Учебный курс

Подготовка к аттестации 1С:Специалист по платформе 1С:Предприятие 8.3

Общие, универсальные механизмы, которые используются для решения задач всех разделов

Главы 11 – 21

Версия 1.1.1

Проект Курсы-по-1С.рф

2019

Оглавление

11. Какие навыки в использовании управляемых блокировок потребуются на экзамене

В один момент времени можно выполнить только одну операцию записи данных в конкретный регистр. При одновременной попытке такой записи платформа позволит это сделать первому, кто пытается выполнить эту операцию, а второму придется ожидать своей очереди.

Момент, когда регистр занят и недоступен для чтения или записи другим пользователям, называется «блокировкой».

Когда база данных одновременно используется несколькими пользователями, в работе механизмов списания товаров могут возникать проблемы. Например, могут появиться отрицательные остатки товаров. Так бывает, когда пользователи практически одновременно пытаются списать один и тот же товар (например, оформить продажу). То есть система сообщает один и тот же результат при получении данных об остатке товаров, если информацию запрашивает более чем один пользователь (например, два).

Почему так происходит

Начнем с понятия транзакция. **Транзакция** – это последовательность изменений в базе данных, которая может быть завершена только полностью, иначе производится отмена всех изменений. Для обеспечения целостности данных все объекты БД записываются в транзакции.

Транзакция при проведении документа длится некоторое время. Остатки по номенклатуре получаются внутри транзакции, но изменяются только по ее завершении. Что происходит, когда пользователи проводят документы одновременно:

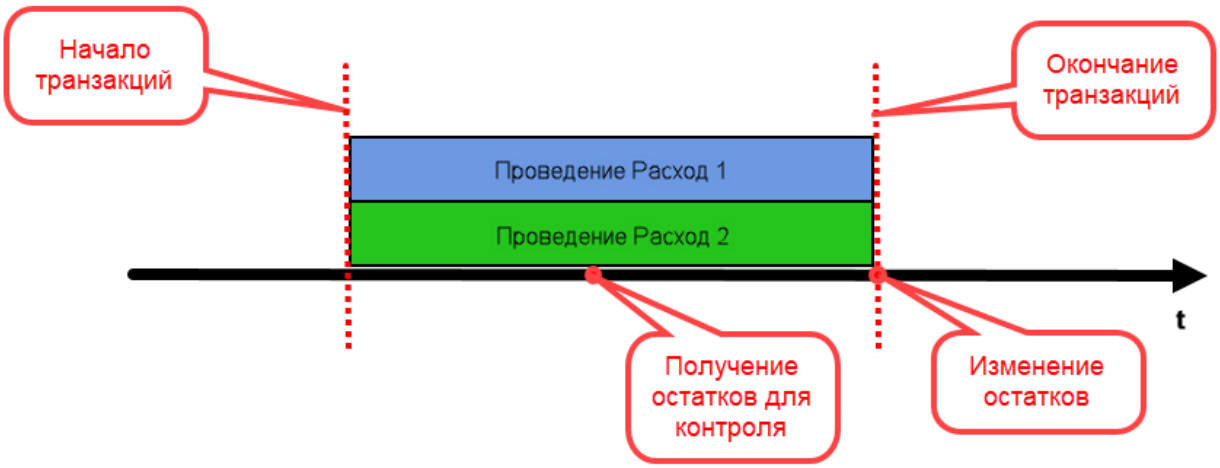


Рисунок 1 – Получение остатков при одновременном проведении документов

Как видно на схеме (рис. 1), при проведении двух документов одновременно получаются одинаковые остатки номенклатуры. Полученный остаток может быть больше, чем требуется каждому пользователю в отдельности, но меньше, чем им требуется вместе. При этом оба документа могут быть успешно проведены. В результате итоговый остаток получится отрицательным. Контроль остатков не срабатывает.

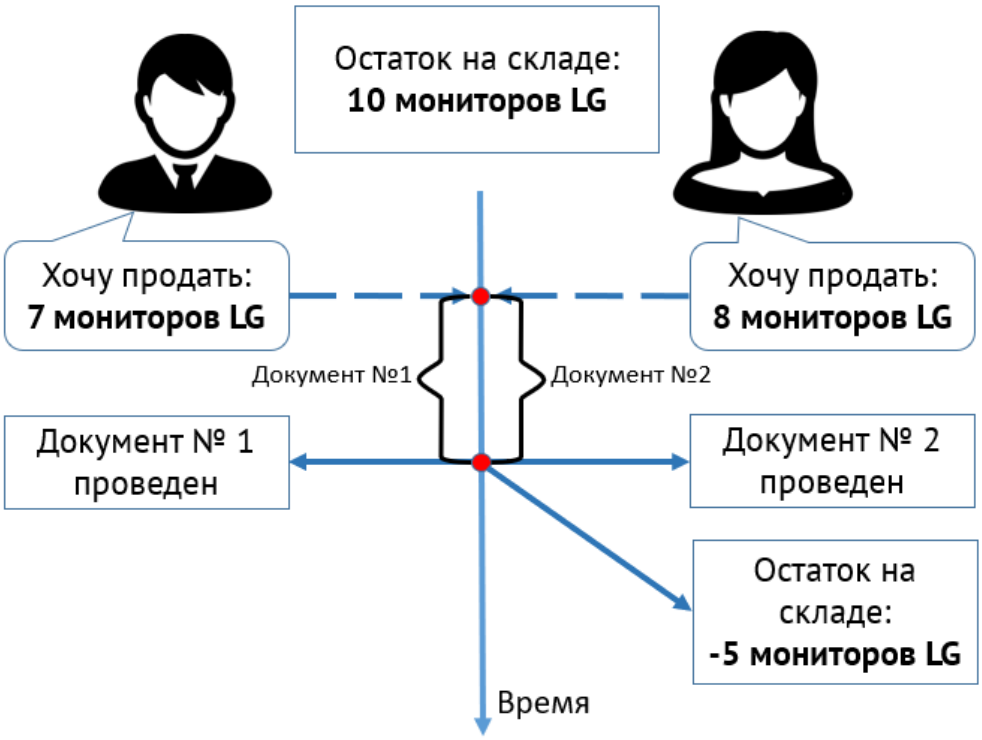


Рисунок 2 – Одновременное проведение документов пользователями

На рисунке два пользователя одновременно проводят продажу мониторов LG. При контроле остатков система получает остаток 10 шт., и оба документа проводятся. Как итог – минус 5 мониторов LG.

Если Документ № 2 будет проводиться после окончания проведения Документа № 1, то контроль остатков отработает корректно и не даст отгрузить товар в минус:

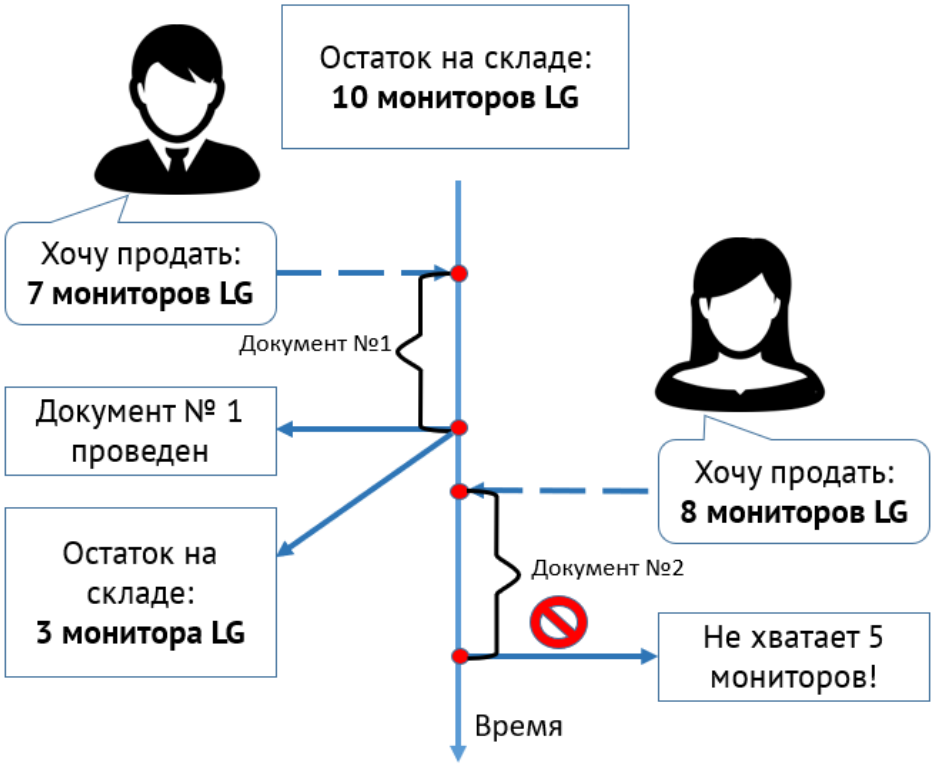


Рисунок 3 – Последовательное проведение документов пользователями

Не стоит недооценивать данную проблему даже в том случае, если в системе работают всего два пользователя. Они могут проводить документы практически одновременно, например, если один из них выполняет групповое проведение документов.

Как решить проблему

Чтобы решить проблему с параллельным проведением нужно в транзакции при проведении Документа № 1 заблокировать в регистре накопления остатки по товарам, которые присутствуют в документе. Тогда другие транзакции не смогут прочитать остатки по данным товарам. По окончании проведения Документа № 1 транзакция завершится, а остатки по товарам разблокируются и

измененные актуальные остатки будут доступны другим транзакциям. Рассмотрим данный пример на схеме:

Рисунок 4 – Получение остатков при установленной блокировке данных

Для установки блокировок на данные используются специальные объекты – блокировки. В платформе «1С:Предприятие» существуют объектные и транзакционные блокировки.

Объектные блокировки не позволяют двум пользователям одновременно интерактивно изменять один объект (например, пользователь открыл форму документа и работает с ней).

Транзакционные блокировки позволяют программно оперировать **актуальными** данными при выполнении движений по регистрам. Эти блокировки устанавливаются в транзакциях. Самый частый случай их использования – при проведении документов.

В данной главе мы рассмотрим только работу с транзакционными блокировками, так как объектные блокировки на экзамене не используются.

Когда нужно использовать блокировки

Задача установки блокировок становится актуальной, как только появляется вероятность одновременного проведения документов.

Для оптимизации параллельной работы пользователей блокировки требуется накладывать на строки со значениями измерений регистров, которые в рамках транзакций читаются и изменяются:

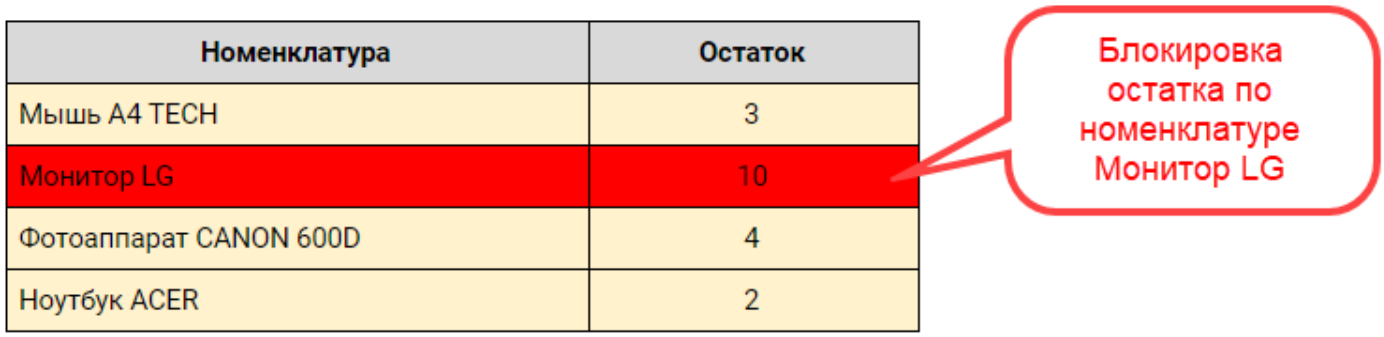


Рисунок 5 – Блокировка остатка по номенклатуре

Наглядным примером их использования является проведение документов с использованием «старой» методики контроля остатков, когда система должна заблокировать остаток товаров с момента получения данных об остатке до момента полного завершения транзакции (записи коллекции движений в регистр или отказа от записи).

С помощью блокировки система понимает, что только эта транзакция может работать с этим списком товаров. Если другая транзакция будет пытаться наложить блокировку хоть на один из ранее заблокированных товаров, она попадет в очередь ожидания. Через некоторое время (тайм-аут) будет сделана еще одна попытка наложить блокировку.

Блокировки не позволяют читать неактуальные остатки при параллельной работе пользователей. То есть транзакции будут выстраиваться в очередь для получения актуальных остатков. При этом транзакции по непересекающемуся набору товаров (на рисунке выделены зеленым фоном) будут выполняться параллельно:

Номенклатура	Остаток
Мышь A4 TECH	3
Монитор LG	10
Фотоаппарат CANON 600D	4
Ноутбук ACER	2

Рисунок 6 – Остатки по непересекающемуся набору товаров

Документ № 3, в котором продается Ноутбук ACER, можно одновременно провести с Документом № 1:

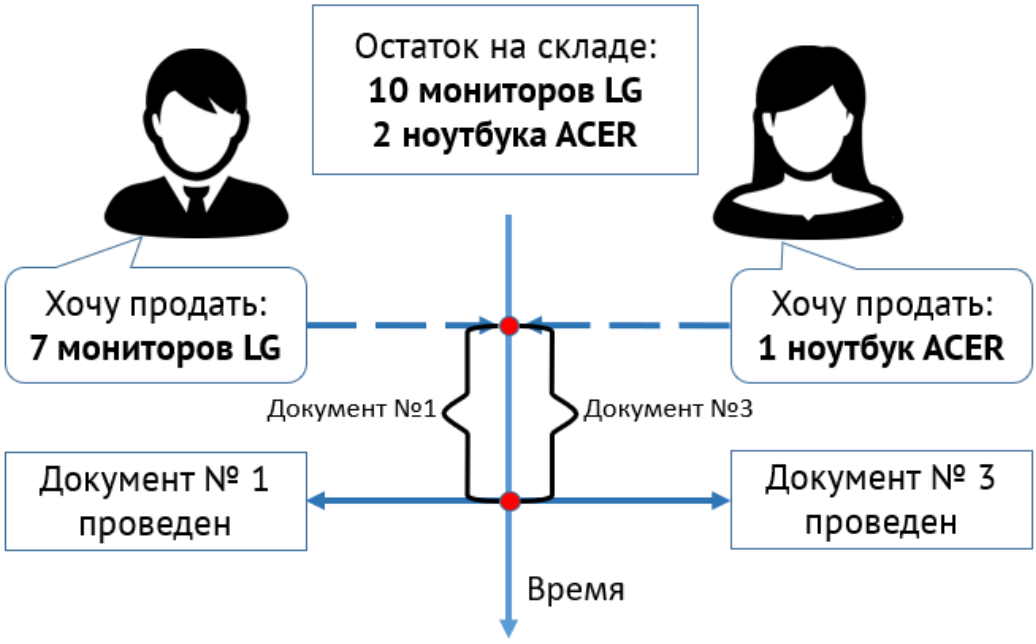


Рисунок 7 – Одновременное проведение документов с разной номенклатурой

Следует отметить, что не стоит устанавливать блокировки при проведении абсолютно всех документов, так как блокировки снижают возможности параллельной работы пользователей (масштабируемость системы).

Ярким примером, когда не нужно накладывать блокировку, является проведение документа «Поступление товаров», так как в этом случае нет никакой конкуренции за ресурсы (остатки). По сути, блокировка только уменьшит производительность системы.

Также следует отметить, что правильный вариант наложения блокировки внутри транзакции – как можно позже, но до чтения или изменения остатков:

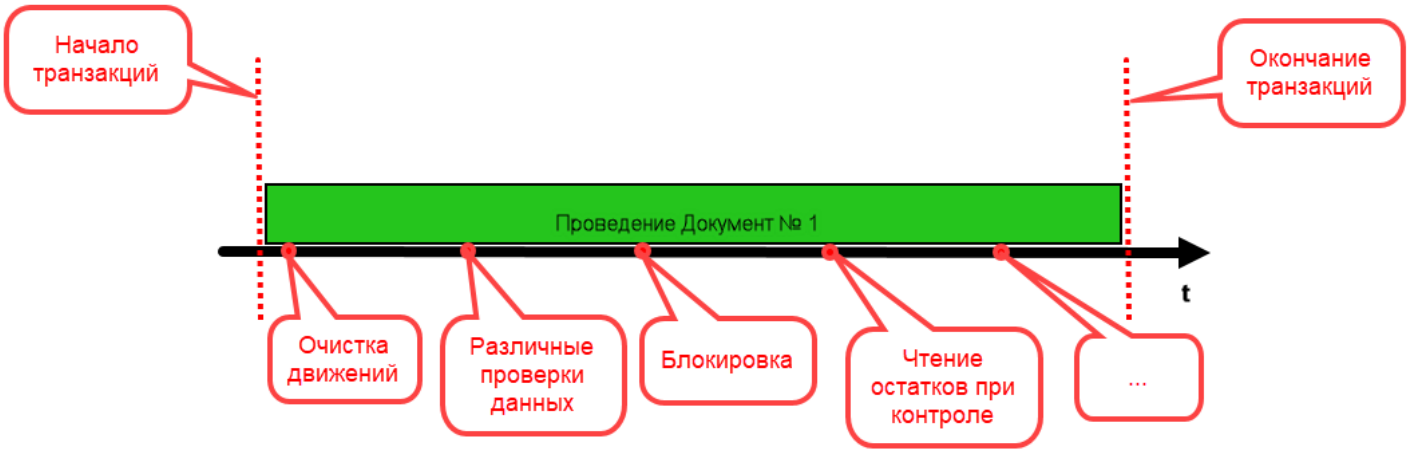


Рисунок 8 – Правильный вариант установки блокировки

Существует два режима работы с транзакционными блокировками:

- Автоматический режим блокировок
- Управляемый режим блокировок.

Далее рассмотрим каждый из этих режимов.

Автоматический режим блокировок

В этом режиме «1С:Предприятие» целиком «полагается» на возможности, предоставляемые СУБД (менеджером блокировок СУБД). Такой подход позволяет разработчику не задумываться о вопросах блокирования нужных данных в транзакции. Однако СУБД не имеет информации о логической структуре данных «1С:Предприятия», поэтому, чтобы обеспечить целостность и непротиворечивость данных, реализуется избыточная блокировка данных. Например, вместо блокировки записей может использоваться блокировка таблиц целиком.

Плюсы:

- Простота разработки: не нужно думать, как накладывать блокировки на данные.

Минусы:

- Избыточная блокировка данных мешает параллельной работе пользователей.

Управляемый режим блокировок

Все современные типовые конфигурации работают с использованием управляемых блокировок, так как они являются более оптимальными.

При управляемом режиме блокировок платформа «1С:Предприятие» использует собственный механизм транзакционных блокировок, независимый от СУБД. Это позволяет более точно блокировать необходимые данные.

Плюсы:

- Большие возможности по обеспечению параллельной работы пользователей, так как данные могут блокироваться более избирательно.

Минусы:

- Сложность разработки, необходимо устанавливать блокировки самостоятельно.

Установка режима блокировок в конфигурации

Устанавливать режим управления блокировкой данных можно как на уровне конфигурации (в свойствах конфигурации), так и на уровне объекта метаданных (в свойствах объекта).

На уровне конфигурации определяется значение свойства *Режим управления блокировкой данных*. При выборе для конфигурации значений *Автоматический* или *Управляемый* такой же режим блокировок при чтении или записи данных будет использоваться для любого объекта конфигурации, независимо от настроек собственных аналогичных свойств объекта:

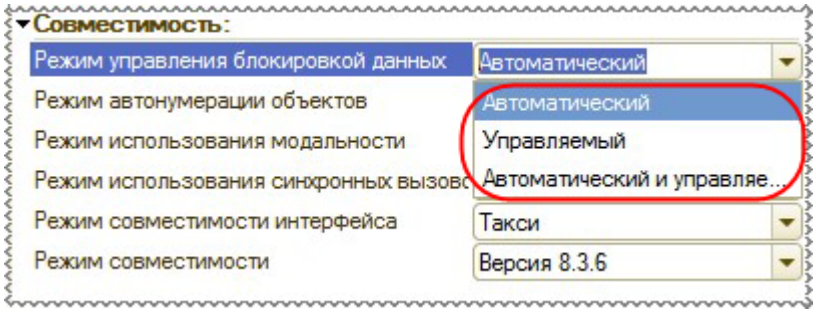


Рисунок 9 – Установка режима блокировок на уровне конфигурации

Если для соответствующего свойства конфигурации выбран режим *Автоматический и управляемый*, то для конкретного объекта конфигурации режим блокировки будет определяться его собственными настройками (*Автоматический* или *Управляемый*):

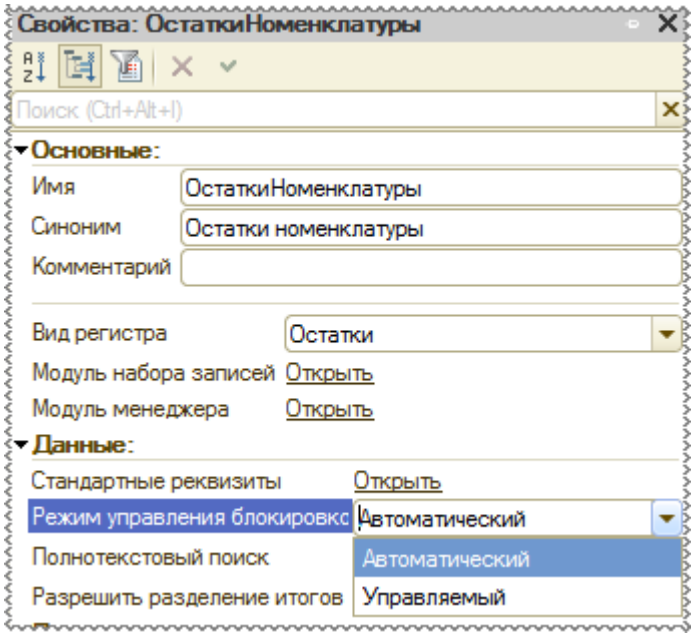


Рисунок 10 – Установка режима блокировок на уровне объекта конфигурации

Для конфигурации режим блокировок *Автоматический и Управляемый* используется, когда конфигурация находится в стадии перехода от автоматического к управляемому режиму блокировок. Все объекты метаданных постепенно переводятся в управляемый режим блокировок.

Важно. На экзамене необходимо установить режим управления блокировки *Управляемый* на уровне конфигурации, иначе экзаменатор снимет 1,5 балла:

Использование менее эффективной методики проведения документов	1,0
Использование автоматических блокировок в транзакции или неправильное использование управляемых блокировок данных в транзакции при проведения документов	1,5
Решение разных разделов (Оперативный учет, Бухгалтерский учет и т.п.) в отдельных информационных базах или с использованием отдельных конфигураций	3,0
Дублирование объектов конфигурации при решении отдельных микрозадач (например,	2,0

Рисунок 11 – Фрагмент из списка часто встречающихся ошибок

Работа с управляемыми блокировками средствами встроенного языка

Рассмотрим работу с управляемыми блокировками на примере контроля остатков в главе **«9. Как выполнить контроль остатков по старой методике»**. Для изучения блокировок усложним задачу: дополнительно учет будет вестись в разрезе складов.

В каркасную конфигурацию добавим:

- Справочник «Склады»
- В документы «Расходная накладная» и «Приходная накладная» реквизит «Склад».

Для того чтобы не получить отрицательные остатки при проведении документа «Расходная накладная», необходимо устанавливать управляемую блокировку по полям *Номенклатура* и *Склад*.

Листинг кода по блокировке будет следующим:

```
// Создание объекта блокировка
Блокировка = Новый БлокировкаДанных;
// Добавление элемента блокировки
Элемент = Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
// Установка режима блокировки
Элемент.Режим = РежимБлокировкиДанных.Исключительный;
// Установка полей блокировки
Элемент.УстановитьЗначение("Склад", Склад);
Элемент.ИсточникДанных = СписокНоменклатуры;
Элемент.ИспользоватьИзИсточникаДанных("Номенклатура", "Номенклатура");

//Блокировка данных
Блокировка.Заблокировать();
```

Рассмотрим ключевые точки алгоритма.

Создание объекта блокировки

Для работы с управляемыми блокировками во встроенном языке существует специальный объект *БлокировкаДанных*. Он создается с помощью конструктора *Новый*.

БлокировкаДанных представляет собой коллекцию элементов блокировки:

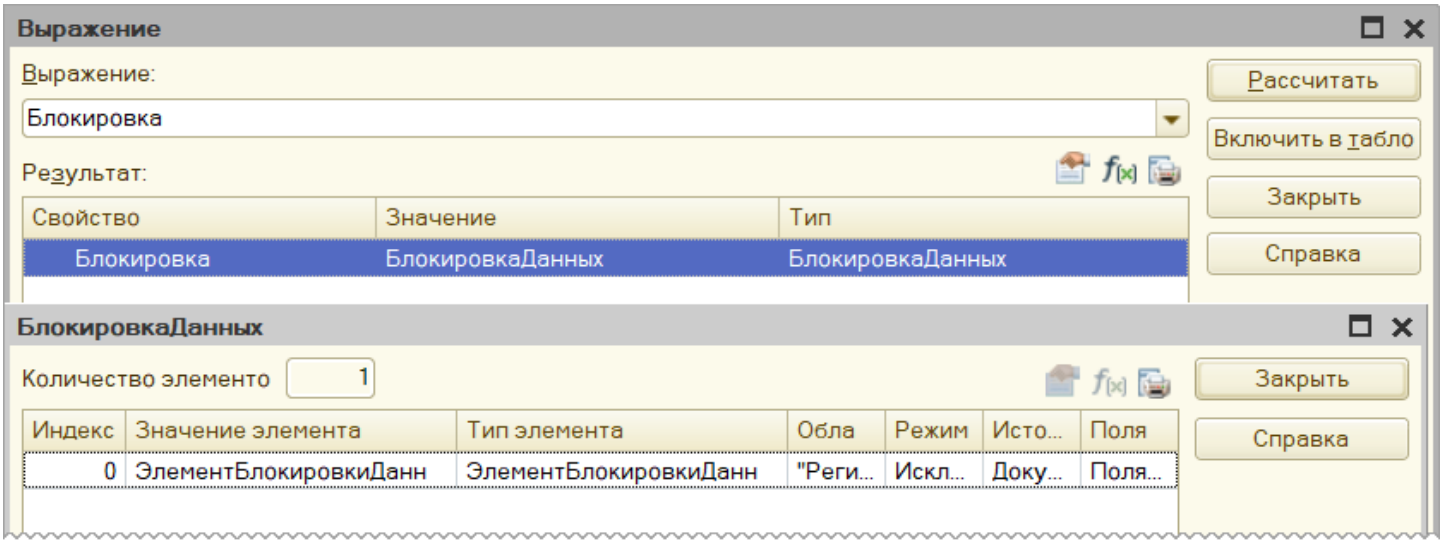


Рисунок 12 – Элементы блокировки данных

Добавление элемента блокировки

Элемент блокировки добавляется с помощью метода *Добавить* с указанием имени блокируемого объекта метаданных. Например: *РегистрНакопления.СебестоимостьТоваров*, *РегистБухгалтерии.Управленческий*, и т.д.

Установка режима блокировки

Существует два режима блокировок:

- **Исключительный** – заблокированные данные не могут быть прочитаны и изменены другой транзакцией
- **Разделяемый** – заблокированные данные могут быть прочитаны другой транзакцией, но не могут быть изменены.

Важно. На экзамене необходимо использовать *Исключительный режим блокировки*, так как для контроля остатков необходимо блокировать данные для чтения и изменения.

Установка полей блокировки

Помимо объекта блокировки указываются **поля блокировки и значения полей**. Блокировка будет наложена только на те записи, которые содержат значения в полях блокировки.

На рисунке отражена блокировка по полям: Номенклатура, Склад. В качестве значений полей были переданы: Монитор LG, Основной:

Номенклатура	Склад	Остаток
Мышь A4 TECH	Основной	3
Монитор LG	Основной	10
Монитор LG	Розничный	10
Фотоаппарат CANON 600D	Основной	4
Ноутбук ACER	Розничный	2

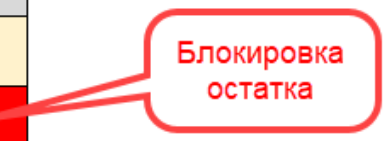


Рисунок 13 – Блокировка остатка по номенклатуре и складу

Поля блокировки можно задать двумя способами:

- С помощью метода *УстановитьЗначение()*. При этом указывается имя поля блокировки и значение. Данный способ используется, когда значение для поля блокировки находится в реквизите объекта. В данном примере *Склад*
- С помощью указания **Источника данных**, где хранятся значения полей блокировки. Для сопоставления имен полей источника данных и полей блокировки используется метод *ИспользоватьИзИсточникаДанных()*. Данный способ используется, когда значения для поля блокировки находится в таблице значений, табличной части или результате запроса.

Установка полей блокировки – важный шаг. Ошибки в нем ведут к избыточным или некорректным блокировкам. Рассмотрим несколько примеров **ошибочной** установки полей блокировок.

1) Листинг кода:

```
Блокировка = Новый БлокировкаДанных;
Элемент = Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
Элемент.Режим = РежимБлокировкиДанных.Исключительный;
Элемент.ИсточникДанных = СписокНоменклатуры;
Элемент.ИспользоватьИзИсточникаДанных("Номенклатура", "Номенклатура");
Блокировка.Заблокировать();
```

В рассмотренном выше примере не установили блокировку по полю *Склад*. В результате избыточная блокировка – заблокировались остатки по номенклатуре Монитор LG на всех складах:

Номенклатура	Склад	Остаток
Мышь A4 TECH	Основной	3
Монитор LG	Основной	10
Монитор LG	Розничный	10
Фотоаппарат CANON 600D	Основной	4
Ноутбук ACER	Розничный	2

Рисунок 14 – Избыточная блокировка остатка по складу

2) Листинг кода:

```
Блокировка = Новый БлокировкаДанных;
Элемент = Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
Элемент.Режим = РежимБлокировкиДанных.Исключительный;
Элемент.УстановитьЗначение("Склад", Склад);
Блокировка.Заблокировать();
```

В рассмотренном выше примере не установили блокировку по полю *Номенклатура*. В результате избыточная блокировка – на складе «Основной» заблокировались остатки по всей номенклатуре:

Номенклатура	Склад	Остаток
Мышь A4 TECH	Основной	3
Монитор LG	Основной	10
Монитор LG	Розничный	10
Фотоаппарат CANON 600D	Основной	4
Ноутбук ACER	Розничный	2

Рисунок 15 – Избыточная блокировка остатка по номенклатуре

3) Листинг кода:

```
Блокировка = Новый БлокировкаДанных;
Элемент = Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
Элемент.Режим = РежимБлокировкиДанных.Исключительный;
Блокировка.Заблокировать();
```

В рассмотренном выше примере не установили поля блокировки. В результате избыточная блокировка – заблокировался весь регистр накопления:

Номенклатура	Склад	Остаток
Мышь A4 TECH	Основной	3
Монитор LG	Основной	10
Монитор LG	Розничный	10
Фотоаппарат CANON 600D	Основной	4
Ноутбук ACER	Розничный	2

Рисунок 16 – Избыточная блокировка всего регистра

Блокировка данных

После того как все элементы блокировки добавлены, для осуществления блокировки данных необходимо вызвать метод *Заблокировать()*.

Неявные блокировки данных

Разберемся, всегда ли для установки управляемой блокировки данных нужно использовать объект *БлокировкаДанных*.

При записи любого объекта платформа неявно устанавливает исключительную управляемую блокировку. При этом управляемая блокировка устанавливается:

- Для ссылочных объектов (справочники, документы и т.д) на ссылку
- Для наборов записей регистров на значение измерений в данном наборе.

Для наборов записей регистров добавлено дополнительное свойство *БлокироватьДляИзменения*. При этом блокировка будет наложена вне зависимости от значения данного свойства. Если *БлокироватьДляИзменения* имеет значение «Истина», то блокировка будет без учета разделителя итогов, иначе – с учетом разделителя итогов. Установка данного свойства в значение «Истина» имеет смысл только для регистров, у которых включен режим *Разрешить разделение итогов:*

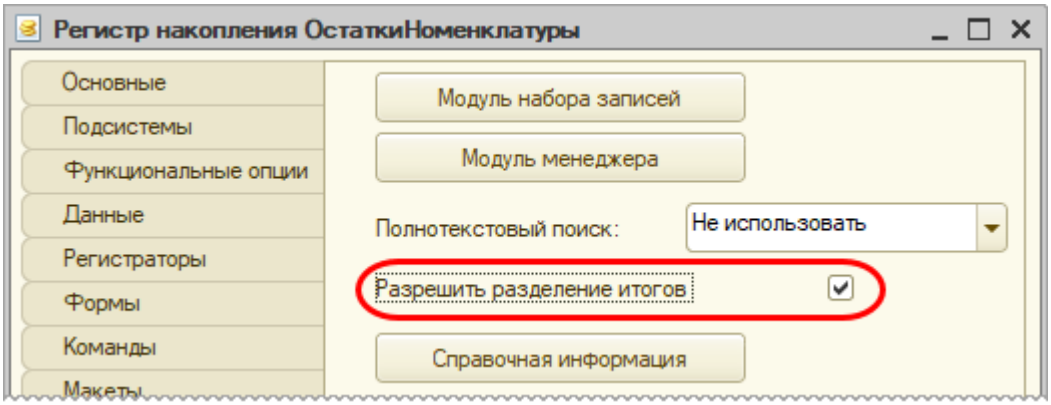


Рисунок 17 – Закладка *Прочее* РН «ОстаткиНоменклатуры»

При добавлении новых регистров накопления и бухгалтерии режим разделения итогов включен по умолчанию.

Для корректной работы контроля остатков при включенном режиме *Разрешить разделение итогов* для набора записей необходимо установить свойство *БлокироватьДляИзменения* в значение «Истина».

> *Что такое разделение итогов и когда его нужно использовать, подробно описано в статье **«Оптимизация 1С – Режим разделения итогов»** (https://курсы-по-1с.рф/articles/режим-разделения-итогов).*

Важно. На экзамене рекомендуется устанавливать свойство *БлокироватьДляИзменения* в значение «Истина».

Подведем итоги: нет необходимости использовать объект БлокировкаДанных для установки управляемой блокировки перед записью данных. Платформа сама установит неявную блокировку данных до конца транзакции.

Управляемая блокировка при контроле остатков

Распространенный пример использования управляемых блокировок – блокировка данных регистра для целей контроля остатков. В этом случае использование блокировки данных необходимо, чтобы несколько пользователей, работая параллельно, не списали какой-либо остаток (товаров, взаиморасчетов, денежных средств) в минус.

Управляемая блокировка с помощью объекта *БлокировкаДанных* была установлена при решении задачи контроля остатков при проведении документа «Расходная накладная» по «старой» методике.

В «новой» методике контроля остатков блокировки накладываются в момент записи движений в регистр «Остатки номенклатуры». При этом в «новой» методике контроля остатков для блокировки разделяемых ресурсов нужна лишь одна строка кода. Нужно просто установить для свойства *БлокироватьДляИзменения* набора записей регистра значение *Истина* (см. ранее рассмотренное

решение задачи контроля остатков при проведении документа «Расходная накладная» по «новой» методике).

Подведение итогов

Почти во всех современных типовых конфигурациях 1С используются управляемый режим блокировки данных. Для успешной сдачи экзамена «1С:Специалист по платформе» в разрабатываемой конфигурации необходимо использовать именно этот режим.

Установку управляемых блокировок нужно обязательно выполнять на конкурирующие ресурсы, которые используются в транзакциях (в частности, при проведении документов). Под такими конкурирующими ресурсами подразумеваются, например, товары, реализацию которых могут оформлять несколько пользователей одновременно.

В «старой» методике контроля остатков используются программные блокировки средствами встроенного языка.

В «новой» методике используется свойство набора записей регистра *БлокироватьДляИзменения*.

> *Обратите внимание, что проверить работу управляемых блокировок в файловой БД, используемой на экзамене, невозможно. Управляемые блокировки эффективны только в клиент-серверном варианте работы БД. В связи с этим, в текущем курсе информация о блокировках приведена в сжатом объеме, достаточном для сдачи экзамена. Если необходима полная информация по теории, настройке и оптимизации блокировок, то найти ее можно в курсе «**Ускорение и оптимизация систем на 1С:Предприятие 8.3 + подготовка на 1С:Эксперт**».*

12. Что такое «проблема копеек» и как ее решить

Упоминание о существовании некой «проблемы копеек» встречается в требованиях к экзамену в списке наиболее частых ошибок, характерных для любой учетной задачи:

Отсутствие в решении проверок на правильное заполнение ресурсов регистра, приводящее, например, к появлению отрицательных остатков товаров на складе. Наличие отрицательных значений ресурсов регистра допустимо, только если об этом явно сказано в задании или следует из логики учетной схемы, не противоречащей ситуации, возникающей в реальной практике ведения учета	1,0 - 2,0
Использование неправильных или упрощенных алгоритмов при расчете значений ресурсов регистра. Например, при решении «проблемы копеек»	0,5 - 2,0
Построенная в решении учетная схема не обеспечивает правильного занесения данных в регистры. Например, необходимо списать 1000, а списывается 500	2,0

Рисунок 1 – Фрагмент из списка часто встречающихся ошибок

Что же такое «проблема копеек»?

В задачах списания товаров «Проблема копеек» выражается в следующем: при полном списании товара по количеству его себестоимость не списывается в «ноль». Остается несколько копеек. Связано это с округлениями при расчете себестоимости списания.

> *Видим, что следствием нерешенной «проблемы копеек» является еще и другая ошибка: не будут выведены одновременно в ноль все ресурсы регистра остатков. За такую ошибку на экзамене снижают от 0,5 до 3 баллов.*

Пример: на остатке есть 3 ручки общей стоимостью 100,00 руб. Спишем 3 ручки. Расчет себестоимости списания выполним по средней.

Формула: Себестоимость Остаток / Количество Остаток * Количество Списания

Себестоимость единицы товара составит 100 / 3 = 33,33 руб.

Себестоимость списания согласно формуле составит 33,33 * 3 = 99,99 руб.

Итого после списания останется 0 шт по количеству и 1 копейка по стоимости.

Возникает вопрос: зачем рассчитывать себестоимость за единицу и уж тем более ее округлять? Давайте сразу применим формулу без промежуточных округлений.

Вычисления система будет выполнять последовательно:

100 / 3*3 = 33.33333...* 3 = 99.99999…

Ресурсы для хранения показателей в денежном выражении имеют точность два знака после запятой. При записи значения в ресурс регистра произойдет округление по математическим правилам. Записано будет 100,00 руб.

Казалось бы, все хорошо. Но в таком варианте на экзамене «проблема копеек» будет считаться нерешенной. Почему? Из-за тех самых 99.999999. Правильный результат в данном случае был получен в результате автоматического округления.

Как решить «проблему копеек»?

Если на результат влияет порядок операндов, давайте его изменим.

Вариант формулы № 2: Количество списания / Количество остаток х Сумма остаток

Себестоимость списания трех ручек из примера составит 3 / 3 * 100 = 1 * 100 = 100 руб.

Вариант формулы № 3: Сумма остаток х Количество списания / Количество остаток.

На данных из примера получается: 100 * 3 / 3 = 300 / 3 = 100.

Видим, что в обоих случаях автоматическое округление при записи в регистр выполняться не будет.

Означает ли это, что поменяв местами операнды, мы решили «проблему копеек»?

Ответ: не во всех случаях. Есть ограничение: «проблема копеек» будет решена только в том случае, если на последнем списании значения переменных **КоличествоОстаток** и **КоличествоСписания** будут равны. Тогда на последнем списании формула № 2 превращается в вариант:

Количество остаток / Количество остаток х Сумма остаток = 1 * Сумма остаток = Сумма остаток.

В таком случае решение по формуле с правильной последовательностью операндов принимается на экзамене. Но на очной аттестации нужно быть готовым доказать, почему в случае Вашей задачи такой метод подходит.

Далее при разборе практического примера увидим, что с определением момента последнего списания не всегда все так просто.

Надежный способ решения «проблемы копеек»

Суть метода проста: при последнем списании товара списываем весь остаток стоимости, не применяя формулу для расчета суммы списания.

Именно этот метод рекомендуется использовать на аттестации.

Как определить, что списание последнее? При последнем списании количество остатка совпадет с количеством списания.

Усовершенствованная формула для расчета себестоимости списания будет выглядеть так:

```
Если КоличествоСписания = КоличествоОстаток Тогда
      СебестоимостьСписания = СебестоимостьОстаток
Иначе
      СебестоимостьСписания = СебестоимостьОстаток*КоличествоСписания/КоличествоОстаток
КонецЕсли;
```

или так:

```
СебестоимостьСписания=?(КоличествоСписания = КоличествоОстаток, СебестоимостьОстаток,
СебестоимостьОстаток*КоличествоСписания/КоличествоОстаток);
```

Будем называть эту формулу **формулой с учетом последнего списания**.

Как обстоят дела с использованием этой формулы? Просто везде и всегда в алгоритме вставляем ее и считаем, что «проблема копеек» у нас теперь заведомо решена? Нет. Есть все тот же нюанс в определении момента последнего списания. Но в этой формуле ничто не мешает вставить в конструкцию «Если…» другое условие для определения момента последнего списания.

Практический пример расчета себестоимости по средней с решением «проблемы копеек»

Постановка задачи:

Складской учет товаров не ведется. Приход товаров оформляется документом «Приходная накладная». Расход товаров оформляется документом «Расходная накладная». Себестоимость списания определяется как средняя по номенклатурной позиции».

> *Фраза в условии «Складской учет товаров не ведется» означает, что приход и расход товаров (количественный учет) не ведется в разрезе складов. Учет по количеству ведется только в разрезе номенклатуры.*

Нам понадобятся объекты метаданных:

- справочник «Номенклатура»
- документ «Приходная накладная»
- документ «Расходная накладная».

Эти объекты в каркасной конфигурации уже есть. Для решения задачи они полностью подходят.

Для расчета себестоимости списания номенклатуры нужно знать остаток себестоимости и остаток по количеству. Для этих целей понадобится регистр накопления. Вид регистра – «Остатки», т.к. интересуют именно остатки стоимости и количества. В регистре должно быть одно измерение *Номенклатура*, т.к. остатки нужны в разрезе номенклатуры. И должно быть два числовых ресурса – *Количество* и *Сумма*.

В каркасной конфигурации есть РН «ОстаткиНоменклатуры».

Измерение «Номенклатура» в нем есть, установим для него свойство *Запрет незаполненных значений* в истину, т.к. движения по регистру с пустой номенклатурой не имеют смысла.

Ресурс *Количество* есть. Добавим ресурс *Сумма* (Число 12, 2):

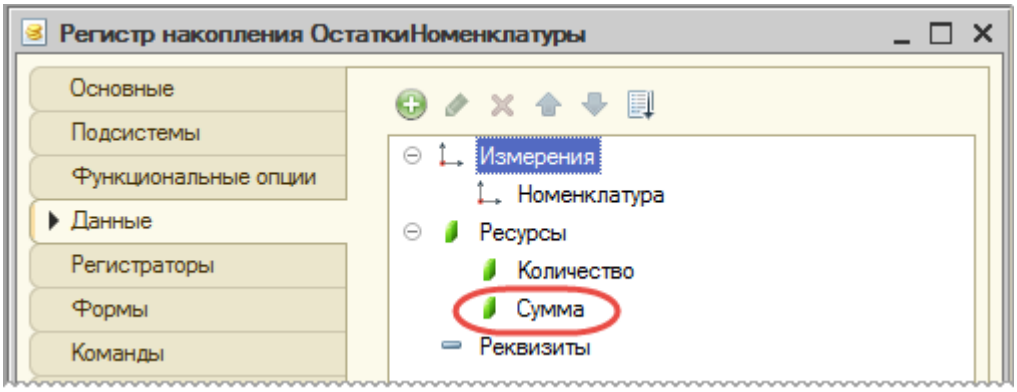


Рисунок 2 – Структура РН «Остатки номенклатуры»

Регистраторы:

- «Приходная накладная»
- «Расходная накладная»:

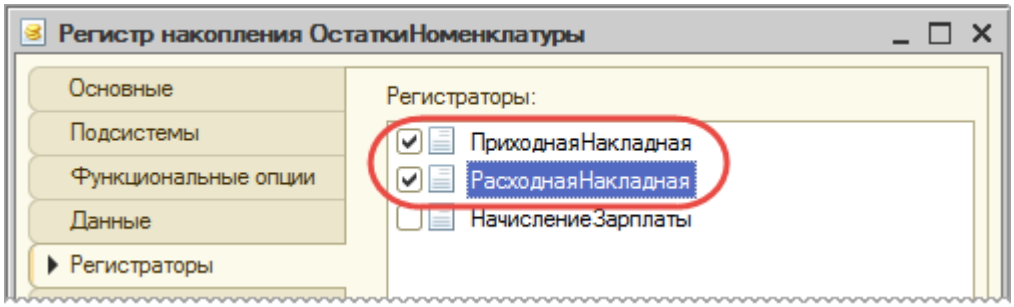


Рисунок 3 – Регистраторы РН «Остатки номенклатуры»

Обработка проведения для документа «Приходная накладная»

С помощью конструктора движений необходимо сформировать движения документа «Приходная накладная» по регистру накопления «Остатки номенклатуры» с видом *Приход*. Как это сделать, подробно рассмотрено в главе **«6. Как реализовать поступление товаров в компанию»**.

