Лучше еще раз обратим внимание на виртуальную таблицу "ОстаткиНоменклатурыОстатки".

Саму таблицу мы получили посредством конструкции языка запроса:

РегистрНакопления.<имяРегистра>.Остатки(<Период>, <Условие>).

<Период> – это параметр типа <Дата>, <Момент времени> или <Граница>, на который будет произведен расчет итогов для формирования таблицы (то есть – не включая). Если период пропущен , данные получаются по самую последнюю запись в регистре, включительно. Если заполнен и имеет тип <Дата> или <Момент времени> , то данные получаются на начало этой хронологической сущности, т.е. не включая записи, попавшие в регистр со значениями, совпадающими со значениями этого параметра.

В качестве <Условия> может выступать логическая конструкция, оперирующая с полями регистра накопления. Используется для ограничения состава записей, по которым будут выбираться итоги. Заметьте, <Условие> здесь применяется к исходным записям, которые содержатся в регистре, а не к отобранным. Поэтому применение условия в параметрах виртуальной таблицы, как правило, сокращает время выполнения запроса.

И последнее замечание. В расчете для данных выходной таблицы будут использоваться только активные записи регистра!

Прием "Запись в регистр неактивных записей" будет рассматриваться в другой части комплексного курса: "Конфигурирование в системе 1С:Предприятие 8. Решение бухгалтерских задач". При решении оперативных задач используется

он очень редко. Но, тем не менее, такая возможность есть. Например, для решения задач, когда "что-то произошло, в регистре это событие как-то отразить надо, но уже поздно – на итоги оно все равно не повлияет".

Итак, на выходе всегда получаем таблицу вида

Таблица 2.4. Выходные поля виртуальной таблицы "Остатки"

РегистрНакопления.<Имя>.Остатки(Период, Условие)	
Поле	**Тип**
<Измерение>	Тип измерения
<Ресурс>Остаток	Число

то есть с измерениями и итогами ресурсов по этим наборам измерений.

В нашем случае получилась таблица:

Таблица 2.5.Таблица результата

Номенклатура	**Склад**	**КоличествоОстаток**	**СуммаОстаток**
Паркер "Golg"	Основной	40	3 800.00
Kohinor тм	Основной	60	360.00

Внимание, обращение к итогам ресурсов требует добавления части слова "Остаток".

Все остальное многообразие выходных отчетных форм по этой таблице уже "дело рук" схемы компоновки данных.

Практикум №5

Постройте конструктором отчет "Задолженность клиентов" по таблице получения остатков регистра "Задолженности" с возможностью выбора клиента и указания даты в форме.

Таблица Оборотов

Работа с данной виртуальной таблицей похожа на то, что только что делали. Однако имеются определенные отличия, поскольку оборот – это характеристика, получаемая в интервале дат.

Для получения используется следующий синтаксис:

РегистрНакопления.<ИмяРегистра>.Обороты(<НачалоПериода>,<КонецПериода>,<Периодичность>, <Условие>).

Обратите внимание, параметров, задающих время формирования таблицы, два:

<НачалоПериода> – это параметр типа <Дата>, <Момент времени> или <Граница>, начиная с которого будет произведен расчет итогов для

формирования таблицы (вот здесь - включая). Если НачалоПериода не указывать, данные будут получены с самой первой записи в регистре.

<КонецПериода> – это параметр типа <Дата>, <Момент времени> или <Граница>, по который будет произведен расчет итогов для формирования таблицы (и здесь - включая). Если КонецПериода не указывать , данные будут получены по самую последнюю запись в регистре.

Кроме того, если данные в запросе надо вывести не "целым куском", а "с нарезкой", то используется параметр <Периодичность>. С его помощью можно задавать указание дополнительного разворота итогов по периодичности. Может принимать значения:

Период - тогда не разворачиваются;

Регистратор - по документу-регистратору;

День – по дням;

Неделя, Месяц, Квартал, Год – соответственно.

Авто – вариант, позволяющий за счет выбора выходных полей создавать разворот внутри других разворотов, то есть развернуть показатели по годам, при этом каждый год – по кварталам, квартал – по дням и т.п. Если период не указывать , опять же дополнительного разворачивания не будет.

В качестве <Условия> может выступать логическая конструкция, оперирующая с полями регистра накопления. Используется для ограничения состава записей, по которым будут выбираться итоги.

При получении итогов выходной таблицы будут использоваться только активные записи регистра!

На выходе получаем таблицу вида:

Таблица 2.6. Выходные поля виртуальной таблицы "Обороты"

РегистрНакопления.<Имя>.Обороты(Начало, Окончание, Периодичность, Условие)	
Поле	**Тип**
Период	Дата
Регистратор	ДокументСсылка.<Имя>
<Измерение>	Тип измерения
<Ресурс>Оборот	Число
<Ресурс>Приход	Число
<Ресурс>Расход	Число

Но имейте в виду, что поле "Период" будет существовать только в том случае, если параметр "Периодичность" при формировании таблицы получил не пустое значение (а, например, день, месяц или год). Аналогично поле

"Регистратор" будет существовать, только если параметр "периодичность" был установлен при формировании запроса в значении "Регистратор" или "Авто".

Практикум №6

Постройте отчет "Платежи клиентов" запросом по таблице получения оборотов регистра "Задолженности"

Таблица Остатков и Оборотов

Можно сказать, что таблица получения остатков и оборотов представляет собой комбинацию двух вышеописанных способов. Основное её достоинство заключается в том, что по одной виртуальной таблице можно получать данные о начальном состоянии регистра, приходе, расходе, обороте и конечном состоянии.

Для получения используется следующий синтаксис:

РегистрНакопления.<ИмяРегистра>.ОстаткиИОбороты(

<НачалоПериода>,<КонецПериода>,<Периодичность>,

<МетодДополнения>, <Условие>).

Основная масса параметров заполняется так, как это делалось бы при построении виртуальной таблицы "Обороты".

Но есть еще один параметр:

<МетодДополнения> - это параметр, определяющий необходимость показа данных периодов между <НачалоПериода> и <КонецПериода>, в которых не было никаких движений.

Практикум №7

Сформируйте отчет "Материальная ведомость" по регистру "ОстаткиНоменклатуры". Отчет должен показывать данные в любом выбранном пользователем интервале дат и с разворотом по Номенклатуре и регистратору.

В качестве показателей должны выступать НачальныйОстаток, Приход, Расход, КонечныйОстаток по ресурсу "Количество".

Практикум №8

Обеспечьте, чтобы отчет практикума 7? в отдельном своем варианте показывал дни, в которые совокупная стоимость товаров, лежащих на складе, превышала 100000 руб.

3. Технологии проведения документов

3.1. "Обусловленное" проведение

До сих пор с проведением документов мы сталкивались лишь как со способом записать движения в регистр. Причем в регистр информация вносилась непосредственно из документа. Если такие документы, не изменяя, просто перепроводить, то алгоритм проведения всегда будет отрабатывать одинаково, то есть результат проведения всегда будет одним и тем же. Поэтому проведение таких документов можно назвать безусловным.

Но кроме такой безусловной записи будет встречаться масса задач, когда при проведении надо использовать информацию, взятую из других объектов, то есть в таких ситуациях работа алгоритма проведения обусловлена состоянием этих других объектов. И вот тут уже, даже если мы сам документ не изменяем , его перепроведение может привести к совсем иным результатам. Ведь те самые сторонние данные, на которые опирается наш алгоритм, к моменту повторного перепроведения документа могут измениться. С прикладной точки зрения подобные ситуации по праву называются коллизиями. Ведь как еще оценить факт, что, например, прошлогодняя расходная накладная больше не перепроводится, уверяя, что товара на складе не хватает. Хотя ,когда первый раз ее проводили, все было в порядке.

Итак, алгоритм проведения документа в ситуации, когда данных только самого документа не хватает, можно назвать обусловленным.

При реализации обусловленного проведения менее опасной, с точки зрения потенциальных коллизий, является ситуация, когда другими объектами, из которых приходится брать данные, являются зависимые регистры. Дело в том, что тогда вот эти сторонние данные, по крайней мере, можно проверить, найти документ регистратор, из-за которого они изменились, и попытаться исправить ситуацию или найти "крайнего", то есть автора документа-нарушителя.

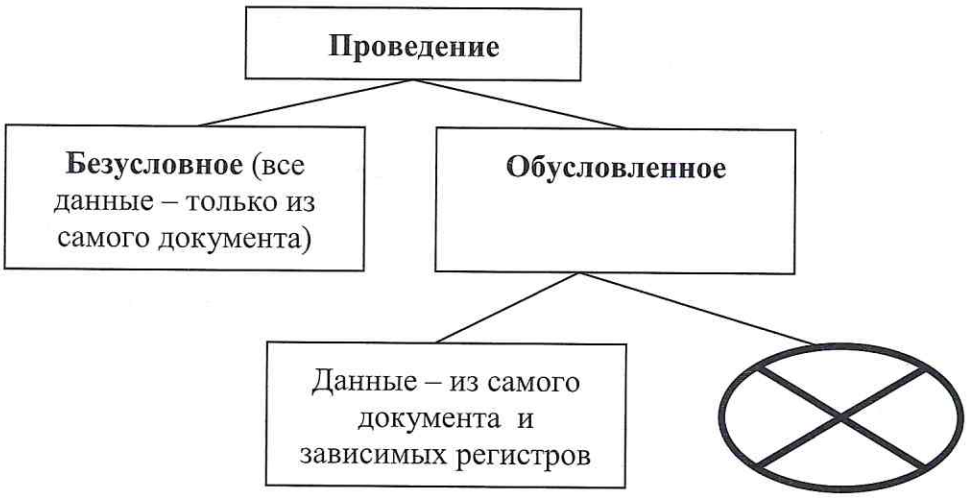


Рис. 3.1. Классификация алгоритмов проведения с точки зрения получения данных

С точки зрения реализации самого алгоритма проведения необходимо обеспечивать его максимальное быстродействие. Поскольку путей оптимизации достаточно много, подробнее о них поговорим потом.

Сейчас возьмем на вооружение один из основных приемов. Необходимо строить алгоритм так, чтобы было как можно меньше обращений к базе данных. Для простейших случаев возьмем за основу алгоритм, при котором будем иметь дело только с двумя обращениями к базе данных: один раз – для чтения недостающих для алгоритма данных, второй – для записи результата в базу данных. И в подобное количество обращений вполне реально "уложиться". Просто недостающие данные надо будет читать запросом (причем лучше всего вообще одним), а запись сформированных наборов записей в регистр можно доверить самой системе по окончании транзакции (только не забудем для этого установить соответствующее свойство каждому из таких наборов записей).

3.2. Сборка алгоритма проведения документа "ПродажаТоваров"

Важно!

Для того чтобы процесс сборки алгоритма проведения занимал как можно меньшее времени и при этом как можно меньше было потенциальных ошибок ,лучше придерживаться следующей технологии:

1) Всегда начинать сборку алгоритма проведения с конструктора движений. Причина очень проста: как бы мы не строили процедуру , в ней обязательно будет программный код, формирующий и заполняющий наборы записей для регистров. А с этой работой быстрее и правильнее всего справляется именно конструктор движений.

2) В построенной конструктором процедуре "ОбработкаПроведения" специальными комментариями-маркерами обозначить недостающие действия.

3) Если по указанным маркерам видно, что для реализации этих недостающих действий будет достаточно данных только самого документа, значит, имеем дело с безусловным проведением. То есть для реализации алгоритма вполне будет достаточно использовать только объектную модель получения данных. Ведь в контексте процедуры "ОбработкаПроведения" весь объект документа , все необходимые для проведения данные. То есть можно пропустить нижеописанные пункты с 4 по 8 включительно.

4) Если же видно, что без привлечения данных сторонних объектов эти недостающие действия реализовать невозможно, значит, имеем дело с обусловленным проведением , то есть придется недостающие данные получать запросом, а значит , первым шагом к построению запроса будет построение схемы запроса на бумаге.

5) Отработка текста запроса должна выполняться там, где можно мгновенно получить его результат, т.е. в консоли запросов.

6) Сборка текста запроса должна выполняться по шагам, от простого к сложному, обязательно выполняя запрос после каждого шага (это позволит быстро находить и исправлять ошибки).

7) Получить просто работающий текст запроса не достаточно, необходимо обеспечить его максимальное быстродействие для данной конкретной задачи.

8) Готовый текст запроса использовать для сборки фрагмента процедуры посредством конструктора запроса с обработкой результата.

9) Дальнейшая сборка алгоритма может выполняться согласно вышеуказанным маркерам в режиме отладки.

10) Работу нельзя считать законченной, пока не реализованы все возможные пути обеспечения максимального быстродействия выполнения всего механизма.

Давайте убедимся в действенности этой технологии на примере проведения документа "ПродажаТоваров".

Постановка задачи разработки алгоритма этого документа звучит следующим образом:

Необходимо проводить расходную накладную только в том случае, если товара на складе хватает (количество в регистре "Остатки номенклатуры" больше, чем количество в документе).

При проведении расходной накладной необходимо списывать по регистру "ОстаткиНоменклатуры" себестоимость, вычисленную средневзвешенным способом, то есть, если в остатке на складе (в регистре) лежит 10 яблок на сумму 20 рублей, то при продаже 2 яблок надо списать:

(20/10) *2 = 4 рубля.

Обеспечить корректное поведение системы, когда в строке табличной части документа указана услуга. Услуги на склад не приходуются, и себестоимость оказания услуг на автоматизированном предприятии не учитывается (считается нулевой).

При проведении документа "ПродажаТоваров" должны быть удалены записи этого же документа по регистру "БронированиеТоваров" (то есть товар, который мы этим документом списываем, логично выводить из состояния бронирования).

Должно выполниться движение по регистру "Задолженности" для увеличения долга контрагента перед нашей организацией на сумму документа.

Начинаем сборку алгоритма с конструктора движений. Вызываем его для объекта конфигурации документа "ПродажаТоваров" и заполняем по необходимым регистрам.

Сама работа с конструктором аналогична тому, что рассматривали в предыдущих темах, кроме работы с регистром "БронированиеТоваров".

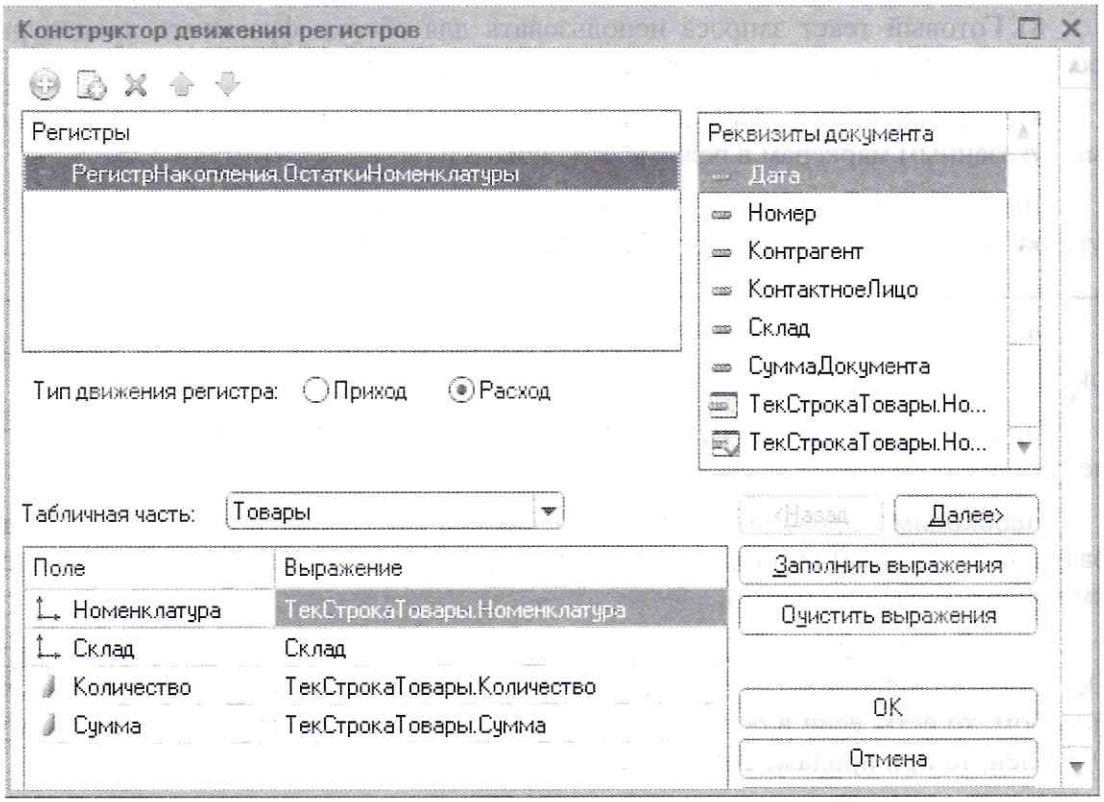


Рис. 3.2.Сборка движений по регистру "ОстаткиНоменклатуры"

В наших интересах, чтобы документ при своем проведении не новые движения по регистру "БронированиеТоваров" формировал, а чтобы "зачищал" старые. Поэтому, добавив регистр в перечень регистров в конструкторе, сами поля регистра при этом заполнять не будем (см. рис. 3.3. Сборка движений по регистру "БронированиеТоваров").

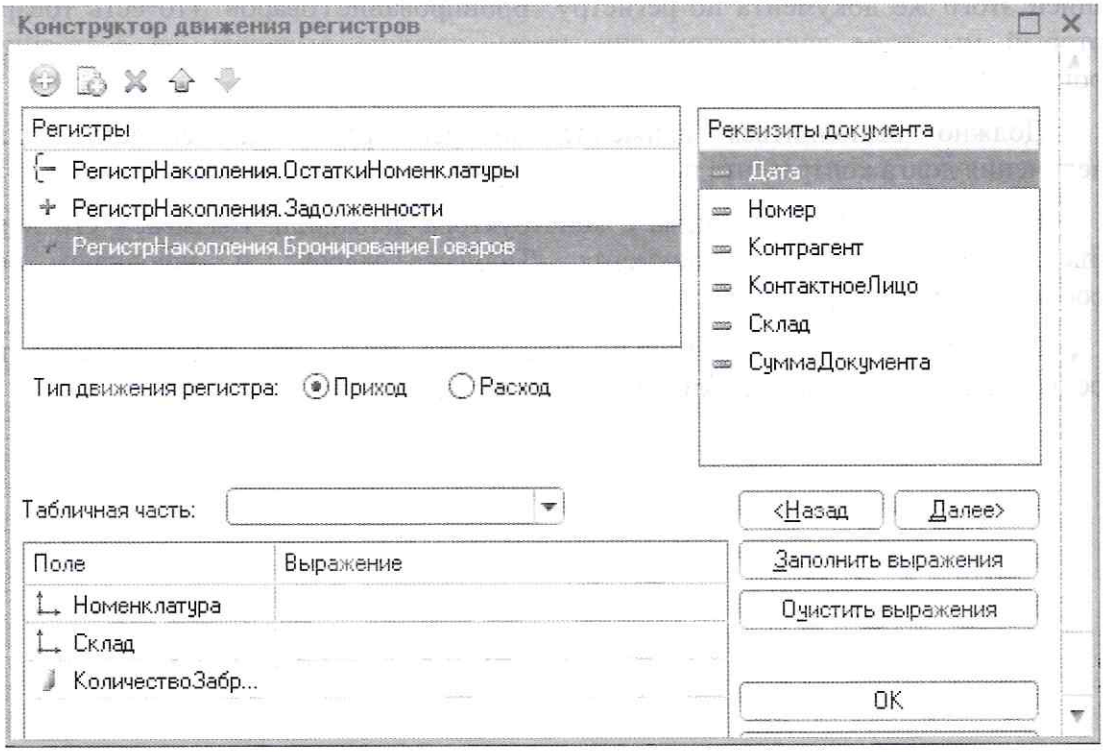


Рис. 3.3.Сборка движений по регистру "БронированиеТоваров"

После нажатия кнопки "ОК" в конструкторе получим процедуру подобного вида:

```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
	Движения.ОстаткиНоменклатуры.Записывать = Истина;
	Для Каждого ТекСтрокаТовары Из Товары Цикл
		// регистр ОстаткиНоменклатуры Расход
		Движение = Движения.ОстаткиНоменклатуры.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
		Движение.Период = Дата;
		Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
		Движение.Склад = Склад;
		Движение.Количество = ТекСтрокаТовары.Количество;
		Движение.Сумма = ТекСтрокаТовары.Сумма;
	КонецЦикла;
	Движения.Задолженности.Записывать = Истина;
	// регистр Задолженности Приход
	Движение = Движения.Задолженности.Добавить();
	Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
	Движение.Период = Дата;
	Движение.Контрагент = Контрагент;
	Движение.СуммаДолга = СуммаДокумента;
	Движения.БронированиеТоваров.Записывать = Истина;
	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Теперь определяем недостающие действия:

Первым делом можно обратить внимание на необходимость вычисления себестоимости списываемых со склада товаров, ведь в самом документе этих данных нет.

Вычисление себестоимости немыслимо без контроля отрицательных остатков.

И расчет себестоимости, и контроль остатков должны касаться только товаров из табличной части документа.

Поскольку для формирования движений мы будем использовать данные, полученные из регистра, необходимо обеспечить неизменность этих данных в течение выполнения всей процедуры (ведь другие пользователи в это же время вполне могут параллельно проводить другие продажи товаров).

Поскольку проводиться может не только новый документ, но и исправленный ранее проведенный (в том числе могла измениться и его дата), то

важно обеспечить, чтобы при расчете себестоимости данные из регистра брались без учета старых движений нашего документа, то есть перед получением данных из регистра старые движения надо обязательно удалить.

Вписываем в процедуру комментарии - маркеры этих недостающих действий:

```
Процедура ОбработкаПроведения(Отказ, Режим)

    Движения.ОстаткиНоменклатуры.Записывать = Истина;

    //!!! ОБЕСПЕЧЕНИЕ НЕИЗМЕННОСТИ ДАННЫХ МЕЖДУ
    //  РАСЧЕТОМ И ОКОНЧАНИЕМ ПРОВЕДЕНИЯ !!!

    //!!! УДАЛЕНИЕ СОБСТВЕННЫХ СТАРЫХ ДВИЖЕНИЙ
    // ПО РЕГИСТРУ ОСТАТКИ НОМЕНКЛАТУРЫ !!!

    //!!! ПОЛУЧЕНИЕ ДАННЫХ ТОЛЬКО ПО ТОВАРАМ !!!
    Для Каждого ТекСтрокаТовары Из Товары Цикл

        // регистр ОстаткиНоменклатуры Расход
        Движение = Движения.ОстаткиНоменклатуры.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
        Движение.Склад = Склад;
        Движение.Количество = ТекСтрокаТовары.Количество;
    // !!! КОНТРОЛЬ ОТРИЦАТЕЛЬНЫХ ОСТАТКОВ СПИСЫВАЕМЫХ ТОВАРОВ !!!
    // !!! ВЫЧИСЛЕНИЕ СЕБЕСТОИМОСТИ СПИСЫВАЕМЫХ ТОВАРОВ!!!
        //Движение.Сумма = ТекСтрокаТовары.Сумма;
    КонецЦикла;

    Движения.Задолженности.Записывать = Истина;
    // регистр Задолженности Приход
    Движение = Движения.Задолженности.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Контрагент = Контрагент;
    Движение.СуммаДолга = СуммаДокумента;

    Движения.БронированиеТоваров.Записывать = Истина;