Обработка проведения для документа «Расходная накладная»

Для документа «Расходная накладная» обойтись возможностями конструктора движений не получится, потому что данных из табличной части документа недостаточно для формирования движений по списанию товаров. Понадобятся еще данные об остатках количества и себестоимости по РН «Остатки номенклатуры».

Создадим процедуру *ОбработкаПроведения* в модуле объекта и заполним ее вручную.

> *Примечание: При решении данной задачи ограничимся только решением задачи списания себестоимости и «проблемы копеек». Контроль отрицательных остатков, установку блокировки в рамках данного решения рассматривать и выполнять не будем. Про блокировки можно узнать в главе **«11. Какие навыки в использовании управляемых блокировок потребуются на экзамене»**, про контроль отрицательных остатков – **см. главу «9. Как выполнить контроль остатков по старой методике»**.*

Листинг обработки проведения:

В коде преднамеренно закомментированы некоторые строки. Позже разберем последствия.

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
	// 1. Подготовка наборов записей регистра
	Движения.ОстаткиНоменклатуры.Записывать = Истина;
	Движения.Записать();
	// 2. Установка маркера Записи у регистра
	Движения.ОстаткиНоменклатуры.Записывать = Истина;

	// 3. Запрос для получения данных для расчета себестоимости
	Запрос = новый Запрос("ВЫБРАТЬ
	|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура КАК Номенклатура,
	//  |	СУММА(РасходнаяНакладнаяСписокНоменклатуры.Количество) КАК Количество
	|	РасходнаяНакладнаяСписокНоменклатуры.Количество КАК Количество
	|ПОМЕСТИТЬ ТЧСписокНоменклатуры
	|ИЗ
```

```
	|		Документ.РасходнаяНакладная.СписокНоменклатуры КАК
РасходнаяНакладнаяСписокНоменклатуры
	|ГДЕ
	|		РасходнаяНакладнаяСписокНоменклатуры.Ссылка = &Ссылка
	|
	//	|СГРУППИРОВАТЬ ПО
	//	|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура
	|
	|ИНДЕКСИРОВАТЬ ПО
	|		Номенклатура
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|		ТЧСписокНоменклатуры.Номенклатура,
	|		ТЧСписокНоменклатуры.Количество,
	|		ЕСТЬNULL(ОстаткиНоменклатурыОстатки.СуммаОстаток, 0) КАК СуммаОстаток,
	|		ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) КАК КоличествоОстаток
	|ИЗ
	|		ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|			ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиНоменклатуры.Остатки(
	|					&МоментВремени,
	|					Номенклатура В
	|						(ВЫБРАТЬ
	|							ТЧСписокНоменклатуры.Номенклатура
	|						ИЗ
	|							ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры)) КАК
ОстаткиНоменклатурыОстатки
	|			ПО ТЧСписокНоменклатуры.Номенклатура =
ОстаткиНоменклатурыОстатки.Номенклатура");

	//4. Заполнение параметров запроса
	Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
	Запрос.УстановитьПараметр("Ссылка", Ссылка);

	//5. выполняем запрос, обходим выборку, рассчитываем себестоимость и формируем движения
	Результат = Запрос.Выполнить();
	ВыборкаНоменклатура = Результат.Выбрать();
	Пока ВыборкаНоменклатура.Следующий() Цикл

		КоличествоСписания = ВыборкаНоменклатура.Количество;
		КоличествоОстаток = ВыборкаНоменклатура.КоличествоОстаток;
		СуммаОстаток = ВыборкаНоменклатура.СуммаОстаток;

		Движение = Движения.ОстаткиНоменклатуры.ДобавитьРасход();
		Движение.Период = Дата;
		Движение.Номенклатура = ВыборкаНоменклатура.Номенклатура;
		Движение.Количество = КоличествоСписания;

		//6. расчет себестоимости списания по формулам с разным порядком операндов
		ВариантРасчета1 = СуммаОстаток/КоличествоОстаток*КоличествоСписания;
		Сообщить("Вариант 1: СуммаОстаток/КоличествоОстаток*КоличествоСписания
```

```
="+строка(ВариантРасчета1));
                Движение.Сумма = ВариантРасчета1;
                Сообщить("Вариант 1: Сумма в регистре ="+строка(Движение.Сумма));

                ВариантРасчета2 = КоличествоСписания/КоличествоОстаток*СуммаОстаток;
                Сообщить("Вариант 2: КоличествоСписания/КоличествоОстаток*СуммаОстаток
="+строка(ВариантРасчета2));
                Движение.Сумма = ВариантРасчета2;
                Сообщить("Вариант 2: Сумма в регистре ="+строка(Движение.Сумма));

                ВариантРасчета3 = СуммаОстаток*КоличествоСписания/КоличествоОстаток;
                Сообщить("Вариант 3: СуммаОстаток*КоличествоСписания/КоличествоОстаток
="+строка(ВариантРасчета3));
                Движение.Сумма = ВариантРасчета3;
                Сообщить("Вариант 3: Сумма в регистре ="+строка(Движение.Сумма));

                //7. списание полного остатка стоимости при последнем списании
                ВариантРасчета4=?(КоличествоСписания = КоличествоОстаток,СуммаОстаток,
                                        СуммаОстаток*КоличествоСписания/КоличествоОстаток);
                Сообщить("Вариант 4: метод с контролем последнего списания ="+строка(ВариантРасчета4));
                Движение.Сумма = ВариантРасчета4;
                Сообщить("Вариант 4: Сумма в регистре ="+строка(Движение.Сумма));

        КонецЦикла;

КонецПроцедуры
```

Разберем ключевые точки алгоритма

Подготовка регистров к работе (п. 1, 2)

1. Очищаем движения, чтобы при перепроведении документа не учитывались старые движения документа. Это происходит, когда дата документа при перепроведении сдвигается вперед. Для очистки старых движений документа используем принудительную запись пустого набора. В этом случае при чтении данных из регистра старые движения документа в расчет виртуальных таблиц не попадут.

2. Устанавливаем маркер записи движений в значение *Истина*, чтобы при проведении документа записались сформированные в обработке проведения движения. В этом случае в конце процедуры обработки проведения использовать метод *Движения.Записать()* не нужно.

Запрос получения данных для расчета себестоимости (п. 3)

Первый запрос пакета – это получение данных из табличной части документа.

- **Внимание**: группировка строк по номенклатуре закомментирована преднамеренно! Позже разберем, к чему это приведет.
- В списке полей выбора оставляем только *Номенклатуру* и *Количество*. Для расчета себестоимости списания нужны только они. Представление номенклатуры не выводим, т.к. не планируем выводить его в сообщениях и т.п.
- Устанавливаем отбор по *Ссылке* для отбора данных только по нашему документу.
- Для оптимизации производительности индексируем поле *Номенклатура*. Далее по нему будем соединяться с другой таблицей.
- Результат помещаем во временную таблицу *ТЧСписокНоменклатуры*.

Второй запрос пакета

Источниками данных для второго запроса послужат:

- Временная таблица *ТЧСписокНоменклатуры*, подготовленная первым запросом
- Виртуальная таблица остатков РН «ОстаткиНоменклатуры» (*ОстаткиНоменклатурыОстатки)*.

  Будем использовать следующие параметры виртуальной таблицы остатков:

  - *Период* – это значение параметра *МоментВремени*
  - *Условие* – ограничиваем выборку только номенклатурой из временной таблицы *ТЧСписокНоменклатуры*. Интересуют остатки только по этой номенклатуре:

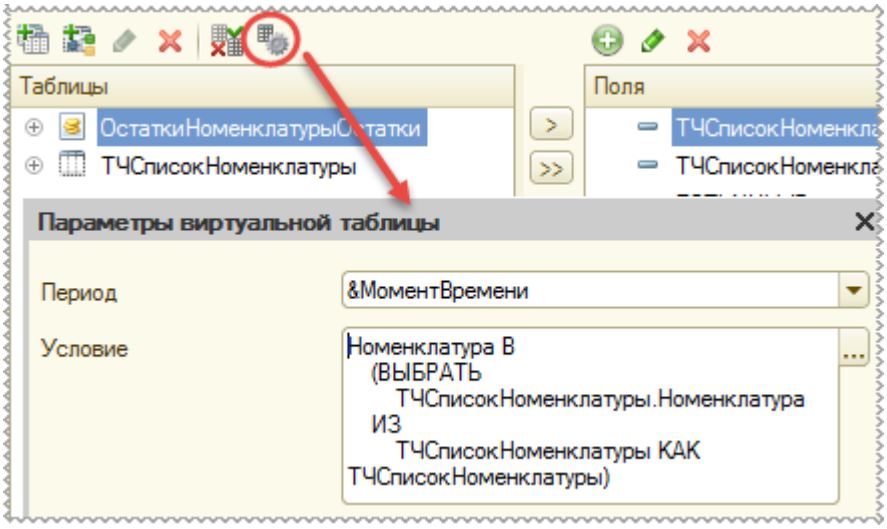


Рисунок 4 – Параметры виртуальной таблицы остатков РН «Остатки номенклатуры»

Временную таблицу *ТЧСписокНоменклатуры* соединяем с таблицей *ОстаткиНоменклатурыОстатки* по полю *Номенклатура*. Используем левое соединение на случай, если соответствующих записей в таблице остатков не будет. По этой же причине используем функцию ЕстьNULL() применительно к полям выбора из регистра остатков.

Заполнение параметров запроса (п. 4)

Для заполнения параметра *Момент времени* будем использовать момент времени документа.

МоментВремени рассмотрен в главе 3 «Понятие момента времени».

Цикл по выборке из запроса (п. 5)

Обходим выборку из результата запроса и на каждую строку выборки создаем движение по расходу по РН «Остатки номенклатуры». Внутри цикла выполняем расчет себестоимости.

Расчет себестоимости и формирование движений (п. 6)

В алгоритме приведены все варианты построения формулы для расчета себестоимости списания, которые разбирались в теоретической части. Цель – на практике посмотреть, как влияет перестановка операндов на результат. При решении задачи на экзамене этот фрагмент, конечно же, не нужен.

Применение формулы, рекомендуемой на экзамене (п. 7)

Последний вариант – это как раз вариант, рекомендуемый на экзамене. Т.к. этот вариант расчета выполняется последним в списке формул, то в регистр будет записан результат расчета именно по этой формуле.

Проверка результата в режиме «1С:Предприятие»

Создадим документ «Приходная накладная»:

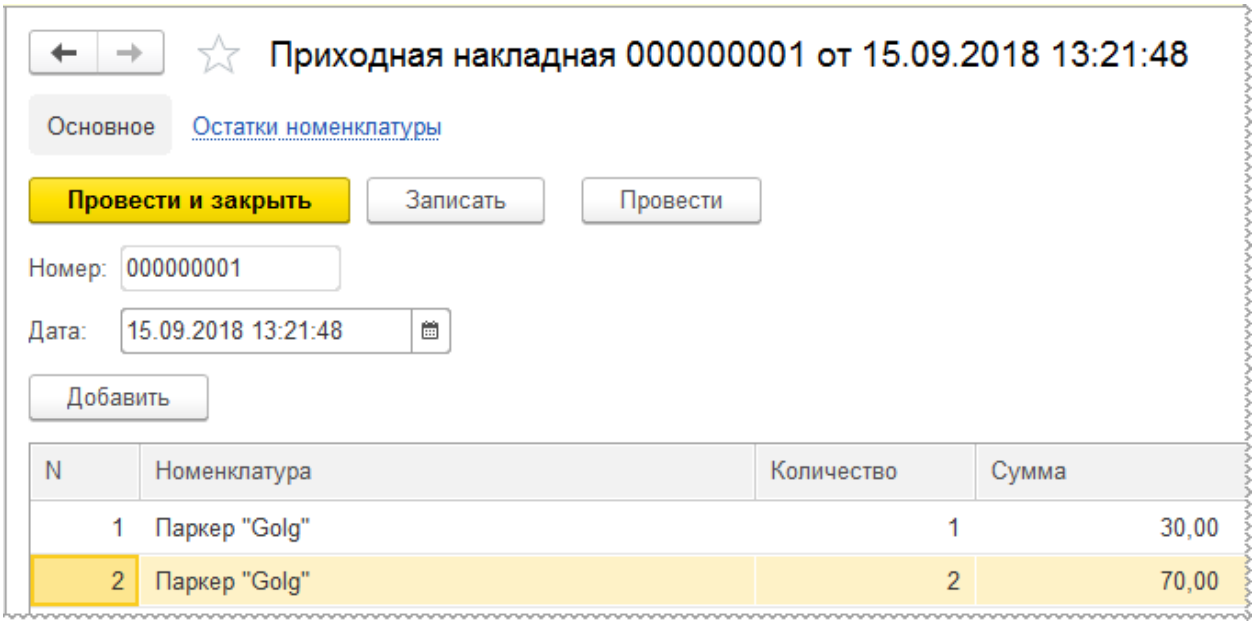


Рисунок 5 – Документ «Приходная накладная № 1»

Проверим движения по РН «Остатки номенклатуры»:

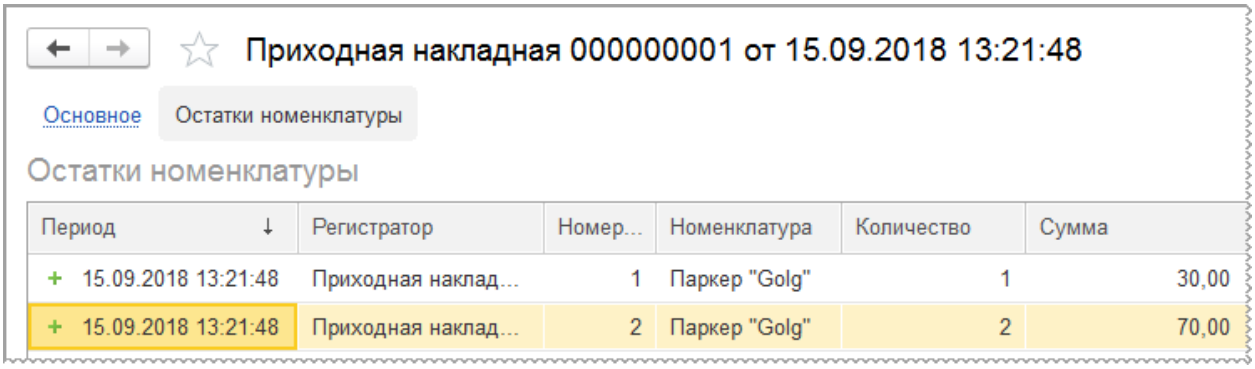


Рисунок 6 – Движения документа «Приходная накладная № 1»

Приход сформирован. Займемся списанием товаров.

Создадим документ «Расходная накладная». Выполним в нем списание всех трех ручек, которые имеются на остатке.

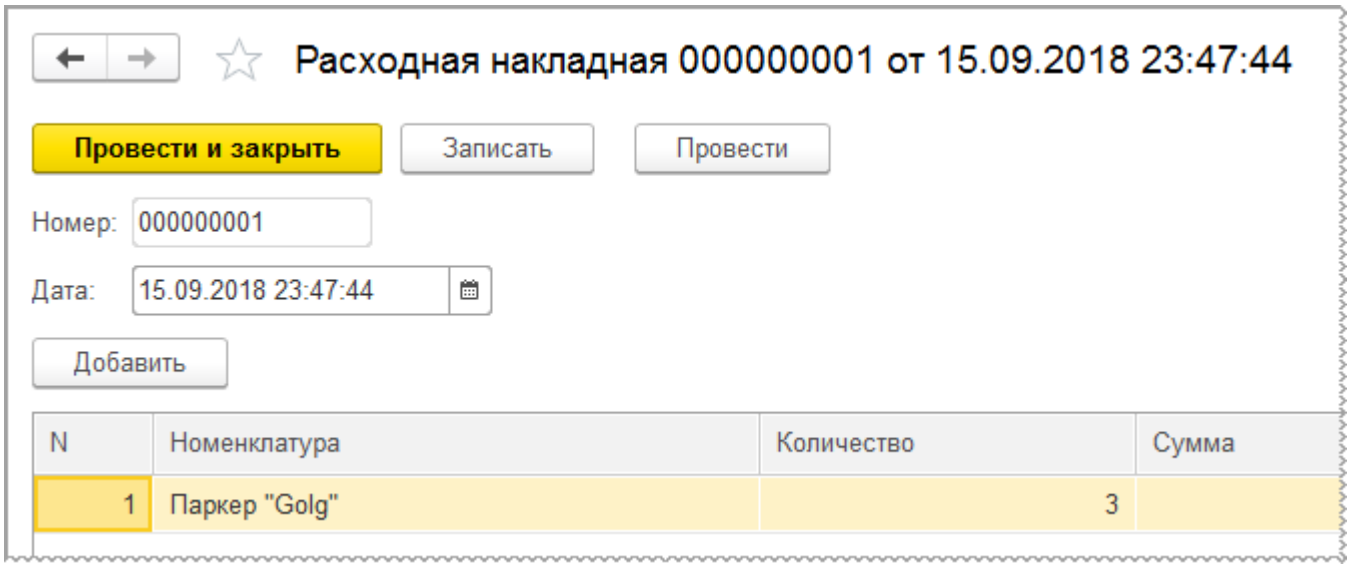


Рисунок 7 – Документ «Расходная накладная № 1»

При проведении получим следующие сообщения:

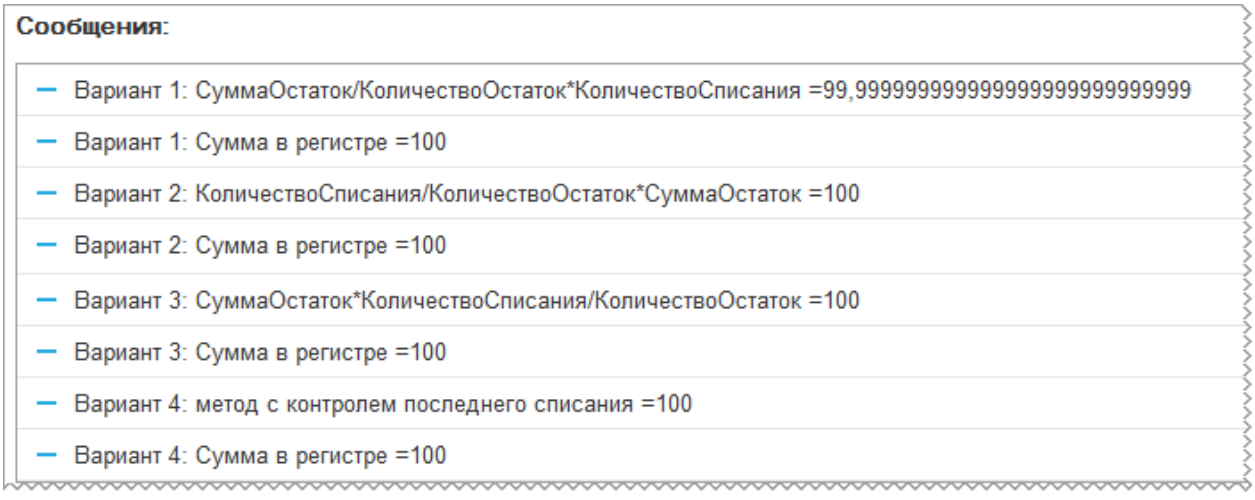
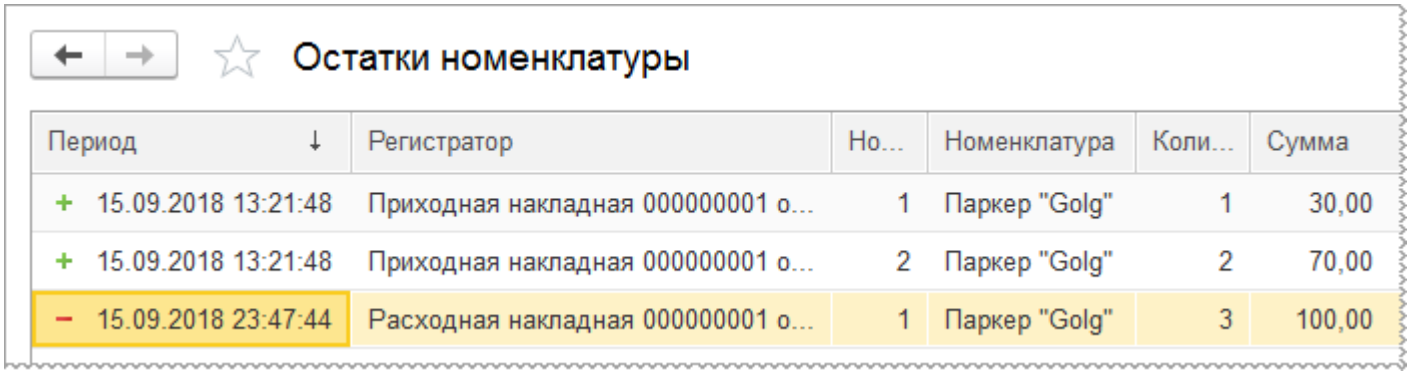


Рисунок 8 – Сообщения при проведении документа «Расходная накладная № 1»

Полученные результаты вполне ожидаемы и подтверждают теоретические изыскания.

Если заглянуть в движения по РН «Остатки номенклатуры», то там тоже все отлично – никаких копеек:



Остатки номенклатуры

Период	Регистратор	Но...	Номенклатура	Коли...	Сумма
+ 15.09.2018 13:21:48	Приходная накладная 000000001 о...	1	Паркер "Golg"	1	30,00
+ 15.09.2018 13:21:48	Приходная накладная 000000001 о...	2	Паркер "Golg"	2	70,00
− 15.09.2018 23:47:44	Расходная накладная 000000001 о...	1	Паркер "Golg"	3	100,00

Рисунок 9 – Движения по РН «Остатки номенклатуры»

Подведем итоги

Мы выяснили, что скрывается под названием «проблема копеек», разобрали методы ее решения, проверили их на практике, решив пример с расчетом себестоимости списания по средней.

Осталось выяснить, в чем может быть нюанс с определением момента последнего списания, о котором упоминали при разборе формул. Сделаем это в следующем блоке материалов.

13. «Подводный камень» проблемы копеек: определение момента последнего списания

Вернемся к тестовому примеру из предыдущего блока. В документе «Расходная накладная № 1» внесем некоторые изменения – создадим строки с одинаковой номенклатурой:

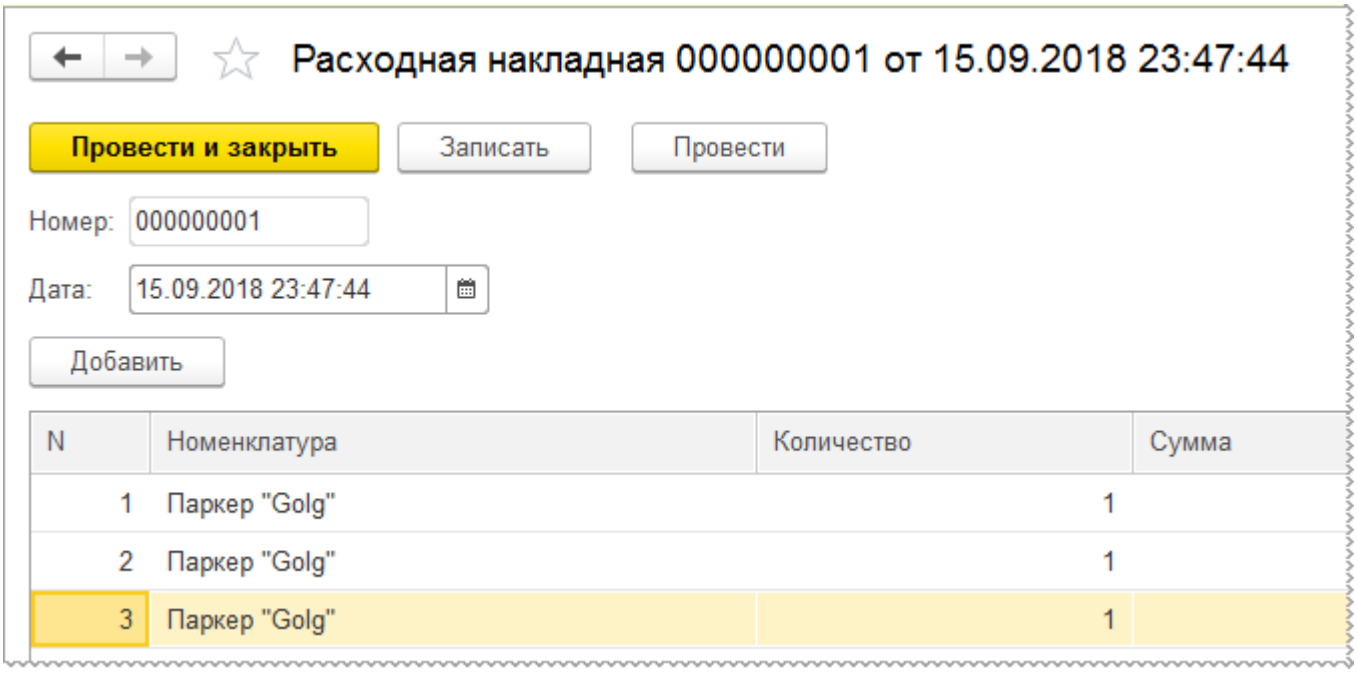


Рисунок 1 – Документ «Расходная накладная № 1»

Проводим документ и в движениях по регистру видим нерешенную «проблему копеек»:

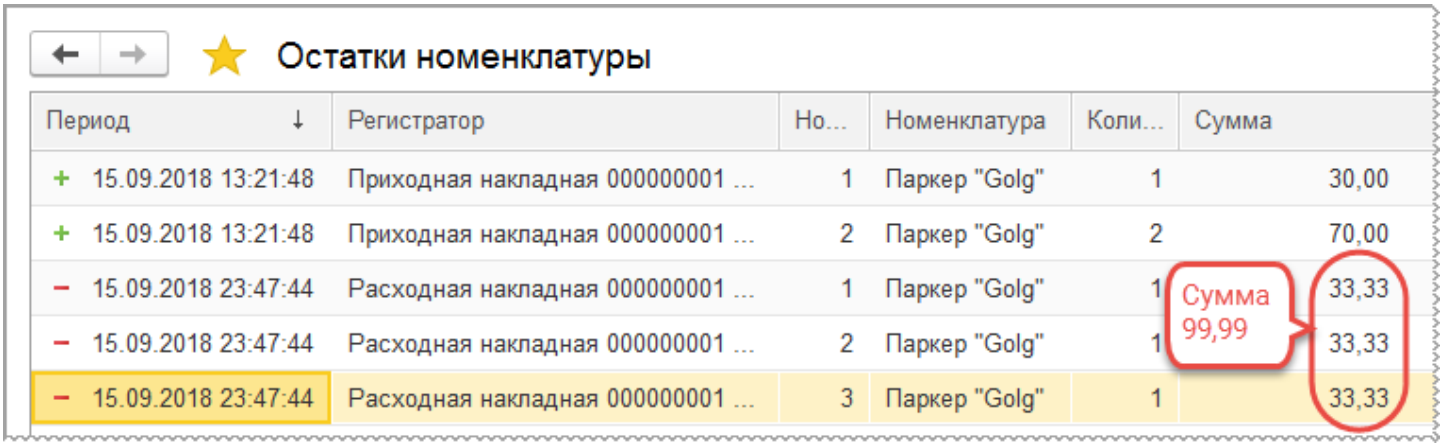


Рисунок 2 – Движения по РН «Остатки номенклатуры»

При этом результат будет совершенно одинаков для каждой формулы из списка. Чтобы убедиться, достаточно посмотреть в диагностические сообщения при проведении «Расходной накладной».

Давайте разбираться, в чем дело.

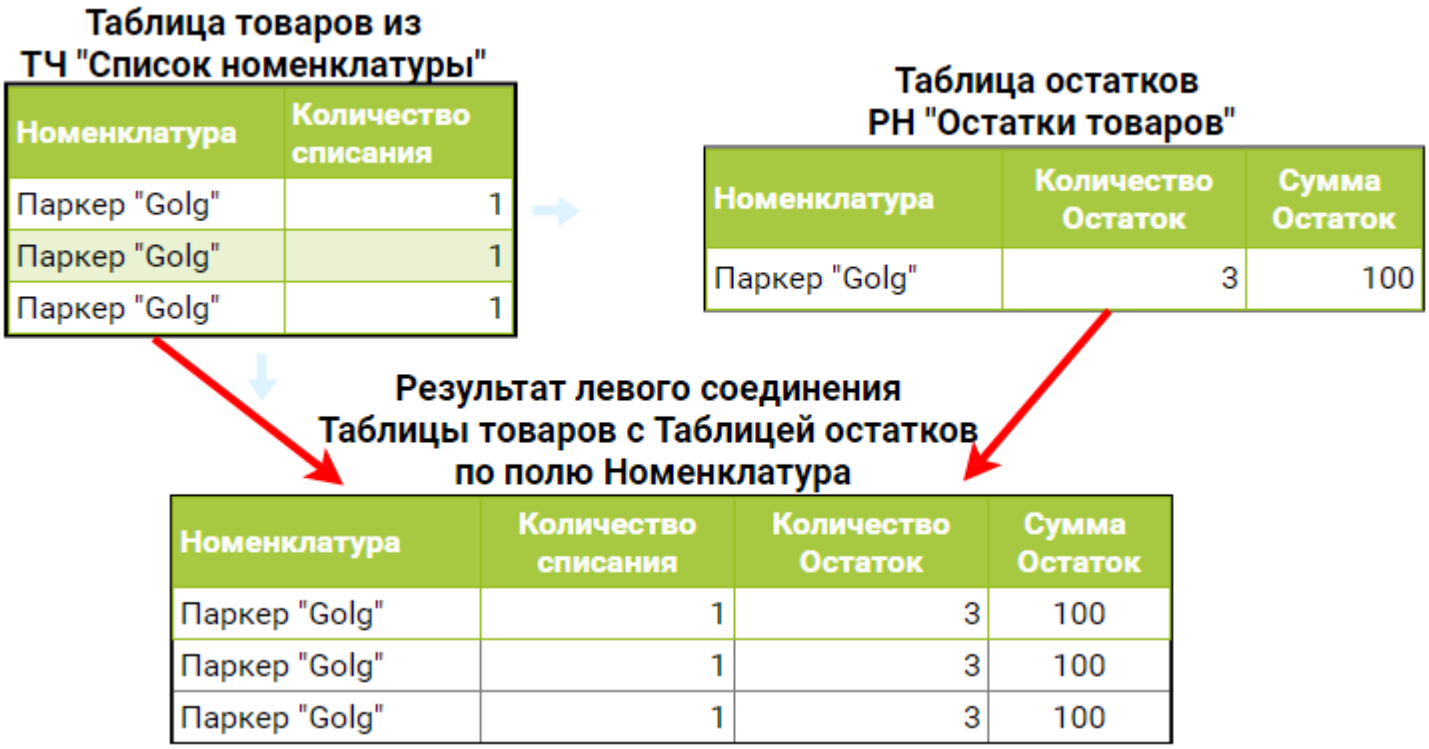


Рисунок 3 – Результат запроса получения данных для расчета себестоимости

Из рисунка видим, что при обходе выборки из результата запроса условие *КоличествоОстаток=КоличествоСписания* для каждой строки всегда равно *ложь,* поэтому всегда выполняется расчет по формуле 100 * 1 / 3 = 33,33 руб.

Вспомним, что при подготовке запроса был преднамеренно закомментирован код для группировки по номенклатуре. Вот это и есть последствия: если забыть сделать группировку товаров, то «проблема копеек» останется нерешенной.

В процессе решения задачи группировка данных табличной части документа может быть не только «забытой». Она может быть нецелесообразной из-за каких-либо условий задачи или, например, можно получать одним и тем же запросом данные не только для формирования движений по списанию себестоимости, но и по другому регистру.

Так, в реальной практике может быть необходимо записывать в движения по списанию себестоимости еще какую-либо дополнительную информацию из табличной части в реквизиты

регистра, например, цену продажи или склад-отправитель. В задачах из сборника таких требований нет, но в жизни такие требования вполне могут быть.

Разберем, что в такой ситуации нужно изменить в алгоритме, чтобы все-таки решить «проблему копеек».

Все просто. В результирующем запросе будем получать итоги по полям, в разрезе которых получаем данные из таблицы остатков. В нашей задаче это – *Номенклатура*:

Рисунок 6 – Движения по РН «Остатки номенклатуры»

Из схемы видим, что можно определить, является ли списание последним, если сравнить списанное количество по детальным записям со значением *КоличествоОстаток* из строки итогов.

Для обхода выборки из результата запроса нам понадобятся два цикла: один для обхода по иерархии, а второй по детальным записям.

Новый листинг обработки проведения:

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
	// 1. Подготовка наборов записей регистра
	Движения.ОстаткиНоменклатуры.Записывать = Истина;
	Движения.Записать();
	// 2. Установка маркера Записи у регистра
	Движения.ОстаткиНоменклатуры.Записывать = Истина;

	// 3. Запрос для получения данных для расчета себестоимости
	Запрос = новый Запрос("ВЫБРАТЬ
	|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура КАК Номенклатура,
	|	РасходнаяНакладнаяСписокНоменклатуры.Количество КАК Количество
```

```
	|ПОМЕСТИТЬ ТЧСписокНоменклатуры
	|ИЗ
	|	Документ.РасходнаяНакладная.СписокНоменклатуры КАК
РасходнаяНакладнаяСписокНоменклатуры
	|ГДЕ
	|	РасходнаяНакладнаяСписокНоменклатуры.Ссылка = &Ссылка
	|
	|ИНДЕКСИРОВАТЬ ПО
	|	Номенклатура
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	ТЧСписокНоменклатуры.Номенклатура КАК Номенклатура,
	|	ТЧСписокНоменклатуры.Количество КАК Количество,
	|	ЕСТЬNULL(ОстаткиНоменклатурыОстатки.СуммаОстаток, 0) КАК СуммаОстаток,
	|	ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) КАК КоличествоОстаток
	|ИЗ
	|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиНоменклатуры.Остатки(
	|				&МоментВремени,
	|				Номенклатура В
	|					(ВЫБРАТЬ
	|						ТЧСписокНоменклатуры.Номенклатура
	|					ИЗ
	|						ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры)) КАК
ОстаткиНоменклатурыОстатки
	|		ПО ТЧСписокНоменклатуры.Номенклатура = ОстаткиНоменклатурыОстатки.Номенклатура
	|ИТОГИ
	|	СУММА(Количество),
	|	МАКСИМУМ(СуммаОстаток),
	|	МАКСИМУМ(КоличествоОстаток)
	|ПО
	|	Номенклатура");

	//4. Заполнение параметров запроса
	Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
	Запрос.УстановитьПараметр("Ссылка", Ссылка);

	//5.выполнение запроса и организация цикла по итоговым записям
	Результат = Запрос.Выполнить();
	ВыборкаНоменклатура = Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

	Пока ВыборкаНоменклатура.Следующий() Цикл

		//6. зафиксируем значения вспомогательных переменных
		КоличествоОстаток = ВыборкаНоменклатура.КоличествоОстаток;
		СуммаОстаток = ВыборкаНоменклатура.СуммаОстаток;
		СписаноКоличество = 0;
		СписаноСумма = 0;

		//6. организуем цикл по детальным записям
```

```
            ВыборкаДетальныеЗаписи = ВыборкаНоменклатура.Выбрать();

            Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

                Движение = Движения.ОстаткиНоменклатуры.ДобавитьРасход();
                Движение.Период = Дата;
                Движение.Номенклатура = ВыборкаНоменклатура.Номенклатура;

                //6.1 записываем в движение количество списания
                КоличествоСписания = ВыборкаДетальныеЗаписи.Количество;
                Движение.Количество = КоличествоСписания;
                //6.2 увеличиваем счетчик списанного количества
                СписаноКоличество = СписаноКоличество+КоличествоСписания;

                //6.3 расчет себестоимости списания
                Себестоимость=?(СписаноКоличество = КоличествоОстаток,
                                        СуммаОстаток - СписаноСумма,
                                        СуммаОстаток*КоличествоСписания/КоличествоОстаток);

                Движение.Сумма = Себестоимость;

                //6.4 увеличиваем счетчик списанной суммы именно по данным движений!
                //Это важно, чтобы учесть ошибку округления при последнем списании.
                СписаноСумма = СписаноСумма + Движение.Сумма;
            КонецЦикла;

        КонецЦикла;

КонецПроцедуры
```

Разберем отличия алгоритма

Запрос получения данных (п. 3)

Первый запрос остался без изменений. Во втором запросе добавлены итоги по номенклатуре. В итогах подсчитаем общее количество списания по номенклатуре *СУММА(Количество)*. По полям из таблицы остатков регистра *КоличествоОстаток* и *СтоимостьОстаток* для итогов берем максимум. Их суммировать не нужно, т.к. это и есть остатки по номенклатуре.

Обход выборки из результата запроса (п.6, 6.1 – 6.4)

Получаем результат запроса и обходим выборку в цикле.

У нас будет два цикла:

- Внешний цикл – обход выборки по номенклатуре
- Вложенный цикл – обход детальных записей.

В рамках итерации внешнего цикла (обхода по номенклатуре) заводим переменные *СписаноКоличество* и *СписаноСумма*. Устанавливаем их значение в 0.

В цикле по детальным записям создаем движения для записи в регистр.

Списываемое количество берем из детальных записей. Не забываем увеличить счетчик списанного количества – значение переменной *СписаноКоличество*.

Списание будет являться последним, если *СписаноКоличество* будет равно *КоличествоОстаток*. При последнем списании не будем рассчитывать сумму списания по формуле, а возьмем значение *СуммыОстаток* из строки итогов за вычетом суммы, которую мы уже успели списать и накопили в переменной *СписаноСумма*.

Записываем найденную сумму списания в ресурс регистра.

Не забываем увеличить значение переменной *СписаноСумма*.

Важно: для увеличения переменной *СписаноСумма* прибавляем именно значение *Движение.Сумма*, а не значение переменной *Себестоимость*. Нужно помнить, что при записи в ресурс регистра происходит автоматическое округление, а в переменной *СписаноСумма* нужно накапливать именно суммы в том виде, как они были записаны в регистр. Только в этом случае при последнем списании мы сможем получить с помощью формулы *СуммаОстаток – СписаноСумма* значение с учетом ошибки округления и, таким образом, вывести суммовой остаток по регистру в ноль.

Проверим результат работы нашего алгоритма. Для этого перепроведем документ «Расходная накладная № 1»:

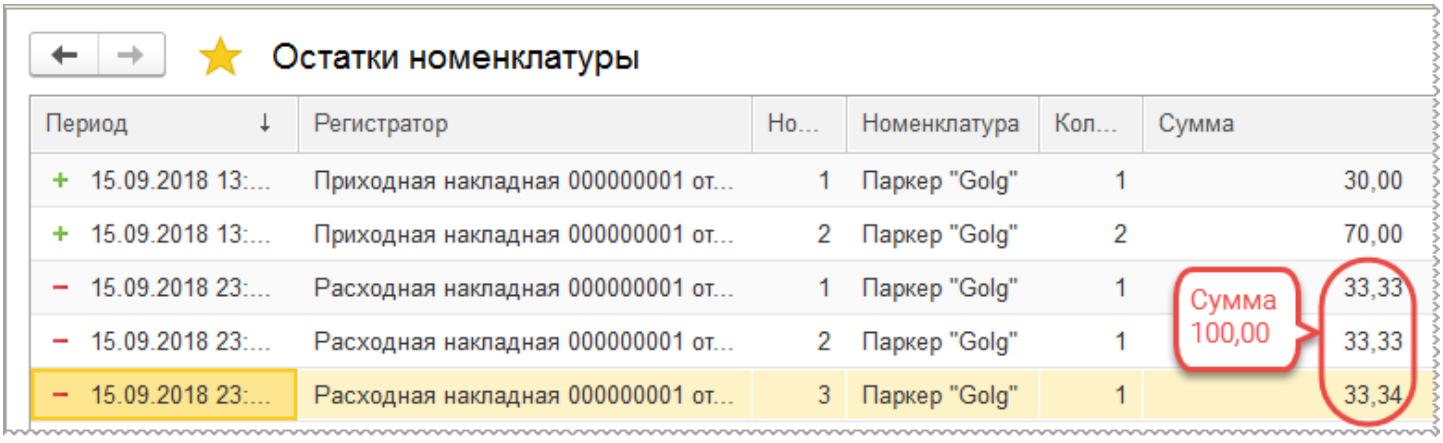


Рисунок 6 – Движения по РН «Остатки номенклатуры»

Видим, что теперь все в порядке. «Проблема копеек» решена.

Подведем итоги

Мы завершили разбор «проблемы копеек» и теперь знаем, что для ее решения и на экзамене, и в реальной практике нужно:

- Использовать **формулу с учетом последнего списания**
- Правильно определять условие, при котором списание действительно будет являться последним.

Также на практике убедились, что при проверке решения задачи совсем не лишними бывают тестовые примеры с дублями строк в табличной части документа.

14. Что нужно знать о партионном учете и методах учета себестоимости для успешной сдачи экзамена

Задача, которая рассматривается в данном блоке, довольно часто встречается:

- В реальной практике при работе с клиентами
- На экзамене «1С:Специалист» (в задачах № 1.1, 1.5 и других из сборника для подготовки к экзамену «1С: Специалист по платформе»)
- Как тестовое задание при собеседовании на должность 1С:Программист.

Изучив данный материал, вы узнаете:

- Как организовать партионный учет товара в разрабатываемой конфигурации
- Особенности получения и контроля остатков при оперативном проведении документов.

Подходы и методы учета товаров

Партионный учет представляет собой учет продукции, при котором во внимание принимается каждая партия. Партия – это «учетный» признак товара, который может отражать производство, поступление или отгрузку однородного товара с соблюдением следующих условий:

- Производство, поступление или отгрузка в одну и ту же дату
- Перевозка одним и тем же видом транспорта.

При учете себестоимости товаров существует два основных подхода: вести учет по партиям или по средней. Именно они встречаются в экзаменационных задачах.

Для ведения учета по партиям могут использоваться методы (FIFO и LIFO) или ручной учет, при котором явно указывается партия для списания.

FIFO (First in, First out) – первый вошел, первый вышел: партии списываются, начиная с самой ранней из имеющихся. Графически метод можно представить следующим образом:

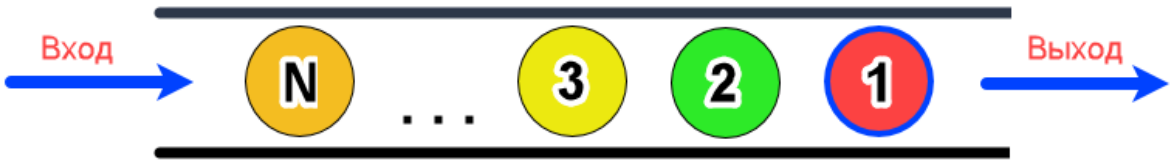


Рисунок 1 – Метод FIFO

LIFO (Last in, First out) – последний вошел, первый вышел: партии списываются, начиная с последней поступившей. Графически метод можно представить следующим образом:

Рисунок 2 – Метод LIFO

Если учет по партиям не ведется, то при списании товаров сумма списания рассчитывается по средней себестоимости (средневзвешенная оценка или среднескользящая оценка).

Средневзвешенная оценка предполагает нахождение средней стоимости за месяц исходя из количества и стоимости материалов на начало месяца и всех поступлений за месяц.

При среднескользящей оценке расчет производится на каждый момент выбытия товара. Соответственно, средняя скользящая стоимость считается исходя из количества и стоимости товара на начало месяца и его поступления до момента реализации.

Сравним списание товаров при использовании методов FIFO, LIFO и по средней стоимости. В рассматриваемом примере всего один момент выбытия товара, поэтому средневзвешенная и среднескользящая оценки будут совпадать.

В таблице отражена информация по движению товара *Холодильник СТИНОЛ 101:*

Дата	Вид движения	Документ	Количество	Цена (руб.)	Сумма (руб.)
02.04.2018	Приход	Приход 1	10	10 000	100 000
12.04.2018	Приход	Приход 2	5	11 000	55 000
16.04.2018	Приход	Приход 3	6	11 500	69 000
17.04.2018	Расход	Расход 1	12	x	x

В качестве партии используется документ «Приход». Графически данную таблицу в количественном выражении можно представить следующим образом:

Рисунок 3 – Движение товара «Холодильник СТИНОЛ 101»

Сумма поступления получается из документа «Приход».

Для документа «Расход» сумма будет рассчитываться согласно используемому на предприятии методу ведения партионного учета.

Рассмотрим порядок списания товаров для каждого метода.

LIFO

При использовании метода *LIFO* партии списываются начиная с последней, поэтому в приведенном выше примере полностью спишутся партии 3 и 2, частично спишется партия 1:

Рисунок 4 – Списание товара по методу LIFO

Таблица движений при использовании метода *LIFO* будет выглядеть следующим образом:

Дата	Вид движения	Документ	Партия	Количество	Цена (руб.)	Сумма (руб.)
02.04.2018	Приход	Приход 1	Приход 1	10	10 000	100 000
12.04.2018	Приход	Приход 2	Приход 2	5	11 000	55 000
16.04.2018	Приход	Приход 3	Приход 3	6	11 500	69 000
17.04.2018	Расход	Расход 1	Приход 3	6	11 500	69 000
17.04.2018	Расход	Расход 1	Приход 2	5	11 000	55 000
17.04.2018	Расход	Расход 1	Приход 1	1	10 000	10 000

FIFO

При использовании метода *FIFO* партии списываются начиная с самой первой, поэтому в приведенном выше примере полностью спишется партия 1 и частично спишется партия 2:

Рисунок 5 – Списание товара по методу FIFO

Таблица движений при использовании метода *FIFO* будет выглядеть следующим образом:

Дата	Вид движения	Документ	Партия	Количество	Цена (руб.)	Сумма (руб.)
02.04.2018	Приход	Приход 1	Приход 1	10	10 000	100 000
12.04.2018	Приход	Приход 2	Приход 2	5	11 000	55 000
16.04.2018	Приход	Приход 3	Приход 3	6	11 500	69 000
17.04.2018	Расход	Расход 1	Приход 1	10	10 000	100 000
17.04.2018	Расход	Расход 1	Приход 2	2	11 000	22 000

По средней

При использовании метода «*По средней*» партия не учитывается, стоимость списания рассчитывается по формуле:

Количество списания х Сумма остаток / Количество остаток

Таблица движений при использовании метода *По средней* будет выглядеть следующим образом:

Дата	Вид движения	Документ	Партия	Количество	Цена (руб.)	Сумма (руб.)
02.04.2018	Приход	Приход 1	-	10	10 000	100 000
12.04.2018	Приход	Приход 2	-	5	11 000	55 000
16.04.2018	Приход	Приход 3	-	6	11 500	69 000
17.04.2018	Расход	Расход 1	-	10	10 666,67	106 666,67

Сумма списания была рассчитана следующим образом:

10 * (100 000 + 55 000 + 69 000) / (10 + 5 + 6) = 106 666,67

Для сравнения сводные данные по сумме списания приведены в таблице ниже.

Метод	Количество списания	Сумма списания (руб.)
LIFO	10	113 000
FIFO	10	100 000
По средней	10	106 666,67

15. Что такое учетная политика и какие варианты настроек объектов метаданных могут встретиться в экзаменационных задачах

Учетная политика предприятия – это бухгалтерский термин, определение которому дано в законе о бухгалтерском учете. В целом учетная политика – это совокупность способов ведения экономическим субъектом бухгалтерского учета. Применительно к задачам экзамена понятие учетная политика гораздо уже – она используется для выбора метода списания себестоимости: FIFO, LIFO или по средней.

Выбор метода ведения партионного учета и хранение выбранного варианта – одна из часто встречающихся задач в блоках оперативного и бухгалтерского учета. Примеры таких задач: 1.1, 1.2, 2.16, 2.17 и другие.

Учетная политика предприятия может меняться с определенной периодичностью. В реальном учете изменения принимаются раз в год, в крайнем случае – с начала очередного квартала. В задачах сборника встречается периодичность год и даже день.

Данные об учетной политике вводятся обычно вручную, но в некоторых задачах может встретиться требование устанавливать учетную политику специальным документом.

Разберемся, каким образом организовать ввод и хранение информации об учетной политике при решении экзаменационной задачи.

Постановка задачи:

Компания занимается торговлей товарами. Списание себестоимости товаров должно быть организовано по партиям в зависимости от текущего значения, принятого на этот год в учетной политике метода списания себестоимости (FIFO, LIFO).

Реализуем настройки каркасной конфигурации для решения данной задачи.

В каркасной конфигурации имеется перечисление «УчетнаяПолитика» с вариантами ведения партионного учета, подходящими для решения задачи:

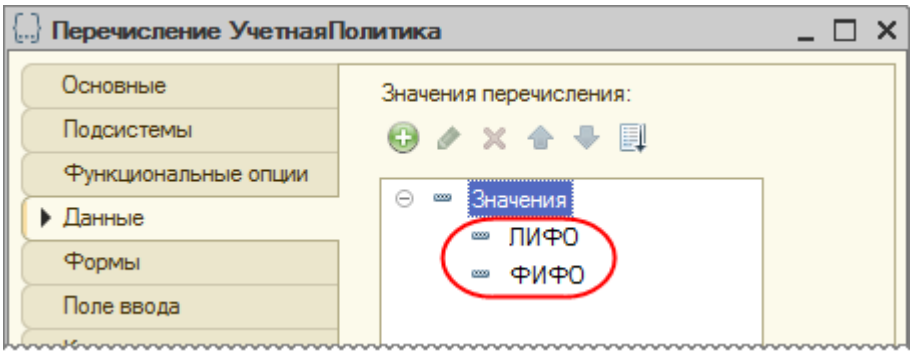


Рисунок 1 – Значения перечисления «УчетнаяПолитика»

В «1С:Предприятии 8» информация, привязанная к периоду, хранится в периодическом регистре сведений. Подходящего для целей решения задачи регистра сведений в каркасной конфигурации нет, поэтому создадим регистр сведений «УчетнаяПолитикаПредприятия» со свойствами:

- *Периодичность – В пределах года*
- *Режим записи – Независимый*, то есть значение метода списания будет устанавливаться в форме регистра сведений (дополнительный документ-регистратор не нужен, т.к. в задании такого требования нет):

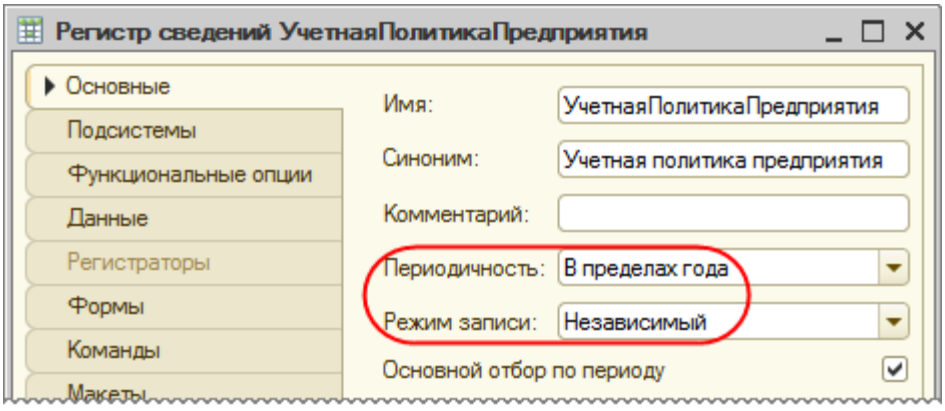


Рисунок 2 – Закладка *Основные* регистра сведений «УчетнаяПолитикаПредприятия»

В окне редактирования объекта конфигурации на закладке *Данные* создадим ресурс «МетодСписанияСебестоимости» (тип *ПеречислениеСсылка.УчетнаяПолитика*):

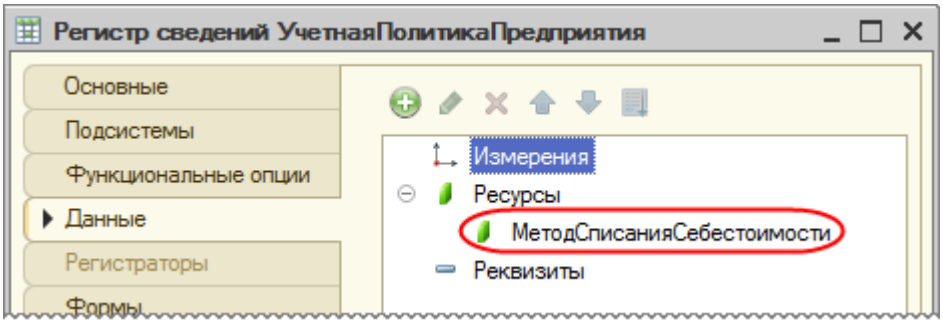


Рисунок 3 – Закладка *Данные* регистра сведений «УчетнаяПолитикаПредприятия»

Форму для регистра сведений создавать необязательно. В режиме «1С:Предприятие» после сохранения настроек появляется возможность ввода информации об учетной политике предприятия:

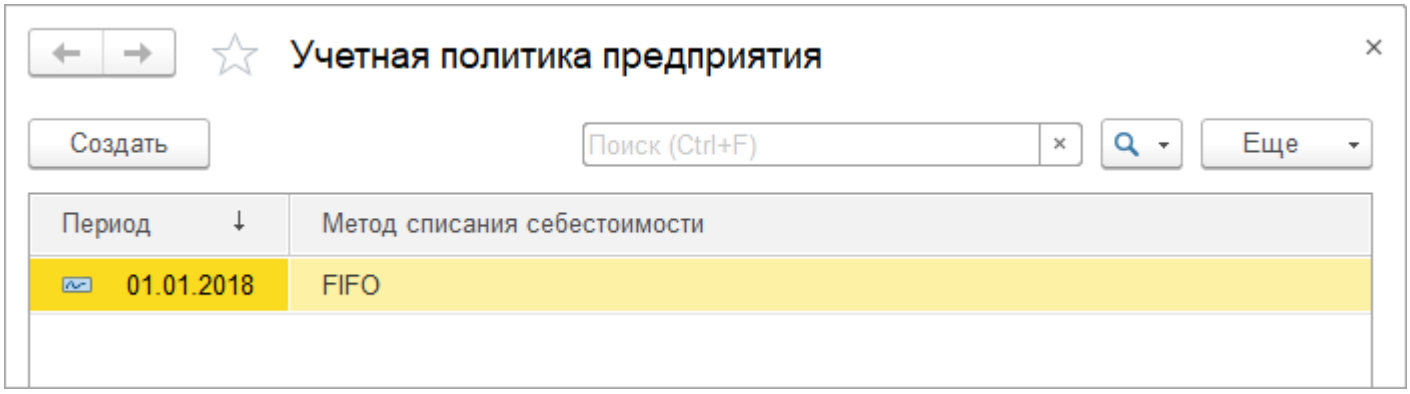


Рисунок 4 – Ввод информации в регистр сведений «Учетная политика предприятия» в режиме «1С:Предприятие»

В случае, когда условие задачи требует изменения учетной политики с помощью документа (например, в задаче 1.5 сборника), настройки немного сложнее.

Постановка задачи:

Компания занимается торговлей товарами. Списание себестоимости товаров должно быть организовано по партиям в зависимости от текущего значения, принятого в учетной политике метода списания себестоимости (FIFO, LIFO). Учетная политика может меняться каждый день, ее изменение фиксируется соответствующим документом.

Реализуем настройки каркасной конфигурации для решения данной задачи.

Создадим регистр сведений «УчетнаяПолитикаПредприятия» со свойствами:

- *Периодичность – В пределах дня*
- *Режим записи – Подчинение регистратору*.

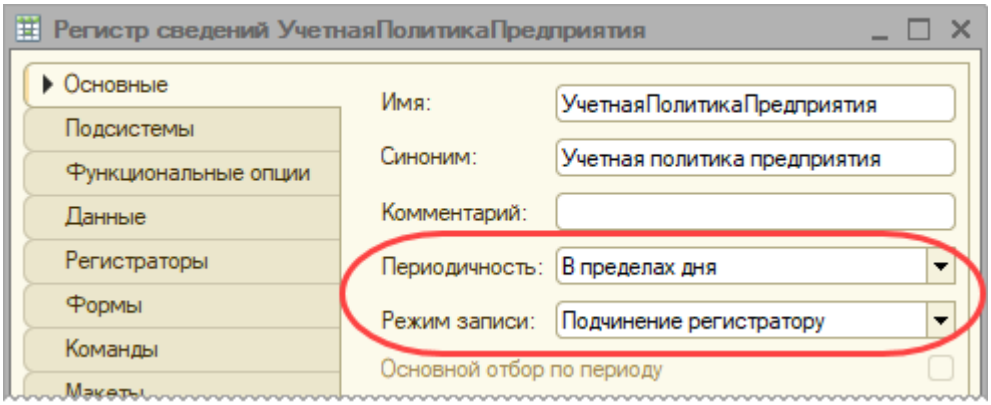


Рисунок 5 – Закладка *Основные* регистра сведений «УчетнаяПолитикаПредприятия»

В окне редактирования объекта конфигурации на закладке *Данные* создадим ресурс «МетодСписанияСебестоимости» (тип *ПеречислениеСсылка.УчетнаяПолитика*).

Добавим в конфигурацию документ «УстановкаУчетнойПолитики». На закладке *Данные* окна свойств документа добавим один реквизит «МетодСписанияСебестоимости» (тип *ПеречислениеСсылка.УчетнаяПолитика*):

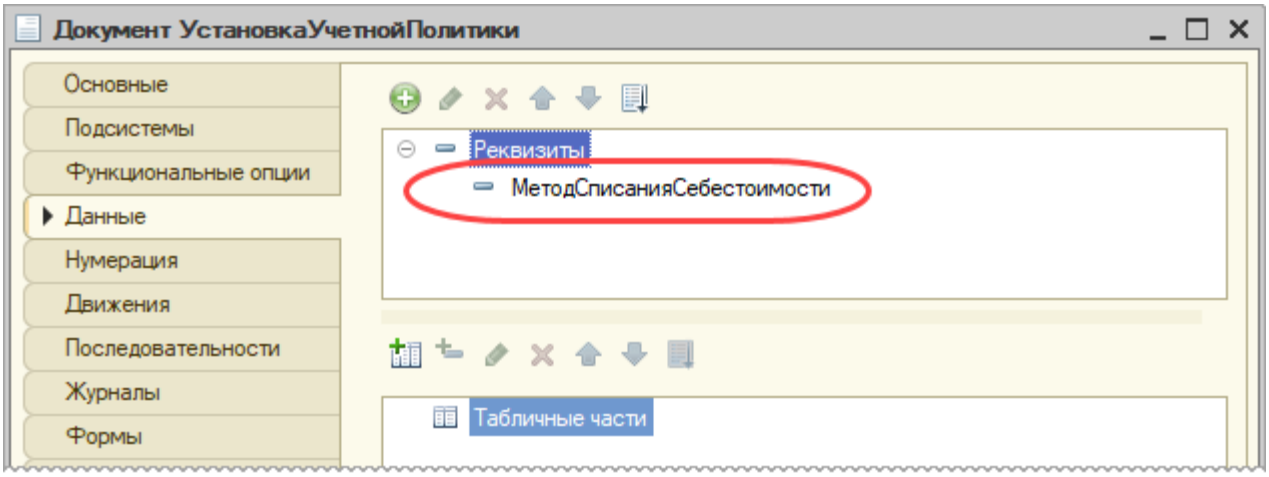


Рисунок 6 – Закладка *Данные* документа «УстановкаУчетнойПолитики»

На закладке Движения укажем, что документ является регистратором для регистра сведений «УчетнаяПолитикаПредприятия»:

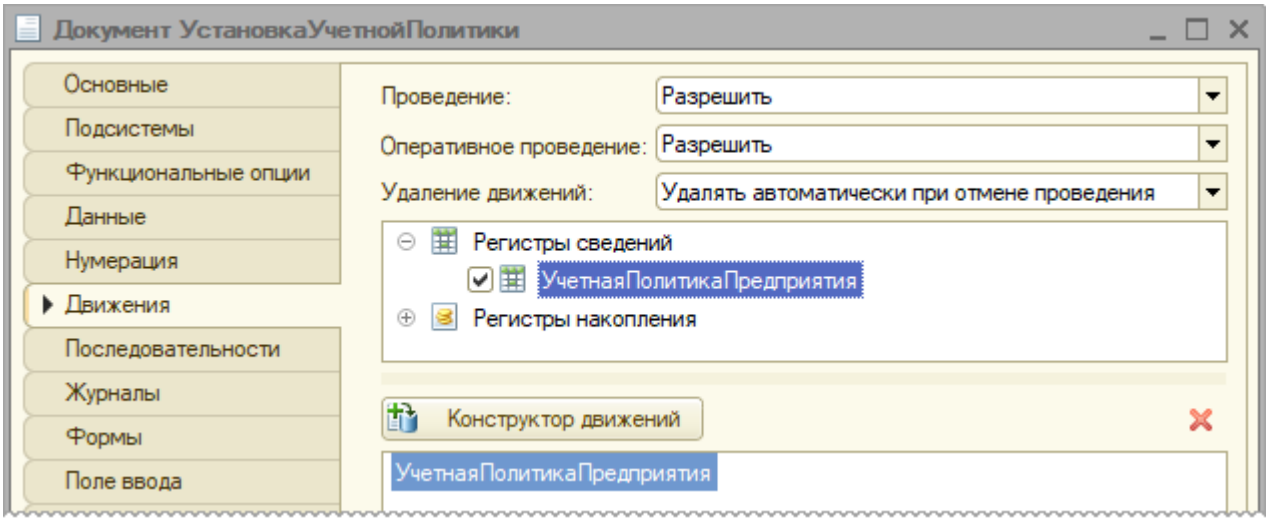


Рисунок 7 – Закладка *Движения* документа «УстановкаУчетнойПолитики»

С помощью конструктора движений создадим процедуру *ОбработкаПроведения* в модуле документа:

```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр УчетнаяПолитикаПредприятия
	Движения.УчетнаяПолитикаПредприятия.Записывать = Истина;
	Движение = Движения.УчетнаяПолитикаПредприятия.Добавить();
	Движение.Период = Дата;
	Движение.МетодСписанияСебестоимости = МетодСписанияСебестоимости;

	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Документ готов. Проверим его работу в режиме «1С:Предприятие».

Введем документ и посмотрим, какие записи он сделает в регистре сведений «Учетная политика предприятия»:

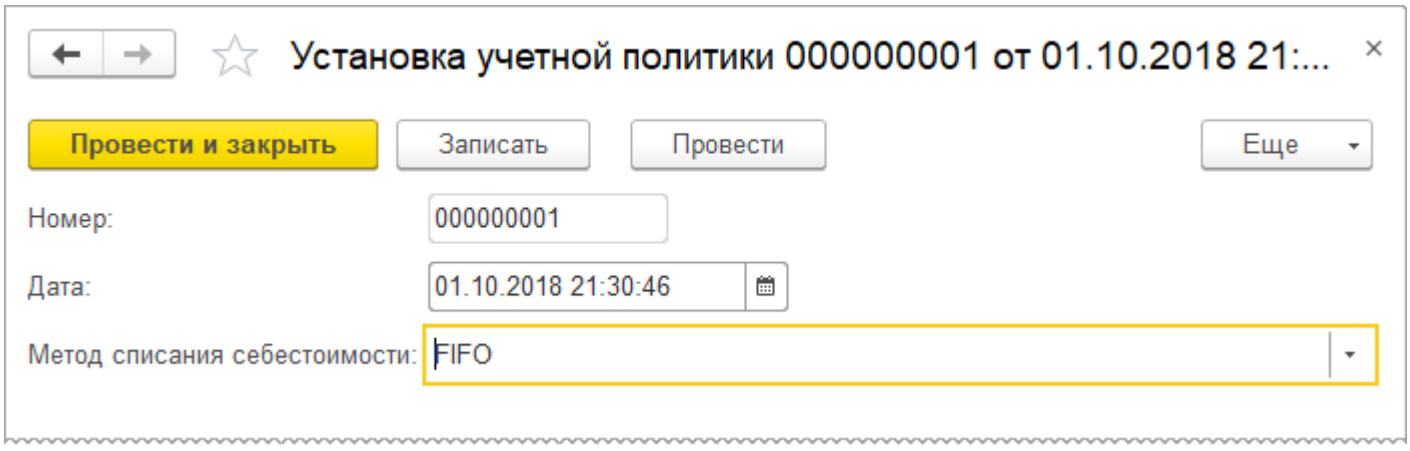


Рисунок 8 – Пример документа «Установка учетной политики» в режиме «1С:Предприятие»

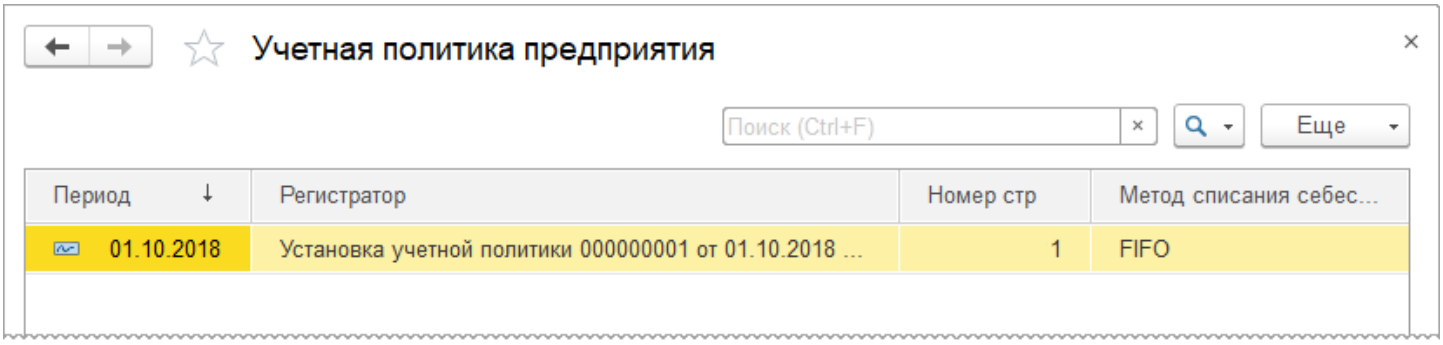


Рисунок 9 – Результаты проведения документа в режиме «1С:Предприятие»

Задача хранения учетной политики решена.

Перейдем к разбору практической задачи партионного учета.

16. Как правильно и максимально быстро организовать партионный учет товара в каркасной конфигурации

Постановка задачи:

Компания занимается торговлей холодильников. Поступление товаров отражается документом «Приходная накладная», продажа – «Расходная накладная». Списание себестоимости товаров должно быть организовано по партиям, в зависимости от текущего значения принятого на этот год в учетной политике метода списания себестоимости (FIFO, LIFO).

Для решения поставленной задачи потребуются следующие объекты конфигурации:

- Справочник «Номенклатура»
- Перечисление со значениями методов списания (*FIFO*, *LIFO*)
- Периодический регистр сведений для хранения значения учетной политики компании
- Регистр накопления «ОстаткиНоменклатуры»
- Регистр накопления «СебестоимостьТоваров»
- Документ «ПриходнаяНакладная»
- Документ «РасходнаяНакладная».

Для реализации решения необходимо выполнить следующие действия:

- Проверить, какие объекты уже имеются в каркасной конфигурации и могут быть использованы для решения поставленной задачи
- Создать недостающие объекты конфигурации
- Сформировать процедуру *ОбработкаПроведения* для документа «ПриходнаяНакладная» (с помощью конструктора движений)
- Сформировать процедуру ОбработкаПроведения для документа «РасходнаяНакладная»:
    - Выполнить контроль остатков по регистру «ОстаткиНоменклатуры»
    - Определить метод ведения партионного учета на дату документа
    - Получить себестоимость товаров в разрезе партий
    - Установить сортировку партий в зависимости от метода партионного учета
    - Рассчитать себестоимость и сформировать движения
- Проверить результаты в режиме «1С:Предприятие».

Используемые объекты каркасной конфигурации

Справочник «Номенклатура» в каркасной конфигурации уже имеется. Для решения задачи достаточно стандартных реквизитов этого справочника:

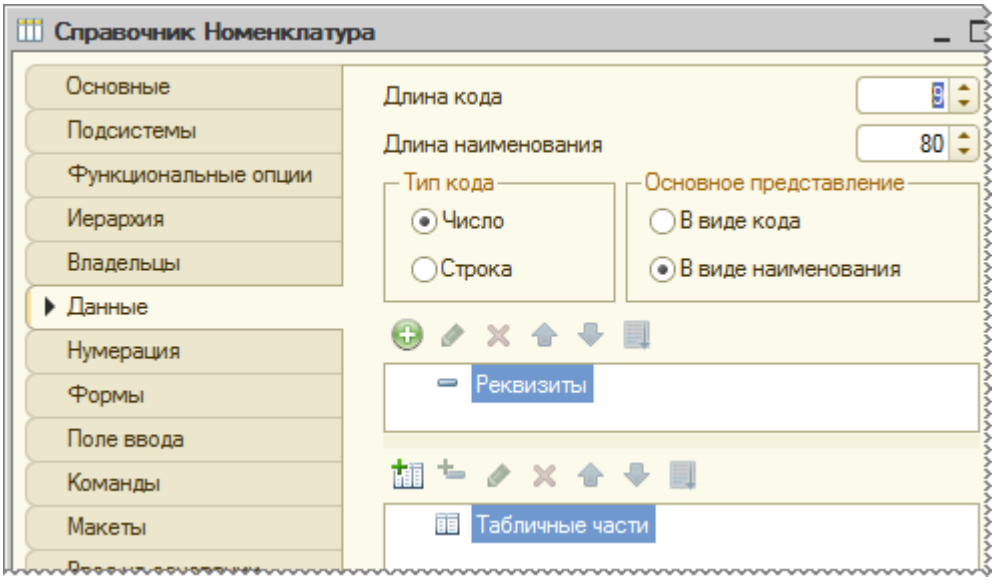


Рисунок 1 – Закладка *Данные* справочника «Номенклатура»

В каркасной конфигурации имеется перечисление «УчетнаяПолитика» с требуемыми значениями. Для реализации решения оно подойдет:

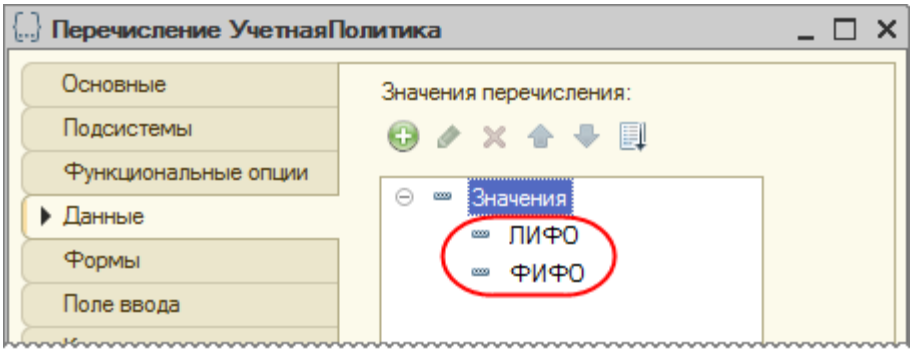


Рисунок 2 – Закладка *Данные* перечисления «УчетнаяПолитика»

Также в каркасной конфигурации уже имеется регистр накопления «ОстаткиНоменклатуры»:

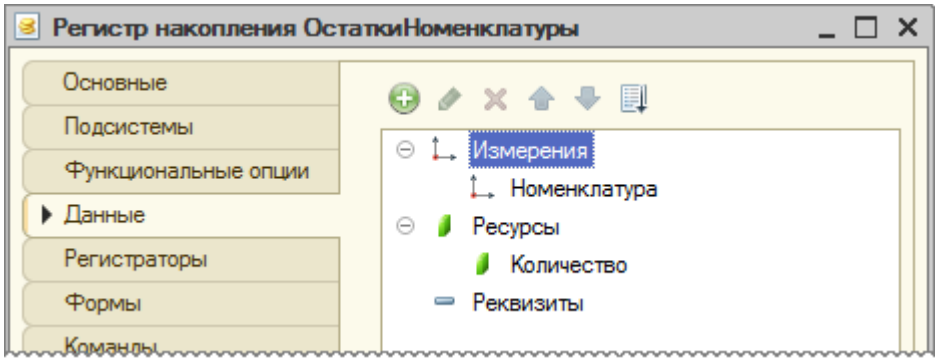


Рисунок 3 – Закладка *Данные* РН «ОстаткиНоменклатуры»

Структура регистра полностью подходит для решения поставленной задачи. Единственное, что нужно изменить в структуре регистра – это включить свойство «Запрет незаполненных значений» у измерения «Номенклатура». Включение данного свойства необходимо, так как движения с пустой номенклатурой будут считаться учетной ошибкой.

В каркасной конфигурации уже присутствуют документы: «ПриходнаяНакладная» и «РасходнаяНакладная»:

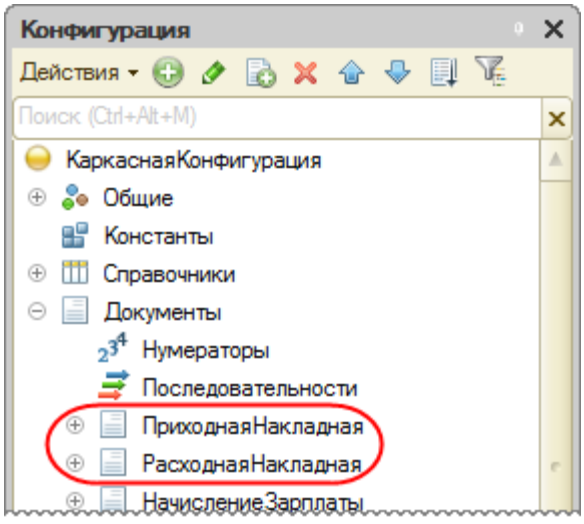


Рисунок 4 – Необходимые документы в дереве конфигурации

Структура документов и используемые типы данных удовлетворяют условию задачи:

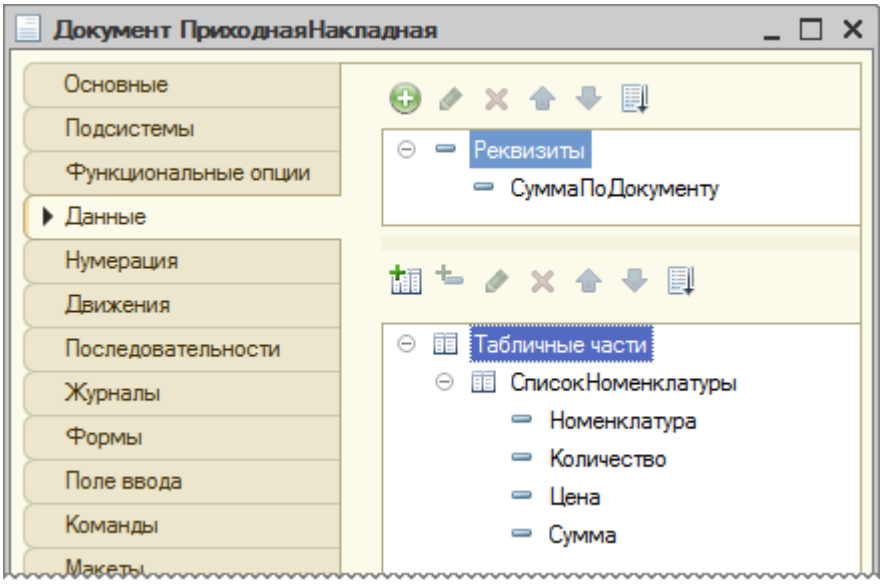


Рисунок 5 – Закладка *Данные* документа «ПриходнаяНакладная»

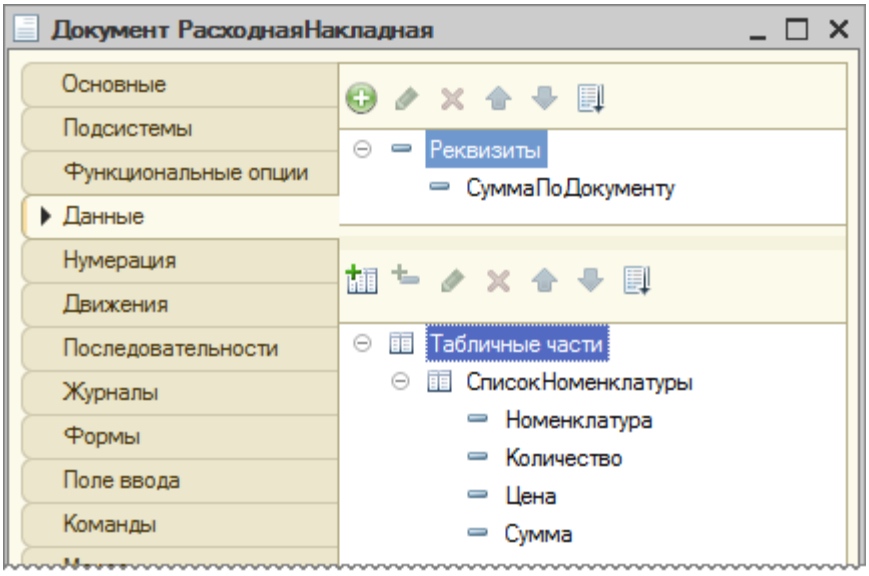


Рисунок 6 – Закладка *Данные* документа «РасходнаяНакладная»

Реквизит *СуммаПоДокументу (Число 12, 2)*.

Табличная часть *СписокНоменклатуры*:

- *Номенклатура (СправочникСсылка.Номенклатура)*
- *Количество (Число 10, 0)*
- *Цена (Число 10, 2)*
- *Сумма (Число 12, 2)*.

Для экономии времени создавать формы документов не будем.

Создание недостающих объектов конфигурации

Для хранения значения используемого метода партионного учета необходимо добавить регистр сведений *Учетная политика предприятия*. Как это сделать подробно рассмотрено в главе «**14. Что такое учетная политика и какие варианты настроек объектов метаданных могут встретиться в экзаменационных задачах**».

Для ведения учёта стоимости товаров в разрезе партий (партионного учета) создадим регистр накопления «СебестоимостьТоваров» с видом *Остатки*:

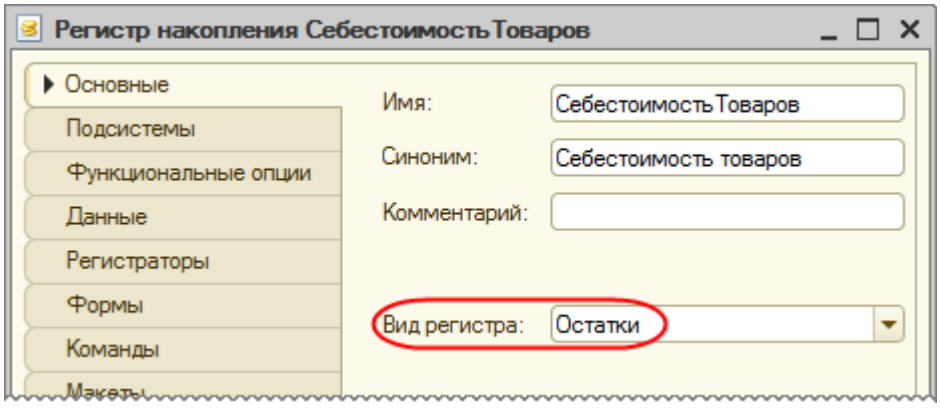


Рисунок 7 – Закладка *Основные* РН «СебестоимостьТоваров»

Измерения:

- *Партия (ДокументСсылка.ПриходнаяНакладная)*
- *Номенклатура (ссылка на справочник «Номенклатура»)*.

Ресурсы:

- *Количество (Число 10, 0)*
- *Сумма (Число 12, 2)*

Типы данных для ресурсов соответствуют типам, используемым в табличной части *СписокНоменклатуры* документов «Приходная накладная» и «Расходная накладная»:

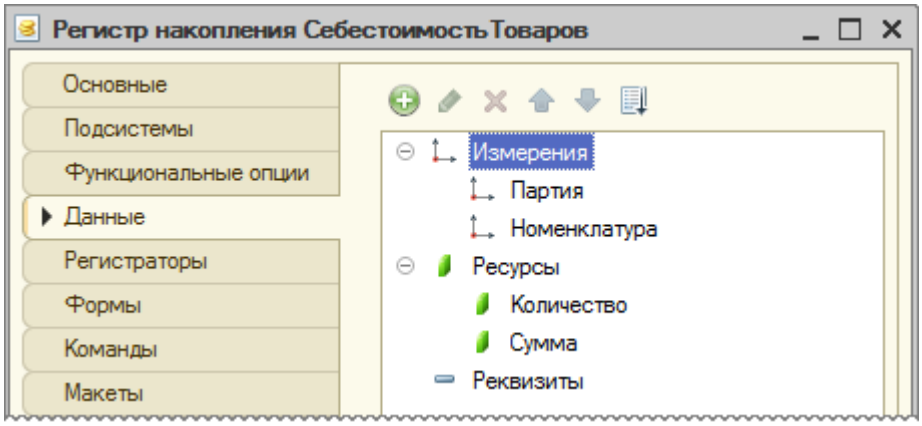


Рисунок 8 – Закладка *Данные* РН «СебестоимостьТоваров»

Регистраторы:

- «ПриходнаяНакладная»
- «РасходнаяНакладная»:

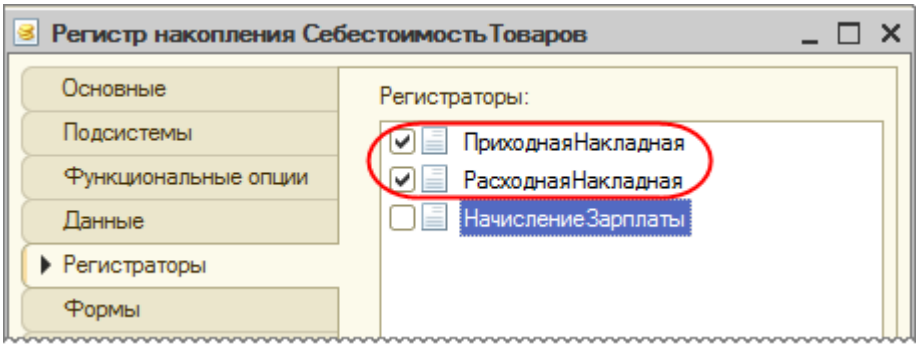


Рисунок 9 – Закладка *Регистраторы* РН «СебестоимостьТоваров»

Обработка проведения для документа «Приходная накладная»

Необходимо сформировать движения документа «Приходная накладная» по регистрам накопления «СебестоимостьТоваров» и «ОстаткиНоменклатуры». Как это сделать по регистру накопления «ОстаткиНоменклатуры» подробно рассмотрено в главе **«6. Как реализовать поступление товаров в компанию»**.

Добавим в конструкторе движений регистр накопления «СебестоимостьТоваров»:

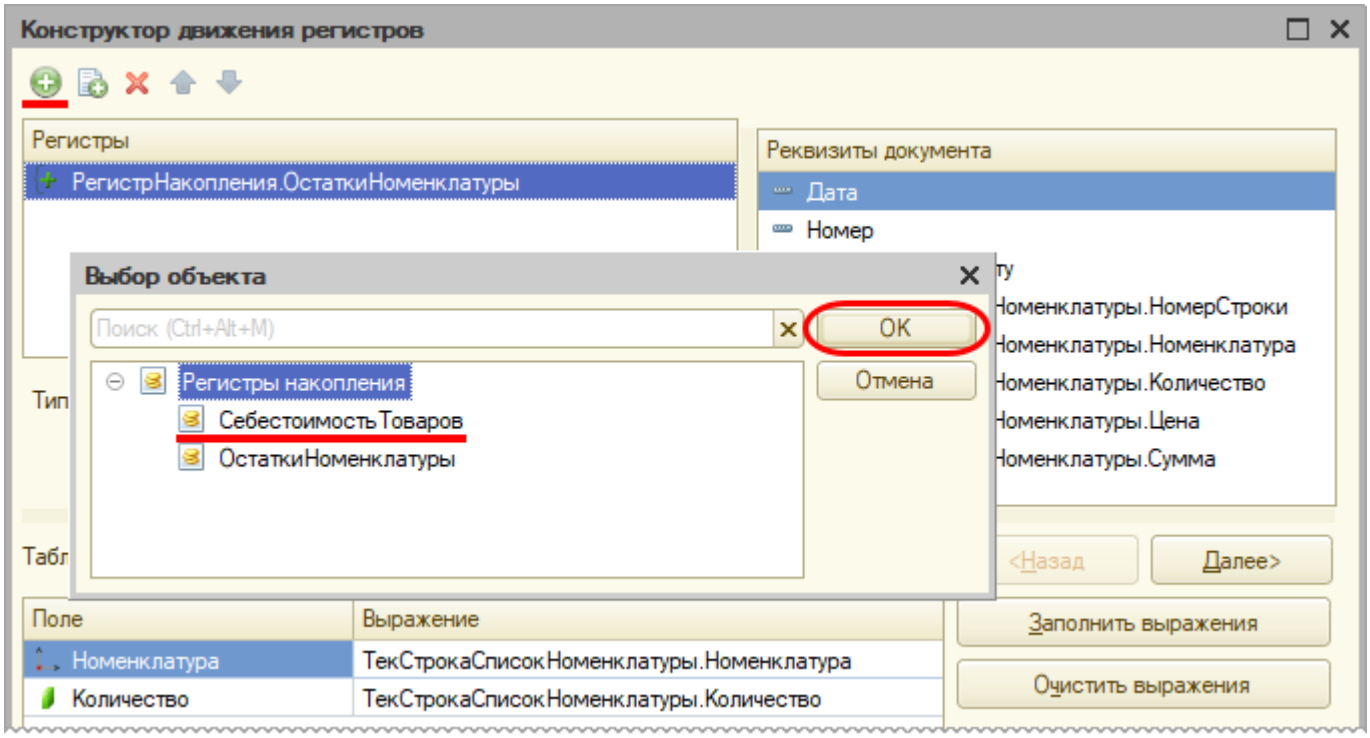


Рисунок 10 – Добавление движений по РН «СебестоимостьТоваров»

Для данного регистра Установим следующие настройки:

- *Тип движения регистра – Приход*
- *Табличная часть – СписокНоменклатуры*.

Заполним выражения для полей с помощью кнопки *Заполнить выражения:*

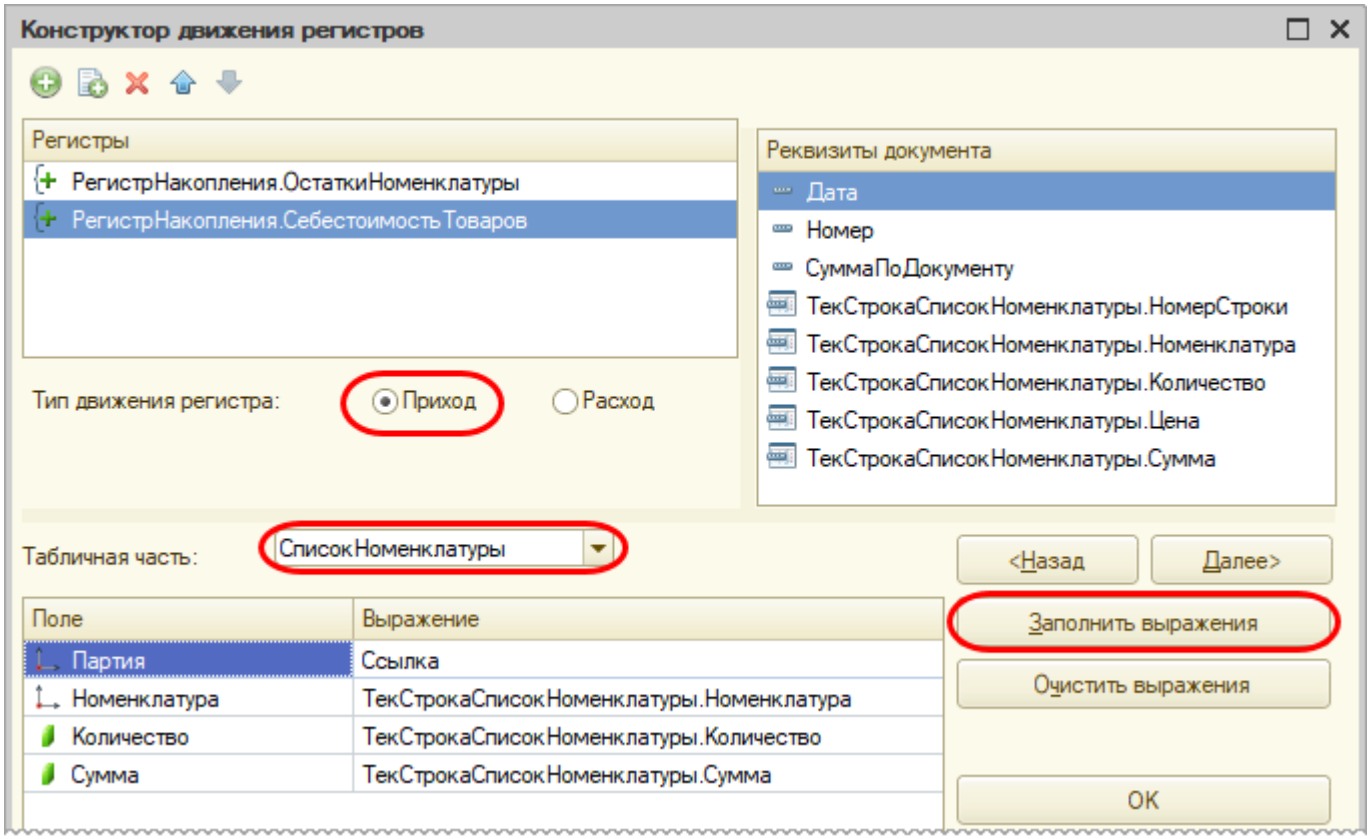


Рисунок 11 – Настройка конструктора движений регистров

Выражение для поля *Партия – Ссылка* придется заполнить вручную.

После нажатия кнопки *ОК* в модуле объекта появится процедура *ОбработкаПроведения* со следующим кодом:

```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр СебестоимостьТоваров Приход
	Движения.СебестоимостьТоваров.Записывать = Истина;
	Для Каждого ТекСтрокаСписокНоменклатуры Из СписокНоменклатуры Цикл
		Движение = Движения.СебестоимостьТоваров.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Партия = Ссылка;
		Движение.Номенклатура = ТекСтрокаСписокНоменклатуры.Номенклатура;
		Движение.Количество = ТекСтрокаСписокНоменклатуры.Количество;
		Движение.Сумма = ТекСтрокаСписокНоменклатуры.Сумма;
	КонецЦикла;

	// регистр ОстаткиНоменклатуры Приход
	Движения.ОстаткиНоменклатуры.Записывать = Истина;
	Для Каждого ТекСтрокаСписокНоменклатуры Из СписокНоменклатуры Цикл
```

```
            Движение = Движения.ОстаткиНоменклатуры.Добавить();
            Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
            Движение.Период = Дата;
            Движение.Номенклатура = ТекСтрокаСписокНоменклатуры.Номенклатура;
            Движение.Количество = ТекСтрокаСписокНоменклатуры.Количество;
        КонецЦикла;

        //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

17. Использование обработки проведения для документа «Расходная накладная»

Вопросы реализации контроля остатков при проведении документа «РасходнаяНакладная» подробно рассмотрены в главах, посвященных контролю остатков. В рамках решения данной задачи подзадачу контроля остатков по регистру «ОстаткиНоменклатуры» рассматривать не будем. При этом следует отметить, что необходимо использовать «новую» методику контроля остатков, так как все необходимые данные для формирования движений по регистру накопления «Остатки номенклатуры» присутствуют в самом документе (предварительное получение данных из базы не требуется).