КонецПроцедуры
```

Теперь анализируем маркеры.

Маркер "//!!! УДАЛЕНИЕ СОБСТВЕННЫХ СТАРЫХ ДВИЖЕНИЙ ПО РЕГИСТРУ ОСТАТКИ НОМЕНКЛАТУРЫ !!!" в чтении данных не нуждается.

Маркер "ОБЕСПЕЧЕНИЕ НЕИЗМЕННОСТИ ДАННЫХ МЕЖДУ РАСЧЕТОМ И ОКОНЧАНИЕМ ПРОВЕДЕНИЯ" необходимо будет отработать за счет наложения управляемой блокировки на таблицу регистра по тем номенклатурным позициям, которые указаны у нас в табличной части документа. То есть для реализации этого маркера достаточно читать данные из самого документа.

Для остальных маркеров данных самого документа недостаточно. Придется получать данные из регистра "ОстаткиНоменклатуры" и, кроме того, из справочника "Номенклатура" (ведь только в справочнике есть реквизит "ВидНоменклатуры", который понадобится для отсева услуг).

Таблица 3.1. Выходная таблица запроса.

Номенклатура	КолДок	Склад	КолОст	СуммаОст
Паркер "Golg"	20	Основной	30	380.00
Kohinor тм	50	Основной		
Паркер "Golg"	15	Основной	30	380.00

Анализируя эти маркеры, составляем начальный эскиз выходной таблицы запроса (см. Таблица 3.1 Выходная таблица запроса).

При составлении эскиза важно отразить не только комфортные условия для проведения нашего документа, но и максимально "неудобные".

Например, когда какого-то товара нет на складе, когда пользователь один и тот же товар ввел в табличную часть документа дважды и т.п.

Смысл такого заполнения эскизных данных – заставить самого себя побеспокоиться о корректности нашего механизма проведения заранее, еще на этапе составления запроса. Иначе впоследствии рискуем или что-то не предусмотреть, или опомниться слишком поздно, и тогда весь уже почти собранный алгоритм придется переписывать почти полностью.

Далее превращаем эскиз в схему запроса, то есть нумеруем выходные поля, определяем таблицы-источники и определяем дополнительные действия, которые необходимо будет реализовать в запросе.

В качестве дополнительных действий напрашиваются:

- отбор данных только текущего проводимого документа;
- отсечение строк с услугами;
- группировка по номенклатуре таблицы документа для учета наличия "дублей" товаров;

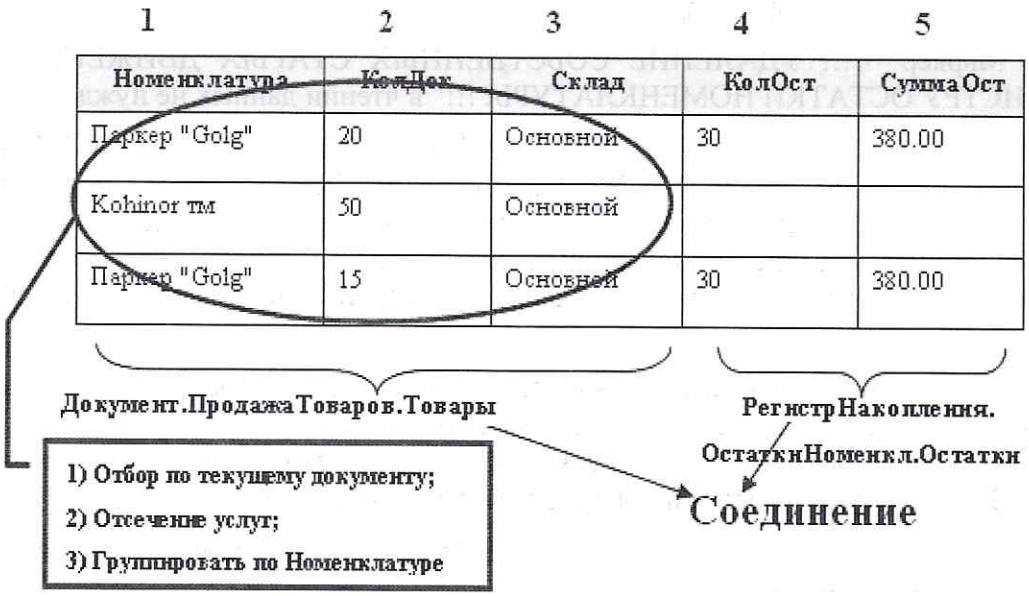


Рис. 3.4.Схема запроса для проведения документа "ПродажаТоваров"

Кроме того, судя по тому, что получается на схеме запроса после определения таблиц-источников, мы имеем дело с построением запроса по нескольким источникам. А значит, в запросе придется применять или "Объединение" или "Соединение".

Важно!

Для облегчения реализации запроса по нескольким источникам можно руководствоваться таким правилом:

Если на схеме запроса видно, что второй источник имеет право добавлять новые группировочные сущности (в нашем случае – новые номенклатурные позиции), то лучше применять ОБЪЕДИНЕНИЕ. Если не имеет, то есть добавляет информацию только в новые колонки, то лучше применять СОЕДИНЕНИЕ.

Если следовать этому правилу , нам лучше применить СОЕДИНЕНИЕ таблиц-источников.

Когда схема готова, можно приступать к ее реализации.

Обработка "Консоль запросов для обычного приложения" есть в составе учебной конфигурации. Но для работы в этой обработке, как следует из ее названия, необходимо будет запустить систему в режиме обычного приложения.

Далее запускаем по нажатию на эту кнопку программу в режиме обычного приложения, входим в меню "Операции / Обработка" и выбираем запуск обработки "Консоль запросов для обычного приложения".

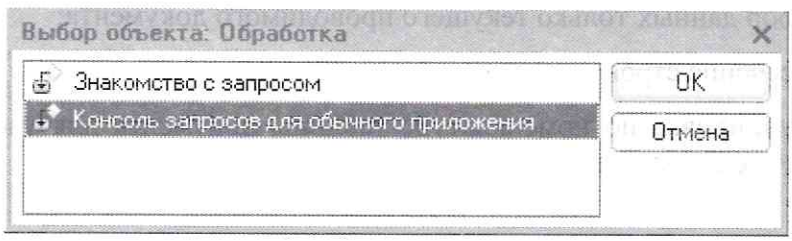


Рис. 3.5.Выбор обработки "Консоль запросов"

Работа в самой консоли запросов начинается обычно с добавления новой ветви к дереву запросов. Это можно сделать посредством вызова контекстного меню (клик правой кнопкой мышки на дереве на слове "Запросы") и выбора пункта "Добавить". Далее даем название своему запросу. Например, "Проведение продажи товаров".

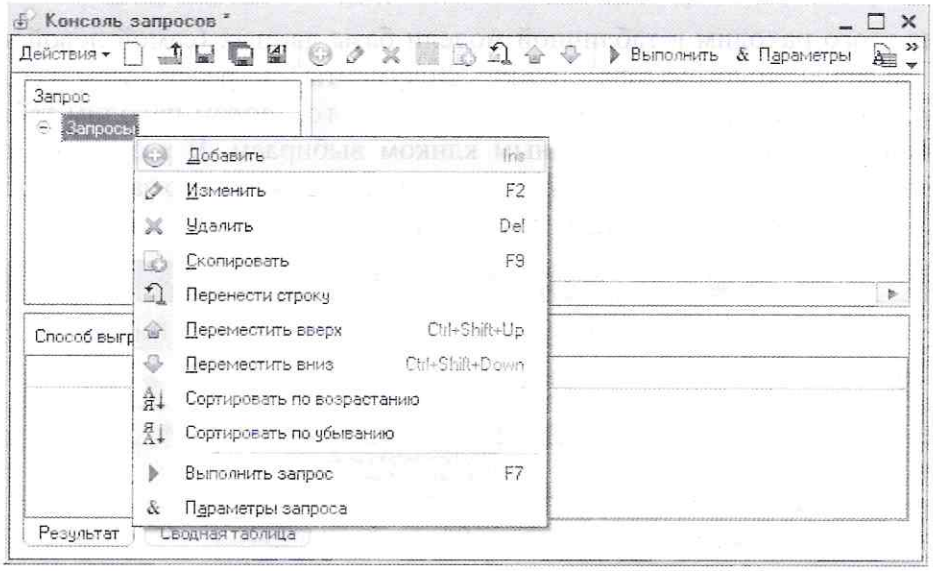


Рис. 3.6.Добавление новой ветки на дереве запросов

Написание самого текста запроса выполняется в текстовом поле, расположенном в верхней правой части обработки, и лучше всего посредством конструктора запроса.

Для вызова конструктора запроса, опять же можно использовать соответствующий пункт контекстного меню (клик правой кнопкой мышки на поле текста запроса, см. рис. 3.7. Вызов конструктора запроса).

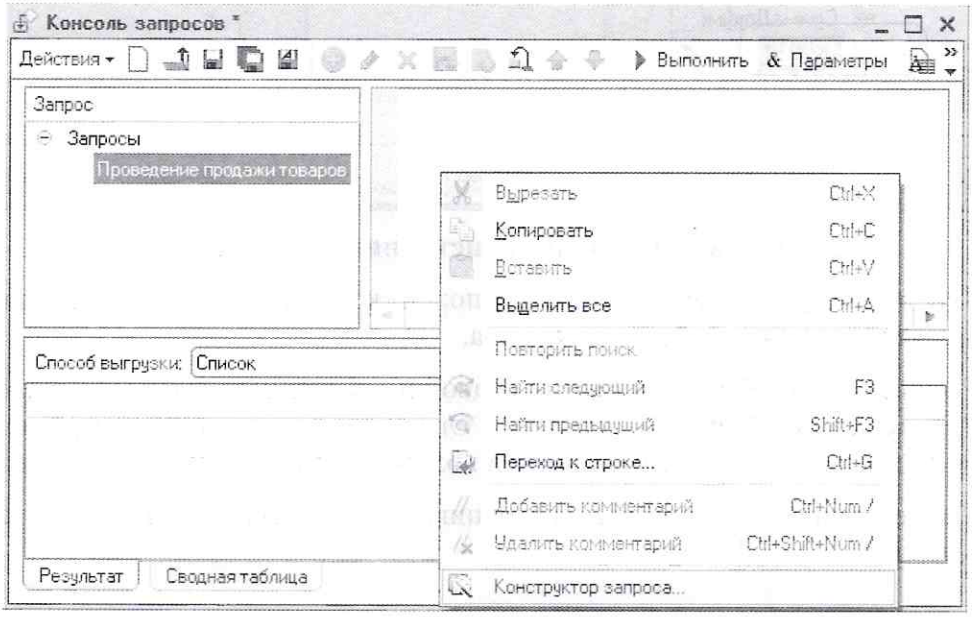


Рис. 3.7.Вызов конструктора запроса

Согласно схеме запроса (см. рис.3.4. Схема запроса для проведения документа "ПродажаТоваров".) нам необходимо реализовать соединение

обработанной дополнительными действиями (отбор, отсев услуг, группировка) табличной части документа с виртуальной таблицей регистра.

Согласно пункту шестому технологии сборки алгоритма проведения текст запроса будем отрабатывать по шагам. Первым шагом получим желаемые данные документа с необходимой обработкой.

Для этого находим в табличной модели базы данных (самой левой колонке первой закладки конструктора) пункт "Документы" и раскрываем его, потом находим пункт "ПродажаТоваров" и раскрываем его, потом находим табличную часть "Товары" документа и двойным кликом выбираем. В результате система выберет эту таблицу в качестве таблицы-источника для нашего запроса (вставит в среднюю колонку первой закладки конструктора).

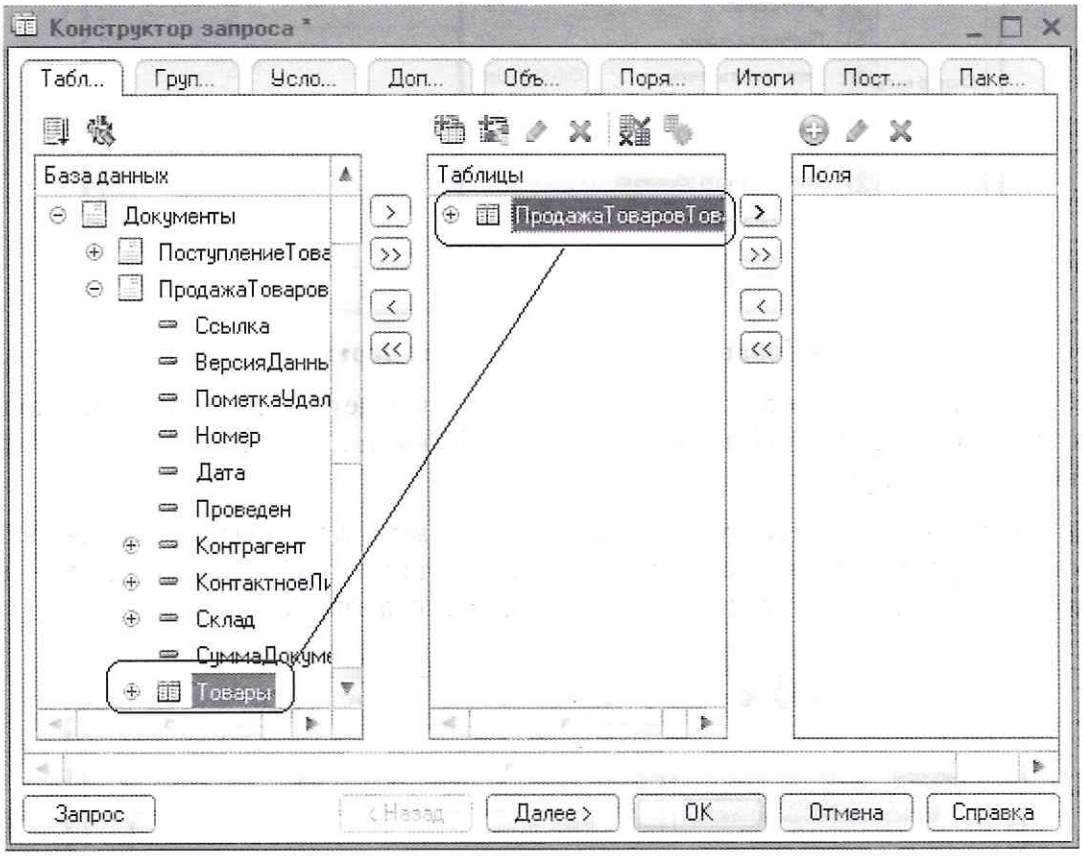


Рис. 3.8.Указание таблицы-источника в конструкторе

Теперь наступает черед выходных полей, которые согласно схеме запроса мы должны получить из нашего источника.

Раскрываем в таблицу-источник и двойными кликами выбираем выходные поля "Номенклатура" и "Количество". Они, соответственно, размещаются в области выходных полей (в самой правой колонке первой закладки конструктора).

Также раскрываем в таблице источнике поле "Ссылка" и двойным кликом выбираем поле "Склад".

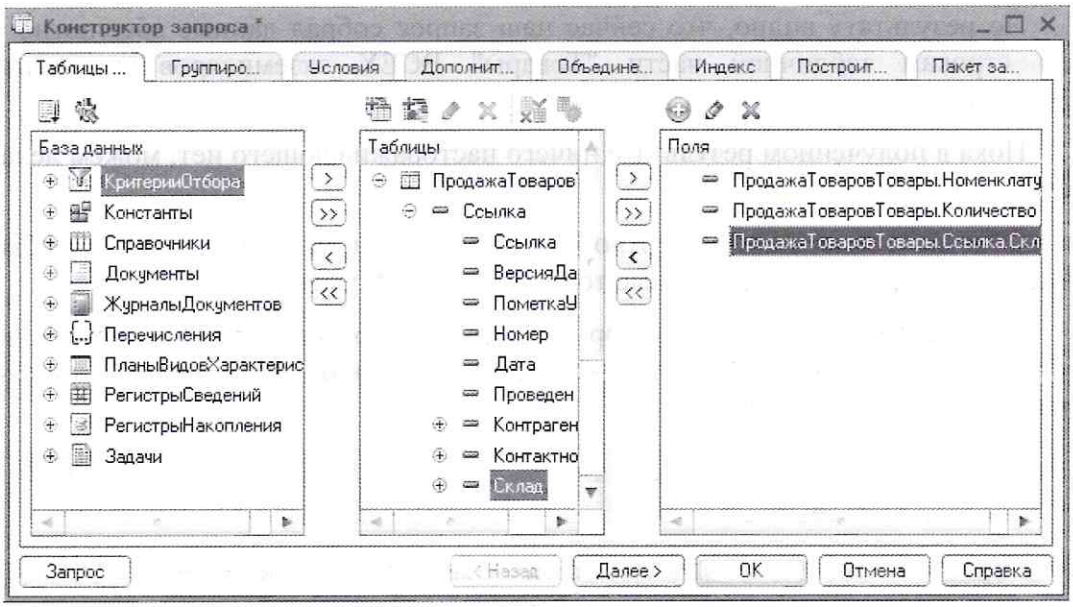


Рис. 3.8.Указание выходных полей в конструкторе

Теперь самое время первый раз убедиться в том, что мы на верном пути, то есть первый раз выполнить тот запрос, текст которого складывается на текущий момент.

Нажимаем кнопку "ОК" в конструкторе запроса. Наблюдаем в верхней правой части консоли запросов текст запроса. В тексте запроса нет надписей зеленого цвета, то есть указания параметров запроса. Значит, запрос может выполняться сразу. Для этого в верхней командной панели нажимаем кнопку "Выполнить".

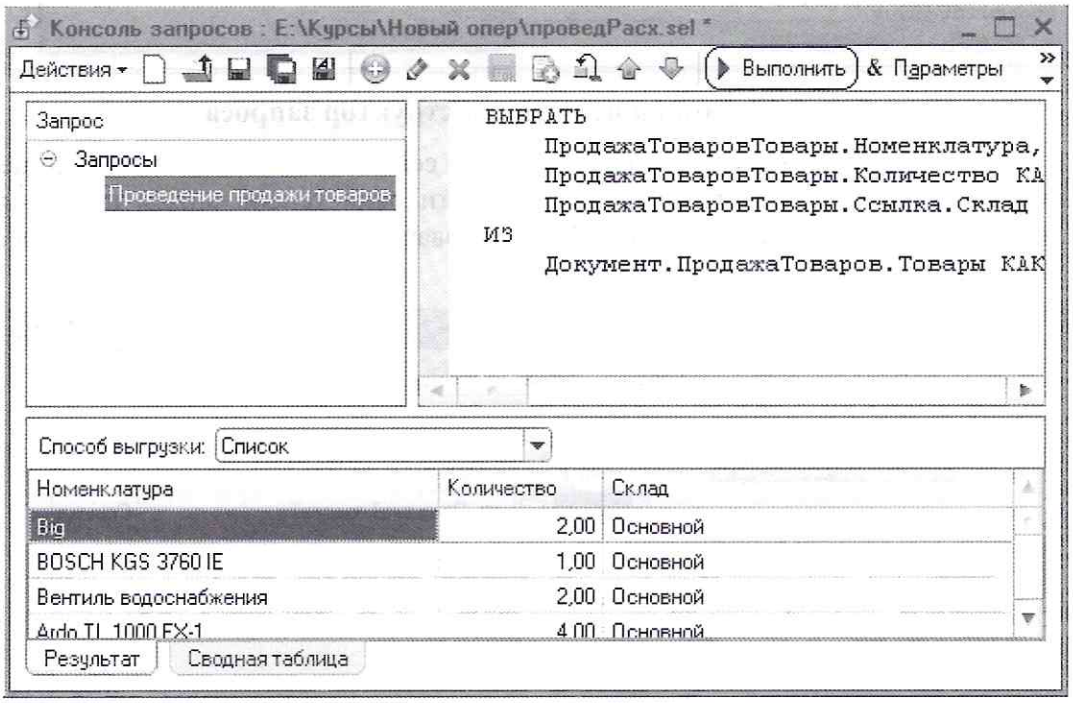


Рис. 3.9.Первое выполнение запроса

Результат выполнения нашего запроса видим в нижнем табличном поле консоли запроса.

По результату видно, что сейчас наш запрос собрал данные абсолютно по всем строкам табличной части "Товары" ВСЕХ экземпляров документов "ПродажаТоваров".

Пока в полученном результате ничего настораживающего нет, можем делать следующий шаг.

Судя по схеме запроса, нужно сейчас наложить условие отбора , чтобы в результат запроса попадали данные только одного документа.

Возвращаемся в конструктор запроса. Для этого опять же достаточно выбрать соответствующий пункт контекстного меню (клик правой кнопкой мышки на поле текста запроса)

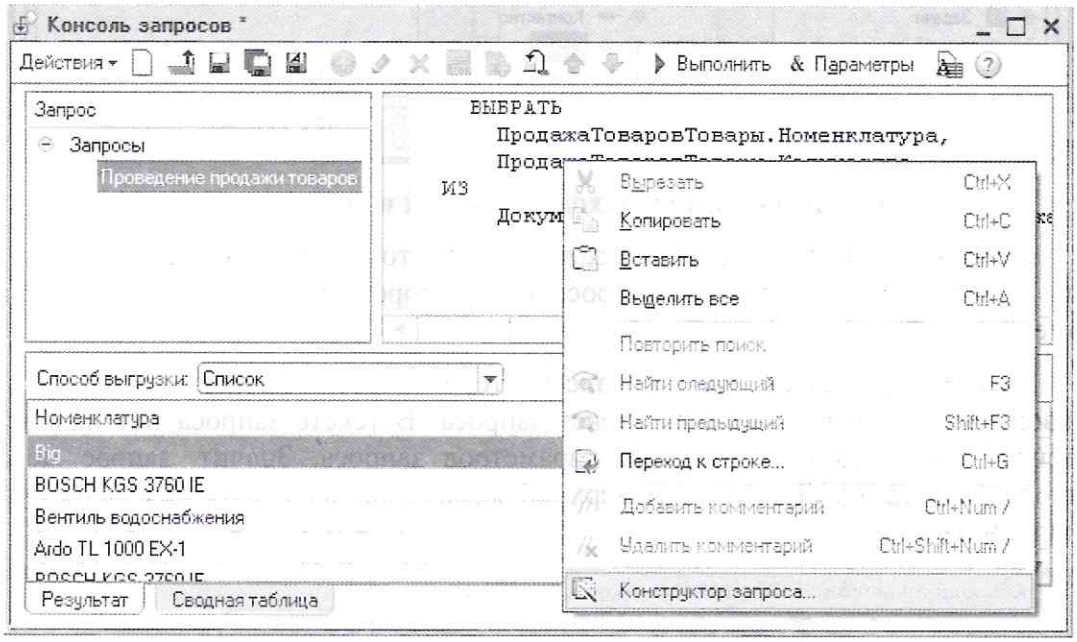


Рис. 3.10. Возвращение в конструктор запроса

В конструкторе запроса, поскольку имеем сейчас дело с реальной таблицей, открываем закладку "Условие". В правой части окна теперь видим псевдоним нашей таблицы-источника "ПродажаТоваровТовары", раскрываем ее и выбираем двойным кликом поле "Ссылка".

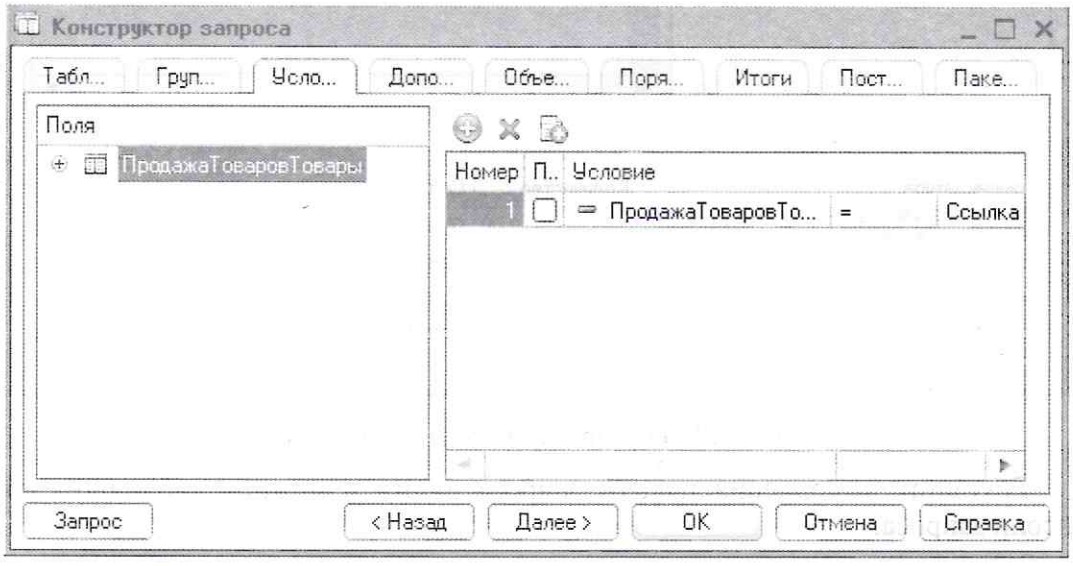


Рис. 3.11.Сборка условия по ссылке на документ

Очередной шаг сделан, нужно убедиться в его корректности.

Нажимаем кнопку "ОК". Попадаем опять в окно консоли запросов. Но обратите внимание – теперь в тексте запроса есть описание параметра "&Ссылка".

Значит, нужно нажать кнопку "&Параметры" в верхней командной панели, чтобы установить значение этого параметра, иначе запрос не сможет выполниться.

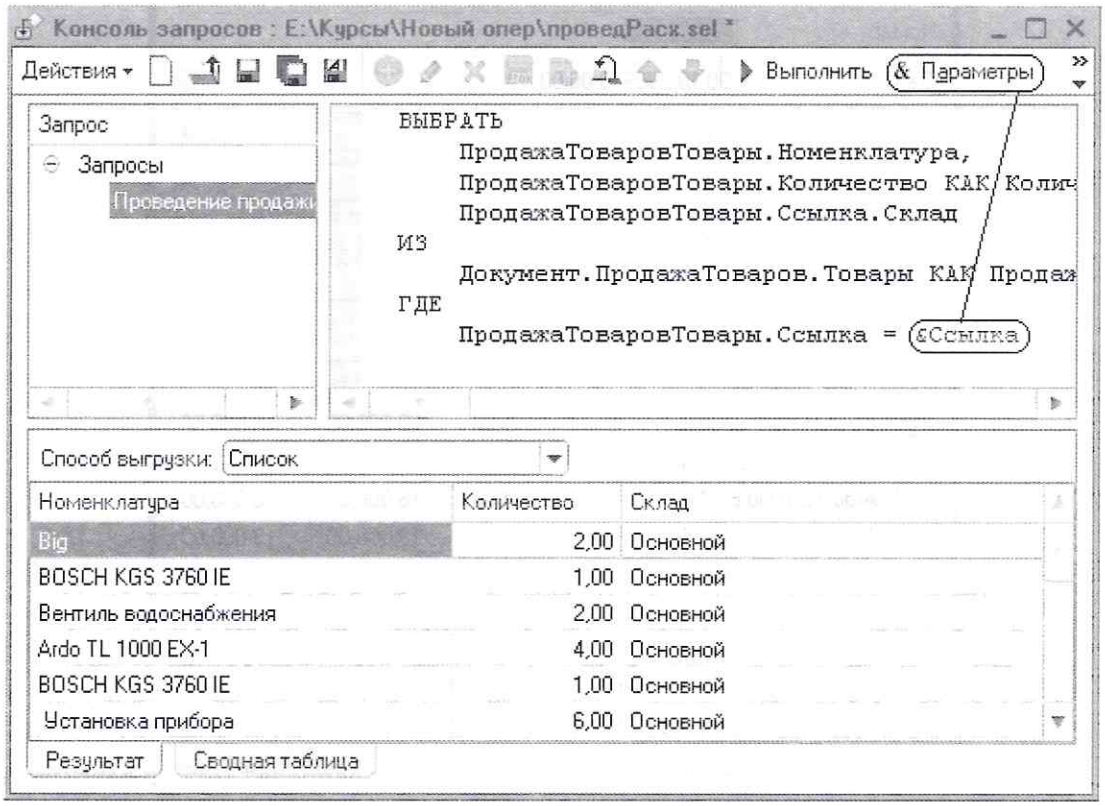


Рис. 3.12. Кнопка "&Параметры"

В открывшемся окне "Параметры запроса" лучше всего нажать кнопку "Получить из запроса". Это позволит системе самой определить по тексту запроса, какие параметры нужны. Для каждого параметра прописать в табличном поле окна название параметра и установить тип значения в каждой ячейке заполнения значений параметров.

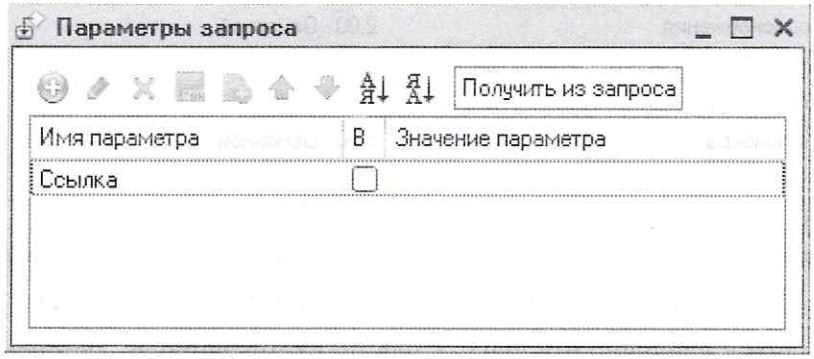


Рис. 3.13. Кнопка "Получить из запроса"

В нашем случае такой параметр только один – это "Ссылка". Для выбора значения делаем сначала двойной клик в соответствующем поле, потом нажимаем на кнопку выбора и из открывшегося списка документов давайте выберем самый

первый документ "Продажа товаров 000000001". Дело в том, что в демобазе при заполнении этого документа намерено в табличную часть дважды внесли товар "BOSCH KGS 3760 IE". Чуть позже нам это пригодится для тестирования учета наличия "дублей строк в документе".

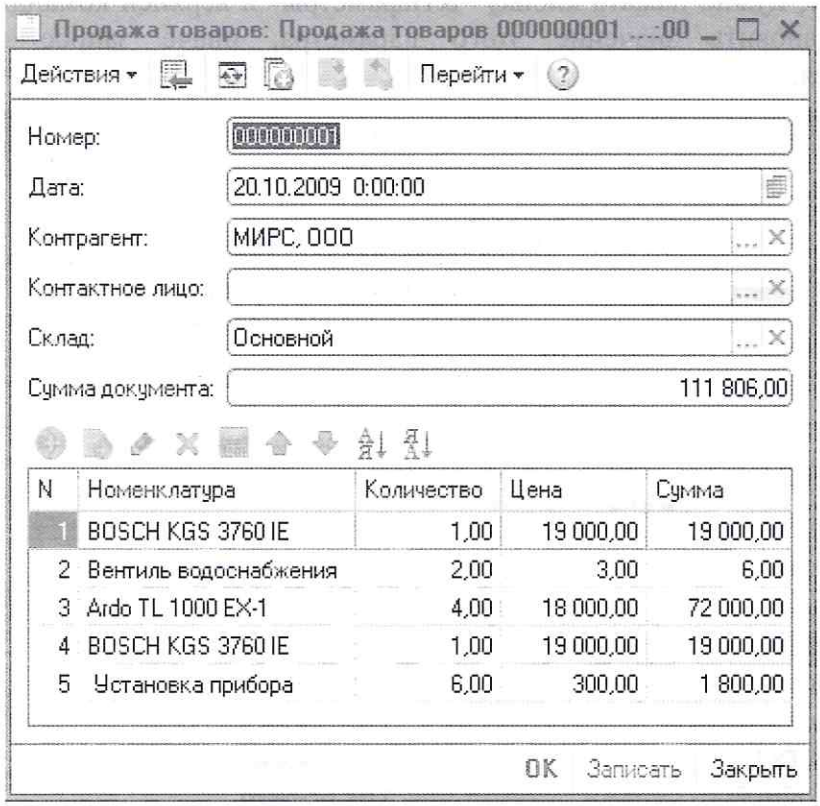


Рис. 3.14. Тестовый документ

Заполнив значение параметра, окно "Параметры запроса" можно закрывать, а сам запрос выполнить.

Рассматривая результат выполнения запроса, видим, что он соответствует табличной части именно нашего документа.

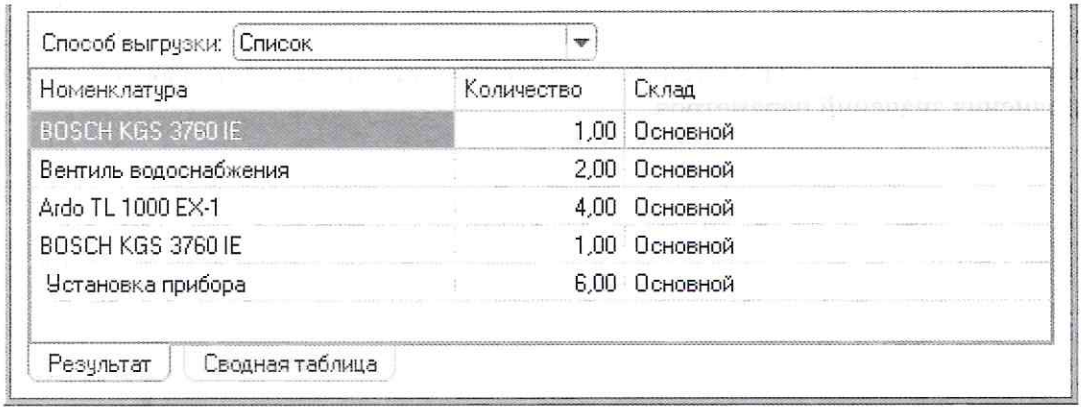


Рис. 3.15. Результат запроса с отбором по документу

Переходим к следующему шагу. Судя по схеме запроса , на очереди "отсев" услуг.

Для этого возвращаемся в конструктор нашего запроса, опять открываем закладку "Условия". Раскрываем таблицу источник, раскрываем "плюсик"

напротив поля "Номенклатура", находим среди свойств номенклатуры "ВидНоменклатуры" и двойным кликом формируем по этому полю условие.

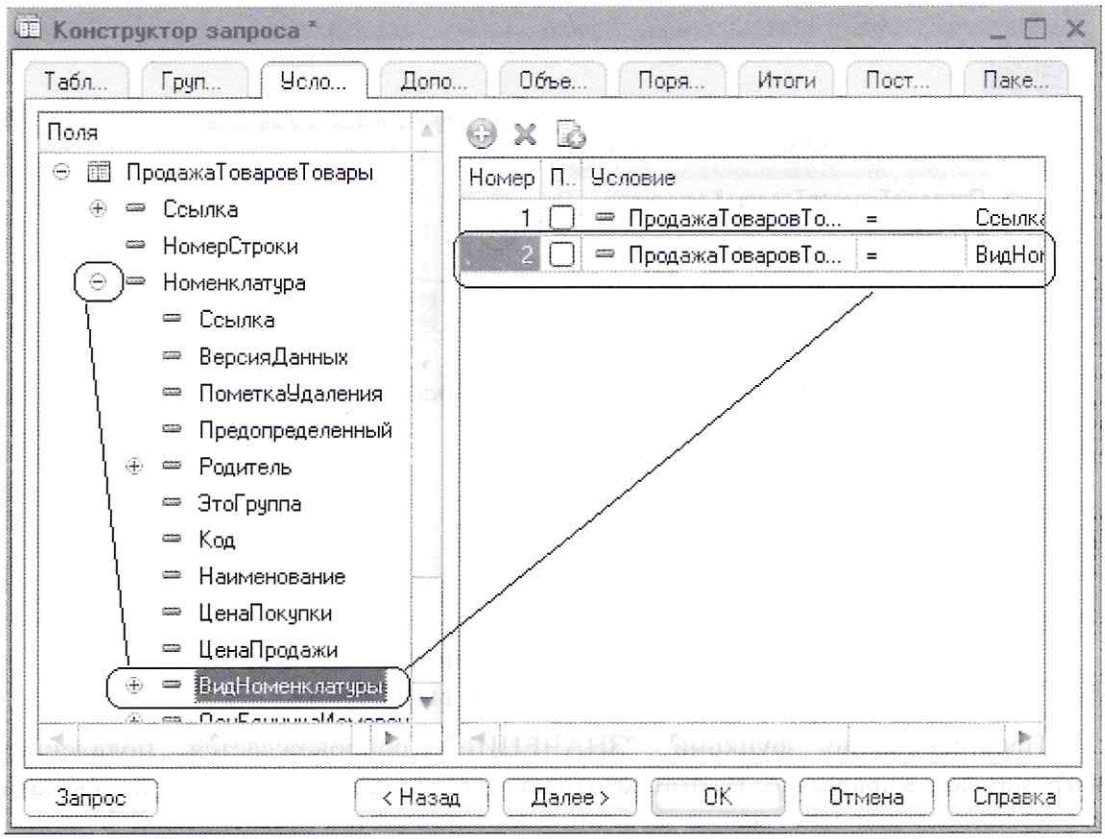


Рис. 3.16. Условие по виду номенклатуры

На текущий момент конструктор собрал условие типа "равенство значению внешнего параметра", но нам нужно - другое. Нам нужно условие неравенства значению "Услуга" перечисления "ВидыТоваров".

Запросы умеют работать с предопределенными значениями в самом тексте запроса. Поэтому в добавленном втором условии сначала меняем тип условия с "равно" на "неравно" (т.е. с "=" на "<>"). А потом ставим флажок "Произвольное условие" (см. рис. 3.17. Превращение условия в произвольное).

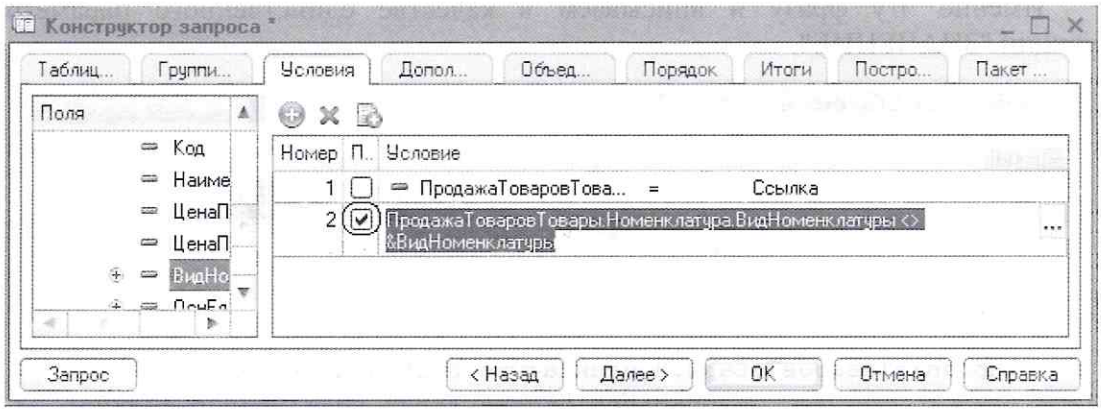


Рис. 3.17. Превращение условия в произвольное

Теперь можно нажать появившуюся в конце строки условия кнопку выбора. Открывается окно конструирования произвольного выражения. Убираем в нем окончание фразы "&ВидНоменклатуры", а в верхней правой части окна открываем раздел "Функции языка запросов", в нем открываем подраздел

"Прочее" и перетаскиваем мышкой функцию "ЗНАЧЕНИЕ" в область произвольного выражения (нижнюю часть окна) после значка неравенства.

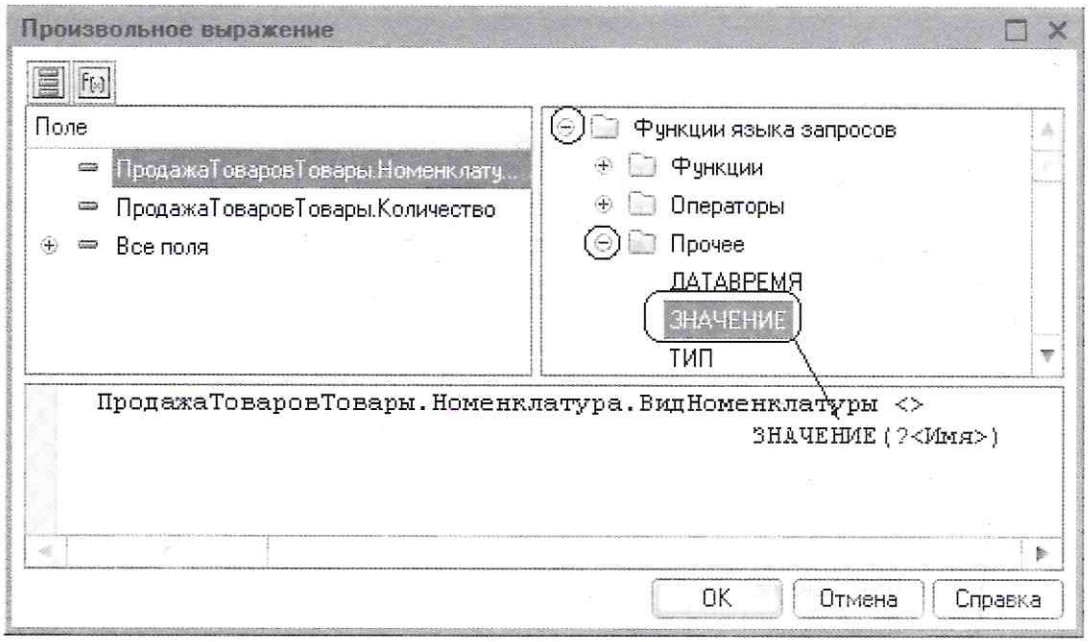


Рис. 3.18. Перетаскивание функции языка запроса в область произвольного выражения

Перетаскивание функции "ЗНАЧЕНИЕ" сопровождается появлением заготовки под выражение. В этой заготовке система всегда пытается подсказать, чего не хватает для заполнения того или иного параметра применяемой функции. В нашем случае не хватает имени значения, с которым надо будет впоследствии сравнивать значение поля "ВидНоменклатуры".

Чтобы собрать такое имя ,лишний раз стоит проверить в конфигураторе, какой именно тип значения у поля и какое значение мы будем "ловить". В нашем случае тип у поля <ПеречислениеСсылка.ВидыТоваров>, а название нужного значения этого перечисления "Услуга". Значит, все выражение должно выглядеть так:

Перечисление.ВидыТоваров.Услуга.

Именно эту фразу и вписываем в качестве единственного параметра функции "ЗНАЧЕНИЕ".

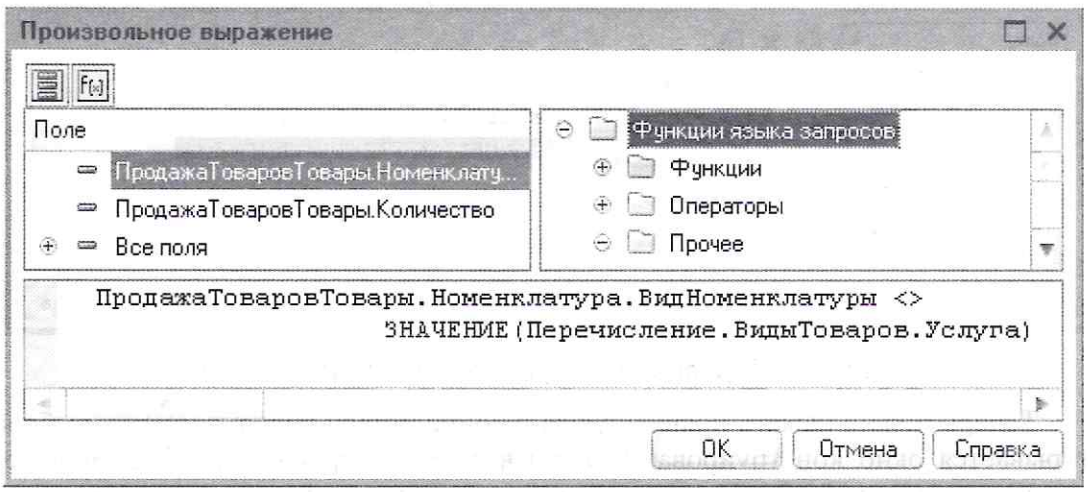


Рис. 3.19. Заполнение параметра функции "ЗНАЧЕНИЕ"

Теперь можно нажимать кнопку "ОК" в окне произвольного выражения, нажимать кнопку "ОК" в окне конструктора запроса. И поскольку нам удалось избежать добавления новых внешних параметров, можно сразу выполнять запрос. Убедимся, что строчка с услугой "Установка прибора" из результата запроса по нашему тестовому документу убралась.

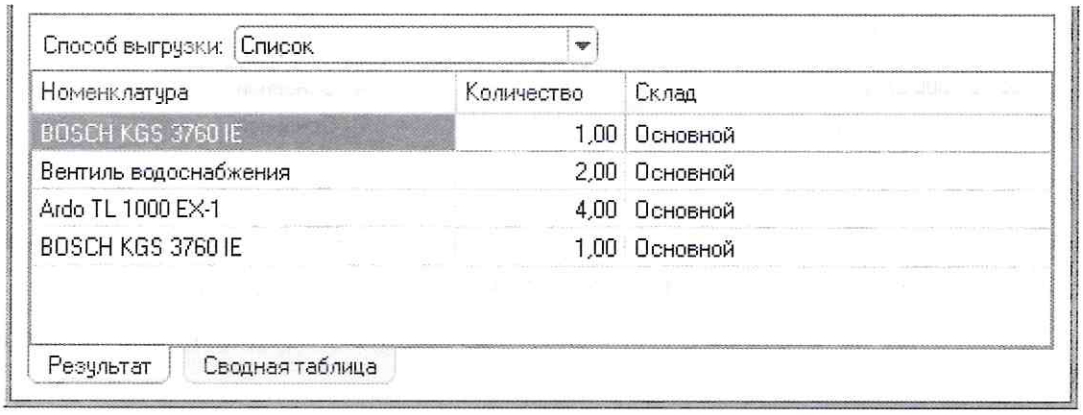


Рис. 3.20. Результат запроса с отбором "без услуг"

Переходим к следующему шагу. Судя по схеме запроса, это группировка (сворачивание) таблицы. Действительно, в тестовом примере товар "BOSCH KGS 3760 IE" введен в табличную часть документа несколько раз.

Возвращаемся в конструктор нашего запроса. Открываем закладку "Группировка" и перетаскиваем мышкой поля "ПродажаТоваровТовары.Номенклатура" и "ПродажаТоваровТовары.Ссылка.Склад" из левой части экрана в область "Групповое поле" (верхнюю правую область), а поле "Количество" -в область "Суммируемое поле" (нижнюю правую область). Автоматически подставленная при этом агрегатная "Сумма" нас очень даже устраивает. Ведь группирование именно для того и делаем, чтобы узнать, сколько штук товара "BOSCH KGS 3760 IE" было указано в документе всего.

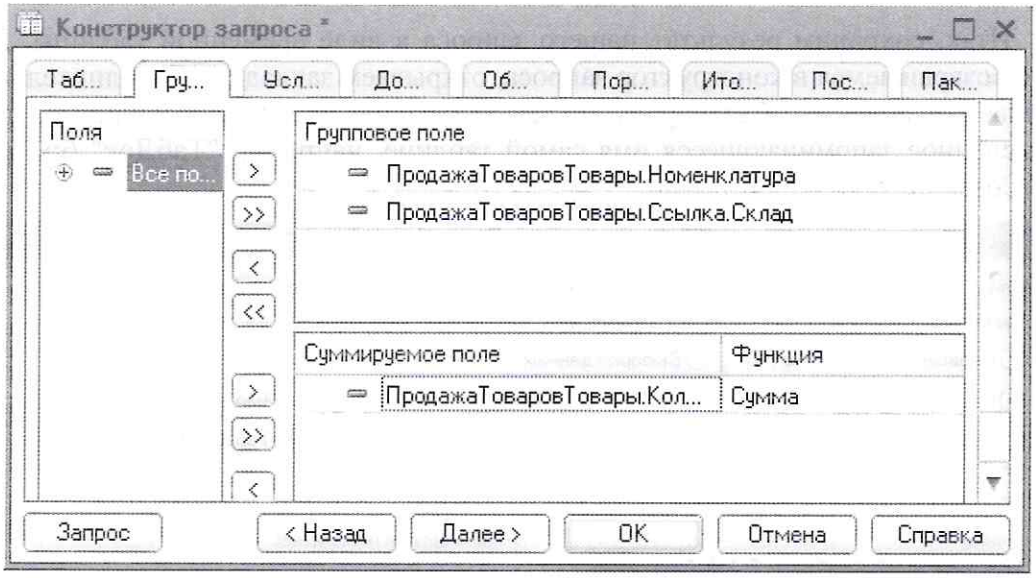


Рис.3.21. Группировка в конструкторе запроса

После нажимаем кнопку "ОК" в окне конструктора запроса и, поскольку новых параметров мы не добавили, можем сразу выполнять запрос.

В результате в выходной таблице запроса по нашему тестовому примеру видим, что операции отбор по текущему документу, отсев услуг и группировка таблицы по номенклатуре нам удались.

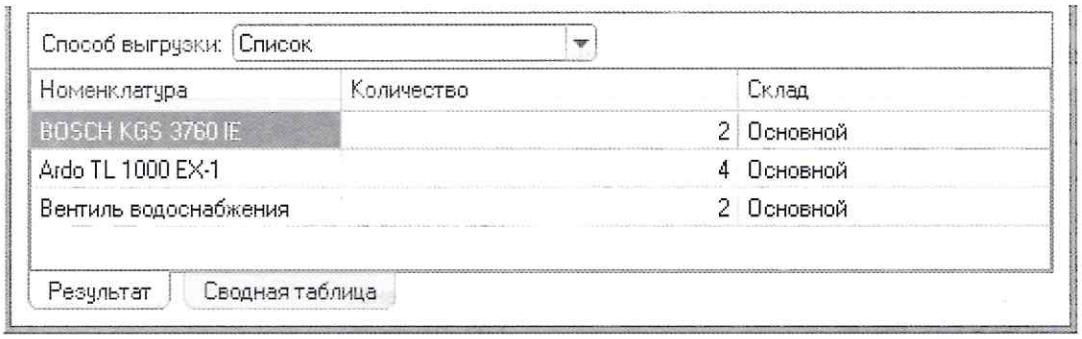


Рис. 3.22. Результат группировки по номенклатуре

Итак, судя по схеме запроса, можем переходить к соединению с таблицей регистра.

Но прежде чем это сделать, давайте лучше результат наших предыдущих усилий закрепим в виде временной таблицы пакета запросов. Почему именно для нашей задачи это будет очень полезно , мы рассмотрим чуть позже. Пока же хотелось бы сказать вот что.

Если в Вы не программировали предварительно года два-три в SQL, если написание запроса для "задачки посложнее" Вы не считаете "гимнастикой для ума, приносящей удовольствие" , то совершенно незазорно вообще любую задачу на запрос пытаться решать как "пакет запросов с сохранением промежуточных результатов в виде временных таблиц".

И хотя при желании любую идею, конечно, можно довести до абсурда, но, надеюсь, Вы не будете в процессе своей работы руководствоваться именно желанием абсурда. А примеров, когда прием "Разбиение запроса на части в виде пакета запросов" избыточен и вреден, гораздо меньше, чем примеров, когда без него запросы получаются очень запутанными и некорректными.

Итак, сохраним результат нашего запроса в виде временной таблицы. Для этого возвращаемся в конструктор запроса, открываем закладку "Дополнительно", отмечаем в области "ТипЗапроса" пункт "Создание временной таблицы", даем осмысленное запоминающееся имя самой таблице, например "ТабДок" (см. рис. 3.23.Создание временной таблицы).

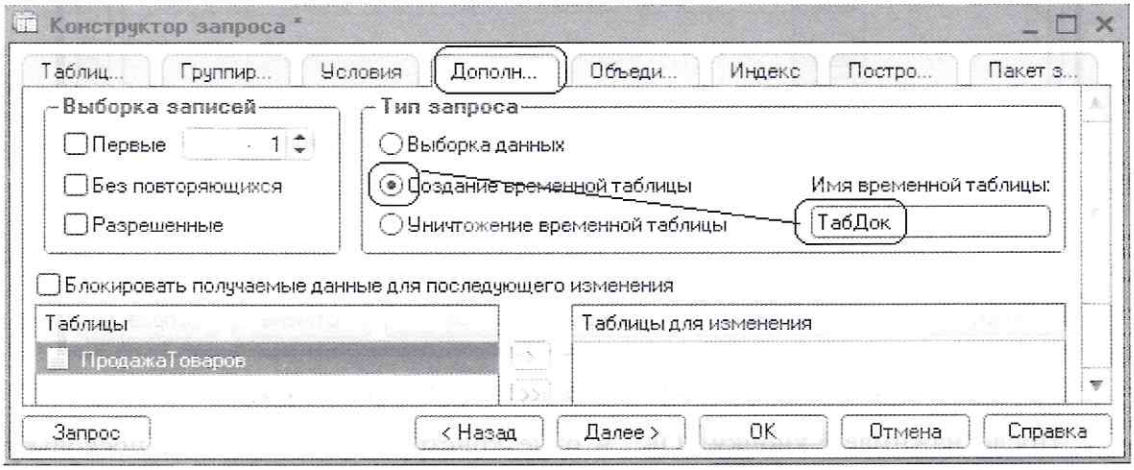


Рис. 3.23. Создание временной таблицы

Теперь можем переходить к очередному этапу сборки запроса, начиная его с добавления еще одного запроса в пакет.

Для этого открываем закладку "Пакет запросов". Видим, что пока в пакете только один запрос. Чтобы добавить еще один, в верхнем левом углу нажимаем кнопку "Добавить".

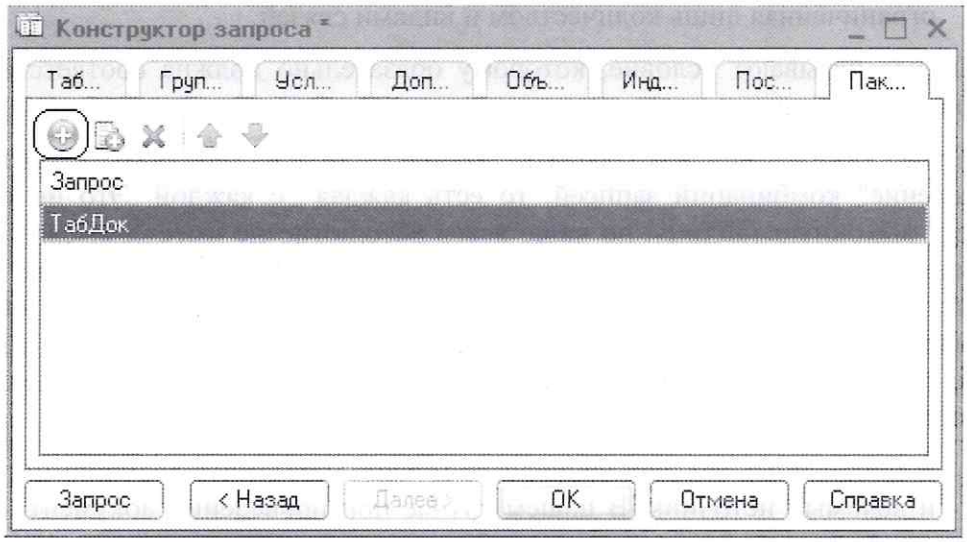


Рис. 3.24. Добавление запроса в пакет

Открывается окно конструктора запроса. Причем того же самого конструктора запроса. Просто по правой стороне окна видно, что мы сейчас находимся на закладке "Запрос пакета 2"

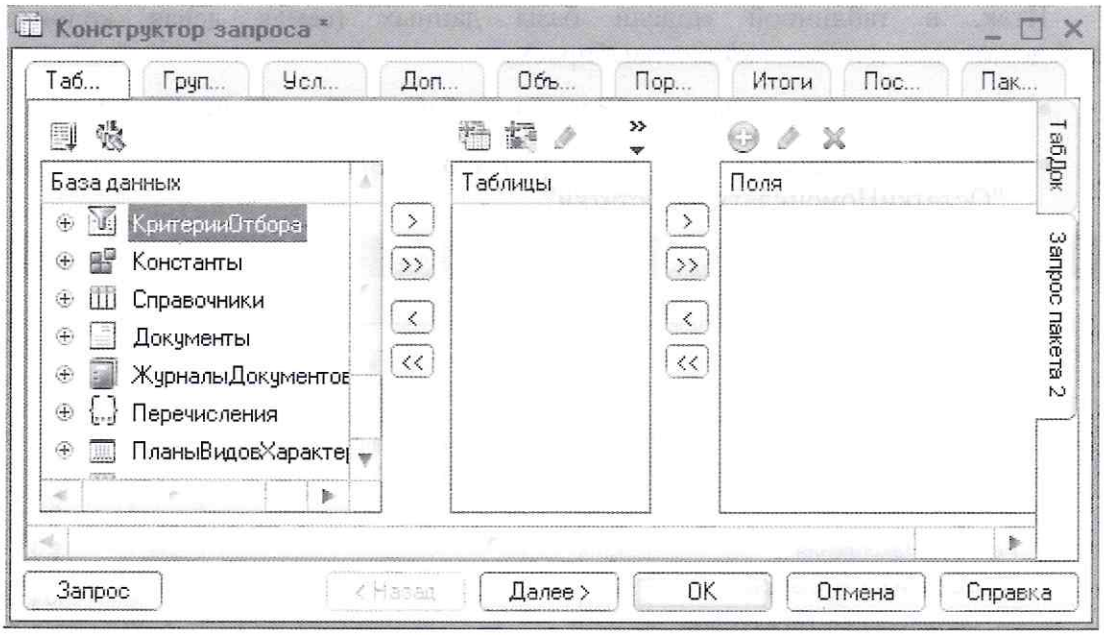


Рис. 3.25. Запрос пакета 2

Вот в этом- то "Запросе пакета 2" мы и реализуем сейчас соединение нашей временной таблицы "ТабДок" и виртуальной таблицы "РегистрНакопления.ОстаткиНоменклатуры.Остатки".

Важно!

С точки зрения техники выполнения запроса желательно помнить несколько эмпирических правил:

- Соединение – это, по сути, попытка в одном запросе получить все возможные комбинации записей исходных таблиц источников, ограниченная лишь количеством и видами связей.

Связью называют условие, которому обязательно должна соответствовать очередная комбинация.

Если связей не накладывать вообще, будем получать "декартово произведение" комбинаций записей, то есть каждая с каждой. Это не только "раздует" выходную таблицу, но чаще всего абсолютно не имеет смысла. Зачем, например, нам в результате запроса строки, где 7 яблокам в документе будут соответствовать 3 карандаша на складе? Поэтому в прикладных задачах обычно выполняется следующее правило: "Минимальное количество связей на единицу меньше количества таблиц источников". То есть больше связей быть может, а вот меньше нельзя!

- В прикладных задачах чаще всего можно определить ведущий источник и ведомый источник. В нашем случае при проведении документа более важным источником является табличная часть нашего документа, данные же регистра просто прикрепляются к ней. Так вот: выходные поля группировочных сущностей (у нас - номенклатура) должны определяться из ведущих источников.

Итак, в табличной модели базы данных (самая левая колонка) последовательно двойным кликом выбираем:

- из самого нижнего раздела "ВременныеТаблицы" - таблицу "ТабДок";
- из раздела "РегистрыНакопления" - таблицу "ОстаткиНоменклатуры.Остатки".

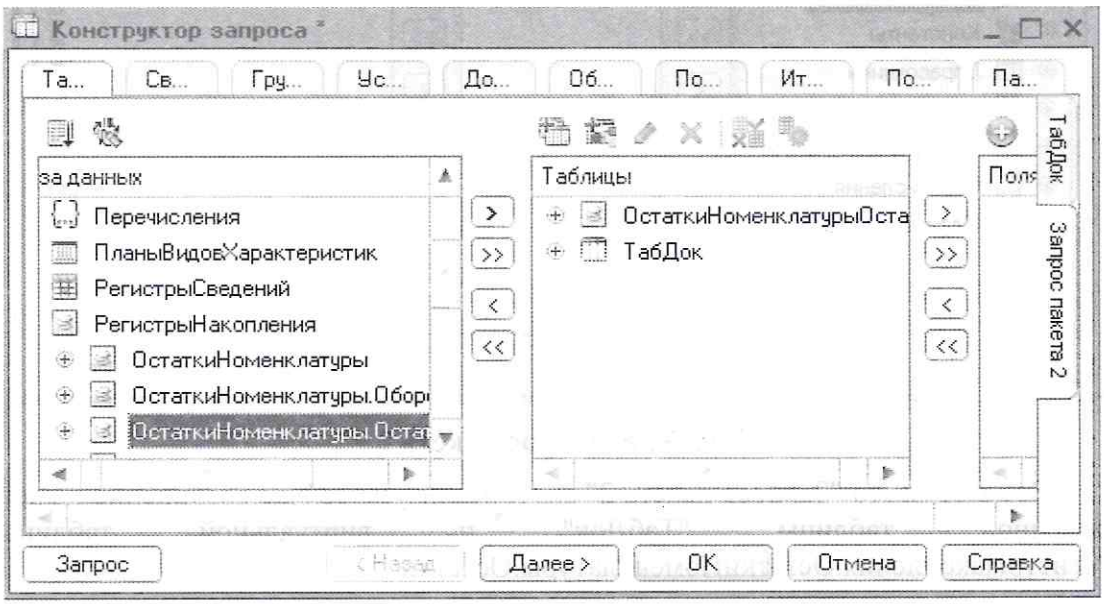


Рис. 3.26. Выбор таблиц-источников для реализации соединения

Следующим шагом должно быть указание значения для параметра "Период" виртуальной таблицы "ОстаткиНоменклатурыОстатки".

Важно!

При написании запроса, если только это не запрос для Системы Компоновки Данных (СКД) , важно привить себе следующий рефлекс: как только в качестве источника выбрана виртуальная таблица, надо сразу же войти в свойства виртуальной таблицы и указать имена всем внешним параметрам, связанным со временем построения виртуальной таблицы.

Объяснение рефлексу очень простое: если этого не сделать сразу, то можно забыть сделать потом. А ведь без указания значений таких параметров виртуальная таблица будет строиться с учетом абсолютно всех активных записей регистра, что чаще всего приводит к болезненным коллизиям в работе учетных механизмов. Например, нельзя же при перепроведении прошлогоднего документа считать себестоимость по актуальным данным. Надо использовать данные, полученные именно на момент документа.

Насчет исключения из этого правила СКД все очень просто. Важность указания временных параметров виртуальных таблиц при построении отчетов такая, что система сама, без напоминания, "достраивает" такие параметры. Так что тут исключение, только подтверждающее само правило!

В ситуации же, когда потом выясняется, что для Вашего алгоритма нужны именно данные, построенные с учетом всех активных записей регистра, недолго потом этот параметр из текста запроса убрать или перед выполнением запроса задать ему, например, значение "Неопределенно". Задача тогда будет решена именно так, как нужно.

Итак, этот рефлекс просто помогает заранее предотвращать потенциально возможную ошибку.

Активизируем таблицу-источник "ОстаткиНоменклатурыОстатки" и нажмём кнопку "Параметры виртуальной таблицы".

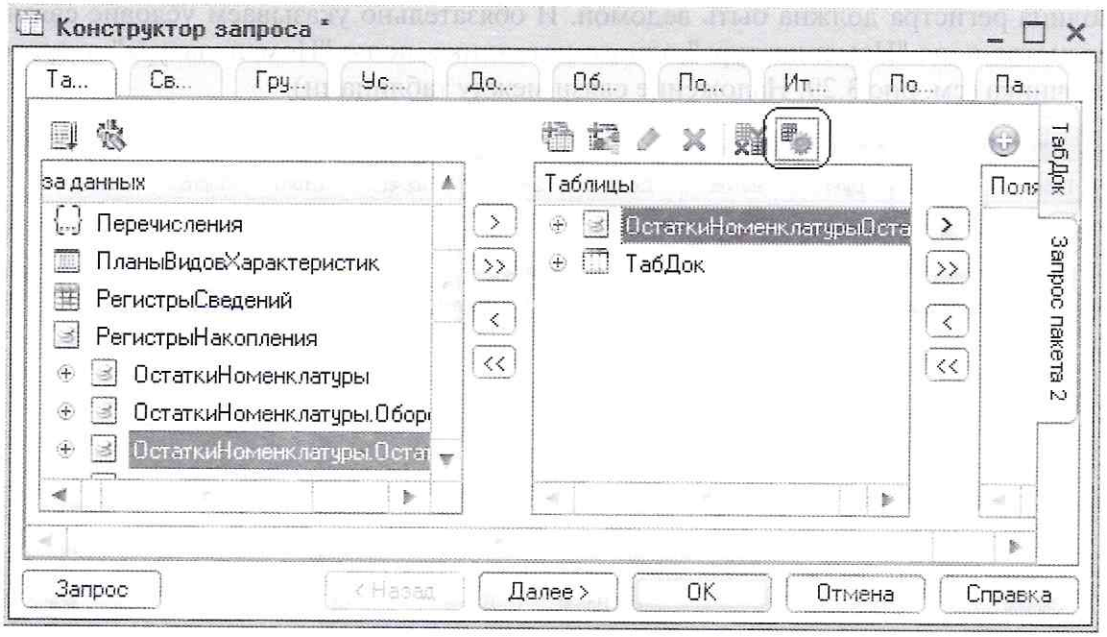


Рис. 3.27. Кнопка "Параметры виртуальной таблицы"

В открывшемся окне "Параметры виртуальной таблицы" вписываем в поле "Период" вызов внешнего параметра. Хотя в общем случае имя параметра совершенно неважно, но в "воспитательных" целях впишем "&Момент". Впоследствии перед выполнением запроса при установке значения этому параметру важно будет передать именно момент времени документа, а не его дату.

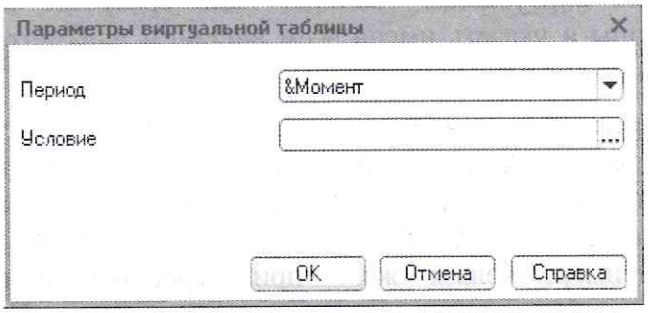


Рис. 3.28. Указание параметра, связанного со временем формирования виртуальной таблицы

После этого можно нажимать кнопку "ОК" в окне редактирования параметров виртуальной таблицы.

Далее приступаем к наложению связей при соединении наших таблиц-источников. С прикладной точки зрения временную таблицу "ТабДок" можно считать ведущим источником, а виртуальную таблицу остатков – ведомым. Ведь на схеме запроса четко видно, что если в документе товар есть, а в остатках его не оказалось, то строка по этой номенклатурной позиции должна присутствовать в выходной таблице запроса. И наоборот , если какой-то товар в остатках есть, но в документе не указан , то для нашей задачи информация по такому товару в выходной таблице запроса излишняя.

Итак, открываем закладку "Связи", добавляем новую связь нажатием кнопки "Добавить". В качестве значения поля "Таблица 1" указываем "ТабДок", ставим флажок "Все" для этой таблицы, в качестве значения поля "Таблица 2" указываем "ОстаткиНоменклатуры.Остатки". Тут как раз флажок "Все" не ставим! Ведь таблица регистра должна быть ведомой. И обязательно указываем условие связи: равенство поля "Номенклатура" одного источника полю "Номенклатура" второго источника (см. рис.3.29. Наложение связи между таблицами).

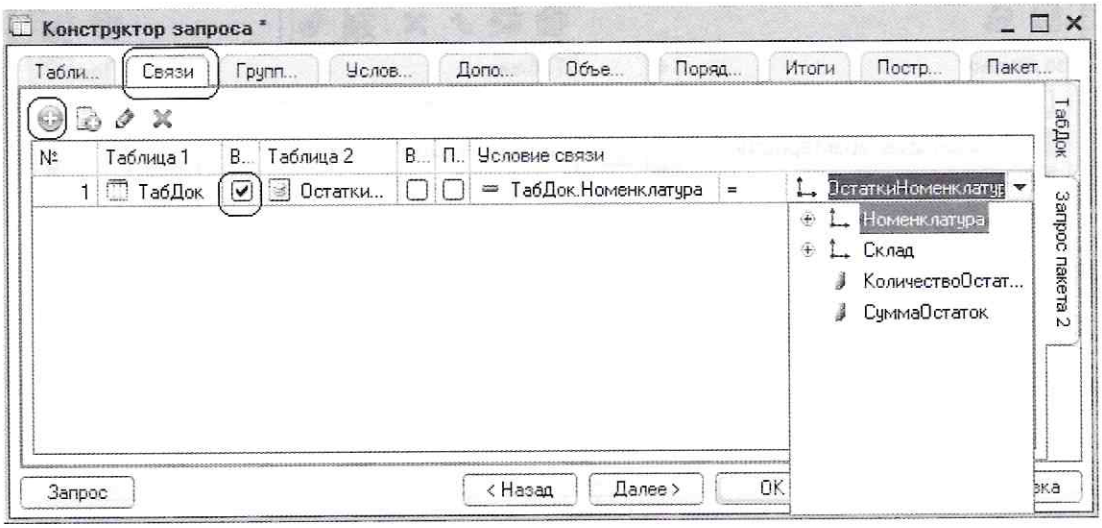


Рис. 3.29. Наложение связи между таблицами

Теперь можно возвращаться на первую закладку "Таблицы и поля" и согласно схеме выбирать три поля из "ТабДок" и два поля из виртуальной таблицы регистра.

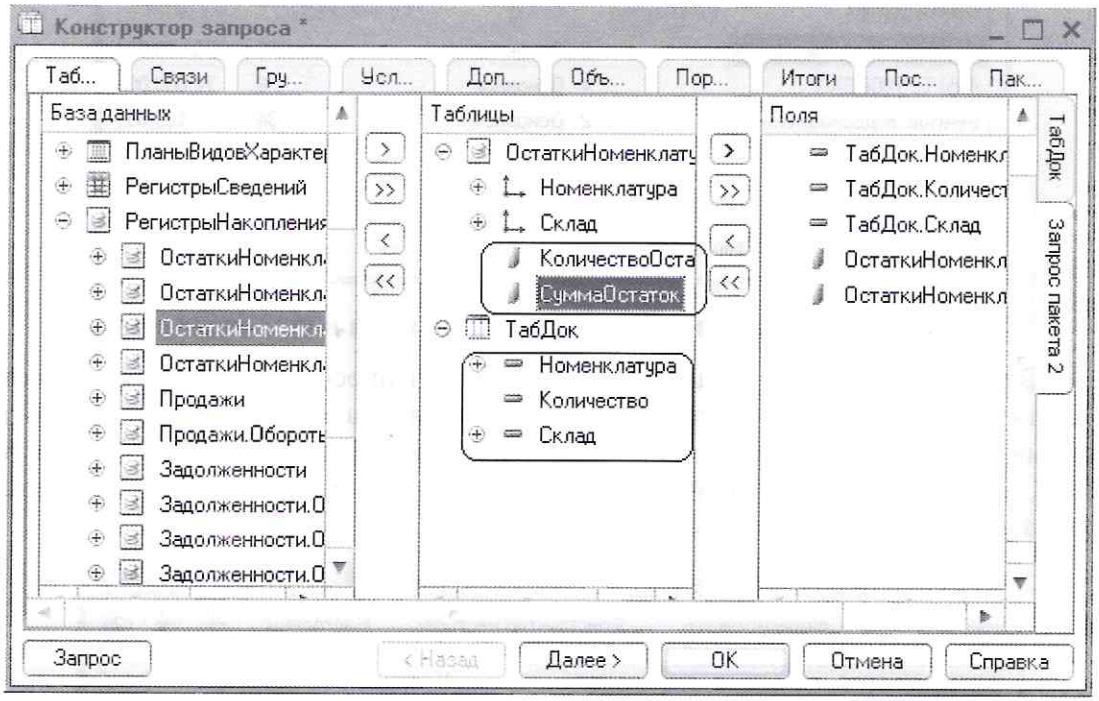


Рис. 3.30. Выбор выходных полей

Теперь можно нажимать кнопку "ОК".

Перед выполнением запроса не забудем про то, что в запросе добавился параметр для виртуальной таблицы регистра. То есть в окне конструктора запроса нажимаем кнопку "&Параметры", в открывшемся окне нажимаем кнопку "Получить из запроса" и заполняем значение параметра "Момент" каким-нибудь значением. Например, сегодняшней датой. Почему "каким-нибудь"? Дело в том, что хронологическая точность получения данных запросом нам будет важна уже при применении этого запроса. А сейчас наша цель – получить только текст запроса. Потому заполняем какой-нибудь датой, когда остатки на складе точно были. Кстати, можно было бы и не выбирать само значение даты, тогда виртуальная таблица сообразила бы, что нас интересуют актуальные остатки.

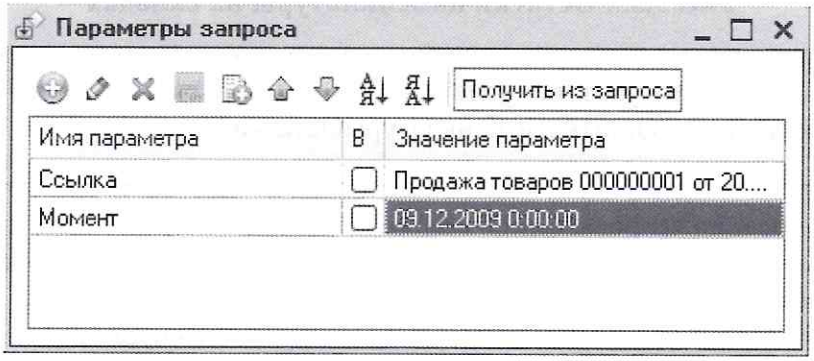


Рис. 3.31. Заполнение внешних параметров

После выполнения запроса получаем результат.

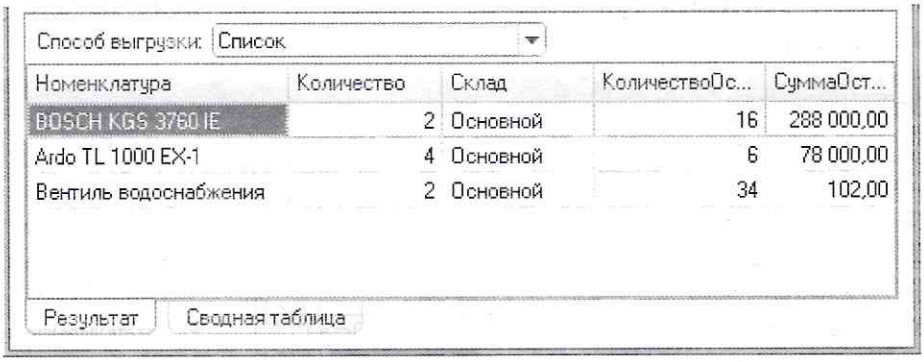


Рис. 3.32. Результат запроса после реализации в нем соединения

Однако, сравнивая данные, полученные запросом, с данными отчета "Остатки номенклатуры", можно обнаружить, что в наш запрос закралась ошибка. Списывать- то мы будем с того склада, который указан в накладной. Тем не менее, судя по товару "BOSCH KGS 3760 IE", сейчас наш запрос пытается получать общие остатки товаров на всех складах.

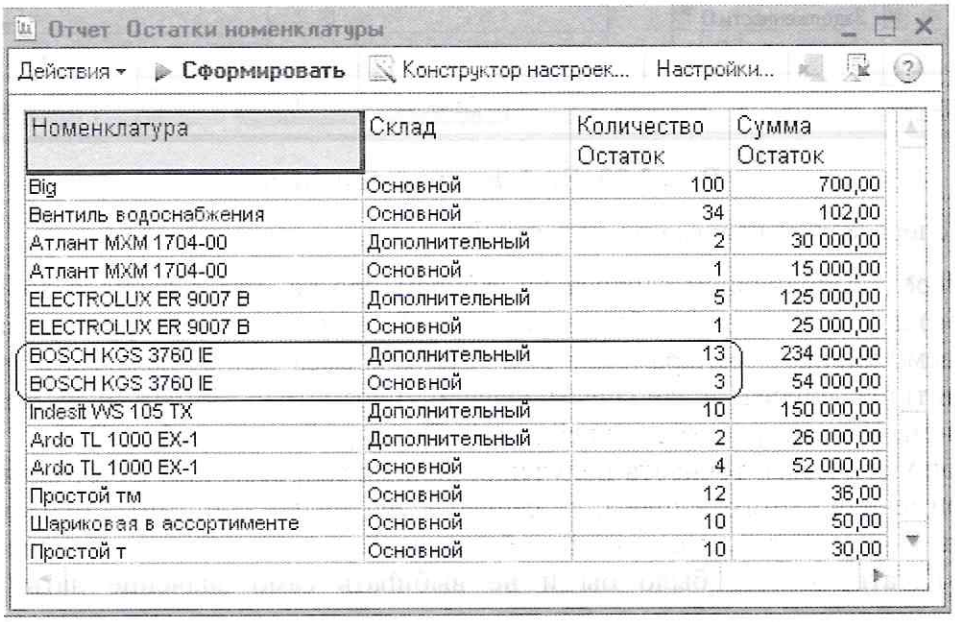


Рис. 3.33. Остатки номенклатуры на складах

Возвращаемся в конструктор запроса и пытаемся исправить ошибку. Для этого можно просто указать условие отбора по складу при построении виртуальной таблицы "ОстаткиНоменклатурыОстатки".

Важно!

При написании запроса, если в самом запросе необходимо наложить условие на виртуальную таблицу и условие касается отбора по значению (значениям) измерения (измерений),такое условие должно накладываться в параметрах виртуальной таблицы! Тогда это условие будет выполняться сразу, при построении виртуальной таблицы.

Попытка наложить УСЛОВИЕ по измерению ПОСЛЕ построения виртуальной таблицы – ГРУБЕЙШАЯ ОШИБКА с точки зрения быстродействия запроса.

Войдя в конструктор запроса, активизируем закладку "Запрос пакета 2", проверяем, что сейчас активна таблица "ОстаткиНоменклатурыОстатки", нажимаем кнопку редактирования параметров виртуальной таблицы. Для заполнения поля "Условие" лучше всего нажать кнопку выбора в этом поле.

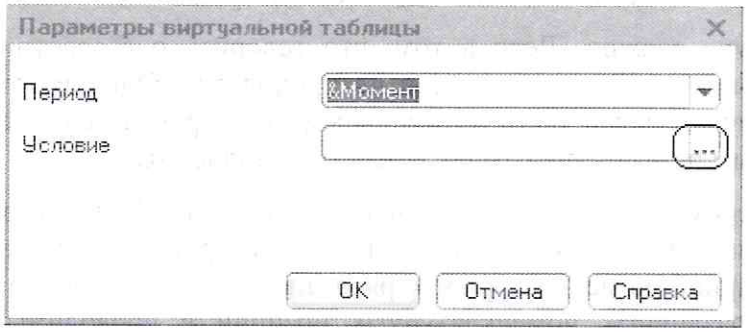


Рис. 3.34. Работа с параметрами виртуальной таблицы

Открывается знакомое уже нам окно "Произвольное выражение". В нижнюю часть окна перетаскиваем поле "Склад", а дальше дописываем выражение, чтобы получилось:

Склад = &Склад

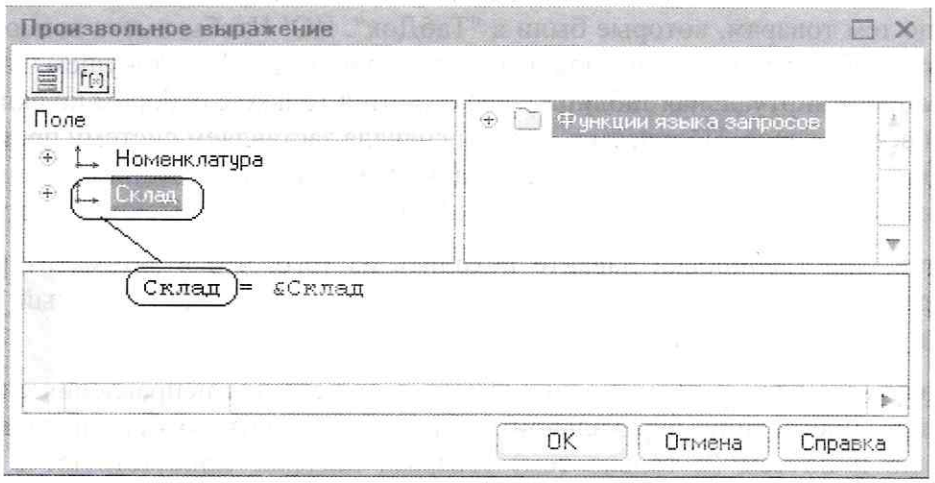


Рис. 3.35. Сборка условия по складу

Далее нажимаем кнопку "ОК" в окне "Произвольное выражение", потом нажимаем кнопку "ОК" в окне "Параметры виртуальной таблицы", потом нажимаем кнопку "ОК" в окне конструктора запроса.

После заполнения параметра "Склад" соответствующим значением (а именно, склад "Основной") можем выполнить запрос и получить желаемый результат.

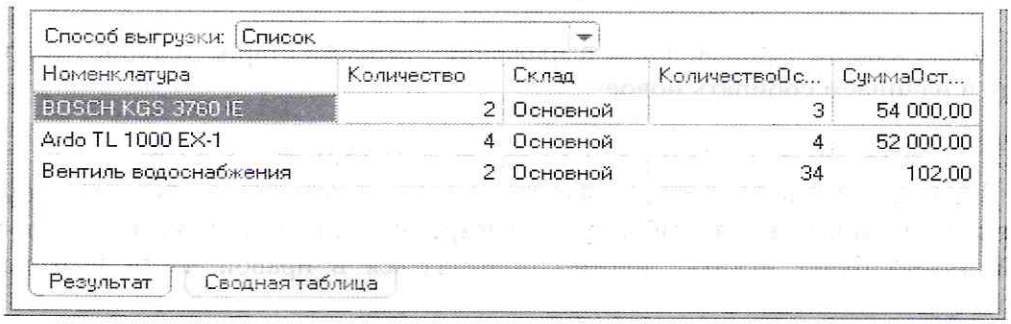


Рис. 3.36. Результат запроса с учетом отбора остатков по складу

Итак, мы вроде бы получили работающий текст запроса, который решает нашу задачу.

Но вспомним пункт седьмой технологии сборки алгоритма проведения документа – нужно обеспечить максимальное быстродействие нашего запроса.

А вот тут сложно. Дело в том, что говорить о стереотипных приемах обеспечения быстродействия запросов не приходится. Один и тот же запрос в отдельных случаях может быть оптимальным по скорости исполнения, а других случаях – самым неэффективным по скорости исполнения.

За примером далеко ходить не надо. Наш текущий запрос оптимален для такой ситуации: "Количество различных номенклатурных позиций на складе (еще говорят – ширина номенклатурного ряда на складе) примерно одинаково с количеством различных номенклатурных позиций в проводимых документах (шириной номенклатурного ряда на накладных)".

А вот для ситуации "На складе десятки тысяч различных товарных позиций, а в документах продажи обычно не больше десятка строк" наш текст запроса может потерпеть полное фиаско с точки зрения эффективности по скорости. Причина очень проста: да, за счет того, что ведущей у нас является таблица "ТабДок", после соединения в выходной таблице запроса будет информация только по тем товарам, которые были в "ТабДок". НО! Чтобы иметь возможность соединять таблицы, надо сначала иметь эти таблицы. А в текущем варианте текста запроса виртуальная таблица у нас строится только с отбором по складу, то есть для всех товаров. Получается, что мы сначала заставляем систему произвести временный расчет на момент документа для всех товарных позиций на складе , а потом за счет соединения отказываемся от 99,999% уже рассчитанных итогов.

Итак, для тренировки давайте исходить из того, что в автоматизируемом нами предприятии именно такая ситуация: "широкий номенклатурный ряд на складе и узкий – в документах".

Получается, что с точки зрения здравого смысла для исправления ситуации нужно при построении виртуальной таблицы применить отбор не только по складу, но и по тем (и только тем) товарам, которые присутствуют в нашем документе, то есть отбор должен идти сразу по нескольким полям.

И такая возможность для виртуальной таблицы есть. Более того, тут-то мы и воспользуемся тем, что данные, нужные для построения отбора ,уже лежат во временной таблице.

Заходим в конструктор запроса, активизируем закладку "Запрос пакета 2", проверяем, что сейчас активна таблица "ОстаткиНоменклатурыОстатки", нажимаем кнопку редактирования параметров виртуальной таблицы. Опять нажимаем кнопку выбора у поля "Условие".

Попав в окно "Произвольное выражение", стираем старое условие отбора по складу и начинаем собирать новое:

(Номенклатура, Склад) В ( )

В левой части выражения перечислены поля, с отбором по которым нужно будет строить виртуальную таблицу в регистре. А какие именно комбинации этих полей нужно будет искать, должно описываться в правой части выражения, внутри круглых скобок.

Легче всего вот эту правую часть выражения собрать конструктором вложенного запроса по нашей временной таблице. Реализуется это следующим способом. Между правых круглых скобок набираем несколько пробелов, чтобы была возможность эти пробелы выделить мышкой (хотя бы часть из них), потом в области выделенных пробелов вызвать контекстное меню и выбрать пункт "Конструктор запроса" (см. рис. 3.37.Вызов конструктора вложенного запроса.).

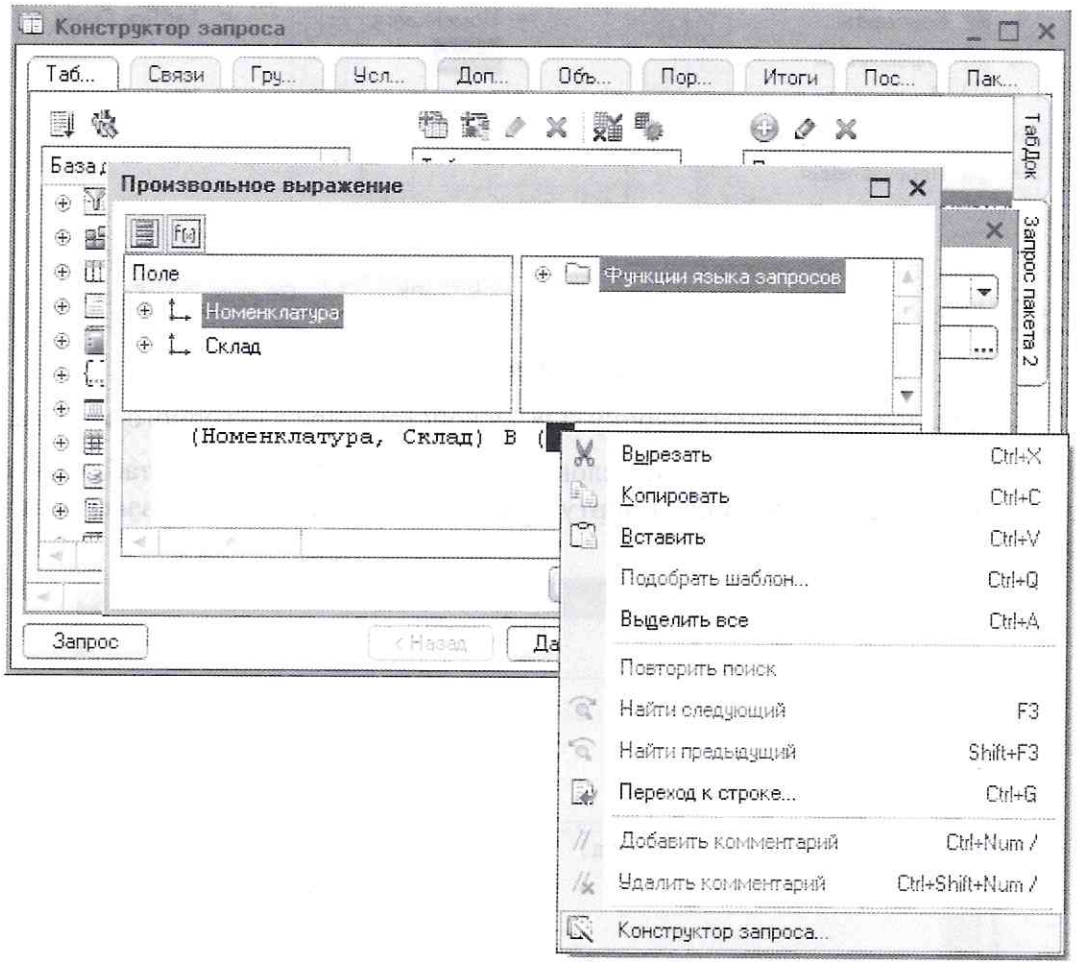


Рис. 3.37.Вызов конструктора вложенного запроса

Поскольку сейчас открылся конструктор вложенного запроса, в качестве источника можем выбрать временную таблицу "ТабДок".

Далее выбираем из таблицы-источника "ТабДок" оба поля ("Номенклатура" и "Склад") в качестве выходных.

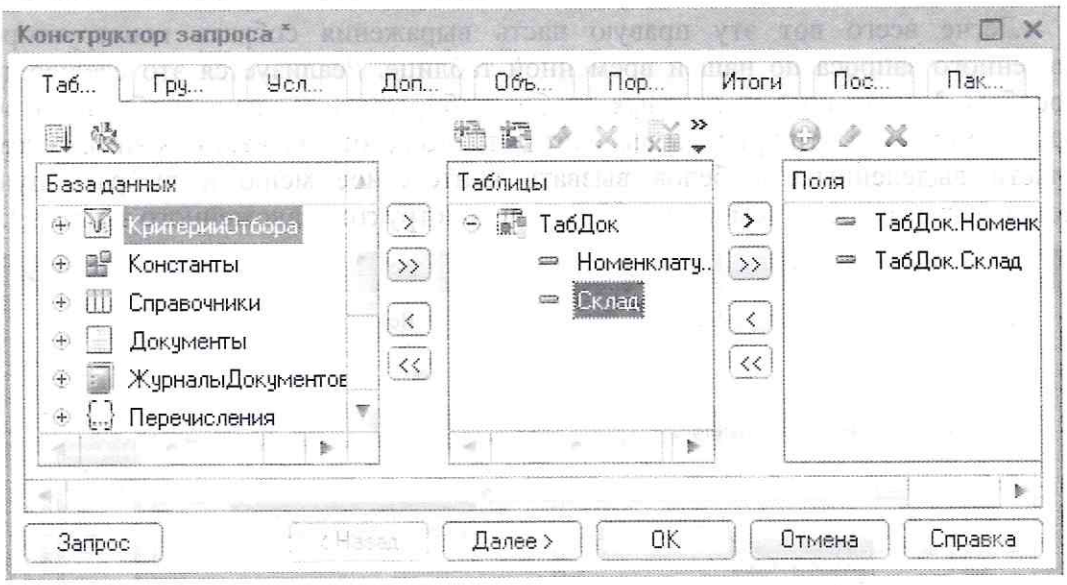


Рис. 3.40. Вложенный запрос

Теперь можно нажимать кнопку "ОК" в конструкторе вложенного запроса.

Система вернет нас в окно условий построения виртуальной таблицы. Но только само условие построения виртуальной таблицы примет уже законченный вид.

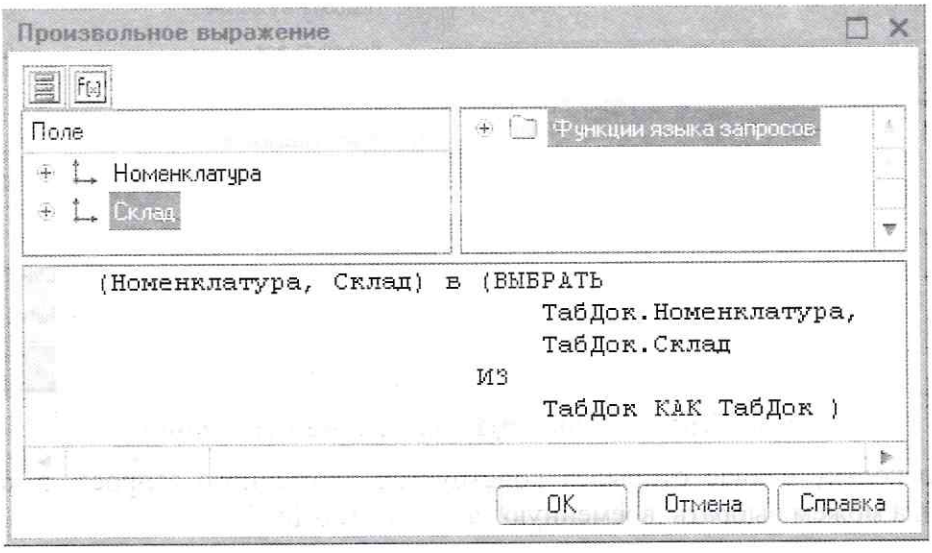


Рис. 3.41. Условие построения виртуальной таблицы

Теперь можно нажимать кнопку "ОК" последовательно во всех окнах, включая окно конструктора запроса.

Выполнив полученный запрос , получим необходимый результат.

А для максимального быстродействия в условиях проведения "больших накладных" еще хорошо бы построить индекс во временной таблице "ТабДок" именно по полям отбора, то есть по полям "Номенклатура" и "Склад". Тогда система будет еще быстрее строить виртуальную таблицу по нашему условию.

Для этого возвращаемся в конструктор запроса. В запросе "ТабДок" открываем закладку "Индекс" двойным кликом выбираем поля для индекса: "Номенклатура" и "Склад".

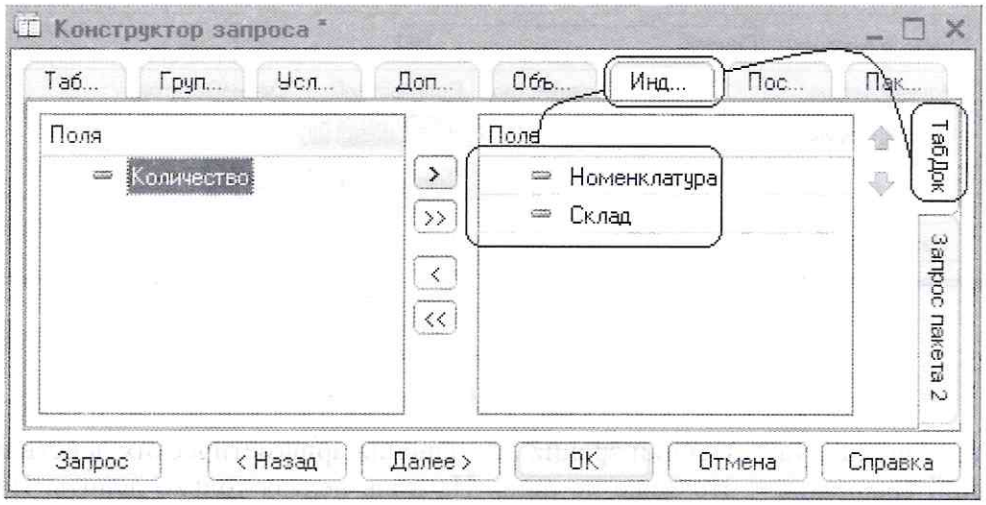


Рис. 3.42. Создание индекса для временной таблицы

Теперь можно нажимать кнопку "ОК" в окне конструктора запроса.

После того как убедились, что запрос работает и после обеспечения его максимального быстродействия , есть еще одна вещь, которую нужно будет обеспечить.

Помните, на схеме запроса мы отражали ситуацию, когда некий товар в документе есть, а в остатках этого товара нет. Желательно протестировать запрос и на такой ситуации. В демобазе для этого специально введен документ "Продажа товаров 000000004". Меняем в окне установки параметров значение ссылки на этот документ.

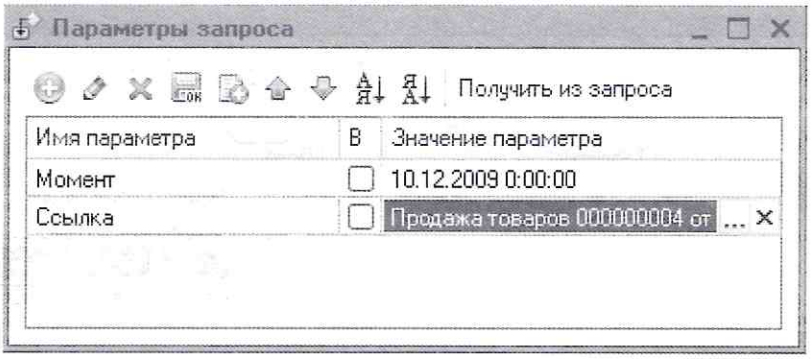


Рис. 3.43. Подбор ссылки на накладную с нехваткой товара

При выполнении запроса видим следующий результат:

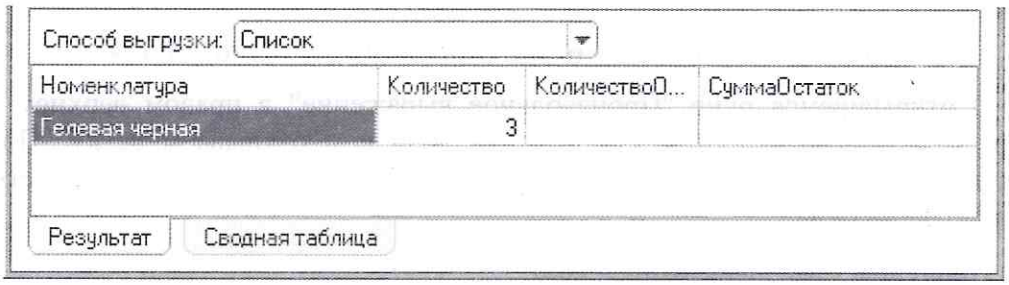


Рис. 3.44. Результат запроса по накладной с нехваткой товара

Если сейчас в строке с товаром "Гелевая черная" сделать двойной клик в поле "КоличествоОстаток" или "СуммаОстаток", то увидим, что в этих полях лежат не нулевые значения, а значения Null.

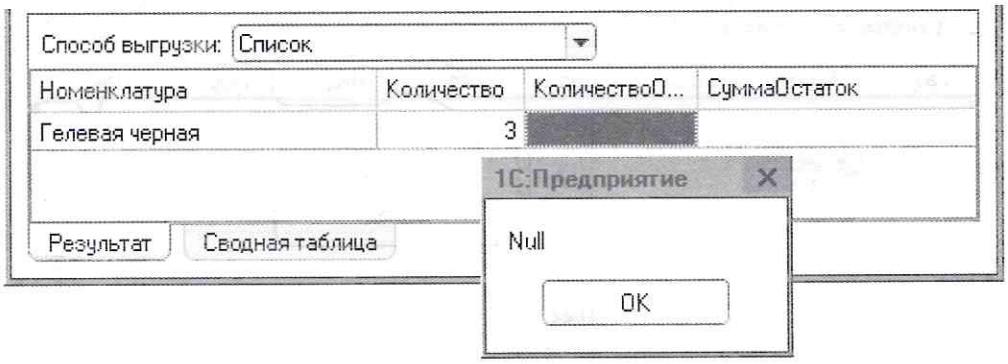


Рис.3.45. Null-значения

А значение Null с точки зрения выполнения арифметических действий или операций сравнения – это даже не ноль. На ноль нельзя только делить. На Null нельзя… ничего. Null – это не данные, это отсутствие данных.

Поэтому, поскольку запрос у нас не самоцель, а средство реализации последующих дополнительных действий в алгоритме, а среди этих действий есть и сравнение, и вычисления с полями "КоличествоОстаток" и "СуммаОстаток", то желательно все возможные Null значения прямо в запросе превращать в нули.

Для этого можно применить специальную функцию "ЕстьNULL".

Возвращаемся в конструктор запроса. Переходим на закладку "Запрос пакета 2", активизируем первое из нуждающихся в такой обработке полей (ОстаткиНоменклатурыОстатки.КоличествоОстаток) и нажимаем кнопку "Изменить текущий элемент" над столбцом выходных полей.

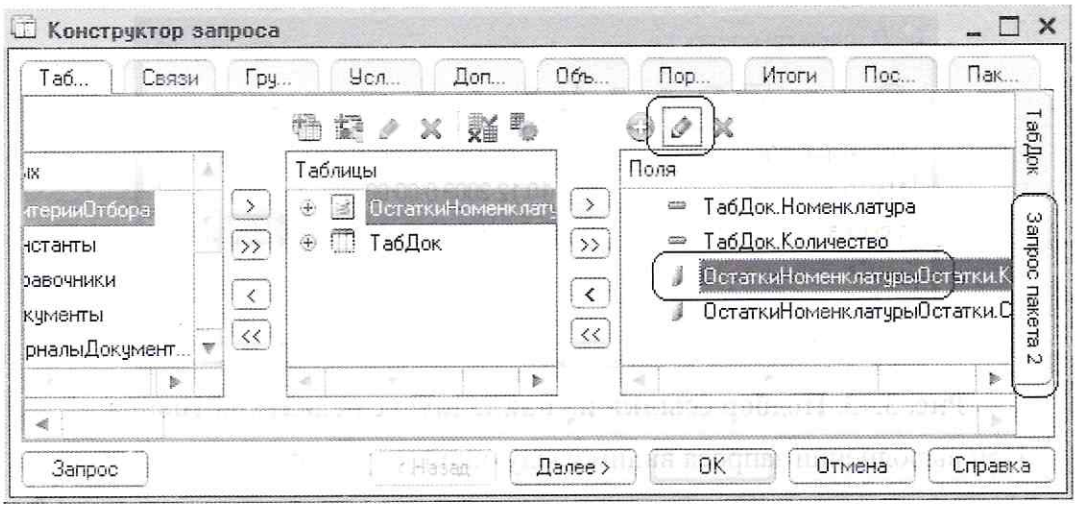


Рис. 3.46. Вызов окна редактирования выходного поля

В открывшемся окне "Произвольное выражение" в правом верхнем углу раскрываем дерево функций языка запросов, в нем открываем ветку "Прочие функции", перетаскиваем мышкой функцию "ЕСТЬNULL" из этой ветки в нижнюю часть окна.

Приводим само выражение в вид:

ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0)

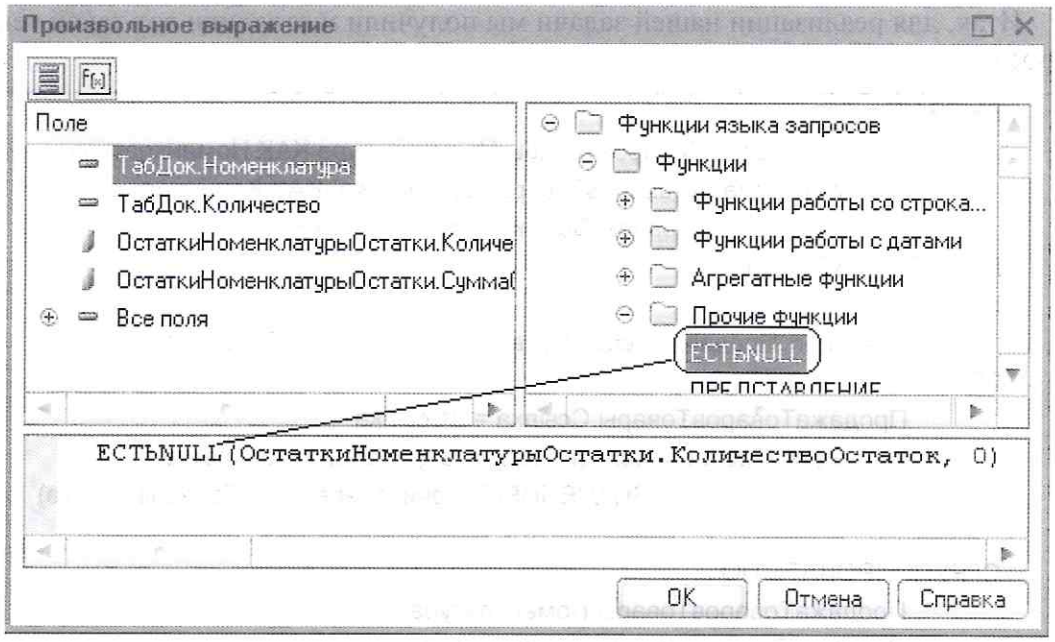


Рис. 3.47. Применение функции для выходного поля

Теперь можно нажимать кнопку "ОК".

Аналогично надо поступить для поля "ОстаткиНоменклатурыОстатки.СуммаОстаток".

На некоторых релизах платформы конструктор запроса после внесения таких изменений в формирование выходного поля может "потерять" старый псевдоним этого поля.

Для проверки этого лучше открыть в этом же запросе закладку "Объединения/Псевдонимы". Если название у полей сменилось (см. рис.3.48.Псевдонимы измененных полей), то надо двойным кликом войти в их редактирование и написать нужные нам.

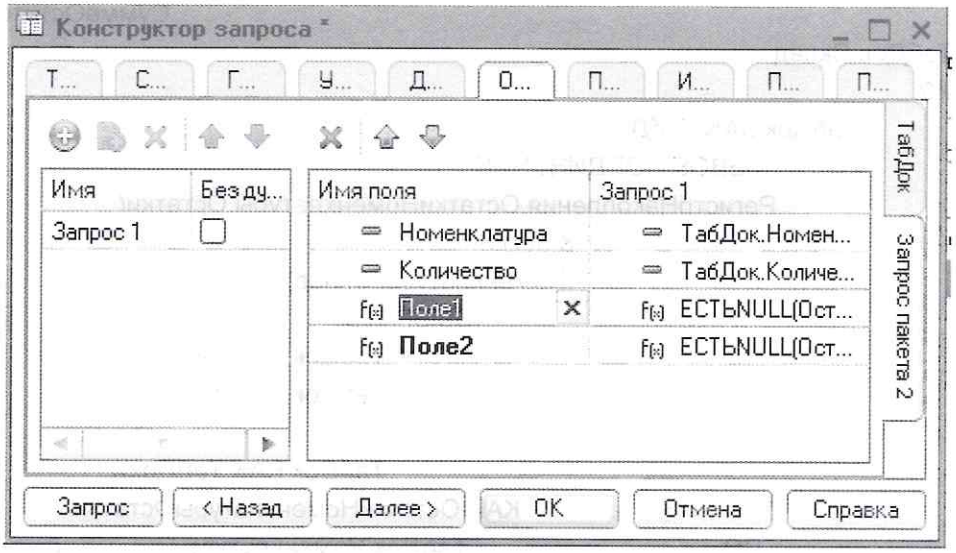


Рис. 3.48. Псевдонимы измененных полей

После этого можно нажимать кнопку "ОК" в конструкторе запроса.

Итак, для реализации нашей задачи мы получили и отладили вот такой текст запроса:

```
ВЫБРАТЬ
	ПродажаТоваровТовары.Номенклатура КАК Номенклатура,
	СУММА(ПродажаТоваровТовары.Количество) КАК Количество,
	ПродажаТоваровТовары.Ссылка.Склад КАК Склад
ПОМЕСТИТЬ ТабДок
ИЗ
	Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
ГДЕ
	ПродажаТоваровТовары.Ссылка = &Ссылка
	И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры <>
		ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)

СГРУППИРОВАТЬ ПО
	ПродажаТоваровТовары.Номенклатура,
	ПродажаТоваровТовары.Ссылка.Склад

ИНДЕКСИРОВАТЬ ПО
	Номенклатура,
	Склад
;
////////////////////////////////////////////////////////////////////////////////
ВЫБРАТЬ
	ТабДок.Номенклатура,
	ТабДок.Количество,
	ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) КАК
		КоличествоОстаток,
	ЕСТЬNULL(ОстаткиНоменклатурыОстатки.СуммаОстаток, 0) КАК
		СуммаОстаток,
ТабДок.Склад
ИЗ
	ТабДок КАК ТабДок
		ЛЕВОЕ СОЕДИНЕНИЕ
		РегистрНакопления.ОстаткиНоменклатуры.Остатки(
			&Момент,
			(Номенклатура, Склад) В
				(ВЫБРАТЬ
					ТабДок.Номенклатура,
					ТабДок.Склад
				ИЗ
					ТабДок КАК ТабДок))
			КАК ОстаткиНоменклатурыОстатки
		ПО ТабДок.Номенклатура =
ОстаткиНоменклатурыОстатки.Номенклатура
```

Теперь нужно применить наш запрос в процедуре "ОбработкаПроведения".

Для этого можно копировать текст запроса в буфер, возвращаться в конфигуратор.

В процедуре ставим курсор под маркером "//!!! ПОЛУЧЕНИЕ ДАННЫХ ТОЛЬКО ПО ТОВАРАМ!!!", вызываем контекстное меню и выбираем пункт "Конструктор запроса с обработкой результат".

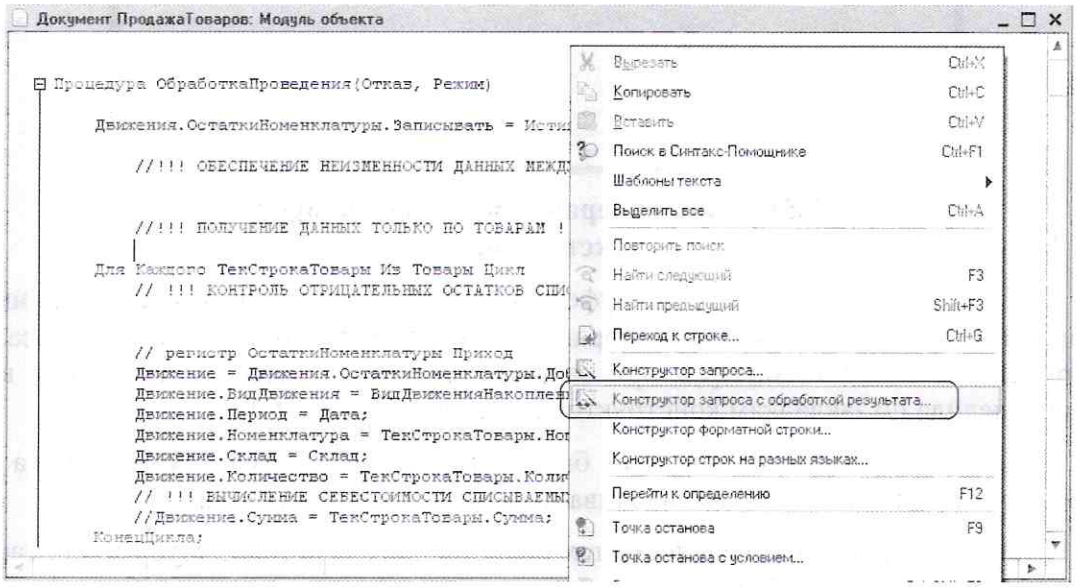


Рис. 3.49. Вызов конструктора запроса с обработкой результат

Система спросит нас, нужно ли создать новый запрос.

Ответим: "Да".

Открывается окно конструктора запроса. Мы хотим не собирать запрос заново, а просто назначить из буфера текст запроса конструктору. Для этого в левом нижнем углу окна нажимаем кнопку "Запрос".

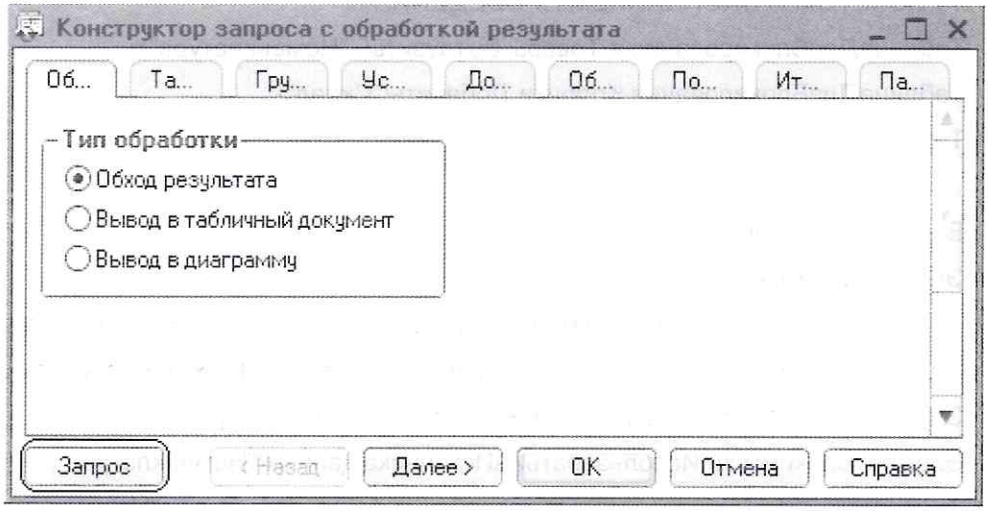


Рис. 3.50. Вызов окна интерактивного редактирования текста запроса

В открывшемся окне "Запрос" надо обязательно нажать единственную кнопку "Редактировать" в левом верхнем углу.

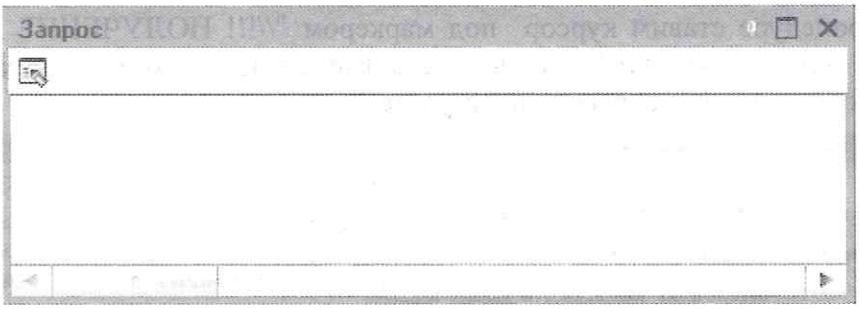


Рис. 3.51. Окно интерактивного редактирования текста запроса

Теперь можно вставить из буфера готовый текст запроса, отжать кнопку редактирования, закрыть окно интерактивного редактирования текста запроса, а вернувшись в окно конструктора запроса, убедиться, что система сама все распределила по закладкам конструктора и нажать кнопку "ОК".

Далее "сращиваем" то, что было получено конструктором запроса и конструктором движений, устанавливаем значения параметров запроса.

Реализуем функциональность тех недостающих действий, которые описаны в виде маркеров в нашей процедуре.

В конечном итоге процедура приобретает следующий вид:

```
Процедура ОбработкаПроведения(Отказ, Режим)
    Движения.ОстаткиНоменклатуры.Записывать = Истина;

    //!!! ОБЕСПЕЧЕНИЕ НЕИЗМЕННОСТИ ДАННЫХ МЕЖДУ
    // РАСЧЕТОМ И ОКОНЧАНИЕМ ПРОВЕДЕНИЯ !!!
    ТаблицаДляБлокирования = Товары.Выгрузить( ,"Номенклатура");
    ТаблицаДляБлокирования.Колонки.Добавить("Склад");
    ТаблицаДляБлокирования.ЗаполнитьЗначения(Склад,"Склад");

    Блокировка = Новый БлокировкаДанных;
    ЭлементБлокировки =
            Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
    ЭлементБлокировки.Режим         = РежимБлокировкиДанных.Исключительный;
    ЭлементБлокировки.ИсточникДанных = ТаблицаДляБлокирования;
    ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Номенклатура",
                                                    "Номенклатура");
    ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Склад", "Склад");
    Блокировка.Заблокировать();

    //!!! УДАЛЕНИЕ СОБСТВЕННЫХ СТАРЫХ ДВИЖЕНИЙ
    // ПО РЕГИСТРУ ОСТАТКИ НОМЕНКЛАТУРЫ !!!
    Движения.ОстаткиНоменклатуры.Записать();

    //!!! ПОЛУЧЕНИЕ ДАННЫХ ТОЛЬКО ПО ТОВАРАМ !!!
```

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|	ПродажаТоваровТовары.Номенклатура КАК Номенклатура,
|	СУММА(ПродажаТоваровТовары.Количество) КАК Количество,
|	ПродажаТоваровТовары.Ссылка.Склад КАК Склад
|ПОМЕСТИТЬ ТабДок
|ИЗ
|	Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
|ГДЕ
|	ПродажаТоваровТовары.Ссылка = &Ссылка
|	И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры <>
|			ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)
|
|СГРУППИРОВАТЬ ПО
|	ПродажаТоваровТовары.Номенклатура,
|	ПродажаТоваровТовары.Ссылка.Склад
|
|ИНДЕКСИРОВАТЬ ПО
|	Номенклатура,
|	Склад
|;
|
|////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|	ТабДок.Номенклатура,
|	ТабДок.Количество,
|	ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0)
|			КАК КоличествоОстаток,
|	ЕСТЬNULL(ОстаткиНоменклатурыОстатки.СуммаОстаток, 0)
|			КАК СуммаОстаток,
|	ТабДок.Склад
|ИЗ
|	ТабДок КАК ТабДок
|		ЛЕВОЕ СОЕДИНЕНИЕ
|			РегистрНакопления.ОстаткиНоменклатуры.Остатки(
|			&Момент,
|			(Номенклатура, Склад) В
|				(ВЫБРАТЬ
|					ТабДок.Номенклатура,
|					ТабДок.Склад
```

```
|                                        ИЗ
|                                              ТабДок КАК ТабДок))
|                                             КАК ОстаткиНоменклатурыОстатки
|                ПО ТабДок.Номенклатура =
|                                    ОстаткиНоменклатурыОстатки.Номенклатура";

Если Режим = РежимПроведенияДокумента.Оперативный Тогда                    //1
        Запрос.УстановитьПараметр("Момент", Неопределено);
Иначе
        Запрос.УстановитьПараметр("Момент", МоментВремени());
КонецЕсли;
Запрос.УстановитьПараметр("Ссылка", Ссылка);

Результат = Запрос.Выполнить();

ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

        // регистр ОстаткиНоменклатуры Расход
        Движение = Движения.ОстаткиНоменклатуры.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
        Движение.Склад =  ВыборкаДетальныеЗаписи.Склад;
        Движение.Количество = ВыборкаДетальныеЗаписи.Количество;

        // !!! КОНТРОЛЬ ОТРИЦАТЕЛЬНЫХ ОСТАТКОВ
        //                              СПИСЫВАЕМЫХ ТОВАРОВ !!!
        // !!! ВЫЧИСЛЕНИЕ СЕБЕСТОИМОСТИ СПИСЫВАЕМЫХ ТОВАРОВ!!!
        Если ВыборкаДетальныеЗаписи.КоличествоОстаток <
                                ВыборкаДетальныеЗаписи.Количество Тогда
                Отказ = Истина;                                            //2
                Сообщение = Новый СообщениеПользователю;
                Нехватка =  ВыборкаДетальныеЗаписи.Количество -
                                ВыборкаДетальныеЗаписи.КоличествоОстаток;
                Сообщение.Текст = "В документе N" + Номер +" от "+Дата+    //3
                                " не хватает  "+Нехватка+" единиц товара "+
                                ВыборкаДетальныеЗаписи.Номенклатура;
                Сообщение.Сообщить();
        ИначеЕсли ВыборкаДетальныеЗаписи.КоличествоОстаток =
```

```
                                    ВыборкаДетальныеЗаписи.Количество  Тогда  //4
                        Движение.Сумма = ВыборкаДетальныеЗаписи.СуммаОстаток;
                ИначеЕсли ВыборкаДетальныеЗаписи.КоличествоОстаток >
                                        ВыборкаДетальныеЗаписи.Количество Тогда
                        Движение.Сумма = ВыборкаДетальныеЗаписи.СуммаОстаток/
                        ВыборкаДетальныеЗаписи.КоличествоОстаток*
                        ВыборкаДетальныеЗаписи.Количество;
                КонецЕсли;

        КонецЦикла;

        Движения.Задолженности.Записывать = Истина;
        // регистр Задолженности Приход
        Движение = Движения.Задолженности.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Период = Дата;
        Движение.Контрагент = Контрагент;
        Движение.СуммаДолга = СуммаДокумента;

        Движения.БронированиеТоваров.Записывать = Истина;

КонецПроцедуры
```

Поскольку текст процедуры мы собирали именно под нашу задачу, подробно обсуждая каждое действие в процессе реализации, развернутое комментирование того, что получилось, было бы излишним.