Для того чтобы документ «РасходнаяНакладная» не проводился с пустой табличной частью «Товары», настроим проверку заполнения:

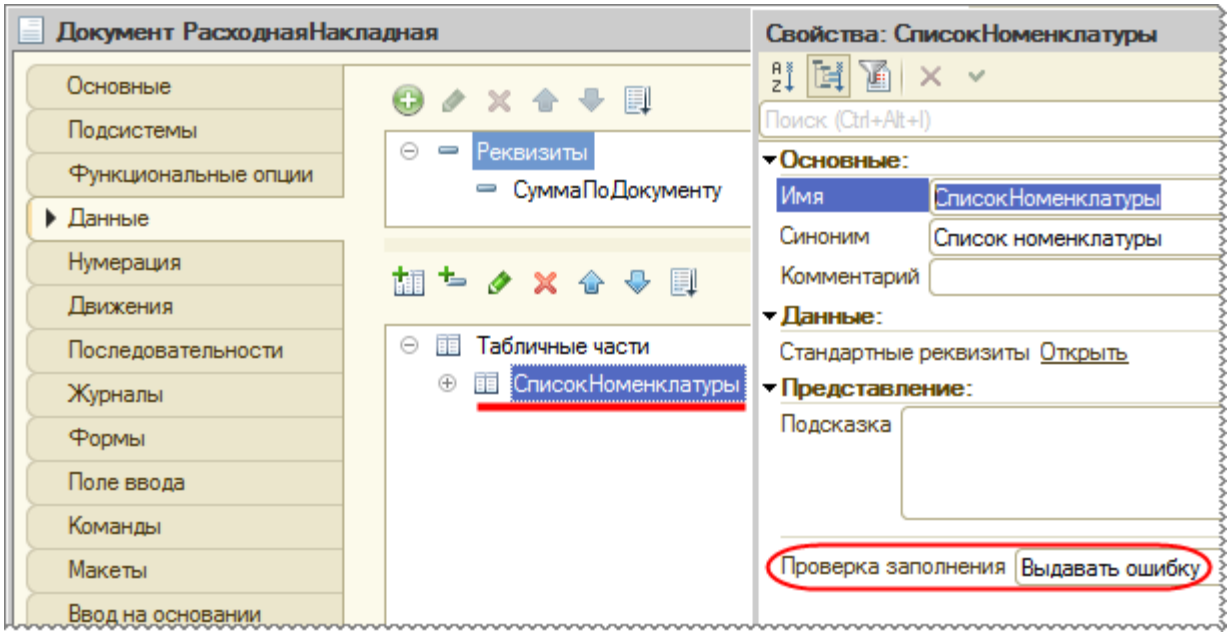


Рисунок 1 – Установка свойства «Проверка заполнения»

Движения документа «РасходнаяНакладная» по регистру накопления «СебестоимостьТоваров» сформируем вручную, так как простого обхода строк в коде, сформированного с помощью конструктора движений для табличной части *СписокНоменклатуры*, будет недостаточно, требуется получение данных о себестоимости товаров из базы с помощью запроса.

Код обработки проведения будет выглядеть следующим образом:

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

	// 1. Подготовка наборов записей регистра
	Движения.СебестоимостьТоваров.Записывать = Истина;
	Движения.Записать();

	// 2. Восстановление для свойства набора движений Записывать значения Истина
	Движения.СебестоимостьТоваров.Записывать = Истина;

	// 3. Инициализация менеджера временных таблиц
	Запрос = Новый Запрос;
	Запрос.МенеджерВременныхТаблиц = Новый МенеджерВременныхТаблиц;

	// 4. Пакет запросов,в котором группируются данные табличной части
	// и получается метод списания себестоимости
	Запрос.Текст =
		"ВЫБРАТЬ
		|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура КАК Номенклатура,
		|	СУММА(РасходнаяНакладнаяСписокНоменклатуры.Количество) КАК Количество,
		|	СУММА(РасходнаяНакладнаяСписокНоменклатуры.Сумма) КАК Сумма
		|ПОМЕСТИТЬ ТЧСписокНоменклатуры
		|ИЗ
		|	Документ.РасходнаяНакладная.СписокНоменклатуры КАК
РасходнаяНакладнаяСписокНоменклатуры
		|ГДЕ
		|	РасходнаяНакладнаяСписокНоменклатуры.Ссылка = &Ссылка
		|
		|СГРУППИРОВАТЬ ПО
		|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура
		|
		|ИНДЕКСИРОВАТЬ ПО
		|	Номенклатура
		|;
		|
		|////////////////////////////////////////////////////////////////////////////////
		|ВЫБРАТЬ
		|	УчетнаяПолитикаПредприятияСрезПоследних.МетодСписанияСебестоимости
		|ИЗ
		|	РегистрСведений.УчетнаяПолитикаПредприятия.СрезПоследних(&МоментВремени, ) КАК
УчетнаяПолитикаПредприятияСрезПоследних
		|;
		|
		|////////////////////////////////////////////////////////////////////////////////
		|ВЫБРАТЬ
		|	ТЧСписокНоменклатуры.Номенклатура
		|ИЗ
		|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры";

	// 5. Определение момента времени для получения остатков
```

```
Запрос.УстановитьПараметр("МоментВремени",  МоментВремени());
Запрос.УстановитьПараметр("Ссылка", Ссылка);

Результаты = Запрос.ВыполнитьПакет();

РезультатМетодСписания = Результаты.Получить(1);

// 6. Проверка наличия учетной политики на момент проведения документа
Если РезультатМетодСписания.Пустой() Тогда
	Отказ = Истина;
	Сообщение = Новый СообщениеПользователю;
	Сообщение.Текст = "На дату документа нет учетной политики. Провести документ невозможно!";
	Сообщение.Сообщить();
	Возврат;
КонецЕсли;

ВыборкаМетодСписания = РезультатМетодСписания.Выбрать();
ВыборкаМетодСписания.Следующий();

МетодСписанияСебестоимости = ВыборкаМетодСписания.МетодСписанияСебестоимости;

// 7. Управляемая блокировка данных регистра
РезультатНоменклатура = Результаты.Получить(2);

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

// 8. Запрос получающий, себестоимость списания номенклатуры
Запрос.Текст = "ВЫБРАТЬ
	|	ТЧСписокНоменклатуры.Номенклатура КАК Номенклатура,
	|	ТЧСписокНоменклатуры.Количество КАК Количество,
	|	ПРЕДСТАВЛЕНИЕ(ТЧСписокНоменклатуры.Номенклатура) КАК
	НоменклатураПредставление,
	|	СебестоимостьТоваровОстатки.Партия,
	|	ЕСТЬNULL(СебестоимостьТоваровОстатки.КоличествоОстаток, 0) КАК КоличествоОстаток,
	|	ЕСТЬNULL(СебестоимостьТоваровОстатки.СуммаОстаток, 0) КАК СуммаОстаток
	|ИЗ
	|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.СебестоимостьТоваров.Остатки(
	|				&МоментВремени,
	|				Номенклатура В
	|					(ВЫБРАТЬ
	|						ТЧСписокНоменклатуры.Номенклатура
	|					ИЗ
	|						ТЧСписокНоменклатуры КАК
	ТЧСписокНоменклатуры)) КАК СебестоимостьТоваровОстатки
```

```
        |                ПО (ТЧСписокНоменклатуры.Номенклатура =
        СебестоимостьТоваровОстатки.Номенклатура)
        |
        |УПОРЯДОЧИТЬ ПО
        |        СебестоимостьТоваровОстатки.Партия.МоментВремени
        |ИТОГИ
        |        МАКСИМУМ(Количество),

        |        СУММА(КоличествоОстаток)
        |ПО
        |        Номенклатура";

// 9. Корректировка сортировки результата запроса в зависимости от метода списания себестоимости
Если МетодСписанияСебестоимости = Перечисления.УчетнаяПолитика.ЛИФО Тогда
        Запрос.Текст = СтрЗаменить(Запрос.Текст,
        "СебестоимостьТоваровОстатки.Партия.МоментВремени",
        "СебестоимостьТоваровОстатки.Партия.МоментВремени УБЫВ");
КонецЕсли;

Результат = Запрос.Выполнить();

// 10. Цикл по номенклатуре документа
ВыборкаНоменклатура = Результат.Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

Пока ВыборкаНоменклатура.Следующий() Цикл

        // 11. Контроль остатков
        Если ВыборкаНоменклатура.Количество>ВыборкаНоменклатура.КоличествоОстаток Тогда

                Нехватает =
ВыборкаНоменклатура.Количество-ВыборкаНоменклатура.КоличествоОстаток;

                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст = "Не хватает остатка по номенклатуре " +
ВыборкаНоменклатура.НоменклатураПредставление
                + ", не хватает: " +  Нехватает;
                Сообщение.Сообщить();

                Отказ = Истина;

        КонецЕсли;

        // 12. Остатка не хватает, нет смысла формировать движения
        Если Отказ Тогда
                Продолжить;
        КонецЕсли;

        // 13. Получим количество для списания
        КоличествоСписать = ВыборкаНоменклатура.Количество;

        // 14. Цикл по партиям
        ВыборкаДетальныеЗаписи = ВыборкаНоменклатура.Выбрать();
```

```
            Пока ВыборкаДетальныеЗаписи.Следующий() И КоличествоСписать <> 0 Цикл
                // 15. Расчет количества для списания
                Количество = МИН(КоличествоСписать, ВыборкаДетальныеЗаписи.КоличествоОстаток);
                // 16. Расчет суммы для списания
                Если Количество = ВыборкаДетальныеЗаписи.КоличествоОстаток Тогда
                    Себестоимость = ВыборкаДетальныеЗаписи.СуммаОстаток;
                Иначе
                    Себестоимость =
Количество*ВыборкаДетальныеЗаписи.СуммаОстаток/ВыборкаДетальныеЗаписи.КоличествоОстаток;
                КонецЕсли;
                // 17. Уменьшим количество для списания
                КоличествоСписать = КоличествоСписать - Количество;

                // 18. Сформируем движение
                Движение = Движения.СебестоимостьТоваров.ДобавитьРасход();
                Движение.Период = Дата;
                Движение.Номенклатура = ВыборкаДетальныеЗаписи.Номенклатура;
                Движение.Партия = ВыборкаДетальныеЗаписи.Партия;
                Движение.Количество = Количество;
                Движение.Сумма = Себестоимость;
            КонецЦикла;

        КонецЦикла;

        КонецПроцедуры
```

Рассмотрим ключевые точки алгоритма.

Подготовка наборов записей регистра (п.1)

Для документа «РасходнаяНакладная» установлен режим удаления движений – «Удалять автоматически при отмене проведения»:

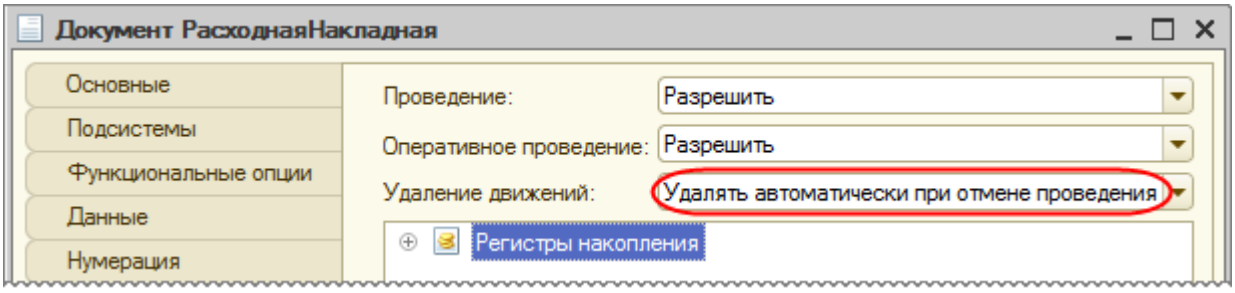


Рисунок 2 – Свойство «Удаление движений»

При такой настройке в случае перепроведения документа движения автоматически удаляться не будут, при обращении к регистру накопления «СебестоимостьТоваров» могут учитываться старые движения самого документа. Это происходит, когда дата документа при перепроведении сдвигается вперед.

Для очистки старых движений документа используем принудительную запись пустого набора. В этом случае при чтении данных из регистра старые движения документа в расчет виртуальных таблиц не попадут.

Важно:

1. Использование режима удаления движений «Удалять автоматически» на экзамене считается ошибкой, так как в этом случае платформа записывает пустые наборы записей в регистры автоматически. Причем это происходит еще до начала выполнения процедуры *ОбработкаПроведения* и приводит к преждевременным блокировкам

2. Ситуация с учетом старых движений документа может возникнуть только в случае сдвига даты документа вперед. Поэтому при решении реальных задач очистку движений рекомендуется делать только в этом случае

3. Движения можно записать, используя метод *Движения.СебестоимостьТоваров.Записать()*. Но более универсальным является метод *Движения.Записать()*. Этот метод записывает сразу все наборы движений по регистрам, для которых свойство *Записывать* установлено в значение *Истина*. После выполнения метода *Движения.Записать()* для всех наборов движений свойство *Записывать* примет значение *Ложь*. При этом для наборов движений, которые нужно записать в конце обработки проведения, требуется заново установить значение *Истина* для свойства *Записывать*.

Восстановление для свойства «Записывать» набора движений значения «Истина» (п.2)

Установим для свойства *Записывать* набора движений по регистру «СебестоимостьТоваров» значение *Истина*, чтобы сформированные движения записались в конце алгоритма проведения.

Важно. В конце обработки проведения система автоматически записывает наборы движений, для которых свойство *Записывать* установлено в значение *Истина*. В конце процедуры использовать метод *Движения.Записать()* не требуется.

Инициализация менеджера временных таблиц (п.3)

Менеджер временных таблиц необходим для того, чтобы обеспечить доступность временной таблицы, созданной в запросе, для использования в очередных запросах. Таким образом, данные табличной части получаются один раз, сохраняются во временную таблицу и далее используются многократно.

Пакет запросов (для получения метода списания себестоимости, п.4)

Сформируем пакет запросов для получения временной таблицы *ТЧСписокНоменклатуры*, получения метода списания себестоимости и списка номенклатуры.

Рассмотрим процесс создания пакета запросов с помощью конструктора запроса подробно.

В первом запросе в качестве источника данных выберем табличную часть *СписокНоменклатуры* документа «Расходная накладная».

Выберем поля:

- *РасходнаяНакладнаяСписокНоменклатуры.Номенклатура*
- *РасходнаяНакладнаяСписокНоменклатуры.Количество*
- *РасходнаяНакладнаяСписокНоменклатуры.Сумма*:

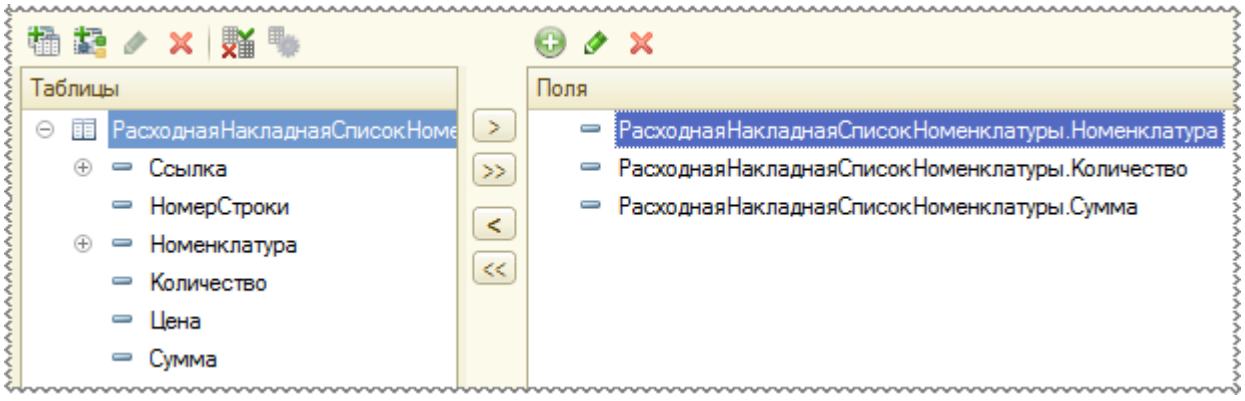


Рисунок 3 – Закладка *Таблицы и поля* конструктора запросов

Поскольку в табличной части *СписокНоменклатуры* может быть несколько строк с одинаковой номенклатурой, сгруппируем данные по полю *РасходнаяНакладнаяСписокНоменклатуры.Номенклатура*.

Для ресурсов *РасходнаяНакладнаяСписокНоменклатуры.Количество* и *РасходнаяНакладнаяСписокНоменклатуры.Сумма* используем агрегатную функцию *Сумма*:.

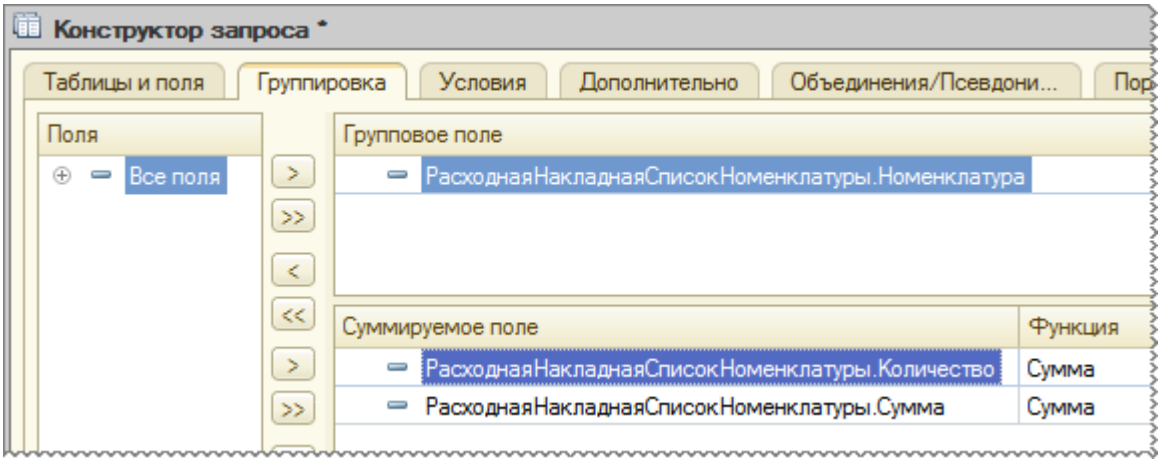


Рисунок 4 – Закладка *Группировка* конструктора запросов

Добавим условие отбора, требуются данные табличной части *СписокНоменклатуры* только текущего документа:

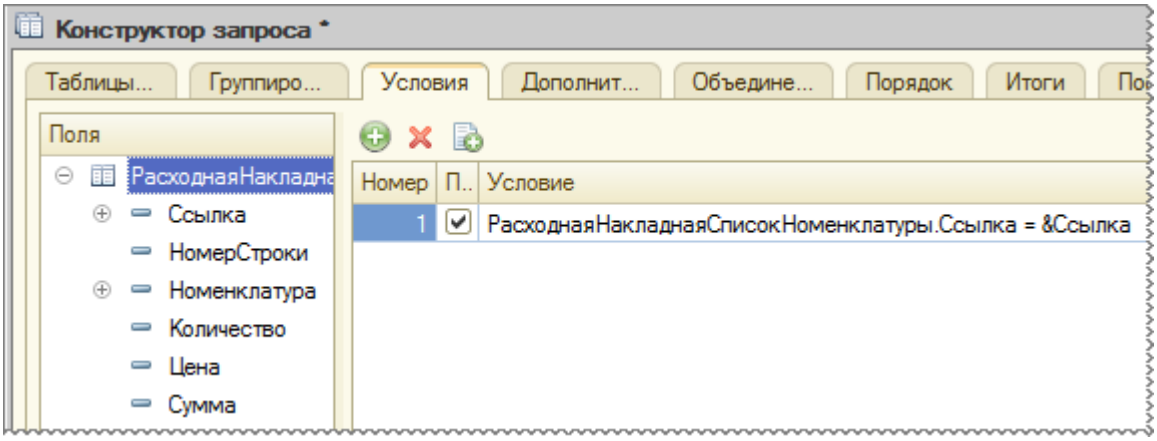


Рисунок 5 – Закладка *Условия* конструктора запросов

Создадим временную таблицу *ТЧСписокНоменклатуры*, в которую будем помещать результаты запроса:

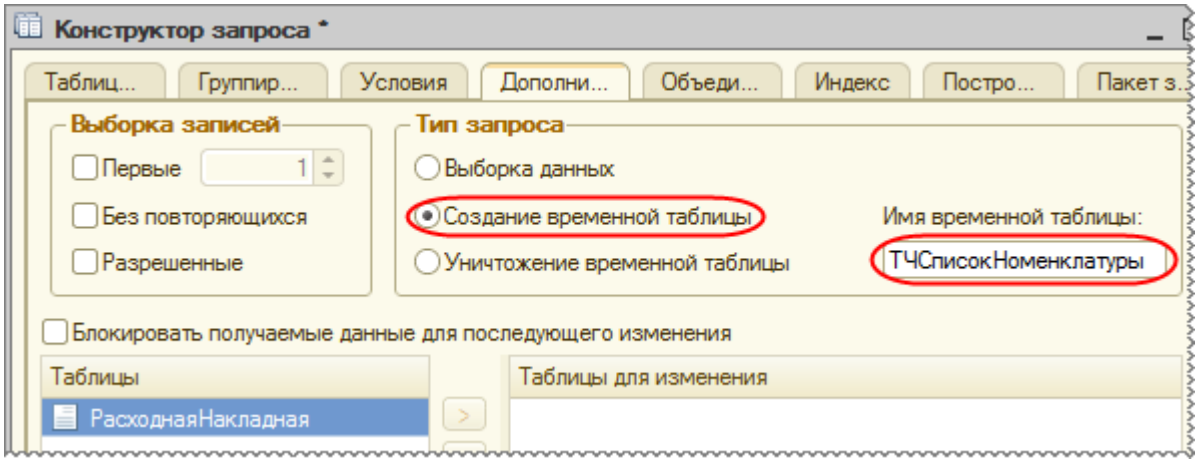


Рисунок 6 – Закладка *Дополнительно* конструктора запросов

В дальнейшем будем использовать временную таблицу *ТЧСписокНоменклатуры* в других запросах и выполнять соединение по полю *Номенклатура*. Поэтому добавим по данному полю индекс, который используется для повышения производительности соединения таблиц:

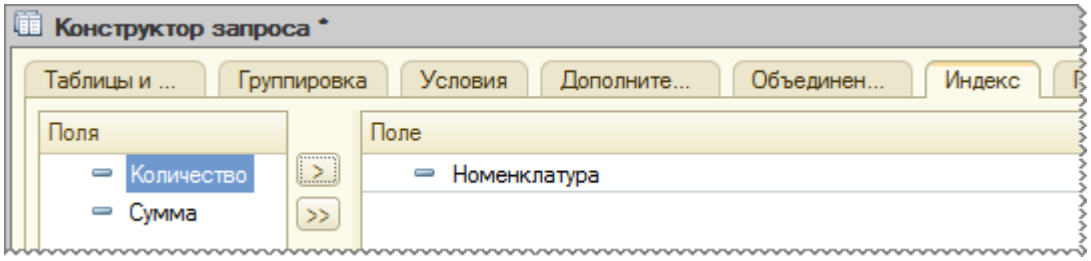


Рисунок 7 – Закладка *Индекс* конструктора запросов

Метод списания себестоимости получим с помощью запроса из регистра сведений «УчетнаяПолитикаПредприятия». Для этого добавим в пакет еще один запрос:

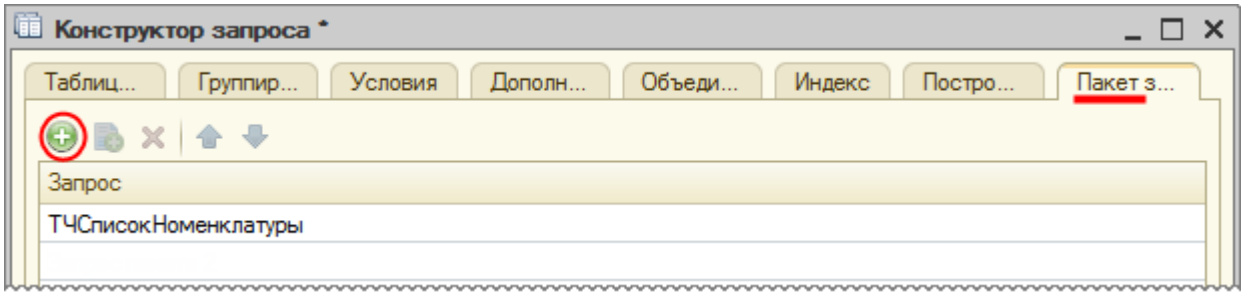


Рисунок 8 – Закладка *Пакет запросов* конструктора запросов

В качестве источника данных выберем виртуальную таблицу регистра сведений *УчетнаяПолитикаПредприятия.СрезПоследних*. Данные будем получать на момент времени документа:

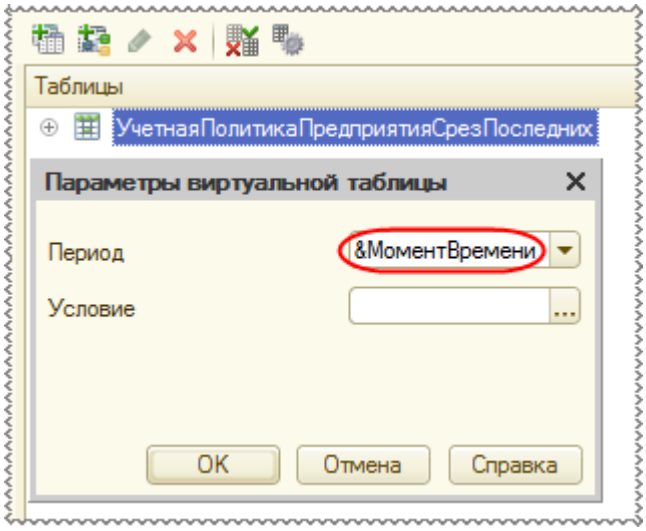


Рисунок 9 – Установка параметров виртуальной таблицы

В список полей перенесем поле *МетодСписанияСебестоимости*:

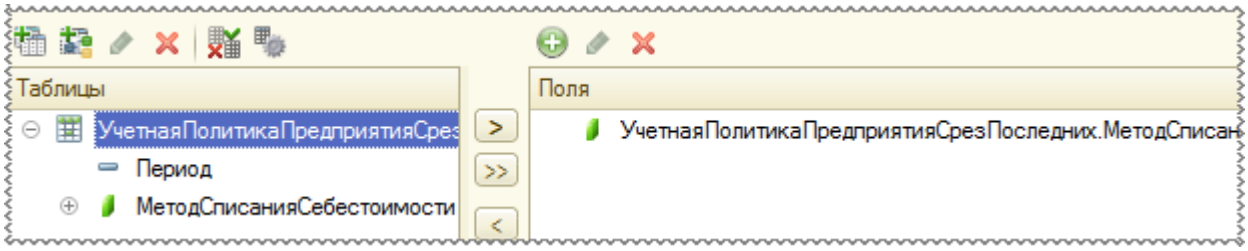


Рисунок 10 – Закладка *Таблицы и поля* конструктора запросов

Добавим в пакет еще один запрос:

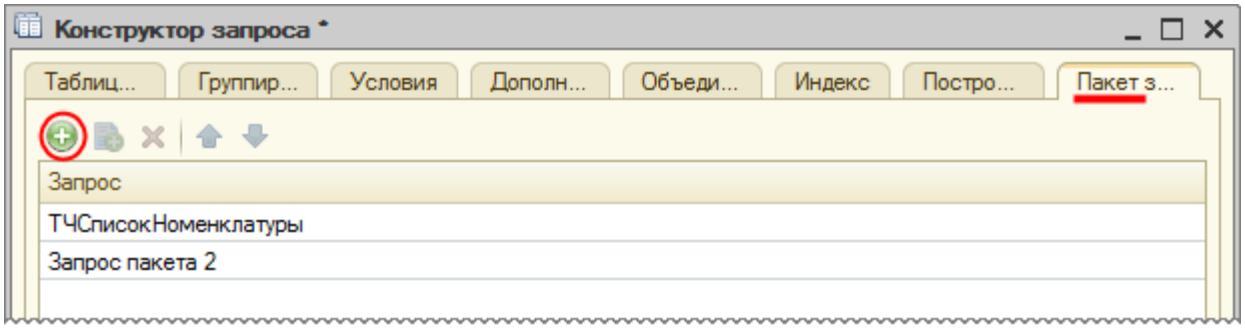


Рисунок 11 – Закладка *Пакет запросов* конструктора запросов

Выберем из временной таблицы *ТЧСписокНоменклатуры* поле *Номенклатура:*

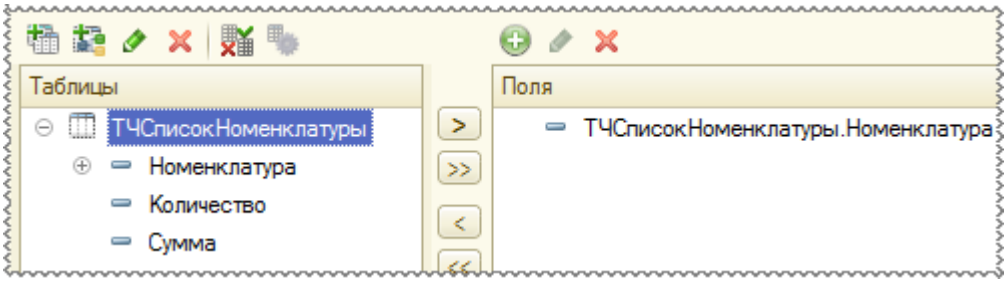


Рисунок 12 – Закладка *Таблицы и поля* конструктора запросов

Определение момента времени для получения остатков (п.5)

Установим значение параметра запроса *МоментВремени*. Будем передавать момент времени документа.

Момент времени подробно рассмотрен в главе **«3. Что такое момент времени и для чего он используется»**.

Важно. Получение остатков на дату документа будет считаться грубейшей ошибкой на экзамене. Также на экзамене нельзя использовать актуальные остатки (в качестве параметра *МоментВремени* передается *Неопределенно*) и конструкцию:

```
МоментВременииОстатков = ?(РежимПроведения = РежимПроведенияДокумента.Оперативный, Неопределено, МоментВремени());

Запрос.УстановитьПараметр("МоментВремени", МоментВременииОстатков)
```

На экзамене это считается ошибкой и из оценки вычитается 1 балл:

В задачах получения данных из информационной базы установка отборов по неиндексированным полям	0,5
Если при проведении документа используются каким-то образом данные, считываемые из регистров, обязательно требуется предусмотреть получение таких данных на момент проведения документа	1,0
Конфигурация должна устоичиво раоотать не только при движении вперед, но и назад.	1,0

Рисунок 17 – Фрагмент из списка часто встречающихся ошибок

Какие проблемы могут быть при использовании актуальных остатков, рассмотрим в отдельной главе.

Проверка наличия учетной политики на момент проведения документа (п.6)

Если результат запроса с методом списания себестоимости пустой, то это означает, что в информационной базе не определена учетная политика. В этом случае выходим из процедуры, поскольку нет смысла продолжать проведение.

Важно. При выводе сообщений пользователю согласно требованиям фирмы 1С к разработке конфигураций используется объект *СообщениеПользователю*.

Управляемая блокировка данных регистра (п.7)

Перед чтением данных установим управляемую блокировку. Блокировать будем регистр накопления *СебестоимостьТоваров* по измерению *Номенклатура* (работа с управляемыми блокировками подробно описана в главе **«11. Какие навыки в использовании управляемых блокировок потребуются на экзамене»**).

Запрос для получения себестоимости списания номенклатуры (п.8)

Сформируем новый запрос. Рассмотрим процесс создания запроса с помощью конструктора запроса подробно.

В качестве источника данных для запроса опишем временную таблицу *ТЧСписокНоменклатуры*, которая ранее была помещена в менеджер временных таблиц. Добавим два поля:

- *Номенклатура*
- *Количество*.

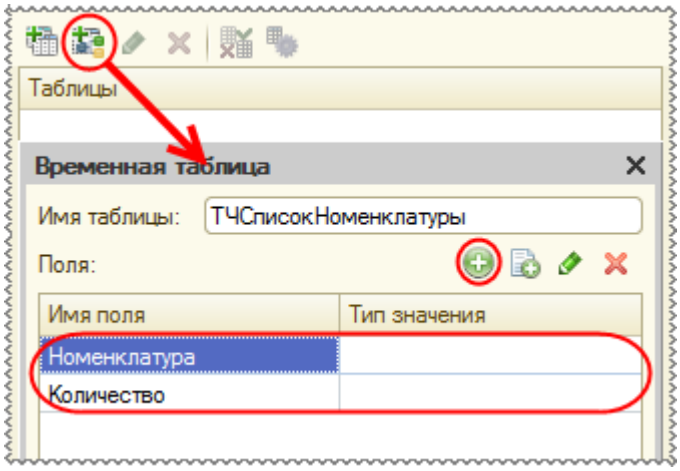


Рисунок 13 – Добавление временной таблицы

Добавим в источники виртуальную таблицу регистра накопления *СебестоимостьТоваров.Остатки*.

Остатки получим на *МоментВремени* и установим в параметрах виртуальной таблицы отбор по номенклатуре:

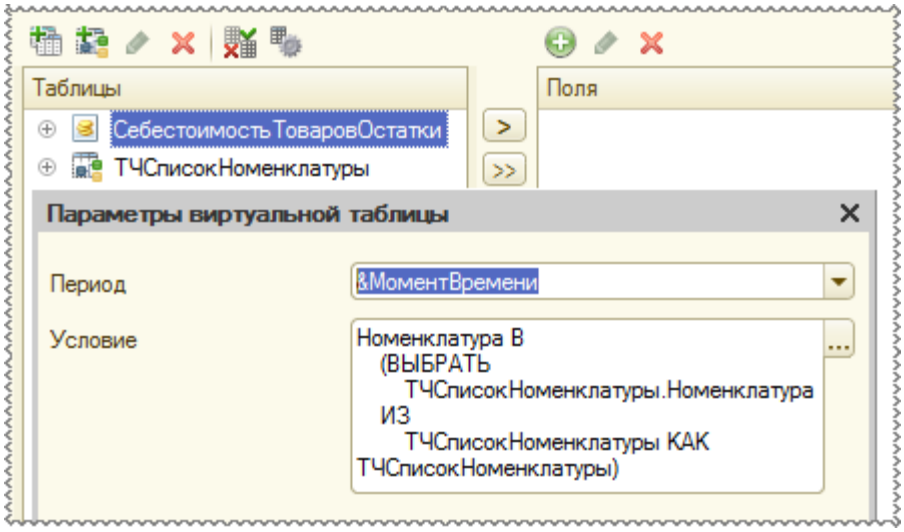


Рисунок 14 – Установка параметров виртуальной таблицы

В список полей перенесем поля:

- из временной таблицы *ТЧСписокНоменклатуры*:
  - *Номенклатура*
  - *Количество*
- из виртуальной таблицы *СебестоимостьТоваров.Остатки:*
  - *Партия.*

Добавим вычисляемые поля:

- *ЕСТЬNULL(СебестоимостьТоваровОстатки.КоличествоОстаток, 0)*
- *ЕСТЬNULL(СебестоимостьТоваровОстатки.СуммаОстаток, 0)*
- ПРЕДСТАВЛЕНИЕ(ТЧСписокНоменклатуры.Номенклатура):

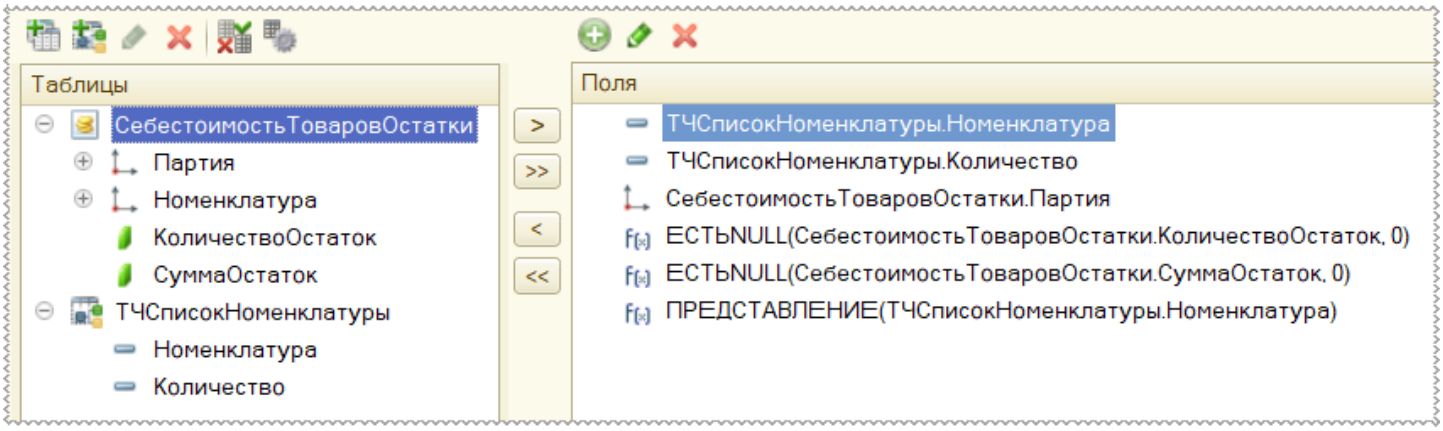


Рисунок 15 – Закладка *Таблицы и поля* конструктора запросов

Для вычисляемых полей определим соответственно псевдонимы:

- *КоличествоОстаток*
- *СуммаОстаток*
- *НоменклатураПредставление:*

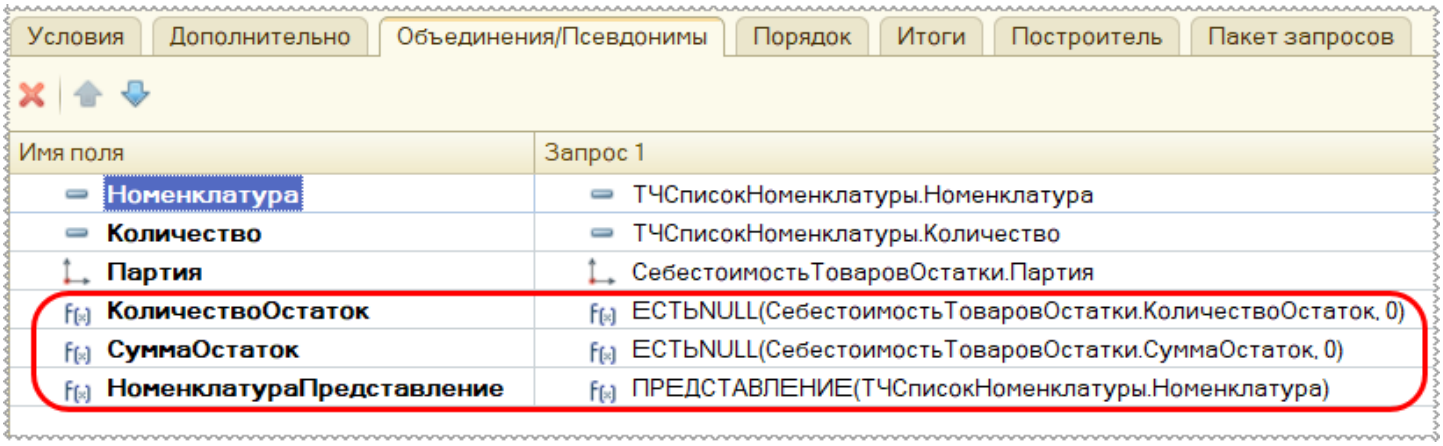


Рисунок 16 – Закладка *Объединения/Псевдонимы* конструктора запросов

Установим связь между таблицами: ЛЕВОЕ СОЕДИНЕНИЕ виртуальной таблицы *ТЧСписокНоменклатуры* с таблицей *СебестоимостьТоваров.Остатки* по полю *Номенклатура:*

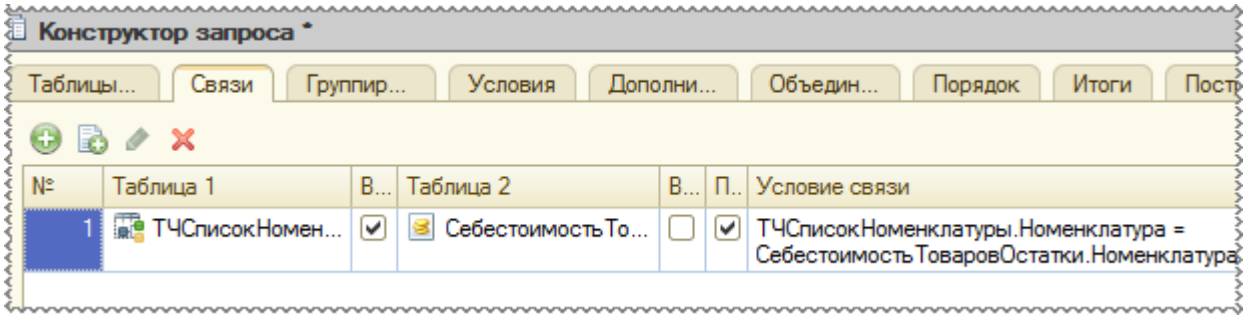


Рисунок 17 – Закладка *Связи* конструктора запросов

Установим по умолчанию сортировку результата запроса по возрастанию значения поля *СебестоимостьТоваровОстатки.Партия.МоментВремени*, что соответствует методике списания себестоимости по методу FIFO. В случае использования методики списания себестоимости по методу LIFO направление сортировки изменим программно:

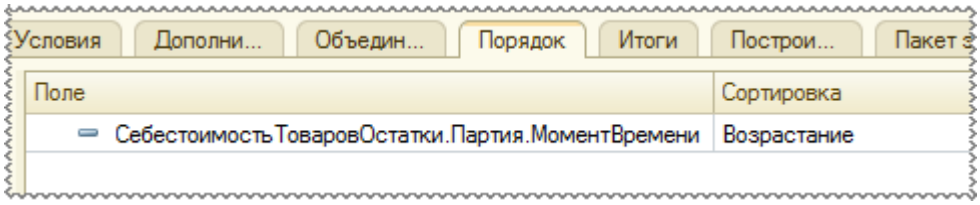


Рисунок 18 – Закладка *Порядок* конструктора запросов

Рассчитаем по группировочному полю *Номенклатура* итоговые поля:

- *Количество* – агрегатная функция *Максимум*
- *КоличествоОстаток* – агрегатная функция *Сумма:*

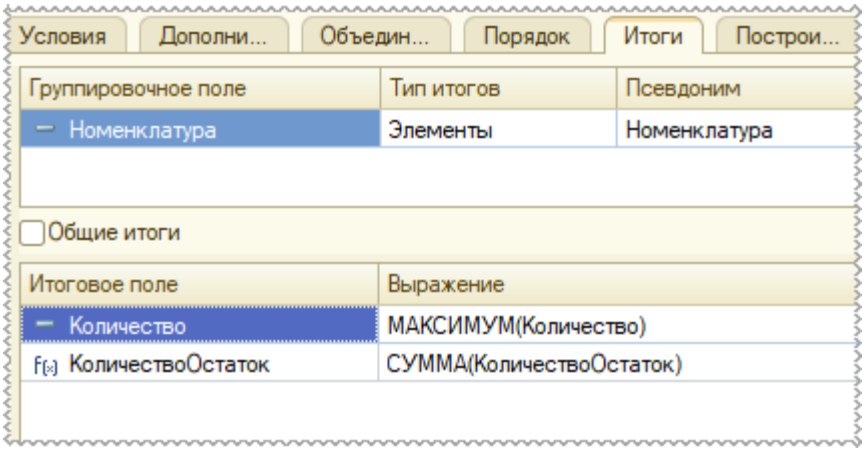


Рисунок 19 – Закладка *Итоги* конструктора запросов

Корректировка сортировки результата запроса (п.9)

По умолчанию сортировка партий в запросе установлена по возрастанию значения момента времени, что соответствует методу списания FIFO. Если согласно учетной политике используется метод списания себестоимости LIFO, то необходимо изменить направление сортировки. Для этого заменяем часть текста запроса программно.

Цикл по номенклатуре документа (п.10)

Благодаря использованию в запросе итогов во внешнем цикле, обходим результаты запроса по группировкам (по номенклатуре из документа).

Контроль остатков (п.11)

Выполним контроль остатков. Если количество в документе больше остатка по регистру, выведем сообщение пользователю. В сообщении будет передаваться информация, какой номенклатуры не хватает и в каком количестве.

Важно. При решении данной задачи на экзамене необходимо выполнять контроль остатков по двум регистрам накопления: «Себестоимость товаров» и «Остатки номенклатуры». Практического смысла контролировать остатки по регистру накопления «Себестоимость товаров» нет, но на экзамене существуют негласные правила, что контролировать нужно оба регистра. Контроль остатков только по регистру «Остатки номенклатуры» будет считаться ошибкой.

Остатка не хватает, нет смысла формировать движения (п.12)

Если остатка не хватает, нет смысла вычислять себестоимость списания и формировать движения.

Получение данных о количестве для списания (п.13)

Сохраняем количество номенклатуры, подлежащее списанию. Далее это количество будет уменьшаться.

Цикл по партиям (п.14)

Вложенный цикл будет содержать партии по текущей номенклатуре. Обходим детальные записи, пока не спишется все количество из переменной *КоличествоСписать*.

Расчет количества для списания (п.15)

Количество для списания — это минимальное значение между остатком партии и тем, что осталось списать.

Расчет суммы для списания (п.16)

Если количество по документу равно остатку по партии, то себестоимость не рассчитываем, а берем сумму из остатка по партии. Иначе рассчитываем себестоимость по простейшей формуле, согласно пропорции.

Важно. Проверку на равенство количества по документу и остатка по партии на экзамене следует делать обязательно. За ее отсутствие проверяющий может снизить оценку на 0,5- 2,0 балла:

Отсутствие в решении проверок на правильное заполнение ресурсов регистра, приводящее, например, к появлению отрицательных остатков товаров на складе. Наличие отрицательных значений ресурсов регистра допустимо, только если об этом явно сказано в задании или следует из логики учетной схемы, не противоречащей ситуации, возникающей в реальной практике ведения учета	1,0 - 2,0
Использование неправильных или упрощенных алгоритмов при расчете значений ресурсов регистра. Например, при решении «проблемы копеек»	0,5 - 2,0
Построенная в решении учетная схема не обеспечивает правильного занесения данных в регистры. Например, необходимо списать 1000, а списывается 500	2,0

Рисунок 20 – Фрагмент из списка часто встречающихся ошибок

Уменьшение количества для списания (п.17)

Уменьшаем переменную *КоличествоСписать* на списанное количество.

Формирование движений (п.18)

Количество и себестоимость рассчитана, формируем движение по регистру накопления *СебестоимостьТоваров*.

Важно. В конце процедуры не следует использовать метод *Движения.Записать()*. Данный метод не используется, так как запись будет произведена автоматически при завершении процедуры проведения документа.

Проверка результатов в режиме «1С:Предприятие»

Сформируем документы «Приходная накладная» и «Расходная накладная» по номенклатуре *Холодильник СТИНОЛ 101* согласно таблице, приведенной в начале главы:

Дата	Вид движения	Документ	Количество	Цена (руб.)	Сумма (руб.)
02.04.2018	Приход	Приход 1	10	10 000	100 000
12.04.2018	Приход	Приход 2	5	11 000	55 000
16.04.2018	Приход	Приход 3	6	11 500	69 000
17.04.2018	Расход	Расход 1	12	x	x

Движения документов в регистре «Себестоимость товаров» при методе списания FIFO:

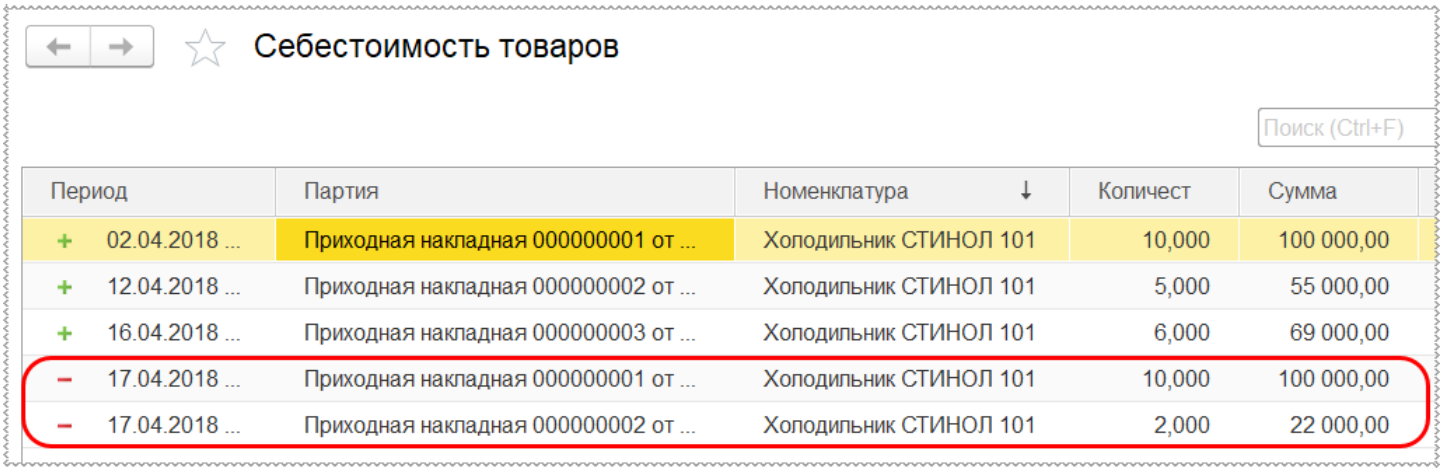
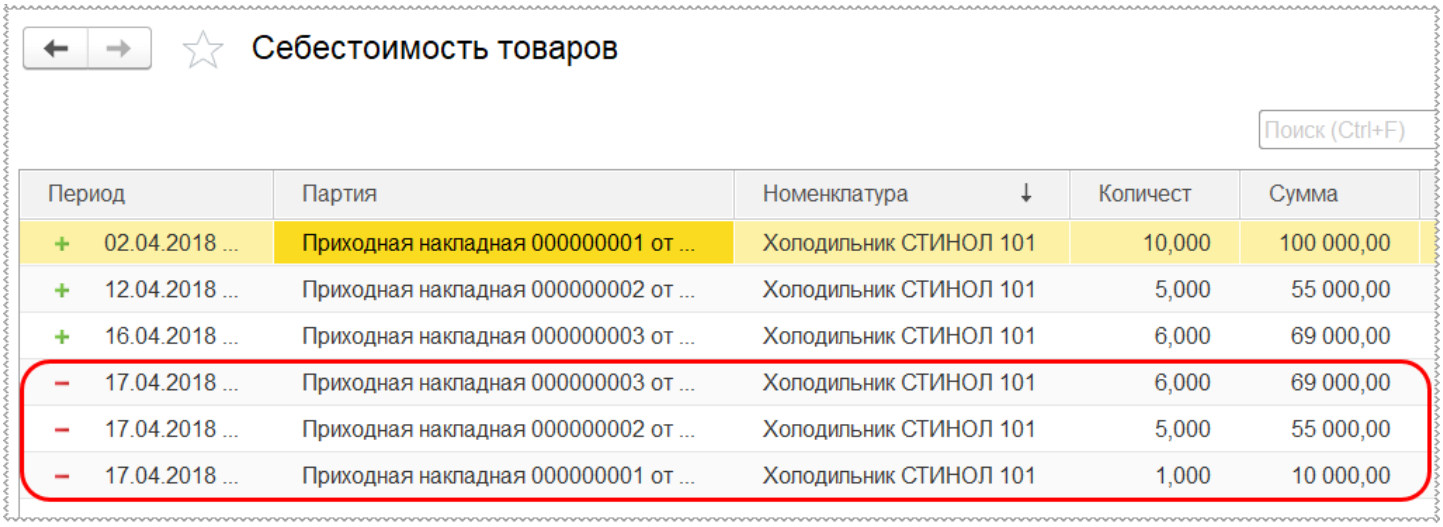


Рисунок 21 – Движения документа при методике FIFO

Движение документов в регистре «Себестоимость товаров» при методе списания LIFO:



Период	Партия	Номенклатура	Количест	Сумма
+ 02.04.2018 ...	Приходная накладная 000000001 от ...	Холодильник СТИНОЛ 101	10,000	100 000,00
+ 12.04.2018 ...	Приходная накладная 000000002 от ...	Холодильник СТИНОЛ 101	5,000	55 000,00
+ 16.04.2018 ...	Приходная накладная 000000003 от ...	Холодильник СТИНОЛ 101	6,000	69 000,00
− 17.04.2018 ...	Приходная накладная 000000003 от ...	Холодильник СТИНОЛ 101	6,000	69 000,00
− 17.04.2018 ...	Приходная накладная 000000002 от ...	Холодильник СТИНОЛ 101	5,000	55 000,00
− 17.04.2018 ...	Приходная накладная 000000001 от ...	Холодильник СТИНОЛ 101	1,000	10 000,00

Рисунок 22 – Движения документа при методике LIFO

Практические результаты совпали с теоретическими расчетами, произведенными в начале главы.

18. Как быстро сформировать печатную форму с помощью Системы Компоновки Данных

В конфигурациях 1С почти для каждого документа созданы печатные формы. Поэтому на экзамене проверяются навыки по разработке печатных форм. Создание печатных форм требуется в ряде задач из сборника для подготовки к экзамену «1С:Специалист по платформе 1С:Предприятие 8.3» (например, 1.14, 3.47, 3.49, 3.50).

Так как на экзамене время ограничено, рассмотрим пример быстрой разработки печатной формы.

> *Обратите внимание, что на реальных внедрениях печатные формы разрабатывают с помощью механизма Внешние печатные формы и БСП (библиотека стандартных подсистем). Подробное объяснение работы с внешними печатными формами приведено в курсах «**1С:Программист - Быстрый старт в профессию!**» и «**Доработка и Адаптация типовых конфигураций УТ 11.4 (11.3), КА 2.4 (2.2) и 1С:ERP 2.4 (2.2) + подготовка к Аттестации 1С:Специалист по конфигурированию торговых решений**».*

Разрабатывать печатную форму будем на базе конфигурации, полученной после разбора «новой» методики контроля остатков.

Постановка задачи:

Для документа Расходная накладная разработать печатную форму. Примерный вид печатной формы:

Расходная накладная № 105 от 01.02.2018

Товар	Количество	Цена	Сумма
Холодильник	3	10 000	30 000
Телевизор	2	7 000	14 000

Выбор механизма разработки

Печатную форму можно разработать с помощью механизмов:

- Система компоновки данных (СКД)
- Конструктор печати.

При использовании каждого из механизмов есть свои плюсы и минусы.

Для системы компоновки данных

Плюсы:

- Минимальное количество кода
- Современное решение, которое используется при выполнении практических задач.

Минусы:

- Используется объемный специфический код по программному формированию схемы компоновки данных, который нужно помнить
- Недостаточная гибкость механизма по сравнению с рисованием печатной формы вручную.

Для конструктора печати

Плюсы:

- Автоматически формируется код для команды печати
- Простота кода.

Минусы:

- После конструктора код нужно оптимизировать
- При решении реальных практических задач редко используется.

На экзамене желательно продемонстрировать экзаменатору умение работать с системой компоновки данных. Поэтому остановимся на механизме разработки печатной формы с помощью СКД.

Для решения поставленной задачи потребуются следующие объекты конфигурации:

- Макет схемы компоновки данных
- Команда «Печать».

Решение поставленной задачи

Для реализации решения необходимо выполнить следующие действия:

- Создать макет – схему компоновки данных
- Создать процедуру печати в модуле менеджера документа
- Создать команду *Печать*.

Создание макета

В каркасной конфигурации для документа «Расходная накладная» создадим макет с типом *Схема компоновки данных*. Определим для макета *Имя – Расходная накладная*:

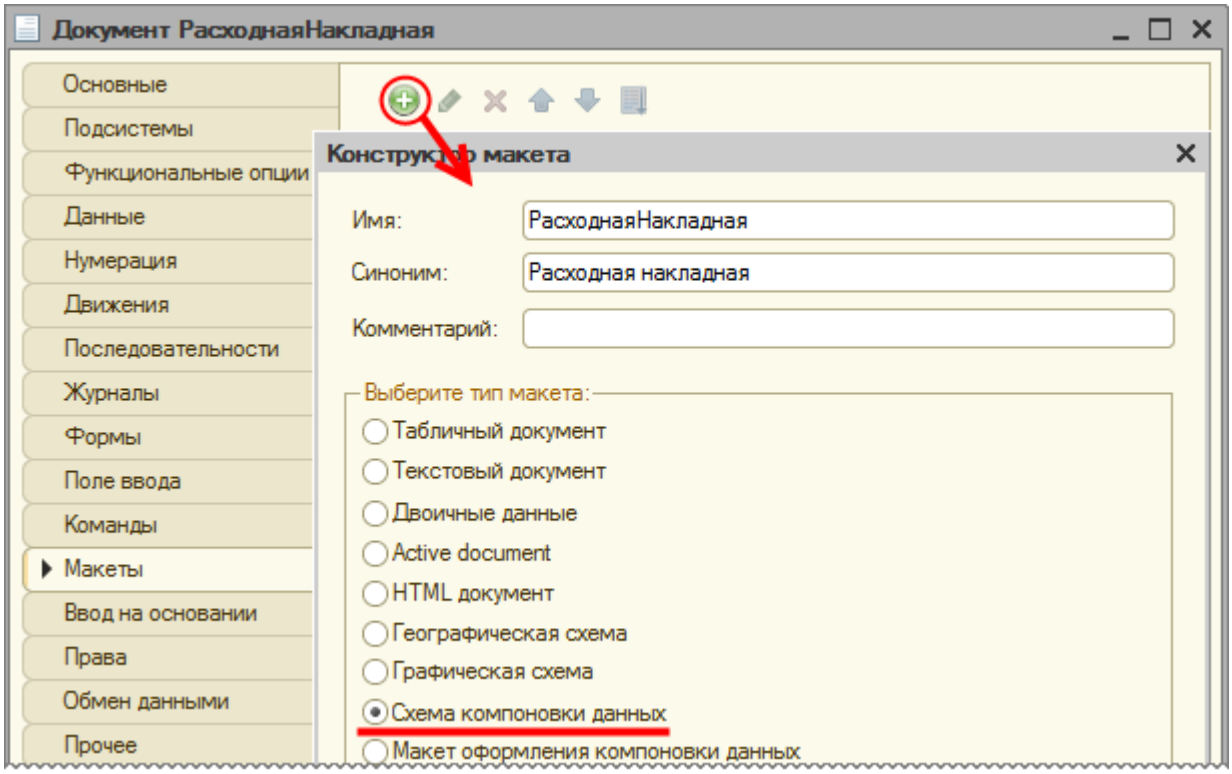


Рисунок 1 – Добавление макета СКД у документа «Расходная накладная»

В открывшейся схеме компоновки данных добавим набор данных – запрос:

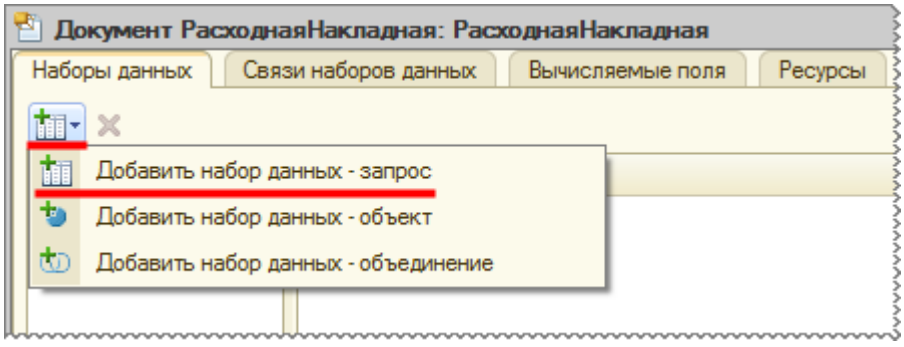


Рисунок 2 – Добавление набора данных-запрос

Составим запрос с помощью *Конструктора запроса*. В качестве источника данных выберем табличную часть *Список номенклатуры* документа «Расходная накладная».

В список полей перенесем поля:

- *Ссылка.Номер*
- *Ссылка.Дата*
- *Номенклатура*
- *Количество*
- *Цена*
- *Сумма:*

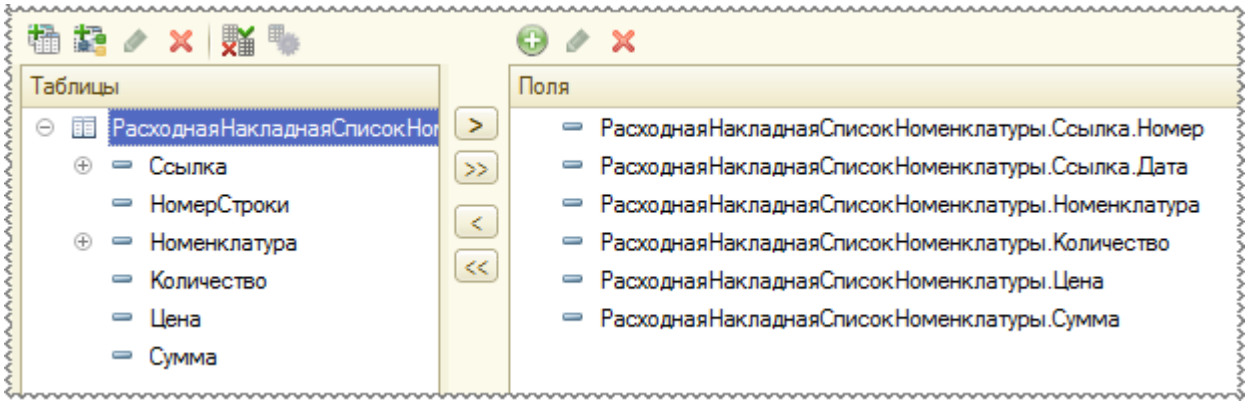


Рисунок 3 – Закладка *Таблицы и поля* конструктора запросов

Поскольку в табличной части «Список номенклатуры» может быть несколько строк с одинаковой номенклатурой, на закладке *Группировка* сгруппируем данные по полям:

- *РасходнаяНакладнаяСписокНоменклатуры.Номенклатура*
- *РасходнаяНакладнаяСписокНоменклатуры.Ссылка.Номер*
- *РасходнаяНакладнаяСписокНоменклатуры.Ссылка.Дата.*

Для ресурсов Количество, Сумма и Цена используем агрегатные функции:

- *Количество – Сумма*
- *Сумма – Сумма*
- *Цена – Среднее:*

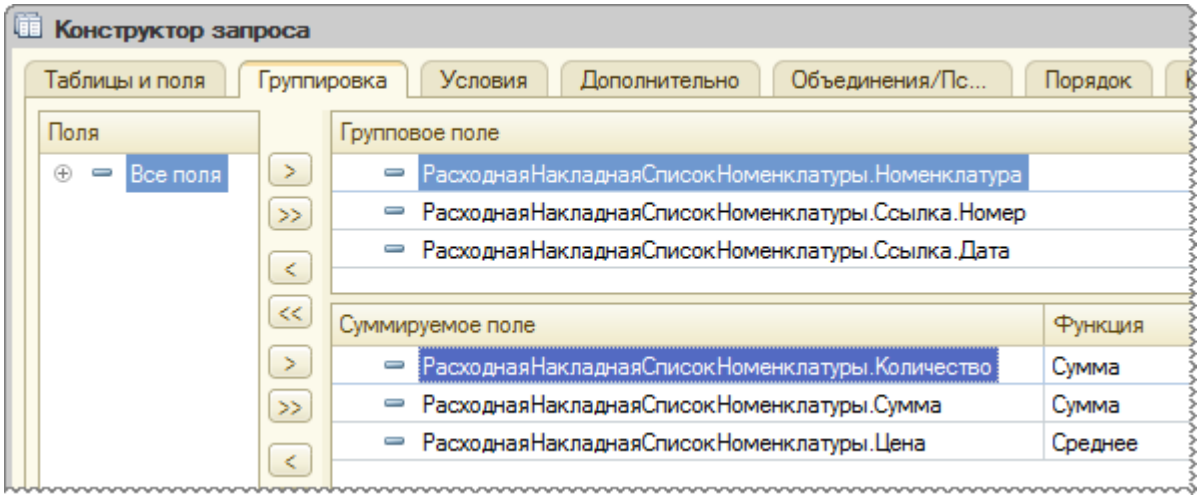


Рисунок 4 – Закладка *Группировка* конструктора запросов

Добавим условие отбора, что требуются данные табличной части *СписокНоменклатуры* только печатаемого документа:

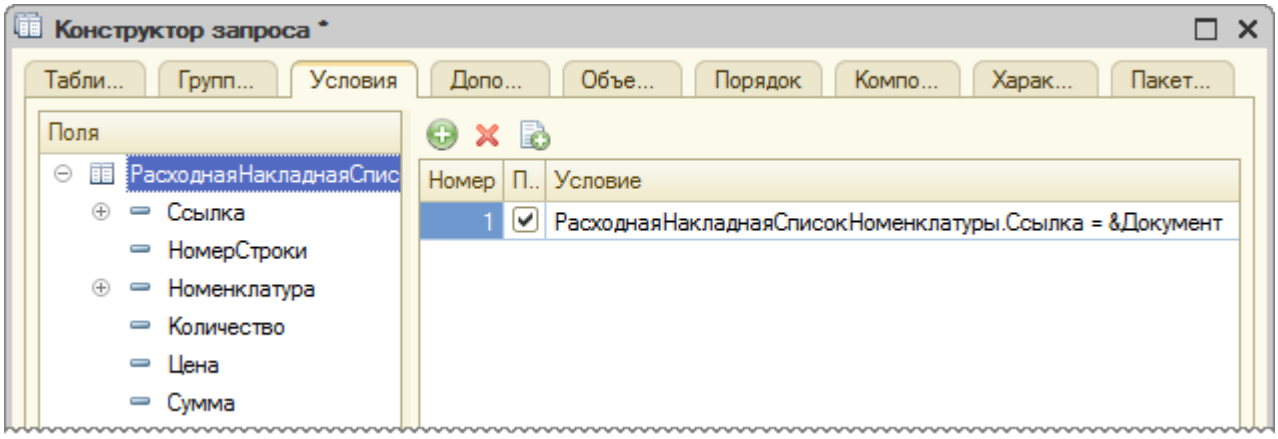


Рисунок 5 – Закладка *Условия* конструктора запросов

Конструктор формирует запрос:

```
ВЫБРАТЬ
	РасходнаяНакладнаяСписокНоменклатуры.Ссылка.Номер,
	РасходнаяНакладнаяСписокНоменклатуры.Ссылка.Дата,
	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура,
	СУММА(РасходнаяНакладнаяСписокНоменклатуры.Количество) КАК Количество,
	СРЕДНЕЕ(РасходнаяНакладнаяСписокНоменклатуры.Цена) КАК Цена,
	СУММА(РасходнаяНакладнаяСписокНоменклатуры.Сумма) КАК Сумма
ИЗ
	Документ.РасходнаяНакладная.СписокНоменклатуры КАК РасходнаяНакладнаяСписокНоменклатуры
ГДЕ
	РасходнаяНакладнаяСписокНоменклатуры.Ссылка = &Документ

СГРУППИРОВАТЬ ПО
```

```
РасходнаяНакладнаяСписокНоменклатуры.Номенклатура,
РасходнаяНакладнаяСписокНоменклатуры.Ссылка.Номер,
РасходнаяНакладнаяСписокНоменклатуры.Ссылка.Дата
```

В схеме компоновки данных перейдем на закладку *Параметры* и у параметра *Документ* снимем флаг *Ограничение доступности*, чтобы данный параметр был доступен в пользовательском режиме:

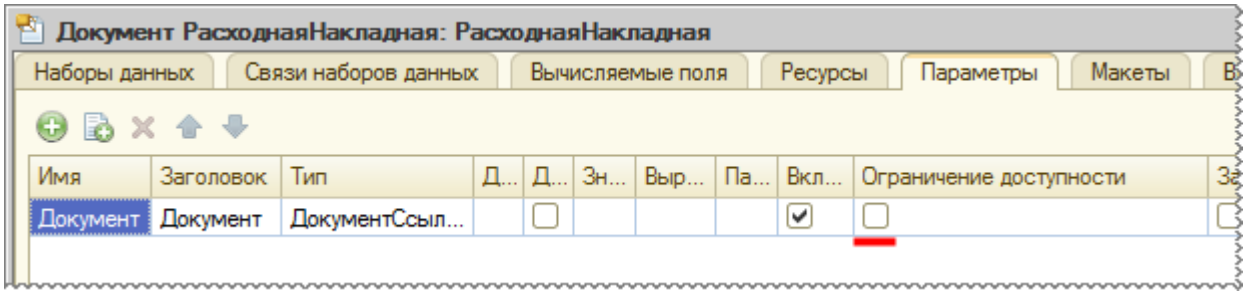


Рисунок 6 – Закладка *Параметры* схемы компоновки данных

На закладке *Макеты* настроим вывод заголовка. Для этого добавим в табличный документ:

- текст «Расходная накладная №»
- параметр *Номер*
- текст «от»
- параметр *Дата:*

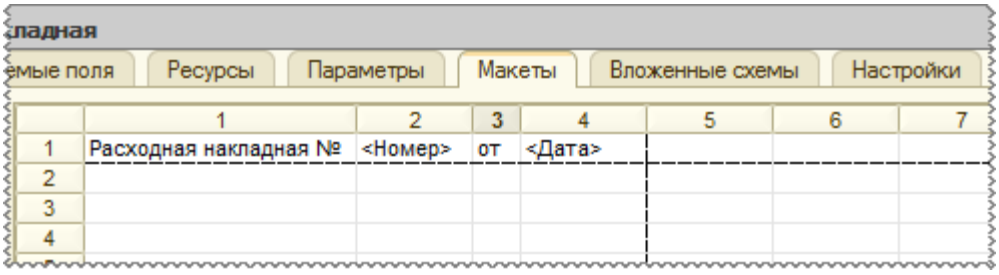


Рисунок 7 – Закладка *Макеты* схемы компоновки данных

Для параметра *Дата* укажем формат: *ДФ=dd.MM.yyyy:*

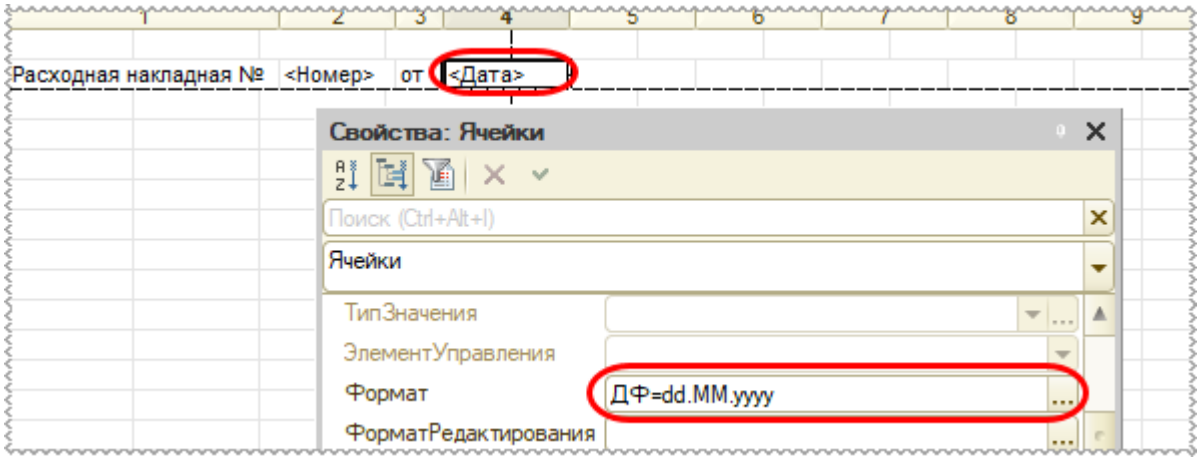


Рисунок 8 – Установка свойства *Формат* у параметра *Дата*

Добавим макет группировки.

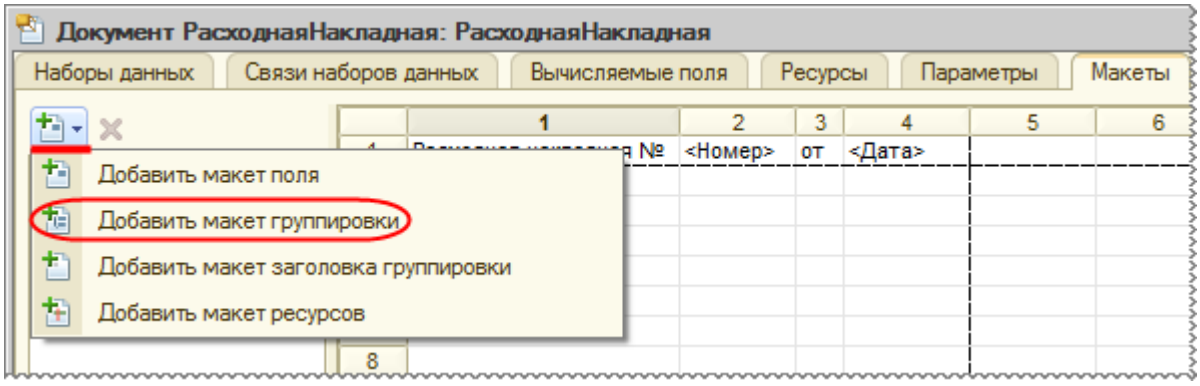


Рисунок 9 – Добавление макета группировки

Для макета группировки определим:

- *Имя группировки* – «Заголовок»
- *Тип макета – Заголовок:*

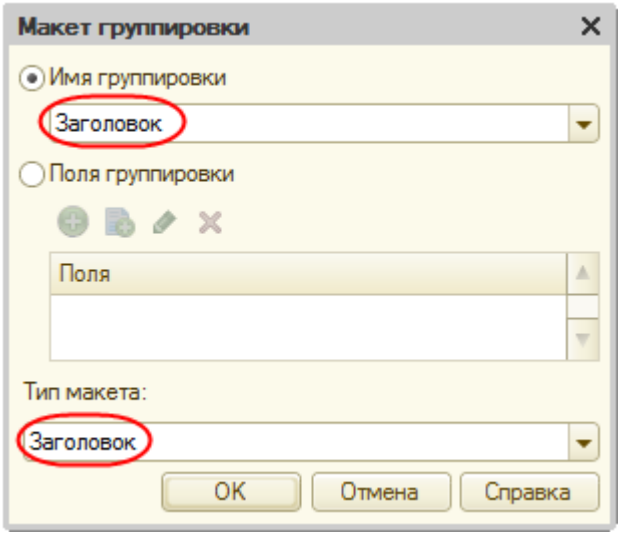


Рисунок 10 – Настройка макета группировки

Для указанного макета укажем область, заполненную ранее текстом и параметрами:

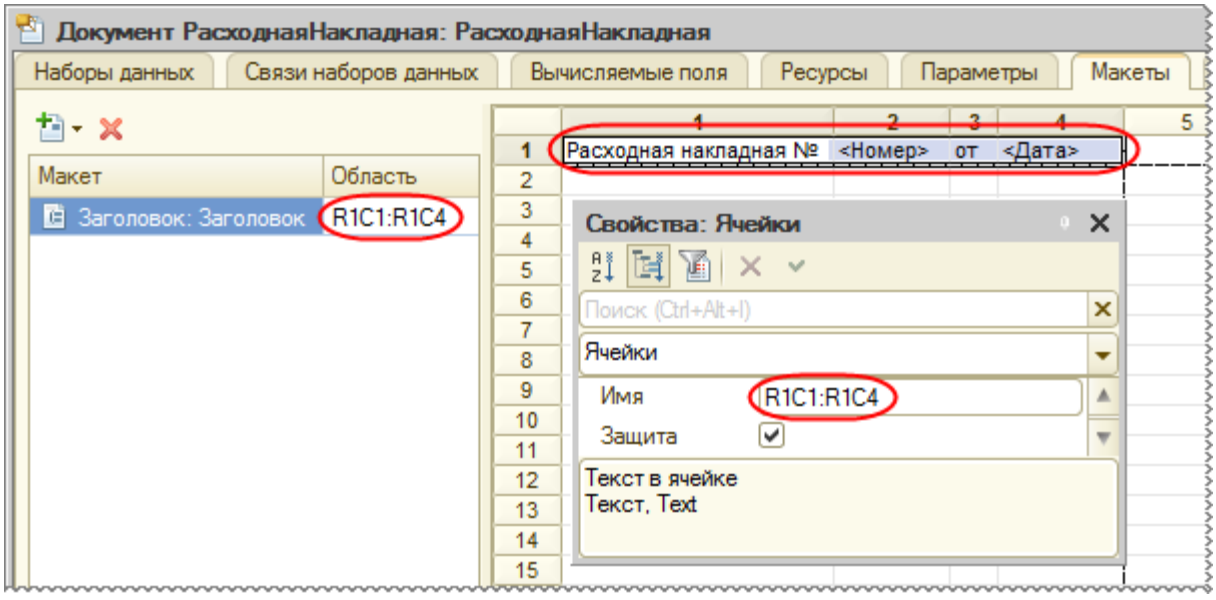


Рисунок 11 – Выбор области для макета группировки

После определения области автоматически заполнятся параметры макета группировки:

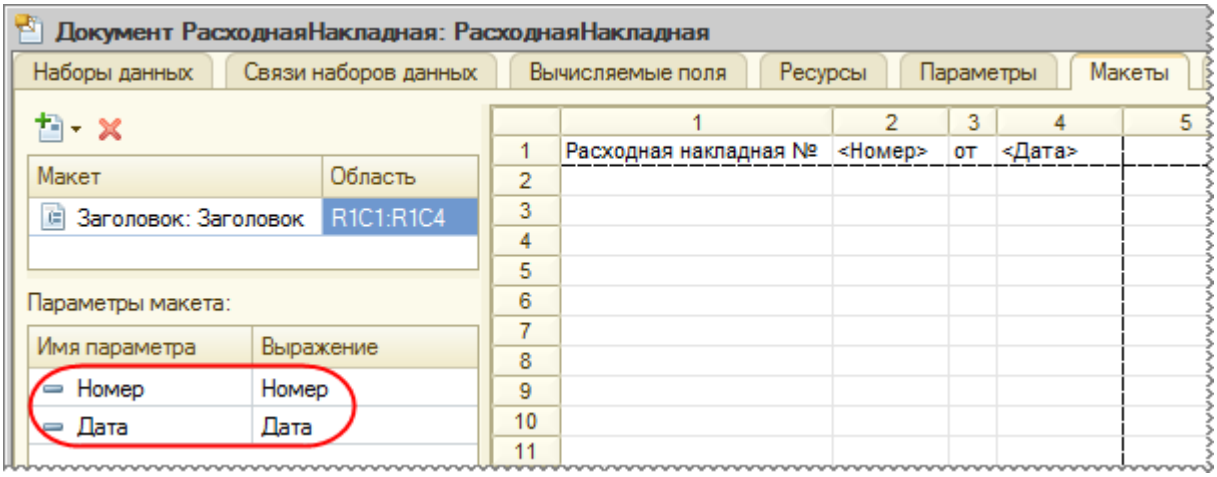


Рисунок 12 – Заполнение параметров макета

На закладке *Настройки* добавим группировки:

- *Номер, Дата*
- *Детальные записи*:

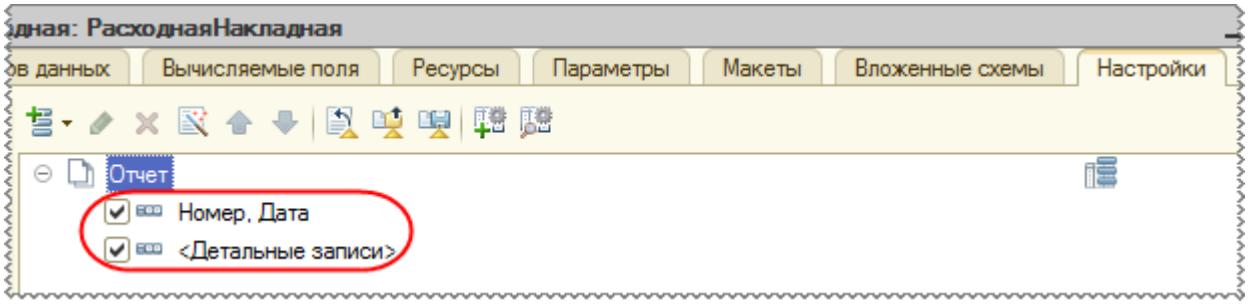


Рисунок 13 – Добавление группировок отчета

Свяжем вывод заголовка с группировкой *Номер, Дата*. С этой целью для группировки *Номер, Дата* установим имя «Заголовок», что соответствует имени группировки, созданной на закладке *Макеты:*

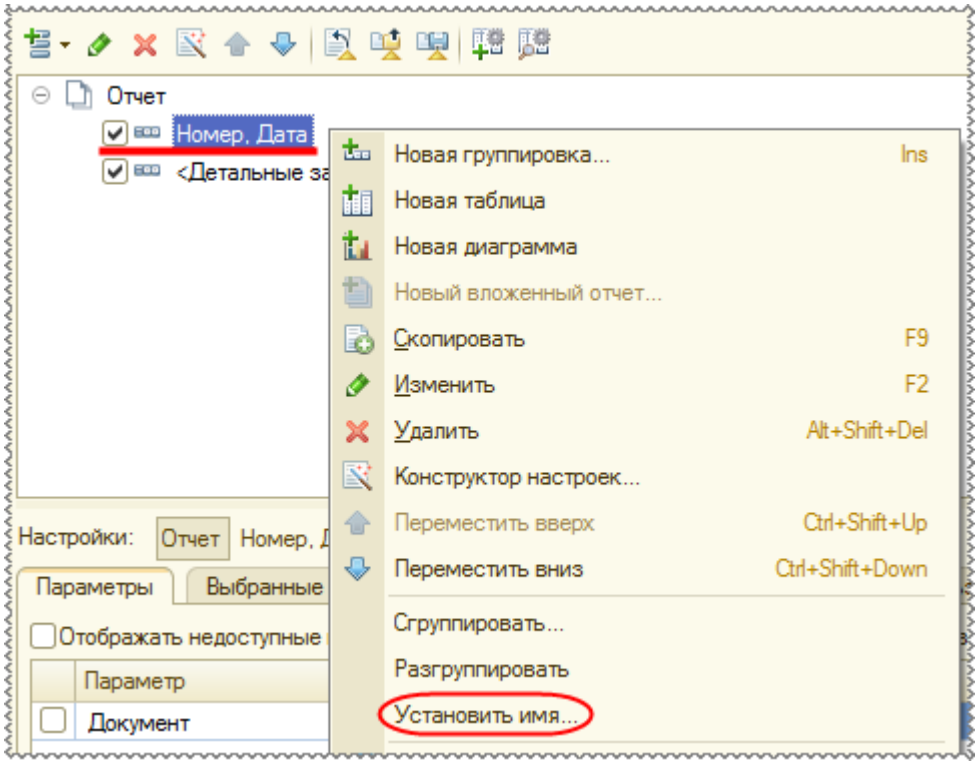


Рисунок 14 – Установка имени для группировки

Перейдем в нижнюю часть окна закладки *Настройки*.

В настройках отчета на закладке *Параметры* для параметра *Документ* установим флаг использования:

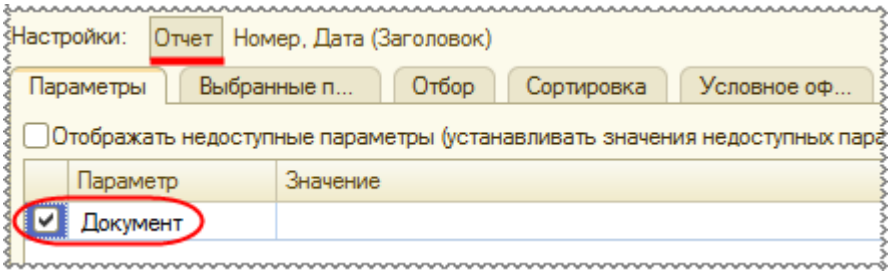


Рисунок 15 – Настройка параметров отчета

В настройках отчета на закладке *Выбранные поля* из доступных полей выберем следующие:

- *Номенклатура*
- *Количество*
- *Цена*

- *Сумма:*

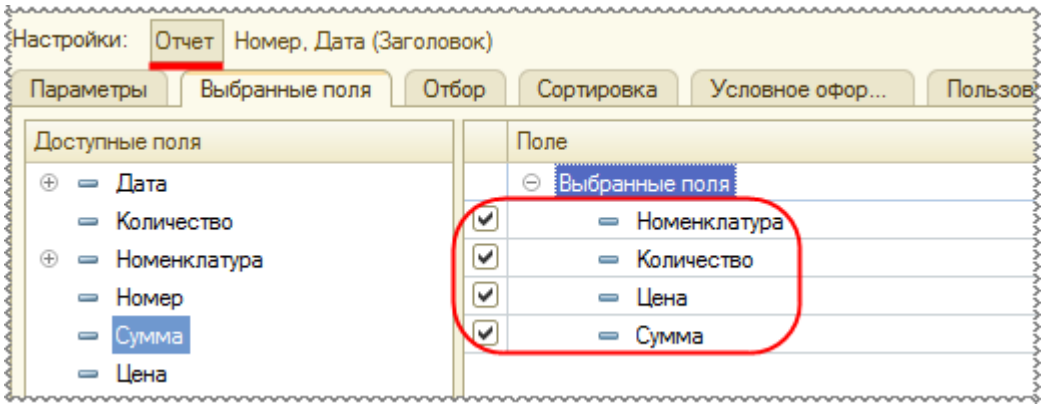


Рисунок 16 – Настройка полей отчета

На закладке *Другие настройки* для настройки *Выводить параметры* установим значение «Не выводить»:

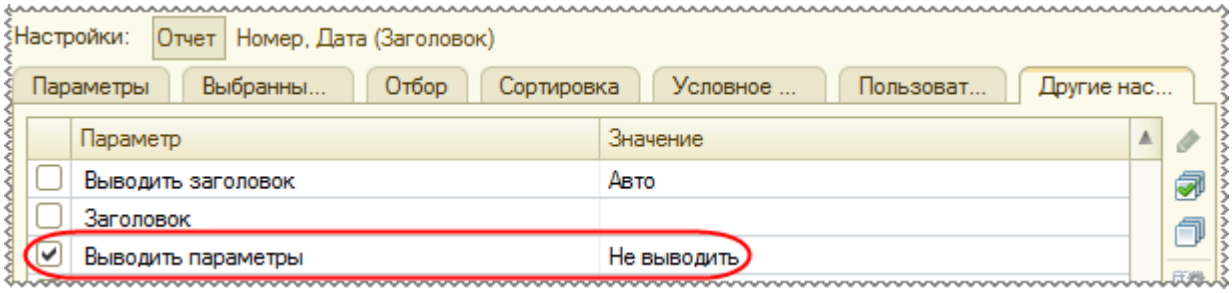


Рисунок 17 – Настройка вывода параметров отчета

Макет готов.

Создание процедуры печати в модуле менеджера документа

Откроем модуль менеджера документа «Расходная накладная» и вручную создадим процедуру *Печать*. В процедуре используется стандартный код по программному формированию макета СКД. К сожалению, в синтакс-помощнике нет примеров его использования, поэтому важно этот код запомнить.