Кроме того, наложение блокировок на таблицы регистра – тема, заслуживающая отдельного внимания (см. раздел 3.4 Блокировка записей регистров). Поэтому сейчас с этим разбираться не будем.

Сейчас хотелось бы обратить внимание на другие моменты:

//1 Оперативно проводимый документ, по идее, занимает последнее положение на временной оси (подробнее об этом ниже – Раздел 3.3 Оперативное и неоперативное проведение). Значит, для такого документа вполне адекватны и актуальные (самые последние) итоги регистров. И в интересах быстродействия запроса важно строить виртуальную таблицу с указанием пустого временного параметра. А если документ проводится неоперативно, то в интересах точности расчета себестоимости нас должны интересовать остатки номенклатуры, взятые именно на момент проведения документа.

//2 Если мы убедились, что товара не хватает по какой-то из позиций, необходимо отменить проведение всего документа. Это легко достигается присвоением параметру процедуры (посмотрите на заголовок процедуры) "Отказ" значения "Истина". Причем ход выполнения данной процедуры тут не прерывается, то есть цикл по выходной таблице запроса будет "крутиться" и дальше, хотя уже ясно, что свое предназначение процедура не выполнит. Однако это может оказаться удобным. Поскольку мы не просто отменяем проведение

документа, но и предупреждаем, из-за каких товаров. В результате в окне сообщений будет информация не по одной товарной позиции, а по всем случаям нехватки.

//3 Если мы программно отказываемся проводить какой-то документ, обязательно надо сообщить пользователю не только почему, но и какой именно документ не проводим! Иначе при групповом проведении документов пользователь никогда в жизни не догадается, с каким из документов проблема.

//4 Поскольку при средневзвешенном способе мы пользуемся пропорцией, а пропорция – это всегда операция деления, а деление – это всегда угроза погрешностей (и "деления на ноль"), лучше подстраховаться, и для случая, когда списывается все количество из регистра, "вдогонку" списать всю оставшуюся себестоимость.

Практикум №9.1

При проведении документа мало контролировать только остатки товаров в регистре "ОстаткиНоменклатуры", необходимо еще учесть и данные регистра "БронированиеТоваров"!

Необходимо к выходной таблице запроса добавить поле "КоличествоЗабронированного", заполнив его данными по нужным товарам из регистра "БронированиеТоваров". И при определении возможности проведения документа это количество, естественно, должно увеличивать нехватку.

Итак, мы с Вами освоили технологию сборки алгоритмов проведения. На самом деле технология во многих моментах пригодна для решения и других задач, в которых необходимо выполнять чтение из базы данных и запись данных в нее. В этом можно убедиться при решении практических задач.

Практикум №9.2

Добавьте, пожалуйста, в оба механизма бронирования товаров (из объекта документа и снаружи) предварительный контроль возможности выполнения этой операции, то есть если на складе 100 шт. некого товара, а под другие документы уже забронировано 70 шт., то мы не можем бронировать более 30 шт.

Кроме того, услуги бронировать не нужно!

3.3. Оперативное и неоперативное проведение

Программа учета или управления практически всегда отражает реальность той предметной области, для которой она реализована. И, как мы уже обсуждали во введении, программа должна фиксировать (регистрировать, запоминать) те события из реальности, которые оказывают влияние на контролируемые показатели. А вот при регистрации событий всегда приходится сталкиваться с понятием времени.

Итак, для любого реального объекта существует три понятия времени: "прошлое", "настоящее" и "будущее". Понятий-то три, но на самом деле …

"Будущее" еще не наступило, и говорить о нем серьезно могут только люди с богатым воображением. "Прошлое" уже прошло, кончилось, не существует. И утверждать, что можно реально вернуться в него ,могут только люди с еще более богатым воображением.

Фактически получается, что для реальности существует только "настоящее", то есть реальные события происходят именно в реальном времени.

И идеальная система учета как отображение реальности стремится фиксировать события в момент их происхождения , то есть в реальном времени.

Именно поэтому в системе "1С:Предприятие 8" для документов существует возможность использовать оперативное проведение.

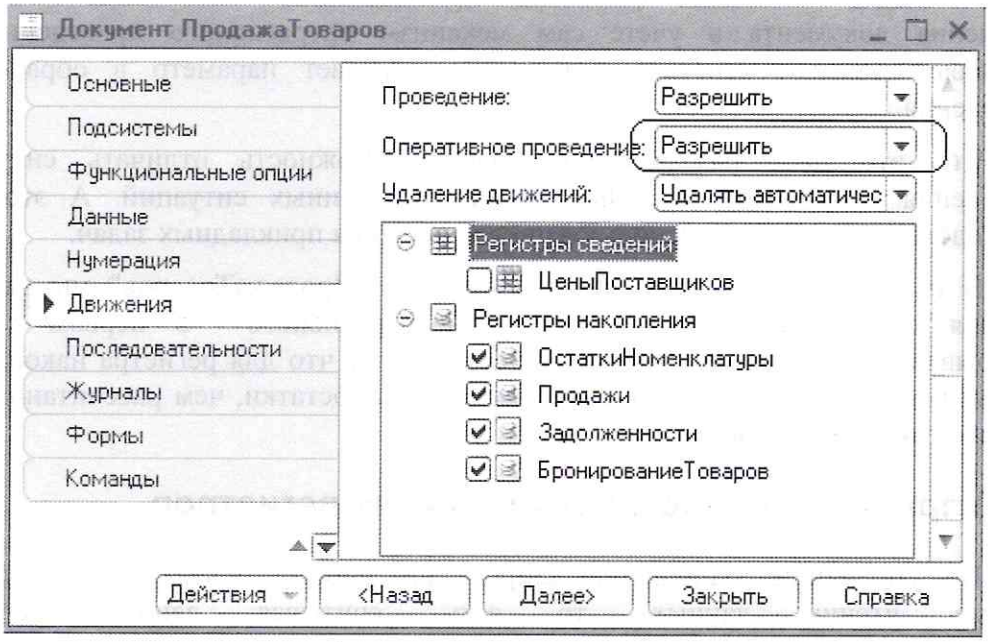


Рис. 3.52. Свойство "Оперативное проведение"

Рассмотрим, как работает этот механизм при интерактивном проведении из формы документа. Пусть документу, как объекту метаданных, разрешено оперативное проведение, и свойство "ИспользоватьРежимПроведения" расширения формы документа имеет значение "Автоматически".

При проведении такого документа система прежде всего проверит положение даты документа относительно системной даты.

Если дата документа соответствует дню позже системной даты, пользователь получит предупреждение: "Дата оперативно проводимого документа больше текущей. Документ не может быть проведен оперативно". Проведение при этом не состоится.

Если дата документа соответствует дню меньше системной даты , документ будет проведен "Неоперативно".

Если обе даты определяют один и тот же день, система принимает решение, что документ будет проведен "Оперативно", то есть реальным временем.

Для того чтобы документы располагались последовательно и не попадали внутрь одной секунды, т.к. внутри одной секунды порядок следования будет неопределен, система выполняет автоматическое изменение времени документа. Дата документа получает значение оперативной отметки времени, которая рассчитывается системой по определенному алгоритму: дата оперативно проводимого документа получается равной текущей дате сеанса или на секунду больше предыдущей выданной оперативной отметки времени.

Далее управление передается обработчику события "Обработка проведения", при этом параметр "Режим" получает значение "РежимПроведенияДокумента.Оперативный".

Итак, при оперативном проведении документов система позволяет добиться расстановки документов в идеальной хронологической последовательности (каждое событие зарегистрировано на момент его отражения в учете) и, кроме того, известить об этом обработчика проведения, то есть с точки зрения отражения документа в учете сам механизм оперативного проведения не выполняет никаких действий, он только передает параметр в обработчик проведения.

Что это дает разработчику? Дает возможность отличать ситуации проведения документов реальным временем от иных ситуаций. А это уже впоследствии может быть использовано для решения прикладных задач.

В нашем примере проведения документа "ПродажаТоваров" мы за счет отличия оперативного проведения от неоперативного в первом случае выигрываем скорость выполнения запроса, потому что для регистра накопления остатков гораздо быстрее получаются актуальные остатки, чем рассчитанные на произвольный момент времени.

3.4. Управляемая блокировка записей регистров

По умолчанию выполнение запроса по любому объекту происходит в режиме чтения данных, то есть, считывая данные регистра "ОстаткиНоменклатуры", мы совершенно не мешаем другим процессам обращаться к регистру также для чтения данных.

Но некоторые алгоритмы не могут при таком положении вещей корректно работать: если представить два параллельно проводимых документа, в которые пытаются конкурентно списывать товар со склада, то нормальной как раз является ситуация, когда какому-то документу товара не хватит, потому что более "удачливый" его коллега этот товар уже списал.

Кроме того, вполне возможна ситуация конкурентного проведения двух документов с разным временем, затрачиваемым на проведение (попросту один документ очень большой, а другой очень маленький). В такой ситуации важно, чтобы не получилось, что большой документ, только-только получив запросом данные, нужные для расчета себестоимости, уже начинает эти данные обрабатывать, а его более мелкий и юркий коллега взял, да и списал за эти доли секунды часть товара , то есть уже внес изменения учет товаров на складе. Ведь тогда даже если нашему "неповоротливому гиганту" товара реально хватает, то себестоимость он будет списывать рассчитанную по старым, уже не соответствующим действительности данным.

Поэтому важно бывает обеспечить неизменность части данных в БД от начала действия алгоритма до полного его завершения. Или еще говорят – наложить блокировку на часть данных.

При наложении блокировки важно опять же не перегнуть палку. Одно дело, продавая яблоко и грушу со склада "Основной", блокировать таблицы регистра только по яблоку и груше на складе "Основной". Другое – блокировать всю таблицу целиком. В первой ситуации мы будем придерживать конкурентов до окончания нашего алгоритма , только если у конкурента в документе есть яблоко

или груша и склад "Основной". Во второй ситуации мы будем придерживать вообще все документы, которые пытаются хоть что-то списать с какого угодно склада, то есть совершенно напрасно заблокируем работу вообще всех других пользователей, проводящих документы продажи до момента списания наших несчастных фруктов.

Сам по себе факт блокировки не летален, остальные сеансы просто выстраиваются в очередь и терпеливо дожидаются, когда же ресурсы освободятся (правда, если ожидание затянется и тайм-аут закончится, система "сдастся" и просто выдаст пользователю предупреждение о невозможности выполнения операции). Но ведь если в этих документов нет ни груш, ни яблок или склад совсем другой , можно ведь было уже давно такие документы провести и спокойно переходить к другой работе.

Итак, излишняя блокировка снижает параллельность работы пользователей и, как следствие , снижает всю эффективности системы.

Для тонкого управления этим вопросом - что именно блокировать – в рассмотренном выше алгоритме проведения документа "ПродажаТоваров" был применен вот такой фрагмент:

```
//!!! ОБЕСПЕЧЕНИЕ НЕИЗМЕННОСТИ ДАННЫХ МЕЖДУ
// РАСЧЕТОМ И ОКОНЧАНИЕМ ПРОВЕДЕНИЯ !!!
Блокировка = Новый БлокировкаДанных;                                    //1
ЭлементБлокировки =
        Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");   //2
ЭлементБлокировки.Режим  = РежимБлокировкиДанных.Исключительный;        //3
ЭлементБлокировки. УстановитьЗначение("Склад", Склад);                  //4
ЭлементБлокировки.ИсточникДанных =
                                Товары.Выгрузить( ,"Номенклатура");     //5
ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Номенклатура",
                                                "Номенклатура"); //6
Блокировка.Заблокировать();                                             //7
```

Комментарии к фрагменту:

//1 Создаем конструктором объект "БлокировкаДанных".

//2 В общем случае в объект "БлокировкаДанных" может входить несколько элементов, просто каждый элемент будет обслуживать блокировки записей своего объекта базы данных. Для нашего алгоритма важно блокировать только записи по регистру "ОстаткиНоменклатуры". Именно такой элемент мы и добавляем в блокировку.

//3 Устанавливаем режим блокировки. Нам важно, чтобы данные, которые будут заблокированы, не только не изменялись до конца транзакции (тогда можно было бы выбрать вариант "РежимБлокировкиДанных.Разделяемый"), но чтобы другие транзакции даже не начали читать эти заблокированные данные, пока мы не закончим с ними работать (ведь мы же своим алгоритмом изменим ту же себестоимость и т.п.), поэтому применяем вариант исключительной блокировки.

//4 Начинаем указывать , по каким именно значениям, каких полей регистра надо будет накладывать блокировку. Поскольку склад у нас в документе один, то отбор по складу накладываем методом "УстановитьЗначение".

//5 А вот номенклатурных позиций у нас в документе много. Поэтому применяем назначение источника данных нашему элементу блокировки. В качестве значения передаем таблицу значений, которую мы получаем выгрузкой из табличной части документа. Причем выгружаем только заполненную в документе колонку "Номенклатура".

//6 Синтаксис работы с источником данных блокировки ориентирован на различные ситуации и готов работать с данными типа <РезультатЗапроса>, <Табличная часть>, <ТаблицаЗначений>. Логично при этом указать, для какого поля блокируемого объекта (у нас - регистра) надо брать значения из какого поля источника данных (у нас – из таблицы значений).

//7 Все подготовительные этапы выполнены. Даем исполнительную команду на применение блокировки.

3.5. Возможные коллизии при проведении документов и борьба с ними. Объект "Последовательности"

Пока документы проводятся оперативно , наша система будет работать как часы. Если товар есть , документ "ПродажаТоваров" проводится. Если товара нет – не проводится…

Но рассмотрим ситуацию:

1 января – купили 10 яблок;

3 марта – продали 2 яблока.

Если проводить расходную накладную 4 марта – сколько яблок можем продать? Правильно – четыре!

А если … 4 января?

По данным программы в тот день на складе было 10 яблок. Значит, все 10 и могли быть тогда проданы! И будьте уверены, если Ваш пользователь может продать 10 яблок задним числом, он таки обязательно это сделает!

Кстати, если Вас ничего не смутило двумя абзацами выше, прежде чем приступать к освоению материала этой главы , лучше отдохните, а потом настройтесь на работу. 10-2 =8 , а не 4. Тем, кто подвох заметил, самая искренняя признательность за внимательность.

Продолжим. Принципиально то, что, проводя что-то задним числом, т.е. в прошлом, пользователь легко может нарушить логику формирования и проведения последующих документов.

Фактически эта тема – тема "Путешествий во времени". И обыграна она во многих фантастических произведениях. Основная мысль сводится к следующему: если слетать в прошлое и не дать своей бабушке встретиться со своим дедушкой, то непонятно, откуда взялся ты, если ты останешься; а если исчезнешь, то непонятно, кто тогда не даст встретиться бабушке с дедушкой?

Выход из коллизии обычно заключается в сохранении только одного из варианта миров – либо того, где никогда не встречались бабушка с дедушкой и возмутитель спокойствия на свет не появлялся, либо того, где возмутитель появлялся, но не мешал своим родственникам, а еще лучше вообще задним числом ничего не делал.

Пользователи в своих базах иногда порождают коллизии и более странные. Самое милое дело – удалить задним числом приход товара, который чуть позже весь был израсходован.

Так вот существует специальный объект, предназначенный для обеспечения проведения определенных документов в строгой хронологической последовательности. Он так и называется "Последовательности".

Если проводилось несколько документов задним числом и документы меняли данные в критичном к логике учета в системе регистре (регистрах), то коллизий следует ожидать уже с момента времени самого раннего (по положению в последовательности) такого документа. Именно на него системой автоматически устанавливается ***граница*** последовательности (ГП). Проведет пользователь еще более старый документ – ГП на него переустановится.

Чтобы исправить ситуацию, достаточно будет потом перепровести все документы, входящие в последовательность, начиная от момента ГП по настоящее (можно по рабочую дату, а можно по любую другую). Этим занимается специальная системная функция. При таком перепроведении, если правильно написаны процедуры "ОбработкаПроведения", система будет все перерасчитывать и при нехватке товара сообщать пользователям, а так же не давать проводить некорректные документы и т.д. Зато по окончании обработки останется в Вашей базе полностью отлаженный, логически целостный вариант последовательности документов.

Для того чтобы и в нашей конфигурации автоматически отслеживалась последовательность товарных документов, создаем последовательность "Товарная".

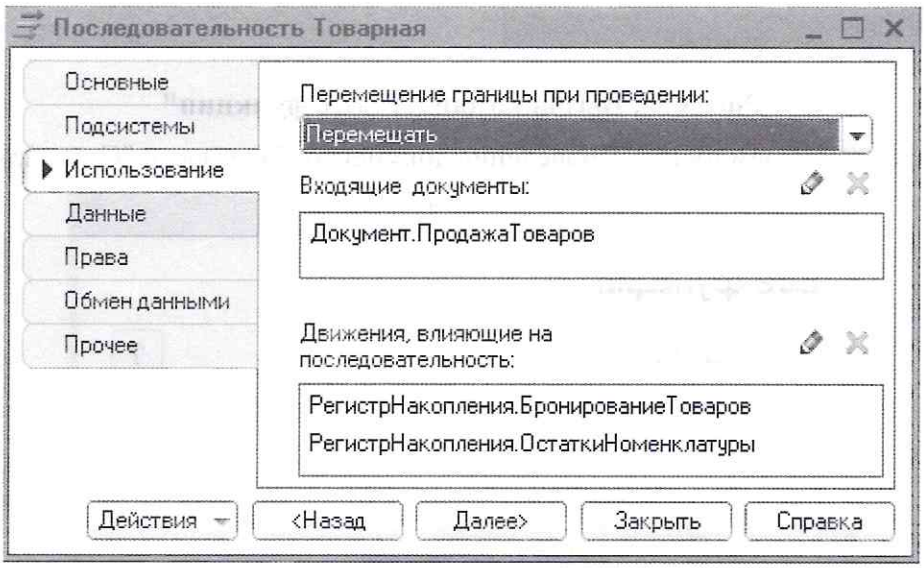


Рис. 3.53. Последовательность "Товарная"

В качестве "Движений, влияющих на последовательность" выбираем регистры, критичные к логике учета: "ОстаткиНоменклатуры" и "БронированиеТоваров". Именно при изменении их данных возможны нарушения последовательности документов, потому что данные именно этих регистров используются (читаются) в алгоритмах обусловлено проводимых документов.

В качестве документов, которые надо будет перепроводить для восстановления последовательности, указываем документы, которые:

а) выполняют при проведении движения по данным регистрам;

б) при этом имеют "обусловленный" тип проведения, то есть при проведении используют не только собственные данные, но и "опираются" на данные регистров.

Эти условия должны выполняться одновременно. Им соответствует только документ "ПродажаТоваров". Дело в том, что документ "ПоступлениеТоваров", безусловно проводимый, сможет только подтвердить свои же прежние движения при перепроведении. А зачем на это время тратить?

Итак, состав последовательности описали. Теперь система будет сама следить за последовательностью наших торговых документов.

В штатном режиме восстановление последовательностей осуществляется в режиме группового проведения документов. Доступ – в пользовательском режиме через команду "Все функции (см.рис. 3.54. Вызов команды "Все функции".)

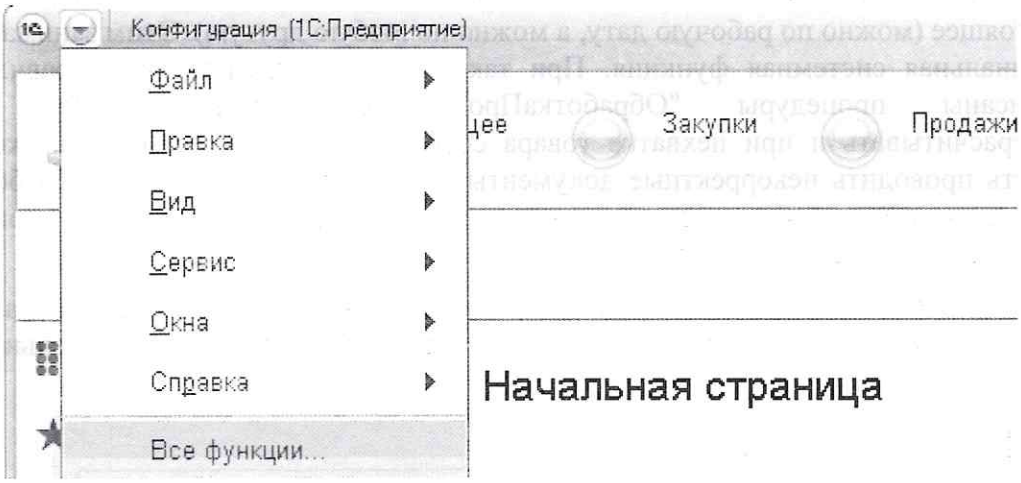


Рис. 3.54. Вызов команды "Все функции"

Далее выбираем пункт "Проведение документов" в разделе "Стандартные".

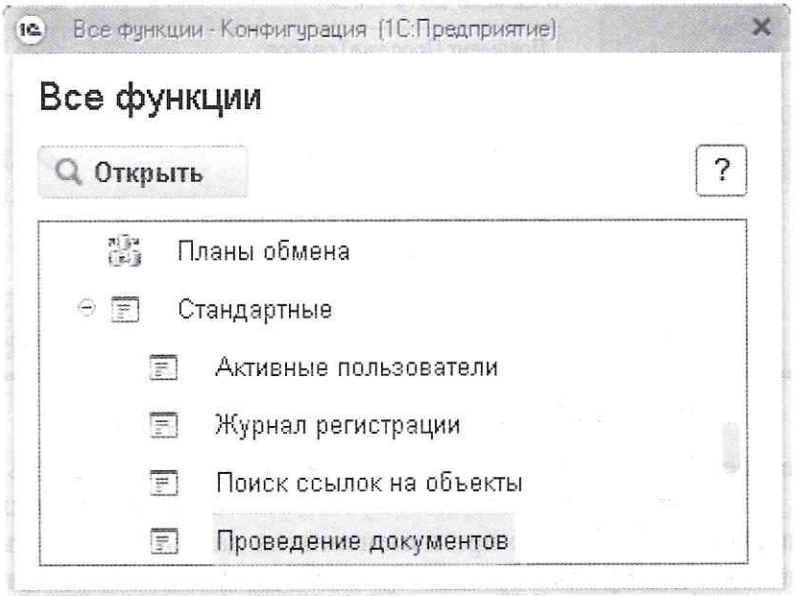


Рис. 3.55. Выбор пункта "Проведение документов"

3.8.2. Облегченный алгоритм проведения документа "ПродажаТоваров"

Теперь переходим к отработке проведения документа "ПродажаТоваров".

Подробно сборку алгоритма проведения мы разбирали в главе "3.2. Сборка алгоритма проведения документа "ПродажаТоваров". Для нашей ситуации повторяем примерно то же самое, за исключением вышеописанных изменений (т.е. отказ от формирования движений по регистру "ОстаткиНоменклатуры", сдвиг контроля отрицательных остатков в самый конец процедуры и т.д.). В результате получаем следующую процедуру:

```
Процедура ОбработкаПроведения(Отказ, Режим)

    Движения.Задолженности.Записывать = Истина;
    // регистр Задолженности Приход
    Движение = Движения.Задолженности.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Контрагент = Контрагент;
    Движение.СуммаДолга = СуммаДокумента;

    Движения.БронированиеТоваров.Записывать = Истина;

    //Установка флага последующего при записи наложения блокировки
    Движения.СвободныеОстатки.БлокироватьДляИзменения = Истина;          //1

    Движения.СвободныеОстатки.Записывать = Истина;
    //Запрос для определения только товаров документа
    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |    ПродажаТоваровТовары.Номенклатура,
    |    СУММА(ПродажаТоваровТовары.Количество) КАК Количество
    |ИЗ
    |    Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
    |ГДЕ
    |    ПродажаТоваровТовары.Ссылка = &Ссылка
    |    И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры <>
    |                    ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)
    |
    |СГРУППИРОВАТЬ ПО
    |    ПродажаТоваровТовары.Номенклатура";
```

```
Запрос.УстановитьПараметр("Ссылка", Ссылка);
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

        // регистр СвободныеОстатки Расход
        Движение = Движения.СвободныеОстатки.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
        Движение.Склад = Склад;
        Движение.КоличествоВСвободномОстатке =
                                        ВыборкаДетальныеЗаписи.Количество;
КонецЦикла;

//Подготовим менеджер временных таблиц для заполнения перечня товаров
//изменившихся свободных остатков в модуле набора записей регистра
МенеджерВременныхТаблиц = Новый МенеджерВременныхТаблиц;          //2
Движения.СвободныеОстатки.ДополнительныеСвойства.Вставить(
      "МенеджерВременныхТаблицПроведенияПродажиТоваров",
       МенеджерВременныхТаблиц);

//Общая запись всех движений
Движения.Записать();                                               //3

//Контроль  отрицательных остатков
Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц =  МенеджерВременныхТаблиц;        //4

Запрос.Текст =
"ВЫБРАТЬ
|       СвободныеОстаткиОстатки.Номенклатура,
|       СвободныеОстаткиОстатки.КоличествоВСвободномОстаткеОстаток
|ИЗ
|       РегистрНакопления.СвободныеОстатки.Остатки(
|                       ,
|                       (Номенклатура, Склад) В
|                               (ВЫБРАТЬ
|                                       ТабИзменившихсяТоваров.Номенклатура,
|                                       ТабИзменившихсяТоваров.Склад
```

```
        |                                ИЗ
        |                                        ТабИзменившихсяТоваров
        |                КАК ТабИзменившихсяТоваров)) КАК СвободныеОстаткиОстатки
        |ГДЕ
        |        СвободныеОстаткиОстатки.КоличествоВСвободномОстаткеОстаток < 0";

        Результат = Запрос.Выполнить();

        ВыборкаДетальныеЗаписи = Результат.Выбрать();

        Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

                Отказ = Истина;
                Сообщение = Новый СообщениеПользователю;
                Нехватка =
                    - ВыборкаДетальныеЗаписи.КоличествоВСвободномОстаткеОстаток;
                Сообщение.Текст = "В документе N" + Номер +" от "+Дата+" не хватает  "
                  +Нехватка+" единиц товара "+ВыборкаДетальныеЗаписи.Номенклатура;
                Сообщение.Сообщить();

        КонецЦикла;

КонецПроцедуры
```

Большей частью текст процедуры нам знаком и понятен. Но некоторые действия требуют отдельных комментариев:

//1 Устанавливаем флаг "БлокироватьДляИзменения" для набора записей движений нашего документа по регистру "СвободныеОстатки". Обратите внимание – в этот момент не происходит ничего, кроме собственно установки флага. Происходить все будет позднее, когда этот набор записей будет записываться в регистр. Вот в этот момент система и заблокирует до конца транзакции записи с наборами значений измерений, соответствующими записываемым данным, то есть в отличие от прошлого алгоритма, поскольку мы поменяли очередность операций с регистром на "Запись/Чтение", программисту даже не обязательно теперь самому описывать объект "БлокировкаДанных". Достаточно только установить флаг "БлокироватьДляИзменения" , чтобы показать системе, что блокировать все же нужно. А что именно блокировать система определит сама в момент записи по данным записываемого набора записей.

//2 Заранее готовимся еще больше ускорить запрос для контроля отрицательных остатков, настраиваясь на ситуацию перепроведения документа. Ведь тогда можно будет контролировать отрицательные остатки только по тем товарам, по которым количество отгрузки только увеличилось. Определить, что именно изменилось в регистре "СвободныеОстатки", можно, сличив состояние движений нашего документа по этому регистру перед записью набора записей и

после записи. Значит, заниматься этим вопросом будет удобнее всего в модуле набора записей регистра – в одноименных обработчиках событий. Но мы хотим потом получить результат этого сличения обратно в нашу процедуру. А поскольку результат нам нужен будет для обеспечения выполнения запроса контроля отрицательных остатков , то удобнее будет получить это в виде временной таблицы. Готовим соответствующий менеджер временных таблиц и передаем его в дополнительные свойства набора записей регистра.

//3 Программно вызываем общую запись всех наборов записей, у которых в рамках данной процедуры установлен флаг "Записывать". Данное действие система выполнит максимально эффективно. Особенно если имеем дело с перепроведением документа. Тогда ранее существующие наборы записей движений этого документа заместятся новыми не в два захода , а в один.

//4 Поскольку запись наборов записей в регистры уже состоялась, то значит, к этому моменту уже сработал механизм определения изменений в наборе записей по регистру "СвободныеОстатки". В запросе используем результат этого механизма – временную таблицу "ТабИзменившихсяТоваров". То есть виртуальная таблица будет получать из базы данных актуальные остатки только для тех товаров, ситуация с которыми для свободных остатков стала "хуже" в результате проведения нашего документа. ВНИМАНИЕ: Сам механизм получения "ТабИзменившихсяТоваров" описан ниже, в разделе 3.8.3!

3.8.3. Получение таблицы товаров, по которым в результате проведения документа свободные остатки уменьшились

Очень часто перепроведение документа вызвано просто нажатием пользователем кнопки "Провести и закрыть" в форме уже проведенного документа, то есть пользователь хотел посмотреть на старый документ, открыл его форму, посмотрел и … закрыл форму самой красивой кнопкой.

С точки зрения устойчивости системы ничего страшного произойти не должно. То, что при этом стартует вся транзакция "Запись документа с проведением" , не должно изменить в учете ровным счетом ничего, если учет велся корректно. Но хотелось бы при этом еще и время не терять на отработку самой транзакции в полном объеме.

Или другой пример : пользователь открыл форму старого проведенного документа для исправления какой-то ошибки. Скорее всего, ошибка касалась одной-двух товарных позиций в документе, но никак не всех десятков или сотен строк табличной части. Да и, возможно, исправлением будет как раз удаление строки с неправильно оформленной товарной позицией.

В этих описанных примерах мы видим возможность сэкономить время перепроведения документа за счет облегчения запроса контроля остатков. Запрос надо строить только по тем товарным позициям, по которым ситуация из-за проведения нашего документа – ухудшится, то есть свободных остатков станет меньше.

Получим перечень именно таких товарных позиций на сличении результатов запросов к движениям нашего документа по регистру перед записью и после записи в регистр.

Механизм реализуется на одноименных обработчиках событий в модуле набора записей регистра "СвободныеОстатки":

```
Процедура ПередЗаписью(Отказ, Замещение)
        Если ДополнительныеСвойства.
            Свойство("МенеджерВременныхТаблицПроведенияПродажиТоваров")   //1
                                                                    Тогда
            //Надо получать данные старых движений
            Запрос = Новый Запрос;
            Запрос.МенеджерВременныхТаблиц = ДополнительныеСвойства.
                        МенеджерВременныхТаблицПроведенияПродажиТоваров; //2

            Запрос.Текст =
            "ВЫБРАТЬ                                                    //3
            |       СвободныеОстатки.Номенклатура,
            |       СвободныеОстатки.Склад,
            |       СвободныеОстатки.КоличествоВСвободномОстатке
            |                   КАК КоличествоВСвободномОстаткеСтарое
            |ПОМЕСТИТЬ ТабСтарыхДвижений
            |ИЗ
            |       РегистрНакопления.СвободныеОстатки КАК СвободныеОстатки
            |ГДЕ
            |       СвободныеОстатки.Регистратор = &Регистратор
            |       И СвободныеОстатки.ВидДвижения =
            |               ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход)";

            Запрос.УстановитьПараметр("Регистратор",
                                ЭтотОбъект.Отбор.регистратор.Значение);     //4
            Запрос.Выполнить();                                             //5

        КонецЕсли;
КонецПроцедуры

Процедура ПриЗаписи(Отказ, Замещение)
        Если ДополнительныеСвойства.
            Свойство("МенеджерВременныхТаблицПроведенияПродажиТоваров")     //6
                                                                    Тогда
            //Надо сформировать таблицу товаров, по которым произошло
            // ухудшение свободного остатка в процессе проведения

            Запрос = Новый Запрос;
            Запрос.МенеджерВременныхТаблиц = ДополнительныеСвойства.
```

```
                МенеджерВременныхТаблицПроведенияПродажиТоваров; //7

Запрос.Текст =
"ВЫБРАТЬ                                                          //8
|ВложенныйЗапрос.Номенклатура,
|ВложенныйЗапрос.Склад,
|СУММА(ВложенныйЗапрос.КоличествоВСвободномОстаткеПриращение)
|                        КАК КоличествоВСвободномОстаткеПриращение
|ПОМЕСТИТЬ ТабИзменившихсяТоваров
|ИЗ
|(ВЫБРАТЬ
|    ТабСтарыхДвижений.Номенклатура КАК Номенклатура,
|    ТабСтарыхДвижений.Склад КАК Склад,
|    -ТабСтарыхДвижений.КоличествоВСвободномОстаткеСтарое
|            КАК КоличествоВСвободномОстаткеПриращение
|ИЗ
|    ТабСтарыхДвижений КАК ТабСтарыхДвижений
|
|ОБЪЕДИНИТЬ ВСЕ
|
|ВЫБРАТЬ
|    СвободныеОстатки.Номенклатура,
|    СвободныеОстатки.Склад,
|    СвободныеОстатки.КоличествоВСвободномОстатке
|ИЗ
|    РегистрНакопления.СвободныеОстатки КАК СвободныеОстатки
|ГДЕ
|    СвободныеОстатки.Регистратор = &Регистратор
|    И СвободныеОстатки.ВидДвижения =
|            ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход)
|                                    ) КАК ВложенныйЗапрос
|
|СГРУППИРОВАТЬ ПО
|    ВложенныйЗапрос.Номенклатура,
|    ВложенныйЗапрос.Склад
|
|ИМЕЮЩИЕ
|СУММА(ВложенныйЗапрос.КоличествоВСвободномОстаткеПриращение)
|                                                        > 0
|
|ИНДЕКСИРОВАТЬ ПО
```

```
                |       Номенклатура,
                |       Склад";


                Запрос.УстановитьПараметр("Регистратор",
                                        ЭтотОбъект.Отбор.регистратор.Значение); //9
                Запрос.Выполнить();                                             //10

        КонецЕсли;
КонецПроцедуры
```

Комментарии к текстам процедур:

//1 Проверяем, вызвана ли запись набора записей отработкой проведения документа "ПродажаТоваров", потому что только в обработке проведения программно вводится в дополнительные свойства объекта набора записей элемент "МенеджерВременныхТаблицПроведенияПродажиТоваров". Поскольку у дополнительных свойств тип значения "Структура", то достаточно методом "Свойства" проверить наличие элемента с одноименным ключом.

//2 Создаем запрос и подключаем к нему менеджер временных таблиц, созданный в рамках процедуры "ОбработкаПроведения" документа "ПродажаТоваров".

//3 Текст запроса пишем так, чтобы выбрать записи, где проводимый документ формировал движения "расход". Поскольку этот запрос к базе данных выполняется перед записью набора записей регистра, то явно речь пойдет о старых движениях. Результат запроса будет размещен во временной таблице "ТабСтарыхДвижений", подключенной к запросу менеджера временных таблиц.

//4 Значение ссылки на проводимый документ получаем из предустановленного отбора записываемого набора записей.

//5 Выполняем запрос. Результатом выполнения запроса окажется заполнение временной таблицы, описанной в тексте нашего запроса. А больше нам от него ничего и не нужно.

//6 В процедуре "ПриЗаписи" аналогично стоит сначала проверить, вызвана ли запись набора записей отработкой проведения документа "ПродажаТоваров", потому что в прочих ситуациях наш механизм просто излишен.

//7 Подключаем новый формируемый запрос к тому же самому менеджеру временных таблиц. Напомним, именно в нем сейчас лежит временная таблица, сформированная при выполнении запроса в предыдущей процедуре.

//8 Текст запроса строим так, чтобы во вложенном запросе объединить записи из временной таблицы "ТабСтарыхДвижений" с таблицей текущих движений вида "Расход" нашего документа (уже после записи набора записей). Причем количество в таблице старых движений делаем со знаком "минус", а в таблице новых со знаком "плюс". Тогда после свертки полученной объединенной таблицы по полям "Номенклатура" и "Склад" в количественном поле будет получено значение приращения, произошедшего при проведении нашего документа. Далее будут отброшены все записи, приращение в которых не

произошло. Результат запроса просим разместить во временной таблице "ТабИзменившихсяТоваров".

//9 Значение ссылки на проводимый документ получаем из предустановленного отбора записываемого набора записей.

//10 Само выполнение запроса и вызовет появление временной таблицы "ТабИзменившихсяТоваров" в менеджере временных таблиц, подключенных к нашему запросу.

Вот, собственно, и все. Как далее будет использована временная таблица "ТабИзменившихсяТоваров" , разбиралось в предыдущем разделе "3.8.2. Облегченный алгоритм проведения документа "ПродажаТоваров".

Практикум №11.2

Перепишите оба алгоритма бронирования документа "ПродажаТоваров" по методике "ускоренного проведения", то есть сначала наборы записей должны попасть в регистры, а потом уже должен быть выполнен контрольный функционал достаточности товаров в регистре "СвободныеОстатки".

3.8.4. Регламентное списание себестоимости для документов продажи

Механизм партионного списания себестоимости при проведении документа "ПродажаТоваров" мы разрабатывали в разделе "3.6. Организация партионного учета". В ситуации выноса этого механизма из обработки проведения сначала давайте определимся с его размещением.

Сам по себе механизм должен работать без лишних обращений к объекту документа. То есть модуль документа нам не подходит. Но с другой стороны, видов документов отгрузки в нашей конфигурации не много , точнее, всего один, документ "ПродажаТоваров". Поэтому уместно вынести партионное списание в модуль менеджера документа "ПродажаТоваров".

Далее, списывать товар надо будет последовательно: для каждого экземпляра документа отдельно. Ведь очередность списания партий многое значит для разрабатываемого механизма.

Кроме того, не факт что при попытке проведения очередного документа по партиям товара хватит, значит, надо как-то отслеживать границу, до которой товар по партиям распределился и в приходных, и в расходных документах, а после которой товар по партиям не распределен.

Ну и сам механизм распределения по партиям надо иметь возможность вызывать как-то автоматически с такой частотой, чтобы несильно нагружать ресурсы системы.

Учет всех этих требований приводит нас к конструированию механизма, включающего в себя задействование нескольких объектов: последовательности, функции списания партий в модуле менеджера документа, обработки, вызывающей функцию в цикле перебора ссылок на документы из последовательности, регламентного задания, вызывающего обработку.

Делаем все по-порядку.

Отслеживать временную границу, начиная с которой товар надо распределять по партиям , лучше всего на объекте "Последовательность". Благо у нас уже есть последовательность "Товарная". С точки зрения изменения самой последовательности нам нужно только добавить еще один регистр к списку регистров, сбивающих границу последовательности и отключить автоматическое перемещение границы последовательности вперед при проведении документов "ПродажаТоваров". Это сделать очень просто – устанавливаем соответствующее свойства последовательности, как на рис.3.66. Изменение свойств последовательности.

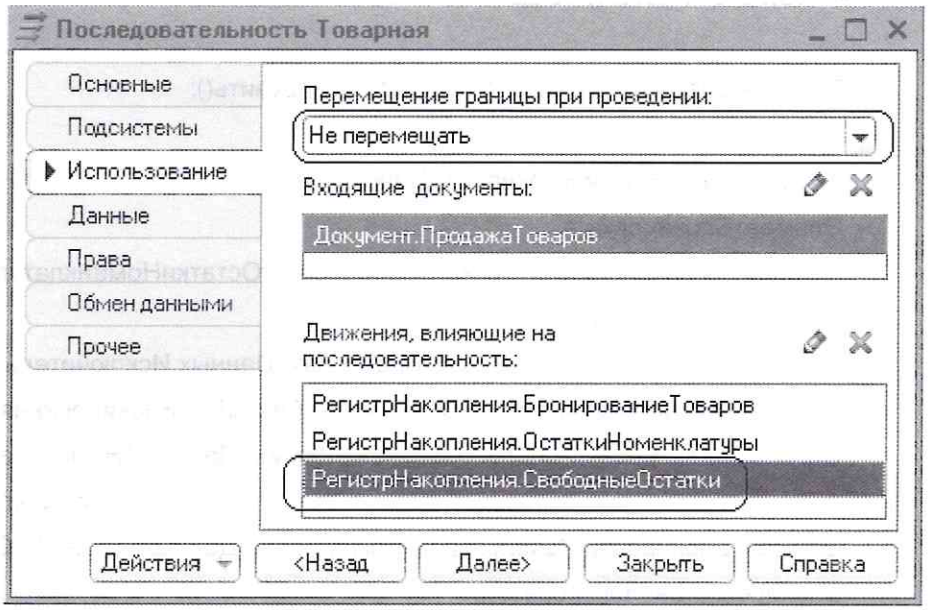


Рис. 3.66. Изменение свойств последовательности

Далее, поскольку списание партий не всегда будет проходить успешно, лучше сам механизм списания заключить в функцию, которая кроме списания еще возвращала бы флаг успешности выполнения этой операции.

Размещаем в модуле менеджера документа "ПродажаТоваров" такую функцию:

```
Функция СписатьСебестоимостьТоваровДокумента(ДокументСсылка)  Экспорт    //1

    НачатьТранзакцию(РежимУправленияБлокировкойДанных.Управляемый);   //2

    //Обеспечение неизменности данных по себестоимости до конца транзакции
    Запрос = Новый Запрос;                                                  //3
    Запрос.Текст =
    "ВЫБРАТЬ РАЗЛИЧНЫЕ
    |       ПродажаТоваровТовары.Номенклатура,
    |       ПродажаТоваровТовары.Ссылка.Склад
    |ИЗ
    |       Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
    |ГДЕ
```

```
|        ПродажаТоваровТовары.Ссылка = &Ссылка
|        И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры <>
|                              ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)";

Запрос.УстановитьПараметр("Ссылка", ДокументСсылка);

Результат = Запрос.Выполнить();

Если не  Результат.Пустой() Тогда

        ТаблицаДляБлокирования = Результат.Выгрузить();

        Блокировка = Новый БлокировкаДанных;
        ЭлементБлокировки =
           Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
        ЭлементБлокировки.Режим         =
                                    РежимБлокировкиДанных.Исключительный;
        ЭлементБлокировки.ИсточникДанных = ТаблицаДляБлокирования;
        ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Номенклатура",
                                                        "Номенклатура");
        ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Склад", "Склад");
        Блокировка.Заблокировать();

КонецЕсли;


ФлагУспешностиСписания = Истина;                                        //4
//Зачищаем от старых движений
НаборЗаписей=РегистрыНакопления.ОстаткиНоменклатуры.
                                                СоздатьНаборЗаписей();
НаборЗаписей.Отбор.Регистратор.Установить(ДокументСсылка);
НаборЗаписей.Записать();                                                //5


//расчет себестоимости
//отсечение услуг
Запрос = Новый Запрос;                                                  //6
Запрос.Текст =
"ВЫБРАТЬ
|        ПродажаТоваровТовары.Номенклатура,
|        ПродажаТоваровТовары.Ссылка.Склад,
```

```
|        СУММА(ПродажаТоваровТовары.Количество) КАК Количество,
|        ПродажаТоваровТовары.Ссылка.Дата
|ПОМЕСТИТЬ ТабДок
|ИЗ
|        Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
|ГДЕ
|        ПродажаТоваровТовары.Ссылка = &Ссылка
|        И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры <>
|                        ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)
|
|СГРУППИРОВАТЬ ПО
|        ПродажаТоваровТовары.Номенклатура,
|        ПродажаТоваровТовары.Ссылка.Склад,
|        ПродажаТоваровТовары.Ссылка.Дата
|
|ИНДЕКСИРОВАТЬ ПО
|        Номенклатура,
|        Склад
|;
|
|////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|        ТабДок.Дата,
|        ТабДок.Склад,
|        ТабДок.Номенклатура КАК Номенклатура,
|        ТабДок.Количество КАК Количество,
|        ОстаткиНоменклатурыОстатки.Партия,
|        ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0)
|                                                КАК КоличествоОстаток,
|        ЕСТЬNULL(ОстаткиНоменклатурыОстатки.СуммаОстаток, 0)
|                                                КАК СуммаОстаток
|ИЗ
|        ТабДок КАК ТабДок
|                ЛЕВОЕ СОЕДИНЕНИЕ
|  РегистрНакопления.ОстаткиНоменклатуры.Остатки(
|                        &Момент,
|                        (Номенклатура, Склад) В
|                                (ВЫБРАТЬ
|                                        ТабДок.Номенклатура,
|                                        ТабДок.Склад
|                                ИЗ
```

```
|                                                    ТабДок КАК ТабДок)
|                                                ) КАК ОстаткиНоменклатурыОстатки
|                    ПО ТабДок.Номенклатура =
|                                        ОстаткиНоменклатурыОстатки.Номенклатура
|ИТОГИ
|        МАКСИМУМ(Количество),
|        СУММА(КоличествоОстаток),
|        СУММА(СуммаОстаток)
|ПО
|        Номенклатура";

Запрос.УстановитьПараметр("Ссылка", ДокументСсылка);
Запрос.УстановитьПараметр("Момент", ДокументСсылка.МоментВремени()); //7
Результат = Запрос.Выполнить();

ВыборкаНоменклатура =
            Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

Пока ВыборкаНоменклатура.Следующий() Цикл

        Если ВыборкаНоменклатура.КоличествоОстаток <
                                        ВыборкаНоменклатура.Количество Тогда
            // в отрицательных количествах себестоимости не бывает
            ФлагУспешностиСписания = Ложь;                              //8

            Сообщение = Новый СообщениеПользователю;
            Нехватка =  ВыборкаНоменклатура.Количество
                        - ВыборкаНоменклатура.КоличествоОстаток;
            Сообщение.Текст = "В документе " + ДокументСсылка +
                            " не хватает  "+Нехватка+" единиц товара "+
                                ВыборкаНоменклатура.Номенклатура;
            Сообщение.Сообщить();
        Иначе
            КоличествоНадоСписать = ВыборкаНоменклатура.Количество;

            ВыборкаДетальныеЗаписи = ВыборкаНоменклатура.Выбрать();
            Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

                Если ВыборкаДетальныеЗаписи.КоличествоОстаток <=
                                                КоличествоНадоСписать Тогда
```

```
                    //списываем всю партию
                    КоличествоКСписанию =
                        ВыборкаДетальныеЗаписи.КоличествоОстаток;
                    СтоимостьКСписанию =
                        ВыборкаДетальныеЗаписи.СуммаОстаток;
                    //уменьшаем количество к списанию
                    КоличествоНадоСписать = КоличествоНадоСписать
                                            - КоличествоКСписанию;

Иначе

        //списываем часть партии
        КоличествоКСписанию = КоличествоНадоСписать;
        Если ВыборкаДетальныеЗаписи.КоличествоОстаток <> 0 Тогда
            СтоимостьКСписанию =
                ВыборкаДетальныеЗаписи.СуммаОстаток/
              ВыборкаДетальныеЗаписи.КоличествоОстаток
                *КоличествоКСписанию;
        иначе
            СтоимостьКСписанию =
                ВыборкаДетальныеЗаписи.СуммаОстаток;
        КонецЕсли;

            //обнуляем количество к списанию
            КоличествоНадоСписать = 0;

        КонецЕсли;
//формируем набор записей
Движение = НаборЗаписей.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
Движение.Период = ВыборкаДетальныеЗаписи.Дата;
Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
Движение.Склад = ВыборкаДетальныеЗаписи.Склад;
Движение.Партия = ВыборкаДетальныеЗаписи.Партия;
Движение.Количество = КоличествоКСписанию;
Движение.Сумма = СтоимостьКСписанию;

//проверяем необходимость дальнейшего перебора партий
Если КоличествоНадоСписать <= 0 Тогда
        Прервать;
КонецЕсли;
```

```
        КонецЦикла; // по партиям
    КонецЕсли;
КонецЦикла; // по номенклатурным позициям

    Если ФлагУспешностиСписания и НаборЗаписей.Количество()<>0 Тогда
        //запись рассчитанных данных  в регистр
        НаборЗаписей.Записать();                                    //9
    КонецЕсли;

    ЗафиксироватьТранзакцию();                                      //10

    Возврат ФлагУспешностиСписания;                                 //11

КонецФункции
```

Комментарии к тексту процедуры:

//1 В заголовке функции указываем, что ждем в качестве параметра ссылку на документ "ПродажаТоваров", который надо списать по партиям;

//2 Для обеспечения корректности разрабатываемого механизма включаем все его действия в единую транзакцию;

//3 Выполняем маленький запрос по данным документа для последующего наложения блокировки, обеспечивающей неизменность тех данных, с которыми будем работать, до конца транзакции;

//4 Создаем флаг успешности выполнения нашей функции. Поначалу полагаем, что все пройдет удачно, так что присваиваем ему значение "Истина";

//5 Перед тем как выполнять запрос для списания партий ,надо очистить регистр от старых движений нашего документа (а вдруг такие найдутся!);

//6 Выполняем запрос для последующего партионного списания. Запрос практически такой же, как мы разрабатывали раньше, только в него еще добавлено выходное поле с датой документа для последующего заполнения поля "Период" регистра;

//7 При установке параметров определяем, что виртуальная таблица должна строиться на момент проведения документа. Для себестоимости это очень важно!

//8 Если по каким-то причинам товара не хватает, то нельзя рассчитывать себестоимость , то есть нельзя документ проводить по партиям. Устанавливаем соответствующее значение флагу успешности списания;

//9 Если все удачно, запишем сформированный набор записей в регистр;

//10 Зафиксируем транзакцию;

//11 Вернем флаг успешности списания, каким бы он не оказался.

Теперь создадим обработку, которая будет определять по последовательности , для каких документов надо вызывать выше разработанную функцию "СписатьСебестоимостьТоваровДокумента".

Назовем обработку "СписаниеСебестомостиДляДокументовПродажи".

Включим ее в подсистему "Документы" подсистемы "Продажи".

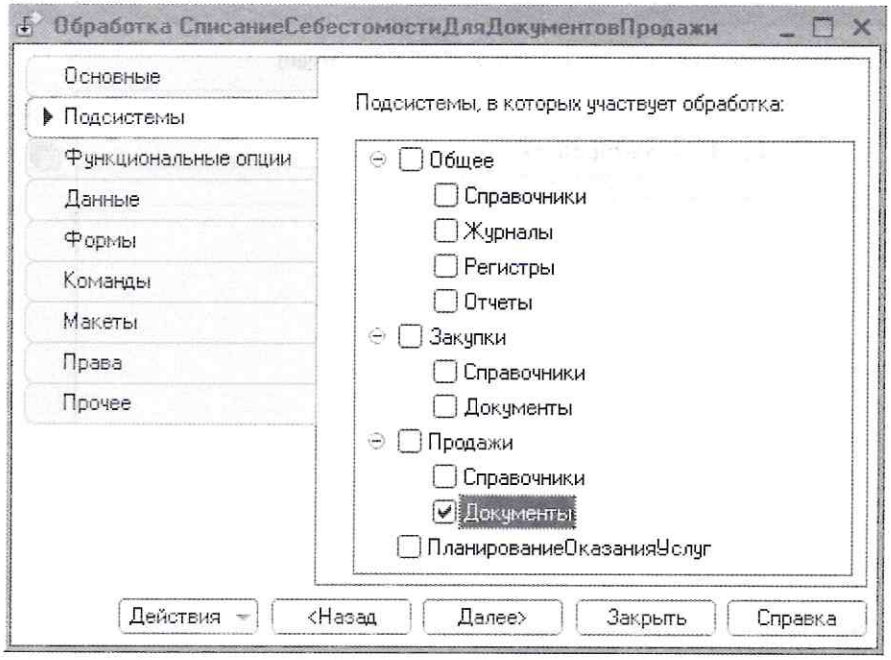


Рис. 3.67. Новая обработка

Обработка должна работать в некоем интервале дат. Поэтому введем в нее два реквизита "ДатаНачалаОбработки" и "ДатаОкончанияОбработки" с типом значения <Дата> и составом даты "Дата и время". Это позволит нам впоследствии легко задавать значения интервала обработки как при интерактивной работе с формой обработки, так и при программном ее вызове на сервере, где формы не будет.

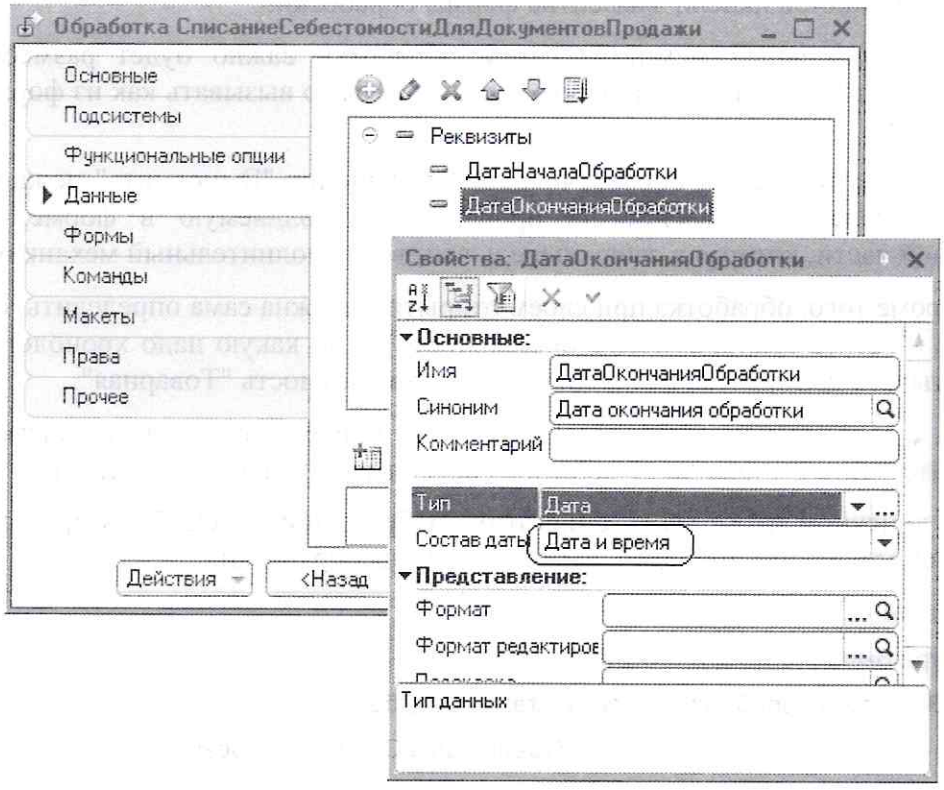


Рис. 3.68. Реквизиты обработки

Создадим форму обработки конструктором формы. Обязательно включим в состав элементов управления, располагаемых на диалоге реквизиты обработки.

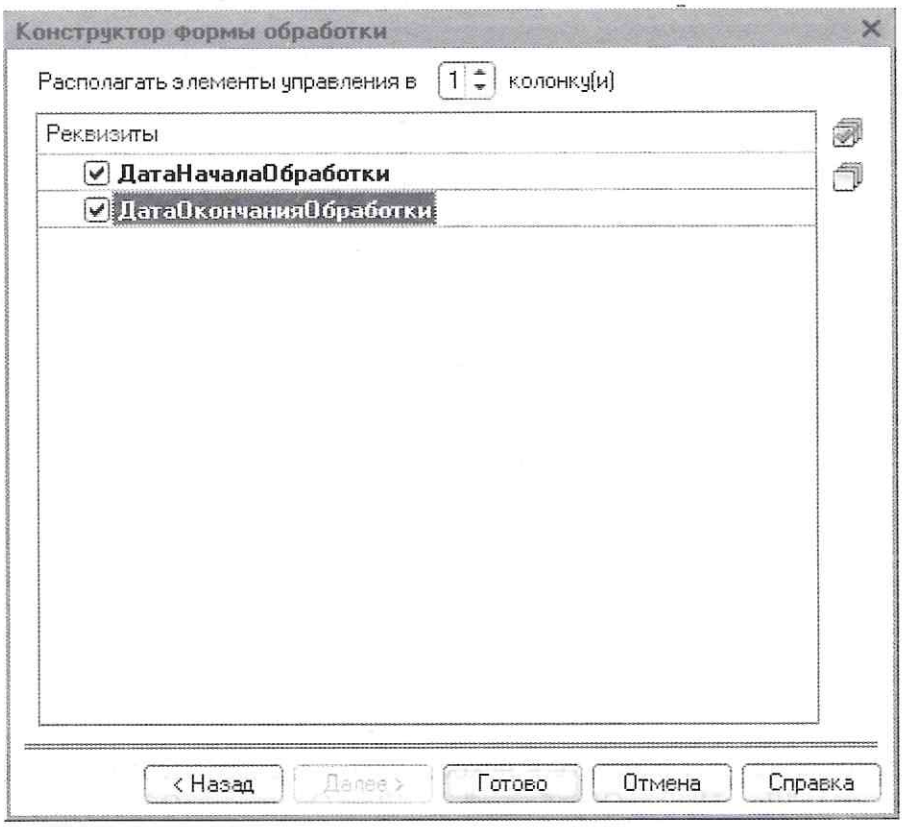


Рис. 3.68. Конструктор формы новой обработки

Нажав кнопку "Готово" ,получим форму обработки. В ней создадим новую команду формы "ВыполнитьОбработку" и перетащим мышкой эту команду в область командной панели элементов формы обработки.

Исполнительный механизм самой обработки важно будет разместить в модуле менеджера обработки, чтобы его можно было вызывать как из формы, так и без формы.

Поэтому в качестве обработчика нажатия кнопки "Выполнить" можно будет просто передать управление на процедуру, выполняемую в форме, но на серверной части, которая в свою очередь вызовет исполнительный механизм.

Кроме того, обработка при своем открытии должна сама определить, с какой даты (с момента границы последовательности) и по какую надо хронологически обрабатывать документы, входящие в последовательность "Товарная".

И после выполнения восстановления последовательности показать на реквизите "Дата окончания обработки" , насколько она в этом преуспела.

В конечном итоге реализация этих соображений может быть представлена следующим набором процедур на модуле формы обработки:

```
&НаКлиенте
Процедура ВыполнитьОбработку (Команда)
        ЗапуститьОбработку(Объект.ДатаНачалаОбработки,
                                Объект.ДатаОкончанияОбработки);
        //перезаполним положение ГП
```

```
        Объект.ДатаНачалаОбработки = ОпределитьДатуГП();

КонецПроцедуры

&НаСервереБезКонтекста
Процедура ЗапуститьОбработку(ДатаНачала,ДатаОкончания)

        Обработки.СписаниеСебестомостиДляДокументовПродажи.
            ВыполнитьСписаниеИСместитьГП(ДатаНачала,ДатаОкончания);

КонецПроцедуры



&НаСервереБезКонтекста
Функция ОпределитьДатуГП()

        возврат Последовательности.Товарная.ПолучитьГраницу().Дата;

КонецФункции

&НаКлиенте
Процедура ПриОткрытии(Отказ)

        Объект.ДатаНачалаОбработки = ОпределитьДатуГП();
        Объект.ДатаОкончанияОбработки = ТекущаяДата();

КонецПроцедуры
```

Ну а в модуле менеджера обработки располагаем саму процедуру "ВыполнитьСписаниеИСместитьГП":

```
Процедура ВыполнитьСписаниеИСместитьГП(ДатаНачалаОбработки,
                                       ДатаОкончанияОбработки) Экспорт

        Запрос = Новый Запрос;
        Запрос.Текст =                                                  //1
            "ВЫБРАТЬ
            |       Товарная.Регистратор КАК Регистратор,
            |       Товарная.Период КАК Период
            |ИЗ
            |       Последовательность.Товарная КАК Товарная
            |ГДЕ
            |       Товарная.Период >= &ДатаНачалаОбработки
```

```
            |        И Товарная.Период <= &ДатаОкончанияОбработки
            |        И Товарная.Регистратор.Проведен
            |
            |УПОРЯДОЧИТЬ ПО
            |        Период,
            |        Регистратор";

    Запрос.УстановитьПараметр("ДатаНачалаОбработки", ДатаНачалаОбработки);
    Запрос.УстановитьПараметр("ДатаОкончанияОбработки",
                                                ДатаОкончанияОбработки);

    Результат = Запрос.Выполнить();

    ВыборкаДетальныеЗаписи = Результат.Выбрать();

    СсылкаПоследнегоОбработанногоДокумента = Неопределено;                    //2

    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

        Если Документы.ПродажаТоваров.
            СписатьСебестоимостьТоваровДокумента(
                ВыборкаДетальныеЗаписи.Регистратор) Тогда                     //3
            СсылкаПоследнегоОбработанногоДокумента =
                                    ВыборкаДетальныеЗаписи.Регистратор;       //4
            Иначе
                Прервать;                                                     //5
        КонецЕсли;

    КонецЦикла;

    Если ЗначениеЗаполнено(СсылкаПоследнегоОбработанногоДокумента) Тогда
        // Смещаем границу последовательности                                //6
        ГраницаПоследнегоОбработанногоДокумента =
            СсылкаПоследнегоОбработанногоДокумента.МоментВремени();
        Последовательности.Товарная.УстановитьГраницу(
                                    ГраницаПоследнегоОбработанногоДокумента);
    КонецЕсли;


КонецПроцедуры
```

Комментарий к тексту процедуры:

//1 Запрос строим в рамках переданного интервала дат так, чтобы в нем получить в порядке хронологии ссылки на документы, которые ниже надо обрабатывать;

//2 Готовим промежуточную переменную под запоминание последнего обработанного документа;

//3 В цикле перебора выборки результата запроса вызываем функцию "СписатьСебестоимостьТоваровДокумента" для каждого обрабатываемого документа. Напомним, что функция, кроме непосредственного списания, еще возвращает флаг успешности списания;

//4 Если списание прошло успешно, запоминаем ссылку на обработанный документ в качестве значения переменной СсылкаПоследнегоОбработанногоДокумента;

//5 Если списание прошло неудачно, незачем пытаться списывать последующие документы. С партиями уже что-то не так;

//6 Смещаем границу последовательности на момент последнего обработанного документа, только если хоть один документ обработан.

Теперь можно приступать к созданию регламентного задания, которое будет отрабатывать вызов нашей обработки.

Итак, создаем новое регламентное задание.

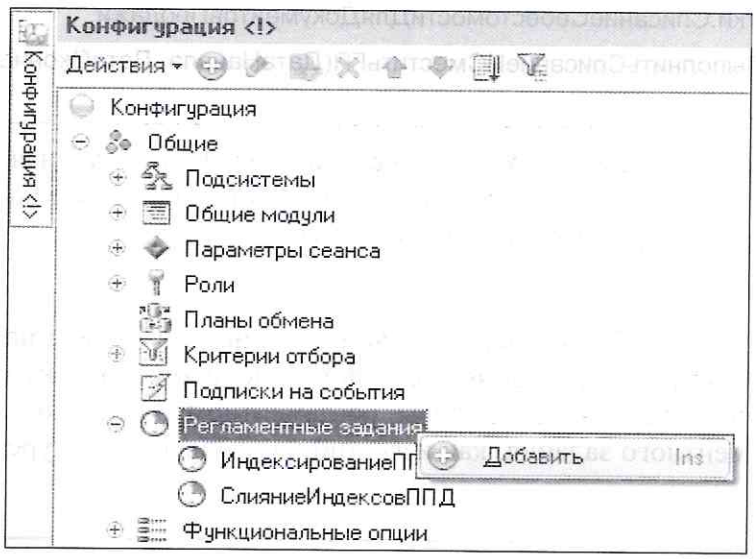


Рис. 3.70. Добавление нового регламентного задания

Дадим ему название "СписаниеСебестоимостиДляДокументовПродажи" и вызовем формирование обработчика события , нажав кнопку с лупой в свойстве "Имя метода".

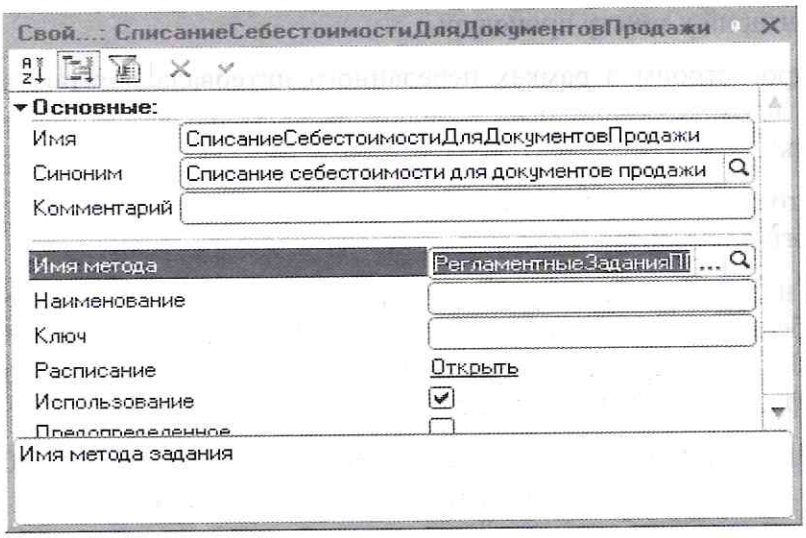


Рис. 3.71. Свойства нового регламентного задания

В качестве тела обработчика события выполнения нашего регламентного задания создаем такую процедуру:

```
Процедура СписаниеСебестоимостиДляДокументовПродажи() Экспорт
    ДатаНачала = Последовательности.Товарная.ПолучитьГраницу().Дата;
    ДатаОкончания = ТекущаяДата();

    Обработки.СписаниеСебестомостиДляДокументовПродажи.
        ВыполнитьСписаниеИСместитьГП(ДатаНачала, ДатаОкончания);
КонецПроцедуры
```

То есть определяем по границе последовательности, начиная с какой даты надо выполнять обработку, и вызываем собственно исполнительный механизм обработки – процедуру "ВыполнитьСписаниеИСместитьГП".

Ну, вот и все.

Теперь при необходимости механизм списания себестоимости можно задействовать интерактивно , вызвав соответствующую обработку из интерфейса программы или доверить его вызовы регламентному заданию. А расписание вызовов регламентного задания, как известно из предыдущего курса, может быть любым.

Практикум №11.3

Реализованный механизм партионного списания отрабатывает только метод FIFO. Необходимо дать возможность пользователю самому выбирать, как организовать списание партий: по FIFO (списывать, начиная с самых старых) или по LIFO (списывать, начиная с самых "свежих"). Данные надо брать все из того же регистра сведений "УчетнаяПолитика".

4. Решение задач анализа показателей движения. Использование реквизитов регистра остатков и оборотных регистров

4.1. Отчет "АнализПродажДок". Построение отчета запросом по документам

Для работы как руководства отдела закупок, так и отдела продаж крайне интересно знать, какие товары являются ходовыми, а какие нет. Но еще важнее знать эти данные как за текущий период, так и в сравнении с какими-то периодом ***предыдущим***, чтобы видеть тенденцию продаж того или иного товара.

Причем границы каждого из периодов должен иметь право выбирать сам пользователь. Захочет – сравнит продажи текущего месяца с прошлым, а захочет – сравнит продажи первых шести месяцев года с августом прошлого года (если, например, речь о торговле свежими фруктами , еще неизвестно, где больше окажется).

То есть нужен отчет такого типа (см. рис. 4.1)

Номенклатура	КолТек	СумТек	КолПред	СумПред	ОстТек
Паркер "Golg"	4	480			20
Kohinor тм	14	140	1	10	10
Тетради общие 96листов			2	90	1
Шариковая синяя					60
Набор "Радуга" 6цв					40
нов	13	65			7
Общий объем продаж:		685		100	

Рис. 4.1. Анализ продаж

Можем ли мы построить его, основываясь на существующих данных конфигурации? Можем.

Более того, можем собрать эту информацию различными способами. Единственное, что мы не можем – это собрать этот отчет эффективно по скорости при помощи объектной модели получения информации, то есть за одно обращение к базе данных собирать такие отчеты можно только при помощи запроса. Поэтому для себя надо поставить знак равенства: "Говорим, что нужен отчет ,значит, прежде всего, нужен запрос, который реализует этот отчет".

А вот для того, чтобы всегда иметь спектр вариантов получения отчета , принимаем первую вводную. Допустим, речь идет о построении отчета в уже "брошенной" базе данных, , то есть предприятие вело-вело учет в некой базе, но с какого-то момента было принято решение о переходе на новую версию конфигурации без конвертации всей базы, и сейчас учет ведется в новой базе, в которую перенесли справочники и остатки, а вот данные о торговых операциях прошлых лет переносить не стали. Они остались в старой "брошенной" базе.

Вот именно по данным этой брошенной базы нам и надо получить отчет.

Если постановка задачи именно такая , то фактически тем самым снимается требование оперативности построения отчета, к нему, скорее всего, будут

обращаться не каждый час или день. А хорошо, если его построят вообще больше трех-четырех раз. База-то "брошенная".

В такой ситуации в ходе решения отчетной задачи очень не хотелось бы вносить какие-то структурные изменения в конфигурацию базы данных, а заведомо без претензий на скорость хотелось бы просто получить этот отчет.

Итак, приступаем к работе.

Написание любого запроса предваряется составлением схемы запроса. Как составлять схему запроса , мы обсуждали в главе "2.3.2. Использование табличной модели системы "1С:Предприятие" ("Запрос")". Превратите самостоятельно рис.4.1. Анализ продаж в схему запроса.

При отработке пункта "3.Обозначить для каждого поля таблицу-источник" определим, что последняя колонка будет получена из виртуальной таблицы "РегистрНакопления.ОстаткиНоменклатуры.Остатки". Согласно вводной о "брошенности" базы колонки "КолПред" и "СумПред" будут получены из табличных частей первичных документов, т.е. "Документ.ПродажаТоваров.Товары". Аналогична судьба колонок "КолТек" и "СумТек" - лучше их получать отдельным обращением к "Документ.ПродажаТоваров.Товары". Почему? Правильно , интервалы предыдущих и текущих продаж разные, поэтому получать данные лучше за два отдельных обращения.

Прежде чем определить судьбу "группировочной сущности", т.е. колонки "Номенклатура", необходимо по схеме запроса определиться : в нашем запросе будет иметь место СОЕДИНЕНИЕ данных таблиц-источников или ОБЪЕДИНЕНИЕ.

Поскольку по эскизу заполнения выходной таблицы запроса видно , что ЛЮБОЙ источник имеет право добавлять в отчет новые товары , выбираем ОБЪЕДИНЕНИЕ.

Важно!

С точки зрения техники выполнения запроса желательно помнить несколько эмпирических правил:

- Объединение – это, по сути, добавление снизу к выходной таблице запроса по первому источнику выходных таблиц запросов по остальным источникам.
- Значит, когда имеем дело с объединением ,то для каждого источника нужен самодостаточный запрос;
- А чтобы каждый запрос был самодостаточным, колонки с группировочными сущностями (в нашем случае - номенклатура) должны указываться в качестве выходных полей в каждом из этих запросов;
- Поскольку выходная таблица запроса может быть только прямоугольной, то при реализации объединения важно, чтобы каждый из объединяемых запросов имел одинаковое количество полей, то есть равное количеству полей на схеме запроса.

Итак, добавляем в состав метаданных новый отчет "АнализПродажПоДок". Включим его в состав подсистемы "Продажи".

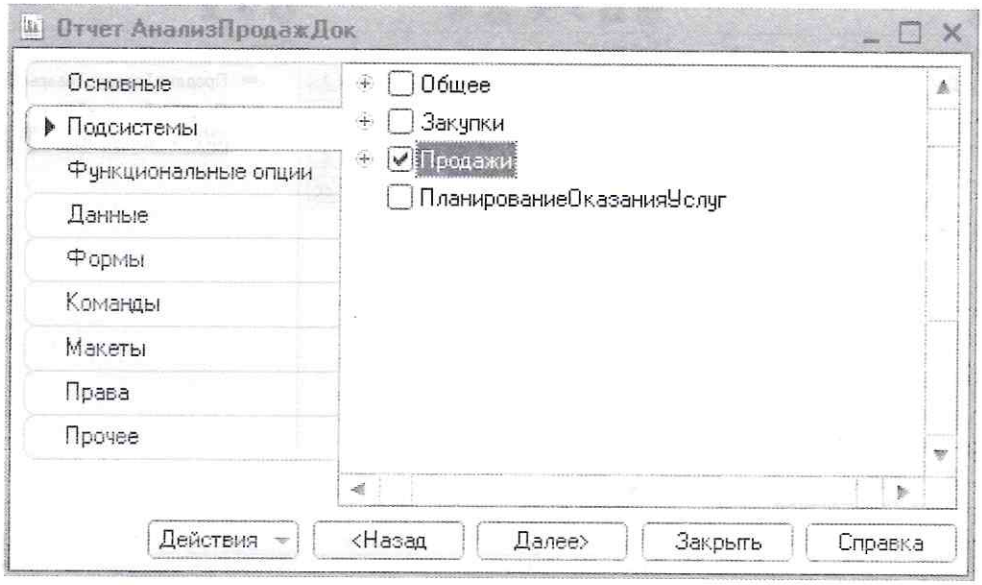


Рис. 4.1. Новый отчет

Вызываем для него конструктор схемы компоновки данных.

В основной схеме компоновки данных добавляем набор данных – запрос.

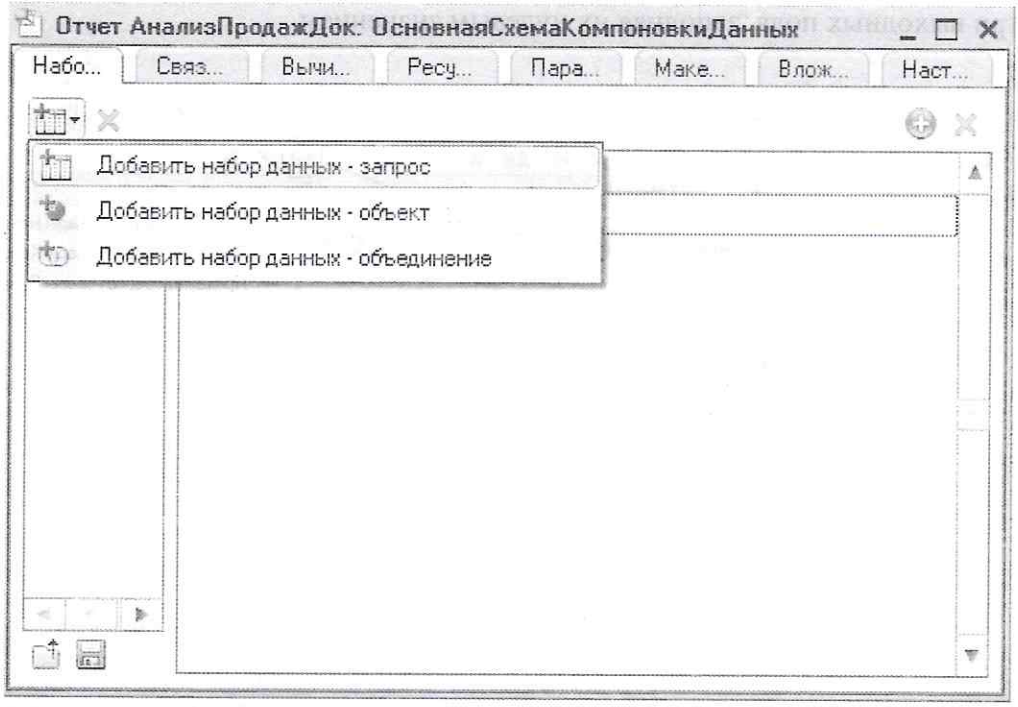


Рис. 4.3. Набор данных - запрос

Далее нажимаем кнопку "Конструктор запроса".

Первый запрос строим по табличной части "Товары" документа "ПродажаТоваров".

В качестве полей нам понадобятся "Номенклатура", "Количество" и "Сумма".

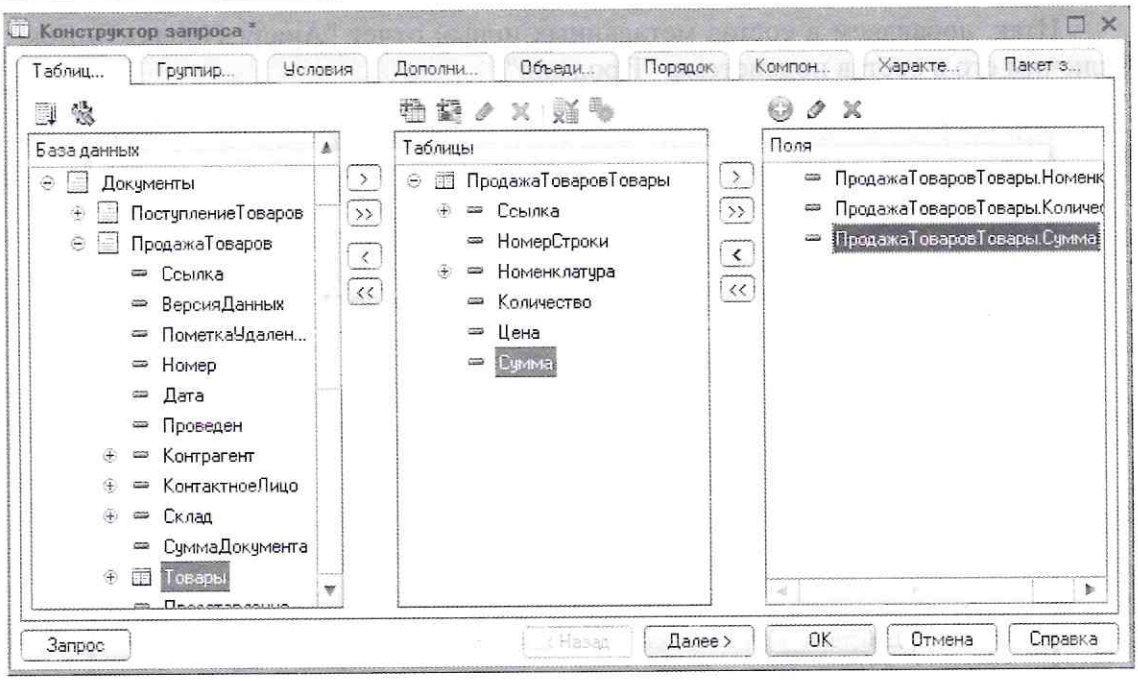


Рис. 4.4. Выходные поля из источника

Кроме того, поскольку в результате объединения результирующая таблица должна содержать еще три поля ("КолПред", "СумПред" и "Ост"), то для них первый запрос должен будет поставлять нулевые значения. Поэтому добавляем еще три выходных поля, заполняя их нулевым значением.

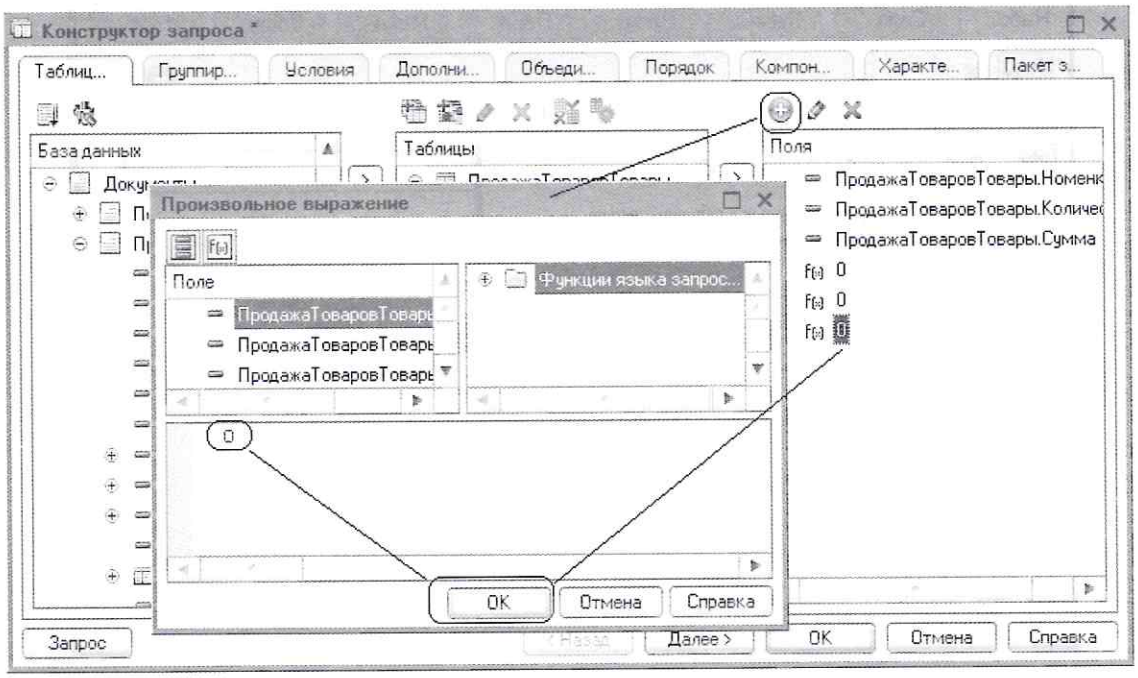


Рис. 4.5. Добавление необходимого количества нулевых полей

Указываем условие, по которому, собственно, и определяем, что данные в первом запросе нужно получать за некий текущий интервал и, кроме того, необходимо собирать данные только по проведенным документам.

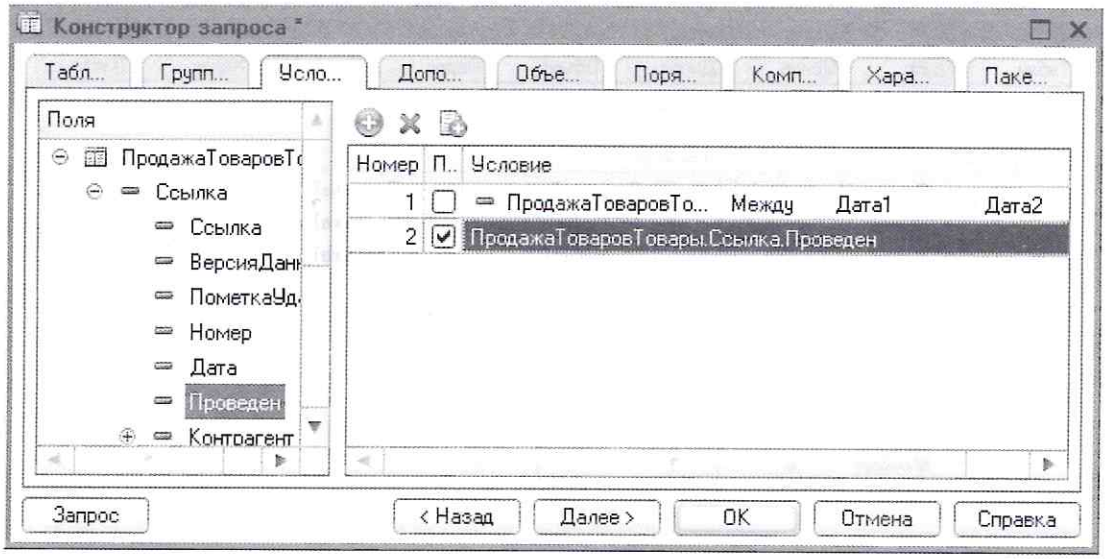


Рис. 4.6. Условия

Запрос по первому источнику мы описали. Чтобы приступить к описанию второго запроса(поскольку с ним мы будем объединяться) , то открываем закладку "Объединения/Псевдонимы". Теперь можем в левом верхнем углу добавить следующий запрос, нажав кнопку "Добавить" или клавишу "Ins".

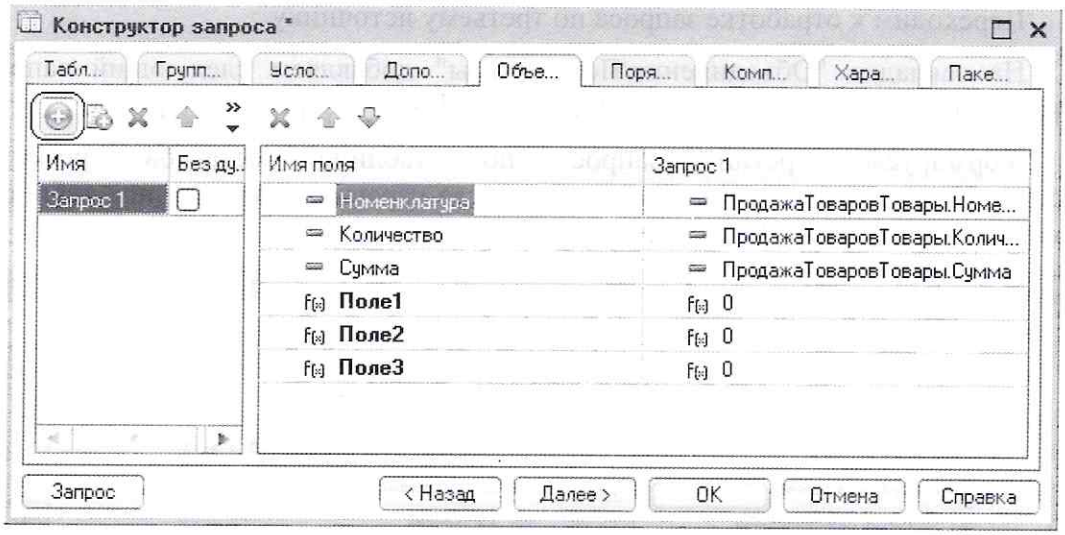


Рис. 4.7. Добавление запроса по следующему источнику

"Запрос2" формируем аналогично первому, то есть тоже три "осмысленных" поля из источника ("Номенклатура", "Количество", "Сумма") и три нулевых поля , чтобы довести количество выходных полей до шести.

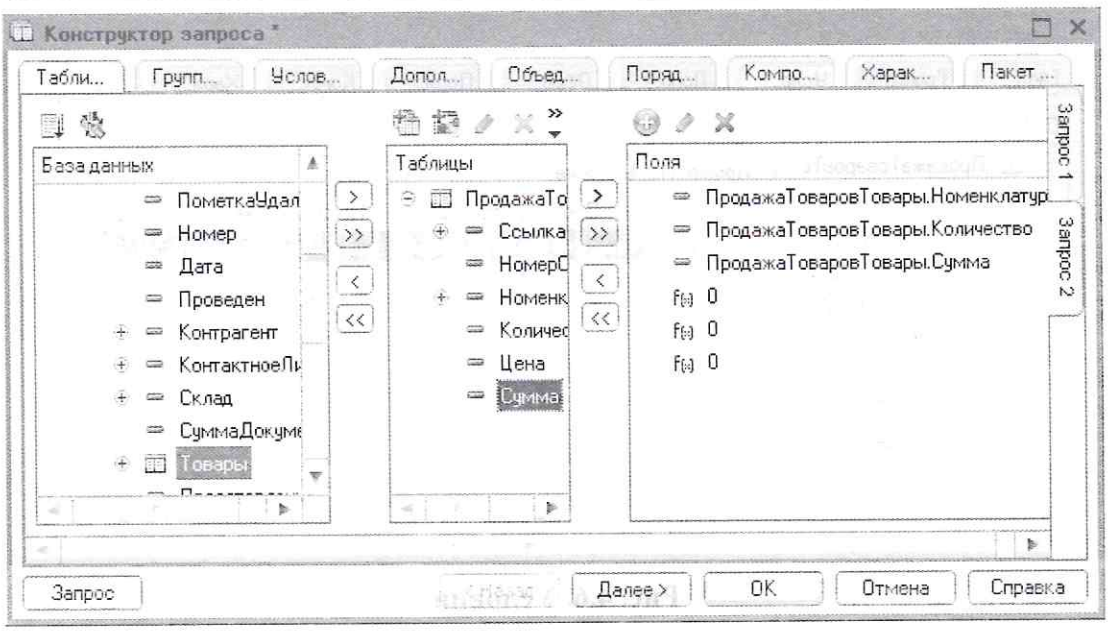


Рис. 4.8. Поля второго запроса

А вот условие будет по другому интервалу, от "Дата3" до "Дата4". И не забудьте указать, что нас интересуют данные только проведенных документов!

Переходим к отработке запроса по третьему источнику.

На закладке "Объединения/Псевдонимы" добавляем следующий запрос, нажав соответствующую кнопку или клавишу "Ins".

Формируем третий запрос по таблице остатков регистра "ОстаткиНоменклатуры". Указываем в качестве параметра виртуальной таблицы, что остатки надо получать на значение "Дата2".

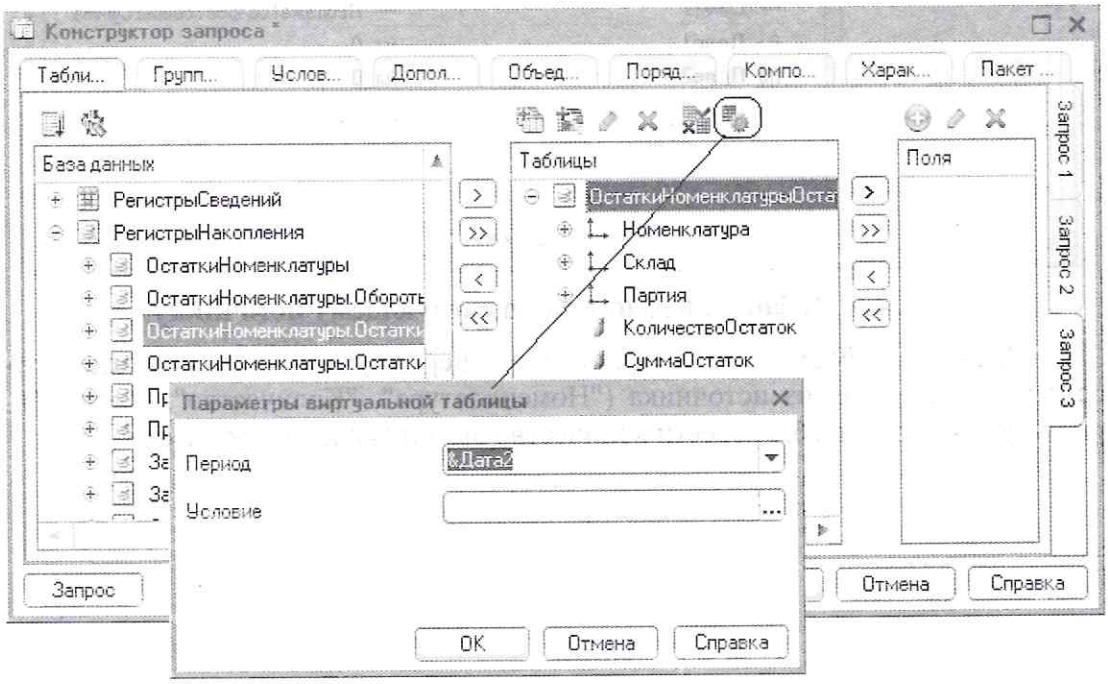


Рис. 4.9. Третий источник

В качестве выходных полей сначала выбираем "осмысленные" из источника, согласно схеме запроса. Потом наступает черед нулевых полей. Сейчас их оказалось уже четыре.