```
Процедура Печать(Документ, ТабДок) Экспорт

	// Получение макета СКД
	Макет = Документы.РасходнаяНакладная.ПолучитьМакет("РасходнаяНакладная");

	// Инициализация компоновщика настроек и загрузка настроек
	КомпоновщикНастроек = Новый КомпоновщикНастроекКомпоновкиДанных;
```

```
	КомпоновщикНастроек.ЗагрузитьНастройки(Макет.НастройкиПоУмолчанию);

	// Установка в параметре печатаемого документа

КомпоновщикНастроек.Настройки.ПараметрыДанных.УстановитьЗначениеПараметра("Документ",Документ);

	// Инициализация компоновщика макета и процессора компоновки данных
	КомпоновщикМакета = Новый КомпоновщикМакетаКомпоновкиДанных;
	МакетКомпоновки = КомпоновщикМакета.Выполнить(Макет, КомпоновщикНастроек.Настройки);
	ПроцессорКомпоновки = Новый ПроцессорКомпоновкиДанных;
	ПроцессорКомпоновки.Инициализировать(МакетКомпоновки);

	//Вывод данных в табличный документ
	ПроцессорВывода = Новый ПроцессорВыводаРезультатаКомпоновкиДанныхВТабличныйДокумент;
	ПроцессорВывода.УстановитьДокумент(ТабДок);
	ПроцессорВывода.Вывести(ПроцессорКомпоновки);

КонецПроцедуры
```

Создание команды «Печать»

Для того чтобы печатная форма была доступна в форме документа и в форме списка документов, создадим команду *Печать* документа «Расходная накладная»:

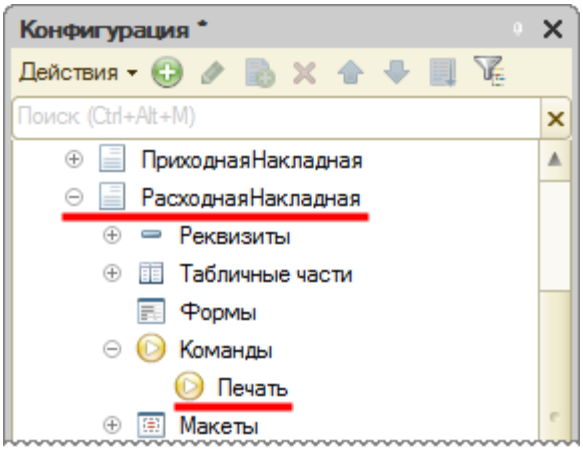


Рисунок 18 – Создание команды *Печать*

Для команды определим следующие значения свойств:

- *Группа – Командная панель формы.Важное*
- *Тип параметра команды – ДокументСсылка.РасходнаяНакладная*
- *Изменяет данные – Истина*. Данный флаг устанавливается, чтобы пользователь не смог распечатать модифицированный документ. При выполнении команды документ автоматически запишется.

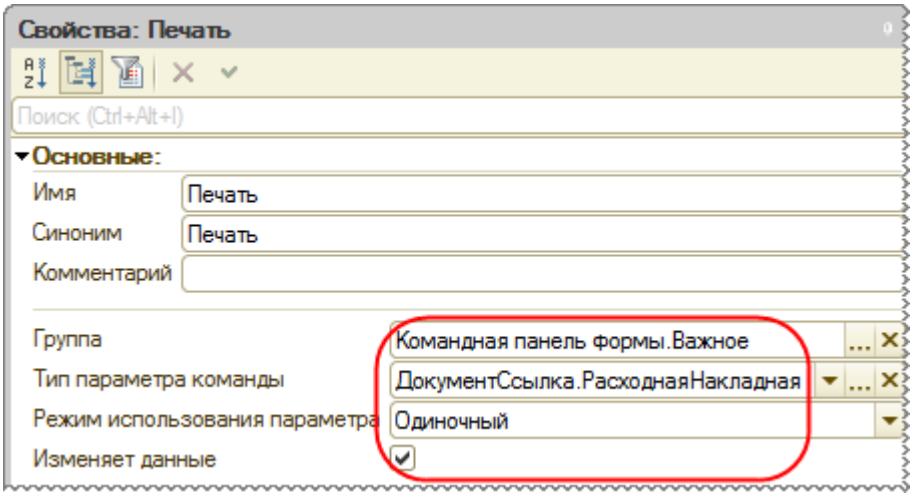


Рисунок 19 – Свойства команды *Печать*

В модуле команды сформируем процедуры и функции, необходимые для печати:

- Процедура *ОбработкаКоманды* – создается в модуле любой команды
- Функция *ДокументПроведен* – для проверки проведения документа, вызывается из процедуры *ОбработкаКоманды*
- Процедура *Печать* – для вызова ранее созданной процедуры печати модуля менеджера документа.

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

	// 1. Проверка проведения документа
	Если Не ДокументПроведен(ПараметрКоманды) Тогда
		//  2. Вывод предупреждения
		ПоказатьПредупреждение(Новый ОписаниеОповещения(), "Для печати документа необходимо его провести!",60);
		Возврат;
	КонецЕсли;

	//  3. Создание и заполнение табличного документа
	ТабДок = Новый ТабличныйДокумент;
	Печать(ПараметрКоманды, ТабДок);

	// 4. Вывод табличного документа пользователю
	ТабДок.ОтображатьСетку = Ложь;
	ТабДок.Показать();
```

```
КонецПроцедуры

&НаСервере
Функция ДокументПроведен(ПараметрКоманды)
	Возврат ПараметрКоманды.Проведен;
КонецФункции

&НаСервере
Процедура Печать(ПараметрКоманды, ТабДок)
	Документы.РасходнаяНакладная.Печать(ПараметрКоманды, ТабДок);
КонецПроцедуры
```

Рассмотрим ключевые точки алгоритма.

Проверка проведения документа (п. 1)

Выполним проверку как в типовых конфигурациях 1С. Печатать будем только проведенный документ. С целью проверки, что документ проведен, в модуле команды сформирована функция *ДокументПроведен*, которая возвращает соответствующее значение: *Истина* или *Ложь*.

Вывод предупреждения (п. 2)

Если документ не проведен, система будет выводить предупреждение и выполнять возврат.

Создание и заполнение табличного документа (п. 3)

Создается табличный документ. Для вызова ранее созданной процедуры печати модуля менеджера документа и передачи в нее табличного документа сформирована процедура *Печать* в модуле команды.

Вывод табличного документа пользователю (п. 4)

Сформированный табличный документ показывается пользователю.

Обновим конфигурацию базы данных и в режиме «1С:Предприятие» сформируем печатную форму документа:

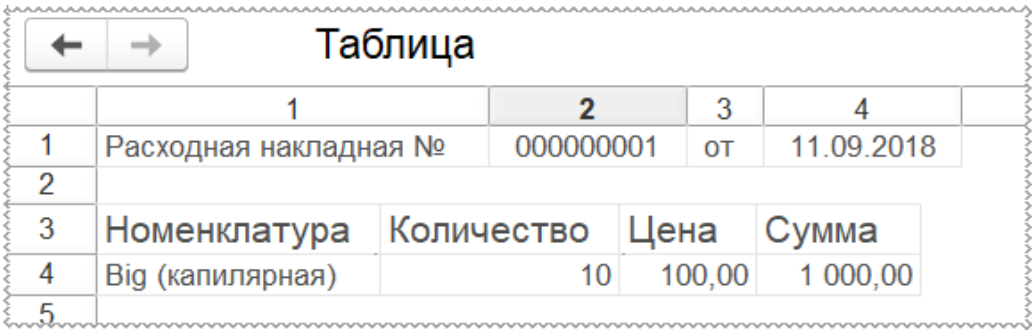


Рисунок 19 – Сформированная печатная форма

Поставленная задача решена. Печатная форма документа разработана с использованием системы компоновки данных.

19. О каких особенностях операции «Комплектация номенклатуры» нужно знать на экзамене

Комплектация номенклатуры – это операция, в результате которой перечень товарно-материальных ценностей (ТМЦ) преобразуется в новую номенклатуру – комплект. Комплектация может выполняться отдельным документом, который оприходует собранный комплект на склад до момента его списания. Также комплектация может выполняться непосредственно при проведении документа, которым производится списание комплекта. В операции комплектации не учитывают дополнительные затраты и/или услуги, связанные со сборкой, то есть комплектация – это операция только с ТМЦ.

Комплект – это набор, состоящий из нескольких ТМЦ, объединенных в процессе комплектации (сборки).

Комплектующие – это ТМЦ, которые необходимо использовать для получения конечной номенклатуры (комплекта).

Список комплектующих хранится для каждого комплекта. В нем указывается из какого количества деталей будет состоять определенное количество готовых комплектов. Например, один комплект «Стеллаж «Стандартный» состоит из четырех стоек, трех полок и двенадцати болтов:

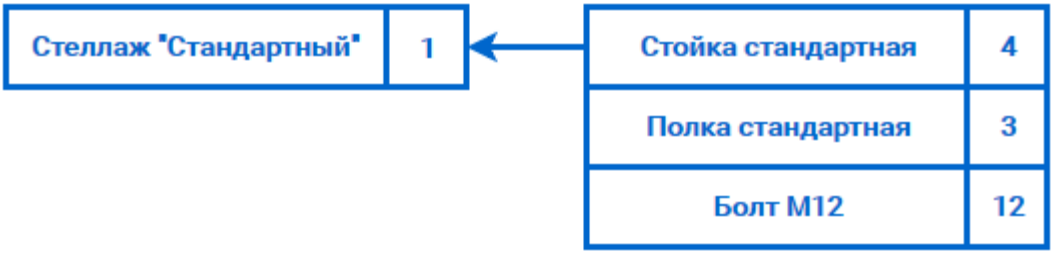


Рисунок 1 – Состав комплекта

В качестве комплектующей может быть указан другой комплект, тогда процедура расчета необходимых для сборки итогового комплекта деталей несколько усложняется:

Рисунок 2 – Состав комплекта, в который входит другой комплект

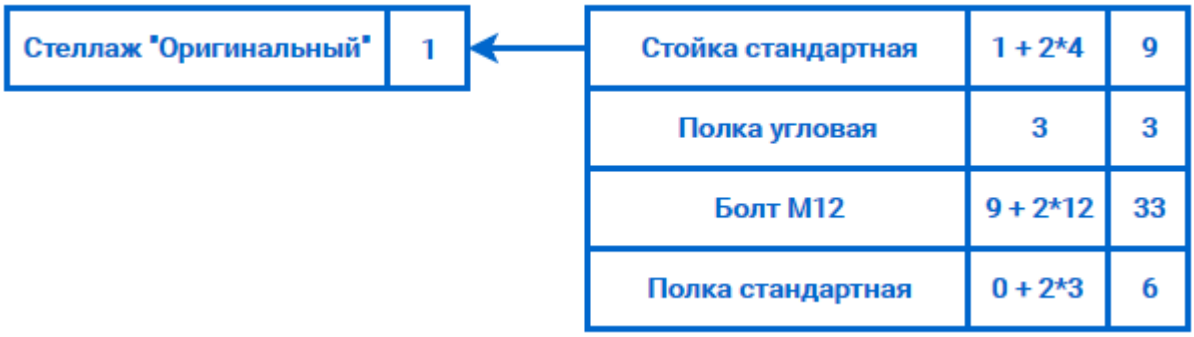


Рисунок 3 – Расчет комплектующих для комплекта, в который входит другой комплект

При этом комплектующие – это такие же номенклатурные позиции, как и комплекты. Чаще всего они могут продаваться как самостоятельный товар наравне с комплектами.

На экзамене задачи на комплектацию встречаются в разделах как оперативного, так и бухгалтерского учета. Пример задач – 1.13, 1.16, 1.49 (в состав комплекта может входить другой комплект), 2.27, 2.28. Но, независимо от типа учета, существуют общие принципы решения таких задач. Один из таких принципов – это **организация хранения информации о составе комплекта**.

Рассмотрим этап создания системы хранения данных о составе комплекта подробно.

Организация хранения информации о комплектах

Список комплектов и комплектующих должен храниться в одном справочнике – «Номенклатура», так как и те, и другие являются самостоятельными товарами. Однако эти номенклатурные позиции следует разделить по видам, то есть для комплектов указать, что они отличаются от комплектующих.

Деление номенклатуры по видам необходимо по нескольким причинам:

- В большинстве задач сборника явно указывается, что в состав одного комплекта не может входить другой комплект. Деление по виду номенклатуры позволит настроить ограничение выбора номенклатуры в реквизитах документов, фиксирующих состав комплекта и операцию сборки комплекта
- В некоторых задачах присутствует условие, по которому в документе продажи в одной табличной части могут быть указаны и комплекты, и комплектующие. При проведении таких документов следует выполнять операцию сборки комплектов перед их продажей. Чтобы определить, по каким позициям требуется выполнить комплектацию, необходимо разделять товары по видам.

Список комплектующих для комплекта правильнее всего хранить в периодическом регистре сведений. Причина такого выбора в том, что во многих задачах на экзамене звучит требование: при проведении документа задним числом его движения должны оставаться без изменений. Документ продажи при проведении использует информацию о составе комплектов, а состав комплекта нужно периодически изменять.

Для выполнения требования неизменности движений документа необходимо хранить изменения состава комплекта в привязке к периоду. В этом случае ранее проведенные документы получат из базы тот состав, который был введен на дату их создания.

Например, состав комплекта «Стеллаж «Стандартный» был установлен на начало 2018 года, а с начала июня месяца изменился:

Рисунок 4 – Изменение состава комплекта со временем

В июле месяце стали перепроводить документы за второй квартал 2018 года. Документы апреля и мая должны провести комплектацию по старому составу, а документы июня – по новому. При отсутствии привязки состава комплекта к периоду такое перепроведение правильно выполнить не

получится. При проведении документов апреля и мая система получит данные о составах комплектов, актуальные на июнь, и движения этих документов изменятся.

В «1С:Предприятии 8» объектом, который хранит привязку справочной информации к периоду, является регистр сведений. Поэтому его и выбирают для хранения информации о составе комплекта.

Строго говоря, даже использование периодического регистра сведений не гарантирует стопроцентной неизменности информации при перепроведении документов задним числом.

В приведенном примере пользователь может исправить состав комплекта от 01.01.18. В этом случае перепроведение задним числом документов, использовавших этот набор комплектующих, приведет к изменению данных в их движениях. В реальных конфигурациях рекомендуется фиксировать данные, полученные из дополнительных источников, в реквизитах самого документа. Но на экзамене такое решение задачи комплектации номенклатуры будет излишне сложным и займет неоправданно большое количество времени.

Регистр сведений, который будет хранить информацию о составе комплекта, подчиняем регистратору. С помощью документа фиксировать информацию в регистр сведений будет удобнее.

Решение практической задачи

Постановка задачи:

Компания занимается сборкой и реализацией комплектов. Для каждого комплекта вводится информация о составе (какие номенклатурные позиции и в каком количестве входят в данный комплект). В состав комплекта не могут входить другие комплекты. И комплекты, и комплектующие могут указываться в одной табличной части документа реализации.

Для решения поставленной задачи потребуются следующие объекты конфигурации:

- Перечисление «ВидыНоменклатуры»
- Справочник «Номенклатура»
- Регистр сведений «СоставКомплектов»
- Документ «ИзменениеСоставаКомплекта».

Настройку будем выполнять по следующему плану действий:

- Проверим, какие объекты уже имеются в каркасной конфигурации и могут быть использованы для решения поставленной задачи
- Создадим недостающие объекты метаданных
- Проверим результаты в режиме «1С:Предприятие».

Используемые объекты каркасной конфигурации

Перечисление «ВидыНоменклатуры» в каркасной конфигурации уже имеется. Добавим значение «Комплект»:

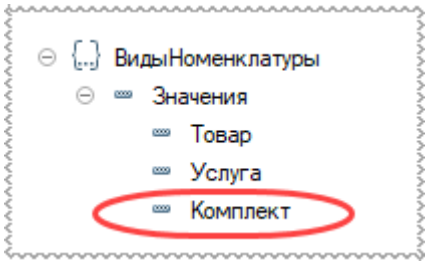


Рисунок 5 – Значения перечисления «ВидыНоменклатуры»

Справочник «Номенклатура» в каркасной конфигурации тоже имеется. Добавим реквизит «ВидНоменклатуры» (ПеречислениеСсылка.ВидыНоменклатуры):

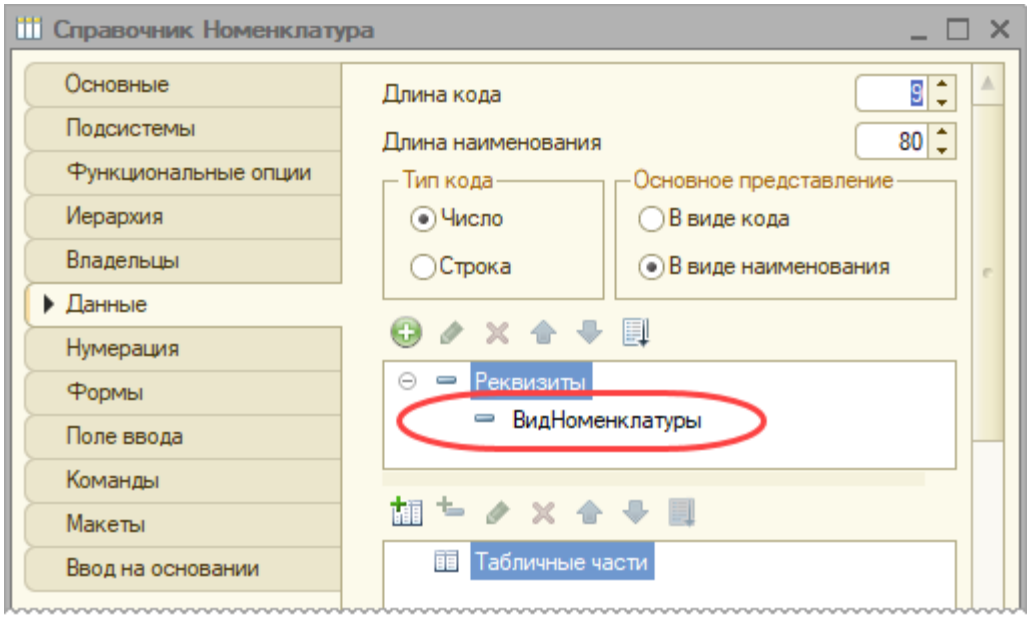


Рисунок 6 – Закладка *Данные* справочника «Номенклатура»

Сделаем для справочника настройку, запрещающую менять значение реквизита «ВидНоменклатуры» для записанного элемента. Запрет нужен для выполнения условия задачи, по которому в состав комплекта не может входить другой комплект. Существующий элемент справочника с видом номенклатуры «Товар» может быть указан в качестве детали для какого-либо комплекта. Изменение его вида на «Комплект» приведет к тому, что в составе одного комплекта появится другой комплект, что противоречит условию задачи.

На закладке *Формы* добавим основную форму справочника:

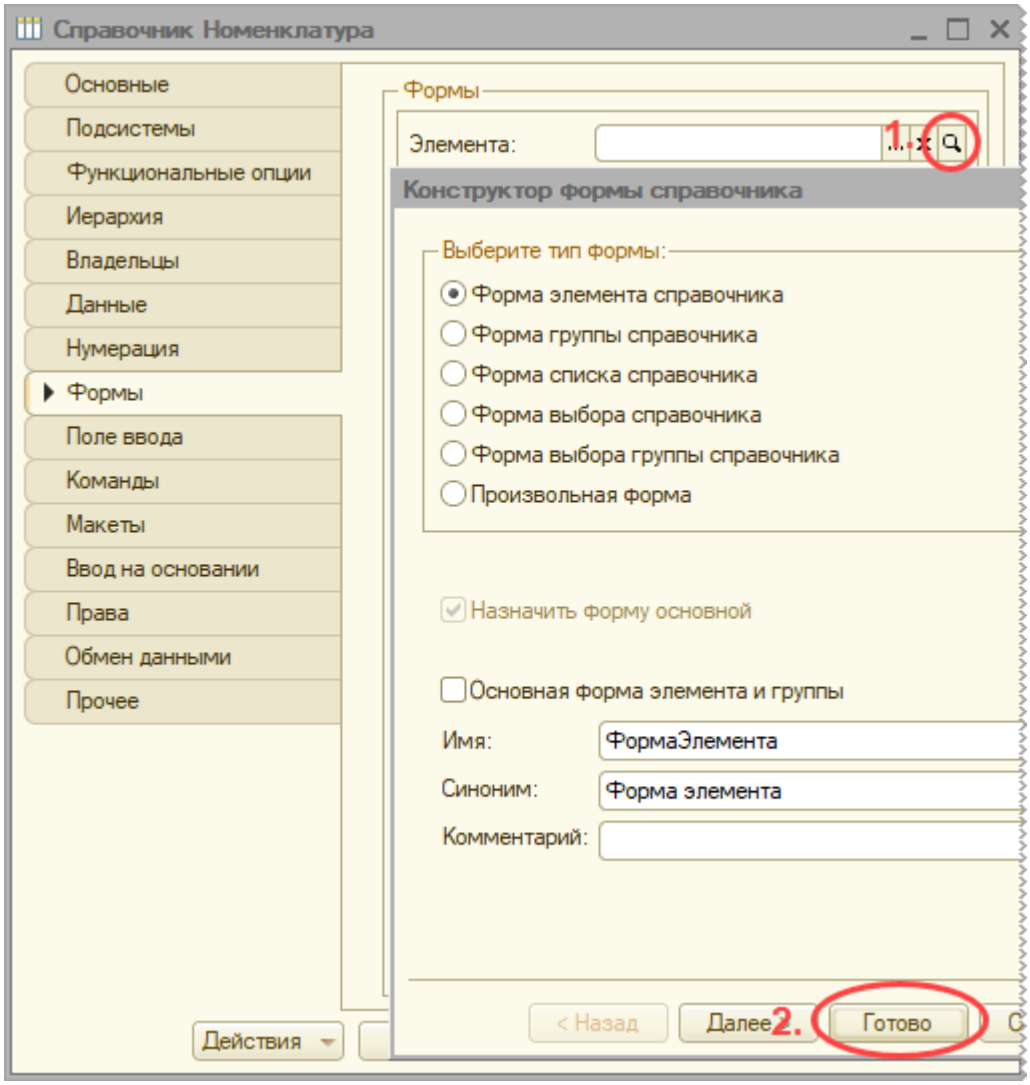


Рисунок 7 – Вызов конструктора формы справочника

В модуле вновь созданной формы создадим обработчик события *ПриСозданииНаСервере*, в котором программно установим для элемента формы «ВидНоменклатуры» свойство *ТолькоПросмотр*:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

	Если ЗначениеЗаполнено(Объект.Ссылка) Тогда
		Элементы.ВидНоменклатуры.ТолькоПросмотр = Истина;
	КонецЕсли;

КонецПроцедуры
```

Создание регистра сведений «Состав комплекта» и документа «Изменение состава комплекта»

Для хранения состава комплектов создадим периодический регистр сведений «СоставКомплекта» со значениями свойств:

- *Режим записи – По позиции регистратора*, то есть записи в регистр сведений будут попадать при проведении документа-регистратора
- *Периодичность – Подчинение регистратору:*

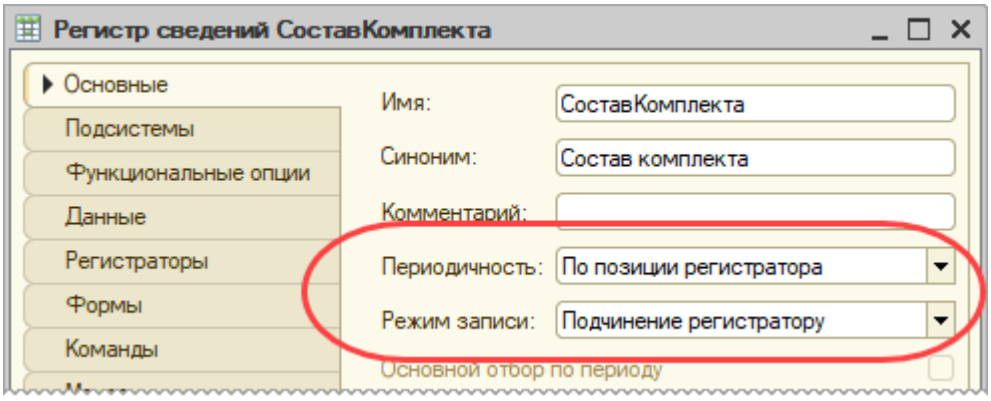


Рисунок 8 – Закладка *Основные* регистра сведений «СоставКомплекта»

В окне редактирования объекта конфигурации на закладке *Данные* создадим структуру регистра сведений.

Измерения:

- «Комплект» (СправочникСсылка.Номенклатура)
- «Комплектующая» (СправочникСсылка.Номенклатура).

Для измерений включим свойство «Запрет незаполненных значений», чтобы система автоматически запрещала записи с незаполненными измерениями в регистр. Теперь не будет необходимости в дополнительной проверке на заполненность измерений при формировании кода записи движений в регистр сведений.

Ресурсы:

- «Количество» (Число 10, 0).

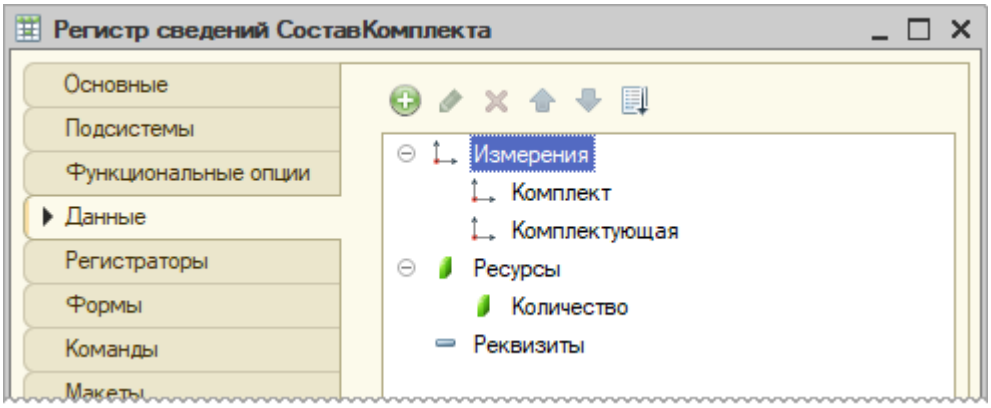


Рисунок 9 – Закладка *Данные* регистра сведений «СоставКомплекта»

Для заполнения созданного регистра сведений добавим в конфигурацию новый документ «ИзменениеСоставаКомплекта»:

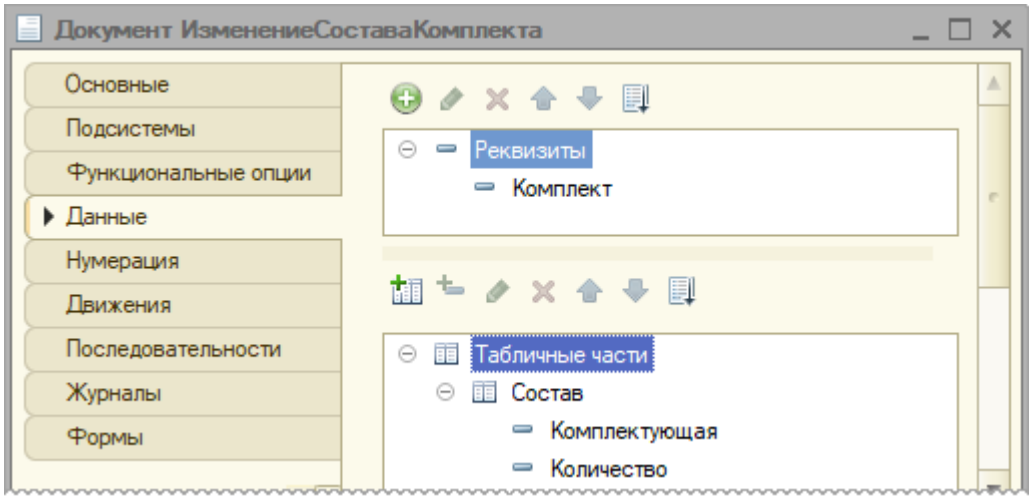


Рисунок 10 – Закладка *Данные* документа «ИзменениеСоставаКомплекта»

В окне редактирования объекта конфигурации на закладке *Данные* создадим структуру документа:

- Реквизит «Комплект» (СправочникСсылка.Номенклатура).

Для реквизита «Комплект» настроим параметры выбора значения:

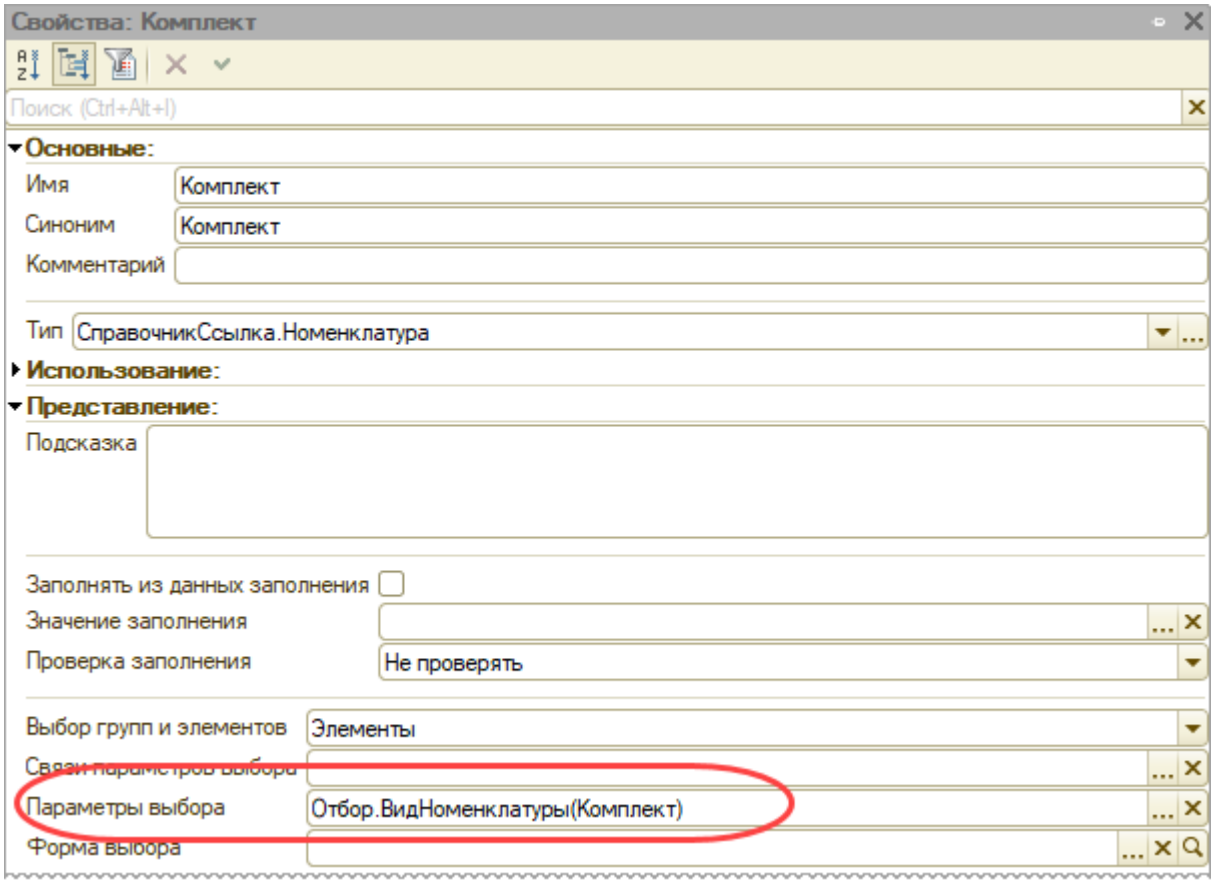


Рисунок 11 – Настройка параметров выбора реквизита «Комплект»

С помощью этой настройки запретим в пользовательском режиме указывать в шапке документа не комплекты.

Табличная часть «Состав»:

- «Комплектующая» (СправочникСсылка.Номенклатура)
- «Количество» (Число 10, 0).

Для реквизита «Комплектующая» табличной части «Состав» настроим параметры выбора значения:

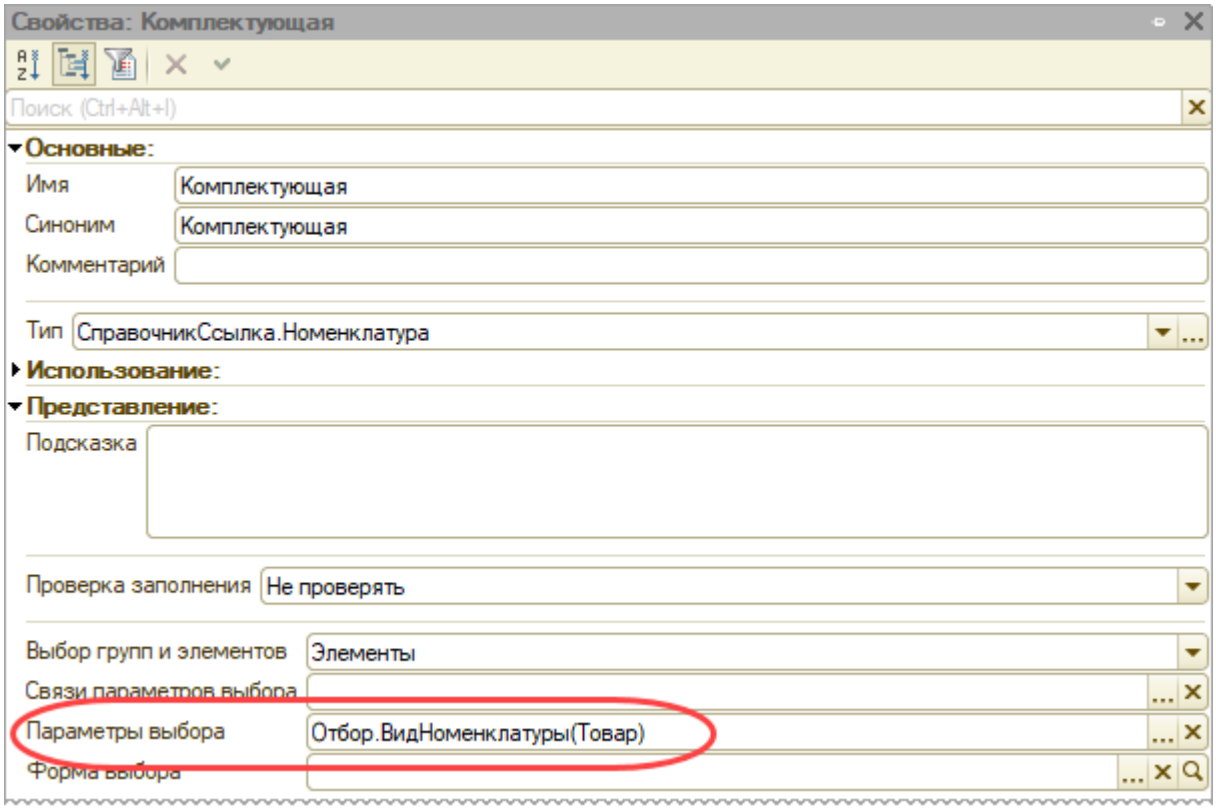


Рисунок 12 – Настройка параметров выбора реквизита «Комплектующая»

Таким образом, выполним условие задания, по которому в состав комплекта не могут входить другие комплекты.

При создании структуры документа будем считать, что состав вводится из расчета на один комплект (т.к. в задании не сказано иного), поэтому количество комплектов в шапке указывать не будем.

В окне редактирования объекта конфигурации на закладке *Движения* укажем, что документ будет регистратором для регистра сведений «СоставКомплекта»:

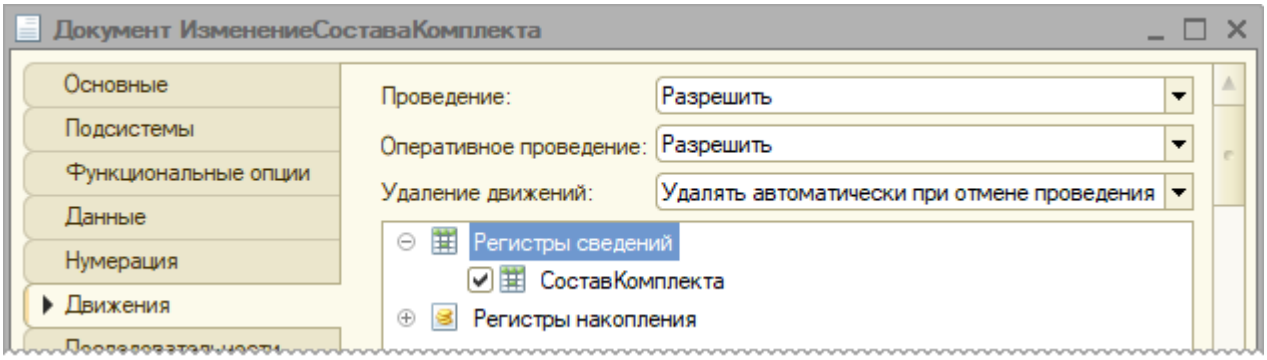


Рисунок 13 – Закладка *Движения* документа «ИзменениеСоставаКомплекта»

Создадим процедуру *ОбработкаПроведения* для документа «ИзменениеСоставаКомплекта». Для этого воспользуемся конструктором движений, работа с которым подробно описана в блоке **«2. Как правильно реализовать движения документов по регистрам»**:

```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр СоставКомплекта
	Движения.СоставКомплекта.Записывать = Истина;
	Для Каждого ТекСтрокаСостав Из Состав Цикл
		Движение = Движения.СоставКомплекта.Добавить();
		Движение.Период = Дата;
		Движение.Комплект = Комплект;
		Движение.Комплектующая = ТекСтрокаСостав.Комплектующая;
		Движение.Количество = ТекСтрокаСостав.Количество;
	КонецЦикла;

	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Для экономии времени создавать форму документа «ИзменениеСоставаКомплекта» не будем.