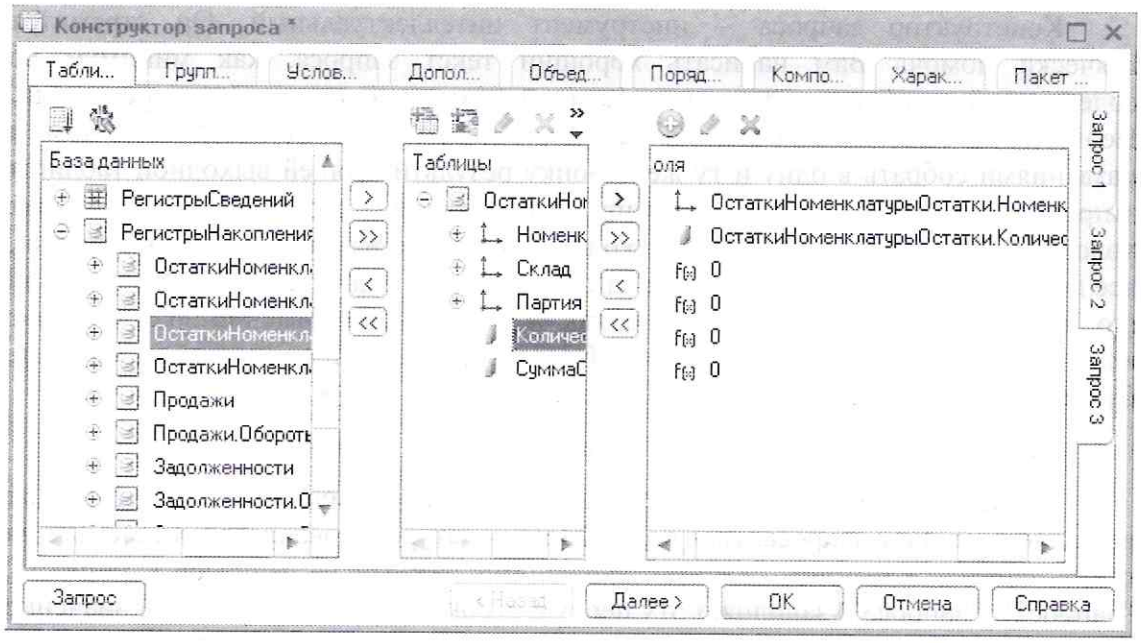


Рис. 4.10. Поля третьего запроса

Теперь нам потребуются предельная аккуратность и внимание. Наступает черед объяснений, какой из запросов, какие поля выходной таблицы чем должен заполнять.

Для этого открываем опять закладку "Объединения/Псевдонимы".

Обратите внимание: основную часть закладки занимает табличное поле, символизирующее выходную таблицу запроса в "положенном на бок" виде. То, что будет колонками выходной таблицы , там отображается в виде строк. Первая строка соответствует первой колонке, вторая – второй и т.д. (см. рис.4.11. "Положенная на бок" выходная таблица запроса).

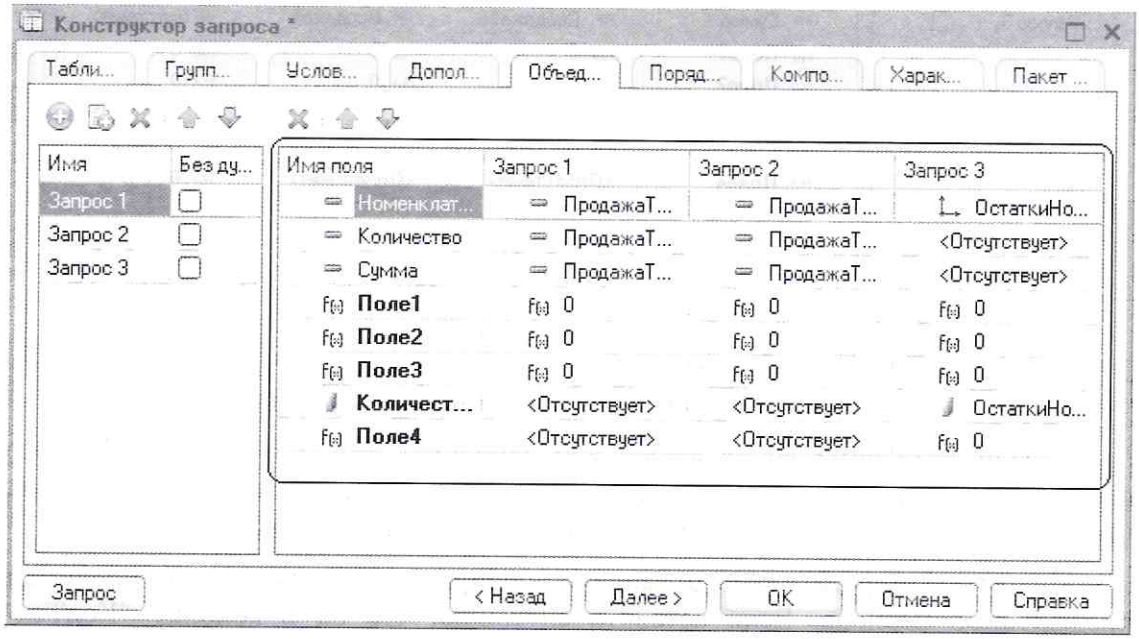


Рис. 4.11. "Положенная на бок" выходная таблица запроса

В колонках же таблицы указаны имя выходного поля и чем каждый из трех запросов это поле собирается заполнять.

Конструктор запроса – инструмент интеллектуальный. Он стремится всячески помочь Вам написать хороший текст запроса, как минимум не содержащий ошибок, а по возможности максимально похожий на требуемый. Поэтому система пыталась поля исходных запросов с одинаковым типом и названиями собрать в одну и ту же колонку результирующей выходной таблицы запроса, а то, что отличалось по типу или названию , разместить в отдельных колонках. Ну а чтобы количество выходных полей в каждом из запросов совпало , все появляющиеся при этом лакуны система пыталась заполнить значениями Null (то, что на экране сейчас обозначено, как <Отсутствует>). Именно поэтому у нас появилось более шести строк в этом табличном поле. Если оставить все, как есть, будет у нас в запросе более шести выходных полей.

Но оставлять-то мы не будем. Давайте наводить порядок.

Начинаем с первой строки, которая соответствует первой колонке будущей выходной таблицы запроса. Имя поля ("Номенклатура") совпадает с желаемым на схеме запроса. Смотрим , какими значениями каждый из запросов ("Запрос1", "Запрос2", "Запрос3") заполняет эту первую строку. Так, все туда кладут значение "Номенклатура". Отлично! Мы только что отработали первую строку, то есть первую колонку будущей выходной таблицы запроса.

Переходим ко второй строке. Судя по схеме , надо изменить имя поля на "КолТек". Делаем двойной клик на имени поля "Количество" и изменяем его на "КолТек".

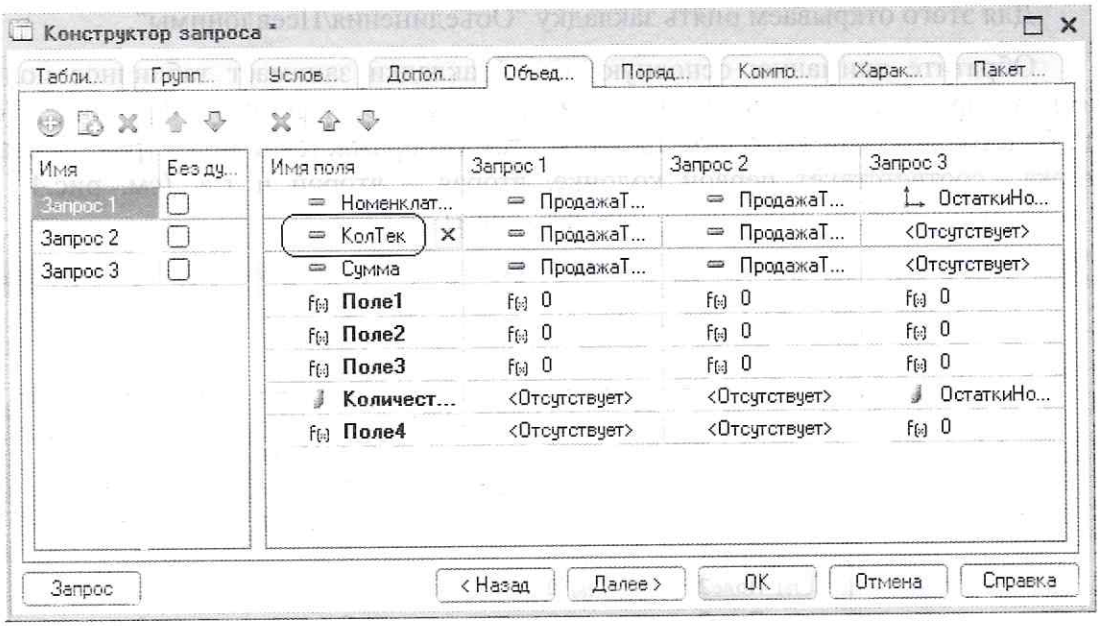


Рис. 4.12. Изменение имени выходного поля

Теперь смотрим, какой запрос чем заполняет это поле. "Запрос1" – это собственно и есть запрос, собирающий текущие продажи, поэтому очень хорошо, что он кладет в это поле значение "ПродажаТоваровТовары.Количество".

А вот "Запрос2" - это запрос, собирающий данные по продажам предыдущего периода. Если оставить ситуацию, что он в колонку "КолТек" кладет "ПродажаТоваровТовары.Количество", то получится, что под видом текущих продаж у нас собираются количество проданного и в текущем, и в прошлом периоде. Но ведь это же неправильно. "Запрос2" должен в колонку "КолТек" класть только нулевые значения и больше ничего! Потому что если он

вдруг положит в эту колонку значение Null, то при подведении итогов наш отчет всегда будет получать Null по этой колонке. Итак, ноль и только ноль!

Делаем двойной клик на значении, заполняющем вторую строку под колонкой "Запрос2", нажимаем на появившуюся кнопочку выбора из списка и в выпавшем списке выбираем первое нулевое значение.

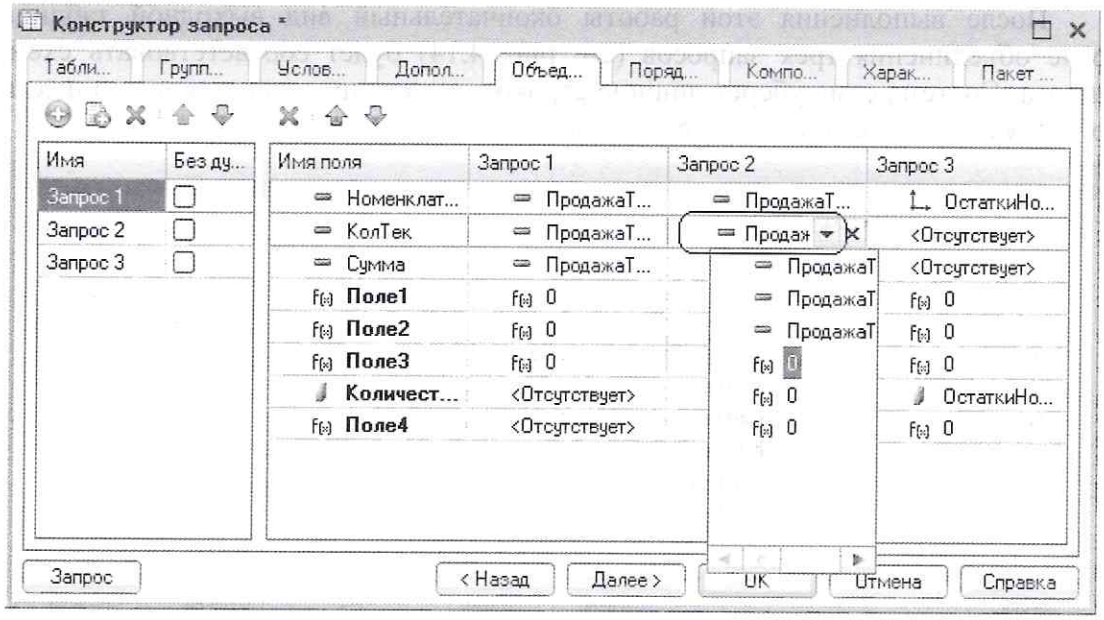


Рис. 4.13. Изменение значения выходного поля

После выбора нажимаем "Enter". Обратите внимание : два поля поменялись местами, то есть там, где раньше лежало первое нулевое значение, теперь лежит "ПродажаТоваровТовары.Количество" и наоборот.

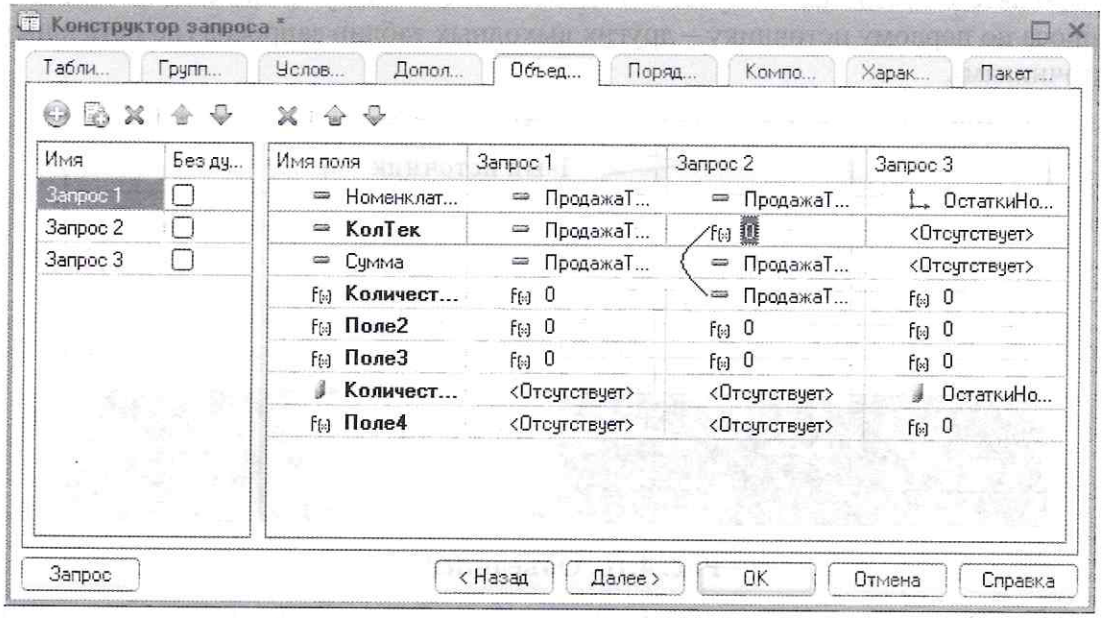


Рис. 4.14. Изменение значения поля обеспечивается переменой мест

Поэтому, впоследствии работая с остальными выходными полями, заполняемыми "Запросом2", важно помнить, что первое нулевое поле уже задействовано и больше его нигде выбирать нельзя. В следующий раз надо будет выбирать второе нулевое поле, потом третье и т.д.

Но закончим с отработкой строки "КолТек". Необходимо, чтобы "Запрос3" также положил в это поле первое нулевое значение, а не <Отсутствует>.

Аналогично отрабатывается третья строка "СумТек" и все остальные строки. Важно сначала указывать псевдоним поля, соответствующий названию на схеме запроса, а потом указывать, какой запрос это поле чем должен заполнить.

После выполнения этой работы окончательный вид выходной таблицы после объединения трех запросов (см. рис. 4.14) будет соответствовать схеме запроса. Система сама уберет лишние строки , то есть их останется ровно шесть, по числу выходных полей на схеме запроса.

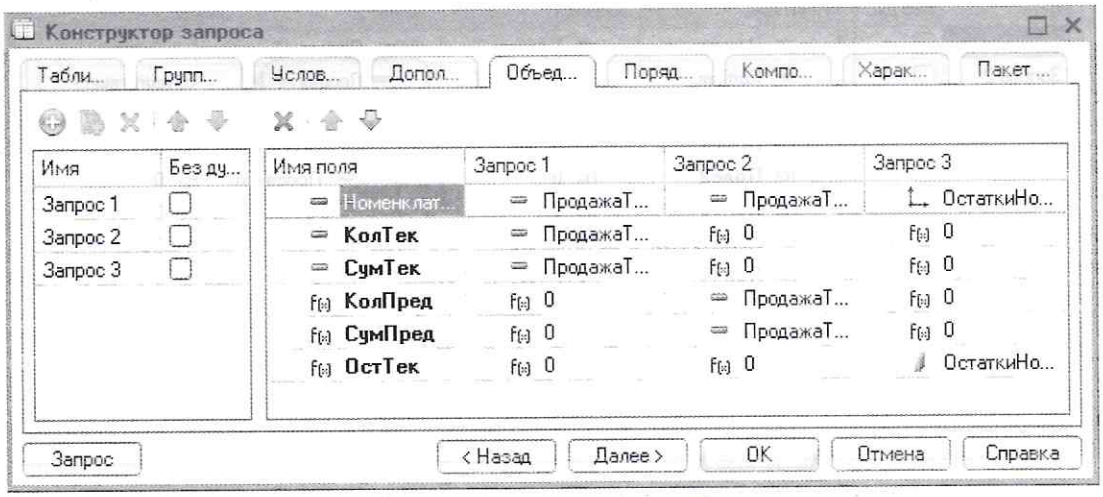


Рис. 4.15. Окончательный вид на закладке Объединения/Псевдонимы

Что дальше? С одной стороны, чудес, вроде, не бывает. Если ничего не предпримем, то результат запроса с объединением в оперативной памяти – это всего-навсего таблица, полученная добавлением снизу к выходной таблице запроса по первому источнику – других выходных таблиц запросов (по остальным источникам).

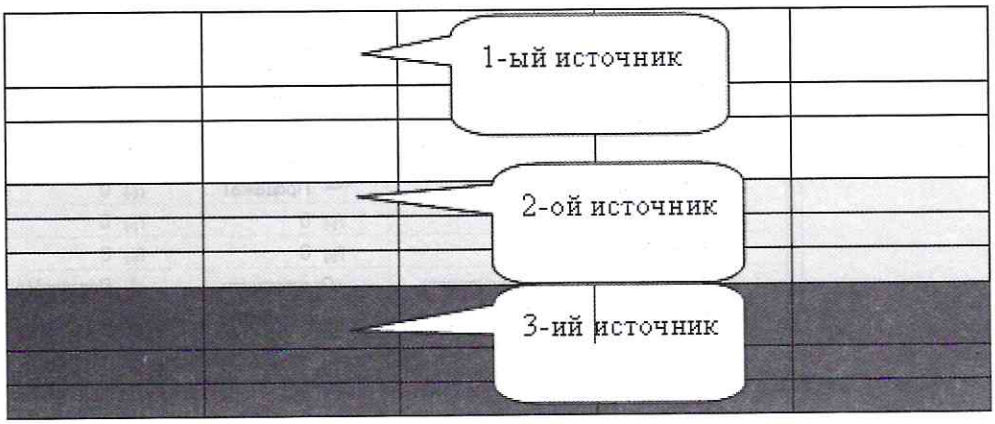


Рис. 4.16. Объединение

То есть, если некий товар присутствовал в каждой из трех таблиц, то он много раз проявится в выходной таблице.

Значит, нам надо сгруппировать (свернуть) выходную таблицу запроса. Но дело в том, что операции группирования идут до объединения и касаются исходных запросов по источникам.

Выходов из создавшегося положения два.

Например, полученный текст запроса можно использовать как текст вложенного запроса. А потом во внешнем запросе данные выходной таблицы вложенного сгруппировать. И если бы речь шла о жестком отчете как о цели нашей работы, то есть чтобы пользователь могу увидеть только то, что находится в оперативной памяти после выполнения запроса , именно такой подход бы мы и применили.

Второй вариант основан на … чуде. Чудом называется система компоновки данных. Дело в том, что этот инструмент предназначен для получения исходного запроса только в качестве "приглашения к разговору" по поводу заполнения настроек. А вот в зависимости от того, какие настройки выбрал пользователь, система умеет приводить отчетные формы в максимально удобный для пользователя вид.

Именно этим мы и воспользуемся.

Итак, нажимаем кнопку "ОК" в окне конструктора запроса.

Можно посмотреть текст запроса, сформированный конструктором. Скорее всего это будет:

```
        ВЫБРАТЬ
        ПродажаТоваровТовары.Номенклатура,
        ПродажаТоваровТовары.Количество КАК КолТек,
        ПродажаТоваровТовары.Сумма КАК СумТек,
        0 КАК КолПред,
        0 КАК СумПред,
        0 КАК ОстТек
ИЗ
        Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
ГДЕ
        ПродажаТоваровТовары.Ссылка.Дата МЕЖДУ &Дата1 И &Дата2
        И ПродажаТоваровТовары.Ссылка.Проведен

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
        ПродажаТоваровТовары.Номенклатура,
        0,
        0,
        ПродажаТоваровТовары.Количество,
        ПродажаТоваровТовары.Сумма,
        0
ИЗ
        Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
ГДЕ
        ПродажаТоваровТовары.Ссылка.Дата МЕЖДУ &Дата3 И &Дата4
        И ПродажаТоваровТовары.Ссылка.Проведен

ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
        ОстаткиНоменклатурыОстатки.Номенклатура,
```

```
            0,
            0,
            0,
            0,
            ОстаткиНоменклатурыОстатки.КоличествоОстаток
ИЗ
            РегистрНакопления.ОстаткиНоменклатуры.Остатки(&Дата2, )
                                    КАК ОстаткиНоменклатурыОстатки
```

Теперь открываем закладку "Ресурсы" и выбираем все поля, содержащие числовые данные для нашего отчета (см.рис.4.16.Ресурсы СКД).

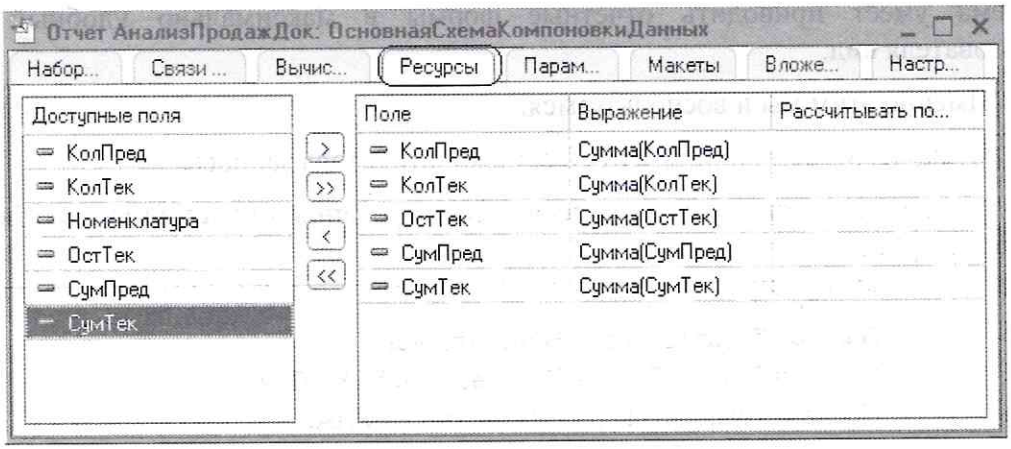


Рис. 4.16. Ресурсы СКД

Теперь наводим порядок с параметрами. Мы хотим, чтобы виртуальная таблица регистра строилась именно на значение, взятое из &Дата2. Кроме того, нужно дать пользователю возможность указывать значения для всех четырех временных параметров.

Поэтому открываем закладку "Параметры" и приводим ее содержимое к виду, как на рис. 4.17.Параметры СКД.

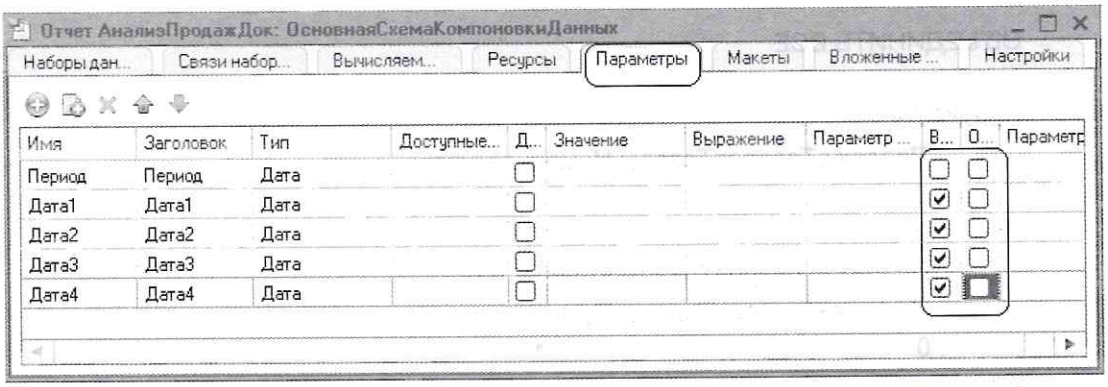


Рис. 4.17. Параметры СКД

Потом открываем закладку "Настройки" и отрабатываем визуализацию параметров для пользователя. Для этого сначала расставляем флажки отображения на форме каждого из параметров ("Дата1", "Дата2", "Дата3" и "Дата4"). Затем, последовательно активизируя каждый из этих параметров, нажимаем кнопку "Пользовательские настройки элемента". И каждый раз в открывающемся окне ставим флажок "Включать в пользовательские настройки" (см. рис. 4.18. Параметры выходной формы).

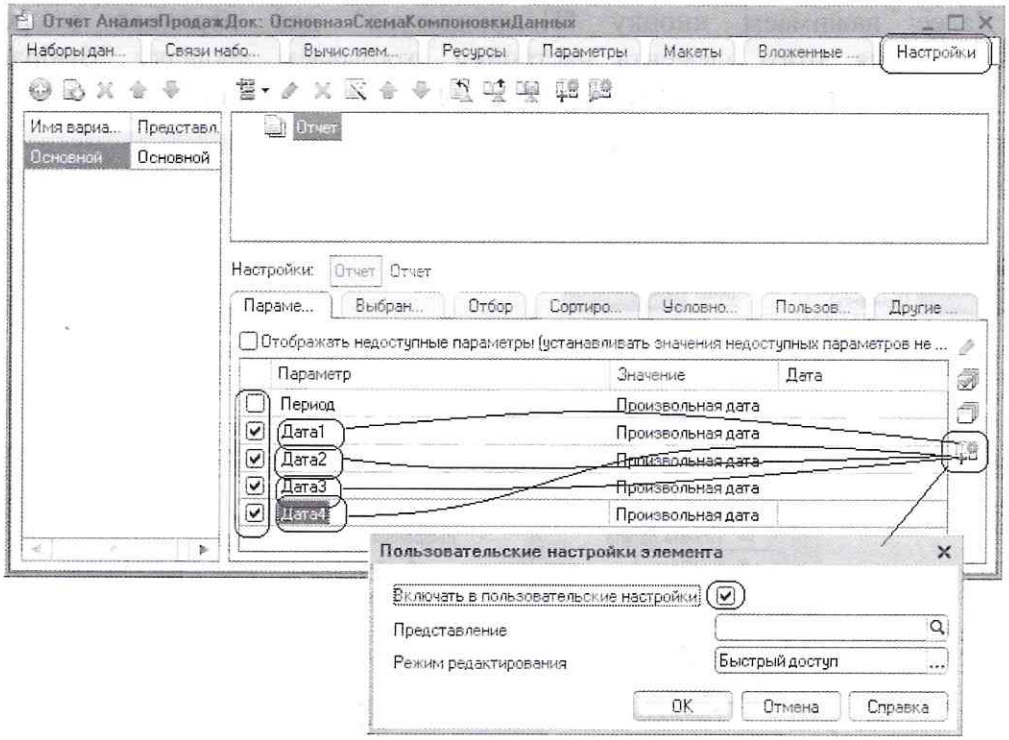


Рис. 4.18. Параметры выходной формы

Теперь добавляем новый элемент настройки – группировку.

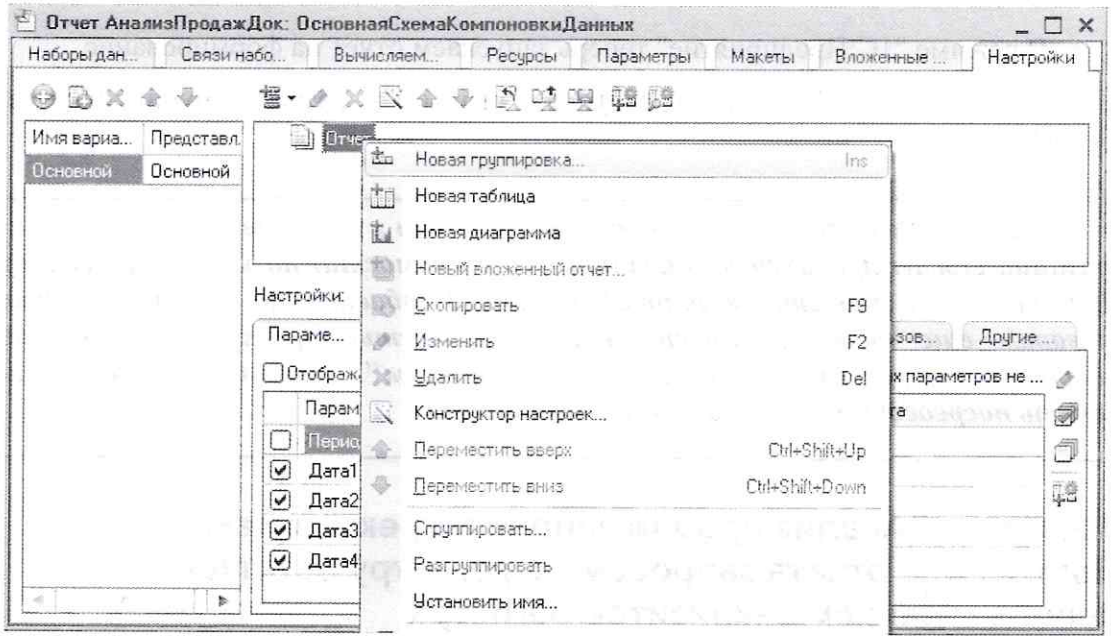


Рис. 4.19. Добавление группировки в качестве элемента настройки

В качестве значения группировки выбираем поле "Номенклатура".

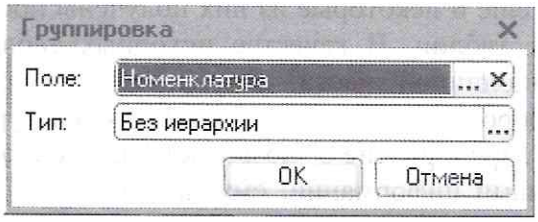


Рис. 4.20. Заполнение значения группировки

Далее нажимаем кнопку "Номенклатура" для перехода в режим редактирования этого элемента настройки, открываем закладку "Выбранные поля", выбираем и упорядочиваем все нужные поля согласно исходному эскизу.

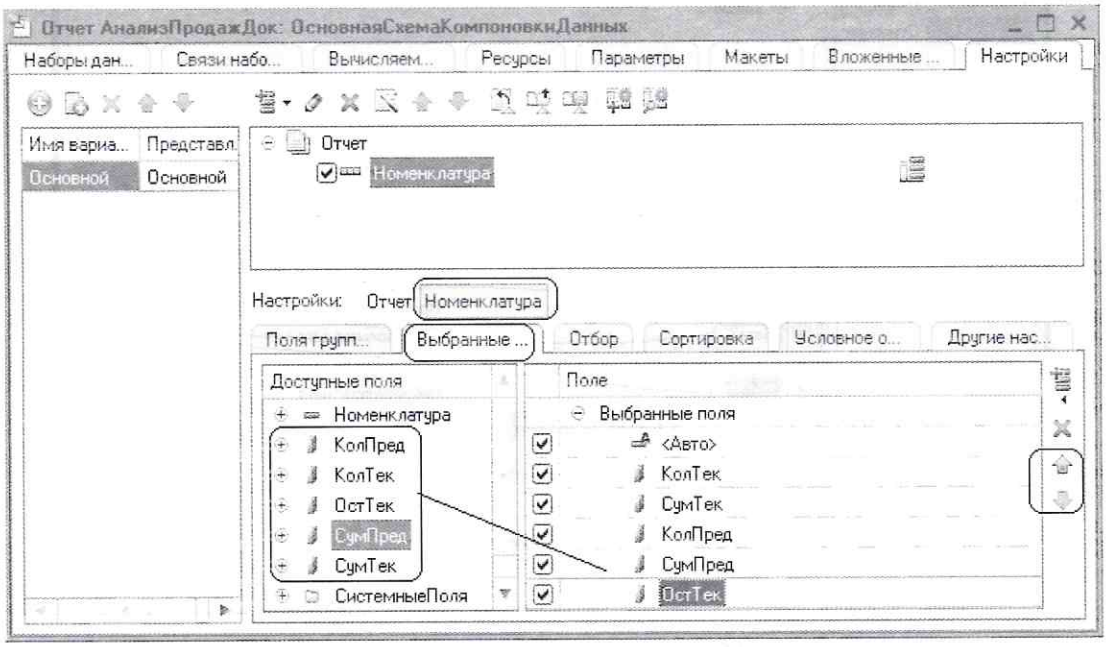


Рис. 4.21. Заполнение выходных полей группировки

Отчет готов!

В режиме "1С:Предприятие" теперь запускаем отчет на формирование.

Как видите , СКД сама свернула данные в разрезе группировки по номенклатуре.

Практикум №12

К данному отчету необходимо добавить колонки с вычислением "Прибыли" (разница суммы продажи и списанной себестоимости) по каждому периоду. Для этого Вам придется в Запрос1 и Запрос2 добавить по имеющимся там товарам еще поля со списанной себестоимостью (расход) из таблицы оборотов по регистру "ОстаткиНоменклатуры". Подсказка: лучше это делать посредством левого внешнего соединения.

4.2. Отчет "Анализ продаж запрос по реквизитам" . Построение отчета запросом по регистру остатков с использованием реквизитов регистра

Особенно после выполнения последнего практикума виден определенный недостаток предыдущего способа получения отчета – мы объединяли три исходных запроса, да еще и некоторые из них получены посредством соединения нескольких исходных таблиц. В качестве исходных таблиц у нас выступали Таблица "Товары" табличной части документа "ПродажаТоваров", таблица остатков и таблица оборотов регистра "ОстаткиНоменклатуры". И хотя механизм запросов системы "1С:Предприятие 8" позволяет достаточно быстро работать и со значительными объемами информации, смущает большое количество исходных таблиц.

Попробуем добиться повышения производительности за счет оптимизации структуры конфигурации.

Что если все данные выбрать из таблиц регистра "ОстаткиНоменклатуры"?

Можно получить значение конечного остатка. Можно получить значение количества проданного товара, но вот сумму проданного товара, пока получить не можем. Регистр работает только с себестоимостью.

Что если внести новый ресурс "СуммаПродажи"?

Если так сделать, получим грубейшую ошибку в работе с регистром остатков.

Напомним, что регистр остатков в качестве ресурсов накапливает данные по показателям остатка, то есть по таким показателям, по которым принципиально есть положительные и отрицательные движения. Итоги каждого ресурса регистра остатков рано или поздно должны иметь возможность выводиться "в ноль".

"СуммаПродажи" не является показателем остатка. Если документ "ПродажаТоваров" будет выполнять по нему движения как по ресурсу, то некому делать движения с обратным знаком, некому принципиально выводить подобный ресурс в ноль!

В результате при хранении итогов будет неоправданно разбухать физически существующая таблица итогов, опираясь на данные которой "1С:Предприятие 8" действительно оперативно выдает остатки по регистру.

Таким образом, мы вышли на определенное ограничение. Пользуясь ресурсами регистра остатков ,нельзя учитывать показатели движения оборотов!

Отсюда же, кстати, вытекает и второе ограничение:

Нельзя по какому-либо из ресурсов регистра остатков делать рассогласованные по набору измерений движения.

То есть, например, при проведении приходной накладной указывать значения измерений "Номенклатура", "Партия", а при проведении расходной – только "Номенклатура". В результате по конкретным партиям будут только накапливаться положительные значения ресурса, по пустой партии – только отрицательные, то есть вместо хранения данных одного показателя остатка храним значения двух показателей движения.

Но что же делать, если данные о сумах продажи все же хочется получать из регистра "ОстаткиНоменклатуры"?

Для учета характеристик движения на регистрах остатков можно пользоваться "Реквизитами" регистра.

При использовании реквизитов информация об их значениях прописывается только в таблицу записей регистра и не используется в физически существующей таблице итогов. В результате значения реквизитов могут быть получены запросом из основной таблицы регистра и как угодно обработаны. Например , использованы в качестве группировок или исходных данных для расчета функций. При этом не грозит разбухание физической таблицы хранения итогов.

Итак, изменяем структуру регистра накопления "ОстаткиНоменклатуры", вносим в нее реквизит "СуммаПродажи". Тип значения , конечно, <Число>. А поскольку учитываем деньги – точность – два знака после запятой.

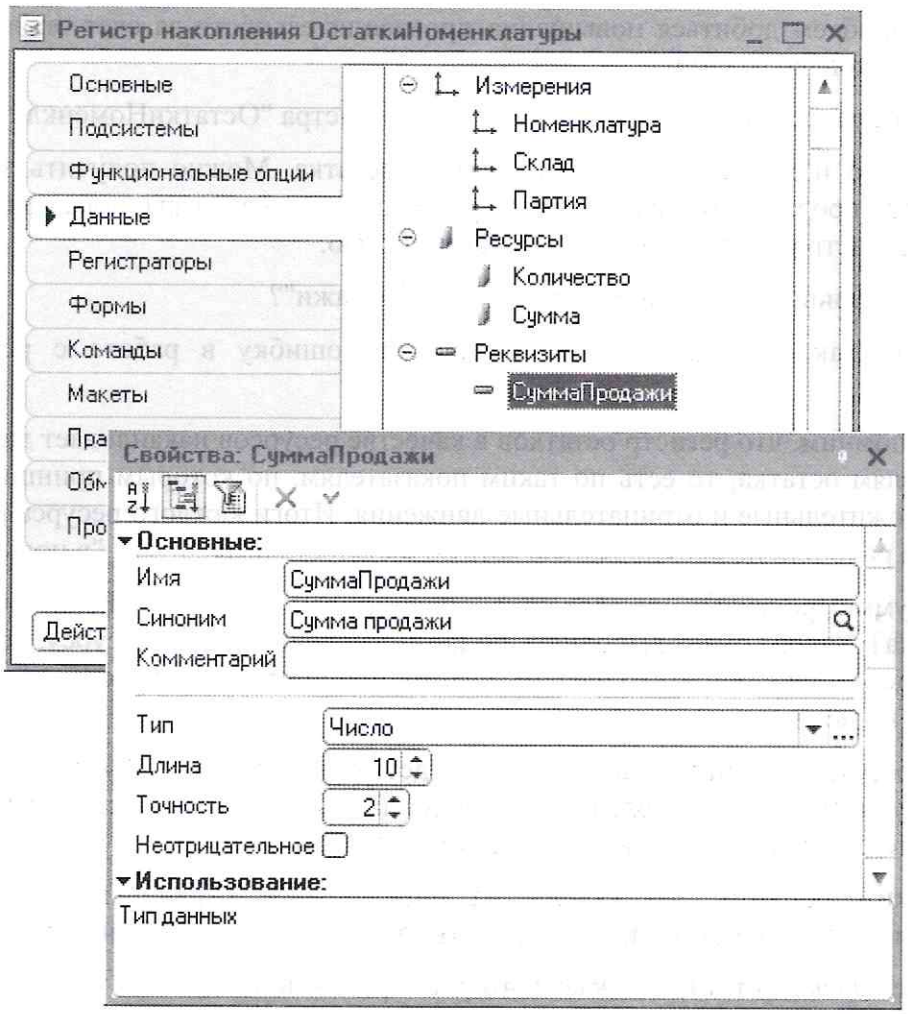


Рис. 4.25. Добавление реквизита

Внесем изменения в механизм формирования движений по регистру "ОстаткиНоменклатуры" для документа "ПродажаТоваров".

Напомним, сейчас у нас это выполняется функцией "СписатьСебестоимостьТоваровДокумента" в модуле менеджера документа "ПродажаТоваров".

Для начала придется в запросе добавить еще одно выходное поле "СуммаДок".

```
ВЫБРАТЬ
ПродажаТоваровТовары.Номенклатура КАК Номенклатура,
ПродажаТоваровТовары.Ссылка.Склад КАК Склад,
СУММА(ПродажаТоваровТовары.Количество) КАК Количество,
ПродажаТоваровТовары.Ссылка.Дата,
СУММА(ПродажаТоваровТовары.Сумма) КАК Сумма
ПОМЕСТИТЬ ТабДок
ИЗ
Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
ГДЕ
ПродажаТоваровТовары.Ссылка = &Ссылка
```

```
        И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры <>
ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)

        СГРУППИРОВАТЬ ПО
        ПродажаТоваровТовары.Номенклатура,
        ПродажаТоваровТовары.Ссылка.Склад,
        ПродажаТоваровТовары.Ссылка.Дата

        ИНДЕКСИРОВАТЬ ПО
        Номенклатура,
        Склад
        ;

        ////////////////////////////////////////////////////////////////////////////////
        ВЫБРАТЬ
        ТабДок.Дата,
        ТабДок.Склад,
        ТабДок.Номенклатура КАК Номенклатура,
        ТабДок.Количество КАК Количество,
        ОстаткиНоменклатурыОстатки.Партия,
        ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) КАК
КоличествоОстаток,
        ЕСТЬNULL(ОстаткиНоменклатурыОстатки.СуммаОстаток, 0) КАК СуммаОстаток,
        ТабДок.Сумма КАК СуммаДок
        ИЗ
        ТабДок КАК ТабДок
                ЛЕВОЕ СОЕДИНЕНИЕ
                        РегистрНакопления.ОстаткиНоменклатуры.Остатки(
                                &Момент,
        ...
        ИТОГИ
        МАКСИМУМ(Количество),
        СУММА(КоличествоОстаток),
        СУММА(СуммаОстаток),
        МАКСИМУМ(СуммаДок)
        ПО
        Номенклатура
```

Далее вносим изменения во фрагмент кода, формирующий набор записей для регистра. При этом обязательно учтем, что проведение документа происходит по партиям. А значит, для каждой партии в качестве суммы продажи надо прописывать ее долю:

```
...
//формируем набор записей
Движение = НаборЗаписей.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
Движение.Период = ВыборкаДетальныеЗаписи.Дата;
Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
Движение.Склад = ВыборкаДетальныеЗаписи.Склад;
Движение.Партия = ВыборкаДетальныеЗаписи.Партия;
Движение.Количество = КоличествоКСписанию;
Движение.Сумма = СтоимостьКСписанию;

Если ВыборкаНоменклатура.Количество<>0 Тогда
        Движение.СуммаПродажи =
                        КоличествоКСписанию*
                        ВыборкаНоменклатура.СуммаДок/
                           ВыборкаНоменклатура.Количество;
КонецЕсли;
...
```

Что теперь надо делать?

Правильно, надо обеспечить перезаполнение данных регистра "ОстаткиНоменклатуры". Для этого лучше всего воспользоваться обработкой "СписаниеСебестомостиДляДокументовПродажи" за интервал, заведомо включающий в себя все документы нашей учебной базы.

Вот теперь можно переходить к формированию отчета "АнализПродажПоРегОстаткиНоменклатуры".

Технология его полностью повторяет технологию создания предыдущего отчета. Даже схема запроса остается практически той же, только в качестве исходных таблиц для "Запрос1" и "Запрос2" берем основную таблицу регистра "ОстаткиНоменклатуры".

Сделайте самостоятельно этот новый отчет, пожалуйста.

Небольшая подсказка: при построении условий, кроме как по дате, поставим еще по виду движения, чтобы в отчет не мешались приходы.

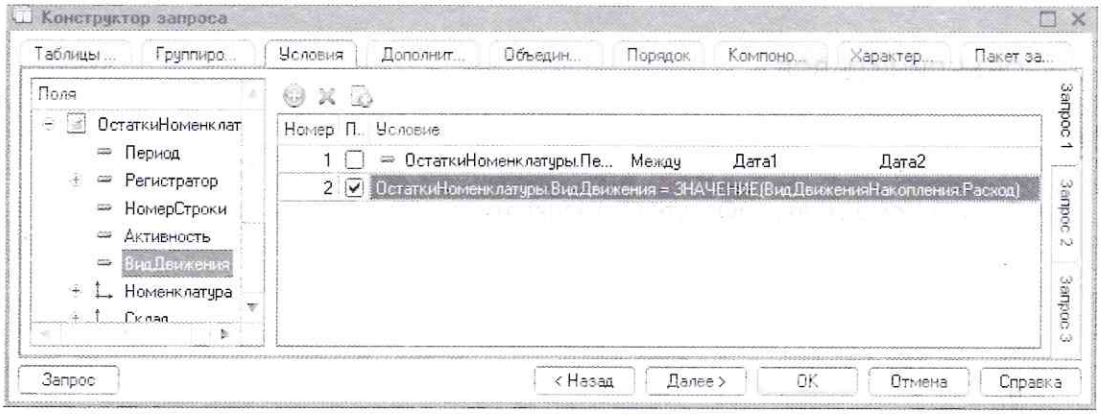


Рис. 4.26. Условия

Результат работы конструктора будет, скорее всего, представлен следующим текстом запроса:

```
ВЫБРАТЬ
    ОстаткиНоменклатуры.Номенклатура,
    ОстаткиНоменклатуры.Количество КАК КолТек,
    ОстаткиНоменклатуры.СуммаПродажи КАК СумТек,
    0 КАК КолПред,
    0 КАК СумПред,
    0 КАК ОстТек
    ИЗ
    РегистрНакопления.ОстаткиНоменклатуры КАК ОстаткиНоменклатуры
    ГДЕ
    ОстаткиНоменклатуры.Период МЕЖДУ &Дата1 И &Дата2
    И ОстаткиНоменклатуры.ВидДвижения =
                ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход)

    ОБЪЕДИНИТЬ ВСЕ

    ВЫБРАТЬ
    ОстаткиНоменклатуры.Номенклатура,
    0,
    0,
    ОстаткиНоменклатуры.Количество,
    ОстаткиНоменклатуры.СуммаПродажи,
    0
    ИЗ
    РегистрНакопления.ОстаткиНоменклатуры КАК ОстаткиНоменклатуры
    ГДЕ
    ОстаткиНоменклатуры.Период МЕЖДУ &Дата3 И &Дата4
    И ОстаткиНоменклатуры.ВидДвижения =
                ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход)
```

```
ОБЪЕДИНИТЬ ВСЕ

ВЫБРАТЬ
ОстаткиНоменклатурыОстатки.Номенклатура,
0,
0,
0,
0,
ОстаткиНоменклатурыОстатки.КоличествоОстаток
ИЗ
РегистрНакопления.ОстаткиНоменклатуры.Остатки(&Дата2, )
                                    КАК ОстаткиНоменклатурыОстатки
```

При попытке выполнения отчет даст те же результаты, что и его собрат в предыдущей главе.