Проверка результатов в режиме «1С:Предприятие»

В пользовательском режиме введем набор тестовых данных для проверки работы сделанных изменений в конфигурации:

1. Введем в справочник «Номенклатура» несколько комплектов и деталей к ним:

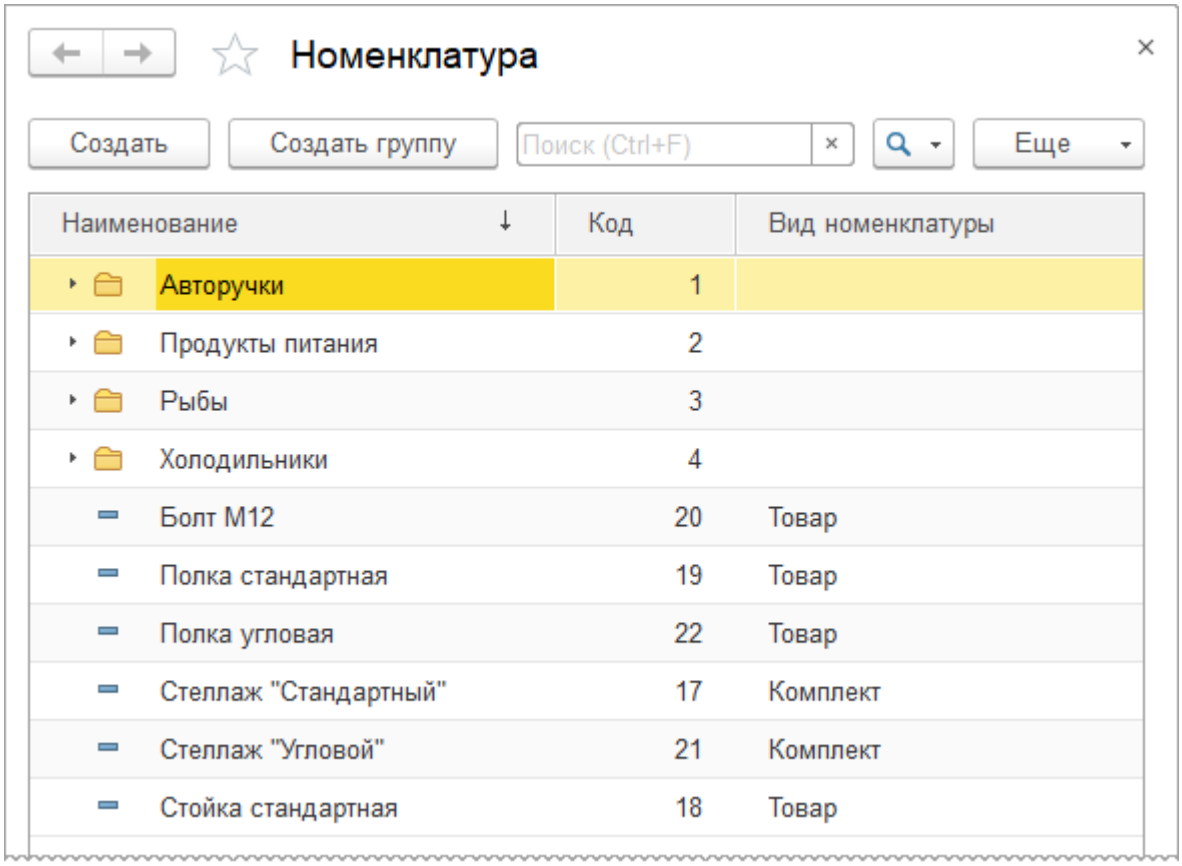


Рисунок 14 – Пример заполнения справочника «Номенклатура» в режиме «1С:Предприятие»

2. Введем документы «Изменение состава комплекта» для «Стеллаж «Стандартный» и «Стеллаж «Угловой»:

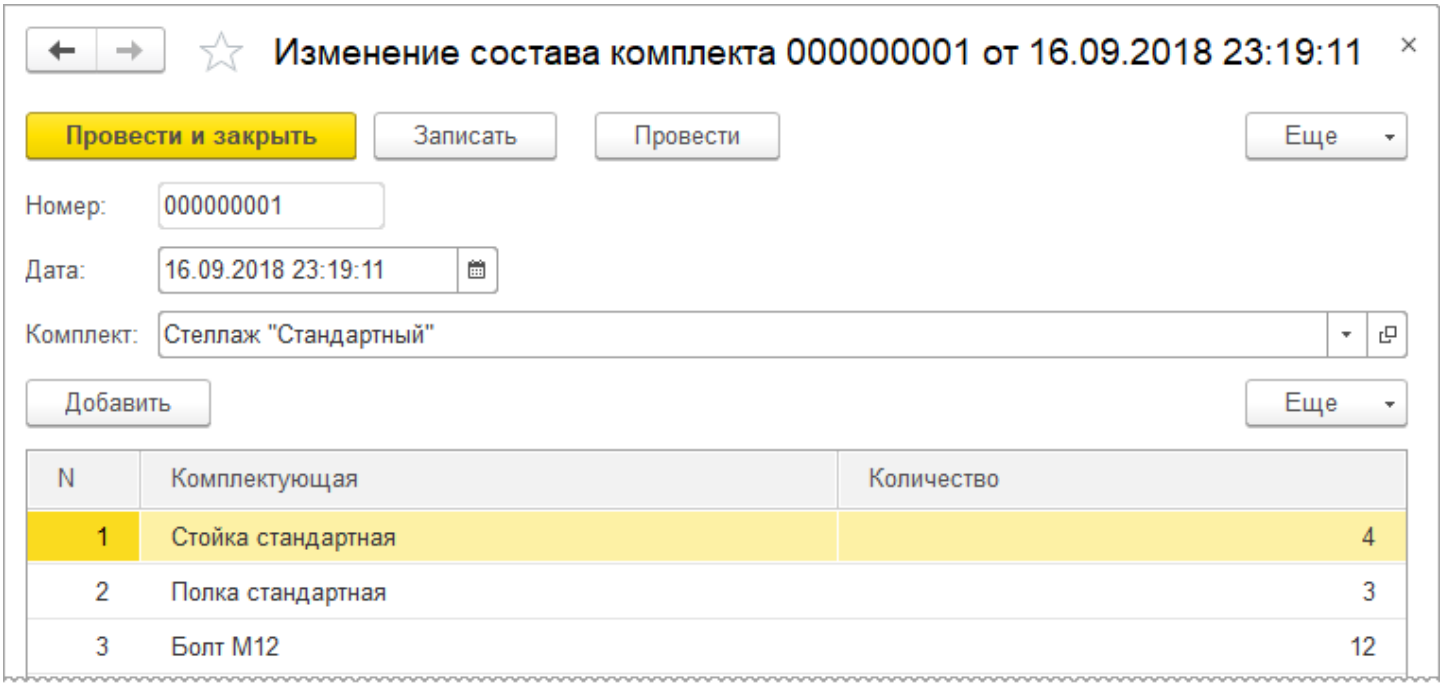


Рисунок 15 – Пример документа «Изменение состава комплекта» в режиме «1С:Предприятие»

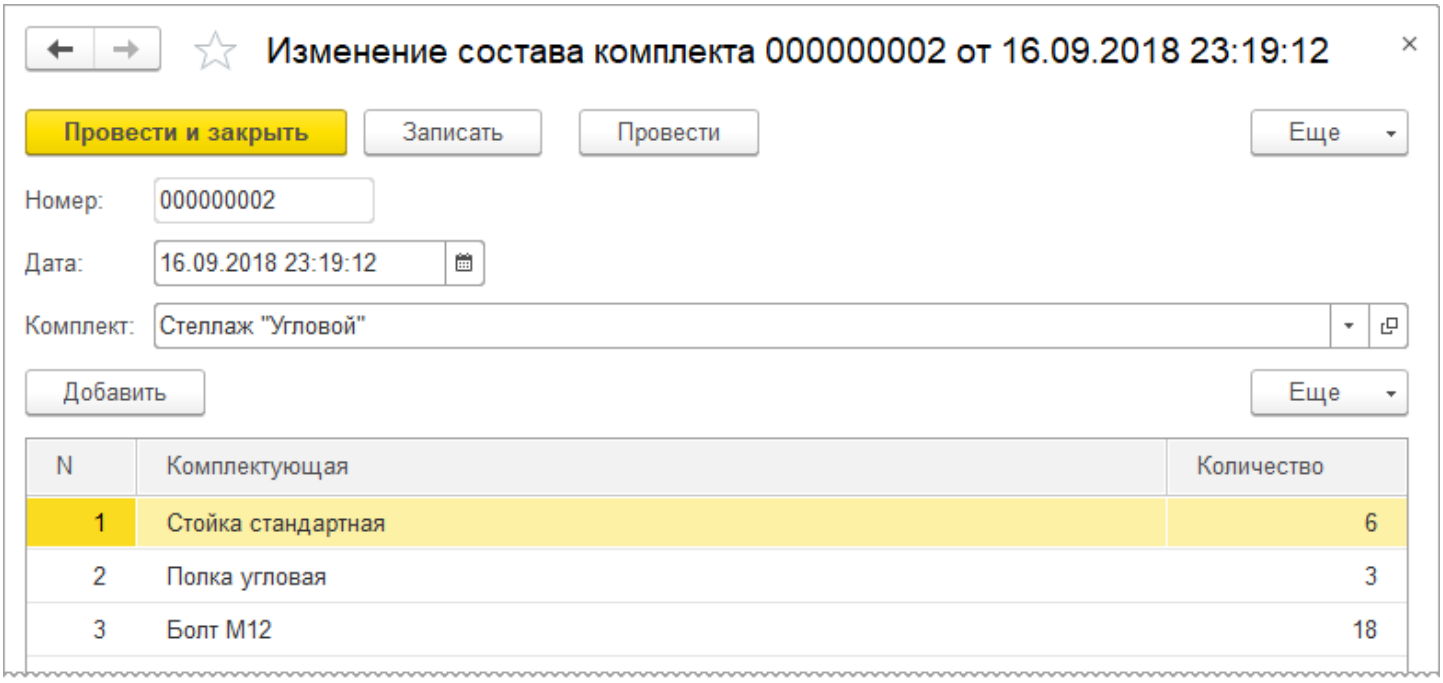


Рисунок 16 – Пример документа «Изменение состава комплекта» в режиме «1С:Предприятие»

3. Проверим результат проведения документов в регистре сведений «Состав комплекта»:

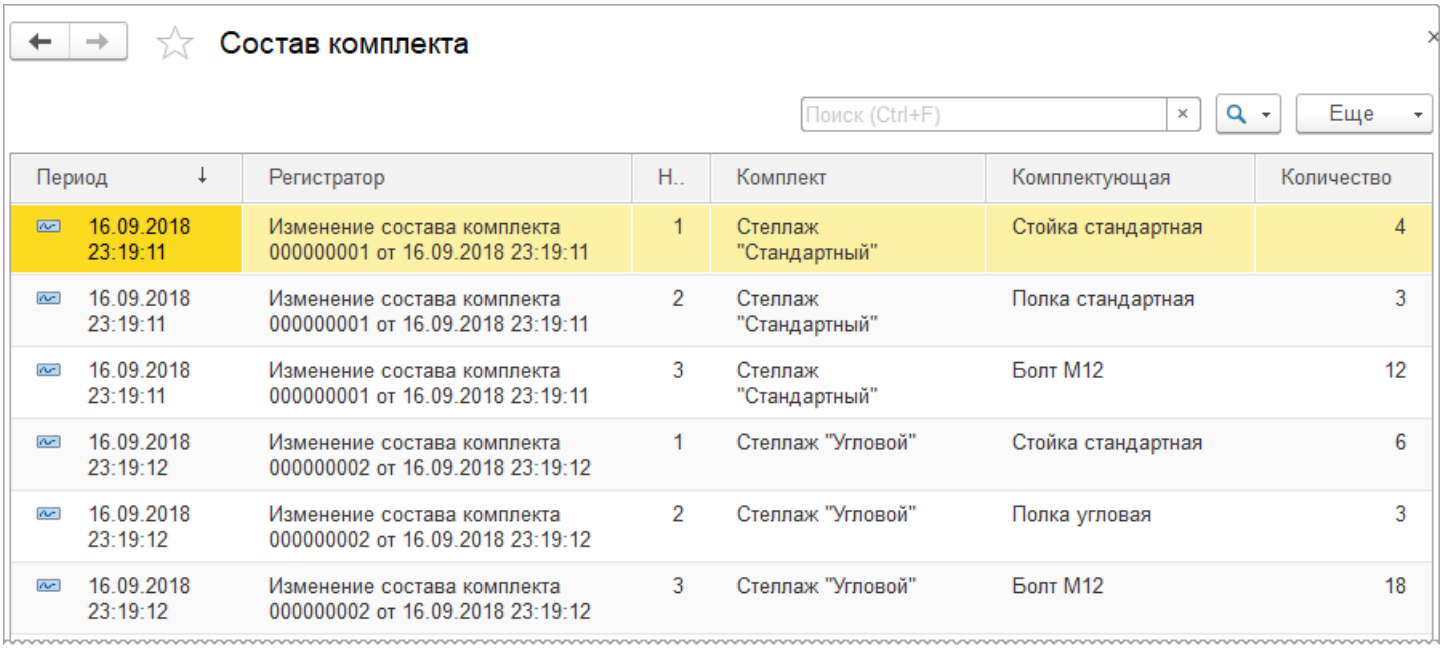


Рисунок 17 – Результаты проведения документов в режиме «1С:Предприятие»

Подведение итогов

В этом разделе мы получили определение операции комплектации, комплекта и комплектующей. Выяснили, как рассчитывается состав комплекта, если в качестве комплектующей указан другой комплект.

Разобрались, как организовать хранение информации о комплектах:

- Для чего нужно сделать деление номенклатуры по видам
- В каком объекте хранить информацию о составе комплекта
- Как настроить объекты конфигурации для выполнения условия задачи, по которому в состав одного комплекта не может входить другой комплект.

На практическом примере реализовали настройку каркасной конфигурации для хранения справочной информации о комплектах.

В следующем блоке приступаем к реализации задачи сборки комплекта.

20. Как выполнить сборку комплекта специальным документом

Процедура создания комплекта из комплектующих называется **сборкой** или **комплектацией**. Преобразование комплектующих в комплект может осуществляться двумя способами:

- **Сборкой** – в момент создания специализированного документа
- **На лету** – в момент продажи комплекта без предварительной сборки.

При любом из этих способов результатом преобразования будет оприходование указанного количества комплекта и списание с остатков расчетного количества комплектующих. При этом себестоимость комплекта равна сумме себестоимостей списанных комплектующих.

Таблица движений при сборке 5 комплектов «Стеллаж «Стандартный» будет выглядеть следующим образом:

Вид движения	Номенклатура	Количество	Себестоимость (руб.)
Расход	Стойка стандартная	20	5 000
Расход	Полка стандартная	15	7 500
Расход	Болт М12	60	30
Приход	Стеллаж «Стандартный»	5	12 530

Процедура сборки комплекта из комплектующих является общей для решения задач блоков оперативного и бухгалтерского учетов. В текущем блоке рассмотрим задачу комплектации в момент создания специализированного документа. Перечислим этапы решения задачи комплектации номенклатуры:

- Необходимо создать специализированный документ «Сборка», оформляющий операцию комплектации
- Необходимо создать процедуру заполнения табличной части документа «Сборка» на основании данных о составе комплекта
- Необходимо проконтролировать наличие достаточного для комплектации количества комплектующих на остатках.

Рассмотрим эти этапы подробно.

Решение практической задачи

Постановка задачи:

Компания занимается сборкой и реализацией комплектов. Для каждого комплекта вводится информация о его составе (какие номенклатурные позиции и в каком количестве входят в данный комплект). В состав комплекта не могут входить другие комплекты.

Операция сборки комплекта оформляется в системе с помощью документа «Сборка». В шапке этого документа указывается комплект и его количество. Фактический перечень и количество комплектующих, потребовавшихся для сборки, указывается в табличной части документа «Сборка» и может отличаться от его справочного состава. Например, деталь из справочного состава может быть заменена на аналогичную. Документ «Сборка» не должен проводиться, если соответствующих комплектующих не оказалось в компании.

Решение первой части задачи, связанной с организацией хранения информации о составе комплекта, приведено в блоке «**19. О каких особенностях операции “Комплектация номенклатуры” нужно знать на экзамене**».

Приведем решение второй части задачи. Для решения нам потребуется информационная база, настроенная в предыдущем блоке **«19. О каких особенностях операции “Комплектация номенклатуры” нужно знать на экзамене**». При решении опустим суммовой учет товаров, будем работать только с количеством.

Для реализации поставленной задачи потребуются следующие объекты конфигурации:

- Регистр накопления «ОстаткиНоменклатуры»
- Регистр сведений «СоставКомплекта»
- Документ «Сборка»
- Документ «ПриходнаяНакладная».

Потребуется выполнить следующие действия:

- Проверить, какие объекты уже имеются в каркасной конфигурации и могут быть использованы для решения поставленной задачи
- Создать недостающие объекты метаданных
- Создать в документе «Сборка» возможность заполнения табличной части справочным составом комплекта
- Сформировать процедуру ОбработкаПроведения для документа «Сборка»:
    - Выполнить контроль остатков комплектующих
    - Сформировать движения

- Проверить результаты в режиме «1С:Предприятие».

Используемые объекты каркасной конфигурации

В каркасной конфигурации уже имеется регистр накопления «ОстаткиНоменклатуры». Структура регистра полностью подходит для решения поставленной задачи. Единственное, что нужно изменить в структуре регистра, – это включить свойство «Запрет незаполненных значений» у измерения «Номенклатура». Включение указанного свойства позволит не писать программно контроль правильного заполнения записей регистра накопления.

Документ «ПриходнаяНакладная» также входит в состав каркасной конфигурации. Структура документа и используемые типы данных удовлетворяют условию задачи. Для экономии времени создавать форму документа не будем.

Регистр сведений «СоставКомплекта» уже был создан в каркасной конфигурации при настройке для блока «**19. О каких особенностях операции “Комплектация номенклатуры” нужно знать на экзамене**».

Обработка проведения для документа «Приходная накладная»

Создание процедуры *ОбработкаПроведения* для документа «ПриходнаяНакладная» подробно описано в блоке «**6. Как реализовать поступление товаров в компанию»**.

Создание документа «Сборка»

С помощью документа «Сборка» в задаче будет оформляться операция изготовления комплекта из комплектующих. Причем по условию задачи фактический список комплектующих, использованных для сборки комплекта, указывается в табличной части документа. Для удобства работы с документом настроим возможность заполнения табличной части списком деталей из регистра сведений «СоставКомплектов».

Добавим в конфигурацию новый документ «Сборка». Структура нового документа практически один в один совпадает со структурой документа «ИзменениеСоставаКомплекта», созданного при решении практической задачи в блоке «**Комплектация номенклатуры – справочная информация**». Скопируем его и изменим имя нового объекта на «Сборка»:

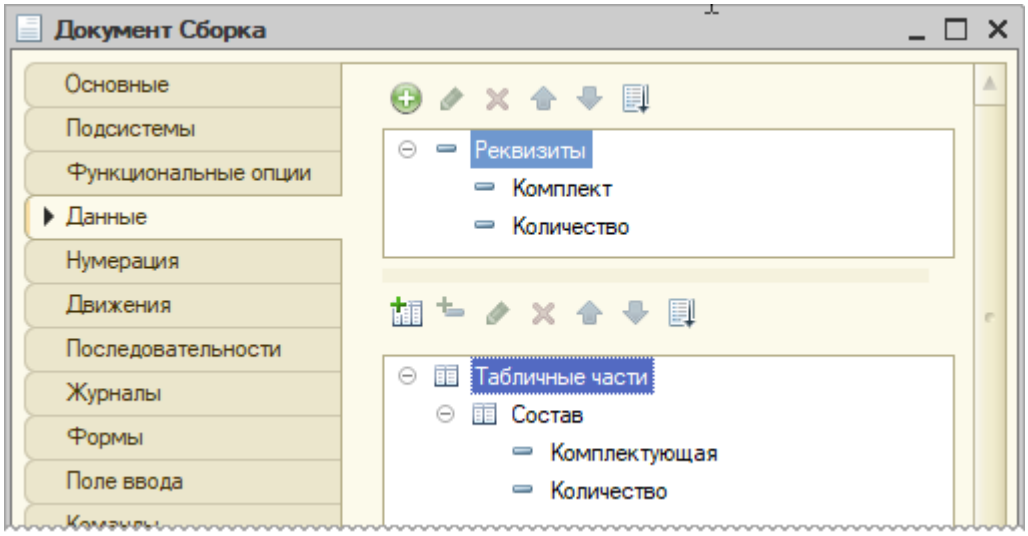


Рисунок 1 – Закладка *Данные* документа «Сборка»

В окне редактирования объекта конфигурации на закладке *Данные* добавим в структуру документа:

- Реквизит «Количество» (Число 10, 0).

Так как для документа «ИзменениеСоставаКомплекта» были настроены параметры выбора для реквизитов «Комплект» и «Комплектующая», то эти настройки будут присутствовать и в документе «Сборка». Таким образом, у нас будет осуществлен контроль условия задачи, чтобы в состав комплекта не был включен другой комплект.

В окне редактирования объекта конфигурации на закладке *Движения* укажем, что документ будет регистратором для регистра накопления «ОстаткиНоменклатуры»:

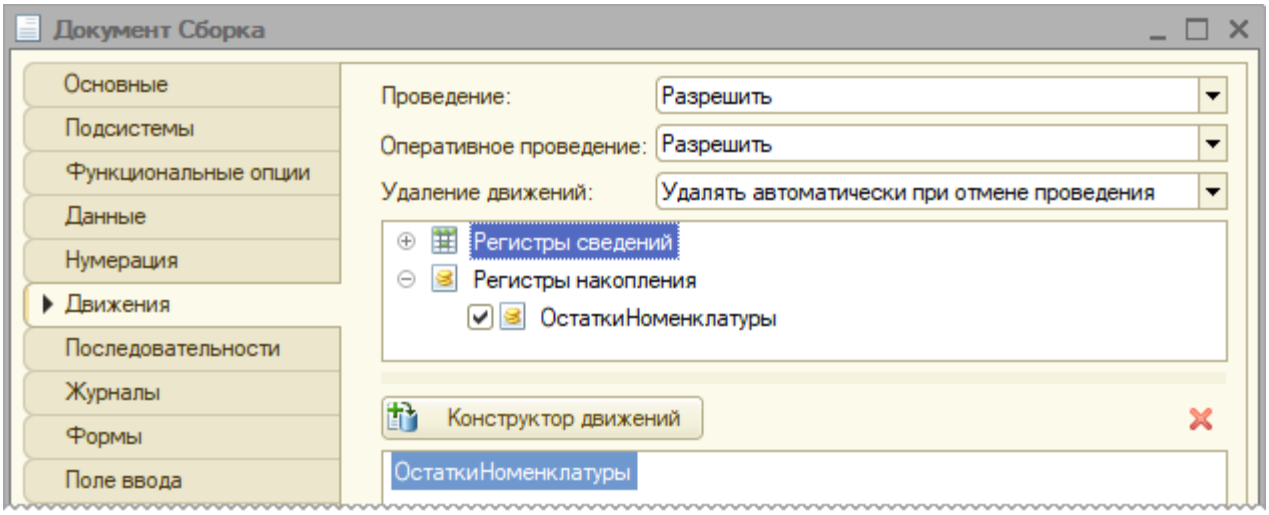


Рисунок 2 – Закладка *Движения* документа «Сборка»

Перейдем на закладку *Формы* и создадим основную форму объекта:

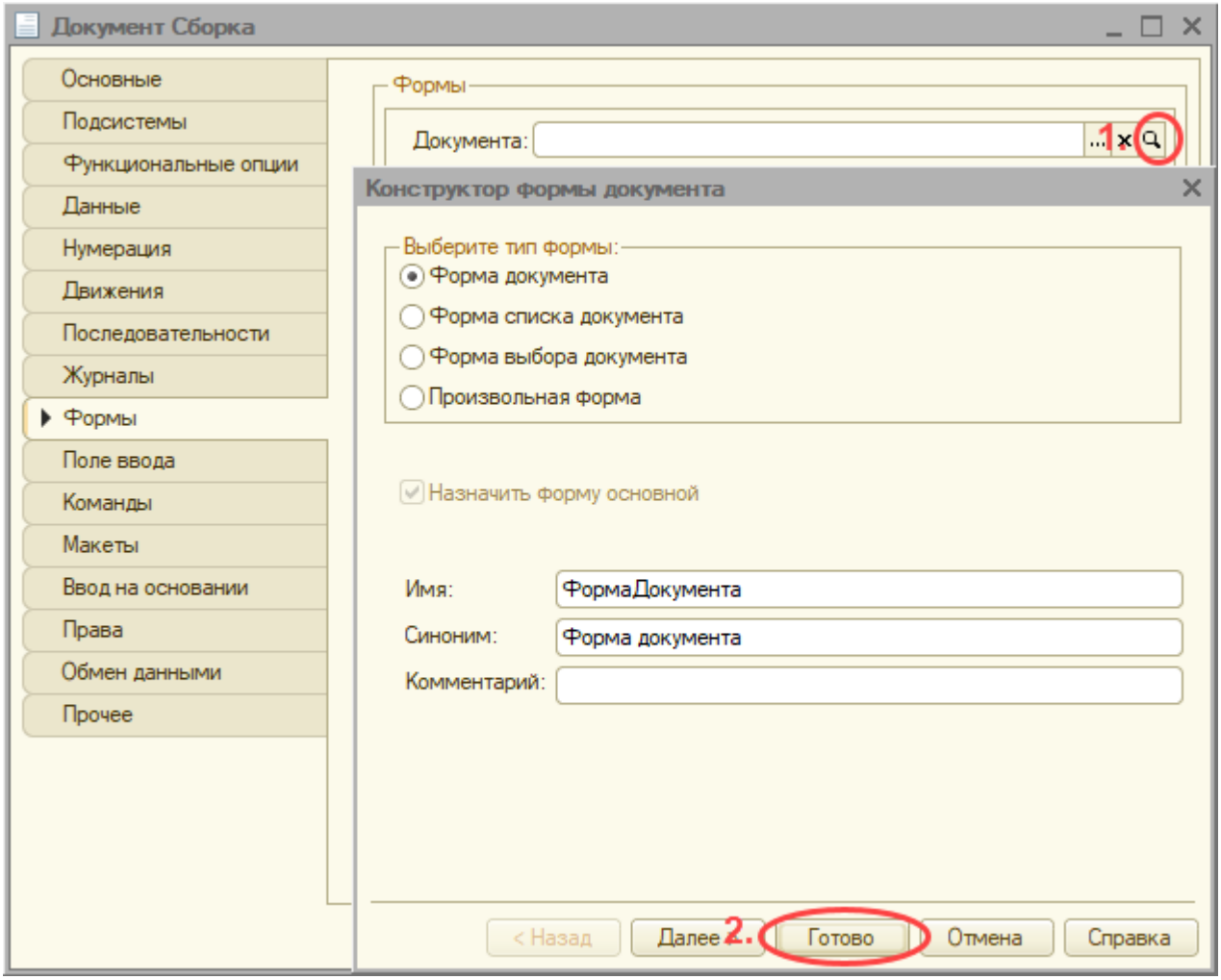


Рисунок 3 – Вызов конструктора формы документа «Сборка»

В открывшейся форме документа переходим на закладку *Команды*, создаем новую команду *Заполнить* и перетаскиваем ее в командную панель табличной части «Состав»:

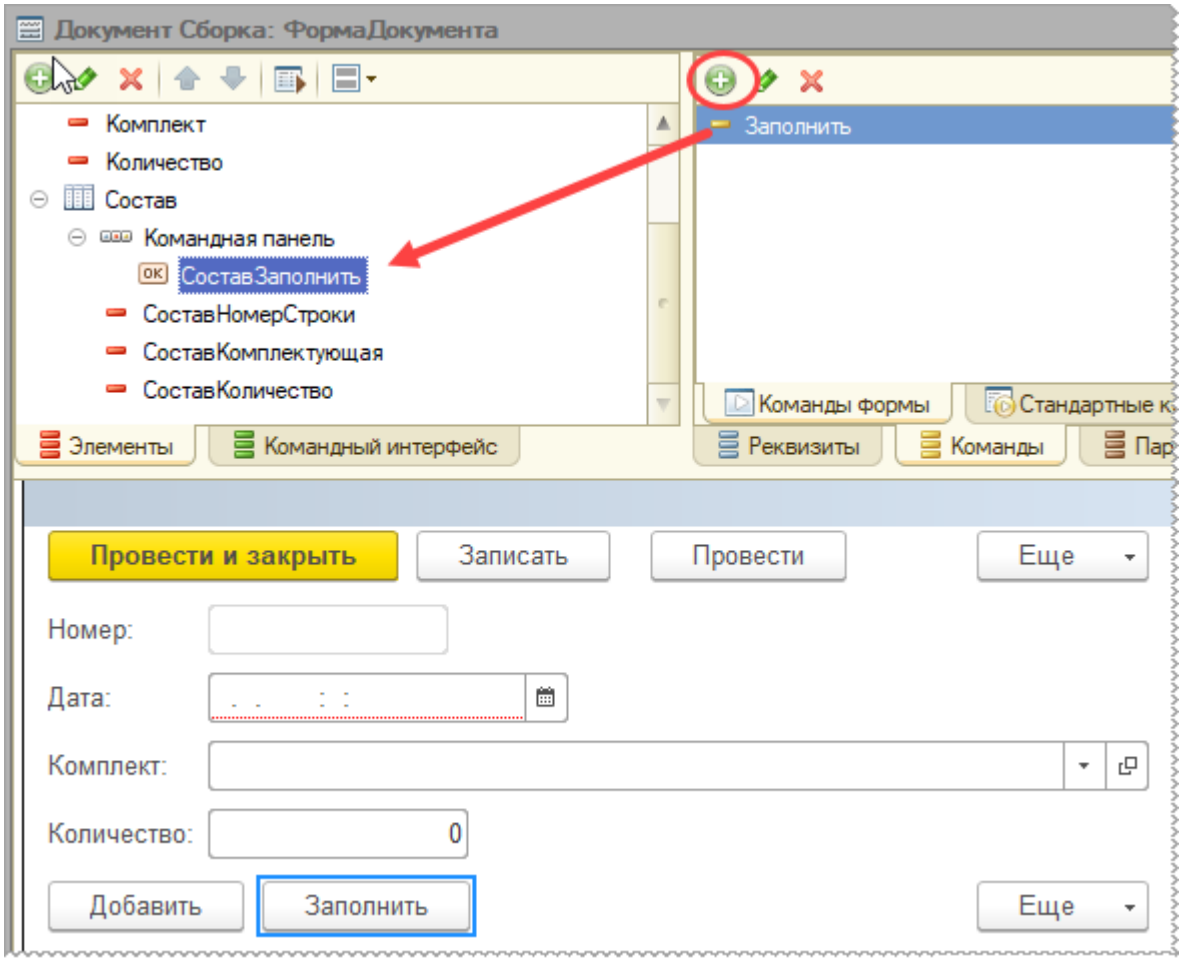


Рисунок 4 – Добавление команды *Заполнить* на форму документа

В окне свойств новой команды создаем обработчик действия. Выбираем вариант *Создать на клиенте и процедуру на сервере*:

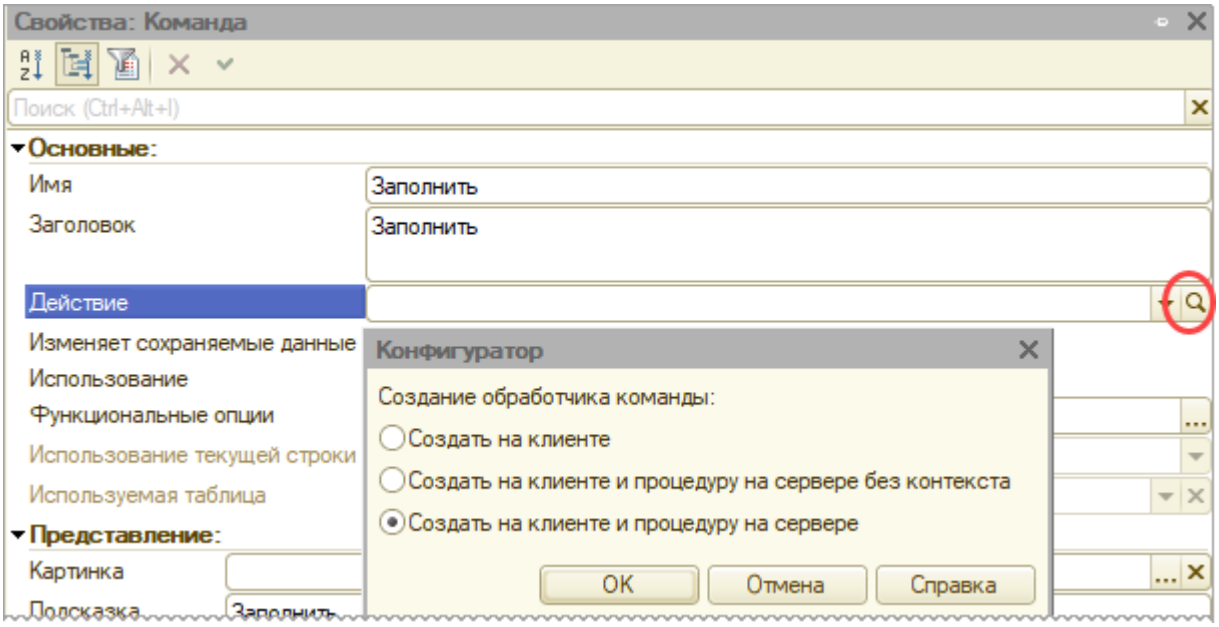


Рисунок 5 – Создание обработчика команды

После нажатия ОК переходим в модуль формы документа и формируем следующий текст процедуры *ЗаполнитьНаСервере*:

```
&НаСервере
Процедура ЗаполнитьНаСервере()

	//1. Формирование запроса к регистру сведений Состав комплекта с отбором по комплекту
	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	СоставКомплектаСрезПоследних.Комплектующая,
		|	СоставКомплектаСрезПоследних.Количество * &КоличествоКомплекта КАК Количество
		|ИЗ
		|	РегистрСведений.СоставКомплекта.СрезПоследних(&ДатаСреза, Комплект = &Комплект)
КАК СоставКомплектаСрезПоследних";

	Запрос.УстановитьПараметр("КоличествоКомплекта", Объект.Количество);
	Запрос.УстановитьПараметр("Комплект", Объект.Комплект);
	Запрос.УстановитьПараметр("ДатаСреза", ?(Не ЗначениеЗаполнено(Объект.Ссылка), Объект.Дата,
Объект.Ссылка.МоментВремени()));



	РезультатЗапроса = Запрос.Выполнить();

	//2. Загрузка таблицы, выгруженной из результата запроса в табличную часть Состав
	Объект.Состав.Загрузить(РезультатЗапроса.Выгрузить());

КонецПроцедуры
```

Прокомментируем основные моменты процедуры *ЗаполнитьНаСервере*.

Запрос (п.1)

Выполняем запрос к виртуальной таблице РегистрСведений.СоставКомплекта.СрезПоследних. В параметрах виртуальной таблицы устанавливаем:

- «ДатаСреза»: если заполняется незаписанный ранее документ, передаем реквизит *Дата*, иначе *МоментВремени* документа
- «Комплект»: значение реквизита «Комплект» из шапки документа.

В запросе сразу вычисляем количество комплектующих, требующееся для выпуска указанного в шапке документа количества комплектов.

Заполнение табличной части (п.2)

Полученную из результата запроса таблицу загружаем в табличную часть «Состав».

Теперь откорректируем процедуру *ОбработкаПроведения* в модуле документа «Сборка»:

```
Процедура ОбработкаПроведения(Отказ, Режим)

	//1. Установка маркера Записи у регистра
	Движения.ОстаткиНоменклатуры.Записывать = Истина;

	//2. Инициализация менеджера временных таблиц
	Запрос = Новый Запрос;
	Запрос.МенеджерВременныхТаблиц = Новый МенеджерВременныхТаблиц;

	//3. Запрос, в котором группируются данные табличной части и подготавливается таблица значений для загрузки в коллекцию движений по регистру
	// регистр ОстаткиНоменклатуры Расход
	Запрос.Текст =
	"ВЫБРАТЬ
	|	СборкаСостав.Комплектующая КАК Комплектующая,
	|	СУММА(СборкаСостав.Количество) КАК Количество
	|ПОМЕСТИТЬ ТЧСостав
	|ИЗ
	|	Документ.Сборка.Состав КАК СборкаСостав
	|ГДЕ
	|	СборкаСостав.Ссылка = &Ссылка
	|
	|СГРУППИРОВАТЬ ПО
	|	СборкаСостав.Комплектующая
	|
```

```
|ИНДЕКСИРОВАТЬ ПО
|	Комплектующая
|;
|
|////////////////////////////////////////////////////////////////////////////////
|ВЫБРАТЬ
|	&Период КАК Период,
|	&ВидДвижения КАК ВидДвижения,
|	ТЧСостав.Комплектующая КАК Номенклатура,
|	ТЧСостав.Количество
|ИЗ
|	ТЧСостав КАК ТЧСостав";

Запрос.УстановитьПараметр("Ссылка", Ссылка);
Запрос.УстановитьПараметр("Период", Дата);
Запрос.УстановитьПараметр("ВидДвижения", ВидДвиженияНакопления.Расход);

РезультатЗапроса = Запрос.Выполнить();

//4. Загрузка таблицы значений в коллекцию движений
Движения.ОстаткиНоменклатуры.Загрузить(РезультатЗапроса.Выгрузить());

//5. Формируем поступление комплекта
Движение = Движения.ОстаткиНоменклатуры.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
Движение.Период = Дата;
Движение.Номенклатура = Комплект;
Движение.Количество = Количество;

//6. Управляемая блокировка данных регистра
Движения.ОстаткиНоменклатуры.БлокироватьДляИзменения = Истина;

//7. Запись коллекции движений
Движения.Записать();

//8. Запрос на получение отрицательных остатков
Запрос.Текст =
"ВЫБРАТЬ
|	ОстаткиНоменклатурыОстатки.Номенклатура КАК Номенклатура,
|	ОстаткиНоменклатурыОстатки.Номенклатура.Представление КАК НоменклатураПредставление,
|	ОстаткиНоменклатурыОстатки.КоличествоОстаток КАК КоличествоОстаток
|ИЗ
|	РегистрНакопления.ОстаткиНоменклатуры.Остатки(
|			&ГраницаОстатков,
|			Номенклатура В
|				(ВЫБРАТЬ
|					ТЧСостав.Комплектующая
|				ИЗ
|					ТЧСостав КАК ТЧСостав)) КАК ОстаткиНоменклатурыОстатки
|ГДЕ
|	ОстаткиНоменклатурыОстатки.КоличествоОстаток < 0";
```

```
	//9. Определение момента времени для получения остатков
	ГраницаОстатков = Новый Граница(МоментВремени(), ВидГраницы.Включая);
	Запрос.УстановитьПараметр("ГраницаОстатков", ГраницаОстатков);

	Результат = Запрос.Выполнить();

	//10. Если запрос не пустой, появились отрицательные остатки
	Если НЕ Результат.Пустой() Тогда
		Отказ = Истина;

		// Выдача сообщений пользователю о наличии отрицательных остатков
		Выборка = Результат.Выбрать();
		Пока Выборка.Следующий() Цикл
			Сообщение = Новый СообщениеПользователю;
			Сообщение.Текст = "Недостаточно комплектующей " +
Выборка.НоменклатураПредставление + ", не хватает: " + (-Выборка.КоличествоОстаток);
			Сообщение.Сообщить();
		КонецЦикла;

	КонецЕсли;

КонецПроцедуры
```

При проведении документа «Сборка» используем новую методику контроля остатков: сначала зафиксируем движения в регистр, после этого проанализируем наличие отрицательных остатков по регистру накопления «ОстаткиНоменклатуры».

Прокомментируем основные моменты кода процедуры *ОбработкаПроведения*.

Установка маркера записи регистра накопления «Остатки номенклатуры» (п.1)

Установим для свойства *Записывать* набора движений по регистру «ОстаткиНоменклатуры» значение *Истина*, чтобы сформированные движения записались в момент вызова метода *Записать()* свойства *Движения*.

Запрос (п. 2, 3)

Запрос получает таблицу «Состав» текущего документа «Сборка». Тот факт, что будут выбраны строки табличной части именно текущего документа, определяется отбором по полю «Ссылка».

Группируем строки с суммированием реквизита «Количество» и помещаем результат во временную таблицу «ТЧСостав».

Временную таблицу индексируем по полю «Комплектующая», потому что в следующих запросах будет выполняться отбор по этому полю.

Так как перед выполнением запроса для него был определен менеджер временных таблиц, временная таблица «ТЧСостав» будет доступна в текстах последующих запросов модуля.

В следующем пакете запроса формируется таблица для загрузки в регистр накопления «ОстаткиНоменклатуры». Таблица движений регистра, кроме колонок измерений «Номенклатура» и «Количество», имеет специальные колонки «Период» и «ВидДвижения», поэтому создадим одноименные поля в тексте запроса. Значения этих полей будут переданы через параметры запроса.

Запись коллекции движений (п. 4, 5, 6, 7)

Загрузим таблицу значений, полученную из результата запроса в таблицу движений регистра накопления. После этого добавим в регистр запись о поступлении комплекта.

Установим для регистра накопления «ОстаткиНоменклатуры» свойство *БлокироватьдляИзменения = Истина*. Подробное описание назначения этого свойства приведено в блоке **«11. Какие навыки в использовании управляемых блокировок потребуются на экзамене»**.

Выполним запись коллекций движений для того, чтобы в регистре накопления «ОстаткиНоменклатуры» изменились остатки. Далее по тексту проанализируем полученные остатки на предмет наличия отрицательного количества.

Запрос на получение отрицательных остатков (п.8,9)

Движения по регистру накопления «ОстаткиНоменклатуры» сформированы и записаны, проверим, появились ли после этого отрицательные остатки.

Для этого сформируем еще один текст запроса.

В качестве источника данных запроса выберем виртуальную таблицу регистра накопления «ОстаткиНоменклатуры.Остатки». Зададим параметры виртуальной таблицы:

- «ГраницаОстатков»: значение параметра = *Граница*
- «Номенклатура»: значение поля должно входить в список, полученный вложенным запросом к временной таблице «ТЧСостав».

В запросе поставим условие по отрицательному значению поля «КоличествоОстаток».

В качестве полей запроса выберем не только значение измерения «Номенклатура», но и текстовое представление этого поля. Текстовое представление необходимо, чтобы не выполнялся дополнительный запрос к базе данных при выводе пользователям сообщений об ошибках.

Обработка ситуации наличия отрицательных остатков (п.10)

В случае, если в регистре существуют отрицательные остатки, второй запрос модуля вернет непустой результат.

Получим выборку из результата запроса, обойдем элементы выборки и выведем сообщение пользователю о нехватке конкретной комплектующей с указанием недостающего количества.

Переменную *Отказ* установим в значение *Истина*, чтобы проведение документе не было выполнено.

Проверка результатов в режиме «1С:Предприятие»

Проверку результатов решения задачи будем выполнять в базе, полученной после решения блока **«Комплектация номенклатуры – справочная информация»**.

1. С помощью документа «Приходная накладная» введем остатки комплектующих:

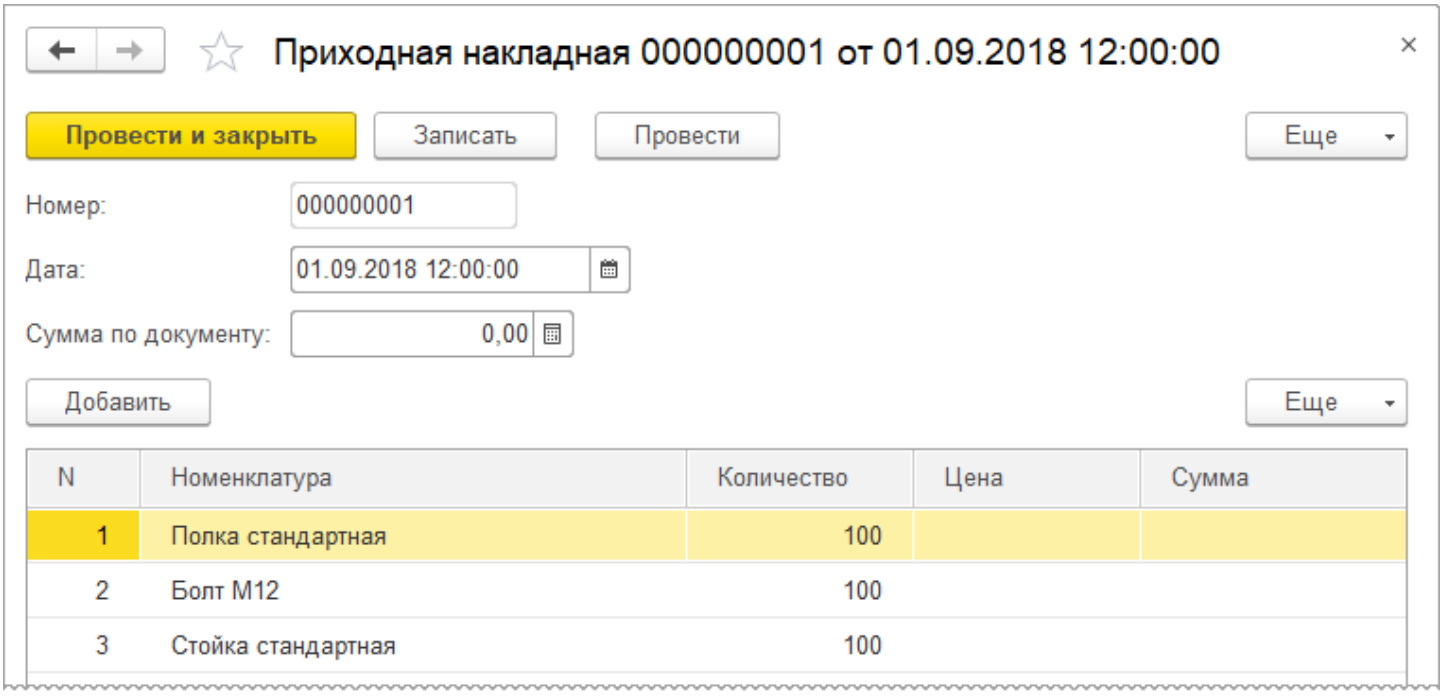


Рисунок 6 – Пример документа «Приходная накладная» в режиме «1С:Предприятие»

2. Создадим документ «Сборка»:

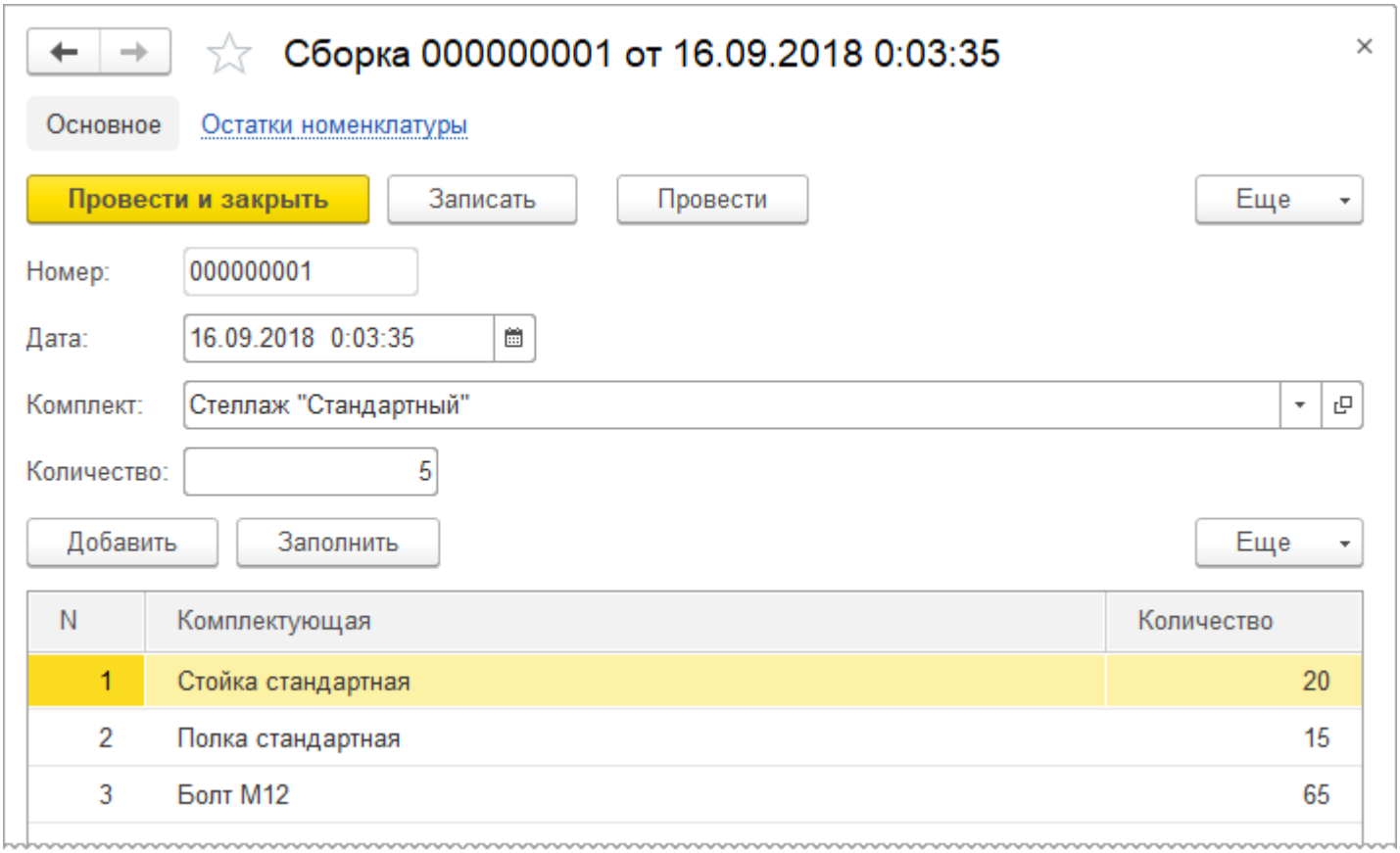
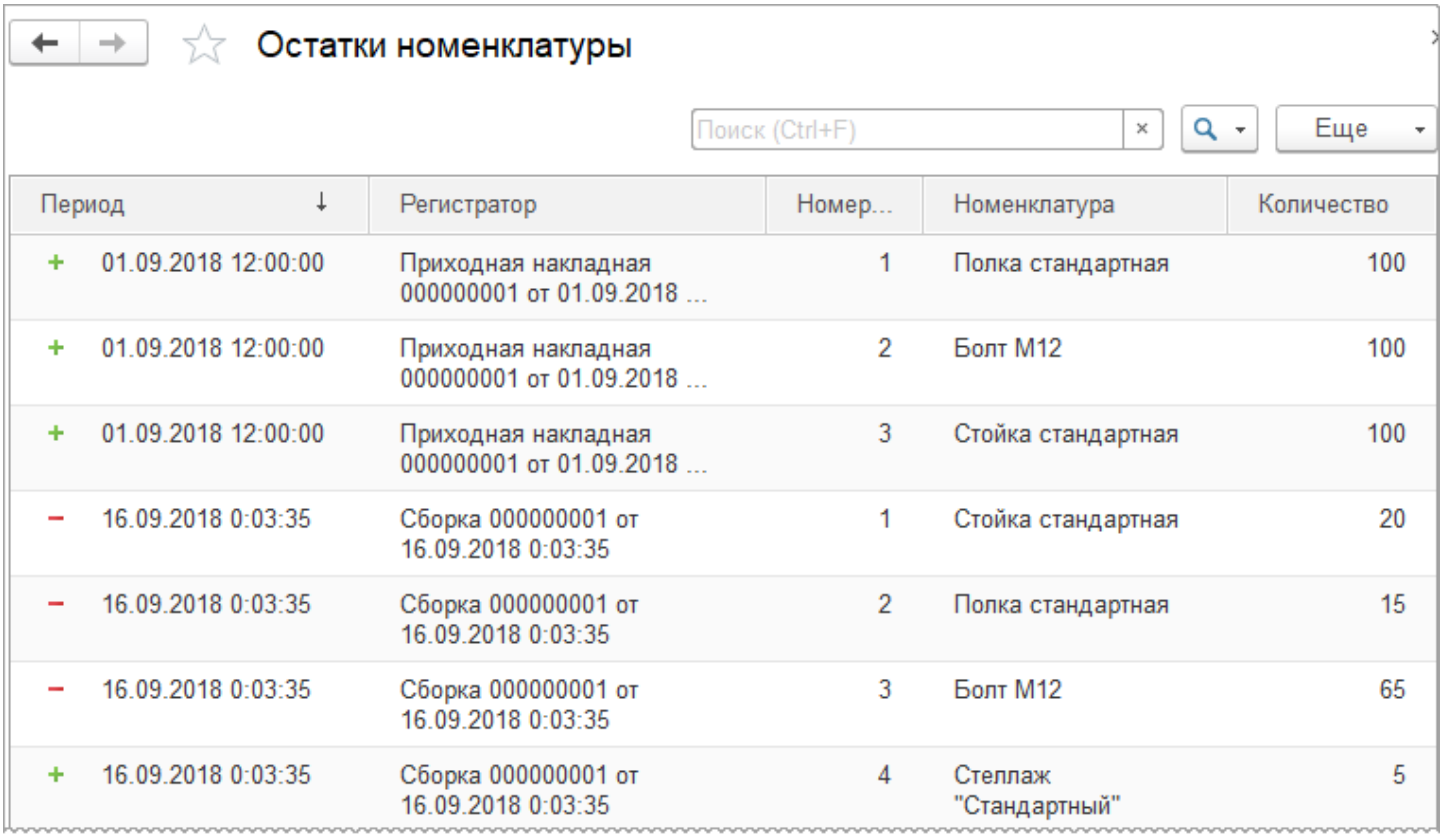


Рисунок 7 – Пример документа «Сборка» в режиме «1С:Предприятие»

3. Результатом ввода всех этих данных будет следующий набор записей в регистре накопления «Остатки номенклатуры»:



Период	Регистратор	Номер...	Номенклатура	Количество
+ 01.09.2018 12:00:00	Приходная накладная 000000001 от 01.09.2018 ...	1	Полка стандартная	100
+ 01.09.2018 12:00:00	Приходная накладная 000000001 от 01.09.2018 ...	2	Болт М12	100
+ 01.09.2018 12:00:00	Приходная накладная 000000001 от 01.09.2018 ...	3	Стойка стандартная	100
− 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	1	Стойка стандартная	20
− 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	2	Полка стандартная	15
− 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	3	Болт М12	65
+ 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	4	Стеллаж "Стандартный"	5

Рисунок 8 – Результаты проведения документов в режиме «1С:Предприятие»

Подведение итогов

В этом разделе мы определили, что такое сборка и какие варианты этой операции могут встретиться в задачах. Узнали, что себестоимость комплекта складывается из себестоимости комплектующих, входящих в него.