"Ну и в чем повышение эффективности?" – спросите Вы. ☺

А вот чтобы увидеть его, выполним практикум:

Практикум №13

К данному отчету необходимо добавить колонки с вычислением "Прибыли" (разница суммы продажи и списанной себестоимости) по каждому периоду. Но для этого опять же изменим структуру регистра – добавим еще один реквизит "Прибыль". И при проведении расходной накладной будем "бросать" туда разницу между суммой продажи по данной партии и ее себестоимостью. Тогда на этапе формирования запроса данные можно будет взять все из той же основной таблицы регистра "ОстаткиНоменклатуры"!

Как видите, использование запросов по регистрам гораздо технологичнее как с точки зрения количества задействованных таблиц-источников, так и с точки зрения простоты будущей реализации задач расширения функциональности системы.

4.3. Отчет "АнализПродажПоОборотномуРегистру". Построение отчета запросом по регистру "Продажи"

И все равно что-то смущает в производительности механизма формирования последнего отчета. Уж слишком много лишней информации пришлось отсекать. Ведь в основной таблице регистра хранятся данные как по расходам, так и по приходам товара.

Как бы тут еще повысить скорость формирования отчета?

Возвращаемся к началу курса. Правильно , самое надежное средство повышения оперативности формирование отчета – это наличие в системе отчетных показателей, сразу хранимых в соответствующем регистре.

Если вернуться к теме "Определение видов показателей", то легко определить, что показателям "СуммаПроданного" и "КоличествоПроданного" будет соответствовать оборотный регистр, ведь это показатели оборотов!

Итак, строим наш отчет третьим способом – посредством использования оборотного регистра.

Оборотный регистр накопления "Продажи" должен остаться еще от предыдущего курса "Введение в конфигурирование…".

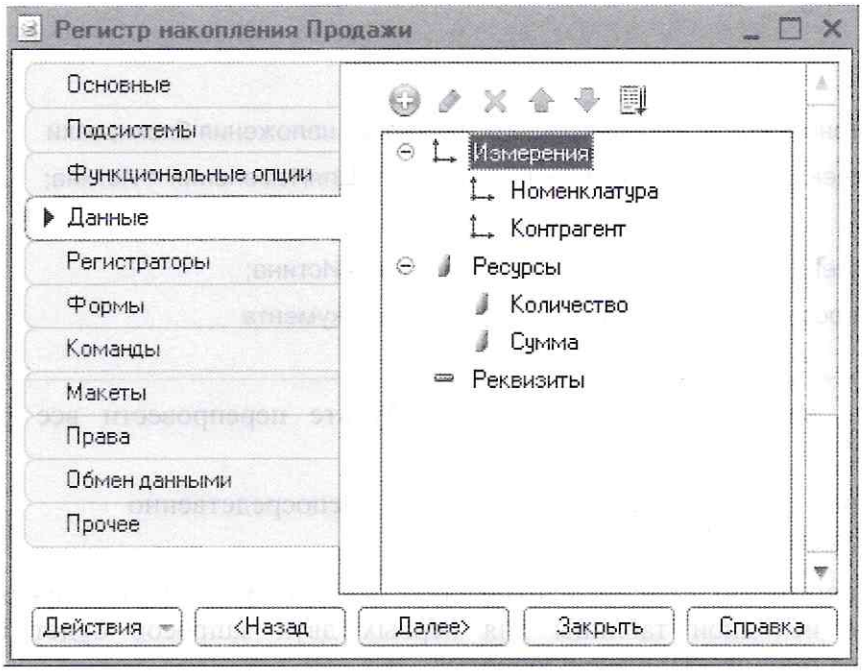


Рис. 4.27. Регистр "Продажи"

Для формирования движений по нему можно изменить процедуру проведения документа "ПродажаТоваров". Например, так:

```
Процедура ОбработкаПроведения(Отказ, Режим)

    Движения.Задолженности.Записывать = Истина;
    // регистр Задолженности Приход
    Движение = Движения.Задолженности.Добавить();
    Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
    Движение.Период = Дата;
    Движение.Контрагент = Контрагент;
    Движение.СуммаДолга = СуммаДокумента;

    Движения.БронированиеТоваров.Записывать = Истина;

    // регистр Продажи
    Движения.Продажи.Записывать = Истина;
    Для Каждого ТекСтрокаТовары Из Товары Цикл
        // регистр Продажи
```

```
            Движение = Движения.Продажи.Добавить();
            Движение.Период = Дата;
            Движение.Контрагент = Контрагент;
            Движение.Номенклатура = ТекСтрокаТовары.Номенклатура;
            Движение.Количество = ТекСтрокаТовары.Количество;
            Движение.Сумма = ТекСтрокаТовары.Сумма;
        КонецЦикла;


        //Установка флага последующего при записи наложения блокировки
        Движения.СвободныеОстатки.БлокироватьДляИзменения = Истина;

        Движения.СвободныеОстатки.Записывать = Истина;
        //Запрос для определения только товаров документа
        ...
```

После внесения изменений не забудьте перепровести все расходные накладные!

Теперь строим непосредственно отчет "АнализПродажПоОборотномуРегистру".

Технология остается абсолютно та же, как и в предыдущих случаях, только в качестве исходной таблицы для первых двух запросов будет выступать виртуальная таблица "ПродажиОбороты".

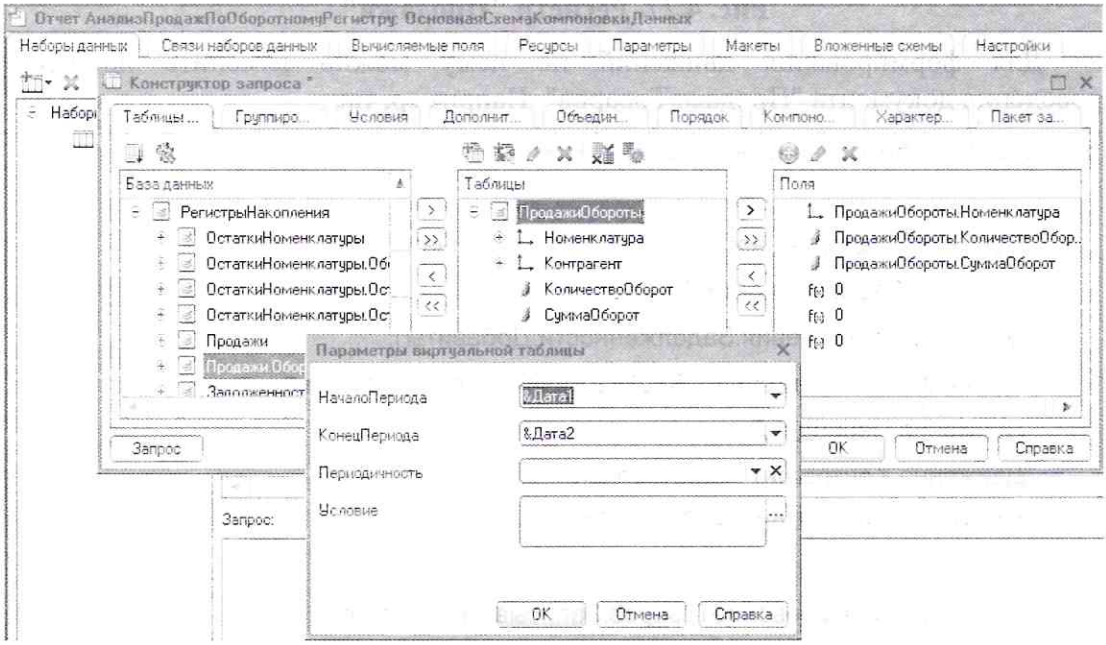


Рис. 4.28. Работа с виртуальной таблицей

Выставляем параметры виртуальной таблицы еще на этапе ее формирования, определяем поля выходной таблицы.

Далее на закладке "Объединения/Псевдонимы" добавляем второй запрос:

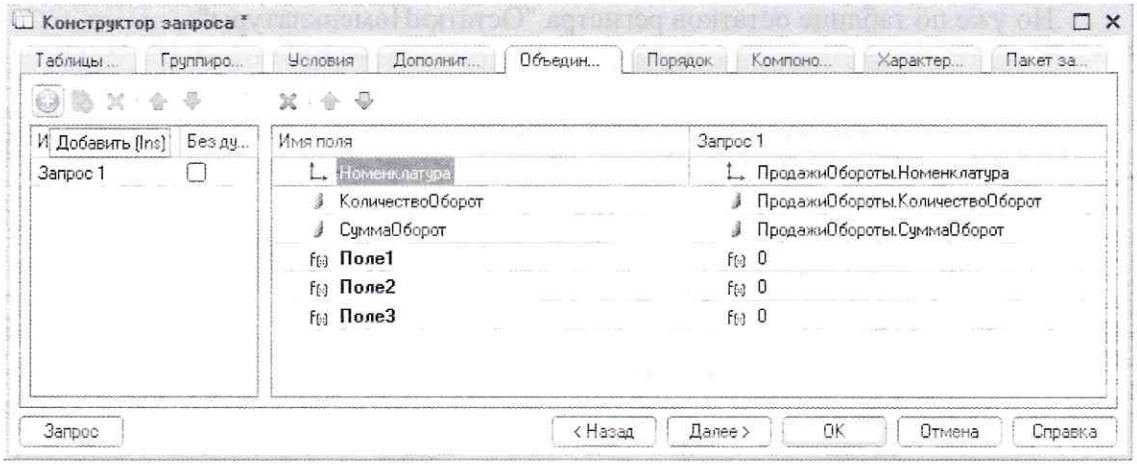


Рис. 4.29. Объединения/Псевдонимы

И аналогично отрабатываем "Запрос2", но уже с другими временными параметрами формирования.

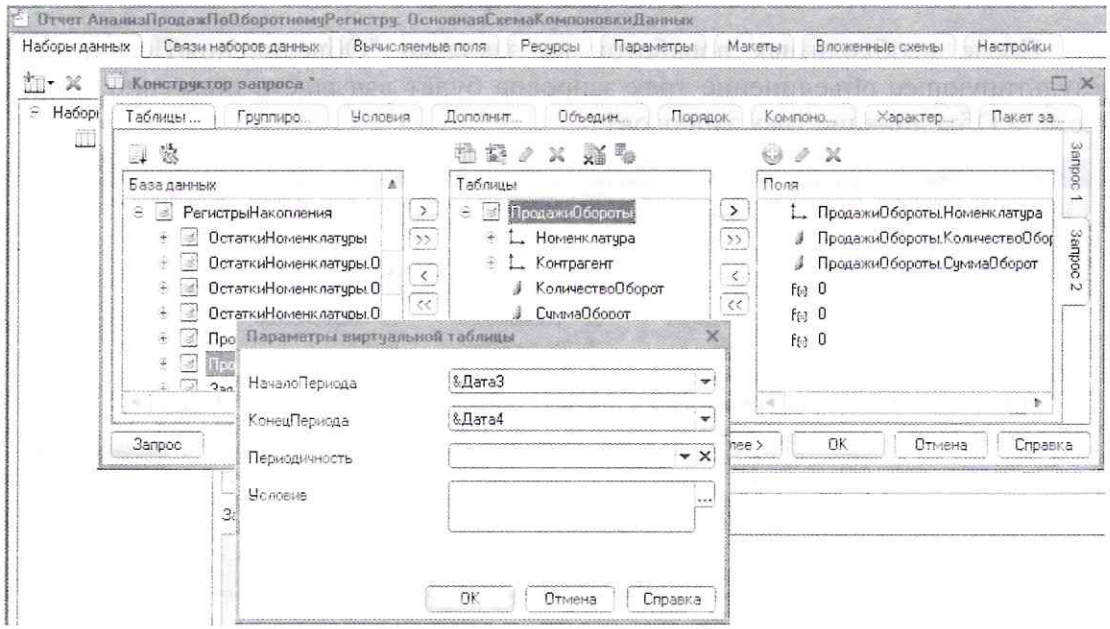


Рис. 4.30. Запрос 2

Потом добавляем аналогично "Запрос3".

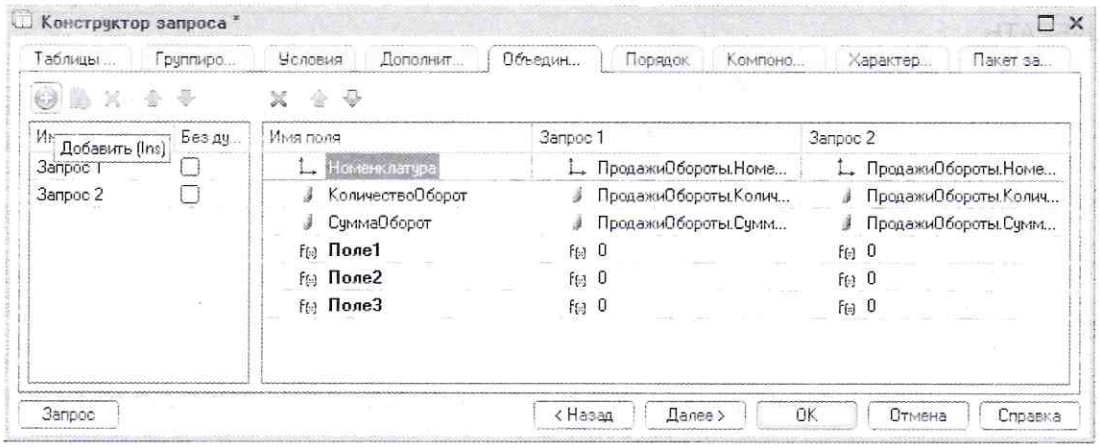


Рис. 4.31. Объединения/Псевдонимы

Но уже по таблице остатков регистра "ОстаткиНоменклатуры".

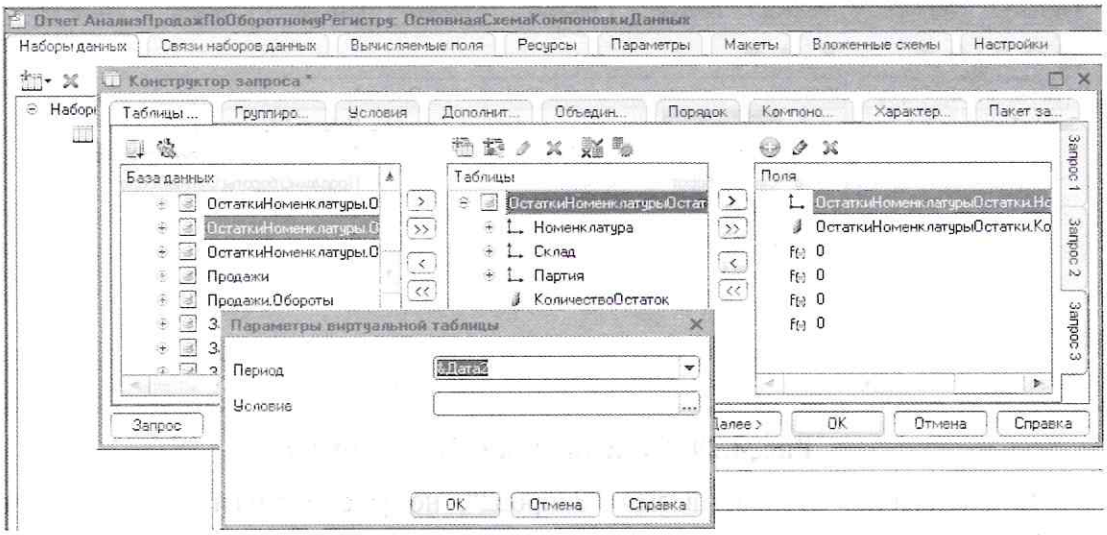


Рис. 4.32. Запрос 3

После необходимых доработок на закладке "Объединения/Псевдонимы" результирующее объединение трех запросов будет выглядеть примерно так (см. рис.4.33.Объединение трех запросов):

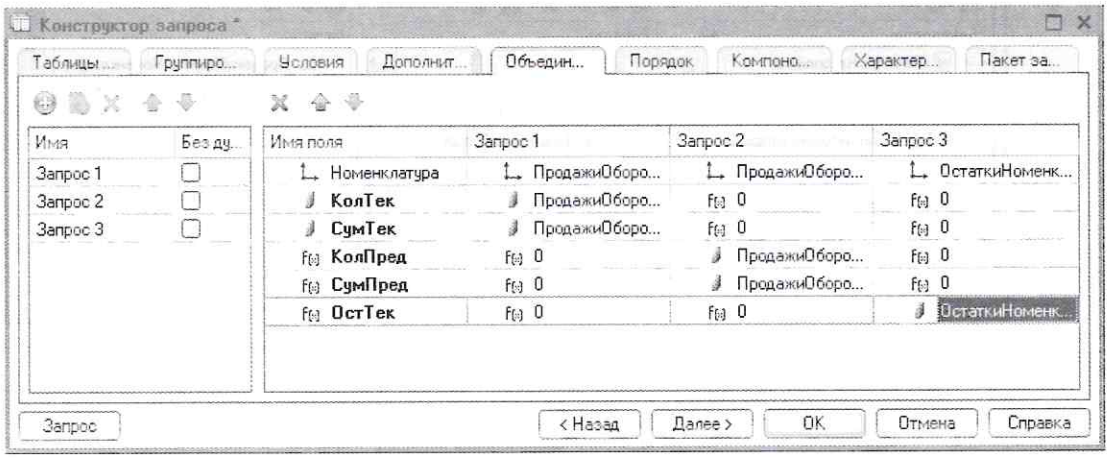


Рис. 4.33. Объединение трех запросов

Результат работы конструктора будет, скорее всего, представлен следующим текстом запроса:

```
ВЫБРАТЬ
	ПродажиОбороты.Номенклатура,
	ПродажиОбороты.КоличествоОборот КАК КолТек,
	ПродажиОбороты.СуммаОборот КАК СумТек,
	0 КАК КолПред,
	0 КАК СумПред,
	0 КАК ОстТек
	ИЗ
	РегистрНакопления.Продажи.Обороты(&Дата1, &Дата2, , ) КАК ПродажиОбороты

ОБЪЕДИНИТЬ ВСЕ
```

```
        ВЫБРАТЬ
        ПродажиОбороты.Номенклатура,
        0,
        0,
        ПродажиОбороты.КоличествоОборот,
        ПродажиОбороты.СуммаОборот,
        0
        ИЗ
        РегистрНакопления.Продажи.Обороты(&Дата3, &Дата4, , ) КАК ПродажиОбороты

ОБЪЕДИНИТЬ ВСЕ

        ВЫБРАТЬ
        ОстаткиНоменклатурыОстатки.Номенклатура,
        0,
        0,
        0,
        0,
        ОстаткиНоменклатурыОстатки.КоличествоОстаток
        ИЗ
        РегистрНакопления.ОстаткиНоменклатуры.Остатки(&Дата2, )
                                        КАК ОстаткиНоменклатурыОстатки
```

После создания необходимой настройки и выполнения результат внешне будет тот же, но получен он будет гораздо быстрее!

Практикум №14

Как вы уже догадались, к данному отчету необходимо добавить колонки с вычислением "Прибыли" (разница суммы продажи и списанной себестоимости) по каждому периоду. Для этого опять же изменим структуру регистра "Продажи" – добавим еще один ресурс "Прибыль".

Заполните, пожалуйста, и это поле при формировании движений по регистру "Продажи". Тогда на этапе формирования запроса данные можно будет взять все из той же таблицы оборотов регистра "Продажи"!

4.4. Варианты структурной оптимизации оборотных регистров

Мы разрабатывали конфигурацию для небольшого предприятия с не очень большой аналитикой отчетов.

Но ведь вполне могут иметь место ситуации, когда число разрезов аналитики исчисляется десятками, включая также дополнительные разрезы, касающиеся периодичности демонстрации этой аналитики.

Разберем два разных механизма оптимизации хранения данных оборотных регистров в базе данных. Ведь, как мы убеждаемся на протяжении всего курса, именно оптимизация хранения готовых значений показателей в регистре – наиболее эффективный способ обеспечения оперативности.

4.4.1. Работа с итогами. Исключение неважных измерений из таблицы итогов

Можно практически разделять разрезы учета оборотных показателей на:

а) требующие быстрой обработки;

б) не требующие быстрой обработки.

Например, в состав нашего регистра "Продажи" для измерения "Контрагент" можно снять флажок "Использование в итогах".

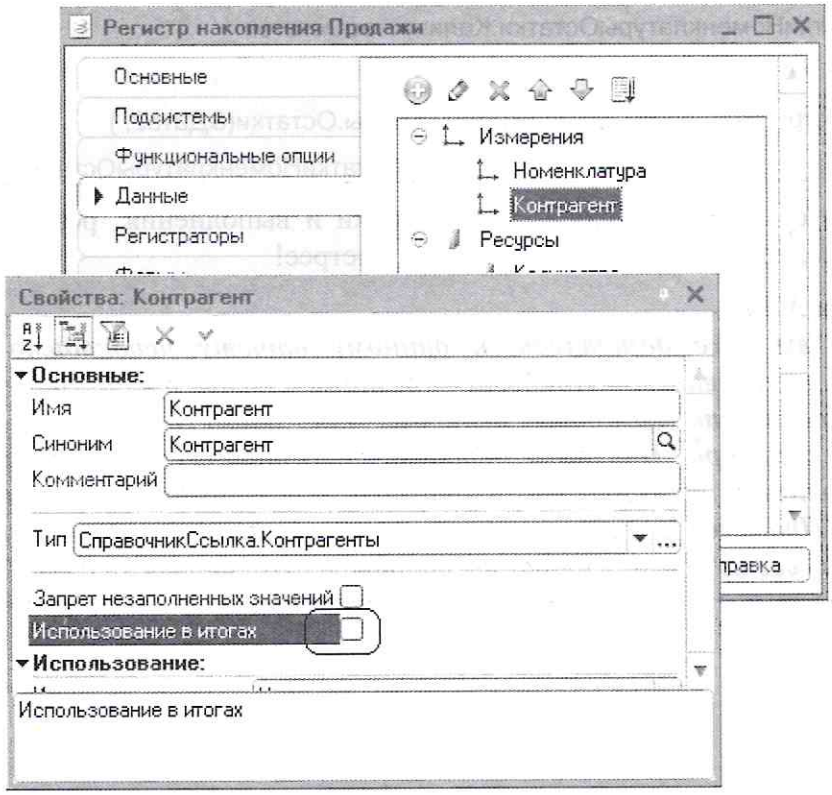


Рис. 4.30. Свойство "Использование в итогах"

В результате впоследствии виртуальная таблица "Обороты" позволит нам работать с полем измерения "Контрагент" (в отличие от поля реквизита регистра). НО в таблице хранения итогов такого поля не будет, то есть ежемесячные

автоматически вычисляемые итоги не будут вычисляться в разрезе контрагентов! И таблица хранения итогов в базе данных станет значительно меньшего размера!

Рассмотрим это на примере.

Допустим, в таблицу движений нашего оборотного регистра попали такие записи, как приведенные в таблице 4.1.

Таблица 4.1. Таблица записей регистра

Период (записи)	Номенклатура	Контрагент	Количество	Сумма
01.01.2010	Груша	Шанс, ООО	1	6
01.01.2010	Яблоко	Шанс, ООО	10	100
03.01.2010	Груша	Шанс, ООО	2	12
25.01.2010	Яблоко	Шанс, ООО	3	30
30.01. 2010	Яблоко	МИРС, ЗАО	100	800
28.02. 2010	Груша	Шанс, ООО	3	18

Тогда в ситуации, если оба измерения используются в итогах, таблица итогов будет хранить такую информацию, как в табл.4.2.

Таблица 4.2. Таблица итогов, если оба измерения используются

Период (итога)	Номенклатура	Контрагент	Количество	Сумма
Январь 2010	Груша	Шанс, ООО	3	18
Январь 2010	Яблоко	МИРС, ЗАО	100	800
Январь 2010	Яблоко	Шанс, ООО	13	130
Февраль 2010	Груша	Шанс, ООО	3	18

Если же для измерения "Контрагент" отключено использование в итогах , то в базе данных будет храниться такая таблица итогов нашего регистра, как в табл.4.3.

Таблица 4.3. Таблица итогов, если отключено использование итогов у "Контрагента"

Период (итога)	Номенклатура	Количество	Сумма
Январь 2010	Груша	3	18
Январь 2010	Яблоко	113	930
Февраль 2010	Груша	3	18

Мало того что таблица итогов занимает тогда меньше места в базе данных, так еще и те отчеты, которые не требуют разрезов по контрагенту, будут строиться очень быстро, поскольку таблица итогов будет уже заранее содержать итоги только по номенклатуре.

А те отчеты, которые нужно строить не столько быстро, сколько подробно, смогут быть получены посредством все той же виртуальной таблицы "Продажи.Обороты" просто с использованием поля "Контрагент". То, что при этом не будет задействована таблица накопления итогов регистра, на точности полученного результата не скажется.

Данный прием – отказ от использования в итогах неважных измерений – особенно эффективен в ситуации, когда количество измерений превышает восемь. Дело в том, что внутри базы данных для таблицы итогов регистра накопления система автоматически создает индекс в порядке следования первых восьми измерений. Да и подсчет самих итогов с большим количеством измерений сопряжен с обработкой большого количества комбинаций значений этих измерений (да-да, все то же старое доброе "декартово произведение") ,то есть очень сильно "раздувает" таблицу итогов в базе данных со всеми вытекающими последствиями.

Итак, в ситуации создания систем с аналитическими отчетами различной детальности, но с максимальной детальностью более чем в восемь разрезов есть смысл подумать о возможности разделения измерений на два класса:

а) требующие быстрой обработки;

б) не требующие быстрой обработки.

4.4.2. Работа с агрегатами

Анализируя примеры в материале выше , можно прийти к заключению, что использование таблицы итогов для регистра накопления имеет ряд жестких ограничений. Во-первых, периодичность промежуточных итогов всегда месяц. Во-вторых, индексирование таблицы итогов всегда строится в порядке следования измерений.

Кроме того, есть еще и "в-третьих". Таблица итогов оборотного регистра обязательно заполняется или видоизменяется в момент формирования движений. А поскольку чаще всего движения формируются при проведении документов , то чаще всего система при проведении документов пытается что-то делать в таблице итогов нашего регистра. Если для регистра включено разделение итогов (для снижения уровня конкуренции при наложении блокировок записей) , это

приводит к еще большему, чем мы рассматривали выше, "разрастанию" таблицы итогов.

Все эти факторы обернутся против оперативности нашей системы в том случае, если мы имеем дело с реально большой базой, то есть где количество ежемесячно попадающих в таблицу движения записей или количество комбинаций значений измерений в хранимых итогах составляют десятки и сотни тысяч. При меньших объемах и существующие механизмы использования итогов весьма оперативно справляются со всеми своими задачами, даже с взаимопротиворечивыми.

Например:

Некоторые из пользователей часто строят отчет "АнализПродажПоОборотномуРегистру" сравнивая текущий месяц с прошлым.

Для других пользователей при выписке документов продажи надо оперативно видеть, на какую сумму этому контрагенту уже прошли отгрузки в этом квартале.

Для третьих пользователей требуется мгновенно оценивать совокупные объемы продаж номенклатурных позиций в количественном выражении за весь период существования нашего предприятия.

Сама идея оперативности получения данных о состоянии показателей за счет хранения этих показателей уже в готовом виде в базе данных говорит нам о том, что по-хорошему на каждое из таких требований нужна своя таблица итогов. Да еще и в некоторых случаях с периодичностью, отличной от той, что может предоставить регистр накопления оборотов. Пока таких таблиц итогов нет, очень часто системе придется считать данные для построения виртуальной таблицы "Обороты" по реальной таблице движений регистра.

Итак, представим себе, что у нас очень и очень большая база с сильно заполненным регистром "Продажи".

В таком случае для повышения оперативности обслуживания аналитических задач можно вообще отказаться от использования штатной таблицы итогов в пользу использования таблиц "Агрегатов".

Что такое агрегаты? Это по сути те же самые таблицы итогов, но у которых состав разрезов и периодичность хранения показателей задает разработчик. То есть при использовании агрегатов разработчик имеет право самостоятельно задавать, какие и какого состава таблицы оборотов должны храниться в базе данных.

Например, для рассмотренных выше требований неплохо было бы иметь в системе такие агрегаты:

Таблица 4.4. Таблица агрегата с разрезами "ПериодМесяц", "Номенклатура"

ПериодМесяц	Номенклатура	Количество	Сумма
Январь 2010	Груша	3	18
Январь 2010	Яблоко	113	930
Февраль 2010	Груша	3	18

Таблица 4.5. Таблица агрегата с разрезами "ПериодКвартал", "Контрагент"

ПериодКвартал	Контрагент	Количество	Сумма
I кв. 2010	Шанс, ООО	19	166
I кв. 2010	МИРС, ЗАО	100	800

Таблица 4.6. Таблица непериодического агрегата с разрезом "Номенклатура"

Номенклатура	Количество	Сумма
Груша	6	36
Яблоко	113	930

Как этого добиться?

Эксплуатация агрегатов в системе обычно разделяется на три вопроса:

1) **Создание агрегатов,** то есть добавление новых агрегатов и описание их состава. При этом можно указывать , должен этот агрегат использоваться "Всегда" или "Авто".

2) **Перестроение сети агрегатов.** Если для ряда агрегатов использование определено как "Авто", то система имеет право согласно накопленной статистике реального использования информации при выполнении запросов принимать решение, стоит ли тратить ресурсы на заполнение и поддержание того или иного из этих агрегатов. По некоторым может быть принято решение о временном "выводе из эксплуатации", если они никак в последнее время не помогали запросам, по крайней мере до следующего перестроения сети агрегатов.

3) **Обновление агрегатов.** В отличие от механизма итогов само пополнение таблиц агрегатов новой информацией происходит не в момент формирования движений по регистру, а когда будет запущен специальный процесс обновления агрегатов. Не пугайтесь, корректность данных запросов при этом не пострадает. Просто, если система чего-то не найдет в агрегатах, она это недостающее найдет уже в таблице движений регистра.

Ну а теперь более подробно рассмотрим вопросы обслуживания агрегатов.

4.4.2.1 Создание агрегатов

Если ранее для нашего регистра агрегаты не использовались, создание их начинается с создания базового агрегата, то есть агрегата, включающего все измерения и периодичность "Авто".

Для этого в окне редактирования объекта конфигурации регистра накопления "Продажи" открываем закладку "Данные" и нажимаем кнопку "Агрегаты" - мы вызовем окно конструктора агрегатов.

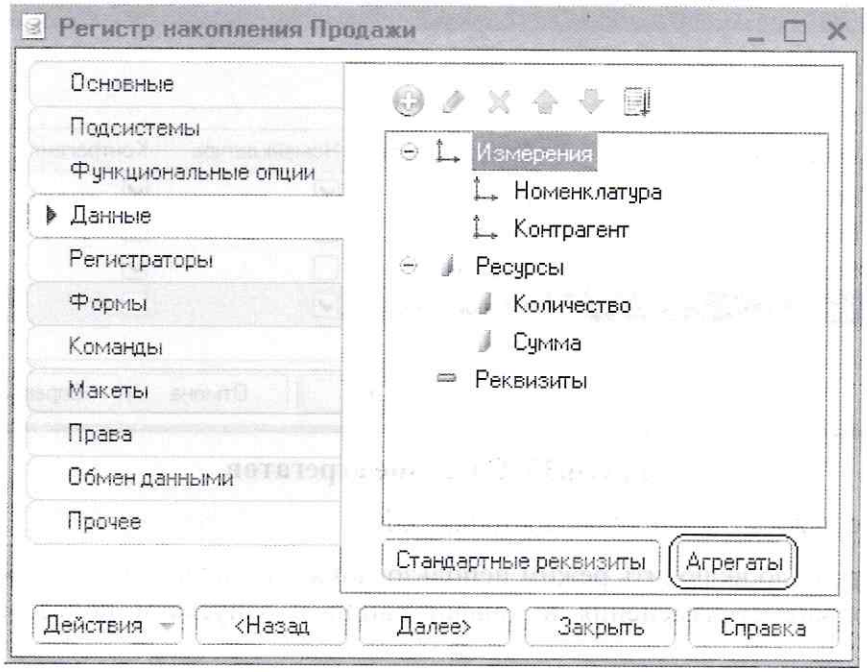


Рис. 4.31. Вызов окна конструктора агрегатов

В открывшемся окне нажимаем кнопку "Добавить" и в появившейся строке агрегата проставляем флажки на каждом из измерений.

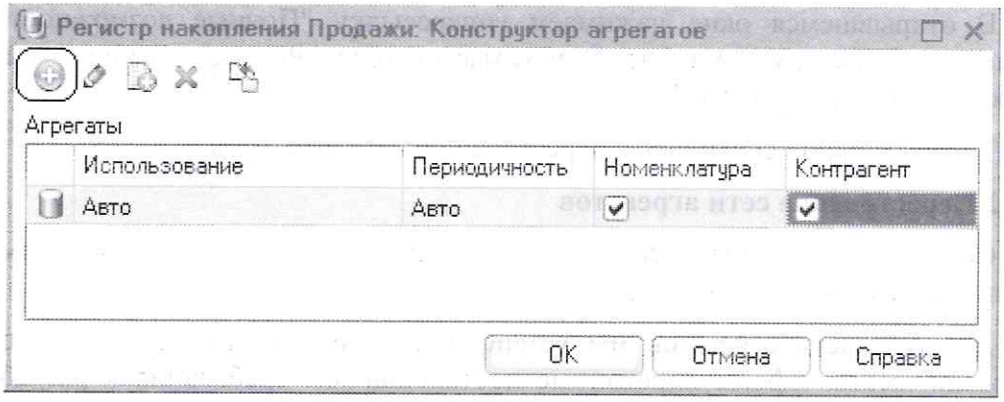


Рис. 4.32. Создание базового агрегата

Кроме того, вспомним исходную задачу. Надо максимально оперативно обслуживать следующие виды обращений к оборотам:

Некоторые из пользователей часто строят отчет "АнализПродажПоОборотномуРегистру", сравнивая текущий месяц с прошлым.

Для других пользователей при выписке документов продажи надо оперативно видеть, на какую сумму этому контрагенту уже прошли отгрузки в этом квартале.

Для третьих пользователей требуется мгновенно оценивать совокупные объемы продаж номенклатурных позиций в количественном выражении за весь период существования нашего предприятия.

Поэтому добавляем еще три агрегата, но для них свойство использование ставим уже в положение "Всегда" (см.рис.4.33.Создание агрегатов):

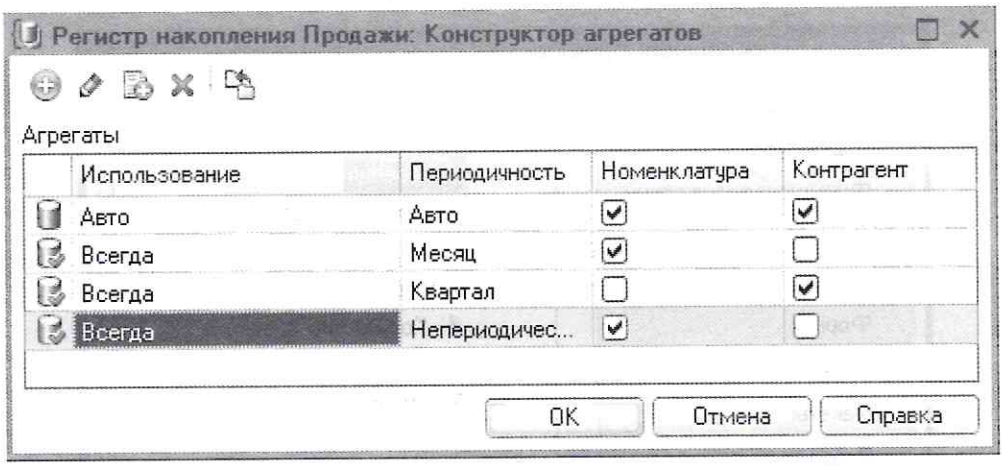


Рис. 4.33. Создание агрегатов

Далее нажимаем внизу окна кнопку "ОК".

Теперь надо включить режим использования агрегатов для нашего регистра, потому сохраняем изменения в конфигурации и запускаем пользовательский режим.

Доступ к включению использования агрегатов в пользовательском режиме через команду "Все функции".

Далее выбираем пункт "Управление итогами" в разделе "Стандартные".

В открывшемся окне нажимаем гиперссылку "Полные возможности" и переходим на закладку "Агрегаты", нажимаем кнопку "Режим" и выбираем пункт "Включить режим агрегатов".

Теперь можно переходить к перестроению сети агрегатов.

4.4.2.2 Перестроение сети агрегатов

Создание агрегатов для системы еще не означает необходимость обязательность их использования.

Да, для части агрегатов мы установили использование "Всегда", но это диктовали жестко сформулированные постановщиком требования к разработке системы.

Ниже мы увидим, что система и сама не прочь поделиться советами, какие еще агрегаты оптимально бы добавить. А вот для этих "насоветованных" агрегатов чаще всего будет стоять свойство "Использование" в положении "Авто". А раз так , то система сможет сама управлять вопросом подключения или отключения этих агрегатов (еще говорят: "перестраивать сеть агрегатов") в зависимости от того, какая статистика обращений к данным регистра собралась на текущий момент.

Вызывать перестроение сети агрегатов можно программно. Например, при помощи регламентного задания, в котором используется команда:

```
РегистрыНакопления.Продажи.ПерестроитьИспользованиеАгрегатов()
```

Периодичность расписания такого регламентного задания можно сделать раз в неделю или раз сутки, лучше всего ночью, потому что для большой базы длительность выполнения этой операции может исчисляться десятками минут.

Но можно и интерактивно. Благо, учебная база у нас небольшая. Да и удобно, что нужное окно уже открыто.

Нажимаем кнопку "Перестроить…"

В появившемся окне запроса параметров перестроения сейчас лучше всего просто нажать кнопку "ОК"".

В результате система приступит непосредственно к перестроению и заполнению агрегатов. А в конце это работы покажет относительный эффект применения агрегатов. В моем случае он оказался равным 19%. Это значит, что согласно статистике уже выполненных в процессе эксплуатации запросов применение такой конфигурации агрегатов дает выигрыш в 19% по сравнению с использованием режима итогов.

4.4.2.3 Обновление агрегатов

Хотелось бы обратить внимание на то, что использование агрегатов оборотного регистра – это альтернатива использования итогов оборотного регистра. Если используется механизм агрегатов – отключается механизм итогов. И наоборот, возврат к использованию механизма итогов отключает использование механизма агрегатов.

Дело в том, что слишком уж они разные с точки зрения выполняемых системой действий.

При задействовании механизма итогов информация в таблицу итогов попадает в момент отработки системой записи набора записей в таблицу движений.

При задействовании механизма агрегатов в момент отработки системой записи набора записей информация попадает только в промежуточную таблицу изменения агрегатов. А уже из этой таблицы в сами агрегаты данные попадают только в момент выполнения операции обновления или перестроения агрегатов.

Причин тому масса. Одна из них заключается в повышении параллельности работы в системе.

Дело в том, что, как правило, процесс формирования движений по регистрам (записи наборов записей) – это процесс конкурентный. Много пользователей параллельно пытается проводить документы. А вот обновление или перестроение агрегатов обычно вызывается регламентно и выполняется одним-единственным процессом на сервере. Поэтому промежуточную таблицу изменений агрегатов система терпеливо позволяет изменять сколь угодно часто массе сеансов обращения к регистру. А вот в момент обновления агрегатов можно будет "спокойно" снять получившиеся данные из таблицы изменений, ни с кем "не толкаясь локтями", обобщить, и также, "с достоинством, никому не мешая", внести требуемые изменения в таблицы агрегатов. То, что при этом будут сыпаться уже новые записи в таблицу изменений агрегатов , опять же совершенно

не беспокоит. Дойдет дело и до них, но уже в момент следующего обновления агрегатов.

Вызвать обновление агрегатов можно программно:

```
РегистрыНакопления.Продажи. ОбновитьАгрегаты()
```

Данное выражение, скорее всего, будет применяться в отдельном регламентном задании. Периодичность такого регламентного задания можно ставить достаточно частую: хоть каждую секунду. Параллельности работы пользователей с системой это мешать не будет, а наличие "свежеобновленных" агрегатов положительно скажется на скорости построения виртуальных таблиц в запросах по регистру. С другой стороны, если на Вашем предприятии за день выписывается всего с десяток документов (просто они очень большие), то незачем ситуацию доводить до абсурда с посекундным обновлением агрегатов.

Интерактивно обновить агрегаты тоже можно. Но, наверное, не стоит всерьез обсуждать возможность того, что это удобно. Все-таки регламентное задание – более оптимальный для этого инструмент.

4.4.2.4 Получение и использование списка оптимальных агрегатов

Поскольку в специальных системных таблицах регистров накопления система хранит статистику обращения к данным самих регистров, то эта информация может быть использована не только в рамках перестроения сети агрегатов. Система может еще посоветовать разработчику, каких именно агрегатов не хватает для еще большей оперативности работы с регистром.

В том же самом окне управления итогами, на закладке "Агрегаты" нажимаем кнопку "Оптимальные…"

Открывается окно, в котором система спрашивает нас параметры расчета. Пока просто нажимаем там "ОК".

Далее система попросит указать каталог, куда надо выложить xml-файл, содержащий результаты расчета. После выбора каталога именно в нем и ищите этот файл. Название файла будет совпадать с названием регистра и иметь расширение "xml"(т.е. в нашем случае "Продажи.xml").

Теперь попытаемся применить результаты расчетов.

Для этого возвращаемся в конфигуратор.

Опять открываем окно конструктора агрегатов нашего регистра.

Нажимаем кнопку "Открыть оптимальные агрегаты".

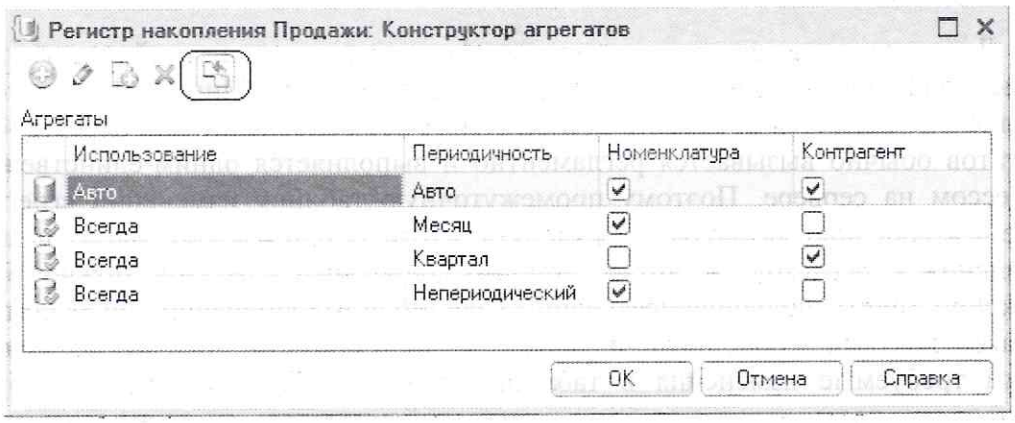


Рис. 4.40. Кнопка "Открыть оптимальные агрегаты"

Появляется окно выбора файла.

Выбираем наш "Продажи.xml".

В результате выбора система изменит окно конструктора агрегатов.

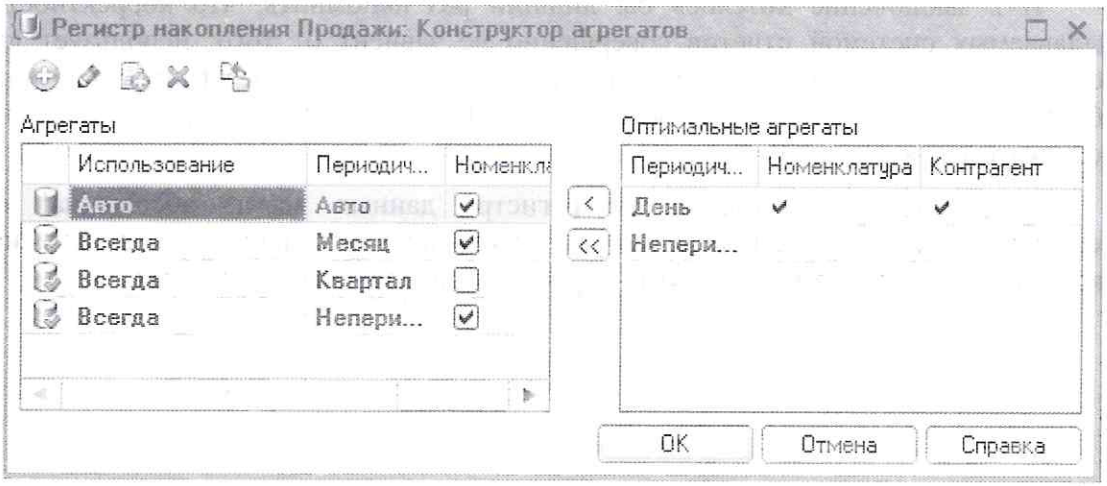


Рис. 4.41. Видоизмененный конструктор агрегатов

То, что выделено жирным шрифтом в левой части экрана, – это те существующие агрегаты, которые, по мнению системы, стоило бы вообще удалить. В правой части экрана жирным шрифтом выделены те агрегаты, которых, по мнению системы, не хватает.