На практическом примере настроили документ «Сборка», в котором реализовали:

- Заполнение табличной части данными из регистра сведений «СоставКомплекта»
- Контроль отрицательных остатков комплектующих при проведении документа по «новой» методике.

В следующем разделе рассмотрим задачу сборки комплекта «на лету» – в момент проведения документа «Расходная накладная».

21. Как выполнить сборку комплекта «на лету» – в момент проведения документа продажи

Комплектация «на лету» – это сборка комплекта в момент его продажи, без предварительного оформления документа «Сборка». Особенности такой сборки:

- Расчет количества комплектующих, нужных для создания запрошенного количества комплектов, происходит при проведении документа продажи
- Если по условию задачи в одной табличной части документа продажи могут указываться и комплекты, и комплектующие, расчет количества должен происходить только для тех строк, в которых указаны комплекты
- Если в задаче требуется реализовать операцию комплектации документом «Сборка» и документом продажи одновременно, то перед расчетом комплектующих необходимо определить, нет ли на остатках комплектов, которые указаны в документе продажи. Расчет комплектующих нужен только на то количество комплектов, которого на остатках нет.

Рассмотрим эти особенности подробно при решении практической задачи.

Решение практической задачи

Постановка задачи:

Компания занимается сборкой и реализацией комплектов. Для каждого комплекта вводится информация о составе (какие номенклатурные позиции и в каком количестве входят в данный комплект). В состав комплекта не могут входить другие комплекты.

Операция сборки комплекта оформляется в системе с помощью документа «Сборка». В шапке этого документа указывается комплект и его количество. Фактический перечень и количество комплектующих, потребовавшихся для сборки, указывается в табличной части документа «Сборка» и может отличаться от его справочного состава. Например, деталь из справочного состава может быть заменена на аналогичную. Документ «Сборка» не должен проводиться, если соответствующих комплектующих не оказалось в компании.

Продажи осуществляются документом «Расходная накладная», причем в табличной части могут указываться как комплекты, так и обычные комплектующие. В том случае, если каких-либо комплектов

нет при отгрузке, должна произойти сборка непосредственно при проведении «Расходной накладной». Документ «Расходная накладная» не должен проводиться, если соответствующих комплектующих не оказалось в компании.

Решение первых двух абзацев задачи приведено в блоках «**19. О каких особенностях операции "Комплектация номенклатуры" нужно знать на экзамене**» и «**20. Как выполнить сборку комплекта специальным документом**».

Приведем решение последней части задачи, связанной со сборкой в момент проведения «Расходной накладной». Для решения нам потребуется информационная база, настроенная в предыдущих блоках «**19. О каких особенностях операции "Комплектация номенклатуры" нужно знать на экзамене**» и «**20. Как выполнить сборку комплекта специальным документом**». При решении опустим суммовой учет товаров, будем работать только с количеством.

Для реализации поставленной задачи потребуются следующие объекты конфигурации:

- Регистр накопления «ОстаткиНоменклатуры»
- Регистр сведений «СоставКомплекта»
- Документ «Сборка»
- Документ «ПриходнаяНакладная»
- Документ «РасходнаяНакладная».

Потребуется выполнить следующие действия:

- Проверить, какие объекты уже имеются в каркасной конфигурации и могут быть использованы для решения поставленной задачи
- Создать недостающие объекты метаданных
- Сформировать процедуру ОбработкаПроведения для документа «РасходнаяНакладная»:
    - Определить, какое количество комплектов нужно собрать
    - Определить необходимые для этого комплектующие
    - Выполнить контроль остатков комплектующих
    - Сформировать движения.

Проверить результаты в режиме «1С:Предприятие».

Приступим к реализации задания:

Используемые объекты каркасной конфигурации

В каркасной конфигурации уже имеется регистр накопления «ОстаткиНоменклатуры». Структура регистра полностью настроена для решения поставленной задачи.

Регистр сведений «СоставКомплекта» уже был создан в каркасной конфигурации при настройке для блока «**19. О каких особенностях операции "Комплектация номенклатуры" нужно знать на экзамене**».

Документы «Сборка» и «ПриходнаяНакладная» настроены при решении задачи в блоке «**20. Как выполнить сборку комплекта специальным документом**».

Документ «РасходнаяНакладная» уже входит в состав каркасной конфигурации. Структура документа и используемые типы данных удовлетворяют условию задачи. Для экономии времени создавать форму документа не будем.

Обработка проведения для документа «Расходная накладная»

Документ «РасходнаяНакладная» в задаче используется для реализации как комплектов, так и комплектующих. Если в документе указаны комплекты, которых нет на остатке, то при его проведении должна произойти сборка недостающего количества комплектов.

Создадим процедуру *ОбработкаПроведения* для документа «РасходнаяНакладная». План проведения документа следующий:

- Нужно определить, какого количества комплектов из указанных в табличной части не хватает на остатках
- Разложить недостающие комплекты на комплектующие и определить необходимое количество комплектующих
- Проверить, хватает ли на остатках комплектующих, указанных в табличной части документа, и комплектующих, необходимых для сборки комплектов
- В случае нехватки номенклатуры отменить проведение
- В случае достаточности товаров списать все комплектующие, оприходовать те комплекты, которые собраны в ходе проведения, затем списать все комплекты.

Видим, что уже на первом шаге потребуется запрос к остаткам номенклатуры для определения нехватки комплектов. То есть информация об итогах регистра используется для дальнейших расчетов. Значит, в задаче будем использовать «старую» методику контроля остатков.

Итоговый текст процедуры проведения:

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

	//1. Подготовка наборов записей регистра
	Движения.ОстаткиНоменклатуры.Записывать = Истина;
	Движения.Записать();
```

```
	//2. Восстановление для свойства набора движений Записывать значения Истина
	Движения.ОстаткиНоменклатуры.Записывать = Истина;

	Запрос = Новый Запрос;
	Запрос.МенеджерВременныхТаблиц = Новый МенеджерВременныхТаблиц;

	//3. Запрос, формирующий временные таблицы для контроля остатков
	//и перечень номенклатуры для управляемой блокировки регистра накопления ОстаткиНоменлатуры
	Запрос.Текст =
	"ВЫБРАТЬ
	|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура КАК Номенклатура,
	|	ВЫБОР
	|		КОГДА РасходнаяНакладнаяСписокНоменклатуры.Номенклатура.ВидНоменклатуры =
ЗНАЧЕНИЕ(Перечисление.ВидыНоменклатуры.Комплект)
	|			ТОГДА ИСТИНА
	|		ИНАЧЕ ЛОЖЬ
	|	КОНЕЦ КАК ЭтоКомплект,
	|	СУММА(РасходнаяНакладнаяСписокНоменклатуры.Количество) КАК Количество
	|ПОМЕСТИТЬ ТЧСписокНоменклатуры
	|ИЗ
	|	Документ.РасходнаяНакладная.СписокНоменклатуры КАК
РасходнаяНакладнаяСписокНоменклатуры
	|ГДЕ
	|	РасходнаяНакладнаяСписокНоменклатуры.Ссылка = &Ссылка
	|	И НЕ РасходнаяНакладнаяСписокНоменклатуры.Номенклатура.ВидНоменклатуры =
ЗНАЧЕНИЕ(Перечисление.ВидыНоменклатуры.Услуга)
	|
	|СГРУППИРОВАТЬ ПО
	|	РасходнаяНакладнаяСписокНоменклатуры.Номенклатура,
	|	ВЫБОР
	|		КОГДА РасходнаяНакладнаяСписокНоменклатуры.Номенклатура.ВидНоменклатуры =
ЗНАЧЕНИЕ(Перечисление.ВидыНоменклатуры.Комплект)
	|			ТОГДА ИСТИНА
	|		ИНАЧЕ ЛОЖЬ
	|	КОНЕЦ
	|
	|ИНДЕКСИРОВАТЬ ПО
	|	Номенклатура
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	СоставКомплектаСрезПоследних.Комплектующая,
	|	СоставКомплектаСрезПоследних.Комплект,
	|	СоставКомплектаСрезПоследних.Количество
	|ПОМЕСТИТЬ СоставКомплектов
	|ИЗ
	|	РегистрСведений.СоставКомплекта.СрезПоследних(
	|			&МоментВремени,
	|			Комплект В
	|				(ВЫБРАТЬ
	|					ТЧСписокНоменклатуры.Номенклатура
	|				ИЗ
	|					ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|				ГДЕ
```

```
	|                                        ТЧСписокНоменклатуры.ЭтоКомплект)) КАК
СоставКомплектаСрезПоследних
	|
	|ИНДЕКСИРОВАТЬ ПО
	|	СоставКомплектаСрезПоследних.Комплект
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	ТЧСписокНоменклатуры.Номенклатура
	|ИЗ
	|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|
	|ОБЪЕДИНИТЬ
	|
	|ВЫБРАТЬ
	|	СоставКомплектов.Комплектующая
	|ИЗ
	|	СоставКомплектов КАК СоставКомплектов";

	//4. Установка параметров запроса

	Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
	Запрос.УстановитьПараметр("Ссылка", Ссылка);

	РезультатНоменклатура = Запрос.Выполнить();

	//5. Управляемая блокировка данных регистра
	Блокировка = Новый БлокировкаДанных;
	ЭлементБлокировки = Блокировка.Добавить("РегистрНакопления.ОстаткиНоменклатуры");
	ЭлементБлокировки.Режим = РежимБлокировкиДанных.Исключительный;
	ЭлементБлокировки.ИсточникДанных = РезультатНоменклатура;
	ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Номенклатура", "Номенклатура");
	Блокировка.Заблокировать();

	//6. Запрос для контроля отрицательных остатков
	//и формирования таблицы движений
	Запрос.Текст =
	"ВЫБРАТЬ
	|	ТЧСписокНоменклатуры.Номенклатура КАК Комплект,
	|	ТЧСписокНоменклатуры.Количество -
ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) КАК Дефицит
	|ПОМЕСТИТЬ ДефицитКомплектов
	|ИЗ
	|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиНоменклатуры.Остатки(
	|				&МоментВремени,
	|				Номенклатура В
	|					(ВЫБРАТЬ
	|						ТЧСписокНоменклатуры.Номенклатура
	|					ИЗ
	|						ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|					ГДЕ
	|						ТЧСписокНоменклатуры.ЭтоКомплект)) КАК
ОстаткиНоменклатурыОстатки
```

```
	|		ПО ТЧСписокНоменклатуры.Номенклатура = ОстаткиНоменклатурыОстатки.Номенклатура
	|ГДЕ
	|	ТЧСписокНоменклатуры.ЭтоКомплект
	|	И ТЧСписокНоменклатуры.Количество -
ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) > 0
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	СоставКомплектов.Комплектующая КАК Комплектующая,
	|	СоставКомплектов.Количество * ДефицитКомплектов.Дефицит КАК КоличествоКомплектующей
	|ПОМЕСТИТЬ ВсеКомплектующиеИТовары
	|ИЗ
	|	ДефицитКомплектов КАК ДефицитКомплектов
	|		ВНУТРЕННЕЕ СОЕДИНЕНИЕ СоставКомплектов КАК СоставКомплектов
	|		ПО ДефицитКомплектов.Комплект = СоставКомплектов.Комплект
	|
	|ОБЪЕДИНИТЬ ВСЕ
	|
	|ВЫБРАТЬ
	|	ТЧСписокНоменклатуры.Номенклатура,
	|	ТЧСписокНоменклатуры.Количество
	|ИЗ
	|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|ГДЕ
	|	НЕ ТЧСписокНоменклатуры.ЭтоКомплект
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	ВсеКомплектующиеИТовары.Комплектующая,
	|	СУММА(ВсеКомплектующиеИТовары.КоличествоКомплектующей) КАК КоличествоКомплектующей
	|ПОМЕСТИТЬ СписываемыеКомплектующиеИТовары
	|ИЗ
	|	ВсеКомплектующиеИТовары КАК ВсеКомплектующиеИТовары
	|
	|СГРУППИРОВАТЬ ПО
	|	ВсеКомплектующиеИТовары.Комплектующая
	|
	|ИНДЕКСИРОВАТЬ ПО
	|	ВсеКомплектующиеИТовары.Комплектующая
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	СписываемыеКомплектующиеИТовары.Комплектующая КАК Комплектующая,
	|	&КомплектующаяПредставление КАК КомплектующаяПредставление,
	|	СписываемыеКомплектующиеИТовары.КоличествоКомплектующей -
ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) КАК ДефицитКомплектующей
	|ИЗ
	|	СписываемыеКомплектующиеИТовары КАК СписываемыеКомплектующиеИТовары
	|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиНоменклатуры.Остатки(
	|				&МоментВремени,
	|				Номенклатура В
	|					(ВЫБРАТЬ
```

```
	|                                                СписываемыеКомплектующиеИТовары.Комплектующая
	|                                            ИЗ
	|                                                СписываемыеКомплектующиеИТовары КАК
СписываемыеКомплектующиеИТовары)) КАК ОстаткиНоменклатурыОстатки
	|                    ПО СписываемыеКомплектующиеИТовары.Комплектующая =
ОстаткиНоменклатурыОстатки.Номенклатура
	|ГДЕ
	|	СписываемыеКомплектующиеИТовары.КоличествоКомплектующей -
ЕСТЬNULL(ОстаткиНоменклатурыОстатки.КоличествоОстаток, 0) > 0
	|;
	|
	|////////////////////////////////////////////////////////////////////////////////
	|ВЫБРАТЬ
	|	СписываемыеКомплектующиеИТовары.Комплектующая КАК Номенклатура,
	|	СписываемыеКомплектующиеИТовары.КоличествоКомплектующей КАК Количество,
	|	ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход) КАК ВидДвижения,
	|	&Период
	|ИЗ
	|	СписываемыеКомплектующиеИТовары КАК СписываемыеКомплектующиеИТовары
	|
	|ОБЪЕДИНИТЬ ВСЕ
	|
	|ВЫБРАТЬ
	|	ДефицитКомплектов.Комплект,
	|	ДефицитКомплектов.Дефицит,
	|	ЗНАЧЕНИЕ(ВидДвиженияНакопления.Приход),
	|	&Период
	|ИЗ
	|	ДефицитКомплектов КАК ДефицитКомплектов
	|
	|ОБЪЕДИНИТЬ ВСЕ
	|
	|ВЫБРАТЬ
	|	ТЧСписокНоменклатуры.Номенклатура,
	|	ТЧСписокНоменклатуры.Количество,
	|	ЗНАЧЕНИЕ(ВидДвиженияНакопления.Расход),
	|	&Период
	|ИЗ
	|	ТЧСписокНоменклатуры КАК ТЧСписокНоменклатуры
	|ГДЕ
	|	ТЧСписокНоменклатуры.Комплект";

	Запрос.Текст = СтрЗаменить(Запрос.Текст, "&КомплектующаяПредставление",
"СписываемыеКомплектующиеИТовары.Комплектующая.Представление");

	Запрос.УстановитьПараметр("Период", Дата);

	ПакетРезультатов = Запрос.ВыполнитьПакет();

	//7. Если есть отрицательные остатки
	//в четвертом пакете результатов (индекс четвертого пакета = 3) будут записи
	//обработка ситуации наличия отрицательных остатков
	Если НЕ ПакетРезультатов[3].Пустой() Тогда
		Отказ = Истина;
```

```
                // Выдача сообщений пользователю о наличии отрицательных остатков
                Выборка = ПакетРезультатов[3].Выбрать();
                Пока Выборка.Следующий() Цикл
                        Сообщение = Новый СообщениеПользователю;
                        Сообщение.Текст = "Недостаточно комплектующей " +
Выборка.КомплектующаяПредставление + ", не хватает: " + Выборка.ДефицитКомплектующей;
                        Сообщение.Сообщить();
                КонецЦикла;
        КонецЕсли;

        Если Отказ Тогда
                Возврат;
        КонецЕсли;

        //8. Загрузка таблицы из результата запроса в набор записей и запись коллекции движений
        Движения.ОстаткиНоменклатуры.Загрузить(ПакетРезультатов[4].Выгрузить());

        Движения.Записать();

КонецПроцедуры
```

Прокомментируем основные моменты кода процедуры *ОбработкаПроведения*.:

Подготовка наборов записей регистра «Остатки номенклатуры» (п.1,2)

Выполним очистку существующих движений документа, принудительно записав пустой набор в регистр накопления «ОстаткиНоменклатуры». После очистки установим для свойства *Записывать* набора движений по регистру «ОстаткиНоменклатуры» значение *Истина*.

Запрос, формирующий источник данных для блокировки (п.3,4)

В первом запросе получаем временные таблицы, требующиеся для проведения документа. Так как эти таблицы будут использоваться в следующем запросе, то предварительно устанавливаем менеджер временных таблиц.

Цель формирования первого запроса: получение списка номенклатуры, которую нужно передать в источник данных управляемой блокировки. Цель наложения управляемой блокировки: минимизировать время и ресурсы, заблокированные на чтение, поэтому накладывать блокировку на остатки всей номенклатуры будет ошибкой.

Чтобы получить минимальный, но достаточный перечень номенклатуры для блокировки, в первом пакете запроса получим все позиции, указанные в табличной части «СписокНоменклатуры» документа.

Но по условию задачи в документе могут быть комплекты, которые надо собрать из комплектующих, то есть в список номенклатуры для блокировки должны попасть и эти комплектующие. Однако на

текущем этапе проведения мы не знаем, какие комплекты нужно будет собирать, а какие есть на остатках. Поэтому разложим на детали все комплекты, указанные в табличной части «СписокНоменклатуры» документа. Сделаем это во втором пакете запроса.

В последнем, третьем, пакете запроса объединим информацию, полученную в первых двух пакетах. Результат этого запроса можно будет использовать как источник данных управляемой блокировки.

Рассмотрим особенности формирования каждого из пакетов подробнее:

1. В первом пакете получаем содержимое табличной части «СписокНоменклатуры» документа. Строки табличной части группируем по полю «Номенклатура», «Количество» при этом суммируем. Для определения вида номенклатуры добавляем вычисляемое поле «ЭтоКомплект», в которое помещаем *Истина*, если реквизит «ВидНоменклатуры» у элемента номенклатуры имеет значение «Комплект». Устанавливаем отбор в запросе по ссылке на текущий документ. Включаем индексирование по полю «Номенклатура» для последующих соединений с другими таблицами
2. Во втором пакете получаем данные виртуальной таблицы *СрезПоследних* регистра сведений «СоставКомплекта». Для виртуальной таблицы устанавливаем параметры:
    - «МоментВремени»: значение параметра = *МоментВремени*
    - «Комплект»: значение поля должно входить в список, полученный вложенным запросом к временной таблице «ТЧСписокНоменклатуры» с отбором по полю «ЭтоКомплект» = *Истина*. То есть во вложенном запросе нас интересуют только позиции, являющиеся комплектами

3. В третьем пакете объединяем данные двух временных таблиц из первого и второго пакетов с тем, чтобы получить единый список номенклатуры. Чтобы исключить повторяющиеся позиции, объединение выполним с помощью ключевого слова *ОБЪЕДИНИТЬ* без ключевого слова *ВСЕ*. В конструкторе запроса такая конструкция настраивается следующим образом:

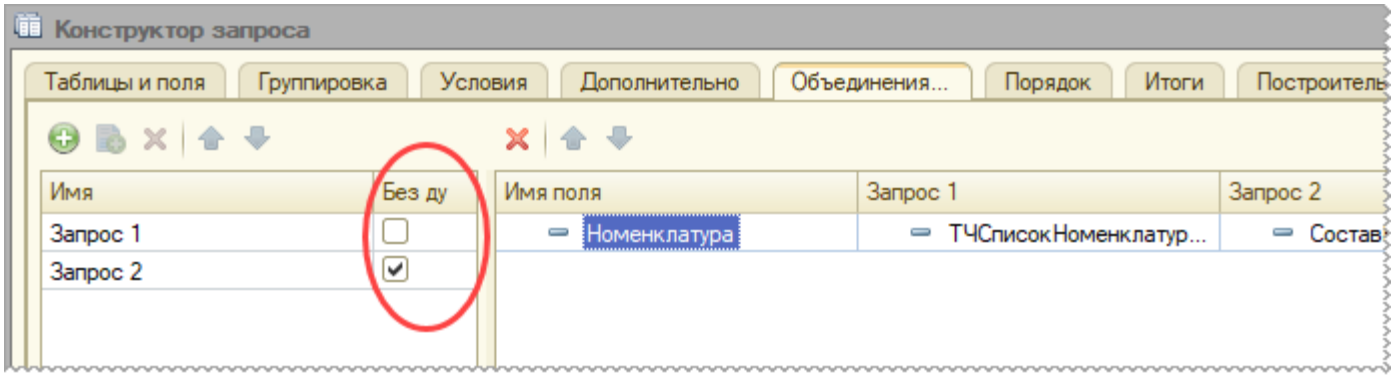


Рисунок 1 – Закладка *Объединения, псевдонимы* конструктора запросов

Установка управляемой блокировки (п.5)

При использовании старой методики контроля остатков нужно наложить управляемую блокировку до выполнения запроса получения данных для расчета. В качестве источника блокировки передаем результат запроса, полученного пунктом ранее.

Запрос, выполняющий контроль остатков и формирующий таблицу движений регистра (п.6)

Во втором запросе процедуры *ОбработкаПроведения* получим информацию о недостающих комплектующих и сформируем таблицу для загрузки в набор записей регистра накопления.

В первом пакете запроса определяем дефицит комплектов. Для этого соединяем временную таблицу «ТЧСписокНоменклатуры», отобранную по полю «ЭтоКомплект» = *Истина*, с виртуальной таблицей регистра накопления «ОстаткиНоменклатуры.Остатки». Для виртуальной таблицы устанавливаем параметры:

- «МоментВремени»: значение параметра = *МоментВремени*
- «Номенклатура»: значение поля должно входить в список, полученный вложенным запросом к временной таблице «ТЧСписокНоменклатуры» с отбором по полю «ЭтоКомплект» = *Истина*. То есть во вложенном запросе нас интересуют только позиции, являющиеся комплектами.

Выбираем во временную таблицу «ДефицитКомплектов» только те записи, в которых поле «Количество», полученное из табличной части документа, больше поля «КоличествоОстаток», полученного из остатков регистра накопления.

Во втором пакете запроса получаем все комплектующие и товары, которые надо списать при проведении. Это два блока данных:

- Комплектующие, необходимые для выпуска дефицита комплектов
- Товары, указанные непосредственно в табличной части «СписокНоменклатуры» документа.

Комплектующие, необходимые для выпуска дефицита комплектов, получим, соединив виртуальные таблицы «ДефицитКомплектов» и «СоставКомплектов». Напомним, что временная таблица «СоставКомплектов» была получена в первом запросе процедуры. Перемножив поля «Дефицит», в котором содержится количество комплектов к сборке, и «Количество», в котором хранится количество деталей на один комплект, получаем количество комплектующих, необходимое для сборки дефицита комплектов.

Комплектующие, указанные непосредственно в табличной части «СписокНоменклатуры» документа, получаем из временной таблицы «ТЧСписокНоменклатуры» с отбором по полю «ЭтоКомплект» = *Ложь*.

Результат помещаем во временную таблицу «ВсеКомплектующиеИТовары».

В третьем пакете запроса выбираем временную таблицу «ВсеКомплектующиеИТовары» и группируем ее по полю «Комплектующая», суммируя поле «КоличествоКомплектующей». Результат помещаем во временную таблицу «СписываемыеКомплектующиеИТовары».

В четвертом пакете выполняем проверку списываемых комплектующих на наличие в остатках. Для этого соединяем временную таблицу «СписываемыеКомплектующиеИТовары» с виртуальной таблицей регистра накопления «ОстаткиНоменклатуры.Остатки». Для виртуальной таблицы устанавливаем параметры:

- «МоментВремени»: значение параметра = *МоментВремени*
- «Номенклатура»: значение поля должно входить в список, полученный вложенным запросом к временной таблице «СписываемыеКомплектующиеИТовары».

Устанавливаем условие, по которому в результат пакета попадут только те записи, поле «КоличествоКомплектующей» в которых больше поля «КоличествоОстаток» из таблицы регистра накопления.

Таким образом, если в данном пакете будет хотя бы одна запись, контроль отрицательных остатков не выполнен, нужно будет отменять проведение документа.

При наличии отрицательных остатков собираемся выдавать пользователю сообщение об ошибке. В текстовое сообщение об ошибке нужно выводить представление ссылки вместо самой ссылки, чтобы избежать обращения к информационной базе в цикле вывода сообщений.

К сожалению, конструктор запросов не даст добавить в пакет поле «СписываемыеКомплектующиеИТовары.Комплектующая.Представление», потому что временная таблица, из которой получены данные, была определена в предыдущем запросе. Чтобы не терять возможность работы с конструктором запроса и добавить нужное поле, добавим произвольное выражение «&КомплектующаяПредставление КАК КомплектующаяПредставление,».

После формирования текста запроса заменим строку «&КомплектующаяПредставление» на «СписываемыеКомплектующиеИТовары.Комплектующая.Представление». Таким образом мы сохраним работоспособность конструктора запросов и получим нужные данные после выполнения запроса.

В пятом, последнем, пакете формируем таблицу для загрузки в набор записей регистра накопления «ОстаткиНоменклатуры». Здесь объединим данные из трех источников:

- Временная таблица «СписываемыеКомплектующиеИТовары». Для этих записей должно быть установлено свойство *ВидДвижения = ВидДвиженияНакопления.Расход*

- Временная таблица «ДефицитКомплектов». Для этих записей должно быть установлено свойство *ВидДвижения* = *ВидДвиженияНакопления.Приход,* так как комплекты были собраны при проведении документа

- Временная таблица «ТЧСписокНоменклатуры» с отбором по полю «ЭтоКомплект» = *Истина*. Для этих записей должно быть установлено свойство *ВидДвижения* = *ВидДвиженияНакопления.Расход*, то есть списываем комплекты, указанные в табличной части документа.

Во всех таблицах добавляем поле «Период» для заполнения такого же поля в наборе записей регистра.

Запрос готов.

Обработка ситуации наличия отрицательных остатков (п.7)

Запрос, формирующий таблицу контроля отрицательных остатков, состоит из нескольких пакетов, поэтому выполнять его будем с помощью метода *ВыполнитьПакет*. Этот метод возвращает массив результатов запроса. В массив входят результаты всех пакетов запроса, содержащие как обычные таблицы, так и временные. Поэтому, чтобы получить информацию о наличии отрицательных остатков, нужно выбрать третий элемент массива результатов (элементы массива нумеруются с 0).

В случае, если в регистре существуют отрицательные остатки, выбранный результат будет непустой.

Получим выборку из результата запроса, обойдем элементы выборки и выведем сообщение пользователю о нехватке конкретной комплектующей с указанием недостающего количества.

Переменную *Отказ* установим в значение *Истина*, чтобы проведение документе не было выполнено, и прервем выполнение процедуры.

Запись коллекции движений (п.8)

Когда контроль отрицательных остатков на предыдущем шаге пройден, получим таблицу движений из четвертого элемента массива результатов запроса. Загрузим таблицу в набор записей регистра накопления и зафиксируем коллекцию движений.

Проверка результатов в режиме «1С:Предприятие»

1. Для проверки результатов в режиме «1С:Предприятие» будем использовать базу, заполненную в блоках «**19. О каких особенностях операции “Комплектация номенклатуры” нужно знать на**

экзамене» и «**20. Как выполнить сборку комплекта специальным документом**».. Введем документ «Расходная накладная» и проверим работу механизма контроля остатков:

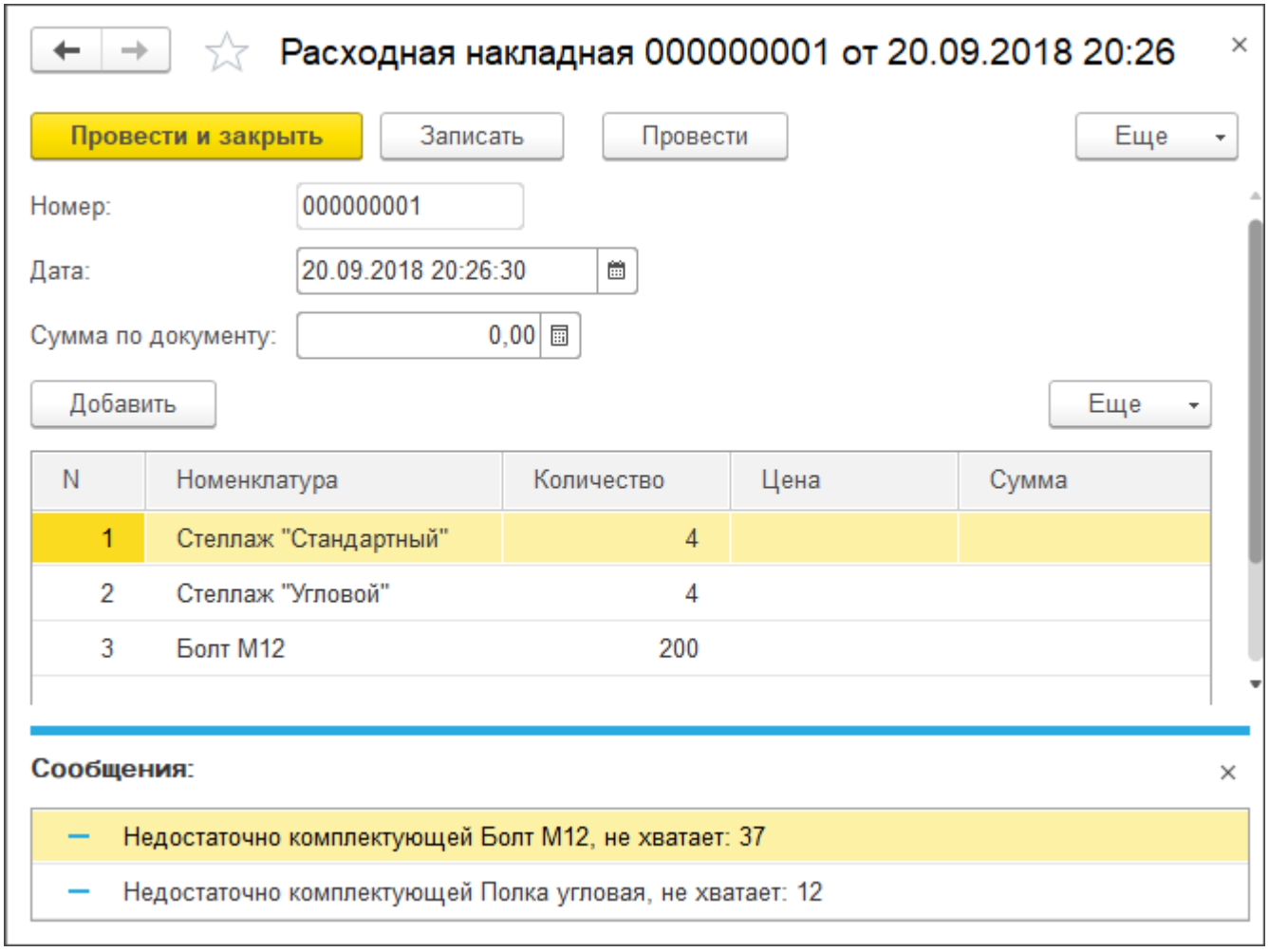


Рисунок 2 – Ошибка при проведении документа «Расходная накладная»

Видим, что документ не провелся и выдал сообщение о нехватке комплектующих.

2. Добавим на остатки деталей:

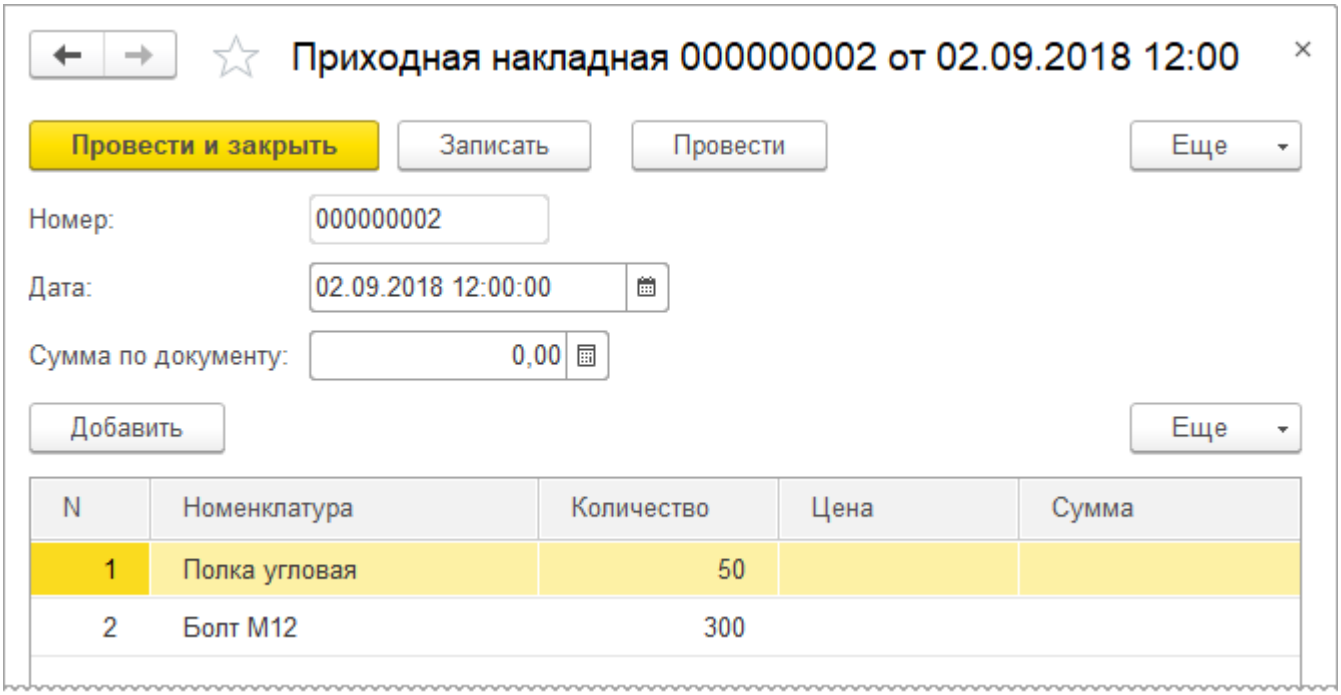
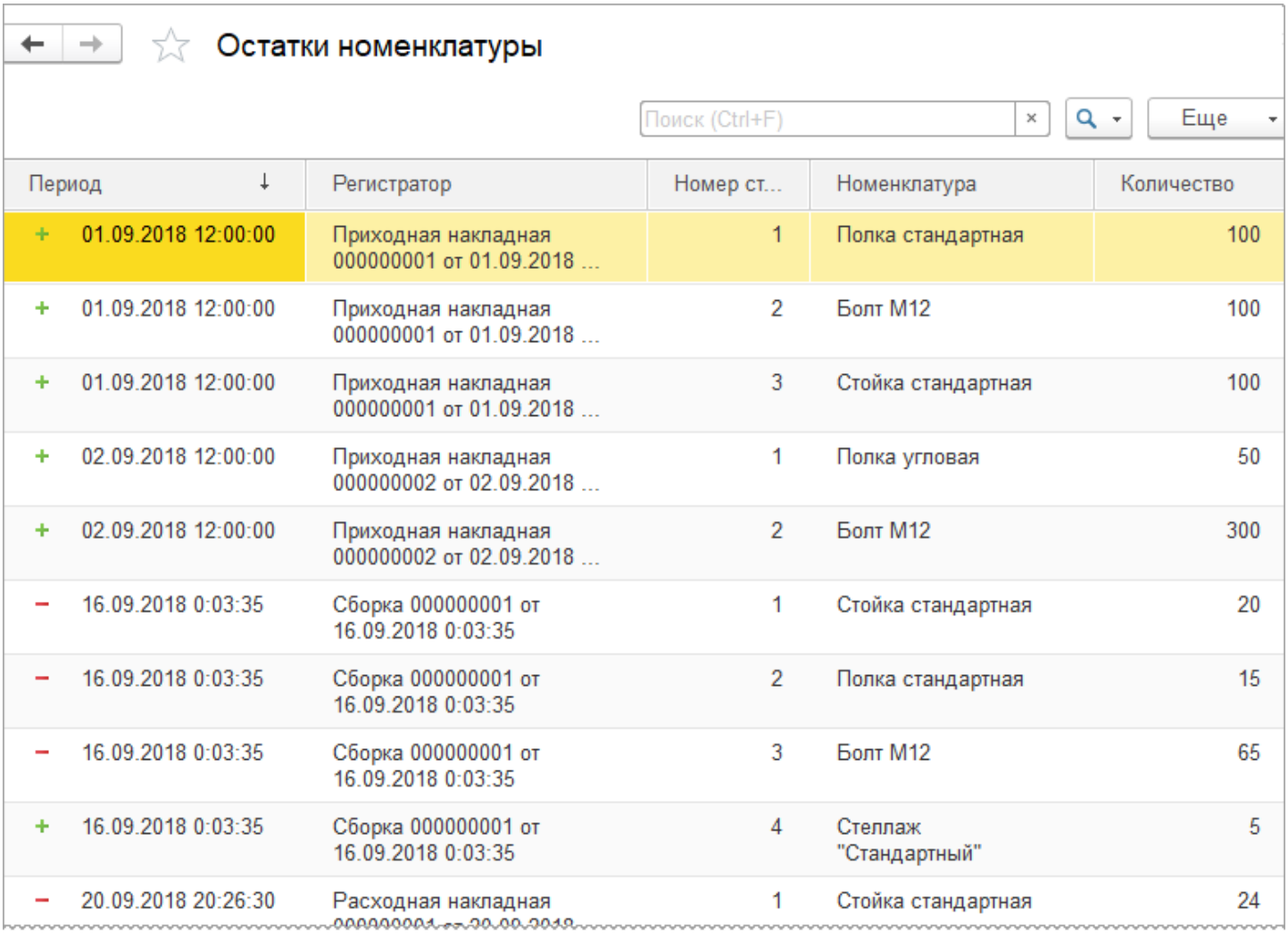


Рисунок 3 – Пример документа «Приходная накладная» в режиме «1С:Предприятие»

Проведем «Расходную накладную» еще раз. Итогом проведения будет следующий набор записей в регистр накопления «Остатки номенклатуры»:



Остатки номенклатуры

Поиск (Ctrl+F) | Еще

Период	Регистратор	Номер ст...	Номенклатура	Количество
+ 01.09.2018 12:00:00	Приходная накладная 000000001 от 01.09.2018 ...	1	Полка стандартная	100
+ 01.09.2018 12:00:00	Приходная накладная 000000001 от 01.09.2018 ...	2	Болт М12	100
+ 01.09.2018 12:00:00	Приходная накладная 000000001 от 01.09.2018 ...	3	Стойка стандартная	100
+ 02.09.2018 12:00:00	Приходная накладная 000000002 от 02.09.2018 ...	1	Полка угловая	50
+ 02.09.2018 12:00:00	Приходная накладная 000000002 от 02.09.2018 ...	2	Болт М12	300
− 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	1	Стойка стандартная	20
− 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	2	Полка стандартная	15
− 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	3	Болт М12	65
+ 16.09.2018 0:03:35	Сборка 000000001 от 16.09.2018 0:03:35	4	Стеллаж "Стандартный"	5
− 20.09.2018 20:26:30	Расходная накладная	1	Стойка стандартная	24

Рисунок 4 – Результаты проведения документов в режиме «1С:Предприятие»

Задача решена.

Подведение итогов

В этом разделе мы разобрали особенности операции комплектации «на лету» – сборки комплектов в момент проведения документа продажи.

На практическом примере реализовали схему проведения документа «Расходная накладная» с учетом следующих требований:

- Определили, каких комплектов из указанных в табличной части не хватает на остатках
- Рассчитали, сколько комплектующих нужно, чтобы собрать недостающее количество комплектов

- Сформировали движения по сборке комплектов, по реализации комплектующих и комплектов, указанных в табличной части.

На этом изучение общих этапов задач по комплектации номенклатуры, встречающихся как в оперативном, так и в бухгалтерском учете, закончено. Перейдем к рассмотрению следующей темы: взаиморасчеты.