Насчет удаления принимаем решение, что система "погорячилась". Во-первых, система не может знать о требованиях выдвинутых заказчиком. Во-вторых, если какой-то из агрегатов система в процессе эксплуатации посчитает излишним и у него использование стоит в положении "Авто",- ничто не помешает самой же системе при следующем перестроении сети агрегатов "гордо проигнорировать" такой агрегат.

А вот за совет по добавлению скажем:" Спасибо".

Нажимаем в центре окна кнопку добавления всех насоветованных агрегатов "<<".

Видим, что агрегаты добавились.

Теперь нажимаем кнопку "ОК" внизу окна конструктора агрегатов.

Далее можно сохранять изменения в конфигурации, запускать пользовательский режим, а там выполнить перестроение сети агрегатов.

Полезный эффект от нашей оптимизации в моей учебной базе поднялся до 20%.

По сравнению с предыдущими девятнадцатью не так много, но не забывайте, что мы сейчас работаем с очень маленькой учебной базой. А в условиях реально эксплуатируемой системы все было бы гораздо эффективнее.

Периодичность выполнения такой операции, как получение и применение списка оптимальных агрегатов, зависит прежде всего от реальной в этом необходимости, ну и от возможности разработчика выделить на это время.

Если ситуация с информационными потребностями меняется очень часто , наверное, можно выполнять такую работу раз в неделю или в месяц. Если все

более-менее монотонно, то раз в квартал или даже в год , наверное, самое подходящее.

Итак, мы с Вами освоили практически все вопросы эксплуатации агрегатов.

И в заключение хотелось бы лишний раз напомнить, что корректность выдаваемых системой отчетов совершенно не зависит от того, использует ли регистр механизм итогов или агрегатов, часто ли перестраиваются или обновляются агрегаты, разделяются ли итоги, есть или нет измерения, не используемые в итогах.

Корректность получаемых из регистра данных всегда абсолютна! А перечисленные выше вопросы могут только повлиять или не повлиять на скорость получения этих абсолютно корректных аналитических данных.

5. Организация планирования процесса оказания постпродажных услуг. Работа с регистром сведений

5.1. Постановка задачи. Создание необходимых объектов

Кроме собственно торговли наше автоматизируемое предприятие занимается оказанием услуг, сопровождающих продажу товаров. К ним можно относить услуги по доставке, упаковке, установке и прочим видам послепродажного сервиса.

Взаиморасчеты по этим услугам оформляются теми же документами, что и продажа товаров. А этот вопрос мы фактически уже автоматизировали.

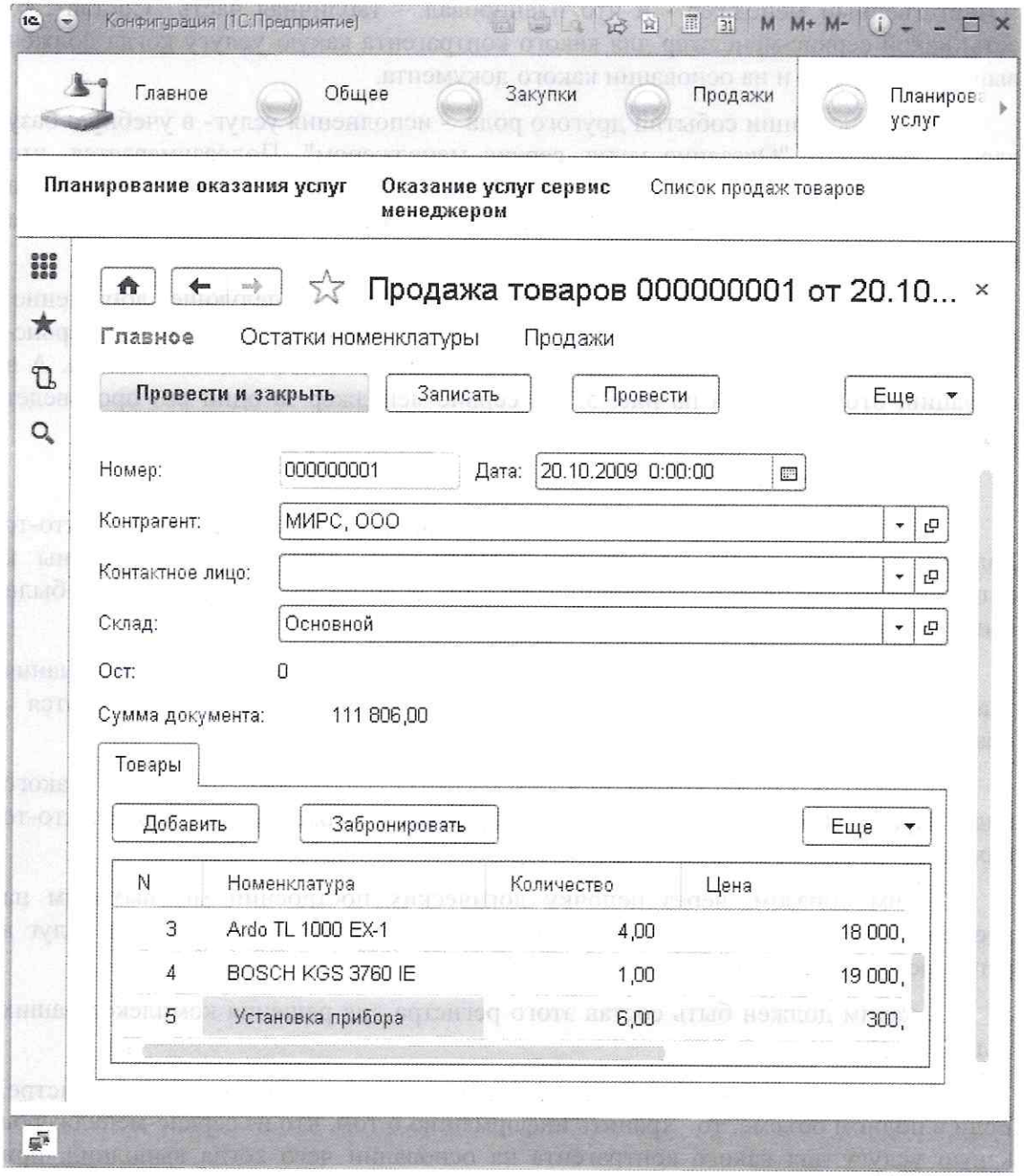


Рис. 5.1. Услуги в документе "Продажа товаров"

Но есть некий отдел сервисного обслуживания покупателей. А значит, придется решать вопросы планирования деятельности этого отдела. Ведь, продав стиральную машину с доставкой и установкой , надо определиться, кто и когда ее доставит, а кто установит. Далее надо будет проконтролировать выполнение этих услуг, и, наверняка, понадобится анализ , что и как часто выполнялось, в каких объемах, какими сотрудниками и т.д.

Соответственно, нам понадобится целая подсистема. И в учебной базе она уже реализована под названием "Планирование оказания услуг".

Сам факт планирования сроков и исполнителей оказания услуг является событием. Поэтому для фиксации таких событий в учебную базу уже введен документ "Планирование оказания услуг".

Документ позволяет зарегистрировать, кто планировал – реквизит "Ответственный менеджер"- и что планировал – табличная часть "Состав", то есть: какой сервис-менеджер для какого контрагента какую услугу когда должен выполнить да еще и на основании какого документа.

Для регистрации событий другого рода - исполнения услуг- в учебную базу введен документ "Оказание услуг сервис менеджером". Подразумевается, что данный документ будет заполняться ровно после оказания услуг и фиксировать на дату документа, что для какого контрагента было выполнено и на каком основании.

Да, для упрощения нашей учебной системы принято следующее допущение: услуги оказываются одномоментно. То есть не может быть ситуации, что сервис-менеджер выполнил половину услуги сегодня, а остальное доделает завтра. А в ситуации, отображенной на рис. 5.1 - сервис-менеджер за один раз произведет установку шести приборов.

Итак, сами документы уже есть.

Но прежде всего нужно помочь их правильно заполнять. То есть кто-то должен помнить, какие услуги уже проданы, но еще не запланированы к исполнению;, какие уже запланированы, но еще не выполнены, когда были выполнены те или иные услуги.

Вспоминаем начало курса : самым надежным средством запоминания значений показателей для будущего оперативного использования являются , правильно,- регистры.

Показатель что кто-то должен для кого-то сделать то-то тогда-то – какого вида? Правильно, накапливать тут нечего, то есть должен и все. Вернее, кто-то находится в состоянии необходимости делать то-то тогда-то и т.д.

Таким образом, через цепочку логических построений мы выходим на необходимость хранения информации по планированию и выполнению услуг в регистре сведений. Именно он оперирует показателями состояния.

А каким должен быть состав этого регистра для решения комплекса наших задач?

Попытаемся набросать эскиз того, что мы хотим хранить в этом регистре. Если в полном объеме, то хранить информацию о том, кто из сервис-менеджеров какую услугу для какого контрагента на основании чего когда выполнил, при этом надо так же знать на когда все это было запланировано, а то вдруг у нас постоянные просрочки – неудобно ведь будет перед покупателями.

Да, и самый главный вопрос при структурировании регистра сведений: хотим мы сделать регистр периодическим или нет? Если регистр будет непериодическим, то в нем удастся хранить только актуальные состояния. Если будет периодическим ,можно будет хранить всю историю развития состояний.

Согласно вышеизложенным требованиям, видим, что в нашем случае лучше все-таки хранить всю историю. И тогда на любой конкретный момент времени из регистра можно будет получать срез. Скорее всего, "срез последних", то есть набор записей, характеризующий ситуацию, все еще актуальную на момент среза.

Чисто технически "срез последних" строится за счет отбора записей, наиболее близких (из прошлого) к моменту среза, уникальных по набору значений измерений, то есть если при построении среза для одного и того же набора значений измерений есть несколько записей в регистре , то будет выбрана наиболее поздняя из них, но с моментом времени, не превышающим момент времени среза.

Тогда эскиз получает примерно таким, как в табл.5.1.

Таблица 5.1. Таблица планирования-оказания услуг

Период	Контраг-т	Услуга	Документ основание	Сервис-Менедж.	Планир. дата выполн.	Реальная дата выполн.
19.10.2009	МИРС, ООО	Доставка	Продажа №1 от 20.10.09			
19.10.2009	МИРС, ООО	Подключение электр.	Продажа №1 от 20.10.09			
20.10.2009	МИРС, ООО	Доставка	Продажа №1 от 20.10.09	Ступин	21.10.2009	
20.10.2009	МИРС, ООО	Подключение электр.	Продажа №1 от 20.10.09	Зайцев	23.10.2009	
21.10.2009	МИРС, ООО	Доставка	Продажа №1 от 20.10.09	Ступин	21.10.2009	21.10.2009

В данном случае мы видим, что в этой таблице можно отследить всю историю:

19.10.2009 было что-то продано ООО "Мирс" с доставкой и подключением.

20.10.2009 было запланировано, кто и когда выполнит эти услуги.

Ну а 21.10.2009 сервис-менеджер Ступин с задачей справился - доставку уже произвел.

После составления эскиза переходим к структурированию регистра сведений. Центральным вопросом является что из указанных полей делать измерениями, а что делать ресурсами.

Для того чтобы правильно решить этот вопрос, важно понимать, что в любом регистре могут существовать записи, только абсолютно уникальные по набору значений ключевых полей. К ключевым полям регистра сведений относятся:

- измерения;
- поле "Период" ,если регистр периодический;
- поле "Регистратор", если периодичность регистра "по позиции регистратора".

Кроме того, с точки зрения получения функциональности периодического регистра сведений, то есть получения среза первых или среза последних, **конкретный набор значений измерений может существовать только в единственном экземпляре**, то есть он уникален!

Таким образом, все приводит к следующему выводу: если в момент среза для некой сущности потребуется помнить много значений сразу ,– эту сущность надо оформлять в виде измерений. Если некая сущность в сопоставлении с конкретным набором значений измерений может существовать только в единственном экземпляре – это ресурс.

Проверяем наш вывод на нашей задаче. Для этого, последовательно перебирая колонки таблицы, пытаемся определить, в момент среза нужно помнить много значений или достаточно только одно. И если много – то это измерение. Если одно-единственное значение – значит это ресурс, характеризующий ту конкретную комбинацию значений измерений, что сложилась к тому моменту.

Верно ли то, что нам понадобится запоминать информацию только для одного контрагента? Нет, конечно. Значит , чтобы можно было запоминать информацию для многих контрагентов , это поле должно быть ключевым, то есть контрагент должен быть измерением.

Дальше рассуждаем: в момент среза верно ли, что для конкретного контрагента можно планировать выполнение только одной услуги, то есть нельзя запланировать, например, и доставку, и установку сразу? Нет, конечно. Значит, поле "Услуга" тоже должно быть измерением.

Верно ли, что для конкретного контрагента конкретную услугу можем планировать только в привязке только к одному документу продажи? То есть будем прямо отказывать покупателю в подводке воды к посудомоечной машине, мотивируя тем, что в прошлом году он у нас уже что-то покупал и мы уже один раз подводили воду (например, к стиральной машине)? Конечно – нет. Значит, поле "Документ основание" тоже должен быть измерением.

Верно ли, что для конкретного контрагента конкретную услугу на основании конкретного документа продажи может выполнять только один сервис-инженер? Нет, конечно, ведь есть даже понятие "бригада грузчиков". Значит, и поле "Сервис-инженер" должно стать измерением.

Верно ли, что для конкретного контрагента конкретную услугу на основании конкретного документа продажи конкретный сервис-инженер должен будет выполнить в конкретный день? А вот тут – да. Помните, мы договаривались, что выполнение услуги одномоментно как с точки зрения планирования, так и с точки зрения, собственно, выполнения. Значит, поля

"Планируемая дата выполнения" и "Реальная дата выполнения" в нашей задаче необходимо реализовать в качестве ресурсов. Вот что является показателями того или иного состояния в нашей задаче!

Проверяем по таблице 5.1. Если сделать "срез последних" на вечер 19.10.2009 , то от нашей таблицы должны остаться только две верхние записи. И действительно, в этот момент уже есть потребность в выполнении услуг, но еще не распланировано , кто и когда будет это делать.

А вот если сделать срез последних на 22.10.2009 , то получим такие записи, актуальные на этот момент и уникальные с точки зрения набора значений измерений (см. табл. 5.2).

Таблица 5.2. Таблица среза последних на 22.10.2009

Период	Контраг-т	Услуга	Документ основание	Сервис-Менедж.	Планир. дата выполн.	Реальная дата выполн.
20.10.2009	МИРС, ООО	Подключение электр.	Продажа №1 от 20.10.09	Зайцев	23.10.2009	
21.10.2009	МИРС, ООО	Доставка	Продажа №1 от 20.10.09	Ступин	21.10.2009	21.10.2009

И действительно, с прикладной точки зрения на 22.10.2009 можно констатировать, что доставка уже выполнена, а вот подключение электричества – еще нет, эта услуга еще только запланирована к исполнению 23.10.2009.

Итак, добавляем в состав конфигурации новый регистр сведений "ПланированиеОказаниеУслуг".

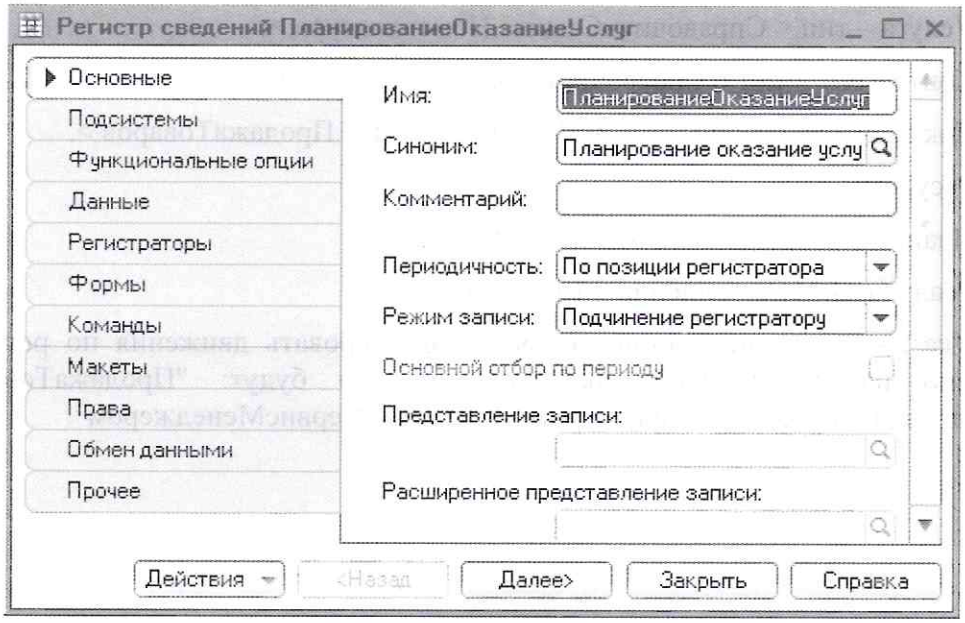


Рис. 5.5. Регистр "ПланированиеОказаниеУслуг"

Поскольку подразумевается активное использование документов в нашей подсистеме, логично сделать режим записи "Подчинение регистратору", а периодичность "По позиции регистратора".

Наш регистр лучше всего также отнести к подсистеме "ПланированиеОказаниеУслуг" на соответствующей закладке.

Определяем состав регистра согласно тому, что обсуждали выше:

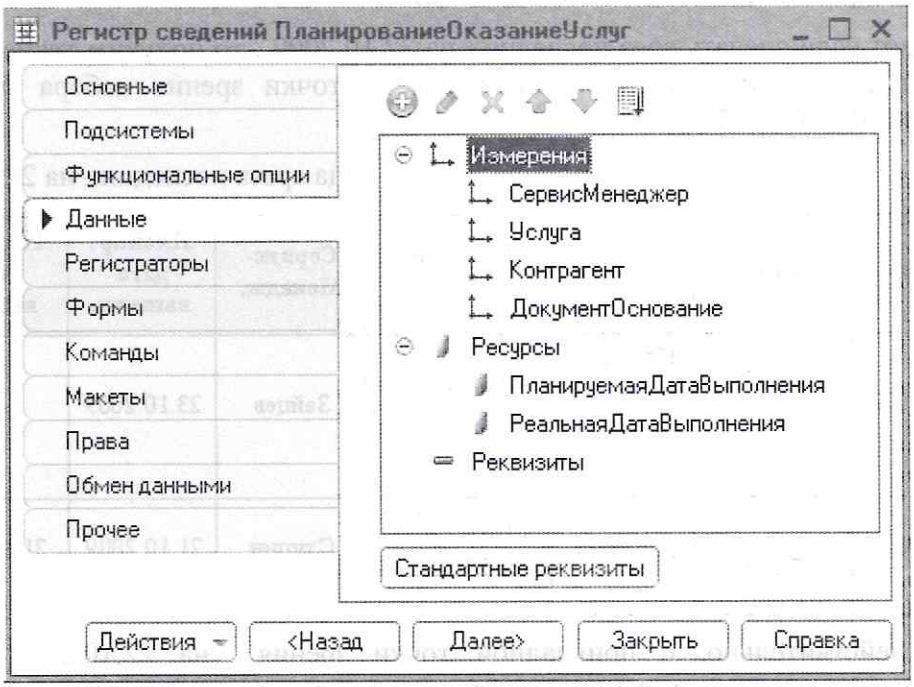


Рис.5.6. Данные регистра "ПланированиеОказаниеУслуг"

Измерения:

СервисМенеджер – тип <СправочникСсылка.ФизическиеЛица >;

Услуга – тип < СправочникСсылка.Номенклатура >;

Контрагент – тип < СправочникСсылка.Контрагенты >;

ДокументОснование – тип < ДокументСсылка.ПродажаТоваров >.

Ресурсы:

ПланируемаяДатаВыполнения – тип <Дата>;

РеальнаяДатаВыполнения – тип <Дата>;

Указываем, какие документы могут формировать движения по регистру. Согласно начальной постановке задачи это будут: "ПродажаТоваров"; "ПланированиеОказаниеУслуг" и "ОказаниеУслугСервисМенеджером".

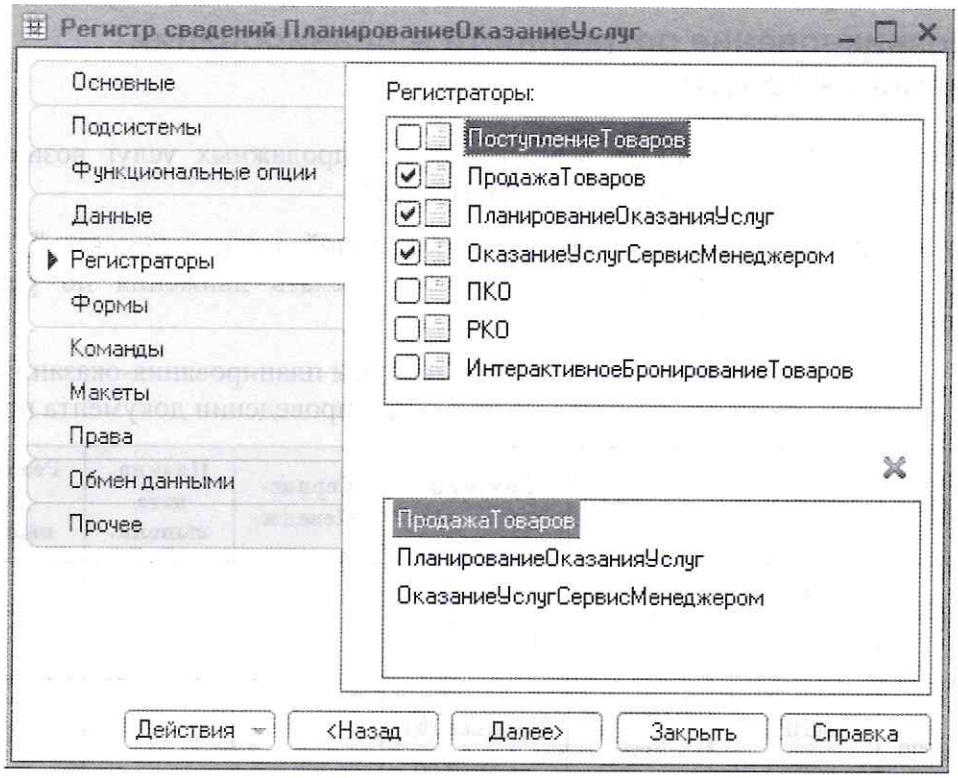


Рис. 5.7. Указание регистраторов

Теперь необходимо продумать этапы работы с этим регистром.

Согласно постановке задачи нам необходимо будет реализовать ситуации:

1) Возникновение потребности в планировании выполнения услуги.
2) Планирование выполнения услуги: заполнение и проведение документа.
3) Оказание услуги: заполнение и проведение документа.
4) Формирование отчетности.

Итак, приступим к поэтапной отработке.

5.2. Возникновение потребности в планировании выполнения услуги

Потребность планировать выполнения постпродажных услуг возникает в момент проведения документа "ПродажаТоваров".

С технической точки зрения, если в табличной части документа "Товары" указана услуга, то этого достаточно чтобы сделать движения по регистру "ПланированиеОказаниеУслуг" (см. табл. 5.3).

Таблица 5.3. Заполнение таблицы планирования-оказания услуг при проведении документа продажи

Период	Контраг-т	Услуга	Документ основание	Сервис-Менедж.	Планир. дата выполн.	Реальная дата выполн.
19.10.2009	МИРС, ООО	Доставка	Продажа №1 от 20.10.09			
19.10.2009	МИРС, ООО	Подключение электр.	Продажа №1 от 20.10.09			

А значит, нам пригодится технология, отработанная в разделе "3.1. "Обусловленное проведение документа".

Временно переименовываем процедуру "ОбработкаПроведения" документа "ПродажаТоваров", собираем конструктором движения по регистру "ПланированиеОказаниеУслуг".

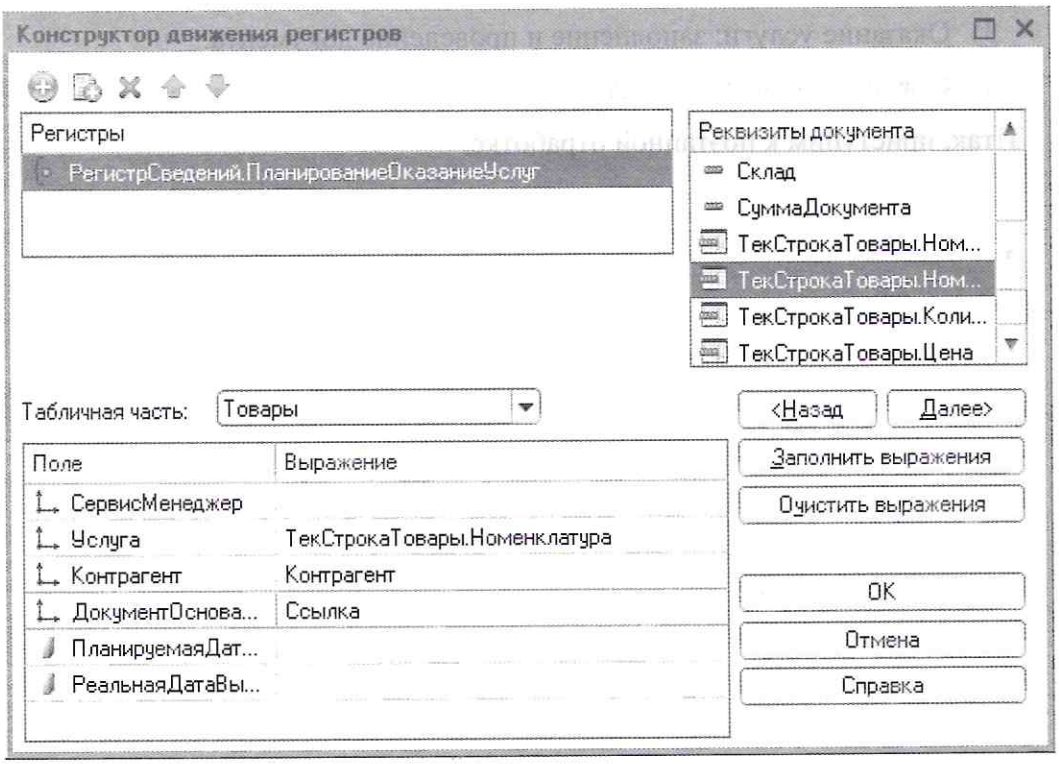


Рис. 5.8. Конструктор движений

В полученной процедуре дописываем комментарий - маркер недостающего действия:

```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
	Движения.ПланированиеОказаниеУслуг.Записывать = Истина;

	//!!! ДВИЖЕНИЯ НАДО ДЕЛАТЬ ТОЛЬКО ДЛЯ УСЛУГ!!!

	Для Каждого ТекСтрокаТовары Из Товары Цикл
		// регистр ПланированиеОказаниеУслуг
		Движение = Движения.ПланированиеОказаниеУслуг.Добавить();
		Движение.Период = Дата;
		Движение.Услуга = ТекСтрокаТовары.Номенклатура;
		Движение.Контрагент = Контрагент;
		Движение.ДокументОснование = Ссылка;
	КонецЦикла;
	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Для реализации этого недостающего действия надо будет не просто взять номенклатурные позиции из табличной части, а определить , действительно ли это услуги. А вот признак "ВидНоменклатуры" находится в одноименном реквизите справочника. А значит, чтобы собрать эту информацию за одно обращение к базе данных , понадобится запрос.

Схема запроса в данном случае простейшая. Думаю, Вы его уже сразу же себе представляете, а значит, можете тут же реализовать сам запрос в консоли запросов.

В конечном итоге получиться нечто такое:

```
ВЫБРАТЬ
	ПродажаТоваровТовары.Номенклатура
ИЗ
	Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
ГДЕ
	ПродажаТоваровТовары.Ссылка = &Ссылка
	И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры =
				ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)

СГРУППИРОВАТЬ ПО
	ПродажаТоваровТовары.Номенклатура
```

Окончательный вид механизма формирования движений документа "ПродажаТоваров" по регистру "ПланированиеОказаниеУслуг" может быть таким:

```
Процедура СформироватьДвиженияПоРегиструПланированияОказанияУслуг()
    Движения.ПланированиеОказаниеУслуг.Записывать = Истина;

    //!!! ДВИЖЕНИЯ НАДО ДЕЛАТЬ ТОЛЬКО ДЛЯ УСЛУГ!!!
    Запрос = Новый Запрос;
    Запрос.Текст =
        "ВЫБРАТЬ
        |    ПродажаТоваровТовары.Номенклатура
        |ИЗ
        |    Документ.ПродажаТоваров.Товары КАК ПродажаТоваровТовары
        |ГДЕ
        |    ПродажаТоваровТовары.Ссылка = &Ссылка
        |    И ПродажаТоваровТовары.Номенклатура.ВидНоменклатуры =
        |                ЗНАЧЕНИЕ(Перечисление.ВидыТоваров.Услуга)
        |
        |СГРУППИРОВАТЬ ПО
        |    ПродажаТоваровТовары.Номенклатура";

    Запрос.УстановитьПараметр("Ссылка", Ссылка);
    Результат = Запрос.Выполнить();
    ВыборкаДетальныеЗаписи = Результат.Выбрать();

    Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
        Движение = Движения.ПланированиеОказаниеУслуг.Добавить();
        Движение.Период = Дата;
        Движение.Услуга = ВыборкаДетальныеЗаписи.Номенклатура;
        Движение.Контрагент = Контрагент;
        Движение.ДокументОснование = Ссылка;
    КонецЦикла;

КонецПроцедуры
```

Не забудьте вернуть процедуре "ОбработкаПроведения" документа "ПродажаТоваров" первоначальное имя и добавить в нее вызов вышеописанной процедуры "СформироватьДвиженияПоРегиструПланированияОказанияУслуг".

Очевидно, что делать это надо до момента записи всех движений в процедуре:

```
Процедура ОбработкаПроведения(Отказ, Режим)
    ...
    СформироватьДвиженияПоРегиструПланированияОказанияУслуг();

    //Общая запись всех движений
    Движения.Записать();

    //Контроль  отрицательных остатков
    ...
КонецПроцедуры
```

Далее отлаживаем механизм на перепроведении документов продажи учебной базы.

Для удобства тестирования не забудьте "попросить" систему отобразить в форме документа гиперссылку на окно движений документа по нашему новому регистру.

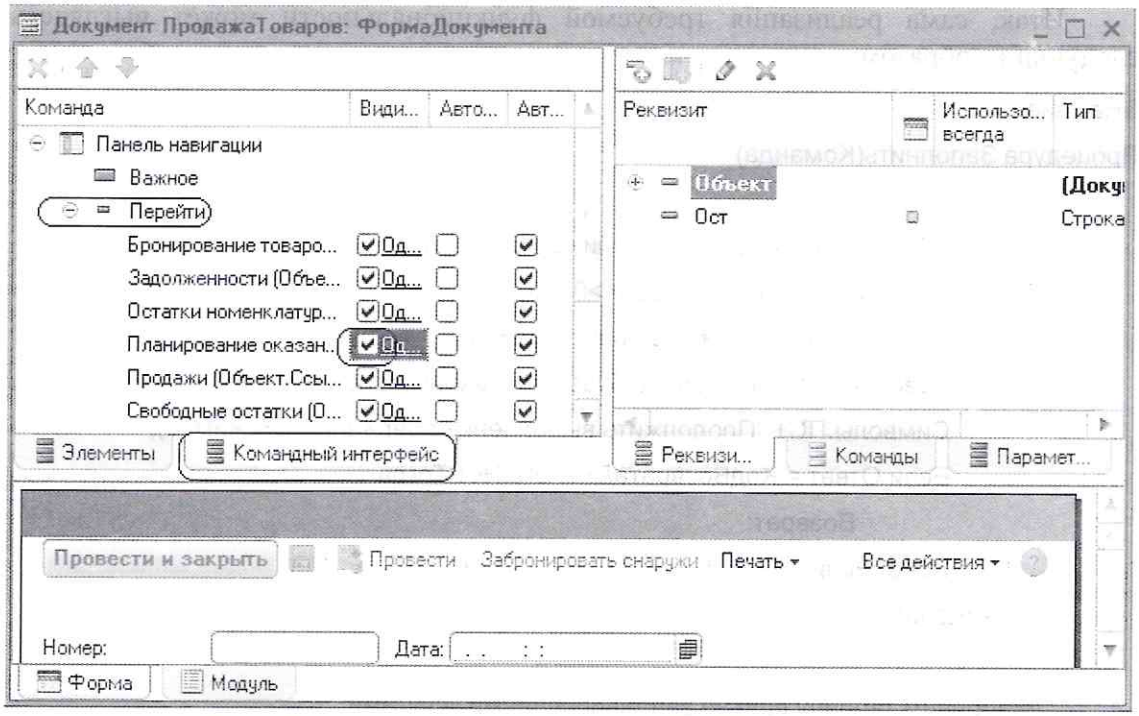


Рис. 5.9. Обеспечение навигации для перехода к движениям документа

В учебной базе протестировать можно на документах "ПродажаТоваров" №№ 000000001, 000000003, 000000005 и 000000006.

5.3. Планирование выполнения услуги: заполнение и проведение документа

Для помощи пользователю в автоматическом заполнении размещаем в форме документа "ПланированиеОказанияУслуг" новую команду формы "Заполнить" и размещаем ее вызов в командной панели табличного поля "Состав".

В качестве обработки события нажатия пользователем кнопки "Заполнить" необходимо:

- Проверить, не стоит ли очистить табличную часть документа от прошлого заполнения.
- Заполнить таблицу документа услугами, еще ни на какую конкретную дату не запланированными, но уже попавшими в регистр сведений "ПланированиеОказаниеУслуг".

Если первое можно делать непосредственно на клиенте, то второе можно делать только на сервере (поскольку обращение к регистру – это будет обращение к базе данных). Причем получать данные из регистра нужно будет срезом последних.

Итак, сама реализация требуемой функциональности может выглядеть следующим образом:

```
&НаКлиенте
Процедура Заполнить(Команда)

        // Проверим, не заполнена ли она была ранее
    Если Объект.Состав.Количество()<>0 Тогда
        Режим = РежимДиалогаВопрос.ДаНет;
        Ответ = Вопрос("Таблица будет предварительно очищена." +
        Символы.ПС+ "Продолжить выполнение операции?", Режим, 0);
        Если Ответ = КодВозвратаДиалога.Нет Тогда
            Возврат;
        КонецЕсли;
    КонецЕсли;

    //Заполним таблицу еще не запланированнми услугами
    ЗаполнитьТаблицуЕщеНезапланированнымиУслугами();

КонецПроцедуры
&НаСервере
Процедура ЗаполнитьТаблицуЕщеНезапланированнымиУслугами()
Объект.Состав.Очистить();

Запрос = Новый Запрос;
```

```
Запрос.Текст =
        "ВЫБРАТЬ
        |       ПланированиеОказаниеУслугСрезПоследних.Услуга,
        |       ПланированиеОказаниеУслугСрезПоследних.Контрагент,
        |       ПланированиеОказаниеУслугСрезПоследних.ДокументОснование,
        |
        МАКСИМУМ(ПланированиеОказаниеУслугСрезПоследних.СервисМенеджер)
        |                                                   КАК СервисМенеджер
        |ИЗ
        |       РегистрСведений.ПланированиеОказаниеУслуг.СрезПоследних
        |                               КАК ПланированиеОказаниеУслугСрезПоследних
        |
        |СГРУППИРОВАТЬ ПО
        |       ПланированиеОказаниеУслугСрезПоследних.Услуга,
        |       ПланированиеОказаниеУслугСрезПоследних.Контрагент,
        |       ПланированиеОказаниеУслугСрезПоследних.ДокументОснование
        |
        |ИМЕЮЩИЕ
        |
        МАКСИМУМ(ПланированиеОказаниеУслугСрезПоследних.СервисМенеджер)
        |                       = ЗНАЧЕНИЕ(Справочник.ФизическиеЛица.ПустаяСсылка)";
Запрос.УстановитьПараметр("ПустаяДата", Дата(1,1,1));
Результат = Запрос.Выполнить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
        НовСтрокаСостава = Объект.Состав.Добавить();
        ЗаполнитьЗначенияСвойств(НовСтрокаСостава,ВыборкаДетальныеЗаписи);
КонецЦикла;
КонецПроцедуры
```

Текст процедур в отдельных комментариях не нуждается, за одним исключением. По смыслу задачи нет нужды заполнять документ теми услугами, которые уже были запланированы на кого-то. Это условие отрабатывается в запросе.

Примерно похожего результат можно было бы добиться проверкой полей "ПланируемаяДатаВыполнения" и "РеальнаяДатаВыполнения" на незаполненность.

Только незаполненная дата – это так называемая "дата по умолчанию", это первая секунда нашей эры. Получить эту первую секунду можно при помощи литерала '00010101000000' (в апострофах) или функции Дата(1,1,1).

Теперь можно сохранять изменения и тестировать в пользовательском режиме.

Практикум №15

Сформируйте процедуру проведения документа "ПланированиеОказанияУслуг" по регистру "ПланированиеОказаниеУслуг". При этом надо учесть, что недопустимы ситуации, когда непонятно, кто занимался этим планированием (то есть не заполнен реквизит "ОтветственныйМенеджер") и выполнение услуги вдруг оказалось запланированным на никого (пустое поле "СервисМенеджер" в строке планирования).

5.4. Оказание услуги: заполнение и проведение документа

Аналогично предыдущим наработкам реализуем заполнение и проведение документа "ОказаниеУслугСервисМенеджером".

Практикум №16

Реализуйте, пожалуйста, механизм заполнения документа "ОказаниеУслугСервисМенеджером" за счет добавления новых кнопок на форму документа:
а) "Заполнить сегодняшними" - при нажатии на кнопку надо заполнять табличную часть услугами, запланированными на сервис-менеджера, указанного в шапке документа, теми заданиями, которые запланированы на сегодняшний день и еще не выполнены;
б) "Заполнить невыполненными" - при нажатии на кнопку надо заполнять табличную часть услугами, запланированными на сервис-менеджера, указанного в шапке документа, теми заданиями, которые еще не выполнены.

Сформируйте процедуру проведения документа "ОказаниеУслугСервисМенеджером" по регистру "ПланированиеОказаниеУслуг". При этом надо учесть, что недопустимы ситуации, когда непонятно, кто именно эту услугу выполнил (пустое поле "СервисМенеджер" в документе).

5.5. Отчетность планирования и выполнения услуг

Практикум №17

Создайте отчет "Отчет по планированию и выполнению услуг". У отчета должно быть три варианта формирования:
- еще не распределенные услуги;
- запланированные услуги;
- выполненные услуги.

Данные должны выводиться согласно смыслу названия каждого из вариантов.

6. Большая самостоятельная работа "Резервирование товаров и развитие системы планирования выполнения услуг"

Практикум №18

Необходимо ввести в систему возможность резервировать товары на складах фирмы под конкретные счета, выписанные для покупателей. Смысл работы механизма резервирования сводится к следующему:
При проведении документа "Счет" товар попадает в резерв под этого клиента.

Далее на основании документа "Счет" выписывается документ "ПродажаТоваров". Пользователь может оставить все товарные позиции и их количество, унаследованные из счета, а может часть товара удалить из табличной части документа "ПродажаТоваров". Это значит, что клиент забирает зарезервированный товар по частям.

Далее документ "ПродажаТоваров" проводится, товар при этом должен списываться не только с остатков склада, но и с резерва.

Таким образом, когда на основании счета заводится документ "ПродажаТоваров", то в табличную часть должны переходить только те товары, которые на данный момент остаются в резерве.

Кроме того, необходимо предусмотреть ситуацию, когда по каким-то причинам покупатель не будет забирать остаток резерва товаров, сделанный каким-то счетом. Чтобы разблокировать такой товар, в систему необходимо ввести документ "СнятиеРезервaТоваров".

Ну и, естественно, при проведении счетов и документов "ПродажаТоваров" необходимо контролировать появление отрицательных свободных остатков!

Кроме всего прочего, необходимо создать отчеты, которые бы показывали, что из товаров на какую-то дату находится в резерве, под какие счета, как происходило движение резерва товаров в интервале дат (какими документами товар заводился в резерв, какими списывался).

Создать блок аналитических отчетов, показывающих активность сервис-менеджеров.

Этапы работ.

1. Создать документ "Счет". В шапке реквизиты "Контрагент", "Склад" и числовой "СрокРезервирования" (имеется в виду число дней). Табличная часть "Товары" полностью, как у документа "ПродажаТоваров". (Срок резервирования понадобится в будущем для принятия решения снимать просроченный резерв или нет. В табличной части для самого резервирования нужны только реквизиты "Номенклатура" и "Количество". Но ведь в реальной жизни "Счет" - это счет на оплату. Так что без денег не обойтись, еще понадобятся "Цена" и "Сумма"). Кроме того, не забывайте о сервисе для пользователя. При подборе товара из справочника должна подставляться в документ цена продажи; своевременно должна перерасчитываться общая сумма в строке.

2. Создать регистр "РезервыНоменклатуры". Поскольку для отчетности по резервам нужны разрезы "Номенклатура", "Склад" и "ПоСчету" – это и

будут измерения нового регистра (тип значений соответствующий). Ресурсом будет только "Резерв". (Про деньги нас никто не спрашивал, и, кроме того, не требуется резервировать конкретные партии. Главное, чтобы покупателю просто хватило товара, когда он придет забирать свой резерв).

3. Прописать проведение документа "Счет" по регистру "РезервыНоменклатуры". Не забудьте предварительно проверить, достаточно ли для проведения документа товара в свободном остатке. Напомним, свободный остаток – это то, что сейчас на складе, но при этом не забронировано и не зарезервировано. То есть проведение документа "Счет" тоже влияет на свободные остатки

4. Прописать ввод документа "ПродажаТоваров" на основании документа "Счет". Контрагента и склад необходимо брать из "шапки". А вот при наследовании реквизитов табличной части необходимо проверять, находится ли этот товар до сих пор в резерве (может, он уже отпущен другой расходной накладной). И количество, соответственно, брать из остатка данного резерва в регистре "РезервыНоменклатуры". Напоминаем, чтобы документ "ПродажаТоваров" помнил, на основании какого счета он выписан, необходимо добавить в него реквизит "шапки" типа <ДокументСсылка.Счет> и при вводе на основании вписывать в него документ-источник, т.е. сам "Счет".

5. Прописать проведение документа "ПродажаТоваров" по регистру "РезервыНоменклатуры". Необходимо списывать с набора измерений "Номенклатура", "Склад" и "ПоСчету" указанное в табличной части "Товары" количество товара. Кроме того, предусмотрите случаи:
- когда документ "ПродажаТоваров" вводится вообще без счета в основании;
- когда пользователь в документе "ПродажаТоваров" поставил количество большее, чем в счете.
Нельзя при этом списывать с резервов больше, чем резервировали под наш документ!

6. Необходимо создать документ "СнятиеСРезерва". В табличной части этого документа должен быть только один реквизит - "Счет". Т.е. одно "СнятиеСРезерва" сможет снять резервы с многих счетов. Кроме того, при позиционировании на конкретной строке документа можно показать в форме еще Дату Счета, Дату окончания резервирования, Клиента и Сумму счета.

7. Прописать проведение документа "СнятиеСРезерва". При этом необходимо выбирать остатки резервов на текущий документ по каждому из снимаемых счетов из регистра "РезервыНоменклатуры" и выполнять для них отрицательные движения. Тем самым мы будем улучшать свободные остатки!

8. Сделать на форме документа "СнятиеСРезерва" кнопку "Заполнить" и создать механизм автоматического заполнения табличной части просроченными счетами. Для этого необходимо из регистра "РезервыНоменклатуры" выбирать счета, для которых есть ненулевые

резервы и сравнивать: Дата документа "Счет" плюс срок резервирования меньше даты документа "СнятиеСРезерва" или нет.
9. Создать отчет "Зарезервировано" на произвольную дату.
10. Создать отчет "ДвижениеРезервов" в интервале дат, используя запрос по регистру "РезервыНоменклатуры". Отчет должен показывать информацию о начальном остатке, приходе, расходе и конечном остатке резервов. В отчете должны быть виды: "Кратко" - когда представляются обобщенные данные только по товарам и складам,- и "Подробно" - когда движения по регистру представляются в разрезах и товаров, и документов, эти движения выполнивших.

11. Создать оборотный регистр "Оказание услуг" и обеспечить его заполнение при проведении документа "ОказаниеУслугСервисМенеджером", на основании данных которого построить отчет "Оказание услуг", демонстрирующий, какие сервис-менеджеры, сколько и каких услуг оказали каким контрагентам.

12. Создать отчет "Исполнительность сервис-менеджеров", позволяющий анализировать, сколько и каких услуг какие сервис-менеджеры выполняли с просрочкой (дата реального исполнения больше планируемой даты выполнения), а сколько - в срок или даже раньше срока.

13. Создать отчет "Обоснованность оказания услуг", позволяющий проанализировать случаи исполнения услуг без указания документа-основания, а также с указанием неверного документа-основания (то есть в самом конкретном документе-основании "ПродажаТоваров" именно такая услуга и не продавалась).

Заключение

Итак, уважаемый Коллега, позади пройденный курс.

Если чувствуете, что в силах решить любую оперативную задачу посредством использования регистров , Вы не ошиблись, так и есть. А значит , не сдерживайте себя – беритесь сразу за работу.

А если "в голове теперь много всего", но вот уверенности не очень? Хм, признайтесь , ведь не все практикумы выполнили? И большую самостоятельную работу, наверное, до последнего пункта не доделали… Значит – ближайшие задачи ясны? Не стесняйтесь – чем раньше сделаете, тем увереннее себя будете чувствовать.

Если уверенности совсем нет , начните самостоятельно все с первой страницы еще раз. Люди не рождаются разработчиками в системе "1С:Предприятие 8". Все проходят через стадию "вот вроде все понимаю, но сам сделать ничего не могу". Главное захотеть и не сдаваться.

В случае полноценного прохождения всех курсов линейки:

Введение в конфигурирование в системе "1С:Предприятие 8". Основные объекты.

Конфигурирование в системе "1С:Предприятие 8". Решение оперативных задач.

Конфигурирование в системе "1С:Предприятие 8". Решение бухгалтерских задач.

Конфигурирование в системе "1С:Предприятие 8". Решение расчетных задач.

Вы будете вполне готовы приступить к непосредственной подготовке к экзаменам "1С:Профессионал по платформе "1С:Предприятие 8" и экзамену "1С:Специалист по платформе "1С:Предприятие 8".

Кроме того, навыки, полученные в процессе изучения этих курсов, будут не лишними при сдаче экзаменов "1С:Профессионал" и "1С:Специалист" по соответствующим типовым решениям.

В любом случае, даже если сдача экзаменов не является целью, ко всем пожелание только одно – побольше практики! Ведь как говорил классик?

"ПРАКТИКА - КРИТЕРИЙ ИСТИНЫ"

Успехов в Вашей, иногда нелегкой, но такой захватывающей работе!